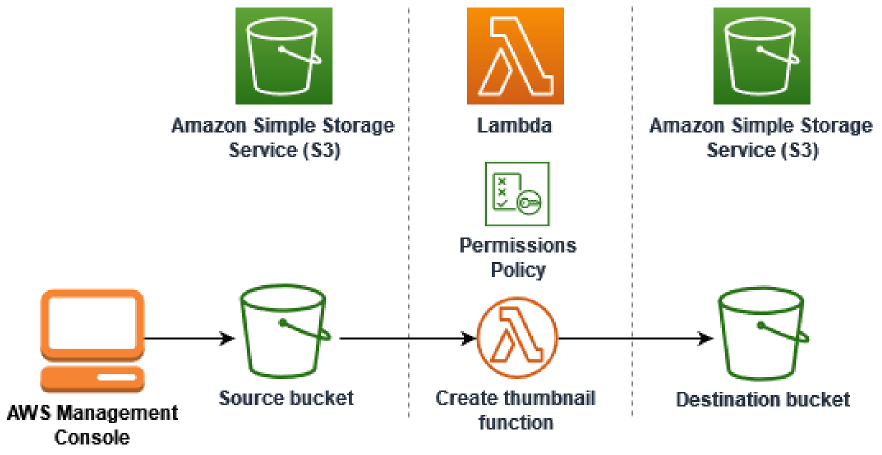
Name: Bhagyesh Patil **Div:D15A** **Roll:36**

ADVANCE DEVOPS PRACTICAL EXAM

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

Case Study Overview: This case study demonstrates an automated image resizing system using AWS Lambda and Amazon S3. The serverless architecture allows for dynamic resizing of images uploaded to one S3 bucket, generating thumbnails that are stored in another bucket. The process is event-driven, triggered by image uploads that invoke the Lambda function. This highlights the benefits of cloud-native technologies and serverless computing, providing scalability and efficiency without infrastructure management.

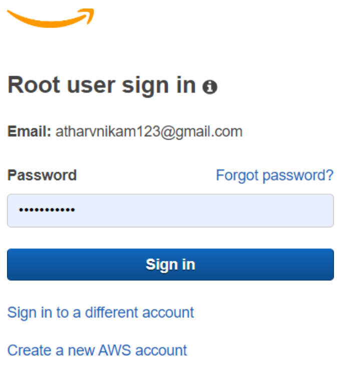

Amazon Simple Storage Service (S3)
Lambda
Amazon Simple Storage Service (S3)
Permissions Policy
AWS Management Console
Source bucket
Create thumbnail function
Destination bucket

Key Features and Applications

- **Seamless Automation**: Automatically resizes images using an event-driven approach.
- **Triggering Mechanism**: Lambda function is invoked immediately upon image upload to the designated S3 bucket.
- **Thumbnail Generation**: Processes images and saves smaller versions in a separate S3 bucket.
- **Optimized Image Delivery**: Ideal for websites, mobile apps, and CDNs, reducing bandwidth usage and improving page loading times.
- **Enhanced User Experience**: Faster loading times lead to better overall user experiences.
- **Scalability**: AWS Lambda automatically scales functions based on traffic without manual intervention.

Step 1: Initial Setup – Creating S3 Buckets

1. **AWS Account Sign-In**: Sign in as the root user.



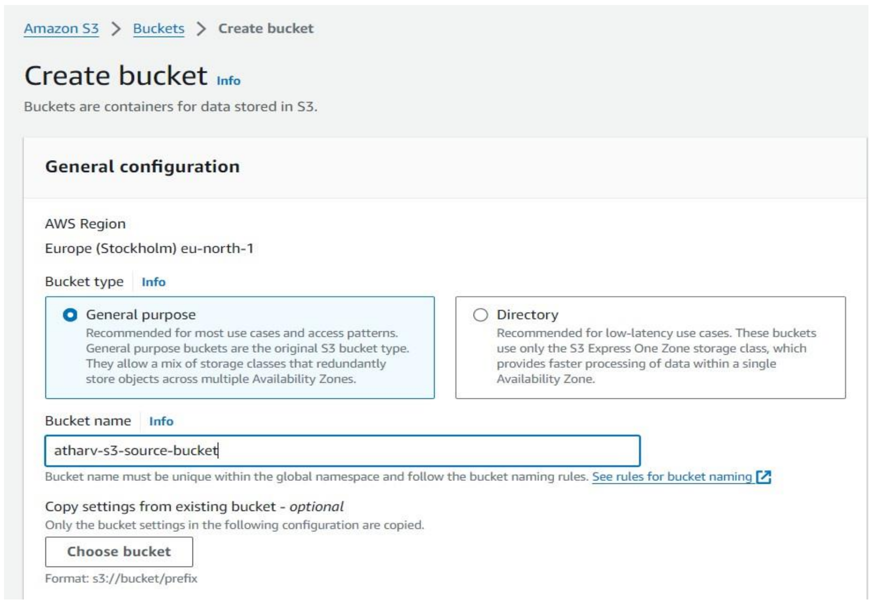
Root user sign in

Email: atharvnikam123@gmail.com

Password

•••••••••••

2. **Create Buckets**:
   - **Source Bucket**: amzn-s3-bbp-source-bucket
   - **Destination Bucket**: amzn-s3-bbp-source-bucket-resized



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

atharv-s3-source-bucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

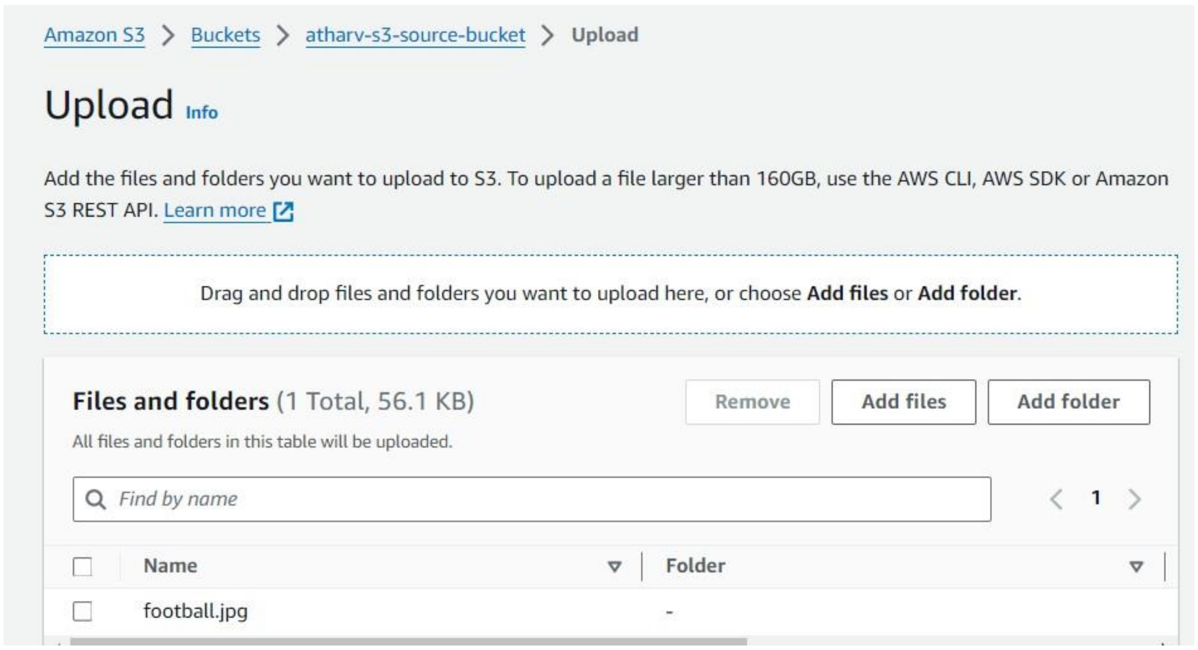
Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Upload a Test Image

1. Open the S3 console and select the source bucket.
2. Click **Upload**, add a JPG/PNG file, and 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**.

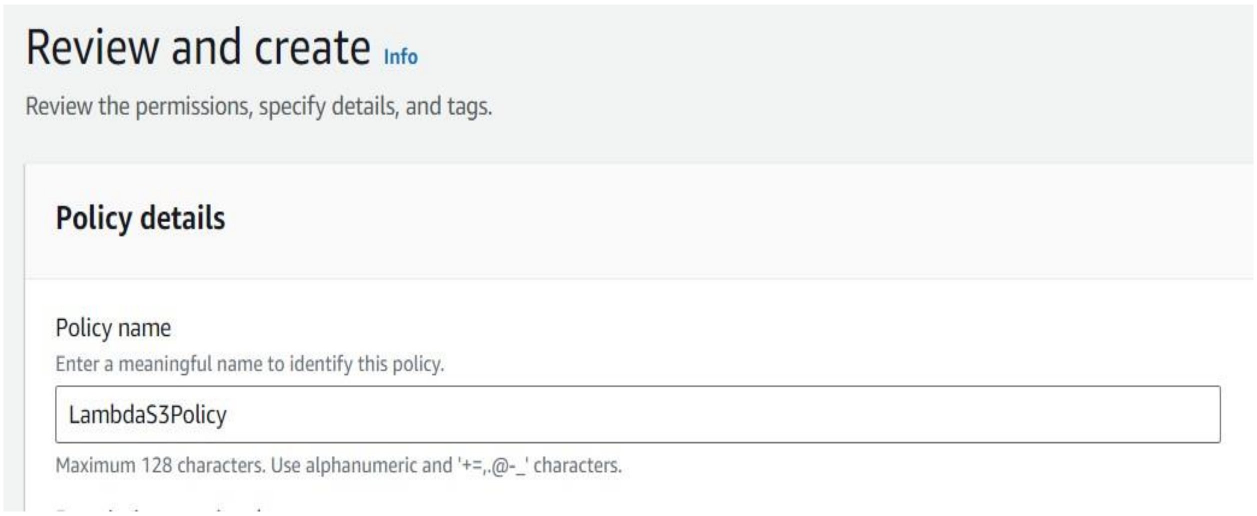
Files and folders (1 Total, 56.1 KB)

All files and folders in this table will be uploaded.

Name	Folder
football.jpg	-

Step 2: Permissions Setup

1. **Create IAM Policy**:
   - **Policy Name**: `LambdaS3Policy`


Review and create Info

Review the permissions, specify details, and tags.

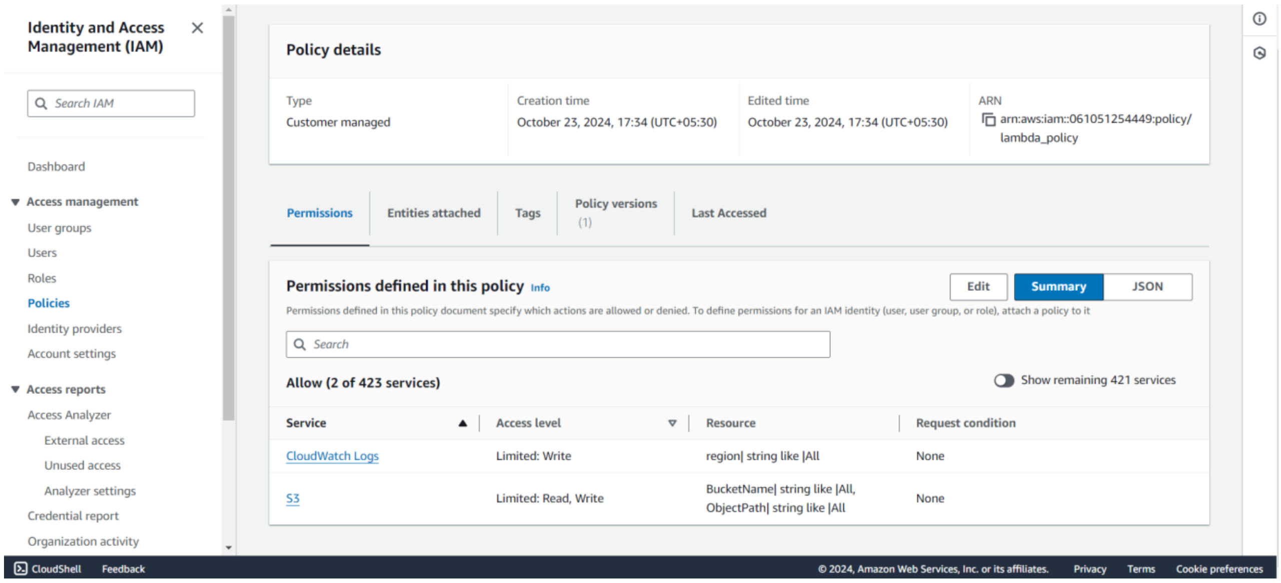
Policy details

Policy name

Enter a meaningful name to identify this policy.

LambdaS3Policy

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



Policy details

Type	Creation time	Edited time	ARN
Customer managed	October 23, 2024, 17:34 (UTC+05:30)	October 23, 2024, 17:34 (UTC+05:30)	arn:aws:iam::061051254449:policy/lambda_policy

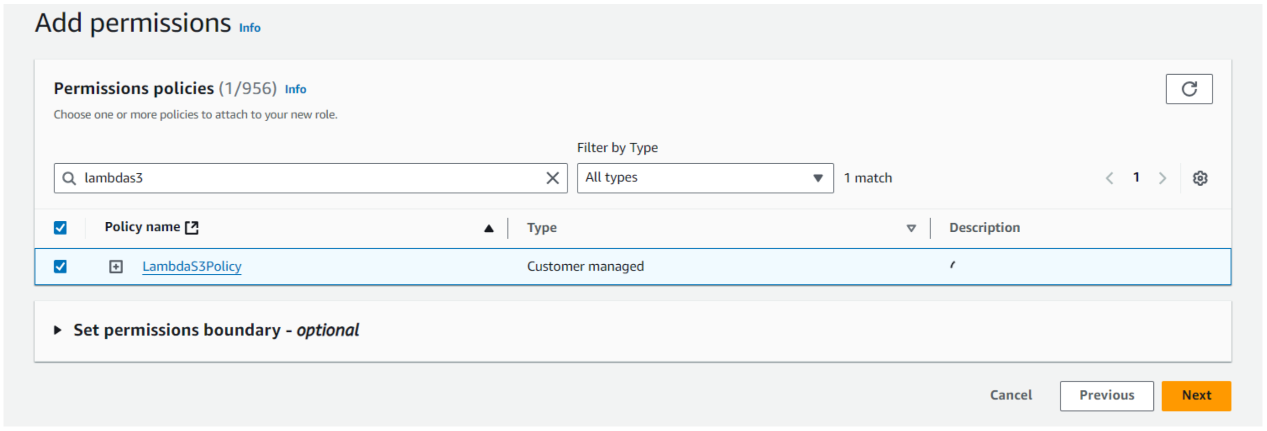
Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Allow (2 of 423 services)

Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	region| string like |All	None
S3	Limited: Read, Write	BucketName| string like |All, ObjectPath| string like |All	None

2. **Create Execution Role**:
   - Name: `LambdaS3Role`
   - Attach the `LambdaS3Policy` to it.



Add permissions Info

Permissions policies (1/956) Info

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

Filter by Type

lambdas3 | All types | 1 match

	Policy name	Type	Description
✓	LambdaS3Policy	Customer managed	

Set permissions boundary - *optional*

Step 3: Create Lambda Function

Lambda Function Code (`lambda_function.py`):

```
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))
```

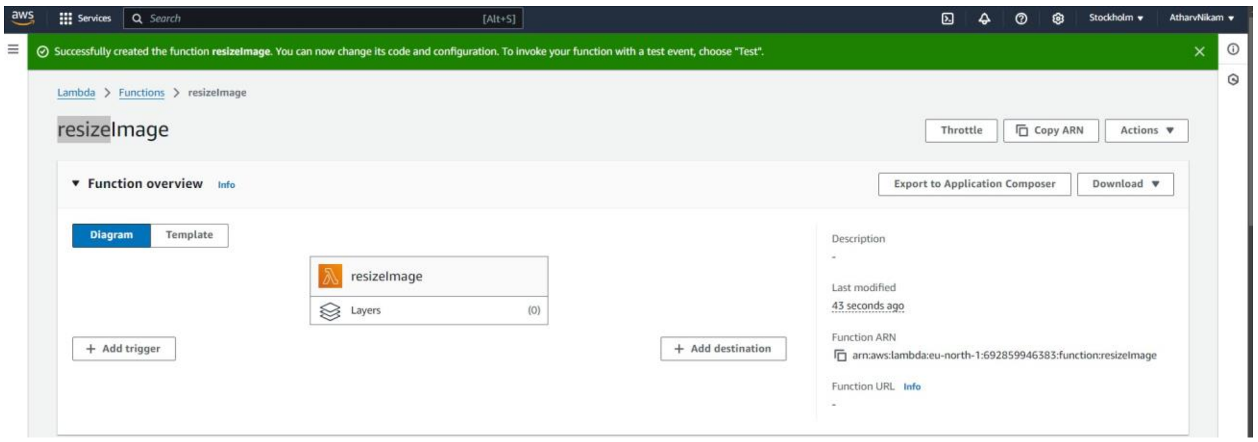
1. **Install Dependencies**:

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

2. **Package the Application**: Zip `lambda_function.py` and the `package` folder.

Step 4: Create the Lambda Function

1. **Access Lambda Console**: Go to the AWS Lambda console.
2. **Create Function**:
   - Name: `resizeImage`
   - Runtime: Python 3.12
   - Role: Select `LambdaS3Role`.
3. **Upload Code**: Upload your `.zip` file and save.



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

Lambda > Functions > resizeImage

resizeImage

Function overview

Description
-

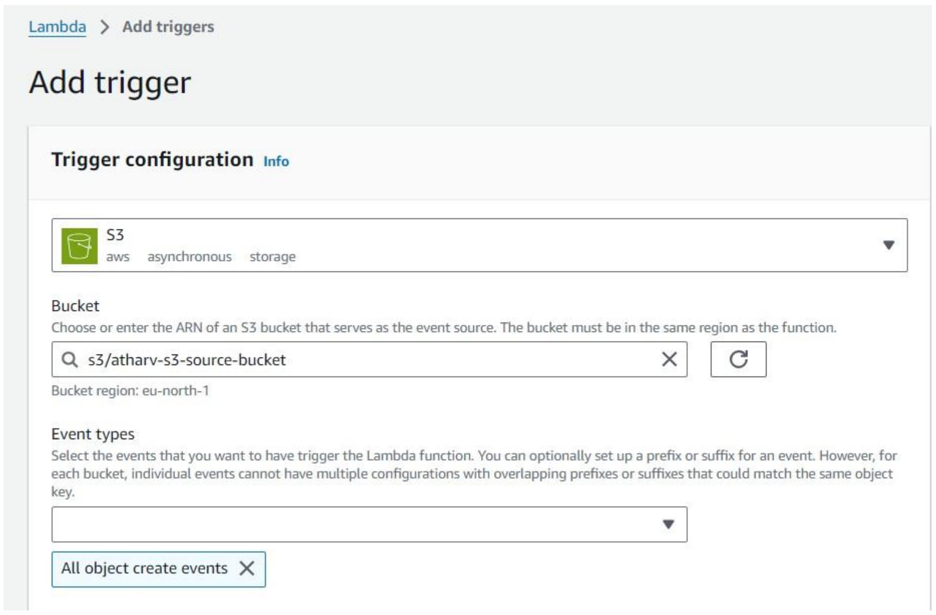
Last modified
43 seconds ago

Function ARN
arn:aws:lambda:eu-north-1:692859946383:function:resizeImage

Function URL
-

Step 5: Configure S3 Trigger

1. **Select Lambda Function**: Go to the `resizeImage` function.
2. **Add Trigger**: Choose S3, select the source bucket, and configure for "All object create events".



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

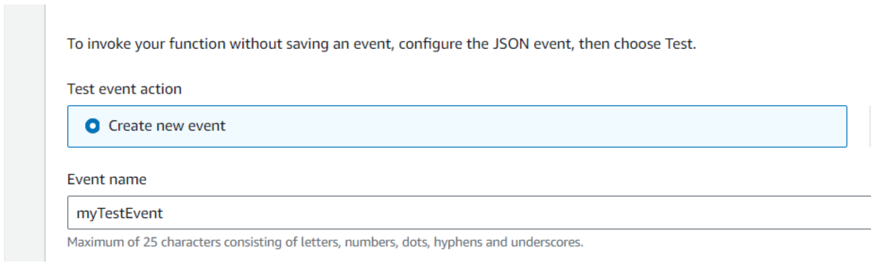
Bucket region: eu-north-1

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

Step 6: Test Lambda Function

1. **Create Test Event**:
   - Name: `myTestEvent`
   - Template: S3 Put.
   - Customize JSON with relevant parameters.
2. **Save and Test**: Click **Test** to verify functionality.
3. **Check Resized Image**: Look for the resized image in the destination bucket.



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

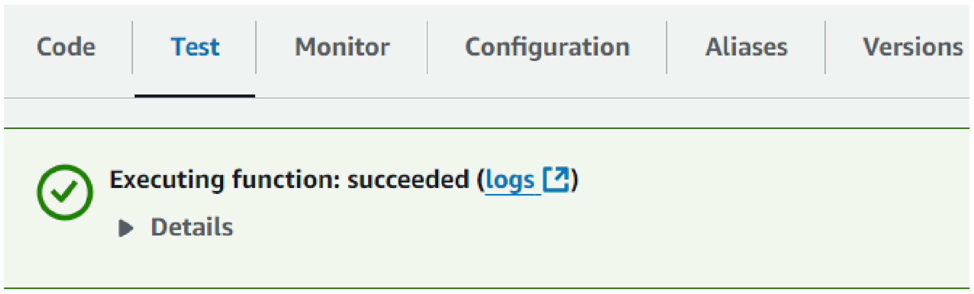
Test event action

Create new event

Event name

myTestEvent

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

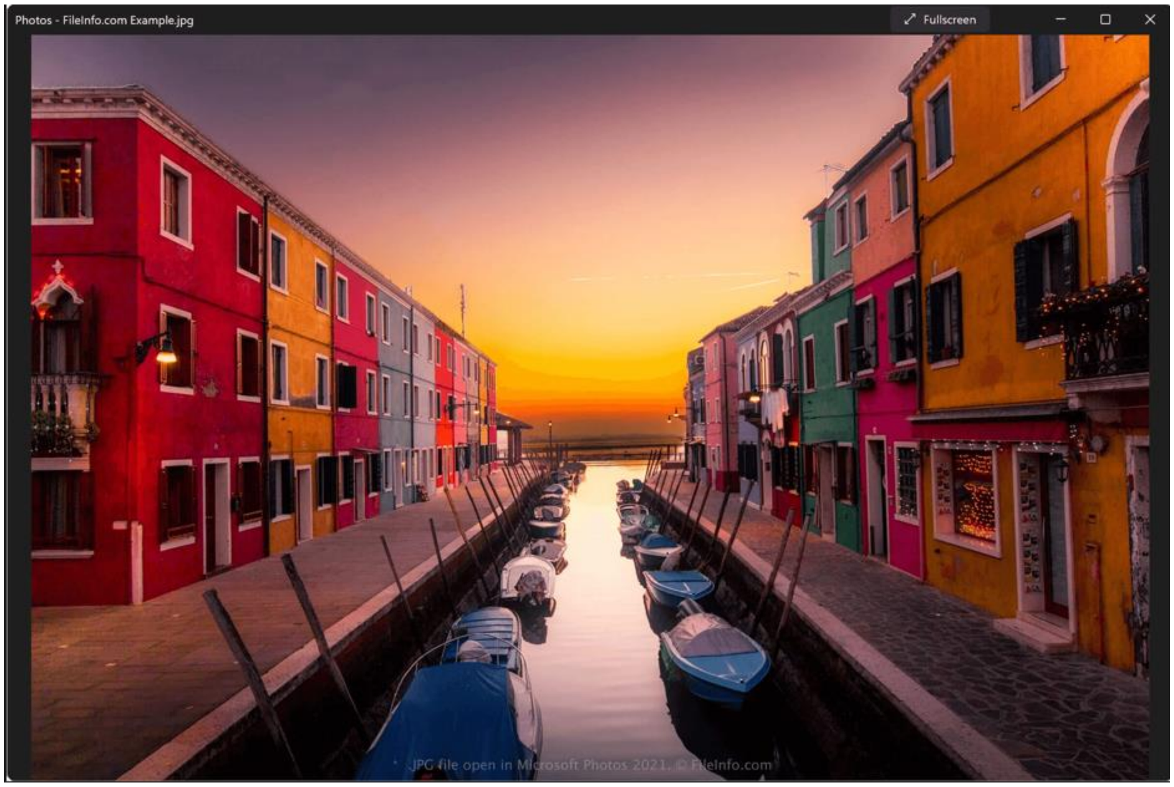


Executing function: succeeded (logs)

Details

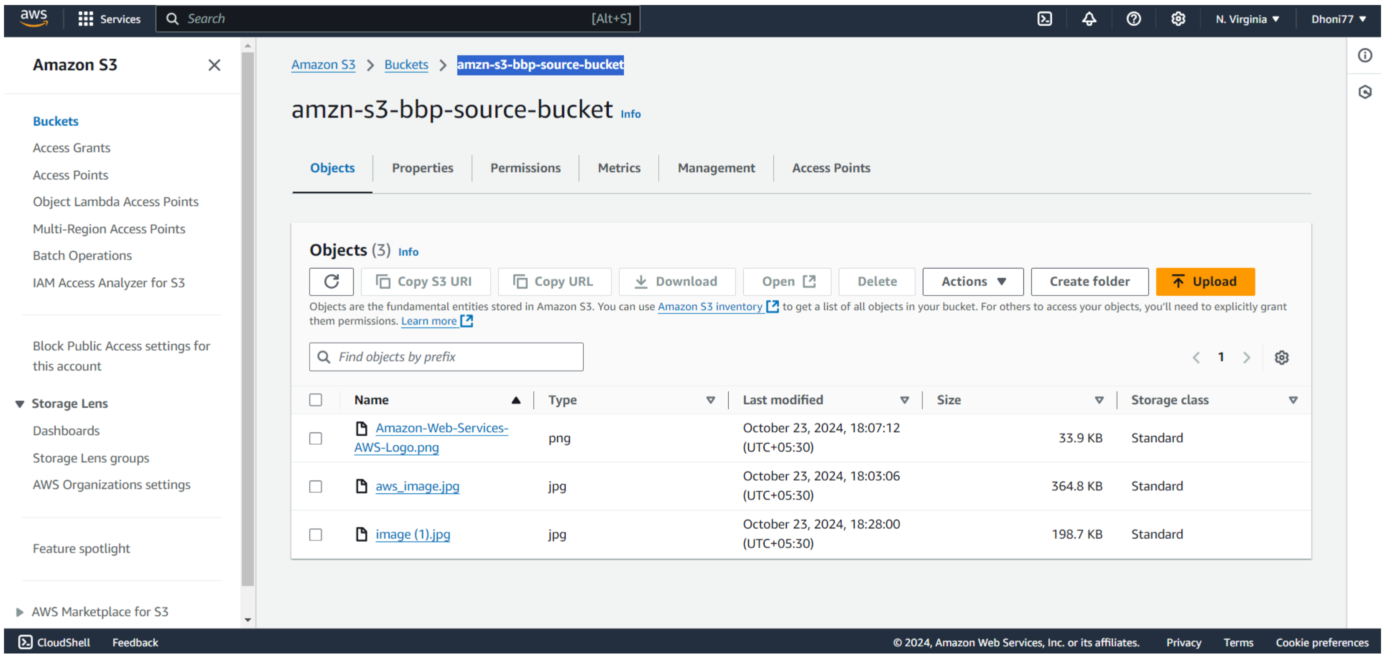
Step 7: Final Test

1. **Upload Image**: Upload an image to the source bucket to trigger the Lambda function.
2. **Verify Output**: Check the destination bucket for resized images.


Photos - FileInfo.com Example.jpg
Fullscreen
JPG file open in Microsoft Photos 2021. © FileInfo.com

Final Output in:- amzn-s3-bbp-source-bucket-resized

Photos - FileInfo.com Example.jpg
Fullscreen
JPG file open in Microsoft Photos 2021. © FileInfo.com

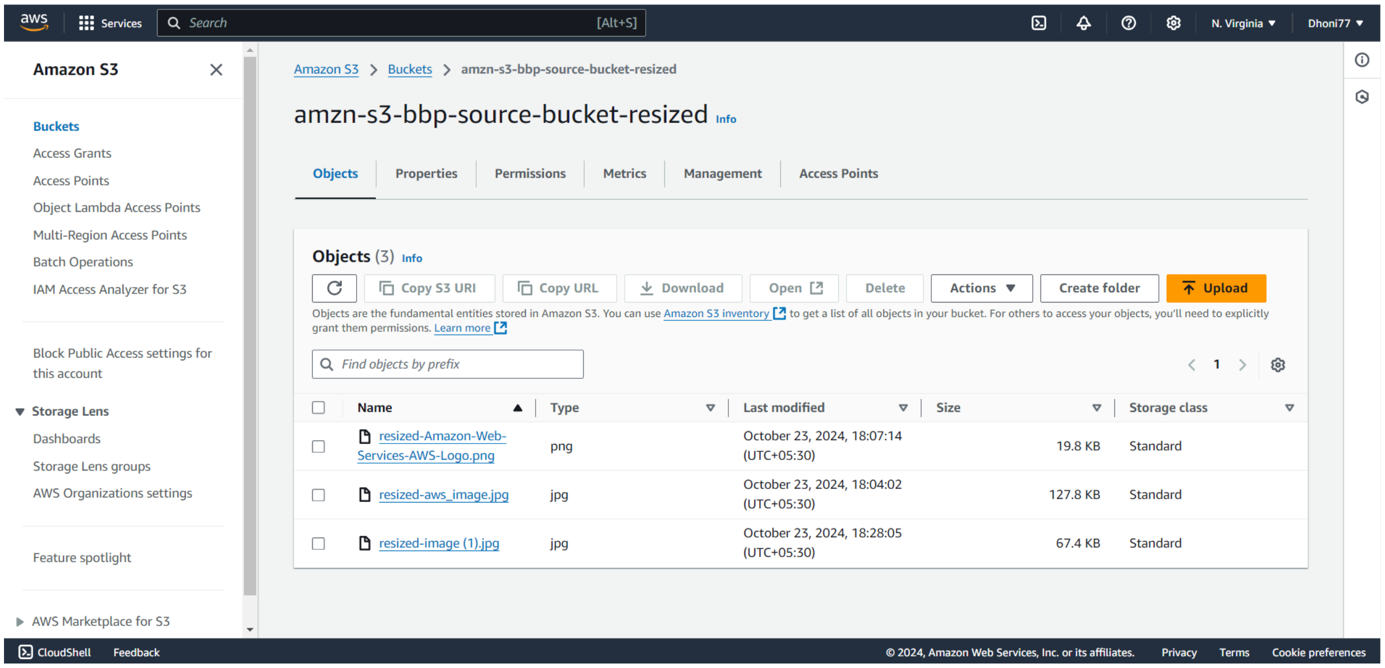


amzn-s3-bbp-source-bucket Info

Objects (3) Info

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

Name	Type	Last modified	Size	Storage class
Amazon-Web-Services-AWS-Logo.png	png	October 23, 2024, 18:07:12 (UTC+05:30)	33.9 KB	Standard
aws_image.jpg	jpg	October 23, 2024, 18:03:06 (UTC+05:30)	364.8 KB	Standard
image (1).jpg	jpg	October 23, 2024, 18:28:00 (UTC+05:30)	198.7 KB	Standard



amzn-s3-bbp-source-bucket-resized Info

Objects (3) Info

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

Name	Type	Last modified	Size	Storage class
resized-Amazon-Web-Services-AWS-Logo.png	png	October 23, 2024, 18:07:14 (UTC+05:30)	19.8 KB	Standard
resized-aws_image.jpg	jpg	October 23, 2024, 18:04:02 (UTC+05:30)	127.8 KB	Standard
resized-image (1).jpg	jpg	October 23, 2024, 18:28:05 (UTC+05:30)	67.4 KB	Standard

Conclusion: This project shows how AWS Lambda and S3 can automatically resize images without needing to manage any servers. The process is simple and efficient, with images being resized when uploaded and stored in a separate bucket. It improves loading times and works well for websites and apps. Hosting the frontend on Netlify makes the deployment easier, and overall, the system works smoothly for the intended purpose.

Bibliography

Aws documentation : https://docs.aws.amazon.com/lambda/latest/dg/with-s3-tutorial.html