Project 1

Serverless Image Processing

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

AWS Lambda

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. File processing: You can use Lambda for processing files as they are uploaded in an S3 bucket or whenever some event triggers the function.
2. Data and analytics: You can pass a data stream to your Lambda function and then create analysis from that.
3. Website: 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.

Why is AWS Lambda an essential part of the Serverless architecture?

When building Serverless applications, AWS Lambda is one of the main candidates for running the application code. Typically, to complete a Serverless stack you'll need:

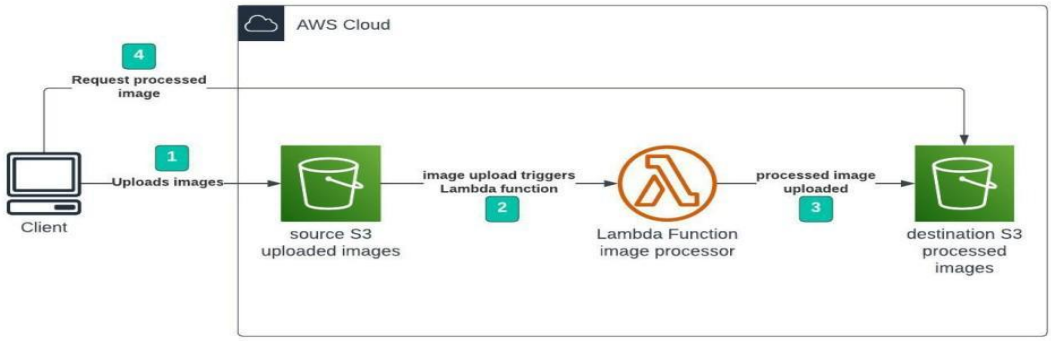
· a computing service

· a database service

· an HTTP gateway service

Serverless Image Processing Flow

- User uploads a file to the source S3 bucket (which is used for storing uploaded images).
- When the image is uploaded to a source S3 bucket, it triggers an event which invokes the Lambda function. The lambda function processes the image.
- Processed image is stored in the destination S3 bucket.



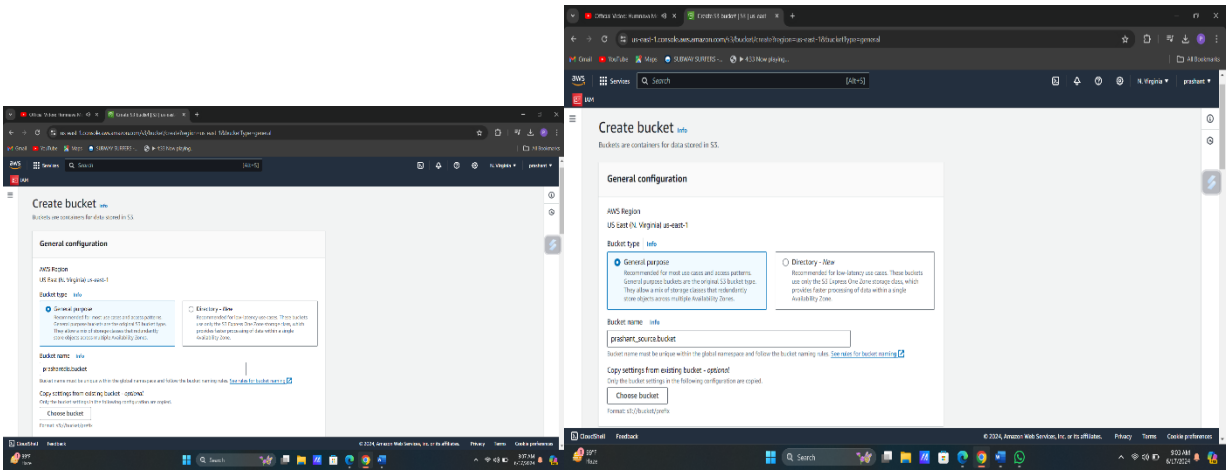
Serverless Image Processor

Step 1 – Creating S3 buckets

We will use two S3 buckets:

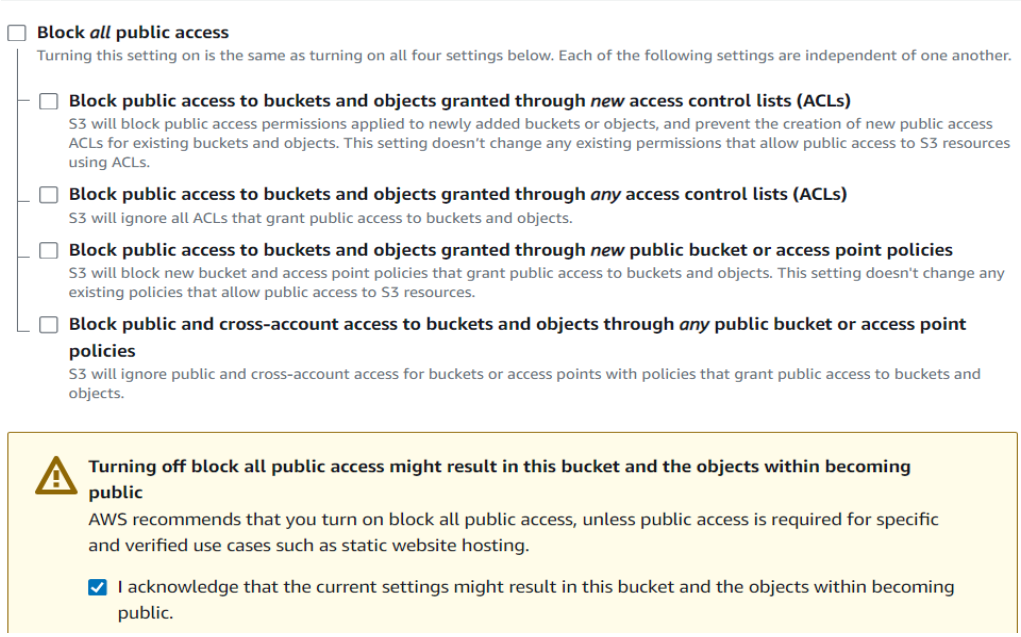
- source Bucket: For storing uploaded images.
- destination Bucket: For storing processed images.

Go to S3 console and click Create bucket. Enter bucket name as ‘serverless-bucket-uploaded-images’. Choose any AWS region as ‘ap-south-1’.



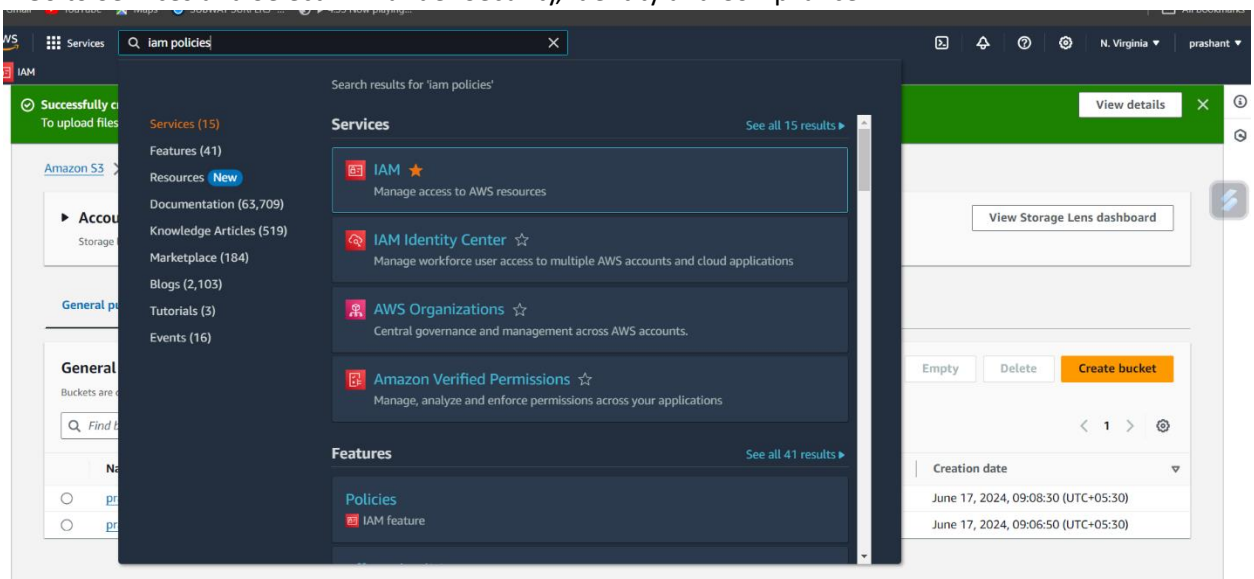
Step 2 – Configuring S3 bucket policy

In 'Block Public Access settings for this bucket' section disable "block all public access". You will get a warning that the bucket and its objects might become public. Agree to the warning. (Note: we are making this bucket public only for this project, it is not recommended to make an S3 bucket public if not needed).

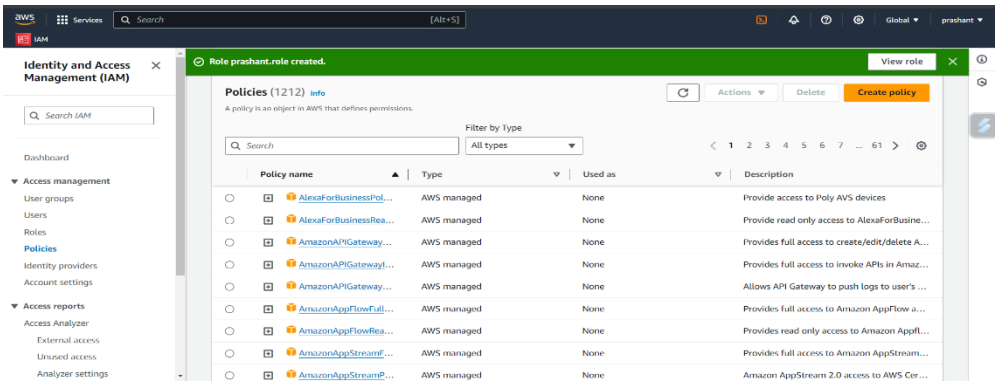


Task 3: Create an IAM Policy

1.Go to Services and Select IAM under Security, Identity and Compliance.



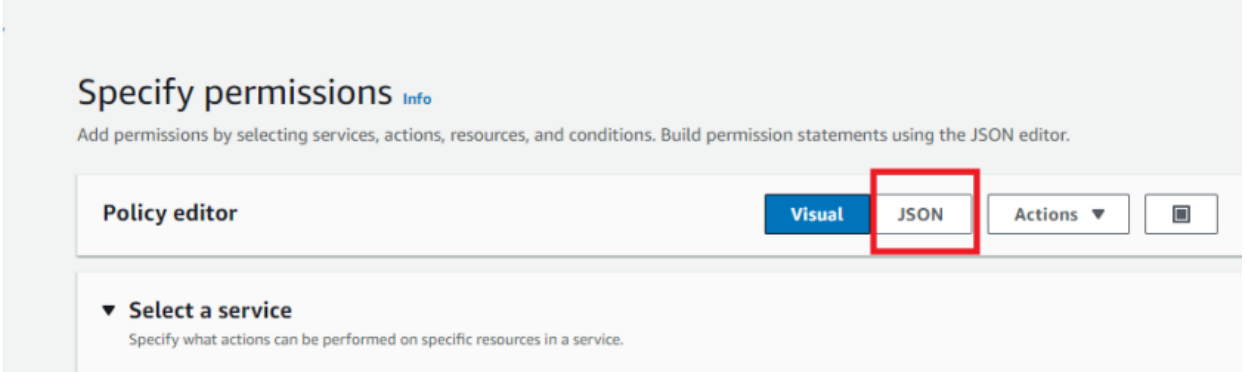
2. Click on Policies in the left navigation bar and click on the Create policy button.



3.Click on the JSON tab, Remove the existing code and copy-paste the below policy statement into the editor:

```
{
 "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",
```

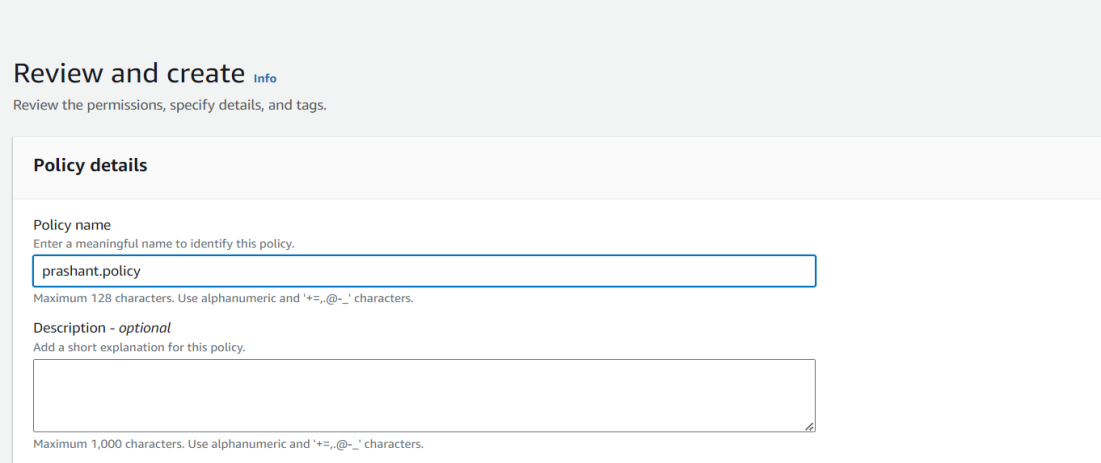
```
    "Action": ["s3:PutObject"],
    "Resource": "arn:aws:s3:::DEST_BUCKET/*"
  }
 ]
}
```



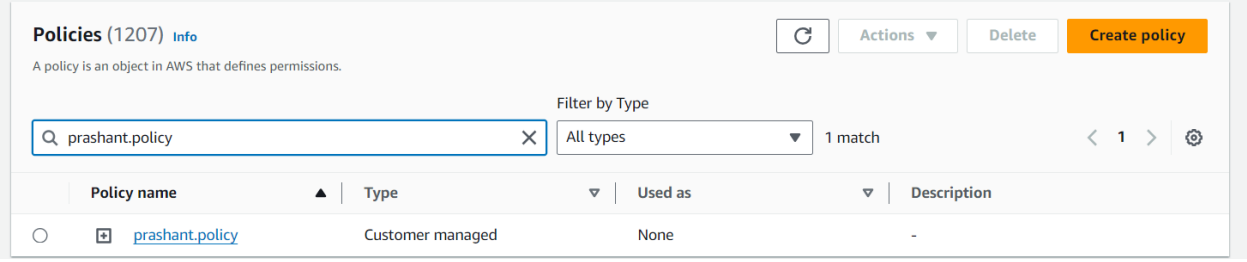
4.Leave Everything as default and click on Next button.

5.On the Review Policy page:

6. Enter Policy Name and Click on the Create policy button.

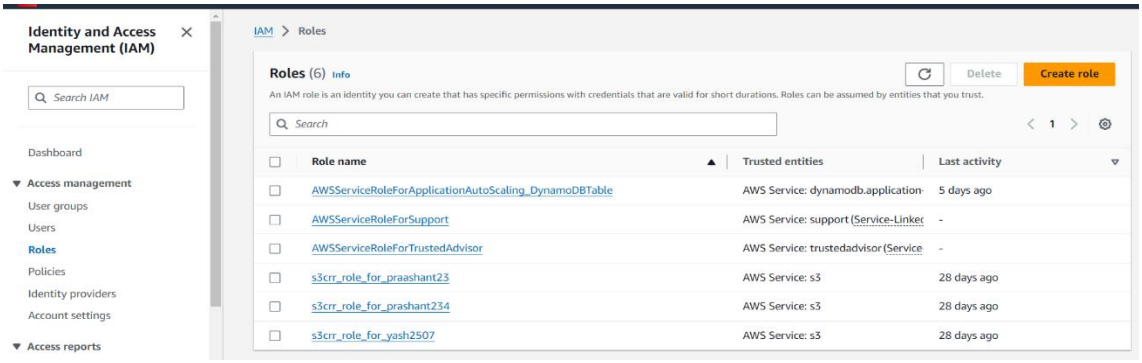


7.An IAM Policy with the name prashant.policy is created



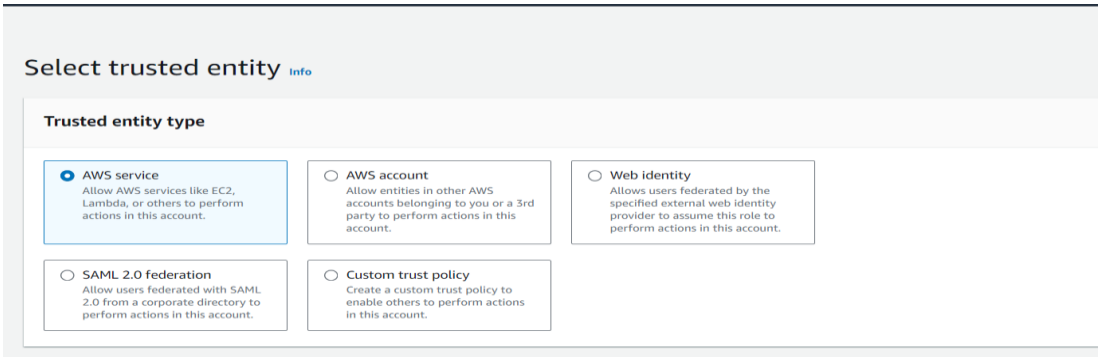
Task 4: Create an IAM Role

1.In the left menu, click on Roles and click on the Create Role button.

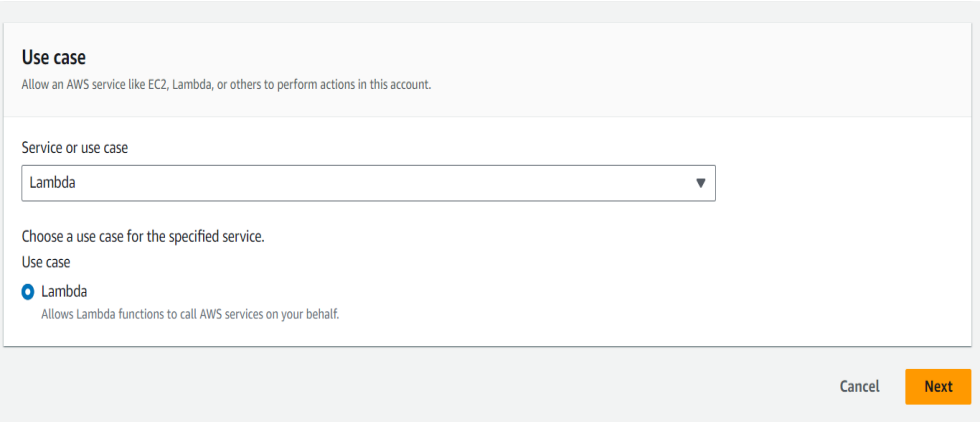


2. Select Lambda from AWS Services list.

- From Trusted Entity Type: Select AWS Service
- From Use case: Select Lambda

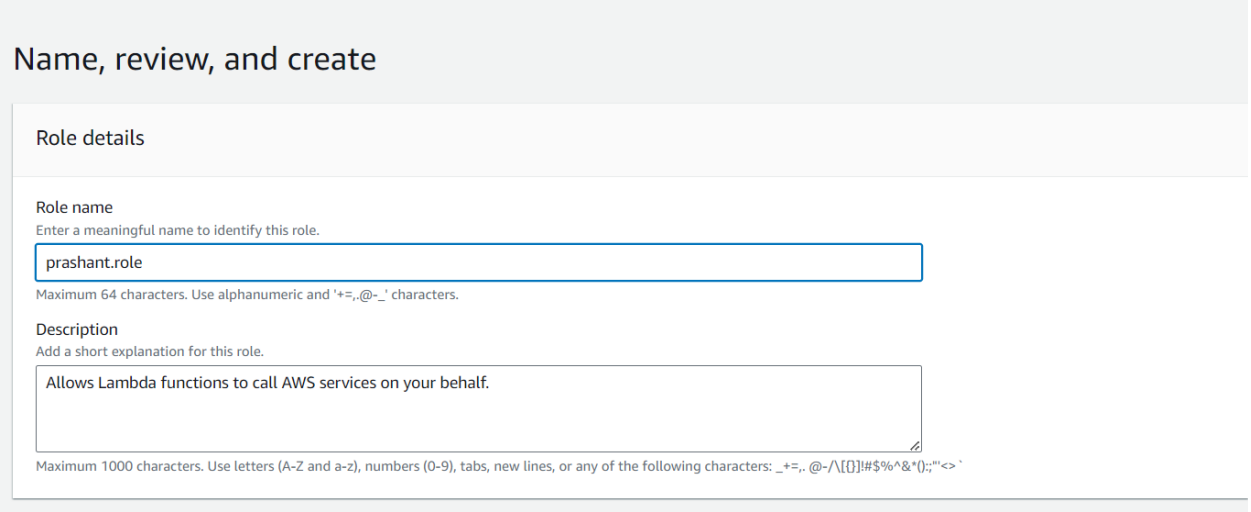


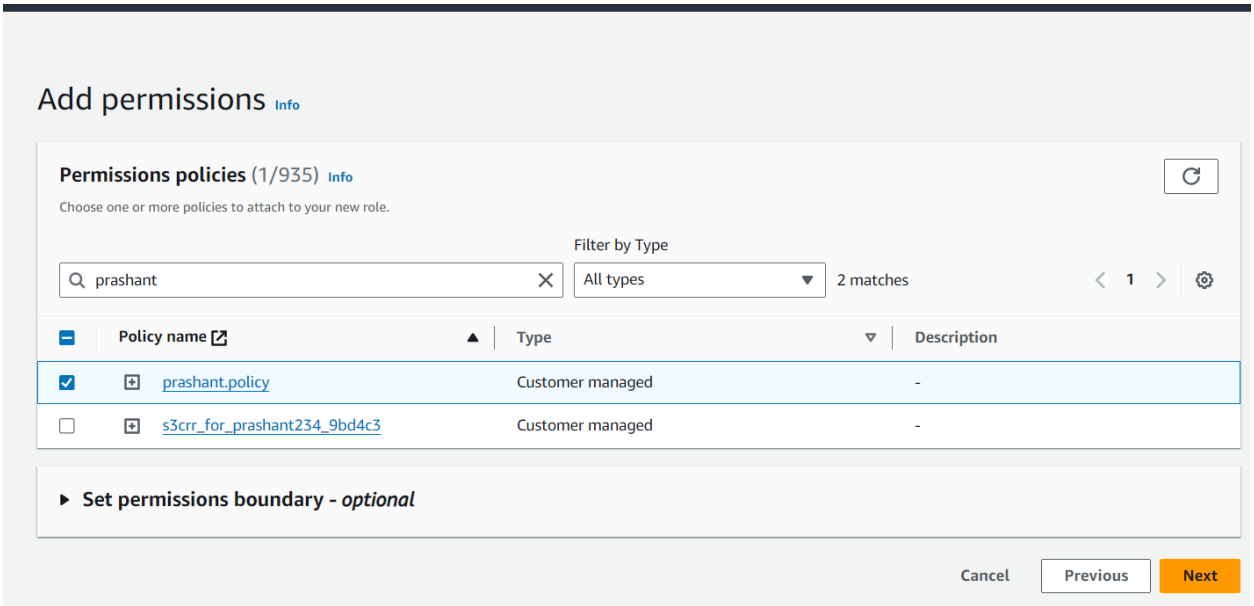
- Click on Next button.



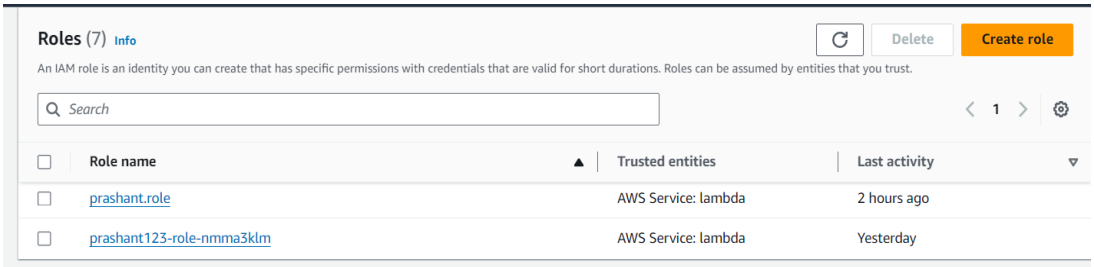
4.Role Name: Enter cloudrole

5.Click on the Create Role button



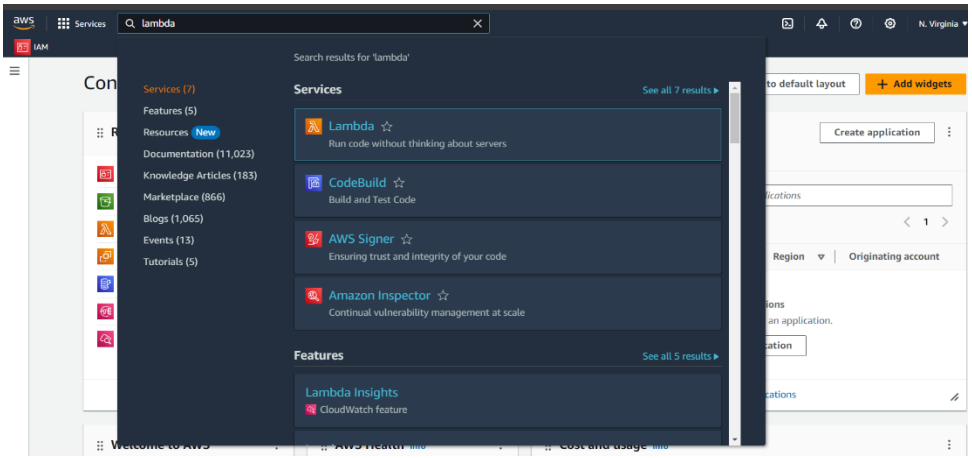


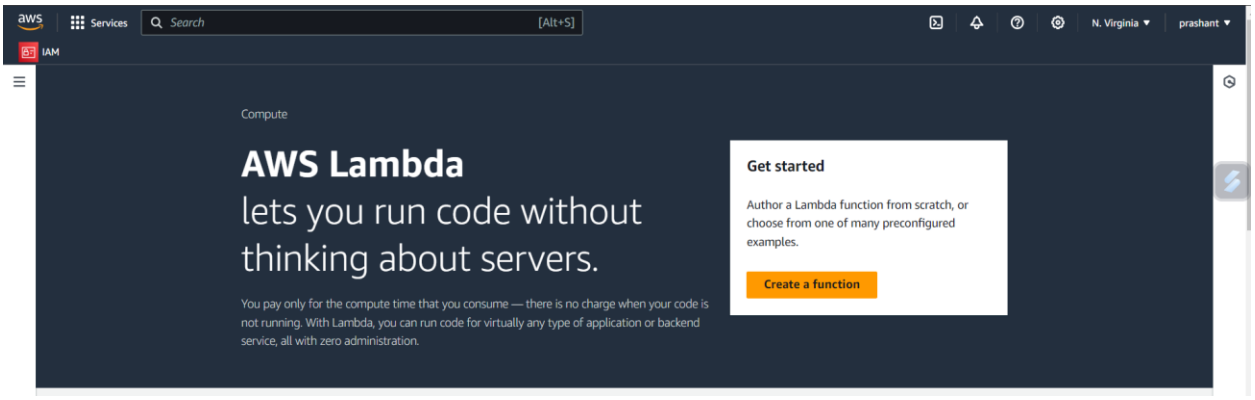
- You have successfully created an IAM role by name prashant.role.



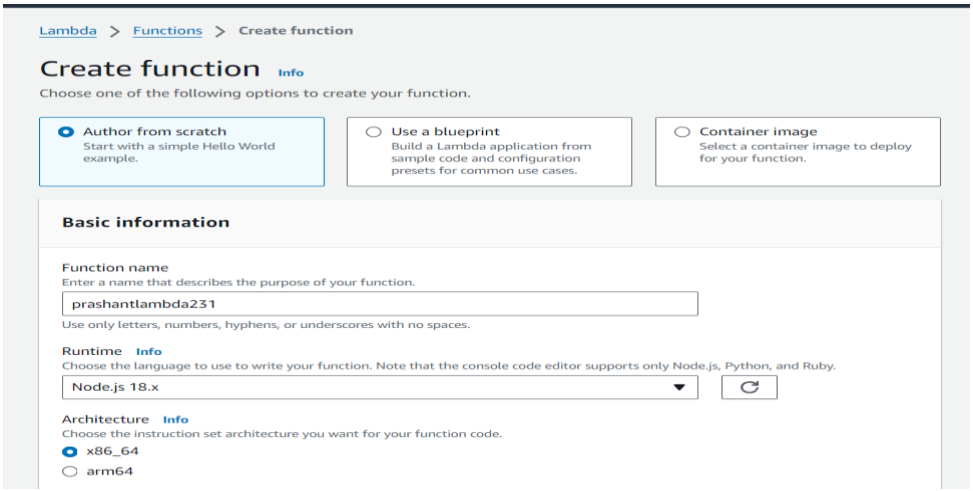
Task 5: Creating Lambda function

1. Go to AWS Lambda Console, Navigate to functions section . Click Create function

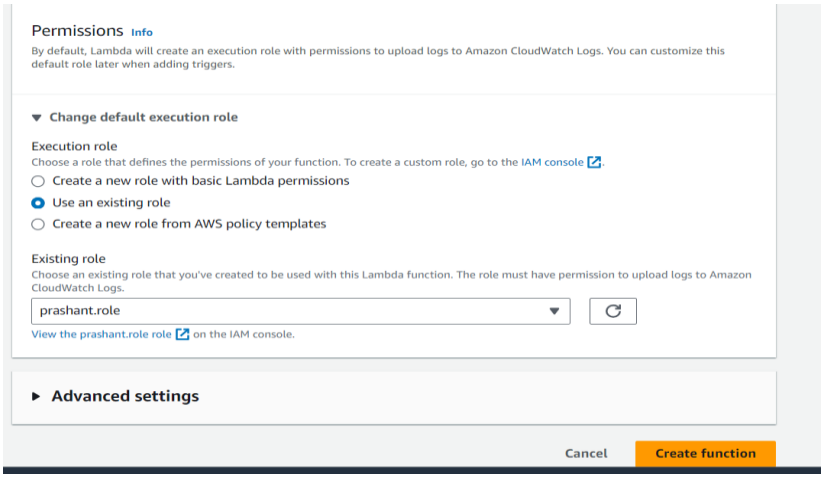




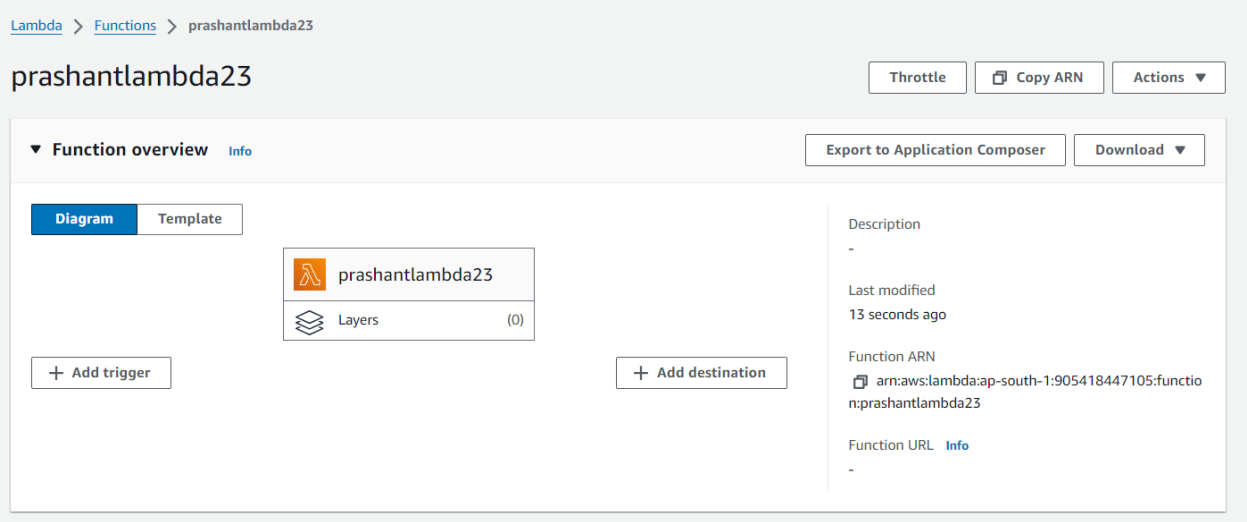
2. Name it and select runtime and Leave all other settings as default



3. Change Default execution role and create function

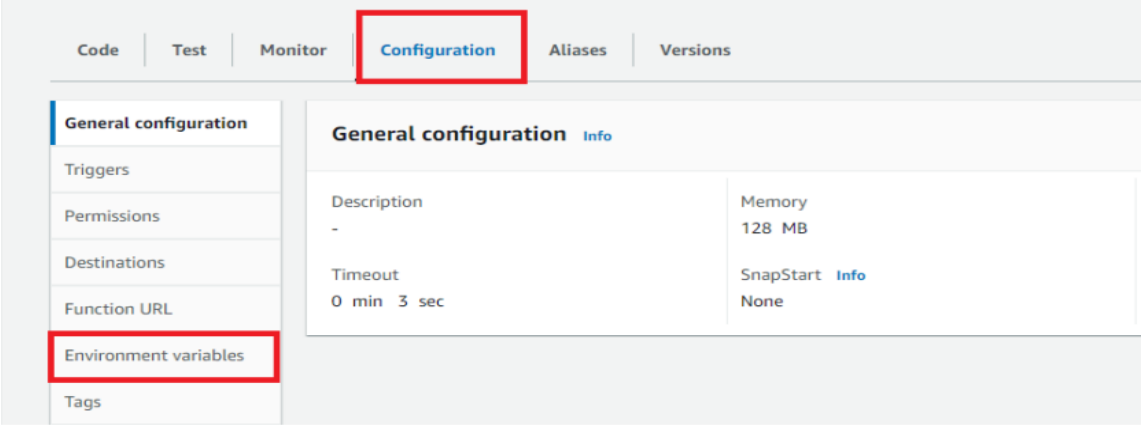


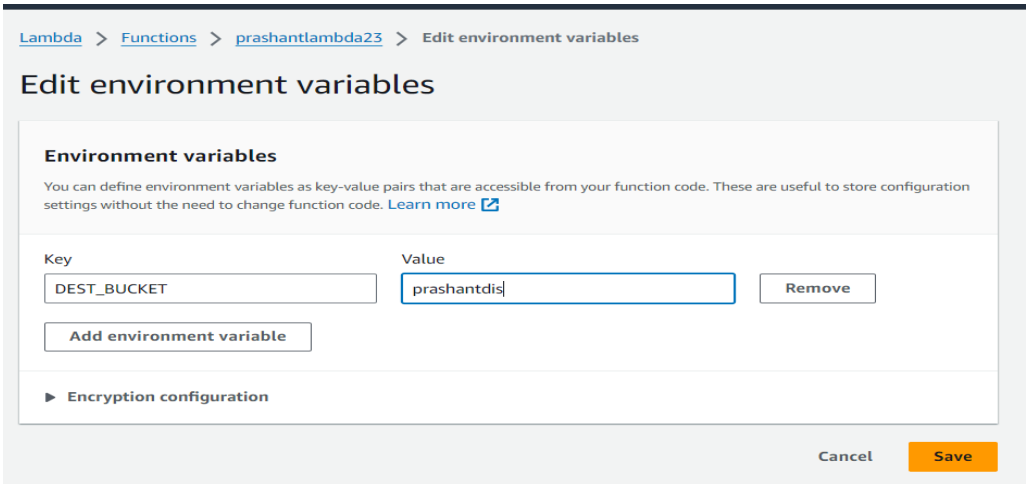
Task 6 -upload zip file in Lambda function



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

4.Edit Environment Variable





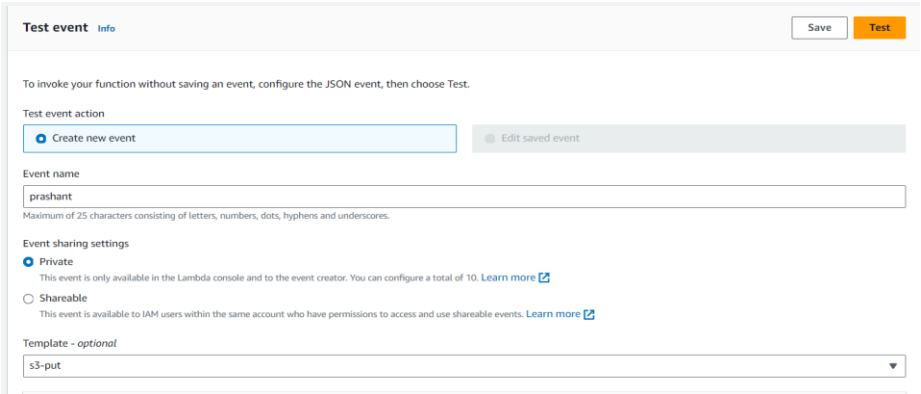
Task 7: Test Lambda Function

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

*open function then will be created

*open test console

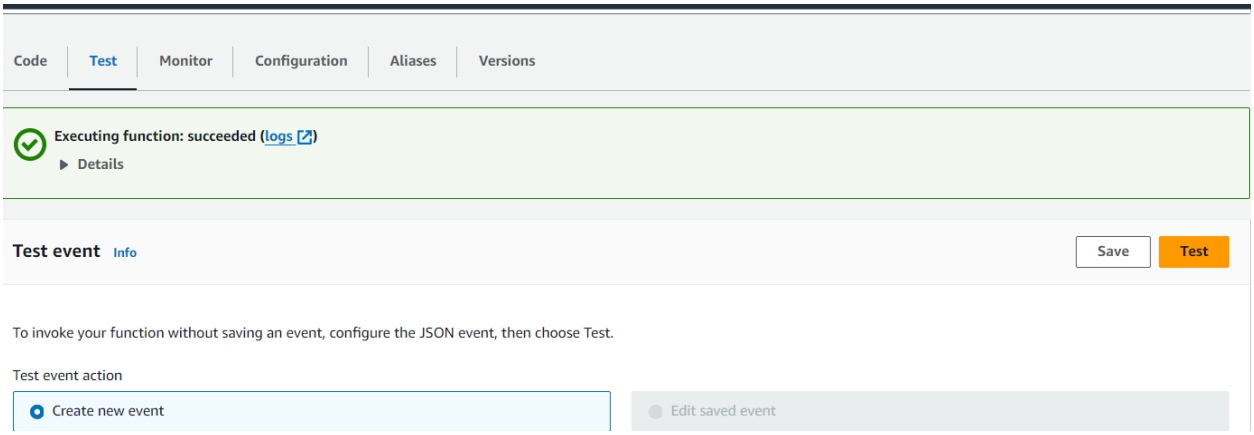
*template=s3-put



{
"Records": [

```
   {
    "eventVersion": "2.0",
    "eventSource": "aws:s3",
    "awsRegion": "us-east-1",
    "eventTime": "1970-01-01T00:00:00.000Z",
    "eventName": "ObjectCreated:Put",
    "userIdentity": {
     "principalId": "EXAMPLE"
    },
    "requestParameters": {
     "sourceIPAddress": "127.0.0.1"
    },
    "responseElements": {
     "x-amz-request-id": "EXAMPLE123456789",
     "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
    },
    "s3": {
     "s3SchemaVersion": "1.0",
     "configurationId": "testConfigRule",
     "bucket": {
      "name": "prashantsource",
      "ownerIdentity": {
       "principalId": "EXAMPLE"
      },
      "arn": "arn:aws:s3:::prashantsource"
     },
     "object": {
      "key": "test%2Fkey",
      "size": 1024,
      "eTag": "0123456789abcdef0123456789abcdef",
      "sequencer": "0A1B2C3D4E5F678901"
     }
    }
   }
 ]
}
```


Now We can Test:



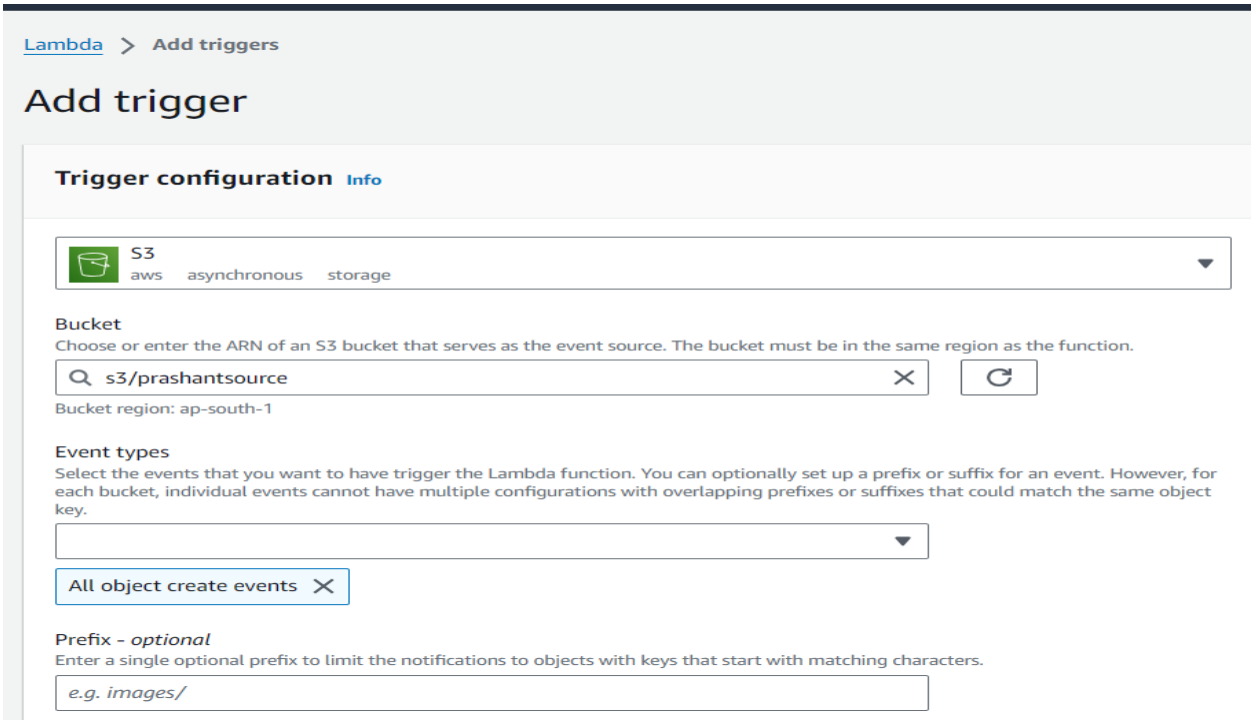
Task 8: Creating S3 Trigger

*Add trigger

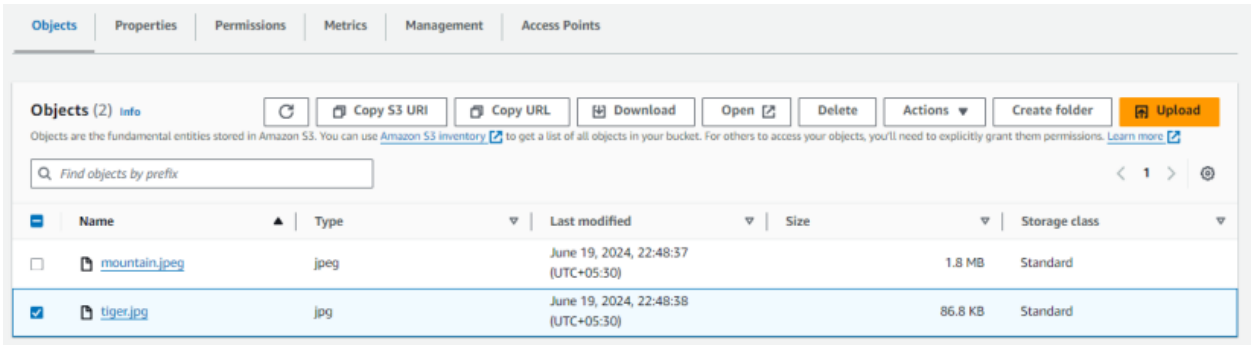
*Select s3

*choose source Bucket name

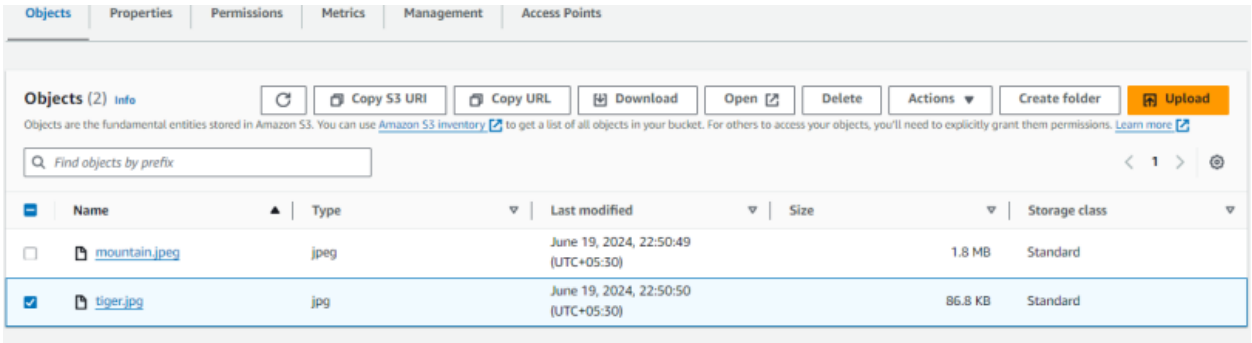
*Now Add



Task 9: Upload image in Source Bucket



Original Image



Resize Image