By AWS re/Start NCL2 Team

User Guide

A serverless image recognition app

# Table of Contents

[1 Mission 2](#_Toc69386490)

[2 Solution 2](#_Toc69386491)

[3 Workflow 2](#_Toc69386492)

[4 Python Functions 3](#_Toc69386493)

[5 Sample Execution. 4](#_Toc69386494)

# Mission

The goal of the project is to develop a Serverless Image Recognition App.

# Solution

This can be achieved using AWS lambda to trigger AI image recognition service when user upload the image to S3 bucket, then store the label/result of the image in DynamoDB.

This can also be viewed from CloudWatch logs.

# Workflow

Diagram

Description automatically generated

Prerequisites

1. Python3 installed (<https://www.python.org/downloads/>)
2. Boto3 package installed (<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html>).
3. AWS free tier account created.
4. AWS CLI configuration (<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html>)
5. Necessary IAM role configuration and permissions (Figure 1).
6. S3 bucket for image upload (prefix: images).
7. Lambda trigger configuration (Figure 2).
8. Pre-existing DynamoDB table (named ImageLabels).
9. Git/Github account (optional).

# Python Functions

The solution achieved is divided into two main Python files: s3upload.py and lambda\_function.py.

**s3upload.py**: This file contains the following functions to upload an image selected by the user to a specified s3 bucket.

* **get\_existing\_objects** – this function gets the list of existing objects from the s3 bucket.
* **upload\_file** – this function uploads the selected image to the desired s3 bucket.

Graphical user interface, text, application

Description automatically generated

**Lambda\_function.py:** This file is consists of the following functions:

Graphical user interface, text, application

Description automatically generated

* **detect\_labels**: this function uses the recognition service to detect the labels in a given image and return the labels with the respective confidence levels.
* **lambda\_handler**: is the main execution point of the lambda function. It calls the detect\_labels function with the bucket and image arguments and inserts the returned results into a DynamoDB table.

# Sample Execution.

Example of uploaded image.

A picture containing outdoor, sky, tree, house

Description automatically generated

The Python console application for uploading images to AWS S3 service.

Text

Description automatically generated

The S3 bucket after the image being uploaded.

Graphical user interface, text, application

Description automatically generated

The Lambda function architecture.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedThe lambda function python code to detect labels in uploaded image.

The output of the detection function monitored in CloudWatch.

Graphical user interface, text, application, email

Description automatically generated

The DynamoDB table

Graphical user interface, website

Description automatically generated

Git/GitHub

Graphical user interface, text, application, email

Description automatically generated

GitHub Desktop

Graphical user interface, text

Description automatically generated

References

Table

Description automatically generated

Figure - Basic Execution Role and Policies for Lambda

Graphical user interface, text, application, email

Description automatically generated

Figure -Lambda Trigger configuration