



Panduan Label Kustom

# Rekognition



# Rekognition: Panduan Label Kustom

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

---

# Table of Contents

- Apa itu Label Kustom Rekognition Amazon? ..... 1
  - Manfaat utama ..... 2
  - Memilih untuk menggunakan Label Kustom Amazon Rekognition ..... 2
    - Amazon Rekognition Deteksi label gambar ..... 3
    - Amazon Rekognition Custom Labels ..... 3
  - Apakah Anda pengguna Label Kustom Amazon Rekognition pertama kali? ..... 4
- Menyiapkan Label Kustom Rekognition Amazon ..... 5
  - Langkah 1: Buat AWS akun ..... 5
    - Mendaftar untuk Akun AWS ..... 6
    - Akses terprogram ..... 6
  - Langkah 2: Siapkan izin konsol ..... 8
    - Mengizinkan akses konsol ..... 9
    - Mengakses Bucket Amazon S3 eksternal ..... 10
    - Menetapkan izin ..... 11
  - Langkah 3: Buat bucket konsol ..... 11
  - Langkah 4: Mengatur AWS CLI and AWS SDK ..... 12
    - Instal AWS SDK ..... 13
    - Memberikan akses programatis ..... 6
    - Siapkan izin SDK ..... 17
    - Panggil operasi ..... 19
  - Langkah 5: (Opsional) Enkripsi file pelatihan ..... 23
    - Mendekripsi file yang dienkripsi dengan AWS Key Management Service ..... 23
    - Mengkripsi gambar pelatihan dan uji yang disalin ..... 24
  - Langkah 6: (Opsional) Kaitkan kumpulan data sebelumnya ..... 24
    - Menggunakan dataset sebelumnya sebagai dataset pengujian ..... 25
- Memahami Label Kustom Rekognition Amazon ..... 27
  - Tentukan jenis model Anda ..... 27
    - Temukan objek, adegan, dan konsep ..... 28
    - Temukan lokasi objek ..... 29
    - Temukan lokasi merek ..... 29
  - Buat model ..... 30
    - Membuat proyek ..... 30
    - Buat kumpulan data pelatihan dan uji ..... 31
    - Latih model Anda ..... 32

Tingkatkan model Anda .....	33
Evaluasi model Anda .....	33
Tingkatkan model Anda .....	34
Mulai model Anda .....	34
Mulai model Anda (konsol) .....	34
Mulai model Anda .....	34
Menganalisis gambar .....	35
Hentikan model Anda .....	36
Hentikan model Anda (Konsol) .....	36
Hentikan model Anda (SDK) .....	36
Memulai .....	37
Video tutorial .....	37
Contoh proyek .....	38
Klasifikasi gambar .....	38
Klasifikasi gambar multi-label .....	38
Deteksi merek .....	39
Lokalisasi objek .....	39
Menggunakan contoh proyek .....	40
Membuat proyek contoh .....	40
Melatih model .....	41
Menggunakan model .....	41
Langkah selanjutnya .....	41
Langkah 1: Pilih contoh proyek .....	41
Langkah 2: Latih model Anda .....	44
Langkah 3: Mulai model Anda .....	47
Langkah 4: Analisis gambar dengan model Anda .....	48
Mendapatkan contoh gambar .....	53
Langkah 5: Hentikan model Anda .....	54
Langkah 6: Langkah selanjutnya .....	56
Mengklasifikasikan gambar .....	58
Langkah 1: Kumpulkan gambar Anda .....	58
Langkah 2: Tentukan kelas Anda .....	59
Langkah 3: Buat proyek .....	60
Langkah 4: Buat kumpulan data pelatihan dan uji .....	61
Langkah 5: Tambahkan label ke proyek .....	66
Langkah 6: Tetapkan label tingkat gambar ke kumpulan data pelatihan dan uji .....	66

Langkah 7: Latih model Anda .....	68
Langkah 8: Mulai model Anda .....	71
Langkah 9: Analisis gambar dengan model Anda .....	72
Langkah 10: Hentikan model Anda .....	75
Membuat model .....	78
Membuat proyek .....	78
Membuat Proyek (Konsol) .....	78
Membuat proyek (SDK) .....	79
Buat format permintaan proyek .....	84
Membuat set data .....	84
Mengarahkan kumpulan data .....	85
Mempersiapkan gambar .....	90
Membuat dataset dengan gambar .....	92
Pelabelan gambar .....	153
Debugging kumpulan data .....	162
Melatih model .....	169
Melatih model (Konsol) .....	171
Melatih model (SDK) .....	174
Pelatihan model debugging .....	185
Kesalahan terminal .....	185
Daftar kesalahan validasi baris JSON non-terminal .....	187
Memahami ringkasan manifes .....	189
Memahami manifestasi hasil validasi pelatihan dan pengujian .....	192
Mendapatkan hasil validasi .....	198
Memperbaiki kesalahan pelatihan .....	201
Kesalahan file manifes terminal .....	202
Kesalahan konten manifes terminal .....	204
Kesalahan Validasi Jalur JSON Non-Terminal .....	214
Meningkatkan model yang terlatih .....	238
Metrik untuk mengevaluasi model Anda .....	238
Mengevaluasi kinerja model .....	239
Ambang batas yang diasumsikan .....	240
presisi .....	240
Ingat .....	241
F1 .....	241
Menggunakan metrik .....	242

Mengakses metrik evaluasi (Konsol) .....	242
Mengakses metrik evaluasi (SDK) .....	245
Mengakses file ringkasan model .....	246
Menafsirkan snapshot manifes evaluasi .....	247
Mengakses file ringkasan dan snapshot manifes evaluasi (SDK) .....	251
Melihat matriks kebingungan untuk model .....	252
Referensi: File Ringkasan .....	259
Meningkatkan model .....	261
Data .....	262
Mengurangi positif palsu (presisi yang lebih baik) .....	262
Mengurangi negatif palsu (mengingat lebih baik) .....	263
Menjalankan model terlatih .....	264
Unit inferensi .....	264
Mengelola throughput dengan unit inferensi .....	265
Zona Ketersediaan .....	267
Memulai model .....	267
Memulai atau menghentikan model (Konsol) .....	268
Memulai model (SDK) .....	269
Menghentikan model .....	279
Menghentikan model (Konsol) .....	279
Menghentikan model (SDK) .....	280
Durasi pelaporan dan unit inferensi .....	288
Menganalisis gambar dengan model terlatih .....	292
DetectCustomLabels permintaan operasi .....	318
DetectCustomLabels respon operasi .....	319
Mengelola sumber daya .....	320
Mengelola proyek .....	320
Menghapus proyek .....	321
Menjelaskan proyek (SDK) .....	331
Membuat proyek dengan AWS CloudFormation .....	337
Mengelola kumpulan data .....	338
Menambahkan dataset .....	339
Menambahkan lebih banyak gambar .....	348
Membuat kumpulan data menggunakan dataset (SDK) yang ada .....	358
Menjelaskan kumpulan data (SDK) .....	367
Daftar entri kumpulan data (SDK) .....	372

Mendistribusikan dataset pelatihan (SDK) .....	378
Menghapus dataset .....	388
Mengelola model .....	395
Menghapus model .....	396
Menandai model .....	405
Menjelaskan model (SDK) .....	412
Menyalin model (SDK) .....	420
Contoh Label Kustom .....	457
Meningkatkan model dengan umpan balik Model .....	457
Demonstrasi Label Kustom Rekognition Amazon .....	458
Mendeteksi Label Kustom dalam video .....	458
Menganalisis gambar dengan suatu AWS Lambda fungsi .....	461
Langkah 1: Buat AWS Lambda fungsi (konsol) .....	462
Langkah 2: (Opsional) Buat layer (konsol) .....	464
Langkah 3: Tambahkan kode Python (konsol) .....	465
Langkah 4: Coba fungsi Lambda Anda .....	468
Keamanan .....	473
Mengamankan proyek Label Kustom Rekognition Amazon .....	473
Mengamankan DetectCustomLabels .....	475
Kebijakan terkelola AWS .....	475
Pedoman dan kuota .....	476
Wilayah yang Didukung .....	476
Kuota .....	476
Pelatihan .....	476
Pengujian .....	477
Deteksi .....	478
Penyalinan model .....	478
Referensi API .....	479
Melatih model Anda .....	488
Proyek .....	488
Kebijakan Proyek .....	488
Set Data .....	489
Model .....	489
Tanda .....	489
Menggunakan model Anda .....	489
Riwayat dokumen .....	490

---

..... cdxvi

# Apa itu Label Kustom Rekognition Amazon?

Dengan Label Kustom Amazon Rekognition, Anda dapat mengidentifikasi objek, logo, dan adegan dalam gambar yang spesifik untuk kebutuhan bisnis Anda. Misalnya, Anda dapat menemukan logo Anda di posting media sosial, mengidentifikasi produk Anda di rak-rak toko, mengklasifikasikan suku cadang mesin di jalur perakitan, membedakan tanaman yang sehat dan terinfeksi, atau mendeteksi karakter animasi dalam gambar.

Mengembangkan model khusus untuk menganalisis gambar adalah usaha penting yang membutuhkan waktu, keahlian, dan sumber daya. Seringkali butuh waktu berbulan-bulan untuk menyelesaikannya. Selain itu, dapat memerlukan ribuan atau puluhan ribu gambar berlabel tangan untuk menyediakan model dengan data yang cukup untuk membuat keputusan secara akurat. Menghasilkan data ini dapat memakan waktu berbulan-bulan untuk dikumpulkan, dan dapat membutuhkan tim besar pelabel untuk mempersiapkannya untuk digunakan dalam pembelajaran mesin.

Amazon Rekognition Custom Labels memperluas kemampuan Amazon Rekognition yang ada, yang sudah dilatih pada puluhan juta gambar di banyak kategori. Alih-alih ribuan gambar, Anda dapat mengunggah satu set kecil gambar pelatihan (biasanya beberapa ratus gambar atau kurang) yang khusus untuk kasus penggunaan Anda. Anda dapat melakukan ini dengan menggunakan easy-to-use konsol. Jika gambar Anda sudah diberi label, Amazon Rekognition Custom Labels dapat mulai melatih model dalam waktu singkat. Jika tidak, Anda dapat memberi label gambar secara langsung di dalam antarmuka pelabelan, atau Anda dapat menggunakan Amazon SageMaker AI Ground Truth untuk memberi label untuk Anda.

Setelah Amazon Rekognition Custom Labels memulai pelatihan dari set gambar Anda, itu dapat menghasilkan model analisis gambar khusus untuk Anda hanya dalam beberapa jam. Di balik layar, Amazon Rekognition Custom Labels secara otomatis memuat dan memeriksa data pelatihan, memilih algoritma pembelajaran mesin yang tepat, melatih model, dan menyediakan metrik kinerja model. Anda kemudian dapat menggunakan model kustom Anda melalui Amazon Rekognition Custom Labels API dan mengintegrasikannya ke dalam aplikasi Anda.

## Topik

- [Manfaat utama](#)
- [Memilih untuk menggunakan Label Kustom Amazon Rekognition](#)
- [Apakah Anda pengguna Label Kustom Amazon Rekognition pertama kali?](#)

## Manfaat utama

### Pelabelan data yang disederhanakan

Konsol Amazon Rekognition Custom Labels menyediakan antarmuka visual untuk membuat pelabelan gambar Anda cepat dan sederhana. Antarmuka memungkinkan Anda untuk menerapkan label ke seluruh gambar. Anda juga dapat mengidentifikasi dan memberi label objek tertentu dalam gambar menggunakan kotak pembatas dengan click-and-drag antarmuka. Sebagai alternatif, jika Anda memiliki kumpulan data yang besar, Anda dapat menggunakan Amazon [SageMakerGround Truth](#) untuk memberi label pada gambar Anda secara efisien dalam skala besar.

### Pembelajaran mesin otomatis

Tidak diperlukan keahlian pembelajaran mesin untuk membangun model kustom Anda. Amazon Rekognition Custom Labels mencakup kemampuan pembelajaran mesin otomatis (AutoML) yang menangani pembelajaran mesin untuk Anda. Saat gambar pelatihan disediakan, Label Kustom Rekognition Amazon dapat memuat dan memeriksa data secara otomatis, memilih algoritme pembelajaran mesin yang tepat, melatih model, dan menyediakan metrik kinerja model.

### Evaluasi model, inferensi, dan umpan balik yang disederhanakan

Anda mengevaluasi kinerja model kustom Anda pada set pengujian Anda. Untuk setiap gambar dalam set pengujian, Anda dapat melihat side-by-side perbandingan prediksi model vs. label yang ditetapkan. Anda juga dapat meninjau metrik kinerja terperinci seperti presisi, ingatan, skor F1, dan skor kepercayaan diri. Anda dapat mulai menggunakan model Anda segera untuk analisis gambar, atau Anda dapat mengulangi dan melatih ulang versi baru dengan lebih banyak gambar untuk meningkatkan kinerja. Setelah Anda mulai menggunakan model Anda, Anda melacak prediksi Anda, memperbaiki kesalahan, dan menggunakan data umpan balik untuk melatih kembali versi model baru dan meningkatkan kinerja.

## Memilih untuk menggunakan Label Kustom Amazon Rekognition

Amazon Rekognition menyediakan dua fitur yang dapat Anda gunakan untuk menemukan label (objek, adegan, dan konsep) dalam gambar: Amazon Rekognition Custom Labels [dan](#) Amazon Rekognition Image label detection. Gunakan informasi berikut untuk menentukan fitur mana yang harus Anda gunakan.

## Amazon Rekognition Deteksi label gambar

Anda dapat menggunakan fitur deteksi label di Amazon Rekognition Image untuk mengidentifikasi, mengklasifikasikan, dan mencari label umum dalam gambar dan video—dalam skala besar dan tanpa harus membuat model pembelajaran mesin. Misalnya, Anda dapat dengan mudah mendeteksi ribuan objek umum, seperti mobil dan truk, tomat, bola basket, dan bola sepak.

Jika aplikasi Anda perlu menemukan label umum, sebaiknya gunakan deteksi label Gambar Rekognition Amazon, karena Anda tidak perlu melatih model. [Untuk mendapatkan daftar label yang ditemukan oleh deteksi label Gambar Rekognition Amazon, lihat Mendeteksi label.](#)

Jika aplikasi Anda perlu menemukan label yang tidak ditemukan oleh deteksi label Gambar Rekognition Amazon, seperti suku cadang mesin khusus pada jalur perakitan, sebaiknya gunakan Label Kustom Amazon Rekognition.

## Amazon Rekognition Custom Labels

Anda dapat menggunakan Amazon Rekognition Custom Labels untuk dengan mudah melatih model pembelajaran mesin yang menemukan label (objek, logo, adegan, dan konsep) dalam gambar yang unik untuk kebutuhan bisnis Anda.

Amazon Rekognition Custom Labels dapat mengklasifikasikan gambar (prediksi tingkat gambar) atau mendeteksi lokasi objek dalam gambar (prediksi level objek/kotak pembatas).

Amazon Rekognition Custom Labels memberikan fleksibilitas yang lebih besar dalam jenis objek dan adegan yang dapat Anda deteksi. Misalnya, Anda dapat menggunakan deteksi label Gambar Rekognition Amazon untuk menemukan tanaman dan daun. Untuk membedakan antara tanaman yang sehat, rusak, dan terinfeksi, Anda perlu menggunakan Label Kustom Rekognition Amazon.

Berikut ini adalah contoh bagaimana Anda dapat menggunakan Amazon Rekognition Custom Labels.

- Identifikasi logo tim pada kaus dan helm pemain
- Bedakan antara suku cadang atau produk mesin tertentu pada jalur perakitan
- Identifikasi karakter kartun di perpustakaan media
- Temukan produk dari merek tertentu di rak ritel
- Klasifikasi kualitas hasil pertanian (seperti busuk, matang, atau mentah)

**Note**

Amazon Rekognition Custom Labels tidak dirancang untuk menganalisis wajah, mendeteksi teks, atau menemukan konten gambar yang tidak aman dalam gambar. Untuk melakukan tugas-tugas ini, Anda dapat menggunakan Amazon Rekognition Image. Untuk informasi selengkapnya, lihat [Apa itu Rekognition Amazon](#).

## Apakah Anda pengguna Label Kustom Amazon Rekognition pertama kali?

Jika Anda adalah pengguna pertama kali Amazon Rekognition Custom Labels, kami sarankan Anda membaca bagian berikut secara berurutan:

1. [Menyiapkan Label Kustom Rekognition Amazon](#)— Di bagian ini, Anda mengatur detail akun Anda.
2. [Memahami Label Kustom Rekognition Amazon](#)— Di bagian ini, Anda belajar tentang alur kerja untuk membuat model.
3. [Memulai Label Kustom Amazon Rekognition](#)— Di bagian ini, Anda melatih model menggunakan contoh proyek yang dibuat oleh Amazon Rekognition Custom Labels.
4. [Mengklasifikasikan gambar](#)— Di bagian ini, Anda mempelajari cara melatih model yang mengklasifikasikan gambar dengan kumpulan data yang Anda buat.

# Menyiapkan Label Kustom Rekognition Amazon

Petunjuk berikut menunjukkan cara mengatur konsol Amazon Rekognition Custom Labels dan SDK.

Perhatikan bahwa Anda dapat menggunakan konsol Label Kustom Amazon Rekognition dengan browser berikut:

- Chrome - Versi 21 atau yang lebih baru
- Firefox — Versi 27 atau yang lebih baru
- Microsoft Edge - Versi 88 atau yang lebih baru
- Safari — Versi 7 atau yang lebih baru. Selain itu, Anda tidak dapat menggunakan Safari untuk menggambar kotak pembatas dengan konsol Amazon Rekognition Custom Labels. Untuk informasi selengkapnya, lihat [Pelabelan objek dengan kotak pembatas](#).

Sebelum Anda menggunakan Label Kustom Amazon Rekognition untuk pertama kalinya, selesaikan tugas-tugas berikut:

Topik

- [Langkah 1: Buat AWS akun](#)
- [Langkah 2: Siapkan izin konsol Amazon Rekognition Custom Labels](#)
- [Langkah 3: Buat bucket konsol](#)
- [Langkah 4: Mengatur AWS CLI and AWS SDK](#)
- [Langkah 5: \(Opsional\) Enkripsi file pelatihan](#)
- [Langkah 6: \(Opsional\) Kaitkan kumpulan data sebelumnya dengan proyek baru](#)

## Langkah 1: Buat AWS akun

Pada langkah ini, Anda membuat AWS akun, membuat pengguna administratif, dan mempelajari tentang pemberian akses terprogram ke SDK AWS .

Topik

- [Mendaftar untuk Akun AWS](#)
- [Akses terprogram](#)

## Mendaftar untuk Akun AWS

Untuk memulai AWS, Anda membutuhkan Akun AWS. Untuk informasi tentang membuat Akun AWS, lihat [Memulai dengan Akun AWS](#) di Panduan AWS Account Management Referensi.

### Akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. Konsol Manajemen AWS Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Disarankan) Gunakan kredensial konsol sebagai kredensial sementara untuk menandatangani permintaan terprogram ke, SDK AWS CLI, AWS atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk itu AWS CLI, lihat <a href="#">Login untuk pengembangan AWS lokal</a> di Panduan AWS Command Line Interface Pengguna.</li> <li>• Untuk AWS SDK, lihat <a href="#">Login untuk pengembangan AWS lokal</a> di AWS SDK dan Panduan Referensi Alat.</li> </ul>
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk AWS CLI, lihat <a href="#">Mengkonfigurasi yang akan AWS CLI digunakan AWS</a></li> </ul>

Pegguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		<p><a href="#">IAM Identity Center</a> dalam Panduan AWS Command Line Interface Pengguna.</p> <ul style="list-style-type: none"> <li>• Untuk AWS SDK, alat, dan AWS API, lihat <a href="#">otentikasi Pusat Identitas IAM</a> di Panduan Referensi AWS SDK dan Alat.</li> </ul>
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk dalam <a href="#">Menggunakan kredensial sementara dengan AWS sumber daya</a> di Panduan Pengguna IAM.

Pegguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk mengetahui AWS CLI, lihat <a href="#">Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna</a>.AWS Command Line Interface</li> <li>• Untuk AWS SDK dan alat bantu, lihat <a href="#">Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi AWS</a> SDK dan Alat.</li> <li>• Untuk AWS API, lihat <a href="#">Mengelola kunci akses untuk pengguna IAM</a> di Panduan Pengguna IAM.</li> </ul>

## Langkah 2: Siapkan izin konsol Amazon Rekognition Custom Labels

Untuk menggunakan konsol Amazon Rekognition, Anda perlu menambahkan untuk memiliki izin yang sesuai. Jika Anda ingin menyimpan file latihan di bucket selain [bucket konsol](#), Anda memerlukan izin tambahan.

### Topik

- [Mengizinkan akses konsol](#)
- [Mengakses Bucket Amazon S3 eksternal](#)

- [Menetapkan izin](#)

## Mengizinkan akses konsol

Untuk menggunakan konsol Amazon Rekognition Custom Labels, Anda memerlukan kebijakan IAM berikut yang mencakup Amazon S3, SageMaker AI Ground Truth, dan Amazon Rekognition Custom Labels. Untuk informasi tentang menetapkan izin, lihat. [Menetapkan izin](#)

JSON

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

```

```
        "s3:PutObjectVersionTagging"
    ],
    "Resource": [
        "arn:aws:s3:::custom-labels-console-*"
    ]
},
{
    "Sid": "rekognitionPolicies",
    "Effect": "Allow",
    "Action": [
        "rekognition:*"
    ],
    "Resource": "*"
},
{
    "Sid": "groundTruthPolicies",
    "Effect": "Allow",
    "Action": [
        "groundtruthlabeling:*"
    ],
    "Resource": "*"
}
]
```

## Mengakses Bucket Amazon S3 eksternal

Saat pertama kali membuka konsol Amazon Rekognition Custom Labels di Wilayah AWS baru, Amazon Rekognition Custom Labels akan membuat bucket (bucket konsol) yang digunakan untuk menyimpan file proyek. Atau, Anda dapat menggunakan bucket Amazon S3 (bucket eksternal) Anda sendiri untuk mengunggah gambar atau file manifes ke konsol. Untuk menggunakan bucket eksternal, tambahkan blok kebijakan berikut ke kebijakan sebelumnya. Ganti `amzn-s3-demo-bucket` dengan nama ember.

```
{
    "Sid": "s3ExternalBucketPolicies",
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
```

```
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectVersion",
        "s3:GetObjectTagging",
        "s3:ListBucket",
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket*"
    ]
}
```

## Menetapkan izin

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Buat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) dalam Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Buat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.
- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

## Langkah 3: Buat bucket konsol

Anda menggunakan project Amazon Rekognition Custom Labels untuk membuat dan mengelola model Anda. Saat pertama kali membuka konsol Amazon Rekognition Custom Labels di Wilayah AWS baru, Amazon Rekognition Custom Labels membuat bucket Amazon S3 (bucket konsol) untuk menyimpan proyek Anda. Anda harus mencatat nama bucket konsol di suatu tempat di mana Anda

dapat merujuknya nanti karena Anda mungkin perlu menggunakan nama bucket dalam operasi AWS SDK atau tugas konsol, seperti membuat kumpulan data.

Format nama bucket adalah `custom-labels-console - <region> -<random value>`. Nilai acak memastikan bahwa tidak ada tabrakan antara nama bucket.

Untuk membuat bucket konsol

1. Pastikan bahwa pengguna memiliki izin yang benar. Untuk informasi selengkapnya, lihat [Mengizinkan akses konsol](#).
2. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
3. Pilih Mulai.
4. Jika ini adalah pertama kalinya Anda membuka konsol di Wilayah AWS saat ini, lakukan hal berikut di kotak dialog Penyiapan Pertama Kali:
  - a. Salin nama bucket Amazon S3 yang ditampilkan. Anda akan memerlukan informasi ini nanti.
  - b. Pilih Buat bucket S3 agar Label Kustom Amazon Rekognition dapat membuat bucket Amazon S3 (bucket konsol) atas nama Anda.
5. Tutup jendela browser.

## Langkah 4: Mengatur AWS CLI and AWS SDK

Anda dapat menggunakan Label Kustom Amazon Rekognition dengan AWS CLI() dan AWS Command Line Interface SDK. AWS Jika Anda perlu menjalankan operasi Amazon Rekognition Custom Labels dari terminal, instal. AWS CLI Jika Anda membuat aplikasi, unduh AWS SDK untuk bahasa pemrograman yang Anda gunakan.

Topik

- [Instal AWS SDK](#)
- [Memberikan akses programatis](#)
- [Siapkan izin SDK](#)
- [Panggil operasi Label Kustom Rekognition Amazon](#)

## Instal AWS SDK

Ikuti langkah-langkah untuk mengunduh dan mengonfigurasi AWS SDK.

Untuk mengatur AWS CLI dan AWS SDK

- Unduh dan instal [AWS CLI](#) dan AWS SDK yang ingin Anda gunakan. Panduan ini memberikan contoh untuk [Java AWS CLI](#), dan [Python](#). Untuk informasi tentang menginstal AWS SDK, lihat [Alat untuk Amazon Web Services](#).

## Memberikan akses programatis

Anda dapat menjalankan contoh AWS CLI dan kode dalam panduan ini di komputer lokal atau AWS lingkungan lain, seperti instans Amazon Elastic Compute Cloud. Untuk menjalankan contoh, Anda perlu memberikan akses ke operasi AWS SDK yang digunakan contoh.

Topik

- [Menjalankan kode di komputer lokal Anda](#)
- [Menjalankan kode di AWS lingkungan](#)

## Menjalankan kode di komputer lokal Anda

Untuk menjalankan kode di komputer lokal, sebaiknya gunakan kredensial jangka pendek untuk memberikan akses pengguna ke operasi AWS SDK. Untuk informasi spesifik tentang menjalankan contoh kode AWS CLI dan pada komputer lokal, lihat [Menggunakan profil di komputer lokal Anda](#).

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. Konsol Manajemen AWS Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Disarankan) Gunakan kredenal konsol sebagai kredensial sementara untuk menandatangani permintaan terprogram ke, SDK AWS CLI, AWS atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk itu AWS CLI, lihat <a href="#">Login untuk pengembangan AWS lokal</a> di Panduan AWS Command Line Interface Pengguna.</li> <li>• Untuk AWS SDK, lihat <a href="#">Login untuk pengembangan AWS lokal</a> di AWS SDK dan Panduan Referensi Alat.</li> </ul>
<p>Identitas tenaga kerja</p> <p>(Pengguna yang dikelola di Pusat Identitas IAM)</p>	Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk AWS CLI, lihat <a href="#">Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center</a> dalam Panduan AWS Command Line Interface Pengguna.</li> <li>• Untuk AWS SDK, alat, dan AWS API, lihat <a href="#">autentikasi Pusat Identitas IAM</a> di Panduan Referensi AWS SDK dan Alat.</li> </ul>
IAM	Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke	Mengikuti petunjuk dalam <a href="#">Menggunakan kredensial sementara dengan AWS</a>

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
	AWS CLI, AWS SDK, atau API. AWS	<a href="#">sumber daya</a> di Panduan Pengguna IAM.
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk mengetahui AWS CLI, lihat <a href="#">Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna</a>.AWS Command Line Interface</li> <li>• Untuk AWS SDK dan alat bantu, lihat <a href="#">Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi AWS</a> SDK dan Alat.</li> <li>• Untuk AWS API, lihat <a href="#">Mengelola kunci akses untuk pengguna IAM</a> di Panduan Pengguna IAM.</li> </ul>

### Menggunakan profil di komputer lokal Anda

Anda dapat menjalankan contoh AWS CLI dan kode dalam panduan ini dengan kredensial jangka pendek yang Anda buat. [Menjalankan kode di komputer lokal Anda](#) Untuk mendapatkan kredensial dan informasi pengaturan lainnya, contoh menggunakan profil bernama custom-labels-access Misalnya:

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

Pengguna yang diwakili profil harus memiliki izin untuk memanggil operasi SDK Label Kustom Rekognition Amazon AWS dan operasi SDK lainnya yang diperlukan oleh contoh. Untuk informasi selengkapnya, lihat [Siapkan izin SDK](#). Untuk menetapkan izin, lihat [Siapkan izin SDK](#).

Untuk membuat profil yang sesuai dengan contoh kode AWS CLI dan, pilih salah satu dari berikut ini. Pastikan nama profil yang Anda buat adalah `custom-labels-access`.

- Pengguna yang dikelola oleh IAM — Ikuti petunjuk di [Beralih ke peran IAM \(AWS CLI\)](#).
- Identitas tenaga kerja (Pengguna dikelola oleh AWS IAM Identity Center) — Ikuti petunjuk di [Mengonfigurasi AWS CLI](#) untuk digunakan. AWS IAM Identity Center Untuk contoh kode, sebaiknya gunakan Integrated Development Environment (IDE), yang mendukung AWS Toolkit yang mengaktifkan otentikasi melalui IAM Identity Center. Untuk contoh Java, lihat [Mulai membangun dengan Java](#). Untuk contoh Python, lihat [Mulai membangun dengan Python](#). Untuk informasi selengkapnya, lihat [kredensial Pusat Identitas IAM](#).

#### Note

Anda dapat menggunakan kode untuk mendapatkan kredensi jangka pendek. Untuk informasi selengkapnya, lihat [Beralih ke peran IAM \(AWS API\)](#). Untuk Pusat Identitas IAM, dapatkan kredensi jangka pendek untuk suatu peran dengan mengikuti instruksi di [Mendapatkan kredensial peran IAM](#) untuk akses CLI.

## Menjalankan kode di AWS lingkungan

Anda tidak boleh menggunakan kredensial pengguna untuk menandatangani panggilan AWS SDK di AWS lingkungan, seperti kode produksi yang berjalan dalam suatu fungsi. AWS Lambda Sebagai gantinya, Anda mengonfigurasi peran yang menentukan izin yang dibutuhkan kode Anda. Anda kemudian melampirkan peran ke lingkungan tempat kode Anda berjalan. Cara Anda melampirkan peran dan membuat kredensial sementara tersedia bervariasi tergantung pada lingkungan tempat kode Anda berjalan:

- AWS Lambda fungsi — Gunakan kredensial sementara yang secara otomatis disediakan Lambda ke fungsi Anda saat mengasumsikan peran eksekusi fungsi Lambda. Kredensialnya tersedia di variabel lingkungan Lambda. Anda tidak perlu menentukan profil. Untuk informasi selengkapnya, silakan lihat [Peran eksekusi Lambda](#).

- Amazon EC2 — Gunakan penyedia kredensial titik akhir metadata instans Amazon EC2. Penyedia secara otomatis membuat dan menyegarkan kredensial untuk Anda menggunakan profil instans Amazon EC2 yang Anda lampirkan ke instans Amazon EC2. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2](#)
- Amazon Elastic Container Service — Gunakan penyedia kredensi Container. Amazon ECS mengirim dan menyegarkan kredensial ke titik akhir metadata. Peran IAM tugas yang Anda tentukan menyediakan strategi untuk mengelola kredensial yang digunakan aplikasi Anda. Untuk informasi selengkapnya, lihat [Berinteraksi dengan layanan AWS](#).

Untuk informasi selengkapnya tentang penyedia kredensi, lihat Penyedia kredensi [terstandarisasi](#).

## Siapkan izin SDK

Untuk menggunakan operasi Amazon Rekognition Custom Labels SDK, Anda memerlukan izin akses ke Amazon Rekognition Custom Labels API dan bucket Amazon S3 yang digunakan untuk pelatihan model.

### Topik

- [Memberikan izin operasi SDK](#)
- [Pembaruan kebijakan untuk menggunakan AWS SDK](#)
- [Menetapkan izin](#)

## Memberikan izin operasi SDK

Sebaiknya Anda hanya memberikan izin yang diperlukan untuk melakukan tugas (izin hak istimewa paling sedikit). Misalnya, untuk menelepon [DetectCustomLabels](#), Anda memerlukan izin untuk melakukan `rekognition:DetectCustomLabels`. Untuk menemukan izin operasi, periksa [referensi API](#).

Ketika Anda baru memulai dengan aplikasi, Anda mungkin tidak tahu izin spesifik yang Anda butuhkan, sehingga Anda dapat mulai dengan izin yang lebih luas. AWS kebijakan terkelola memberikan izin untuk membantu Anda memulai. Anda dapat menggunakan kebijakan `AmazonRekognitionCustomLabelsFullAccess` AWS terkelola untuk mendapatkan akses lengkap ke API Label Kustom Rekognition Amazon. Untuk informasi selengkapnya, lihat [kebijakan terkelola AWS: AmazonRekognitionCustomLabelsFullAccess](#). Bila Anda mengetahui izin yang dibutuhkan aplikasi Anda, kurangi izin lebih lanjut dengan menentukan kebijakan terkelola pelanggan

husus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola pelanggan](#).

Untuk menetapkan izin, lihat. [Menetapkan izin](#)

## Pembaruan kebijakan untuk menggunakan AWS SDK

Untuk menggunakan AWS SDK dengan rilis terbaru Label Kustom Amazon Rekognition, Anda tidak perlu lagi memberikan izin Label Kustom Rekognition Amazon untuk mengakses bucket Amazon S3 yang berisi gambar pelatihan dan pengujian Anda. Jika sebelumnya Anda telah menambahkan izin, Anda tidak perlu menghapusnya. Jika Anda memilih untuk, hapus kebijakan apa pun dari keranjang tempat layanan untuk kepala sekolah beradarekognition.amazonaws.com. Contoh:

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

Untuk informasi selengkapnya, lihat [Menggunakan kebijakan bucket](#).

## Menetapkan izin

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Buat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) dalam Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Buat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

## Panggil operasi Label Kustom Rekognition Amazon

Jalankan kode berikut untuk mengonfirmasi bahwa Anda dapat melakukan panggilan ke Amazon Rekognition Custom Labels API. Kode ini mencantumkan proyek di AWS akun Anda, di AWS Wilayah saat ini. Jika sebelumnya Anda belum membuat proyek, responsnya kosong, tetapi mengonfirmasi bahwa Anda dapat memanggil DescribeProjects operasi.

Secara umum, memanggil fungsi contoh memerlukan klien Rekognition AWS SDK dan parameter lain yang diperlukan. Klien AWS SDK dideklarasikan dalam fungsi utama.

Jika kode gagal, periksa apakah pengguna yang Anda gunakan memiliki izin yang benar. Periksa juga AWS Wilayah yang Anda gunakan sebagai Label Kustom Rekognition Amazon tidak tersedia di semua Wilayah. AWS

Untuk memanggil operasi Label Kustom Rekognition Amazon

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode contoh berikut untuk melihat proyek Anda.

### CLI

Gunakan `describe-projects` perintah untuk membuat daftar proyek di akun Anda.

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

### Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
This example shows how to describe your Amazon Rekognition Custom Labels  
projects.  
If you haven't previously created a project in the current AWS Region,  
the response is an empty list, but does confirm that you can call an  
Amazon Rekognition Custom Labels operation.  
"""
```

```
from botocore.exceptions import ClientError
import boto3

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

    : param rekognition_client: A Boto3 Rekognition client.
    """
    try:
        response = rekognition_client.describe_projects()
        for project in response["ProjectDescriptions"]:
            print("Status: " + project["Status"])
            print("ARN: " + project["ProjectArn"])
            print()
        print("Done!")
    except ClientError as err:
        print(f"Couldn't describe projects. \n{err}")
        raise

def main():
    """
    Entrypoint for script.
    """

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

    describe_projects(rekognition_client)

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

## Java V2

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

```
package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class Hello {

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

    public static void describeMyProjects(RekognitionClient rekClient) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument

        List<String> projectNames = new ArrayList<String>();

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

        // Display useful information for each project.

        DescribeProjectsResponse resp =
rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
        {

            System.out.println("ARN: " + projectDescription.projectArn());
            System.out.println("Status: " +
projectDescription.statusAsString());
```

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

}

public static void main(String[] args) {

    try {

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

        // Describe projects

        describeMyProjects(rekClient);

        rekClient.close();

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

}

}
```

## Langkah 5: (Opsional) Enkripsi file pelatihan

Anda dapat memilih salah satu opsi berikut untuk mengenkripsi file manifes Label Kustom Rekognition Amazon dan file gambar yang ada di bucket konsol atau bucket Amazon S3 eksternal.

- Gunakan tombol Amazon S3 (SSE-S3).
- Gunakan Anda AWS KMS key.

### Note

[Prinsipal IAM](#) panggilan memerlukan izin untuk mendekripsi file. Untuk informasi selengkapnya, lihat [Mendekripsi file yang dienkripsi dengan AWS Key Management Service](#).

Untuk informasi tentang mengenkripsi bucket Amazon S3, [lihat Menyetel perilaku enkripsi sisi server default](#) untuk bucket Amazon S3.

## Mendekripsi file yang dienkripsi dengan AWS Key Management Service

Jika Anda menggunakan AWS Key Management Service (KMS) untuk mengenkripsi file manifes Label Kustom Rekognition Amazon dan file gambar, tambahkan prinsip IAM yang memanggil Label Kustom Rekognition Amazon ke kebijakan kunci KMS. Melakukan hal ini memungkinkan Amazon Rekognition Custom Labels mendekripsi file manifes dan gambar Anda sebelum pelatihan. Untuk informasi selengkapnya, lihat [Bucket Amazon S3 saya memiliki enkripsi default menggunakan kunci AWS KMS khusus. Bagaimana cara mengizinkan pengguna mengunduh dan mengunggah ke bucket?](#)

Prinsipal IAM membutuhkan izin berikut pada kunci KMS.

- km: GenerateDataKey
- kms:Decrypt

Untuk informasi selengkapnya, lihat [Melindungi Data Menggunakan Server-Side Enkripsi dengan kunci KMS yang Disimpan di AWS Key Management Service \(SSE-KMS\)](#).

## Menkripsi gambar pelatihan dan uji yang disalin

Untuk melatih model Anda, Label Kustom Amazon Rekognition membuat salinan pelatihan sumber dan tes citra Anda. Secara default, gambar yang disalin dienkripsi saat istirahat dengan kunci yang dimiliki dan dikelola AWS. Anda juga dapat memilih untuk menggunakan sendiri AWS KMS key. Jika Anda menggunakan kunci KMS Anda sendiri, Anda memerlukan izin berikut pada kunci KMS.

- km: CreateGrant
- km: DescribeKey

Anda secara opsional menentukan tombol KMS saat Anda melatih model dengan konsol atau saat Anda memanggil operasi. `CreateProjectVersion` Kunci KMS yang Anda gunakan tidak harus berupa kunci KMS yang sama dengan yang Anda gunakan untuk mengenkripsi file manifes dan gambar di bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat [Langkah 5: \(Opsional\) Enkripsi file pelatihan](#).

Untuk informasi selengkapnya, lihat [Konsep AWS Key Management Service](#). Citra sumber Anda tidak terpengaruh.

Untuk informasi tentang melatih model, lihat [Melatih model Label Kustom Rekognition Amazon](#).

## Langkah 6: (Opsional) Kaitkan kumpulan data sebelumnya dengan proyek baru

Amazon Rekognition Custom Labels sekarang mengelola kumpulan data dengan proyek. Dataset sebelumnya (sebelumnya) yang Anda buat adalah read-only dan harus dikaitkan dengan proyek sebelum Anda dapat menggunakannya. Saat Anda membuka halaman detail untuk proyek dengan konsol, kami secara otomatis mengaitkan kumpulan data yang melatih versi terbaru model proyek dengan proyek. Asosiasi otomatis kumpulan data dengan proyek tidak terjadi jika Anda menggunakan AWS SDK.

Kumpulan data sebelumnya yang tidak terkait tidak pernah digunakan untuk melatih model atau, digunakan untuk melatih versi model sebelumnya. Halaman kumpulan data Sebelumnya menampilkan semua kumpulan data yang terkait dan tidak terkait.

Untuk menggunakan kumpulan data sebelumnya yang tidak terkait, Anda membuat proyek baru di halaman Kumpulan data Sebelumnya. Dataset menjadi dataset pelatihan untuk proyek baru.

Anda juga dapat membuat proyek untuk kumpulan data yang sudah terkait karena kumpulan data sebelumnya dapat memiliki beberapa asosiasi.

Untuk mengaitkan kumpulan data sebelumnya ke proyek baru

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan.
3. Di panel navigasi kiri, pilih Kumpulan data sebelumnya.
4. Dalam tampilan kumpulan data, pilih kumpulan data sebelumnya yang ingin Anda kaitkan dengan proyek.
5. Pilih Buat proyek dengan kumpulan data.
6. Pada halaman Buat proyek, masukkan nama untuk proyek baru Anda di nama Proyek.
7. Pilih Buat proyek untuk membuat proyek. Proyek ini mungkin membutuhkan waktu beberapa saat untuk dibuat.
8. Gunakan proyek. Untuk informasi selengkapnya, lihat [Memahami Label Kustom Rekognition Amazon](#).

## Menggunakan dataset sebelumnya sebagai dataset pengujian

Anda dapat menggunakan kumpulan data sebelumnya sebagai kumpulan data pengujian untuk proyek yang ada dengan terlebih dahulu mengaitkan kumpulan data sebelumnya dengan proyek baru. Anda kemudian menyalin kumpulan data pelatihan proyek baru ke kumpulan data pengujian proyek yang ada.

Untuk menggunakan dataset sebelumnya sebagai dataset pengujian

1. Ikuti petunjuk di [Langkah 6: \(Opsional\) Kaitkan kumpulan data sebelumnya dengan proyek baru](#) untuk mengaitkan kumpulan data sebelumnya dengan proyek baru.
2. Buat kumpulan data pengujian di proyek yang ada dengan menggunakan penyalinan kumpulan data pelatihan dari proyek baru. Untuk informasi selengkapnya, lihat [Menyalin konten dari kumpulan data yang ada](#).
3. Ikuti petunjuk di [Menghapus proyek Label Kustom Rekognition Amazon \(Konsol\)](#) untuk menghapus proyek baru.

Atau, Anda dapat membuat kumpulan data pengujian dengan menggunakan file manifes untuk kumpulan data sebelumnya. Lihat informasi yang lebih lengkap di [Membuat file manifes](#).

# Memahami Label Kustom Rekognition Amazon

Bagian ini memberi Anda ikhtisar alur kerja untuk melatih dan menggunakan model Label Kustom Rekognition Amazon dengan konsol dan SDK. AWS

## Note

Amazon Rekognition Custom Labels sekarang mengelola kumpulan data dalam sebuah proyek. Anda dapat membuat kumpulan data untuk proyek Anda dengan konsol dan dengan SDK AWS . Jika sebelumnya Anda telah menggunakan Label Kustom Amazon Rekognition, kumpulan data lama Anda mungkin perlu dikaitkan dengan proyek baru. Untuk informasi selengkapnya, lihat [Langkah 6: \(Opsional\) Kaitkan kumpulan data sebelumnya dengan proyek baru](#)

## Topik

- [Tentukan jenis model Anda](#)
- [Buat model](#)
- [Tingkatkan model Anda](#)
- [Mulai model Anda](#)
- [Menganalisis gambar](#)
- [Hentikan model Anda](#)

## Tentukan jenis model Anda

Pertama-tama Anda memutuskan jenis model mana yang ingin Anda latih, yang tergantung pada tujuan bisnis Anda. Misalnya, Anda dapat melatih model untuk menemukan logo Anda di pos media sosial, mengidentifikasi produk Anda di rak-rak toko, atau mengklasifikasikan suku cadang mesin di jalur perakitan.

Amazon Rekognition Custom Labels dapat melatih jenis model berikut:

- [Temukan objek, adegan, dan konsep](#)
- [Temukan lokasi objek](#)

- [Temukan lokasi merek](#)

Untuk membantu Anda memutuskan jenis model yang akan dilatih, Amazon Rekognition Custom Labels menyediakan contoh proyek yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat [Memulai Label Kustom Amazon Rekognition](#).

## Temukan objek, adegan, dan konsep

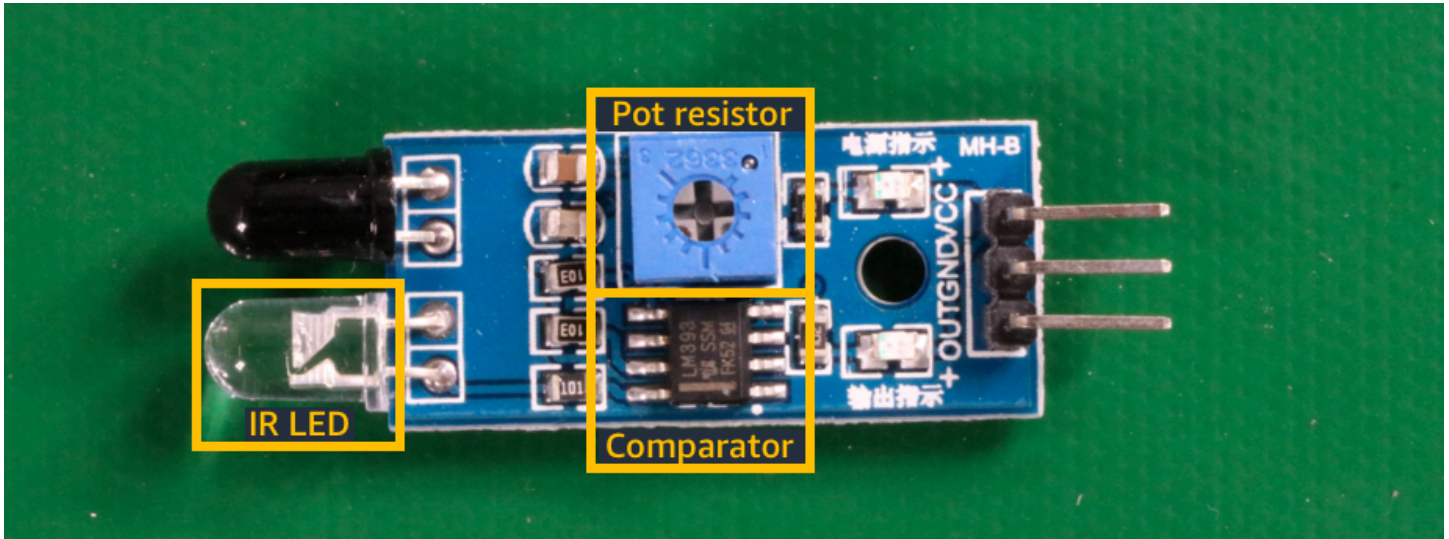
Model memprediksi klasifikasi untuk objek, adegan, dan konsep yang terkait dengan keseluruhan gambar. Misalnya, Anda dapat melatih model yang menentukan apakah suatu gambar berisi objek wisata, atau tidak. Untuk contoh proyek, lihat [Klasifikasi gambar](#). Gambar danau berikut ini adalah contoh jenis gambar yang dapat Anda kenali objek, pemandangan, dan konsep.



Atau, Anda dapat melatih model yang mengkategorikan gambar ke dalam beberapa kategori. Misalnya, gambar sebelumnya mungkin memiliki kategori seperti warna langit, refleksi, atau danau. Untuk contoh proyek, lihat [Klasifikasi gambar multi-label](#).

## Temukan lokasi objek

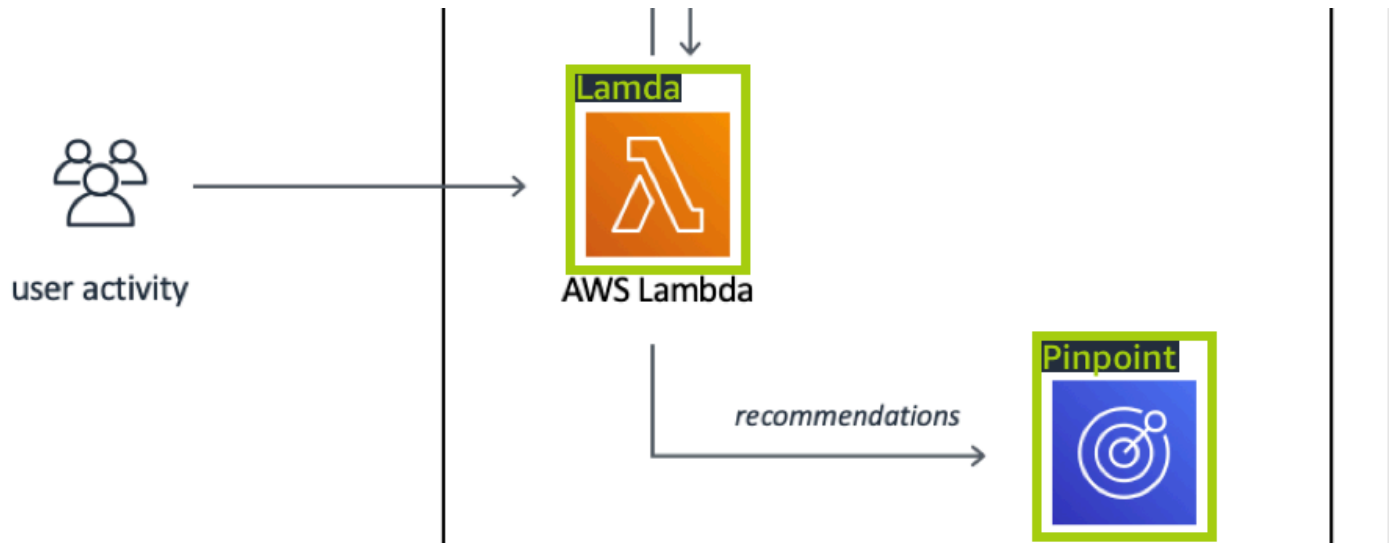
Model memprediksi lokasi objek pada gambar. Prediksi mencakup informasi kotak pembatas untuk lokasi objek dan label yang mengidentifikasi objek di dalam kotak pembatas. Misalnya, gambar berikut menunjukkan kotak pembatas di sekitar berbagai bagian papan sirkuit, seperti komparator atau resistor pot.



Proyek [Lokalisasi objek](#) contoh menunjukkan bagaimana Amazon Rekognition Custom Labels menggunakan kotak pembatas berlabel untuk melatih model yang menemukan lokasi objek.

## Temukan lokasi merek

Amazon Rekognition Custom Labels dapat melatih model yang menemukan lokasi merek, seperti logo, pada gambar. Prediksi tersebut mencakup informasi kotak pembatas untuk lokasi merek dan label yang mengidentifikasi objek di dalam kotak pembatas. Untuk contoh proyek, lihat [Deteksi merek](#). Gambar berikut adalah contoh dari beberapa merek yang dapat dideteksi oleh model.



## Buat model

Langkah-langkah untuk membuat model adalah membuat proyek, membuat kumpulan data pelatihan dan pengujian, dan melatih model.

## Membuat proyek

Proyek Amazon Rekognition Custom Labels adalah sekelompok sumber daya yang diperlukan untuk membuat dan mengelola model. Sebuah proyek mengelola hal-hal berikut:

- **Datasets** — Gambar dan label gambar yang digunakan untuk melatih model. Sebuah proyek memiliki dataset pelatihan dan dataset pengujian.
- **Model** — Perangkat lunak yang Anda latih untuk menemukan konsep, adegan, dan objek yang unik untuk bisnis Anda. Anda dapat memiliki beberapa versi model dalam sebuah proyek.

Kami menyarankan Anda menggunakan proyek untuk kasus penggunaan tunggal, seperti menemukan bagian papan sirkuit pada papan sirkuit.

Anda dapat membuat proyek dengan konsol Amazon Rekognition Custom Labels dan dengan API. [CreateProject](#) Untuk informasi selengkapnya, lihat [Membuat proyek](#).

## Buat kumpulan data pelatihan dan uji

Dataset adalah sekumpulan gambar dan label yang menggambarkan gambar-gambar tersebut. Dalam proyek Anda, Anda membuat kumpulan data pelatihan dan kumpulan data pengujian yang digunakan Label Kustom Amazon Rekognition untuk melatih dan menguji model Anda.

Label mengidentifikasi objek, adegan, konsep, atau kotak pembatas di sekitar objek dalam gambar. Label ditetapkan ke seluruh gambar (tingkat gambar) atau ditetapkan ke kotak pembatas yang mengelilingi objek pada gambar.

### Important

Cara Anda memberi label pada gambar dalam kumpulan data menentukan jenis model yang dibuat Label Kustom Rekognition Amazon. Misalnya, untuk melatih model yang menemukan objek, adegan, dan konsep, Anda menetapkan label tingkat gambar ke gambar dalam kumpulan data pelatihan dan pengujian Anda. Untuk informasi selengkapnya, lihat [Mengarahkan kumpulan data](#).

Gambar harus dalam format PNG dan JPEG, dan Anda harus mengikuti rekomendasi gambar input. Untuk informasi selengkapnya, lihat [Mempersiapkan gambar](#).

## Buat kumpulan data pelatihan dan uji (Konsol)

Anda dapat memulai proyek dengan satu kumpulan data, atau dengan kumpulan data pelatihan dan pengujian terpisah. Jika Anda memulai dengan satu kumpulan data, Amazon Rekognition Custom Labels membagi kumpulan data Anda selama pelatihan untuk membuat kumpulan data pelatihan (80%) dan kumpulan data pengujian (20%) untuk proyek Anda. Mulailah dengan satu kumpulan data jika Anda ingin Label Kustom Rekognition Amazon memutuskan gambar mana yang digunakan untuk pelatihan dan pengujian. Untuk kontrol penuh atas pelatihan, pengujian, dan penyetelan kinerja, kami menyarankan Anda memulai proyek dengan kumpulan data pelatihan dan pengujian terpisah.

Untuk membuat kumpulan data untuk proyek, Anda mengimpor gambar dengan salah satu cara berikut:

- Impor gambar dari komputer lokal Anda.
- Impor gambar dari ember S3. Amazon Rekognition Custom Labels dapat memberi label gambar menggunakan nama folder yang berisi gambar.

- Impor file manifes Amazon SageMaker AI Ground Truth.
- Salin kumpulan data Label Kustom Rekognition Amazon yang ada.

Untuk informasi selengkapnya, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#).

Tergantung dari mana Anda mengimpor gambar, gambar Anda mungkin tidak berlabel. Misalnya, gambar yang diimpor dari komputer lokal tidak diberi label. Gambar yang diimpor dari file manifes Amazon SageMaker AI Ground Truth diberi label. Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk menambahkan, mengubah, dan menetapkan label. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#).

Untuk membuat kumpulan data pelatihan dan pengujian Anda dengan konsol, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#) Untuk tutorial yang mencakup pembuatan kumpulan data pelatihan dan pengujian, lihat [Mengklasifikasikan gambar](#)

## Buat kumpulan data pelatihan dan pengujian (SDK)

Untuk membuat kumpulan data pelatihan dan pengujian, Anda menggunakan API. CreateDataset Anda dapat membuat kumpulan data dengan menggunakan file manifes format Amazon Sagemaker atau dengan menyalin kumpulan data Label Kustom Rekognition Amazon yang ada. Untuk informasi selengkapnya, lihat [Buat kumpulan data pelatihan dan pengujian \(SDK\)](#) Jika perlu, Anda dapat membuat file manifes Anda sendiri. Untuk informasi selengkapnya, lihat [the section called “Membuat file manifes”](#).

## Latih model Anda

Latih model Anda dengan dataset pelatihan. Versi baru dari sebuah model dibuat setiap kali dilatih. Selama pelatihan, Amazon Rekognition Custom Labels menguji kinerja model terlatih Anda. Anda dapat menggunakan hasilnya untuk mengevaluasi dan meningkatkan model Anda. Membutuhkan waktu beberapa saat untuk menyelesaikan pelatihan. Anda hanya dikenakan biaya untuk pelatihan model yang sukses. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#). Jika pelatihan model gagal, Amazon Rekognition Custom Labels menyediakan informasi debugging yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat [Mendebug pelatihan model yang gagal](#).

## Latih model Anda (Konsol)

Untuk melatih model Anda dengan konsol, lihat [Melatih model \(Konsol\)](#).

## Melatih model (SDK)

Anda melatih model Label Kustom Rekognition Amazon dengan menelepon [CreateProjectVersion](#). Untuk informasi selengkapnya, lihat [Melatih model \(SDK\)](#).

## Tingkatkan model Anda

Selama pengujian, Amazon Rekognition Custom Labels membuat metrik evaluasi yang dapat Anda gunakan untuk meningkatkan model terlatih Anda.

## Evaluasi model Anda

Evaluasi kinerja model Anda dengan menggunakan metrik kinerja yang dibuat selama pengujian. Metrik kinerja, seperti F1, presisi, dan penarikan, memungkinkan Anda memahami kinerja model terlatih Anda, dan memutuskan apakah Anda siap menggunakannya dalam produksi. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#).

## Evaluasi model (konsol)

Untuk melihat metrik kinerja, lihat [Mengakses metrik evaluasi \(Konsol\)](#).

## Mengevaluasi model (SDK)

Untuk mendapatkan metrik kinerja, Anda menelepon [DescribeProjectVersions](#) untuk mendapatkan hasil pengujian. Untuk informasi selengkapnya, lihat [Mengakses metrik evaluasi Label Kustom Rekognition Amazon \(SDK\)](#). Hasil pengujian mencakup metrik yang tidak tersedia di konsol, seperti matriks kebingungan untuk hasil klasifikasi. Hasil pengujian dikembalikan dalam format berikut:

- Skor F1 — Nilai tunggal yang mewakili kinerja keseluruhan presisi dan penarikan untuk model. Untuk informasi selengkapnya, lihat [F1](#).
- Lokasi file ringkasan — Ringkasan pengujian mencakup metrik evaluasi agregat untuk seluruh kumpulan data pengujian dan metrik untuk setiap label individual. [DescribeProjectVersions](#) mengembalikan bucket S3 dan lokasi folder dari file ringkasan. Untuk informasi selengkapnya, lihat [Mengakses file ringkasan model](#).
- Lokasi snapshot manifes evaluasi — Snapshot berisi rincian tentang hasil tes, termasuk peringkat kepercayaan dan hasil tes klasifikasi biner, seperti positif palsu. [DescribeProjectVersions](#) mengembalikan bucket S3 dan lokasi folder dari file snapshot. Untuk informasi selengkapnya, lihat [Menafsirkan snapshot manifes evaluasi](#).

## Tingkatkan model Anda

Jika perbaikan diperlukan, Anda dapat menambahkan lebih banyak gambar pelatihan atau meningkatkan pelabelan kumpulan data. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Rekognition Amazon](#). Anda juga dapat memberikan umpan balik tentang prediksi yang dibuat model Anda dan menggunakannya untuk melakukan perbaikan pada model Anda. Untuk informasi selengkapnya, lihat [Meningkatkan model dengan umpan balik Model](#).

### Tingkatkan model Anda (konsol)

Untuk menambahkan gambar ke kumpulan data, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#). Untuk menambah atau mengubah label, lihat [the section called “Pelabelan gambar”](#).

Untuk melatih kembali model Anda, lihat [Melatih model \(Konsol\)](#).

### Tingkatkan model Anda (SDK)

Untuk menambahkan gambar ke kumpulan data atau mengubah label untuk gambar, gunakan API. `UpdateDatasetEntries` `UpdateDatasetEntries` memperbarui atau menambahkan baris JSON ke file manifes. Setiap baris JSON berisi informasi untuk satu gambar, seperti label yang ditetapkan atau informasi kotak pembatas. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar \(SDK\)](#). Untuk melihat entri dalam kumpulan data, gunakan API. `ListDatasetEntries`

Untuk melatih kembali model Anda, lihat [Melatih model \(SDK\)](#).

## Mulai model Anda

Sebelum dapat menggunakan model, Anda memulai model dengan menggunakan konsol Amazon Rekognition Custom Labels atau API. `StartProjectVersion` Anda dikenakan biaya untuk jumlah waktu yang dijalankan model Anda. Untuk informasi selengkapnya, lihat [Menjalankan model terlatih](#).

### Mulai model Anda (konsol)

Untuk memulai model Anda menggunakan konsol, lihat [Memulai model Label Kustom Rekognition Amazon \(Konsol\)](#).

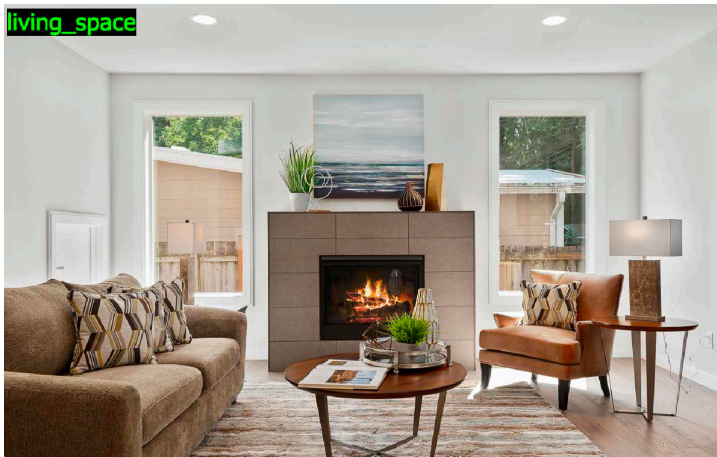
### Mulai model Anda

Anda memulai panggilan model Anda [StartProjectVersion](#). Untuk informasi selengkapnya, lihat [Memulai model Label Kustom Rekognition Amazon \(SDK\)](#).

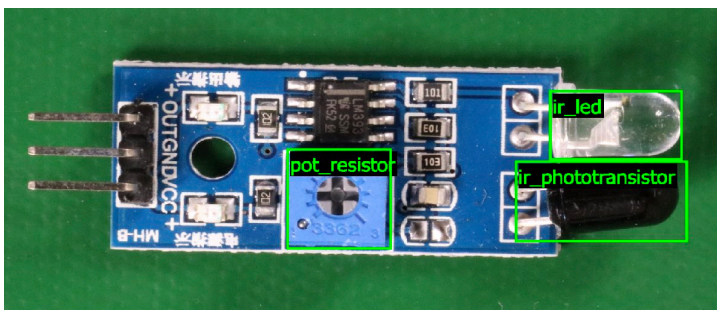
# Menganalisis gambar

Untuk menganalisis gambar dengan model Anda, Anda menggunakan DetectCustomLabels API. Anda dapat menentukan gambar lokal, atau gambar yang disimpan dalam ember S3. Operasi ini juga memerlukan Nama Sumber Daya Amazon (ARN) dari model yang ingin Anda gunakan.

Jika model Anda menemukan objek, adegan, dan konsep, responsnya menyertakan daftar label tingkat gambar yang ditemukan dalam gambar. Misalnya, gambar berikut menunjukkan label tingkat gambar yang ditemukan menggunakan proyek contoh Kamar.



Jika model menemukan lokasi objek, responsnya mencakup daftar kotak pembatas berlabel yang ditemukan pada gambar. Kotak pembatas mewakili lokasi objek pada gambar. Anda dapat menggunakan informasi kotak pembatas untuk menggambar kotak pembatas di sekitar objek. Misalnya, gambar berikut menunjukkan kotak pembatas di sekitar bagian papan sirkuit yang ditemukan menggunakan proyek contoh papan Sirkuit.



Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

## Hentikan model Anda

Anda dikenakan biaya untuk waktu model Anda berjalan. Jika Anda tidak lagi menggunakan model, hentikan model dengan menggunakan konsol Amazon Rekognition Custom Labels, atau dengan menggunakan API. `StopProjectVersion` Untuk informasi selengkapnya, lihat [Menghentikan model Label Kustom Rekognition Amazon](#).

### Hentikan model Anda (Konsol)

Untuk menghentikan model yang sedang berjalan dengan konsol, lihat [Menghentikan model Label Kustom Rekognition Amazon \(Konsol\)](#).

### Hentikan model Anda (SDK)

Untuk menghentikan model yang sedang berjalan, panggil [StopProjectVersion](#). Lihat informasi yang lebih lengkap di [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#).

# Memulai Label Kustom Amazon Rekognition

Sebelum memulai ini Instruksi Memulai, kami sarankan Anda membaca [Memahami Label Kustom Rekognition Amazon](#).

Anda menggunakan Label Kustom Amazon Rekognition untuk melatih model pembelajaran mesin. Model terlatih menganalisis gambar untuk menemukan objek, adegan, dan konsep yang unik untuk kebutuhan bisnis Anda. Misalnya, Anda dapat melatih model untuk mengklasifikasikan gambar rumah, atau menemukan lokasi komponen elektronik pada papan sirkuit tercetak.

Untuk membantu Anda memulai, Amazon Rekognition Custom Labels menyertakan video tutorial dan proyek contoh.

## Note

Untuk informasi tentang AWS Wilayah dan titik akhir yang didukung oleh Label Kustom Rekognition Amazon, [lihat Titik akhir dan kuota Rekognition](#).

## Video tutorial

Video menunjukkan cara menggunakan Label Kustom Amazon Rekognition untuk melatih dan menggunakan model.

Untuk melihat video tutorial

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan. Jika Anda tidak melihat Gunakan Label Kustom, periksa apakah [AWS Wilayah](#) yang Anda gunakan mendukung Label Kustom Amazon Rekognition.
3. Di panel navigasi, pilih Memulai.
4. Dalam Apa itu Label Kustom Rekognition Amazon? , pilih video untuk menonton video ikhtisar.
5. Di panel navigasi, pilih Tutorial.
6. Pada halaman Tutorial, pilih video tutorial yang ingin Anda tonton.

## Contoh proyek

Amazon Rekognition Custom Labels menyediakan contoh proyek berikut.

### Klasifikasi gambar

Proyek klasifikasi gambar (Kamar) melatih model yang menemukan satu atau lebih lokasi rumah tangga dalam sebuah gambar, seperti halaman belakang, dapur, dan teras. Gambar pelatihan dan tes mewakili satu lokasi. Setiap gambar diberi label dengan label tingkat gambar tunggal, seperti dapur, teras, atau living\_space. Untuk gambar yang dianalisis, model terlatih mengembalikan satu atau lebih label yang cocok dari kumpulan label tingkat gambar yang digunakan untuk pelatihan. Misalnya, model mungkin menemukan label living\_space pada gambar berikut. Untuk informasi selengkapnya, lihat [Temukan objek, adegan, dan konsep](#).



### Klasifikasi gambar multi-label

Proyek klasifikasi gambar multi-label (Bunga) melatih model yang mengkategorikan gambar bunga menjadi tiga konsep (jenis bunga, keberadaan daun, dan tahap pertumbuhan).

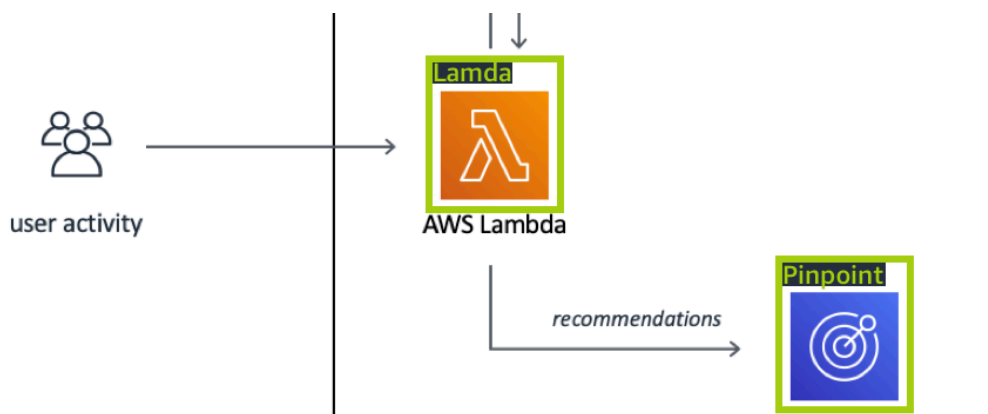
Gambar pelatihan dan uji memiliki label tingkat gambar untuk setiap konsep, seperti kamelia untuk jenis bunga, dengan\_daun untuk bunga dengan daun, dan sepenuhnya tumbuh untuk bunga yang sudah dewasa.

Untuk gambar yang dianalisis, model terlatih mengembalikan label yang cocok dari kumpulan label tingkat gambar yang digunakan untuk pelatihan. Misalnya, model mengembalikan label mediterranean\_spurge dan with\_leaves untuk gambar berikut. Untuk informasi selengkapnya, lihat [Temukan objek, adegan, dan konsep](#).



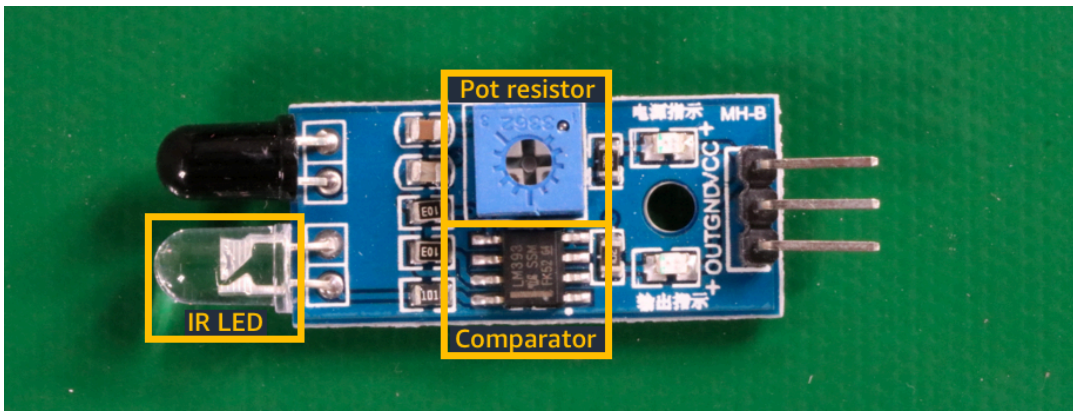
## Deteksi merek

Proyek deteksi merek (Logos) melatih model yang menemukan lokasi AWS logo tertentu seperti Amazon Textract, dan AWS lambda. Gambar pelatihan hanya dari logo dan memiliki label tingkat gambar tunggal, seperti lambda atau textract. Dimungkinkan juga untuk melatih model deteksi merek dengan gambar pelatihan yang memiliki kotak pembatas untuk lokasi merek. Gambar uji memiliki label kotak pembatas yang mewakili lokasi logo di lokasi alami, seperti diagram arsitektur. Model terlatih menemukan logo dan mengembalikan kotak pembatas berlabel untuk setiap logo yang ditemukan. Untuk informasi selengkapnya, lihat [Temukan lokasi merek](#).



## Lokalisasi objek

Proyek lokalisasi objek (Papan sirkuit) melatih model yang menemukan lokasi bagian pada papan sirkuit tercetak, seperti komparator atau dioda pemancar lampu merah infra. Gambar pelatihan dan uji termasuk kotak pembatas yang mengelilingi bagian papan sirkuit dan label yang mengidentifikasi bagian dalam kotak pembatas. Dalam contoh gambar berikut, nama label adalah ir\_phototransistor, ir\_led, pot\_resistor, dan komparator. Model terlatih menemukan bagian papan sirkuit dan mengembalikan batas berlabel untuk setiap bagian sirkuit yang ditemukan. Untuk informasi selengkapnya, lihat [Temukan lokasi objek](#).



## Menggunakan contoh proyek

Instruksi Memulai ini menunjukkan kepada Anda cara melatih model dengan menggunakan contoh proyek yang dibuat Label Kustom Amazon Rekognition untuk Anda. Ini juga menunjukkan kepada Anda bagaimana memulai model dan menggunakannya untuk menganalisis gambar.

## Membuat proyek contoh

Untuk memulai, putuskan proyek mana yang akan digunakan. Untuk informasi selengkapnya, lihat [Langkah 1: Pilih contoh proyek.](#)

Amazon Rekognition Custom Labels menggunakan dataset untuk melatih dan mengevaluasi (menguji) model. Dataset mengelola gambar dan label yang mengidentifikasi konten gambar. Contoh proyek mencakup kumpulan data pelatihan dan kumpulan data pengujian di mana semua gambar diberi label. Anda tidak perlu melakukan perubahan apa pun sebelum melatih model Anda. Contoh proyek menunjukkan dua cara di mana Amazon Rekognition Custom Labels menggunakan label untuk melatih berbagai jenis model.

- image-level — Label mengidentifikasi objek, adegan, atau konsep yang mewakili seluruh gambar.
- bounding box — Label mengidentifikasi isi kotak pembatas. Kotak pembatas adalah satu set koordinat gambar yang mengelilingi objek dalam gambar.

Kemudian, saat Anda membuat proyek dengan gambar Anda sendiri, Anda harus membuat kumpulan data pelatihan dan pengujian, dan juga memberi label pada gambar Anda. Untuk informasi selengkapnya, lihat [Tentukan jenis model Anda.](#)

## Melatih model

Setelah Amazon Rekognition Custom Labels membuat proyek contoh, Anda dapat melatih model. Untuk informasi selengkapnya, lihat [Langkah 2: Latih model Anda](#). Setelah pelatihan selesai, Anda biasanya mengevaluasi kinerja model. Gambar dalam kumpulan data contoh sudah membuat model berkinerja tinggi, dan Anda tidak perlu mengevaluasi model sebelum menjalankan model. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).

## Menggunakan model

Selanjutnya Anda memulai model. Untuk informasi selengkapnya, lihat [Langkah 3: Mulai model Anda](#).

Setelah Anda mulai menjalankan model Anda, Anda dapat menggunakannya untuk menganalisis gambar baru. Untuk informasi selengkapnya, lihat [Langkah 4: Analisis gambar dengan model Anda](#).

Anda dikenakan biaya untuk jumlah waktu yang dijalankan model Anda. Ketika Anda selesai menggunakan model contoh, Anda harus menghentikan model. Untuk informasi selengkapnya, lihat [Langkah 5: Hentikan model Anda](#).

## Langkah selanjutnya

Ketika Anda siap, Anda dapat membuat proyek Anda sendiri. Untuk informasi selengkapnya, lihat [Langkah 6: Langkah selanjutnya](#).

## Langkah 1: Pilih contoh proyek

Pada langkah ini Anda gunakan pilih contoh proyek. Label Kustom Amazon Rekognition kemudian membuat proyek dan kumpulan data untuk Anda. Sebuah proyek mengelola file yang digunakan untuk melatih model Anda. Untuk informasi selengkapnya, lihat [Mengelola proyek Label Kustom Rekognition Amazon](#). Kumpulan data berisi gambar, label yang ditetapkan, dan kotak pembatas yang Anda gunakan untuk melatih dan menguji model. Untuk informasi selengkapnya, lihat [the section called "Mengelola kumpulan data"](#).

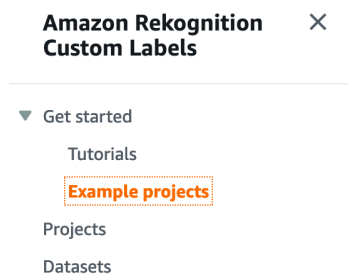
Untuk informasi tentang proyek contoh, lihat [Contoh proyek](#).

Pilih contoh proyek

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>

2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan. Jika Anda tidak melihat Gunakan Label Kustom, periksa apakah [AWS Wilayah](#) yang Anda gunakan mendukung Label Kustom Amazon Rekognition.
3. Pilih Mulai.

Bagian Label Kustom Rekognition Amazon menampilkan Memulai, Tutorial dengan “Contoh proyek” disorot, Proyek, dan Kumpulan Data.



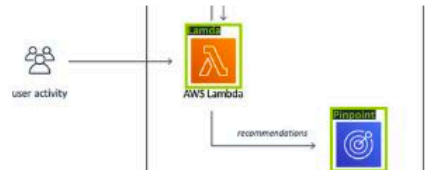
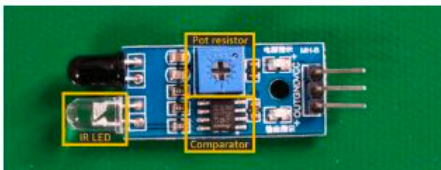


4. Dalam Jelajahi proyek contoh, pilih Coba contoh proyek.
5. Tentukan proyek mana yang ingin Anda gunakan dan pilih Buat proyek **project name** "" di bagian contoh. Amazon Rekognition Custom Labels kemudian membuat proyek contoh untuk Anda.

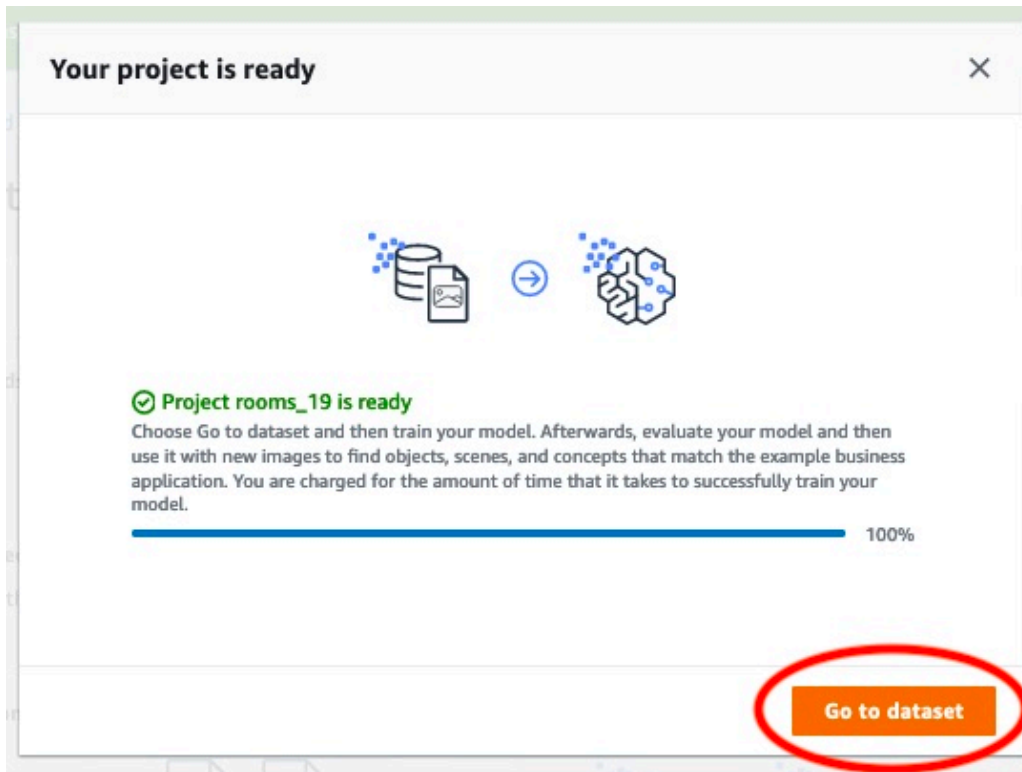
#### Note

Jika ini adalah pertama kalinya Anda membuka konsol di AWS Wilayah saat ini, kotak dialog First Time Set Up akan ditampilkan. Lakukan hal-hal berikut:

1. Perhatikan nama bucket Amazon S3 yang ditampilkan.
2. Pilih Lanjutkan agar Label Kustom Amazon Rekognition membuat bucket Amazon S3 (bucket konsol) atas nama Anda. Gambar konsol di bawah ini menunjukkan contoh dengan tombol “Buat proyek” untuk Klasifikasi Gambar (Kamar), Klasifikasi multi-label (Bunga), Deteksi merek (Logo), dan Pelokalan Objek (Papan sirkuit).

<p><b>Image Classification</b> Recommended for content categorization</p>  <p>Classify images as belonging to a set of predefined labels. For example, real estate companies can use Amazon Rekognition Custom Labels to categorize their images of living rooms, backyards, bedrooms, and other household locations.</p> <p><b>Create project "Rooms"</b></p>	<p><b>Multi-label classification</b> Recommended for inventory management</p>  <p>Classify images into multiple categories, such as the color, size, texture, and type of a flower. For example, plant growers can use Amazon Rekognition Custom Labels to distinguish between different types of flowers and if they are healthy, damaged, or infested.</p> <p><b>Create project "Flowers"</b></p>
<p><b>Brand detection</b> Recommended for retail, media networks, and advertising</p>  <p>Use brand detection to find the location of commercial brands in images. For example, to report on advertiser coverage, media networks can use Amazon Rekognition Custom Labels to report on the location of sponsor logos in photographs.</p> <p><b>Create project "Logos"</b></p>	<p><b>Object localization</b> Recommended for manufacturing and production chains</p>  <p>Use object localization to locate parts used in production or manufacturing lines. For example, in the electronics industry, Amazon Rekognition Custom Labels can help count the number of capacitors on a circuit board.</p> <p><b>Create project "Circuit boards"</b></p>

- Setelah proyek Anda siap, pilih Buka set data. Gambar berikut menunjukkan seperti apa panel proyek saat proyek siap.

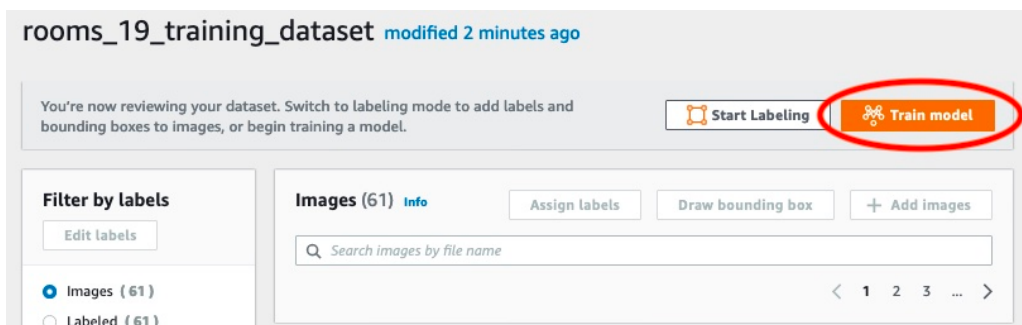


## Langkah 2: Latih model Anda

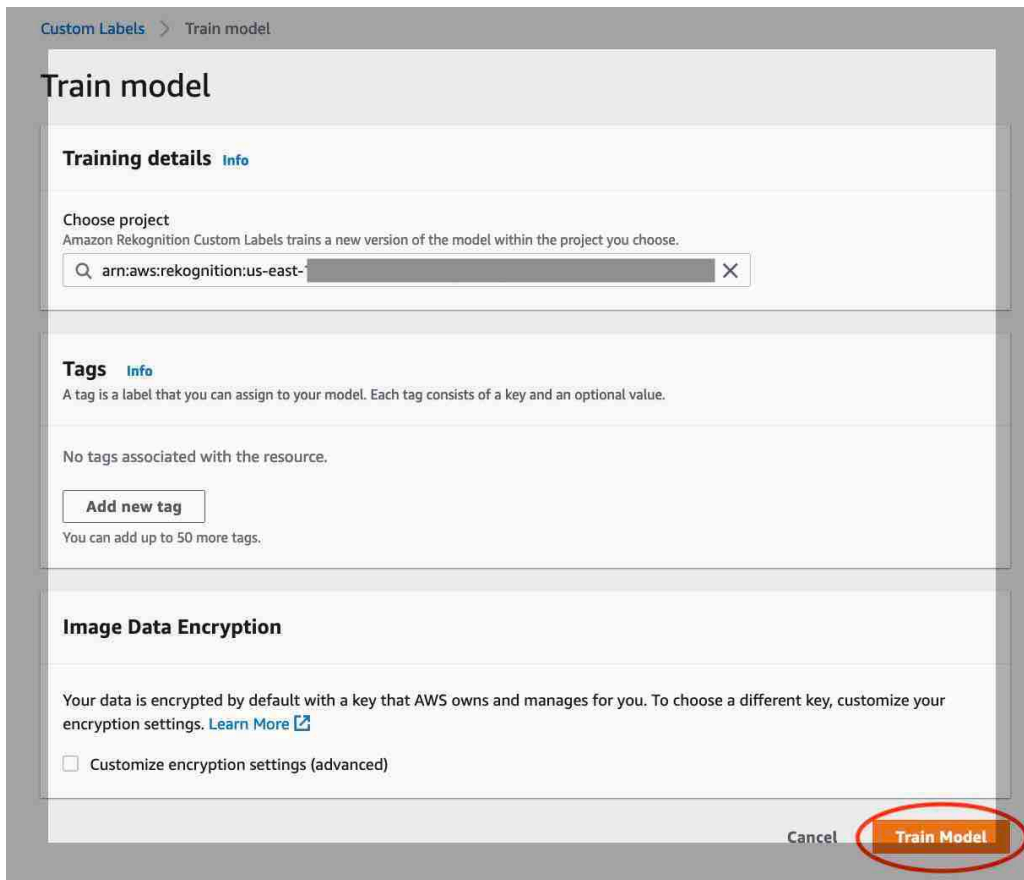
Pada langkah ini Anda melatih model Anda. Kumpulan data pelatihan dan pengujian secara otomatis dikonfigurasi untuk Anda. Setelah pelatihan berhasil diselesaikan, Anda dapat melihat hasil evaluasi keseluruhan, dan hasil evaluasi untuk gambar tes individu. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#).

Untuk melatih model Anda

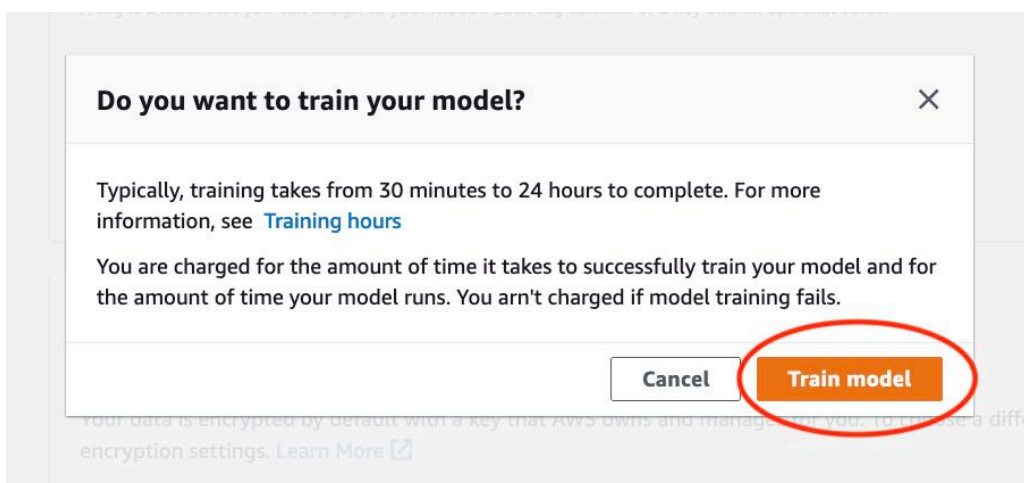
1. Pada halaman dataset, pilih model Train. Gambar berikut menunjukkan konsol dengan tombol model kereta.



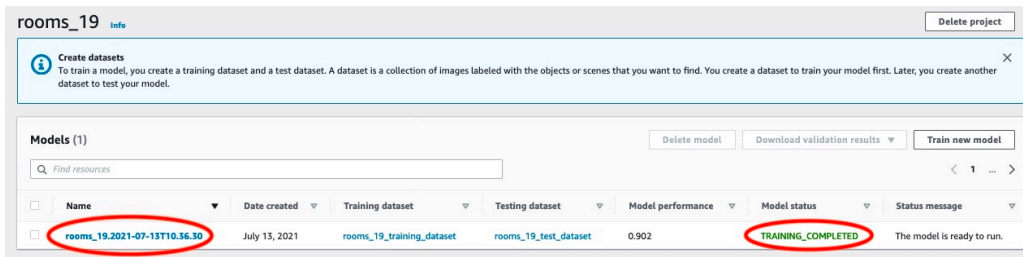
2. Pada halaman model Kereta, Pilih model Kereta. Gambar di bawah ini menunjukkan tombol Train model, perhatikan bahwa Amazon Resource Name (ARN) untuk proyek Anda ada di kotak Pilih project edit.



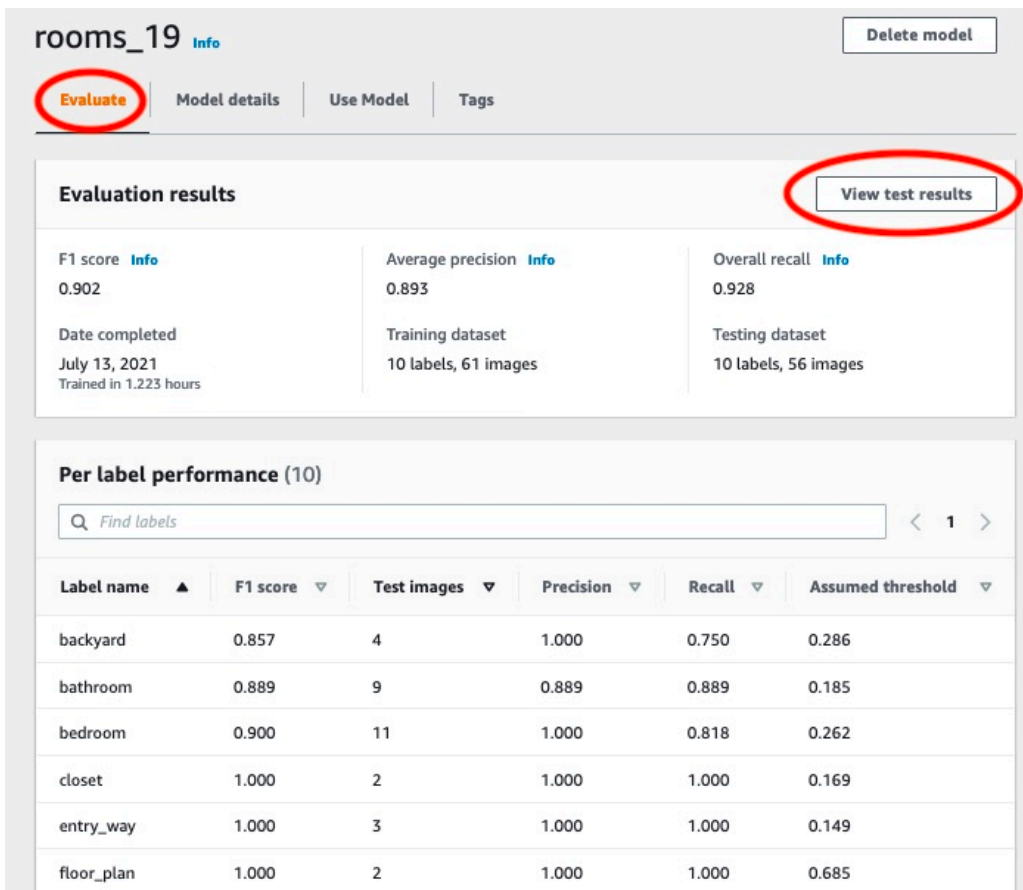
3. Di Apakah Anda ingin melatih model Anda? kotak dialog, yang ditunjukkan pada gambar berikut, pilih Model kereta.



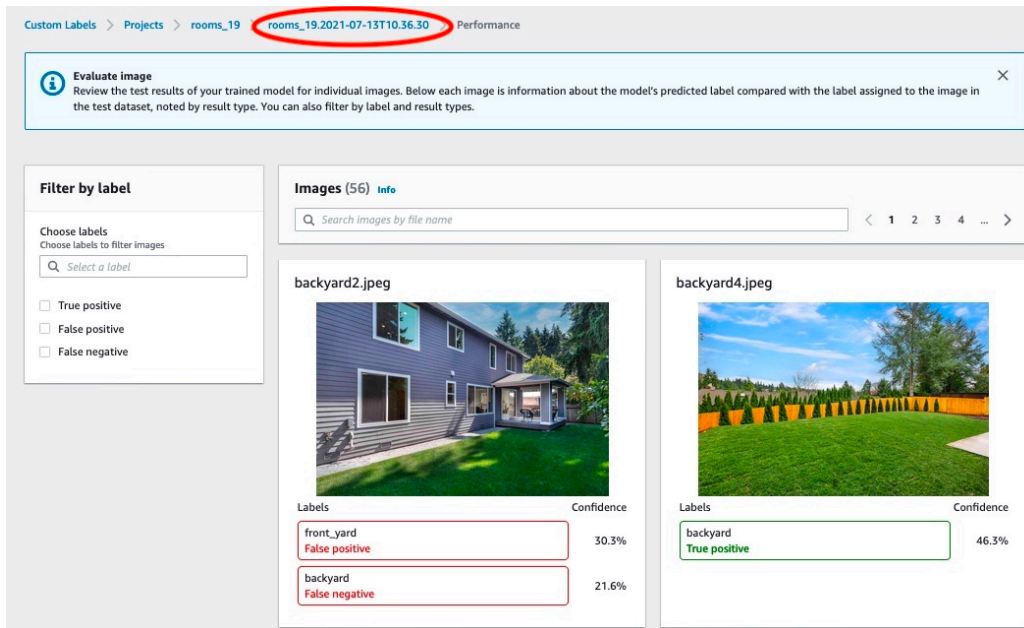
4. Setelah pelatihan selesai, pilih nama model. Pelatihan selesai ketika status model TRAINING\_COMPLETED, seperti yang ditunjukkan pada tangkapan layar konsol berikut.



5. Pilih tombol Evaluasi untuk melihat hasil evaluasi. Untuk informasi tentang mengevaluasi model, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).
6. Pilih Lihat hasil tes untuk melihat hasil gambar pengujian individual. Seperti yang terlihat pada tangkapan layar berikut, dasbor evaluasi menunjukkan metrik seperti skor F1, presisi, dan penarikan untuk setiap label bersama dengan jumlah gambar uji. Metrik keseluruhan seperti rata-rata, presisi, dan penarikan juga ditampilkan.



7. Setelah melihat hasil pengujian, pilih nama model untuk kembali ke halaman model. Tangkapan layar berikut dari dasbor kinerja tempat Anda dapat mengklik kembali ke halaman model.



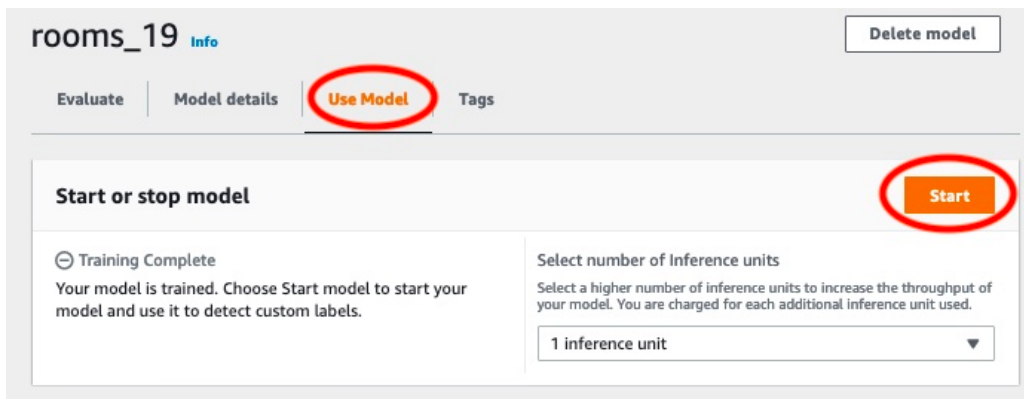
### Langkah 3: Mulai model Anda

Pada langkah ini Anda memulai model Anda. Setelah model Anda dimulai, Anda dapat menggunakannya untuk menganalisis gambar.

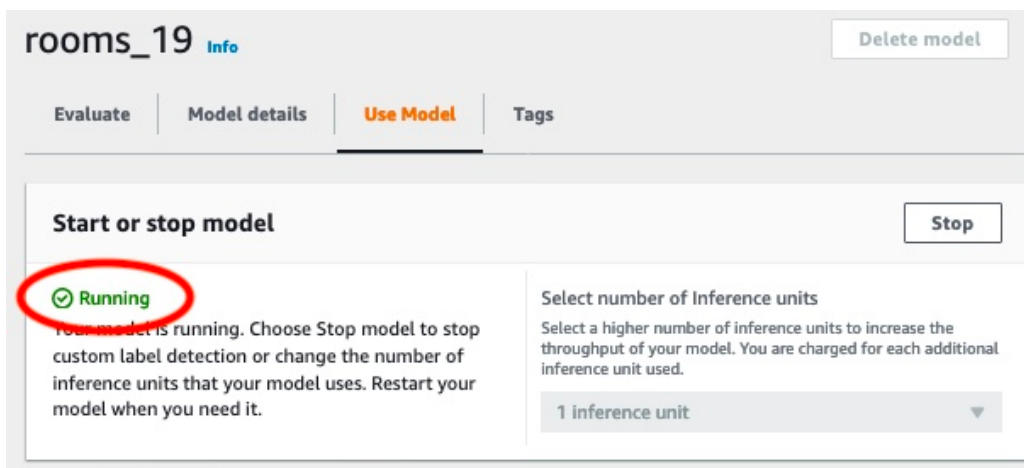
Anda dikenakan biaya untuk jumlah waktu yang dijalankan model Anda. Hentikan model Anda jika Anda tidak perlu menganalisis gambar. Anda dapat me-restart model Anda di lain waktu. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

Untuk memulai model Anda

1. Pilih tab Gunakan model pada halaman model.
2. Di bagian Mulai atau hentikan model lakukan hal berikut:
  - a. Pilih Mulai.
  - b. Dalam kotak dialog Mulai model, pilih Mulai. Gambar berikut menunjukkan tombol Start di panel kontrol model.



3. Tunggu sampai model berjalan. Tangkapan layar berikut menunjukkan konsol saat model sedang berjalan, di mana status di bagian Mulai atau hentikan model sedang Berjalan.



4. Gunakan model Anda untuk mengklasifikasikan gambar. Untuk informasi selengkapnya, lihat [Langkah 4: Analisis gambar dengan model Anda](#).

## Langkah 4: Analisis gambar dengan model Anda

Anda menganalisis gambar dengan memanggil [DetectCustomLabels](#) API. Pada langkah ini, Anda menggunakan perintah `detect-custom-labels` AWS Command Line Interface (AWS CLI) untuk menganalisis contoh gambar. Anda mendapatkan AWS CLI perintah dari konsol Amazon Rekognition Custom Labels. Konsol mengonfigurasi AWS CLI perintah untuk menggunakan model Anda. Anda hanya perlu menyediakan gambar yang disimpan di bucket Amazon S3. Topik ini memberikan gambar yang dapat Anda gunakan untuk setiap contoh proyek.

**Note**

Konsol juga menyediakan kode contoh Python.

Output dari `detect-custom-labels` menyertakan daftar label yang ditemukan dalam gambar, kotak pembatas (jika model menemukan lokasi objek), dan keyakinan yang dimiliki model dalam keakuratan prediksi.

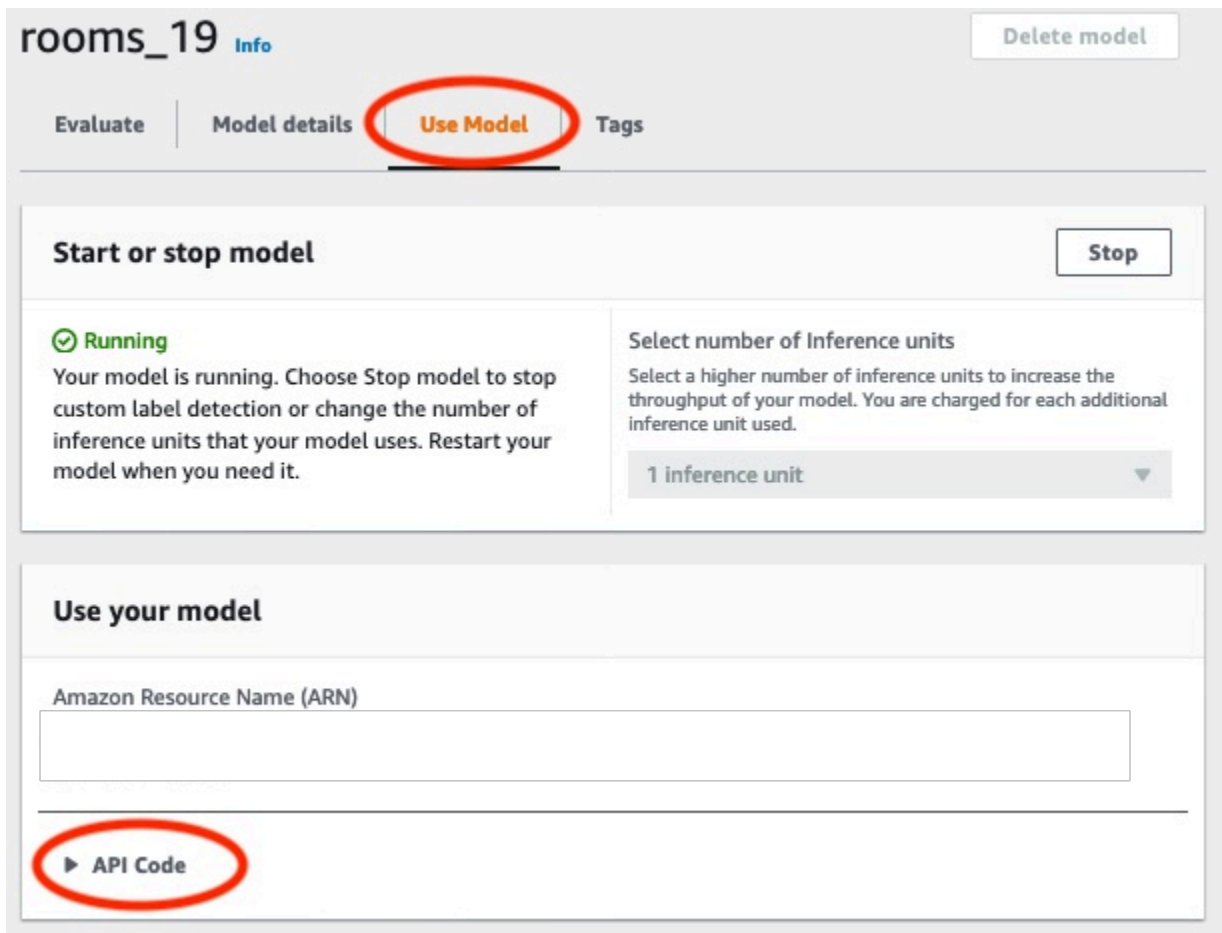
Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

Untuk menganalisis gambar (konsol)

1. `<textobject><phrase>`Status model ditampilkan sebagai Running, dengan tombol Stop untuk menghentikan model yang sedang berjalan. `</phrase></textobject>`

Jika Anda belum melakukannya, atur AWS CLI. Untuk petunjuk, lihat [the section called “Langkah 4: Mengatur AWS CLI and AWS SDK”](#).

2. Jika Anda belum melakukannya, mulailah menjalankan model Anda. Untuk informasi selengkapnya, lihat [Langkah 3: Mulai model Anda](#).
3. Pilih tab Use Model dan kemudian pilih kode API. Panel status model yang ditunjukkan di bawah ini menunjukkan model sebagai Running, dengan tombol Stop untuk menghentikan model yang sedang berjalan, dan opsi untuk menampilkan API.



4. Pilih perintah AWS CLI.
5. Di bagian Analisis gambar, salin AWS CLI perintah yang memanggil `detect-custom-labels`. Gambar konsol Rekognition berikut menunjukkan bagian “Analisis Gambar” dengan perintah AWS CLI untuk mendeteksi label khusus pada gambar menggunakan model pembelajaran mesin, dan instruksi untuk memulai model dan memberikan detail gambar.

**Use your model**

Amazon Resource Name (ARN)

▼ API Code

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

AWS CLI command

Python

**Start model**  
Command used to start the rooms\_ [ ] model.

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

**Analyze image**  
Command used to use analyze an image with the rooms\_ [ ] model. Replace MY\_BUCKET and PATH\_TO\_MY\_IMAGE with your S3 bucket name and image path.

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

6. Unggah contoh gambar ke bucket Amazon S3. Untuk petunjuk, lihat [Mendapatkan contoh gambar](#).
7. Pada prompt perintah, masukkan AWS CLI perintah yang Anda salin pada langkah sebelumnya. Seharusnya terlihat seperti contoh berikut.

Nilai `--project-version-arn` harus Amazon Resource Name (ARN) model Anda. Nilai `--region` harus AWS Wilayah di mana Anda membuat model.

Ubah `MY_BUCKET` dan `PATH_TO_MY_IMAGE` ke bucket Amazon S3 dan gambar yang Anda gunakan pada langkah sebelumnya.

Jika Anda menggunakan [custom-labels-access](#) profil untuk mendapatkan kredensialnya, tambahkan parameternya. `--profile custom-labels-access`

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

Jika model menemukan objek, adegan, dan konsep, output JSON dari AWS CLI perintah akan terlihat mirip dengan berikut ini. Name adalah nama label tingkat gambar yang ditemukan model. Confidence(0-100) adalah kepercayaan model dalam keakuratan prediksi.

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

Jika model menemukan lokasi objek atau menemukan merek, kotak pembatas berlabel dikembalikan. BoundingBox berisi lokasi kotak yang mengelilingi objek. Name adalah objek yang ditemukan model di kotak pembatas. Confidence adalah keyakinan model bahwa kotak pembatas berisi objek.

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

8. Terus gunakan model untuk menganalisis gambar lain. Hentikan model jika Anda tidak lagi menggunakannya. Untuk informasi selengkapnya, lihat [Langkah 5: Hentikan model Anda](#).

## Mendapatkan contoh gambar

Anda dapat menggunakan gambar berikut dengan DetectCustomLabels operasi. Ada satu gambar untuk setiap proyek. Untuk menggunakan gambar, Anda mengunggahnya ke bucket S3.

Untuk menggunakan contoh gambar

1. Klik kanan gambar berikut yang cocok dengan proyek contoh yang Anda gunakan. Kemudian pilih Simpan gambar untuk menyimpan gambar ke komputer Anda. Opsi menu mungkin berbeda, tergantung pada browser mana yang Anda gunakan.
2. Unggah gambar ke bucket Amazon S3 yang dimiliki oleh AWS akun Anda dan berada di AWS wilayah yang sama dengan tempat Anda menggunakan Label Kustom Rekognition Amazon.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

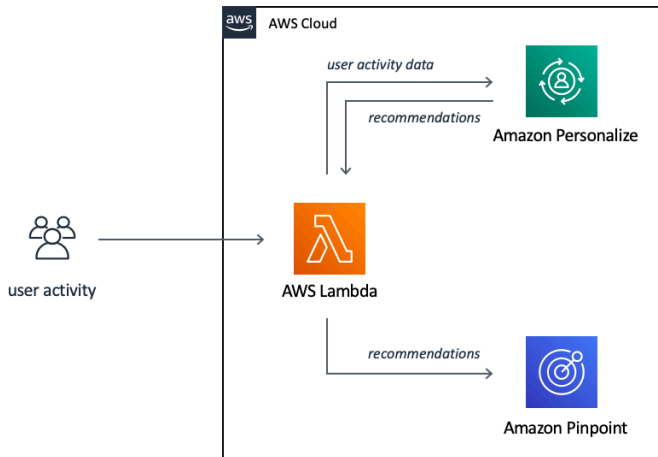
## Klasifikasi gambar



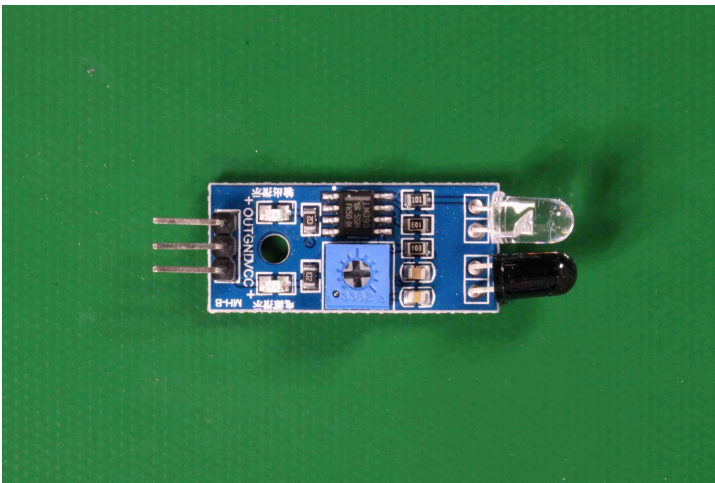
## Klasifikasi multi-label



## Deteksi merek



## Lokalisasi objek

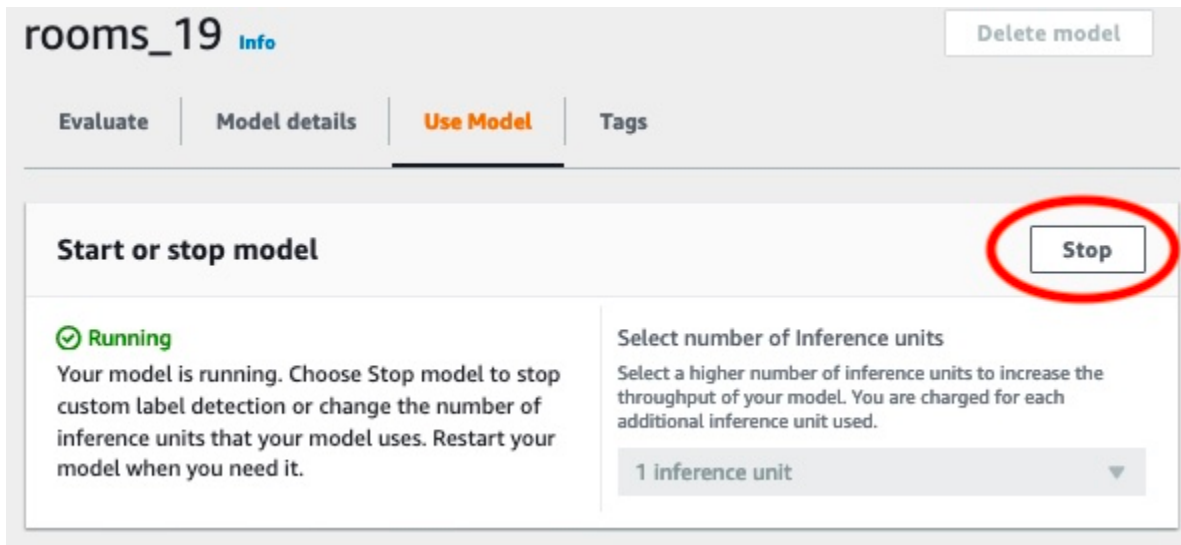


## Langkah 5: Hentikan model Anda

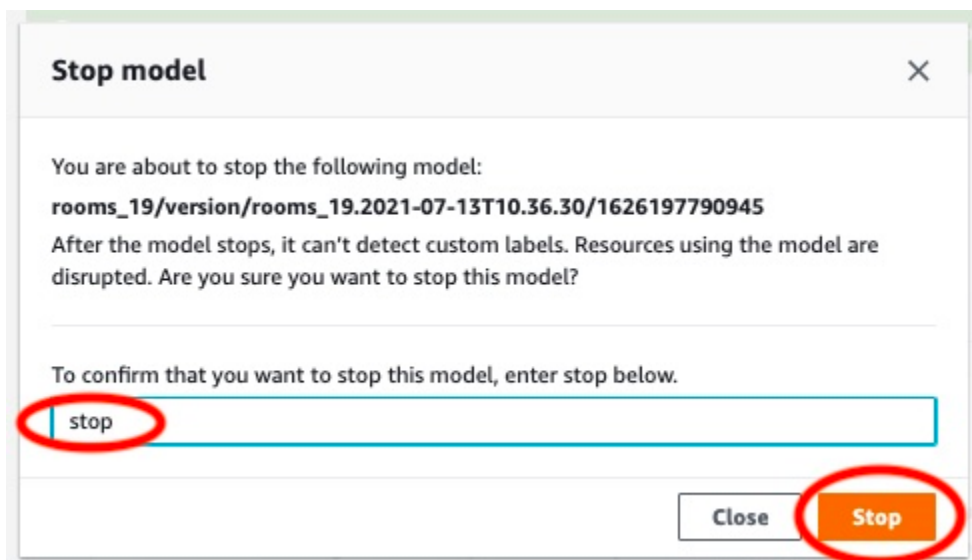
Pada langkah ini Anda berhenti menjalankan model Anda. Anda dikenakan biaya untuk jumlah waktu model Anda berjalan. Jika Anda telah selesai menggunakan model, Anda harus menghentikannya.

Untuk menghentikan model Anda

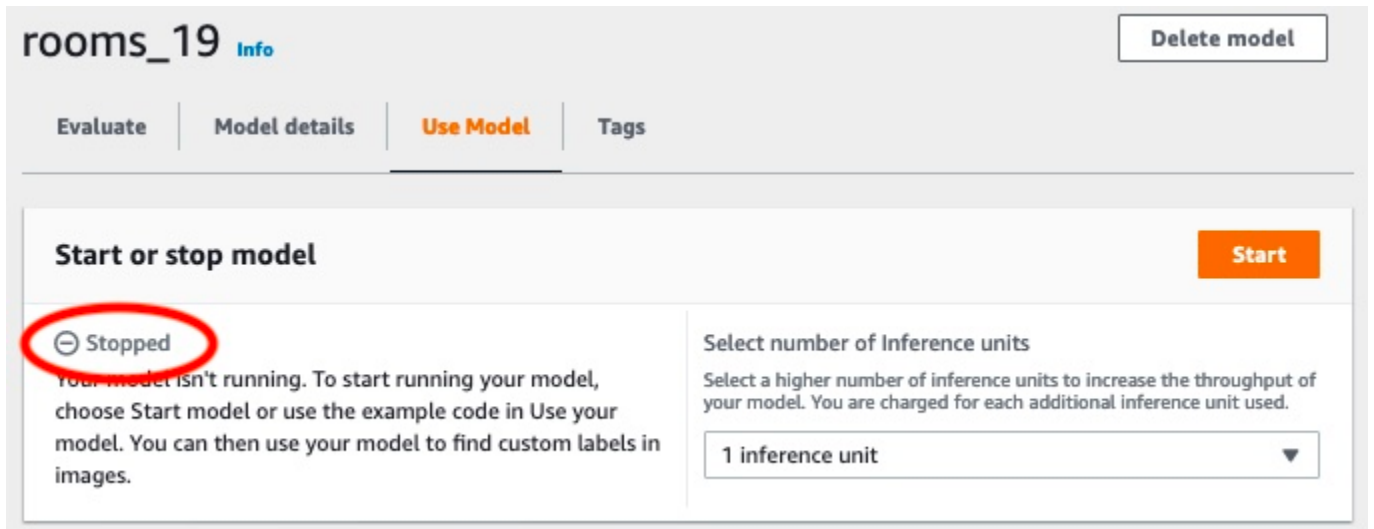
1. Di bagian Mulai atau hentikan model pilih Berhenti.



2. Dalam kotak dialog Stop model, masukkan stop untuk mengonfirmasi bahwa Anda ingin menghentikan model.



3. Pilih Berhenti untuk menghentikan model Anda. Model telah berhenti ketika status di bagian Mulai atau berhenti model Dihentikan. Pada tangkapan layar berikut, bagian Antarmuka pengguna memiliki opsi untuk memulai atau menghentikan model pembelajaran mesin. Status model ditampilkan sebagai "Berhenti" dengan tombol "Mulai" untuk memulai model dan tarik-turun untuk memilih jumlah unit inferensi.



## Langkah 6: Langkah selanjutnya

Setelah Anda selesai mencoba proyek contoh, Anda dapat menggunakan gambar dan kumpulan data Anda sendiri untuk membuat model Anda sendiri. Untuk informasi selengkapnya, lihat [Memahami Label Kustom Rekognition Amazon](#).

Gunakan informasi pelabelan dalam tabel berikut untuk melatih model yang mirip dengan proyek contoh.

Contoh	Gambar pelatihan	Gambar uji
Klasifikasi gambar (Kamar)	1 Label tingkat gambar per gambar	1 Label tingkat gambar per gambar
Klasifikasi multi-label (Bunga)	Beberapa label tingkat gambar per gambar	Beberapa label tingkat gambar per gambar
Deteksi merek (Logo)	label tingkat gambar (Anda juga dapat menggunakan kotak pembatas berlabel)	Kotak pembatas berlabel
Lokalisasi gambar (Papan sirkuit)	Kotak pembatas berlabel	Kotak pembatas berlabel

Ini [Mengklasifikasikan gambar](#) menunjukkan kepada Anda cara membuat proyek, kumpulan data, dan model untuk model klasifikasi Gambar.

Untuk informasi rinci tentang membuat kumpulan data dan model pelatihan, lihat. [Membuat model Label Kustom Rekognition Amazon](#)

# Mengklasifikasikan gambar

Tutorial ini menunjukkan cara membuat proyek dan dataset untuk model yang mengklasifikasikan objek, adegan, dan konsep yang ditemukan dalam gambar. Model mengklasifikasikan seluruh gambar. Misalnya, dengan mengikuti tutorial ini, Anda dapat melatih model untuk mengenali lokasi rumah tangga seperti ruang tamu atau dapur. Tutorial ini juga menunjukkan cara menggunakan model untuk menganalisis gambar.

Sebelum memulai tutorial, kami sarankan Anda membaca [Memahami Label Kustom Rekognition Amazon](#).

Dalam tutorial ini, Anda membuat kumpulan data pelatihan dan pengujian dengan mengunggah gambar dari komputer lokal Anda. Kemudian Anda menetapkan label tingkat gambar ke gambar dalam kumpulan data pelatihan dan pengujian Anda.

Model yang Anda buat mengklasifikasikan gambar sebagai milik kumpulan label tingkat gambar yang Anda tetapkan ke gambar kumpulan data pelatihan. Misalnya, jika kumpulan label tingkat gambar dalam kumpulan data pelatihan Anda adalah `kitchen`, `living_room`, dan `patiobackyard`, model berpotensi menemukan semua label tingkat gambar tersebut dalam satu gambar.

## Note

Anda dapat membuat model untuk tujuan yang berbeda seperti menemukan lokasi objek pada gambar. Untuk informasi selengkapnya, lihat [Tentukan jenis model Anda](#).

## Langkah 1: Kumpulkan gambar Anda

Anda membutuhkan dua set gambar. Satu set untuk ditambahkan ke dataset pelatihan Anda. Set lain untuk ditambahkan ke dataset pengujian Anda. Gambar harus mewakili objek, adegan, dan konsep yang Anda ingin model Anda untuk mengklasifikasikan. Gambar harus dalam format PNG atau JPEG. Untuk informasi selengkapnya, lihat [Mempersiapkan gambar](#).

Anda harus memiliki setidaknya 10 gambar untuk kumpulan data pelatihan Anda dan 10 gambar untuk kumpulan data pengujian Anda.

Jika Anda belum memiliki gambar, gunakan gambar dari proyek klasifikasi contoh Kamar. Setelah membuat proyek, gambar pelatihan dan pengujian berada di lokasi bucket Amazon S3 berikut:

- Gambar pelatihan - `s3://custom-labels-console-region-numbers/assets/rooms_version number_test_dataset/`
- Gambar uji - `s3://custom-labels-console-region-numbers/assets/rooms_version number_test_dataset/`

*region* adalah AWS Wilayah tempat Anda menggunakan konsol Label Kustom Rekognition Amazon. *numbers* adalah nilai yang diberikan konsol ke nama bucket. *Version number* adalah nomor versi untuk proyek contoh, mulai dari 1.

Prosedur berikut menyimpan gambar dari proyek Kamar ke folder lokal di komputer Anda bernama `training` dan `test`.

Untuk mengunduh file gambar proyek contoh Kamar

1. Buat proyek Kamar. Untuk informasi selengkapnya, lihat [Langkah 1: Pilih contoh proyek](#).
2. Buka prompt perintah dan masukkan perintah berikut untuk mengunduh gambar pelatihan.

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

3. Pada prompt pujian, masukkan perintah berikut untuk mengunduh gambar uji.

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

4. Pindahkan dua gambar dari folder pelatihan ke folder terpisah yang Anda pilih. Anda akan menggunakan gambar untuk mencoba model terlatih Anda [Langkah 9: Analisis gambar dengan model Anda](#).

## Langkah 2: Tentukan kelas Anda

Buat daftar kelas yang Anda ingin model Anda temukan. Misalnya, jika Anda melatih model untuk mengenali kamar di rumah, Anda dapat mengklasifikasikan gambar berikut sebagai `living_room`.



Setiap kelas memetakan ke label tingkat gambar. Kemudian Anda menetapkan label tingkat gambar ke gambar dalam kumpulan data pelatihan dan pengujian Anda.

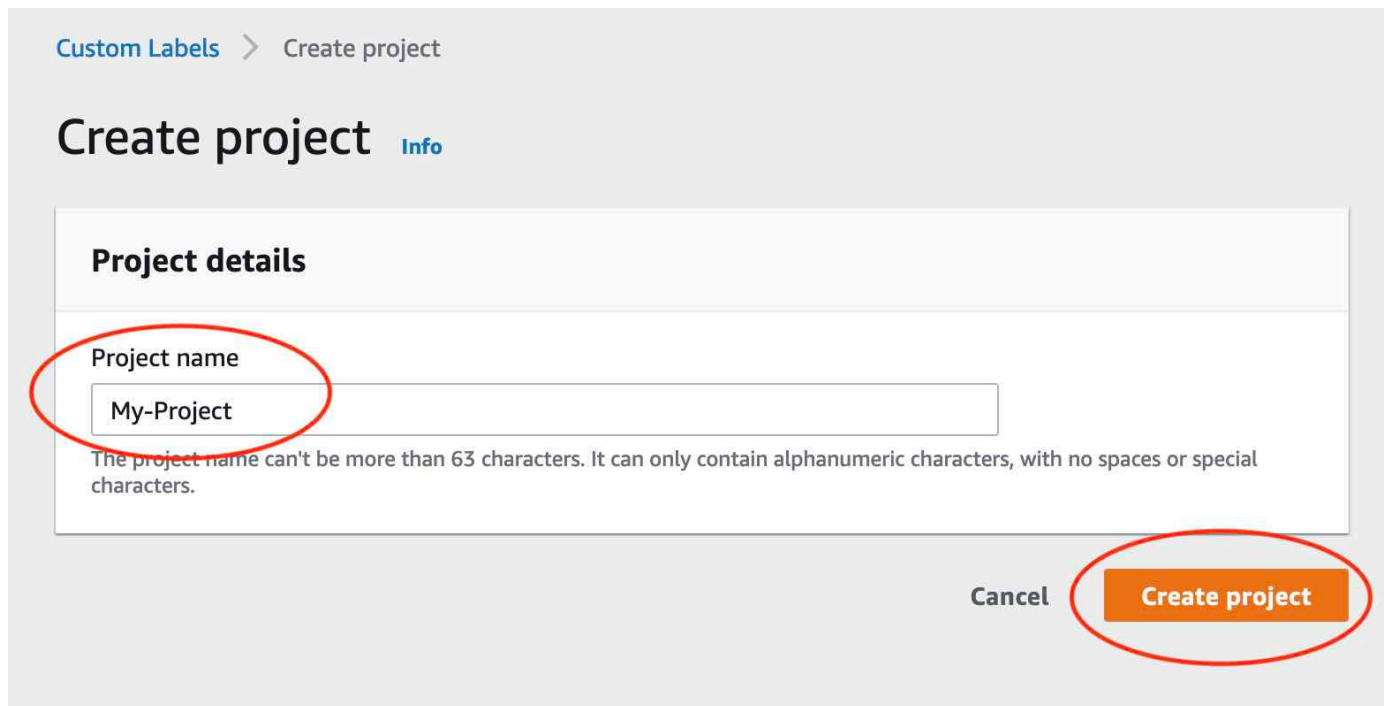
Jika Anda menggunakan gambar dari proyek contoh Kamar, label tingkat gambar adalah halaman belakang, kamar mandi, kamar tidur, lemari, entry\_way, floor\_plan, front\_yard, dapur, living\_space, dan patio.

## Langkah 3: Buat proyek

Untuk mengelola kumpulan data dan model, Anda membuat proyek. Setiap proyek harus membahas kasus penggunaan tunggal, seperti mengenali kamar di rumah.

Untuk membuat proyek (konsol)

1. Jika Anda belum melakukannya, siapkan konsol Label Kustom Rekognition Amazon. Untuk informasi selengkapnya, lihat [Menyiapkan Label Kustom Rekognition Amazon](#).
2. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
3. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan.
4. Halaman landing Label Kustom Rekognition Amazon, pilih Memulai
5. Di panel navigasi kiri, pilih Proyek.
6. Pada halaman proyek, pilih Buat Proyek.
7. Dalam Nama proyek, masukkan nama untuk proyek Anda.
8. Pilih Buat proyek untuk membuat proyek Anda.



Custom Labels > Create project

## Create project Info

### Project details

Project name

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

Cancel **Create project**

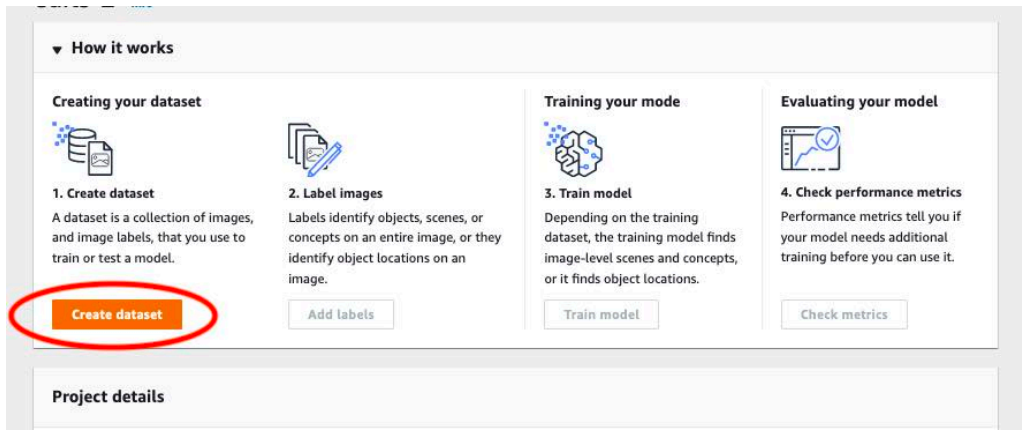
## Langkah 4: Buat kumpulan data pelatihan dan uji

Pada langkah ini Anda membuat kumpulan data pelatihan dan kumpulan data pengujian dengan mengunggah gambar dari komputer lokal Anda. Anda dapat mengunggah sebanyak 30 gambar sekaligus. Jika Anda memiliki banyak gambar untuk diunggah, pertimbangkan untuk membuat kumpulan data dengan mengimpor gambar dari bucket Amazon S3. Untuk informasi selengkapnya, lihat [Mengimpor gambar dari bucket Amazon S3](#).

Untuk informasi selengkapnya tentang kumpulan data, lihat [Mengelola kumpulan data](#)

Untuk membuat kumpulan data menggunakan gambar di komputer lokal (konsol)

1. Pada halaman detail proyek, pilih Buat kumpulan data.



2. Di bagian Konfigurasi awal, pilih Mulai dengan kumpulan data pelatihan dan kumpulan data pengujian.
3. Di bagian Detail kumpulan data pelatihan, pilih Unggah gambar dari komputer Anda.
4. Di bagian Uji detail kumpulan data, pilih Unggah gambar dari komputer Anda.
5. Pilih Buat kumpulan data.

### Create dataset Info

#### Starting configuration

Configuration options

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

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

**What are training datasets and test datasets?**

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

#### Training dataset details

Import images Info

Import images from one of the sources below.

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

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

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

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

#### Test dataset details

Import images Info

Import images from one of the sources below.

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

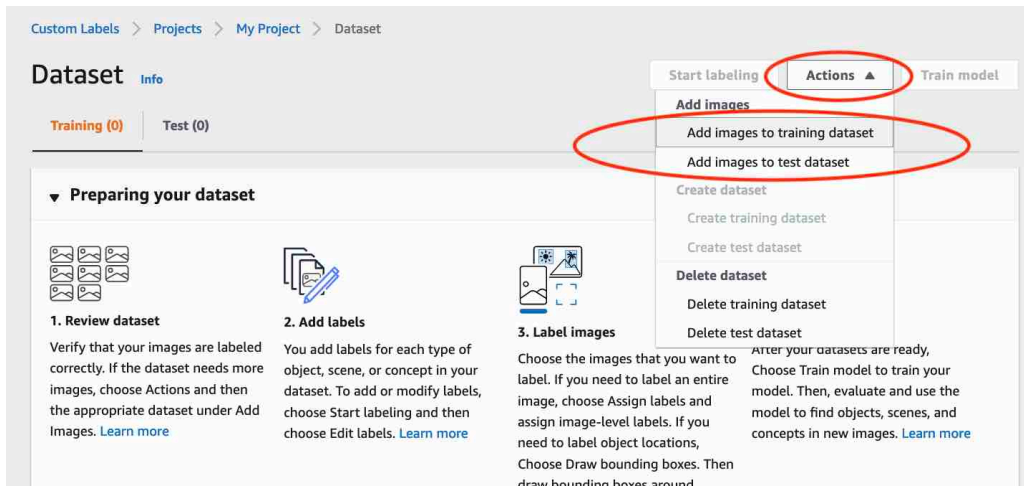
Upload images from your computer  
Add images by uploading files from your local computer. You're limited to uploading 50 images at one time.

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

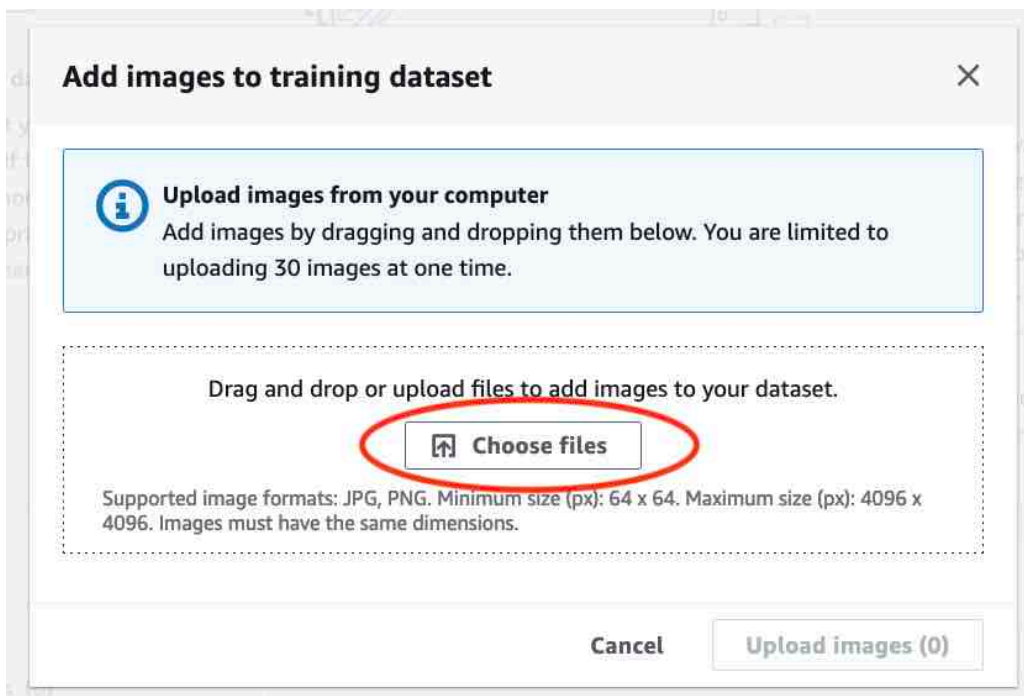
Import images labeled by SageMaker Ground Truth  
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.

Cancel

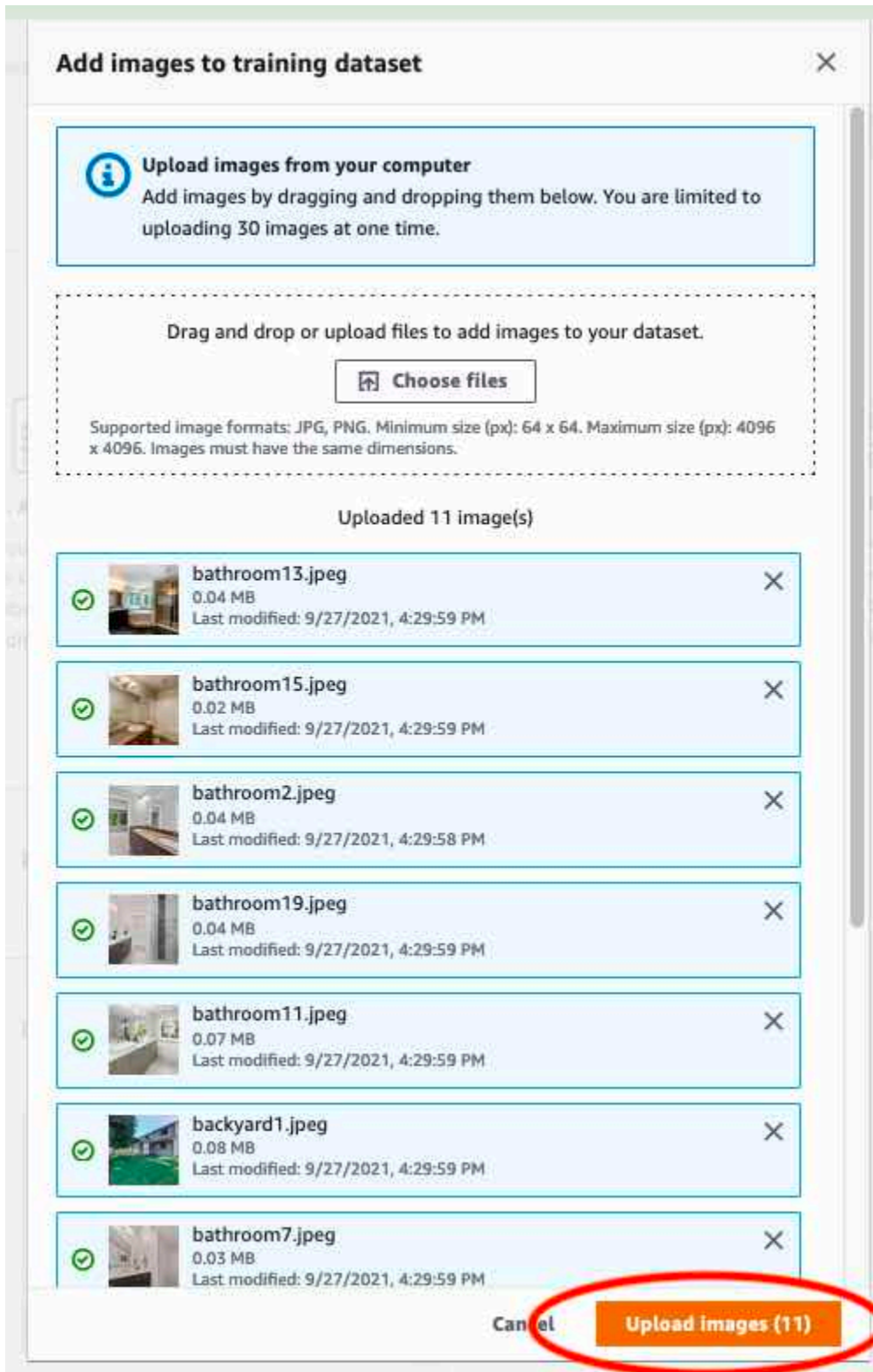
6. Halaman kumpulan data muncul dengan tab Pelatihan dan tab Uji untuk kumpulan data masing-masing.
7. Pada halaman dataset, pilih tab Pelatihan.
8. Pilih Tindakan, lalu pilih Tambahkan gambar ke kumpulan data pelatihan.



9. Dalam kotak dialog Tambahkan gambar ke kumpulan data pelatihan, pilih Pilih file.



10. Pilih gambar yang ingin Anda unggah ke kumpulan data. Anda dapat mengunggah sebanyak 30 gambar sekaligus.
11. Pilih Unggah gambar. Mungkin perlu beberapa detik untuk Amazon Rekognition Custom Labels untuk menambahkan gambar ke dataset.



12. Jika Anda memiliki lebih banyak gambar untuk ditambahkan ke kumpulan data pelatihan, ulangi langkah 9-12.

13. Pilih tab Uji.

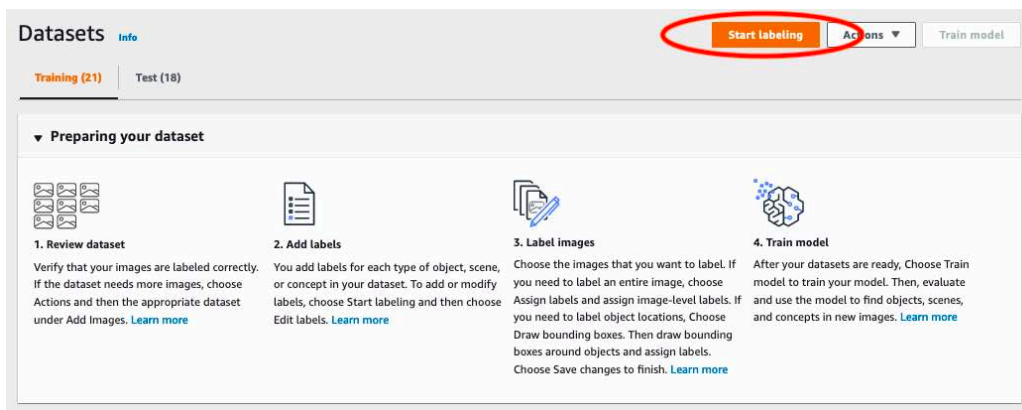
14. Ulangi langkah 8 - 12 untuk menambahkan gambar ke kumpulan data pengujian. Untuk langkah 8, pilih Tindakan dan kemudian pilih Tambahkan gambar untuk menguji kumpulan data.

## Langkah 5: Tambahkan label ke proyek

Pada langkah ini Anda menambahkan label ke proyek untuk setiap kelas yang Anda identifikasi dalam langkah [Langkah 2: Tentukan kelas Anda](#).

Untuk menambahkan label baru (konsol)

1. Pada halaman galeri kumpulan data, pilih Mulai pelabelan untuk masuk ke mode pelabelan.



2. Di bagian Label galeri kumpulan data, pilih Edit label untuk membuka kotak dialog Kelola label.
3. Di kotak edit, masukkan nama label baru.
4. Pilih Tambahkan label.
5. Ulangi langkah 3 dan 4 sampai Anda telah membuat semua label yang Anda butuhkan.
6. Pilih Simpan untuk menyimpan label yang Anda tambahkan.

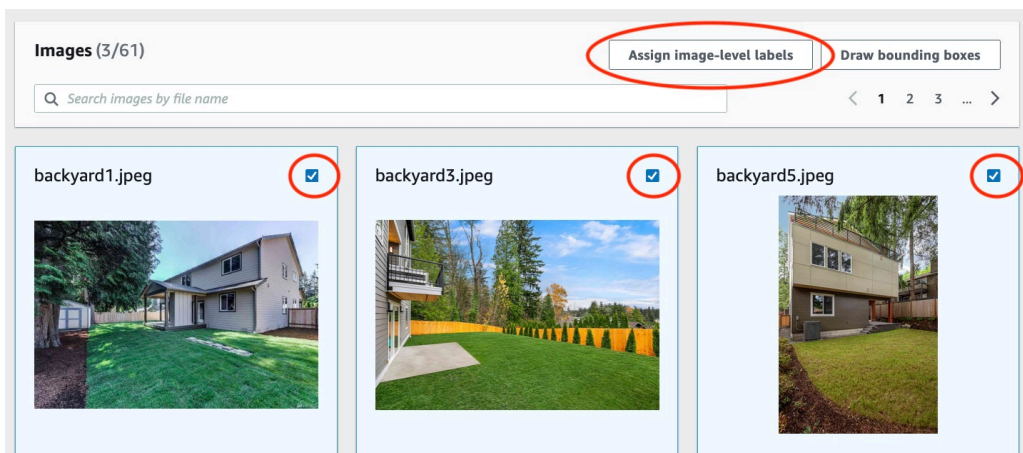
## Langkah 6: Tetapkan label tingkat gambar ke kumpulan data pelatihan dan uji

Pada langkah ini Anda menetapkan satu tingkat gambar untuk setiap gambar dalam kumpulan data pelatihan dan pengujian Anda. Label tingkat gambar adalah kelas yang diwakili oleh setiap gambar.

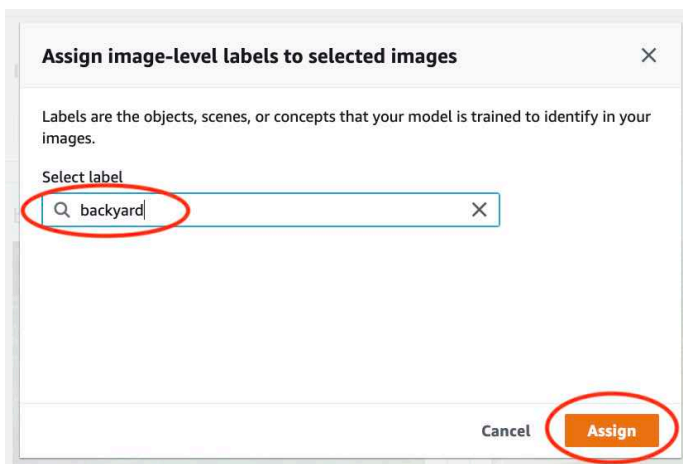
Untuk menetapkan label tingkat gambar ke gambar (konsol)

1. Pada halaman Datasets, pilih tab Pelatihan.

2. Pilih Mulai pelabelan untuk masuk ke mode pelabelan.
3. Pilih satu atau beberapa gambar yang ingin Anda tambahkan label. Anda hanya dapat memilih gambar pada satu halaman pada satu waktu. Untuk memilih rentang gambar yang berdekatan pada halaman:
  - a. Pilih gambar pertama.
  - b. Tekan dan tahan tombol shift.
  - c. Pilih gambar kedua. Gambar antara gambar pertama dan kedua juga dipilih.
  - d. Lepaskan tombol shift.
4. Pilih Tetapkan label tingkat gambar.



5. Di kotak dialog Tetapkan label tingkat gambar ke gambar yang dipilih, pilih label yang ingin Anda tetapkan ke gambar atau gambar.
6. Pilih Tetapkan untuk menetapkan label pada gambar.



7. Ulangi pelabelan hingga setiap gambar dianotasi dengan label yang diperlukan.
8. Pilih tab Uji.

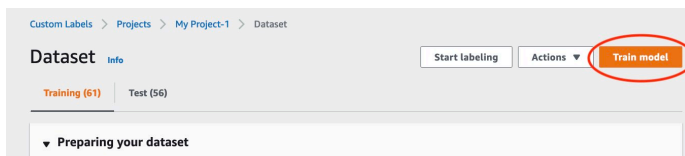
9. Ulangi langkah untuk menetapkan label tingkat gambar ke gambar kumpulan data pengujian.

## Langkah 7: Latih model Anda

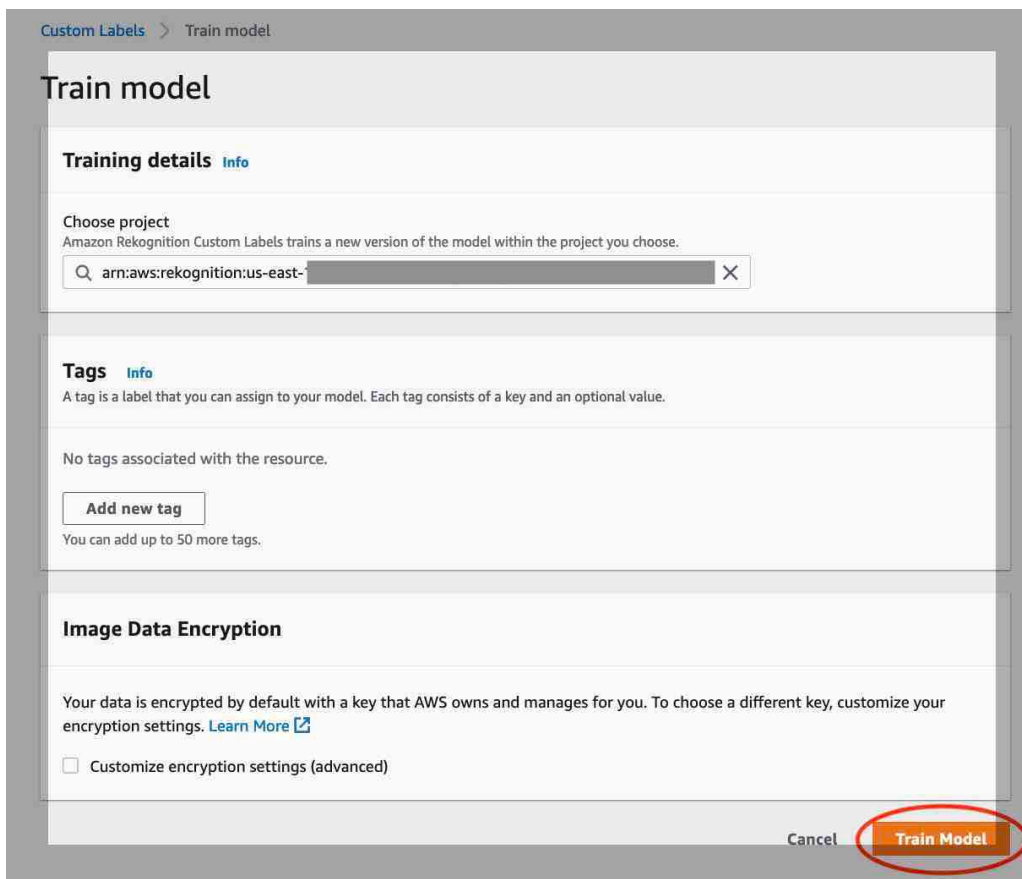
Gunakan langkah-langkah berikut untuk melatih model Anda. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#).

Untuk melatih model Anda (konsol)

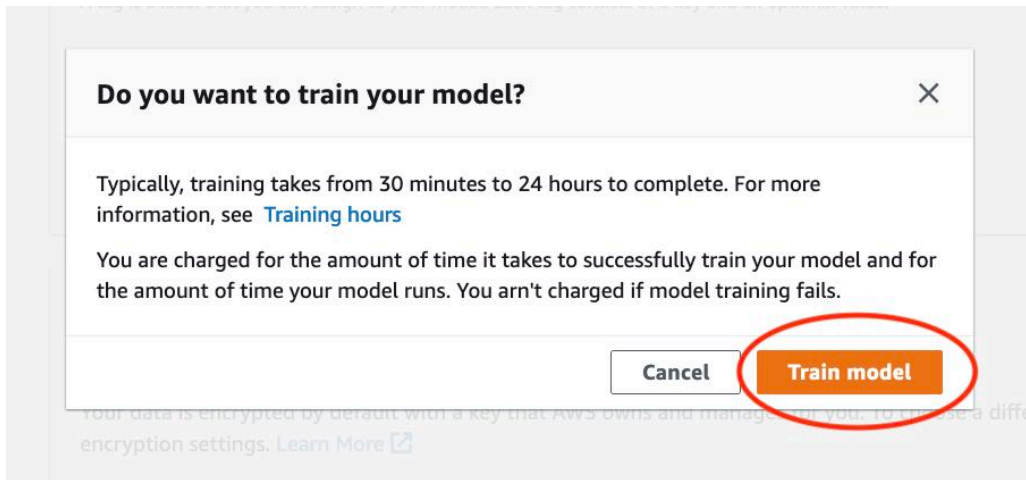
1. Pada halaman Dataset, pilih Train model.



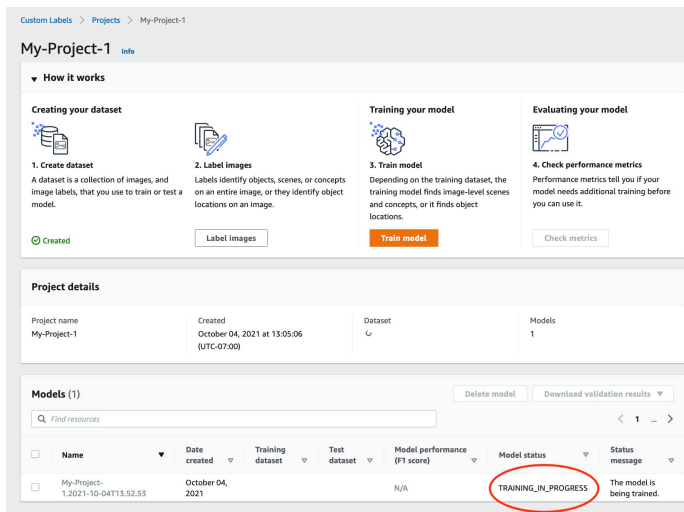
2. Pada halaman model Kereta, pilih Model kereta. Nama Sumber Daya Amazon (ARN) untuk proyek Anda ada di kotak Pilih edit proyek.



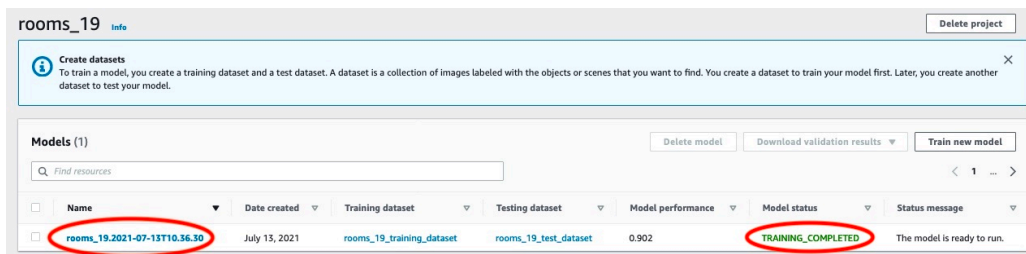
3. Di Apakah Anda ingin melatih model Anda? kotak dialog, pilih Model kereta.



- Di bagian Model pada halaman proyek, Anda dapat melihat bahwa pelatihan sedang berlangsung. Anda dapat memeriksa status saat ini dengan melihat Model Status kolom untuk versi model. Pelatihan model membutuhkan waktu beberapa saat untuk diselesaikan.



- Setelah pelatihan selesai, pilih nama model. Pelatihan selesai ketika status model adalah TRAINING\_COMPLETED.



- Pilih tombol Evaluasi untuk melihat hasil evaluasi. Untuk informasi tentang mengevaluasi model, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih.](#)

- Pilih Lihat hasil tes untuk melihat hasil gambar pengujian individual. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#).

**rooms\_19** Info Delete model

**Evaluate** | Model details | Use Model | Tags

### Evaluation results

**View test results**

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

### Per label performance (10)

Find labels

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

- Setelah melihat hasil pengujian, pilih nama model untuk kembali ke halaman model.

Custom Labels > Projects > rooms\_19 **rooms\_19.2021-07-13T10.36.30** Performance

**Evaluate image**

Review the test results of your trained model for individual images. Below each image is information about the model's predicted label compared with the label assigned to the image in the test dataset, noted by result type. You can also filter by label and result types.

**Filter by label**

Choose labels  
Choose labels to filter images

Select a label

True positive

False positive

False negative

**Images (56)** Info

Search images by file name

**backyard2.jpeg**

Labels: front\_yard (False positive) 30.3%, backyard (False negative) 21.6%

**backyard4.jpeg**

Labels: backyard (True positive) 46.3%

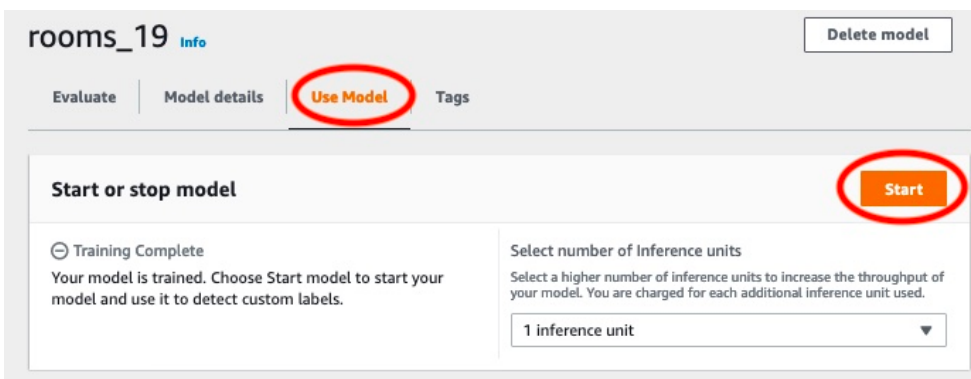
## Langkah 8: Mulai model Anda

Pada langkah ini Anda memulai model Anda. Setelah model Anda dimulai, Anda dapat menggunakannya untuk menganalisis gambar.

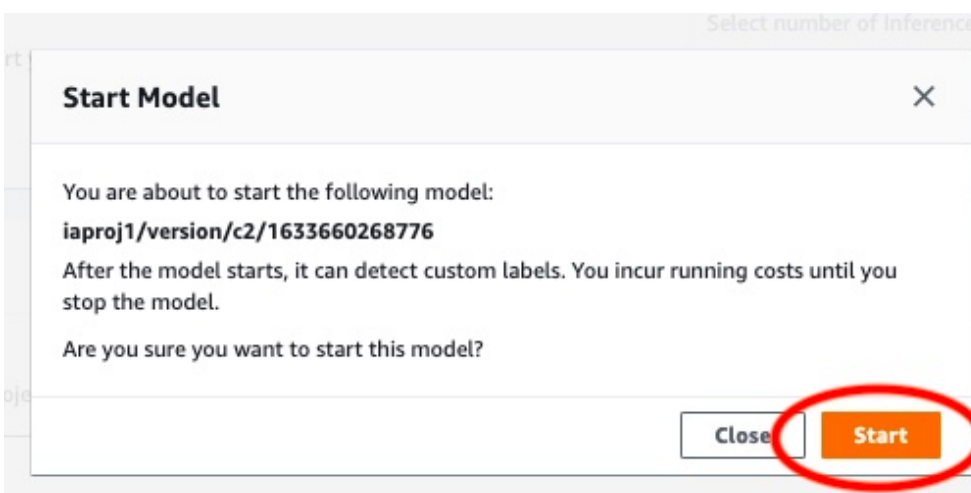
Anda dikenakan biaya untuk jumlah waktu yang dijalankan model Anda. Hentikan model Anda jika Anda tidak perlu menganalisis gambar. Anda dapat me-restart model Anda di lain waktu. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

Untuk memulai model Anda

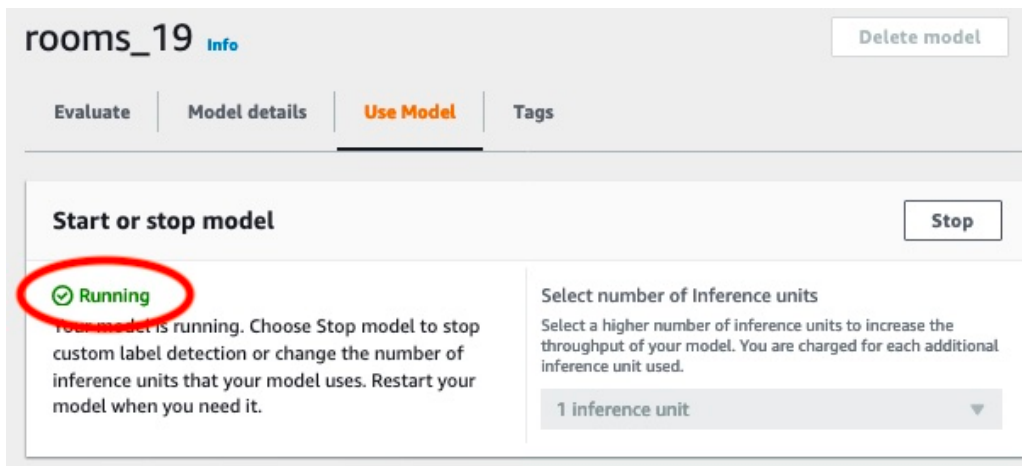
1. Pilih tab Gunakan model pada halaman model.
2. Di bagian Mulai atau hentikan model lakukan hal berikut:
  - a. Pilih Mulai.



- b. Dalam kotak dialog Mulai model, pilih Mulai.



3. Tunggu sampai model berjalan. Model berjalan ketika status di bagian Mulai atau berhenti model sedang berjalan.



## Langkah 9: Analisis gambar dengan model Anda

Anda menganalisis gambar dengan memanggil [DetectCustomLabels](#) API. Pada langkah ini, Anda menggunakan perintah `detect-custom-labels` AWS Command Line Interface (AWS CLI) untuk menganalisis contoh gambar. Anda mendapatkan AWS CLI perintah dari konsol Amazon Rekognition Custom Labels. Konsol mengonfigurasi AWS CLI perintah untuk menggunakan model Anda. Anda hanya perlu menyediakan gambar yang disimpan di bucket Amazon S3.

### Note

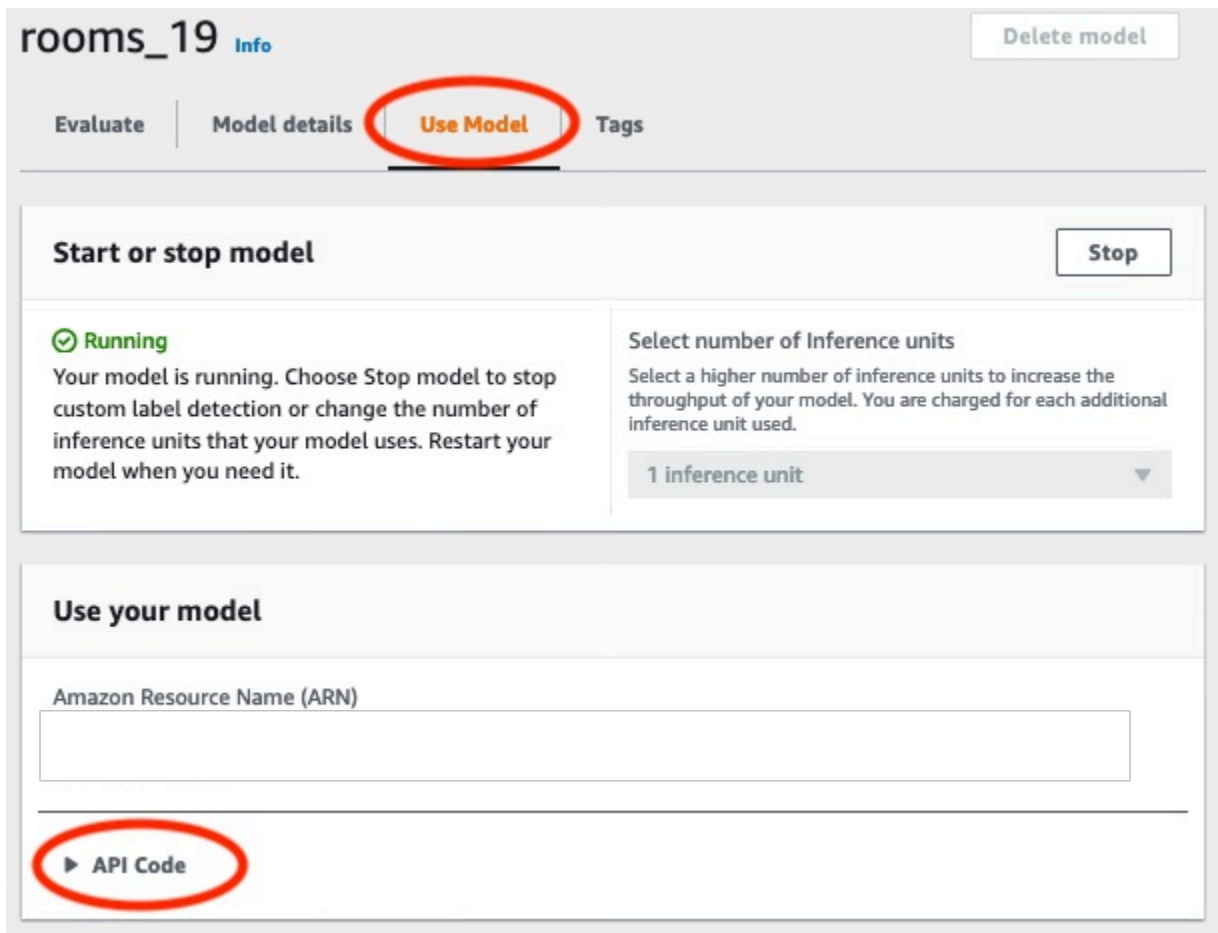
Konsol juga menyediakan kode contoh Python.

Output dari `detect-custom-labels` menyertakan daftar label yang ditemukan dalam gambar, kotak pembatas (jika model menemukan lokasi objek), dan keyakinan yang dimiliki model dalam keakuratan prediksi.

Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

Untuk menganalisis gambar (konsol)

1. Jika Anda belum melakukannya, atur AWS CLI. Untuk petunjuk, lihat [the section called “Langkah 4: Mengatur AWS CLI and AWS SDK”](#).
2. Pilih tab Use Model dan kemudian pilih kode API.



3. Pilih perintah AWS CLI.
4. Di bagian Analisis gambar, salin AWS CLI perintah yang memanggil detect-custom-labels.

**Use your model**

Amazon Resource Name (ARN)

▼ API Code

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

AWS CLI command

Python

**Start model**  
Command used to start the rooms\_ [ ] model.

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

**Analyze image**  
Command used to use analyze an image with the rooms\_ [ ] model. Replace MY\_BUCKET and PATH\_TO\_MY\_IMAGE with your S3 bucket name and image path.

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

- Unggah gambar ke bucket Amazon S3. Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Jika Anda menggunakan gambar dari proyek Rooms, gunakan salah satu gambar yang Anda pindahkan ke folder terpisah [Langkah 1: Kumpulkan gambar Anda](#).
- Pada prompt perintah, masukkan AWS CLI perintah yang Anda salin pada langkah sebelumnya. Seharusnya terlihat seperti contoh berikut.

Nilai `--project-version-arn` harus Amazon Resource Name (ARN) model Anda. Nilai `--region` harus AWS Wilayah di mana Anda membuat model.

Ubah `MY_BUCKET` dan `PATH_TO_MY_IMAGE` ke bucket Amazon S3 dan gambar yang Anda gunakan pada langkah sebelumnya.

Jika Anda menggunakan [custom-labels-access](#) profil untuk mendapatkan kredensial, tambahkan parameternya. `--profile custom-labels-access`

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

Output JSON dari AWS CLI perintah akan terlihat mirip dengan berikut ini. Name adalah nama label tingkat gambar yang ditemukan model. Confidence(0-100) adalah kepercayaan model dalam keakuratan prediksi.

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

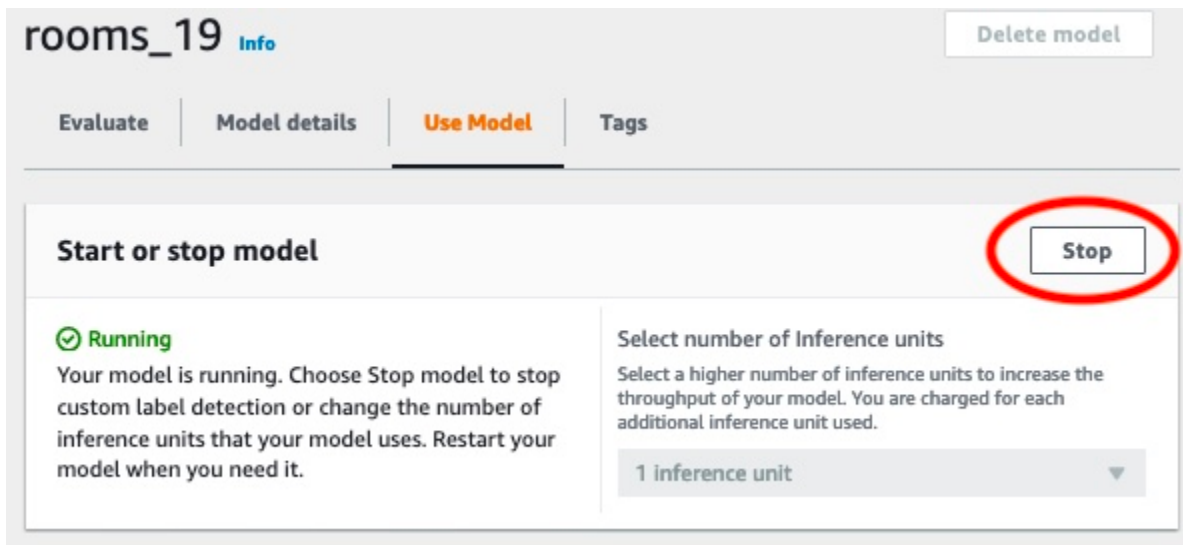
7. Terus gunakan model untuk menganalisis gambar lain. Hentikan model jika Anda tidak lagi menggunakannya.

## Langkah 10: Hentikan model Anda

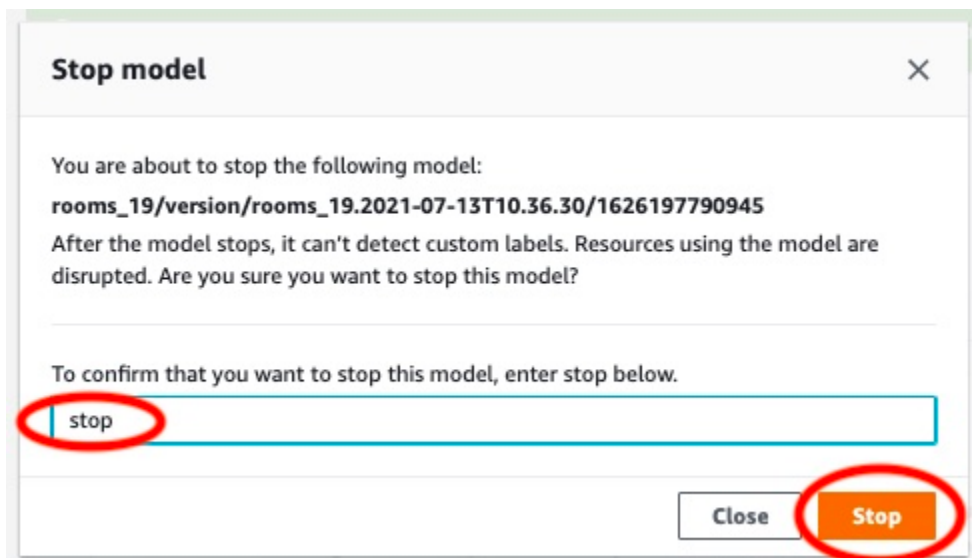
Pada langkah ini Anda berhenti menjalankan model Anda. Anda dikenakan biaya untuk jumlah waktu model Anda berjalan. Jika Anda telah selesai menggunakan model, Anda harus menghentikannya.

Untuk menghentikan model Anda

1. Di bagian Mulai atau hentikan model pilih Berhenti.



2. Dalam kotak dialog Stop model, masukkan stop untuk mengonfirmasi bahwa Anda ingin menghentikan model.



3. Pilih Berhenti untuk menghentikan model Anda. Model telah berhenti ketika status di bagian Mulai atau berhenti model Dihentikan.

rooms\_19 Info
Delete model

Evaluate
Model details
Use Model
Tags

---

### Start or stop model

⊖ Stopped

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

#### Select number of Inference units

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

1 inference unit
▼

Start

# Membuat model Label Kustom Rekognition Amazon

Model adalah perangkat lunak yang Anda latih untuk menemukan konsep, adegan, dan objek yang unik untuk bisnis Anda. Anda dapat membuat model dengan konsol Amazon Rekognition Custom Labels atau dengan SDK. AWS Sebelum membuat model Amazon Rekognition Custom Labels, kami sarankan Anda membaca [Memahami Label Kustom Rekognition Amazon](#)

Bagian ini menyediakan informasi konsol dan SDK tentang membuat proyek, membuat kumpulan data pelatihan dan pengujian untuk berbagai jenis model, dan melatih model. Bagian selanjutnya menunjukkan cara meningkatkan dan menggunakan model Anda. Untuk tutorial yang menunjukkan cara membuat dan menggunakan jenis model tertentu dengan konsol, lihat [Mengklasifikasikan gambar](#).

Topik

- [Membuat proyek](#)
- [Membuat kumpulan data pelatihan dan pengujian](#)
- [Melatih model Label Kustom Rekognition Amazon](#)
- [Mendebug pelatihan model yang gagal](#)

## Membuat proyek

Sebuah proyek mengelola versi model, kumpulan data pelatihan, dan kumpulan data pengujian untuk model. Anda dapat membuat project dengan konsol Amazon Rekognition Custom Labels atau dengan API. Untuk tugas proyek lainnya, seperti menghapus proyek, lihat [Mengelola proyek Label Kustom Rekognition Amazon](#).

Anda dapat menggunakan tag untuk mengkategorikan dan mengelola sumber daya Label Kustom Rekognition Amazon, termasuk proyek Anda.

[CreateProject](#) Operasi ini memungkinkan Anda menentukan tag secara opsional saat membuat proyek baru, memberikan Tag sebagai pasangan nilai kunci yang dapat Anda gunakan untuk mengkategorikan dan mengelola sumber daya Anda.

## Membuat Proyek Label Kustom Rekognition Amazon (Konsol)

Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk membuat proyek. Saat pertama kali menggunakan konsol di AWS Wilayah baru, Amazon Rekognition Custom Labels

meminta untuk membuat bucket Amazon S3 (bucket konsol) di akun Anda. AWS Bucket digunakan untuk menyimpan file proyek. Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon kecuali bucket konsol dibuat.

Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk membuat proyek.

Untuk membuat proyek (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan.
3. Halaman landing Label Kustom Rekognition Amazon, pilih Memulai.
4. Di panel kiri, Pilih Proyek.
5. Pilih Buat Proyek.
6. Dalam Nama proyek, masukkan nama untuk proyek Anda.
7. Pilih Buat proyek untuk membuat proyek Anda.
8. Ikuti langkah-langkah [Membuat kumpulan data pelatihan dan pengujian](#) untuk membuat kumpulan data pelatihan dan pengujian untuk proyek Anda.

## Membuat proyek Amazon Rekognition Custom Labels (SDK)

Anda membuat proyek Label Kustom Rekognition Amazon dengan menelepon. [CreateProject](#) Responsnya adalah Amazon Resource Name (ARN) yang mengidentifikasi proyek. Setelah membuat proyek, Anda membuat kumpulan data untuk pelatihan dan pengujian model. Untuk informasi selengkapnya, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#).

Untuk membuat proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk membuat proyek.

AWS CLI

Contoh berikut membuat proyek dan menampilkan ARN nya.

Ubah nilai `project-name` menjadi nama proyek yang ingin Anda buat.

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

## Python

Contoh berikut membuat proyek dan menampilkan ARN nya. Berikan argumen baris perintah berikut:

- `project_name`— nama proyek yang ingin Anda buat.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
import argparse  
import logging  
import boto3  
  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def create_project(rek_client, project_name):  
    """  
    Creates an Amazon Rekognition Custom Labels project  
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.  
    :param project_name: A name for the new prooject.  
    """  
  
    try:  
        #Create the project.  
        logger.info("Creating project: %s",project_name)  
  
        response=rek_client.create_project(ProjectName=project_name)  
  
        logger.info("project ARN: %s",response['ProjectArn'])  
  
        return response['ProjectArn']
```

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

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

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

def main():

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

    try:

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

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

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

        project_arn=create_project(rekognition_client,
            args.project_name)

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

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

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

## Java V2

Contoh berikut membuat proyek dan menampilkan ARN nya.

Berikan argumen baris perintah berikut:

- `project_name`— nama proyek yang ingin Anda buat.

```
/*  
 Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 SPDX-License-Identifier: Apache-2.0  
*/  
package com.example.rekognition;  
  
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rekognition.RekognitionClient;  
import software.amazon.awssdk.services.rekognition.model.CreateProjectRequest;  
import software.amazon.awssdk.services.rekognition.model.CreateProjectResponse;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class CreateProject {  
  
    public static final Logger logger =  
        Logger.getLogger(CreateProject.class.getName());  
  
    public static String createMyProject(RekognitionClient rekClient, String  
        projectName) {  
  
        try {  
  
            logger.log(Level.INFO, "Creating project: {0}", projectName);  
            CreateProjectRequest createProjectRequest =  
                CreateProjectRequest.builder().projectName(projectName).build();
```

```
        CreateProjectResponse response =
rekClient.createProject(createProjectRequest);

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

        return response.projectArn();

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

}

public static void main(String[] args) {

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

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

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

    try {

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

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

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

        rekClient.close();

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

}

}
```

3. Perhatikan nama proyek ARN yang ditampilkan dalam respons. Anda akan membutuhkannya untuk membuat model.
4. Ikuti langkah-langkah [Buat kumpulan data pelatihan dan pengujian \(SDK\)](#) untuk membuat kumpulan data pelatihan dan pengujian untuk proyek Anda.

## CreateProject permintaan operasi

Berikut ini adalah format permintaan CreateProject operasi:

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

## Membuat kumpulan data pelatihan dan pengujian

Dataset adalah sekumpulan gambar dan label yang menggambarkan gambar-gambar tersebut. Proyek Anda membutuhkan dataset pelatihan dan dataset pengujian. Amazon Rekognition Custom

Labels menggunakan dataset pelatihan untuk melatih model Anda. Setelah pelatihan, Amazon Rekognition Custom Labels menggunakan dataset pengujian untuk memverifikasi seberapa baik model terlatih memprediksi label yang benar.

Anda dapat membuat kumpulan data dengan konsol Amazon Rekognition Custom Labels atau dengan SDK. AWS Sebelum membuat kumpulan data, sebaiknya baca [Memahami Label Kustom Rekognition Amazon](#). Untuk tugas kumpulan data lainnya, lihat [Mengelola kumpulan data](#).

Langkah-langkah membuat kumpulan data pelatihan dan pengujian untuk sebuah proyek adalah:

Untuk membuat kumpulan data pelatihan dan pengujian untuk proyek Anda

1. Tentukan bagaimana Anda perlu memberi label kumpulan data pelatihan dan pengujian Anda. Untuk informasi selengkapnya, lihat [Mengarahkan kumpulan data](#).
2. Kumpulkan gambar untuk kumpulan data pelatihan dan uji Anda. Untuk informasi selengkapnya, lihat [the section called "Mempersiapkan gambar"](#).
3. Buat kumpulan data pelatihan dan uji. Untuk informasi selengkapnya, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#). Jika Anda menggunakan AWS SDK, lihat [Buat kumpulan data pelatihan dan pengujian \(SDK\)](#).
4. Jika perlu, tambahkan label tingkat gambar atau kotak pembatas ke gambar kumpulan data Anda. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#).

Setelah Anda membuat kumpulan data, Anda dapat [melatih model](#).

Topik

- [Mengarahkan kumpulan data](#)
- [Mempersiapkan gambar](#)
- [Membuat kumpulan data pelatihan dan uji dengan gambar](#)
- [Pelabelan gambar](#)
- [Debugging kumpulan data](#)

## Mengarahkan kumpulan data

Cara Anda memberi label pada kumpulan data pelatihan dan pengujian dalam proyek Anda menentukan jenis model yang Anda buat. Dengan Amazon Rekognition Custom Labels Anda dapat membuat model yang melakukan hal berikut.

- [Temukan objek, adegan, dan konsep](#)
- [Temukan lokasi objek](#)
- [Temukan lokasi merek](#)

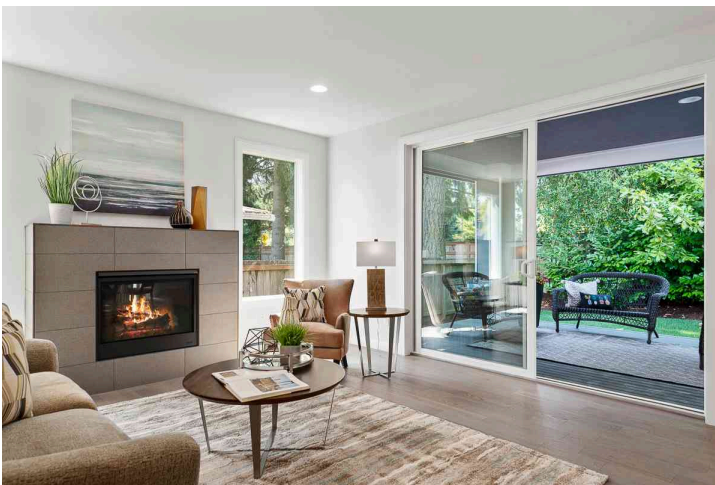
## Temukan objek, adegan, dan konsep

Model mengklasifikasikan objek, adegan, dan konsep yang terkait dengan keseluruhan gambar.

Anda dapat membuat dua jenis model klasifikasi, klasifikasi gambar dan klasifikasi multi-label. Untuk kedua jenis model klasifikasi, model menemukan satu atau lebih label yang cocok dari set lengkap label yang digunakan untuk pelatihan. Kumpulan data pelatihan dan pengujian keduanya membutuhkan setidaknya dua label.

### Klasifikasi gambar

Model mengklasifikasikan gambar sebagai milik satu set label yang telah ditentukan. Misalnya, Anda mungkin menginginkan model yang menentukan apakah gambar berisi ruang hidup. Gambar berikut mungkin memiliki label tingkat gambar `living_space`.



Untuk jenis model ini, tambahkan satu label tingkat gambar ke setiap gambar kumpulan data pelatihan dan uji. Untuk contoh proyek, lihat [Klasifikasi gambar](#).

### Klasifikasi multi-label

Model ini mengklasifikasikan gambar ke dalam beberapa kategori, seperti jenis bunga dan apakah memiliki daun, atau tidak. Misalnya, gambar berikut mungkin memiliki label level gambar `mediterranean_spurge` dan `no_leaves`.



Untuk jenis model ini, tetapkan label tingkat gambar untuk setiap kategori ke gambar kumpulan data pelatihan dan uji. Untuk contoh proyek, lihat [Klasifikasi gambar multi-label](#).

### Menetapkan label tingkat gambar

Jika gambar Anda disimpan dalam bucket Amazon S3, Anda dapat menggunakan [nama folder](#) untuk menambahkan label tingkat gambar secara otomatis. Untuk informasi selengkapnya, lihat [Mengimpor gambar dari bucket Amazon S3](#). Anda juga dapat menambahkan label tingkat gambar ke gambar setelah membuat kumpulan data, Untuk informasi selengkapnya, lihat [the section called “Menetapkan label tingkat gambar ke gambar”](#) Anda dapat menambahkan label baru saat Anda membutuhkannya. Untuk informasi selengkapnya, lihat [Mengelola label](#).

### Temukan lokasi objek

Untuk membuat model yang memprediksi lokasi objek dalam gambar Anda, Anda menentukan kotak pembatas lokasi objek dan label untuk gambar dalam kumpulan data pelatihan dan pengujian Anda. Kotak pembatas adalah kotak yang mengelilingi objek dengan erat. Misalnya, gambar berikut menunjukkan kotak pembatas di sekitar Amazon Echo dan Amazon Echo Dot. Setiap kotak pembatas memiliki label yang ditetapkan (Amazon Echo atau Amazon Echo Dot).



Untuk menemukan lokasi objek, kumpulan data Anda membutuhkan setidaknya satu label. Selama pelatihan model, label lebih lanjut secara otomatis dibuat yang mewakili area di luar kotak pembatas pada gambar.

### Menetapkan kotak pembatas

Saat membuat kumpulan data, Anda dapat menyertakan informasi kotak pembatas untuk gambar Anda. Misalnya, Anda dapat mengimpor [file manifest](#) format SageMaker AI Ground Truth yang berisi kotak pembatas. Anda juga dapat menambahkan kotak pembatas setelah membuat kumpulan data. Untuk informasi selengkapnya, lihat [Pelabelan objek dengan kotak pembatas](#). Anda dapat menambahkan label baru saat Anda membutuhkannya. Untuk informasi selengkapnya, lihat [Mengelola label](#).

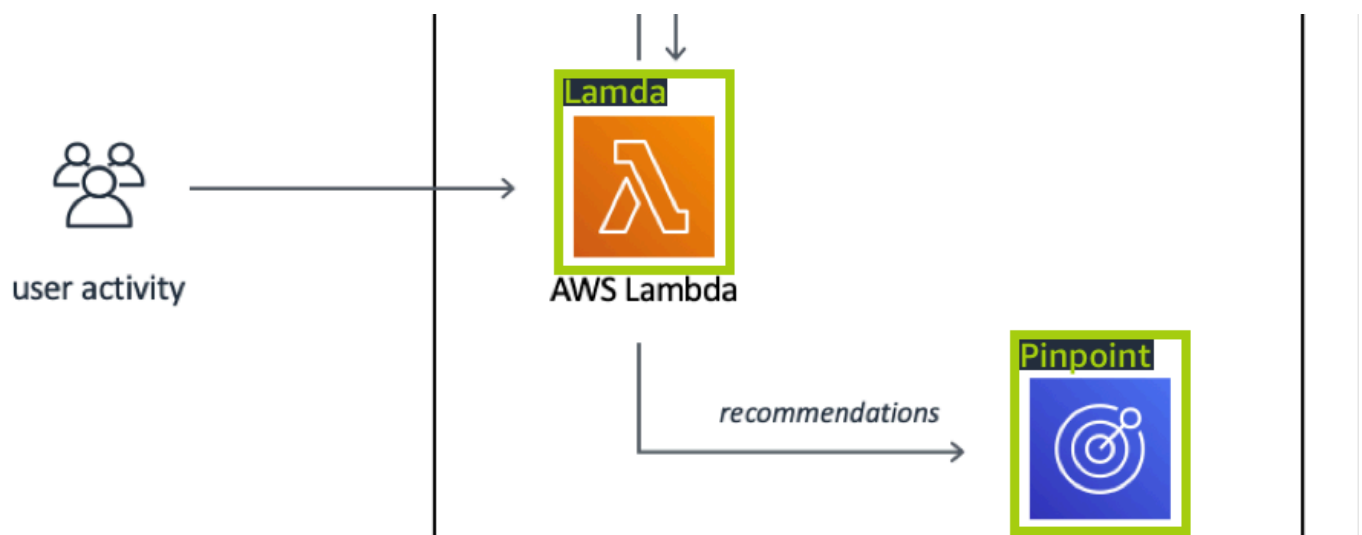
### Temukan lokasi merek

Jika Anda ingin menemukan lokasi merek, seperti logo dan karakter animasi, Anda dapat menggunakan dua jenis gambar yang berbeda untuk gambar kumpulan data pelatihan Anda.

- Gambar yang hanya dari logo. Setiap gambar membutuhkan label tingkat gambar tunggal yang mewakili nama logo. Misalnya, label tingkat gambar untuk gambar berikut dapat berupa Lambda.



- Gambar yang berisi logo di lokasi alami, seperti pertandingan sepak bola atau diagram arsitektural. Setiap gambar pelatihan membutuhkan kotak pembatas yang mengelilingi setiap instance logo. Misalnya, gambar berikut menunjukkan diagram arsitektur dengan kotak pembatas berlabel yang mengelilingi logo AWS Lambda dan Amazon Pinpoint.



Kami menyarankan Anda untuk tidak mencampur label tingkat gambar dan kotak pembatas dalam gambar pelatihan Anda.

Gambar uji harus memiliki kotak pembatas di sekitar contoh merek yang ingin Anda temukan. Anda dapat membagi kumpulan data pelatihan untuk membuat kumpulan data pengujian, hanya jika gambar pelatihan menyertakan kotak pembatas berlabel. Jika gambar pelatihan hanya memiliki label tingkat gambar, Anda harus membuat kumpulan data pengujian yang menyertakan gambar dengan kotak pembatas berlabel. Jika Anda melatih model untuk menemukan lokasi merek, lakukan

[Pelabelan objek dengan kotak pembatas](#) dan [Menetapkan label tingkat gambar ke gambar](#) sesuai dengan cara Anda memberi label pada gambar Anda.

Proyek [Deteksi merek](#) contoh menunjukkan bagaimana Amazon Rekognition Custom Labels menggunakan kotak pembatas berlabel untuk melatih model yang menemukan lokasi objek.

## Persyaratan label untuk jenis model

Gunakan tabel berikut untuk menentukan cara memberi label pada gambar Anda.

Anda dapat menggabungkan label tingkat gambar dan gambar berlabel kotak pembatas dalam satu kumpulan data. Dalam hal ini, Amazon Rekognition Custom Labels memilih apakah akan membuat model tingkat gambar atau model lokasi objek.

Contoh	Gambar pelatihan	Gambar uji
<a href="#">Klasifikasi gambar</a>	1 Label tingkat gambar per gambar	1 Label tingkat gambar per gambar
<a href="#">Klasifikasi multi-label</a>	Beberapa label tingkat gambar per gambar	Beberapa label tingkat gambar per gambar
<a href="#">Temukan lokasi merek</a>	label tingkat gambar (Anda juga dapat menggunakan kotak pembatas berlabel)	Kotak pembatas berlabel
<a href="#">Temukan lokasi objek</a>	Kotak pembatas berlabel	Kotak pembatas berlabel

## Mempersiapkan gambar

Gambar dalam kumpulan data pelatihan dan pengujian Anda berisi objek, adegan, atau konsep yang Anda ingin model Anda temukan.

Isi gambar harus dalam berbagai latar belakang dan pencahayaan yang mewakili gambar yang Anda inginkan untuk diidentifikasi oleh model terlatih.

Bagian ini memberikan informasi tentang gambar dalam kumpulan data pelatihan dan pengujian Anda.

## Format gambar

Anda dapat melatih model Amazon Rekognition Custom Labels dengan gambar yang dalam format PNG dan JPEG. Demikian pula, untuk mendeteksi label khusus menggunakan `DetectCustomLabels`, Anda memerlukan gambar yang dalam format PNG dan JPEG.

## Rekomendasi gambar masukan

Amazon Rekognition Custom Labels memerlukan gambar untuk melatih dan menguji model Anda. Untuk menyiapkan gambar Anda, pertimbangkan sebagai berikut:

- Pilih domain tertentu untuk model yang ingin Anda buat. Misalnya, Anda dapat memilih model untuk pemandangan indah dan model lain untuk objek seperti suku cadang mesin. Label Kustom Rekognition Amazon berfungsi paling baik jika gambar Anda berada di domain yang dipilih.
- Gunakan setidaknya 10 gambar untuk melatih model Anda.
- Gambar harus dalam format PNG atau JPEG.
- Gunakan gambar yang menunjukkan objek dalam berbagai pencahayaan, latar belakang, dan resolusi.
- Pelatihan dan pengujian gambar harus serupa dengan gambar yang ingin Anda gunakan modelnya.
- Tentukan label apa yang akan ditetapkan ke gambar.
- Pastikan resolusi citra cukup besar. Untuk informasi selengkapnya, lihat [Pedoman dan kuota di Label Kustom Rekognition Amazon](#).
- Pastikan oklusi tidak mengaburkan objek yang ingin Anda deteksi.
- Gunakan citra yang memiliki kontras yang cukup dengan latar belakang.
- Gunakan citra yang terang dan tajam. Hindari penggunaan gambar yang mungkin buram karena subjek dan gerakan kamera sebanyak mungkin.
- Gunakan gambar di mana objek menempati sebagian besar gambar.
- Gambar dalam kumpulan data pengujian Anda tidak boleh berupa gambar yang ada dalam kumpulan data pelatihan. Mereka harus mencakup objek, adegan, dan konsep yang dilatih untuk dianalisis oleh model.

## Ukuran set gambar

Amazon Rekognition Custom Labels menggunakan satu set gambar untuk melatih model. Minimal, Anda harus menggunakan setidaknya 10 gambar untuk pelatihan. Amazon Rekognition Custom Labels menyimpan gambar pelatihan dan pengujian dalam kumpulan data. Untuk informasi selengkapnya, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#).

## Membuat kumpulan data pelatihan dan uji dengan gambar

Anda dapat memulai dengan proyek yang memiliki satu kumpulan data, atau proyek yang memiliki kumpulan data pelatihan dan pengujian terpisah. Jika Anda memulai dengan satu kumpulan data, Amazon Rekognition Custom Labels membagi kumpulan data Anda selama pelatihan untuk membuat kumpulan data pelatihan (80%) dan kumpulan data pengujian (% 20) untuk proyek Anda. Mulailah dengan satu kumpulan data jika Anda ingin Label Kustom Rekognition Amazon memutuskan di mana gambar digunakan untuk pelatihan dan pengujian. Untuk kontrol penuh atas pelatihan, pengujian, dan penyetelan kinerja, kami menyarankan Anda memulai proyek Anda dengan kumpulan data pelatihan dan pengujian terpisah.

Anda dapat membuat kumpulan data pelatihan dan pengujian untuk proyek dengan mengimpor gambar dari salah satu lokasi berikut:

- [Mengimpor gambar dari bucket Amazon S3](#)
- [Mengimpor gambar dari komputer lokal](#)
- [Menggunakan file manifes untuk mengimpor gambar](#)
- [Menyalin konten dari kumpulan data yang ada](#)

Jika Anda memulai proyek dengan kumpulan data pelatihan dan pengujian terpisah, Anda dapat menggunakan lokasi sumber yang berbeda untuk setiap kumpulan data.

Tergantung dari mana Anda mengimpor gambar, gambar Anda mungkin tidak berlabel. Misalnya, gambar yang diimpor dari komputer lokal tidak diberi label. Gambar yang diimpor dari file manifes Amazon SageMaker AI Ground Truth diberi label. Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk menambahkan, mengubah, dan menetapkan label. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#).

Jika gambar diunggah dengan kesalahan, gambar hilang, atau label hilang dari gambar, baca [Mendebug pelatihan model yang gagal](#).

Untuk informasi selengkapnya tentang kumpulan data, lihat [Mengelola kumpulan data](#)

## Buat kumpulan data pelatihan dan pengujian (SDK)

Anda dapat menggunakan AWS SDK untuk membuat kumpulan data pelatihan dan pengujian.

`CreateDatasetOperasi` ini memungkinkan Anda menentukan tag secara opsional saat membuat kumpulan data baru, untuk tujuan mengkategorikan dan mengelola sumber daya Anda.

### Dataset pelatihan

Anda dapat menggunakan AWS SDK untuk membuat kumpulan data pelatihan dengan cara berikut.

- Gunakan [CreateDataset](#) dengan file manifes format Amazon Sagemaker yang Anda berikan. Untuk informasi selengkapnya, lihat [the section called “Membuat file manifes”](#). Untuk kode sampel, lihat [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).
- Gunakan `CreateDataset` untuk menyalin kumpulan data Label Kustom Amazon Rekognition yang ada. Untuk kode sampel, lihat [Membuat kumpulan data menggunakan dataset \(SDK\) yang ada](#).
- Buat dataset kosong dengan `CreateDataset` dan tambahkan entri dataset di lain waktu dengan [UpdateDatasetEntries](#) Untuk membuat kumpulan data kosong, lihat [Menambahkan kumpulan data ke proyek](#). Untuk menambahkan gambar ke kumpulan data, lihat [Menambahkan lebih banyak gambar \(SDK\)](#). Anda perlu menambahkan entri dataset sebelum Anda dapat melatih model.

### Dataset uji

Anda dapat menggunakan AWS SDK untuk membuat kumpulan data pengujian dengan cara berikut:

- Gunakan [CreateDataset](#) dengan file manifes format Amazon Sagemaker yang Anda berikan. Untuk informasi selengkapnya, lihat [the section called “Membuat file manifes”](#). Untuk kode sampel, lihat [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).
- Gunakan `CreateDataset` untuk menyalin kumpulan data Label Kustom Amazon Rekognition yang ada. Untuk kode sampel, lihat [Membuat kumpulan data menggunakan dataset \(SDK\) yang ada](#).
- Buat dataset kosong dengan `CreateDataset` dan tambahkan entri dataset di lain waktu dengan [UpdateDatasetEntries](#) Untuk membuat kumpulan data kosong, lihat [Menambahkan kumpulan data ke proyek](#). Untuk menambahkan gambar ke kumpulan data, lihat [Menambahkan lebih banyak gambar \(SDK\)](#). Anda perlu menambahkan entri dataset sebelum Anda dapat melatih model.
- Pisahkan kumpulan data pelatihan menjadi kumpulan data pelatihan dan pengujian terpisah. Pertama buat dataset pengujian kosong dengan `CreateDataset`. Kemudian pindahkan

20% entri kumpulan data pelatihan ke dalam kumpulan data pengujian dengan menelepon.

[DistributeDatasetEntries](#) Untuk membuat kumpulan data kosong, lihat [Menambahkan kumpulan data ke proyek \(SDK\)](#). Untuk membagi kumpulan data pelatihan, lihat [Mendistribusikan dataset pelatihan \(SDK\)](#).

## Mengimpor gambar dari bucket Amazon S3

Gambar diimpor dari ember Amazon S3. Anda dapat menggunakan bucket konsol, atau bucket Amazon S3 lainnya di akun Anda AWS . Jika Anda menggunakan bucket konsol, izin yang diperlukan sudah diatur. Jika Anda tidak menggunakan bucket konsol, lihat [Mengakses Bucket Amazon S3 eksternal](#).

### Note

Anda tidak dapat menggunakan AWS SDK untuk membuat kumpulan data langsung dari gambar di bucket Amazon S3. Sebagai gantinya, buat file manifes yang mereferensikan lokasi sumber gambar. Untuk informasi selengkapnya, lihat [Menggunakan file manifes untuk mengimpor gambar](#)

Selama pembuatan kumpulan data, Anda dapat memilih untuk menetapkan nama label ke gambar berdasarkan nama folder yang berisi gambar. Folder harus merupakan turunan dari jalur folder Amazon S3 yang Anda tentukan di lokasi folder S3 selama pembuatan kumpulan data. Untuk membuat kumpulan data, lihat [Membuat kumpulan data dengan mengimpor gambar dari bucket S3](#).

Misalnya, asumsikan struktur folder berikut di bucket Amazon S3. Jika Anda menentukan lokasi folder Amazon S3 sebagai S3-bucket/Alexa-devices, gambar di folder echo diberi label echo. Demikian pula, gambar di folder echo-dot diberi label echo-dot. Nama-nama folder anak yang lebih dalam tidak digunakan untuk memberi label pada gambar. Sebagai gantinya, folder anak yang sesuai dari lokasi folder Amazon S3 digunakan. Misalnya, gambar dalam folder white-echo-dotsdiberi label echo-dot. Gambar di tingkat lokasi folder S3 (alexa-devices) tidak memiliki label yang ditetapkan padanya.

Folder yang lebih dalam dalam struktur folder dapat digunakan untuk memberi label gambar dengan menentukan lokasi folder S3 yang lebih dalam. Misalnya, Jika Anda menentukan S3- bucket/alexa-devices/echo -dot, Gambar dalam folder white-echo-dotdiberi label. white-echo-dot Gambar di luar lokasi folder s3 yang ditentukan, seperti echo, tidak diimpor.

S3-bucket

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

Sebaiknya gunakan bucket Amazon S3 (bucket konsol) yang dibuat untuk Anda oleh Amazon Rekognition saat pertama kali membuka konsol di wilayah saat ini. AWS Jika bucket Amazon S3 yang Anda gunakan berbeda (eksternal) dengan bucket konsol, konsol akan meminta Anda untuk menyiapkan izin yang sesuai selama pembuatan kumpulan data. Untuk informasi selengkapnya, lihat [the section called “Langkah 2: Siapkan izin konsol”](#).

Membuat kumpulan data dengan mengimpor gambar dari bucket S3

Prosedur berikut menunjukkan cara membuat kumpulan data menggunakan gambar yang disimpan di bucket Console S3. Gambar secara otomatis diberi label dengan nama folder tempat mereka disimpan.

Setelah mengimpor gambar, Anda dapat menambahkan lebih banyak gambar, menetapkan label, dan menambahkan kotak pembatas dari halaman galeri kumpulan data. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#).

Unggah gambar Anda ke bucket Amazon Simple Storage Service

1. Buat folder di sistem file lokal Anda. Gunakan nama folder seperti alexa-devices.
2. Di dalam folder yang baru saja Anda buat, buat folder yang diberi nama setelah setiap label yang ingin Anda gunakan. Misalnya, echo dan echo-dot. Struktur folder harus mirip dengan yang berikut ini.

```
alex-a-devices
```

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

3. Tempatkan gambar yang sesuai dengan label ke dalam folder dengan nama label yang sama.
4. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon S3 di <https://console.aws.amazon.com/s3/>
5. [Tambahkan folder yang](#) Anda buat di langkah 1 ke bucket Amazon S3 (bucket konsol) yang dibuat untuk Anda oleh Label Kustom Rekognition Amazon selama Penyiapan Pertama Kali. Untuk informasi selengkapnya, lihat [Mengelola proyek Label Kustom Rekognition Amazon](#).
6. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
7. Pilih Gunakan Label Kustom.
8. Pilih Mulai.
9. Di panel navigasi kiri, pilih Proyek.
10. Di halaman Proyek, pilih proyek yang ingin Anda tambahkan dataset. Halaman detail untuk proyek Anda ditampilkan.
11. Pilih Buat kumpulan data. Halaman Buat dataset ditampilkan.
12. Dalam konfigurasi Mulai, pilih Mulai dengan satu set data atau Mulai dengan kumpulan data pelatihan. Untuk membuat model berkualitas lebih tinggi, kami sarankan memulai dengan kumpulan data pelatihan dan pengujian terpisah.

### Single dataset

- a. Di bagian Detail kumpulan data Pelatihan, pilih Impor gambar dari bucket S3.
- b. Di bagian Detail kumpulan data pelatihan, Masukkan informasi untuk langkah 13 - 15 di bagian Konfigurasi sumber gambar.

### Separate training and test datasets

- a. Di bagian Detail kumpulan data Pelatihan, pilih Impor gambar dari bucket S3.

- b. Di bagian Detail kumpulan data Pelatihan, masukkan informasi untuk langkah 13 - 15 di bagian Konfigurasi sumber gambar.
  - c. Di bagian Uji detail kumpulan data, pilih Impor gambar dari ember S3.
  - d. Di bagian Uji detail kumpulan data, masukkan informasi untuk langkah 13 - 15 di bagian Konfigurasi sumber gambar.
13. Pilih Impor gambar dari ember Amazon S3.
  14. Di URI S3, masukkan lokasi bucket Amazon S3 dan jalur folder.
  15. Pilih Secara otomatis melampirkan label ke gambar berdasarkan folder.
  16. Pilih Buat Kumpulan Data. Halaman kumpulan data untuk proyek Anda terbuka.
  17. Jika Anda perlu menambahkan atau mengubah label, lakukan [Pelabelan gambar](#).
  18. Ikuti langkah-langkah [Melatih model \(Konsol\)](#) untuk melatih model Anda.

## Mengimpor gambar dari komputer lokal

Gambar dimuat langsung dari komputer Anda. Anda dapat mengunggah hingga 30 gambar sekaligus.

Gambar yang Anda unggah tidak akan memiliki label yang terkait dengannya. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#). Jika Anda memiliki banyak gambar untuk diunggah, pertimbangkan untuk menggunakan bucket Amazon S3. Untuk informasi selengkapnya, lihat [Mengimpor gambar dari bucket Amazon S3](#).

### Note

Anda tidak dapat menggunakan AWS SDK untuk membuat kumpulan data dengan gambar lokal. Sebagai gantinya, buat file manifes dan unggah gambar ke bucket Amazon S3. Untuk informasi selengkapnya, lihat [Menggunakan file manifes untuk mengimpor gambar](#).

Untuk membuat kumpulan data menggunakan gambar di komputer lokal (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.

5. Di halaman Proyek, pilih proyek yang ingin Anda tambahkan dataset. Halaman detail untuk proyek Anda ditampilkan.
6. Pilih Buat kumpulan data. Halaman Buat dataset ditampilkan.
7. Dalam konfigurasi Mulai, pilih Mulai dengan satu set data atau Mulai dengan kumpulan data pelatihan. Untuk membuat model berkualitas lebih tinggi, kami sarankan memulai dengan kumpulan data pelatihan dan pengujian terpisah.

#### Single dataset

- a. Di bagian Detail kumpulan data pelatihan, pilih Unggah gambar dari komputer Anda.
- b. Pilih Buat Dataset.
- c. Pada halaman kumpulan data proyek, pilih Tambahkan gambar.
- d. Pilih gambar yang ingin Anda unggah ke kumpulan data dari file komputer Anda. Anda dapat menyeret gambar atau memilih gambar yang ingin Anda unggah dari komputer lokal Anda.
- e. Pilih Unggah gambar.

#### Separate training and test datasets

- a. Di bagian Detail kumpulan data pelatihan, pilih Unggah gambar dari komputer Anda.
- b. Di bagian Uji detail kumpulan data, pilih Unggah gambar dari komputer Anda.

#### Note

Kumpulan data pelatihan dan pengujian Anda dapat memiliki sumber gambar yang berbeda.

- c. Pilih Buat Kumpulan Data. Halaman kumpulan data proyek Anda muncul dengan tab Pelatihan dan tab Uji untuk kumpulan data masing-masing.
- d. Pilih Tindakan, lalu pilih Tambahkan gambar ke kumpulan data pelatihan.
- e. Pilih gambar yang ingin Anda unggah ke kumpulan data. Anda dapat menyeret gambar atau memilih gambar yang ingin Anda unggah dari komputer lokal Anda.
- f. Pilih Unggah gambar.
- g. Ulangi langkah 5e - 5g. Untuk langkah 5e, pilih Tindakan dan kemudian pilih Tambahkan gambar untuk menguji kumpulan data.

8. Ikuti langkah-langkah [Pelabelan gambar](#) untuk memberi label pada gambar Anda.
9. Ikuti langkah-langkah [Melatih model \(Konsol\)](#) untuk melatih model Anda.

## Menggunakan file manifes untuk mengimpor gambar

Anda dapat membuat kumpulan data menggunakan file manifes format Amazon SageMaker AI Ground Truth. Anda dapat menggunakan file manifes dari pekerjaan Amazon SageMaker AI Ground Truth. Jika gambar dan label Anda tidak dalam format file manifes SageMaker AI Ground Truth, Anda dapat membuat file manifes format SageMaker AI dan menggunakannya untuk mengimpor gambar berlabel.

CreateDatasetOperasi diperbarui untuk memungkinkan Anda menentukan tag secara opsional saat membuat kumpulan data baru. Tag adalah pasangan nilai kunci yang dapat Anda gunakan untuk mengkategorikan dan mengelola sumber daya Anda.

### Topik

- [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(Console\)](#)
- [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#)
- [Buat permintaan dataset](#)
- [Memberi label gambar dengan pekerjaan Amazon SageMaker AI Ground Truth](#)
- [Membuat file manifes](#)
- [Mengimpor label tingkat gambar dalam file manifes](#)
- [Lokalisasi objek dalam file manifes](#)
- [Aturan validasi untuk file manifes](#)
- [Mengonversi format dataset lain ke file manifes](#)

### Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth (Console)

Prosedur berikut menunjukkan cara membuat kumpulan data dengan menggunakan file manifes format SageMaker AI Ground Truth.

1. Buat file manifes untuk kumpulan data pelatihan Anda dengan melakukan salah satu hal berikut:
  - Buat file manifes dengan SageMaker AI GroundTruth Job dengan mengikuti petunjuk di [Memberi label gambar dengan pekerjaan Amazon SageMaker AI Ground Truth](#).

- Buat file manifes Anda sendiri dengan mengikuti petunjuk di [Membuat file manifes](#).

Jika Anda ingin membuat kumpulan data pengujian, ulangi langkah 1 untuk membuat kumpulan data pengujian.

2. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
3. Pilih Gunakan Label Kustom.
4. Pilih Mulai.
5. Di panel navigasi kiri, pilih Proyek.
6. Di halaman Proyek, pilih proyek yang ingin Anda tambahkan dataset. Halaman detail untuk proyek Anda ditampilkan.
7. Pilih Buat kumpulan data. Halaman Buat dataset ditampilkan.
8. Dalam konfigurasi Mulai, pilih Mulai dengan satu set data atau Mulai dengan kumpulan data pelatihan. Untuk membuat model berkualitas lebih tinggi, kami sarankan memulai dengan kumpulan data pelatihan dan pengujian terpisah.

#### Single dataset

- a. Di bagian Detail kumpulan data Pelatihan, pilih Impor gambar berlabel SageMaker Ground Truth.
- b. Di lokasi file.manifest masukkan lokasi file manifes yang Anda buat di langkah 1.
- c. Pilih Buat Dataset. Halaman kumpulan data untuk proyek Anda terbuka.

#### Separate training and test datasets

- a. Di bagian Detail kumpulan data Pelatihan, pilih Impor gambar berlabel SageMaker Ground Truth.
- b. Di lokasi file.manifest masukkan lokasi file manifes kumpulan data pelatihan yang Anda buat di langkah 1.
- c. Di bagian Test dataset details, pilih Impor gambar berlabel SageMaker Ground Truth.

#### Note

Kumpulan data pelatihan dan pengujian Anda dapat memiliki sumber gambar yang berbeda.

- d. Di lokasi file.manifest masukkan lokasi file manifest kumpulan data pengujian yang Anda buat di langkah 1.
  - e. Pilih Buat Kumpulan Data. Halaman kumpulan data untuk proyek Anda terbuka.
9. Jika Anda perlu menambahkan atau mengubah label, lakukan [Pelabelan gambar](#).
  10. Ikuti langkah-langkah [Melatih model \(Konsol\)](#) untuk melatih model Anda.

## Membuat kumpulan data dengan file manifest SageMaker AI Ground Truth (SDK)

Prosedur berikut menunjukkan cara membuat kumpulan data pelatihan atau pengujian dari file manifest menggunakan API. [CreateDataset](#)

Anda dapat menggunakan file manifest yang ada, seperti output dari [pekerjaan SageMaker AI Ground Truth](#), atau membuat [file manifest](#) Anda sendiri.

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Buat file manifest untuk kumpulan data pelatihan Anda dengan melakukan salah satu hal berikut:
  - Buat file manifest dengan SageMaker AI GroundTruth Job dengan mengikuti petunjuk di [Memberi label gambar dengan pekerjaan Amazon SageMaker AI Ground Truth](#).
  - Buat file manifest Anda sendiri dengan mengikuti petunjuk di [Membuat file manifest](#).

Jika Anda ingin membuat kumpulan data pengujian, ulangi langkah 2 untuk membuat kumpulan data pengujian.

3. Gunakan kode contoh berikut untuk membuat kumpulan data pelatihan dan pengujian.

### AWS CLI

Gunakan kode berikut untuk membuat kumpulan data. Ganti yang berikut ini:

- `project_arn`— ARN dari proyek yang ingin Anda tambahkan dataset pengujian.
- `type`— jenis dataset yang ingin Anda buat (TRAIN atau TEST)
- `bucket`— bucket yang berisi file manifest untuk dataset.
- `manifest_file`— jalur dan nama file dari file manifest.

```
aws rekognition create-dataset --project-arn project_arn \
```

```
--dataset-type type \
--dataset-source '{ "GroundTruthManifest": { "S3object": { "Bucket": "bucket",
"Name": "manifest_file" } } }' \
--profile custom-labels-access
--tags '{"key1": "value1", "key2": "value2"}'
```

## Python

Gunakan nilai berikut untuk membuat kumpulan data. Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda tambahkan dataset pengujian.
- `dataset_type`— jenis dataset yang ingin Anda buat (`trainatautest`).
- `bucket`— bucket yang berisi file manifes untuk dataset.
- `manifest_file`— jalur dan nama file dari file manifes.

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

import argparse
import logging
import time
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

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

```
:param manifest_file: The path and filename of the manifest file.
"""

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

    dataset_type = dataset_type.upper()

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

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

    dataset_arn=response['DatasetArn']

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

    finished=False
    while finished is False:

        dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

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

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

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

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

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

    return dataset_arn

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

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

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

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

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

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

```
def main():

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

    try:

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

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

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

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

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

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

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

## Java V2

Gunakan nilai berikut untuk membuat kumpulan data. Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda tambahkan dataset pengujian.

- `dataset_type`— jenis dataset yang ingin Anda buat (`trainatautest`).
- `bucket`— bucket yang berisi file manifes untuk dataset.
- `manifest_file`— jalur dan nama file dari file manifes.

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;

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

public class CreateDatasetManifestFiles {

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

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

        try {
```

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

        DatasetType requestDatasetType = null;

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

        GroundTruthManifest groundTruthManifest =
GroundTruthManifest.builder()

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

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

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

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

        CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

        boolean created = false;

        do {

            DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()

                .datasetArn(response.datasetArn()).build();
```

```
        DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        DatasetStatus status = datasetDescription.status();

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

        switch (status) {

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

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

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

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

    } while (created == false);

    return response.datasetArn();

} catch (RekognitionException e) {
```

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

}

public static void main(String[] args) {

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

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

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

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

    try {

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

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

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

        rekClient.close();

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

4. Jika Anda perlu menambahkan atau mengubah label, lihat [Mengelola Label \(SDK\)](#).
5. Ikuti langkah-langkah [Melatih model \(SDK\)](#) untuk melatih model Anda.

Buat permintaan dataset

Berikut ini adalah format permintaan CreateDataset operasi:

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

```
"DatasetType": "string",
"ProjectArn": "string",
"Tags": {
  "string": "string"
}
}
```

## Memberi label gambar dengan pekerjaan Amazon SageMaker AI Ground Truth

Dengan Amazon SageMaker AI Ground Truth, Anda dapat menggunakan pekerja dari Amazon Mechanical Turk, perusahaan vendor yang Anda pilih, atau tenaga kerja pribadi internal bersama dengan pembelajaran mesin yang memungkinkan Anda membuat kumpulan gambar berlabel. Amazon Rekognition Custom Labels mengimpor file manifes SageMaker AI Ground Truth dari bucket Amazon S3 yang Anda tentukan.

Amazon Rekognition Custom Labels mendukung tugas AI SageMaker Ground Truth berikut.

- [Klasifikasi Gambar](#)
- [Kotak Bounding](#)

File yang Anda impor adalah gambar dan file manifes. File manifes berisi informasi label dan kotak pembatas untuk gambar yang Anda impor.

Amazon Rekognition memerlukan izin untuk mengakses bucket Amazon S3 tempat gambar Anda disimpan. Jika Anda menggunakan bucket konsol yang disiapkan untuk Anda oleh Label Kustom Rekognition Amazon, izin yang diperlukan sudah disiapkan. Jika Anda tidak menggunakan bucket konsol, lihat [Mengakses Bucket Amazon S3 eksternal](#).

## Membuat file manifes dengan pekerjaan SageMaker AI Ground Truth (Console)

Prosedur berikut menunjukkan kepada Anda cara membuat kumpulan data dengan menggunakan gambar yang diberi label oleh pekerjaan SageMaker AI Ground Truth. File keluaran pekerjaan disimpan di bucket konsol Label Kustom Rekognition Amazon Anda.

Untuk membuat kumpulan data menggunakan gambar yang diberi label oleh pekerjaan SageMaker AI Ground Truth (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon S3 di <https://console.aws.amazon.com/s3/>

- Di bucket konsol, [buat folder](#) untuk menyimpan gambar latihan Anda.

**Note**

Bucket konsol dibuat saat Anda pertama kali membuka konsol Label Kustom Rekognition Amazon di suatu Wilayah. AWS Untuk informasi selengkapnya, lihat [Mengelola proyek Label Kustom Rekognition Amazon](#).

- [Unggah gambar Anda](#) ke folder yang baru saja Anda buat.
- Di bucket konsol, buat folder untuk menyimpan output dari pekerjaan Ground Truth.
- Buka konsol SageMaker AI di <https://console.aws.amazon.com/sagemaker/>.
- Buat pekerjaan pelabelan Ground Truth. Anda memerlukan Amazon S3 URLs untuk folder yang Anda buat di langkah 2 dan langkah 4. Untuk informasi selengkapnya, lihat [Menggunakan Amazon SageMaker Ground Truth untuk Pelabelan Data](#).
- Perhatikan lokasi output `.manifest` file di folder yang Anda buat di langkah 4. Itu harus di sub-folder `Ground-Truth-Job-Name/manifests/output`.
- Ikuti petunjuk di [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(Console\)](#) untuk membuat kumpulan data dengan file manifes yang diunggah. Untuk langkah 8, di lokasi file.manifest, masukkan URL Amazon S3 untuk lokasi yang Anda catat di langkah sebelumnya. Jika Anda menggunakan AWS SDK, lakukan [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).
- Ulangi langkah 1 - 6 untuk membuat pekerjaan SageMaker AI Ground Truth untuk kumpulan data pengujian Anda.

## Membuat file manifes

Anda dapat membuat kumpulan data pengujian atau pelatihan dengan mengimpor file manifes format SageMaker AI Ground Truth. Jika gambar Anda diberi label dalam format yang bukan file manifes SageMaker AI Ground Truth, gunakan informasi berikut untuk membuat file manifes format SageMaker AI Ground Truth.

File manifes dalam format [baris JSON](#) di mana setiap baris adalah objek JSON lengkap yang mewakili informasi pelabelan untuk gambar. Amazon Rekognition Custom Labels mendukung manifes AI SageMaker Ground Truth dengan garis JSON dalam format berikut:

- [Classification Job Output](#) - Gunakan untuk menambahkan label tingkat gambar ke gambar. Label tingkat gambar mendefinisikan kelas adegan, konsep, atau objek (jika informasi lokasi objek tidak

diperlukan) yang ada pada gambar. Sebuah gambar dapat memiliki lebih dari satu label tingkat gambar. Untuk informasi selengkapnya, lihat [Mengimpor label tingkat gambar dalam file manifes](#).

- [Bounding Box Job Output](#) - Gunakan untuk memberi label kelas dan lokasi satu atau lebih objek pada gambar. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

Baris JSON tingkat gambar dan pelokalan (kotak pembatas) dapat dirantai bersama dalam file manifes yang sama.

#### Note

Contoh baris JSON di bagian ini diformat agar mudah dibaca.

Saat Anda mengimpor file manifes, Label Kustom Rekognition Amazon menerapkan aturan validasi untuk batas, sintaks, dan semantik. Untuk informasi selengkapnya, lihat [Aturan validasi untuk file manifes](#).

Gambar yang direferensikan oleh file manifes harus ditempatkan di bucket Amazon S3 yang sama. File manifes dapat ditemukan di bucket Amazon S3 yang berbeda dari bucket Amazon S3 yang menyimpan gambar. Anda menentukan lokasi gambar di `source-ref` bidang garis JSON.

Amazon Rekognition memerlukan izin untuk mengakses bucket Amazon S3 tempat gambar Anda disimpan. Jika Anda menggunakan bucket konsol yang disiapkan untuk Anda oleh Label Kustom Rekognition Amazon, izin yang diperlukan sudah disiapkan. Jika Anda tidak menggunakan bucket konsol, lihat [Mengakses Bucket Amazon S3 eksternal](#).

#### Topik

- [Membuat file manifes](#)

#### Membuat file manifes

Prosedur berikut membuat proyek dengan dataset pelatihan dan pengujian. Kumpulan data dibuat dari file manifes pelatihan dan pengujian yang Anda buat.

Untuk membuat kumpulan data menggunakan file manifes format SageMaker AI Ground Truth (konsol)

1. Di bucket konsol, [buat folder untuk](#) menyimpan file manifes Anda.

2. Di bucket konsol, buat folder untuk menyimpan gambar Anda.
3. Unggah gambar Anda ke folder yang baru saja Anda buat.
4. Buat file manifes format SageMaker AI Ground Truth untuk kumpulan data pelatihan Anda. Untuk informasi selengkapnya, lihat [Mengimpor label tingkat gambar dalam file manifes](#) dan [Lokalisasi objek dalam file manifes](#).


 Important

Nilai `source-ref` bidang di setiap baris JSON harus dipetakan ke gambar yang Anda unggah.

5. Buat file manifes format SageMaker AI Ground Truth untuk kumpulan data pengujian Anda.
6. [Unggah file manifes Anda](#) ke folder yang baru saja Anda buat.
7. Perhatikan lokasi file manifes.
8. Ikuti petunjuk di [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(Console\)](#) untuk membuat kumpulan data dengan file manifes yang diunggah. Untuk langkah 8, di lokasi `file.manifest`, masukkan URL Amazon S3 untuk lokasi yang Anda catat di langkah sebelumnya. Jika Anda menggunakan AWS SDK, lakukan [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).

## Mengimpor label tingkat gambar dalam file manifes

Untuk mengimpor label tingkat gambar (gambar berlabel adegan, konsep, atau objek yang tidak memerlukan informasi pelokalan), Anda menambahkan baris JSON format [Output Pekerjaan](#) Klasifikasi Ground SageMaker Truth AI ke file manifes. File manifes terbuat dari satu atau lebih baris JSON, satu untuk setiap gambar yang ingin Anda impor.

 Tip

Untuk menyederhanakan pembuatan file manifes, kami menyediakan skrip Python yang membuat file manifes dari file CSV. Untuk informasi selengkapnya, lihat [Membuat file manifes dari file CSV](#).

## Untuk membuat file manifes untuk label tingkat gambar

1. Buat file teks kosong.

2. Tambahkan baris JSON untuk setiap gambar yang ingin Anda impor. Setiap baris JSON akan terlihat mirip dengan yang berikut ini.

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

3. Simpan file tersebut. Anda dapat menggunakan ekstensi `.manifest`, tetapi tidak diperlukan.
4. Buat kumpulan data menggunakan file manifes yang Anda buat. Untuk informasi selengkapnya, lihat [Untuk membuat kumpulan data menggunakan file manifes format SageMaker AI Ground Truth \(konsol\)](#).

### Garis JSON Tingkat Gambar

Di bagian ini, kami menunjukkan cara membuat garis JSON untuk satu gambar. Pertimbangkan citra berikut. Adegan untuk gambar berikut mungkin disebut Sunrise.



Garis JSON untuk gambar sebelumnya, dengan adegan Sunrise, mungkin sebagai berikut.

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

Perhatikan informasi berikut.

sumber-ref

(Wajib) Lokasi Amazon S3 dari gambar. Formatnya adalah "`s3://BUCKET/OBJECT_PATH`". Gambar dalam kumpulan data yang diimpor harus disimpan dalam bucket Amazon S3 yang sama.

### ***testdataset-classification\_Sunrise***

(Wajib) Atribut label. Anda memilih nama bidang. Nilai bidang (1 dalam contoh sebelumnya) adalah pengenalan atribut label. Ini tidak digunakan oleh Amazon Rekognition Custom Labels dan dapat berupa nilai integer apa pun. Harus ada metadata terkait yang diidentifikasi dengan nama bidang dengan `-metadata` ditambahkan. Misalnya, "`testdataset-classification_Sunrise-metadata`".

### ***testdataset-classification\_Sunrise***-metadata

(Wajib) Metadata tentang atribut label. Nama bidang harus sama dengan atribut label dengan `-metadata` ditambahkan.

kepercayaan

(Wajib) Saat ini tidak digunakan oleh Label Kustom Amazon Rekognition tetapi nilai antara 0 dan 1 harus diberikan.

nama-pekerjaan

(Opsional) Nama yang Anda pilih untuk pekerjaan yang memproses gambar.

nama kelas

(Wajib) Nama kelas yang Anda pilih untuk adegan atau konsep yang berlaku untuk gambar. Misalnya, "`Sunrise`".

beranotasi manusia

(Wajib) Tentukan "`yes`", jika anotasi diselesaikan oleh manusia. Jika tidak "`no`".

kreasi-tanggal

(Wajib) Tanggal dan waktu Universal Terkoordinasi (UTC) saat label dibuat.

jenis

(Wajib) Jenis pemrosesan yang harus diterapkan pada gambar. Untuk label tingkat gambar, nilainya adalah. "`groundtruth/image-classification`"

## Menambahkan beberapa label tingkat gambar ke gambar

Anda dapat menambahkan beberapa label ke gambar. Misalnya, JSON berikut menambahkan dua label, sepak bola dan bola ke satu gambar.

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

## Lokalisasi objek dalam file manifes

Anda dapat mengimpor gambar berlabel informasi pelokalan objek dengan menambahkan SageMaker AI Ground Truth [Bounding Box Job Output](#) format baris JSON ke file manifes.

Informasi lokalisasi mewakili lokasi objek pada gambar. Lokasi diwakili oleh kotak pembatas yang mengelilingi objek. Struktur kotak pembatas berisi koordinat kiri atas kotak pembatas dan lebar dan tinggi kotak pembatas. Garis JSON format kotak pembatas mencakup kotak pembatas untuk lokasi satu atau lebih objek pada gambar dan kelas setiap objek pada gambar.

File manifes terbuat dari satu atau lebih baris JSON, setiap baris berisi informasi untuk satu gambar.

Untuk membuat file manifes untuk lokalisasi objek

1. Buat file teks kosong.

2. Tambahkan baris JSON untuk setiap gambar yang ingin Anda impor. Setiap baris JSON akan terlihat mirip dengan yang berikut ini.

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

3. Simpan file tersebut. Anda dapat menggunakan ekstensi `.manifest`, tetapi tidak diperlukan.
4. Buat kumpulan data menggunakan file yang baru saja Anda buat. Untuk informasi selengkapnya, lihat [Untuk membuat kumpulan data menggunakan file manifest format SageMaker AI Ground Truth \(konsol\)](#).

### Objek pembatas kotak garis JSON

Di bagian ini, kami menunjukkan cara membuat garis JSON untuk satu gambar. Gambar berikut menunjukkan kotak pembatas di sekitar perangkat Amazon Echo dan Amazon Echo Dot.



Berikut ini adalah garis JSON kotak pembatas untuk gambar sebelumnya.

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

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

Perhatikan informasi berikut.

sumber-ref

(Wajib) Lokasi Amazon S3 dari gambar. Formatnya adalah "`s3://BUCKET/OBJECT_PATH`". Gambar dalam kumpulan data yang diimpor harus disimpan dalam bucket Amazon S3 yang sama.

### ***bounding-box***

(Wajib) Atribut label. Anda memilih nama bidang. Berisi ukuran gambar dan kotak pembatas untuk setiap objek yang terdeteksi dalam gambar. Harus ada metadata terkait yang diidentifikasi dengan nama bidang dengan -metadata ditambahkan. Misalnya, "bounding-box-metadata".

image\_size

(Wajib) Sebuah array elemen tunggal yang berisi ukuran gambar dalam piksel.

- tinggi - (Wajib) Ketinggian gambar dalam piksel.

- lebar - (Wajib) Kedalaman gambar dalam piksel.
- kedalaman — (Diperlukan) Jumlah saluran dalam gambar. Untuk gambar RGB, nilainya adalah 3. Saat ini tidak digunakan oleh Amazon Rekognition Custom Labels, tetapi nilai diperlukan.

#### anotasi

(Wajib) Sebuah array informasi kotak pembatas untuk setiap objek yang terdeteksi dalam gambar.

- class\_id — (Wajib) Memetakan ke label di peta kelas. Pada contoh sebelumnya, objek dengan class\_id dari 1 adalah Echo Dot pada gambar.
- atas — (Wajib) Jarak dari atas gambar ke bagian atas kotak pembatas, dalam piksel.
- kiri - (Wajib) Jarak dari kiri gambar ke kiri kotak pembatas, dalam piksel.
- lebar - (Wajib) Lebar kotak pembatas, dalam piksel.
- tinggi - (Wajib) Ketinggian kotak pembatas, dalam piksel.

#### **bounding-box**-metadata

(Wajib) Metadata tentang atribut label. Nama bidang harus sama dengan atribut label dengan - metadata ditambahkan. Array informasi kotak pembatas untuk setiap objek yang terdeteksi dalam gambar.

#### Objek

(Wajib) Sebuah array objek yang ada dalam gambar. Memetakan ke array anotasi berdasarkan indeks. Atribut confidence tidak digunakan oleh Amazon Rekognition Custom Labels.

#### peta kelas

(Wajib) Peta kelas yang berlaku untuk objek yang terdeteksi dalam gambar.

#### jenis

(Wajib) Jenis pekerjaan klasifikasi. "groundtruth/object-detection" mengidentifikasi pekerjaan sebagai deteksi objek.

#### kreasi-tanggal

(Wajib) Tanggal dan waktu Universal Terkoordinasi (UTC) saat label dibuat.

#### beranotasi manusia

(Wajib) Tentukan "yes", jika anotasi diselesaikan oleh manusia. Jika tidak "no".

## nama-pekerjaan

(Opsional) Nama pekerjaan yang memproses gambar.

### Aturan validasi untuk file manifes

Saat Anda mengimpor file manifes, Label Kustom Rekognition Amazon menerapkan aturan validasi untuk batas, sintaks, dan semantik. Skema SageMaker AI Ground Truth memberlakukan validasi sintaks. Untuk informasi selengkapnya, lihat [Output](#). Berikut ini adalah aturan validasi untuk batas dan semantik.

#### Note

- Aturan ketidakabsahan 20% berlaku secara kumulatif di semua aturan validasi. Jika impor melebihi batas 20% karena kombinasi apa pun, seperti 15% JSON tidak valid dan 15% gambar tidak valid, impor gagal.
- Setiap objek dataset adalah garis dalam manifes. Baris kosong/tidak valid juga dihitung sebagai objek dataset.
- Tumpang tindih adalah (label umum antara tes dan kereta)/(label kereta).

### Topik

- [Batas](#)
- [Semantik](#)

### Batas

Validasi	Kuota	Kesalahan dimunculkan
Ukuran file manifes	Maksimal 1 GB	Kesalahan
Jumlah baris maksimum untuk file manifes	Maksimum 250.000 objek dataset sebagai garis dalam manifes.	Kesalahan

Validasi	Kuota	Kesalahan dimunculkan
Batas bawah pada jumlah total objek dataset yang valid per label	$\geq 1$	Kesalahan
Batas bawah pada label	$\geq 2$	Kesalahan
Batas atas pada label	$\leq 250$	Kesalahan
Kotak pembatas minimum per gambar	0	Tidak ada
Kotak pembatas maksimum per gambar	50	Tidak ada

## Semantik

Validasi	Kuota	Kesalahan dimunculkan
Manifes kosong		Kesalahan
Objek referensi sumber yang hilang/dapat diakses	Jumlah objek kurang dari 20%	Peringatan
Objek referensi sumber yang hilang/dapat diakses	Jumlah objek $> 20\%$	Kesalahan
Label uji tidak ada dalam kumpulan data pelatihan	Setidaknya 50% tumpang tindih dalam label	Kesalahan
Campuran contoh label vs. objek untuk label yang sama dalam kumpulan data. Klasifikasi dan deteksi untuk kelas yang sama dalam objek dataset.		Tidak ada kesalahan atau peringatan

Validasi	Kuota	Kesalahan dimunculkan
Aset yang tumpang tindih antara tes dan kereta	Seharusnya tidak ada tumpang tindih antara set data tes dan pelatihan.	
Gambar dalam kumpulan data harus dari ember yang sama	Kesalahan jika objek berada di ember yang berbeda	Kesalahan

## Mengonversi format dataset lain ke file manifes

Anda dapat menggunakan informasi berikut untuk membuat file manifes format Amazon SageMaker AI dari berbagai format kumpulan data sumber. Setelah membuat file manifes, gunakan untuk membuat dataset. Untuk informasi selengkapnya, lihat [Menggunakan file manifes untuk mengimpor gambar](#).

### Topik

- [Mengubah dataset COCO menjadi format file manifes](#)
- [Mengubah file manifes SageMaker AI Ground Truth multi-label](#)
- [Membuat file manifes dari file CSV](#)

## Mengubah dataset COCO menjadi format file manifes

[COCO](#) adalah format untuk menentukan deteksi objek skala besar, segmentasi, dan kumpulan data teks. [Contoh Python ini menunjukkan kepada Anda cara mengubah kumpulan data format deteksi objek COCO menjadi file manifes format kotak pembatas Amazon Rekognition Custom Labels](#).

Bagian ini juga mencakup informasi yang dapat Anda gunakan untuk menulis kode Anda sendiri.

File JSON format COCO terdiri dari lima bagian yang menyediakan informasi untuk seluruh kumpulan data. Untuk informasi selengkapnya, lihat [Format dataset COCO](#).

- `info`— informasi umum tentang dataset.
- `licenses` — informasi lisensi untuk gambar dalam dataset.
- `images`— daftar gambar dalam dataset.
- `annotations`— daftar anotasi (termasuk kotak pembatas) yang ada di semua gambar dalam kumpulan data.

- [categories](#)— daftar kategori label.

Anda memerlukan informasi dari `images`, `annotations`, dan `categories` daftar untuk membuat file manifes Label Kustom Rekognition Amazon.

File manifes Label Kustom Rekognition Amazon dalam format baris JSON di mana setiap baris memiliki kotak pembatas dan informasi label untuk satu atau beberapa objek pada gambar. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

### Memetakan Objek COCO ke Garis JSON Label Kustom

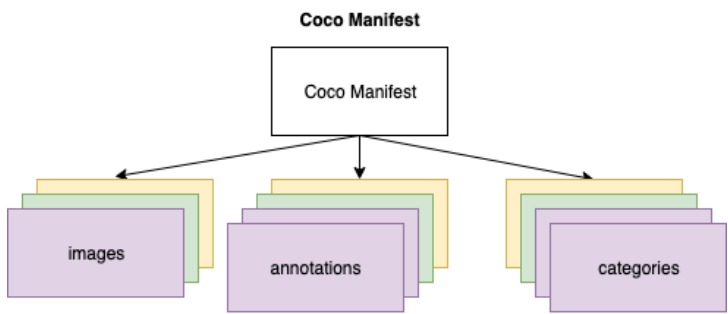
Untuk mengubah kumpulan data format COCO, Anda memetakan kumpulan data COCO ke file manifes Label Kustom Rekognition Amazon untuk pelokalan objek. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#). Untuk membuat baris JSON untuk setiap gambar, file manifes perlu memetakan kumpulan data COCO `imageannotation`, dan `category` bidang objek. IDs

Berikut ini adalah contoh file manifes COCO. Untuk informasi selengkapnya, lihat [Format dataset COCO](#).

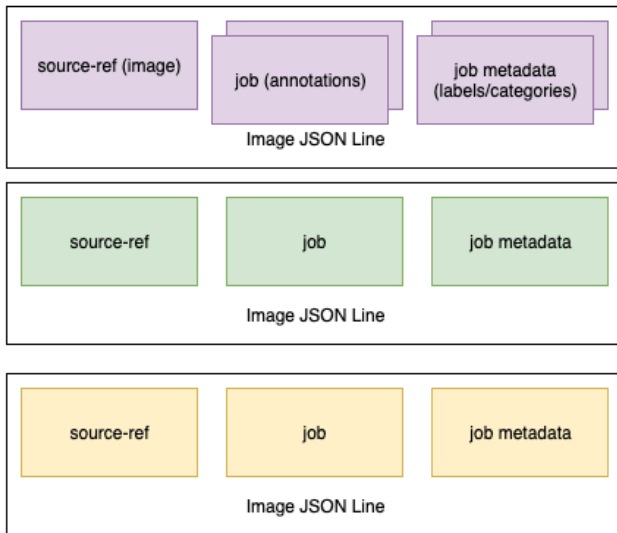
```
{
  "info": {
    "description": "COCO 2017 Dataset", "url": "http://cocodataset.org", "version":
"1.0", "year": 2017, "contributor": "COCO Consortium", "date_created": "2017/09/01"
  },
  "licenses": [
    {"url": "http://creativecommons.org/licenses/by/2.0/", "id": 4, "name":
"Attribution License"}
  ],
  "images": [
    {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxxx.jpg",
"date_captured": "2013-11-15 02:41:42"},
    {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnnn.jpg",
"date_captured": "2013-11-18 02:53:27"}
  ],
  "annotations": [
```

```
    {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
417.51,.....167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
[19.23, 383.18, 314.5, 244.46]},
    {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
238.8,.....382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
"bbox": [399, 251, 155, 101]},
    {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
239.01,.....495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
"bbox": [86, 65, 220, 334]}
  ],
  "categories": [
    {"supercategory": "speaker","id": 0,"name": "echo"},
    {"supercategory": "speaker","id": 1,"name": "echo dot"}
  ]
}
```

Diagram berikut menunjukkan bagaimana kumpulan data COCO mencantumkan peta kumpulan data ke baris JSON Label Kustom Rekognition Amazon untuk gambar. Setiap baris JSON untuk gambar memiliki kolom sumber referensi, pekerjaan, dan metadata pekerjaan. Warna yang cocok menunjukkan informasi untuk satu gambar. Perhatikan bahwa dalam manifes, gambar individu mungkin memiliki beberapa anotasi dan metadata/kategori.



**Custom Labels JSON Lines**



Untuk mendapatkan objek COCO untuk satu baris JSON

1. Untuk setiap gambar dalam daftar gambar, dapatkan anotasi dari daftar anotasi di mana nilai bidang anotasi `image_id` cocok dengan bidang gambar. `id`
2. Untuk setiap anotasi yang cocok di langkah 1, baca `categories` daftar dan dapatkan masing-masing `category` nilai bidang yang `id` cocok dengan `category` bidang `annotation` objek `category_id`.
3. Buat garis JSON untuk gambar menggunakan objek yang cocok `imageannotation`, dan `category`. Untuk memetakan bidang, lihat [Memetakan bidang objek COCO ke bidang objek garis JSON Label Kustom](#).
4. Ulangi langkah 1-3 sampai Anda telah membuat baris JSON untuk setiap `image` objek dalam daftar. `images`

Untuk kode sampel, lihat [Mengubah dataset COCO](#).

## Memetakan bidang objek COCO ke bidang objek garis JSON Label Kustom

Setelah Anda mengidentifikasi objek COCO untuk baris JSON Label Kustom Rekognition Amazon, Anda perlu memetakan bidang objek COCO ke bidang objek baris JSON Label Kustom Amazon Rekognition masing-masing. Contoh berikut Amazon Rekognition Custom Labels JSON line memetakan satu gambar id (000000245915=) ke contoh COCO JSON sebelumnya. Perhatikan informasi berikut.

- `source-ref` adalah lokasi gambar dalam ember Amazon S3. Jika gambar COCO Anda tidak disimpan dalam bucket Amazon S3, Anda harus memindahkannya ke bucket Amazon S3.
- `annotations` Daftar berisi `annotation` objek untuk setiap objek pada gambar. `annotation` Objek mencakup informasi kotak pembatas (`top`, `left`, `width`, `height`) dan pengenalan label (`class_id`).
- Pengenalan label (`class_id`) memetakan ke `class-map` daftar dalam metadata. Ini mencantumkan label yang digunakan pada gambar.

```
{
  "source-ref": "s3://custom-labels-bucket/images/000000245915.jpg",
  "bounding-box": {
    "image_size": {
      "width": 640,
      "height": 480,
      "depth": 3
    },
    "annotations": [{
      "class_id": 0,
      "top": 251,
      "left": 399,
      "width": 155,
      "height": 101
    }, {
      "class_id": 1,
      "top": 65,
      "left": 86,
      "width": 220,
      "height": 334
    }
  ],
  "bounding-box-metadata": {
    "objects": [{
```

```
  "confidence": 1
}, {
  "confidence": 1
}],
"class-map": {
  "0": "Echo",
  "1": "Echo Dot"
},
"type": "groundtruth/object-detection",
"human-annotated": "yes",
"creation-date": "2018-10-18T22:18:13.527256",
"job-name": "my job"
}
}
```

Gunakan informasi berikut untuk memetakan kolom file manifes Label Kustom Rekognition Amazon ke bidang JSON kumpulan data COCO.

#### sumber-ref

URL format S3 untuk lokasi gambar. Gambar harus disimpan dalam ember S3. Untuk informasi selengkapnya, lihat [sumber-ref](#). Jika bidang `coco_url` COCO menunjuk ke lokasi bucket S3, Anda dapat menggunakan nilai `coco_url` untuk nilai `source-ref`. Atau, Anda dapat memetakan `source-ref` ke bidang `file_name` (COCO) dan dalam kode transformasi Anda, tambahkan jalur S3 yang diperlukan ke tempat gambar disimpan.

#### ***bounding-box***

Nama atribut label yang Anda pilih. Untuk informasi selengkapnya, lihat [bounding-box](#).

#### image\_size

Ukuran gambar dalam piksel. Peta ke `image` objek dalam daftar [gambar](#).

- `height`-> [image](#).height
- `width`-> [image](#).width
- `depth`-> Tidak digunakan oleh Label Kustom Rekognition Amazon tetapi nilainya harus diberikan.

#### anotasi

Daftar objek annotation. Ada satu annotation untuk setiap objek pada gambar.

## anotasi

Berisi informasi kotak pembatas untuk satu contoh objek pada gambar.

- `class_id`-> pemetaan id numerik ke daftar Custom Label. `class-map`
- `top` -> `bbox[1]`
- `left` -> `bbox[0]`
- `width` -> `bbox[2]`
- `height` -> `bbox[3]`

## ***bounding-box***-metadata

Metadata untuk atribut label. Termasuk label dan pengidentifikasi label. Untuk informasi selengkapnya, lihat [bounding-box-metadata](#).

## Objek

Array objek dalam gambar. Peta ke `annotations` daftar berdasarkan indeks.

## Objek

- `confidence`-> Tidak digunakan oleh Amazon Rekognition Custom Labels, tetapi nilai (1) diperlukan.

## peta kelas

Peta label (kelas) yang berlaku untuk objek yang terdeteksi dalam gambar. Peta ke objek kategori dalam daftar [kategori](#).

- `id` -> `category.id`
- `id value` -> `category.name`

## jenis

Harus `groundtruth/object-detection`

## beranotasi manusia

Tentukan `yes` atau `no`. Untuk informasi selengkapnya, lihat [bounding-box-metadata](#).

## [kreasi-tanggal -> gambar .date\\_capture](#)

Tanggal dan waktu pembuatan gambar. Memetakan ke bidang [gambar](#) .date\_capture dari gambar dalam daftar gambar COCO. Amazon Rekognition Custom Labels mengharapkan format *creation-date* menjadi Y-M-DTH:M: S.

nama-pekerjaan

Nama pekerjaan yang Anda pilih.

### Format dataset COCO

Dataset COCO terdiri dari lima bagian informasi yang memberikan informasi untuk seluruh kumpulan data. Format untuk kumpulan data deteksi objek COCO didokumentasikan di Format Data [COCO](#).

- [info](#) — informasi umum tentang dataset.
- [lisensi](#) — informasi lisensi untuk gambar dalam dataset.
- [gambar](#) — daftar gambar dalam dataset.
- [anotasi](#) — daftar anotasi (termasuk kotak pembatas) yang ada di semua gambar dalam kumpulan data.
- [kategori](#) — daftar kategori label.

Untuk membuat manifes Label Kustom, Anda menggunakan `images`, `annotations`, dan `categories` daftar dari file manifes COCO. Bagian lain (`info`, `licences`) tidak diperlukan. Berikut ini adalah contoh file manifes COCO.

```
{
  "info": {
    "description": "COCO 2017 Dataset", "url": "http://cocodataset.org", "version":
"1.0", "year": 2017, "contributor": "COCO Consortium", "date_created": "2017/09/01"
  },
  "licenses": [
    {"url": "http://creativecommons.org/licenses/by/2.0/", "id": 4, "name":
"Attribution License"}
  ],
  "images": [
    {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxx.jpg",
"date_captured": "2013-11-15 02:41:42"},
```

```

    {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
  "date_captured": "2013-11-18 02:53:27"}
  ],
  "annotations": [
    {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
417.51,.....167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
[19.23, 383.18, 314.5, 244.46]},
    {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
238.8,.....382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
"bbox": [399, 251, 155, 101]},
    {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
239.01,.....495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
"bbox": [86, 65, 220, 334]}
  ],
  "categories": [
    {"supercategory": "speaker","id": 0,"name": "echo"},
    {"supercategory": "speaker","id": 1,"name": "echo dot"}
  ]
}

```

## daftar gambar

Gambar yang direferensikan oleh dataset COCO tercantum dalam larik gambar. Setiap objek gambar berisi informasi tentang gambar seperti nama file gambar. Dalam contoh objek gambar berikut, perhatikan informasi berikut dan bidang mana yang diperlukan untuk membuat file manifes Label Kustom Rekognition Amazon.

- `id`— (Diperlukan) Pengidentifikasi unik untuk gambar. `id` Bidang memetakan ke `id` bidang dalam array anotasi (tempat informasi kotak pembatas disimpan).
- `license`— (Tidak Diperlukan) Peta ke array lisensi.
- `coco_url`— (Opsional) Lokasi gambar.
- `flickr_url`— (Tidak diperlukan) Lokasi gambar di Flickr.
- `width`— (Wajib) Lebar gambar.
- `height`— (Wajib) Ketinggian gambar.
- `file_name`— (Wajib) Nama file gambar. Dalam contoh ini, `file_name` dan `id` cocok, tetapi ini bukan persyaratan untuk kumpulan data COCO.
- `date_captured`— (Wajib) tanggal dan waktu gambar diambil.

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

### daftar anotasi (kotak pembatas)

Informasi kotak pembatas untuk semua objek pada semua gambar disimpan daftar anotasi. Objek anotasi tunggal berisi informasi kotak pembatas untuk satu objek dan label objek pada gambar. Ada objek anotasi untuk setiap instance objek pada gambar.

Dalam contoh berikut, perhatikan informasi berikut dan bidang mana yang diperlukan untuk membuat file manifes Label Kustom Rekognition Amazon.

- `id`— (Tidak diperlukan) Pengidentifikasi untuk anotasi.
- `image_id`— (Wajib) Sesuai dengan gambar `id` dalam array gambar.
- `category_id`— (Wajib) Pengidentifikasi untuk label yang mengidentifikasi objek dalam kotak pembatas. Ini memetakan ke `id` bidang array kategori.
- `iscrowd`— (Tidak diperlukan) Menentukan apakah gambar berisi kerumunan objek.
- `segmentation`— (Tidak diperlukan) Informasi segmentasi untuk objek pada gambar. Amazon Rekognition Custom Labels tidak mendukung segmentasi.
- `area`— (Tidak diperlukan) Area anotasi.
- `bbox`— (Wajib) Berisi koordinat, dalam piksel, dari kotak pembatas di sekitar objek pada gambar.

```
{
  "id": 1409619,
  "category_id": 1,
  "iscrowd": 0,
  "segmentation": [
    [86.0, 238.8, .....382.74, 241.17]
  ],
}
```

```
"image_id": 245915,  
"area": 3556.21970000000015,  
"bbox": [86, 65, 220, 334]  
}
```

## daftar kategori

Informasi label disimpan array kategori. Dalam objek kategori contoh berikut, perhatikan informasi berikut dan bidang mana yang diperlukan untuk membuat file manifes Label Kustom Rekognition Amazon.

- `supercategory`— (Tidak wajib) Kategori induk untuk label.
- `id`— (Wajib) Pengidentifikasi label. `id` Bidang memetakan ke `category_id` bidang dalam suatu `annotation` objek. Dalam contoh berikut, Pengidentifikasi untuk titik gema adalah 2.
- `name`— (Diperlukan) nama label.

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

## Mengubah dataset COCO

Gunakan contoh Python berikut untuk mengubah informasi kotak pembatas dari kumpulan data format COCO menjadi file manifes Label Kustom Rekognition Amazon. Kode mengunggah file manifes yang dibuat ke bucket Amazon S3 Anda. Kode ini juga menyediakan perintah AWS CLI yang dapat Anda gunakan untuk mengunggah gambar Anda.

Untuk mengubah dataset COCO (SDK)

1. Jika belum:
  - a. Pastikan Anda memiliki `AmazonS3FullAccess` izin. Untuk informasi selengkapnya, lihat [Siapkan izin SDK](#).
  - b. Instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode Python berikut untuk mengubah dataset COCO. Tetapkan nilai-nilai berikut.
  - `s3_bucket`— Nama bucket S3 tempat Anda ingin menyimpan gambar dan file manifes Label Kustom Rekognition Amazon.

- `s3_key_path_images`— Jalur ke tempat Anda ingin menempatkan gambar di dalam ember S3 (`s3_bucket`).
- `s3_key_path_manifest_file`— Jalur ke tempat Anda ingin menempatkan file manifes Label Kustom dalam bucket S3 (`s3_bucket`).
- `local_path`— Jalur lokal ke tempat contoh membuka kumpulan data COCO input dan juga menyimpan file manifes Label Kustom baru.
- `local_images_path`— Jalur lokal ke gambar yang ingin Anda gunakan untuk pelatihan.
- `coco_manifest`— Nama file dataset COCO masukan.
- `cl_manifest_file`— Nama untuk file manifes yang dibuat oleh contoh. File disimpan di lokasi yang ditentukan oleh `local_path`. Dengan konvensi, file memiliki ekstensi `.manifest`, tetapi ini tidak diperlukan.
- `job_name`— Nama untuk pekerjaan Label Kustom.

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

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

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

label_attribute = 'bounding-box'
```

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

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

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

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

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

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

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

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

with open(coco_json_file) as f:

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

```
annotations = js['annotations']

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

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

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

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

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}

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

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

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

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

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

print('Done parsing annotations')
```

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

for im in images_dict.values():

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

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

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

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

3. Jalankan kode tersebut.
4. Dalam output program, perhatikan `s3 sync` perintahnya. Anda membutuhkannya di langkah berikutnya.
5. Pada prompt perintah, jalankan `s3 sync` perintah. Gambar Anda diunggah ke bucket S3. Jika perintah gagal selama upload, jalankan lagi hingga gambar lokal Anda disinkronkan dengan bucket S3.
6. Dalam output program, perhatikan jalur URL S3 ke file manifes. Anda membutuhkannya di langkah berikutnya.
7. Ikuti instruksi di [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(Console\)](#) untuk membuat kumpulan data dengan file manifes yang diunggah. Untuk langkah 8, di lokasi `file.manifest`, masukkan URL Amazon S3 yang Anda catat di langkah sebelumnya.

Jika Anda menggunakan AWS SDK, lakukan [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).

## Mengubah file manifes SageMaker AI Ground Truth multi-label

Topik ini menunjukkan kepada Anda cara mengubah file manifes Amazon SageMaker AI Ground Truth multi-label menjadi file manifes format Label Kustom Rekognition Amazon.

SageMaker File manifes AI Ground Truth untuk pekerjaan multi-label diformat secara berbeda dari file manifes format Amazon Rekognition Custom Labels. Klasifikasi multi-label adalah ketika gambar diklasifikasikan ke dalam satu set kelas, tetapi mungkin milik beberapa kelas sekaligus. Dalam hal ini, gambar berpotensi memiliki beberapa label (multi-label), seperti sepak bola dan bola.

Untuk informasi tentang pekerjaan SageMaker AI Ground Truth multi-label, lihat [Klasifikasi Gambar \(Multi-label\)](#). Untuk informasi tentang file manifes Label Kustom Amazon Rekognition format multi-label, lihat [the section called “Menambahkan beberapa label tingkat gambar ke gambar”](#)

## Mendapatkan file manifes untuk pekerjaan SageMaker AI Ground Truth

Prosedur berikut menunjukkan cara mendapatkan file manifes keluaran (`output.manifest`) untuk pekerjaan Amazon SageMaker AI Ground Truth. Anda menggunakan `output.manifest` sebagai masukan untuk prosedur berikutnya.

## Untuk mengunduh file manifes pekerjaan SageMaker AI Ground Truth

1. Buka <https://console.aws.amazon.com/sagemaker/>.
2. Di panel navigasi, pilih Ground Truth lalu pilih Labeling Jobs.
3. Pilih pekerjaan pelabelan yang berisi file manifes yang ingin Anda gunakan.
4. Pada halaman detail, pilih tautan di bawah Lokasi set data keluaran. Konsol Amazon S3 dibuka di lokasi dataset.
5. Pilih `Manifests`, `output` dan kemudian `output.manifest`.
6. Pilih Tindakan Objek dan kemudian pilih Unduh untuk mengunduh file manifes.

## Mengubah file manifes SageMaker AI multi-label

Prosedur berikut membuat file manifes Amazon Rekognition Custom Labels format multi-label dari file manifes AI format SageMaker GroundTruth multi-label yang ada.

**Note**

Untuk menjalankan kode, Anda memerlukan Python versi 3, atau lebih tinggi.

Untuk mengubah file manifes SageMaker AI multi-label

1. Jalankan kode python berikut. Berikan nama file manifes yang Anda buat [Mendapatkan file manifes untuk pekerjaan SageMaker AI Ground Truth](#) sebagai argumen baris perintah.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to create and Amazon Rekognition Custom Labels format
manifest file from an Amazon SageMaker Ground Truth Image
Classification (Multi-label) format manifest file.
"""
import json
import logging
import argparse
import os.path

logger = logging.getLogger(__name__)

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

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

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

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

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

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

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

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

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

```
    return new_manifest_file

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

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

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

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

2. Perhatikan nama file manifes baru yang ditampilkan skrip. Anda menggunakannya di langkah berikutnya.
3. [Unggah file manifes Anda](#) ke bucket Amazon S3 yang ingin Anda gunakan untuk menyimpan file manifes.

#### Note

Pastikan Label Kustom Amazon Rekognition memiliki akses ke bucket Amazon S3 yang direferensikan di bidang baris JSON file `source-ref` manifes. Untuk informasi selengkapnya, lihat [Mengakses Bucket Amazon S3 eksternal](#). Jika lowongan Ground

Truth menyimpan gambar di Bucket Konsol Label Kustom Amazon Rekognition, Anda tidak perlu menambahkan izin.

- Ikuti petunjuk di [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(Console\)](#) untuk membuat kumpulan data dengan file manifes yang diunggah. Untuk langkah 8, di lokasi file.manifest, masukkan URL Amazon S3 untuk lokasi file manifes. Jika Anda menggunakan AWS SDK, lakukan [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).

## Membuat file manifes dari file CSV

Contoh skrip Python ini menyederhanakan pembuatan file manifes dengan menggunakan file Comma Separated Values (CSV) untuk memberi label gambar. Anda membuat file CSV. File manifes cocok untuk [klasifikasi gambar Multi-label](#) atau [Klasifikasi gambar multi-label](#). Untuk informasi selengkapnya, lihat [Temukan objek, adegan, dan konsep](#).

### Note

Skrip ini tidak membuat file manifes yang cocok untuk menemukan [lokasi objek](#) atau untuk menemukan [lokasi merek](#).

File manifes menjelaskan gambar yang digunakan untuk melatih model. Misalnya, lokasi gambar dan label yang ditetapkan untuk gambar. File manifes terdiri dari satu atau lebih baris JSON. Setiap baris JSON menggambarkan satu gambar. Untuk informasi selengkapnya, lihat [the section called “Mengimpor label tingkat gambar dalam file manifes”](#).

File CSV mewakili data tabular di beberapa baris dalam file teks. Bidang pada baris dipisahkan dengan koma. Untuk informasi selengkapnya, lihat [nilai yang dipisahkan koma](#). Untuk skrip ini, setiap baris dalam file CSV Anda mewakili satu gambar dan memetakan ke Baris JSON dalam file manifes. Untuk membuat file CSV untuk file manifes yang mendukung [klasifikasi gambar Multi-label](#), Anda menambahkan satu atau beberapa label tingkat gambar ke setiap baris. Untuk membuat file manifes yang cocok [Klasifikasi gambar](#), Anda menambahkan satu label tingkat gambar ke setiap baris.

Misalnya, File CSV berikut menjelaskan gambar dalam proyek [Klasifikasi gambar multi-label](#) (Bunga) Memulai.

```
camellia1.jpg,camellia,with_leaves
```

```

camellia2.jpg,camellia,with_leaves
camellia3.jpg,camellia,without_leaves
helleborus1.jpg,helleborus,without_leaves,not_fully_grown
helleborus2.jpg,helleborus,with_leaves,fully_grown
helleborus3.jpg,helleborus,with_leaves,fully_grown
jonquil1.jpg,jonquil,with_leaves
jonquil2.jpg,jonquil,with_leaves
jonquil3.jpg,jonquil,with_leaves
jonquil4.jpg,jonquil,without_leaves
mauve_honey_myrtle1.jpg,mauve_honey_myrtle,without_leaves
mauve_honey_myrtle2.jpg,mauve_honey_myrtle,with_leaves
mauve_honey_myrtle3.jpg,mauve_honey_myrtle,with_leaves
mediterranean_spurge1.jpg,mediterranean_spurge,with_leaves
mediterranean_spurge2.jpg,mediterranean_spurge,without_leaves

```

Script menghasilkan JSON Lines untuk setiap baris. Sebagai contoh, berikut ini adalah JSON Line untuk baris pertama (`camellia1.jpg,camellia,with_leaves`).

```

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

```

Dalam contoh CSV, jalur Amazon S3 ke gambar tidak ada. Jika file CSV Anda tidak menyertakan jalur Amazon S3 untuk gambar, gunakan `--s3_path` argumen baris perintah untuk menentukan jalur Amazon S3 ke gambar.

Skrip merekam entri pertama untuk setiap gambar dalam file CSV gambar yang tidak digandakan. File CSV gambar yang dideduplikasi berisi satu contoh dari setiap gambar yang ditemukan dalam file CSV input. Kemunculan lebih lanjut dari gambar dalam file CSV input direkam dalam file CSV gambar duplikat. Jika skrip menemukan gambar duplikat, tinjau file CSV gambar duplikat dan perbarui file CSV gambar yang tidak digandakan seperlunya. Jalankan kembali skrip dengan file deduplikat. Jika tidak ada duplikat yang ditemukan dalam file CSV input, skrip menghapus file CSV gambar yang tidak digandakan dan gambar duplikat, karena kosong. CSVfile

Dalam prosedur ini, Anda membuat file CSV dan menjalankan skrip Python untuk membuat file manifes.

Untuk membuat file manifes dari file CSV

1. Buat file CSV dengan bidang berikut di setiap baris (satu baris per gambar). Jangan menambahkan baris header ke file CSV.

Bidang 1	Bidang 2	Bidang n
Nama gambar atau jalur Amazon S3 pada gambar. Misalnya, <code>s3://my-bucket/flowers/train/camellia1.jpg</code> . Anda tidak dapat memiliki campuran gambar dengan jalur Amazon S3 dan gambar tanpa.	Label tingkat gambar pertama untuk gambar.	Satu atau lebih label tingkat gambar tambahan dipisahkan dengan koma. Tambahkan hanya jika Anda ingin membuat file manifes yang mendukung <a href="#">klasifikasi gambar Multi-label</a> .

Misalnya `camellia1.jpg,camellia,with_leaves` atau `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`

2. Simpan file CSV.
3. Jalankan skrip Python berikut. Berikan argumen berikut:
  - `csv_file`— File CSV yang Anda buat di langkah 1.
  - `manifest_file`— Nama file manifes yang ingin Anda buat.
  - (Opsional) `--s3_path s3://path_to_folder/` - Jalur Amazon S3 untuk ditambahkan ke nama file gambar (bidang 1). Gunakan `--s3_path` jika gambar di bidang 1 belum berisi jalur S3.

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

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

```
import json

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service documentation.
Shows how to create an image-level (classification) manifest file from a CSV file.
You can specify multiple image level labels per image.
CSV file format is
image,label,label,..
If necessary, use the bucket argument to specify the S3 bucket folder for the
images.
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-gt-cl-
transform.html
"""

logger = logging.getLogger(__name__)

def check_duplicates(csv_file, deduplicated_file, duplicates_file):
    """
    Checks for duplicate images in a CSV file. If duplicate images
    are found, deduplicated_file is the deduplicated CSV file - only the first
    occurrence of a duplicate is recorded. Other duplicates are recorded in
    duplicates_file.
    :param csv_file: The source CSV file.
    :param deduplicated_file: The deduplicated CSV file to create. If no duplicates
    are found
    this file is removed.
    :param duplicates_file: The duplicate images CSV file to create. If no
    duplicates are found
    this file is removed.
    :return: True if duplicates are found, otherwise false.
    """

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

    duplicates_found = False

    # Find duplicates.
    with open(csv_file, 'r', newline='', encoding="UTF-8") as f,\
        open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
        open(duplicates_file, 'w', encoding="UTF-8") as duplicates:

        reader = csv.reader(f, delimiter=',')
```

```
dedup_writer = csv.writer(dedup)
duplicates_writer = csv.writer(duplicates)

entries = set()
for row in reader:
    # Skip empty lines.
    if not ''.join(row).strip():
        continue

    key = row[0]
    if key not in entries:
        dedup_writer.writerow(row)
        entries.add(key)
    else:
        duplicates_writer.writerow(row)
        duplicates_found = True

if duplicates_found:
    logger.info("Duplicates found check %s", duplicates_file)

else:
    os.remove(duplicates_file)
    os.remove(deduplicated_file)

return duplicates_found

def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Reads a CSV file and creates a Custom Labels classification manifest file.
    :param csv_file: The source CSV file.
    :param manifest_file: The name of the manifest file to create.
    :param s3_path: The S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s", csv_file)

    image_count = 0
    label_count = 0

    with open(csv_file, newline='', encoding="UTF-8") as csvfile,\
        open(manifest_file, "w", encoding="UTF-8") as output_file:

        image_classifications = csv.reader(
            csvfile, delimiter=',', quotechar='|')
```

```
# Process each row (image) in CSV file.
for row in image_classifications:
    source_ref = str(s3_path)+row[0]

    image_count += 1

    # Create JSON for image source ref.
    json_line = {}
    json_line['source-ref'] = source_ref

    # Process each image level label.
    for index in range(1, len(row)):
        image_level_label = row[index]

        # Skip empty columns.
        if image_level_label == '':
            continue
        label_count += 1

    # Create the JSON line metadata.
    json_line[image_level_label] = 1
    metadata = {}
    metadata['confidence'] = 1
    metadata['job-name'] = 'labeling-job/' + image_level_label
    metadata['class-name'] = image_level_label
    metadata['human-annotated'] = "yes"
    metadata['creation-date'] = \
        datetime.now(timezone.utc).strftime('%Y-%m-%dT%H:%M:%S.%f')
    metadata['type'] = "groundtruth/image-classification"

    json_line[f'{image_level_label}-metadata'] = metadata

    # Write the image JSON Line.
    output_file.write(json.dumps(json_line))
    output_file.write('\n')

output_file.close()
logger.info("Finished creating manifest file %s\nImages: %s\nLabels: %s",
            manifest_file, image_count, label_count)

return image_count, label_count
```

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

    parser.add_argument(
        "csv_file", help="The CSV file that you want to process."
    )

    parser.add_argument(
        "--s3_path", help="The S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the S3 path.",
        required=False
    )

def main():

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

    try:

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

        s3_path = args.s3_path
        if s3_path is None:
            s3_path = ''

        # Create file names.
        csv_file = args.csv_file
        file_name = os.path.splitext(csv_file)[0]
        manifest_file = f'{file_name}.manifest'
        duplicates_file = f'{file_name}-duplicates.csv'
        deduplicated_file = f'{file_name}-deduplicated.csv'

        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
                  f"and then update {deduplicated_file}. ")
```

```

        print(f"{deduplicated_file} contains the first occurrence of a
duplicate. "
        "Update as necessary with the correct label information.")
        print(f"Re-run the script with {deduplicated_file}")
    else:
        print("No duplicates found. Creating manifest file.")

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

        print(f"Finished creating manifest file: {manifest_file} \n"
              f"Images: {image_count}\nLabels: {label_count}")

    except FileNotFoundError as err:
        logger.exception("File not found: %s", err)
        print(f"File not found: {err}. Check your input CSV file.")

if __name__ == "__main__":
    main()

```

4. Jika Anda berencana menggunakan kumpulan data pengujian, ulangi langkah 1-3 untuk membuat file manifes untuk kumpulan data pengujian Anda.
5. Jika perlu, salin gambar ke jalur bucket Amazon S3 yang Anda tentukan di kolom 1 file CSV (atau ditentukan dalam `--s3_path` baris perintah). Anda dapat menggunakan perintah AWS S3 berikut.

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

6. [Unggah file manifes Anda](#) ke bucket Amazon S3 yang ingin Anda gunakan untuk menyimpan file manifes.

#### Note

Pastikan Label Kustom Amazon Rekognition memiliki akses ke bucket Amazon S3 yang direferensikan di bidang baris JSON file `source-ref` manifes. Untuk informasi selengkapnya, lihat [Mengakses Bucket Amazon S3 eksternal](#). Jika lowongan Ground

Truth menyimpan gambar di Bucket Konsol Label Kustom Amazon Rekognition, Anda tidak perlu menambahkan izin.

- Ikuti petunjuk di [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(Console\)](#) untuk membuat kumpulan data dengan file manifes yang diunggah. Untuk langkah 8, di lokasi file.manifest, masukkan URL Amazon S3 untuk lokasi file manifes. Jika Anda menggunakan AWS SDK, lakukan [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).

## Menyalin konten dari kumpulan data yang ada

Jika sebelumnya Anda telah membuat kumpulan data, Anda dapat menyalin kontennya ke kumpulan data baru. Untuk membuat kumpulan data dari kumpulan data yang ada dengan AWS SDK, lihat.

[Membuat kumpulan data menggunakan dataset \(SDK\) yang ada](#)

Untuk membuat kumpulan data menggunakan dataset Amazon Rekognition Custom Labels (konsol) yang ada

- Buka konsol Amazon Rekognition di. <https://console.aws.amazon.com/rekognition/>
- Pilih Gunakan Label Kustom.
- Pilih Mulai.
- Di panel navigasi kiri, pilih Proyek.
- Di halaman Proyek, pilih proyek yang ingin Anda tambahkan dataset. Halaman detail untuk proyek Anda ditampilkan.
- Pilih Buat kumpulan data. Halaman Buat dataset ditampilkan.
- Dalam konfigurasi Mulai, pilih Mulai dengan satu set data atau Mulai dengan kumpulan data pelatihan. Untuk membuat model berkualitas lebih tinggi, kami sarankan memulai dengan kumpulan data pelatihan dan pengujian terpisah.

### Single dataset

- Di bagian Detail kumpulan data Pelatihan, pilih Salin kumpulan data Label Kustom Rekognition Amazon yang ada.
- Di bagian Detail kumpulan data pelatihan, di kotak edit Dataset, ketik atau pilih nama kumpulan data yang ingin Anda salin.
- Pilih Buat Dataset. Halaman kumpulan data untuk proyek Anda terbuka.

## Separate training and test datasets

- a. Di bagian Detail kumpulan data Pelatihan, pilih Salin kumpulan data Label Kustom Rekognition Amazon yang ada.
- b. Di bagian Detail kumpulan data pelatihan, di kotak edit Dataset, ketik atau pilih nama kumpulan data yang ingin Anda salin.
- c. Di bagian Uji detail kumpulan data, pilih Salin kumpulan data Label Kustom Rekognition Amazon yang ada.
- d. Di bagian Uji detail kumpulan data, di kotak edit Dataset, ketik atau pilih nama kumpulan data yang ingin Anda salin.

### Note

Kumpulan data pelatihan dan pengujian Anda dapat memiliki sumber gambar yang berbeda.

- e. Pilih Buat Kumpulan Data. Halaman kumpulan data untuk proyek Anda terbuka.
8. Jika Anda perlu menambahkan atau mengubah label, lakukan [Pelabelan gambar](#).
  9. Ikuti langkah-langkah [Melatih model \(Konsol\)](#) untuk melatih model Anda.

## Pelabelan gambar

Label mengidentifikasi objek, adegan, konsep, atau kotak pembatas di sekitar objek dalam gambar. Misalnya, jika kumpulan data Anda berisi gambar anjing-kucing, Anda dapat menambahkan label untuk trah anjing-anjingnya.

Setelah mengimpor gambar ke dalam kumpulan data, Anda mungkin perlu menambahkan label ke gambar atau memperbaiki gambar yang salah label. Misalnya, gambar tidak diberi label jika diimpor dari komputer lokal. Anda menggunakan galeri kumpulan data untuk menambahkan label baru ke kumpulan data dan menetapkan label dan kotak pembatas ke gambar dalam kumpulan data.

Cara Anda memberi label pada gambar dalam kumpulan data menentukan jenis model yang dilatih Label Kustom Rekognition Amazon. Untuk informasi selengkapnya, lihat [Mengarahkan kumpulan data](#).

Topik

- [Mengelola label](#)
- [Menetapkan label tingkat gambar ke gambar](#)
- [Pelabelan objek dengan kotak pembatas](#)

## Mengelola label

Anda dapat mengelola label dengan menggunakan konsol Amazon Rekognition Custom Labels. Tidak ada API khusus untuk mengelola label — label ditambahkan ke kumpulan data saat Anda membuat kumpulan data dengan `CreateDataset` atau saat Anda menambahkan lebih banyak gambar ke kumpulan data dengan `UpdateDatasetEntries`.

### Topik

- [Mengelola label \(Konsol\)](#)
- [Mengelola Label \(SDK\)](#)

### Mengelola label (Konsol)

Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk menambah, mengubah, atau menghapus label dari kumpulan data. Untuk menambahkan label ke kumpulan data, Anda dapat menambahkan label baru yang Anda buat atau impor label dari kumpulan data yang ada di Rekognition.

### Topik

- [Tambahkan label baru \(Konsol\)](#)
- [Ubah dan hapus label \(Konsol\)](#)

### Tambahkan label baru (Konsol)

Anda dapat menentukan label baru yang ingin Anda tambahkan ke kumpulan data Anda.

Tambahkan label menggunakan jendela pengeditan

Untuk menambahkan label baru (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.

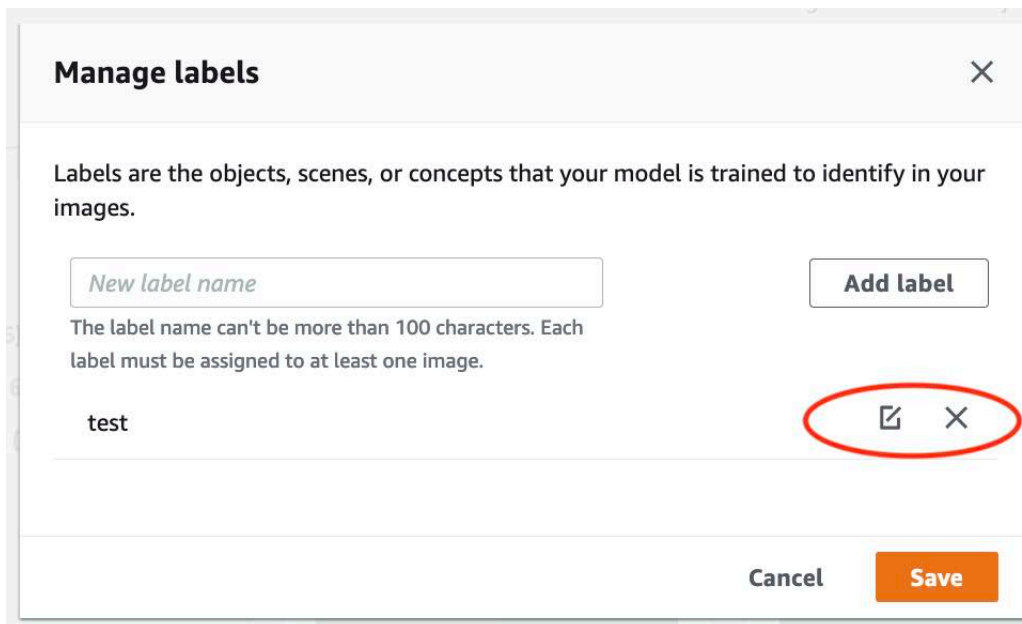
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Di halaman Proyek, pilih proyek yang ingin Anda gunakan. Halaman detail untuk proyek Anda ditampilkan.
6. Jika Anda ingin menambahkan label ke kumpulan data pelatihan, pilih tab Pelatihan. Jika tidak, pilih tab Uji untuk menambahkan label ke kumpulan data pengujian.
7. Pilih Mulai pelabelan untuk masuk ke mode pelabelan.
8. Di bagian Label galeri kumpulan data, pilih Kelola label untuk membuka kotak dialog Kelola label.
9. Di kotak edit, masukkan nama label baru.
10. Pilih Tambahkan label.
11. Ulangi langkah 9 dan 10 sampai Anda telah membuat semua label yang Anda butuhkan.
12. Pilih Simpan untuk menyimpan label yang Anda tambahkan.

### Ubah dan hapus label (Konsol)

Anda dapat mengganti nama atau menghapus label setelah menambahkannya ke kumpulan data. Anda hanya dapat menghapus label yang tidak ditetapkan ke gambar apa pun.

Untuk mengganti nama atau menghapus label yang ada (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Di halaman Proyek, pilih proyek yang ingin Anda gunakan. Halaman detail untuk proyek Anda ditampilkan.
6. Jika Anda ingin mengubah atau menghapus label dalam kumpulan data pelatihan, pilih tab Pelatihan. Jika tidak, pilih tab Uji untuk mengubah atau menghapus label ke kumpulan data pengujian.
7. Pilih Mulai pelabelan untuk masuk ke mode pelabelan.
8. Di bagian Label galeri kumpulan data, pilih Kelola label untuk membuka kotak dialog Kelola label.
9. Pilih label yang ingin Anda edit atau hapus.



- a. Jika Anda memilih ikon hapus (X), label akan dihapus dari daftar.
- b. Jika Anda ingin mengubah label, pilih ikon edit (pensil dan paper pad) dan masukkan nama label baru di kotak edit.

10. Pilih Simpan untuk menyimpan perubahan Anda.

## Mengelola Label (SDK)

Tidak ada API unik yang mengelola label dataset. Jika Anda membuat kumpulan data dengan `CreateDataset`, label yang ditemukan di file manifes atau kumpulan data yang disalin, buat kumpulan label awal. Jika Anda menambahkan lebih banyak gambar dengan `UpdateDatasetEntries` API, label baru yang ditemukan di entri akan ditambahkan ke kumpulan data. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar \(SDK\)](#). Untuk menghapus label dari kumpulan data, Anda harus menghapus semua anotasi label dalam kumpulan data.

Untuk menghapus label dari kumpulan data

1. Hubungi `ListDatasetEntries` untuk mendapatkan entri dataset. Untuk kode sampel, lihat [Daftar entri kumpulan data \(SDK\)](#).
2. Dalam file, hapus anotasi label apa pun. Untuk informasi selengkapnya, lihat [Mengimpor label tingkat gambar dalam file manifes](#) dan [the section called "Lokalisasi objek dalam file manifes"](#).

- Gunakan file untuk memperbarui kumpulan data dengan UpdateDatasetEntries API. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar \(SDK\)](#).

## Menetapkan label tingkat gambar ke gambar

Anda menggunakan label tingkat gambar untuk melatih model yang mengklasifikasikan gambar ke dalam kategori. Label tingkat gambar menunjukkan bahwa gambar berisi objek, adegan, atau konsep. Misalnya, gambar berikut menunjukkan sungai. Jika model Anda mengklasifikasikan gambar sebagai berisi sungai, Anda akan menambahkan label tingkat gambar sungai. Untuk informasi selengkapnya, lihat [Mengarahkan kumpulan data](#).



Dataset yang berisi label tingkat gambar, membutuhkan setidaknya dua label yang ditentukan. Setiap gambar membutuhkan setidaknya satu label yang ditetapkan yang mengidentifikasi objek, adegan, atau konsep dalam gambar.

Untuk menetapkan label tingkat gambar ke gambar (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Di halaman Proyek, pilih proyek yang ingin Anda gunakan. Halaman detail untuk proyek Anda ditampilkan.
6. Di panel navigasi kiri, pilih Dataset.
7. Jika Anda ingin menambahkan label ke kumpulan data pelatihan, pilih tab Pelatihan. Jika tidak, pilih tab Uji untuk menambahkan label ke kumpulan data pengujian.
8. Pilih Mulai pelabelan untuk masuk ke mode pelabelan.
9. Di galeri gambar, pilih satu atau beberapa gambar yang ingin Anda tambahkan labelnya. Anda hanya dapat memilih gambar pada satu halaman pada satu waktu. Untuk memilih rentang gambar yang berdekatan pada halaman:
  - a. Pilih gambar pertama dalam kisaran.
  - b. Tekan dan tahan tombol shift.
  - c. Pilih rentang gambar terakhir. Gambar antara gambar pertama dan kedua juga dipilih.
  - d. Lepaskan tombol shift.
10. Pilih Tetapkan label tingkat gambar.
11. Dalam kotak dialog Tetapkan label tingkat gambar ke gambar yang dipilih, pilih label yang ingin Anda tetapkan ke gambar atau gambar.
12. Pilih Tetapkan untuk menetapkan label pada gambar.
13. Ulangi pelabelan hingga setiap gambar dianotasi dengan label yang diperlukan.
14. Pilih Simpan perubahan untuk menyimpan perubahan Anda.

Tetapkan label tingkat gambar (SDK)

Anda dapat menggunakan `UpdateDatasetEntries` API untuk menambahkan atau memperbarui label tingkat gambar yang ditetapkan ke gambar. `UpdateDatasetEntries` mengambil satu atau lebih baris JSON. Setiap JSON Line mewakili satu gambar. Untuk gambar dengan label tingkat gambar, Garis JSON terlihat mirip dengan yang berikut ini.

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

source-refBidang menunjukkan lokasi gambar. Garis JSON juga menyertakan label tingkat gambar yang ditetapkan ke gambar. Untuk informasi selengkapnya, lihat [the section called “Mengimpor label tingkat gambar dalam file manifes”](#).

Untuk menetapkan label tingkat gambar ke gambar

1. Dapatkan JSON Line get untuk gambar yang ada dengan menggunakan `ListDatasetEntries` Untuk source-ref bidang, tentukan lokasi gambar yang ingin Anda tetapkan labelnya. Untuk informasi selengkapnya, lihat [Daftar entri kumpulan data \(SDK\)](#).
2. Perbarui JSON Line yang dikembalikan pada langkah sebelumnya menggunakan informasi di [Mengimpor label tingkat gambar dalam file manifes](#).
3. Panggil `UpdateDatasetEntries` untuk memperbarui gambar. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#).

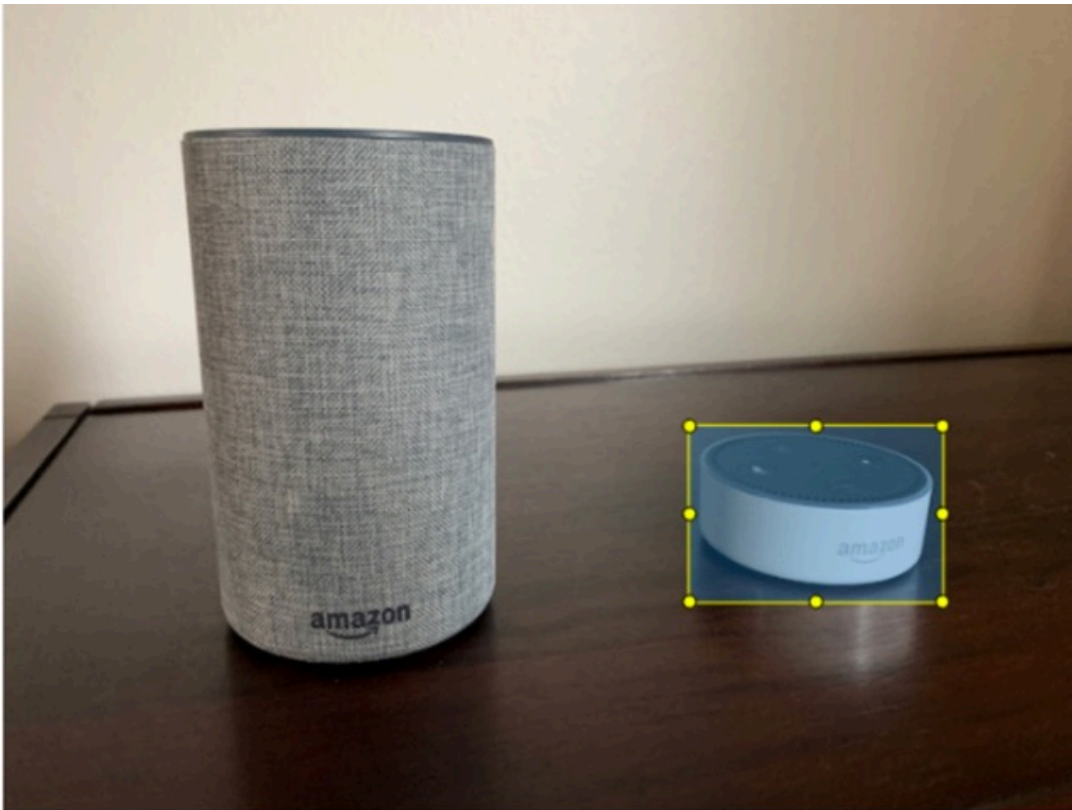
## Pelabelan objek dengan kotak pembatas

Jika Anda ingin model Anda mendeteksi lokasi objek dalam gambar, Anda harus mengidentifikasi apa objek itu dan di mana objek itu berada dalam gambar. Kotak pembatas adalah kotak yang mengisolasi objek dalam gambar. Anda menggunakan kotak pembatas untuk melatih model untuk mendeteksi objek yang berbeda dalam gambar yang sama. Anda mengidentifikasi objek dengan menetapkan label ke kotak pembatas.

### Note

Jika Anda melatih model untuk menemukan objek, adegan, dan konsep dengan label tingkat gambar, Anda tidak perlu melakukan langkah ini.

Misalnya, jika Anda ingin melatih model yang mendeteksi perangkat Amazon Echo Dot, Anda menggambar kotak pembatas di sekitar setiap Echo Dot dalam gambar dan menetapkan label bernama Echo Dot ke kotak pembatas. Gambar berikut menunjukkan kotak pembatas di sekitar perangkat Echo Dot. Gambar juga berisi Amazon Echo tanpa kotak pembatas.



Temukan objek dengan kotak pembatas (Konsol)

Dalam prosedur ini, Anda menggunakan konsol untuk menggambar kotak pembatas di sekitar objek dalam gambar Anda. Anda juga dapat mengidentifikasi objek dalam gambar dengan menetapkan label ke kotak pembatas.

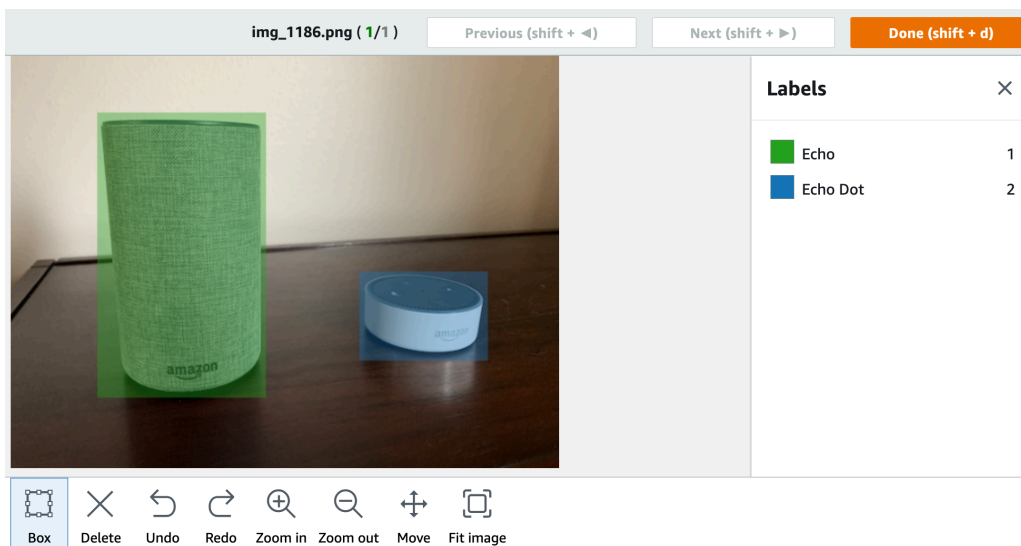
**Note**

Anda tidak dapat menggunakan browser Safari untuk menambahkan kotak pembatas ke gambar. Untuk browser yang didukung, lihat [Menyiapkan Label Kustom Rekognition Amazon](#).

Sebelum Anda dapat menambahkan kotak pembatas, Anda harus menambahkan setidaknya satu label ke kumpulan data. Untuk informasi selengkapnya, lihat [Tambahkan label baru \(Konsol\)](#).

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.

5. Di halaman Proyek, pilih proyek yang ingin Anda gunakan. Halaman detail untuk proyek Anda ditampilkan.
6. Pada halaman detail proyek, pilih gambar Label
7. Jika Anda ingin menambahkan kotak pembatas ke gambar kumpulan data pelatihan Anda, pilih tab Pelatihan. Jika tidak, pilih tab Uji untuk menambahkan kotak pembatas ke gambar kumpulan data pengujian.
8. Pilih Mulai pelabelan untuk masuk ke mode pelabelan.
9. Di galeri gambar, pilih gambar yang ingin Anda tambahkan kotak pembatas.
10. Pilih Draw bounding box. Serangkaian tips ditampilkan sebelum editor kotak pembatas ditampilkan.
11. Di panel Label di sebelah kanan, pilih label yang ingin Anda tetapkan ke kotak pembatas.
12. Dalam alat gambar, letakkan pointer Anda di area kiri atas objek yang diinginkan.
13. Tekan tombol kiri mouse dan gambar kotak di sekitar objek. Cobalah untuk menggambar kotak pembatas sedekat mungkin dengan objek.
14. Lepaskan tombol mouse. Kotak pembatas disorot.
15. Pilih Berikutnya jika Anda memiliki lebih banyak gambar untuk diberi label. Jika tidak, pilih Selesai untuk menyelesaikan pelabelan.



16. Ulangi langkah 1-7 sampai Anda telah membuat kotak pembatas di setiap gambar yang berisi objek.
17. Pilih Simpan perubahan untuk menyimpan perubahan Anda.
18. Pilih Keluar untuk keluar dari mode pelabelan.

## Temukan objek dengan kotak pembatas (SDK)

Anda dapat menggunakan `UpdateDatasetEntries` API untuk menambahkan atau memperbarui informasi lokasi objek untuk gambar. `UpdateDatasetEntries` mengambil satu atau lebih baris JSON. Setiap JSON Line mewakili satu gambar. Untuk lokalisasi objek, JSON Line terlihat mirip dengan berikut ini.

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

`source-ref` Bidang menunjukkan lokasi gambar. Garis JSON juga menyertakan kotak pembatas berlabel untuk setiap objek pada gambar. Untuk informasi selengkapnya, lihat [the section called “Lokalisasi objek dalam file manifes”](#).

Untuk menetapkan kotak pembatas ke gambar

1. Dapatkan JSON Line get untuk gambar yang ada dengan menggunakan `ListDatasetEntries` Untuk `source-ref` bidang, tentukan lokasi gambar yang ingin Anda tetapkan label tingkat gambar. Untuk informasi selengkapnya, lihat [Daftar entri kumpulan data \(SDK\)](#).
2. Perbarui JSON Line yang dikembalikan pada langkah sebelumnya menggunakan informasi di [Lokalisasi objek dalam file manifes](#).
3. Panggil `UpdateDatasetEntries` untuk memperbarui gambar. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#).

## Debugging kumpulan data

Selama pembuatan dataset ada dua jenis kesalahan yang dapat terjadi - kesalahan terminal dan kesalahan non-terminal. Kesalahan terminal dapat menghentikan pembuatan atau pembaruan kumpulan data. Kesalahan non-terminal tidak menghentikan pembuatan atau pembaruan kumpulan data.

### Topik

- [Kesalahan kumpulan data terminal debugging](#)
- [Mendebug kesalahan kumpulan data non-terminal](#)

## Kesalahan kumpulan data terminal debugging

Ada dua jenis kesalahan terminal — kesalahan file yang menyebabkan pembuatan kumpulan data gagal, dan kesalahan konten yang dihapus oleh Label Kustom Rekognition Amazon dari kumpulan data. Pembuatan dataset gagal jika ada terlalu banyak kesalahan konten.

### Topik

- [Kesalahan file terminal](#)
- [Kesalahan konten terminal](#)

### Kesalahan file terminal

Berikut ini adalah kesalahan file. Anda bisa mendapatkan informasi tentang kesalahan file dengan menelepon `DescribeDataset` dan memeriksa `Status` dan `StatusMessage` bidang. Untuk kode sampel, lihat [Menjelaskan kumpulan data \(SDK\)](#).

- [ERROR\\_MANIFEST\\_INACCESSIBLE\\_OR\\_UNSUPPORTED\\_FORMAT](#)
- [ERROR\\_MANIFEST\\_SIZE\\_TOO\\_LARGE](#).
- [ERROR\\_MANIFEST\\_ROWS\\_EXCEEDS\\_MAXIMUM](#)
- [ERROR\\_INVALID\\_PERMISSIONS\\_MANIFEST\\_S3\\_BUCKET](#)
- [ERROR\\_TOO\\_MANY\\_RECORDS\\_IN\\_ERROR](#)
- [ERROR\\_MANIFEST\\_TOO\\_MANY\\_LABELS](#)
- [ERROR\\_INSUFFICIENT\\_IMAGES\\_PER\\_LABEL\\_FOR\\_DISTRIBUSIKAN](#)

`ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT`

### Pesan kesalahan

Ekstensi atau konten file manifes tidak valid.

File manifes pelatihan atau pengujian tidak memiliki ekstensi file atau isinya tidak valid.

Untuk memperbaiki kesalahan

## ERROR\_MANIFEST\_INACCESSIBLE\_OR\_UNSUPPORTED\_FORMAT

- Periksa kemungkinan penyebab berikut dalam pelatihan dan pengujian file manifes.
  - File manifes tidak memiliki ekstensi file. Dengan konvensi ekstensi file adalah `.manifest`.
  - Bucket atau kunci Amazon S3 untuk file manifes tidak dapat ditemukan.

## ERROR\_MANIFEST\_SIZE\_TOO\_LARGE

Pesan kesalahan

Ukuran file manifes melebihi ukuran maksimum yang didukung.

Ukuran file manifes pelatihan atau pengujian (dalam byte) terlalu besar. Untuk informasi selengkapnya, lihat [Pedoman dan kuota di Label Kustom Rekognition Amazon](#). File manifes dapat memiliki kurang dari jumlah maksimum JSON Lines dan masih melebihi ukuran file maksimum.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan Ukuran file manifes melebihi ukuran maksimum yang didukung.

Untuk memperbaiki kesalahan ERROR\_MANIFEST\_SIZE\_TOO\_LARGE

1. Periksa manifes pelatihan dan pengujian mana yang melebihi ukuran file maksimum.
2. Kurangi jumlah JSON Lines dalam file manifes yang terlalu besar. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

## ERROR\_MANIFEST\_ROWS\_EXCEEDS\_MAXIMUM

Pesan kesalahan

File manifes memiliki terlalu banyak baris.

Informasi selengkapnya

Jumlah Garis JSON (jumlah gambar) dalam file manifes lebih besar dari batas yang diizinkan. Batasnya berbeda untuk model tingkat gambar dan model lokasi objek. Untuk informasi selengkapnya, lihat [Pedoman dan kuota di Label Kustom Rekognition Amazon](#).

Kesalahan JSON Line divalidasi hingga jumlah JSON Lines mencapai batas.

## ERROR\_MANIFEST\_ROWS\_EXCEEDS\_MAXIMUM

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan. `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`

Untuk memperbaiki **`ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`**

- Kurangi jumlah Garis JSON dalam manifes. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

`ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET`

Pesan kesalahan

Izin bucket S3 salah.

Label Kustom Rekognition Amazon tidak memiliki izin ke satu atau beberapa bucket yang berisi file manifes pelatihan dan pengujian.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

Untuk memperbaiki kesalahan `ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET`

- Periksa izin untuk bucket yang berisi manifes pelatihan dan pengujian. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan izin konsol Amazon Rekognition Custom Labels](#).

`ERROR_TOO_MANY_RECORDS_IN_ERROR`

Pesan kesalahan

File manifes memiliki terlalu banyak kesalahan terminal.

Untuk memperbaiki **`ERROR_TOO_MANY_RECORDS_IN_ERROR`**

- Kurangi jumlah Garis JSON (gambar) dengan kesalahan konten terminal. Untuk informasi selengkapnya, lihat [Kesalahan konten manifes terminal](#).

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_MANIFEST\_TOO\_MANY\_LABELS

### Pesan kesalahan

File manifes memiliki terlalu banyak label.

### Informasi selengkapnya

Jumlah label unik dalam manifes (dataset) lebih dari batas yang diizinkan. Jika kumpulan data pelatihan dibagi untuk membuat kumpulan data pengujian, jumlah label ditentukan setelah pemisahan.

Untuk memperbaiki ERROR\_MANIFEST\_TOO\_MANY\_LABELS (Konsol)

- Hapus label dari kumpulan data. Untuk informasi selengkapnya, lihat [Mengelola label](#). Label secara otomatis dihapus dari gambar dan kotak pembatas di kumpulan data Anda.

Untuk memperbaiki ERROR\_MANIFEST\_TOO\_MANY\_LABELS (JSON Line)

- Manifestasi dengan garis JSON tingkat gambar - Jika gambar memiliki label tunggal, hapus Garis JSON untuk gambar yang menggunakan label yang diinginkan. Jika JSON Line berisi beberapa label, hapus hanya objek JSON untuk label yang diinginkan. Untuk informasi selengkapnya, lihat [Menambahkan beberapa label tingkat gambar ke gambar](#).

Manifestasi dengan lokasi objek JSON Lines - Hapus kotak pembatas dan informasi label terkait untuk label yang ingin Anda hapus. Lakukan ini untuk setiap JSON Line yang berisi label yang diinginkan. Anda perlu menghapus label dari `class-map` array dan objek yang sesuai dalam `objects` dan `annotations` array. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

## ERROR\_INSUFFICIENT\_IMAGES\_PER\_LABEL\_FOR\_DISTRIBUSIKAN

### Pesan kesalahan

File manifes tidak memiliki cukup gambar berlabel untuk mendistribusikan kumpulan data.

Distribusi kumpulan data terjadi saat Amazon Rekognition Custom Labels membagi kumpulan data pelatihan untuk membuat kumpulan data pengujian. Anda juga dapat membagi kumpulan data dengan memanggil `DistributeDatasetEntries` API.

## Untuk memperbaiki kesalahan ERROR\_MANIFEST\_TOO\_MANY\_LABELS

- Tambahkan lebih banyak gambar berlabel ke kumpulan data pelatihan

### Kesalahan konten terminal

Berikut ini adalah kesalahan konten terminal. Selama pembuatan kumpulan data, gambar yang memiliki kesalahan konten terminal dihapus dari kumpulan data. Dataset masih dapat digunakan untuk pelatihan. Jika ada terlalu banyak kesalahan konten, dataset/update gagal. Kesalahan konten terminal yang terkait dengan operasi kumpulan data tidak ditampilkan di konsol atau dikembalikan dari DescribeDataset atau API lainnya. Jika Anda melihat bahwa gambar atau anotasi hilang dari kumpulan data Anda, periksa file manifes kumpulan data Anda untuk masalah berikut:

- Panjang garis JSON terlalu panjang. Panjang maksimum adalah 100.000 karakter.
- `source-ref` Nilai hilang dari JSON Line.
- Format `source-ref` nilai dalam JSON Line tidak valid.
- Isi dari JSON Line tidak valid.
- Nilai `source-ref` bidang muncul lebih dari sekali. Gambar hanya dapat direferensikan sekali dalam kumpulan data.

Untuk informasi tentang `source-ref` lapangan, lihat [Membuat file manifes](#).

### Mendebug kesalahan kumpulan data non-terminal

Berikut ini adalah kesalahan non-terminal yang dapat terjadi selama pembuatan atau pembaruan dataset. Kesalahan ini dapat membatalkan seluruh JSON Line atau membatalkan anotasi dalam JSON Line. Jika JSON Line memiliki kesalahan, itu tidak digunakan untuk pelatihan. Jika anotasi dalam JSON Line memiliki kesalahan, JSON Line masih digunakan untuk pelatihan, tetapi tanpa anotasi yang rusak. Untuk informasi lebih lanjut tentang JSON Lines, lihat [Membuat file manifes](#).

Anda dapat mengakses kesalahan non-terminal dari konsol dan dengan memanggil `ListDatasetEntries` API. Untuk informasi selengkapnya, lihat [Daftar entri kumpulan data \(SDK\)](#).

Kesalahan berikut juga dikembalikan selama pelatihan. Kami menyarankan Anda memperbaiki kesalahan ini sebelum melatih model Anda. Untuk informasi lebih lanjut, lihat [Kesalahan Validasi Jalur JSON Non-Terminal](#)

- [ERROR\\_NO\\_LABEL\\_ATTRIBUTES](#)

- [ERROR\\_INVALID\\_LABEL\\_ATTRIBUTE\\_FORMAT](#)
- [ERROR\\_INVALID\\_LABEL\\_ATTRIBUTE\\_METADATA\\_FORMAT](#)
- [ERROR\\_NO\\_VALID\\_LABEL\\_ATTRIBUTES](#)
- [ERROR\\_INVALID\\_BOUNDING\\_BOX](#)
- [ERROR\\_INVALID\\_IMAGE\\_DIMENSION](#)
- [ERROR\\_BOUNDING\\_BOX\\_TOO\\_SMALL](#)
- [ERROR\\_NO\\_VALID\\_ANNOTATIONS](#)
- [ERROR\\_MISSING\\_BOUNDING\\_BOX\\_CONFIDENCE](#)
- [ERROR\\_MISSING\\_CLASS\\_MAP\\_ID](#)
- [ERROR\\_TOO\\_MANY\\_BOUNDING\\_BOXES](#)
- [ERROR\\_UNSUPPORTED\\_USE\\_CASE\\_TYPE](#)
- [ERROR\\_INVALID\\_LABEL\\_NAME\\_LENGTH](#)

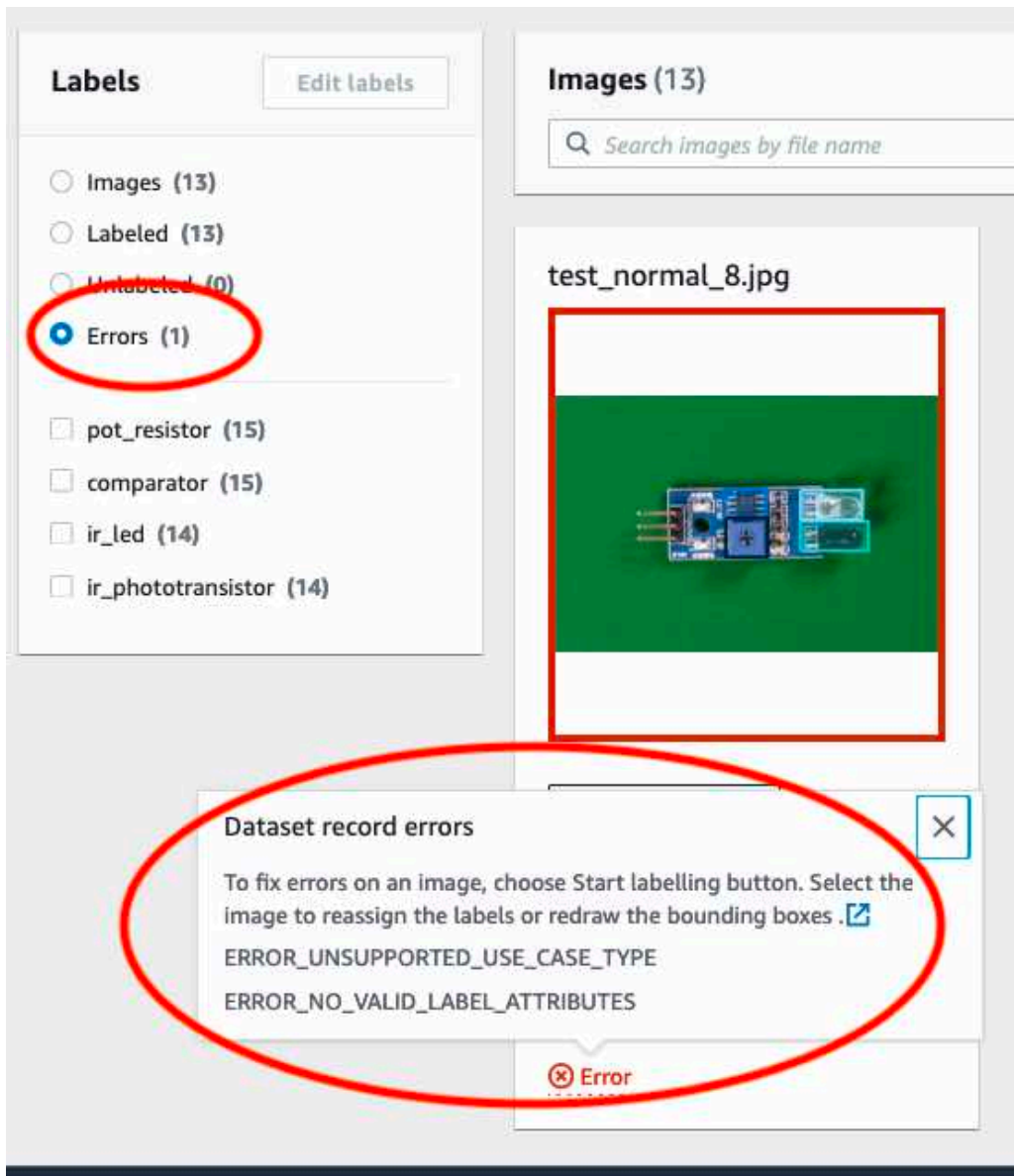
## Mengakses kesalahan non-terminal

Anda dapat menggunakan konsol untuk mengetahui gambar mana dalam kumpulan data yang memiliki kesalahan non-terminal. Anda juga dapat memanggil, memanggil `ListDatasetEntries` API untuk mendapatkan pesan kesalahan. Untuk informasi selengkapnya, lihat [Daftar entri kumpulan data \(SDK\)](#).

Untuk mengakses kesalahan non-terminal (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Di halaman Proyek, pilih proyek yang ingin Anda gunakan. Halaman detail untuk proyek Anda ditampilkan.
6. Jika Anda ingin melihat kesalahan non-terminal dalam kumpulan data pelatihan, pilih tab Pelatihan. Jika tidak, pilih tab Uji untuk melihat kesalahan non-terminal dalam kumpulan data pengujian Anda.
7. Di bagian Label galeri kumpulan data, pilih Kesalahan. Galeri dataset difilter untuk hanya menampilkan gambar dengan kesalahan.

- Pilih Kesalahan di bawah gambar untuk melihat kode kesalahan. Gunakan informasi di [Kesalahan Validasi Jalur JSON Non-Terminal](#) untuk memperbaiki kesalahan.



## Melatih model Label Kustom Rekognition Amazon

Anda dapat melatih model dengan menggunakan konsol Amazon Rekognition Custom Labels, atau dengan Amazon Rekognition Custom Labels API. Jika pelatihan model gagal, gunakan informasi [Mendebug pelatihan model yang gagal](#) untuk menemukan penyebab kegagalan.

**Note**

Anda dikenakan biaya untuk jumlah waktu yang diperlukan untuk berhasil melatih model. Biasanya pelatihan membutuhkan waktu 30 menit hingga 24 jam untuk diselesaikan. Untuk informasi selengkapnya, lihat [Jam pelatihan](#).

Versi baru model dibuat setiap kali model dilatih. Amazon Rekognition Custom Labels membuat nama untuk model yang merupakan kombinasi dari nama proyek dan stempel waktu ketika model dibuat.

Untuk melatih model Anda, Label Kustom Amazon Rekognition membuat salinan pelatihan sumber dan tes citra Anda. Secara default, gambar yang disalin dienkripsi saat istirahat dengan kunci yang dimiliki dan dikelola AWS. Anda juga dapat memilih untuk menggunakan milik Anda sendiri AWS KMS key. Jika Anda menggunakan kunci KMS Anda sendiri, Anda memerlukan izin berikut pada kunci KMS.

- km: CreateGrant
- km: DescribeKey

Untuk informasi selengkapnya, lihat [Konsep AWS Key Management Service](#). Citra sumber Anda tidak terpengaruh.

Anda dapat menggunakan enkripsi sisi server KMS (SSE-KMS) untuk mengenkripsi pelatihan dan menguji gambar di bucket Amazon S3 Anda, sebelum disalin oleh Label Kustom Rekognition Amazon. Untuk mengizinkan Label Kustom Amazon Rekognition mengakses gambar Anda, akun AWS Anda memerlukan izin berikut pada kunci KMS.

- km: GenerateDataKey
- kms:Decrypt

Untuk informasi selengkapnya, lihat [Melindungi Data Menggunakan Enkripsi Sisi Server dengan kunci KMS yang Disimpan di AWS Key Management Service \(SSE-KMS\)](#).

Setelah melatih model, Anda dapat mengevaluasi kinerjanya dan melakukan perbaikan. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).

Untuk tugas model lainnya, seperti menandai model, lihat [Mengelola model Label Kustom Rekognition Amazon](#).

## Topik

- [Melatih model \(Konsol\)](#)
- [Melatih model \(SDK\)](#)

## Melatih model (Konsol)

Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk melatih model.

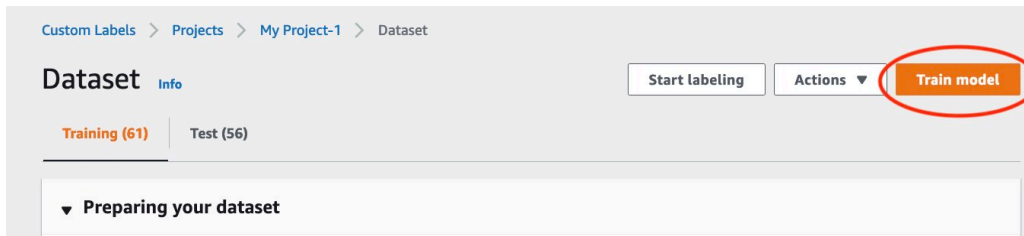
Pelatihan membutuhkan proyek dengan dataset pelatihan dan dataset pengujian. Jika project Anda tidak memiliki kumpulan data pengujian, konsol Amazon Rekognition Custom Labels akan membagi kumpulan data pelatihan selama pelatihan untuk membuatnya untuk project Anda. Gambar yang dipilih adalah pengambilan sampel yang representatif dan tidak digunakan dalam kumpulan data pelatihan. Sebaiknya pisahkan kumpulan data pelatihan Anda hanya jika Anda tidak memiliki kumpulan data pengujian alternatif yang dapat Anda gunakan. Memisahkan kumpulan data pelatihan mengurangi jumlah gambar yang tersedia untuk pelatihan.

### Note

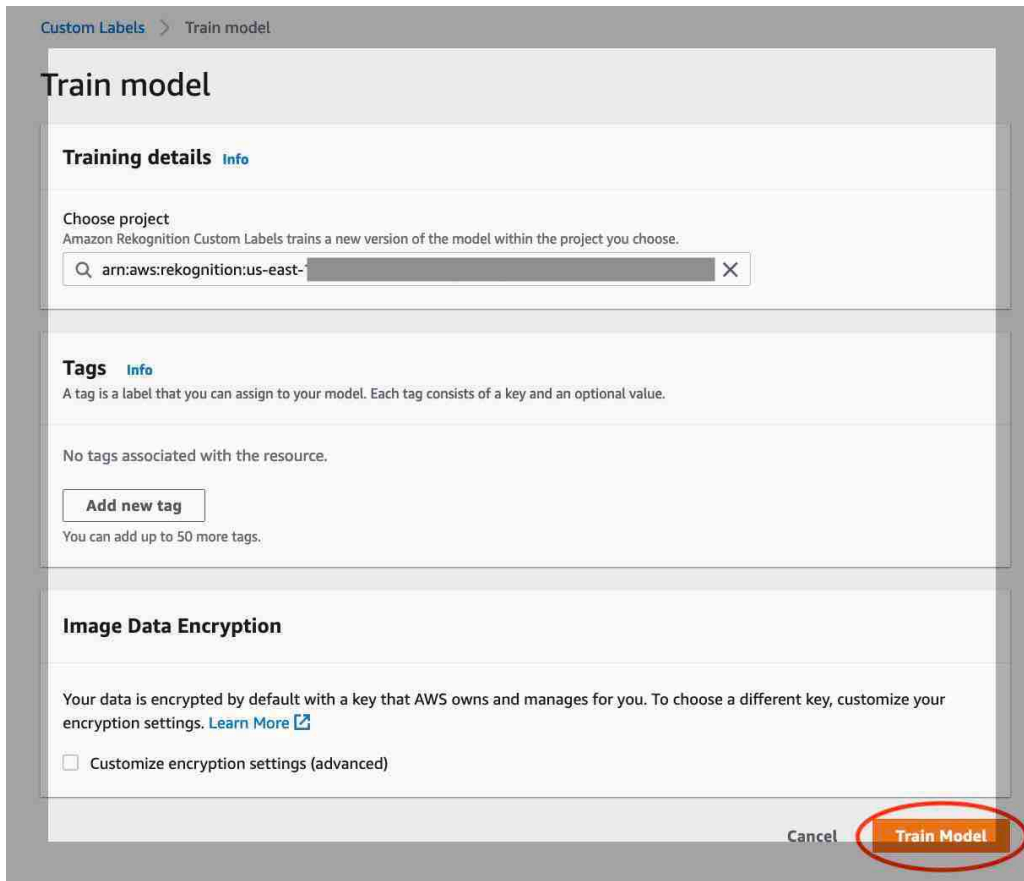
Anda dikenakan biaya untuk jumlah waktu yang diperlukan untuk melatih model. Untuk informasi selengkapnya, lihat [Jam pelatihan](#).

Untuk melatih model Anda (konsol)

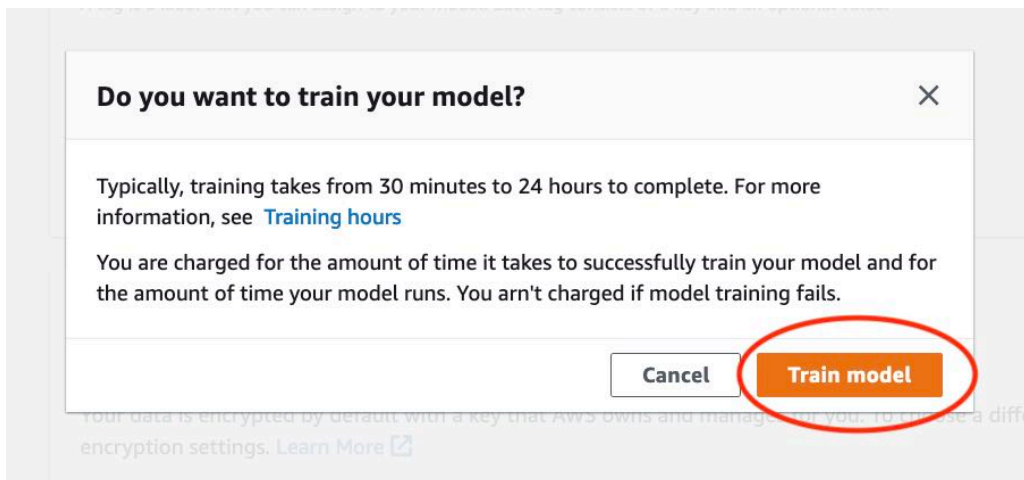
1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Di panel navigasi kiri, pilih Proyek.
4. Di halaman Proyek, pilih proyek yang berisi model yang ingin Anda latih.
5. Pada halaman Proyek, pilih Model kereta.



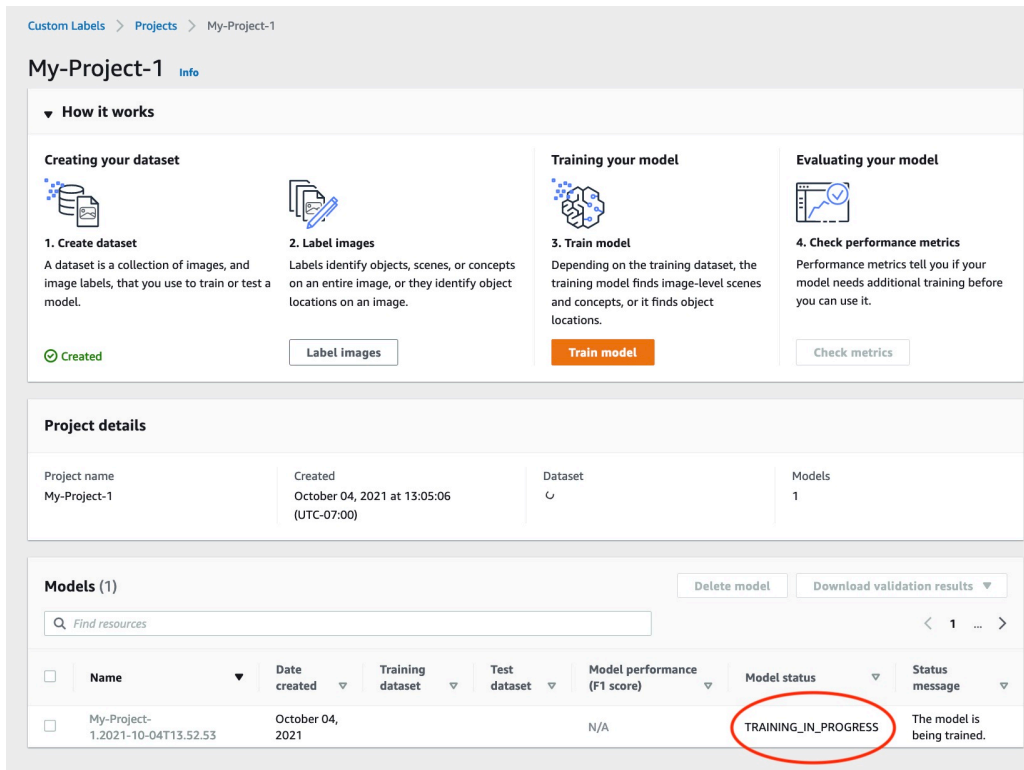
6. (Opsional) Jika Anda ingin menggunakan kunci enkripsi AWS KMS Anda sendiri, lakukan hal berikut:
  - a. Di Enkripsi data gambar pilih Sesuaikan pengaturan enkripsi (lanjutan).
  - b. Di encryption.aws\_kms\_key masukkan Nama Sumber Daya Amazon (ARN) kunci Anda, atau pilih kunci AWS KMS yang ada. Untuk membuat kunci baru, pilih Buat kunci AWS IMS.
7. (Opsional) jika Anda ingin menambahkan tag ke model Anda lakukan hal berikut:
  - a. Di bagian Tag, pilih Tambahkan tag baru.
  - b. Masukkan yang berikut ini:
    - i. Nama kunci di Key.
    - ii. Nilai kunci dalam Nilai.
  - c. Untuk menambahkan lebih banyak tag, ulangi langkah 6a dan 6b.
  - d. (Opsional) Jika Anda ingin menghapus tag, pilih Hapus di samping tag yang ingin Anda hapus. Jika Anda menghapus tag yang disimpan sebelumnya, tag tersebut akan dihapus saat Anda menyimpan perubahan.
8. Pada halaman model Kereta, Pilih model Kereta. Nama Sumber Daya Amazon (ARN) untuk proyek Anda harus ada di kotak Pilih edit proyek. Jika tidak, masukkan ARN untuk proyek Anda.



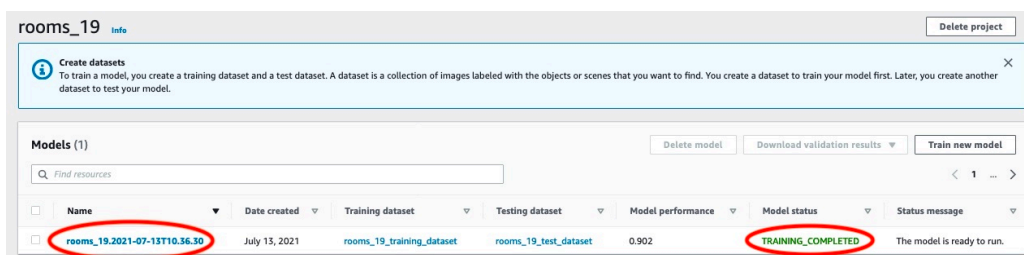
9. Di Apakah Anda ingin melatih model Anda? kotak dialog, pilih Model kereta.



10. Di bagian Model pada halaman proyek, Anda dapat memeriksa status saat ini di Model Status kolom, tempat pelatihan sedang berlangsung. Pelatihan model membutuhkan waktu beberapa saat untuk diselesaikan.



11. Setelah pelatihan selesai, pilih nama model. Pelatihan selesai ketika status model adalah TRAINING\_COMPLETED. Jika pelatihan gagal, baca [Mendebug pelatihan model yang gagal](#).



12. Langkah selanjutnya: Evaluasi model Anda. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).

## Melatih model (SDK)

Anda melatih model dengan menelepon [CreateProjectVersion](#). Untuk melatih model, informasi berikut diperlukan:

- Nama — Nama unik untuk versi model.
- Proyek ARN — Nama Sumber Daya Amazon (ARN) dari proyek yang mengelola model.
- Lokasi hasil pelatihan — Lokasi Amazon S3 tempat hasil ditempatkan. Anda dapat menggunakan lokasi yang sama dengan bucket Amazon S3 konsol, atau Anda dapat memilih lokasi yang

berbeda. Sebaiknya pilih lokasi yang berbeda karena ini memungkinkan Anda menyetel izin dan menghindari potensi konflik penamaan dengan output pelatihan menggunakan konsol Label Kustom Rekognition Amazon.

Pelatihan menggunakan kumpulan data pelatihan dan pengujian yang terkait dengan proyek. Untuk informasi selengkapnya, lihat [Mengelola kumpulan data](#).

#### Note

Secara opsional, Anda dapat menentukan pelatihan dan menguji file manifes kumpulan data yang berada di luar proyek. Jika Anda membuka konsol setelah melatih model dengan file manifes eksternal, Label Kustom Rekognition Amazon akan membuat kumpulan data untuk Anda dengan menggunakan kumpulan file manifes terakhir yang digunakan untuk pelatihan. Anda tidak dapat lagi melatih versi model untuk proyek dengan menentukan file manifes eksternal. Untuk informasi selengkapnya, lihat [CreateProjectVersion](#).

Tanggapan dari `CreateProjectVersion` adalah ARN yang Anda gunakan untuk mengidentifikasi versi model dalam permintaan berikutnya. Anda juga dapat menggunakan ARN untuk mengamankan versi model. Untuk informasi selengkapnya, lihat [Mengamankan proyek Label Kustom Rekognition Amazon](#).

Pelatihan versi model membutuhkan waktu beberapa saat untuk diselesaikan. Contoh Python dan Java dalam topik ini menggunakan pelayan untuk menunggu pelatihan selesai. Pelayan adalah metode utilitas yang melakukan polling untuk keadaan tertentu terjadi. Atau, Anda bisa mendapatkan status pelatihan saat ini dengan `describeProjectVersions`. Pelatihan selesai ketika nilai `Status` lapangan `TRAINING_COMPLETED`. Setelah pelatihan selesai, Anda dapat mengevaluasi kualitas model dengan meninjau hasil evaluasi.

## Melatih model (SDK)

Contoh berikut menunjukkan cara melatih model dengan menggunakan kumpulan data pelatihan dan pengujian yang terkait dengan proyek.

Untuk melatih model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).

## 2. Gunakan kode contoh berikut untuk melatih proyek.

### AWS CLI

Contoh berikut menciptakan model. Dataset pelatihan dibagi untuk membuat dataset pengujian. Ganti yang berikut ini:

- `my_project_arn` dengan Nama Sumber Daya Amazon (ARN) proyek.
- `version_name` dengan nama versi unik yang Anda pilih.
- `output_bucket` dengan nama bucket Amazon S3 tempat Amazon Rekognition Custom Labels menyimpan hasil pelatihan.
- `output_folder` dengan nama folder tempat hasil pelatihan disimpan.
- (parameter opsional) `--kms-key-id` dengan pengenal untuk kunci master pelanggan AWS Key Management Service Anda.

```
aws rekognition create-project-version \  
  --project-arn project_arn \  
  --version-name version_name \  
  --output-config '{"S3Bucket": "output_bucket", "S3KeyPrefix": "output_folder"}' \  
  \  
  --profile custom-labels-access
```

### Python

Contoh berikut menciptakan model. Berikan argumen baris perintah berikut:

- `project_arn`— Nama Sumber Daya Amazon (ARN) dari proyek.
- `version_name`— Nama versi unik untuk model yang Anda pilih.
- `output_bucket`— nama ember Amazon S3 tempat Label Kustom Rekognition Amazon menyimpan hasil pelatihan.
- `output_folder`— nama folder tempat hasil pelatihan disimpan.

Secara opsional, berikan parameter baris perintah berikut untuk melampirkan tag ke model Anda:

- `tag`— nama tag pilihan Anda yang ingin Anda lampirkan ke model.
- `tag_value` nilai tag.

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

import argparse
import logging
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def train_model(rek_client, project_arn, version_name, output_bucket,
output_folder, tag_key, tag_key_value):
    """
    Trains an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to train a
model.
    :param version_name: A version for the model.
    :param output_bucket: The S3 bucket that hosts training output.
    :param output_folder: The path for the training output within output_bucket
    :param tag_key: The name of a tag to attach to the model. Pass None to
exclude
    :param tag_key_value: The value of the tag. Pass None to exclude

    """

    try:
        #Train the model

        status=""
        logger.info("training model version %s for project %s",
            version_name, project_arn)

        output_config = json.loads(
            '{"S3Bucket": "'
            + output_bucket
```

```
        + '"', "S3KeyPrefix": "'
        + output_folder
        + '" } '
    )

    tags={}

    if tag_key is not None and tag_key_value is not None:
        tags = json.loads(
            '{"' + tag_key + '":"' + tag_key_value + '"}'
        )

    response=rek_client.create_project_version(
        ProjectArn=project_arn,
        VersionName=version_name,
        OutputConfig=output_config,
        Tags=tags
    )

    logger.info("Started training: %s", response['ProjectVersionArn'])

    # Wait for the project version training to complete.

    project_version_training_completed_waiter =
rek_client.get_waiter('project_version_training_completed')
    project_version_training_completed_waiter.wait(ProjectArn=project_arn,
    VersionNames=[version_name])

    # Get the completion status.

describe_response=rek_client.describe_project_versions(ProjectArn=project_arn,
    VersionNames=[version_name])
    for model in describe_response['ProjectVersionDescriptions']:
        logger.info("Status: %s", model['Status'])
        logger.info("Message: %s", model['StatusMessage'])
        status=model['Status']

    logger.info("finished training")

    return response['ProjectVersionArn'], status
```

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

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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to train a
model"
    )

    parser.add_argument(
        "version_name", help="A version name of your choosing."
    )

    parser.add_argument(
        "output_bucket", help="The S3 bucket that receives the training
results."
    )

    parser.add_argument(
        "output_folder", help="The folder in the S3 bucket where training
results are stored."
    )

    parser.add_argument(
        "--tag_name", help="The name of a tag to attach to the model",
required=False
    )

    parser.add_argument(
        "--tag_value", help="The value for the tag.", required=False
    )

def main():
```

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

try:

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

    print(f"Training model version {args.version_name} for project
{args.project_arn}")

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

    model_arn, status=train_model(rekognition_client,
        args.project_arn,
        args.version_name,
        args.output_bucket,
        args.output_folder,
        args.tag_name,
        args.tag_value)

    print(f"Finished training model: {model_arn}")
    print(f"Status: {status}")

except ClientError as err:
    logger.exception("Problem training model: %s", err)
    print(f"Problem training model: {err}")
except Exception as err:
    logger.exception("Problem training model: %s", err)
    print(f"Problem training model: {err}")

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

## Java V2

Contoh berikut melatih model. Berikan argumen baris perintah berikut:

- `project_arn`— Nama Sumber Daya Amazon (ARN) dari proyek.
- `version_name`— Nama versi unik untuk model yang Anda pilih.
- `output_bucket`— nama ember Amazon S3 tempat Label Kustom Rekognition Amazon menyimpan hasil pelatihan.
- `output_folder`— nama folder tempat hasil pelatihan disimpan.

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.CreateProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.CreateProjectVersionResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

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

public class TrainModel {
```

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

public static String trainMyModel(RekognitionClient rekClient, String
projectArn, String versionName,
String outputBucket, String outputFolder) {

    try {

        OutputConfig outputConfig =
OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

        logger.log(Level.INFO, "Training Model for project {0}",
projectArn);
        CreateProjectVersionRequest createProjectVersionRequest =
CreateProjectVersionRequest.builder()

.projectArn(projectArn).versionName(versionName).outputConfig(outputConfig).build();

        CreateProjectVersionResponse response =
rekClient.createProjectVersion(createProjectVersionRequest);

        logger.log(Level.INFO, "Model ARN: {0}",
response.projectVersionArn());
        logger.log(Level.INFO, "Training model...");

        // wait until training completes

        DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
            .versionNames(versionName)
            .projectArn(projectArn)
            .build();

        RekognitionWaiter waiter = rekClient.waiter();

        WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionTrainingCompleted(describeProjectVersionsRequest);

        Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();
```

```
        DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

        for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
            .projectVersionDescriptions()) {
            System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
            System.out.println("Status: " +
projectVersionDescription.statusAsString());
            System.out.println("Message: " +
projectVersionDescription.statusMessage());
        }

        return response.projectVersionArn();

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

public static void main(String args[]) {

    String versionName = null;
    String projectArn = null;
    String projectVersionArn = null;
    String bucket = null;
    String location = null;

    final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<output_bucket> <output_folder>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that you want to use.
\n\n"
        + "    version_name - A version name for the model.\n\n"
        + "    output_bucket - The S3 bucket in which to place the
training output. \n\n"
        + "    output_folder - The folder within the bucket that the
training output is stored in. \n\n";

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

```
        System.exit(1);
    }

    projectArn = args[0];
    versionName = args[1];
    bucket = args[2];
    location = args[3];

    try {

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

        // Train model
        projectVersionArn = trainMyModel(rekClient, projectArn, versionName,
bucket, location);

        System.out.println(String.format("Created model: %s for Project ARN:
%s", projectVersionArn, projectArn));

        rekClient.close();

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

}

}
```

3. Jika pelatihan gagal, baca [Mendebug pelatihan model yang gagal](#).

# Mendebug pelatihan model yang gagal

Anda mungkin mengalami kesalahan selama pelatihan model. Amazon Rekognition Custom Labels melaporkan kesalahan pelatihan di konsol dan respons dari [DescribeProjectVersions](#)

Kesalahan adalah terminal (pelatihan tidak dapat dilanjutkan), atau non-terminal (pelatihan dapat dilanjutkan). Untuk kesalahan yang berhubungan dengan isi kumpulan data pelatihan dan pengujian, Anda dapat mengunduh hasil validasi ([ringkasan manifes dan manifes validasi pelatihan dan pengujian](#)). Gunakan kode kesalahan dalam hasil validasi untuk menemukan informasi lebih lanjut di bagian ini. Bagian ini juga menyediakan informasi untuk kesalahan file manifes (kesalahan terminal yang terjadi sebelum isi file manifes divalidasi).

**Note**

Sebuah manifes adalah file yang digunakan untuk menyimpan isi dari dataset.

Anda dapat memperbaiki beberapa kesalahan dengan menggunakan konsol Amazon Rekognition Custom Labels. Kesalahan lain mungkin mengharuskan Anda untuk membuat pembaruan pada pelatihan atau pengujian file manifes. Anda mungkin perlu membuat perubahan lain, seperti izin IAM. Untuk informasi selengkapnya, lihat dokumentasi untuk kesalahan individual.

## Kesalahan terminal

Kesalahan terminal menghentikan pelatihan model. Ada 3 kategori kesalahan pelatihan terminal — kesalahan layanan, kesalahan file manifes, dan kesalahan konten manifes.

Di konsol, Amazon Rekognition Custom Labels menampilkan error terminal untuk model di kolom pesan Status halaman proyek. Dasbor manajemen proyek yang menampilkan daftar proyek dengan nama, versi, tanggal dibuat, kinerja model, dan pesan status yang menunjukkan status model seperti pelatihan selesai atau gagal

Name	Versions	Date created	Model performance	Model status	Status message
[redacted]		2020-10-05	0.608	TRAINING_COMPLETED	The model is ready to run.
[redacted]	19	2020-09-29			
test_4		2020-09-30	0.261	STOPPED	The model has stopped running.
test_20		2020-10-05	N/A	TRAINING_FAILED	Amazon Rekognition experienced a service issue.

Jika Anda menggunakan AWS SDK, Anda dapat mengetahui apakah kesalahan file manifes terminal atau kesalahan konten manifes terminal telah terjadi dengan memeriksa respons dari [DescribeProjectVersions](#). Dalam hal ini, Status nilainya TRAINING\_FAILED dan StatusMessage bidang berisi kesalahan.

## Kesalahan layanan

Kesalahan layanan terminal terjadi ketika Amazon Rekognition mengalami masalah layanan dan tidak dapat melanjutkan pelatihan. Misalnya, kegagalan layanan lain yang bergantung pada Label Kustom Rekognition Amazon. Amazon Rekognition Custom Labels melaporkan kesalahan layanan di konsol karena Amazon Rekognition mengalami masalah layanan. Jika Anda menggunakan AWS SDK, kesalahan layanan yang terjadi selama pelatihan dimunculkan sebagai `InternalServerError` pengecualian oleh [CreateProjectVersion](#) dan [DescribeProjectVersions](#).

Jika terjadi kesalahan layanan, coba lagi pelatihan model. Jika pelatihan terus gagal, hubungi [AWS Support](#) dan sertakan informasi kesalahan yang dilaporkan dengan kesalahan layanan.

## Daftar kesalahan file manifes terminal

Kesalahan file manifes adalah kesalahan terminal, dalam kumpulan data pelatihan dan pengujian, yang terjadi pada tingkat file, atau di beberapa file. Kesalahan file manifes terdeteksi sebelum konten kumpulan data pelatihan dan pengujian divalidasi. Kesalahan file manifes mencegah pelaporan kesalahan [validasi non-terminal](#). Misalnya, file manifes pelatihan kosong menghasilkan file manifes adalah kesalahan kosong. Karena file kosong, tidak ada kesalahan validasi JSON Line non-terminal yang dapat dilaporkan. Ringkasan manifes juga tidak dibuat.

Anda harus memperbaiki kesalahan file manifes sebelum Anda dapat melatih model Anda.

Berikut ini mencantumkan kesalahan file manifes.

- [Ekstensi atau konten file manifes tidak valid.](#)
- [File manifes kosong.](#)
- [Ukuran file manifes melebihi ukuran maksimum yang didukung.](#)
- [Tidak dapat menulis ke ember S3 keluaran.](#)
- [Izin bucket S3 salah.](#)

## Daftar kesalahan konten manifes terminal

Kesalahan konten manifes adalah kesalahan terminal yang berhubungan dengan konten dalam manifes. Misalnya, jika Anda mendapatkan kesalahan [File manifes berisi gambar berlabel tidak cukup per label untuk melakukan pemisahan otomatis](#), pelatihan tidak dapat diselesaikan karena tidak ada cukup gambar berlabel dalam kumpulan data pelatihan untuk membuat kumpulan data pengujian.

Selain dilaporkan di konsol dan dalam respons dari `DescribeProjectVersions`, kesalahan dilaporkan dalam ringkasan manifes bersama dengan kesalahan konten manifes terminal lainnya. Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#).

Kesalahan JSON Line non terminal juga dilaporkan dalam manifes hasil pelatihan dan validasi pengujian terpisah. Kesalahan Jalur JSON non-terminal yang ditemukan oleh Label Kustom Rekognition Amazon belum tentu terkait dengan kesalahan konten manifes yang menghentikan pelatihan. Untuk informasi selengkapnya, lihat [Memahami manifestasi hasil validasi pelatihan dan pengujian](#).

Anda harus memperbaiki kesalahan konten manifes sebelum dapat melatih model Anda.

Berikut ini adalah pesan kesalahan untuk kesalahan konten manifes.


- [File manifes berisi terlalu banyak baris yang tidak valid.](#)
- [File manifes berisi gambar dari beberapa bucket S3.](#)
- [Id pemilik tidak valid untuk bucket S3 gambar.](#)
- [File manifes berisi gambar berlabel tidak cukup per label untuk melakukan pemisahan otomatis.](#)
- [File manifes memiliki terlalu sedikit label.](#)
- [File manifes memiliki terlalu banyak label.](#)
- [Kurang dari {}% label tumpang tindih antara pelatihan dan pengujian file manifes.](#)
- [File manifes memiliki terlalu sedikit label yang dapat digunakan.](#)
- [Kurang dari {}% label yang dapat digunakan tumpang tindih antara file manifes pelatihan dan pengujian.](#)
- [Gagal menyalin gambar dari ember S3.](#)

## Daftar kesalahan validasi baris JSON non-terminal

Kesalahan validasi JSON Line adalah kesalahan non-terminal yang tidak memerlukan Label Kustom Rekognition Amazon untuk menghentikan pelatihan model.

Kesalahan validasi JSON Line tidak ditampilkan di konsol.

Dalam kumpulan data pelatihan dan pengujian, JSON Line mewakili informasi pelatihan atau pengujian untuk satu gambar. Kesalahan validasi di JSON Line, seperti gambar yang tidak valid, dilaporkan dalam manifes validasi pelatihan dan pengujian. Amazon Rekognition Custom Labels menyelesaikan pelatihan menggunakan Garis JSON lain yang valid yang ada di manifes. Untuk informasi selengkapnya, lihat [Memahami manifestasi hasil validasi pelatihan dan pengujian](#). Untuk informasi tentang aturan validasi, lihat [Aturan validasi untuk file manifes](#).

 Note

Pelatihan gagal jika ada terlalu banyak kesalahan JSON Line.

Kami menyarankan Anda juga memperbaiki kesalahan JSON Line non-terminal karena berpotensi menyebabkan kesalahan masa depan atau memengaruhi pelatihan model Anda.

Label Kustom Rekognition Amazon dapat menghasilkan kesalahan validasi Jalur JSON non-terminal berikut.

- [Kunci source-ref tidak ada.](#)
- [Format nilai source-ref tidak valid.](#)
- [Tidak ada atribut label yang ditemukan.](#)
- [Format atribut label {} tidak valid.](#)
- [Format label attributemetadata tidak valid.](#)
- [Tidak ditemukan atribut label yang valid.](#)
- [Satu atau lebih kotak pembatas memiliki nilai kepercayaan yang hilang.](#)
- [Salah satu id kelas lainnya hilang dari peta kelas.](#)
- [JSON Line memiliki format yang tidak valid.](#)
- [Gambar tidak valid. Periksa properti and/or gambar jalur S3.](#)
- [Kotak pembatas memiliki nilai off frame.](#)
- [Tinggi dan lebar kotak pembatas terlalu kecil.](#)
- [Ada lebih banyak kotak pembatas daripada maksimum yang diizinkan.](#)
- [Tidak ditemukan anotasi yang valid.](#)

## Memahami ringkasan manifes

Ringkasan manifes berisi informasi berikut.

- Informasi kesalahan tentang [Daftar kesalahan konten manifes terminal](#) ditemui selama validasi.
- Informasi lokasi kesalahan untuk [Daftar kesalahan validasi baris JSON non-terminal](#) dalam kumpulan data pelatihan dan pengujian.
- Statistik kesalahan seperti jumlah total Garis JSON yang tidak valid yang ditemukan dalam kumpulan data pelatihan dan pengujian.

Ringkasan manifes dibuat selama pelatihan jika tidak ada [Daftar kesalahan file manifes terminal](#). Untuk mendapatkan lokasi file ringkasan manifes (manifest\_summary.json), lihat [Mendapatkan hasil validasi](#)

### Note

[Kesalahan layanan dan kesalahan file manifes](#) tidak dilaporkan dalam ringkasan manifes. Untuk informasi selengkapnya, lihat [Kesalahan terminal](#).

Untuk informasi tentang kesalahan konten manifes tertentu, lihat [Kesalahan konten manifes terminal](#).

## Format file ringkasan manifes

File manifes memiliki 2 bagian, `statistics` dan `errors`.

`statistik`

`statistics` berisi informasi tentang kesalahan dalam kumpulan data pelatihan dan pengujian.

- `training`— statistik dan kesalahan yang ditemukan dalam dataset pelatihan.
- `testing`— statistik dan kesalahan yang ditemukan dalam dataset pengujian.

Objek dalam `errors` array berisi kode kesalahan dan pesan untuk kesalahan konten manifes.

`error_line_indices` Array berisi nomor baris untuk setiap Garis JSON dalam manifes pelatihan atau pengujian yang memiliki kesalahan. Untuk informasi selengkapnya, lihat [Memperbaiki kesalahan pelatihan](#).

## kesalahan

Kesalahan yang mencakup kumpulan data pelatihan dan pengujian. Misalnya, [ERROR\\_INSUFFICIENT\\_USABLE\\_LABEL\\_TUMPANG TINDIH](#) terjadi ketika tidak ada cukup label yang dapat digunakan yang tumpang tindih dengan kumpulan data pelatihan dan pengujian.

```
{
  "statistics": {
    "training":
      {
        "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
        OBJECT_LOCALIZATION and NOT_DETERMINED
        "total_json_lines": Number, # Total number json lines (images) in the
        training manifest.
        "valid_json_lines": Number, # Total number of JSON Lines (images)
        that can be used for training.
        "invalid_json_lines": Number, # Total number of invalid JSON Lines.
        They are not used for training.
        "ignored_json_lines": Number, # JSON Lines that have a valid schema but
        have no annotations. The aren't used for training and aren't counted as invalid.
        "error_json_line_indices": List[int], # Contains a list of line numbers
        for JSON line errors in the training dataset.
        "errors": [
          {
            "code": String, # Error code for a training manifest content
            error.
            "message": String # Description for a training manifest content
            error.
          }
        ]
      },
    "testing":
      {
        "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
        OBJECT_LOCALIZATION and NOT_DETERMINED
        "total_json_lines": Number, # Total number json lines (images) in the
        manifest.
        "valid_json_lines": Number, # Total number of JSON Lines (images) that
        can be used for testing.
        "invalid_json_lines": Number, # Total number of invalid JSON Lines.
        They are not used for testing.
        "ignored_json_lines": Number, # JSON Lines that have a valid schema but
        have no annotations. They aren't used for testing and aren't counted as invalid.
      }
  }
}
```

```

        "error_json_line_indices": List[int], # contains a list of error record
line numbers in testing dataset.
        "errors": [
            {
                "code": String,  # # Error code for a testing manifest content
error.
                "message": String # Description for a testing manifest content
error.
            }
        ]
    },
    "errors": [
        {
            "code": String, # # Error code for errors that span the training and
testing datasets.
            "message": String # Description of the error.
        }
    ]
}

```

## Contoh ringkasan manifes

Contoh berikut adalah ringkasan manifes sebagian yang menunjukkan kesalahan konten manifes terminal ([ERROR\\_TOO\\_MANY\\_INVALID\\_ROWS\\_IN\\_MANIFEST](#)).

`error_json_line_indices` Array berisi nomor baris kesalahan Jalur JSON non-terminal dalam manifes validasi pelatihan atau pengujian yang sesuai.

```

{
  "errors": [],
  "statistics": {
    "training": {
      "use_case": "NOT_DETERMINED",
      "total_json_lines": 301,
      "valid_json_lines": 146,
      "invalid_json_lines": 155,
      "ignored_json_lines": 0,
      "errors": [
        {
          "code": "ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST",
          "message": "The manifest file contains too many invalid rows."
        }
      ]
    },

```

```
    "error_json_line_indices": [  
      15,  
      16,  
      17,  
      22,  
      23,  
      24,  
      .  
      .  
      .  
      .  
      300  
    ]  
  },  
  "testing": {  
    "use_case": "NOT_DETERMINED",  
    "total_json_lines": 15,  
    "valid_json_lines": 13,  
    "invalid_json_lines": 2,  
    "ignored_json_lines": 0,  
    "errors": [],  
    "error_json_line_indices": [  
      13,  
      15  
    ]  
  }  
}
```

## Memahami manifestasi hasil validasi pelatihan dan pengujian

Selama pelatihan, Amazon Rekognition Custom Labels membuat manifes hasil validasi untuk menahan kesalahan JSON Line non-terminal. Manifestasi hasil validasi adalah salinan dari kumpulan data pelatihan dan pengujian dengan informasi kesalahan ditambahkan. Anda dapat mengakses manifes validasi setelah pelatihan selesai. Untuk informasi selengkapnya, lihat [Mendapatkan hasil validasi](#). Label Kustom Rekognition Amazon juga membuat ringkasan manifes yang menyertakan informasi ikhtisar untuk kesalahan JSON Line, seperti lokasi kesalahan dan jumlah kesalahan JSON Line. Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#).

**Note**

Hasil validasi (Training and Testing Validation Result Manifest and Manifest Summary) hanya dibuat jika tidak ada. [Daftar kesalahan file manifes terminal](#)

Manifes berisi Garis JSON untuk setiap gambar dalam kumpulan data. Dalam manifes hasil validasi, informasi kesalahan JSON Line ditambahkan ke JSON Lines di mana kesalahan terjadi.

Kesalahan JSON Line adalah kesalahan non-terminal yang terkait dengan satu gambar.

Kesalahan validasi non-terminal dapat membatalkan seluruh JSON Line atau hanya sebagian.

Misalnya, jika gambar yang direferensikan dalam JSON Line tidak dalam format PNG atau JPG, [ERROR\\_INVALID\\_IMAGE](#) kesalahan terjadi dan seluruh JSON Line dikecualikan dari pelatihan.

Pelatihan berlanjut dengan JSON Lines valid lainnya.

Dalam JSON Line, kesalahan mungkin berarti JSON Line masih dapat digunakan untuk pelatihan.

Misalnya, jika nilai kiri untuk salah satu dari empat kotak pembatas yang terkait dengan label negatif, model masih dilatih menggunakan kotak pembatas valid lainnya. Informasi kesalahan JSON Line dikembalikan untuk kotak pembatas ( ) yang tidak valid. [ERROR\\_INVALID\\_BOUNDING\\_BOX](#) Dalam contoh ini, informasi kesalahan ditambahkan ke annotation objek tempat kesalahan terjadi.

Kesalahan peringatan, seperti [WARNING\\_NO\\_ANNOTATIONS](#), tidak digunakan untuk pelatihan dan dihitung sebagai baris JSON yang diabaikan (`ignored_json_lines`) dalam ringkasan manifes.

Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#). Selain itu, JSON Lines yang diabaikan tidak dihitung terhadap ambang kesalahan 20% untuk pelatihan dan pengujian.

Untuk informasi tentang kesalahan validasi data non-terminal tertentu, lihat. [Kesalahan Validasi Jalur JSON Non-Terminal](#)

**Note**

Jika ada terlalu banyak kesalahan validasi data, pelatihan dihentikan dan kesalahan [ERROR\\_TOO\\_MANY\\_INVALID\\_ROWS\\_IN\\_MANIFEST](#) terminal dilaporkan dalam ringkasan manifes.

Untuk informasi tentang mengoreksi kesalahan JSON Line, lihat. [Memperbaiki kesalahan pelatihan](#)

## Format kesalahan baris JSON

Amazon Rekognition Custom Labels menambahkan informasi kesalahan validasi non-terminal ke tingkat gambar dan format pelokalan objek JSON Lines. Untuk informasi selengkapnya, lihat [the section called “Membuat file manifes”](#).

### Kesalahan Tingkat Gambar

Contoh berikut menunjukkan `Error` array dalam tingkat gambar JSON Line. Ada dua set kesalahan. Kesalahan yang terkait dengan metadata atribut label (dalam contoh ini, `sport-metadata`) dan kesalahan yang terkait dengan gambar. Kesalahan termasuk kode kesalahan (kode), pesan kesalahan (pesan). Untuk informasi selengkapnya, lihat [Mengimpor label tingkat gambar dalam file manifes](#).

```
{
  "source-ref": String,
  "sport": Number,
  "sport-metadata": {
    "class-name": String,
    "confidence": Float,
    "type": String,
    "job-name": String,
    "human-annotated": String,
    "creation-date": String,
    "errors": [
      {
        "code": String, # error codes for label
        "message": String # Description and additional contextual details of
the error
      }
    ],
  },
  "errors": [
    {
      "code": String, # error codes for image
      "message": String # Description and additional contextual details of the
error
    }
  ]
}
```

## Kesalahan pelokalan objek

Contoh berikut menunjukkan array kesalahan dalam pelokalan objek JSON Line. Garis JSON berisi informasi `Errors` array untuk bidang di bagian JSON Line berikut. Setiap `Error` objek menyertakan kode kesalahan dan pesan kesalahan.

- `label atribut` — Kesalahan untuk bidang atribut label. Lihat `bounding-box` dalam contoh.
- `anotasi` — Kesalahan anotasi (kotak pembatas) disimpan dalam `annotations` array di dalam atribut label.
- `label atribut-metadata` — Kesalahan untuk metadata atribut label. Lihat `bounding-box-metadata` dalam contoh.
- `image` — Kesalahan yang tidak terkait dengan atribut label, anotasi, dan bidang metadata atribut label.

Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

```
{
  "source-ref": String,
  "bounding-box": {
    "image_size": [
      {
        "width": Int,
        "height": Int,
        "depth": Int,
      }
    ],
    "annotations": [
      {
        "class_id": Int,
        "left": Int,
        "top": Int,
        "width": Int,
        "height": Int,
        "errors": [ # annotation field errors
          {
            "code": String, # annotation field error code
            "message": String # Description and additional contextual
details of the error
          }
        ]
      }
    ]
  }
}
```

```
    ],
    "errors": [ #label attribute field errors
      {
        "code": String, # error code
        "message": String # Description and additional contextual details of
the error
      }
    ]
  },
  "bounding-box-metadata": {
    "objects": [
      {
        "confidence": Float
      }
    ],
    "class-map": {
      String: String
    },
    "type": String,
    "human-annotated": String,
    "creation-date": String,
    "job-name": String,
    "errors": [ #metadata field errors
      {
        "code": String, # error code
        "message": String # Description and additional contextual details of
the error
      }
    ]
  },
  "errors": [ # image errors
    {
      "code": String, # error code
      "message": String # Description and additional contextual details of the
error
    }
  ]
}
```

## Contoh kesalahan baris JSON

Lokalisasi objek berikut JSON Line (diformat untuk keterbacaan) menunjukkan kesalahan.

[ERROR\\_BOUNDING\\_BOX\\_TOO\\_SMALL](#) Dalam contoh ini, dimensi kotak pembatas (tinggi dan lebar) tidak lebih besar dari 1 x 1.

```
{
  "source-ref": "s3://bucket/Manifests/images/199940-1791.jpg",
  "bounding-box": {
    "image_size": [
      {
        "width": 3000,
        "height": 3000,
        "depth": 3
      }
    ],
    "annotations": [
      {
        "class_id": 1,
        "top": 0,
        "left": 0,
        "width": 1,
        "height": 1,
        "errors": [
          {
            "code": "ERROR_BOUNDING_BOX_TOO_SMALL",
            "message": "The height and width of the bounding box is too
small."
          }
        ]
      }
    ],
    {
      "class_id": 0,
      "top": 65,
      "left": 86,
      "width": 220,
      "height": 334
    }
  ]
},
  "bounding-box-metadata": {
    "objects": [
      {
```

```
        "confidence": 1
      },
      {
        "confidence": 1
      }
    ],
    "class-map": {
      "0": "Echo",
      "1": "Echo Dot"
    },
    "type": "groundtruth/object-detection",
    "human-annotated": "yes",
    "creation-date": "2019-11-20T02:57:28.288286",
    "job-name": "my job"
  }
}
```

## Mendapatkan hasil validasi

Hasil validasi berisi informasi kesalahan untuk [Daftar kesalahan konten manifes terminal](#) dan [Daftar kesalahan validasi baris JSON non-terminal](#). Ada tiga file hasil validasi.

- `training_manifest_with_validation.json` — Salinan file manifes kumpulan data pelatihan dengan informasi kesalahan JSON Line ditambahkan.
- `testing_manifest_with_validation.json` - Salinan file manifes dataset pengujian dengan informasi kesalahan JSON Line ditambahkan.
- `manifest_summary.json` — Ringkasan kesalahan konten manifes dan kesalahan JSON Line yang ditemukan dalam kumpulan data pelatihan dan pengujian. Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#).

Untuk informasi tentang isi manifes pelatihan dan validasi pengujian, lihat [Mendebug pelatihan model yang gagal](#).

### Note

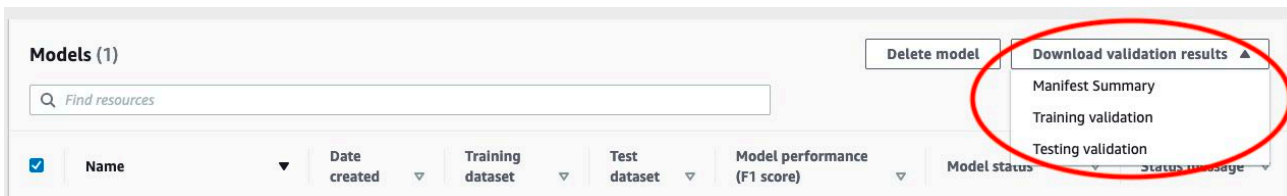
- Hasil validasi dibuat hanya jika tidak [Daftar kesalahan file manifes terminal](#) dihasilkan selama pelatihan.

- Jika [kesalahan layanan](#) terjadi setelah manifes pelatihan dan pengujian divalidasi, hasil validasi akan dibuat, tetapi respons dari [DescribeProjectVersions](#) tidak menyertakan lokasi file hasil validasi.

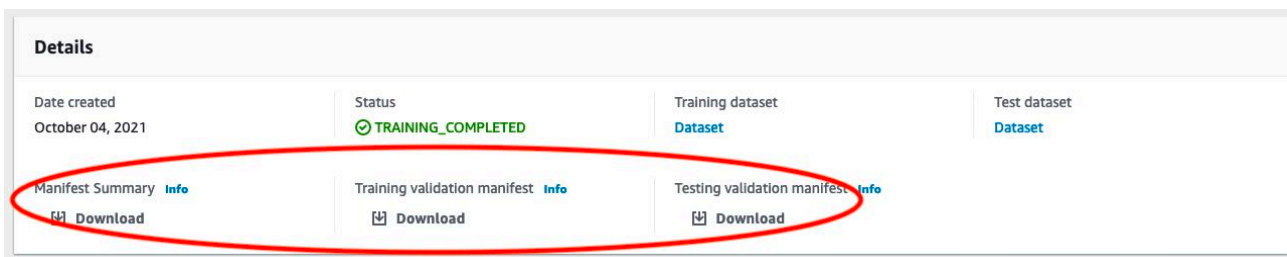
Setelah pelatihan selesai atau gagal, Anda dapat mengunduh hasil validasi menggunakan konsol Amazon Rekognition Custom Labels atau mendapatkan lokasi bucket Amazon S3 dengan memanggil API. [DescribeProjectVersions](#)

## Mendapatkan hasil validasi (Konsol)

Jika Anda menggunakan konsol untuk melatih model Anda, Anda dapat mengunduh hasil validasi dari daftar model proyek, seperti yang ditunjukkan pada diagram berikut. Panel Model menunjukkan pelatihan model dan hasil validasi dengan opsi untuk mengunduh hasil validasi.



Anda juga dapat mengakses unduhan hasil validasi dari halaman detail model. Halaman detail menampilkan detail kumpulan data dengan status, kumpulan data pelatihan dan pengujian, dan tautan unduhan untuk ringkasan manifes, manifes validasi pelatihan, dan manifes validasi pengujian.



Untuk informasi selengkapnya, lihat [Melatih model \(Konsol\)](#).

## Mendapatkan hasil validasi (SDK)

Setelah pelatihan model selesai, Label Kustom Rekognition Amazon menyimpan hasil validasi di bucket Amazon S3 yang ditentukan selama pelatihan. Anda bisa mendapatkan lokasi bucket S3 dengan memanggil [DescribeProjectVersions](#) API, setelah pelatihan selesai. Untuk melatih model, lihat [Melatih model \(SDK\)](#).

Sebuah [ValidationData](#) objek dikembalikan untuk dataset pelatihan ([TrainingDataResult](#)) dan dataset pengujian ([TestingDataResult](#)). Ringkasan manifes dikembalikan [ManifestSummary](#).

Setelah Anda mendapatkan lokasi bucket Amazon S3, Anda dapat mengunduh hasil validasi. Untuk informasi selengkapnya, lihat [Bagaimana cara mengunduh objek dari bucket S3?](#) . Anda juga dapat menggunakan [GetObject](#) operasi ini.

Untuk mendapatkan data validasi (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan contoh berikut untuk mendapatkan lokasi hasil validasi.

Python

Ganti `project_arn` dengan Amazon Resource Name (ARN) dari proyek yang berisi model. Untuk informasi selengkapnya, lihat [Mengelola proyek Label Kustom Rekognition Amazon](#). Ganti `version_name` dengan nama versi model. Untuk informasi selengkapnya, lihat [Melatih model \(SDK\)](#).

```
import boto3
import io
from io import BytesIO
import sys
import json

def describe_model(project_arn, version_name):

    client=boto3.client('rekognition')

    response=client.describe_project_versions(ProjectArn=project_arn,
        VersionNames=[version_name])

    for model in response['ProjectVersionDescriptions']:
        print(json.dumps(model,indent=4,default=str))

def main():

    project_arn='project_arn'
    version_name='version_name'
```

```
describe_model(project_arn, version_name)

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

3. Dalam output program, perhatikan Validation bidang di dalam TestingDataResult dan TrainingDataResult objek. Ringkasan manifes ada diManifestSummary.

## Memperbaiki kesalahan pelatihan

Anda menggunakan ringkasan manifes untuk mengidentifikasi [Daftar kesalahan konten manifes terminal](#) dan [Daftar kesalahan validasi baris JSON non-terminal](#) ditemui selama pelatihan. Anda harus memperbaiki kesalahan konten manifes. Kami menyarankan Anda juga memperbaiki kesalahan JSON Line non-terminal. Untuk informasi tentang kesalahan tertentu, lihat [Kesalahan Validasi Jalur JSON Non-Terminal](#) dan [Kesalahan konten manifes terminal](#).

Anda dapat membuat perbaikan pada kumpulan data pelatihan atau pengujian yang digunakan untuk pelatihan. Atau, Anda dapat membuat perbaikan dalam pelatihan dan pengujian file manifes validasi dan menggunakannya untuk melatih model.

Setelah melakukan perbaikan, Anda perlu mengimpor manifes yang diperbarui dan melatih kembali modelnya. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Prosedur berikut menunjukkan cara menggunakan ringkasan manifes untuk memperbaiki kesalahan konten manifes terminal. Prosedur ini juga menunjukkan kepada Anda cara menemukan dan memperbaiki kesalahan JSON Line dalam manifes validasi pelatihan dan pengujian.

Untuk memperbaiki kesalahan pelatihan Label Kustom Rekognition Amazon

1. Unduh file hasil validasi. Nama file adalah training\_manifest\_with\_validation.json, testing\_manifest\_with\_validation.json dan manifest\_summary.json. Untuk informasi selengkapnya, lihat [Mendapatkan hasil validasi](#).
2. Buka file ringkasan manifes (manifest\_summary.json).
3. Perbaiki kesalahan apa pun dalam ringkasan manifes. Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#).
4. Dalam ringkasan manifes, ulangi melalui error\_line\_indices array training dan perbaiki kesalahan training\_manifest\_with\_validation.json pada nomor JSON Line yang sesuai. Untuk informasi selengkapnya, lihat [the section called "Memahami manifestasi hasil validasi pelatihan dan pengujian"](#).

5. Iterasi melalui `error_line_indices` array `testing` dan perbaiki kesalahan `testing_manifest_with_validation.json` pada nomor JSON Line yang sesuai.
6. Latih kembali model menggunakan file manifes validasi sebagai kumpulan data pelatihan dan pengujian. Untuk informasi selengkapnya, lihat [the section called “Melatih model”](#).

Jika Anda menggunakan AWS SDK dan memilih untuk memperbaiki kesalahan dalam pelatihan atau file manifes data validasi pengujian, gunakan lokasi file manifes data validasi dalam parameter [TrainingData](#) dan [TestingData](#) input untuk [CreateProjectVersion](#). Untuk informasi selengkapnya, lihat [Melatih model \(SDK\)](#).

## Prioritas kesalahan baris JSON

Kesalahan JSON Line berikut terdeteksi terlebih dahulu. Jika salah satu kesalahan ini terjadi, validasi kesalahan JSON Line dihentikan. Anda harus memperbaiki kesalahan ini sebelum Anda dapat memperbaiki kesalahan JSON Line lainnya

- MISSING\_SOURCE\_REF
- ERROR\_INVALID\_SOURCE\_REF\_FORMAT
- ERROR\_NO\_LABEL\_ATTRIBUTES
- ERROR\_INVALID\_LABEL\_ATTRIBUTE\_FORMAT
- ERROR\_INVALID\_LABEL\_ATTRIBUTE\_METADATA\_FORMAT
- ERROR\_MISSING\_BOUNDING\_BOX\_CONFIDENCE
- ERROR\_MISSING\_CLASS\_MAP\_ID
- ERROR\_INVALID\_JSON\_LINE

## Kesalahan file manifes terminal

Topik ini menjelaskan [Daftar kesalahan file manifes terminal](#). Kesalahan file manifes tidak memiliki kode kesalahan terkait. Manifestasi hasil validasi tidak dibuat ketika terjadi kesalahan file manifes terminal. Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#). Kesalahan manifes terminal mencegah pelaporan [Kesalahan Validasi Jalur JSON Non-Terminal](#).

Ekstensi atau konten file manifes tidak valid.

File manifes pelatihan atau pengujian tidak memiliki ekstensi file atau isinya tidak valid.

Untuk memperbaiki kesalahan Ekstensi file manifes atau konten tidak valid.

- Periksa kemungkinan penyebab berikut dalam pelatihan dan pengujian file manifes.
  - File manifes tidak memiliki ekstensi file. Dengan konvensi ekstensi file adalah `.manifest`.
  - Bucket atau kunci Amazon S3 untuk file manifes tidak dapat ditemukan.

File manifes kosong.

File manifes pelatihan atau pengujian yang digunakan untuk pelatihan ada, tetapi kosong. File manifes membutuhkan JSON Line untuk setiap gambar yang Anda gunakan untuk pelatihan dan pengujian.

Untuk memperbaiki kesalahan File manifes kosong.

1. Periksa manifes pelatihan atau pengujian mana yang kosong.
2. Tambahkan JSON Lines ke file manifes kosong. Untuk informasi selengkapnya, lihat [Membuat file manifes](#). Atau, buat kumpulan data baru dengan konsol. Untuk informasi selengkapnya, lihat [the section called “Membuat dataset dengan gambar”](#).

Ukuran file manifes melebihi ukuran maksimum yang didukung.

Ukuran file manifes pelatihan atau pengujian (dalam byte) terlalu besar. Untuk informasi selengkapnya, lihat [Pedoman dan kuota di Label Kustom Rekognition Amazon](#). File manifes dapat memiliki kurang dari jumlah maksimum JSON Lines dan masih melebihi ukuran file maksimum.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan Ukuran file manifes melebihi ukuran maksimum yang didukung.

Untuk memperbaiki kesalahan Ukuran file manifes melebihi ukuran maksimum yang didukung.

1. Periksa manifes pelatihan dan pengujian mana yang melebihi ukuran file maksimum.
2. Kurangi jumlah JSON Lines dalam file manifes yang terlalu besar. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

## Izin bucket S3 salah.

Label Kustom Rekognition Amazon tidak memiliki izin ke satu atau beberapa bucket yang berisi file manifes pelatihan dan pengujian.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

Untuk memperbaiki kesalahan Izin bucket S3 salah.

- Periksa izin untuk bucket yang berisi manifes pelatihan dan pengujian. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan izin konsol Amazon Rekognition Custom Labels](#).

## Tidak dapat menulis ke ember S3 keluaran.

Layanan ini tidak dapat menghasilkan file output pelatihan.

Untuk memperbaiki kesalahan Tidak dapat menulis ke ember S3 keluaran.

- Periksa apakah informasi bucket Amazon S3 dalam parameter [OutputConfig](#)input [CreateProjectVersion](#) sudah benar.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## Kesalahan konten manifes terminal

Topik ini menjelaskan [Daftar kesalahan konten manifes terminal](#) laporan dalam ringkasan manifes. Ringkasan manifes mencakup kode kesalahan dan pesan untuk setiap kesalahan yang terdeteksi. Untuk informasi selengkapnya, lihat [Memahami ringkasan manifes](#). Kesalahan konten manifes terminal tidak menghentikan pelaporan [Daftar kesalahan validasi baris JSON non-terminal](#).

### ERROR\_TOO\_MANY\_INVALID\_ROWS\_IN\_MANIFEST

Pesan kesalahan

File manifes berisi terlalu banyak baris yang tidak valid.

## Informasi selengkapnya

Terjadi `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` kesalahan jika ada terlalu banyak JSON Lines yang berisi konten yang tidak valid.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan. `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST`

Untuk memperbaiki `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST`

1. Periksa manifes untuk kesalahan JSON Line. Untuk informasi selengkapnya, lihat [Memahami manifestasi hasil validasi pelatihan dan pengujian](#).
2. Perbaiki Garis JSON yang memiliki kesalahan Untuk informasi selengkapnya, lihat [Kesalahan Validasi Jalur JSON Non-Terminal](#).

## `ERROR_IMAGES_IN_MULTIPLE_S3_EMBER`

### Pesan kesalahan

File manifes berisi gambar dari beberapa bucket S3.

### Informasi selengkapnya

Manifes hanya dapat mereferensikan gambar yang disimpan dalam satu ember. Setiap JSON Line menyimpan lokasi Amazon S3 dari lokasi gambar dalam nilai `source-ref` Dalam contoh berikut, nama bucket adalah `my-bucket`.

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

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

Untuk memperbaiki `ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS`

- Pastikan semua gambar Anda berada di bucket Amazon S3 yang sama dan nilai `source-ref` di setiap JSON Line mereferensikan bucket tempat gambar Anda disimpan. Atau, pilih bucket Amazon S3 pilihan dan hapus JSON Lines yang `source-ref` tidak mereferensikan bucket pilihan Anda.

## ERROR\_INVALID\_PERMISSIONS\_IMAGES\_S3\_BUCKET

### Pesan kesalahan

Izin untuk bucket S3 gambar tidak valid.

### Informasi selengkapnya

Izin pada bucket Amazon S3 yang berisi gambar tidak benar.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

Untuk memperbaiki **ERROR\_INVALID\_PERMISSIONS\_IMAGES\_S3\_BUCKET**

- Periksa izin bucket yang berisi gambar. Nilai `source-ref` untuk gambar berisi lokasi bucket.

## ERROR\_INVALID\_IMAGES\_S3\_BUCKET\_OWNER

### Pesan kesalahan

Id pemilik tidak valid untuk bucket S3 gambar.

### Informasi selengkapnya

Pemilik ember yang berisi gambar pelatihan atau tes berbeda dari pemilik ember yang berisi manifes pelatihan atau tes. Anda dapat menggunakan perintah berikut untuk menemukan pemilik ember.

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

OWNERID harus cocok untuk ember yang menyimpan gambar dan file manifes.

Untuk memperbaiki **ERROR\_INVALID\_IMAGES\_S3\_BUCKET\_OWNER**

1. Pilih pemilik ember pelatihan, pengujian, output, dan gambar yang diinginkan. Pemilik harus memiliki izin untuk menggunakan Label Kustom Amazon Rekognition.
2. Untuk setiap bucket yang saat ini tidak dimiliki oleh pemilik yang diinginkan, buat bucket Amazon S3 baru yang dimiliki oleh pemilik pilihan.

3. Salin isi ember lama ke ember baru. Untuk informasi selengkapnya, [lihat Bagaimana cara menyalin objek di antara bucket Amazon S3?](#) .

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_INSUFFICIENT\_IMAGES\_PER\_LABEL\_FOR\_AUTOSPLIT

Pesan kesalahan

File manifes berisi gambar berlabel tidak cukup per label untuk melakukan pemisahan otomatis.

Informasi selengkapnya

Selama pelatihan model, Anda dapat membuat kumpulan data pengujian dengan menggunakan 20% gambar dari kumpulan data pelatihan.

ERROR\_INSUFFICIENT\_IMAGES\_PER\_LABEL\_FOR\_AUTOSPLIT terjadi ketika tidak ada cukup gambar untuk membuat kumpulan data pengujian yang dapat diterima.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

Untuk memperbaiki ERROR\_INSUFFICIENT\_IMAGES\_PER\_LABEL\_FOR\_AUTOSPLIT

- Tambahkan lebih banyak gambar berlabel ke kumpulan data pelatihan Anda. Anda dapat menambahkan gambar di konsol Label Kustom Rekognition Amazon dengan menambahkan gambar ke kumpulan data pelatihan, atau dengan menambahkan JSON Lines ke manifes pelatihan Anda. Untuk informasi selengkapnya, lihat [Mengelola kumpulan data](#).

## ERROR\_MANIFEST\_TOO\_FEW\_LABELS

Pesan kesalahan

File manifes memiliki terlalu sedikit label.

Informasi selengkapnya

Kumpulan data pelatihan dan pengujian memiliki jumlah label minimum yang diperlukan. Minimum tergantung pada apakah dataset trains/tests model untuk mendeteksi label tingkat gambar

(klasifikasi) atau jika model mendeteksi lokasi objek. Jika kumpulan data pelatihan dibagi untuk membuat kumpulan data pengujian, jumlah label dalam kumpulan data ditentukan setelah kumpulan data pelatihan dibagi. Untuk informasi selengkapnya, lihat [Pedoman dan kuota di Label Kustom Rekognition Amazon](#).

Untuk memperbaiki ERROR\_MANIFEST\_TOO\_FEW\_LABELS (konsol)

1. Tambahkan lebih banyak label baru ke kumpulan data. Untuk informasi selengkapnya, lihat [Mengelola label](#).
2. Tambahkan label baru ke gambar di kumpulan data. Jika model Anda mendeteksi label tingkat gambar, lihat [Menetapkan label tingkat gambar ke gambar](#). Jika model Anda mendeteksi lokasi objek, lihat [the section called "Pelabelan objek dengan kotak pembatas"](#).

Untuk memperbaiki ERROR\_MANIFEST\_TOO\_FEW\_LABELS (JSON Line)

- Tambahkan JSON Lines untuk gambar baru yang memiliki label baru. Untuk informasi selengkapnya, lihat [Membuat file manifes](#). Jika model Anda mendeteksi label tingkat gambar, Anda menambahkan nama label baru ke bidang `class-name`. Misalnya, label untuk gambar berikut adalah Sunrise.

```
{
  "source-ref": "s3://bucket/images/sunrise.png",
  "testdataset-classification_Sunrise": 1,
  "testdataset-classification_Sunrise-metadata": {
    "confidence": 1,
    "job-name": "labeling-job/testdataset-classification_Sunrise",
    "class-name": "Sunrise",
    "human-annotated": "yes",
    "creation-date": "2018-10-18T22:18:13.527256",
    "type": "groundtruth/image-classification"
  }
}
```

Jika model Anda mendeteksi lokasi objek, tambahkan label baru ke `class-map`, seperti yang ditunjukkan pada contoh berikut.

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
```

```
"image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
}],
"annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
}, {
  "class_id": 0,
  "top": 65,
  "left": 86,
  "width": 220,
  "height": 334
}]
},
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
}
```

Anda perlu memetakan tabel peta kelas ke anotasi kotak pembatas. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

## ERROR\_MANIFEST\_TOO\_MANY\_LABELS

### Pesan kesalahan

File manifes memiliki terlalu banyak label.

### Informasi selengkapnya

Jumlah label unik dalam manifes (dataset) lebih dari batas yang diizinkan. Jika kumpulan data pelatihan dibagi untuk membuat kumpulan data pengujian, jumlah label ditentukan setelah pemisahan.

Untuk memperbaiki ERROR\_MANIFEST\_TOO\_MANY\_LABELS (Konsol)

- Hapus label dari kumpulan data. Untuk informasi selengkapnya, lihat [Mengelola label](#). Label secara otomatis dihapus dari gambar dan kotak pembatas di kumpulan data Anda.

Untuk memperbaiki ERROR\_MANIFEST\_TOO\_MANY\_LABELS (JSON Line)

- Manifestasi dengan garis JSON tingkat gambar - Jika gambar memiliki label tunggal, hapus Garis JSON untuk gambar yang menggunakan label yang diinginkan. Jika JSON Line berisi beberapa label, hapus hanya objek JSON untuk label yang diinginkan. Untuk informasi selengkapnya, lihat [Menambahkan beberapa label tingkat gambar ke gambar](#).

Manifestasi dengan lokasi objek JSON Lines - Hapus kotak pembatas dan informasi label terkait untuk label yang ingin Anda hapus. Lakukan ini untuk setiap JSON Line yang berisi label yang diinginkan. Anda perlu menghapus label dari `class-map` array dan objek yang sesuai dalam `objects` dan `annotations` array. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

## ERROR\_INSUFFICIENT\_LABEL\_TUMPANG TINDIH

### Pesan kesalahan

Kurang dari {}% label tumpang tindih antara pelatihan dan pengujian file manifes.

### Informasi selengkapnya

Ada kurang dari 50% tumpang tindih antara nama label dataset pengujian dan nama label dataset pelatihan.

## Untuk memperbaiki ERROR\_INSUFFICIENT\_LABEL\_OVERLAP (Konsol)

- Hapus label dari kumpulan data pelatihan. Atau, tambahkan label yang lebih umum ke kumpulan data pengujian Anda. Untuk informasi selengkapnya, lihat [Mengelola label](#). Label secara otomatis dihapus dari gambar dan kotak pembatas di kumpulan data Anda.

Untuk memperbaiki ERROR\_INSUFFICIENT\_LABEL\_OVERLAP dengan menghapus label dari kumpulan data pelatihan (JSON Line)

- Manifestasi dengan garis JSON tingkat gambar - Jika gambar memiliki label tunggal, hapus Garis JSON untuk gambar yang menggunakan label yang diinginkan. Jika JSON Line berisi beberapa label, hapus hanya objek JSON untuk label yang diinginkan. Untuk informasi selengkapnya, lihat [Menambahkan beberapa label tingkat gambar ke gambar](#). Lakukan ini untuk setiap JSON Line dalam manifes yang berisi label yang ingin Anda hapus.

Manifestasi dengan lokasi objek JSON Lines - Hapus kotak pembatas dan informasi label terkait untuk label yang ingin Anda hapus. Lakukan ini untuk setiap JSON Line yang berisi label yang diinginkan. Anda perlu menghapus label dari `class-map` array dan objek yang sesuai dalam `objects` dan `annotations` array. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

Untuk memperbaiki ERROR\_INSUFFICIENT\_LABEL\_OVERLAP dengan menambahkan label umum ke kumpulan data pengujian (JSON Line)

- Tambahkan JSON Lines ke kumpulan data pengujian yang menyertakan gambar berlabel label yang sudah ada dalam kumpulan data pelatihan. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

## ERROR\_MANIFEST\_TOO\_FEW\_USABLE\_LABELS

Pesan kesalahan

File manifes memiliki terlalu sedikit label yang dapat digunakan.

Informasi selengkapnya

Manifes pelatihan dapat berisi JSON Lines dalam format label tingkat gambar dan dalam format lokasi objek. Bergantung pada jenis Garis JSON yang ditemukan dalam manifes pelatihan,

Label Kustom Amazon Rekognition memilih untuk membuat model yang mendeteksi label tingkat gambar, atau model yang mendeteksi lokasi objek. Amazon Rekognition Custom Labels menyaring catatan JSON yang valid untuk JSON Lines yang tidak dalam format yang dipilih. `ERROR_MANIFEST_TOO_FEW_USABLE_LABELS` terjadi ketika jumlah label dalam manifes tipe model yang dipilih tidak cukup untuk melatih model.

Minimal 1 label diperlukan untuk melatih model yang mendeteksi label tingkat gambar. Minimal 2 label diperlukan untuk melatih model yang objek lokasi.

Untuk memperbaiki `ERROR_MANIFEST_TOO_FEW_USABLE_LABELS` (Konsol)

1. Periksa `use_case` bidang dalam ringkasan manifes.
2. Tambahkan lebih banyak label ke kumpulan data pelatihan untuk kasus penggunaan (tingkat gambar atau pelokalan objek) yang cocok dengan nilai `use_case` Untuk informasi selengkapnya, lihat [Mengelola label](#). Label secara otomatis dihapus dari gambar dan kotak pembatas di kumpulan data Anda.

Untuk memperbaiki `ERROR_MANIFEST_TOO_FEW_USABLE_LABELS` (JSON Line)

1. Periksa `use_case` bidang dalam ringkasan manifes.
2. Tambahkan lebih banyak label ke kumpulan data pelatihan untuk kasus penggunaan (tingkat gambar atau pelokalan objek) yang cocok dengan nilai `use_case` Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

## `ERROR_INSUFFICIENT_USABLE_LABEL_TUMPAANG TINDIH`

Pesan kesalahan

Kurang dari {}% label yang dapat digunakan tumpang tindih antara file manifes pelatihan dan pengujian.

Informasi selengkapnya

Manifes pelatihan dapat berisi JSON Lines dalam format label tingkat gambar dan dalam format lokasi objek. Bergantung pada format yang ditemukan dalam manifes pelatihan, Label Kustom Amazon Rekognition memilih untuk membuat model yang mendeteksi label tingkat gambar, atau model yang mendeteksi lokasi objek. Label Kustom Rekognition Amazon tidak

menggunakan catatan JSON yang valid untuk JSON Lines yang tidak dalam format model yang dipilih. `ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP` terjadi ketika ada kurang dari 50% tumpang tindih antara label pengujian dan pelatihan yang digunakan.

Untuk memperbaiki `ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP` (Konsol)

- Hapus label dari kumpulan data pelatihan. Atau, tambahkan label yang lebih umum ke kumpulan data pengujian Anda. Untuk informasi selengkapnya, lihat [Mengelola label](#). Label secara otomatis dihapus dari gambar dan kotak pembatas di kumpulan data Anda.

Untuk memperbaiki `ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP` dengan menghapus label dari kumpulan data pelatihan (JSON Line)

- Kumpulan data yang digunakan untuk mendeteksi label tingkat gambar - Jika gambar memiliki label tunggal, hapus Garis JSON untuk gambar yang menggunakan label yang diinginkan. Jika JSON Line berisi beberapa label, hapus hanya objek JSON untuk label yang diinginkan. Untuk informasi selengkapnya, lihat [Menambahkan beberapa label tingkat gambar ke gambar](#). Lakukan ini untuk setiap JSON Line dalam manifes yang berisi label yang ingin Anda hapus.

Kumpulan data yang digunakan untuk mendeteksi lokasi objek - Hapus kotak pembatas dan informasi label terkait untuk label yang ingin Anda hapus. Lakukan ini untuk setiap JSON Line yang berisi label yang diinginkan. Anda perlu menghapus label dari `class-map` array dan objek yang sesuai dalam `objects` dan `annotations` array. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

Untuk memperbaiki `ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP` dengan menambahkan label umum ke kumpulan data pengujian (JSON Line)

- Tambahkan JSON Lines ke kumpulan data pengujian yang menyertakan gambar berlabel label yang sudah ada dalam kumpulan data pelatihan. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

## ERROR\_FAILED\_IMAGES\_S3\_COPY

Pesan kesalahan

Gagal menyalin gambar dari ember S3.

## Informasi selengkapnya

Layanan tidak dapat menyalin gambar apa pun di kumpulan data Anda.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

Untuk memperbaiki `ERROR_FAILED_IMAGES_S3_COPY`

1. Periksa izin gambar Anda.
2. Jika Anda menggunakan AWS KMS, periksa kebijakan bucket. Untuk informasi selengkapnya, lihat [Mendekripsi file yang dienkrpsi dengan AWS Key Management Service](#).

File manifes memiliki terlalu banyak kesalahan terminal.

Ada terlalu banyak baris JSON dengan kesalahan konten terminal.

Untuk memperbaiki `ERROR_TOO_MANY_RECORDS_IN_ERROR`

- Kurangi jumlah Garis JSON (gambar) dengan kesalahan konten terminal. Untuk informasi selengkapnya, lihat [Kesalahan konten manifes terminal](#).

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## Kesalahan Validasi Jalur JSON Non-Terminal

Topik ini mencantumkan kesalahan validasi Jalur JSON non-terminal yang dilaporkan oleh Label Kustom Rekognition Amazon selama pelatihan. Kesalahan dilaporkan dalam manifes validasi pelatihan dan pengujian. Untuk informasi selengkapnya, lihat [Memahami manifestasi hasil validasi pelatihan dan pengujian](#). Anda dapat memperbaiki kesalahan Jalur JSON non-terminal dengan memperbarui JSON Line dalam file manifes pelatihan atau pengujian. Anda juga dapat menghapus JSON Line dari manifes, tetapi hal itu dapat mengurangi kualitas model Anda. Jika ada banyak kesalahan validasi non-terminal, Anda mungkin merasa lebih mudah untuk membuat ulang file manifes. Kesalahan validasi biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#). Untuk informasi tentang memperbaiki kesalahan validasi, lihat [Memperbaiki kesalahan pelatihan](#). Beberapa kesalahan dapat diperbaiki dengan menggunakan konsol Amazon Rekognition Custom Labels.

## ERROR\_MISSING\_SOURCE\_REF

Pesan kesalahan

Kunci source-ref tidak ada.

Informasi selengkapnya

source-refBidang JSON Line menyediakan lokasi Amazon S3 dari sebuah gambar. Kesalahan ini terjadi ketika source-ref kunci hilang atau salah eja. Kesalahan ini biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Untuk memperbaiki **ERROR\_MISSING\_SOURCE\_REF**

1. Periksa apakah source-ref kuncinya ada dan dieja dengan benar. source-refKunci dan nilai lengkap mirip dengan yang berikut. adalah. "source-ref": "s3://bucket/path/image"
2. Perbarui atau source-ref kunci di JSON Line. Atau, hapus, JSON Line dari file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_INVALID\_SOURCE\_REF\_FORMAT

Pesan kesalahan

Format nilai source-ref tidak valid.

Informasi selengkapnya

source-refKuncinya ada di JSON Line, tetapi skema jalur Amazon S3 salah. Misalnya, jalannya https://.... bukanS3://..... Kesalahan ERROR\_INVALID\_SOURCE\_REF\_FORMAT biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Untuk memperbaiki **ERROR\_INVALID\_SOURCE\_REF\_FORMAT**

1. Periksa apakah skema tersebut. "source-ref": "s3://bucket/path/image" Misalnya, "source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg".
2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaikinya.

ERROR\_INVALID\_SOURCE\_REF\_FORMAT

ERROR\_NO\_LABEL\_ATTRIBUTES

Pesan kesalahan

Tidak ada atribut label yang ditemukan.

Informasi selengkapnya

Atribut label atau nama -metadata kunci atribut label (atau keduanya) tidak valid atau hilang. Dalam contoh berikut, ERROR\_NO\_LABEL\_ATTRIBUTES terjadi setiap kali bounding-box-metadata kunci bounding-box atau (atau keduanya) hilang. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [{
      "class_id": 1,
      "top": 251,
      "left": 399,
      "width": 155,
      "height": 101
    }, {
      "class_id": 0,
      "top": 65,
      "left": 86,
      "width": 220,
      "height": 334
    }
  ]
},
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }
]
```

```
  ]],  
  "class-map": {  
    "0": "Echo",  
    "1": "Echo Dot"  
  },  
  "type": "groundtruth/object-detection",  
  "human-annotated": "yes",  
  "creation-date": "2018-10-18T22:18:13.527256",  
  "job-name": "my job"  
}  
}
```

**ERROR\_NO\_LABEL\_ATTRIBUTES** Kesalahan biasanya terjadi dalam file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Untuk memperbaiki **ERROR\_NO\_LABEL\_ATTRIBUTES**

1. Periksa apakah pengidentifikasi atribut label dan -metadata kunci pengidentifikasi atribut label ada dan bahwa nama kunci dieja dengan benar.
2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaikinya.

**ERROR\_NO\_LABEL\_ATTRIBUTES**

## ERROR\_INVALID\_LABEL\_ATTRIBUTE\_FORMAT

Pesan kesalahan

Format atribut label {} tidak valid.

Informasi selengkapnya

Skema untuk kunci atribut label hilang atau tidak valid. Kesalahan

**ERROR\_INVALID\_LABEL\_ATTRIBUTE\_FORMAT** biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#)

Untuk memperbaiki **ERROR\_INVALID\_LABEL\_ATTRIBUTE\_FORMAT**

1. Periksa apakah bagian JSON Line untuk kunci atribut label sudah benar. Dalam contoh lokasi objek berikut, `image_size` dan `annotations` objek harus benar. Kunci atribut label diberi nama `bounding-box`.

```
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
  }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
  }]
},
```

## 2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

### ERROR\_INVALID\_LABEL\_ATTRIBUTE\_METADATA\_FORMAT

Pesan kesalahan

Format metadata atribut label tidak valid.

Informasi selengkapnya

Skema untuk kunci metadata atribut label hilang atau tidak valid. Kesalahan `ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT` biasanya terjadi dalam file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

## Untuk memperbaiki **ERROR\_INVALID\_LABEL\_ATTRIBUTE\_FORMAT**

1. Periksa apakah skema JSON Line untuk kunci metadata atribut label mirip dengan contoh berikut. Kunci metadata atribut label diberi nama. `bounding-box-metadata`

```
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
```

2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_NO\_VALID\_LABEL\_ATTRIBUTES

Pesan kesalahan

Tidak ditemukan atribut label yang valid.

Informasi selengkapnya

Tidak ada atribut label yang valid ditemukan di JSON Line. Amazon Rekognition Custom Labels memeriksa atribut label dan identifier atribut label. Kesalahan **ERROR\_INVALID\_LABEL\_ATTRIBUTE\_FORMAT** biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [Membuat file manifes](#)

Jika Garis JSON tidak dalam format manifes SageMaker AI yang didukung, Label Kustom Rekognition Amazon menandai Garis JSON sebagai tidak valid

ERROR\_NO\_VALID\_LABEL\_ATTRIBUTES dan kesalahan dilaporkan. Saat ini, Amazon Rekognition Custom Labels mendukung pekerjaan klasifikasi dan format kotak pembatas. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Untuk memperbaiki **ERROR\_NO\_VALID\_LABEL\_ATTRIBUTES**

1. Periksa apakah JSON untuk kunci atribut label dan metadata atribut label sudah benar.
2. Perbarui, atau hapus, JSON Line dalam file manifes. Untuk informasi selengkapnya, lihat [the section called “Membuat file manifes”](#).

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_MISSING\_BOUNDING\_BOX\_CONFIDENCE

Pesan kesalahan

Satu atau lebih kotak pembatas memiliki nilai kepercayaan yang hilang.

Informasi selengkapnya

Kunci kepercayaan hilang untuk satu atau lebih kotak pembatas lokasi objek. Kunci kepercayaan untuk kotak pembatas ada di metadata atribut label, seperti yang ditunjukkan pada contoh berikut. Kesalahan ERROR\_MISSING\_BOUNDING\_BOX\_CONFIDENCE biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [the section called “Lokalisasi objek dalam file manifes”](#).

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

Untuk memperbaiki **ERROR\_MISSING\_BOUNDING\_BOX\_CONFIDENCE**

1. Periksa apakah `objects` array dalam atribut label berisi jumlah kunci kepercayaan yang sama karena ada objek dalam `annotations` array atribut label.
2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_MISSING\_CLASS\_MAP\_ID

Pesan kesalahan

Salah satu id kelas lainnya hilang dari peta kelas.

Informasi selengkapnya

Objek `class_id` dalam anotasi (kotak pembatas) tidak memiliki entri yang cocok di peta kelas metadata atribut label (). `class-map` Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#). Kesalahan `ERROR_MISSING_CLASS_MAP_ID` biasanya terjadi pada file manifes yang dibuat secara manual.

Untuk memperbaiki `ERROR_MISSING_CLASS_MAP_ID`

1. Periksa bahwa `class_id` nilai dalam setiap anotasi (bounding box) objek memiliki nilai yang sesuai dalam `class-map` array, seperti yang ditunjukkan pada contoh berikut. `annotationsArray` dan `class_map` array harus memiliki jumlah elemen yang sama.

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [{
      "class_id": 1,
      "top": 251,
      "left": 399,
      "width": 155,
      "height": 101
    }, {
      "class_id": 0,
      "top": 65,
      "left": 86,
      "width": 220,
```

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

2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_INVALID\_JSON\_LINE

Pesan kesalahan

JSON Line memiliki format yang tidak valid.

Informasi selengkapnya

Karakter tak terduga ditemukan di JSON Line. JSON Line diganti dengan JSON Line baru yang hanya berisi informasi kesalahan. Kesalahan ERROR\_INVALID\_JSON\_LINE biasanya terjadi pada file manifes yang dibuat secara manual. Untuk informasi selengkapnya, lihat [the section called "Lokalisasi objek dalam file manifes"](#).

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## Untuk memperbaiki **ERROR\_INVALID\_JSON\_LINE**

1. Buka file manifes dan arahkan ke JSON Line tempat kesalahan **ERROR\_INVALID\_JSON\_LINE** terjadi.
2. Periksa apakah JSON Line tidak berisi karakter yang tidak valid dan karakter yang diperlukan ; atau tidak , hilang.
3. Perbarui, atau hapus, JSON Line dalam file manifes.

## ERROR\_INVALID\_IMAGE

### Pesan kesalahan

Gambar tidak valid. Periksa properti and/or gambar jalur S3.

### Informasi selengkapnya

File yang direferensikan oleh `source-ref` bukan gambar yang valid. Penyebab potensial termasuk rasio aspek gambar, ukuran gambar, dan format gambar.

Untuk informasi selengkapnya, lihat [Pedoman dan kuota](#).

## Untuk memperbaiki **ERROR\_INVALID\_IMAGE**

1. Periksa yang berikut ini.
  - Rasio aspek gambar kurang dari 20:1.
  - Ukuran gambar lebih besar dari 15 MB
  - Gambar dalam format PNG atau JPEG.
  - Jalur menuju gambar `source-ref` sudah benar.
  - Dimensi gambar minimum gambar lebih besar 64 piksel x 64 piksel.
  - Dimensi gambar maksimum gambar kurang dari 4096 piksel x 4096 piksel.
2. Perbarui, atau hapus, JSON Line dalam file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_INVALID\_IMAGE\_DIMENSION

### Pesan kesalahan

Dimensi gambar tidak sesuai dengan dimensi yang diizinkan.

### Informasi selengkapnya

Gambar yang direferensikan oleh `source-ref` tidak sesuai dengan dimensi gambar yang diizinkan. Dimensi minimum adalah 64 piksel. Dimensi maksimum adalah 4096 piksel.

`ERROR_INVALID_IMAGE_DIMENSION` dilaporkan untuk gambar dengan kotak pembatas.

Untuk informasi selengkapnya, lihat [Pedoman dan kuota](#).

### Untuk memperbaiki `ERROR_INVALID_IMAGE_DIMENSION` (Konsol)

1. Perbarui gambar di bucket Amazon S3 dengan dimensi yang dapat diproses oleh Label Kustom Rekognition Amazon.
2. Di konsol Amazon Rekognition Custom Labels, lakukan hal berikut:
  - a. Hapus kotak pembatas yang ada dari gambar.
  - b. Tambahkan kembali kotak pembatas ke gambar.
  - c. Simpan perubahan Anda.

Untuk informasi selengkapnya, lihat [Pelabelan objek dengan kotak pembatas](#).

### Untuk memperbaiki `ERROR_INVALID_IMAGE_DIMENSION` (SDK)

1. Perbarui gambar di bucket Amazon S3 dengan dimensi yang dapat diproses oleh Label Kustom Rekognition Amazon.
2. Dapatkan JSON Line yang ada untuk gambar dengan menelepon [ListDatasetEntries](#). Untuk parameter `SourceRefContains` input, tentukan lokasi Amazon S3 dan nama file gambar.
3. Panggil [UpdateDatasetEntries](#) dan berikan baris JSON untuk gambar. Pastikan nilai `source-ref` cocok dengan lokasi gambar di bucket Amazon S3. Perbarui anotasi kotak pembatas agar sesuai dengan dimensi kotak pembatas yang diperlukan untuk gambar yang diperbarui.

```
{  
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
```

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

## ERROR\_INVALID\_BOUNDING\_BOX

### Pesan kesalahan

Kotak pembatas memiliki nilai off frame.

## Informasi selengkapnya

Informasi kotak pembatas menentukan gambar yang berada di luar bingkai gambar atau berisi nilai negatif.

Untuk informasi selengkapnya, lihat [Pedoman dan kuota](#).

### Untuk memperbaiki **ERROR\_INVALID\_BOUNDING\_BOX**

1. Centang nilai kotak pembatas dalam annotations array.

```
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
  }]
},
```

2. Perbarui, atau hapus, JSON Line dari file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_NO\_VALID\_ANNOTATIONS

### Pesan kesalahan

Tidak ditemukan anotasi yang valid.

### Informasi selengkapnya

Tak satu pun dari objek anotasi di JSON Line berisi informasi kotak pembatas yang valid.

## Untuk memperbaiki **ERROR\_NO\_VALID\_ANNOTATIONS**

1. Perbarui annotations array untuk menyertakan objek kotak pembatas yang valid. Juga, periksa apakah informasi kotak pembatas yang sesuai (confidencedanclass\_map) dalam metadata atribut label sudah benar. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [
      {
        "class_id": 1,      #annotation object
        "top": 251,
        "left": 399,
        "width": 155,
        "height": 101
      }, {
        "class_id": 0,
        "top": 65,
        "left": 86,
        "width": 220,
        "height": 334
      }
    ]
  },
  "bounding-box-metadata": {
    "objects": [
      >{
        "confidence": 1      #confidence object
      },
      {
        "confidence": 1
      }
    ]
  },
  "class-map": {
    "0": "Echo",      #label
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
}
```

```
"human-annotated": "yes",
"creation-date": "2018-10-18T22:18:13.527256",
"job-name": "my job"
}
}
```

2. Perbarui, atau hapus, JSON Line dari file manifes.

Anda tidak dapat menggunakan konsol Label Kustom Rekognition Amazon untuk memperbaiki kesalahan ini.

## ERROR\_BOUNDING\_BOX\_TOO\_SMALL

Pesan kesalahan

Tinggi dan lebar kotak pembatas terlalu kecil.

Informasi selengkapnya

Dimensi kotak pembatas (tinggi dan lebar) harus lebih besar dari 1 x 1 piksel.

Selama pelatihan, Amazon Rekognition Custom Labels mengubah ukuran gambar jika salah satu dimensinya lebih besar dari 1280 piksel (gambar sumber tidak terpengaruh). Tinggi dan lebar kotak pembatas yang dihasilkan harus lebih besar dari 1 x 1 piksel. Lokasi kotak pembatas disimpan dalam `annotations` array lokasi objek JSON Line. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#)

```
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
  }]
},
```

Informasi kesalahan ditambahkan ke objek anotasi.

Untuk memperbaiki `ERROR_BOUNDING_BOX_TOO_SMALL`

- Pilih salah satu opsi berikut.
  - Tingkatkan ukuran kotak pembatas yang terlalu kecil.
  - Hapus kotak pembatas yang terlalu kecil. Untuk informasi tentang menghapus kotak pembatas, lihat [ERROR\\_TOO\\_MANY\\_BOUNDING\\_BOXES](#).
  - Hapus gambar (JSON Line) dari manifes.

## `ERROR_TOO_MANY_BOUNDING_BOXES`

Pesan kesalahan

Ada lebih banyak kotak pembatas daripada maksimum yang diizinkan.

Informasi selengkapnya

Ada lebih banyak kotak pembatas daripada batas yang diizinkan (50). Anda dapat menghapus kotak pembatas berlebih di konsol Label Kustom Rekognition Amazon, atau Anda dapat menghapusnya dari JSON Line.

Untuk memperbaiki `ERROR_TOO_MANY_BOUNDING_BOXES` (Konsol).

1. Tentukan kotak pembatas mana yang akan dihapus.
2. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
3. Pilih Gunakan Label Kustom.
4. Pilih Mulai.
5. Di panel navigasi kiri, pilih proyek yang berisi kumpulan data yang ingin Anda gunakan.
6. Di bagian Datasets, pilih dataset yang ingin Anda gunakan.
7. Di halaman galeri kumpulan data, pilih Mulai pelabelan untuk masuk ke mode pelabelan.
8. Pilih gambar yang ingin Anda hapus dari kotak pembatas.

9. Pilih Draw bounding box.
10. Di alat gambar, pilih kotak pembatas yang ingin Anda hapus.
11. Tekan tombol hapus pada keyboard Anda untuk menghapus kotak pembatas.
12. Ulangi 2 langkah sebelumnya sampai Anda telah menghapus cukup kotak pembatas.
13. Pilih Selesai
14. Pilih Simpan perubahan untuk menyimpan perubahan Anda.
15. Pilih Keluar untuk keluar dari mode pelabelan.

Untuk memperbaiki ERROR\_TOO\_MANY\_BOUNDING\_BOXES (JSON Line).

1. Buka file manifes dan arahkan ke JSON Line di mana kesalahan ERROR\_TOO\_MANY\_BOUNDING\_BOXES terjadi.
2. Hapus yang berikut ini untuk setiap kotak pembatas yang ingin Anda hapus.
  - Hapus `annotation` objek yang diperlukan dari `annotations` array.
  - Hapus `confidence` objek yang sesuai dari `objects` array dalam metadata atribut label.
  - Jika tidak lagi digunakan oleh kotak pembatas lainnya, lepaskan label dari `class-map`

Gunakan contoh berikut untuk mengidentifikasi item mana yang akan dihapus.

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [
      {
        "class_id": 1,      #annotation object
        "top": 251,
        "left": 399,
        "width": 155,
        "height": 101
      }, {
        "class_id": 0,
```

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

## WARNING\_UNANNOTATED\_RECORD

### Pesan Peringatan

Rekaman tidak dianotasi.

### Informasi selengkapnya

Gambar yang ditambahkan ke kumpulan data menggunakan konsol Label Kustom Rekognition Amazon tidak diberi label. Garis JSON untuk gambar tidak digunakan untuk pelatihan.

```
{
  "source-ref": "s3://bucket/images/IMG_1186.png",
  "warnings": [
    {
      "code": "WARNING_UNANNOTATED_RECORD",
      "message": "Record is unannotated."
    }
  ]
}
```

```
    }  
  ]  
}
```

Untuk memperbaiki `WARNING_UNANNOTATED_RECORD`

- Beri label pada gambar dengan menggunakan konsol Amazon Rekognition Custom Labels. Untuk petunjuk, lihat [Menetapkan label tingkat gambar ke gambar](#).

## WARNING\_NO\_ANNOTATIONS

Pesan Peringatan

Tidak ada anotasi yang disediakan.

Informasi selengkapnya

Garis JSON dalam format Pelokalan Objek tidak berisi informasi kotak pembatas, meskipun dianotasi oleh human (). `human-annotated = yes` Jalur JSON valid, tetapi tidak digunakan untuk pelatihan. Untuk informasi selengkapnya, lihat [Memahami manifestasi hasil validasi pelatihan dan pengujian](#).

```
{  
  "source-ref": "s3://bucket/images/IMG_1186.png",  
  "bounding-box": {  
    "image_size": [  
      {  
        "width": 640,  
        "height": 480,  
        "depth": 3  
      }  
    ],  
    "annotations": [  
  
    ],  
    "warnings": [  
      {  
        "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",  
        "message": "No attribute annotations were found."  
      }  
    ]  
  }  
}
```

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

    ],
    "class-map": {

    },
    "type": "groundtruth/object-detection",
    "human-annotated": "yes",
    "creation-date": "2013-11-18 02:53:27",
    "job-name": "my job"
  },
  "warnings": [
    {
      "code": "WARNING_NO_ANNOTATIONS",
      "message": "No annotations were found."
    }
  ]
}
```

## Untuk memperbaiki WARNING\_NO\_ANNOTATIONS

- Pilih salah satu opsi berikut.
  - Tambahkan informasi bounding box (annotations) ke JSON Line. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).
  - Hapus gambar (JSON Line) dari manifes.

## WARNING\_NO\_ATTRIBUTE\_ANNOTATIONS

### Pesan Peringatan

Tidak ada anotasi atribut yang disediakan.

### Informasi selengkapnya

Garis JSON dalam format Pelokalan Objek tidak berisi informasi anotasi kotak pembatas, meskipun dianotasi oleh human (). human-annotated = yes annotationsArray tidak ada atau tidak

dipopulasikan. Jalur JSON valid, tetapi tidak digunakan untuk pelatihan. Untuk informasi selengkapnya, lihat [Memahami manifestasi hasil validasi pelatihan dan pengujian](#).

```
{
  "source-ref": "s3://bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [
      {
        "width": 640,
        "height": 480,
        "depth": 3
      }
    ],
    "annotations": [
      ],
    "warnings": [
      {
        "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
        "message": "No attribute annotations were found."
      }
    ]
  },
  "bounding-box-metadata": {
    "objects": [
      ],
    "class-map": {
      },
    "type": "groundtruth/object-detection",
    "human-annotated": "yes",
    "creation-date": "2013-11-18 02:53:27",
    "job-name": "my job"
  },
  "warnings": [
    {
      "code": "WARNING_NO_ANNOTATIONS",
      "message": "No annotations were found."
    }
  ]
}
```

## Untuk memperbaiki WARNING\_NO\_ATTRIBUTE\_ANNOTATIONS

- Pilih salah satu opsi berikut.
  - Tambahkan satu atau lebih annotation objek kotak pembatas ke JSON Line. Untuk informasi selengkapnya, lihat [Lokalisasi objek dalam file manifes](#).
  - Hapus atribut kotak pembatas.
  - Hapus gambar (JSON Line) dari manifes. Jika atribut kotak pembatas valid lainnya ada di JSON Line, Anda dapat menghapus hanya atribut kotak pembatas yang tidak valid dari JSON Line.

## ERROR\_UNSUPPORTED\_USE\_CASE\_TYPE

### Pesan Peringatan

### Informasi selengkapnya

Nilai type bidang tidak groundtruth/image-classification atau groundtruth/object-detection. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

```
{
  "source-ref": "s3://bucket/test_normal_8.jpg",
  "BB": {
    "annotations": [
      {
        "left": 1768,
        "top": 1007,
        "width": 448,
        "height": 295,
        "class_id": 0
      },
      {
        "left": 1794,
        "top": 1306,
        "width": 432,
        "height": 411,
        "class_id": 1
      },
      {
        "left": 2568,
        "top": 1346,
```

```
        "width": 710,
        "height": 305,
        "class_id": 2
    },
    {
        "left": 2571,
        "top": 1020,
        "width": 644,
        "height": 312,
        "class_id": 3
    }
],
"image_size": [
    {
        "width": 4000,
        "height": 2667,
        "depth": 3
    }
]
},
"BB-metadata": {
    "job-name": "labeling-job/BB",
    "class-map": {
        "0": "comparator",
        "1": "pot_resistor",
        "2": "ir_phototransistor",
        "3": "ir_led"
    },
    "human-annotated": "yes",
    "objects": [
        {
            "confidence": 1
        },
        {
            "confidence": 1
        },
        {
            "confidence": 1
        },
        {
            "confidence": 1
        }
    ],
    "creation-date": "2021-06-22T09:58:34.811Z",
```

```
    "type": "groundtruth/wrongtype",
    "cl-errors": [
      {
        "code": "ERROR_UNSUPPORTED_USE_CASE_TYPE",
        "message": "The use case type of the BB-metadata label attribute
metadata is unsupported. Check the type field."
      }
    ],
    "cl-metadata": {
      "is_labeled": true
    },
    "cl-errors": [
      {
        "code": "ERROR_NO_VALID_LABEL_ATTRIBUTES",
        "message": "No valid label attributes found."
      }
    ]
  }
}
```

Untuk memperbaiki ERROR\_UNSUPPORTED\_USE\_CASE\_TYPE

- Pilih salah satu opsi berikut:
  - Ubah nilai type bidang menjadi groundtruth/image-classification atau groundtruth/object-detection, tergantung pada jenis model yang ingin Anda buat. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).
  - Hapus gambar (JSON Line) dari manifes.

## ERROR\_INVALID\_LABEL\_NAME\_LENGTH

Informasi selengkapnya

Panjang nama label terlalu panjang. Panjang maksimum adalah 256 karakter.

Untuk memperbaiki ERROR\_INVALID\_LABEL\_NAME\_LENGTH

- Pilih salah satu opsi berikut:
  - Kurangi panjang nama label menjadi 256 karakter atau kurang.
  - Hapus gambar (JSON Line) dari manifes.

# Meningkatkan model Label Kustom Amazon Rekognition yang terlatih

Saat pelatihan selesai, Anda mengevaluasi kinerja model. Untuk membantu Anda, Label Kustom Amazon Rekognition menyediakan metrik ringkasan dan metrik evaluasi untuk setiap label. Untuk informasi tentang metrik yang tersedia, lihat [Metrik untuk mengevaluasi model Anda](#). Untuk meningkatkan model Anda menggunakan metrik, lihat [Meningkatkan model Label Kustom Rekognition Amazon](#).

Jika Anda puas dengan keakuratan model Anda, Anda dapat mulai menggunakannya. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

## Topik

- [Metrik untuk mengevaluasi model Anda](#)
- [Mengakses metrik evaluasi \(Konsol\)](#)
- [Mengakses metrik evaluasi Label Kustom Rekognition Amazon \(SDK\)](#)
- [Meningkatkan model Label Kustom Rekognition Amazon](#)

## Metrik untuk mengevaluasi model Anda

Setelah model Anda dilatih, Amazon Rekognition Custom Labels mengembalikan metrik dari pengujian model, yang dapat Anda gunakan untuk mengevaluasi kinerja model Anda. Topik ini menjelaskan metrik yang tersedia untuk Anda, dan bagaimana memahami apakah model terlatih Anda berkinerja baik.

Konsol Label Kustom Rekognition Amazon menyediakan metrik berikut sebagai ringkasan hasil pelatihan dan sebagai metrik untuk setiap label:

- [presisi](#)
- [Ingat](#)
- [F1](#)

Setiap metrik yang kami sediakan adalah metrik yang umum digunakan untuk mengevaluasi kinerja model Machine Learning. Amazon Rekognition Custom Labels mengembalikan metrik untuk hasil

pengujian di seluruh kumpulan data pengujian, bersama dengan metrik untuk setiap label kustom. Anda juga dapat meninjau kinerja model kustom terlatih Anda untuk setiap gambar dalam kumpulan data pengujian Anda. Untuk informasi selengkapnya, lihat [Mengakses metrik evaluasi \(Konsol\)](#).

## Mengevaluasi kinerja model

Selama pengujian, Amazon Rekognition Custom Labels memprediksi jika gambar uji berisi label kustom. Skor kepercayaan adalah nilai yang mengukur kepastian prediksi model.

Jika skor kepercayaan untuk label khusus melebihi nilai ambang batas, output model akan menyertakan label ini. Prediksi dapat dikategorikan dengan cara berikut:

- Benar positif - Model Label Kustom Rekognition Amazon dengan benar memprediksi keberadaan label khusus dalam gambar uji. Artinya, label yang diprediksi juga merupakan label “kebenaran dasar” untuk gambar itu. Misalnya, Amazon Rekognition Custom Labels mengembalikan label bola sepak dengan benar ketika bola sepak hadir dalam gambar.
- Positif palsu - Model Label Kustom Rekognition Amazon salah memprediksi keberadaan label khusus dalam gambar uji. Artinya, label yang diprediksi bukanlah label kebenaran dasar untuk gambar. Misalnya, Amazon Rekognition Custom Labels mengembalikan label bola sepak, tetapi tidak ada label bola sepak di dasar kebenaran untuk gambar itu.
- Negatif palsu - Model Label Kustom Rekognition Amazon tidak memprediksi bahwa label khusus ada dalam gambar, tetapi “kebenaran dasar” untuk gambar tersebut menyertakan label ini. Misalnya, Amazon Rekognition Custom Labels tidak mengembalikan label kustom 'bola sepak' untuk gambar yang berisi bola sepak.
- Benar-benar negatif - Model Amazon Rekognition Custom Labels memprediksi dengan benar bahwa label kustom tidak ada dalam gambar pengujian. Misalnya, Amazon Rekognition Custom Labels tidak mengembalikan label bola sepak untuk gambar yang tidak berisi bola sepak.

Konsol menyediakan akses ke nilai positif benar, positif palsu, dan negatif palsu untuk setiap gambar dalam kumpulan data pengujian Anda. Untuk informasi selengkapnya, lihat [Mengakses metrik evaluasi \(Konsol\)](#).

Hasil prediksi ini digunakan untuk menghitung metrik berikut untuk setiap label, dan agregat untuk seluruh rangkaian pengujian Anda. Definisi yang sama berlaku untuk prediksi yang dibuat oleh model pada tingkat kotak pembatas, dengan perbedaan bahwa semua metrik dihitung pada setiap kotak pembatas (prediksi atau kebenaran dasar) di setiap gambar uji.

## Persimpangan di atas Union (IoU) dan deteksi objek

Intersection over Union (IoU) mengukur persentase tumpang tindih antara dua kotak pembatas objek di atas area gabungannya. Kisarannya adalah 0 (tumpang tindih terendah) hingga 1 (tumpang tindih lengkap). Selama pengujian, kotak pembatas yang diprediksi benar ketika IoU dari kotak pembatas kebenaran tanah dan kotak pembatas yang diprediksi setidaknya 0,5.

## Ambang batas yang diasumsikan

Amazon Rekognition Custom Labels secara otomatis menghitung nilai ambang batas yang diasumsikan (0-1) untuk setiap label kustom Anda. Anda tidak dapat menyetel nilai ambang batas yang diasumsikan untuk label kustom. Ambang batas yang diasumsikan untuk setiap label adalah nilai di atas yang prediksi dihitung sebagai positif benar atau salah. Ini diatur berdasarkan kumpulan data pengujian Anda. Ambang batas yang diasumsikan dihitung berdasarkan skor F1 terbaik yang dicapai pada dataset tes selama pelatihan model.

Anda bisa mendapatkan nilai ambang batas yang diasumsikan untuk label dari hasil pelatihan model. Untuk informasi selengkapnya, lihat [Mengakses metrik evaluasi \(Konsol\)](#).

Perubahan pada nilai ambang batas yang diasumsikan biasanya digunakan untuk meningkatkan presisi dan penarikan kembali model. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Rekognition Amazon](#). Karena Anda tidak dapat menetapkan ambang batas asumsi model untuk label, Anda dapat mencapai hasil yang sama dengan menganalisis gambar dengan `DetectCustomLabels` dan menentukan parameter `MinConfidence` input. Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

## presisi

Amazon Rekognition Custom Labels menyediakan metrik presisi untuk setiap label dan metrik presisi rata-rata untuk seluruh kumpulan data pengujian.

Presisi adalah fraksi prediksi yang benar (positif benar) atas semua prediksi model (positif benar dan salah) pada ambang batas yang diasumsikan untuk label individu. Saat ambang batas meningkat, model mungkin membuat prediksi yang lebih sedikit. Secara umum, bagaimanapun, ia akan memiliki rasio positif sejati yang lebih tinggi daripada positif palsu dibandingkan dengan ambang batas yang lebih rendah. Nilai yang mungkin untuk rentang presisi dari 0-1, dan nilai yang lebih tinggi menunjukkan presisi yang lebih tinggi.

Misalnya, ketika model memprediksi bahwa bola sepak hadir dalam sebuah gambar, seberapa sering prediksi itu benar? Misalkan ada gambar dengan 8 bola sepak dan 5 batu. Jika model memprediksi 9

bola sepak — 8 diprediksi dengan benar dan 1 positif palsu — maka presisi untuk contoh ini adalah 0,89. Namun, jika model memprediksi 13 bola sepak pada gambar dengan 8 prediksi yang benar dan 5 salah, maka presisi yang dihasilkan lebih rendah.

Untuk informasi lebih lanjut, lihat [Presisi dan ingat](#).

## Ingat

Label Kustom Rekognition Amazon menyediakan metrik penarikan rata-rata untuk setiap label dan metrik penarikan rata-rata untuk seluruh kumpulan data pengujian.

Ingat adalah fraksi dari label set pengujian Anda yang diprediksi dengan benar di atas ambang batas yang diasumsikan. Ini adalah ukuran seberapa sering model dapat memprediksi label khusus dengan benar ketika benar-benar ada dalam gambar set pengujian Anda. Kisaran untuk mengingat adalah 0-1. Nilai yang lebih tinggi menunjukkan penarikan yang lebih tinggi.

Misalnya, jika sebuah gambar berisi 8 bola sepak, berapa banyak dari mereka yang terdeteksi dengan benar? Dalam contoh ini di mana gambar memiliki 8 bola sepak dan 5 batu, jika model mendeteksi 5 bola sepak, nilai recall adalah 0,62. Jika setelah pelatihan ulang, model baru mendeteksi 9 bola sepak, termasuk semua 8 yang ada dalam gambar, maka nilai recall adalah 1,0.

Untuk informasi lebih lanjut, lihat [Presisi dan ingat](#).

## F1

Label Kustom Rekognition Amazon menggunakan metrik skor F1 untuk mengukur kinerja model rata-rata setiap label dan kinerja model rata-rata dari seluruh kumpulan data pengujian.

Kinerja model adalah ukuran agregat yang memperhitungkan presisi dan penarikan kembali semua label. (misalnya, skor F1 atau presisi rata-rata). Skor kinerja model adalah nilai antara 0 dan 1. Semakin tinggi nilainya, semakin baik kinerja model untuk penarikan dan presisi. Secara khusus, kinerja model untuk tugas klasifikasi biasanya diukur dengan skor F1. Skor itu adalah rata-rata harmonik dari skor presisi dan ingatan pada ambang batas yang diasumsikan. Misalnya, untuk model dengan presisi 0,9 dan penarikan 1,0, skor F1 adalah 0,947.

Nilai tinggi untuk skor F1 menunjukkan bahwa model berkinerja baik untuk presisi dan penarikan. Jika model tidak berkinerja baik, misalnya, dengan presisi rendah 0,30 dan penarikan tinggi 1,0, skor F1 adalah 0,46. Demikian pula jika presisi tinggi (0,95) dan penarikan rendah (0,20), skor F1 adalah 0,33. Dalam kedua kasus, skor F1 rendah dan menunjukkan masalah dengan model.

Untuk informasi lebih lanjut, lihat [Skor F1](#).

## Menggunakan metrik

Untuk model tertentu yang telah Anda latih dan tergantung pada aplikasi Anda, Anda dapat membuat trade-off antara presisi dan recall dengan menggunakan parameter `MinConfidence` input ke `DetectCustomLabels`. Pada `MinConfidence` nilai yang lebih tinggi, Anda biasanya mendapatkan presisi yang lebih tinggi (prediksi bola sepak yang lebih benar), tetapi ingatan yang lebih rendah (bola sepak yang lebih aktual akan terlewatkan). Pada `MinConfidence` nilai yang lebih rendah, Anda mendapatkan daya ingat yang lebih tinggi (lebih banyak bola sepak aktual diprediksi dengan benar), tetapi presisi yang lebih rendah (lebih banyak prediksi itu akan salah). Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

Metrik juga memberi tahu Anda tentang langkah-langkah yang mungkin Anda ambil untuk meningkatkan kinerja model jika diperlukan. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Rekognition Amazon](#).

### Note

`DetectCustomLabels` mengembalikan prediksi mulai dari 0 hingga 100, yang sesuai dengan kisaran metrik 0-1.

## Mengakses metrik evaluasi (Konsol)

Selama pengujian, model dievaluasi kinerjanya terhadap dataset pengujian. Label dalam kumpulan data pengujian dianggap 'kebenaran dasar' karena mewakili apa yang diwakili oleh gambar sebenarnya. Selama pengujian, model membuat prediksi menggunakan dataset pengujian. Label yang diprediksi dibandingkan dengan label kebenaran dasar dan hasilnya tersedia di halaman evaluasi konsol.

Konsol Label Kustom Amazon Rekognition menampilkan metrik ringkasan untuk seluruh model dan metrik untuk masing-masing label. Metrik yang tersedia di konsol adalah recall presisi, skor F1, kepercayaan diri, dan ambang kepercayaan. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).

Anda dapat menggunakan konsol untuk fokus pada metrik individual. Misalnya, untuk menyelidiki masalah presisi untuk label, Anda dapat memfilter hasil pelatihan berdasarkan label dan hasil positif palsu. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#).

Setelah pelatihan, kumpulan data pelatihan hanya-baca. Jika Anda memutuskan untuk meningkatkan model, Anda dapat menyalin kumpulan data pelatihan ke kumpulan data baru. Anda menggunakan salinan dataset untuk melatih versi baru model.

Pada langkah ini, Anda menggunakan konsol untuk mengakses hasil pelatihan di konsol.

Untuk mengakses metrik evaluasi (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Di halaman Proyek, pilih proyek yang berisi model terlatih yang ingin Anda evaluasi.
6. Dalam Model, pilih model yang ingin Anda evaluasi.
7. Pilih tab Evaluasi untuk melihat hasil evaluasi. Untuk informasi tentang mengevaluasi model, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).
8. Pilih Lihat hasil tes untuk melihat hasil gambar pengujian individual. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#). Tangkapan layar berikut dari ringkasan evaluasi model menunjukkan skor F1, presisi rata-rata, dan penarikan keseluruhan untuk 6 label dengan hasil tes dan metrik kinerja. Detail tentang penggunaan model terlatih juga disediakan.

**rooms\_19** Info Delete model

**Evaluate** | Model details | Use Model | Tags

---

**Evaluation results** View test results

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

---

**Per label performance (10)**

Find labels

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

9. Setelah melihat hasil pengujian, pilih nama proyek untuk kembali ke halaman model. Halaman hasil tes menunjukkan gambar dengan label yang diprediksi dan skor kepercayaan untuk model pembelajaran mesin yang dilatih pada kategori gambar halaman belakang dan halaman depan. Dua contoh gambar ditampilkan.

Custom Labels > Projects > rooms\_19 **rooms\_19.2021-07-13T10.36.30** Performance

**Evaluate image**  
Review the test results of your trained model for individual images. Below each image is information about the model's predicted label compared with the label assigned to the image in the test dataset, noted by result type. You can also filter by label and result types.

---

**Filter by label**

Choose labels  
Choose labels to filter images  
Select a label


True positive  
 False positive  
 False negative

---

**Images (56)** Info


Search images by file name

**backyard2.jpeg**



Labels	Confidence
front_yard <small>False positive</small>	30.3%
backyard <small>False negative</small>	21.6%

**backyard4.jpeg**



Labels	Confidence
backyard <small>True positive</small>	46.3%

- Gunakan metrik untuk mengevaluasi kinerja model. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Rekognition Amazon](#).

## Mengakses metrik evaluasi Label Kustom Rekognition Amazon (SDK)

[DescribeProjectVersions](#) Operasi ini menyediakan akses ke metrik di luar yang disediakan di konsol.

Seperti konsol, `DescribeProjectVersions` menyediakan akses ke metrik berikut sebagai informasi ringkasan untuk hasil pengujian dan sebagai hasil pengujian untuk setiap label:

- [presisi](#)
- [Ingat](#)
- [F1](#)

Ambang batas rata-rata untuk semua label dan ambang batas untuk masing-masing label dikembalikan.

`DescribeProjectVersions` juga menyediakan akses ke metrik berikut untuk klasifikasi dan deteksi gambar (lokasi objek pada gambar).

- Matriks Kebingungan untuk klasifikasi gambar. Untuk informasi selengkapnya, lihat [Melihat matriks kebingungan untuk model](#).
- Mean Average Precision (MAP) untuk deteksi gambar.
- Mean Average Recall (MaR) untuk deteksi gambar.

`DescribeProjectVersions` juga menyediakan akses ke nilai positif benar, positif palsu, negatif palsu, dan negatif sejati. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#).

Metrik skor F1 agregat dikembalikan langsung oleh `DescribeProjectVersions`. Metrik lain dapat diakses dari [Menafsirkan snapshot manifes evaluasi](#) file [Mengakses file ringkasan model](#) dan yang disimpan dalam bucket Amazon S3. Untuk informasi selengkapnya, lihat [Mengakses file ringkasan dan snapshot manifes evaluasi \(SDK\)](#).

Topik

- [Mengakses file ringkasan model](#)

- [Menafsirkan snapshot manifes evaluasi](#)
- [Mengakses file ringkasan dan snapshot manifes evaluasi \(SDK\)](#)
- [Melihat matriks kebingungan untuk model](#)
- [Referensi: File ringkasan hasil pelatihan](#)

## Mengakses file ringkasan model

File ringkasan berisi informasi hasil evaluasi tentang model secara keseluruhan dan metrik untuk setiap label. Metriknya adalah presisi, recall, skor F1. Nilai ambang untuk model juga disediakan. Lokasi file ringkasan dapat diakses dari `EvaluationResult` objek yang dikembalikan oleh `DescribeProjectVersions`. Untuk informasi selengkapnya, lihat [Referensi: File ringkasan hasil pelatihan](#).

Berikut ini adalah contoh file ringkasan.

```
{
  "Version": 1,
  "AggregatedEvaluationResults": {
    "ConfusionMatrix": [
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "CAP",
        "Value": 0.9948717948717949
      },
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "WATCH",
        "Value": 0.008547008547008548
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "CAP",
        "Value": 0.1794871794871795
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "WATCH",
        "Value": 0.7008547008547008
      }
    ]
  }
}
```

```

    "F1Score": 0.9726959470546408,
    "Precision": 0.9719115848331294,
    "Recall": 0.9735042735042735
  },
  "EvaluationDetails": {
    "EvaluationEndTimestamp": "2019-11-21T07:30:23.910943",
    "Labels": [
      "CAP",
      "WATCH"
    ],
    "NumberOfTestingImages": 624,
    "NumberOfTrainingImages": 5216,
    "ProjectVersionArn": "arn:aws:rekognition:us-east-1:nnnnnnnnn:project/my-project/
version/v0/1574317227432"
  },
  "LabelEvaluationResults": [
    {
      "Label": "CAP",
      "Metrics": {
        "F1Score": 0.9794344473007711,
        "Precision": 0.9819587628865979,
        "Recall": 0.9769230769230769,
        "Threshold": 0.9879502058029175
      },
      "NumberOfTestingImages": 390
    },
    {
      "Label": "WATCH",
      "Metrics": {
        "F1Score": 0.9659574468085106,
        "Precision": 0.961864406779661,
        "Recall": 0.9700854700854701,
        "Threshold": 0.014450683258473873
      },
      "NumberOfTestingImages": 234
    }
  ]
}

```

## Menafsirkan snapshot manifes evaluasi

Snapshot manifes evaluasi berisi informasi rinci tentang hasil tes. Cuplikan tersebut mencakup peringkat kepercayaan untuk setiap prediksi. Ini juga mencakup klasifikasi prediksi dibandingkan

dengan klasifikasi gambar yang sebenarnya (positif benar, negatif benar, positif palsu, atau negatif palsu).

File adalah snapshot karena hanya gambar yang dapat digunakan untuk pengujian dan pelatihan yang disertakan. Gambar yang tidak dapat diverifikasi, seperti gambar dalam format yang salah, tidak disertakan dalam manifes. Lokasi snapshot pengujian dapat diakses dari `TestingDataResult` objek yang dikembalikan oleh `DescribeProjectVersions`. Lokasi snapshot pelatihan dapat diakses dari `TrainingDataResult` objek yang dikembalikan oleh `DescribeProjectVersions`.

Snapshot dalam format output manifes SageMaker AI Ground Truth dengan bidang yang ditambahkan untuk memberikan informasi tambahan, seperti hasil klasifikasi biner deteksi. Cuplikan berikut menunjukkan bidang tambahan.

```
"rekognition-custom-labels-evaluation-details": {
  "version": 1,
  "is-true-positive": true,
  "is-true-negative": false,
  "is-false-positive": false,
  "is-false-negative": false,
  "is-present-in-ground-truth": true
  "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- `versi` — Versi format `rekognition-custom-labels-evaluation-details` bidang dalam snapshot manifes.
- `is-true-positive...` — Klasifikasi biner prediksi berdasarkan bagaimana skor kepercayaan dibandingkan dengan ambang minimum untuk label.
- `is-present-in-ground-kebenaran` — Benar jika prediksi yang dibuat oleh model hadir di dasar informasi kebenaran yang digunakan untuk pelatihan, jika tidak salah. Nilai ini tidak didasarkan pada apakah skor kepercayaan melebihi ambang minimum yang dihitung oleh model.
- `ground-truth-labeling-jobs`— Daftar bidang kebenaran dasar di garis manifes yang digunakan untuk pelatihan.

Untuk informasi tentang format manifes SageMaker AI Ground Truth, lihat [Output](#).

Berikut ini adalah contoh pengujian snapshot manifes yang menunjukkan metrik untuk klasifikasi gambar dan deteksi objek.

```
// For image classification
```

```
{
  "source-ref": "s3://amzn-s3-demo-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
    "human-annotated": "no",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  }
}

// For object detection
{
  "source-ref": "s3://amzn-s3-demo-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      }
    ],
  },
}
```

```
    ...
  ],
  "image_size": [
    {
      "width": 1024,
      "depth": 3,
      "height": 768
    }
  ]
},
"rekognition-custom-labels-training-0-metadata": {
  "job-name": "rekognition-custom-labels-training-job",
  "class-map": {
    "0": "Cap",
    ...
  },
  "human-annotated": "yes",
  "objects": [
    {
      "confidence": 1.0
    },
    ...
  ],
  "creation-date": "2019-10-21T22:02:18.432644",
  "type": "groundtruth/object-detection"
},
"rekognition-custom-labels-evaluation": {
  "annotations": [
    {
      "class_id": 0,
      "width": 39,
      "top": 409,
      "height": 63,
      "left": 712
    },
    ...
  ],
  "image_size": [
    {
      "width": 1024,
      "depth": 3,
      "height": 768
    }
  ]
}
```

```
},
"rekognition-custom-labels-evaluation-metadata": {
  "confidence": 0.95,
  "job-name": "rekognition-custom-labels-evaluation-job",
  "class-map": {
    "0": "Cap",
    ...
  },
"human-annotated": "no",
"objects": [
  {
    "confidence": 0.95,
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  },
  ...
],
"creation-date": "2019-10-21T22:02:18.432644",
"type": "groundtruth/object-detection"
}
}
```

## Mengakses file ringkasan dan snapshot manifes evaluasi (SDK)

Untuk mendapatkan hasil pelatihan, Anda menelepon [DescribeProjectVersions](#). Untuk kode sampel, lihat [Menjelaskan model \(SDK\)](#).

Lokasi metrik dikembalikan dalam `ProjectVersionDescription` respons dari `DescribeProjectVersions`.

- `EvaluationResult`— Lokasi file ringkasan.
- `TestingDataResult`— Lokasi snapshot manifes evaluasi yang digunakan untuk pengujian.

Skor F1 dan lokasi file ringkasan dikembalikan. `EvaluationResult` Contoh:

```
"EvaluationResult": {
  "F1Score": 1.0,
  "Summary": {
    "S3Object": {
      "Bucket": "echo-dot-scans",
      "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
    }
  }
}
```

Snapshot manifes evaluasi disimpan di lokasi yang ditentukan dalam parameter `--output-config` input yang Anda tentukan. [Melatih model \(SDK\)](#)

#### Note

Jumlah waktu, dalam hitungan detik, Anda ditagih untuk pelatihan dikembalikan.  
`BillableTrainingTimeInSeconds`

Untuk informasi tentang metrik yang ditampilkan oleh Label Kustom Rekognition Amazon, lihat. [Mengakses metrik evaluasi Label Kustom Rekognition Amazon \(SDK\)](#)

## Melihat matriks kebingungan untuk model

Matriks kebingungan memungkinkan Anda melihat label yang membingungkan model Anda dengan label lain dalam model Anda. Dengan menggunakan matriks kebingungan, Anda dapat memfokuskan perbaikan Anda pada model.

Selama evaluasi model, Amazon Rekognition Custom Labels membuat matriks kebingungan dengan menggunakan gambar uji untuk mengidentifikasi label yang salah diidentifikasi (bingung). Amazon Rekognition Custom Labels hanya membuat matriks kebingungan untuk model klasifikasi. Matriks klasifikasi dapat diakses dari file ringkasan yang dibuat Label Kustom Rekognition Amazon selama pelatihan model. Anda tidak dapat melihat matriks kebingungan di konsol Label Kustom Rekognition Amazon.

### Topik

- [Menggunakan matriks kebingungan](#)
- [Mendapatkan matriks kebingungan untuk sebuah model](#)

## Menggunakan matriks kebingungan

Tabel berikut adalah matriks kebingungan untuk proyek contoh [klasifikasi gambar Kamar](#). Judul kolom adalah label (label kebenaran dasar) yang ditetapkan untuk gambar uji. Judul baris adalah label yang diprediksi model untuk gambar uji. Setiap sel adalah persentase prediksi untuk label (baris) yang seharusnya menjadi label kebenaran dasar (kolom). Misalnya, 67% prediksi untuk kamar mandi diberi label dengan benar sebagai kamar mandi. 33% persen kamar mandi salah diberi label sebagai dapur. Model berkinerja tinggi memiliki nilai sel tinggi ketika label yang diprediksi cocok dengan label kebenaran dasar. Anda dapat melihat ini sebagai garis diagonal dari label kebenaran yang diprediksi pertama hingga terakhir. Jika nilai sel adalah 0, tidak ada prediksi yang dibuat untuk label prediksi sel yang seharusnya menjadi label kebenaran dasar sel.

### Note

Karena model non-deterministik, nilai sel matriks kebingungan yang Anda dapatkan dari pelatihan proyek Rooms mungkin berbeda dari tabel berikut.

Matriks kebingungan mengidentifikasi area untuk fokus. Misalnya, matriks kebingungan menunjukkan bahwa 50% dari waktu model membingungkan lemari untuk kamar tidur. Dalam situasi ini, Anda harus menambahkan lebih banyak gambar lemari dan kamar tidur ke kumpulan data pelatihan Anda. Periksa juga apakah gambar lemari dan kamar tidur yang ada diberi label dengan benar. Ini akan membantu model membedakan kedua label dengan lebih baik. Untuk menambahkan lebih banyak gambar ke kumpulan data, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#).

Meskipun matriks kebingungan sangat membantu, penting untuk mempertimbangkan metrik lainnya. Misalnya, 100% prediksi menemukan label `floor_plan` dengan benar, yang menunjukkan kinerja yang sangat baik. Namun, kumpulan data pengujian hanya memiliki 2 gambar dengan label `floor_plan`. Ini juga memiliki 11 gambar dengan label `living_space`. Ketidakseimbangan ini juga ada dalam kumpulan data pelatihan (13 gambar `living_space` dan 2 gambar lemari). Untuk mendapatkan evaluasi yang lebih akurat, seimbangkan kumpulan data pelatihan dan uji dengan menambahkan lebih banyak gambar label yang kurang terwakili (denah lantai dalam contoh ini). Untuk mendapatkan jumlah gambar uji per label, lihat [Mengakses metrik evaluasi \(Konsol\)](#).

Tabel berikut adalah contoh matriks kebingungan, membandingkan label yang diprediksi (pada sumbu y) dengan label kebenaran dasar:

Label yang diprediksi	halaman belakang	Kamar mandi	kamar tidur	lemari pakaian	jalan masuk_	lantai_pl an	front_yar d	dapur	living_sp ace	teras
halaman belakang	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%
Kamar mandi	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
kamar tidur	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
lemari pakaian	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%
jalan masuk_	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
lantai_pl an	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
front_yar d	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
dapur	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
living_sp ace	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
teras	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

### Mendapatkan matriks kebingungan untuk sebuah model

Kode berikut menggunakan [DescribeProjects](#) dan [DescribeProjectVersions](#) operasi untuk mendapatkan [file ringkasan](#) untuk model. Kemudian menggunakan file ringkasan untuk menampilkan matriks kebingungan untuk model.

## Untuk menampilkan matriks kebingungan untuk model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk menampilkan matriks kebingungan untuk model. Berikan argumen baris perintah berikut:
  - `project_name`— nama proyek yang ingin Anda gunakan. Anda bisa mendapatkan nama proyek dari halaman proyek di konsol Amazon Rekognition Custom Labels.
  - `version_name`— versi model yang ingin Anda gunakan. Anda bisa mendapatkan nama versi dari halaman detail proyek di konsol Amazon Rekognition Custom Labels.

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

"""
Purpose

Shows how to display the confusion matrix for an Amazon Rekognition Custom labels
image
classification model.
"""

import json
import argparse
import logging
import boto3
import pandas as pd
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_model_summary_location(rek_client, project_name, version_name):
    """
    Get the summary file location for a model.

    :param rek_client: A Boto3 Rekognition client.
```

```
:param project_arn: The Amazon Resource Name (ARN) of the project that contains
the model.
:param model_arn: The Amazon Resource Name (ARN) of the model.
:return: The location of the model summary file.
"""

try:
    logger.info(
        "Getting summary file for model %s in project %s.", version_name,
project_name)

    summary_location = ""

    # Get the project ARN from the project name.
    response = rek_client.describe_projects(ProjectNames=[project_name])

    assert len(response['ProjectDescriptions']) > 0, \
        f"Project {project_name} not found."

    project_arn = response['ProjectDescriptions'][0]['ProjectArn']

    # Get the summary file location for the model.
    describe_response =
rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])
    assert len(describe_response['ProjectVersionDescriptions']) > 0, \
        f"Model {version_name} not found."

    model=describe_response['ProjectVersionDescriptions'][0]

    evaluation_results=model['EvaluationResult']

    summary_location=(f"s3://{evaluation_results['Summary']['S3Object']
['Bucket']}"
                    f"/{evaluation_results['Summary']['S3Object']
['Name']}")

    return summary_location

except ClientError as err:
    logger.exception(
        "Couldn't get summary file location: %s", err.response['Error']
['Message'])
```

```
raise

def show_confusion_matrix(summary):
    """
    Shows the confusion matrix for an Amazon Rekognition Custom Labels
    image classification model.
    :param summary: The summary file JSON object.
    """
    pd.options.display.float_format = '{:.0%}'.format

    # Load the model summary JSON into a DataFrame.

    summary_df = pd.DataFrame(
        summary['AggregatedEvaluationResults']['ConfusionMatrix'])

    # Get the confusion matrix.
    confusion_matrix = summary_df.pivot_table(index='PredictedLabel',
                                              columns='GroundTruthLabel',
                                              fill_value=0.0).astype(float)

    # Display the confusion matrix.
    print(confusion_matrix)

def get_summary(s3_resource, summary):
    """
    Gets the summary file.
    : return: The summary file in bytes.
    """
    try:
        summary_bucket, summary_key = summary.replace(
            "s3://", "").split("/", 1)

        bucket = s3_resource.Bucket(summary_bucket)
        obj = bucket.Object(summary_key)
        body = obj.get()['Body'].read()
        logger.info(
            "Got summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't get summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
```

```
        raise
    else:
        return body

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

    parser.add_argument(
        "project_name", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to describe."
    )

def main():
    """
    Entry point for script.
    """

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

    try:

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

        print(
            f"Showing confusion matrix for: {args.version_name} for project
{args.project_name}.")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
        s3_resource = session.resource('s3')

        # Get the summary file for the model.
```

```
summary_location = get_model_summary_location(rekognition_client,
args.project_name,
                                             args.version_name
                                             )
summary = json.loads(get_summary(s3_resource, summary_location))

# Check that the confusion matrix is available.
assert 'ConfusionMatrix' in summary['AggregatedEvaluationResults'], \
    "Confusion matrix not found in summary. Is the model a classification
model?"

# Show the confusion matrix.
show_confusion_matrix(summary)
print("Done")

except ClientError as err:
    logger.exception("Problem showing confusion matrix: %s", err)
    print(f"Problem describing model: {err}")

except AssertionError as err:
    logger.exception(
        "Error: %s.\n", err)
    print(
        f"Error: {err}\n")

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

## Referensi: File ringkasan hasil pelatihan

Ringkasan hasil pelatihan berisi metrik yang dapat Anda gunakan untuk mengevaluasi model Anda. File ringkasan juga digunakan untuk menampilkan metrik di halaman hasil pelatihan konsol. File ringkasan disimpan dalam ember Amazon S3 setelah pelatihan. Untuk mendapatkan file ringkasan, hubungi `DescribeProjectVersion`. Untuk kode sampel, lihat [Mengakses file ringkasan dan snapshot manifes evaluasi \(SDK\)](#).

## File ringkasan

JSON berikut adalah format file ringkasan.

## EvaluationDetails (bagian 3)

Ikhtisar informasi tentang tugas pelatihan. Ini termasuk ARN dari proyek yang dimiliki model (`ProjectVersionArn`), tanggal dan waktu pelatihan selesai, versi model yang dievaluasi (`EvaluationEndTimestamp`), dan daftar label yang terdeteksi selama pelatihan (`Labels`). Juga termasuk jumlah gambar yang digunakan untuk pelatihan (`NumberOfTrainingImages`) dan evaluasi (`NumberOfTestingImages`).

## AggregatedEvaluationResults (bagian 1)

Anda dapat menggunakan `AggregatedEvaluationResults` untuk mengevaluasi kinerja keseluruhan model terlatih saat digunakan dengan kumpulan data pengujian. Metrik agregat disertakan untuk `Precision`, `Recall`, dan `F1Score` metrik. Untuk deteksi objek (lokasi objek pada gambar), metrik `AverageRecall (mAr)` dan `AveragePrecision (mAp)` dikembalikan. Untuk klasifikasi (jenis objek dalam gambar), metrik matriks kebingungan dikembalikan.

## LabelEvaluationResults (bagian 2)

Anda dapat menggunakan `LabelEvaluationResults` untuk mengevaluasi kinerja masing-masing label. Label diurutkan berdasarkan skor F1 dari setiap label. Metrik yang disertakan adalah `Precision`, `Recall`, `F1Score`, dan `Threshold` (digunakan untuk klasifikasi).

Nama file diformat sebagai berikut: `EvaluationSummary-ProjectName-VersionName.json`.

```
{
  "Version": "integer",
  // section-3
  "EvaluationDetails": {
    "ProjectVersionArn": "string",
    "EvaluationEndTimestamp": "string",
    "Labels": "[string]",
    "NumberOfTrainingImages": "int",
    "NumberOfTestingImages": "int"
  },
  // section-1
  "AggregatedEvaluationResults": {
    "Metrics": {
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float",
```

```
// The following 2 fields are only applicable to object detection
"AveragePrecision": "float",
"AverageRecall": "float",
// The following field is only applicable to classification
"ConfusionMatrix":[
  {
    "GroundTruthLabel": "string",
    "PredictedLabel": "string",
    "Value": "float"
  },
  ...
],
}
},
// section-2
"LabelEvaluationResults": [
  {
    "Label": "string",
    "NumberOfTestingImages", "int",
    "Metrics": {
      "Threshold": "float",
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float"
    },
  },
  ...
]
}
```

## Meningkatkan model Label Kustom Rekognition Amazon

Kinerja model pembelajaran mesin sangat tergantung pada faktor-faktor seperti kompleksitas dan variabilitas label khusus Anda (objek dan adegan spesifik yang Anda minati), kualitas dan kekuatan representatif dari kumpulan data pelatihan yang Anda berikan, dan kerangka kerja model dan metode pembelajaran mesin yang digunakan untuk melatih model.

Amazon Rekognition Custom Labels, membuat proses ini lebih sederhana, dan tidak diperlukan keahlian pembelajaran mesin. Namun, proses membangun model yang baik sering melibatkan iterasi atas data dan perbaikan model untuk mencapai kinerja yang diinginkan. Berikut ini adalah informasi tentang cara meningkatkan model Anda.

## Data

Secara umum, Anda dapat meningkatkan kualitas model Anda dengan jumlah data berkualitas lebih baik yang lebih besar. Gunakan gambar pelatihan yang menunjukkan objek atau pemandangan dengan jelas dan tidak berantakan dengan item yang tidak dibutuhkan. Untuk membatasi kotak di sekitar objek, gunakan gambar pelatihan yang menunjukkan objek sepenuhnya terlihat dan tidak tersumbat oleh objek lain.

Pastikan bahwa kumpulan data pelatihan dan pengujian Anda cocok dengan jenis gambar yang pada akhirnya akan Anda jalankan inferensi. Untuk objek, seperti logo, di mana Anda hanya memiliki beberapa contoh pelatihan, Anda harus menyediakan kotak pembatas di sekitar logo dalam gambar pengujian Anda. Gambar-gambar ini mewakili atau menggambarkan skenario di mana Anda ingin melokalisasi objek.

Untuk menambahkan lebih banyak gambar ke kumpulan data pelatihan atau pengujian, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#).

### Mengurangi positif palsu (presisi yang lebih baik)

- Pertama, periksa apakah meningkatkan ambang batas yang diasumsikan memungkinkan Anda menyimpan prediksi yang benar, sambil mengurangi positif palsu. Pada titik tertentu, ini memiliki keuntungan yang berkurang karena trade-off antara presisi dan penarikan untuk model tertentu. Anda tidak dapat mengatur ambang batas yang diasumsikan untuk label, tetapi Anda dapat mencapai hasil yang sama dengan menentukan nilai tinggi untuk `DetectCustomLabels` parameter `MinConfidence` input. Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).
- Anda mungkin melihat satu atau lebih label khusus yang Anda minati (A) secara konsisten bingung dengan kelas objek yang sama (tetapi bukan label yang Anda minati) (B). Untuk membantu, tambahkan B sebagai label kelas objek ke kumpulan data pelatihan Anda (bersama dengan gambar yang Anda dapatkan positif palsu). Secara efektif, Anda membantu model belajar memprediksi B dan bukan A melalui gambar pelatihan baru. Untuk menambahkan gambar ke kumpulan data pelatihan, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#).
- Anda mungkin menemukan bahwa model dikacaukan oleh dua label khusus Anda (A dan B) —gambar uji dengan label A diprediksi memiliki label B dan sebaliknya. Dalam hal ini, periksa dulu gambar yang salah label di set pelatihan dan tes Anda. Gunakan galeri kumpulan data untuk mengelola label yang ditetapkan ke kumpulan data. Untuk informasi selengkapnya, lihat [Mengelola label](#). Selain itu, menambahkan lebih banyak gambar pelatihan yang terkait dengan

jenis kebingungan ini akan membantu model yang dilatih ulang membedakan antara A dan B. Untuk menambahkan gambar ke kumpulan data pelatihan, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#)

## Mengurangi negatif palsu (mengingat lebih baik)

- Gunakan nilai yang lebih rendah untuk ambang batas yang diasumsikan. Anda tidak dapat mengatur ambang batas yang diasumsikan untuk label, tetapi Anda dapat mencapai hasil yang sama dengan menentukan parameter `MinConfidence` input yang lebih rendah. `DetectCustomLabels` Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).
- Gunakan contoh yang lebih baik untuk memodelkan variasi objek dan gambar di mana mereka muncul.
- Pisahkan label Anda menjadi dua kelas yang lebih mudah dipelajari. Misalnya, alih-alih cookie yang baik dan cookie yang buruk, Anda mungkin menginginkan cookie yang baik, cookie yang dibakar, dan cookie yang rusak untuk membantu model mempelajari setiap konsep unik dengan lebih baik.

# Menjalankan model Label Kustom Rekognition Amazon yang terlatih

Ketika Anda puas dengan kinerja model, Anda dapat mulai menggunakannya. Anda dapat memulai dan menghentikan model dengan menggunakan konsol atau AWS SDK. Konsol juga menyertakan contoh operasi SDK yang dapat Anda gunakan.

Topik

- [Unit inferensi](#)
- [Zona Ketersediaan](#)
- [Memulai model Label Kustom Rekognition Amazon](#)
- [Menghentikan model Label Kustom Rekognition Amazon](#)
- [Melaporkan durasi berjalan dan unit inferensi yang digunakan](#)

## Unit inferensi

Saat memulai model, Anda menentukan jumlah sumber daya komputasi, yang dikenal sebagai unit inferensi, yang digunakan model.

### Important

Anda dikenakan biaya untuk jumlah jam yang dijalankan model Anda dan untuk jumlah unit inferensi yang digunakan model Anda saat berjalan, berdasarkan cara Anda mengonfigurasi pengoperasian model Anda. Misalnya, jika Anda memulai model dengan dua unit inferensi dan menggunakan model selama 8 jam, Anda akan dikenakan biaya selama 16 jam inferensi (8 jam waktu berjalan \* dua unit inferensi). Untuk informasi lebih lanjut, lihat [Jam inferensi](#). Jika Anda tidak secara eksplisit [menghentikan model Anda](#), Anda dikenakan biaya bahkan jika Anda tidak secara aktif menganalisis gambar dengan model Anda.

Transaksi per detik (TPS) yang didukung oleh unit inferensi tunggal dipengaruhi oleh hal-hal berikut.

- Model yang mendeteksi label tingkat gambar (klasifikasi) umumnya memiliki TPS yang lebih tinggi daripada model yang mendeteksi dan melokalisasi objek dengan kotak pembatas (deteksi objek).

- Kompleksitas model.
- Gambar beresolusi lebih tinggi membutuhkan lebih banyak waktu untuk analisis.
- Lebih banyak objek dalam gambar membutuhkan lebih banyak waktu untuk analisis.
- Gambar yang lebih kecil dianalisis lebih cepat daripada gambar yang lebih besar.
- Gambar yang diteruskan sebagai byte gambar dianalisis lebih cepat daripada pertama kali mengunggah gambar ke bucket Amazon S3 dan kemudian merujuk gambar yang diunggah. Gambar yang diteruskan sebagai byte gambar harus lebih kecil dari 4,0 MB. Kami menyarankan Anda menggunakan byte gambar untuk pemrosesan gambar secara real time dan ketika ukuran gambar kurang dari 4,0 MB. Misalnya, gambar yang diambil dari kamera IP.
- Memproses gambar yang disimpan dalam bucket Amazon S3 lebih cepat daripada mengunduh gambar, mengonversi ke byte gambar, dan kemudian meneruskan byte gambar untuk dianalisis.
- Menganalisis gambar yang sudah disimpan dalam bucket Amazon S3 mungkin lebih cepat daripada menganalisis gambar yang sama yang diteruskan sebagai byte gambar. Itu terutama benar jika ukuran gambar lebih besar.

Jika jumlah panggilan `DetectCustomLabels` melebihi TPS maksimum yang didukung oleh jumlah unit inferensi yang digunakan model, Amazon Rekognition Custom Labels mengembalikan pengecualian. `ProvisionedThroughputExceededException`

## Mengelola throughput dengan unit inferensi

Anda dapat menambah atau mengurangi throughput model Anda tergantung pada permintaan pada aplikasi Anda. Untuk meningkatkan throughput, gunakan unit inferensi tambahan. Setiap unit inferensi tambahan meningkatkan kecepatan pemrosesan Anda dengan satu unit inferensi. Untuk informasi tentang menghitung jumlah unit inferensi yang Anda butuhkan, lihat [Menghitung unit inferensi untuk Label Kustom Amazon Rekognition dan Amazon Lookout for Vision](#) model. Jika Anda ingin mengubah throughput model yang didukung, Anda memiliki dua opsi:

### Menambahkan atau menghapus unit inferensi secara manual

[Hentikan](#) model dan kemudian [restart](#) dengan jumlah unit inferensi yang diperlukan. Kerugian dengan pendekatan ini adalah model tidak dapat menerima permintaan saat memulai ulang dan tidak dapat digunakan untuk menangani lonjakan permintaan. Gunakan pendekatan ini jika model Anda memiliki throughput yang stabil dan kasus penggunaan Anda dapat mentolerir waktu henti 10-20 menit. Contohnya adalah jika Anda ingin melakukan batch panggilan ke model Anda menggunakan jadwal mingguan.

## Unit inferensi skala otomatis

Jika model Anda harus mengakomodasi lonjakan permintaan, Label Kustom Rekognition Amazon dapat secara otomatis menskalakan jumlah unit inferensi yang digunakan model Anda. Seiring meningkatnya permintaan, Amazon Rekognition Custom Labels menambahkan unit inferensi tambahan ke model dan menghapusnya saat permintaan menurun.

Untuk memungkinkan Amazon Rekognition Custom Labels secara otomatis menskalakan unit inferensi untuk model, mulai model dan atur jumlah maksimum unit inferensi yang dapat digunakan dengan menggunakan parameter. `MaxInferenceUnits` Menetapkan jumlah maksimum unit inferensi memungkinkan Anda mengelola biaya menjalankan model dengan membatasi jumlah unit inferensi yang tersedia untuknya. Jika Anda tidak menentukan jumlah unit maksimum, Label Kustom Rekognition Amazon tidak akan secara otomatis menskalakan model Anda, hanya menggunakan jumlah unit inferensi yang Anda mulai. Untuk informasi mengenai jumlah maksimum unit inferensi, lihat [Service Quotas](#).

Anda juga dapat menentukan jumlah minimum unit inferensi dengan menggunakan `MinInferenceUnits` parameter. Ini memungkinkan Anda menentukan throughput minimum untuk model Anda, di mana satu unit inferensi mewakili 1 jam waktu pemrosesan.

### Note

Anda tidak dapat menyetel jumlah maksimum unit inferensi dengan konsol Amazon Rekognition Custom Labels. Sebagai gantinya, tentukan parameter `MaxInferenceUnits` input ke `StartProjectVersion` operasi.

Label Kustom Rekognition Amazon menyediakan metrik Log CloudWatch Amazon berikut yang dapat Anda gunakan untuk menentukan status penskalaan otomatis saat ini untuk model.

Metrik	Deskripsi
<code>DesiredInferenceUnits</code>	Jumlah unit inferensi yang meningkatkan atau menurunkan Label Kustom Rekognition Amazon.
<code>InServiceInferenceUnits</code>	Jumlah unit inferensi yang digunakan model.

Jika `DesiredInferenceUnits = InServiceInferenceUnits`, Amazon Rekognition Custom Labels saat ini tidak menskalakan jumlah unit inferensi.

Jika `DesiredInferenceUnits > InServiceInferenceUnits`, Amazon Rekognition Custom Labels ditingkatkan hingga nilai `DesiredInferenceUnits`

Jika `DesiredInferenceUnits < InServiceInferenceUnits`, Amazon Rekognition Custom Labels diperkecil ke nilai `DesiredInferenceUnits`

[Untuk informasi selengkapnya mengenai metrik yang ditampilkan oleh Label Kustom Amazon Rekognition dan dimensi pemfilteran, CloudWatch lihat metrik untuk Rekognition.](#)

Untuk mengetahui jumlah maksimum unit inferensi yang Anda minta untuk model, panggil `DescribeProjectsVersion` dan periksa `MaxInferenceUnits` bidang dalam respons. Untuk kode sampel, lihat [Menjelaskan model \(SDK\)](#).

## Zona Ketersediaan

Amazon Rekognition Custom Labels mendistribusikan unit inferensi di beberapa Availability Zone dalam AWS suatu Wilayah untuk meningkatkan ketersediaan. Untuk informasi selengkapnya, lihat [Availability Zone](#). Untuk membantu melindungi model produksi Anda dari pemadaman Availability Zone dan kegagalan unit inferensi, mulailah model produksi Anda dengan setidaknya dua unit inferensi.

Jika terjadi pemadaman Availability Zone, semua unit inferensi di Availability Zone tidak tersedia dan kapasitas model berkurang. Panggilan ke [DetectCustomLabels](#) didistribusikan kembali di seluruh unit inferensi yang tersisa. Panggilan tersebut berhasil jika tidak melebihi Transaksi Per Detik (TPS) yang didukung dari unit inferensi yang tersisa. Setelah AWS memperbaiki Availability Zone, unit inferensi dimulai ulang, dan kapasitas penuh dipulihkan.

Jika unit inferensi tunggal gagal, Label Kustom Rekognition Amazon secara otomatis memulai unit inferensi baru di Availability Zone yang sama. Kapasitas model dikurangi sampai unit inferensi baru dimulai.

## Memulai model Label Kustom Rekognition Amazon

Anda dapat mulai menjalankan model Amazon Rekognition Custom Labels dengan menggunakan konsol atau dengan menggunakan operasi. [StartProjectVersion](#)

**⚠ Important**

Anda dikenakan biaya untuk jumlah jam yang dijalankan model Anda dan untuk jumlah unit inferensi yang digunakan model Anda saat sedang berjalan. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

Memulai model mungkin membutuhkan waktu beberapa menit untuk menyelesaikannya. Untuk memeriksa status kesiapan model saat ini, periksa halaman detail untuk proyek atau penggunaan [DescribeProjectVersions](#).

Setelah model dimulai Anda menggunakan [DetectCustomLabels](#), untuk menganalisis gambar menggunakan model. Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#). Konsol juga menyediakan contoh kode untuk dipanggil `DetectCustomLabels`.

**Topik**

- [Memulai model Label Kustom Rekognition Amazon \(Konsol\)](#)
- [Memulai model Label Kustom Rekognition Amazon \(SDK\)](#)

## Memulai model Label Kustom Rekognition Amazon (Konsol)

Gunakan prosedur berikut untuk mulai menjalankan model Amazon Rekognition Custom Labels dengan konsol. Anda dapat memulai model langsung dari konsol atau menggunakan kode AWS SDK yang disediakan oleh konsol.

Untuk memulai model (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Pada halaman Sumber daya proyek, pilih proyek yang berisi model terlatih yang ingin Anda mulai.
6. Di bagian Model, pilih model yang ingin Anda mulai.
7. Pilih tab Gunakan model.
8. Lakukan salah satu tindakan berikut:

## Start model using the console

Di bagian Mulai atau hentikan model lakukan hal berikut:

1. Pilih jumlah unit inferensi yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).
2. Pilih Mulai.
3. Dalam kotak dialog Mulai model, pilih Mulai.

## Start model using the AWS SDK

Di bagian Gunakan model Anda lakukan hal berikut:

1. Pilih Kode API.
2. Pilih AWS CLI atau Python.
3. Dalam model Mulai salin kode contoh.
4. Gunakan kode contoh untuk memulai model Anda. Untuk informasi selengkapnya, lihat [Memulai model Label Kustom Rekognition Amazon \(SDK\)](#).
9. Untuk kembali ke halaman ikhtisar proyek, pilih nama proyek Anda di bagian atas halaman.
10. Di bagian Model, periksa status model. Saat status model sedang BERJALAN, Anda dapat menggunakan model untuk menganalisis gambar. Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

## Memulai model Label Kustom Rekognition Amazon (SDK)

Anda memulai model dengan memanggil [StartProjectVersion](#) API dan meneruskan Amazon Resource Name (ARN) model dalam parameter `ProjectVersionArn` input. Anda juga menentukan jumlah unit inferensi yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

Sebuah model mungkin membutuhkan waktu beberapa saat untuk memulai. Contoh Python dan Java dalam topik ini menggunakan pelayan untuk menunggu model dimulai. Pelayan adalah metode utilitas yang melakukan polling untuk keadaan tertentu terjadi. Atau, Anda dapat memeriksa status saat ini dengan menelepon [DescribeProjectVersions](#).

## Untuk memulai model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode contoh berikut untuk memulai model.

### CLI

Ubah nilai `project-version-arn` ke ARN model yang ingin Anda mulai. Ubah nilai `--min-inference-units` ke jumlah unit inferensi yang ingin Anda gunakan. Secara opsional, ubah `--max-inference-units` ke jumlah maksimum unit inferensi yang dapat digunakan Label Kustom Rekognition Amazon untuk menskalakan model secara otomatis.

```
aws rekognition start-project-version --project-version-arn model_arn \  
  --min-inference-units minimum number of units \  
  --max-inference-units maximum number of units \  
  --profile custom-labels-access
```

### Python

Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang berisi model yang ingin Anda mulai.
- `model_arn`— ARN dari model yang ingin Anda mulai.
- `min_inference_units`— jumlah unit inferensi yang ingin Anda gunakan.
- (Opsional) `--max_inference_units` Jumlah maksimum unit inferensi yang dapat digunakan Label Kustom Rekognition Amazon untuk menskalakan model secara otomatis.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to start running an Amazon Lookout for Vision model.  
"""  
  
import argparse  
import logging
```

```
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The name of the project that you want to use.
    :param model_arn: The name of the model that you want the status for.
    :return: The model status
    """

    logger.info("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]

    models = rek_client.describe_project_versions(ProjectArn=project_arn,
                                                  VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:

        logger.info("Status: %s", model['StatusMessage'])
        return model["Status"]

    error_message = f"Model {model_arn} not found."
    logger.exception(error_message)
    raise Exception(error_message)

def start_model(rek_client, project_arn, model_arn, min_inference_units,
               max_inference_units=None):
    """
    Starts the hosting of an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The name of the project that contains the
    model that you want to start hosting.
    :param min_inference_units: The number of inference units to use for
    hosting.
    :param max_inference_units: The number of inference units to use for auto-
    scaling
    """
```

```
the model. If not supplied, auto-scaling does not happen.
"""

try:
    # Start the model
    logger.info(f"Starting model: {model_arn}. Please wait....")

    if max_inference_units is None:
        rek_client.start_project_version(ProjectVersionArn=model_arn,
MinInferenceUnits=int(min_inference_units))
    else:
        rek_client.start_project_version(ProjectVersionArn=model_arn,
                                        MinInferenceUnits=int(
                                            min_inference_units),
MaxInferenceUnits=int(max_inference_units))

    # Wait for the model to be in the running state
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
    project_version_running_waiter = rek_client.get_waiter(
        'project_version_running')
    project_version_running_waiter.wait(
        ProjectArn=project_arn, VersionNames=[version_name])

    # Get the running status
    return get_model_status(rek_client, project_arn, model_arn)

except ClientError as err:
    logger.exception("Client error: Problem starting model: %s", err)
    raise

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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains that the model
you want to start."
    )
    parser.add_argument(
```

```
        "model_arn", help="The ARN of the model that you want to start."
    )
    parser.add_argument(
        "min_inference_units", help="The minimum number of inference units to
use."
    )
    parser.add_argument(
        "--max_inference_units", help="The maximum number of inference units to
use for auto-scaling the model.", required=False
    )

def main():

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

    try:

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

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

        status = start_model(rekognition_client,
                             args.project_arn, args.model_arn,
                             args.min_inference_units,
                             args.max_inference_units)

        print(f"Finished starting model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        error_message = f"Client error: Problem starting model: {err}"
        logger.exception(error_message)
        print(error_message)

    except Exception as err:
        error_message = f"Problem starting model:{err}"
        logger.exception(error_message)
```

```
print(error_message)

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

## Java V2

Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang berisi model yang ingin Anda mulai.
- `model_arn`— ARN dari model yang ingin Anda mulai.
- `min_inference_units`— jumlah unit inferensi yang ingin Anda gunakan.
- (Opsional) `max_inference_units` — jumlah maksimum unit inferensi yang dapat digunakan Label Kustom Rekognition Amazon untuk menskalakan model secara otomatis. Jika Anda tidak menentukan nilai, penskalaan otomatis tidak akan terjadi.

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.StartProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartProjectVersionResponse;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;
```

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

public class StartModel {

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

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;
    }

    public static void startMyModel(RekognitionClient rekClient, String
projectArn, String modelArn,
        Integer minInferenceUnits, Integer maxInferenceUnits
        ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Starting model: {0}", modelArn);

            StartProjectVersionRequest startProjectVersionRequest = null;

            if (maxInferenceUnits == null) {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                    .projectVersionArn(modelArn)
                    .minInferenceUnits(minInferenceUnits)
                    .build();
            }
            else {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                    .projectVersionArn(modelArn)
```

```
        .minInferenceUnits(minInferenceUnits)
        .maxInferenceUnits(maxInferenceUnits)
        .build();

    }

    StartProjectVersionResponse response =
rekClient.startProjectVersion(startProjectVersionRequest);

    logger.log(Level.INFO, "Status: {0}", response.statusAsString() );

    // Get the model version

    int start = findForwardSlash(modelArn, 3) + 1;
    int end = findForwardSlash(modelArn, 4);

    String versionName = modelArn.substring(start, end);

    // wait until model starts

    DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
        .versionNames(versionName)
        .projectArn(projectArn)
        .build();

    RekognitionWaiter waiter = rekClient.waiter();

    WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

    .waitUntilProjectVersionRunning(describeProjectVersionsRequest);

    Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

    DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

    for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
        .projectVersionDescriptions()) {
```

```
        if(projectVersionDescription.status() ==
ProjectVersionStatus.RUNNING) {
            logger.log(Level.INFO, "Model is running" );

        }
        else {
            String error = "Model training failed: " +
projectVersionDescription.statusAsString() + " "
                + projectVersionDescription.statusMessage() + " " +
modelArn;

            logger.log(Level.SEVERE, error);
            throw new Exception(error);
        }
    }

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

}

public static void main(String[] args) {

    String modelArn = null;
    String projectArn = null;
    Integer minInferenceUnits = null;
    Integer maxInferenceUnits = null;

    final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<min_inference_units> <max_inference_units>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that contains the
model that you want to start. \n\n"
        + "    model_arn - The ARN of the model version that you want to
start.\n\n"
        + "    min_inference_units - The number of inference units to
start the model with.\n\n"
```

```
        + "    max_inference_units - The maximum number of inference
units that Custom Labels can use to "
        + "    automatically scale the model. If the value is null,
automatic scaling doesn't happen.\n\n";

    if (args.length < 3 || args.length >4) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];
    modelArn = args[1];
    minInferenceUnits=Integer.parseInt(args[2]);

    if (args.length == 4) {
        maxInferenceUnits = Integer.parseInt(args[3]);
    }

    try {

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

        // Start the model.
        startMyModel(rekClient, projectArn, modelArn, minInferenceUnits,
maxInferenceUnits);

        System.out.println(String.format("Model started: %s", modelArn));

        rekClient.close();

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

```
    }  
  
  }  
  
}
```

## Menghentikan model Label Kustom Rekognition Amazon

Anda dapat berhenti menjalankan model Amazon Rekognition Custom Labels dengan menggunakan konsol atau dengan menggunakan operasi. [StopProjectVersion](#)

Topik

- [Menghentikan model Label Kustom Rekognition Amazon \(Konsol\)](#)
- [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#)

## Menghentikan model Label Kustom Rekognition Amazon (Konsol)

Gunakan prosedur berikut untuk menghentikan model Amazon Rekognition Custom Labels yang sedang berjalan dengan konsol. Anda dapat menghentikan model langsung dari konsol atau menggunakan kode AWS SDK yang disediakan oleh konsol.

Untuk menghentikan model (konsol)

1. Buka konsol Amazon Rekognition di. <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Pada halaman Sumber daya proyek, pilih proyek yang berisi model terlatih yang ingin Anda hentikan.
6. Di bagian Model, pilih model yang ingin Anda hentikan.
7. Pilih tab Gunakan model.
8. Stop model using the console
  1. Di bagian Mulai atau hentikan model, pilih Berhenti.

2. Dalam kotak dialog Stop model, masukkan stop untuk mengonfirmasi bahwa Anda ingin menghentikan model.
3. Pilih Berhenti untuk menghentikan model Anda.

### Stop model using the AWS SDK

Di bagian Gunakan model Anda lakukan hal berikut:

1. Pilih Kode API.
2. Pilih AWS CLI atau Python.
3. Dalam model Stop salin kode contoh.
4. Gunakan kode contoh untuk menghentikan model Anda. Untuk informasi selengkapnya, lihat [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#).
9. Pilih nama proyek Anda di bagian atas halaman untuk kembali ke halaman ikhtisar proyek.
10. Di bagian Model, periksa status model. Model telah berhenti ketika status model BERHENTI.

## Menghentikan model Label Kustom Rekognition Amazon (SDK)

Anda menghentikan model dengan memanggil [StopProjectVersion](#) API dan meneruskan Amazon Resource Name (ARN) model dalam parameter `ProjectVersionArn` input.

Seorang model mungkin membutuhkan waktu beberapa saat untuk berhenti. Untuk memeriksa status saat ini, gunakan `DescribeProjectVersions`.

Untuk menghentikan model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode contoh berikut untuk menghentikan model yang sedang berjalan.

CLI

Ubah nilai `project-version-arn` ke ARN dari versi model yang ingin Anda hentikan.

```
aws rekognition stop-project-version --project-version-arn "model arn" \  
--profile custom-labels-access
```

## Python

Contoh berikut menghentikan model yang sudah berjalan.

Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang berisi model yang ingin Anda hentikan.
- `model_arn`— ARN dari model yang ingin Anda hentikan.

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

"""
Purpose
Shows how to stop a running Amazon Lookout for Vision model.
"""

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The name of the project that you want to use.
    :param model_arn: The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]
```

```
# Get the model status.
models=rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])

for model in models['ProjectVersionDescriptions']:
    logger.info("Status: %s",model['StatusMessage'])
    return model["Status"]

# No model found.
logger.exception("Model %s not found.", model_arn)
raise Exception("Model %s not found.", model_arn)

def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn: The ARN of the model (ProjectVersion) that you want to
    stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])

        # stops when hosting has stopped or failure.
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
```

```
        finished = True
        continue

        error_message = f"Error stopping model. Unexpected state: {status}"
        logger.exception(error_message)
        raise Exception(error_message)

    logger.info("finished. Status %s", status)
    return status

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

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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )

def main():

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

    try:

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

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

```
        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

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

## Java V2

Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang berisi model yang ingin Anda hentikan.
- `model_arn`— ARN dari model yang ingin Anda hentikan.

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.StopProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StopProjectVersionResponse;

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

public class StopModel {

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

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;
    }

    public static void stopMyModel(RekognitionClient rekClient, String
projectArn, String modelArn)
        throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Stopping {0}", modelArn);

            StopProjectVersionRequest stopProjectVersionRequest =
                StopProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            StopProjectVersionResponse response =
                rekClient.stopProjectVersion(stopProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString());

            // Get the model version
```

```
int start = findForwardSlash(modelArn, 3) + 1;
int end = findForwardSlash(modelArn, 4);

String versionName = modelArn.substring(start, end);

// wait until model stops

DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
    .projectArn(projectArn).versionNames(versionName).build();

boolean stopped = false;

// Wait until create finishes

do {

    DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

    .describeProjectVersions(describeProjectVersionsRequest);

    for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
        .projectVersionDescriptions()) {

        ProjectVersionStatus status =
projectVersionDescription.status();

        logger.log(Level.INFO, "stopping model: {0} ", modelArn);

        switch (status) {

            case STOPPED:
                logger.log(Level.INFO, "Model stopped");
                stopped = true;
                break;

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

            case FAILED:
```

```
        String error = "Model stopping failed: " +
projectVersionDescription.statusAsString() + " "
        + projectVersionDescription.statusMessage() + "
" + modelArn;

        logger.log(Level.SEVERE, error);
        throw new Exception(error);

    default:
        String unexpectedError = "Unexpected stopping state: "
+ projectVersionDescription.statusAsString() + "
"
        + projectVersionDescription.statusMessage() + "
" + modelArn;

        logger.log(Level.SEVERE, unexpectedError);
        throw new Exception(unexpectedError);
    }
}

} while (stopped == false);

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

}

public static void main(String[] args) {

    String modelArn = null;
    String projectArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>\n
\n" + "Where:\n"
        + "    project_arn - The ARN of the project that contains the
model that you want to stop. \n\n"
        + "    model_arn - The ARN of the model version that you want to
stop.\n\n";

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }
}
```

```
    }

    projectArn = args[0];
    modelArn = args[1];

    try {

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

        // Stop model
        stopMyModel(rekClient, projectArn, modelArn);

        System.out.println(String.format("Model stopped: %s", modelArn));

        rekClient.close();

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

}

}
```

## Melaporkan durasi berjalan dan unit inferensi yang digunakan

Jika Anda melatih dan memulai model setelah Agustus 2022, Anda dapat menggunakan CloudWatch metrik `InServiceInferenceUnits` Amazon untuk menentukan berapa jam model berjalan dan jumlah [unit inferensi](#) yang digunakan selama jam-jam tersebut.

**Note**

Jika Anda hanya memiliki satu model di suatu AWS wilayah, Anda juga bisa mendapatkan waktu berjalan untuk model tersebut dengan melacak panggilan yang berhasil ke `StartProjectVersion` dan `StopProjectVersion` masuk CloudWatch. Pendekatan ini tidak berfungsi jika Anda menjalankan lebih dari satu model di AWS Wilayah karena metrik tidak menyertakan informasi tentang model.

Atau, Anda dapat menggunakan AWS CloudTrail untuk melacak panggilan ke `StartProjectVersion` dan `StopProjectVersion` (yang mencakup model ARN di `requestParameters` bidang [riwayat acara](#)). CloudTrail acara dibatasi hingga 90 hari, tetapi Anda dapat menyimpan acara hingga 7 tahun di [CloudTrailDanau](#).

Prosedur berikut membuat grafik untuk yang berikut:

- Jumlah jam yang dijalankan oleh model.
- Jumlah unit inferensi yang digunakan model.

Anda dapat memilih jangka waktu hingga 15 bulan yang lalu. Untuk informasi selengkapnya tentang retensi metrik, lihat [Retensi metrik](#).

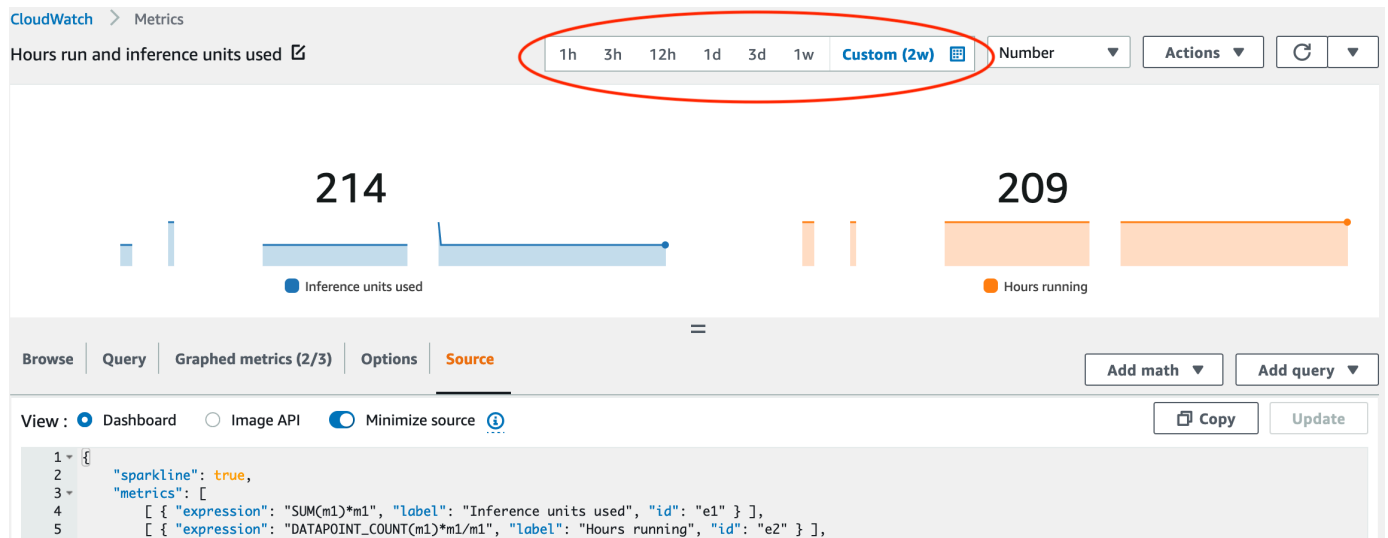
Untuk menentukan durasi model dan unit inferensi yang digunakan untuk model

1. Masuk ke Konsol Manajemen AWS dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Di panel navigasi kiri, pilih Semua metrik di bawah Metrik.
3. Di panel konten, pilih tab Sumber.
4. Pastikan tombol Dashboard dipilih.
5. Di kotak edit, ganti JSON yang ada dengan JSON berikut. Ubah nilai berikut:
  - `Project_Name`— Proyek yang berisi model yang ingin Anda grafik.
  - `Version_Name`— Versi model yang ingin Anda grafik.
  - `AWS_Region`— AWS Wilayah yang berisi model. Pastikan CloudWatch konsol berada di AWS Wilayah yang sama, dengan memeriksa pemilih Wilayah di bilah navigasi di bagian atas halaman. Perbarui seperlunya.

```
{
  "sparkline": true,
  "metrics": [
    [
      {
        "expression": "SUM(m1)*m1",
        "label": "Inference units used",
        "id": "e1"
      }
    ],
    [
      {
        "expression": "DATAPoint_COUNT(m1)*m1/m1",
        "label": "Hours running",
        "id": "e2"
      }
    ],
    [
      "AWS/Rekognition",
      "InServiceInferenceUnits",
      "ProjectName",
      "Project_Name",
      "VersionName",
      "Version_Name",
      {
        "id": "m1",
        "visible": false
      }
    ]
  ],
  "view": "singleValue",
  "stacked": false,
  "region": "AWS_Region",
  "stat": "Average",
  "period": 3600,
  "title": "Hours run and inference units used"
}
```

6. Pilih Perbarui.
7. Di bagian atas halaman, pilih garis waktu. Anda akan melihat angka untuk unit inferensi yang digunakan dan jam berjalan selama timeline. Kesenjangan dalam grafik menunjukkan waktu

ketika model tidak berjalan. Tangkapan layar konsol di bawah ini menunjukkan unit inferensi yang digunakan dan jam berjalan selama periode waktu, dengan waktu khusus 2 minggu ditetapkan, dengan nilai tertinggi 214 unit inferensi dan 209 jam berjalan.



- (Opsional) Tambahkan grafik ke dasbor dengan memilih Tindakan dan kemudian Tambahkan ke dasbor - ditingkatkan.

## Menganalisis gambar dengan model terlatih

Untuk menganalisis gambar dengan model Label Kustom Amazon Rekognition yang terlatih, Anda memanggil API. [DetectCustomLabels](#) Hasil dari `DetectCustomLabels` adalah prediksi bahwa gambar berisi objek, adegan, atau konsep tertentu.

Untuk menelepon `DetectCustomLabels`, Anda menentukan yang berikut:

- Nama Sumber Daya Amazon (ARN) dari model Label Kustom Rekognition Amazon yang ingin Anda gunakan.
- Gambar yang Anda inginkan untuk membuat prediksi dengan model. Anda dapat menyediakan citra input sebagai array bit citra (bit citra yang dikodekan base64), atau sebagai objek Amazon S3. Untuk informasi selengkapnya, lihat [Gambar](#).

Label kustom dikembalikan dalam array objek [Custom Label](#). Setiap label kustom mewakili satu objek, adegan, atau konsep yang ditemukan dalam gambar. Label khusus meliputi:

- Label untuk objek, adegan, atau konsep yang ditemukan dalam gambar.
- Kotak pembatas untuk objek yang ditemukan dalam gambar. Koordinat kotak pembatas menunjukkan di mana objek berada pada gambar sumber. Nilai koordinat adalah rasio ukuran citra secara keseluruhan. Untuk informasi lebih lanjut, lihat [Bounding Box](#). `DetectCustomLabels` mengembalikan kotak pembatas hanya jika model dilatih untuk mendeteksi lokasi objek.
- Keyakinan yang dimiliki Amazon Rekognition Custom Labels dalam keakuratan label dan kotak pembatas.

Untuk memfilter label berdasarkan kepercayaan deteksi, tentukan nilai `MinConfidence` yang sesuai dengan tingkat kepercayaan yang Anda inginkan. Misalnya, jika Anda harus sangat yakin dengan prediksi, tentukan nilai tinggi untuk `MinConfidence`. Untuk mendapatkan semua label, terlepas dari kepercayaan diri, tentukan `MinConfidence` nilai 0.

Kinerja model Anda diukur, sebagian, dengan metrik penarikan dan presisi yang dihitung selama pelatihan model. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#).

Untuk meningkatkan presisi model Anda, tetapkan nilai yang lebih tinggi untuk `MinConfidence`. Untuk informasi selengkapnya, lihat [Mengurangi positif palsu \(presisi yang lebih baik\)](#).

Untuk meningkatkan daya ingat model Anda, gunakan nilai yang lebih rendah untuk `MinConfidence`. Untuk informasi selengkapnya, lihat [Mengurangi negatif palsu \(mengingat lebih baik\)](#).

Jika Anda tidak menentukan nilai untuk `MinConfidence`, Amazon Rekognition Custom Labels mengembalikan label berdasarkan ambang batas yang diasumsikan untuk label tersebut. Untuk informasi selengkapnya, lihat [Ambang batas yang diasumsikan](#). Anda bisa mendapatkan nilai ambang batas yang diasumsikan untuk label dari hasil pelatihan model. Untuk informasi selengkapnya, lihat [Melatih model \(Konsol\)](#).

Dengan menggunakan parameter `MinConfidence` input, Anda menentukan ambang batas yang diinginkan untuk panggilan tersebut. Label yang terdeteksi dengan keyakinan di bawah nilai `MinConfidence` tidak dikembalikan dalam respons. Juga, ambang batas yang diasumsikan untuk label tidak memengaruhi penyertaan label dalam respons.

#### Note

Metrik Amazon Rekognition Custom Labels menyatakan ambang batas yang diasumsikan sebagai nilai floating point antara 0-1. Kisaran `MinConfidence` menormalkan ambang batas ke nilai persentase (0-100). Tanggapan kepercayaan dari `DetectCustomLabels` juga dikembalikan sebagai persentase.

Anda mungkin ingin menentukan ambang batas untuk label tertentu. Misalnya, ketika metrik presisi dapat diterima untuk Label A, tetapi tidak untuk Label B. Saat menentukan ambang batas (`MinConfidence`) yang berbeda, pertimbangkan hal berikut.

- Jika Anda hanya tertarik pada satu label (A), tetapkan nilai `MinConfidence` ke nilai ambang yang diinginkan. Sebagai tanggapan, prediksi untuk label A dikembalikan (bersama dengan label lain) hanya jika kepercayaan lebih besar dari `MinConfidence`. Anda perlu menyaring label lain yang dikembalikan.
- Jika Anda ingin menerapkan ambang batas yang berbeda ke beberapa label, lakukan hal berikut:
  1. Gunakan nilai 0 untuk `MinConfidence`. Nilai 0 memastikan bahwa semua label dikembalikan, terlepas dari kepercayaan deteksi.
  2. Untuk setiap label yang dikembalikan, terapkan ambang batas yang diinginkan dengan memeriksa apakah kepercayaan label lebih besar dari ambang batas yang Anda inginkan untuk label.

Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Amazon Rekognition yang terlatih](#).

Jika Anda menemukan nilai kepercayaan yang dikembalikan `DetectCustomLabels` terlalu rendah, pertimbangkan untuk melatih kembali modelnya. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#). Anda dapat membatasi jumlah label khusus yang dikembalikan `DetectCustomLabels` dengan menentukan parameter `MaxResults` input. Hasilnya dikembalikan diurutkan dari kepercayaan tertinggi ke terendah.

Untuk contoh lain yang memanggil `DetectCustomLabels`, lihat [Contoh Label Kustom](#).

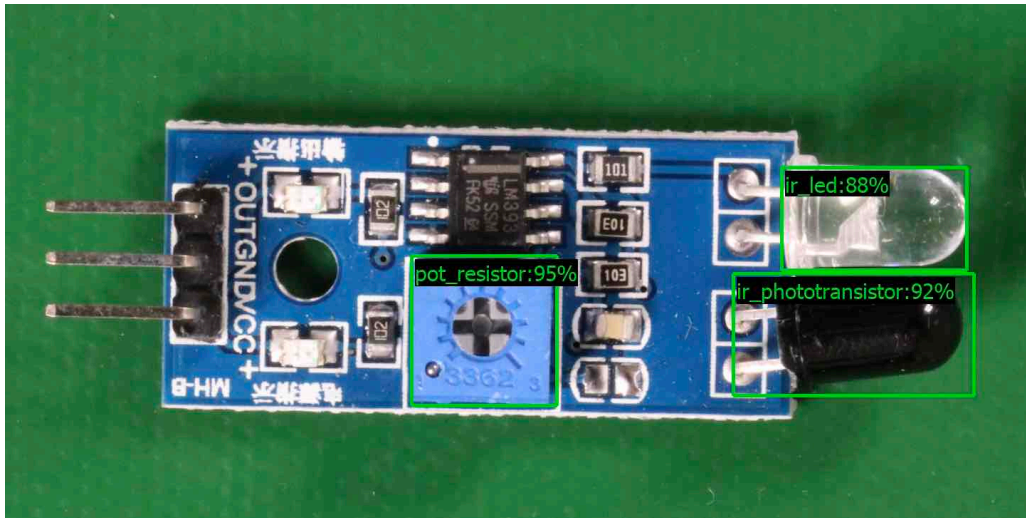
Untuk informasi tentang mengamankan `DetectCustomLabels`, lihat [Mengamankan DetectCustomLabels](#).

Untuk mendeteksi label kustom (API)

1. Jika belum:
  - a. Pastikan Anda memiliki `DetectCustomLabels` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Siapkan izin SDK](#).
  - b. Instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Latih dan terapkan model Anda. Untuk informasi selengkapnya, lihat [Membuat model Label Kustom Rekognition Amazon](#).
3. Pastikan panggilan pengguna `DetectCustomLabels` memiliki akses ke model yang Anda gunakan pada langkah 2. Untuk informasi selengkapnya, lihat [Mengamankan DetectCustomLabels](#).
4. Unggah gambar yang ingin Anda analisis ke bucket S3.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Contoh Python, Java, dan Java 2 juga menunjukkan cara menggunakan file gambar lokal untuk meneruskan gambar dengan menggunakan byte mentah. File harus lebih kecil dari 4 MB.

5. Gunakan contoh berikut untuk memanggil operasi `DetectCustomLabels`. Contoh Python dan Java menunjukkan gambar dan melapisi hasil analisis, mirip dengan gambar berikut. Gambar berikut berisi kotak pembatas dan label untuk papan sirkuit dengan potensiometer, fototransistor inframerah, dan komponen LED.



## AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi DetectCustomLabels CLI. Ubah nilai parameter input berikut.

- bucket dengan nama bucket Amazon S3 yang Anda gunakan di langkah 4.
- image dengan nama file gambar input yang Anda unggah di langkah 4.
- projectVersionArn dengan ARN model yang ingin Anda gunakan.

```
aws rekognition detect-custom-labels --project-version-arn model_arn \  
  --image '{"S3Object":{"Bucket":"bucket", "Name":"image"}}' \  
  --min-confidence 70 \  
  --profile custom-labels-access
```

## Python

Kode contoh berikut menampilkan kotak pembatas dan label tingkat gambar yang ditemukan dalam gambar.

Untuk menganalisis gambar lokal, jalankan program dan berikan argumen baris perintah berikut:

- ARN dari model yang ingin Anda analisis gambar.
- Nama dan lokasi file gambar lokal.

Untuk menganalisis gambar yang disimpan dalam bucket Amazon S3, jalankan program dan berikan argumen baris perintah berikut:

- ARN dari model yang ingin Anda analisis gambar.
- Nama dan lokasi gambar dalam bucket Amazon S3 yang Anda gunakan pada langkah 4.
- `--bucket`*bucket name*- Bucket Amazon S3 yang Anda gunakan pada langkah 4.

Perhatikan bahwa contoh ini mengasumsikan bahwa versi Pillow Anda adalah  $\geq 8.0.0$ .

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Amazon Rekognition Custom Labels detection example used in the service
documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/detecting-custom-
labels.html
Shows how to detect custom labels by using an Amazon Rekognition Custom Labels
model.
The image can be stored on your local computer or in an Amazon S3 bucket.
"""

import io
import logging
import argparse
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def analyze_local_image(rek_client, model, photo, min_confidence):
    """
    Analyzes an image stored as a local file.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
    want to use.
    """
```

```
:param photo: The name and file path of the photo that you want to analyze.
:param min_confidence: The desired threshold/confidence for the call.
"""

try:
    logger.info("Analyzing local file: %s", photo)
    image = Image.open(photo)
    image_type = Image.MIME[image.format]

    if (image_type == "image/jpeg" or image_type == "image/png") is False:
        logger.error("Invalid image type for %s", photo)
        raise ValueError(
            f"Invalid file format. Supply a jpeg or png format file:
{photo}"
        )

    # get images bytes for call to detect_anomalies
    image_bytes = io.BytesIO()
    image.save(image_bytes, format=image.format)
    image_bytes = image_bytes.getvalue()

    response = rek_client.detect_custom_labels(Image={'Bytes': image_bytes},
                                              MinConfidence=min_confidence,
                                              ProjectVersionArn=model)

    show_image(image, response)
    return len(response['CustomLabels'])

except ClientError as client_err:
    logger.error(format(client_err))
    raise
except FileNotFoundError as file_error:
    logger.error(format(file_error))
    raise

def analyze_s3_image(rek_client, s3_connection, model, bucket, photo,
                    min_confidence):
    """
    Analyzes an image stored in the specified S3 bucket.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
    want to use.
```

```
    :param bucket: The name of the S3 bucket that contains the image that you
    want to analyze.
    :param photo: The name of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        # Get image from S3 bucket.

        logger.info("analyzing bucket: %s image: %s", bucket, photo)
        s3_object = s3_connection.Object(bucket, photo)
        s3_response = s3_object.get()

        stream = io.BytesIO(s3_response['Body'].read())
        image = Image.open(stream)

        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
{photo}")

        ImageDraw.Draw(image)

        # Call DetectCustomLabels.
        response = rek_client.detect_custom_labels(
            Image={'S3Object': {'Bucket': bucket, 'Name': photo}},
            MinConfidence=min_confidence,
            ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as err:
        logger.error(format(err))
        raise

def show_image(image, response):
    """
    Displays the analyzed image and overlays analysis results
    :param image: The analyzed image
```

```
:param response: the response from DetectCustomLabels
"""
try:
    font_size = 40
    line_width = 5

    img_width, img_height = image.size
    draw = ImageDraw.Draw(image)

    # Calculate and display bounding boxes for each detected custom label.
    image_level_label_height = 0

    for custom_label in response['CustomLabels']:
        confidence = int(round(custom_label['Confidence'], 0))
        label_text = f"{custom_label['Name']}:{confidence}%"
        fnt = ImageFont.truetype('Tahoma.ttf', font_size)
        text_left, text_top, text_right, text_bottom = draw.textbbox((0, 0),
label_text, fnt)
        text_width, text_height = text_right - text_left, text_bottom -
text_top

        logger.info("Label: %s", custom_label['Name'])
        logger.info("Confidence: %s", confidence)

    # Draw bounding boxes, if present
    if 'Geometry' in custom_label:
        box = custom_label['Geometry']['BoundingBox']
        left = img_width * box['Left']
        top = img_height * box['Top']
        width = img_width * box['Width']
        height = img_height * box['Height']

        logger.info("Bounding box")
        logger.info("\tLeft: {0:.0f}".format(left))
        logger.info("\tTop: {0:.0f}".format(top))
        logger.info("\tLabel Width: {0:.0f}".format(width))
        logger.info("\tLabel Height: {0:.0f}".format(height))

        points = (
            (left, top),
            (left + width, top),
            (left + width, top + height),
            (left, top + height),
            (left, top))
```

```
        # Draw bounding box and label text
        draw.line(points, fill="limegreen", width=line_width)
        draw.rectangle([(left + line_width, top+line_width),
                        (left + text_width + line_width, top +
line_width + text_height)], fill="black")
        draw.text((left + line_width, top + line_width),
                  label_text, fill="limegreen", font=fnt)

        # draw image-level label text.
    else:
        draw.rectangle([(10, image_level_label_height),
                        (text_width + 10, image_level_label_height
+text_height)], fill="black")
        draw.text((10, image_level_label_height),
                  label_text, fill="limegreen", font=fnt)

        image_level_label_height += text_height

    image.show()

except Exception as err:
    logger.error(format(err))
    raise

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

    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "image", help="The path and file name of the image that you want to
analyze"
    )

    parser.add_argument(
        "--bucket", help="The bucket that contains the image. If not supplied,
image is assumed to be a local file.", required=False
    )
```

```
def main():

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

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

        label_count = 0
        min_confidence = 50

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

        if args.bucket is None:
            # Analyze local image.
            label_count = analyze_local_image(rekognition_client,
                                              args.model_arn,
                                              args.image,
                                              min_confidence)

        else:
            # Analyze image in S3 bucket.
            s3_connection = session.resource('s3')
            label_count = analyze_s3_image(rekognition_client,
                                          s3_connection,
                                          args.model_arn,
                                          args.bucket,
                                          args.image,
                                          min_confidence)

        print(f"Custom labels detected: {label_count}")

    except ClientError as client_err:
        print("A service client error occurred: " +
              format(client_err.response["Error"]["Message"]))

    except ValueError as value_err:
        print("A value error occurred: " + format(value_err))

    except FileNotFoundError as file_error:
```

```
        print("File not found error: " + format(file_error))

    except Exception as err:
        print("An error occurred: " + format(err))

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

## Java

Kode contoh berikut menampilkan kotak pembatas dan label tingkat gambar yang ditemukan dalam gambar.

Untuk menganalisis gambar lokal, jalankan program dan berikan argumen baris perintah berikut:

- ARN dari model yang ingin Anda analisis gambar.
- Nama dan lokasi file gambar lokal.

Untuk menganalisis gambar yang disimpan dalam bucket Amazon S3, jalankan program dan berikan argumen baris perintah berikut:

- ARN dari model yang ingin Anda analisis gambar.
- Nama dan lokasi gambar dalam bucket Amazon S3 yang Anda gunakan pada langkah 4.
- Bucket Amazon S3 yang berisi gambar yang Anda gunakan pada langkah 4.

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

package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
```

```
import java.io.FileNotFoundException;
import java.awt.font.FontRenderContext;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CustomLabel;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.util.IOUtils;

// Calls DetectCustomLabels and displays a bounding box around each detected
// image.
public class DetectCustomLabels extends JPanel {

    private transient DetectCustomLabelsResult response;
    private transient Dimension dimension;
    private transient BufferedImage image;

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

    // Finds custom labels in an image stored in an S3 bucket.
    public DetectCustomLabels(AmazonRekognition rekClient,
```

```
        AmazonS3 s3client,
        String projectVersionArn,
        String bucket,
        String key,
        Float minConfidence) throws AmazonRekognitionException,
AmazonS3Exception, IOException {

    logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
Object[] { bucket, key });

    // Get image from S3 bucket and create BufferedImage
    com.amazonaws.services.s3.model.S3Object s3object =
s3client.getObject(bucket, key);
    S3ObjectInputStream inputStream = s3object.getObjectContent();
    image = ImageIO.read(inputStream);

    // Set image size
    setWindowDimensions();

    DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
        .withProjectVersionArn(projectVersionArn)
        .withImage(new Image().withS3Object(new
S3Object().withName(key).withBucket(bucket)))
        .withMinConfidence(minConfidence);

    // Call DetectCustomLabels

    response = rekClient.detectCustomLabels(request);
    logFoundLabels(response.getCustomLabels());
    drawLabels();

}

// Finds custom label in a local image file.
public DetectCustomLabels(AmazonRekognition rekClient,
    String projectVersionArn,
    String photo,
    Float minConfidence)
    throws IOException, AmazonRekognitionException {

    logger.log(Level.INFO, "Processing local file: {0}", photo);

    // Get image bytes and buffered image
    ByteBuffer imageBytes;
```

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

// Get image for display
InputStream imageBytesStream;
imageBytesStream = new ByteArrayInputStream(imageBytes.array());

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

// Set image size
setWindowDimensions();

// Analyze image
DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
    .withProjectVersionArn(projectVersionArn)
    .withImage(new Image()
        .withBytes(imageBytes))
    .withMinConfidence(minConfidence);

response = rekClient.detectCustomLabels(request);

logFoundLabels(response.getCustomLabels());

drawLabels();
}

// Log the labels found by DetectCustomLabels
private void logFoundLabels(List<CustomLabel> customLabels) {
    logger.info("Custom labels found");
    if (customLabels.isEmpty()) {
        logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
    } else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                new Object[] { customLabel.getName(),
customLabel.getConfidence() });
        }
    }
}
```

```
}

// Sets window dimensions to 1/2 screen size, unless image is smaller
public void setWindowDimensions() {
    dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

    dimension.width = (int) dimension.getWidth() / 2;
    if (image.getWidth() < dimension.width) {
        dimension.width = image.getWidth();
    }
    dimension.height = (int) dimension.getHeight() / 2;

    if (image.getHeight() < dimension.height) {
        dimension.height = image.getHeight();
    }

    setPreferredSize(dimension);
}

// Draws the image containing the bounding boxes and labels.
@Override
public void paintComponent(Graphics g) {

    Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

    // Draw the image.
    g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);
}

public void drawLabels() {
    // Draws bounding boxes (if present) and label text.

    int boundingBoxBorderWidth = 5;
    int imageHeight = image.getHeight(this);
    int imageWidth = image.getWidth(this);

    // Set up drawing
    Graphics2D g2d = image.createGraphics();
    g2d.setColor(Color.GREEN);
    g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
    Font font = g2d.getFont();
    FontRenderContext frc = g2d.getFontRenderContext();
```

```
g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

List<CustomLabel> customLabels = response.getCustomLabels();

int imageLevelLabelHeight = 0;
for (CustomLabel customLabel : customLabels) {

    String label = customLabel.getName();

    int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
    int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

    // Draw bounding box, if present
    if (customLabel.getGeometry() != null) {

        BoundingBox box = customLabel.getGeometry().getBoundingBox();
        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Draw black rectangle
        g2d.setColor(Color.BLACK);
        g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                    textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

        // Write label onto black rectangle
        g2d.setColor(Color.GREEN);
        g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

        // Draw bounding box around label location
        g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.getWidth())),
                    Math.round((imageHeight * box.getHeight())));
    }
    // Draw image level labels.
    else {
        // Draw black rectangle
        g2d.setColor(Color.BLACK);
        g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
        g2d.setColor(Color.GREEN);
```

```
        g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

        imageLevelLabelHeight += textHeight;
    }

}

g2d.dispose();

}

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

    String photo = null;
    String bucket = null;
    String projectVersionArn = null;
    float minConfidence = 50;

    final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n"
        + "\n" + "Where:\n"
        + "    model_arn - The ARN of the model that you want to use. \n"
        + "\n"
        + "    image - The location of the image on your local file
        system or within an S3 bucket.\n\n"
        + "    bucket - The S3 bucket that contains the image. Don't
        specify if image is local.\n\n";

    // Collect the arguments. If 3 arguments are present, the image is
    assumed to be
    // in an S3 bucket.

    if (args.length < 2 || args.length > 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectVersionArn = args[0];
    photo = args[1];

    if (args.length == 3) {
        bucket = args[2];
    }

    DetectCustomLabels panel = null;
```

```
try {

    AWSCredentialsProvider provider =new
ProfileCredentialsProvider("custom-labels-access");

    AmazonRekognition rekClient =
AmazonRekognitionClientBuilder.standard()
        .withCredentials(provider)
        .withRegion(Regions.US_WEST_2)
        .build();

    AmazonS3 s3client = AmazonS3ClientBuilder.standard()
        .withCredentials(provider)
        .withRegion(Regions.US_WEST_2)
        .build();

    // Create frame and panel.
    JFrame frame = new JFrame("Custom Labels");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    if (args.length == 2) {
        // Analyze local image
        panel = new DetectCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
    } else {
        // Analyze image in S3 bucket
        panel = new DetectCustomLabels(rekClient, s3client,
projectVersionArn, bucket, photo, minConfidence);
    }

    frame.setContentPane(panel);
    frame.pack();
    frame.setVisible(true);

} catch (AmazonRekognitionException rekError) {
    String errorMessage = "Rekognition client error: " +
rekError.getMessage();
    logger.log(Level.SEVERE, errorMessage);
    System.out.println(errorMessage);
    System.exit(1);
} catch (FileNotFoundException fileError) {
    String errorMessage = "File not found: " + photo;
    logger.log(Level.SEVERE, errorMessage);
```

```
        System.out.println(errorMessage);
        System.exit(1);
    } catch (IOException fileError) {
        String errorMessage = "Input output exception: " +
fileError.getMessage();
        logger.log(Level.SEVERE, errorMessage);
        System.out.println(errorMessage);
        System.exit(1);
    } catch (AmazonS3Exception s3Error) {
        String errorMessage = "S3 error: " + s3Error.getErrorMessage();
        logger.log(Level.SEVERE, errorMessage);
        System.out.println(errorMessage);
        System.exit(1);
    }
}
}
```

## Java V2

Kode contoh berikut menampilkan kotak pembatas dan label tingkat gambar yang ditemukan dalam gambar.

Untuk menganalisis gambar lokal, jalankan program dan berikan argumen baris perintah berikut:

- `projectVersionArn`— ARN dari model yang ingin Anda analisis gambar.
- `photo`— nama dan lokasi file gambar lokal.

Untuk menganalisis gambar yang disimpan dalam bucket S3, jalankan program dan berikan argumen baris perintah berikut:

- ARN dari model yang ingin Anda analisis gambar.
- Nama dan lokasi gambar dalam bucket S3 yang Anda gunakan pada langkah 4.
- Bucket Amazon S3 yang berisi gambar yang Anda gunakan pada langkah 4.

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

```
package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import java.awt.*;
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

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

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {
```

```
private transient BufferedImage image;
private transient DetectCustomLabelsResponse response;
private transient Dimension dimension;
public static final Logger logger =
Logger.getLogger>ShowCustomLabels.class.getName());

// Finds custom labels in an image stored in an S3 bucket.
public ShowCustomLabels(RekognitionClient rekClient,
    S3Client s3client,
    String projectVersionArn,
    String bucket,
    String key,
    Float minConfidence) throws RekognitionException,
NoSuchBucketException, NoSuchKeyException, IOException {

    logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
Object[] { bucket, key });
    // Get image from S3 bucket and create BufferedImage
    GetObjectRequest requestObject =
GetObjectRequest.builder().bucket(bucket).key(key).build();
    ResponseBytes<GetObjectResponse> result =
s3client.getObject(requestObject, ResponseTransformer.toBytes());
    ByteArrayInputStream bis = new
ByteArrayInputStream(result.asByteArray());
    image = ImageIO.read(bis);

    // Set image size
    setWindowDimensions();

    // Construct request parameter for DetectCustomLabels
    S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();

    Image s3Image = Image.builder().s3Object(s3Object).build();

    DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

    response = rekClient.detectCustomLabels(request);
    logFoundLabels(response.customLabels());
    drawLabels();
```

```
}

// Finds custom label in a local image file.
public ShowCustomLabels(RekognitionClient rekClient,
    String projectVersionArn,
    String photo,
    Float minConfidence)
    throws IOException, RekognitionException {

    logger.log(Level.INFO, "Processing local file: {0}", photo);
    // Get image bytes and buffered image
    InputStream sourceStream = new FileInputStream(new File(photo));
    SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
    ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
    image = ImageIO.read(inputStream);

    setWindowDimensions();

    // Construct request parameter for DetectCustomLabels
    Image localImageBytes = Image.builder().bytes(imageBytes).build();

    DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

    response = rekClient.detectCustomLabels(request);

    logFoundLabels(response.customLabels());
    drawLabels();

}

// Sets window dimensions to 1/2 screen size, unless image is smaller
public void setWindowDimensions() {
    dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

    dimension.width = (int) dimension.getWidth() / 2;
    if (image.getWidth() < dimension.width) {
        dimension.width = image.getWidth();
    }
    dimension.height = (int) dimension.getHeight() / 2;
```

```
        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);
    }

    // Draws bounding boxes (if present) and label text.
    public void drawLabels() {

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.customLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.name();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.geometry() != null) {

                BoundingBox box = customLabel.geometry().boundingBox();
                float left = imageWidth * box.left();
                float top = imageHeight * box.top();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
```

```
        g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                    textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

        // Write label onto black rectangle
        g2d.setColor(Color.GREEN);
        g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

        // Draw bounding box around label location
        g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                    Math.round((imageHeight * box.height())));
    }
    // Draw image level labels.
    else {
        // Draw black rectangle
        g2d.setColor(Color.BLACK);
        g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
        g2d.setColor(Color.GREEN);
        g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

        imageLevelLabelHeight += textHeight;
    }

}
g2d.dispose();

}

// Log the labels found by DetectCustomLabels
private void logFoundLabels(List<CustomLabel> customLabels) {
    logger.info("Custom labels found:");
    if (customLabels.isEmpty()) {
        logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
    }
    else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
```

```
        new Object[] { customLabel.name(),
customLabel.confidence() } );
    }
}

// Draws the image containing the bounding boxes and labels.
@Override
public void paintComponent(Graphics g) {

    Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

    // Draw the image.
    g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

}

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

    String photo = null;
    String bucket = null;
    String projectVersionArn = null;

    final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n"
\n" + "Where:\n"
        + "    model_arn - The ARN of the model that you want to use. \n"
\n"
        + "    image - The location of the image on your local file
system or within an S3 bucket.\n\n"
        + "    bucket - The S3 bucket that contains the image. Don't
specify if image is local.\n\n";

    // Collect the arguments. If 3 arguments are present, the image is
assumed to be
    // in an S3 bucket.

    if (args.length < 2 || args.length > 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectVersionArn = args[0];
    photo = args[1];
}
```

```
    if (args.length == 3) {
        bucket = args[2];
    }

    float minConfidence = 50;

    ShowCustomLabels panel = null;

    try {
        // Get the Rekognition client

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

        S3Client s3Client = S3Client.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Create frame and panel.
        JFrame frame = new JFrame("Custom Labels");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        if (args.length == 2) {
            // Analyze local image
            panel = new ShowCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
        } else {
            // Analyze image in S3 bucket
            panel = new ShowCustomLabels(rekClient, s3Client,
projectVersionArn, bucket, photo, minConfidence);
        }

        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);
    }
```

```
    } catch (RekognitionException rekError) {  
  
        String errorMessage = "Rekognition client error: " +  
rekError.getMessage();  
        logger.log(Level.SEVERE, errorMessage);  
        System.out.println(errorMessage);  
        System.exit(1);  
    } catch (FileNotFoundException fileError) {  
        String errorMessage = "File not found: " + photo;  
        logger.log(Level.SEVERE, errorMessage);  
        System.out.println(errorMessage);  
        System.exit(1);  
    } catch (IOException fileError) {  
        String errorMessage = "Input output exception: " +  
fileError.getMessage();  
        logger.log(Level.SEVERE, errorMessage);  
        System.out.println(errorMessage);  
        System.exit(1);  
    } catch (NoSuchKeyException bucketError) {  
        String errorMessage = String.format("Image not found: %s in bucket  
%s.", photo, bucket);  
        logger.log(Level.SEVERE, errorMessage);  
        System.out.println(errorMessage);  
        System.exit(1);  
    } catch (NoSuchBucketException bucketError) {  
        String errorMessage = "Bucket not found: " + bucket;  
        logger.log(Level.SEVERE, errorMessage);  
        System.out.println(errorMessage);  
        System.exit(1);  
    }  
    }  
}
```

## DetectCustomLabels permintaan operasi

Dalam operasi `DetectCustomLabels`, Anda menyediakan citra input, sebagai array bit yang dikodekan base64 atau sebagai citra yang disimpan dalam bucket Amazon S3. Contoh permintaan JSON berikut menunjukkan citra yang dimuat dari bucket Amazon S3.

```
{  
  "ProjectVersionArn": "string",
```

```
"Image":{
  "S3Object":{
    "Bucket":"string",
    "Name":"string",
    "Version":"string"
  }
},
"MinConfidence": 90,
"MaxLabels": 10,
}
```

## DetectCustomLabels respon operasi

Respon JSON berikut dari DetectCustomLabels operasi menunjukkan label kustom yang terdeteksi pada gambar berikut.

```
{
  "CustomLabels": [
    {
      "Name": "MyLogo",
      "Confidence": 77.7729721069336,
      "Geometry": {
        "BoundingBox": {
          "Width": 0.198987677693367,
          "Height": 0.31296101212501526,
          "Left": 0.07924537360668182,
          "Top": 0.4037395715713501
        }
      }
    }
  ]
}
```

# Mengelola sumber daya Label Kustom Amazon Rekognition

Bagian ini memberi Anda gambaran umum tentang sumber daya Label Kustom Rekognition Amazon yang Anda gunakan untuk melatih dan mengelola model. Juga disertakan informasi ikhtisar untuk menggunakan AWS SDK untuk melatih dan menggunakan model.

Amazon Rekognition Custom Labels mengandalkan tiga sumber daya yang berbeda untuk mendeteksi label kustom Anda: project, dataset, dan model.

- **Proyek** - Digunakan untuk mengelompokkan sumber daya lain seperti dataset, versi model, dan evaluasi model bersama-sama.
- **Datasets** - Mendefinisikan gambar dan metadata terkait untuk digunakan dalam model pelatihan dan pengujian. Anda dapat membuat kumpulan data menggunakan file manifes format SageMaker AI atau dengan menyalin kumpulan data Label Kustom Rekognition Amazon yang ada.
- **Model** - Model matematika yang benar-benar memprediksi keberadaan objek, adegan, dan konsep dalam gambar dengan mengidentifikasi pola dalam gambar yang digunakan untuk melatih model.

## Topik

- [Mengelola proyek Label Kustom Rekognition Amazon](#)
- [Mengelola kumpulan data](#)
- [Mengelola model Label Kustom Rekognition Amazon](#)

# Mengelola proyek Label Kustom Rekognition Amazon

Dalam Label Kustom Amazon Rekognition, Anda menggunakan project untuk mengelola model yang Anda buat untuk kasus penggunaan tertentu. Sebuah proyek mengelola kumpulan data, pelatihan model, versi model, evaluasi model, dan menjalankan model proyek Anda.

## Topik

- [Menghapus proyek Label Kustom Rekognition Amazon](#)
- [Menjelaskan proyek \(SDK\)](#)
- [Membuat proyek dengan AWS CloudFormation](#)

## Menghapus proyek Label Kustom Rekognition Amazon

Anda dapat menghapus proyek dengan menggunakan konsol Amazon Rekognition atau dengan memanggil API. [DeleteProject](#) Untuk menghapus proyek, Anda harus terlebih dahulu menghapus setiap model terkait. Proyek atau model yang dihapus tidak dapat dihapus.

### Topik

- [Menghapus proyek Label Kustom Rekognition Amazon \(Konsol\)](#)
- [Menghapus proyek Amazon Rekognition Custom Labels \(SDK\)](#)

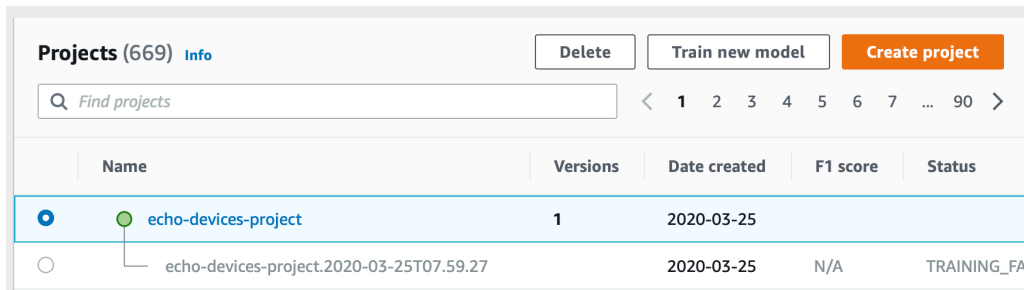
## Menghapus proyek Label Kustom Rekognition Amazon (Konsol)

Anda dapat menghapus proyek dari halaman proyek, atau Anda dapat menghapus proyek dari halaman detail proyek. Prosedur berikut menunjukkan cara menghapus proyek menggunakan halaman proyek.

Konsol Label Kustom Amazon Rekognition menghapus model dan kumpulan data terkait untuk Anda selama penghapusan proyek. Anda tidak dapat menghapus proyek jika salah satu modelnya sedang berjalan atau dilatih. Untuk menghentikan model yang sedang berjalan, lihat [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#). Jika model sedang dilatih, tunggu sampai selesai sebelum Anda menghapus proyek.

Untuk menghapus proyek (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Pada halaman Proyek, pilih tombol radio untuk proyek yang ingin Anda hapus. Daftar proyek ditampilkan echo-devices-project, dengan 1 versi dibuat pada 2020-03-25, dan opsi untuk Hapus, Latih model baru, atau Buat proyek.

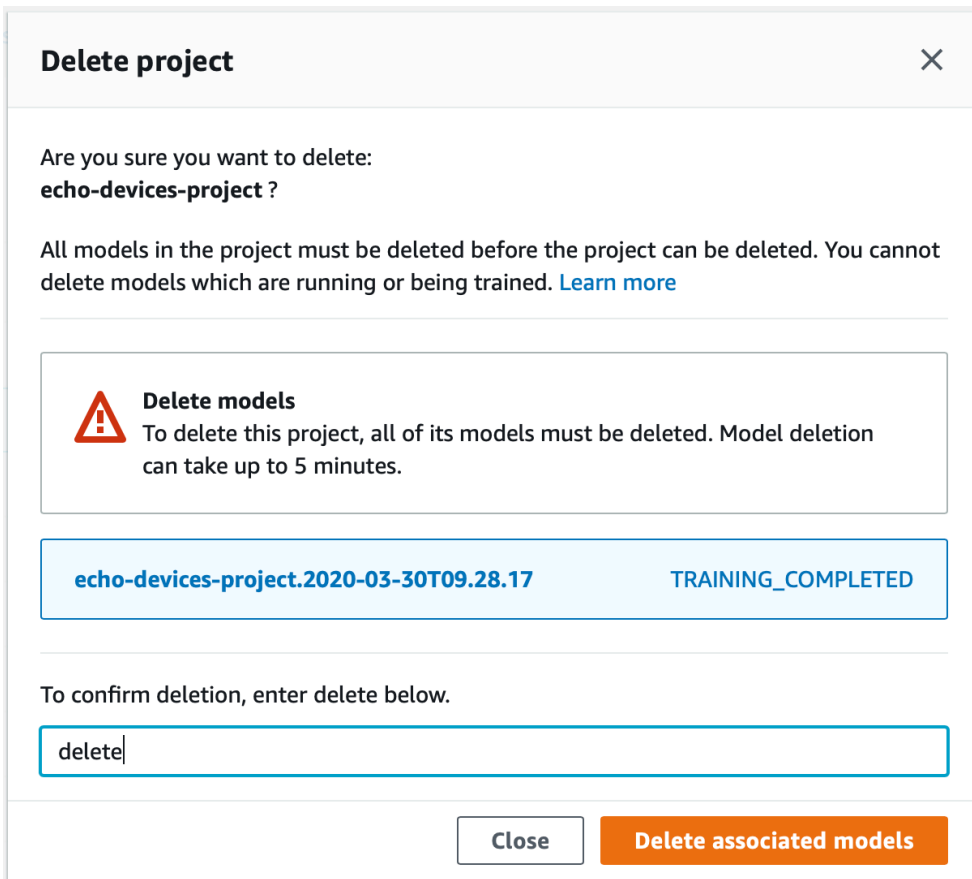


6. Pilih Hapus di bagian atas halaman. Kotak dialog Delete project ditampilkan.
7. Jika proyek tidak memiliki model terkait:
  - a. Masukkan hapus untuk menghapus proyek.
  - b. Pilih Hapus untuk menghapus proyek.
8. Jika proyek memiliki model atau kumpulan data terkait:
  - a. Masukkan hapus untuk mengonfirmasi bahwa Anda ingin menghapus model dan kumpulan data.
  - b. Pilih Hapus model terkait atau Hapus kumpulan data terkait atau Hapus kumpulan data dan model terkait, tergantung pada apakah model memiliki kumpulan data, model, atau keduanya. Penghapusan model mungkin membutuhkan waktu beberapa saat untuk diselesaikan.

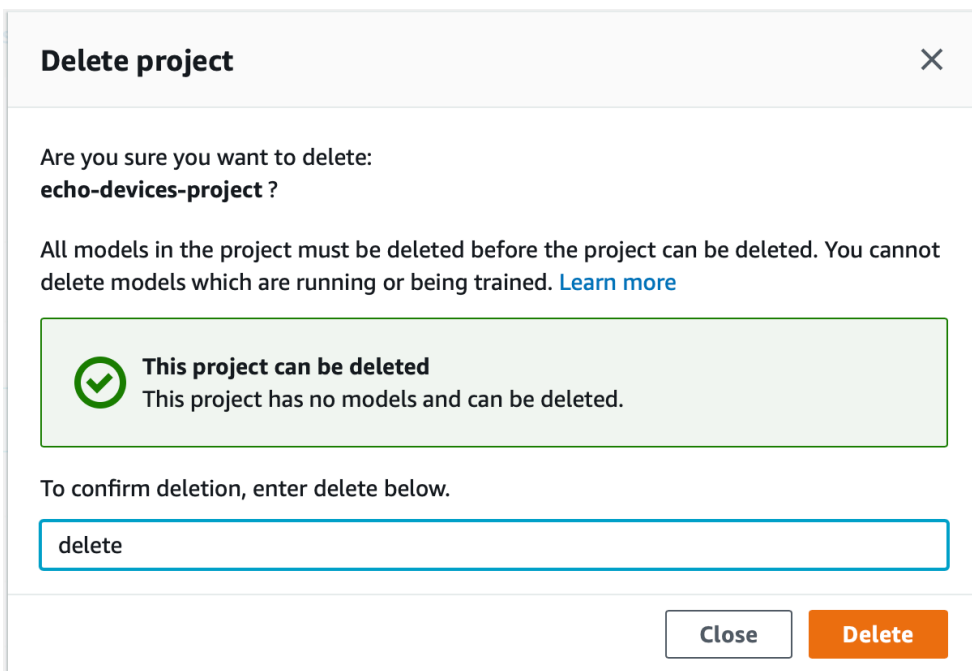
**Note**

Konsol tidak dapat menghapus model yang sedang dalam pelatihan atau berjalan. Coba lagi setelah menghentikan model yang sedang berjalan yang terdaftar, dan tunggu hingga model terdaftar sebagai pelatihan selesai. Jika Anda Menutup kotak dialog selama penghapusan model, model masih dihapus. Nanti, Anda dapat menghapus proyek dengan mengulangi prosedur ini.

Panel untuk menghapus model memberi Anda instruksi eksplisit untuk menghapus model terkait.



- c. Masukkan hapus untuk mengonfirmasi bahwa Anda ingin menghapus proyek.
- d. Pilih Hapus untuk menghapus proyek.



## Menghapus proyek Amazon Rekognition Custom Labels (SDK)

Anda menghapus proyek Label Kustom Rekognition Amazon dengan [DeleteProject](#) memanggil dan memasok Nama Sumber Daya Amazon (ARN) proyek yang ingin Anda hapus. Untuk mendapatkan ARNs proyek di AWS akun Anda, hubungi [DescribeProjects](#). Responsnya mencakup array [ProjectDescription](#) objek. Proyek ARN adalah lapangan. `ProjectArn` Anda dapat menggunakan nama proyek untuk mengidentifikasi ARN proyek. Misalnya, `arn:aws:rekognition:us-east-1:123456789010:project/project name/1234567890123`.

Sebelum Anda dapat menghapus proyek, Anda harus terlebih dahulu menghapus semua model dan kumpulan data dalam proyek. Untuk informasi selengkapnya, lihat [Menghapus model Label Kustom Rekognition Amazon \(SDK\)](#) dan [Menghapus dataset](#).

Proyek ini mungkin membutuhkan beberapa saat untuk dihapus. Selama waktu itu, status proyek adalah `DELETING`. Proyek akan dihapus jika panggilan berikutnya [DescribeProjects](#) tidak menyertakan proyek yang Anda hapus.

Untuk menghapus proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk menghapus proyek.

### AWS CLI

Ubah nilai `project-arn` menjadi nama proyek yang ingin Anda hapus.

```
aws rekognition delete-project --project-arn project_arn \  
--profile custom-labels-access
```

### Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda hapus.

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

```
"""
Purpose
Amazon Rekognition Custom Labels project example used in the service
documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-
project.html
Shows how to delete an existing Amazon Rekognition Custom Labels project.
You must first delete any models and datasets that belong to the project.
"""

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurrences.
    :param input_string: The string you want to search
    : n: the occurrence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position

def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)
```

```
response = rek_client.delete_project(ProjectArn=project_arn)

logger.info("project status: %s",response['Status'])

deleted = False

logger.info("waiting for project deletion: %s", project_arn)

# Get the project name
start = find_forward_slash(project_arn, 1) + 1
end = find_forward_slash(project_arn, 2)
project_name = project_arn[start:end]

project_names = [project_name]

while deleted is False:

    project_descriptions = rek_client.describe_projects(
        ProjectNames=project_names)['ProjectDescriptions']

    if len(project_descriptions) == 0:
        deleted = True

    else:
        time.sleep(5)

logger.info("project deleted: %s",project_arn)

return True

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

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

```
parser.add_argument(
    "project_arn", help="The ARN of the project that you want to delete."
)

def main():

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

    try:

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

        print(f"Deleting project: {args.project_arn}")

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

        delete_project(rekognition_client,
                       args.project_arn)

        print(f"Finished deleting project: {args.project_arn}")

    except ClientError as err:
        error_message = f"Problem deleting project: {err}"
        logger.exception(error_message)
        print(error_message)

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda hapus.

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

package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProject {

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

    public static void deleteMyProject(RekognitionClient rekClient, String
        projectArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting project: {0}", projectArn);

            // Delete the project

            DeleteProjectRequest deleteProjectRequest =
                DeleteProjectRequest.builder().projectArn(projectArn).build();
            DeleteProjectResponse response =
                rekClient.deleteProject(deleteProjectRequest);
```

```
        logger.log(Level.INFO, "Status: {0}", response.status());

        // Wait until deletion finishes

        Boolean deleted = false;

        do {

            DescribeProjectsRequest describeProjectsRequest =
DescribeProjectsRequest.builder().build();
                DescribeProjectsResponse describeResponse =
rekClient.describeProjects(describeProjectsRequest);
                List<ProjectDescription> projectDescriptions =
describeResponse.projectDescriptions();

                deleted = true;

                for (ProjectDescription projectDescription :
projectDescriptions) {

                    if (Objects.equals(projectDescription.projectArn(),
projectArn)) {

                        deleted = false;
                        logger.log(Level.INFO, "Not deleted: {0}",
projectDescription.projectArn());
                        Thread.sleep(5000);
                        break;
                    }
                }

            } while (Boolean.FALSE.equals(deleted));

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

        } catch (

            RekognitionException e) {
                logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                throw e;
            }
        }
    }
```

```
public static void main(String[] args) {

    final String USAGE = "\n" + "Usage: " + "<project_arn>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that you want to delete.\n\n";

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

    String projectArn = args[0];

    try {

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

        // Delete the project.
        deleteMyProject(rekClient, projectArn);

        System.out.println(String.format("Project deleted: %s",
projectArn));

        rekClient.close();

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

    catch (InterruptedException intError) {
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
        System.exit(1);
    }

}
```

```
}
```

## Menjelaskan proyek (SDK)

Anda dapat menggunakan `DescribeProjects` API untuk mendapatkan informasi tentang proyek Anda.

Untuk menggambarkan proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan contoh kode berikut untuk menggambarkan sebuah proyek. Ganti `project_name` dengan nama proyek yang ingin Anda gambarkan. Jika Anda tidak menentukan `--project-names`, deskripsi untuk semua proyek dikembalikan.

### AWS CLI

```
aws rekognition describe-projects --project-names project_name \  
  --profile custom-labels-access
```

### Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_name` — nama proyek yang ingin Anda gambarkan. Jika Anda tidak menentukan nama, deskripsi untuk semua proyek akan dikembalikan.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to describe an Amazon Rekognition Custom Labels project.  
"""  
  
import argparse  
import logging  
import json  
import boto3  
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

def display_project_info(project):
    """
    Displays information about a Custom Labels project.
    :param project: The project that you want to display information about.
    """
    print(f"Arn: {project['ProjectArn']}")
    print(f"Status: {project['Status']}")

    if len(project['Datasets']) == 0:
        print("Datasets: None")
    else:
        print("Datasets:")

        for dataset in project['Datasets']:
            print(f"\tCreated: {str(dataset['CreationTimestamp'])}")
            print(f"\tType: {dataset['DatasetType']}")
            print(f"\tARN: {dataset['DatasetArn']}")
            print(f"\tStatus: {dataset['Status']}")
            print(f"\tStatus message: {dataset['StatusMessage']}")
            print(f"\tStatus code: {dataset['StatusMessageCode']}")
            print()
        print()

def describe_projects(rek_client, project_name):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The project you want to describe. Pass None to describe
    all projects.
    """

    try:
        # Describe the project
        if project_name is None:
            logger.info("Describing all projects.")
        else:
            logger.info("Describing project: %s.", project_name)

        if project_name is None:
```

```
        response = rek_client.describe_projects()
    else:
        project_names = json.loads('["' + project_name + "']')
        response = rek_client.describe_projects(ProjectNames=project_names)

    print('Projects\n-----')
    if len(response['ProjectDescriptions']) == 0:
        print("Project(s) not found.")
    else:
        for project in response['ProjectDescriptions']:
            display_project_info(project)

    logger.info("Finished project description.")

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

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

    parser.add_argument(
        "--project_name", help="The name of the project that you want to
describe.", required=False
    )

def main():

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

    try:

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

```

args = parser.parse_args()

print(f"Describing projects: {args.project_name}")

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

describe_projects(rekognition_client,
                  args.project_name)

if args.project_name is None:
    print("Finished describing all projects.")
else:
    print("Finished describing project %s.", args.project_name)

except ClientError as err:
    error_message = f"Problem describing project: {err}"
    logger.exception(error_message)
    print(error_message)

if __name__ == "__main__":
    main()

```

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_name`— ARN dari proyek yang ingin Anda gambarkan. Jika Anda tidak menentukan nama, deskripsi untuk semua proyek akan dikembalikan.

```

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

package com.example.rekognition;

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

```

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DescribeProjects {

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

    public static void describeMyProjects(RekognitionClient rekClient, String
        projectName) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument

        List<String> projectNames = new ArrayList<String>();

        if (projectName == null) {
            descProjects = DescribeProjectsRequest.builder().build();
        } else {
            projectNames.add(projectName);
            descProjects =
                DescribeProjectsRequest.builder().projectNames(projectNames).build();
        }

        // Display useful information for each project.

        DescribeProjectsResponse resp =
            rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
        {

            System.out.println("ARN: " + projectDescription.projectArn());
        }
    }
}
```

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

}

public static void main(String[] args) {

    String projectArn = null;

    // Get command line arguments

    final String USAGE = "\n" + "Usage: " + "<project_name>\n\n" + "Where:
\n"
        + "    project_name - (Optional) The name of the project that you
want to describe. If not specified, all projects "
        + "are described.\n\n";

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

    if (args.length == 1) {
        projectArn = args[0];
    }

    try {

        // Get the Rekognition client
        RekognitionClient rekClient = RekognitionClient.builder()
```

```
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

    // Describe projects

    describeMyProjects(rekClient, projectArn);

    rekClient.close();

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

}

}
```

## Membuat proyek dengan AWS CloudFormation

Amazon Rekognition Custom Labels terintegrasi AWS CloudFormation dengan, layanan yang membantu Anda memodelkan dan menyiapkan sumber daya sehingga AWS Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat template yang menjelaskan semua AWS sumber daya yang Anda inginkan, dan CloudFormation mengurus penyediaan dan konfigurasi sumber daya tersebut untuk Anda.

Anda dapat menggunakan CloudFormation untuk menyediakan dan mengonfigurasi proyek Label Kustom Amazon Rekognition.

Bila Anda menggunakan CloudFormation, Anda dapat menggunakan kembali template Anda untuk menyiapkan proyek Amazon Rekognition Custom Labels secara konsisten dan berulang kali. Cukup jelaskan proyek Anda sekali, lalu sediakan proyek yang sama berulang-ulang di beberapa AWS akun dan Wilayah.

## Amazon Rekognition Custom Label dan template CloudFormation

[Untuk menyediakan dan mengonfigurasi proyek untuk Label Kustom Amazon Rekognition dan layanan terkait, Anda harus memahami templat.CloudFormation](#) Templat adalah file teks dengan

format JSON atau YAML. Template ini menjelaskan sumber daya yang ingin Anda sediakan di CloudFormation tumpukan Anda. Jika Anda tidak terbiasa dengan JSON atau YAMAL, Anda dapat menggunakan CloudFormation Designer untuk membantu Anda memulai dengan template. CloudFormation Untuk informasi selengkapnya, lihat [Apa itu CloudFormation Designer?](#) di Panduan Pengguna AWS CloudFormation .

Untuk informasi referensi tentang proyek Label Kustom Rekognition Amazon, termasuk contoh templat JSON dan YAMAL, [lihat](#) Referensi jenis sumber daya Rekognition.

## Pelajari lebih lanjut tentang CloudFormation

Untuk mempelajari selengkapnya CloudFormation, lihat sumber daya berikut:

- [AWS CloudFormation](#)
- [AWS CloudFormation Panduan Pengguna](#)
- [CloudFormation Referensi API](#)
- [AWS CloudFormation Panduan Pengguna Antarmuka Baris Perintah](#)

## Mengelola kumpulan data

Dataset berisi gambar dan label yang ditetapkan yang Anda gunakan untuk melatih atau menguji model. Topik di bagian ini menunjukkan cara mengelola kumpulan data dengan konsol Amazon Rekognition Custom Labels dan SDK. AWS

### Topik

- [Menambahkan kumpulan data ke proyek](#)
- [Menambahkan lebih banyak gambar ke kumpulan data](#)
- [Membuat kumpulan data menggunakan dataset \(SDK\) yang ada](#)
- [Menjelaskan kumpulan data \(SDK\)](#)
- [Daftar entri kumpulan data \(SDK\)](#)
- [Mendistribusikan dataset pelatihan \(SDK\)](#)
- [Menghapus dataset](#)

## Menambahkan kumpulan data ke proyek

Anda dapat menambahkan kumpulan data pelatihan atau kumpulan data pengujian ke proyek yang sudah ada. Jika Anda ingin mengganti kumpulan data yang ada, hapus dulu kumpulan data yang ada. Untuk informasi selengkapnya, lihat [Menghapus dataset](#). Kemudian, tambahkan dataset baru.

### Topik

- [Menambahkan kumpulan data ke proyek \(Konsol\)](#)
- [Menambahkan kumpulan data ke proyek \(SDK\)](#)

## Menambahkan kumpulan data ke proyek (Konsol)

Anda dapat menambahkan kumpulan data pelatihan atau pengujian ke project menggunakan konsol Amazon Rekognition Custom Labels.

Untuk menambahkan kumpulan data ke proyek

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan.
3. Di panel navigasi kiri, pilih Proyek. Tampilan Proyek ditampilkan.
4. Pilih proyek yang ingin Anda tambahkan dataset.
5. Di panel navigasi kiri, di bawah nama proyek, pilih Datasets.
6. Jika proyek tidak memiliki kumpulan data yang ada, halaman Buat dataset akan ditampilkan. Lakukan hal-hal berikut:
  - a. Pada halaman Buat kumpulan data, masukkan informasi sumber gambar. Untuk informasi selengkapnya, lihat [the section called "Membuat dataset dengan gambar"](#).
  - b. Pilih Buat kumpulan data untuk membuat kumpulan data.
7. Jika proyek memiliki kumpulan data yang ada (pelatihan atau pengujian), halaman detail proyek ditampilkan. Lakukan hal-hal berikut:
  - a. Pada halaman detail proyek, pilih Tindakan.
  - b. Jika Anda ingin menambahkan kumpulan data pelatihan, pilih Buat kumpulan data pelatihan.
  - c. Jika Anda ingin menambahkan kumpulan data pengujian, pilih Buat kumpulan data pengujian.

- d. Pada halaman Buat kumpulan data, masukkan informasi sumber gambar. Untuk informasi selengkapnya, lihat [the section called “Membuat dataset dengan gambar”](#).
  - e. Pilih Buat kumpulan data untuk membuat kumpulan data.
8. Tambahkan gambar ke kumpulan data Anda. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar \(konsol\)](#).
  9. Tambahkan label ke kumpulan data Anda. Untuk informasi selengkapnya, lihat [Tambahkan label baru \(Konsol\)](#).
  10. Tambahkan label ke gambar Anda. Jika Anda menambahkan label tingkat gambar, lihat [the section called “Menetapkan label tingkat gambar ke gambar”](#) Jika Anda menambahkan kotak pembatas, lihat [Pelabelan objek dengan kotak pembatas](#). Untuk informasi selengkapnya, lihat [Mengarahkan kumpulan data](#).

## Menambahkan kumpulan data ke proyek (SDK)

Anda dapat menambahkan kumpulan data kereta atau pengujian ke proyek yang ada dengan cara berikut:

- Buat kumpulan data menggunakan file manifes. Untuk informasi selengkapnya, lihat [Membuat kumpulan data dengan file manifes SageMaker AI Ground Truth \(SDK\)](#).
- Buat kumpulan data kosong dan isi kumpulan data sesudahnya. Contoh berikut menunjukkan cara membuat dataset kosong. Untuk menambahkan entri setelah Anda membuat kumpulan data kosong, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#)

Untuk menambahkan kumpulan data ke proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan contoh berikut untuk menambahkan baris JSON ke kumpulan data.

CLI

Ganti `project_arn` dengan proyek yang ingin Anda tambahkan set dataset. Ganti `dataset_type` dengan TRAIN untuk membuat kumpulan data pelatihan, atau TEST untuk membuat kumpulan data pengujian.

```
aws rekognition create-dataset --project-arn project_arn \
```

```
--dataset-type dataset_type \  
--profile custom-labels-access
```

## Python

Gunakan kode berikut untuk membuat kumpulan data. Berikan opsi baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda tambahkan dataset pengujian.
- `type`— jenis dataset yang ingin Anda buat (melatih atau menguji)

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
import argparse  
import logging  
import time  
import boto3  
  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def create_empty_dataset(rek_client, project_arn, dataset_type):  
    """  
    Creates an empty Amazon Rekognition Custom Labels dataset.  
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.  
    :param project_arn: The ARN of the project in which you want to create a  
    dataset.  
    :param dataset_type: The type of the dataset that you want to create (train  
    or test).  
    """  
  
    try:  
        #Create the dataset.  
        logger.info("Creating empty %s dataset for project %s",  
                    dataset_type, project_arn)  
  
        dataset_type=dataset_type.upper()  
  
        response = rek_client.create_dataset(  
            ProjectArn=project_arn, DatasetType=dataset_type  
        )
```

```
dataset_arn=response['DatasetArn']

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

finished=False
while finished is False:

    dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

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

    if status == "CREATE_IN_PROGRESS":

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

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

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

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

    return dataset_arn

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

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

```
:param parser: The command line parser.
"""

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

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

def main():

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

    try:

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

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

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

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

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

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

```
print(f"Problem creating empty dataset: {err}")

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

## Java V2

Gunakan kode berikut untuk membuat kumpulan data. Berikan opsi baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda tambahkan dataset pengujian.
- `type`— jenis dataset yang ingin Anda buat (melatih atau menguji)

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class CreateEmptyDataset {

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

```
public static String createMyEmptyDataset(RekognitionClient rekClient,
String projectArn, String datasetType)
    throws Exception, RekognitionException {

    try {

        logger.log(Level.INFO, "Creating empty {0} dataset for project :
{1}",
            new Object[] { datasetType.toString(), projectArn });

        DatasetType requestDatasetType = null;

        switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
                throw new Exception("Unrecognized dataset type: " +
datasetType);
        }

        CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
            .datasetType(requestDatasetType).build();

        CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

        boolean created = false;

        //Wait until updates finishes

        do {

            DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                .datasetArn(response.datasetArn()).build();
```

```
        DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        DatasetStatus status = datasetDescription.status();

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

        switch (status) {

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

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

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

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

    } while (created == false);

    return response.datasetArn();

} catch (RekognitionException e) {
```

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

}

public static void main(String args[]) {

    String datasetType = null;
    String datasetArn = null;
    String projectArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n
\n" + "Where:\n"
        + "    project_arn - the ARN of the project that you want to add
copy the data to.\n\n"
        + "    dataset_type - the type of the empty dataset that you want
to create (train or test).\n\n";

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];
    datasetType = args[1];

    try {

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

        // Create the dataset
        datasetArn = createMyEmptyDataset(rekClient, projectArn,
datasetType);
    }
}
```

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

        rekClient.close();

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

3. Tambahkan gambar ke kumpulan data. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar \(SDK\)](#).

## Menambahkan lebih banyak gambar ke kumpulan data

Anda dapat menambahkan lebih banyak gambar ke kumpulan data menggunakan konsol Amazon Rekognition Custom Labels atau dengan memanggil API. `UpdateDatasetEntries`

Topik

- [Menambahkan lebih banyak gambar \(konsol\)](#)
- [Menambahkan lebih banyak gambar \(SDK\)](#)

### Menambahkan lebih banyak gambar (konsol)

Saat Anda menggunakan konsol Amazon Rekognition Custom Labels, Anda mengunggah gambar dari komputer lokal Anda. Gambar ditambahkan ke lokasi bucket Amazon S3 (konsol atau eksternal) tempat penyimpanan gambar yang digunakan untuk membuat kumpulan data.

Untuk menambahkan lebih banyak gambar ke kumpulan data Anda (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>

2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan.
3. Di panel navigasi kiri, pilih Proyek. Tampilan Proyek ditampilkan.
4. Pilih proyek yang ingin Anda gunakan.
5. Di panel navigasi kiri, di bawah nama proyek, pilih Dataset.
6. Pilih Tindakan dan pilih kumpulan data yang ingin Anda tambahkan gambar.
7. Pilih gambar yang ingin Anda unggah ke kumpulan data. Anda dapat menyeret gambar atau memilih gambar yang ingin Anda unggah dari komputer lokal Anda. Anda dapat mengunggah hingga 30 gambar sekaligus.
8. Pilih Unggah gambar.
9. Pilih Simpan perubahan.
10. Beri label pada gambar. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#).

## Menambahkan lebih banyak gambar (SDK)

UpdateDatasetEntries memperbarui atau menambahkan baris JSON ke file manifes. Anda melewati garis JSON sebagai objek data yang dikodekan byte64 di lapangan. GroundTruth Jika Anda menggunakan AWS SDK untuk memanggil UpdateDatasetEntries, SDK mengkodekan data untuk Anda. Setiap baris JSON berisi informasi untuk satu gambar, seperti label yang ditetapkan atau informasi kotak pembatas. Contoh:

```
{"source-ref":"s3://bucket/image","BB":{"annotations":
[{"left":1849,"top":1039,"width":422,"height":283,"class_id":0},
{"left":1849,"top":1340,"width":443,"height":415,"class_id":1},
{"left":2637,"top":1380,"width":676,"height":338,"class_id":2},
{"left":2634,"top":1051,"width":673,"height":338,"class_id":3}],"image_size":
[{"width":4000,"height":2667,"depth":3}]},"BB-metadata":{"job-name":"labeling-job/
BB"},"class-map":
{"0":"comparator","1":"pot_resistor","2":"ir_phototransistor","3":"ir_led"},"human-
annotated":"yes","objects":[{"confidence":1},{confidence":1},{confidence":1},
{"confidence":1}],"creation-date":"2021-06-22T10:11:18.006Z","type":"groundtruth/
object-detection"}}
```

Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Gunakan `source-ref` bidang sebagai kunci untuk mengidentifikasi gambar yang ingin Anda perbarui. Jika kumpulan data tidak berisi nilai `source-ref` bidang yang cocok, baris JSON ditambahkan sebagai gambar baru.

Untuk menambahkan lebih banyak gambar ke kumpulan data (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan contoh berikut untuk menambahkan baris JSON ke kumpulan data.

## CLI

Ganti nilai `GroundTruth` dengan JSON Lines yang ingin Anda gunakan. Anda harus melarikan diri dari karakter khusus dalam JSON Line.

```
aws rekognition update-dataset-entries\
  --dataset-arn dataset_arn \
  --changes '{"GroundTruth" : "{\\"source-ref\\":\\"s3://your_bucket/your_image
\\",\\"BB\\":{\\"annotations\\":[{\\"left\\":1776,\\"top\\":1017,\\"width\\":458,\\"height
\\":317,\\"class_id\\":0},{\\"left\\":1797,\\"top\\":1334,\\"width\\":418,\\"height
\\":415,\\"class_id\\":1},{\\"left\\":2597,\\"top\\":1361,\\"width\\":655,\\"height
\\":329,\\"class_id\\":2},{\\"left\\":2581,\\"top\\":1020,\\"width\\":689,\\"height
\\":338,\\"class_id\\":3}],\\"image_size\\":[{\\"width\\":4000,\\"height\\":2667,
\\"depth\\":3}],\\"BB-metadata\\":{\\"job-name\\":\\"labeling-job/BB\\",\\"class-map
\\":{\\"0\\":\\"comparator\\",\\"1\\":\\"pot_resistor\\",\\"2\\":\\"ir_phototransistor\\",
\\"3\\":\\"ir_led\\"},\\"human-annotated\\":\\"yes\\",\\"objects\\":[{\\"confidence\\":1},
{\\"confidence\\":1},{\\"confidence\\":1}],\\"creation-date\\":
\\"2021-06-22T10:10:48.492Z\\",\\"type\\":\\"groundtruth/object-detection\\"}]"}' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

## Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `dataset_arn` — ARN dari dataset yang ingin Anda perbarui.
- `updates_file` — file yang berisi pembaruan JSON Line.

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

```
"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
    updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)

        with open(updates_file) as f:
            manifest_file = f.read()

        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )
```

```
finished=False
while finished is False:

    dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

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

    if status == "UPDATE_IN_PROGRESS":

        logger.info("Updating dataset: %s ", dataset_arn)
        time.sleep(5)
        continue

    if status == "UPDATE_COMPLETE":
        logger.info("Dataset updated: %s : %s : %s",
            status, status_message, dataset_arn)
        finished=True
        continue

    if status == "UPDATE_FAILED":
        error_message = f"Dataset update failed: {status} :
{status_message} : {dataset_arn}"
        logger.exception(error_message)
        raise Exception (error_message)

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

    logger.info("Added entries to dataset")

    return status, status_message

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

def add_arguments(parser):
    """
```

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

parser.add_argument(
    "dataset_arn", help="The ARN of the dataset that you want to update."
)

parser.add_argument(
    "updates_file", help="The manifest file of JSON Lines that contains the
updates."
)

def main():

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

    try:

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

        print(f"Updating dataset {args.dataset_arn} with entries from
{args.updates_file}.")

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

        status, status_message=update_dataset_entries(rekognition_client,
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")

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

    except Exception as err:
        logger.exception("Problem updating dataset: %s", err)
```

```
print(f"Problem updating dataset: {err}")

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

## Java V2

- `dataset_arn` — ARN dari dataset yang ingin Anda perbarui.
- `update_file` — file yang berisi pembaruan JSON Line.

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetChanges;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesRequest;
import
    software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesResponse;

import java.io.FileInputStream;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

public class UpdateDatasetEntries {
```

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

public static String updateMyDataset(RekognitionClient rekClient, String
datasetArn,
String updateFile
) throws Exception, RekognitionException {

try {

logger.log(Level.INFO, "Updating dataset {0}",
new Object[] { datasetArn});

InputStream sourceStream = new FileInputStream(updateFile);
SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

DatasetChanges datasetChanges = DatasetChanges.builder()
    .groundTruth(sourceBytes).build();

UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
    .changes(datasetChanges)
    .datasetArn(datasetArn)
    .build();

UpdateDatasetEntriesResponse response =
rekClient.updateDatasetEntries(updateDatasetEntriesRequest);

boolean updated = false;

//Wait until update completes

do {

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

DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();
```

```
DatasetStatus status = datasetDescription.status();

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

switch (status) {

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

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

case UPDATE_FAILED:
    String error = "Dataset update failed: " +
datasetDescription.statusAsString() + " "
                + datasetDescription.statusMessage() + " " +
datasetArn;

    logger.log(Level.SEVERE, error);
    throw new Exception(error);

default:
    String unexpectedError = "Unexpected update state: " +
datasetDescription.statusAsString() + " "
                + datasetDescription.statusMessage() + " " +
datasetArn;

    logger.log(Level.SEVERE, unexpectedError);
    throw new Exception(unexpectedError);
}

} while (updated == false);

return datasetArn;

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

}
```

```
public static void main(String args[]) {

    String updatesFile = null;
    String datasetArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
<updates_file>\n\n" + "Where:\n"
        + "    dataset_arn - the ARN of the dataset that you want to
update.\n\n"
        + "    update_file - The file that includes in JSON Line updates.
\n\n";

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }

    datasetArn = args[0];
    updatesFile = args[1];

    try {

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

        // Update the dataset
        datasetArn = updateMyDataset(rekClient, datasetArn, updatesFile);

        System.out.println(String.format("Dataset updated: %s",
datasetArn));

        rekClient.close();

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

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

## Membuat kumpulan data menggunakan dataset (SDK) yang ada

Prosedur berikut menunjukkan cara membuat kumpulan data dari kumpulan data yang ada dengan menggunakan operasi. [CreateDataset](#)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode contoh berikut untuk membuat kumpulan data dengan menyalin kumpulan data lain.

### AWS CLI

Gunakan kode berikut untuk membuat kumpulan data. Ganti yang berikut ini:

- `project_arn`— ARN dari proyek yang ingin Anda tambahkan dataset.
- `dataset_type`— dengan jenis dataset (TRAIN atau TEST) yang ingin Anda buat dalam proyek.
- `dataset_arn`— dengan ARN dari dataset yang ingin Anda salin.

```
aws rekognition create-dataset --project-arn project_arn \  
  --dataset-type dataset_type \  
  --dataset-source '{ "DatasetArn" : "dataset_arn" }' \  
  --profile custom-labels-access
```

### Python

Contoh berikut membuat dataset menggunakan dataset yang ada dan menampilkan ARN nya.

Untuk menjalankan program, berikan argumen baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda gunakan.
- `dataset_type`— jenis dataset proyek yang ingin Anda buat (`trainatautest`).
- `dataset_arn`— ARN dari dataset tempat Anda ingin membuat dataset dari.

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

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset_from_existing_dataset(rek_client, project_arn, dataset_type,
dataset_arn):
    """
    Creates an Amazon Rekognition Custom Labels dataset using an existing
    dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
    dataset.
    :param dataset_type: The type of the dataset that you want to create (train
    or test).
    :param dataset_arn: The ARN of the existing dataset that you want to use.
    """

    try:
        # Create the dataset

        dataset_type=dataset_type.upper()

        logger.info(
            "Creating %s dataset for project %s from dataset %s.",
            dataset_type,project_arn, dataset_arn)
```

```
dataset_source = json.loads(
    '{ "DatasetArn": "' + dataset_arn + "'"
)

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

dataset_arn = response['DatasetArn']

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

finished = False
while finished is False:

    dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

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

    if status == "CREATE_IN_PROGRESS":

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

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

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

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

return dataset_arn
```

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

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

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

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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to copy from."
    )

def main():

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

    try:

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

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

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

dataset_arn = create_dataset_from_existing_dataset(rekognition_client,
                                                  args.project_arn,
                                                  args.dataset_type,
                                                  args.dataset_arn)

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

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

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

## Java V2

Contoh berikut membuat dataset menggunakan dataset yang ada dan menampilkan ARN nya.

Untuk menjalankan program, berikan argumen baris perintah berikut:

- `project_arn`— ARN dari proyek yang ingin Anda gunakan.
- `dataset_type`— jenis dataset proyek yang ingin Anda buat (`trainatautest`).
- `dataset_arn`— ARN dari dataset tempat Anda ingin membuat dataset dari.

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class CreateDatasetExisting {

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

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

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
                dataset {2} ",
                new Object[] { datasetType.toString(), projectArn,
                    existingDatasetArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
                case "train":
                    requestDatasetType = DatasetType.TRAIN;
                    break;
                case "test":
                    requestDatasetType = DatasetType.TEST;
                    break;
                default:
                    logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
                        datasetType);
            }
        }
    }
}
```

```
        throw new Exception("Unrecognized dataset type: " +
datasetType);
    }

    DatasetSource datasetSource =
DatasetSource.builder().datasetArn(existingDatasetArn).build();

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

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

    CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

    boolean created = false;

    //Wait until create finishes

    do {

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

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        DatasetStatus status = datasetDescription.status();

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

        switch (status) {

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

            case CREATE_IN_PROGRESS:
```

```
        Thread.sleep(5000);
        break;

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

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

    } while (created == false);

    return response.datasetArn();

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

}

public static void main(String[] args) {

    String datasetType = null;
    String datasetArn = null;
    String projectArn = null;
    String datasetSourceArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
        + "    project_arn - the ARN of the project that you want to add
copy the data to.\n\n"
```

```
        + "    dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
        + "    dataset_arn - the ARN of the dataset that you want to copy
from.\n\n";

    if (args.length != 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];
    datasetType = args[1];
    datasetSourceArn = args[2];

    try {

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

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

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

        rekClient.close();

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

}
```

```
}
```

## Menjelaskan kumpulan data (SDK)

Anda dapat menggunakan DescribeDataset API untuk mendapatkan informasi tentang kumpulan data.

Untuk mendeskripsikan kumpulan data (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode contoh berikut untuk menggambarkan kumpulan data.

### AWS CLI

Ubah nilai `dataset-arn` ke ARN dari kumpulan data yang ingin Anda gambarkan.

```
aws rekognition describe-dataset --dataset-arn dataset_arn \  
--profile custom-labels-access
```

### Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `dataset_arn` — ARN dari dataset yang ingin Anda gambarkan.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to describe an Amazon Rekognition Custom Labels dataset.  
"""  
  
import argparse  
import logging  
import boto3  
  
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

def describe_dataset(rek_client, dataset_arn):
    """
    Describes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to describe.
    """

    try:
        # Describe the dataset
        logger.info("Describing dataset %s", dataset_arn)

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        description = dataset['DatasetDescription']

        print(f"Created: {str(description['CreationTimestamp'])}")
        print(f"Updated: {str(description['LastUpdatedTimestamp'])}")
        print(f"Status: {description['Status']}")
        print(f"Status message: {description['StatusMessage']}")
        print(f"Status code: {description['StatusMessageCode']}")
        print("Stats:")
        print(
            f"\tLabeled entries: {description['DatasetStats']
            ['LabeledEntries']}")
        print(
            f"\tTotal entries: {description['DatasetStats']['TotalEntries']}")
        print(f"\tTotal labels: {description['DatasetStats']['TotalLabels']}")

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

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

```
parser.add_argument(
    "dataset_arn", help="The ARN of the dataset that you want to describe."
)

def main():

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

    try:

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

        print(f"Describing dataset {args.dataset_arn}")

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

        describe_dataset(rekognition_client, args.dataset_arn)

        print(f"Finished describing dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message=f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)

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

## Java V2

- `dataset_arn` — ARN dari dataset yang ingin Anda gambarkan.

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStats;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class DescribeDataset {

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

    public static void describeMyDataset(RekognitionClient rekClient, String
datasetArn) {

        try {

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

            DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();
            DatasetStats datasetStats = datasetDescription.datasetStats();

            System.out.println("ARN: " + datasetArn);
```

```
        System.out.println("Created: " +
datasetDescription.creationTimestamp().toString());
        System.out.println("Updated: " +
datasetDescription.lastUpdatedTimestamp().toString());
        System.out.println("Status: " +
datasetDescription.statusAsString());
        System.out.println("Message: " +
datasetDescription.statusMessage());
        System.out.println("Total Labels: " +
datasetStats.totalLabels().toString());
        System.out.println("Total entries: " +
datasetStats.totalEntries().toString());
        System.out.println("Entries with labels: " +
datasetStats.labeledEntries().toString());
        System.out.println("Entries with at least 1 error: " +
datasetStats.errorEntries().toString());

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

}

public static void main(String[] args) {

    final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
        + "    dataset_arn - The ARN of the dataset that you want to
describe.\n\n";

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

    String datasetArn = args[0];

    try {

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

```
        .region(Region.US_WEST_2)
        .build();

        // Describe the dataset.
        describeMyDataset(rekClient, datasetArn);

        rekClient.close();

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

## Daftar entri kumpulan data (SDK)

Anda dapat menggunakan `ListDatasetEntries` API untuk membuat daftar baris JSON untuk setiap gambar dalam kumpulan data. Untuk informasi selengkapnya, lihat [Membuat file manifes](#).

Untuk mencantumkan entri kumpulan data (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan contoh kode berikut daftar entri dalam dataset

### AWS CLI

Ubah nilai `dataset-arn` ke ARN dari kumpulan data yang ingin Anda daftarkan.

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \  
--profile custom-labels-access
```

Untuk mencantumkan hanya baris JSON dengan kesalahan, tentukan `has-errors`.

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \  
--has-errors \  

```

```
--profile custom-labels-access
```

## Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `dataset_arn` — ARN dari dataset yang ingin Anda daftar.
- `show_errors_only` — tentukan `true` jika Anda ingin melihat kesalahan saja. `false` jika tidak.

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

"""
Purpose
Shows how to list the entries in an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def list_dataset_entries(rek_client, dataset_arn, show_errors):
    """
    Lists the entries in an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataet that you want to use.
    """

    try:
        # List the entries.
        logger.info("Listing dataset entries for the dataset %s.", dataset_arn)

        finished = False
        count = 0
        next_token = ""
        show_errors_only = False
```

```
if show_errors.lower() == "true":
    show_errors_only = True

while finished is False:

    response = rek_client.list_dataset_entries(
        DatasetArn=dataset_arn,
        HasErrors=show_errors_only,
        MaxResults=100,
        NextToken=next_token)

    count += len(response['DatasetEntries'])

    for entry in response['DatasetEntries']:
        print(entry)

    if 'NextToken' not in response:
        finished = True
        logger.info("No more entries. Total:%s", count)
    else:
        next_token = next_token = response['NextToken']
        logger.info("Getting more entries. Total so far :%s", count)

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

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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to list."
    )

    parser.add_argument(
```

```
        "show_errors_only", help="true if you want to see errors only. false
otherwise."
    )

def main():

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

    try:

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

        print(f"Listing entries for dataset {args.dataset_arn}")

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

        list_dataset_entries(rekognition_client,
                             args.dataset_arn,
                             args.show_errors_only)

        print(f"Finished listing entries for dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `dataset_arn` — ARN dari dataset yang ingin Anda daftar.
- `show_errors_only` — tentukan `true` jika Anda ingin melihat kesalahan saja. `false` jika tidak.

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.ListDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.paginators.ListDatasetEntriesIterable;

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

public class ListDatasetEntries {

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

    public static void listMyDatasetEntries(RekognitionClient rekClient, String
        datasetArn, boolean showErrorsOnly)
        throws Exception, RekognitionException {

        try {
```

```
        logger.log(Level.INFO, "Listing dataset {0}", new Object[]
{ datasetArn });

        ListDatasetEntriesRequest listDatasetEntriesRequest =
ListDatasetEntriesRequest.builder()

.hasErrors(showErrorsOnly).datasetArn(datasetArn).maxResults(1).build();

        ListDatasetEntriesIterable datasetEntriesList = rekClient
                .listDatasetEntriesPaginator(listDatasetEntriesRequest);

        datasetEntriesList.stream().flatMap(r ->
r.datasetEntries().stream())
                .forEach(datasetEntry ->
System.out.println(datasetEntry.toString()));

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

}

public static void main(String args[]) {

    boolean showErrorsOnly = false;
    String datasetArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
<updates_file>\n\n" + "Where:\n"
        + "    dataset_arn - the ARN of the dataset that you want to
update.\n\n"
        + "    show_errors_only - true to show only errors. false
otherwise.\n\n";

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }

    datasetArn = args[0];
    if (args[1].toLowerCase().equals("true")) {
```

```
        showErrorsOnly = true;
    }

    try {

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

        // list the dataset entries.

        listMyDatasetEntries(rekClient, datasetArn, showErrorsOnly);

        System.out.println(String.format("Finished listing entries for :
%s", datasetArn));

        rekClient.close();

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

}

}
```

## Mendistribusikan dataset pelatihan (SDK)

Label Kustom Rekognition Amazon memerlukan kumpulan data pelatihan dan kumpulan data pengujian untuk melatih model Anda.

Jika Anda menggunakan API, Anda dapat menggunakan [DistributeDatasetEntries](#) API untuk mendistribusikan 20% kumpulan data pelatihan ke dalam kumpulan data pengujian kosong. Mendistribusikan kumpulan data pelatihan dapat berguna jika Anda hanya memiliki satu file manifes

yang tersedia. Gunakan file manifes tunggal untuk membuat kumpulan data pelatihan Anda. Kemudian buat kumpulan data pengujian kosong dan gunakan `DistributeDatasetEntries` untuk mengisi kumpulan data pengujian.

### Note

Jika Anda menggunakan konsol Label Kustom Amazon Rekognition dan memulai dengan satu proyek kumpulan data, Label Kustom Rekognition Amazon membagi (mendistribusikan) kumpulan data pelatihan, selama pelatihan, untuk membuat kumpulan data pengujian. 20% entri kumpulan data pelatihan dipindahkan ke kumpulan data pengujian.

Untuk mendistribusikan dataset pelatihan (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Buat proyek. Untuk informasi selengkapnya, lihat [Membuat proyek Amazon Rekognition Custom Labels \(SDK\)](#).
3. Buat dataset pelatihan Anda. Untuk informasi tentang kumpulan data, lihat [Membuat kumpulan data pelatihan dan pengujian](#).
4. Buat dataset pengujian kosong.
5. Gunakan kode contoh berikut untuk mendistribusikan 20% entri kumpulan data pelatihan ke dalam kumpulan data pengujian. Anda bisa mendapatkan Amazon Resource Names (ARN) untuk kumpulan data proyek dengan menelepon [DescribeProjects](#). Untuk kode sampel, lihat [Menjelaskan proyek \(SDK\)](#).

### AWS CLI

Ubah nilai `training_dataset-arn` dan `test_dataset_arn` dengan ARNS dari kumpulan data yang ingin Anda gunakan.

```
aws rekognition distribute-dataset-entries --datasets [{"Arn":  
  "training_dataset_arn"}, {"Arn": "test_dataset_arn"}] \  
  --profile custom-labels-access
```

### Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `training_dataset_arn` — ARN dari kumpulan data pelatihan tempat Anda mendistribusikan entri.
- `test_dataset_arn` — ARN dari kumpulan data pengujian tempat Anda mendistribusikan entri.

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

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def check_dataset_status(rek_client, dataset_arn):
    """
    Checks the current status of a dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The dataset that you want to check.
    :return: The dataset status and status message.
    """
    finished = False
    status = ""
    status_message = ""

    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

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

        if status == "UPDATE_IN_PROGRESS":

            logger.info("Distributing dataset: %s ", dataset_arn)
            time.sleep(5)
```

```
        continue

    if status == "UPDATE_COMPLETE":
        logger.info(
            "Dataset distribution complete: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        continue

    if status == "UPDATE_FAILED":
        logger.exception(
            "Dataset distribution failed: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        break

    logger.exception(
        "Failed. Unexpected state for dataset distribution: %s : %s : %s",
        status, status_message, dataset_arn)
    finished = True
    status_message = "An unexpected error occurred while distributing the
dataset"
    break

    return status, status_message

def distribute_dataset_entries(rek_client, training_dataset_arn,
test_dataset_arn):
    """
    Distributes 20% of the supplied training dataset into the supplied test
dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param training_dataset_arn: The ARN of the training dataset that you
distribute entries from.
    :param test_dataset_arn: The ARN of the test dataset that you distribute
entries to.
    """

    try:
        # List dataset labels.
        logger.info("Distributing training dataset entries (%s) into test
dataset (%s).",
                    training_dataset_arn, test_dataset_arn)
```

```
    datasets = json.loads(
        '[{"Arn" : "' + str(training_dataset_arn) + '"}, {"Arn" : "' +
str(test_dataset_arn) + '"}]')

    rek_client.distribute_dataset_entries(
        Datasets=datasets
    )

    training_dataset_status, training_dataset_status_message =
check_dataset_status(
    rek_client, training_dataset_arn)
    test_dataset_status, test_dataset_status_message = check_dataset_status(
        rek_client, test_dataset_arn)

    if training_dataset_status == 'UPDATE_COMPLETE' and test_dataset_status
== "UPDATE_COMPLETE":
        print("Distribution complete")
    else:
        print("Distribution failed:")
        print(
            f"\ttraining dataset: {training_dataset_status} :
{training_dataset_status_message}")
        print(
            f"\ttest dataset: {test_dataset_status} :
{test_dataset_status_message}")

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

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

    parser.add_argument(
        "training_dataset_arn", help="The ARN of the training dataset that you
want to distribute from."
    )
```

```
    parser.add_argument(
        "test_dataset_arn", help="The ARN of the test dataset that you want to
distribute to."
    )

def main():

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

    try:

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

        print(
            f"Distributing training dataset entries
({args.training_dataset_arn}) "\
            f"into test dataset ({args.test_dataset_arn}).")

        # Distribute the datasets.

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

        distribute_dataset_entries(rekognition_client,
                                  args.training_dataset_arn,
                                  args.test_dataset_arn)

        print("Finished distributing datasets.")

    except ClientError as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem listing dataset labels: {err}")
    except Exception as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem distributing datasets: {err}")

if __name__ == "__main__":
```

```
main()
```

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `training_dataset_arn` — ARN dari kumpulan data pelatihan tempat Anda mendistribusikan entri.
- `test_dataset_arn` — ARN dari kumpulan data pengujian tempat Anda mendistribusikan entri.

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DistributeDataset;
import
    software.amazon.awssdk.services.rekognition.model.DistributeDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class DistributeDatasetEntries {

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

    public static DatasetStatus checkDatasetStatus(RekognitionClient rekClient,
        String datasetArn)
```

```
throws Exception, RekognitionException {

    boolean distributed = false;
    DatasetStatus status = null;

    // Wait until distribution completes

    do {

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

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        status = datasetDescription.status();

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

        switch (status) {

            case UPDATE_COMPLETE:
                logger.log(Level.INFO, "Dataset updated");
                distributed = true;
                break;

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

            case UPDATE_FAILED:
                String error = "Dataset distribution failed: " +
datasetDescription.statusAsString() + " "
                    + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, error);
                break;

            default:
                String unexpectedError = "Unexpected distribution state: " +
datasetDescription.statusAsString() + " "
                    + datasetDescription.statusMessage() + " " + datasetArn;
```

```
        logger.log(Level.SEVERE, unexpectedError);

    }

} while (distributed == false);

return status;

}

public static void distributeMyDatasetEntries(RekognitionClient rekClient,
String trainingDatasetArn,
    String testDatasetArn) throws Exception, RekognitionException {

    try {

        logger.log(Level.INFO, "Distributing {0} dataset to {1} ",
            new Object[] { trainingDatasetArn, testDatasetArn });

        DistributeDataset distributeTrainingDataset =
DistributeDataset.builder().arn(trainingDatasetArn).build();

        DistributeDataset distributeTestDataset =
DistributeDataset.builder().arn(testDatasetArn).build();

        ArrayList<DistributeDataset> datasets = new ArrayList();

        datasets.add(distributeTrainingDataset);
        datasets.add(distributeTestDataset);

        DistributeDatasetEntriesRequest distributeDatasetEntriesRequest =
DistributeDatasetEntriesRequest.builder()
            .datasets(datasets).build();

        rekClient.distributeDatasetEntries(distributeDatasetEntriesRequest);

        DatasetStatus trainingStatus = checkDatasetStatus(rekClient,
trainingDatasetArn);
        DatasetStatus testStatus = checkDatasetStatus(rekClient,
testDatasetArn);

        if (trainingStatus == DatasetStatus.UPDATE_COMPLETE && testStatus ==
DatasetStatus.UPDATE_COMPLETE) {
```

```
        logger.log(Level.INFO, "Successfully distributed dataset: {0}",
trainingDatasetArn);

        } else {

            throw new Exception("Failed to distribute dataset: " +
trainingDatasetArn);
        }

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

}

public static void main(String[] args) {

    String trainingDatasetArn = null;
    String testDatasetArn = null;

    final String USAGE = "\n" + "Usage: " + "<training_dataset_arn>
<test_dataset_arn>\n\n" + "Where:\n"
        + "    training_dataset_arn - the ARN of the dataset that you
want to distribute from.\n\n"
        + "    test_dataset_arn - the ARN of the dataset that you want to
distribute to.\n\n";

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }

    trainingDatasetArn = args[0];
    testDatasetArn = args[1];

    try {

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

```
        .build();

        // Distribute the dataset
        distributeMyDatasetEntries(rekClient, trainingDatasetArn,
testDatasetArn);

        System.out.println("Datasets distributed.");

        rekClient.close();

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

## Menghapus dataset

Anda dapat menghapus kumpulan data pelatihan dan pengujian dari proyek.

### Topik

- [Menghapus kumpulan data \(Konsol\)](#)
- [Menghapus kumpulan data \(SDK\) Label Kustom Rekognition Amazon](#)

### Menghapus kumpulan data (Konsol)

Gunakan prosedur berikut untuk menghapus kumpulan data. Setelah itu, jika proyek memiliki satu set data yang tersisa (kereta atau tes), halaman detail proyek ditampilkan. Jika proyek tidak memiliki kumpulan data yang tersisa, halaman Buat dataset ditampilkan.

Jika Anda menghapus kumpulan data pelatihan, Anda harus membuat kumpulan data pelatihan baru untuk proyek tersebut sebelum Anda dapat melatih model. Untuk informasi selengkapnya, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#).

Jika Anda menghapus kumpulan data pengujian, Anda dapat melatih model tanpa membuat kumpulan data pengujian baru. Selama pelatihan, kumpulan data pelatihan dibagi untuk membuat kumpulan data pengujian baru untuk proyek tersebut. Memisahkan kumpulan data pelatihan mengurangi jumlah gambar yang tersedia untuk pelatihan. Untuk menjaga kualitas, kami sarankan untuk membuat kumpulan data pengujian baru sebelum melatih model. Untuk informasi selengkapnya, lihat [Menambahkan kumpulan data ke proyek](#).

Untuk menghapus dataset

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Di panel kiri, pilih Gunakan Label Kustom. Halaman landing Label Kustom Rekognition Amazon ditampilkan.
3. Di panel navigasi kiri, pilih Proyek. Tampilan Proyek ditampilkan.
4. Pilih proyek yang berisi kumpulan data yang ingin Anda hapus.
5. Di panel navigasi kiri, di bawah nama proyek, pilih Dataset
6. Pilih Tindakan
7. Untuk menghapus kumpulan data pelatihan, pilih Hapus kumpulan data pelatihan.
8. Untuk menghapus kumpulan data pengujian, pilih Hapus kumpulan data pengujian.
9. Dalam kotak dialog Delete train atau test dataset, masukkan delete untuk mengonfirmasi bahwa Anda ingin menghapus dataset.
10. Pilih Hapus kumpulan data kereta atau uji untuk menghapus kumpulan data.

## Menghapus kumpulan data (SDK) Label Kustom Rekognition Amazon

Anda menghapus kumpulan data Label Kustom Rekognition Amazon dengan memanggil [DeleteDataset](#) dan memasok Nama Sumber Daya Amazon (ARN) dari kumpulan data yang ingin Anda hapus. Untuk mendapatkan kumpulan data pelatihan dan pengujian dalam sebuah proyek, hubungi ARNs [DescribeProjects](#) Responsnya mencakup array [ProjectDescription](#) objek. Dataset ARNs (`DatasetArn`) dan tipe dataset (`DatasetType`) ada dalam daftar. `Datasets`

Jika Anda menghapus kumpulan data pelatihan, Anda perlu membuat kumpulan data pelatihan baru untuk proyek tersebut sebelum Anda dapat melatih model. Jika Anda menghapus kumpulan data pengujian, Anda perlu membuat kumpulan data pengujian baru sebelum dapat melatih modelnya. Untuk informasi selengkapnya, lihat [Menambahkan kumpulan data ke proyek \(SDK\)](#).

## Untuk menghapus dataset (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk menghapus kumpulan data.

### AWS CLI

Ubah nilai `dataset-arn` dengan ARN dari kumpulan data yang ingin Anda hapus.

```
aws rekognition delete-dataset --dataset-arn dataset-arn \  
--profile custom-labels-access
```

### Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `dataset_arn` — ARN dari dataset yang ingin Anda hapus.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to delete an Amazon Rekognition Custom Labels dataset.  
"""  
  
import argparse  
import logging  
import time  
import boto3  
  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def delete_dataset(rek_client, dataset_arn):  
    """  
    Deletes an Amazon Rekognition Custom Labels dataset.  
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.  
    :param dataset_arn: The ARN of the dataset that you want to delete.  
    """
```

```
"""

try:
    # Delete the dataset,
    logger.info("Deleting dataset: %s", dataset_arn)

    rek_client.delete_dataset(DatasetArn=dataset_arn)

    deleted = False

    logger.info("waiting for dataset deletion %s", dataset_arn)

    # Dataset might not be deleted yet, so wait.
    while deleted is False:
        try:
            rek_client.describe_dataset(DatasetArn=dataset_arn)
            time.sleep(5)
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.info("dataset deleted: %s", dataset_arn)
                deleted = True
            else:
                raise

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

    return True

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

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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to delete."
    )
```

```
def main():

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

    try:

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

        print(f"Deleting dataset: {args.dataset_arn}")

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

        delete_dataset(rekognition_client,
                       args.dataset_arn)

        print(f"Finished deleting dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem deleting dataset: {err}"
        logger.exception(error_message)
        print(error_message)

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `dataset_arn` — ARN dari dataset yang ingin Anda hapus.

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

```
*/
package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteDataset {

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

    public static void deleteMyDataset(RekognitionClient rekClient, String
datasetArn) throws InterruptedException {

        try {

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

            // Delete the dataset

            DeleteDatasetRequest deleteDatasetRequest =
DeleteDatasetRequest.builder().datasetArn(datasetArn).build();

            DeleteDatasetResponse response =
rekClient.deleteDataset(deleteDatasetRequest);

            // Wait until deletion finishes

            DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
                .build();

            Boolean deleted = false;

            do {
```

```
        try {
            rekClient.describeDataset(describeDatasetRequest);
            Thread.sleep(5000);
        } catch (RekognitionException e) {
            String errorCode = e.awsErrorDetails().errorCode();
            if (errorCode.equals("ResourceNotFoundException")) {
                logger.log(Level.INFO, "Dataset deleted: {0}",
datasetArn);
                deleted = true;
            } else {
                logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                throw e;
            }
        }

        } while (Boolean.FALSE.equals(deleted));

        logger.log(Level.INFO, "Dataset deleted: {0} ", datasetArn);

    } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
            + "    dataset_arn - The ARN of the dataset that you want to
delete.\n\n";

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

        String datasetArn = args[0];
```

```
try {

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

    // Delete the dataset
    deleteMyDataset(rekClient, datasetArn);

    System.out.println(String.format("Dataset deleted: %s",
datasetArn));

    rekClient.close();

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

catch (InterruptedException intError) {
    logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
    System.exit(1);
}

}

}
```

## Mengelola model Label Kustom Rekognition Amazon

Model Amazon Rekognition Custom Labels adalah model matematika yang memprediksi keberadaan objek, adegan, dan konsep dalam gambar baru. Ini dilakukan dengan menemukan pola dalam gambar yang digunakan untuk melatih model. Bagian ini menunjukkan cara melatih model, mengevaluasi kinerjanya, dan melakukan perbaikan. Ini juga menunjukkan kepada Anda bagaimana

membuat model tersedia untuk digunakan dan cara menghapus model saat Anda tidak lagi membutuhkannya.

## Topik

- [Menghapus model Label Kustom Rekognition Amazon](#)
- [Menandai model](#)
- [Menjelaskan model \(SDK\)](#)
- [Menyalin model Amazon Rekognition Custom Labels \(SDK\)](#)

## Menghapus model Label Kustom Rekognition Amazon

Anda dapat menghapus model dengan menggunakan konsol Amazon Rekognition Custom Labels atau dengan menggunakan API. [DeleteProjectVersion](#) Anda tidak dapat menghapus model jika model sedang berjalan atau jika model sedang dalam pelatihan. Untuk menghentikan model yang sedang berjalan, gunakan [StopProjectVersion](#) API. Untuk informasi selengkapnya, lihat [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#). Jika model sedang dilatih, tunggu hingga selesai sebelum Anda menghapus model.

Model yang dihapus tidak dapat dihapus.

## Topik

- [Menghapus model Label Kustom Rekognition Amazon \(Konsol\)](#)
- [Menghapus model Label Kustom Rekognition Amazon \(SDK\)](#)


## Menghapus model Label Kustom Rekognition Amazon (Konsol)

Prosedur berikut menunjukkan cara menghapus model dari halaman detail proyek. Anda juga dapat menghapus model dari halaman detail model.

Untuk menghapus model (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Gunakan Label Kustom.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Pilih proyek yang berisi model yang ingin Anda hapus. Halaman detail proyek terbuka.

6. Di bagian Model, pilih model yang ingin Anda hapus.


 Note

Jika model tidak dapat dipilih, model sedang berjalan atau sedang dilatih, dan tidak dapat dihapus. Periksa bidang Status dan coba lagi setelah menghentikan model yang sedang berjalan, atau tunggu hingga pelatihan selesai.

7. Pilih Hapus model dan kotak dialog Delete model ditampilkan.

8. Masukkan hapus untuk mengonfirmasi penghapusan.

9. Pilih Hapus untuk menghapus model. Menghapus model mungkin perlu beberapa saat untuk diselesaikan.

 Note

Jika Anda Menutup kotak dialog selama penghapusan model, model masih dihapus.

## Menghapus model Label Kustom Rekognition Amazon (SDK)

Anda menghapus model Label Kustom Rekognition Amazon dengan [DeleteProjectVersion](#) memanggil dan memasok Nama Sumber Daya Amazon (ARN) model yang ingin Anda hapus. Anda bisa mendapatkan ARN model dari bagian Use your model pada halaman detail model di konsol Amazon Rekognition Custom Labels. Atau, hubungi [DescribeProjectVersions](#) dan berikan yang berikut ini.

- ARN proyek (`ProjectArn`) yang terkait dengan model tersebut.
- Nama versi (`VersionNames`) dari model.

Model ARN adalah `ProjectVersionArn` bidang dalam [ProjectVersionDescription](#) objek, dari respons. `DescribeProjectVersions`

Anda tidak dapat menghapus model jika sedang berjalan atau sedang dilatih. Untuk menentukan apakah model sedang berjalan atau pelatihan, panggil [DescribeProjectVersions](#) dan periksa Status bidang [ProjectVersionDescription](#) objek model. Untuk menghentikan model yang sedang berjalan, gunakan [StopProjectVersion](#) API. Untuk informasi selengkapnya, lihat [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#). Anda harus menunggu model untuk menyelesaikan pelatihan sebelum Anda dapat menghapusnya.

## Untuk menghapus model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk menghapus model.

### AWS CLI

Ubah nilai `project-version-arn` menjadi nama proyek yang ingin Anda hapus.

```
aws rekognition delete-project-version --project-version-arn model_arn \  
--profile custom-labels-access
```

### Python

Menyediakan parameter baris perintah berikut

- `project_arn`— ARN dari proyek yang berisi model yang ingin Anda hapus.
- `model_arn`— ARN dari versi model yang ingin Anda hapus.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to delete an existing Amazon Rekognition Custom Labels model.  
"""  
  
import argparse  
import logging  
import time  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def find_forward_slash(input_string, n):  
    """  
    Returns the location of '/' after n number of occurrences.  
    """
```

```
:param input_string: The string you want to search
: n: the occurrence that you want to find.
"""
position = input_string.find('/')
while position >= 0 and n > 1:
    position = input_string.find('/', position + 1)
    n -= 1
return position

def delete_model(rek_client, project_arn, model_arn):
    """
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """

    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]

        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)

        logger.info("model deleted: %s", model_arn)
```

```
        return True

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

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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
you want to delete."
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
delete."
    )

def confirm_model_deletion(model_arn):
    """
    Confirms deletion of the model. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

    start = input("Enter delete to delete your model: ")
    if start == "delete":
        return True
    else:
        return False

def main():

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

```
try:

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

    if confirm_model_deletion(args.model_arn) is True:
        print(f"Deleting model: {args.model_arn}")

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

        delete_model(rekognition_client,
                     args.project_arn,
                     args.model_arn)

        print(f"Finished deleting model: {args.model_arn}")
    else:
        print(f"Not deleting model {args.model_arn}")

except ClientError as err:
    print(f"Problem deleting model: {err}")

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

## Java V2

- `project_arn`— ARN dari proyek yang berisi model yang ingin Anda hapus.
- `model_arn`— ARN dari versi model yang ingin Anda hapus.

```
//Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;

import
    software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteModel {

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

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;
    }

    public static void deleteMyModel(RekognitionClient rekClient, String
projectArn, String modelArn)
        throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting model: {0}", projectArn);

            // Delete the model

            DeleteProjectVersionRequest deleteProjectVersionRequest =
DeleteProjectVersionRequest.builder()
                .projectVersionArn(modelArn).build();

            DeleteProjectVersionResponse response =
```

```
        rekClient.deleteProjectVersion(deleteProjectVersionRequest);

        logger.log(Level.INFO, "Status: {0}", response.status());

        // Get the model version

        int start = findForwardSlash(modelArn, 3) + 1;
        int end = findForwardSlash(modelArn, 4);

        String versionName = modelArn.substring(start, end);

        Boolean deleted = false;

        DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
        .projectArn(projectArn).versionNames(versionName).build();

        // Wait until model is deleted.

        do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

        .describeProjectVersions(describeProjectVersionsRequest);

                if
(describeProjectVersionsResponse.projectVersionDescriptions().size()==0) {
                logger.log(Level.INFO, "Waiting for model deletion: {0}",
modelArn);

                Thread.sleep(5000);
                } else {
                deleted = true;
                logger.log(Level.INFO, "Model deleted: {0}", modelArn);
                }

        } while (Boolean.FALSE.equals(deleted));

        logger.log(Level.INFO, "Model deleted: {0}", modelArn);

    } catch (

        RekognitionException e) {
```

```
        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: " + "<project_arn> <model_arn>\n\n"
+ "Where:\n"
        + "    project_arn - The ARN of the project that contains the
model that you want to delete.\n\n"
        + "    model_version - The ARN of the model that you want to
delete.\n\n";

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String projectArn = args[0];
    String modelVersion = args[1];

    try {

        RekognitionClient rekClient = RekognitionClient.builder().build();

        // Delete the model
        deleteMyModel(rekClient, projectArn, modelVersion);

        System.out.println(String.format("model deleted: %s",
modelVersion));

        rekClient.close();

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

    catch (InterruptedException intError) {
```

```
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
        System.exit(1);
    }
}
}
```

## Menandai model

Anda dapat mengidentifikasi, mengatur, mencari, dan memfilter model Rekognition Amazon Anda dengan menggunakan tag. Setiap tag adalah label yang terdiri dari kunci dan nilai yang ditentukan pengguna. Misalnya, untuk membantu menentukan penagihan untuk model Anda, beri tag model Anda dengan `Cost center` kunci dan tambahkan nomor pusat biaya yang sesuai sebagai nilai. Untuk informasi selengkapnya, lihat [Menandai sumber daya AWS](#).

Gunakan tag untuk:

- Lacak penagihan untuk model dengan menggunakan tag alokasi biaya. Untuk informasi selengkapnya, lihat [Menggunakan Tag Alokasi Biaya](#).
- Kontrol akses ke model dengan menggunakan Identity and Access Management (IAM) Untuk informasi selengkapnya, lihat [Mengontrol akses ke sumber daya AWS menggunakan tanda sumber daya](#).
- Mengotomatiskan manajemen model. Misalnya, Anda dapat menjalankan skrip start atau stop otomatis yang mematikan model pengembangan selama jam non-bisnis untuk mengurangi biaya. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

Anda dapat menandai model dengan menggunakan konsol Amazon Rekognition atau dengan menggunakan AWS SDKs

Topik

- [Model penandaan \(konsol\)](#)
- [Melihat tag model](#)
- [Model penandaan \(SDK\)](#)

## Model penandaan (konsol)

Anda dapat menggunakan konsol Rekognition untuk menambahkan tag ke model, melihat tag yang dilampirkan ke model, dan menghapus tag.

### Menambah atau menghapus tag

Prosedur ini menjelaskan cara menambahkan tag ke, atau menghapus tag dari, model yang ada. Anda juga dapat menambahkan tag ke model baru saat dilatih. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#).

Untuk menambahkan tag ke, atau menghapus tag dari, model yang ada menggunakan konsol

1. Buka konsol Amazon Rekognition di. <https://console.aws.amazon.com/rekognition/>
2. Pilih Mulai.
3. Di panel navigasi, pilih Proyek.
4. Pada halaman Sumber daya proyek, pilih proyek yang berisi model yang ingin Anda tag.
5. Di panel navigasi, di bawah proyek yang Anda pilih sebelumnya, pilih Model.
6. Di bagian Model, pilih model yang ingin Anda tambahkan tag.
7. Pada halaman detail model, pilih tab Tag.
8. Di bagian Tag, pilih Kelola tag.
9. Pada halaman Kelola tag, pilih Tambahkan tag baru.
10. Masukkan kunci dan nilai.
  - a. Untuk Key, masukkan nama untuk kunci tersebut.
  - b. Untuk Nilai, masukkan nilai.
11. Untuk menambahkan lebih banyak tag, ulangi langkah 9 dan 10.
12. (Opsional) Untuk menghapus tag, pilih Hapus di samping tag yang ingin Anda hapus. Jika Anda menghapus tag yang disimpan sebelumnya, tag tersebut akan dihapus saat Anda menyimpan perubahan.
13. Pilih Simpan perubahan untuk menyimpan perubahan Anda.

### Melihat tag model

Anda dapat menggunakan konsol Amazon Rekognition untuk melihat tag yang dilampirkan ke model.

Untuk melihat tag yang dilampirkan ke semua model dalam proyek, Anda harus menggunakan AWS SDK. Untuk informasi selengkapnya, lihat [Daftar tag model](#).

Untuk melihat tag yang dilampirkan ke model

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>
2. Pilih Mulai.
3. Di panel navigasi, pilih Proyek.
4. Pada halaman Sumber daya proyek, pilih proyek yang berisi model yang tagnya ingin Anda lihat.
5. Di panel navigasi, di bawah proyek yang Anda pilih sebelumnya, pilih Model.
6. Di bagian Model, pilih model yang tagnya ingin Anda lihat.
7. Pada halaman detail model, pilih tab Tag. Tag ditampilkan di bagian Tag.

## Model penandaan (SDK)

Anda dapat menggunakan AWS SDK untuk:

- Tambahkan tag ke model baru
- Tambahkan tag ke model yang ada
- Buat daftar tag yang dilampirkan pada model
- Hapus tag dari model

Tag dalam AWS CLI contoh berikut adalah dalam format berikut.

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

Atau, Anda dapat menggunakan format ini.

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

Jika Anda belum menginstal AWS CLI, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).

Menambahkan tag ke model baru

Anda dapat menambahkan tag ke model saat Anda membuatnya menggunakan [CreateProjectVersion](#) operasi. Tentukan satu tanda atau lebih dalam parameter input array Tags.

```
aws rekognition create-project-version --project-arn project_arn \  
  --version-name version_name \  
  --output-config '{ "S3Location": { "Bucket": "output_bucket", "Prefix": "output  
folder" } }' \  
  --tags '{"key1": "value1", "key2": "value2"}' \  
  --profile custom-labels-access
```

Untuk informasi tentang membuat dan melatih model, lihat [Melatih model \(SDK\)](#).

Menambahkan tag ke model yang ada

Untuk menambahkan satu atau beberapa tag ke model yang ada, gunakan [TagResource](#) operasi. Tentukan Amazon Resource Name (ARN) (`ResourceArn`) model dan tag (`Tags`) yang ingin Anda tambahkan. Contoh berikut menunjukkan cara menambahkan dua tanda.

```
aws rekognition tag-resource --resource-arn resource-arn \  
  --tags '{"key1": "value1", "key2": "value2"}' \  
  --profile custom-labels-access
```

Anda bisa mendapatkan ARN untuk model dengan menelepon. [CreateProjectVersion](#)

Daftar tag model

Untuk membuat daftar tag yang dilampirkan pada model, gunakan [ListTagsForResource](#) operasi dan tentukan ARN model (`ResourceArn`). Responsnya adalah peta kunci tag dan nilai yang dilampirkan ke model yang ditentukan.

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

Output menampilkan daftar tag yang dilampirkan pada model.

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

Untuk melihat model mana dalam proyek yang memiliki tag tertentu, hubungi `DescribeProjectVersions` untuk mendapatkan daftar model. Kemudian panggil `ListTagsForResource` setiap model dalam tanggapan dari `DescribeProjectVersions`. Periksa respons dari `ListTagsForResource` untuk melihat apakah tag yang diperlukan ada.

Contoh Python 3 berikut menunjukkan kepada Anda bagaimana mencari semua proyek Anda untuk kunci dan nilai tag tertentu. Outputnya mencakup proyek ARN dan model ARN di mana kunci yang cocok ditemukan.

Untuk mencari nilai tag

1. Simpan kode berikut ke file bernama `find_tag.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""
import logging
import argparse
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_tag_in_projects(rekognition_client, key, value):
    """
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
    key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False
```

```
projects = rekognition_client.describe_projects()
# Iterate through each project and models within a project.
for project in projects["ProjectDescriptions"]:
    logger.info("Searching project: %s ...", project["ProjectArn"])

    models = rekognition_client.describe_project_versions(
        ProjectArn=(project["ProjectArn"])
    )

    for model in models["ProjectVersionDescriptions"]:
        logger.info("Searching model %s", model["ProjectVersionArn"])

        tags = rekognition_client.list_tags_for_resource(
            ResourceArn=model["ProjectVersionArn"]
        )

        logger.info(
            "\tSearching model: %s for tag: %s value: %s.",
            model["ProjectVersionArn"],
            key,
            value,
        )
        # Check if tag exists.

        if key in tags["Tags"]:
            if tags["Tags"][key] == value:
                found = True
                logger.info(
                    "\t\tMATCH: Project: %s: model version %s",
                    project["ProjectArn"],
                    model["ProjectVersionArn"],
                )
                found_tags.append(
                    {
                        "Project": project["ProjectArn"],
                        "ModelVersion": model["ProjectVersionArn"],
                    }
                )

        if found is False:
            logger.info("No match for Tag %s with value %s.", key, value)
    return found_tags
except ClientError as err:
```

```
        logger.info("Problem finding tags: %s. ", format(err))
        raise

def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")

    args = parser.parse_args()
    key = args.tag
    value = args.value

    print(f"Searching your models for tag: {key} with value: {value}.")

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

    # Get tagged models for all projects.
    tagged_models = find_tag_in_projects(rekognition_client, key, value)

    print("Matched models\n-----")
    if len(tagged_models) > 0:
        for model in tagged_models:
            print(
                "Project: {project}\nModel version: {version}\n".format(
                    project=model["Project"], version=model["ModelVersion"]
                )
            )
    else:
        print("No matches found.")

    print("Done.")
```

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

2. Di prompt perintah, masukkan perintah berikut. Ganti *key* dan *value* dengan nama kunci dan nilai kunci yang ingin Anda temukan.

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

## Menghapus tag dari model

Untuk menghapus satu atau lebih tag dari model, gunakan [UntagResource](#) operasi. Tentukan ARN dari model (ResourceArn) dan tombol tanda (Tag-Keys) yang ingin Anda hapus.

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

Atau, Anda dapat menentukan tag-keys dalam format ini.

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

## Menjelaskan model (SDK)

Anda dapat menggunakan DescribeProjectVersions API untuk mendapatkan informasi tentang versi model. Jika Anda tidak menentukan VersionName, DescribeProjectVersions mengembalikan deskripsi untuk semua versi model dalam proyek.

Untuk menggambarkan model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode contoh berikut untuk menggambarkan versi model.

### AWS CLI

Ubah nilai `project-arn` ke ARN proyek yang ingin Anda gambarkan. Ubah nilai `version-name` ke versi model yang ingin Anda gambarkan.

```
aws rekognition describe-project-versions --project-arn project_arn \  
--version-names version_name \  
--profile custom-labels-access
```

## Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn` — ARN dari model yang ingin Anda gambarkan.
- `model_version` — versi model yang ingin Anda gambarkan.

Misalnya: `python describe_model.py project_arn model_version`

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to describe an Amazon Rekognition Custom Labels model.  
"""  
  
import argparse  
import logging  
import boto3  
  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def describe_model(rek_client, project_arn, version_name):  
    """  
    Describes an Amazon Rekognition Custom Labels model.  
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.  
    :param project_arn: The ARN of the project that contains the model.  
    :param version_name: The version name of the model that you want to  
    describe.  
    """  
  
    try:  
        # Describe the model  
        logger.info("Describing model: %s for project %s",
```

```

        version_name, project_arn)

    describe_response =
    rek_client.describe_project_versions(ProjectArn=project_arn,

    VersionNames=[version_name])
    for model in describe_response['ProjectVersionDescriptions']:
        print(f"Created: {str(model['CreationTimestamp'])} ")
        print(f"ARN: {str(model['ProjectVersionArn'])} ")
        if 'BillableTrainingTimeInSeconds' in model:
            print(
                f"Billing training time (minutes):
    {str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
                print(
                    f"\tSummary location: s3://{evaluation_results['Summary']
    ['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

            if 'ManifestSummary' in model:
                print(
                    f"Manifest summary location: s3://{model['ManifestSummary']
    ['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
                if 'OutputConfig' in model:
                    print(
                        f"Training output location: s3://{model['OutputConfig']
    ['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
                    if 'MinInferenceUnits' in model:
                        print(
                            f"Minimum inference units:
    {str(model['MinInferenceUnits'])}")
                        if 'MaxInferenceUnits' in model:
                            print(
                                f"Maximum Inference units:
    {str(model['MaxInferenceUnits'])}")

                    print("Status: " + model['Status'])
                    print("Message: " + model['StatusMessage'])

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

```

```
        raise

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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
describe."
    )

def main():

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

    try:

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

        print(
            f"Describing model: {args.version_name} for project
{args.project_arn}.")

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

        describe_model(rekognition_client, args.project_arn,
                       args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
{args.project_arn}.")
```

```
except ClientError as err:
    error_message = f"Problem describing model: {err}"
    logger.exception(error_message)
    print(error_message)
except Exception as err:
    error_message = f"Problem describing model: {err}"
    logger.exception(error_message)
    print(error_message)

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn` — ARN dari model yang ingin Anda gambarkan.
- `model_version` — versi model yang ingin Anda gambarkan.

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.EvaluationResult;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

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

public class DescribeModel {

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

    public static void describeMyModel(RekognitionClient rekClient, String
    projectArn, String versionName) {

        try {

            // If a single version name is supplied, build request argument

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
            null;

            if (versionName == null) {
                describeProjectVersionsRequest =
                DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                .build();
            } else {
                describeProjectVersionsRequest =
                DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                .versionNames(versionName).build();
            }

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
            rekClient
                .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
            describeProjectVersionsResponse
                .projectVersionDescriptions()) {

                System.out.println("ARN: " +
                projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
                projectVersionDescription.statusAsString());
                System.out.println("Message: " +
                projectVersionDescription.statusMessage());
            }
        }
    }
}
```

```
        if (projectVersionDescription.billableTrainingTimeInSeconds() !=
null) {
            System.out.println(
                "Billable minutes: " +
(projectVersionDescription.billableTrainingTimeInSeconds() / 60));
        }

        if (projectVersionDescription.evaluationResult() != null) {
            EvaluationResult evaluationResult =
projectVersionDescription.evaluationResult();

            System.out.println("F1 Score: " +
evaluationResult.f1Score());
            System.out.println("Summary location: s3://" +
evaluationResult.summary().s3object().bucket() + "/"
                + evaluationResult.summary().s3object().name());
        }

        if (projectVersionDescription.manifestSummary() != null) {
            GroundTruthManifest manifestSummary =
projectVersionDescription.manifestSummary();
            System.out.println("Manifest summary location: s3://" +
manifestSummary.s3object().bucket() + "/"
                + manifestSummary.s3object().name());
        }

        if (projectVersionDescription.outputConfig() != null) {
            OutputConfig outputConfig =
projectVersionDescription.outputConfig();
            System.out.println(
                "Training output: s3://" + outputConfig.s3Bucket() +
"/" + outputConfig.s3KeyPrefix());
        }

        if (projectVersionDescription.minInferenceUnits() != null) {
            System.out.println("Min inference units: " +
projectVersionDescription.minInferenceUnits());
        }

        System.out.println();
    }
}
```

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

}

public static void main(String args[]) {

    String projectArn = null;
    String versionName = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <version_name>\n
\n" + "Where:\n"
        + "    project_arn - The ARN of the project that contains the
models you want to describe.\n\n"
        + "    version_name - (optional) The version name of the model
that you want to describe. \n\n"
        + "                                If you don't specify a value, all model
versions are described.\n\n";

    if (args.length > 2 || args.length == 0) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];

    if (args.length == 2) {
        versionName = args[1];
    }

    try {

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

        // Describe the model
        describeMyModel(rekClient, projectArn, versionName);
    }
}
```

```
        rekClient.close();

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

}

}
```

## Menyalin model Amazon Rekognition Custom Labels (SDK)

Anda dapat menggunakan [CopyProjectVersion](#) operasi untuk menyalin versi model Label Kustom Rekognition Amazon dari sumber proyek Amazon Rekognition Custom Labels ke proyek tujuan. Proyek tujuan dapat berada di AWS akun yang berbeda, atau di AWS akun yang sama. Skenario tipikal adalah menyalin model yang diuji dari AWS akun pengembangan ke AWS akun produksi.

Atau, Anda dapat melatih model di akun tujuan dengan kumpulan data sumber. Menggunakan [CopyProjectVersion](#) operasi memiliki keuntungan sebagai berikut.

- Perilaku model konsisten. Pelatihan model bersifat non-deterministik dan dua model yang dilatih dengan kumpulan data yang sama tidak dijamin untuk membuat prediksi yang sama. Menyalin model dengan [CopyProjectVersion](#) membantu memastikan bahwa perilaku model yang disalin konsisten dengan model sumber dan Anda tidak perlu menguji ulang model.
- Pelatihan model tidak diperlukan. Ini menghemat uang Anda karena Anda dikenakan biaya untuk setiap pelatihan model yang berhasil.

Untuk menyalin model ke AWS akun lain, Anda harus memiliki proyek Label Kustom Rekognition Amazon di akun tujuan. AWS Untuk informasi tentang membuat proyek, lihat [Membuat proyek](#). Pastikan untuk membuat proyek di AWS akun tujuan.

[Kebijakan proyek adalah kebijakan](#) berbasis sumber daya yang menetapkan izin penyalinan untuk versi model yang ingin Anda salin. Anda perlu menggunakan [kebijakan proyek](#) ketika proyek tujuan berada di AWS akun yang berbeda dari proyek sumber.

Anda tidak perlu menggunakan [kebijakan proyek](#), saat menyalin versi model dalam akun yang sama. Namun, Anda dapat memilih untuk menggunakan [kebijakan proyek](#) pada proyek antar-akun jika Anda ingin lebih banyak kontrol atas sumber daya ini.

Anda melampirkan kebijakan proyek ke proyek sumber dengan memanggil [PutProjectPolicy](#) operasi.

Anda tidak dapat menggunakan `CopyProjectVersion` untuk menyalin model ke proyek di AWS Wilayah yang berbeda. Selain itu, Anda tidak dapat menyalin model dengan konsol Label Kustom Rekognition Amazon. Dalam kasus ini, Anda dapat melatih model dalam proyek tujuan dengan kumpulan data yang digunakan untuk melatih model sumber. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#).

Untuk menyalin model dari proyek sumber ke proyek tujuan, lakukan hal berikut:

Untuk menyalin model

1. [Buat dokumen kebijakan proyek](#).
2. [Lampirkan kebijakan proyek ke proyek sumber](#).
3. [Salin model dengan CopyProjectVersion operasi](#).

Untuk menghapus kebijakan proyek dari proyek, hubungi [DeleteProjectPolicy](#). Untuk mendapatkan daftar kebijakan proyek yang dilampirkan pada proyek, hubungi [ListProjectPolicies](#).

Topik

- [Membuat dokumen kebijakan proyek](#)
- [Melampirkan kebijakan proyek \(SDK\)](#)
- [Menyalin model \(SDK\)](#)
- [Daftar kebijakan proyek \(SDK\)](#)
- [Menghapus kebijakan proyek \(SDK\)](#)

## Membuat dokumen kebijakan proyek

Rekognition Custom Labels menggunakan kebijakan berbasis sumber daya, yang dikenal sebagai kebijakan proyek, untuk mengelola izin penyalinan untuk versi model. Kebijakan proyek adalah dokumen format JSON.

Kebijakan proyek mengizinkan atau menolak izin [utama](#) untuk menyalin versi model dari proyek sumber ke proyek tujuan. Anda memerlukan kebijakan proyek jika proyek tujuan berada di AWS

akun yang berbeda. Itu juga benar jika proyek tujuan berada di AWS akun yang sama dengan proyek sumber dan Anda ingin membatasi akses ke versi model tertentu. Misalnya, Anda mungkin ingin menolak izin salin ke peran IAM tertentu dalam akun AWS .

Contoh berikut memungkinkan kepala sekolah `arn:aws:iam::111111111111:role/Admin` untuk menyalin versi model `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

## JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/Admin"
      },
      "Action": "rekognition:CopyProjectVersion",
      "Resource": "arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

### Note

Action, Resource, Principal, dan Effect merupakan bidang wajib dalam dokumen kebijakan proyek.

Satu-satunya yang didukung action adalah `rekognition:CopyProjectVersion`.

`NotAction`, `NotResource`, dan `NotPrincipal` merupakan bidang yang dilarang dan tidak boleh ada dalam dokumen kebijakan proyek.

Jika Anda tidak menentukan kebijakan proyek, prinsipal di AWS akun yang sama dengan proyek sumber masih dapat menyalin model, jika prinsipal memiliki kebijakan berbasis Identitas, seperti `AmazonRekognitionCustomLabelsFullAccess`, yang memberikan izin untuk menelepon `CopyProjectVersion`

Prosedur berikut membuat file dokumen kebijakan proyek yang dapat Anda gunakan dengan contoh Python di [Melampirkan kebijakan proyek \(SDK\)](#) Jika Anda menggunakan `put-project-policy` AWS CLI perintah, Anda menyediakan kebijakan proyek sebagai string JSON.

Untuk membuat dokumen kebijakan proyek

1. Dalam editor teks, buat dokumen berikut. Ubah nilai berikut:
  - Efek - Tentukan `ALLOW` untuk memberikan izin salinan. Tentukan `DENY` untuk menolak izin penyalinan.
  - Principal — Untuk prinsipal yang ingin Anda izinkan atau tolak akses ke versi model yang Anda tentukan `Resource`. Misalnya, Anda dapat menentukan [pokok akun AWS](#) untuk AWS akun yang berbeda. Kami tidak membatasi prinsip yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat [Menentukan prinsipal](#).
  - Sumber Daya - Nama Sumber Daya Amazon (ARN) dari versi model yang ingin Anda tentukan izin salin. Jika Anda ingin memberikan izin ke semua versi model dalam proyek sumber, gunakan format berikut `arn:aws:rekognition:region:account:project/source project/version/*`
2. Simpan kebijakan proyek ke komputer Anda.
3. Lampirkan kebijakan proyek ke proyek sumber dengan mengikuti instruksi di [Melampirkan kebijakan proyek \(SDK\)](#).

## Melampirkan kebijakan proyek (SDK)

Anda melampirkan kebijakan proyek ke proyek Label Kustom Rekognition Amazon dengan memanggil operasi [PutProjectPolicy](#)

Lampirkan beberapa kebijakan proyek ke proyek dengan memanggil `PutProjectPolicy` setiap kebijakan proyek yang ingin Anda tambahkan. Anda dapat melampirkan hingga lima kebijakan proyek ke proyek. Jika Anda perlu melampirkan lebih banyak kebijakan proyek, Anda dapat meminta kenaikan [batas](#).

Saat pertama kali melampirkan kebijakan proyek unik ke proyek, jangan tentukan ID revisi dalam parameter `PolicyRevisionId` input. Tanggapan dari `PutProjectPolicy` adalah ID revisi untuk kebijakan proyek yang dibuat Label Kustom Rekognition Amazon untuk Anda. Anda dapat menggunakan ID revisi untuk memperbarui atau menghapus revisi terbaru dari kebijakan proyek. Amazon Rekognition Custom Labels hanya menyimpan revisi terbaru dari kebijakan proyek.

Jika Anda mencoba memperbarui atau menghapus revisi kebijakan proyek sebelumnya, Anda mendapatkan `InvalidPolicyRevisionIdException` kesalahan.

Untuk memperbarui kebijakan proyek yang ada, tentukan ID revisi kebijakan proyek dalam parameter `PolicyRevisionId` input. Anda bisa mendapatkan revisi IDs untuk kebijakan proyek dalam proyek dengan menelepon [ListProjectPolicies](#).

Setelah Anda melampirkan kebijakan proyek ke proyek sumber, Anda dapat menyalin model dari proyek sumber ke proyek tujuan. Untuk informasi selengkapnya, lihat [Menyalin model \(SDK\)](#).

Untuk menghapus kebijakan proyek dari proyek, hubungi [DeleteProjectPolicy](#). Untuk mendapatkan daftar kebijakan proyek yang dilampirkan pada proyek, hubungi [ListProjectPolicies](#).

Untuk melampirkan kebijakan proyek ke proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. [Buat dokumen kebijakan proyek](#).
3. Gunakan kode berikut untuk melampirkan kebijakan proyek ke proyek, di AWS akun kepercayaan, yang berisi versi model yang ingin Anda salin. Untuk mendapatkan proyek ARN, hubungi [DescribeProjects](#) Untuk mendapatkan versi model panggilan ARN. [DescribeProjectVersions](#)

## AWS CLI

Ubah nilai berikut:

- `project-arn` ARN proyek sumber di AWS akun kepercayaan yang berisi versi model yang ingin Anda salin.
- `policy-name` untuk nama kebijakan yang Anda pilih.
- `principal` Untuk prinsipal yang ingin Anda izinkan atau tolak akses ke versi model yang Anda tentukan `Model version` ARN.
- `project-version-arn` ARN dari versi model yang ingin Anda salin.

Jika Anda ingin memperbarui kebijakan proyek yang ada, tentukan `policy-revision-id` parameter dan berikan ID revisi dari kebijakan proyek yang diinginkan.

```
aws rekognition put-project-policy \
```

```

--project-arn project-arn \
--policy-name policy-name \
--policy-document '{ "Version": "2012-10-17",          "Statement":
[{"Effect":"ALLOW or DENY", "Principal":{"AWS":"principal" },
  "Action":"rekognition:CopyProjectVersion", "Resource":"project-version-
arn" }]} ' \
--profile custom-labels-access

```

## Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek sumber yang ingin Anda lampirkan kebijakan proyek.
- `policy_name`— Nama kebijakan yang Anda pilih.
- `project_policy`— File yang berisi dokumen kebijakan proyek,.
- `policy_revision_id`— (Opsional). Jika Anda ingin memperbarui revisi kebijakan proyek yang ada, tentukan ID revisi kebijakan proyek.

```

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

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to attach a project policy to an Amazon Rekognition Custom Labels
project.
"""

import boto3
import argparse
import logging
import json
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

```

```
def put_project_policy(rek_client, project_arn, policy_name,
    policy_document_file, policy_revision_id=None):
    """
    Attaches a project policy to an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param policy_name: A name for the project policy.
    :param project_arn: The Amazon Resource Name (ARN) of the source project
    that you want to attach the project policy to.
    :param policy_document_file: The JSON project policy document to
    attach to the source project.
    :param policy_revision_id: (Optional) The revision of an existing policy to
    update.
    Pass None to attach new policy.
    :return The revision ID for the project policy.
    """

    try:

        policy_document_json = ""
        response = None

        with open(policy_document_file, 'r') as policy_document:
            policy_document_json = json.dumps(json.load(policy_document))

        logger.info(
            "Attaching %s project_policy to project %s.",
            policy_name, project_arn)

        if policy_revision_id is None:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                PolicyName=policy_name,
                PolicyDocument=policy_document_json)

        else:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                PolicyName=policy_name,
                PolicyDocument=policy_document_json,
                PolicyRevisionId=policy_revision_id)

            new_revision_id = response['PolicyRevisionId']
```

```
        logger.info(
            "Finished creating project policy %s. Revision ID: %s",
            policy_name, new_revision_id)

        return new_revision_id

    except ClientError as err:
        logger.exception(
            "Couldn't attach %s project policy to project %s: %s }",
            policy_name, project_arn, err.response['Error']['Message'] )
        raise

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

    parser.add_argument(
        "project_arn", help="The Amazon Resource Name (ARN) of the project "
        "that you want to attach the project policy to."
    )
    parser.add_argument(
        "policy_name", help="A name for the project policy."
    )

    parser.add_argument(
        "project_policy", help="The file containing the project policy JSON"
    )

    parser.add_argument(
        "--policy_revision_id", help="The revision of an existing policy to
        update. "
        "If you don't supply a value, a new project policy is created.",
        required=False
    )

def main():

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

```
try:

    # get command line arguments
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)

    args = parser.parse_args()

    print(f"Attaching policy to {args.project_arn}")

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

    # Attach a new policy or update an existing policy.

    response = put_project_policy(rekognition_client,
                                  args.project_arn,
                                  args.policy_name,
                                  args.project_policy,
                                  args.policy_revision_id)

    print(
        f"project policy {args.policy_name} attached to project
{args.project_arn}")
    print(f"Revision ID: {response}")

except ClientError as err:
    print("Problem attaching project policy: %s", err)

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn`— ARN dari proyek sumber yang ingin Anda lampirkan kebijakan proyek.
- `project_policy_name`— Nama kebijakan yang Anda pilih.
- `project_policy_document`— File yang berisi dokumen kebijakan proyek.

- `project_policy_revision_id`— (Opsional). Jika Anda ingin memperbarui revisi kebijakan proyek yang ada, tentukan ID revisi kebijakan proyek.

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

package com.example.rekognition;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.PutProjectPolicyRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class PutProjectPolicy {

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

    public static void putMyProjectPolicy(RekognitionClient rekClient, String
        projectArn, String projectPolicyName,
        String projectPolicyFileName, String projectPolicyRevisionId)
        throws IOException {

        try {

            Path filePath = Path.of(projectPolicyFileName);

            String policyDocument = Files.readString(filePath);
```

```
        String[] logArguments = new String[] { projectPolicyFileName,
projectPolicyName };

        PutProjectPolicyRequest putProjectPolicyRequest = null;

        logger.log(Level.INFO, "Attaching Project policy: {0} to project:
{1}", logArguments);

        // Attach the project policy.

        if (projectPolicyRevisionId == null) {
            putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)

.policyName(projectPolicyName).policyDocument(policyDocument).build();
        } else {
            putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)

.policyName(projectPolicyName).policyRevisionId(projectPolicyRevisionId)
                .policyDocument(policyDocument)

                .build();
        }

        rekClient.putProjectPolicy(putProjectPolicyRequest);

        logger.log(Level.INFO, "Attached Project policy: {0} to project:
{1}", logArguments);

    } catch (

        RekognitionException e) {
        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: "
```

```
        + "<project_arn> <project_policy_name> <policy_document>
<project_policy_revision_id>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that you want to
attach the project policy to.\n\n"
        + "    project_policy_name - A name for the project policy.\n\n"
        + "    project_policy_document - The file name of the project
policy.\n\n"
        + "    project_policy_revision_id - (Optional) The revision ID of
the project policy that you want to update.\n\n";

    if (args.length < 3 || args.length > 4) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String projectArn = args[0];
    String projectPolicyName = args[1];
    String projectPolicyDocument = args[2];
    String projectPolicyRevisionId = null;

    if (args.length == 4) {
        projectPolicyRevisionId = args[3];
    }

    try {

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

        // Attach the project policy.
        putMyProjectPolicy(rekClient, projectArn, projectPolicyName,
projectPolicyDocument,
            projectPolicyRevisionId);

        System.out.println(
            String.format("project policy %s: attached to project: %s",
projectPolicyName, projectArn));

        rekClient.close();
    }
}
```

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

    catch (IOException intError) {
        logger.log(Level.SEVERE, "Exception while reading policy document:
{0}", intError.getMessage());
        System.exit(1);
    }
}
}
```

4. Salin versi model dengan mengikuti instruksi di [Menyalin model \(SDK\)](#).

## Menyalin model (SDK)

Anda dapat menggunakan CopyProjectVersion API untuk menyalin versi model dari proyek sumber ke proyek tujuan. Proyek tujuan dapat berada di AWS akun yang berbeda tetapi harus AWS Wilayah yang sama. Jika proyek tujuan berada di AWS akun yang berbeda (atau jika Anda ingin memberikan izin khusus untuk versi model yang disalin dalam AWS akun), Anda harus melampirkan kebijakan proyek ke proyek sumber. Untuk informasi selengkapnya, lihat [Membuat dokumen kebijakan proyek](#). CopyProjectVersionAPI memerlukan akses ke bucket Amazon S3 Anda.

Model yang disalin mencakup hasil pelatihan untuk model sumber, tetapi tidak menyertakan kumpulan data sumber.

AWS Akun sumber tidak memiliki kepemilikan atas model yang disalin ke akun tujuan, kecuali jika Anda menyiapkan izin yang sesuai.

### Untuk menyalin model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Lampirkan kebijakan proyek ke proyek sumber dengan mengikuti instruksi di [Melampirkan kebijakan proyek \(SDK\)](#).

3. Jika Anda menyalin model ke AWS akun lain, pastikan Anda memiliki proyek di AWS akun tujuan.
4. Gunakan kode berikut untuk menyalin versi model ke proyek tujuan.

## AWS CLI

Ubah nilai berikut:

- `source-project-arn` ARN dari proyek sumber yang berisi versi model yang ingin Anda salin.
- `source-project-version-arn` ARN dari versi model yang ingin Anda salin.
- `destination-project-arn` ARN proyek tujuan yang ingin Anda salin modelnya.
- `version-name` nama versi untuk model dalam proyek tujuan.
- `bucket` bucket S3 tempat Anda ingin hasil pelatihan untuk model sumber disalin.
- `folder` folder di bucket mana Anda ingin hasil pelatihan untuk model sumber disalin.
- (Opsional) `kms-key-id` ke ID kunci AWS Key Management Service untuk model.
- (Opsional) `key` ke kunci tag yang Anda pilih.
- (Opsional) `value` ke nilai tag yang Anda pilih.

```
aws rekognition copy-project-version \  
  --source-project-arn source-project-arn \  
  --source-project-version-arn source-project-version-arn \  
  --destination-project-arn destination-project-arn \  
  --version-name version-name \  
  --output-config '{"S3Bucket": "bucket", "S3KeyPrefix": "folder"}' \  
  --kms-key-id arn:myKey \  
  --tags '{"key": "key"}' \  
  --profile custom-labels-access
```

## Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `source_project_arn`— ARN dari proyek sumber di AWS akun sumber yang berisi versi model yang ingin Anda salin.
- `source_project_version_arn`— ARN versi model di AWS akun sumber yang ingin Anda salin.

- `destination_project_arn`— ARN dari proyek tujuan yang ingin Anda salin modelnya.
- `destination_version_name`— nama versi untuk model dalam proyek tujuan.
- `training_results`— lokasi S3 tempat Anda ingin hasil pelatihan untuk versi model sumber disalin.
- (Opsional) `kms_key_id` ke ID kunci AWS Key Management Service untuk model.
- (Opsional) `tag_name` ke kunci tag yang Anda pilih.
- (Opsional) `tag_value` ke nilai tag yang Anda pilih.

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

import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def copy_model(
    rekognition_client, source_project_arn, source_project_version_arn,
    destination_project_arn, training_results, destination_version_name):
    """
    Copies a version of a Amazon Rekognition Custom Labels model.

    :param rekognition_client: A Boto3 Amazon Rekognition Custom Labels client.
    :param source_project_arn: The ARN of the source project that contains the
    model that you want to copy.
    :param source_project_version_arn: The ARN of the model version that you
    want
    to copy.
    :param destination_project_arn: The ARN of the project that you want to copy
    the model
    to.
    :param training_results: The Amazon S3 location where training results for
    the model
    should be stored.
    return: The model status and version.
    """
```

```
try:
    logger.info("Copying model...%s from %s to %s ",
source_project_version_arn,
                source_project_arn,
                destination_project_arn)

    output_bucket, output_folder = training_results.replace(
        "s3://", "").split("/", 1)
    output_config = {"S3Bucket": output_bucket,
                    "S3KeyPrefix": output_folder}

    response = rekognition_client.copy_project_version(
        DestinationProjectArn=destination_project_arn,
        OutputConfig=output_config,
        SourceProjectArn=source_project_arn,
        SourceProjectVersionArn=source_project_version_arn,
        VersionName=destination_version_name
    )

    destination_model_arn = response["ProjectVersionArn"]

    logger.info("Destination model ARN: %s", destination_model_arn)

    # Wait until training completes.
    finished = False
    status = "UNKNOWN"
    while finished is False:
        model_description =
rekognition_client.describe_project_versions(ProjectArn=destination_project_arn,
                                             VersionNames=[destination_version_name])
        status = model_description["ProjectVersionDescriptions"][0]
["Status"]

        if status == "COPYING_IN_PROGRESS":
            logger.info("Model copying in progress...")
            time.sleep(60)
            continue

        if status == "COPYING_COMPLETED":
            logger.info("Model was successfully copied.")

        if status == "COPYING_FAILED":
            logger.info(
                "Model copy failed: %s ",
```

```
        model_description["ProjectVersionDescriptions"][0]
["StatusMessage"])

        finished = True
    except ClientError:
        logger.exception("Couldn't copy model.")
        raise
    else:
        return destination_model_arn, status

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

    parser.add_argument(
        "source_project_arn",
        help="The ARN of the project that contains the model that you want to
copy."
    )

    parser.add_argument(
        "source_project_version_arn",
        help="The ARN of the model version that you want to copy."
    )

    parser.add_argument(
        "destination_project_arn",
        help="The ARN of the project which receives the copied model."
    )

    parser.add_argument(
        "destination_version_name",
        help="The version name for the model in the destination project."
    )

    parser.add_argument(
        "training_results",
        help="The S3 location in the destination account that receives the
training results for the copied model."
    )
```

```
def main():

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

    try:

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

        print(
            f"Copying model version {args.source_project_version_arn} to project
{args.destination_project_arn}")

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

        # Copy the model.

        model_arn, status = copy_model(rekognition_client,
                                       args.source_project_arn,
                                       args.source_project_version_arn,
                                       args.destination_project_arn,
                                       args.training_results,
                                       args.destination_version_name,
                                       )

        print(f"Finished copying model: {model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        print(f"Problem copying model: {err}")

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `source_project_arn`— ARN dari proyek sumber di AWS akun sumber yang berisi versi model yang ingin Anda salin.
- `source_project_version-arn`— ARN versi model di AWS akun sumber yang ingin Anda salin.
- `destination_project_arn`— ARN dari proyek tujuan yang ingin Anda salin modelnya.
- `destination_version_name`— nama versi untuk model dalam proyek tujuan.
- `output_bucket`— bucket S3 yang Anda inginkan hasil pelatihan untuk versi model sumber disalin.
- `output_folder`— folder di S3 tempat Anda ingin hasil pelatihan untuk versi model sumber disalin.

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

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.CopyProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.CopyProjectVersionResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;

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

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

public class CopyModel {
```

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

public static ProjectVersionDescription copyMyModel(RekognitionClient
rekClient,
    String sourceProjectArn,
    String sourceProjectVersionArn,
    String destinationProjectArn,
    String versionName,
    String outputBucket,
    String outputFolder) throws InterruptedException {

    try {

        OutputConfig outputConfig =
OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

        String[] logArguments = new String[] { versionName,
sourceProjectArn, destinationProjectArn };

        logger.log(Level.INFO, "Copying model {0} for from project {1} to
project {2}", logArguments);

        CopyProjectVersionRequest copyProjectVersionRequest =
CopyProjectVersionRequest.builder()
            .sourceProjectArn(sourceProjectArn)
            .sourceProjectVersionArn(sourceProjectVersionArn)
            .versionName(versionName)
            .destinationProjectArn(destinationProjectArn)
            .outputConfig(outputConfig)
            .build();

        CopyProjectVersionResponse response =
rekClient.copyProjectVersion(copyProjectVersionRequest);

        logger.log(Level.INFO, "Destination model ARN: {0}",
response.projectVersionArn());
        logger.log(Level.INFO, "Copying model...");

        // wait until copying completes.

        boolean finished = false;

        ProjectVersionDescription copiedModel = null;
```

```
        while (Boolean.FALSE.equals(finished)) {
            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                .versionNames(versionName)
                .projectArn(destinationProjectArn)
                .build();

            DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

                .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                .projectVersionDescriptions()) {

                copiedModel = projectVersionDescription;

                switch (projectVersionDescription.status()) {

                    case COPYING_IN_PROGRESS:
                        logger.log(Level.INFO, "Copying model...");
                        Thread.sleep(5000);
                        continue;

                    case COPYING_COMPLETED:
                        finished = true;
                        logger.log(Level.INFO, "Copying completed");
                        break;

                    case COPYING_FAILED:
                        finished = true;
                        logger.log(Level.INFO, "Copying failed...");
                        break;

                    default:
                        finished = true;
                        logger.log(Level.INFO, "Unexpected copy status %s",
                            projectVersionDescription.statusAsString());
                        break;

                }
            }
        }
```

```
        }

    }

    logger.log(Level.INFO, "Finished copying model {0} for from project
{1} to project {2}", logArguments);

    return copiedModel;

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

}

public static void main(String args[]) {

    String sourceProjectArn = null;
    String sourceProjectVersionArn = null;
    String destinationProjectArn = null;
    String versionName = null;
    String bucket = null;
    String location = null;

    final String USAGE = "\n" + "Usage: "
        + "<source_project_arn> <source_project_version_arn>
<destination_project_arn> <version_name> <output_bucket> <output_folder>\n\n"
        + "Where:\n"
        + "    source_project_arn - The ARN of the project that contains
the model that you want to copy. \n\n"
        + "    source_project_version_arn - The ARN of the project that
contains the model that you want to copy. \n\n"
        + "    destination_project_arn - The ARN of the destination
project that you want to copy the model to. \n\n"
        + "    version_name - A version name for the copied model.\n\n"
        + "    output_bucket - The S3 bucket in which to place the
training output. \n\n"
        + "    output_folder - The folder within the bucket that the
training output is stored in. \n\n";

    if (args.length != 6) {
        System.out.println(USAGE);
    }
}
```

```
        System.exit(1);
    }

    sourceProjectArn = args[0];
    sourceProjectVersionArn = args[1];
    destinationProjectArn = args[2];
    versionName = args[3];
    bucket = args[4];
    location = args[5];

    try {

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

        // Copy the model.
        ProjectVersionDescription copiedModel = copyMyModel(rekClient,
            sourceProjectArn,
            sourceProjectVersionArn,
            destinationProjectArn,
            versionName,
            bucket,
            location);

        System.out.println(String.format("Model copied: %s Status: %s",
            copiedModel.projectVersionArn(),
            copiedModel.statusMessage()));

        rekClient.close();

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

```
}  
  
}
```

## Daftar kebijakan proyek (SDK)

Anda dapat menggunakan [ListProjectPolicies](#) operasi untuk mencantumkan kebijakan proyek yang dilampirkan ke proyek Label Kustom Rekognition Amazon.

Untuk mencantumkan kebijakan proyek yang dilampirkan ke proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk membuat daftar kebijakan proyek.

### AWS CLI

Ubah `project-arn` ke Nama Sumber Daya Amazon dari proyek yang ingin Anda cantumkan kebijakan proyek terlampir.

```
aws rekognition list-project-policies \  
  --project-arn project-arn \  
  --profile custom-labels-access
```

### Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn` — Nama Sumber Daya Amazon dari proyek yang ingin Anda cantumkan kebijakan proyek terlampir.

Misalnya: `python list_project_policies.py project_arn`

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Amazon Rekognition Custom Labels model example used in the service  
documentation:
```

```
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to list the project policies in an Amazon Rekognition Custom Labels
project.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def display_project_policy(project_policy):
    """
    Displays information about a Custom Labels project policy.
    :param project_policy: The project policy (ProjectPolicy)
    that you want to display information about.
    """
    print(f"Policy name: {(project_policy['PolicyName'])}")
    print(f"Project Arn: {project_policy['ProjectArn']}")
    print(f"Document: {(project_policy['PolicyDocument'])}")
    print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
    print()

def list_project_policies(rek_client, project_arn):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n-----')
```

```
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
                NextToken=pagination_token)

            for project in response['ProjectPolicies']:
                display_project_policy(project)

            if 'NextToken' in response:
                pagination_token = response['NextToken']
            else:
                finished = True

        logger.info("Finished listing project policies.")

    except ClientError as err:
        logger.exception(
            "Couldn't list policies for - %s: %s",
            project_arn, err.response['Error']['Message'])
        raise

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

    parser.add_argument(
        "project_arn", help="The Amazon Resource Name of the project for which
        you want to list project policies."
    )

def main():

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

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
```

```
add_arguments(parser)
args = parser.parse_args()

print(f"Listing project policies in: {args.project_arn}")

# List the project policies.

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

list_project_policies(rekognition_client,
                      args.project_arn)

except ClientError as err:
    print(f"Problem list project_policies: {err}")

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `project_arn` — ARN proyek yang memiliki kebijakan proyek yang ingin Anda daftarkan.

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

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesRequest;
import
    software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.ProjectPolicy;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class ListProjectPolicies {

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

    public static void listMyProjectPolicies(RekognitionClient rekClient, String
    projectArn) {

        try {

            logger.log(Level.INFO, "Listing project policies for project: {0}",
            projectArn);

            // List the project policies.

            Boolean finished = false;
            String nextToken = null;

            while (Boolean.FALSE.equals(finished)) {

                ListProjectPoliciesRequest listProjectPoliciesRequest =
                ListProjectPoliciesRequest.builder()
                    .maxResults(5)
                    .projectArn(projectArn)
                    .nextToken(nextToken)
                    .build();

                ListProjectPoliciesResponse response =
                rekClient.listProjectPolicies(listProjectPoliciesRequest);

                for (ProjectPolicy projectPolicy : response.projectPolicies()) {

                    System.out.println(String.format("Name: %s",
                    projectPolicy.policyName()));
                    System.out.println(String.format("Revision ID: %s\n",
                    projectPolicy.policyRevisionId()));

                }

                nextToken = response.nextToken();

            }

        }

    }

}
```

```
        if (nextToken == null) {
            finished = true;
        }
    }

    logger.log(Level.INFO, "Finished listing project policies for
project: {0}", projectArn);

} catch (

    RekognitionException e) {
    logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: " + "<project_arn> \n\n" + "Where:
\n"
        + "    project_arn - The ARN of the project with the project
policies that you want to list.\n\n";
    ;

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

    String projectArn = args[0];

    try {

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

        // List the project policies.
```

```
        listMyProjectPolicies(rekClient, projectArn);

        rekClient.close();

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

}

}
```

## Menghapus kebijakan proyek (SDK)

Anda dapat menggunakan [DeleteProjectPolicy](#) operasi untuk menghapus revisi kebijakan proyek yang ada dari proyek Label Kustom Rekognition Amazon. Jika Anda ingin menghapus semua revisi kebijakan proyek yang dilampirkan ke proyek, gunakan [ListProjectPolicies](#) untuk mendapatkan revisi setiap kebijakan proyek IDs yang dilampirkan ke proyek. Kemudian panggil `DeletePolicy` untuk setiap nama kebijakan.

### Untuk menghapus revisi kebijakan proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
2. Gunakan kode berikut untuk menghapus kebijakan proyek.

`DeletePolicy` mengambil `ProjectARN`, `PolicyName` dan `PolicyRevisionId`.

`ProjectARN` dan `PolicyName` diperlukan untuk API ini. `PolicyRevisionId` bersifat opsional, tetapi dapat dimasukkan untuk keperluan pembaruan atom.

### AWS CLI

Ubah nilai berikut:

- `policy-named` dengan nama kebijakan proyek yang ingin Anda hapus.
- `policy-revision-id` ke ID revisi kebijakan proyek yang ingin Anda hapus.

- `project-arn` Nama Sumber Daya Amazon dari proyek yang berisi revisi kebijakan proyek yang ingin Anda hapus.

```
aws rekognition delete-project-policy \  
  --policy-name policy-name \  
  --policy-revision-id policy-revision-id \  
  --project-arn project-arn \  
  --profile custom-labels-access
```

## Python

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `policy-name`— Nama kebijakan proyek yang ingin Anda hapus.
- `project-arn`— Nama Sumber Daya Amazon dari proyek yang berisi kebijakan proyek yang ingin Anda hapus.
- `policy-revision-id`— ID revisi kebijakan proyek yang ingin Anda hapus.

Sebagai contoh: `python delete_project_policy.py policy_name project_arn policy_revision_id`

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Amazon Rekognition Custom Labels model example used in the service  
documentation:  
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-  
sdk.html  
Shows how to delete a revision of a project policy.  
"""  
  
import argparse  
import logging  
import boto3  
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

def delete_project_policy(rekognition_client, policy_name, project_arn,
                          policy_revision_id=None):
    """
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revision ID for the project policy that you
    want to delete.
    :param project_arn: The Amazon Resource Name of the project that contains
    the project policy
    that you want to delete.
    """
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)

        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise

def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you want to delete project policy {policy_name} ?\n",
        policy_name)
```

```
delete = input("Enter delete to delete your project policy: ")
if delete == "delete":
    return True
else:
    return False

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

    parser.add_argument(
        "policy_name", help="The ARN of the project that contains the project
policy that you want to delete."
    )

    parser.add_argument(
        "project_arn", help="The ARN of the project project policy you want to
delete."
    )

    parser.add_argument(
        "--policy_revision_id", help="(Optional) The revision ID of the project
policy that you want to delete.",
        required=False
    )

def main():

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

    try:

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

        if confirm_project_policy_deletion(args.policy_name) is True:
```

```
print(f"Deleting project_policy: {args.policy_name}")

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

# Delete the project policy.

delete_project_policy(rekognition_client,
                      args.policy_name,
                      args.project_arn,
                      args.policy_revision_id)

print(f"Finished deleting project policy: {args.policy_name}")
else:
    print(f"Not deleting project policy {args.policy_name}")
except ClientError as err:
    print(f"Couldn't delete project policy in {args.policy_name}: {err}")

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

## Java V2

Gunakan kode berikut. Sediakan parameter baris perintah berikut:

- `policy-name`— Nama kebijakan proyek yang ingin Anda hapus.
- `project-arn`— Nama Sumber Daya Amazon dari proyek yang berisi kebijakan proyek yang ingin Anda hapus.
- `policy-revision-id`— ID revisi kebijakan proyek yang ingin Anda hapus.

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

package com.example.rekognition;

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DeleteProjectPolicyRequest;

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

public class DeleteProjectPolicy {

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

    public static void deleteMyProjectPolicy(RekognitionClient rekClient, String
projectArn,
        String projectPolicyName,
        String projectPolicyRevisionId)
        throws InterruptedException {

        try {
            String[] logArguments = new String[] { projectPolicyName,
projectPolicyRevisionId };

            logger.log(Level.INFO, "Deleting: Project policy: {0} revision:
{1}", logArguments);

            // Delete the project policy.

            DeleteProjectPolicyRequest deleteProjectPolicyRequest =
DeleteProjectPolicyRequest.builder()
                .policyName(projectPolicyName)
                .policyRevisionId(projectPolicyRevisionId)
                .projectArn(projectArn).build();

            rekClient.deleteProjectPolicy(deleteProjectPolicyRequest);

            logger.log(Level.INFO, "Deleted: Project policy: {0} revision: {1}",
logArguments);

        } catch (

            RekognitionException e) {
```

```
        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: " + "<project_arn>
<project_policy_name> <project_policy_revision_id>\n\n"
        + "Where:\n"
        + "    project_arn - The ARN of the project that has the project
policy that you want to delete.\n\n"
        + "    project_policy_name - The name of the project policy that
you want to delete.\n\n"
        + "    project_policy_revision_id - The revision of the project
policy that you want to delete.\n\n";

    if (args.length != 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String projectArn = args[0];
    String projectPolicyName = args[1];
    String projectPolicyRevisionId = args[2];

    try {

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

        // Delete the project policy.
        deleteMyProjectPolicy(rekClient, projectArn, projectPolicyName,
projectPolicyRevisionId);

        System.out.println(String.format("project policy deleted: %s
revision: %s", projectPolicyName,
projectPolicyRevisionId));
    }
}
```

```
        rekClient.close();

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

    catch (InterruptedException intError) {
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
        System.exit(1);
    }

}

}
```

## Contoh Label Kustom

Bagian ini berisi contoh yang menunjukkan cara Anda menggunakan kemampuan Amazon Rekognition Custom Labels.

Contoh	Deskripsi
<a href="#">Meningkatkan model dengan umpan balik Model</a>	Menunjukkan cara meningkatkan model menggunakan verifikasi manusia untuk membuat kumpulan data pelatihan baru.
<a href="#">Demonstrasi Label Kustom Rekognition Amazon</a>	Demonstrasi antarmuka pengguna yang menampilkan hasil panggilan ke <code>DetectCustomLabels</code> .
<a href="#">Mendeteksi Label Kustom dalam video</a>	Menunjukkan bagaimana Anda dapat menggunakan <code>DetectCustomLabels</code> dengan bingkai yang diekstrak dari video.
<a href="#">Menganalisis gambar dengan suatu AWS Lambda fungsi</a>	Menunjukkan bagaimana Anda dapat menggunakan <code>DetectCustomLabels</code> dengan fungsi Lambda.
<a href="#">Membuat file manifes dari file CSV</a>	Menunjukkan cara menggunakan file CSV untuk membuat file manifes yang cocok untuk menemukan <a href="#">objek, adegan, dan konsep</a> yang terkait dengan keseluruhan gambar (klasifikasi).

## Meningkatkan model dengan umpan balik Model

Solusi Umpan Balik Model memungkinkan Anda memberikan umpan balik tentang prediksi model Anda dan melakukan perbaikan dengan menggunakan verifikasi manusia. Bergantung pada kasus penggunaan, Anda dapat berhasil dengan kumpulan data pelatihan yang hanya memiliki beberapa gambar. Set pelatihan beranotasi yang lebih besar mungkin diperlukan untuk membangun model

yang lebih akurat. Dengan menggunakan solusi Umpan Balik Model, Anda dapat membuat kumpulan data yang lebih besar melalui bantuan model.

Untuk menginstal dan mengonfigurasi solusi Umpan Balik Model, lihat [Solusi Umpan Balik Model](#).

Alur kerja untuk perbaikan model berkelanjutan adalah sebagai berikut:

1. Latih versi pertama model Anda (mungkin dengan kumpulan data pelatihan kecil).
2. Berikan kumpulan data yang tidak dijelaskan untuk solusi Umpan Balik Model.
3. Solusi Umpan Balik Model menggunakan model saat ini. Ini memulai pekerjaan verifikasi manusia untuk membubuhi keterangan kumpulan data baru.
4. Berdasarkan umpan balik manusia, solusi Umpan Balik Model menghasilkan file manifes yang Anda gunakan untuk membuat model baru.

## Demonstrasi Label Kustom Rekognition Amazon

Demonstrasi Label Kustom Amazon Rekognition menampilkan antarmuka pengguna yang menganalisis gambar dari komputer lokal Anda dengan menggunakan API. [DetectCustomLabels](#)

Aplikasi ini menampilkan informasi tentang model Label Kustom Rekognition Amazon di akun Anda. AWS Setelah Anda memilih model yang sedang berjalan, Anda dapat menganalisis gambar dari komputer lokal Anda. Jika perlu, Anda dapat memulai model. Anda juga dapat menghentikan model yang sedang berjalan. Aplikasi ini menunjukkan integrasi dengan Layanan AWS lainnya seperti Amazon Cognito, Amazon S3, dan Amazon CloudFront

Untuk informasi selengkapnya, lihat Demo [Label Kustom Rekognition Amazon](#).

## Mendeteksi Label Kustom dalam video

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan `DetectCustomLabels` dengan frame yang diekstrak dari video. Kode telah diuji dengan file video dalam format mov dan mp4.

Menggunakan **`DetectCustomLabels`** dengan bingkai yang ditangkap

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).

2. Pastikan Anda memiliki `rekognition:DetectCustomLabels` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
3. Gunakan kode contoh berikut. Ubah nilai `videoFile` menjadi nama file video. Ubah nilai `projectVersionArn` ke Nama Sumber Daya Amazon (ARN) model Label Kustom Rekognition Amazon Anda.

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

"""
Purpose
Shows how to analyze a local video with an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import json
import math
import cv2
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def analyze_video(rek_client, project_version_arn, video_file):
    """
    Analyzes a local video file with an Amazon Rekognition Custom Labels model.
    Creates a results JSON file based on the name of the supplied video file.
    :param rek_client: A Boto3 Amazon Rekognition client.
    :param project_version_arn: The ARN of the Custom Labels model that you want to
    use.
    :param video_file: The video file that you want to analyze.
    """

    custom_labels = []
    cap = cv2.VideoCapture(video_file)
    frame_rate = cap.get(5) # Frame rate.
    while cap.isOpened():
        frame_id = cap.get(1) # Current frame number.
        print(f"Processing frame id: {frame_id}")
```

```
ret, frame = cap.read()
if ret is not True:
    break
if frame_id % math.floor(frame_rate) == 0:
    has_frame, image_bytes = cv2.imencode(".jpg", frame)

    if has_frame:
        response = rek_client.detect_custom_labels(
            Image={
                'Bytes': image_bytes.tobytes(),
            },
            ProjectVersionArn=project_version_arn
        )

        for elabel in response["CustomLabels"]:
            elabel["Timestamp"] = (frame_id/frame_rate)*1000
            custom_labels.append(elabel)

print(custom_labels)

with open(video_file + ".json", "w", encoding="utf-8") as f:
    f.write(json.dumps(custom_labels))

cap.release()

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

    parser.add_argument(
        "project_version_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "video_file", help="The local path to the video that you want to analyze."
    )

def main():

    logging.basicConfig(level=logging.INFO,
```

```
format="%levelname)s: %(message)s")

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

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

    analyze_video(rekognition_client,
                  args.project_version_arn, args.video_file)

except ClientError as err:
    print(f"Couldn't analyze video: {err}")

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

## Menganalisis gambar dengan suatu AWS Lambda fungsi

AWS Lambda adalah layanan komputasi yang memungkinkan Anda menjalankan kode tanpa menyediakan atau mengelola server. Misalnya, Anda dapat menganalisis gambar yang dikirimkan dari aplikasi seluler tanpa harus membuat server untuk meng-host kode aplikasi. Petunjuk berikut menunjukkan cara membuat fungsi Lambda dengan Python yang memanggil [DetectCustomLabels](#) Fungsi menganalisis gambar yang disediakan dan mengembalikan daftar label yang ditemukan dalam gambar. Instruksi termasuk contoh kode Python yang menunjukkan cara memanggil fungsi Lambda dengan gambar di bucket Amazon S3, atau gambar yang disediakan dari komputer lokal.

### Topik

- [Langkah 1: Buat AWS Lambda fungsi \(konsol\)](#)
- [Langkah 2: \(Opsional\) Buat layer \(konsol\)](#)
- [Langkah 3: Tambahkan kode Python \(konsol\)](#)
- [Langkah 4: Coba fungsi Lambda Anda](#)

## Langkah 1: Buat AWS Lambda fungsi (konsol)

Pada langkah ini, Anda membuat AWS fungsi kosong dan peran eksekusi IAM yang memungkinkan fungsi Anda memanggil DetectCustomLabels operasi. Ini juga memberikan akses ke bucket Amazon S3 yang menyimpan gambar untuk dianalisis. Anda juga menentukan variabel lingkungan untuk berikut ini:

- Model Label Kustom Amazon Rekognition yang Anda inginkan untuk digunakan oleh fungsi Lambda Anda.
- Batas kepercayaan yang Anda ingin model gunakan.

Kemudian Anda menambahkan kode sumber dan secara opsional lapisan ke fungsi Lambda.

Untuk membuat AWS Lambda fungsi (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di <https://console.aws.amazon.com/lambda/>.
2. Pilih Buat fungsi. Untuk informasi selengkapnya, lihat [Membuat Fungsi Lambda dengan Konsol](#).
3. Pilih opsi berikut.
  - Pilih Tulis dari awal.
  - Masukkan nilai untuk nama Fungsi.
  - Untuk Runtime pilih Python 3.10.
4. Pilih Buat fungsi untuk membuat AWS Lambda fungsi.
5. Pada halaman fungsi, Pilih tab Konfigurasi.
6. Pada panel variabel Lingkungan, pilih Edit.
7. Tambahkan variabel lingkungan berikut. Untuk setiap variabel pilih Tambahkan variabel lingkungan dan kemudian masukkan kunci variabel dan nilai.

Key	Nilai
MODEL_ARN	Nama Sumber Daya Amazon (ARN) dari model yang Anda inginkan untuk digunakan oleh fungsi Lambda Anda. Anda bisa mendapatkan ARN model dari tab Use Model

Key	Nilai
	pada halaman detail model di konsol Amazon Rekognition Custom Labels.
KEPERCAYAAN	Nilai minimum (0-100) untuk kepercayaan model dalam prediksi untuk label. Fungsi Lambda tidak mengembalikan label dengan nilai kepercayaan lebih rendah dari nilai ini.

8. Pilih Simpan untuk menyimpan variabel lingkungan.
9. Pada panel Izin, Di bawah nama Peran, pilih peran eksekusi untuk membuka peran di konsol IAM.
10. Di tab Izin, pilih Tambahkan izin, lalu Buat kebijakan sebaris.
11. Pilih JSON dan ganti kebijakan yang ada dengan kebijakan berikut.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "rekognition:DetectCustomLabels",
      "Resource": "*",
      "Effect": "Allow",
      "Sid": "DetectCustomLabels"
    }
  ]
}

```

12. Pilih Berikutnya.
13. Di Detail kebijakan, masukkan nama untuk kebijakan, seperti DetectCustomLabels-access.
14. Pilih Buat kebijakan.
15. Jika Anda menyimpan gambar untuk dianalisis dalam bucket Amazon S3, ulangi langkah 10—14.
  - a. Untuk langkah 11, gunakan kebijakan berikut. Ganti *bucket/folder path* dengan bucket Amazon S3 dan path folder ke gambar yang ingin Anda analisis.

## JSON

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

- b. Untuk langkah 13, pilih nama kebijakan yang berbeda, seperti S3Bucket-access.

## Langkah 2: (Opsional) Buat layer (konsol)

Untuk menjalankan contoh ini, Anda tidak perlu melakukan langkah ini.

DetectCustomLabelsOperasi ini disertakan dalam lingkungan Lambda Python default sebagai bagian dari AWS SDK for Python (Boto3). Jika bagian lain dari fungsi Lambda Anda memerlukan pembaruan AWS layanan terbaru yang tidak ada di lingkungan Lambda Python default, lakukan langkah ini untuk menambahkan rilis Boto3 SDK terbaru sebagai lapisan ke fungsi Anda.

Pertama, Anda membuat arsip file.zip yang berisi Boto3 SDK. Anda kemudian membuat layer dan menambahkan arsip file.zip ke layer. Untuk informasi selengkapnya, lihat [Menggunakan layer dengan fungsi Lambda Anda](#).

Untuk membuat dan menambahkan lapisan (konsol)

1. Buka prompt perintah dan masukkan perintah berikut.

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

2. Perhatikan nama file zip (boto3-layer.zip). Anda membutuhkannya di langkah 6 dari prosedur ini.
3. Buka AWS Lambda konsol di <https://console.aws.amazon.com/lambda/>.
4. Di panel navigasi, pilih Layers (Lapisan).

5. Pilih Buat lapisan.
6. Masukkan nilai untuk Nama dan Deskripsi.
7. Pilih Unggah file.zip dan pilih Unggah.
8. Di kotak dialog, pilih arsip file.zip (boto3-layer.zip) yang Anda buat di langkah 1 prosedur ini.
9. Untuk runtime yang kompatibel, pilih Python 3.9.
10. Pilih Create untuk membuat layer.
11. Pilih ikon menu panel navigasi.
12. Di panel navigasi, pilih Fungsi.
13. Dalam daftar sumber daya, pilih fungsi yang Anda buat [Langkah 1: Buat AWS Lambda fungsi \(konsol\)](#).
14. Pilih tab Kode.
15. Di bagian Layers, pilih Add a layer.
16. Pilih Custom Layers.
17. Di Custom layers, pilih nama layer yang Anda masukkan pada langkah 6.
18. Di Versi pilih versi layer, yang seharusnya 1.
19. Pilih Tambahkan.

### Langkah 3: Tambahkan kode Python (konsol)

Pada langkah ini, Anda menambahkan kode Python ke fungsi Lambda Anda dengan menggunakan editor kode konsol Lambda. Kode menganalisis gambar yang disediakan dengan `DetectCustomLabels` dan mengembalikan daftar label yang ditemukan dalam gambar. Gambar yang disediakan dapat ditemukan di bucket Amazon S3 atau disediakan sebagai byte gambar yang dikodekan `byte64`.

Untuk menambahkan kode Python (konsol)

1. Jika Anda tidak berada di konsol Lambda, lakukan hal berikut:
  - a. Buka AWS Lambda konsol di <https://console.aws.amazon.com/lambda/>.
  - b. Buka fungsi Lambda yang Anda buat. [Langkah 1: Buat AWS Lambda fungsi \(konsol\)](#)
2. Pilih tab Kode.
3. Di Sumber kode, ganti kode di `lambda_function.py` dengan yang berikut ini:

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

"""
Purpose
An AWS lambda function that analyzes images with an the Amazon Rekognition
Custom Labels model.
"""
import json
import base64
from os import environ
import logging
import boto3

from botocore.exceptions import ClientError

# Set up logging.
logger = logging.getLogger(__name__)

# Get the model ARN and confidence.
model_arn = environ['MODEL_ARN']
min_confidence = int(environ.get('CONFIDENCE', 50))

# Get the boto3 client.
rek_client = boto3.client('rekognition')

def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """

    try:

        # Determine image source.
        if 'image' in event:
            # Decode the image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
```

```
        image = {'Bytes': img_b64decoded}

    elif 'S3Object' in event:
        image = {'S3Object':
                {'Bucket': event['S3Object']['Bucket'],
                 'Name': event['S3Object']['Name']}}
    }

    else:
        raise ValueError(
            'Invalid source. Only image base 64 encoded image bytes or S3Object
are supported.')

    # Analyze the image.
    response = rek_client.detect_custom_labels(Image=image,
        MinConfidence=min_confidence,
        ProjectVersionArn=model_arn)

    # Get the custom labels
    labels = response['CustomLabels']

    lambda_response = {
        "statusCode": 200,
        "body": json.dumps(labels)
    }

except ClientError as err:
    error_message = f"Couldn't analyze image. " + \
        err.response['Error']['Message']

    lambda_response = {
        'statusCode': 400,
        'body': {
            "Error": err.response['Error']['Code'],
            "ErrorMessage": error_message
        }
    }
    logger.error("Error function %s: %s",
        context.invoked_function_arn, error_message)

except ValueError as val_error:
    lambda_response = {
```

```
        'statusCode': 400,  
        'body': {  
            "Error": "ValueError",  
            "ErrorMessage": format(val_error)  
        }  
    }  
    logger.error("Error function %s: %s",  
                context.invoked_function_arn, format(val_error))  
  
    return lambda_response
```

4. Pilih Deploy untuk menerapkan fungsi Lambda Anda.

## Langkah 4: Coba fungsi Lambda Anda

Pada langkah ini Anda menggunakan kode Python di komputer Anda untuk meneruskan gambar lokal, atau gambar di bucket Amazon S3, ke fungsi Lambda Anda. Gambar yang dilewatkan dari komputer lokal harus lebih kecil dari 6291456 byte. Jika gambar Anda lebih besar, unggah gambar ke bucket Amazon S3 dan panggil skrip dengan jalur Amazon S3 ke gambar. Untuk informasi tentang mengunggah file gambar ke bucket Amazon S3, [lihat](#) Mengunggah objek.

Pastikan Anda menjalankan kode di AWS Wilayah yang sama di mana Anda membuat fungsi Lambda. [Anda dapat melihat AWS Wilayah untuk fungsi Lambda Anda di bilah navigasi halaman detail fungsi di konsol Lambda.](#)

Jika AWS Lambda fungsi mengembalikan kesalahan batas waktu, perpanjang periode batas waktu untuk fungsi fungsi Lambda, Untuk informasi selengkapnya, lihat [Mengonfigurasi batas waktu fungsi](#) (konsol).

Untuk informasi selengkapnya tentang menjalankan fungsi Lambda dari kode Anda, [lihat AWS Lambda Memanggil Fungsi.](#)

Untuk mencoba fungsi Lambda Anda

1. Pastikan Anda memiliki `lambda:InvokeFunction` izin. Anda dapat menggunakan kebijakan berikut.

[Anda bisa mendapatkan ARN untuk fungsi fungsi Lambda Anda dari ikhtisar fungsi di konsol Lambda.](#)

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Buat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) dalam Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Buat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

2. Instal dan konfigurasi AWS SDK untuk Python. Untuk informasi selengkapnya, lihat [Langkah 4: Mengatur AWS CLI and AWS SDK](#).
3. [Mulai model yang](#) Anda tentukan pada langkah 7 dari [Langkah 1: Buat AWS Lambda fungsi \(konsol\)](#).
4. Simpan kode berikut ke file bernama `client.py`.

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

"""
Purpose
Test code for running the Amazon Rekognition Custom Labels Lambda
function example code.
"""

import argparse
import logging
import base64
import json
import boto3

from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

def analyze_image(function_name, image):
    """Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")

            lambda_payload = {"image": data}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))

    return json.loads(response['Payload'].read().decode())

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

```
"""

parser.add_argument(
    "function", help="The name of the AWS Lambda function that you want " \
    "to use to analyze the image.")
parser.add_argument(
    "image", help="The local image that you want to analyze.")

def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

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

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)

if __name__ == "__main__":
```

```
main()
```

5. Jalankan kode tersebut. Untuk argumen baris perintah, berikan nama fungsi Lambda dan gambar yang ingin Anda analisis. Anda dapat menyediakan jalur ke gambar lokal, atau jalur S3 ke gambar yang disimpan di bucket Amazon S3. Contoh:

```
python client.py function_name s3://bucket/path/image.jpg
```

Jika gambar berada di bucket Amazon S3, pastikan itu adalah bucket yang sama dengan yang Anda tentukan pada langkah 15. [Langkah 1: Buat AWS Lambda fungsi \(konsol\)](#)

Jika berhasil, outputnya adalah daftar label yang ditemukan pada gambar. Jika tidak ada label yang dikembalikan, pertimbangkan untuk menurunkan nilai kepercayaan yang Anda tetapkan pada langkah 7 dari [Langkah 1: Buat AWS Lambda fungsi \(konsol\)](#).

6. Jika Anda telah selesai dengan fungsi Lambda dan model tidak digunakan oleh aplikasi lain, [hentikan](#) model. Ingatlah untuk [memulai model](#) saat berikutnya Anda ingin gunakan fungsi Lambda.

# Keamanan

Anda dapat mengamankan pengelolaan proyek, model, dan DetectCustomLabels operasi yang digunakan pelanggan Anda untuk mendeteksi label khusus.

Untuk informasi selengkapnya tentang mengamankan Amazon Rekognition, lihat [Amazon Rekognition Security](#).

## Mengamankan proyek Label Kustom Rekognition Amazon

Anda dapat mengamankan proyek Label Kustom Rekognition Amazon dengan menentukan izin tingkat sumber daya yang ditentukan dalam kebijakan berbasis identitas. Untuk informasi lebih lanjut, lihat [Kebijakan Berbasis Identitas dan Kebijakan Berbasis Sumber Daya](#).

Sumber daya Amazon Rekognition Custom Labels yang dapat Anda amankan adalah:

Sumber daya	Format Nama Sumber Daya Amazon
Proyek	arn:aws:rekognition: *:*:project//datetime <i>project_name</i>
Model	arn:aws:rekognition: *:*:proyek//versi//datetime <i>project_name name</i>

Contoh kebijakan berikut menunjukkan cara memberikan izin identitas ke:

- Jelaskan semua proyek.
- Buat, mulai, hentikan, dan gunakan model tertentu untuk inferensi.
- Buat proyek. Buat dan jelaskan model tertentu.
- Tolak pembuatan proyek tertentu.

### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Sid": "AllResources",
  "Effect": "Allow",
  "Action": "rekognition:DescribeProjects",
  "Resource": "*"
},
{
  "Sid": "SpecificProjectVersion",
  "Effect": "Allow",
  "Action": [
    "rekognition:StopProjectVersion",
    "rekognition:StartProjectVersion",
    "rekognition:DetectCustomLabels",
    "rekognition:CreateProjectVersion"
  ],
  "Resource": "arn:aws:rekognition:*:*:project/MyProject/
version/MyVersion/*"
},
{
  "Sid": "SpecificProject",
  "Effect": "Allow",
  "Action": [
    "rekognition:CreateProject",
    "rekognition:DescribeProjectVersions",
    "rekognition:CreateProjectVersion"
  ],
  "Resource": "arn:aws:rekognition:*:*:project/MyProject/*"
},
{
  "Sid": "ExplicitDenyCreateProject",
  "Effect": "Deny",
  "Action": [
    "rekognition:CreateProject"
  ],
  "Resource": ["arn:aws:rekognition:*:*:project/SampleProject/*"]
}
]
```

## Mengamankan DetectCustomLabels

Identitas yang digunakan untuk mendeteksi label kustom mungkin berbeda dari identitas yang mengelola model Amazon Rekognition Custom Labels.

Anda dapat mengamankan akses akses identitas DetectCustomLabels dengan menerapkan kebijakan ke identitas. Contoh berikut membatasi akses ke DetectCustomLabels hanya dan ke model tertentu. Identitas tidak memiliki akses ke salah satu operasi Rekognition Amazon lainnya.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rekognition:DetectCustomLabels"
      ],
      "Resource": "arn:aws:rekognition:*:*:project/MyProject/
version/MyVersion/*"
    }
  ]
}
```

## Kebijakan terkelola AWS

Kami menyediakan kebijakan AmazonRekognitionCustomLabelsFullAccess AWS terkelola yang dapat Anda gunakan untuk mengontrol akses ke Label Kustom Rekognition Amazon. Untuk informasi selengkapnya, lihat [kebijakan terkelola AWS: AmazonRekognitionCustomLabelsFullAccess](#).

# Pedoman dan kuota di Label Kustom Rekognition Amazon

Bagian berikut memberikan pedoman dan kuota saat menggunakan Label Kustom Rekognition Amazon.

## Wilayah yang Didukung

Untuk daftar AWS Wilayah tempat Label Kustom Amazon Rekognition tersedia, lihat [Wilayah dan Titik Akhir AWS](#) di Referensi Umum Amazon Web Services.

## Kuota

Berikut ini adalah daftar batasan di Amazon Rekognition Custom Labels. Untuk informasi tentang batasan yang dapat Anda ubah, lihat [Batas AWS Layanan](#). Untuk mengubah batas, lihat [Membuat Kasus](#).

## Pelatihan

- Format file yang didukung adalah format gambar PNG dan JPEG.
- Jumlah maksimum kumpulan data pelatihan dalam versi model adalah 1.
- Ukuran file manifes kumpulan data maksimum adalah 1 GB.
- Jumlah minimum label unik per kumpulan data Objek, Adegan, dan Konsep (klasifikasi) adalah 2.
- Jumlah minimum label unik per kumpulan data Object Location (deteksi) adalah 1.
- Jumlah maksimum label unik per manifes adalah 250.
- Jumlah minimum gambar per label adalah 1.
- Jumlah maksimum gambar per kumpulan data Lokasi Objek (deteksi) adalah 250.000.

Batas wilayah Asia Pasifik (Mumbai) dan Eropa (London) AWS adalah 28.000 gambar.

- Jumlah maksimum gambar per kumpulan data Objek, Adegan, dan Konsep (klasifikasi) adalah 500.000. Defaultnya adalah 250.000. Untuk meminta peningkatan, lihat [Membuat Kasus](#).

Batas wilayah Asia Pasifik (Mumbai) dan Eropa (London) AWS adalah 28.000 gambar. Anda tidak dapat meminta kenaikan batas.

- Jumlah maksimum label per gambar adalah 50.
- Jumlah minimum kotak pembatas dalam gambar adalah 0.

- Jumlah maksimum kotak pembatas dalam gambar adalah 50.
- Dimensi gambar minimum file gambar dalam bucket Amazon S3 adalah 64 piksel x 64 piksel.
- Dimensi gambar maksimum file gambar dalam bucket Amazon S3 adalah 4096 piksel x 4096 piksel.
- Ukuran file maksimum untuk gambar dalam bucket Amazon S3 adalah 15 MB.
- Rasio aspek gambar maksimum adalah 20:1.

## Pengujian

- Jumlah maksimum kumpulan data pengujian dalam versi model adalah 1.
- Ukuran file manifes kumpulan data maksimum adalah 1 GB.
- Jumlah minimum label unik per kumpulan data Objek, Adegan, dan Konsep (klasifikasi) adalah 2.
- Jumlah minimum label unik per kumpulan data Object Location (deteksi) adalah 1.
- Jumlah maksimum label unik per kumpulan data adalah 250.
- Jumlah minimum gambar per label adalah 0.
- Jumlah maksimum gambar per label adalah 1000.
- Jumlah maksimum gambar per kumpulan data Lokasi Objek (deteksi) adalah 250.000.

Batas wilayah Asia Pasifik (Mumbai) dan Eropa (London) AWS adalah 7.000 gambar.

- Jumlah maksimum gambar per kumpulan data Objek, Adegan, dan Konsep (klasifikasi) adalah 500.000. Defaultnya adalah 250.000. Untuk meminta peningkatan, lihat [Membuat Kasus](#).

Batas wilayah Asia Pasifik (Mumbai) dan Eropa (London) AWS adalah 7.000 gambar. Anda tidak dapat meminta kenaikan batas.

- Jumlah minimum label per gambar per manifes adalah 0.
- Jumlah maksimum label per gambar per manifes adalah 50.
- Jumlah minimum kotak pembatas dalam gambar per manifes adalah 0.
- Jumlah maksimum kotak pembatas dalam gambar per manifes adalah 50.
- Dimensi gambar minimum file gambar dalam bucket Amazon S3 adalah 64 piksel x 64 piksel.
- Dimensi gambar maksimum file gambar dalam bucket Amazon S3 adalah 4096 piksel x 4096 piksel.
- Ukuran file maksimum untuk gambar dalam bucket Amazon S3 adalah 15 MB.
- Format file yang didukung adalah format gambar PNG dan JPEG.

- Rasio aspek gambar maksimum adalah 20:1.

## Deteksi

- Ukuran maksimum gambar yang diteruskan sebagai byte mentah adalah 4 MB.
- Ukuran file maksimum untuk gambar dalam bucket Amazon S3 adalah 15 MB.
- Dimensi gambar minimum dari file gambar input (disimpan dalam bucket Amazon S3 atau disediakan sebagai byte gambar) adalah 64 piksel x 64 piksel.
- Dimensi gambar maksimum dari file gambar input (disimpan di Amazon S3 atau disediakan sebagai byte gambar) adalah 4096 piksel x 4096 piksel.
- Format file yang didukung adalah format gambar PNG dan JPEG.
- Rasio aspek gambar maksimum adalah 20:1.

## Penyalinan model

- Jumlah maksimum kebijakan proyek yang dapat Anda [lampirkan](#) ke proyek adalah 5.
- Jumlah maksimum pekerjaan penyalinan bersamaan di suatu tujuan adalah 5.

# Referensi API Label Kustom Amazon Rekognition

API Label Kustom Amazon Rekognition didokumentasikan sebagai bagian dari konten referensi API Rekognition Amazon. Ini adalah daftar operasi API Label Kustom Amazon Rekognition dengan tautan ke topik referensi API Rekognition Amazon yang sesuai. Selain itu, tautan referensi API dalam dokumen ini masuk ke topik referensi API Panduan Pengembang Amazon Rekognition yang sesuai. Untuk informasi tentang penggunaan API, lihat

Bagian ini memberi Anda ikhtisar alur kerja untuk melatih dan menggunakan model Label Kustom Rekognition Amazon dengan konsol dan SDK. AWS

## Note

Amazon Rekognition Custom Labels sekarang mengelola kumpulan data dalam sebuah proyek. Anda dapat membuat kumpulan data untuk proyek Anda dengan konsol dan dengan

SDK AWS . Jika sebelumnya Anda telah menggunakan Label Kustom Amazon Rekognition,

kumpulan data lama Anda mungkin perlu dikaitkan dengan proyek baru. Untuk informasi

selengkapnya, lihat [Langkah 6: \(Opsional\) Kaitkan kumpulan data sebelumnya dengan proyek baru](#)

## Topik

- [Tentukan jenis model Anda](#)
- [Buat model](#)
- [Tingkatkan model Anda](#)
- [Mulai model Anda](#)
- [Menganalisis gambar](#)
- [Hentikan model Anda](#)

## Tentukan jenis model Anda

Pertama-tama Anda memutuskan jenis model mana yang ingin Anda latih, yang tergantung pada tujuan bisnis Anda. Misalnya, Anda dapat melatih model untuk menemukan logo Anda di pos media sosial, mengidentifikasi produk Anda di rak-rak toko, atau mengklasifikasikan suku cadang mesin di jalur perakitan.

---

Amazon Rekognition Custom Labels dapat melatih jenis model berikut:

---

- [Temukan objek, adegan, dan konsep](#)
- 

- [Temukan lokasi objek](#)
- 

- [Temukan lokasi merek](#)
- 

Untuk membantu Anda memutuskan jenis model yang akan dilatih, Amazon Rekognition Custom Labels menyediakan contoh proyek yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat [Memulai Label Kustom Amazon Rekognition](#).

---

## Temukan objek, adegan, dan konsep

---

Model memprediksi klasifikasi untuk objek, adegan, dan konsep yang terkait dengan keseluruhan gambar. Misalnya, Anda dapat melatih model yang menentukan apakah suatu gambar berisi objek wisata, atau tidak. Untuk contoh proyek, lihat [Klasifikasi gambar](#). Gambar danau berikut ini adalah contoh jenis gambar yang dapat Anda kenali objek, pemandangan, dan konsep.

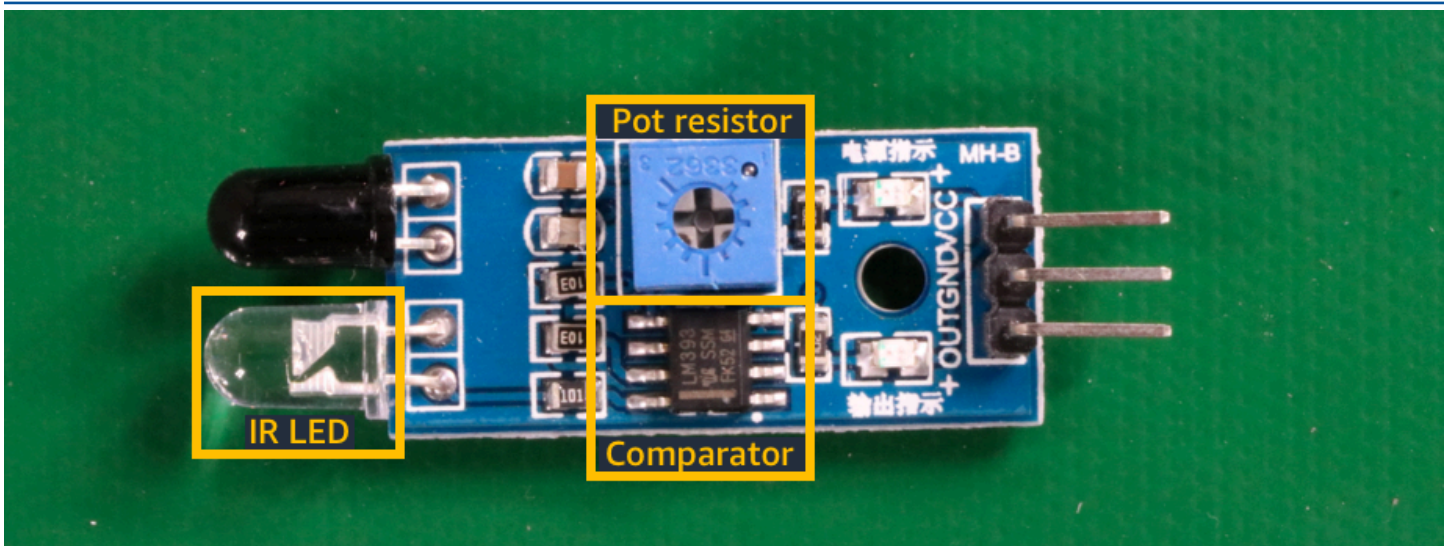
---



Atau, Anda dapat melatih model yang mengkategorikan gambar ke dalam beberapa kategori. Misalnya, gambar sebelumnya mungkin memiliki kategori seperti warna langit, refleksi, atau danau. Untuk contoh proyek, lihat [Klasifikasi gambar multi-label](#).

## Temukan lokasi objek

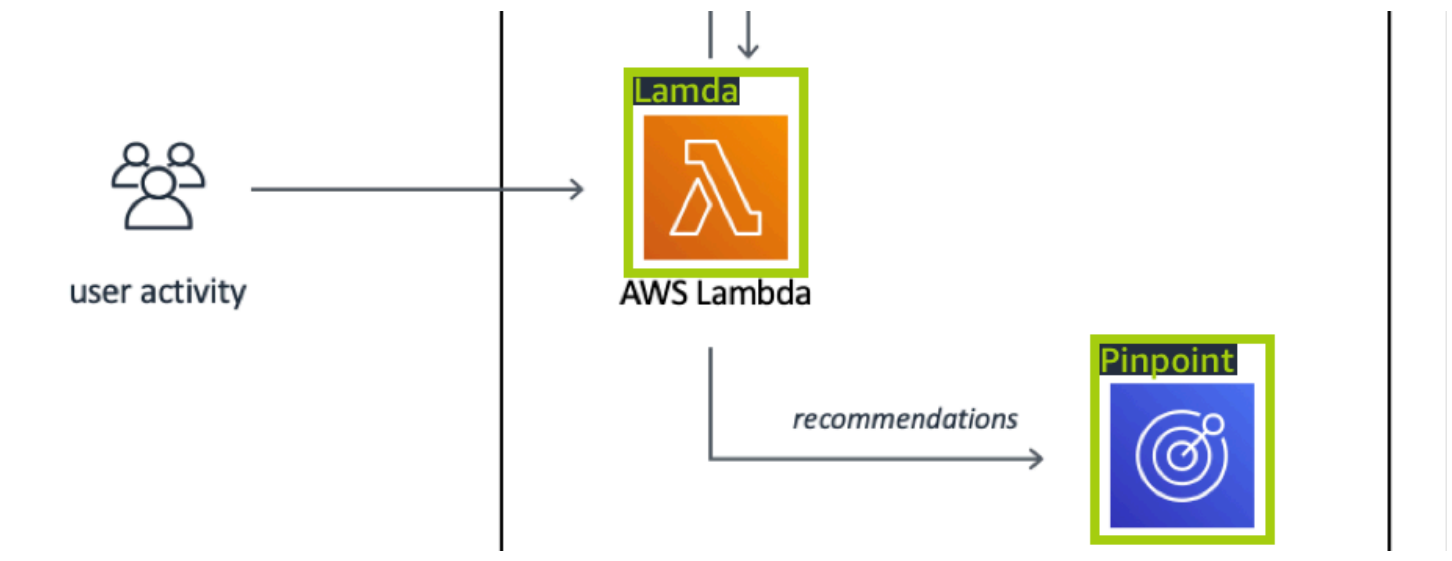
Model memprediksi lokasi objek pada gambar. Prediksi mencakup informasi kotak pembatas untuk lokasi objek dan label yang mengidentifikasi objek di dalam kotak pembatas. Misalnya, gambar berikut menunjukkan kotak pembatas di sekitar berbagai bagian papan sirkuit, seperti komparator atau resistor pot.



Proyek [Lokalisasi objek](#) contoh menunjukkan bagaimana Amazon Rekognition Custom Labels menggunakan kotak pembatas berlabel untuk melatih model yang menemukan lokasi objek.

## Temukan lokasi merek

Amazon Rekognition Custom Labels dapat melatih model yang menemukan lokasi merek, seperti logo, pada gambar. Prediksi tersebut mencakup informasi kotak pembatas untuk lokasi merek dan label yang mengidentifikasi objek di dalam kotak pembatas. Untuk contoh proyek, lihat [Deteksi merek](#). Gambar berikut adalah contoh dari beberapa merek yang dapat dideteksi oleh model.



## Buat model

Langkah-langkah untuk membuat model adalah membuat proyek, membuat kumpulan data pelatihan dan pengujian, dan melatih model.

## Membuat proyek

Proyek Amazon Rekognition Custom Labels adalah sekelompok sumber daya yang diperlukan untuk membuat dan mengelola model. Sebuah proyek mengelola hal-hal berikut:

- **Datasets** — Gambar dan label gambar yang digunakan untuk melatih model. Sebuah proyek memiliki dataset pelatihan dan dataset pengujian.
- **Model** — Perangkat lunak yang Anda latih untuk menemukan konsep, adegan, dan objek yang unik untuk bisnis Anda. Anda dapat memiliki beberapa versi model dalam sebuah proyek.

Kami menyarankan Anda menggunakan proyek untuk kasus penggunaan tunggal, seperti menemukan bagian papan sirkuit pada papan sirkuit.

Anda dapat membuat proyek dengan konsol Amazon Rekognition Custom Labels dan dengan API. [CreateProject](#) Untuk informasi selengkapnya, lihat [Membuat proyek](#).

## Buat kumpulan data pelatihan dan uji

Dataset adalah sekumpulan gambar dan label yang menggambarkan gambar-gambar tersebut. Dalam proyek Anda, Anda membuat kumpulan data pelatihan dan kumpulan data pengujian yang digunakan Label Kustom Amazon Rekognition untuk melatih dan menguji model Anda.

Label mengidentifikasi objek, adegan, konsep, atau kotak pembatas di sekitar objek dalam gambar. Label ditetapkan ke seluruh gambar (tingkat gambar) atau ditetapkan ke kotak pembatas yang mengelilingi objek pada gambar.

### Important

Cara Anda memberi label pada gambar dalam kumpulan data menentukan jenis model yang dibuat Label Kustom Rekognition Amazon. Misalnya, untuk melatih model yang

menemukan objek, adegan, dan konsep, Anda menetapkan label tingkat gambar ke gambar

dalam kumpulan data pelatihan dan pengujian Anda. Untuk informasi selengkapnya, lihat [Mengarahkan kumpulan data](#).

Gambar harus dalam format PNG dan JPEG, dan Anda harus mengikuti rekomendasi gambar input. Untuk informasi selengkapnya, lihat [Mempersiapkan gambar](#).

## Buat kumpulan data pelatihan dan uji (Konsol)

Anda dapat memulai proyek dengan satu kumpulan data, atau dengan kumpulan data pelatihan dan pengujian terpisah. Jika Anda memulai dengan satu kumpulan data, Amazon Rekognition Custom Labels membagi kumpulan data Anda selama pelatihan untuk membuat kumpulan data pelatihan (80%) dan kumpulan data pengujian (20%) untuk proyek Anda. Mulailah dengan satu kumpulan data jika Anda ingin Label Kustom Rekognition Amazon memutuskan gambar mana yang digunakan untuk pelatihan dan pengujian. Untuk kontrol penuh atas pelatihan, pengujian, dan penyetelan kinerja, kami menyarankan Anda memulai proyek dengan kumpulan data pelatihan dan pengujian terpisah.

Untuk membuat kumpulan data untuk proyek, Anda mengimpor gambar dengan salah satu cara berikut:

- Impor gambar dari komputer lokal Anda.
- **Impor gambar dari ember S3.** Amazon Rekognition Custom Labels dapat memberi label gambar menggunakan nama folder yang berisi gambar.

- Impor file manifes Amazon SageMaker AI Ground Truth.

- Salin kumpulan data Label Kustom Rekognition Amazon yang ada.

Untuk informasi selengkapnya, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#).

Tergantung dari mana Anda mengimpor gambar, gambar Anda mungkin tidak berlabel. Misalnya, gambar yang diimpor dari komputer lokal tidak diberi label. Gambar yang diimpor dari file manifes Amazon SageMaker AI Ground Truth diberi label. Anda dapat menggunakan konsol Amazon Rekognition Custom Labels untuk menambahkan, mengubah, dan menetapkan label. Untuk informasi selengkapnya, lihat [Pelabelan gambar](#).

Untuk membuat kumpulan data pelatihan dan pengujian Anda dengan konsol, lihat [Membuat kumpulan data pelatihan dan uji dengan gambar](#) Untuk tutorial yang mencakup pembuatan kumpulan data pelatihan dan pengujian, lihat [Mengklasifikasikan gambar](#)

## Buat kumpulan data pelatihan dan pengujian (SDK)

Untuk membuat kumpulan data pelatihan dan pengujian, Anda menggunakan API. `CreateDataset` Anda dapat membuat kumpulan data dengan menggunakan file manifes format Amazon Sagemaker atau dengan menyalin kumpulan data Label Kustom Rekognition Amazon yang ada. Untuk informasi selengkapnya, lihat [Buat kumpulan data pelatihan dan pengujian \(SDK\)](#) Jika perlu, Anda dapat membuat file manifes Anda sendiri. Untuk informasi selengkapnya, lihat [the section called “Membuat file manifes”](#).

## Latih model Anda

Latih model Anda dengan dataset pelatihan. Versi baru dari sebuah model dibuat setiap kali dilatih. Selama pelatihan, Amazon Rekognition Custom Labels menguji kinerja model terlatih Anda. Anda dapat menggunakan hasilnya untuk mengevaluasi dan meningkatkan model Anda. Membutuhkan waktu beberapa saat untuk menyelesaikan pelatihan. Anda hanya dikenakan biaya untuk pelatihan model yang sukses. Untuk informasi selengkapnya, lihat [Melatih model Label Kustom Rekognition Amazon](#). Jika pelatihan model gagal, Amazon Rekognition Custom Labels menyediakan informasi debugging yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat [Mendebug pelatihan model yang gagal](#).

## Latih model Anda (Konsol)

Untuk melatih model Anda dengan konsol, lihat [Melatih model \(Konsol\)](#).

---

## Melatih model (SDK)

---

Anda melatih model Label Kustom Rekognition Amazon dengan menelepon [CreateProjectVersion](#). Untuk informasi selengkapnya, lihat [Melatih model \(SDK\)](#).

---

---

## Tingkatkan model Anda

---

Selama pengujian, Amazon Rekognition Custom Labels membuat metrik evaluasi yang dapat Anda gunakan untuk meningkatkan model terlatih Anda.

---

---

## Evaluasi model Anda

---

Evaluasi kinerja model Anda dengan menggunakan metrik kinerja yang dibuat selama pengujian. Metrik kinerja, seperti F1, presisi, dan penarikan, memungkinkan Anda memahami kinerja model terlatih Anda, dan memutuskan apakah Anda siap menggunakannya dalam produksi. Untuk informasi selengkapnya, lihat [Metrik untuk mengevaluasi model Anda](#).

---

---

## Evaluasi model (konsol)

---

Untuk melihat metrik kinerja, lihat [Mengakses metrik evaluasi \(Konsol\)](#).

---

---

## Mengevaluasi model (SDK)

---

Untuk mendapatkan metrik kinerja, Anda menelepon [DescribeProjectVersions](#) untuk mendapatkan hasil pengujian. Untuk informasi selengkapnya, lihat [Mengakses metrik evaluasi Label Kustom Rekognition Amazon \(SDK\)](#). Hasil pengujian mencakup metrik yang tidak tersedia di konsol, seperti matriks kebingungan untuk hasil klasifikasi. Hasil pengujian dikembalikan dalam format berikut:

---

- Skor F1 — Nilai tunggal yang mewakili kinerja keseluruhan presisi dan penarikan untuk model. Untuk informasi selengkapnya, lihat [F1](#).
  - Lokasi file ringkasan — Ringkasan pengujian mencakup metrik evaluasi agregat untuk seluruh kumpulan data pengujian dan metrik untuk setiap label individual. [DescribeProjectVersions](#) mengembalikan bucket S3 dan lokasi folder dari file ringkasan. Untuk informasi selengkapnya, lihat [Mengakses file ringkasan model](#).
  - Lokasi snapshot manifes evaluasi — Snapshot berisi rincian tentang hasil tes, termasuk peringkat kepercayaan dan hasil tes klasifikasi biner, seperti positif palsu. [DescribeProjectVersions](#) mengembalikan bucket S3 dan lokasi folder dari file snapshot. Untuk informasi selengkapnya, lihat [Menafsirkan snapshot manifes evaluasi](#).
-

---

## Tingkatkan model Anda

---

Jika perbaikan diperlukan, Anda dapat menambahkan lebih banyak gambar pelatihan atau meningkatkan pelabelan kumpulan data. Untuk informasi selengkapnya, lihat [Meningkatkan model Label Kustom Rekognition Amazon](#). Anda juga dapat memberikan umpan balik tentang prediksi yang dibuat model Anda dan menggunakannya untuk melakukan perbaikan pada model Anda. Untuk informasi selengkapnya, lihat [Meningkatkan model dengan umpan balik Model](#).

---

### Tingkatkan model Anda (konsol)

---

Untuk menambahkan gambar ke kumpulan data, lihat [Menambahkan lebih banyak gambar ke kumpulan data](#). Untuk menambah atau mengubah label, lihat [the section called "Pelabelan gambar"](#).

---

Untuk melatih kembali model Anda, lihat [Melatih model \(Konsol\)](#).

---

### Tingkatkan model Anda (SDK)

---

Untuk menambahkan gambar ke kumpulan data atau mengubah label untuk gambar, gunakan API. `UpdateDatasetEntries` `UpdateDatasetEntries` memperbarui atau menambahkan baris JSON ke file manifest. Setiap baris JSON berisi informasi untuk satu gambar, seperti label yang ditetapkan atau informasi kotak pembatas. Untuk informasi selengkapnya, lihat [Menambahkan lebih banyak gambar \(SDK\)](#). Untuk melihat entri dalam kumpulan data, gunakan API. `ListDatasetEntries`

---

Untuk melatih kembali model Anda, lihat [Melatih model \(SDK\)](#).

---

## Mulai model Anda

---

Sebelum dapat menggunakan model, Anda memulai model dengan menggunakan konsol Amazon Rekognition Custom Labels atau API. `StartProjectVersion` Anda dikenakan biaya untuk jumlah waktu yang dijalankan model Anda. Untuk informasi selengkapnya, lihat [Menjalankan model terlatih](#).

---

### Mulai model Anda (konsol)

---

Untuk memulai model Anda menggunakan konsol, lihat [Memulai model Label Kustom Rekognition Amazon \(Konsol\)](#).

---

### Mulai model Anda

---

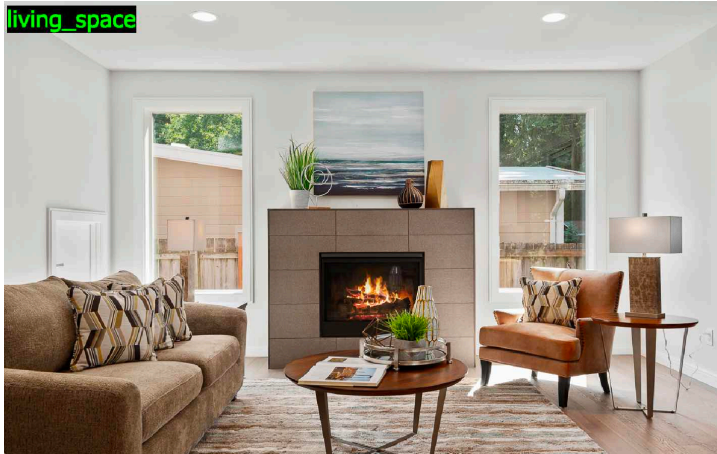
Anda memulai panggilan model Anda `StartProjectVersion`. Untuk informasi selengkapnya, lihat [Memulai model Label Kustom Rekognition Amazon \(SDK\)](#).

---

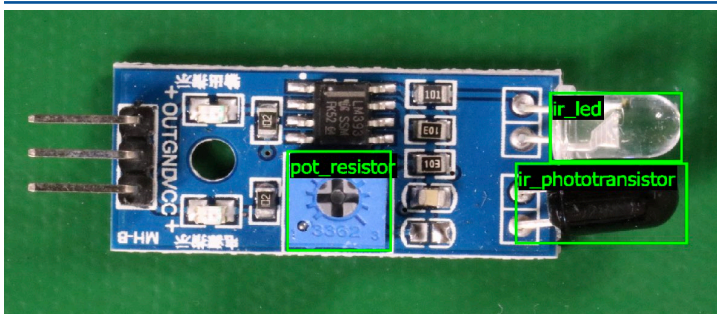
## Menganalisis gambar

Untuk menganalisis gambar dengan model Anda, Anda menggunakan DetectCustomLabels API. Anda dapat menentukan gambar lokal, atau gambar yang disimpan dalam ember S3. Operasi ini juga memerlukan Nama Sumber Daya Amazon (ARN) dari model yang ingin Anda gunakan.

Jika model Anda menemukan objek, adegan, dan konsep, responsnya menyertakan daftar label tingkat gambar yang ditemukan dalam gambar. Misalnya, gambar berikut menunjukkan label tingkat gambar yang ditemukan menggunakan proyek contoh Kamar.



Jika model menemukan lokasi objek, responsnya mencakup daftar kotak pembatas berlabel yang ditemukan pada gambar. Kotak pembatas mewakili lokasi objek pada gambar. Anda dapat menggunakan informasi kotak pembatas untuk menggambar kotak pembatas di sekitar objek. Misalnya, gambar berikut menunjukkan kotak pembatas di sekitar bagian papan sirkuit yang ditemukan menggunakan proyek contoh papan Sirkuit.



Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan model terlatih](#).

---

## Hentikan model Anda

---

Anda dikenakan biaya untuk waktu model Anda berjalan. Jika Anda tidak lagi menggunakan model, hentikan model dengan menggunakan konsol Amazon Rekognition Custom Labels, atau dengan menggunakan API. [StopProjectVersion](#) Untuk informasi selengkapnya, lihat [Menghentikan model Label Kustom Rekognition Amazon](#).

---

### Hentikan model Anda (Konsol)

---

Untuk menghentikan model yang sedang berjalan dengan konsol, lihat [Menghentikan model Label Kustom Rekognition Amazon \(Konsol\)](#).

---

### Hentikan model Anda (SDK)

---

Untuk menghentikan model yang sedang berjalan, panggil [StopProjectVersion](#). Lihat informasi yang lebih lengkap di [Menghentikan model Label Kustom Rekognition Amazon \(SDK\)](#).

---

## Melatih model Anda

### Proyek

- [CreateProject](#)— Membuat proyek Label Kustom Rekognition Amazon Anda yang merupakan pengelompokan sumber daya yang logis (gambar, Label, model) dan operasi (pelatihan, evaluasi, dan deteksi).
- [DeleteProject](#)— Menghapus proyek Label Kustom Rekognition Amazon.
- [DescribeProjects](#)— Mengembalikan daftar semua proyek Label Kustom Rekognition Amazon Anda.

### Kebijakan Proyek

- [PutProjectPolicy](#)— Melampirkan kebijakan proyek ke proyek Label Kustom Rekognition Amazon di akun terpercaya. AWS
- [ListProjectPolicies](#)— Mengembalikan daftar kebijakan proyek yang dilampirkan pada proyek.
- [DeleteProjectPolicy](#)— Menghapus kebijakan proyek yang ada.

## Set Data

- [CreateDataset](#)— Membuat kumpulan data Label Kustom Rekognition Amazon.
- [DeleteDataset](#)— Menghapus kumpulan data Label Kustom Rekognition Amazon.
- [DescribeDataset](#)— Menjelaskan kumpulan data Label Kustom Rekognition Amazon.
- [DistributeDatasetEntries](#)— Mendistribusikan entri (gambar) dalam kumpulan data pelatihan di seluruh kumpulan data pelatihan dan kumpulan data pengujian untuk suatu proyek.
- [ListDatasetEntries](#)— Mengembalikan daftar entri (gambar) dalam kumpulan data Label Kustom Rekognition Amazon.
- [ListDatasetLabels](#)— Mengembalikan daftar label yang ditetapkan ke kumpulan data Label Kustom Rekognition Amazon.
- [UpdateDatasetEntries](#)— Menambahkan atau memperbarui entri (gambar) dalam kumpulan data Label Kustom Rekognition Amazon.

## Model

- [CreateProjectVersion](#)— Melatih model Label Kustom Rekognition Amazon Anda.
- [CopyProjectVersion](#)— Salin model Label Kustom Rekognition Amazon Anda.
- [DeleteProjectVersion](#)— Menghapus model Label Kustom Rekognition Amazon.
- [DescribeProjectVersions](#)— Mengembalikan daftar semua model Label Kustom Rekognition Amazon dalam proyek tertentu.

## Tanda

- [TagResource](#)— Menambahkan satu atau lebih tag nilai kunci ke model Label Kustom Rekognition Amazon.
- [UntagResource](#)— Menghapus satu atau lebih tag dari model Label Kustom Rekognition Amazon.

## Menggunakan model Anda

- [DetectCustomLabels](#)— Menganalisis gambar dengan model label kustom Anda.
- [StartProjectVersion](#)— Memulai model label kustom Anda.
- [StopProjectVersion](#)— Menghentikan model label kustom Anda.

# Riwayat dokumen untuk Label Kustom Rekognition Amazon

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Pengembang Label Kustom Rekognition Amazon. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Pembaruan dokumentasi terbaru: 19 April 2023

Perubahan	Deskripsi	Tanggal
<a href="#">Ditambahkan topik durasi model</a>	Menunjukkan cara mendapatkan jumlah jam berjalan dan unit inferensi yang digunakan oleh model. Untuk informasi selengkapnya, lihat <a href="#">Melaporkan durasi berjalan dan unit inferensi yang digunakan</a> .	19 April 2023
<a href="#">Konten kumpulan data yang direorganisasi</a>	Memindahkan konten pembuatan file <a href="#">manifes ke file Manifest</a> . <a href="#">Memindahkan topik konversi kumpulan data ke Mengonversi format kumpulan data lain menjadi file manifes</a> .	Februari 20, 2023
<a href="#">Memperbarui panduan IAM untuk AWS WAF</a>	Memperbarui panduan untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi lebih lanjut, lihat <a href="#">Praktik terbaik keamanan di IAM</a> .	15 Februari 2023
<a href="#">Lihat matriks kebingungan untuk model klasifikasi</a>	Konsol Label Kustom Rekognition Amazon tidak menampilkan matriks kebingungan untuk model klasifikasi. Sebagai gantinya, Anda dapat menggunakan	4 Januari 2023

AWS SDK untuk mendapatkan dan menampilkan matriks kebingungan. Untuk informasi selengkapnya, lihat [Melihat matriks kebingungan untuk model](#).

[Contoh fungsi Lambda yang diperbarui](#)

Contoh fungsi Lambda sekarang menunjukkan cara menganalisis gambar yang diteruskan dari file lokal atau bucket Amazon S3. Untuk informasi selengkapnya, lihat [Menganalisis gambar dengan fungsi AWS Lambda](#).

Desember 2, 2022

[Amazon Rekognition Custom Labels sekarang dapat menyalin model terlatih](#)

Anda sekarang dapat menyalin model terlatih dari satu AWS akun ke AWS akun lain dalam AWS Wilayah yang sama. Untuk informasi selengkapnya, lihat [Menyalin model Label Kustom Amazon Rekognition](#) (SDK).

Agustus 16, 2022

[Amazon Rekognition Custom Labels kini dapat secara otomatis menskalakan unit inferensi](#).

Untuk membantu lonjakan permintaan, Amazon Rekognition Custom Labels kini dapat menskalakan jumlah unit inferensi yang digunakan model Anda. Untuk informasi selengkapnya, lihat [Menjalankan model Label Kustom Rekognition Amazon yang terlatih](#).

Agustus 16, 2022

---

<a href="#">Buat file manifes dari file CSV</a>	Anda sekarang dapat menyederhanakan pembuatan file manifes dengan menggunakan skrip yang membaca informasi klasifikasi gambar dari file CSV. Untuk informasi selengkapnya, lihat <a href="#">Membuat file manifes dari file CSV</a> .	2 Februari 2022
<a href="#">Amazon Rekognition Custom Labels sekarang mengelola kumpulan data dengan proyek</a>	Anda dapat menggunakan proyek untuk mengelola kumpulan data pelatihan dan pengujian yang Anda gunakan untuk membuat model. Untuk informasi selengkapnya, lihat <a href="#">Memahami Label Kustom Rekognition Amazon</a> .	November 1, 2021
<a href="#">Amazon Rekognition Custom Labels terintegrasi dengan AWS CloudFormation</a>	Anda dapat menggunakan CloudFormation untuk menyediakan dan mengonfigurasi proyek Label Kustom Amazon Rekognition. Untuk informasi selengkapnya, lihat <a href="#">Membuat proyek dengan AWS CloudFormation</a> .	21 Oktober 2021
<a href="#">Pengalaman memulai yang diperbarui</a>	Konsol Amazon Rekognition Custom Labels sekarang menyertakan video tutorial dan proyek contoh. Untuk informasi selengkapnya, lihat <a href="#">Memulai Label Kustom Amazon Rekognition</a> .	22 Juli 2021

<a href="#">Informasi terbaru tentang ambang batas dan menggunakan metrik</a>	Informasi tentang pengaturan nilai ambang yang diinginkan dengan menggunakan parameter <code>MinConfidence</code> input untuk <code>DetectCustomLabels</code> . Untuk informasi lebih lanjut, lihat <a href="#">Menganalisis gambar dengan model terlatih</a> .	8 Juni 2021
<a href="#">Menambahkan AWS KMS key dukungan</a>	Anda sekarang dapat menggunakan kunci KMS Anda sendiri untuk mengenkripsi pelatihan dan menguji gambar Anda. Untuk informasi selengkapnya, lihat <a href="#">Melatih model</a> .	19 Mei 2021
<a href="#">Ditambahkan tagging</a>	Amazon Rekognition Custom Labels sekarang mendukung penandaan. Anda dapat menggunakan tag untuk mengidentifikasi, mengatur, mencari, dan memfilter model Label Kustom Rekognition Amazon Anda. Untuk informasi selengkapnya, lihat <a href="#">Menandai model</a> .	25 Maret 2021
<a href="#">Topik persiapan yang diperbarui</a>	Informasi persiapan yang diperbarui tentang cara mengenkripsi file pelatihan. Untuk informasi selengkapnya, lihat <a href="#">Langkah 5: (Opsional) Mengenkripsi file pelatihan</a> .	18 Maret 2021

<a href="#">Menambahkan topik salinan dataset</a>	Informasi tentang cara menyalin kumpulan data ke AWS Wilayah yang berbeda. Untuk informasi selengkapnya, lihat <a href="#">Menyalin kumpulan data ke wilayah lain AWS</a> .	5 Maret 2021
<a href="#">Menambahkan topik transformasi manifes GroundTruth multi-label Amazon SageMaker AI</a>	Informasi tentang cara mengubah manifes format GroundTruth multi-label Amazon SageMaker AI menjadi file manifes format Label Kustom Rekognition Amazon. Untuk informasi selengkapnya, lihat <a href="#">Mengubah file manifes SageMaker AI Ground Truth multi-label</a> .	22 Februari 2021
<a href="#">Menambahkan informasi debugging untuk pelatihan model</a>	Anda sekarang dapat menggunakan manifes hasil validasi untuk mendapatkan informasi debugging mendalam tentang kesalahan pelatihan model. Untuk informasi selengkapnya, lihat <a href="#">Mendebug pelatihan model yang gagal</a> .	8 Oktober 2020
<a href="#">Ditambahkan COCO mengubah informasi dan contoh</a>	Informasi tentang cara mengubah kumpulan data format deteksi objek COCO menjadi file manifes Label Kustom Rekognition Amazon. Untuk informasi lebih lanjut, lihat <a href="#">Mengubah kumpulan data COCO</a> .	2 September 2020

---

<a href="#">Amazon Rekognition Custom Labels sekarang mendukung pelatihan objek tunggal</a>	Untuk membuat model Label Kustom Rekognition Amazon yang menemukan lokasi satu objek, kini Anda dapat membuat kumpulan data yang hanya memerlukan satu label. Untuk informasi lebih lanjut, lihat <a href="#">Menggambar kotak pembatas</a> .	25 Juni 2020
<a href="#">Operasi penghapusan proyek dan model ditambahkan</a>	Sekarang Anda dapat menghapus proyek dan model Label Kustom Rekogniti on Amazon dengan konsol dan dengan API. Untuk informasi selengkapnya, lihat <a href="#">Menghapus Model Label Kustom Rekognition Amazon dan Menghapus proyek Label Kustom Rekognition Amazon</a>	1 April 2020
<a href="#">Ditambahkan contoh Java</a>	Menambahkan contoh Java yang mencakup pembuatan proyek, pelatihan model, menjalankan model, dan analisis gambar.	13 Desember 2019
<a href="#">Fitur dan panduan baru</a>	Ini adalah rilis awal fitur Label Kustom Rekognition Amazon dan Panduan Pengembang Label Kustom Rekognition Amazon.	3 Desember 2019

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.