Session-15-Serverless

Lab1: Serverless Image Upload and Processing System

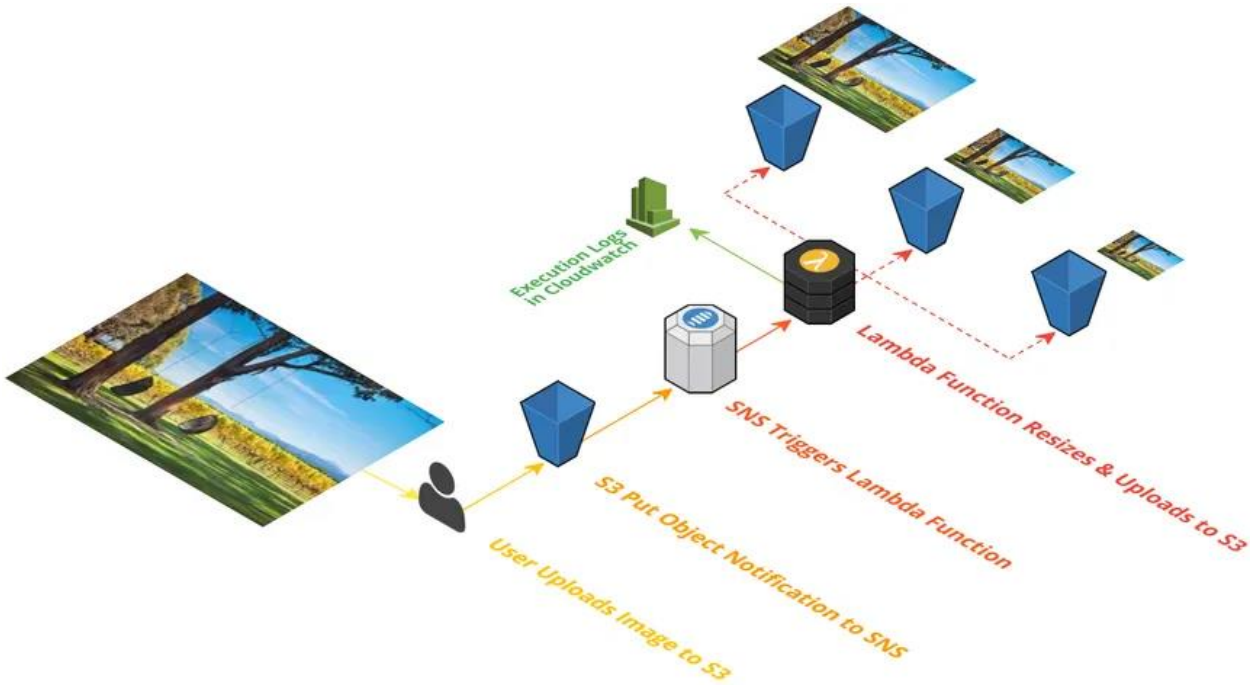
Objective

Automatically convert user-uploaded images (e.g., PNG to JPEG) using:

- Amazon S3 for uploads and output storage
- AWS Lambda for image processing using the `Pillow` (PIL) library
- S3 Event-based trigger to invoke Lambda

This lab is commonly used in photo apps, e-commerce platforms, and moderation systems.

Architecture Overview



Prerequisites

- AWS account with permissions for S3 and Lambda
- IAM role for Lambda with access to S3
- AWS Console or AWS CLI access

Step-by-Step Setup

Step 1: Create Two S3 Buckets

1. Open the **S3 Console**.
2. Create a bucket:
    - Name: `image-upload-bucket-<yourname>`
    - Disable public access
3. Create a second bucket:
    - Name: `image-output-bucket-<yourname>`

Step 2: Create a Lambda Execution Role

1. Open the **IAM Console → Roles**
2. Click **Create role**
3. Choose **Lambda** as the trusted entity
4. Attach the following policies:
    - `AmazonS3FullAccess` (for testing; use tighter permissions in production)
    - `AWSLambdaBasicExecutionRole`
5. Name the role: `lambda-s3-image-role`

Step 3: Create the Lambda Function

1. Open the **Lambda Console**
2. Click **Create function**
    - Name: `ImageConverter`
    - Runtime: Python 3.12
    - Role: Use existing role `lambda-s3-image-role`
3. Replace the default code with the following:

```
import boto3
import os
```

```
from PIL import Image
from io import BytesIO

s3 = boto3.client('s3')
OUTPUT_BUCKET = 'image-output-bucket-<yourname>'

def lambda_handler(event, context):
    src_bucket = event['Records'][0]['s3']['bucket']['name']
    src_key = event['Records'][0]['s3']['object']['key']

    obj = s3.get_object(Bucket=src_bucket, Key=src_key)
    image_data = obj['Body'].read()

    with Image.open(BytesIO(image_data)) as img:
        img = img.convert("RGB")
        buffer = BytesIO()
        output_key = os.path.splitext(src_key)[0] + ".jpg"
        img.save(buffer, "JPEG")
        buffer.seek(0)

        s3.put_object(Bucket=OUTPUT_BUCKET, Key=output_key, Body=buffer,
ContentType='image/jpeg')

    return {
        'statusCode': 200,
        'body': f'Converted and uploaded to {OUTPUT_BUCKET}/{output_key}'
    }
```

Replace `image-output-bucket-<yourname>` with your actual output bucket name.

4. Deploy the function.

Step 4: Add PIL (Pillow) Layer to Lambda

PIL is not built into Lambda. Use a public layer:

- For `us-east-1`, Python 3.12:

```
arn:aws:lambda:us-east-1:770693421928:layer:Klayers-p312-Pillow:1
```

1. Go to your Lambda function
2. Click **Layers → Add a layer**
3. Choose **"Specify an ARN"** and paste the layer ARN

Adjust the region and version if needed.

Step 5: Configure S3 Event Trigger

1. Go to your **image-upload** bucket
2. Navigate to **Properties → Event notifications**
3. Add an event:
    - Name: `trigger-lambda-on-upload`
    - Event type: `PUT`
    - Optional prefix: `uploads/`
    - Suffix: `.png`
    - Destination: Lambda
    - Choose your Lambda function `ImageConverter`

Test the System

Upload a PNG to your upload bucket

You can upload using the console or AWS CLI:

```
Upload Image to your bucket manually, or via CLI

aws s3 cp test.png s3://image-upload-bucket-<yourname>/uploads/test.png
```

Check the output bucket

The converted `.jpg` file should appear in:

```
s3://image-output-bucket-<yourname>/uploads/test.jpg
```

Optional Enhancements

- Add resizing support (e.g., max width/height)
- Store outputs in a specific subfolder
- Support multiple file formats (e.g., BMP, GIF)
- Tag files with metadata