Serverless Image Processing

“Serverless Image Processing with AWS Lambda and S3”

Prepared by: Shavez Khan
BCA 2nd Semester

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

Prepared by: Shavez Khan
BCA 2nd Semester

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.

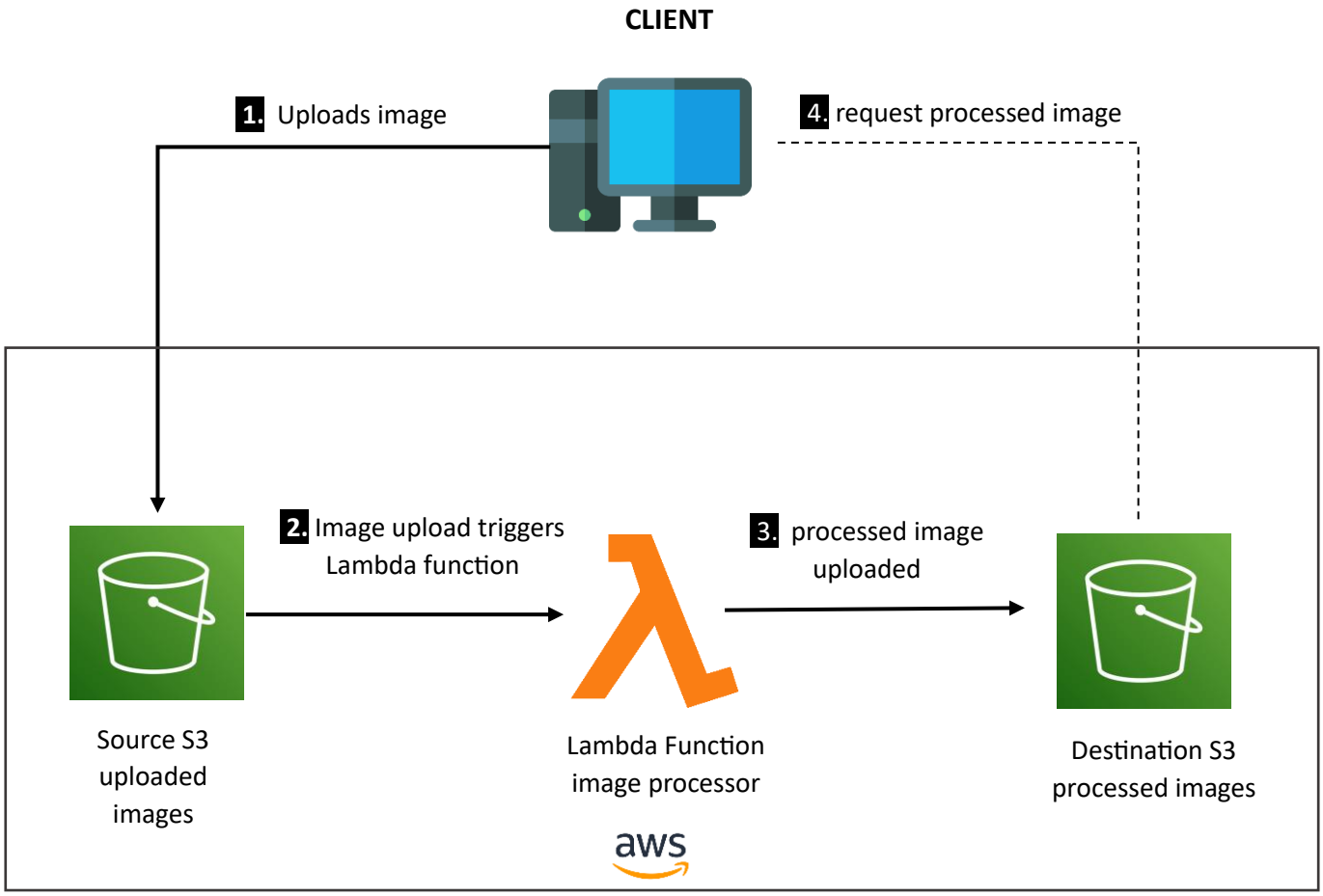
4. WORKFLOW

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.

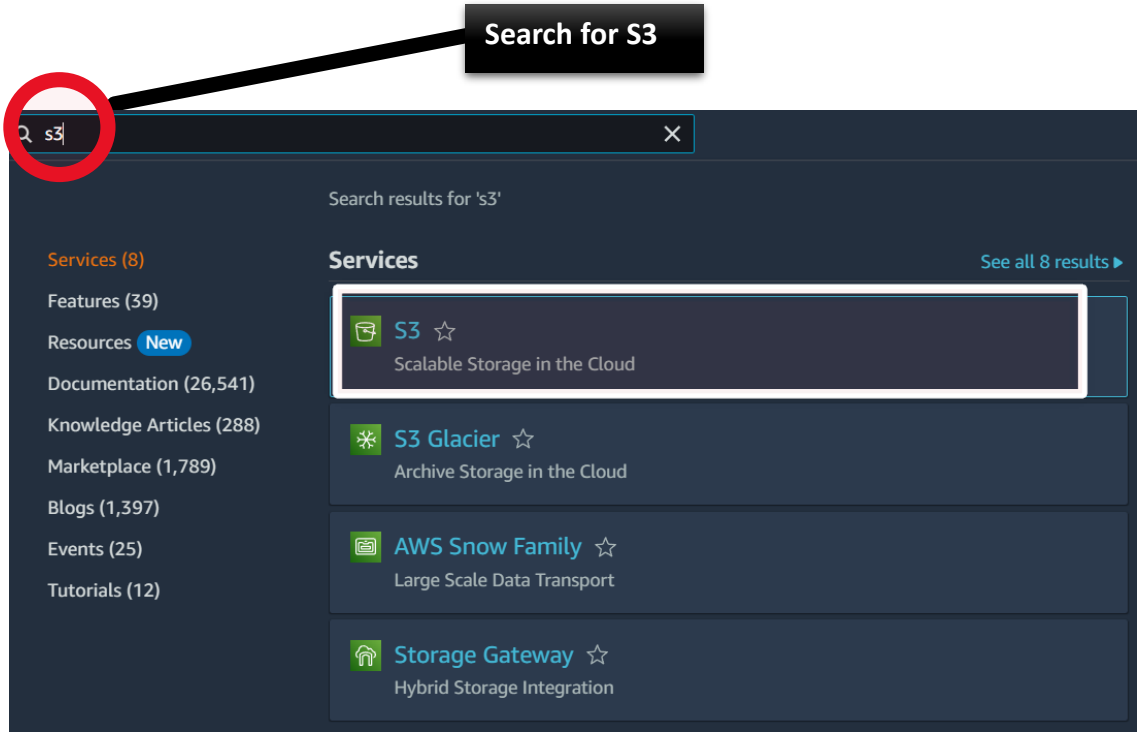
4. Storage and Retrieval: The processed image is saved to a destination S3 bucket, from where the user can request and retrieve the processed image.

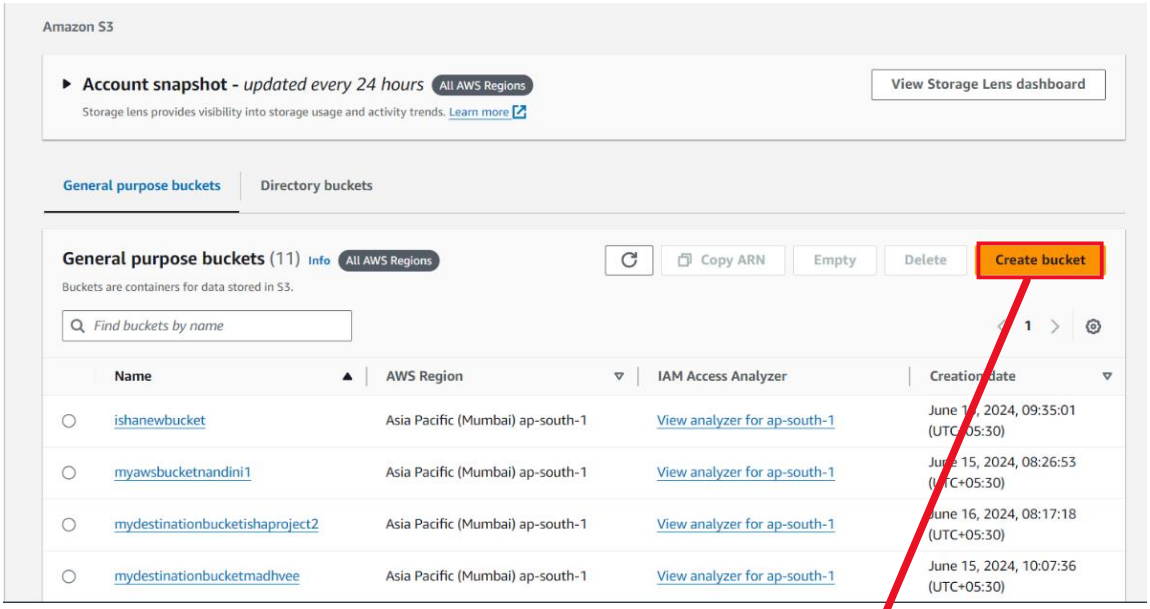


5. STEPS FOR DEPLOYMENT

STEP 1 :

Create S3 Buckets

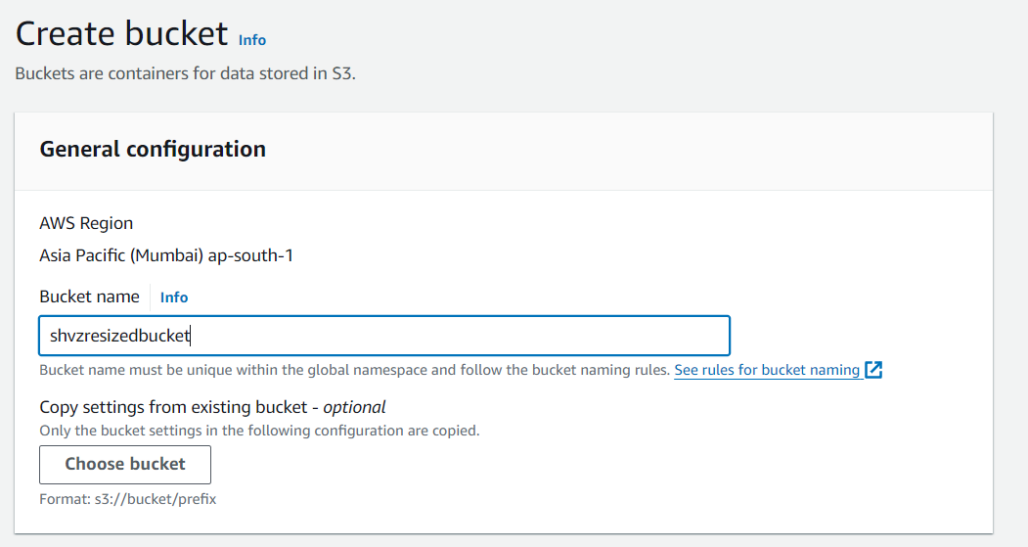




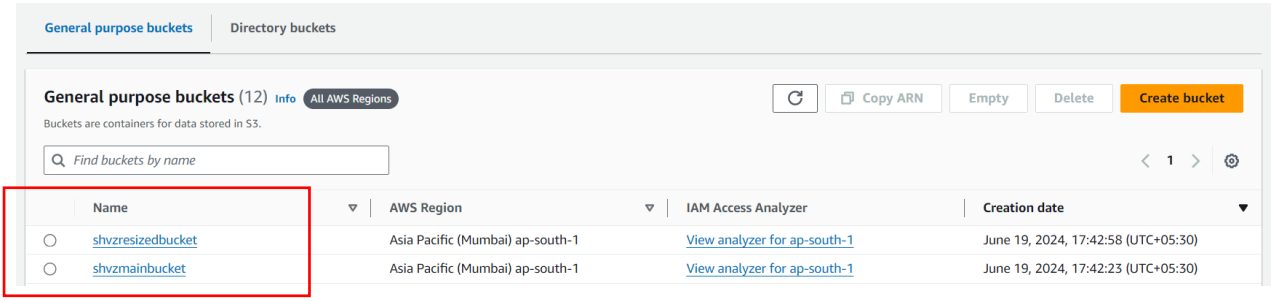
Click here to create a bucket

Prepared by: Shavez Khan
BCA 2nd Semester

Follow the same steps and create another bucket



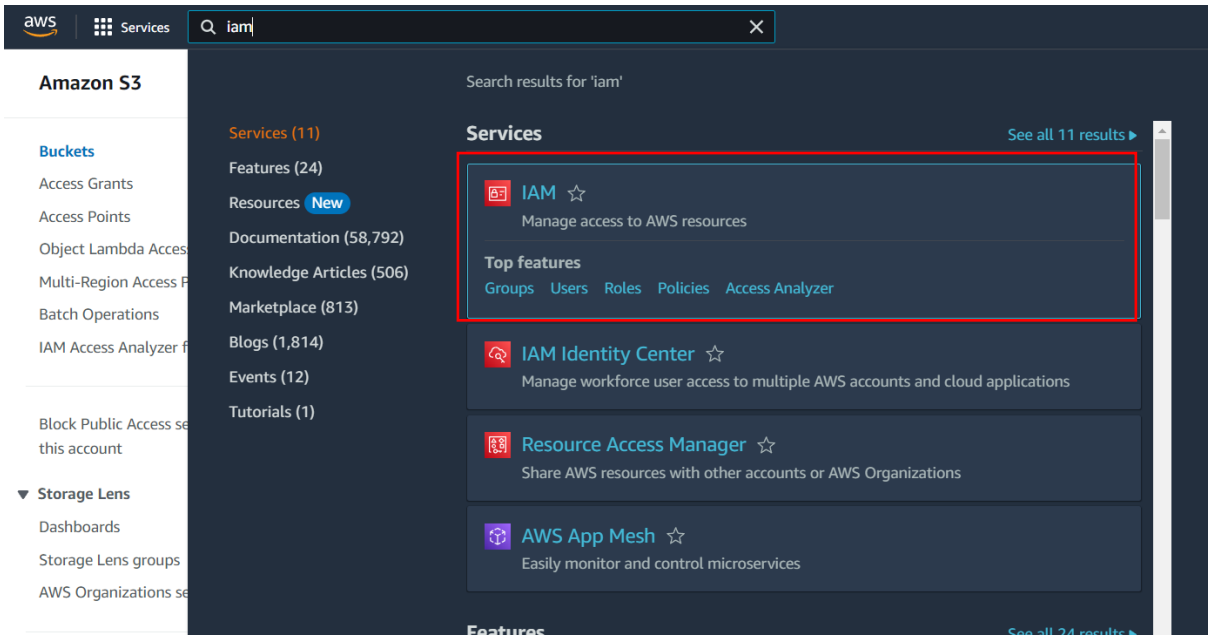
Two buckets have been set up for storing and processing images.



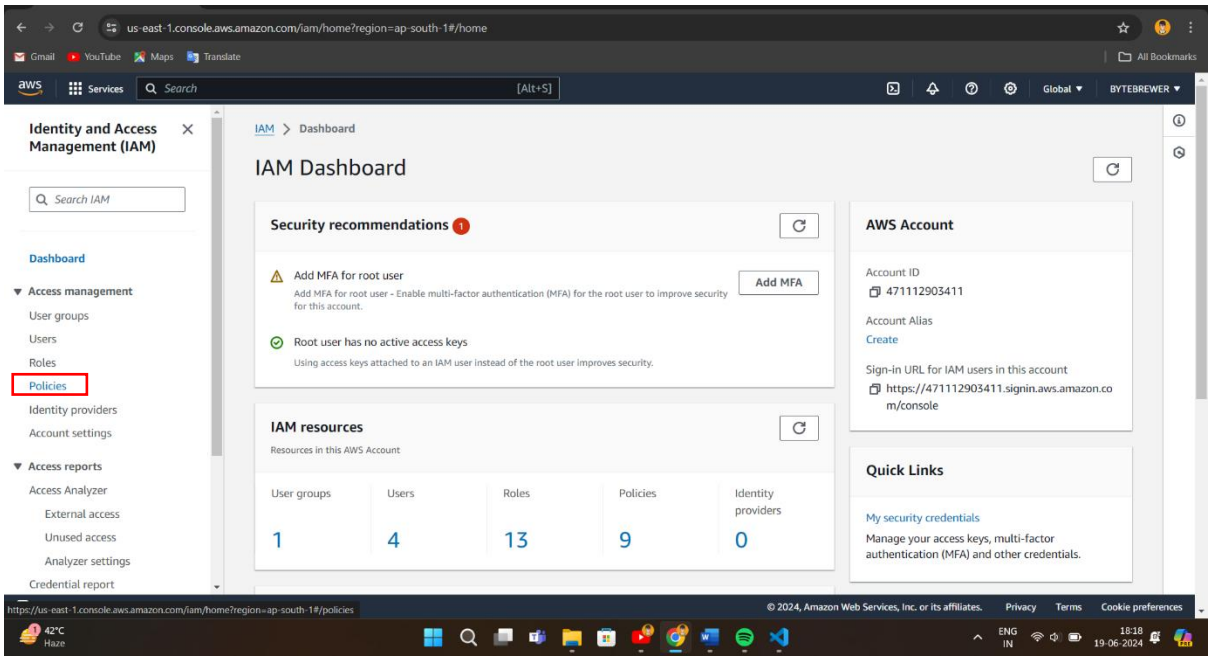
STEP 2 :

Create S3 Bucket Policy

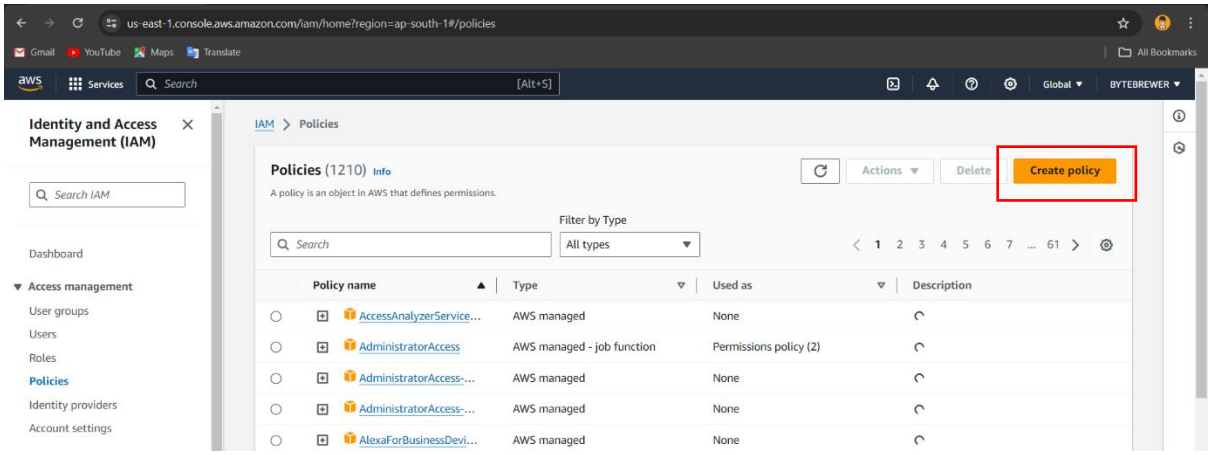
Search for ***IAM*** > Click on ***IAM***



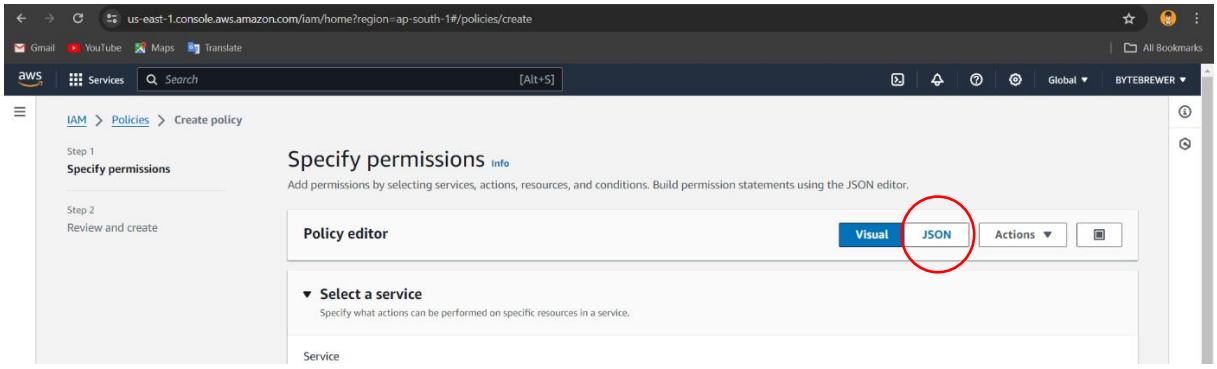
Click on **Policies** > Click on **Create Policy**



Prepared by: Shavez Khan
BCA 2nd Semester



Click on **JSON**

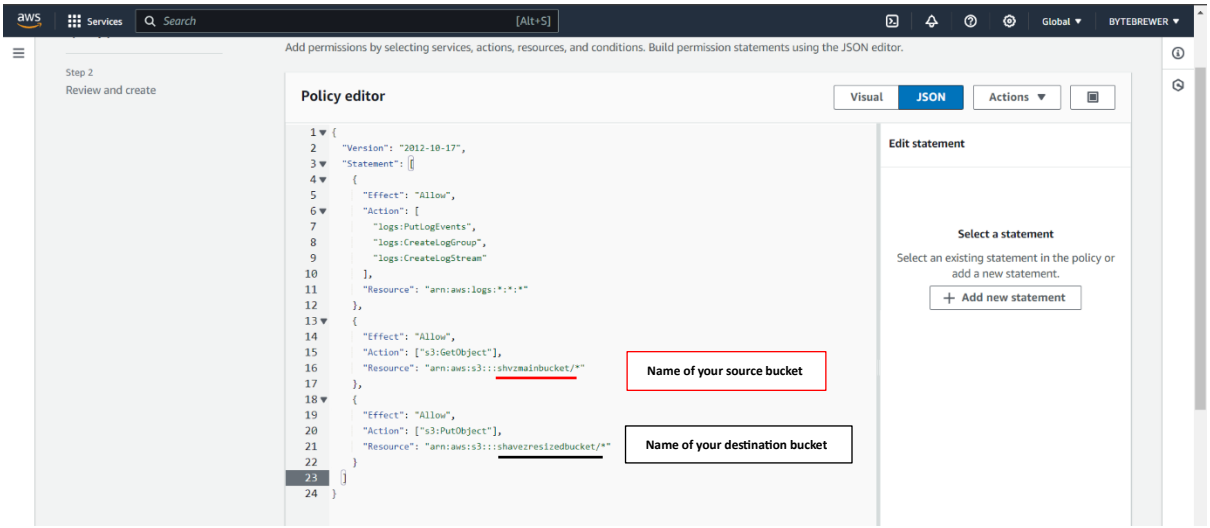


Copy the following JSON code

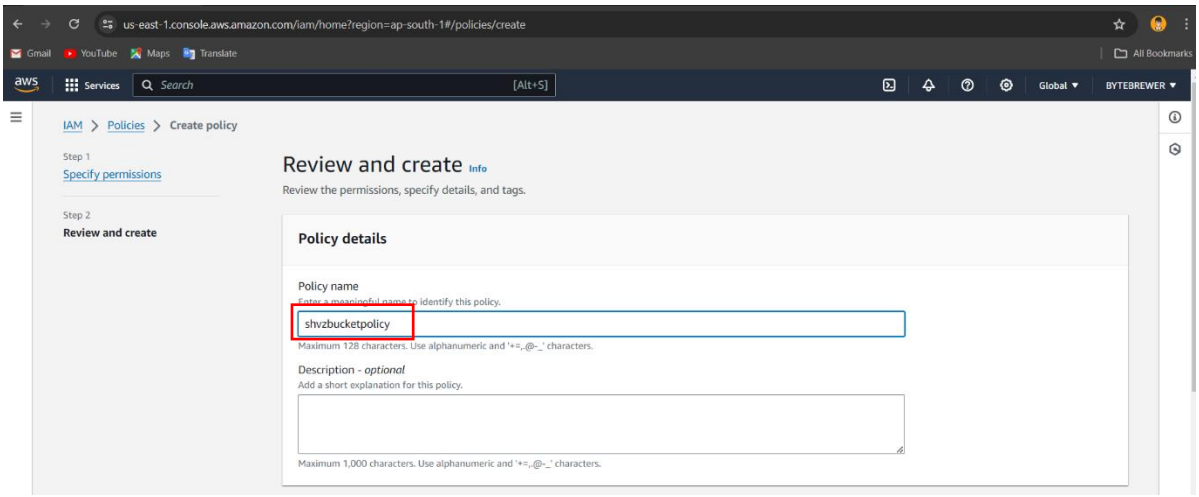
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],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

Prepared by: Shavez Khan
BCA 2nd Semester

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



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

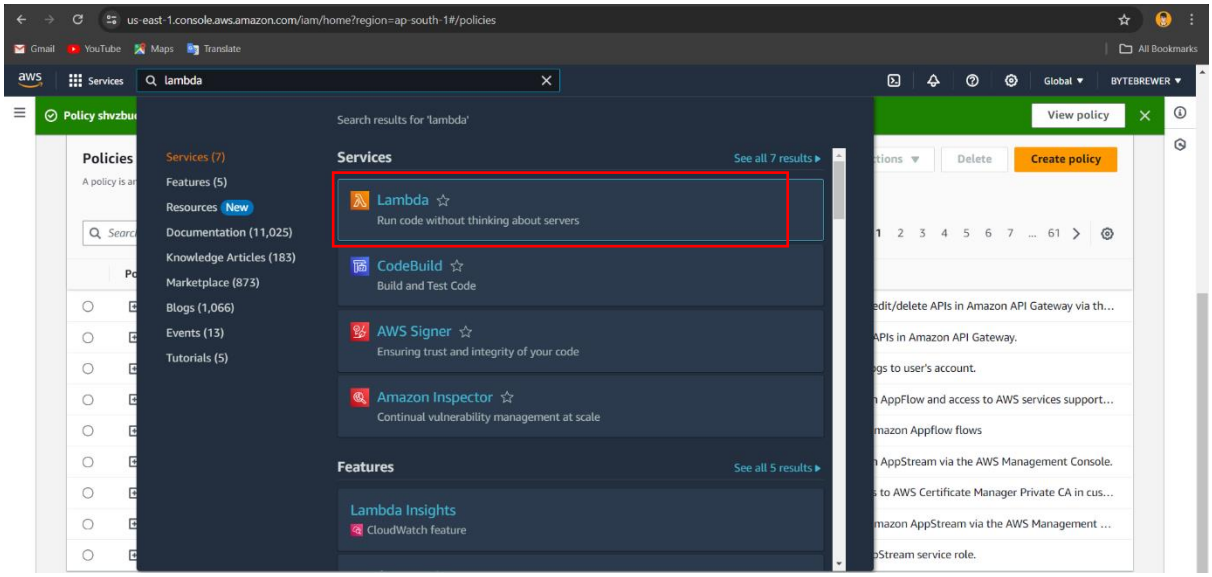


Prepared by: Shavez Khan
BCA 2nd Semester

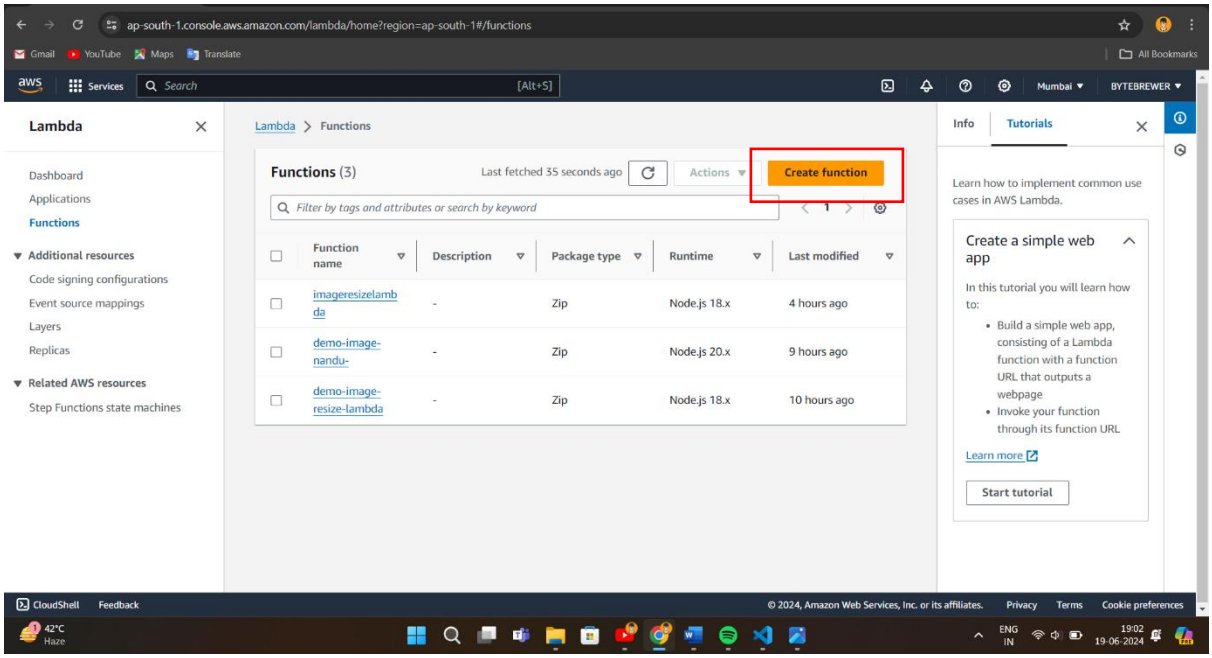
STEP 3 :

Create Lambda Function

- Search for **Lambda**

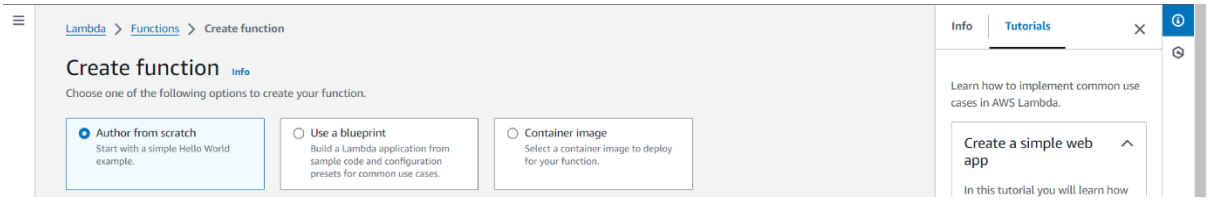


- Click on **Create Function**

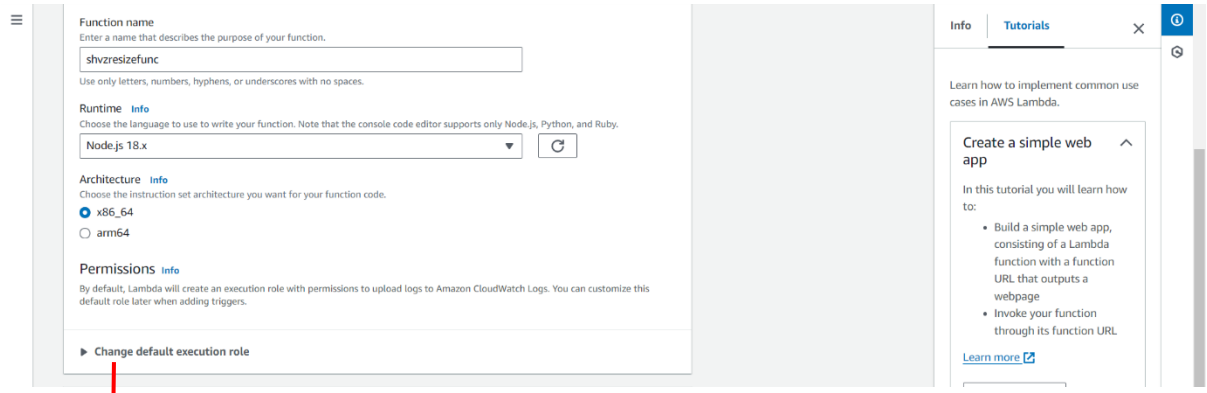


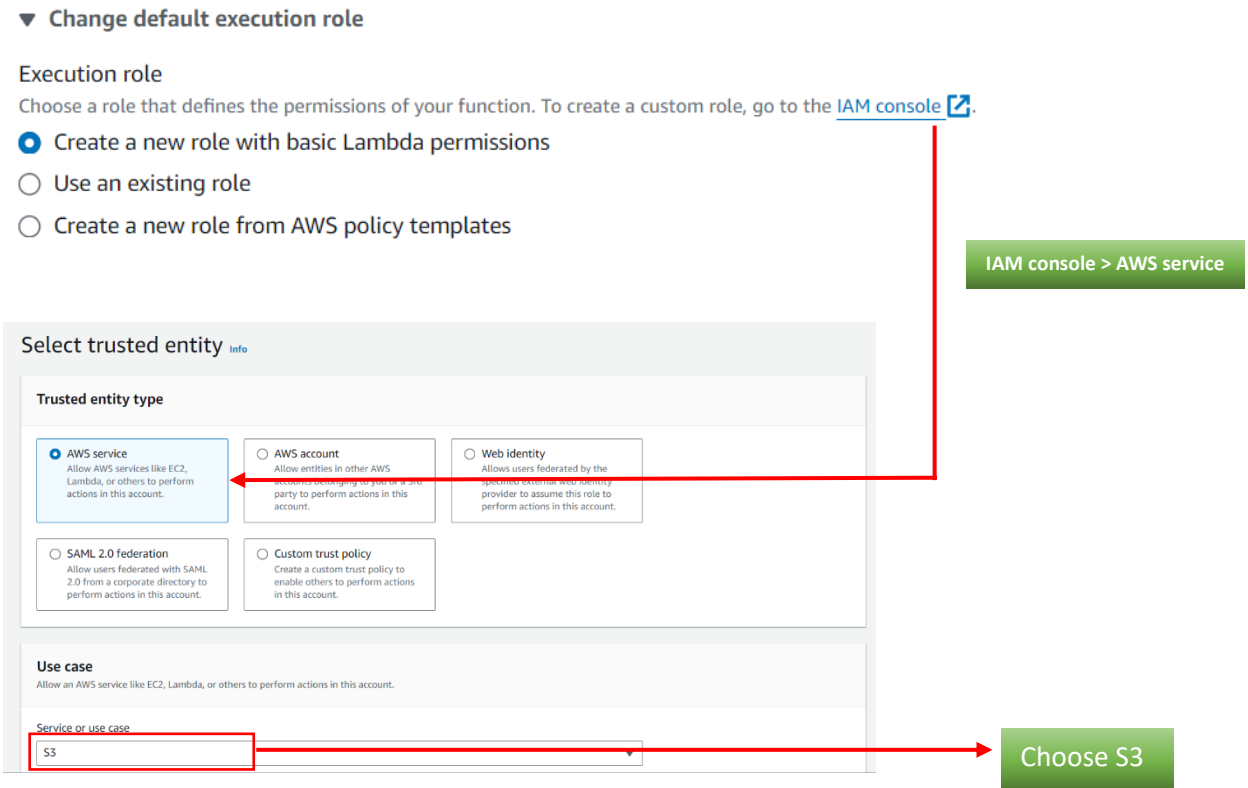
Prepared by: Shavez Khan
BCA 2nd Semester

- Select **Author from Scratch**



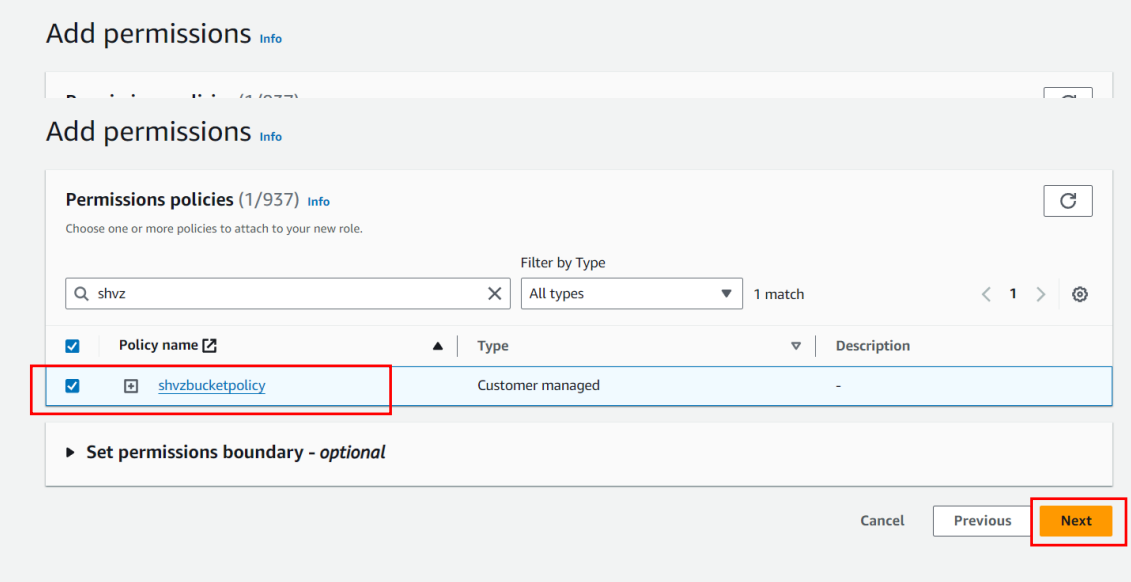
- Give a name to your Function and Choose the language to use to write your function (Here I'm using Node.js 18.x)
- Click on Change default execution role > IAM console



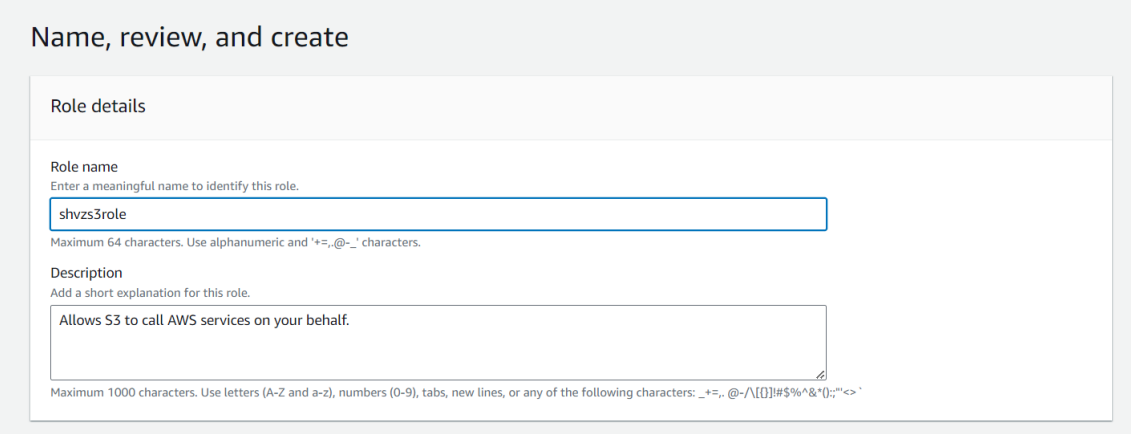


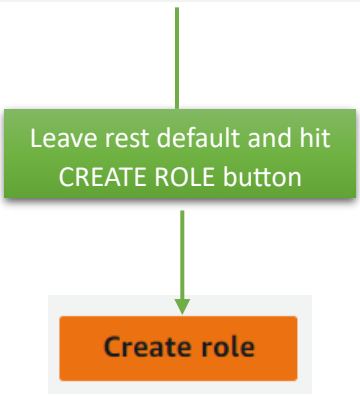
Prepared by: Shavez Khan
BCA 2nd Semester

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

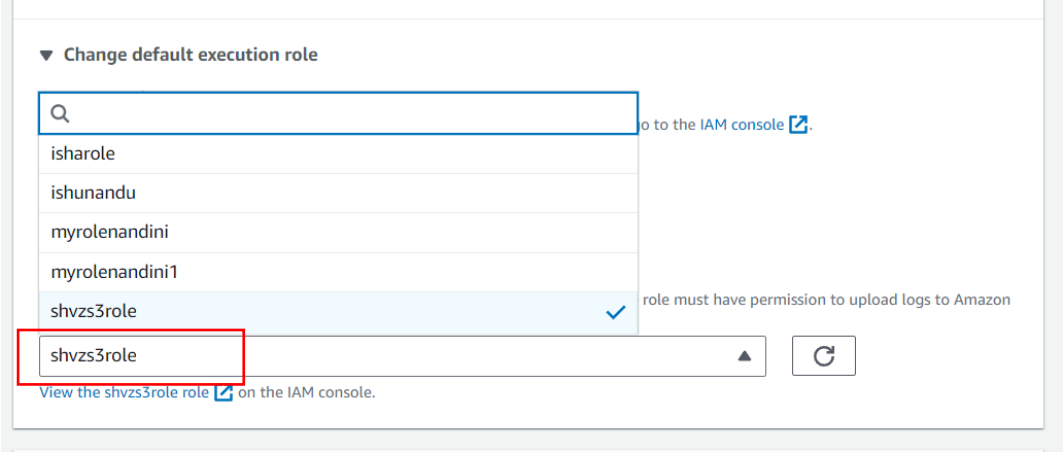


Click on **NEXT** > Give a meaningful name to IDENTIFY this **ROLE**.

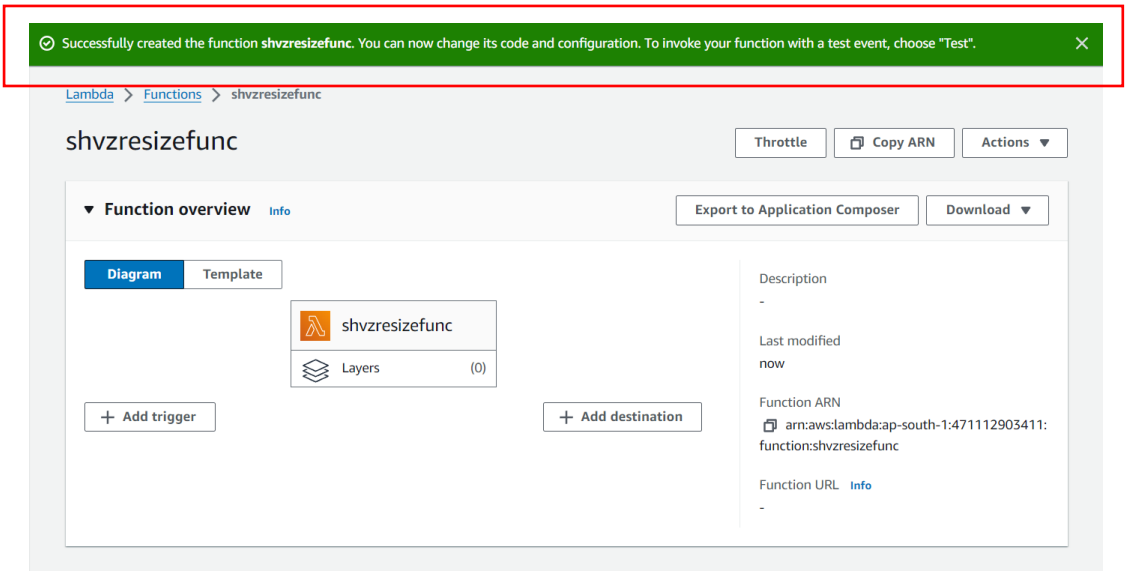




- Head back to **Create function** tab, choose ***"use an existing role"*** and select the role we just created now then **HIT** the Create Function Button.



- **HIT** the Create Function Button.



STEP 4 :

Test The Application

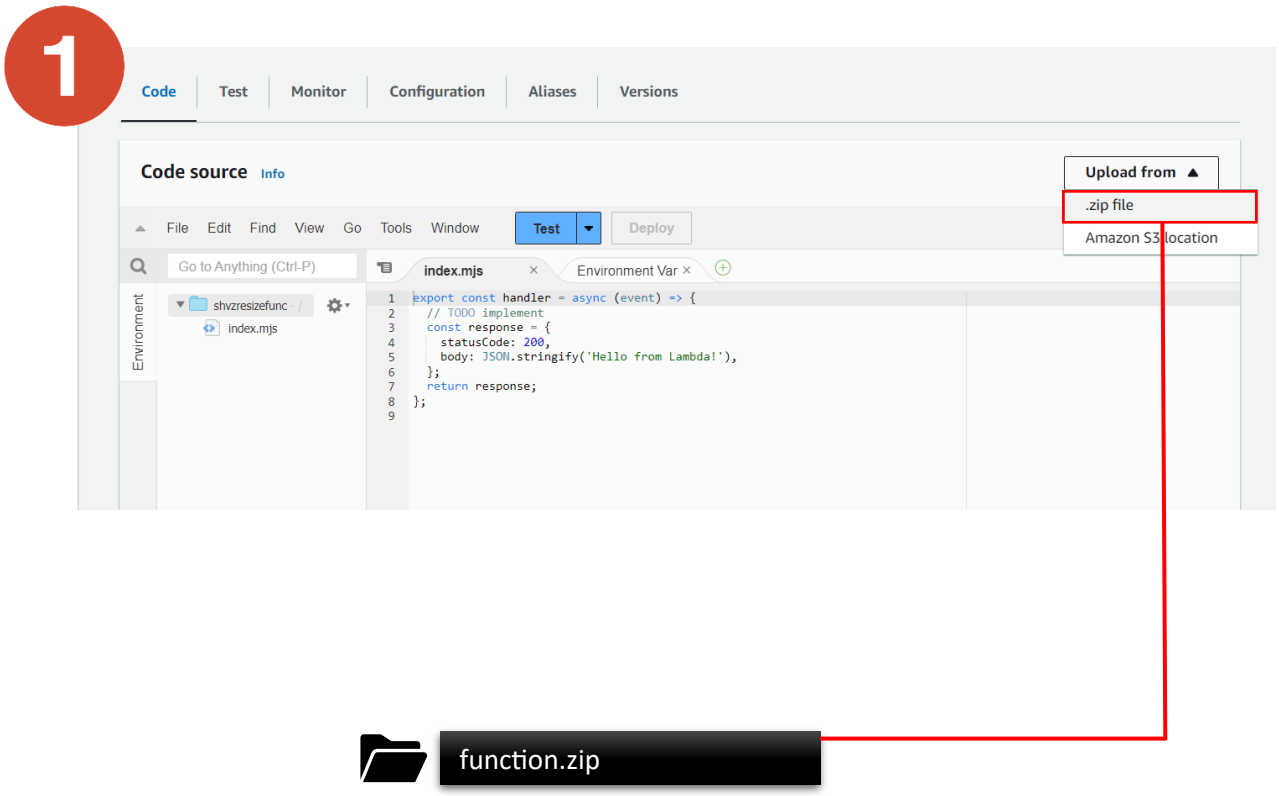
1. Upload the *function.zip* file that contain the code to resize images.

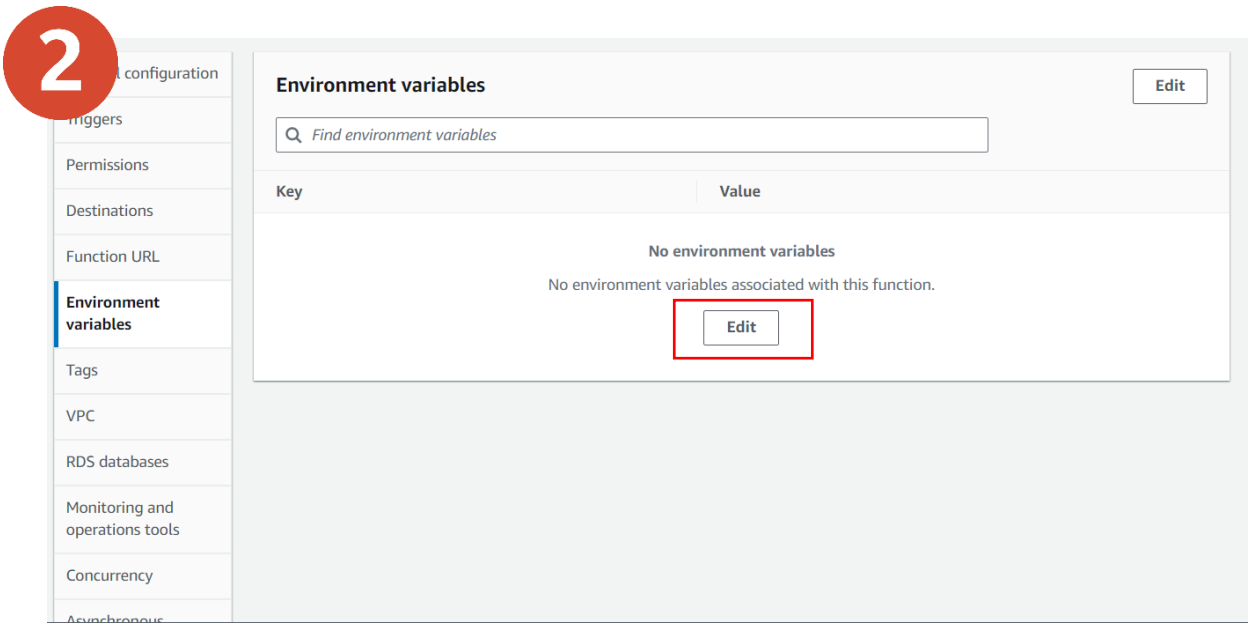
2. Add the **Environment Variable**

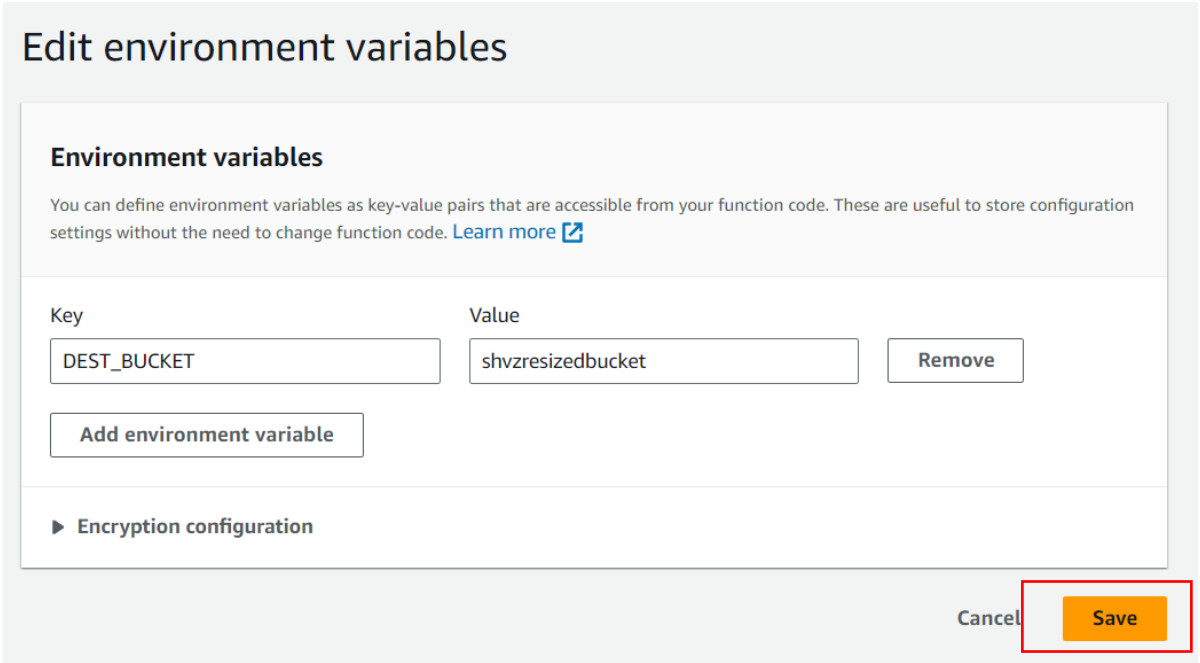
Key: as per you code *(Here, it is **DEST_BUCKET**)*

Value: Destination Bucket Name *(Here, it is **shvzresizedbucket**)*

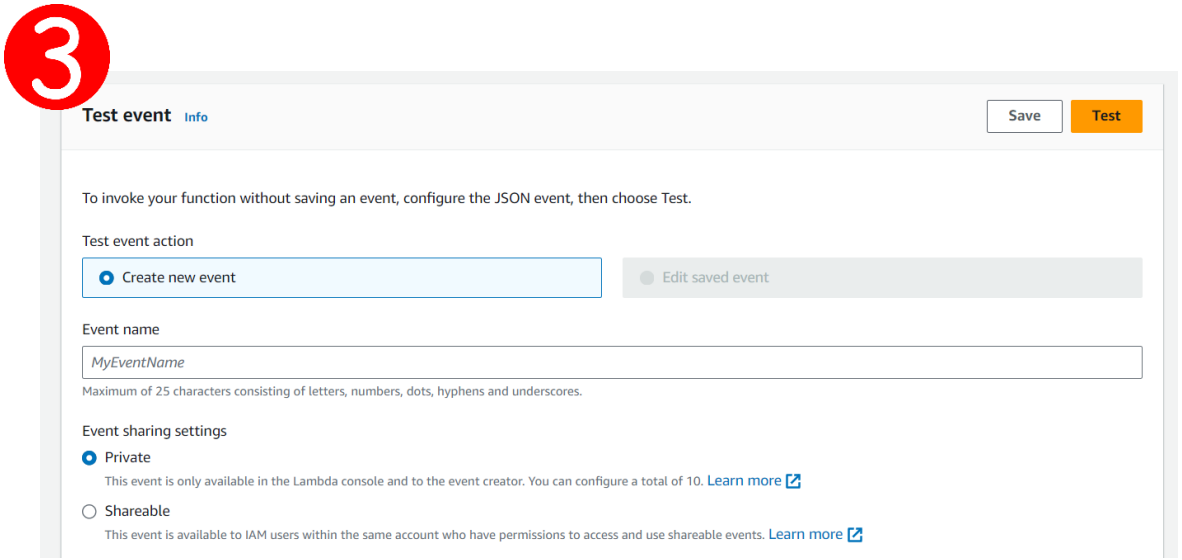
3. Test the Event

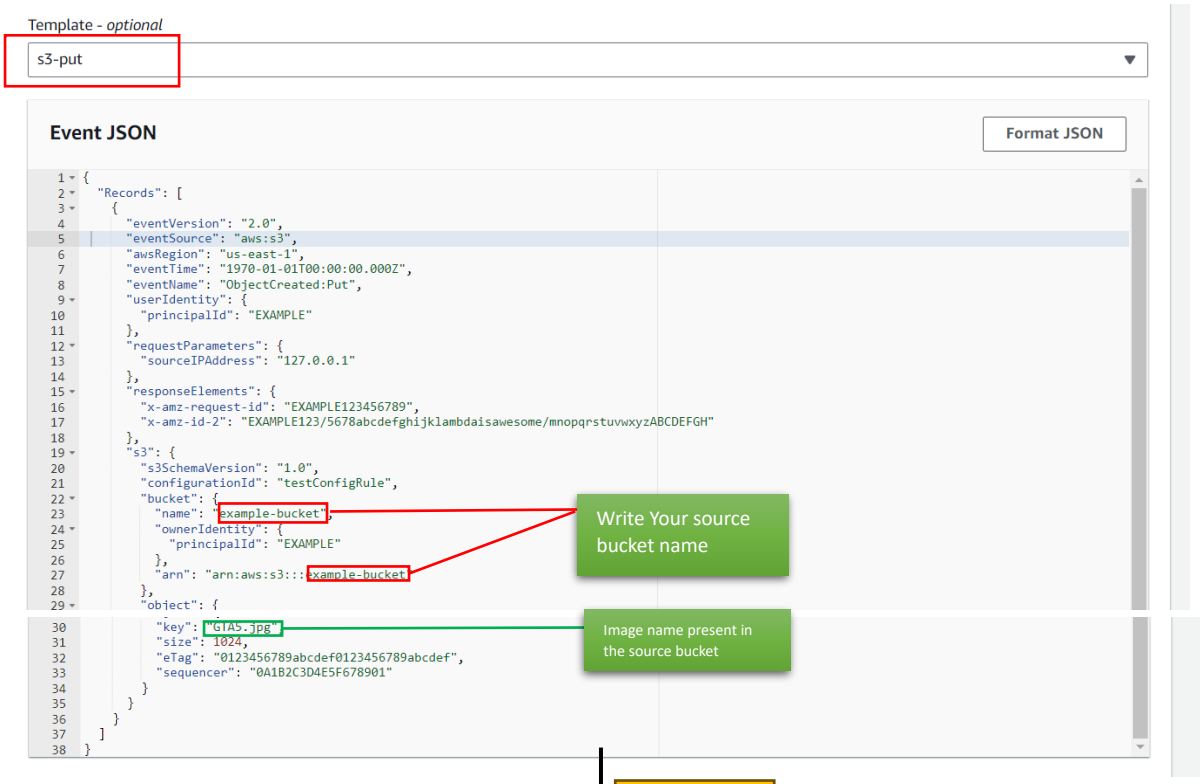


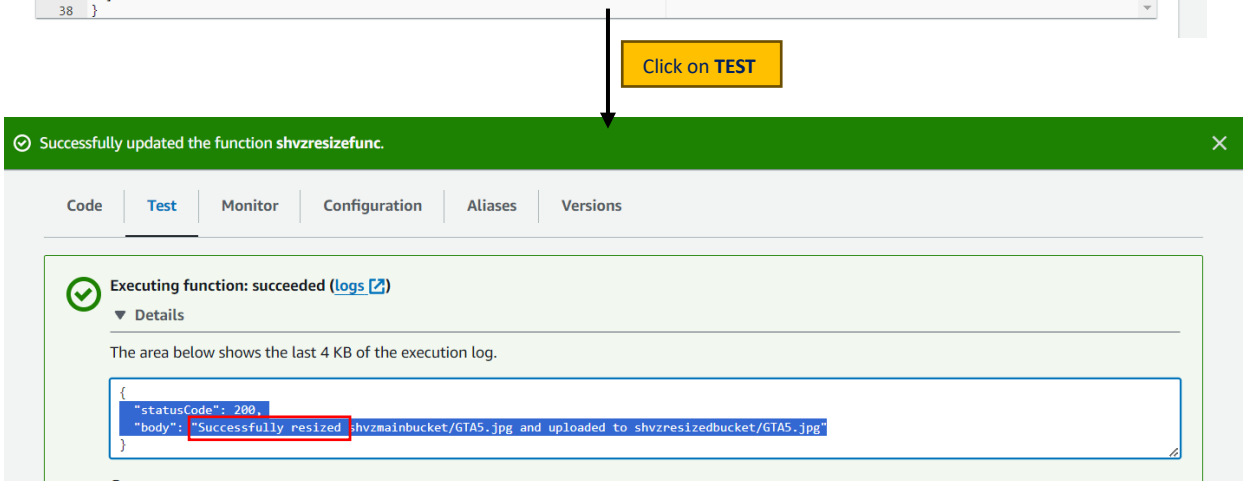




Prepared by: Shavez Khan
BCA 2nd Semester







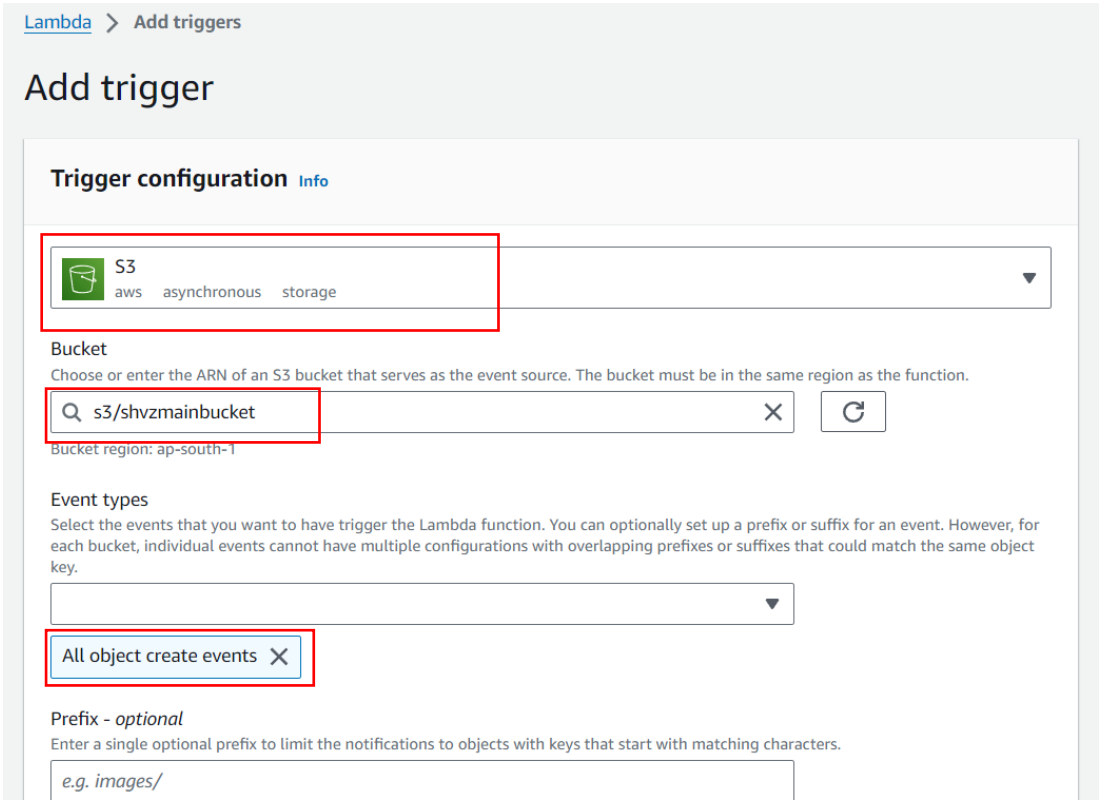
Prepared by: Shavez Khan
BCA 2nd Semester

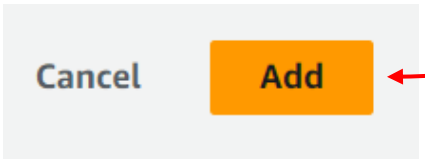
STEP 5 :

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”.





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