aws

Custom Labels 指南

Rekognition

Rekognition: Custom Labels 指南

Table of Contents

什么是 Amazon Rekognition Custom Labels？

借助 Amazon Rekognition Custom Labels，您可以根据自己的业务需求识别图像中的物体、徽标和场景。例如，您可以在社交媒体文章中查找您的徽标、在商店货架上识别您的产品、在装配线上对机器部件进行分类、区分运行状况良好的工厂和受感染的工厂，或在图像中检测动画角色。

开发用于分析图像的自定义模型是一项艰巨的任务，需要时间、专业知识和资源。这通常需要几个月才能完成。此外，可能还需要成千上万张手工标注的图像来为模型提供足够的数据来准确地做出决策。生成这些数据可能需要耗时几个月来进行收集，并且可能需要庞大的标注人员团队来准备数据以在机器学习中使用。

Amazon Rekognition Custom Labels 扩展了 Amazon Rekognition 的现有功能，这些功能已针对许多类别的数千万张图像进行过训练。您可以仅上传一组特定于您的使用场景的训练图像（通常是几百张或更少），而不是数千张图像。可使用易于使用的控制台执行此操作。如果您的图像已经带有标签，Amazon Rekognition Custom Labels 可以快速开始训练模型。如果没有，您可以直接在标注界面中为图像添加标签，或者也可以使用 Amazon SageMaker Ground Truth 添加标签。

在 Amazon Rekognition Custom Labels 从您的图像集开始训练后，它可在短短几个小时内为您生成自定义图像分析模型。在后台，Amazon Rekognition Custom Labels 会自动加载和检查训练数据，选择正确的机器学习算法，训练模型，并提供模型性能指标。然后，您就可以通过 Amazon Rekognition Custom Labels API 使用您的自定义模型，并将其集成到您的应用程序中。

主题

主要优势

简化了数据标注

Amazon Rekognition Custom Labels 控制台提供了一个可视化界面，让您能够快速、简单地标注图像。该界面允许您将标签应用于整个图像。您也可以使用带有点击拖动界面的边界框来识别和标注图像中的特定物体。或者，如果您的数据集很大，也可以使用 Amazon SageMaker Ground Truth 高效地批量标注图像。

自动机器学习

无需任何机器学习专业知识即可构建您的自定义模型。Amazon Rekognition Custom Labels 包含自动机器学习 (AutoML) 功能，可为您处理机器学习方面的工作。在您提供训练图像后，Amazon Rekognition Custom Labels 会自动加载和检查训练数据，选择正确的机器学习算法，训练模型，并提供模型性能指标。

简化了模型评估、推断和反馈

您可以根据自己的测试集评估自定义模型的性能。对于测试集中的每张图像，您都可以看到模型预测与分配的标签的并排比较。您还可以查看详细的性能指标，例如精度、召回率、F1 分数和置信度分数。您可以立即开始使用模型进行图像分析，也可以使用更多图像对新版本进行迭代和重新训练以提高性能。开始使用模型后，您可以跟踪预测，更正任何错误，并使用反馈数据重新训练新的模型版本以提高性能。

选择使用 Amazon Rekognition Custom Labels

Amazon Rekognition 提供了两个可用于在图像中查找标签（物体、场景和概念）的功能：Amazon Rekognition Custom Labels 和 Amazon Rekognition Image 标签检测功能。可根据以下信息确定应使用哪个功能。

Amazon Rekognition Image 标签检测

您可以使用 Amazon Rekognition Image 中的标签检测功能在图像和视频中批量识别、分类和搜索常见标签，而无需创建机器学习模型。例如，您可以轻松检测成千上万个常见物体，例如汽车和卡车、番茄、篮球和足球。

如果您的应用程序需要查找常见标签，建议您使用 Amazon Rekognition Image 标签检测功能，因为您无需训练模型。要获取 Amazon Rekognition Image 标签检测功能查找的标签列表，请参阅检测标签。

如果您的应用程序需要查找 Amazon Rekognition Image 标签检测功能无法找到的标签，例如装配线上的自定义机器零件，建议您使用 Amazon Rekognition Custom Labels。

Amazon Rekognition Custom Labels

您可以使用 Amazon Rekognition Custom Labels 轻松训练机器学习模型，用于在图像中查找符合您业务需求的独特标签（物体、徽标、场景和概念）。

Amazon Rekognition Custom Labels 可以对图像进行分类（图像级预测）或检测图像中的物体位置（物体/边界框级预测）。

Amazon Rekognition Custom Labels 在可以检测的物体和场景类型方面提供了更大的灵活性。例如，您可以使用 Amazon Rekognition Image 标签检测功能来查找植物和树叶。但如果要区分健康、受损和受感染的植物，则需要使用 Amazon Rekognition Custom Labels。

以下示例说明了如何使用 Amazon Rekognition Custom Labels。

- 识别球员球衣和头盔上的球队标志
- 在装配线上区分特定的机器零件或产品
- 识别媒体库中的卡通人物
- 在零售货架上找到特定品牌的产品
- 对农产品质量进行分类（例如腐烂的、成熟的或生的）

> **Note**
>
> Amazon Rekognition Custom Labels 不适用于分析人脸、检测文本或在图像中查找不安全的图像内容。要执行这些任务，可以使用 Amazon Rekognition Image。有关更多信息，请参阅什么是 Amazon Rekognition。

您是首次使用 Amazon Rekognition Custom Labels 的用户吗？

如果您是 Amazon Rekognition Custom Labels 的新用户，建议您按顺序阅读以下内容：

1. 设置 Amazon Rekognition Custom Labels：在本节中，您将设置您的账户详细信息。
2. 了解 Amazon Rekognition Custom Labels：在本节中，您将了解创建模型的工作流程。
3. Amazon Rekognition Custom Labels 入门：在本节中，您将使用由 Amazon Rekognition Custom Labels 创建的示例项目来训练模型。
4. 教程：对图像进行分类：在本节中，您将学习如何使用您创建的数据集训练对图像进行分类的模型。

设置 Amazon Rekognition Custom Labels

以下说明介绍了如何设置 Amazon Rekognition Custom Labels 控制台和 SDK。

请注意，您可通过以下浏览器使用 Amazon Rekognition Custom Labels 控制台：

- Chrome：版本 21 或更高版本
- Firefox：版本 27 或更高版本
- Microsoft Edge：版本 88 或更高版本
- Safari：版本 7 或更高版本 此外，使用 Safari 时，无法通过 Amazon Rekognition Custom Labels 控制台绘制边界框。有关更多信息，请参见 使用边界框标注物体。

首次使用 Amazon Rekognition Custom Labels 前，请完成以下任务：

主题

步骤 1：创建 AWS 账户

在此步骤中，您将创建一个 AWS 账户、创建管理用户，并了解如何授予对 AWS SDK 的编程访问权限。

主题

注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为管理用户分配管理访问权限，并且只使用根用户执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建管理用户

注册 AWS 账户 后，保护您的 AWS 账户根用户，启用 AWS IAM Identity Center，创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 对您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建管理用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为管理用户授予管理访问权限。

有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的使用默认 IAM Identity Center 目录 配置用户访问权限。

作为管理用户登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的登录 AWS 访问门户。

以编程方式访问

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI 以使用 AWS IAM Identity Center。 • 有关 AWS软件开发工具包、工具和 AWS API 的更多信息，请参阅《AWS软件开发工具包和工具参考指南》中的 IAM Identity Center 身份验证。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用临时凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。	按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 IAM 用户凭证进行身份验证。 • 有关 AWS 软件开发工具包和工具的更多信息，请参阅《AWS 软件开发工具包和工具参考指南》中的使用长期凭证进行身份验证。 • 有关 AWS API 的更多信息，请参阅《IAM 用户指南》中的管理 IAM 用户的访问密钥。

步骤 2：设置 Amazon Rekognition Custom Labels 控制台权限

要使用 Amazon Rekognition 控制台，您必须具有相应的权限。如果想将训练文件存储在控制台存储桶以外的存储桶中，则需要额外的权限。

主题

- 允许访问控制台
- 访问外部 Amazon S3 存储桶
- 分配权限

允许访问控制台

要使用亚马逊 Rekognition 自定义标签控制台，您需要以下涵盖亚马逊 S3、G SageMaker round Truth 和亚马逊 Rekognition 自定义标签的 IAM 政策。有关分配权限的信息，请参阅分配权限。

```
{
    "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": "*"
        }
    ]
}
```

访问外部 Amazon S3 存储桶

首次在新 AWS 区域中打开 Amazon Rekognition Custom Labels 控制台时，Amazon Rekognition Custom Labels 会创建一个用于存储项目文件的存储桶（控制台存储桶）。或者，您也可以使用自己的 Amazon S3 存储桶（外部存储桶）将图像或清单文件上传到控制台。要使用外部存储桶，请将以下策略块添加到先前的策略中。将 `my-bucket` 替换为存储桶名称。

```
        {
            "Sid": "s3ExternalBucketPolicies",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket*"
            ]
```

```
        }
```

分配权限

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和群组：

  创建权限集。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以代入的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户群组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

步骤 3：创建控制台存储桶

您使用 Amazon Rekognition Custom Labels 项目来创建和管理您的模型。首次在新 AWS 区域中打开 Amazon Rekognition Custom Labels 控制台时，Amazon Rekognition Custom Labels 会创建一个用于存储项目的 Amazon S3 存储桶（控制台存储桶）。您应记下该控制台存储桶的名称以便之后参考，因为您可能需要在 AWS SDK 操作或控制台任务（如创建数据集）中使用该存储桶名称。

该存储桶名称的格式为 `custom-labels-console-<区域>-<随机值>`。该随机值可确保存储桶名称不会重复。

创建控制台存储桶

1. 请确保用户具有正确的权限。有关更多信息，请参见 允许访问控制台。
2. 通过以下网址登录 AWS Management Console 并打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
3. 选择开始。
4. 如果这是您首次在当前 AWS 区域中打开控制台，请在首次设置对话框中执行以下操作：

   a. 记下所显示的 Amazon S3 存储桶的名称。稍后您将需要此信息。

b. 选择创建 S3 存储桶，让 Amazon Rekognition Custom Labels 为您创建一个 Amazon S3 存储桶（控制台存储桶）。

5. 关闭浏览器窗口。

步骤 4：设置 AWS CLI 和 AWS SDK

您可以将 Amazon Rekognition Custom Labels 与 AWS Command Line Interface (AWS CLI) 和 AWS SDK 一起使用。如果需要从终端运行 Amazon Rekognition Custom Labels 操作，请安装 AWS CLI。如果要创建应用程序，请下载与您所使用的编程语言对应的 AWS SDK。

主题

安装 AWS SDK

按照以下步骤下载和配置 AWS SDK。

设置 AWS CLI 和 AWS SDK

- 下载并安装 AWS CLI 和您要使用的 AWS SDK。本指南提供了分别适用于 AWS CLI、Java 和 Python 的示例。有关安装 AWS SDK 的信息，请参阅用于 Amazon Web Services 的工具。

授予编程式访问权限

您可以在本地计算机或其他 AWS 环境（如 Amazon Elastic Compute Cloud 实例）上运行 AWS CLI 和本指南中的代码示例。要运行这些示例，您需要授予对示例使用的 AWS SDK 操作的访问权限。

主题

在本地计算机上运行代码

要在本地计算机上运行代码，我们建议使用短期凭证，向用户授予对 AWS SDK 操作的访问权限。有关在本地计算机上运行 AWS CLI 和代码示例的具体信息，请参阅在本地计算机上使用配置文件。

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI 以使用 AWS IAM Identity Center。 • 有关 AWS软件开发工具包、工具和 AWS API 的更多信息，请参阅《AWS软件开发工具包和工具参考指南》中的 IAM Identity Center 身份验证。
IAM	使用临时凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。	按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中

哪个用户需要编程式访问权限？	目的	方式
		的使用 IAM 用户凭证进行身份验证。 • 有关 AWS 软件开发工具包和工具的更多信息，请参阅《AWS 软件开发工具包和工具参考指南》中的使用长期凭证进行身份验证。 • 有关 AWS API 的更多信息，请参阅《IAM 用户指南》中的管理 IAM 用户的访问密钥。

在本地计算机上使用配置文件

您可以使用在在本地计算机上运行代码中创建的短期凭证，运行 AWS CLI 和本指南中的代码示例。为了获取凭证和其他设置信息，这些示例使用名为 `custom-labels-access` 的配置文件。例如：

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

配置文件所代表的用户必须有权调用 Amazon Rekognition Custom Labels SDK 操作，以及示例所需的其他 AWS SDK 操作。有关更多信息，请参见 设置 SDK 权限。若要分配权限，请参阅设置 SDK 权限。

要创建能够与 AWS CLI 和代码示例配合使用的配置文件，请选择以下选项之一。确保您创建的配置文件的名称为 `custom-labels-access`。

- 由 IAM 管理的用户 — 按照切换到 IAM 角色 (AWS CLI) 部分的说明进行操作。
- 员工身份（由 AWS IAM Identity Center 管理的用户）— 按照配置 AWS CLI 以使用 AWS IAM Identity Center 部分的说明进行操作。对于这些代码示例，我们建议使用集成式开发环境 (IDE)，该环境支持 AWS Toolkit，可通过 IAM Identity Center 实现身份验证。有关 Java 示例，请参阅使用 Java 开始构建。有关 Python 示例，请参阅使用 Python 开始构建。有关更多信息，请参阅 IAM Identity Center 凭证。

> **Note**
>
> 您可以使用代码获取短期凭证。有关更多信息，请参阅切换到 IAM 角色 (AWS API)。对于 IAM Identity Center，请按照获取用于 CLI 访问的 IAM 角色凭证部分的说明操作，获取角色的短期凭证。

在 AWS 环境中运行代码

您不应使用用户凭证在 AWS 环境中签署 AWS SDK 调用，例如在 AWS Lambda 函数中运行的生产代码。相反，您应该配置一个角色来定义代码所需的权限。然后，将该角色附加到运行代码的环境。关于如何附加角色和提供可用的临时凭证，取决于运行代码的环境：

- AWS Lambda 函数 — 使用当您的函数承担 Lambda 函数的执行角色时，Lambda 自动为之提供的临时凭证。这些凭证在 Lambda 环境变量中可用。您不需要指定配置文件。有关更多信息，请参阅 Lambda 执行角色。
- Amazon EC2 — 使用 Amazon EC2 实例元数据端点凭证提供程序。该提供程序会使用您附加到 Amazon EC2 实例的 Amazon EC2 实例配置文件，自动为您生成和刷新凭证。有关更多信息，请参阅使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。
- Amazon Elastic Container Service — 使用 Container 凭证提供程序。Amazon ECS 会向元数据端点发送和刷新凭证。您指定的任务 IAM 角色会提供一项策略，用于管理您的应用程序所使用的凭证。有关更多信息，请参阅与 AWS 服务交互。

有关凭证提供程序的更多信息，请参阅标准化凭证提供程序。

设置 SDK 权限

要使用 Amazon Rekognition Custom Labels SDK 操作，您需要具有 Amazon Rekognition Custom Labels API 和用于模型训练的 Amazon S3 存储桶的访问权限。

主题

- 授予 SDK 操作权限
- 关于使用 AWS SDK 的策略更新
- 分配权限

授予 SDK 操作权限

建议仅授予执行任务所必需的权限（最低权限）。例如，要拨打电话 DetectCustomLabels，您需要获得执行权限rekognition:DetectCustomLabels。要查找操作的权限，请查看 API 参考。

刚开始使用应用程序时，您可能不知道具体需要哪些权限，因此可以从广泛权限入手。AWS 托管式策略可以提供一些权限来帮助您入门。您可以使用 AmazonRekognitionCustomLabelsFullAccess AWS 托管式策略来获取对 Amazon Rekognition Custom Labels API 的完整访问权限。有关更多信息，请参阅 AWS 托管策略： AmazonRekognitionCustomLabelsFullAccess。如果知道应用程序所需的权限，则可定义特定于您的使用场景的客户管理型策略，从而进一步减少权限。有关更多信息，请参阅客户管理型策略。

若要分配权限，请参阅分配权限。

关于使用 AWS SDK 的策略更新

将 AWS SDK 与最新版本的 Amazon Rekognition Custom Labels 一起使用时，您无需再向 Amazon Rekognition Custom Labels 授予访问包含您的训练和测试图像的 Amazon S3 存储桶的权限。如果之前添加了权限，也不需要将其移除。您可以选择从主体的服务为 rekognition.amazonaws.com 的存储桶中移除任何策略。例如：

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

有关更多信息，请参阅使用存储桶策略。

分配权限

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和群组：

  创建权限集。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以代入的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。

- （不推荐使用）将策略直接附加到用户或将用户添加到用户群组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

调用 Amazon Rekognition Custom Labels 操作

运行以下代码，确认您可以调用 Amazon Rekognition Custom Labels API。该代码列出了您的 AWS 账户在当前 AWS 区域中的项目。如果您之前未创建项目，则响应为空，但确实会确认您可以调用 `DescribeProjects` 操作。

通常，调用示例函数需要 AWS SDK Rekognition 客户端和任何其他所需的参数。AWS SDK 客户端会在主函数中声明。

如果该代码失败，请检查所用的用户是否具有正确的权限。此外还应检查您所使用的 AWS 区域，因为 Amazon Rekognition Custom Labels 并非在所有 AWS 地区都可用。

调用 Amazon Rekognition Custom Labels 操作

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参见 步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码查看您的项目。

   CLI

   使用 `describe-projects` 命令列出您账户中的项目。

   ```
   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);
        }

    }
```

```
}
```

步骤 5：（可选）加密训练文件

您可以选择以下选项之一来加密控制台存储桶或外部 Amazon S3 存储桶中的 Amazon Rekognition Custom Labels 清单文件和图像文件。

- 使用 Amazon S3 密钥 (SSE-S3)。
- 使用您的 AWS KMS key。

> **Note**
>
> 调用 IAM 主体需要解密文件的权限。有关更多信息，请参见 解密使用 AWS Key Management Service 加密的文件。

有关如何加密 Amazon S3 存储桶的信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

解密使用 AWS Key Management Service 加密的文件

如果使用 AWS Key Management Service (KMS) 加密 Amazon Rekognition Custom Labels 清单文件和图像文件，请将调用 Amazon Rekognition Custom Labels 的 IAM 主体添加到 KMS 密钥的密钥策略中。这样做可以让 Amazon Rekognition Custom Labels 在训练之前解密清单和图像文件。有关更多信息，请参阅我的 Amazon S3 存储桶使用自定义 AWS KMS 密钥进行默认加密。如何允许用户从存储桶下载内容以及上传内容到存储桶？

IAM 主体需要对 KMS 密钥具有以下权限。

- kms: GenerateDataKey
- kms:Decrypt

有关更多信息，请参阅使用存储在 AWS Key Management Service 中的 KMS 密钥通过服务器端加密 (SSE-KMS) 保护数据。

对复制的训练和测试图像进行加密

为了训练您的模型，Amazon Rekognition Custom Labels 会复制您的源训练图像和测试图像。默认情况下，复制的图像使用 AWS 拥有和管理的密钥进行静态加密。您也可以选择使用自己的 AWS KMS key。如果使用自己的 KMS 密钥，则需要对该 KMS 密钥具有以下权限。

- kms: CreateGrant
- kms: DescribeKey

在使用控制台训练模型或者在调用 `CreateProjectVersion` 操作时，可以选择指定 KMS 密钥。所使用的 KMS 密钥不必与用于加密 Amazon S3 存储桶中的清单文件和图像文件的 KMS 密钥相同。有关更多信息，请参见 步骤 5：（可选）加密训练文件。

有关更多信息，请参阅 AWS Key Management Service 概念。源图像不受影响。

有关训练模型的信息，请参阅训练 Amazon Rekognition Custom Labels 模型。

步骤 6：（可选）将先前的数据集与新项目关联

Amazon Rekognition Custom Labels 现在可以通过项目管理数据集。您创建的早期（先前）数据集是只读的，必须先与项目关联才能使用。通过控制台打开项目的详细信息页面时，我们会自动将训练项目最新版本模型的数据集与项目关联起来。如果使用的是 AWS SDK，则不会自动将数据集与项目关联。

未关联的先前数据集从未用于训练过模型，或者曾用于训练过模型的先前版本。“之前的数据集”页面会显示所有关联和未关联的数据集。

要使用未关联的先前数据集，请在“之前的数据集”页面上创建一个新项目。该数据集将成为该新项目的训练数据集。您也可以为已关联的数据集创建项目，因为先前的数据集可以有多个关联。

将先前的数据集与新项目相关联

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。
3. 在左侧导航窗格中，选择之前的数据集。
4. 在数据集视图中，选择一个要与项目关联的先前数据集。
5. 选择使用数据集创建项目。
6. 在创建项目页面上，于项目名称中输入新项目的名称。

7. 选择创建项目，创建项目。该项目可能需要一段时间才能创建。
8. 保存项目。有关更多信息，请参见 了解 Amazon Rekognition Custom Labels。

使用先前的数据集作为测试数据集

要将先前数据集作为现有项目的测试数据集，需要先将该先前数据集与新项目相关联。然后，再将该新项目的训练数据集复制到该现有项目的测试数据集中。

使用先前的数据集作为测试数据集

1. 按照步骤 6：（可选）将先前的数据集与新项目关联中的说明将先前的数据集与新项目相关联。
2. 使用从该新项目中复制的训练数据集，在现有项目中创建测试数据集。有关更多信息，请参见 现有数据集。
3. 按照删除 Amazon Rekognition Custom Labels 项目（控制台）中的说明将该新项目删除。

或者，也可以使用先前数据集的清单文件来创建测试数据集。有关更多信息，请参阅 创建清单文件。

了解 Amazon Rekognition Custom Labels

本节简要介绍通过控制台和 AWS SDK 训练和使用 Amazon Rekognition Custom Labels 模型的工作流程。

> **Note**
>
> Amazon Rekognition Custom Labels 现在可以管理项目内的数据集。您可以使用控制台和 AWS SDK 为项目创建数据集。如果之前使用过 Amazon Rekognition Custom Labels，则可能需要将旧数据集与新项目关联。有关更多信息，请参阅步骤 6：（可选）将先前的数据集与新项目关联。

主题

确定您的模型类型

您首先需要决定要训练哪种类型的模型，这取决于您的业务目标。例如，您可以训练模型在社交媒体文章中查找您的徽标、在商店货架上识别您的产品，或者在装配线上对机器部件进行分类。

Amazon Rekognition Custom Labels 可以训练以下类型的模型：

- 查找物体、场景和概念
- 查找物体位置
- 查找品牌位置

为帮助您决定要训练的模型类型，Amazon Rekognition Custom Labels 提供了您可以使用的示例项目。有关更多信息，请参阅Amazon Rekognition Custom Labels 入门。

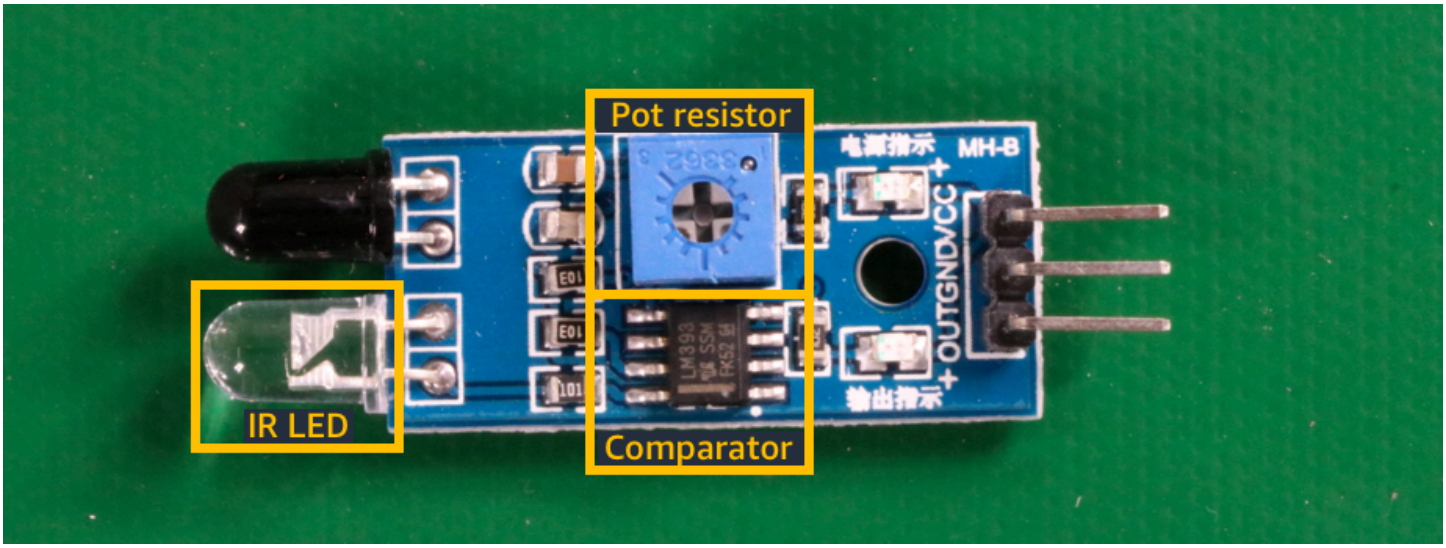
查找物体、场景和概念

该模型可以预测与整张图像关联的物体、场景和概念的分类。例如，可以训练一个模型来确定图像是否包含旅游景点。如需查看示例项目，请参阅图像分类。

或者，也可以训练模型来将图像分为多个类别。例如，上面这张图像可能包含天空颜色、反射或湖泊等类别。如需查看示例项目，请参阅多标签图像分类。

查找物体位置

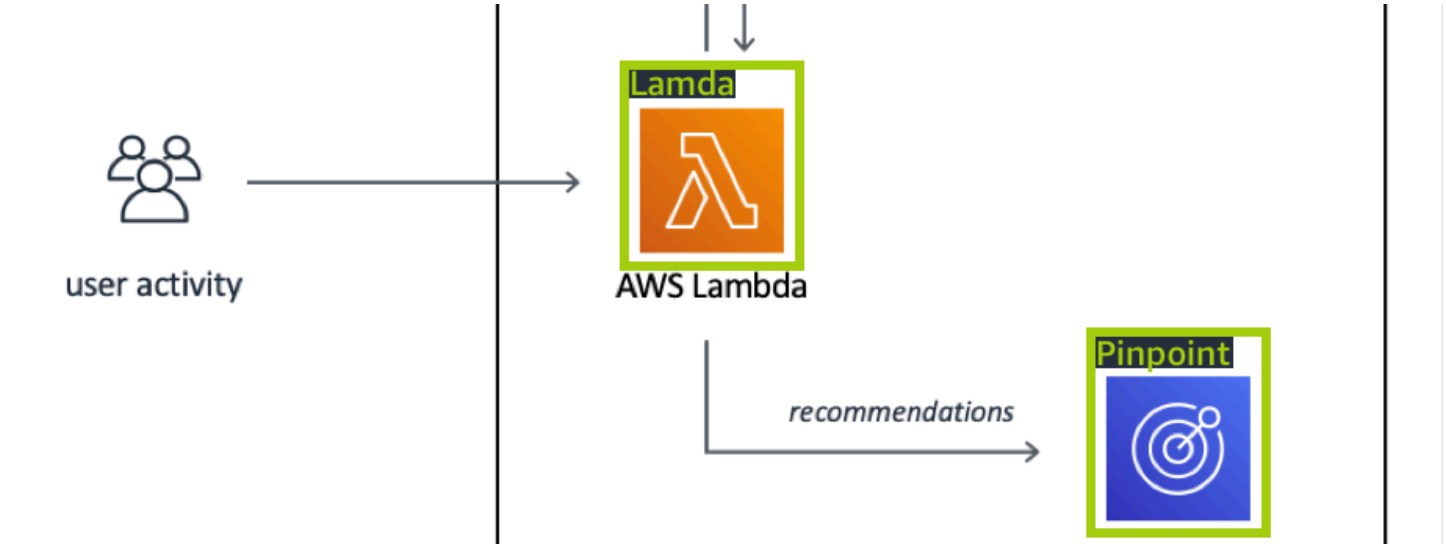
该模型可预测物体在图像上的位置。预测包括物体位置的边界框信息以及用于标识边界框内物体的标签。例如，下图显示了电路板各个零件（例如 comparator 或 pot resistor）周围的边界框。



物体定位示例项目展示了 Amazon Rekognition Custom Labels 如何使用带标签的边界框来训练查找物体位置的模型。

查找品牌位置

Amazon Rekognition Custom Labels 可以训练模型在图像上查找品牌（例如徽标）的位置。预测包括品牌位置的边界框信息以及用于标识边界框内物体的标签。如需查看示例项目，请参阅品牌检测。



创建模型

创建模型的步骤包括创建项目、创建训练和测试数据集以及训练模型。

创建项目

Amazon Rekognition Custom Labels 项目是指创建和管理模型所需的一组资源。项目用于管理以下内容：

- 数据集：用于训练模型的图像和图像标签。一个项目具有一个训练数据集和一个测试数据集。
- 模型：训练来查找符合您业务需求的独特概念、场景和物体的软件。一个项目可以有多个版本的模型。

建议将一个项目用于单个使用场景，例如在电路板上查找电路板零件。

可以使用 Amazon Rekognition Custom Labels 控制台和 CreateProject API 创建项目。有关更多信息，请参阅创建项目。

创建训练和测试数据集

数据集是由图像和描述这些图像的标签组成的集合。在您的项目中，您可以创建一个训练数据集和一个测试数据集，Amazon Rekognition Custom Labels 将使用这些数据集来训练和测试您的模型。

标签用于标识图像中的物体、场景、概念或物体周围的边界框。标签要么分配给整个图像（图像级），要么分配给围绕图像上的物体的边界框。

> **Important**
>
> 如何为数据集中的图像添加标签决定了 Amazon Rekognition Custom Labels 创建的模型的类型。例如，要训练查找物体、场景和概念的模型，请为训练和测试数据集中的图像分配图像级标签。有关更多信息，请参阅确定数据集用途。

图像必须采用 PNG 和 JPEG 格式，并且您应遵循输入图像的建议。有关更多信息，请参阅准备图像。

创建训练和测试数据集（控制台）

可以使用单个数据集或单独的训练数据集和测试数据集开始项目。如果从单个数据集开始，Amazon Rekognition Custom Labels 会在训练期间拆分该数据集，来为项目创建训练数据集 (80%) 和测试数据集 (20%)。如果想让 Amazon Rekognition Custom Labels 决定使用哪些图像进行训练和哪些图像进行测试，请使用单个数据集开始项目。为了能够完全控制训练、测试和性能调整，建议您使用单独的训练数据集和测试数据集开始您的项目。

要为项目创建数据集，请通过以下方式之一导入图像：

- 从本地计算机导入图像。
- 从 S3 存储桶导入图像。Amazon Rekognition Custom Labels 可以使用包含图像的文件夹名称为图像添加标签。
- 导入 Amazon SageMaker Ground Truth 清单文件。
- 复制现有的 Amazon Rekognition Custom Labels 数据集。

有关更多信息，请参阅使用图像创建训练和测试数据集。

根据导入图像的方式，您的图像可能没有标签。例如，从本地计算机导入的图像就没有标签。从 Amazon SageMaker Ground Truth 清单文件导入的图像则带有标签。您可以使用 Amazon Rekognition Custom Labels 控制台添加、更改和分配标签。有关更多信息，请参阅标注图像。

要使用控制台创建训练和测试数据集，请参阅使用图像创建训练和测试数据集。如需查看包含创建训练和测试数据集相关内容的教程，请参阅教程：对图像进行分类。

创建训练和测试数据集 (SDK)

要创建训练和测试数据集，请使用 `CreateDataset` API。您可以使用 Amazon SageMaker 格式的清单文件或通过复制现有的 Amazon Rekognition Custom Labels 数据集来创建数据集。有关更多信息，请参阅创建训练和测试数据集 (SDK)。如有必要，您可以创建自己的清单文件。有关更多信息，请参阅the section called “创建清单文件”。

训练模型

使用训练数据集训练您的模型。每次训练模型时都会创建一个新的模型版本。训练期间，Amazon Rekognition Custom Labels 会测试训练后的模型的性能。您可以使用测试结果来评估和改进模型。训练需要一段时间才能完成。您只需为成功的模型训练付费。有关更多信息，请参阅训练 Amazon Rekognition Custom Labels 模型。如果模型训练失败，Amazon Rekognition Custom Labels 会提供调试信息供您使用。有关更多信息，请参阅调试失败的模型训练。

训练模型（控制台）

要使用控制台训练模型，请参阅训练模型（控制台）。

训练模型 (SDK)

可通过调用 CreateProjectVersion 来训练 Amazon Rekognition Custom Labels 模型。有关更多信息，请参阅训练模型 (SDK)。

改进模型

测试期间，Amazon Rekognition Custom Labels 会创建评估指标，您可以使用这些指标来改进训练后的模型。

评估模型

使用在测试期间创建的性能指标，评估模型的性能。F1、精度和召回率等性能指标可让您了解训练后的模型的性能，并决定是否已准备好在生产中使用它。有关更多信息，请参阅评估模型的指标。

评估模型（控制台）

如需查看性能指标，请参阅获取评估指标（控制台）。

评估模型 (SDK)

要获得性能指标，可调用 DescribeProjectVersions 来获取测试结果。有关更多信息，请参阅获取 Amazon Rekognition Custom Labels 评估指标 (SDK)。测试结果包含控制台中未提供的指标，例如分类结果的混淆矩阵。测试结果以下列格式返回：

- F1 分数：代表模型的精度和召回率整体表现的单个值。有关更多信息，请参阅F1。
- 摘要文件位置：测试摘要包含整个测试数据集的综合评估指标和每个标签的指标。`DescribeProjectVersions` 会返回摘要文件所在的 S3 存储桶和文件夹位置。有关更多信息，请参阅摘要文件。
- 评估清单快照位置：快照包含有关测试结果的详细信息，包括置信度评分和二进制分类测试的结果，例如假正例。`DescribeProjectVersions` 会返回快照文件所在的 S3 存储桶和文件夹位置。有关更多信息，请参阅评估清单快照。

改进模型

如需进行改进，可以添加更多训练图像或改进数据集标注方式。有关更多信息，请参阅改进 Amazon Rekognition Custom Labels 模型。您还可以就模型所做的预测提供反馈，并据其改进模型。有关更多信息，请参阅模型反馈解决方案。

改进模型（控制台）

要向数据集中添加图像，请参阅向数据集中添加更多图像。要添加或更改标签，请参阅the section called “标注图像”。

要重新训练模型，请参阅训练模型（控制台）。

改进模型 (SDK)

要向数据集中添加图像或更改图像的标注方式，请使用 UpdateDatasetEntries API。UpdateDatasetEntries 会在清单文件中更新或添加 JSON 行。每个 JSON 行都包含单张图像的信息，例如分配的标签或边界框信息。有关更多信息，请参阅添加更多图像 (SDK)。要查看数据集中的条目，请使用 ListDatasetEntries API。

要重新训练模型，请参阅训练模型 (SDK)。

启动模型

在使用模型之前，需要先使用 Amazon Rekognition Custom Labels 控制台或 StartProjectVersion API 启动模型。您将根据模型运行时间付费。有关更多信息，请参阅运行经过训练的模型。

启动模型（控制台）

要使用控制台启动模型，请参阅启动 Amazon Rekognition Custom Labels 模型（控制台）。

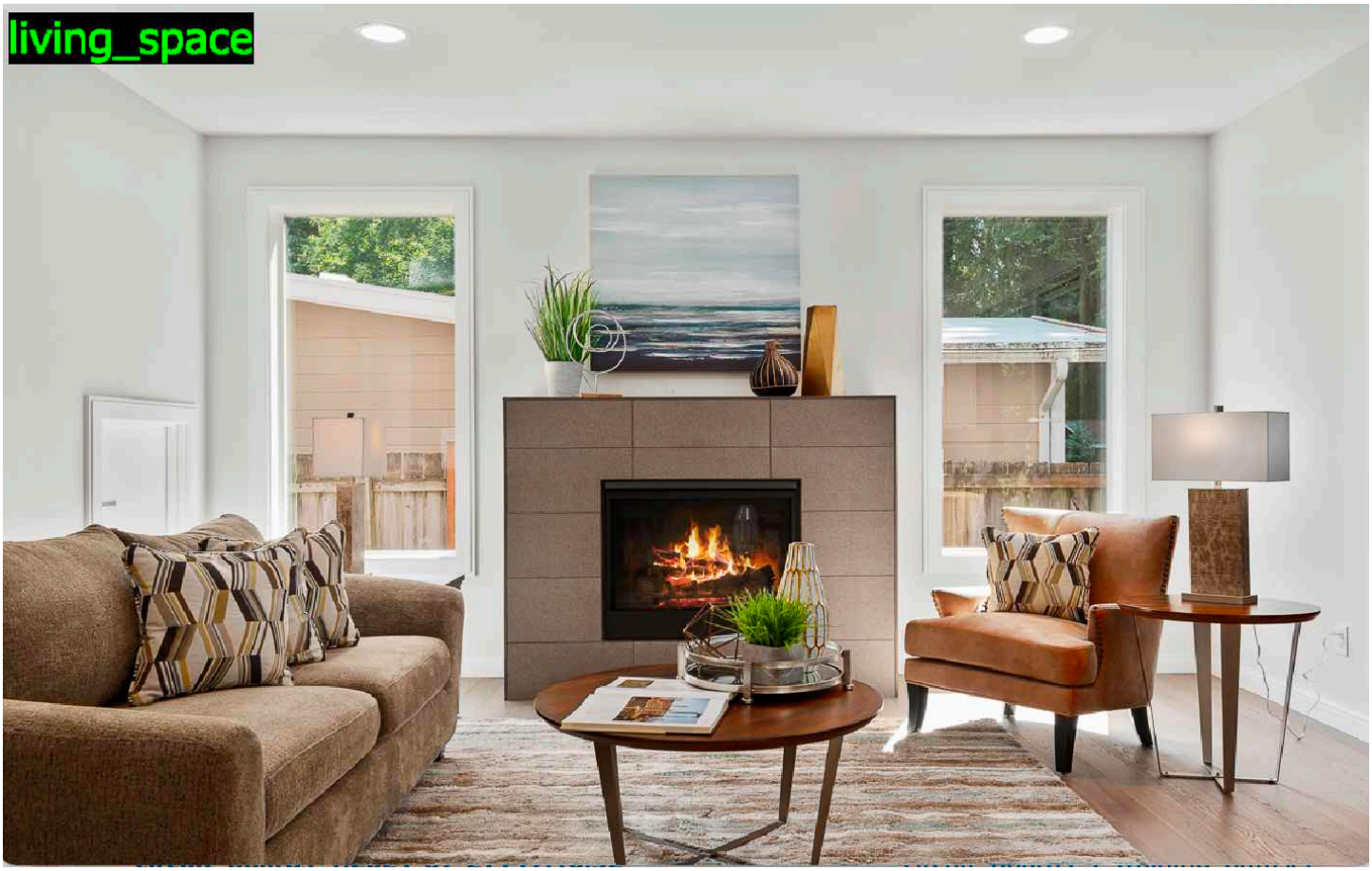
启动模型

您可以通过调用 StartProjectVersion 来启动模型。有关更多信息，请参阅启动 Amazon Rekognition Custom Labels 模型 (SDK)。

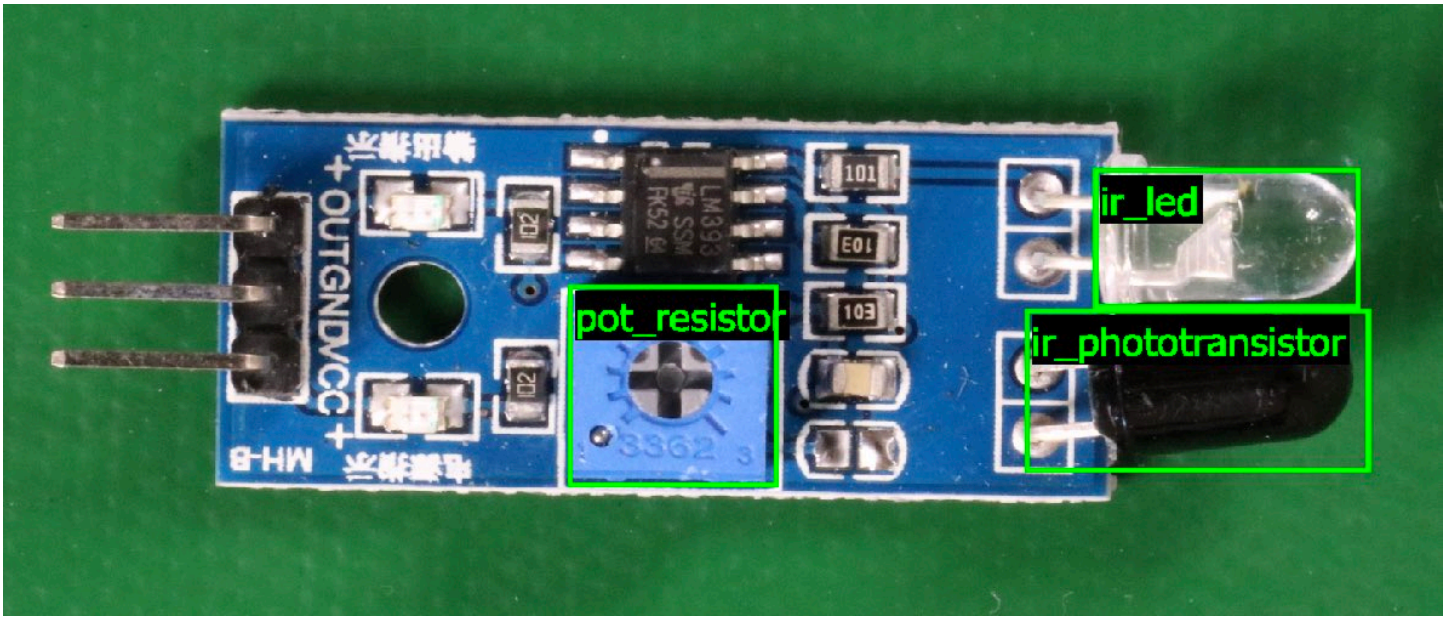
分析图像

要使用模型分析图像，可以使用 DetectCustomLabels API。可以指定本地图像或存储在 S3 存储桶中的图像。该操作还要求提供要使用的模型的 Amazon 资源名称 (ARN)。

如果模型用于查找物体、场景和概念，则响应会包含在图像中找到的图像级标签的列表。例如，下图显示了使用房间示例项目找到的图像级标签。



如果模型用于查找物体位置，则响应将包含在图像中找到的带标签的边界框列表。边界框表示物体在图像上的位置。您可以使用边界框信息在物体周围绘制边界框。例如，下图显示了使用电路板示例项目找到的电路板零件周围的边界框。



有关更多信息，请参阅使用经过训练的模型分析图像。

停止模型

您将根据模型运行时间付费。如果您不再使用模型，请使用 Amazon Rekognition Custom Labels 控制台或 `StopProjectVersion` API 停止该模型。有关更多信息，请参阅停止 Amazon Rekognition Custom Labels 模型。

停止模型（控制台）

要使用控制台停止正在运行的模型，请参阅停止 Amazon Rekognition Custom Labels 模型（控制台）。

停止模型 (SDK)

要停止正在运行的模型，请调用 StopProjectVersion。有关更多信息，请参阅停止 Amazon Rekognition Custom Labels 模型 (SDK)。

Amazon Rekognition Custom Labels 入门

在开始阅读这些入门说明之前，建议您先阅读了解 Amazon Rekognition Custom Labels。

您可以使用 Amazon Rekognition Custom Labels 来训练机器学习模型。经过训练的模型可用于分析图像，以查找符合您业务需求的独特物体、场景和概念。例如，您可以训练模型以对房屋图像进行分类，或者在印刷电路板上查找电子零件的位置。

为帮助您入门，Amazon Rekognition Custom Labels 提供了教程视频和示例项目。

> **Note**
>
> 有关 Amazon Rekognition Custom Labels 支持的 AWS 区域和端点的信息，请参阅 Rekognition 端点和配额。

教程视频

这些视频向您展示了如何使用 Amazon Rekognition Custom Labels 来训练和使用模型。

观看教程视频

1. 通过以下网址登录 AWS Management Console 并打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。如果没有看到使用自定义标签，请检查您使用的 AWS 区域是否支持 Amazon Rekognition Custom Labels。
3. 从导航窗格中，选择开始。
4. 在什么是 Amazon Rekognition Custom Labels？中，选择观看概述视频。
5. 在导航窗格中，选择教程。
6. 在教程页面上，选择要观看的教程视频。

示例项目

Amazon Rekognition Custom Labels 提供了以下示例项目。

图像分类

该图像分类项目 (Rooms) 会训练一种模型，用于在图像中查找一个或多个家庭位置，例如 backyard、kitchen 和 patio。训练和测试图像代表单个位置。每张图像都带有单个图像级标签，例如 kitchen、patio 或 living_space。分析图像后，经过训练的模型会返回用于训练的图像级标签集中的一个或多个匹配标签。例如，模型可能会在下图中找到标签 living_space。有关更多信息，请参阅查找物体、场景和概念。

多标签图像分类

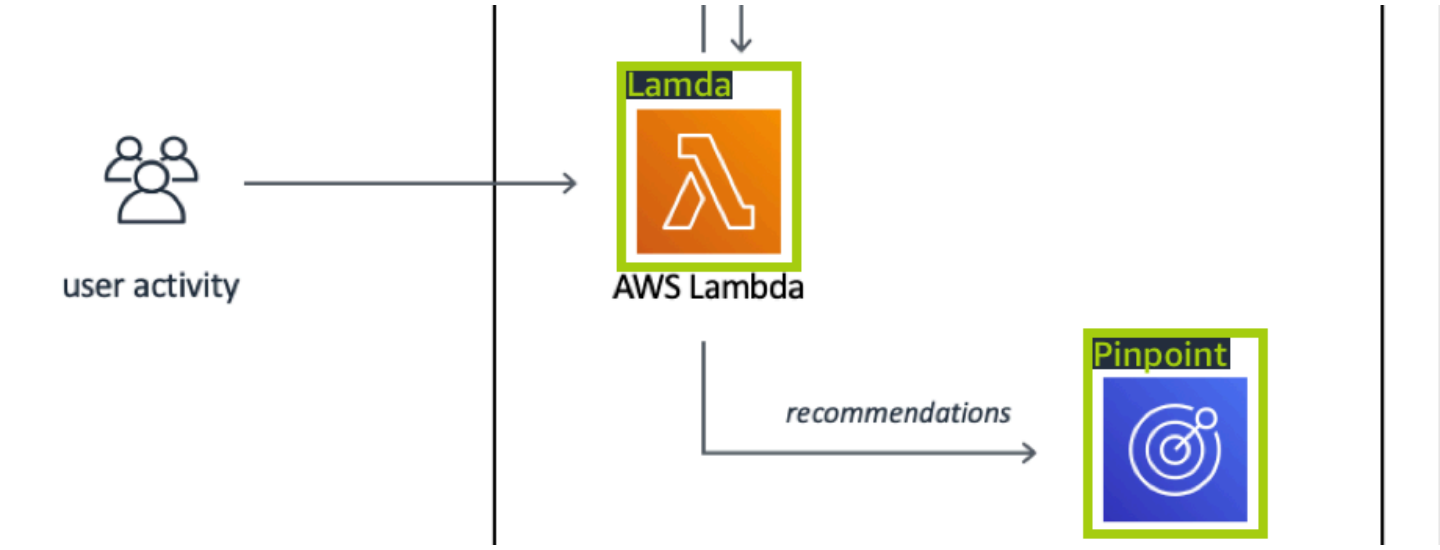
多标签图像分类项目 (Flowers) 训练了一个模型，用于将花朵图像分类为三个概念（花朵类型、树叶存在和生长阶段）。

训练和测试图像对每个概念都有相应的图像级标签，例如 camellia 代表花朵类型、with_leaves 代表有叶子的花，fully_grown 代表完全生长的花。

分析图像后，经过训练的模型会返回用于训练的图像级标签集中的匹配标签。例如，对于下图，该模型返回了标签 mediterranean_spurge 和 with_leaves。有关更多信息，请参阅查找物体、场景和概念。

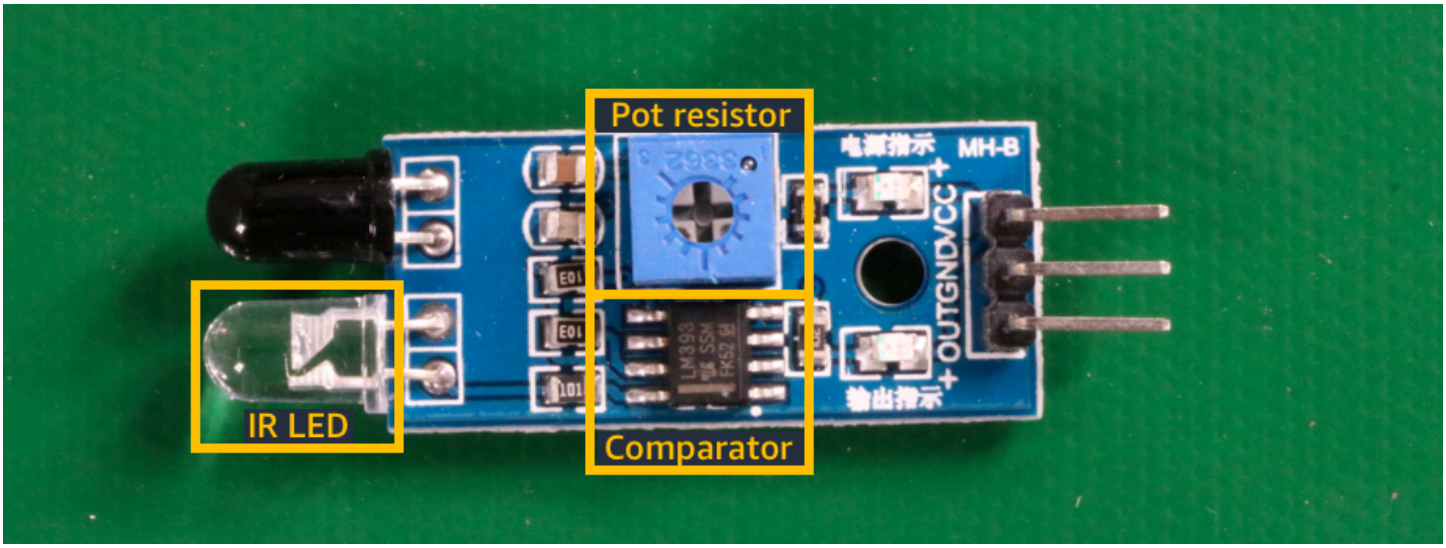
品牌检测

品牌检测项目 (Logos) 训练了一个模型，用于查找某些 AWS 徽标（例如 Amazon Textract 和 AWS lambda）的位置。训练图像仅包含徽标，并且具有单个图像级标签，例如 lambda 或 textract。也可以使用带有品牌位置边界框的训练图像来训练品牌检测模型。测试图像上标有代表徽标在自然位置（例如架构图）中的位置的边界框。经过训练的模型会查找徽标，并为找到的每个徽标返回一个带标签的边界框。有关更多信息，请参阅查找品牌位置。



物体定位

物体定位项目 (Circuit boards) 训练了一个模型，用于查找印刷电路板上各个零件（例如 comparator 或 ir led）的位置。训练和测试图像包含围绕电路板零件的边界框以及用于标识边界盒内零件的标签。标签名称为 ir_phototransistor、ir_led、pot_resistor 和 comparator。经过训练的模型会查找电路板零件，并为找到的每个电路零件返回一个带标签的边界框。有关更多信息，请参阅查找物体位置。



使用示例项目

这些入门说明介绍了如何使用 Amazon Rekognition Custom Labels 为您创建的示例项目来训练模型。还说明了如何启动模型并用它来分析图像。

创建示例项目

首先，请决定要使用哪个项目。有关更多信息，请参阅步骤 1：选择一个示例项目。

Amazon Rekognition Custom Labels 使用数据集来训练和评估（测试）模型。数据集用于管理图像和标识图像内容的标签。示例项目包含一个训练数据集和一个测试数据集，其中所有图像均带有标签。在训练模型之前，您不需要进行任何更改。示例项目说明了 Amazon Rekognition Custom Labels 使用标签训练不同类型模型的两种方式。

- 图像级：标签标识代表整个图像的物体、场景或概念。
- 边界框：标签标识边界框的内容。边界框是一组围绕图像中物体的图像坐标。

稍后，当您使用自己的图像创建项目时，必须创建训练和测试数据集，并对图像进行标注。有关更多信息，请参阅确定您的模型类型。

训练模型

Amazon Rekognition Custom Labels 创建示例项目后，您即可训练模型。有关更多信息，请参阅步骤 2：训练模型。训练结束后，您通常会评估模型的性能。示例数据集中的图像已经创建了一个高性能模型，您无需在运行模型之前对模型进行评估。有关更多信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。

使用模型

接下来，您需要启动模型。有关更多信息，请参阅步骤 3：启动模型。

开始运行模型后，您可以用它来分析新图像。有关更多信息，请参阅步骤 4：使用模型分析图像。

您将根据模型运行时间付费。使用完示例模型后，应停止模型。有关更多信息，请参阅步骤 5：停止模型。

后续步骤

准备就绪后，您可以创建自己的项目。有关更多信息，请参阅步骤 6：后续步骤。

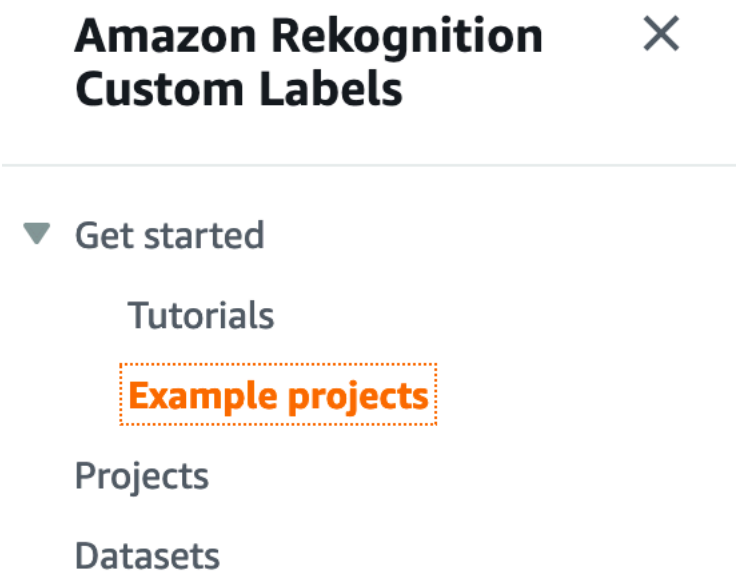
步骤 1：选择一个示例项目

在此步骤中，您将选择一个示例项目。然后，Amazon Rekognition Custom Labels 会为您创建一个项目和一个数据集。项目用于管理用于训练模型的文件。有关更多信息，请参阅管理 Amazon Rekognition Custom Labels 项目。数据集包含用于训练和测试模型的图像、分配的标签和边界框。有关更多信息，请参阅the section called “管理数据集”。

有关示例项目的更多信息，请参阅示例项目。

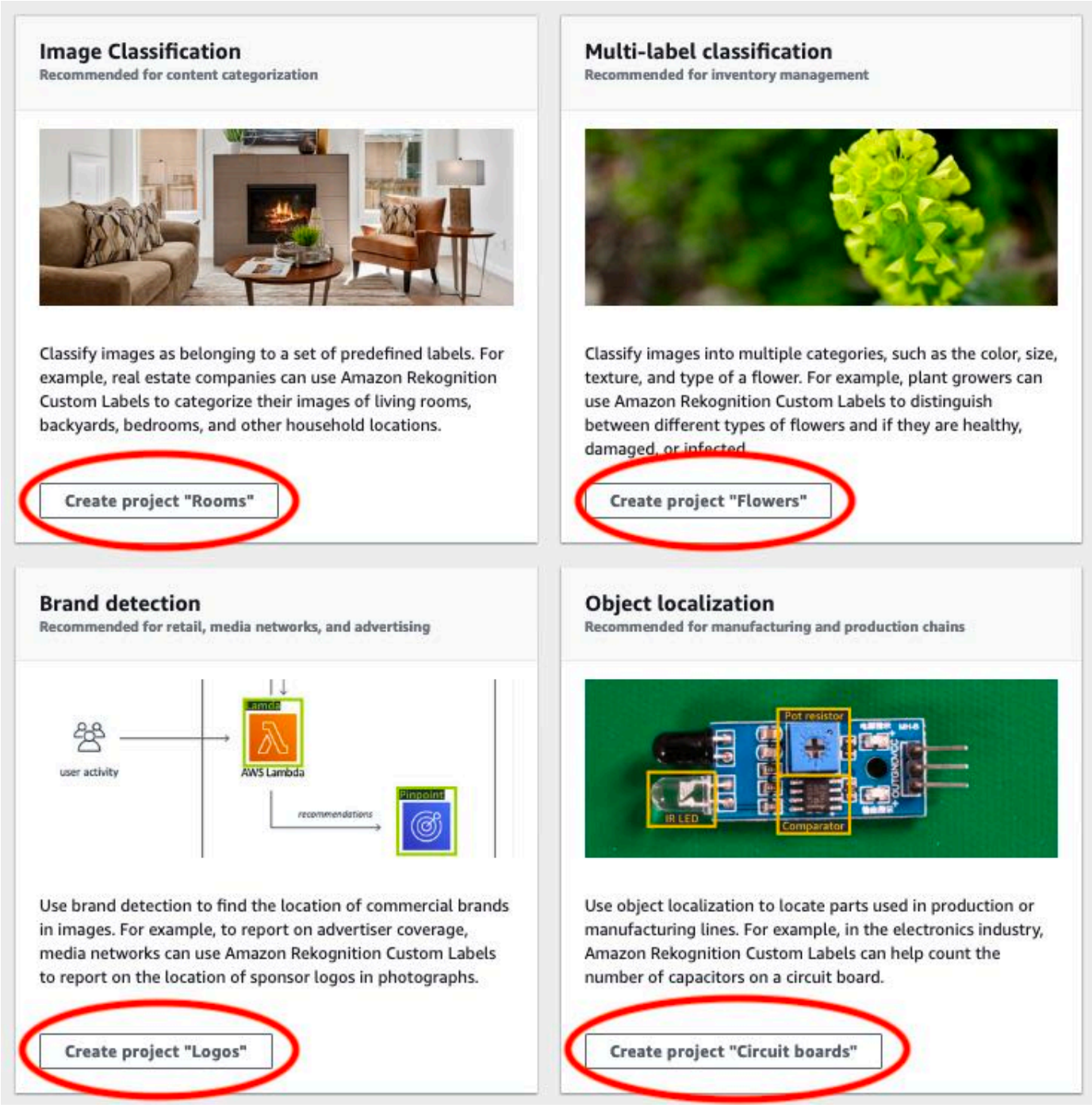
选择一个示例项目

1. 通过以下网址登录 AWS Management Console 并打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。如果没有看到使用自定义标签，请检查您使用的 AWS 区域是否支持 Amazon Rekognition Custom Labels。
3. 选择开始。

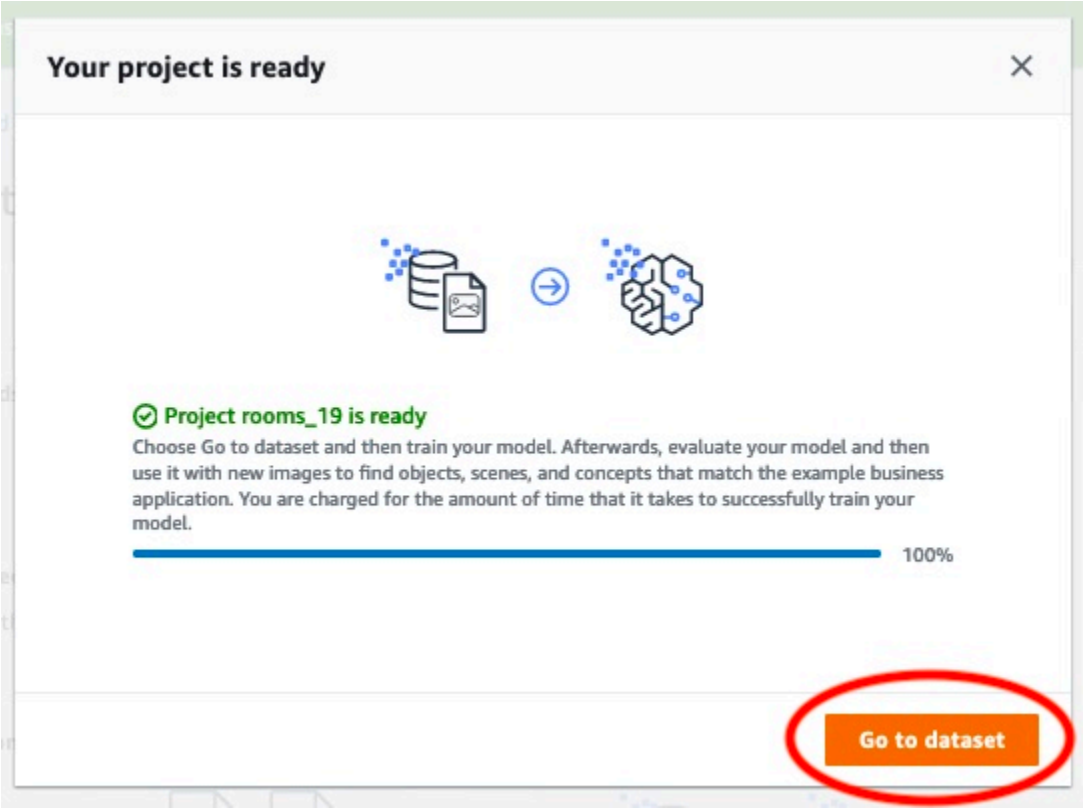


4. 在浏览示例项目中，选择尝试示例项目。
5. 确定要使用哪个项目，然后在示例部分中选择创建**##“####**”。然后，Amazon Rekognition Custom Labels 便会为您创建示例项目。

> **Note**
>
> 如果这是您首次在当前 AWS 区域中打开控制台，这时会显示首次设置对话框。执行以下操作：
>
> 1. 记下所显示的 Amazon S3 存储桶的名称。
> 2. 选择继续，让 Amazon Rekognition Custom Labels 为您创建一个 Amazon S3 存储桶（控制台存储桶）。



6. 项目准备就绪后，选择转到数据集。

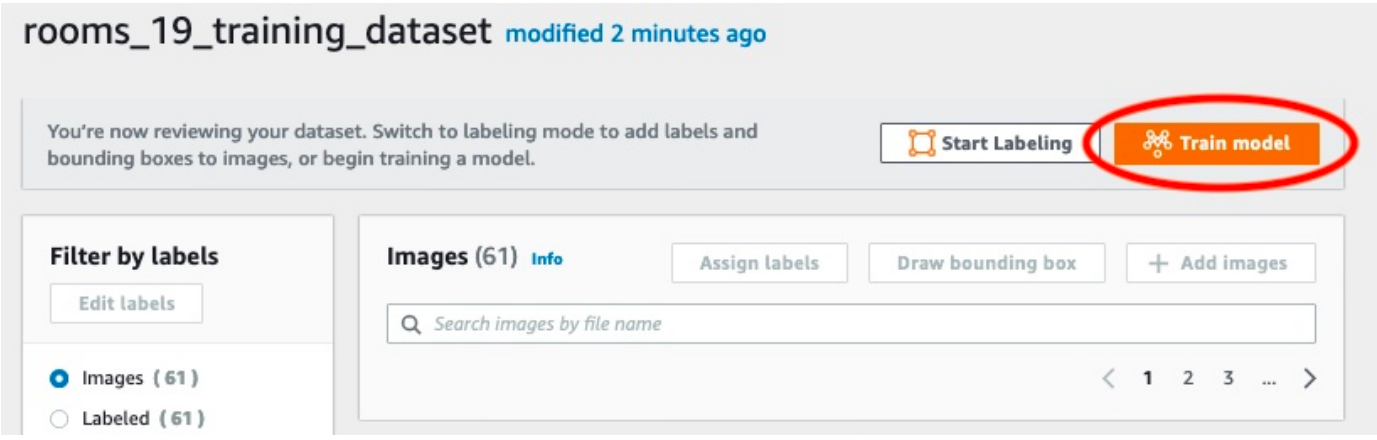


步骤 2：训练模型

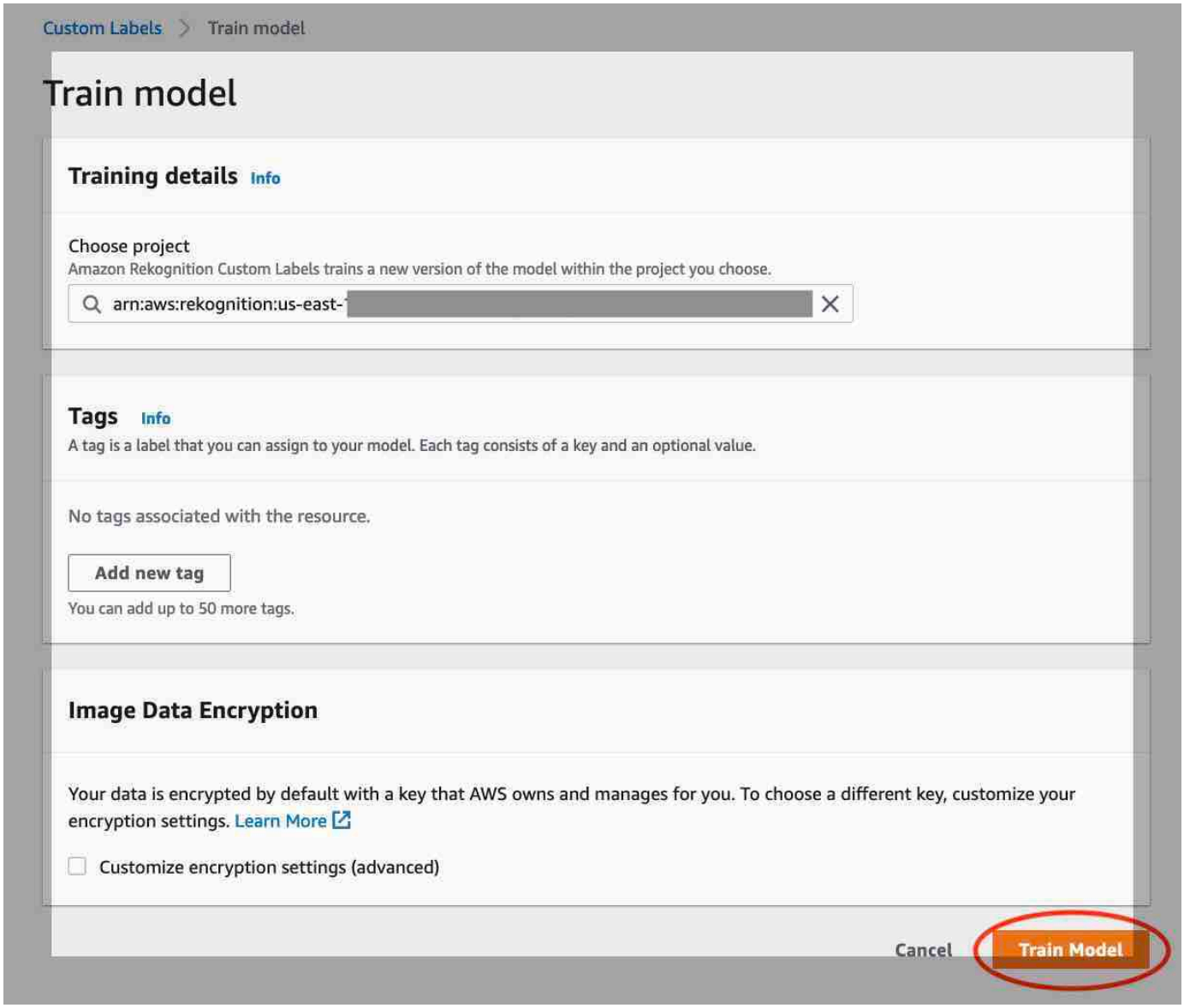
在此步骤中，您将训练您的模型。系统会自动为您配置训练和测试数据集。训练成功完成后，您可以看到整体的评估结果，以及单个测试图像的评估结果。有关更多信息，请参阅[训练 Amazon Rekognition Custom Labels 模型]。

训练您的模型

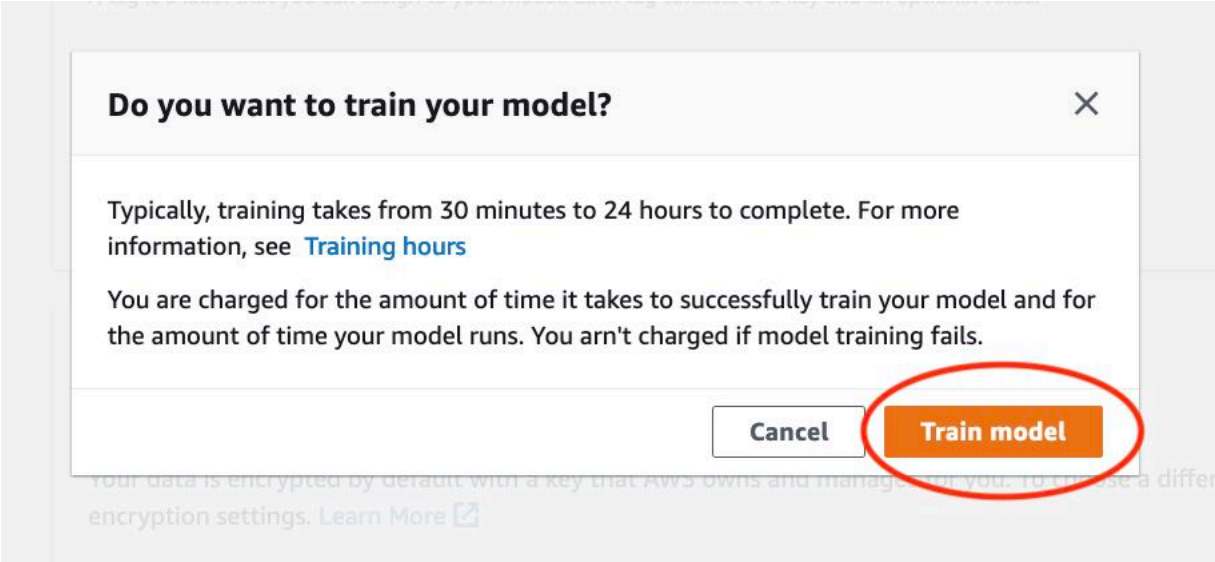
1. 在数据集页面上，选择训练模型。



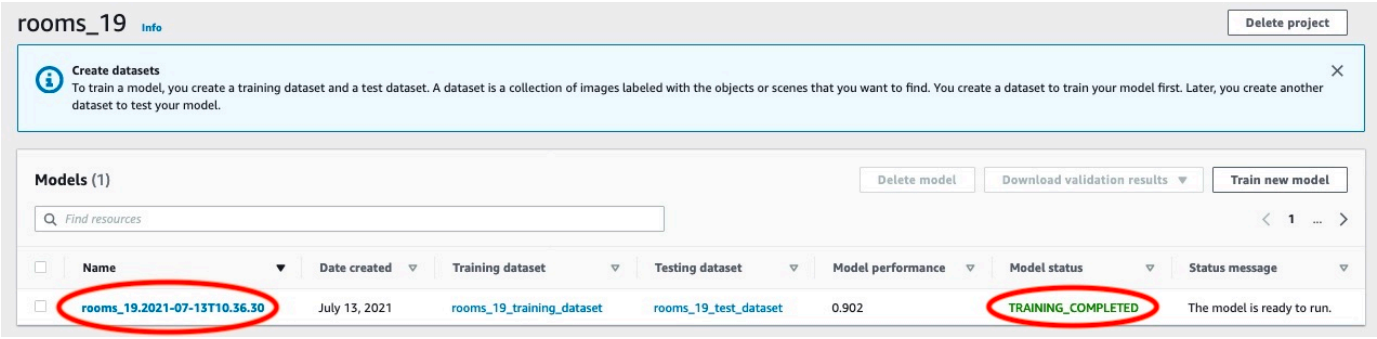
2. 在训练模型页面上，选择训练模型。项目的 Amazon 资源名称 (ARN) 位于选择项目编辑框中。



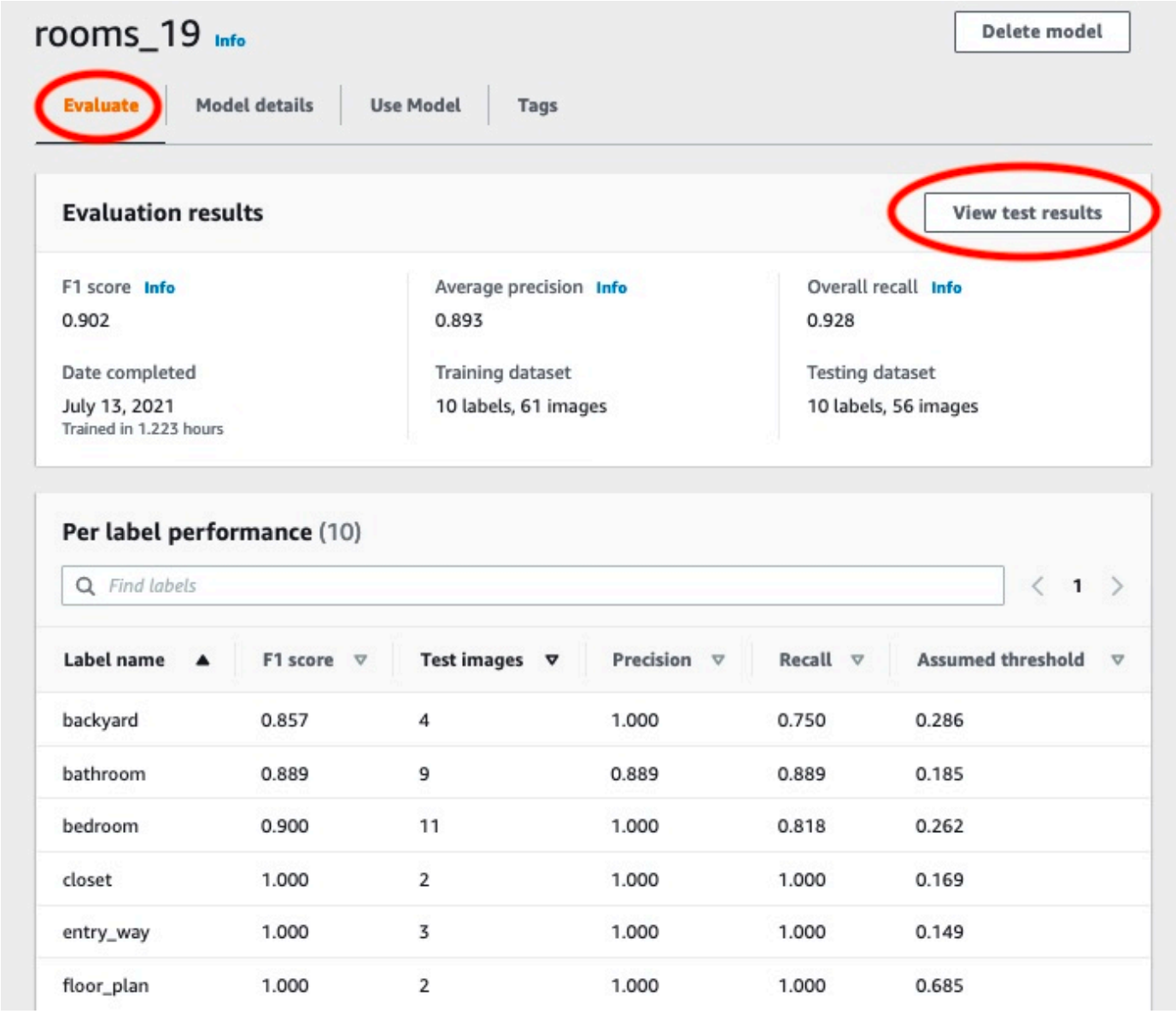
3. 在是否要训练您的模型？对话框中，选择训练模型。



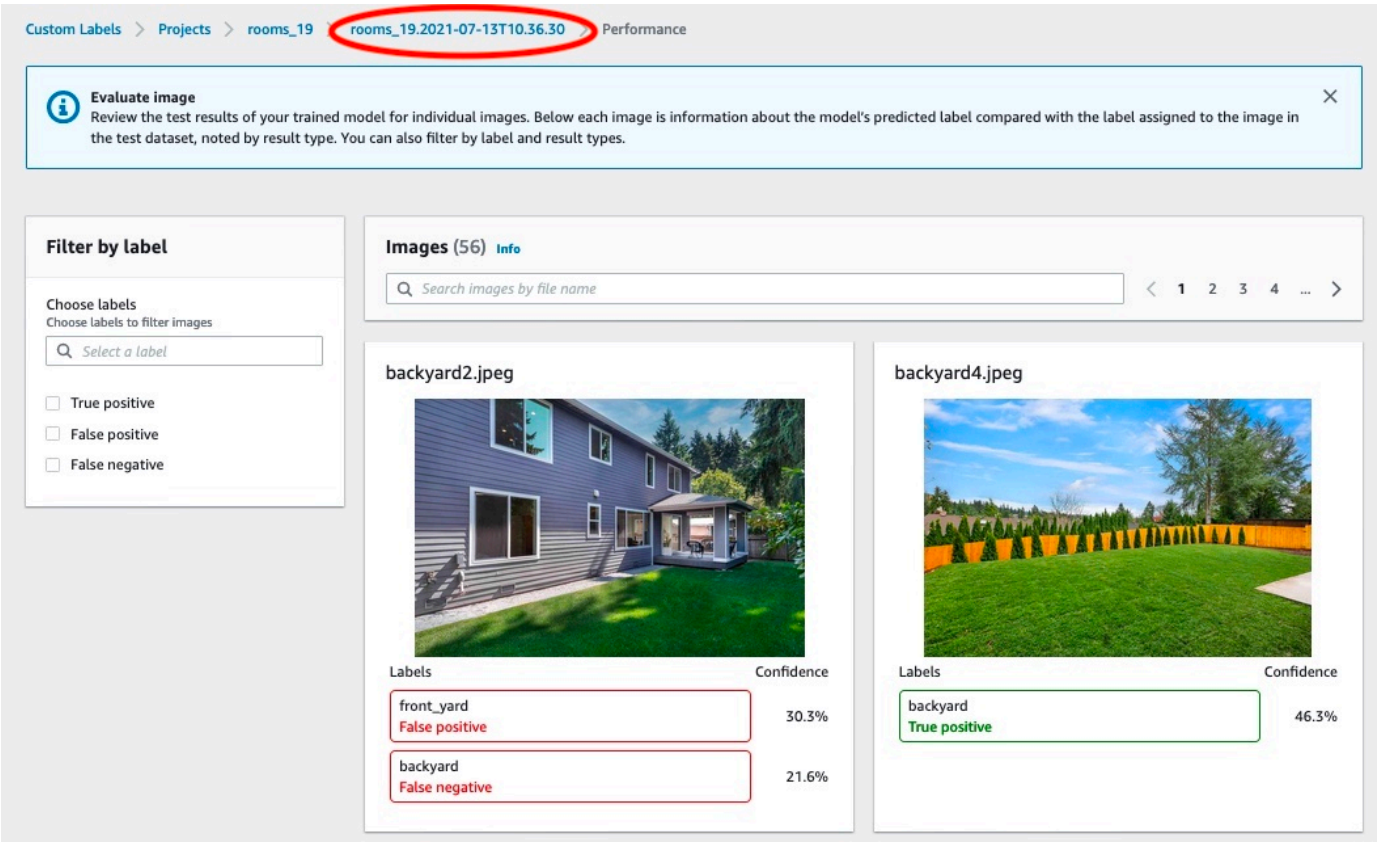
4. 训练完成后，选择模型名称。当模型状态为 TRAINING_COMPLETED 时，训练即告完成。



5. 选择评估按钮，以查看评估结果。有关评估模型的信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。
6. 选择查看测试结果，以查看单个测试图像的结果。



7. 查看测试结果后，选择模型名称返回模型页面。



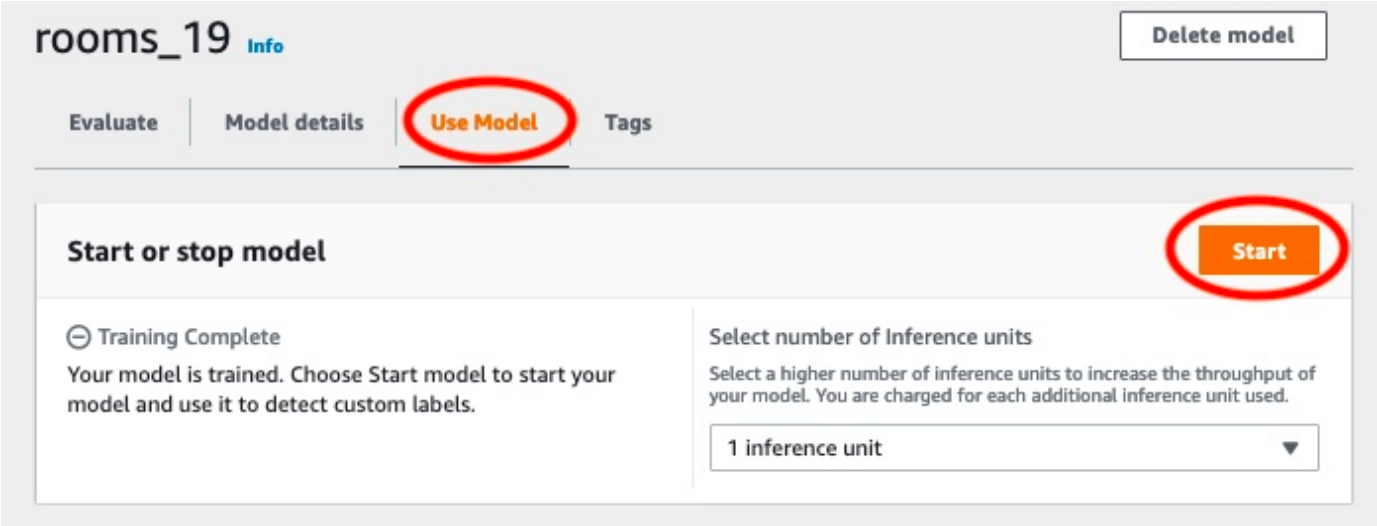
步骤 3：启动模型

在此步骤中，您将启动您的模型。模型启动后，您可以用它来分析图像。

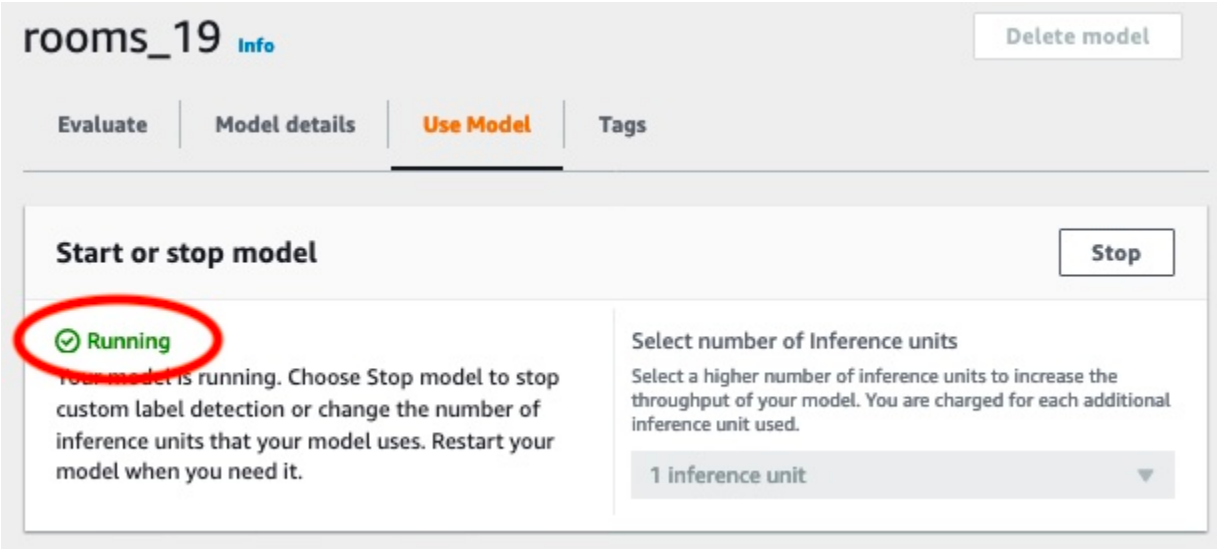
您将根据模型运行时间付费。如果您不需要分析图像，请停止模型。您可以稍后重新启动模型。有关更多信息，请参阅运行经过训练的 Amazon Rekognition Custom Labels 模型。

启动模型

1. 在模型页面上选择使用模型选项卡。
2. 在启动或停止模型部分中，执行以下操作：
   a. 选择启动。
   b. 在启动模型对话框中，选择启动。



3. 等到模型开始运行。当启动或停止模型部分中的状态为正在运行时，即表示模型正在运行。



4. 使用模型对图像进行分类。有关更多信息，请参阅步骤 4：使用模型分析图像。

步骤 4：使用模型分析图像

您可以通过调用 DetectCustomLabels API 来分析图像。在此步骤中，您将使用 `detect-custom-labels` AWS Command Line Interface (AWS CLI) 命令分析示例图像。您可以从 Amazon Rekognition Custom Labels 控制台获取 AWS CLI 命令。控制台将 AWS CLI 命令配置为使用您的模型。您只需要提供存储在 Amazon S3 存储桶中的图像即可。本主题提供了可用于每个示例项目的图像。

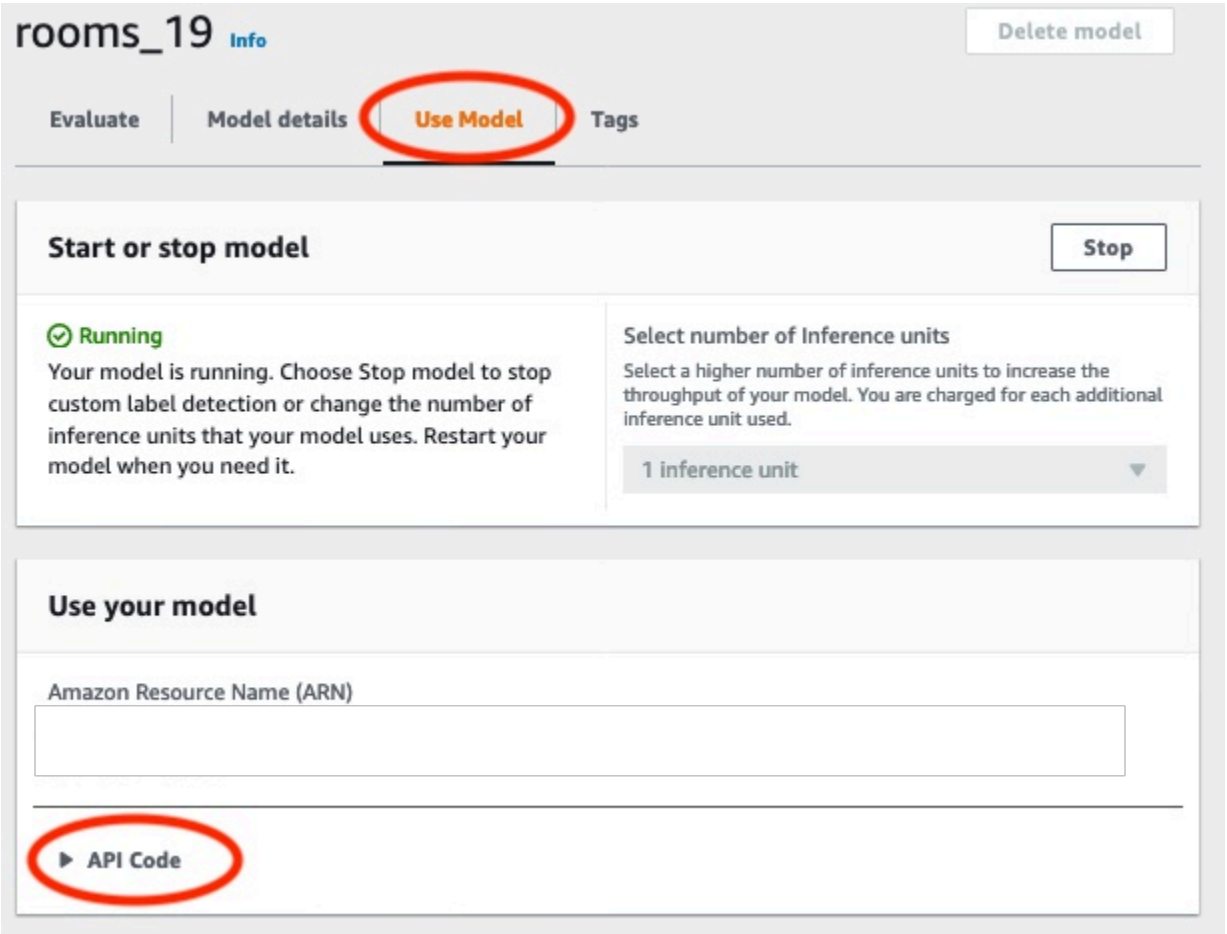
> **Note**
>
> 控制台还提供了调用 Python 示例代码。

`detect-custom-labels` 的输出包括在图像中找到的标签列表、边界框（如果模型会查找物体位置）以及模型对预测准确性的置信度。

有关更多信息，请参阅使用经过训练的模型分析图像。

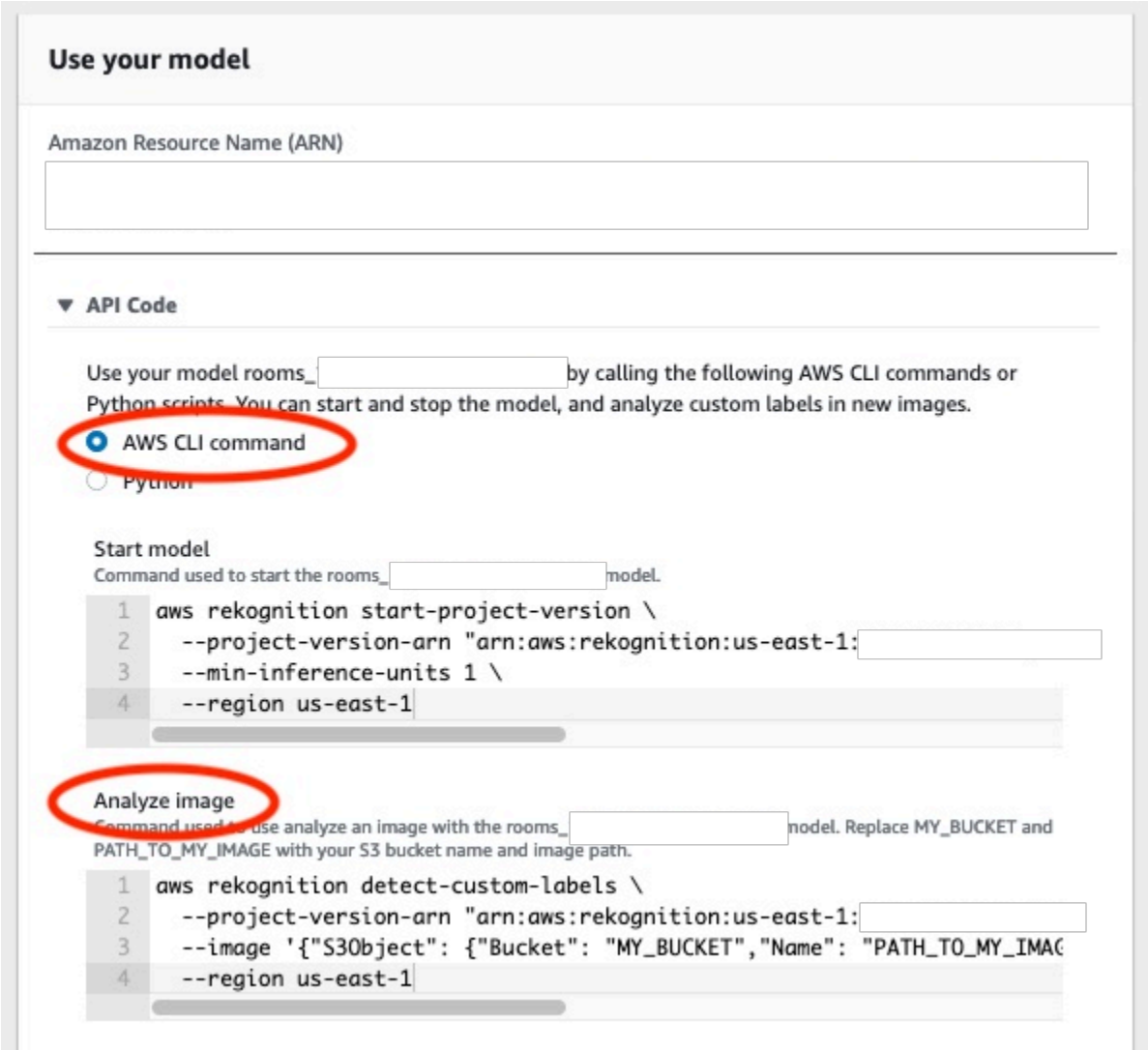
分析图像（控制台）

1. 设置 AWS CLI（如果尚未如此）。有关说明，请参阅the section called “步骤 4：设置 AWS CLI 和 AWS SDK”。
2. 开始运行您的模型（如果尚未如此）。有关更多信息，请参阅步骤 3：启动模型。
3. 选择使用模型选项卡，然后选择 API 代码。



4. 选择 AWS CLI 命令。

5. 在分析图像部分中，复制调用 `detect-custom-labels` 的 AWS CLI 命令。



6. 将示例图像上传到 Amazon S3 存储桶。有关说明，请参阅获取示例图像。

7. 在命令提示符处，输入您在上一步中复制的 AWS CLI 命令。它应该类似于以下示例。

   `--project-version-arn` 的值应为模型的 Amazon 资源名称 (ARN)。`--region` 的值应为您在其中创建模型的 AWS 区域。

   将 `MY_BUCKET` 和 `PATH_TO_MY_IMAGE` 更改为您在上一步中使用的 Amazon S3 存储桶和图像。

   如果您使用 custom-labels-access 配置文件来获取凭证，请添加 `--profile custom-labels-access` 参数。

```
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
```

如果模型查找物体、场景和概念，则 AWS CLI 命令的 JSON 输出应类似于以下内容。Name 是模型找到的图像级标签的名称。Confidence (0-100) 是模型对预测准确性的置信度。

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

如果模型查找物体位置或品牌，则会返回带标签的边界框。BoundingBox 包含围绕物体的方框的位置。Name 是模型在边界框中找到的物体。Confidence 是模型对边界框所包含物体的置信度。

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

8. 继续使用该模型分析其他图像。如果不再使用该模型，请停止模型。有关更多信息，请参阅步骤 5：停止模型。

获取示例图像

您可以将下列图像与 DetectCustomLabels 操作结合使用。每个项目都有一个图像。要使用这些图像，您需要将它们上传到 S3 存储桶。

使用示例图像

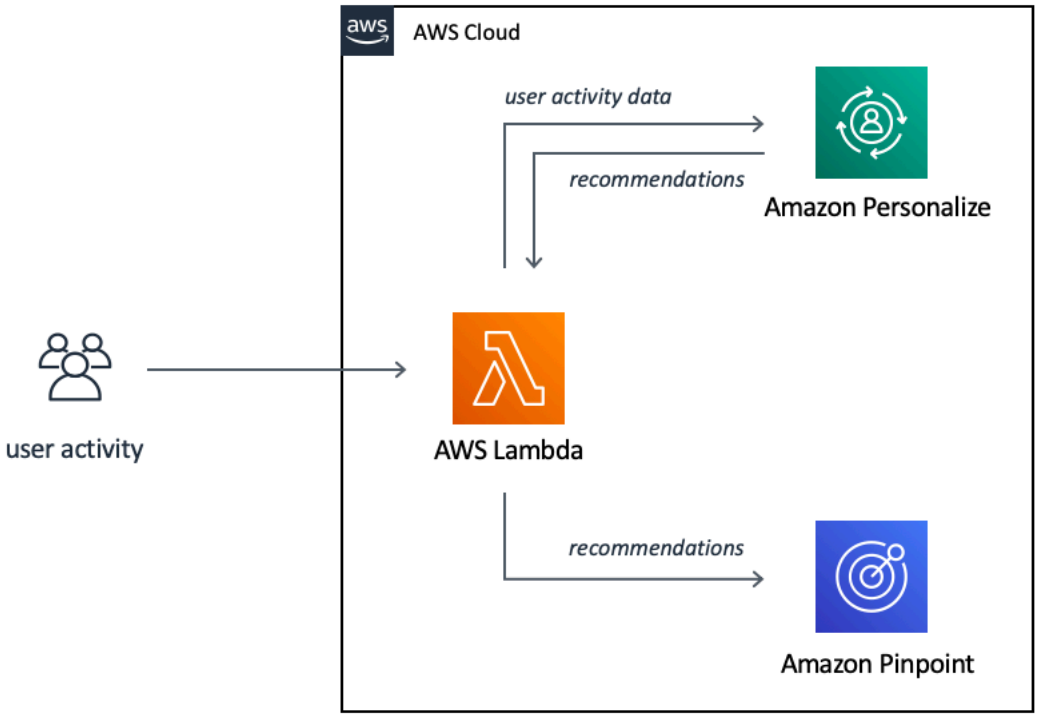
1. 右键单击下面与您正在使用的示例项目匹配的图像。然后选择保存图像，将图像保存到您的计算机。菜单选项可能会有所不同，具体取决于您使用的浏览器。
2. 将图像上传到您的 AWS 账户拥有的 Amazon S3 存储桶，该存储桶位于您在其中使用 Amazon Rekognition Custom Labels 的同一 AWS 区域。

   有关说明，请参阅《Amazon Simple Storage Service 用户指南》中的将对象上传到 Amazon S3。

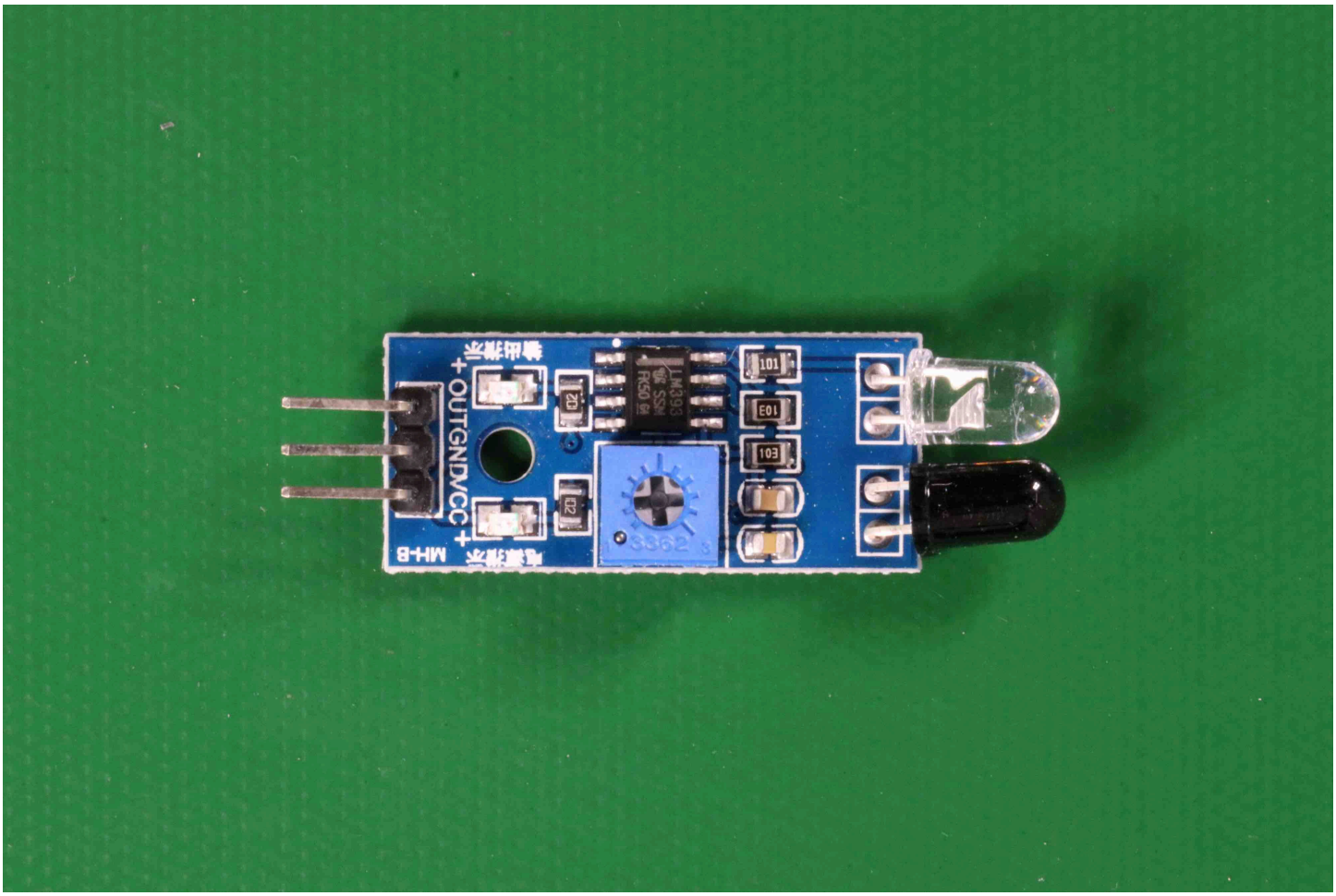
图像分类

多标签分类

品牌检测


AWS Cloud
user activity data
recommendations
Amazon Personalize
user activity
AWS Lambda
recommendations
Amazon Pinpoint

物体定位

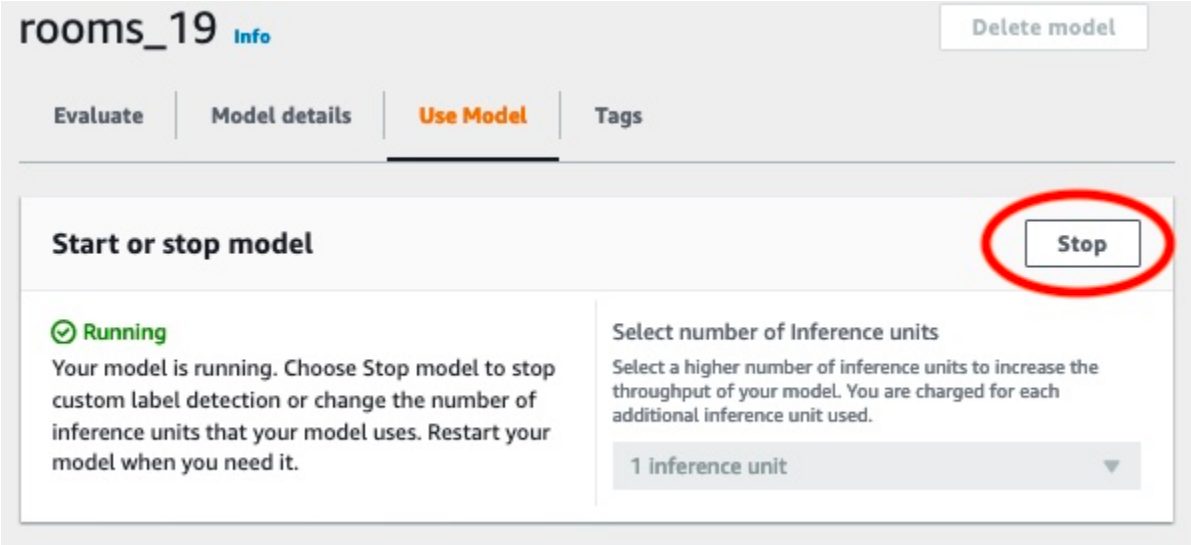


步骤 5：停止模型

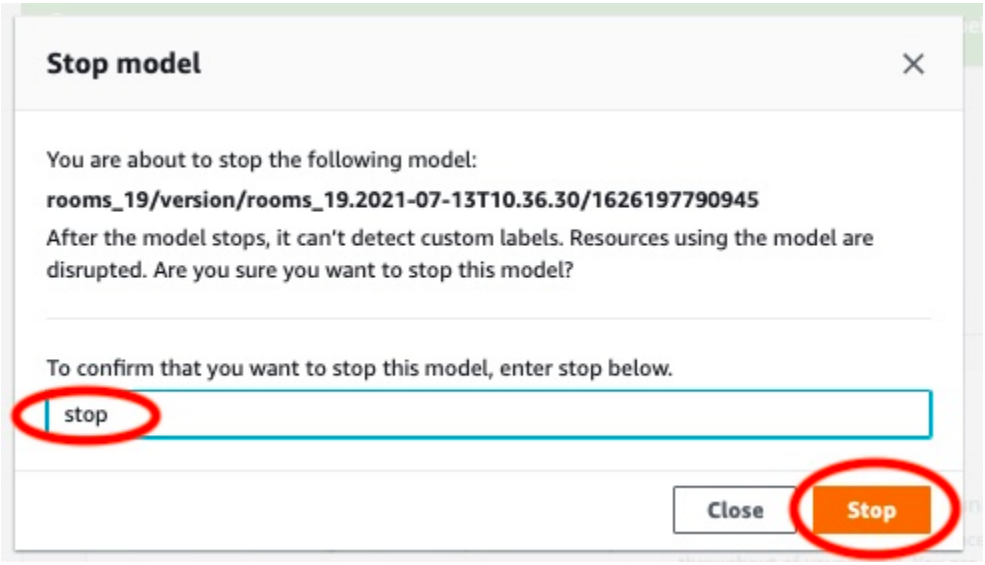
在此步骤中，您将停止运行您的模型。您需要按照模型运行的时间付费。如果您已使用完模型，应将其停止。

停止模型

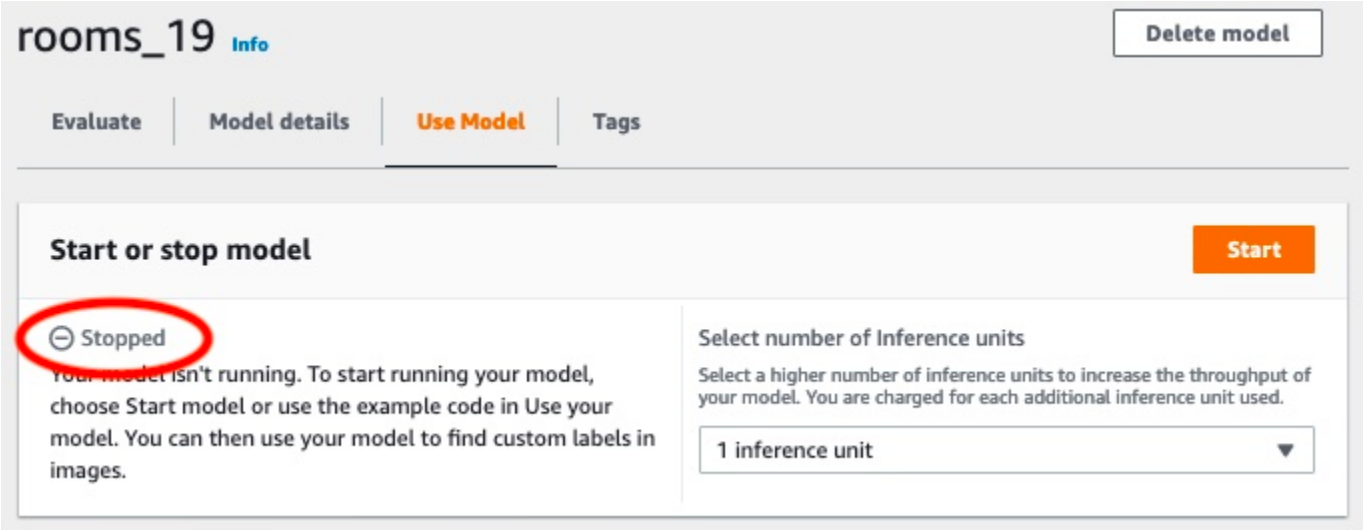
1. 在启动或停止模型部分中，选择停止。



2. 在停止模型对话框中，输入 stop 以确认要停止模型。



3. 选择停止以停止模型。当启动或停止模型部分中的状态为已停止时，即表示模型已停止。



步骤 6：后续步骤

完成示例项目的试用后，您可以使用自己的图像和数据集来创建自己的模型。有关更多信息，请参阅了解 Amazon Rekognition Custom Labels。

使用下表中的标注信息来训练与示例项目相似的模型。

示例	训练图像	测试图像
图像分类 (Rooms)	每张图像 1 个图像级标签	每张图像 1 个图像级标签
多标签分类 (Flowers)	每张图像多个图像级标签	每张图像多个图像级标签
品牌检测 (Logos)	图像级标签（也可以使用带标签的边界框）	带标签的边界框
图像定位 (Circuit boards)	带标签的边界框	带标签的边界框

教程：对图像进行分类说明了如何为图像分类模型创建项目、数据集和模型。

有关创建数据集和训练模型的详细信息，请参阅创建 Amazon Rekognition Custom Labels 模型。

教程：对图像进行分类

本教程介绍如何为可对图像中找到的物体、场景和概念进行分类的模型创建项目和数据集。模型对整个图像进行分类。例如，通过遵循本教程，您可以训练模型来识别客厅或厨房等家庭位置。本教程还介绍了如何使用模型来分析图像。

在开始本教程之前，建议您先阅读了解 Amazon Rekognition Custom Labels。

在本教程中，您将通过从本地计算机上传图像来创建训练和测试数据集。稍后，您将为训练和测试数据集中的图像分配图像级标签。

您创建的模型会将图像归类为属于您分配给训练数据集图像的图像级标签集。例如，如果训练数据集中的图像级标签集为 `kitchen`、`living_room`、`patio` 和 `backyard`，则模型有可能在单张图像中找到所有这些图像级标签。

> **Note**
>
> 您可以为不同的目的创建模型，例如在图像上查找物体的位置。有关更多信息，请参阅确定您的模型类型。

步骤 1：收集图像

您需要两组图像。一组添加到您的训练数据集中。另一组添加到您的测试数据集中。这些图像应代表您希望模型分类的物体、场景和概念。这些图像必须是 PNG 或 JPEG 格式。有关更多信息，请参阅准备图像。

训练数据集应至少有 10 张图像，测试数据集应至少有 10 张图像。

如果您还没有图像，请使用 Rooms 示例分类项目中的图像。创建项目后，训练和测试图像将位于以下 Amazon S3 存储桶位置：

- 训练图像：`s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`
- 测试图像：`s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`

region 是您使用 Amazon Rekognition Custom Labels 控制台的 AWS 区域。numbers 是控制台为存储桶名称分配的值。Version number 是示例项目的版本号，从 1 开始。

以下过程会将 Rooms 项目中的图像存储到计算机上名为 training 和 test 的本地文件夹中。

下载 Rooms 示例项目图像文件

1. 创建 Rooms 项目。有关更多信息，请参阅步骤 1：选择一个示例项目。
2. 打开命令提示符，输入以下命令以下载训练图像。

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

3. 在命令提示符下，输入以下命令以下载测试图像。

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

4. 将 training 文件夹中的两张图像移动到您选择的单独文件夹中。您将在步骤 9：使用模型分析图像中使用这些图像来试用训练后的模型。

步骤 2：决定类别

列出您希望模型查找的类别。例如，如果要训练模型来识别房子中的房间，则可以将以下图像分类为 living_room。

每个类别都映射到一个图像级标签。稍后，您将为训练和测试数据集中的图像分配图像级标签。

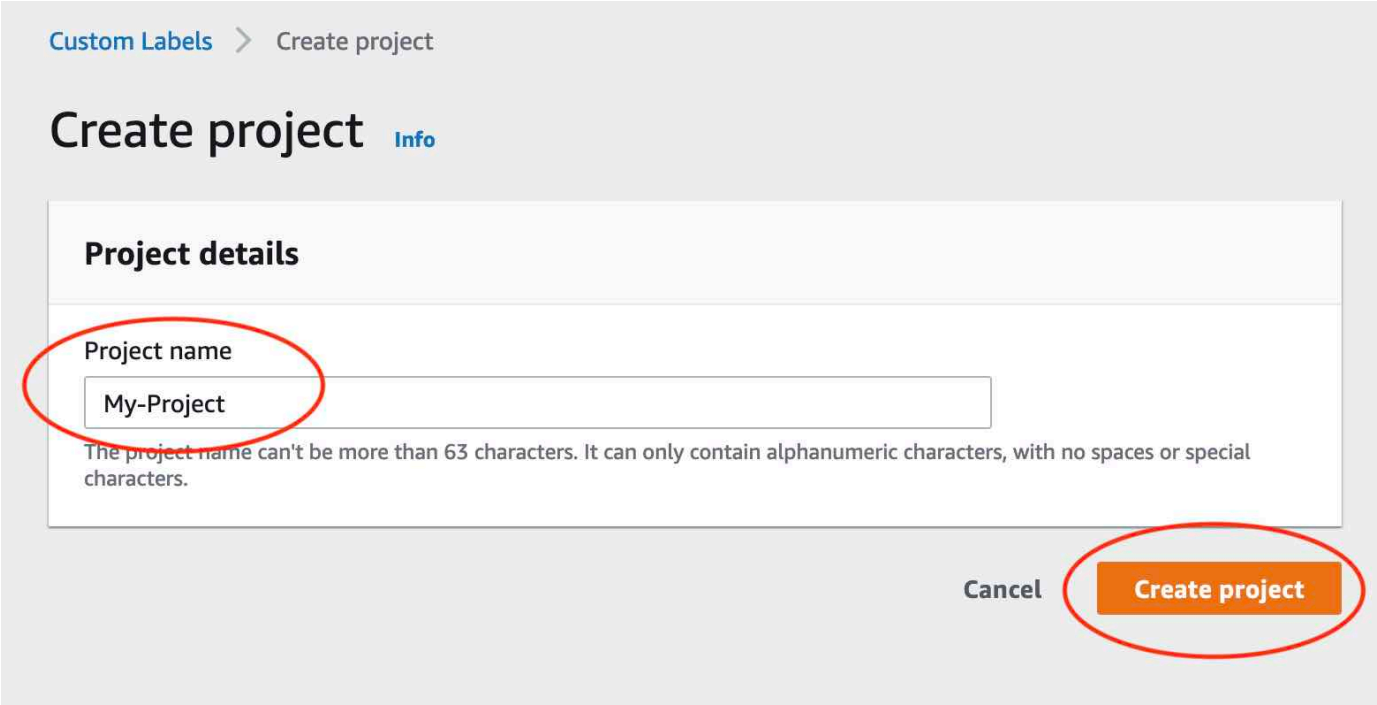
如果您使用的是 Rooms 示例项目中的图像，则图像级标签为 backyard、bathroom、bedroom、closet、entry_way、floor_plan、front_yard、kitchen、living_space 和 patio。

步骤 3：创建项目

要管理您的数据集和模型，您需要创建一个项目。每个项目都应对应一个使用场景，例如识别房子中的房间。

创建项目（控制台）

1. 设置 Amazon Rekognition Custom Labels 控制台（如果尚未如此）。有关更多信息，请参阅设置 Amazon Rekognition Custom Labels。
2. 通过以下网址登录 AWS Management Console 并打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
3. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。
4. 在 Amazon Rekognition Custom Labels 登录页面上，选择开始。
5. 在左侧导航窗格中，选择项目。
6. 在项目页面上，选择创建项目。
7. 在项目名称中输入项目名称。
8. 选择创建项目，创建您的项目。



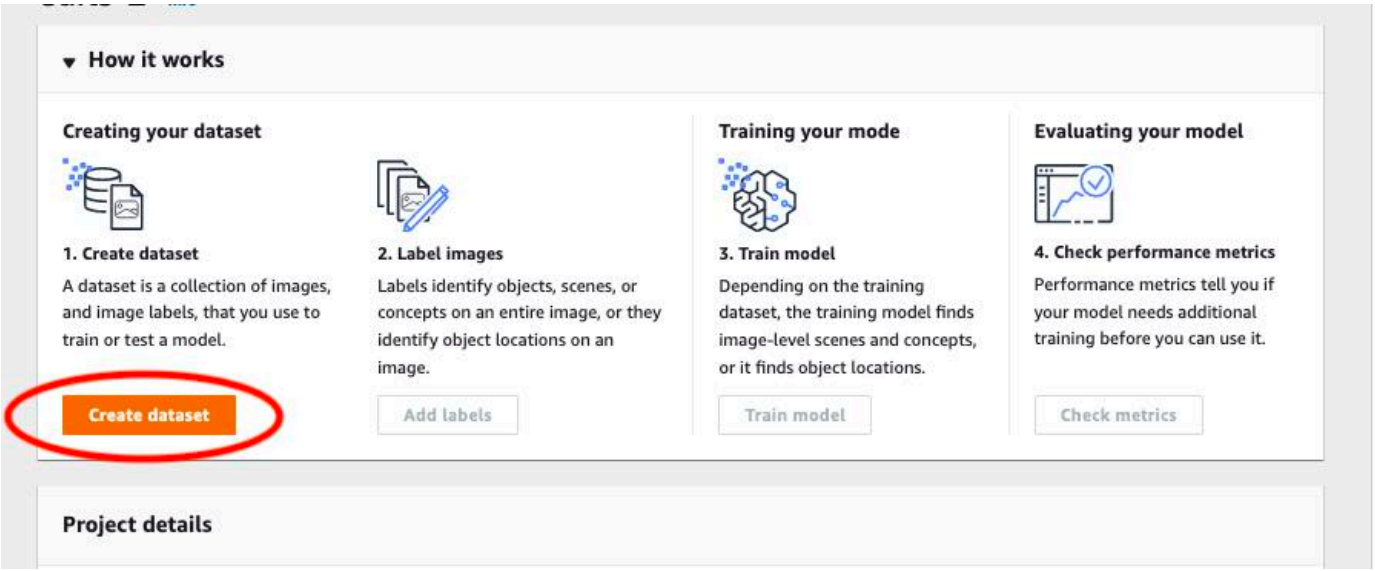
步骤 4：创建训练和测试数据集

在本步骤中，您将通过从本地计算机上传图像来创建训练和测试数据集。一次最多可以上传 30 张图像。如果您要上传的图像很多，可以考虑通过从 Amazon S3 存储桶导入图像来创建数据集。有关更多信息，请参阅Amazon S3 存储桶。

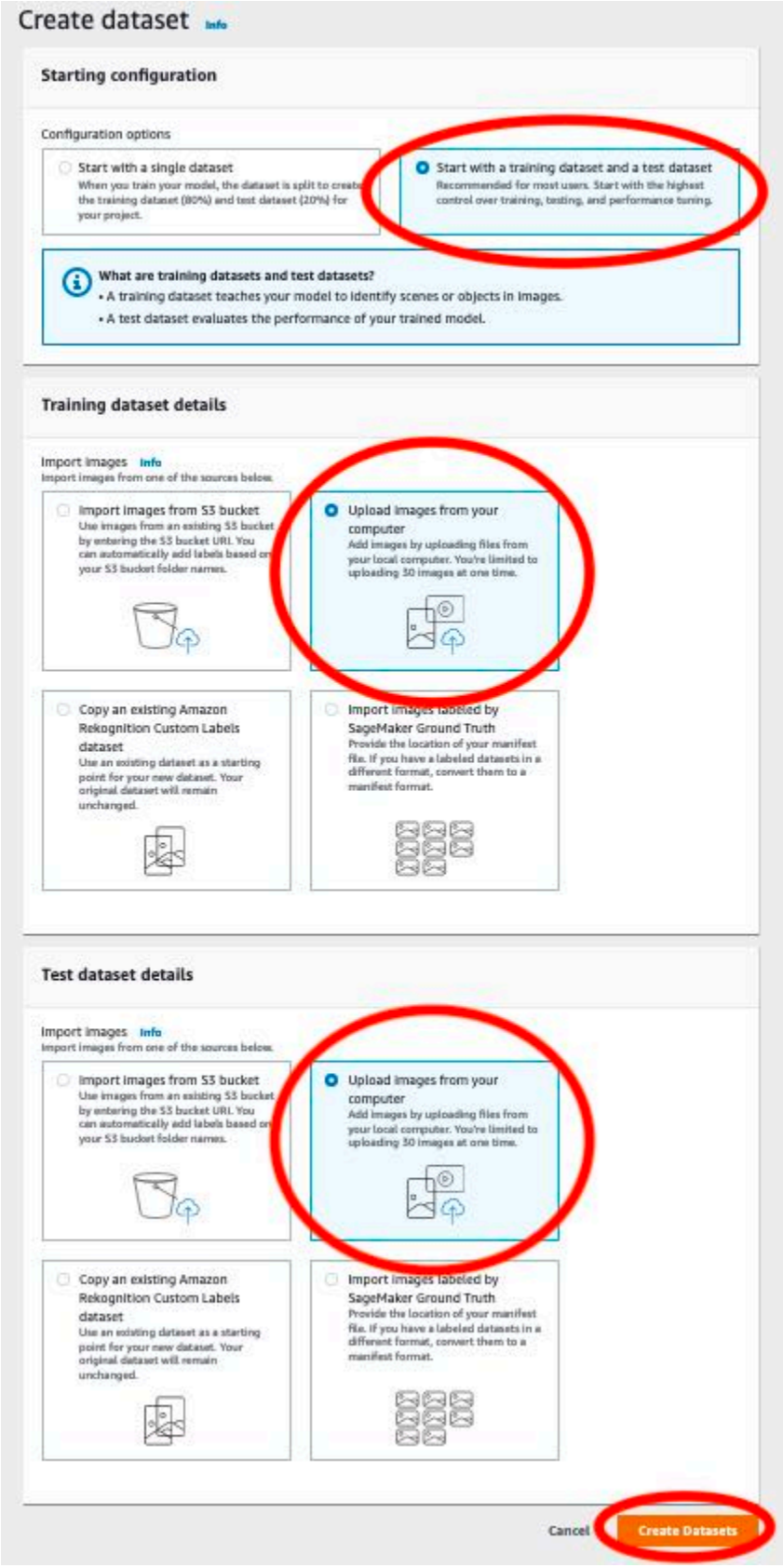
有关数据集的更多信息，请参阅管理数据集。

使用本地计算机上的图像创建数据集（控制台）

1. 在项目详细信息页面上，选择创建数据集。

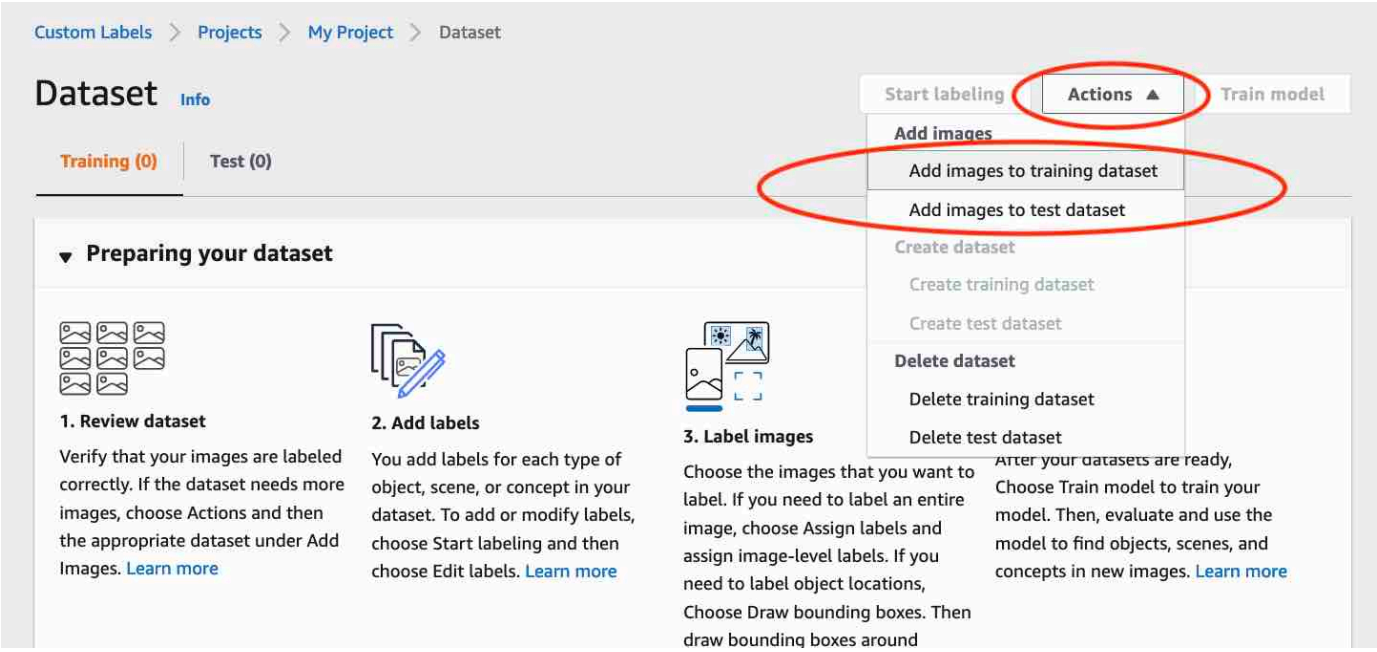


2. 在开始配置部分，选择从一个训练数据集和一个测试数据集开始。
3. 在训练数据集详细信息部分中，选择从您的计算机上传图像。
4. 在测试数据集详细信息部分中，选择从您的计算机上传图像。
5. 选择创建数据集。

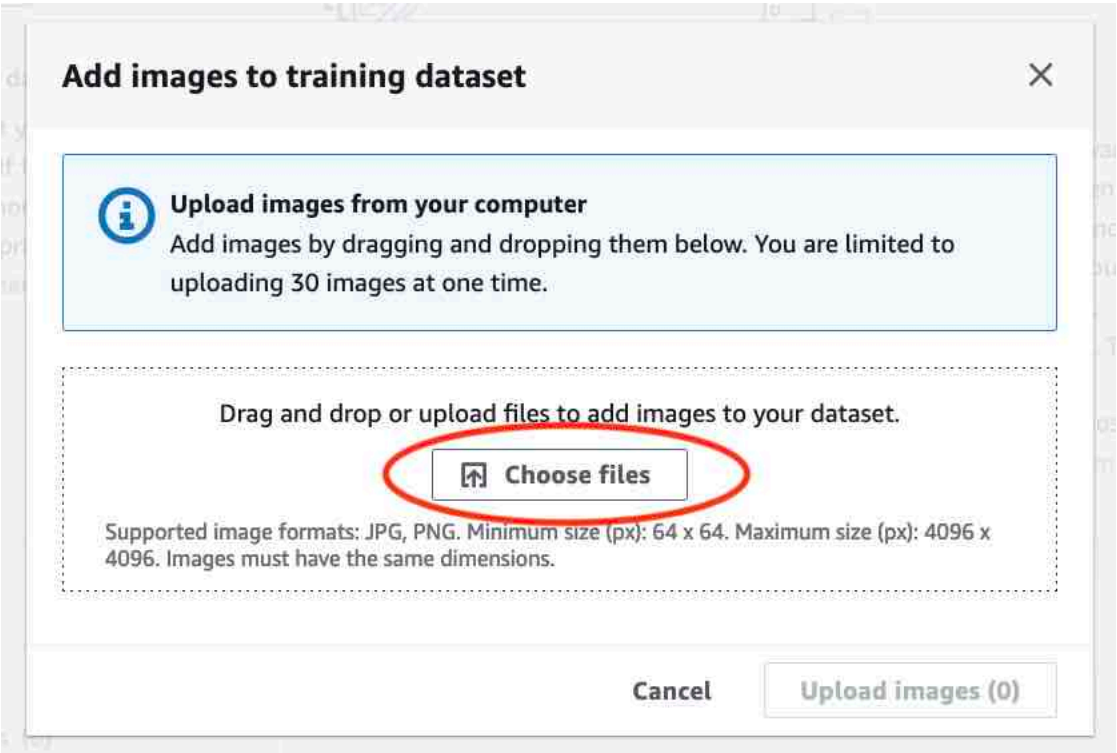


6. 此时会显示一个数据集页面，其中包含训练选项卡和测试选项卡，分别对应于相应的数据集。

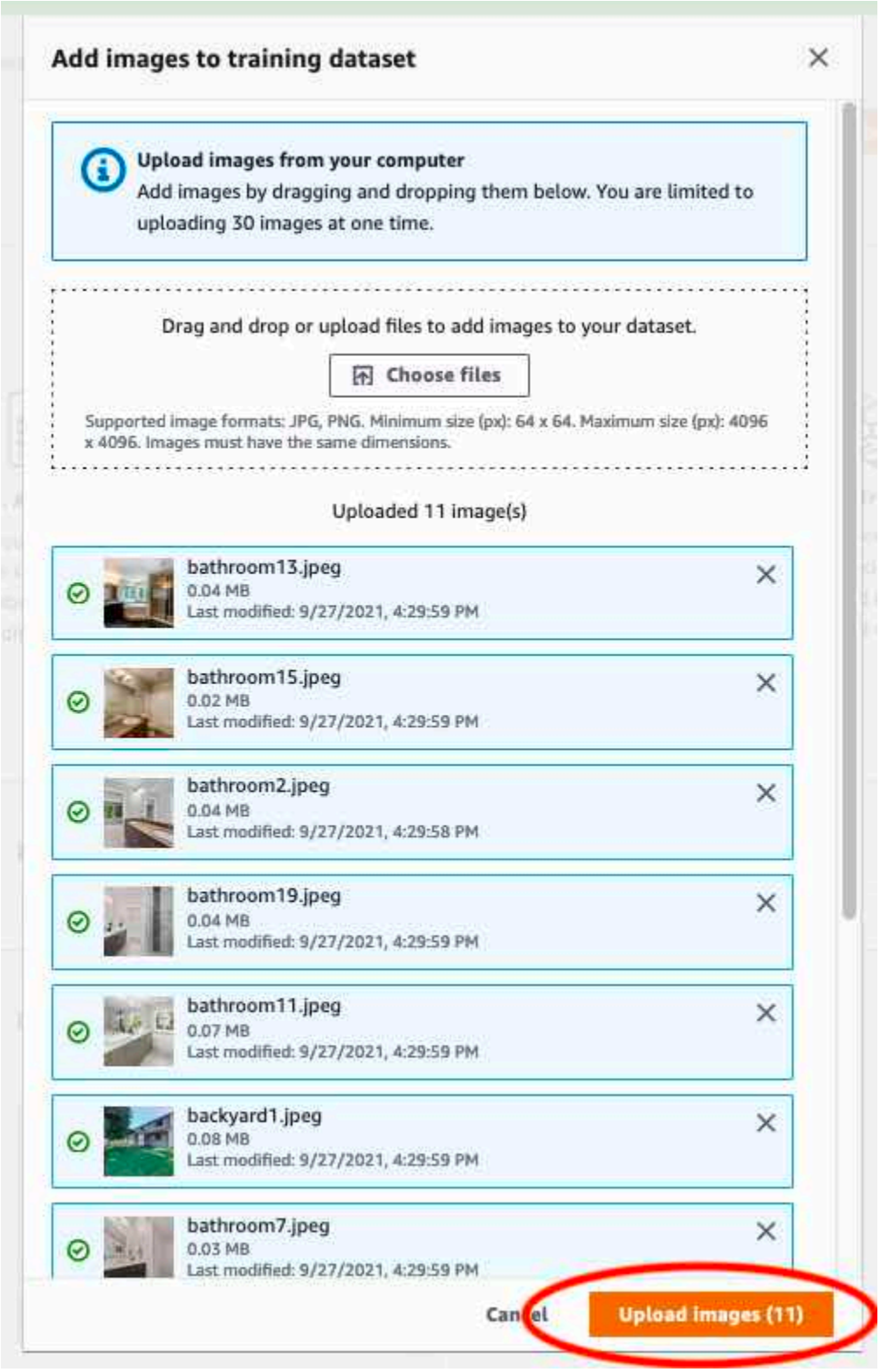
7. 在数据集页面上，选择训练选项卡。
8. 选择操作，然后选择将图像添加到训练数据集。



9. 在将图像添加到训练数据集对话框中，选择选择文件。



10. 选择要上传到数据集的图像。一次最多可以上传 30 张图像。
11. 选择上传图像。Amazon Rekognition Custom Labels 可能需要几秒钟才能将图像添加到数据集。



12. 如果您有更多图像要添加到训练数据集中，请重复步骤 9-12。
13. 选择测试选项卡。

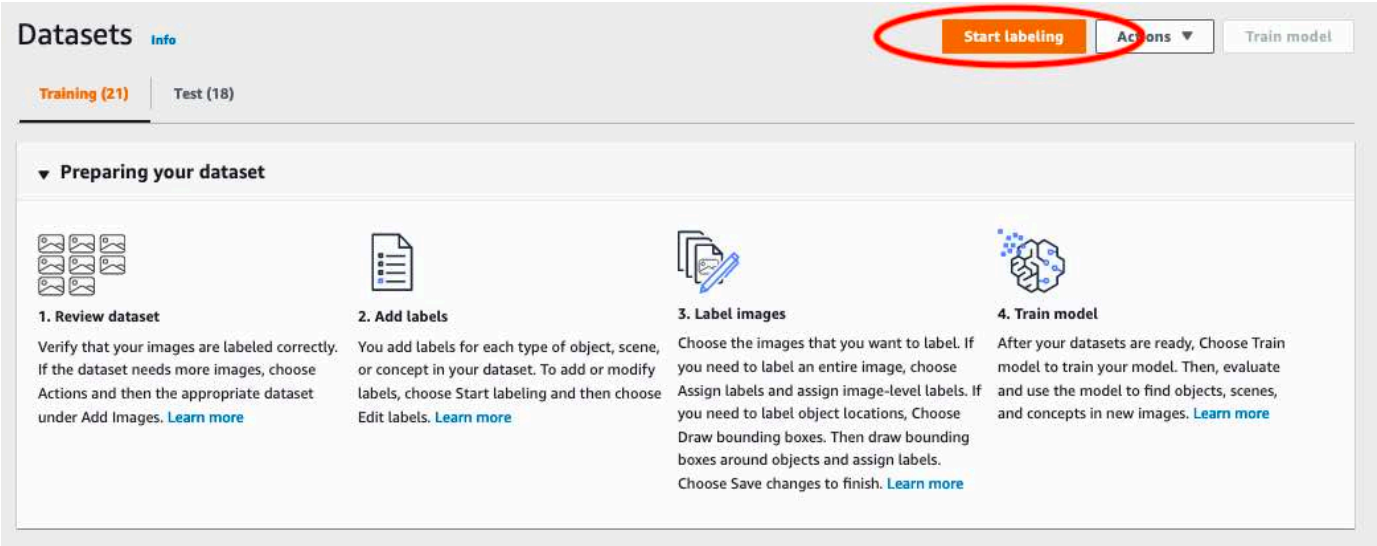
14. 重复步骤 8-12，将图像添加到测试数据集。在步骤 8 中，选择操作，然后选择将图像添加到测试数据集。

步骤 5：向项目中添加标签

在此步骤中，您将为在步骤步骤 2：决定类别中确定的每一个类别向项目中添加一个标签。

添加新标签（控制台）

1. 在数据集库页面中，选择开始标注进入标注模式。



2. 在数据集库的标签部分，选择管理标签，打开管理标签对话框。
3. 在编辑框中，输入新标签名称。
4. 选择添加标签。
5. 重复步骤 3 和 4，直到创建完所需的所有标签。
6. 选择保存，保存您添加的标签。

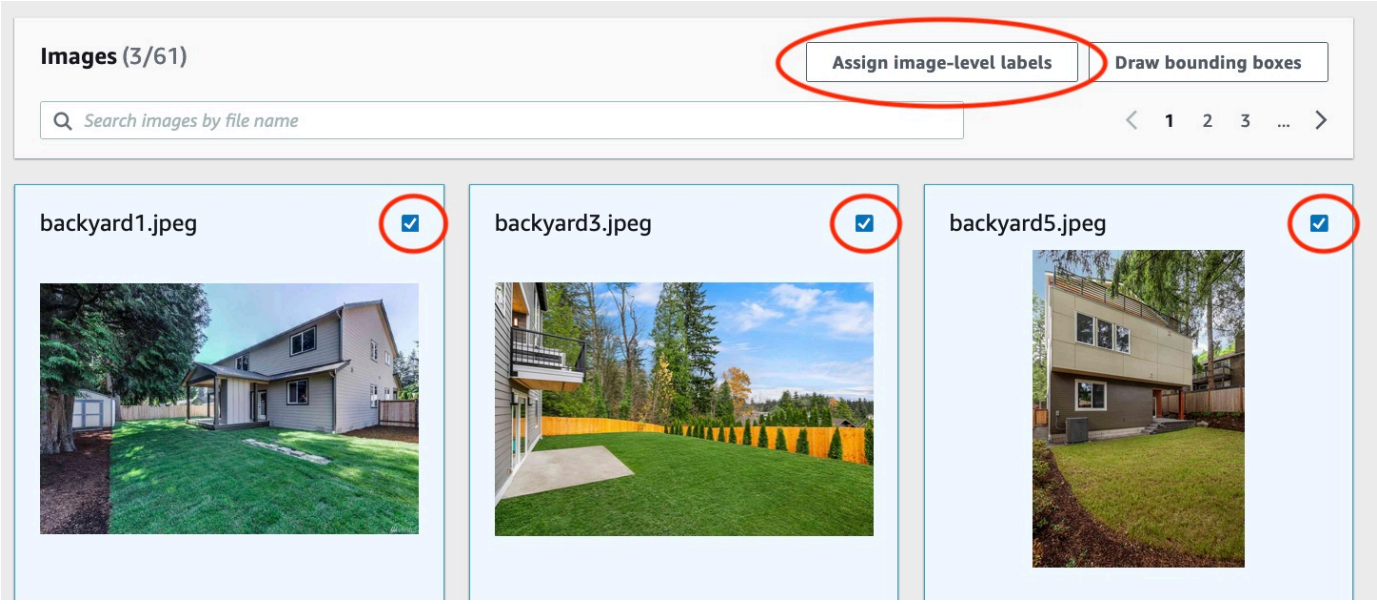
步骤 6：为训练和测试数据集分配图像级标签

在此步骤中，您将为训练和测试数据集中的每张图像分配一个图像级标签。图像级标签是每张图像所代表的类别。

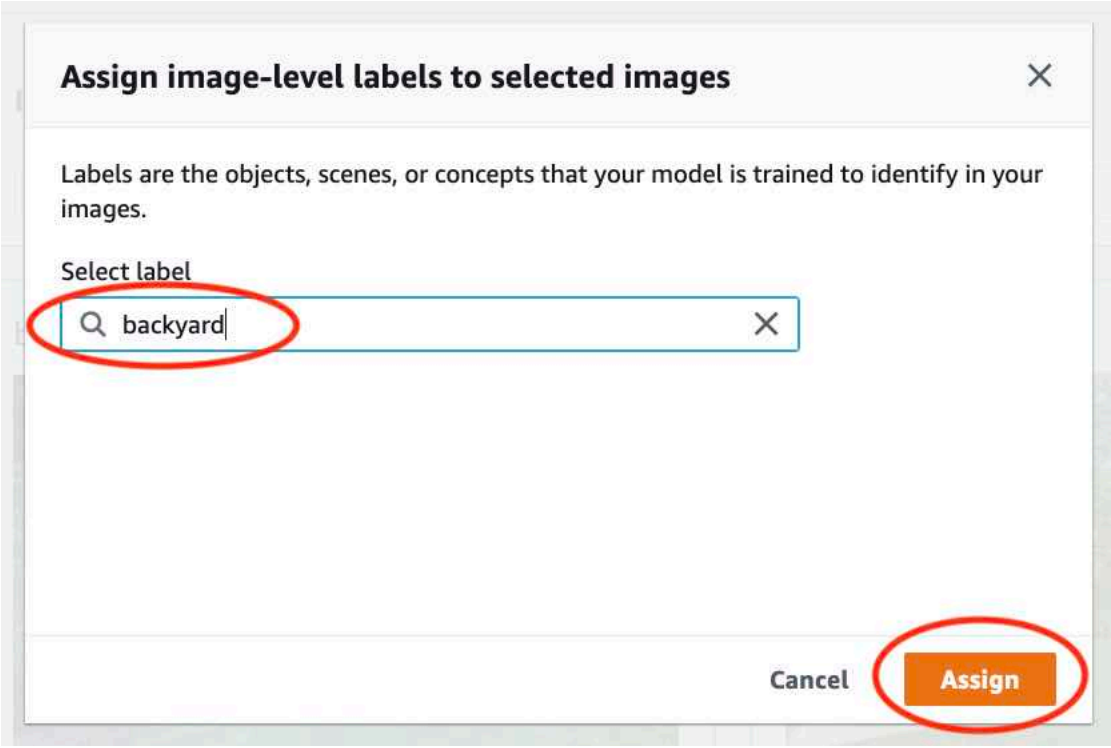
为图像分配图像级标签（控制台）

1. 在数据集页面上，选择训练选项卡。

2. 选择开始标注，进入标注模式。
3. 选择要添加标签的一张或多张图像。一次只能选择一个页面上的图像。要在一个页面上选择连续范围的图像，请执行以下操作：
   a. 选择第一张图像。
   b. 按住 Shift 键。
   c. 选择第二张图像。这样便可将第一张和第二张图像之间的图像全部选中。
   d. 松开 Shift 键。
4. 选择分配图像级标签。



5. 在向选定图像分配图像级标签对话框中，选择要分配给该等图像的标签。
6. 选择分配，将标签分配给图像。



7. 重复标注，直到每张图像都用所需的标签进行注释。
8. 选择测试选项卡。
9. 重复上述步骤为测试数据集中的图像分配图像级标签。

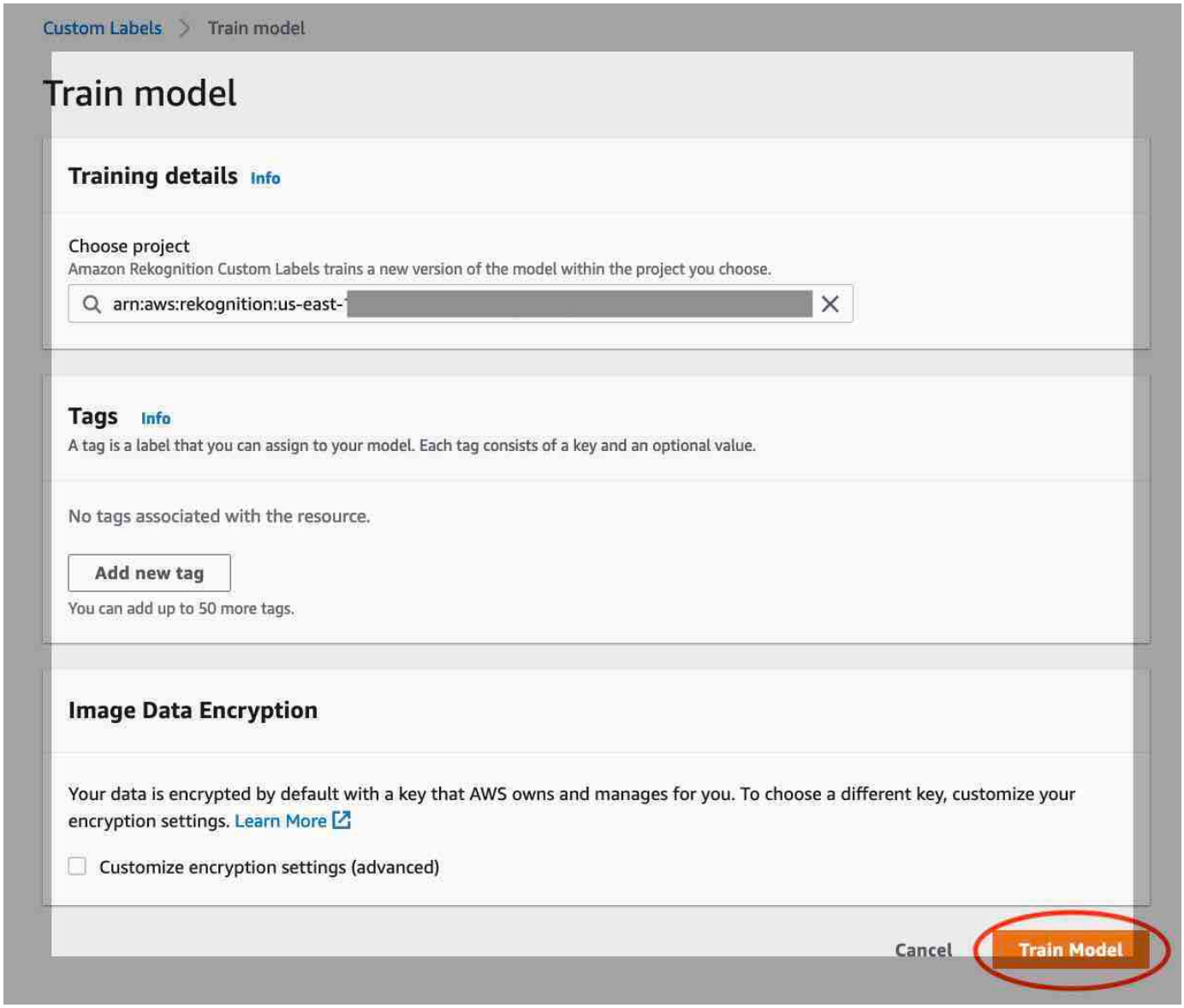
步骤 7：训练模型

按照以下步骤训练您的模型。有关更多信息，请参阅训练 Amazon Rekognition Custom Labels 模型。

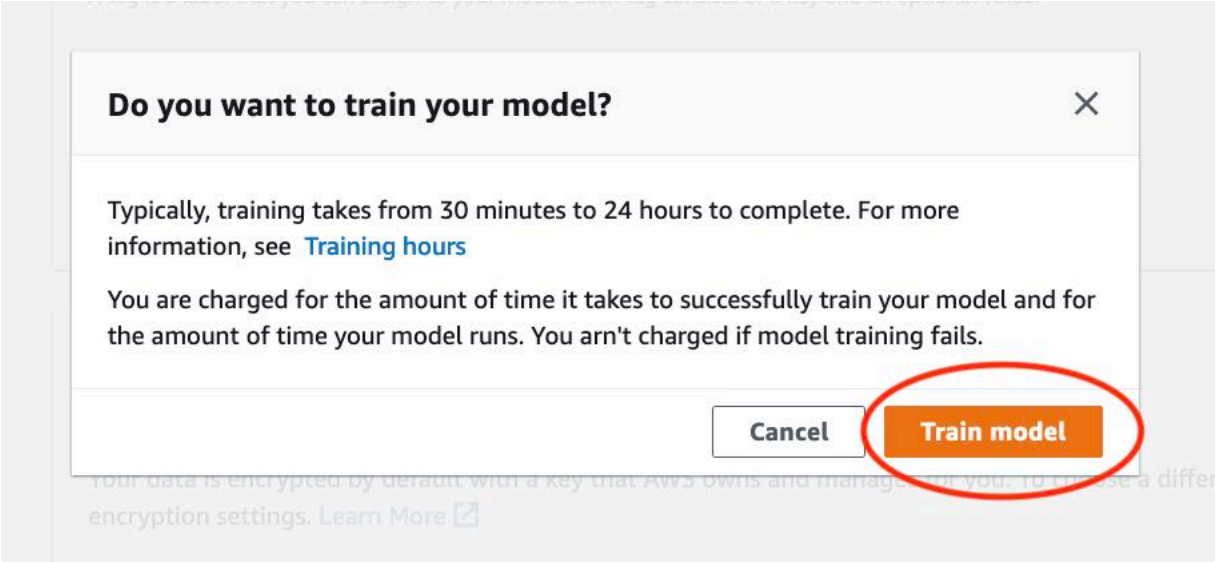
训练模型（控制台）

1. 在数据集页面上，选择训练模型。

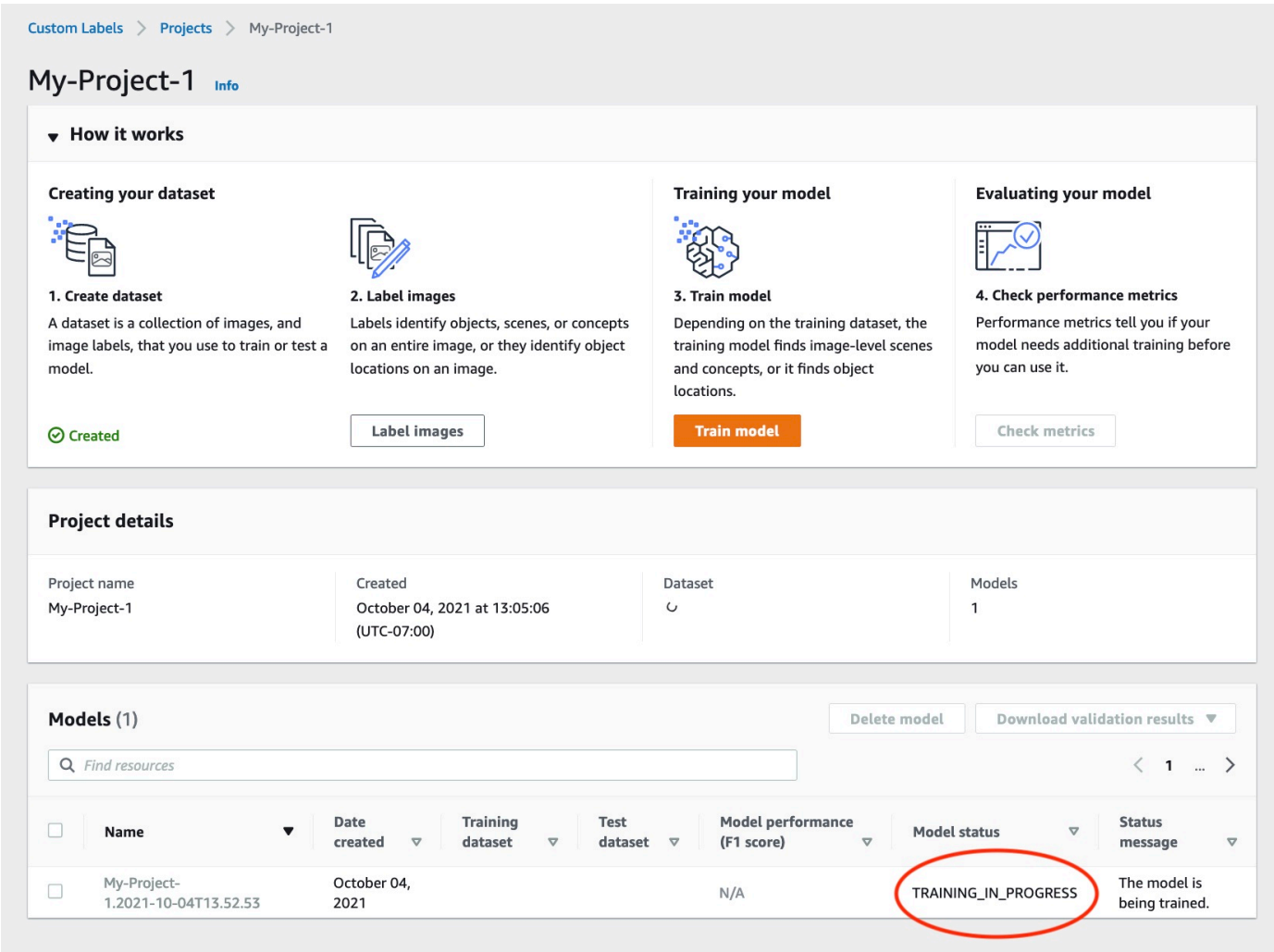
2. 在训练模型页面上，选择训练模型。项目的 Amazon 资源名称 (ARN) 位于选择项目编辑框中。



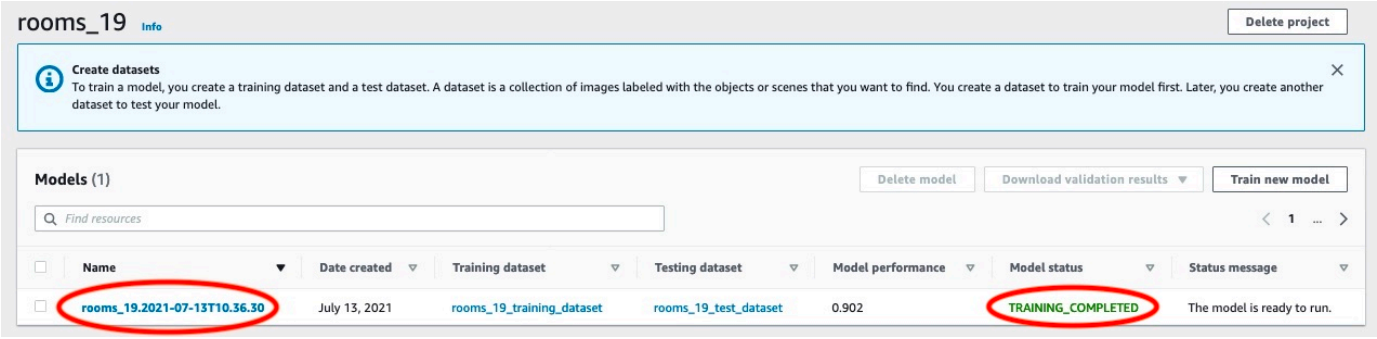
3. 在是否要训练您的模型？对话框中，选择训练模型。



4. 在项目页面的模型部分，您可以看到训练正在进行。可以通过查看模型版本的 `Model Status` 列，查看当前状态。训练模型需要一些时间才能完成。

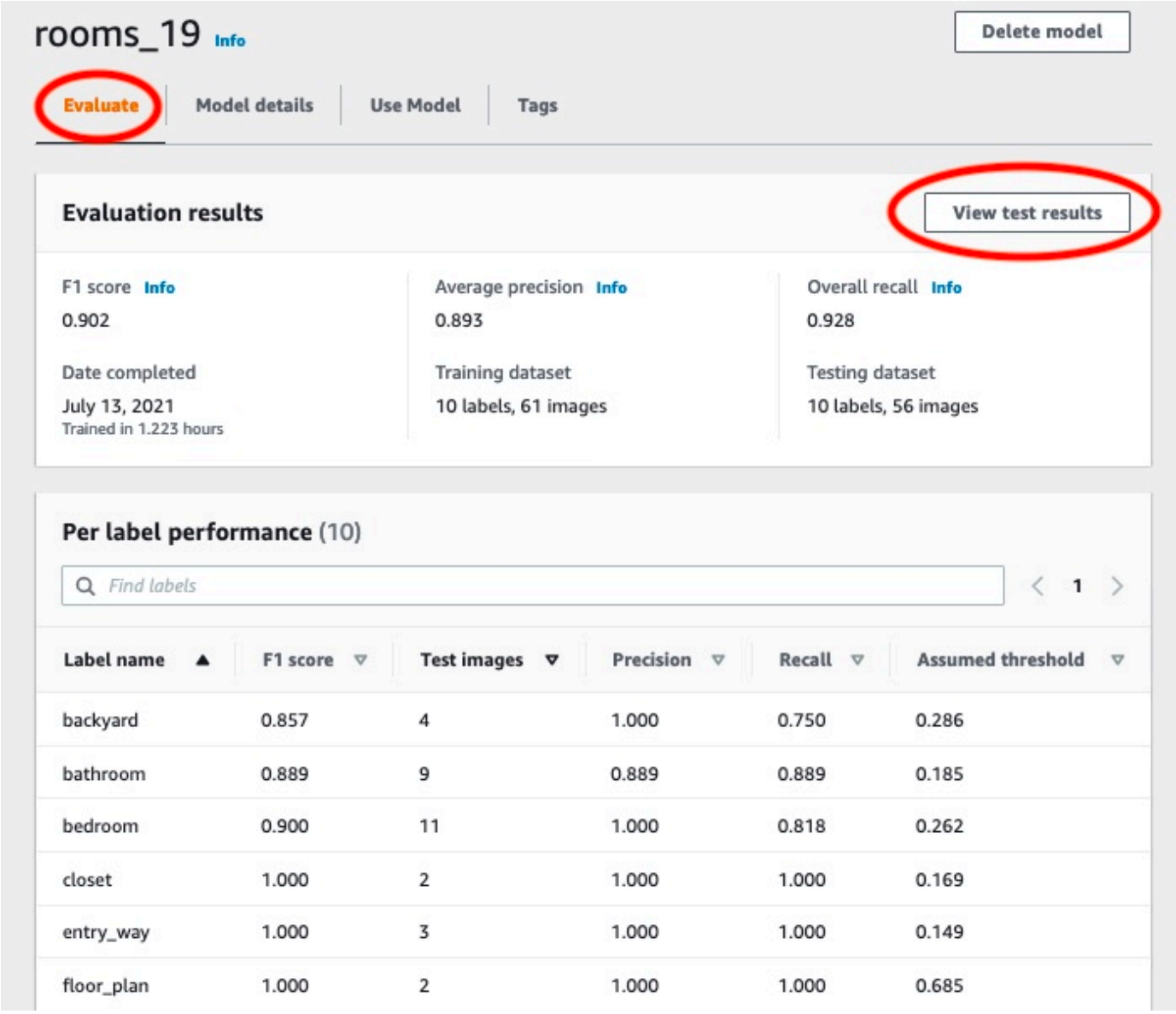


5. 训练完成后，选择模型名称。当模型状态为 TRAINING_COMPLETED 时，训练即告完成。

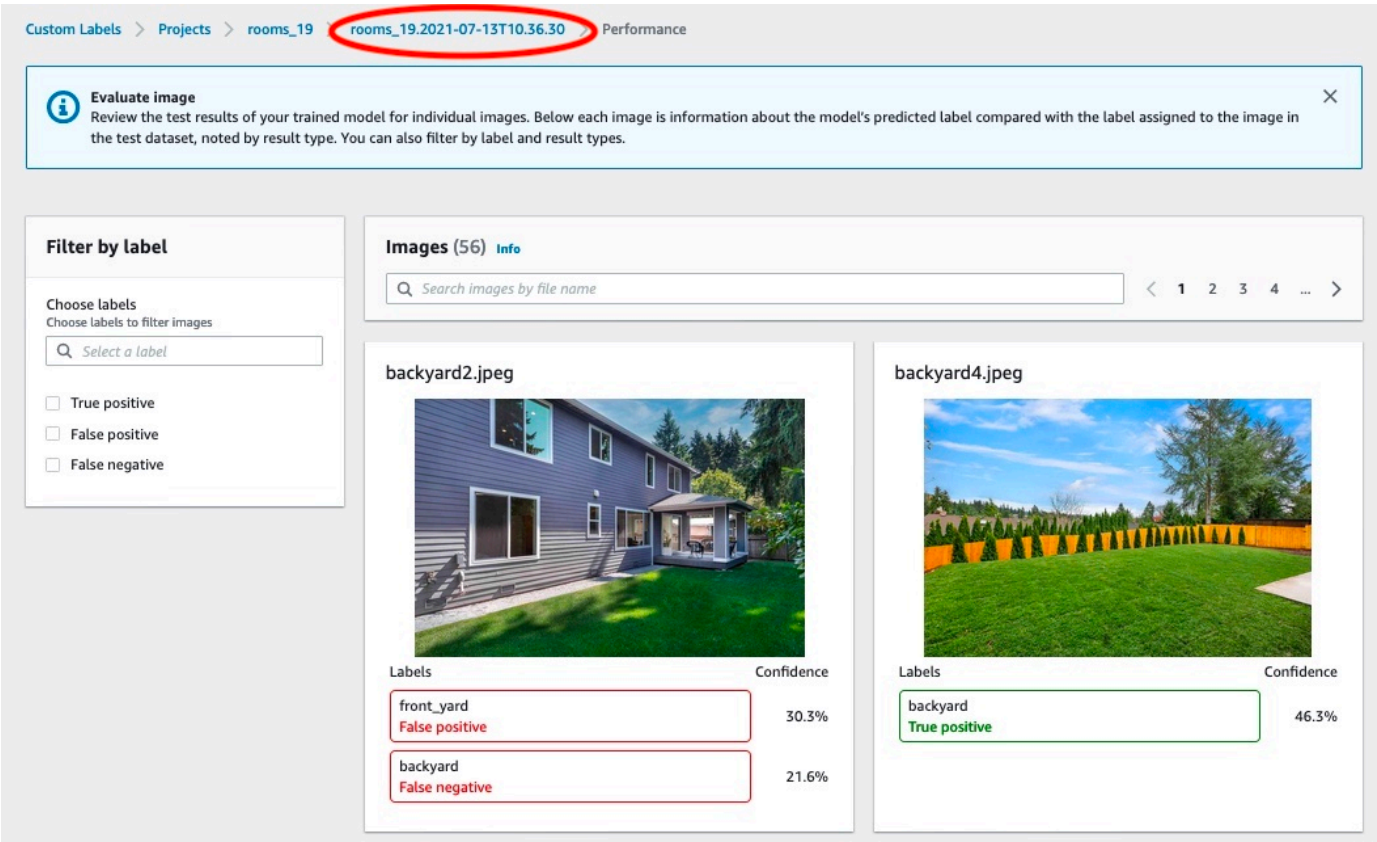


6. 选择评估按钮，以查看评估结果。有关评估模型的信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。

7. 选择查看测试结果，以查看单个测试图像的结果。有关更多信息，请参阅评估模型的指标。



8. 查看测试结果后，选择模型名称返回模型页面。



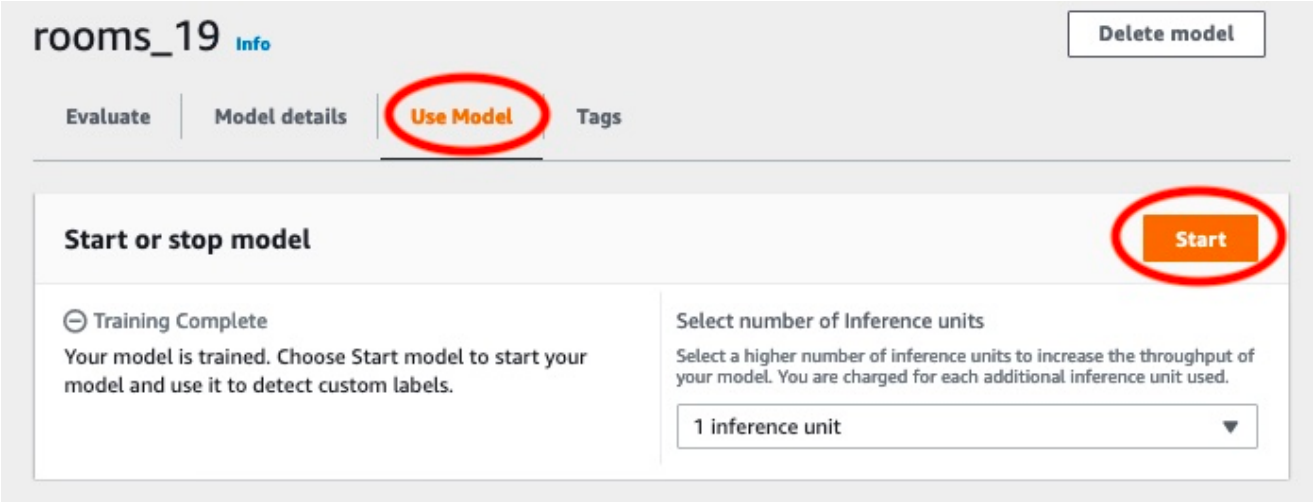
步骤 8：启动模型

在此步骤中，您将启动您的模型。模型启动后，您可以用它来分析图像。

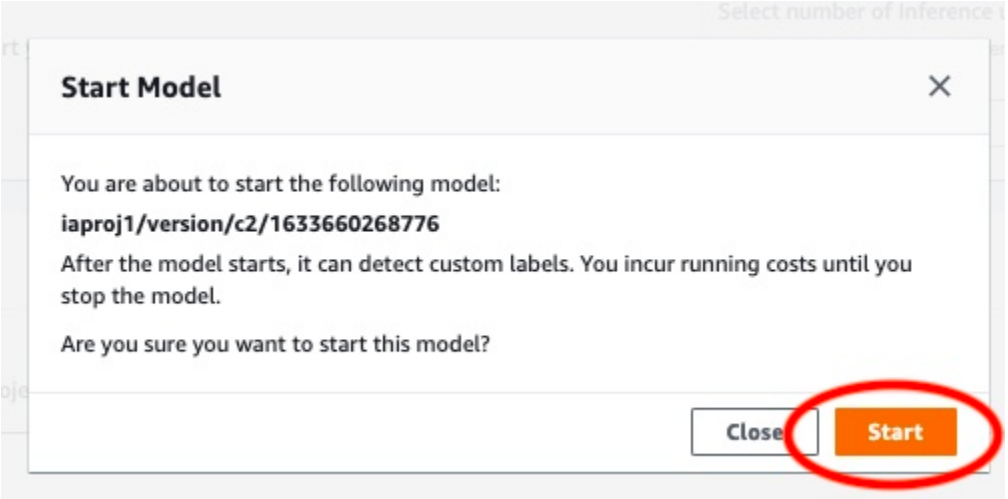
您将根据模型运行时间付费。如果您不需要分析图像，请停止模型。您可以稍后重新启动模型。有关更多信息，请参阅运行经过训练的 Amazon Rekognition Custom Labels 模型。

启动模型

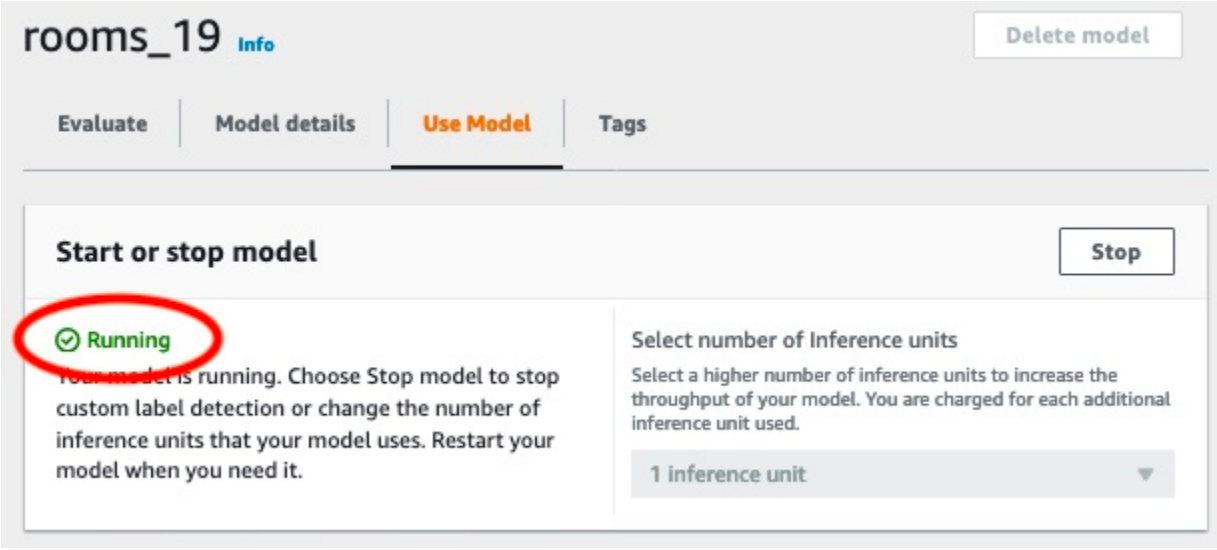
1. 在模型页面上选择使用模型选项卡。
2. 在启动或停止模型部分中，执行以下操作：
    a. 选择启动。



b. 在启动模型对话框中，选择启动。



3. 等到模型开始运行。当启动或停止模型部分中的状态为正在运行时，即表示模型正在运行。



步骤 9：使用模型分析图像

您可以通过调用 DetectCustomLabels API 来分析图像。在此步骤中，您将使用 `detect-custom-labels` AWS Command Line Interface (AWS CLI) 命令分析示例图像。您可以从 Amazon Rekognition Custom Labels 控制台获取 AWS CLI 命令。控制台将 AWS CLI 命令配置为使用您的模型。您只需要提供存储在 Amazon S3 存储桶中的图像即可。

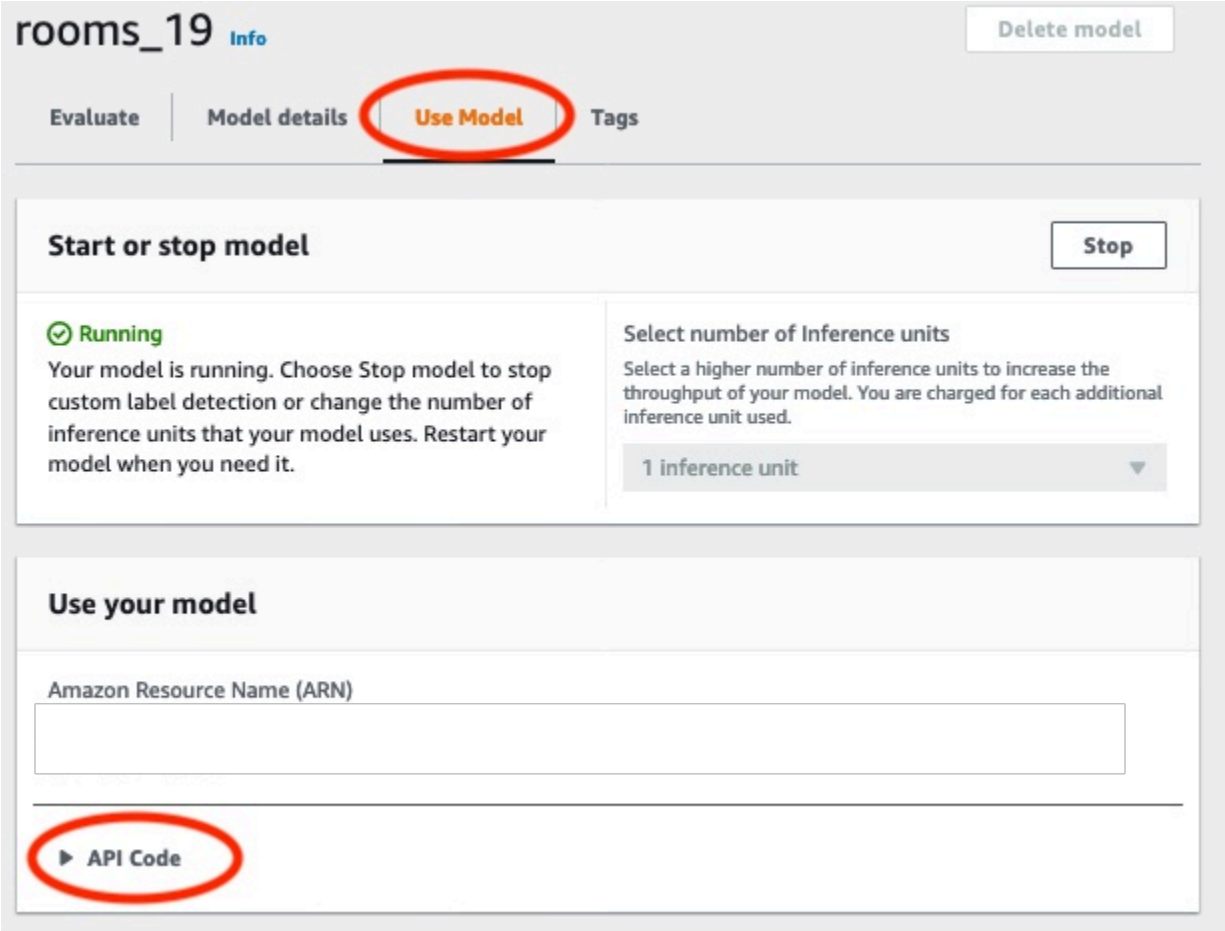
> **Note**
>
> 控制台还提供了调用 Python 示例代码。

`detect-custom-labels` 的输出包括在图像中找到的标签列表、边界框（如果模型会查找物体位置）以及模型对预测准确性的置信度。

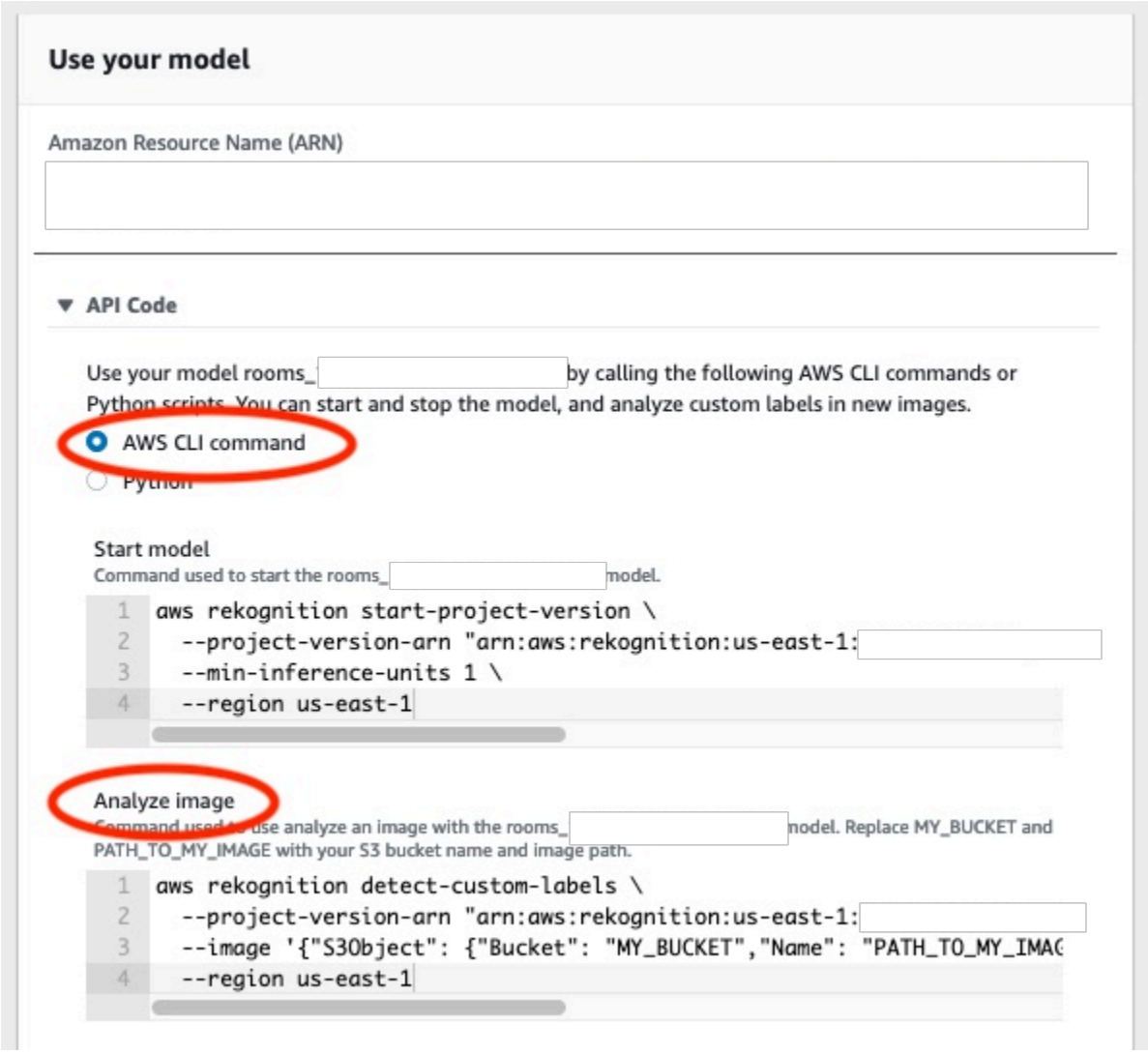
有关更多信息，请参阅使用经过训练的模型分析图像。

分析图像（控制台）

1. 设置 AWS CLI（如果尚未如此）。有关说明，请参阅the section called “步骤 4：设置 AWS CLI 和 AWS SDK”。
2. 选择使用模型选项卡，然后选择 API 代码。



3. 选择 AWS CLI 命令。
4. 在分析图像部分中，复制调用 `detect-custom-labels` 的 AWS CLI 命令。



5. 将图像上传到 Amazon S3 存储桶。有关说明，请参阅《Amazon Simple Storage Service 用户指南》中的将对象上传到 Amazon S3。如果您使用的是 Rooms 项目中的图像，请使用在步骤 1：收集图像中移动到单独文件夹中的图像之一。

6. 在命令提示符处，输入您在上一步中复制的 AWS CLI 命令。它应该类似于以下示例。

   `--project-version-arn` 的值应为模型的 Amazon 资源名称 (ARN)。`--region` 的值应为您在其中创建模型的 AWS 区域。

   将 `MY_BUCKET` 和 `PATH_TO_MY_IMAGE` 更改为您在上一步中使用的 Amazon S3 存储桶和图像。

   如果您使用 custom-labels-access 配置文件来获取凭证，请添加 `--profile custom-labels-access` 参数。

```
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
```

AWS CLI 命令的 JSON 输出应类似于以下内容。Name 是模型找到的图像级标签的名称。Confidence (0-100) 是模型对预测准确性的置信度。

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

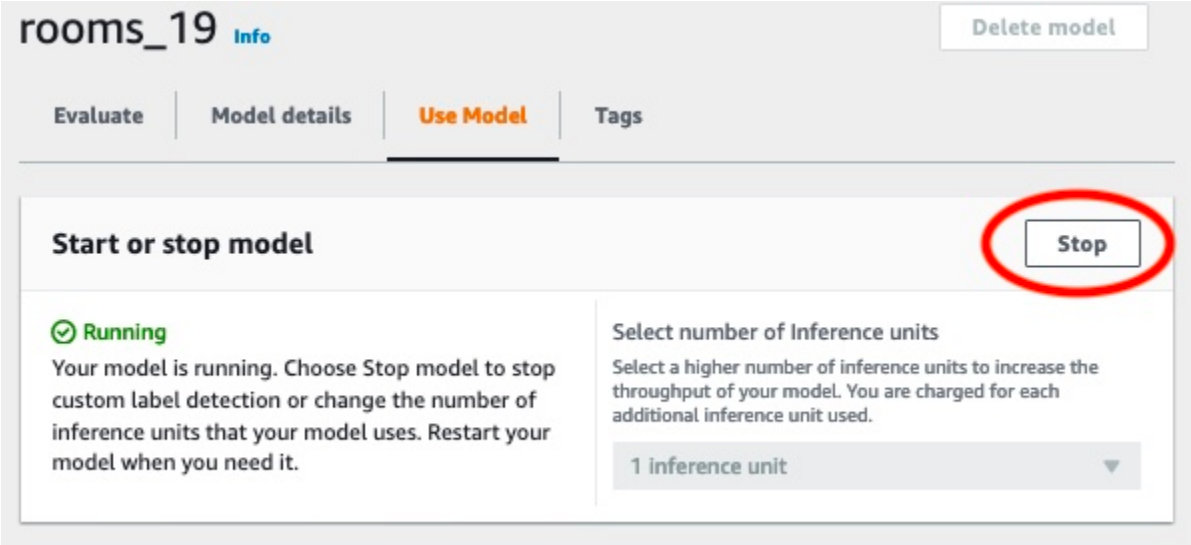
7. 继续使用该模型分析其他图像。如果不再使用该模型，请停止模型。

步骤 10：停止模型

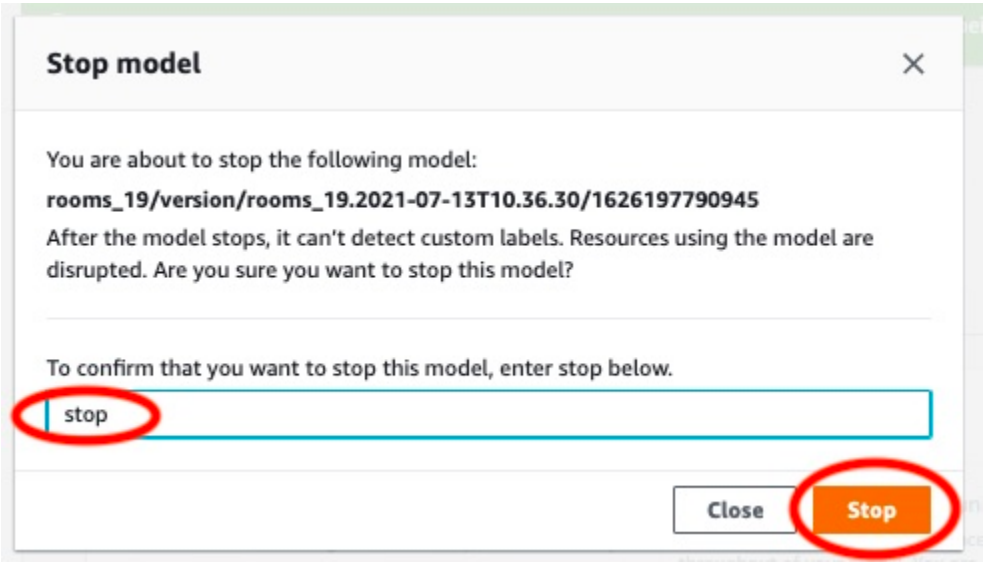
在此步骤中，您将停止运行您的模型。您需要按照模型运行的时间付费。如果您已使用完模型，应将其停止。

停止模型

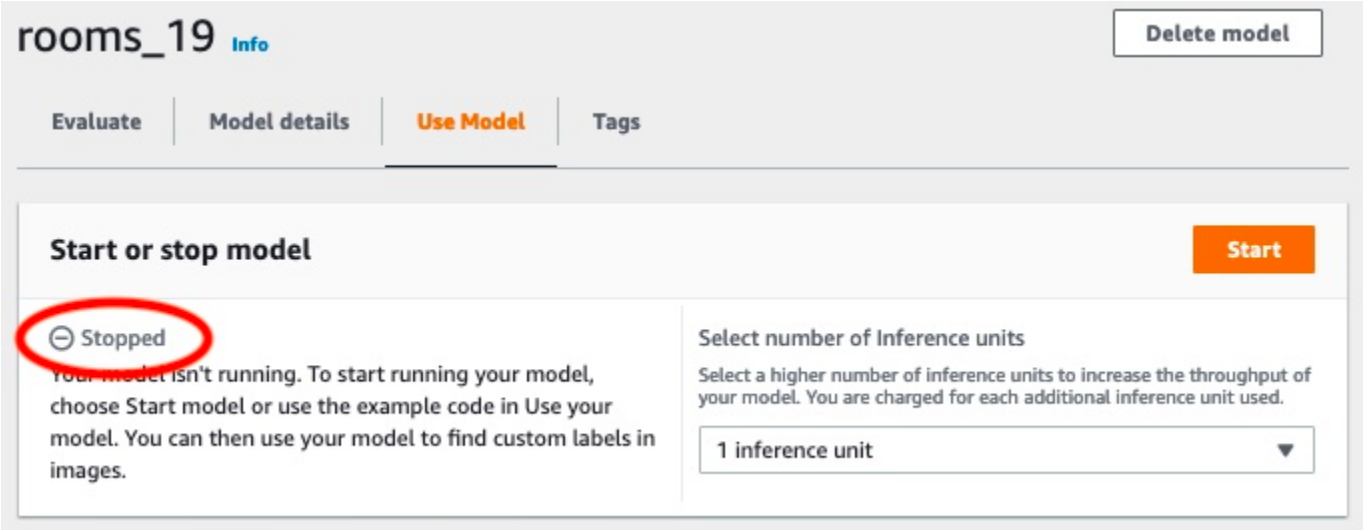
1. 在启动或停止模型部分中，选择停止。



2. 在停止模型对话框中，输入 stop 以确认要停止模型。



3. 选择停止以停止模型。当启动或停止模型部分中的状态为已停止时，即表示模型已停止。


rooms_19 Info
Delete model
Evaluate
Model details
Use Model
Tags
Start or stop model
Start
Stopped
isn't running. To start running your model, choose Start model or use the example code in Use your model. You can then use your model to find custom labels in images.
Select number of Inference units
Select a higher number of inference units to increase the throughput of your model. You are charged for each additional inference unit used.
1 inference unit

创建 Amazon Rekognition Custom Labels 模型

模型是训练来查找符合您业务需求的独特概念、场景和物体的软件。可以使用 Amazon Rekognition Custom Labels 控制台或 AWS SDK 创建模型。创建 Amazon Rekognition Custom Labels 模型之前，建议您先阅读了解 Amazon Rekognition Custom Labels。

本节介绍有关创建项目、为不同模型类型创建训练和测试数据集以及训练模型的控制台和 SDK 信息。后面的章节将介绍如何改进和使用模型。有关介绍如何通过控制台创建和使用特定类型模型的教程，请参阅教程：对图像进行分类。

主题

创建项目

项目用于管理模型的模型版本、训练数据集和测试数据集。可以使用 Amazon Rekognition Custom Labels 控制台或 API 创建项目。有关其他项目任务（例如删除项目），请参阅管理 Amazon Rekognition Custom Labels 项目。

创建 Amazon Rekognition Custom Labels 项目（控制台）

可以使用 Amazon Rekognition Custom Labels 控制台创建项目。首次在新 AWS 区域中使用控制台时，Amazon Rekognition Custom Labels 会要求在您的 AWS 账户中创建一个 Amazon S3 存储桶（控制台存储桶）。该存储桶用于存储您的项目文件。必须创建控制台存储桶，才能使用 Amazon Rekognition Custom Labels 控制台。

可以使用 Amazon Rekognition Custom Labels 控制台创建项目。

创建项目（控制台）

1. 通过以下网址登录 AWS Management Console 并打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。

3. 在 Amazon Rekognition Custom Labels 登录页面上，选择开始。
4. 在左侧窗格中，选择项目。
5. 选择创建项目。
6. 在项目名称中输入项目名称。
7. 选择创建项目，创建您的项目。
8. 按照创建训练和测试数据集中的步骤为项目创建训练和测试数据集。

创建 Amazon Rekognition Custom Labels 项目 (SDK)

可通过调用 CreateProject 创建 Amazon Rekognition Custom Labels 项目。响应是标识项目的 Amazon 资源名称 (ARN)。创建项目后，可以创建用于训练和测试模型的数据集。有关更多信息，请参阅使用图像创建训练和测试数据集。

创建项目 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下代码创建项目。

   AWS CLI

   以下示例会创建一个项目并显示其 ARN。

   将 `project-name` 的值更改为要创建的项目的名称。

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

   Python

   以下示例会创建一个项目并显示其 ARN。提供以下命令行参数：

   - `project_name`：要创建的项目的名称。

   ```
   # 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

以下示例会创建一个项目并显示其 ARN。

提供以下命令行参数：

- project_name：要创建的项目的名称。

```
/*
   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. 记下响应中显示的项目 ARN 的名称。您将需要它来创建模型。
4. 按照[创建训练和测试数据集 (SDK)](#) 中的步骤为项目创建训练和测试数据集。

创建训练和测试数据集

数据集是由图像和描述这些图像的标签组成的集合。项目需要一个训练数据集和一个测试数据集。Amazon Rekognition Custom Labels 使用训练数据集来训练模型。训练结束后，Amazon Rekognition Custom Labels 会使用测试数据集来验证训练后的模型预测正确标签的效果。

可以使用 Amazon Rekognition Custom Labels 控制台或 AWS SDK 创建数据集。在创建数据集之前，建议先阅读了解 Amazon Rekognition Custom Labels。有关其他数据集任务，请参阅管理数据集。

为项目创建训练和测试数据集的步骤如下：

为项目创建训练和测试数据集

1. 确定需要如何标注训练和测试数据集。有关更多信息，请参阅 确定数据集用途。
2. 为训练和测试数据集收集图像。有关更多信息，请参见 the section called “准备图像”。
3. 创建训练和测试数据集。有关更多信息，请参见 使用图像创建训练和测试数据集。如果使用的是 AWS SDK，请参阅创建训练和测试数据集 (SDK) 。
4. 如有必要，可以向数据集图像添加图像级标签或边界框。有关更多信息，请参见 标注图像。

创建数据集后，即可训练模型。

主题

- 确定数据集用途
- 准备图像
- 使用图像创建训练和测试数据集
- 标注图像
- 调试数据集

确定数据集用途

如何标注项目中的训练和测试数据集决定了所创建的模型的类型。使用 Amazon Rekognition Custom Labels，可以创建执行以下操作的模型。

- 查找物体、场景和概念
- 查找物体位置

- 查找品牌位置

查找物体、场景和概念

该模型可对与整个图像相关的物体、场景和概念进行分类。

您可以创建两种类型的分类模型：图像分类和多标签分类。这两种类型的分类模型都会从用于训练的整套标签中查找一个或多个匹配的标签。训练和测试数据集都需要至少两个标签。

图像分类

该模型可将图像归类为属于一组预定义的标签。例如，您可能需要一个模型来确定图像中是否包含生活空间。下图可能具有 living_space 图像级标签。

对于这种类型的模型，请为每个训练和测试数据集图像添加一个图像级标签。如需查看示例项目，请参阅图像分类。

多标签分类

该模型可将图像分为多个类别，例如花的类型以及是否有叶子。例如，下图可能具有 mediterranean_spurge 和 no_leaves 图像级标签。

对于这种类型的模型，请将每个类别的图像级标签分配给训练和测试数据集图像。如需查看示例项目，请参阅多标签图像分类。

分配图像级标签

如果图像存储在 Amazon S3 存储桶中，则可以使用文件夹名称自动添加图像级标签。有关更多信息，请参见 Amazon S3 存储桶。您也可以在创建数据集后向图像添加图像级标签。有关更多信息，请参阅the section called “为图像分配图像级标签”。您可以根据需要添加新标签。有关更多信息，请参见 管理标签。

查找物体位置

要创建预测图像中物体位置的模型，您需要为训练和测试数据集中的图像定义物体位置边界框和标签。边界框是紧紧围绕物体的方框。例如，下图显示了 Amazon Echo 和 Amazon Echo Dot 周围的边界框。每个边界框都有分配的标签（Amazon Echo 或 Amazon Echo Dot）。

要查找物体位置，数据集至少需要一个标签。在模型训练期间，会自动创建另一个标签，用于代表图像上边界框之外的区域。

分配边界框

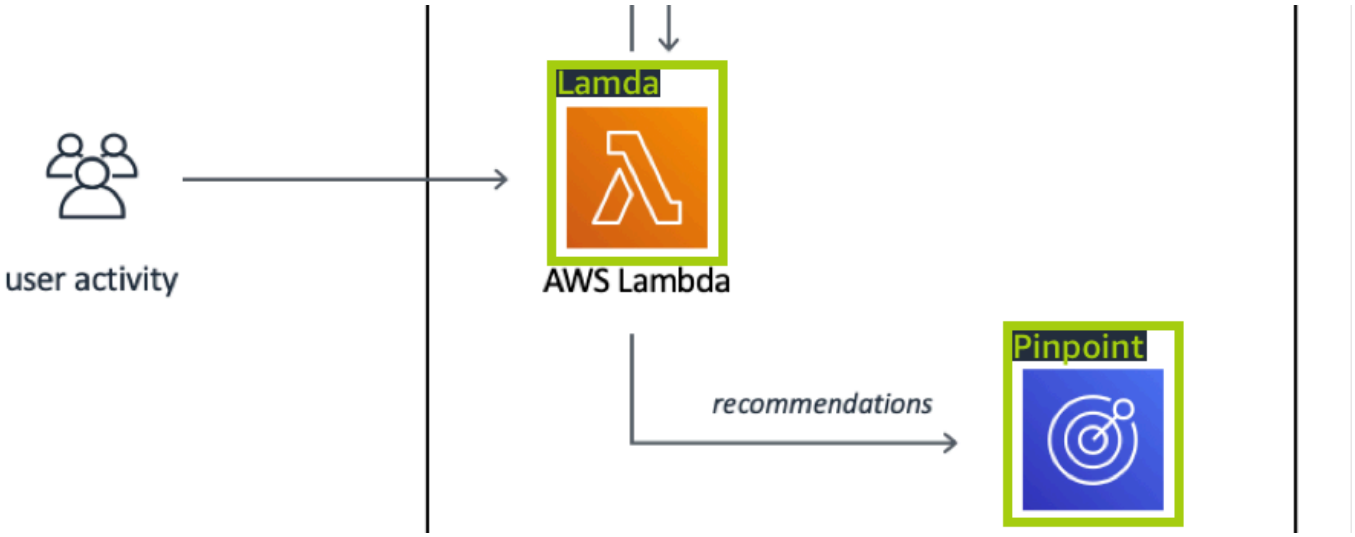
创建数据集时，可以包含图像的边界框信息。例如，您可以导入包含边界框的 SageMaker Ground Truth 格式的清单文件。或者，也可以在创建数据集之后添加边界框。有关更多信息，请参见 使用边界框标注物体。您可以根据需要添加新标签。有关更多信息，请参见 管理标签。

查找品牌位置

如果要查找品牌（例如徽标和动画角色）的位置，则可以为训练数据集图像使用两种不同类型的图像。

- 仅包含徽标的图像。每张图像都需要一个代表徽标名称的图像级标签。例如，下图的图像级标签可以是 Lambda。

- 在自然位置（例如足球比赛或架构图）包含徽标的图像。每张训练图像都需要围绕每个徽标实例的边界框。例如，下图显示了一张架构图，其中包含了围绕 AWS Lambda 和 Amazon Pinpoint 徽标且带有标签的边界框。



建议您不要在训练图像中混用图像级标签和边界框。

测试图像必须在您想要查找的品牌实例周围有边界框。只有当训练图像包含带标签的边界框时，才能拆分训练数据集来创建测试数据集。如果训练图像只有图像级标签，则必须创建一个测试数据集，该数据集中包含的图像必须具有带标签的边界框。如果训练模型来查找品牌位置，请根据为图像添加标签的方式执行以下操作：使用边界框标注物体和为图像分配图像级标签。

品牌检测 示例项目展示了 Amazon Rekognition Custom Labels 如何使用带标签的边界框来训练查找物体位置的模型。

不同模型类型的标签要求

根据下表确定如何为图像添加标签。

可以在单个数据集中混合带图像级标签的图像和包含带标签的边界框的图像。在这种情况下，Amazon Rekognition Custom Labels 会选择是创建图像级模型还是创建物体位置模型。

示例	训练图像	测试图像
图像分类	每张图像 1 个图像级标签	每张图像 1 个图像级标签
多标签分类	每张图像多个图像级标签	每张图像多个图像级标签
查找品牌位置	图像级标签（也可以使用带标签的边界框）	带标签的边界框
查找物体位置	带标签的边界框	带标签的边界框

准备图像

训练和测试数据集中的图像包含您希望模型查找的物体、场景或概念。

图像的内容应采用各种背景和光线，以代表您希望经过训练的模型识别的图像。

本节提供训练和测试数据集中的图像的相关信息。

图像格式

可以使用 PNG 和 JPEG 格式的图像训练 Amazon Rekognition Custom Labels 模型。同样，要使用 `DetectCustomLabels` 检测自定义标签，您需要采用 PNG 和 JPEG 格式的图像。

输入图像建议

Amazon Rekognition Custom Labels 需要图像来训练和测试模型。准备图像时，请考虑以下几点：

- 为要创建的模型选择一个特定的领域。例如，您可以为风景视图选择一个模型，为机器零件等物体选择另一个模型。当图像属于所选领域时，Amazon Rekognition Custom Labels 的效果最佳。
- 使用不少于 10 张图像来训练模型。
- 图像必须是 PNG 或 JPEG 格式。

- 使用在各种光线、背景和分辨率下显示该物体的图像。
- 训练和测试图像应与您要使用模型分析的图像相似。
- 决定要为图像分配的标签。
- 确保图像的分辨率足够大。有关更多信息，请参见 Amazon Rekognition Custom Labels 中的准则和配额。
- 确保遮挡不会遮蔽要检测的物体。
- 使用与背景对比鲜明的图像。
- 使用明亮清晰的图像。尽量避免使用可能因拍摄主体和相机移动而模糊的图像。
- 使用物体占图像很大比例的图像。
- 测试数据集中的图像不应是训练数据集中的图像。这些图像应包含训练模型来分析的物体、场景和概念。

图像集大小

Amazon Rekognition Custom Labels 使用一组图像来训练模型。至少应使用 10 张图像进行训练。Amazon Rekognition Custom Labels 将训练和测试图像存储在数据集中。有关更多信息，请参见使用图像创建训练和测试数据集。

使用图像创建训练和测试数据集

可以从具有单个数据集的项目或者具有单独的训练和测试数据集的项目开始。如果从单个数据集开始，Amazon Rekognition Custom Labels 会在训练期间拆分该数据集，来为项目创建训练数据集 (80%) 和测试数据集 (%20)。如果想让 Amazon Rekognition Custom Labels 决定使用哪些图像进行训练和哪些图像进行测试，请从单个数据集开始。为了能够完全控制训练、测试和性能调整，建议您使用单独的训练数据集和测试数据集开始您的项目。

可以通过从以下位置之一导入图像来为项目创建训练和测试数据集：

- Amazon S3 存储桶
- 本地计算机
- 清单文件
- 现有数据集

如果使用单独的训练和测试数据集来开始项目，可以为每个数据集使用不同的源位置。

根据导入图像的方式，您的图像可能没有标签。例如，从本地计算机导入的图像就没有标签。从 Amazon G SageMaker round Truth 清单文件导入的图片已贴上标签。您可以使用 Amazon Rekognition Custom Labels 控制台添加、更改和分配标签。有关更多信息，请参见 标注图像。

如果上传图像时出现错误、图像丢失或图像中缺少标签，请阅读调试失败的模型训练。

有关数据集的更多信息，请参阅管理数据集。

创建训练和测试数据集 (SDK)

可以使用 AWS API 创建训练和测试数据集。

训练数据集

可以通过以下方式使用 AWS SDK 创建训练数据集。

- CreateDataset与您提供的亚马逊 Sagemaker 格式的清单文件一起使用。有关更多信息，请参见 the section called “创建清单文件”。有关代码示例，请参阅 使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。
- 使用 `CreateDataset` 复制现有的 Amazon Rekognition Custom Labels 数据集。有关代码示例，请参阅 使用现有数据集创建数据集 (SDK)。
- 使用 `CreateDataset` 创建一个空数据集，稍后使用 UpdateDatasetEntries 添加数据集条目。要创建空数据集，请参阅向项目添加数据集。要向数据集中添加图像，请参阅添加更多图像 (SDK)。需要先添加数据集条目，然后才能训练模型。

测试数据集

可以通过以下方式使用 AWS SDK 创建测试数据集。

- CreateDataset与您提供的亚马逊 Sagemaker 格式的清单文件一起使用。有关更多信息，请参见 the section called “创建清单文件”。有关代码示例，请参阅 使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。
- 使用 `CreateDataset` 复制现有的 Amazon Rekognition Custom Labels 数据集。有关代码示例，请参阅 使用现有数据集创建数据集 (SDK)。
- 使用 `CreateDataset` 创建一个空数据集，稍后使用 `UpdateDatasetEntries` 添加数据集条目。要创建空数据集，请参阅向项目添加数据集。要向数据集中添加图像，请参阅添加更多图像 (SDK)。需要先添加数据集条目，然后才能训练模型。
- 将训练数据集拆分为单独的训练数据集和测试数据集。先使用 `CreateDataset` 创建一个空的测试数据集。然后通过调用，将 20% 的训练数据集条目移到测试数据集中DistributeDatasetEntries。

要创建空数据集，请参阅向项目添加数据集 (SDK)。要拆分训练数据集，请参阅分配训练数据集 (SDK)。

Amazon S3 存储桶

从 Amazon S3 存储桶中导入图像。可以使用控制台存储桶或 AWS 账户中的另一个 Amazon S3 存储桶。如果使用的是控制台存储桶，则所需权限已设置完毕。如果使用的不是控制台存储桶，请参阅访问外部 Amazon S3 存储桶。

> **Note**
>
> 不能使用 AWS SDK 直接从 Amazon S3 存储桶中的图像创建数据集。相反，应创建一个引用这些图像的源位置的清单文件。有关更多信息，请参阅 清单文件。

在创建数据集期间，可以选择根据包含图像的文件夹的名称，为图像分配标签名称。这些文件夹必须是您在创建数据集期间在 S3 文件夹位置中指定的 Amazon S3 文件夹路径的子文件夹。要创建数据集，请参阅通过从 S3 存储桶导入图像来创建数据集。

例如，假设 Amazon S3 存储桶中的文件夹结构如下。如果将 Amazon S3 文件夹位置指定为 S3-bucket/alexa-devices，则 echo 文件夹中的图像会被分配 echo 标签。同样，echo-dot 文件夹中的图像会被分配 echo-dot 标签。不会使用更深层子文件夹的名称来标注图像，而会使用 Amazon S3 文件夹位置的相应子文件夹。例如，文件夹中的图像white-echo-dots被分配了标签 e cho- dot。S3 文件夹位置 (alexa-devices) 层级的图像不会被分配标签。

可以通过指定更深层的 S3 文件夹位置，使用文件夹结构中的更深层文件夹来标注图像。例如，如果您指定 s3-bucket/Alexa-devices/echo-do t，则会标记文件夹中的图像。white-echo-dotwhite-echo-dot不会导入指定 s3 文件夹位置（例如 echo）之外的图像。

```
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
        ### .
        ### .
```

建议您使用首次在当前 AWS 区域中打开控制台时 Amazon Rekognition 为您创建的 Amazon S3 存储桶（控制台存储桶）。如果您使用的 Amazon S3 存储桶不是控制台存储桶，而是外部存储桶，则控制台会在创建数据集期间提示您设置适当的权限。有关更多信息，请参见 the section called “步骤 2：设置控制台权限”。

通过从 S3 存储桶导入图像来创建数据集

以下过程介绍如何使用存储在 S3 控制台存储桶中的图像来创建数据集。这些图像会自动使用存储它们的文件夹的名称进行标注。

导入图像后，就可以从数据集的图库页面添加更多图像、分配标签和添加边界框。有关更多信息，请参见 标注图像。

将图像上传到 Amazon Simple Storage Service 存储桶。

1. 在本地文件系统上创建文件夹。使用诸如 alexa-devices 之类的文件夹名称。
2. 在刚刚创建的文件夹中，创建以要使用的每个标签命名的文件夹。例如，echo 和 echo-dot。文件夹结构应类似于以下结构。

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

3. 将与标签对应的图像放入与标签同名的文件夹中。
4. 通过以下网址登录 AWS Management Console 并打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

5. 将您在步骤 1 中创建的文件夹添加到首次设置时由 Amazon Rekognition Custom Labels 为您创建的 Amazon S3 存储桶（控制台存储桶）中。有关更多信息，请参见 管理 Amazon Rekognition Custom Labels 项目。
6. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
7. 选择使用自定义标签。
8. 选择开始。
9. 在左侧导航窗格中，选择项目。
10. 在项目页面上，选择要向其添加数据集的项目。此时将显示项目的详细信息页面。
11. 选择创建数据集。此时将显示创建数据集页面。
12. 在开始配置中，选择从单个数据集开始或从训练数据集开始。要创建更高质量的模型，建议从单独的训练和测试数据集开始。

    Single dataset

    a. 在训练数据集详细信息部分中，选择从 S3 存储桶导入图像。
    b. 在训练数据集详细信息部分中，于图像源配置部分输入步骤 13-15 的信息。

    Separate training and test datasets

    a. 在训练数据集详细信息部分中，选择从 S3 存储桶导入图像。
    b. 在训练数据集详细信息部分中，于图像源配置部分输入步骤 13-15 的信息。
    c. 在测试数据集详细信息部分中，选择从 S3 存储桶导入图像。
    d. 在测试数据集详细信息部分中，于图像源配置部分输入步骤 13-15 的信息。
13. 选择从 Amazon S3 存储桶导入图像。
14. 在 S3 URI 中，输入 Amazon S3 存储桶的位置和文件夹路径。
15. 选择根据文件夹自动为图像附加标签。
16. 选择创建数据集。这时会打开项目的数据集页面。
17. 如果需要添加或更改标签，请执行标注图像中的操作。
18. 按照训练模型（控制台）中的步骤训练您的模型。

本地计算机

直接从您的计算机加载图像。一次最多可以上传 30 张图像。

您上传的图像不会有与之关联的标签。有关更多信息，请参见 标注图像。如果需要上传的图像很多，请考虑使用 Amazon S3 存储桶。有关更多信息，请参见 Amazon S3 存储桶。

> **Note**
>
> 无法通过 AWS SDK 创建包含本地图像的数据集。您可以创建一个清单文件，然后将图像上传到 Amazon S3 存储桶。有关更多信息，请参见 清单文件。

使用本地计算机上的图像创建数据集（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要向其添加数据集的项目。此时将显示项目的详细信息页面。
6. 选择创建数据集。此时将显示创建数据集页面。
7. 在开始配置中，选择从单个数据集开始或从训练数据集开始。要创建更高质量的模型，建议从单独的训练和测试数据集开始。

   Single dataset

   a. 在训练数据集详细信息部分中，选择从您的计算机上传图像。
   b. 选择创建数据集。
   c. 在项目的数据集页面上，选择添加图像。
   d. 从计算机文件中选择要上传到数据集的图像。可以从本地计算机拖动或选择要上传的图像。
   e. 选择上传图像。

   Separate training and test datasets

   a. 在训练数据集详细信息部分中，选择从您的计算机上传图像。
   b. 在测试数据集详细信息部分中，选择从您的计算机上传图像。

   > **Note**
   >
   > 训练数据集和测试数据集可以有不同的图像源。

c. 选择创建数据集。此时会显示项目的数据集页面，其中包含训练选项卡和测试选项卡，分别对应于相应的数据集。

d. 选择操作，然后选择将图像添加到训练数据集。

e. 选择要上传到数据集的图像。可以从本地计算机拖动或选择要上传的图像。

f. 选择上传图像。

g. 重复步骤 5e-5g。在步骤 5e 中，选择操作，然后选择将图像添加到测试数据集。

8. 按照标注图像中的步骤为图像添加标签。
9. 按照训练模型（控制台）中的步骤训练您的模型。

清单文件

您可以使用 Amazon G SageMaker round Truth 格式的清单文件创建数据集。你可以使用 Amazon G SageMaker round Truth 任务中的清单文件。如果您的图像和标签不是 G SageMaker round Truth 清单文件的格式，则可以创建 SageMaker 格式清单文件并使用它来导入已贴标签的图像。

主题

- 使用 G SageMaker round Truth 清单文件创建数据集（控制台）
- 使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集
- 亚马逊 G SageMaker round Truth 工作
- 创建清单文件
- 将其他数据集格式转换为清单文件

使用 G SageMaker round Truth 清单文件创建数据集（控制台）

以下过程向您展示如何使用 G SageMaker round Truth 格式的清单文件创建数据集。

1. 通过执行下列操作之一，为训练数据集创建清单文件：

   - 按照中的说明创建包含 SageMaker GroundTruth Job 的清单文件亚马逊 G SageMaker round Truth 工作。
   - 按照创建清单文件中的说明创建您自己的清单文件。

   如果要创建测试数据集，请重复步骤 1 以创建测试数据集。
2. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。

3. 选择使用自定义标签。
4. 选择开始。
5. 在左侧导航窗格中，选择项目。
6. 在项目页面上，选择要向其添加数据集的项目。此时将显示项目的详细信息页面。
7. 选择创建数据集。此时将显示创建数据集页面。
8. 在开始配置中，选择从单个数据集开始或从训练数据集开始。要创建更高质量的模型，建议从单独的训练和测试数据集开始。

   Single dataset

   a. 在训练数据集详细信息部分，选择导入 SageMaker 由 Ground Truth 标注的图像。
   b. 在 .manifest 文件位置中，输入您在步骤 1 中创建的清单文件的位置。
   c. 选择创建数据集。这时会打开项目的数据集页面。

   Separate training and test datasets

   a. 在训练数据集详细信息部分，选择导入 SageMaker 由 Ground Truth 标注的图像。
   b. 在 .manifest 文件位置中，输入您在步骤 1 中创建的训练数据集清单文件的位置。
   c. 在测试数据集详细信息部分，选择导入 SageMaker 由 Ground Truth 标注的图像。

   > **Note**
   > 训练数据集和测试数据集可以有不同的图像源。

   d. 在 .manifest 文件位置中，输入您在步骤 1 中创建的测试数据集清单文件的位置。
   e. 选择创建数据集。这时会打开项目的数据集页面。
9. 如果需要添加或更改标签，请执行标注图像中的操作。
10. 按照训练模型（控制台）中的步骤训练您的模型。

使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集

以下过程向您展示如何使用 CreateDatasetAPI 从清单文件创建训练或测试数据集。

您可以使用现有的清单文件，例如 G SageMaker round Truth 任务的输出，也可以创建自己的清单文件。

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参见 步骤 4：设置 AWS CLI 和 AWS SDK。
2. 通过执行下列操作之一，为训练数据集创建清单文件：

   - 按照中的说明创建包含 SageMaker GroundTruth Job 的清单文件亚马逊 G SageMaker round Truth 工作。
   - 按照创建清单文件中的说明创建您自己的清单文件。

   如果要创建测试数据集，请重复步骤 2 以创建测试数据集。
3. 使用以下示例代码创建训练和测试数据集。

   AWS CLI

   使用以下代码创建数据集。替换以下内容：

   - `project_arn`：要添加测试数据集的项目的 ARN。
   - `type`：要创建的数据集的类型（TRAIN 或 TEST）
   - `bucket`：包含数据集清单文件的存储桶。
   - `manifest_file`：清单文件的路径和文件名。

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

   Python

   使用以下值创建数据集。提供以下命令行参数：

   - `project_arn`：要添加测试数据集的项目的 ARN。
   - `dataset_type`：要创建的数据集的类型（`train` 或 `test`）。
   - `bucket`：包含数据集清单文件的存储桶。
   - `manifest_file`：清单文件的路径和文件名。

   ```
   #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

使用以下值创建数据集。提供以下命令行参数：

- project_arn：要添加测试数据集的项目的 ARN。
- dataset_type：要创建的数据集的类型（train 或 test）。
- bucket：包含数据集清单文件的存储桶。
- manifest_file：清单文件的路径和文件名。

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
```

```
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;

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

public class CreateDatasetManifestFiles {

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

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

        try {

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

            DatasetType requestDatasetType = null;

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

            }
```

```
            GroundTruthManifest groundTruthManifest =
GroundTruthManifest.builder()

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

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

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

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

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            do {

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

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

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

                switch (status) {

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

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

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

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

            } while (created == false);

            return response.datasetArn();

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

    }

    public static void main(String[] args) {

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

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
```

```
                + "   bucket - the S3 bucket that contains the manifest file.\n
\n"
                + "   name - the location and name of the manifest file within
 the bucket.\n\n";

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

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

        try {

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


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

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

            rekClient.close();

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

    }
```

```
}
```

4. 如果需要添加或更改标签，请参阅管理标签 (SDK)。
5. 按照训练模型 (SDK)中的步骤训练您的模型。

亚马逊 G SageMaker round Truth 工作

借助 Amazon G SageMaker round Truth，您可以使用来自您选择的供应商公司 Amazon Mechanical Turk 的工作人员，也可以使用内部私人员工以及允许您创建带标签的图像集的机器学习。亚马逊 Rekognition 自定义标签会从你指定的亚马逊 S3 存储桶中导入 SageMaker Ground Truth 清单文件。

亚马逊 Rekognition 自定义标签支持以下 SageMaker Ground Truth 任务。

- 图像分类
- 边界框

导入的文件包括图像和清单文件。清单文件中包含导入的图像的标签和边界框信息。

Amazon Rekognition 需要具有访问存储图像的 Amazon S3 存储桶的权限。如果使用的是 Amazon Rekognition Custom Labels 为您设置的控制台存储桶，则所需权限已设置完毕。如果使用的不是控制台存储桶，请参阅访问外部 Amazon S3 存储桶。

使用 G SageMaker round Truth 任务创建清单文件（控制台）

以下过程向您展示如何使用由 G SageMaker round Truth 作业标记的图像来创建数据集。该作业的输出文件存储在您的 Amazon Rekognition Custom Labels 控制台存储桶中。

使用由 G SageMaker round Truth 作业标记的图像创建数据集（控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在控制台存储桶中，创建一个文件夹来存放训练图像。

> **Note**
>
> 控制台存储桶是在您首次在 AWS 区域中打开 Amazon Rekognition Custom Labels 控制台时创建的。有关更多信息，请参见 管理 Amazon Rekognition Custom Labels 项目。

3. 上传图像至您刚才创建的文件夹。

4. 在控制台存储桶中，创建一个文件夹来存放 Ground Truth 作业的输出。
5. 打开 SageMaker 控制台，网址为 https://console.aws.amazon.com/sagemaker/。
6. 创建 Ground Truth 标注作业。您需要在步骤 2 和步骤 4 中创建的文件夹的 Amazon S3 URL。有关更多信息，请参阅使用 Amazon G SageMaker round Truth 进行数据标签。
7. 记下 `output.manifest` 文件在步骤 4 中创建的文件夹中的位置。它应该位于子文件夹 *`Ground-Truth-Job-Name`*`/manifests/output` 中。
8. 按照使用 G SageMaker round Truth 清单文件创建数据集（控制台）中的说明，使用上传的清单文件创建数据集。对于步骤 8，在 .manifest 文件位置中，输入您在上一步中记下的该位置的 Amazon S3 URL。如果使用的是 AWS SDK，请执行使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。
9. 重复步骤 1-6，为您的测试数据集创建 SageMaker Ground Truth 作业。

创建清单文件

您可以通过导入 G SageMaker round Truth 格式的清单文件来创建测试或训练数据集。如果您的图像的标签格式不是 G SageMaker round Truth 清单文件，请使用以下信息创建 G SageMaker round Truth 格式的清单文件。

清单文件采用 JSON 行格式，其中的每一行都是一个代表图像标注信息的完整 JSON 对象。亚马逊 Rekognition 自定义标签支持 SageMaker 带有以下格式的 JSON 行的 Ground Truth 清单：

- 分类作业输出：用于向图像添加图像级标签。图像级标签定义了图像上的场景、概念或物体（如果不需要物体位置信息）的类别。一张图像可以有多个图像级标签。有关更多信息，请参见 清单文件中的图像级标签。
- 边界框作业输出：用于标注图像上一个或多个物体的类别和位置。有关更多信息，请参见 清单文件中的物体定位。

图像级和定位（边界框）JSON 行可在同一个清单文件中链接在一起。

> **Note**
>
> 本部分的 JSON 行示例为便于阅读调整了格式。

当您导入清单文件时，Amazon Rekognition Custom Labels 会应用关于限制、语法和语义的验证规则。有关更多信息，请参见 清单文件的验证规则。

清单文件引用的图像必须位于同一 Amazon S3 存储桶中。清单文件与图像可以位于不同于的 Amazon S3 存储桶中。您应在 JSON 行的 `source-ref` 字段中指定图像的位置。

Amazon Rekognition 需要具有访问存储图像的 Amazon S3 存储桶的权限。如果使用的是 Amazon Rekognition Custom Labels 为您设置的控制台存储桶，则所需权限已设置完毕。如果使用的不是控制台存储桶，请参阅访问外部 Amazon S3 存储桶。

主题

- 创建清单文件
- 清单文件中的图像级标签
- 清单文件中的物体定位
- 清单文件的验证规则

创建清单文件

以下过程创建包含训练和测试数据集的项目。这些数据集是从您创建的训练和测试清单文件创建的。

使用 G SageMaker round Truth 格式的清单文件创建数据集（控制台）

1. 在控制台存储桶中，创建一个文件夹来存放清单文件。
2. 在控制台存储桶中，创建一个文件夹来存放图像。
3. 上传图像至您刚才创建的文件夹。
4. 为您的训练数据集创建 G SageMaker round Truth 格式的清单文件。有关更多信息，请参阅 清单文件中的图像级标签和 清单文件中的物体定位：

   > **Important**
   >
   > 每个 JSON 行中的 `source-ref` 字段值必须映射到您上传的一张图像。

5. 为您的测试数据集创建 G SageMaker round Truth 格式的清单文件。
6. 上传清单文件至您刚才创建的文件夹。
7. 记下清单文件的位置。
8. 按照使用 G SageMaker round Truth 清单文件创建数据集（控制台）中的说明，使用上传的清单文件创建数据集。对于步骤 8，在 .manifest 文件位置中，输入您在上一步中记下的该位置的 Amazon S3 URL。如果使用的是 AWS SDK，请执行使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。

清单文件中的图像级标签

要导入图像级标签（标有场景、概念或不需要本地化信息的对象的图像），可以将 G SageMaker round Truth 分类作业输出格式 JOB 行添加到清单文件中。清单文件由一个或多个 JSON 行组成，每个行对应一张您要导入的图像。

> **Tip**
>
> 为了简化清单文件的创建，我们提供了从 CSV 文件创建清单文件的 Python 脚本。有关更多信息，请参见 从 CSV 文件创建清单文件。

创建图像级标签的清单文件

1. 创建一个空文本文件。
2. 为要导入的每张图像各添加一个 JSON 行。每个 JSON 行应该与下面类似。

```
{"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. 保存该文件。您可以使用扩展名 `.manifest`，但不要求必须如此。
4. 使用您创建的清单文件，创建一个数据集。有关更多信息，请参见 使用 G SageMaker round Truth 格式的清单文件创建数据集（控制台）。

图像级 JSON 行

在本部分中，我们将介绍如何为单张图像创建 JSON 行。请考虑以下图像。下图的场景可能叫做 Sunrise。

上面这张包含 Sunrise 场景的图像的 JSON 行可能如下所示。

```
{
    "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"
    }
}
```

请注意以下信息。

source-ref

（必需）图像的 Amazon S3 位置。格式为 "s3://*BUCKET*/*OBJECT_PATH*"。所导入数据集中的图像必须存储在同一 Amazon S3 存储桶中。

testdataset-classification_Sunrise

（必需）标签属性。您选择的字段名称。该字段的值（在前面的示例中为 1）是标签属性的标识符。Amazon Rekognition Custom Labels 不使用它，它可以是任何整数值。必须有相应的元数据，这些元数据通过字段名称进行标识，并附有 -metadata。例如，"testdataset-classification_Sunrise-metadata"。

testdataset-classification_Sunrise-metadata

（必需）与标签属性相关的元数据。该字段名称必须与标签属性附加 -metadata 之后相同。

confidence

（必需）Amazon Rekognition Custom Labels 目前未使用该属性，但必须为其提供介于 0 和 1 之间的值。

job-name

（可选）您为用于处理图像的作业选择的名称。

class-name

（必需）您为适用于图像的场景或概念选择的类别名称。例如，"Sunrise"。

human-annotated

（必需）如果注释由人工完成，请指定 "yes"。否则为 "no"。

creation-date

（必需）创建标签的协调世界时 (UTC) 日期和时间。

类型

（必需）应该应用于图像的处理类型。对于图像级标签，该值为 "groundtruth/image-classification"。

为图像添加多个图像级标签

可以为图像添加多个标签。例如，以下 JSON 为单个图像添加两个标签：football 和 ball。

```
{
    "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
```

清单文件中的物体定位

通过将 G SageMaker round Truth Bounding Box Job Output 格式的 JSON 行添加到清单文件中，可以导入标有对象本地化信息的图像。

定位信息表示物体在图像上的位置。该位置由围绕物体的边界框表示。边界框结构包含边界框的左上角坐标和边界框的宽度与高度。边界框格式的 JSON 行包含图像上一个或多个物体的位置的边界框以及图像上每个物体的类别。

清单文件由一个或多个 JSON 行组成，每行包含单张图像的信息。

创建物体定位的清单文件

1. 创建一个空文本文件。
2. 为要导入的每张图像各添加一个 JSON 行。每个 JSON 行应该与下面类似。

```
{"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. 保存该文件。您可以使用扩展名 `.manifest`，但不要求必须如此。
4. 使用您刚才创建的文件创建数据集。有关更多信息，请参见 使用 G SageMaker round Truth 格式的清单文件创建数据集（控制台）。

物体边界框 JSON 行

在本部分中，我们将介绍如何为单张图像创建 JSON 行。下图显示了 Amazon Echo 和 Amazon Echo Dot 设备周围的边界框。

以下是上面这张图像的边界框 JSON 行。

```
{
 "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"
  }
}
```

请注意以下信息。

source-ref

（必需）图像的 Amazon S3 位置。格式为 "s3://*BUCKET*/*OBJECT_PATH*"。所导入数据集中的图像必须存储在同一 Amazon S3 存储桶中。

bounding-box

（必需）标签属性。您选择的字段名称。包含图像大小和图像中检测到的每个物体的边界框。必须有相应的元数据，这些元数据通过字段名称进行标识，并附有 -metadata。例如，"bounding-box-metadata"。

image_size

（必需）包含图像大小（以像素为单位）的单个元素数组。

- height：（必需）图像的高度（以像素为单位）。

- width：（必需）图像的深度（以像素为单位）。
- depth：（必需）图像的通道数。对于 RGB 图像，该值为 3。Amazon Rekognition Custom Labels 目前未使用该属性，但必须为其提供一个值。

annotations

（必需）图像中检测到的每个物体的边界框信息数组。

- class_id：（必需）映射到 class-map 中的标签。在上面的示例中，class_id 为 1 的物体是图像中的 Echo Dot。
- top：（必需）从图像顶部到边界框顶部的距离（以像素为单位）。
- left：（必需）从图像左侧到边界框左侧的距离（以像素为单位）。
- width：（必需）边界框的宽度（以像素为单位）。
- height：（必需）边界框的高度（以像素为单位）。

bounding-box-metadata

（必需）与标签属性相关的元数据。该字段名称必须与标签属性附加 -metadata 之后相同。图像中检测到的每个物体的边界框信息数组。

Objects

（必需）图像中的物体数组。按索引映射到 annotations 数组。Amazon Rekognition Custom Labels 不使用 confidence 属性。

class-map

（必需）适用于图像中检测到的物体的类别映射。

类型

（必需）分类作业的类型。`"groundtruth/object-detection"` 将作业标识为物体检测。

creation-date

（必需）创建标签的协调世界时 (UTC) 日期和时间。

human-annotated

（必需）如果注释由人工完成，请指定 `"yes"`。否则为 `"no"`。

job-name

（可选）处理图像的作业的名称。

清单文件的验证规则

当您导入清单文件时，Amazon Rekognition Custom Labels 会应用关于限制、语法和语义的验证规则。G SageMaker round Truth 架构强制执行语法验证。有关更多信息，请参阅输出。以下是限制和语义的验证规则。

> **Note**
>
> - 所有验证规则累计遵循 20% 的无效规则。如果由于任意组合（例如 15% 的无效 JSON 和 15% 的无效图像）导致导入超过 20% 的限制，则导入将失败。
> - 每个数据集对象都对应于清单中的一行。空行/无效行也算作数据集对象。
> - 重叠度为（测试和训练之间的共用标签数）/（训练标签数）。

主题

- 限制
- 语义

限制

验证	限制	出现错误
清单文件大小	最大 1 GB	错误
清单文件的最大行数	一个清单中最多可以包含 250,000 个数据集对象行。	错误
每个标签的有效数据集对象总数的下限	>=1	错误
标签数下限	>=2	错误
标签数上限	<=250	错误
每张图像的最小边界框数	0	无
每张图像的最大边界框数	50	无

语义

验证	限制	出现错误
空清单		错误
source-ref 对象丢失/无法访问	对象数量小于 20%	Warning
source-ref 对象丢失/无法访问	对象数量 > 20%	错误
训练数据集中不存在测试标签	标签中至少有 50% 的重叠	错误
在数据集中混合标签与同一标签的对象示例。针对同一类别在数据集对象中进行分类和检测。		没有错误或警告
测试与训练之间的重叠资产	测试数据集与训练数据集之间不应有重叠。	
数据集中的图像必须来自同一个存储桶	如果对象位于不同的存储桶中，则会出错	错误

将其他数据集格式转换为清单文件

您可以使用以下信息从各种源数据集 SageMaker 格式创建 Amazon 格式的清单文件。创建清单文件后，即可使用它来创建数据集。有关更多信息，请参见 清单文件。

主题

- 转换 COCO 数据集
- 转换多标签 G SageMaker round Truth 清单文件
- 从 CSV 文件创建清单文件

转换 COCO 数据集

COCO 是一种用于指定大规模物体检测、分段和字幕数据集的格式。此 Python 示例说明了如何将 COCO 物体检测格式的数据集转换为 Amazon Rekognition Custom Labels 边界框格式的清单文件。本部分还提供了可用于编写自己的代码的信息。

COCO 格式的 JSON 文件由五个部分组成，提供了整个数据集的信息。有关更多信息，请参见 COCO 格式。

- `info`：有关数据集的一般信息。
- `licenses` ：数据集中图像的许可信息。
- `images`：数据集中图像的列表。
- `annotations`：数据集中所有图像中存在的注释（包括边界框）的列表。
- `categories`：标签类别列表。

您需要来自 `images`、`annotations` 和 `categories` 列表的信息，才能创建 Amazon Rekognition Custom Labels 清单文件。

Amazon Rekognition Custom Labels 清单文件采用 JSON 行格式，其中每行都包含了一张图像上的一个或多个物体的边界框和标签信息。有关更多信息，请参见 清单文件中的物体定位。

将 COCO 对象映射到自定义标签 JSON 行

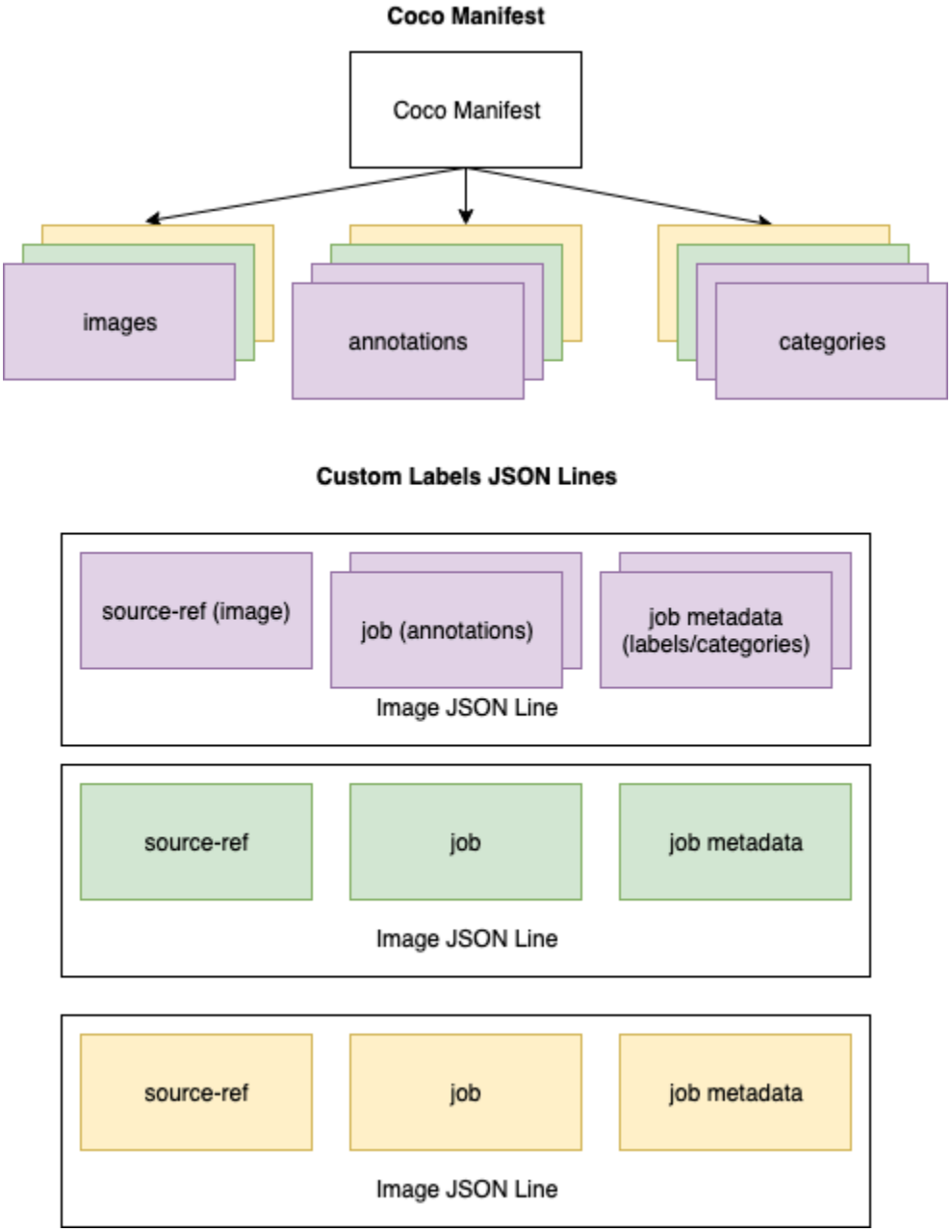
要转换 COCO 格式的数据集，需要将 COCO 数据集映射到 Amazon Rekognition Custom Labels 清单文件以进行物体定位。有关更多信息，请参见 清单文件中的物体定位。要为每张图像构建一个 JSON 行，清单文件需要映射 COCO 数据集的 `image`、`annotation` 和 `category` 对象字段 ID。

下面是一个 COCO 清单文件示例。有关更多信息，请参见 COCO 格式。

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

```
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
       {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
       {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

下图显示了数据集的 COCO 数据集列表如何映射到图像的 Amazon Rekognition Custom Labels JSON 行。匹配的颜色表示单张图像的信息。



获取单个 JSON 行的 COCO 对象

1. 对于图像列表中的每张图像，从注释列表中获取注释，其中注释字段 `image_id` 的值与图像 `id` 字段匹配。
2. 对于步骤 1 中匹配的每个注释，请通读 `categories` 列表并获取 `category` 字段 `id` 的值与 `annotation` 对象 `category_id` 字段匹配的每个 `category`。
3. 使用匹配的 `image`、`annotation` 和 `category` 对象为图像创建 JSON 行。要映射字段，请参阅将 COCO 对象字段映射到自定义标签 JSON 行对象字段。

4. 重复步骤 1-3，直到为 image 列表中的每个 images 对象创建 JSON 行。

有关代码示例，请参阅 转换 COCO 数据集。

将 COCO 对象字段映射到自定义标签 JSON 行对象字段

确定 Amazon Rekognition Custom Labels JSON 行的 COCO 对象后，需要将 COCO 对象字段映射到相应的 Amazon Rekognition Custom Labels JSON 行对象字段。以下示例 Amazon Rekognition Custom Labels JSON 行将一张图像 (id=000000245915) 映射到上面的 COCO JSON 示例。请注意以下信息。

- source-ref 是图像在 Amazon S3 存储桶中的位置。如果 COCO 图像不是存储在 Amazon S3 存储桶中，则需要将它们移到 Amazon S3 存储桶中。
- annotations 列表中包含了图像上每个物体的 annotation 对象。annotation 对象包含边界框信息（top、left、width、height）和标签标识符 (class_id)。
- 标签标识符 (class_id) 映射到元数据中的 class-map 列表。该列表会列出图像上使用的标签。

```
{
 "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"
 }
}
```

按照以下信息将 Amazon Rekognition Custom Labels 清单文件字段映射到 COCO 数据集 JSON 字段。

source-ref

图像位置的 S3 格式 URL。该图像必须存储在 S3 存储桶中。有关更多信息，请参见 source-ref。如果 `coco_url` COCO 字段指向 S3 存储桶位置，则可以使用 `coco_url` 的值作为 `source-ref` 的值。或者，也可以将 `source-ref` 映射到 `file_name` (COCO) 字段，然后在转换代码中将所需的 S3 路径添加到图像的存储位置。

bounding-box

您选择的标签属性名称。有关更多信息，请参见 *bounding-box*。

image_size

图像大小（以像素为单位）。映射到 images 列表中的 `image` 对象。

- `height->` `image`.height
- `width->` `image`.width
- `depth`：Amazon Rekognition Custom Labels 未使用该属性，但必须为其提供一个值。

annotations

`annotation` 对象的列表。图像上的每个物体都有一个 `annotation`。

annotation

包含图像上物体的一个实例的边界框信息。

- `class_id`：映射到自定义标签的 `class-map` 列表的数字 ID。
- `top` -> `bbox[1]`
- `left` -> `bbox[0]`
- `width` -> `bbox[2]`
- `height` -> `bbox[3]`

***bounding-box*-metadata**

标签属性的元数据。包含标签和标签标识符。有关更多信息，请参见 *bounding-box*-metadata。

Objects

图像中的物体数组。按索引映射到 `annotations` 列表。

Object

- Amazon Rekognition Custom Labels 未使用该属性，但必须为其指定值 (1)。

class-map

适用于图像中检测到的物体的标签（类别）映射。映射到 categories 列表中的 category 对象。

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

类型

必须是 `groundtruth/object-detection`

human-annotated

指定 `yes` 或 `no`。有关更多信息，请参见 *bounding-box*-metadata。

creation-date -> image.date_captured

图像的创建日期和时间。映射到 COCO 图像列表中的图像的 image.date_captured 字段。Amazon Rekognition Custom Labels 期望的 `creation-date` 格式为 Y-M-DTH:M:S。

job-name

您选择的作业名称。

COCO 格式

COCO 数据集由五个信息部分组成，提供了整个数据集的信息。COCO 物体检测数据集的格式记录在 COCO 数据格式中。

- info：有关数据集的一般信息。
- licenses：数据集中图像的许可信息。
- images：数据集中图像的列表。
- annotations：数据集中所有图像中存在的注释（包括边界框）的列表。
- categories：标签类别列表。

要创建自定义标签清单，需要使用 COCO 清单文件中的 `images`、`annotations` 和 `categories` 列表。其他两个部分（`info`、`licences`）不是必需的。下面是一个 COCO 清单文件示例。

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

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

图像列表

COCO 数据集引用的图像列在图像数组中。每个图像对象都包含了有关该图像的信息，例如图像文件名。在以下示例图像对象中，请注意以下信息以及创建 Amazon Rekognition Custom Labels 清单文件需要哪些字段。

- id：（必需）图像的唯一标识符。id 字段映射到注释数组（存储边界框信息的地方）中的 id 字段。
- license：（非必需）映射到许可证数组。
- coco_url：（可选）图像的位置。
- flickr_url：（非必需）图像在 Flickr 上的位置。
- width：（必需）图像的宽度。
- height：（必需）图像的高度。
- file_name：（必需）图像文件名。在本示例中，file_name 与 id 匹配，但 COCO 数据集对此不作要求。
- date_captured：（必需）图像拍摄的日期和时间。

```
{
    "id": 245915,
    "license": 4,
```

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

注释（边界框）列表

所有图像上的所有物体的边界框信息都存储在注释列表中。单个注释对象包含单个物体的边界框信息以及该物体在图像上的标签。图像上物体的每个实例都有一个注释对象。

在以下示例中，请注意以下信息以及创建 Amazon Rekognition Custom Labels 清单文件需要哪些字段。

- id：（非必需）注释的标识符。
- image_id：（必需）对应于图像数组中的图像 id。
- category_id：（必需）用于标识边界框内物体的标签的标识符。它映射到类别数组的 id 字段。
- iscrowd：（非必需）指定图像中是否包含物体群。
- segmentation：（非必需）图像中物体的分段信息。Amazon Rekognition Custom Labels 不支持分段。
- area：（非必需）注释的区域。
- bbox：（必需）包含图像中物体的边界框的坐标（以像素为单位）。

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

类别列表

标签信息存储在类别数组中。在以下示例类别对象中，请注意以下信息以及创建 Amazon Rekognition Custom Labels 清单文件需要哪些字段。

- `supercategory`：（非必需）标签的父类别。
- `id`：（必需）标签标识符。`id` 字段映射到 `annotation` 对象中的 `category_id` 字段。在以下示例中，echo dot 的标识符为 2。
- `name`：（必需）标签名称。

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

转换 COCO 数据集

使用以下 Python 示例将边界框信息从 COCO 格式的数据集转换为 Amazon Rekognition Custom Labels 清单文件。该代码会将创建的清单文件上载到您的 Amazon S3 存储桶。该代码还提供了一个 AWS CLI 命令，您可以使用该命令上传您的图像。

转换 COCO 数据集 (SDK)

1. 如果您尚未执行以下操作，请：
   a. 确保您具有 `AmazonS3FullAccess` 权限。有关更多信息，请参见 设置 SDK 权限。
   b. 安装并配置 AWS CLI 和 AWS SDK。有关更多信息，请参见 步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下 Python 代码转换 COCO 数据集。设置以下值。
   - `s3_bucket`：要在其中存储图像和 Amazon Rekognition Custom Labels 清单文件的 S3 存储桶的名称。
   - `s3_key_path_images`：S3 存储桶 (`s3_bucket`) 中将要放置图像的位置的路径。
   - `s3_key_path_manifest_file`：S3 存储桶 (`s3_bucket`) 中将要放置自定义标签清单文件的位置的路径。
   - `local_path`：示例打开输入 COCO 数据集并保存新的自定义标签清单文件的位置的本地路径。
   - `local_images_path`：要用于训练的图像的本地路径。
   - `coco_manifest`：输入 COCO 数据集的文件名。

- cl_manifest_file：该示例创建的清单文件的名称。该文件保存在 local_path 指定的位置。按照惯例，该文件具有扩展名 .manifest，但这不是必需要求的。
- job_name：自定义标签作业的名称。

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

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

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

label_attribute ='bounding-box'

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

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

        #Get image info. Annotations are dealt with seperately
        sizes=[]
        image_size={}
        image_size["width"] = img["width"]
        image_size["depth"] = 3
```

```
        image_size["height"] = img["height"]
        sizes.append(image_size)

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

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

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

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


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

with open(coco_json_file) as f:

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

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


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

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

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

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}

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

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

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


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


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

print('Done parsing annotations')

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

for im in images_dict.values():

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

# Upload manifest file to S3 bucket.
print ('Uploading Custom Labels manifest file to S3 bucket')
```

```
print('Uploading'  + local_path + cl_manifest_file + ' to ' +
 s3_key_path_manifest_file)
print(s3_bucket)
s3 = boto3.resource('s3')
s3.Bucket(s3_bucket).upload_file(local_path + cl_manifest_file,
 s3_key_path_manifest_file + cl_manifest_file)

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

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

3. 运行该代码。
4. 在程序输出中，记下 `s3 sync` 命令。您在下一个步骤中需要用到它。
5. 在命令提示符处，运行 `s3 sync` 命令。您的图像将上传到 S3 存储桶。如果该命令在上传过程中失败，请再次运行它，直到您的本地图像与 S3 存储桶同步为止。
6. 在程序输出中，记下清单文件的 S3 URL 路径。您在下一个步骤中需要用到它。
7. 按照使用 G SageMaker round Truth 清单文件创建数据集（控制台）中的说明，使用上传的清单文件创建数据集。对于步骤 8，在 .manifest 文件位置中，输入您在上一步中记下的 Amazon S3 URL。如果使用的是 AWS SDK，请执行使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。

转换多标签 G SageMaker round Truth 清单文件

本主题向您展示如何将多标签 Amazon G SageMaker round Truth 清单文件转换为亚马逊 Rekognition 自定义标签格式的清单文件。

SageMaker 多标签任务的 Ground Truth 清单文件的格式与 Amazon Rekognition 自定义标签格式清单文件的格式不同。多标签分类是指将一个图像分类为一组类别，但可能同时属于多个类别。在这种情况下，该图像可能会有多个标签（多标签），例如 football 和 ball。

有关多标签 G SageMaker round Truth 作业的信息，请参阅图像分类（多标签）。有关多标签格式的 Amazon Rekognition Custom Labels 清单文件的信息，请参阅the section called “为图像添加多个图像级标签”。

获取 G SageMaker round Truth 任务的清单文件

以下过程向您展示了如何获取 Amazon G SageMaker round Truth 任务的输出清单文件 (`output.manifest`)。您可以将 `output.manifest` 用作下一过程的输入。

下载 G SageMaker round Truth 任务清单文件

1. 打开 https://console.aws.amazon.com/sagemaker/。
2. 在导航窗格中，选择 Ground Truth，然后选择标注作业。
3. 选择包含要使用的清单文件的标注作业。
4. 在详细信息页面上，选择输出数据集位置下的链接。Amazon S3 控制台将在数据集所在位置打开。
5. 依次选择 `Manifests`、`output` 和 `output.manifest`。
6. 选择对象操作，然后选择下载，下载清单文件。

转换多标签 SageMaker 清单文件

以下过程根据现有的多标签格式清单文件创建多标签格式的 Amazon Rekognition 自定义标签清单文件。 SageMaker GroundTruth

> **Note**
>
> 要运行此代码，您需要使用 Python 版本 3 或更高版本。

转换多标签 SageMaker 清单文件

1. 运行以下 Python 代码。提供您在获取 G SageMaker round Truth 任务的清单文件中创建的清单文件名称作为命令行参数。

```
# 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. 记下脚本显示的新清单文件的名称。您将在下一个步骤中使用它。
3. 上传清单文件至要用于存储清单文件的 Amazon S3 存储桶。

> **Note**
>
> 确保 Amazon Rekognition Custom Labels 可以访问清单文件 JSON 行的 `source-ref` 字段中引用的 Amazon S3 存储桶。有关更多信息，请参见 访问外部 Amazon S3 存储桶。如果 Ground Truth 作业将图像存储在 Amazon Rekognition Custom Labels 控制台存储桶中，则无需添加权限。

4. 按照使用 G SageMaker round Truth 清单文件创建数据集（控制台）中的说明，使用上传的清单文件创建数据集。对于步骤 8，在 .manifest 文件位置中，请输入清单文件位置的 Amazon S3 URL。如果使用的是 AWS SDK，请执行使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。

从 CSV 文件创建清单文件

此示例 Python 脚本使用逗号分隔值 (CSV) 文件来标注图像，从而简化了清单文件的创建工作。您需要创建 CSV 文件。该清单文件适用于多标签图像分类或多标签图像分类。有关更多信息，请参见 查找物体、场景和概念。

> **Note**
>
> 此脚本无法创建适用于查找物体位置或品牌位置的清单文件。

清单文件描述了用于训练模型的图像。例如，图像位置和分配给图像的标签。清单文件由一个或多个 JSON 行组成。每个 JSON 行都描述了一张图像。有关更多信息，请参见 the section called “清单文件中的图像级标签”。

CSV 文件代表文本文件中多行的表格数据。一行中的各个字段用逗号分隔。有关更多信息，请参阅逗号分隔的值。对于此脚本，CSV 文件中的每一行都代表一张图像，并映射到清单文件中的一个 JSON 行。要为支持多标签图像分类的清单文件创建 CSV 文件，您需要向每行添加一个或多个图像级标签。要创建适用于图像分类的清单文件，请在每行中添加一个图像级标签。

例如，以下 CSV 文件描述了多标签图像分类 (Flowers) 入门项目中的图像。

```
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
```

该脚本会为每一行生成 JSON 行。例如，以下是第一行 (`camellia1.jpg,camellia,with_leaves`) 的 JSON 行。

```
{"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"}}
```

在示例 CSV 中，没有图像的 Amazon S3 路径。如果您的 CSV 文件不包含图像的 Amazon S3 路径，请使用 `--s3_path` 命令行参数指定图像的 Amazon S3 路径。

该脚本会在去重图像 CSV 文件中记录每张图像的第一个条目。去重图像 CSV 文件包含在输入 CSV 文件中找到的每张图像的单个实例。输入 CSV 文件中存在的图像的其他实例将记录在重复图像 CSV 文件中。如果脚本发现重复的图像，便会检查重复图像 CSV 文件并根据需要更新去重图像 CSV 文件。使用去重文件，重新运行该脚本。如果在输入 CSV 文件中未发现重复项，则脚本会删除去重图像 CSV 文件和重复图像 CSV 文件，因为它们是空的。

在此过程中，您将创建 CSV 文件并运行 Python 脚本以创建清单文件。

通过 CSV 文件创建清单文件

1. 创建一个 CSV 文件，并且在每一行中包含以下字段（每张图像占一行）。请勿在 CSV 文件中添加标题行。

字段 1	字段 2	字段 n
图像名称或图像的 Amazon S3 路径。例如，`s3://my-bucket/flowers/train/camellia1.jpg` 。您不能混合使用带有 Amazon S3 路径的图像和不带 Amazon S3 路径的图像。	图像的第一个图像级标签。	一个或多个其他图像级标签（以逗号分隔）。仅当您想要创建支持多标签图像分类的清单文件时才添加。

   例如，`camellia1.jpg,camellia,with_leaves` 或 `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`

2. 保存 CSV 文件。
3. 运行以下 Python 脚本。提供以下参数：

   - `csv_file`：您在步骤 1 中创建的 CSV 文件。
   - `manifest_file`：您要创建的清单文件的名称。
   - （可选）`--s3_path` *`s3://path_to_folder/`*：要添加到图像文件名（字段 1）的 Amazon S3 路径。如果字段 1 中的图像不包含 S3 路径，则使用 `--s3_path`。

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

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

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

logger = logging.getLogger(__name__)


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

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

    duplicates_found = False

    # Find duplicates.
```

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

        reader = csv.reader(f, delimiter=',')
        dedup_writer = csv.writer(dedup)
        duplicates_writer = csv.writer(duplicates)

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

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

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

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

    return duplicates_found


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

    image_count = 0
    label_count = 0
```

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

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

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

            image_count += 1

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

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

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

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

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

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

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

```
                manifest_file, image_count, label_count)

    return image_count, label_count


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

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

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


def main():

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

    try:

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

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

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

```
        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
                  f"and then update {deduplicated_file}. ")
            print(f"{deduplicated_file} contains the first occurence of a
 duplicate. "
                  "Update as necessary with the correct label information.")
            print(f"Re-run the script with {deduplicated_file}")
        else:
            print("No duplicates found. Creating manifest file.")

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

            print(f"Finished creating manifest file: {manifest_file} \n"
                  f"Images: {image_count}\nLabels: {label_count}")

    except FileNotFoundError as err:
        logger.exception("File not found: %s", err)
        print(f"File not found: {err}. Check your input CSV file.")


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

4. 如果您计划使用测试数据集，请重复步骤 1-3，以便为测试数据集创建清单文件。
5. 如有必要，请将图像复制到您在 CSV 文件第 1 列中指定的（或在 `--s3_path` 命令行中指定的）Amazon S3 存储桶路径。您可使用以下 AWS S3 命令。

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

6. 上传清单文件至要用于存储清单文件的 Amazon S3 存储桶。

> **Note**
>
> 确保 Amazon Rekognition Custom Labels 可以访问清单文件 JSON 行的 `source-ref` 字段中引用的 Amazon S3 存储桶。有关更多信息，请参见 访问外部 Amazon S3 存储桶。

如果 Ground Truth 作业将图像存储在 Amazon Rekognition Custom Labels 控制台存储桶中，则无需添加权限。

7. 按照使用 G SageMaker round Truth 清单文件创建数据集（控制台）中的说明，使用上传的清单文件创建数据集。对于步骤 8，在 .manifest 文件位置中，请输入清单文件位置的 Amazon S3 URL。如果使用的是 AWS SDK，请执行使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。

现有数据集

如果您之前已创建数据集，可以将其内容复制到新数据集中。要使用 AWS SDK 从现有数据集创建数据集，请参阅使用现有数据集创建数据集 (SDK)。

使用现有的 Amazon Rekognition Custom Labels 数据集创建数据集（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要向其添加数据集的项目。此时将显示项目的详细信息页面。
6. 选择创建数据集。此时将显示创建数据集页面。
7. 在开始配置中，选择从单个数据集开始或从训练数据集开始。要创建更高质量的模型，建议从单独的训练和测试数据集开始。

   Single dataset

   a. 在训练数据集详细信息部分，选择复制现有的 Amazon Rekognition Custom Labels 数据集。
   b. 在训练数据集详细信息部分，于数据集编辑框中，输入或选择要复制的数据集的名称。
   c. 选择创建数据集。这时会打开项目的数据集页面。

   Separate training and test datasets

   a. 在训练数据集详细信息部分，选择复制现有的 Amazon Rekognition Custom Labels 数据集。
   b. 在训练数据集详细信息部分，于数据集编辑框中，输入或选择要复制的数据集的名称。

c. 在测试数据集详细信息部分，选择复制现有的 Amazon Rekognition Custom Labels 数据集。

d. 在测试数据集详细信息部分，于数据集编辑框中，输入或选择要复制的数据集的名称。

> **Note**
>
> 训练数据集和测试数据集可以有不同的图像源。

e. 选择创建数据集。这时会打开项目的数据集页面。

8. 如果需要添加或更改标签，请执行标注图像中的操作。
9. 按照训练模型（控制台）中的步骤训练您的模型。

标注图像

标签用于标识图像中的物体、场景、概念或物体周围的边界框。例如，如果数据集包含狗的图像，则可以添加表示犬种的标签。

将图像导入数据集后，您可能需要为图像添加标签或更正标注错误的图像。例如，从本地计算机导入的图像就没有标签。可以使用数据集库向数据集添加新标签，并为数据集中的图像分配标签和边界框。

如何为数据集中的图像添加标签决定了 Amazon Rekognition Custom Labels 训练的模型的类型。有关更多信息，请参见 确定数据集用途。

主题

- 管理标签
- 为图像分配图像级标签
- 使用边界框标注物体

管理标签

可以使用 Amazon Rekognition Custom Labels 控制台管理标签。没有用于管理标签的特定 API - 当您使用 `CreateDataset` 创建数据集或使用 `UpdateDatasetEntries` 向数据集添加更多图像时，便会将标签添加到数据集中。

主题

- 管理标签（控制台）

- 管理标签 (SDK)

管理标签（控制台）

可以使用 Amazon Rekognition Custom Labels 控制台从数据集中添加、更改或移除标签。要向数据集添加标签，您可以在 Rekognition 中添加自己创建的新标签或从现有数据集导入标签。

主题

- 添加新标签（控制台）
- 更改和移除标签（控制台）

添加新标签（控制台）

您可以指定要添加到数据集的新标签。

使用编辑窗口添加标签

添加新标签（控制台）

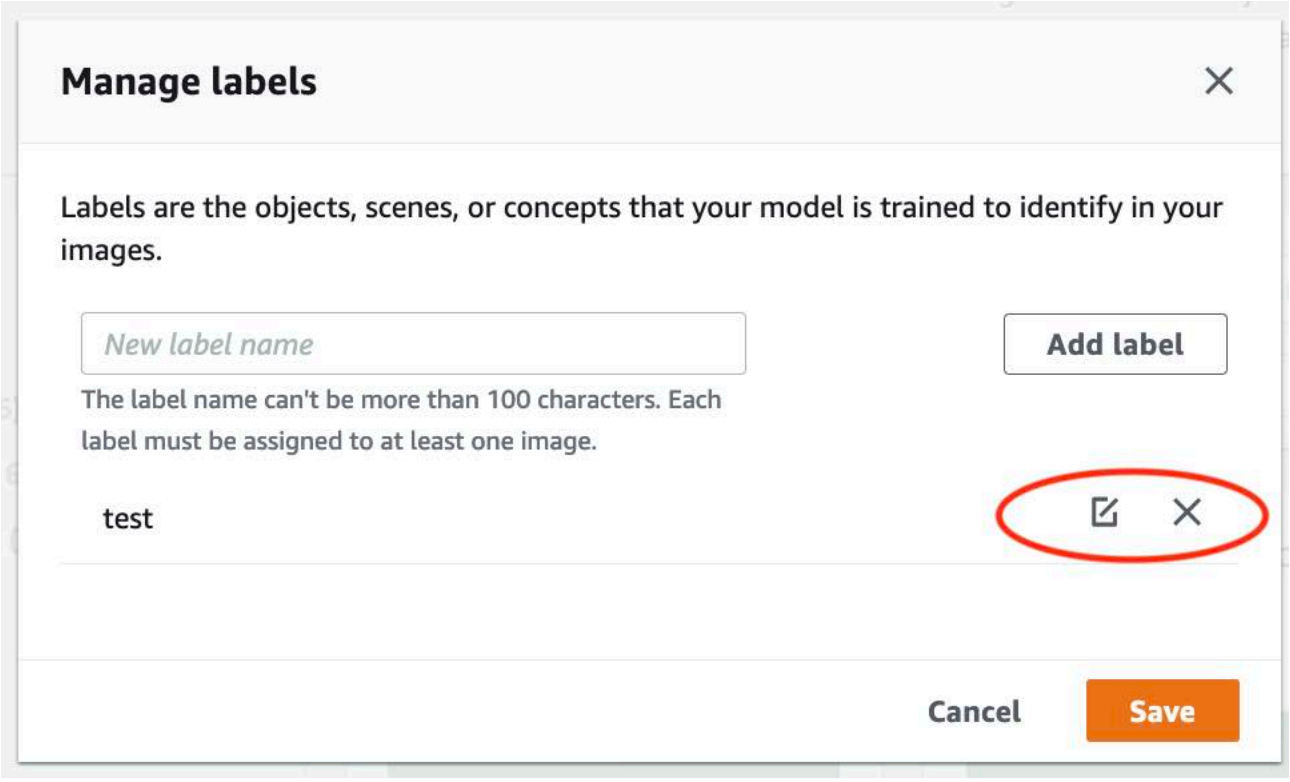
1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要使用的项目。此时将显示项目的详细信息页面。
6. 如果要向训练数据集添加标签，请选择训练选项卡。否则，请选择测试选项卡，向测试数据集添加标签。
7. 选择开始标注，进入标注模式。
8. 在数据集库的标签部分，选择管理标签，打开管理标签对话框。
9. 在编辑框中，输入新标签名称。
10. 选择添加标签。
11. 重复步骤 9 和 10，直到创建完所需的所有标签。
12. 选择保存，保存您添加的标签。

更改和移除标签（控制台）

将标签添加到数据集后，可以对其进行重命名或移除。只能移除尚未分配给任何图像的标签。

重命名或移除现有标签（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要使用的项目。此时将显示项目的详细信息页面。
6. 如果要更改或删除训练数据集中的标签，请选择训练选项卡。否则，请选择测试选项卡，以更改或删除测试数据集中的标签。
7. 选择开始标注，进入标注模式。
8. 在数据集库的标签部分，选择管理标签，打开管理标签对话框。
9. 选择要编辑或删除的标签。



   a. 如果选择删除图标 (X)，则会从列表中移除标签。
   b. 如果要更改标签，请选择编辑图标（铅笔和便签组合成的图标），然后在编辑框中输入新标签名称。
10. 选择保存，保存您的更改。

管理标签 (SDK)

没有专门的 API 来管理数据集标签。如果使用 CreateDataset、在清单文件中找到的标签或复制的数据集来创建数据集，则会创建初始标签集。如果使用 UpdateDatasetEntries API 添加更多图像，则会将在这些条目中发现的新标签添加到数据集中。有关更多信息，请参见 添加更多图像 (SDK)。要从数据集中删除标签，必须移除数据集中的所有标签注释。

删除数据集中的标签

1. 调用 ListDatasetEntries 获取数据集条目。有关代码示例，请参阅 列出数据集条目 (SDK)。
2. 在文件中，移除所有标签注释。有关更多信息，请参阅 清单文件中的图像级标签和 the section called “清单文件中的物体定位”：
3. 使用该文件通过 UpdateDatasetEntries API 更新数据集。有关更多信息，请参见 添加更多图像 (SDK)。

为图像分配图像级标签

可以使用图像级标签来训练将图像分类的模型。图像级标签表示图像包含物体、场景或概念。例如，下图显示了一条河。如果模型将图像分类为包含河流，则需要添加 river 图像级标签。有关更多信息，请参见 确定数据集用途。

包含图像级标签的数据集需要至少定义两个标签。每张图像都至少需要分配一个用于标识图像中的物体、场景或概念的标签。

为图像分配图像级标签（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要使用的项目。此时将显示项目的详细信息页面。
6. 在左侧导航窗格中，选择数据集。
7. 如果要向训练数据集添加标签，请选择训练选项卡。否则，请选择测试选项卡，向测试数据集添加标签。
8. 选择开始标注，进入标注模式。

9. 在图像库中，选择要为其添加标签的一张或多张图像。一次只能选择一个页面上的图像。要在一个页面上选择连续范围的图像，请执行以下操作：
   a. 选择范围中的第一张图像。
   b. 按住 Shift 键。
   c. 选择范围中的最后一张图像。这样便可将第一张和第二张图像之间的图像全部选中。
   d. 松开 Shift 键。
10. 选择分配图像级标签。
11. 在向选定图像分配图像级标签对话框中，选择要分配给该等图像的标签。
12. 选择分配，将标签分配给图像。
13. 重复标注，直到每张图像都用所需的标签进行注释。
14. 选择保存更改以保存您的更改。

分配图像级标签 (SDK)

可以使用 `UpdateDatasetEntries` API 添加或更新分配给图像的图像级标签。`UpdateDatasetEntries` 接受一个或多个 JSON 行。每个 JSON 行代表一张图像。对于带有图像级标签的图像，JSON 行类似如下。

```
{"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-ref` 字段表示图像的位置。JSON 行还包含分配给图像的图像级标签。有关更多信息，请参见 the section called “清单文件中的图像级标签”。

为图像分配图像级标签

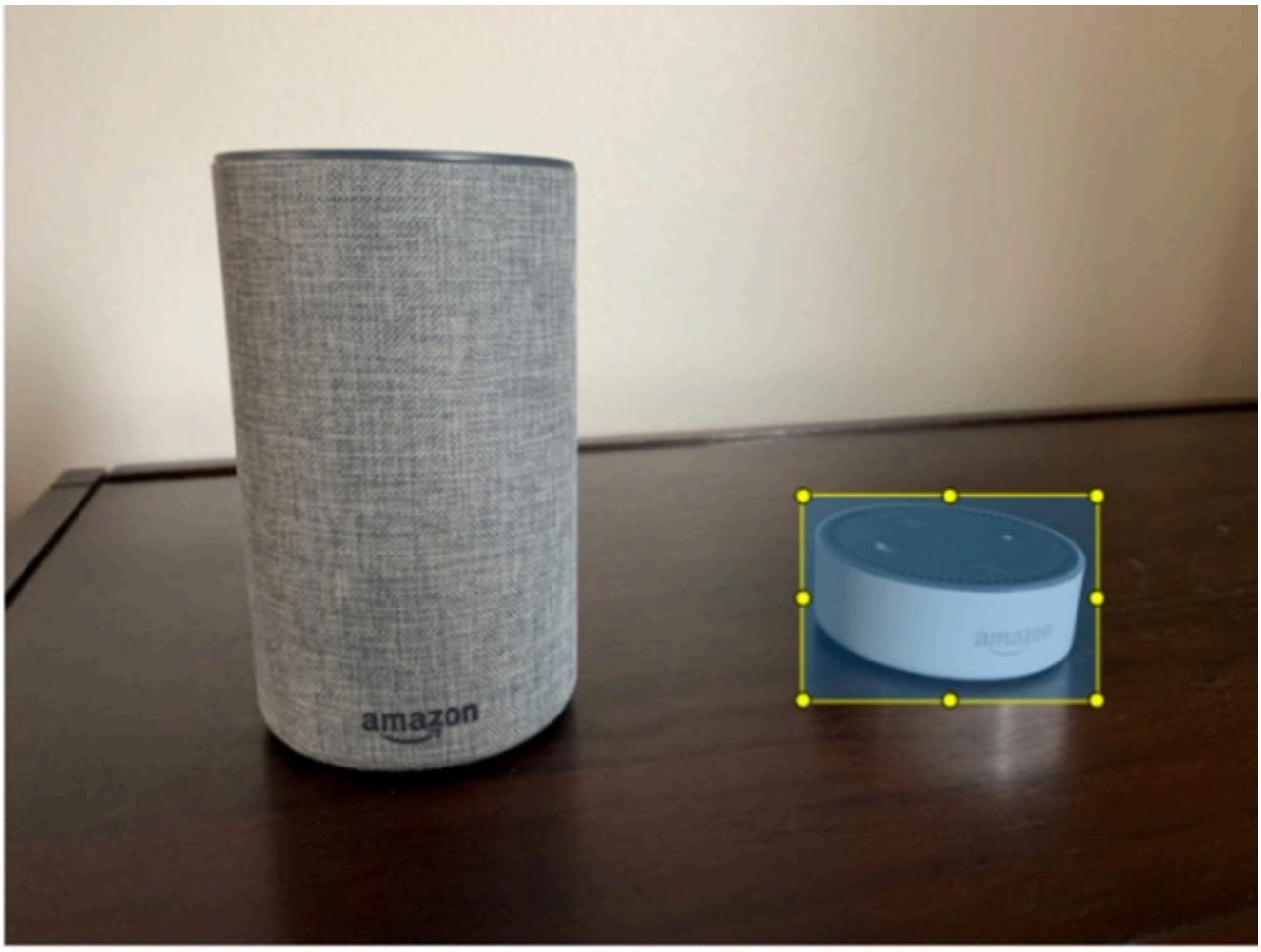
1. 使用 `ListDatasetEntries` 获取现有图像的 JSON 行。在 `source-ref` 字段中，指定要为其分配标签的图像的位置。有关更多信息，请参见 列出数据集条目 (SDK)。
2. 按照清单文件中的图像级标签中的信息更新上一步中返回的 JSON 行。
3. 调用 `UpdateDatasetEntries` 来更新图像。有关更多信息，请参见 向数据集中添加更多图像。

使用边界框标注物体

如果要让模型检测图像中物体的位置，则必须标识物体是什么以及它在图像中的位置。边界框是用来隔离图像中物体的方框。可以使用边界框来训练模型以检测同一图像中的不同物体。可以通过为边界框分配标签来标识物体。

> **Note**
>
> 如果要训练模型查找带有图像级标签的物体、场景和概念，则无需执行此步骤。

例如，如果要训练检测 Amazon Echo Dot 设备的模型，则可以在图像中的每个 Echo Dot 周围绘制一个边界框，再为边界框分配名为 Echo Dot 的标签。下图显示了 Echo Dot 设备周围的边界框。该图像还包含了一个没有边界框的 Amazon Echo。



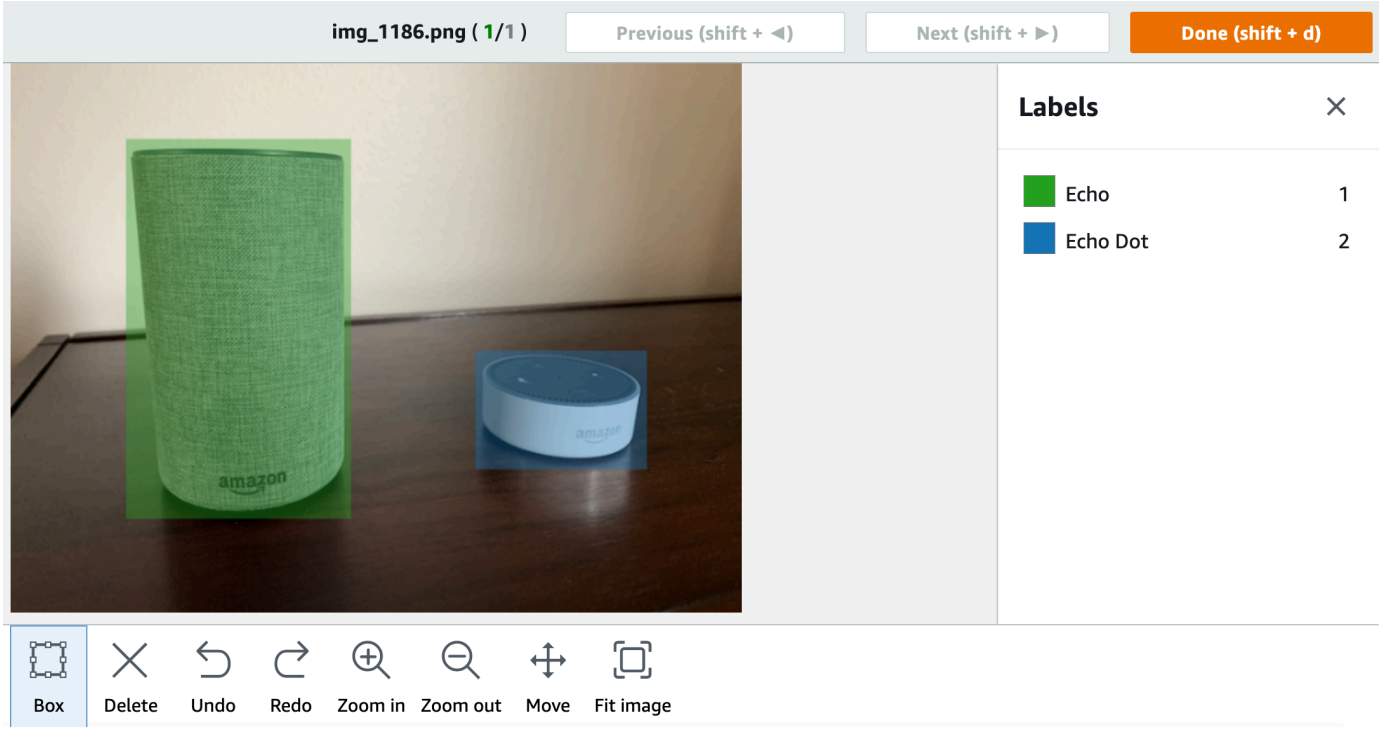
使用边界框定位物体（控制台）

在此过程中，您将使用控制台在图像中物体的周围绘制边界框。您还可以通过为边界框分配标签来标识图像中的物体。

> **Note**
>
> 不能使用 Safari 浏览器为图像添加边界框。有关支持的浏览器，请参阅设置 Amazon Rekognition Custom Labels。

在添加边界框之前，必须至少向数据集添加一个标签。有关更多信息，请参见 添加新标签（控制台）。

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要使用的项目。此时将显示项目的详细信息页面。
6. 在项目详细信息页面上，选择对图像进行贴标
7. 如果要向训练数据集图像添加边界框，请选择训练选项卡。否则，请选择测试选项卡，向测试数据集图像添加边界框。
8. 选择开始标注，进入标注模式。
9. 在图像库中，选择要向其添加边界框的图像。
10. 选择绘制边界框。在显示边界框编辑器之前，会显示一系列提示。
11. 在右侧的标签窗格中，选择要分配给边界框的标签。
12. 在绘图工具中，将指针放在所需物体的左上角区域。
13. 按住鼠标左键，在物体周围绘制一个方框。尝试将边界框绘制得尽量靠近物体。
14. 松开鼠标键。该边界框会突出显示。
15. 如果有更多图像需要标注，请选择下一步。否则，请选择完成，完成标注。



16. 重复步骤 1—7，直到在每张包含物体的图像中都创建了一个边界框。
17. 选择保存更改以保存您的更改。
18. 选择退出，退出标注模式。

使用边界框定位物体 (SDK)

可以使用 UpdateDatasetEntries API 来添加或更新图像的物体位置信息。UpdateDatasetEntries 接受一个或多个 JSON 行。每个 JSON 行代表一张图像。对于物体定位，JSON 行类似如下。

```
{"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 字段表示图像的位置。JSON 行还包含了图像上每个物体的带标签的边界框。有关更多信息，请参见 the section called “清单文件中的物体定位”。

为图像分配边界框

1. 使用 ListDatasetEntries 获取现有图像的 JSON 行。在 source-ref 字段中，指定要为其分配图像级标签的图像的位置。有关更多信息，请参见 列出数据集条目 (SDK)。
2. 按照清单文件中的物体定位中的信息更新上一步中返回的 JSON 行。
3. 调用 UpdateDatasetEntries 来更新图像。有关更多信息，请参见 向数据集中添加更多图像。

调试数据集

在数据集创建过程中，可能会发生两种类型的错误：终止性错误和非终止性错误。终止性错误可能会停止数据集的创建或更新。非终止性错误不会停止数据集的创建或更新。

主题

- 终止性错误
- 非终止性错误

终止性错误

终止性错误有两种类型：导致数据集创建失败的文件错误，以及 Amazon Rekognition Custom Labels 从数据集中移除的内容错误。如果内容错误太多，数据集创建会失败。

主题

- 终止性文件错误
- 终止性内容错误

终止性文件错误

以下是文件错误。可以通过调用 DescribeDataset 并检查 Status 和 StatusMessage 字段来获取有关文件错误的信息。有关代码示例，请参阅 描述数据集 (SDK)。

- ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT
- ERROR_MANIFEST_SIZE_TOO_LARGE.
- ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM
- ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET
- ERROR_TOO_MANY_RECORDS_IN_ERROR

- ERROR_MANIFEST_TOO_MANY_LABELS
- ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

错误消息

清单文件扩展名或内容无效。

训练或测试清单文件没有文件扩展名或其内容无效。

修复错误 ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

- 在训练和测试清单文件中检查以下可能的原因。
  - 清单文件缺少文件扩展名。按照惯例，文件扩展名为 `.manifest`。
  - 找不到清单文件的 Amazon S3 存储桶或密钥。

ERROR_MANIFEST_SIZE_TOO_LARGE

错误消息

清单文件大小超过了支持的最大大小。

训练或测试清单文件大小（以字节为单位）太大。有关更多信息，请参见 Amazon Rekognition Custom Labels 中的准则和配额。清单文件可能会 JSON 行数少于最大值，但文件大小超过最大值。

无法使用 Amazon Rekognition Custom Labels 控制台修复错误：清单文件大小超过了支持的最大大小。

修复错误 ERROR_MANIFEST_SIZE_TOO_LARGE

1. 检查哪些训练和测试清单文件超过了最大文件大小。
2. 减少过大的清单文件中的 JSON 行数。有关更多信息，请参见 创建清单文件。

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

错误消息

清单文件中的行数太多。

更多信息

清单文件中的 JSON 行数（图像数）大于允许的限制。图像级模型和物体位置模型的限制不同。有关更多信息，请参见 Amazon Rekognition Custom Labels 中的准则和配额。

系统会验证 JSON 行错误，直至 JSON 行数达到 ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM 限制。

无法使用 Amazon Rekognition Custom Labels 控制台修复 ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM 错误。

修复 **ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM**

- 减少清单中的 JSON 行数。有关更多信息，请参见 创建清单文件。

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

错误消息

S3 存储桶权限不正确。

Amazon Rekognition Custom Labels 不具有对一个或多个包含训练和测试清单文件的存储桶的权限。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复错误 ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

- 检查对包含训练和测试清单的存储桶的权限。有关更多信息，请参见 步骤 2：设置 Amazon Rekognition Custom Labels 控制台权限。

ERROR_TOO_MANY_RECORDS_IN_ERROR

错误消息

清单文件有太多终止性错误。

修复 **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- 减少有终止性内容错误的 JSON 行（图像）的数量。有关更多信息，请参见 终止性清单内容错误。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_MANIFEST_TOO_MANY_LABELS

错误消息

清单文件包含的标签太多。

更多信息

清单（数据集）中唯一标签的数量超过了允许的限制。如果拆分训练数据集来创建测试数据集，则标签数量将在拆分后确定。

修复 ERROR_MANIFEST_TOO_MANY_LABELS（控制台）

- 从数据集中移除标签。有关更多信息，请参见 管理标签。标签会自动从数据集中的图像和边界框中移除。

修复 ERROR_MANIFEST_TOO_MANY_LABELS（JSON 行）

- 清单包含图像级 JSON 行 - 如果图像只有一个标签，请移除使用所需标签的图像对应的 JSON 行。如果相关 JSON 行包含多个标签，请仅移除所需标签对应的 JSON 对象。有关更多信息，请参见 为图像添加多个图像级标签。

  清单包含物体位置 JSON 行 - 移除要移除的标签所对应的边界框和关联的标签信息。对包含所需标签的每个 JSON 行执行此操作。需要从 `class-map` 数组中移除标签并从 `objects` 和 `annotations` 数组中移除相应的对象。有关更多信息，请参见 清单文件中的物体定位。

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

错误消息

清单文件中没有足够的带标签的图像来分配数据集。

当 Amazon Rekognition Custom Labels 拆分训练数据集来创建测试数据集时，就会发生数据集分配。也可以通过调用 `DistributeDatasetEntries` API 来拆分数据集。

修复错误 ERROR_MANIFEST_TOO_MANY_LABELS

- 向训练数据集中添加更多带标签的图像

终止性内容错误

以下是终止性内容错误。在创建数据集的过程中，会从数据集中移除包含终止性内容错误的图像。该数据集仍可用于训练。如果内容错误太多，数据集创建/更新会失败。与数据集操作相关的终止性内容错误不会显示在控制台中，也不会从 DescribeDataset 或其他 API 返回。如果您发现数据集中缺少图像或注释，请检查您的数据集清单文件是否存在以下问题：

- JSON 行的长度太长。最大长度为 100,000 个字符。
- JSON 行中缺少 source-ref 值。
- JSON 行中 source-ref 值的格式无效。
- JSON 行的内容无效。
- 一个 source-ref 字段值出现了多次。一张图像在一个数据集中只能被引用一次。

有关 source-ref 字段的信息，请参阅创建清单文件。

非终止性错误

以下是在数据集创建或更新过程中可能发生的非终止性错误。这些错误可能会使整个 JSON 行失效或使 JSON 行中的注释失效。如果 JSON 行有错误，则不会将其用于训练。如果 JSON 行中的注释有错误，仍会将该 JSON 行用于训练，但不包括损坏的注释。有关 JSON 行的更多信息，请参阅创建清单文件。

可以通过控制台以及通过调用 ListDatasetEntries API 来访问非终止性错误。有关更多信息，请参见 列出数据集条目 (SDK)。

训练期间还会返回以下错误。建议您在训练模型之前先修复这些错误。有关更多信息，请参阅非终止性 JSON 行验证错误。

- 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

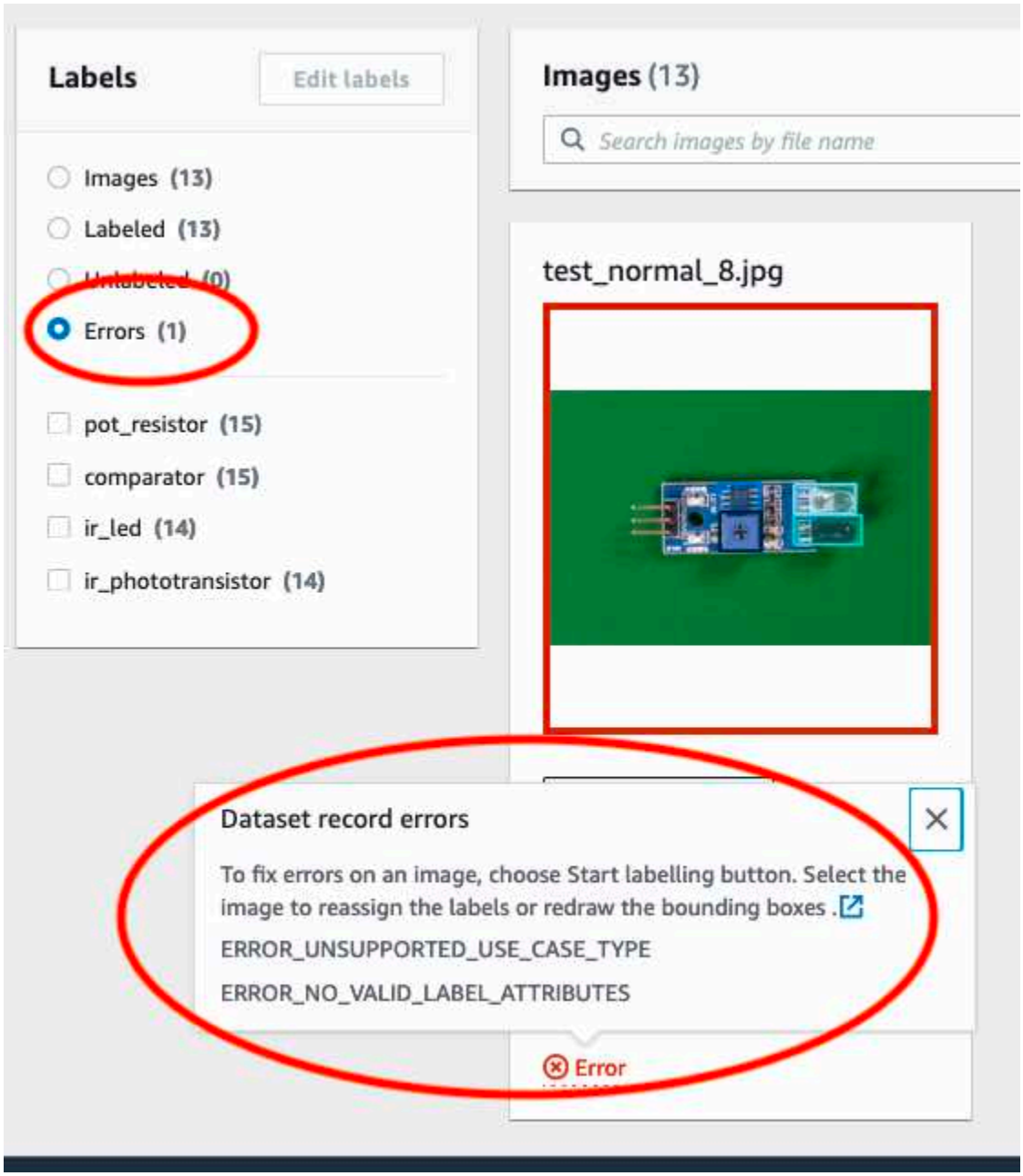
- ERROR_NO_VALID_ANNOTATIONS
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_TOO_MANY_BOUNDING_BOXES
- ERROR_UNSUPPORTED_USE_CASE_TYPE
- ERROR_INVALID_LABEL_NAME_LENGTH

访问非终止性错误

可以使用控制台来找出数据集中存在非终止性错误的图像。也可以调用 `ListDatasetEntries` API 来获取错误消息。有关更多信息，请参见 列出数据集条目 (SDK)。

访问非终止性错误（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择要使用的项目。此时将显示项目的详细信息页面。
6. 如果要查看训练数据集中的非终止性错误，请选择训练选项卡。否则，请选择测试选项卡，以查看测试数据集中的非终止性错误。
7. 在数据集库的标签部分，选择错误。数据集库经过筛选，仅显示有错误的图像。
8. 选择图像下方的错误以查看错误代码。按照非终止性 JSON 行验证错误中的说明修复错误。



训练 Amazon Rekognition Custom Labels 模型

可以使用 Amazon Rekognition Custom Labels 控制台或 Amazon Rekognition Custom Labels API 来训练模型。如果模型训练失败，请按照调试失败的模型训练中的说明查找失败的原因。

> **Note**
>
> 您需要按照成功训练模型所花费的时间付费。通常，训练需要 30 分钟到 24 小时才能完成。有关更多信息，请参阅训练时长。

每次训练模型都会创建一个新的模型版本。Amazon Rekognition Custom Labels 会为模型创建一个名称，该名称是项目名称和模型创建时的时间戳的组合。

为了训练您的模型，Amazon Rekognition Custom Labels 会复制您的源训练图像和测试图像。默认情况下，复制的图像使用 AWS 拥有和管理的密钥进行静态加密。您也可以选择使用自己的 AWS KMS key。如果使用自己的 KMS 密钥，则需要对该 KMS 密钥具有以下权限。

- kms:CreateGrant
- kms:DescribeKey

有关更多信息，请参阅 AWS Key Management Service 概念。源图像不受影响。

可以使用 KMS 服务器端加密 (SSE-KMS) 加密 Amazon S3 存储桶中的训练和测试图像，然后再将它们复制到 Amazon Rekognition Custom Labels 中。要允许 Amazon Rekognition Custom Labels 访问您的图像，您的 AWS 账户需要对 KMS 密钥拥有以下权限。

- kms:GenerateDataKey
- kms:Decrypt

有关更多信息，请参阅使用存储在 AWS Key Management Service 中的 KMS 密钥通过服务器端加密 (SSE-KMS) 保护数据。

训练模型后，您可以评估其性能并进行改进。有关更多信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。

有关其他模型任务（例如标记模型），请参阅管理 Amazon Rekognition Custom Labels 模型。

主题

- 训练模型（控制台）
- 训练模型 (SDK)

训练模型（控制台）

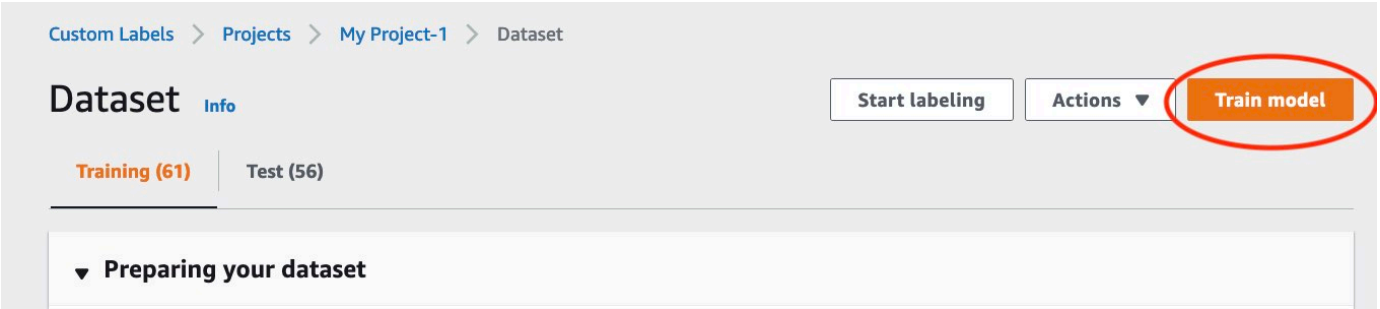
可以使用 Amazon Rekognition Custom Labels 控制台训练项目。

训练需要一个包含训练数据集和测试数据集的项目。如果项目没有测试数据集，Amazon Rekognition Custom Labels 控制台会在训练期间拆分训练数据集，为项目创建一个测试数据集。所选图像是具有代表性的采样，不会用于训练数据集。建议您仅在没有可供使用的替代测试数据集时才拆分训练数据集。拆分训练数据集会减少可用于训练的图像数量。

> **Note**
>
> 您需要按照训练模型所花费的时间付费。有关更多信息，请参阅训练时长。

训练模型（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 在左侧导航窗格中，选择项目。
4. 在项目页面上，选择包含要训练的模型的项目。
5. 在项目页面上，选择训练模型。



6. （可选）如果要使用自己的 AWS KMS 加密密钥，请执行以下操作：
   a. 在图像数据加密中，选择自定义加密设置(高级)。
   b. 在 encryption.aws_kms_key 中，输入您的密钥的 Amazon 资源名称 (ARN)，或者选择现有的 AWS KMS 密钥。要创建新密钥，请选择创建 AWS IMS 密钥。
7. （可选）如果要向模型添加标签，请执行以下操作：
   a. 在标签部分中，选择添加新标签。
   b. 输入以下信息：

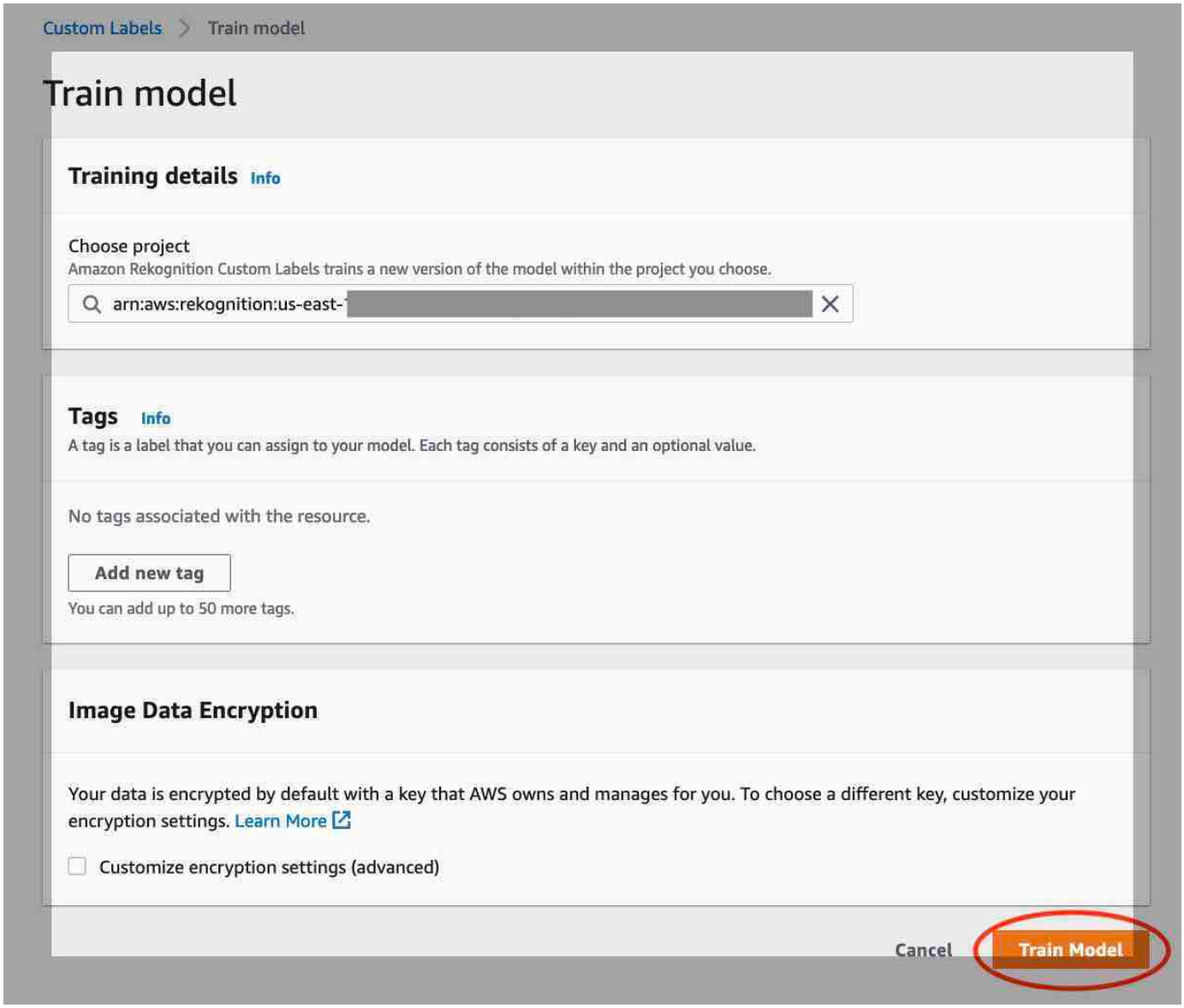
i. 在键中输入键名称。

ii. 在值中输入键值。

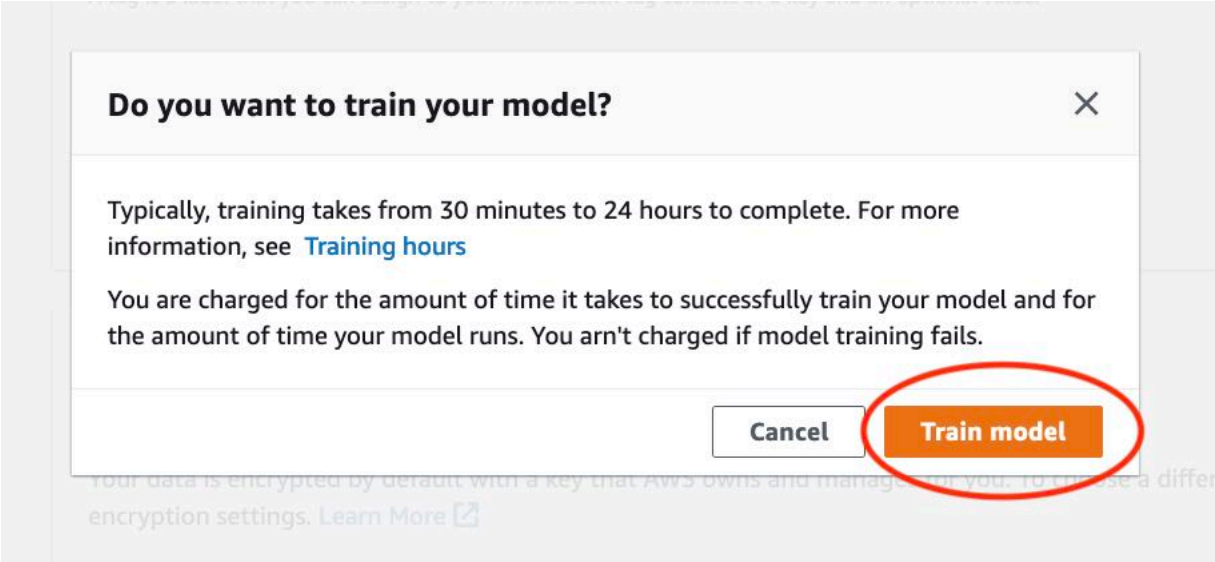
c. 要添加更多标签，请重复步骤 6a 和 6b。

d. （可选）如果要移除标签，请选择要移除的标签旁的移除。如果移除的是先前保存的标签，则会在保存更改时将其移除。

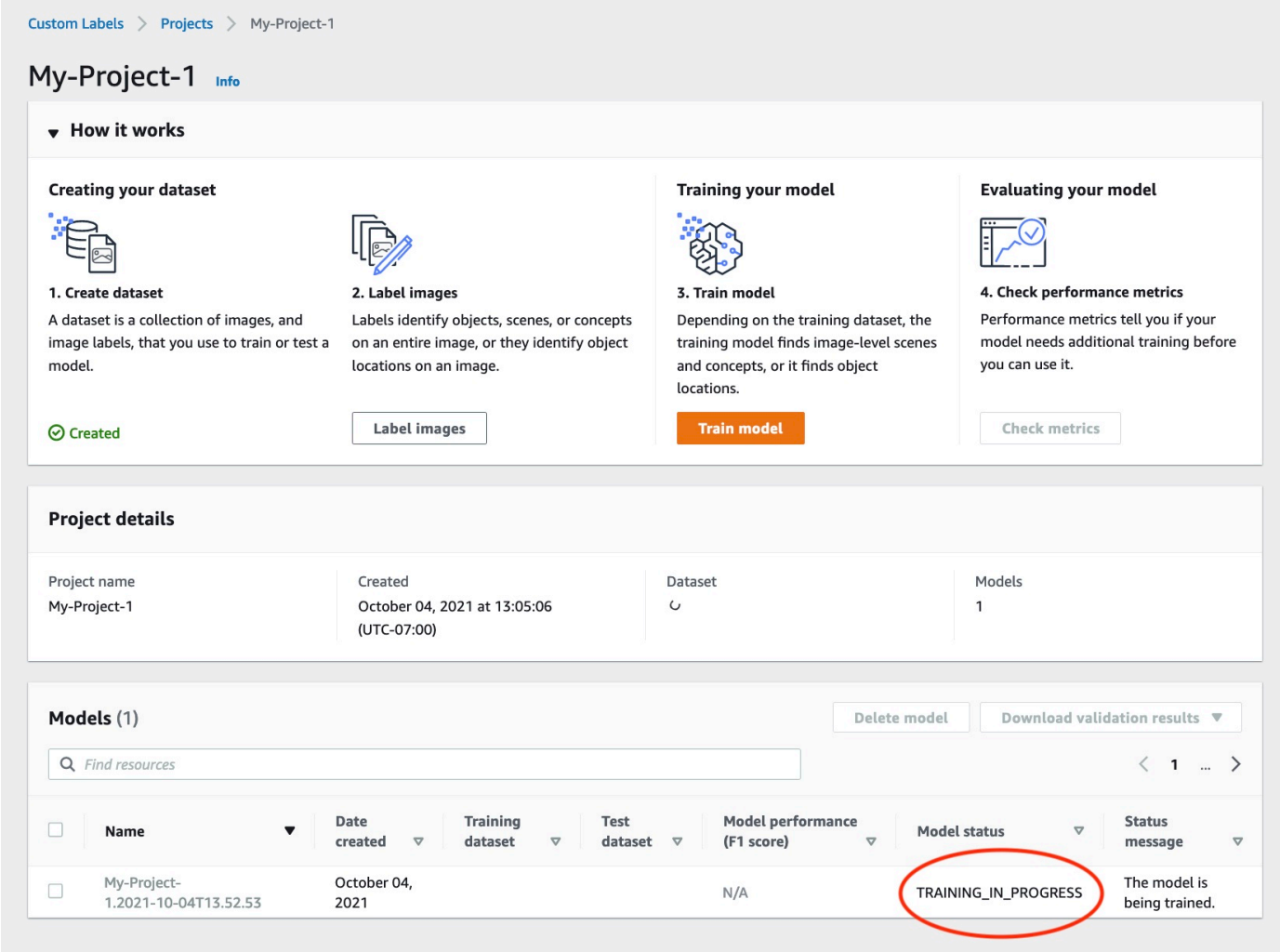
8. 在训练模型页面上，选择训练模型。项目的 Amazon 资源名称 (ARN) 应位于选择项目编辑框中。如果没有，请输入项目的 ARN。



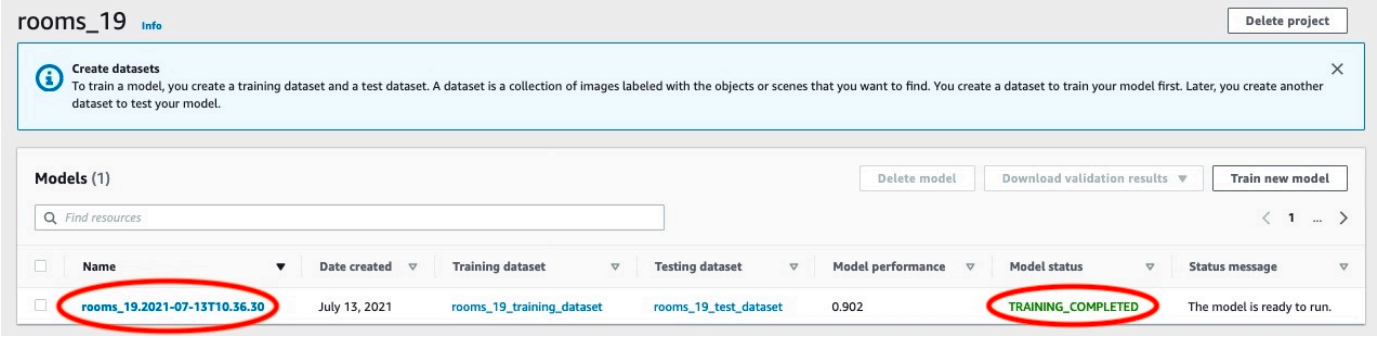
9. 在是否要训练您的模型？对话框中，选择训练模型。



10. 在项目页面的模型部分，可以在 Model Status 列中查看当前状态，状态显示训练正在进行。训练模型需要一些时间才能完成。



11. 训练完成后，选择模型名称。当模型状态为 TRAINING_COMPLETED 时，训练即告完成。如果训练失败，请参阅调试失败的模型训练。



12. 下一步：评估您的模型。有关更多信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。

训练模型 (SDK)

您可以通过调用 CreateProjectVersion 来训练模型。要训练模型，需要提供以下信息：

- 名称：模型版本的唯一名称。
- 项目 ARN：管理模型的项目的 Amazon 资源名称 (ARN)。
- 训练结果位置：存放结果的 Amazon S3 位置。可以使用与控制台 Amazon S3 存储桶相同的位置，也可以选择其他位置。建议您选择其他位置，因为这样您就可以设置权限，并避免与使用 Amazon Rekognition Custom Labels 控制台时的训练输出发生潜在的命名冲突。

训练使用与项目关联的训练和测试数据集。有关更多信息，请参阅管理数据集。

> **Note**
>
> 或者，也可以指定项目外部的训练和测试数据集清单文件。如果在使用外部清单文件训练模型后打开控制台，Amazon Rekognition Custom Labels 会使用最后一组用于训练的清单文件为您创建数据集。不能再通过指定外部清单文件来训练项目的模型版本。有关更多信息，请参阅 CreatePrjectVersion。

`CreateProjectVersion` 的响应是一个 ARN，用于在后续请求中识别模型版本。您还可以使用 ARN 来保护模型版本。有关更多信息，请参阅保护 Amazon Rekognition Custom Labels 项目。

训练模型版本需要一些时间才能完成。本主题中的 Python 和 Java 示例使用 waiter 来等待训练完成。waiter 是一种实用程序方法，用于轮询是否发生了特定状态。或者，您也可以通过调用 `DescribeProjectVersions` 获取训练的当前状态。当 `Status` 字段的值为 `TRAINING_COMPLETED` 时，即表示训练已完成。训练完成后，您可以通过查看评估结果来评估模型的质量。

训练模型 (SDK)

以下示例说明了如何使用与项目关联的训练和测试数据集来训练模型。

训练模型 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码来训练项目。

   AWS CLI

   以下示例会创建模型。会拆分训练数据集以创建测试数据集。替换以下内容：

   - 将 `my_project_arn` 替换为项目的 Amazon 资源名称 (ARN)。
   - 将 `version_name` 替换为您选择的唯一版本名称。
   - 将 `output_bucket` 替换为 Amazon Rekognition Custom Labels 保存训练结果的 Amazon S3 存储桶的名称。
   - 将 `output_folder` 替换为保存训练结果的文件夹的名称。
   - （可选参数）将 `--kms-key-id` 替换为您的 AWS Key Management Service 客户主密钥的标识符。

   ```
   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

   以下示例会创建模型。提供以下命令行参数：

- project_arn：项目的 Amazon 资源名称 (ARN)。
- version_name：您选择的模型的唯一版本名称。
- output_bucket：Amazon Rekognition Custom Labels 保存训练结果的 Amazon S3 存储桶的名称。
- output_folder：保存训练结果的文件夹的名称。

或者，提供以下命令行参数以将标签附加到模型：

- tag：您选择的要附加到模型的标签名称。
- tag_value：标签值。

```
#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

以下示例会训练模型。提供以下命令行参数：

- `project_arn`：项目的 Amazon 资源名称 (ARN)。
- `version_name`：您选择的模型的唯一版本名称。
- `output_bucket`：Amazon Rekognition Custom Labels 保存训练结果的 Amazon S3 存储桶的名称。
- `output_folder`：保存训练结果的文件夹的名称。

```
/*
   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. 如果训练失败，请参阅调试失败的模型训练。

调试失败的模型训练

在模型训练期间，您可能会遇到错误。Amazon Rekognition Custom Labels 会在控制台和 DescribeProjectVersions 的响应中报告训练错误。

错误要么是终止性错误（训练无法继续），要么是非终止性错误（训练可以继续）。对于与训练和测试数据集内容相关的错误，您可以下载验证结果（清单摘要以及训练和测试验证清单）。使用验证结果中的错误代码在本节中查找更多信息。本节还提供了清单文件错误（在验证清单文件内容之前发生的终止性错误）的信息。

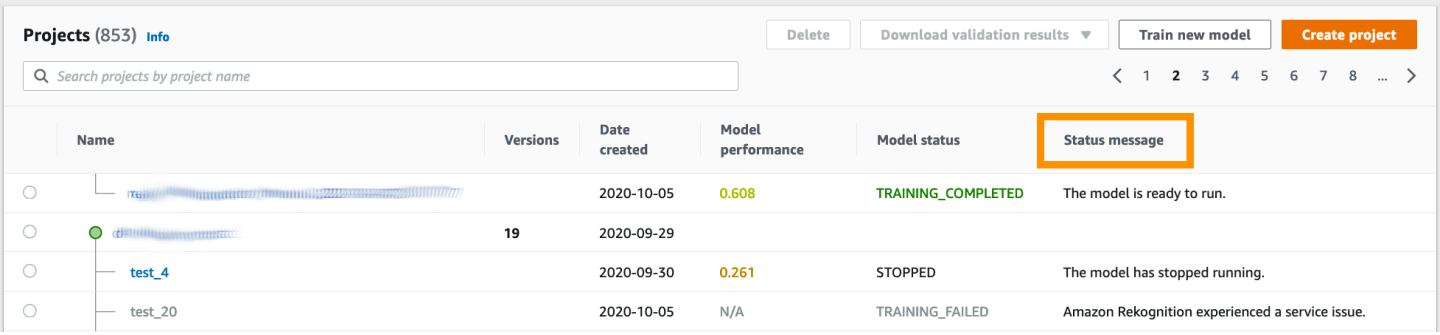
> **Note**
>
> 清单是用于存储数据集内容的文件。

可以使用 Amazon Rekognition Custom Labels 控制台修复部分错误。其他错误可能需要您更新训练或测试清单文件。您可能需要进行其他更改，例如 IAM 权限。有关更多信息，请参阅文档中的各个错误。

终止性错误

终止性错误会停止模型的训练。终止性训练错误分为 3 类：服务错误、清单文件错误和清单内容错误。

在控制台中，Amazon Rekognition Custom Labels 会在项目页面的状态消息列中显示模型的终止性错误。



如果使用的是 AWS SDK，则可以通过查看 DescribeProjectVersions 的响应来了解是否发生了终止性清单文件错误或终止性清单内容错误。在本例中，`Status` 值为 `TRAINING_FAILED`，`StatusMessage` 字段包含错误。

服务错误

当 Amazon Rekognition 遇到服务问题且无法继续训练时，就会发生终止性服务错误。例如，Amazon Rekognition Custom Labels 所依赖的另一项服务出现故障。当 Amazon Rekognition 遇到服务问题时，Amazon Rekognition Custom Labels 就会在控制台中报告服务错误。如果使用的是 AWS SDK，则训练期间发生的服务错误会通过 CreateProjectVersion 和 DescribeProjectVersions 引发 `InternalServerError` 异常。

如果发生服务错误，请重试模型训练。如果训练仍然失败，请联系 AWS Support，并附上与服务错误一起报告的所有错误信息。

终止性清单文件错误

清单文件错误是终止性错误，在训练和测试数据集中的文件级或跨多个文件发生。清单文件错误可在验证训练和测试数据集内容之前检测到。清单文件错误会阻止报告非终止性验证错误。例如，空的训练清单文件会生成清单文件为空错误。由于文件为空，因此无法报告任何非终止性 JSON 行验证错误。也不会创建清单摘要。

必须先修复清单文件错误，然后才能训练模型。

下面列出了清单文件错误。

- 清单文件扩展名或内容无效。
- 清单文件为空。
- 清单文件大小超过了支持的最大大小。
- 无法写入到输出 S3 存储桶。
- S3 存储桶权限不正确。

终止性清单内容错误

清单内容错误是与清单中的内容相关的终止性错误。例如，如果收到错误清单文件中每个标签包含的带标签图像不足，无法执行自动拆分，训练将无法完成，因为训练数据集中没有足够的带标签图像来创建测试数据集。

该错误除了在控制台和 DescribeProjectVersions 的响应中报告以外，还与任何其他终止性清单内容错误一起在清单摘要中报告。有关更多信息，请参阅了解清单摘要。

非终止性 JSON 行错误还会在单独的训练和测试验证结果清单中报告。Amazon Rekognition Custom Labels 发现的非终止性 JSON 行错误不一定与停止训练的清单内容错误有关。有关更多信息，请参阅了解训练和测试验证结果清单。

必须先修复清单内容错误，然后才能训练模型。

以下是清单内容错误的错误消息。

- 清单文件包含太多无效行。
- 清单文件包含来自多个 S3 存储桶的图像。
- 图像 S3 存储桶的拥有者 ID 无效。
- 清单文件中每个标签包含的带标签图像不足，无法执行自动拆分。
- 清单文件包含的标签太少。
- 清单文件包含的标签太多。
- 训练和测试清单文件之间的标签重叠度小于 {}%。
- 清单文件包含的可用标签太少。
- 训练和测试清单文件之间的可用标签重叠度小于 {}%。
- 无法从 S3 存储桶复制图像。

非终止性 JSON 行验证错误

JSON 行验证错误是非终止性错误，不需要 Amazon Rekognition Custom Labels 停止训练模型。

JSON 行验证错误不会显示在控制台中。

在训练和测试数据集中，JSON 行代表单张图像的训练或测试信息。JSON 行中的验证错误（例如图像无效）会在训练和测试验证清单中报告。Amazon Rekognition Custom Labels 会使用清单中的其他有效 JSON 行完成训练。有关更多信息，请参阅了解训练和测试验证结果清单。有关验证规则的信息，请参阅清单文件的验证规则。

> **Note**
>
> 如果 JSON 行错误太多，训练就会失败。

建议将非终止性 JSON 行错误也一并修复，因为它们可能会在将来引起错误或影响您的模型训练。

Amazon Rekognition Custom Labels 可能会生成以下非终止性 JSON 行验证错误。

- 缺少 source-ref 键。
- source-ref 值的格式无效。
- 未找到标签属性。
- 标签属性 {} 的格式无效。
- 标签属性元数据的格式无效。
- 未找到有效的标签属性。
- 一个或多个边界框缺少置信度值。
- 类别映射中缺少一个或多个类别 ID。
- JSON 行的格式无效。
- 图像无效。请检查 S3 路径和/或图像属性。
- 边界框具有超出边框的值。
- 边界框的高度和宽度太小。
- 边界框的数量超出了允许的最大数量。
- 未找到有效的注释。

了解清单摘要

清单摘要包含以下信息。

- 有关在验证期间发生的终止性清单内容错误的错误信息。
- 训练和测试数据集中的非终止性 JSON 行验证错误的错误位置信息。
- 错误统计信息，例如在训练和测试数据集中发现的无效 JSON 行总数。

如果没有终止性清单文件错误，则会在训练期间创建清单摘要。要获取清单摘要文件 (manifest_summary.json) 的位置，请参阅获取验证结果。

> **Note**
>
> 清单摘要中不会报告服务错误和清单文件错误。有关更多信息，请参阅终止性错误。

有关特定清单内容错误的信息，请参阅终止性清单内容错误。

清单摘要文件格式

清单文件包含 2 个部分：`statistics` 和 `errors`。

statistics

`statistics` 包含有关训练和测试数据集中错误的信息。

- `training`：训练数据集中的统计信息和发现的错误。
- `testing`：测试数据集中的统计信息和发现的错误。

`errors` 数组中的对象包含清单内容错误的错误代码和错误消息。

`error_line_indices` 数组包含训练或测试清单中存在错误的每个 JSON 行的行号。有关更多信息，请参阅修复训练错误。

errors

跨越训练和测试数据集的错误。例如，当没有足够的可用标签与训练和测试数据集重叠时，就会发生 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP。

```
{
    "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.
        }
    ]
```

```
}
```

示例清单摘要

以下示例是显示终止性清单内容错误 (ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST) 的一部分清单摘要。error_json_line_indices 数组包含相应训练或测试验证清单中非终止性 JSON 行错误的行号。

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

了解训练和测试验证结果清单

在训练期间，Amazon Rekognition Custom Labels 会创建验证结果清单来保存非终止性 JSON 行错误。验证结果清单是添加了错误信息的训练和测试数据集的副本。训练完成后，您可以访问验证清单。有关更多信息，请参阅获取验证结果。Amazon Rekognition Custom Labels 还会创建一份清单摘要，其中包含 JSON 行错误的概要信息，例如错误位置和 JSON 行错误计数。有关更多信息，请参阅了解清单摘要。

> **Note**
>
> 只有在没有终止性清单文件错误时，才会创建验证结果（训练和测试验证结果清单和清单摘要）。

清单包含数据集中每张图像的 JSON 行。在验证结果清单中，JSON 行错误信息会添加到发生错误的 JSON 行中。

JSON 行错误是指与单张图像相关的非终止性错误。非终止性验证错误可能会使整个 JSON 行或其中的一部分失效。例如，如果 JSON 行中引用的图像不是 PNG 或 JPG 格式，则会发生 ERROR_INVALID_IMAGE 错误并将整个 JSON 行排除在训练之外。继续使用其他有效的 JSON 行进行训练。

在一个 JSON 行中，错误可能意味着该 JSON 行仍可用于训练。例如，如果与标签关联的四个边界框中有一个的左值为负，则仍将使用其他有效的边界框训练模型。系统会返回无效边界框的 JSON 行错误信息 (ERROR_INVALID_BOUNDING_BOX)。在此示例中，错误信息会被添加到发生错误的 `annotation` 对象。

警告错误（例如 WARNING_NO_ANNOTATIONS）不会用于训练，并会在清单摘要中计为忽略的 JSON 行 (ignored_json_lines)。有关更多信息，请参阅了解清单摘要。此外，忽略的 JSON 行也不会计入训练和测试的 20% 错误阈值。

有关特定非终止性数据验证错误的信息，请参见非终止性 JSON 行验证错误。

> **Note**
>
> 如果数据验证错误太多，将会停止训练，并在清单摘要中报告 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST终止性错误。

有关更正 JSON 行错误的信息，请参阅修复训练错误。

JSON 行错误格式

Amazon Rekognition Custom Labels 会将非终止性验证错误信息添加到图像级和物体定位格式 JSON 行中。有关更多信息，请参阅the section called “创建清单文件”。

图像级错误

以下示例显示了图像级 JSON 行中的 Error 数组。有两组错误。与标签属性元数据相关的错误（在本例中为 sport-metadata）以及与图像相关的错误。错误中包含错误代码 (code) 和错误消息 (message)。有关更多信息，请参阅清单文件中的图像级标签。

```
{
    "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
        }
    ]
}
```

物体定位错误

以下示例显示了一个物体定位 JSON 行中的错误数组。该 JSON 行包含以下 JSON 行部分中的字段的 `Errors` 数组信息。每个 `Error` 对象都包含错误代码和错误消息。

- label attribute：标签属性字段的错误。请参阅示例中的 `bounding-box`。
- annotations：注释错误（边界框）存储在标签属性内的 `annotations` 数组中。
- label attribute-metadata：标签属性元数据的错误。请参阅示例中的 `bounding-box-metadata`。
- image：与标签属性、注释和标签属性元数据字段无关的错误。

有关更多信息，请参阅清单文件中的物体定位。

```
{
    "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
        }
    ]
```

```
}
```

JSON 行错误示例

以下物体定位 JSON 行（为便于阅读调整了格式）显示了 ERROR_BOUNDING_BOX_TOO_SMALL 错误。在本示例中，边界框尺寸（高度和宽度）不大于 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"
    }
}
```

获取验证结果

验证结果包含终止性清单内容错误和非终止性 JSON 行验证错误的错误信息。有三个验证结果文件。

- training_manifest_with_validation.json：添加了 JSON 行错误信息的训练数据集清单文件的副本。
- testing_manifest_with_validation.json：添加了 JSON 行错误信息的测试数据集清单文件的副本。
- manifest_summary.json：训练和测试数据集中发现的清单内容错误和 JSON 行错误的摘要。有关更多信息，请参阅了解清单摘要。

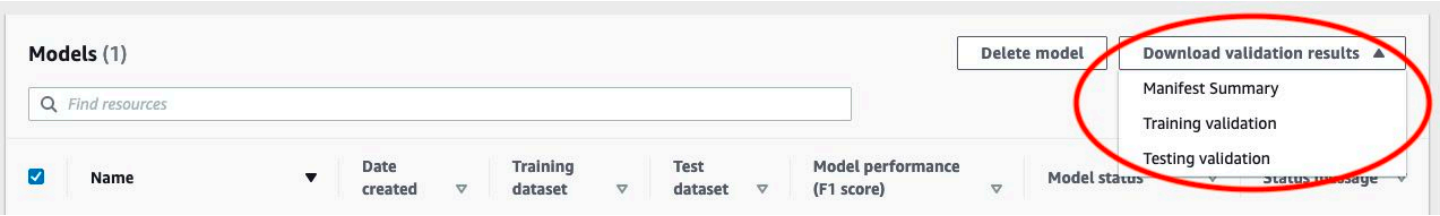
有关训练和测试验证清单内容的信息，请参阅调试失败的模型训练。

> **Note**
>
> - 只有在训练期间未发生终止性清单文件错误时，才会创建验证结果。
> - 如果在验证训练和测试清单后出现服务错误，则会创建验证结果，但 DescribeProjectVersions 的响应不会包含验证结果文件的位置。

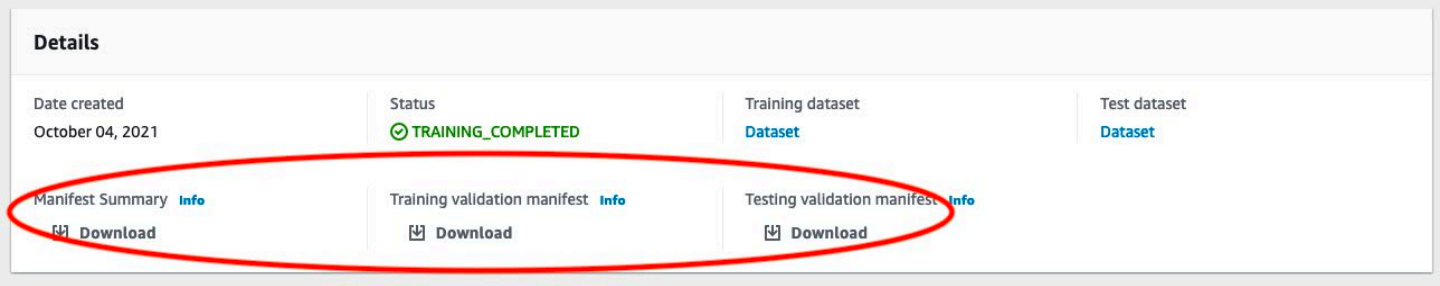
训练完成或失败后，可以使用 Amazon Rekognition Custom Labels 控制台下载验证结果，也可通过调用 DescribeProjectVersions API 获取 Amazon S3 存储桶的位置。

获取验证结果（控制台）

如果使用控制台训练模型，则可以从项目的模型列表下载验证结果，如下图所示。



您也可以从模型的详细信息页面下载验证结果。



有关更多信息，请参阅训练模型（控制台）。

获取验证结果 (SDK)

模型训练完成后，Amazon Rekognition Custom Labels 会将验证结果存储在训练期间指定的 Amazon S3 存储桶中。训练完成后，可以通过调用 DescribeProjectVersions API 获取 S3 存储桶的位置。要训练模型，请参阅训练模型 (SDK)。

系统会为训练数据集 (TrainingDataResult) 和测试数据集 (TestingDataResult) 返回一个 ValidationData 对象。清单摘要将在 `ManifestSummary` 中返回。

获取 Amazon S3 存储桶位置后，即可下载验证结果。有关更多信息，请参阅如何从 S3 存储桶下载对象？。您也可以使用 GetObject 操作。

获取验证数据 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例获取验证结果的位置。

Python

将 project_arn 替换为模型所属项目的 Amazon 资源名称 (ARN)。有关更多信息，请参阅管理 Amazon Rekognition Custom Labels 项目。将 version_name 替换为模型版本的名称。有关更多信息，请参阅训练模型 (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. 在程序输出中，注意 TestingDataResult 和 TrainingDataResult 对象中的 Validation 字段。清单摘要在 ManifestSummary 中。

修复训练错误

可以使用清单摘要来识别训练期间发生的终止性清单内容错误和非终止性 JSON 行验证错误。必须修复清单内容错误。建议将非终止性 JSON 行错误也一并修复。有关特定错误的信息，请参阅非终止性 JSON 行验证错误和终止性清单内容错误。

您可以修复用于训练的训练或测试数据集。或者，也可以在训练和测试验证清单文件中进行修复，然后使用它们来训练模型。

修复完毕后，您需要导入更新后的清单并重新训练模型。有关更多信息，请参阅创建清单文件。

以下过程说明了如何使用清单摘要来修复终止性清单内容错误。该过程还说明了如何定位和修复训练和测试验证清单中的 JSON 行错误。

修复 Amazon Rekognition Custom Labels 训练错误

1. 下载验证结果文件。文件名为 training_manifest_with_validation.json、testing_manifest_with_validation.json 和 manifest_summary.json。有关更多信息，请参阅获取验证结果。
2. 打开清单摘要文件 (manifest_summary.json)。
3. 修复清单摘要中的所有错误。有关更多信息，请参阅了解清单摘要。
4. 在清单摘要中，遍历 `training` 中的 `error_line_indices` 数组，并根据相应的 JSON 行号修复 `training_manifest_with_validation.json` 中的错误。有关更多信息，请参阅the section called “了解训练和测试验证结果清单”。
5. 遍历 `testing` 中的 `error_line_indices` 数组，并根据相应的 JSON 行号修复 `testing_manifest_with_validation.json` 中的错误。
6. 使用验证清单文件作为训练和测试数据集重新训练模型。有关更多信息，请参阅the section called “训练模型”。

如果您使用的是 AWS SDK 并选择修复训练或测试验证数据清单文件中的错误，请在 CreateProjectVersion 的 TrainingData 和 TestingData 输入参数中使用验证数据清单文件的位置。有关更多信息，请参阅训练模型 (SDK)。

JSON 行错误优先级

会先检测以下 JSON 行错误。如果出现这些错误中的任何一个，则会停止验证 JSON 行错误。必须先修复这些错误，然后才能修复任何其他 JSON 行错误

- 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

终止性清单文件错误

本主题将介绍终止性清单文件错误。清单文件错误没有相关的错误代码。当发生终止性清单文件错误时，将不会创建验证结果清单。有关更多信息，请参阅了解清单摘要。终止性清单错误会阻止报告非终止性 JSON 行验证错误。

清单文件扩展名或内容无效。

训练或测试清单文件没有文件扩展名或其内容无效。

修复清单文件扩展名或内容无效错误

- 在训练和测试清单文件中检查以下可能的原因。
  - 清单文件缺少文件扩展名。按照惯例，文件扩展名为 `.manifest`。
  - 找不到清单文件的 Amazon S3 存储桶或密钥。

清单文件为空。

存在用于训练的训练或测试清单文件，但文件是空的。清单文件需要为用于训练和测试的每张图像都提供一个 JSON 行。

修复清单文件为空错误

1. 检查哪些训练或测试清单是空的。
2. 将 JSON 行添加到空清单文件中。有关更多信息，请参阅创建清单文件。或者，使用控制台创建新数据集。有关更多信息，请参阅the section called “使用图像创建数据集”。

清单文件大小超过了支持的最大大小。

训练或测试清单文件大小（以字节为单位）太大。有关更多信息，请参阅Amazon Rekognition Custom Labels 中的准则和配额。清单文件可能会 JSON 行数少于最大值，但文件大小超过最大值。

无法使用 Amazon Rekognition Custom Labels 控制台修复错误：清单文件大小超过了支持的最大大小。

修复清单文件大小超过了支持的最大大小错误

1. 检查哪些训练和测试清单文件超过了最大文件大小。
2. 减少过大的清单文件中的 JSON 行数。有关更多信息，请参阅创建清单文件。

S3 存储桶权限不正确。

Amazon Rekognition Custom Labels 不具有对一个或多个包含训练和测试清单文件的存储桶的权限。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复 S3 存储桶权限不正确错误

- 检查对包含训练和测试清单的存储桶的权限。有关更多信息，请参阅步骤 2：设置 Amazon Rekognition Custom Labels 控制台权限。

无法写入到输出 S3 存储桶。

服务无法生成训练输出文件。

修复无法写入到输出 S3 存储桶错误

- 检查 CreateProjectVersion 的 OutputConfig 输入参数中的 Amazon S3 存储桶信息是否正确。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

终止性清单内容错误

本主题将介绍清单摘要中报告的终止性清单内容错误。清单摘要中包含检测到的每个错误的错误代码和错误消息。有关更多信息，请参阅了解清单摘要。终止性清单内容错误不会阻止报告非终止性 JSON 行验证错误。

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

错误消息

清单文件包含太多无效行。

更多信息

如果包含无效内容的 JSON 行过多，则会发生 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST 错误。

无法使用 Amazon Rekognition Custom Labels 控制台修复 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST 错误。

修复 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

1. 检查清单中是否包含 JSON 行错误。有关更多信息，请参阅了解训练和测试验证结果清单。
2. 修复有错误的 JSON 行。有关更多信息，请参阅非终止性 JSON 行验证错误。

ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS

错误消息

清单文件包含来自多个 S3 存储桶的图像。

更多信息

清单只能引用存储在单个存储桶中的图像。每个 JSON 行都将一张图像位置的 Amazon S3 位置存储在 source-ref 的值中。在以下示例中，存储桶名称为 my-bucket。

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

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复 **ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS**

- 确保所有图像都在同一 Amazon S3 存储桶中，并且每个 JSON 行中的 source-ref 的值都引用存储图像的存储桶。或者，也可以选择首选的 Amazon S3 存储桶，并删除其 source-ref 未引用首选存储桶的 JSON 行。

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

错误消息

对图像 S3 存储桶的权限无效。

更多信息

对包含图像的 Amazon S3 存储桶的权限不正确。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复 **ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET**

- 检查对包含图像的存储桶的权限。图像的 `source-ref` 的值包含存储桶位置。

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

错误消息

图像 S3 存储桶的拥有者 ID 无效。

更多信息

包含训练或测试图像的存储桶的拥有者与包含训练或测试清单的存储桶的拥有者不同。可以使用以下命令查找存储桶的拥有者。

```
aws s3api get-bucket-acl --bucket bucket name
```

`OWNER ID` 必须与存储图像和清单文件的存储桶相匹配。

修复 ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

1. 选择训练、测试、输出和图像存储桶的所需拥有者。拥有者必须有权使用 Amazon Rekognition Custom Labels。
2. 对于当前未由所需拥有者拥有的每个存储桶，请创建一个由首选拥有者拥有的新 Amazon S3 存储桶。
3. 将旧存储桶中的内容复制到新存储桶。有关更多信息，请参阅如何在 Amazon S3 存储桶之间复制对象？。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

错误消息

清单文件中每个标签包含的带标签图像不足，无法执行自动拆分。

更多信息

在模型训练期间，可以使用训练数据集中 20% 的图像来创建测试数据集。当没有足够的图像来创建可接受的测试数据集时，就会发生 ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT 错误。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复 ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

- 向训练数据集中添加更多带标签的图像。可以在 Amazon Rekognition Custom Labels 控制台中添加图像，方法是：向训练数据集添加图像，或者向训练清单中添加 JSON 行。有关更多信息，请参阅管理数据集。

ERROR_MANIFEST_TOO_FEW_LABELS

错误消息

清单文件包含的标签太少。

更多信息

训练和测试数据集具有最低标签数量要求。此最低数量取决于数据集训练/测试的模型是用于检测图像级标签（分类），还是检测物体位置。如果拆分训练数据集来创建测试数据集，则数据集中的标签数量将在拆分训练数据集后确定。有关更多信息，请参阅Amazon Rekognition Custom Labels 中的准则和配额。

修复 ERROR_MANIFEST_TOO_FEW_LABELS（控制台）

1. 向数据集添加更多新标签。有关更多信息，请参阅管理标签。
2. 向数据集中的图像添加新标签。如果模型用于检测图像级标签，请参阅为图像分配图像级标签。如果模型用于检测物体位置，请参阅the section called “使用边界框标注物体”。

修复 ERROR_MANIFEST_TOO_FEW_LABELS（JSON 行）

- 为带有新标签的新图像添加 JSON 行。有关更多信息，请参阅创建清单文件。如果模型用于检测图像级标签，则可以在 `class-name` 字段中添加新标签名称。例如，下图的标签是 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"
    }
}
```

如果模型用于检测物体位置，请添加新标签至 `class-map`，如以下示例所示。

```
{
 "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"
 }
}
```

您需要将类别映射表映射到边界框注释。有关更多信息，请参阅清单文件中的物体定位。

ERROR_MANIFEST_TOO_MANY_LABELS

错误消息

清单文件包含的标签太多。

更多信息

清单（数据集）中唯一标签的数量超过了允许的限制。如果拆分训练数据集来创建测试数据集，则标签数量将在拆分后确定。

修复 ERROR_MANIFEST_TOO_MANY_LABELS（控制台）

- 从数据集中移除标签。有关更多信息，请参阅管理标签。标签会自动从数据集中的图像和边界框中移除。

修复 ERROR_MANIFEST_TOO_MANY_LABELS（JSON 行）

- 清单包含图像级 JSON 行 - 如果图像只有一个标签，请移除使用所需标签的图像对应的 JSON 行。如果相关 JSON 行包含多个标签，请仅移除所需标签对应的 JSON 对象。有关更多信息，请参阅为图像添加多个图像级标签。

清单包含物体位置 JSON 行 - 移除要移除的标签所对应的边界框和关联的标签信息。对包含所需标签的每个 JSON 行执行此操作。需要从 `class-map` 数组中移除标签并从 `objects` 和 `annotations` 数组中移除相应的对象。有关更多信息，请参阅清单文件中的物体定位。

ERROR_INSUFFICIENT_LABEL_OVERLAP

错误消息

训练和测试清单文件之间的标签重叠度小于 {}%。

更多信息

测试数据集标签名称和训练数据集标签名称之间的重叠度小于 50%。

修复 ERROR_INSUFFICIENT_LABEL_OVERLAP（控制台）

- 从训练数据集中移除标签。或者，向测试数据集中添加更多共用标签。有关更多信息，请参阅管理标签。标签会自动从数据集中的图像和边界框中移除。

通过从训练数据集中移除标签来修复 ERROR_INSUFFICIENT_LABEL_OVERLAP（JSON 行）

- 清单包含图像级 JSON 行：如果图像只有一个标签，请移除使用所需标签的图像对应的 JSON 行。如果相关 JSON 行包含多个标签，请仅移除所需标签对应的 JSON 对象。有关更多信息，请参阅为图像添加多个图像级标签。对清单中每个包含您要移除的标签的 JSON 行执行此操作。

  清单包含物体位置 JSON 行 - 移除要移除的标签所对应的边界框和关联的标签信息。对包含所需标签的每个 JSON 行执行此操作。需要从 `class-map` 数组中移除标签并从 `objects` 和 `annotations` 数组中移除相应的对象。有关更多信息，请参阅清单文件中的物体定位。

通过向测试数据集中添加共用标签来修复 ERROR_INSUFFICIENT_LABEL_OVERLAP（JSON 行）

- 向测试数据集中添加 JSON 行，这些行应包含带有训练数据集中已有标签的图像。有关更多信息，请参阅创建清单文件。

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS

错误消息

清单文件包含的可用标签太少。

更多信息

训练清单可以包含采用图像级标签格式和物体位置格式的 JSON 行。根据在训练清单中找到的 JSON 行的类型，Amazon Rekognition Custom Labels 会选择创建检测图像级标签或检测物体位置的模型。Amazon Rekognition Custom Labels 会过滤掉未采用所选格式的 JSON 行的有效 JSON 记录。当所选模型类型清单中的标签数量不足以训练模型时，就会发生 ERROR_MANIFEST_TOO_FEW_USABLE_LABELS 错误。

训练检测图像级标签的模型至少需要 1 个标签。训练检测物体位置的模型至少需要 2 个标签。

修复 ERROR_MANIFEST_TOO_FEW_USABLE_LABELS（控制台）

1. 检查清单摘要中的 use_case 字段。
2. 针对与 use_case 的值匹配的使用场景（图像级或物体定位），向训练数据集中添加更多标签。有关更多信息，请参阅管理标签。标签会自动从数据集中的图像和边界框中移除。

修复 ERROR_MANIFEST_TOO_FEW_USABLE_LABELS（JSON 行）

1. 检查清单摘要中的 use_case 字段。
2. 针对与 use_case 的值匹配的使用场景（图像级或物体定位），向训练数据集中添加更多标签。有关更多信息，请参阅创建清单文件。

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

错误消息

训练和测试清单文件之间的可用标签重叠度小于 {}%。

更多信息

训练清单可以包含采用图像级标签格式和物体位置格式的 JSON 行。根据在训练清单中找到的格式，Amazon Rekognition Custom Labels 会选择创建检测图像级标签的模型或检测物

体位置的模型。Amazon Rekognition Custom Labels 不会使用未采用所选模型格式的 JSON 行的有效 JSON 记录。当所用的测试标签和训练标签之间的重叠度小于 50% 时，就会出现 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP 错误。

修复 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP（控制台）

- 从训练数据集中移除标签。或者，向测试数据集中添加更多共用标签。有关更多信息，请参阅管理标签。标签会自动从数据集中的图像和边界框中移除。

通过从训练数据集中移除标签来修复 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP（JSON 行）

- 用于检测图像级标签的数据集：如果图像只有一个标签，请移除使用所需标签的图像对应的 JSON 行。如果相关 JSON 行包含多个标签，请仅移除所需标签对应的 JSON 对象。有关更多信息，请参阅为图像添加多个图像级标签。对清单中每个包含您要移除的标签的 JSON 行执行此操作。

  用于检测物体位置的数据集：移除要移除的标签对应的边界框和关联的标签信息。对包含所需标签的每个 JSON 行执行此操作。需要从 `class-map` 数组中移除标签并从 `objects` 和 `annotations` 数组中移除相应的对象。有关更多信息，请参阅清单文件中的物体定位。

通过向测试数据集中添加共用标签来修复 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP（JSON 行）

- 向测试数据集中添加 JSON 行，这些行应包含带有训练数据集中已有标签的图像。有关更多信息，请参阅创建清单文件。

ERROR_FAILED_IMAGES_S3_COPY

错误消息

无法从 S3 存储桶复制图像。

更多信息

服务无法复制数据集中的任何图像。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复 ERROR_FAILED_IMAGES_S3_COPY

1. 检查对图像的权限。
2. 如果使用的是 AWS KMS，请检查存储桶策略。有关更多信息，请参阅解密使用 AWS Key Management Service 加密的文件。

清单文件有太多终止性错误。

有太多存在终止性内容错误的 JSON 行。

修复 **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- 减少有终止性内容错误的 JSON 行（图像）的数量。有关更多信息，请参阅终止性清单内容错误。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

非终止性 JSON 行验证错误

本主题列出了 Amazon Rekognition Custom Labels 在训练期间报告的非终止性 JSON 行验证错误。这些错误会在训练和测试验证清单中报告。有关更多信息，请参阅了解训练和测试验证结果清单。可以通过更新训练或测试清单文件中的 JSON 行来修复非终止性 JSON 行错误。您也可以从清单中删除相关的 JSON 行，但这样做可能会降低模型的质量。如果非终止性验证错误太多，您可能会发现重新创建清单文件更容易。验证错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。有关修复验证错误的信息，请参阅修复训练错误。有些错误可以使用 Amazon Rekognition Custom Labels 控制台修复。

ERROR_MISSING_SOURCE_REF

错误消息

缺少 source-ref 键。

更多信息

JSON 行的 `source-ref` 字段提供图像的 Amazon S3 位置。如果 `source-ref` 键缺失或拼写错误，便会发生此错误。此错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。

修复 ERROR_MISSING_SOURCE_REF

1. 检查 source-ref 键是否存在且拼写正确。完整的 source-ref 键和值类似如下："source-ref": "s3://bucket/path/image"。
2. 更新 JSON 行中的 source-ref 键。或者，从清单文件中删除该 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_INVALID_SOURCE_REF_FORMAT

错误消息

source-ref 值的格式无效。

更多信息

JSON 行中具有 source-ref 键，但 Amazon S3 路径的架构不正确。例如，路径是 https://... 而不是 S3://....。ERROR_INVALID_SOURCE_REF_FORMAT 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。

修复 ERROR_INVALID_SOURCE_REF_FORMAT

1. 检查架构是否为 "source-ref": "s3://bucket/path/image"。例如，"source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"。
2. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此 ERROR_INVALID_SOURCE_REF_FORMAT。

ERROR_NO_LABEL_ATTRIBUTES

错误消息

未找到标签属性。

更多信息

标签属性或标签属性 -metadata 键名（或两者）无效或缺失。在以下示例中，每当缺少 bounding-box 或 bounding-box-metadata 键（或两者）时都会发生 ERROR_NO_LABEL_ATTRIBUTES。有关更多信息，请参阅创建清单文件。

```
{
 "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 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。

修复 ERROR_NO_LABEL_ATTRIBUTES

1. 检查标签属性标识符和标签属性标识符 -metadata 键是否存在，以及键名称拼写是否正确。
2. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复 ERROR_NO_LABEL_ATTRIBUTES。

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

错误消息

标签属性 {} 的格式无效。

更多信息

标签属性键的架构缺失或无效。ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。

修复 ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

1. 检查 JSON 行的标签属性键部分是否正确。在以下物体位置示例中，image_size 和 annotations 对象必须正确。该标签属性键名为 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. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

错误消息

标签属性元数据的格式无效。

更多信息

标签属性元数据键的架构缺失或无效。ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。

修复 **ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT**

1. 检查标签属性元数据键的 JSON 行架构是否与以下示例类似。该标签属性元数据键名为 `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. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_NO_VALID_LABEL_ATTRIBUTES

错误消息

未找到有效的标签属性。

更多信息

在 JSON 行中未找到有效的标签属性。Amazon Rekognition Custom Labels 会同时检查标签属性和标签属性标识符。ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅创建清单文件。

如果 JSON 行不是受支持的 SageMaker 清单格式，Amazon Rekognition Custom Labels 会将 JSON 行标记为无效并报告 ERROR_NO_VALID_LABEL_ATTRIBUTES 错误。目前，Amazon Rekognition Custom Labels 支持分类作业和边界框格式。有关更多信息，请参阅创建清单文件。

修复 **ERROR_NO_VALID_LABEL_ATTRIBUTES**

1. 检查 JSON 行中的标签属性键和标签属性元数据是否正确。
2. 更新或移除清单文件中的相关 JSON 行。有关更多信息，请参阅the section called “创建清单文件”。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

错误消息

一个或多个边界框缺少置信度值。

更多信息

一个或多个物体位置边界框缺少 confidence 键。边界框的 confidence 键位于标签属性元数据中，如以下示例中所示。ERROR_MISSING_BOUNDING_BOX_CONFIDENCE 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅the section called “清单文件中的物体定位”。

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

```
  }, {
   "confidence": 1
  }],
```

修复 **ERROR_MISSING_BOUNDING_BOX_CONFIDENCE**

1. 检查标签属性中的 `objects` 数组包含的 confidence 键数量是否与标签属性 `annotations` 数组中的对象数量相同。
2. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_MISSING_CLASS_MAP_ID

错误消息

类别映射中缺少一个或多个类别 ID。

更多信息

注释（边界框）对象中的 `class_id` 在标签属性元数据类别映射 (`class-map`) 中没有匹配的条目。有关更多信息，请参阅清单文件中的物体定位。ERROR_MISSING_CLASS_MAP_ID 错误通常发生在手动创建的清单文件中。

修复 ERROR_MISSING_CLASS_MAP_ID

1. 检查每个注释（边界框）对象中的 `class_id` 值在 `class-map` 数组中是否具有对应的值，如以下示例中所示。`annotations` 数组和 `class_map` 数组具有的元素数量应该相同。

```
{
 "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. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_INVALID_JSON_LINE

错误消息

JSON 行的格式无效。

更多信息

在 JSON 行中发现了意外字符。JSON 行被替换成了仅包含错误信息的新 JSON 行。ERROR_INVALID_JSON_LINE 错误通常发生在手动创建的清单文件中。有关更多信息，请参阅the section called “清单文件中的物体定位”。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

修复 ERROR_INVALID_JSON_LINE

1. 打开清单文件并导航到发生 ERROR_INVALID_JSON_LINE 错误的 JSON 行。
2. 检查该 JSON 行是否包含无效字符，以及是否缺少必要的 ; 或 , 字符。
3. 更新或移除清单文件中的相关 JSON 行。

ERROR_INVALID_IMAGE

错误消息

图像无效。请检查 S3 路径和/或图像属性。

更多信息

`source-ref` 引用的文件不是有效的图像。可能的原因包括图像的宽高比、图像大小和图像格式。

有关更多信息，请参阅准则和配额。

修复 ERROR_INVALID_IMAGE

1. 检查以下事项。
   - 图像的宽高比是否小于 20:1。
   - 图像大小是否大于 15 MB。
   - 图像是否为 PNG 或 JPEG 格式。
   - `source-ref` 中的图像路径是否正确。
   - 图像的最小图像尺寸是否大于 64 像素 x 64 像素。
   - 图像的最大图像尺寸是否大于 4096 像素 x 4096 像素。
2. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_INVALID_IMAGE_DIMENSION

错误消息

图像尺寸不符合允许的尺寸。

更多信息

source-ref 引用的图像不符合允许的图像尺寸。最小尺寸为 64 像素。最大尺寸为 4096 像素。会针对带边界框的图像报告 ERROR_INVALID_IMAGE_DIMENSION。

有关更多信息，请参阅准则和配额。

修复 **ERROR_INVALID_IMAGE_DIMENSION**（控制台）

1. 使用 Amazon Rekognition Custom Labels 能够处理的尺寸更新 Amazon S3 存储桶中的图像。
2. 在 Amazon Rekognition Custom Labels 控制台中，执行以下操作：
   a. 从图像中移除现有的边界框。
   b. 将边界框重新添加到图像中。
   c. 保存您的更改。

   有关更多信息，请参阅使用边界框标注物体。

修复 **ERROR_INVALID_IMAGE_DIMENSION** (SDK)

1. 使用 Amazon Rekognition Custom Labels 能够处理的尺寸更新 Amazon S3 存储桶中的图像。
2. 通过调用 ListDatasetEntries 获取该图像的现有 JSON 行。对于 SourceRefContains 输入参数，请指定图像的 Amazon S3 位置和文件名。
3. 调用 UpdateDatasetEntries 并为图像提供 JSON 行。确保 source-ref 的值与 Amazon S3 存储桶中的图像位置匹配。更新边界框注释，使其与更新后的图像所需的边界框尺寸相匹配。

```
{
 "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

错误消息

边界框具有超出边框的值。

更多信息

边界框信息指定的图像要么不在图像边框内，要么包含负值。

有关更多信息，请参阅准则和配额。

修复 ERROR_INVALID_BOUNDING_BOX

1. 检查 annotations 数组中边界框的值。

```
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }]
 },
```

2. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_NO_VALID_ANNOTATIONS

错误消息

未找到有效的注释。

更多信息

JSON 行中的所有注释对象均不包含有效的边界框信息。

修复 ERROR_NO_VALID_ANNOTATIONS

1. 更新 `annotations` 数组以包含有效的边界框对象。此外，请检查标签属性元数据中的相应边界框信息（`confidence` 和 `class_map`）是否正确。有关更多信息，请参阅清单文件中的物体定位。

```
{
 "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. 更新或移除清单文件中的相关 JSON 行。

无法使用 Amazon Rekognition Custom Labels 控制台修复此错误。

ERROR_BOUNDING_BOX_TOO_SMALL

错误消息

边界框的高度和宽度太小。

更多信息

边界框尺寸（高度和宽度）必须大于 1 x 1 像素。

在训练过程中，如果图像的任何尺寸大于 1280 像素，Amazon Rekognition Custom Labels 就会调整图像的大小（源图像不受影响）。生成的边界框高度和宽度必须大于 1 x 1 像素。边界框位置存储在物体位置 JSON 行的 annotations 数组中。有关更多信息，请参阅清单文件中的物体定位。

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

错误信息会被添加到注释对象中。

修复 ERROR_BOUNDING_BOX_TOO_SMALL

- 选择以下选项之一。
  - 增大太小的边界框的大小。
  - 移除太小的边界框。有关移除边界框的信息，请参阅ERROR_TOO_MANY_BOUNDING_BOXES。
  - 从清单中移除图像（JSON 行）。

ERROR_TOO_MANY_BOUNDING_BOXES

错误消息

边界框的数量超出了允许的最大数量。

更多信息

边界框的数量超出了允许的最大数量 (50)。您可以在 Amazon Rekognition Custom Labels 控制台中移除多余的边界框，也可从 JSON 行中将其移除。

修复 **ERROR_TOO_MANY_BOUNDING_BOXES**（控制台）。

1. 决定要移除哪些边界框。
2. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
3. 选择使用自定义标签。
4. 选择开始。
5. 在左侧导航窗格中，选择包含要使用的数据集的项目。
6. 在数据集部分中，选择要使用的数据集。
7. 在数据集库页面中，选择开始标注进入标注模式。
8. 选择要从中移除边界框的图像。
9. 选择绘制边界框。
10. 在绘图工具中，选择要删除的边界框。
11. 按键盘上的删除键即可删除边界框。
12. 重复前面的 2 个步骤，直到删除了足够的边界框。
13. 选择完成。
14. 选择保存更改以保存您的更改。
15. 选择退出，退出标注模式。

修复 ERROR_TOO_MANY_BOUNDING_BOXES（JSON 行）。

1. 打开清单文件并导航到发生 ERROR_TOO_MANY_BOUNDING_BOXES 错误的 JSON 行。
2. 对于要移除的每个边界框，移除以下内容。
   - 从 `annotations` 数组中移除所需的 `annotation` 对象。

- 从标签属性元数据中的 objects 数组中移除相应的 confidence 对象。
- 如果该标签不再被其他边界框使用，请从 class-map 中将其移除。

使用以下示例来确定要移除的项目。

```
{
 "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

警告消息

记录未添加注释。

更多信息

使用 Amazon Rekognition Custom Labels 控制台添加到数据集的图像未添加标签。图像对应的 JSON 行不会用于训练。

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "warnings": [
        {
            "code": "WARNING_UNANNOTATED_RECORD",
            "message": "Record is unannotated."
        }
    ]
}
```

修复 WARNING_UNANNOTATED_RECORD

- 使用 Amazon Rekognition Custom Labels 控制台为图像添加标签。有关说明，请参阅为图像分配图像级标签。

WARNING_NO_ANNOTATIONS

警告消息

未提供注释。

更多信息

尽管由人工 (human-annotated = yes) 注释，但物体定位格式的 JSON 行不包含任何边界框信息。该 JSON 行有效，但不会用于训练。有关更多信息，请参阅了解训练和测试验证结果清单。

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

修复 WARNING_NO_ANNOTATIONS

- 选择以下选项之一。
  - 将边界框 (annotations) 信息添加到 JSON 行。有关更多信息，请参阅清单文件中的物体定位。
  - 从清单中移除图像（JSON 行）。

WARNING_NO_ATTRIBUTE_ANNOTATIONS

警告消息

未提供属性注释。

更多信息

尽管由人工 (human-annotated = yes) 注释，但物体定位格式的 JSON 行不包含任何边界框注释信息。annotations 数组不存在或未填充。该 JSON 行有效，但不会用于训练。有关更多信息，请参阅了解训练和测试验证结果清单。

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

修复 WARNING_NO_ATTRIBUTE_ANNOTATIONS

- 选择以下选项之一。
  - 将一个或多个边界框 annotation 对象添加到 JSON 行。有关更多信息，请参阅清单文件中的物体定位。
  - 移除边界框属性。
  - 从清单中移除图像（JSON 行）。如果 JSON 行中存在其他有效的边界框属性，则可以仅从 JSON 行中移除无效的边界框属性。

ERROR_UNSUPPORTED_USE_CASE_TYPE

警告消息

更多信息

type 字段的值不是 groundtruth/image-classification 或 groundtruth/object-detection。有关更多信息，请参阅创建清单文件。

```
{
```

```
    "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."
        }
    ]
}
```

修复 ERROR_UNSUPPORTED_USE_CASE_TYPE

- 选择以下选项之一：

- 根据要创建的模型类型，将 `type` 字段的值更改为 `groundtruth/image-classification` 或 `groundtruth/object-detection`。有关更多信息，请参阅创建清单文件。
- 从清单中移除图像（JSON 行）。

ERROR_INVALID_LABEL_NAME_LENGTH

更多信息

标签名称的长度太长。最大长度为 256 个字符。

修复 ERROR_INVALID_LABEL_NAME_LENGTH

- 选择以下选项之一：
  - 将标签名称的长度减少到 256 个字符以内。
  - 从清单中移除图像（JSON 行）。

改进经过训练的 Amazon Rekognition Custom Labels 模型

训练完成后，您将评估模型的性能。为了帮助您评估，Amazon Rekognition Custom Labels 会提供每个标签的摘要指标和评估指标。有关可用的指标的信息，请参阅评估模型的指标。要使用指标改进模型，请参阅改进 Amazon Rekognition Custom Labels 模型。

如果对模型的准确性感到满意，则可以开始使用它。有关更多信息，请参阅运行经过训练的 Amazon Rekognition Custom Labels 模型。

主题

评估模型的指标

模型训练完毕后，Amazon Rekognition Custom Labels 会返回模型测试的指标，您可以使用这些指标来评估模型的性能。本主题介绍可供您使用的指标，以及如何了解您训练的模型是否运行良好。

Amazon Rekognition Custom Labels 控制台提供以下指标作为训练结果摘要和每个标签的指标：

- 精度
- 召回率
- F1

我们提供的每个指标都是评估机器学习模型性能的常用指标。Amazon Rekognition Custom Labels 会返回整个测试数据集的测试结果指标，以及每个自定义标签的指标。您还可以查看训练后的自定义模型在测试数据集中每张图像上的表现。有关更多信息，请参阅获取评估指标（控制台）。

评估模型性能

在测试期间，Amazon Rekognition Custom Labels 会预测测试图像中是否包含自定义标签。置信度分数是一个量化模型预测确定性的值。

如果自定义标签的置信度分数超过阈值，则模型输出将包含此标签。预测可以按以下方式进行分类：

- 真正例：Amazon Rekognition Custom Labels 模型可以正确预测测试图像中是否存在自定义标签。也就是说，预测的标签也是该图像的“ground truth”标签。例如，当图像中存在足球时，Amazon Rekognition Custom Labels 会正确返回 soccer ball 标签。
- 假正例：Amazon Rekognition Custom Labels 模型无法正确预测测试图像中是否存在自定义标签。也就是说，预测的标签不是该图像的“ground truth”标签。例如，Amazon Rekognition Custom Labels 会返回一个 soccer ball 标签，但该图像的 ground truth 中不包含 soccer ball 标签。
- 假负例：Amazon Rekognition Custom Labels 模型并不能预测图像中存在某个自定义标签，但该图像的“ground truth”包含此标签。例如，Amazon Rekognition Custom Labels 不会为包含足球的图像返回“soccer ball”自定义标签。
- 真负例：Amazon Rekognition Custom Labels 模型可以正确预测测试图像中不存在某个自定义标签。例如，对于不包含足球的图像，Amazon Rekognition Custom Labels 不会返回 soccer ball 标签。

可通过控制台获取测试数据集中每张图像的真正例、假正例和假负例值。有关更多信息，请参阅获取评估指标（控制台）。

这些预测结果会用于计算每个标签的以下指标，以及整个测试集的汇总。相同的定义适用于模型在边界框级做出的预测，不同之处在于，所有指标都是针对每个测试图像中的每个边界框（预测或 ground truth）计算得出的。

交并比（Intersection over Union，IoU）和物体检测

IoU 用于测量两个物体边界框的重叠部分占其合并区域的百分比。范围为 0（最低重叠）到 1（完全重叠）。在测试过程中，当 ground truth 边界框与预测的边界框的 IoU 不低于 0.5 时，即表示预测的边界框是正确的。

假设阈值

Amazon Rekognition Custom Labels 会自动为您的每个自定义标签计算一个假设阈值 (0-1)。无法为自定义标签设置假设阈值。每个标签的假设阈值都是预测被计为真正例或假正例的阈值。该阈值根据您的测试数据集设置。假设阈值是根据模型训练期间在测试数据集上获得的最佳 F1 分数计算得出的。

可以从模型的训练结果中获取标签的假设阈值。有关更多信息，请参阅获取评估指标（控制台）。

更改假设阈值通常用于提高模型的精度和召回率。有关更多信息，请参阅改进 Amazon Rekognition Custom Labels 模型。虽然无法设置模型对于标签的假设阈值，但可以通过使用 `DetectCustomLabels` 分析图像并指定 `MinConfidence` 输入参数来达到相同的效果。有关更多信息，请参阅使用经过训练的模型分析图像。

精度

Amazon Rekognition Custom Labels 提供每个标签的精度指标，以及整个测试数据集的平均精度指标。

精度是指在单个标签的假设阈值下，正确预测（真正例）数量占所有模型预测（真正例与假正例之和）数量的比例。随着阈值的增加，模型做出的预测可能会减少。但是，总的来说，与较低的阈值相比，更高的阈值下，真正例与假正例数量之比会更高。可能的精度值介于 0-1 之间，值越高表示精度越高。

例如，当模型预测图像中有足球时，预测正确的概率是多少？ 假设有一张包含 8 个足球和 5 个石头的图像。如果模型预测有 9 个足球（8 个正确预测，1 个假正例），则此示例的精度为 0.89。但是，如果模型预测图像中有 13 个足球，其中有 8 个正确预测，5 个错误，则产生的精度会更低。

有关更多信息，请参阅精度和召回率。

召回率

Amazon Rekognition Custom Labels 提供每个标签的平均召回率指标，以及整个测试数据集的平均召回率指标。

召回率是指正确预测的假设阈值之上的测试集标签所占的比例。该指标可以衡量当测试集图像中确实存在自定义标签时，模型可以正确预测该自定义标签的频率。召回率介于 0-1 之间。值越高表示召回率越高。

例如，如果一张图像包含 8 个足球，其中有多少被正确检测到？ 在此示例中，图像中包含 8 个足球和 5 个石头，如果模型检测到 5 个足球，则召回率为 0.62。如果在重新训练后，新模型检测到 9 个足球，包含了图像中存在的所有 8 个足球，则召回率为 1.0。

有关更多信息，请参阅精度和召回率。

F1

Amazon Rekognition Custom Labels 使用 F1 分数指标来衡量每个标签的平均模型性能和整个测试数据集的平均模型性能。

模型性能是一个综合衡量指标，它考虑了所有标签的精度和召回率。（例如，F1 分数或平均精度）。模型性能分数是介于 0 和 1 之间的值。该值越高，表明模型在召回率和精度方面的表现越好。具体而言，分类任务的模型性能通常由 F1 分数衡量。该分数是假设阈值下的精度和召回率分数的调和平均数。例如，对于精度为 0.9、召回率为 1.0 的模型，F1 分数为 0.947。

较高的 F1 分数值表示模型在精度和召回率方面都表现良好。如果模型表现不佳，例如精度低至 0.30，召回率高达 1.0，则 F1 分数为 0.46。同样，如果精度较高 (0.95)，而召回率较低 (0.20)，则 F1 分数为 0.33。这两种情况下，F1 分数都较低，表明模型存在问题。

有关更多信息，请参阅 F1 分数。

使用指标

对于经过训练的给定模型，根据应用程序的不同，可以使用 DetectCustomLabels 的 MinConfidence 输入参数在精度和召回率之间进行平衡。MinConfidence 值越高，通常会获得更高的精度（正确预测的足球数量更多），但召回率越低（会漏掉更多实际的足球）。MinConfidence 值越低，召回率越高（正确预测的实际足球更多），但精度越低（这些预测中出错情况的更多）。有关更多信息，请参阅使用经过训练的模型分析图像。

这些指标还会告知您，如果需要，您可以采取哪些步骤来改进模型性能。有关更多信息，请参阅改进 Amazon Rekognition Custom Labels 模型。

> **Note**
>
> DetectCustomLabels 会返回介于 0 到 100 之间的预测值，对应于 0-1 的指标范围。

获取评估指标（控制台）

在测试期间，将根据测试数据集评估模型的性能。测试数据集中的标签被视为“ground truth”，因为它们代表了实际图像所代表的内容。在测试期间，模型使用测试数据集进行预测。将预测的标签与“ground truth”标签进行比较，结果可在控制台评估页面中找到。

Amazon Rekognition Custom Labels 控制台会显示整个模型的摘要指标和单个标签的指标。控制台中提供的指标包括精度、召回率、F1 分数、置信度和置信度阈值。有关更多信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。

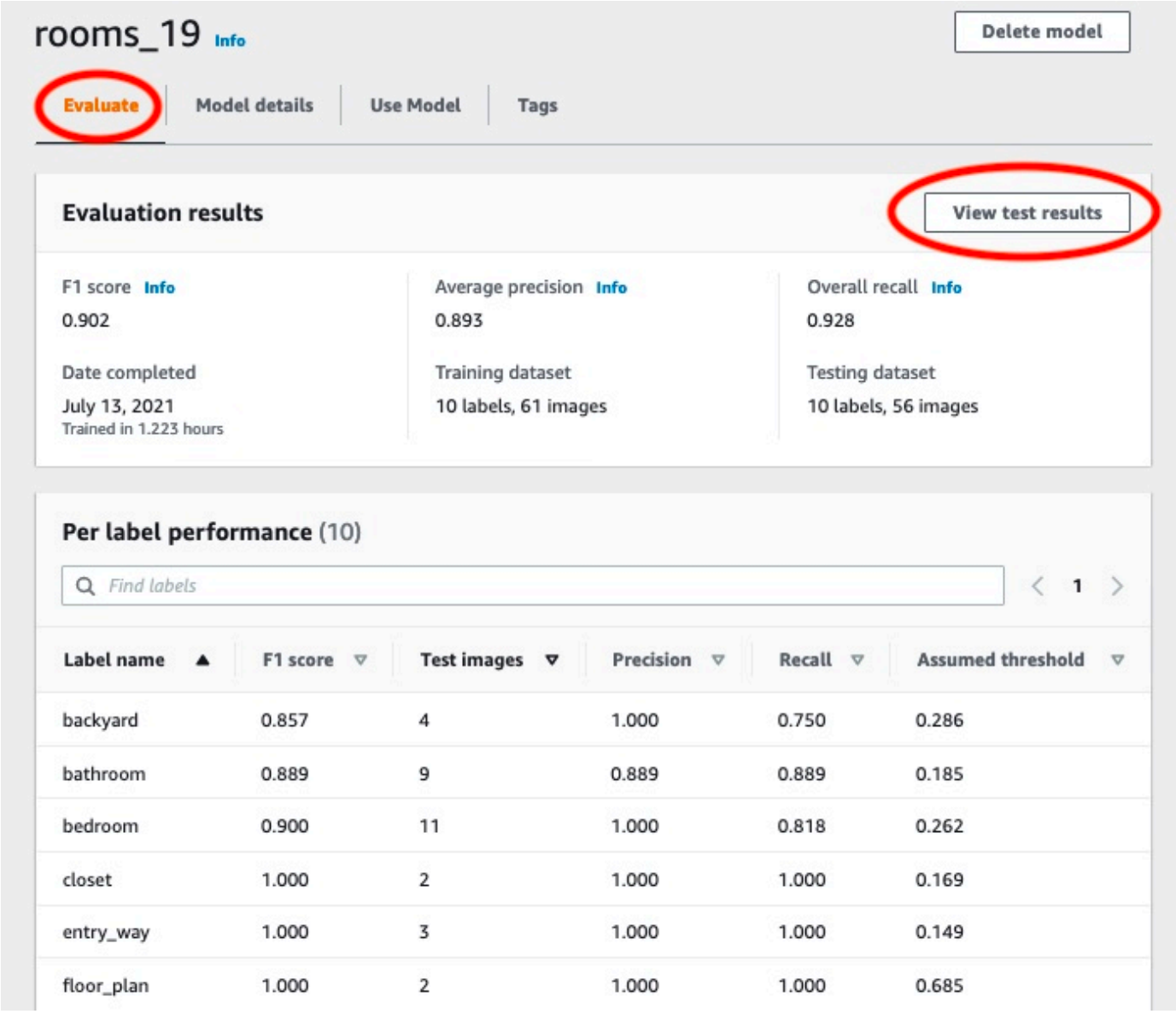
您可以通过控制台重点关注单个指标。例如，要调查标签的精度问题，可以按标签和假正例结果筛选训练结果。有关更多信息，请参阅评估模型的指标。

训练完成后，训练数据集为只读状态。如果决定改进模型，可以将训练数据集复制到新的数据集中。您可以使用该数据集的副本来训练模型的新版本。

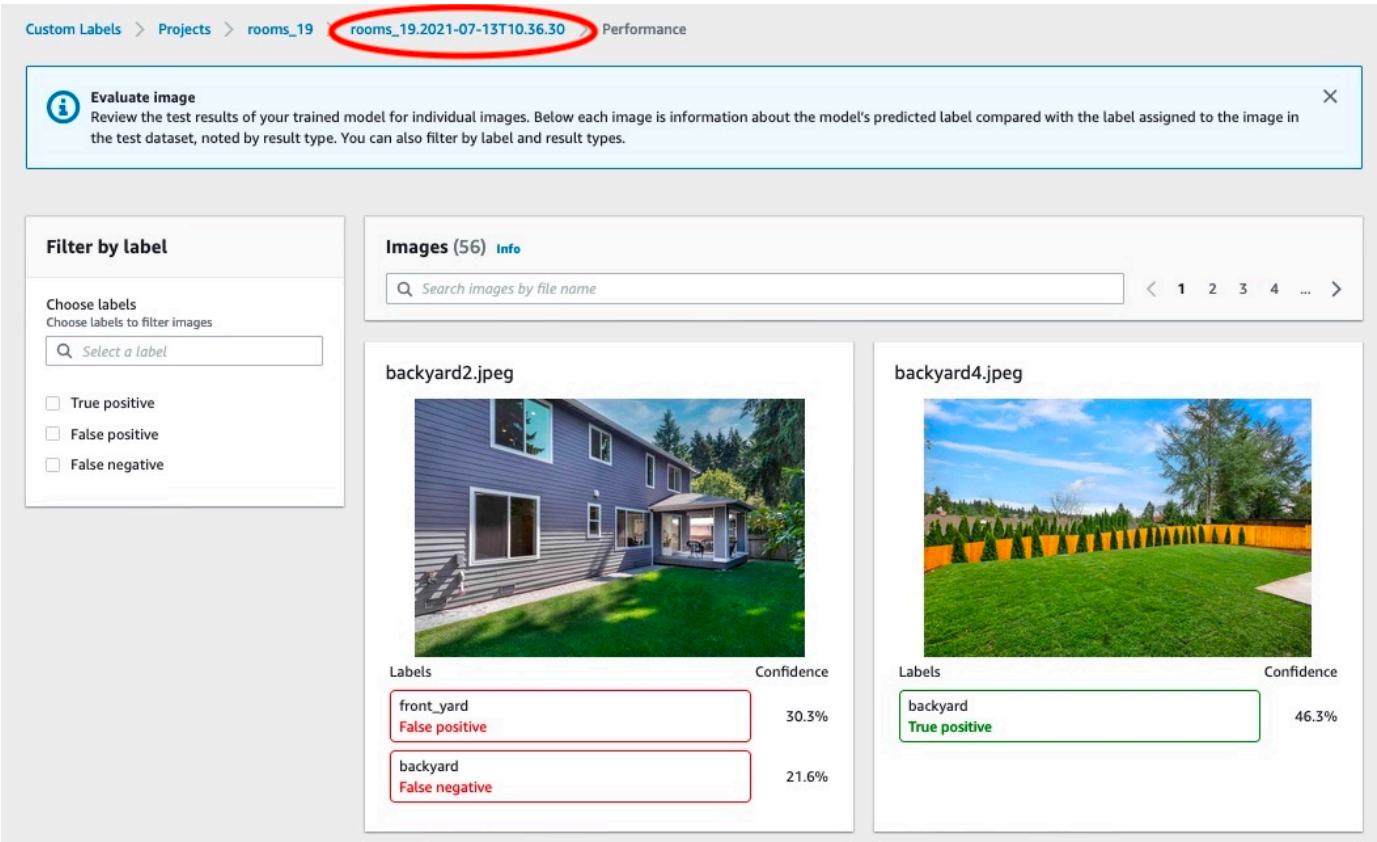
在此步骤中，您将通过控制台获取训练结果。

获取评估指标（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选择包含要评估的已训练模型的项目。
6. 在模型部分，选择要评估的模型。
7. 选择评估选项卡，以查看评估结果。有关评估模型的信息，请参阅改进经过训练的 Amazon Rekognition Custom Labels 模型。
8. 选择查看测试结果，以查看单个测试图像的结果。有关更多信息，请参阅评估模型的指标。



9. 查看测试结果后，选择项目名称返回模型页面。



10. 使用这些指标来评估模型的性能。有关更多信息，请参阅改进 Amazon Rekognition Custom Labels 模型。

获取 Amazon Rekognition Custom Labels 评估指标 (SDK)

可通过 DescribeProjectVersions 操作获取控制台中提供的指标以外的指标。

与控制台一样，通过 `DescribeProjectVersions` 可获取以下指标，这些指标作为测试结果的摘要信息以及每个标签的测试结果：

- 精度
- 召回率
- F1

系统会返回所有标签的平均阈值和单个标签的阈值。

通过 `DescribeProjectVersions` 还可获取以下用于分类和图像检测（图像上的物体位置）的指标。

- 用于图像分类的混淆矩阵。有关更多信息，请参阅查看模型的混淆矩阵。
- 用于图像检测的平均精度均值 (mAP)。
- 用于图像检测的平均召回率均值 (mAR)。

通过 DescribeProjectVersions 还可获取真正例、假正例、假负例和真负例值。有关更多信息，请参阅评估模型的指标。

F1 总分指标由 DescribeProjectVersions 直接返回。其他指标可从存储在 Amazon S3 存储桶中的摘要文件和评估清单快照文件获取。有关更多信息，请参阅获取摘要文件和评估清单快照 (SDK)。

主题

摘要文件

摘要文件包含有关整个模型的评估结果信息以及每个标签的指标。这些指标包括精度、召回率、F1 分数。此外，还提供了模型的阈值。可从 DescribeProjectVersions 返回的 EvaluationResult 对象获取摘要文件的位置。有关更多信息，请参阅参考：训练结果摘要文件。

下面是一个示例摘要文件。


```
{
  "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
    }
  ]
}
```

评估清单快照

评估清单快照包含有关测试结果的详细信息。快照包含每个预测的置信度评级。此外，还包含图像的实际分类与预测分类的比较（真正例、真负例、假正例或假负例）。

这些文件是快照，因为只包含可用于测试和训练的图像。无法验证的图像（例如格式错误的图像）不包含在清单中。可从 `DescribeProjectVersions` 返回的 `TestingDataResult` 对象获取测试快照的位置。可从 `DescribeProjectVersions` 返回的 `TrainingDataResult` 对象获取训练快照的位置。

快照采用 SageMaker Ground Truth 清单输出格式，其中添加了提供其他信息（例如检测的二进制分类结果）的字段。以下代码段显示了其他字段。

```
"rekognition-custom-labels-evaluation-details": {
    "version": 1,
    "is-true-positive": true,
    "is-true-negative": false,
    "is-false-positive": false,
    "is-false-negative": false,
    "is-present-in-ground-truth": true
    "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- version：清单快照中 `rekognition-custom-labels-evaluation-details` 字段格式的版本。
- is-true-positive... ：基于置信度分数与标签最小阈值的比较结果对预测进行的二进制分类。
- is-present-in-ground-truth：如果模型所做的预测存在于用于训练的 ground truth 信息中，则为 true，否则为 false。该值不是基于置信度分数是否超过模型计算的最小阈值确定的。
- ground-truth-labeling-jobs：清单行中用于训练的 ground truth 字段列表。

有关 SageMaker Ground Truth 清单格式的信息，请参阅输出。

下面是一个示例测试清单快照，其中显示了用于图像分类和物体检测的指标。

```
// For image classification
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
    "human-annotated": "no",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  }
}


// For object detection
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
```

```
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-training-0-metadata": {
    "job-name": "rekognition-custom-labels-training-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "yes",
    "objects": [
      {
        "confidence": 1.0
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  },
  "rekognition-custom-labels-evaluation": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
```

```
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-evaluation-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "no",
    "objects": [
      {
        "confidence": 0.95,
        "rekognition-custom-labels-evaluation-details": {
          "version": 1,
          "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
          "is-true-positive": true,
          "is-true-negative": false,
          "is-false-positive": false,
          "is-false-negative": false,
          "is-present-in-ground-truth": true
        }
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  }
}
```

获取摘要文件和评估清单快照 (SDK)

要获取训练结果，您可以调用 DescribeProjectVersions。有关示例代码，请参阅描述模型 (SDK)。

这些指标的位置将在 DescribeProjectVersions 的 ProjectVersionDescription 响应中返回。

- EvaluationResult：摘要文件的位置。
- TestingDataResult：用于测试的评估清单快照的位置。

F1 分数和摘要文件位置将在 EvaluationResult 中返回。例如：

```
"EvaluationResult": {
                "F1Score": 1.0,
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
            }
```

评估清单快照存储在您在训练模型 (SDK)中指定的 --output-config 输入参数中指定的位置。

> **Note**
>
> BillableTrainingTimeInSeconds 中会返回您需要付费的训练时长（以秒为单位）。

有关 Amazon Rekognition Custom Labels 返回的指标的信息，请参阅获取 Amazon Rekognition Custom Labels 评估指标 (SDK)。

查看模型的混淆矩阵

您可以通过混淆矩阵查看模型与模型中的其他标签混淆的标签。通过使用混淆矩阵，您可以将改进的重点放在模型上。

在模型评估期间，Amazon Rekognition Custom Labels 会使用测试图像来识别错误识别（混淆）的标签，从而创建混淆矩阵。Amazon Rekognition Custom Labels 只会为分类模型创建混淆矩阵。可以从 Amazon Rekognition Custom Labels 在模型训练期间创建的摘要文件中获取分类矩阵。无法在 Amazon Rekognition Custom Labels 控制台中查看混淆矩阵。

主题

- 使用混淆矩阵
- 获取模型的混淆矩阵

使用混淆矩阵

下表是 Rooms 图像分类示例项目的混淆矩阵。列标题是分配给测试图像的标签（ground truth 标签）。行标题是模型为测试图像预测的标签。每个单元格是对标签（行）应为 ground truth 标签（列）的预测的百分比。例如，对浴室的预测有 67% 被正确地标注为浴室。33% 的浴室被错误地标注为厨房。当预测的标签与 ground truth 标签匹配时，高性能模型具有高单元格值。可以将这些看作是从第一个预测和 ground truth 标签到最后一个预测和 ground truth 标签的对角线。如果单元格值为 0，则表示对单元格的预测标签应为单元格的 ground truth 标签的预测为 0。

> **Note**
>
> 由于模型是不确定的，您通过训练 Rooms 项目获得的混淆矩阵单元格值可能与下表不同。

混淆矩阵确定了需要关注的领域。例如，混淆矩阵显示，模型有 50% 的时间将衣柜与卧室混淆。在这种情况下，您就应该在训练数据集中添加更多衣柜和卧室的图像。此外，还要检查现有的衣柜和卧室图像是否被正确标注。这应该有助于模型更好地区分这两个标签。要向数据集中添加更多图像，请参阅向数据集中添加更多图像。

虽然混淆矩阵很有帮助，但考虑其他指标也很重要。例如，100% 的预测正确找到了 floor_plan 标签，这表明性能优异。但是，测试数据集只有 2 张带有 floor_plan 标签的图像。它还有 11 张带有 living_space 标签的图像。这种不平衡也存在于训练数据集中（13 张 living_space 图像和 2 张衣柜图像）。要获得更准确的评估，请通过添加更多代表不足的标签的图像（本示例中的平面图）来平衡训练和测试数据集。要获取每个标签的测试图像数量，请参阅获取评估指标（控制台）。

	Ground Truth 标签									
预测标签	backyard	bathroom	bedroom	closet	entry_way	floor_plan	front_yard	kitchen	living_space	patio
backyard	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%
bathroom	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
bedroom	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
closet	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%
entry_way	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%

	Ground Truth 标签									
预测标签	backyard	bathroom	bedroom	closet	entry_way	floor_plan	front_yard	kitchen	living_space	patio
floor_plan	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
front_yard	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
kitchen	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
living_space	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
patio	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

获取模型的混淆矩阵

以下代码使用 DescribeProjects 和 DescribeProjectVersions 操作获取模型的摘要文件。然后，使用摘要文件来显示模型的混淆矩阵。

显示模型的混淆矩阵 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下代码显示模型的混淆矩阵。提供以下命令行参数：

   - `project_name`：要使用的项目的名称。可以从 Amazon Rekognition Custom Labels 控制台的项目页面获取项目名称。
   - `version_name`：要使用的模型版本。可以从 Amazon Rekognition Custom Labels 控制台的项目详细信息页面获取版本名称。

```
# 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()
```

参考：训练结果摘要文件

训练结果摘要包含可用于评估模型的指标。该摘要文件还用于在控制台训练结果页面中显示指标。训练结束后，该摘要文件存储在 Amazon S3 存储桶中。要获取摘要文件，请调用 DescribeProjectVersion。有关示例代码，请参阅获取摘要文件和评估清单快照 (SDK)。

摘要文件

以下 JSON 采用摘要文件的格式。

EvaluationDetails（第 3 部分）

有关训练任务的概要信息。其中包括模型所属项目的 ARN (ProjectVersionArn)、训练结束的日期和时间、评估的模型版本 (EvaluationEndTimestamp) 以及训练期间检测到的标签列表 (Labels)。此外，还包括用于训练 (NumberOfTrainingImages) 和评估 (NumberOfTestingImages) 的图像数量。

AggregatedEvaluationResults（第 1 部分）

可以结合使用 AggregatedEvaluationResults 与测试数据集来评估经过训练的模型的整体性能。包括 Precision、Recall 和 F1Score 指标的汇总指标。对于物体检测（图像上的物体位置），将返回 AverageRecall (mAR) 和 AveragePrecision (mAP) 指标。对于分类（图像中物体的类型），将返回混淆矩阵指标。

LabelEvaluationResults（第 2 部分）

可以使用 labelEvaluationResults 来评估单个标签的性能。这些标签按各标签的 F1 分数排序。包含的指标有 Precision、Recall、F1Score 和 Threshold（用于分类）。

文件名格式如下：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"
      },
    },
    ...
  ]
```

```
}
```

改进 Amazon Rekognition Custom Labels 模型

机器学习模型的性能在很大程度上取决于诸如以下的因素：您的自定义标签（您感兴趣的特定物体和场景）的复杂性和可变性、您提供的训练数据集的质量和代表性，以及用于训练模型的模型框架和机器学习方法。

Amazon Rekognition Custom Labels 简化了这个过程，而且不要求您具备任何机器学习专业知识。但是，构建优质模型的过程通常涉及数据迭代和模型改进，以实现所需的性能。以下是有关如何改进模型的信息。

数据

通常，可以使用更多更优质的数据来提高模型的质量。使用清晰显示物体或场景且不包含不需要的杂乱物品的训练图像。对于物体周围的边界框，请使用所包含的物体完全可见且未被其他物体遮挡的训练图像。

确保您的训练和测试数据集与您最终要对其运行推理的图像的类型匹配。对于只有几个训练示例的物体（例如徽标），则应在测试图像中的徽标周围提供边界框。这些图像代表或描绘了您想要在其中定位物体的场景。

要向训练或测试数据集中添加更多图像，请参阅向数据集中添加更多图像。

减少假正例（更高的精度）

- 首先，检查提高假设阈值是否可以保持正确的预测，同时减少假正例。在某个时刻，由于给定模型的精度和召回率之间的权衡，这种做法的收益会递减。您无法为标签设置假设阈值，但可以通过为 `DetectCustomLabels` 的 `MinConfidence` 输入参数指定一个较高的值来达到相同的效果。有关更多信息，请参阅使用经过训练的模型分析图像。
- 您可能会看到一个或多个您感兴趣的自定义标签 (A) 一直与同一类物体（但不是您感兴趣的标签）(B) 混淆。为了帮助改进此问题，请将 B 作为物体类别标签添加到训练数据集（以及获得假正例的图像）。实际上，您是在通过新的训练图像帮助模型学会预测 B 而不是 A。要向训练数据集中添加图像，请参阅向数据集中添加更多图像。
- 您可能会发现模型被两个自定义标签（A 和 B）所混淆：具有标签 A 的测试图像被预测为具有标签 B，反之亦然。在这种情况下，请先检查训练集和测试集中是否有错误标注的图像。使用数据集库管理分配给数据集的标签。有关更多信息，请参阅管理标签。此外，添加更多与此类混淆相关的训练图

像将有助于重新训练的模型更好地区分 A 和 B。要向训练数据集添加图像，请参阅向数据集中添加更多图像。

减少假负例（更好的召回率）

- 使用较低的假设阈值。您无法为标签设置假设阈值，但可以通过为 DetectCustomLabels 指定一个较低的 MinConfidence 输入参数来达到相同的效果。有关更多信息，请参阅使用经过训练的模型分析图像。
- 使用更好的示例对物体及其出现的图像的多样性进行建模。
- 将标签分成两个更容易学习的类别。例如，您可能想要的不是好饼干和坏饼干，而是好饼干、烧焦的饼干和碎了的饼干，以帮助模型更好地学习每个独特的概念。

运行经过训练的 Amazon Rekognition Custom Labels 模型

当您对模型性能感到满意后，就可以开始使用它了。您可以使用控制台或 AWS SDK 来启动和停止模型。控制台还包含您可以使用的示例 SDK 操作。

主题

推理单元

启动模型时，需要指定模型使用的计算资源（称为推理单元）的数量。

> **Important**
>
> 根据您配置模型运行的方式，您需要为模型运行的小时数和模型在运行时使用的推理单元数付费。例如，如果使用两个推理单元启动模型并使用模型 8 小时，则需要支付 16 个推理小时（8 小时运行时间 x 两个推理单元）的费用。有关更多信息，请参阅推理小时。如果没有明确停止模型，则即使没有主动使用模型来分析图像，也需要付费。

单个推理单元支持的每秒事务数 (TPS) 受以下因素影响。

- 检测图像级标签（分类）的模型通常比使用边界框检测和定位物体（物体检测）的模型具有更高的 TPS。
- 模型的复杂性。
- 分辨率更高的图像需要的分析时间更多。
- 图像中包含的物体越多，需要的分析时间也越多。
- 小图像的分析速度比大图像更快。
- 以图像字节形式传递的图像与先将图像上传到 Amazon S3 存储桶，然后再引用上传的图像相比，前者的分析速度更快。以图像字节形式传递的图像必须小于 4.0 MB。在近乎实时地处理大小小于 4.0 MB 的图像时，建议使用图像字节。例如，从 IP 摄像头捕获的图像。

- 处理存储在 Amazon S3 存储桶中的图像，比下载图像、转换为图像字节，然后再传递图像字节进行分析要快。
- 分析已存储在 Amazon S3 存储桶中的图像可能比分析作为图像字节传递的相同图像要快。当图像较大时，尤其如此。

如果调用 `DetectCustomLabels` 的次数超过了模型使用的推理单元总数所支持的最大 TPS，Amazon Rekognition Custom Labels 将返回 `ProvisionedThroughputExceededException` 异常。

使用推理单元管理吞吐量

可以根据应用程序的需求提高或降低模型的吞吐量。可通过增加推理单元来提高吞吐量。每增加一个推理单元，您的处理速度就会增加一个推理单元。有关如何计算所需的推理单元数，请参阅计算 Amazon Rekognition Custom Labels 和 Amazon Lookout for Vision 模型的推理单元数量。如果要更改模型支持的吞吐量，您有两种选择：

手动添加或移除推理单元

停止模型，然后使用所需数量的推理单元重新启动模型。这种方法的缺点是，模型在重新启动时无法接收请求，也不能用于应对需求高峰。如果您的模型具有稳定的吞吐量，并且您的使用场景可以容忍 10-20 分钟的停机时间，请使用此方法。例如，您想使用每周计划对模型进行批量调用。

自动扩缩推理单元数量

如果模型必须应对需求高峰，Amazon Rekognition Custom Labels 可以自动扩缩模型使用的推理单元数量。随着需求的增加，Amazon Rekognition Custom Labels 会向模型添加额外的推理单元，并在需求减少时将其移除。

要让 Amazon Rekognition Custom Labels 自动扩缩模型的推理单元，请启动模型并使用 `MaxInferenceUnits` 参数设置模型可使用的最大推理单元数。通过设置最大推理单元数，您可以通过限制模型可使用的推理单元数量来管理模型的运行成本。如果不指定最大单元数，Amazon Rekognition Custom Labels 将不会自动扩缩模型，而只会使用启动时所用的推理单元数。有关最大推理单元数的信息，请参阅服务限额。

您也可使用 `MinInferenceUnits` 参数指定最小推理单元数。这可让您为模型指定最小吞吐量，这里的一个推理单元代表 1 小时的处理时间。

> **Note**
>
> 无法通过 Amazon Rekognition Custom Labels 控制台设置最大推理单元数，而应通过为 `StartProjectVersion` 操作指定 `MaxInferenceUnits` 输入参数来设置。

Amazon Rekognition 自定义标签提供了以下 CloudWatch 亚马逊日志指标，您可以使用这些指标来确定模型当前的自动扩展状态。

指标	描述
`DesiredInferenceUnits`	Amazon Rekognition Custom Labels 要增加或缩减到的推理单元数量。
`InServiceInferenceUnits`	模型正在使用的推理单元数量。

如果 `DesiredInferenceUnits` = `InServiceInferenceUnits`，Amazon Rekognition Custom Labels 目前不会增减推理单元的数量。

如果 `DesiredInferenceUnits` > `InServiceInferenceUnits`，Amazon Rekognition Custom Labels 会将数量增加到 `DesiredInferenceUnits` 的值。

如果 `DesiredInferenceUnits` < `InServiceInferenceUnits`，Amazon Rekognition Custom Labels 会将数量缩减到 `DesiredInferenceUnits` 的值。

有关亚马逊 Rekognition 自定义标签和筛选维度返回的指标的更多信息，请参阅 Rekognition 的指标。CloudWatch

要了解您为模型请求的最大推理单元数，请调用 `DescribeProjectsVersion` 并检查响应中的 `MaxInferenceUnits` 字段。有关代码示例，请参阅 描述模型 (SDK)。

可用区

Amazon Rekognition Custom Labels 会将推理单元分配到 AWS 区域内的多个可用区，以提高可用性。有关更多信息，请参阅可用区。为帮助保护生产模型免受可用区中断和推理单元故障的影响，请使用至少两个推理单元启动生产模型。

如果可用区中断，可用区中的所有推理单元都将无法使用，模型容量也会减少。对的调DetectCustomLabels用将重新分配到其余的推理单元中。如果此类调用不超过其余推理单元支持的

每秒事务数 (TPS)，则会成功。AWS 修复可用区后，推理单元将重新启动，模型也将恢复其全部容量。

如果单个推理单元出现故障，Amazon Rekognition Custom Labels 会自动在同一可用区内启动新的推理单元。在新推理单元启动之前，模型容量会降低。

启动 Amazon Rekognition Custom Labels 模型

您可以使用控制台或操作开始运行 Amazon Rekognition 自定义标签模型。StartProjectVersion

> **Important**
>
> 您需要为模型运行的小时数和模型在运行时使用的推理单元数付费。有关更多信息，请参见 运行经过训练的 Amazon Rekognition Custom Labels 模型。

启动模型可能需要几分钟才能完成。要检查模型的当前就绪状态，请查看项目或用途的详细信息页面DescribeProjectVersions。

模型启动后，您可以使用 DetectCustomLabels，使用模型分析图像。有关更多信息，请参见 使用经过训练的模型分析图像。控制台还提供了调用 `DetectCustomLabels` 的示例代码。

主题

- 启动 Amazon Rekognition Custom Labels 模型（控制台）
- 启动 Amazon Rekognition Custom Labels 模型 (SDK)

启动 Amazon Rekognition Custom Labels 模型（控制台）

按照以下过程通过控制台启动运行 Amazon Rekognition Custom Labels 模型。可以直接从控制台启动模型，也可使用控制台提供的 AWS SDK 代码启动模型。

启动模型（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。

5. 在项目资源页面上，选择包含要启动的已训练模型的项目。
6. 在模型部分中，选择要启动的模型。
7. 选择使用模型选项卡。
8. 执行下列操作之一：

   Start model using the console

   在启动或停止模型部分中，执行以下操作：

   1. 选择要使用的推理单元的数量。有关更多信息，请参见 运行经过训练的 Amazon Rekognition Custom Labels 模型。
   2. 选择启动。
   3. 在启动模型对话框中，选择启动。

   Start model using the AWS SDK

   在使用模型部分中，执行以下操作：

   1. 选择 API 代码。
   2. 选择 AWS CLI 或 Python。
   3. 在启动模型中，复制示例代码。
   4. 使用示例代码启动模型。有关更多信息，请参见 启动 Amazon Rekognition Custom Labels 模型 (SDK)。
9. 要返回到项目概述页面，请在页面顶部选择项目名称。
10. 在模型部分中，检查模型的状态。当模型状态为 RUNNING 时，就可以使用模型来分析图像。有关更多信息，请参见 使用经过训练的模型分析图像。

启动 Amazon Rekognition Custom Labels 模型 (SDK)

您可以通过调用 StartProjectVersionAPI 并在`ProjectVersionArn`输入参数中传递模型的 Amazon 资源名称 (ARN) 来启动模型。此外，还需要指定要使用的推理单元数量。有关更多信息，请参见 运行经过训练的 Amazon Rekognition Custom Labels 模型。

模型可能需要一段时间才能启动。本主题中的 Python 和 Java 示例使用 waiter 来等待模型启动。waiter 是一种实用程序方法，用于轮询是否发生了特定状态。或者，您可以通过致电来查看当前状态DescribeProjectVersions。

启动模型 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参见 步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码启动模型。

   CLI

   将 `project-version-arn` 的值更改为要启动的模型的 ARN。将 `--min-inference-units` 的值更改为要使用的推理单元数。（可选）将 `--max-inference-units` 更改为 Amazon Rekognition Custom Labels 可用于自动扩缩模型的最大推理单元数。

   ```
   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

   提供以下命令行参数：

   - `project_arn`：包含要启动的模型的项目的 ARN。
   - `model_arn`：要启动的模型的 ARN。
   - `min_inference_units`：要使用的推理单元数。
   - （可选）`--max_inference_units`：Amazon Rekognition Custom Labels 可用于自动扩缩模型的最大推理单元数。

   ```
   # 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

提供以下命令行参数：

- project_arn：包含要启动的模型的项目的 ARN。
- model_arn：要启动的模型的 ARN。
- min_inference_units：要使用的推理单元数。
- （可选）max_inference_units：Amazon Rekognition Custom Labels 可用于自动扩缩模型的最大推理单元数。如果不指定值，则不会进行自动扩缩。

```
/*
   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);
        }
```

```
    }
}
```

停止 Amazon Rekognition Custom Labels 模型

您可以使用控制台或使用操作来停止运行 Amazon Rekognition 自定义标签模型。StopProjectVersion

主题

- 停止 Amazon Rekognition Custom Labels 模型（控制台）
- 停止 Amazon Rekognition Custom Labels 模型 (SDK)

停止 Amazon Rekognition Custom Labels 模型（控制台）

按照以下过程通过控制台停止运行 Amazon Rekognition Custom Labels 模型。可以直接从控制台停止模型，也可使用控制台提供的 AWS SDK 代码停止模型。

停止模型（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目资源页面上，选择包含要停止的已训练模型的项目。
6. 在模型部分中，选择要停止的模型。
7. 选择使用模型选项卡。
8. Stop model using the console
    1. 在启动或停止模型部分中，选择停止。
    2. 在停止模型对话框中，输入 stop 以确认要停止模型。
    3. 选择停止以停止模型。

Stop model using the AWS SDK

在使用模型部分中，执行以下操作：

1. 选择 API 代码。
2. 选择 AWS CLI 或 Python。
3. 在停止模型中，复制示例代码。
4. 使用示例代码停止模型。有关更多信息，请参见 停止 Amazon Rekognition Custom Labels 模型 (SDK)。

9. 在页面顶部选择项目名称，返回项目概述页面。
10. 在模型部分中，检查模型的状态。当模型状态为 STOPPED 时，即表示模型已停止。

停止 Amazon Rekognition Custom Labels 模型 (SDK)

您可以通过调用 StopProjectVersionAPI 并在`ProjectVersionArn`输入参数中传递模型的 Amazon 资源名称 (ARN) 来停止模型。

模型可能需要一段时间才能停止。要检查当前状态，请使用 `DescribeProjectVersions`。

停止模型 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参见 步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码停止正在运行的模型。

CLI

将 `project-version-arn` 的值更改为要停止的模型的 ARN。

```
aws rekognition stop-project-version --project-version-arn "model arn" \
  --profile custom-labels-access
```

Python

以下示例会停止已在运行的模型。

提供以下命令行参数：

- project_arn：包含要停止的模型的项目的 ARN。
- model_arn：要停止的模型的 ARN。

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

"""
Purpose
Shows how to stop a running Amazon Lookout for Vision model.
"""

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]

    # Get the model status.
    models=rek_client.describe_project_versions(ProjectArn=project_arn,
    VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:
        logger.info("Status: %s",model['StatusMessage'])
```

```
        return model["Status"]

    # No model found.
    logger.exception("Model %s not found.", model_arn)
    raise Exception("Model %s not found.", model_arn)


def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn:  The ARN of the model (ProjectVersion) that you want to
 stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])

        # stops when hosting has stopped or failure.
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
                finished = True
                continue

            error_message = f"Error stopping model. Unexepected state: {status}"
            logger.exception(error_message)
            raise Exception(error_message)
```

```
        logger.info("finished. Status %s", status)
        return status

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


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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )

def main():

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

    try:

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

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

        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")
```

```
    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

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

Java V2

提供以下命令行参数：

- project_arn：包含要停止的模型的项目的 ARN。
- model_arn：要停止的模型的 ARN。

```
/*
   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);
        }

    }

}
```

报告运行时长和使用的推理单元

如果您在 2022 年 8 月之后训练并启动模型，则可以使用 `InServiceInferenceUnits` Amazon CloudWatch 指标来确定模型运行了多少小时以及这些时段内使用的推理单位数量。

> **Note**
>
> 如果您在一个AWS区域中只有一个模型，则还可以通过跟踪成功调入`StartprojectVersion`和调入来获取模型`StopProjectVersion`的运行时间CloudWatch。如果在该 AWS 区域中运行多个模型，则不能使用这种方法，因为这些指标不包含有关模型的信息。

或者，您可以使用跟踪AWS CloudTrail对StartProjectVersion和的调用StopProjectVersion（包括事件历史记录requestParameters字段中的模型 ARN）。CloudTrail 活动限制在 90 天内，但您可以在CloudTrail湖中存储长达 7 年的活动。

以下过程将创建以下内容的图表：

- 模型运行的小时数。
- 模型已使用的推理单元数。

您最多可以选择过去 15 个月的时间段。有关指标保留的更多信息，请参阅指标保留。

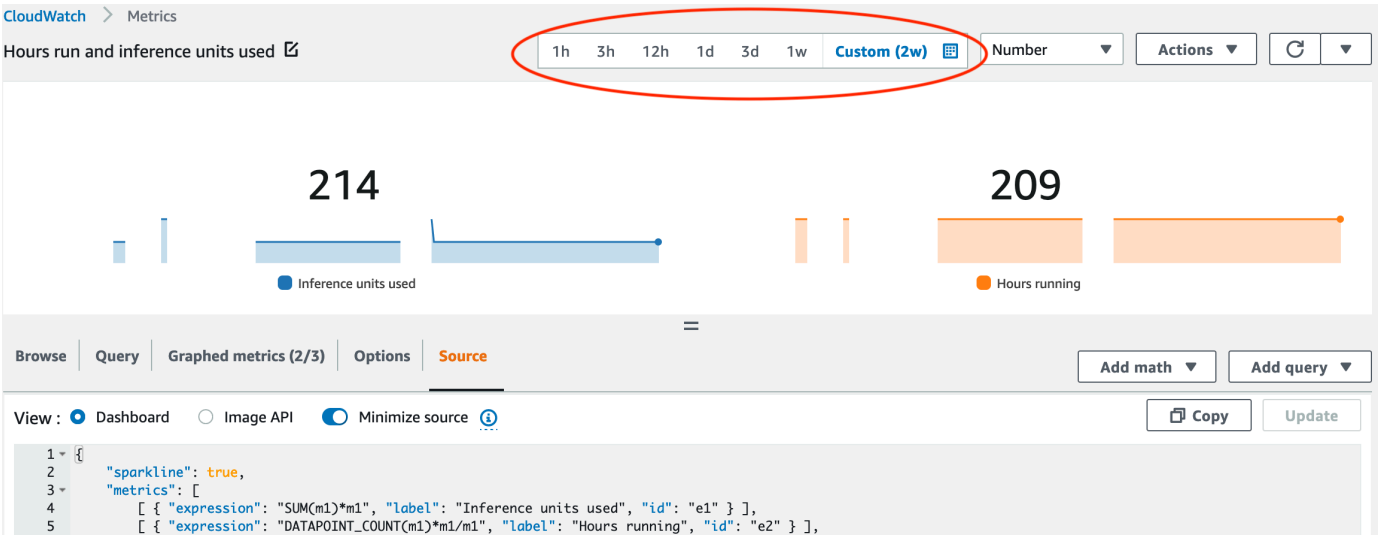
确定模型运行时间和所用推理单元

1. 登录AWS Management Console并打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中，依次选择指标、所有指标。
3. 在内容窗格中，选择来源选项卡。
4. 确保选中控制面板按钮。
5. 在编辑框中，将现有 JSON 替换为以下 JSON。更改以下值：
   - Project_Name：包含要创建图表的模型的项目。
   - Version_Name：要创建图表的模型版本。
   - AWS_Region：包含模型的 AWS 区域。通过选中页面顶部导航栏中的区域选择器，确保 CloudWatch 控制台位于同一AWS区域。根据需要更新。

```
{
    "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. 选择更新。
7. 在页面顶部，选择一个时间线。您会看到在该时间线上使用的推理单元数和运行的小时数。图表中的间隙表示模型未运行的时间。



8. （可选）依次选择操作、添加到控制面板 - 已改进，将图表添加到控制面板。

使用经过训练的模型分析图像

要使用经过训练的 Amazon Rekognition 自定义标签模型分析图片，您可以调用 API。DetectCustomLabelsDetectCustomLabels 可以预测图像中是否包含特定的物体、场景或概念。

要调用 DetectCustomLabels，请指定以下内容：

- 要使用的 Amazon Rekognition Custom Labels 模型的 Amazon 资源名称 (ARN)。
- 您希望模型进行预测的图像。您可以提供输入图像作为图像字节数组（base64 编码的图像字节）或 Amazon S3 对象。有关更多信息，请参阅图像。

自定义标签以自定义标签对象数组的形式返回。每个自定义标签代表图像中的单个物体、场景或概念。自定义标签包括：

- 在图像中找到的物体、场景或概念的标签。
- 在图像中找到的物体的边界框。边界框坐标显示物体在源图像中的位置。坐标值是占图像总体尺寸的比率。有关更多信息，请参阅BoundingBox。 DetectCustomLabels仅当模型经过训练可以检测物体位置时，才会返回边界框。
- Amazon Rekognition Custom Labels 对标签和边界框的准确性的置信度。

要根据检测置信度筛选标签，请为 MinConfidence 指定一个与所需置信度相匹配的值。例如，如果您需要对预测非常有信心，请为 MinConfidence 指定一个较高的值。要获取所有标签（无论置信度如何），请将 MinConfidence 的值指定为 0。

模型的性能部分通过模型训练期间计算的召回率和精度指标来衡量。有关更多信息，请参阅 评估模型的指标。

要提高模型的精度，请为 MinConfidence 设置更高的值。有关更多信息，请参阅 减少假正例（更高的精度）。

要提高模型的召回率，请使用较低的 MinConfidence 值。有关更多信息，请参阅 减少假负例（更好的召回率）。

如果不为 MinConfidence 指定值，Amazon Rekognition Custom Labels 将根据该标签的假设阈值返回一个标签。有关更多信息，请参阅 假设阈值。可以从模型的训练结果中获取标签的假设阈值。有关更多信息，请参阅 训练模型（控制台）。

通过使用 MinConfidence 输入参数，可为调用指定所需的阈值。检测到的置信度低于 MinConfidence 的值的标签不会在响应中返回。此外，标签的假设阈值不会影响在响应中包含该标签。

> **Note**
>
> Amazon Rekognition Custom Labels 指标将假设阈值表示为 0 到 1 之间的浮点值。MinConfidence 范围将阈值标准化为百分比值 (0-100)。来自的置信度响应 DetectCustomLabels 也会以百分比形式返回。

您可能想要为特定标签指定阈值。例如，当精度指标对于标签 A 来说是可接受的，但对于标签 B 来说却不可接受时。指定不同的阈值 (MinConfidence) 时，请考虑以下几点。

- 如果您只对单个标签 (A) 感兴趣，请将 MinConfidence 的值设置为所需的阈值。在响应中，只有当置信度大于 MinConfidence 时，才会返回标签 A 的预测（以及其他标签）。您需要筛选掉返回的所有其他标签。
- 如果要对多个标签应用不同的阈值，请执行以下操作：
  1. 为 MinConfidence 指定值 0。值 0 可确保返回所有标签（无论测检置信度如何）。
  2. 对于返回的每个标签，通过检查标签置信度是否大于您想要的标签阈值来应用所需的阈值。

有关更多信息，请参阅 改进经过训练的 Amazon Rekognition Custom Labels 模型。

如果您发现 DetectCustomLabels 返回的置信度值过低，请考虑重新训练模型。有关更多信息，请参阅 训练 Amazon Rekognition Custom Labels 模型。您可以通过指定 MaxResults 输入参数来限制 DetectCustomLabels 返回的自定义标签的数量。返回的结果按置信度由高到低的顺序排列。

有关调用 DetectCustomLabels 的其他示例，请参阅示例。

有关如何保护 DetectCustomLabels 的信息，请参阅保护 DetectCustomLabels

检测自定义标签 (API)

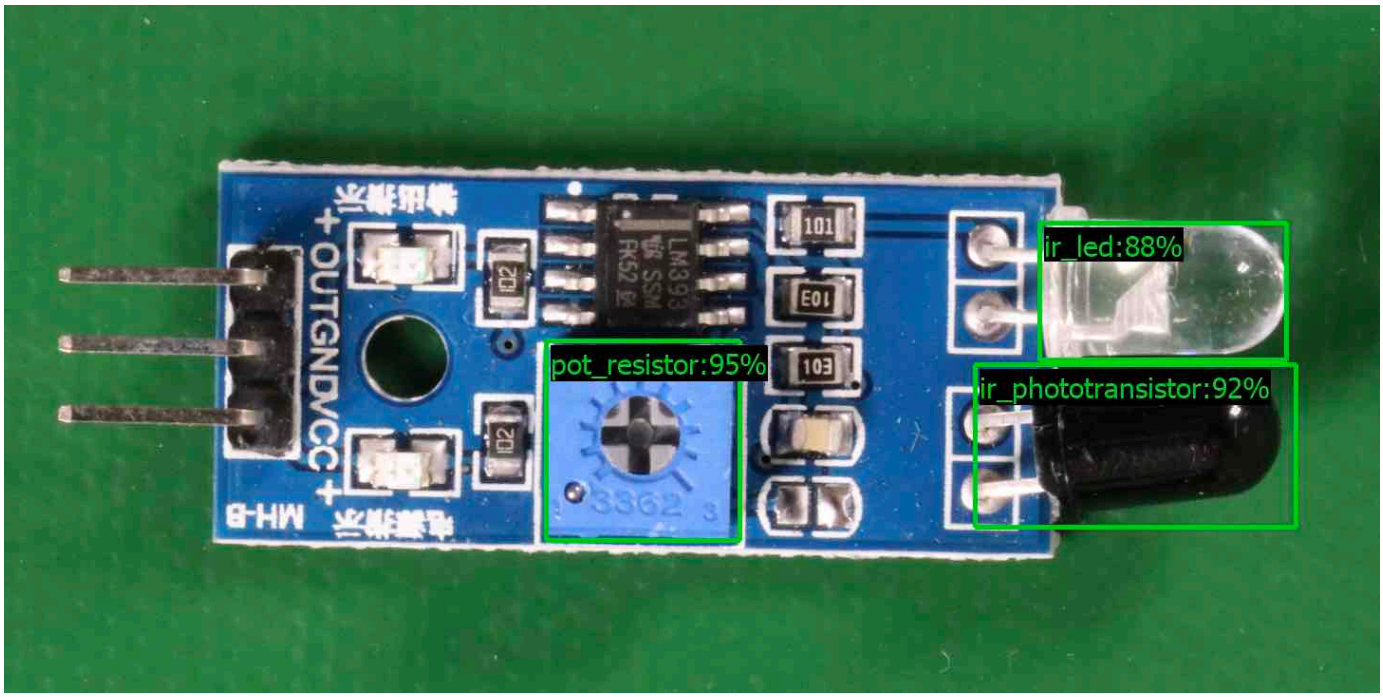
1. 如果您尚未执行以下操作，请：

   a. 确保您具有 DetectCustomLabels 和 AmazonS3ReadOnlyAccess 权限。有关更多信息，请参阅 设置 SDK 权限。

   b. 安装和配置 AWS CLI 和 AWS SDK。有关更多信息，请参阅 步骤 4：设置 AWS CLI 和 AWS SDK。

2. 训练和部署您的模型。有关更多信息，请参阅 创建 Amazon Rekognition Custom Labels 模型。
3. 确保调用 DetectCustomLabels 的用户可以访问您在步骤 2 中使用的模型。有关更多信息，请参阅 保护 DetectCustomLabels。
4. 将要分析的图像上传到 S3 存储桶。

   有关说明，请参阅《Amazon Simple Storage Service 用户指南》中的将对象上传到 Amazon S3。Python、Java 和 Java 2 示例还向您展示了如何使用本地图像文件通过原始字节传递图像。文件必须小于 4 MB。
5. 使用以下示例调用 DetectCustomLabels 操作。Python 和 Java 示例显示叠加了分析结果的图像，类似于如下图像。



AWS CLI

此 AWS CLI 命令显示 DetectCustomLabels CLI 操作的 JSON 输出。更改以下输入参数的值。

- 将 bucket 更改为在步骤 4 中使用的 Amazon S3 存储桶的名称。
- 将 image 更改为在步骤 4 中上传的输入图像文件的名称。
- 将 projectVersionArn 更改为要使用的模型的 ARN。

```
aws rekognition detect-custom-labels --project-version-arn model_arn \
```

```
    --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \
    --min-confidence 70 \
    --profile custom-labels-access
```

Python

以下示例代码显示在图像中找到的边界框和图像级标签。

要分析本地图像，请运行该程序并提供以下命令行参数：

- 要用于分析图像的模型的 ARN。
- 本地图像文件的名称和位置。

要分析存储在 Amazon S3 存储桶中的图像，请运行该程序并提供以下命令行参数：

- 要用于分析图像的模型的 ARN。
- 在步骤 4 中使用的 Amazon S3 存储桶中的图像的名称和位置。
- `--bucket #####`：在步骤 4 中使用的 Amazon S3 存储桶。

请注意，此示例假设你的 Pillow 版本为 >= 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

以下示例代码显示在图像中找到的边界框和图像级标签。

要分析本地图像，请运行该程序并提供以下命令行参数：

- 要用于分析图像的模型的 ARN。
- 本地图像文件的名称和位置。

要分析存储在 Amazon S3 存储桶中的图像，请运行该程序并提供以下命令行参数：

- 要用于分析图像的模型的 ARN。
- 在步骤 4 中使用的 Amazon S3 存储桶中的图像的名称和位置。
- 包含在步骤 4 中使用的图像的 Amazon S3 存储桶。

```
/*
   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

以下示例代码显示在图像中找到的边界框和图像级标签。

要分析本地图像，请运行该程序并提供以下命令行参数：

- projectVersionArn：要用于分析图像的模型的 ARN。
- photo：本地图像文件的名称和位置。

要分析存储在 S3 存储桶中的图像，请运行该程序并提供以下命令行参数：

- 要用于分析图像的模型的 ARN。
- 在步骤 4 中使用的 S3 存储桶中的图像的名称和位置。

- 包含在步骤 4 中使用的图像的 Amazon S3 存储桶。

```
/*
   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 操作请求

在 DetectCustomLabels 操作中，您可将输入图像作为 base64 编码的字节数组提供，也可将其作为 Amazon S3 存储桶中存储的图像提供。以下示例 JSON 请求显示从 Amazon S3 存储桶加载的图像。

```
{
    "ProjectVersionArn": "string",
     "Image":{
        "S3Object":{
            "Bucket":"string",
            "Name":"string",
            "Version":"string"
         }
    },
    "MinConfidence": 90,
    "MaxLabels": 10,
}
```

DetectCustomLabels 操作响应

来自 DetectCustomLabels 操作的以下 JSON 响应显示了在下图中检测到的自定义标签。

```
{
    "CustomLabels": [
        {
            "Name": "MyLogo",
            "Confidence": 77.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

管理 Amazon Rekognition Custom Labels 资源

本节简要介绍训练和使用 Amazon Rekognition Custom Labels 模型的工作流程，以及使用 AWS SDK 训练和使用模型的过程。

管理 Amazon Rekognition Custom Labels 项目

在 Amazon Rekognition Custom Labels 中，可以通过项目来管理为特定使用场景创建的模型。通过项目可以管理数据集、模型训练、模型版本、模型评估和项目模型的运行。

主题

- 删除 Amazon Rekognition Custom Labels 项目
- 描述项目 (SDK)
- 使用 AWS CloudFormation 创建项目

删除 Amazon Rekognition Custom Labels 项目

可以使用 Amazon Rekognition 控制台或通过调用 DeleteProject API 来删除项目。要删除项目，必须先删除所有关联的模型。已删除的项目或模型无法取消删除。

主题

- 删除 Amazon Rekognition Custom Labels 项目（控制台）
- 删除 Amazon Rekognition Custom Labels 项目 (SDK)

删除 Amazon Rekognition Custom Labels 项目（控制台）

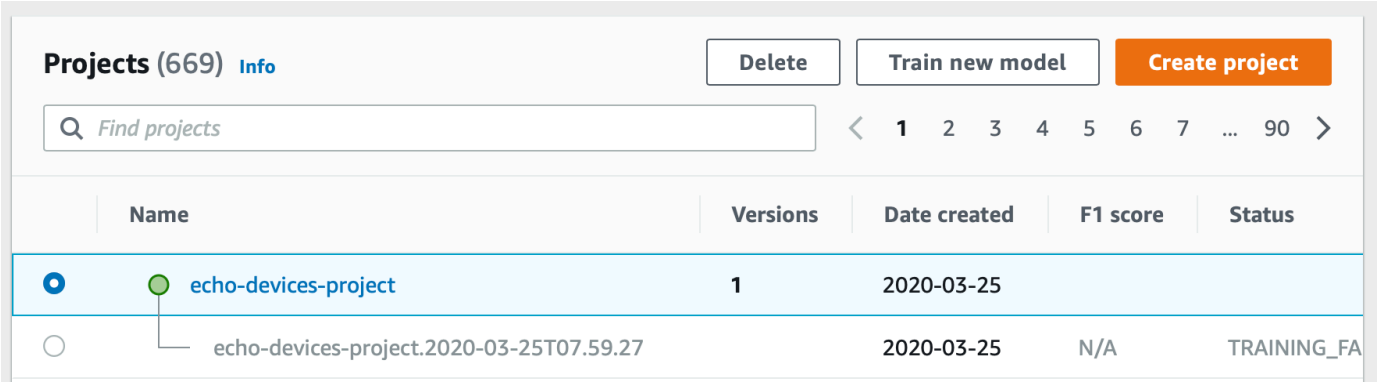
您可以从项目页面删除项目，也可以从项目的详细信息页面删除项目。以下过程介绍了如何通过项目页面删除项目。

在删除项目的过程中，Amazon Rekognition Custom Labels 控制台会为您删除关联的模型和数据集。如果项目的任何模型正在运行或训练，则无法将其删除。要停止正在运行的模型，请参阅停止 Amazon Rekognition Custom Labels 模型 (SDK)。如果模型正在训练，请等到模型训练完成后再删除项目。

删除项目（控制台）

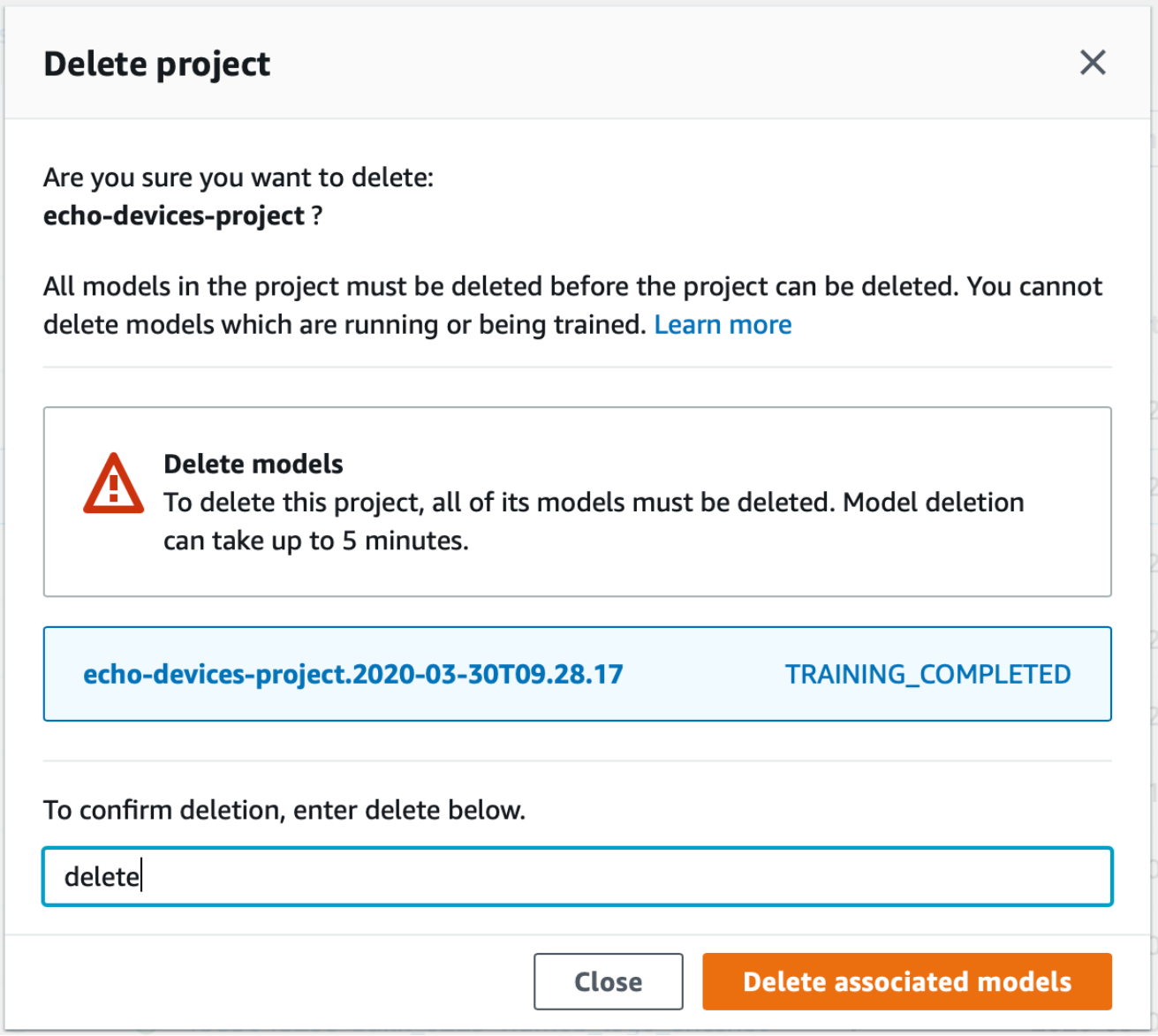
1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。

2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 在项目页面上，选中要删除的项目对应的单选按钮。



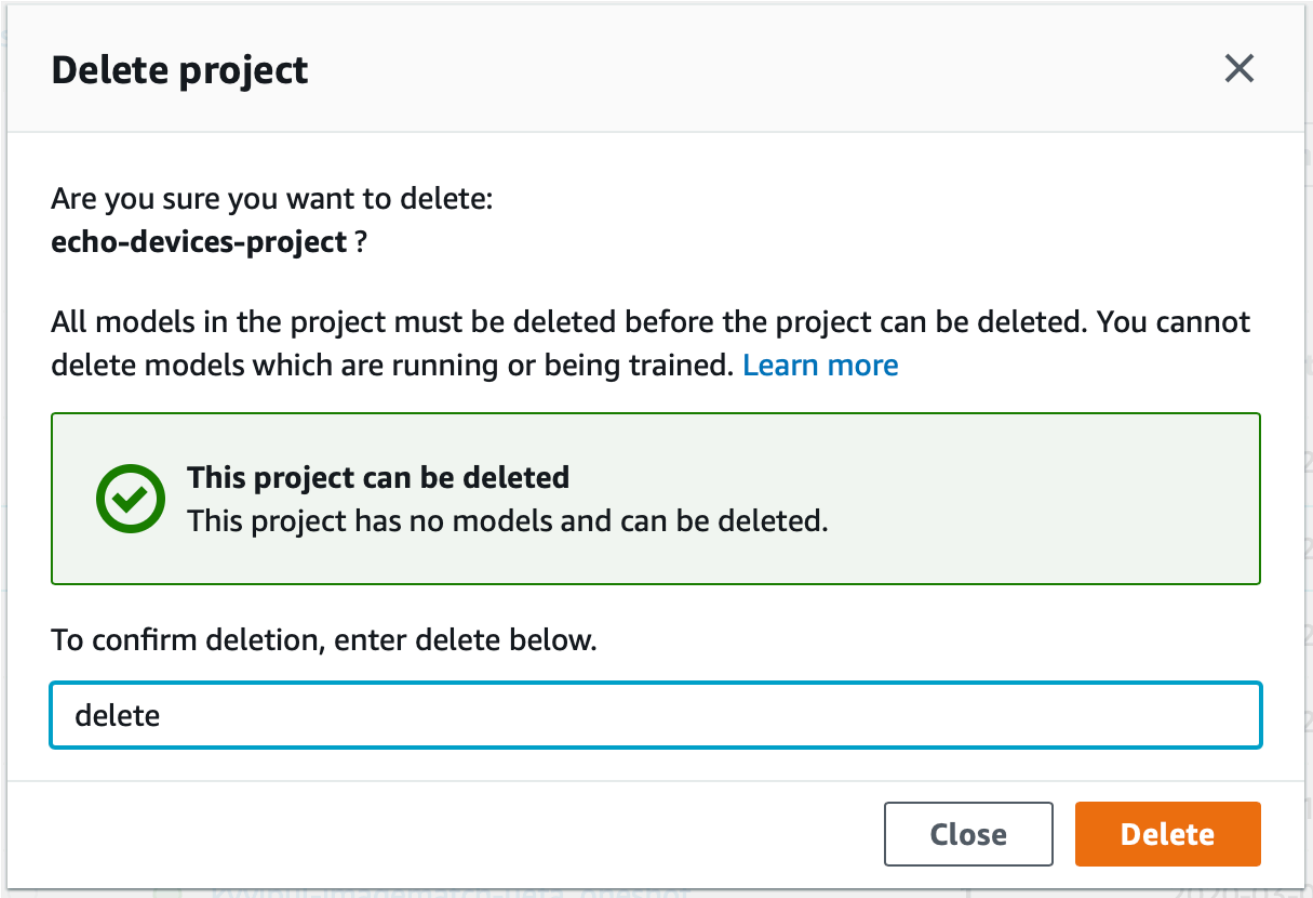
6. 在该页面的顶部，选择删除。随后会显示删除项目对话框。
7. 如果项目没有关联的模型：
   a. 输入 delete 以删除项目。
   b. 选择删除以删除项目。
8. 如果项目有关联的模型或数据集：
   a. 输入 delete 确认要删除这些模型和数据集。
   b. 根据项目是否具有数据集和/或模型，选择删除关联的模型、删除关联的数据集或删除关联的数据集和模型。模型删除可能需要一段时间才能完成。

> **Note**
>
> 控制台无法删除正在训练或运行的模型。请停止列出的所有正在运行的模型，并等待列出为正在训练的模型完成训练，然后再试一次。
> 如果在模型删除过程中关闭对话框，模型仍会被删除。稍后，您可以重复此过程来删除该项目。



c. 输入 delete 确认要删除该项目。

d. 选择删除以删除项目。



删除 Amazon Rekognition Custom Labels 项目 (SDK)

可通过调用 DeleteProject 并提供要删除的项目的 Amazon 资源名称 (ARN) 来删除 Amazon Rekognition Custom Labels 项目。要获取您 AWS 账户中的项目的 ARN，请调用 DescribeProjects。该响应包含一个 ProjectDescription 对象数组。项目 ARN 就是 `ProjectArn` 字段。可以使用项目名称来标识项目的 ARN。例如，`arn:aws:rekognition:us-east-1:123456789010:project/`*`project name`*`/1234567890123`。

在删除项目之前，必须先删除项目中的所有模型和数据集。有关更多信息，请参阅删除 Amazon Rekognition Custom Labels 模型 (SDK)和删除数据集。

项目可能需要几分钟才能删除。在此期间，项目状态为 `DELETING`。如果随后调用 DescribeProjects 的响应不包括您删除的项目，则表示该项目已被删除。

删除项目 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。

2. 使用以下代码删除项目。

AWS CLI

将 project-arn 的值更改为要删除的项目的名称。

```
aws rekognition delete-project --project-arn project_arn \
  --profile custom-labels-access
```

Python

使用以下代码。提供以下命令行参数：

- project_arn：要删除的项目的 ARN。

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

"""
Purpose
Amazon Rekognition Custom Labels project example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-
project.html
Shows how to delete an existing Amazon Rekognition Custom Labels project.
You must first delete any models and datasets that belong to the project.
"""

import argparse
import logging
import time
import boto3


from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
```

```
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)

        response = rek_client.delete_project(ProjectArn=project_arn)

        logger.info("project status: %s",response['Status'])

        deleted = False

        logger.info("waiting for project deletion: %s", project_arn)

        # Get the project name
        start = find_forward_slash(project_arn, 1) + 1
        end = find_forward_slash(project_arn, 2)
        project_name = project_arn[start:end]

        project_names = [project_name]

        while deleted is False:

            project_descriptions = rek_client.describe_projects(
                ProjectNames=project_names)['ProjectDescriptions']

            if len(project_descriptions) == 0:
                deleted = True
```

```
            else:
                time.sleep(5)

        logger.info("project deleted: %s",project_arn)

        return True

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


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

    parser.add_argument(
        "project_arn", help="The ARN of the project that you want to delete."
    )


def main():

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

    try:

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

        print(f"Deleting project: {args.project_arn}")

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

```
        delete_project(rekognition_client,
                       args.project_arn)

        print(f"Finished deleting project: {args.project_arn}")

    except ClientError as err:
        error_message = f"Problem deleting project: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用以下代码。提供以下命令行参数：

- project_arn：要删除的项目的 ARN。

```
/*
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);
        }
    }

}
```

描述项目 (SDK)

可以使用 `DescribeProjects` API 获取有关项目的信息。

描述项目 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码描述项目。将 `project_name` 替换为要描述的项目的名称。如果不指定 `--project-names`，则会返回所有项目的描述。

   AWS CLI

   ```
   aws rekognition describe-projects --project-names project_name \
     --profile custom-labels-access
   ```

   Python

   使用以下代码。提供以下命令行参数：

- project_name：要描述的项目的名称。如果不指定名称，则会返回所有项目的描述。

```
# 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

使用以下代码。提供以下命令行参数：

- project_name：要描述的项目的 ARN。如果不指定名称，则会返回所有项目的描述。

```
/*
   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);
        }

    }

}
```

使用 AWS CloudFormation 创建项目

Amazon Rekognition Custom Labels 集成了 AWS CloudFormation 服务，该服务可帮助您对 AWS 资源进行建模和设置，这样就可以减少创建和管理资源与基础设施所需的时间。您可以创建一个模板来描述所需的所有 AWS 资源，然后 AWS CloudFormation 可为您调配和配置这些资源。

您可以使用 AWS CloudFormation 调配和配置 Amazon Rekognition Custom Labels 项目。

使用 AWS CloudFormation 时，可重复使用您的模板来不断地重复设置您的 Amazon Rekognition Custom Labels 项目。只需描述一次项目，然后就可以在多个 AWS 账户和区域中反复调配相同的项目。

Amazon Rekognition Custom Labels 和 AWS CloudFormation 模板

要为 Amazon Rekognition Custom Labels 和相关的服务调配和配置项目，必须先了解 AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板可描述您要在 AWS CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 AWS CloudFormation Designer 的帮助下开始使用 AWS CloudFormation 模板。有关更多信息，请参阅 AWS CloudFormation 用户指南中的什么是 AWS CloudFormation Designer？。

有关 Amazon Rekognition Custom Labels 项目的参考信息（包括 JSON 和 YAML 模板示例），请参阅 Rekognition 资源类型参考。

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation API 参考
- AWS CloudFormation 命令行界面用户指南

管理数据集

数据集包含用于训练或测试模型的图像及分配的标签。本节中的主题介绍如何通过 Amazon Rekognition Custom Labels 控制台和 AWS SDK 管理数据集。

主题

向项目添加数据集

您可以向现有项目添加训练数据集或测试数据集。如果要替换现有数据集，请先将现有数据集删除。有关更多信息，请参阅删除数据集。然后，添加新的数据集。

主题

向项目添加数据集（控制台）

您可以使用 Amazon Rekognition Custom Labels 控制台向项目添加训练或测试数据集。

向项目添加数据集

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。
3. 在左侧导航窗格中，选择项目。随后将显示“项目”视图。
4. 选择要添加数据集的项目。
5. 在左侧导航窗格中，于项目名称下选择数据集。
6. 如果项目没有现有数据集，则会显示创建数据集页面。执行以下操作：
    a. 在创建数据集页面上，输入图像源信息。有关更多信息，请参阅the section called “使用图像创建数据集”。
    b. 选择创建数据集以创建数据集。

7. 如果项目已有数据集（训练或测试），则会显示项目详细信息页面。执行以下操作：

   a. 在项目详细信息页面上，选择操作。

   b. 如果想添加训练数据集，请选择创建训练数据集。

   c. 如果想添加训练数据集，请选择创建训练数据集。

   d. 在创建数据集页面上，输入图像源信息。有关更多信息，请参阅the section called “使用图像创建数据集”。

   e. 选择创建数据集以创建数据集。

8. 向数据集中添加图像。有关更多信息，请参阅添加更多图像（控制台）。

9. 向数据集中添加标签。有关更多信息，请参阅添加新标签（控制台）。

10. 为图像添加标签。如果要添加图像级标签，请参阅the section called “为图像分配图像级标签”。如果要添加边界框，请参阅使用边界框标注物体。有关更多信息，请参阅确定数据集用途。

向项目添加数据集 (SDK)

您可以通过以下方式向现有项目添加训练或测试数据集：

- 使用清单文件创建数据集。有关更多信息，请参阅使用 G SageMaker round Truth 清单文件 (SDK) 创建数据集。
- 创建一个空数据集，然后填充该数据集。以下示例说明了如何创建空数据集。要在创建空数据集后添加条目，请参阅向数据集中添加更多图像。

向项目添加数据集 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。

2. 使用以下示例向数据集添加 JSON 行。

   CLI

   将 `project_arn` 替换为要添加数据集的项目。将 `dataset_type` 替换为 `TRAIN` 可创建训练数据集，替换为 `TEST` 可创建测试数据集。

```
aws rekognition create-dataset --project-arn project_arn \
  --dataset-type dataset_type \
  --profile custom-labels-access
```

Python

使用以下代码创建数据集。提供以下命令行选项：

- project_arn：要添加测试数据集的项目的 ARN。
- type：要创建的数据集的类型（train 或 test）

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

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_empty_dataset(rek_client, project_arn, dataset_type):
    """
    Creates an empty Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    """

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

        dataset_type=dataset_type.upper()

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

        dataset_arn=response['DatasetArn']
```

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

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

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

            if status == "CREATE_IN_PROGRESS":

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

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

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

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

        return dataset_arn

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

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

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

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


def main():

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

    try:

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

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

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

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

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


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

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

Java V2

使用以下代码创建数据集。提供以下命令行选项：

- project_arn：要添加测试数据集的项目的 ARN。
- type：要创建的数据集的类型（train 或 test）

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

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

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

public class CreateEmptyDataset {

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

    public static String createMyEmptyDataset(RekognitionClient rekClient,
 String projectArn, String datasetType)
```

```
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating empty {0} dataset for project :
{1}",
                    new Object[] { datasetType.toString(), projectArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
                throw new Exception("Unrecognized dataset type: " +
datasetType);

            }

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
                    .datasetType(requestDatasetType).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until updates finishes

            do {

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

```
                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

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

                switch (status) {

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

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

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

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

            } while (created == false);

            return response.datasetArn();

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

```
        }

    }


    public static void main(String args[]) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n
\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the empty dataset that you want
 to create (train or test).\n\n";


        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];

        try {

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

            // Create the dataset
            datasetArn = createMyEmptyDataset(rekClient, projectArn,
 datasetType);

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

```
            rekClient.close();

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

    }

}
```

3. 向数据集中添加图像。有关更多信息，请参阅添加更多图像 (SDK)。

向数据集中添加更多图像

可以使用 Amazon Rekognition Custom Labels 控制台或通过调用 `UpdateDatasetEntries` API 向数据集中添加更多图像。

主题

- 添加更多图像（控制台）
- 添加更多图像 (SDK)

添加更多图像（控制台）

使用 Amazon Rekognition Custom Labels 控制台时，可以从本地计算机上传图像。图像将添加到存储用于创建数据集的图像的 Amazon S3 存储桶位置（控制台或外部）。

向数据集中添加更多图像（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。
3. 在左侧导航窗格中，选择项目。随后将显示“项目”视图。
4. 选择要使用的项目。
5. 在左侧导航窗格中，于项目名称下选择数据集。

6. 选择操作，然后选择要向其添加图像的数据集。
7. 选择要上传到数据集的图像。可以从本地计算机拖动或选择要上传的图像。一次最多可以上传 30 张图像。
8. 选择上传图像。
9. 选择保存更改。
10. 为图像添加标签。有关更多信息，请参阅标注图像。

添加更多图像 (SDK)

UpdateDatasetEntries 可更新清单文件中的 JSON 行或向其中添加 JSON 行。将 JSON 行作为 byte64 编码的数据对象在 GroundTruth 字段中传递。如果使用 AWS SDK 调用 UpdateDatasetEntries，该 SDK 会为您编码数据。每个 JSON 行都包含单张图像的信息，例如分配的标签或边界框信息。例如：

```
{"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"}}
```

有关更多信息，请参阅创建清单文件。

使用 source-ref 字段作为键来识别要更新的图像。如果数据集不包含匹配的 source-ref 字段值，则将该 JSON 行添加为新图像。

向数据集中添加更多图像 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例向数据集添加 JSON 行。

CLI

将 GroundTruth 的值替换为要使用的 JSON 行。您需要对 JSON 行中的任何特殊字符进行转义。

```
aws rekognition update-dataset-entries\
  --dataset-arn dataset_arn \
  --changes '{"GroundTruth" : "{\"source-ref\":\"s3://your_bucket/your_image
\",\"BB\":{\"annotations\":[{\"left\":1776,\"top\":1017,\"width\":458,\"height
\":317,\"class_id\":0},{\"left\":1797,\"top\":1334,\"width\":418,\"height
\":415,\"class_id\":1},{\"left\":2597,\"top\":1361,\"width\":655,\"height
\":329,\"class_id\":2},{\"left\":2581,\"top\":1020,\"width\":689,\"height
\":338,\"class_id\":3}],\"image_size\":[{\"width\":4000,\"height\":2667,
\"depth\":3}]},\"BB-metadata\":{\"job-name\":\"labeling-job/BB\",\"class-map
\":{\"0\":\"comparator\",\"1\":\"pot_resistor\",\"2\":\"ir_phototransistor\",
\"3\":\"ir_led\"},\"human-annotated\":\"yes\",\"objects\":[{\"confidence\":1},
{\"confidence\":1},{\"confidence\":1},{\"confidence\":1}],\"creation-date\":
\"2021-06-22T10:10:48.492Z\",\"type\":\"groundtruth/object-detection\"}}" }' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

Python

使用以下代码。提供以下命令行参数：

- dataset_arn：要更新的数据集的 ARN。
- updates_file：包含 JSON 行更新的文件。

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

"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
```

```
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that yuo want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
 updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)


        with open(updates_file) as f:
            manifest_file = f.read()


        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

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

            if status == "UPDATE_IN_PROGRESS":
```

```
                logger.info("Updating dataset: %s ", dataset_arn)
                time.sleep(5)
                continue

            if status == "UPDATE_COMPLETE":
                logger.info("Dataset updated: %s : %s : %s",
                    status, status_message, dataset_arn)
                finished=True
                continue

            if status == "UPDATE_FAILED":
                error_message = f"Dataset update failed: {status} :
 {status_message} : {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

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

        logger.info("Added entries to dataset")

        return status, status_message


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

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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to update."
    )

    parser.add_argument(
```

```
        "updates_file", help="The manifest file of JSON Lines that contains the
 updates."
    )

def main():

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

    try:

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

        print(f"Updating dataset {args.dataset_arn} with entries from
 {args.updates_file}.")

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

        status, status_message=update_dataset_entries(rekognition_client,
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")


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

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


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

Java V2

- dataset_arn：要更新的数据集的 ARN。
- update_file：包含 JSON 行更新的文件。

```
/*
   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);
        }

    }
```

```
}
```

使用现有数据集创建数据集 (SDK)

以下过程说明了如何使用 CreateDataset 操作从现有数据集创建数据集。

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码通过复制另一个数据集来创建数据集。

   AWS CLI

   使用以下代码创建数据集。替换以下内容：

   - project_arn：替换为要添加数据集的项目的 ARN。
   - dataset_type：替换为要在项目中创建的数据集的类型（TRAIN 或 TEST）。
   - dataset_arn：替换为要复制的数据集的 ARN。

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --dataset-source '{ "DatasetArn" : "dataset_arn" }' \
     --profile custom-labels-access
   ```

   Python

   以下示例使用现有的数据集创建数据集并显示其 ARN。

   要运行该程序，请提供以下命令行参数：

   - project_arn：要使用的项目的 ARN。
   - dataset_type：要创建的项目数据集的类型（train 或 test）。
   - dataset_arn：要从其创建数据集的数据集的 ARN。

   ```
   # 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

以下示例使用现有的数据集创建数据集并显示其 ARN。

要运行该程序，请提供以下命令行参数：

- `project_arn`：要使用的项目的 ARN。
- `dataset_type`：要创建的项目数据集的类型（`train` 或 `test`）。
- `dataset_arn`：要从其创建数据集的数据集的 ARN。

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

package com.example.rekognition;

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

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

public class CreateDatasetExisting {

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

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

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 dataset {2} ",
                    new Object[] { datasetType.toString(), projectArn,
 existingDatasetArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
 datasetType);
                throw new Exception("Unrecognized dataset type: " +
 datasetType);

            }

            DatasetSource datasetSource =
 DatasetSource.builder().datasetArn(existingDatasetArn).build();

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

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

```
            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until create finishes

            do {

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

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

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

                switch (status) {

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

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

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

                default:
```

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

            } while (created == false);

            return response.datasetArn();

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

    }

    public static void main(String[] args) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;
        String datasetSourceArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
                + "   dataset_arn - the ARN of the dataset that you want to copy
from.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        datasetSourceArn = args[2];
```

```
        try {

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

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

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

            rekClient.close();

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

    }

}
```

描述数据集 (SDK)

可以使用 `DescribeDataset` API 获取有关数据集的信息。

描述数据集 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码描述数据集。

AWS CLI

将 dataset-arn 的值更改为要描述的数据集的 ARN。

```
aws rekognition describe-dataset --dataset-arn dataset_arn \
  --profile custom-labels-access
```

Python

使用以下代码。提供以下命令行参数：

- dataset_arn：要描述的数据集的 ARN。

```
# 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：要描述的数据集的 ARN。

```
/*
   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);
        }
```

```
    }

}
```

列出数据集条目 (SDK)

可以使用 ListDatasetEntries API 列出数据集中每张图像的 JSON 行。有关更多信息，请参阅创建清单文件。

列出数据集条目 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码列出数据集中的条目

   AWS CLI

   将 dataset-arn 的值更改为要列出的数据集的 ARN。

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   要仅列出有错误的 JSON 行，请指定 has-errors。

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --has-errors \
     --profile custom-labels-access
   ```

   Python

   使用以下代码。提供以下命令行参数：

   - dataset_arn：要列出的数据集的 ARN。
   - show_errors_only：如果只想查看有错误的条目，请指定 true，否则指定 false。

   ```
   # 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

使用以下代码。提供以下命令行参数：

- dataset_arn：要列出的数据集的 ARN。
- show_errors_only：如果只想查看有错误的条目，请指定 `true`，否则指定 `false`。

```
/*
   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);
        }

    }

}
```

分配训练数据集 (SDK)

Amazon Rekognition Custom Labels 需要一个训练数据集和一个测试数据集来训练模型。

如果使用的是 API，可以使用 DistributeDatasetEntries API 将 20% 的训练数据集分配到一个空的测试数据集中。如果只有一个清单文件可用，则分配训练数据集会很有用。使用单个清单文件创建训练数据集。然后，创建一个空的测试数据集并使用 DistributeDatasetEntries 填充测试数据集。

> **Note**
>
> 如果使用的是 Amazon Rekognition Custom Labels 控制台并从单数据集项目开始，Amazon Rekognition Custom Labels 将在训练期间拆分（分配）训练数据集以创建测试数据集。20% 的训练数据集条目将移至测试数据集。

分配训练数据集 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 创建项目。有关更多信息，请参阅创建 Amazon Rekognition Custom Labels 项目 (SDK)。
3. 创建训练数据集。有关数据集的信息，请参阅创建训练和测试数据集。
4. 创建空测试数据集。
5. 使用以下示例代码将 20% 的训练数据集条目分配到测试数据集。可以通过调用 DescribeProjects 获取项目数据集的 Amazon 资源名称 (ARN)。有关示例代码，请参阅描述项目 (SDK)。

AWS CLI

将 training_dataset-arn 和 test_dataset_arn 的值更改为要使用的数据集的 ARN。

```
aws rekognition distribute-dataset-entries --datasets ['{"Arn":
 "training_dataset_arn"}, {"Arn": "test_dataset_arn"}'] \
  --profile custom-labels-access
```

Python

使用以下代码。提供以下命令行参数：

- training_dataset_arn：从中分配条目的训练数据集的 ARN。
- test_dataset_arn：将条目分配到的测试数据集的 ARN。

```
# 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

使用以下代码。提供以下命令行参数：

- training_dataset_arn：从中分配条目的训练数据集的 ARN。
- test_dataset_arn：将条目分配到的测试数据集的 ARN。

```
/*
   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);
        }

    }

}
```

删除数据集

可以从项目中删除训练和测试数据集。

主题

- 删除数据集（控制台）
- 删除 Amazon Rekognition Custom Labels 数据集 (SDK)

删除数据集（控制台）

可以使用以下过程删除数据集。删除后，如果项目还有剩余的数据集（训练或测试），则会显示项目详细信息页面。如果项目没有剩余的数据集，则会显示创建数据集页面。

如果删除训练数据集，则必须先为项目创建新的训练数据集，然后才能训练模型。有关更多信息，请参阅使用图像创建训练和测试数据集。

如果删除测试数据集，则无需创建新的测试数据集也能训练模型。在训练期间，将拆分训练数据集来为项目创建新的测试数据集。拆分训练数据集会减少可用于训练的图像数量。为了保持质量，建议在训练模型之前创建一个新的测试数据集。有关更多信息，请参阅向项目添加数据集。

删除数据集

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 在左侧窗格中，选择使用自定义标签。随后将显示 Amazon Rekognition Custom Labels 登录页面。
3. 在左侧导航窗格中，选择项目。随后将显示“项目”视图。
4. 选择包含要删除的数据集的项目。
5. 在左侧导航窗格中，于项目名称下选择数据集。
6. 选择操作。
7. 要删除训练数据集，请选择删除训练数据集。
8. 要删除测试数据集，请选择删除测试数据集。
9. 在删除训练或测试数据集对话框中，输入 delete 以确认要删除该数据集。
10. 选择删除训练或测试数据集，删除该数据集。

删除 Amazon Rekognition Custom Labels 数据集 (SDK)

可通过调用 DeleteDataset 并提供要删除的数据集的 Amazon 资源名称 (ARN) 来删除 Amazon Rekognition Custom Labels 数据集。要获取项目内的训练和测试数据集的 ARN，请调用 DescribeProjects。该响应包含一个 ProjectDescription 对象数组。`Datasets` 列表中包含数据集 ARN (`DatasetArn`) 和数据集类型 (`DatasetType`)。

如果删除训练数据集，则需要先为项目创建新的训练数据集，然后才能训练模型。如果删除测试数据集，则需要先创建新的测试数据集，然后才能训练模型。有关更多信息，请参阅向项目添加数据集 (SDK)。

删除数据集 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下代码删除数据集。

   AWS CLI

   将 dataset-arn 的值更改为要删除的数据集的 ARN。

```
aws rekognition delete-dataset --dataset-arn dataset-arn \
  --profile custom-labels-access
```

   Python

   使用以下代码。提供以下命令行参数：

   - dataset_arn：要删除的数据集的 ARN。

```
# 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

使用以下代码。提供以下命令行参数：

- dataset_arn：要删除的数据集的 ARN。

```
/*
   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);
        }

    }

}
```

管理 Amazon Rekognition Custom Labels 模型

Amazon Rekognition Custom Labels 模型是一种用于预测新图像中存在的物体、场景和概念的数学模型。它通过在用于训练模型的图像中查找相应图形来实现此目的。本节介绍如何训练模型、评估其性能和进行改进。此外，还会介绍如何使模型可供使用，以及如何在不再需要时将其删除。

主题

删除 Amazon Rekognition Custom Labels 模型

可以使用 Amazon Rekognition Custom Labels 控制台或 DeleteProjectVersion API 删除模型。无法删除正在运行或正在训练的模型。要停止正在运行的模型，请使用 StopProjectVersion API。有关更多信息，请参阅停止 Amazon Rekognition Custom Labels 模型 (SDK)。如果模型正在训练，请等到模型训练完成后再将其删除。

已删除的模型无法取消删除。

主题

删除 Amazon Rekognition Custom Labels 模型（控制台）

以下过程说明了如何从项目详细信息页面删除模型。也可以从模型详细信息页面删除模型。

删除模型（控制台）

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择使用自定义标签。
3. 选择开始。
4. 在左侧导航窗格中，选择项目。
5. 选择包含要删除的模型的项目。随后将打开项目详细信息页面。
6. 在模型部分中，选择要删除的模型。

> **Note**
>
> 如果无法选择模型，则表示该模型要么正在运行，要么正在训练，因此无法删除。查看状态字段，然后在停止正在运行的模型后重试，或者等到训练完成。

7. 选择删除模型，随后将显示删除模型对话框。
8. 输入 delete 以确认删除。
9. 选择删除以删除模型。删除模型可能需要一段时间才能完成。

> **Note**
>
> 如果在模型删除过程中关闭对话框，模型仍会被删除。

删除 Amazon Rekognition Custom Labels 模型 (SDK)

可通过调用 DeleteProjectVersion 并提供要删除的模型的 Amazon 资源名称 (ARN) 来删除 Amazon Rekognition Custom Labels 模型。可以从 Amazon Rekognition Custom Labels 控制台中的模型详细信息页面上的使用您的模型部分获取模型的 ARN。或者，也可以调用 DescribeProjectVersions 并提供以下信息。

- 与模型关联的项目的 ARN (`ProjectArn`)。
- 模型的版本名称 (`VersionNames`)。

模型 ARN 是 `DescribeProjectVersions` 响应中的 ProjectVersionDescription 对象中的 `ProjectVersionArn` 字段。

无法删除正在运行或正在训练的模型。要确定模型是否正在运行或训练，可以调用 DescribeProjectVersions 并检查模型的 ProjectVersionDescription 对象的 `Status` 字段。要停止正在运行的模型，请使用 StopProjectVersion API。有关更多信息，请参阅停止 Amazon Rekognition Custom Labels 模型 (SDK)。必须等到模型完成训练后才能将其删除。

删除模型 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下代码删除模型。

   AWS CLI

   将 `project-version-arn` 的值更改为要删除的项目的名称。

```
aws rekognition delete-project-version --project-version-arn model_arn \
```

```
  --profile custom-labels-access
```

Python

提供以下命令行参数：

- project_arn：包含要删除的模型的项目的 ARN。
- model_arn：要删除的模型版本的 ARN。

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

"""
Purpose
Shows how to delete an existing Amazon Rekognition Custom Labels model.
"""


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

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_model(rek_client, project_arn, model_arn):
    """
```

```
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """

    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]

        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)

        logger.info("model deleted: %s", model_arn)

        return True

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


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

```
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to delete."
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
 delete."
    )


def confirm_model_deletion(model_arn):
    """
    Confirms deletion of the model. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

    start = input("Enter delete to delete your model: ")
    if start == "delete":
        return True
    else:
        return False


def main():

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

    try:

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

        if confirm_model_deletion(args.model_arn) is True:
            print(f"Deleting model: {args.model_arn}")

            # Delete the model.
            session = boto3.Session(profile_name='custom-labels-access')
```

```
            rekognition_client = session.client("rekognition")

            delete_model(rekognition_client,
                         args.project_arn,
                         args.model_arn)

            print(f"Finished deleting model: {args.model_arn}")
        else:
            print(f"Not deleting model {args.model_arn}")

    except ClientError as err:
        print(f"Problem deleting model: {err}")


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

Java V2

- project_arn：包含要删除的模型的项目的 ARN。
- model_arn：要删除的模型版本的 ARN。

```
//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);
        }

    }

}
```

标记模型

可以使用标签识别、整理、搜索和筛选 Amazon Rekognition 模型。每个标签都是由用户定义的键和值组成的标签。例如，为了帮助确定模型的账单，可使用 `Cost center` 键标记模型，并添加相应的成本中心编号作为值。有关更多信息，请参阅标记 AWS 资源。

可以使用标签执行以下操作：

- 使用成本分配标签跟踪模型的账单。有关更多信息，请参阅使用成本分配标签。
- 可以使用 Identity and Access Management (IAM) 控制对模型的访问。有关更多信息，请参阅使用资源标签控制对 AWS 资源的访问。
- 自动管理模型。例如，可以运行自动启动或停止脚本，这些脚本可在非工作时间内关闭开发模型以降低成本。有关更多信息，请参阅运行经过训练的 Amazon Rekognition Custom Labels 模型。

可以使用 Amazon Rekognition 控制台或 AWS SDK 来标记模型。

主题

标记模型（控制台）

可以使用 Rekognition 控制台为模型添加标签、查看附加到模型的标签以及移除标签。

添加或删除标签

此过程说明了如何向现有模型添加标签或从现有模型中移除标签。也可以在训练新模型后向其添加标签。有关更多信息，请参阅训练 Amazon Rekognition Custom Labels 模型。

使用控制台向现有模型添加标签或从现有模型中移除标签

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择开始。
3. 在导航窗格中，选择项目。
4. 在项目资源页面上，选择包含要标记的模型的项目。

5. 在导航窗格中，于之前选择的项目下，选择模型。
6. 在模型部分中，选择要为其添加标签的模型。
7. 在模型的详细信息页面上，选择标签选项卡。
8. 在标签部分中，选择管理标签。
9. 在管理标签页面上，选择添加新标签。
10. 输入键和值。
    a. 对于键，输入键名称。
    b. 对于值，输入值。
11. 要添加更多标签，请重复步骤 9 和 10。
12. （可选）要移除标签，请选择要移除的标签旁的移除。如果移除的是先前保存的标签，则会在保存更改时将其移除。
13. 选择保存更改以保存您的更改。

查看模型标签

可以使用 Amazon Rekognition 控制台查看附加到模型的标签。

要查看附加到项目内所有模型的标签，必须使用 AWS SDK。有关更多信息，请参阅列出模型标签。

查看附加到模型的标签

1. 通过以下网址打开 Amazon Rekognition 控制台：https://console.aws.amazon.com/rekognition/。
2. 选择开始。
3. 在导航窗格中，选择项目。
4. 在项目资源页面上，选择包含要查看其标签的模型的项目。
5. 在导航窗格中，于之前选择的项目下，选择模型。
6. 在模型部分中，选择要查看其标签的模型。
7. 在模型的详细信息页面上，选择标签选项卡。标签部分中便会显示标签。

标记模型 (SDK)

可以使用 AWS SDK 执行以下操作：

- 向新模型添加标签

- 向现有模型添加标签
- 列出附加到模型的标签
- 从模型中移除标签

以下 AWS CLI 示例中的标签采用以下格式。

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

或者，也可以使用此格式。

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

如果尚未安装 AWS CLI，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。

向新模型添加标签

您可以在使用 CreateProjectVersion 操作创建模型时向其添加标签。在 Tags 数组输入参数中指定一个或多个标签。

```
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
```

有关创建和训练模型的信息，请参阅训练模型 (SDK)。

向现有模型添加标签

要向现有模型添加一个或多个标签，请使用 TagResource 操作。指定模型的 Amazon 资源名称 (ARN) (ResourceArn) 和要添加的标签 (Tags)。以下示例说明了如何添加两个标签。

```
aws rekognition tag-resource --resource-arn resource-arn \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

可通过调用 CreateProjectVersion 获取模型的 ARN。

列出模型标签

要列出附加到模型的标签，请使用 ListTagsForResource 操作并指定模型的 ARN (ResourceArn)。响应是附加到指定模型的标签键和值的映射。

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

输出将显示附加到模型的标签列表。

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

要查看项目中哪些模型具有特定标签，可调用 DescribeProjectVersions 获取模型列表。然后，为 DescribeProjectVersions 响应中的每个模型调用 ListTagsForResource。检查 ListTagsForResource 的响应，看是否存在所需的标签。

以下 Python 3 示例说明了如何在所有项目中搜索特定的标签键和值。输出包含在其中找到匹配键的项目 ARN 和模型 ARN。

搜索标签值

1. 将以下代码保存到名为 find_tag.py 的文件中。

```
# 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. 在命令提示符处，输入以下命令。将 *key* 和 *value* 替换为要查找的键名称和键值。

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

从模型中删除标签

要从模型中移除一个或多个标签，请使用 UntagResource 操作。指定模型的 ARN (ResourceArn) 和要移除的标签键 (Tag-Keys)。

```
aws rekognition untag-resource --resource-arn resource-arn \
  --tag-keys '["key1","key2"]' \
  --profile custom-labels-access
```

或者，也可以使用以下格式指定 tag-keys。

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

描述模型 (SDK)

可以使用 DescribeProjectVersions API 获取有关模型版本的信息。如果不指定 VersionName，DescribeProjectVersions 会返回项目中所有模型版本的描述。

描述模型 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下示例代码描述模型版本。

   AWS CLI

   将 project-arn 的值更改为要描述的项目的 ARN。将 version-name 的值更改为要描述的模型的版本。

   ```
   aws rekognition describe-project-versions --project-arn project_arn \
     --version-names version_name \
     --profile custom-labels-access
   ```

   Python

   使用以下代码。提供以下命令行参数：

   - project_arn：要描述的模型的 ARN。
   - model_version：要描述的模型的版本。

   例如：python describe_model.py *project_arn model_version*

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

   """
   Purpose
   Shows how to describe an Amazon Rekognition Custom Labels model.
   """
   import argparse
   import logging
   import boto3
   ```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_model(rek_client, project_arn, version_name):
    """
    Describes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the prject that contains the model.
    :param version_name: The version name of the model that you want to
 describe.
    """

    try:
        # Describe the model
        logger.info("Describing model: %s for project %s",
                    version_name, project_arn)

        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print(f"Created: {str(model['CreationTimestamp'])} ")
            print(f"ARN: {str(model['ProjectVersionArn'])} ")
            if 'BillableTrainingTimeInSeconds' in model:
                print(
                    f"Billing training time (minutes):
 {str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
                print(
                    f"\tSummary location: s3://{evaluation_results['Summary']
['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

            if 'ManifestSummary' in model:
                print(
                    f"Manifest summary location: s3://{model['ManifestSummary']
['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
            if 'OutputConfig' in model:
                print(
```

```
                    f"Training output location: s3://{model['OutputConfig']
['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
            if 'MinInferenceUnits' in model:
                print(
                    f"Minimum inference units:
 {str(model['MinInferenceUnits'])}")
            if 'MaxInferenceUnits' in model:
                print(
                    f"Maximum Inference units:
 {str(model['MaxInferenceUnits'])}")

            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])

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


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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
 describe."
    )


def main():

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

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
```

```
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Describing model: {args.version_name} for project
 {args.project_arn}.")

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

        describe_model(rekognition_client, args.project_arn,
                       args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
 {args.project_arn}.")

    except ClientError as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用以下代码。提供以下命令行参数：

- project_arn：要描述的模型的 ARN。
- model_version：要描述的模型的版本。

```
/*
   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);
        }

    }

}
```

复制 Amazon Rekognition Custom Labels 模型 (SDK)

可以使用 CopyProjectVersion 操作将 Amazon Rekognition Custom Labels 模型版本从源 Amazon Rekognition Custom Labels 项目复制到目标项目。目标项目可以在不同的 AWS 账户中，也可在相同的 AWS 账户中。典型场景是将经过测试的模型从开发 AWS 账户复制到生产 AWS 账户。

或者，也可以使用源数据集在目标账户中训练模型。使用 CopyProjectVersion 操作具有以下优势。

- 模型行为会保持一致。模型训练具有不确定性，不能保证使用相同数据集训练的两个模型会做出相同的预测。使用 CopyProjectVersion 复制模型有助于确保复制的模型的行为与源模型一致，并且无需重新测试模型。

- 不需要训练模型。这样可以为您节省成本，因为每次成功的模型训练都会向您收取费用。

要将模型复制到其他 AWS 账户，目标 AWS 账户中必须有 Amazon Rekognition Custom Labels 项目。有关创建项目的信息，请参阅创建项目。请务必在目标 AWS 账户中创建项目。

项目策略是一种基于资源的策略，用于为要复制的模型版本设置复制权限。当目标项目与源项目属于不同的 AWS 账户时，就需要使用项目策略。

在同一账户中复制模型版本时，则无需使用项目策略。但是，如果想更好地掌控这些资源，则可以选择对账户间项目使用项目策略。

可以通过调用 PutProjectPolicy 操作将项目策略附加到源项目。

不能使用 `CopyProjectVersion` 将模型复制到其他 AWS 区域的项目。此外，也无法使用 Amazon Rekognition Custom Labels 控制台复制模型。在这些情况下，可以使用用于训练源模型的数据集训练目标项目中的模型。有关更多信息，请参阅训练 Amazon Rekognition Custom Labels 模型。

要将模型从源项目复制到目标项目，请执行以下操作：

复制模型

1. 创建项目策略文档。
2. 将项目策略附加到源项目。
3. 使用 `CopyProjectVersion` 操作复制模型。

要从项目中移除项目策略，请调用 DeleteProjectPolicy。要获取附加到项目的项目策略列表，请调用 ListProjectPolicies。

主题

- 创建项目策略文档
- 附加项目策略 (SDK)
- 复制模型 (SDK)
- 列出项目策略 (SDK)
- 删除项目策略 (SDK)

创建项目策略文档

Rekognition Custom Labels 使用基于资源的策略（称为项目策略）来管理模型版本的复制权限。项目策略是一个 JSON 格式的文档。

项目策略用于允许或拒绝将模型版本从源项目复制到目标项目的主体权限。如果目标项目属于不同的 AWS 账户，则需要使用项目策略。如果目标项目与源项目属于同一个 AWS 账户，而您想限制对特定模型版本的访问权限，则也需要使用项目策略。例如，您可能想拒绝授予 AWS 账户中的特定 IAM 角色复制权限。

以下示例允许主体 `arn:aws:iam::111111111111:role/Admin` 复制模型版本 `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::111111111111:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

> **Note**
>
> `Action`、`Resource`、`Principal` 和 `Effect` 是项目策略文档中的必填字段。
> 唯一支持的 `action` 是 `rekognition:CopyProjectVersion`。
> `NotAction`、`NotResource` 和 `NotPrincipal` 是禁止的字段，不得出现在项目策略文档中。

如果不指定项目策略，则与源项目属于同一 AWS 账户的主体仍然可以复制模型，只要该主体具有基于身份的策略（例如 `AmazonRekognitionCustomLabelsFullAccess`）来授予其调用 `CopyProjectVersion` 的权限即可。

以下过程会创建一个项目策略文档文件，您可将其与 附加项目策略 (SDK) 中的 Python 示例结合使用。如果您使用的是 put-project-policy AWS CLI 命令，则可将项目策略作为 JSON 字符串提供。

创建项目策略文档

1. 在文本编辑器中，创建以下文档。更改以下值：

   - Effect：指定 ALLOW 可授予复制权限。指定 DENY 可拒绝复制权限。
   - Principal：更改为要允许或拒绝其对您在 Resource 中指定的模型版本进行访问的主体。例如，您可以指定其他 AWS 账户的 AWS 账户主体。我们不限制您可以使用的主体。有关更多信息，请参阅指定主体。
   - Resource：要指定其复制权限的模型版本的 Amazon 资源名称 (ARN)。如果要授予对源项目内所有模型版本的权限，请使用以下格式 arn:aws:rekognition:*region*:*account*:project/*source project*/version/*

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"ALLOW or DENY",
      "Principal":{
        "AWS":"principal"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"Model version ARN"
    }
  ]
}
```

2. 将项目策略保存到计算机中。
3. 按照附加项目策略 (SDK)中的说明将项目策略附加到源项目。

附加项目策略 (SDK)

可以通过调用 PutProjectpolicy 操作将项目策略附加到 Amazon Rekognition Custom Labels 项目。

附加多个项目策略到一个项目时，请对要添加的每个项目策略调用 PutProjectPolicy。最多可以附加五个项目策略到一个项目。如果需要附加更多项目策略，可以请求提高限制。

首次附加唯一的项目策略到项目时，请不要在 PolicyRevisionId 输入参数中指定修订版 ID。PutProjectPolicy 的响应是 Amazon Rekognition Custom Labels 为您创建的项目策略的修订版 ID。您可以使用该修订版 ID 来更新或删除项目策略的最新修订版。Amazon Rekognition Custom Labels 仅保留项目策略的最新修订版。如果尝试更新或删除项目策略的上一个修订版，会发生 InvalidPolicyRevisionIdException 错误。

要更新现有项目策略，请在 PolicyRevisionId 输入参数中指定项目策略的修订版 ID。可通过调用 ListProjectPolicies 来获取项目中项目策略的修订版 ID。

附加项目策略到源项目后，就可以将模型从源项目复制到目标项目。有关更多信息，请参阅复制模型 (SDK)。

要从项目中移除项目策略，请调用 DeleteProjectPolicy。要获取附加到项目的项目策略列表，请调用 ListProjectPolicies。

附加项目策略到项目 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 创建项目策略文档。
3. 使用以下代码将项目策略附加到包含要复制的模型版本的信任的 AWS 账户中的项目。要获取项目 ARN，请调用 DescribeProjects。要获取模型版本 ARN，请调用 DescribeProjectVersions。

   AWS CLI

   更改以下值：

   - 将 project-arn 更改为包含要复制的模型版本的信任的 AWS 账户中的源项目的 ARN。
   - 将 policy-name 更改为您选择的策略名称。
   - 将 principal 更改为要允许或拒绝其对您在 Model version ARN 中指定的模型版本进行访问的主体。
   - 将 project-version-arn 更改为要复制的模型版本的 ARN。

   如果要更新现有项目策略，请指定 policy-revision-id 参数并提供所需项目策略的修订版 ID。

```
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

使用以下代码。提供以下命令行参数：

- project_arn：要将项目策略附加到的源项目的 ARN。
- policy_name：您选择的策略名称。
- project_policy：包含项目策略文档的文件。
- policy_revision_id – (可选) 。如果要更新项目策略的现有修订版，请指定项目策略的修订版 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 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

使用以下代码。提供以下命令行参数：

- project_arn：要将项目策略附加到的源项目的 ARN。
- project_policy_name：您选择的策略名称。
- project_policy_document：包含项目策略文档的文件。
- project_policy_revision_id – (可选) 。如果要更新项目策略的现有修订版，请指定项目策略的修订版 ID。

```
/*
   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. 按照复制模型 (SDK)中的说明复制模型版本。

复制模型 (SDK)

可以使用 CopyProjectVersion API 将模型版本从源项目复制到目标项目。目标项目可以在不同的 AWS 账户中，但必须在相同的 AWS 区域中。如果目标项目属于不同的 AWS 账户（或者您想授予对 AWS 账户中的复制模型版本的特定权限），则必须将项目策略附加到源项目。有关更多信息，请参阅创建项目策略文档。CopyProjectVersion API 需要访问您的 Amazon S3 存储桶。

复制的模型包含源模型的训练结果，但不包含源数据集。

源 AWS 账户对复制到目标账户的模型没有所有权，除非您设置相应的权限。

复制模型 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 按照附加项目策略 (SDK)中的说明将项目策略附加到源项目。
3. 如果要将模型复制到其他 AWS 账户，请确保在目标 AWS 账户中有项目。
4. 使用以下代码将模型版本复制到目标项目。

   AWS CLI

   更改以下值：

   - 将 source-project-arn 更改为包含要复制的模型版本的源项目的 ARN。
   - 将 source-project-version-arn 更改为要复制的模型版本的 ARN。
   - 将 destination-project-arn 更改为要将模型复制到的目标项目的 ARN。
   - 将 version-name 更改为目标项目中模型的版本名称。

- 将 bucket 更改为要将源模型的训练结果复制到其中的 S3 存储桶。
- 将 folder 更改为要将源模型的训练结果复制到其中的 bucket 中的文件夹。
- （可选）将 kms-key-id 更改为模型的 AWS Key Management Service 密钥 ID。
- （可选）将 key 更改为您选择的标签键。
- （可选）将 value 更改为您选择的标签值。

```
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

使用以下代码。提供以下命令行参数：

- source_project_arn：源 AWS 账户中包含要复制的模型版本的源项目的 ARN。
- source_project_version-arn：源 AWS 账户中要复制的模型版本的 ARN。
- destination_project_arn：要将模型复制到的目标项目的 ARN。
- destination_version_name：目标项目中模型的版本名称。
- training_results：要将源模型版本的训练结果复制到其中的 S3 位置。
- （可选）将 kms_key_id 更改为模型的 AWS Key Management Service 密钥 ID。
- （可选）将 tag_name 更改为您选择的标签键。
- （可选）将 tag_value 更改为您选择的标签值。

```
# 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

使用以下代码。提供以下命令行参数：

- source_project_arn：源 AWS 账户中包含要复制的模型版本的源项目的 ARN。
- source_project_version-arn：源 AWS 账户中要复制的模型版本的 ARN。
- destination_project_arn：要将模型复制到的目标项目的 ARN。
- destination_version_name：目标项目中模型的版本名称。
- output_bucket：要将源模型版本的训练结果复制到其中的 S3 存储桶。
- output_folder：要将源模型版本的训练结果复制到其中的 S3 中的文件夹。

```
/*
   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);
        }

    }

}
```

列出项目策略 (SDK)

可以使用 ListProjectPolicies 操作列出附加到 Amazon Rekognition Custom Labels 项目的项目策略。

列出附加到项目的项目策略 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下代码列出项目策略。

   AWS CLI

   将 `project-arn` 更改为要列出其所附加的项目策略的项目的 Amazon 资源名称。

```
aws rekognition list-project-policies \
  --project-arn project-arn \
```

```
  --profile custom-labels-access
```

Python

使用以下代码。提供以下命令行参数：

- project_arn：要列出其所附加的项目策略的项目的 Amazon 资源名称。

例如：python list_project_policies.py *project_arn*

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

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to list the project policies in an Amazon Rekogntion Custom Labels
 project.
"""


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

logger = logging.getLogger(__name__)


def display_project_policy(project_policy):
    """
    Displays information about a Custom Labels project policy.
    :param project_policy: The project policy (ProjectPolicy)
    that you want to display information about.
    """
    print(f"Policy name: {(project_policy['PolicyName'])}")
    print(f"Project Arn: {project_policy['ProjectArn']}")
    print(f"Document: {(project_policy['PolicyDocument'])}")
    print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
```

```
    print()



def list_project_policies(rek_client, project_arn):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n--------')
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
 NextToken=pagination_token)

            for project in response['ProjectPolicies']:
                display_project_policy(project)

            if 'NextToken' in response:
                pagination_token = response['NextToken']
            else:
                finished = True

        logger.info("Finished listing project policies.")

    except ClientError as err:
        logger.exception(
            "Couldn't list policies for - %s: %s",
            project_arn,err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
```

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

    parser.add_argument(
        "project_arn",  help="The Amazon Resource Name of the project for which
 you want to list project policies."
    )


def main():

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

    try:

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

        print(f"Listing project policies in: {args.project_arn}")

        # List the project policies.

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

        list_project_policies(rekognition_client,
                              args.project_arn)

    except ClientError as err:
        print(f"Problem list project_policies: {err}")


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

Java V2

使用以下代码。提供以下命令行参数：

- project_arn：具有要列出的项目策略的项目的 ARN。

```
/*
   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);
        }

    }

}
```

删除项目策略 (SDK)

可以使用 DeleteProjectPolicy 操作从 Amazon Rekognition Custom Labels 项目中删除现有项目策略的修订版。如果要删除附加到项目的项目策略的所有修订版，请使用 ListProjectPolicies 获取附加到该项目的每个项目策略的修订版 ID。然后，为每个策略名称调用 DeletePolicy。

删除项目策略的修订版 (SDK)

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 使用以下代码删除项目策略。

   DeletePolicy 接受 ProjectARN、PolicyName 和 PolicyRevisionId。ProjectARN 和 PolicyName 对于此 API 是必需的。PolicyRevisionId 是可选的，但在进行原子更新时可以包含此参数。

   AWS CLI

   更改以下值：

   - 将 policy-name 更改为要删除的项目策略的名称。
   - 将 policy-revision-id 更改为要删除的项目策略修订版 ID。
   - 将 project-arn 更改为包含要删除的项目策略修订版的项目的 Amazon 资源名称。

   ```
   aws rekognition delete-project-policy \
       --policy-name policy-name \
       --policy-revision-id policy-revision-id \
       --project-arn project-arn \
     --profile custom-labels-access
   ```

   Python

   使用以下代码。提供以下命令行参数：

   - policy-name：要删除的项目策略的名称。
   - project-arn：包含要删除的项目策略的项目的 Amazon 资源名称。
   - policy-revision-id：要删除的项目策略的修订版 ID。

   例如：python delete_project_policy.py *policy_name project_arn policy_revision_id*

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

```
"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to delete a revision of a project policy.
"""

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

logger = logging.getLogger(__name__)


def delete_project_policy(rekognition_client, policy_name,  project_arn,
 policy_revision_id=None):
    """
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revsion ID for the project policy that you
 want to delete.
    :param project_arn: The Amazon Resource Name of the project that contains
 the project policy
    that you want to delete.
    """
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)
```

```
        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise


def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you wany to delete project policy {policy_name} ?\n",
 policy_name)

    delete = input("Enter delete to delete your project policy: ")
    if delete == "delete":
        return True
    else:
        return False


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

    parser.add_argument(
        "policy_name", help="The ARN of the project that contains the project
 policy that you want to delete."
    )

    parser.add_argument(
        "project_arn", help="The ARN of the project project policy you want to
 delete."
    )

    parser.add_argument(
        "--policy_revision_id", help="(Optional) The revision ID of the project
 policy that you want to delete.",
        required=False
    )
```

```
def main():

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

    try:

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

        if confirm_project_policy_deletion(args.policy_name) is True:
            print(f"Deleting project_policy: {args.policy_name}")

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

            # Delete the project policy.

            delete_project_policy(rekognition_client,
                                  args.policy_name,
                                  args.project_arn,
                                  args.policy_revision_id)

            print(f"Finished deleting project policy: {args.policy_name}")
        else:
            print(f"Not deleting project policy {args.policy_name}")
    except ClientError as err:
        print(f"Couldn't delete project policy in {args.policy_name}: {err}")



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

Java V2

使用以下代码。提供以下命令行参数：

- `policy-name`：要删除的项目策略的名称。

- project-arn：包含要删除的项目策略的项目的 Amazon 资源名称。
- policy-revision-id：要删除的项目策略的修订版 ID。

```
/*
   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);
        }

    }

}
```

示例

本节包含有关可用于 Amazon Rekognition Custom Labels 的示例的信息。

示例	描述
模型反馈解决方案	说明如何使用人工验证创建新的训练数据集来改进模型。
Amazon Rekognition Custom Labels 演示	演示显示 DetectCustomLabels 调用结果的用户界面。
视频分析	说明如何将 DetectCustomLabels 与从视频中提取的帧一起使用。
使用 AWS Lambda 函数分析图像	说明如何将 DetectCustomLabels 与 Lambda 函数一起使用。
从 CSV 文件创建清单文件	说明如何使用 CSV 文件创建适用于查找与整张图像关联的物体、场景和概念（分类）的清单文件。

模型反馈解决方案

借助模型反馈解决方案，您可以对模型的预测提供反馈，并通过人工验证进行改进。根据具体使用场景，可能使用仅包含少数几张图像的训练数据集也能成功。但要构建更精确的模型，可能就需要更大的带注释的训练集。使用模型反馈解决方案，您可以通过模型辅助创建更大的数据集。

要安装和配置模型反馈解决方案，请参阅模型反馈解决方案。

持续改进模型的工作流程如下：

1. 训练模型的第一个版本（可能使用一个小的训练数据集）。
2. 为模型反馈解决方案提供未注释的数据集。
3. 模型反馈解决方案使用当前模型。它会启动人工验证作业来注释新的数据集。
4. 根据人工反馈，模型反馈解决方案会生成一个清单文件，用于创建新模型。

Amazon Rekognition Custom Labels 演示

Amazon Rekognition Custom Labels 演示展示了一个用户界面，该界面使用 DetectCustomLabels API 分析来自本地计算机的图像。

该应用程序会向您显示有关您 AWS 账户中的 Amazon Rekognition Custom Labels 模型的信息。选择正在运行的模型后，便可分析来自您本地计算机的图像。如有必要，您可以启动模型。您也可以停止正在运行的模型。该应用程序显示了与其他 AWS 服务（例如 Amazon Cognito、Amazon S3 和 Amazon CloudFront）的集成。

有关更多信息，请参阅 Amazon Rekognition Custom Labels 演示。

视频分析

以下示例说明如何将 DetectCustomLabels 与从视频中提取的帧一起使用。该代码已使用 mov 和 mp4 格式的视频文件进行过测试。

将 DetectCustomLabels 与捕获的帧一起使用

1. 安装并配置 AWS CLI 和 AWS SDK（如果尚未如此）。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
2. 确保您具有 rekognition:DetectCustomLabels 和 AmazonS3ReadOnlyAccess 权限。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
3. 使用以下示例代码。将 videoFile 的值更改为视频文件名。将 projectVersionArn 的值更改为 Amazon Rekognition Custom Labels 模型的 Amazon 资源名称 (ARN)。

```
# 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()
```

使用 AWS Lambda 函数分析图像

AWS Lambda 是一项计算服务，可使您无需调配或管理服务器即可运行代码。例如，您可以分析从移动应用程序提交的图像，而不必创建服务器来托管应用程序代码。以下说明展示了如何在 Python 中创建用来调用 DetectCustomLabels 的 Lambda 函数。该函数会分析提供的图像并返回在图像中找到的标签列表。这些说明包括示例 Python 代码，该代码展示了如何使用 Amazon S3 存储桶中的图像或本地计算机提供的图像调用 Lambda 函数。

主题

步骤 1：创建 AWS Lambda 函数（控制台）

在此步骤中，您将创建一个空 AWS 函数，以及一个允许您的函数调用 `DetectCustomLabels` 操作的 IAM 执行角色。它还授予对存储用于分析的图像的 Amazon S3 存储桶的访问权限。您还可以为以下项指定环境变量：

- 您希望 Lambda 函数使用的 Amazon Rekognition Custom Labels 模型。
- 您希望模型使用的置信度限制。

随后，您可以向 Lambda 函数添加源代码，也可以选择添加一个层。

创建 AWS Lambda 函数（控制台）

1. 通过以下网址登录 AWS Management Console 并打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。有关更多信息，请参阅使用控制台创建 Lambda 函数。
3. 选择以下选项。
   - 选择从头开始编写。
   - 为函数名称输入一个值。
   - 对于运行时，请选择 Python 3.10。

4. 选择创建函数以创建 AWS Lambda 函数。
5. 在函数页面上，选择配置选项卡。
6. 在环境变量窗格中，选择编辑。
7. 添加以下环境变量。对于每个变量，请选择添加环境变量，然后输入变量键和值。

键	值
MODEL_ARN	您希望 Lambda 函数使用的模型的 Amazon 资源名称 (ARN) 可以从 Amazon Rekognition Custom Labels 控制台中的模型详细信息页面上的使用模型选项卡获取模型 ARN。
CONFIDENCE	模型的标签预测置信度的最小值 (0-100)。Lambda 函数不会返回置信度值低于此值的标签。

8. 选择保存以保存环境变量。
9. 在权限窗格的角色名称下，选择执行角色以在 IAM 控制台中打开角色。
10. 在权限选项卡中，选择添加权限，然后选择创建内联策略。
11. 选择 JSON，将现有策略替换为以下策略。

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

12. 选择下一步。
13. 在策略详细信息中，输入策略的名称，如 DetectCustomLabels-access。
14. 选择创建策略。
15. 如果供分析的图像存储在 Amazon S3 存储桶中，请重复步骤 10-14。

a. 在步骤 11 中，使用以下策略。将 *bucket/folder path* 替换为要分析的图像所在的 Amazon S3 存储桶和文件夹路径。

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

b. 在步骤 13 中，选择其他策略名称，如 S3Bucket-access。

步骤 2：（可选）创建层（控制台）

要运行此示例，您无需执行此步骤。DetectCustomLabels 操作会作为适用于 Python 的 AWS SDK (Boto3) 的一部分，包含在默认 Lambda Python 环境中。如果 Lambda 函数的其他部分需要最新的 AWS 服务更新，而这些更新不在默认 Lambda Python 环境中，请执行此步骤，以便将最新的 Boto3 SDK 版本作为一个层添加到函数中。

首先，创建包含 Boto3 SDK 的 .zip 文件归档。然后，创建一个层并将 .zip 文件归档添加到该层。有关更多信息，请参阅组合使用层与 Lambda 函数。

创建并添加层（控制台）

1. 打开命令提示符，然后输入以下命令。

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

2. 记下压缩文件的名称 (boto3-layer.zip)。您需要在本过程的步骤 6 中使用它。
3. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
4. 在导航窗格中，选择层。
5. 选择创建层。

6. 为名称和描述输入值。
7. 选择上传 .zip 文件，然后选择上传。
8. 在对话框中，选择您在本过程步骤 1 中创建的 .zip 文件归档 (boto3-layer.zip)。
9. 对于兼容的运行时，请选择 Python 3.9。
10. 选择创建以创建层。
11. 选择导航窗格菜单图标。
12. 在导航窗格中，选择函数。
13. 在资源列表中，选择您在步骤 1：创建 AWS Lambda 函数（控制台）中创建的函数。
14. 选择代码选项卡。
15. 在层区域中，选择添加层。
16. 选择自定义层。
17. 在自定义层中，选择您在步骤 6 中输入的层名称。
18. 在版本中，选择层版本，该版本应为 1。
19. 选择添加。

步骤 3：添加 Python 代码（控制台）

在此步骤中，您将使用 Lambda 控制台代码编辑器，向您的 Lambda 函数添加 Python 代码。该代码会使用 `DetectCustomLabels` 分析提供的图像并返回在图像中找到的标签列表。所提供的图像可以位于 Amazon S3 存储桶中，或者以 byte64 编码图像字节的形式提供。

添加 Python 代码（控制台）

1. 如果您不在 Lambda 控制台中，请执行以下操作：
   a. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
   b. 打开您在步骤 1：创建 AWS Lambda 函数（控制台）中创建的 Lambda 函数。
2. 选择代码选项卡。
3. 在代码源中，将 lambda_function.py 中的代码替换为以下代码：

```
# 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. 选择部署以部署您的 Lambda 函数。

步骤 4：试用您的 Lambda 函数

在此步骤中，您将使用计算机上的 Python 代码，将本地图像或 Amazon S3 存储桶中的图像传递给您的 Lambda 函数。从本地计算机传递的图像必须小于 6291456 字节。如果图像较大，请将图像上传到 Amazon S3 存储桶，然后使用该图像的 Amazon S3 路径调用脚本。有关将图像文件上传到 Amazon S3 存储桶的信息，请参阅上传对象。

确保您运行代码的 AWS 区域与创建 Lambda 函数时所在的区域相同。您可以在 Lambda 控制台的函数详细信息页面的导航栏中，查看 Lambda 函数的 AWS 区域。

如果 AWS Lambda 函数返回超时错误，请延长 Lambda 函数的超时时间。有关更多信息，请参阅配置函数超时（控制台）。

有关从您的代码调用 Lambda 函数的更多信息，请参阅调用 AWS Lambda 函数。

试用您的 Lambda 函数

1. 确保您具有 `lambda:InvokeFunction` 权限。您可以使用以下策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

   您可以从 Lambda 控制台中的函数概览，获取 Lambda 函数的 ARN。

   要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和组：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中 创建权限集 的说明进行操作。

- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。

- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

2. 安装并配置适用于 Python 的 AWS SDK。有关更多信息，请参阅步骤 4：设置 AWS CLI 和 AWS SDK。
3. 启动模型，即您在步骤 1：创建 AWS Lambda 函数（控制台）的步骤 7 中指定的模型。
4. 将以下代码保存到名为 `client.py` 的文件中。

```
# 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. 运行该代码。对于命令行参数，请提供 Lambda 函数名称和要分析的图像。您可以提供指向本地图像的路径，或存储在 Amazon S3 存储桶中的图像的 S3 路径。例如：

```
python client.py function_name s3://bucket/path/image.jpg
```

如果图像位于 Amazon S3 存储桶中，请确保该存储桶与您在步骤 1：创建 AWS Lambda 函数（控制台）的步骤 15 中指定的存储桶相同。

如果成功，则会输出在图像中找到的标签列表。如果未返回标签，请考虑降低您在步骤 1：创建 AWS Lambda 函数（控制台）的步骤 7 中设置的置信度值。

6. 如果已结束使用 Lambda 函数并且模型未被其他应用程序使用，请停止模型。下次要使用 Lambda 函数时，请记得启动模型。

安全性

您可以安全地管理项目、模型以及客户用来检测自定义标签的 `DetectCustomLabels` 操作。

有关 Amazon Rekognition 的安全保护的更多信息，请参阅 Amazon Rekognition 的安全性。

保护 Amazon Rekognition Custom Labels 项目

您可以通过指定基于身份的策略中指定的资源级权限来保护您的 Amazon Rekognition Custom Labels 项目。有关更多信息，请参阅基于身份的策略和基于资源的策略。

您可以保护的 Amazon Rekognition Custom Labels 资源包括：

资源	Amazon 资源名称格式
项目	arn:aws:rekognition:*:*:project/*project_name*/datetime
模型	arn:aws:rekognition:*:*:project/*project_name*/version/*name*/datetime

以下示例策略显示如何向身份授予执行以下操作的权限：

- 描述所有项目。
- 创建、启动、停止和使用特定模型进行推理。
- 创建项目。创建和描述特定模型。
- 拒绝创建特定项目。

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

保护 DetectCustomLabels

用于检测自定义标签的身份可能与管理 Amazon Rekognition Custom Labels 模型的身份不同。

您可以通过对身份应用策略来保护身份对 `DetectCustomLabels` 的访问。以下示例限制只能访问 `DetectCustomLabels` 和特定模型。该身份无法访问任何其他 Amazon Rekognition 操作。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectCustomLabels"


            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        }
    ]
}
```

AWS 托管式策略

我们提供 AmazonRekognitionCustomLabelsFullAccess AWS 托管式策略，可用于控制对 Amazon Rekognition Custom Labels 的访问。有关更多信息，请参阅 AWS 托管式策略：AmazonRekognitionCustomLabelsFullAccess。

Amazon Rekognition Custom Labels 中的准则和配额

以下几个部分介绍了使用 Amazon Rekognition Custom Labels 时的准则和配额。

支持的区域

有关提供了 Amazon Rekognition Custom Labels 的 AWS 区域的列表，请参阅 Amazon Web Services 一般参考中的 AWS 区域和端点。

配额

以下是 Amazon Rekognition Custom Labels 中的限制列表。有关可更改的限制的信息，请参阅 AWS 服务限制。要更改限制，请参阅创建案例。

训练

- 支持的文件格式为 PNG 和 JPEG 图像格式。
- 一个模型版本中的最大训练数据集数为 1。
- 数据集清单文件的最大大小为 1 GB。
- 每个物体、场景和概念（分类）数据集的最小唯一标签数为 2。
- 每个物体位置（检测）数据集的最小唯一标签数为 1。
- 每个清单的最大唯一标签数为 250。
- 每个标签的最小图像数为 1。
- 每个物体位置（检测）数据集的最大图像数为 250,000。

  亚太地区（孟买）和欧洲地区（伦敦）AWS 区域的最大图像数为 28,000。
- 每个物体、场景和概念（分类）数据集的最大图像数为 500,000。默认值为 250,000。要请求提高限制，请参阅创建案例。

  亚太地区（孟买）和欧洲地区（伦敦）AWS 区域的最大图像数为 28,000。您不能请求提高限制。
- 每张图像的最大标签数为 50。
- 一张图像中的最小边界框数为 0。
- 一张图像中的最大边界框数为 50。
- Amazon S3 存储桶中图像文件的最小图像尺寸为 64 像素 x 64 像素。

- Amazon S3 存储桶中图像文件的最大图像尺寸为 4096 像素 x 4096 像素。
- Amazon S3 存储桶中图像的最大文件大小为 15 MB。
- 图像的最大宽高比为 20:1。

测试

- 一个模型版本中的最大测试数据集数为 1。
- 数据集清单文件的最大大小为 1 GB。
- 每个物体、场景和概念（分类）数据集的最小唯一标签数为 2。
- 每个物体位置（检测）数据集的最小唯一标签数为 1。
- 每个数据集的最大唯一标签数为 250。
- 每个标签的最小图像数为 0。
- 每个标签的最大图像数为 1000。
- 每个物体位置（检测）数据集的最大图像数为 250,000。

  亚太地区（孟买）和欧洲地区（伦敦）AWS 区域的最大图像数为 7,000。
- 每个物体、场景和概念（分类）数据集的最大图像数为 500,000。默认值为 250,000。要请求提高限制，请参阅创建案例。

  亚太地区（孟买）和欧洲地区（伦敦）AWS 区域的最大图像数为 7,000。您不能请求提高限制。
- 每个清单中每张图像的最小标签数为 0。
- 每个清单中每张图像的最大标签数为 50。
- 每个清单中每张图像中的最小边界框数为 0。
- 每个清单中每张图像中的最大边界框数为 50。
- Amazon S3 存储桶中图像文件的最小图像尺寸为 64 像素 x 64 像素。
- Amazon S3 存储桶中图像文件的最大图像尺寸为 4096 像素 x 4096 像素。
- Amazon S3 存储桶中图像的最大文件大小为 15 MB。
- 支持的文件格式为 PNG 和 JPEG 图像格式。
- 图像的最大宽高比为 20:1。

检测

- 以原始字节形式传递的图像的最大大小为 4 MB。

- Amazon S3 存储桶中图像的最大文件大小为 15 MB。
- 输入图像文件（存储在 Amazon S3 存储桶中或以图像字节的形式提供）的最小图像尺寸为 64 像素 x 64 像素。
- 输入图像文件（存储在 Amazon S3 存储桶中或以图像字节的形式提供）的最大图像尺寸为 4096 像素 x 4096 像素。
- 支持的文件格式为 PNG 和 JPEG 图像格式。
- 图像的最大宽高比为 20:1。

模型复制

- 可为一个项目附加的项目策略的最大数量为 5。
- 一个目标中的最大并发复制作业数量为 5。

Amazon Rekognition Custom Labels API 参考

Amazon Rekognition Custom Labels API 的参考信息作为 Amazon Rekognition API 参考内容的一部分记录在本文档中。这是 Amazon Rekognition Custom Labels API 操作的列表，其中包含了相应 Amazon Rekognition API 参考主题的链接。此外，本文档中的 API 参考链接可转至相应的 Amazon Rekognition 开发者指南 API 参考主题。有关使用该 API 的信息，请参阅

本节简要介绍通过控制台和 AWS SDK 训练和使用 Amazon Rekognition Custom Labels 模型的工作流程。

> **Note**
>
> Amazon Rekognition Custom Labels 现在可以管理项目内的数据集。您可以使用控制台和 AWS SDK 为项目创建数据集。如果之前使用过 Amazon Rekognition Custom Labels，则可能需要将旧数据集与新项目关联。有关更多信息，请参阅步骤 6：（可选）将先前的数据集与新项目关联。

主题

确定您的模型类型

您首先需要决定要训练哪种类型的模型，这取决于您的业务目标。例如，您可以训练模型在社交媒体文章中查找您的徽标、在商店货架上识别您的产品，或者在装配线上对机器部件进行分类。

Amazon Rekognition Custom Labels 可以训练以下类型的模型：

- 查找物体、场景和概念
- 查找物体位置

- 查找品牌位置

为帮助您决定要训练的模型类型，Amazon Rekognition Custom Labels 提供了您可以使用的示例项目。有关更多信息，请参阅Amazon Rekognition Custom Labels 入门。

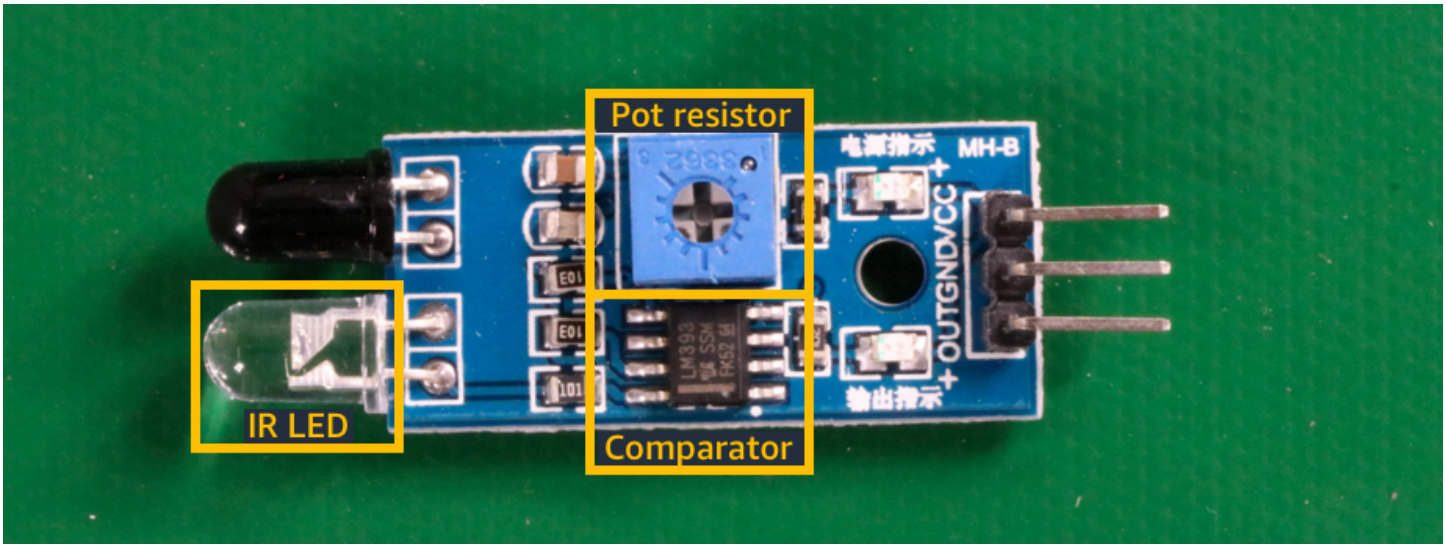
查找物体、场景和概念

该模型可以预测与整张图像关联的物体、场景和概念的分类。例如，可以训练一个模型来确定图像是否包含旅游景点。如需查看示例项目，请参阅图像分类。

或者，也可以训练模型来将图像分为多个类别。例如，上面这张图像可能包含天空颜色、反射或湖泊等类别。如需查看示例项目，请参阅多标签图像分类。

查找物体位置

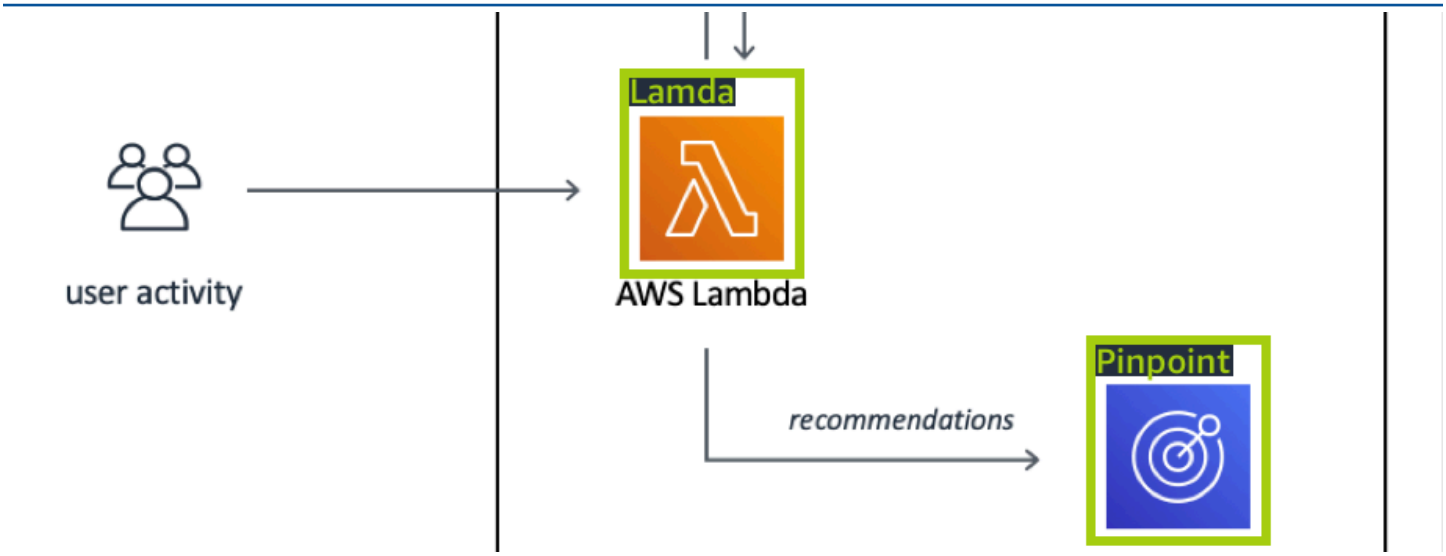
该模型可预测物体在图像上的位置。预测包括物体位置的边界框信息以及用于标识边界框内物体的标签。例如，下图显示了电路板各个零件（例如 comparator 或 pot resistor）周围的边界框。



物体定位示例项目展示了 Amazon Rekognition Custom Labels 如何使用带标签的边界框来训练查找物体位置的模型。

查找品牌位置

Amazon Rekognition Custom Labels 可以训练模型在图像上查找品牌（例如徽标）的位置。预测包括品牌位置的边界框信息以及用于标识边界框内物体的标签。如需查看示例项目，请参阅品牌检测。



创建模型

创建模型的步骤包括创建项目、创建训练和测试数据集以及训练模型。

创建项目

Amazon Rekognition Custom Labels 项目是指创建和管理模型所需的一组资源。项目用于管理以下内容：

- 数据集：用于训练模型的图像和图像标签。一个项目具有一个训练数据集和一个测试数据集。
- 模型：训练来查找符合您业务需求的独特概念、场景和物体的软件。一个项目可以有多个版本的模型。

建议将一个项目用于单个使用场景，例如在电路板上查找电路板零件。

可以使用 Amazon Rekognition Custom Labels 控制台和 CreateProject API 创建项目。有关更多信息，请参阅创建项目。

创建训练和测试数据集

数据集是由图像和描述这些图像的标签组成的集合。在您的项目中，您可以创建一个训练数据集和一个测试数据集，Amazon Rekognition Custom Labels 将使用这些数据集来训练和测试您的模型。

标签用于标识图像中的物体、场景、概念或物体周围的边界框。标签要么分配给整个图像（图像级），要么分配给围绕图像上的物体的边界框。

> **Important**
>
> 如何为数据集中的图像添加标签决定了 Amazon Rekognition Custom Labels 创建的模型的类型。例如，要训练查找物体、场景和概念的模型，请为训练和测试数据集中的图像分配图像级标签。有关更多信息，请参阅确定数据集用途。

图像必须采用 PNG 和 JPEG 格式，并且您应遵循输入图像的建议。有关更多信息，请参阅准备图像。

创建训练和测试数据集（控制台）

可以使用单个数据集或单独的训练数据集和测试数据集开始项目。如果从单个数据集开始，Amazon Rekognition Custom Labels 会在训练期间拆分该数据集，来为项目创建训练数据集 (80%) 和测试数据集 (20%)。如果想让 Amazon Rekognition Custom Labels 决定使用哪些图像进行训练和哪些图像进行测试，请使用单个数据集开始项目。为了能够完全控制训练、测试和性能调整，建议您使用单独的训练数据集和测试数据集开始您的项目。

要为项目创建数据集，请通过以下方式之一导入图像：

- 从本地计算机导入图像。

- 从 S3 存储桶导入图像。Amazon Rekognition Custom Labels 可以使用包含图像的文件夹名称为图像添加标签。
- 导入 Amazon SageMaker Ground Truth 清单文件。
- 复制现有的 Amazon Rekognition Custom Labels 数据集。

有关更多信息，请参阅使用图像创建训练和测试数据集。

根据导入图像的方式，您的图像可能没有标签。例如，从本地计算机导入的图像就没有标签。从 Amazon SageMaker Ground Truth 清单文件导入的图像则带有标签。您可以使用 Amazon Rekognition Custom Labels 控制台添加、更改和分配标签。有关更多信息，请参阅标注图像。

要使用控制台创建训练和测试数据集，请参阅使用图像创建训练和测试数据集。如需查看包含创建训练和测试数据集相关内容的教程，请参阅教程：对图像进行分类。

创建训练和测试数据集 (SDK)

要创建训练和测试数据集，请使用 `CreateDataset` API。您可以使用 Amazon SageMaker 格式的清单文件或通过复制现有的 Amazon Rekognition Custom Labels 数据集来创建数据集。有关更多信息，请参阅创建训练和测试数据集 (SDK) 。如有必要，您可以创建自己的清单文件。有关更多信息，请参阅the section called “创建清单文件”。

训练模型

使用训练数据集训练您的模型。每次训练模型时都会创建一个新的模型版本。训练期间，Amazon Rekognition Custom Labels 会测试训练后的模型的性能。您可以使用测试结果来评估和改进模型。训练需要一段时间才能完成。您只需为成功的模型训练付费。有关更多信息，请参阅训练 Amazon Rekognition Custom Labels 模型。如果模型训练失败，Amazon Rekognition Custom Labels 会提供调试信息供您使用。有关更多信息，请参阅调试失败的模型训练。

训练模型（控制台）

要使用控制台训练模型，请参阅训练模型（控制台）。

训练模型 (SDK)

可通过调用 CreateProjectVersion 来训练 Amazon Rekognition Custom Labels 模型。有关更多信息，请参阅训练模型 (SDK)。

改进模型

测试期间，Amazon Rekognition Custom Labels 会创建评估指标，您可以使用这些指标来改进训练后的模型。

评估模型

使用在测试期间创建的性能指标，评估模型的性能。F1、精度和召回率等性能指标可让您了解训练后的模型的性能，并决定是否已准备好在生产中使用它。有关更多信息，请参阅评估模型的指标。

评估模型（控制台）

如需查看性能指标，请参阅获取评估指标（控制台）。

评估模型 (SDK)

要获得性能指标，可调用 DescribeProjectVersions 来获取测试结果。有关更多信息，请参阅获取 Amazon Rekognition Custom Labels 评估指标 (SDK)。测试结果包含控制台中未提供的指标，例如分类结果的混淆矩阵。测试结果以下列格式返回：

- F1 分数：代表模型的精度和召回率整体表现的单个值。有关更多信息，请参阅F1。

- 摘要文件位置：测试摘要包含整个测试数据集的综合评估指标和每个标签的指标。`DescribeProjectVersions` 会返回摘要文件所在的 S3 存储桶和文件夹位置。有关更多信息，请参阅摘要文件。
- 评估清单快照位置：快照包含有关测试结果的详细信息，包括置信度评分和二进制分类测试的结果，例如假正例。`DescribeProjectVersions` 会返回快照文件所在的 S3 存储桶和文件夹位置。有关更多信息，请参阅评估清单快照。

改进模型

如需进行改进，可以添加更多训练图像或改进数据集标注方式。有关更多信息，请参阅改进 Amazon Rekognition Custom Labels 模型。您还可以就模型所做的预测提供反馈，并据其改进模型。有关更多信息，请参阅模型反馈解决方案。

改进模型（控制台）

要向数据集中添加图像，请参阅向数据集中添加更多图像。要添加或更改标签，请参阅the section called “标注图像”。

要重新训练模型，请参阅训练模型（控制台）。

改进模型 (SDK)

要向数据集中添加图像或更改图像的标注方式，请使用 UpdateDatasetEntries API。UpdateDatasetEntries 会在清单文件中更新或添加 JSON 行。每个 JSON 行都包含单张图像的信息，例如分配的标签或边界框信息。有关更多信息，请参阅添加更多图像 (SDK)。要查看数据集中的条目，请使用 ListDatasetEntries API。

要重新训练模型，请参阅训练模型 (SDK)。

启动模型

在使用模型之前，需要先使用 Amazon Rekognition Custom Labels 控制台或 StartProjectVersion API 启动模型。您将根据模型运行时间付费。有关更多信息，请参阅运行经过训练的模型。

启动模型（控制台）

要使用控制台启动模型，请参阅启动 Amazon Rekognition Custom Labels 模型（控制台）。

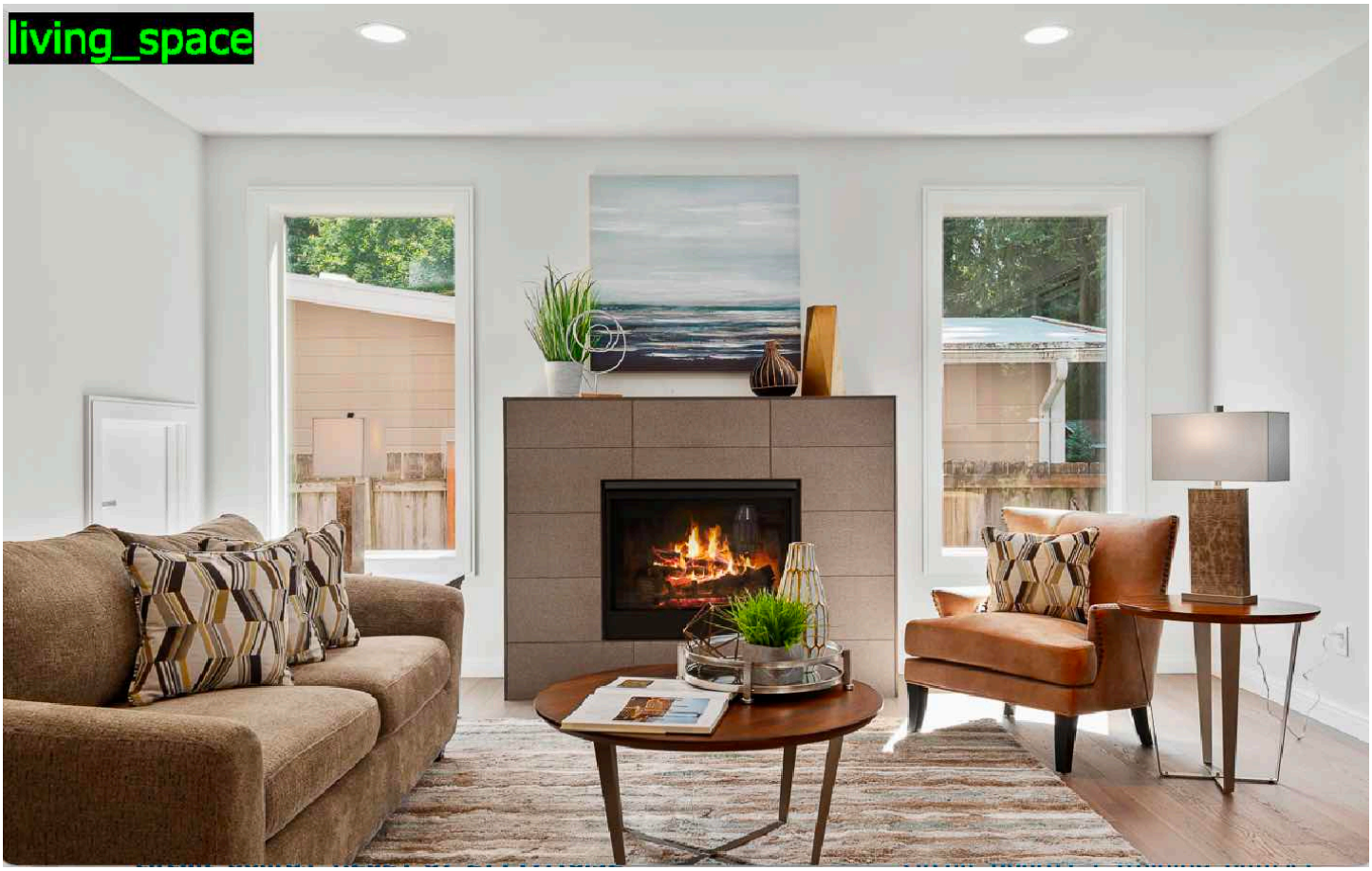
启动模型

您可以通过调用 StartProjectVersion 来启动模型。有关更多信息，请参阅启动 Amazon Rekognition Custom Labels 模型 (SDK)。

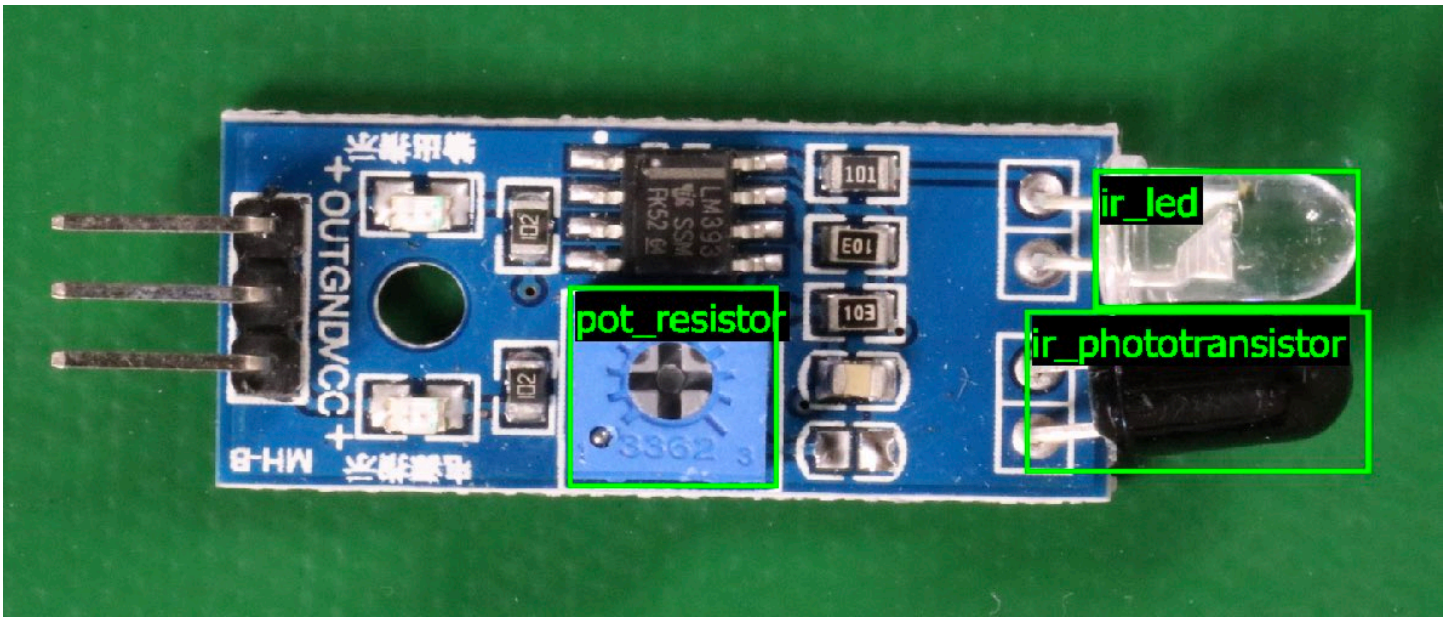
分析图像

要使用模型分析图像，可以使用 DetectCustomLabels API。可以指定本地图像或存储在 S3 存储桶中的图像。该操作还要求提供要使用的模型的 Amazon 资源名称 (ARN)。

如果模型用于查找物体、场景和概念，则响应会包含在图像中找到的图像级标签的列表。例如，下图显示了使用房间示例项目找到的图像级标签。



如果模型用于查找物体位置，则响应将包含在图像中找到的带标签的边界框列表。边界框表示物体在图像上的位置。您可以使用边界框信息在物体周围绘制边界框。例如，下图显示了使用电路板示例项目找到的电路板零件周围的边界框。



有关更多信息，请参阅使用经过训练的模型分析图像。

停止模型

您将根据模型运行时间付费。如果您不再使用模型，请使用 Amazon Rekognition Custom Labels 控制台或 `StopProjectVersion` API 停止该模型。有关更多信息，请参阅停止 Amazon Rekognition Custom Labels 模型。

停止模型（控制台）

要使用控制台停止正在运行的模型，请参阅停止 Amazon Rekognition Custom Labels 模型（控制台）。

停止模型 (SDK)

要停止正在运行的模型，请调用 StopProjectVersion。有关更多信息，请参阅停止 Amazon Rekognition Custom Labels 模型 (SDK)。

。

训练模型

项目

- CreateProject：创建 Amazon Rekognition Custom Labels 项目，该项目是资源（图像、标签、模型）和操作（训练、评估和检测）的逻辑组合。
- DeleteProject：删除 Amazon Rekognition Custom Labels 项目。
- DescribeProjects：返回所有 Amazon Rekognition Custom Labels 项目的列表。

项目策略

- PutProjectPolicy：将项目策略附加到信任的 AWS 账户中的 Amazon Rekognition Custom Labels 项目。
- ListProjectPolicies：返回附加到项目的项目策略的列表。
- DeleteProjectPolicy：删除现有的项目策略。

数据集

- CreateDataset：创建 Amazon Rekognition Custom Labels 数据集。
- DeleteDataset：删除 Amazon Rekognition Custom Labels 数据集。
- DescribeDataset：描述 Amazon Rekognition Custom Labels 数据集。
- DistributeDatasetEntries：跨项目的训练数据集和测试数据集分配训练数据集中的条目（图像）。
- ListDatasetEntries：返回 Amazon Rekognition Custom Labels 数据集中的条目（图像）的列表。
- ListDatasetLabels：返回分配给 Amazon Rekognition Custom Labels 数据集的标签列表。
- UpdateDatasetEntries：添加或更新 Amazon Rekognition Custom Labels 数据集中的条目（图像）。

模型

- CreateProjectVersion：训练 Amazon Rekognition Custom Labels 模型。
- CopyProjectVersion：复制 Amazon Rekognition Custom Labels 模型。
- DeleteProjectVersion：删除 Amazon Rekognition Custom Labels 模型。
- DescribeProjectVersions：返回特定项目中所有 Amazon Rekognition Custom Labels 模型的列表。

标签

- TagResource：向 Amazon Rekognition Custom Labels 模型添加一个或多个键值标签。
- UntagResource：从 Amazon Rekognition Custom Labels 模型中移除一个或多个标签。

使用模型

- DetectCustomLabels：使用自定义标签模型来分析图像。
- StartProjectVersion：启动自定义标签模型。
- StopProjectVersion：停止自定义标签模型。

Amazon Rekognition Custom Labels 文档历史记录

下表列出了《Amazon Rekognition Custom Labels 开发人员指南》每次发布时进行的重要修改。要获得本文档的更新通知，您可以订阅 RSS 源。

- 最近一次更新文档的日期：2023 年 4 月 19 日

变更	说明	日期
添加了模型运行时长主题	说明如何获取模型的运行时数和使用的推理单元。有关更多信息，请参阅报告运行时长和使用的推理单元。	2023 年 4 月 19 日
重新组织了数据集内容	将清单文件创建内容移到了清单文件中。将数据集转换主题移到了将其他数据集格式转换为清单文件中。	2023 年 2 月 20 日
更新了 AWS WAF 的 IAM 指南	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践。	2023 年 2 月 15 日
查看分类模型的混淆矩阵	Amazon Rekognition Custom Labels 控制台不会显示分类模型的混淆矩阵。您可以使用 AWS SDK 来获取并显示混淆矩阵。有关更多信息，请参阅查看模型的混淆矩阵。	2023 年 1 月 4 日
更新了 Lambda 函数示例	Lambda 函数示例现在说明如何分析从本地文件或 Amazon S3 存储桶传递的图像。有关更多信息，请参阅使用 AWS Lambda 函数分析图像。	2022 年 12 月 2 日

Amazon Rekognition Custom Labels 现在可以复制经过训练的模型	现在，您可以将经过训练的模型从一个 AWS 账户复制到同一 AWS 区域内的另一个 AWS 账户。有关更多信息，请参阅复制 Amazon Rekognition Custom Labels 模型 (SDK)。	2022 年 8 月 16 日
Amazon Rekognition Custom Labels 现在可以自动扩缩推理单元。	为了帮助应对需求高峰，Amazon Rekognition Custom Labels 现在可以自动扩缩供模型使用的推理单位数量。有关更多信息，请参阅运行经过训练的 Amazon Rekognition Custom Labels 模型。	2022 年 8 月 16 日
通过 CSV 文件创建清单文件	现在，您可以使用脚本从 CSV 文件中读取图像分类信息，从而简化清单文件的创建。有关更多信息，请参阅通过 CSV 文件创建清单文件。	2022 年 2 月 2 日
Amazon Rekognition Custom Labels 现在可以通过项目管理数据集	您可以使用项目来管理用于创建模型的训练和测试数据集。有关更多信息，请参阅了解 Amazon Rekognition Custom Labels。	2021 年 11 月 1 日
Amazon Rekognition Custom Labels 已与 AWS CloudFormation 集成	您可以使用 AWS CloudFormation 调配和配置 Amazon Rekognition Custom Labels 项目。有关更多信息，请参阅使用 AWS CloudFormation 创建项目。	2021 年 10 月 21 日

更新了入门体验	Amazon Rekognition Custom Labels 控制台现在包含教程视频和示例项目。有关更多信息，请参阅 Amazon Rekognition Custom Labels 入门。	2021 年 7 月 22 日
更新了有关阈值和使用指标的信息	有关使用 `DetectCustomLabels` 的 `MinConfidence` 输入参数设置所需阈值的信息。有关更多信息，请参阅使用经过训练的模型分析图像。	2021 年 6 月 8 日
增加了 AWS KMS key 支持	现在，您可以使用自己的 KMS 密钥对训练和测试图像进行加密。有关更多信息，请参阅训练模型。	2021 年 5 月 19 日
增加了标记功能	Amazon Rekognition Custom Labels 现在支持标记功能。可以使用标签识别、整理、搜索和筛选 Amazon Rekognition Custom Labels 模型。有关更多信息，请参阅标记模型。	2021 年 3 月 25 日
更新了设置主题	更新了有关如何加密训练文件的设置信息。有关更多信息，请参阅步骤 5：（可选）加密训练文件。	2021 年 3 月 18 日
添加了数据集复制主题	有关如何将数据集复制到其他 AWS 区域的信息。有关更多信息，请参阅将数据集复制到其他 AWS 区域。	2021 年 3 月 5 日

添加了 Amazon SageMaker GroundTruth 多标签清单转换主题	有关如何将 Amazon SageMaker Ground Truth 多标签格式清单转换为 Amazon Rekognition Custom Labels 格式清单文件的信息。有关更多信息，请参阅转换多标签 SageMaker Ground Truth 清单文件。	2021 年 2 月 22 日
添加了模型训练调试信息	现在，您可以使用验证结果清单来获取有关模型训练错误的深入调试信息。有关更多信息，请参阅调试失败的模型训练。	2020 年 10 月 8 日
添加了 COCO 转换信息和示例	有关如何将 COCO 物体检测格式数据集转换为 Amazon Rekognition Custom Labels 清单文件的信息。有关更多信息，请参阅转换 COCO 数据集。	2020 年 9 月 2 日
Amazon Rekognition Custom Labels 现在支持单个物体训练	要创建查找单个物体位置的 Amazon Rekognition Custom Labels 模型，您现在可以创建只需要一个标签的数据集。有关更多信息，请参阅绘制边界框。	2020 年 6 月 25 日
添加了项目和模型删除操作	您现在可以使用控制台和 API 删除 Amazon Rekognition Custom Labels 项目和模型。有关更多信息，请参阅删除 Amazon Rekognition Custom Labels 模型和删除 Amazon Rekognition Custom Labels 项目	2020 年 4 月 1 日

添加了 Java 示例	添加了涵盖项目创建、模型训练、模型运行和图像分析的 Java 示例。	2019 年 12 月 13 日
新功能和指南	这是 Amazon Rekognition Custom Labels 功能和《Amazon Rekognition Custom Labels 开发人员指南》的首次发布。	2019 年 12 月 3 日

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。