Project introduction

Topic:-Create a serverless image processing application that automatically resizes and optimizes images uploaded to an Amazon S3 bucket.

AWS S3 (Simple Storage Service) is a cloud data storage service. It is one of the most popular services of AWS. It has high scalability, availability, security and is cost effective. S3 has different storage tiers depending on the use case. Some common use cases of AWS S3 are:

- **Storage: It can be used for storing large amounts of data.**
- **Backup and Archive: S3 has different storage tiers based on how frequent the data is accessed which can be used to backup critical data at low costs.**
- **Static website: S3 offers static website hosting through HTML files stored in S3.**
- **Data lakes and big data analytics: Companies can use AWS S3 as a data lake and then run analytics on it for getting business insights and take critical decisions.**

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend

service without provisioning or managing servers. Lambda functions run on demand i.e. they execute only when needed and you pay only for what you compute. Lambda is well integrated with may other AWS services. It supports a wide variety of programming languages.

Some common use cases for AWS Lambda are:-

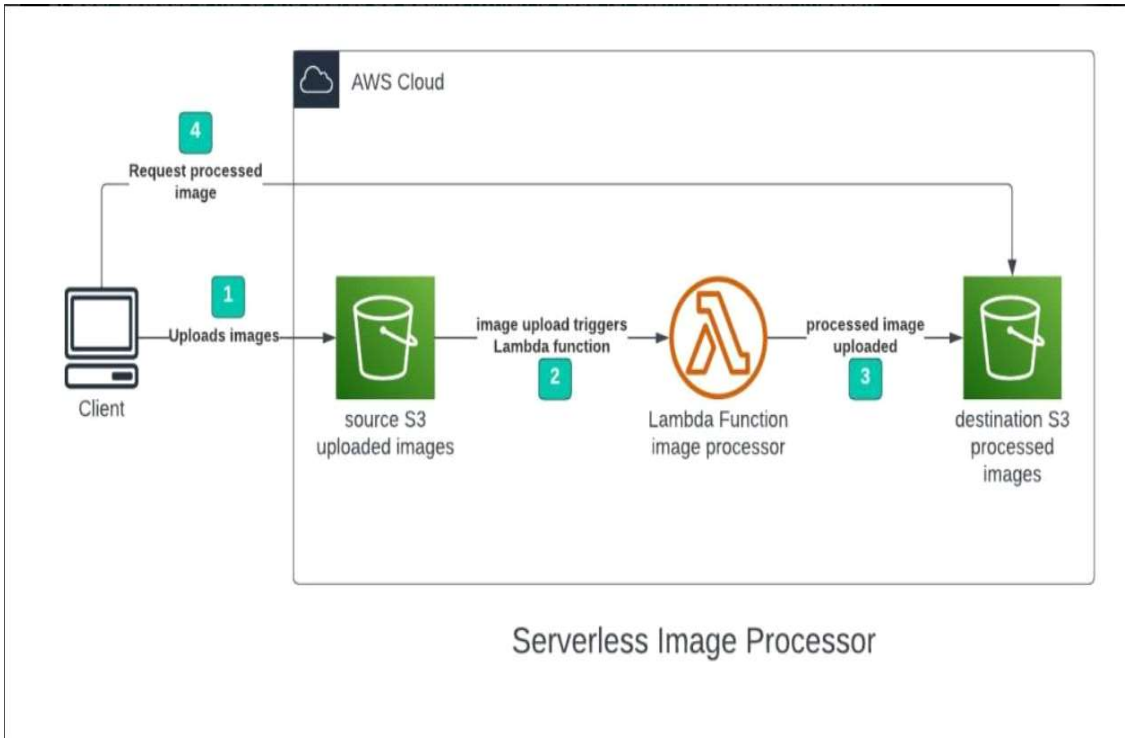
1.You can use Lambda for processing files as they are uploaded in an S3 bucket or whenever some event triggers the function.

2. Lambda can also be used for creating websites. This is cost effective because you are charged only for the time when the servers are running.

3. You can pass a data stream to your Lambda function and then create analysis from that.

Serverless Image Processing

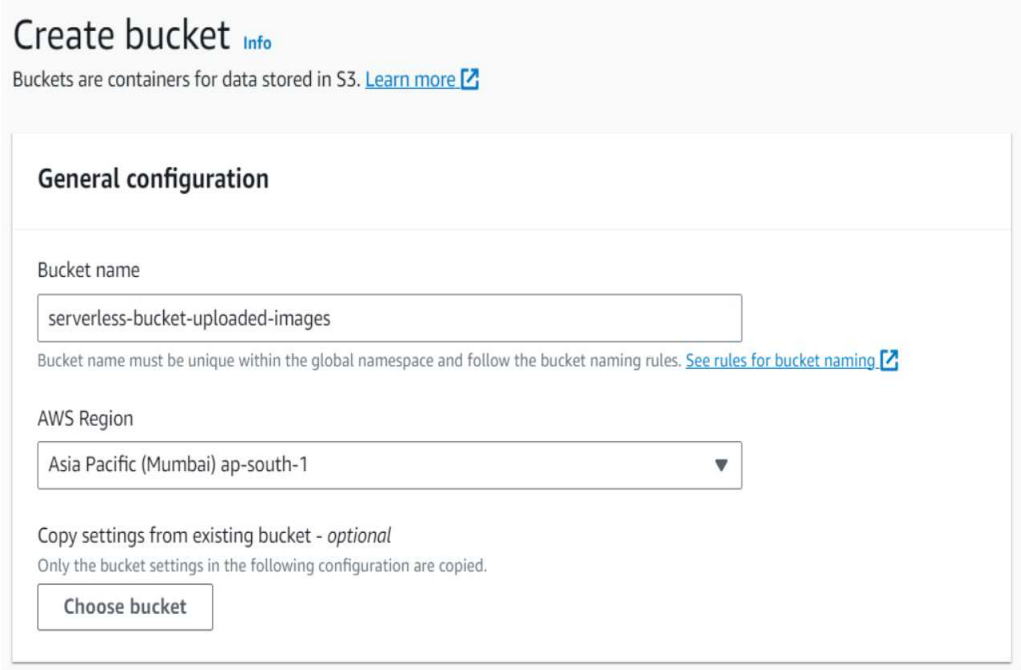
The Serverless Image Handler solution helps you embed images on your websites and mobile applications to drive user engagement. It uses the sharp Node.js library to provide high-speed image processing without sacrificing image quality. To minimize your costs of image optimization, manipulation, and processing,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



Step 1 – Creating S3 buckets

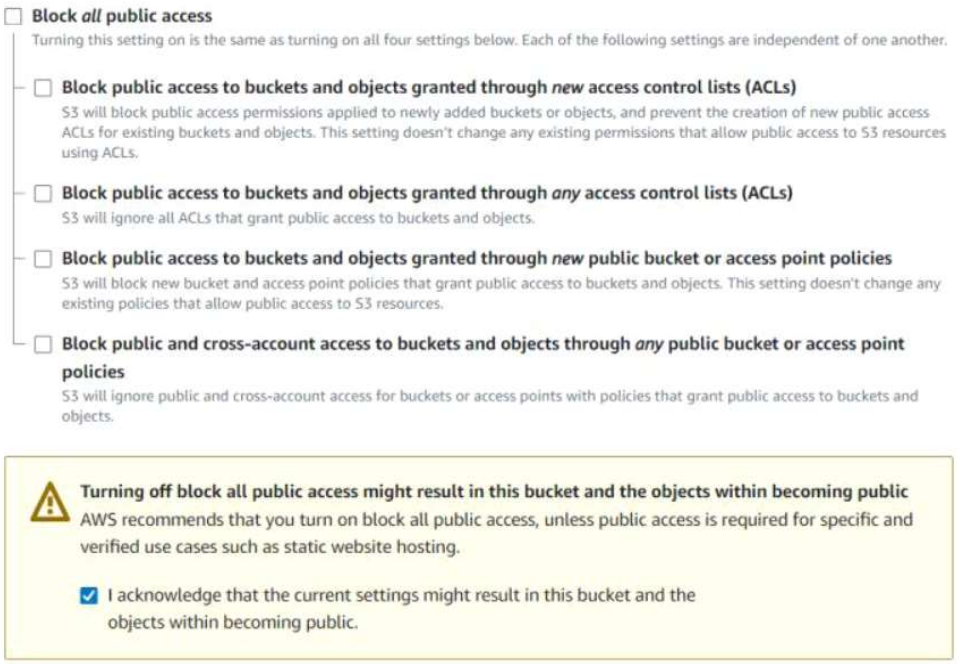
We will use two S3 buckets:

1. source Bucket: For storing uploaded images.
2. destination Bucket: For storing processed images.



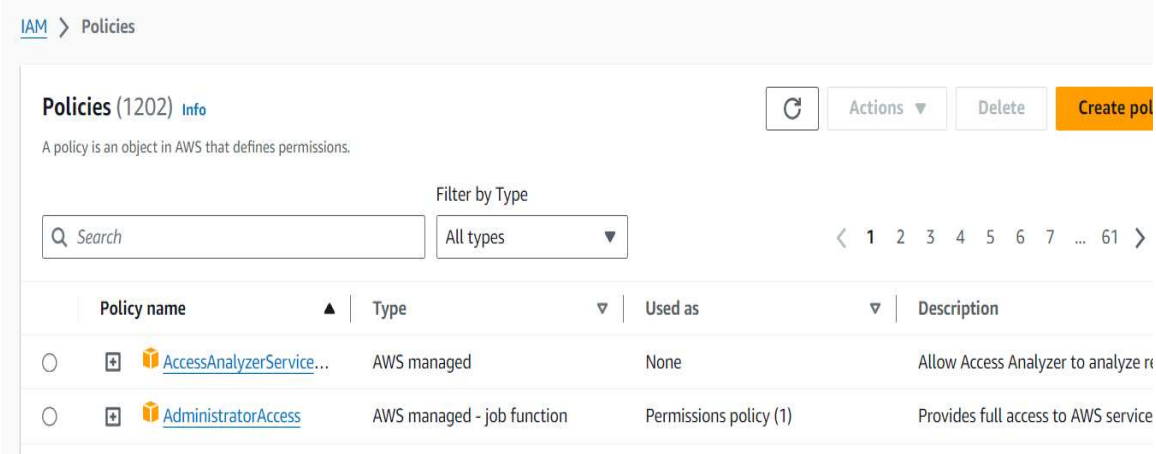
Step 2 – Configuring S3 bucket policy

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"],
```

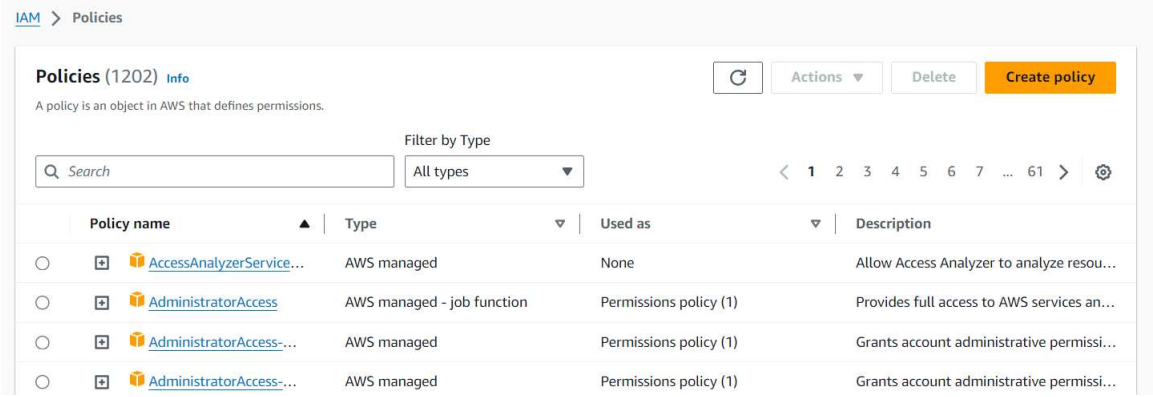
```
    "Resource": "arn:aws:s3:::DEST_BUCKET/*"
   }
  ]
}
```

***BUCKET_NAME = SOURSE BUCKET NAME**
***DEST_BUCKET = DESTINATION BUCKET NAME**

Step 4 –creating role in I am

Following Steps are Follows

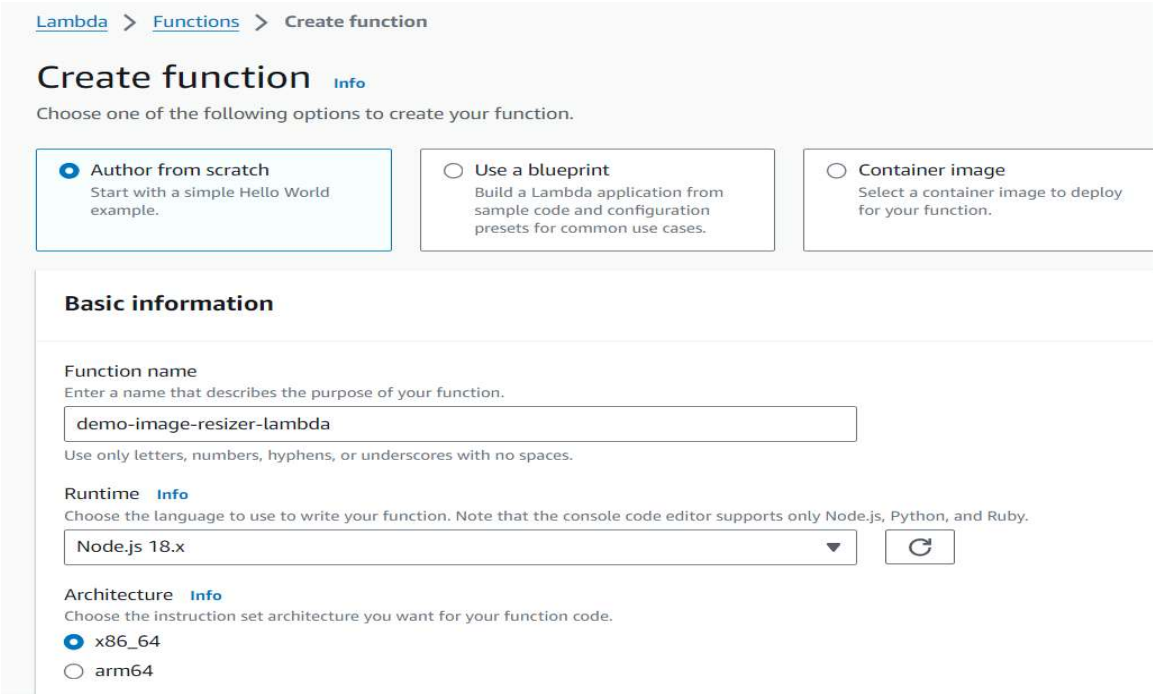
*Go to aws I am console
*Create role
*name imageresizerlambdarole
*Use case – Lambda
*Select-ImageBucketPolicy
*Then create role



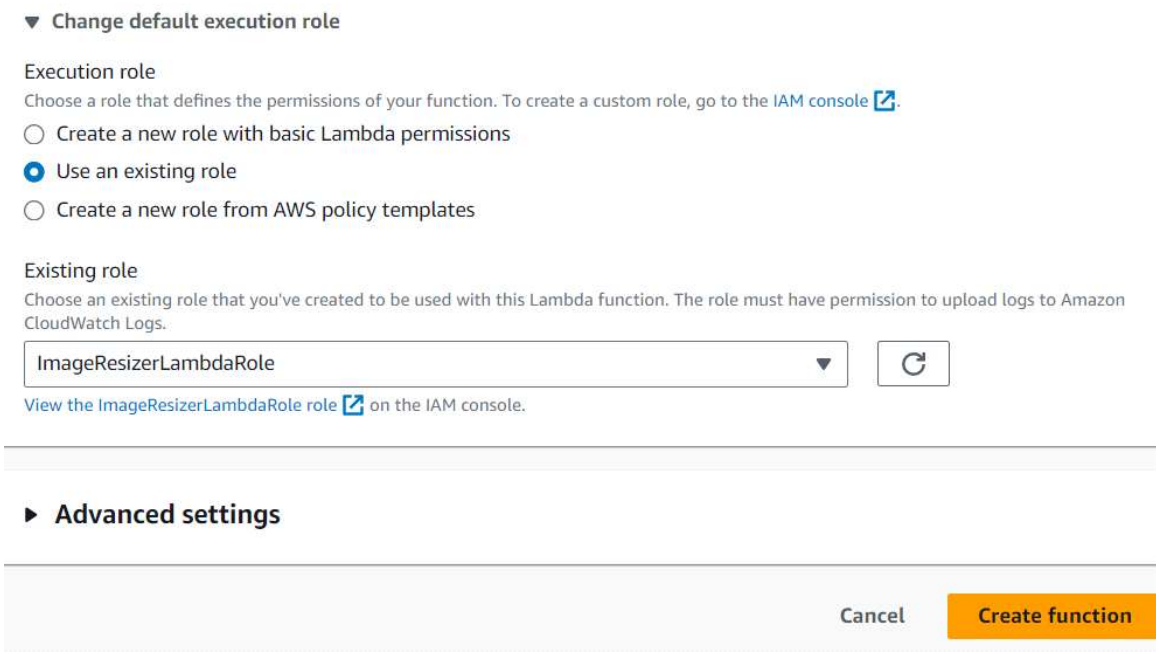
Step 3 – Creating Lambda function

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.

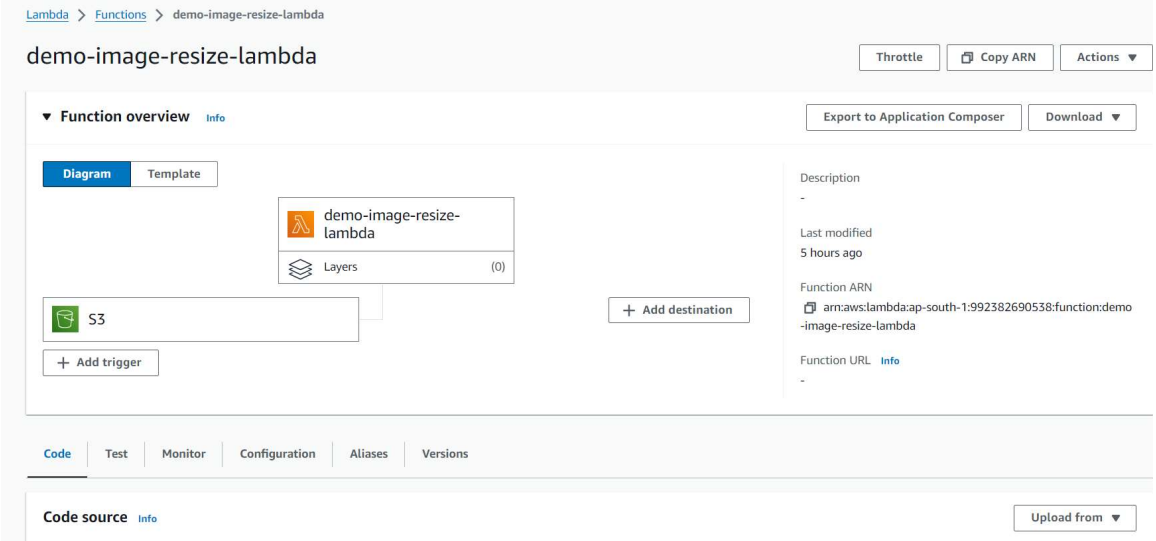


***Change default execution role**



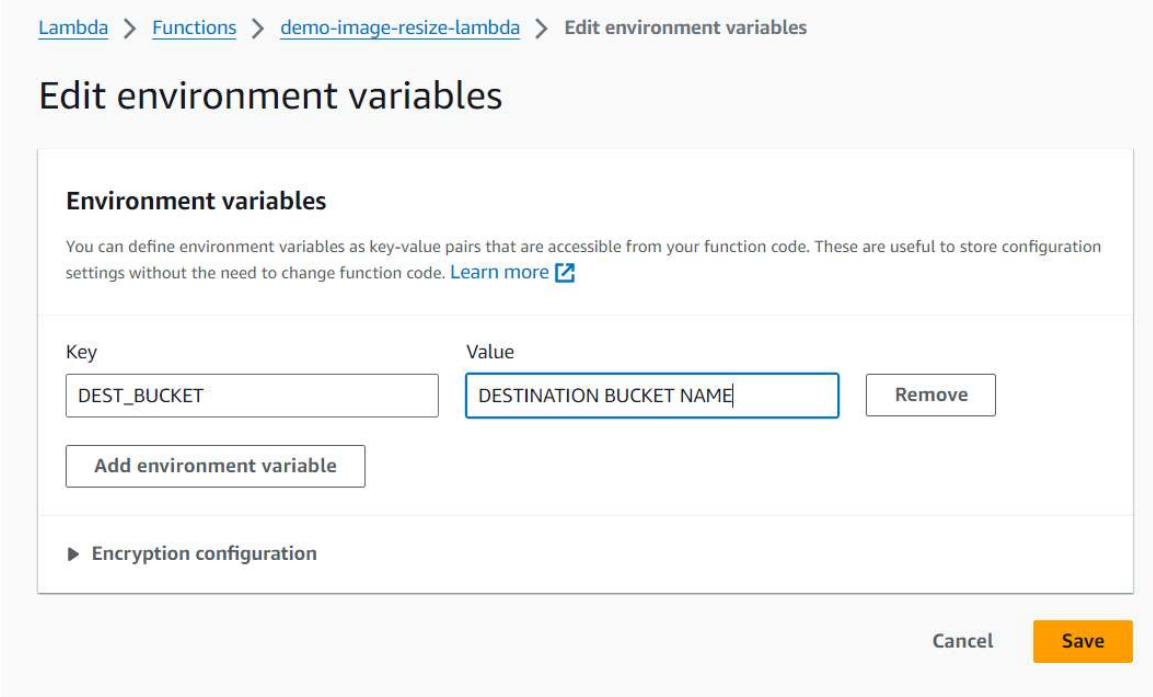
***Create function**

Step 6 -upload zip file in Lambda function



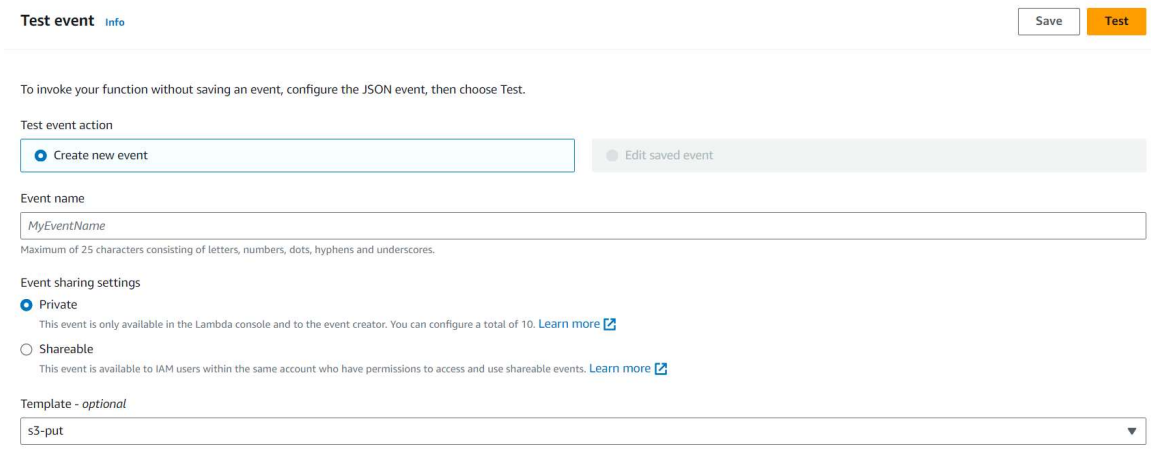
*Zip file link-https://github.com/OneLightWebDev/image-resizer-lambda

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

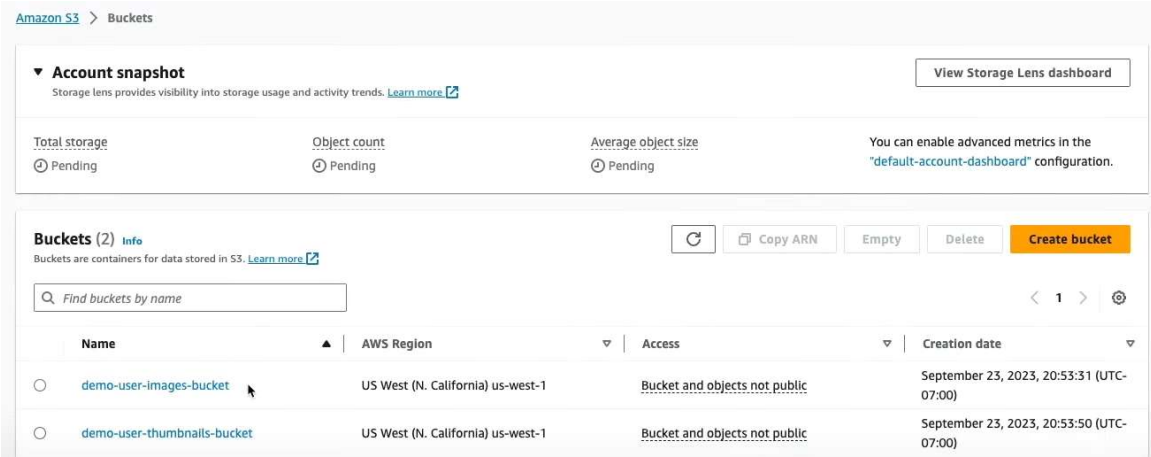


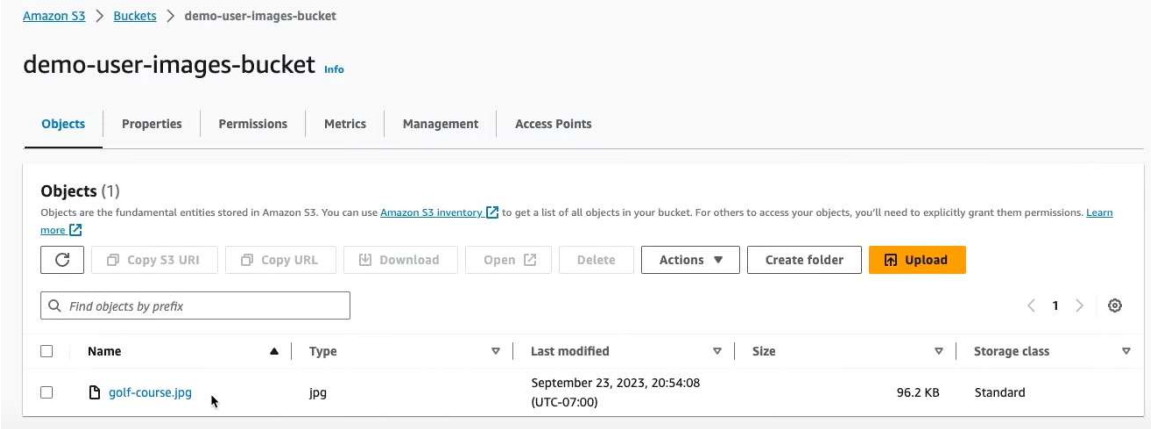
EVENT JSON

In event json we can change only 3 value
*name, arn, key

```
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambda
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "test%2Fkey",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

"name":"demo-user-images-bucket",->
"arn": "arn:aws:s3::: demo-user-images-bucket "->
"key": "golf-course.jpg",->

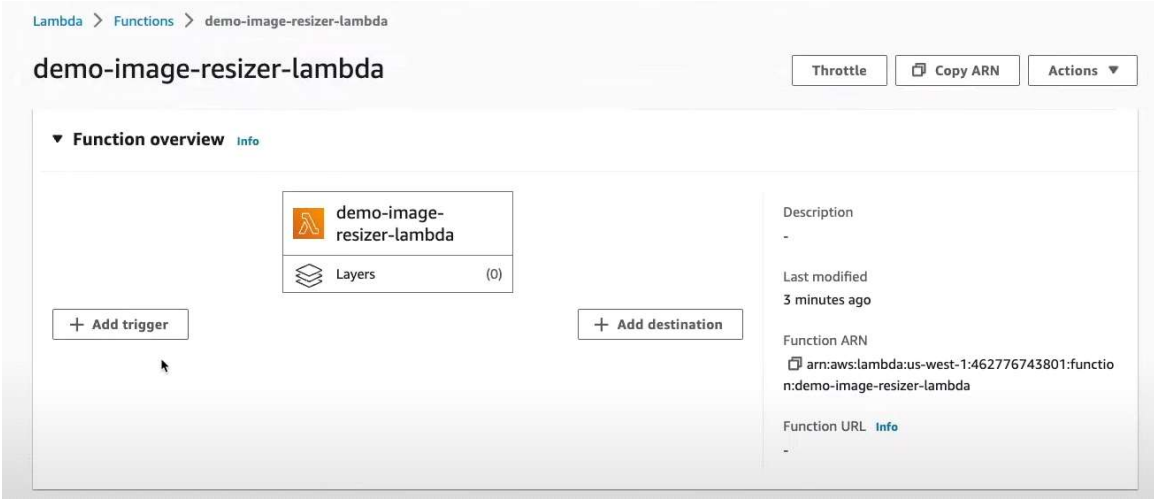


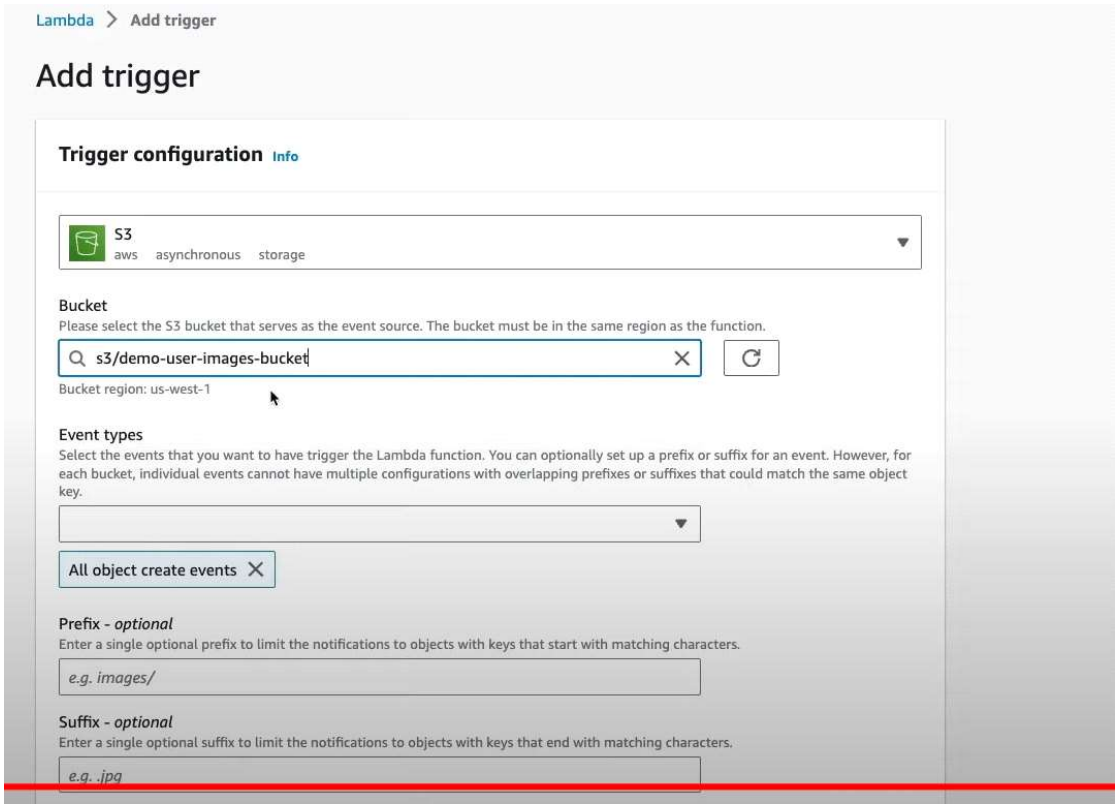


Now we can test

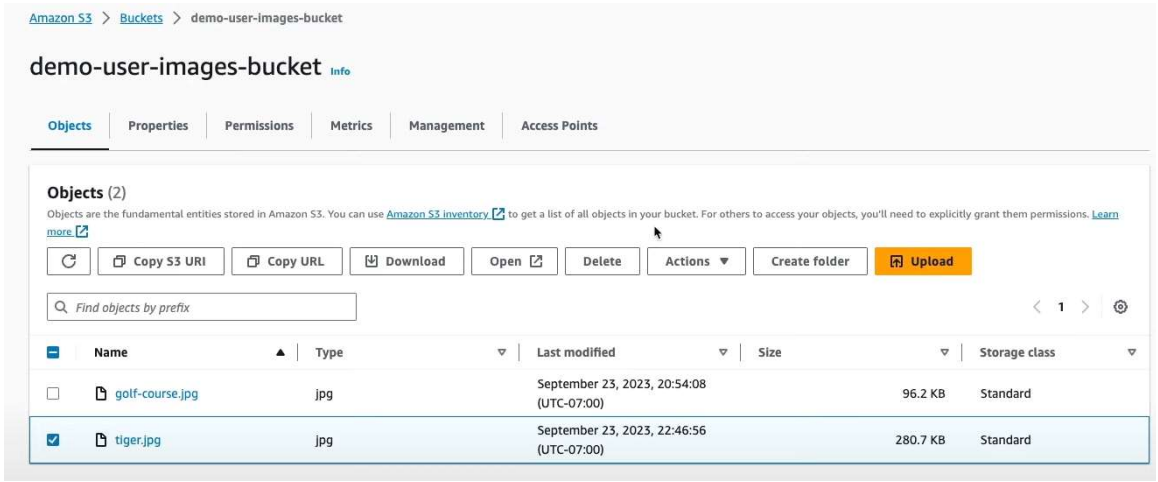
Step 5 – Creating S3 trigger

*Add trigger
*Select s3
*choose source Bucket name
*Now Add

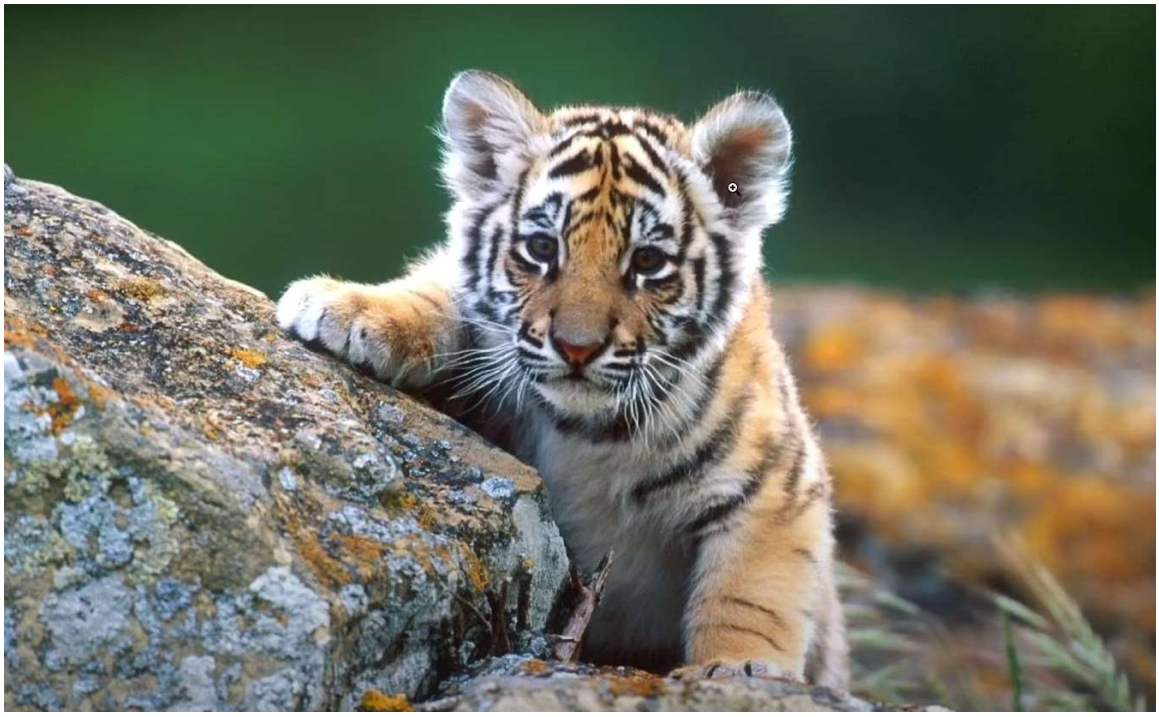




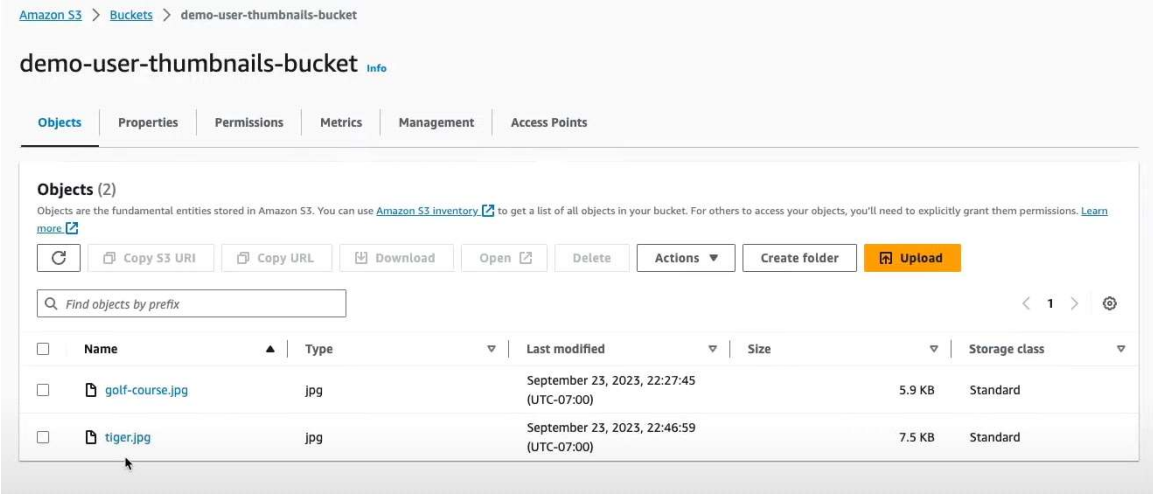
Upload image in source Bucket



Original image



Destination Bucket



Resize Image