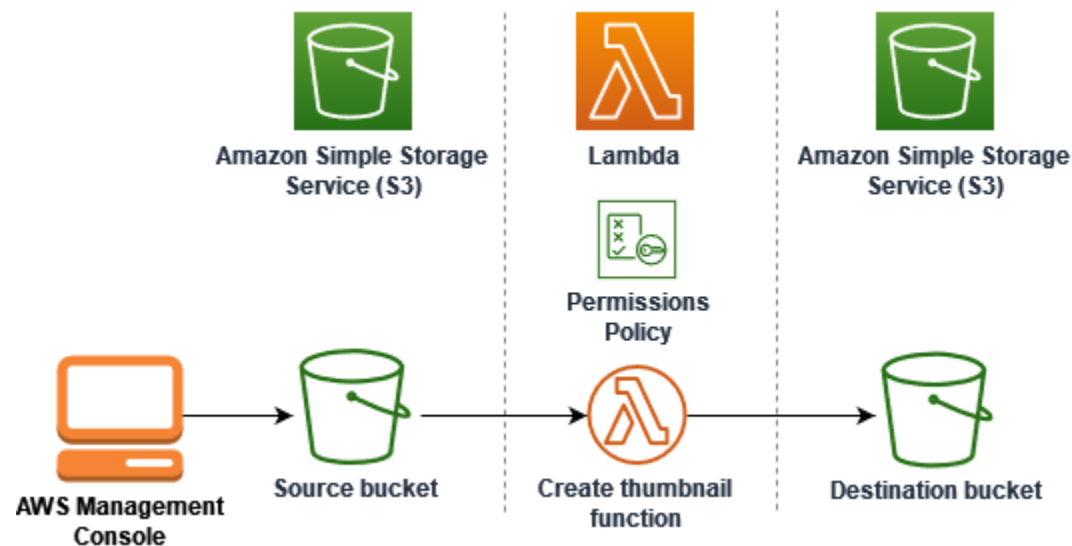


## ADVANCE DEVOPS PRACTICAL EXAM

### 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.

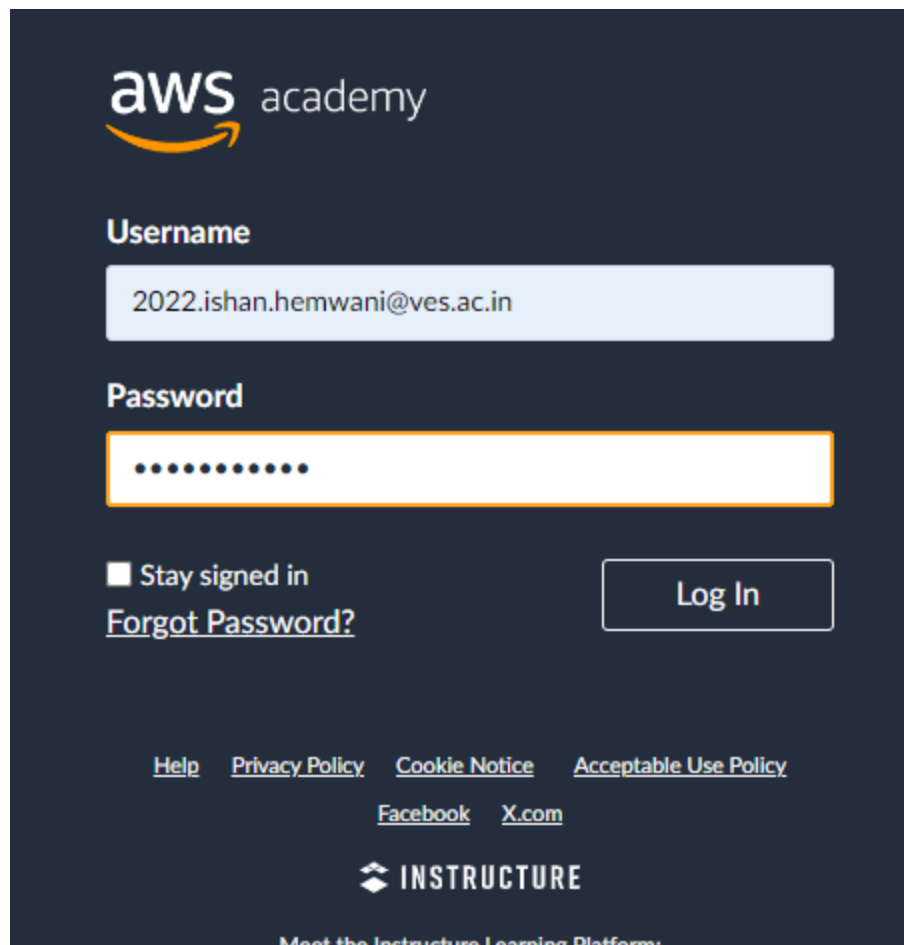
Furthermore, the solution is highly scalable, as AWS Lambda automatically handles the scaling of functions in response to increased traffic, without requiring any manual intervention. It is also cost-effective, as you only pay for the compute time you use, making it ideal for applications with fluctuating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

Sign in to aws academy and entering your AWS account email address. On the next page, enter your password.

Logging in AWS academy to perform AWS Lambda function



aws academy

Username

2022.ishan.hemwani@ves.ac.in

Password

.....

☐ Stay signed in

[Forgot Password?](#)

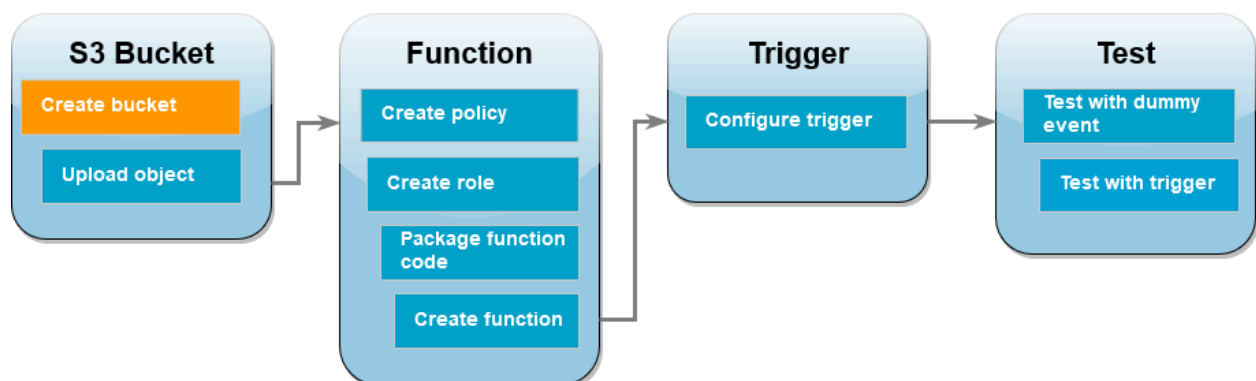
Log In

[Help](#) [Privacy Policy](#) [Cookie Notice](#) [Acceptable Use Policy](#)

[Facebook](#) [X.com](#)

INSTRUCTURE

Meet the Instructure Learning Platform:



### Create Source and Destination Buckets:

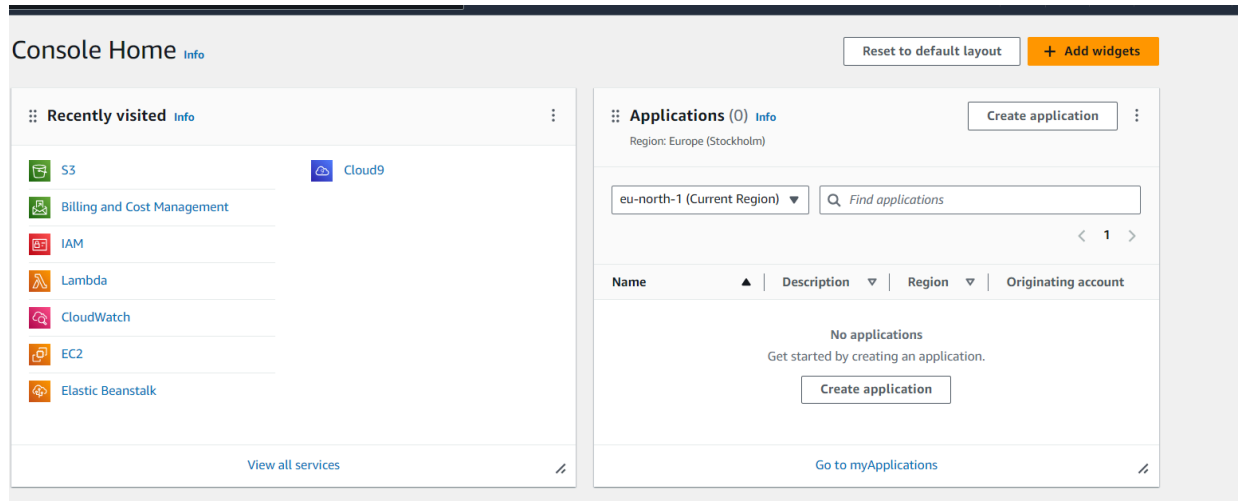
I created two S3 buckets – one for uploading the original images (source bucket) and another to store the resized images (destination bucket).

*Example:*

- Source Bucket: `ishan-s3-source-bucket`

- Destination Bucket: **ishan-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: **ishan-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**.

[Amazon S3](#) > [Buckets](#) > Create bucket

## Create bucket [Info](#)

Buckets are containers for data stored in S3.

### General configuration

AWS Region  
US East (N. Virginia) us-east-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

✔ **Successfully created bucket "Ishan-s3-source-bucket"**  
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

[Amazon S3](#) > [Buckets](#)

▶ **Account snapshot** - updated every 24 hours [All AWS Regions](#)  
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[General purpose buckets](#) | [Directory buckets](#)

**General purpose buckets (3)** [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

	Name ▲	AWS Region ▼	IAM Access Analyzer
<input type="radio"/>	<a href="#">elasticbeanstalk-us-east-1-365817213500</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>
<input type="radio"/>	<a href="#">ishan-s3-source-bucket</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>

## Create a Destination Bucket:

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

## Create bucket [Info](#)

Buckets are containers for data stored in S3.

### General configuration

AWS Region

US East (N. Virginia) us-east-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](#)

ishan-s3-source-bucket-resized.

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

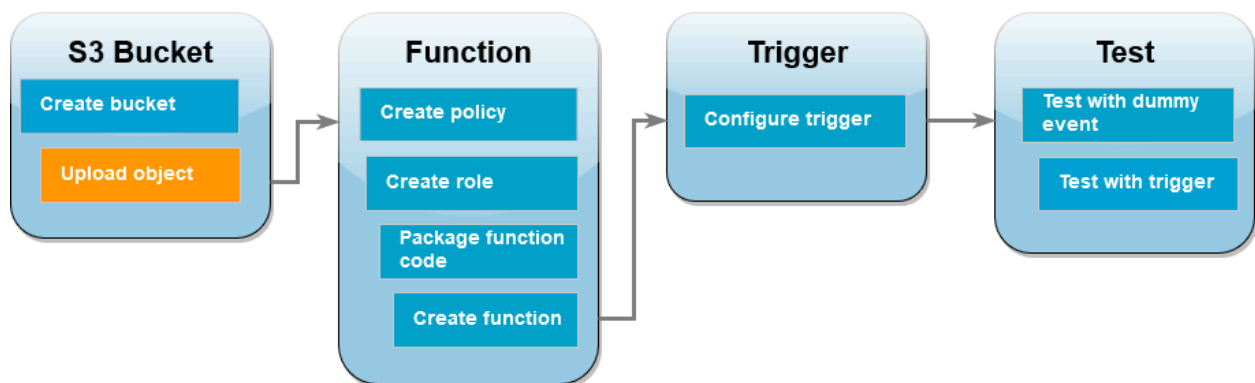
### ► Advanced settings



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

Cancel

Create bucket

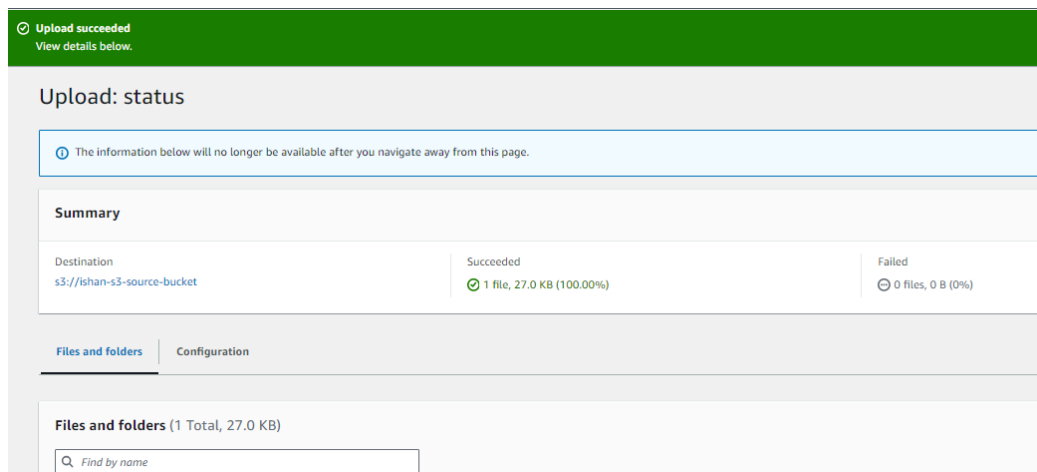
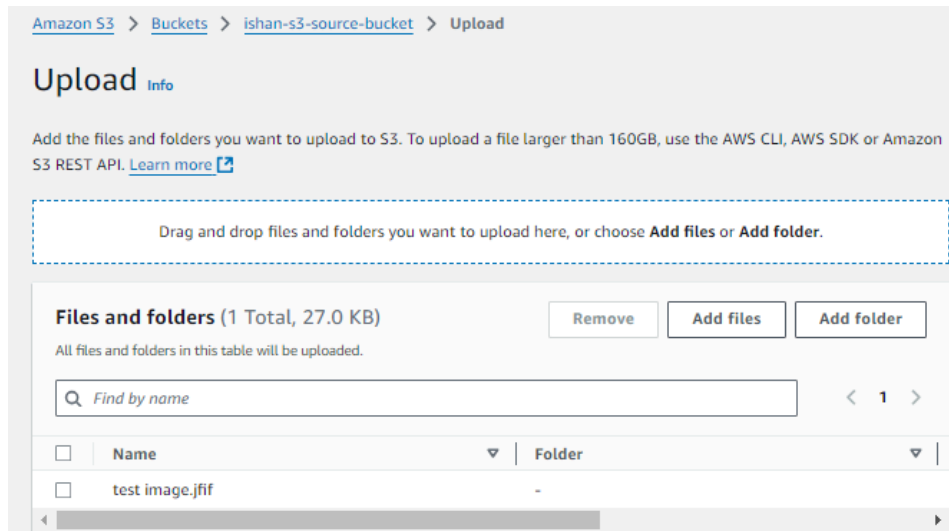


## Uploading a Test Image to the Source Bucket (via AWS Console)

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**.



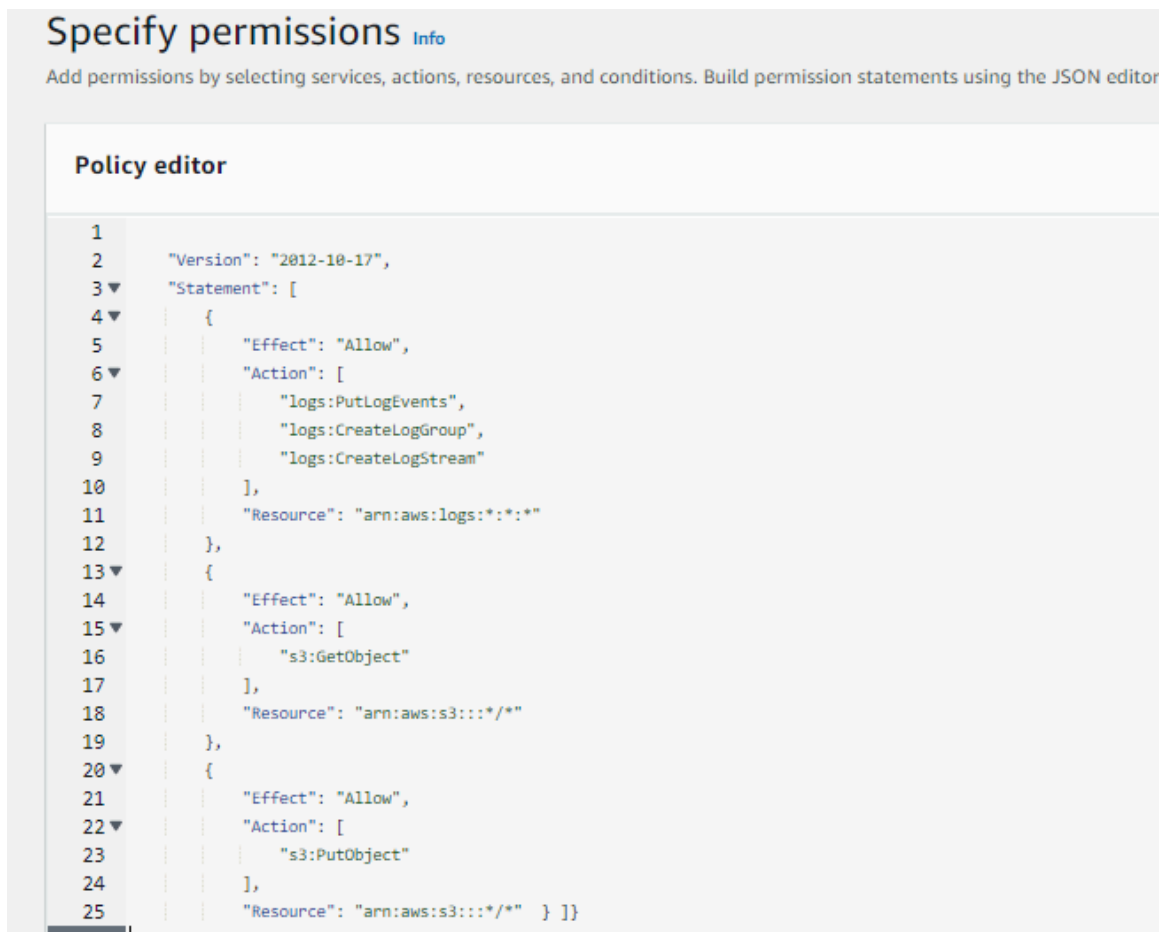
## Step 2: Permissions Setup

- **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.



The screenshot shows the 'Specify permissions' page in the AWS IAM console. The page title is 'Specify permissions' with an 'Info' link. Below the title is a subtitle: 'Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor'. The main content area is titled 'Policy editor' and displays a JSON policy document. The policy is named 'LambdaS3Policy' and is in the 'JSON' tab. The JSON document is as follows:

```
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:::*/*" } ] }
```



## 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*

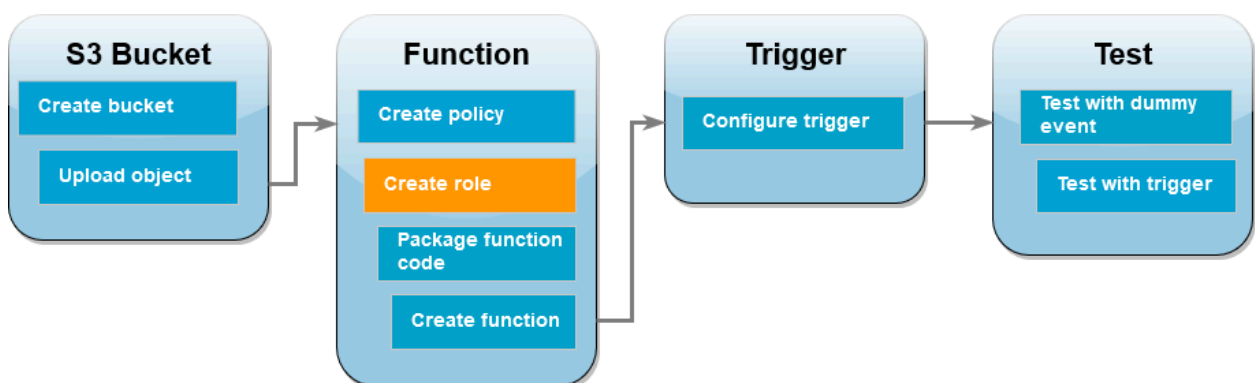
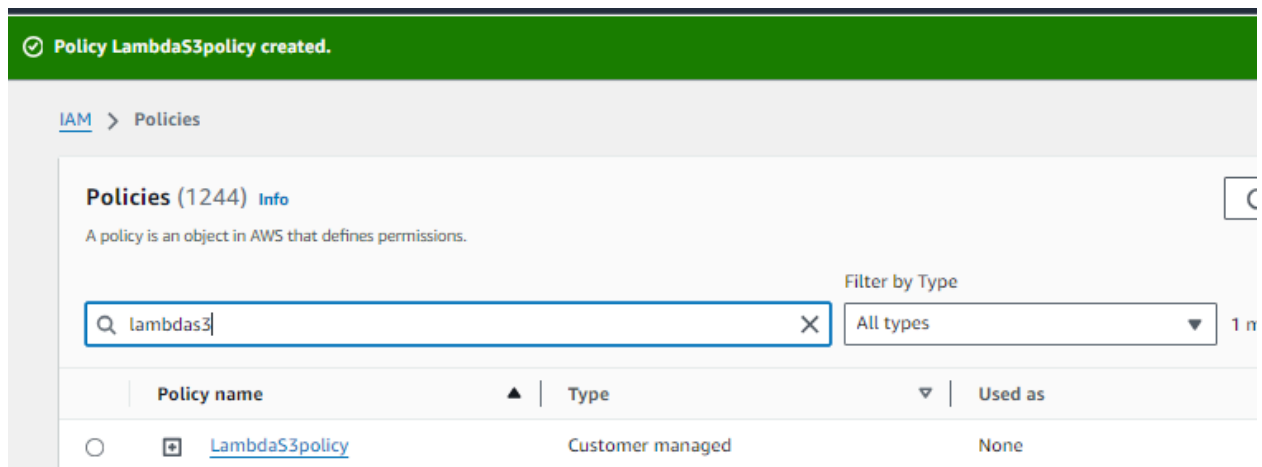
Add a short explanation for this policy.

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

Code:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "logs:PutLogEvents",  
        "logs:CreateLogGroup",  
        "logs:CreateLogStream"  
      ],  
      "Resource": "arn:aws:logs:*:*:*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject"  
      ],  
      "Resource": "arn:aws:s3:::*/*"
```

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

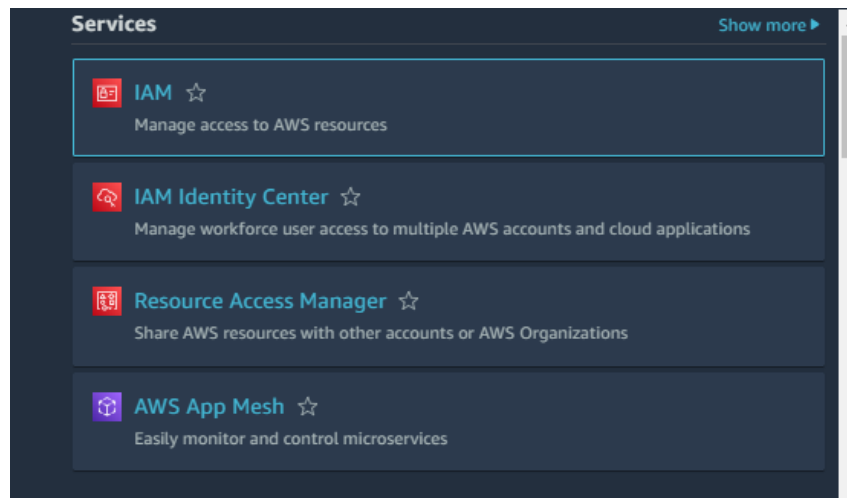


## Creating an Execution Role and Attaching Permissions Policy

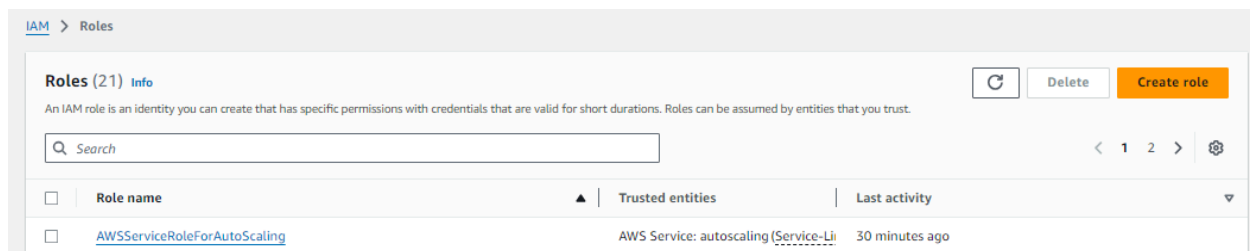
To enable the Lambda function to interact with the Amazon S3 buckets, it requires an **IAM execution role**. This role grants the necessary permissions for the function to read from the source bucket and write to the destination bucket, which are essential for the image resizing process.

### 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**.

### Select trusted entity Info

**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

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

### Add permissions Info

**Permissions policies (1/960)** Info

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

Filter by Type

Q Lambdas3 X All types 1 match

<input checked="" type="checkbox"/>	Policy name <small>?</small>	Type	Description
<input checked="" type="checkbox"/>	<a href="#">LambdaS3policy</a>	Customer managed	-

► Set permissions boundary - *optional*

- Click **Next** to proceed.

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

### Name, review, and create

#### Role details

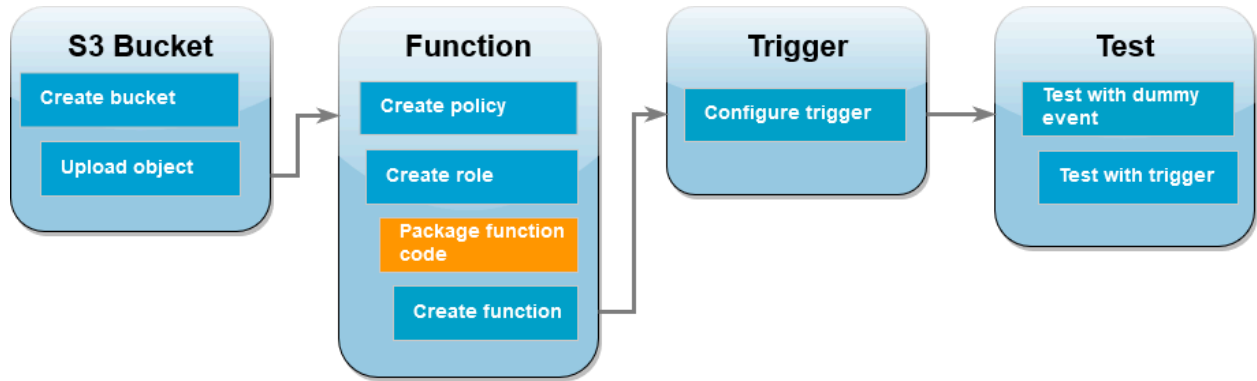
**Role name**  
Enter a meaningful name to identify this role.

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

**Description**  
Add a short explanation for this role.

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
```

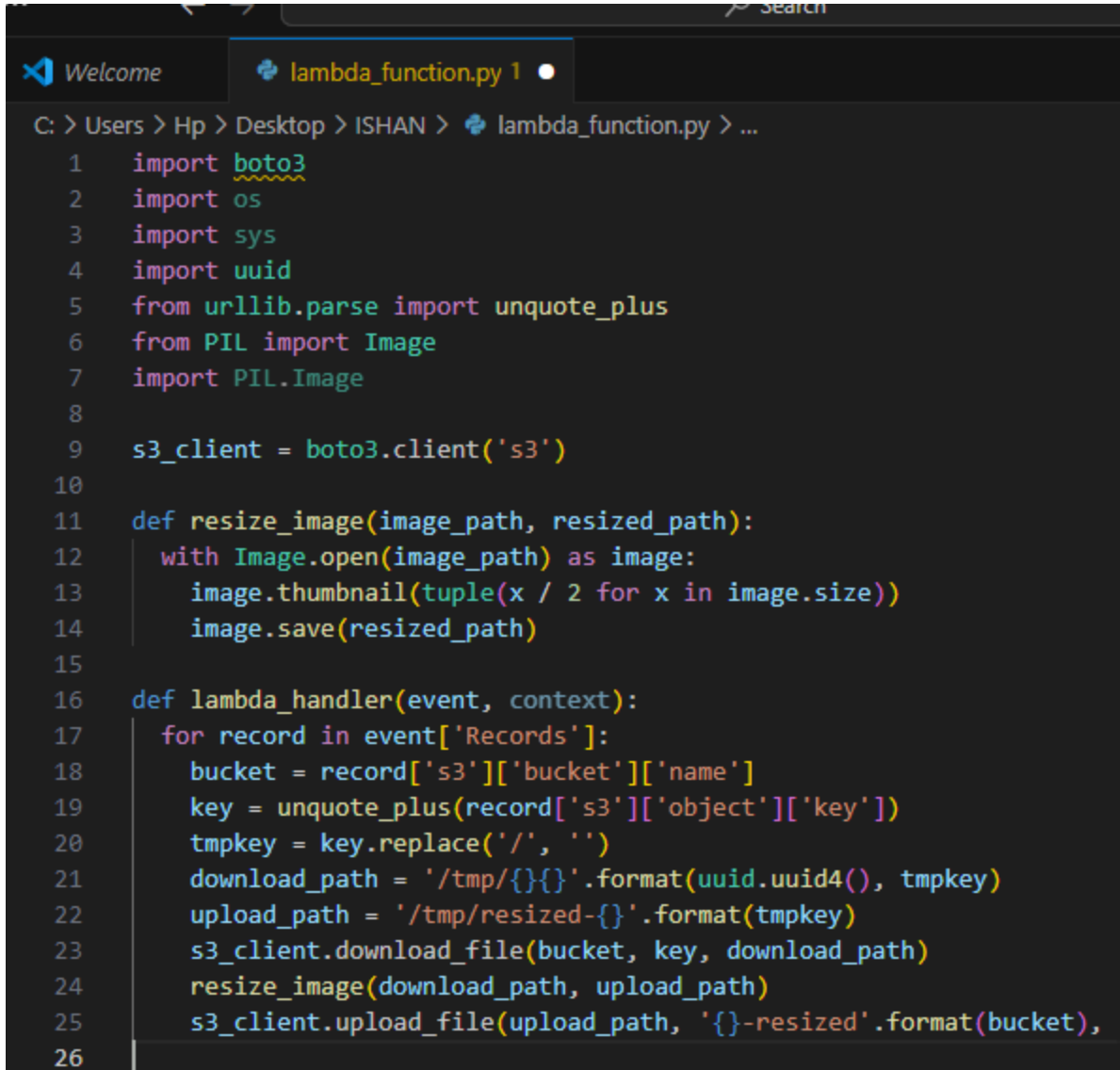
```
import PIL.Image
```

```
s3_client = boto3.client('s3')
```

```
def resize_image(image_path, resized_path):
```

```
    with Image.open(image_path) as image:
```

```
s3_client.upload_file(upload_path, '{}-resized'.format(bucket),
'resized-{}'.format(key))
```



```
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),
26
```

### 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:

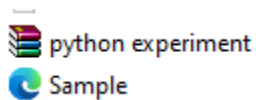
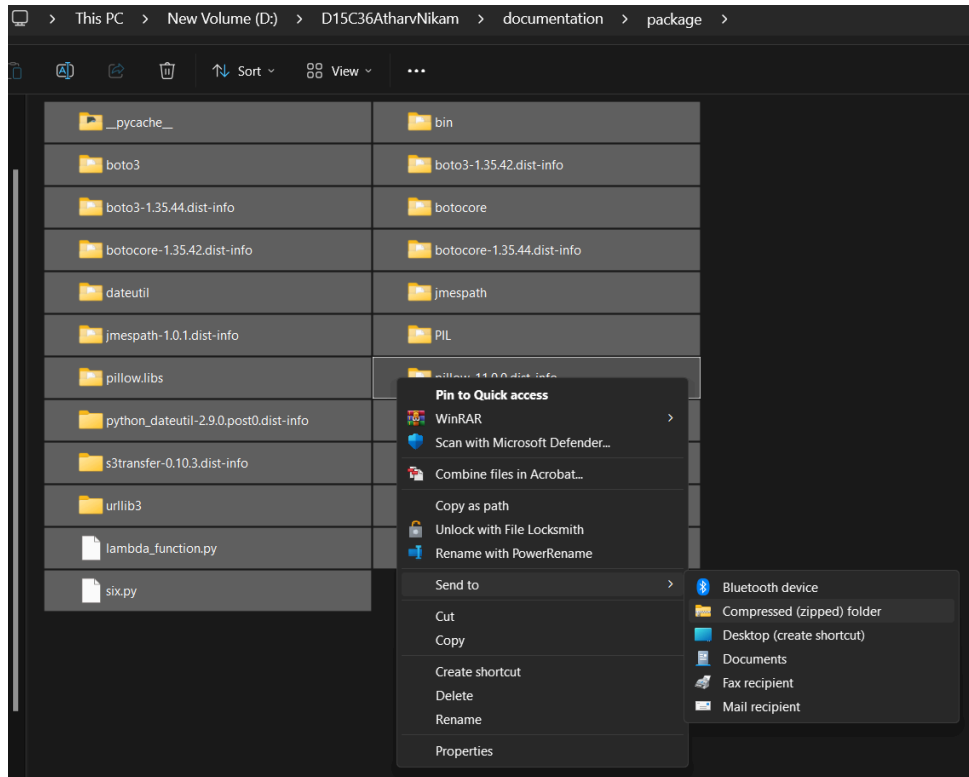
```
pip install --platform manylinux2014_x86_64 --target=package --implementation
cp --python-version 3.12 --only-binary=:all: --upgrade pillow boto3
```



```
Downloading pillow-11.0.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.1 kB)
collecting boto3
  Downloading boto3-1.35.45-py3-none-any.whl.metadata (6.7 kB)
collecting botocore<1.36.0,>=1.35.45 (from boto3)
  Downloading botocore-1.35.45-py3-none-any.whl.metadata (5.7 kB)
collecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
collecting s3transfer<0.11.0,>=0.10.0 (from boto3)
  Downloading s3transfer-0.10.3-py3-none-any.whl.metadata (1.7 kB)
collecting python-dateutil<3.0.0,>=2.1 (from botocore<1.36.0,>=1.35.45->boto3)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
collecting urllib3!=2.2.0,<3,>=1.25.4 (from botocore<1.36.0,>=1.35.45->boto3)
  Downloading urllib3-2.2.3-py3-none-any.whl.metadata (6.5 kB)
collecting six>=1.5 (from python-dateutil<3.0.0,>=2.1->botocore<1.36.0,>=1.35.45->boto3)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Downloading pillow-11.0.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.3 MB)
----- 4.3/4.3 MB 405.1 kB/s eta 0:00:00
```

### Package the Application:

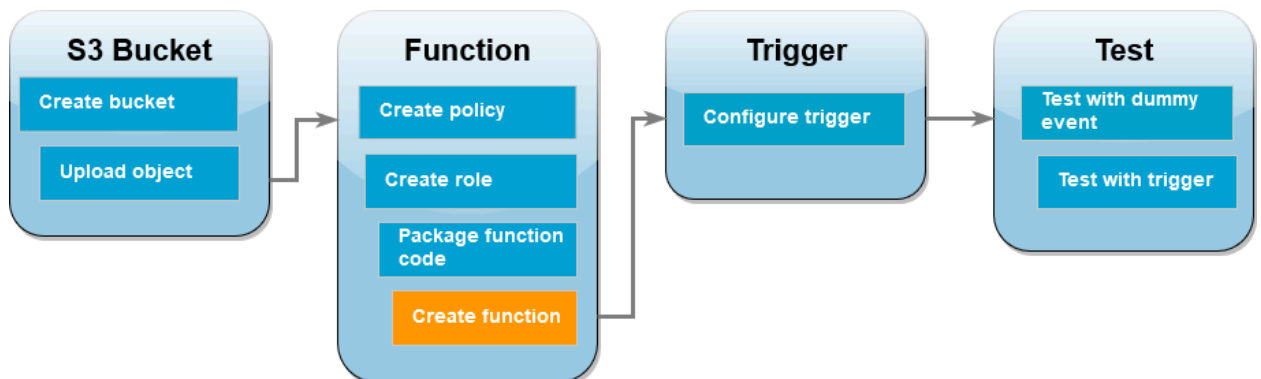
Use any ZIP tool to create `lambda_function.zip` with `lambda_function.py` and the package folder contents.



10/22/2024 3:49 AM  
8/16/2024 7:47 PM

WinRAR ZIP archive  
Microsoft Edge H...

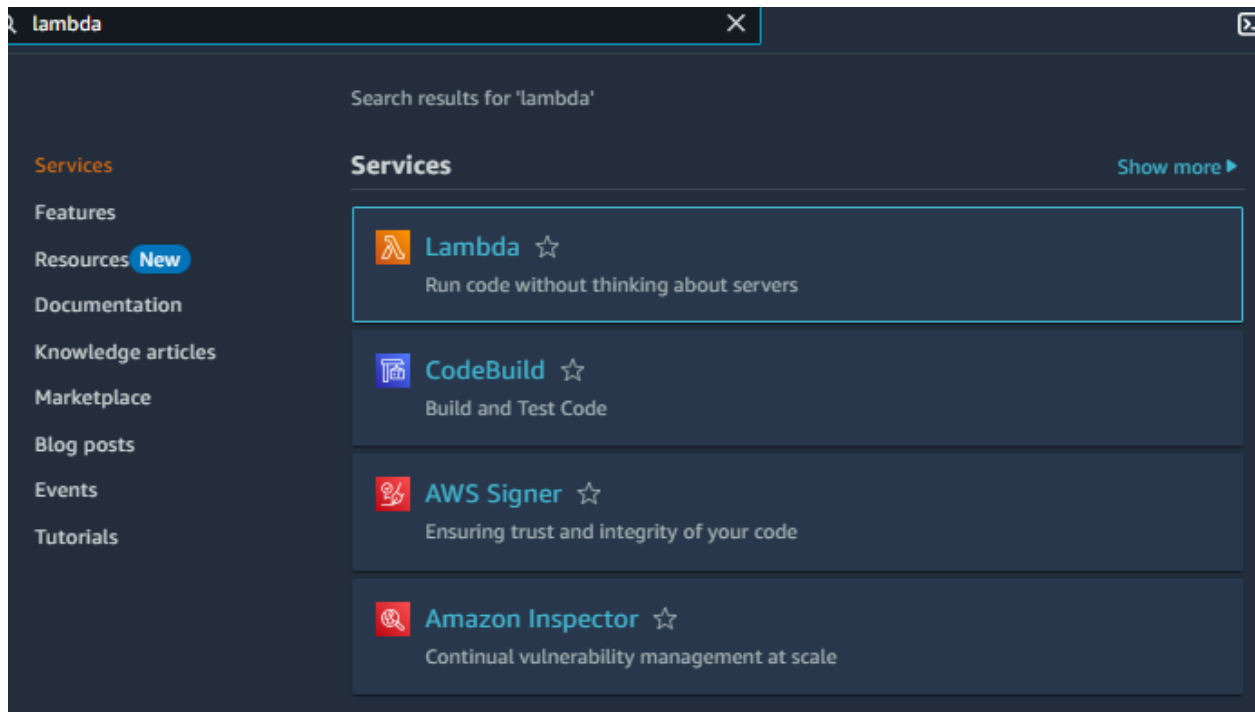
19,368 KB  
6 KB



## 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.

### 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**.

**▼ Change default execution role****Execution role**

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☐ Create a new role with basic Lambda permissions
- ☒ 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.



[View the LabRole role](#) on the IAM console.

✓ Successfully created the function **resizeImage**. You can now change its code and configuration. To invoke your function with a test event, choose "

[Lambda](#) > [Functions](#) > resizeImage

## resizeImage

**▼ Function overview** [Info](#)**Diagram**

Template

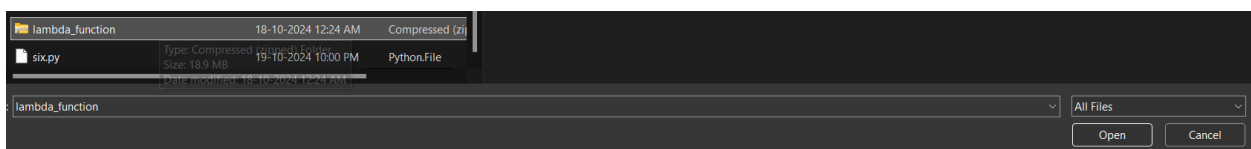
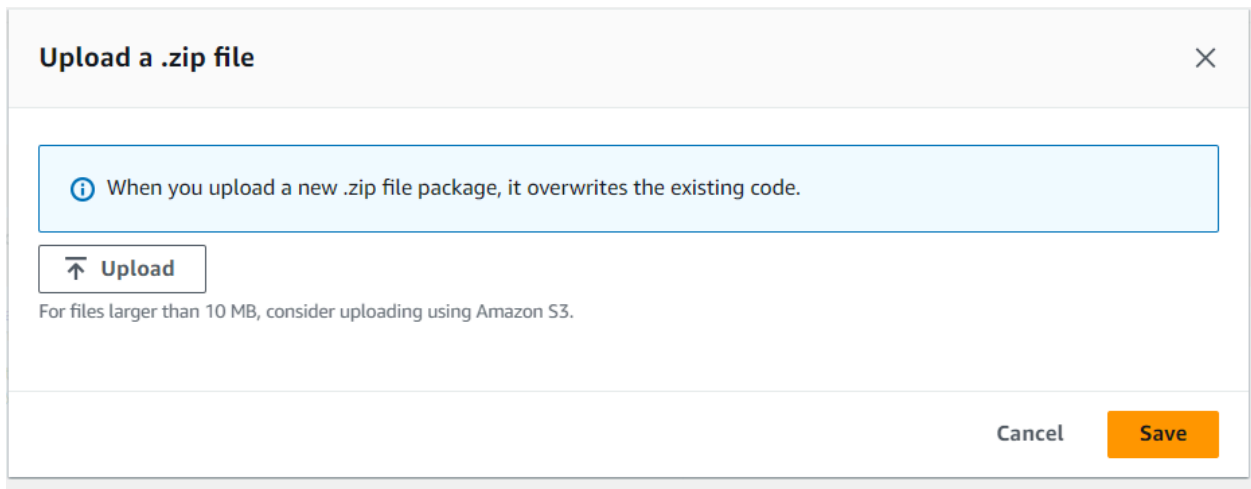
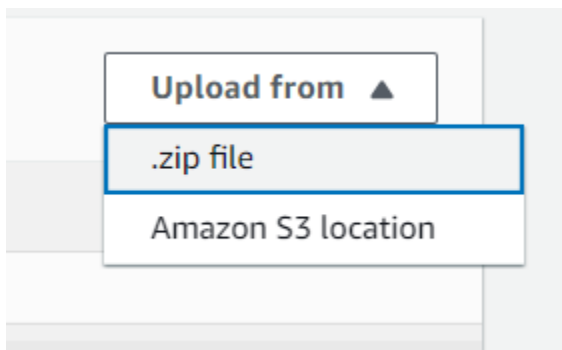
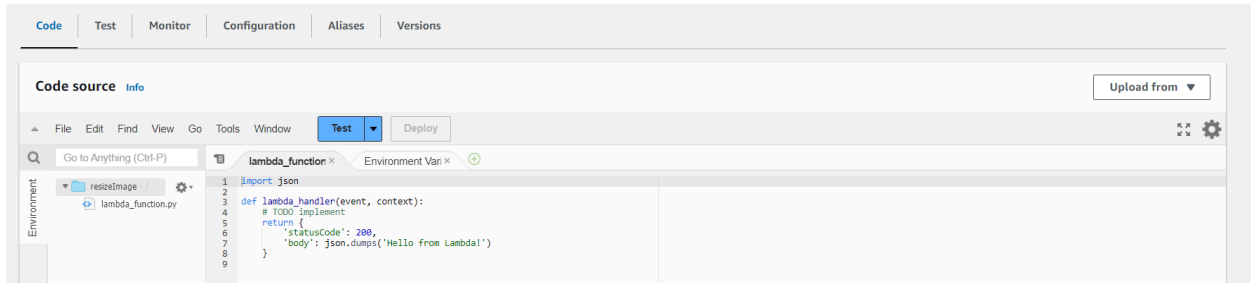
**resizeImage**

Layers

(0)

[+ Add trigger](#)[+ Add destination](#)**Upload Function Code:**

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



Upload a .zip file

When you upload a new .zip file package, it overwrites the existing code.

Upload

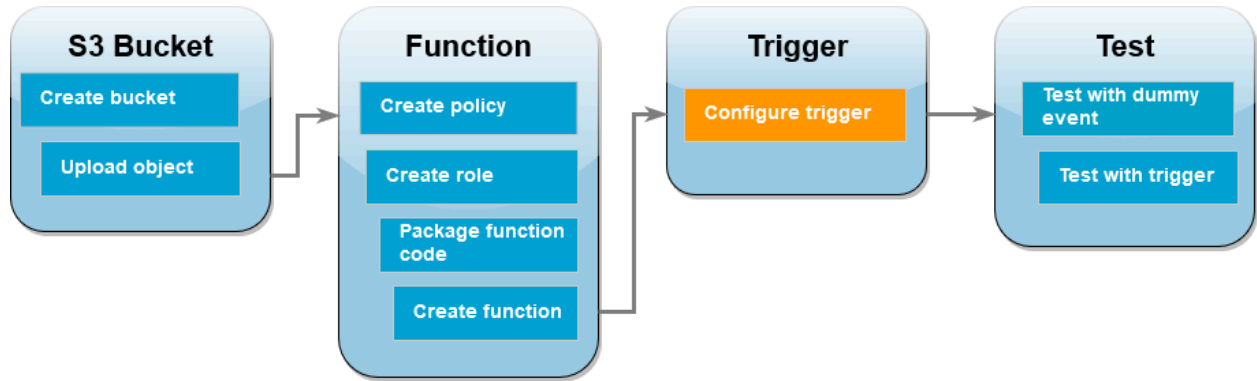
python experiment.zip  
19.83 MB

For files larger than 10 MB, consider uploading using Amazon S3.

Cancel

Save

Successfully updated the function `resizeImage`.



## 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.



[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.  
    
Bucket region: us-east-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

**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.

Search

- ☒ 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 X

Prefix - optional

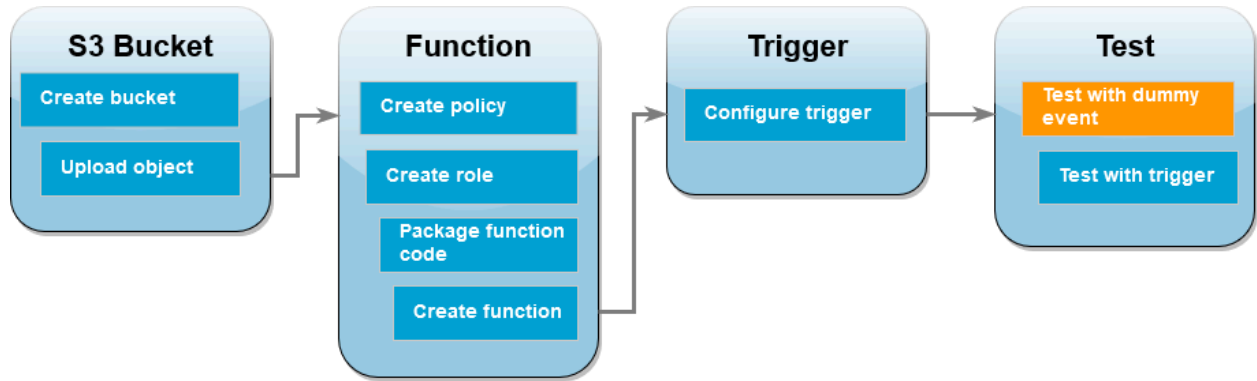
more

- ☒ 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**.

Code | **Test** | Monitor | Configuration | Aliases | Versions

**Test event** [Info](#)

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

Event name

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private  
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable  
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

### 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**.

s3-put
Q
S3 Batch Operat
S3 Delete
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":
"EXAMPLE123/5678abcdefghijklmbdaisawesome/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:

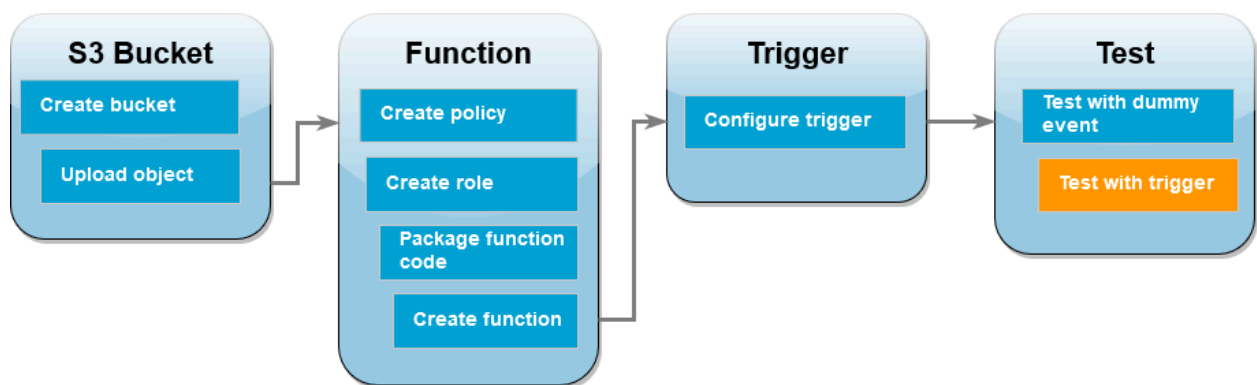
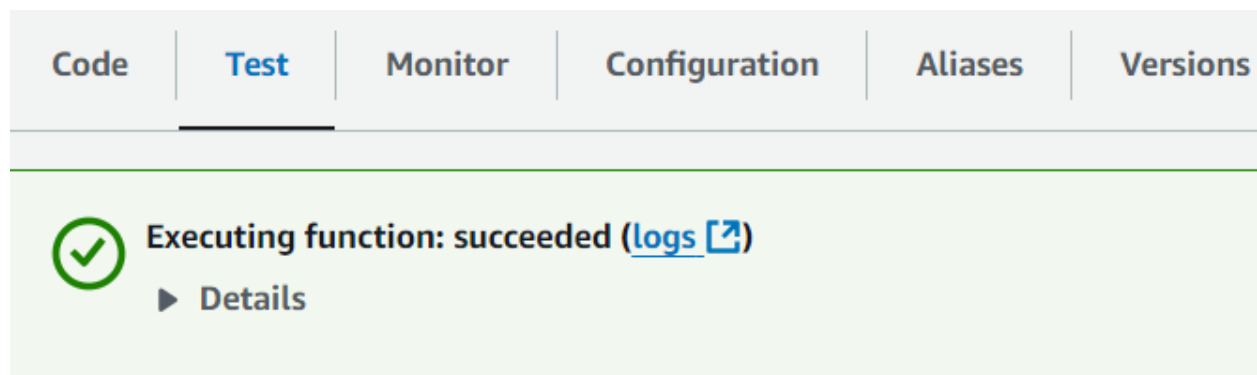
- Click **Save**.
- In the **Test event** pane, choose **Test**.

### 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.

The screenshot displays the AWS Lambda console interface for the 'resizeImage' function. At the top, a green notification bar confirms that the test event 'myTestEvent' was saved successfully and that the S3 trigger 'ishan-s3-source-bucket' has been added. The 'Function overview' section shows the function's configuration, including its description, last modified time (28 minutes ago), and its ARN. Below this, a red error banner indicates that the function execution failed, with a link to view logs and a button to diagnose with Amazon Q. The bottom of the console shows the 'Test event' pane with a 'Test' button.

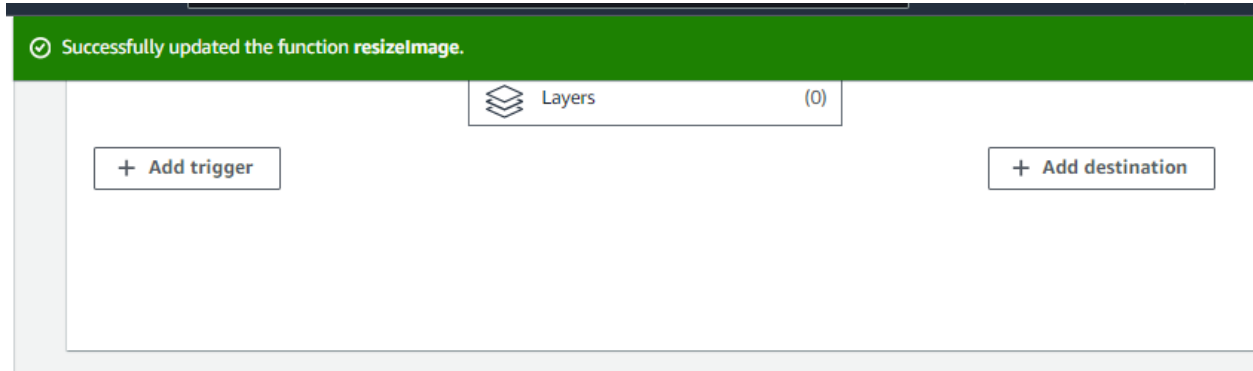
✔ Successfully updated the function **resizeImage**.



## 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

>

ishan-s3-source-bucket-resized

ishan-s3-source-bucket-resized

Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (1)

Info

Refresh

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

< 1 > ⚙

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 <a href="#">test image.jfif</a>	jfif	October 22, 2024, 04:54:20 (UTC+05:30)	27.0 KB	Standard



Name: Ishan Hemwani

Div:D15C

Roll:16