**Job-Oriented Project:**

**Real-Time *Image* Processing**

***using***

**AWS Lambda and EC2**

This project will demonstrate how to set up a real-time image processing system using a **combination** of AWS Lambda and EC2. The system will process images **uploaded** to an S3 bucket. AWS Lambda will **trigger** the processing **event** for simple image **resizing**, and an EC2 instance will be used for **more** computationally intensive operations such as applying **filters** or transformations.

**Project Architecture**

* **S3: Used to store raw and processed images.**
* **Lambda: Automatically resizes images when uploaded to an S3 bucket.**
* **EC2: Processes images by applying complex filters (e.g., color corrections, sharpening).**
* **CloudWatch: Monitors the Lambda function and EC2 instance.**
* **IAM: Provides the necessary permissions for Lambda, EC2, and S3 interaction.**

**Step-by-Step Implementation**

**Step 1: Set Up S3 Bucket**

1. Create an S3 bucket\*\* for **raw** images.

**aws s3 mb s3://**my-image-bucket

2. Enable **versioning** for the bucket to track image changes.

**aws s3api put-bucket-versioning --bucket** my-image-bucket --versioning-configuration Status=Enabled

3. Create another S3 bucket\*\* for **processed** images.

aws s3 **mb** s3://my-processed-image-bucket

**Step 2: Create an IAM Role for Lambda**

1. Create an IAM role with policies for Lambda to access S3 and CloudWatch.

aws iam create-role --role-name LambdaS3Role --assume-role-policy-document file://trust-policy.json

**vim** trust-policy.json

**{**

**"Version": "2012-10-17",**

**"Statement": {**

**"Effect": "Allow",**

**"Principal": {**

**"Service": "lambda.amazonaws.com"**

**},**

**"Action": "sts:AssumeRole"**

**}**

**}**

2. Attach S3 and CloudWatch policies to the role.

aws iam **attach**-role-policy --role-name LambdaS3Role --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

aws iam attach-role-policy --role-name LambdaS3Role --policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess

**Step 3: Deploy the Lambda Function**

1. Create a Python function to resize the image.

**import boto3**

**from PIL import Image**

**import io**

**def lambda\_handler(event, context):**

**s3 = boto3.client('s3')**

**bucket = 'my-image-bucket'**

**key = event['Records'][0]['s3']['object']['key']**

**response = s3.get\_object(Bucket=bucket, Key=key)**

**img = Image.open(io.BytesIO(response['Body'].read()))**

**# Resize the image**

**img = img.resize((200, 200))**

**out\_img = io.BytesIO()**

**img.save(out\_img, format='JPEG')**

**out\_img.seek(0)**

**# Save the resized image to the processed bucket**

**s3.put\_object(Bucket='my-processed-image-bucket', Key=f'resized-{key}', Body=out\_img)**

**return 'Image resized and uploaded.'**

2. Create a Lambda function and upload the Python code.

**zip function.zip lambda\_function.py**

**aws lambda create-function --function-name ImageResizer \**

**--runtime python3.8 \**

**--role arn:aws:iam::123456789012:role/LambdaS3Role \**

**--handler lambda\_function.lambda\_handler \**

**--zip-file fileb://function.zip**

3. Set up an S3 **trigger** to invoke Lambda when a new image is uploaded.

**aws lambda add-permission --function-name ImageResizer \**

**--principal s3.amazonaws.com \**

**--statement-id S3Invoke \**

**--action "lambda:InvokeFunction" \**

**--source-arn arn:aws:s3:::my-image-bucket**

**aws s3api put-bucket-notification-configuration \**

**--bucket my-image-bucket \**

**--notification-configuration file://notification.json**

**vim** notification.json

**{**

**"LambdaFunctionConfigurations": [**

**{**

**"LambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:ImageResizer",**

**"Events": ["s3:ObjectCreated:\*"]**

**}**

**]**

**}**

**Step 4: Launch an EC2 Instance for Advanced Image Processing**

1. Launch an EC2 instance to apply complex filters to images.

**aws ec2 run-instances \**

**--image-id ami-0123456789abcdef0 \**

**--count 1 \**

**--instance-type t2.medium \**

**--key-name MyKeyPair \**

**--security-group-ids sg-0123456789abcdef0 \**

**--subnet-id subnet-0123456789abcdef0**

2. SSH into the EC2 instance:

**ssh** **-i** "MyKeyPair.pem" ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com

3. Install Python and image processing libraries:

**sudo yum install python3**

**pip3 install pillow boto3**

4. Create a Python script on EC2 to apply filters to the images.

Example code:

**from PIL import Image, ImageFilter**

**import boto3**

**import io**

**s3 = boto3.client('s3')**

**def process\_image(bucket, key):**

**response = s3.get\_object(Bucket=bucket, Key=key)**

**img = Image.open(io.BytesIO(response['Body'].read()))**

**# Apply a filter**

**img = img.filter(ImageFilter.CONTOUR)**

**out\_img = io.BytesIO()**

**img.save(out\_img, format='JPEG')**

**out\_img.seek(0)**

**# Upload the processed image**

**s3.put\_object(Bucket='my-processed-image-bucket', Key=f'filtered-{key}', Body=out\_img)**

**# Sample execution**

**process\_image('my-image-bucket', 'image.jpg')**

**Step 5: Create an IAM Role for EC2**

1. Create an EC2 **role** with access to S3.

aws iam create-role --role-name EC2S3Role --assume-role-policy-document file://ec2-trust-policy.json

2. Attach S3 permissions:

aws iam attach-role-policy --role-name EC2S3Role --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

3. Assign the role to your EC2 instance using the AWS Console:

- Go to EC2 > Instances.

- Select your instance, click **Actions**, and **Modify IAM role**.

- Assign the `EC2S3Role` to the instance.

**Step 6: Monitor with CloudWatch**

1. View Lambda logs in CloudWatch:

- Go to CloudWatch > Logs.

- Locate the log group for the Lambda function to monitor executions and check for errors.

2. Set up alarms for EC2:

- Go to CloudWatch > Alarms.

- Create an alarm to monitor **CPU usage** or **memory** for the EC2 instance.

**Step 7: Test the System**

1. **Upload** an image to the S3 bucket:

**aws s3 cp** test-image.jpg s3://my-image-bucket/

2. Verify that Lambda resized the image:

- Go to S3 > my-processed-image-bucket and check for the resized image (`resized-test-image.jpg`).

3. Run the EC2 image processing script to apply filters:

python3 process\_image.py

4. Verify the filtered image in the `my-**processed**-image-bucket`.

**Summary**

- AWS Lambda handles the initial image resizing triggered by S3 events, allowing for fast, on-demand processing.

- EC2 is used for more complex image manipulation, such as applying filters, and can be scaled or optimized based on workload.

- S3 serves as the central storage for raw and processed images.

- CloudWatch monitors the health and performance of both Lambda and EC2 components.

This project demonstrates the balance of using AWS Lambda for **lightweight, event-driven** processing and EC2 for more **complex**, **long-running** tasks.