**Introduction**

**Title:** Image Processing with AWS Lambda

* **Introduction:**
  + Overview of the project.
  + Goals: Convert uploaded images to black and white, store in S3, and email results.

To accomplish the described task, you can follow these general steps using AWS services:

**Architecture Overview**

**Title:** System Architecture

* **Components:**
  + S3 Buckets (Source)
  + Lambda Function
  + SES (Simple Email Service)
* **Flow:**
  + Image upload triggers Lambda.
  + Lambda processes the image and stores it in the Destination S3 bucket.
  + Email sent with original and processed images.

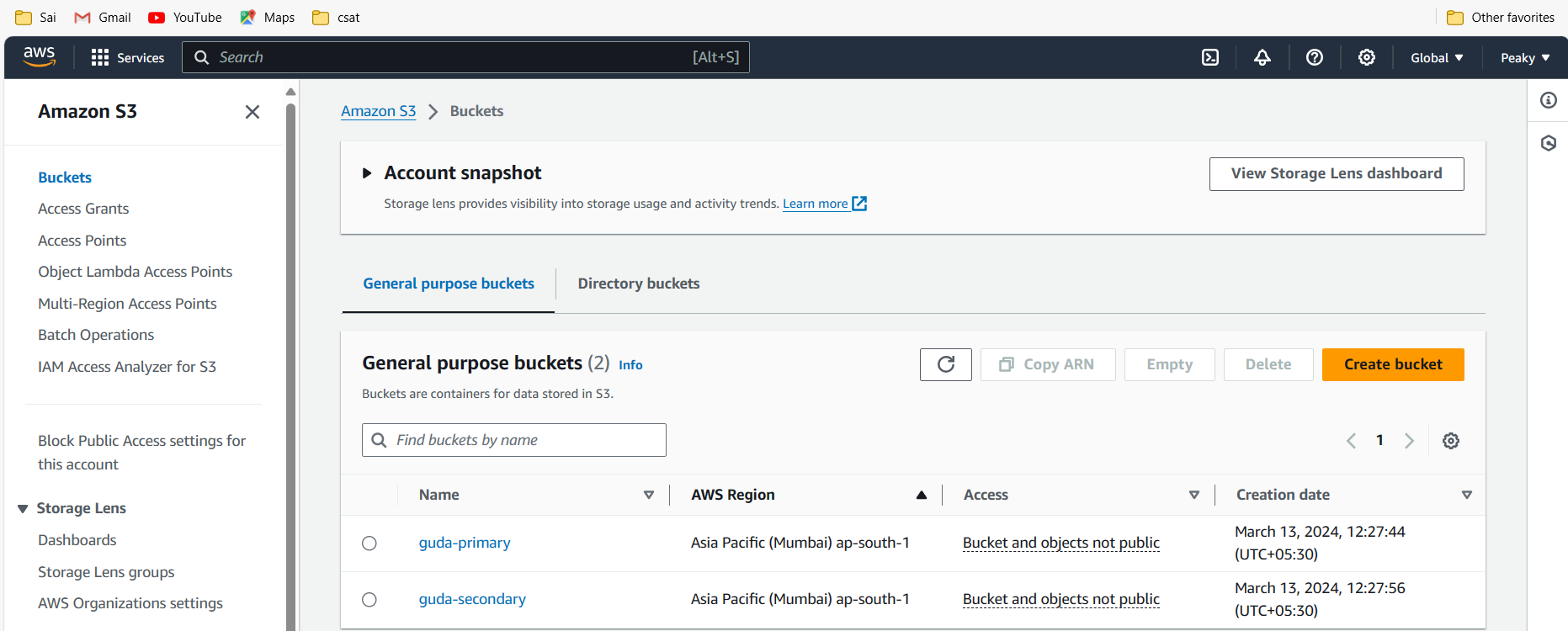
**AWS Resources Setup**

**Title:** Setting Up AWS Resources

* **Tasks:**
  + Create two S3 buckets.
  + Set up an IAM role for Lambda.
  + Configure SNS for sending emails.

**1.0 Amazon S3:**

* + Create two S3 buckets, one for the original images and one for the processed (black/white) images.



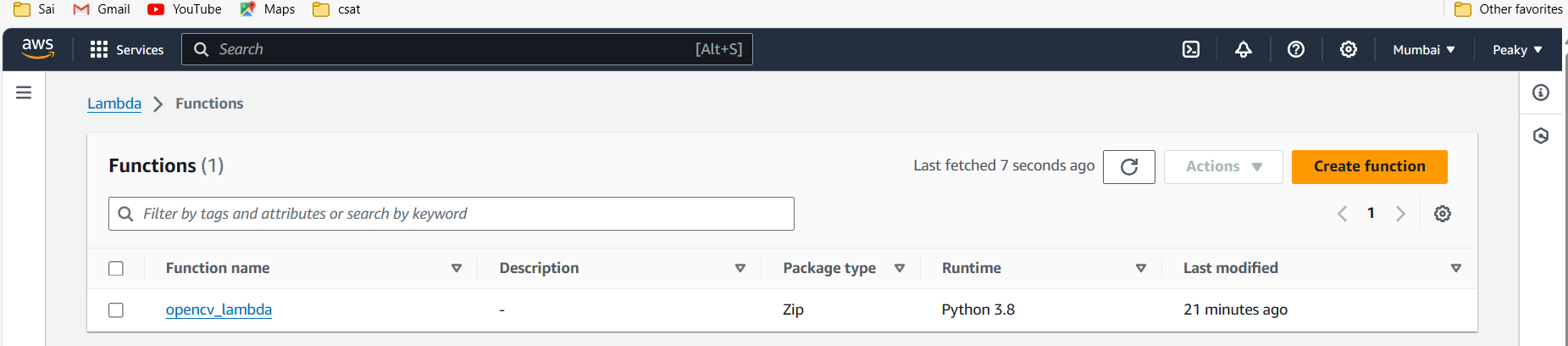
**Lambda Function**

**Title:** Lambda Function Configuration

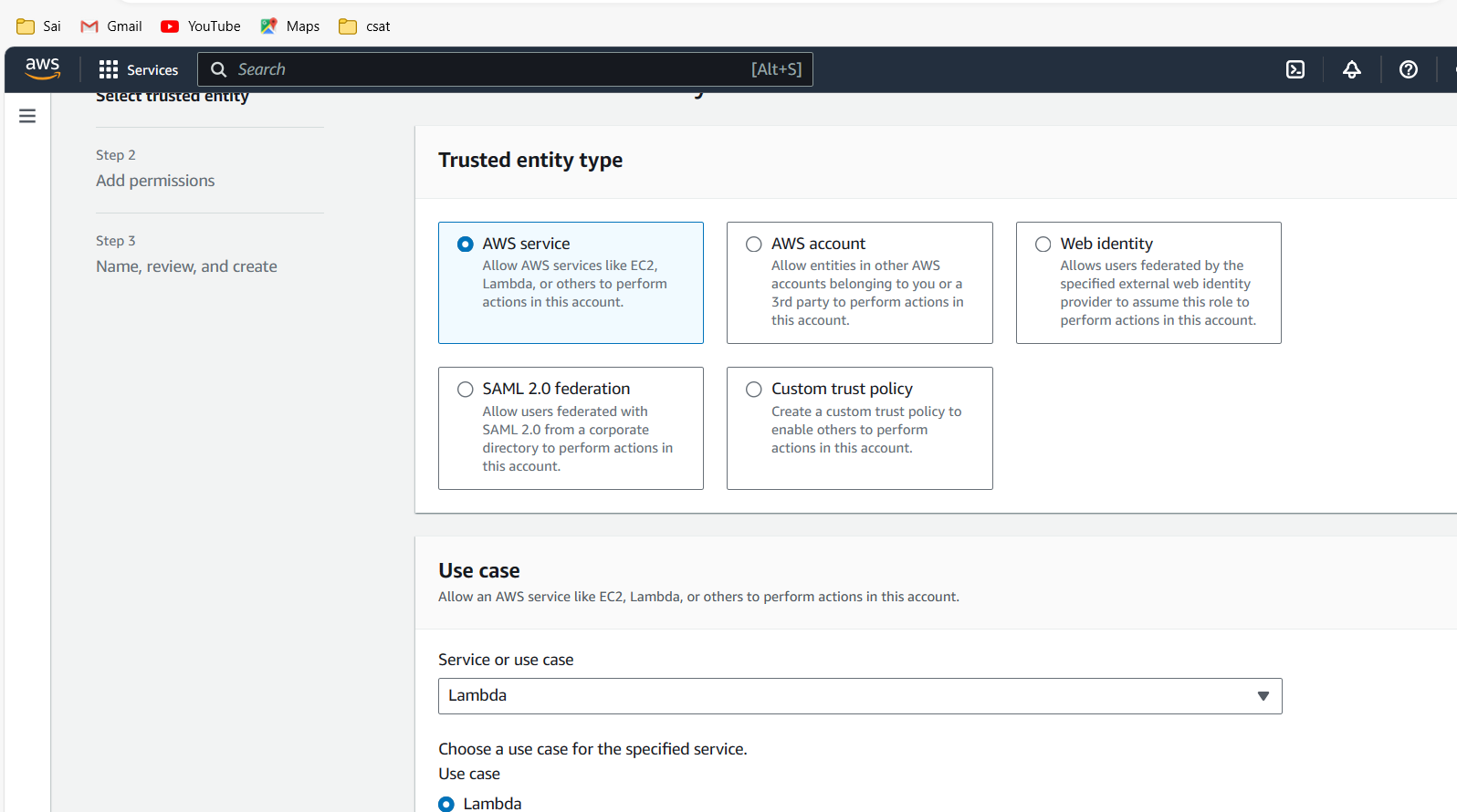
* **Configuration:**
  + Create a Lambda function.
  + Set up the IAM role with necessary permissions.
  + Configure S3 trigger event.
* **Code:**
  + Python code using OpenCV for image processing.

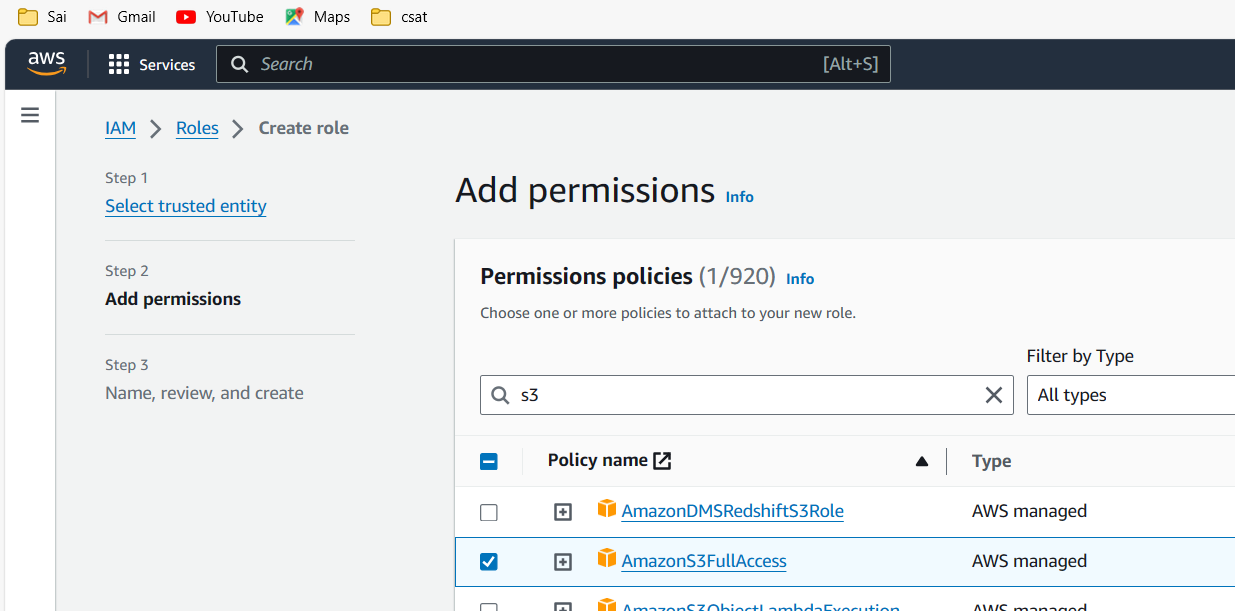
**2.0 AWS Lambda:**

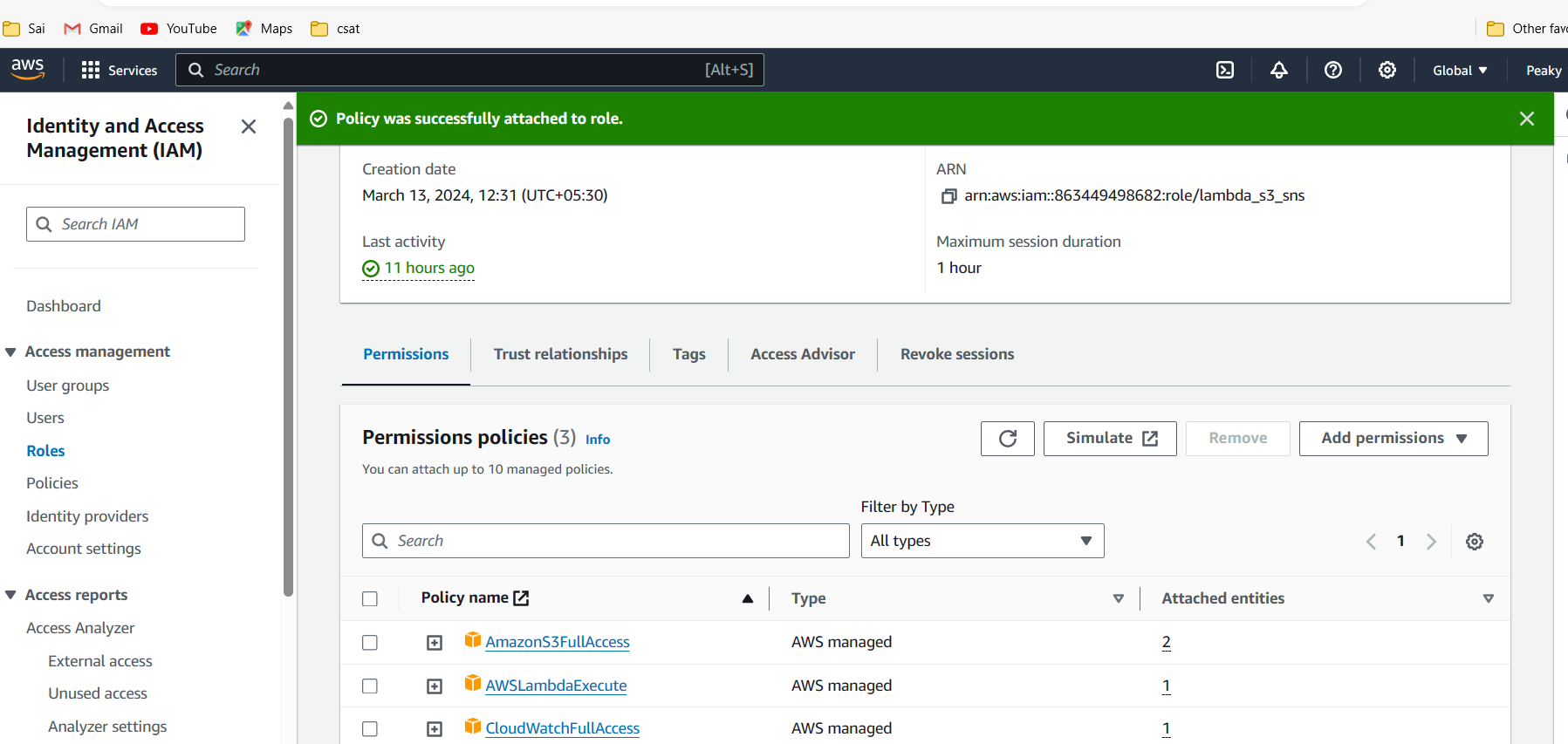
* Create an AWS Lambda function that triggers on an S3 bucket event (e.g., object creation).



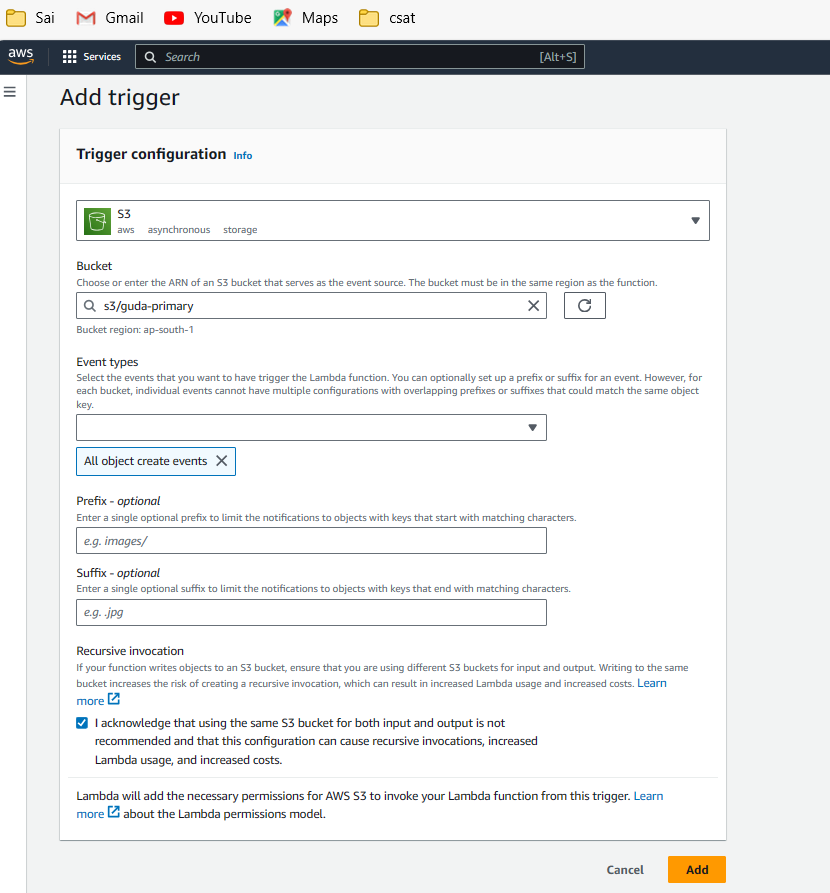
* Configure the Lambda function to use the appropriate IAM role with permission to read from the source S3 bucket, write to the destination S3 bucket, and send emails.







In Lambda-Add Trigger:



**3.0 Python with OpenCV:**

* Write a Lambda function in Python that uses the OpenCV library to convert the image to black/white.
* Ensure that the necessary OpenCV dependencies are included in the deployment package.

**OpenCV** is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human.

Certainly! **OpenCV** (Open Source Computer Vision Library) is a powerful tool for computer vision, image processing, and machine learning tasks. It provides a wide range of functions and algorithms to work with images, videos, and other visual data. Here are some details about using OpenCV in Python:

**Installation**:

* + To install OpenCV in Python, you can use **pip**, the package manager for Python. Depending on your operating system, execute one of the following commands:
    - **Windows**: pip install opencv-python
    - **macOS**: brew install opencv3 --with-contrib --with-python3
    - **Linux**: sudo apt-get install libopencv-dev python-opencv

**Importing Package into our Lambda Layers:-**

**Steps:**

1. Connect to UBUNTU VM from EC2 and run the following commands to install required packages for the project.

sudo apt-get update -y

python3 --version

sudo apt install python3-pip

sudo apt install awscli

mkdir -p build/python/lib/python3.8/site-packages

pip3 install opencv-python-headless -t build/python/lib/python3.8/site-packages

cd build

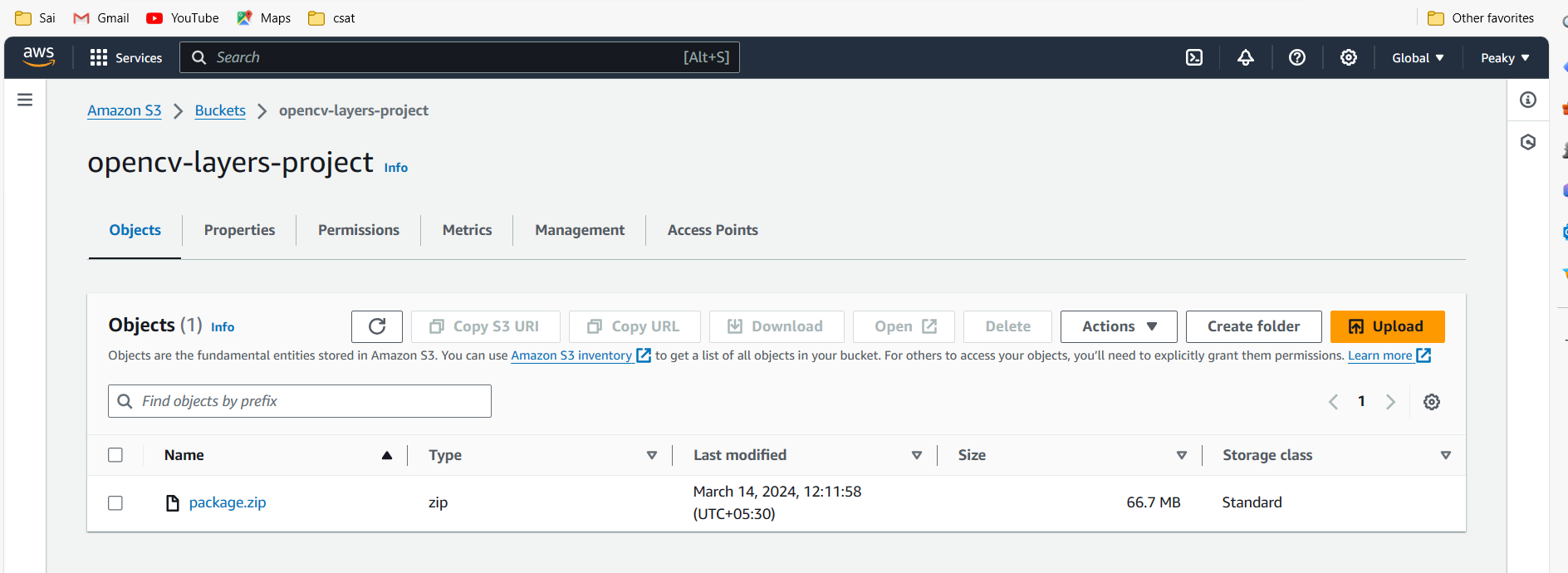
sudo apt install zip

zip -r package.zip .

ls (#we can notice our zip file)

aws s3 cp package.zip bucket\_name\_in\_s3

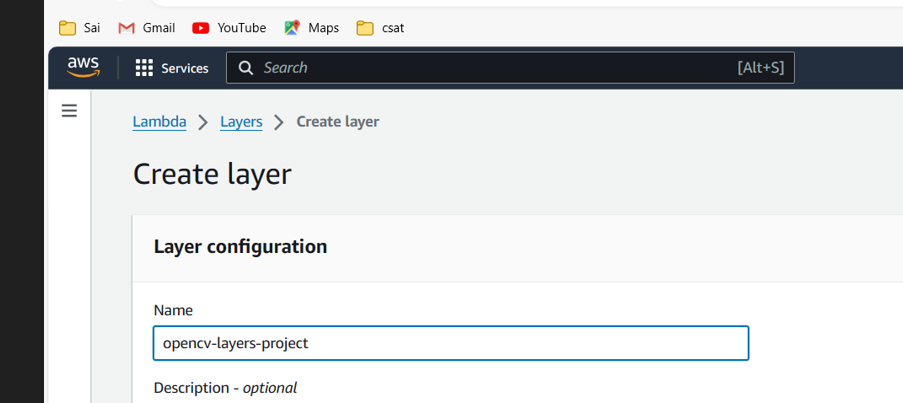
(#this command will help u to copy the zip file into s3 bucket u have created, in my case I have created OpenCV-layers-project)

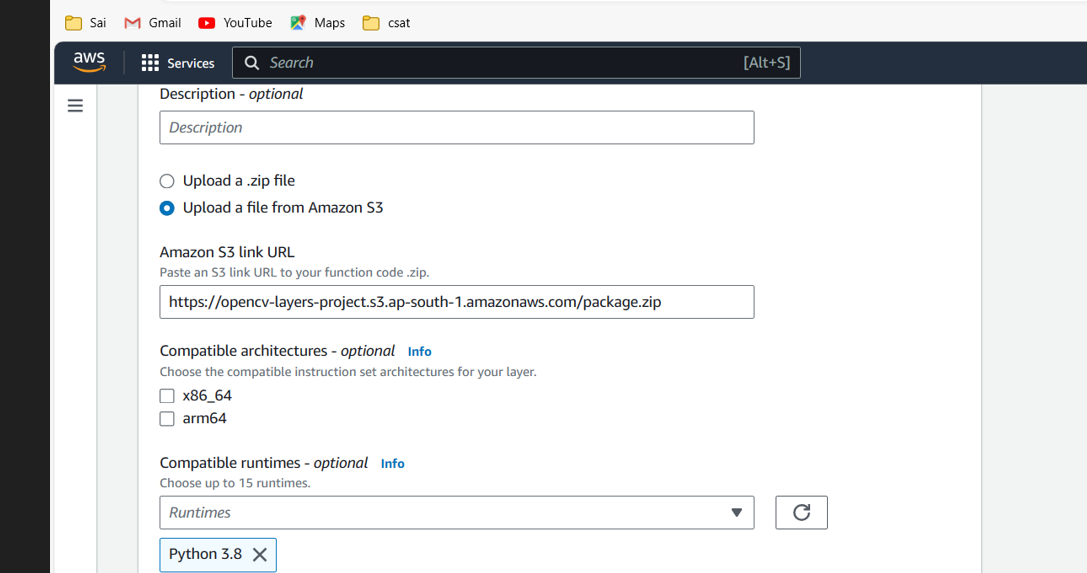


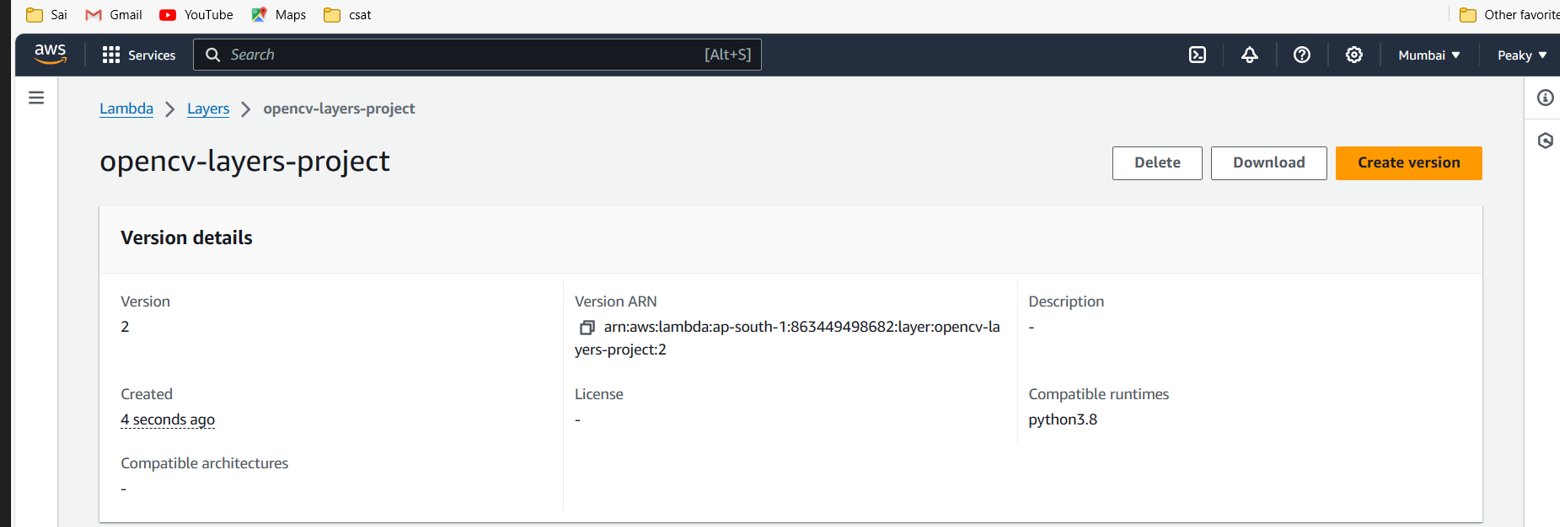
**Lambda Code Deployment**

**Title:** Deploying Lambda Function

* **Deployment:**
  + Create a layer in our lambda function and upload our zip file from S3 bucket. i.e. layers help the lambda function to import the packages required to execute our code.







Now, upload this code in the lambda function:-

import os

import cv2

import boto3

import numpy as np

s3 = boto3.client('s3')

def lambda\_handler(event, context):

# Get the bucket and key from the S3 event

source\_bucket = event['Records'][0]['s3']['bucket']['name']

source\_key = event['Records'][0]['s3']['object']['key']

# Read the image from the source S3 bucket

response = s3.get\_object(Bucket=source\_bucket, Key=source\_key)

image\_content = response['Body'].read()

nparr = np.frombuffer(image\_content, np.uint8)

img = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

# Convert image to black and white

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Encode black and white image to bytes

\_, bw\_image\_bytes = cv2.imencode('.jpg', gray\_img)

# Destination bucket where the black and white image will be uploaded

destination\_bucket = "guda-secondary"

# Upload the black and white image to the destination S3 bucket

destination\_key = 'black\_and\_white\_' + os.path.basename(source\_key)

s3.put\_object(Body=bw\_image\_bytes.tobytes(), Bucket=destination\_bucket, Key=destination\_key)

return {

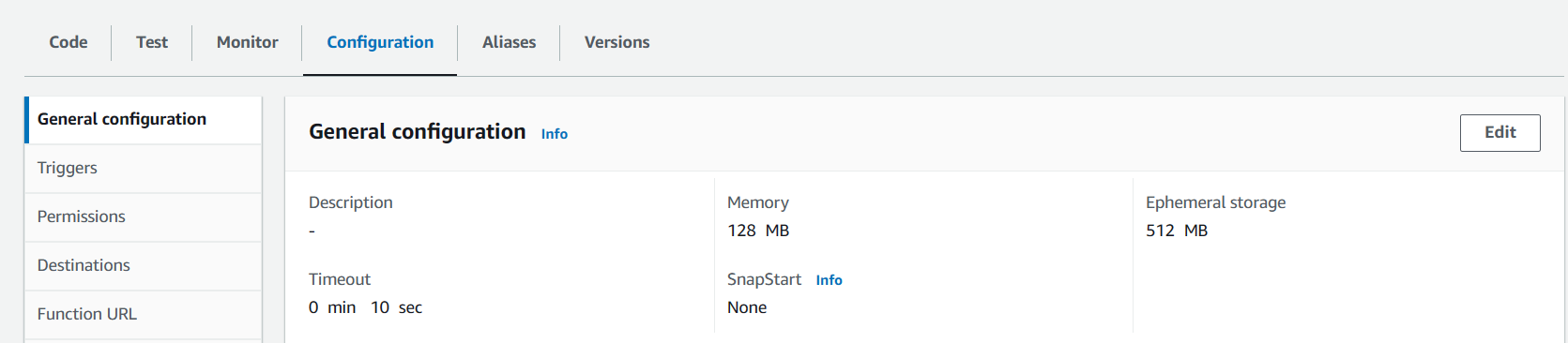
'statusCode': 200,

'body': f'Image converted to black and white and saved as {destination\_key}'

}

Now, deploy the file, so it will save your code.

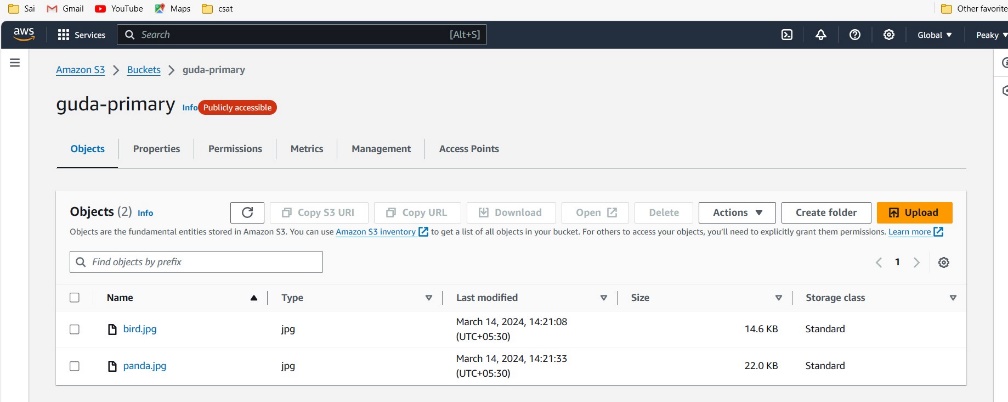
If you are get any errors, kindly edit the general configuration of lambda under configuration tab, increase the timeout seconds.



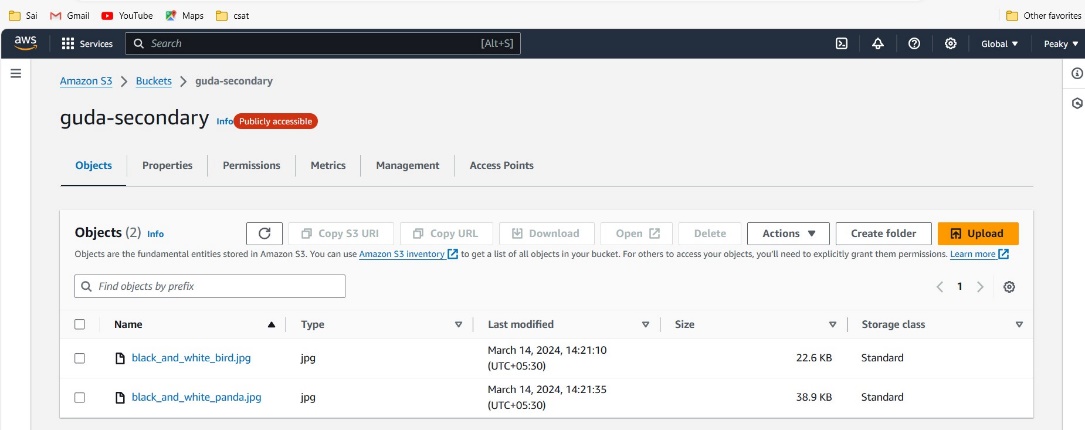
Now, upload a image in your primary bucket, in my case it is (guda-primary).

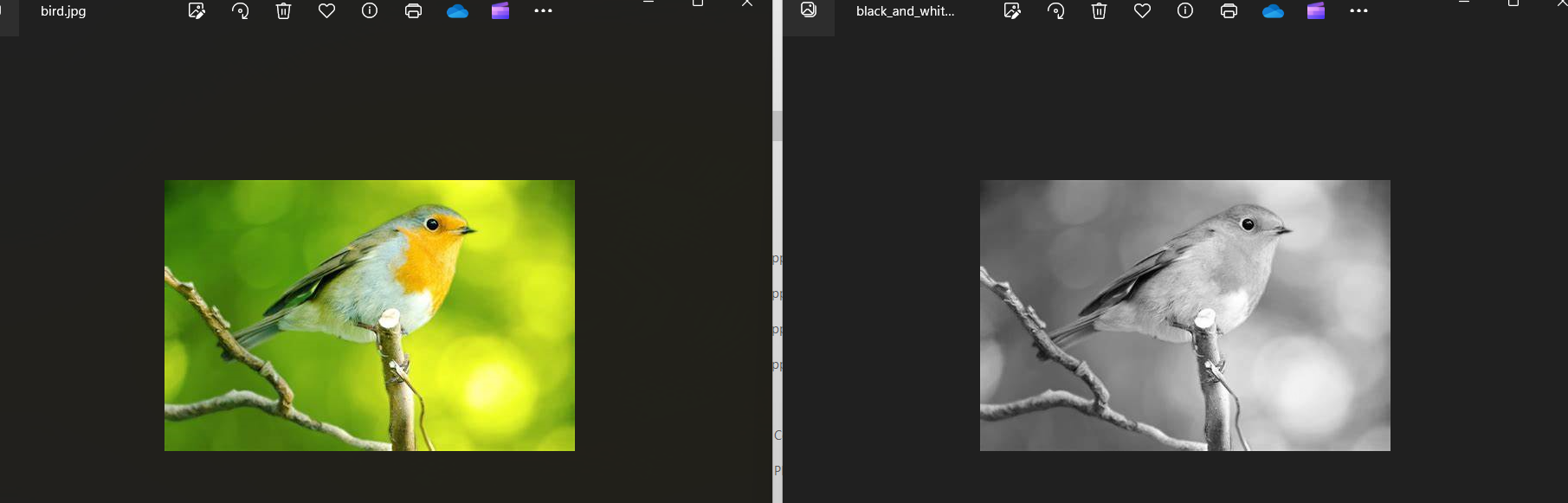
This will automatically trigger our lambda function to convert our image into black&white and uploaded back into our destination bucket as we mentioned in our code(guda-secondary).

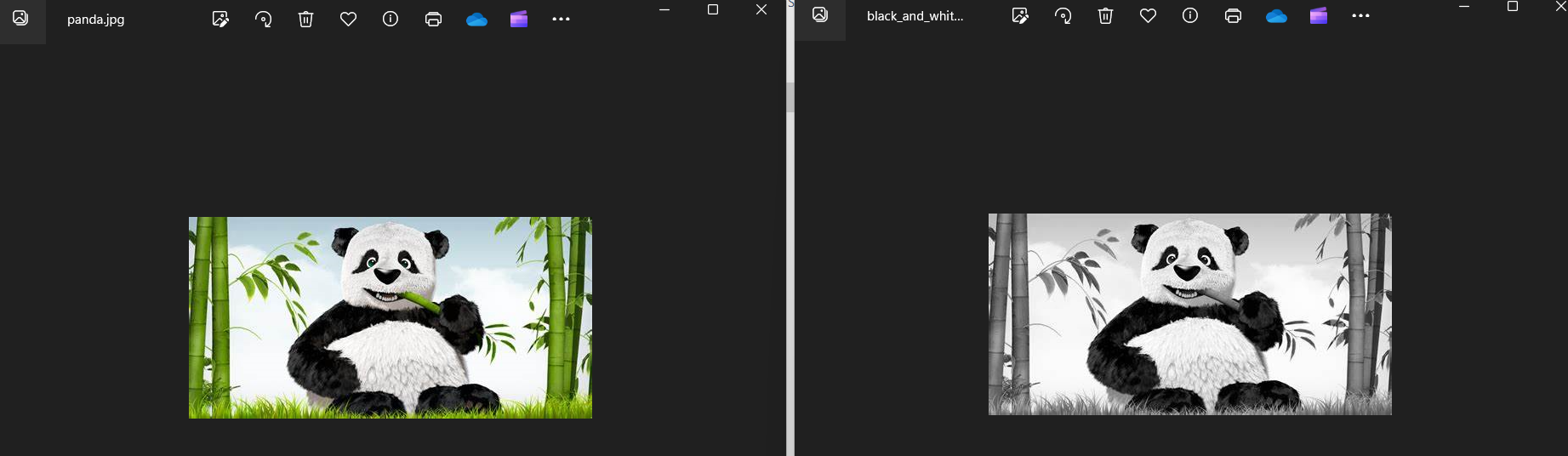
My primary bucket with uploaded images…



My secondary bucket with black and white conversion by lambda function….







I hope you enjoyed the project in detail, I am giving a task to add the SES (simple email notification) service to this project and try it again.

Thank you,

Regards:

SAI GUDA