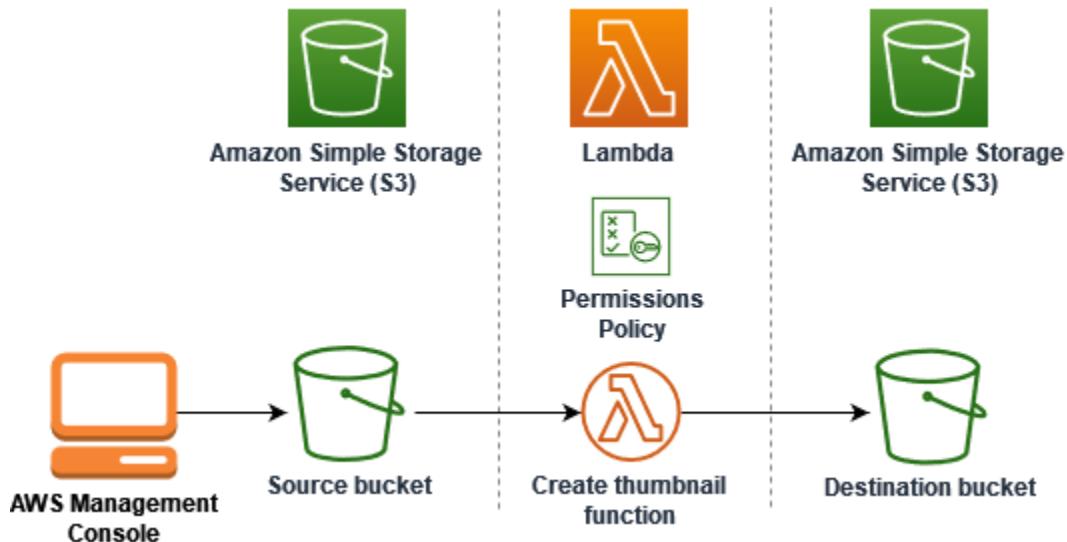


ADVANCE DEVOPS PRACTICAL EXAM

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.

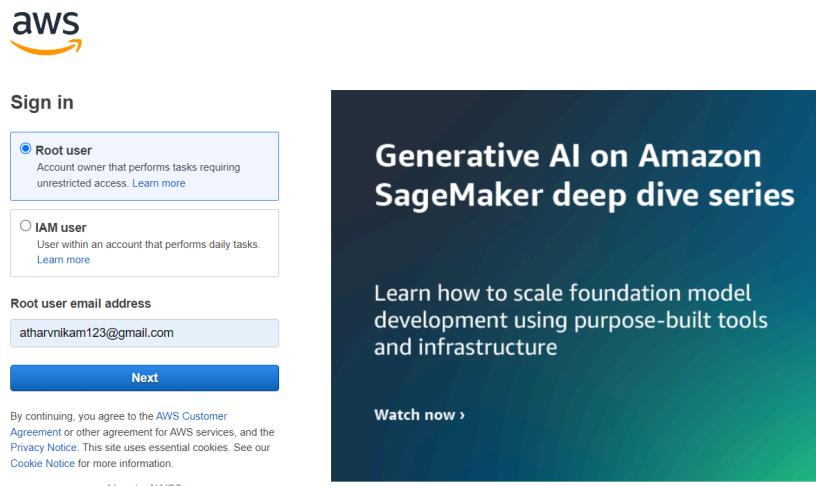
Furthermore, the solution is highly scalable, as AWS Lambda automatically handles the scaling of functions in response to increased traffic, without requiring any manual intervention. It is also cost-effective, as you only pay for the compute time you use, making it ideal for applications with fluctuating 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*.





Root user sign in

Email: atharvnikam123@gmail.com

Password

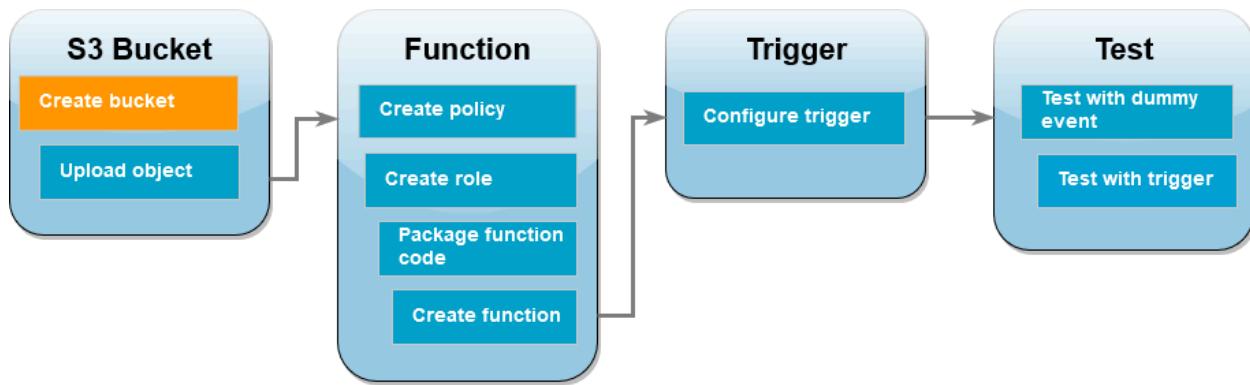
[Forgot password?](#)

.....

[Sign in](#)

[Sign in to a different account](#)

[Create a new AWS account](#)



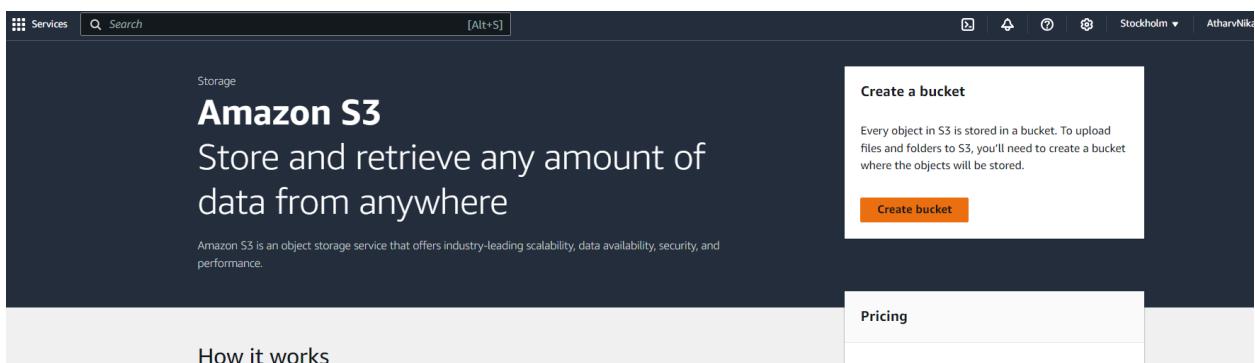
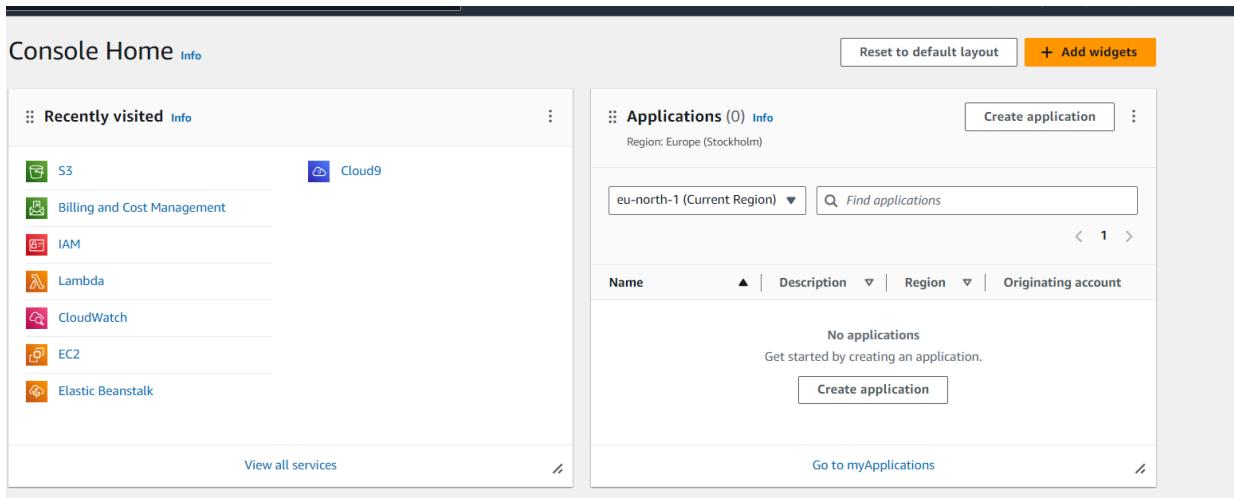
Create Source and Destination Buckets:

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

Example:

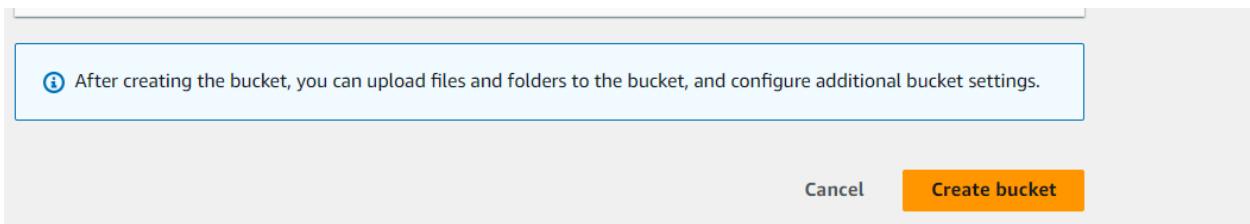
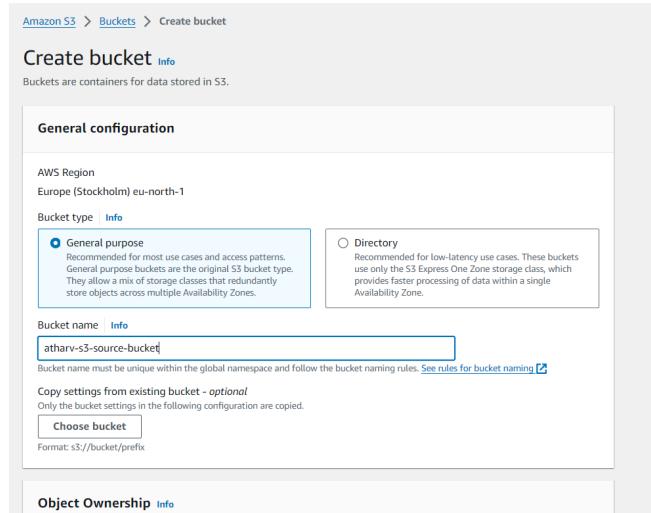
- Source Bucket: `atharv-s3-source-bucket`
- Destination Bucket: `atharv-s3-source-bucket-resized`

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**.

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
Europe (Stockholm) eu-north-1

Bucket type Info

General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

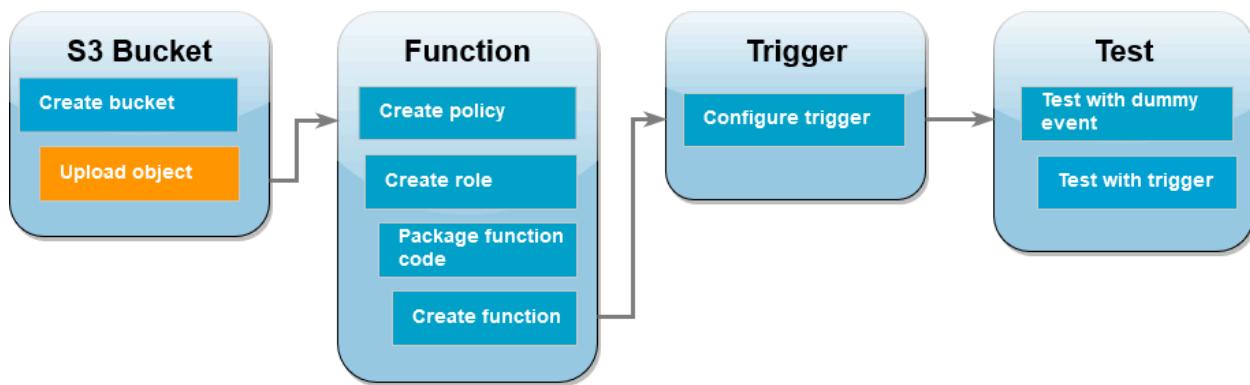
Format: s3://bucket/prefix

Object Ownership Info

► Advanced settings

i After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)



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**.

Amazon S3 > Buckets > atharv-s3-source-bucket > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files or Add folder.

Files and folders (1 Total, 56.1 KB)		Remove	Add files	Add folder
All files and folders in this table will be uploaded.				
<input type="text"/> <small>Find by name</small>		<	1	>
<input type="checkbox"/>	Name	Folder		
<input type="checkbox"/>	football.jpg	-		

The screenshot shows the AWS S3 console after a file has been uploaded. At the top, a green header bar indicates "Upload succeeded" with a link to "View details below". Below this, the title "Upload: status" is displayed, along with a "Close" button. A message box states: "The information below will no longer be available after you navigate away from this page." The main area is titled "Summary" and contains three sections: "Destination" (s3://atharv-s3-source-bucket), "Succeeded" (1 file, 56.1 KB (100.0%)), and "Failed" (0 files, 0 B (0%)). Below the summary, there are tabs for "Files and folders" (selected) and "Configuration". Under "Files and folders", a table lists one item: "football.jpg" (image/jpeg, 56.1 KB, Status: Succeeded). There is also a "Find by name" search bar and navigation arrows.

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.

```

1 * {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "logs:PutLogEvents",
8                 "logs:CreateLogGroup",
9                 "logs:CreateLogStream"
10            ],
11            "Resource": "arn:aws:logs:!*:*"
12        },
13        {
14            "Effect": "Allow",
15            "Action": [
16                "s3:GetObject"
17            ],
18            "Resource": "arn:aws:s3:::/*"
19        },
20        {
21            "Effect": "Allow",
22            "Action": [
23                "s3:PutObject"
24            ],
25            "Resource": "arn:aws:s3:::/*"
26        }
27    ]
28 }

```

Review and create Info

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+,-,@-' characters.

Description - optional

Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+,-,@-' characters.

Code:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs>CreateLogGroup",

```

Name:Atharv Sanjay Nikam

Div:D15C

Roll:36

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

⌚ Policy LambdaS3Policy created.

[View policy](#) [X](#)

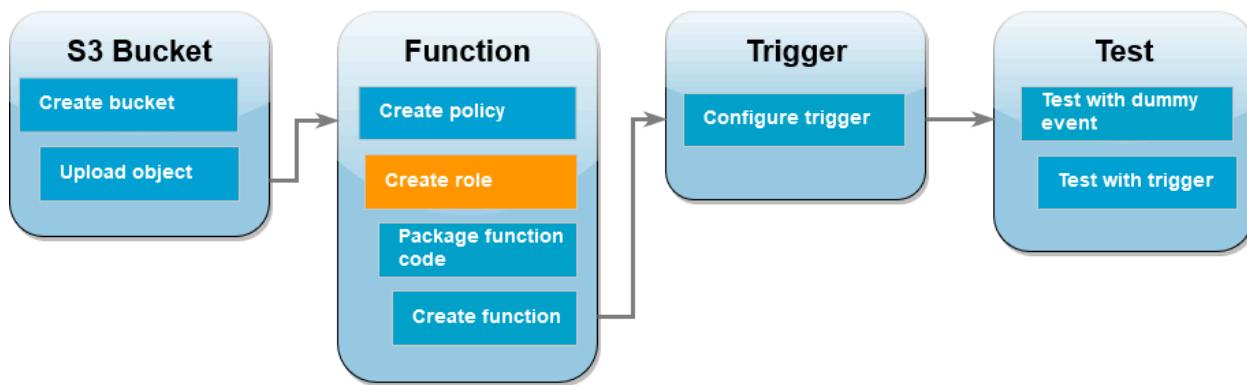
IAM > Policies

Policies (1240) Info

A policy is an object in AWS that defines permissions.

Filter by Type

Policy name	Type	Used as	Description
LambdaS3Policy	Customer managed	None	-

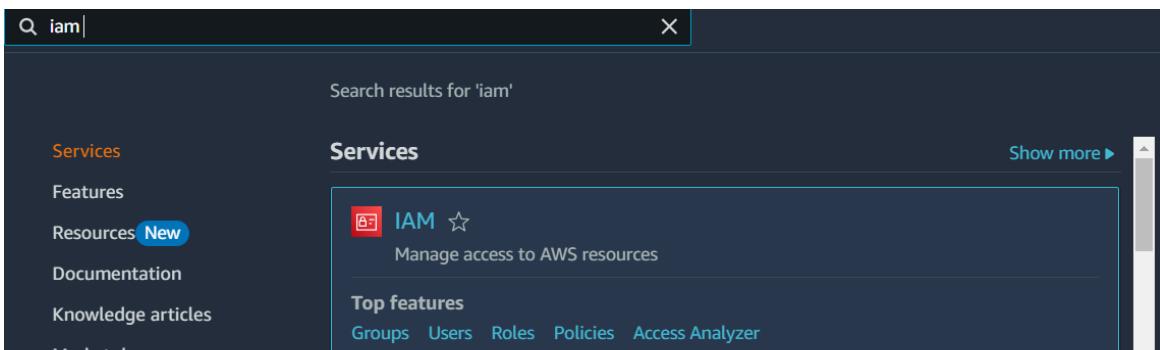


Creating an Execution Role and Attaching Permissions Policy

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**.

The screenshot shows the 'Roles' page in the IAM section of the AWS Management Console. The URL is 'IAM > Roles'. The page title is 'Roles (2) Info'. It includes a search bar and buttons for 'Delete' and 'Create role'. A descriptive text states: 'An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.' Below this, there is a table with two rows of data. The columns are 'Role name', 'Trusted entities', and 'Last activity'. The first row shows 'AWSServiceRoleForSupport' with 'AWS Service: support (Service-Linker)' in the Trusted entities column and '-' in the Last activity column. The second row shows 'AWSServiceRoleForTrustedAdvisor' with 'AWS Service: trustedadvisor (Service-Linker)' in the Trusted entities column and '-' in the Last activity column. There are also navigation controls like '< 1 >' and a settings gear icon.

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linker)	-

3. Under Trusted entity type, select AWS service.

Trusted entity type

- AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
Lambda

Choose a use case for the specified service.
Use case

- Lambda
Allows Lambda functions to call AWS services on your behalf.

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.

Add permissions [Info](#)

Permissions policies (1/956) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type		
<input type="text"/> lambdas3 X All types ▼ 1 match C		
<input checked="" type="checkbox"/> Policy name C	Type	Description
<input checked="" type="checkbox"/> LambdaS3Policy	Customer managed	/
Set permissions boundary - optional		

Cancel Previous Next

8. Click **Next** to proceed.

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

Name, review, and create

Role details

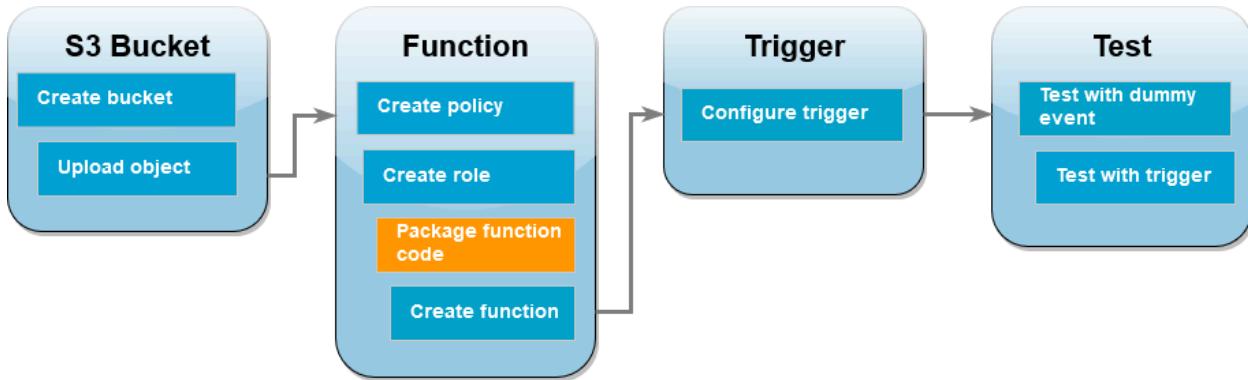
Role name
Enter a meaningful name to identify this role.
 LambdaS3Role

Maximum 64 characters. Use alphanumeric and '+=_,@-_` characters.

Description
Add a short explanation for this role.
 Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=_., @_-/[]!#\$%^&*();;"<>`

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))
```

```

❸ lambda_function.py 4 ✘
❹ lambda_function.py > ❺ lambda_handler
1 import boto3
2 import os
3 import sys
4 import uuid
5 from urllib.parse import unquote_plus
6 from PIL import Image
7 import PIL.Image
8
9 s3_client = boto3.client('s3')
10
11 def resize_image(image_path, resized_path):
12     with Image.open(image_path) as image:
13         image.thumbnail(tuple(x / 2 for x in image.size))
14         image.save(resized_path)
15
16 def lambda_handler(event, context):
17     for record in event['Records']:
18         bucket = record['s3']['bucket']['name']
19         key = unquote_plus(record['s3']['object']['key'])
20         tmpkey = key.replace('/', '')
21         download_path = '/tmp/{}{}'.format(uuid.uuid4(), tmpkey)
22         upload_path = '/tmp/resized-{}'.format(tmpkey)
23         s3_client.download_file(bucket, key, download_path)
24         resize_image(download_path, upload_path)
25         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
```

```

❸ PS D:\D15C36AtharvNikam\documentation> mkdir package1

Directory: D:\D15C36AtharvNikam\documentation

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

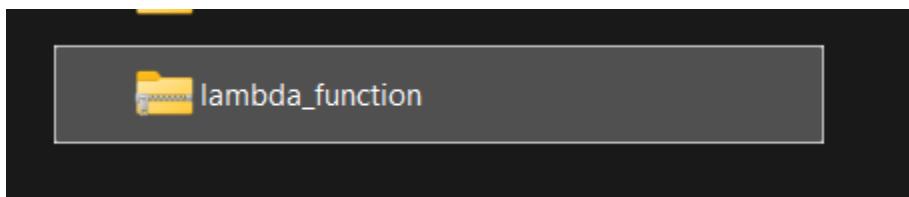
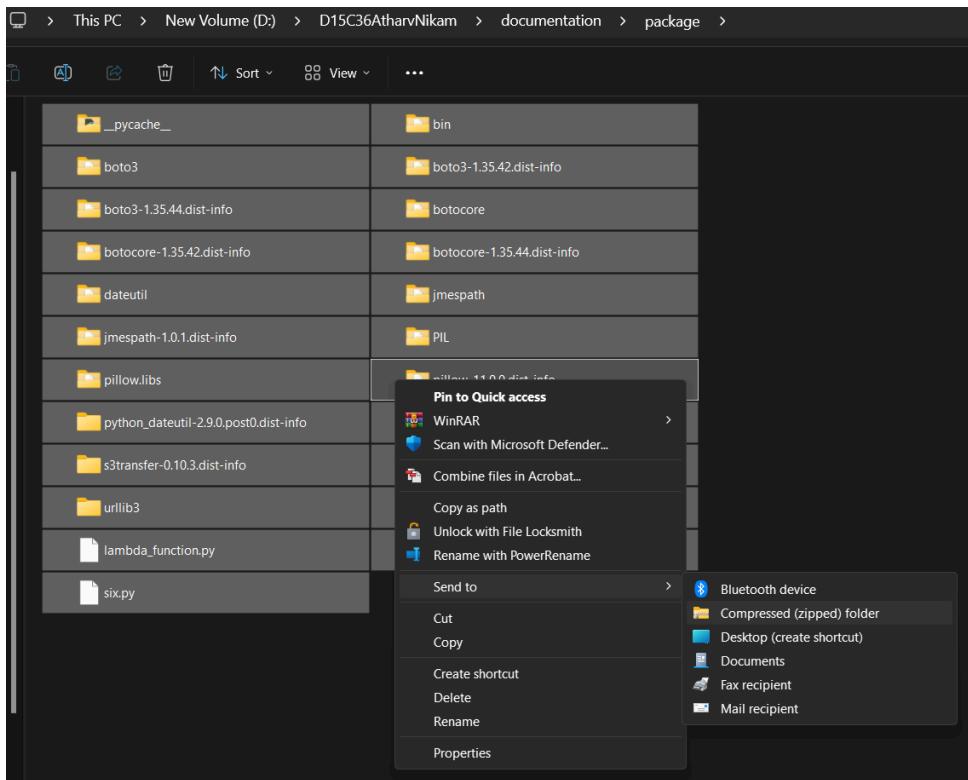
❹ PS D:\D15C36AtharvNikam\documentation> pip install `

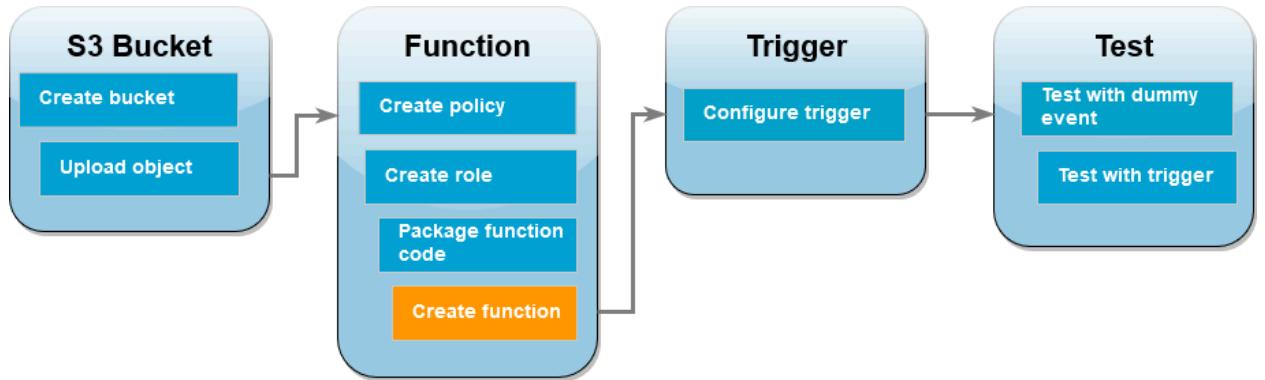
>> --platform manylinux2014_x86_64 `
>> --target=package `
>> --implementation cp `
>> --python-version 3.12 `
>> --only-binary=:all: `
>> --upgrade `
>> pillow boto3
Collecting pillow
  Using cached pillow-11.0.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.1 kB)
Collecting 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.

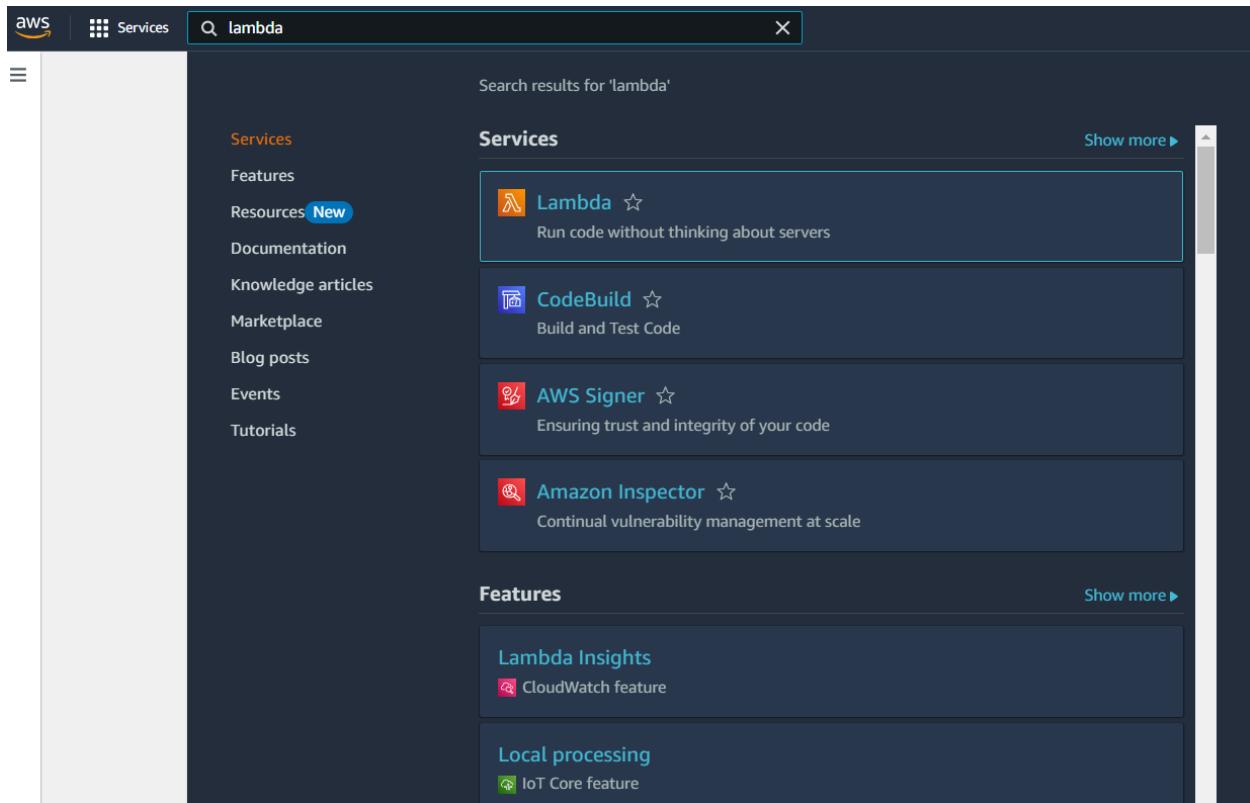




Steps to Create the Lambda Function (Console)

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**.

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy for your function.

Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

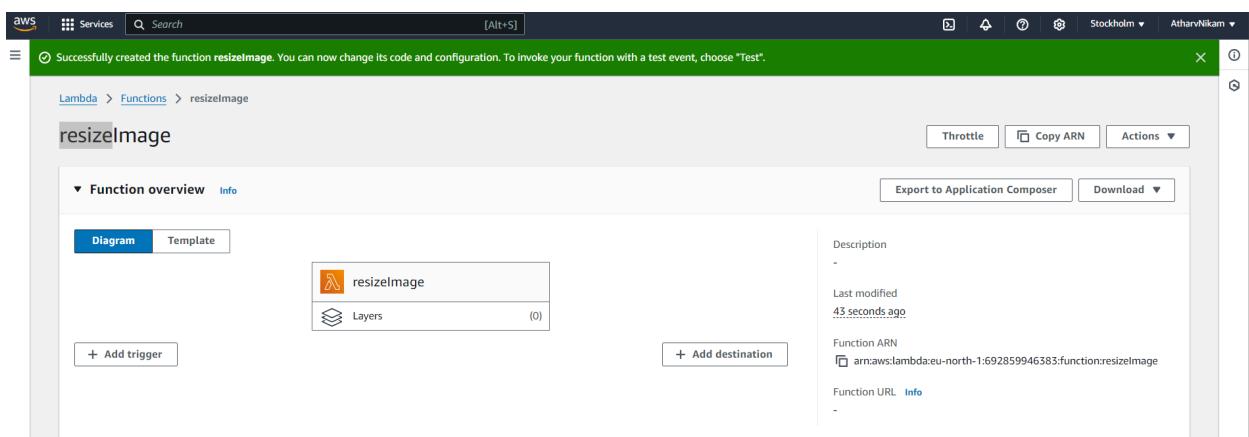
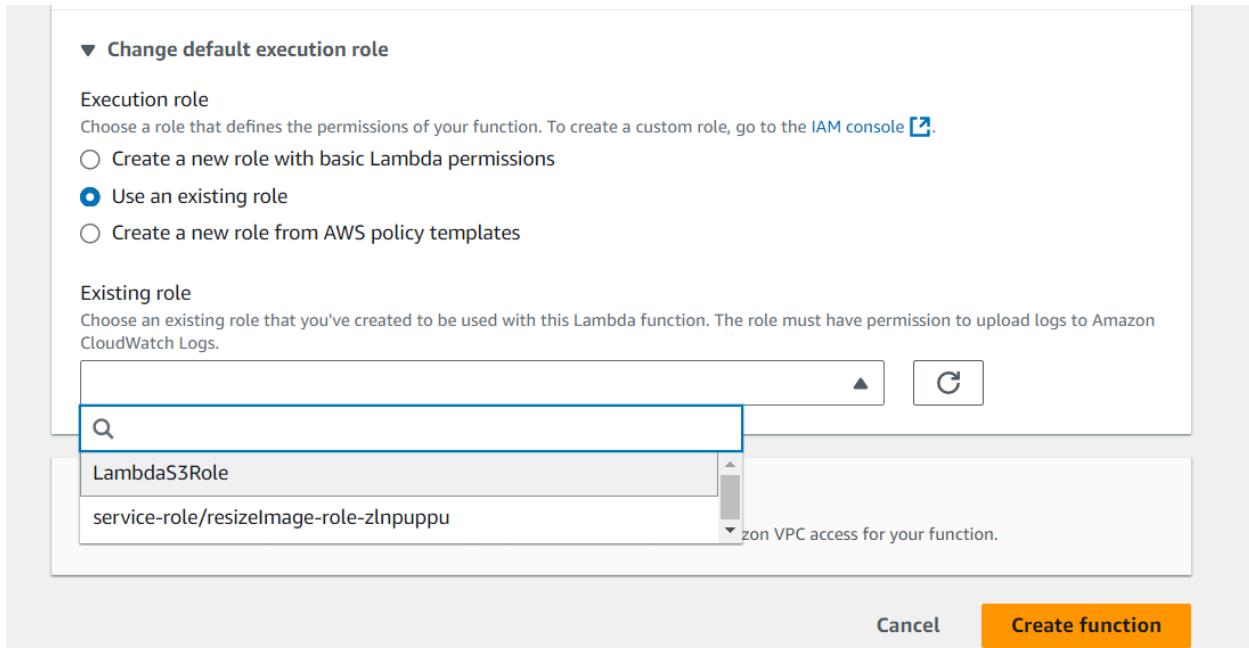
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture Info
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

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

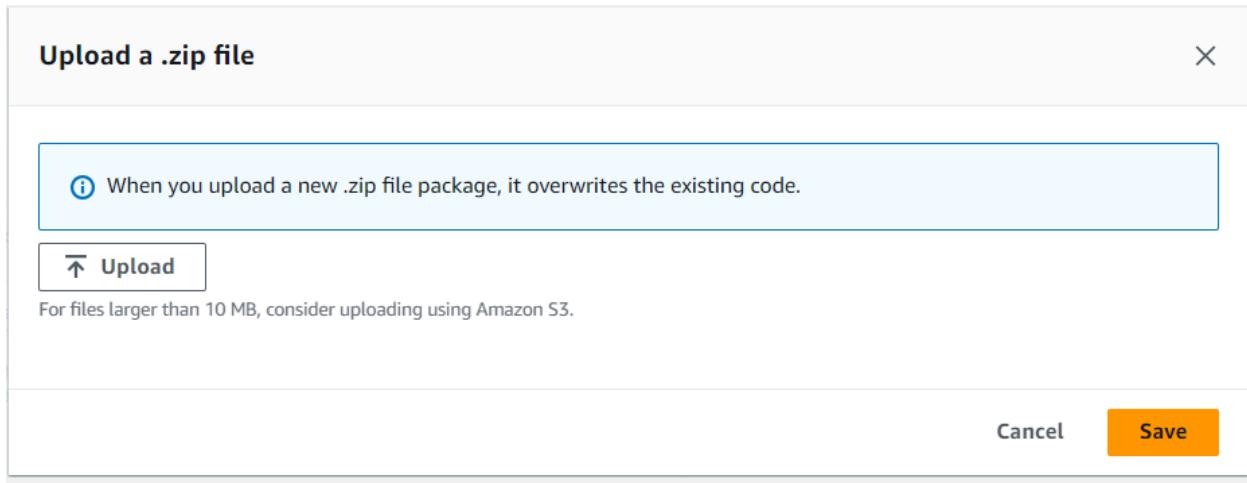
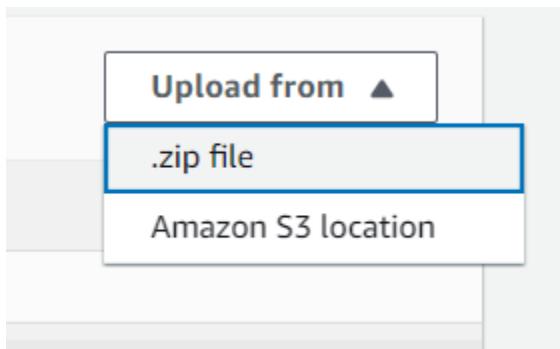
- Expand the tab and select **Use an existing role**.
- Choose the **LambdaS3Role** you created earlier.

Click **Create function**.



Upload Function Code:

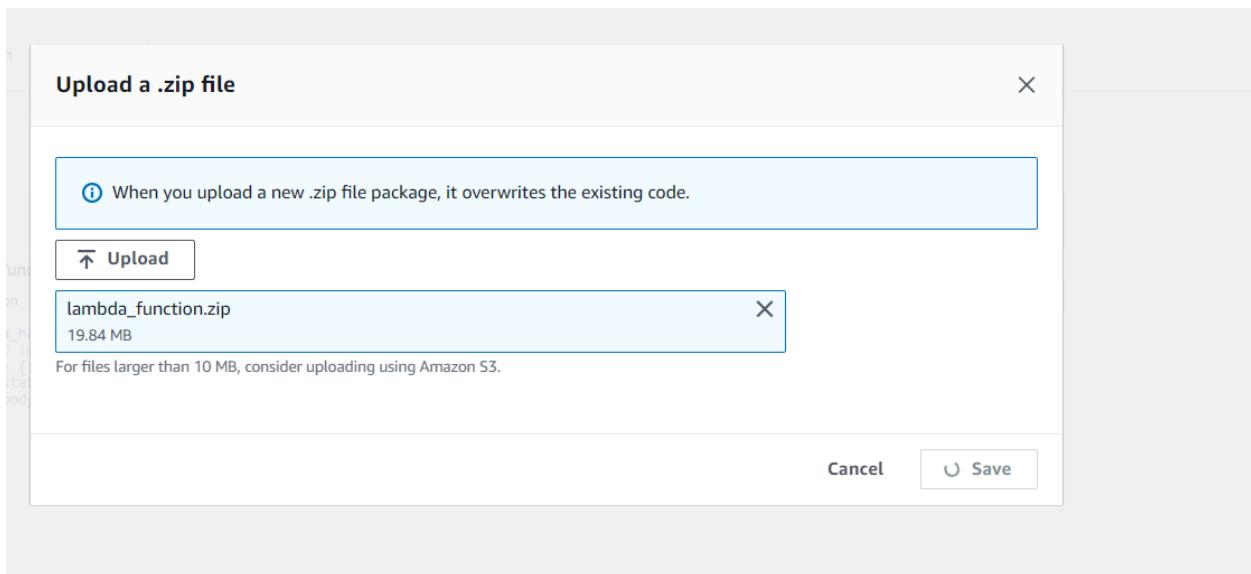
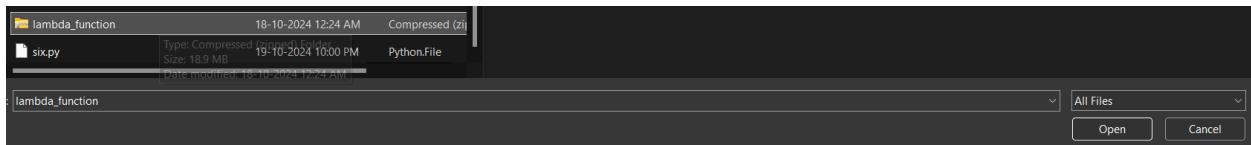
- In the **Code source** pane, select **Upload from**.
- Choose **.zip file**.
- Select your **.zip** file and click **Open**.
- Click **Save**.

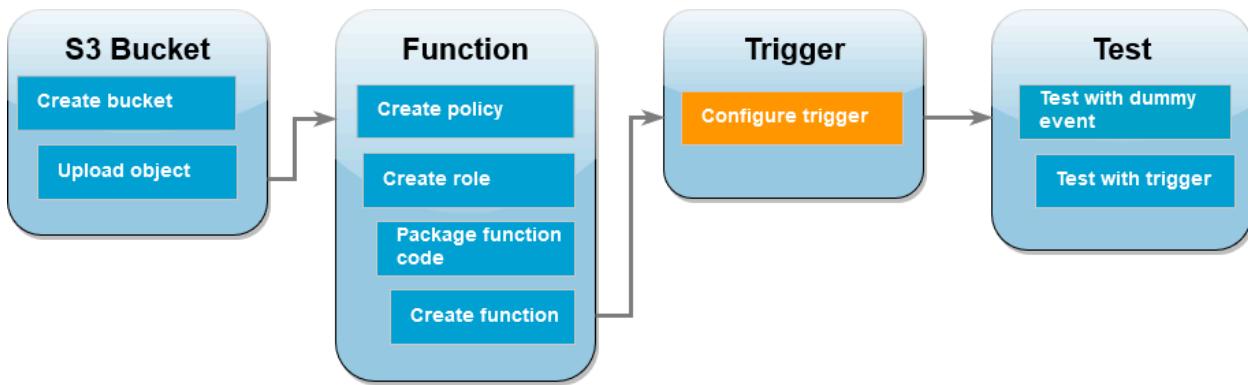


Name:Atharv Sanjay Nikam

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:resizelimage

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.

Lambda > Add triggers

Add trigger

Trigger configuration [Info](#)

S3 aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/atharv-s3-source-bucket [X](#) [C](#)

Bucket region: eu-north-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events [X](#)

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.
e.g. images/

Suffix - optional

The screenshot shows the configuration of a Lambda trigger for an S3 bucket. The trigger type is set to "All object create events". Under the "Event details" section, the "Event name" is "All object create events" and the "Prefix" is left empty. A note states: "must be in the same region as the Lambda function". Below the event details, there is a checkbox acknowledgment: "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." A note below this checkbox says: "Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model." At the bottom right, there are "Cancel" and "Add" buttons.

All object create events

must be in the same region as the Lambda function

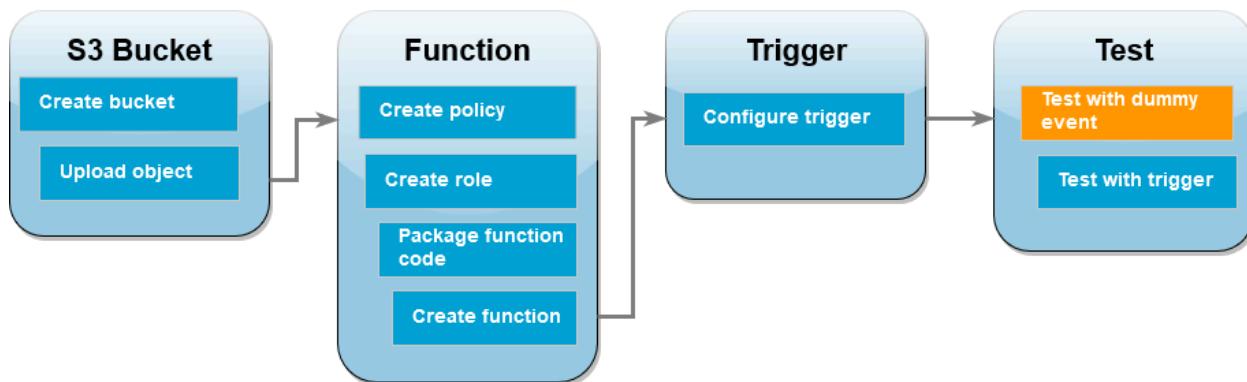
All object create events

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.

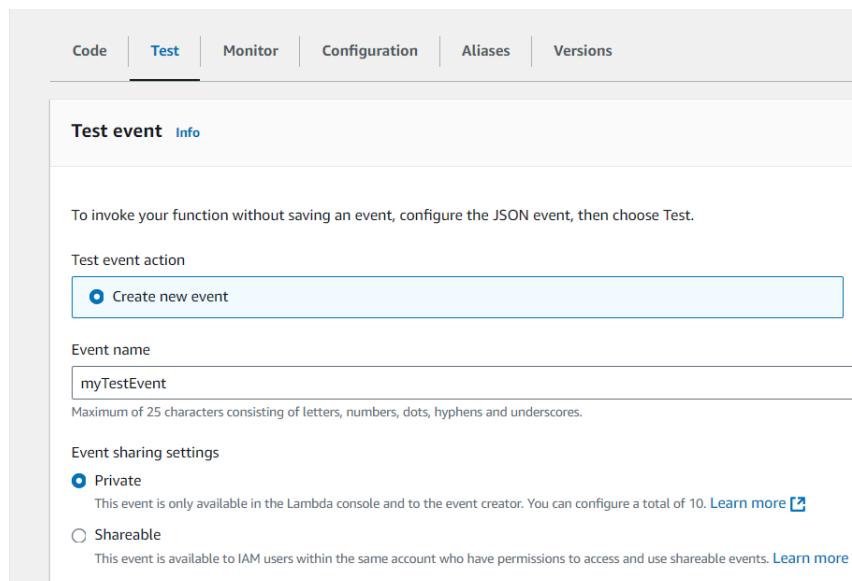
Cancel Add



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

1. Access Your Lambda Function:

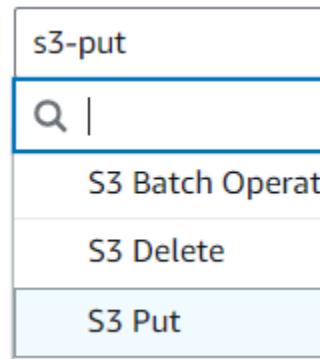
- Open the **Functions** page in the AWS Lambda console.
- Select your function: **CreateThumbnail**.



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": "EXAMPLE123456789"
      }
    },
    "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.

The screenshot shows the AWS Lambda function configuration interface. At the top, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Test tab is currently selected. Below the tabs, a red error message box contains the text "Executing function: failed (logs [?])" and a "Details" link. The main configuration area has tabs for General configuration, Triggers, Permissions, Destinations, Function URL, and Environment variables. The General configuration tab is selected. It displays settings for Description (empty), Memory (128 MB), Ephemeral storage (512 MB), Timeout (0 min 3 sec), and SnapStart (None). An "Edit" button is located in the top right corner of this section.

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/resizelimage-role-zlnpuppu

[View the resizelimage-role-zlnpuppu role](#) on the IAM console.

[Cancel](#) [Save](#)

✔ Successfully updated the function **resizelimage**.

Code [Test](#) Monitor Configuration Aliases Versions

Executing function: succeeded ([logs](#))

▶ Details

Name:Atharv Sanjay Nikam

Div:D15C

Roll:36

Amazon S3 > Buckets > atharv-s3-source-bucket-resized

atharv-s3-source-bucket-resized [Info](#)

Objects Properties Permissions Metrics Management Access Points

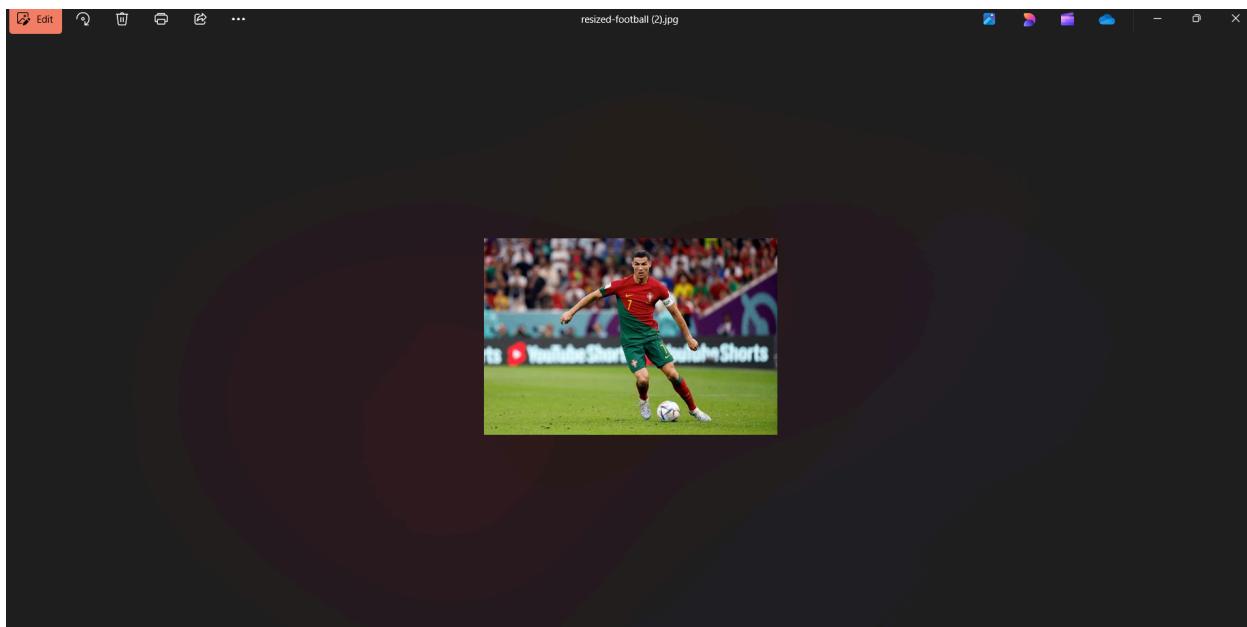
Objects (1) [Info](#)

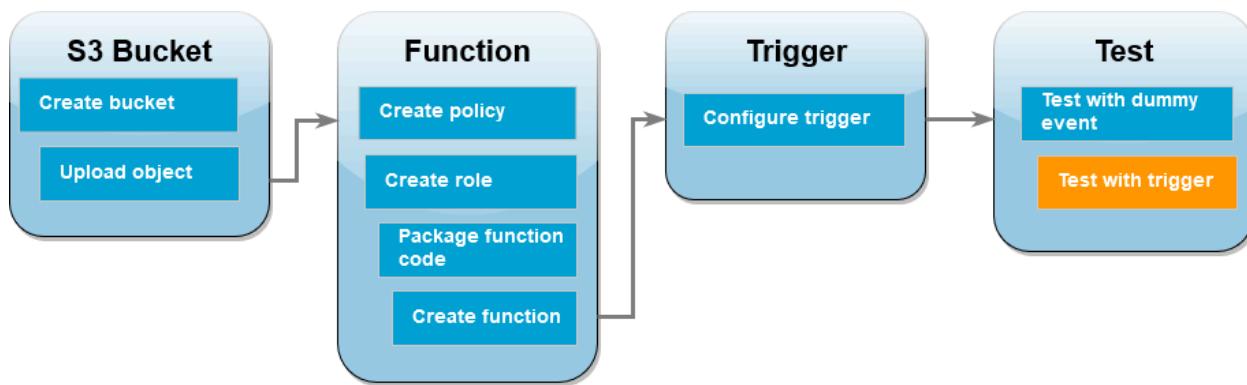
C Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	resized-football.jpg	jpg	October 19, 2024, 22:49:11 (UTC+05:30)	22.8 KB	Standard





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**.

Amazon S3 > Buckets > atharv-s3-source-bucket > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

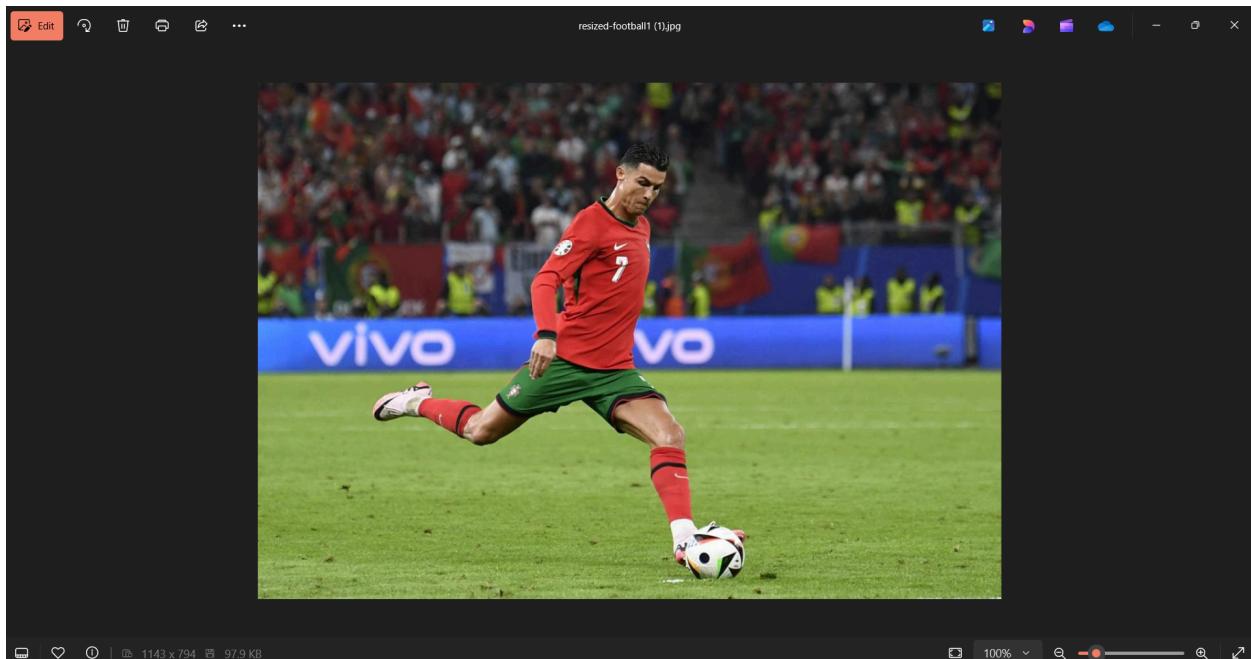
Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 990.9 KB)		Remove	Add files	Add folder
All files and folders in this table will be uploaded.				
<input type="text"/> Find by name		<	1	>
<input type="checkbox"/>	Name	Folder	Type	
<input type="checkbox"/>	football1.jpg	-	image/jpeg	

Name:Atharv Sanjay Nikam

Div:D15C

Roll:36



Amazon S3 > Buckets > atharv-s3-source-bucket-resized

atharv-s3-source-bucket-resized [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#)

Objects (2) [Info](#)

[Op...](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to grant them permissions. [Learn more](#)

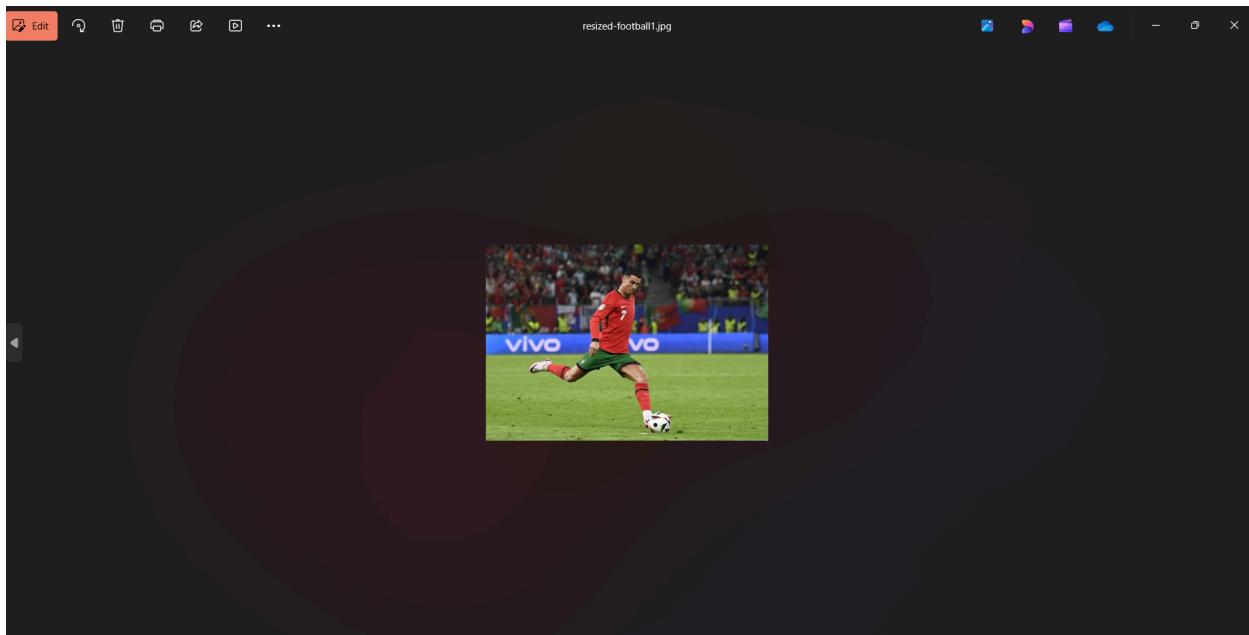
[Find objects by prefix](#)

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	resized-football.jpg	jpg	October (UTC+0)
<input type="checkbox"/>	resized-football1.jpg	jpg	October (UTC+0)

Name:Atharv Sanjay Nikam

Div:D15C

Roll:36



Name:Atharv Sanjay Nikam

Div:D15C

Roll:36