**AWS Rekognition Background:** AWS service that to add powerful visual analysis to your applications. There are two different API sets: One for analyzing images and one for the video analysis.

* **Rekognition Image**
* IT detects objects, scenes, and faces; extracts text etc and allows you to search and compare faces. Common use cases are- Searchable Image Library, Face-Based User Verification, Sentiment Analysis, Facial Recognition, Image Moderation
* Currently supports the **JPEG and PNG** image formats as **S3 object (up to 15MB) or as a byte array (up to 5MB)**.
* Rekognition Image returns the bounding box for each face detected in an image along with its attributes such as sex, accessories, facial features, etc. Using the CompareFaces API, lets you measure the likelihood that faces in two images are of the same person.
* **Image Operations**
* Detect Labels
* Detect Faces
* Compare Faces
* **Rekognition Video**
  + - It detects activities; understands the movement of people in frame; and recognizes objects, celebrities, text, scenes, and many more in a video. Common use cases are search Index for video archives, easy filtering of video for explicit and suggestive content etc
    - Rekognition Video operations can analyze videos (up to 8GB) stored in Amazon S3 buckets. The video must be encoded using the H.264 codec. The supported file formats are **MPEG-4 and MOV**.
* **Video Operations**
* Preparations
* Detect Labels
* Track Persons
  + **Key concepts and terminology**
    - A **label** is an object, scene, or concept found in an image based on its contents.
    - Each label comes with a confidence score. A **confidence score** is a number between 0 and 100 that indicates the probability that a given prediction is correct.
    - **Object and Scene Detection** is the process of analyzing an image or video to assign labels based on its visual content. Rekognition Image does this through the DetectLabels API.
    - **Unsafe Content Detection** is a deep-learning based API for detection of explicit, rude and suggestive adult content in images. Very useful for filtering inappropriate content.
    - **Facial Recognition** is the process of identifying or verifying a person’s identity by searching for their face in a collection of faces. You can create a face collection as your dataset for comparison.
    - **Text in Image** allows you to detect and recognize text within an image, and is specifically built to work with real-world images rather than document images.
    - **Celebrity Recognition** is Amazon Rekognition’s feature for recognizing celebrities within supplied images and in videos

**Steps to setup:**

**Step 1: Set up an AWS account and create an IAM user**

Before you can use Rekognition, you'll need to create an AWS account and obtain an AWS account ID. You will also want to create an IAM user, which enables the Amazon Rekognition system to determine if you have the permissions needed to access its resources.

**Step 2: Set up the AWS CLI and AWS SDKs**

After creating your accounts, you'll want to install and configure the AWS CLI and AWS SDKs. The AWS CLI lets you interact with Amazon Rekognition and other services through the command line, while the AWS SDKs let you use programming languages like Java and Python to interact with Amazon Rekognition.

**Step 3: Using the AWS CLI and AWS SDK API**

**The following explains detection of labels in an image stored in S3.**

Upload an image that contains one or more objects—such as trees, houses, and boat—to your S3 bucket. The image must be in .jpg or .png format.

Call the DetectLabels operation.

def main():

photo=''

bucket=''

label\_count=detect\_labels(photo, bucket)

print("Labels detected: " + str(label\_count))

The response from DetectLabels is an array of labels detected in the image and the level of confidence by which they were detected.

{

"Name": "Car",

"Confidence": 99.15271759033203,

"Instances": [

{

"BoundingBox": {

"Width": 0.10616336017847061,

"Height": 0.18528179824352264,

"Left": 0.0037978808395564556,

"Top": 0.5039216876029968

},

"Confidence": 99.15271759033203

},

Similarly, other image and video operations can be performed.

**Step 4: Using the Amazon Rekognition console**

Rekognition console allows you to use a subset of Amazon Rekognition's capabilities such as object and scene detection, facial analysis, and face comparison in a set of images. This can be used just to get and idea of the accuracy coming in your video/image datasets. It is simply drag and drop functionality which accepts your sample image and return confidence score along with other parameters.

**AWS Lex Background:** Amazon Lex enables you to build applications using a speech or text interface powered by the same technology that powers Amazon Alexa.

* + **Key concepts and terminology**
* **Bot** – A bot performs automated tasks such as ordering a pizza, booking a hotel, ordering flowers, and so on. An Amazon Lex bot is powered by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) capabilities. Each bot must have a unique name within your account.

Amazon Lex bots can understand user input provided with text or speech and converse in natural language. You can create Lambda functions and add them as code hooks in your intent configuration to perform user data validation and fulfillment tasks.

* **Intent** – An intent represents an action that the user wants to perform. You create a bot to support one or more related intents. For each intent, you provide the following required information:
* **Intent name**– A descriptive name for the intent. For example, **OrderPizza**. Intent names must be unique within your account.
* **Sample utterances** – How a user might convey the intent. For example, a user might say "Can I order a pizza please" or "I want to order a pizza".
* **How to fulfill the intent** – How you want to fulfill the intent after the user provides the necessary information**.**
* **Slot** – An intent can require zero or more slots or parameters. You add slots as part of the intent configuration. At runtime, Amazon Lex prompts the user for specific slot values. The user must provide values for all required slots before Amazon Lex can fulfill the intent.

For example, the OrderPizza intent requires slots such as pizza size, crust type, and number of pizzas.

* **Slot type** – Each slot has a type. You can create your custom slot types or use built-in slot types. Each slot type must have a unique name within your account. For example, you might create and use the following slot types for the OrderPizza intent:
  + - Size – With enumeration values Small, Medium, and Large.
    - Crust – With enumeration values Thick and Thin.

**Steps to create a custom bot:**

[**Step 1: Create a Lambda Function**](https://docs.aws.amazon.com/lex/latest/dg/gs2-prepare.html)

**C**reate a Lambda function which fulfills forsay, a pizza order. This function is specified in your Amazon Lex bot, which you create in the next section. When a user provides all of the slot data required to fulfill the intent, your Lambda function will be invoked, if enabled. Lex basically sends data to the lambda function in a specific format. The Lambda function thus performs the business logic.

[**Step 2: Create a Bot**](https://docs.aws.amazon.com/lex/latest/dg/gs2-create-bot.html)

After finishing all the configurations, you need to build your bot before you start testing. Lex provides a test window where you can test your bot and see how the bot replies to the user’s queries.

[**Step 3: Build and Test the Bot**](https://docs.aws.amazon.com/lex/latest/dg/gs2-build-and-test.html)

Please note that if you are changing any intent configuration you have to build it again to make the latest changes visible in the test window. Publishing the bot creates a new version of it. It allows you to create a different version of your bots and you can control the version which your application use.