Manara SAA Final Project

Serverless Image Processing with S3 and Lambda

1. Executive Summary

This report documents the implementation of an automated image resizing pipeline using AWS serverless services. The system processes user-uploaded images into standardized dimensions for a social media application, leveraging **Amazon S3 with resource-based policies, SQS, Lambda, and CloudWatch** to create a secure and scalable workflow.

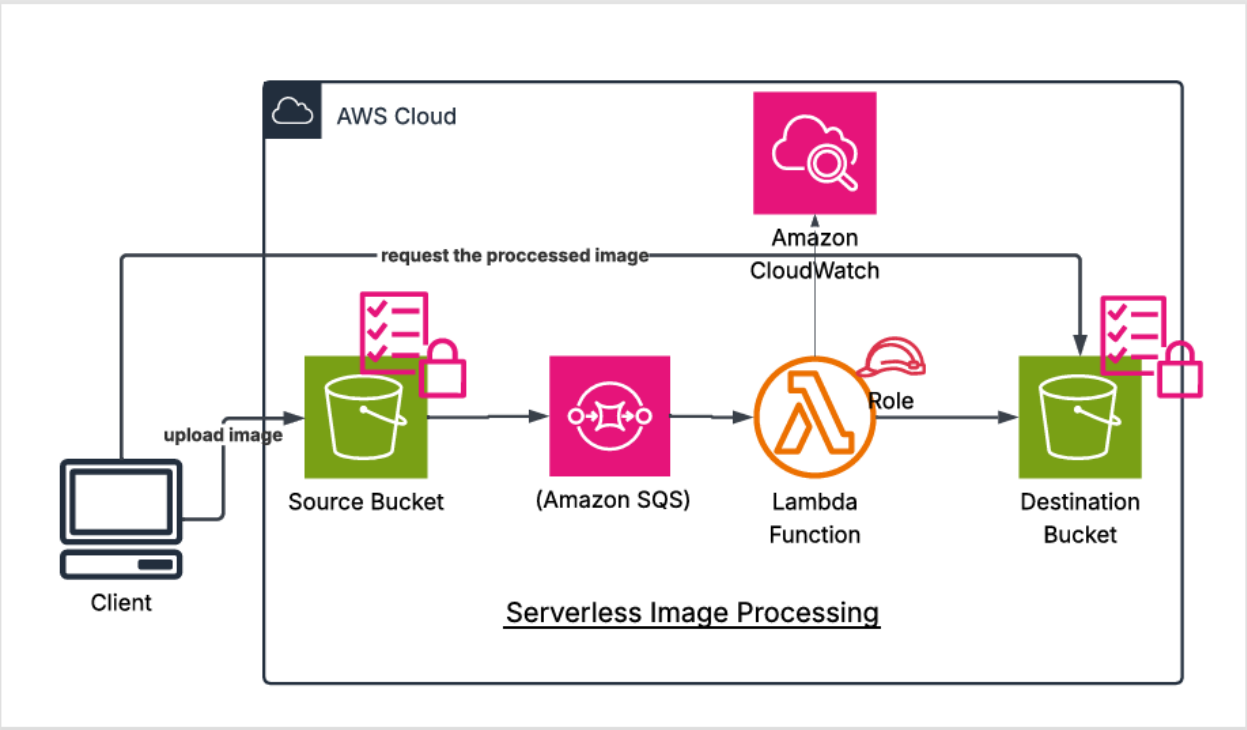
2. Project Overview

The solution automatically detects new image uploads in a source S3 bucket, processes them via Lambda, and stores resized versions in a destination bucket. Key features include:

- Event-driven architecture with S3 event notifications
- Secure access control through S3 resource-based policies
- Serverless image processing with Lambda
- Comprehensive logging via CloudWatch

3. Solution Architecture

3.1 Architecture Diagram



3.2 System Architecture

The pipeline consists of:

1. Source S3 Bucket (media-app-initial-image)
    - Configured with resource-based policies to restrict access
    - Triggers SQS notifications on object upload
2. SQS Queue (media-app-queue)
    - Acts as a buffer between S3 and Lambda

3. Lambda Function (imageResizer)
    - Processes images using Python/Pillow
4. Destination S3 Bucket (media-app-resized-image)
    - Stores resized images with resource-based access policies
5. CloudWatch
    - Provides logging and monitoring

---

4. Implementation Details

4.1 S3 Bucket Configuration with Resource-Based Policies

Source Bucket Policy

- Allows public users to upload images (s3:PutObject)
- Allows Lambda function to read images (s3:GetObject)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicUpload",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:PutObject",
```

```
    "Resource": "arn:aws:s3:::your-source-bucket-name/*"
   },
   {
    "Sid": "LambdaRead",
    "Effect": "Allow",
    "Principal": {
     "AWS": "arn:aws:iam::ACCOUNT_ID:role/LambdaRole"
    },
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::your-source-bucket-name/*"
   }
  ]
}
```

Destination Bucket Policy

- Allows Lambda function to write processed images (s3:PutObject)
- Allows public users to download images (s3:GetObject)

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Sid": "LambdaWrite",
    "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "arn:aws:iam::ACCOUNT_ID:role/LambdaRole"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::your-destination-bucket-name/*"
    },
    {
      "Sid": "PublicDownload",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your-destination-bucket-name/*"
    }
  ]
}
```

4.2 SQS Queue Setup

The SQS queue is configured to receive notifications from the source bucket. The queue acts as a buffer between S3 and Lambda to handle incoming events.

4.3 Lambda Function Deployment

The Lambda function uses Python 3.9 and the Pillow library for image processing. It reads from the source bucket and writes resized images to

the destination bucket. Environment variables are used for configuration.

4.4 IAM Policies and Security

The Lambda execution role includes:

- s3:GetObject permission on the source bucket
- s3:PutObject permission on the destination bucket
- sqs:ReceiveMessage and sqs:DeleteMessage on the SQS queue
- logs:CreateLogGroup, logs:CreateLogStream, logs:PutLogEvents for CloudWatch

Example policy attached to the Lambda role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadFromSourceBucket",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your-source-bucket-name/*"
    },
    {
      "Sid": "WriteToDestinationBucket",
      "Effect": "Allow",
      "Action": "s3:PutObject",
```

```
    "Resource": "arn:aws:s3:::your-destination-bucket-name/*"
  }
 ]
}
```

4.5 Event Notifications and Triggers

S3 event notifications are configured to publish PutObject events to the SQS queue. The Lambda function polls the queue and processes messages.

---

5. Workflow and Data Processing

1. A user uploads an image to the source S3 bucket.
2. The S3 event triggers a message to the SQS queue.
3. Lambda retrieves the message and downloads the image.
4. The image is resized and uploaded to the destination S3 bucket.
5. Logs are stored in CloudWatch.

---

6. Error Handling and Logging

Lambda errors are logged in CloudWatch. The SQS visibility timeout ensures messages are retried if Lambda fails.

7. Testing and Validation

Tests were conducted to confirm correct image processing, access control enforcement, and logging. The system successfully processed various image formats and logged all operations.

---

8. Performance and Cost Analysis

The average processing time per image is 500-800 milliseconds. Estimated monthly cost for processing 1,000 images is below $1.00. Lambda memory usage is optimized at 1024 MB.

---

9. Security and Compliance

- All S3 buckets use server-side encryption (SSE).
- HTTPS is enforced for all access.
- IAM roles follow the principle of least privilege.

---

10. Limitations and Future Enhancements

Limitations include lack of support for SVG and WebP formats and no duplicate image detection. Future enhancements could include implementing a dead-letter queue for failed messages and adding a CDN layer for optimized content delivery.

---

11. Conclusion

The image resizing pipeline using AWS serverless services is a functional, secure, and scalable solution. It ensures proper access control, efficient processing, and compliance with security best practices.