
Serverless Image Processing



“Serverless Image Processing with AWS Lambda and S3”

TABLE OF CONTENTS

1. Introduction
2. AWS S3
3. AWS Lambda
4. Workflow for Serverless Image Processing with
AWS Lambda and S3
5. Steps for Deployment
 - Step 1: Create S3 Buckets
 - Step 2: Create S3 Bucket Policy
 - Step 3: Create Lambda Function
 - Step 4: Test the application
 - Step 5: Creating S3 trigger
6. Conclusion
7. References

1. INTRODUCTION

In today's digital landscape, managing and processing images efficiently is crucial for a wide range of applications, from web development to mobile apps and beyond. Traditionally, setting up and maintaining servers for image processing can be both time-consuming and costly. However, with the advent of serverless computing, there is a more streamlined and cost-effective way to handle image processing tasks.

Serverless computing allows developers to build and run applications without having to manage infrastructure. AWS Lambda, a key player in the serverless ecosystem, lets you run code in response to events without provisioning or managing servers. When combined with Amazon S3, a scalable object storage service, you can create a powerful and efficient image processing workflow.

We'll explore how to set up a serverless image processing pipeline using AWS Lambda and S3. We'll walk through the steps of configuring S3 to store images, setting up Lambda functions to process these images on-the-fly, and leveraging other AWS services to enhance this workflow. By the end of this, you'll have a robust and scalable solution for handling image processing tasks without the overhead of traditional server management.



2. AWS S3



Amazon S3 (Simple Storage Service) is a scalable and secure object storage service by Amazon Web Services (AWS). It offers:

Scalability: Virtually unlimited data storage.

Durability and Availability: 99.999999999% durability and 99.99% availability.

Security: Encryption, fine-grained access controls, and IAM integration.

Cost-Effectiveness: Pay-as-you-go pricing with various storage classes.

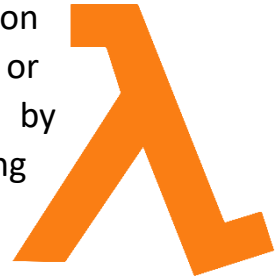
Flexibility: Stores any type of data and integrates with other AWS services.

Performance: Low latency and high throughput.

Ease of Use: Simple web interface, APIs, and SDKs.

3. AWS LAMBDA

AWS Lambda is a serverless compute service provided by Amazon Web Services that allows you to run code without provisioning or managing servers. It automatically scales your applications by running code in response to events and managing the underlying compute resources. AWS Lambda is highly efficient, cost-effective, and ideal for a variety of applications. Here are some common use cases for AWS Lambda:



- **Event-Driven Processing:** Lambda functions can be triggered by events such as changes to data in S3, updates to a DynamoDB table, or HTTP requests via Amazon API Gateway.
- **Real-Time File Processing:** Automatically process files uploaded to S3, such as generating thumbnails for images or transcribing audio files.
- **Data Transformation and ETL:** Extract, transform, and load (ETL) data from one format or service to another, enabling seamless data integration.
- **Backend for Web and Mobile Applications:** Build and deploy backends for web and mobile apps, handling tasks such as API requests, authentication, and server-side logic.
- **Automation and Maintenance Tasks:** Schedule automated tasks, such as cleaning up old log files, synchronizing data across services, or monitoring resource usage.
- **Microservices Architecture:** Develop microservices that perform discrete functions, promoting a modular and scalable application design.

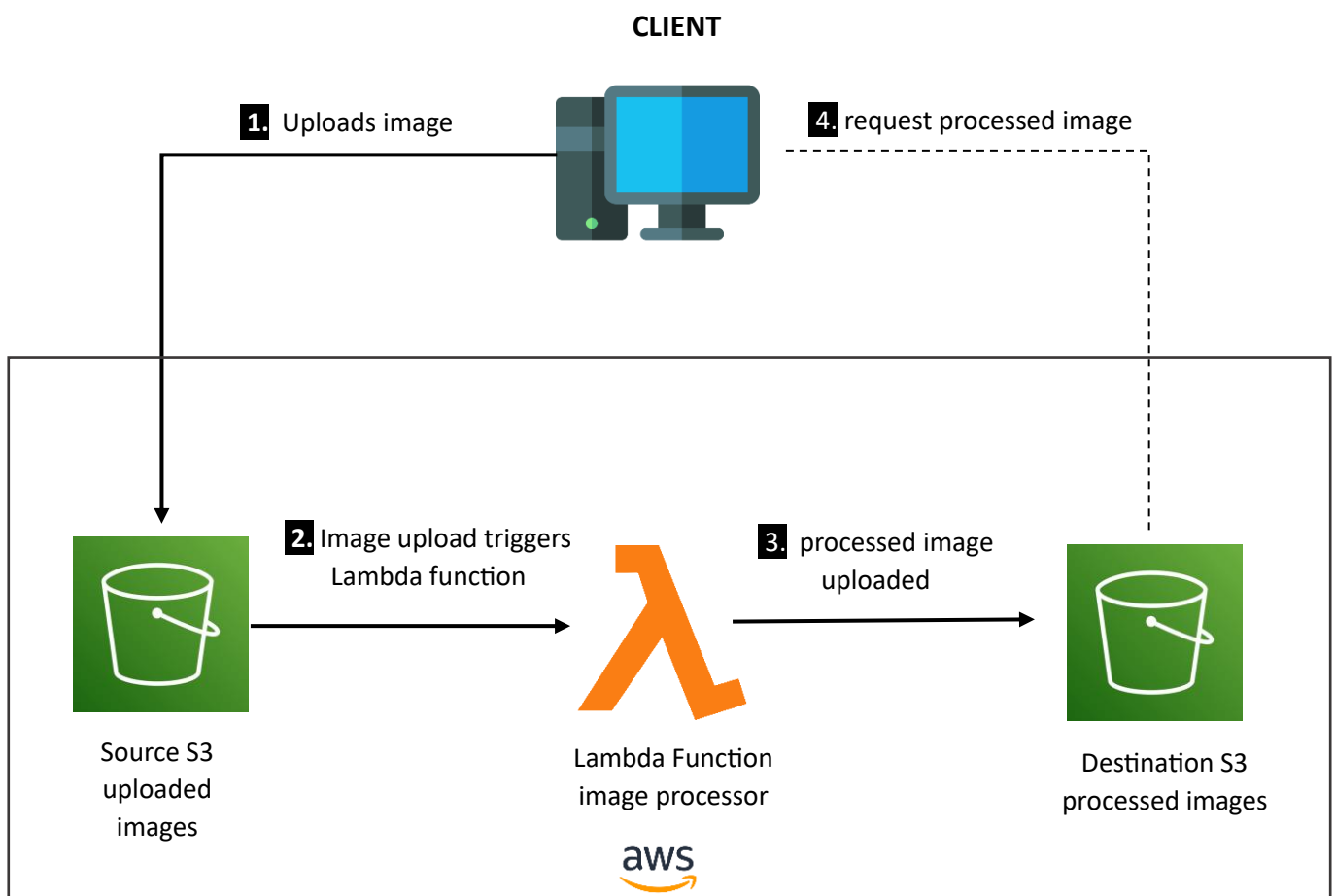
4. WORKFLOW

1. Image Upload: A user uploads an image to the designated source S3 bucket, which serves as the repository for incoming images.

2. Event Trigger: The upload event in the source S3 bucket triggers a notification that invokes a Lambda function.

3. Image Processing: The invoked Lambda function processes the image, performing tasks such as resizing, filtering, or format conversion.

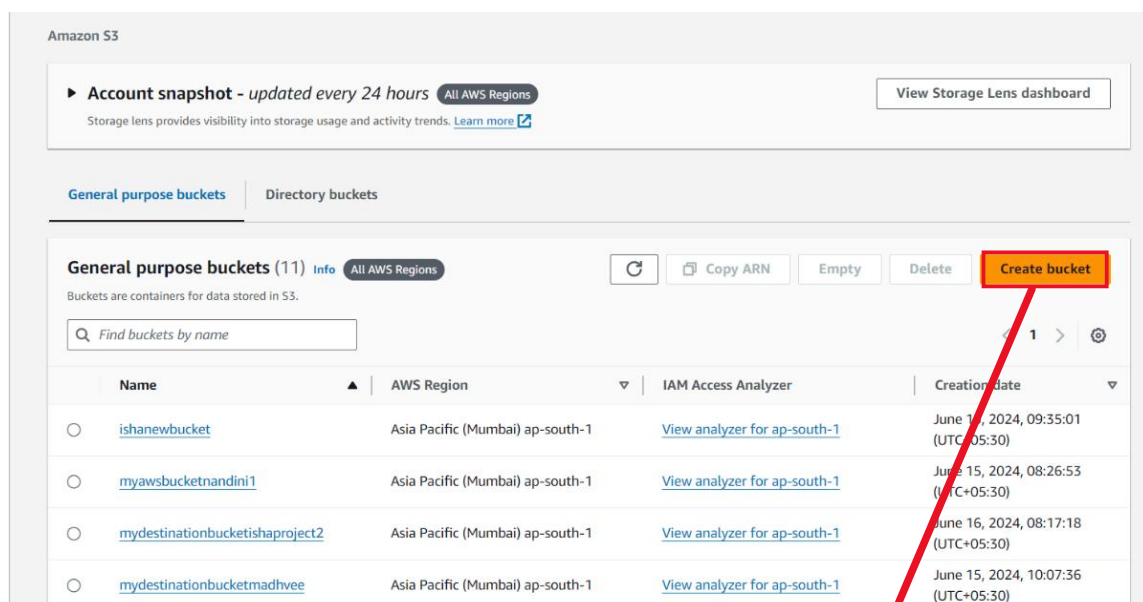
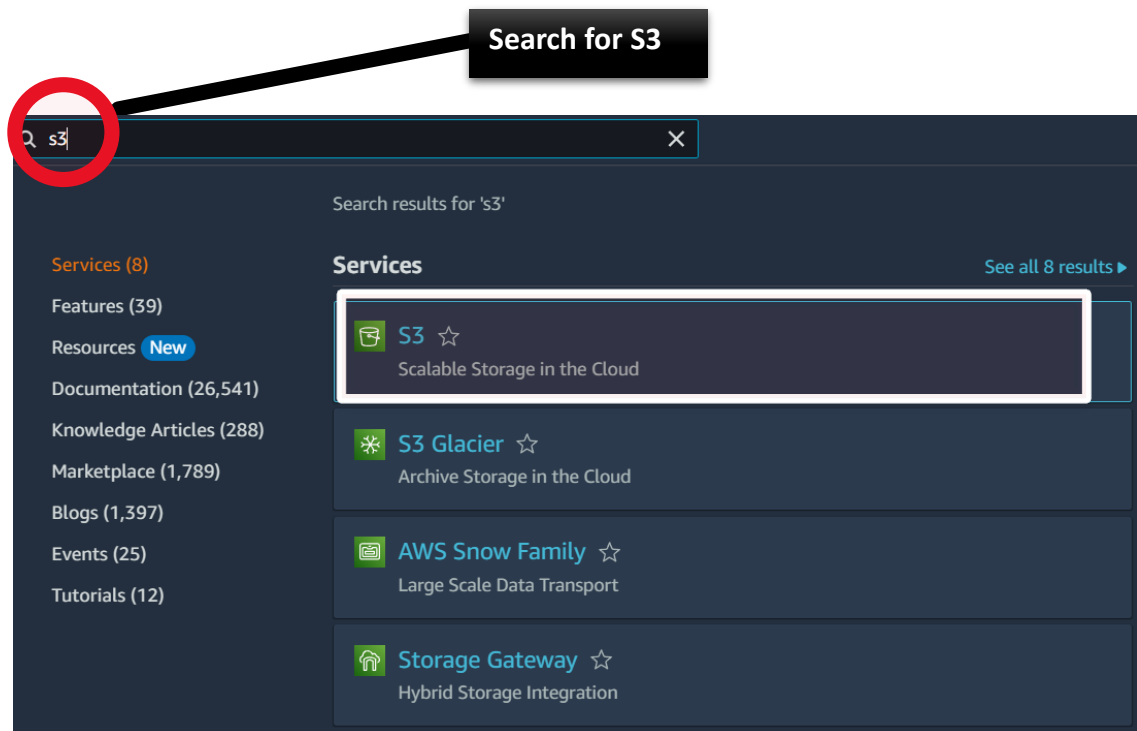
4. Storage and Retrieval: The processed image is saved to a destination S3 bucket, from where the user can request and retrieve the processed image.



5. STEPS FOR DEPLOYMENT

STEP 1 :

Create S3 Buckets



Create bucket

Click here to create a bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region

Asia Pacific (Mumbai) ap-south-1

Bucket name [Info](#)

shvzmainbucket

Provide a globally unique name for bucket

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

UNCHECK "Block all public

☐ 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.

Click on "**I Acknowledge current settings**" and Leave the Remaining settings on DEFAULT

Create bucket

Follow the same steps and create another bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket name Info

shvzresizedbucket

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

Two buckets have been set up for storing and processing images.

General purpose buckets

Directory buckets

General purpose buckets (12)

Info

All AWS Regions

↺

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 >

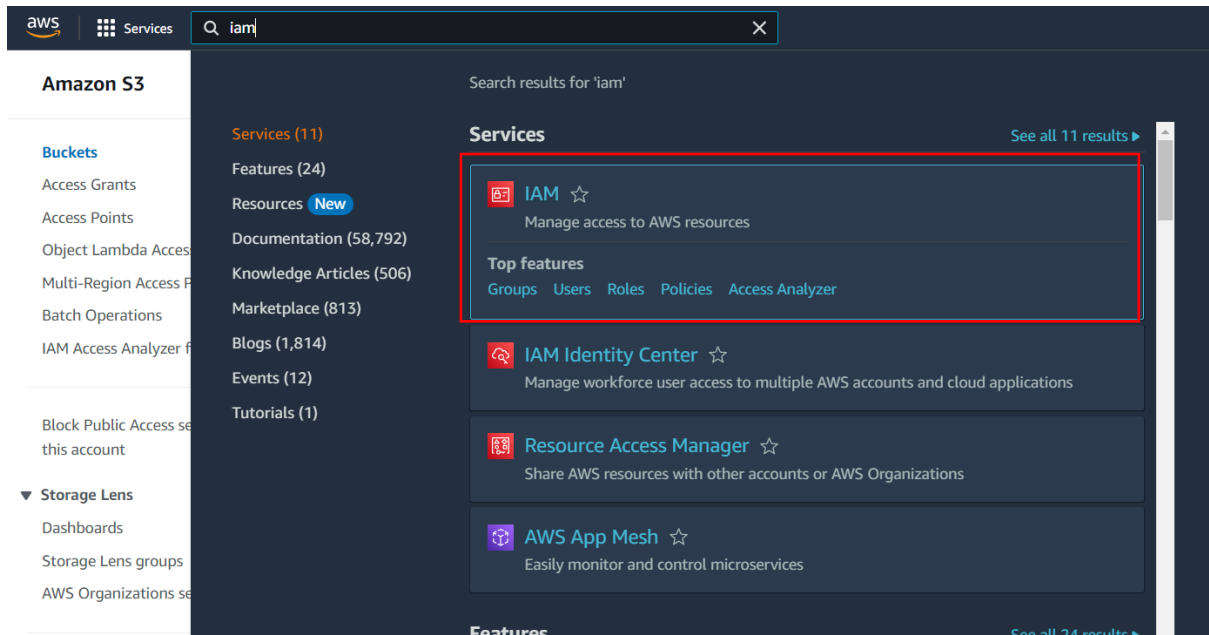
⚙

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	shvzresizedbucket	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 19, 2024, 17:42:58 (UTC+05:30)
<input type="radio"/>	shvzmainbucket	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 19, 2024, 17:42:23 (UTC+05:30)

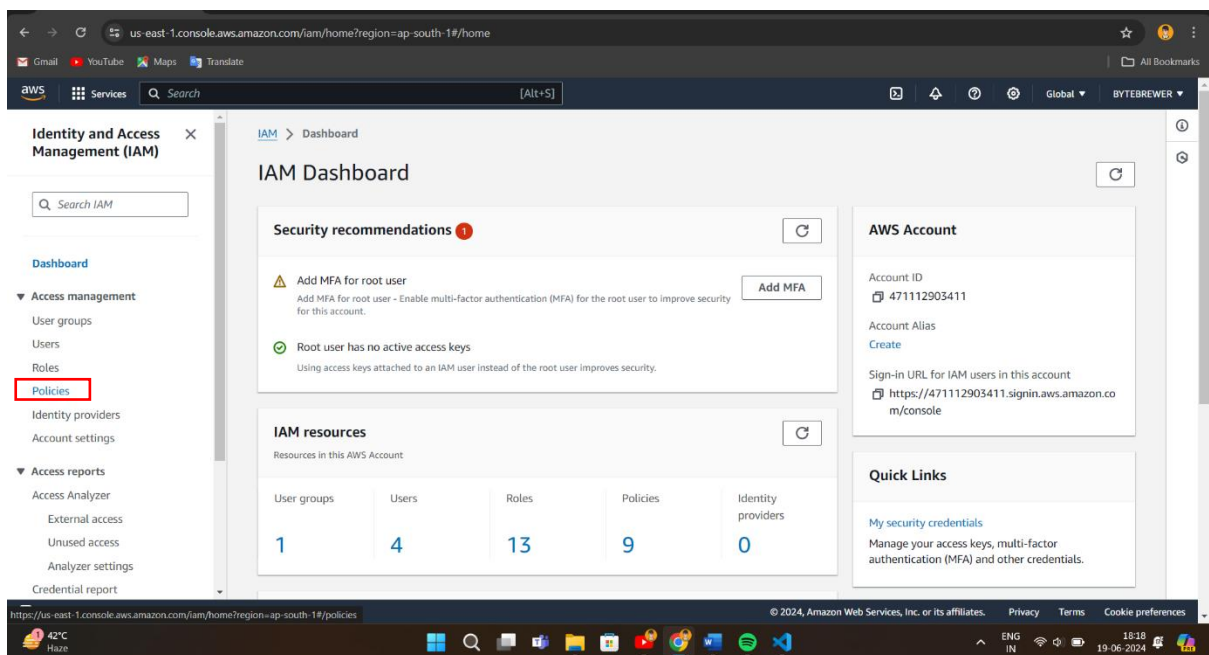
STEP 2 :

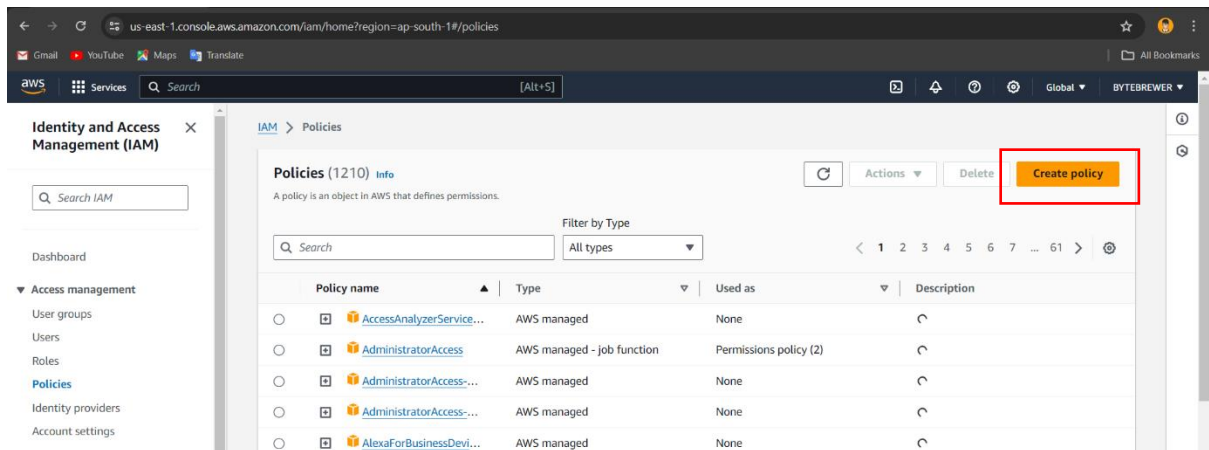
Create S3 Bucket Policy

Search for IAM > Click on IAM

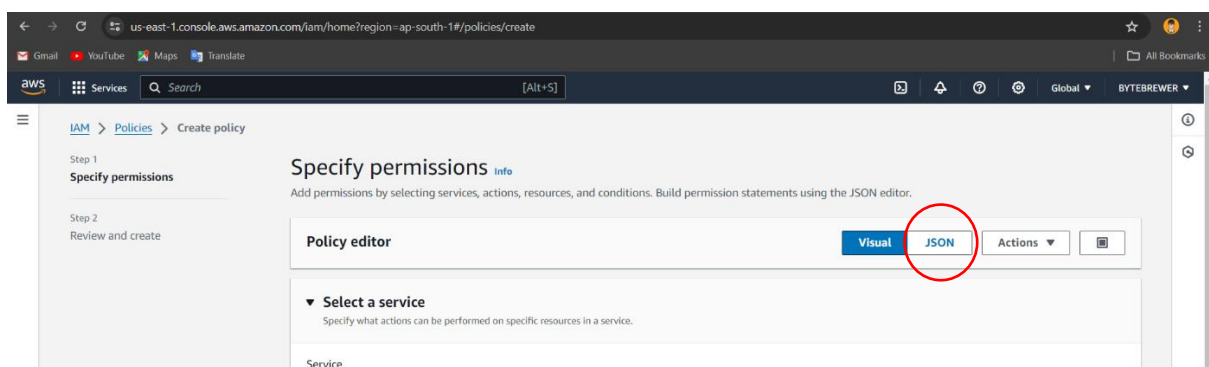


Click on Policies > Click on Create Policy





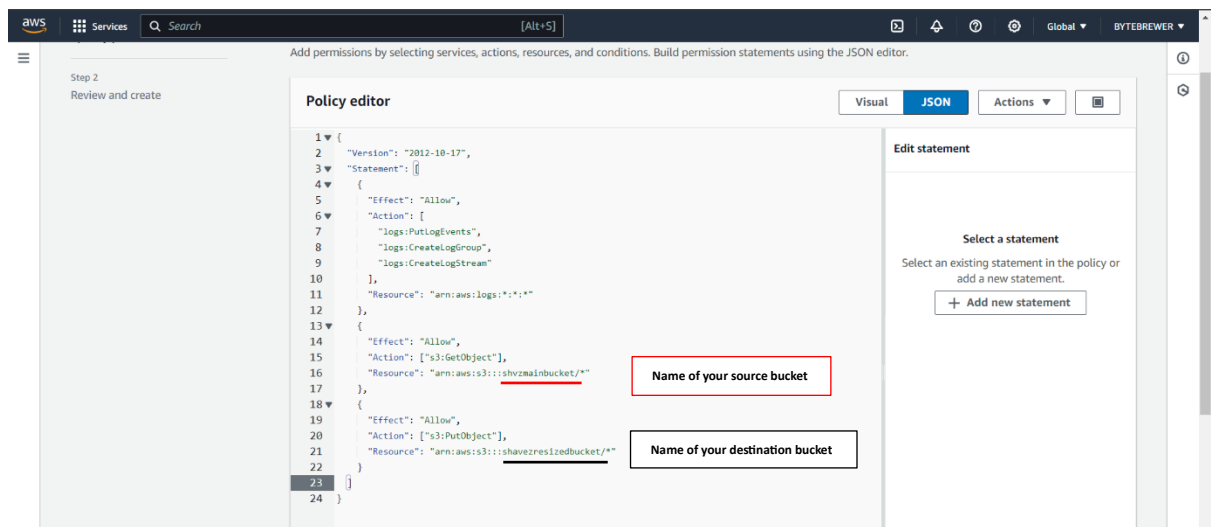
Click on **JSON**



Copy the following JSON code

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::BUCKET_NAME/*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:PutObject"],
      "Resource": "arn:aws:s3:::DEST_BUCKET/*"
    }
  ]
}
```

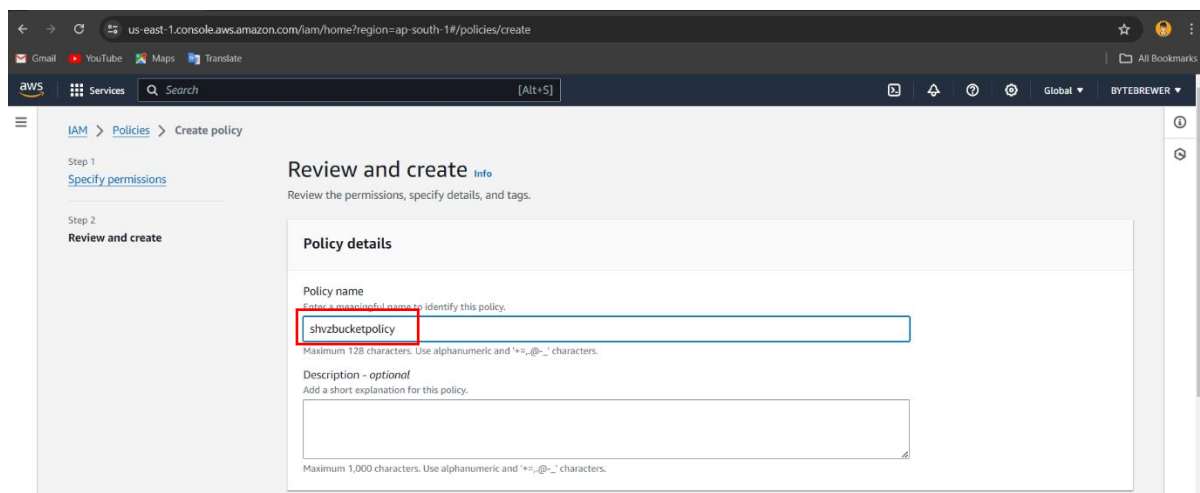
Paste the code in the **Policy editor** > update buckets name> click on **Next**



The screenshot shows the AWS IAM Policy editor interface. The left sidebar indicates "Step 2: Review and create". The main area is titled "Policy editor" and shows a JSON policy document. The policy has two statements: one for logs and one for S3 buckets. The resource ARNs for the S3 buckets are highlighted with red boxes and labeled "Name of your source bucket" and "Name of your destination bucket".

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "logs:PutLogEvents",
8         "logs:CreateLogGroup",
9         "logs:CreateLogStream"
10      ],
11       "Resource": "arn:aws:logs:*:*:*"
12     },
13     {
14       "Effect": "Allow",
15       "Action": ["s3:GetObject"],
16       "Resource": "arn:aws:s3:::shvzmainbucket/*"
17     },
18     {
19       "Effect": "Allow",
20       "Action": ["s3:PutObject"],
21       "Resource": "arn:aws:s3:::shvzresizedbucket/*"
22     }
23   ]
24 }
```

Give a **name** to your Policy > Click on **Create Policy**

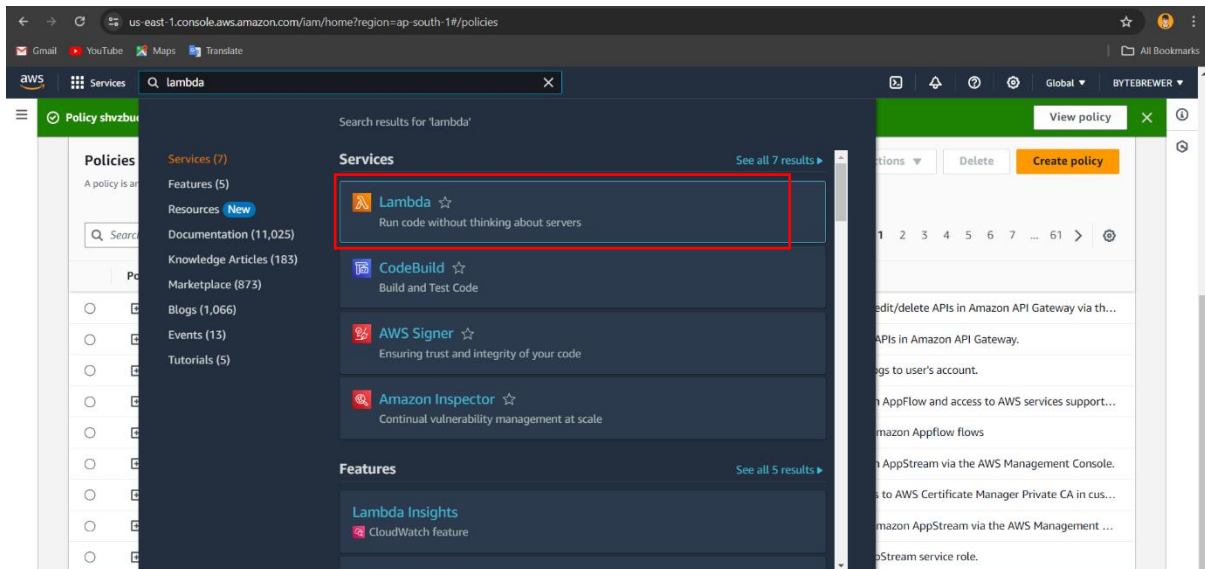


The screenshot shows the "Review and create" step of the AWS IAM Policy creation process. The page title is "Review and create". The "Policy details" section contains a "Policy name" field, which is highlighted with a red box and contains the text "shvzbucketpolicy". Below it is a "Description - optional" field. The page also shows the "IAM > Policies > Create policy" breadcrumb and the "Step 2: Review and create" label.

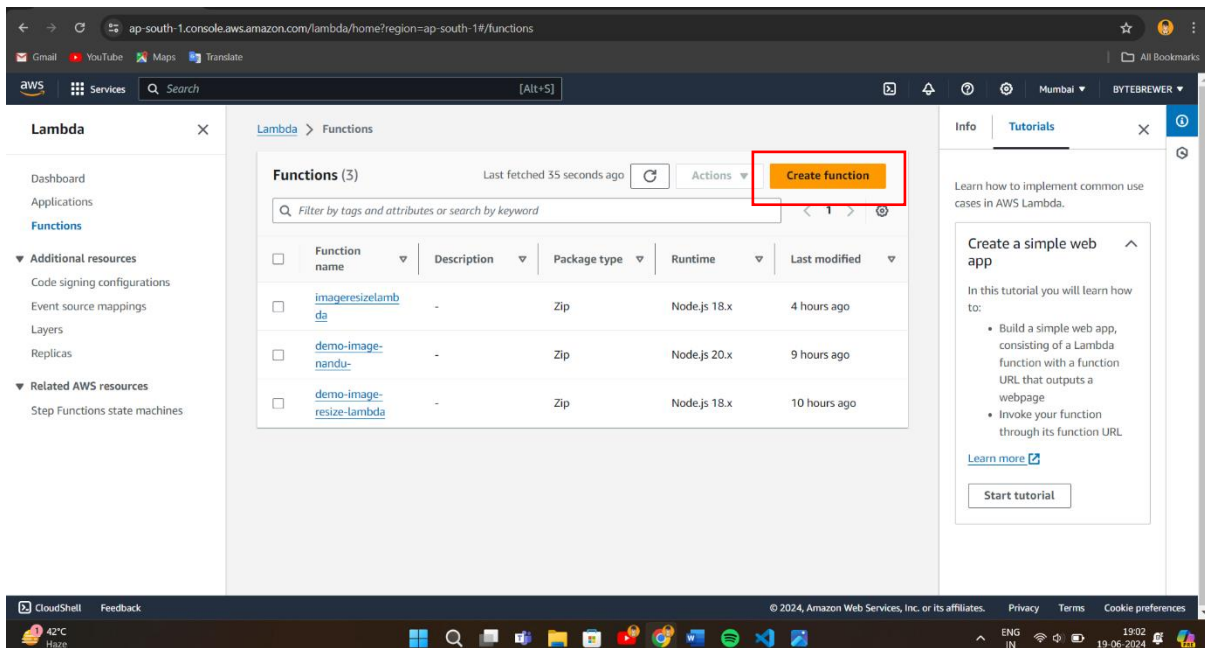
STEP 3 :

Create Lambda Function

Search for **Lambda**



Click on **Create Function**



Select Author from Scratch

Create function [Info](#)

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.

Info **Tutorials**

Learn how to implement common use cases in AWS Lambda.

Create a simple web app [^](#)

In this tutorial you will learn how

Give a name to your Function and Choose the language to use to write your function (Here I'm using Node.js 18.x)

Click on Change default execution role > IAM console

Function name [Info](#)

Enter a name that describes the purpose of your function.

shvzresizefunc

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)

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

Node.js 18.x

Architecture [Info](#)

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

☒ x86_64

☐ arm64

Permissions [Info](#)

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](#)

Info **Tutorials**

Learn how to implement common use cases in AWS Lambda.

Create a simple web app [^](#)

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#)

▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☒ Create a new role with basic Lambda permissions
- ☐ Use an existing role
- ☐ Create a new role from AWS policy templates

IAM console > AWS service

Select trusted entity [Info](#)

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a third party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

S3

Choose S3

✚ After clicking on **NEXT**, Search for the ***Policy*** that we created earlier and select it.

Add permissions [Info](#)

Add permissions [Info](#)

Permissions policies (1/937) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

shvz X All types 1 match

<input checked="" type="checkbox"/>	Policy name ?	Type	Description
<input checked="" type="checkbox"/>	shvzbucketpolicy	Customer managed	-

► Set permissions boundary - *optional*

Cancel Previous **Next**

✚ Click on **NEXT** > Give a meaningful name to IDENTIFY this **ROLE**.

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

shvzs3role

Maximum 64 characters. Use alphanumeric and '+=, @-._' characters.

Description

Add a short explanation for this role.

Allows S3 to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=, @-./\[\]!#\$%^&*(){};:'" <>`

Leave rest default and hit
CREATE ROLE button

Create role

Head back to **Create function** tab, choose ***"use an existing role"*** and select the role we just created now then **HIT** the Create Function Button.

▼ Change default execution role

Search

isharole

ishunandu

myrolenandini

myrolenandini1

shvzs3role

shvzs3role

role must have permission to upload logs to Amazon

View the shvzs3role role on the IAM console.

HIT the Create Function Button.

☑ Successfully created the function shvzresizefunc. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > shvzresizefunc

shvzresizefunc

Throttle Copy ARN Actions

▼ Function overview Info

Export to Application Composer Download

Diagram Template

shvzresizefunc

Layers (0)

+ Add trigger + Add destination

Description

Last modified now

Function ARN
arn:aws:lambda:ap-south-1:471112903411:function:shvzresizefunc

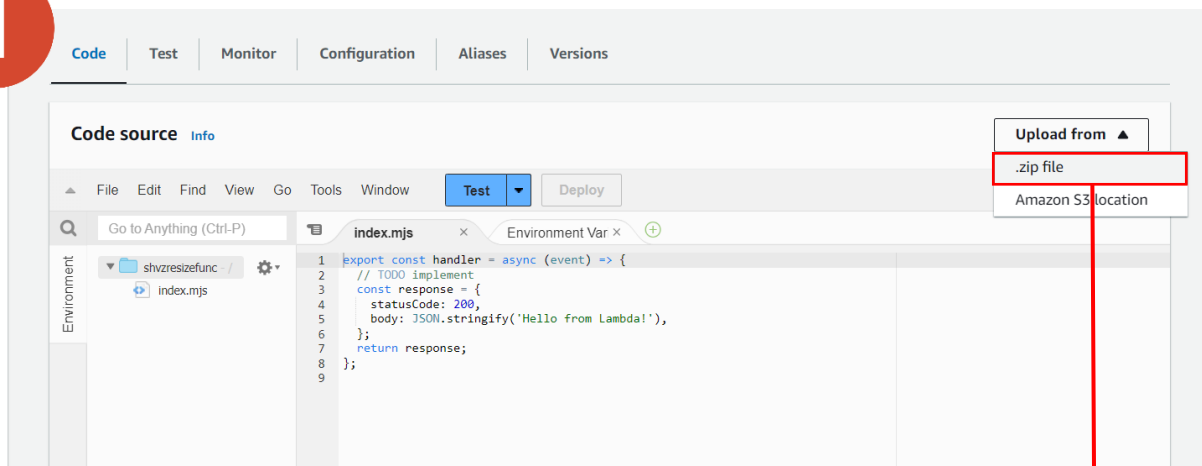
Function URL Info

STEP 4 :

Test The Application

1. Upload the function.zip file that contain the code to resize images.
2. Add the **Environment Variable**
Key: as per you code (Here, it is **DEST_BUCKET**)
Value: Destination Bucket Name (Here, it is **shvzresizedbucket**)
3. Test the Event

1



function.zip

2

Environment configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

RDS databases

Monitoring and operations tools

Concurrency

Asynchronous

Environment variables

Edit

Key	Value
No environment variables	
No environment variables associated with this function.	
Edit	

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

Key	Value	
<input type="text" value="DEST_BUCKET"/>	<input type="text" value="shvzresizedbucket"/>	Remove
<input type="button" value="Add environment variable"/>		

► Encryption configuration

Cancel

Save

3

Test event [Info](#)

SaveTest

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

☐ Edit saved event

Event name

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

☐ Shareable

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

s3-put

Format JSON

Click on **TEST**

Code	Test	Monitor	Configuration	Aliases	Versions
------	------	---------	---------------	---------	----------

▼ Details

The area below shows the last 4 KB of the execution log.

Prepared by: Shavez Khan
BCA 2nd Semester

STEP 5 :

Create Lambda Function


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** ("shvzmainbucket").
- 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".

[Lambda](#) > Add triggers

Add trigger

Trigger configuration [Info](#)

 **S3**
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

Bucket region: ap-south-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

Upload an image file to the source S3 bucket named “shvzmainbucket” After a brief wait, inspect the destination bucket called “shvzresizedbucket”. You should find two images there: one thumbnail and one cover photo.




Serverless image processing application has been created successfully

6. CONCLUSION

In this project, we successfully implemented a serverless image processing application using AWS S3 buckets. By uploading an image file to the source bucket “**shvzmainbucket**”, the system automatically processed the image and generated two versions—a thumbnail and a cover photo—which were then stored in the destination bucket “**shvzresizedbucket**.” This automation not only showcases the efficiency and scalability of serverless architectures but also highlights their potential for handling real-time data processing tasks without the need for dedicated server management.

Overall, this project demonstrates the effectiveness of serverless solutions in simplifying complex workflows, reducing operational costs, and enhancing application scalability and reliability. The successful completion of this project marks a significant step forward in leveraging cloud technologies for efficient and automated image processing.

7. REFERENCES

-  [Geeksforgeeks](#)
-  [Amazon Web Services](#)
-  [Github](#)