



Entwicklerhandbuch

Amazon Lookout für Vision



Amazon Lookout für Vision: Entwicklerhandbuch

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

.....	ix
Was ist Amazon Lookout for Vision?	1
Wichtigste Vorteile	1
Sind Sie zum ersten Mal ein Endbenutzer von Amazon Lookout for Vision?	1
Amazon Lookout for Vision einrichten	3
Schritt 1: Erstellen Sie ein Konto AWS	3
Melden Sie sich an für ein AWS-Konto	3
Erstellen eines Benutzers mit Administratorzugriff	4
Schritt 2: Berechtigungen einrichten	5
Einrichten des Konsolenzugriffs mit verwalteten Richtlinien AWS	6
Amazon S3 S3-Bucket-Berechtigungen einrichten	6
Zuweisen von Berechtigungen	8
Schritt 3: Erstellen Sie den Konsolen-Bucket.	8
Erstellen des Konsolen-Buckets mit der Amazon Lookout for Vision Vision-Konsole	9
Den Konsolen-Bucket mit Amazon S3 erstellen	10
Bucket-Einstellungen für die Konsole	11
Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs	12
Installieren Sie das AWS SDKS	12
Erteilen programmgesteuerten Zugriffs	12
Einrichten von SDK-Berechtigungen	16
Rufen Sie einen Amazon Lookout for Vision Vision-Betrieb an	21
Schritt 5: (Optional) Verwenden Sie Ihren eigenen AWS KMS Schlüssel	25
Amazon Lookout for Vision verstehen	27
Wählen Sie Ihren Modelltyp	28
Modell zur Bildklassifizierung	28
Modell der Bildsegmentierung	29
Erstellen Sie Ihr Modell	31
Erstellen eines Projekts	31
Erstellen eines Datensatzes	31
Trainieren Ihres Modells	33
Bewerten Ihres Modells	33
Verwenden Sie Ihr Modell	34
Verwenden Sie Ihr Modell auf einem Edge-Gerät	34
Verwenden Sie Ihr Dashboard	35

Erste Schritte	36
Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch	38
Schritt 2: Erstellen Sie das Modell	39
Schritt 3: Starten Sie das Modell	47
Schritt 4: Analysieren Sie ein Bild	49
Schritt 5: Stoppen Sie das Modell	54
Nächste Schritte	56
Erstellen Sie Ihr Modell	57
Erstellen Sie Ihr Projekt	57
Ein Projekt erstellen (Konsole)	58
Erstellen eines Projekts (SDK)	58
Erstellen Sie Ihren Datensatz	61
Bilder für einen Datensatz vorbereiten	61
Den Datensatz erstellen	63
Lokaler Computer	64
Amazon-S3-Bucket	66
Manifestdatei	69
Labeling von Bildern	98
Auswahl des Modelltyps	98
Bilder klassifizieren (Konsole)	99
Bilder segmentieren (Konsole)	100
Trainieren Ihres Modells	104
Ein Modell (Konsole) trainieren	105
Ein Modell trainieren (SDK)	106
Problembehandlung beim Modelltraining	113
Die Farben der Beschriftungen für Anomalien stimmen nicht mit der Farbe der Anomalien im Maskenbild überein	113
Maskenbilder liegen nicht im PNG-Format vor	115
Segmentierungs- oder Klassifizierungsbezeichnungen sind ungenau oder fehlen	115
Verbessern Ihres Modells	117
Schritt 1: Bewerten Sie die Leistung Ihres Modells	117
Metriken zur Bildklassifizierung	117
Modellmetriken zur Bildsegmentierung	118
Genauigkeit	118
Wiedererkennung	119
F1-Punktzahl	120

Durchschnittliche Schnittmenge über Union (IoU)	120
Ergebnisse der Tests	121
Schritt 2: Verbessern Sie Ihr Modell	122
Anzeigen von Leistungsmetriken	123
Leistungsmetriken anzeigen (Konsole)	123
Leistungskennzahlen (SDK) anzeigen	125
Verifizieren Sie Ihr Modell	129
Ausführen einer Aufgabe zur Erkennung von Testversionen	130
Überprüfung der Ergebnisse der Studienerkennung	131
Korrigieren von Segmentierungsbeschriftungen mit dem Annotationstool	133
Führen Sie Ihr Modell aus	135
Inferenzeinheiten	135
Verwaltung des Durchsatzes mit Inferenzeinheiten	136
Availability Zones	138
Starten Sie Ihr Modell	138
Starten Sie Ihr Modell (Konsole)	139
Starten Sie Ihr Modell (SDK)	140
Ihr Modell beenden	145
Stoppen Sie Ihr Modell (Konsole)	146
Ihr Modell beenden (SDK)	147
Erkennung von Anomalien in einem Bild	152
Anrufen DetectAnomalies	152
Die Antwort von verstehen DetectAnomalies	156
Klassifikationsmodell	156
Segmentierungsmodell	157
Feststellen, ob ein Bild anomal ist	159
Klassifizierung	159
Segmentierung	161
Informationen zur Klassifizierung und Segmentierung werden angezeigt	166
Auffinden von Anomalien mit einer Funktion AWS Lambda	181
Schritt 1: Erstellen Sie eine AWS Lambda Funktion (Konsole)	181
Schritt 2: (Optional) Erstellen einer Ebene (Konsole)	184
Schritt 3: Hinzufügen von Python-Code (Konsole)	185
Schritt 4: Testen Ihrer Lambda-Funktion	189
Verwenden Sie Ihr Modell auf einem Edge-Gerät	194
Bereitstellen eines Modells auf einem Kerngerät	196

Kernanforderungen an Geräte	197
Getestete Geräte, Chiparchitekturen und Betriebssysteme	197
Speicher und Speicher des Kerngeräts	198
Erforderliche Software	199
Ihr Kerngerät einrichten	200
Ihr Core-Gerät einrichten	201
Verpacken Sie Ihr Modell	202
Einstellungen für das Package	203
Verpacken Sie Ihr Modell (Konsole)	205
Verpacken Sie Ihr Modell (SDK)	206
Informationen zu Paketierungsaufträgen für Modelle abrufen	211
Schreiben Sie Ihre Client-Anwendungskomponente	213
Einrichten Ihrer Umgebung	214
Verwenden eines Modells	215
Die Komponente der Client-Anwendung wird erstellt	220
Bereitstellen Ihrer Komponenten auf einem Gerät	226
IAM-Berechtigungen für die Bereitstellung von Komponenten	227
Bereitstellen Ihrer Komponenten (Konsole)	227
Bereitstellung der Komponenten (SDK)	229
Lookout for Vision Edge Agent API-Referenz	231
Erkennung von Anomalien mit einem Modell	231
Modellinformationen abrufen	231
Ein Modell ausführen	231
DetectAnomalies	232
DescribeModel	238
ListModels	240
StartModel	241
StopModel	243
ModelStatus	245
Verwenden des Dashboards	246
Verwaltung Ihrer Ressourcen	249
Ihre Projekte anzeigen	250
Ihre Projekte anzeigen (Konsole)	250
Ihre Projekte anzeigen (SDK)	250
Löschen eines Projekts	253
Löschen eines Projekts (Konsole)	253

Löschen eines Projekts (SDK)	254
Ihre Datensätze anzeigen	256
Die Datensätze in einem Projekt anzeigen (Konsole)	256
Die Datensätze in einem Projekt (SDK) anzeigen	257
Hinzufügen von Bildern zu Ihrem Datensatz	260
Weitere Bilder hinzufügen	260
Weitere Bilder hinzufügen (SDK)	261
Bilder aus Ihrem Datensatz entfernen	266
Bilder aus einem Datensatz entfernen (Konsole)	267
Bilder aus einem Datensatz entfernen (SDK)	268
Löschen eines Datensatzes	269
Löschen eines Datensatzes (Konsole)	256
Löschen eines Datensatzes (SDK)	270
Exportieren von Datensätzen aus einem Projekt (SDK)	272
Deine Modelle ansehen	281
Ihre Modelle anzeigen (Konsole)	281
Ihre Modelle anzeigen (SDK)	282
Löschen eines Modells	284
Löschen eines Modells (Konsole)	285
Löschen eines Modells (SDK)	285
Modelle taggen	288
Taggen von Modellen (Konsole)	289
Modelle kennzeichnen (SDK)	291
Ihre Aufgaben zur Erkennung von Studien anzeigen	293
Ihre Aufgaben zur Erkennung von Testversionen anzeigen (Konsole)	293
Beispielcode und Datensätze	294
Beispiel-Code	294
Beispieldatensätze	294
Datensätze zur Bildsegmentierung	295
Datensatz zur Bildklassifizierung	295
Sicherheit	298
Datenschutz	298
Datenverschlüsselung	300
Richtlinie für den Datenverkehr zwischen Netzwerken	301
Identity and Access Management	301
Zielgruppe	302

Authentifizierung mit Identitäten	302
Verwalten des Zugriffs mit Richtlinien	306
So funktioniert Amazon Lookout for Vision mit IAM	309
Beispiele für identitätsbasierte Richtlinien	317
Von AWS verwaltete Richtlinien	320
Fehlerbehebung	332
Compliance-Validierung	334
Ausfallsicherheit	335
Sicherheit der Infrastruktur	335
Überwachen	337
Überwachung mit CloudWatch	338
CloudTrail protokolliert	340
Halten Sie Ausschau nach Informationen zum Thema Vision in CloudTrail	341
Grundlegendes zu Lookout for Vision Vision-Protokolldateieinträgen	342
AWS CloudFormation Ressourcen	345
Lookout for Vision und Vorlagen AWS CloudFormation	345
Erfahren Sie mehr über AWS CloudFormation	345
AWS PrivateLink	347
Überlegungen zu VPC-Endpunkten von Lookout for Vision	347
Erstellen eines VPC-Schnittstellen-Endpunkts für Lookout for Vision	347
Erstellen einer VPC-Endpunktrichtlinie für Lookout for Vision	348
Kontingente	350
Modellkontingente	350
Dokumentverlauf	353
AWS Glossar	359

Hinweis zum Ende des Supports: Am 31. Oktober 2025 AWS wird der Support für Amazon Lookout for Vision eingestellt. Nach dem 31. Oktober 2025 können Sie nicht mehr auf die Lookout for Vision Vision-Konsole oder die Lookout for Vision Vision-Ressourcen zugreifen. Weitere Informationen finden Sie in diesem [Blogbeitrag](#).

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Was ist Amazon Lookout for Vision?

Sie können Amazon Lookout for Vision verwenden, um Sehfehler an Industrieprodukten präzise und maßstabsgetreu zu finden. Es verwendet Computer Vision, um fehlende Komponenten in einem Industrieprodukt, Schäden an Fahrzeugen oder Strukturen, Unregelmäßigkeiten in Produktionslinien und sogar winzige Defekte an Siliziumwafern — oder jedem anderen physischen Gegenstand, bei dem es auf Qualität ankommt, wie z. B. einen fehlenden Kondensator auf Leiterplatten, zu identifizieren.

Wichtigste Vorteile

Amazon Lookout for Vision bietet die folgenden Vorteile:

- Prozesse schnell und effizient verbessern — Sie können Amazon Lookout for Vision verwenden, um computergestützte Inspektionen in industriellen Prozessen schnell, effizient und in großem Umfang zu implementieren. Sie können nur 30 gute Basisbilder bereitstellen, und Lookout for Vision kann automatisch ein benutzerdefiniertes ML-Modell für die Fehlererkennung erstellen. Anschließend können Sie Bilder von IP-Kameras stapelweise oder in Echtzeit verarbeiten, um Anomalien wie Dellen, Risse und Kratzer schnell und genau zu identifizieren.
- Steigern Sie schnell die Produktionsqualität — Mit Amazon Lookout for Vision können Sie Fehler in Produktionsprozessen in Echtzeit reduzieren. Es identifiziert visuelle Anomalien und meldet sie in einem Dashboard, sodass Sie schnell Maßnahmen ergreifen können, um das Auftreten weiterer Fehler zu verhindern und so die Produktionsqualität zu erhöhen und die Kosten zu senken.
- Senken Sie die Betriebskosten — Amazon Lookout for Vision meldet Trends in Ihren visuellen Inspektionsdaten, z. B. bei der Identifizierung von Prozessen mit der höchsten Fehlerrate oder der Kennzeichnung kürzlich aufgetretener Abweichungen von Fehlern. Anhand dieser Informationen können Sie entscheiden, ob Sie Wartungsarbeiten an der Prozesslinie einplanen oder die Produktion auf eine andere Maschine umleiten möchten, bevor kostspielige, ungeplante Ausfallzeiten auftreten.

Sind Sie zum ersten Mal ein Endbenutzer von Amazon Lookout for Vision?

Wenn Sie Amazon Lookout for Vision zum ersten Mal verwenden, empfehlen wir Ihnen, die folgenden Abschnitte der Reihe nach zu lesen:

1. [Amazon Lookout for Vision einrichten](#) – In diesem Abschnitt legen Sie die Details Ihres Kontos fest.
2. [Erste Schritte mit Amazon Lookout for Vision](#)— In diesem Abschnitt erfahren Sie, wie Sie Ihr erstes Amazon Lookout for Vision Vision-Modell erstellen.

Amazon Lookout for Vision einrichten

In diesem Abschnitt registrieren Sie sich für ein AWS-Konto und richten Amazon Lookout for Vision ein.

Informationen zu den AWS Regionen, die Amazon Lookout for Vision unterstützen, finden Sie unter [Amazon Lookout for Vision Endpoints](#) and Quotas.

Themen

- [Schritt 1: Erstellen Sie ein Konto AWS](#)
- [Schritt 2: Berechtigungen einrichten](#)
- [Schritt 3: Erstellen Sie den Konsolen-Bucket.](#)
- [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#)
- [Schritt 5: \(Optional\) Verwenden Ihres eigenen AWS Key Management Service Service-Schlüssels](#)

Schritt 1: Erstellen Sie ein Konto AWS

In diesem Schritt registrieren Sie sich für ein AWS Konto und erstellen einen Administratorbenutzer.

Themen

- [Melden Sie sich an für ein AWS-Konto](#)
- [Erstellen eines Benutzers mit Administratorzugriff](#)

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie <https://portal.aws.amazon.com/billing/die-Anmeldung>.
2. Folgen Sie den Online-Anweisungen.

Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontos wird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um [Aufgaben auszuführen, die Root-Benutzerzugriff erfordern](#).

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu <https://aws.amazon.com/> gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich [AWS Management Console](#) als Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter [Anmelden als Root-Benutzer](#) im AWS-Anmeldung Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

Anweisungen finden Sie unter [Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

Anweisungen finden Sie unter [Aktivieren AWS IAM Identity Center](#) im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter [Benutzerzugriff mit der Standardeinstellung konfigurieren](#).AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie [im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal](#).

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

Anweisungen hierzu finden Sie unter [Berechtigungssatz erstellen](#) im AWS IAM Identity Center Benutzerhandbuch.

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

Eine genaue Anleitung finden Sie unter [Gruppen hinzufügen](#) im AWS IAM Identity Center Benutzerhandbuch.

Schritt 2: Berechtigungen einrichten

Um Amazon Lookout for Vision verwenden zu können, benötigen Sie Zugriffsberechtigungen für die Lookout for Vision Vision-Konsole, AWS SDK-Operationen und den Amazon S3 S3-Bucket, den Sie für das Modelltraining verwenden.

 Note

Wenn Sie nur AWS SDK-Operationen verwenden, können Sie Richtlinien verwenden, die auf SDK-Operationen beschränkt sind. AWS Weitere Informationen finden Sie unter [Einrichten von SDK-Berechtigungen](#).

Themen

- [Einrichten des Konsolenzugriffs mit verwalteten Richtlinien AWS](#)
- [Amazon S3 S3-Bucket-Berechtigungen einrichten](#)
- [Zuweisen von Berechtigungen](#)

Einrichten des Konsolenzugriffs mit verwalteten Richtlinien AWS

Verwenden Sie die folgenden AWS verwalteten Richtlinien, um die entsprechenden Zugriffsberechtigungen für die Amazon Lookout for Vision Vision-Konsole und den SDK-Betrieb anzuwenden.

- [AmazonLookoutVisionConsoleFullAccess](#)— ermöglicht vollen Zugriff auf die Amazon Lookout for Vision Vision-Konsole und die SDK-Operationen. Sie benötigen AmazonLookoutVisionConsoleFullAccess Berechtigungen, um den Konsolen-Bucket zu erstellen. Weitere Informationen finden Sie unter [Schritt 3: Erstellen Sie den Konsolen-Bucket](#).
- [AmazonLookoutVisionConsoleReadOnlyAccess](#)— ermöglicht schreibgeschützten Zugriff auf die Amazon Lookout for Vision Vision-Konsole und SDK-Operationen.

Informationen zum Zuweisen von Berechtigungen finden Sie unter [Zuweisen von Berechtigungen](#).

Informationen zu AWS verwalteten Richtlinien finden Sie unter Von [AWS verwaltete Richtlinien](#).

Amazon S3 S3-Bucket-Berechtigungen einrichten

Amazon Lookout for Vision verwendet einen Amazon S3 S3-Bucket zum Speichern der folgenden Dateien:

- Datensatz-Bilder — Bilder, die zum Trainieren eines Modells verwendet werden. Weitere Informationen finden Sie unter [Erstellen Sie Ihren Datensatz](#).

- Manifestdateien im Amazon SageMaker AI Ground Truth Format. Zum Beispiel die Ausgabe der Manifestdatei aus einem SageMaker GroundTruth AI-Job. Weitere Informationen finden Sie unter [Erstellen eines Datensatzes mithilfe einer Amazon SageMaker AI Ground Truth Manifestdatei](#).
- Die Ausgabe des Modelltrainings.

Wenn Sie die Konsole verwenden, erstellt Lookout for Vision einen Amazon S3 S3-Bucket (Konsolen-Bucket) zur Verwaltung Ihrer Projekte. Die `LookoutVisionConsoleReadOnlyAccess` und `LookoutVisionConsoleFullAccess` verwalteten Richtlinien beinhalten Amazon S3 S3-Zugriffsberechtigungen für den Konsolen-Bucket.

Sie können den Konsolen-Bucket verwenden, um Datensatzbilder und Manifestdateien im SageMaker AI Ground Truth Format zu speichern. Alternativ können Sie einen anderen Amazon S3 S3-Bucket verwenden. Der Bucket muss Ihrem AWS-Konto gehören und sich in der AWS Region befinden, in der Sie Lookout for Vision verwenden.

Um einen anderen Bucket zu verwenden, fügen Sie dem gewünschten Benutzer oder der gewünschten Gruppe die folgende Richtlinie hinzu. `amzn-s3-demo-bucket` Ersetzen Sie es durch den Namen des gewünschten Buckets. Informationen zum Hinzufügen von IAM-Richtlinien finden Sie unter [IAM-Richtlinien erstellen](#).

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

Informationen zum Zuweisen von Berechtigungen finden Sie unter [Zuweisen von Berechtigungen](#).

Zuweisen von Berechtigungen

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:
 - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
 - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Schritt 3: Erstellen Sie den Konsolen-Bucket.

Um die Amazon Lookout for Vision Vision-Konsole verwenden zu können, benötigen Sie einen Amazon S3 S3-Bucket, der als Konsolen-Bucket bezeichnet wird. Der Konsolen-Bucket speichert Folgendes:

- Bilder, die Sie mit der Konsole in einen Datensatz [hochladen](#).
- Trainingsergebnisse für das [Modelltraining](#), das Sie mit der Konsole beginnen.

- Ergebnisse der [Versuchserkennung](#).
- Temporäre Manifestdateien, die von der Konsole erstellt werden, wenn Sie die Konsole verwenden, um einen Datensatz zu erstellen, indem Bilder in einem S3-Bucket [automatisch beschriftet](#) werden. Die Konsole löscht die Manifestdateien nicht.

Wenn Sie die Amazon Lookout for Vision-Konsole zum ersten Mal in einer neuen AWS Region öffnen, erstellt Lookout for Vision den Konsolen-Bucket in Ihrem Namen. Notieren Sie sich den Namen des Konsolen-Buckets, da Sie den Bucket-Namen möglicherweise bei AWS SDK-Vorgängen oder Konsolenaufgaben wie der Erstellung eines Datensatzes verwenden müssen.

Alternativ können Sie den Konsolen-Bucket mithilfe von Amazon S3 erstellen. Verwenden Sie diesen Ansatz, wenn die Amazon S3 S3-Bucket-Richtlinien es nicht zulassen, dass die Amazon Lookout for Vision Vision-Konsole den Konsolen-Bucket erfolgreich erstellt. Zum Beispiel eine Richtlinie, die die automatische Erstellung eines Amazon S3 S3-Buckets verbietet.

Note

Wenn Sie nur das AWS SDK und nicht die Lookout for Vision Vision-Konsole verwenden, müssen Sie den Konsolen-Bucket nicht erstellen. Sie können einen anderen S3-Bucket mit einem Namen Ihrer Wahl verwenden.

Das Format des Konsolen-Bucket-Namens ist `lookoutvision - <region> -<random value>`. Der Zufallswert stellt sicher, dass es nicht zu einer Kollision zwischen Bucket-Namen kommt.

Themen

- [Erstellen des Konsolen-Buckets mit der Amazon Lookout for Vision Vision-Konsole](#)
- [Den Konsolen-Bucket mit Amazon S3 erstellen](#)
- [Bucket-Einstellungen für die Konsole](#)

Erstellen des Konsolen-Buckets mit der Amazon Lookout for Vision Vision-Konsole

Gehen Sie wie folgt vor, um den Konsolen-Bucket für eine AWS Region mit der Amazon Lookout for Vision Vision-Konsole zu erstellen. Informationen zu den S3-Bucket-Einstellungen, die wir aktivieren, finden Sie unter [Bucket-Einstellungen für die Konsole](#).

So erstellen Sie den Konsolen-Bucket mithilfe der Amazon Lookout for Vision Vision-Konsole

1. Stellen Sie sicher, dass der Benutzer oder die Gruppe, die Sie verwenden, über die entsprechenden `AmazonLookoutVisionConsoleFullAccess` Berechtigungen verfügt. Weitere Informationen finden Sie unter [Schritt 2: Berechtigungen einrichten](#).
2. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
3. Wählen Sie in der Navigationsleiste die Option Region auswählen aus. Wählen Sie dann die AWS Region aus, für die Sie den Konsolen-Bucket erstellen möchten.
4. Wählen Sie Erste Schritte.
5. Wenn Sie die Konsole in der aktuellen AWS-Region zum ersten Mal öffnen, gehen Sie im Dialogfeld Erste Einrichtung wie folgt vor:
 - a. Kopieren Sie den Namen des Amazon-S3-Buckets, der angezeigt wird. Sie benötigen diese Informationen später wieder.
 - b. Wählen Sie S3-Bucket erstellen, damit Amazon Lookout for Vision den Konsolen-Bucket in Ihrem Namen erstellen kann.

Das Dialogfeld „Erstkonfiguration“ wird nicht angezeigt, wenn der Konsolen-Bucket für die aktuelle AWS Region bereits vorhanden ist.

6. Schließen Sie das Browserfenster.

Den Konsolen-Bucket mit Amazon S3 erstellen

Sie können Amazon S3 verwenden, um den Konsolen-Bucket zu erstellen. Sie müssen den Bucket mit aktivierter [Amazon S3 S3-Versionierung](#) erstellen. Wir empfehlen, eine [Amazon S3 S3-Lebenszykluskonfiguration](#) zu verwenden, um nicht aktuelle (frühere) Versionen eines Objekts zu entfernen und unvollständige mehrteilige Uploads zu löschen. Wir empfehlen keine Lebenszykluskonfiguration, die aktuelle Versionen eines Objekts löscht. Informationen zu den S3-Bucket-Einstellungen, die wir für Konsolen-Buckets aktivieren, die Sie mit der Amazon Lookout for Vision Vision-Konsole erstellen, finden Sie unter [Bucket-Einstellungen für die Konsole](#)

1. Entscheiden Sie sich für die AWS Region, in der Sie einen Konsolen-Bucket erstellen möchten. Informationen zu den unterstützten Regionen finden Sie unter [Amazon Lookout for Vision Endpoints and Quotas](#).

2. Erstellen Sie einen Bucket mithilfe der Anweisungen der S3-Konsole unter Bucket [erstellen](#). Gehen Sie wie folgt vor:
 - a. Geben Sie für Schritt 3 einen Bucket-Namen an, dem Folgendes vorangestellt wird. `lookoutvision-region-your-identifier` Wechseln Sie *region* zu dem Regionalcode, den Sie im vorherigen Schritt ausgewählt haben. Wechseln *your-identifier* Sie zu einer eindeutigen Kennung Ihrer Wahl.
 - b. Wählen Sie für Schritt 4 die AWS Region aus, die Sie verwenden möchten.
3. Aktivieren Sie die Versionierung für den Bucket, indem Sie den Anweisungen der S3-Konsole unter [Versionierung für Buckets aktivieren](#) folgen.
4. (Optional) Geben Sie eine Lebenszykluskonfiguration für den Bucket an, indem Sie den Anweisungen der S3-Konsole unter [Lebenszykluskonfiguration für einen Bucket einrichten](#) folgen. Gehen Sie wie folgt vor, um veraltete (frühere) Versionen eines Objekts zu entfernen und unvollständige mehrteilige Uploads zu löschen. Sie müssen die Schritte 6, 8, 9, 10 nicht ausführen.
 - a. Wählen Sie für Schritt 5 die Option Auf alle Objekte im Bucket anwenden aus.
 - b. Wählen Sie für Schritt 7 die Optionen Nicht aktuelle Versionen von Objekten dauerhaft löschen und Abgelaufene Objekte löschen, Markierungen löschen oder unvollständige mehrteilige Uploads aus.
 - c. Geben Sie für Schritt 11 die Anzahl der Tage ein, die gewartet werden sollen, bevor veraltete Versionen eines Objekts gelöscht werden.
 - d. Geben Sie für Schritt 12 ein, wie viele Tage gewartet werden soll, bevor unvollständige mehrteilige Uploads gelöscht werden.

Bucket-Einstellungen für die Konsole

Wenn Sie den Konsolen-Bucket mit der Amazon Lookout for Vision Vision-Konsole erstellen, aktivieren wir die folgenden Einstellungen im Konsolen-Bucket.

- [Versionierung](#) von Objekten im Konsolen-Bucket.
- [Serverseitige Verschlüsselung](#) von Objekten im Konsolen-Bucket.
- [Eine Lebenszykluskonfiguration](#) für das Löschen nicht aktueller Objekte (30 Tage) und unvollständiger mehrteiliger Uploads (3 Tage).
- [Sperrt den öffentlichen Zugriff auf den Konsolen-Bucket](#).

Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs

Die folgenden Schritte zeigen Ihnen, wie Sie AWS Command Line Interface (AWS CLI) und installieren AWS SDKs. Die Beispiele in dieser Dokumentation verwenden Python und Java AWS SDKs. AWS CLI

Themen

- [Installieren Sie das AWS SDKS](#)
- [Erteilen programmgesteuerten Zugriffs](#)
- [Einrichten von SDK-Berechtigungen](#)
- [Rufen Sie einen Amazon Lookout for Vision Vision-Betrieb an](#)

Installieren Sie das AWS SDKS

Befolgen Sie die Schritte zum Herunterladen und Konfigurieren der AWS SDKs.

Um das AWS CLI und das einzurichten AWS SDKs

- Laden Sie das und das herunter [AWS CLI](#) und installieren Sie AWS SDKs es, das Sie verwenden möchten. Dieses Handbuch enthält Beispiele für [Java](#) und [Python](#). AWS CLI Informationen zur Installation AWS SDKs finden Sie unter [Tools für Amazon Web Services](#).

Erteilen programmgesteuerten Zugriffs

Sie können die Codebeispiele AWS CLI und die Codebeispiele in diesem Handbuch auf Ihrem lokalen Computer oder in anderen AWS Umgebungen, z. B. einer Amazon Elastic Compute Cloud-Instance, ausführen. Um die Beispiele auszuführen, müssen Sie Zugriff auf die AWS SDK-Operationen gewähren, die in den Beispielen verwendet werden.

Themen

- [Ausführen von Code auf Ihrem lokalen Computer](#)
- [Code in AWS Umgebungen ausführen](#)

Ausführen von Code auf Ihrem lokalen Computer

Um Code auf einem lokalen Computer auszuführen, empfehlen wir, dass Sie kurzfristige Anmeldeinformationen verwenden, um einem Benutzer Zugriff auf AWS SDK-Operationen zu gewähren. Spezifische Informationen zum Ausführen von AWS CLI und zu Codebeispielen auf einem lokalen Computer finden Sie unter [Verwenden eines Profils auf Ihrem lokalen Computer](#).

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb des AWS Management Console interagieren möchten. Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. - Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. - Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldinformationen im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Verwenden eines Profils auf Ihrem lokalen Computer

Sie können die Codebeispiele AWS CLI und die Codebeispiele in diesem Handbuch mit den kurzfristigen Anmeldeinformationen ausführen, die Sie in [Ausführen von Code auf Ihrem lokalen Computer](#) erstellen. Um die Anmeldeinformationen und andere Einstellungsinformationen abzurufen, verwenden die Beispiele ein Profil mit dem Namen `lookoutvision-access`. Zum Beispiel:

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

Der Benutzer, den das Profil repräsentiert, muss berechtigt sein, die Lookout for Vision SDK-Operationen und andere AWS SDK-Operationen, die in den Beispielen benötigt werden, aufzurufen. Weitere Informationen finden Sie unter [Einrichten von SDK-Berechtigungen](#). Informationen zum Zuweisen von Berechtigungen finden Sie unter [Zuweisen von Berechtigungen](#).

Um ein Profil zu erstellen, das mit den Codebeispielen AWS CLI und -Codebeispielen funktioniert, wählen Sie eine der folgenden Optionen. Stellen Sie sicher, dass der Name des von Ihnen erstellten Profils `lookoutvision-access` lautet.

- Von IAM verwaltete Benutzer — Folgen Sie den Anweisungen unter [Zu einer IAM-Rolle wechseln \(AWS CLI\)](#).
- Personalidentität (Benutzer verwaltet von AWS IAM Identity Center) — Folgen Sie den Anweisungen unter [Konfiguration der zu verwendenden AWS-CLI AWS IAM Identity Center](#). Für die Codebeispiele empfehlen wir die Verwendung einer integrierten Entwicklungsumgebung (IDE), die das AWS-Toolkit unterstützt und die Authentifizierung über das IAM Identity Center ermöglicht. Die Java-Beispiele finden Sie unter [Mit Java entwickeln](#). Die Python-Beispiele finden Sie unter [Mit Python entwickeln](#). Weitere Informationen finden Sie unter [Anmeldeinformationen für das IAM Identity Center](#).

Note

Sie können Code verwenden, um kurzfristige Anmeldeinformationen zu erhalten. Weitere Informationen finden Sie unter [Wechseln zu einer IAM-Rolle \(AWS API\)](#). Rufen Sie für IAM Identity Center die kurzfristigen Anmeldeinformationen für eine Rolle ab, indem Sie den Anweisungen unter [Abrufen von IAM-Rollenanmeldeinformationen für den CLI-Zugriff](#) folgen.

Code in AWS Umgebungen ausführen

Sie sollten keine Benutzeranmeldedaten verwenden, um AWS SDK-Aufrufe in AWS Umgebungen zu signieren, wie z. B. Produktionscode, der in einer AWS Lambda Funktion ausgeführt wird. Stattdessen konfigurieren Sie eine Rolle, die die Berechtigungen definiert, die Ihr Code benötigt. Anschließend weisen Sie die Rolle der Umgebung zu, in der Ihr Code ausgeführt wird. Wie Sie die

Rolle zuordnen und temporäre Anmeldeinformationen verfügbar machen, hängt von der Umgebung ab, in der Ihr Code ausgeführt wird:

- **AWS Lambda function** — Verwenden Sie die temporären Anmeldeinformationen, die Lambda Ihrer Funktion automatisch zur Verfügung stellt, wenn sie die Ausführungsrolle der Lambda-Funktion übernimmt. Die Anmeldeinformationen sind in den Lambda-Umgebungsvariablen verfügbar. Sie müssen kein Profil angeben. Weitere Informationen finden Sie unter [Lambda-Ausführungsrolle](#).
- **Amazon EC2** — Verwenden Sie den Anbieter für Anmeldeinformationen für EC2 Amazon-Instance-Metadaten. Der Anbieter generiert und aktualisiert automatisch Anmeldeinformationen für Sie mithilfe des EC2 Amazon-Instance-Profiles, das Sie der EC2 Amazon-Instance zuordnen. Weitere Informationen finden Sie unter [Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf EC2 Amazon-Instances ausgeführt werden](#).
- **Amazon Elastic Container Service** — Verwenden Sie den Anbieter für Container-Anmeldeinformationen. Amazon ECS sendet und aktualisiert Anmeldeinformationen an einen Metadaten-Endpunkt. Eine von Ihnen angegebene Aufgaben-IAM-Rolle bietet eine Strategie für die Verwaltung der Anmeldeinformationen, die Ihre Anwendung verwendet. Weitere Informationen finden Sie unter [Interagieren mit AWS-Services](#).
- **Greengrass-Core-Gerät** — Verwenden Sie X.509-Zertifikate, um mithilfe von TLS-Protokollen für die gegenseitige Authentifizierung eine Verbindung zu AWS IoT Core herzustellen. Mit diesen Zertifikaten können Geräte ohne AWS-Anmeldeinformationen mit AWS IoT interagieren. Der Anbieter von AWS-IoT-Anmeldeinformationen authentifiziert Geräte mithilfe des X.509-Zertifikats und stellt AWS-Anmeldeinformationen in Form eines temporären Sicherheitstokens mit eingeschränkten Rechten aus. Weitere Informationen finden Sie unter [Interagieren mit AWS-Services](#).

Weitere Informationen zu Anbietern von Anmeldeinformationen finden Sie unter [Standardisierte Anmeldeinformationsanbieter](#).

Einrichten von SDK-Berechtigungen

Um Amazon Lookout for Vision SDK-Operationen verwenden zu können, benötigen Sie Zugriffsberechtigungen für die Lookout for Vision API und den Amazon S3 S3-Bucket, der für das Modelltraining verwendet wird.

Themen

- [Erteilen von SDK-Betriebsberechtigungen](#)

- [Erteilen von Amazon S3 S3-Bucket-Berechtigungen](#)
- [Zuweisen von Berechtigungen](#)

Erteilen von SDK-Betriebsberechtigungen

Es wird empfohlen, nur die Berechtigungen zu erteilen, die zum Ausführen einer Aufgabe erforderlich sind (geringste Berechtigungen). Um beispielsweise einen Anruf tätigen zu können [DetectAnomalies](#), benötigen Sie eine entsprechende Genehmigung. `lookoutvision:DetectAnomalies` Die Berechtigungen für einen Vorgang finden Sie in der [API-Referenz](#).

Wenn Sie gerade erst mit einer Anwendung beginnen, wissen Sie möglicherweise nicht, welche spezifischen Berechtigungen Sie benötigen, sodass Sie mit umfassenderen Berechtigungen beginnen können. AWS verwaltete Richtlinien stellen Berechtigungen bereit, die Ihnen bei den ersten Schritten helfen.

- [AmazonLookoutVisionFullAccess](#)— ermöglicht vollen Zugriff auf Amazon Lookout for Vision SDK-Operationen.
- [AmazonLookoutVisionReadOnlyAccess](#)— ermöglicht den Zugriff auf die schreibgeschützten SDK-Operationen.

Die verwalteten Richtlinien für die Konsole bieten auch Zugriffsberechtigungen für SDK-Operationen. Weitere Informationen finden Sie unter [Schritt 2: Berechtigungen einrichten](#).

Informationen zu AWS verwalteten Richtlinien finden Sie unter Von [AWS verwaltete Richtlinien](#).

Wenn Sie wissen, welche Berechtigungen Ihre Anwendung benötigt, können Sie die Berechtigungen weiter reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen zu den vom Kunden verwalteten Richtlinien finden Sie unter [Vom Kunden verwaltete Richtlinien](#).

Note

Für die Anweisungen für die ersten Schritte sind `s3:PutObject` Berechtigungen erforderlich. Weitere Informationen finden Sie unter [Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch](#).

Informationen zum Zuweisen von Berechtigungen finden Sie unter [Zuweisen von Berechtigungen](#).

Erteilen von Amazon S3 S3-Bucket-Berechtigungen

Um ein Modell zu trainieren, benötigen Sie einen Amazon S3 S3-Bucket mit entsprechenden Berechtigungen zum Speichern der Bilder, Manifestdateien und Trainingsausgaben. Der Bucket muss Ihrem AWS-Konto gehören und sich in der AWS-Region befinden, in der Sie Amazon Lookout for Vision verwenden.

Die nur vom SDK verwalteten Richtlinien

(`AmazonLookoutVisionFullAccess` und `AmazonLookoutVisionReadOnlyAccess`) beinhalten keine Amazon S3 S3-Bucket-Berechtigungen. Sie müssen die folgende Berechtigungsrichtlinie anwenden, um auf die von Ihnen verwendeten Buckets zuzugreifen, einschließlich vorhandener Konsolen-Buckets.

Die von der Konsole verwalteten Richtlinien

(`AmazonLookoutVisionConsoleFullAccess` und `AmazonLookoutVisionConsoleReadOnlyAccess`) beinhalten Zugriffsberechtigungen für den Konsolen-Bucket. Wenn Sie mit SDK-Vorgängen auf den Konsolen-Bucket zugreifen und über die Konsole verwaltete Richtlinienberechtigungen verfügen, müssen Sie die folgende Richtlinie nicht verwenden. Weitere Informationen finden Sie unter [Schritt 2: Berechtigungen einrichten](#).

Festlegung von Aufgabenberechtigungen

Entscheiden Sie anhand der folgenden Informationen, welche Berechtigungen für die Aufgaben erforderlich sind, die Sie ausführen möchten.

Einen Datensatz erstellen

Um einen Datensatz mit zu erstellen [CreateDataset](#), benötigen Sie die folgenden Berechtigungen.

- `s3:GetBucketLocation`— ermöglicht Lookout for Vision zu überprüfen, ob sich Ihr Bucket in derselben Region befindet, in der Sie Lookout for Vision verwenden.
- `s3:GetObject`— Ermöglicht den Zugriff auf die im Eingabeparameter angegebene Manifestdatei. `DatasetSource` Wenn Sie eine exakte S3-Objektversion der Manifestdatei angeben möchten, müssen Sie dies auch in `s3:GetObjectVersion` der Manifestdatei tun. Weitere Informationen finden Sie unter [Verwenden der Versionierung in S3-Buckets](#).

Erstellen eines Modells

Um ein Modell mit zu erstellen [CreateModel](#), benötigen Sie die folgenden Berechtigungen.

- `s3:GetBucketLocation`— ermöglicht Lookout for Vision zu überprüfen, ob sich Ihr Bucket in derselben Region befindet, in der Sie Lookout for Vision verwenden.
- `s3:GetObject`— ermöglicht den Zugriff auf die Bilder, die in den Trainings- und Testdatensätzen des Projekts angegeben sind.
- `s3:PutObject`— ermöglicht die Erlaubnis, Trainingsausgaben im angegebenen Bucket zu speichern. Sie geben den Speicherort des Ausgabe-Buckets im `OutputConfig` Parameter an. Optional können Sie die Berechtigungen nur auf Objektschlüssel beschränken, die im `Prefix` Feld des `S3Location` Eingabefeldes angegeben sind. Weitere Informationen finden Sie unter [OutputConfig](#).

Zugreifen auf Bilder, Manifestdateien und Trainingsausgaben

Amazon S3 S3-Bucket-Berechtigungen sind nicht erforderlich, um die Antworten von Amazon Lookout for Vision Vision-Vorgängen anzuzeigen. Sie benötigen eine `s3:GetObject` entsprechende Genehmigung, wenn Sie auf Bilder, Manifeste, Dateien und Trainingsdaten zugreifen möchten, auf die in den Betriebsantworten verwiesen wird. Wenn Sie auf ein versioniertes Amazon S3 S3-Objekt zugreifen, benötigen Sie eine `s3:GetObjectVersion` Genehmigung.

Amazon S3 S3-Bucket-Richtlinie einrichten

Sie können die folgende Richtlinie verwenden, um die Amazon S3 S3-Bucket-Berechtigungen anzugeben, die für die Erstellung eines Datensatzes (`CreateDataset`), die Erstellung eines Modells (`CreateModel`) und den Zugriff auf Bilder, Manifestdateien und Trainingsausgaben erforderlich sind. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Namen des Buckets, den Sie verwenden möchten.

Sie können die Richtlinie an Ihre Bedürfnisse anpassen. Weitere Informationen finden Sie unter [Festlegung von Aufgabenberechtigungen](#). Fügen Sie die Richtlinie dem gewünschten Benutzer hinzu. Weitere Informationen finden Sie unter [IAM-Richtlinien erstellen](#).

```
{
  "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"
      }
    }
  }
]
}

```

Informationen zum Zuweisen von Berechtigungen finden Sie unter [Zuweisen von Berechtigungen](#).

Zuweisen von Berechtigungen

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Rufen Sie einen Amazon Lookout for Vision Vision-Betrieb an

Führen Sie den folgenden Code aus, um zu bestätigen, dass Sie Aufrufe an die Amazon Lookout for Vision API tätigen können. Der Code listet die Projekte in Ihrem AWS Konto in der aktuellen AWS Region auf. Wenn Sie noch kein Projekt erstellt haben, ist die Antwort leer, bestätigt aber, dass Sie den ListProjects-Vorgang aufrufen können.

Im Allgemeinen erfordert das Aufrufen einer Beispielfunktion einen AWS SDK Lookout for Vision Vision-Client und alle anderen erforderlichen Parameter. Der AWS SDK Lookout for Vision Vision-Client ist in der Hauptfunktion deklariert.

Wenn der Code fehlschlägt, überprüfen Sie, ob der von Ihnen verwendete Benutzer über die richtigen Berechtigungen verfügt. Überprüfen Sie auch die AWS Region, die Sie verwenden, da Amazon Lookout for Vision nicht in allen AWS Regionen verfügbar ist.

Um einen Amazon Lookout for Vision Vision-Betrieb anzurufen

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um Ihre Projekte anzusehen.

CLI

Verwenden Sie den `list-projects`-Befehl, um die Projekte in Ihrem Konto aufzulisten.

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

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
# 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
/*
    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",
lvfError.getMessage()));
        System.exit(1);
    }

}

}
```

Schritt 5: (Optional) Verwenden Ihres eigenen AWS Key Management Service Service-Schlüssels

Sie können den AWS Key Management Service (KMS) verwenden, um die Verschlüsselung der Eingabebilder zu verwalten, die Sie in Amazon S3 S3-Buckets speichern.

Standardmäßig werden Ihre Bilder mit einem Schlüssel verschlüsselt, den AWS besitzt und verwaltet. Sie können sich auch dafür entscheiden, Ihren eigenen AWS Key Management Service (KMS) - Schlüssel zu verwenden. Weitere Informationen finden Sie unter [AWS Key Management Service-Konzepte](#).

Wenn Sie Ihren eigenen KMS-Schlüssel verwenden möchten, verwenden Sie die folgende Richtlinie, um den KMS-Schlüssel anzugeben. Wechseln Sie *kms_key_arn* zum ARN des KMS-Schlüssels (oder KMS-Alias-ARN), den Sie verwenden möchten. Geben Sie alternativ * an, dass Sie einen beliebigen KMS-Schlüssel verwenden möchten. Informationen zum Hinzufügen der Richtlinie zu einem Benutzer oder einer Rolle finden Sie unter [IAM-Richtlinien erstellen](#).

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

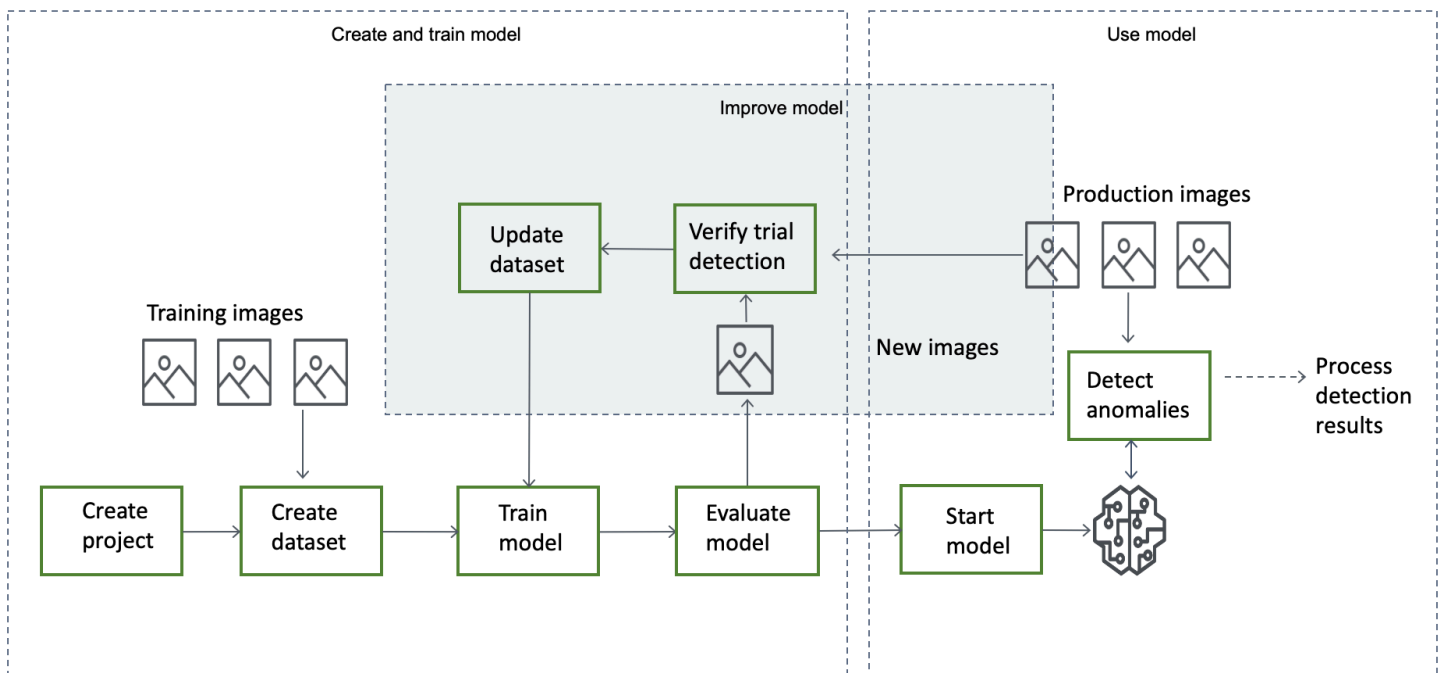
Amazon Lookout for Vision verstehen

Sie können Amazon Lookout for Vision verwenden, um Sehfehler an Industrieprodukten präzise und maßstabsgetreu zu finden, und zwar für Aufgaben wie:

- Erkennung beschädigter Teile — Punktuelle Schäden an der Oberflächenqualität, Farbe und Form eines Produkts während des Herstellungs- und Montageprozesses.
- Identifizierung fehlender Komponenten — Ermitteln Sie fehlende Komponenten anhand des Fehlens, Vorhandenseins oder der Platzierung von Objekten. Zum Beispiel ein fehlender Kondensator auf einer Leiterplatte.
- Aufdeckung von Prozessproblemen — Erkennen Sie Fehler anhand sich wiederholender Muster, wie z. B. wiederholte Kratzer an derselben Stelle auf einem Siliziumwafer.

Mit Lookout for Vision erstellen Sie ein Computer-Vision-Modell, das das Vorhandensein von Anomalien in einem Bild vorhersagt. Sie stellen die Bilder bereit, die Amazon Lookout for Vision zum Trainieren und Testen Ihres Modells verwendet. Amazon Lookout for Vision bietet Kennzahlen, mit denen Sie Ihr trainiertes Modell bewerten und verbessern können. Sie können das trainierte Modell in der AWS Cloud hosten oder das Modell auf einem Edge-Gerät bereitstellen. Eine einfache API-Operation gibt die Vorhersagen zurück, die Ihr Modell macht.

Der allgemeine Arbeitsablauf für die Erstellung, Evaluierung und Verwendung eines Modells sieht wie folgt aus:



Themen

- [Wählen Sie Ihren Modelltyp](#)
- [Erstellen Sie Ihr Modell](#)
- [Bewerten Ihres Modells](#)
- [Verwenden Sie Ihr Modell](#)
- [Verwenden Sie Ihr Modell auf einem Edge-Gerät](#)
- [Verwenden Sie Ihr Dashboard](#)

Wählen Sie Ihren Modelltyp

Bevor Sie ein Modell erstellen können, müssen Sie entscheiden, welchen Modelltyp Sie möchten. Sie können zwei Modelltypen erstellen: Bildklassifizierung und Bildsegmentierung. Sie entscheiden anhand Ihres Anwendungsfalls, welcher Modelltyp erstellt werden soll.

Modell zur Bildklassifizierung

Wenn Sie nur wissen möchten, ob ein Bild eine Anomalie enthält, aber nicht wissen müssen, wo sie sich befindet, erstellen Sie ein Bildklassifizierungsmodell. Ein Bildklassifizierungsmodell macht eine Vorhersage darüber, ob ein Bild eine Anomalie enthält. Die Vorhersage beinhaltet das Vertrauen des

Modells in die Genauigkeit der Vorhersage. Das Modell liefert keine Informationen über die Position von Anomalien, die auf dem Bild gefunden wurden.

Modell der Bildsegmentierung

Wenn Sie die Position einer Anomalie kennen müssen, z. B. die Position eines Kratzers, erstellen Sie ein Bildsegmentierungsmodell. Amazon Lookout for Vision Vision-Modelle verwenden semantische Segmentierung, um die Pixel auf einem Bild zu identifizieren, bei denen die Arten von Anomalien (z. B. ein Kratzer oder ein fehlender Teil) vorhanden sind.

Note

Ein semantisches Segmentierungsmodell lokalisiert verschiedene Arten von Anomalien. Es bietet keine Instanzinformationen für einzelne Anomalien. Wenn ein Bild beispielsweise zwei Dellen enthält, gibt Lookout for Vision Informationen über beide Dellen in einer einzigen Entität zurück, die den Typ der Dellenanomalie darstellt.

Ein Segmentierungsmodell von Amazon Lookout for Vision prognostiziert Folgendes:

Klassifizierung

Das Modell gibt eine Klassifizierung für ein analysiertes Bild zurück (Normal/Anomalie), die das Vertrauen des Modells in die Vorhersage einschließt. Klassifizierungsinformationen werden getrennt von Segmentierungsinformationen berechnet, und Sie sollten nicht von einer Beziehung zwischen ihnen ausgehen.

Segmentierung

Das Modell gibt eine Bildmaske zurück, die die Pixel markiert, an denen Anomalien im Bild auftreten. Verschiedene Arten von Anomalien sind entsprechend der Farbe, die der Anomaliebezeichnung im Datensatz zugewiesen wurde, farblich gekennzeichnet. Ein Anomalie-Label steht für den Typ einer Anomalie. Die blaue Maske in der folgenden Abbildung markiert beispielsweise die Position einer Kratzanomalie, die an einem Auto gefunden wurde.



Das Modell gibt den Farbcode für jede Anomaliebezeichnung in der Maske zurück. Das Modell gibt auch die prozentuale Abdeckung des Bilds zurück, die ein Anomalie-Label aufweist.

Mit einem Segmentierungsmodell von Lookout for Vision können Sie verschiedene Kriterien verwenden, um die Analyseergebnisse des Modells zu analysieren. Zum Beispiel:

- Lokalisation von Anomalien — Wenn Sie wissen möchten, wo sich Anomalien befinden, verwenden Sie die Segmentierungsinformationen, um Masken zu finden, die Anomalien abdecken.
- Arten von Anomalien — Verwenden Sie Segmentierungsinformationen, um zu entscheiden, ob ein Bild mehr als eine akzeptable Anzahl von Anomaliearten enthält.
- Erfassungsbereich — Verwenden Sie Segmentierungsinformationen, um zu entscheiden, ob ein Anomalie-Typ mehr als einen akzeptablen Bereich eines Bildes abdeckt.
- Bildklassifizierung — Wenn Sie nicht wissen müssen, wo sich Anomalien befinden, verwenden Sie Klassifizierungsinformationen, um festzustellen, ob ein Bild Anomalien enthält.

Beispielcode finden Sie unter [Erkennung von Anomalien in einem Bild](#).

Nachdem Sie sich für einen Modelltyp entschieden haben, erstellen Sie ein Projekt und einen Datensatz zur Verwaltung Ihres Modells. Mithilfe von Labels können Sie Bilder als normale Bilder oder als anomale Bilder klassifizieren. Labels identifizieren auch Segmentierungsinformationen wie Masken und Anomaliearten. Wie Sie die Bilder in Ihrem Datensatz beschriften, bestimmt den Modelltyp, den Lookout for Vision für Sie erstellt.

Die Kennzeichnung eines Bildsegmentierungsmodells ist komplexer als die Kennzeichnung eines Bildklassifizierungsmodells. Um ein Segmentierungsmodell zu trainieren, müssen Sie die Trainingsbilder als normal oder anomal klassifizieren. Sie müssen auch Anomaliemasken und Anomalie-Typen für jedes anomale Bild definieren. Bei einem Klassifizierungsmodell müssen Sie lediglich Trainingsbilder als normal oder anomal identifizieren.

Erstellen Sie Ihr Modell

Die Schritte zum Erstellen eines Modells umfassen das Erstellen eines Projekts, das Erstellen eines Datensatzes und das Trainieren des Modells wie folgt:

Erstellen eines Projekts

Erstellen Sie ein Projekt, um die von Ihnen erstellten Datensätze und Modelle zu verwalten. Ein Projekt sollte für einen einzigen Anwendungsfall verwendet werden, z. B. für die Erkennung von Anomalien in einem einzigen Maschinentyp.

Sie können das Dashboard verwenden, um sich einen Überblick über Ihre Projekte zu verschaffen. Weitere Informationen finden Sie unter [Verwenden des Amazon Lookout for Vision Vision-Dashboards](#).

Weitere Informationen: [Erstelle dein Projekt](#).

Erstellen eines Datensatzes

Um ein Modell zu trainieren, benötigt Amazon Lookout for Vision Bilder von normalen und anomalen Objekten für Ihren Anwendungsfall. Sie stellen diese Bilder in einem Datensatz bereit.

Ein Datensatz besteht aus einer Reihe von Bildern und Labels, die diese Bilder beschreiben. Die Bilder sollten einen einzigen Objekttyp darstellen, an dem Anomalien auftreten können. Weitere Informationen finden Sie unter [Bilder für einen Datensatz vorbereiten](#).

Mit Amazon Lookout for Vision können Sie ein Projekt haben, das einen einzelnen Datensatz verwendet, oder ein Projekt, das separate Trainings- und Testdatensätze enthält. Wir empfehlen, ein einzelnes Datensatzprojekt zu verwenden, es sei denn, Sie möchten eine genauere Kontrolle über Training, Tests und Leistungsoptimierung.

Sie erstellen einen Datensatz, indem Sie die Bilder importieren. Je nachdem, wie Sie die Bilder importieren, können die Bilder auch beschriftet werden. Wenn nicht, verwenden Sie die Konsole, um die Bilder zu beschriften.

Bilder importieren

Wenn Sie den Datensatz mit der Lookout for Vision Vision-Konsole erstellen, können Sie die Bilder auf eine der folgenden Arten importieren:

- [Importieren Sie Bilder von Ihrem lokalen Computer](#). Die Bilder sind nicht beschriftet.
- [Importiert Bilder aus einem S3-Bucket](#). Amazon Lookout for Vision kann die Bilder anhand der Ordernamen klassifizieren, die die Bilder enthalten. `normal` Für normale Bilder verwenden. `anomaly` Für anomale Bilder verwenden. Sie können Segmentierungslabels nicht automatisch zuweisen.
- [Importieren Sie eine Amazon SageMaker AI Ground Truth Manifestdatei](#). Bilder in einer Manifestdatei sind beschriftet. Sie können Ihre eigene Manifestdatei erstellen und importieren. Wenn Sie viele Bilder haben, sollten Sie den SageMaker AI Ground Truth Labeling-Service in Betracht ziehen. Anschließend importieren Sie die Ausgabe-Manifestdatei aus dem Amazon SageMaker AI Ground Truth Job.

Labeling von Bildern

Labels beschreiben ein Bild in einem Datensatz. Labels geben an, ob ein Bild normal oder ungewöhnlich ist (Klassifizierung). Beschriftungen beschreiben auch die Position von Anomalien auf einem Bild (Segmentierung).

Wenn Ihre Bilder nicht beschriftet sind, können Sie sie mit der Konsole beschriften.

Die Beschriftungen, die Sie Bildern in Ihrem Datensatz zuweisen, bestimmen den Modelltyp, den Lookout for Vision erstellt:

Bildklassifizierung

Um ein Bildklassifizierungsmodell zu erstellen, verwenden Sie die Lookout for Vision [Vision-Konsole](#), um Bilder im Datensatz als normal oder als Anomalie zu klassifizieren.

[Sie können den CreateDataset Vorgang auch verwenden, um einen Datensatz aus einer Manifestdatei zu erstellen, die Klassifizierungsinformationen enthält.](#)

Bildsegmentierung

Um ein Bildsegmentierungsmodell zu erstellen, verwenden Sie die Lookout for Vision [Vision-Konsole](#), um Bilder im Datensatz als normal oder als Anomalie zu klassifizieren. Sie geben auch Pixelmasken für anomale Bereiche auf dem Bild (falls vorhanden) sowie eine Anomaliebezeichnung für einzelne Anomaliemasken an.

Sie können den CreateDataset Vorgang auch verwenden, um einen Datensatz aus einer Manifestdatei zu erstellen, die [Segmentierungs](#) - und Klassifizierungsinformationen enthält.

Wenn Ihr Projekt separate Trainings- und Testdatensätze enthält, verwendet Lookout for Vision den Trainingsdatensatz, um zu lernen und den Modelltyp zu bestimmen. Sie sollten die Bilder in Ihrem Testdatensatz auf die gleiche Weise beschriften.

Weitere Informationen: [Erstellen Sie Ihren Datensatz](#).

Trainieren Ihres Modells

Beim Training wird ein Modell erstellt und trainiert, um das Vorhandensein von Anomalien in Bildern vorherzusagen. Bei jedem Training wird eine neue Version Ihres Modells erstellt.

Zu Beginn des Trainings wählt Amazon Lookout for Vision den am besten geeigneten Algorithmus für das Training Ihres Modells aus. Das Modell wird trainiert und anschließend getestet. Wenn Sie ein einzelnes Datensatzprojekt trainieren, wird der Datensatz intern aufgeteilt, um einen Trainingsdatensatz und einen Testdatensatz zu erstellen. [Erste Schritte mit Amazon Lookout for Vision](#) Sie können auch ein Projekt mit separaten Trainings- und Testdatensätzen erstellen. In dieser Konfiguration trainiert Amazon Lookout for Vision Ihr Modell mit dem Trainingsdatensatz und testet das Modell mit dem Testdatensatz.

Important

Ihnen wird die Zeit in Rechnung gestellt, die benötigt wird, um Ihr Modell erfolgreich zu trainieren.

Weitere Informationen: [Trainieren Sie Ihr Modell](#).

Bewerten Ihres Modells

Bewerten Sie die Leistung Ihres Modells anhand der beim Testen erstellten Leistungsmetriken.

Mithilfe von Leistungskennzahlen können Sie die Leistung Ihres trainierten Modells besser verstehen und entscheiden, ob Sie es in der Produktion einsetzen möchten.

Weitere Informationen: [Verbesserung Ihres Modells](#).

Wenn die Leistungskennzahlen darauf hinweisen, dass Verbesserungen erforderlich sind, können Sie weitere Trainingsdaten hinzufügen, indem Sie eine Aufgabe zur Versuchserkennung mit neuen Bildern ausführen. Nach Abschluss der Aufgabe können Sie die Ergebnisse überprüfen

und die verifizierten Bilder zu Ihrem Trainingsdatensatz hinzufügen. Alternativ können Sie neue Trainingsbilder direkt zum Datensatz hinzufügen. Als Nächstes trainieren Sie Ihr Modell erneut und überprüfen die Leistungskennzahlen erneut.

Weitere Informationen: [Verifizieren Ihres Modells mit einer Aufgabe zur Erkennung von Versuchen](#).

Verwenden Sie Ihr Modell

Bevor Sie Ihr Modell in der AWS Cloud verwenden können, starten Sie das Modell mit der [StartModel](#) Operation. Sie können den `StartModel` CLI-Befehl für Ihr Modell von der Konsole abrufen.

Weitere Informationen: [Starten Sie Ihr Modell](#).

Ein trainiertes Amazon Lookout for Vision Vision-Modell sagt voraus, ob ein Eingabebild normalen oder anomalen Inhalt enthält. Wenn es sich bei Ihrem Modell um ein Segmentierungsmodell handelt, umfasst die Vorhersage eine Anomalienmaske, die die Pixel markiert, an denen Anomalien gefunden werden.

Um mit Ihrem Modell eine Vorhersage zu treffen, rufen Sie die [DetectAnomalies](#) Operation auf und übergeben Sie ein Eingabebild von Ihrem lokalen Computer. Sie können den CLI-Befehl, der aufgerufen wird, `DetectAnomalies` von der Konsole aus abrufen.

Weitere Informationen: [Erkennen Sie Anomalien in einem Bild](#).

Important

Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird.

Wenn Sie Ihr Modell nicht mehr verwenden, verwenden Sie den [StopModel](#) Vorgang, um das Modell zu beenden. Sie können den CLI-Befehl von der Konsole abrufen.

Weitere Informationen: [Stoppen Sie Ihr Modell](#).

Verwenden Sie Ihr Modell auf einem Edge-Gerät

Sie können ein Lookout for Vision Vision-Modell auf einem AWS IoT Greengrass Version 2 Core-Gerät verwenden.

Weitere Informationen: [Verwenden Ihres Amazon Lookout for Vision Vision-Modells auf einem Edge-Gerät.](#)

Verwenden Sie Ihr Dashboard

Sie können das Dashboard verwenden, um sich einen Überblick über all Ihre Projekte und Übersichtsinformationen für einzelne Projekte zu verschaffen.

Weitere Informationen: [Nutze dein Dashboard.](#)

Erste Schritte mit Amazon Lookout for Vision

Bevor Sie mit diesen Anleitungen für die Ersten Schritte beginnen, empfehlen wir Ihnen, [Amazon Lookout for Vision verstehen](#) zu lesen.

In der Anleitung Erste Schritte erfahren Sie, wie Sie ein [Bildsegmentierungsmodell](#) erstellen können. Wenn Sie ein Beispiel für ein [Bildklassifizierungsmodell](#) erstellen möchten, finden Sie weitere Informationen unter [Datensatz zur Bildklassifizierung](#).

Wenn Sie schnell ein Beispielmmodell ausprobieren möchten, stellen wir Ihnen Beispiel-Trainings- und Maskenbilder zur Verfügung. Wir stellen auch ein Python-Skript zur Verfügung, das eine [Manifestdatei für die Bildsegmentierung](#) erstellt. Sie verwenden die Manifestdatei, um einen Datensatz für Ihr Projekt zu erstellen, und Sie müssen die Bilder im Datensatz nicht beschriften. Wenn Sie ein Modell mit Ihren eigenen Bildern erstellen, müssen Sie die Bilder im Datensatz beschriften. Weitere Informationen finden Sie unter [Erstellen Sie Ihren Datensatz](#).

Bei den von uns bereitgestellten Bildern handelt es sich um normale und anomale Cookies. Ein anomaler Keks hat einen Riss in der Keksform. Das Modell, das Sie mit den Bildern trainieren, prognostiziert eine Klassifizierung (normal oder anomal) und ermittelt den Bereich (Maske) mit Rissen in einem anomalen Cookie, wie im folgenden Beispiel gezeigt.



Themen

- [Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch](#)
- [Schritt 2: Erstellen Sie das Modell](#)
- [Schritt 3: Starten Sie das Modell](#)
- [Schritt 4: Analysieren Sie ein Bild](#)
- [Schritt 5: Stoppen Sie das Modell](#)
- [Nächste Schritte](#)

Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch

In diesem Verfahren klonen Sie das Dokumentationsarchiv von Amazon Lookout for Vision auf Ihren Computer. Anschließend verwenden Sie ein Python-Skript (Version 3.7 oder höher), um eine Manifestdatei zu erstellen und die Trainingsbilder und Maskenbilder an einen von Ihnen angegebenen Amazon S3 S3-Speicherort hochzuladen. Sie verwenden die Manifestdatei, um Ihr Modell zu erstellen. Später verwenden Sie Testbilder im lokalen Repository, um Ihr Modell zu testen.

Um die Manifestdatei zu erstellen und Bilder hochzuladen

1. Richten Sie Amazon Lookout for Vision ein, indem Sie den Anweisungen unter [Amazon Lookout for Vision einrichten](#) folgen. Stellen Sie sicher, dass Sie das [AWS SDK für Python](#) installieren.
2. [Erstellen Sie in der AWS Region, in der Sie Lookout for Vision verwenden möchten, einen S3-Bucket.](#)
3. Erstellen Sie im Amazon S3 S3-Bucket [einen Ordner](#) mit dem Namen `getting-started`.
4. Notieren Sie sich den Amazon S3 S3-URI und den Amazon-Ressourcennamen (ARN) für den Ordner. Sie verwenden sie, um Berechtigungen einzurichten und das Skript auszuführen.
5. Stellen Sie sicher, dass der Benutzer, der das Skript aufruft, berechtigt ist, den `s3:PutObject` Vorgang aufzurufen. Sie können die folgende Richtlinie verwenden. Informationen zum Zuweisen von Berechtigungen finden Sie unter [Zuweisen von Berechtigungen](#).

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

6. Stellen Sie sicher, dass Sie ein lokales Profil mit dem Namen `lookoutvision-access` und dass der Profilbenutzer über die im vorherigen Schritt angegebenen Berechtigungen verfügt. Weitere Informationen finden Sie unter [Verwenden eines Profils auf Ihrem lokalen Computer](#).

7. Laden Sie die ZIP-Datei [getting-started.zip](#) herunter. Die ZIP-Datei enthält den Datensatz „Erste Schritte“ und das Setup-Skript.
8. Entpacken Sie die Datei `getting-started.zip`.
9. Gehen Sie in der Befehlszeile wie folgt vor:
 - a. Navigieren Sie zum Verzeichnis `getting-started`.
 - b. Führen Sie den folgenden Befehl aus, um eine Manifestdatei zu erstellen und die Trainingsbilder und Bildmasken in den Amazon S3 S3-Pfad hochzuladen, den Sie in Schritt 4 notiert haben.

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

- c. Wenn das Skript abgeschlossen ist, notieren Sie sich den Pfad zu der `train.manifest` Datei, nach der das Skript angezeigt wird. `Create dataset using manifest file:.` Der Pfad sollte ähnlich sein wie `3://path to getting started folder/manifests/train.manifest`.

Schritt 2: Erstellen Sie das Modell

In diesem Verfahren erstellen Sie ein Projekt und einen Datensatz mit den Bildern und der Manifestdatei, die Sie zuvor in Ihren Amazon S3 S3-Bucket hochgeladen haben. Anschließend erstellen Sie das Modell und sehen sich die Bewertungsergebnisse des Modelltrainings an.

Da Sie den Datensatz anhand der Manifestdatei „Erste Schritte“ erstellen, müssen Sie die Bilder des Datensatzes nicht beschriften. Wenn Sie einen Datensatz mit Ihren eigenen Bildern erstellen, müssen Sie die Bilder beschriften. Weitere Informationen finden Sie unter [Labeling von Bildern](#).

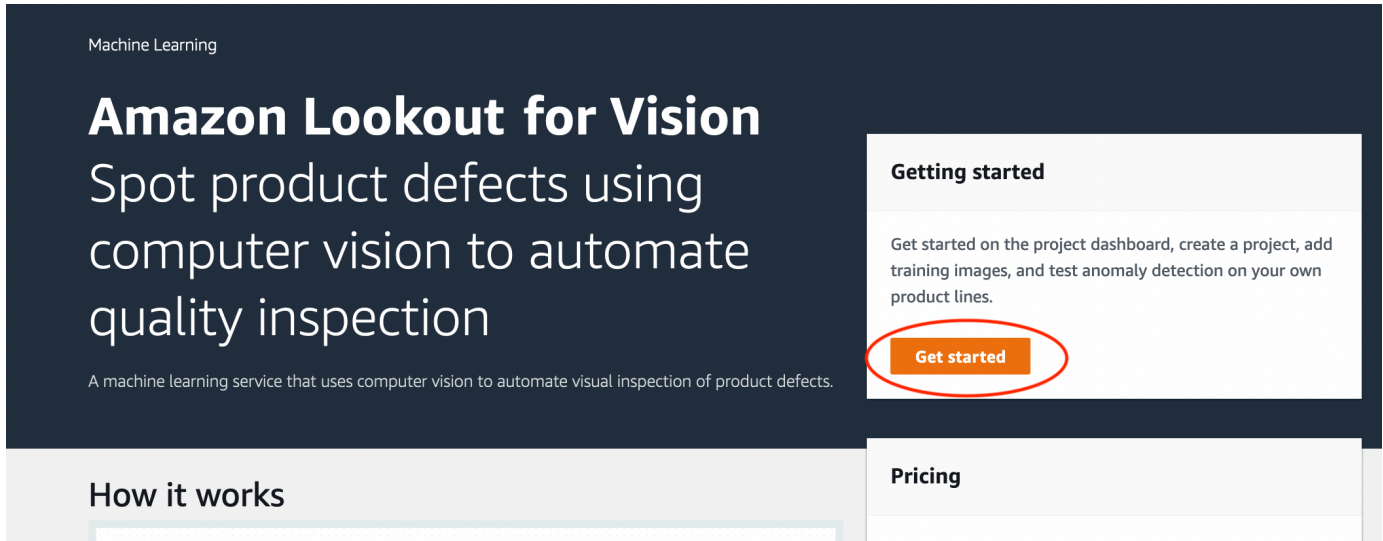
Important

Das erfolgreiche Training eines Modells wird Ihnen in Rechnung gestellt.

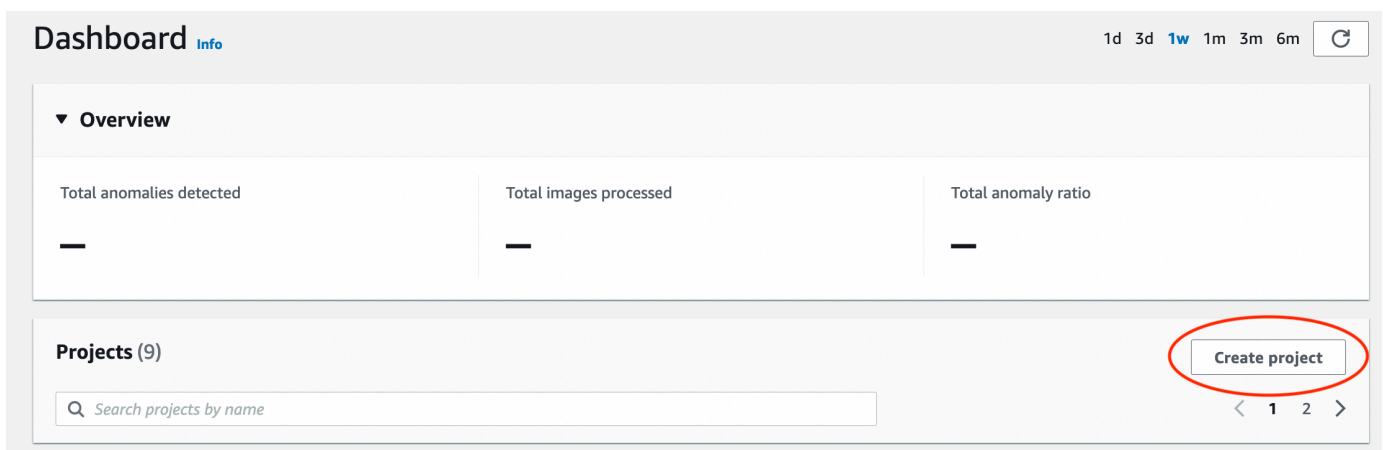
So erstellen Sie ein Modell

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.

2. Stellen Sie sicher, dass Sie sich in derselben AWS Region befinden, in der Sie den Amazon S3 S3-Bucket erstellt haben [Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch](#). Um die Region zu ändern, wählen Sie den Namen der aktuell angezeigten Region in der Navigationsleiste. Wählen Sie dann die Region aus, zu der Sie wechseln möchten.
3. Wählen Sie Erste Schritte.



4. Wählen Sie im Bereich Projekte die Option Projekt erstellen aus.



5. Gehen Sie auf der Seite Projekt erstellen wie folgt vor:
 - a. Geben Sie im Feld Projektname den Wert eingetting-started.
 - b. Wählen Sie Create project (Projekt erstellen) aus.

Create project [Info](#)

 The first step in creating an anomaly detection model is to create a project. A project manages the datasets and the versions of a model that you create. To ensure the best results, your project should address a single use case. 

Project details

Project name

getting-started

The project name must have no more than 255 characters. Valid characters are a-z, A-Z, 0-9, - and _ only. Name must begin with an alphanumeric character.

Cancel

Create project

6. Wählen Sie auf der Projektseite im Abschnitt So funktioniert's die Option Datensatz erstellen aus.

getting-started Info

▼ How it works

How to prepare your dataset



Create dataset

Add images to your dataset. The images are used to train and test your model. For better results, include images with normal and anomalous content.

Create dataset



Add labels

Add labels to classify the images in your dataset as normal or anomalous.

Add labels

How to train your model



Train model

Train your model with your dataset. After training, your model can detect anomalies in new images. Your model might require further training before you can use it.

Train model

7. Gehen Sie auf der Seite Datensatz erstellen wie folgt vor:
 - a. Wählen Sie „Einen einzelnen Datensatz erstellen“ aus.
 - b. Wählen Sie im Abschnitt Konfiguration der Bildquelle die Option Bilder importieren mit der Bezeichnung SageMaker Ground Truth aus.
 - c. Geben Sie für den Speicherort der Datei .manifest den Amazon S3 S3-Speicherort der Manifest-Datei ein, den Sie in Schritt 6.c. von notiert haben. [Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch](#) Der Amazon S3 S3-Standort sollte ähnlich sein wie `s3://path to getting started folder/manifests/train.manifest`
 - d. Wählen Sie Datensatz erstellen.

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.

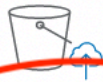
Image source configuration

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. Sehen Sie sich auf der Seite mit den Projektdetails im Abschnitt Bilder die Datensatzbilder an. Sie können die Klassifizierungs- und Bildsegmentierungsinformationen (Masken- und Anomaliebezeichnungen) für jedes Datensatzbild einsehen. Sie können auch nach Bildern suchen, Bilder nach dem Beschriftungsstatus (beschriftet/unbeschriftet) filtern oder Bilder nach den ihnen zugewiesenen Anomalie-Labels filtern.

The screenshot shows the 'Images (27)' section of the Amazon Lookout for Vision project details page. On the left, the 'Filters' panel shows 'All images (63)' selected, with 'Labeled (63)' and 'Unlabeled (0)' also listed. Below this, the 'Anomaly labels' section shows 'cracked (32)' selected. The main area displays three image thumbnails: 'anomaly-0.jpg', 'anomaly-10.jpg', and 'anomaly-11.jpg'. Each thumbnail shows a chocolate chip cookie with a green mask indicating a crack. Below each image, the word 'Anomaly' is written in red, and a dropdown menu shows 'Anomaly labels (1)' with 'cracked' selected. A 'Start labeling' button is visible in the top right corner.

9. Wählen Sie auf der Seite mit den Projektdetails die Option Zugmodell aus.

The screenshot shows the 'getting-started' page of the Amazon Lookout for Vision project details. At the top right, the 'Train model' button is highlighted with a red circle. Below this, the 'How it works: Prepare your datasets' section is expanded, showing two steps: '1. Classify images' and '2. Add anomalous areas'. A green checkmark icon indicates that the user has enough labeled images to train a model. The text below the checkmark states: 'You have enough labeled images to train a model.' and lists three bullet points: 'You can improve the quality of your model by adding more labeled images.', 'Unlabeled images aren't used for training.', and 'Click 'Train model' above to start training a model.'

10. Wählen Sie auf der Seite mit den Details zum Zugmodell die Option Zugmodell aus.

11. Wählen Sie im Dialogfenster Möchten Sie Ihr Modell trainieren? Modell trainieren aus.
12. Auf der Seite mit den Projektmodellen können Sie sehen, dass das Training begonnen hat. Überprüfen Sie den aktuellen Status, indem Sie die Spalte Status für die Modellversion aufrufen. Das Training des Modells dauert mindestens 30 Minuten. Die Schulung wurde erfolgreich abgeschlossen, wenn sich der Status in Training abgeschlossen ändert.
13. Wenn die Schulung abgeschlossen ist, wählen Sie auf der Seite Modelle das Modell Model 1 aus.

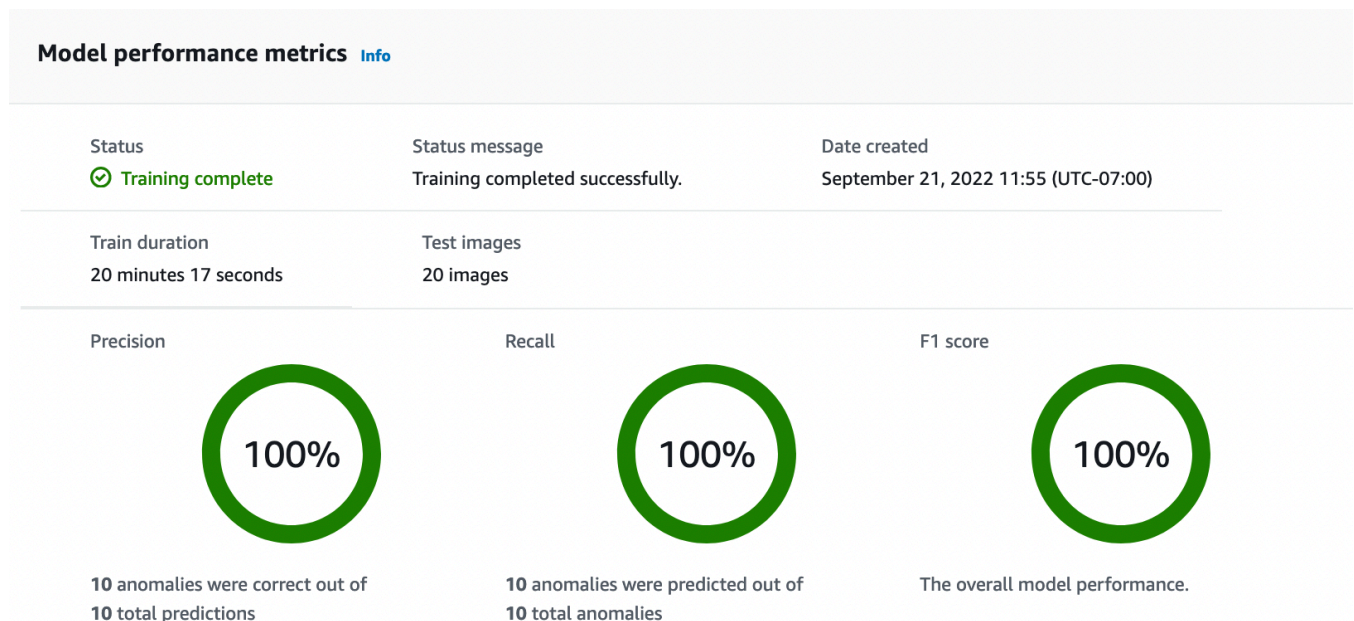
Amazon Lookout for Vision > Projects > getting-started > Models

Models (1) Info Delete Use model ▼

Q Search project models by project model name < 1 ... >

Model	Status	Date created	Precision	Recall
☐ Model 1	✓ Training complete	September 21st, 2022	100%	100%

14. Sehen Sie sich auf der Detailseite des Modells die Bewertungsergebnisse auf der Registerkarte Leistungskennzahlen an. Es gibt Metriken für Folgendes:
 - Allgemeine Modellleistungskennzahlen ([Präzision](#), [Erinnerungsvermögen](#) und [F1-Score](#)) für die vom Modell getroffenen Klassifikationsvorhersagen.



- Leistungskennzahlen für Anomaliekennzeichnungen, die in den Testbildern gefunden wurden ([durchschnittliche IoU](#), F1-Score)

Performance per label (1) [Info](#)
 Search performance labels by label name

< 1 >

Label ▲	Test images ▼	F1 score ▼	Average IoU ▼
cracked	10	86.1%	74.53%

- Vorhersagen für [Testbilder](#) (Klassifizierung, Segmentierungsmasken und Anomalie-Labels)

Images (20) [Info](#)
 Find images

< 1 2 3 ... >

normal-125.jpg



✓ Correct

Prediction

Normal

Confidence

95%

anomaly-38.jpg



✓ Correct

Prediction

Anomaly

Confidence

95.3%

▼ Anomaly labels (1)

■ cracked

anomaly-35.jpg



✓ Correct

Prediction

Anomaly

Confidence

95.4%

▶ Anomaly labels (1)



Da das Modelltraining nicht deterministisch ist, können Ihre Bewertungsergebnisse von den Ergebnissen auf dieser Seite abweichen. Weitere Informationen finden Sie unter [Verbesserung Ihres Amazon Lookout for Vision Vision-Modells](#).

Schritt 3: Starten Sie das Modell

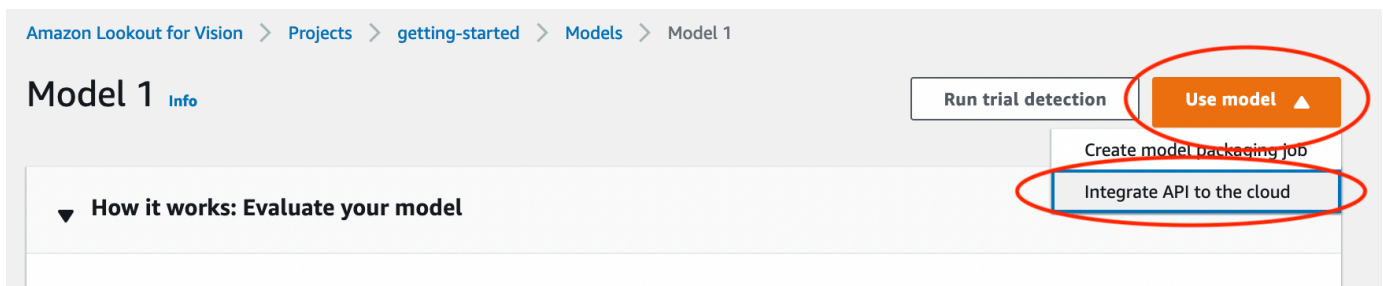
In diesem Schritt beginnen Sie mit dem Hosten des Modells, damit es für die Analyse von Bildern bereit ist. Weitere Informationen finden Sie unter [Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells](#).

Note

Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Sie stoppen Ihr Modell in [Schritt 5: Stoppen Sie das Modell](#).

Um das Modell zu starten.

1. Wählen Sie auf der Detailseite des Modells die Option Modell verwenden und dann API in die Cloud integrieren aus.



2. Kopieren Sie den `start-model` AWS CLI Befehl im Abschnitt AWS CLI Befehle.

AWS CLI commands

Use CLI commands to start, stop your model or detect anomalies in images.

Start model

Use `start-model` to start running your model. A model takes a few minutes to start. Check the status in the performance metrics tab or the models view for your project.

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

 Copy

3. Stellen Sie sicher, dass der für die Ausführung in derselben AWS Region konfiguriert AWS CLI ist, in der Sie die Amazon Lookout for Vision Vision-Konsole verwenden. Informationen zum Ändern der AWS Region, die von AWS CLI verwendet wird, finden Sie unter [Installieren Sie das AWS SDKs](#).

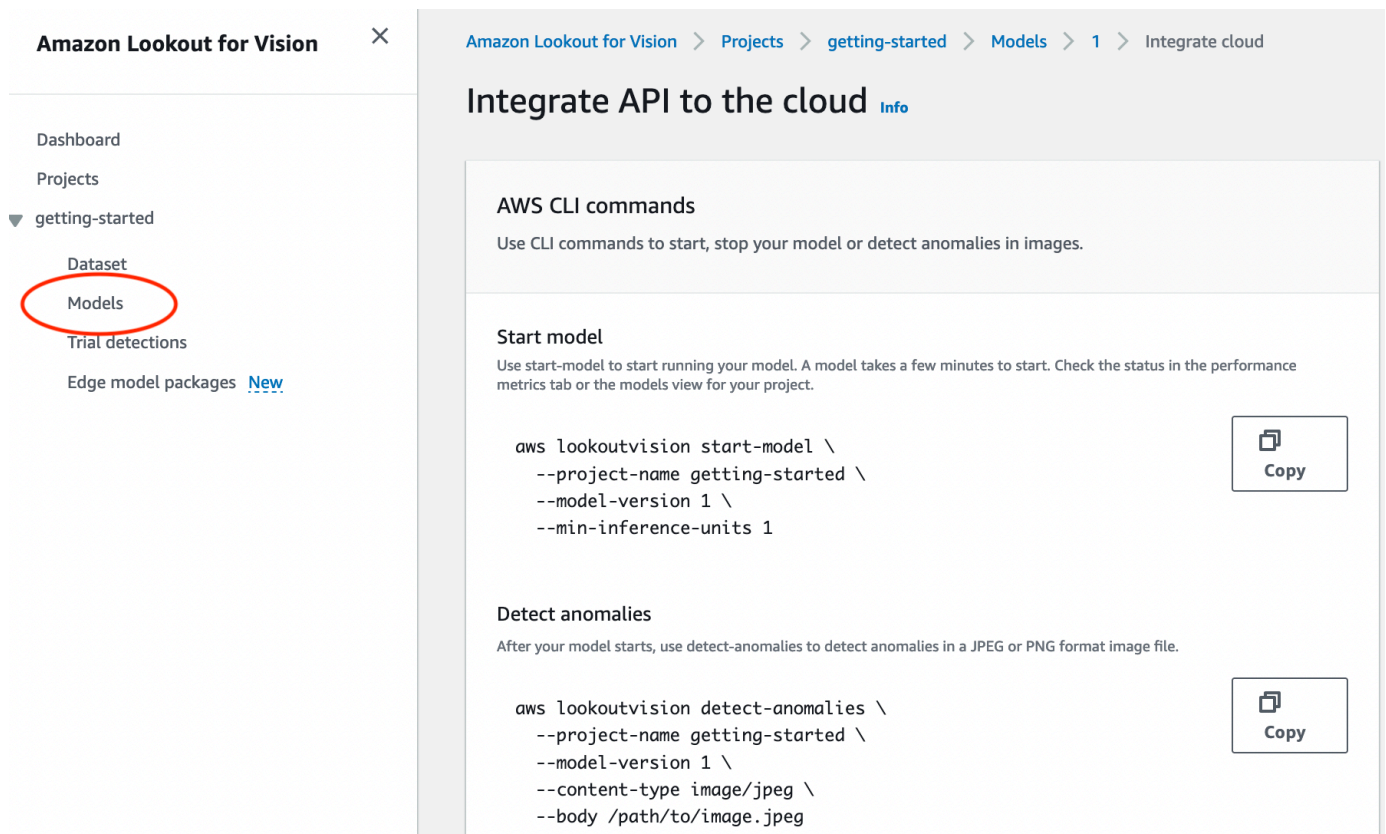
4. Starten Sie das Modell in der Befehlszeile, indem Sie den `start-model` Befehl eingeben. Wenn Sie das `lookoutvision` Profil zum Abrufen von Anmeldeinformationen verwenden, fügen Sie den `--profile lookoutvision-access` Parameter hinzu. Zum Beispiel:

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

Wenn der Anruf erfolgreich ist, wird die folgende Ausgabe angezeigt:

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

5. Zurück in der Konsole wählen Sie im Navigationsbereich Modelle aus.



The screenshot shows the Amazon Lookout for Vision console. On the left, the navigation menu is expanded, and the 'Models' option is circled in red. The main content area is titled 'Integrate API to the cloud' and contains two sections: 'AWS CLI commands' and 'Start model'. The 'Start model' section includes a code block with the command to start the model and a 'Copy' button. The 'Detect anomalies' section includes a code block with the command to detect anomalies and a 'Copy' button.

Amazon Lookout for Vision ×

Dashboard
Projects
▼ getting-started
Dataset
Models
Trial detections
Edge model packages [New](#)

Amazon Lookout for Vision > Projects > getting-started > Models > 1 > Integrate cloud

Integrate API to the cloud [Info](#)

AWS CLI commands
Use CLI commands to start, stop your model or detect anomalies in images.

Start model
Use `start-model` to start running your model. A model takes a few minutes to start. Check the status in the performance metrics tab or the models view for your project.

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

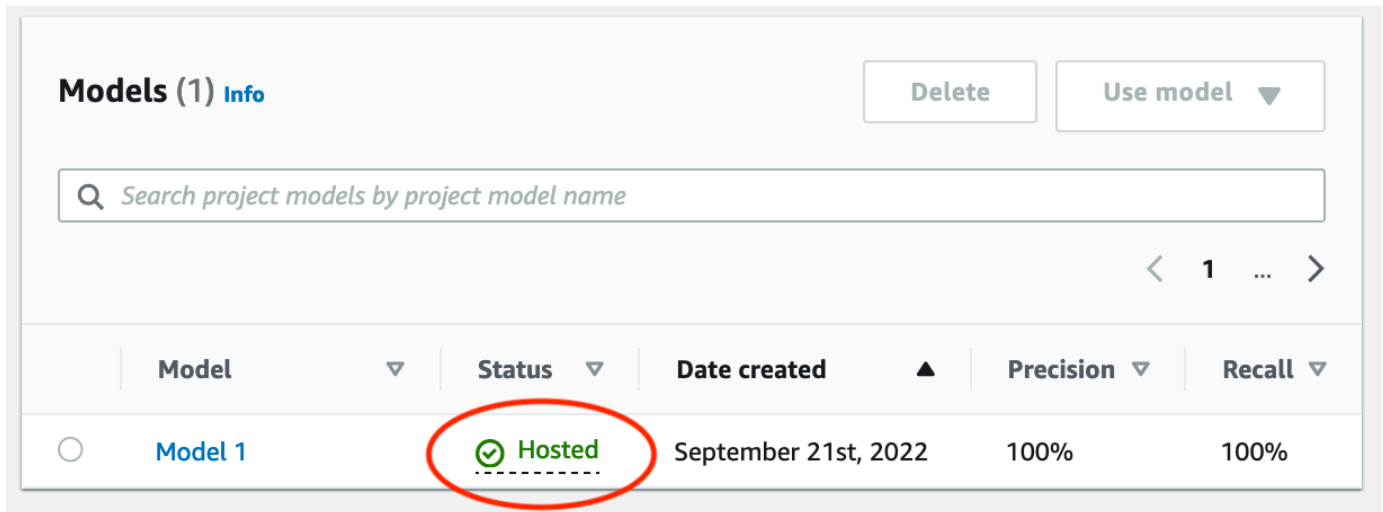
[Copy](#)

Detect anomalies
After your model starts, use `detect-anomalies` to detect anomalies in a JPEG or PNG format image file.

```
aws lookoutvision detect-anomalies \  
  --project-name getting-started \  
  --model-version 1 \  
  --content-type image/jpeg \  
  --body /path/to/image.jpeg
```

[Copy](#)

6. Warten Sie, bis der Status des Modells (Modell 1) in der Spalte Status den Wert `Gehostet` anzeigt. Wenn Sie zuvor ein Modell im Projekt trainiert haben, warten Sie, bis die neueste Modellversion abgeschlossen ist.

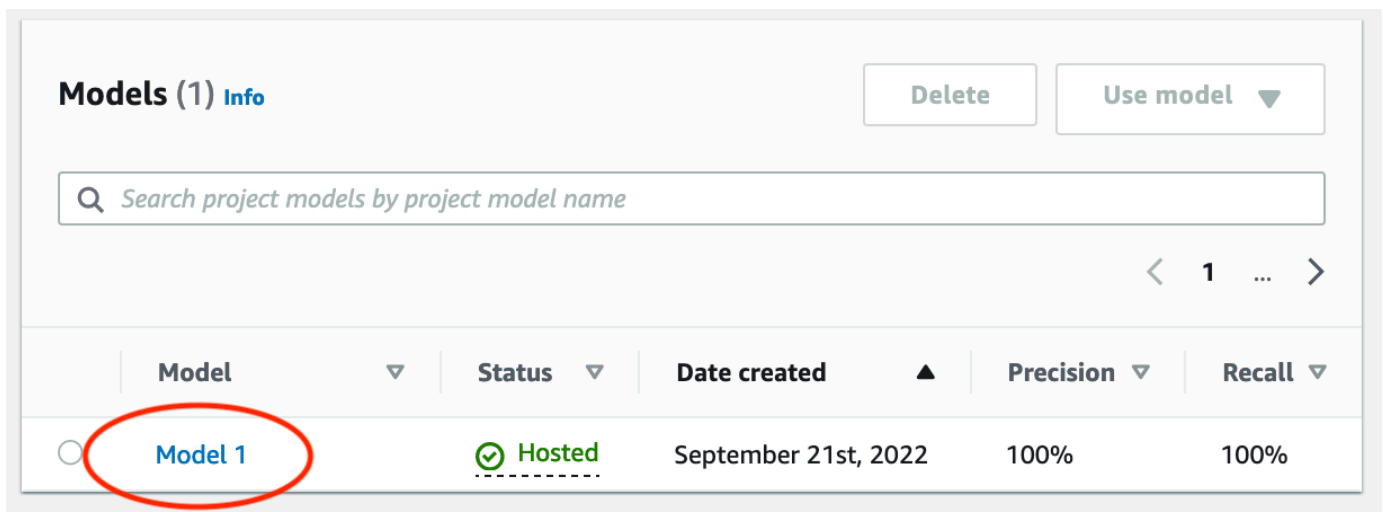


Schritt 4: Analysieren Sie ein Bild

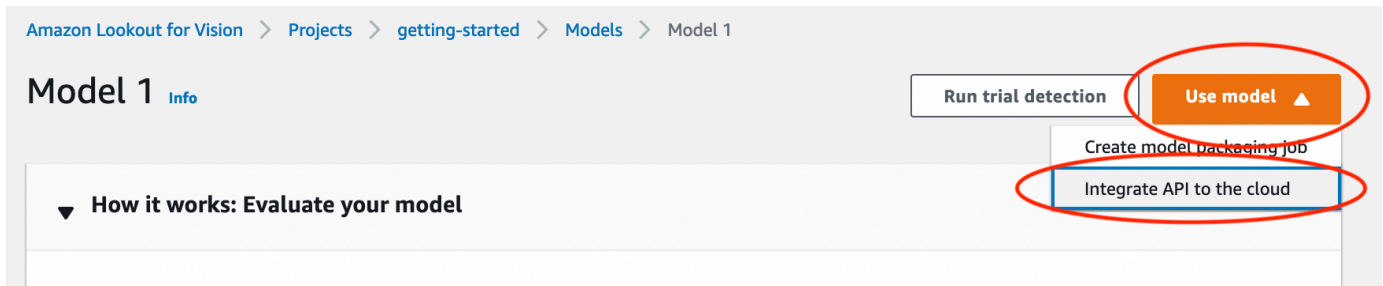
In diesem Schritt analysieren Sie ein Bild mit Ihrem Modell. Wir stellen Beispielbilder zur Verfügung, die Sie im `test-images` Ordner Erste Schritte im Dokumentationsarchiv von Lookout for Vision auf Ihrem [Computer](#) verwenden können. Weitere Informationen finden Sie unter [Erkennung von Anomalien in einem Bild](#).

Um ein Bild zu analysieren

1. Wählen Sie auf der Seite Modelle das Modell Model 1 aus.



2. Wählen Sie auf der Detailseite des Modells die Option Modell verwenden und dann API in die Cloud integrieren aus.



3. Kopieren Sie den detect-anomalies AWS CLI Befehl im Abschnitt AWS CLI Befehle.

Detect anomalies

After your model starts, use detect-anomalies to detect anomalies in a JPEG or PNG format image file.

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/image.jpeg
```

 Copy

4. Analysieren Sie in der Befehlszeile ein anomales Bild, indem Sie den detect-anomalies Befehl aus dem vorherigen Schritt eingeben. [Geben Sie für den --body Parameter ein anomales Bild aus dem test-images Ordner Erste Schritte auf Ihrem Computer an.](#) Wenn Sie das lookoutvision Profil zum Abrufen von Anmeldeinformationen verwenden, fügen Sie den --profile lookoutvision-access Parameter hinzu. Zum Beispiel:

```
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
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
{
  "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": "iVBORw0KGgoAAAANSUhEUgAAkAAAAMACA....."
}

```

5. Beachten Sie in der Ausgabe Folgendes:

- `IsAnomalous` ist ein boolescher Wert für die vorhergesagte Klassifizierung. `true` wenn das Bild anomal ist, andernfalls. `false`
- `Confidence` ist ein Gleitkommawert, der das Vertrauen darstellt, das Amazon Lookout for Vision in die Prognose hat. 0 ist das niedrigste Vertrauen, 1 das höchste Vertrauen.
- `Anomalies` ist eine Liste der im Bild gefundenen Anomalien. `Name` ist das Anomalie-Label. `PixelAnomaly` beinhaltet die prozentuale Gesamtfläche der Anomalie (`TotalPercentageArea`) und eine Farbe (`Color`) für die Kennzeichnung der Anomalie. Die Liste enthält auch eine „Hintergrundanomalie“, die den Bereich außerhalb der auf dem Bild gefundenen Anomalien abdeckt.
- `AnomalyMask` ist ein Maskenbild, das die Position der Anomalien auf dem analysierten Bild zeigt.

Sie können die Informationen in der Antwort verwenden, um eine Mischung aus dem analysierten Bild und der Anomalienmaske anzuzeigen, wie im folgenden Beispiel gezeigt. Beispielcode finden Sie unter [Informationen zur Klassifizierung und Segmentierung werden angezeigt](#).

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



6. Analysieren Sie in der Befehlszeile ein normales Bild aus dem `test-images` Ordner Erste Schritte. Wenn Sie das `lookoutvision` Profil zum Abrufen von Anmeldeinformationen verwenden, fügen Sie den `--profile lookoutvision-access` Parameter hinzu. Zum Beispiel:

```
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
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
{
  "DetectAnomalyResult": {
    "Source": {
      "Type": "direct"
    },
    "IsAnomalous": false,
    "Confidence": 0.9916400909423828,
    "Anomalies": [
      {
        "Name": "background",
        "PixelAnomaly": {
          "TotalPercentageArea": 1.0,
          "Color": "#FFFFFF"
        }
      }
    ],
    "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAkAAAAA...."
  }
}
```

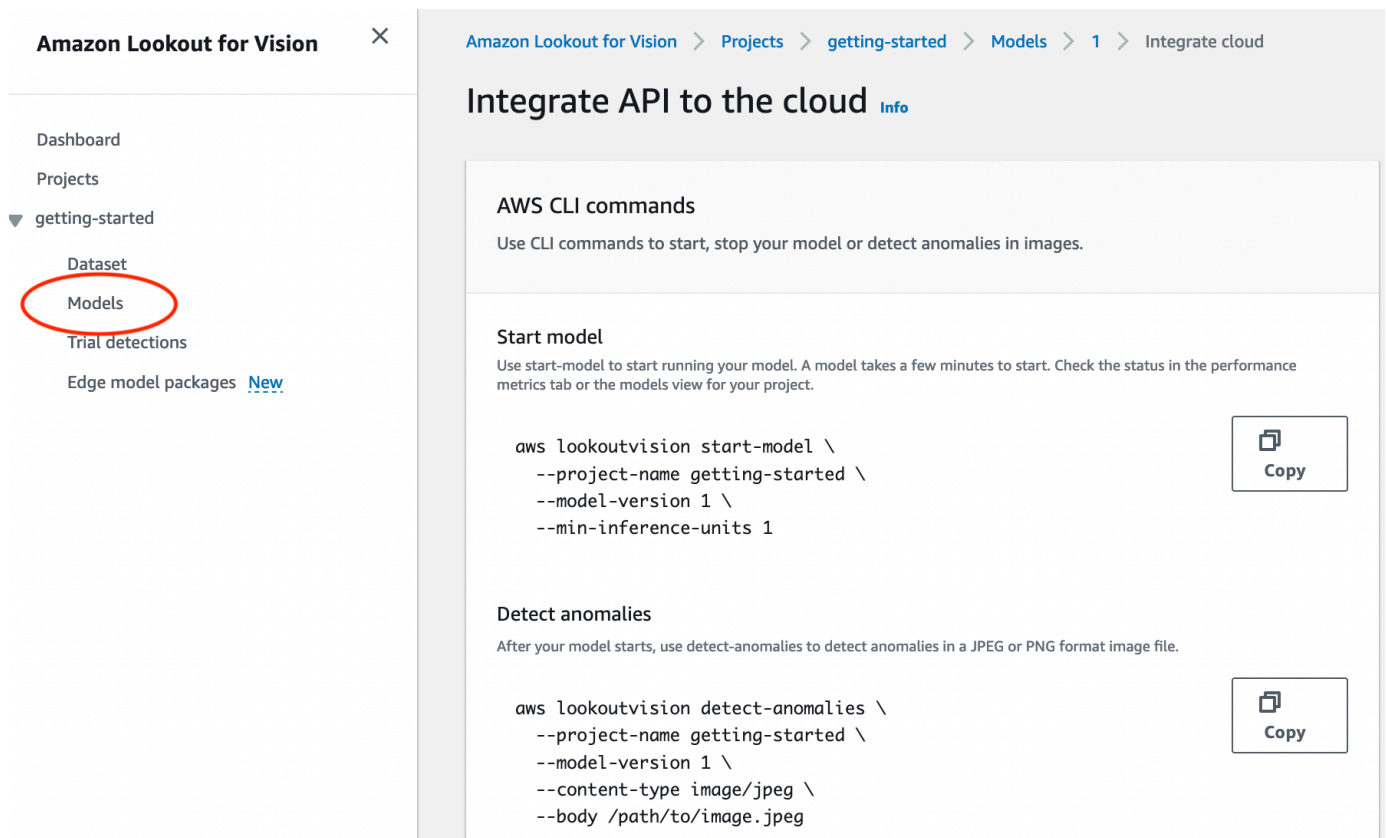
7. Beachten Sie in der Ausgabe, dass der `false` Wert für das Bild so `IsAnomalous` klassifiziert, dass es keine Anomalien aufweist. Verwenden Sie diese `Confidence` Option, um festzustellen, ob Sie der Klassifizierung vertrauen. Außerdem hat das `Anomalies` Array nur das `background` Anomalie-Label.

Schritt 5: Stoppen Sie das Modell

In diesem Schritt beenden Sie das Hosten des Modells. Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Wenn Sie das Modell nicht verwenden, sollten Sie es beenden. Sie können das Modell neu starten, wenn Sie es das nächste Mal benötigen. Weitere Informationen finden Sie unter [Starten Sie Ihr Amazon Lookout for Vision Vision-Modell](#).

Um das Modell zu stoppen.

1. Wählen Sie im Navigationsbereich Modelle aus.



The screenshot shows the Amazon Lookout for Vision console. On the left, the navigation pane is open, and the 'Models' option is highlighted with a red circle. The main content area is titled 'Integrate API to the cloud' and contains two sections: 'AWS CLI commands' and 'Start model'. The 'Start model' section includes a code block with the following command:

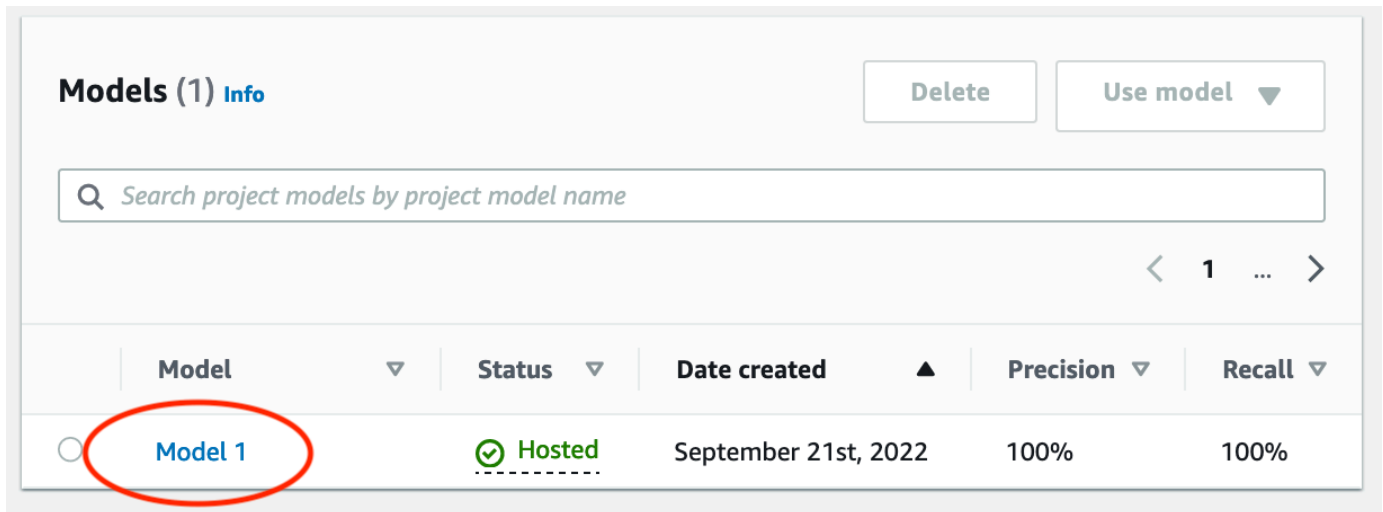
```
aws lookoutvision start-model \
  --project-name getting-started \
  --model-version 1 \
  --min-inference-units 1
```

Below this, the 'Detect anomalies' section includes a code block with the following command:

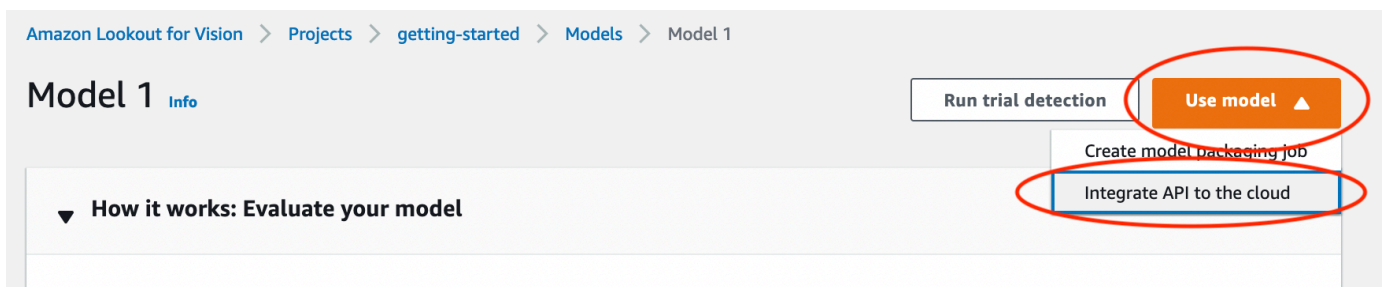
```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/image.jpeg
```

Each code block has a 'Copy' button next to it.

2. Wählen Sie auf der Seite Modelle das Modell Model 1 aus.



- Wählen Sie auf der Detailseite des Modells die Option Modell verwenden und dann API in die Cloud integrieren aus.



- Kopieren Sie den `stop-model` AWS CLI Befehl im Abschnitt AWS CLI Befehle.

Stop model

Use `stop-model` to stop your model running. You are charged for the amount of time your model runs.

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

Copy

- Beenden Sie das Modell in der Befehlszeile, indem Sie den `stop-model` AWS CLI Befehl aus dem vorherigen Schritt eingeben. Wenn Sie das `lookoutvision` Profil zum Abrufen von Anmeldeinformationen verwenden, fügen Sie den `--profile lookoutvision-access` Parameter hinzu. Zum Beispiel:

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

Wenn der Anruf erfolgreich ist, wird die folgende Ausgabe angezeigt:

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

6. Zurück in der Konsole wählen Sie auf der linken Navigationsseite Modelle aus.
7. Das Modell wurde gestoppt, wenn der Status des Modells in der Spalte Status den Status Training abgeschlossen hat.

Nächste Schritte

Wenn Sie bereit sind, ein Modell mit Ihren eigenen Bildern zu erstellen, folgen Sie zunächst den Anweisungen unter [Erstellen Sie Ihr Projekt](#). Die Anweisungen enthalten Schritte zum Erstellen eines Modells mit der Amazon Lookout for Vision Vision-Konsole und mit dem AWS SDK.

Wenn Sie andere Beispieldatensätze ausprobieren möchten, finden Sie weitere Informationen unter [Beispielcode und Datensätze](#)

Ihr Amazon Lookout for Vision Vision-Modell erstellen

Ein Modell von Amazon Lookout for Vision ist ein Modell für maschinelles Lernen, das das Vorhandensein von Anomalien in neuen Bildern vorhersagt, indem es Muster in Bildern findet, die zum Trainieren des Modells verwendet werden. In diesem Abschnitt erfahren Sie, wie Sie ein Modell erstellen und trainieren. Nachdem Sie Ihr Modell trainiert haben, bewerten Sie dessen Leistung. Weitere Informationen finden Sie unter [Verbesserung Ihres Amazon Lookout for Vision Vision-Modells](#).

Bevor Sie Ihr erstes Modell erstellen, empfehlen wir Ihnen, [Amazon Lookout for Vision verstehen](#) und zu lesen [Erste Schritte mit Amazon Lookout for Vision](#). Wenn Sie das AWS SDK verwenden, lesen Sie [Rufen Sie einen Amazon Lookout for Vision Vision-Betrieb an](#).

Themen

- [Erstellen Sie Ihr Projekt](#)
- [Erstellen Sie Ihren Datensatz](#)
- [Labeling von Bildern](#)
- [Trainieren Ihres Modells](#)
- [Problembehandlung beim Modelltraining](#)

Erstellen Sie Ihr Projekt

Ein Amazon Lookout for Vision Vision-Projekt ist eine Gruppierung der Ressourcen, die zur Erstellung und Verwaltung eines Lookout for Vision Vision-Modells benötigt werden. Ein Projekt verwaltet Folgendes:

- Datensatz — Die Bilder und Bildbeschriftungen, die zum Trainieren eines Modells verwendet wurden. Weitere Informationen finden Sie unter [Erstellen Sie Ihren Datensatz](#).
- Modell — Die Software, die Sie darauf trainieren, Anomalien zu erkennen. Sie können mehrere Versionen eines Modells haben. Weitere Informationen finden Sie unter [Trainieren Ihres Modells](#).

Wir empfehlen, dass Sie ein Projekt für einen einzigen Anwendungsfall verwenden, z. B. für die Erkennung von Anomalien in einem einzigen Maschinentyp.

 Note

Sie können AWS CloudFormation Amazon Lookout for Vision Vision-Projekte bereitstellen und konfigurieren. Weitere Informationen finden Sie unter [Erstellen von Amazon Lookout for Vision Vision-Ressourcen mit AWS CloudFormation](#).

Um Ihre Projekte anzusehen, sehen [Ihre Projekte anzeigen](#) oder öffnen Sie die [Verwenden des Amazon Lookout for Vision Vision-Dashboards](#). Informationen zum Löschen eines Modells finden Sie unter [Löschen eines Modells](#).

Themen

- [Ein Projekt erstellen \(Konsole\)](#)
- [Erstellen eines Projekts \(SDK\)](#)

Ein Projekt erstellen (Konsole)

Das folgende Verfahren zeigt Ihnen, wie Sie ein Projekt mithilfe der Konsole erstellen.

So erstellen Sie ein Projekt (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
3. Wählen Sie Create project (Projekt erstellen) aus.
4. Geben Sie im Feld Projektname einen Namen für Ihr Projekt an.
5. Wählen Sie Create project (Projekt erstellen) aus. Die Detailseite für Ihr Projekt wird angezeigt.
6. Folgen Sie den Schritten unter [Erstellen Sie Ihren Datensatz](#), um Ihren Datensatz zu erstellen.

Erstellen eines Projekts (SDK)

Sie verwenden den [CreateProject](#) Vorgang, um ein Amazon Lookout for Vision Vision-Projekt zu erstellen. Das CreateProject Antwortformular enthält den Projektnamen und den Amazon-Ressourcennamen (ARN) des Projekts. Rufen Sie anschließend an, [CreateDataset](#) um Ihrem Projekt einen Schulungs- und einen Testdatensatz hinzuzufügen. Weitere Informationen finden Sie unter [Einen Datensatz mit einer Manifestdatei \(SDK\) erstellen](#).

Rufen Sie an, um sich die Projekte anzusehen, die Sie in einem Projekt erstellt haben `ListProjects`. Weitere Informationen finden Sie unter [Ihre Projekte anzeigen](#).

So erstellen Sie ein Projekt (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um ein Modell zu erstellen.

CLI

Ändern Sie den Wert von `project-name` in den Namen, den Sie für das Projekt verwenden möchten.

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

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem GitHub Repository mit den Beispielen für das AWS Documentation SDK. Das vollständige Beispiel finden Sie [hier](#).

```
/**
 * 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. Folgen Sie den Schritten unter [Erstellen eines Datensatzes mithilfe einer Amazon SageMaker AI Ground Truth Manifestdatei](#), um Ihren Datensatz zu erstellen.

Erstellen Sie Ihren Datensatz

Ein Datensatz enthält die Bilder und zugewiesenen Beschriftungen, die Sie zum Trainieren und Testen eines Modells verwenden. Sie erstellen den Datensatz für Ihr Projekt mit der Amazon Lookout for Vision Vision-Konsole oder mit der [CreateDataset](#) Operation. Die Datensatz-Bilder müssen entsprechend dem Modelltyp, den Sie erstellen möchten, beschriftet werden (Bildklassifizierung oder Bildsegmentierung).

Themen

- [Bilder für einen Datensatz vorbereiten](#)
- [Den Datensatz erstellen](#)
- [Erstellen eines Datensatzes mit Bildern, die auf Ihrem lokalen Computer gespeichert sind](#)
- [Erstellen eines Datensatzes mit Bildern, die in einem Amazon S3 S3-Bucket gespeichert sind](#)
- [Erstellen eines Datensatzes mithilfe einer Amazon SageMaker AI Ground Truth Manifestdatei](#)

Bilder für einen Datensatz vorbereiten

Sie benötigen eine Sammlung von Bildern, um einen Datensatz zu erstellen. Ihre Bilder müssen Dateien im PNG- oder JPEG-Format sein. Die Anzahl und Art der benötigten Bilder hängt davon ab, ob Ihr Projekt einen einzelnen Datensatz oder separate Trainings- und Testdatensätze enthält.

Projekt mit einem einzigen Datensatz

Um ein Modell zur Bildklassifizierung zu erstellen, benötigen Sie Folgendes, um mit dem Training zu beginnen:

- Mindestens 20 Bilder von normalen Objekten.
- Mindestens 10 Bilder von anomalen Objekten.

Um ein Bildsegmentierungsmodell zu erstellen, benötigen Sie Folgendes, um mit dem Training zu beginnen:

- Mindestens 20 Bilder von jedem Anomalie-Typ.
- Jedes anomale Bild (Bild mit vorhandenen Anomaliearten) darf nur eine Art von Anomalie aufweisen.

- Mindestens 20 Bilder von normalen Objekten.

Separates Projekt für Trainings- und Testdatensätze

Um ein Modell zur Bildklassifizierung zu erstellen, benötigen Sie Folgendes:

- Mindestens 10 Bilder von normalen Objekten im Trainingsdatensatz.
- Mindestens 10 Bilder von normalen Objekten im Testdatensatz.
- Mindestens 10 Bilder von anomalen Objekten im Testdatensatz.

Um ein Bildsegmentierungsmodell zu erstellen, benötigen Sie Folgendes:

- Jeder Datensatz benötigt mindestens 10 Bilder jedes Anomalie-Typs.
- Jedes anomale Bild (Bild mit vorhandenen Anomaliearten) darf nur eine Art von Anomalie enthalten.
- Jeder Datensatz muss mindestens 10 Bilder von normalen Objekten enthalten.

Verwenden Sie mehr als die Mindestanzahl an Bildern, um ein Modell mit höherer Qualität zu erstellen. Wenn Sie ein Segmentierungsmodell erstellen, empfehlen wir, Bilder mit mehreren Anomalietypen einzubeziehen. Diese zählen jedoch nicht zu dem Minimum, das Lookout for Vision benötigt, um mit dem Training zu beginnen.

Ihre Bilder sollten von einem einzigen Objekttyp sein. Außerdem sollten Sie einheitliche Bedingungen für die Bildaufnahme haben, z. B. Kamerapositionierung, Beleuchtung und Objektposition.

Alle Bilder in den Trainings- und Testdatensätzen müssen dieselben Abmessungen haben. Später müssen die Bilder, die Sie mit Ihrem trainierten Modell analysieren, dieselben Abmessungen haben wie die Bilder der Trainings- und Testdatensätze. Weitere Informationen finden Sie unter [Erkennung von Anomalien in einem Bild](#).

Alle Trainings- und Testbilder müssen eindeutige Bilder sein, vorzugsweise von eindeutigen Objekten. Normale Bilder sollten die normalen Variationen des zu analysierenden Objekts erfassen. Bei anomalen Bildern sollte eine Vielzahl von Anomalien erfasst werden.

Amazon Lookout for Vision bietet Beispielbilder, die Sie verwenden können. Weitere Informationen finden Sie unter [Datensatz zur Bildklassifizierung](#).

Informationen zu Bildbeschränkungen finden Sie unter [Kontingente](#).

Den Datensatz erstellen

Wenn Sie den Datensatz für Ihr Projekt erstellen, wählen Sie die anfängliche Datensatzkonfiguration Ihres Projekts aus. Sie wählen auch aus, woher Lookout for Vision die Bilder importiert.

Wählen Sie eine Datensatzkonfiguration für Ihr Projekt

Wenn Sie den ersten Datensatz in Ihrem Projekt erstellen, wählen Sie eine der folgenden Datensatzkonfigurationen:

- **Einzelner Datensatz** — Ein einzelnes Datensatzprojekt verwendet einen einzelnen Datensatz, um Ihr Modell zu trainieren und zu testen. Die Verwendung eines einzigen Datensatzes vereinfacht das Training, da Amazon Lookout for Vision die Trainings- und Testbilder auswählen kann. Während des Trainings teilt Amazon Lookout for Vision den Datensatz intern in einen Trainingsdatensatz und einen Testdatensatz auf. Sie haben keinen Zugriff auf die geteilten Datensätze. Für die meisten Szenarien empfehlen wir, ein einzelnes Datensatzprojekt zu verwenden.
- **Separate Trainings- und Testdatensätze** — Wenn Sie eine genauere Kontrolle über Training, Tests und Leistungsoptimierung wünschen, können Sie Ihr Projekt so konfigurieren, dass es separate Trainings- und Testdatensätze enthält. Verwenden Sie einen separaten Testdatensatz, wenn Sie die Kontrolle über die zum Testen verwendeten Bilder haben möchten oder wenn Sie bereits über einen Benchmarksatz von Bildern verfügen, den Sie verwenden möchten.

Sie können einem vorhandenen Einzeldatensatzprojekt einen Testdatensatz hinzufügen. Der einzelne Datensatz wird dann zum Trainingsdatensatz. Wenn Sie den Testdatensatz aus einem Projekt mit separaten Trainings- und Testdatensätzen entfernen, wird das Projekt zu einem einzigen Datensatzprojekt. Weitere Informationen finden Sie unter [Löschen eines Datensatzes](#).

Bilder werden importiert

Wenn Sie einen Datensatz erstellen, wählen Sie aus, woher die Bilder importiert werden sollen. Je nachdem, wie Sie die Bilder importieren, sind die Bilder möglicherweise bereits beschriftet. Wenn die Bilder nach der Erstellung des Datensatzes nicht beschriftet wurden, finden Sie weitere Informationen unter [Labeling von Bildern](#).

Sie erstellen einen Datensatz und importieren seine Bilder auf eine der folgenden Arten:

- [Importieren Sie Bilder von Ihrem lokalen Computer](#). Die Bilder sind nicht beschriftet. Sie fügen Beschriftungen mithilfe der Lookout for Vision Vision-Konsole hinzu.

- [Importieren Sie Bilder aus einem S3-Bucket](#). Amazon Lookout for Vision kann Bilder klassifizieren, indem es die Ordernamen verwendet, um die Bilder zu beschriften. normalFür normale Bilder verwenden. anomalyFür anomale Bilder verwenden. Sie können Segmentierungslabels nicht automatisch zuweisen.
- [Importieren Sie eine Amazon SageMaker AI Ground Truth Manifestdatei](#), die beschriftete Bilder enthält. Sie können Ihre eigene Manifestdatei erstellen und importieren. Wenn Sie viele Bilder haben, sollten Sie den SageMaker AI Ground Truth Labeling-Service in Betracht ziehen. Anschließend importieren Sie die Ausgabe-Manifestdatei aus dem Amazon SageMaker AI Ground Truth Job. Bei Bedarf können Sie die Lookout for Vision Vision-Konsole verwenden, um Beschriftungen hinzuzufügen oder zu ändern.

Wenn Sie das AWS SDK verwenden, erstellen Sie einen Datensatz mit einer Amazon SageMaker AI Ground Truth Manifestdatei. Weitere Informationen finden Sie unter [Erstellen eines Datensatzes mithilfe einer Amazon SageMaker AI Ground Truth Manifestdatei](#).

Wenn Ihre Bilder nach der Erstellung Ihres Datensatzes beschriftet sind, können Sie [das Modell trainieren](#). Wenn die Bilder nicht beschriftet sind, fügen Sie die Beschriftungen entsprechend dem Modelltyp hinzu, den Sie erstellen möchten. Weitere Informationen finden Sie unter [Labeling von Bildern](#).

Sie können einem vorhandenen Datensatz weitere Bilder hinzufügen. Weitere Informationen finden Sie unter [Hinzufügen von Bildern zu Ihrem Datensatz](#).

Erstellen eines Datensatzes mit Bildern, die auf Ihrem lokalen Computer gespeichert sind

Sie können einen Datensatz mithilfe von Bildern erstellen, die direkt von Ihrem Computer geladen werden. Sie können bis zu 30 Bilder gleichzeitig hochladen. In diesem Verfahren können Sie ein einzelnes Datensatzprojekt oder ein Projekt mit separaten Trainings- und Testdatensätzen erstellen.

Note

Wenn Sie den Vorgang gerade abgeschlossen haben [Erstellen Sie Ihr Projekt](#), sollte in der Konsole Ihr Projekt-Dashboard angezeigt werden und Sie müssen die Schritte 1 bis 3 nicht ausführen.

So erstellen Sie einen Datensatz mit Bildern auf einem lokalen Computer (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
3. Wählen Sie auf der Seite Projekte das Projekt aus, dem Sie den Datensatz hinzufügen möchten.
4. Wählen Sie auf der Seite mit den Projektdetails die Option Datensatz erstellen aus.
5. Wählen Sie die Registerkarte Einzelner Datensatz oder Separate Trainings- und Testdatensätze und folgen Sie den Schritten.

Single dataset

- a. Wählen Sie im Abschnitt Datensatzkonfiguration die Option Einzelnes Dataset erstellen aus.
- b. Wählen Sie im Abschnitt Konfiguration der Bildquelle die Option Bilder von Ihrem Computer hochladen aus.
- c. Wählen Sie Datensatz erstellen.
- d. Wählen Sie auf der Datensatzseite die Option Bilder hinzufügen aus.
- e. Wählen Sie die Bilder aus Ihren Computerdateien aus, die Sie in den Datensatz hochladen möchten. Sie können die Bilder ziehen oder die Bilder auswählen, die Sie von Ihrem lokalen Computer hochladen möchten.
- f. Wählen Sie Bilder hochladen.

Separate training and test datasets

- a. Wählen Sie im Abschnitt Datensatzkonfiguration die Option Trainingsdatensatz und Testdatensatz erstellen aus.
- b. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Bilder von Ihrem Computer hochladen aus.
- c. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Bilder von Ihrem Computer hochladen aus.



Note

Ihre Trainings- und Testdatensätze können unterschiedliche Bildquellen haben.

- d. Wählen Sie Datensatz erstellen. Eine Datensatzseite mit den Registerkarten Training und Test für die jeweiligen Datensätze wird angezeigt.
 - e. Wählen Sie Aktionen und anschließend Bilder zum Trainingsdatensatz hinzufügen aus.
 - f. Wählen Sie die Bilder aus, die Sie in den Datensatz hochladen möchten. Sie können die Bilder ziehen oder die Bilder auswählen, die Sie von Ihrem lokalen Computer hochladen möchten.
 - g. Wählen Sie Bilder hochladen.
 - h. Wiederholen Sie die Schritte 5e-5g. Wählen Sie für Schritt 5e Aktionen und dann Bilder zum Testdatensatz hinzufügen aus.
6. Folgen Sie den Schritten unter [Labeling von Bildern](#), um Ihre Bilder zu beschriften.
 7. Folgen Sie den Anweisungen unter [Trainieren Ihres Modells](#), um Ihr Modell zu trainieren.

Erstellen eines Datensatzes mit Bildern, die in einem Amazon S3 S3-Bucket gespeichert sind

Sie können einen Datensatz mit Bildern erstellen, die in einem Amazon S3 S3-Bucket gespeichert sind. Mit dieser Option können Sie die Ordnerstruktur in Ihrem Amazon S3 S3-Bucket verwenden, um Ihre Bilder automatisch zu klassifizieren. Sie können die Bilder im Konsolen-Bucket oder einem anderen Amazon S3 S3-Bucket in Ihrem Konto speichern.

Ordner für die automatische Kennzeichnung einrichten

Während der Datensatzerstellung können Sie festlegen, dass Bildern Labelnamen zugewiesen werden, die auf dem Namen des Ordners basieren, der die Bilder enthält. Die Ordner müssen ein untergeordnetes Element des Amazon S3 S3-Ordnerpfads sein, den Sie bei der Erstellung des Datensatzes in S3-URI angeben.

Im Folgenden finden Sie den `train` Ordner für die Beispielbilder von Getting Started. Wenn Sie den Speicherort des Amazon S3 S3-Ordners als `s3-bucket/circuitboard/train/` angeben, wird den Bildern im Ordner `normal` das Label `Normal` zugewiesen. Den Bildern im Ordner `anomaly` wird das Label `Anomaly` zugewiesen. Die Namen untergeordneter Ordner werden nicht für Bilder-Labels verwendet.

```
s3-bucket
### circuitboard
```



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

Erstellen eines Datensatzes mit Bildern aus einem Amazon S3 S3-Bucket

Das folgende Verfahren erstellt einen Datensatz mit den [Klassifizierungsbeispielbildern](#), die in einem Amazon S3 S3-Bucket gespeichert sind. Um Ihre eigenen Bilder zu verwenden, erstellen Sie die unter beschriebene Ordnerstruktur [Ordner für die automatische Kennzeichnung einrichten](#).

Das Verfahren zeigt auch, wie Sie ein einzelnes Datensatzprojekt oder ein Projekt, das separate Trainings- und Testdatensätze verwendet, erstellen.

Wenn Sie sich nicht dafür entscheiden, Ihre Bilder automatisch zu beschriften, müssen Sie die Bilder erst nach der Erstellung der Datensätze beschriften. Weitere Informationen finden Sie unter [Bilder klassifizieren \(Konsole\)](#).

Note

Wenn Sie den Vorgang gerade abgeschlossen haben [Erstellen Sie Ihr Projekt](#), sollte in der Konsole Ihr Projekt-Dashboard angezeigt werden und Sie müssen die Schritte 1 bis 4 nicht ausführen.

Um einen Datensatz mit Bildern zu erstellen, die in einem Amazon S3 S3-Bucket gespeichert sind

1. Falls Sie dies noch nicht getan haben, laden Sie die Bilder für die ersten Schritte in Ihren Amazon S3 S3-Bucket hoch. Weitere Informationen finden Sie unter [Datensatz zur Bildklassifizierung](#).
2. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.

3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie auf der Seite Projekte das Projekt aus, dem Sie den Datensatz hinzufügen möchten. Die Detailseite für Ihr Projekt wird angezeigt.
5. Wählen Sie Datensatz erstellen. Die Seite Datensatz erstellen wird angezeigt.

 Tip

Wenn Sie die Anweisungen „Erste Schritte“ befolgen, wählen Sie Trainingsdatensatz und Testdatensatz erstellen.

6. Wählen Sie die Registerkarte Einzelner Datensatz oder Separate Trainings- und Testdatensätze und folgen Sie den Schritten.

Single dataset

- a. Wählen Sie im Abschnitt Datensatzkonfiguration die Option Einzelnes Dataset erstellen aus.
- b. Geben Sie die Informationen für die Schritte 7 bis 9 im Abschnitt Konfiguration der Bildquelle ein.

Separate training and test datasets

- a. Wählen Sie im Abschnitt Datensatzkonfiguration die Option Trainingsdatensatz und Testdatensatz erstellen aus.
- b. Geben Sie für Ihren Trainingsdatensatz die Informationen für die Schritte 7 bis 9 im Abschnitt Details zum Trainingsdatensatz ein.
- c. Geben Sie für Ihren Testdatensatz die Informationen für die Schritte 7 bis 9 im Abschnitt Testdatensatzdetails ein.

 Note

Ihre Trainings- und Testdatensätze können unterschiedliche Bildquellen haben.

7. Wählen Sie Bilder aus Amazon-S3-Bucket importieren aus.
8. Geben Sie unter S3-URI den Speicherort und den Ordnerpfad des Amazon-S3-Buckets ein. Ändern Sie bucket in den Namen Ihres Amazon-S3-Buckets.

- a. Wenn Sie ein einzelnes Datensatzprojekt oder einen Trainingsdatensatz erstellen, geben Sie Folgendes ein:

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

- b. Wenn Sie einen Testdatensatz erstellen, geben Sie Folgendes ein:

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

9. Wählen Sie Je nach Ordner automatisch Labels an Bilder anhängen aus.
10. Wählen Sie Datensatz erstellen. Eine Datensatzseite mit Ihren beschrifteten Bildern wird geöffnet.
11. Folgen Sie den Anweisungen unter [Trainieren Ihres Modells](#), um Ihr Modell zu trainieren.

Erstellen eines Datensatzes mithilfe einer Amazon SageMaker AI Ground Truth Manifestdatei

Eine Manifestdatei enthält Informationen über die Bilder und Bildbeschriftungen, die Sie zum Trainieren und Testen eines Modells verwenden können. Sie können eine Manifestdatei in einem Amazon S3 S3-Bucket speichern und damit einen Datensatz erstellen. Sie können Ihre eigene Manifestdatei erstellen oder eine vorhandene Manifestdatei verwenden, z. B. die Ausgabe eines Amazon SageMaker AI Ground Truth Jobs.

Themen

- [Einen Amazon Sagemaker Ground Truth Job verwenden](#)
- [Erstellen einer Manifestdatei](#)

Einen Amazon Sagemaker Ground Truth Job verwenden

Das Etikettieren von Bildern kann viel Zeit in Anspruch nehmen. Es kann beispielsweise zehn Sekunden dauern, bis eine Anomalie präzise maskiert ist. Wenn Sie Hunderte von Bildern haben, kann es mehrere Stunden dauern, sie zu beschriften. Als Alternative zur eigenen Kennzeichnung der Bilder sollten Sie Amazon SageMaker Ground Truth in Betracht ziehen.

Mit Amazon SageMaker AI Ground Truth können Sie Mitarbeiter von Amazon Mechanical Turk, einem von Ihnen ausgewählten Anbieter, oder interne, private Mitarbeiter einsetzen, um einen

beschrifteten Satz von Bildern zu erstellen. Weitere Informationen finden Sie unter [Verwenden von Amazon SageMaker AI Ground Truth zur Kennzeichnung von Daten](#).

Die Nutzung von Amazon Mechanical Turk ist kostenpflichtig. Außerdem kann es mehrere Tage dauern, bis ein Amazon Ground Truth Kennzeichnungsauftrag abgeschlossen ist. Wenn die Kosten ein Problem darstellen oder wenn Sie Ihr Modell schnell trainieren müssen, empfehlen wir Ihnen, die Amazon Lookout for Vision Vision-Konsole zu verwenden, um Ihre Bilder zu [beschriften](#).

Sie können einen Amazon SageMaker AI Ground Truth Labeling-Job verwenden, um Bilder zu kennzeichnen, die für Bildklassifizierungsmodelle und Bildsegmentierungsmodelle geeignet sind. Nach Abschluss des Jobs verwenden Sie die Ausgabe-Manifestdatei, um einen Amazon Lookout for Vision Vision-Datensatz zu erstellen.

Bildklassifizierung

Um Bilder für ein Bildklassifizierungsmodell zu kennzeichnen, erstellen Sie einen Label-Job für eine Aufgabe zur [Bildklassifizierung \(Single Label\)](#).

Bildsegmentierung

Um Bilder für ein Bildsegmentierungsmodell zu kennzeichnen, erstellen Sie einen Label-Job für eine Aufgabe zur Bildklassifizierung (Single Label). [Verketteten](#) Sie dann den Job, um einen Label-Job für eine Aufgabe zur [semantischen Bildsegmentierung](#) zu erstellen.

Sie können einen Labeling-Job auch verwenden, um eine partielle Manifestdatei für ein Bildsegmentierungsmodell zu erstellen. Sie können Bilder beispielsweise mit einer Aufgabe zur Bildklassifizierung (Single Label) klassifizieren. Nachdem Sie einen Lookout for Vision Vision-Datensatz mit der Jobausgabe erstellt haben, verwenden Sie die Amazon Lookout for Vision Vision-Konsole, um Segmentierungsmasken und Anomalie-Labels zu den Datensatzbildern hinzuzufügen.

Bilder mit Amazon SageMaker AI Ground Truth beschriften

Das folgende Verfahren zeigt, wie Bilder mit Amazon SageMaker AI Ground Truth Bildbeschriftungsaufgaben beschriftet werden. Das Verfahren erstellt eine Manifestdatei zur Bildklassifizierung und verkettet optional die Aufgabe zur Bildbeschriftung, um eine Manifestdatei für die Bildsegmentierung zu erstellen. Wenn Sie möchten, dass Ihr Projekt über einen separaten Testdatensatz verfügt, wiederholen Sie dieses Verfahren, um die Manifestdatei für den Testdatensatz zu erstellen.

So beschriften Sie Bilder mit Amazon SageMaker AI Ground Truth (Konsole)

1. Erstellen Sie einen Ground-Truth-Job für eine Aufgabe zur Bildklassifizierung (Single Label), indem Sie den Anweisungen unter [Labeling-Job erstellen \(Konsole\)](#) folgen.
 - a. Wählen Sie für Schritt 10 im Dropdownmenü „Aufgabenkategorie“ die Option „Bild“ und als Aufgabentyp „Bildklassifizierung (Einzeletikett)“ aus.
 - b. Fügen Sie für Schritt 16 im Bereich Beschriftungstool zur Bildklassifizierung (Einzeletikett) zwei Beschriftungen hinzu: normal und anomal.
2. Warten Sie, bis die Belegschaft mit der Klassifizierung Ihrer Bilder fertig ist.
3. Wenn Sie einen Datensatz für ein Bildsegmentierungsmodell erstellen, gehen Sie wie folgt vor. Andernfalls fahren Sie mit Schritt 4 fort.
 - a. Öffnen Sie in der Amazon SageMaker AI Ground Truth Konsole die Seite Labeling-Jobs.
 - b. Wählen Sie den Job aus, den Sie zuvor erstellt haben. Das Menü Actions (Aktionen) wird aktiviert.
 - c. Wählen Sie im Menü Actions (Aktionen) die Option Chain (Verketten) aus. Die Seite mit den Jobdetails wird geöffnet.
 - d. Wählen Sie unter Aufgabentyp die Option Semantische Segmentierung aus.
 - e. Wählen Sie Weiter.
 - f. Fügen Sie im Abschnitt Kennzeichnungstool für semantische Segmentierung Anomalie-Labels für jeden Anomalie-Typ hinzu, den Ihr Modell finden soll.
 - g. Wählen Sie Create (Erstellen) aus.
 - h. Warten Sie, bis die Belegschaft Ihre Bilder beschriftet hat.
4. Öffnen Sie die Ground Truth Konsole und öffnen Sie die Seite Labeling-Jobs.
5. Wenn Sie ein Bildklassifizierungsmodell erstellen, wählen Sie den Job aus, den Sie in Schritt 1 erstellt haben. Wenn Sie ein Bildsegmentierungsmodell erstellen, wählen Sie den in Schritt 3 erstellten Job aus.
6. Öffnen Sie in der Zusammenfassung des Labeling-Jobs den Speicherort S3 unter Speicherort des Ausgabe-Datensatzes. Notieren Sie sich den Speicherort der Manifestdatei, der sein sollte `s3://output-dataset-location/manifests/output/output.manifest`.
7. Wiederholen Sie dieses Verfahren, wenn Sie eine Manifestdatei für einen Testdatensatz erstellen möchten. Folgen Sie andernfalls den Anweisungen unter [Den Datensatz erstellen](#) So erstellen Sie einen Datensatz mit der Manifestdatei.

Den Datensatz erstellen

Gehen Sie wie folgt vor, um einen Datensatz in einem Lookout for Vision Vision-Projekt mit der Manifestdatei zu erstellen, die Sie in Schritt 6 von [Bilder mit Amazon SageMaker AI Ground Truth beschriften](#) notiert haben. Die Manifestdatei erstellt den Trainingsdatensatz für ein einzelnes Datensatzprojekt. Wenn Sie möchten, dass Ihr Projekt über einen separaten Testdatensatz verfügt, können Sie einen weiteren Amazon SageMaker AI Ground Truth Job ausführen, um eine Manifestdatei für den Testdatensatz zu erstellen. Oder Sie können die Manifestdatei selbst [erstellen](#). Sie können auch Bilder aus einem Amazon S3 S3-Bucket oder von Ihrem lokalen Computer in Ihren Testdatensatz importieren. (Die Bilder müssen möglicherweise beschriftet werden, bevor Sie das Modell trainieren können).

Bei diesem Verfahren wird davon ausgegangen, dass Ihr Projekt keine Datensätze enthält.

So erstellen Sie einen Datensatz mit Lookout for Vision (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie das Projekt aus, das Sie hinzufügen möchten, um es mit der Manifestdatei zu verwenden.
5. Wählen Sie im Abschnitt So funktioniert's die Option Datensatz erstellen aus.
6. Wählen Sie die Registerkarte Einzelner Datensatz oder Separate Trainings- und Testdatensätze und folgen Sie den Schritten.

Single dataset

1. Wählen Sie „Einen einzelnen Datensatz erstellen“ aus.
2. Wählen Sie im Abschnitt Konfiguration der Bildquelle die Option Bilder importieren mit der Bezeichnung SageMaker Ground Truth aus.
3. Geben Sie für den Speicherort der Manifest-Datei den Speicherort der Manifest-Datei ein, den Sie in Schritt 6 von [Bilder mit Amazon SageMaker AI Ground Truth beschriften](#) notiert haben.

Separate training and test datasets

1. Wählen Sie Trainingsdatensatz und Testdatensatz erstellen aus.
2. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Mit SageMaker Ground Truth beschriftete Bilder importieren aus.
3. Geben Sie unter .manifest-Datei den Speicherort der Manifest-Datei ein, die Sie in Schritt 6 von [Bilder mit Amazon SageMaker AI Ground Truth beschriften](#) notiert haben.
4. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Bilder importieren mit der Bezeichnung SageMaker Ground Truth aus.
5. Geben Sie unter .manifest-Datei den Speicherort der Manifest-Datei ein, die Sie in Schritt 6 von [Bilder mit Amazon SageMaker AI Ground Truth beschriften](#) notiert haben. Denken Sie daran, dass Sie eine separate Manifestdatei für den Testdatensatz benötigen.
7. Wählen Sie Absenden aus.
8. Folgen Sie den Anweisungen unter [Trainieren Ihres Modells](#), um Ihr Modell zu trainieren.

Erstellen einer Manifestdatei

Sie können einen Datensatz erstellen, indem Sie eine Manifestdatei im SageMaker AI Ground Truth Format importieren. Wenn Ihre Bilder in einem Format beschriftet sind, das keine SageMaker AI Ground Truth-Manifestdatei ist, verwenden Sie die folgenden Informationen, um eine Manifestdatei im SageMaker AI Ground Truth Format zu erstellen.

Manifestdateien haben das [JSON-Zeilen](#)-Format, wobei jede Zeile ein vollständiges JSON-Objekt ist, das die Label-Informationen für ein Bild darstellt. Es gibt verschiedene Formate für die [Bildklassifizierung](#) und [Bildsegmentierung](#). Manifestdateien müssen mit UTF-8-Kodierung codiert werden.

Note

Die Beispiele für JSON-Zeilen in diesem Abschnitt sind aus Gründen der Lesbarkeit formatiert.

Die Bilder, auf die eine Manifestdatei verweist, müssen sich in demselben Amazon-S3-Bucket befinden. Die Manifestdatei kann sich in einem anderen Bucket befinden. Sie geben die Position eines Bildes im `source-ref`-Feld einer JSON-Zeile an.

Sie können eine Manifestdatei mithilfe von Code erstellen. Das Python-Notizbuch von [Amazon Lookout for Vision Lab](#) zeigt, wie Sie eine Manifestdatei zur Bildklassifizierung für die Leiterplatten-Beispielbilder erstellen. Alternativ können Sie den [Beispielcode Datasets im Code Examples Repository](#) verwenden. AWS Sie können ganz einfach eine Manifestdatei erstellen, indem Sie eine CSV-Datei (Comma Separated Values) verwenden. Weitere Informationen finden Sie unter [Eine Klassifizierungsmanifestdatei aus einer CSV-Datei erstellen](#).

Themen

- [Definition von JSON-Zeilen für die Bildklassifizierung](#)
- [Definition von JSON-Zeilen für die Bildsegmentierung](#)
- [Eine Klassifizierungsmanifestdatei aus einer CSV-Datei erstellen](#)
- [Einen Datensatz mit einer Manifestdatei erstellen \(Konsole\)](#)
- [Einen Datensatz mit einer Manifestdatei \(SDK\) erstellen](#)

Definition von JSON-Zeilen für die Bildklassifizierung

Sie definieren eine JSON-Zeile für jedes Bild, das Sie in einer Amazon Lookout for Vision Vision-Manifestdatei verwenden möchten. Wenn Sie ein Klassifizierungsmodell erstellen möchten, muss die JSON-Zeile eine Bildklassifizierung enthalten, die entweder normal oder eine Anomalie ist. Eine JSON-Zeile hat das SageMaker AI Ground Truth [Classification Job Output-Format](#). Eine Manifestdatei besteht aus einer oder mehreren JSON-Zeilen, eine für jedes Bild, das Sie importieren möchten.

Um eine Manifestdatei für klassifizierte Bilder zu erstellen

1. Erstellen Sie eine leere Textdatei.
2. Fügen Sie eine JSON-Zeile für jedes Bild hinzu, das Sie importieren möchten. Jede JSON-Zeile sollte wie folgt aussehen:

```
{"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. Speichern Sie die Datei.

Note

Sie können die Erweiterung `.manifest` verwenden, sie ist jedoch nicht erforderlich.

4. Erstellen Sie einen Datensatz mit der von Ihnen erstellten Manifestdatei. Weitere Informationen finden Sie unter [Erstellen einer Manifestdatei](#).

Klassifizierung: JSON-Zeilen

In diesem Abschnitt erfahren Sie, wie Sie eine JSON-Zeile erstellen, die ein Bild als normal oder ungewöhnlich klassifiziert.

JSON-Zeile mit Anomalie

Die folgende JSON-Zeile zeigt ein Bild, das als Anomalie gekennzeichnet ist. Beachten Sie, dass der Wert von `class-name` `istanomaly` ist.

```
{
  "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
}
```

Normale JSON-Zeile

Die folgende JSON-Zeile zeigt ein Bild, das als normal gekennzeichnet ist. Beachten Sie, dass der Wert von `class-name` `istnormal` ist.

```
{
  "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
}
```

JSON-Zeilenschlüssel und -werte

Die folgenden Informationen beschreiben die Schlüssel und Werte in einer Amazon Lookout for Vision JSON-Zeile.

Quellennachweis

(Erforderlich) Der Amazon S3-Speicherort des Bildes. Das Format ist

"s3://**BUCKET/OBJECT_PATH**". Bilder in einem importierten Datensatz müssen im gleichen Amazon-S3-Bucket gespeichert werden.

Bezeichnung einer Anomalie

(Erforderlich) Das Label-Attribut. Verwenden Sie den Schlüssel oder einen anderen Schlüsselnamen `anomaly-label`, den Sie wählen. Der Schlüsselwert (0 im vorherigen Beispiel) wird von Amazon Lookout for Vision benötigt, aber nicht verwendet. Das von Amazon Lookout for Vision erstellte Ausgabemanifest konvertiert den Wert in 1 für ein ungewöhnliches Bild und einen Wert von 0 für ein normales Bild. Der Wert von `class-name` bestimmt, ob das Bild normal oder ungewöhnlich ist.

Es müssen entsprechende Metadaten vorhanden sein, die durch den Feldnamen mit angehängtem -Metadaten identifiziert werden. Beispiel, "anomaly-label-metadata".

anomaly-label-metadata

(Erforderlich) Metadaten zum Label-Attribut. Der Feldname muss mit dem Label-Attribut identisch sein, wobei -Metadaten angehängt ist.

Konfidenz

(Optional) Wird derzeit nicht von Amazon Lookout for Vision verwendet. Wenn Sie einen Wert angeben, verwenden Sie den Wert. 1

Jobname

(Optional) Ein Name, den Sie für den Job wählen, der das Bild verarbeitet.

Klassenname

(Erforderlich) Wenn das Bild normalen Inhalt enthält, geben Sie dies `normal`, andernfalls geben Sie es `anomaly`. Wenn der Wert von `class-name` ein anderer Wert ist, wird das Bild dem Datensatz als unbeschriftetes Bild hinzugefügt. Informationen zur Kennzeichnung eines Bilds finden Sie unter [Hinzufügen von Bildern zu Ihrem Datensatz](#).

Mit menschlichen Anmerkungen versehen

(Erforderlich) Geben Sie "yes" an, wenn die Anmerkung von einem Menschen ausgefüllt wurde. Andernfalls geben Sie "no" an.

Erstellungsdatum

(Optional) Datum und Uhrzeit der Erstellung des Labels in koordinierter Weltzeit (Coordinated Universal Time, UTC).

Typ

(Erforderlich) Die Art der Verarbeitung, die auf das Bild angewendet werden soll. Für Anomalie-Labels auf Bildebene ist der Wert. "groundtruth/image-classification"

Definition von JSON-Zeilen für die Bildsegmentierung

Sie definieren eine JSON-Zeile für jedes Bild, das Sie in einer Amazon Lookout for Vision Vision-Manifestdatei verwenden möchten. Wenn Sie ein Segmentierungsmodell erstellen möchten, muss die JSON-Zeile Segmentierungs- und Klassifizierungsinformationen für das Bild enthalten. Eine Manifestdatei besteht aus einer oder mehreren JSON-Zeilen, eine für jedes Bild, das Sie importieren möchten.

Um eine Manifestdatei für segmentierte Bilder zu erstellen

1. Erstellen Sie eine leere Textdatei.
2. Fügen Sie eine JSON-Zeile für jedes Bild hinzu, das Sie importieren möchten. Jede JSON-Zeile sollte wie folgt aussehen:

```
{
  "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. Speichern Sie die Datei.

Note

Sie können die Erweiterung `.manifest` verwenden, sie ist jedoch nicht erforderlich.

4. Erstellen Sie einen Datensatz mit der von Ihnen erstellten Manifestdatei. Weitere Informationen finden Sie unter [Erstellen einer Manifestdatei](#).

Segmentierung: JSON-Zeilen

In diesem Abschnitt erfahren Sie, wie Sie eine JSON-Zeile erstellen, die Segmentierungs- und Klassifizierungsinformationen für ein Bild enthält.

Die folgende JSON-Zeile zeigt ein Bild mit Segmentierungs- und Klassifizierungsinformationen. `anomaly-label-metadata` enthält Klassifizierungsinformationen. `anomaly-mask-ref` und `anomaly-mask-ref-metadata` enthalten Segmentierungsinformationen.

```
{
  "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"
    }
  }
}

```

JSON-Zeilenschlüssel und -werte

Die folgenden Informationen beschreiben die Schlüssel und Werte in einer Amazon Lookout for Vision JSON-Zeile.

Quellennachweis

(Erforderlich) Der Amazon S3-Speicherort des Bildes. Das Format ist

"s3://*BUCKET/OBJECT_PATH*". Bilder in einem importierten Datensatz müssen im gleichen Amazon-S3-Bucket gespeichert werden.

Bezeichnung einer Anomalie

(Erforderlich) Das Label-Attribut. Verwenden Sie den Schlüssel oder einen anderen

Schlüsselnamen `anomaly-label`, den Sie wählen. Der Schlüsselwert (1im vorherigen Beispiel) wird

von Amazon Lookout for Vision benötigt, aber nicht verwendet. Das von Amazon Lookout for Vision erstellte Ausgabemanifest konvertiert den Wert in 1 für ein ungewöhnliches Bild und einen Wert von 0 für ein normales Bild. Der Wert von `class-name` bestimmt, ob das Bild normal oder ungewöhnlich ist.

Es müssen entsprechende Metadaten vorhanden sein, die durch den Feldnamen mit angehängtem -Metadaten identifiziert werden. Beispiel, "anomaly-label-metadata".

anomaly-label-metadata

(Erforderlich) Metadaten zum Label-Attribut. Enthält Klassifizierungsinformationen. Der Feldname muss mit dem Label-Attribut identisch sein, wobei -Metadaten angehängt ist.

Konfidenz

(Optional) Wird derzeit nicht von Amazon Lookout for Vision verwendet. Wenn Sie einen Wert angeben, verwenden Sie den Wert. 1

Jobname

(Optional) Ein Name, den Sie für den Job wählen, der das Bild verarbeitet.

Klassenname

(Erforderlich) Wenn das Bild normalen Inhalt enthält, geben Sie dies `normal`, andernfalls geben Sie es `anomaly`. Wenn der Wert von `class-name` ein anderer Wert ist, wird das Bild dem Datensatz als unbeschriftetes Bild hinzugefügt. Informationen zur Kennzeichnung eines Bilds finden Sie unter [Hinzufügen von Bildern zu Ihrem Datensatz](#).

Mit menschlichen Anmerkungen versehen

(Erforderlich) Geben Sie "yes" an, wenn die Anmerkung von einem Menschen ausgefüllt wurde. Andernfalls geben Sie "no" an.

Erstellungsdatum

(Optional) Datum und Uhrzeit der Erstellung des Labels in koordinierter Weltzeit (Coordinated Universal Time, UTC).

Typ

(Erforderlich) Die Art der Verarbeitung, die auf das Bild angewendet werden soll. Verwenden Sie den Wert "groundtruth/image-classification".

anomaly-mask-ref

(Erforderlich) Der Amazon S3 S3-Speicherort des Maskenbilds. Verwenden Sie `anomaly-mask-ref` ihn als Schlüsselnamen oder verwenden Sie einen Schlüsselnamen Ihrer Wahl. Der Schlüssel muss mit `enden-ref`. Das Maskenbild muss farbige Masken für jeden Anomalietyp `internal-color-map` enthalten. Das Format ist "`s3://BUCKET/OBJECT_PATH`". Bilder in einem importierten Datensatz müssen im gleichen Amazon-S3-Bucket gespeichert werden. Das Maskenbild muss ein Bild im PNG-Format (Portable Network Graphic) sein.

anomaly-mask-ref-metadata

(Erforderlich) Segmentierungsmetadaten für das Bild. Verwenden Sie es `anomaly-mask-ref-metadata` für den Schlüsselnamen oder verwenden Sie einen Schlüsselnamen Ihrer Wahl. Der Schlüsselname muss mit `enden-ref-metadata`.

internal-color-map

(Erforderlich) Eine Farbkarte, die einzelnen Anomalietypen zugeordnet ist. Die Farben müssen mit den Farben im Maskenbild (`anomaly-mask-ref`) übereinstimmen.

key

(Erforderlich) Der Schlüssel zur Map. Der Eintrag `0` muss den Klassennamen `BACKGROUND` enthalten, der Bereiche außerhalb von Anomalien auf dem Bild darstellt.

Klassenname

(Erforderlich) Der Name des Anomalie-Typs, z. B. `Kratzer` oder `Delle`.

hexadezimale Farbe

(Erforderlich) Die Hex-Farbe für den Anomalie-Typ, z. B. `#2ca02c`. Die Farbe muss mit einer Farbe in `anomaly-mask-ref` übereinstimmen. Der Wert für den `BACKGROUND` Anomalie-Typ ist immer `#ffffff`.

Konfidenz

(Erforderlich) Wird derzeit nicht von Amazon Lookout for Vision verwendet, es ist jedoch ein Float-Wert erforderlich.

Mit menschlichen Anmerkungen versehen

(Erforderlich) Geben Sie `"yes"` an, wenn die Anmerkung von einem Menschen ausgefüllt wurde. Andernfalls geben Sie `"no"` an.

Erstellungsdatum

(Optional) Datum und Uhrzeit der Erstellung der Segmentierungsinformationen in koordinierter Weltzeit (Coordinated Universal Time, UTC).

Typ

(Erforderlich) Die Art der Verarbeitung, die auf das Bild angewendet werden soll. Verwenden Sie den Wert "groundtruth/semantic-segmentation".

Eine Klassifizierungsmanifestdatei aus einer CSV-Datei erstellen

Dieses Python-Beispielskript vereinfacht die Erstellung einer Klassifikationsmanifestdatei, indem es eine CSV-Datei (Comma Separated Values) zur Kennzeichnung von Bildern verwendet. So erstellen Sie die CSV-Datei

Eine Manifestdatei beschreibt die Bilder, die zum Trainieren eines Modells verwendet werden. Eine Manifestdatei besteht aus einer oder mehreren JSON-Zeilen. Jede JSON-Zeile beschreibt ein einzelnes Bild. Weitere Informationen finden Sie unter [Definition von JSON-Zeilen für die Bildklassifizierung](#).

Eine CSV-Datei stellt tabellarische Daten über mehrere Zeilen in einer Textdatei dar. Felder in einer Zeile werden durch ein Komma getrennt. Weitere Informationen finden Sie unter [Kommagetrennte Werte](#). Bei diesem Skript enthält jede Zeile in Ihrer CSV-Datei den S3-Speicherort eines Bilds und die Anomalieklassifizierung für das Bild (normal oder anomaly). Jede Zeile ist einer JSON-Zeile in der Manifestdatei zugeordnet.

In der folgenden CSV-Datei werden beispielsweise einige der Bilder in den [Beispielbildern](#) beschrieben.

```
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
```

Das Skript generiert JSON-Zeilen für jede Zeile. Das Folgende ist beispielsweise die JSON-Zeile für die erste Zeile (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"}}
```

Wenn Ihre CSV-Datei den Amazon S3 S3-Pfad für die Bilder nicht enthält, verwenden Sie das `--s3-path` Befehlszeilenargument, um den Amazon S3-Pfad zu den Bildern anzugeben.

Vor der Erstellung der Manifestdatei sucht das Skript nach doppelten Bildern in der CSV-Datei und nach Bildklassifizierungen, bei denen es sich nicht um `normal` oder `anomaly` handelt. Wenn duplizierte Bilder oder Fehler bei der Bildklassifizierung gefunden werden, geht das Skript wie folgt vor:

- Zeichnet den ersten gültigen Bildeintrag für alle Bilder in einer deduplizierten CSV-Datei auf.
- Zeichnet doppelte Vorkommen eines Bilds in der Fehlerdatei auf.
- Zeichnet Bildklassifizierungen auf, die nicht `normal` oder `anomaly` in der Fehlerdatei enthalten sind.
- Erstellt keine Manifestdatei.

Die Fehlerdatei enthält die Zeilennummer, in der ein doppeltes Bild oder ein Klassifizierungsfehler in der CSV-Eingabedatei gefunden wurde. Verwenden Sie die CSV-Fehlerdatei, um die CSV-Eingabedatei zu aktualisieren, und führen Sie das Skript dann erneut aus. Verwenden Sie alternativ die CSV-Datei mit Fehlern, um die deduplizierte CSV-Datei zu aktualisieren, die nur eindeutige Bildeinträge und Bilder ohne Fehler bei der Bildklassifizierung enthält. Führen Sie das Skript mit der aktualisierten deduplizierten CSV-Datei erneut aus.

Wenn in der CSV-Eingabedatei keine Duplikate oder Fehler gefunden werden, löscht das Skript die deduplizierte CSV-Bilddatei und die Fehlerdatei, da sie leer sind.

In diesem Verfahren erstellen Sie die CSV-Datei und führen das Python-Skript aus, um die Manifestdatei zu erstellen. Das Skript wurde mit Python Version 3.7 getestet.

So erstellen Sie eine Manifestdatei aus einer CSV-Datei

1. Erstellen Sie eine CSV-Datei mit den folgenden Feldern in jeder Zeile (eine Zeile pro Bild). Fügen Sie der CSV-Datei keine Kopfzeile hinzu.

Feld 1	Feld 2
Der Bildname oder der Amazon S3-Pfad des Bildes. Beispiel, s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg . Sie können keine Mischung aus Bildern mit dem Amazon S3-Pfad und Bildern ohne ihn verwenden.	Die Anomalieklassifizierung für das Bild (normal oder anomaly).

Zum Beispiel s3://s3bucket/circuitboard/train/anomaly/image_10.jpg, anomaly oder image_11.jpg, normal

2. Speichern Sie die CSV-Datei.
3. Führen Sie das folgende Python-Skript aus. Stellen Sie die folgenden Argumente bereit:
 - `csv_file` – Die CSV-Datei, die Sie in Schritt 1 erstellt haben.
 - (Optional) `--s3-path s3://path_to_folder/` – Der Amazon S3-Pfad, der den Bilddateinamen hinzugefügt werden soll (Feld 1). Verwenden Sie `--s3-path`, wenn die Bilder in Feld 1 noch keinen S3-Pfad enthalten.

```
# 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"
                  f"{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. Wenn doppelte Bilder oder Klassifizierungsfehler auftreten:
 - a. Verwenden Sie die Fehlerdatei, um die deduplizierte CSV-Datei oder die CSV-Eingabedatei zu aktualisieren.
 - b. Führen Sie das Skript erneut mit der aktualisierten deduplizierten CSV-Datei oder der aktualisierten CSV-Eingabedatei aus.
5. Wenn Sie einen Testdatensatz verwenden möchten, wiederholen Sie die Schritte 1—4, um eine Manifestdatei für Ihren Testdatensatz zu erstellen.
6. Kopieren Sie bei Bedarf die Bilder von Ihrem Computer in den Amazon S3 S3-Bucket-Pfad, den Sie in Spalte 1 der CSV-Datei (oder in der `--s3-path` Befehlszeile) angegeben haben. Um die Bilder zu kopieren, geben Sie in der Befehlszeile den folgenden Befehl ein.

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

7. Folgen Sie den Anweisungen unter [Einen Datensatz mit einer Manifestdatei erstellen \(Konsole\)](#), um einen Datensatz zu erstellen. Wenn Sie das AWS SDK verwenden, finden Sie weitere Informationen unter [Einen Datensatz mit einer Manifestdatei \(SDK\) erstellen](#).

Einen Datensatz mit einer Manifestdatei erstellen (Konsole)

Das folgende Verfahren zeigt Ihnen, wie Sie einen Trainings- oder Testdatensatz erstellen, indem Sie eine Manifestdatei im SageMaker AI-Format importieren, die in einem Amazon S3 S3-Bucket gespeichert ist.

Nachdem Sie den Datensatz erstellt haben, können Sie dem Datensatz weitere Bilder hinzufügen oder Bildern Beschriftungen hinzufügen. Weitere Informationen finden Sie unter [Hinzufügen von Bildern zu Ihrem Datensatz](#).

So erstellen Sie einen Datensatz mit einer Manifestdatei im SageMaker AI Ground Truth Format (Konsole)

1. Erstellen oder verwenden Sie eine vorhandene, mit Amazon Lookout for Vision kompatible Manifestdatei im SageMaker AI Ground Truth Format. Weitere Informationen finden Sie unter [Erstellen einer Manifestdatei](#).
2. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
3. [Erstellen Sie in einem Amazon S3 S3-Bucket einen Ordner](#) für Ihre Manifestdatei.
4. [Laden Sie Ihre Manifestdatei](#) in den Ordner hoch, den Sie gerade erstellt haben.
5. Erstellen Sie im Amazon S3 S3-Bucket einen Ordner zum Speichern Ihrer Bilder.
6. Laden Sie Ihre Bilder in den nun erstellten Ordner hoch.

 Important

Der `source-ref` Feldwert in jeder JSON-Zeile muss den Bildern im Ordner zugeordnet sein.

7. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
8. Wählen Sie Erste Schritte.
9. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
10. Wählen Sie das Projekt aus, das Sie hinzufügen möchten, um es mit der Manifestdatei zu verwenden.
11. Wählen Sie im Abschnitt So funktioniert's die Option Datensatz erstellen aus.
12. Wählen Sie die Registerkarte Einzelner Datensatz oder Separate Trainings- und Testdatensätze und folgen Sie den Schritten.

Single dataset

1. Wählen Sie „Einen einzelnen Datensatz erstellen“ aus.

2. Wählen Sie im Abschnitt Konfiguration der Bildquelle die Option Bilder importieren mit der Bezeichnung SageMaker Ground Truth aus.
3. Geben Sie für den Speicherort der Datei `.manifest` den Speicherort Ihrer Manifest-Datei ein.

Separate training and test datasets

1. Wählen Sie Trainingsdatensatz und Testdatensatz erstellen aus.
2. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Mit SageMaker Ground Truth beschriftete Bilder importieren aus.
3. Geben Sie unter Speicherort der Datei „manifest“ den Speicherort Ihrer Trainingsmanifestdatei ein.
4. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Bilder importieren mit der Bezeichnung SageMaker Ground Truth aus.
5. Geben Sie unter Speicherort der Datei „manifest“ den Speicherort Ihrer Testmanifestdatei ein.



Note

Ihre Trainings- und Testdatensätze können unterschiedliche Bildquellen haben.

13. Wählen Sie Absenden aus.
14. Folgen Sie den Anweisungen unter [Trainieren Ihres Modells](#), um Ihr Modell zu trainieren.

Amazon Lookout for Vision erstellt einen Datensatz im Amazon S3 `datasets` S3-Bucket-Ordner. Ihre `.manifest` Originaldatei bleibt unverändert.

Einen Datensatz mit einer Manifestdatei (SDK) erstellen

Sie verwenden den [CreateDataset](#) Vorgang, um die Datensätze zu erstellen, die mit einem Amazon Lookout for Vision Vision-Projekt verknüpft sind.

Wenn Sie einen einzelnen Datensatz für Schulungen und Tests verwenden möchten, erstellen Sie einen einzelnen Datensatz mit dem `DatasetType` Wert auf `train`. Während des Trainings wird der Datensatz intern aufgeteilt, sodass ein Trainings- und Testdatensatz entsteht. Sie haben keinen Zugriff auf die geteilten Trainings- und Testdatensätze. Wenn Sie einen separaten Testdatensatz benötigen, rufen Sie `CreateDataset` with the `DatasetType` value set `test` ein zweites Mal auf.

Während des Trainings werden die Trainings- und Testdatensätze verwendet, um das Modell zu trainieren und zu testen.

Sie können den `DatasetSource` Parameter optional verwenden, um den Speicherort einer Manifestdatei im SageMaker AI Ground Truth Format anzugeben, die zum Auffüllen des Datensatzes verwendet wird. In diesem Fall erfolgt der Aufruf von `asynchron. CreateDataset` Um den aktuellen Status zu überprüfen, rufen Sie `DescribeDataset` auf. Weitere Informationen finden Sie unter [Ihre Datensätze anzeigen](#). Wenn beim Import ein Validierungsfehler auftritt, Status wird der Wert von `CREATE_FAILED` gesetzt und die Statusmeldung (`StatusMessage`) wird gesetzt.

Tip

Wenn Sie einen Datensatz mit dem Beispieldatensatz [Erste Schritte](#) erstellen, verwenden Sie die Manifestdatei (`getting-started/dataset-files/manifests/train.manifest`), in der das Skript erstellt. [Schritt 1: Erstellen Sie die Manifest-Datei und laden Sie Bilder hoch](#) Wenn Sie einen Datensatz mit den [Leiterplatten-Beispielbildern](#) erstellen, haben Sie zwei Möglichkeiten:

1. Erstellen Sie die Manifestdatei mithilfe von Code. Das Python-Notizbuch von [Amazon Lookout for Vision Lab](#) zeigt, wie die Manifestdatei für die Leiterplatten-Beispielbilder erstellt wird. Verwenden Sie alternativ den [Beispielcode Datasets im Code Examples Repository](#). AWS
2. Wenn Sie die Amazon Lookout for Vision-Konsole bereits verwendet haben, um einen Datensatz mit den Leiterplatten-Beispielbildern zu erstellen, verwenden Sie die von Amazon Lookout for Vision für Sie erstellten Manifestdateien erneut. Die Speicherorte der Trainings- und Testmanifestdateien sind. `s3://bucket/datasets/project name/train or test/manifests/output/output.manifest`

Wenn Sie nichts angeben `DatasetSource`, wird ein leerer Datensatz erstellt. In diesem Fall erfolgt der Aufruf von `synchron. CreateDataset` Später können Sie dem Datensatz durch Aufrufen `UpdateDatasetEntries` eine Bezeichnung für Bilder hinzufügen. Beispielcode finden Sie unter [Weitere Bilder hinzufügen \(SDK\)](#).

Wenn Sie einen Datensatz ersetzen möchten, löschen Sie zuerst den vorhandenen Datensatz durch [DeleteDataset](#) und erstellen Sie dann einen neuen Datensatz desselben Datentyps, indem Sie `asynchron. CreateDataset` aufrufen. Weitere Informationen finden Sie unter [Löschen eines Datensatzes](#).

Nachdem Sie die Datensätze erstellt haben, können Sie das Modell erstellen. Weitere Informationen finden Sie unter [Ein Modell trainieren \(SDK\)](#).

Sie können die beschrifteten Bilder (JSON-Zeilen) innerhalb eines Datensatzes anzeigen, indem Sie Folgendes aufrufen [ListDatasetEntries](#). Sie können beschriftete Bilder hinzufügen, indem Sie aufrufen `UpdateDatasetEntries`.

Informationen zu den Test- und Trainingsdatensätzen finden Sie unter [Ihre Datensätze anzeigen](#).

So erstellen Sie einen Datensatz (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um einen Datensatz zu erstellen.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, dem Sie den Datensatz zuordnen möchten.
- `dataset-type` auf den Datensatztyp, den Sie erstellen möchten (`train` oder `test`).
- `dataset-source` zum Amazon S3 S3-Speicherort der Manifestdatei.
- `Bucket` auf den Namen des Amazon S3 S3-Buckets, der die Manifestdatei enthält.
- `Key` zum Pfad und Dateinamen der Manifestdatei im Amazon S3 S3-Bucket.

```
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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem GitHub Repository mit den Beispielen für das AWS Documentation SDK. Das vollständige Beispiel finden Sie [hier](#).

```

/**
 * Creates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvcClient    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. Trainieren Sie Ihr Modell, indem Sie den Schritten unter folgen [Ein Modell trainieren \(SDK\)](#).

Labeling von Bildern

Sie können die Amazon Lookout for Vision Vision-Konsole verwenden, um die den Bildern in Ihrem Datensatz zugewiesenen Beschriftungen hinzuzufügen oder zu ändern. Wenn Sie das SDK verwenden, sind die Labels Teil der Manifestdatei, für die Sie Daten bereitstellen. [CreateDataset](#) Sie können die Labels für ein Bild aktualisieren, indem Sie aufrufen [UpdateDatasetEntries](#). Beispielcode finden Sie unter [Weitere Bilder hinzufügen \(SDK\)](#).

Auswahl des Modelltyps

Die Beschriftungen, die Sie Bildern zuweisen, bestimmen den [Modelltyp](#), den Lookout for Vision erstellt. Wenn Ihr Projekt über einen separaten Testdatensatz verfügt, beschriften Sie die Bilder auf dieselbe Weise.

Modell zur Bildklassifizierung

Um ein Bildklassifizierungsmodell zu erstellen, klassifizieren Sie jedes Bild als normal oder anomal. Tun Sie dies für jedes Bild. [Bilder klassifizieren \(Konsole\)](#)

Modell der Bildsegmentierung

Um ein Bildsegmentierungsmodell zu erstellen, klassifizieren Sie jedes Bild als normal oder anomal. Für jedes anomale Bild geben Sie außerdem eine Pixelmaske für jeden anomalen Bereich im Bild

und eine Anomaliebezeichnung für den Typ der Anomalie innerhalb der Pixelmaske an. Die blaue Maske in der folgenden Abbildung markiert beispielsweise die Position einer Kratzanomalie an einem Auto. Sie können in einem Bild mehr als einen Typ von Anomaliekennzeichnung angeben. Tun [Bilder segmentieren \(Konsole\)](#) Sie dies für jedes Bild.



Bilder klassifizieren (Konsole)

Sie verwenden die Lookout for Vision Vision-Konsole, um Bilder in einem Datensatz als normal oder als Anomalie zu klassifizieren. Unklassifizierte Bilder werden nicht zum Trainieren Ihres Modells verwendet.

Wenn Sie ein Bildsegmentierungsmodell erstellen, überspringen Sie dieses Verfahren und tun Sie es [Bilder segmentieren \(Konsole\)](#), das Schritte zur Klassifizierung von Bildern beinhaltet.

Note

Wenn Sie den Vorgang gerade abgeschlossen haben [Erstellen Sie Ihren Datensatz](#), sollte die Konsole derzeit Ihr Modell-Dashboard anzeigen und Sie müssen die Schritte 1 bis 4 nicht ausführen.

Um deine Bilder zu klassifizieren (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
3. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten.
4. Wählen Sie im linken Navigationsbereich Ihres Projekts die Option Datensatz aus.

5. Wenn Sie separate Trainings- und Testdatensätze haben, wählen Sie die Registerkarte für den Datensatz aus, den Sie verwenden möchten.
6. Wählen Sie mit der Kennzeichnung beginnen aus.
7. Wählen Sie Alle Bilder auf dieser Seite auswählen.
8. Wenn die Bilder normal sind, wählen Sie „Als normal klassifizieren“, andernfalls „Als Anomalie klassifizieren“. Unter jedem Bild wird eine Bezeichnung angezeigt.
9. Wenn Sie die Bezeichnung für ein Bild ändern müssen, gehen Sie wie folgt vor:
 - a. Wählen Sie unter dem Bild „Anomalie“ oder „Normal“.
 - b. Wenn Sie die richtige Bezeichnung für ein Bild nicht ermitteln können, vergrößern Sie das Bild, indem Sie das Bild in der Galerie auswählen.

Note

Sie können Bildbeschriftungen filtern, indem Sie im Abschnitt Filter die gewünschte Bezeichnung oder den gewünschten Labelstatus auswählen.

10. Wiederholen Sie bei Bedarf die Schritte 7-9 auf jeder Seite, bis alle Bilder im Datensatz korrekt beschriftet sind.
11. Wählen Sie Änderungen speichern.
12. Wenn Sie mit dem Beschriften Ihrer Bilder fertig sind, können Sie Ihr Modell [trainieren](#).

Bilder segmentieren (Konsole)

Wenn Sie ein Bildsegmentierungsmodell erstellen, müssen Sie Bilder als normal oder anomal klassifizieren. Sie müssen auch Segmentierungsinformationen zu anomalen Bildern hinzufügen. Um Segmentierungsinformationen anzugeben, geben Sie zunächst Anomaliebezeichnungen für jede Art von Anomalie an, z. B. eine Delle oder einen Kratzer, die Ihr Modell finden soll. Anschließend geben Sie für jede Anomalie auf anomalen Bildern in Ihrem Datensatz eine Anomalienmaske und eine Anomaliebezeichnung an.

Note

Wenn Sie ein Bildklassifizierungsmodell erstellen, müssen Sie keine Bilder segmentieren und Sie müssen keine Anomalie-Labels angeben.

Themen

- [Spezifizieren von Bezeichnungen für Anomalien](#)
- [Ein Bild beschriften](#)
- [Segmentieren eines Bilds mit dem Annotationstool](#)

Spezifizieren von Bezeichnungen für Anomalien

Sie definieren ein Anomalie-Label für jede Art von Anomalie, die in den Datensatzbildern vorkommt.

Geben Sie Anomaliebezeichnungen an

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
3. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten.
4. Wählen Sie im linken Navigationsbereich Ihres Projekts die Option Datensatz aus.
5. Wählen Sie unter Anomalie-Labels die Option Anomalie-Labels hinzufügen aus. Wenn Sie zuvor ein Anomalie-Label hinzugefügt haben, wählen Sie Verwalten aus.
6. Führen Sie im Dialogfeld folgende Schritte aus:
 - a. Geben Sie das Anomalie-Label ein, das Sie hinzufügen möchten, und wählen Sie Anomalie-Label hinzufügen aus.
 - b. Wiederholen Sie den vorherigen Schritt, bis Sie alle Anomalie-Labels eingegeben haben, die Ihr Modell finden soll.
 - c. (Optional) Wählen Sie das Bearbeitungssymbol, um den Labelnamen zu ändern.
 - d. (Optional) Wählen Sie das Symbol „Löschen“, um ein neues Anomalie-Label zu löschen. Sie können keine Anomalietypen löschen, die Ihr Datensatz derzeit verwendet.
7. Wählen Sie Bestätigen, um die neuen Anomalie-Labels zum Datensatz hinzuzufügen.

Nachdem Sie die Anomalie-Labels angegeben haben, beschriften Sie die Bilder wie folgt. [Ein Bild beschriften](#)

Ein Bild beschriften

Um ein Bild für die Bildsegmentierung zu kennzeichnen, klassifizieren Sie das Bild als normal oder als Anomalie. Verwenden Sie dann das Annotationstool, um das Bild zu segmentieren, indem Sie Masken zeichnen, die die Bereiche für jede Art von Anomalie im Bild dicht abdecken.

Um ein Bild zu beschriften

1. Wenn Sie separate Trainings- und Testdatensätze haben, wählen Sie die Registerkarte für den Datensatz, den Sie verwenden möchten.
2. Falls Sie dies noch nicht getan haben, geben Sie die Anomalietypen für Ihren Datensatz an, indem Sie [Spezifizieren von Bezeichnungen für Anomalien](#)
3. Wählen Sie mit der Kennzeichnung beginnen aus.
4. Wählen Sie Alle Bilder auf dieser Seite auswählen.
5. Wenn die Bilder normal sind, wählen Sie „Als normal klassifizieren“, andernfalls „Als Anomalie klassifizieren“.
6. Um die Bezeichnung für ein einzelnes Bild zu ändern, wählen Sie unter dem Bild „Normal“ oder „Anomalie“ aus.

Note

Sie können Bildbeschriftungen filtern, indem Sie im Abschnitt Filter die gewünschte Bezeichnung oder den gewünschten Labelstatus auswählen. Im Bereich Sortieroptionen können Sie nach dem Konfidenzwert sortieren.

7. Wählen Sie für jedes anomale Bild das Bild aus, um das Annotationstool zu öffnen. Fügen Sie Segmentierungsinformationen hinzu, indem Sie [Segmentieren eines Bilds mit dem Annotationstool](#)
8. Wählen Sie Änderungen speichern.
9. Wenn Sie mit der Kennzeichnung Ihrer Bilder fertig sind, können Sie Ihr Modell [trainieren](#).

Segmentieren eines Bilds mit dem Annotationstool

Sie verwenden das Annotationstool, um ein Bild zu segmentieren, indem Sie anomale Bereiche mit einer Maske markieren.

Um ein Bild mit dem Annotationstool zu segmentieren

1. Öffnen Sie das Annotationstool, indem Sie das Bild in der Datensatz-Galerie auswählen. Wählen Sie bei Bedarf Beschriftung starten, um in den Beschriftungsmodus zu wechseln.
2. Wählen Sie im Abschnitt Anomalie-Labels das Anomalie-Label aus, das Sie markieren möchten. Wählen Sie bei Bedarf die Option Anomalie-Labels hinzufügen aus, um ein neues Anomalie-Label hinzuzufügen.
3. Wählen Sie unten auf der Seite ein Zeichenwerkzeug aus und zeichnen Sie Masken, die anomale Bereiche dicht abdecken, um die Anomaliebeschriftung zu erstellen. Die folgende Abbildung zeigt ein Beispiel für eine Maske, die eine Anomalie dicht bedeckt.



Das Folgende ist ein Beispiel für eine schlechte Maske, die eine Anomalie nicht dicht verdeckt.



4. Wenn Sie mehr Bilder segmentieren müssen, wählen Sie Weiter und wiederholen Sie die Schritte 2 und 3.
5. Wählen Sie „Senden“ und „Schließen“, um die Segmentierung der Bilder abzuschließen.

Trainieren Ihres Modells

Nachdem Sie Ihre Datensätze erstellt und die Bilder beschriftet haben, können Sie Ihr Modell trainieren. Im Rahmen des Trainingsprozesses wird ein Testdatensatz verwendet. Wenn Sie ein einzelnes Datensatzprojekt haben, werden die Bilder im Datensatz im Rahmen des Trainingsprozesses automatisch in einen Testdatensatz und einen Trainingsdatensatz aufgeteilt. Wenn Ihr Projekt über einen Trainings- und einen Testdatensatz verfügt, werden diese verwendet, um den Datensatz separat zu trainieren und zu testen.

Nach Abschluss des Trainings können Sie die Leistung des Modells bewerten und gegebenenfalls Verbesserungen vornehmen. Weitere Informationen finden Sie unter [Verbesserung Ihres Amazon Lookout for Vision Vision-Modells](#).

Um Ihr Modell zu trainieren, erstellt Amazon Lookout for Vision eine Kopie Ihrer ursprünglichen Trainings- und Testbilder. Standardmäßig werden die kopierten Bilder mit einem Schlüssel verschlüsselt, den AWS besitzt und verwaltet. Sie können sich auch dafür entscheiden, Ihren eigenen

AWS Key Management Service (KMS) -Schlüssel zu verwenden. Weitere Informationen finden Sie unter [AWS Key Management Service-Konzepte](#). Ihre Quellbilder sind davon nicht betroffen.

Sie können Ihrem Modell Metadaten in Form von Tags zuweisen. Weitere Informationen finden Sie unter [Modelle taggen](#).

Jedes Mal, wenn Sie ein Modell trainieren, wird eine neue Version des Modells erstellt. Wenn Sie eine Version eines Modells nicht mehr benötigen, können Sie sie löschen. Weitere Informationen finden Sie unter [Löschen eines Modells](#).

Ihnen wird die Zeit in Rechnung gestellt, die benötigt wird, um Ihr Modell erfolgreich zu trainieren. Weitere Informationen finden Sie unter [Trainingszeiten](#).

Um die vorhandenen Modelle in einem Projekt anzuzeigen, [Deine Modelle ansehen](#).

Note

Wenn Sie gerade abgeschlossen haben [Erstellen Sie Ihren Datensatz](#) oder [Hinzufügen von Bildern zu Ihrem Datensatz](#). Auf der Konsole sollte derzeit Ihr Modell-Dashboard angezeigt werden und Sie müssen die Schritte 1 bis 4 nicht ausführen.

Themen

- [Ein Modell \(Konsole\) trainieren](#)
- [Ein Modell trainieren \(SDK\)](#)

Ein Modell (Konsole) trainieren

Das folgende Verfahren zeigt Ihnen, wie Sie Ihr Modell mit der Konsole trainieren.

So trainieren Sie Ihr Modell (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
3. Wählen Sie auf der Seite Projekte das Projekt aus, das das zu trainierende Modell enthält.
4. Wählen Sie auf der Seite mit den Projektdetails die Option Train Model aus. Die Schaltfläche Modell trainieren ist verfügbar, wenn Sie über genügend beschriftete Bilder verfügen, um das

- Modell zu trainieren. Wenn die Schaltfläche nicht verfügbar ist, [fügen Sie weitere Bilder](#) hinzu, bis Sie genügend beschriftete Bilder haben.
5. (Optional) Wenn Sie Ihren eigenen AWS KMS-Verschlüsselungsschlüssel verwenden möchten, gehen Sie wie folgt vor:
 - a. Wählen Sie unter Bilddatenverschlüsselung die Option Verschlüsselungseinstellungen anpassen (erweitert) aus.
 - b. Geben Sie in `encryption.aws_kms_key` den Amazon-Ressourcennamen (ARN) Ihres Schlüssels ein, oder wählen Sie einen vorhandenen AWS-KMS-Schlüssel aus. Um einen neuen Schlüssel zu erstellen, wählen Sie `AWS KMS-Schlüssel erstellen`.
 6. (Optional) Gehen Sie wie folgt vor, wenn Sie Ihr Modell mit Tags versehen möchten:
 - a. Klicken Sie im Abschnitt Tags auf `Neuen Tag hinzufügen`.
 - b. Geben Sie Folgendes ein:
 - i. Den Namen des Schlüssels im Feld `Schlüssel`.
 - ii. Den Wert des Schlüssels im Feld `Wert`.
 - c. Um weitere Tags hinzuzufügen, wiederholen Sie die Schritte 6a und 6b.
 - d. (Optional) Wenn Sie ein Tag entfernen möchten, wählen Sie neben dem Tag, das Sie entfernen möchten, die Option `Entfernen` aus. Wenn Sie ein zuvor gespeichertes Tag entfernen, wird es entfernt, wenn Sie Ihre Änderungen speichern.
 7. Wählen Sie `Train Model`.
 8. Wählen Sie im Dialogfenster `Möchten Sie Ihr Modell trainieren?` `Modell trainieren` aus.
 9. In der Modellansicht können Sie sehen, dass das Training begonnen hat, und Sie können den aktuellen Status überprüfen, indem Sie sich die `Status` Spalte für die Modellversion ansehen. Das Trainieren eines Modells ist zeitaufwändig.
 10. Wenn das Training abgeschlossen ist, können Sie seine Leistung bewerten. Weitere Informationen finden Sie unter [Verbesserung Ihres Amazon Lookout for Vision Vision-Modells](#).

Ein Modell trainieren (SDK)

Sie verwenden die [CreateModel](#) Operation, um mit dem Training, dem Testen und der Evaluierung eines Modells zu beginnen. Amazon Lookout for Vision trainiert das Modell anhand des mit dem Projekt verknüpften Trainings- und Testdatensatzes. Weitere Informationen finden Sie unter [Erstellen eines Projekts \(SDK\)](#).

Bei jedem Aufruf `CreateModel` wird eine neue Version des Modells erstellt. Die Antwort von `CreateModel` enthält die Version des Modells.

Jede erfolgreiche Modellschulung wird Ihnen in Rechnung gestellt. Verwenden Sie den `ClientToken` Eingabeparameter, um Gebühren aufgrund unnötiger oder versehentlicher Wiederholungen des Modelltrainings durch Ihre Benutzer zu vermeiden. `ClientToken` ist ein idempotenter Eingabeparameter, der sicherstellt, dass für einen bestimmten Parametersatz `CreateModel` nur einmal abgeschlossen wird. Ein wiederholter Aufruf von `CreateModel` mit demselben `ClientToken` Wert stellt sicher, dass das Training nicht wiederholt wird. Wenn Sie keinen Wert für `angebenClientToken`, fügt das von Ihnen verwendete AWS-SDK einen Wert für Sie ein. Dadurch wird verhindert, dass bei erneuten Versuchen nach einem Netzwerkfehler mehrere Trainingsjobs gestartet werden. Sie müssen jedoch Ihren eigenen Wert für Ihre eigenen Anwendungsfälle angeben. Weitere Informationen finden Sie unter [CreateModel](#).

Es dauert eine Weile, bis das Training abgeschlossen ist. Um den aktuellen Status zu überprüfen, rufen Sie `DescribeModel` den Projektnamen (der im Call to angegeben wurde `CreateProject`) und die Modellversion an und übergeben Sie sie. Das `status` Feld zeigt den aktuellen Status des Modelltrainings an. Beispielcode finden Sie unter [Ihre Modelle anzeigen \(SDK\)](#).

Wenn das Training erfolgreich ist, können Sie das Modell auswerten. Weitere Informationen finden Sie unter [Verbesserung Ihres Amazon Lookout for Vision Vision-Modells](#).

Rufen Sie an, um sich die Modelle anzusehen, die Sie in einem Projekt erstellt haben `ListModels`. Beispielcode finden Sie unter [Deine Modelle ansehen](#).

So trainieren Sie ein Modell (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um ein Modell zu trainieren.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das das Modell enthält, das Sie erstellen möchten.
- `output-config` zu dem Speicherort, an dem Sie die Trainingsergebnisse speichern möchten. Ersetzen Sie die folgenden Werte:

- `output_bucket` mit dem Namen des Amazon S3 S3-Buckets, in dem Amazon Lookout for Vision die Trainingsergebnisse speichert.
- `output_folder` mit dem Namen des Ordners, in dem Sie die Trainingsergebnisse speichern möchten.
- `Key` mit dem Namen eines Tag-Schlüssels.
- `Value` mit einem Wert, dem zugeordnet werden soll `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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem GitHub Repository mit den Beispielen für das AWS Documentation SDK. Das vollständige Beispiel finden Sie [hier](#).

```

/**
 * 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. Wenn die Schulung abgeschlossen ist, können Sie ihre Leistung bewerten. Weitere Informationen finden Sie unter [Verbesserung Ihres Amazon Lookout for Vision Vision-Modells](#).

Problembehandlung beim Modelltraining

Probleme mit Ihrer Manifestdatei oder Ihren Trainingsbildern können dazu führen, dass das Modelltraining fehlschlägt. Bevor Sie Ihr Modell erneut trainieren, überprüfen Sie die folgenden potenziellen Probleme.

Die Farben der Beschriftungen für Anomalien stimmen nicht mit der Farbe der Anomalien im Maskenbild überein

Wenn Sie ein Bildsegmentierungsmodell trainieren, muss die Farbe des Anomalie-Labels in der Manifestdatei mit der Farbe im Maskenbild übereinstimmen. Die JSON-Zeile für ein Bild in der Manifestdatei enthält Metadaten (`internal-color-map`), die Amazon Lookout for Vision mitteilen, welche Farbe einem Anomalie-Label entspricht. Die Farbe für das `scratch` Anomalie-Label in der folgenden JSON-Zeile lautet beispielsweise `#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"
}

```

Wenn die Farben im Maskenbild nicht mit den Werten in übereinstimmen `hex-color`, schlägt das Training fehl und Sie müssen die Manifestdatei aktualisieren.

Um die Farbwerte in einer Manifestdatei zu aktualisieren

1. Öffnen Sie mit einem Texteditor die Manifestdatei, mit der Sie den Datensatz erstellt haben.
2. Überprüfen Sie für jede JSON-Zeile (Bild), ob die Farben (`hex-color`) innerhalb des `internal-color-map` Felds mit den Farben für die Anomaliebezeichnungen im Maskenbild übereinstimmen.

Sie können die Position des Maskenbilds aus dem `anomaly-mask-ref` Feld ermitteln. Laden Sie das Bild auf Ihren Computer herunter und verwenden Sie den folgenden Code, um die Farben in einem Bild zu ermitteln.

```

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. Aktualisieren Sie für jedes Bild mit einer falschen Farbzuzuweisung das `hex-color` Feld in der JSON-Zeile für das Bild.
4. Speichern Sie die Aktualisierungsmanifestdatei.
5. [Löschen Sie](#) den vorhandenen Datensatz aus dem Projekt.
6. [Erstellen Sie](#) einen neuen Datensatz im Projekt mit der aktualisierten Manifestdatei.

7. [Trainieren](#) Sie das Modell.

Alternativ können Sie in den Schritten 5 und 6 einzelne Bilder im Datensatz aktualisieren, indem Sie den [UpdateDatasetEntries](#)Vorgang aufrufen und aktualisierte JSON-Zeilen für die Bilder angeben, die Sie aktualisieren möchten. Beispielpcode finden Sie unter [Weitere Bilder hinzufügen \(SDK\)](#).

Maskenbilder liegen nicht im PNG-Format vor

Wenn Sie ein Bildsegmentierungsmodell trainieren, müssen die Maskenbilder im PNG-Format vorliegen. Wenn Sie einen Datensatz aus einer Manifestdatei erstellen, stellen Sie sicher, dass die Maskenbilder, auf die Sie verweisen, im PNG-Format *anomaly-mask-ref* sind. Wenn die Maskenbilder nicht im PNG-Format vorliegen, müssen Sie sie in das PNG-Format konvertieren. Es reicht nicht aus, die Erweiterung für eine Bilddatei in umzubenennen .png.

Maskenbilder, die Sie in der Amazon Lookout for Vision Vision-Konsole oder mit einem SageMaker AI Ground Truth Job erstellen, werden im PNG-Format erstellt. Sie müssen das Format dieser Bilder nicht ändern.

Um Maskenbilder in einer Manifestdatei zu korrigieren, die nicht im PNG-Format sind

1. Öffnen Sie mit einem Texteditor die Manifestdatei, mit der Sie den Datensatz erstellt haben.
2. Stellen Sie für jede JSON-Zeile (Bild) sicher, dass das Bild *anomaly-mask-ref* auf ein Bild im PNG-Format verweist. Weitere Informationen finden Sie unter [Erstellen einer Manifestdatei](#).
3. Speichern Sie die aktualisierte Manifestdatei.
4. [Löschen Sie](#) den vorhandenen Datensatz aus dem Projekt.
5. [Erstellen Sie](#) einen neuen Datensatz im Projekt mit der aktualisierten Manifestdatei.
6. [Trainieren](#) Sie das Modell.

Segmentierungs- oder Klassifizierungsbezeichnungen sind ungenau oder fehlen

Fehlende oder ungenaue Bezeichnungen können dazu führen, dass das Training fehlschlägt oder ein Modell entsteht, das schlecht abschneidet. Wir empfehlen, dass Sie alle Bilder in Ihrem Datensatz beschriften. Wenn Sie nicht alle Bilder beschriften und das Modelltraining fehlschlägt oder Ihr Modell schlecht abschneidet, fügen Sie weitere Bilder hinzu.

Überprüfen Sie, ob Folgendes der Fall ist:

- Wenn Sie ein Segmentierungsmodell erstellen, müssen Masken die Anomalien auf Ihren Datensatzbildern gut abdecken. Um die Masken in Ihrem Datensatz zu überprüfen, sehen Sie sich die Bilder in der Datensatz-Galerie des Projekts an. Falls erforderlich, zeichnen Sie die Bildmasken neu. Weitere Informationen finden Sie unter [Bilder segmentieren \(Konsole\)](#).
- Stellen Sie sicher, dass anomale Bilder in Ihren Datensatzbildern klassifiziert sind. Wenn Sie ein Bildsegmentierungsmodell erstellen, stellen Sie sicher, dass anomale Bilder über Anomaliebezeichnungen und Bildmasken verfügen.

Es ist wichtig, dass Sie sich daran erinnern, welche Art von Modell ([Segmentierung](#) oder [Klassifizierung](#)) Sie erstellen. Ein Klassifikationsmodell benötigt keine Bildmasken für anomale Bilder. Fügen Sie Datensatzbildern, die für ein Klassifikationsmodell bestimmt sind, keine Masken hinzu.

Um fehlende Beschriftungen zu aktualisieren

1. [Öffnen Sie](#) die Datensatz-Galerie des Projekts.
2. Filtern Sie Bilder ohne Etikett, um zu sehen, welche Bilder keine Beschriftungen haben.
3. Führen Sie eine der folgenden Aktionen aus:
 - Wenn Sie ein Bildklassifizierungsmodell erstellen, [klassifizieren](#) Sie jedes unbeschriftete Bild.
 - Wenn Sie ein Bildsegmentierungsmodell erstellen, [klassifizieren und segmentieren](#) Sie jedes unbeschriftete Bild.
4. Wenn Sie ein Bildsegmentierungsmodell erstellen, [fügen Sie](#) Masken zu allen klassifizierten anomalen Bildern hinzu, bei denen Masken fehlen.
5. [Trainieren](#) Sie das Modell.

Wenn Sie schlechte oder fehlende Beschriftungen nicht korrigieren möchten, empfehlen wir Ihnen, weitere beschriftete Bilder hinzuzufügen oder die betroffenen Bilder aus dem Datensatz zu entfernen. Sie können mehr über die Konsole oder mithilfe des [UpdateDatasetEntries](#) Vorgangs hinzufügen. Weitere Informationen finden Sie unter [Hinzufügen von Bildern zu Ihrem Datensatz](#).

Wenn Sie die Bilder entfernen möchten, müssen Sie den Datensatz ohne die betroffenen Bilder neu erstellen, da Sie kein Bild aus einem Datensatz löschen können. Weitere Informationen finden Sie unter [Bilder aus Ihrem Datensatz entfernen](#).

Verbesserung Ihres Amazon Lookout for Vision Vision-Modells

Während des Trainings testet Lookout for Vision Ihr Modell mit dem Testdatensatz und verwendet die Ergebnisse, um Leistungskennzahlen zu erstellen. Sie können Leistungsmetriken verwenden, um die Leistung Ihres Modells zu bewerten. Bei Bedarf können Sie Maßnahmen ergreifen, um Ihre Datensätze zu verbessern, und anschließend Ihr Modell neu trainieren.

Wenn Sie mit der Leistung Ihres Modells zufrieden sind, können Sie damit beginnen, es zu verwenden. Weitere Informationen finden Sie unter [Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells](#).

Themen

- [Schritt 1: Bewerten Sie die Leistung Ihres Modells](#)
- [Schritt 2: Verbessern Sie Ihr Modell](#)
- [Anzeigen von Leistungsmetriken](#)
- [Verifizierung Ihres Modells mit einer Aufgabe zur Versuchserkennung](#)

Schritt 1: Bewerten Sie die Leistung Ihres Modells

Sie können von der Konsole und vom [DescribeModel](#)-Vorgang aus auf die Leistungskennzahlen zugreifen. Amazon Lookout for Vision bietet zusammenfassende Leistungskennzahlen für den Testdatensatz und die prognostizierten Ergebnisse für alle einzelnen Bilder. Wenn es sich bei Ihrem Modell um ein Segmentierungsmodell handelt, zeigt die Konsole auch zusammenfassende Kennzahlen für jedes Anomalie-Label an.

Informationen zur Anzeige der Leistungskennzahlen und Prognosen für Testbilder in der Konsole finden Sie unter [Leistungsmetriken anzeigen \(Konsole\)](#). Informationen zum Zugriff auf Leistungsmetriken und Test-Image-Vorhersagen im Rahmen des `DescribeModel`-Vorgangs finden Sie unter [Leistungskennzahlen \(SDK\) anzeigen](#).

Metriken zur Bildklassifizierung

Amazon Lookout for Vision bietet die folgenden zusammenfassenden Kennzahlen für die Klassifizierungen, die ein Modell beim Testen vornimmt:

- [Genauigkeit](#)
- [Wiedererkennung](#)
- [F1-Punktzahl](#)

Modellmetriken zur Bildsegmentierung

Wenn es sich bei dem Modell um ein Bildsegmentierungsmodell handelt, bietet Amazon Lookout for Vision zusammenfassende [Bildklassifizierungsmetriken](#) und zusammenfassende Leistungskennzahlen für jedes Anomalie-Label:

- [F1-Punktzahl](#)
- [Durchschnittliche Schnittmenge über Union \(IoU\)](#)

Genauigkeit

Die Präzisionsmetrik beantwortet die Frage: Wenn das Modell vorhersagt, dass ein Bild eine Anomalie enthält, wie oft ist diese Vorhersage richtig?

Präzision ist eine nützliche Kennzahl für Situationen, in denen die Kosten eines falsch positiven Ergebnisses hoch sind. Zum Beispiel die Kosten für das Entfernen eines Maschinenteils, das nicht defekt ist, aus einer zusammengebauten Maschine.

Amazon Lookout for Vision bietet einen zusammenfassenden Präzisionsmetrikwert für den gesamten Testdatensatz.

Präzision ist der Anteil der korrekt vorhergesagten Anomalien (wahre positive Ergebnisse) an allen vorhergesagten Anomalien (wahr und falsch positiv). Die Formel für Präzision lautet wie folgt.

Genauigkeitswert = wahre positive Ergebnisse / (wahre positive + falsch positive Ergebnisse)

Die möglichen Werte für die Genauigkeit liegen zwischen 0 und 1. Die Amazon Lookout for Vision Vision-Konsole zeigt die Genauigkeit als Prozentwert (0—100) an.

Ein höherer Genauigkeitswert bedeutet, dass mehr der vorhergesagten Anomalien korrekt sind. Nehmen wir beispielsweise an, Ihr Modell sagt voraus, dass 100 Bilder anomal sind. Wenn 85 der Vorhersagen richtig sind (die wahren positiven Ergebnisse) und 15 falsch (die falsch positiven), wird die Genauigkeit wie folgt berechnet:

$85 \text{ wahre positive Ergebnisse} / (85 \text{ wahre positive} + 15 \text{ falsch positive Ergebnisse}) = 0,85$

Präzisionswert

Wenn das Modell jedoch nur 40 Bilder von 100 Anomalievorhersagen korrekt vorhersagt, ist der resultierende Präzisionswert mit 0,40 niedriger (d. h. $40 / (40 + 60) = 0,40$). In diesem Fall macht Ihr Modell mehr falsche Vorhersagen als richtige Vorhersagen. Um dieses Problem zu beheben, sollten Sie in Erwägung ziehen, Ihr Modell zu verbessern. Weitere Informationen finden Sie unter [Schritt 2: Verbessern Sie Ihr Modell](#).

Weitere Informationen finden Sie unter [Präzision und Erinnerung](#).

Wiedererkennung

Die Recall-Metrik beantwortet die Frage: Wie viele von der Gesamtzahl der anomalen Bilder im Testdatensatz wurden korrekt als anomal vorhergesagt?

Die Erinnerungsmetrik ist nützlich für Situationen, in denen die Kosten eines falsch negativen Ergebnisses hoch sind. Zum Beispiel, wenn die Kosten hoch sind, wenn ein defektes Teil nicht entfernt wird. Amazon Lookout for Vision bietet einen zusammenfassenden Wert der Erinnerungsmetrik für den gesamten Testdatensatz.

Bei der Rückrufaktion handelt es sich um den Bruchteil der fehlerhaften Testbilder, die korrekt erkannt wurden. Es ist ein Maß dafür, wie oft das Modell ein anomales Bild korrekt vorhersagen kann, obwohl es tatsächlich in den Bildern Ihres Testdatensatzes vorhanden ist. Die Formel für den Rückruf wird wie folgt berechnet:

$\text{Erinnerungswert} = \text{wahre positive Ergebnisse} / (\text{wahre positive Werte} + \text{falsch negative Werte})$

Der Bereich für das Erinnerungsvermögen liegt zwischen 0 und 1. In der Amazon Lookout for Vision Vision-Konsole wird der Rückruf als Prozentwert (0—100) angezeigt.

Ein höherer Erinnerungswert bedeutet, dass mehr der anomalen Bilder korrekt identifiziert wurden. Nehmen wir zum Beispiel an, der Testdatensatz enthält 100 anomale Bilder. Wenn das Modell 90 der 100 anomalen Bilder korrekt erkennt, erfolgt der Abruf wie folgt:

$90 \text{ echte positive Ergebnisse} / (90 \text{ wahre positive} + 10 \text{ falsch negative Ergebnisse}) = 0,90$

Erinnerungswert

Ein Erinnerungswert von 0,90 bedeutet, dass Ihr Modell die meisten anomalen Bilder im Testdatensatz korrekt vorhersagt. Wenn das Modell nur 20 der anomalen Bilder korrekt vorhersagt, ist der Erinnerungswert mit 0,20 niedriger (d. h. $20 / (20 + 80) = 0,20$).

In diesem Fall sollten Sie erwägen, Ihr Modell zu verbessern. Weitere Informationen finden Sie unter [Schritt 2: Verbessern Sie Ihr Modell](#).

Weitere Informationen finden Sie unter [Präzision und Erinnerung](#).

F1-Punktzahl

Amazon Lookout for Vision bietet einen durchschnittlichen Modellleistungswert für den Testdatensatz. Insbesondere wird die Modellleistung für die Klassifizierung von Anomalien anhand der F1-Score-Metrik gemessen, die das harmonische Mittel der Präzisions- und Erinnerungswerte darstellt.

Der F1-Score ist ein aggregiertes Maß, das sowohl Präzision als auch Erinnerungsvermögen berücksichtigt. Der Modellleistungswert ist ein Wert zwischen 0 und 1. Je höher der Wert, desto besser schneidet das Modell sowohl im Hinblick auf Erinnerung als auch Präzision ab. Für ein Modell mit einer Präzision von 0,9 und einem Erinnerungswert von 1,0 beträgt der F1-Wert beispielsweise 0,947.

Wenn das Modell nicht gut abschneidet, z. B. mit einer niedrigen Präzision von 0,30 und einem hohen Erinnerungsvermögen von 1,0, beträgt der F1-Wert 0,46. In ähnlicher Weise beträgt der F1-Wert 0,33, wenn die Genauigkeit hoch (0,95) und der Erinnerungswert niedrig (0,20) ist. In beiden Fällen ist der F1-Wert niedrig, was auf Probleme mit dem Modell hindeutet.

Weitere Informationen finden Sie unter [F1-Wert](#).

Durchschnittliche Schnittmenge über Union (IoU)

Die durchschnittliche prozentuale Überlappung zwischen den Anomaliemasken in den Testbildern und den Anomaliemasken, die das Modell für die Testbilder vorhersagt. Amazon Lookout for Vision gibt den durchschnittlichen IoU für jedes Anomalie-Label zurück und wird nur von [Bildsegmentierungsmodellen](#) zurückgegeben.

Ein niedriger Prozentwert weist darauf hin, dass das Modell die vorhergesagten Masken für ein Etikett nicht genau mit den Masken in den Testbildern abgleicht.

Das folgende Bild weist einen niedrigen IoU auf. Die orangefarbene Maske entspricht der Vorhersage des Modells und deckt die blaue Maske, die die Maske in einem Testbild darstellt, nicht fest ab.



Das folgende Bild hat eine höhere IoU. Die blaue Maske (Testbild) ist dicht von der orangefarbenen Maske (vorhergesagte Maske) bedeckt.



Ergebnisse der Tests

Während des Tests prognostiziert das Modell die Klassifizierung für jedes Testbild im Testdatensatz. Das Ergebnis jeder Vorhersage wird wie folgt mit der Kennzeichnung (normal oder anomal) des entsprechenden Testbilds verglichen:

- Die korrekte Vorhersage, dass ein Bild anomal ist, wird als wirklich positiv angesehen.
- Die falsche Vorhersage, dass ein Bild anomal ist, wird als falsch positiv angesehen.
- Die korrekte Vorhersage, dass ein Bild normal ist, wird als echtes Negativ angesehen.
- Die falsche Vorhersage, dass ein Bild normal ist, wird als falsch negativ angesehen.

Wenn es sich bei dem Modell um ein Segmentierungsmodell handelt, prognostiziert das Modell auch Masken und Anomaliekennzeichnungen für die Position von Anomalien auf dem Testbild.

Amazon Lookout for Vision verwendet die Ergebnisse der Vergleiche, um die Leistungskennzahlen zu generieren.

Schritt 2: Verbessern Sie Ihr Modell

Die Leistungskennzahlen könnten zeigen, dass Sie Ihr Modell verbessern können. Wenn das Modell beispielsweise nicht alle Anomalien im Testdatensatz erkennt, weist Ihr Modell einen niedrigen Erinnerungswert auf (d. h. die Erinnerungsmetrik hat einen niedrigen Wert). Wenn Sie Ihr Modell verbessern müssen, sollten Sie Folgendes berücksichtigen:

- Vergewissern Sie sich, dass die Bilder der Trainings- und Testdatensätze ordnungsgemäß beschriftet sind.
- Reduzieren Sie die Variabilität der Bildaufnahmebedingungen wie Beleuchtung und Objektposition und trainieren Sie Ihr Modell an Objekten desselben Typs.
- Stellen Sie sicher, dass Ihre Bilder nur den erforderlichen Inhalt zeigen. Wenn Ihr Projekt beispielsweise Anomalien an Maschinenteilen feststellt, stellen Sie sicher, dass sich keine anderen Objekte in Ihren Bildern befinden.
- Fügen Sie Ihren Zug- und Testdatensätzen weitere beschriftete Bilder hinzu. Wenn Ihr Testdatensatz eine hervorragende Erinnerungsfähigkeit und Präzision aufweist, das Modell jedoch bei der Implementierung schlecht abschneidet, ist Ihr Testdatensatz möglicherweise nicht repräsentativ genug und Sie müssen ihn erweitern.
- Wenn Ihr Testdatensatz zu schlechtem Erinnerungsvermögen und schlechter Präzision führt, sollten Sie sich überlegen, wie gut die Anomalien und die Bildaufnahmebedingungen in den Trainings- und Testdatensätzen übereinstimmen. Wenn Ihre Trainingsbilder nicht repräsentativ für die erwarteten Anomalien und Bedingungen sind, die Bilder in den Testbildern jedoch schon, fügen Sie dem Trainingsdatensatz Bilder mit den erwarteten Anomalien und Bedingungen hinzu. Wenn die Bilder des Testdatensatzes nicht den erwarteten Bedingungen entsprechen, die Trainingsbilder jedoch schon, aktualisieren Sie den Testdatensatz.

Weitere Informationen finden Sie unter [Weitere Bilder hinzufügen](#). Eine alternative Möglichkeit, Ihrem Trainingsdatensatz beschriftete Bilder hinzuzufügen, besteht darin, eine Testerkennungsaufgabe auszuführen und die Ergebnisse zu überprüfen. Anschließend können Sie die verifizierten Bilder zum Trainingsdatensatz hinzufügen. Weitere Informationen finden Sie unter [Verifizierung Ihres Modells mit einer Aufgabe zur Versuchserkennung](#).

- Stellen Sie sicher, dass Ihr Trainings- und Testdatensatz ausreichend unterschiedliche normale und anomale Bilder enthält. Die Bilder müssen die Art von normalen und anomalen Bildern repräsentieren, auf die Ihr Modell stoßen wird. Wenn Sie beispielsweise Leiterplatten analysieren, sollten Ihre normalen Bilder die Abweichungen in der Position und der Lötung von Komponenten wie Widerständen und Transistoren darstellen. Die anomalen Bilder sollten die verschiedenen

Arten von Anomalien darstellen, auf die das System stoßen kann, z. B. falsch platzierte oder fehlende Komponenten.

- Wenn Ihr Modell einen niedrigen durchschnittlichen IoU für erkannte Anomaliearten aufweist, überprüfen Sie die Maskenausgaben des Segmentierungsmodells. In einigen Anwendungsfällen, z. B. bei Kratzern, gibt das Modell möglicherweise Kratzer aus, die GroundTruth-Kratzen in den Testbildern sehr ähnlich sind, jedoch eine geringe Pixelüberlappung aufweisen. Zum Beispiel zwei parallel Linien, die 1 Pixel voneinander entfernt sind. In diesen Fällen ist der durchschnittliche IOU ein unzuverlässiger Indikator zur Messung des Erfolgs von Prognosen.
- Wenn die Bildgröße klein oder die Bildauflösung niedrig ist, sollten Sie erwägen, Bilder mit einer höheren Auflösung aufzunehmen. Die Bildabmessungen können zwischen 64 x 64 Pixeln und 4096 Pixeln X 4096 Pixeln liegen.
- Wenn die Größe der Anomalie gering ist, sollten Sie erwägen, die Bilder in separate Kacheln aufzuteilen und die gekachelten Bilder zum Trainieren und Testen zu verwenden. Auf diese Weise kann das Modell Fehler bei einer größeren Größe in einem Bild erkennen.

Nachdem Sie Ihren Trainings- und Testdatensatz verbessert haben, trainieren Sie Ihr Modell erneut und bewerten Sie es erneut. Weitere Informationen finden Sie unter [Trainieren Ihres Modells](#).

Wenn die Metriken zeigen, dass Ihr Modell eine akzeptable Leistung aufweist, können Sie seine Leistung überprüfen, indem Sie dem Testdatensatz die Ergebnisse einer Aufgabe zur Erkennung von Versuchen hinzufügen. Nach der Umschulung sollten die Leistungskennzahlen die Leistungskennzahlen der vorherigen Schulung bestätigen. Weitere Informationen finden Sie unter [Verifizierung Ihres Modells mit einer Aufgabe zur Versuchserkennung](#).

Anzeigen von Leistungsmetriken

Sie können Leistungskennzahlen über die Konsole und durch Aufrufen des `DescribeModel` Vorgangs abrufen.

Themen

- [Leistungsmetriken anzeigen \(Konsole\)](#)
- [Leistungskennzahlen \(SDK\) anzeigen](#)

Leistungsmetriken anzeigen (Konsole)

Nach Abschluss des Trainings werden auf der Konsole die Leistungskennzahlen angezeigt.

Die Amazon Lookout for Vision Vision-Konsole zeigt die folgenden Leistungskennzahlen für die während des Tests vorgenommenen Klassifizierungen an:

- [Genauigkeit](#)
- [Wiedererkennung](#)
- [F1-Punktzahl](#)

Wenn es sich bei dem Modell um ein Segmentierungsmodell handelt, zeigt die Konsole auch die folgenden Leistungskennzahlen für jedes Anomalie-Label an:

- Die Anzahl der Testbilder, auf denen das Anomalie-Label gefunden wurde.
- [F1-Punktzahl](#)
- [Durchschnittliche Schnittmenge über Union \(IoU\)](#)

Im Abschnitt mit der Übersicht der Testergebnisse finden Sie die Gesamtzahl der richtigen und falschen Vorhersagen für Bilder im Testdatensatz. Sie können auch die vorhergesagten und tatsächlichen Labelzuweisungen für einzelne Bilder im Testdatensatz sehen.

Das folgende Verfahren zeigt, wie Sie die Leistungskennzahlen aus der Modelllistenansicht eines Projekts abrufen.

So zeigen Sie Leistungsmetriken an (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie in der Projektansicht das Projekt aus, das die Version des Modells enthält, das Sie anzeigen möchten.
5. Wählen Sie im linken Navigationsbereich unter dem Projektnamen Modelle aus.
6. Wählen Sie in der Modelllistenansicht die Versionen des Modells aus, die Sie anzeigen möchten.
7. Sehen Sie sich auf der Seite mit den Modelldetails die Leistungskennzahlen auf der Registerkarte Leistungskennzahlen an.
8. Beachten Sie Folgendes:

- a. Der Abschnitt Modelleleistungskennzahlen enthält allgemeine Modellmetriken (Präzision, Erinnerung, F1-Score) für die Klassifizierungsvorhersagen, die das Modell für die Testbilder getroffen hat.
- b. Wenn es sich bei dem Modell um ein Bildsegmentierungsmodell handelt, enthält der Abschnitt Leistung pro Etikett die Anzahl der Testbilder, bei denen das Anomalie-Label gefunden wurde. Sie sehen auch Metriken (F1-Score, Average IoU) für jedes Anomalie-Label.
- c. Der Abschnitt Übersicht der Testergebnisse enthält die Ergebnisse für jedes Testbild, das Lookout for Vision zur Bewertung des Modells verwendet. Diese umfasst das Folgende:
 - Die Gesamtzahl der korrekten (richtig positiv) und falschen (falsch negativ) Klassifizierungsvorhersagen (normal oder anomal) für alle Testbilder.
 - Die Klassifizierungsvorhersage für jedes Testbild. Wenn unter einem Bild Korrekt angezeigt wird, entspricht die vorhergesagte Klassifizierung der tatsächlichen Klassifizierung für das Bild. Andernfalls hat das Modell das Bild nicht korrekt klassifiziert.
 - Bei einem Bildsegmentierungsmodell sehen Sie Anomalie-Labels, die das Modell dem Bild zugewiesen hat, und Masken auf dem Bild, die den Farben der Anomalie-Labels entsprechen.

Leistungskennzahlen (SDK) anzeigen

Sie können den [DescribeModel](#)Vorgang verwenden, um die zusammenfassenden Leistungskennzahlen (Klassifizierung) für das Modell, das Bewertungsmanifest und die Bewertungsergebnisse für ein Modell abzurufen.

Abrufen der zusammenfassenden Leistungskennzahlen

Die zusammenfassenden Leistungsmetriken für die Klassifizierungsvorhersagen, die das Modell beim Testen ([GenauigkeitWiedererkennung](#), und [F1-Punktzahl](#)) getroffen hat, werden in dem Performance Feld zurückgegeben, das durch einen Aufruf von `wirdDescribeModel` wird.

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

```
},
```

Das Performance Feld enthält nicht die Leistungsmetriken, die von einem Segmentierungsmodell zurückgegeben wurden, mit der Bezeichnung „Anomalie“. Sie können sie vor Ort abrufen. `EvaluationResult` Weitere Informationen finden Sie unter [Überprüfung des Bewertungsergebnisses](#).

Informationen zu zusammenfassenden Leistungskennzahlen finden Sie unter [Schritt 1: Bewerten Sie die Leistung Ihres Modells](#). Beispielcode finden Sie unter [Ihre Modelle anzeigen \(SDK\)](#).

Verwenden des Bewertungsmanifests

Das Bewertungsmanifest enthält Metriken zur Testvorhersage für die einzelnen Bilder, die zum Testen eines Modells verwendet wurden. Für jedes Bild im Testdatensatz enthält eine JSON-Zeile die ursprünglichen Testinformationen (Ground Truth) und die Vorhersage des Modells für das Bild. Amazon Lookout for Vision speichert das Bewertungsmanifest in einem Amazon S3 S3-Bucket. Sie können den Standort aus dem `EvaluationManifest` Feld in der Antwort vom `DescribeModel` Vorgang abrufen.

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

Das Dateinamenformat ist `EvaluationManifest-project name.json`. Beispielcode finden Sie unter [Deine Modelle ansehen](#).

In der folgenden JSON-Beispielzeile `class-name` ist das die Grundwahrheit für den Inhalt des Bildes. In diesem Beispiel enthält das Bild eine Anomalie. Das `confidence` Feld zeigt das Vertrauen, das Amazon Lookout for Vision in die Prognose hat.

```
{
  "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"
}
}
```

Überprüfung des Bewertungsergebnisses

Das Bewertungsergebnis umfasst die folgenden aggregierten Leistungskennzahlen (Klassifizierung) für den gesamten Satz von Testbildern:

- [Genauigkeit](#)
- [Wiedererkennung](#)
- ROC-Kurve (nicht in der Konsole angezeigt)
- Durchschnittliche Genauigkeit (wird in der Konsole nicht angezeigt)
- [F1-Punktzahl](#)

Das Bewertungsergebnis beinhaltet auch die Anzahl der Bilder, die für das Training und Testen des Modells verwendet wurden.

Wenn es sich bei dem Modell um ein Segmentierungsmodell handelt, umfasst das Bewertungsergebnis auch die folgenden Metriken für jede im Testdatensatz gefundene Anomaliebezeichnung:

- [Genauigkeit](#)
- [Wiedererkennung](#)
- [F1-Punktzahl](#)
- [Durchschnittliche Schnittmenge über Union \(IoU\)](#)

Amazon Lookout for Vision speichert das Bewertungsergebnis in einem Amazon S3 S3-Bucket. Sie können den Standort ermitteln, indem Sie den Wert von `EvaluationResult` in der Antwort vom `DescribeModel` Vorgang überprüfen.

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

Das Dateinamenformat ist `EvaluationResult-project name.json`. Ein Beispiel finden Sie unter [Deine Modelle ansehen](#).

Das folgende Schema zeigt das Bewertungsergebnis.

```
{
  "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 overall 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.
            }
        ]
    }
}
}

```

Verifizierung Ihres Modells mit einer Aufgabe zur Versuchserkennung

Wenn Sie die Qualität Ihres Modells überprüfen oder verbessern möchten, können Sie eine Testerkennungsaufgabe ausführen. Eine Aufgabe zur Erkennung von Versuchen erkennt Anomalien in neuen Bildern, die Sie bereitstellen.

Sie können die Erkennungsergebnisse überprüfen und die verifizierten Bilder zu Ihrem Datensatz hinzufügen. Wenn Sie separate Trainings- und Testdatensätze haben, werden die verifizierten Bilder dem Trainingsdatensatz hinzugefügt.

Sie können Bilder von Ihrem lokalen Computer oder Bilder, die sich in einem Amazon S3 S3-Bucket befinden, überprüfen. Wenn Sie verifizierte Bilder zum Datensatz hinzufügen möchten, müssen sich Bilder, die sich in einem S3-Bucket befinden, im selben S3-Bucket wie die Bilder in Ihrem Datensatz befinden.

Note

Um eine Testerkennungsaufgabe auszuführen, stellen Sie sicher, dass für Ihren S3-Bucket die Versionierung aktiviert ist. Weitere Informationen finden Sie unter [Versionierung verwenden](#). Der Konsolen-Bucket wird mit aktivierter Versionierung erstellt.

Standardmäßig werden Ihre Bilder mit einem Schlüssel verschlüsselt, den AWS besitzt und verwaltet. Sie können sich auch dafür entscheiden, Ihren eigenen AWS Key Management Service (KMS) - Schlüssel zu verwenden. Weitere Informationen finden Sie unter [AWS Key Management Service-Konzepte](#).

Themen

- [Ausführen einer Aufgabe zur Erkennung von Testversionen](#)
- [Überprüfung der Ergebnisse der Studienerkennung](#)
- [Korrigieren von Segmentierungsbeschriftungen mit dem Annotationstool](#)

Ausführen einer Aufgabe zur Erkennung von Testversionen

Führen Sie die folgenden Schritte aus, um eine Aufgabe zur Testerkennung auszuführen.

So führen Sie eine Testerkennung aus (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie in der Projektansicht das Projekt aus, das die Version des Modells enthält, das Sie anzeigen möchten.
5. Wählen Sie im linken Navigationsbereich unter dem Projektnamen die Option Testerkennungen aus.
6. Wählen Sie in der Ansicht Testerkennungen die Option Testerkennung ausführen aus.
7. Geben Sie auf der Seite Testerkennung ausführen im Feld Aufgabenname einen Namen für Ihre Aufgabe zur Testerkennung ein.
8. Wählen Sie unter Modell auswählen die Version des Modells aus, die Sie verwenden möchten.
9. Importieren Sie die Bilder entsprechend der Bildquelle wie folgt:
 - Wenn Sie Ihre Quellbilder aus einem Amazon S3 S3-Bucket importieren, geben Sie den S3-URI ein.

 Tip

Wenn Sie die Getting Started-Beispielbilder verwenden, verwenden Sie den Ordner `extra_images`. Die Amazon S3 S3-URI lautet `s3://your_bucket/circuitboard/extra_images`.

- Wenn Sie Bilder von Ihrem Computer hochladen, fügen Sie die Bilder hinzu, nachdem Sie „Anomalien erkennen“ ausgewählt haben.
10. (Optional) Wenn Sie Ihren eigenen AWS KMS-Verschlüsselungsschlüssel verwenden möchten, gehen Sie wie folgt vor:
 - a. Wählen Sie für die Verschlüsselung von Bilddaten die Option Verschlüsselungseinstellungen anpassen (erweitert).
 - b. Geben Sie in `encryption.aws_kms_key` den Amazon-Ressourcennamen (ARN) Ihres Schlüssels ein, oder wählen Sie einen vorhandenen AWS-KMS-Schlüssel aus. Um einen neuen Schlüssel zu erstellen, wählen Sie AWS KMS-Schlüssel erstellen.
 11. Wählen Sie „Anomalien erkennen“ und anschließend „Testerkennung ausführen“, um die Aufgabe zur Testerkennung zu starten.
 12. Überprüfen Sie den aktuellen Status in der Ansicht „Erkennungen von Testversionen“. Es kann eine Weile dauern, bis die Erkennung der Studie abgeschlossen ist.

Überprüfung der Ergebnisse der Studienerkennung

Die Überprüfung der Ergebnisse einer Studienerkennung kann Ihnen helfen, Ihr Modell zu verbessern.

Wenn die Leistungskennzahlen schlecht sind, verbessern Sie Ihr Modell, indem Sie eine Versuchserkennung durchführen und dann verifizierte Bilder zum Datensatz hinzufügen (Trainingsdatensatz, falls Sie über separate Datensätze verfügen).

Wenn die Leistungskennzahlen des Modells gut sind, die Ergebnisse einer Versuchserkennung jedoch schlecht sind, können Sie Ihr Modell verbessern, indem Sie dem Datensatz (Trainingsdatensatz) verifizierte Bilder hinzufügen. Wenn Sie über einen separaten Testdatensatz verfügen, sollten Sie erwägen, dem Testdatensatz weitere Bilder hinzuzufügen.

Nachdem Sie Ihrem Datensatz verifizierte Bilder hinzugefügt haben, trainieren Sie Ihr Modell erneut und bewerten Sie es erneut. Weitere Informationen finden Sie unter [Trainieren Ihres Modells](#).

Um die Ergebnisse einer Versuchsdetektion zu verifizieren

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
3. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten. Das Dashboard für Ihr Projekt wird angezeigt.
4. Wählen Sie im linken Navigationsbereich die Option Testerkennungen aus.
5. Wählen Sie die Testerkennung aus, die Sie überprüfen möchten.
6. Wählen Sie auf der Seite zur Erkennung von Testversuchen die Option Computervorhersagen überprüfen aus.
7. Wählen Sie Alle Bilder auf dieser Seite auswählen.
8. Wenn die Vorhersagen korrekt sind, wählen Sie Als korrekt verifizieren aus. Wählen Sie andernfalls Als falsch verifizieren aus. Die Prognose und der Zuverlässigkeitswert für Vorhersagen werden unter jedem Bild angezeigt.
9. Wenn Sie die Bezeichnung für ein Bild ändern müssen, gehen Sie wie folgt vor:
 - a. Wählen Sie unter dem Bild „Korrekt“ oder „Falsch“.
 - b. Wenn Sie die richtige Bezeichnung für ein Bild nicht ermitteln können, vergrößern Sie das Bild, indem Sie das Bild in der Galerie auswählen.

 Note

Sie können Bildbeschriftungen filtern, indem Sie im Abschnitt Filter die gewünschte Bezeichnung oder den gewünschten Labelstatus auswählen. Im Bereich Sortieroptionen können Sie nach dem Konfidenzwert sortieren.

10. Wenn es sich bei Ihrem Modell um ein Segmentierungsmodell handelt und die Maske oder die Anomaliebezeichnung für ein Bild falsch ist, wählen Sie „Anomaler Bereich unter dem Bild“ und öffnen Sie das Annotationstool. Aktualisieren Sie die Segmentierungsinformationen, indem Sie [Korrigieren von Segmentierungsbeschriftungen mit dem Annotationstool](#)
11. Wiederholen Sie bei Bedarf die Schritte 7 bis 10 auf jeder Seite, bis alle Bilder verifiziert wurden.
12. Wählen Sie Verifizierte Bilder zum Datensatz hinzufügen aus. Wenn Sie separate Datensätze haben, werden die Bilder dem Trainingsdatensatz hinzugefügt.

13. Trainieren Sie Ihr Modell erneut. Weitere Informationen finden Sie unter [the section called “Trainieren Ihres Modells”](#).

Korrigieren von Segmentierungsbeschriftungen mit dem Annotationstool

Sie verwenden das Annotationstool, um ein Bild zu segmentieren, indem Sie anomale Bereiche mit einer Maske markieren.

Um die Segmentierungsbeschriftungen für ein Bild mit dem Annotationstool zu korrigieren

1. Öffnen Sie das Annotationstool, indem Sie in der Datensatz-Galerie einen anomalen Bereich unter einem Bild auswählen.
2. Wenn das Anomalie-Label für eine Maske nicht korrekt ist, wählen Sie die Maske aus und wählen Sie dann unter Anomalie-Labels das richtige Anomalie-Label aus. Wählen Sie bei Bedarf die Option Anomalie-Label hinzufügen aus, um ein neues Anomalie-Label hinzuzufügen.
3. Wenn die Maske nicht korrekt ist, wählen Sie unten auf der Seite ein Zeichenwerkzeug aus und zeichnen Sie Masken, die anomale Bereiche für die Anomaliebeschriftung dicht abdecken. Die folgende Abbildung zeigt ein Beispiel für eine Maske, die eine Anomalie dicht verdeckt.



Das Folgende ist ein Beispiel für eine schlechte Maske, die eine Anomalie nicht dicht verdeckt.



4. Wenn Sie weitere Bilder korrigieren müssen, wählen Sie Weiter und wiederholen Sie die Schritte 2 und 3.
5. Wählen Sie Senden und schließen, um die Aktualisierung der Bilder abzuschließen.

Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells

Um Anomalien in Bildern mit Ihrem Modell zu erkennen, müssen Sie Ihr Modell zunächst mit der [StartModel](#) Operation starten. Die Amazon Lookout for Vision Vision-Konsole bietet AWS CLI Befehle, mit denen Sie Ihr Modell starten und stoppen können. Dieser Abschnitt enthält Beispielcode, den Sie verwenden können.

Nach dem Start Ihres Modells können Sie den DetectAnomalies Vorgang verwenden, um Anomalien in einem Bild zu erkennen. Weitere Informationen finden Sie unter [Erkennung von Anomalien in einem Bild](#).

Themen

- [Inferenzeinheiten](#)
- [Availability Zones](#)
- [Starten Sie Ihr Amazon Lookout for Vision Vision-Modell](#)
- [Ihr Amazon Lookout for Vision Vision-Modell beenden](#)

Inferenzeinheiten

Wenn Sie Ihr Modell starten, stellt Amazon Lookout for Vision mindestens eine Rechenressource bereit, die als Inferenzeinheit bezeichnet wird. Sie geben die Anzahl der zu verwendenden Inferenzeinheiten im MinInferenceUnits Eingabeparameter für die API an. StartModel Die Standardzuweisung für ein Modell ist 1 Inferenzeinheit.

Important

Ihnen werden die Anzahl der Stunden, in denen Ihr Modell läuft, und die Anzahl der Inferenzeinheiten, die Ihr Modell während der Ausführung verwendet, in Rechnung gestellt, je nachdem, wie Sie den Betrieb Ihres Modells konfigurieren. Wenn Sie das Modell beispielsweise mit zwei Inferenzeinheiten starten und das Modell 8 Stunden lang verwenden, werden Ihnen 16 Inferenzstunden in Rechnung gestellt (8 Stunden Laufzeit x zwei Inferenzeinheiten). Weitere Informationen finden Sie unter [Amazon Lookout for Vision Pricing](#). Wenn Sie Ihr Modell nicht ausdrücklich durch einen Aufruf beenden [StopModel](#), wird Ihnen eine Gebühr berechnet, auch wenn Sie nicht aktiv Bilder mit Ihrem Modell analysieren.

Die Transaktionen pro Sekunde (TPS), die eine einzelne Inferenzeinheit unterstützt, werden von folgenden Faktoren beeinflusst:

- Der Algorithmus, den Lookout for Vision verwendet, um das Modell zu trainieren. Wenn Sie ein Modell trainieren, werden mehrere Modelle trainiert. Lookout for Vision wählt anhand der Größe des Datensatzes und seiner Zusammensetzung aus normalen und anomalen Bildern das Modell mit der besten Leistung aus.
- Bilder mit höherer Auflösung benötigen mehr Zeit für die Analyse.
- Kleinere Bilder (gemessen in MBs) werden schneller analysiert als größere Bilder.

Verwaltung des Durchsatzes mit Inferenzeinheiten

Sie können den Durchsatz Ihres Modells je nach den Anforderungen an Ihre Anwendung erhöhen oder verringern. Verwenden Sie zusätzliche Inferenzeinheiten, um den Durchsatz zu erhöhen. Jede zusätzliche Inferenzeinheit erhöht Ihre Verarbeitungsgeschwindigkeit um eine Inferenzeinheit. Informationen zur Berechnung der Anzahl der benötigten Inferenzeinheiten finden Sie unter [Berechnung von Inferenzeinheiten für Amazon Rekognition Custom Labels und Amazon Lookout für Vision-Modelle](#). Wenn Sie den unterstützten Durchsatz Ihres Modells ändern möchten, haben Sie zwei Möglichkeiten:

Manuelles Hinzufügen oder Entfernen von Inferenzeinheiten

[Stoppen](#) Sie das Modell und [starten](#) Sie es dann mit der erforderlichen Anzahl von Inferenzeinheiten neu. Der Nachteil dieses Ansatzes besteht darin, dass das Modell während des Neustarts keine Anfragen empfangen kann und nicht zur Bewältigung von Nachfragespitzen verwendet werden kann. Verwenden Sie diesen Ansatz, wenn Ihr Modell einen konstanten Durchsatz aufweist und Ihr Anwendungsfall Ausfallzeiten von 10-20 Minuten tolerieren kann. Ein Beispiel wäre, wenn Sie Ihr Modell mithilfe eines wöchentlichen Zeitplans stapelweise aufrufen möchten.

Automatische Skalierung von Inferenzeinheiten

Wenn Ihr Modell Nachfragespitzen bewältigen muss, kann Amazon Lookout for Vision die Anzahl der Inferenzeinheiten, die Ihr Modell verwendet, automatisch skalieren. Wenn die Nachfrage steigt, fügt Amazon Lookout for Vision dem Modell zusätzliche Inferenzeinheiten hinzu und entfernt sie, wenn die Nachfrage sinkt.

Damit Lookout for Vision automatisch die Inferenzeinheiten für ein Modell skaliert, [starten Sie](#) das Modell und legen Sie mithilfe des Parameters die maximale Anzahl von Inferenzeinheiten fest, die

es verwenden kann. `MaxInferenceUnits` Durch die Festlegung einer maximalen Anzahl von Inferenzeinheiten können Sie die Kosten für den Betrieb des Modells verwalten, indem Sie die Anzahl der verfügbaren Inferenzeinheiten einschränken. Wenn Sie keine maximale Anzahl von Einheiten angeben, skaliert Lookout for Vision Ihr Modell nicht automatisch, sondern verwendet nur die Anzahl der Inferenzeinheiten, mit der Sie begonnen haben. Informationen zur maximalen Anzahl von Inferenzeinheiten finden Sie unter [Service Quotas](#).

Mithilfe des `MinInferenceUnits`-Parameters können Sie auch eine Mindestanzahl von Inferenzeinheiten angeben. Auf diese Weise können Sie den Minstdurchsatz für Ihr Modell angeben, wobei eine einzelne Inferenzeinheit einer Stunde Verarbeitungszeit entspricht.

 Note

Sie können die maximale Anzahl von Inferenzeinheiten mit der Lookout for Vision Vision-Konsole nicht festlegen. Geben Sie stattdessen den `MaxInferenceUnits` Eingabeparameter für den `StartModel`-Vorgang an.

Lookout for Vision bietet die folgenden Amazon CloudWatch Logs-Metriken, anhand derer Sie den aktuellen Status der automatischen Skalierung für ein Modell ermitteln können.

Metrik	Beschreibung
<code>DesiredInferenceUnits</code>	Die Anzahl der Inferenzeinheiten, auf die Lookout for Vision nach oben oder unten skaliert.
<code>InServiceInferenceUnits</code>	Die Anzahl der Inferenzeinheiten, die das Modell verwendet.

Falls `DesiredInferenceUnits` = `InServiceInferenceUnits`, skaliert Lookout for Vision derzeit nicht die Anzahl der Inferenzeinheiten.

Wenn `DesiredInferenceUnits` > `InServiceInferenceUnits`, wird Lookout for Vision auf den Wert von `DesiredInferenceUnits` hochskaliert.

Wenn `DesiredInferenceUnits` < `InServiceInferenceUnits`, wird Lookout for Vision auf den Wert von `DesiredInferenceUnits` herunterskaliert.

Weitere Informationen zu den von Lookout for Vision zurückgegebenen Metriken und zum Filtern von Dimensionen finden Sie unter [Überwachen von Lookout for Vision mit Amazon CloudWatch](#).

Rufen Sie [DescribeModel](#) auf und überprüfen Sie das `MaxInferenceUnits`-Feld in der Antwort, um die maximale Anzahl von Inferenzeinheiten zu ermitteln, die Sie für ein Modell angefordert haben.

Availability Zones

Amazon Lookout for Vision; verteilt Inferenzeinheiten über mehrere Availability Zones innerhalb einer AWS Region, um die Verfügbarkeit zu erhöhen. Weitere Informationen finden Sie unter [Availability Zones \(AZ\)](#). Um Ihre Produktionsmodelle vor Ausfällen in der Availability Zone (AZ) und vor Ausfällen von Inferenzeinheiten zu schützen, starten Sie Ihre Produktionsmodelle mit mindestens zwei Inferenzeinheiten.

Bei einem Ausfall der Availability Zone (AZ) sind alle Inferenzeinheiten in der Availability Zone (AZ) nicht verfügbar und die Modellkapazität wird reduziert. Aufrufe an [DetectAnomalies](#) werden auf die verbleibenden Inferenzeinheiten umverteilt. Solche Aufrufe sind erfolgreich, wenn sie die unterstützten Transaktionen pro Sekunde (TPS) der verbleibenden Inferenzeinheiten nicht überschreiten. Nach der AWS Reparatur der Availability Zone werden die Inferenzeinheiten neu gestartet und die volle Kapazität wird wiederhergestellt.

Wenn eine einzelne Inferenzeinheit ausfällt, startet Amazon Lookout for Vision automatisch eine neue Inferenzeinheit in derselben Availability Zone. Die Modellkapazität wird reduziert, bis die neue Inferenzeinheit gestartet wird.

Starten Sie Ihr Amazon Lookout for Vision Vision-Modell

Bevor Sie ein Amazon Lookout for Vision Vision-Modell zur Erkennung von Anomalien verwenden können, müssen Sie das Modell zunächst starten. Sie starten ein Modell, indem Sie die [StartModel](#)API aufrufen und Folgendes übergeben:

- `ProjectName`— Der Name des Projekts, das das Modell enthält, das Sie starten möchten.
- `ModelVersion`— Die Version des Modells, das Sie starten möchten.
- `MinInferenceUnits`— Die Mindestanzahl von Inferenzeinheiten. Weitere Informationen finden Sie unter [Inferenzeinheiten](#).

- (Optional) `MaxInferenceUnits`— Die maximale Anzahl von Inferenzeinheiten, die Amazon Lookout for Vision verwenden kann, um das Modell automatisch zu skalieren. Weitere Informationen finden Sie unter [Automatische Skalierung von Inferenzeinheiten](#).

Die Amazon Lookout for Vision Vision-Konsole bietet Beispielcode, mit dem Sie ein Modell starten und stoppen können.

Note

Ihnen wird die Zeit in Rechnung gestellt, in der Ihr Modell läuft. Informationen zum Stoppen eines laufenden Modells finden Sie unter [Ihr Amazon Lookout for Vision Vision-Modell beenden](#).

Sie können das AWS SDK verwenden, um laufende Modelle in allen AWS Regionen anzuzeigen, in denen Lookout for Vision verfügbar ist. Beispielcode finden Sie unter [find_running_models.py](#).

Themen

- [Starten Sie Ihr Modell \(Konsole\)](#)
- [Starten Sie Ihr Amazon Lookout for Vision Vision-Modell \(SDK\)](#)

Starten Sie Ihr Modell (Konsole)

Die Amazon Lookout for Vision Vision-Konsole bietet einen AWS CLI Befehl, mit dem Sie ein Modell starten können. Nach dem Start des Modells können Sie beginnen, Anomalien in Bildern zu erkennen. Weitere Informationen finden Sie unter [Erkennung von Anomalien in einem Bild](#).

So starten Sie ein Modell (Konsole)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.

5. Wählen Sie auf der Ressourcenseite Projekte das Projekt aus, das das trainierte Modell enthält, das Sie starten möchten.
6. Wählen Sie im Abschnitt Modelle das Modell aus, das Sie starten möchten.
7. Wählen Sie auf der Detailseite des Modells die Option Modell verwenden und dann API in die Cloud integrieren aus.

 Tip

Wenn Sie Ihr Modell auf einem Edge-Gerät bereitstellen möchten, wählen Sie Create model packaging job aus. Weitere Informationen finden Sie unter [Verpacken Ihres Amazon Lookout for Vision Vision-Modells](#).

8. Kopieren Sie unter AWS-CLI-Befehle den AWS CLI-Befehl, der aufgerufen wird `start-model`.
9. Geben Sie an der Eingabeaufforderung den `start-model`-Befehl ein, den Sie im vorherigen Schritt kopiert haben. Wenn Sie das `lookoutvision` Profil zum Abrufen von Anmeldeinformationen verwenden, fügen Sie den `--profile lookoutvision-access` Parameter hinzu.
10. Wählen Sie in der Konsole auf der linken Navigationsseite Modelle aus.
11. In der Spalte Status finden Sie den aktuellen Status des Modells. Wenn der Status Gehostet lautet, können Sie das Modell verwenden, um Anomalien in Bildern zu erkennen. Weitere Informationen finden Sie unter [Erkennung von Anomalien in einem Bild](#).

Starten Sie Ihr Amazon Lookout for Vision Vision-Modell (SDK)

Sie starten ein Modell, indem Sie die [StartModel](#)-Operation aufrufen.

Es kann eine Weile dauern, bis ein Modell gestartet wird. Sie können den aktuellen Status überprüfen, indem Sie anrufen [DescribeModel](#). Weitere Informationen finden Sie unter [Deine Modelle ansehen](#).

Um Ihr Modell (SDK) zu starten

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um ein Modell zu starten.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das das Modell enthält, das Sie starten möchten.
- `model-version` zu der Version des Modells, das Sie starten möchten.
- `--min-inference-units` auf die Anzahl der Inferenzeinheiten, die Sie verwenden möchten.
- (Optional) `--max-inference-units` auf die maximale Anzahl von Inferenzeinheiten, die Amazon Lookout for Vision verwenden kann, um das Modell automatisch zu skalieren.

```
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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem GitHub Repository mit den Beispielen für das AWS Documentation SDK. Das vollständige Beispiel finden Sie [hier](#).

```

/**
 * 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 lfvcClient An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to host.
 * @param modelVersion The version of the model that you want to host.
 * @param minInferenceUnits The number of inference units to use for hosting.
 * @param 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 lfvcClient, 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. Wenn die Ausgabe des Codes lautet `Model is hosted and ready for use`, können Sie das Modell verwenden, um Anomalien in Bildern zu erkennen. Weitere Informationen finden Sie unter [Erkennung von Anomalien in einem Bild](#).

Ihr Amazon Lookout for Vision Vision-Modell beenden

Um ein laufendes Modell zu stoppen, rufen Sie die `StopModel` Operation auf und übergeben Folgendes:

- **Projekt** — Der Name des Projekts, das das Modell enthält, das Sie beenden möchten.
- **ModelVersion**— Die Version des Modells, die Sie beenden möchten.

Die Amazon Lookout for Vision Vision-Konsole bietet Beispielcode, mit dem Sie ein Modell stoppen können.

 Note

Ihnen wird die Zeit in Rechnung gestellt, in der Ihr Modell läuft.

Themen

- [Stoppen Sie Ihr Modell \(Konsole\)](#)
- [Ihr Amazon Lookout for Vision Vision-Modell \(SDK\) beenden](#)

Stoppen Sie Ihr Modell (Konsole)

Gehen Sie wie im Folgenden beschrieben vor, um die Verwendung der Konsole durch Ihr Modell zu beenden.

So stoppen Sie Ihr Modell (Konsole)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projektressourcen das Projekt aus, das das laufende Modell enthält, das Sie beenden möchten.
6. Wählen Sie im Abschnitt Modelle das Modell aus, das Sie stoppen möchten.
7. Wählen Sie auf der Detailseite des Modells die Option Modell verwenden und dann API in die Cloud integrieren aus.
8. Kopieren Sie unter AWS-CLI-Befehle den AWS CLI-Befehl, der aufgerufen wird `stop-model`.
9. Geben Sie an der Eingabeaufforderung den `stop-model`-Befehl ein, den Sie im vorherigen Schritt kopiert haben. Wenn Sie das `lookoutvision` Profil zum Abrufen von Anmeldeinformationen verwenden, fügen Sie den `--profile lookoutvision-access` Parameter hinzu.
10. Wählen Sie in der Konsole auf der linken Navigationsseite Modelle aus.

11. In der Spalte Status finden Sie den aktuellen Status des Modells. Das Modell wurde gestoppt, wenn in der Spalte Status der Wert Training abgeschlossen angezeigt wurde.

Ihr Amazon Lookout for Vision Vision-Modell (SDK) beenden

Sie beenden ein Modell, indem Sie die [StopModel](#) Operation aufrufen.

Es kann eine Weile dauern, bis ein Modell gestoppt wird. Um den aktuellen Status zu überprüfen, rufen Sie `DescribeModel` auf.

Um Ihr Modell (SDK) zu stoppen

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um ein laufendes Modell zu stoppen.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das das Modell enthält, das Sie beenden möchten.
- `model-version` zu der Version des Modells, die Sie beenden möchten.

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

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem GitHub Repository mit den Beispielen für das AWS Documentation SDK. Das vollständige Beispiel finden Sie [hier](#).

```
/**
 * 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.
 * @param 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;  
  
    }
```

Erkennung von Anomalien in einem Bild

Um Anomalien in einem Bild mit einem trainierten Amazon Lookout for Vision Vision-Modell zu erkennen, rufen Sie den Vorgang auf. [DetectAnomalies](#) Das Ergebnis von `DetectAnomalies` beinhaltet eine boolesche Vorhersage, die das Bild so klassifiziert, dass es eine oder mehrere Anomalien enthält, und einen Konfidenzwert für die Vorhersage. Handelt es sich bei dem Modell um ein Bildsegmentierungsmodell, umfasst das Ergebnis auch eine farbige Maske, die die Positionen verschiedener Arten von Anomalien zeigt.

Die Bilder, die Sie bereitstellen, `DetectAnomalies` müssen dieselben Breiten- und Höhenmaße haben wie die Bilder, mit denen Sie das Modell trainiert haben.

`DetectAnomalies` akzeptiert Bilder im PNG- oder JPG-Format. Wir empfehlen, dass die Bilder dasselbe Kodierungs- und Komprimierungsformat haben wie die Bilder, die zum Trainieren des Modells verwendet wurden. Wenn Sie das Modell beispielsweise mit Bildern im PNG-Format trainieren, rufen Sie es `DetectAnomalies` mit Bildern im PNG-Format auf.

Bevor Sie anrufen `DetectAnomalies`, müssen Sie Ihr Modell zunächst mit der `StartModel` Operation starten. Weitere Informationen finden Sie unter [Starten Sie Ihr Amazon Lookout for Vision Vision-Modell](#). Ihnen werden die Zeit in Minuten, die ein Modell ausgeführt wird, und die Anzahl der Einheiten zur Erkennung von Anomalien, die Ihr Modell verwendet, in Rechnung gestellt. Wenn Sie kein Modell verwenden, verwenden Sie den `StopModel` Vorgang, um Ihr Modell zu beenden. Weitere Informationen finden Sie unter [Ihr Amazon Lookout for Vision Vision-Modell beenden](#).

Themen

- [Anrufen DetectAnomalies](#)
- [Die Antwort von verstehen DetectAnomalies](#)
- [Feststellen, ob ein Bild anomal ist](#)
- [Informationen zur Klassifizierung und Segmentierung werden angezeigt](#)
- [Auffinden von Anomalien mit einer Funktion AWS Lambda](#)

Anrufen DetectAnomalies

Um anzurufen `DetectAnomalies`, geben Sie Folgendes an:

- **Projekt** — Der Name des Projekts, das das Modell enthält, das Sie verwenden möchten.

- **ModelVersion**— Die Version des Modells, das Sie verwenden möchten.
- **ContentType**— Der Bildtyp, den Sie analysieren möchten. Gültige Werte sind `image/png` (Bilder im PNG-Format) und `image/jpeg` (Bilder im JPG-Format).
- **Body** — Die unverschlüsselten Binärbytes, die das Bild darstellen.

Das Bild muss dieselben Abmessungen haben wie die Bilder, die zum Trainieren des Modells verwendet wurden.

Das folgende Beispiel zeigt, wie man `DetectAnomalies` anruft. Sie können die Funktionsantwort aus den Python- und Java-Beispielen verwenden, um Funktionen aufzurufen [Feststellen, ob ein Bild anomal ist](#).

AWS CLI

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `DetectAnomalies` CLI-Vorgang an. Ändern Sie die Werte der folgenden Eingabeparameter:

- `project name` mit dem Namen des Projekts, das Sie verwenden möchten.
- `model version` mit der Version des Modells, das Sie verwenden möchten.
- `content type` mit dem Typ des Bildes, das Sie verwenden möchten. Gültige Werte sind `image/png` (Bilder im PNG-Format) und `image/jpeg` (Bilder im JPG-Format).
- `file name` mit dem Pfad und dem Dateinamen des Bilds, das Sie verwenden möchten. Stellen Sie sicher, dass der Dateityp dem Wert von `content-type` entspricht.

```
aws lookoutvision detect-anomalies --project-name project name\
  --model-version model version\
  --content-type content type\
  --body file name \
  --profile lookoutvision-access
```

Python

Das vollständige Codebeispiel finden Sie unter [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));
    }

```

Die Antwort von verstehen DetectAnomalies

Die Antwort von DetectAnomalies hängt vom Typ des Modells ab, das Sie trainieren (Klassifikationsmodell oder Segmentierungsmodell). In beiden Fällen handelt es sich bei der Antwort um ein [DetectAnomalyResult](#)-Objekt.

Klassifikationsmodell

Wenn es sich bei Ihrem Modell um ein [Modell zur Bildklassifizierung](#) handelt, enthält DetectAnomalies die Antwort von Folgendes:

- **IsAnomalous**— Ein boolescher Indikator dafür, dass das Bild eine oder mehrere Anomalien enthält.
- **Vertrauen** — Das Vertrauen, das Amazon Lookout for Vision in die Genauigkeit der Anomalievorhersage hat (`IsAnomalous`). `Confidence` ist ein Fließkommawert zwischen 0 und 1. Ein höherer Wert steht für ein höheres Konfidenzniveau.

- Quelle — Informationen über das Bild, an das übergeben wurde `DetectAnomalies`.

```
{
  "DetectAnomalyResult": {
    "Source": {
      "Type": "direct"
    },
    "IsAnomalous": true,
    "Confidence": 0.9996867775917053
  }
}
```

Sie stellen fest, ob ein Bild anomal ist, indem Sie das `IsAnomalous` Feld markieren und sicherstellen, dass der `Confidence` Wert hoch genug für Ihre Bedürfnisse ist.

Wenn Sie feststellen, dass die von `DetectAnomalies` zurückgegebenen Konfidenzwerte zu niedrig sind, sollten Sie das Modell erneut trainieren. Beispielcode finden Sie unter [Klassifizierung](#).

Segmentierungsmodell

Wenn es sich bei Ihrem Modell um ein handelt [Modell der Bildsegmentierung](#), umfasst die Antwort Klassifizierungs- und Segmentierungsinformationen, z. B. eine Bildmaske und Anomalietypen. Klassifizierungsinformationen werden getrennt von Segmentierungsinformationen berechnet, und Sie sollten nicht von einer Beziehung zwischen ihnen ausgehen. Wenn Sie in der Antwort keine Segmentierungsinformationen erhalten, überprüfen Sie, ob Sie die neueste Version des AWS SDK installiert haben (AWS Command Line Interface falls Sie das AWS CLI verwenden). Beispielcode finden Sie unter [Segmentierung](#) und [Informationen zur Klassifizierung und Segmentierung werden angezeigt](#).

- `IsAnomalous`(Klassifizierung) — Ein boolescher Indikator, der das Bild entweder als normal oder als anomal klassifiziert.
- Vertrauen (Klassifizierung) — Das Vertrauen, das Amazon Lookout for Vision in die Genauigkeit der Klassifizierung des Bildes hat (`IsAnomalous`). `Confidence` ist ein Fließkommawert zwischen 0 und 1. Ein höherer Wert steht für ein höheres Konfidenzniveau.
- Quelle — Informationen über das Bild, an das übergeben wurde `DetectAnomalies`.
- `AnomalyMask`(Segmentierung) — Eine Pixelmaske, die die im analysierten Bild gefundenen Anomalien abdeckt. Das Bild kann mehrere Anomalien aufweisen. Die Farbe einer Maskenkarte

gibt den Typ einer Anomalie an. Die Maskenfarben sind den Farben zugeordnet, die den Anomalietypen im Trainingsdatensatz zugewiesen sind. Um den Anomalie-Typ anhand einer Maskenfarbe zu ermitteln, überprüfen Sie das `PixelAnomaly` Feld für jede `Color` in der Liste angegebene Anomalie. Anomalies Beispielcode finden Sie unter [Informationen zur Klassifizierung und Segmentierung werden angezeigt](#).

- **Anomalien (Segmentierung)** — Eine Liste der im Bild gefundenen Anomalien. Jede Anomalie enthält den Anomalie-Typ (`Name`) und Pixelinformationen (`PixelAnomaly`). `TotalPercentageArea` ist der prozentuale Bereich des Bildes, den die Anomalie bedeckt. `Color` ist die Maskenfarbe für die Anomalie.

Das erste Element in der Liste ist immer ein Anomalie-Typ, der den Bildhintergrund (`BACKGROUND`) darstellt, und sollte nicht als Anomalie betrachtet werden. Amazon Lookout for Vision fügt der Antwort automatisch den Typ der Hintergrundanomalie hinzu. Sie müssen in Ihrem Datensatz keinen Hintergrundanomalie-Typ deklarieren.

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

Feststellen, ob ein Bild anomal ist

Sie können auf verschiedene Arten feststellen, ob ein Bild anomal ist. Welche Methode Sie wählen, hängt von Ihrem Anwendungsfall und dem Typ Ihres Modells ab. Im Folgenden sind mögliche Lösungen aufgeführt.

Themen

- [Klassifizierung](#)
- [Segmentierung](#)

Klassifizierung

`IsAnomalous` klassifiziert ein Bild als ungewöhnlich. Verwenden Sie das `Confidence` Feld, um zu entscheiden, ob das Bild tatsächlich anomal ist. Ein höherer Wert bedeutet eine größere Zuverlässigkeit. Sie könnten beispielsweise entscheiden, dass ein Produkt nur dann defekt ist, wenn die Zuverlässigkeit über 80% liegt. Sie können Bilder klassifizieren, die nach Klassifikationsmodellen oder nach Bildsegmentierungsmodellen analysiert wurden.

Python

Das vollständige Codebeispiel finden Sie unter. [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;
    }
```

Segmentierung

Wenn es sich bei Ihrem Modell um ein Bildsegmentierungsmodell handelt, können Sie anhand der Segmentierungsinformationen feststellen, ob ein Bild Anomalien enthält. Sie können auch ein Bildsegmentierungsmodell verwenden, um Bilder zu klassifizieren. Ein Beispiel für Code, der Bildmasken abrufen und anzeigen, finden Sie unter [Informationen zur Klassifizierung und Segmentierung werden angezeigt](#)

Bereich der Anomalie

Verwenden Sie die prozentuale Abdeckung (TotalPercentageArea) einer Anomalie auf dem Bild. Sie könnten beispielsweise entscheiden, dass ein Produkt defekt ist, wenn die Fläche einer Anomalie mehr als 1% des Bildes ausmacht.

Python

Das vollständige Codebeispiel finden Sie unter [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;
}
```

Anzahl der Arten von Anomalien

Verwenden Sie die Anzahl der verschiedenen Anomalietypen (Name), die auf dem Bild gefunden wurden. Sie könnten beispielsweise entscheiden, dass ein Produkt defekt ist, wenn mehr als zwei Arten von Anomalien vorliegen.

Python

Das vollständige Codebeispiel finden Sie unter [GitHub](#).

```
def reject_on_anomaly_types(image, prediction, confidence_limit,
                             anomaly_types_limit):
    """
    Checks if the number of anomaly types is greater 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
prediction
                             (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;
}
```

Informationen zur Klassifizierung und Segmentierung werden angezeigt

Dieses Beispiel zeigt das analysierte Bild und überlagert die Analyseergebnisse. Wenn die Antwort eine Anomalienmaske enthält, wird die Maske in den Farben der zugehörigen Anomaliearten angezeigt.

Um Informationen zur Bildklassifizierung und Bildsegmentierung anzuzeigen

1. Sofern Sie das noch nicht getan haben, führen Sie die folgenden Schritte aus:
 - a. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
 - b. [Trainiere dein Modell](#).
 - c. [Starte dein Modell](#).

2. Stellen Sie sicher, dass der anrufende Benutzer Zugriff auf die Modellversion `DetectAnomalies` hat, die Sie verwenden möchten. Weitere Informationen finden Sie unter [Einrichten von SDK-Berechtigungen](#).
3. Verwenden Sie folgenden Code.

Python

Der folgende Beispielcode erkennt Anomalien in einem von Ihnen bereitgestellten Bild. Das Beispiel verwendet die folgenden Befehlszeilenoptionen:

- `project`— der Name des Projekts, das Sie verwenden möchten.
- `version`— die Version des Modells innerhalb des Projekts, die Sie verwenden möchten.
- `image`— der Pfad und die Datei einer lokalen Bilddatei (JPEG- oder PNG-Format).

```
# 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 occurred: " +
          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 occurred. " + format(err))
else:
    print("Successfully completed analysis.")

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

Java 2

Der folgende Beispielcode erkennt Anomalien in einem von Ihnen bereitgestellten Bild. Das Beispiel verwendet die folgenden Befehlszeilenoptionen:

- **project**— der Name des Projekts, das Sie verwenden möchten.
- **version**— die Version des Modells innerhalb des Projekts, die Sie verwenden möchten.
- **image**— der Pfad und die Datei einer lokalen Bilddatei (JPEG- oder PNG-Format).

```
/*
    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 = lfvcClient.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 specified 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 arbitrary 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. Wenn Sie nicht vorhaben, Ihr Modell weiter zu verwenden, [beenden Sie Ihr Modell](#).

Auffinden von Anomalien mit einer Funktion AWS Lambda

AWS Lambda ist ein Rechenservice, mit dem Sie Code ausführen können, ohne Server bereitzustellen oder zu verwalten. Sie können beispielsweise Bilder analysieren, die von einer mobilen Anwendung aus übermittelt wurden, ohne einen Server für den Anwendungscode einrichten zu müssen. Die folgenden Anweisungen zeigen, wie eine Lambda-Funktion in Python erstellt wird, die [DetectAnomalies](#) aufruft. Die Funktion analysiert ein bereitgestelltes Bild und gibt eine Klassifizierung für das Vorhandensein von Anomalien in diesem Bild zurück. Die Anweisungen enthalten einen Python-Beispielcode, der zeigt, wie die Lambda-Funktion mit einem Bild in einem Amazon-S3-Bucket oder einem von einem lokalen Computer bereitgestellten Bild aufgerufen wird.

Themen

- [Schritt 1: Erstellen Sie eine AWS Lambda Funktion \(Konsole\)](#)
- [Schritt 2: \(Optional\) Erstellen einer Ebene \(Konsole\)](#)
- [Schritt 3: Hinzufügen von Python-Code \(Konsole\)](#)
- [Schritt 4: Testen Ihrer Lambda-Funktion](#)

Schritt 1: Erstellen Sie eine AWS Lambda Funktion (Konsole)

In diesem Schritt erstellen Sie eine leere AWS Funktion und eine IAM-Ausführungsrolle, mit der Ihre Funktion den DetectAnomalies Vorgang aufrufen kann. Es gewährt auch Zugriff auf den Amazon-S3-Bucket, in dem Bilder zur Analyse gespeichert werden. Sie geben auch Umgebungsvariablen für Folgendes an:

- Das Projekt und die Modellversion von Amazon Lookout for Vision, die Ihre Lambda-Funktion verwenden soll.
- Das Konfidenzlimit, das das Modell verwenden soll.

Später fügen Sie der Lambda-Funktion den Quellcode und optional eine Ebene hinzu.

Um eine AWS Lambda Funktion (Konsole) zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter <https://console.aws.amazon.com/lambda/>.
2. Wählen Sie Funktion erstellen aus. Weitere Informationen zur Verwendung von Lambda finden Sie unter [Erstellen einer Lambda-Funktion mit der Konsole](#).
3. Wählen Sie aus den folgenden Optionen.
 - Wählen Sie Von Grund auf neu schreiben aus.
 - Geben Sie einen Wert für Name der Funktion ein.
 - Wählen Sie für Laufzeit die Option Python 3.10 aus.
4. Wählen Sie Funktion erstellen aus, um die AWS Lambda -Funktion zu erstellen.
5. Wählen Sie auf der Seite der Funktion den Tab Konfiguration aus.
6. Wählen Sie im Abschnitt Umgebungsvariablen Bearbeiten aus.
7. Fügen Sie die folgenden Umgebungsvariablen hinzu. Wählen Sie für jede Variable die Option Umgebungsvariable hinzufügen und geben Sie dann den Variablenschlüssel und den Wert ein.

Schlüssel	Value (Wert)
PROJEKTNAME	Das Lookout for Vision Vision-Projekt, das das Modell enthält, das Sie verwenden möchten.
MODELL_VERSION	Die Version des Modells, das Sie verwenden möchten.
KONFIDENZ	Der Mindestwert (0-100) für die Zuverlässigkeit des Modells, dass die Vorhersage anomal ist. Wenn die Konfidenz niedriger ist, gilt die Klassifizierung als normal.

8. Wählen Sie Speichern, um die Umgebungsvariablen zu speichern.
9. Wählen Sie im Bereich Berechtigungen unter Rollenname die Ausführungsrolle aus, um die Rolle in der IAM-Konsole zu öffnen.
10. Wählen Sie auf der Registerkarte Berechtigungen die Optionen Berechtigungen hinzufügen und dann Inline-Richtlinie erstellen aus.

11. Wählen Sie JSON aus und ersetzen Sie die vorhandene Richtlinie durch die folgende.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "lookoutvision:DetectAnomalies",
      "Resource": "*",
      "Effect": "Allow",
      "Sid": "DetectAnomaliesAccess"
    }
  ]
}
```

12. Wählen Sie Weiter.

13. Geben Sie im Feld Richtliniendetails einen Namen für die Richtlinie ein, z. B. DetectAnomalies-access.

14. Wählen Sie Create Policy (Richtlinie erstellen) aus.

15. Wenn Sie Bilder zur Analyse in einem Amazon-S3-Bucket speichern, wiederholen Sie die Schritte 10-14.

- a. Verwenden Sie für Schritt 11 die folgende Richtlinie. *bucket/folder path* Ersetzen Sie durch den Amazon S3 S3-Bucket und den Ordnerpfad zu den Bildern, die Sie analysieren möchten.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3Access",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::bucket/folder path/*"
    }
  ]
}
```

- b. Wählen Sie für Schritt 13 einen anderen Richtliniennamen aus, z. B. S3Bucket-access.

Schritt 2: (Optional) Erstellen einer Ebene (Konsole)

Um dieses Beispiel auszuführen, müssen Sie diesen Schritt nicht ausführen. Die `DetectAnomalies` Operation ist in der standardmäßigen Lambda-Python-Umgebung als Teil des AWS SDK für Python (Boto3) enthalten. Wenn andere Teile Ihrer Lambda-Funktion aktuelle AWS Service-Updates benötigen, die sich nicht in der Standard-Lambda-Python-Umgebung befinden, führen Sie diesen Schritt aus, um Ihrer Funktion die neueste Boto3-SDK-Version als Ebene hinzuzufügen.

Zunächst erstellen Sie ein ZIP-Dateiarchiv, das das Boto3-SDK enthält. Anschließend erstellen Sie eine Ebene und fügen der Ebene das ZIP-Dateiarchiv hinzu. Weitere Informationen finden Sie unter [Verwenden von Ebenen mit Ihrer Lambda-Funktion](#).

So erstellen Sie eine Ebene und fügen eine hinzu (Konsole)

1. Öffnen Sie eine Eingabeaufforderung und geben Sie dann die folgenden Befehle ein.

```
pip install boto3 --target python/.  
zip boto3-layer.zip -r python/
```

2. Notieren Sie sich den Namen der Zip-Datei (`boto3-layer.zip`). Sie benötigen ihm im Schritt 6 dieses Vorgangs.
3. Öffnen Sie die Konsole unter AWS Lambda <https://console.aws.amazon.com/lambda/>
4. Wählen Sie im Navigationsbereich Ebenen aus.
5. Wählen Sie Ebene erstellen aus.
6. Geben Sie einen Namen und eine Beschreibung ein.
7. Wählen Sie Zip-Datei hochladen und dann Upload aus.
8. Wählen Sie im Dialogfeld das ZIP-Dateiarchiv (`boto3-layer.zip`) aus, das Sie in Schritt 1 dieses Vorgangs erstellt haben.
9. Wählen Sie für kompatible Laufzeiten Python 3.9.
10. Wählen Sie Erstellen, um die Ebene zu erstellen.
11. Wählen Sie das Menüsymbol im Navigationsbereich.
12. Wählen Sie im Navigationsbereich Funktionen aus.
13. Wählen Sie in der Ressourcenliste die Funktion aus, die Sie in [Schritt 1: Erstellen Sie eine AWS Lambda Funktion \(Konsole\)](#) erstellt haben.
14. Wählen Sie die Registerkarte Code.

15. Wählen Sie im Bereich Ebenen die Option Ebene hinzufügen aus.
16. Wählen Sie Benutzerdefinierte Ebenen.
17. Wählen Sie unter Benutzerdefinierte Ebenen den Namen der Ebene aus, den Sie in Schritt 6 eingegeben haben.
18. Wählen Sie unter Version die Version der Ebene aus, die 1 sein sollte.
19. Wählen Sie Hinzufügen aus.

Schritt 3: Hinzufügen von Python-Code (Konsole)

In diesem Schritt fügen Sie Python-Code zu Ihrer Lambda-Funktion hinzu, indem Sie den Code-Editor der Lambda-Konsole verwenden. Der Code analysiert ein bereitgestelltes Bild mit DetectAnomalies und gibt eine Klassifizierung zurück (true, wenn das Bild anomal ist, false, wenn das Bild normal ist). Das bereitgestellte Bild kann sich in einem Amazon-S3-Bucket befinden oder als Base64-kodierte Bildbytes bereitgestellt werden.

So fügen Sie einen Python-Code hinzu (Konsole)

1. Wenn Sie nicht in der Lambda-Konsole sind, gehen Sie wie folgt vor:
 - a. Öffnen Sie die AWS Lambda Konsole unter. <https://console.aws.amazon.com/lambda/>
 - b. Öffnen Sie die Lambda-Funktion, die Sie in [Schritt 1: Erstellen Sie eine AWS Lambda Funktion \(Konsole\)](#) erstellt haben.
2. Wählen Sie die Registerkarte Code.
3. Ersetzen Sie in der Codequelle den Code in lambda_function.py durch den folgenden Code:

```
# 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(enviro['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. Wählen Sie Implementieren aus, um Ihre Lambda-Funktion bereitzustellen.

Schritt 4: Testen Ihrer Lambda-Funktion

In diesem Schritt verwenden Sie Python-Code auf Ihrem Computer, um ein lokales Bild oder ein Bild in einem Amazon-S3-Bucket an Ihre Lambda-Funktion zu übergeben. Bilder, die von einem lokalen Computer übertragen werden, müssen kleiner als 6.291.456 Byte sein. Wenn Ihre Bilder größer sind, laden Sie die Bilder in einen Amazon-S3-Bucket hoch und rufen Sie das Skript mit dem Amazon S3-Pfad zum Bild auf. Weitere Informationen zum Hochladen von Dateien auf einen Amazon-S3-Bucket finden Sie unter [Hochladen von Objekten](#).

Stellen Sie sicher, dass Sie den Code in derselben AWS Region ausführen, in der Sie die Lambda-Funktion erstellt haben. Sie können die AWS Region für Ihre Lambda-Funktion in der Navigationsleiste der Funktionsdetailseite in der [Lambda-Konsole](#) anzeigen.

Wenn die AWS Lambda Funktion einen Timeout-Fehler zurückgibt, verlängern Sie den Timeout-Zeitraum für die Lambda-Funktionsfunktion. Weitere Informationen finden Sie unter [Funktionstimeout konfigurieren](#) (Konsole).

Weitere Informationen zum Aufrufen einer Lambda-Funktion aus Ihrem Code finden Sie unter Funktionen [aufrufen AWS Lambda](#).

So testen Sie Ihre Lambda-Funktion

1. Sofern Sie das noch nicht getan haben, führen Sie die folgenden Schritte aus:

- a. Stellen Sie sicher, dass der Benutzer, der den Client-Code verwendet, über die entsprechenden Rechte verfügt. `lambda:InvokeFunction` Sie können die folgenden Berechtigungen verwenden.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LambdaPermission",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "ARN for lambda function"
    }
  ]
}
```

Sie können den ARN für Ihre Lambda-Funktion in der Funktionsübersicht in der [Lambda-Konsole](#) abrufen.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

- b. Installieren und konfigurieren Sie AWS das SDK für Python. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
 - c. [Starten Sie das Modell](#), das Sie in Schritt 7 von [Schritt 1: Erstellen Sie eine AWS Lambda Funktion \(Konsole\)](#) angegeben haben.
2. Speichern Sie den folgenden Code in eine Datei mit dem Namen `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. Führen Sie den Code aus. Geben Sie für das Befehlszeilenargument den Namen der Lambda-Funktion und den Pfad zu einem lokalen Bild an, das Sie analysieren möchten. Zum Beispiel:

```
python client.py function_name /bucket/path/image.jpg
```

Bei Erfolg handelt es sich bei der Ausgabe um eine Klassifizierung der im Bild gefundenen Anomalien. Wenn keine Klassifizierung zurückgegeben wird, sollten Sie erwägen, den Konfidenzwert zu verringern, den Sie in Schritt 7 von festgelegt haben. [Schritt 1: Erstellen Sie eine AWS Lambda Funktion \(Konsole\)](#)

4. Wenn Sie mit der Lambda-Funktion fertig sind und das Modell nicht von anderen Anwendungen verwendet wird, [beenden Sie das Modell](#). Denken Sie daran, [das Modell zu starten](#), wenn Sie das nächste Mal die Lambda-Funktion verwenden möchten.

Verwenden Ihres Amazon Lookout for Vision Vision-Modells auf einem Edge-Gerät

Sie können Ihr Amazon Lookout for Vision Vision-Modell auf Edge-Geräten verwenden, die von verwaltet werden AWS IoT Greengrass Version 2. AWS IoT Greengrass ist ein Open-Source-Edge-Runtime- und Cloud-Service für das Internet der Dinge (IoT). Sie können damit IoT-Anwendungen auf Ihren Geräten erstellen, bereitstellen und verwalten. Weitere Informationen finden Sie unter [AWS IoT Greengrass](#).

Sie stellen dieselben Amazon Lookout for Vision Vision-Modelle, die Sie in der Cloud trainiert haben, auf AWS IoT Greengrass V2 kompatiblen Edge-Geräten bereit. Anschließend können Sie Ihr bereitgestelltes Modell verwenden, um Anomalien vor Ort, z. B. in einer Werkshalle, zu erkennen, ohne ständig Daten in die Cloud streamen zu müssen. Auf diese Weise können Sie die Bandbreitenkosten minimieren und Anomalien mithilfe der Bildanalyse in Echtzeit lokal erkennen.

Tip

Bevor Sie ein Lookout for Vision Vision-Modell mit bereitstellen AWS IoT Greengrass, empfehlen wir Ihnen, das AWS IoT Greengrass Version 2 Entwicklerhandbuch zu lesen. Weitere Informationen finden Sie unter [Was ist AWS IoT Greengrass?](#) .

Um ein Lookout for Vision Vision-Modell auf einem AWS IoT Greengrass V2 Kerngerät zu verwenden, stellen Sie das Modell und die unterstützende Software als Komponenten auf dem Kerngerät bereit. Eine Komponente ist ein Softwaremodul, z. B. ein Modell von Lookout for Vision, das auf einem Greengrass-Core-Gerät ausgeführt wird. Es gibt zwei Arten von Komponenten. Eine benutzerdefinierte Komponente ist eine Komponente, die Sie erstellen und auf die nur Sie zugreifen können. Sie wird auch als private Komponente bezeichnet. Eine AWS mitgelieferte Komponente ist eine vorgefertigte Komponente, die Folgendes AWS bietet. Sie wird auch als öffentliche Komponente bezeichnet. Weitere Informationen finden Sie unter <https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html>.

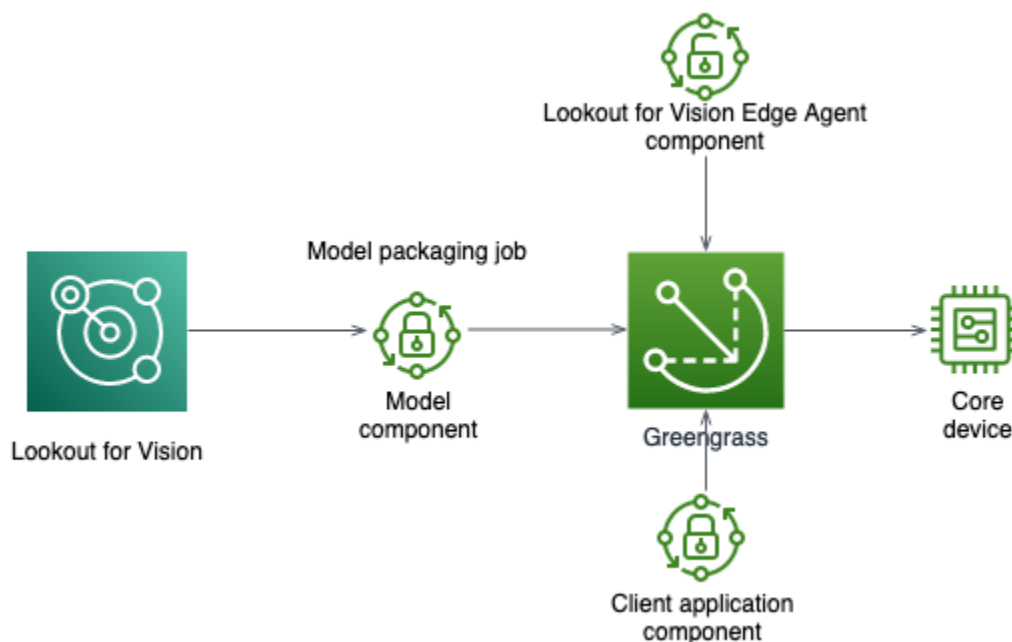
Die Komponenten, die Sie auf einem Kerngerät für ein Lookout for Vision Vision-Modell und unterstützende Software bereitstellen, sind:

- **Modellkomponente.** Eine benutzerdefinierte Komponente, die Ihr Lookout for Vision Vision-Modell enthält. Um die Modellkomponente zu erstellen, verwenden Sie Lookout for Vision, um einen

Modellverpackungsauftrag zu erstellen. Ein Auftrag zur Modellverpackung erstellt eine Komponente für das Modell und stellt sie als benutzerdefinierte Komponente innerhalb AWS IoT Greengrass V2 des Modells zur Verfügung. Weitere Informationen finden Sie unter [Verpacken Ihres Amazon Lookout for Vision Vision-Modells](#).

- Komponente der Client-Anwendung. Eine von Ihnen erstellte benutzerdefinierte Komponente, die den Code für Ihre Geschäftsanforderungen implementiert. Zum Beispiel das Auffinden ungewöhnlicher Leiterplatten anhand von Bildern, die nach der Montage aufgenommen wurden. Weitere Informationen finden Sie unter [Schreiben Sie Ihre Client-Anwendungskomponente](#).
- Komponente Amazon Lookout for Vision Edge Agent. Eine AWS mitgelieferte Komponente, die eine API für die Verwendung und Verwaltung Ihres Modells bereitstellt. Beispielsweise kann Code in Ihrer Client-Anwendungskomponente die DetectAnomalies API verwenden, um Anomalien in Bildern zu erkennen. Die Komponente Lookout for Vision Edge Agent ist eine Abhängigkeit von der Modellkomponente. Sie wird automatisch auf dem Kerngerät installiert, wenn Sie die Modellkomponente bereitstellen. Weitere Informationen finden Sie unter [Referenz zur Amazon Lookout for Vision Edge Agent-API](#).

Nachdem Sie die Modellkomponente und die Client-Anwendungskomponente erstellt haben, können Sie AWS IoT Greengrass V2 sie verwenden, um die Komponenten und Abhängigkeiten auf dem Kerngerät bereitzustellen. Weitere Informationen finden Sie unter [Bereitstellen Ihrer Komponenten auf einem Gerät](#).



 Important

Die Vorhersagen, die Ihr Modell DetectAnomalies auf einem Kerngerät macht, können sich von Vorhersagen unterscheiden, die mit demselben in der Cloud gehosteten Modell gemacht wurden. Wir empfehlen Ihnen, Ihr Modell auf einem Kerngerät zu testen, bevor Sie es in einer Produktionsumgebung verwenden.

Wir empfehlen, die Anzahl normaler und anomaler Bilder in Ihrem Trainingsdatensatz zu erhöhen, um die Prognoseabweichungen zwischen gerätegestützten Modellen und Cloud-Modellen zu verringern. Wir raten davon ab, vorhandene Bilder wiederzuverwenden, um den Trainingsdatensatz zu vergrößern.

Bereitstellen eines Modells und einer Client-Anwendungskomponente auf einem AWS IoT Greengrass Version 2 Kerngerät

Das Verfahren für die Bereitstellung eines Amazon Lookout for Vision Vision-Modells und einer Client-Anwendungskomponente auf einem AWS IoT Greengrass Version 2 Kerngerät lautet wie folgt:

1. [Richten Sie Ihre Kerngeräte](#) mit AWS IoT Greengrass Version 2 ein.
2. [Erstellen Sie mithilfe von Lookout for Vision einen Auftrag zur Modellverpackung](#). Der Job erstellt Ihre Modellkomponente.
3. [Schreiben Sie eine Client-Anwendungskomponente](#). Die Komponente implementiert Ihre Geschäftslogik.
4. [Stellen Sie die Modellkomponente und die Client-Anwendungskomponente](#) auf dem Kerngerät bereit, indem Sie AWS IoT Greengrass V2.

Nachdem die Komponenten und Abhängigkeiten auf dem Kerngerät bereitgestellt wurden, können Sie das Modell auf dem Kerngerät verwenden.

 Note

Sie müssen dieselbe AWS Region und dasselbe AWS Konto verwenden, um Ihr Lookout for Vision Vision-Modell und Ihre Client-Anwendungskomponente zu erstellen und bereitzustellen.

AWS IoT Greengrass Version 2 zentrale Geräteanforderungen

Um ein Amazon Lookout for Vision Vision-Modell auf einem AWS IoT Greengrass Version 2 Kerngerät zu verwenden, muss Ihr Modell verschiedene Anforderungen an das Kerngerät stellen.

Themen

- [Getestete Geräte, Chiparchitekturen und Betriebssysteme](#)
- [Speicher und Speicher des Kerngeräts](#)
- [Erforderliche Software](#)

Getestete Geräte, Chiparchitekturen und Betriebssysteme

Wir gehen davon aus, dass Amazon Lookout for Vision auf der folgenden Hardware funktioniert:

- CPU-Architekturen
 - X86_64 (64-Bit-Version des x86-Befehlssatzes)
 - Aarch64 (64-Bit-CPU) ARMv8
- (Nur GPU-beschleunigte Inferenz) NVIDIA GPU-Beschleuniger mit ausreichender Speicherkapazität (mindestens 6,0 GB für ein laufendes Modell).

Das Amazon Lookout for Vision Vision-Team hat Lookout for Vision Vision-Modelle auf den folgenden Geräten, Chip-Architekturen und Betriebssystemen getestet.

Geräte

Gerät	Betriebssystem	Architektur	Accelerator	Compiler-Optionen
jetson_xavier (NVIDIA® Jetson AGX Xavier)	Linux	Aarch 64	NVIDIA	{ "gpu-code": "sm_72", "trt-ver": "7.1.3", "cuda-ver": "10.2" }

Gerät	Betriebssystem	Architektur	Accelerator	Compiler-Optionen
				<code>{"gpu-code": "sm_72", "trt-ver": "8.2.1", "cuda-ver": "10.2"}</code>
g4dn.xlarge (EC2 Instanzen (G4) mit NVIDIA T4 Tensor Core) GPUs	Linux	X86_64/X86-64	NVIDIA	<code>{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}</code>
g5.xlarge (Instanzen (G5) mit NVIDIA A10G Tensor Core) EC2 GPUs	Linux	X86_64/X86-64	NVIDIA	<code>{"gpu-code": "sm_80", "trt-ver": "8.2.0", "cuda-ver": "11.2"}</code>
c5.2xlarge (Amazon EC2 C5-Instances)	Linux	X86_64/X86-64	CPU	<code>{"mcpu": "core-avx2"}</code>

Speicher und Speicher des Kerngeräts

Um ein einzelnes Modell und den Amazon Lookout for Vision Edge Agent auszuführen, hat Ihr Kerngerät die folgenden Speicher- und Speicheranforderungen. Möglicherweise benötigen Sie mehr Arbeitsspeicher und Speicherplatz für Ihre Client-Anwendungskomponente.

- Speicher — mindestens 1,5 GB.
- Arbeitsspeicher — Mindestens 6,0 GB für ein laufendes Modell.

Erforderliche Software

Für ein Kerngerät ist die folgende Software erforderlich.

Jetson-Geräte

Wenn es sich bei Ihrem Core-Gerät um ein Jetson-Gerät handelt, muss die folgende Software auf dem Core-Gerät installiert sein.

Software	Unterstützte Versionen
Jetpack SDK	4.4 bis 4.6.1
Virtuelle Python- und Python-Umgebung für Lookout for Vision Edge Agent Version 1.x	3.8 oder 3.9

X86-Hardware

Wenn Ihr Core-Gerät x86-Hardware verwendet, muss die folgende Software auf dem Core-Gerät installiert sein.

CPU-Inferenz

Software	Unterstützte Versionen
Virtuelle Python- und Python-Umgebung für Lookout for Vision Edge Agent Version 1.x	3.8 oder 3.9

GPU-beschleunigte Inferenz

Die Softwareversionen variieren je nach der Mikroarchitektur der verwendeten NVIDIA-GPU.

NVIDIA-GPU mit einer Mikroarchitektur vor Ampere (Rechenkapazität ist geringer als 8,0)

Erforderliche Software für eine NVIDIA-GPU mit einer Mikroarchitektur vor Ampere (Rechenkapazität von weniger als 8,0). Der `gpu-code` muss kleiner sein als `sm_80`

Software	Unterstützte Versionen
NVIDIA CUDA	10.2
NVIDIA TensorRT	Mindestens 7.1.3 und weniger als 8.0.0
Virtuelle Python- und Python-Umgebung für Lookout for Vision Edge Agent Version 1.x	3.8 oder 3.9

NVIDIA-GPU mit Ampere-Mikroarchitektur (Rechenleistung 8.0)

Erforderliche Software für eine NVIDIA-GPU mit Ampere-Mikroarchitektur (Rechenkapazität ist 8,0). Das `gpu-code` muss sein `sm_80`.

Software	Unterstützte Versionen
NVIDIA CUDA	11.2
NVIDIA TensorRT	8.2.0
Virtuelle Python- und Python-Umgebung für Lookout for Vision Edge Agent Version 1.x	3.8 oder 3.9

Richten Sie Ihr AWS IoT Greengrass Version 2 Core-Gerät ein

Amazon Lookout for Vision verwendet AWS IoT Greengrass Version 2 , um die Bereitstellung der Modellkomponente, der Amazon Lookout for Vision Edge Agent-Komponente und der Client-Anwendungskomponente auf Ihrem AWS IoT Greengrass V2 Kerngerät zu vereinfachen. Informationen zu den Geräten und der Hardware, die Sie verwenden können, finden Sie unter. [AWS IoT Greengrass Version 2 zentrale Geräteanforderungen](#)

Ihr Core-Gerät einrichten

Verwenden Sie die folgenden Informationen, um Ihr Kerngerät einzurichten.

So richten Sie Ihr Core-Gerät ein

1. Richten Sie Ihre GPU-Bibliotheken ein. Führen Sie diesen Schritt nicht aus, wenn Sie keine GPU-beschleunigte Inferenz verwenden.
 - a. Stellen Sie sicher, dass Sie über eine GPU verfügen, die CUDA unterstützt. Weitere Informationen finden Sie unter [Überprüfen, ob Sie über eine CUDA-fähige GPU verfügen](#).
 - b. Richten Sie CUDA, cuDNN und TensorRT auf Ihrem Gerät ein, indem Sie einen der folgenden Schritte ausführen:
 - Wenn Sie ein Jetson-Gerät verwenden, installieren Sie Version 4.4 - 4.6.1. JetPack
[Weitere Informationen finden Sie unter JetPack Archivieren](#).
 - Wenn Sie x86-basierte Hardware verwenden und Ihre NVIDIA-GPU-Mikroarchitektur älter als Ampere ist (die Rechenkapazität liegt unter 8,0), gehen Sie wie folgt vor:
 1. Richten Sie CUDA Version 10.2 ein, indem Sie den Anweisungen im [NVIDIA CUDA-Installationshandbuch](#) für Linux folgen.
 2. Installieren Sie cuDNN, indem Sie den Anweisungen in der [NVIDIA cuDNN-Dokumentation](#) folgen.
 3. [Richten Sie TensorRT \(Version 7.1.3 oder höher, aber früher als 8.0.0\) ein, indem Sie den Anweisungen unter NVIDIA TENSORRT DOCUMENTATION folgen](#).
 - Wenn Sie x86-basierte Hardware verwenden und Ihre NVIDIA-GPU-Mikroarchitektur Ampere ist (Rechenkapazität ist 8.0), gehen Sie wie folgt vor:
 1. Richten Sie CUDA (Version 11.2) ein, indem Sie den Anweisungen im [NVIDIA CUDA-Installationshandbuch](#) für Linux folgen.
 2. Installieren Sie cuDNN, indem Sie den Anweisungen in der [NVIDIA cuDNN-Dokumentation](#) folgen.
 3. Richten Sie TensorRT (Version 8.2.0) ein, indem Sie den Anweisungen unter [NVIDIA TENSORRT DOCUMENTATION](#) folgen.
2. Installieren Sie die AWS IoT Greengrass Version 2 Kernsoftware auf Ihrem Kerngerät. Weitere Informationen finden [Sie unter Installieren der AWS IoT Greengrass Core-Software](#) im AWS IoT Greengrass Version 2 Entwicklerhandbuch.

3. Um aus dem Amazon S3 S3-Bucket zu lesen, in dem das Modell gespeichert ist, fügen Sie der IAM-Rolle (Token-Exchange-Rolle), die Sie während der AWS IoT Greengrass Version 2 Einrichtung erstellen, eine Berechtigung hinzu. Weitere Informationen finden Sie unter [Zugriff auf S3-Buckets für Komponentenartefakte zulassen](#).
4. Geben Sie in der Befehlszeile den folgenden Befehl ein, um Python und eine virtuelle Python-Umgebung auf dem Core-Gerät zu installieren.

```
sudo apt install python3.8 python3-venv python3.8-venv
```

5. Verwenden Sie den folgenden Befehl, um den Greengrass-Benutzer zur Videogruppe hinzuzufügen. Dadurch können von Greengrass bereitgestellte Komponenten auf die GPU zugreifen:

```
sudo usermod -a -G video ggc_user
```

6. (Optional) Wenn Sie die Agenten-API von Lookout for Vision Edge von einem anderen Benutzer aus aufrufen möchten, fügen Sie den erforderlichen Benutzer dem ggc_group hinzu. Dadurch kann der Benutzer mit dem Lookout for Vision Edge Agent über den Unix-Domain-Socket kommunizieren:

```
sudo usermod -a -G ggc_group $(whoami)
```

Verpacken Ihres Amazon Lookout for Vision Vision-Modells

Ein Auftrag zum Verpacken von Modellen verpackt ein Amazon Lookout for Vision Vision-Modell als Modellkomponente.

Um einen Auftrag zur Modellverpackung zu erstellen, wählen Sie das Modell aus, das Sie verpacken möchten, und geben Einstellungen für die Modellkomponente an, die durch den Auftrag erstellt wird. Sie können nur ein Modell verpacken, das erfolgreich trainiert wurde.

Sie können die Lookout for Vision Vision-Konsole oder AWS das SDK verwenden, um den Auftrag zur Modellverpackung zu erstellen. Sie können auch Informationen zu den von Ihnen erstellten Modellverpackungsaufträgen abrufen. Weitere Informationen finden Sie unter [Informationen zu Paketierungsaufträgen für Modelle abrufen](#). Sie können die AWS IoT Greengrass V2 Konsole oder das AWS SDK verwenden, um die Komponenten auf dem AWS IoT Greengrass Version 2 Kerngerät bereitzustellen.

Themen

- [Einstellungen für das Package](#)
- [Verpacken Sie Ihr Modell \(Konsole\)](#)
- [Verpacken Sie Ihr Modell \(SDK\)](#)
- [Informationen zu Paketierungsaufträgen für Modelle abrufen](#)

Einstellungen für das Package

Verwenden Sie die folgenden Informationen, um die Paketeinstellungen für Ihren Modellverpackungsauftrag festzulegen.

Informationen zum Erstellen eines Modellverpackungsauftrags finden Sie unter [Verpacken Sie Ihr Modell \(Konsole\)](#) oder [Verpacken Sie Ihr Modell \(SDK\)](#).

Themen

- [Zielhardware](#)
- [Einstellungen der Komponenten](#)

Zielhardware

Sie können ein Zielgerät oder eine Zielplattform für Ihr Modell wählen, aber nicht beides. Weitere Informationen finden Sie unter [Getestete Geräte, Chiparchitekturen und Betriebssysteme](#).

Zielgerät

Das Zielgerät für das Modell, z. B. [NVIDIA® Jetson AGX Xavier](#). Sie müssen keine Compiler-Optionen angeben.

Zielplattform

Amazon Lookout for Vision unterstützt die folgenden Plattformkonfigurationen:

- X86_64-Architekturen (64-Bit-Version des x86-Befehlssatzes) und Aarch64-Architekturen (64-Bit-CPU). ARMv8
- Linux-Betriebssystem.
- Inferenz mit NVIDIA- oder CPU-Beschleunigern.

Sie müssen die richtigen Compiler-Optionen für Ihre Zielplattform angeben.

Compiler-Optionen

Mit den Compiler-Optionen können Sie die Zielplattform für Ihr AWS IoT Greengrass Version 2 Kerngerät angeben. Derzeit können Sie die folgenden Compiler-Optionen angeben.

NVIDIA-Beschleuniger

- `gpu-code`— Gibt den GPU-Code des Kerngeräts an, auf dem die Modellkomponente ausgeführt wird.
- `trt-ver`— Spezifiziert die TensorRT-Version im x.y.z-Format.
- `cuda-ver`— Spezifiziert die CUDA-Version im X.Y-Format.

CPU-Beschleuniger

- (Optional) `mcpu` — gibt den Befehlssatz an. Zum Beispiel `core-avx2`. Wenn Sie keinen Wert angeben, verwendet Lookout for Vision den Wert `core-avx2`.

Sie geben die Optionen im JSON-Format an. Zum Beispiel:

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

Weitere Beispiele finden Sie unter [Getestete Geräte, Chiparchitekturen und Betriebssysteme](#).

Einstellungen der Komponenten

Beim Paketieren des Modells wird eine Modellkomponente erstellt, die Ihr Modell enthält. Der Job erzeugt Artefakte, die zur Bereitstellung der Modellkomponente auf dem Kerngerät AWS IoT Greengrass V2 verwendet werden.

Sie können keine Modellkomponente mit demselben Komponentennamen und derselben Komponentenversion wie eine vorhandene Komponente erstellen.

Name der Komponente

Ein Name für die Modellkomponente, die Lookout for Vision beim Paketieren des Modells erstellt. Der von Ihnen angegebene Komponentename wird in der AWS IoT Greengrass V2 Konsole

angezeigt. Sie verwenden den Komponentennamen in dem Rezept, das Sie für die Client-Anwendungskomponente erstellen. Weitere Informationen finden Sie unter [Die Komponente der Client-Anwendung wird erstellt](#).

Beschreibung der Komponente

(Optional) Eine Beschreibung für die Modellkomponente.

Version der Komponente

Eine Versionsnummer für die Modellkomponente. Sie können die Standardversionsnummer akzeptieren oder eine eigene wählen. Die Versionsnummer muss dem semantischen Versionsnummernsystem entsprechen — major.minor.patch. Beispielsweise stellt Version 1.0.0 die erste Hauptversion für eine Komponente dar. Weitere Informationen finden Sie unter [Semantic Versioning 2.0.0](#). Wenn Sie keinen Wert angeben, verwendet Lookout for Vision die Versionsnummer Ihres Modells, um eine Version für Sie zu generieren.

Position der Komponente

Der Amazon S3 S3-Speicherort, an dem der Modellverpackungsauftrag die Modellkomponenten-Artefakte speichern soll. Der Amazon S3 S3-Bucket muss sich in derselben AWS-Region und demselben AWS-Konto befinden, in dem Sie ihn verwenden AWS IoT Greengrass Version 2. Informationen zum Erstellen eines Amazon S3 S3-Buckets finden Sie unter [Bucket erstellen](#).

Tags

Sie können Ihre Komponenten mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht. Die Tags werden an die Modellkomponente angehängt, wenn der Modellverpackungsjob die Modellkomponente in Greengrass erstellt. Eine Komponente ist eine AWS IoT Greengrass V2-Ressource. Die Tags sind keiner Ihrer Lookout for Vision Vision-Ressourcen zugeordnet, z. B. Ihren Modellen. Weitere Informationen finden Sie unter [Taggen von AWS-Ressourcen](#).

Verpacken Sie Ihr Modell (Konsole)

Sie können mithilfe der Amazon Lookout for Vision-Konsole einen Auftrag zur Modellverpackung erstellen.

Informationen zu Paketeinstellungen finden Sie unter [Einstellungen für das Package](#).

So verpacken Sie ein Modell (Konsole)

1. [Erstellen Sie einen Amazon S3 S3-Bucket](#) oder verwenden Sie einen vorhandenen Bucket wieder, den Lookout for Vision zum Speichern der Verpackungsauftragsartefakte (Modellkomponente) verwendet.
2. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie im Abschnitt Projekte das Projekt aus, das das Modell enthält, das Sie verpacken möchten.
6. Wählen Sie im linken Navigationsbereich unter dem Projektnamen die Option Edge-Modellpakete aus.
7. Wählen Sie im Abschnitt Model-Packaging-Jobs die Option Model-Packaging-Job erstellen aus.
8. Geben Sie die Einstellungen für das Paket ein. Weitere Informationen finden Sie unter [Einstellungen für das Package](#).
9. Wählen Sie Modellverpackungsauftrag erstellen aus.
10. Warten Sie, bis der Verpackungsauftrag abgeschlossen ist. Der Job ist abgeschlossen, wenn der Status des Jobs Erfolgreich lautet.
11. Wählen Sie den Paketierungsauftrag im Abschnitt Verpackungsaufträge modellieren aus.
12. Wählen Sie Continue deployment in Greengrass, um die Bereitstellung Ihrer Modellkomponente in AWS IoT Greengrass Version 2 fortzusetzen. Weitere Informationen finden Sie unter [Bereitstellen Ihrer Komponenten auf einem Gerät](#).

Verpacken Sie Ihr Modell (SDK)

Sie verpacken ein Modell als Modellkomponente, indem Sie einen Auftrag zum Paketieren eines Modells erstellen. Um einen Model-Packaging-Job zu erstellen, rufen Sie die [StartModelPackagingJob](#)API auf. Es kann eine Weile dauern, bis der Job abgeschlossen ist. Um den aktuellen Status zu erfahren, rufen Sie an [DescribeModelPackagingJob](#)und überprüfen Sie das Status Feld in der Antwort.

Informationen zu Paketeinstellungen finden Sie unter [Einstellungen für das Package](#).

Das folgende Verfahren zeigt Ihnen, wie Sie einen Paketierungsauftrag mithilfe der AWS CLI starten. Sie können das Modell für eine Zielplattform oder ein Zielgerät verpacken. Ein Beispiel für Java-Code finden Sie unter [StartModelPackagingJob](#).

So verpacken Sie Ihr Modell (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Stellen Sie sicher, dass Sie über die richtigen Berechtigungen verfügen, um einen Model-Packaging-Job zu starten. Weitere Informationen finden Sie unter [StartModelPackagingJob](#).
3. Verwenden Sie die folgenden CLI-Befehle, um Ihr Modell entweder für ein Zielgerät oder eine Zielplattform zu verpacken.

Target platform

Der folgende CLI-Befehl zeigt, wie Sie ein Modell für eine Zielplattform mit einem NVIDIA-Beschleuniger verpacken.

Ändern Sie die folgenden Werte:

- `project_name` zum Namen des Projekts, das das Modell enthält, das Sie verpacken möchten.
- `model_version` zu der Version des Modells, das Sie verpacken möchten.
- (Optional) `description` zu einer Beschreibung für Ihren Auftrag zum Verpacken Ihres Modells.
- `architecture` zur Architektur (ARM64 oder X86_64) des AWS IoT Greengrass Version 2 Kerngeräts, auf dem Sie die Modellkomponente ausführen.
- `gpu_code` zum GPU-Code des Kerngeräts, auf dem Sie die Modellkomponente ausführen.
- `trt_ver` auf die TensorRT-Version, die Sie auf Ihrem Core-Gerät installiert haben.
- `cuda_ver` zu der CUDA-Version, die Sie auf Ihrem Core-Gerät installiert haben.
- `component_name` zu einem Namen für die Modellkomponente, auf AWS IoT Greengrass V2 der Sie erstellen möchten.
- (Optional) `component_version` zu einer Version für die Modellkomponente, die der Paketierungsauftrag erstellt. Verwenden Sie dabei das Format `major.minor.patch`. Beispielsweise steht 1.0.0 für die erste Hauptversion einer Komponente.

- bucket in den Amazon S3 S3-Bucket, in dem der Verpackungsauftrag die Modellkomponenten-Artefakte speichert.
- prefix in den Ort innerhalb des Amazon S3 S3-Buckets, an dem der Verpackungsauftrag die Modellkomponenten-Artefakte speichert.
- (Optional) component_description zu einer Beschreibung der Modellkomponente.
- (Optional) tag_key1 und tag_key2 zu den Schlüsseln für Beschriftungen, die an die Modellkomponente angehängt sind.
- (Optional) tag_value1 und tag_value2 zu den Schlüsselwerten für die Tags, die an die Modellkomponente angehängt sind.

```
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'},CompilerOptions={\"code\": \"gpu_code\", \"trt-ver\": \"trt_ver\", \"cuda-ver\": \"cuda_ver\"}},S3OutputLocation={Bucket='bucket',Prefix='prefix'},ComponentName='ComponentName',Tags=[{Key='tag_key2',Value='tag_value2'}]]" \
  --profile lookoutvision-access
```

Zum Beispiel:

```
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'},CompilerOptions={\"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',ComponentDescription='is my component',Tags=[{Key='modelKey0',Value='modelValue'},{Key='modelKey1',Value='modelValue'}]]" \
  --profile lookoutvision-access
```

Target Device

Verwenden Sie die folgenden CLI-Befehle, um ein Modell für ein Zielgerät zu verpacken.

Ändern Sie die folgenden Werte:

- `project_name` auf den Namen des Projekts, das das Modell enthält, das Sie verpacken möchten.
- `model_version` zu der Version des Modells, das Sie verpacken möchten.
- (Optional) `description` zu einer Beschreibung für Ihren Auftrag zum Verpacken Ihres Modells.
- `component_name` zu einem Namen für die Modellkomponente, auf der Sie etwas erstellen möchten AWS IoT Greengrass V2.
- (Optional) `component_version` zu einer Version für die Modellkomponente, die der Paketierungsauftrag erstellt. Verwenden Sie dabei das Format `major.minor.patch`. Beispielsweise steht 1.0.0 für die erste Hauptversion einer Komponente.
- `bucket` in den Amazon S3 S3-Bucket, in dem der Verpackungsauftrag die Modellkomponenten-Artefakte speichert.
- `prefix` an den Ort innerhalb des Amazon S3 S3-Buckets, an dem der Verpackungsauftrag die Modellkomponenten-Artefakte speichert.
- (Optional) `component_description` zu einer Beschreibung der Modellkomponente.
- (Optional) `tag_key1` und `tag_key2` zu den Schlüsseln für Beschriftungen, die an die Modellkomponente angehängt sind.
- (Optional) `tag_value1` und `tag_value2` zu den Schlüsselwerten für die Tags, die an die Modellkomponente angehängt sind.

```
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
```

Zum Beispiel:

```
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. Notieren Sie sich den Wert von JobName in der Antwort. Sie benötigen ihn im nächsten Schritt.
Zum Beispiel:

```
{  
  "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"  
}
```

5. DescribeModelPackagingJobDient zum Abrufen des aktuellen Status des Jobs. Ändern Sie Folgendes:

- `project_name` in den Namen des Projekts, das Sie verwenden.
- `job_name` auf den Namen des Jobs, den Sie im vorherigen Schritt notiert haben.

```
aws lookoutvision describe-model-packaging-job \  
  --project-name project_name \  
  --job-name job_name \  
  --profile lookoutvision-access
```

Der Auftrag zum Paketieren des Modells ist abgeschlossen, wenn der Wert von Status ist `SUCCEEDED`. Wenn der Wert anders ist, warten Sie eine Minute und versuchen Sie es erneut.

6. Setzen Sie die Bereitstellung fort mit AWS IoT Greengrass V2. Weitere Informationen finden Sie unter [Bereitstellen Ihrer Komponenten auf einem Gerät](#).

Informationen zu Paketierungsaufträgen für Modelle abrufen

Sie können die Amazon Lookout for Vision Vision-Konsole und AWS das SDK verwenden, um Informationen zu den von Ihnen erstellten Modellverpackungsaufträgen abzurufen.

Themen

- [Informationen zu Aufträgen zum Verpacken von Modellen abrufen \(Konsole\)](#)
- [Informationen zum Modellverpackungsauftrag \(SDK\) werden abgerufen](#)

Informationen zu Aufträgen zum Verpacken von Modellen abrufen (Konsole)

Um Auftragsinformationen zur Modellpaketierung abzurufen (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie im Abschnitt Projekte das Projekt aus, das den Modellverpackungsauftrag enthält, den Sie sich ansehen möchten.
5. Wählen Sie im linken Navigationsbereich unter dem Projektnamen die Option Edge-Modellpakete aus.
6. Wählen Sie im Abschnitt Auftrag zur Modellpaketierung den Modellverpackungsauftrag aus, den Sie anzeigen möchten. Die Detailseite für den Auftrag zum Verpacken des Modells wird angezeigt.

Informationen zum Modellverpackungsauftrag (SDK) werden abgerufen

Sie können das AWS SDK verwenden, um die Modellverpackungsaufträge in einem Projekt aufzulisten und Informationen über einen bestimmten Modellverpackungsauftrag abzurufen.

Auflisten von Modellpaketierungsaufträgen

Sie können die Modellverpackungsaufträge in einem Projekt auflisten, indem Sie die [ListModelPackagingJobs](#)API aufrufen. Die Antwort enthält eine Liste von [ModelPackagingJobMetadata](#)Objekten, die Informationen zu den einzelnen Paketierungsaufträgen

für Modelle enthält. Ebenfalls enthalten ist ein Paginierungstoken, mit dem Sie die nächsten Ergebnisse abrufen können, falls die Liste unvollständig ist.

Um Ihre Model-Packaging-Jobs aufzulisten

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden CLI-Befehl. Wechseln Sie `project_name` zum Namen des Projekts, das Sie verwenden möchten.

```
aws lookoutvision list-model-packaging-jobs \  
  --project-name project_name \  
  --profile lookoutvision-access
```

Beschreiben Sie einen Musterauftrag im Bereich Paketierung

Verwenden Sie die [DescribeModelPackagingJob](#)API, um Informationen über einen Modellverpackungsauftrag abzurufen. Die Antwort ist ein [ModelPackagingDescription](#)Objekt, das den aktuellen Status des Jobs und andere Informationen enthält.

Um ein Paket zu beschreiben

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden CLI-Befehl. Ändern Sie Folgendes:
 - `project_name` in den Namen des Projekts, das Sie verwenden.
 - `job_name` zum Namen des Jobs. Sie erhalten den Jobnamen (JobName), wenn Sie anrufen [StartModelPackagingJob](#).

```
aws lookoutvision describe-model-packaging-job \  
  --project-name project_name \  
  --job-name job_name \  
  --profile lookoutvision-access
```

Schreiben Sie Ihre Client-Anwendungskomponente

Eine Client-Anwendungskomponente ist eine benutzerdefinierte AWS IoT Greengrass Version 2 Komponente, die Sie schreiben. Es implementiert die Geschäftslogik, die Sie für die Verwendung eines Amazon Lookout for Vision Vision-Modells auf einem AWS IoT Greengrass Version 2 Kerngerät benötigen.

Um auf ein Modell zuzugreifen, verwendet Ihre Client-Anwendungskomponente die Komponente Lookout for Vision Edge Agent. Die Komponente Lookout for Vision Edge Agent stellt eine API bereit, mit der Sie Bilder mit einem Modell analysieren und die Modelle auf einem Kerngerät verwalten können.

Die Agent-API von Lookout for Vision Edge wird mithilfe von gRPC implementiert, einem Protokoll für Remoteprozeduraufrufe. Weitere Informationen finden Sie unter [gRPC](#). Um Ihren Code zu schreiben, können Sie jede Sprache verwenden, die von gRPC unterstützt wird. Wir stellen Python-Beispielcode zur Verfügung. Weitere Informationen finden Sie unter [Verwenden Sie ein Modell in Ihrer Client-Anwendungskomponente](#).

Note

Die Komponente Lookout for Vision Edge Agent ist von der Modellkomponente abhängig, die Sie bereitstellen. Sie wird automatisch auf dem Kerngerät bereitgestellt, wenn Sie die Modellkomponente auf dem Kerngerät bereitstellen.

Gehen Sie wie folgt vor, um eine Client-Anwendungskomponente zu schreiben.

1. [Richten Sie Ihre Umgebung](#) für die Verwendung von gRPC ein und installieren Sie Bibliotheken von Drittanbietern.
2. [Schreiben Sie Code, um das Modell zu verwenden](#).
3. [Stellen Sie den Code als benutzerdefinierte Komponente](#) auf dem Kerngerät bereit.

Ein Beispiel für eine Client-Anwendungskomponente, die zeigt, wie eine Anomalieerkennung in einer benutzerdefinierten GStreamer Pipeline durchgeführt wird, finden Sie unter <https://github.com/aws-labs/aws-greengrass-labs-lookoutvision-gstreamer>.

Einrichten Ihrer Umgebung

Um Client-Code zu schreiben, stellt Ihre Entwicklungsumgebung eine Remoteverbindung zu einem AWS IoT Greengrass Version 2 Kerngerät her, auf dem Sie eine Modellkomponente und Abhängigkeiten von Amazon Lookout for Vision bereitgestellt haben. Alternativ können Sie Code auf einem Core-Gerät schreiben. Weitere Informationen finden Sie unter [AWS IoT Greengrass-Entwicklungstools](#) und [Entwickeln von AWS IoT Greengrass-Komponenten](#).

Ihr Kundencode sollte den gRPC-Client verwenden, um auf den Amazon Lookout for Vision Edge Agent zuzugreifen. In diesem Abschnitt wird gezeigt, wie Sie Ihre Entwicklungsumgebung mit gRPC einrichten und Abhängigkeiten von Drittanbietern installieren, die für den DetectAnomalies Beispielpcode benötigt werden.

Nachdem Sie Ihren Client-Code geschrieben haben, erstellen Sie eine benutzerdefinierte Komponente und stellen die benutzerdefinierte Komponente auf Ihren Edge-Geräten bereit. Weitere Informationen finden Sie unter [Die Komponente der Client-Anwendung wird erstellt](#).

Themen

- [gRPC einrichten](#)
- [Hinzufügen von Abhängigkeiten von Drittanbietern](#)

gRPC einrichten

In Ihrer Entwicklungsumgebung benötigen Sie einen gRPC-Client, den Sie in Ihrem Code verwenden, um die Agenten-API von Lookout for Vision Edge aufzurufen. Dazu erstellen Sie einen gRPC-Stub mithilfe einer .proto Service-Definitionsdatei für den Lookout for Vision Edge Agent.

Note

Sie können die Dienstdefinitionsdatei auch aus dem Anwendungspaket Lookout for Vision Edge Agent abrufen. Das Anwendungspaket wird installiert, wenn die Komponente Lookout for Vision Edge Agent als Abhängigkeit von der Modellkomponente installiert wird. Das Anwendungspaket befindet sich unter `/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/edge_agent_version/lookoutvision_edge_agent`. *edge_agent_version* Ersetzen Sie es durch die Version des Lookout for Vision Edge Agent, die Sie verwenden. Um das Anwendungspaket zu erhalten, müssen Sie den Lookout for Vision Edge Agent auf einem Kerngerät bereitstellen.

So richten Sie gRPC ein

1. Laden Sie die ZIP-Datei [proto.zip](#) herunter. Die ZIP-Datei enthält die .proto Service-Definitionsdatei (`edge-agent.proto`).
2. Entpacken Sie den Inhalt.
3. Öffnen Sie eine Befehlszeile und navigieren Sie zu dem Ordner, der Folgendes enthält `edge-agent.proto`.
4. Verwenden Sie die folgenden Befehle, um die Python-Client-Schnittstellen zu generieren.

```
%%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
```

Wenn die Befehle erfolgreich sind, werden die Stubs `edge_agent_pb2_grpc.py` und `edge_agent_pb2.py` die Stubs im Arbeitsverzeichnis erstellt.

5. Schreiben Sie den Client-Code, der Ihr Modell verwendet. Weitere Informationen finden Sie unter [Verwenden Sie ein Modell in Ihrer Client-Anwendungskomponente](#).

Hinzufügen von Abhängigkeiten von Drittanbietern

Der `DetectAnomalies` Beispielcode verwendet die [Pillow-Bibliothek](#), um mit Bildern zu arbeiten. Weitere Informationen finden Sie unter [Erkennen von Anomalien mithilfe von Bildbytes](#).

Verwenden Sie den folgenden Befehl, um die Pillow-Bibliothek zu installieren.

```
python3 -m pip install Pillow
```

Verwenden Sie ein Modell in Ihrer Client-Anwendungskomponente

Die Schritte zur Verwendung eines Modells aus einer Client-Anwendungskomponente ähneln der Verwendung eines in der Cloud gehosteten Modells.

1. Starten Sie die Ausführung des Modells.
2. Erkennen Sie Anomalien in Bildern.
3. Stoppen Sie das Modell, falls es nicht mehr benötigt wird.

Der Amazon Lookout for Vision Edge Agent bietet eine API zum Starten eines Modells, zum Erkennen von Anomalien in einem Bild und zum Stoppen eines Modells. Sie können die API auch verwenden, um die Modelle auf einem Gerät aufzulisten und Informationen über ein bereitgestelltes Modell abzurufen. Weitere Informationen finden Sie unter [Referenz zur Amazon Lookout for Vision Edge Agent-API](#).

Sie können Fehlerinformationen abrufen, indem Sie die gRPC-Statuscodes überprüfen. Weitere Informationen finden Sie unter [Fehlerinformationen abrufen](#).

Um Ihren Code zu schreiben, können Sie jede Sprache verwenden, die von gRPC unterstützt wird. Wir stellen Python-Beispielcode zur Verfügung.

Themen

- [Verwenden Sie den Stub in Ihrer Client-Anwendungskomponente](#)
- [Das Modell starten](#)
- [Erkennung von Anomalien](#)
- [Das Modell wird gestoppt](#)
- [Modelle auf einem Gerät auflisten](#)
- [Beschreibung eines Modells](#)
- [Fehlerinformationen abrufen](#)

Verwenden Sie den Stub in Ihrer Client-Anwendungskomponente

Verwenden Sie den folgenden Code, um den Zugriff auf Ihr Modell über den Lookout for Vision Edge Agent einzurichten.

```
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
```

Das Modell starten

Sie starten ein Modell, indem Sie die [StartModel](#) API aufrufen. Es kann eine Weile dauern, bis das Modell gestartet wird. Sie können den aktuellen Status überprüfen, indem Sie aufrufen [DescribeModel](#). Das Modell wird ausgeführt, wenn der Wert des status Felds Wird ausgeführt lautet.

Beispiel-Code

component_name Ersetzen Sie es durch den Namen Ihrer Modellkomponente.

```
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)
```

Erkennung von Anomalien

Sie verwenden die [DetectAnomalies](#) API, um Anomalien in einem Bild zu erkennen.

Der DetectAnomalies Vorgang erwartet, dass die Bild-Bitmap im RGB888 gepackten Format übergeben wird. Das erste Byte steht für den roten Kanal, das zweite Byte für den grünen Kanal und das dritte Byte für den blauen Kanal. Wenn Sie das Bild in einem anderen Format bereitstellen, z. B. in BGR, DetectAnomalies sind die Prognosen von falsch.

Standardmäßig verwendet OpenCV das BGR-Format für Bild-Bitmaps. Wenn Sie OpenCV zum Aufnehmen von Bildern für die Analyse verwenden DetectAnomalies, müssen Sie das Bild in ein RGB888 Format konvertieren, bevor Sie das Bild an übergeben. DetectAnomalies

Die Bilder, die Sie zur Verfügung DetectAnomalies stellen, müssen dieselben Breiten- und Höhenmaße haben wie die Bilder, mit denen Sie das Modell trainiert haben.

Erkennen von Anomalien mithilfe von Bildbytes

Sie können Anomalien in einem Bild erkennen, indem Sie das Bild als Bildbytes angeben. Im folgenden Beispiel werden die Bildbytes aus einem Bild abgerufen, das im lokalen Dateisystem gespeichert ist.

sample.jpg Ersetzen Sie es durch den Namen der Bilddatei, die Sie analysieren möchten.
component_name Ersetzen Sie durch den Namen Ihrer Modellkomponente.

```
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")
```

Erkennung von Anomalien mithilfe eines gemeinsam genutzten Speichersegments

Sie können Anomalien in einem Bild erkennen, indem Sie das Bild als Bildbytes im gemeinsamen POSIX-Speichersegment bereitstellen. Für eine optimale Leistung empfehlen wir, Shared Memory für Anfragen zu verwenden. DetectAnomalies Weitere Informationen finden Sie unter [DetectAnomalies](#).

Das Modell wird gestoppt

Wenn Sie das Modell nicht mehr verwenden, stoppt die [StopModel](#) API die Ausführung des Modells.

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
        model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

Modelle auf einem Gerät auflisten

Sie können die [the section called “ListModels”](#) API verwenden, um die Modelle aufzulisten, die auf einem Gerät bereitgestellt werden.

```
models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

Beschreibung eines Modells

Sie können Informationen über ein Modell abrufen, das auf einem Gerät bereitgestellt wurde, indem Sie die [DescribeModel](#) API aufrufen. Die Verwendung `DescribeModel` ist nützlich, um den aktuellen Status eines Modells abzurufen. Sie müssen beispielsweise wissen, ob ein Modell läuft, bevor Sie es aufrufen können `DetectAnomalies`. Beispielcode finden Sie unter [Das Modell starten](#).

Fehlerinformationen abrufen

gRPC-Statuscodes werden verwendet, um API-Ergebnisse zu melden.

Sie können Fehlerinformationen abrufen, indem Sie die `RpcError` Ausnahme abfangen, wie im folgenden Beispiel gezeigt. Informationen zu den Fehlerstatuscodes finden Sie im [Referenzthema](#) für eine API.

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

Die Komponente der Client-Anwendung wird erstellt

Sie können die Client-Anwendungskomponente erstellen, sobald Sie Ihre gRPC-Stubs generiert haben und Ihren Client-Anwendungscode bereit haben. Die Komponente, die Sie erstellen, ist eine benutzerdefinierte Komponente, mit der Sie sie auf einem AWS IoT Greengrass Version 2 Kerngerät bereitstellen. AWS IoT Greengrass V2 Ein von Ihnen erstelltes Rezept beschreibt Ihre benutzerdefinierte Komponente. Das Rezept beinhaltet alle Abhängigkeiten, die ebenfalls

bereitgestellt werden müssen. In diesem Fall geben Sie die Modellkomponente an, in der Sie erstellen [Verpacken Ihres Amazon Lookout for Vision Vision-Modells](#). Weitere Informationen zu Komponentenrezepten finden Sie in der [Referenz zu AWS IoT Greengrass Version 2 Komponentenrezepten](#).

Die Verfahren zu diesem Thema zeigen, wie Sie die Client-Anwendungskomponente aus einer Rezeptdatei erstellen und als AWS IoT Greengrass V2 benutzerdefinierte Komponente veröffentlichen. Sie können die AWS IoT Greengrass V2 Konsole oder das AWS SDK verwenden, um die Komponente zu veröffentlichen.

Ausführliche Informationen zum Erstellen einer benutzerdefinierten Komponente finden Sie im Folgenden in der AWS IoT Greengrass V2 Dokumentation.

- [Entwickeln und testen Sie eine Komponente auf Ihrem Gerät](#)
- [AWS IoT Greengrass-Komponenten erstellen](#)
- [Veröffentlichen Sie Komponenten zur Bereitstellung auf Ihren Kerngeräten](#)

Themen

- [IAM-Berechtigungen für die Veröffentlichung einer Client-Anwendungskomponente](#)
- [Das Rezept erstellen](#)
- [Veröffentlichen der Client-Anwendungskomponente \(Konsole\)](#)
- [Veröffentlichen der Client-Anwendungskomponente \(SDK\)](#)

IAM-Berechtigungen für die Veröffentlichung einer Client-Anwendungskomponente

Um Ihre Client-Anwendungskomponente zu erstellen und zu veröffentlichen, benötigen Sie die folgenden IAM-Berechtigungen:

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

Das Rezept erstellen

In diesem Verfahren erstellen Sie das Rezept für eine einfache Client-Anwendungskomponente. Der darin enthaltene Code `lookoutvision_edge_agent_example.py` listet die Modelle

auf, die auf dem Gerät bereitgestellt werden, und wird automatisch ausgeführt, nachdem Sie die Komponente auf dem Kerngerät bereitgestellt haben. Um die Ausgabe anzuzeigen, überprüfen Sie das Komponentenprotokoll, nachdem Sie die Komponente bereitgestellt haben. Weitere Informationen finden Sie unter [Bereitstellen Ihrer Komponenten auf einem Gerät](#). Wenn Sie bereit sind, verwenden Sie dieses Verfahren, um das Rezept für den Code zu erstellen, der Ihre Geschäftslogik implementiert.

Sie erstellen das Rezept als Datei im JSON- oder YAML-Format. Sie laden auch den Client-Anwendungscode in einen Amazon S3 S3-Bucket hoch.

Um das Rezept für die Komponente der Client-Anwendung zu erstellen

1. Falls Sie dies noch nicht getan haben, erstellen Sie die gRPC-Stub-Dateien. Weitere Informationen finden Sie unter [gRPC einrichten](#).
2. Speichern Sie den folgenden Code in einer Datei mit dem Namen `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. [Erstellen Sie einen Amazon S3 S3-Bucket](#) (oder verwenden Sie einen vorhandenen Bucket), um die Quelldateien für Ihre Client-Anwendungskomponente zu speichern. Der Bucket muss sich in Ihrem AWS Konto und in derselben AWS Region befinden, in der Sie Amazon Lookout for Vision verwenden AWS IoT Greengrass Version 2 .
4. Laden Sie `lookoutvision_edge_agent_example.py` `edge_agent_pb2_grpc.py` and `edge_agent_pb2.py` es in den Amazon S3 S3-Bucket hoch, den Sie im vorherigen

Schritt erstellt haben. Notieren Sie sich den Amazon S3 S3-Pfad jeder Datei. Sie haben erstellt `edge_agent_pb2_grpc.py` und sind `edge_agent_pb2.py` dabei [gRPC einrichten](#).

5. Erstellen Sie in einem Editor die folgende JSON- oder YAML-Rezeptdatei.

- `model_component` zum Namen Ihrer Modellkomponente. Weitere Informationen finden Sie unter [Einstellungen der Komponenten](#).
- Ändern Sie die URI-Einträge in die S3-Pfade von `lookoutvision_edge_agent_example.py` `edge_agent_pb2_grpc.py`, `und` `edge_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.lookoutvision.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. Speichern Sie die JSON- oder YAML-Datei auf Ihrem Computer.

7. Erstellen Sie die Client-Anwendungskomponente, indem Sie einen der folgenden Schritte ausführen:
 - Wenn Sie die AWS IoT Greengrass Konsole verwenden möchten, tun Sie dies [Veröffentlichen der Client-Anwendungskomponente \(Konsole\)](#).
 - Wenn Sie das AWS SDK verwenden möchten, tun Sie es [Veröffentlichen der Client-Anwendungskomponente \(SDK\)](#).

Veröffentlichen der Client-Anwendungskomponente (Konsole)

Sie können die AWS IoT Greengrass V2 Konsole verwenden, um die Client-Anwendungskomponente zu veröffentlichen.

Um die Client-Anwendungskomponente zu veröffentlichen

1. Falls Sie es noch nicht getan haben, erstellen Sie das Rezept für Ihre Client-Anwendungskomponente, indem Sie [Das Rezept erstellen](#)
2. Öffnen Sie die AWS IoT Greengrass Konsole unter <https://console.aws.amazon.com/iot/>
3. Wählen Sie im linken Navigationsbereich unter Greengrass die Option Components aus.
4. Wählen Sie unter Meine Komponenten die Option Komponente erstellen aus.
5. Wählen Sie auf der Seite Komponente erstellen die Option Rezept als JSON eingeben aus, wenn Sie ein Rezept im JSON-Format verwenden möchten. Wählen Sie „Rezept als YAML eingeben“, wenn Sie ein Rezept im YAML-Format verwenden möchten.
6. Ersetzen Sie unter Rezept das vorhandene Rezept durch das JSON- oder YAML-Rezept, in dem Sie es erstellt haben. [Das Rezept erstellen](#)
7. Wählen Sie Komponente erstellen aus.
8. [Stellen Sie](#) als Nächstes Ihre Client-Anwendungskomponente bereit.

Veröffentlichen der Client-Anwendungskomponente (SDK)

Sie können die Client-Anwendungskomponente mithilfe der [CreateComponentVersion](#) API veröffentlichen.

Um die Client-Anwendungskomponente (SDK) zu veröffentlichen

1. Falls Sie es noch nicht getan haben, erstellen Sie das Rezept für Ihre Client-Anwendungskomponente, indem Sie [Das Rezept erstellen](#)
2. Geben Sie in der Befehlszeile den folgenden Befehl ein, um die Client-Anwendungskomponente zu erstellen. `recipe-file` Ersetzen Sie ihn durch den Namen der Rezeptdatei, in der Sie sie erstellt haben [Das Rezept erstellen](#).

```
aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
```

Notieren Sie sich den ARN der Komponente in der Antwort. Sie benötigen ihn im nächsten Schritt.

3. Verwenden Sie den folgenden Befehl, um den Status der Client-Anwendungskomponente abzurufen. `component-arn` Ersetzen Sie es durch den ARN, den Sie im vorherigen Schritt notiert haben. Die Client-Anwendungskomponente ist bereit, wenn der Wert von `componentState` ist `DEPLOYABLE`.

```
aws greengrassv2 describe-component --arn component-arn
```

4. [Stellen Sie](#) als Nächstes Ihre Client-Anwendungskomponente bereit.

Bereitstellen Ihrer Komponenten auf einem Gerät

Um die Modellkomponente und die Client-Anwendungskomponente auf einem AWS IoT Greengrass Version 2 Kerngerät bereitzustellen, verwenden Sie die AWS IoT Greengrass V2 Konsole oder die [CreateDeployment](#) API. Weitere Informationen finden Sie unter [Bereitstellungen erstellen](#) oder im AWS IoT Greengrass Version 2 Entwicklerhandbuch. Informationen zum Aktualisieren einer Komponente, die auf einem Kerngerät bereitgestellt wird, finden Sie unter [Bereitstellungen überarbeiten](#).

Themen

- [IAM-Berechtigungen für die Bereitstellung von Komponenten](#)
- [Bereitstellen Ihrer Komponenten \(Konsole\)](#)
- [Bereitstellung der Komponenten \(SDK\)](#)

IAM-Berechtigungen für die Bereitstellung von Komponenten

Um eine Komponente mit bereitzustellen, benötigen AWS IoT Greengrass V2 Sie die folgenden Berechtigungen:

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment` und `GetDeployment` haben abhängige Aktionen. Weitere Informationen finden Sie unter [Von AWS IoT Greengrass V2 definierte Aktionen](#).

Informationen zum Ändern von IAM-Berechtigungen finden Sie unter [Ändern der Berechtigungen für einen Benutzer](#).

Bereitstellen Ihrer Komponenten (Konsole)

Gehen Sie wie folgt vor, um die Client-Anwendungskomponente auf einem Kerngerät bereitzustellen. Die Client-Anwendung hängt von der Modellkomponente ab (die wiederum vom Lookout for Vision Edge Agent abhängt). Durch die Bereitstellung der Client-Anwendungskomponente wird auch die Bereitstellung der Modellkomponente und des Lookout for Vision Edge Agent gestartet.

Note

Sie können Ihre Komponenten zu einer vorhandenen Bereitstellung hinzufügen. Sie können Komponenten auch in einer Dinggruppe bereitstellen.

Um dieses Verfahren ausführen zu können, müssen Sie über ein konfiguriertes AWS IoT Greengrass V2 Kerngerät verfügen. Weitere Informationen finden Sie unter [Richten Sie Ihr AWS IoT Greengrass Version 2 Core-Gerät ein](#).

Um Ihre Komponenten auf einem Gerät bereitzustellen

1. Öffnen Sie die AWS IoT Greengrass Konsole unter <https://console.aws.amazon.com/iot/>.

2. Wählen Sie im linken Navigationsbereich unter Greengrass Deployments aus.
3. Wählen Sie unter Deployments die Option Create aus.
4. Gehen Sie auf der Seite Ziel angeben wie folgt vor:
 1. Geben Sie unter Bereitstellungsinformationen den Anzeigenamen für Ihre Bereitstellung ein oder ändern Sie ihn.
 2. Wählen Sie unter Bereitstellungsziel die Option Kerngerät aus und geben Sie einen Zielnamen ein.
 3. Wählen Sie Weiter.
5. Gehen Sie auf der Seite „Komponenten auswählen“ wie folgt vor:
 1. Wählen Sie unter Meine Komponenten den Namen Ihrer Client-Anwendungskomponente (`com.lookoutvision.EdgeAgentPythonExample`) aus.
 2. Wählen Sie Weiter
6. Behalten Sie auf der Seite Komponenten konfigurieren die aktuelle Konfiguration bei und klicken Sie auf Weiter.
7. Behalten Sie auf der Seite Erweiterte Einstellungen konfigurieren die aktuellen Einstellungen bei und wählen Sie Weiter.
8. Wählen Sie auf der Seite „Überprüfen“ die Option Bereitstellen aus, um mit der Bereitstellung Ihrer Komponente zu beginnen.

Überprüfen Sie den Bereitstellungsstatus (Konsole)

Sie können den Status Ihrer Bereitstellung von der AWS IoT Greengrass V2 Konsole aus überprüfen. Wenn Ihre Client-Anwendungskomponente das Beispielrezept und den Code von verwendet [the section called “Die Komponente der Client-Anwendung wird erstellt”](#), sehen Sie [sich nach Abschluss der Bereitstellung das Protokoll](#) der Client-Anwendungskomponente an. Bei Erfolg enthält das Protokoll eine Liste der Lookout for Vision Vision-Modelle, die für die Komponente bereitgestellt wurden.

Informationen zur Verwendung des AWS SDK zur Überprüfung des Bereitstellungsstatus finden Sie unter [Überprüfen des Bereitstellungsstatus](#).

So überprüfen Sie den Bereitstellungsstatus

1. Öffnen Sie die AWS IoT Greengrass Konsole unter <https://console.aws.amazon.com/iot/>

2. Wählen Sie im linken Navigationsbereich die Option Core-Geräte aus.
3. Wählen Sie unter Greengrass Core Devices Ihr Core-Gerät aus.
4. Wählen Sie den Tab Bereitstellungen, um den aktuellen Bereitstellungsstatus einzusehen.
5. Nachdem die Bereitstellungen erfolgreich waren (der Status lautet Abgeschlossen), öffnen Sie ein Terminalfenster auf dem Kerngerät und sehen Sie sich das Protokoll der Client-Anwendungskomponente unter an. `/greengrass/v2/logs/com.lookoutvision.EdgeAgentPythonExample.log` Wenn Ihre Bereitstellung das Beispielrezept und den Beispielcode verwendet, enthält das Protokoll die Ausgabe von `lookoutvision_edge_agent_example.py` Zum Beispiel:

```
Model Details model_component:"ModelComponent"
```

Bereitstellung der Komponenten (SDK)

Gehen Sie wie folgt vor, um die Client-Anwendungskomponente, die Modellkomponente und den Amazon Lookout for Vision Edge Agent auf Ihrem Kerngerät bereitzustellen.

1. Erstellen Sie eine `deployment.json` Datei, um die Bereitstellungsconfiguration für Ihre Komponenten zu definieren. Diese Datei sollte wie im folgenden Beispiel aussehen.

```
{
  "targetArn": "targetArn",
  "components": {
    "com.lookoutvision.EdgeAgentPythonExample": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
      }
    }
  }
}
```

- Ersetzen Sie im `targetArn` Feld *targetArn* durch den Amazon-Ressourcennamen (ARN) des Objekts oder der Objektgruppe, auf die die Bereitstellung ausgerichtet werden soll, und zwar im folgenden Format:
 - Objekt: `arn:aws:iot:region:account-id:thing/thingName`
 - Objektgruppe: `arn:aws:iot:region:account-id:thinggroup/thingGroupName`

2. Prüfen Sie, ob das Bereitstellungsziel über eine bestehende Bereitstellung verfügt, die Sie überarbeiten möchten. Gehen Sie wie folgt vor:
 - a. Führen Sie den folgenden Befehl aus, um die Bereitstellungen für das Bereitstellungsziel aufzulisten. `targetArn` Ersetzen Sie es durch den Amazon-Ressourcennamen (ARN) der AWS IoT-Zielsache oder -Dinggruppe. Verwenden Sie den ARNs Befehl, um die Dinge in der aktuellen AWS-Region abzurufen `aws iot list-things`.

```
aws greengrassv2 list-deployments --target-arn targetArn
```

Die Antwort enthält eine Liste mit der neuesten Bereitstellung für das Ziel. Wenn die Antwort leer ist, ist für das Ziel noch keine Bereitstellung vorhanden, und Sie können mit Schritt 3 fortfahren. Andernfalls kopieren Sie die `deploymentId` aus der Antwort, um sie im nächsten Schritt zu verwenden.

- b. Führen Sie den folgenden Befehl aus, um die Details der Bereitstellung abzurufen. Zu diesen Details gehören Metadaten, Komponenten und die Auftragskonfiguration. `deploymentId` Ersetzen Sie es durch die ID aus dem vorherigen Schritt.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

- c. Kopieren Sie eines der folgenden Schlüssel-Wert-Paare aus der Antwort des vorherigen Befehls in die Datei `deployment.json`. Sie können diese Werte für die neue Bereitstellung ändern.

- `deploymentName`— Der Name des Deployments.
- `components`— Die Komponenten des Deployments. Um eine Komponente zu deinstallieren, entfernen Sie sie aus diesem Objekt.
- `deploymentPolicies`— Die Richtlinien der Bereitstellung.
- `tags`— Die Tags der Bereitstellung.

3. Führen Sie den folgenden Befehl aus, um die Komponenten auf dem Gerät bereitzustellen. Notieren Sie sich den Wert von `deploymentId` in der Antwort.

```
aws greengrassv2 create-deployment \  
  --cli-input-json file://path/to/deployment.json
```

4. Führen Sie den folgenden Befehl aus, um den Status der Bereitstellung abzurufen. Ändern Sie `deployment-id` den Wert, den Sie im vorherigen Schritt notiert haben. Die Bereitstellung wurde erfolgreich abgeschlossen, wenn der Wert von `deploymentStatus` ist. `COMPLETED`

```
aws greengrassv2 get-deployment --deployment-id deployment-id
```

5. Öffnen Sie nach erfolgreicher Bereitstellung ein Terminalfenster auf dem Kerngerät und sehen Sie sich das Protokoll der Client-Anwendungskomponente unter an. `/greengrass/v2/logs/com.lookoutvision.EdgeAgentPythonExample.log` Wenn Ihre Bereitstellung das Beispielrezept und den Beispielcode verwendet, enthält das Protokoll die Ausgabe von `lookoutvision_edge_agent_example.py` Zum Beispiel:

```
Model Details model_component:"ModelComponent"
```

Referenz zur Amazon Lookout for Vision Edge Agent-API

Dieser Abschnitt ist die API-Referenz für den Amazon Lookout for Vision Edge Agent.

Erkennung von Anomalien mit einem Modell

Sie verwenden die [DetectAnomalies](#) API, um Anomalien in Bildern zu erkennen, indem Sie ein laufendes Modell auf einem Kerngerät verwenden. AWS IoT Greengrass Version 2

Modellinformationen abrufen

APIs die Informationen über Modelle abrufen, die auf einem Kerngerät bereitgestellt werden.

- [ListModels](#)
- [DescribeModel](#)

Ein Modell ausführen

APIs zum Starten und Stoppen eines Amazon Lookout for Vision Vision-Modells, das auf einem Kerngerät bereitgestellt wird.

- [StartModel](#)
- [StopModel](#)

DetectAnomalies

Erkennt Anomalien im bereitgestellten Bild.

Die Antwort von `DetectAnomalies` beinhaltet eine boolesche Vorhersage, dass das Bild eine oder mehrere Anomalien enthält, sowie einen Konfidenzwert für die Vorhersage. Handelt es sich bei dem Modell um ein Segmentierungsmodell, umfasst die Antwort Folgendes:

- Ein Maskenbild, das jeden Anomalie-Typ in einer eindeutigen Farbe abdeckt. Sie können das Maskenbild im gemeinsamen Speicher `DetectAnomalies` speichern oder die Maske als Bildbyte zurückgeben.
- Der prozentuale Bereich des Bildes, den ein Anomalie-Typ abdeckt.
- Die Hex-Farbe für einen Anomalie-Typ auf dem Maskenbild.

Note

Das Modell, das Sie mit verwenden, `DetectAnomalies` muss laufen. Sie können den aktuellen Status abrufen, indem Sie anrufen [DescribeModel](#). Informationen zum Starten der Ausführung eines Modells finden Sie unter [StartModel](#).

`DetectAnomalies` unterstützt gepackte Bitmaps (Bilder) im Interleaved-Format RGB888. Das erste Byte steht für den roten Kanal, das zweite Byte für den grünen Kanal und das dritte Byte für den blauen Kanal. Wenn Sie das Bild in einem anderen Format bereitstellen, z. B. in BGR, `DetectAnomalies` sind die Prognosen von falsch.

Standardmäßig verwendet OpenCV das BGR-Format für Bild-Bitmaps. Wenn Sie OpenCV zum Aufnehmen von Bildern für die Analyse verwenden `DetectAnomalies`, müssen Sie das Bild in ein RGB888 Format konvertieren, bevor Sie das Bild an übergeben. `DetectAnomalies`

Die unterstützte Mindestabmessung für Bilder ist 64x64 Pixel. Die maximal unterstützte Bildgröße beträgt 4096x4096 Pixel.

Sie können das Bild in der Protobuf-Nachricht oder über ein gemeinsam genutztes Speichersegment senden. Das Serialisieren großer Bilder in die Protobuf-Nachricht kann die Latenz von Aufrufen von erheblich erhöhen. `DetectAnomalies` Für die geringste Latenz empfehlen wir die Verwendung von Shared Memory.

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

DetectAnomaliesRequest

Die Eingabeparameter für `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

Das Bild, mit `DetectAnomalies` dem Sie analysieren möchten.

`width`

Die Breite des Bildes in Pixeln.

height

Die Höhe des Bildes in Pixeln.

byte_data

In der Protobuf-Nachricht übergebene Bildbytes.

shared_memory_handle

Bildbytes, die im gemeinsamen Speichersegment übergeben wurden.

SharedMemoryHandle

Stellt ein gemeinsam genutztes POSIX-Speichersegment dar.

Name

Der Name des POSIX-Speichersegments. Hinweise zum Erstellen von Shared Memory finden Sie unter [shm_open](#).

size

Die Größe des Bildpuffers in Byte, beginnend mit dem Offset.

offset

Der Offset in Byte bis zum Anfang des Bildpuffers vom Anfang des gemeinsamen Speichersegments.

AnomalyMaskParams

Parameter für die Ausgabe einer Anomaliemaske. (Segmentierungsmodell).

shared_memory_handle

Enthält die Bildbytes für die Maske, falls sie nicht angegeben wurde. shared_memory_handle

DetectAnomaliesRequest

model_component

Der Name der AWS IoT Greengrass V2 Komponente, die das Modell enthält, das Sie verwenden möchten.

Bitmap

Das Bild, mit DetectAnomalies dem Sie analysieren möchten.

anomaly_mask_params

Optionale Parameter für die Ausgabe der Maske. (Segmentierungsmodell).

DetectAnomaliesResponse

Die Antwort von 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

Stellt eine auf einem Bild gefundene Anomalie dar. (Segmentierungsmodell).

Name

Der Name eines Anomalie-Typs, der in einem Bild gefunden wurde. `name` entspricht einem Anomalie-Typ im Trainingsdatensatz. Der Dienst fügt den Typ der Hintergrundanomalie automatisch in das Antwortformular ein. `DetectAnomalies`

`pixel_anomaly`

Informationen über die Pixelmaske, die einen Anomalie-Typ abdeckt.

`PixelAnomaly`

Informationen über die Pixelmaske, die einen Anomalie-Typ abdeckt. (Segmentierungsmodell).

`total_percentage_area`

Der prozentuale Bereich des Bildes, den der Anomalie-Typ abdeckt.

`hex_color`

Ein Hex-Farbwert, der den Typ der Anomalie auf dem Bild darstellt. Die Farbe entspricht der Farbe des Anomalie-Typs, der im Trainingsdatensatz verwendet wird.

`DetectAnomalyResult`

`is_anomalous`

Zeigt an, ob das Bild eine Anomalie enthält. `true` ob das Bild eine Anomalie enthält. `false` wenn das Bild normal ist.

Konfidenz

Das Vertrauen, `DetectAnomalies` das Sie in die Genauigkeit der Vorhersage haben. `confidence` ist ein Fließkommawert zwischen 0 und 1.

`anomaly_mask`

wenn `shared_memory_handle` nicht angegeben wurde, enthält es die Bildbytes für die Maske. (Segmentierungsmodell).

Unregelmäßigkeiten

Eine Liste von 0 oder mehr Anomalien, die im Eingabebild gefunden wurden. (Segmentierungsmodell).

`anomaly_score`

Eine Zahl, die quantifiziert, wie stark die für ein Bild vorhergesagten Anomalien von einem Bild ohne Anomalien abweichen. `anomaly_score` ist ein Gleitkommawert, der von 0.0 bis (niedrigste Abweichung von einem normalen Bild) bis 1,0 (höchste Abweichung von einem normalen Bild) reicht. Amazon Lookout for Vision gibt einen Wert für `anomaly_score` zurück, auch wenn die Vorhersage für ein Bild normal ist.

`anomaly_threshold`

Eine Zahl (Float), die bestimmt, wann die vorhergesagte Klassifizierung für ein Bild normal oder ungewöhnlich ist. Bilder mit `anomaly_score`, deren Wert gleich oder höher als der Wert von `anomaly_threshold` ist, gelten als anomal. Ein `anomaly_score` Wert, der darunter liegt, weist auf ein normales Bild hin. Der Wert `anomaly_threshold`, den ein Modell verwendet, wird von Amazon Lookout for Vision berechnet, wenn Sie das Modell trainieren. Sie können den Wert von nicht festlegen oder ändern `anomaly_threshold`

Statuscodes

Code	Anzahl	Beschreibung
OK	0	<code>DetectAnomalies</code> hat erfolgreich eine Vorhersage gemacht
UNKNOWN	2	Ein unbekannter Fehler ist aufgetreten.
UNGÜLTIGER_ARGUMENT	3	Ein oder mehrere Eingabeparameter sind ungültig. Weitere Informationen finden Sie in der Fehlermeldung.
NOT_GEFUNDEN	5	Ein Modell mit dem angegebenen Namen wurde nicht gefunden.
RESOURCE_EXHAUSTED	8	Es sind nicht genügend Ressourcen vorhanden, um diesen Vorgang durchzuführen. Zum Beispiel kann

Code	Anzahl	Beschreibung
		The Lookout for Vision Edge Agent nicht mit der Rate der Anrufe an DetectAnomalies Schritt halten. Weitere Informationen finden Sie in der Fehlermeldung.
FAILED_PRECONDITION	9	DetectAnomalies wurde für ein Modell aufgerufen, das sich nicht im Status RUNNING befindet.
INTERN	13	Ein interner Fehler ist aufgetreten.

DescribeModel

Beschreibt ein Amazon Lookout for Vision Vision-Modell, das auf einem AWS IoT Greengrass Version 2 Kerngerät bereitgestellt wird.

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

DescribeModelRequest

```
message DescribeModelRequest {  
    string model_component = 1;  
}
```

model_component

Der Name der AWS IoT Greengrass V2 Komponente, die das Modell enthält, das Sie beschreiben möchten.

DescribeModelResponse

```
message ModelDescription {
```

```
string model_component = 1;
string lookout_vision_model_arn = 2;
ModelState status = 3;
string status_message = 4;
}
```

```
message DescribeModelResponse {
  ModelDescription model_description = 1;
}
```

ModelDescription

model_component

Der Name der AWS IoT Greengrass Version 2 Komponente, die das Amazon Lookout for Vision Vision-Modell enthält.

lookout_vision_model_arn

Der Amazon-Ressourcenname ARN des Amazon Lookout for Vision Vision-Modells, das zur Generierung der AWS IoT Greengrass V2 Komponente verwendet wurde.

Status

Der aktuelle Status des Modells. Weitere Informationen finden Sie unter [ModelStatus](#).

status_message

Die Statusmeldung für das Modell.

Statuscodes

Code	Anzahl	Beschreibung
OK	0	Der Anruf war erfolgreich.
UNKNOWN	2	Ein unbekannter Fehler ist aufgetreten.

Code	Anzahl	Beschreibung
UNGÜLTIGER_ARGUMENT	3	Ein oder mehrere Eingabeparameter sind ungültig. Weitere Informationen finden Sie in der Fehlermeldung.
NOT_GEFUNDEN	5	Ein Modell mit dem angegebenen Namen wurde nicht gefunden.
INTERN	13	Ein interner Fehler ist aufgetreten.

ListModels

Listet die auf einem AWS IoT Greengrass Version 2 Kerngerät bereitgestellten Modelle auf.

```
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

model_component

Der Name der AWS IoT Greengrass Version 2 Komponente, die ein Amazon Lookout for Vision Vision-Modell enthält.

lookout_vision_model_arn

Der Amazon-Ressourcenname (ARN) des Amazon Lookout for Vision Vision-Modells, das zur Generierung der AWS IoT Greengrass V2 Komponente verwendet wurde.

Status

Der aktuelle Status des Modells. Weitere Informationen finden Sie unter [ModelStatus](#).

status_message

Die Statusmeldung für das Modell.

Statuscodes

Code	Anzahl	Beschreibung
OK	0	Der Anruf war erfolgreich.
UNKNOWN	2	Ein unbekannter Fehler ist aufgetreten.
INTERN	13	Ein interner Fehler ist aufgetreten.

StartModel

Startet ein Modell, das auf einem AWS IoT Greengrass Version 2 Kerngerät ausgeführt wird. Es kann eine Weile dauern, bis das Modell läuft. Rufen Sie an, um den aktuellen Status zu überprüfen [DescribeModel](#). Das Modell läuft, wenn das Status Feld läuft **RUNNING**.

Die Anzahl der Modelle, die Sie gleichzeitig ausführen können, hängt von der Hardwarespezifikation Ihres Kerngeräts ab.

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

StartModelRequest

```
message StartModelRequest {  
    string model_component = 1;  
}
```

model_component

Der Name der AWS IoT Greengrass Version 2 Komponente, die das Modell enthält, das Sie starten möchten.

StartModelResponse

```
message StartModelResponse {  
    ModelStatus status = 1;  
}
```

Status

Der aktuelle Status des Modells. Die Antwort lautet `STARTING`, ob der Anruf erfolgreich ist. Weitere Informationen finden Sie unter [ModelStatus](#).

Statuscodes

Code	Anzahl	Beschreibung
OK	0	Das Modell wird gestartet
UNKNOWN	2	Ein unbekannter Fehler ist aufgetreten.
UNGÜLTIGER_ARGUMENT	3	Ein oder mehrere Eingabeparameter sind ungültig. Weitere

Code	Anzahl	Beschreibung
		Informationen finden Sie in der Fehlermeldung.
NOT_GEFUNDEN	5	Ein Modell mit dem angegebenen Namen wurde nicht gefunden.
RESOURCE_EXHAUSTED	8	Es stehen nicht genügend Ressourcen zur Verfügung, um diesen Vorgang durchzuführen. Beispielsweise ist nicht genügend Arbeitsspeicher vorhanden, um das Modell zu laden. Weitere Informationen finden Sie in der Fehlermeldung.
FAILED_PRECONDITION	9	Die Methode wurde für ein Modell aufgerufen, das sich nicht im Status STOPPED oder FAILED befindet.
INTERN	13	Ein interner Fehler ist aufgetreten.

StopModel

Stoppt ein Modell, das auf einem AWS IoT Greengrass Version 2 Kerngerät ausgeführt wird. StopModel kehrt zurück, nachdem das Modell gestoppt wurde. Das Modell wurde erfolgreich gestoppt, wenn das Status Feld in der STOPPED Antwort

```
rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {  
    string model_component = 1;  
}
```

model_component

Der Name der AWS IoT Greengrass Version 2 Komponente, die das Modell enthält, das Sie beenden möchten.

StopModelResponse

```
message StopModelResponse {  
    ModelStatus status = 1;  
}
```

Status

Der aktuelle Status des Modells. Die Antwort lautet STOPPED, ob der Anruf erfolgreich ist. Weitere Informationen finden Sie unter [ModelStatus](#).

Statuscodes

Code	Anzahl	Beschreibung
OK	0	Das Modell wird gestoppt.
UNKNOWN	2	Ein unbekannter Fehler ist aufgetreten.
UNGÜLTIGER_ARGUMENT	3	Ein oder mehrere Eingabeparameter sind ungültig. Weitere Informationen finden Sie in der Fehlermeldung.
NOT_GEFUNDEN	5	Ein Modell mit dem angegebenen Namen wurde nicht gefunden.
FEHLGESCHLAGENE_VORBEDINGUNG	9	Die Methode wurde für ein Modell aufgerufen, das sich

Code	Anzahl	Beschreibung
		nicht im Status RUNNING befindet.
INTERN	13	Ein interner Fehler ist aufgetreten.

ModelState

Der Status eines Modells, das auf einem AWS IoT Greengrass Version 2 Kerngerät bereitgestellt wurde. Rufen Sie an, um den aktuellen Status zu erfahren [DescribeModel](#).

```
enum ModelState {  
    STOPPED = 0;  
    STARTING = 1;  
    RUNNING = 2;  
    FAILED = 3;  
    STOPPING = 4;  
}
```

Verwenden des Amazon Lookout for Vision Vision-Dashboards

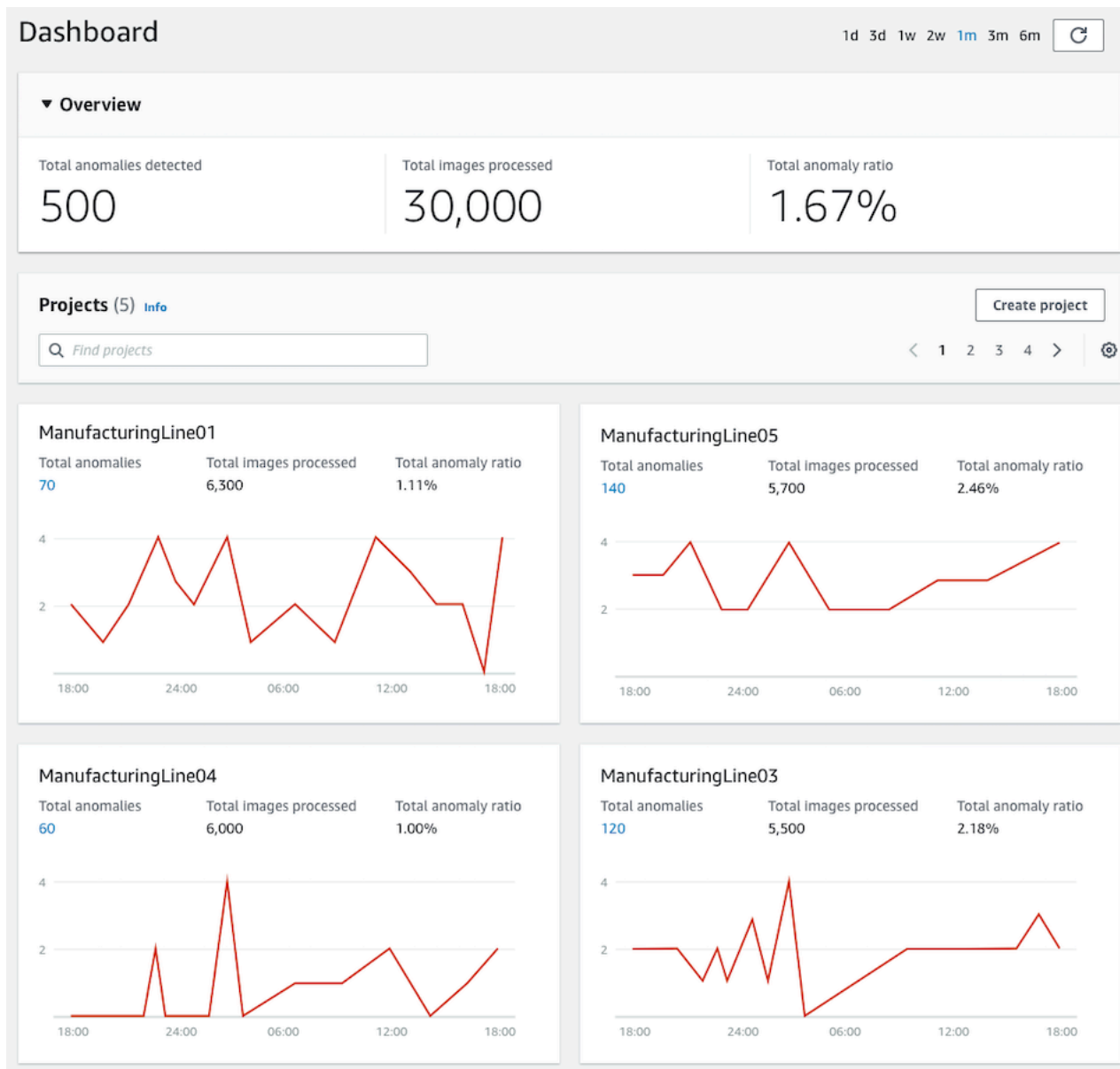
Das Dashboard bietet einen Überblick über die Kennzahlen für Ihre Amazon Lookout for Vision Vision-Projekte, z. B. die Gesamtzahl der in der letzten Woche festgestellten Anomalien. Mit dem Dashboard erhalten Sie einen Überblick über alle Ihre Projekte und einen Überblick über jedes einzelne Projekt. Sie können den Zeitplan wählen, über den die Metriken angezeigt werden. Sie können das Dashboard auch verwenden, um ein neues Projekt zu erstellen.

Im Abschnitt Übersicht werden die Gesamtzahl der Projekte, die Gesamtzahl der Bilder und die Gesamtzahl der Bilder angezeigt, die von all Ihren Projekten erkannt wurden.

Der Abschnitt Projekte enthält die folgenden Übersichtsinformationen für einzelne Projekte:

- Die Gesamtzahl der festgestellten Anomalien.
- Die Gesamtzahl der verarbeiteten Bilder.
- Das Verhältnis von Gesamtanomalien (d. h. der Prozentsatz der Bilder, bei denen eine Anomalie erkannt wurde).
- Ein Diagramm zeigt die Anomalieerkenntnisse im ausgewählten Zeitraum.

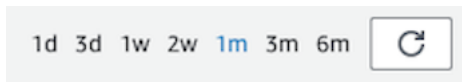
Sie können auch weitere Informationen zu einem Projekt erhalten.



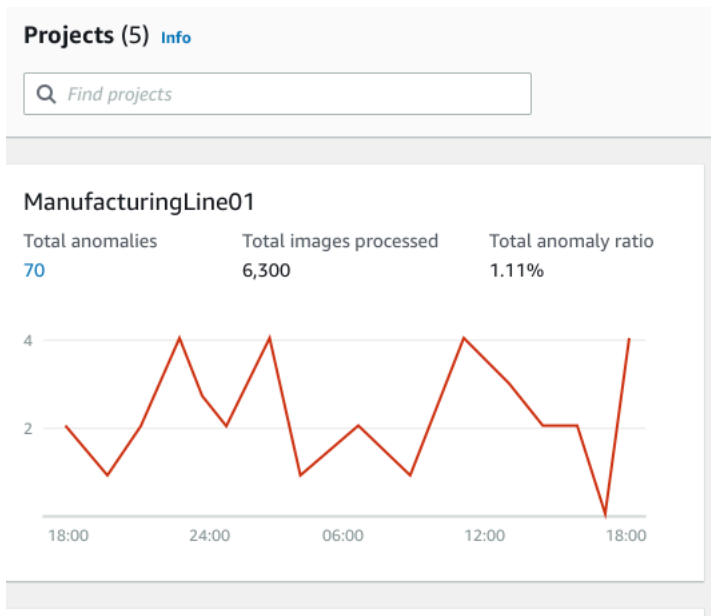
Um dein Dashboard zu verwenden

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich Dashboard aus.
4. Gehen Sie wie folgt vor, um Metriken für einen bestimmten Zeitraum anzuzeigen:
 - a. Wählen Sie den Zeitrahmen oben rechts im Dashboard aus.

- b. Wählen Sie die Schaltfläche „Aktualisieren“, um das Dashboard mit der neuen Zeitleiste anzuzeigen.



5. Um weitere Informationen zu einem Projekt zu erhalten, wählen Sie den Projektnamen im Abschnitt Projekte aus (z. B. ManufacturingLine01).



6. Um ein Projekt zu erstellen, wählen Sie im Abschnitt Projekte die Option Projekt erstellen aus.

Verwaltung Ihrer Amazon Lookout for Vision Vision-Ressourcen

Sie können Ihre Amazon Lookout for Vision Vision-Ressourcen mithilfe der Konsole oder des AWS SDK verwalten. Amazon Lookout for Vision bietet die folgenden Ressourcen:

- Projekte
- Datensätze
- Modelle
- Erkennungen in Studien

Note

Sie können eine Aufgabe zur Erkennung von Versuchen nicht löschen. Außerdem können Sie Erkennungen von Testversionen nicht mithilfe des AWS SDK verwalten.

Themen

- [Ihre Projekte anzeigen](#)
- [Löschen eines Projekts](#)
- [Ihre Datensätze anzeigen](#)
- [Hinzufügen von Bildern zu Ihrem Datensatz](#)
- [Bilder aus Ihrem Datensatz entfernen](#)
- [Löschen eines Datensatzes](#)
- [Exportieren von Datensätzen aus einem Projekt \(SDK\)](#)
- [Deine Modelle ansehen](#)
- [Löschen eines Modells](#)
- [Modelle taggen](#)
- [Ihre Aufgaben zur Erkennung von Studien anzeigen](#)

Ihre Projekte anzeigen

Sie können eine Liste der Amazon Lookout for Vision Vision-Projekte und Informationen zu einzelnen Projekten über die Konsole oder mithilfe des AWS SDK abrufen.

Note

Die Liste der Projekte ist letztendlich konsistent. Wenn Sie ein Projekt erstellen oder löschen, müssen Sie möglicherweise eine Weile warten, bis die Projektliste aktualisiert ist.

Ihre Projekte anzeigen (Konsole)

Führen Sie die Schritte im folgenden Verfahren aus, um Ihre Projekte in der Konsole anzuzeigen.

Um Ihre Projekte anzusehen

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus. Die Projektansicht wird angezeigt.
4. Wählen Sie einen Projektnamen, um die Projektdetails zu sehen.

Ihre Projekte anzeigen (SDK)

Ein Projekt verwaltet die Datensätze und Modelle für einen einzigen Anwendungsfall. Zum Beispiel die Erkennung von Anomalien in Maschinenteilen. Das folgende Beispiel ruft `ListProjects` auf, um eine Liste Ihrer Projekte abzurufen.

Um Ihre Projekte anzusehen (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um Ihre Projekte anzusehen.

CLI

Verwenden Sie den `list-projects`-Befehl, um die Projekte in Ihrem Konto aufzulisten.

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

Verwenden Sie den `describe-project` Befehl, um Informationen zu einem Projekt abzurufen.

Ändern Sie den Wert von `project-name` in den Namen des Projekts, das Sie beschreiben möchten.

```
aws lookoutvision describe-project --project-name project_name \  
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```

/**
 * Lists the Amazon Lookout for Vision projects in the current AWS account and
 * AWS
 * Region.
 *
 * @param lfvcClient 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;
}
```

Löschen eines Projekts

Sie können ein Projekt von der Projektansichtsseite in der Konsole oder mithilfe des `DeleteProject` Vorgangs löschen.

Die Bilder, auf die in den Datensätzen eines Projekts verwiesen wird, werden nicht gelöscht.

Löschen eines Projekts (Konsole)

Gehen Sie wie folgt vor, um ein Projekt zu löschen. Wenn Sie das Konsolenverfahren verwenden, werden die zugehörigen Modellversionen und Datasets für Sie gelöscht.

So löschen Sie ein Projekt

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie löschen möchten.
5. Wählen Sie oben auf der Seite Löschen.
6. Geben Sie im Dialogfeld Löschen den Text Löschen ein, um zu bestätigen, dass Sie das Projekt löschen möchten.
7. Wählen Sie bei Bedarf aus, alle zugehörigen Datensätze und Modelle zu löschen.
8. Wählen Sie Delete project (Projekt löschen).

Löschen eines Projekts (SDK)

Sie löschen ein Amazon Lookout for Vision Vision-Projekt, indem Sie das Projekt, das Sie löschen möchten, anrufen [DeleteProject](#) und den Namen angeben.

Bevor Sie ein Projekt löschen können, müssen Sie zuerst alle Modelle im Projekt löschen. Weitere Informationen finden Sie unter [Löschen eines Modells \(SDK\)](#). Sie müssen auch die mit dem Modell verknüpften Datensätze löschen. Weitere Informationen finden Sie unter [Löschen eines Datensatzes](#).

Es kann einige Momente dauern, das Projekt zu löschen. Während dieser Zeit lautet der Status des Projekts DELETING. Das Projekt wird gelöscht, wenn ein nachfolgender Aufruf DeleteProject das von Ihnen gelöschte Projekt nicht enthält.

So löschen Sie ein Projekt (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Code, um ein Projekt zu löschen.

AWS CLI

Ändern Sie den Wert von `project-name` in den Namen des Projekts, das Sie löschen möchten.


```
aws lookoutvision delete-project --project-name project_name \  
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
/**  
 * 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();
    }
}
```

Ihre Datensätze anzeigen

Ein Projekt kann aus einem einzigen Datensatz bestehen, der zum Trainieren und Testen Ihres Modells verwendet wird. Alternativ können Sie separate Trainings- und Testdatensätze verwenden. Sie können die Konsole verwenden, um Ihre Datensätze anzuzeigen. Sie können den `DescribeDataset` Vorgang auch verwenden, um Informationen zu einem Datensatz abzurufen (Training oder Test).

Die Datensätze in einem Projekt anzeigen (Konsole)

Führen Sie die Schritte im folgenden Verfahren aus, um die Datensätze Ihres Projekts in der Konsole anzuzeigen.

So zeigen Sie Ihre Datensätze an (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.

4. Wählen Sie auf der Seite Projekte das Projekt aus, das die Datensätze enthält, die Sie anzeigen möchten.
5. Wählen Sie im linken Navigationsbereich Datensatz aus, um die Datensatzdetails anzuzeigen. Wenn Sie über einen Trainings- und einen Testdatensatz verfügen, wird für jeden Datensatz eine Registerkarte angezeigt.

Die Datensätze in einem Projekt (SDK) anzeigen

Sie können den `DescribeDataset` Vorgang verwenden, um Informationen über den Trainings- oder Testdatensatz abzurufen, der einem Projekt zugeordnet ist.

So zeigen Sie Ihre Datensätze an (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um einen Datensatz anzuzeigen.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das das Modell enthält, das Sie anzeigen möchten.
- `dataset-type` auf den Datasettyp, den Sie anzeigen möchten (`train` oder `test`).

```
aws lookoutvision describe-dataset --project-name project name \  
  --dataset-type train or test \  
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```

/**
 * Gets the description for a Amazon Lookout for Vision dataset.
 *
 * @param lfvcClient    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 lfvcClient,
        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 =
        lfvcClient.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;
}
```

```
}
```

Hinzufügen von Bildern zu Ihrem Datensatz

Nachdem Sie einen Datensatz erstellt haben, möchten Sie dem Datensatz möglicherweise weitere Bilder hinzufügen. Wenn die Modellauswertung beispielsweise auf ein schlechtes Modell hinweist, können Sie die Qualität Ihres Modells verbessern, indem Sie mehr Bilder hinzufügen. Wenn Sie einen Testdatensatz erstellt haben, kann das Hinzufügen weiterer Bilder die Genauigkeit der Leistungskennzahlen Ihres Modells erhöhen.

Trainieren Sie Ihr Modell nach der Aktualisierung Ihrer Datensätze erneut.

Themen

- [Weitere Bilder hinzufügen](#)
- [Weitere Bilder hinzufügen \(SDK\)](#)

Weitere Bilder hinzufügen

Sie können Ihren Datensätzen weitere Bilder hinzufügen, indem Sie Bilder von Ihrem lokalen Computer hochladen. Verwenden Sie den Vorgang, um mit dem SDK weitere beschriftete Bilder hinzuzufügen. [UpdateDatasetEntries](#)

So fügen Sie Ihrem Datensatz (Konsole) weitere Bilder hinzu

1. Wählen Sie Aktionen aus und wählen Sie den Datensatz aus, dem Sie Bilder hinzufügen möchten.
2. Wählen Sie die Bilder aus, die Sie in den Datensatz hochladen möchten. Sie können die Bilder ziehen oder die Bilder auswählen, die Sie von Ihrem lokalen Computer hochladen möchten. Sie können bis zu 30 Bilder gleichzeitig hochladen.
3. Wählen Sie Bilder hochladen.
4. Wählen Sie Änderungen speichern.

Wenn Sie mit dem Hinzufügen weiterer Bilder fertig sind, müssen Sie sie beschriften, damit sie zum Trainieren des Modells verwendet werden können. Weitere Informationen finden Sie unter [Bilder klassifizieren \(Konsole\)](#).

Weitere Bilder hinzufügen (SDK)

Verwenden Sie den [UpdateDatasetEntries](#) Vorgang, um weitere beschriftete Bilder mit dem SDK hinzuzufügen. Sie stellen eine Manifestdatei bereit, die die Bilder enthält, die Sie hinzufügen möchten. Sie können auch vorhandene Bilder aktualisieren, indem Sie das Bild im `source-ref` Feld der JSON-Zeile in der Manifestdatei angeben. Weitere Informationen finden Sie unter [Erstellen einer Manifestdatei](#).

So fügen Sie mehr Bilder zu einem Datensatz hinzu (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um einem Datensatz weitere Bilder hinzuzufügen.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das den Datensatz enthält, den Sie aktualisieren möchten.
- `dataset-type` auf den Datensatztyp, den Sie aktualisieren möchten (`train` oder `test`).
- `changes` an den Speicherort der Manifestdatei, die Datensatz-Aktualisierungen enthält.

```
aws lookoutvision update-dataset-entries\  
  --project-name project\  
  --dataset-type train or test\  
  --changes fileb://manifest file \  
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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 completed 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
/**
 * Updates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision updates the dataset.
 *
 * @param lfvcClient    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 lfvcClient,
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();

    lfvcClient.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. Wiederholen Sie den vorherigen Schritt und geben Sie Werte für den anderen Datensatztyp an.

Bilder aus Ihrem Datensatz entfernen

Sie können Bilder nicht direkt aus einem Datensatz löschen. Stattdessen müssen Sie den vorhandenen Datensatz löschen und einen neuen Datensatz ohne die Bilder erstellen, die Sie entfernen möchten. Wie Sie Bilder entfernen, hängt davon ab, wie Sie Bilder in den vorhandenen Datensatz importiert haben ([Manifestdatei](#), [Amazon S3 S3-Bucket](#) oder [lokaler Computer](#)).

Sie können das AWS SDK auch verwenden, um Bilder zu entfernen. Dies ist nützlich, wenn Sie ein Bildsegmentierungsmodell ohne eine [Manifestdatei für die Bildsegmentierung](#) erstellen, sodass die Bildmasken nicht mit der Amazon Lookout for Vision Vision-Konsole neu gezeichnet werden müssen.

Themen

- [Bilder aus einem Datensatz entfernen \(Konsole\)](#)
- [Bilder aus einem Datensatz entfernen \(SDK\)](#)

Bilder aus einem Datensatz entfernen (Konsole)

Gehen Sie wie folgt vor, um Bilder mit der Amazon Lookout for Vision Vision-Konsole aus einem Datensatz zu entfernen.

Um Bilder aus einem Datensatz (Konsole) zu entfernen

1. [Öffnen Sie](#) die Datensatz-Galerie des Projekts.
2. Notieren Sie sich den Namen der einzelnen Bilder, die Sie entfernen möchten.
3. [Löschen Sie](#) den vorhandenen Datensatz.
4. Führen Sie eine der folgenden Aktionen aus:
 - Wenn Sie den Datensatz mit einer Manifestdatei erstellt haben, gehen Sie wie folgt vor:
 - a. Öffnen Sie in einem Texteditor die Manifestdatei, mit der Sie den Datensatz erstellt haben.
 - b. Entfernen Sie die JSON-Zeile für jedes Bild, das Sie in Schritt 2 notiert haben. Sie können die JSON-Zeile für ein Bild identifizieren, indem Sie das `source-ref` Feld überprüfen.
 - c. Speichern Sie die Manifestdatei.
 - d. [Erstellen Sie](#) einen neuen Datensatz mit der aktualisierten Manifestdatei.
 - Wenn Sie den Datensatz aus Bildern erstellt haben, die aus einem Amazon S3 S3-Bucket importiert wurden, gehen Sie wie folgt vor:
 - a. [Löschen](#) Sie die Bilder, die Sie in Schritt 2 notiert haben, aus dem Amazon S3 S3-Bucket.

- b. [Erstellen Sie](#) einen neuen Datensatz mit den verbleibenden Bildern im Amazon S3 S3-Bucket. Wenn Sie Bilder nach Ordernamen klassifizieren, müssen Sie die Bilder im nächsten Schritt nicht klassifizieren.
 - c. Führen Sie eine der folgenden Aktionen aus:
 - Wenn Sie ein Bildklassifizierungsmodell erstellen, [klassifizieren](#) Sie jedes unbeschriftete Bild.
 - Wenn Sie ein Bildsegmentierungsmodell erstellen, [klassifizieren und segmentieren](#) Sie jedes unbeschriftete Bild.
 - Wenn Sie den Datensatz aus Bildern erstellt haben, die von einem lokalen Computer importiert wurden, gehen Sie wie folgt vor:
 - a. Erstellen Sie auf Ihrem Computer einen Ordner mit den Bildern, die Sie verwenden möchten. Fügen Sie die Bilder, die Sie aus dem Datensatz entfernen möchten, nicht hinzu. Weitere Informationen finden Sie unter [Erstellen eines Datensatzes mit Bildern, die auf Ihrem lokalen Computer gespeichert sind](#).
 - b. [Erstellen Sie](#) den Datensatz mit den Bildern in dem Ordner, den Sie in Schritt 4.a erstellt haben.
 - c. Führen Sie eine der folgenden Aktionen aus:
 - Wenn Sie ein Bildklassifizierungsmodell erstellen, [klassifizieren](#) Sie jedes unbeschriftete Bild.
 - Wenn Sie ein Bildsegmentierungsmodell erstellen, [klassifizieren und segmentieren](#) Sie jedes unbeschriftete Bild.
5. [Trainieren](#) Sie das Modell.

Bilder aus einem Datensatz entfernen (SDK)

Sie können das AWS SDK verwenden, um Bilder aus einem Datensatz zu entfernen.

Um Bilder aus einem Datensatz zu entfernen (SDK)

1. [Öffnen Sie](#) die Datensatz-Galerie des Projekts.
2. Notieren Sie sich den Namen der einzelnen Bilder, die Sie entfernen möchten.
3. Exportieren Sie die JSON-Zeilen für den Datensatz mithilfe der [ListDatasetEntries](#) Operation.
4. [Erstellen Sie](#) eine Manifestdatei mit den exportierten JSON-Zeilen.

5. Öffnen Sie die Manifestdatei in einem Texteditor.
6. Entfernen Sie die JSON-Zeile für jedes Bild, das Sie in Schritt 2 notiert haben. Sie können die JSON-Zeile für ein Bild identifizieren, indem Sie das `source-ref` Feld überprüfen.
7. Speichern Sie die Manifestdatei.
8. [Löschen Sie](#) den vorhandenen Datensatz.
9. [Erstellen Sie](#) einen neuen Datensatz mit der aktualisierten Manifestdatei.
10. [Trainieren](#) Sie das Modell.

Löschen eines Datensatzes

Sie können einen Datensatz mithilfe der Konsole oder der `DeleteDataset` Operation aus einem Projekt löschen. Die Bilder, auf die ein Datensatz verweist, werden nicht gelöscht. Wenn Sie den Testdatensatz aus einem Projekt löschen, das über einen Trainings- und einen Testdatensatz verfügt, wird das Projekt auf ein einzelnes Datensatzprojekt zurückgesetzt. Der verbleibende Datensatz wird während des Trainings aufgeteilt, um einen Trainings- und Testdatensatz zu erstellen. Wenn Sie den Trainingsdatensatz löschen, können Sie ein Modell im Projekt erst trainieren, wenn Sie einen neuen Trainingsdatensatz erstellt haben.

Löschen eines Datensatzes (Konsole)

Gehen Sie wie im folgenden Verfahren beschrieben vor, um einen Datensatz zu löschen. Wenn Sie alle Datensätze in einem Projekt löschen, wird die Seite Datensatz erstellen angezeigt.

Um einen Datensatz zu löschen (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie auf der Seite Projekte das Projekt aus, das den Datensatz enthält, den Sie löschen möchten.
5. Wählen Sie im linken Navigationsbereich Dataset aus.
6. Wählen Sie Aktionen und dann den Datensatz aus, den Sie löschen möchten.
7. Geben Sie im Dialogfeld Löschen den Text Löschen ein, um zu bestätigen, dass Sie den Datensatz löschen möchten.

8. Wählen Sie Trainingsdatensatz löschen oder Testdatensatz löschen, um den Datensatz zu löschen.

Löschen eines Datensatzes (SDK)

Verwenden Sie den `DeleteDataset` Vorgang, um einen Datensatz zu löschen.

Um einen Datensatz zu löschen (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um ein Modell zu löschen.

CLI

Ändern Sie den Wert des Folgenden

- `project-name` in den Namen des Projekts, das das Modell enthält, das Sie löschen möchten.
- `dataset-type` entweder `train` oder `test`, je nachdem, welchen Datensatz Sie löschen möchten. Wenn Sie ein einzelnes Datensatzprojekt haben, geben Sie `train` an, dass der Datensatz gelöscht werden soll.

```
aws lookoutvision delete-dataset --project-name project name \  
  --dataset-type dataset type \  
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem GitHub Beispiel-Repository des AWS Documentation SDK. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```

/**
 * Deletes the train or test dataset in an Amazon Lookout for Vision project.
 *
 * @param lfvcClient 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 lfvcClient, 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();

lfrClient.deleteDataset(deleteDatasetRequest);

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

Exportieren von Datensätzen aus einem Projekt (SDK)

Sie können das AWS SDK verwenden, um Datensätze aus einem Amazon Lookout for Vision Vision-Projekt an einen Amazon S3 S3-Bucket-Speicherort zu exportieren.

Durch das Exportieren eines Datensatzes können Sie Aufgaben wie das Erstellen eines Lookout for Vision Vision-Projekts mit einer Kopie der Datensätze eines Quellprojekts erledigen. Sie können auch eine Momentaufnahme der Datensätze erstellen, die für eine bestimmte Version eines Modells verwendet wurden.

Der Python-Code in diesem Verfahren exportiert den Trainingsdatensatz (Manifest und Datensatzbilder) für ein Projekt an einen von Ihnen angegebenen Amazon S3 S3-Zielort. Falls im Projekt vorhanden, exportiert der Code auch das Manifest und die Datensatzbilder des Testdatensatzes. Das Ziel kann sich in demselben Amazon S3 S3-Bucket wie das Quellprojekt oder in einem anderen Amazon S3 S3-Bucket befinden. Der Code verwendet den [ListDatasetEntries](#) Vorgang, um die Manifestdateien des Datensatzes abzurufen. Amazon S3 S3-Operationen kopieren die Datensatz-Bilder und die aktualisierten Manifestdateien an den Amazon S3 S3-Zielort.

Dieses Verfahren zeigt, wie die Datensätze eines Projekts exportiert werden. Es zeigt auch, wie Sie mit den exportierten Datensätzen ein neues Projekt erstellen.

Um die Datensätze aus einem Projekt zu exportieren (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).

2. Ermitteln Sie den Amazon S3 S3-Zielpfad für den Datensatzexport. Stellen Sie sicher, dass sich das Ziel in einer [AWS Region](#) befindet, die Amazon Lookout for Vision unterstützt. Informationen zum Erstellen eines neuen Amazon S3 S3-Buckets finden Sie unter [Bucket erstellen](#).
3. Stellen Sie sicher, dass der Benutzer über Zugriffsberechtigungen für den Amazon S3 S3-Zielpfad für den Datensatzexport und die S3-Speicherorte für die Bilddateien in den Quellprojektdatensätzen verfügt. Sie können die folgende Richtlinie verwenden, die davon ausgeht, dass sich die Bilddateien an jedem beliebigen Ort befinden können. *bucket/*
path Ersetzen Sie durch den Ziel-Bucket und den Pfad für den Dataset-Export.

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

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:
 - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
 - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

4. Speichern Sie den folgenden Code in eine Datei mit dem Namen `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. Führen Sie den Code aus. Geben Sie die folgenden Befehlszeilenargumente an:

- **Projekt** — Der Name des Quellprojekts, das die Datensätze enthält, die Sie exportieren möchten.
- **destination** — Der Amazon S3 S3-Zielpfad für die Datensätze.

Beispiel: `python dataset_export.py myproject s3://bucket/path/`

6. Notieren Sie sich die Speicherorte der Manifestdateien, die im Code angezeigt werden. Sie benötigen sie in Schritt 8.
7. Erstellen Sie ein neues Lookout for Vision Vision-Projekt mit exportiertem Datensatz, indem Sie den Anweisungen unter [Erstellen Sie Ihr Projekt](#) folgen.
8. Führen Sie eine der folgenden Aktionen aus:
- Verwenden Sie die Lookout for Vision Vision-Konsole, um Datensätze für Ihr neues Projekt zu erstellen, indem Sie den Anweisungen unter folgen. [Einen Datensatz mit einer Manifestdatei erstellen \(Konsole\)](#) Sie müssen die Schritte 1—6 nicht ausführen.

Gehen Sie für Schritt 12 wie folgt vor:

- a. Wenn das Quellprojekt über einen Testdatensatz verfügt, wählen Sie Separate Trainings- und Testdatensätze aus, andernfalls wählen Sie Einzelner Datensatz.
 - b. Geben Sie für den Speicherort der Datei .manifest den Speicherort der entsprechenden Manifest-Datei (Train oder Test) ein, die Sie in Schritt 6 notiert haben.
 - Verwenden Sie den [CreateDataset](#)Vorgang, um Datensätze für Ihr neues Projekt zu erstellen, indem Sie den Code unter verwenden. [Einen Datensatz mit einer Manifestdatei \(SDK\) erstellen](#) Verwenden Sie für den manifest_file Parameter den Speicherort der Manifestdatei, den Sie in Schritt 6 notiert haben. Wenn das Quellprojekt über einen Testdatensatz verfügt, verwenden Sie den Code erneut, um den Testdatensatz zu erstellen.
9. Wenn Sie bereit sind, trainieren Sie das Modell, indem Sie den Anweisungen unter folgen [Trainieren Ihres Modells](#).

Deine Modelle ansehen

Ein Projekt kann mehrere Versionen eines Modells enthalten. Sie können die Konsole verwenden, um die Modelle in einem Projekt anzuzeigen. Sie können auch die ListModels-Operation verwenden.

Note

Die Liste der Modelle ist letztendlich konsistent. Wenn Sie ein Modell erstellen, müssen Sie möglicherweise eine Weile warten, bis die Modellliste auf dem neuesten Stand ist.

Ihre Modelle anzeigen (Konsole)

Führen Sie die Schritte im folgenden Verfahren aus, um die Modelle Ihres Projekts in der Konsole anzuzeigen.

So zeigen Sie Ihre Modelle an (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie auf der Seite Projekte das Projekt aus, das die Modelle enthält, die Sie anzeigen möchten.

5. Wählen Sie im linken Navigationsbereich Modelle aus und zeigen Sie dann die Modelldetails an.

Ihre Modelle anzeigen (SDK)

Um die Versionen eines Modells anzuzeigen, verwenden Sie die `ListModels` Operation. Verwenden Sie den `DescribeModel` Vorgang, um Informationen zu einer bestimmten Modellversion abzurufen. Im folgenden Beispiel werden alle Modellversionen in einem Projekt aufgeführt und anschließend Leistungs- und Ausgabekonfigurationsinformationen für einzelne Modellversionen angezeigt.

So zeigen Sie Ihre Modelle an (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um Ihre Modelle aufzulisten und Informationen zu einem Modell abzurufen.

CLI

Verwenden Sie den `list-models` Befehl, um die Modelle in einem Projekt aufzulisten.

Ändern Sie den folgenden Wert:

- `project-name` in den Namen des Projekts, das das Modell enthält, das Sie anzeigen möchten.

```
aws lookoutvision list-models --project-name project name \  
--profile lookoutvision-access
```

Verwenden Sie den `describe-model` Befehl, um Informationen zu einem Modell abzurufen. Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das das Modell enthält, das Sie anzeigen möchten.
- `model-version` zu der Version des Modells, das Sie beschreiben möchten.

```
aws lookoutvision describe-model --project-name project name\
  --model-version model version \
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
/**
 * 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();
}

```

Löschen eines Modells

Sie können eine Version eines Modells mithilfe der Konsole oder mithilfe der `DeleteModel` Operation löschen. Sie können keine Modellversion löschen, die gerade läuft oder trainiert wird.

Wenn es sich bei dem Modell um eine laufende Version handelt, beenden Sie mit diesem `StopModel` Vorgang zunächst die Modellversion. Weitere Informationen finden Sie unter [Ihr Amazon Lookout for Vision Vision-Modell beenden](#). Wenn ein Modell trainiert wird, warten Sie, bis es fertig ist, bevor Sie das Modell löschen.

Das Löschen eines Modells kann einige Sekunden dauern. Um festzustellen, ob ein Modell gelöscht wurde, rufen Sie auf [ListProjects](#) und überprüfen Sie, ob sich die Version des Modells (`ModelVersion`) im `Models` Array befindet.

Löschen eines Modells (Konsole)

Gehen Sie wie folgt vor, um ein Modell von der Konsole zu löschen.

So löschen Sie ein Modell (Konsole)

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie auf der Seite Projekte das Projekt aus, das das Modell enthält, das Sie löschen möchten.
5. Wählen Sie im linken Navigationsbereich Models (Modelle) aus.
6. Wählen Sie in der Modellansicht das Optionsfeld für das Modell aus, das Sie löschen möchten.
7. Wählen Sie oben auf der Seite Löschen.
8. Geben Sie im Dialogfeld Löschen den Text Löschen ein, um zu bestätigen, dass Sie das Modell löschen möchten.
9. Wählen Sie Modell löschen, um das Modell zu löschen.

Löschen eines Modells (SDK)

Gehen Sie wie folgt vor, um das Modell während der `DeleteModel` Operation zu löschen.

So löschen Sie ein Modell (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie den AWS CLI und den AWS SDKs. Weitere Informationen finden Sie unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).
2. Verwenden Sie den folgenden Beispielcode, um ein Modell zu löschen.

CLI

Ändern Sie die folgenden Werte:

- `project-name` auf den Namen des Projekts, das das Modell enthält, das Sie löschen möchten.
- `model-version` zu der Version des Modells, das Sie löschen möchten.

```
aws lookoutvision delete-model --project-name project name \  
  --model-version model version \  
  --profile lookoutvision-access
```

Python

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
@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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie [hier](#).

```
/**
 * Deletes an Amazon Lookout for Vision model.
 *
 * @param lfvcClient    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 lfvcClient,
                               String projectName,
                               String modelVersion) throws LookoutVisionException,
                               InterruptedException {

    DeleteModelRequest deleteModelRequest = DeleteModelRequest.builder()
        .projectName(projectName)
        .modelVersion(modelVersion)
        .build();

    lfvcClient.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);
}
```

Modelle taggen

Sie können Ihre Amazon Lookout for Vision-Modelle mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht. Um beispielsweise die Abrechnung Ihrer Modelle zu erleichtern, könnten Sie Ihre Modelle

mit einem `Cost center` Schlüssel versehen und die entsprechende Kostenstellennummer als Wert hinzufügen. Weitere Informationen finden Sie unter [Taggen von AWS-Ressourcen](#).

Verwenden Sie Tags, um:

- Verfolgen Sie die Abrechnung für ein Modell mithilfe von Kostenzuweisungs-Tags. Weitere Informationen finden Sie unter [Verwendung von Kostenzuordnungs-Tags](#).
- Den Zugriff auf ein Modell kontrollieren mit Identity and Access Management (IAM). Weitere Informationen finden Sie unter [Steuern des Zugriffs auf AWS-Ressourcen mithilfe von Ressourcen-Tags](#).
- Die Modellverwaltung automatisieren. Sie können zum Beispiel automatische Start- oder Stoppskripte ausführen, die Entwicklungsmodelle außerhalb der Geschäftszeiten abschalten, um Kosten zu sparen. Weitere Informationen finden Sie unter [Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells](#).

Sie können Modelle mit der Amazon Lookout for Vision Vision-Konsole oder mit der AWS SDKs kennzeichnen.

Themen

- [Taggen von Modellen \(Konsole\)](#)
- [Modelle kennzeichnen \(SDK\)](#)

Taggen von Modellen (Konsole)

Sie können die Amazon Lookout for Vision Vision-Konsole verwenden, um Tags zu Modellen hinzuzufügen, die an ein Modell angehängten Tags anzuzeigen und Tags zu entfernen.

Hinzufügen oder Entfernen von Tags (Konsole)

In diesem Verfahren wird erklärt, wie Sie einem vorhandenen Modell Tags hinzufügen oder Tags daraus entfernen. Sie können einem neuen Modell auch Tags hinzufügen, wenn es trainiert wird. Weitere Informationen finden Sie unter [Trainieren Ihres Modells](#).

Um einem vorhandenen Modell (Konsole) Tags hinzuzufügen oder Tags daraus zu entfernen

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.

2. Wählen Sie Erste Schritte.
3. Klicken Sie im Navigationsbereich auf Projekte.
4. Wählen Sie auf der Ressourcenseite Projekte das Projekt aus, das das Modell enthält, dem Sie Tags hinzufügen möchten.
5. Wählen Sie im Navigationsbereich unter dem Projekt, das Sie zuvor ausgewählt haben, Modelle aus.
6. Wählen Sie im Abschnitt Modelle das Modell aus, dem Sie ein Tag hinzufügen möchten.
7. Wählen Sie auf der Seite mit den Modelldetails die Registerkarte Tags aus.
8. Wählen Sie im Abschnitt Tags (Markierungen) die Option Manage tags (Tags (Markierungen) verwalten).
9. Wählen Sie auf der Seite Tags verwalten Neuen Tag hinzufügen aus.
10. Geben Sie einen Key Schlüssel und einen Wert ein.
 - a. Geben Sie für Schlüssel einen Schlüsselnamen ein.
 - b. Geben Sie unter Value (Wert) einen Wert ein.
11. Wiederholen Sie die Schritte 9 und 10, um weitere Tags hinzuzufügen.
12. (Optional) Wählen Sie zum Entfernen eines Tags neben dem Tag, den Sie entfernen möchten, Entfernen. Wenn Sie ein zuvor gespeichertes Tag entfernen, wird es entfernt, wenn Sie Ihre Änderungen speichern.
13. Wählen Sie Änderungen speichern aus, um Ihre Änderungen zu speichern.

Modell-Tags anzeigen (Konsole)

Sie können die Amazon Lookout for Vision Vision-Konsole verwenden, um die an ein Modell angehängten Tags anzuzeigen.

Um die Tags anzuzeigen, die allen Modellen innerhalb eines Projekts zugeordnet sind, müssen Sie das AWS-SDK verwenden. Weitere Informationen finden Sie unter [Modell-Tags auflisten \(SDK\)](#).

So zeigen Sie die einem Modell angefügten Tags an

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Klicken Sie im Navigationsbereich auf Projekte.

4. Wählen Sie auf der Seite Projektressourcen das Projekt aus, das das Modell enthält, dessen Tag Sie anzeigen möchten.
5. Wählen Sie im Navigationsbereich unter dem Projekt, das Sie zuvor ausgewählt haben, Modelle aus.
6. Wählen Sie im Abschnitt Modelle das Modell aus, dessen Tag Sie anzeigen möchten.
7. Wählen Sie auf der Seite mit den Modelldetails die Registerkarte Tags aus. Die Tags werden im Abschnitt Tags angezeigt.

Modelle kennzeichnen (SDK)

Sie können das AWS SDK verwenden, um:

- Hinzufügen von Tags zu einem neuen Modell
- Hinzufügen von Tags zu einem vorhandenen Modell
- Auflisten der einem Modell angefügten Tags
- Entfernen von Tags von einem Modell

Dieser Abschnitt enthält AWS CLI Beispiele. Falls Sie das nicht installiert haben AWS CLI, finden Sie weitere Informationen unter [Schritt 4: Richten Sie das AWS CLI und ein AWS SDKs](#).

Hinzufügen von Tags zu einem neuen Modell (SDK)

Sie können einem Modell Tags hinzufügen, wenn Sie es mithilfe der [CreateModel](#) Operation erstellen. Geben Sie ein oder mehrere Tags im Tags-Array-Eingabeparameter an.

```
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
```

Informationen zum Erstellen und Trainieren eines Modells finden Sie unter [Ein Modell trainieren \(SDK\)](#).

Hinzufügen von Tags zu einem vorhandenen Modell (SDK)

Verwenden Sie die [TagResource](#) Operation, um einem vorhandenen Modell ein oder mehrere Tags hinzuzufügen. Geben Sie den Amazon-Ressourcennamen (ARN) (ResourceArn) des Modells und die Tags (Tags) an, die Sie hinzufügen möchten.

```
aws lookoutvision tag-resource --resource-arn "resource-arn" \
  --tags '[{"Key": "Key", "Value": "Value"}]' \
  --profile lookoutvision-access
```

Ein Beispiel für Java-Code finden Sie unter [TagModel](#).

Modell-Tags auflisten (SDK)

Um die an ein Modell angehängten Tags aufzulisten, verwenden Sie die [ListTagsForResource](#) Operation und geben Sie den Amazon-Ressourcennamen (ARN) des Modells, die (ResourceArn), an. Die Antwort ist eine Zuordnung von Tag-Schlüsseln und -Werten, die an das angegebene Modell angefügt sind.

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

Um zu sehen, welche Modelle in einem Projekt über ein bestimmtes Tag verfügen, rufen Sie `ListModels` auf, um eine Liste der Modelle abzurufen. Rufen Sie dann in der Antwort von `ListModels` für jedes Modell `ListTagsForResource` auf. Überprüfen Sie die Antwort von `ListTagsForResource`, um festzustellen, ob das erforderliche Tag vorhanden ist.

Ein Beispiel für Java-Code finden Sie unter [ListModelTags](#). Ein Beispiel für Python-Code, der in allen Projekten nach einem Tag-Wert sucht, finden Sie unter [find_tag.py](#).

Tags aus einem Modell entfernen (SDK)

Verwenden Sie die [UntagResource](#) Operation, um ein oder mehrere Tags aus einem Modell zu entfernen. Geben Sie den Amazon-Ressourcennamen (ARN) (ResourceArn) des Modells und die Tag-Schlüssel (Tag-Keys) an, die Sie entfernen möchten.

```
aws lookoutvision untag-resource --resource-arn resource-arn \
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

Ein Beispiel für Java-Code finden Sie unter [UntagModel](#).

Ihre Aufgaben zur Erkennung von Studien anzeigen

Sie können Ihre Erkennungen in der Testversion mithilfe der Konsole anzeigen. Sie können das AWS SDK nicht verwenden, um Aufgaben zur Erkennung von Testversionen anzuzeigen.

Note

Die Liste der Erkennungen von Studien ist letztendlich konsistent. Wenn Sie eine Testererkennung erstellen, müssen Sie möglicherweise eine Weile warten, bis die Liste der erkannten Studien aktualisiert ist.

Ihre Aufgaben zur Erkennung von Testversionen anzeigen (Konsole)

Gehen Sie wie folgt vor, um sich Ihre Testergebnisse anzusehen.

Um Ihre Aufgaben zur Erkennung von Studien einzusehen

1. Öffnen Sie die Amazon Lookout for Vision Vision-Konsole unter <https://console.aws.amazon.com/lookoutvision/>.
2. Wählen Sie Erste Schritte.
3. Wählen Sie im linken Navigationsbereich Trial Detections aus.
4. Wählen Sie auf der Seite zur Erkennung von Testversionen eine Aufgabe zur Erkennung von Testversionen aus, um deren Details anzuzeigen.

Beispielcode und Datensätze

Im Folgenden finden Sie Codebeispiele und Datensätze, die Sie mit Amazon Lookout for Vision verwenden können.

Themen

- [Beispiel-Code](#)
- [Beispieldatensätze](#)

Beispiel-Code

Die folgenden Codebeispiele für Amazon Lookout for Vision sind verfügbar.

Beispiel	Beschreibung
GitHub	Python-Beispielcode, der ein Amazon Lookout for Vision Vision-Modell trainiert und hostet.
Amazon Lookout for Vision Lab	Ein Python-Notebook, mit dem Sie ein Modell mit den Leiterplatten-Beispielbildern erstellen können.
Python-Beispielcode	Python-Beispiele, die in der Dokumentation zu Amazon Lookout for Vision verwendet werden.
Java-Beispielcode	In der Dokumentation zu Amazon Lookout for Vision verwendete Java-Beispiele.

Beispieldatensätze

Im Folgenden finden Sie Beispieldatensätze, die Sie mit Amazon Lookout for Vision verwenden können.

Themen

- [Datensätze zur Bildsegmentierung](#)

- [Datensatz zur Bildklassifizierung](#)

Datensätze zur Bildsegmentierung

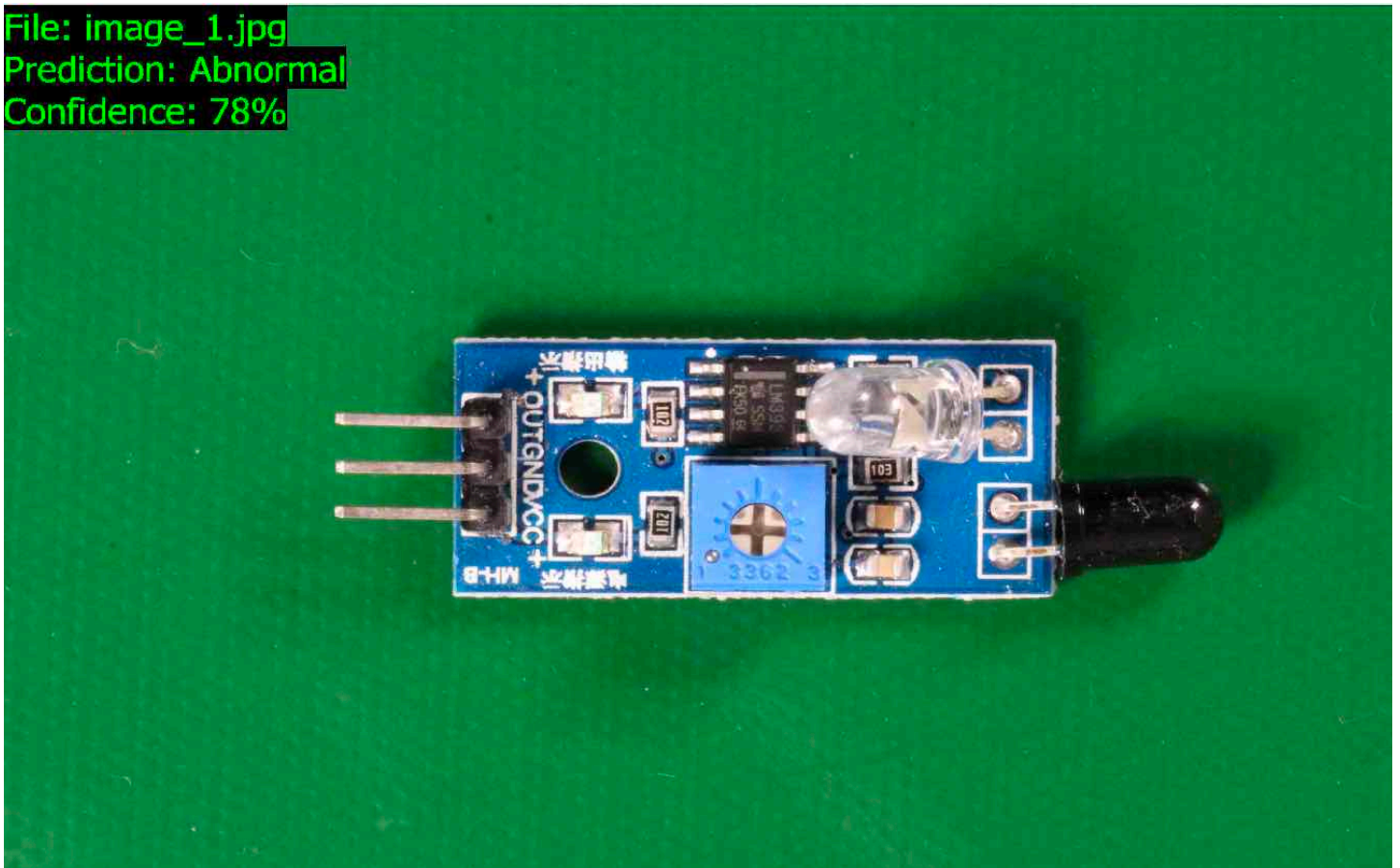
[Erste Schritte mit Amazon Lookout for Vision](#) stellt einen Datensatz mit defekten Cookies bereit, den Sie verwenden können, um ein [Bildsegmentierungsmodell](#) zu erstellen.

Einen weiteren Datensatz, der ein Bildsegmentierungsmodell erstellt, finden Sie unter [Identifizieren der Position von Anomalien mithilfe von Amazon Lookout for Vision am Edge ohne Verwendung einer GPU](#).

Datensatz zur Bildklassifizierung

Amazon Lookout for Vision bietet Beispielbilder von Leiterplatten, mit denen Sie ein [Bildklassifizierungsmodell](#) erstellen können.

File: image_1.jpg
Prediction: Abnormal
Confidence: 78%



Sie können die Bilder aus dem <https://github.com/aws-samples/amazon-lookout-for-vision> GitHubRepository kopieren. Die Bilder befinden sich im circuitboard Ordner.

Der `circuitboard` Ordner hat die folgenden Ordner.

- `train`— Bilder, die Sie in einem Trainingsdatensatz verwenden können.
- `test`— Bilder, die Sie in einem Testdatensatz verwenden können.
- `extra_images`— Bilder, die Sie verwenden können, um eine Testerkennung durchzuführen oder Ihr trainiertes Modell mit der [DetectAnomalies](#) Operation auszuprobieren.

Die `test` Ordner `train` und haben jeweils einen Unterordner mit dem Namen `normal` (enthält normale Bilder) und einen Unterordner mit dem Namen `anomaly` (enthält Bilder mit Anomalien).

 Note

Später, wenn Sie mit der Konsole einen Datensatz erstellen, kann Amazon Lookout for Vision die Ordernamen (`normal` und `anomaly`) verwenden, um die Bilder automatisch zu beschriften. Weitere Informationen finden Sie unter [the section called “Amazon-S3-Bucket”](#).

Um die Datensatzbilder vorzubereiten

1. Klonen Sie das <https://github.com/aws-samples/amazon-lookout-for-vision> Repository auf Ihren Computer. Weitere Informationen finden Sie unter [Ein Repository klonen](#).
2. Erstellen Sie einen Amazon-S3-Bucket. Weitere Informationen finden Sie unter [Wie erstelle ich einen S3-Bucket?](#).
3. Geben Sie an der Befehlszeile den folgenden Befehl ein, um die Datensatzbilder von Ihrem Computer in Ihren Amazon S3 S3-Bucket zu kopieren.

```
aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/circuitboard
```

Nach dem Hochladen der Bilder können Sie ein Modell erstellen. Sie können die Bilder automatisch klassifizieren, indem Sie die Bilder von dem Amazon S3 S3-Speicherort hinzufügen, in den Sie die Leiterplattenbilder zuvor hochgeladen haben. Denken Sie daran, dass Ihnen jedes erfolgreiche Training eines Modells und die Zeit, die ein Modell läuft (gehostet), in Rechnung gestellt werden.

Um ein Klassifikationsmodell zu erstellen

1. Tun [Ein Projekt erstellen \(Konsole\)](#).

2. Tun [Erstellen eines Datensatzes mit Bildern, die in einem Amazon S3 S3-Bucket gespeichert sind](#).
 - Wählen Sie für Schritt 6 die Registerkarte Trainings- und Testdatensätze trennen.
 - Geben Sie für Schritt 8a den S3-URI für die Trainingsbilder ein, die Sie unter [So bereiten Sie die Datensatzbilder](#) vor. Zum Beispiel `s3://your-bucket/circuitboard/train`. Geben Sie für Schritt 8b die S3-URI für den Testdatensatz ein. Beispiel, `s3://your-bucket/circuitboard/test`.
 - Stellen Sie sicher, dass Sie Schritt 9 ausführen.
3. Tun [Ein Modell \(Konsole\) trainieren](#).
4. Tun [Starten Sie Ihr Modell \(Konsole\)](#).
5. Tun [Erkennung von Anomalien in einem Bild](#). Sie können Bilder aus dem test_images Ordner verwenden.
6. Wenn Sie mit dem Modell fertig sind, tun Sie es [Stoppen Sie Ihr Modell \(Konsole\)](#).

Sicherheit in Amazon Lookout for Vision

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- **Sicherheit der Cloud** — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#). Weitere Informationen zu den Compliance-Programmen, die für Amazon Lookout for Vision gelten, finden Sie unter [AWS-Services in Umfang nach Compliance-Programm](#).
- **Sicherheit in der Cloud** — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Lookout for Vision anwenden können. In den folgenden Themen erfahren Sie, wie Sie Lookout for Vision konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS-Services nutzen können, die Sie bei der Überwachung und Sicherung Ihrer Lookout for Vision Vision-Ressourcen unterstützen.

Themen

- [Datenschutz in Amazon Lookout for Vision](#)
- [Identitäts- und Zugriffsmanagement für Amazon Lookout for Vision](#)
- [Konformitätsprüfung für Amazon Lookout for Vision](#)
- [Resilienz in Amazon Lookout for Vision](#)
- [Infrastruktursicherheit in Amazon Lookout for Vision](#)

Datenschutz in Amazon Lookout for Vision

Das AWS [Modell](#) der gilt für den Datenschutz in Amazon Lookout for Vision. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der

AWS Cloud alle Systeme laufen. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#). Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag [AWS -Modell der geteilten Verantwortung und in der DSGVO](#) im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-3](#).

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dazu gehört auch, wenn Sie mit Lookout for Vision oder anderen AWS-Services über die Konsole AWS CLI, API oder AWS SDKs arbeiten. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Datenverschlüsselung

In den folgenden Informationen wird erklärt, wo Amazon Lookout for Vision Datenverschlüsselung zum Schutz Ihrer Daten verwendet.

Verschlüsselung im Ruhezustand

Bilder

Um Ihr Modell zu trainieren, erstellt Amazon Lookout for Vision eine Kopie Ihrer ursprünglichen Trainings- und Testbilder. Die kopierten Bilder werden im Ruhezustand in Amazon Simple Storage Service (S3) mithilfe einer serverseitigen Verschlüsselung mit einem AWS-eigener Schlüssel oder einem von Ihnen bereitgestellten Schlüssel verschlüsselt. Die Schlüssel werden mit dem AWS Key Management Service (SSE-KMS) gespeichert. Ihre Quellbilder sind davon nicht betroffen. Weitere Informationen finden Sie unter [Trainieren Ihres Modells](#).

Amazon Lookout for Vision Vision-Modelle

Standardmäßig werden trainierte Modelle und Manifestdateien in Amazon S3 mit serverseitiger Verschlüsselung mit KMS-Schlüsseln verschlüsselt, die im AWS Key Management Service (SSE-KMS) gespeichert sind. Lookout for Vision verwendet eine AWS-eigener Schlüssel. Weitere Informationen finden Sie unter [Schutz von Daten mittels serverseitiger Verschlüsselung](#). Die Trainingsergebnisse werden in den Bucket geschrieben, der im `output_bucket` Eingabeparameter `to CreateModel` angegeben ist. Die Trainingsergebnisse werden mit den konfigurierten Verschlüsselungseinstellungen für den Bucket (`output_bucket`) verschlüsselt.

Konsolen-Bucket für Amazon Lookout for Vision

Die Amazon Lookout for Vision Vision-Konsole erstellt einen Amazon S3 S3-Bucket (Konsolen-Bucket), mit dem Sie Ihre Projekte verwalten können. Der Konsolen-Bucket ist mit der standardmäßigen Amazon-S3-Verschlüsselung verschlüsselt. Weitere Informationen finden Sie unter [Amazon-Simple-Storage-Service-Standardverschlüsselung für S3-Buckets](#). Wenn Sie Ihren eigenen KMS-Schlüssel verwenden, konfigurieren Sie den Konsolen-Bucket, nachdem er erstellt wurde. Weitere Informationen finden Sie unter [Schutz von Daten mittels serverseitiger Verschlüsselung](#). Amazon Lookout for Vision blockiert den öffentlichen Zugriff auf den Konsolen-Bucket.

Verschlüsselung während der Übertragung

Amazon Lookout for Vision API-Endpunkte unterstützen nur sichere Verbindungen über HTTPS. Die gesamte Kommunikation wird mit Transport Layer Security (TLS) verschlüsselt.

Schlüsselverwaltung

Sie können den AWS Key Management Service (KMS) verwenden, um die Verschlüsselung der Eingabebilder zu verwalten, die Sie in Amazon S3 S3-Buckets speichern. Weitere Informationen finden Sie unter [Schritt 5: \(Optional\) Verwenden Ihres eigenen AWS Key Management Service Service-Schlüssels](#).

Standardmäßig werden Ihre Bilder mit einem Schlüssel verschlüsselt, den AWS besitzt und verwaltet. Sie können sich auch dafür entscheiden, Ihren eigenen AWS Key Management Service (KMS) -Schlüssel zu verwenden. Weitere Informationen finden Sie unter [AWS Key Management Service-Konzepte](#).

Richtlinie für den Datenverkehr zwischen Netzwerken

Ein Amazon Virtual Private Cloud (Amazon VPC) -Endpunkt für Amazon Lookout for Vision ist eine logische Einheit innerhalb einer VPC, die nur Konnektivität zu Amazon Lookout for Vision ermöglicht. Amazon VPC leitet Anfragen an Amazon Lookout for Vision weiter und leitet Antworten zurück an die VPC. Weitere Informationen finden Sie unter [VPC-Endpunkte](#) im Amazon VPC-Benutzerhandbuch. Informationen zur Verwendung von Amazon VPC-Endpunkten mit Amazon Lookout for Vision finden Sie unter [Greifen Sie über einen Schnittstellenendpunkt auf Amazon Lookout for Vision zu \(AWS PrivateLink\)](#)

Identitäts- und Zugriffsmanagement für Amazon Lookout for Vision

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu kontrollieren. AWS IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Lookout for Vision Vision-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [So funktioniert Amazon Lookout for Vision mit IAM](#)
- [Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision](#)
- [AWS verwaltete Richtlinien für Amazon Lookout for Vision](#)

- [Problembehebung bei Identität und Zugriff auf Amazon Lookout for Vision](#)

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Lookout for Vision ausführen.

Dienstbenutzer — Wenn Sie den Lookout for Vision Vision-Dienst für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Da Sie für Ihre Arbeit mehr Funktionen von Lookout for Vision verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Wenn Sie auf eine Funktion in Lookout for Vision nicht zugreifen können, finden Sie weitere Informationen unter [Problembehebung bei Identität und Zugriff auf Amazon Lookout for Vision](#).

Service-Administrator — Wenn Sie in Ihrem Unternehmen für die Ressourcen von Lookout for Vision verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf Lookout for Vision. Es ist Ihre Aufgabe, zu bestimmen, auf welche Funktionen und Ressourcen von Lookout for Vision Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM mit Lookout for Vision nutzen kann, finden Sie unter [So funktioniert Amazon Lookout for Vision mit IAM](#)

IAM-Administrator — Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf Lookout for Vision zu verwalten. Beispiele für identitätsbasierte Richtlinien von Lookout for Vision, die Sie in IAM verwenden können, finden Sie unter [Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision](#)

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen.

Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie

sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter [So melden Sie sich bei Ihrem an AWS-Konto](#) im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter [AWS Signature Version 4 für API-Anforderungen](#) im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter [Multi-Faktor-Authentifizierung](#) im AWS IAM Identity Center - Benutzerhandbuch und [AWS Multi-Faktor-Authentifizierung \(MFA\) in IAM](#) im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern](#) im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter [Was ist IAM Identity Center?](#) im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität innerhalb Ihres Unternehmens AWS-Konto, die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter [Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern](#) im IAM-Benutzerhandbuch.

Eine [IAM-Gruppe](#) ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdmins und dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter [Anwendungsfälle für IAM-Benutzer](#) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität innerhalb von Ihrem AWS-Konto, die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie [von einer Benutzer- zu einer IAM-Rolle \(Konsole\) wechseln](#). Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter [Methoden für die Übernahme einer Rolle](#) im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- **Verbundbenutzerzugriff** – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter [Erstellen von Rollen für externe Identitätsanbieter \(Verbund\)](#) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter [Berechtigungssätze](#) im AWS IAM Identity Center -Benutzerhandbuch.
- **Temporäre IAM-Benutzerberechtigungen** – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- **Kontoübergreifender Zugriff** – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.
- **Serviceübergreifender Zugriff** — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
 - **Forward Access Sessions (FAS)** — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services

könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).

- **Servicerolle** – Eine Servicerolle ist eine [IAM-Rolle](#), die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter [Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service](#) im IAM-Benutzerhandbuch.
- **Dienstbezogene Rolle** — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- **Auf Amazon ausgeführte Anwendungen EC2** — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden](#).

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und

Inhalten von JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter [Auswählen zwischen verwalteten und eingebundenen Richtlinien](#) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und

Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#). Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter [Übersicht über ACLs die Zugriffskontrollliste \(ACL\)](#) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im IAM-Benutzerhandbuch.
- **Dienststeuerungsrichtlinien (SCPs)** — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung

mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter [Richtlinien zur Servicesteuerung](#) im AWS Organizations Benutzerhandbuch.

- **Ressourcenkontrollrichtlinien (RCPs)** — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter [Resource Control Policies \(RCPs\)](#) im AWS Organizations Benutzerhandbuch.
- **Sitzungsrichtlinien** – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter [Bewertungslogik für Richtlinien](#).

So funktioniert Amazon Lookout for Vision mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf Lookout for Vision zu verwalten, sollten Sie sich darüber informieren, welche IAM-Funktionen mit Lookout for Vision verwendet werden können.

IAM-Funktionen, die Sie mit Amazon Lookout for Vision verwenden können

IAM-Feature	Lookout for Vision Vision-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (servicespezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Forward Access Sessions (FAS)	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie Lookout for Vision und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im [AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren](#).

Identitätsbasierte Richtlinien für Lookout for Vision

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der [IAM-Referenz für JSON-Richtlinienelemente](#) im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Lookout for Vision

Beispiele für identitätsbasierte Richtlinien von Lookout for Vision finden Sie unter [Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision](#)

Ressourcenbasierte Richtlinien in Lookout for Vision

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#). Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM-Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Politische Maßnahmen für Lookout for Vision

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Lookout for Vision-Aktionen finden Sie unter [Von Amazon Lookout for Vision definierte Aktionen](#) in der Service Authorization Reference.

Richtlinienaktionen in Lookout for Vision verwenden das folgende Präfix vor der Aktion:

```
lookoutvision
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [  
    "lookoutvision:action1",  
    "lookoutvision:action2"  
]
```

Beispiele für identitätsbasierte Richtlinien von Lookout for Vision finden Sie unter [Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision](#)

Politische Ressourcen für Lookout for Vision

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein `Resource` oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen [Amazon-Ressourcennamen \(ARN\)](#) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*" 
```

Eine Liste der Ressourcentypen von Lookout for Vision und ihrer ARNs Eigenschaften finden Sie unter [Von Amazon Lookout for Vision definierte Ressourcen](#) in der Service Authorization Reference. Informationen darüber, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter [Von Amazon Lookout for Vision definierte Aktionen](#).

Beispiele für identitätsbasierte Richtlinien von Lookout for Vision finden Sie unter. [Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision](#)

Schlüssel zu den politischen Bedingungen für Lookout for Vision

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition block`) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die [Bedingungsoperatoren](#) verwenden, z. B. `ist gleich` oder `kleiner als`, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet die Bedingung mithilfe einer logischen OR Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter [IAM-Richtlinienelemente: Variablen und Tags](#) im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter [Kontextschlüssel für AWS globale Bedingungen](#) im IAM-Benutzerhandbuch.

Eine Liste der Bedingungsschlüssel von Lookout for Vision finden Sie unter [Bedingungsschlüssel für Amazon Lookout for Vision](#) in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter [Von Amazon Lookout for Vision definierte Aktionen](#).

Beispiele für identitätsbasierte Richtlinien von Lookout for Vision finden Sie unter [Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision](#)

ACLs auf Lookout for Vision

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Lookout for Vision

Unterstützt ABAC (Tags in Richtlinien): Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungelement einer [Richtlinie Tag-Informationen](#) an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter [Definieren von Berechtigungen mit ABAC-Autorisierung](#) im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe [Attributbasierte Zugriffskontrolle \(ABAC\)](#) verwenden im IAM-Benutzerhandbuch.

Temporäre Anmeldeinformationen mit Lookout for Vision verwenden

Unterstützt temporäre Anmeldeinformationen: Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services [funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit IAM](#).

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter [Wechseln von einer Benutzerrolle zu einer IAM-Rolle \(Konsole\)](#) im IAM-Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen in IAM](#).

Vorwärtszugriffssitzungen für Lookout for Vision

Unterstützt Forward Access Sessions (FAS): Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion

in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).

Servicerollen für Lookout for Vision

Unterstützt Servicerollen: Nein

Eine Servicerolle ist eine [IAM-Rolle](#), die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter [Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service](#) im IAM-Benutzerhandbuch.

Warning

Das Ändern der Berechtigungen für eine Servicerolle kann dazu führen, dass die Funktionalität von Lookout for Vision beeinträchtigt wird. Bearbeiten Sie Servicerollen nur, wenn Lookout for Vision Sie dazu anleitet.

Servicebezogene Rollen für Lookout for Vision

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter [AWS -Services, die mit IAM funktionieren](#). Suchen Sie in der Tabelle nach einem Service mit einem Yes in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien von Amazon Lookout for Vision

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Lookout for Vision Vision-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe der AWS API, der AWS Management Console, der AWS Command Line Interface (AWS CLI) oder ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter [Erstellen von IAM-Richtlinien \(Konsole\)](#) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Lookout for Vision definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Lookout for Vision](#) in der Service Authorization Reference.

Themen

- [Bewährte Methoden für Richtlinien](#)
- [Zugreifen auf ein einzelnes Amazon Lookout for Vision Vision-Projekt](#)
- [Beispiel für eine tagbasierte Richtlinie](#)

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Lookout for Vision Vision-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr AWS-Konto verursachen. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem AWS-Konto verfügbar. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter [AWS -verwaltete Richtlinien](#) oder [AWS -verwaltete Richtlinien für Auftrags-Funktionen](#) im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter [Richtlinien und Berechtigungen in IAM](#) im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter [IAM-JSON-Richtlinienelemente: Bedingung](#) im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter [Richtlinienvvalidierung mit IAM Access Analyzer](#) im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter [Sicherer API-Zugriff mit MFA](#) im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter [Bewährte Methoden für die Sicherheit in IAM](#) im IAM-Benutzerhandbuch.

Zugreifen auf ein einzelnes Amazon Lookout for Vision Vision-Projekt

In diesem Beispiel möchten Sie einem Benutzer in Ihrem AWS Konto Zugriff auf eines Ihrer Amazon Lookout for Vision Vision-Projekte gewähren.

```
{
```



```
"Sid": "SpecificProjectOnly",
"Effect": "Allow",
"Action": [
    "lookoutvision:DetectAnomalies"
],
"Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

Beispiel für eine tagbasierte Richtlinie

Tag-basierte Richtlinien sind JSON-Richtliniendokumente, die die Aktionen spezifizieren, die ein Principal für markierte Ressourcen ausführen kann.

Verwenden Sie ein Tag, um auf eine Ressource zuzugreifen

Diese Beispielrichtlinie gewährt einem Benutzer oder einer Rolle in Ihrem AWS-Konto die Erlaubnis, den DetectAnomalies Vorgang mit jedem Modell zu verwenden, das mit dem Schlüssel `stage` und dem Wert gekennzeichnet ist `production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "LookoutVision:DetectAnomalies"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/stage": "production"
        }
      }
    }
  ]
}
```

Verwenden Sie ein Tag, um den Zugriff auf bestimmte Amazon Lookout for Vision Vision-Vorgänge zu verweigern

Diese Beispielrichtlinie verweigert einem Benutzer oder einer Rolle in Ihrem AWS-Konto die Erlaubnis, die `DeleteModel` `StopModel` OR-Operationen mit einem beliebigen Modell aufzurufen, das mit dem Schlüssel `stage` und dem Wert `production` gekennzeichnet ist.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "LookoutVision:DeleteModel",
        "LookoutVision:StopModel"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/stage": "production"
        }
      }
    }
  ]
}
```

AWS verwaltete Richtlinien für Amazon Lookout for Vision

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet AWS wird. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie [vom Kunden verwaltete Richtlinien](#) definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzipalidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter [Von AWS verwaltete Richtlinien](#) im IAM-Benutzerhandbuch.

Von AWS verwaltete Richtlinie: AmazonLookoutVisionReadOnlyAccess

Verwenden Sie die AmazonLookoutVisionReadOnlyAccess Richtlinie, um Benutzern mit den folgenden Amazon Lookout for Vision-Aktionen (SDK-Operationen) nur Lesezugriff auf Amazon Lookout for Vision (und seine Abhängigkeiten) zu gewähren. Sie können sie beispielsweise verwenden, `DescribeModel` um Informationen über ein vorhandenes Modell abzurufen.

- [DescribeDataset](#)
- [DescribeModel](#)
- [DescribeModelPackagingJob](#)
- [DescribeProject](#)
- [ListDatasetEntries](#)
- [ListModelPackagingJobs](#)
- [ListModels](#)
- [ListProjects](#)
- [ListTagsForResource](#)

Um schreibgeschützte Aktionen aufzurufen, benötigen Benutzer keine Amazon S3 S3-Bucket-Berechtigungen. Betriebsantworten können jedoch Verweise auf Amazon S3 S3-Buckets enthalten. Beispielsweise `ListDatasetEntries` ist der `source-ref` Eintrag in der Antwort von ein Verweis auf ein Bild in einem Amazon S3 S3-Bucket. Fügen Sie Amazon S3 S3-Bucket-Berechtigungen hinzu, wenn Ihre Benutzer auf referenzierte Buckets zugreifen müssen. Beispielsweise möchte ein Benutzer möglicherweise ein Bild herunterladen, auf das in einem `source-ref` Feld verwiesen wird. Weitere Informationen finden Sie unter [Erteilen von Amazon S3 S3-Bucket-Berechtigungen](#).

Sie können die `AmazonLookoutVisionReadOnlyAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

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

Von AWS verwaltete Richtlinie: `AmazonLookoutVisionFullAccess`

Verwenden Sie die `AmazonLookoutVisionFullAccess` Richtlinie, um Benutzern vollen Zugriff auf Amazon Lookout for Vision (und seine Abhängigkeiten) mit Amazon Lookout for Vision Vision-Aktionen (SDK-Operationen) zu gewähren. Sie können beispielsweise ein Modell trainieren, ohne die Amazon Lookout for Vision Vision-Konsole verwenden zu müssen. Weitere Informationen finden Sie unter [Aktionen](#).

Um einen Datensatz (`CreateDataset`) oder ein Modell (`CreateModel`) zu erstellen, benötigen Ihre Benutzer volle Zugriffsberechtigungen für den Amazon S3 S3-Bucket, in dem Datensatz-Bilder,

Amazon SageMaker AI Ground Truth-Manifestdateien und Trainingsausgaben gespeichert sind. Weitere Informationen finden Sie unter [Schritt 2: Berechtigungen einrichten](#).

Mithilfe der AmazonLookoutVisionConsoleFullAccess Richtlinie können Sie Amazon Lookout for Vision SDK-Aktionen auch die Genehmigung erteilen.

Sie können die AmazonLookoutVisionFullAccess-Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LookoutVisionFullAccess",
      "Effect": "Allow",
      "Action": [
        "lookoutvision:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Von AWS verwaltete Richtlinie: AmazonLookoutVisionConsoleFullAccess

Verwenden Sie die AmazonLookoutVisionFullAccess Richtlinie, um Benutzern vollen Zugriff auf die Amazon Lookout for Vision Vision-Konsole, Aktionen (SDK-Operationen) und alle Abhängigkeiten, die der Service hat, zu gewähren. Weitere Informationen finden Sie unter [Erste Schritte mit Amazon Lookout for Vision](#).

Die LookoutVisionConsoleFullAccess Richtlinie umfasst Berechtigungen für Ihren Amazon Lookout for Vision Vision-Konsolen-Bucket. Informationen zum Konsolen-Bucket finden Sie unter [Schritt 3: Erstellen Sie den Konsolen-Bucket](#). Um Datensätze, Bilder und Amazon SageMaker AI Ground Truth Manifestdateien in einem anderen Amazon S3 S3-Bucket zu speichern, benötigen Ihre Benutzer zusätzliche Berechtigungen. Weitere Informationen finden Sie unter [the section called "Amazon S3 S3-Bucket-Berechtigungen einrichten"](#).

Sie können die `AmazonLookoutVisionConsoleFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Berechtigungsgruppierungen

Diese Richtlinie ist auf der Grundlage der bereitgestellten Berechtigungen in Aussagen gruppiert:

- `LookoutVisionFullAccess`— Ermöglicht den Zugriff auf die Ausführung aller Lookout for Vision Vision-Aktionen.
- `LookoutVisionConsoleS3BucketSearchAccess`— Ermöglicht die Auflistung aller Amazon S3 S3-Buckets, die dem Anrufer gehören. Lookout for Vision verwendet diese Aktion, um den für die AWS-Region spezifischen Lookout for Vision Vision-Konsolen-Bucket zu identifizieren, falls einer im Konto des Anrufers vorhanden ist.
- `LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions`— Ermöglicht das Erstellen und Konfigurieren von Amazon S3 S3-Buckets, die dem Bucket-Namensmuster der Lookout for Vision Vision-Konsole entsprechen. Lookout for Vision verwendet diese Aktionen, um einen regionsspezifischen Lookout for Vision Vision-Konsolen-Bucket zu erstellen und zu konfigurieren, falls kein Bucket gefunden werden kann.
- `LookoutVisionConsoleS3BucketAccess`— Erlaubt abhängige Amazon S3 S3-Aktionen für Buckets, die dem Bucket-Namensmuster der Lookout for Vision Vision-Konsole entsprechen. Lookout for Vision verwendet `s3:ListBucket` die Suche nach Bildobjekten, wenn ein Datensatz aus einem Amazon S3 S3-Bucket erstellt wird und wenn eine Testerkennungsaufgabe gestartet wird. Lookout for Vision verwendet `s3:GetBucketLocation` und `s3:GetBucketVersioning` validiert die AWS Region, den Besitzer und die Konfiguration des Buckets im Rahmen der folgenden Aufgaben:
 - Einen Datensatz erstellen
 - Trainieren des Modells
 - Eine Aufgabe zur Erkennung von Studien wird gestartet
 - Durchführung von Rückmeldungen zur Erkennung von Studien

`LookoutVisionConsoleS3ObjectAccess`— Ermöglicht das Lesen und Schreiben von Amazon S3 S3-Objekten in Buckets, die dem Bucket-Namensmuster der Lookout for Vision Console entsprechen. Lookout for Vision verwendet diese Aktionen, um Bilder in Galerieansichten der Konsole anzuzeigen und neue Bilder zur Verwendung in Datensätzen hochzuladen. Darüber hinaus ermöglichen diese Berechtigungen Lookout for Vision, Metadaten zu schreiben, während

ein Datensatz erstellt, ein Modell trainiert, eine Aufgabe zur Erkennung von Studien gestartet und Feedback zur Studienerkennung durchgeführt wird.

- **LookoutVisionConsoleDatasetLabelingToolsAccess**— Erlaubt abhängige Amazon SageMaker GroundTruth AI-Labeling-Aktionen. Lookout for Vision verwendet diese Aktionen, um S3-Buckets nach Bildern zu durchsuchen, GroundTruth Manifestdateien zu erstellen und die Ergebnisse von Testerkennungsaufgaben mit Validierungsbeschriftungen zu versehen.
- **LookoutVisionConsoleDashboardAccess**- Ermöglicht das Lesen von CloudWatch Amazon-Metriken. Lookout for Vision verwendet diese Aktionen, um die Dashboard-Diagramme und Statistiken zu erkannten Anomalien zu füllen.
- **LookoutVisionConsoleTagSelectorAccess**— Ermöglicht das Lesen von kontospezifischen Vorschlägen für Tag-Schlüssel und Tag-Werte. Lookout for Vision verwendet diese Berechtigungen, um Empfehlungen für Tag-Schlüssel und Tag-Werte auf den Konsolenseiten „Tags verwalten“ bereitzustellen.
- **LookoutVisionConsoleKmsKeySelectorAccess**— Ermöglicht das Auflisten von Schlüsseln und Aliasen AWS Key Management Service (KMS). Amazon Lookout for Vision verwendet diese Berechtigung, um die KMS-Schlüssel in der vorgeschlagenen Tags-Auswahl für bestimmte Lookout for Vision Vision-Aktionen aufzufüllen, die vom Kunden verwaltete KMS-Schlüssel für die Verschlüsselung unterstützen.

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

```

Von AWS verwaltete Richtlinie: AmazonLookoutVisionConsoleReadOnlyAccess

Verwenden Sie die AmazonLookoutVisionConsoleReadOnlyAccess Richtlinie, um Benutzern nur Lesezugriff auf die Amazon Lookout for Vision Vision-Konsole, Aktionen (SDK-Operationen) und alle Abhängigkeiten, die der Service hat, zu gewähren.

Die AmazonLookoutVisionConsoleReadOnlyAccess Richtlinie umfasst Amazon S3 S3-Berechtigungen für den Amazon Lookout for Vision Vision-Konsolen-Bucket. Wenn sich Ihre Datensatz-Bilder oder Amazon SageMaker AI Ground Truth Manifestdateien in einem anderen Amazon S3 S3-Bucket befinden, benötigen Ihre Benutzer zusätzliche Berechtigungen. Weitere Informationen finden Sie unter [the section called “Amazon S3 S3-Bucket-Berechtigungen einrichten”](#).

Sie können die `AmazonLookoutVisionConsoleReadOnlyAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Berechtigungsgruppierungen

Diese Richtlinie ist auf der Grundlage der bereitgestellten Berechtigungen in Aussagen gruppiert:

- `LookoutVisionReadOnlyAccess`— Ermöglicht den Zugriff auf schreibgeschützte Lookout for Vision Vision-Aktionen.
- `LookoutVisionConsoleS3BucketSearchAccess`— Ermöglicht die Auflistung aller S3-Buckets, die dem Aufrufer gehören. Lookout for Vision verwendet diese Aktion, um den für die AWS-Region spezifischen Lookout for Vision Vision-Konsolen-Bucket zu identifizieren, falls es einen im Konto des Anrufers gibt.
- `LookoutVisionConsoleS3ObjectReadAccess`— Ermöglicht das Lesen von Amazon S3 S3-Objekten und Amazon S3 S3-Objektversionen in Konsolen-Buckets von Lookout for Vision. Lookout for Vision verwendet diese Aktionen, um die Bilder in Datensätzen, Modellen und Testerkennungen anzuzeigen.
- `LookoutVisionConsoleDashboardAccess`— Ermöglicht das Lesen von CloudWatch Amazon-Metriken. Lookout for Vision verwendet diese Aktionen, um Statistiken für Dashboard-Diagramme und erkannte Anomalien aufzufüllen.

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

```

Lookout for Vision Vision-Aktualisierungen für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Lookout for Vision an, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Wenn Sie automatische Benachrichtigungen über Änderungen an dieser Seite erhalten möchten, abonnieren Sie den RSS-Feed auf der Seite mit dem Verlauf der Lookout for Vision Vision-Dokumente.

hinzugefügt	packungsvorgänge zu den AmazonLookoutVisionReadOnlyAccessAmazonLookoutVisionConsoleFullAccess Richtlinien hinzugefügt: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision hat die folgenden Modellverpackungsvorgänge zu den AmazonLookoutVisionReadOnlyAccessAmazonLookoutVisionConsoleReadOnlyAccess Richtlinien hinzugefügt: • DescribeModelPackagingJob • ListModelPackagingJobs	
Neue Richtlinien wurden hinzugefügt	Amazon Lookout for Vision hat die folgenden Richtlinien hinzugefügt. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	11. Mai 2021
Lookout for Vision hat begonnen, Änderungen zu verfolgen	Amazon Lookout for Vision hat damit begonnen, Änderunge	1. März 2021

Änderung	Beschreibung	Datum
Operationen zum Verpacken von Modellen wurden hinzugefügt	Amazon Lookout for Vision hat die folgenden Modellverpackungsvorgänge zu den AmazonLookoutVisionFullAccessAmazonLookoutVisionConsoleFullAccess Richtlinien hinzugefügt: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision hat die folgenden Modellverpackungsvorgänge zu den AmazonLookoutVisionReadOnlyAccessAmazonLookoutVisionConsoleReadOnlyAccess Richtlinien hinzugefügt: • DescribeModelPackagingJob • ListModelPackagingJobs	7. Dezember 2021
Neue Richtlinien wurden hinzugefügt	Amazon Lookout for Vision hat die folgenden Richtlinien hinzugefügt. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess	11. Mai 2021
Von AWS verwaltete Richtlinien	• AmazonLookoutVisionConsoleReadOnlyAccess	

n an seinen AWS verwalteten

Problembhebung bei Identität und Zugriff auf Amazon Lookout for Vision

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Lookout for Vision und IAM auftreten können.

Themen

- [Ich bin nicht berechtigt, eine Aktion in Lookout for Vision durchzuführen](#)
- [Ich bin nicht berechtigt, iam auszuführen: PassRole](#)
- [Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Lookout for Vision Vision-Ressourcen ermöglichen](#)

Ich bin nicht berechtigt, eine Aktion in Lookout for Vision durchzuführen

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *my-example-widget*-Ressource anzuzeigen, jedoch nicht über `lookoutvision:GetWidget`-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
lookoutvision:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `lookoutvision:GetWidget`-Aktion auf die *my-example-widget*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Lookout for Vision übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servic Rolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Lookout for Vision auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Lookout for Vision Vision-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Lookout for Vision diese Funktionen unterstützt, finden Sie unter [So funktioniert Amazon Lookout for Vision mit IAM](#).
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter [Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto, den Sie besitzen](#).
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie [AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte](#).
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter [Gewähren von Zugriff für extern authentifizierte Benutzer \(Identitätsverbund\)](#) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Konformitätsprüfung für Amazon Lookout for Vision

Externe Prüfer bewerten die Sicherheit und Konformität von Amazon Lookout for Vision im Rahmen mehrerer AWS Compliance-Programme. Amazon Lookout for Vision entspricht der [Allgemeinen Datenschutzverordnung \(DSGVO\)](#).

Informationen darüber, ob AWS-Service ein in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter [AWS-Services Umfang nach Compliance-Programm AWS-Services unter](#). Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#).

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#).

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- [Compliance und Governance im Bereich Sicherheit](#) – In diesen Anleitungen für die Lösungsimplementierung werden Überlegungen zur Architektur behandelt. Außerdem werden Schritte für die Bereitstellung von Sicherheits- und Compliance-Features beschrieben.
- [Referenz für berechnete HIPAA-Services](#) – Listet berechnete HIPAA-Services auf. Nicht alle AWS-Services sind HIPAA-fähig.
- [AWS Compliance-Ressourcen](#) — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- [AWS Leitfäden zur Einhaltung von Vorschriften für Kunden](#) — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- [Evaluierung von Ressourcen anhand von Regeln](#) im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- [AWS Security Hub](#) — Dies AWS-Service bietet einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen

zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Die Liste der unterstützten Services und Kontrollen finden Sie in der [Security-Hub-Steuerelementreferenz](#).

- [Amazon GuardDuty](#) — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- [AWS Audit Manager](#) — Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Resilienz in Amazon Lookout for Vision

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Infrastruktursicherheit in Amazon Lookout for Vision

Als verwalteter Service ist Amazon Lookout for Vision durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Lookout for Vision zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS Security Token Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Überwachung von Amazon Lookout for Vision

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon Lookout for Vision und Ihren anderen AWS-Lösungen. AWS bietet die folgenden Überwachungstools, um Lookout for Vision zu beobachten, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen:

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarmer festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).
- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr robusten Speicher archivieren. Weitere Informationen finden Sie im [Amazon CloudWatch Logs-Benutzerhandbuch](#).
- Amazon EventBridge kann verwendet werden, um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse im AWS Rahmen von Services werden nahezu EventBridge in Echtzeit zugestellt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen ausgeführt werden sollen, wenn ein Ereignis mit einer Regel übereinstimmt. Weitere Informationen finden Sie im [EventBridge Amazon-Benutzerhandbuch](#).
- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im [AWS CloudTrail -Benutzerhandbuch](#).

Überwachung von Lookout for Vision mit Amazon CloudWatch

Sie können Lookout for Vision mithilfe von Lookout for Vision überwachen CloudWatch, das Rohdaten sammelt und zu lesbaren Metriken verarbeitet, die nahezu in Echtzeit verfügbar sind. Diese Statistiken werden 15 Monate gespeichert, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt werden. Sie können auch Alarme einrichten, die auf bestimmte Grenzwerte achten und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

Der Lookout for Vision Vision-Dienst meldet die folgenden Metriken im AWS/LookoutVision Namespace.

Metrik	Beschreibung
DetectedAnomalyCount	Die Anzahl der in einem Projekt festgestellten Anomalien Gültige Statistiken: Sum, Average Einheit: Anzahl
ProcessedImageCount	Die Gesamtzahl der Bilder, für die eine Anomalieerkennung durchlaufen wurde Gültige Statistiken: Sum, Average Einheit: Anzahl
InvalidImageCount	Die Anzahl der Bilder, die ungültig waren und kein Ergebnis zurückgeben konnten Gültige Statistiken: Sum, Average Einheit: Anzahl
SuccessfulRequestCount	Die Anzahl der erfolgreichen API-Aufrufe Gültige Statistiken: Sum, Average

Metrik	Beschreibung
	Einheit: Anzahl
ErrorCount	Die Anzahl der API-Fehler Gültige Statistiken: Sum, Average Einheit: Anzahl
ThrottledCount	Die Anzahl der API-Fehler, die auf Drosselung zurückzuführen waren Gültige Statistiken: Sum, Average Einheit: Anzahl
Time	Die Zeit in Millisekunden, die Lookout for Vision für die Berechnung der Anomalieerkennung benötigt Gültige Statistiken: Data Samples, Average Einheiten: Millisekunden für Average Statistiken und Anzahl für Statistiken Data Samples
MinInferenceUnits	Die Mindestanzahl von Inferenzeinheiten, die während der StartModel -Anforderung angegeben wurde. Gültige Statistiken: Average Einheit: Anzahl
MaxInferenceUnits	Die Maximalanzahl von Inferenzeinheiten, die während der StartModel -Anforderung angegeben wurde. Gültige Statistiken: Average Einheit: Anzahl

Metrik	Beschreibung
DesiredInferenceUnits	Die Anzahl der Inferenzeinheiten, auf die Lookout for Vision nach oben oder unten skaliert. Gültige Statistiken: Average Einheit: Anzahl
InServiceInferenceUnits	Die Anzahl der Inferenzeinheiten, die das Modell verwendet. Gültige Statistiken: Average Es wird empfohlen, die Durchschnittsstatistik zu verwenden, um den 1-Minuten-Durchschnitt der verwendeten Instances zu ermitteln. Einheit: Anzahl

Die folgenden Dimensionen werden für die Lookout for Vision Vision-Metriken unterstützt.

Dimension	Beschreibung
ProjectName	Sie können die Kennzahlen nach Projekten aufteilen , um zu sehen, bei welchen Projekten Probleme auftreten oder aktualisiert werden müssen.
ModelVersion	Sie können die Metriken nach Modellversionen aufteilen, um zu sehen, bei welchen Modellen Probleme auftreten oder aktualisiert werden müssen.

Protokollieren von Lookout for Vision API-Aufrufen mit AWS CloudTrail

Amazon Lookout for Vision ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen eines Benutzers, einer Rolle oder eines AWS Dienstes in Lookout for Vision bereitstellt.

CloudTrail erfasst alle API-Aufrufe für Lookout for Vision als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Lookout for Vision Vision-Konsole und Code-Aufrufe an die Lookout for Vision Vision-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Lookout for Vision. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Lookout for Vision gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im [AWS CloudTrail Benutzerhandbuch](#).

Halten Sie Ausschau nach Informationen zum Thema Vision in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Lookout for Vision eine Aktivität auftritt, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter [Ereignisse mit CloudTrail Ereignisverlauf anzeigen](#).

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für Lookout for Vision, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- [Übersicht zum Erstellen eines Trails](#)
- [CloudTrail Unterstützte Dienste und Integrationen](#)
- [Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail](#)
- [Empfangen von CloudTrail Protokolldateien aus mehreren Regionen](#) und [Empfangen von CloudTrail Protokolldateien von mehreren Konten](#)

Alle Lookout for Vision-Aktionen werden von der [Referenzdokumentation zur Lookout for Vision API](#) protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe

von `DetectAnomalies` und `StartModel` Aktionen Einträge in den CloudTrail Protokolldateien. `CreateProject`

Wenn Sie API-Aufrufe von Amazon Lookout for Vision überwachen, werden Ihnen möglicherweise Aufrufe zu den folgenden APIs Inhalten angezeigt.

- Lookoutvision: `StartTrialDetection`
- Lookoutvision: `ListTrialDetection`
- Lookoutvision: `DescribeTrialDetection`

Diese speziellen Aufrufe werden von Amazon Lookout for Vision verwendet, um verschiedene Vorgänge im Zusammenhang mit der Erkennung von Studien zu unterstützen. Weitere Informationen finden Sie unter [Verifizierung Ihres Modells mit einer Aufgabe zur Versuchserkennung](#).

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

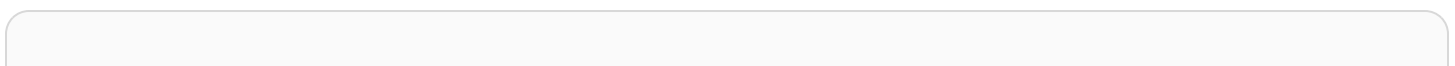
- Ob die Anfrage mit Root- oder AWS Identity and Access Management Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter [CloudTrail userIdentity-Element](#).

Grundlegendes zu Lookout for Vision Vision-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateDataset` Aktion demonstriert.




```
{
  "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"  
}
```

Erstellen von Amazon Lookout for Vision Vision-Ressourcen mit AWS CloudFormation

Amazon Lookout for Vision ist integriert AWS CloudFormation, ein Service, der Sie bei der Modellierung und Einrichtung Ihrer AWS Ressourcen unterstützt, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle gewünschten AWS Ressourcen beschreibt, und AWS CloudFormation kümmert sich um die Bereitstellung und Konfiguration dieser Ressourcen für Sie.

Sie können AWS CloudFormation Amazon Lookout for Vision Vision-Projekte bereitstellen und konfigurieren.

Wenn Sie sie verwenden AWS CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre Lookout for Vision Vision-Projekte konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Projekte einfach einmal und stellen Sie dann dieselben Projekte immer wieder in mehreren AWS-Konten und Regionen bereit.

Lookout for Vision und Vorlagen AWS CloudFormation

Um Projekte für Lookout for Vision und verwandte Dienste bereitzustellen und zu konfigurieren, müssen Sie [AWS CloudFormation Vorlagen](#) verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie AWS CloudFormation Designer verwenden, um Ihnen bei den ersten Schritten mit Vorlagen zu helfen. AWS CloudFormation Weitere Informationen finden Sie unter [Was ist AWS CloudFormation -Designer?](#) im AWS CloudFormation -Benutzerhandbuch.

Referenzinformationen zu Lookout for Vision Vision-Projekten, einschließlich Beispielen für JSON- und YAML-Vorlagen, finden Sie in der Referenz zum [LookoutVision Ressourcentyp](#).

Erfahren Sie mehr über AWS CloudFormation

Weitere Informationen AWS CloudFormation dazu finden Sie in den folgenden Ressourcen:

- [AWS CloudFormation](#)
- [AWS CloudFormation Benutzerhandbuch](#)

- [AWS CloudFormation API Reference](#)
- [AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle](#)

Greifen Sie über einen Schnittstellenendpunkt auf Amazon Lookout for Vision zu (AWS PrivateLink)

Sie können AWS PrivateLink verwenden, um eine private Verbindung zwischen Ihrer VPC und Amazon Lookout for Vision herzustellen. Sie können auf Lookout for Vision zugreifen, als wäre es in Ihrer VPC, ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder AWS Direct Connect eine Verbindung verwenden zu müssen. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um auf Lookout for Vision zuzugreifen.

Sie stellen diese private Verbindung her, indem Sie einen Schnittstellen-Endpunkt erstellen, der von AWS PrivateLink unterstützt wird. Wir erstellen eine Endpunkt-Netzwerkschnittstelle in jedem Subnetz, das Sie für den Schnittstellen-Endpunkt aktivieren. Dabei handelt es sich um vom Anforderer verwaltete Netzwerkschnittstellen, die als Einstiegspunkt für den Datenverkehr dienen, der für Lookout for Vision bestimmt ist.

Weitere Informationen finden Sie im Handbuch unter [Access AWS-Services through AWS PrivateLink](#). AWS PrivateLink

Überlegungen zu VPC-Endpunkten von Lookout for Vision

Bevor Sie einen Benutzeroberflächenendpunkt für Lookout for Vision einrichten, lesen Sie die [Überlegungen](#) im AWS PrivateLink Leitfaden.

Lookout for Vision unterstützt Aufrufe aller seiner API-Aktionen über den Schnittstellenendpunkt.

VPC-Endpunktrichtlinien werden für Lookout for Vision nicht unterstützt. Standardmäßig ist der vollständige Zugriff auf Lookout for Vision über den Endpunkt der Benutzeroberfläche zulässig. Alternativ können Sie den Endpunkt-Netzwerkschnittstellen eine Sicherheitsgruppe zuordnen, um den Datenverkehr zu Lookout for Vision über den Schnittstellenendpunkt zu steuern.

Erstellen eines VPC-Schnittstellen-Endpunkts für Lookout for Vision

Sie können einen Schnittstellenendpunkt für Lookout for Vision entweder mit der Amazon VPC-Konsole oder mit AWS Command Line Interface (AWS CLI) erstellen. Weitere Informationen finden Sie unter [Erstellen eines Schnittstellenendpunkts](#) im AWS PrivateLink -Leitfaden.

Erstellen Sie einen Schnittstellenendpunkt für Lookout for Vision mit dem folgenden Dienstnamen:

```
com.amazonaws.region.lookoutvision
```

Wenn Sie privates DNS für den Schnittstellenendpunkt aktivieren, können Sie API-Anfragen an Lookout for Vision stellen, indem Sie den standardmäßigen regionalen DNS-Namen verwenden. Beispiel, `lookoutvision.us-east-1.amazonaws.com`.

Erstellen einer VPC-Endpunktrichtlinie für Lookout for Vision

Eine Endpunktrichtlinie ist eine IAM-Ressource, die Sie an einen Schnittstellen-Endpunkt anfügen können. Die standardmäßige Endpunktrichtlinie ermöglicht den vollen Zugriff auf Lookout for Vision über den Endpunkt der Benutzeroberfläche. Um den Zugriff auf Lookout for Vision von Ihrer VPC aus zu kontrollieren, fügen Sie dem Schnittstellenendpunkt eine benutzerdefinierte Endpunktrichtlinie hinzu.

Eine Endpunktrichtlinie gibt die folgenden Informationen an:

- Die Prinzipale, die Aktionen ausführen können (AWS-Konten, IAM-Benutzer und IAM-Rollen).
- Aktionen, die ausgeführt werden können
- Die Ressourcen, auf denen die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter [Steuern des Zugriffs auf Services mit Endpunktrichtlinien](#) im AWS PrivateLink -Leitfaden.

Beispiel: VPC-Endpunktrichtlinie für Lookout for Vision Vision-Aktionen

Im Folgenden finden Sie ein Beispiel für eine benutzerdefinierte Endpunktrichtlinie für Lookout for Vision. Wenn Sie diese Richtlinie an Ihren Schnittstellenendpunkt anhängen, gibt sie an, dass alle Benutzer, die Zugriff auf den VPC-Schnittstellenendpunkt haben, den `DetectAnomalies` API-Vorgang für das Lookout for Vision Vision-Modell aufrufen dürfen, das dem Projekt `myModel` zugeordnet ist. `myProject`

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "lookoutvision:DetectAnomalies"
      ]
    }
  ]
}
```

```
    ],  
    "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/  
myModel"  
  }  
]  
}
```

Kontingente in Amazon Lookout for Vision

In den folgenden Tabellen werden die aktuellen Kontingente in Amazon Lookout for Vision beschrieben. Informationen zu Kontingenten, die geändert werden können, finden Sie unter [AWS-Servicekontingente](#).

Modellkontingente

Die folgenden Kontingente gelten für Tests, Schulungen und Funktionen eines Modells.

Ressource	Kontingent
Unterstütztes Dateiformat	PNG- und JPEG-Bildformate
Minimale Bildgröße der Bilddatei in einem Amazon S3 S3-Bucket	64 Pixel x 64 Pixel
Maximale Bildgröße der Bilddatei in einem Amazon S3 S3-Bucket	4096 Pixel x 4096 Pixel ist das Maximum. Kleinere Abmessungen können schneller hochgeladen werden.
Unterschiedliche Bildabmessungen der in einem Projekt verwendeten Bilddateien	Alle Bilder im Datensatz müssen dieselben Abmessungen haben
Maximale Dateigröße für ein Bild in einem Amazon S3 S3-Bucket	8 MB
Fehlende Beschriftungen	Bilder müssen vor dem Training als normal oder ungewöhnlich gekennzeichnet werden. Bilder ohne Beschriftungen werden während des Trainings ignoriert.
Mindestanzahl von Bildern, die im Trainingsdatensatz als normal gekennzeichnet sind	10 für ein Projekt mit separaten Trainings- und Testdatensätzen. 20 für ein Projekt mit einem einzigen Datensatz.

Ressource	Kontingent
Mindestanzahl von Bildern mit der Bezeichnung „Anomalie“ in einem Trainingsdatensatz	0 für ein Projekt mit separaten Trainings- und Testdatensätzen. 10 für ein Projekt mit einem einzigen Datensatz.
Maximale Anzahl von Bildern im Datensatz für das Klassifizierungstraining	16,000
Maximale Anzahl von Bildern in einem Klassifizierungstest-Datensatz	4.000
Mindestanzahl von Bildern, die im Testdatensatz als normal gekennzeichnet sind	10
Mindestanzahl von Bildern, die im Testdatensatz als Anomalie gekennzeichnet sind	10
Maximale Anzahl von Bildern in einem Trainingsdatensatz zur Anomalie-Lokalisierung	8000
Maximale Anzahl von Bildern in einem Testdatensatz zur Anomalie-Lokalisierung	800
Maximale Anzahl von Bildern im Datensatz zur Erkennung von Studien	2.000
Maximale Größe der Datensatz-Manifestdatei	1 GB
Maximale Anzahl von Trainingsdatensätzen in einem Modell	1
Maximale Trainingszeit	24 Stunden
Maximale Testzeit	24 Stunden
Maximale Anzahl von Anomalie-Labels in einem Projekt	100

Ressource	Kontingent
Maximale Anzahl von Anomalie-Labels auf einem Maskenbild	20
Mindestanzahl von Bildern für ein Anomalie-Label. Um zählen zu können, darf das Bild nur einen Typ von Anomaliekennzeichnung enthalten.	20 für ein einzelnes Datensatzprojekt. 10 für jeden Datensatz in einem Projekt mit separaten Trainings- und Testdatensätzen.

Dokumentenverlauf für Amazon Lookout for Vision

In der folgenden Tabelle werden wichtige Änderungen in den einzelnen Versionen des Amazon Lookout for Vision Developer Guide beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

- Letzte Aktualisierung der Dokumentation: 20. Februar 2023

Änderung	Beschreibung	Datum
Ende der Support-Mitteilung	Hinweis zum Ende des Supports: Am 31. Oktober 2025 AWS wird der Support für Amazon Lookout for Vision eingestellt. Nach dem 31. Oktober 2025 können Sie nicht mehr auf die Lookout for Vision Vision-Konsole oder die Lookout for Vision Vision-Ressourcen zugreifen. Weitere Informationen finden Sie in diesem Blogbeitrag .	10. Oktober 2024
Beispiel für eine Lambda-Funktion hinzugefügt	Beispiel, das zeigt, wie man mit einer Funktion Anomalien findet. AWS Lambda Weitere Informationen finden Sie unter Auffinden von Anomalien mit einer AWS-Lambda-Funktion .	20. Februar 2023
Die IAM-Leitlinien für wurden aktualisiert AWS WAF	Aktualisierung des Leitfadens zur Ausrichtung an bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte IAM-Methoden .	8. Februar 2023

[Beispiel für den Export eines Datensatzes hinzugefügt](#)

Es wurde ein Python-Beispiel hinzugefügt, das zeigt, wie der `ListDatasetEntries` Vorgang zum Exportieren der Datensätze aus einem Amazon Lookout for Vision Vision-Projekt verwendet wird. Weitere Informationen finden Sie unter [Exportieren von Datensätzen aus einem Projekt \(SDK\)](#).

2. Dezember 2022

[Das Thema „Erste Schritte“ wurde aktualisiert](#)

Die ersten Schritte wurden aktualisiert und zeigen nun, wie ein Bildsegmentierungsmodell mit einem Beispieldatensatz erstellt wird. Weitere Informationen finden Sie unter [Erste Schritte mit Amazon Lookout for Vision](#).

20. Oktober 2022

[Thema zur Fehlerbehebung hinzugefügt](#)

Thema zur Problembehandlung beim Modelltraining hinzugefügt. Weitere Informationen finden Sie unter [Problembehandlung beim Modelltraining](#).

17. Oktober 2022

[Thema zur Verwendung von Amazon SageMaker AI Ground Truth Jobs hinzugefügt](#)

Anstatt Bilder selbst zu kennzeichnen, können Sie Amazon SageMaker AI Ground Truth Jobs verwenden, um Bilder für Klassifizierungs- und Bildsegmentierungsmodelle zu kennzeichnen. Weitere Informationen finden Sie unter [Verwenden eines Amazon SageMaker AI Ground Truth Jobs](#).

19. August 2022

[Amazon Lookout for Vision bietet jetzt die Lokalisierung von Anomalien.](#)

Sie können ein Segmentierungsmodell erstellen, das die Stellen auf einem Bild ermittelt, an denen verschiedene Arten von Anomalien (z. B. Kratzer, Dellen oder Risse) vorhanden sind, die Bezeichnung der Anomalie und die Größe der Anomalie. Weitere Informationen finden Sie unter [Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells](#).

16. August 2022

[Amazon Lookout for Vision bietet jetzt CPU-Inferenz auf Edge-Geräten.](#)

Amazon Lookout for Vision Vision-Modelle können jetzt bereitgestellt werden, um Inferenzen lokal auf einer x86-Rechenplattform unter Linux mit nur einer CPU auszuführen, ohne dass ein GPU-Beschleuniger erforderlich ist. [Weitere Informationen finden Sie unter CPU-Beschleuniger.](#)

16. August 2022

[Amazon Lookout for Vision kann jetzt automatisch Inferenzeinheiten skalieren.](#)

Um bei Nachfragespitzen zu helfen, kann Amazon Lookout for Vision jetzt die Anzahl der Inferenzeinheiten skalieren, die Ihr Modell verwendet. Weitere Informationen finden Sie unter [Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells.](#)

16. August 2022

[Java-Beispiele hinzugefügt](#)

Das Amazon Lookout for Vision Vision-Entwicklerhandbuch enthält jetzt Java-Beispiele. Weitere Informationen finden Sie unter [Erste Schritte mit dem AWS SDK.](#)

2. Mai 2022

[Allgemeine Verfügbarkeit der Modellbereitstellung auf einem Edge-Gerät](#)

Die Modellbereitstellung auf einem Edge-Gerät, das von verwaltet AWS IoT Greengrass Version 2 wird, ist jetzt allgemein verfügbar. Weitere Informationen finden Sie unter [Verwenden Ihres Amazon Lookout for Vision Vision-Modells auf einem Edge-Gerät.](#)

14. März 2022

[Die Informationen zum Konsolen-Bucket wurden aktualisiert](#)

Aktualisierte Informationen zum Inhalt des Konsolen-Buckets und zu alternativen Ansätzen zur Erstellung des Konsolen-Buckets. Weitere Informationen finden Sie unter [Schritt 4: Den Konsolen-Bucket erstellen.](#)

7. März 2022

[Erstellen einer Manifestdatei aus einer CSV-Datei](#)

Sie können jetzt die Erstellung einer Manifestdatei vereinfachen, indem Sie ein Skript verwenden, das Klassifizierungsinformationen aus einer CSV-Datei liest. Weitere Informationen finden Sie unter [Erstellen einer Manifestdatei aus einer CSV-Datei](#).

10. Februar 2022

[Vorabversion der Modellbereitstellung auf einem Edge-Gerät](#)

Die Vorabversion der Modellbereitstellung auf einem Edge-Gerät, das von verwaltet wird, AWS IoT Greengrass Version 2 ist jetzt verfügbar. Weitere Informationen finden Sie unter [Verwenden Ihres Amazon Lookout for Vision Vision-Modells auf einem Edge-Gerät](#).

7. Dezember 2021

[Neue Python- und Java 2-Beispiele hinzugefügt](#)

Python- und Java 2-Beispiele für die Analyse von Bildern mit hinzugefügt `DetectAnomalies`. Weitere Informationen finden Sie unter [Erkennen von Anomalien in einem Bild](#).

7. September 2021

[Neue AWS verwaltete Richtlinien wurden hinzugefügt.](#)

Amazon Lookout for Vision bietet Unterstützung für AWS verwaltete Richtlinien. Weitere Informationen finden Sie unter [AWS Verwaltete Richtlinien für Amazon Lookout for Vision](#).

11. Mai 2021

[Die Informationen zur Inferenzeinheit wurden aktualisiert.](#)

Es wurden Informationen hinzugefügt, die beschreiben, wie Inferenzeinheiten berechnet werden. Weitere Informationen finden Sie unter [Ausführen Ihres trainierten Amazon Lookout for Vision Vision-Modells](#).

15. März 2021

[Allgemeine Verfügbarkeit für Amazon Lookout for Vision.](#)

Amazon Lookout for Vision ist jetzt allgemein verfügbar. Python-Codebeispiele wurden aktualisiert, um asynchrone Aufgaben wie das [Trainieren eines Modells](#) zu bewältigen.

17. Februar 2021

[Tagging und AWS CloudFormation Unterstützung hinzugefügt.](#)

Sie können jetzt Amazon Lookout for Vision Vision-Modelle taggen und Projekte damit AWS CloudFormation erstellen. Weitere Informationen finden Sie unter Kennzeichnen [von Modellen](#) und [Erstellen von Amazon Lookout for Vision Vision-Projekten mit AWS](#). CloudFormation

31. Januar 2021

[Neue Funktion und Anleitung](#)

Dies ist die erste Version des Amazon Lookout for Vision-Dienstes Amazon Lookout for Vision Developer Guide.

1. Dezember 2020

AWS Glossar

Die neueste AWS Terminologie finden Sie im [AWS Glossar](#) in der AWS-Glossar Referenz.