**Project-1**

# **Creating Serverless Image Processing with AWS Lambda and S3**

**Table of contents**

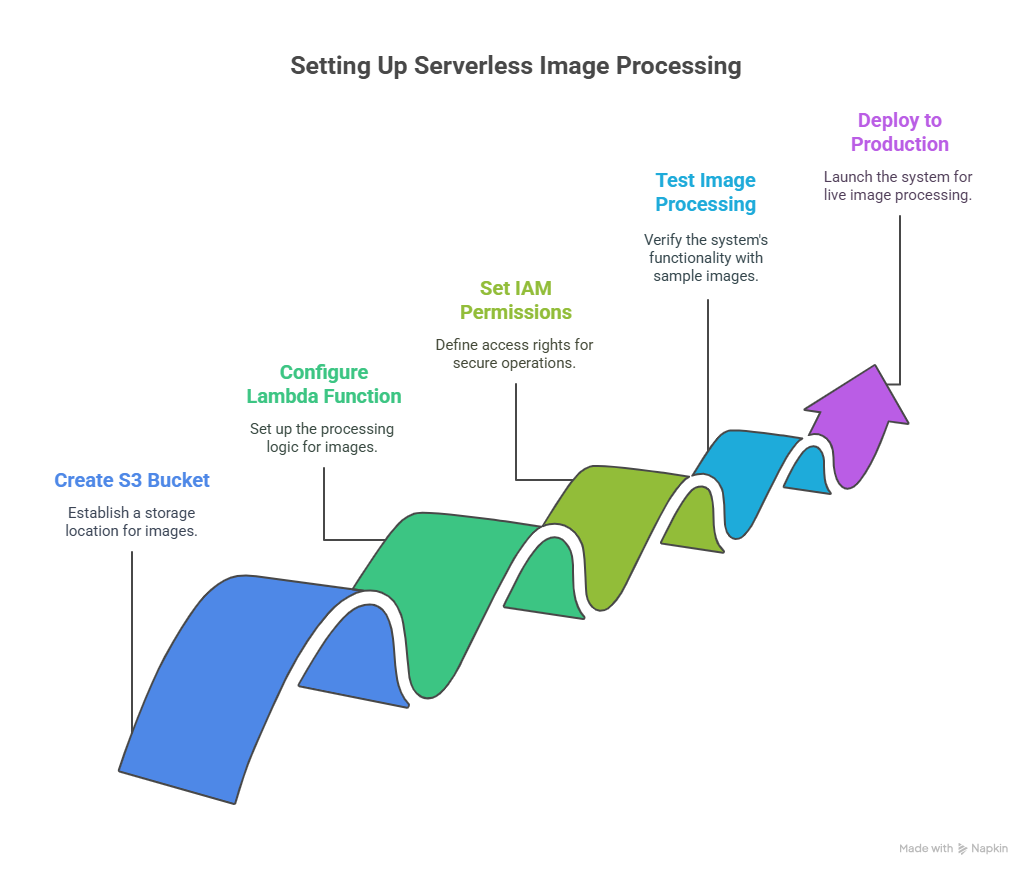
* Introduction
* Architecture overview
* Folder structure
* Step by step guide

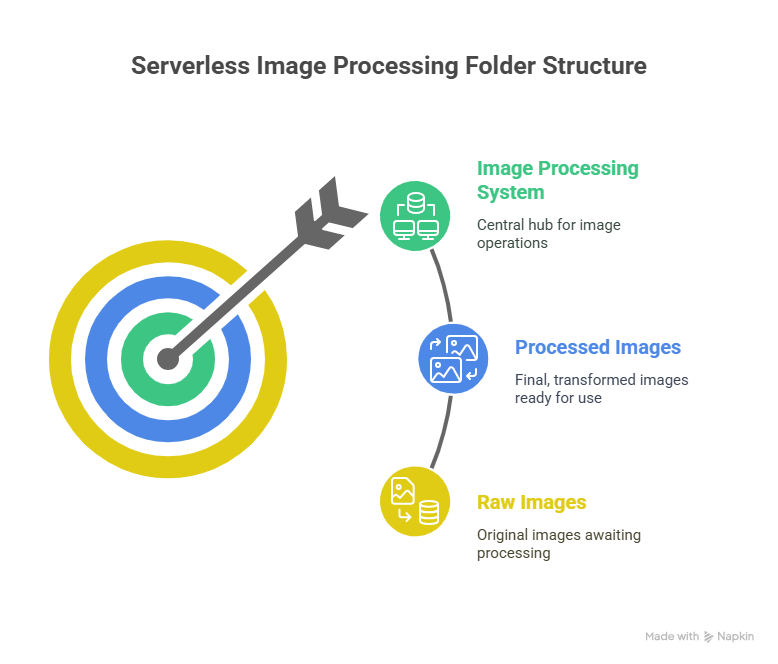
**Introduction**

In today's digital landscape, image processing tasks—such as resizing, compressing, or converting formats—are essential for web and mobile applications. Traditionally, these tasks were handled by dedicated servers, but that approach is often costly, hard to scale, and requires constant maintenance.

**Serverless computing** provides a modern, efficient alternative. By combining **Amazon S3 (Simple Storage Service)** for storage and **AWS Lambda** for compute, developers can create a scalable, cost-effective, and fully managed image processing pipeline. With this serverless setup, images uploaded to an S3 bucket can automatically trigger a Lambda function that processes the image in real-time—without provisioning or managing any servers.

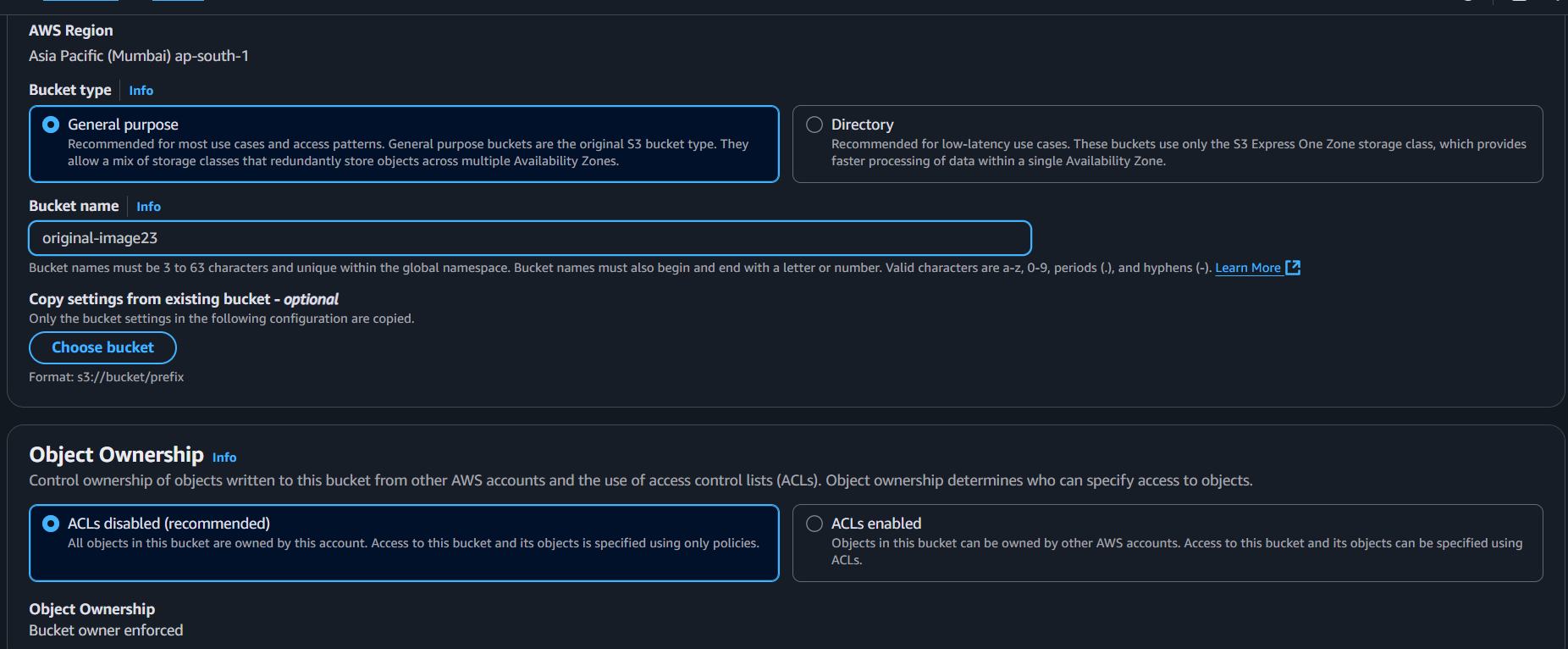
This guide introduces you to the basics of building such a solution, covering how to set up S3 buckets, write Lambda functions, and configure event-driven triggers for fully automated image handling

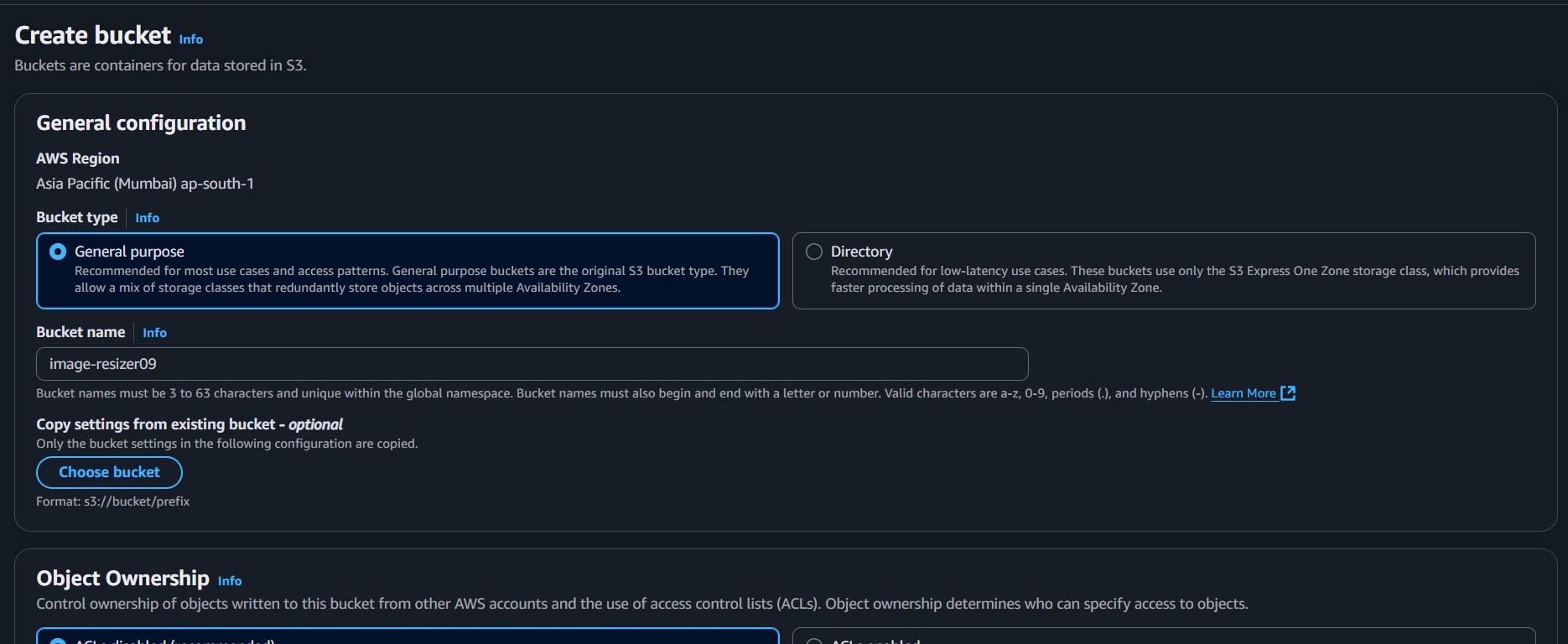
**Architecture overview**

**Folder structure**

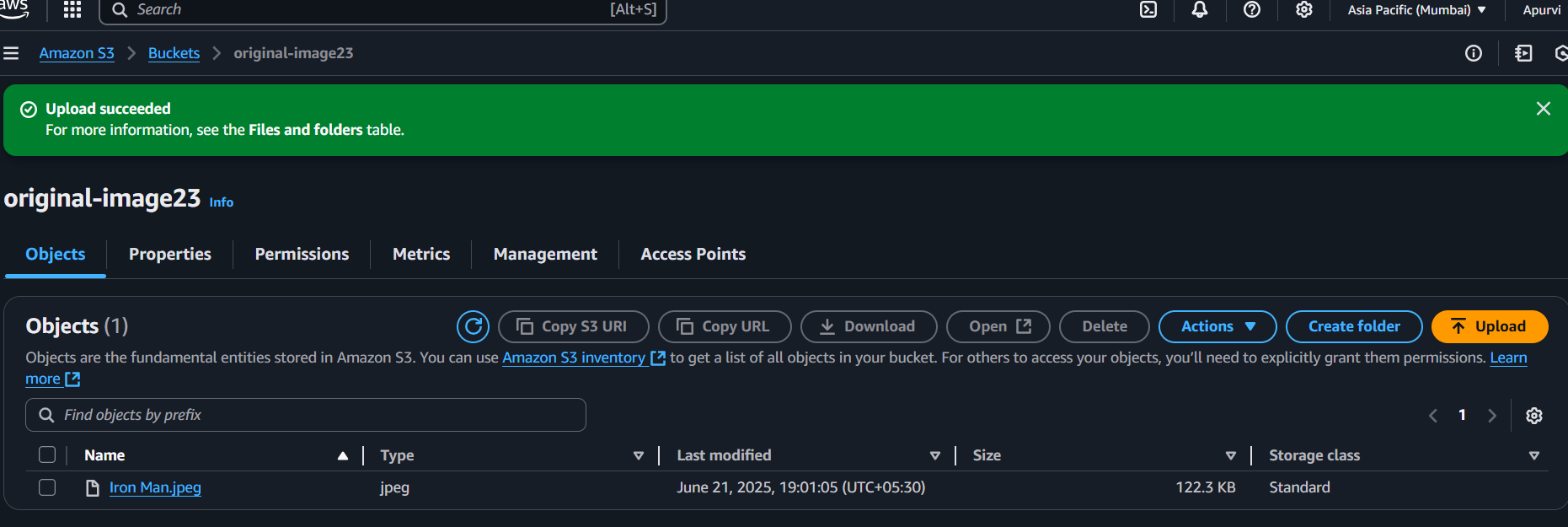
**Step-by-step guide**

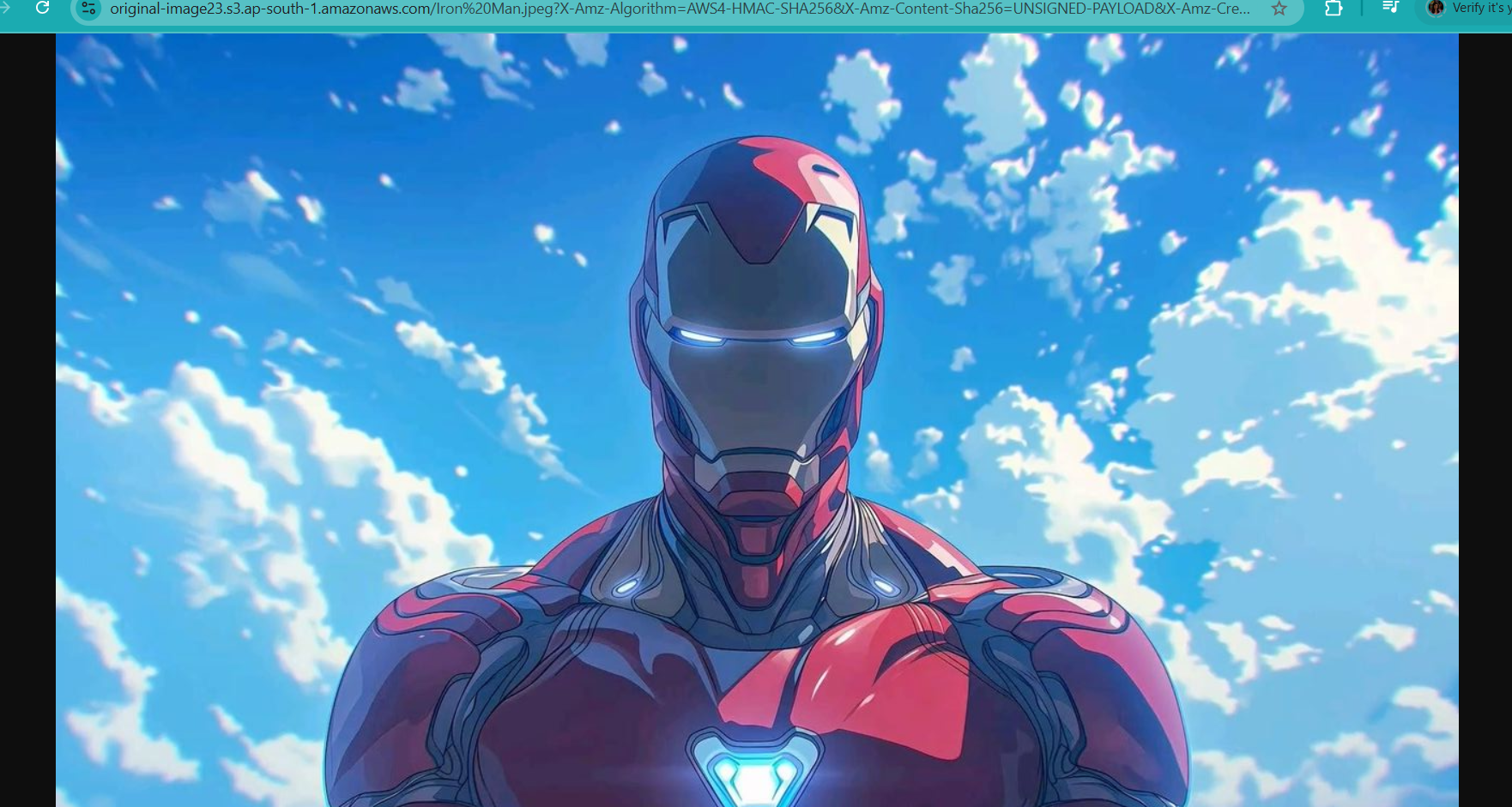
**Setting up s3 buckets**

****1.Create two s3 buckets one for uploading original pictures and second for uploading resized images.



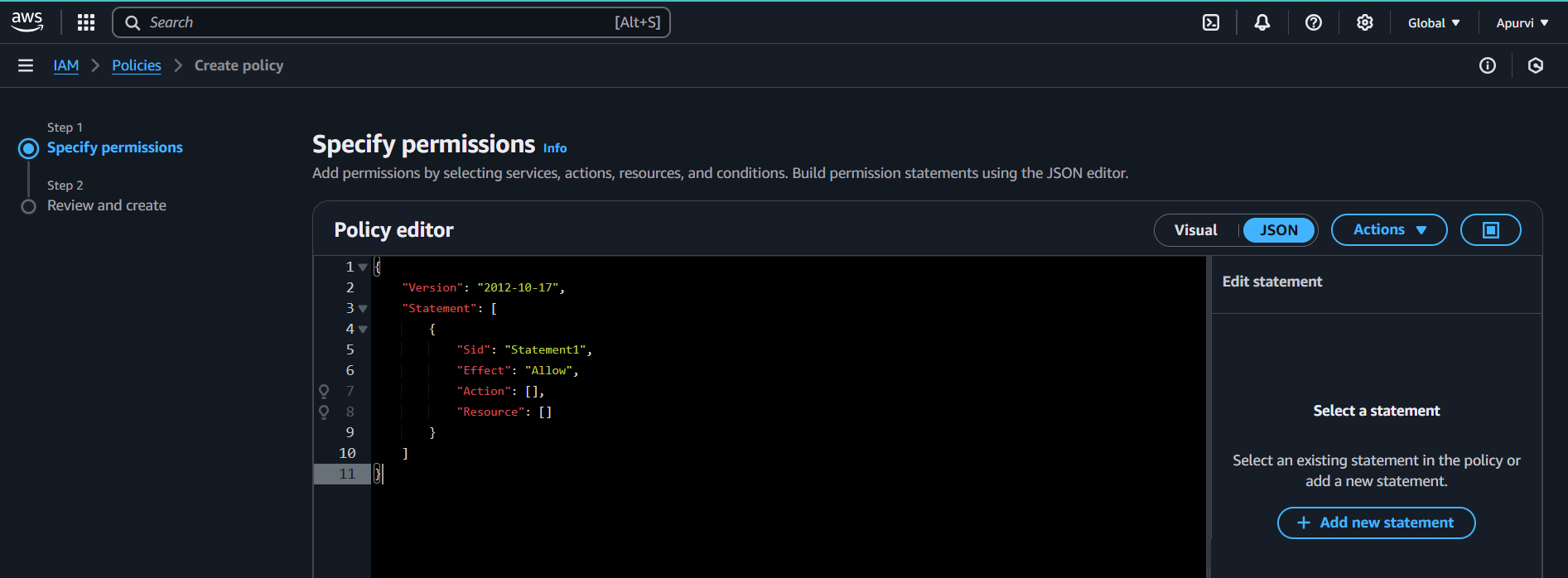
2. leave all the setting in default while creating the two buckets.

3. upload image in a s3 bucket which created to upload original bucket.

4. check that image is opening in the s3 bucket 

5. this is how image looks without resizing with lambda function.

**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",

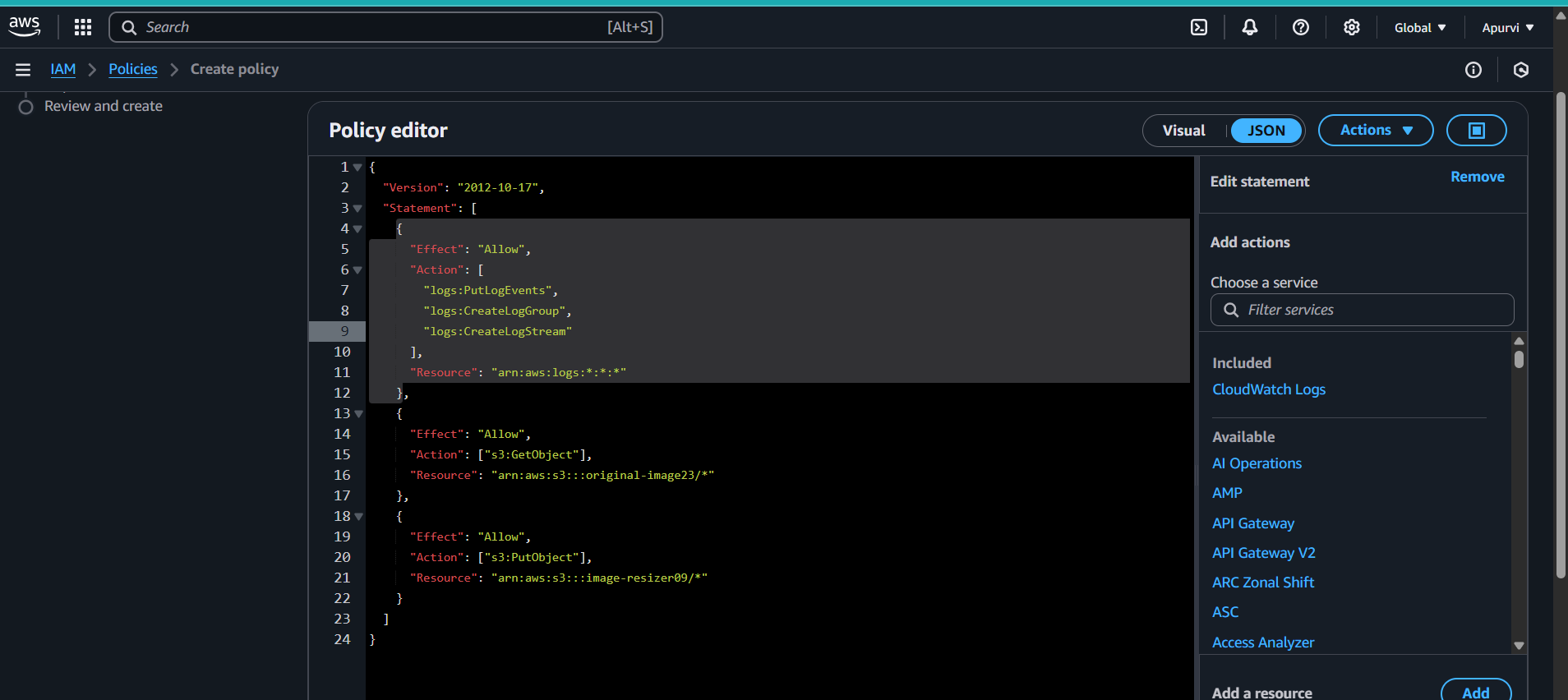
"Action": ["s3:PutObject"],

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

}

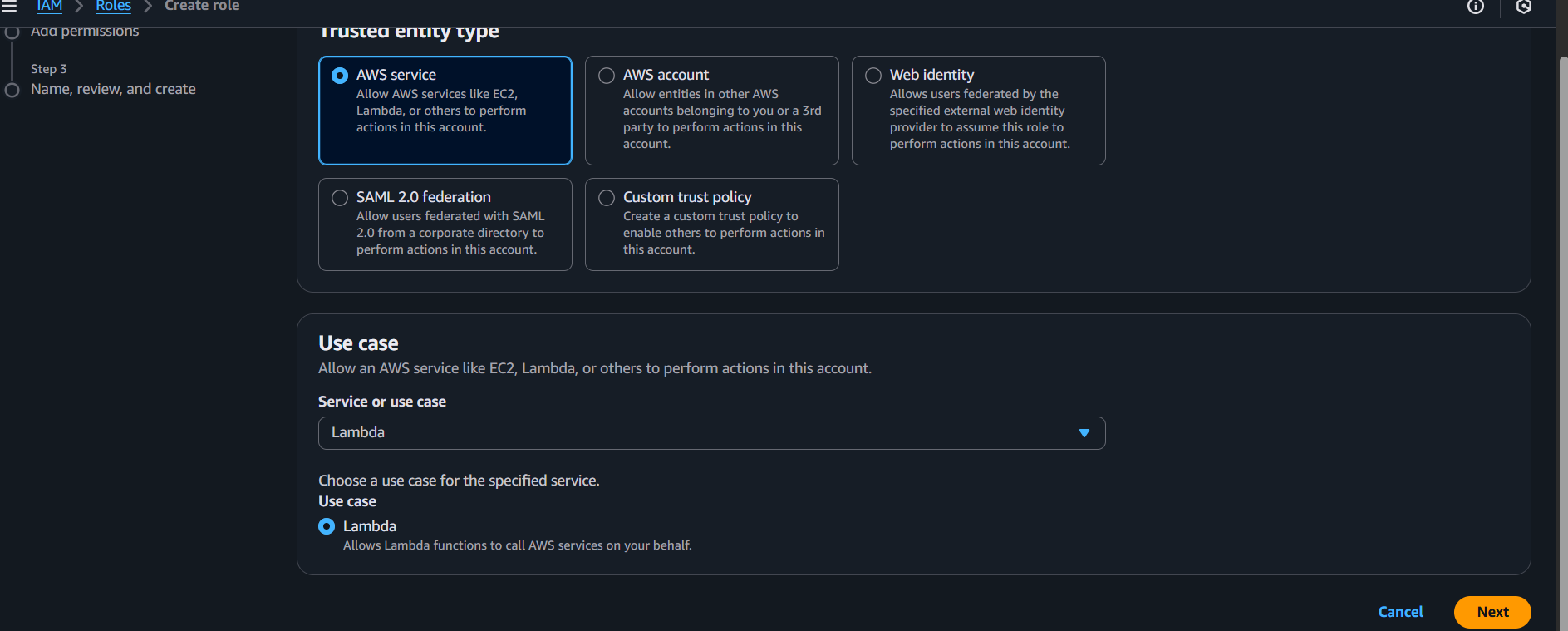
]

}

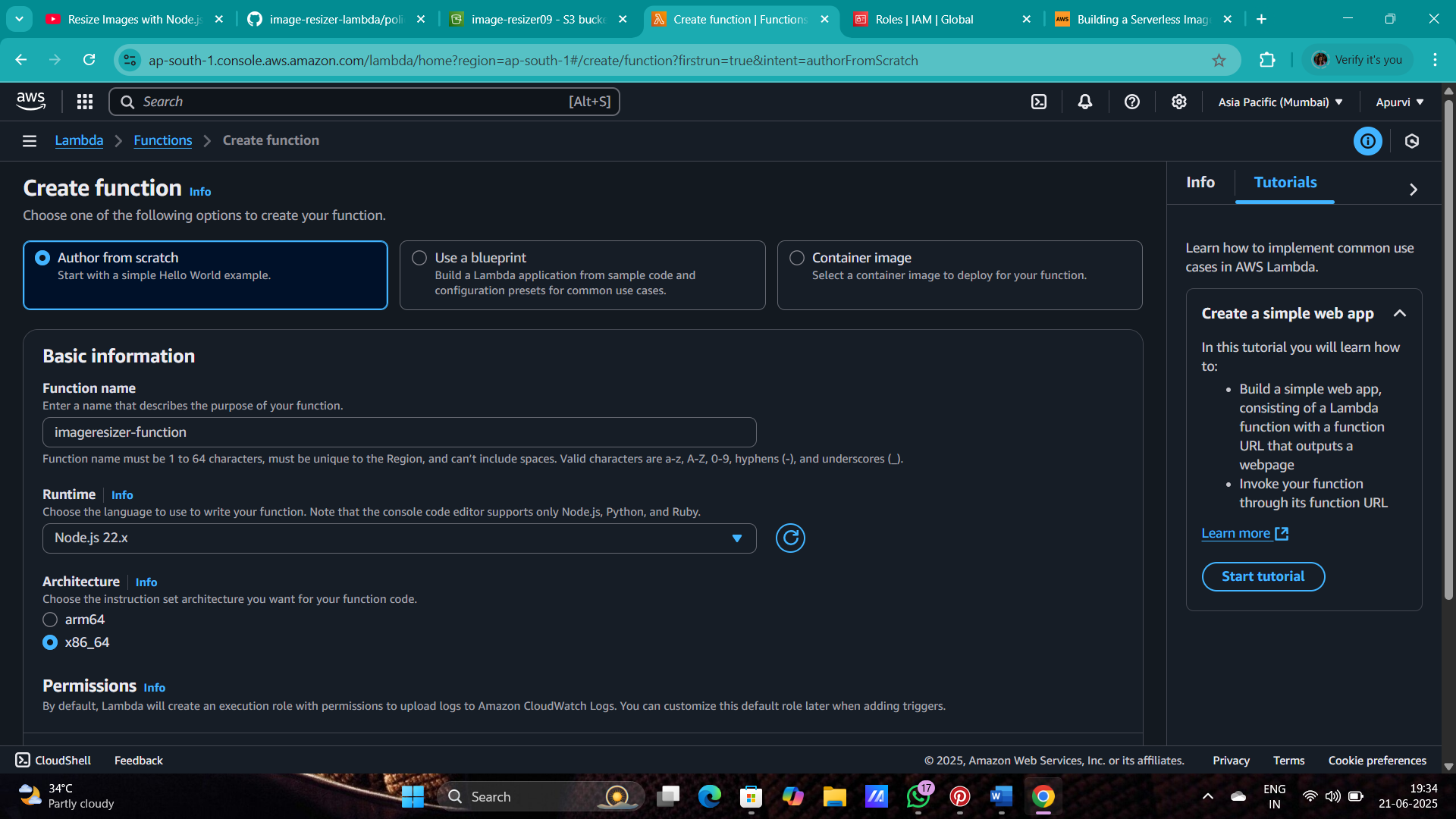
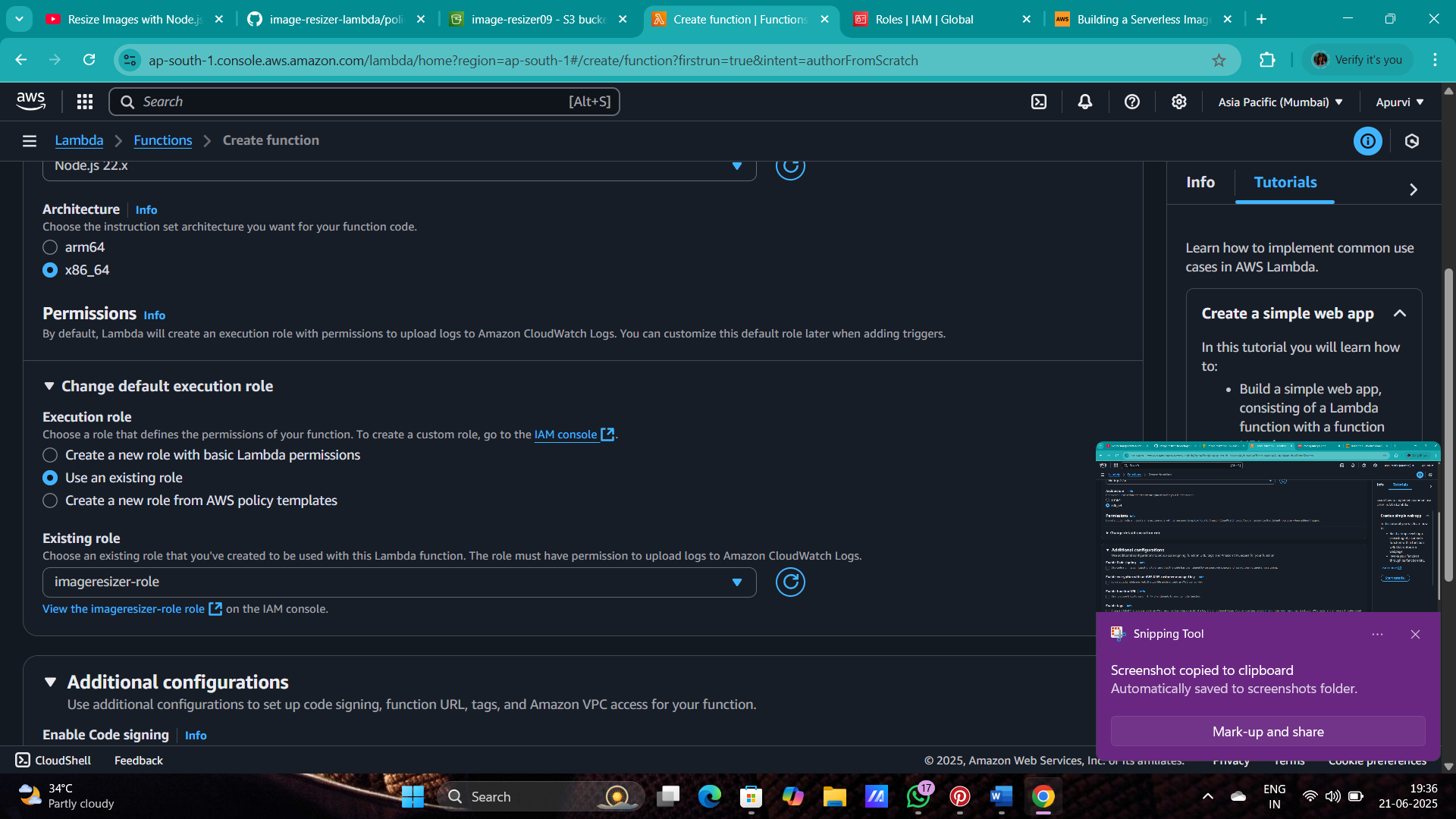


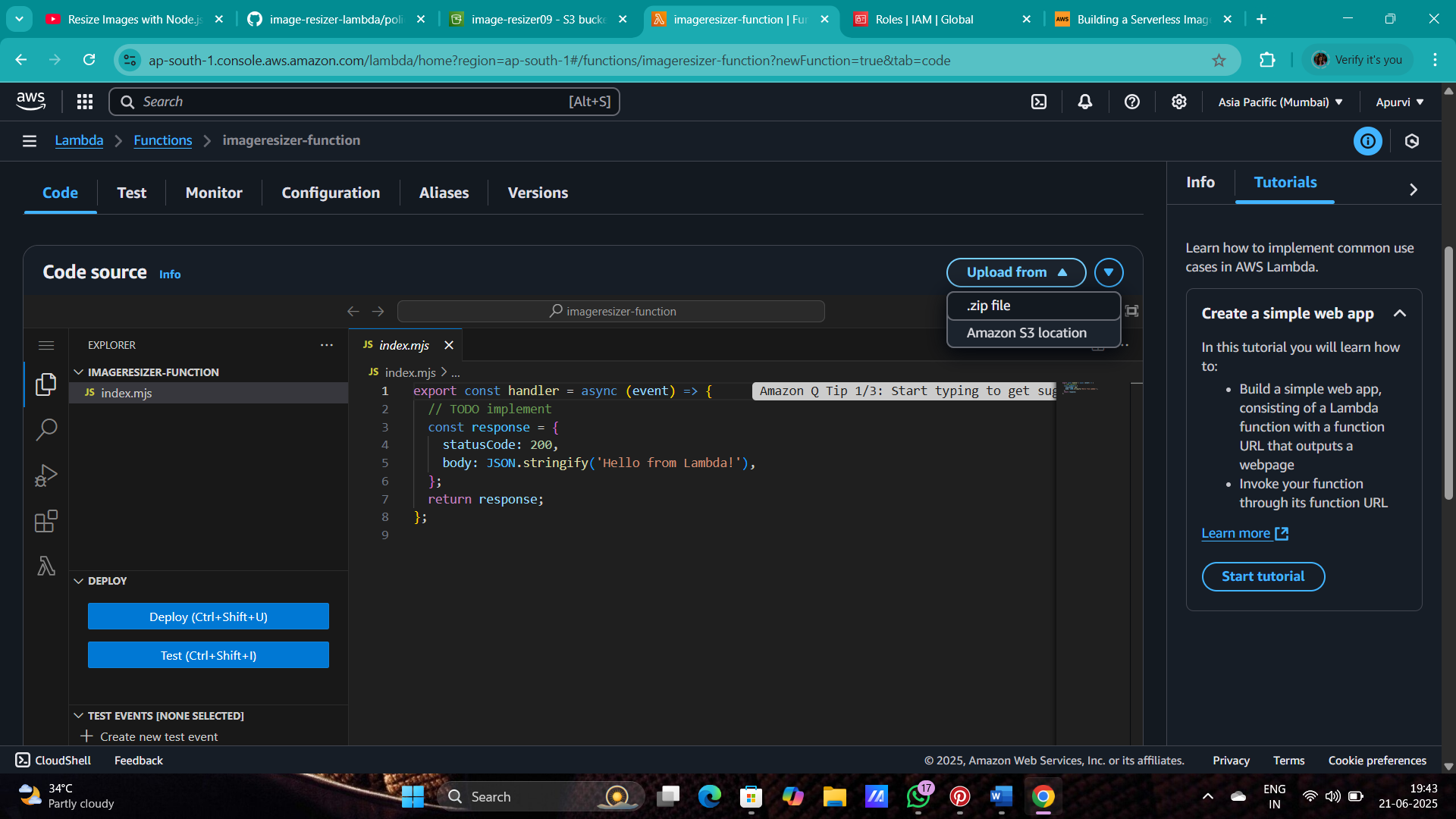
1. Replace the bucket name with your bucket where you have uploaded the image and DEST\_BUCKET with the resizer bucket name.
2. And create the policy.

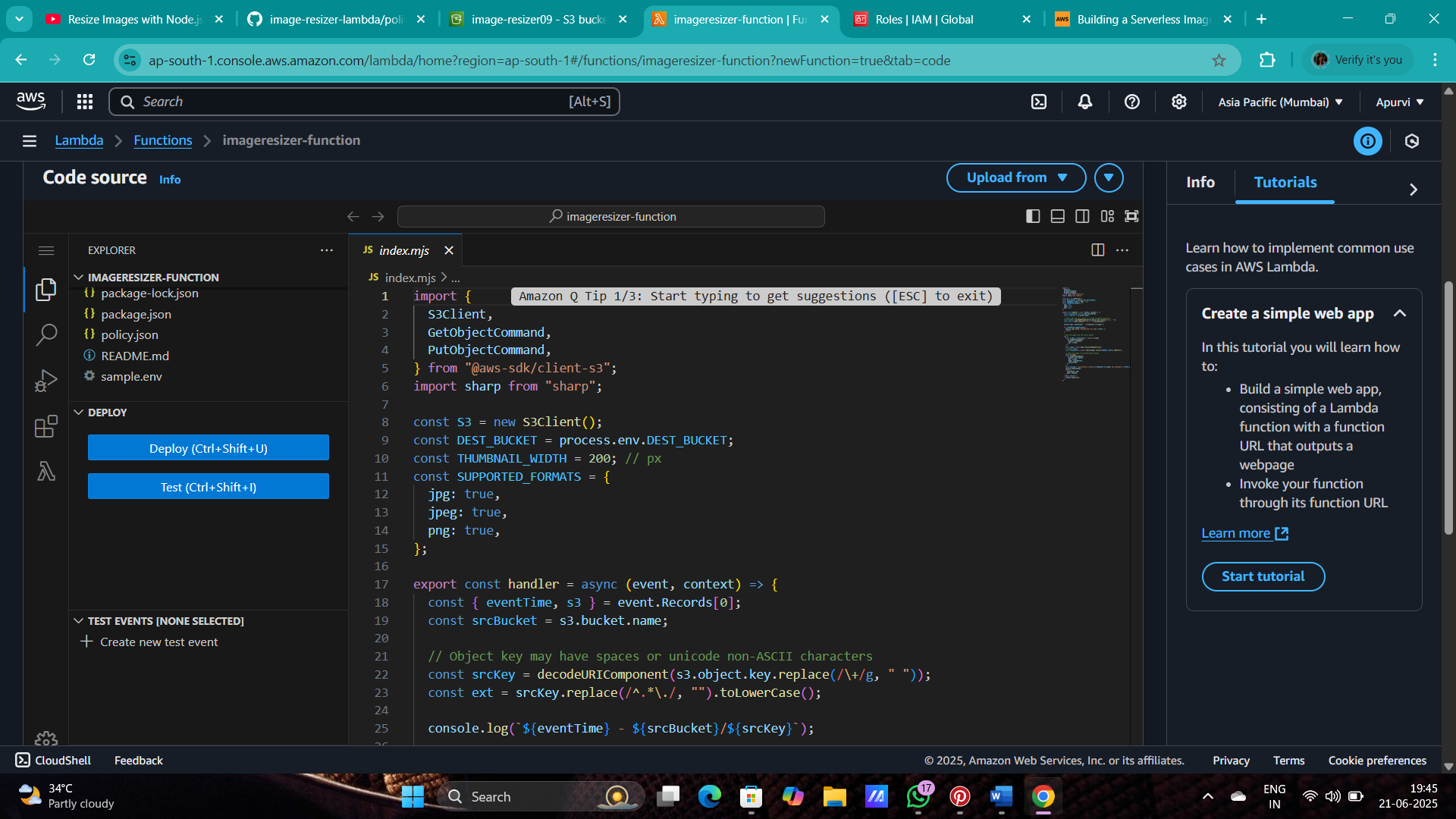
**Setting up of IAM role**

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

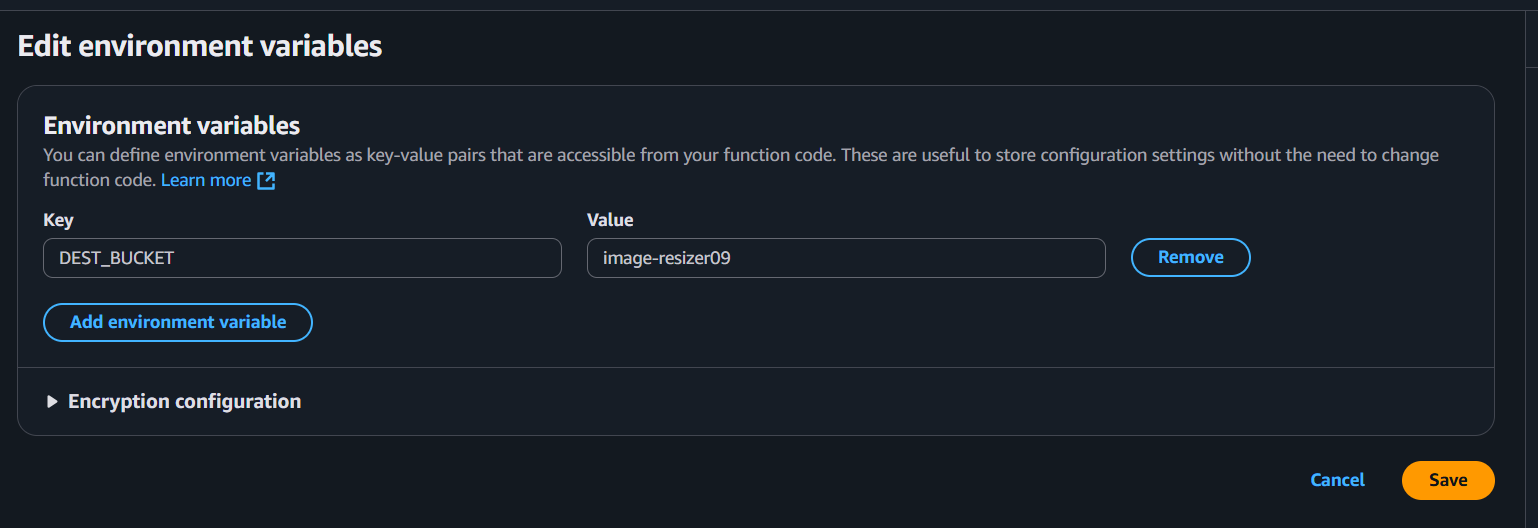
**Making of lambda function**

1. Go to lambda and click on create function.
2. Now add the role which created.
3. And create the lambda function.
4. 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.

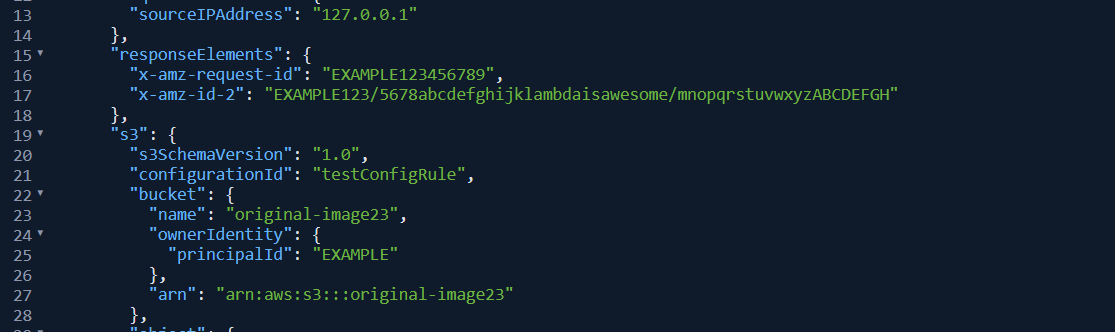


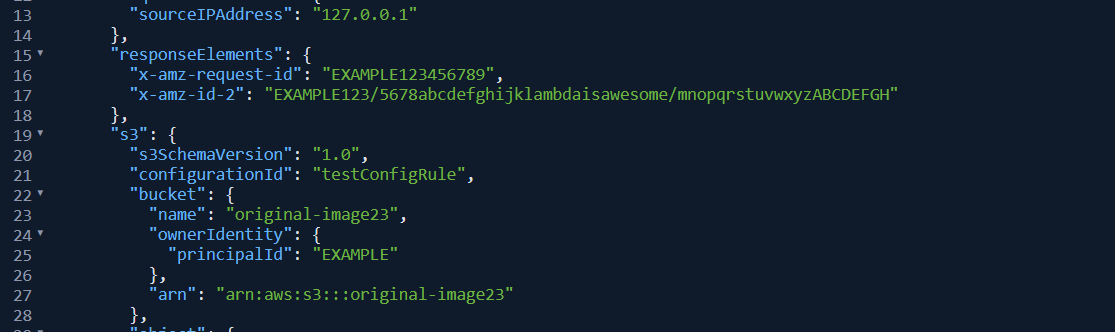
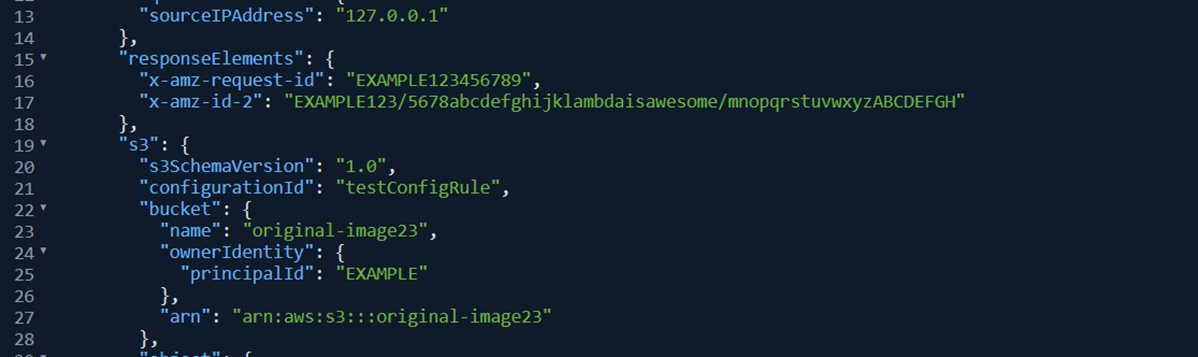


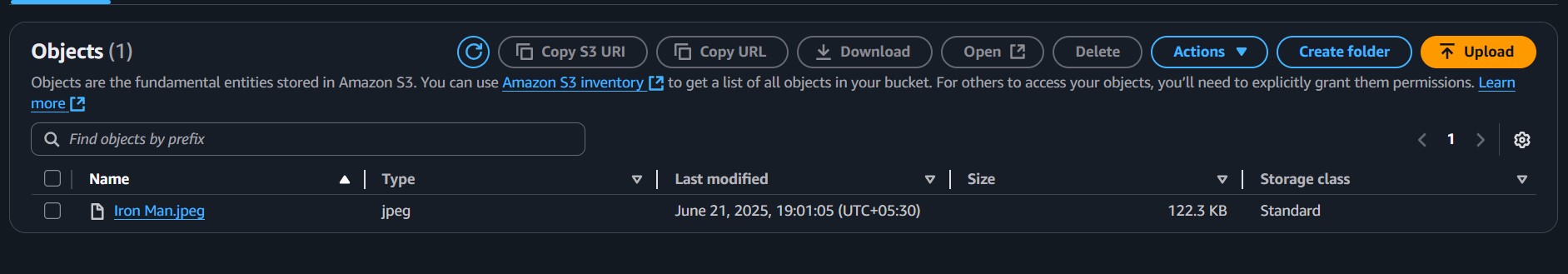
1. Your code will look like this.
2. Now go in configuration and click on environment variables and edit it and click on add environment variables.



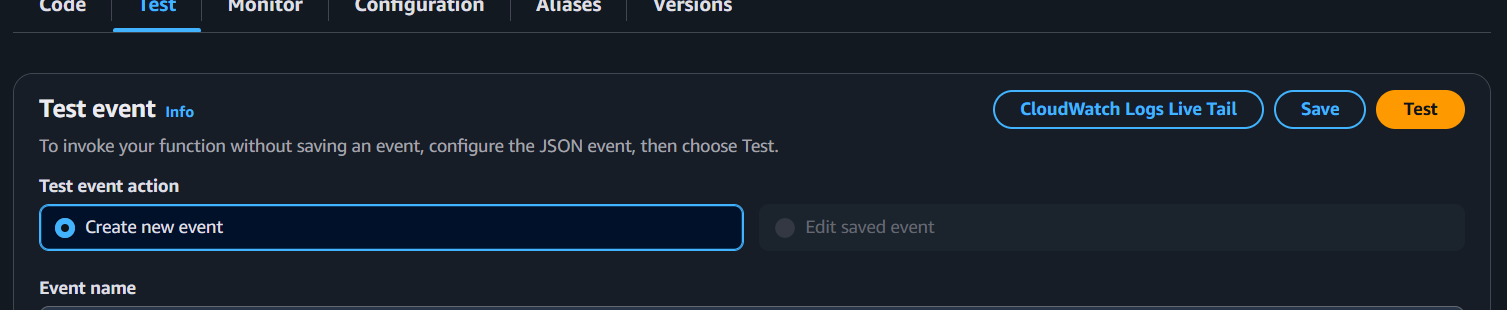
1. And then goes to test event in a code source.
2. now go in the event JSON and the edit the bucket with your bucket where you have uploaded

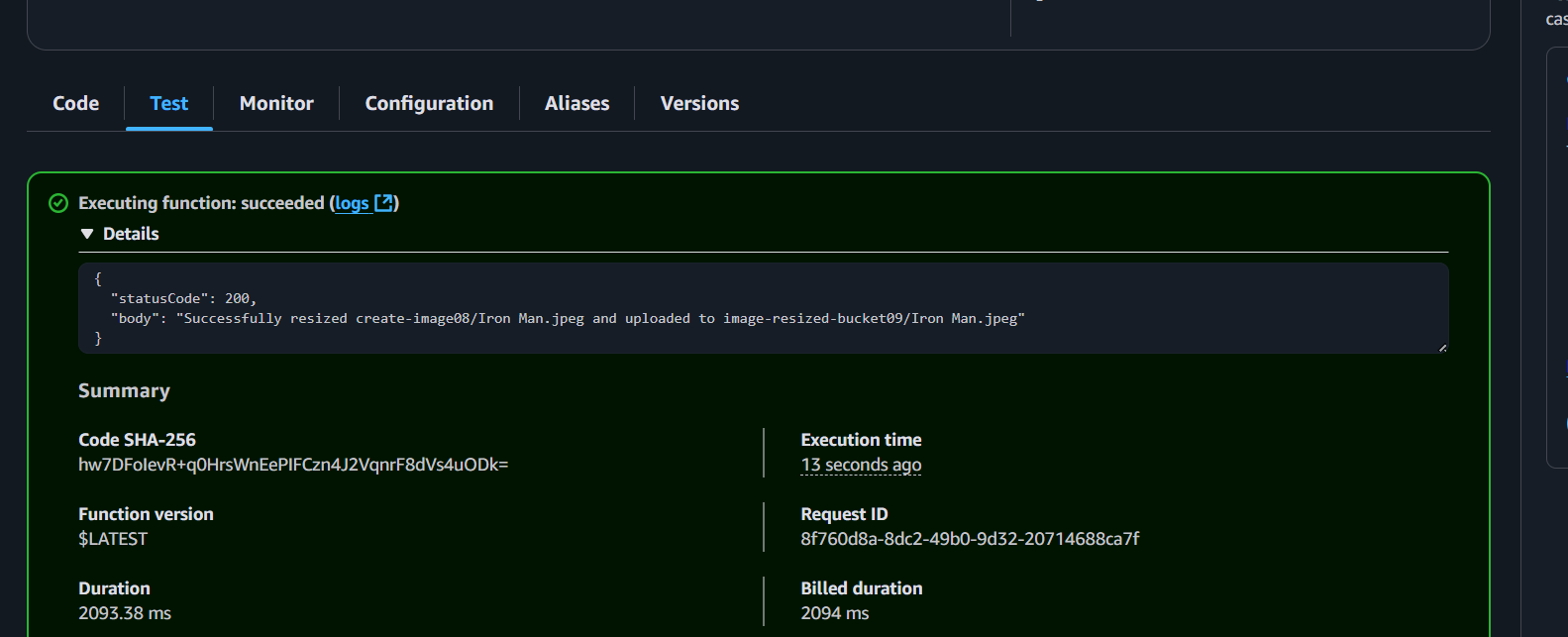


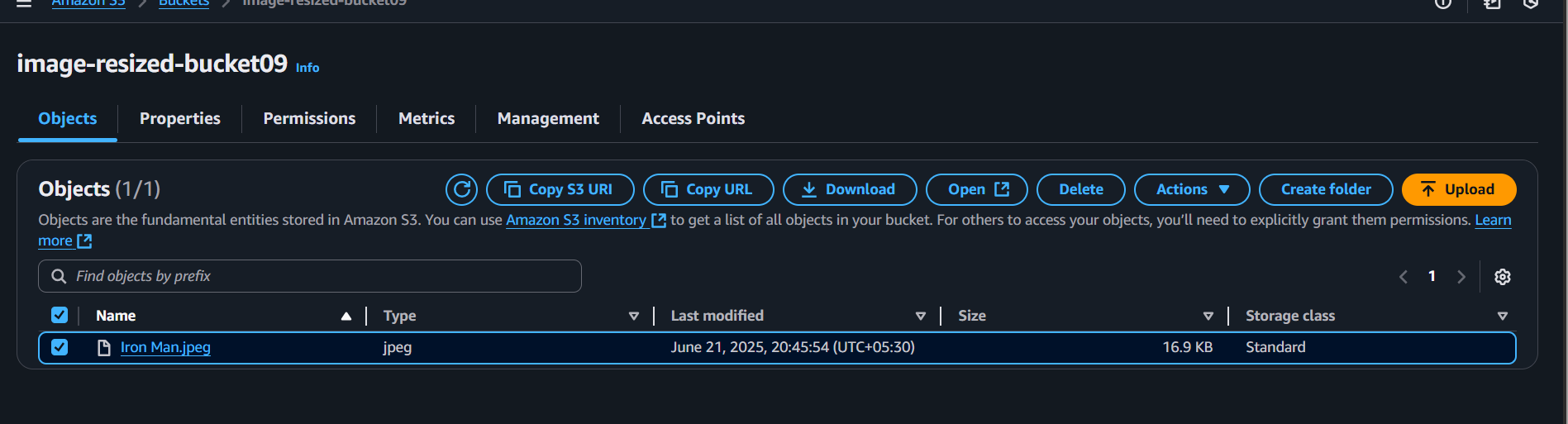
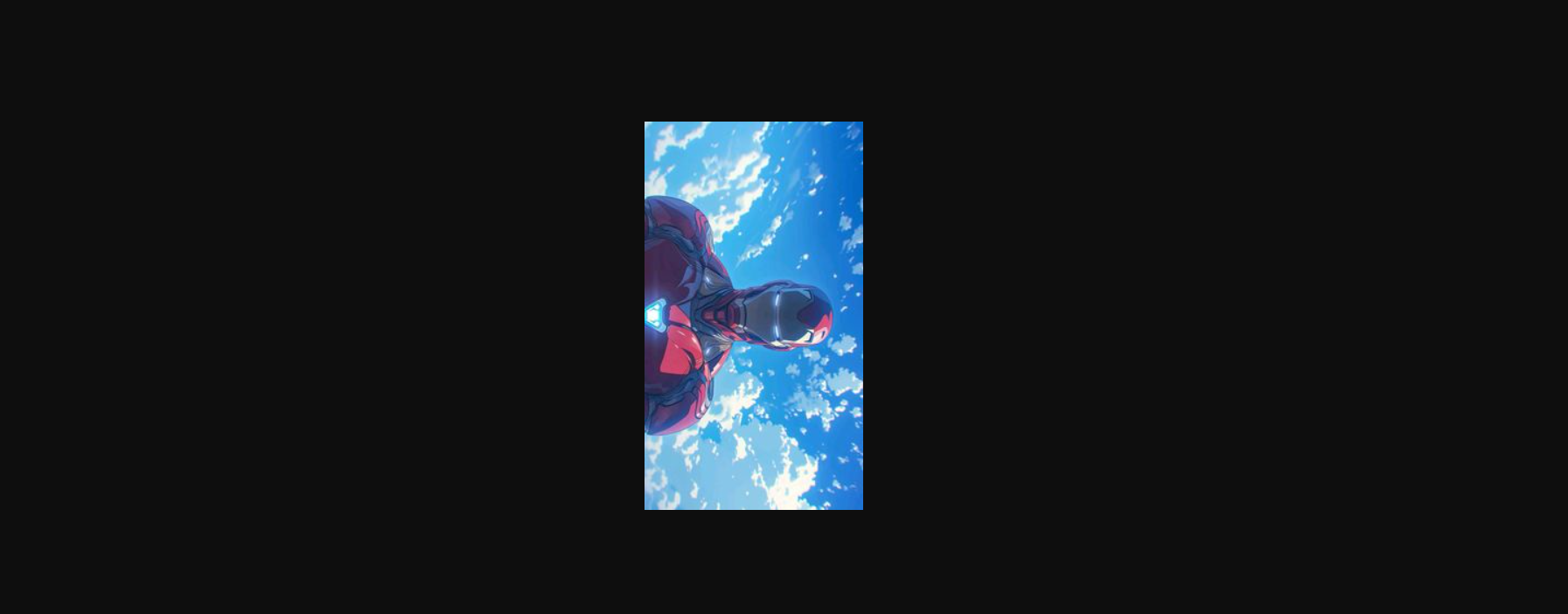


1. now change the key name with the image name which you have uploaded in s3 bucket.

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 .

13. resized image will look like this.