**Project 1**

**Serverless Image Processing**

### 📑 **Table of Contents**

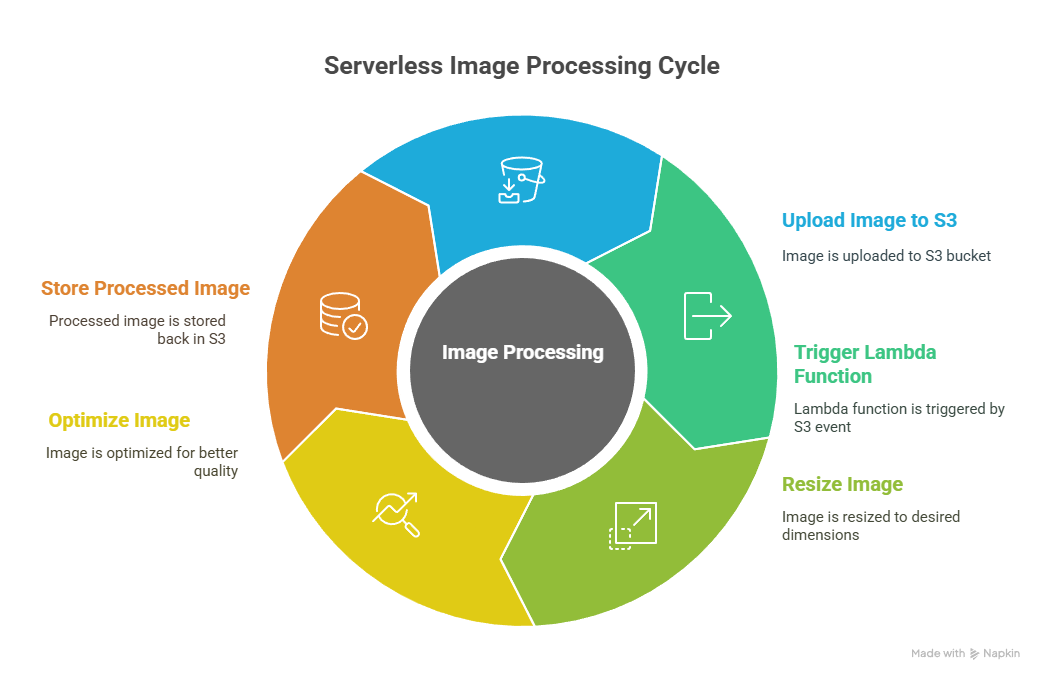
1. **Introduction**
2. **Architecture Overview**
3. **Components Required**
4. **Folder Structure**
5. **Step-by-step setup**
6. **Optional Enhancement**

**Introduction**

### 🌐 What is Serverless Image Processing?

**Serverless image processing** is a scalable, on-demand approach to manipulate images (like resizing, compressing, or converting formats) without managing any servers. Instead of provisioning and maintaining infrastructure, you use cloud services like **AWS Lambda**, **Amazon S3**, and **CloudWatch**.

**Architecture Overview**

1. **User uploads an image to an S3 bucket.**
2. **S3 triggers a Lambda function** upon new object creation.
3. **Lambda resizes and optimizes the image** using a library like Sharp.
4. **Processed image is stored in a separate S3 bucket or prefix**
5. **Logs/metrics are captured using CloudWatch.** 

**🧱 Components Required**

* **Amazon S3**: To store original and processed images.
* **AWS Lambda**: To run the image processing code.
* **Amazon CloudWatch**: For monitoring and debugging.
* **IAM Role**: With permissions to read/write S3 and log to CloudWatch.
* **Node.js + Sharp Library**: For image processing.

**📦 Folder Structure**

bash

CopyEdit

image-resizer/

│

├── index.js # Lambda function code

├── package.json # Node.js dependencies

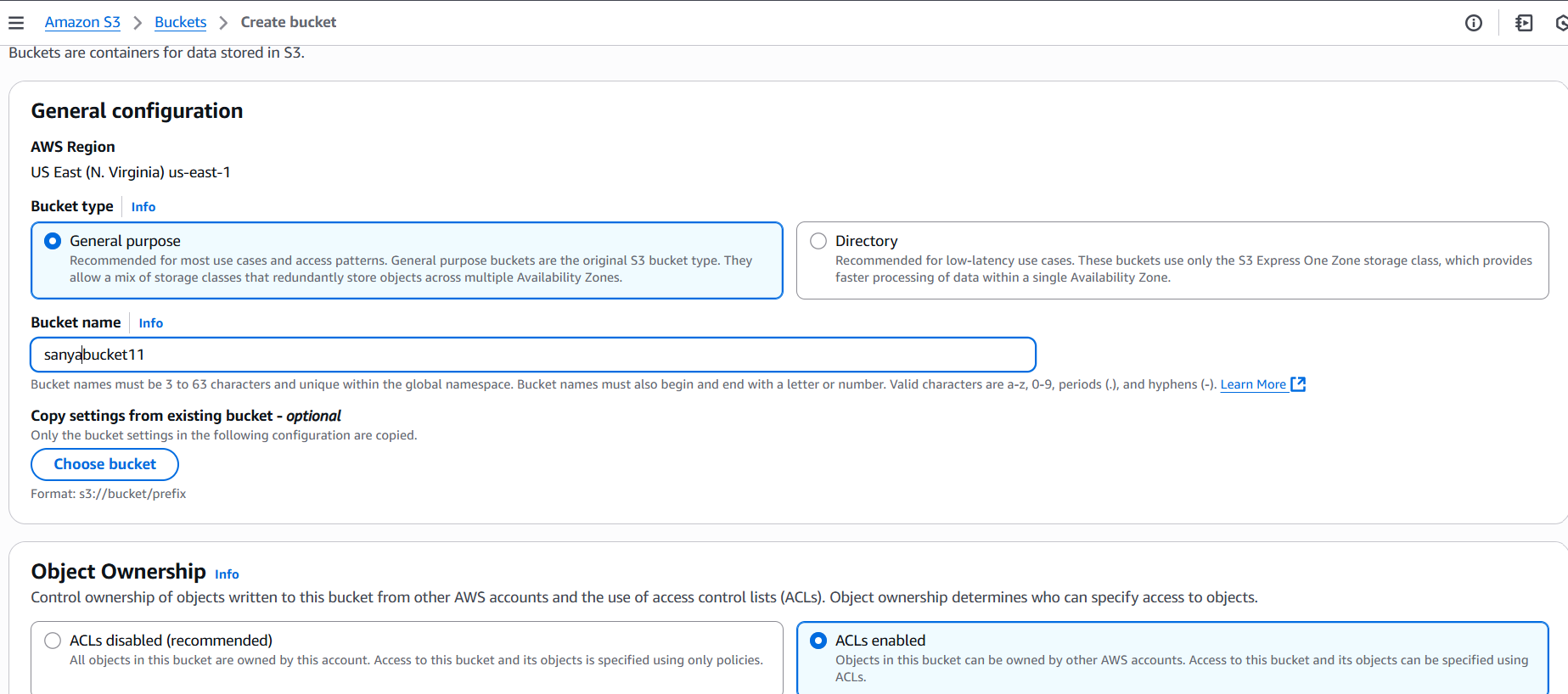
└── layer/ # (Optional) for bundling Sharp library as a

Lambda layer

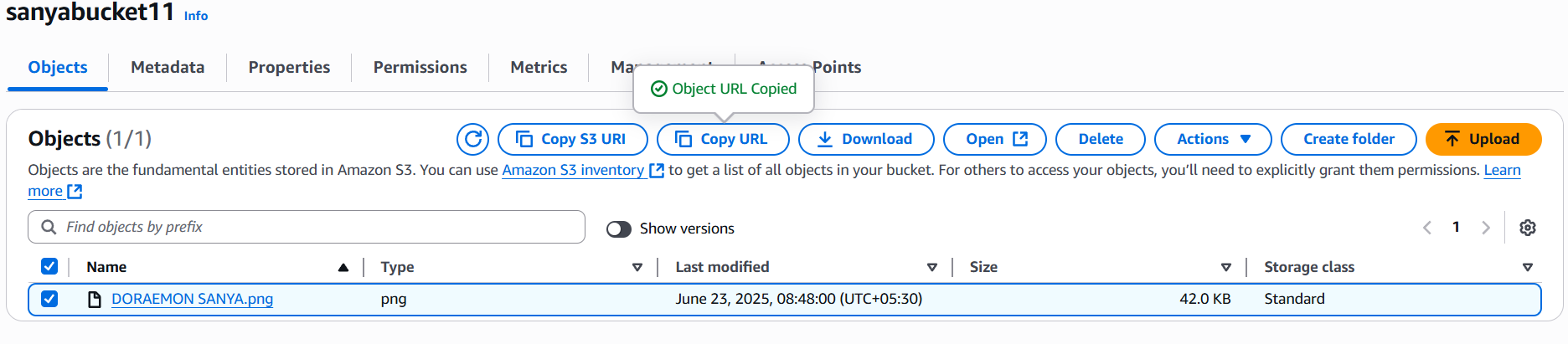
## 🧠 Step-by-Step Setup

### 1. **Create S3 Buckets**

* original-images-bucket — for uploads.



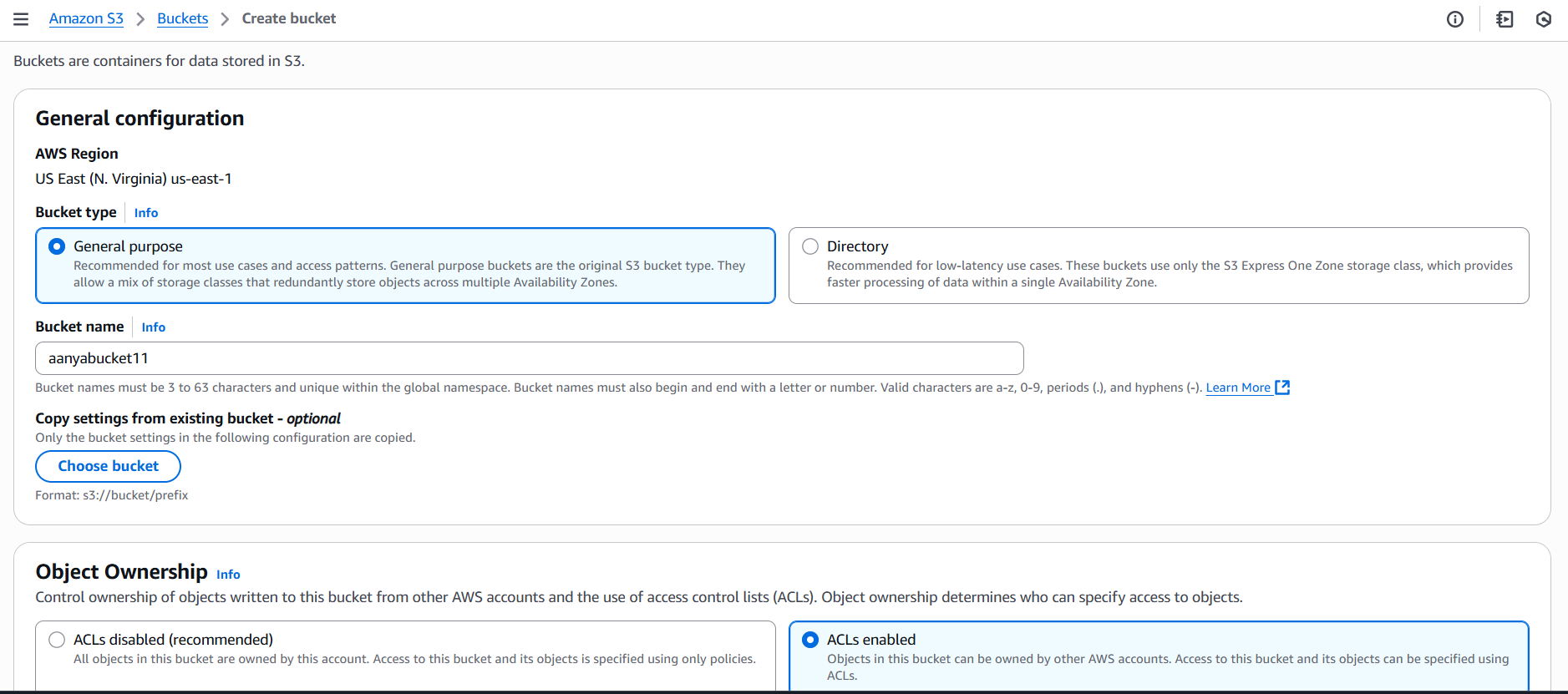
* upload image in a S3 bucket which created to upload original bucket.



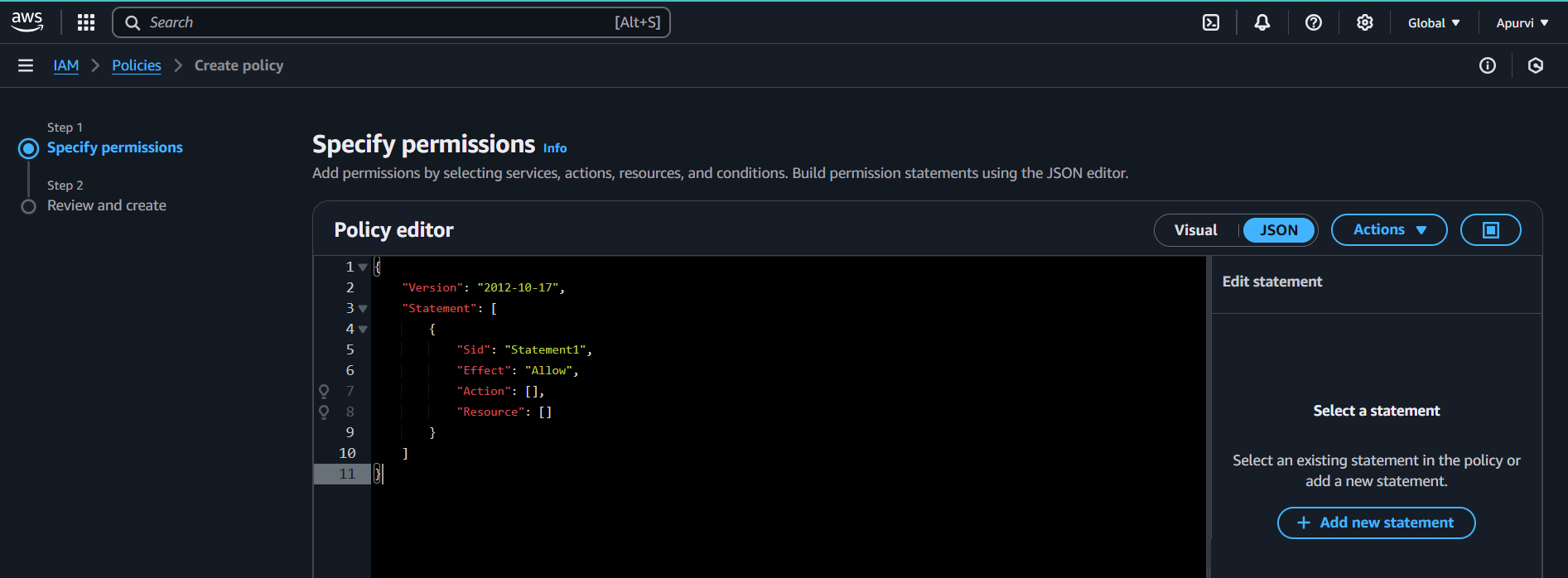
* check that image is opening in the



* this is how image looks without resizing with lambda function.
* resized-images-bucket — for output images.



**Setting up of IAM policy**

1. Go to IAM dashboard and then go to IAM policy.
2.  Click on create policy and choose JSON.
3. Change the given code and replace with this 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",

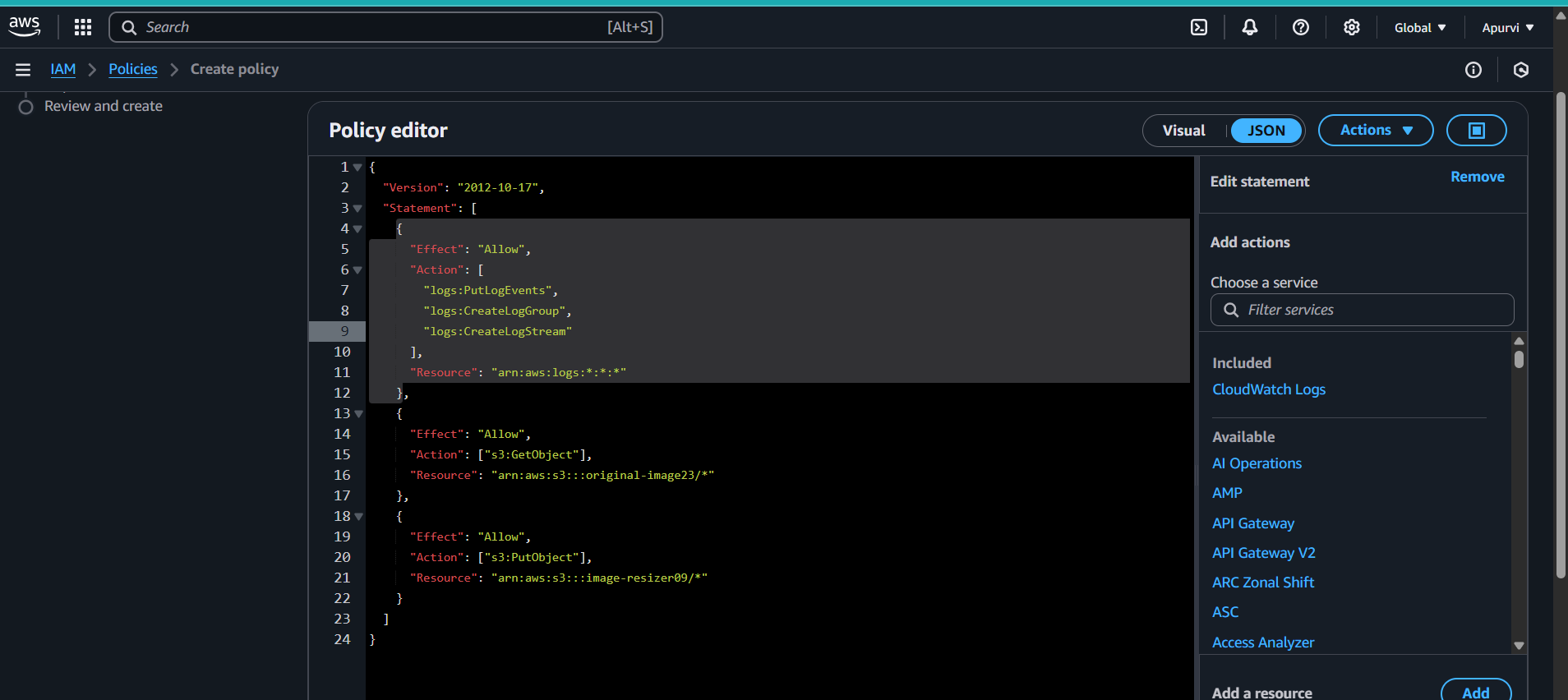
"Action": ["s3:PutObject"],

"Resource": "arn:aws:s3:::DEST\_BUCKET/\*"

}

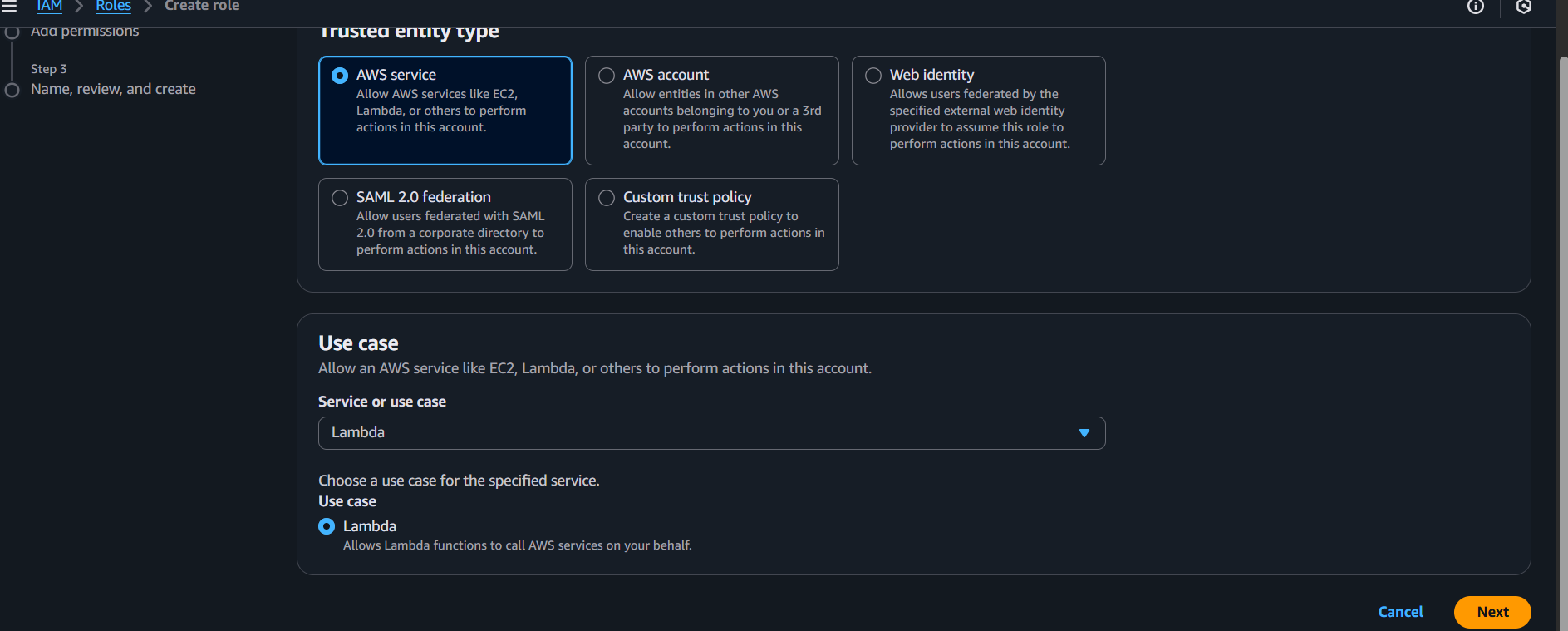
]

}

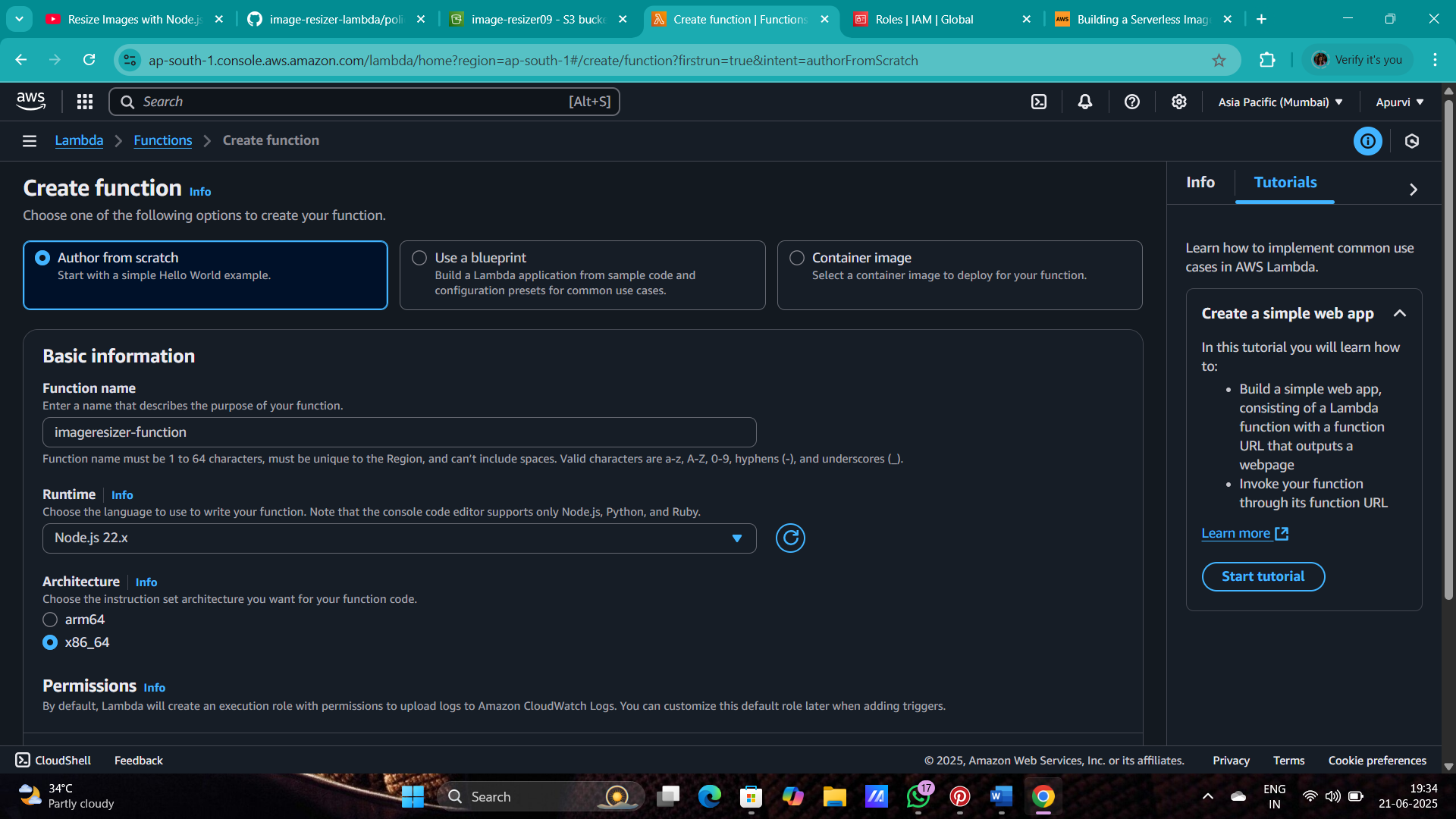


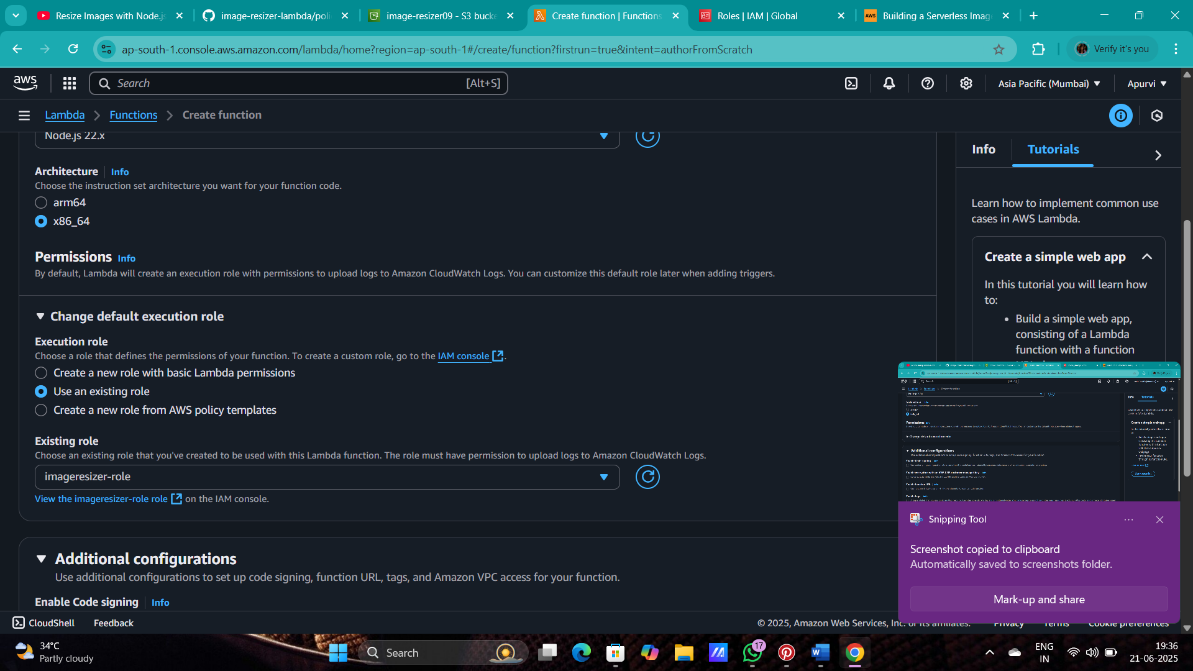
1. Replace the bucket name with your bucket where you have uploaded the image and DEST\_BUCKET with the resizer bucket name.
2. And create the policy.

**Setting up of IAM role**

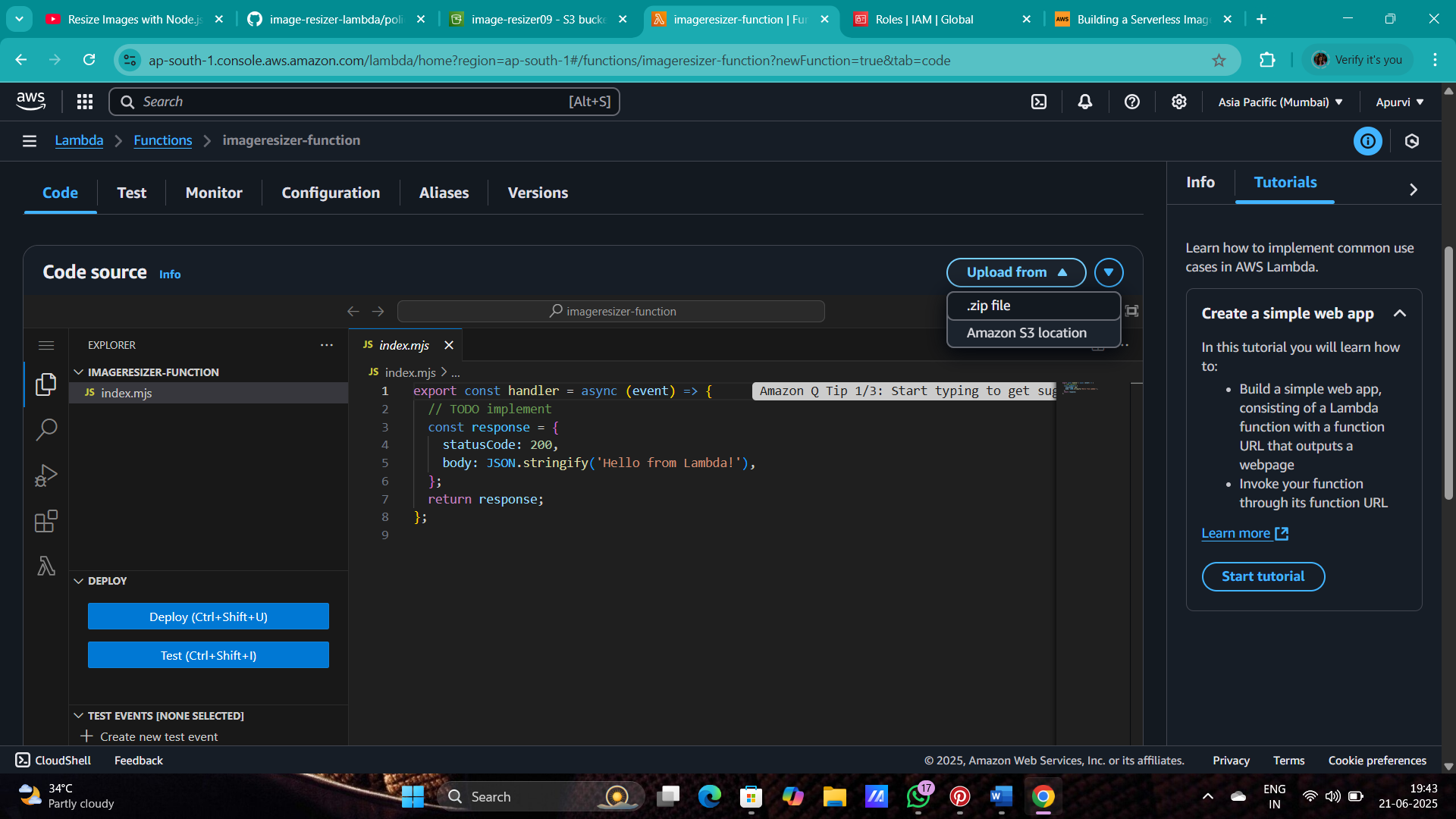
1. Click on IAM role.
2. Choose lambda in use-case
3. Add the policy which you have created and create the role.

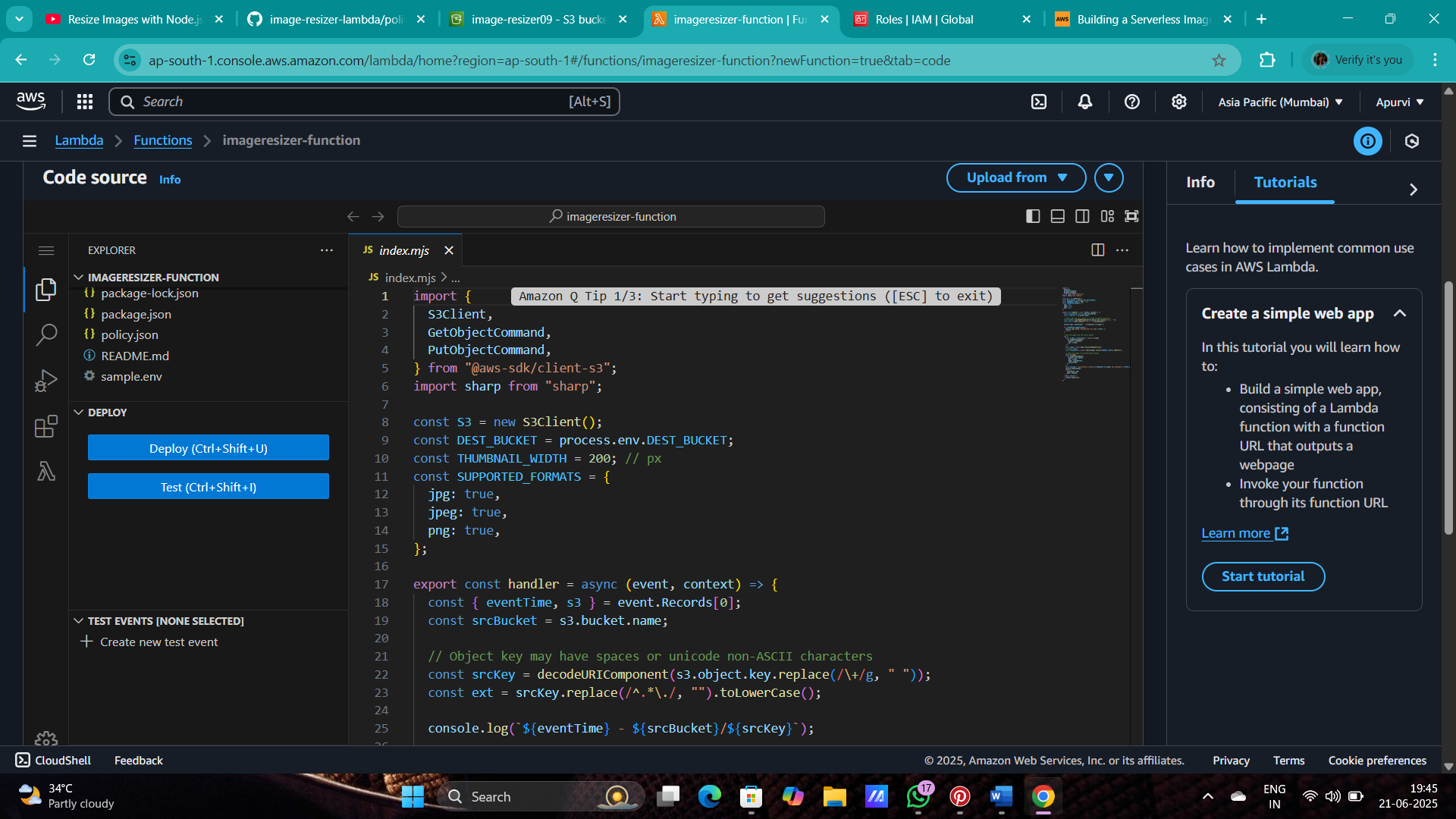
**Making of lambda function**

1. Go to lambda and click on create function.
2. Now add the role which created.



1. And create the lambda function.
2. Now after creating the function , going to the source code and uploading the ([C:\Users\apoor\Downloads\function.zip](file:///C:\Users\apoor\Downloads\function.zip) .) zip file.

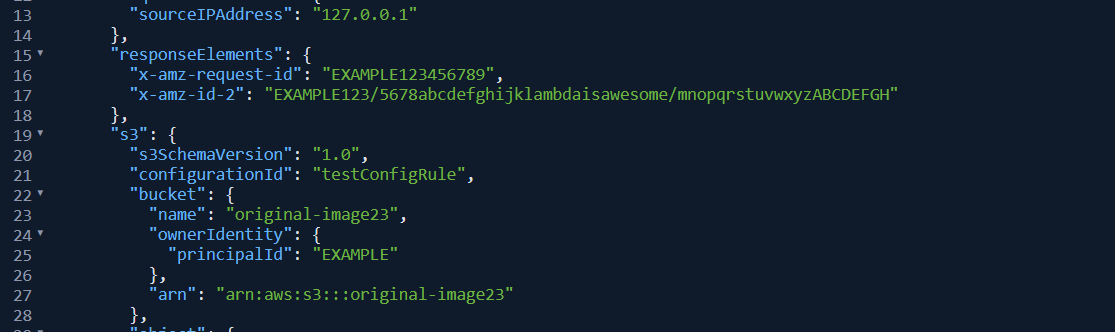


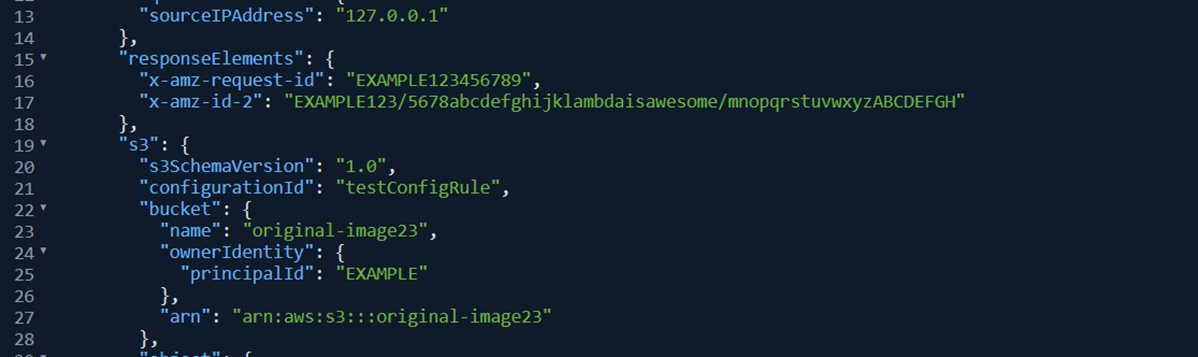


1. Your code will look like this.
2. Now go in configuration and click on environment variables and edit it and click on add environment variables.
3. And then goes to test event in a code source.

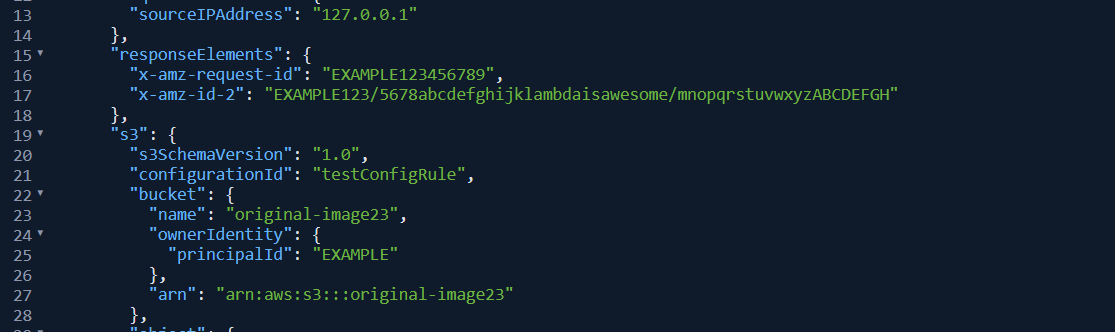


1. now go in the event JSON and the edit the bucket with your bucket where you have uploaded

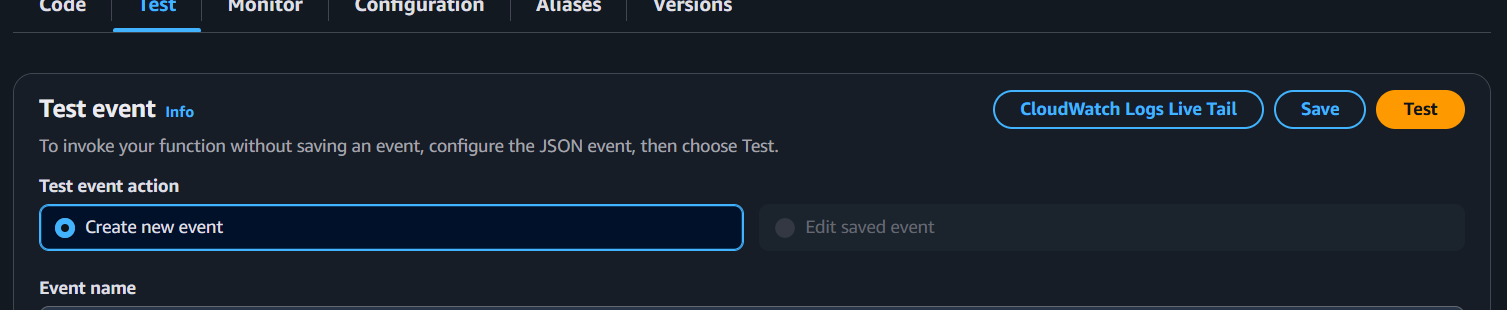


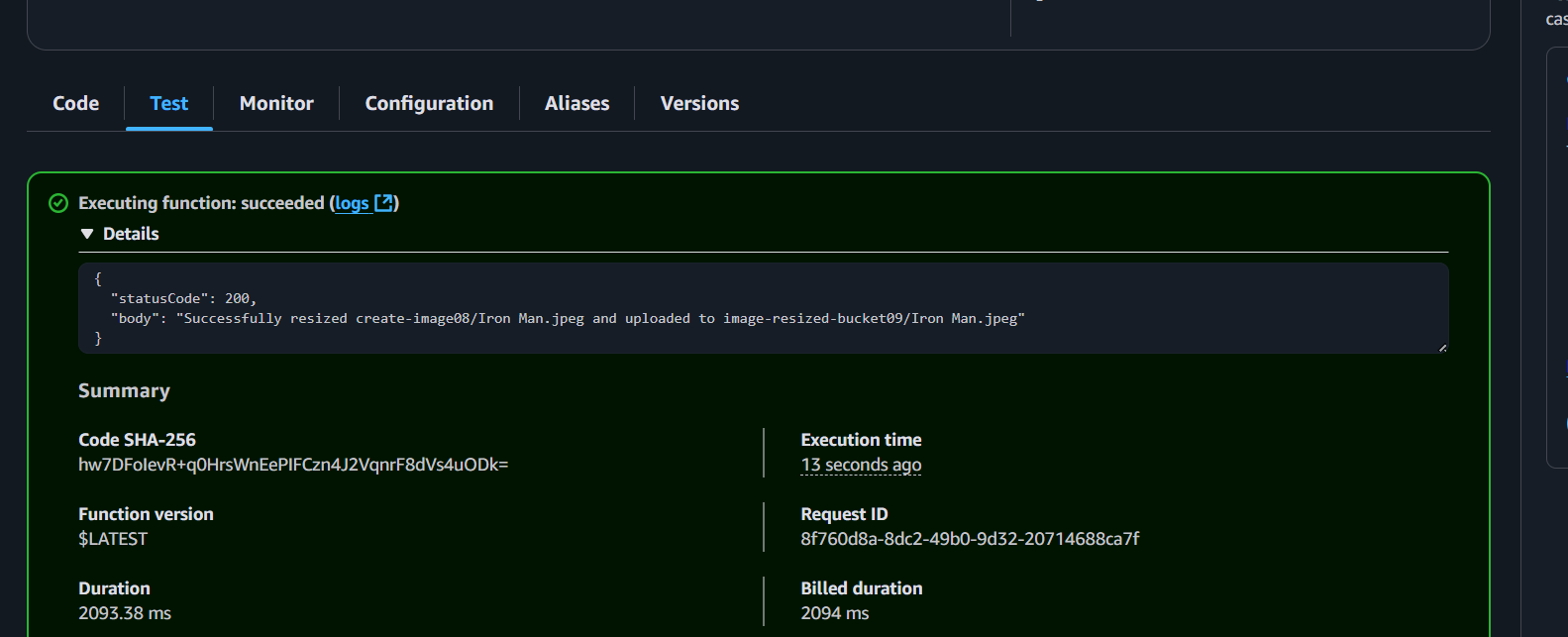


1. now change the key name with the image name which you have uploaded in s3 bucket.

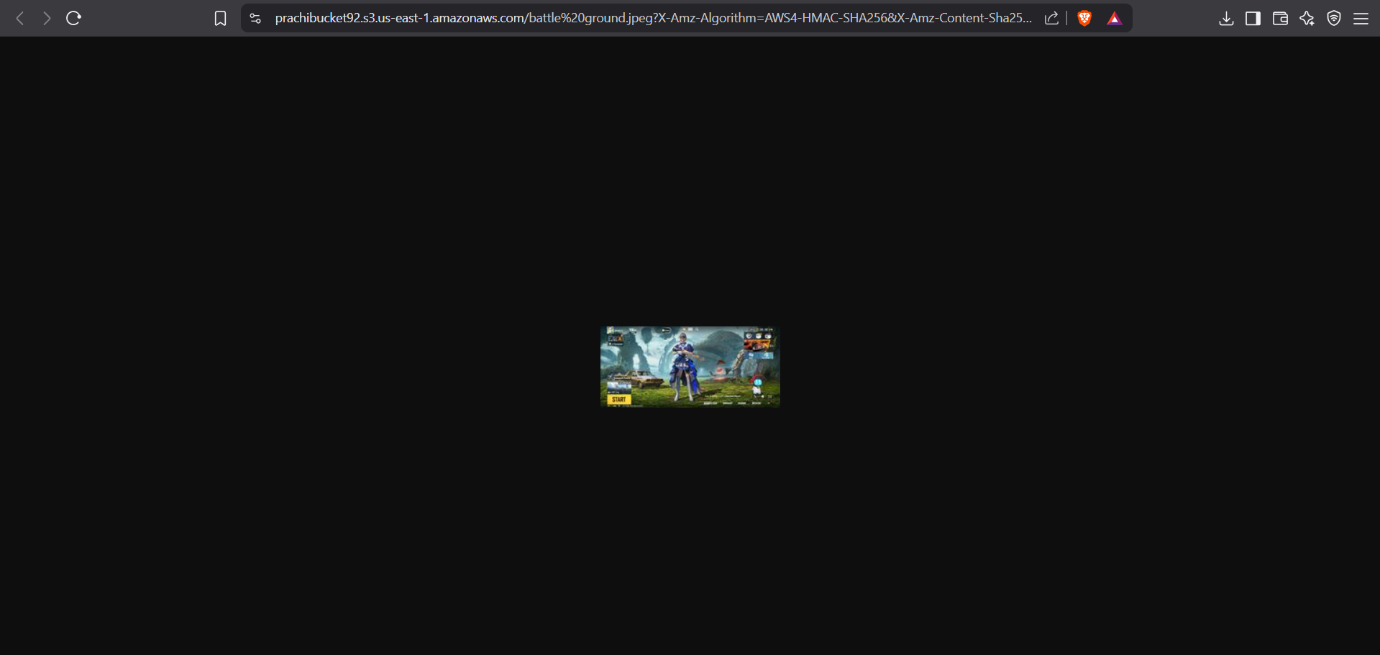


10. Copy the name of image and paste it in the code.



11. now test the code .

12. now the code has been executed and image has been resized to check go to the second bucket which you have created and open the image .



## ✅ Optional Enhancements

* Resize to multiple dimensions (e.g., thumbnail, medium, large)
* Add file type conversion (e.g., PNG to WebP)
* Use S3 Object Metadata for dynamic size config
* Deploy via **AWS SAM** or **CDK** for versioned, scalable deployment