Serverless Image Processing Pipeline

Overview

In this project, I developed a serverless image processing pipeline using AWS Lambda, S3, and API Gateway. The system automatically resizes and optimizes images uploaded to an S3 bucket, storing the processed versions and making them available through a RESTful API.

Key Components

- S3: For storing original and processed images
- Lambda: For image processing logic
- API Gateway: To provide RESTful access to images
- DynamoDB: To store metadata about processed images

Implementation Highlights

Lambda Function (Python)

python

```python
import boto3
from PIL import Image
import io

s3 = boto3.client('s3')
dynamodb = boto3.resource('dynamodb')

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

    # Download the image from S3
    image_obj = s3.get_object(Bucket=bucket, Key=key)
    image_data = image_obj['Body'].read()

    # Open the image
    image = Image.open(io.BytesIO(image_data))

    # Resize the image
    resized_image = image.resize((300, 300))

    # Save the resized image
    buffer = io.BytesIO()
    resized_image.save(buffer, format="JPEG")
    buffer.seek(0)

    # Upload the resized image
    resized_key = f"resized-{key}"
    s3.put_object(Bucket=bucket, Key=resized_key, Body=buffer)

    # Update DynamoDB
    table = dynamodb.Table('ProcessedImages')
    table.put_item(
        Item={
            'original_key': key,
```

```
            'processed_key': resized_key,
            'size': '300x300'
        }
    )

    return {
        'statusCode': 200,
        'body': f"Processed {key} to {resized_key}"
    }
```

API Gateway Configuration

- Set up a GET method to retrieve processed images
- Integrated with Lambda to fetch image metadata from DynamoDB
- Configured appropriate IAM roles for Lambda execution and S3 access

S3 Event Trigger

Configured S3 to trigger the Lambda function on object creation events.

Challenges and Solutions

- Handling various image formats: Implemented logic to detect and process different image types.
- Optimizing Lambda execution: Adjusted memory and timeout settings for efficient processing.

Outcome

This project demonstrated my ability to design and implement serverless architectures, work with AWS Lambda, and integrate multiple AWS services to create a functional and scalable solution.