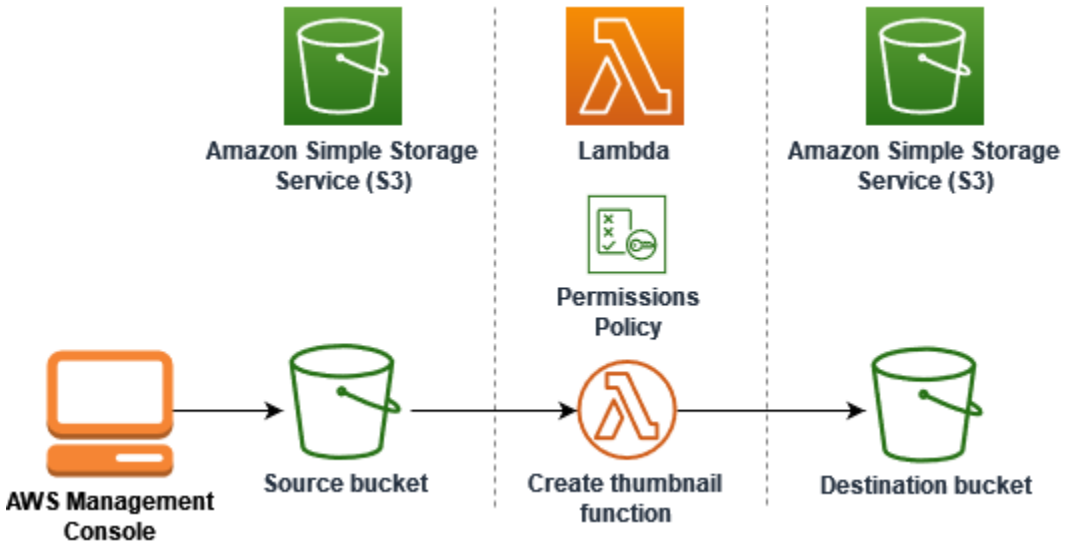
Name:Atharv Sanjay Nikam **Div:D15C** **Roll:36**

ADVANCE DEVOPS PRACTICAL EXAM

AWS Case Study: AWS Lambda-S3 Image Resizing using Python

1. Introduction

Case Study Overview: This case study demonstrates the implementation of an automated image resizing system using AWS Lambda and Amazon S3. By configuring a serverless architecture, the project enables dynamic resizing of images uploaded to an S3 bucket, generating thumbnails which are automatically stored in a different S3 bucket. The entire process is event-driven, triggered by the upload of an image, which invokes the Lambda function. This case study highlights the power of cloud-native technologies and serverless computing, eliminating the need for managing infrastructure while ensuring scalability and efficiency.



Key Feature and Application: The standout feature of this project is its seamless automation of image resizing using an event-driven approach. As soon as an image is uploaded to the designated S3 bucket, the Lambda function is invoked to process the

image, resize it, and save a smaller thumbnail version in the destination bucket. This is especially useful in scenarios where optimized image delivery is critical, such as websites, mobile applications, and content delivery networks (CDNs). The resized images reduce bandwidth usage and improve page loading times, resulting in better user experiences.

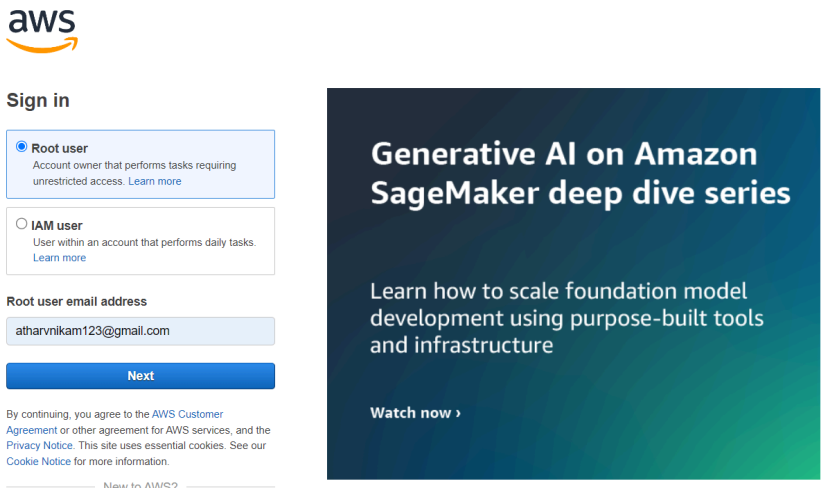
Furthermore, the solution is highly scalable, as AWS Lambda automatically handles the scaling of functions in response to increased traffic, without requiring any manual intervention. It is also cost-effective, as you only pay for the compute time you use, making it ideal for applications with fluctuating demand. By leveraging fully managed services like S3 and Lambda, this architecture not only ensures high availability but also reduces the complexity of managing backend servers and storage systems.

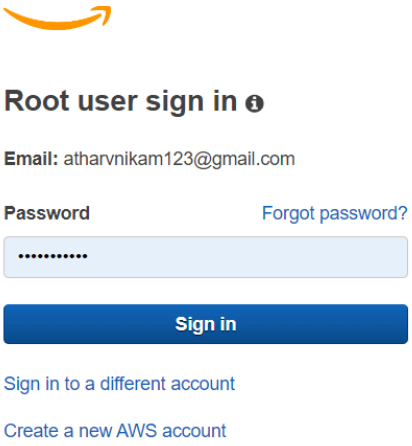
2. Step-by-Step Explanation

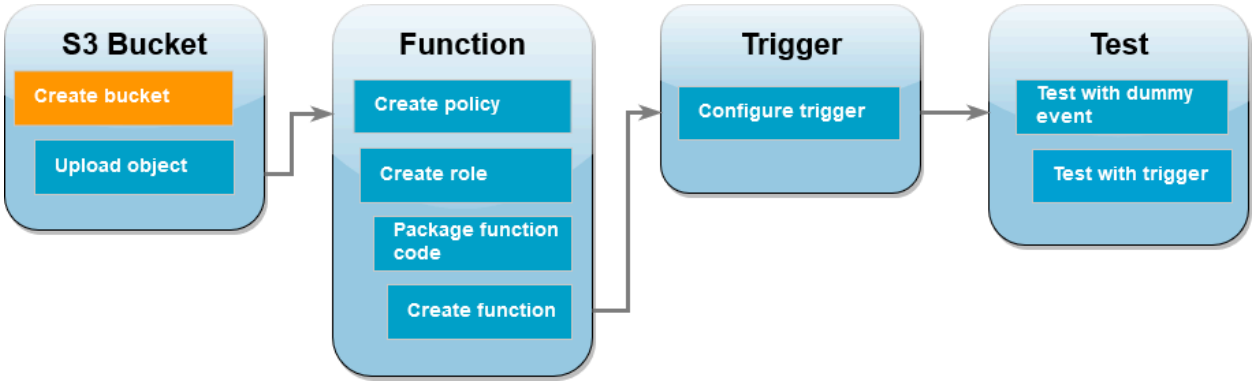
Step 1: Initial Setup – Creating S3 Buckets

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

Sign in to the AWS Management Console as the account owner by choosing Root user and entering your AWS account email address. On the next page, enter your password. For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide.*





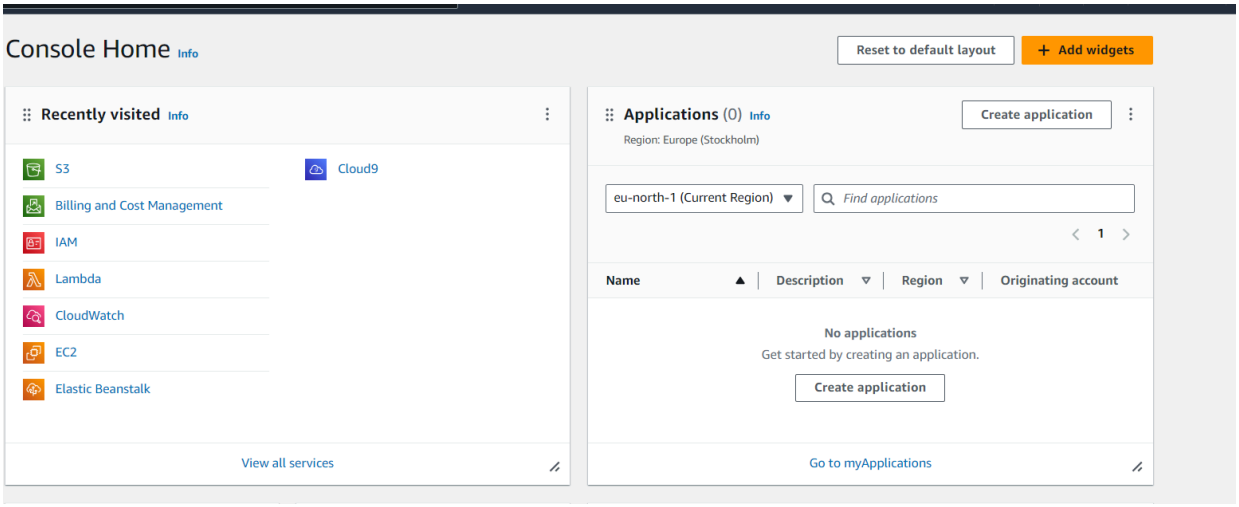


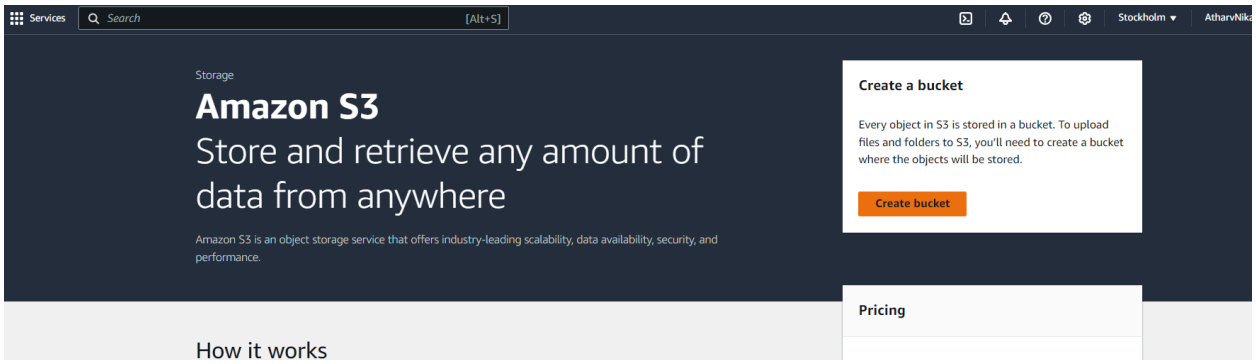
Create Source and Destination Buckets:
I created two S3 buckets – one for uploading the original images (source bucket) and another to store the resized images (destination bucket).

Example:

- Source Bucket: `atharv-s3-source-bucket`
- Destination Bucket: `atharv-s3-source-bucket-resized`

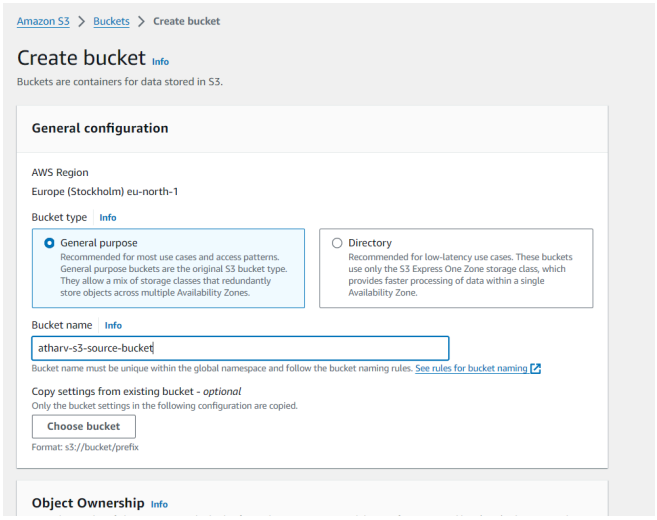
Open the S3 Management Console: Start by navigating to the Amazon S3 console. This is where you'll create and configure your S3 buckets.

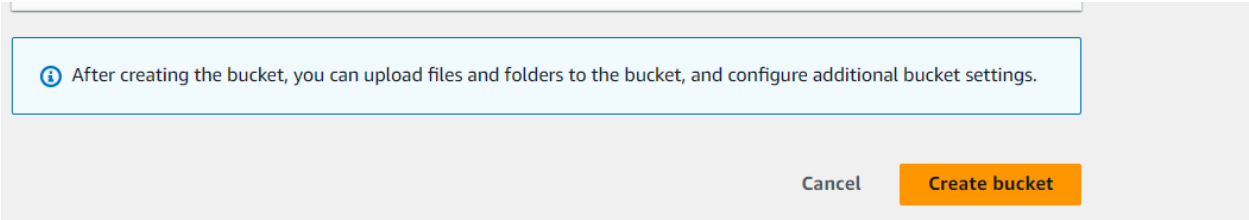




Create a Source Bucket:

- Choose **Create Bucket**.
- In the **General Configuration** section, provide a globally unique name for your bucket, adhering to the Amazon S3 bucket naming rules. The name should contain only lowercase letters, numbers, dots (.), and hyphens (-). For example: `atharv-s3-source-bucket`.
- Select the AWS Region that is geographically closest to you. Ensure that your Lambda function is created in the same region later in the process.
- Leave the other settings at their default values and click **Create Bucket**.

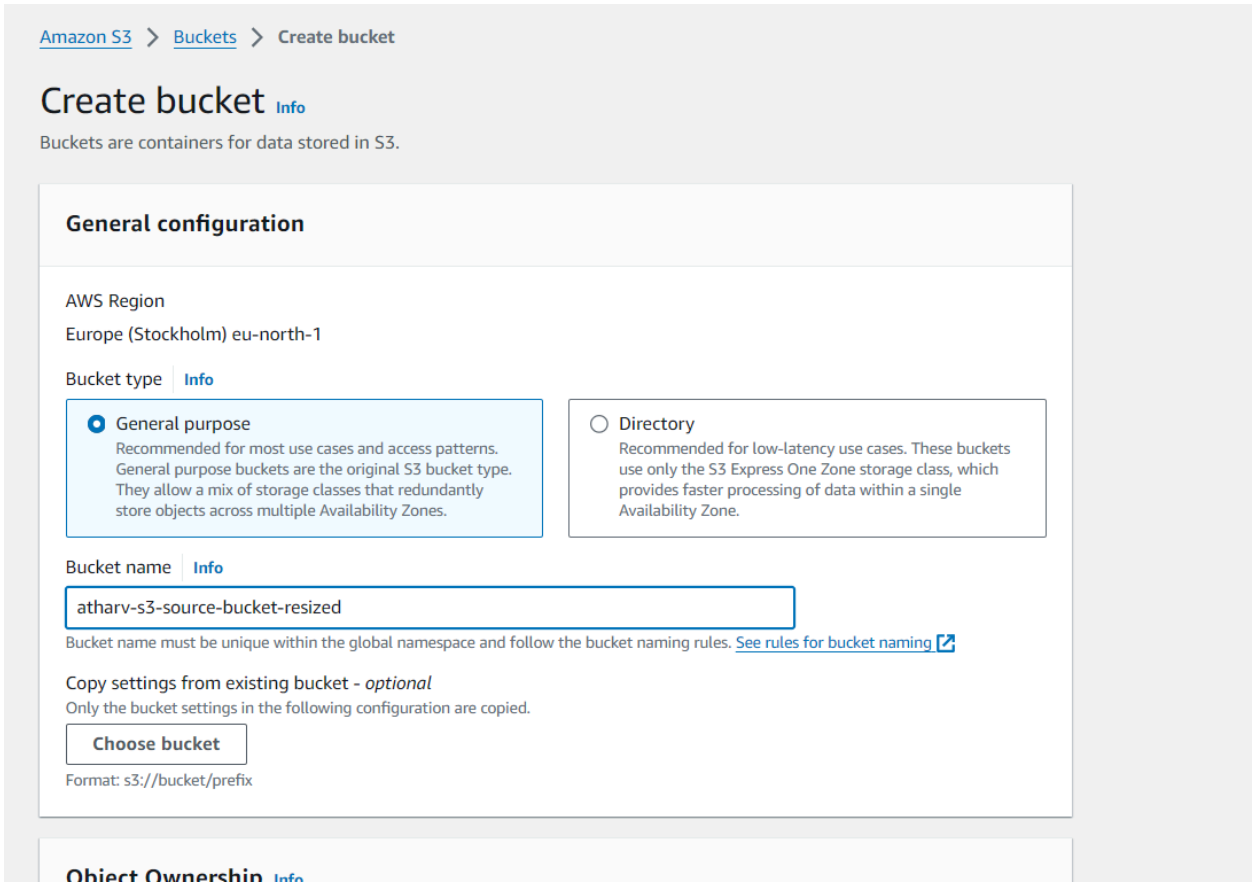


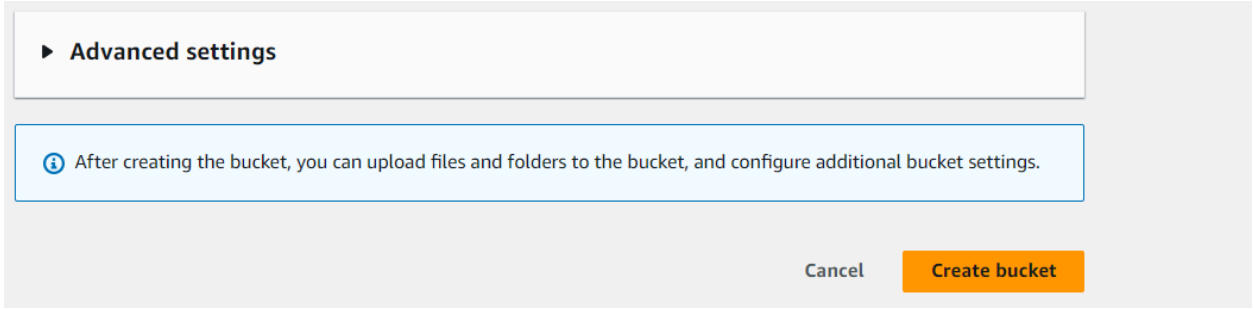


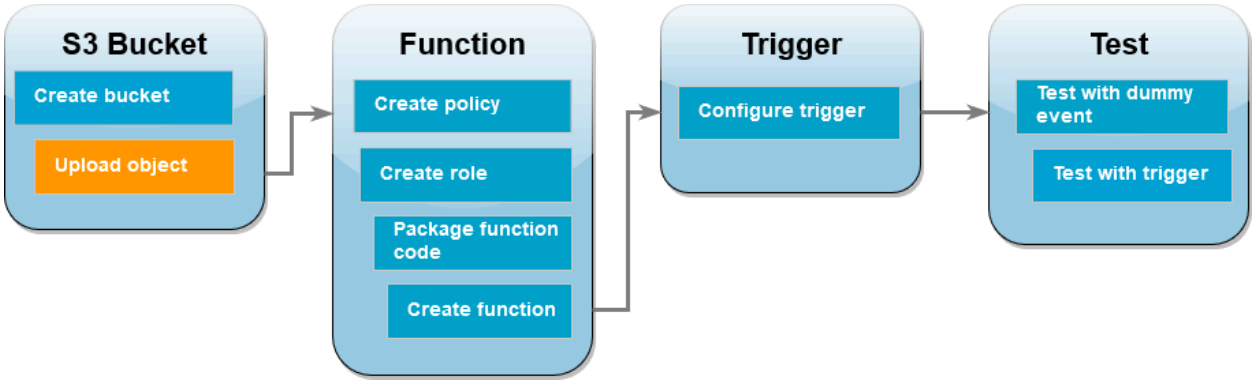
Create a Destination Bucket:

- Repeat the above steps to create a destination bucket, which will store the resized images. For example, name it `atharv-s3-source-bucket-resized`.

Name:Atharv Sanjay Nikam Div:D15C Roll:36





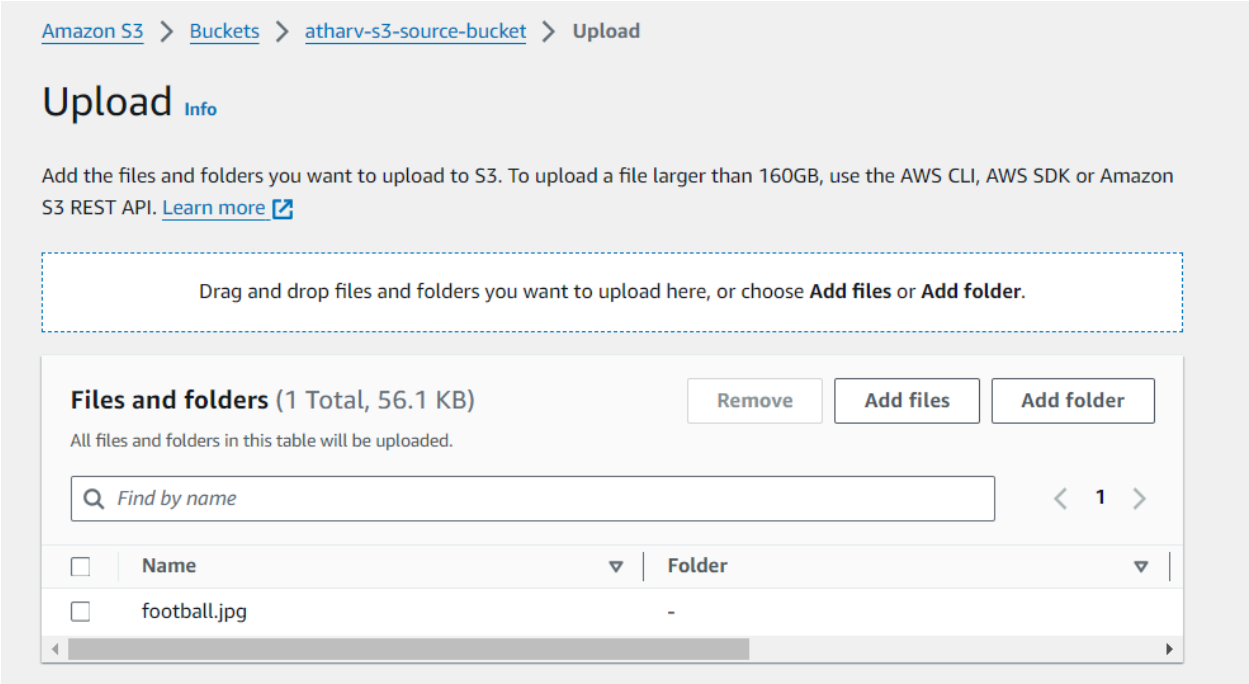


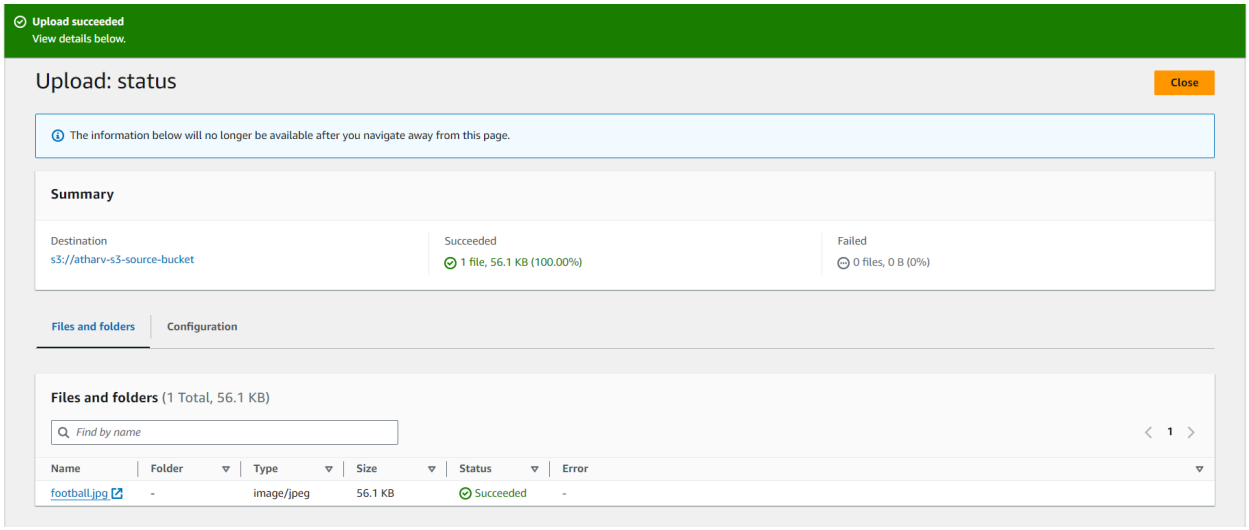
Uploading a Test Image to the Source Bucket (via AWS Console)

Once the Lambda function is set up, you'll need to test its functionality by uploading an image file. This is an essential step to ensure the Lambda function is triggered and processes the image as expected.

Using the AWS Management Console:

1. Open the **Amazon S3** console and navigate to the **Buckets** page.
2. Select the **source bucket** you previously created.
3. Click on **Upload**.
4. Choose **Add files** and select any **JPG or PNG** file you want to test.
5. After selecting the file, click **Open**, then choose **Upload**.





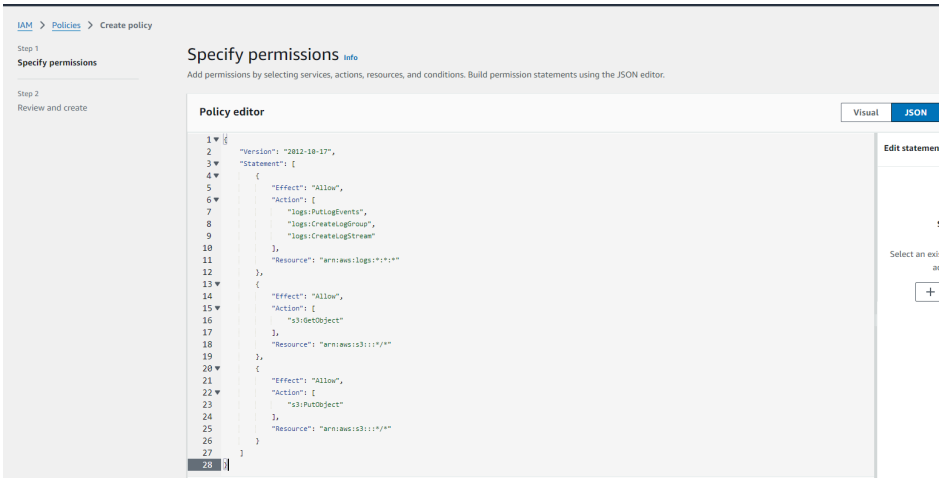
Step 2: Permissions Setup

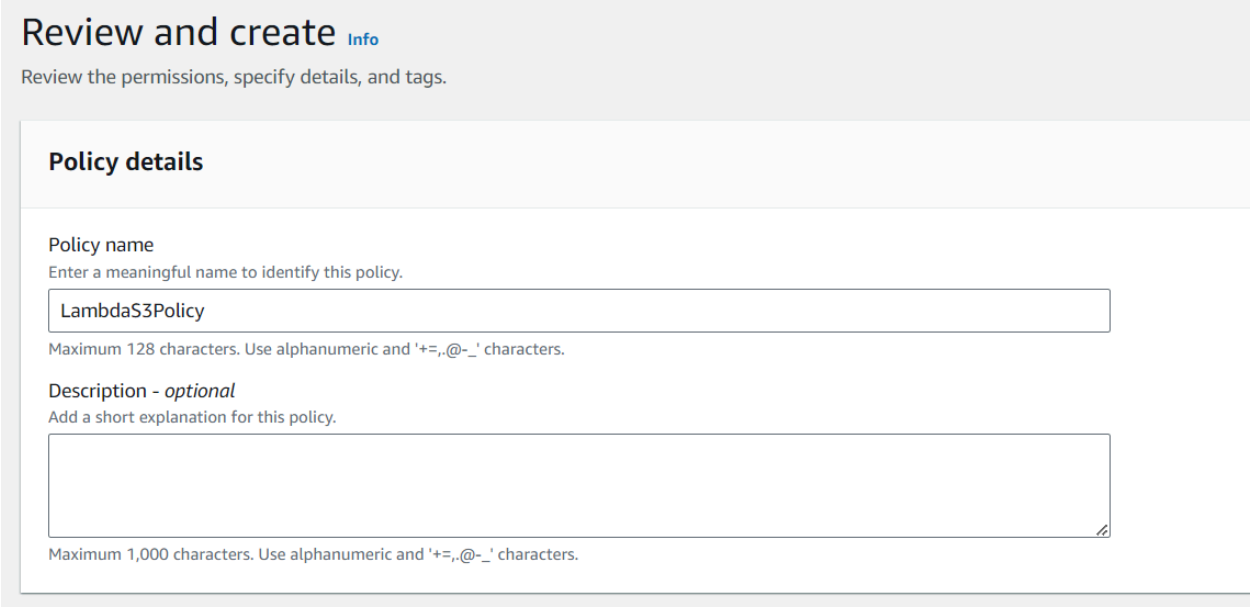
- **Create a Permissions Policy:**
  Using AWS Identity and Access Management (IAM), I created a custom policy named `LambdaS3Policy` to grant Lambda permissions to read from the source bucket and write to the destination bucket, along with logging to CloudWatch.

To create the policy (console)

1. Open the Policies page of the AWS Identity and Access Management (IAM) console.
2. Choose Create policy.
3. Choose the JSON tab, and then paste the following custom policy into the JSON editor.

Name:Atharv Sanjay Nikam Div:D15C Roll:36

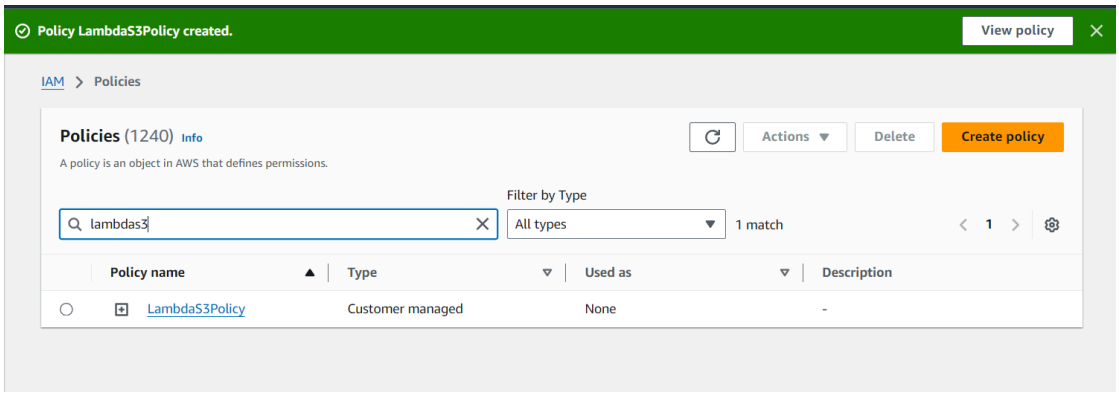


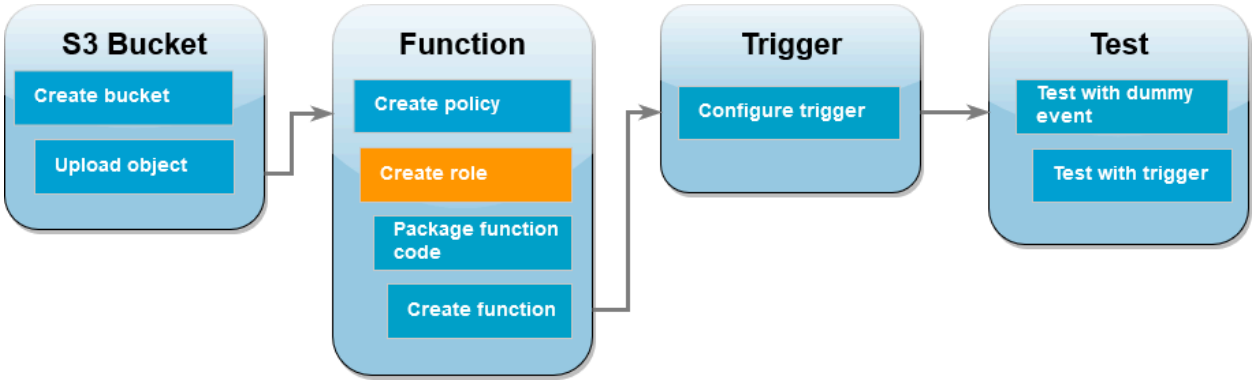


Code:

```
{
    "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:::*/*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::*/*"  } ]}
```



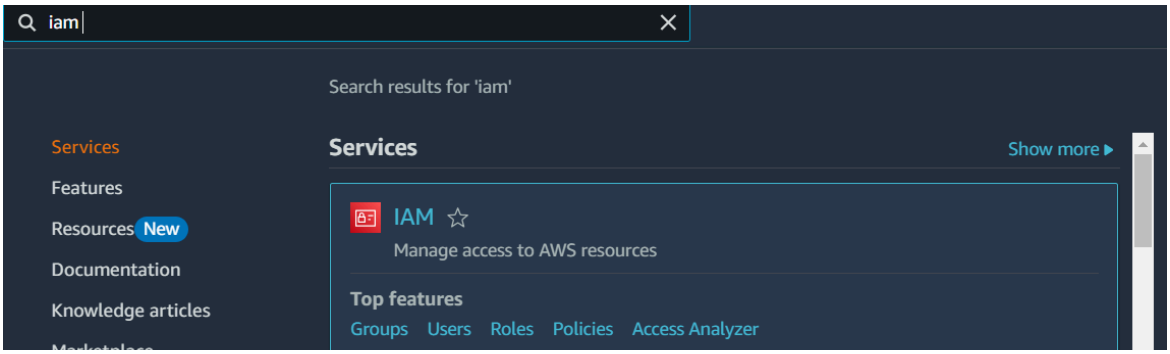


Creating an Execution Role and Attaching Permissions Policy

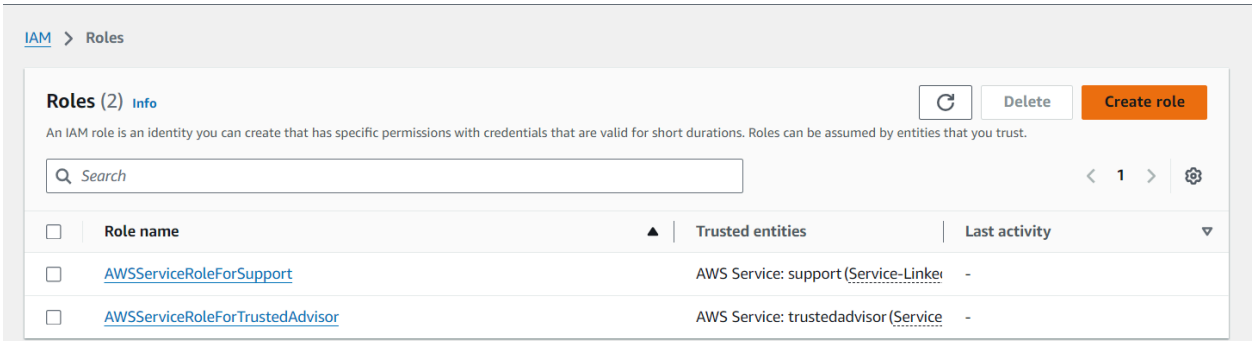
To enable the Lambda function to interact with the Amazon S3 buckets, it requires an **IAM execution role**. This role grants the necessary permissions for the function to read from the source bucket and write to the destination bucket, which are essential for the image resizing process.

Steps to Create an Execution Role and Attach a Permissions Policy (Using the AWS Console):

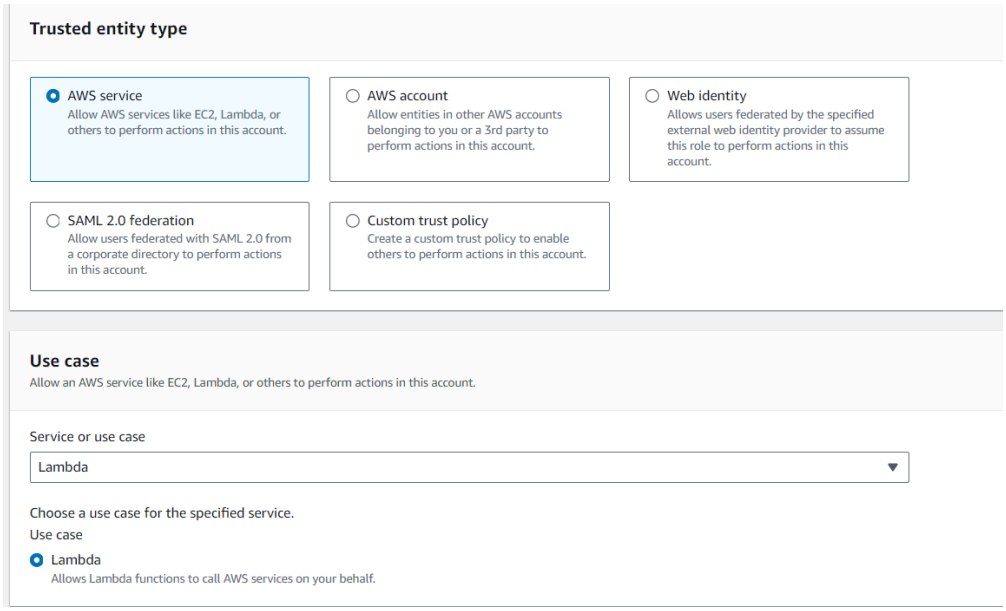
1. Open the **IAM Roles** page in the AWS Management Console.



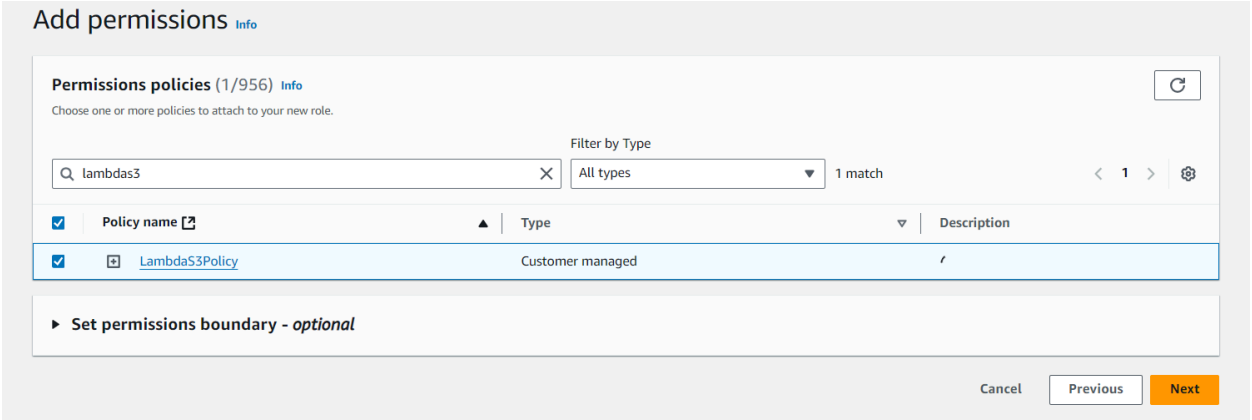
2. Click on **Create role**.



3. Under **Trusted entity type**, select **AWS service**.

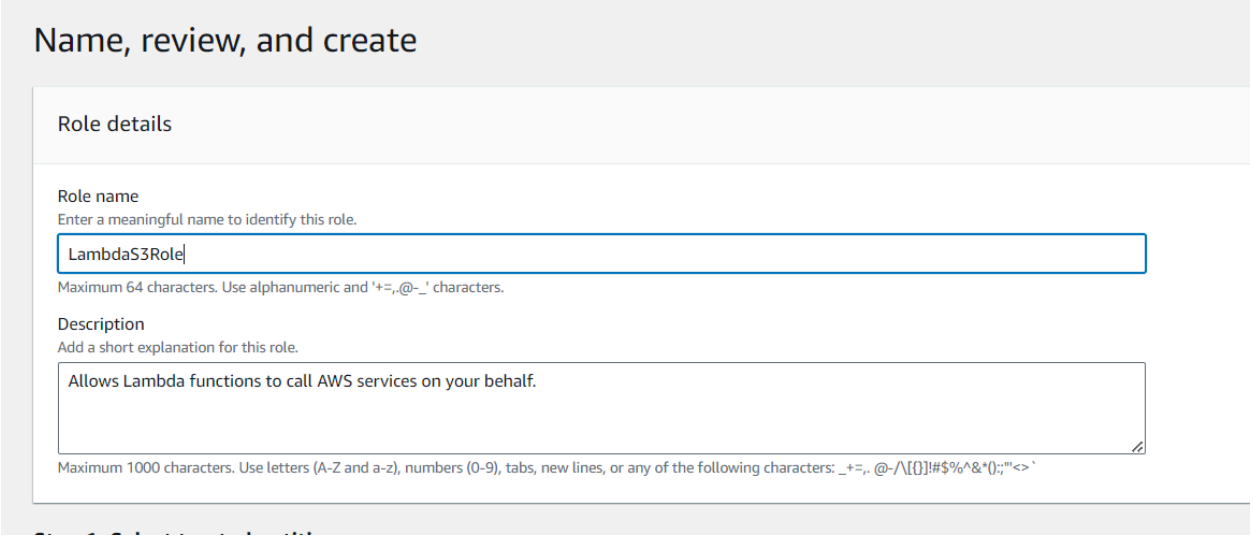


4. For the **Use case**, choose **Lambda**.
5. Click on **Next**.
6. In the **Policy search box**, type **LambdaS3Policy** (the name of the policy you created in the earlier step).
7. From the search results, select the check box next to **LambdaS3Policy** to attach it to the role.

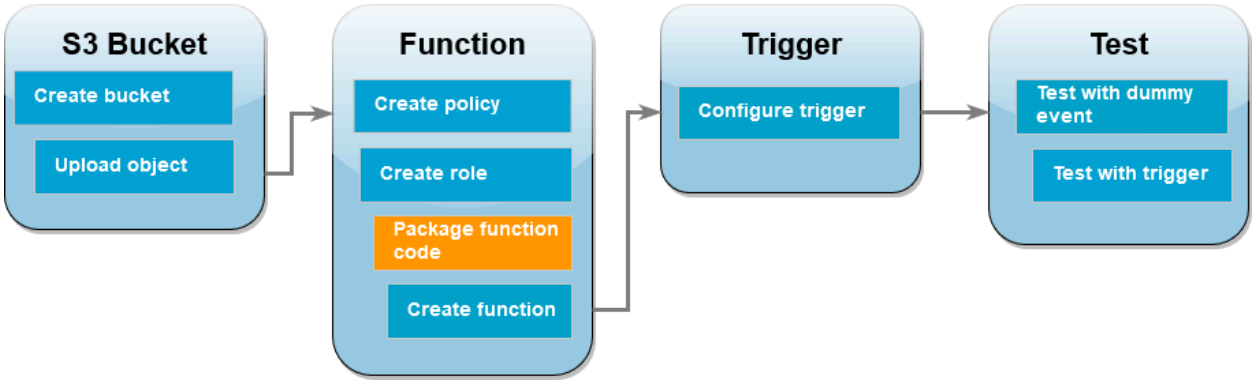


8. Click **Next** to proceed.

9. Under **Role details**, enter **LambdaS3Role** as the **Role name**.



10. Finally, click **Create role**.



Steps to Create the Deployment Package (Python)

Save the Lambda Function Code:

- Save the provided Python code as `lambda_function.py`.

  Code:

```
import boto3
import os
import sys
import uuid
from urllib.parse import unquote_plus
from PIL import Image
import PIL.Image


s3_client = boto3.client('s3')


def resize_image(image_path, resized_path):
  with Image.open(image_path) as image:
```

```
    image.thumbnail(tuple(x / 2 for x in image.size))
    image.save(resized_path)


def lambda_handler(event, context):
  for record in event['Records']:
    bucket = record['s3']['bucket']['name']
    key = unquote_plus(record['s3']['object']['key'])
    tmpkey = key.replace('/', '')
    download_path = '/tmp/{}{}'.format(uuid.uuid4(), tmpkey)
    upload_path = '/tmp/resized-{}'.format(tmpkey)
    s3_client.download_file(bucket, key, download_path)
    resize_image(download_path, upload_path)
    s3_client.upload_file(upload_path, '{}-resized'.format(bucket),
'resized-{}'.format(key))
```

```python
import boto3
import os
import sys
import uuid
from urllib.parse import unquote_plus
from PIL import Image
import PIL.Image

s3_client = boto3.client('s3')

def resize_image(image_path, resized_path):
  with Image.open(image_path) as image:
    image.thumbnail(tuple(x / 2 for x in image.size))
    image.save(resized_path)

def lambda_handler(event, context):
  for record in event['Records']:
    bucket = record['s3']['bucket']['name']
    key = unquote_plus(record['s3']['object']['key'])
    tmpkey = key.replace('/', '')
    download_path = '/tmp/{}{}'.format(uuid.uuid4(), tmpkey)
    upload_path = '/tmp/resized-{}'.format(tmpkey)
    s3_client.download_file(bucket, key, download_path)
    resize_image(download_path, upload_path)
    s3_client.upload_file(upload_path, '{}-resized'.format(bucket), 'resized-{}'.format(key))
```

Install Dependencies:

- In the directory with `lambda_function.py`, create a folder named `package`

  mkdir package

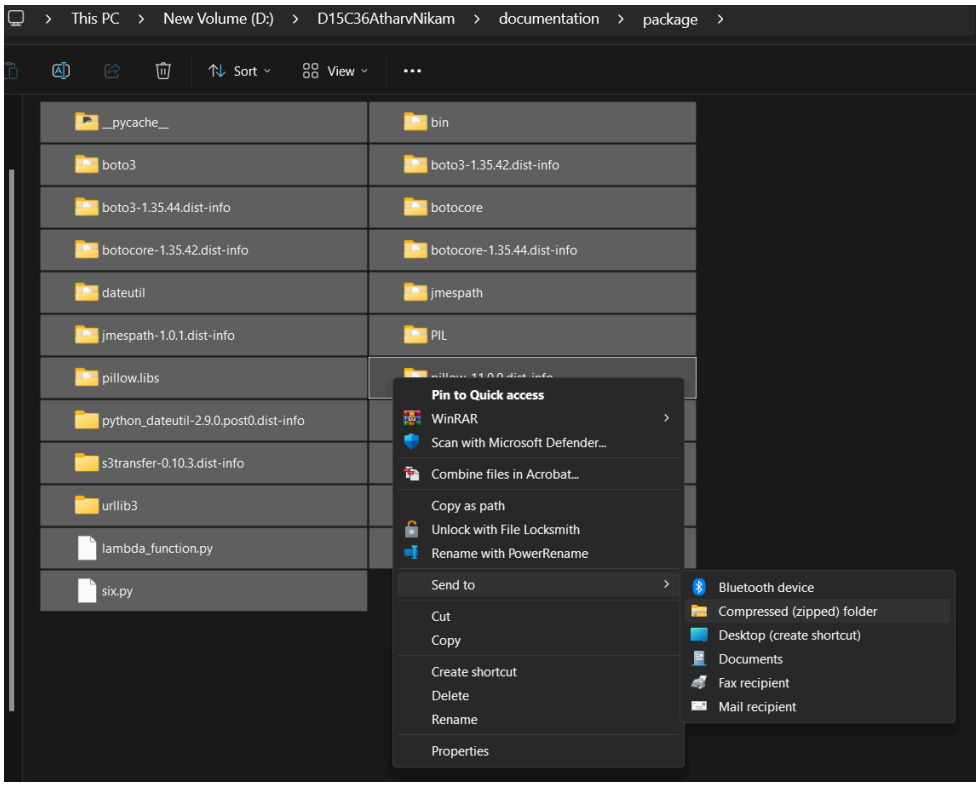
  Install Pillow and Boto3 libraries inside the `package` folder using the command:

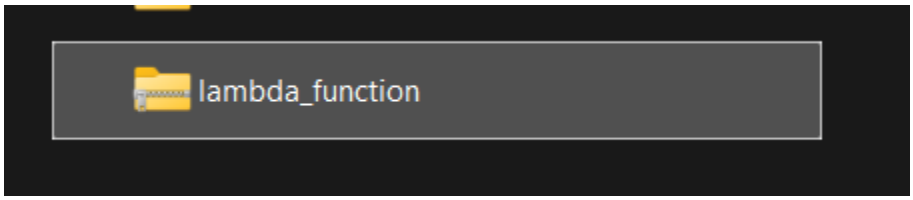
  pip install --platform manylinux2014_x86_64 --target=package --implementation cp --python-version 3.12 --only-binary=:all: --upgrade pillow boto3

```
PS D:\D15C36AtharvNikam\documentation> mkdir package1


    Directory: D:\D15C36AtharvNikam\documentation


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        19-10-2024  10:00 PM                package1


PS D:\D15C36AtharvNikam\documentation> pip install `
>> --platform manylinux2014_x86_64 `
>> --target=package `
>> --implementation cp `
>> --python-version 3.12 `
>> --only-binary=:all: `
>> --upgrade `
>> pillow boto3
Collecting pillow
  Using cached pillow-11.0.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.1 kB)
Collecting boto3
  Downloading boto3-1.35.44-py3-none-any.whl.metadata (6.7 kB)
```

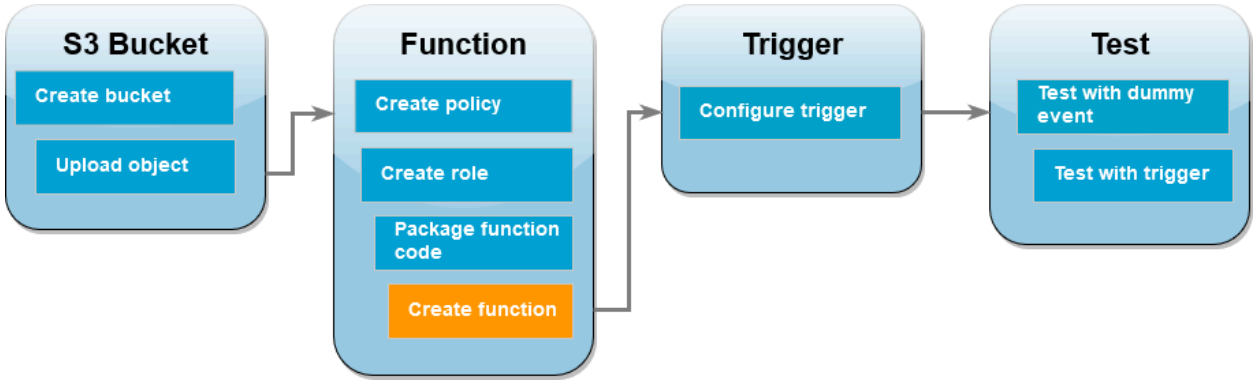
Package the Application:

Use any ZIP tool to create `lambda_function.zip` with `lambda_function.py` and the `package` folder contents.





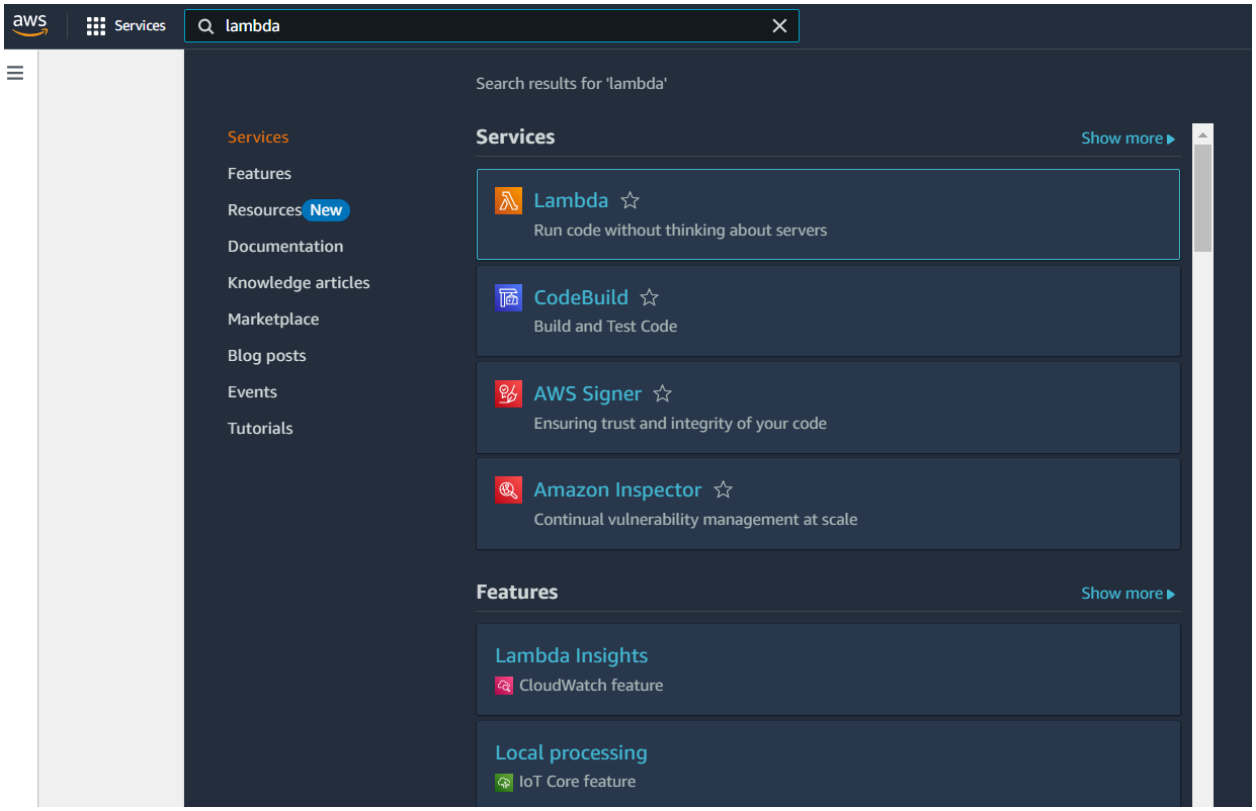
Name:Atharv Sanjay Nikam Div:D15C Roll:36



Steps to Create the Lambda Function (Console)

Access the Lambda Console:

- Open the **Functions** page of the AWS Lambda console.
- Ensure you are in the same AWS Region where you created your S3 bucket (you can change the region from the drop-down menu).



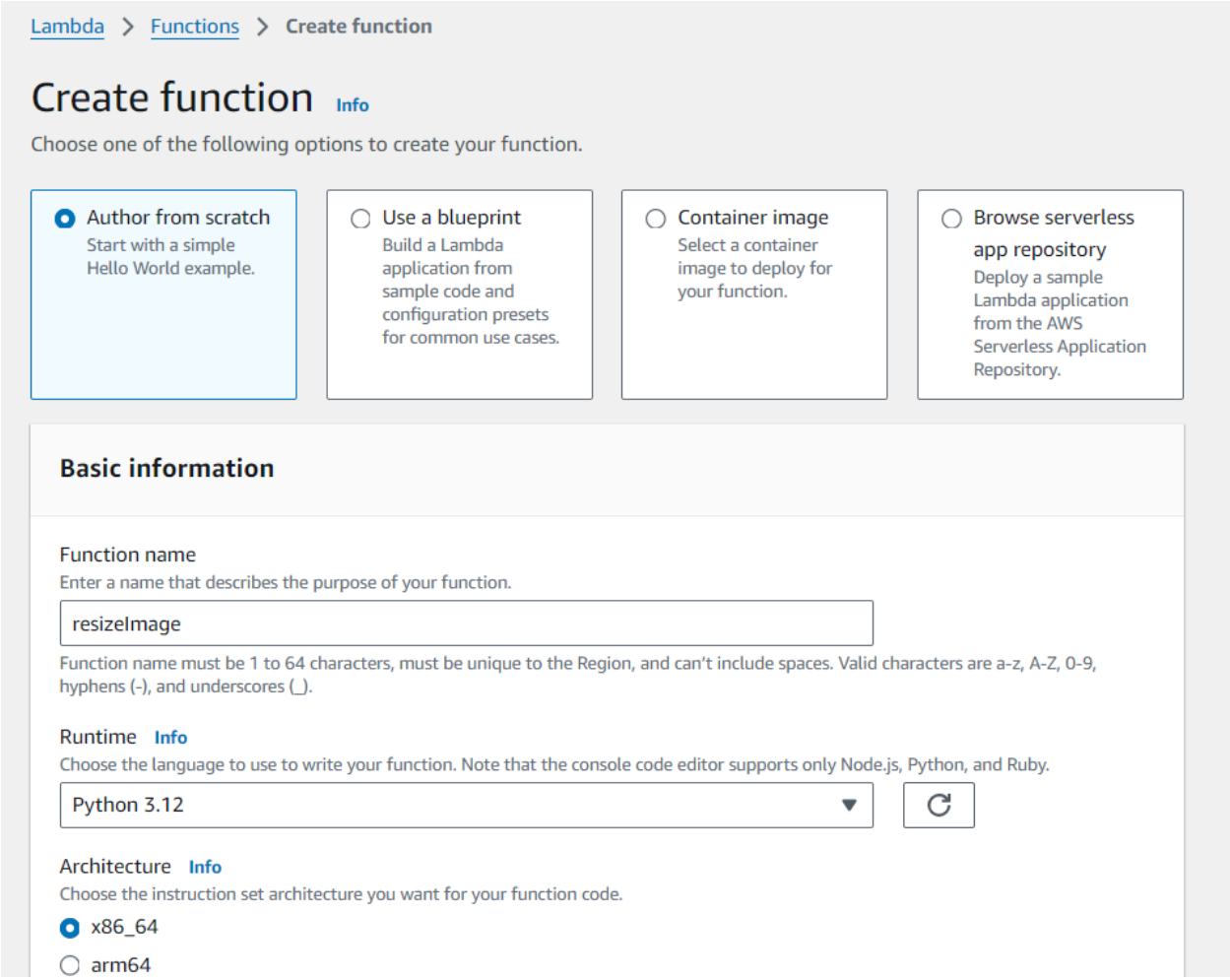
Create a New Function:

Choose **Create function**.

Select **Author from scratch**.

Fill in the **Basic information**:

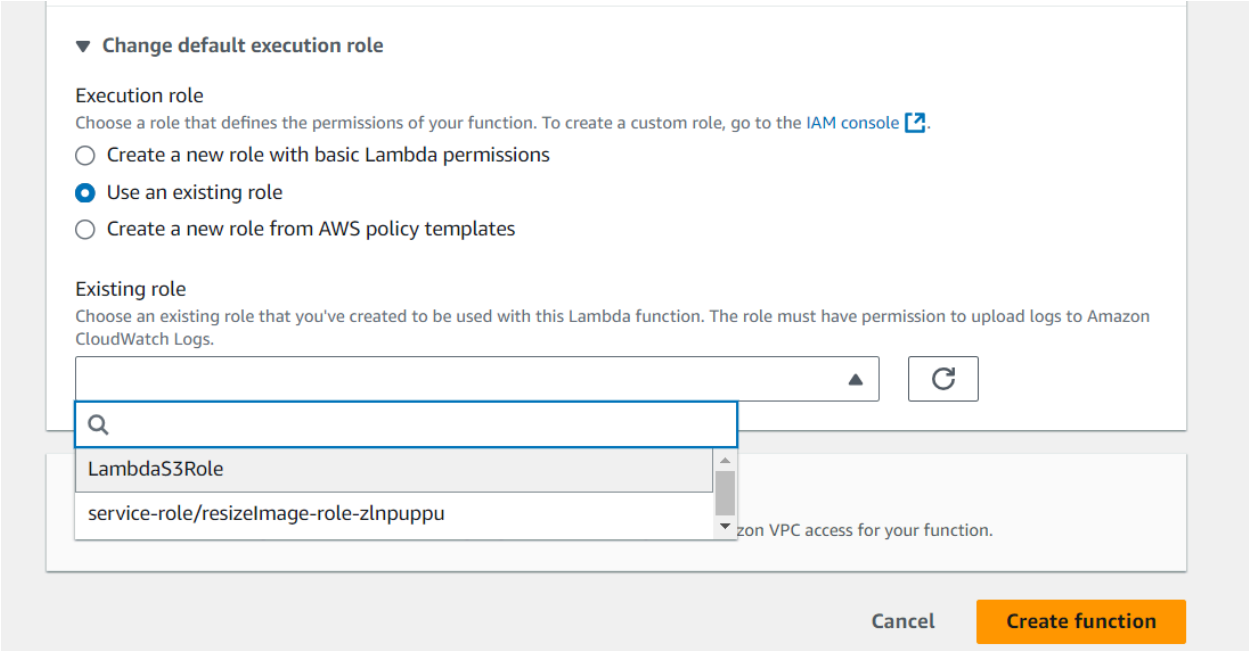
- **Function name**: `resizeImage`
- **Runtime**: Select either **Python 3.12**
- **Architecture**: Choose **x86_64**.

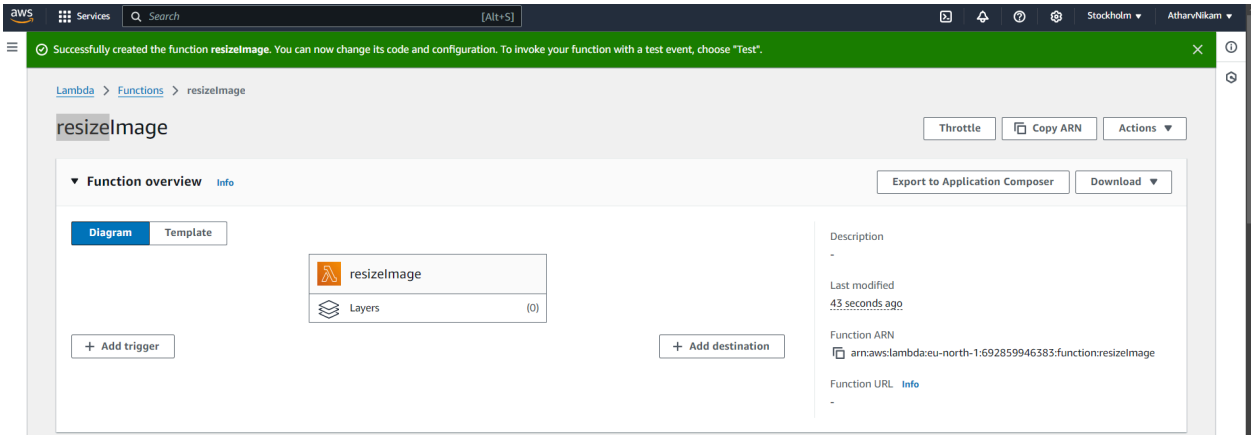


In the **Change default execution role** section:

- Expand the tab and select **Use an existing role**.
- Choose the `LambdaS3Role` you created earlier.

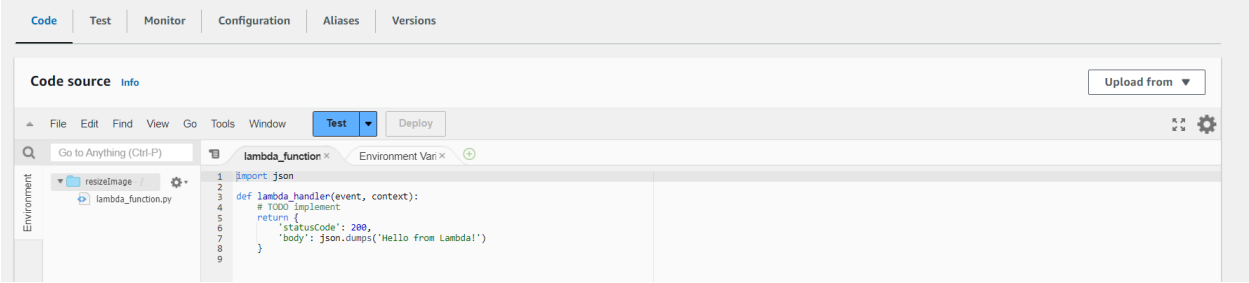
Click **Create function**.

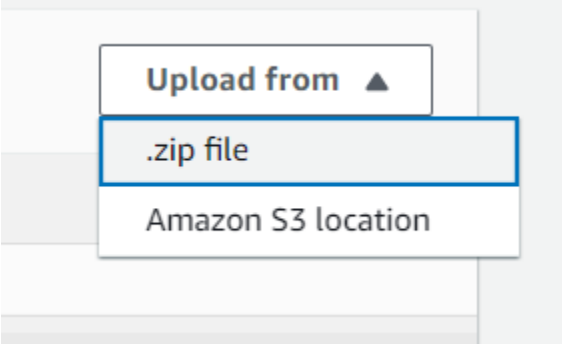


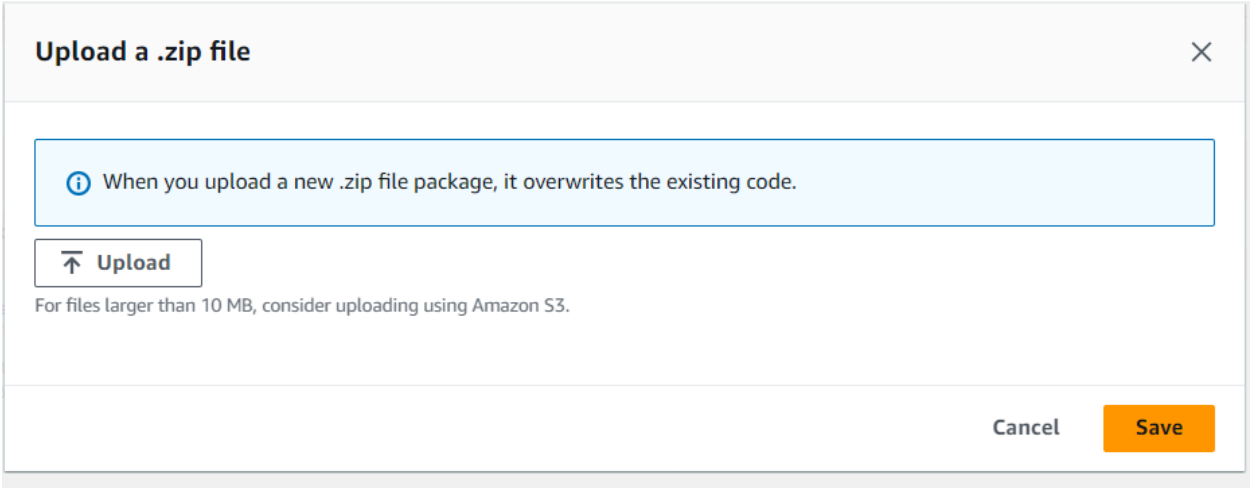


Upload Function Code:

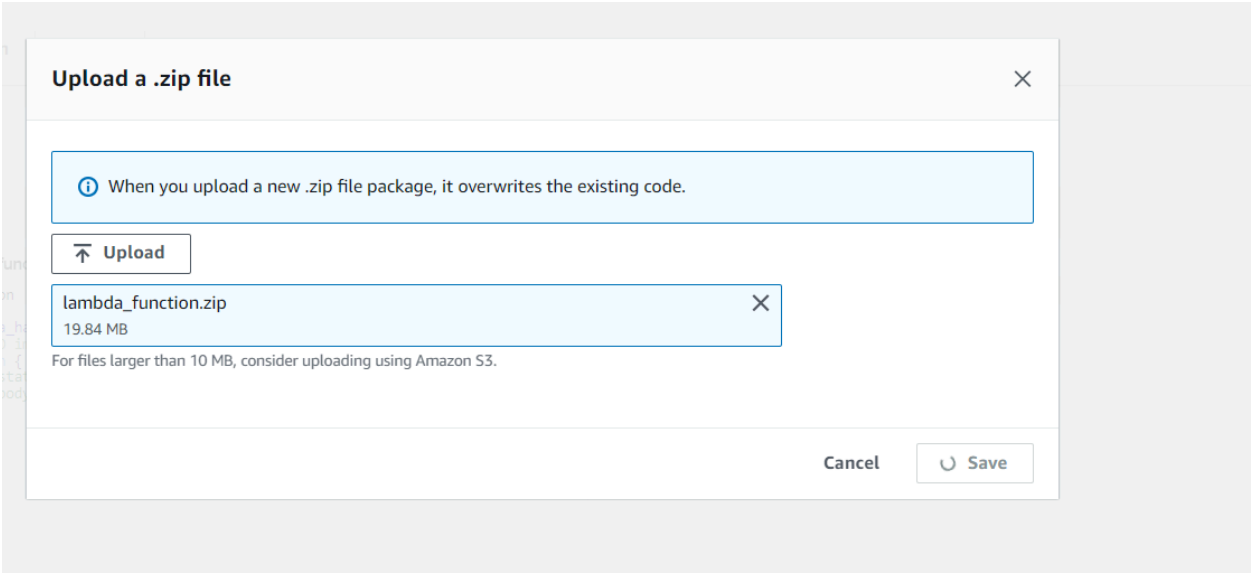
- In the **Code source** pane, select **Upload from**.
- Choose **.zip file**.
- Select your `.zip` file and click **Open**.
- Click **Save**.

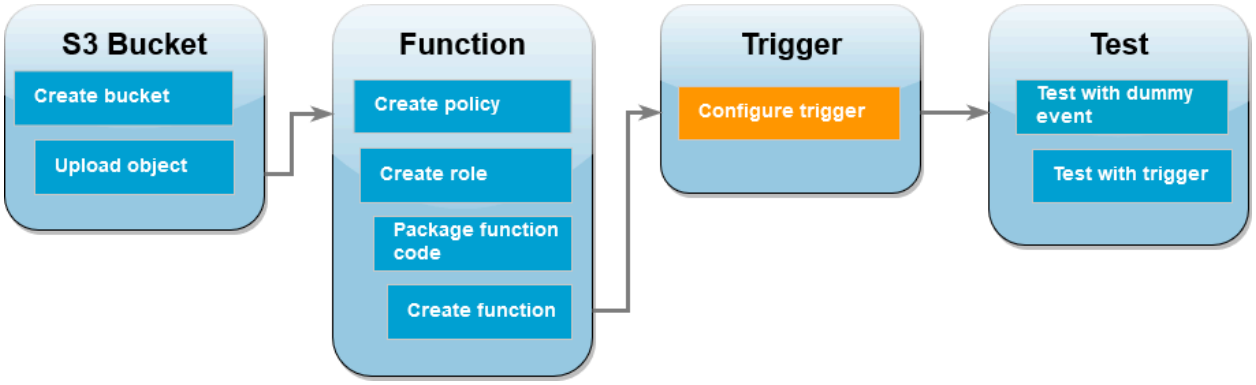






Name:Atharv Sanjay Nikam Div:D15C Roll:36





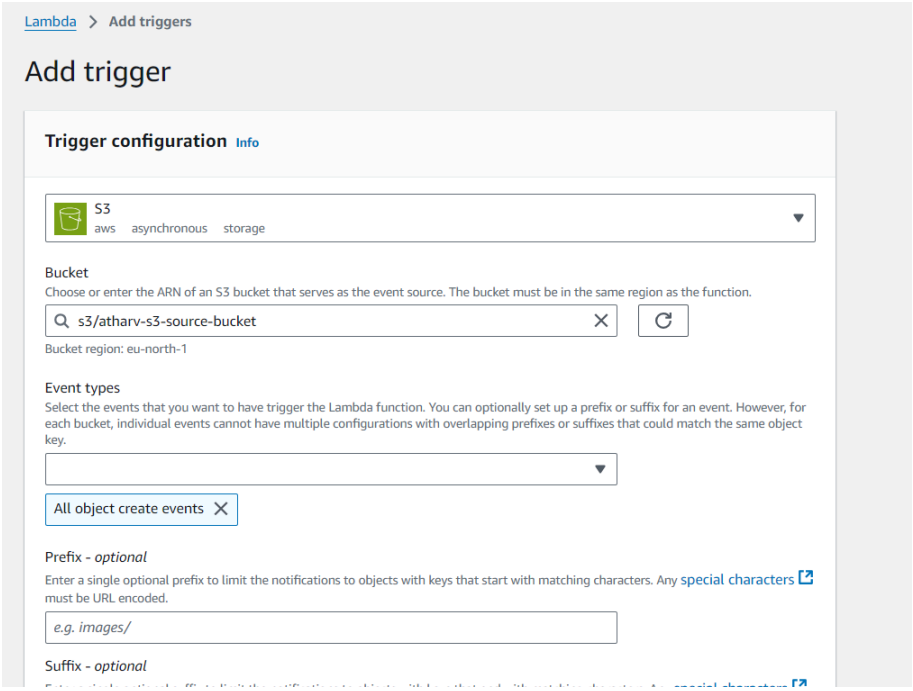
Steps to Configure the Amazon S3 Trigger (Console)

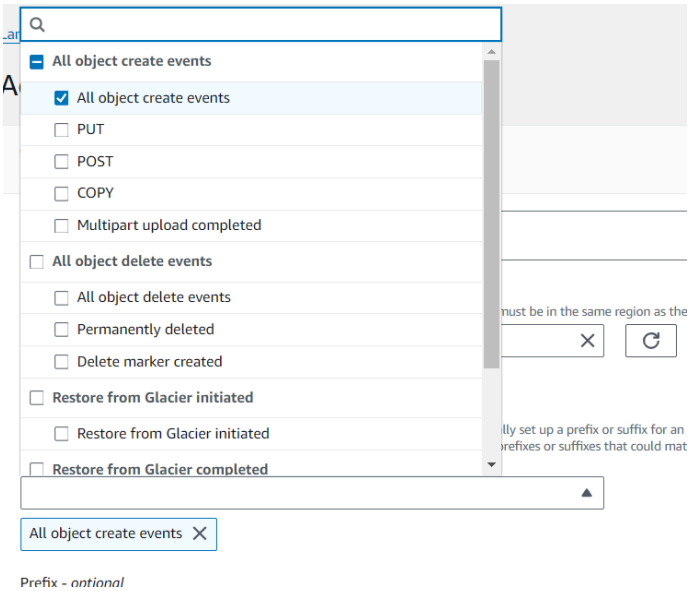
‘Access the Lambda Function:

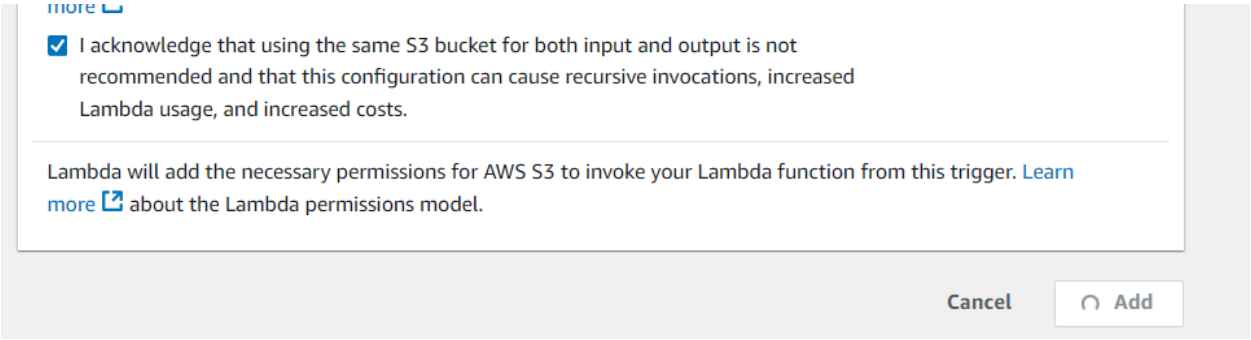
- Open the **Functions** page in the AWS Lambda console.
- Select your function:resizeImage

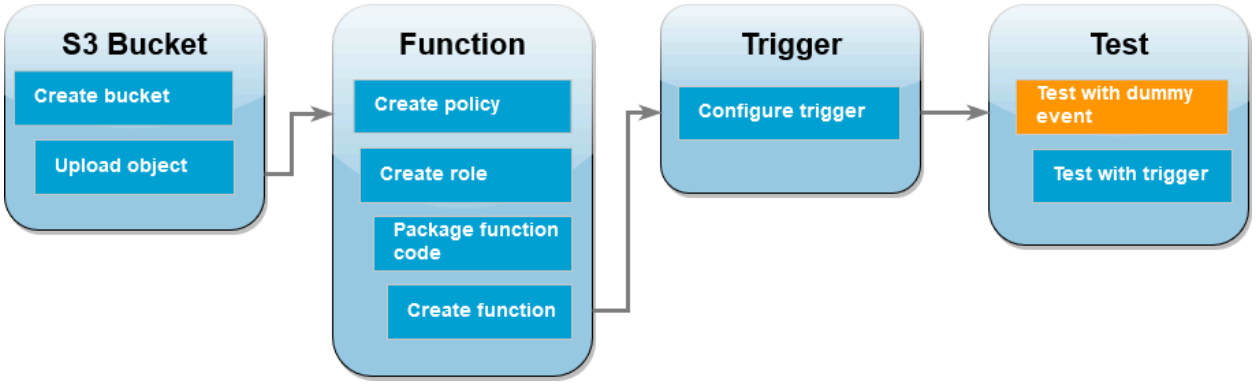
Add S3 Trigger:

- Click on **Add trigger**.
- Choose **S3** as the trigger type.
- Under **Bucket**, select your source bucket.
- Under **Event types**, select **All object create events**.
- Check the box to acknowledge the warning about recursive invocation.



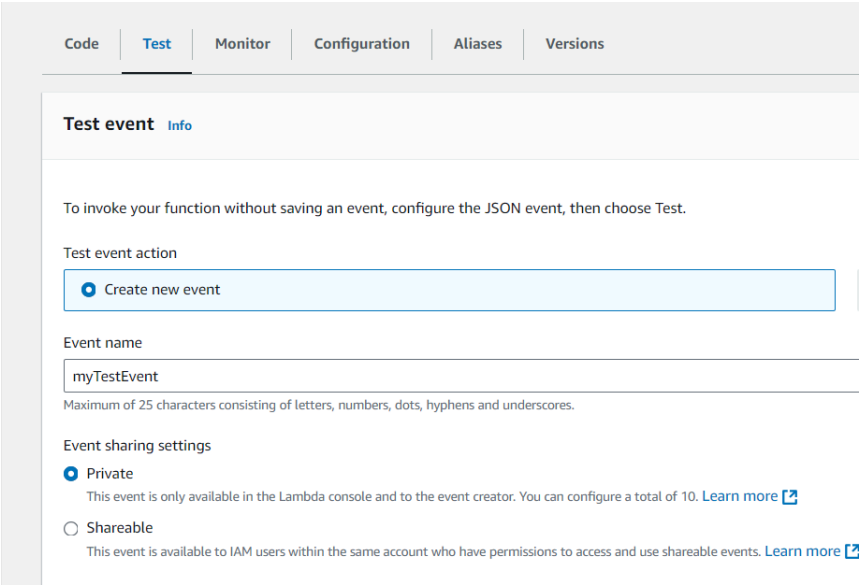

All object create events
All object create events
PUT
POST
COPY
Multipart upload completed
All object delete events
All object delete events
Permanently deleted
Delete marker created
Restore from Glacier initiated
Restore from Glacier initiated
Restore from Glacier completed
All object create events
Prefix - optional


I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.
Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.
Cancel
Add



Steps to Test Your Lambda Function with a Dummy Event (Console)

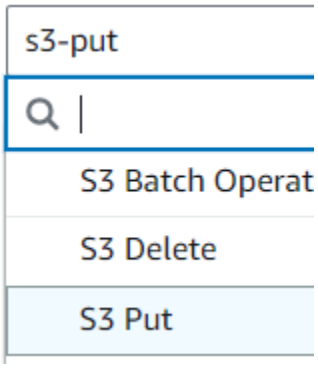
1. **Access Your Lambda Function:**
   - Open the **Functions** page in the AWS Lambda console.
   - Select your function: **CreateThumbnail**.



Create a Test Event:

- Go to the **Test** tab.
- In the **Test event** pane, select **Create new event**.

- **Event name:** Enter `myTestEvent`.
- **Template:** Select **S3 Put**.



Customize Event Parameters:


```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "eu-north-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": "amzn-s3-sourcr-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
```

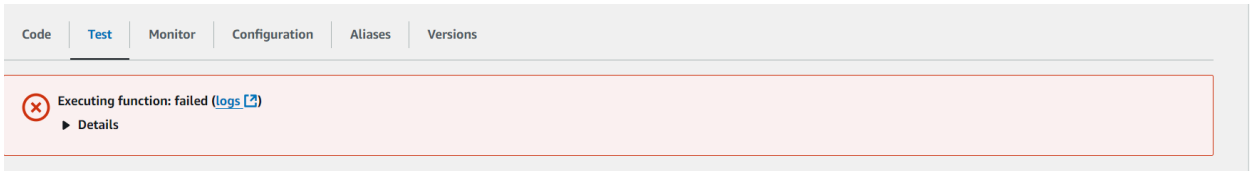
```
      "object": {
        "key": "football",
        "size": 1024,
        "eTag": "0123456789abcdef0123456789abcdef",
        "sequencer": "0A1B2C3D4E5F678901"
      }
    }
  }
 ]
}
```

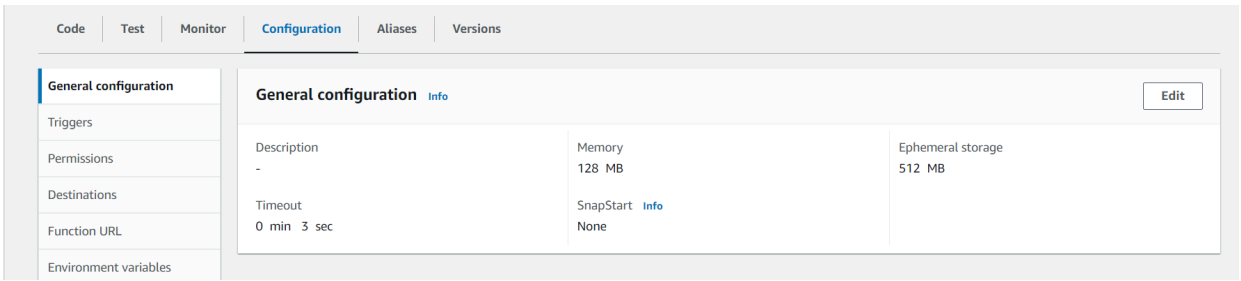
Save and Test:

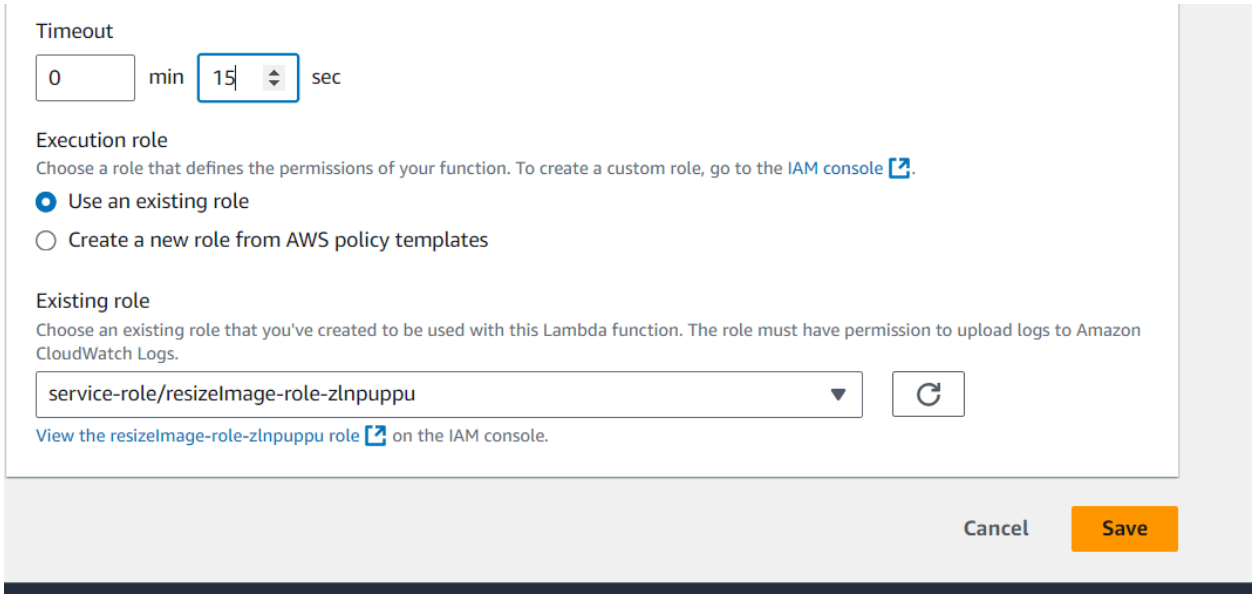
- Click **Save**.
- In the **Test event** pane, choose **Test**.

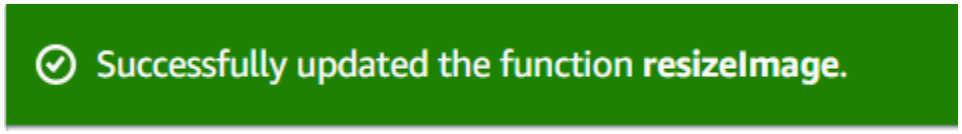
Verify Resized Image:

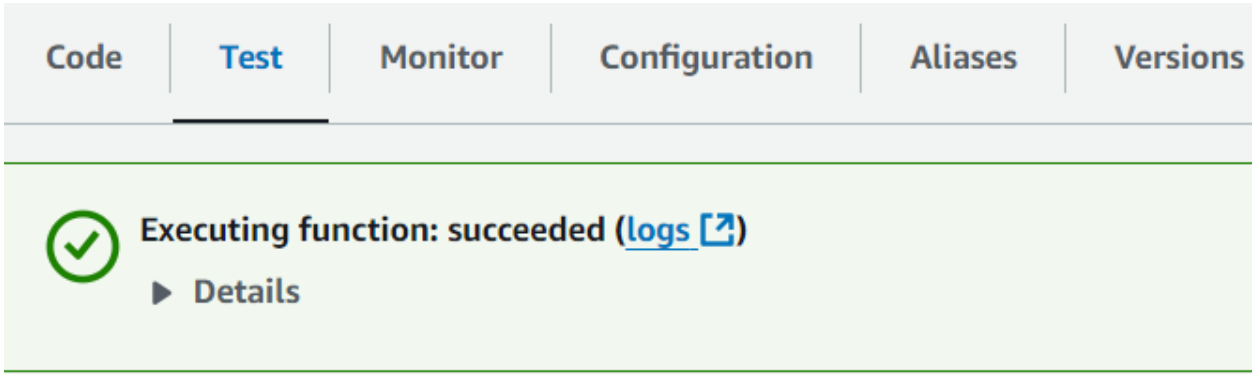
- Open the **Buckets** page in the Amazon S3 console.
- Select your target bucket and check the **Objects** pane for the resized image.



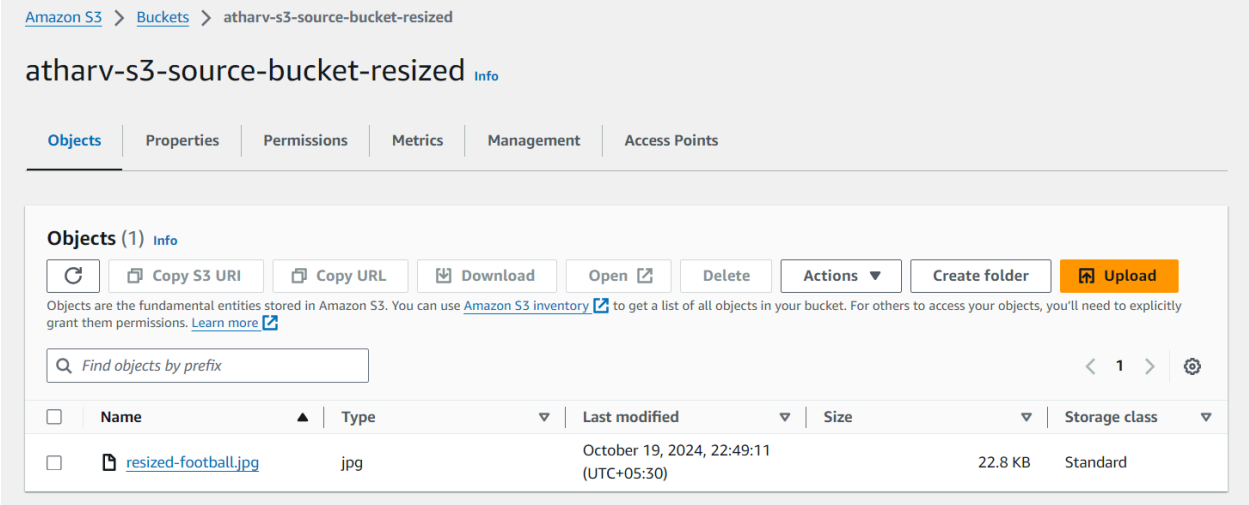


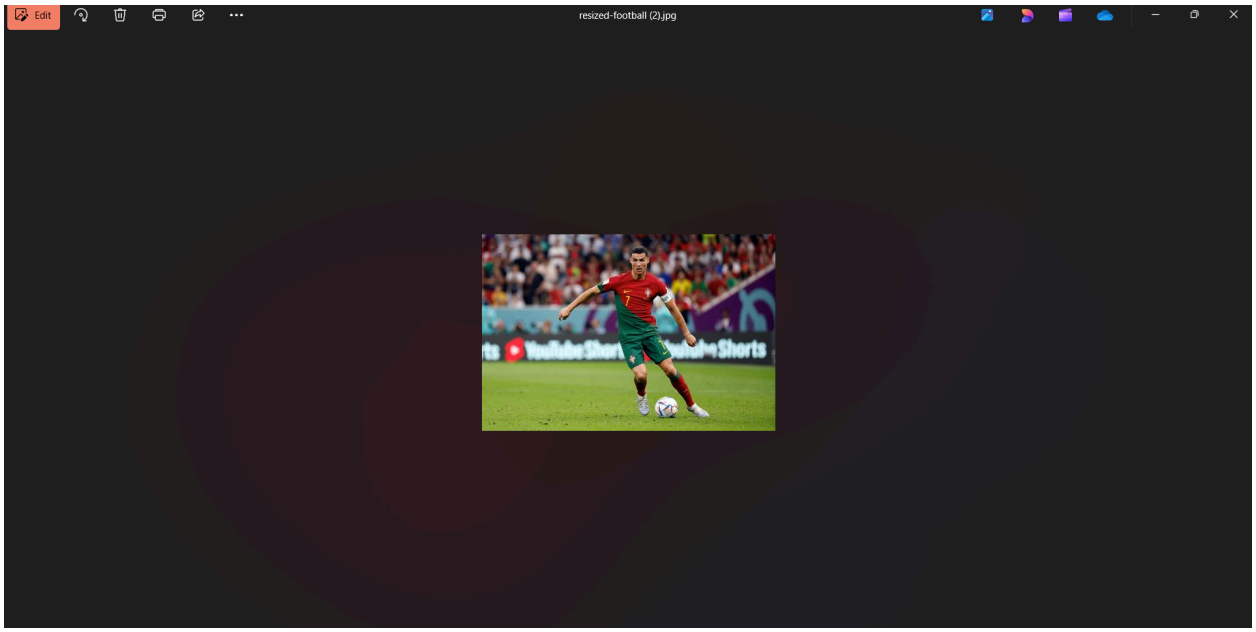

Timeout
0
min
15
sec
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/resizeImage-role-zlnpuppu
View the resizeImage-role-zlnpuppu role on the IAM console.
Cancel
Save

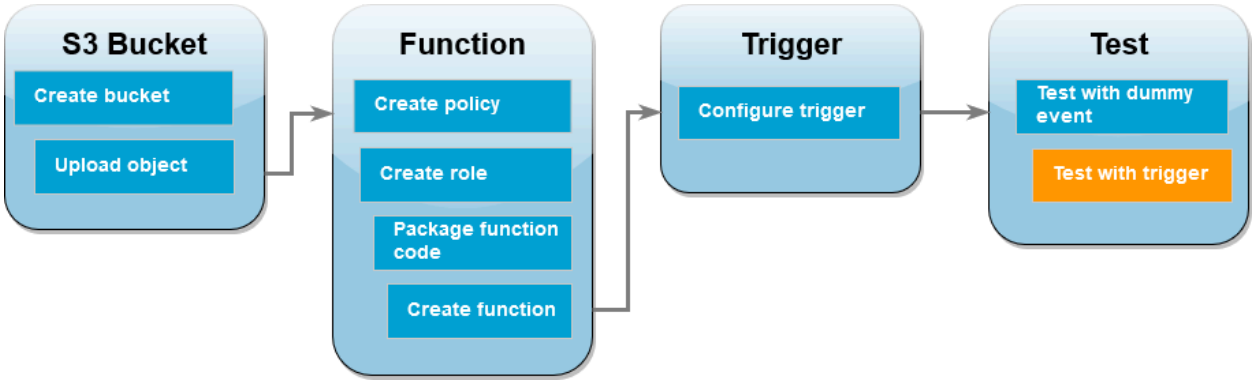

Successfully updated the function resizeImage.


Code
Test
Monitor
Configuration
Aliases
Versions
Executing function: succeeded (logs)
Details

Name:Atharv Sanjay Nikam Div:D15C Roll:36

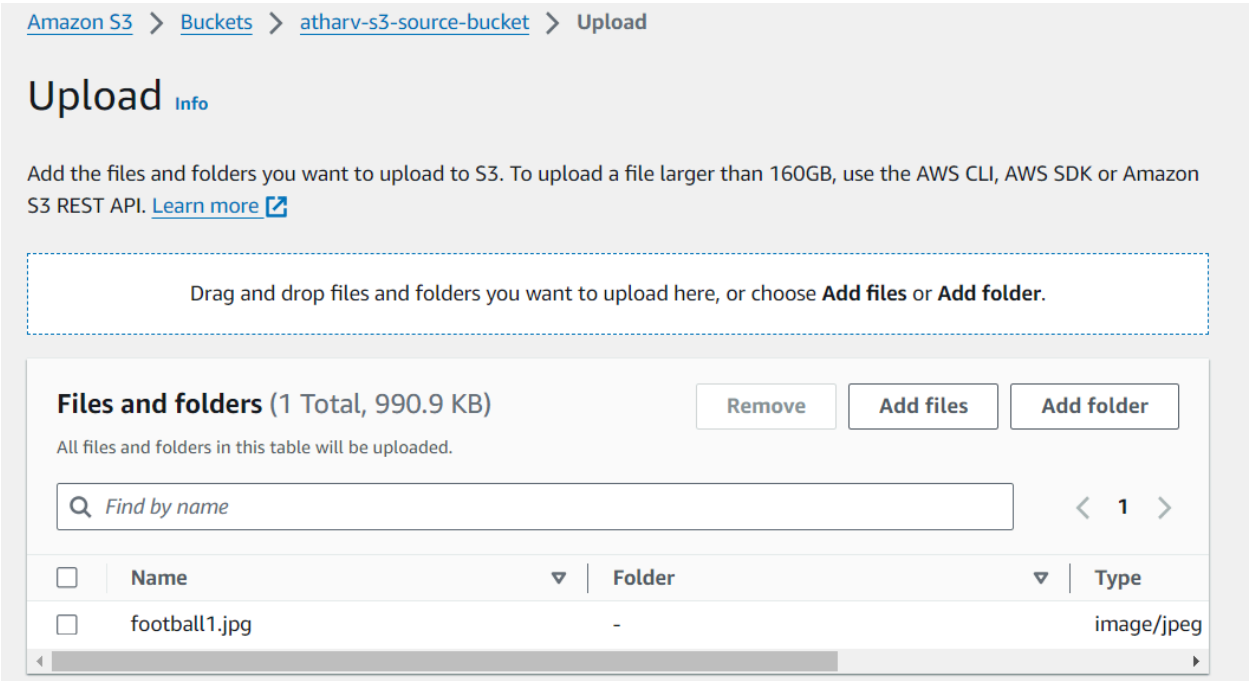




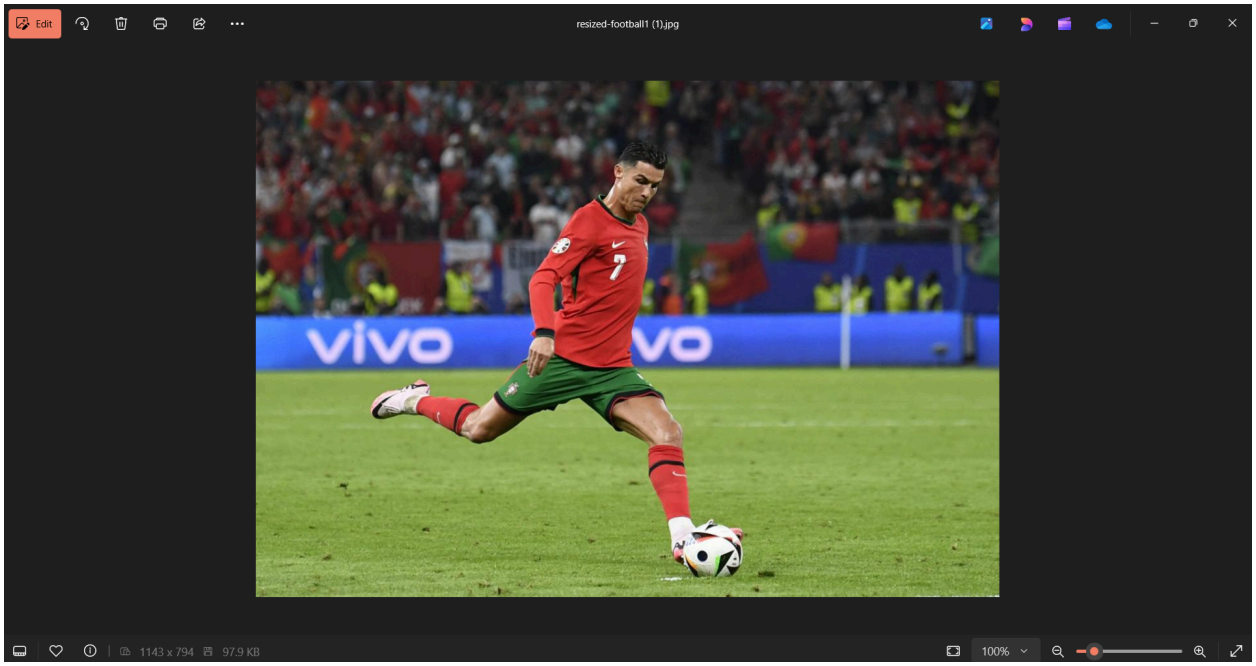


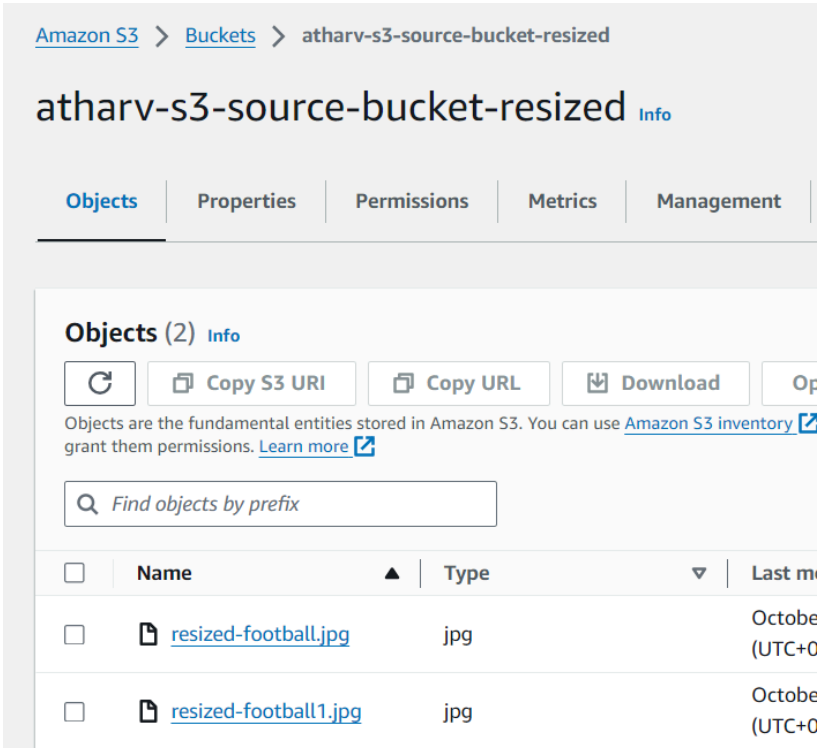
Steps to Test Your Lambda Function Using the Amazon S3 Trigger (Console)

1. **Upload an Image to Your Source Bucket:**
   - Open the **Buckets** page in the Amazon S3 console.
   - Select your **source bucket**.
   - Click on **Upload**.
   - Choose **Add files** and select the image file (any `.jpg` or `.png`).
   - Click **Open**, then choose **Upload**.



Name:Atharv Sanjay Nikam Div:D15C Roll:36





Name:Atharv Sanjay Nikam Div:D15C Roll:36