

Experiment 12

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

- First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

- Set up a new Lambda function using AWS Lambda's console. You can choose a runtime environment like Python, Node.js, or Java.
- Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:

- Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

- Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

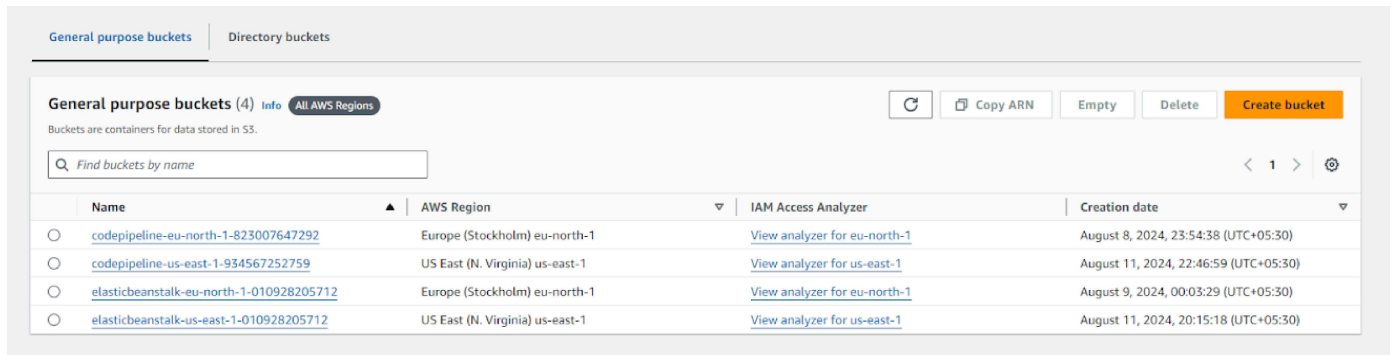
5. Test the Setup:

- Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Prerequisites: AWS Personal Account

Steps To create the lambda function:

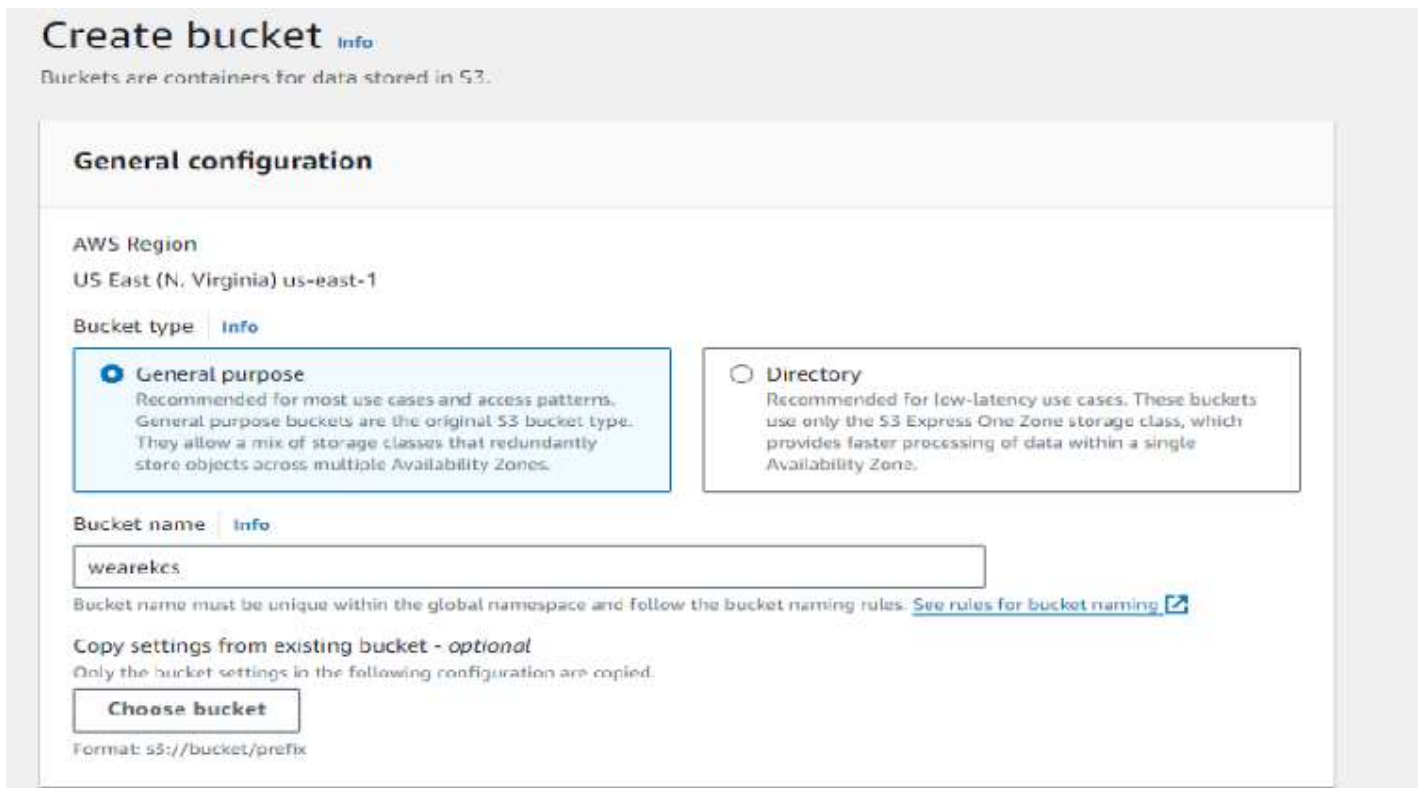
Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



The screenshot shows the AWS S3 console interface. At the top, there are tabs for "General purpose buckets" and "Directory buckets". Below the tabs, there's a header for "General purpose buckets (4)" with an "Info" link and a "All AWS Regions" button. A search bar labeled "Find buckets by name" is present. To the right of the search bar are buttons for "Copy ARN", "Empty", "Delete", and a prominent orange "Create bucket" button. Below these elements is a table listing four buckets. Each row includes a radio button, the bucket name, the AWS Region, a link to the IAM Access Analyzer, and the creation date.

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	codepipeline-eu-north-1-823007647292	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 8, 2024, 23:54:38 (UTC+05:30)
<input type="radio"/>	codepipeline-us-east-1-934567252759	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 11, 2024, 22:46:59 (UTC+05:30)
<input type="radio"/>	elasticbeanstalk-eu-north-1-010928205712	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 9, 2024, 00:03:29 (UTC+05:30)
<input type="radio"/>	elasticbeanstalk-us-east-1-010928205712	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 11, 2024, 20:15:18 (UTC+05:30)

Step 2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



The screenshot shows the "Create bucket" wizard in the AWS S3 console. The title is "Create bucket" with an "Info" link. Below the title, it says "Buckets are containers for data stored in S3." The main section is titled "General configuration". Under "AWS Region", "US East (N. Virginia) us-east-1" is selected. The "Bucket type" section has two options: "General purpose" (selected with a radio button) and "Directory". The "General purpose" option is described as recommended for most use cases and access patterns. Below the bucket type selection, there is a "Bucket name" field with the text "wearekes" entered. A note states that the bucket name must be unique and follow naming rules, with a link to "See rules for bucket naming". There is also a section for "Copy settings from existing bucket - optional" with a "Choose bucket" button and a format example: "Format: s3://bucket/prefix".

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☐ Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

☐ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Successfully created bucket "example"

To upload files and folders, or to configure additional bucket settings, choose View details.

View details

Buckets

Account snapshot - updated every 24 hours

View Storage Lens dashboard

General purpose buckets

Directory buckets

General purpose buckets (1)

All AWS Regions

Copy ARN

Empty

Delete

Create bucket

Find buckets by name

1

10M Access Analytics

Creation date

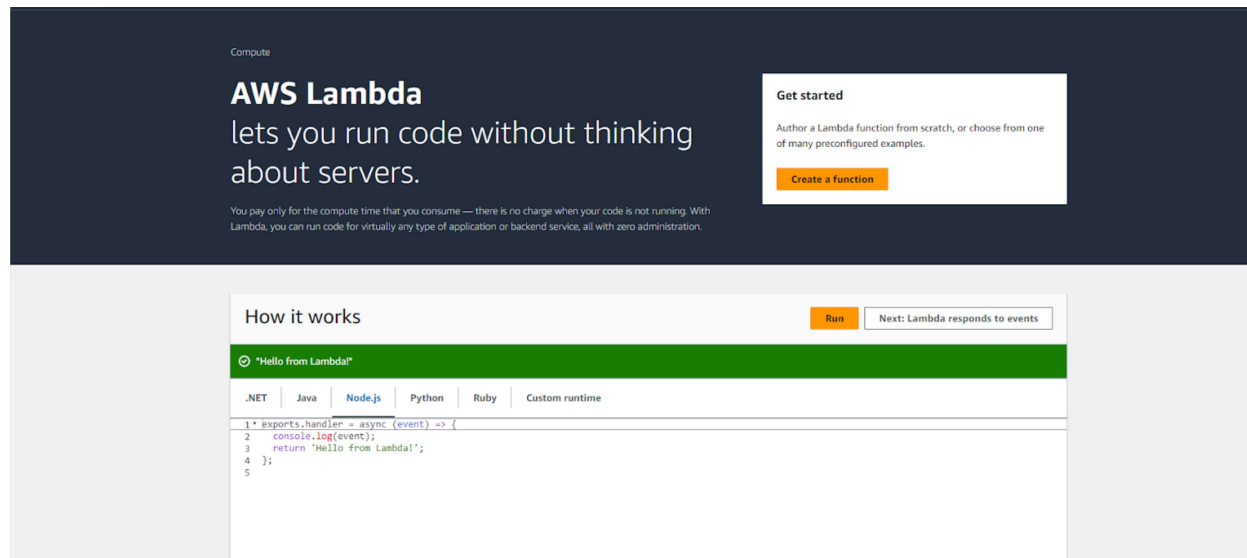
bucket

US East (N. Virginia) us-east-1

View details for us-east-1

October 1, 2024, 13:40:40 (UTC+05:30)

Step 3: Open lambda console and click on create function button.



Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12
, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.

[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.

MyLambda

Use only letters, numbers, hyphens, or underscores with no spaces.

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

Python 3.12

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64

☐ arm64

Python 3.12

↻

Architecture

Info

Choose the instruction set architecture you want for your function code.

☒ x86_64

☐ arm64

Permissions

Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ 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

❗

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named `Bhushan_Lambda-role-pbjr1991`, with permission to upload logs to Amazon CloudWatch Logs.

► Advanced settings

Cancel

Create function

CloudShell

Feedback

Lambda > Functions > myLambda

myLambda

Throttle

Copy ARN

Actions

▼ Function overview

Info

Export to Application Composer

Download

Diagram

Template

myLambda

Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

4 minutes ago

Function ARN

arn:aws:lambda:eu-north-1:860015268757:function:myLambda

Function URL

Info

-

Code

Test

Monitor

Configuration

Aliases

Versions

Code source

Info

Upload from

File

Edit

Find

View

Go

Tools

Window

Test

Deploy

Environment

myLambda

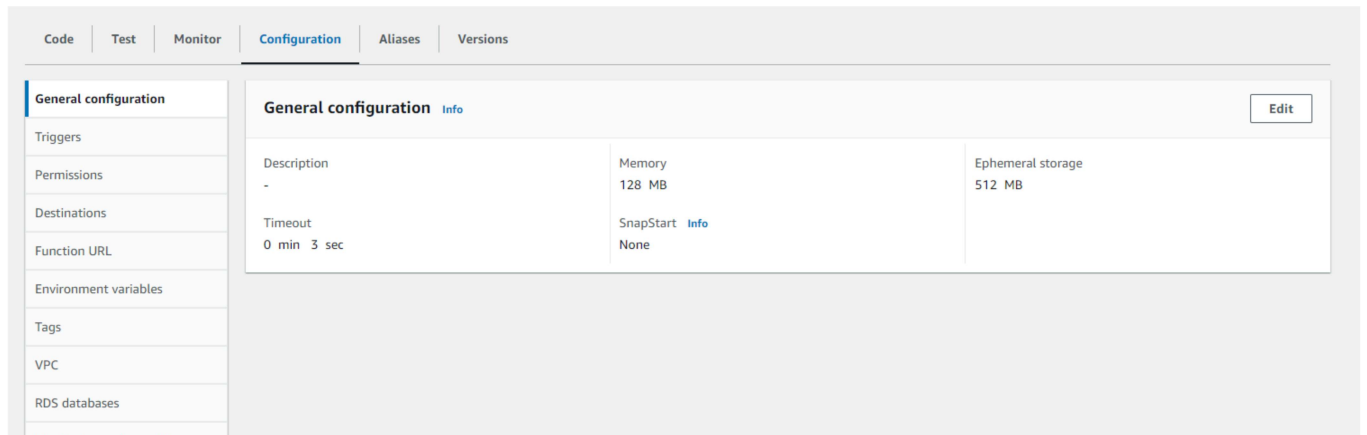
lambda_function.py

lambda_function

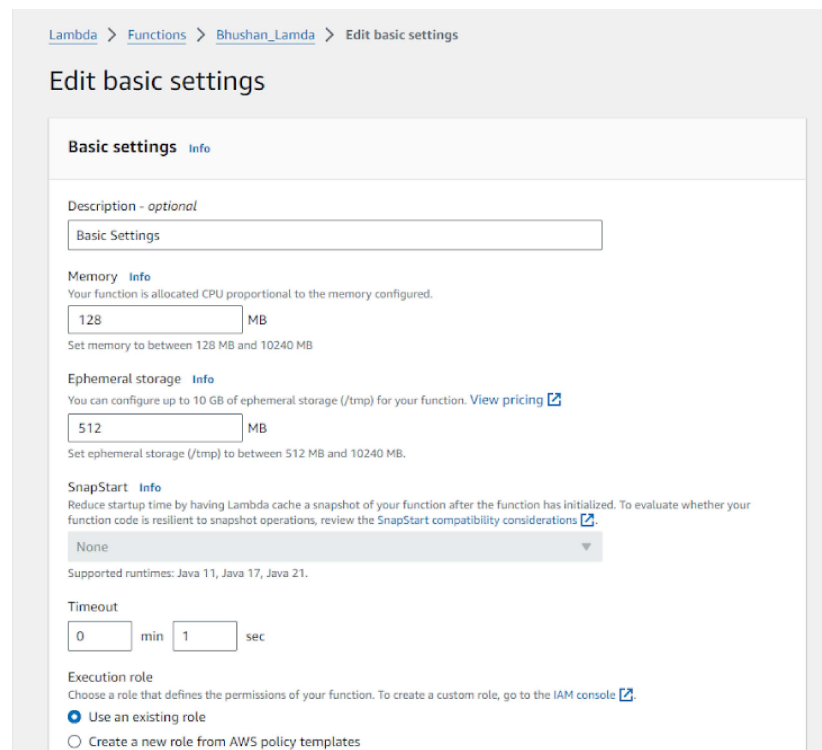
Environment Vari

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO Implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')}
8
9
```

So See or Edit the basic settings go to configuration then click on edit general setting.



Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

Test event [Info](#)

SaveTest

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

Test event action

☒ Create new event

☐ Edit saved event

Event name

MyEvent

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](#)

Template - optional

hello-world

Event JSON

Format JSON

```
1 {
2   "key1": "value1",
3   "key2": "value2",
```

Services [Alt+S]

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

Template - optional

s3-put

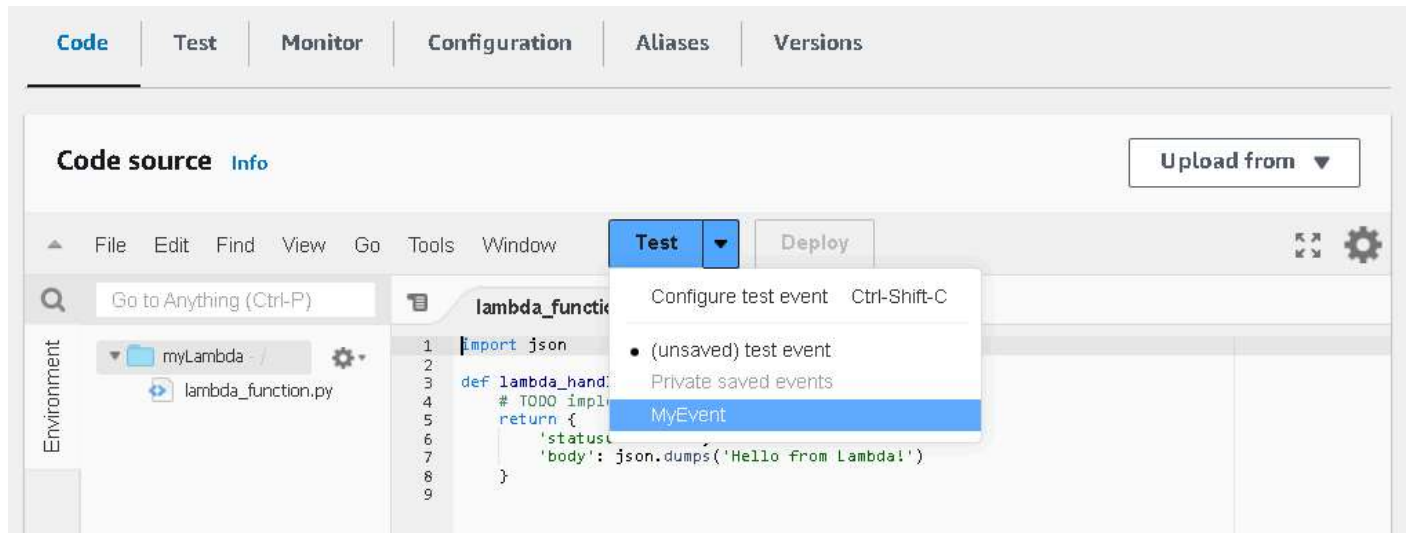
Event JSON

Format JSON

```
2 * "Records": [
3 *   {
4     "eventVersion": "2.0",
5     "eventSource": "aws:s3",
6     "awsRegion": "us-east-1",
7     "eventTime": "1970-01-01T00:00:00.000Z",
8     "eventName": "ObjectCreated:Put",
9     "userIdentity": {
10      "principalId": "EXAMPLE"
11    },
12    "requestParameters": {
13      "sourceIPAddress": "127.0.0.1"
14    },
15    "responseElements": {
16      "x-amz-request-id": "EXAMPLE123456789",
17      "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaisawsome/mnopqrstuvwxyzABCDEFGH"
18    },
19    "s3": {
20      "s3SchemaVersion": "1.0",
21      "configurationId": "testConfigRule",
22      "bucket": {
23        "name": "example-bucket",
24        "ownerIdentity": {
25          "principalId": "EXAMPLE"
26        },
27        "arn": "arn:aws:s3:::example-bucket"
28      },
29      "object": {
30        "key": "test%2Fkey",
31        "size": 1024,
```

1:1 JSON Spaces: 2

Step 6: Now In Code section select the created event from the dropdown .



Step 7: Now In the Lambda function click on add trigger.




Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.

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

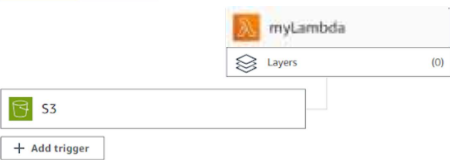
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.


Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.

Recursive invocation

Function overview Info

Diagram Template



 **S3**

+ Add trigger

Export to Application Composer

Download

Description
Basic Settings
Last modified
1 hour ago
Function ARN
arn:aws:lambda:us-east-1:010928205712:function:Bhushan_Lambda
Function URL [Info](#)

Code Test Monitor **Configurations** Aliases Versions

General configuration

Triggers

Access logs

Permissions

Function URL

Environment variables

Tags

VPC

ROS definitions

Monitoring and operations tools

Concurrency and resource detection


Asynchronous invocation

Logging

API gateway

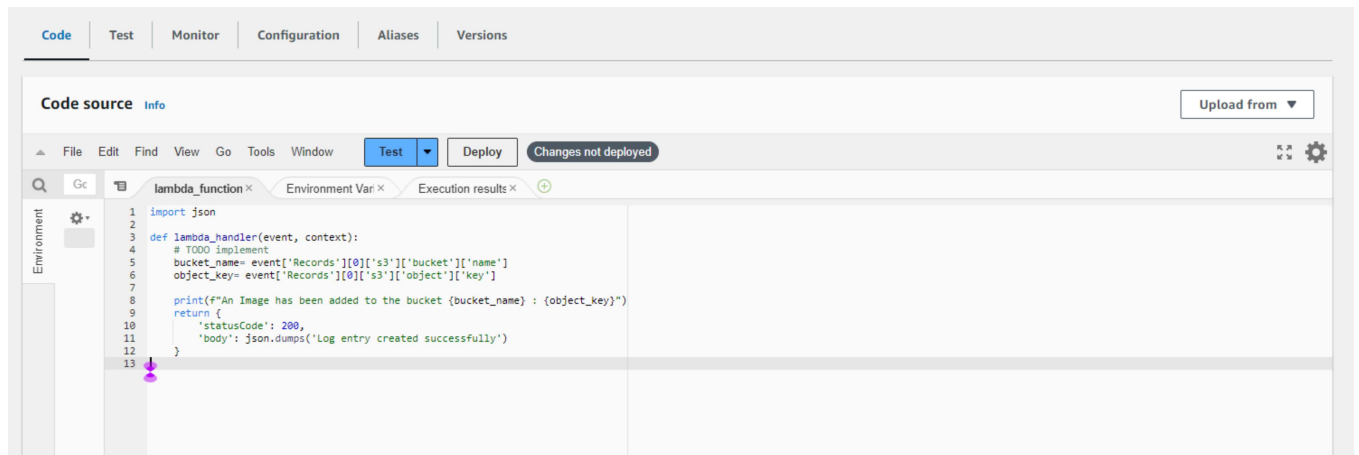
Code insights

Triggers (1) Info

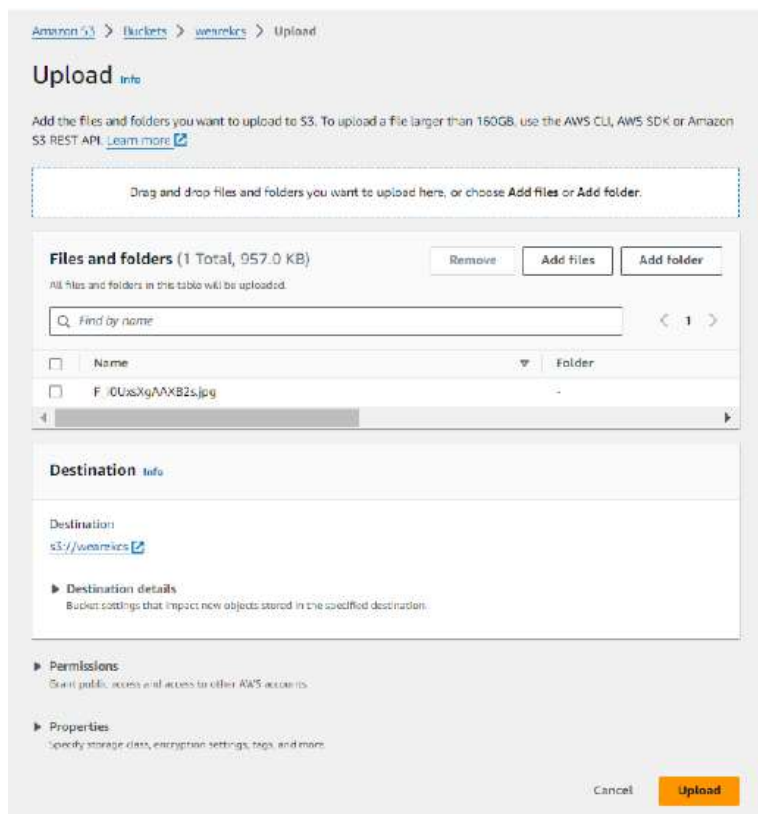
 **S3** warankcs
aws (function)

+ Details

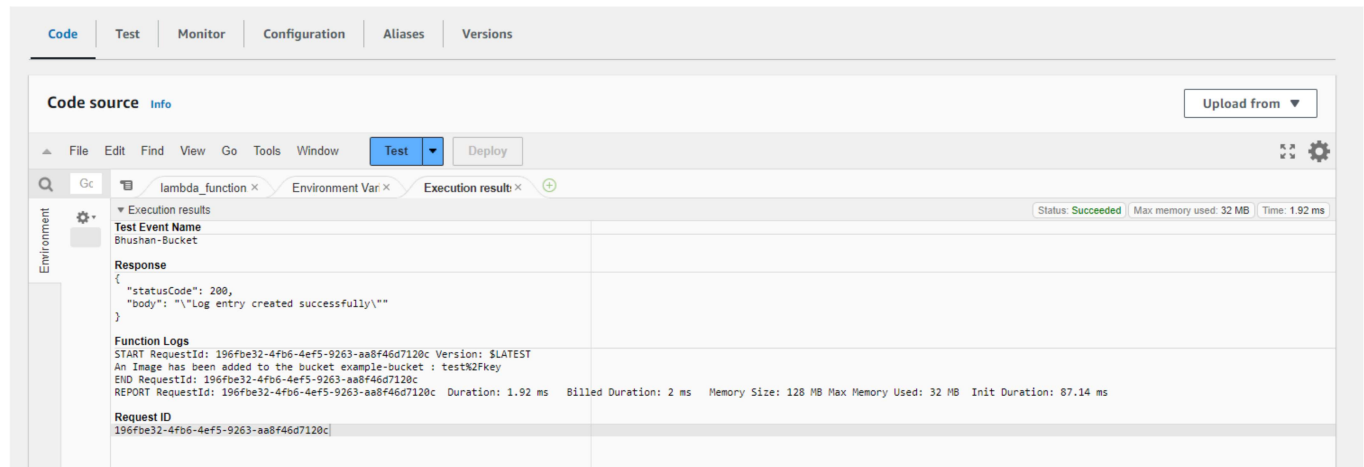
Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.



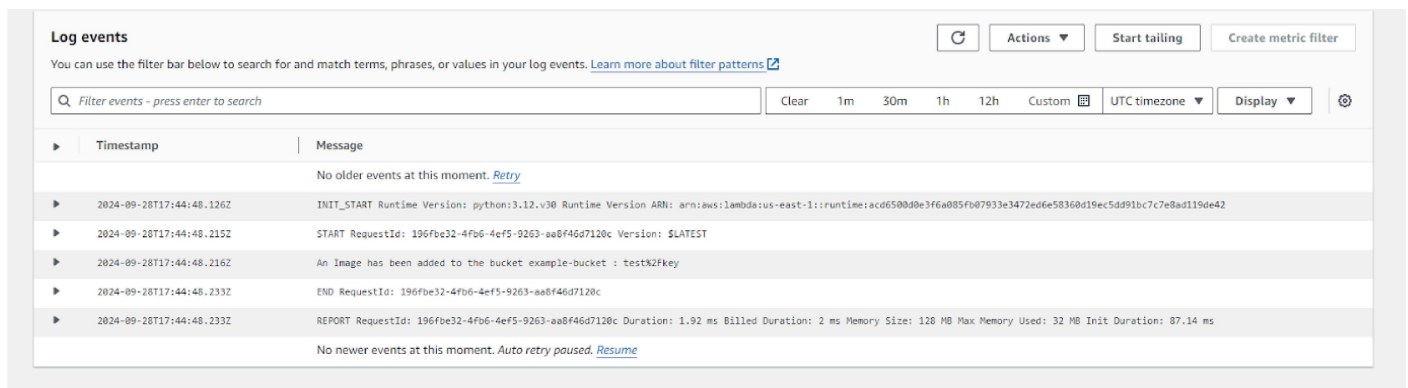
Step 9: Now upload any image to the bucket.



Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.



Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



Conclusion:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It is important to note that we have to select S3-put template in the event otherwise code will give an error. The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.