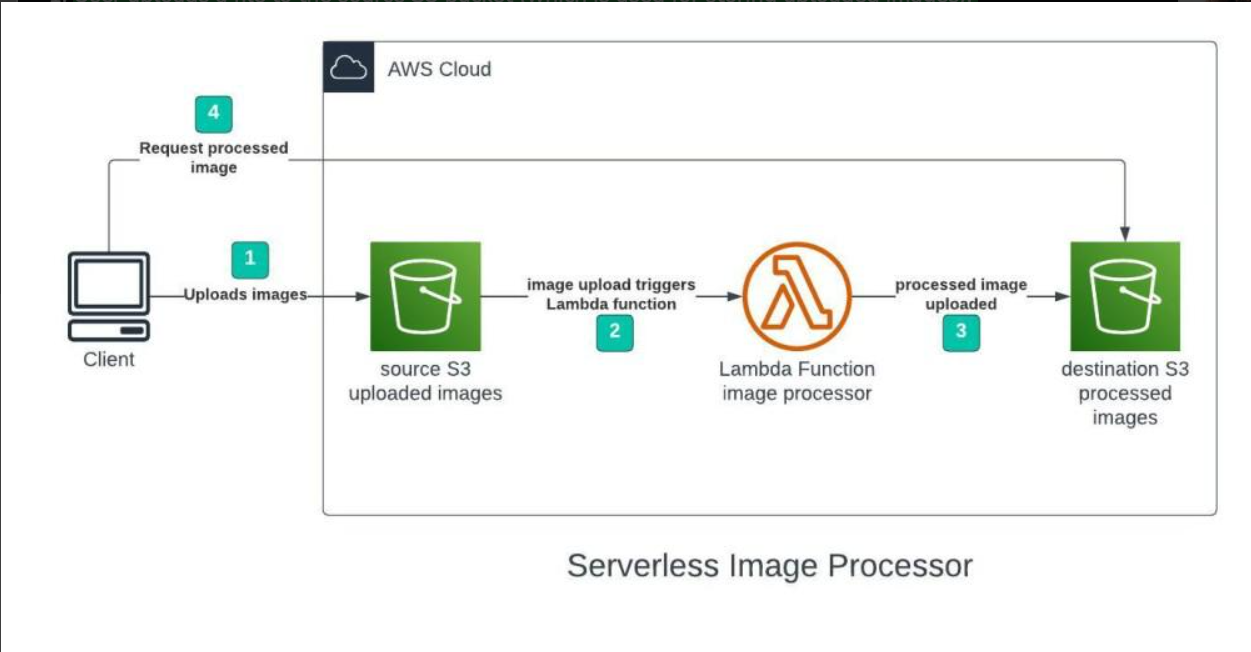
**Serverless Image Processing**

The Serverless Image Handler solution helps you embed images on your websites and mobile applications to drive user engagement. It uses the [sharp](https://sharp.pixelplumbing.com/en/stable/) Node.js library to provide high-speed image processing without sacrificing image quality. To minimize your costs of image optimization, manipulation, and processing, this solution automates version control and provides flexible storage and compute options for file reprocessing

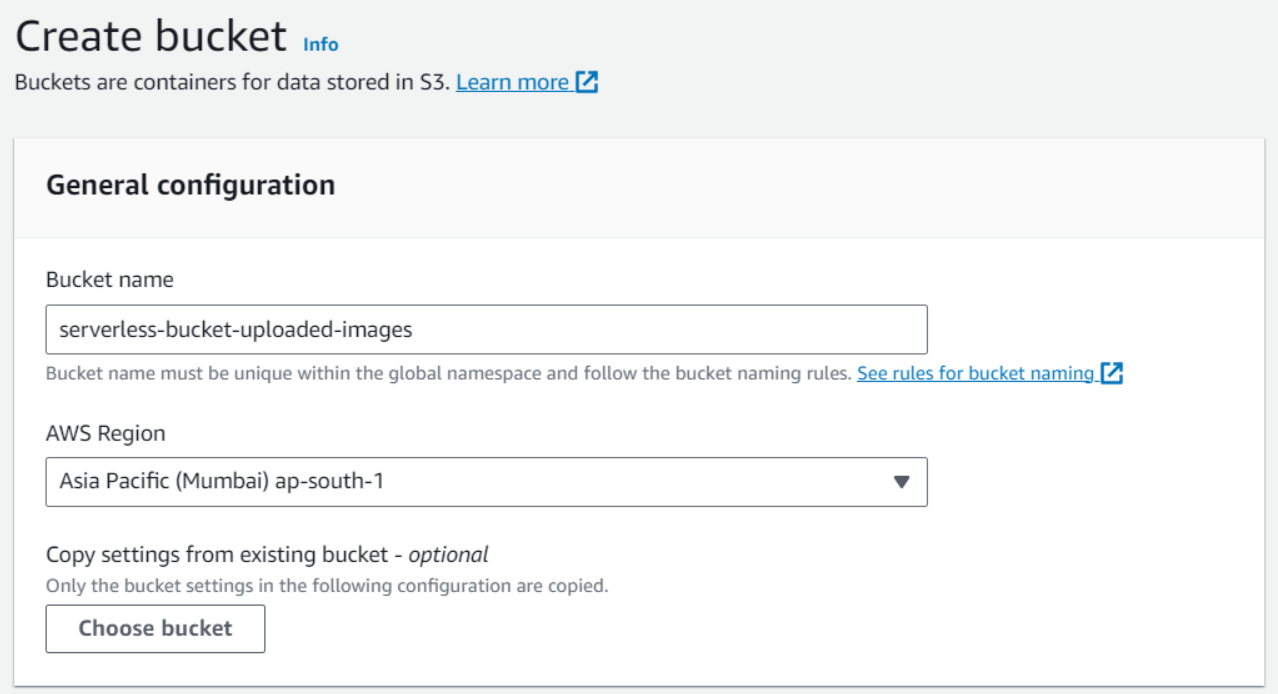


## Step 1 – Creating S3 buckets

### We will use two S3 buckets:

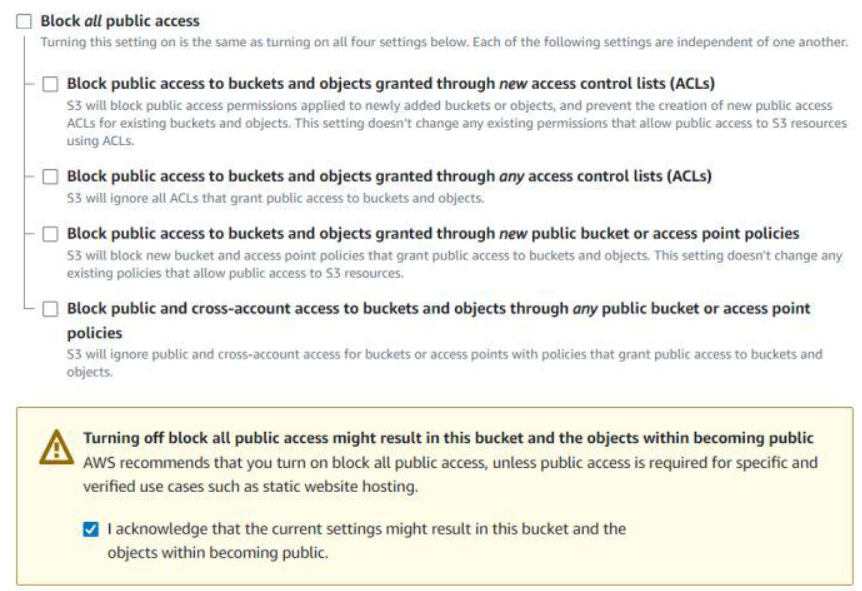
1. source Bucket: For storing uploaded images.

2. destination Bucket: For storing processed images.



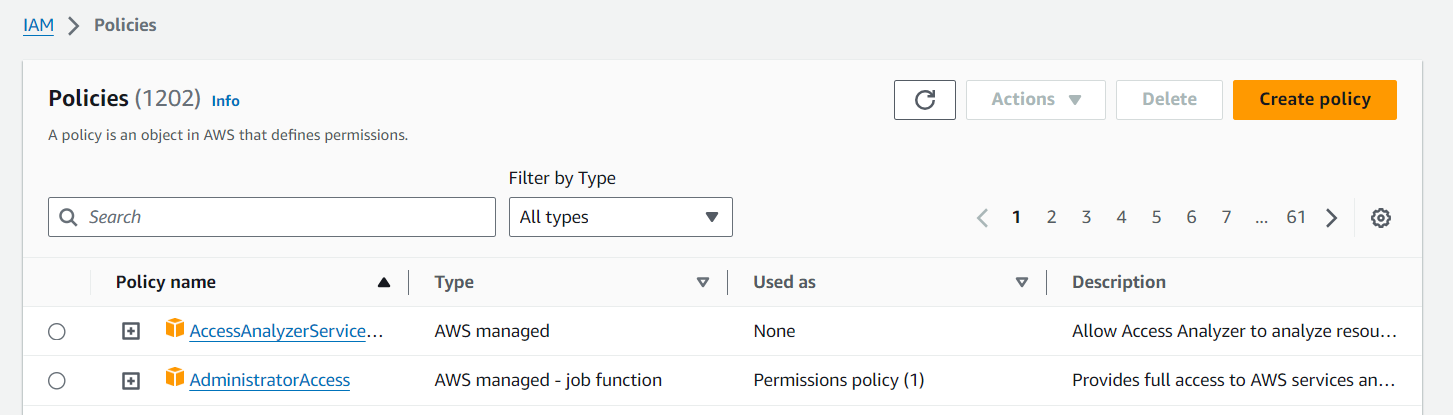
**Step 2 – Configuring S3 bucket policy**

**In ‘Block Public Access settings for this bucket’ section disable “block all public access”.**

****

**Step 3 – Creating police in Iam**

**Go to AWS I am console. Navigate to policies section. Click Create policies in (JSON) and name it “ImageBucketpolicy”. Leave all other settings as default. Create the policy.**

****

**Policy:**

"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/\*"

}

]

}

**\*BUCKET\_NAME = SOURSE BUCKET NAME**

**\*DEST\_BUCKET = DESTINATION BUCKET NAME**

**Step 4 –creating role in I am**

**Following Steps are Follows**

\*Go to aws I am console

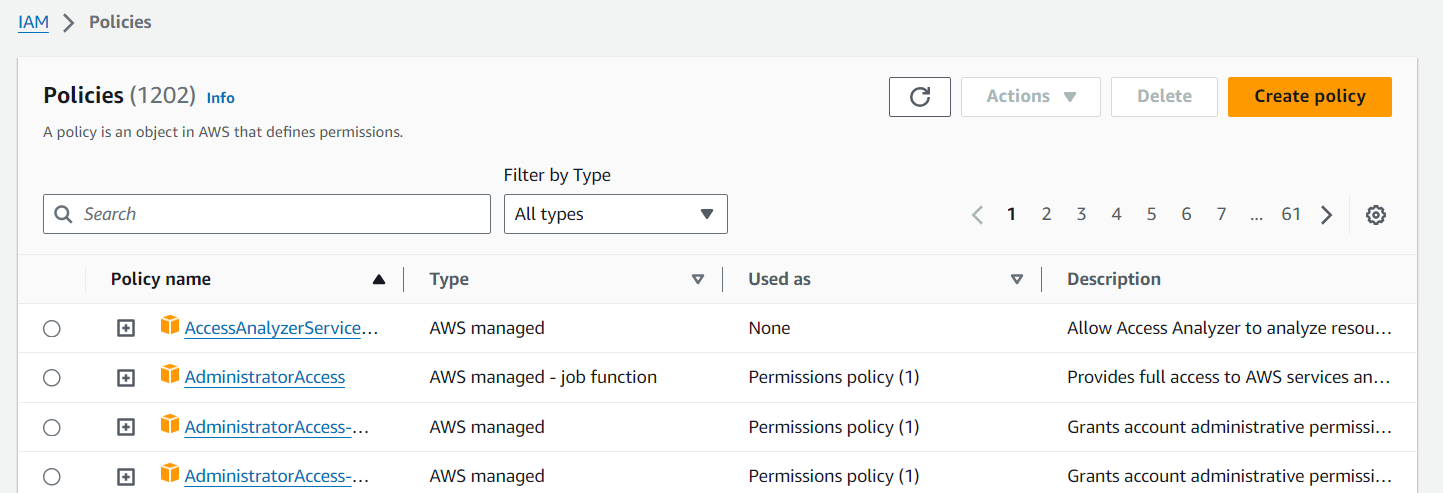
\*Create role

\*name imageresizerlambdarole

\*Use case – Lambda

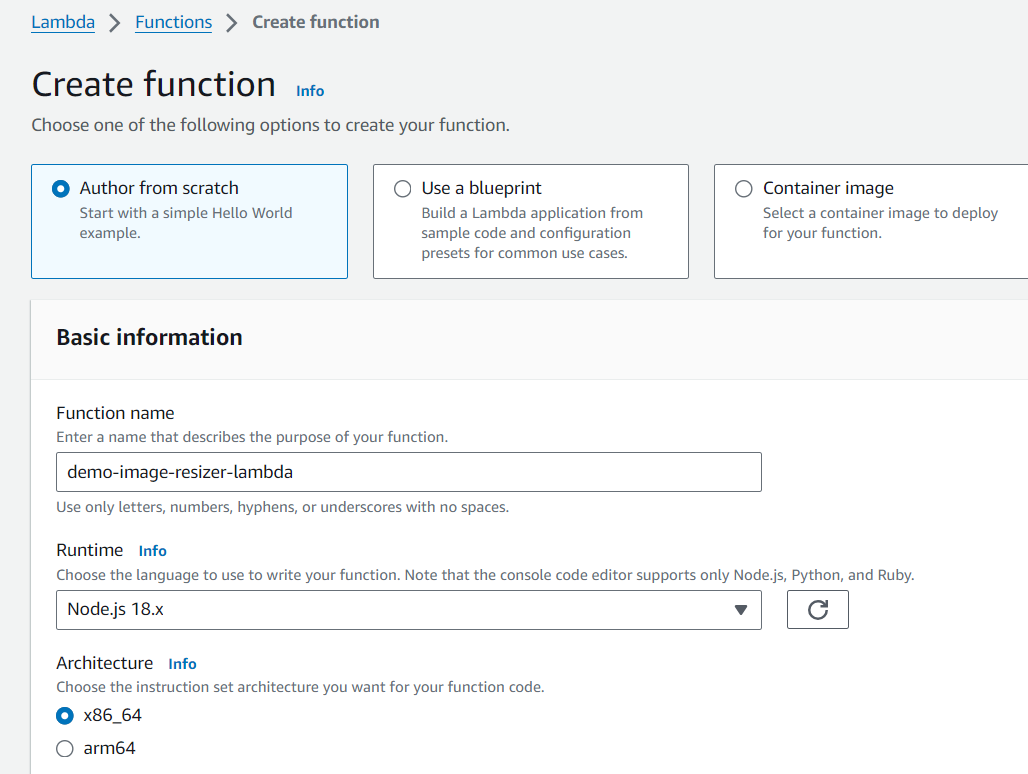
\*Select-ImageBucketPolicy

\*Then create role

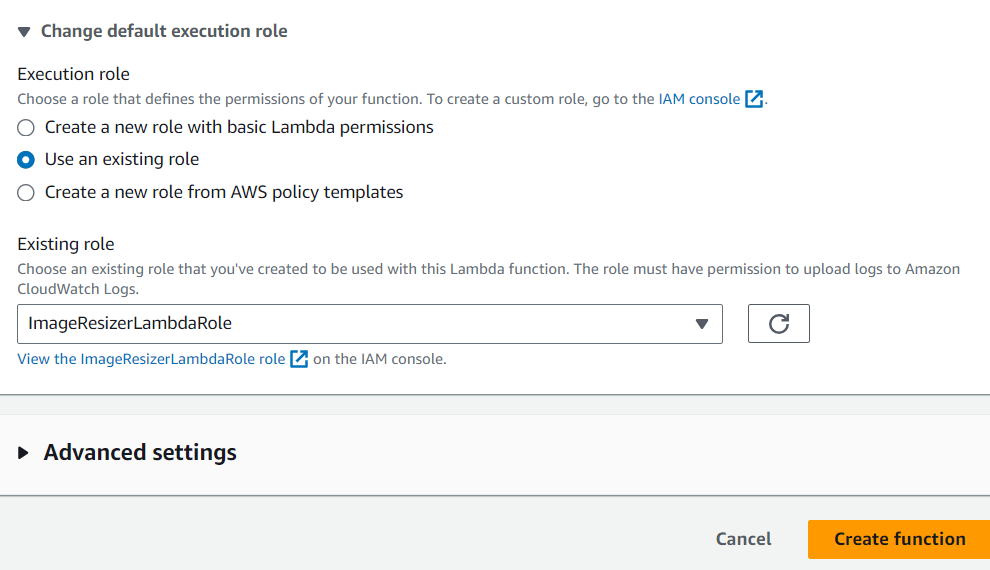
****

**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 16.x” and architecture as “x86\_64”. Leave all other settings as default. Create the function.

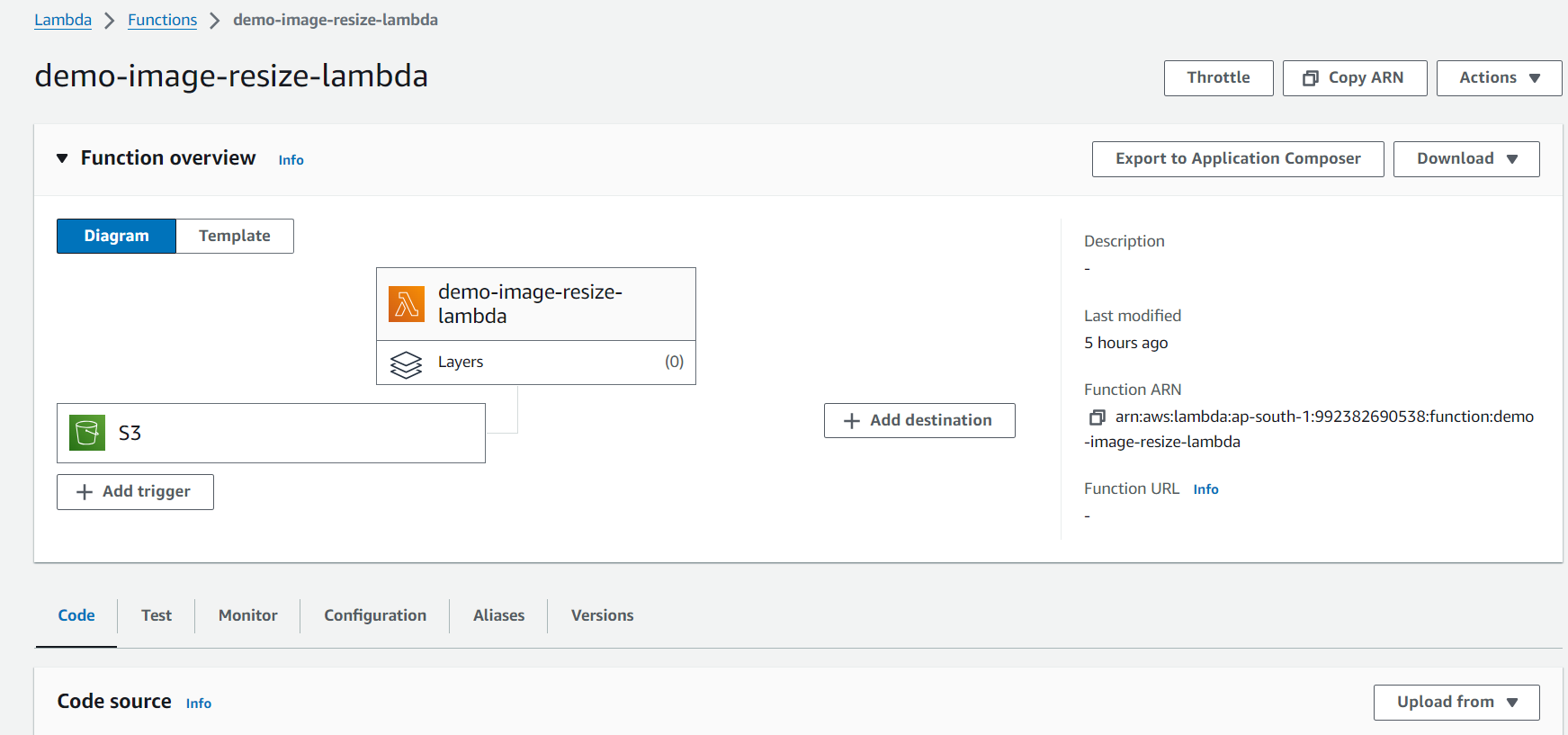


**\*Change default execution role**



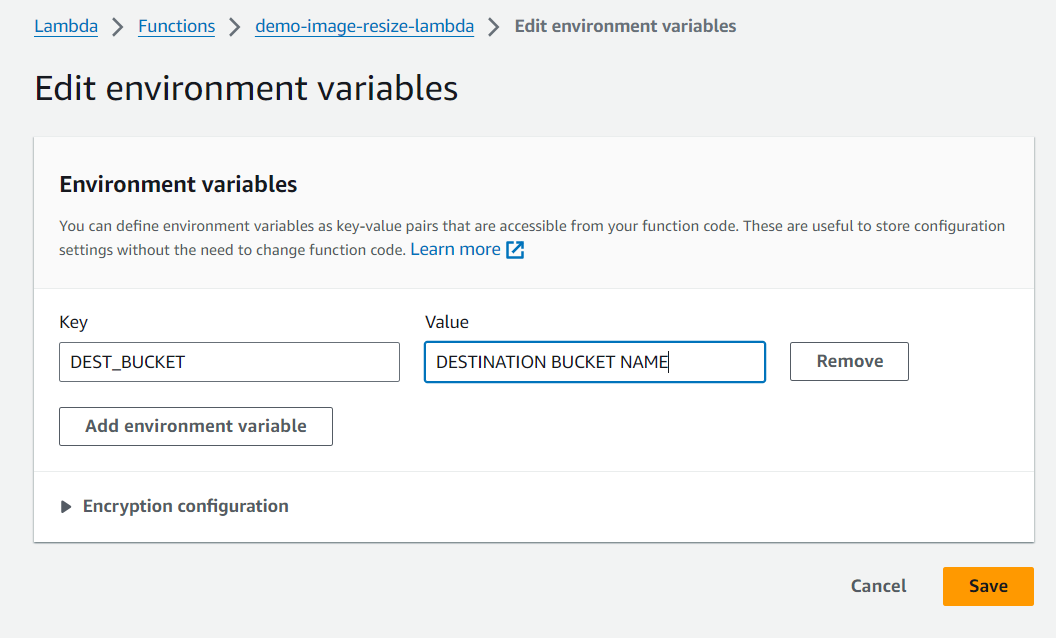
**\*Create function**

**Step 6 -upload zip file in Lambda function**

****

\*Zip file link-https://github.com/OneLightWebDev/image-resizer-lambda

**Step 7** - Edit environment variables



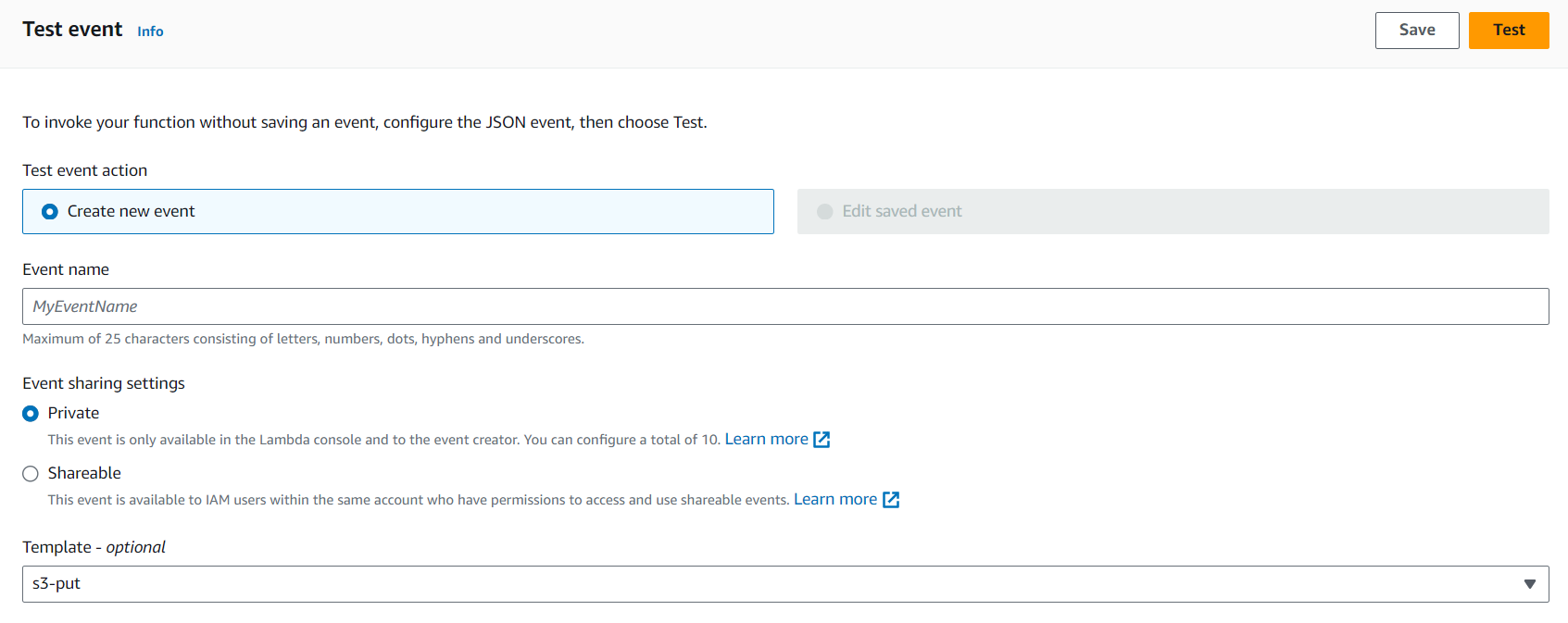
**Step 8 – Test Lambda Function**

\*Go to AWS Lambda console. Navigate to Functions section.

\*open function then will be created

\*open test console

\*template=s3-put

**EVENT JSON**

In event json we can change only 3 value

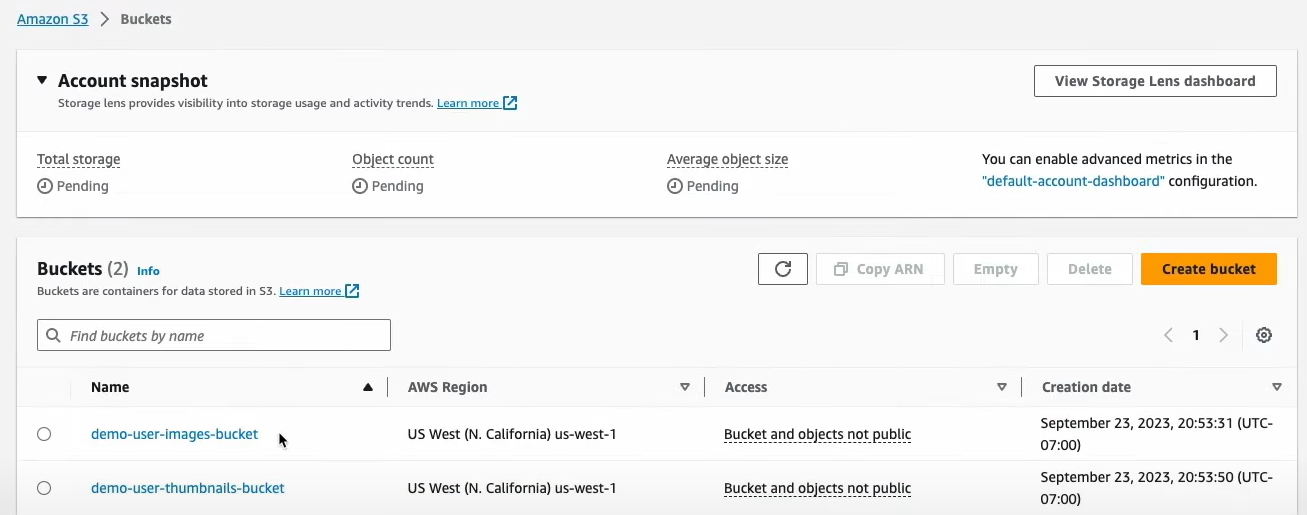
\*name, arn, key

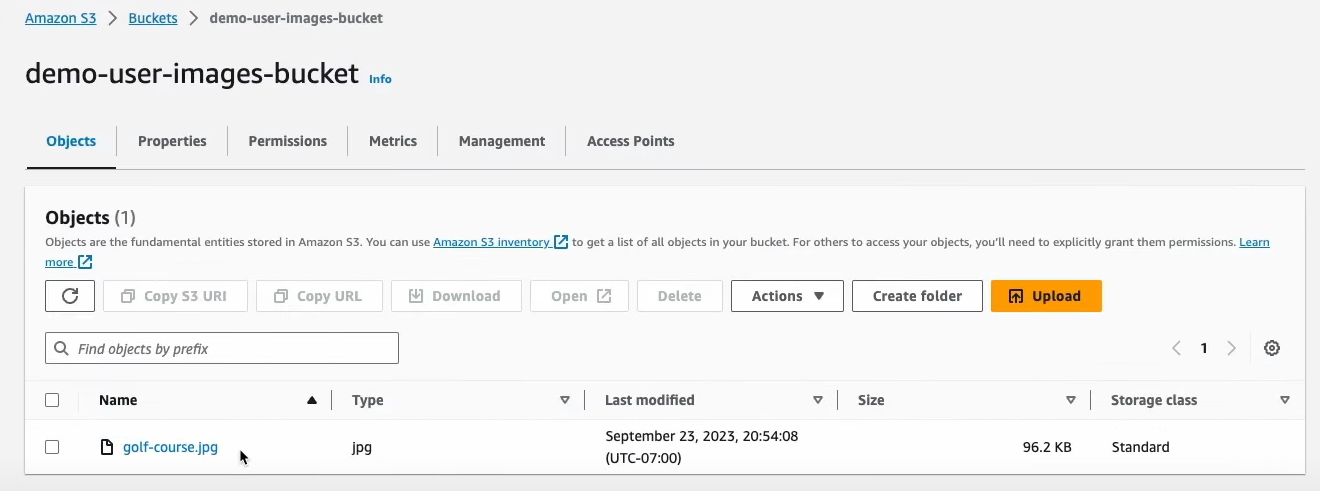


"name":"demo-user-images-bucket",->

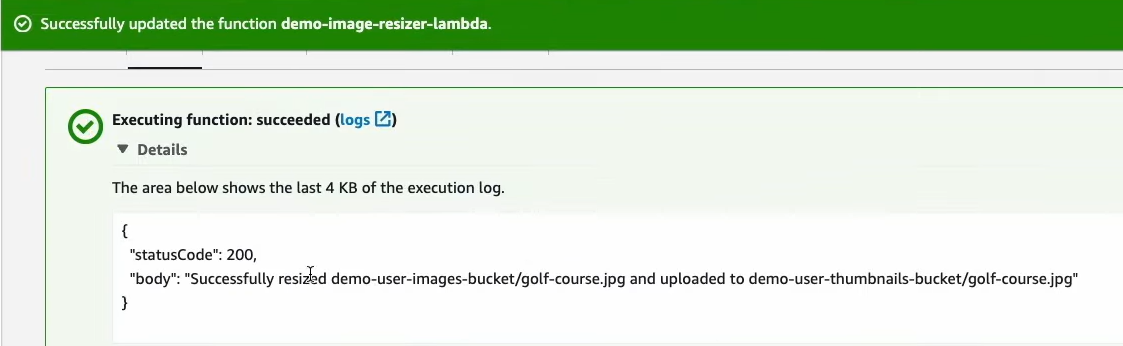
"arn": "arn:aws:s3::: demo-user-images-bucket "->

"key": "golf-course.jpg",->





**Now we can test**



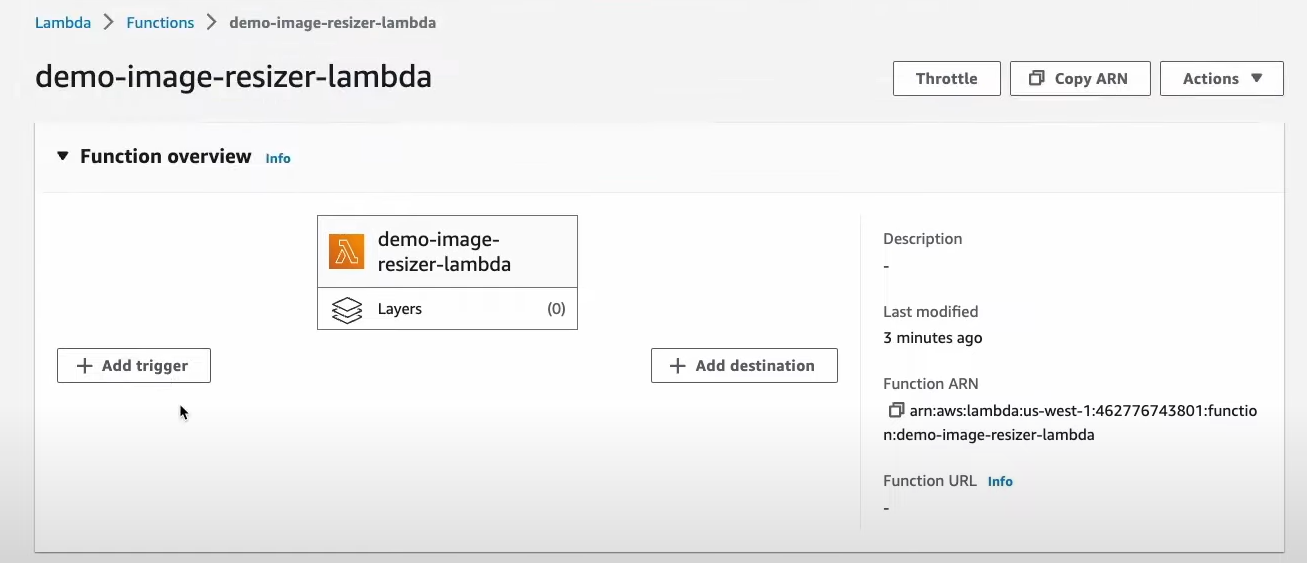
**Step 5 – Creating S3 trigger**

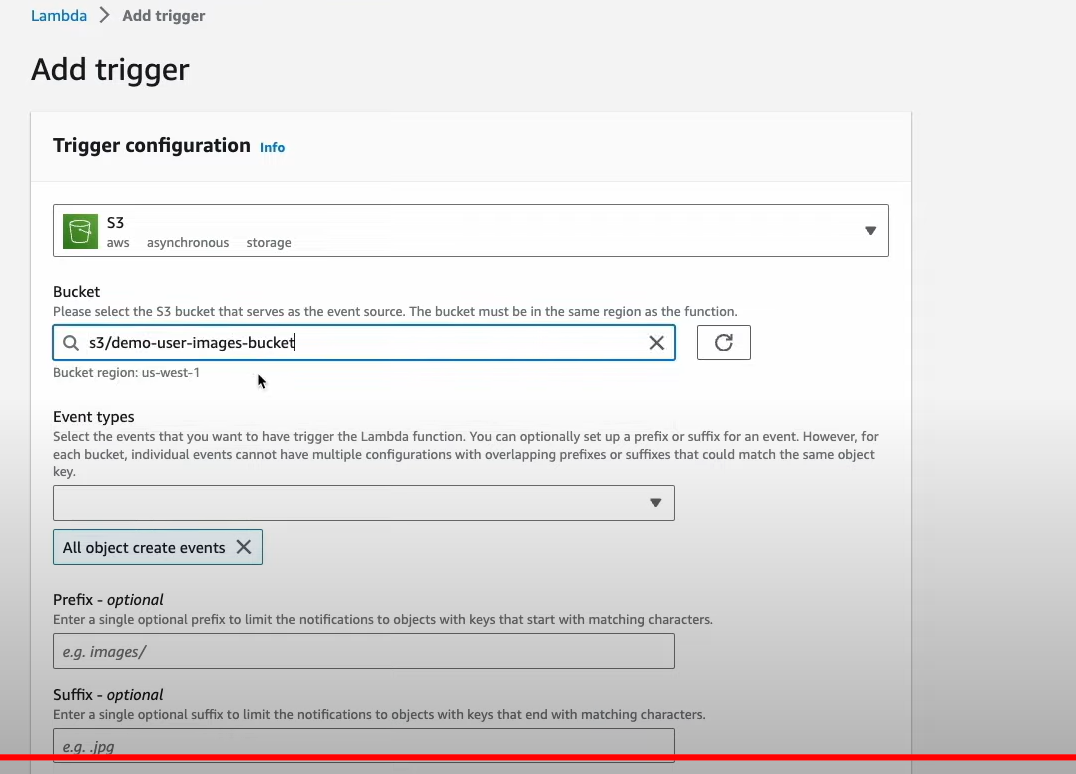
\*Add trigger

\*Select s3

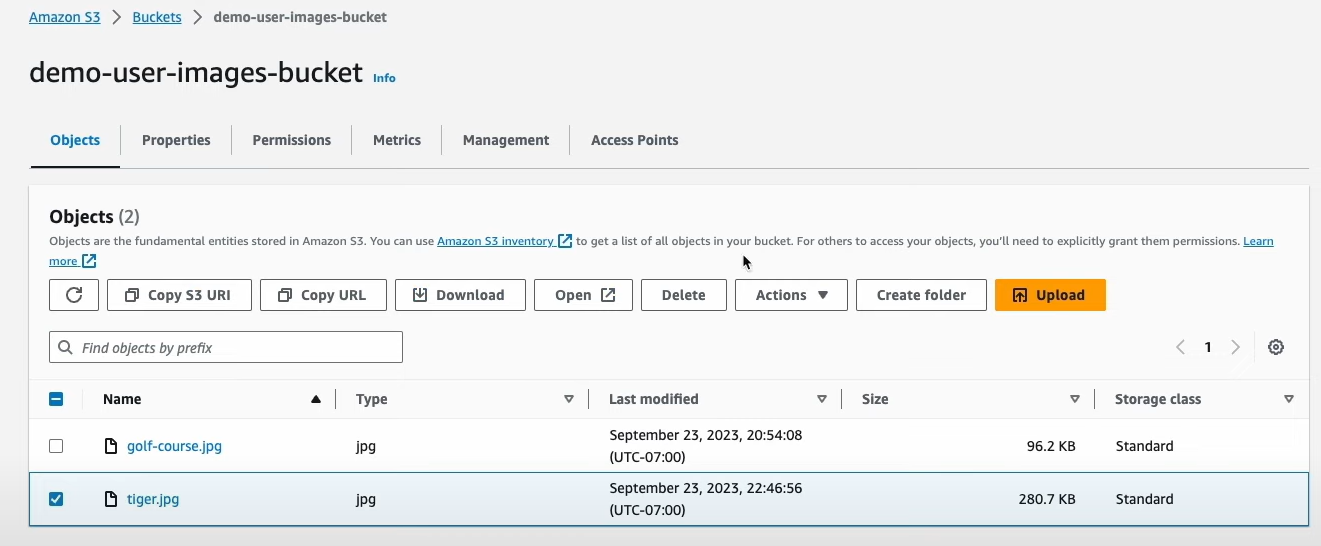
\*choose source Bucket name

\*Now Add





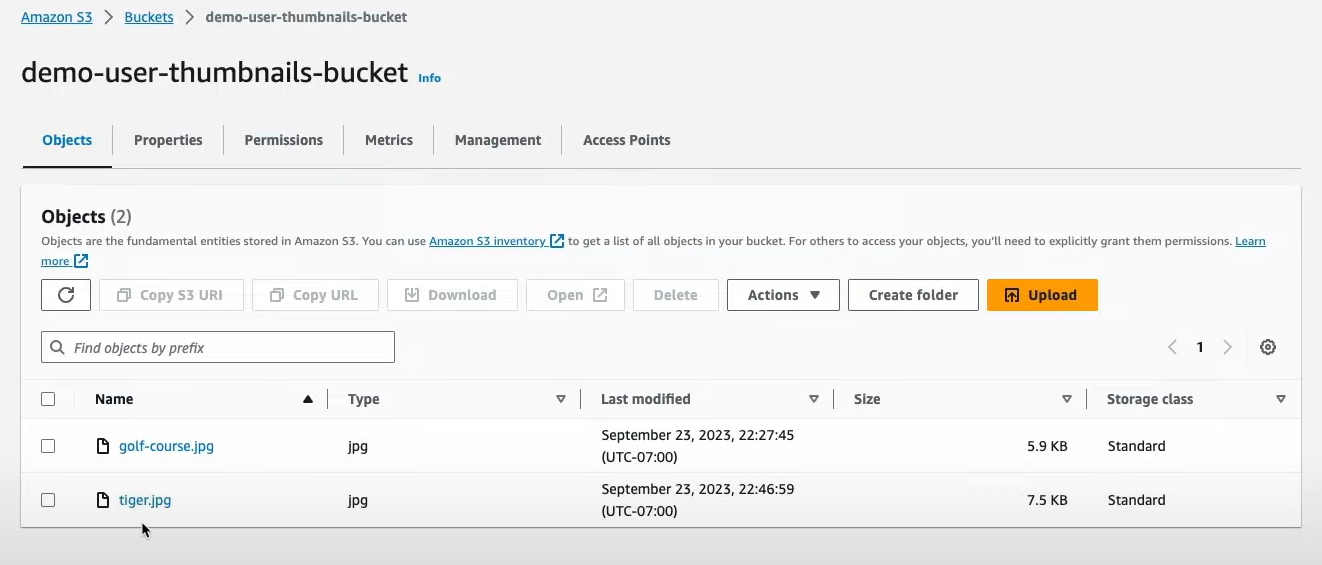
**Upload image in source Bucket**



**Original image**



**Destination Bucket**



**Resize Image**

