

Experiment No :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

CREATING LAMBDA FUNCTION :

Step 1: Log in to your AWS Personal account. Then go to S3 in the services menu and click on "Create S3 Bucket."

The image shows two screenshots from the AWS Management Console. The top screenshot shows the search results for 's3' in the AWS Services menu, highlighting 'S3' (Scalable Storage in the Cloud) and 'S3 Glacier' (Archive Storage in the Cloud). The bottom screenshot shows the Amazon S3 console interface. On the left, there is a navigation menu with options like Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings, Storage Lens, Dashboards, Storage Lens groups, and AWS Organizations settings. The main content area shows the 'General purpose buckets' tab selected. At the top, there is an 'Account snapshot' section. Below that, there are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. A search bar 'Find buckets by name' is present. A table lists the buckets:

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	elasticbeanstalk-us-east-1-073011525842	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 8, 2024, 22:36:14 (UTC+05:30)

Step 2: Give your bucket a name, select "General purpose project," then uncheck "Block public access." Keep the other settings as they are.

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](#)

exp12sandesh

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](#)

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.



Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

- ☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Successfully created bucket "exp12sandesh"
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

Amazon S3 > Buckets

Account snapshot - updated every 24 hours

View Storage Lens dashboard

General purpose buckets

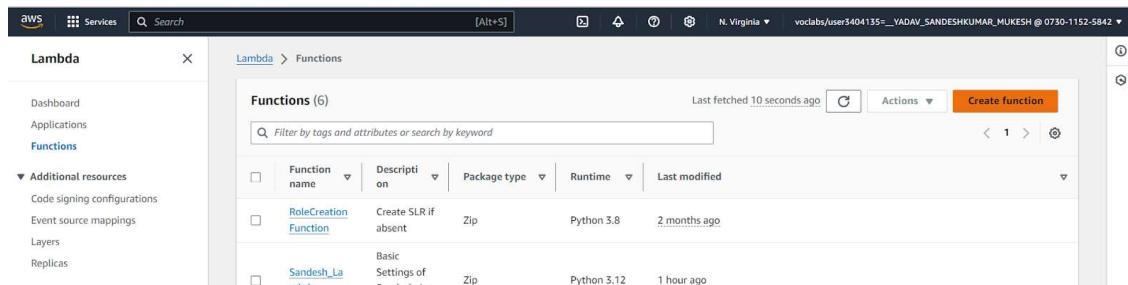
Directory buckets

General purpose buckets (3)

Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-073011525842	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 8, 2024, 22:36:14 (UTC+05:30)
exp12sandesh	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 10, 2024, 20:30:38 (UTC+05:30)
website-for-video	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 26, 2024, 20:03:00 (UTC+05:30)

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



Step 4: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen Python 3.12. Set the architecture to x86. For the execution role, select 'Use an existing role,' then pick 'Lab role' from the dropdown menu under existing roles .

(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)

[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

▼ 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.

LabRole ▼

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

Successfully created the function **Lab_12_Sandesh**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > Lab_12_Sandesh

Lab_12_Sandesh

Throttle Copy ARN Actions ▼

▼ Function overview Info

Export to Application Composer Download ▼

Diagram Template

Lab_12_Sandesh

Layers (0)

+ Add trigger + Add destination

Description

Last modified in 2 seconds

Function ARN
arn:aws:lambda:us-east-1:073011525842:function:Lab_12_Sandesh

Function URL [Info](#)

Step 5: To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)

Code	Test	Monitor	Configuration	Aliases	Versions
------	------	---------	---------------	---------	----------

General configuration

Triggers

Permissions

Destinations

Function URL

General configuration Info

Edit

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart Info	
0 min 3 sec	None	

Basic settings Info

Description - optional

Memory Info

Your function is allocated CPU proportional to the memory configured.

 MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage Info

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

 MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#).

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout

 min 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.

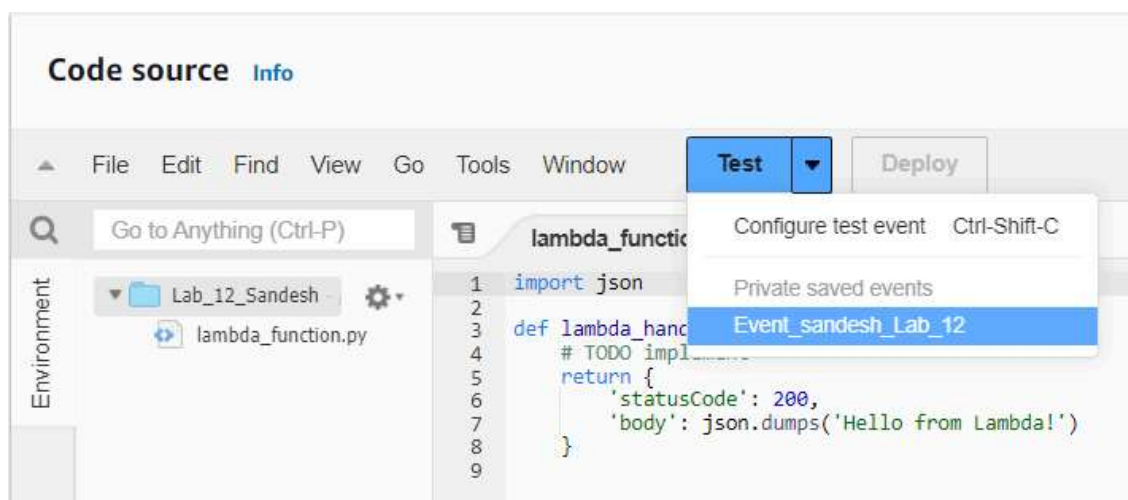
[View the LabRole role](#) on the IAM console.

Step 6: Click on the "Test" tab, then select "Create a new event." Give the event a name, set "Event Sharing" to private, and choose the "S3 Put" template. S3 (Simple Storage Service) template allows you to test your Lambda function specifically for events related to uploading files to an S3 bucket.

The screenshot shows the AWS Lambda console's 'Test' tab for a function. At the top, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Below these, there's a 'Test event' section with 'Save' and 'Test' buttons. A message states: 'To invoke your function without saving an event, configure the JSON event, then choose Test.' Under 'Test event action', 'Create new event' is selected. The 'Event name' field contains 'Event_sandesh_Lab_12'. Under 'Event sharing settings', 'Private' is selected. A 'Template - optional' dropdown shows 's3-put'. The 'Event JSON' section displays a JSON object for an S3 Put event, with a 'Format JSON' button. The JSON includes details like eventVersion, eventSource, awsRegion, eventName, and s3 bucket information.

```
1 {
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": "EXAMPLE123/5678abcdefghijklambdaisawesome/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",
```

Step 7: Now In the Code section select the created event from the dropdown .



Step 8: In the Lambda function, click on "Add Trigger." Adding a trigger allows your Lambda function to automatically run in response to specific events such as uploads to an S3 bucket

The screenshot shows the 'Function overview' page for a Lambda function named 'Lab_12_Sandesh'. The 'Diagram' tab is selected, showing a visual representation of the function with its layers. To the right, the 'Description' section provides details: 'Basic_Setting_Lab_12_Sandesh', 'Last modified 6 minutes ago', 'Function ARN: arn:aws:lambda:us-east-1:073011525842:function:Lab_12_Sandesh', and 'Function URL'. Buttons for 'Export to Application Composer' and 'Download' are at the top right. '+ Add trigger' and '+ Add destination' buttons are located below the diagram.

The screenshot shows the 'Add trigger' configuration page. The breadcrumb 'Lambda > Add triggers' is at the top. The main heading is 'Add trigger'. Below it, the 'Trigger configuration' section has an 'Info' link. A dropdown menu is labeled 'Select a source'. At the bottom right, there are 'Cancel' and 'Add' buttons.

Now select the source as S3, then choose the bucket name from the dropdown menu. Keep the other settings as default, and you can also add a prefix for the image if you want. A prefix for an image (or any file) in S3 is a string that you can use to organize or filter files within a bucket. It acts like a folder name, helping to categorize your files.

This screenshot shows the 'Add trigger' configuration page with 'S3' selected in the 'Select a source' dropdown menu. The dropdown menu is open, showing 'S3' with the AWS logo and the text 'aws asynchronous storage'. The 'Trigger configuration' section header and 'Info' link are visible above the dropdown.

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



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

If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn 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

Lab_12_Sandesh Throttle Copy ARN Actions

✓ The trigger exp12sandesh was successfully added to function Lab_12_Sandesh. The function is now receiving events from the trigger.

▼ **Function overview** Info Export to Application Composer Download

Diagram Template

Lab_12_Sandesh

Layers (0)

S3

+ Add trigger

+ Add destination

Description
Basic_Setting_Lab_12_Sandesh

Last modified
11 minutes ago

Function ARN
arn:aws:lambda:us-east-1:073011525842:function:Lab_12_Sandesh

Function URL [Info](#)

Code Test Monitor **Configuration** Aliases Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Triggers (1) Info Fix errors Edit Delete Add trigger

Find triggers

Trigger

S3: exp12sandesh
arn:aws:s3::exp12sandesh
[Details](#)

Step 9: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

```
import json
def lambda_handler(event, context):
    # TODO implement
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']
    print(f"An image has been added to the bucket {bucket_name}: {object_key}")
    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```

Code source Info

File Edit Find View Go Tools Window Test Deploy Changes not deployed

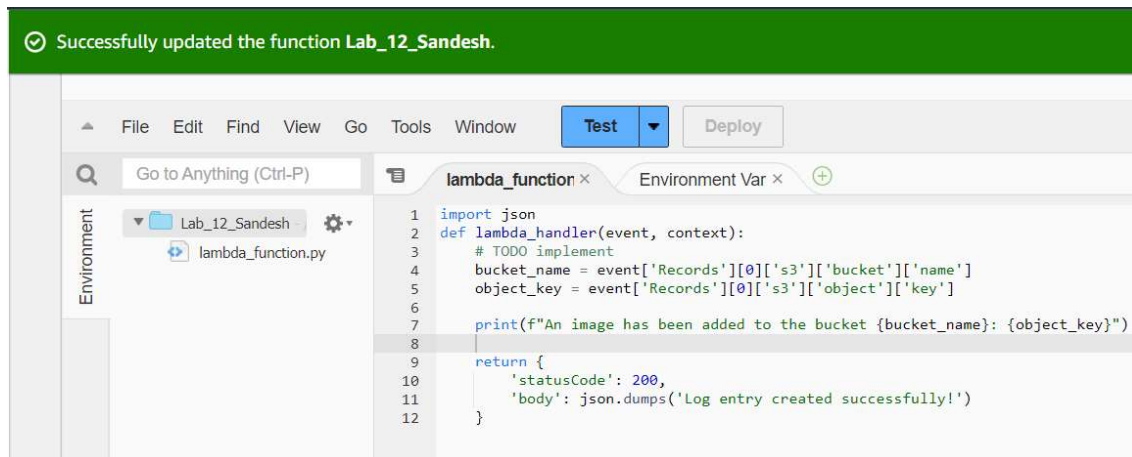
Go to Anything (Ctrl-P)

Environment

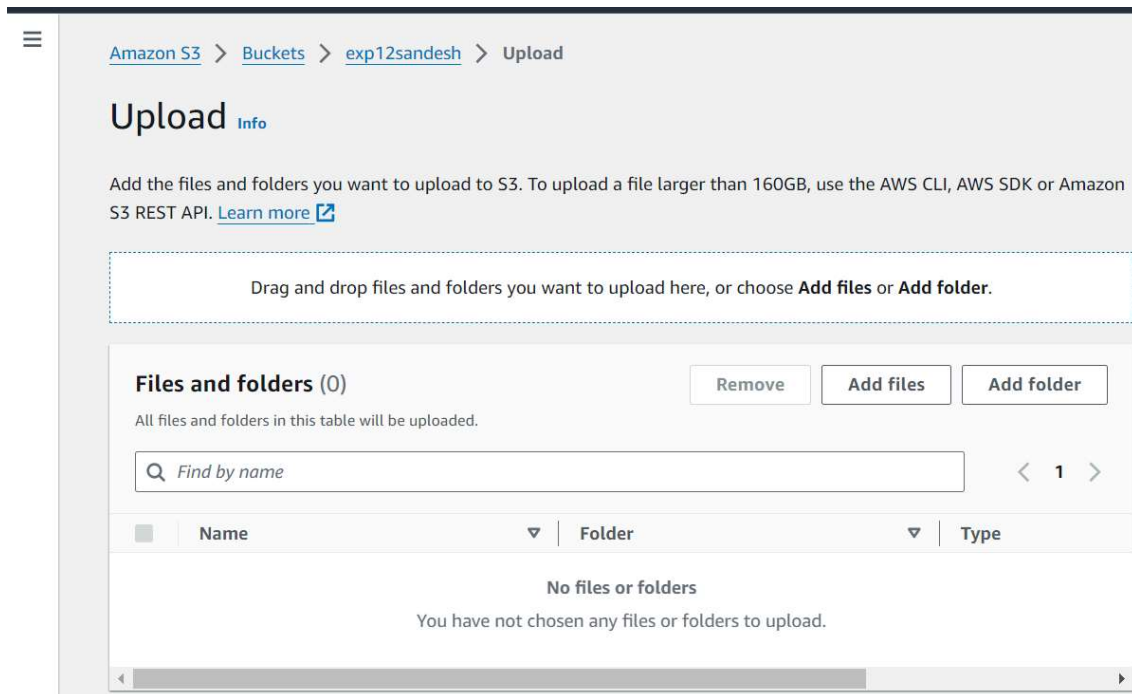
Lab_12_Sandesh

lambda_function.py

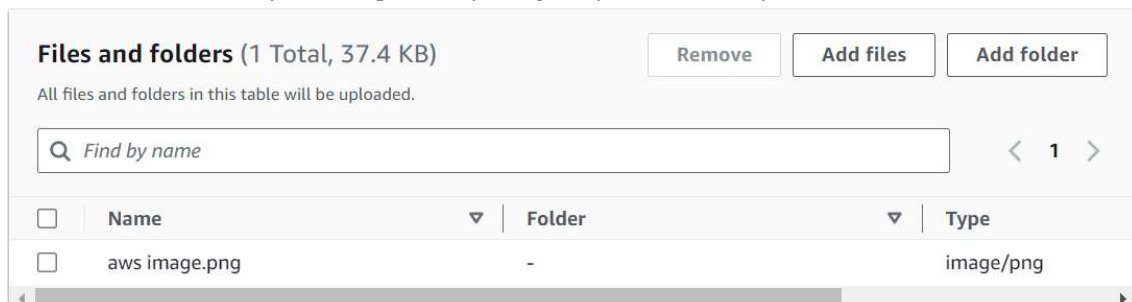
```
1 import json
2 def lambda_handler(event, context):
3     # TODO implement
4     bucket_name = event['Records'][0]['s3']['bucket']['name']
5     object_key = event['Records'][0]['s3']['object']['key']
6     print(f"An image has been added to the bucket {bucket_name}: {object_key}")
7     return {
8         'statusCode': 200,
9         'body': json.dumps('Log entry created successfully!')
10    }
```



Step 10: Now we will upload any image to the bucket

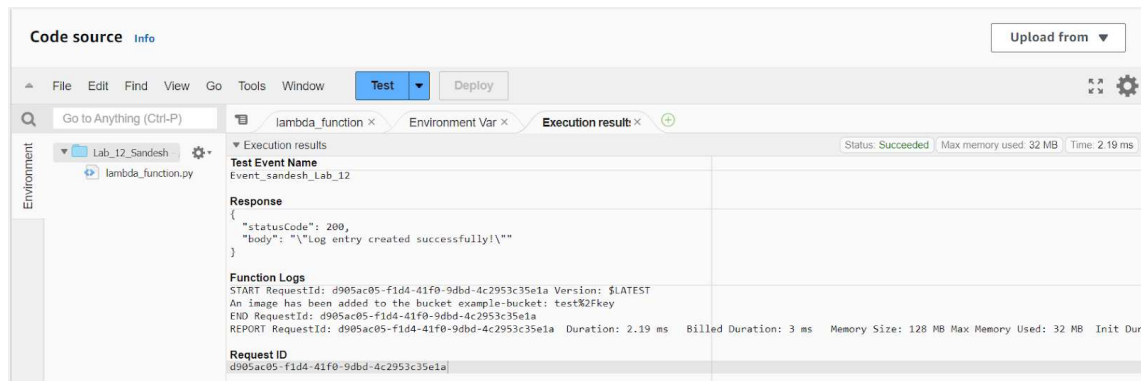


Click on add file where you can upload any image of your choice in your bucket

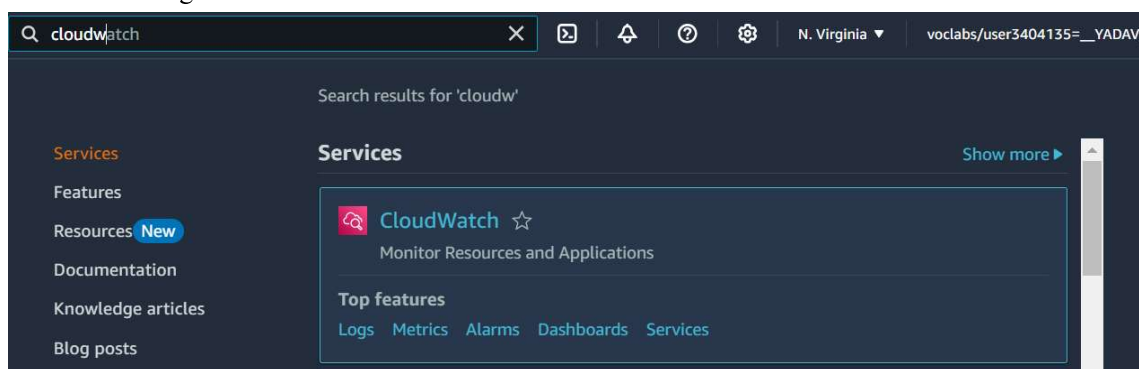


Click on Upload

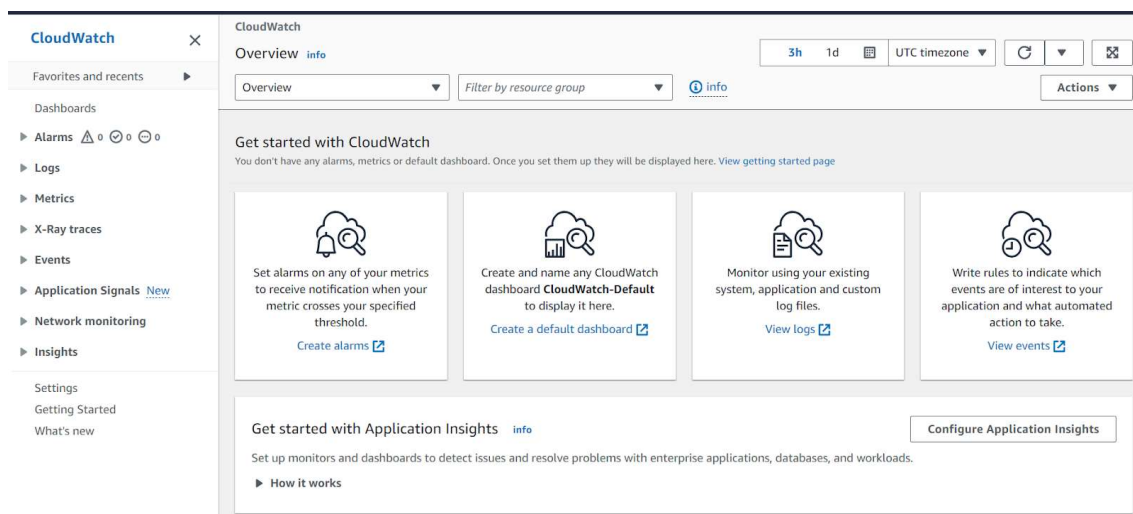
Step 11: Now click on "Test" in Lambda to see if it logs the activity when an image is added to S3.



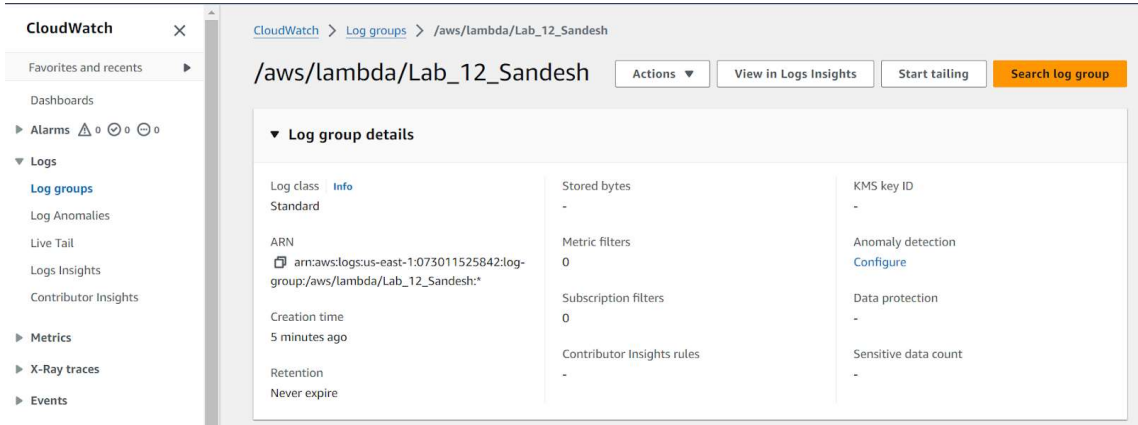
Step 12: Now let's check the logs on CloudWatch. Go to the "Monitor" section and click on "View CloudWatch Logs".



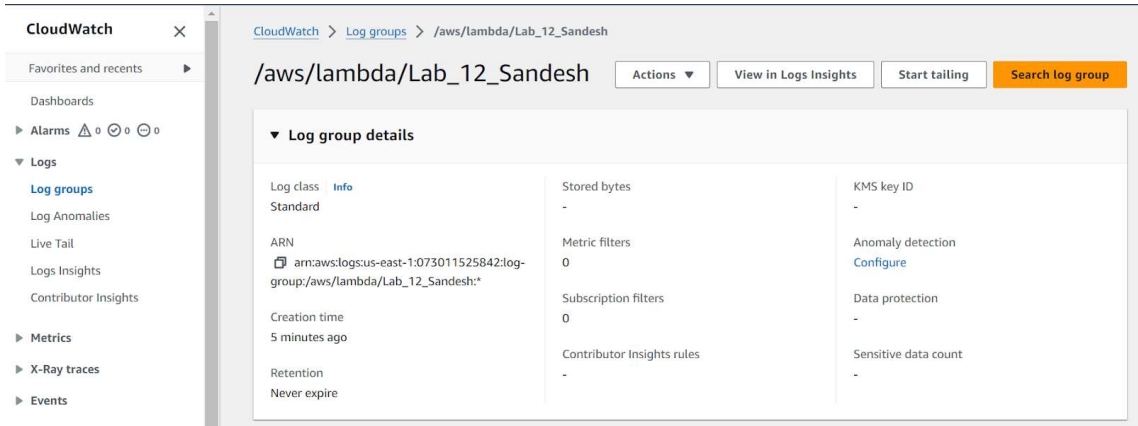
Click on the CloudWatch:



Click on the logs:



Click on the log group:



[CloudWatch](#) > [Log groups](#) > [/aws/lambda/Lab_12_Sandesh](#) > 2024-10-10/[\$LATEST]0568a57e192042548026f753de6d3f56

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

1m
1h
Calendar icon
UTC timezone ▼
Display ▼
Settings icon

▶	Timestamp	Message
No older events at this moment. Retry		
▶	2024-10-10T15:29:36.090Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e27_
▶	2024-10-10T15:29:36.183Z	START RequestId: d905ac05-f1d4-41f0-9dbd-4c2953c35e1a Version: \$LATEST
▶	2024-10-10T15:29:36.184Z	An image has been added to the bucket example-bucket: test%2Fkey
▶	2024-10-10T15:29:36.186Z	END RequestId: d905ac05-f1d4-41f0-9dbd-4c2953c35e1a
▶	2024-10-10T15:29:36.186Z	REPORT RequestId: d905ac05-f1d4-41f0-9dbd-4c2953c35e1a Duration: 2.19 ms Billed Duration: 3 ms Memory Size: 12_
No newer events at this moment. <i>Auto retry paused.</i> Resume		

CONCLUSION:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. Using the S3 Put template ensured that the function was triggered correctly by S3 events. The experiment demonstrated Lambda's event-driven architecture, showing how it can automatically respond to file uploads in S3. Additionally, we learned how to troubleshoot common issues with event setup and verify activity logs using CloudWatch.