Koneru Lakshmaiah Education Foundation

(Deemed to be University)

A Project Based Lab Report

On

Monitoring and Logging System with Amazon CloudWatch

BACHELOR OF TECHNOLGY

SUBMITTED BY

Bodepudi Jothirmaye

2200032386

DEPARTMENT OFCOMPUTER SCIENCE AND ENGINEERING

KONERU LAKSHMAIAH EDUCATION FOUNDATION

Green Fields, Vaddeswaram,
Guntur, Andhra Pradesh, 522 302
INDIA

ACKNOWLEDGEMENTS

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project based laboratory report.

I express the sincere gratitude to our principal **Dr. A. Jagadeesh** for his administration towards our academic growth.

I express sincere gratitude to HOD-CSE **Prof. VSV Prabhakar** for his leadership and constant motivation provided in successful completion of our academic semester. I record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our project supervisor Mrs. Sunitha Bandeela for her novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

ABSTRACT

Serverless image processing with AWS Lambda and S3 offers a highly scalable, cost-effective, and automated approach to handling image transformations in the cloud. By utilizing AWS Lambda, images can be processed dynamically in response to S3 events, eliminating the need for server management. Integration with additional AWS services, such as Amazon SNS and Amazon CloudWatch, enhances workflow automation, monitoring, and alerting.

This method is particularly beneficial for real-time applications such as thumbnail generation, watermarking, and format conversion. It optimizes infrastructure usage, reduces operational costs, and improves scalability. However, considerations such as Lambda execution time, cold start mitigation, and secure access management are crucial for efficient implementation.

Overall, serverless image processing enables businesses to achieve high availability, fault tolerance, and enhanced performance, making it an ideal choice for modern cloud-based applications.

INDEX

PROJECT AIM

In this project, you will create a serverless image processing pipeline that automatically processes images uploaded to Amazon S3. When a user uploads an image to an S3 bucket, the system will trigger an AWS Lambda function that performs tasks such as resizing the image, compressing it, or applying filters.

You will use Amazon S3 to store both the original and processed images, and AWS Lambda will scale automatically based on the number of incoming image uploads. Additionally, you will integrate Amazon SNS to send notifications to the user once the image processing is complete.

This project will teach you how to leverage AWS's serverless capabilities to build an efficient, scalable image processing system.

SERVICES USED:

- ✓ Amazon S3
- ✓ AWS Lambda
- ✓ AWS IAM
- ✓ Amazon SNS(simple notification services)

DESCRIPTION:

Amazon S3: AWS S3 (Simple Storage Service) is a cloud data storage service. It is one of the most popular services of AWS. It has high scalability, availability, security and is cost effective. S3 has different storage tiers depending on the use case. Some common use cases of AWS S3 are:

1. **Storage:** It can be used for storing large amounts of data.
2. **Backup and Archive:** S3 has different storage tiers based on how frequent the data is accessed which can be used to backup critical data at low costs.
3. **Static website:** S3 offers static website hosting through HTML files stored in S3.

4. **Data lakes and big data analytics:** Companies can use AWS S3 as a data lake and then run analytics on it for getting business insights and take critical decisions.

AWS Lambda:

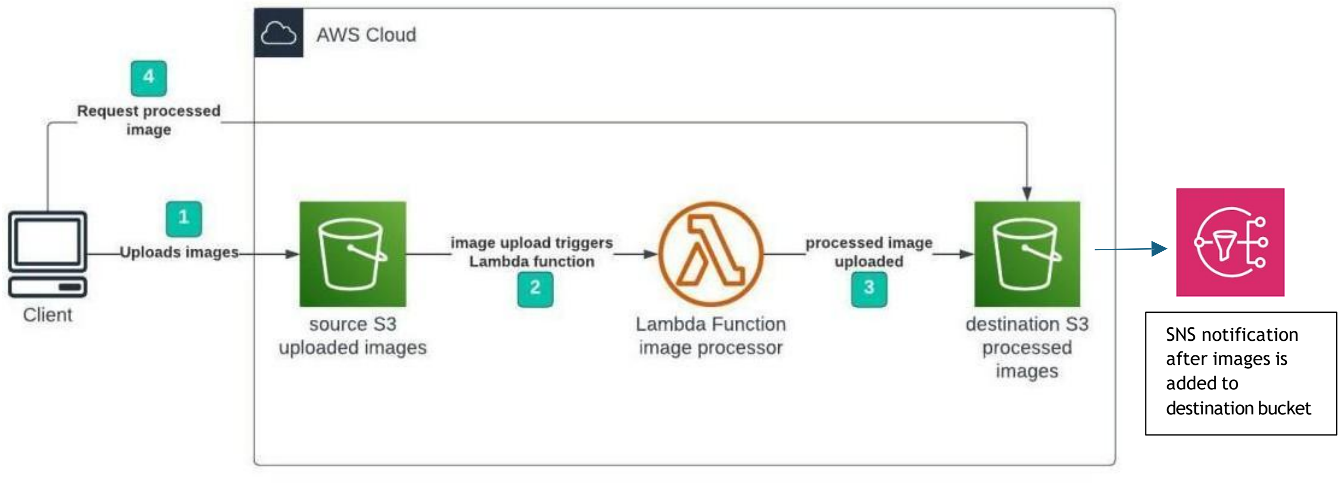
AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. Lambda functions run on demand i.e. they execute only when needed and you pay only for what you compute. Lambda is well integrated with may other AWS services. It supports a wide variety of programming languages.

Some common use cases for AWS Lambda are:

1. **File processing:** You can use Lambda for processing files as they are uploaded in an S3 bucket or whenever some event triggers the function.
2. **Data and analytics:** You can pass a data stream to your Lambda function and then create analysis from that.
3. **Website:** Lambda can also be used for creating websites. This is cost effective because you are charged only for the time when the servers are running.

Serverless Image Processing Flow

1. User uploads a file to the source S3 bucket (which is used for storing uploaded images).
2. When the image is uploaded to a source S3 bucket, it triggers an event which invokes the Lambda function. The lambda function processes the image.
3. Processed image is stored in the destination S3 bucket.
4. The processed image is requested by the user.


AWS Cloud
4
Request processed image
1
Uploads images
Client
source S3 uploaded images
image upload triggers Lambda function
2
Lambda Function image processor
processed image uploaded
3
destination S3 processed images
SNS notification after images is added to destination bucket

Serverless Image Processor

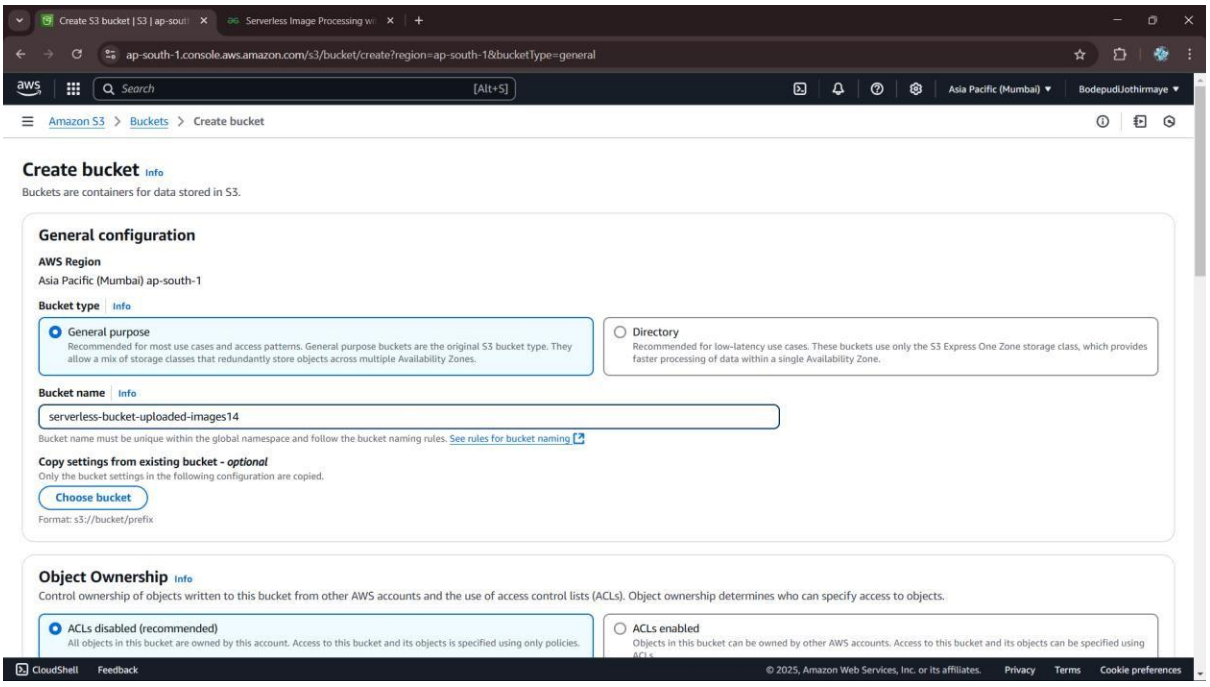
PROCEDURE:

Step 1 - Creating S3 buckets

We will use two S3 buckets:

1. **source Bucket:** For storing uploaded images.
2. **destination Bucket:** For storing processed images.

Go to S3 console and click Create bucket. Enter bucket name as 'serverless-bucket-uploaded- images'. Choose any AWS region as 'ap-south-1'.


Create S3 bucket | S3 | ap-south
Serverless Image Processing wi
ap-south-1.console.aws.amazon.com/s3/bucket/create?region=ap-south-1&bucketType=general
Search
[Alt+S]
Asia Pacific (Mumbai)
BodepudiJothirmaye
Amazon S3 > Buckets > Create bucket
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

Asia Pacific (Mumbai) ap-south-1

Bucket type Info

[✓ General purpose] Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory — Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

serverless-bucket-uploaded-images14

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

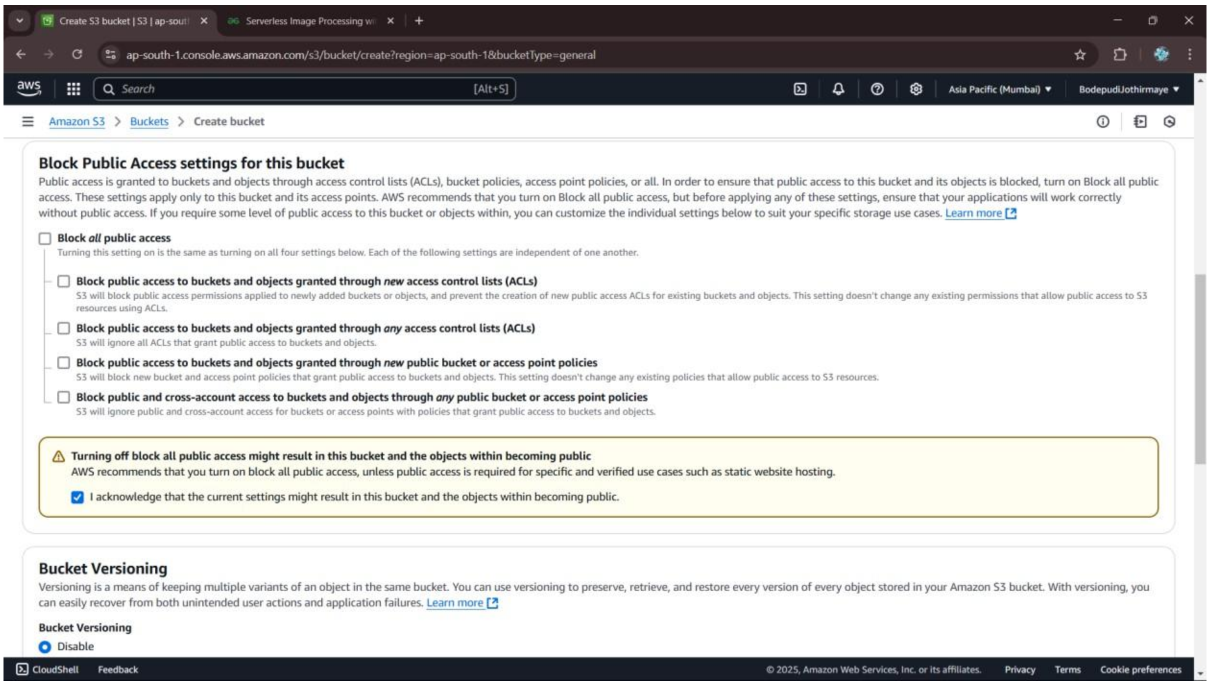
[✓ ACLs disabled (recommended)] All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled — Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Step 2 - Configuring S3 bucket policy

In 'Block Public Access settings for this bucket' section disable "block all public access". You will get a warning that the bucket and its objects might become public. Agree to the warning**. (<u>Note:</u> we are making this bucket public only for this project, it is not recommended to make an S3 bucket public if not needed)**.

Leave all other settings as default and create bucket. Similarly, create another bucket named 'serverless-bucket-processed-images' with the same region. This bucket will be used to store the processed images. Although we enabled public access while creating the buckets, we still need to attach a bucket policy to access the objects stored in it. (Policies in AWS are JSON documents which defines the permissions for performing actions on a certain resource.)


Create S3 bucket | S3 | ap-south
Serverless Image Processing w
ap-south-1.console.aws.amazon.com/s3/bucket/create?region=ap-south-1&bucketType=general
aws
Search
[Alt+S]
Asia Pacific (Mumbai)
BodepudiJothirmaye
Amazon S3 > Buckets > Create bucket

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
  S3 will ignore all ACLs that grant public access to buckets and objects.
- ☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
  S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

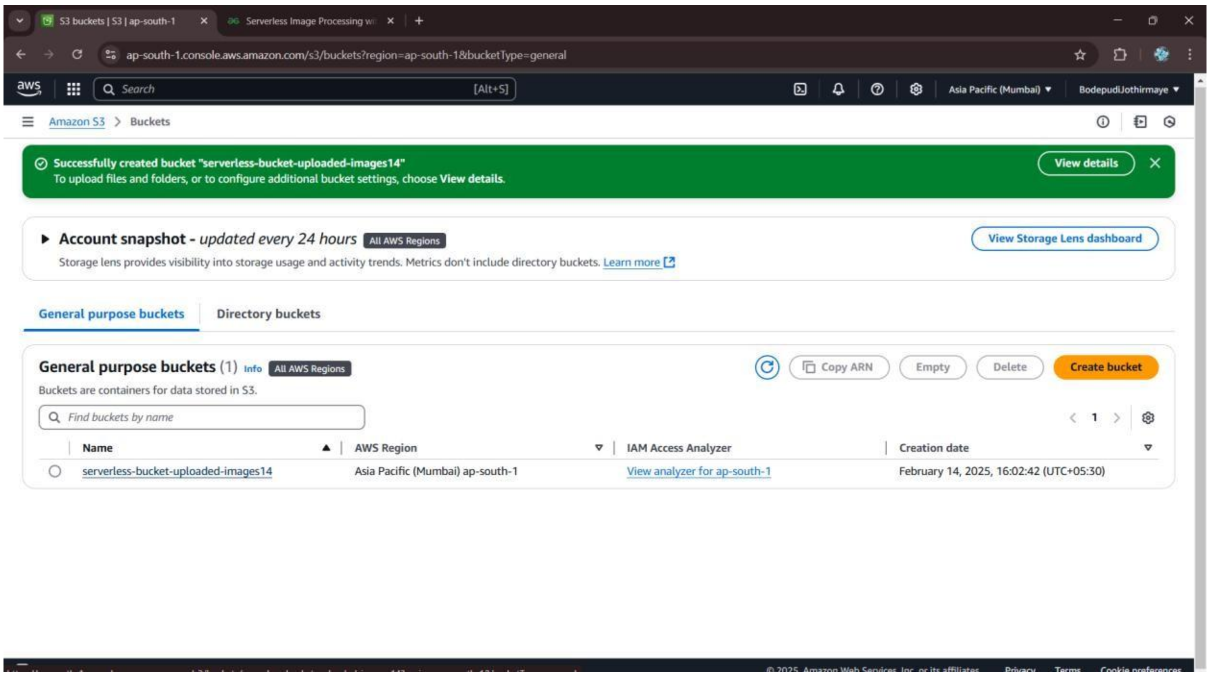
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

✓ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
● Disable



Successfully created bucket "serverless-bucket-uploaded-images14"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

Account snapshot - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more

General purpose buckets (1) Info All AWS Regions
Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
serverless-bucket-uploaded-images14	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	February 14, 2025, 16:02:42 (UTC+05:30)

Go to your source bucket and then click on Permissions tab. Scroll to Bucket Policy tab. Click Edit. You will be redirected to the policy editor. Click on policy generator.

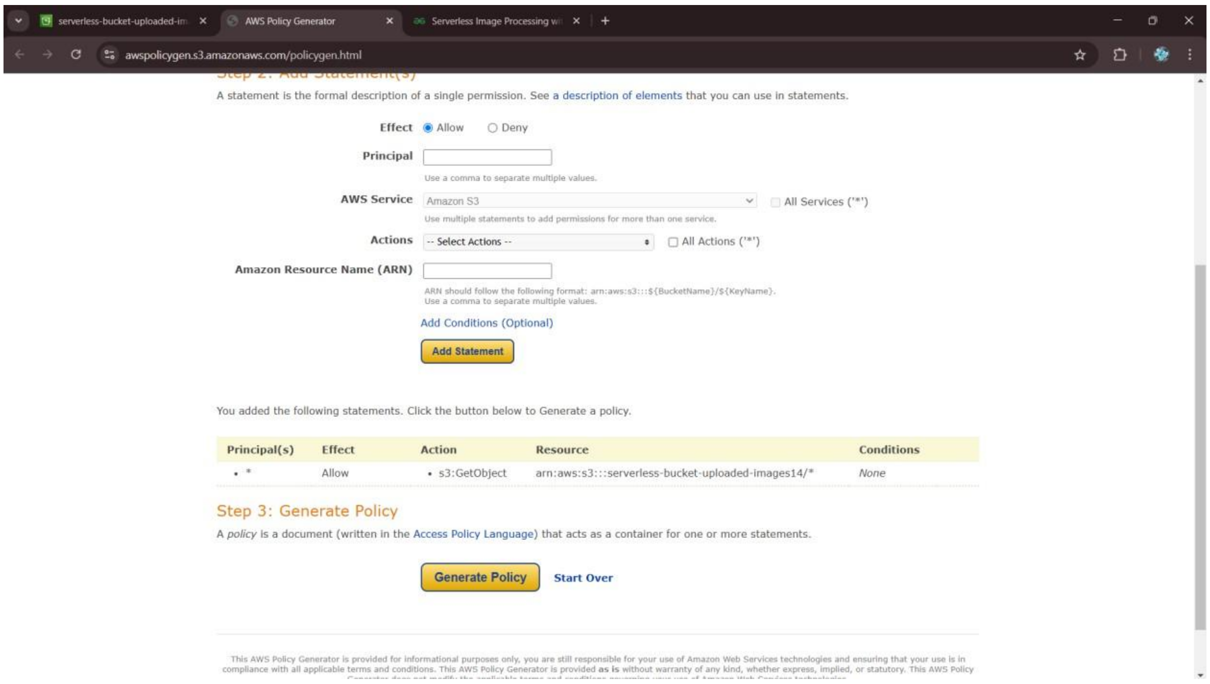
Enter the following settings:

- Type of policy: S3 Bucket Policy
- Effect:Allow
- Principal: *
- Actions: GetObject
- Amazon Resource Name (ARN):

arn:aws:s3:::SOURCE_BUCKET_NAME/*

SOURCE_BUCKET_NAME is the name of the bucket used for uploading the images.

Click Add Statement and then generate policy. Copy the JSON object.


A statement is the formal description of a single permission. See a description of elements that you can use in statements.

Effect	[✓ Allow] Deny
Principal	
AWS Service	Amazon S3 — All Services ('*')
Actions	-- Select Actions -- — All Actions ('*')
Amazon Resource Name (ARN)	

Use a comma to separate multiple values.
Use multiple statements to add permissions for more than one service.
ARN should follow the following format: arn:aws:s3:::${BucketName}/${KeyName}.
Use a comma to separate multiple values.

Add Conditions (Optional)

You added the following statements. Click the button below to Generate a policy.

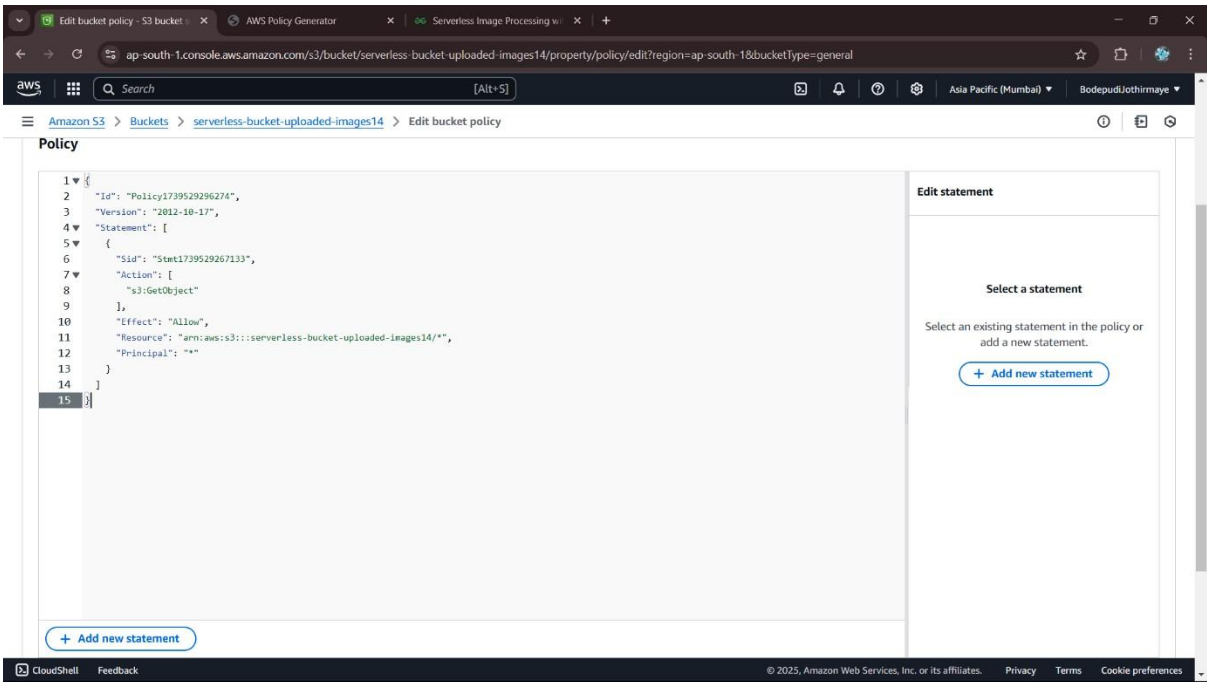
Principal(s)	Effect	Action	Resource	Conditions
• *	Allow	• s3:GetObject	arn:aws:s3:::serverless-bucket-uploaded-images14/*	None

Step 3: Generate Policy

A *policy* is a document (written in the Access Policy Language) that acts as a container for one or more statements.

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided **as is** without warranty of any kind, whether express, implied, or statutory. This AWS Policy

Paste it in the policy editor and then save changes.


Amazon S3 > Buckets > serverless-bucket-uploaded-images14 > Edit bucket policy

Policy

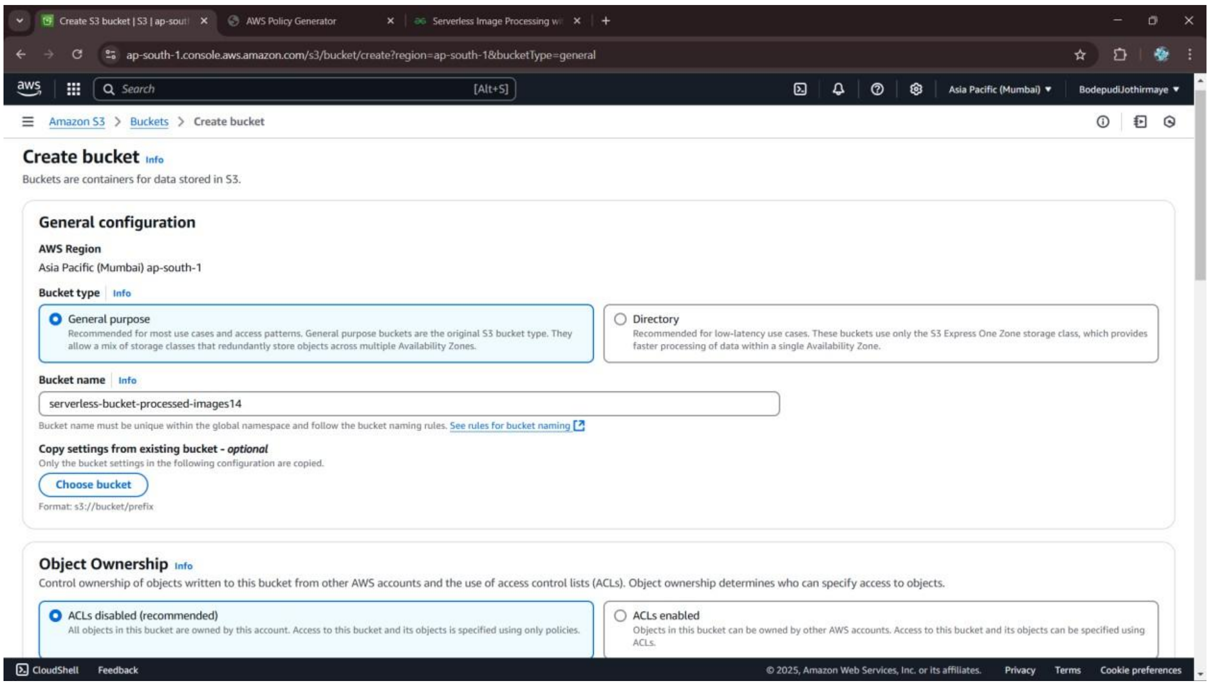
```
{
  "Id": "Policy1739529296274",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1739529267133",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::serverless-bucket-uploaded-images14/*",
      "Principal": "*"
    }
  ]
}
```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

Repeat the above steps and create other Bucket named serverless-bucket-processed-images



Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

Asia Pacific (Mumbai) ap-south-1

Bucket type Info

[✓ General purpose] Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory: Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

serverless-bucket-processed-images14

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Object Ownership Info

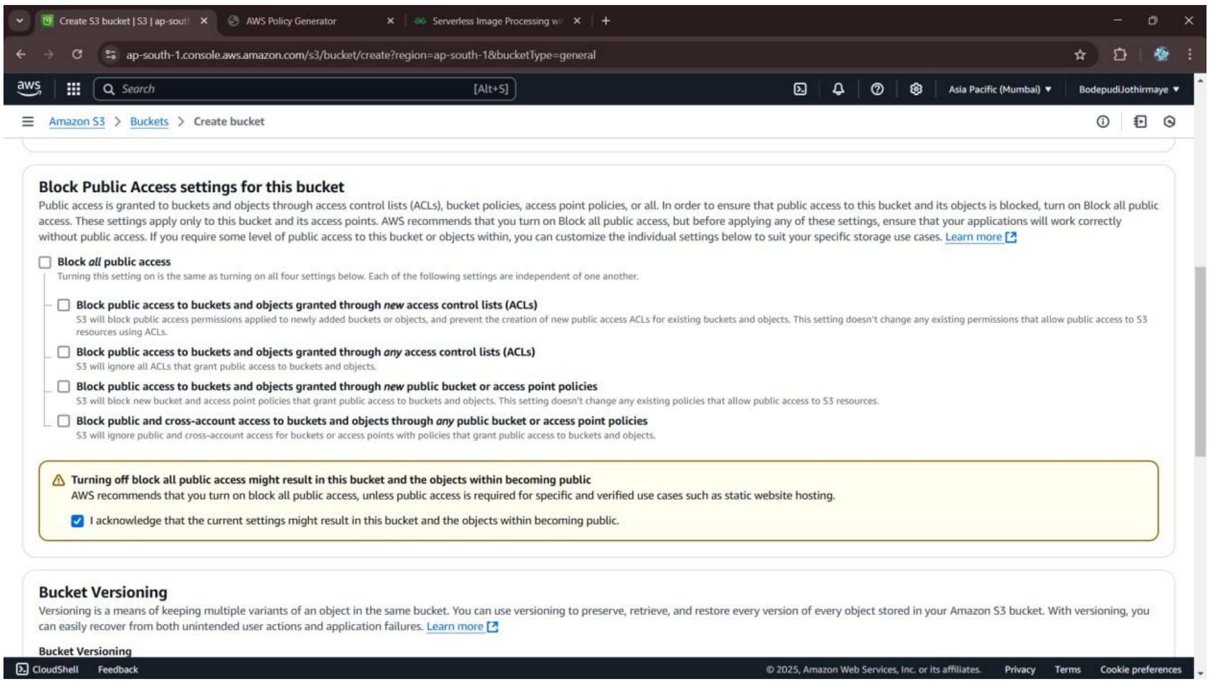
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

[✓ ACLs disabled (recommended)] All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled: Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

© 2025, Amazon Web Services, Inc. or its affiliates.

Create S3 bucket | S3 | ap-sout
AWS Policy Generator
Serverless Image Processing w
ap-south-1.console.aws.amazon.com/s3/bucket/create?region=ap-south-1&bucketType=general
Search [Alt+S]
Asia Pacific (Mumbai)
BodepudiJothirmaye
Choose bucket
CloudShell
Feedback
Privacy
Terms
Cookie preferences



Amazon S3 > Buckets > Create bucket

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- [ ] Block *all* public access
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - [ ] Block public access to buckets and objects granted through *new* access control lists (ACLs)
    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - [ ] Block public access to buckets and objects granted through *any* access control lists (ACLs)
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - [ ] Block public access to buckets and objects granted through *new* public bucket or access point policies
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - [ ] Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

- [x] I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

© 2025, Amazon Web Services, Inc. or its affiliates.

Create S3 bucket | S3 | ap-sout
AWS Policy Generator
Serverless Image Processing w
ap-south-1.console.aws.amazon.com/s3/bucket/create?region=ap-south-1&bucketType=general
Search [Alt+S]
Asia Pacific (Mumbai)
BodepudiJothirmaye
CloudShell
Feedback
Privacy
Terms
Cookie preferences

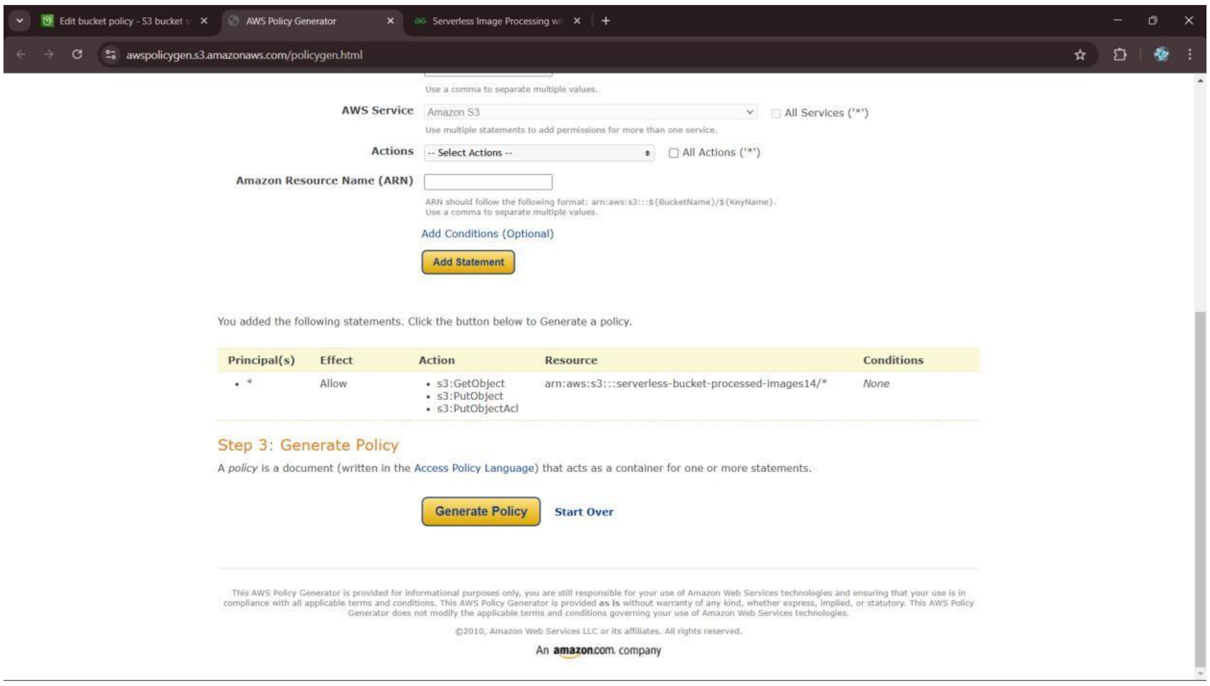
Paste it in the policy editor and then save changes.

Follow same steps to attach a policy to the processed images S3 bucket. The policy settings for destination bucket are:

- Type of policy: S3 Bucket Policy
- Effect:Allow

- Principal:*
- Actions: GetObject, PutObject, and PutObjectAcl
- Amazon Resource Name (ARN): arn:aws:s3:::DESTINATION_BUCKET_NAME/*

DESTINATION_BUCKET_NAME is the name of the bucket used for storing processed images.



Edit bucket policy - S3 bucket
AWS Policy Generator
Serverless Image Processing w
awspolicygen.s3.amazonaws.com/policygen.html

Use a comma to separate multiple values.

AWS Service: Amazon S3 ☐ All Services ('*')

Use multiple statements to add permissions for more than one service.

Actions: -- Select Actions -- ☐ All Actions ('*')

Amazon Resource Name (ARN):

ARN should follow the following format: arn:aws:s3:::${BucketName}/${KeyName}.
Use a comma to separate multiple values.

Add Conditions (Optional)
Add Statement

You added the following statements. Click the button below to Generate a policy.

Principal(s)	Effect	Action	Resource	Conditions
• *	Allow	• s3:GetObject • s3:PutObject • s3:PutObjectAcl	arn:aws:s3:::serverless-bucket-processed-images14/*	*None*

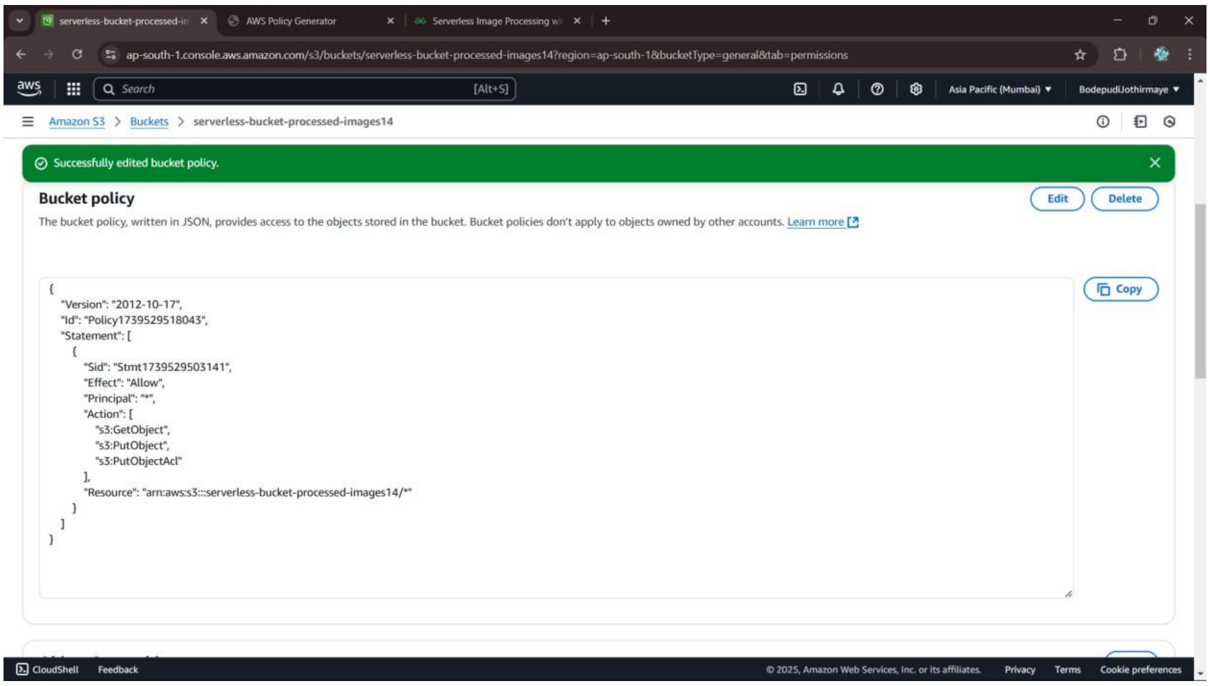
Step 3: Generate Policy

A *policy* is a document (written in the Access Policy Language) that acts as a container for one or more statements.

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided **as is** without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

©2010, Amazon Web Services LLC or its affiliates. All rights reserved.

An amazon.com company



serverless-bucket-processed-in
AWS Policy Generator
Serverless Image Processing w
ap-south-1.console.aws.amazon.com/s3/buckets/serverless-bucket-processed-images14?region=ap-south-1&bucketType=general&tab=permissions
aws
Search [Alt+S]
Asia Pacific (Mumbai)
BodepudiJothirmaye
Amazon S3 > Buckets > serverless-bucket-processed-images14

Successfully edited bucket policy.

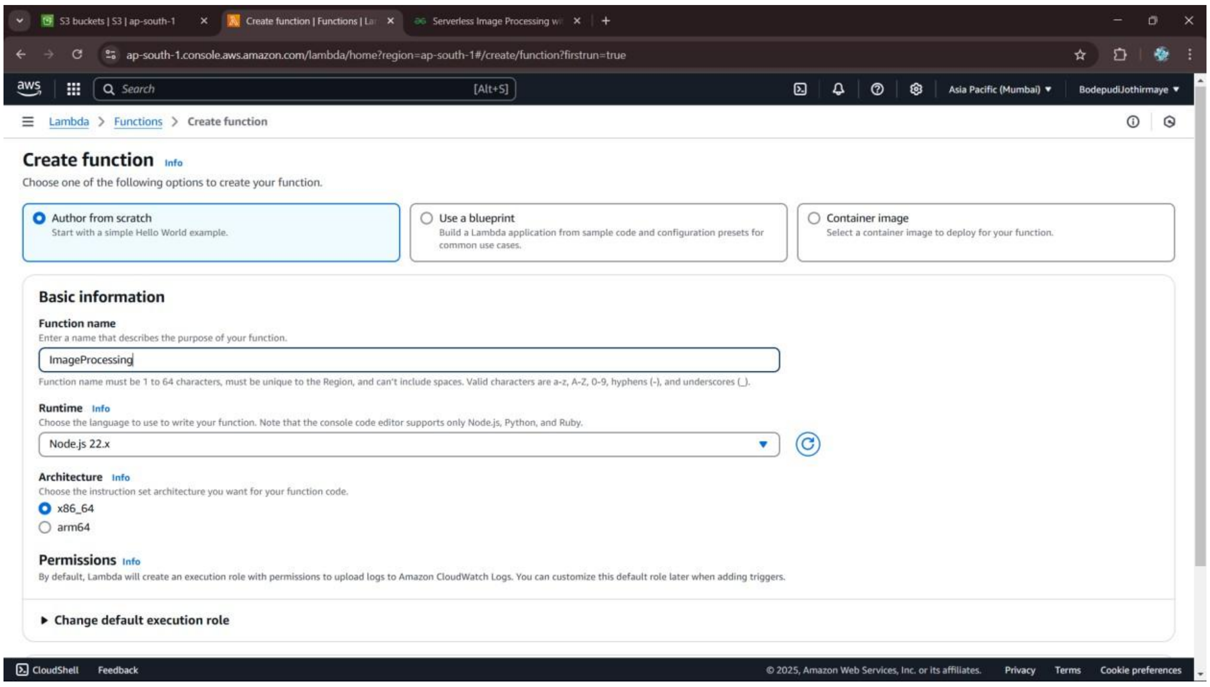
Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

```
{
  "Version": "2012-10-17",
  "Id": "Policy1739529518043",
  "Statement": [
    {
      "Sid": "Stmt1739529503141",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::serverless-bucket-processed-images14/*"
    }
  ]
}
```

Step 3 - Creating Lambda function

Go to AWS Lambda console. Navigate to Functions section. Click Create Function and name it "ImageProcessing". Select runtime as "NodeJS 22.x" and architecture as "x86_64". Leave all other settings as default. Create the function.



Lambda > Functions > Create function

Create function

Choose one of the following options to create your function.

[✓ Author from scratch] Start with a simple Hello World example.

Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.

Container image: Select a container image to deploy for your function.

Basic information

Function name: ImageProcessing

Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime: Node.js 22.x

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture: [✓ x86_64] arm64

Choose the instruction set architecture you want for your function code.

Permissions

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

In the code editor on the Lambda function page paste the following code. This function is executed whenenver an image is uploaded to our source S3 bucket and creates two images (thumbnail (300x300) and coverphoto(800x800)) and stores it in the destination S3 bucket. **(Note: The value of processedImageBucket in the code should be set to the name of the destination bucket).**

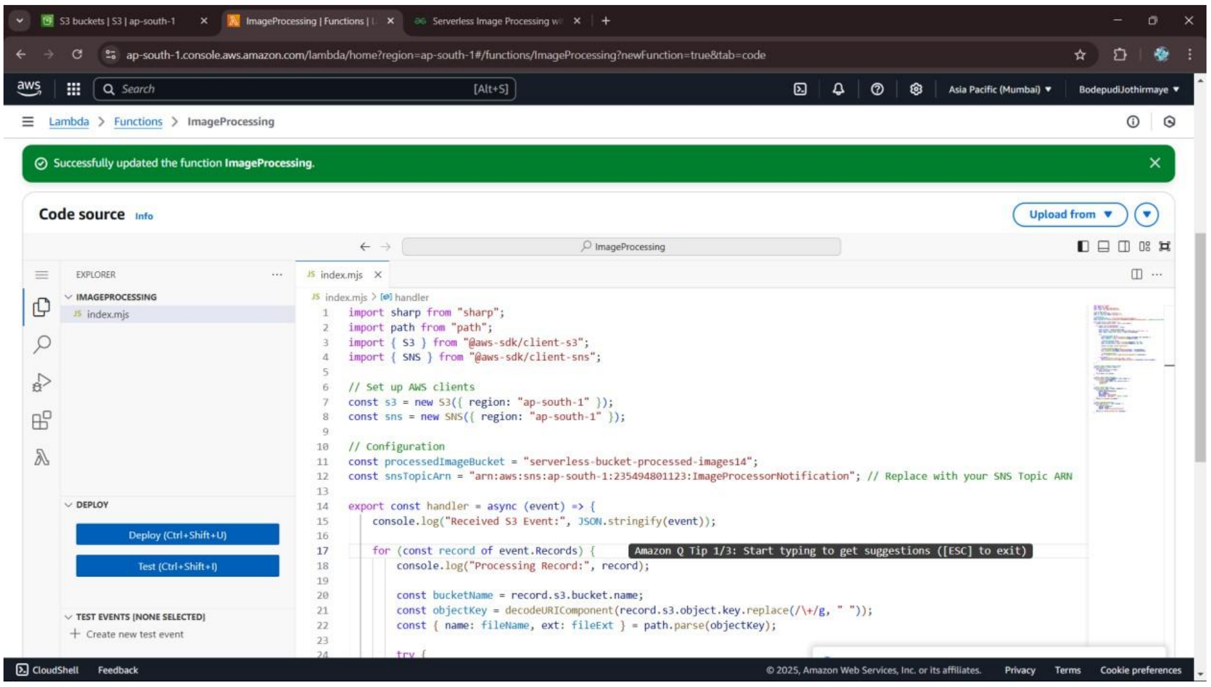
CODE:

```
import sharp from "sharp";

import path from "path";

import { S3 } from "@aws-sdk/client-s3";

import { SNS } from "@aws-sdk/client-sns";

// Set up AWS clients

const s3 = new S3({ region: "ap-south-1" });

const sns = new SNS({ region: "ap-south-1" });
```

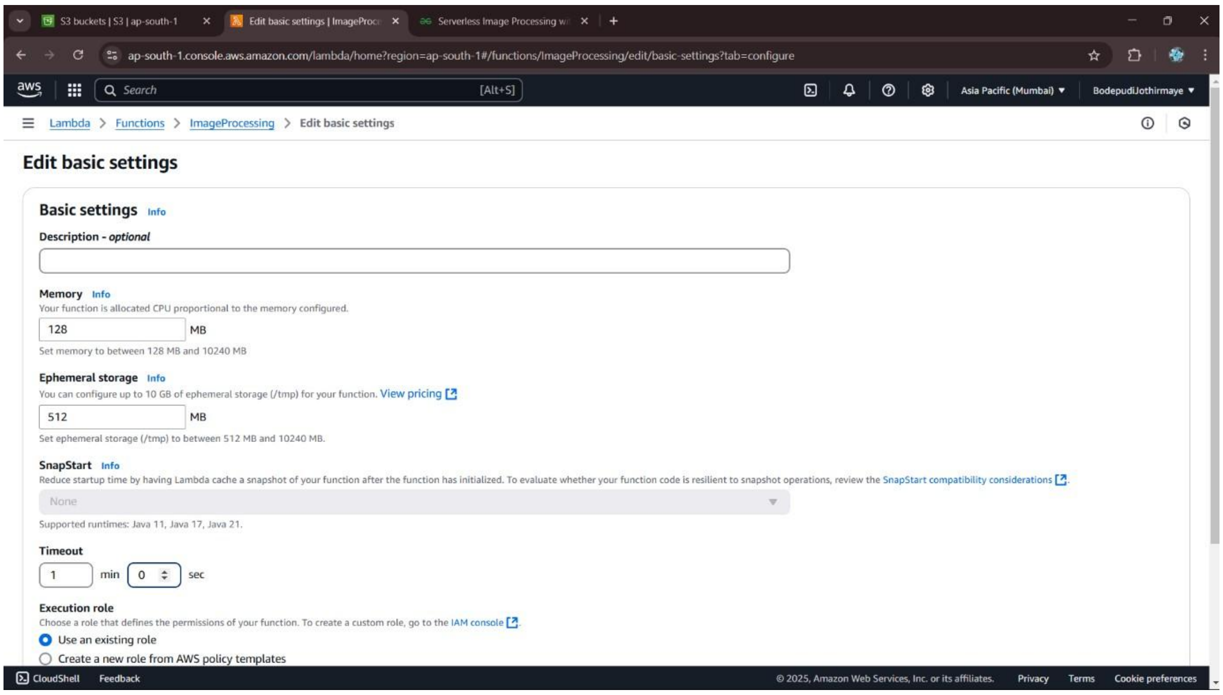
```
// Configuration
const processedImageBucket = "serverless-bucket-processed-images14";
const snsTopicArn = "arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification"; //
Replace with your SNS Topic ARN


export const handler = async (event) => {
  console.log("Received S3 Event:", JSON.stringify(event));


  for (const record of event.Records) {
    console.log("Processing Record:", record);


    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, " "));
    const { name: fileName, ext: fileExt } = path.parse(objectKey);


    try {
      // Fetch the image from S3
      const imageObject = await s3.getObject({ Bucket: bucketName, Key: objectKey });
      const imageBuffer = await streamToBuffer(imageObject.Body);


      // Generate processed images
      const resizedThumbnail = await resizeImage(imageBuffer, 300, 300);
      const resizedCoverPhoto = await resizeImage(imageBuffer, 800, 800);


      console.log("Images resized successfully");


      // Upload processed images to S3
      await uploadToS3(`${fileName}_thumbnail${fileExt}`, resizedThumbnail);
      await uploadToS3(`${fileName}_coverphoto${fileExt}`, resizedCoverPhoto);


      // Send SNS Notification
```

```
        await sendSNSNotification(`Image processing completed for ${objectKey}`);

      } catch (error) {
        console.error("Error processing image:", error);
        await sendSNSNotification(`Image processing failed for ${objectKey}: ${error.message}`);
      }
    }
};

// Helper: Convert readable stream to buffer
const streamToBuffer = async (stream) => {
    const chunks = [];
    for await (const chunk of stream) {
      chunks.push(chunk);
    }
    return Buffer.concat(chunks);
};

// Helper: Resize image using Sharp
const resizeImage = async (imageBuffer, width, height) => {
    return await sharp(imageBuffer)
      .resize({ width, height, fit: sharp.fit.cover })
      .withMetadata()
      .toBuffer();
};

// Helper: Upload image to S3
const uploadToS3 = async (fileName, imageBuffer) => {
    await s3.putObject({
      Bucket: processedImageBucket,
      Key: fileName,
        Body: imageBuffer,
        CacheControl: "max-age=3600",
        ContentType: "image/jpeg", // Modify if needed
    });
    console.log(`Uploaded: ${fileName}`);
};

// Helper: Send SNS Notification
const sendSNSNotification = async (message) => {
    await sns.publish({
        TopicArn: snsTopicArn,
        Message: message,
        Subject: "Image Processing Notification",
    });
    console.log(`SNS Notification Sent: ${message}`);
};
```

Save and click deploy changes.


Lambda > Functions > ImageProcessing
Successfully updated the function ImageProcessing.
Code source Info
Upload from
EXPLORER
IMAGEPROCESSING
index.mjs
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)

```
import sharp from "sharp";
import path from "path";
import { S3 } from "@aws-sdk/client-s3";
import { SNS } from "@aws-sdk/client-sns";

// Set up AWS clients
const s3 = new S3({ region: "ap-south-1" });
const sns = new SNS({ region: "ap-south-1" });

// Configuration
const processedImageBucket = "serverless-bucket-processed-images14";
const snsTopicArn = "arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification"; // Replace with your SNS Topic ARN

export const handler = async (event) => {
    console.log("Received S3 Event:", JSON.stringify(event));

    for (const record of event.Records) {
        console.log("Processing Record:", record);

        const bucketName = record.s3.bucket.name;
        const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, " "));
        const { name: fileName, ext: fileExt } = path.parse(objectKey);

        try {
```



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/ImageProcessing/edit/basic-settings?tab=configure

Lambda > Functions > ImageProcessing > Edit basic settings

Edit basic settings

Basic settings Info

Description - *optional*

Memory Info
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing
512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.
None
Supported runtimes: Java 11, Java 17, Java 21.

Timeout
1 min 0 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
● Use an existing role
○ Create a new role from AWS policy templates

© 2025, Amazon Web Services, Inc. or its affiliates.

Go to Configuration tab and Edit the general configuration. There set the timeout to 1 min (timeout is the maximum time for which a Lambda function will run after which it stops running). We need to increase the timeout because the image can take time to process. Click on Save changes.

Step 4 - Creating Lambda layer and attaching it to Lambda function

Layers in Lambda is used to add dependencies to a Lambda Function. Lambda Layers reduces the code size of Lambda functions as we do not need to upload the dependencies with the function. It also useful for code reusability as we can reuse the layer with multiple functions if they require the same dependencies.

First we need to create a zip file with all the dependencies (node modules in our case) required by our Lambda function.

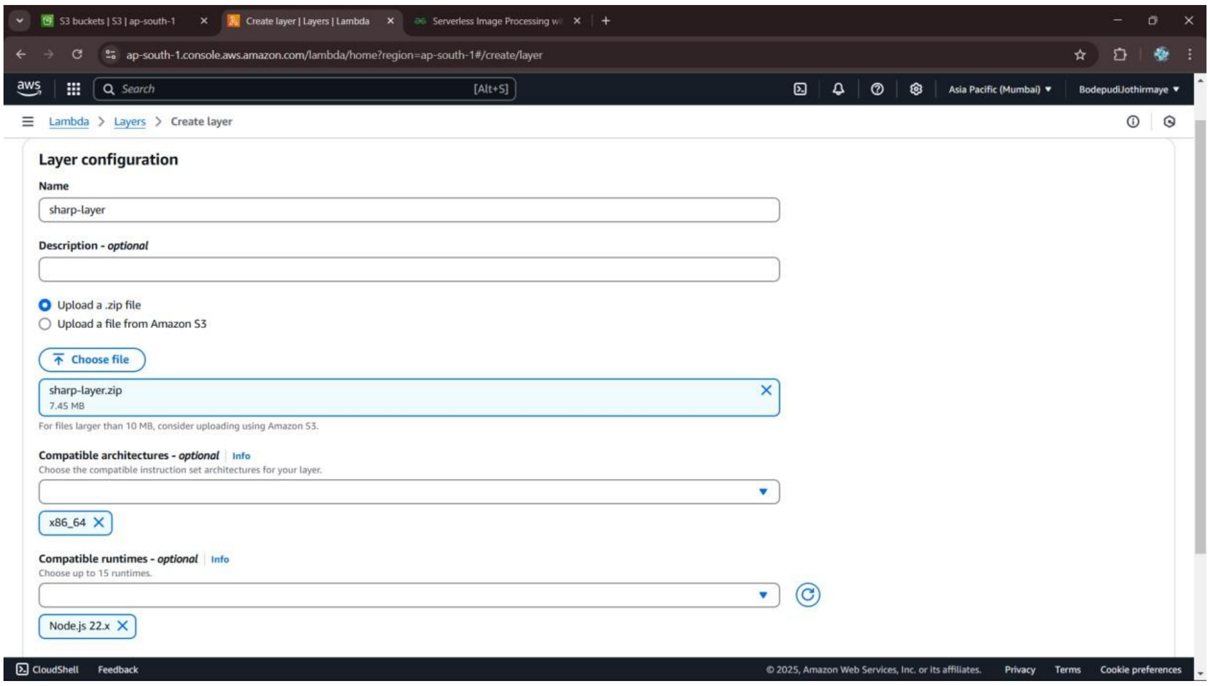
Create a folder "aws-serverless-image-processor". Inside this directory create another directory "nodejs" (it is compulsory to name this as "nodejs"). Open a terminal an go to nodejs directory. Install *sharp* module with the following command (platform is linux because the Lambda function runs on a Linux machine so we require the node_modules for Linux).

```
npm install --arch=x64 --platform=linux sharp
```

Now create a zip file of the nodejs directory and name it "sharplayer.zip".

Go to Layers in Lambda console. Click Create layer. Name it "sharp-layer".

Upload your nodejs "sharplayer.zip" file here. Select x86_64 architecture. Select NodeJS 22.x in compatible runtimes. Click on Create Layer.



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/create/layer

Lambda > Layers > Create layer

Layer configuration

Name: sharp-layer

Description - *optional*

● Upload a .zip file
○ Upload a file from Amazon S3

sharp-layer.zip
7.45 MB

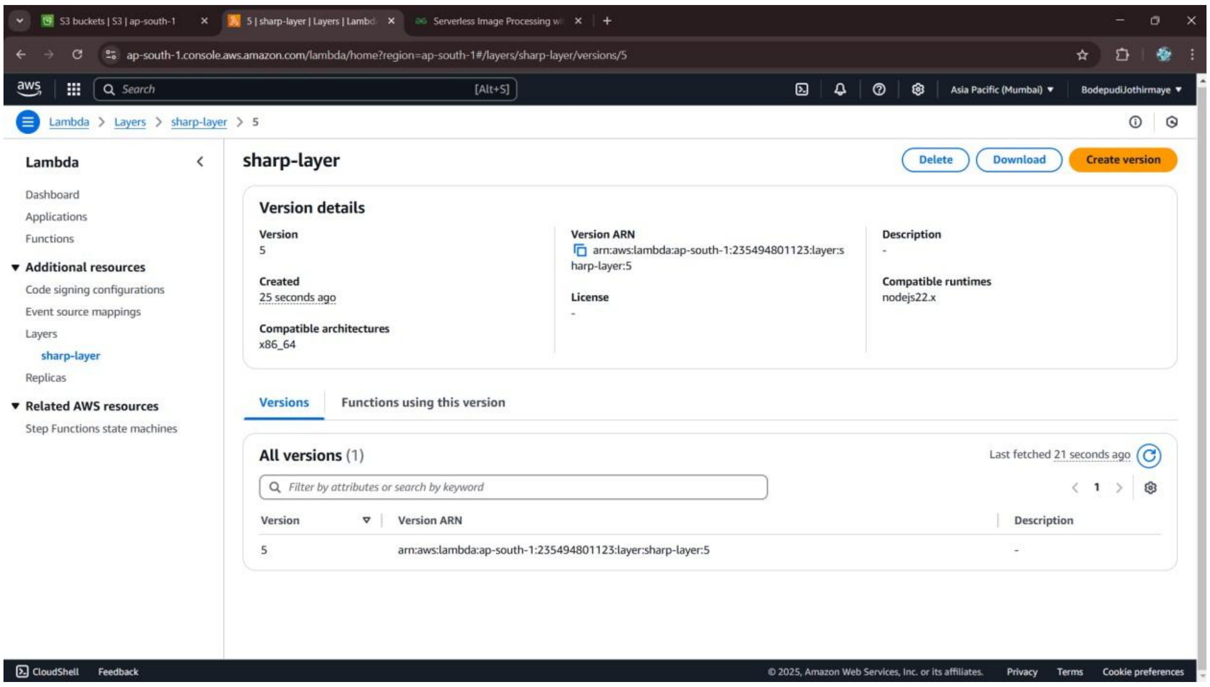
For files larger than 10 MB, consider uploading using Amazon S3.

Compatible architectures - *optional*
Choose the compatible instruction set architectures for your layer.

x86_64

Compatible runtimes - *optional*
Choose up to 15 runtimes.

Node.js 22.x



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/layers/sharp-layer/versions/5

Lambda > Layers > sharp-layer > 5

sharp-layer

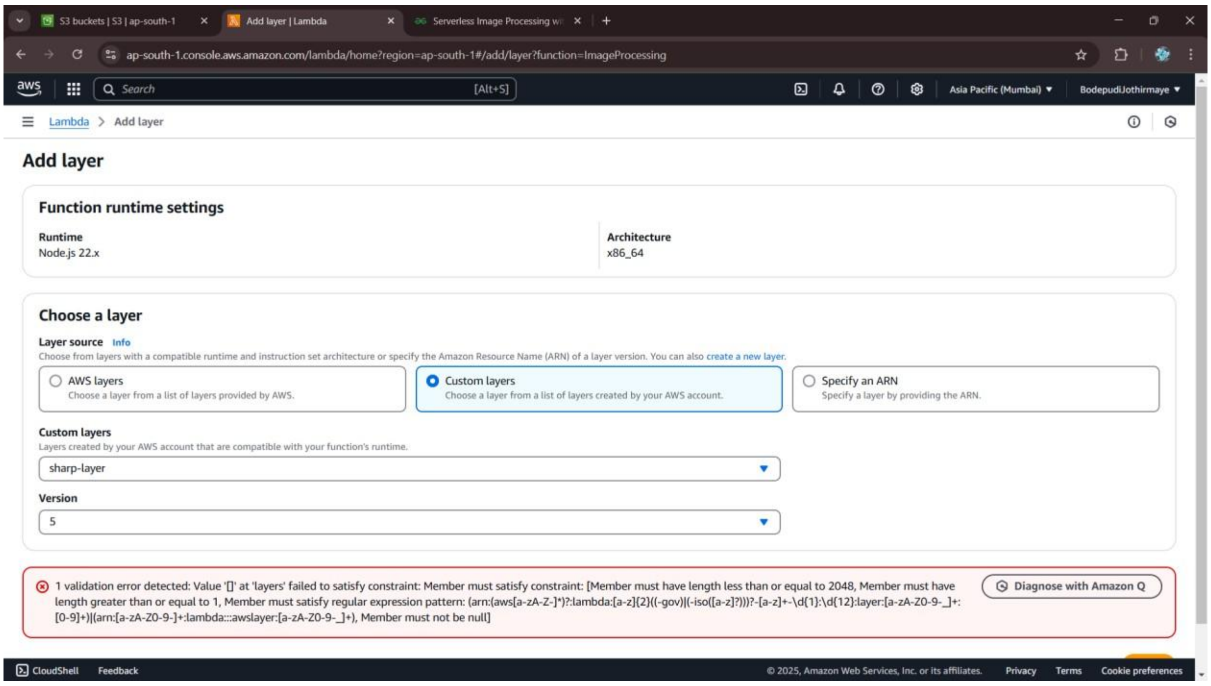
Version details

Version: 5	Version ARN: arn:aws:lambda:ap-south-1:235494801123:layer:sharp-layer:5	Description: -
Created: 25 seconds ago	License: -	Compatible runtimes: nodejs22.x
Compatible architectures: x86_64		

All versions (1)

Last fetched 21 seconds ago

Version	Version ARN	Description
5	arn:aws:lambda:ap-south-1:235494801123:layer:sharp-layer:5	-


Add layer

Function runtime settings

Runtime: Node.js 22.x

Architecture: x86_64

Choose a layer

Layer source: Custom layers

Custom layers: sharp-layer

Version: 5

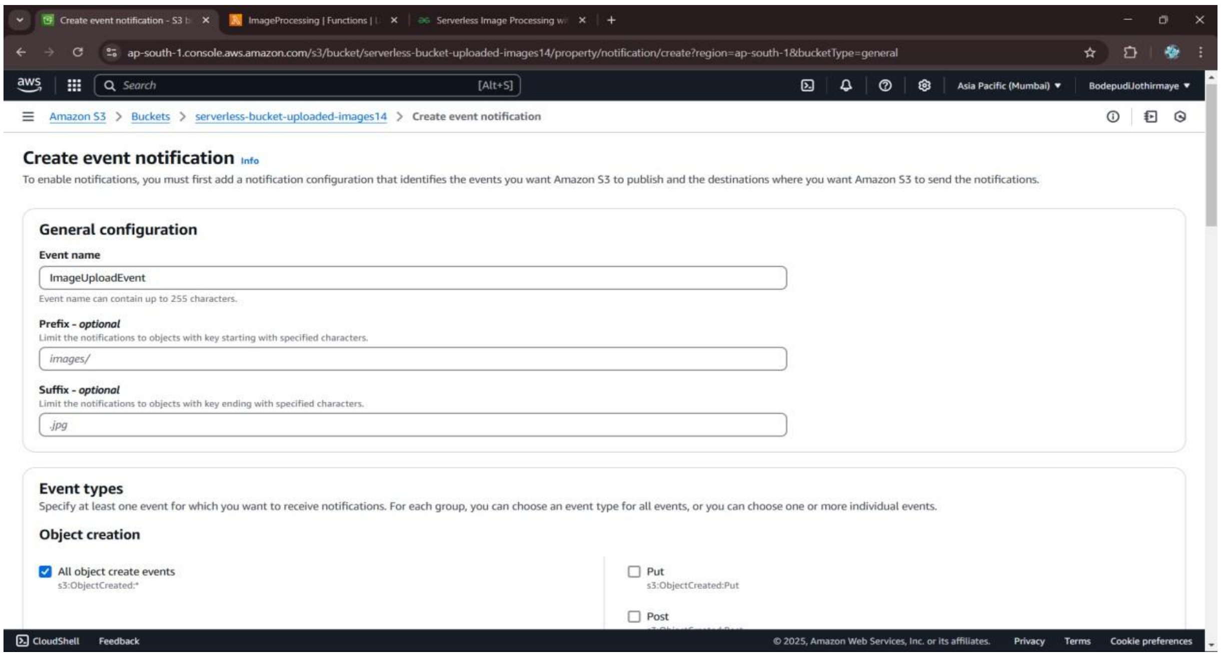
1 validation error detected: Value '[]' at 'layers' failed to satisfy constraint: Member must satisfy constraint: [Member must have length less than or equal to 2048, Member must have length greater than or equal to 1, Member must satisfy regular expression pattern: (arn:(aws[a-zA-Z-]*)?:lambda:[a-z]{2}((-gov)|(-iso([a-z]?)))?-[a-z]+-\d{1}:\d{12}:layer:[a-zA-Z0-9-_]+:[0-9]+)|(arn:[a-zA-Z0-9-]+:lambda:::awslayer:[a-zA-Z0-9-_]+), Member must not be null]

Step 5 - Creating S3 trigger

Now we need our Lambda function to know when an image is uploaded to the source bucket. We can do this by adding an event to the source S3 bucket and configure it to get triggered when an image is uploaded to the bucket which in turn invokes the Lambda function.

Go to S3 console. Select the source bucket ("serverless-bucket-uploaded-images"). Go to the Properties tab. Navigate to "Event Notifications". Click "Create Event Notifications".

Give an appropriate name to the event. Check the "All object create events".


Create event notification

To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.

General configuration

Event name: ImageUploadEvent

Prefix - optional: images/

Suffix - optional: .jpg

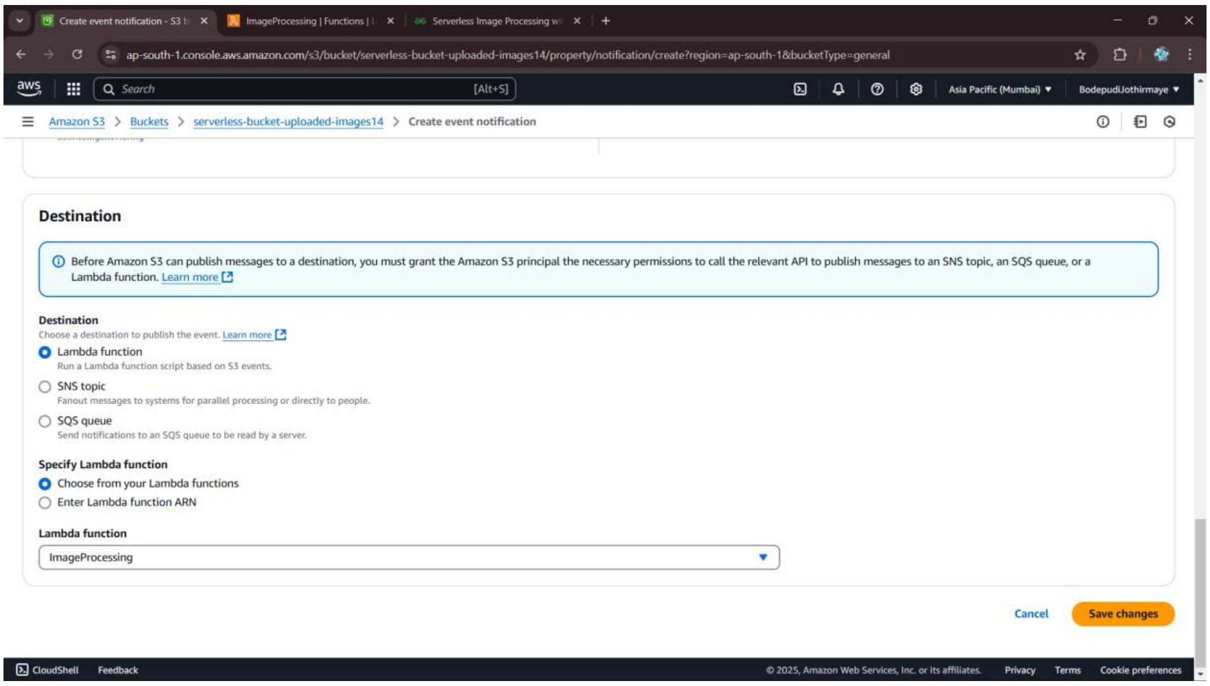
Event types

Object creation

[✓ All object create events] s3:ObjectCreated:*

Put s3:ObjectCreated:Put

Post


Create event notification - S3 b
ImageProcessing | Functions | L
Serverless Image Processing w
Cancel
Save changes
CloudShell
Feedback

ap-south-1.console.aws.amazon.com/s3/bucket/serverless-bucket-uploaded-images14/property/notification/create?region=ap-south-1&bucketType=general

Amazon S3 > Buckets > serverless-bucket-uploaded-images14 > Create event notification

Destination

Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. Learn more

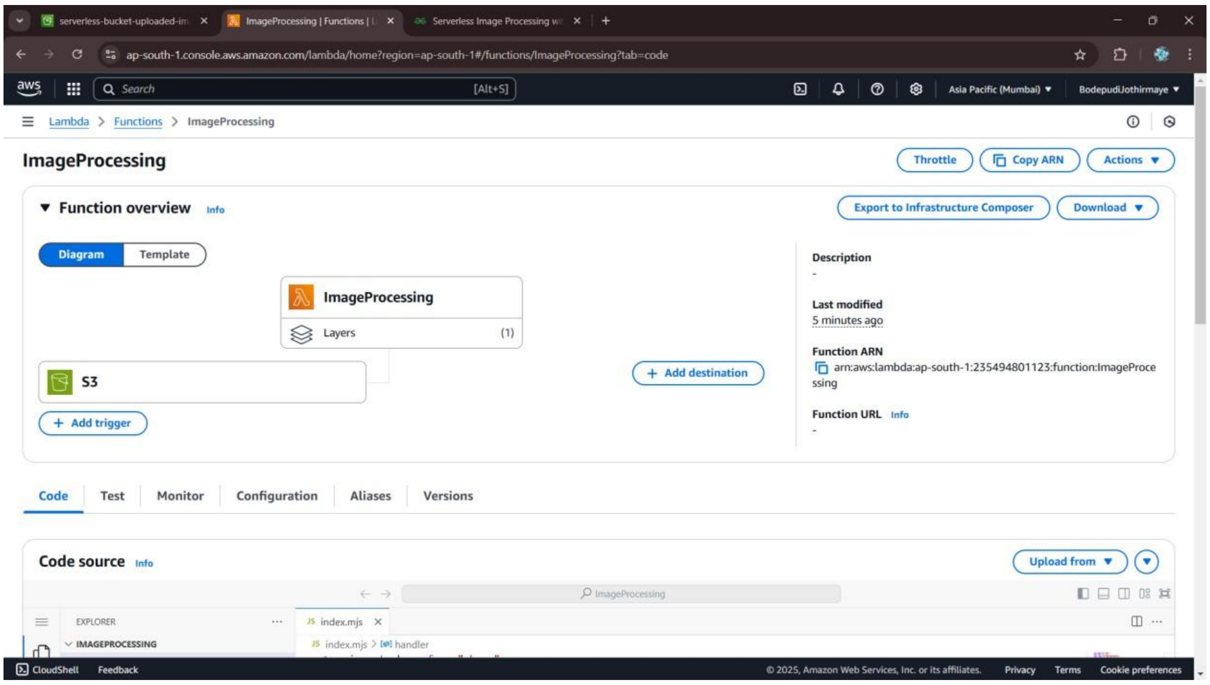
Destination
Choose a destination to publish the event. Learn more

[✓ Lambda function] Run a Lambda function script based on S3 events.
SNS topic — Fanout messages to systems for parallel processing or directly to people.
SQS queue — Send notifications to an SQS queue to be read by a server.

Specify Lambda function
[✓ Choose from your Lambda functions]
Enter Lambda function ARN

Lambda function: ImageProcessing

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/ImageProcessing?tab=code

Lambda > Functions > ImageProcessing

ImageProcessing

Function overview Info

Description
-

Last modified
5 minutes ago

Function ARN
arn:aws:lambda:ap-south-1:235494801123:function:ImageProcessing

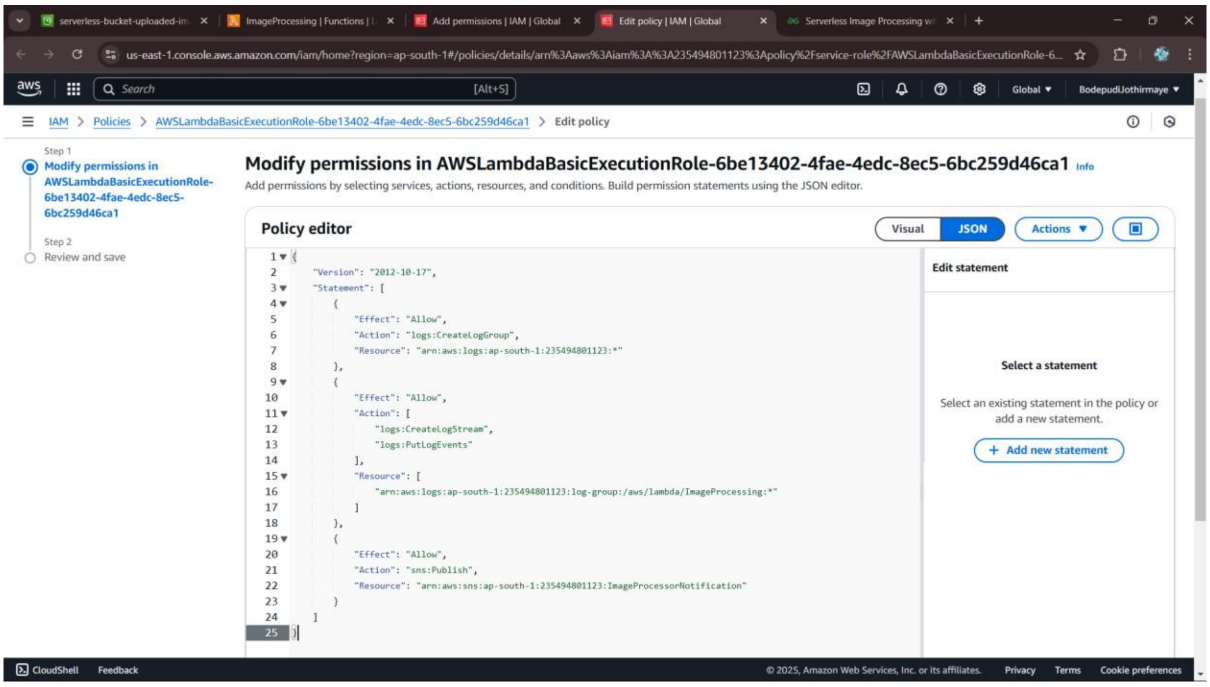
Function URL Info
-

Code source Info

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

STEP 6: In IAM go to roles and click on "Attach Policy" and click on edit policy and add the following

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect":  "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource":  "arn:aws:logs:ap-south-1:235494801123:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:ap-south-1:235494801123:log-group:/aws/lambda/ImageProcessing:*"
      ]
    },
    {
      "Effect":  "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification"
    }
  ]
}
```


serverless-bucket-uploaded-im
ImageProcessing | Functions | L
Add permissions | IAM | Global
Edit policy | IAM | Global
Serverless Image Processing w
us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/policies/details/arn%3Aaws%3Aiam%3A%3A235494801123%3Apolicy%2Fservice-role%2FAWSLambdaBasicExecutionRole-6...
Search
[Alt+S]
Global
BodepudiJothirmaye
IAM > Policies > AWSLambdaBasicExecutionRole-6be13402-4fae-4edc-8ec5-6bc259d46ca1 > Edit policy
Step 1
Modify permissions in AWSLambdaBasicExecutionRole-6be13402-4fae-4edc-8ec5-6bc259d46ca1
Step 2
Review and save
Info
Policy editor
Visual
JSON
Actions
Edit statement
Add new statement
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

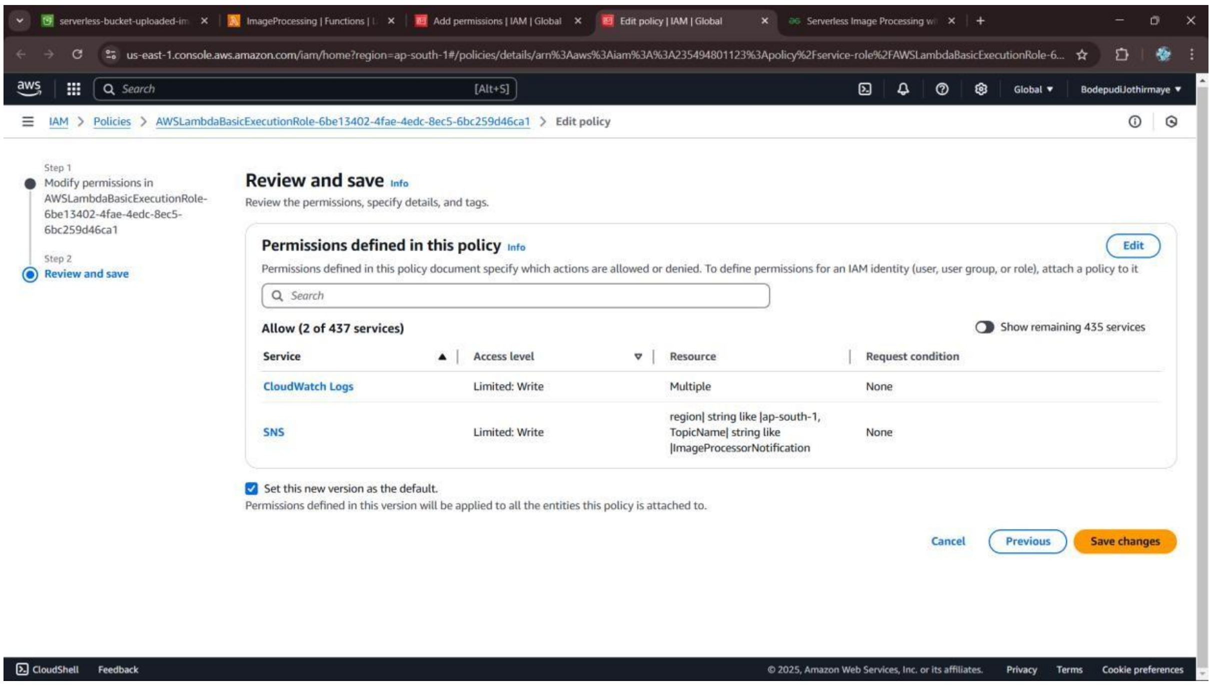
Modify permissions in AWSLambdaBasicExecutionRole-6be13402-4fae-4edc-8ec5-6bc259d46ca1

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:ap-south-1:235494801123:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:ap-south-1:235494801123:log-group:/aws/lambda/ImageProcessing:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification"
        }
    ]
}
```

Select a statement

Select an existing statement in the policy or add a new statement.


serverless-bucket-uploaded-im
ImageProcessing | Functions | L
Add permissions | IAM | Global
Edit policy | IAM | Global
Serverless Image Processing wi
us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/policies/details/arn%3Aaws%3Aiam%3A%3A235494801123%3Apolicy%2Fservice-role%2FAWSLambdaBasicExecutionRole-6...
Search [Alt+S]
Global
BodepudiJothirmaye
IAM > Policies > AWSLambdaBasicExecutionRole-6be13402-4fae-4edc-8ec5-6bc259d46ca1 > Edit policy

Step 1
Modify permissions in AWSLambdaBasicExecutionRole-6be13402-4fae-4edc-8ec5-6bc259d46ca1

Step 2
Review and save

Review and save Info

Review the permissions, specify details, and tags.

Permissions defined in this policy Info

Edit

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (2 of 437 services)

Show remaining 435 services

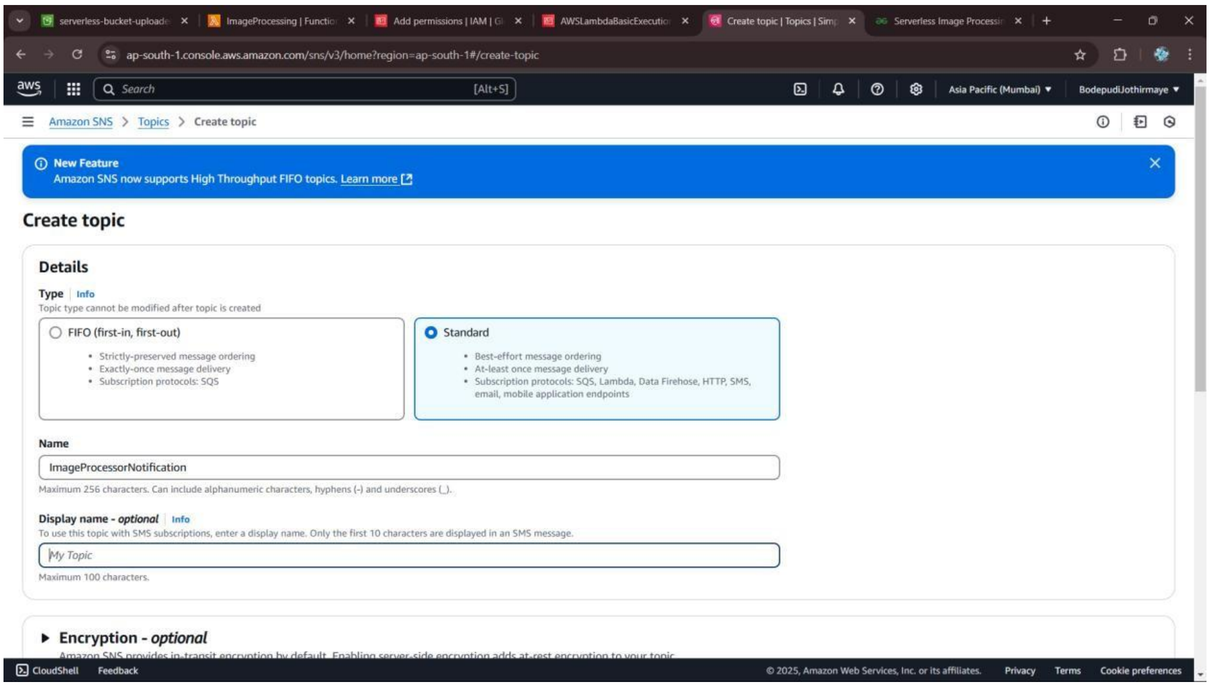
Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	Multiple	None
SNS	Limited: Write	region| string like |ap-south-1, TopicName| string like |ImageProcessorNotification	None

☑ Set this new version as the default.

Permissions defined in this version will be applied to all the entities this policy is attached to.

Cancel Previous Save changes

STEP 7: Create SNS topic named "ImageProcessorNotification" and click on save.


serverless-bucket-uploade
ImageProcessing | Functio
Add permissions | IAM | Gl
AWSLambdaBasicExecutio
Create topic | Topics | Simp
Serverless Image Processi
ap-south-1.console.aws.amazon.com/sns/v3/home?region=ap-south-1#/create-topic
Search [Alt+S]
Asia Pacific (Mumbai)
BodepudiJothirmaye
Amazon SNS > Topics > Create topic

New Feature
Amazon SNS now supports High Throughput FIFO topics. Learn more

Create topic

Details

Type Info

Topic type cannot be modified after topic is created

○ FIFO (first-in, first-out)
- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

● Standard
- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name

ImageProcessorNotification

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* Info

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

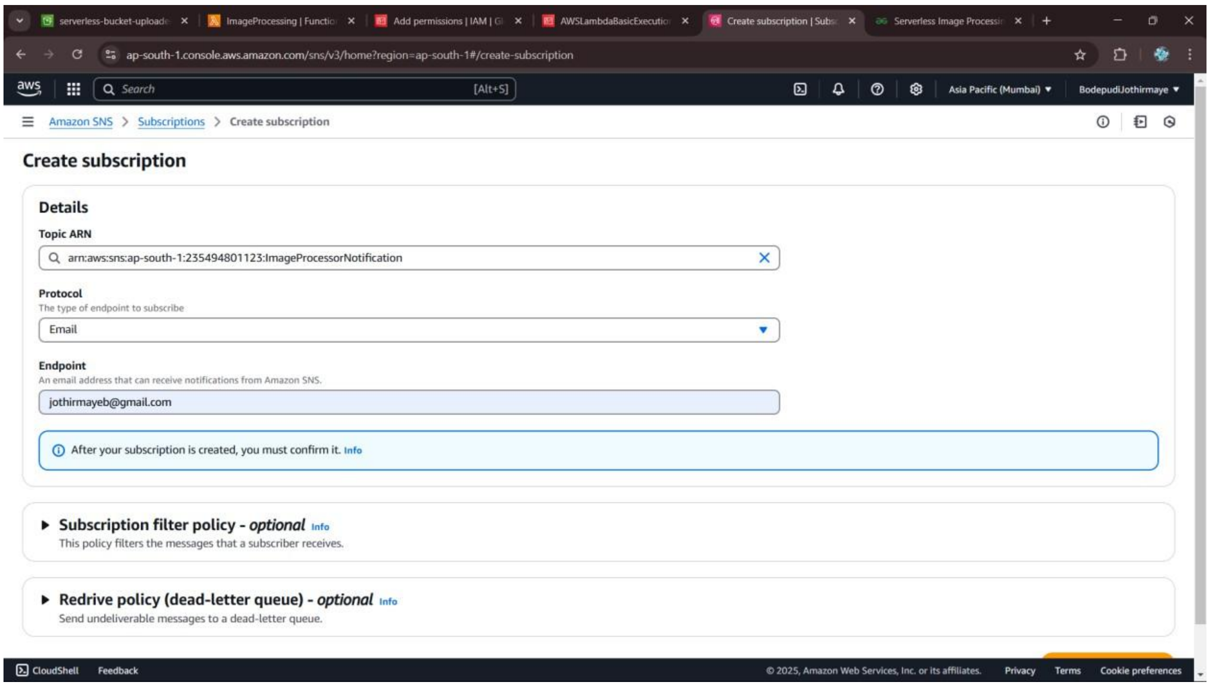
My Topic

Maximum 100 characters.

▶ **Encryption - *optional***

Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Now click on create new subscription and select protocol as email and add you email and click on create new subscription



Amazon SNS > Subscriptions > Create subscription
Subscription filter policy - optional
Redrive policy (dead-letter queue) - optional

Create subscription

Details

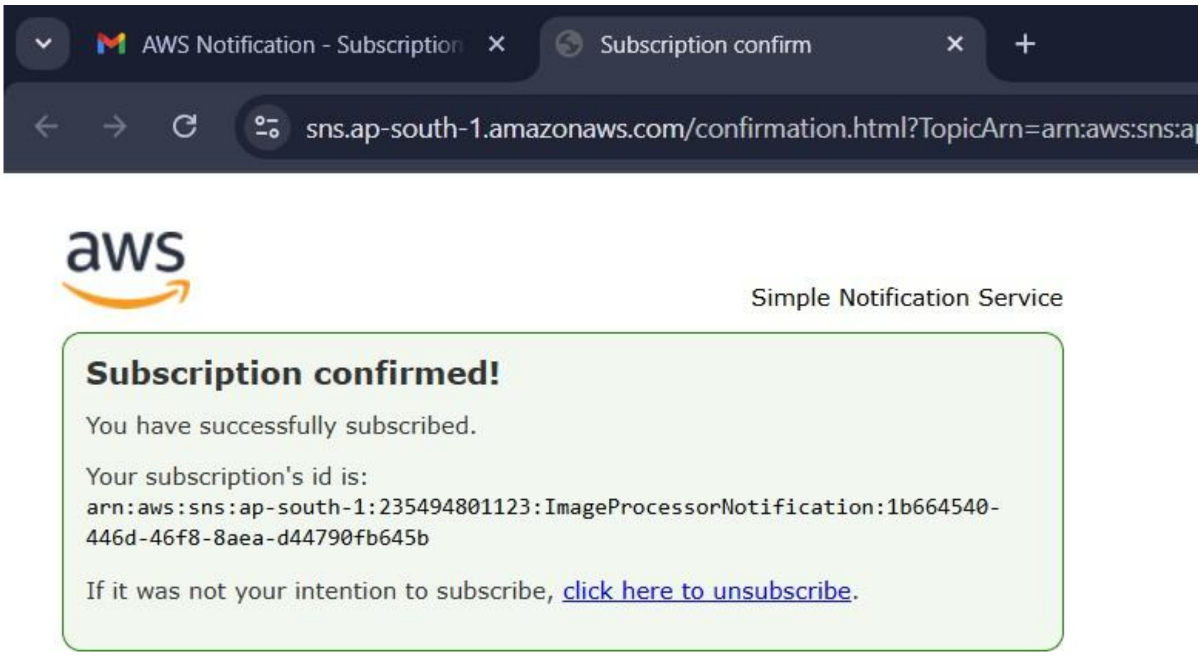
Topic ARN: arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification

Protocol: Email

Endpoint: jothirmayeb@gmail.com

After your subscription is created, you must confirm it. Info

An email is sent to you to confirm the subscription



sns.ap-south-1.amazonaws.com/confirmation.html?TopicArn=arn:aws:sns:a

Simple Notification Service

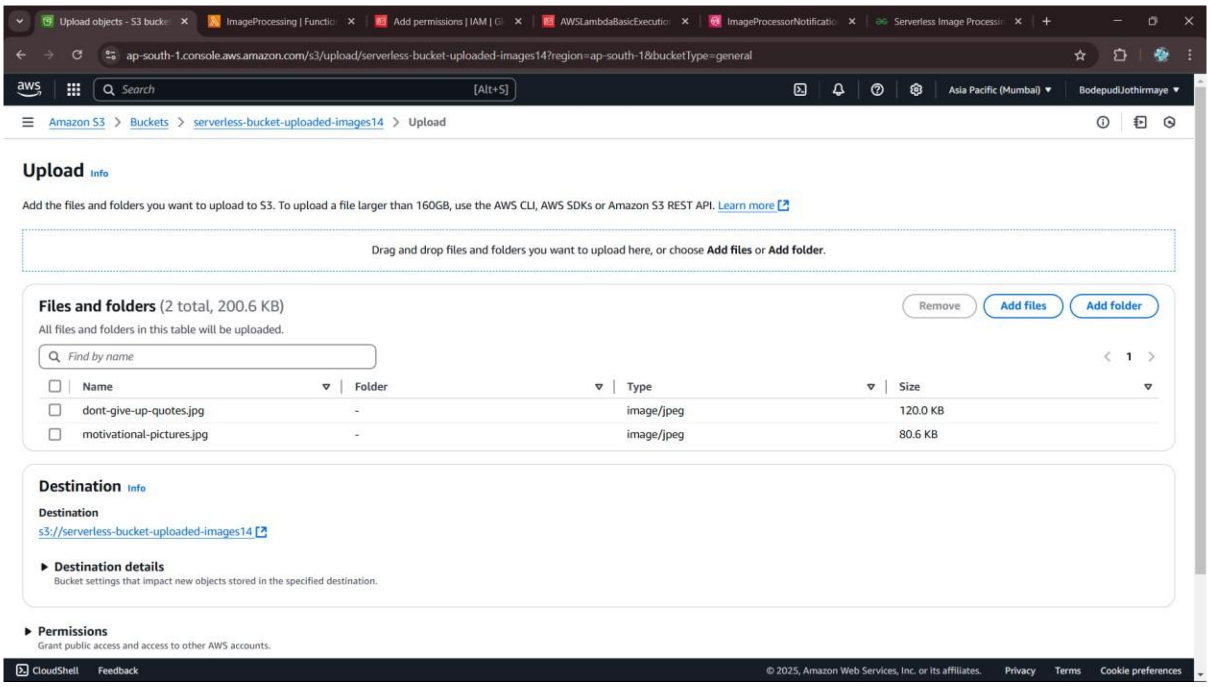
Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification:1b664540-446d-46f8-8aea-d44790fb645b

If it was not your intention to subscribe, click here to unsubscribe.

Now add the images to serverless-bucket-upload-images14



ap-south-1.console.aws.amazon.com/s3/upload/serverless-bucket-uploaded-images14?region=ap-south-1&bucketType=general

Amazon S3 > Buckets > serverless-bucket-uploaded-images14 > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (2 total, 200.6 KB)

All files and folders in this table will be uploaded.

Name	Folder	Type	Size
dont-give-up-quotes.jpg	-	image/jpeg	120.0 KB
motivational-pictures.jpg	-	image/jpeg	80.6 KB

Destination Info

Destination

s3://serverless-bucket-uploaded-images14

Destination details

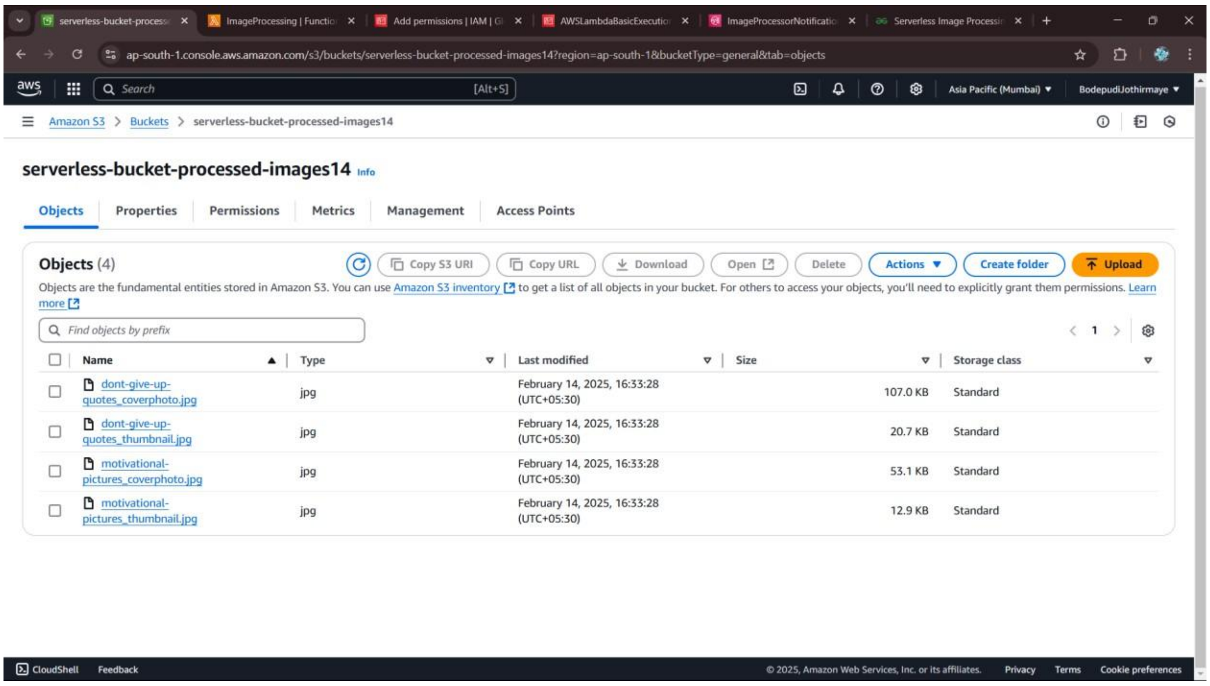
Bucket settings that impact new objects stored in the specified destination.

Permissions

Grant public access and access to other AWS accounts.

© 2025, Amazon Web Services, Inc. or its affiliates.

After the successful process of you image the two different sizes of the images will be added to your destination bucket.



ap-south-1.console.aws.amazon.com/s3/buckets/serverless-bucket-processed-images14?region=ap-south-1&bucketType=general&tab=objects

Amazon S3 > Buckets > serverless-bucket-processed-images14

serverless-bucket-processed-images14 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
dont-give-up-quotes_coverphoto.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	107.0 KB	Standard
dont-give-up-quotes_thumbnail.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	20.7 KB	Standard
motivational-pictures_coverphoto.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	53.1 KB	Standard
motivational-pictures_thumbnail.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	12.9 KB	Standard

© 2025, Amazon Web Services, Inc. or its affiliates.

OUTPUT:

After the successful process an mail will be sent to your email saying “image processing completed for image.jpg”

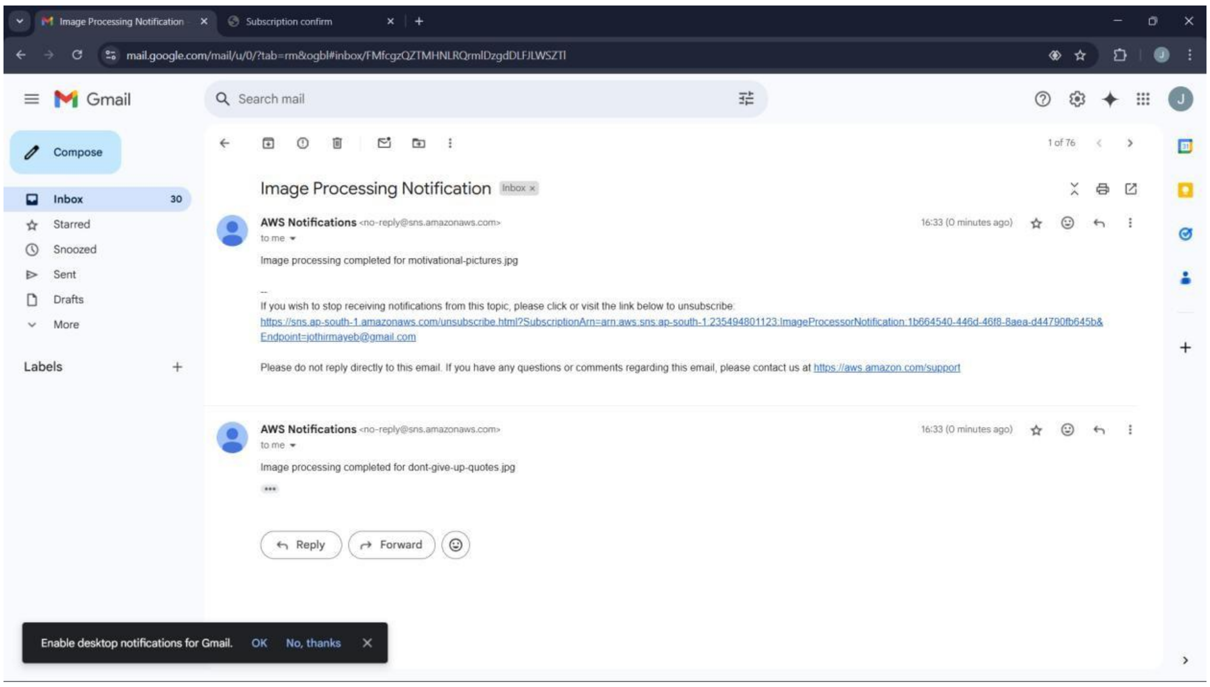


Image Processing Notification

AWS Notifications <no-reply@sns.amazonaws.com>
to me
16:33 (0 minutes ago)

Image processing completed for motivational-pictures.jpg

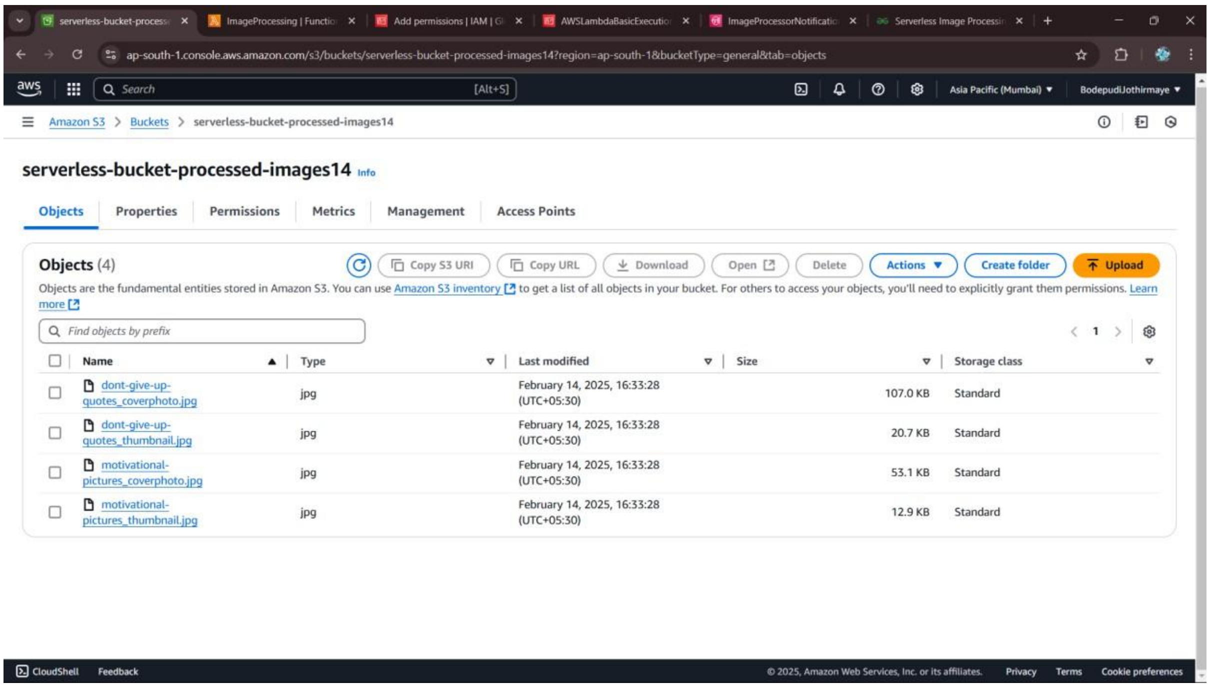
--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:235494801123:ImageProcessorNotification:1b664540-446d-46f8-8aea-d44790fb645b&Endpoint=jothirmayeb@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

AWS Notifications <no-reply@sns.amazonaws.com>
to me
16:33 (0 minutes ago)

Image processing completed for dont-give-up-quotes.jpg

Image Processing Notification
Subscription confirm
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzQZTMHNLRQrmlDzgdDLFJLWSZTl
Gmail
Search mail
Compose
Inbox 30
Starred
Snoozed
Sent
Drafts
More
Labels
1 of 76
Inbox ×
Reply
Forward
Enable desktop notifications for Gmail. OK No, thanks



ap-south-1.console.aws.amazon.com/s3/buckets/serverless-bucket-processed-images14?region=ap-south-1&bucketType=general&tab=objects

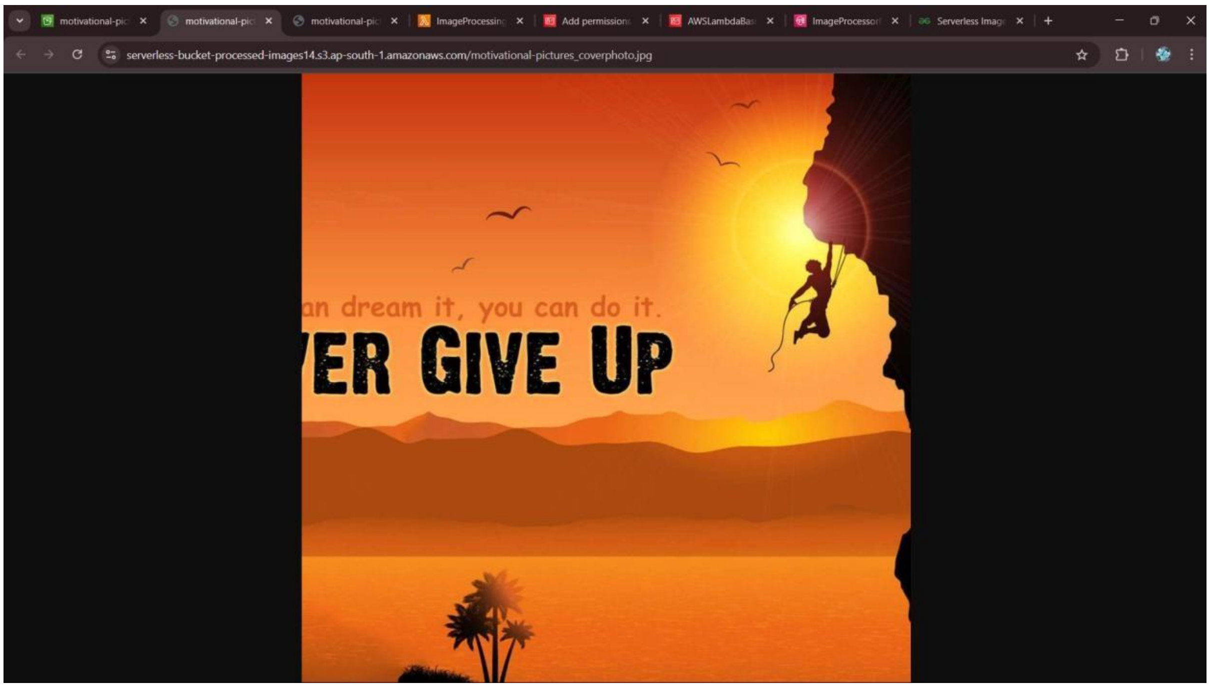
Amazon S3 > Buckets > serverless-bucket-processed-images14

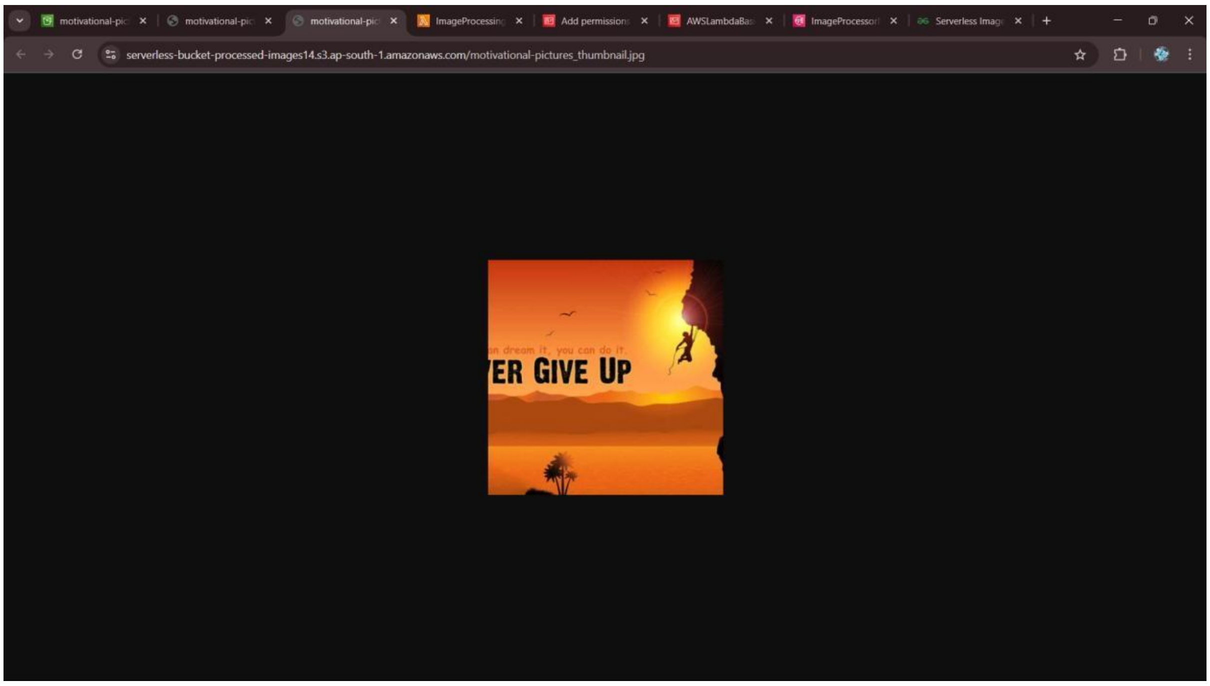
serverless-bucket-processed-images14

Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
dont-give-up-quotes_coverphoto.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	107.0 KB	Standard
dont-give-up-quotes_thumbnail.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	20.7 KB	Standard
motivational-pictures_coverphoto.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	53.1 KB	Standard
motivational-pictures_thumbnail.jpg	jpg	February 14, 2025, 16:33:28 (UTC+05:30)	12.9 KB	Standard


motivational-pic
motivational-pic
motivational-pic
ImageProcessing
Add permissions
AWSLambdaBas
ImageProcessor
Serverless Image
serverless-bucket-processed-images14.s3.ap-south-1.amazonaws.com/motivational-pictures_coverphoto.jpg
an dream it, you can do it.
ER GIVE UP


motivational-pic
motivational-pic
motivational-pic
ImageProcessing
Add permissions
AWSLambdaBas
ImageProcessor
Serverless Image
serverless-bucket-processed-images14.s3.ap-south-1.amazonaws.com/motivational-pictures_thumbnail.jpg
an dream it, you can do it.
ER GIVE UP

CONCLUSION:

Serverless image processing with AWS Lambda and S3 provides a scalable, cost-effective, and efficient solution for handling image transformations in the cloud. By leveraging AWS Lambda, you can automatically process images in response to S3 events without managing servers. Integration with other AWS services, such as Amazon S3, Amazon SNS, and Amazon CloudWatch, enables automated workflows, monitoring, and alerting.

This approach is ideal for real-time applications like thumbnail generation, watermarking, and format conversions. It reduces infrastructure costs, enhances scalability, and simplifies deployment. However, developers must optimize Lambda execution time, handle cold starts efficiently, and manage permissions securely.

By adopting serverless architecture for image processing, businesses can achieve high availability, fault tolerance, and improved performance, making it a practical choice for modern cloud-based applications.