

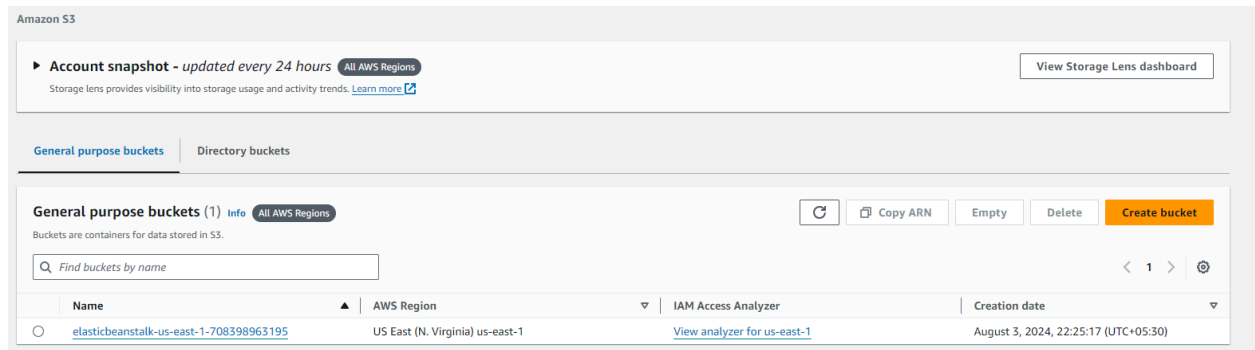
## Experiment No:12

**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

### Steps :

#### 1. Creating the lambda function:

Go to AWS ACADEMY. Now search and open S3 from services. Then click on create bucket.



2. Now Give a name to your Bucket. Here I have given the name as lambda\_buche.

**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](#)  
lambda\_buche

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

3. then remove the Block public access and keep other settings default.

**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

4. Thus, the S3 bucket was created successfully.

General purpose buckets			
Directory buckets			
General purpose buckets (2) <a href="#">Info</a> <a href="#">All AWS Regions</a>			
Buckets are containers for data stored in S3.			
<input type="text" value="Find buckets by name"/>			
<div><div></div><div>1</div><div></div></div>			
Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/> <a href="#">anshi-sadneya-lambda-bucket</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	October 5, 2024, 11:28:16 (UTC+05:30)
<input type="radio"/> <a href="#">elasticbeanstalk-us-east-1-708398963195</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	August 3, 2024, 22:25:17 (UTC+05:30)

5. Search and Open lambda console and click on create function button.

aws

Services

Search

[Alt+S]

N. Virginia

Bhushankor

Compute

# AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

Create a function

How it works

Run

Next: Lambda responds to events

"Hello from Lambda"

.NET | Java | **Node.js** | Python | Ruby | Custom runtime

```
1 * exports.handler = async (event) => {
2   console.log(event);
3   return 'Hello from Lambda!';
4 };
5
```

CloudShell

Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

6. Now Give a name to your Lambda function, Select the language to write your function. Here I have chosen python3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions. Note that the console code editor supports only Node.js, Python, and Ruby

**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

7. Thus the Lambda function was created successfully.

[Lambda](#) > [Functions](#) > anshi-sadneya-lambda

**anshi-sadneya-lambda** Throttle Copy ARN Actions

**Function overview** [Info](#) Export to Application Composer Download

**Diagram** Template

anshi-sadneya-lambda  
Layers (0)

+ Add trigger + Add destination

**Description**  
-

**Last modified**  
2 seconds ago

**Function ARN**  
arn:aws:lambda:us-east-1:708398963195:function:anshi-sadneya-lambda

**Function URL** [Info](#)  
-

8. Then Go into the code section. You will see some default code there.

**Code** Test Monitor Configuration Aliases Versions

**Code source** [Info](#)

File Edit Find View Go Tools Window Test Deploy

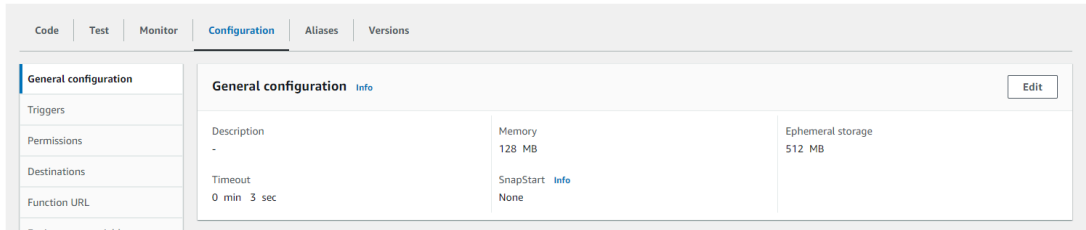
Go to Anything (Ctrl-P)

Environment

- anshi-sadneya-lam
- lambda\_function.py

```
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
```

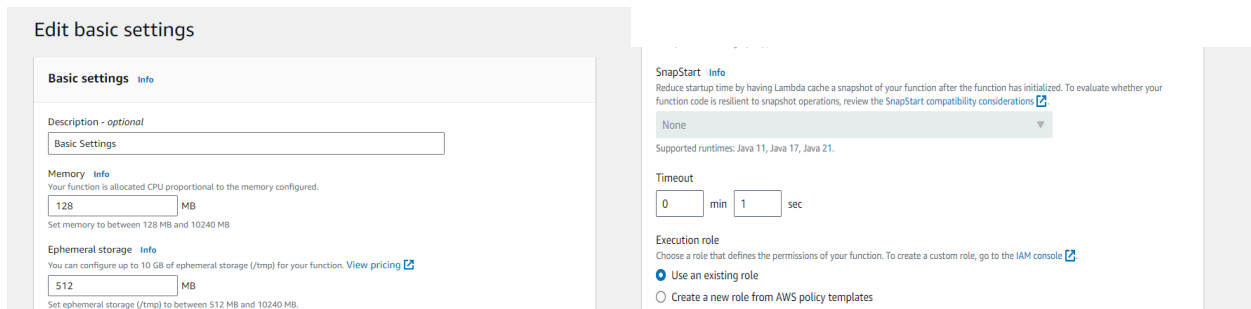
9. To Edit the basic settings go to configuration then click on edit setting.



The screenshot shows the 'Configuration' tab of the AWS Lambda console. The 'General configuration' section is active, displaying a table with the following details:

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

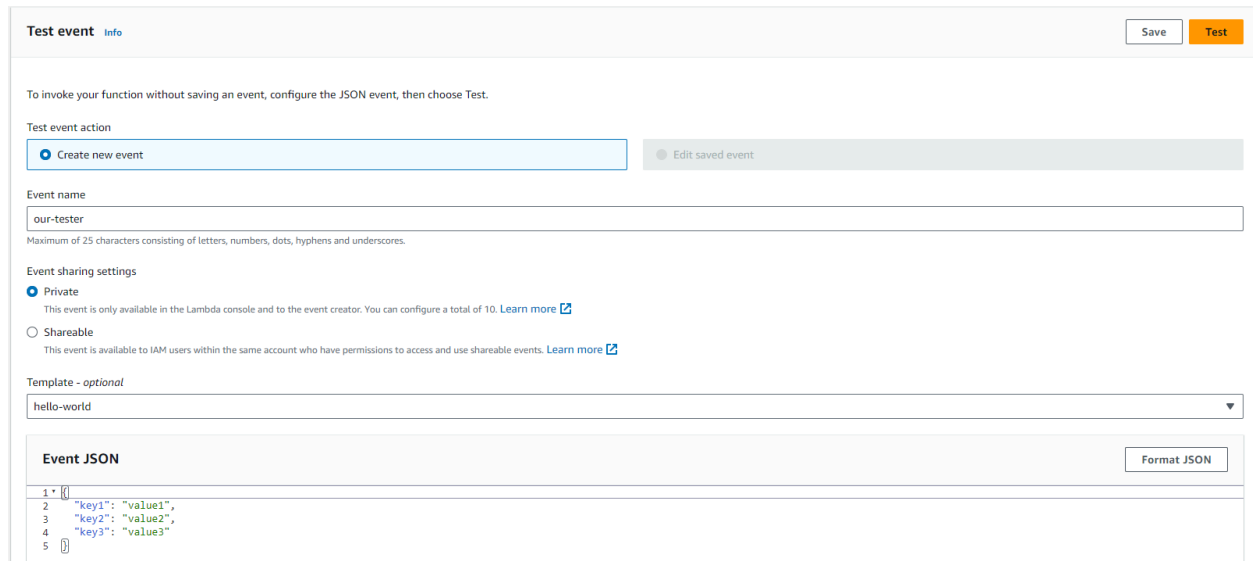
10. Here, enter a description which is optional and change Memory and Timeout.  
I've changed the Timeout period to 1 sec.



The screenshot shows the 'Edit basic settings' page for a Lambda function. The 'Basic settings' section is active, showing the following configuration:

- Description - optional:** Basic Settings
- Memory:** 128 MB (Set memory to between 128 MB and 10240 MB)
- Ephemeral storage:** 512 MB (Set ephemeral storage (/tmp) to between 512 MB and 10240 MB)
- SnapStart:** None (Supported runtimes: Java 11, Java 17, Java 21)
- Timeout:** 0 min 1 sec
- Execution role:** Use an existing role (Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console)

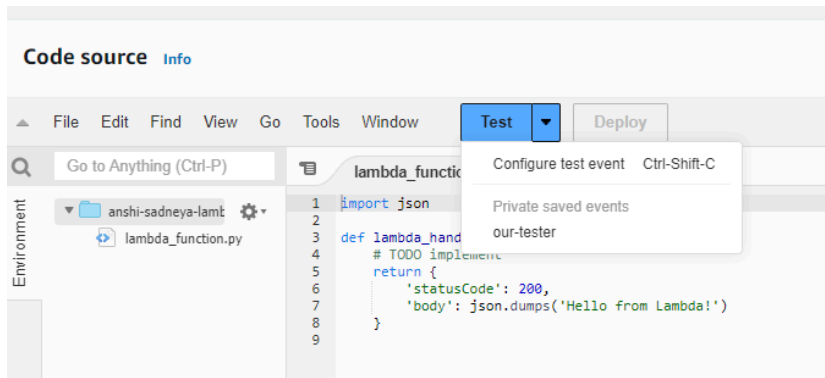
11. Now Click on the Test then select Create a new event, give a name to the event. Here I have given name as 'our-tester' and then select Event Sharing to private, and select s3 put template.



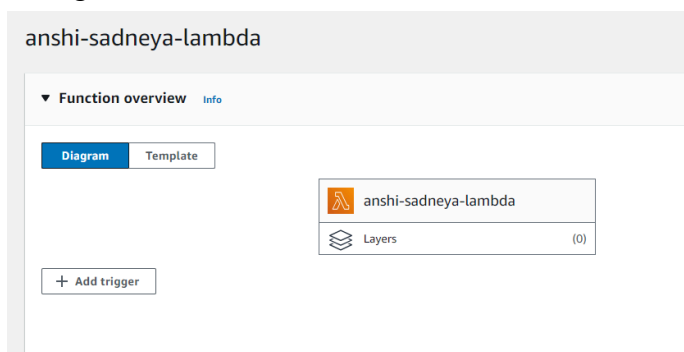
The screenshot shows the 'Test event' page for a Lambda function. The 'Test event action' is set to 'Create new event'. The 'Event name' is 'our-tester'. The 'Event sharing settings' are set to 'Private'. The 'Template - optional' is 'hello-world'. The 'Event JSON' section shows the following JSON:

```
1 {  
2   "key1": "value1",  
3   "key2": "value2",  
4   "key3": "value3"  
5 }
```

12. Now go to the Code section. Then click on the Test dropdown icon and select the event which we have created now ('our-tester').



13. Now go into the Lambda function and then click on add trigger.




14. Now in the Trigger information. Select the source as S3. Then select the bucket which we have created now (lambda\_buche), keep other things default and also you can add prefix to image.

Lambda > Add triggers

### Add trigger

**Trigger configuration** Info

 **S3**  
aws asynchronous storage

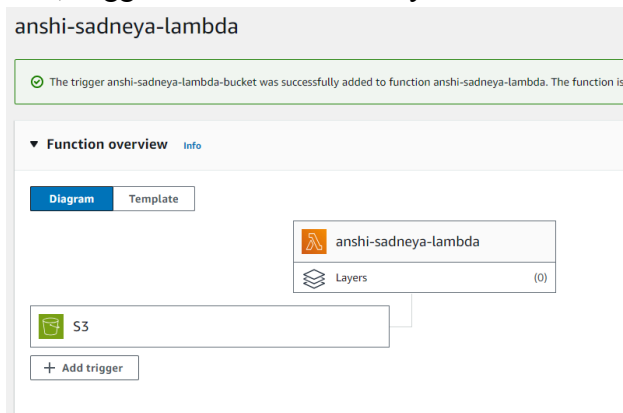
**Bucket**  
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

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.

**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.

15. Thus, Trigger created successfully.



16. You can also check it in the configuration section.

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

General configuration

**Triggers**

Permissions

Destinations


Function URL

Environment variables

**Triggers (1)** Info

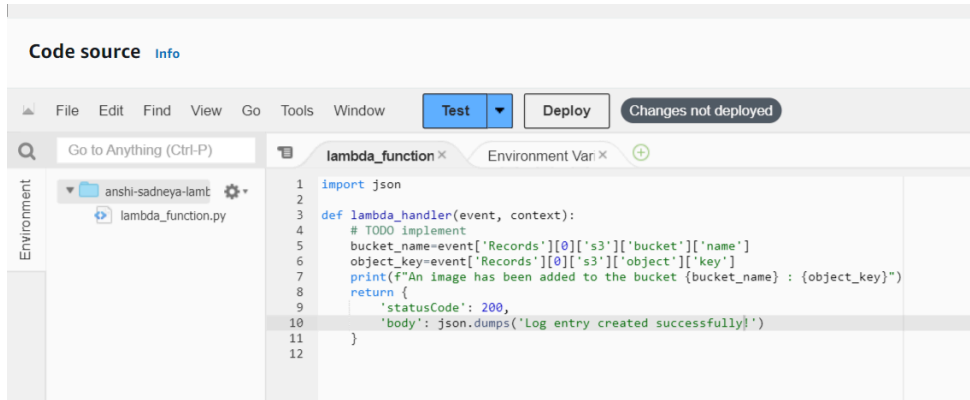
< 1 >

☐ Trigger

☐  **S3: anshi-sadneya-lambda-bucket**  
arn:aws:s3::anshi-sadneya-lambda-bucket

► Details

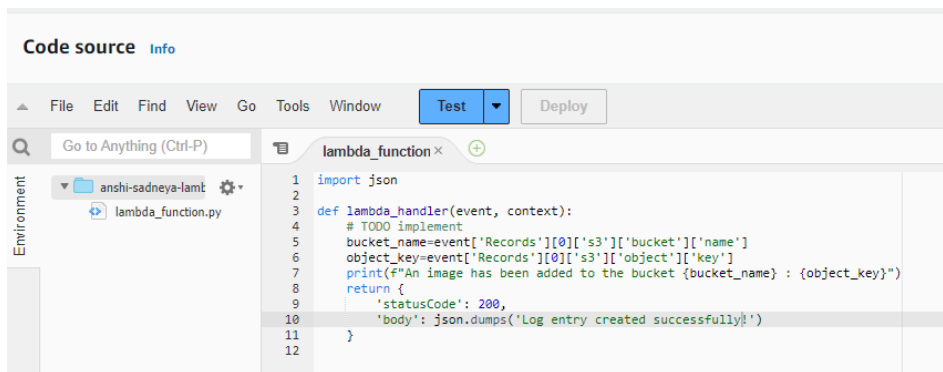
17. Now write a code which logs a message “Log entry crested successfully” when triggered.



```
Code source Info
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (Ctrl-P)
Environment
  anshi-sadneya-lam
  lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     bucket_name=event['Records'][0]['s3']['bucket']['name']
6     object_key=event['Records'][0]['s3']['object']['key']
7     print(f"An image has been added to the bucket {bucket_name} : {object_key}")
8     return {
9         'statusCode': 200,
10        'body': json.dumps('Log entry created successfully!')}
11
12
```

Here changes are not deployed.

18. So now,Save the file by ctrl+s and then click on deploy.



```
Code source Info
File Edit Find View Go Tools Window Test Deploy
Go to Anything (Ctrl-P)
Environment
  anshi-sadneya-lam
  lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     bucket_name=event['Records'][0]['s3']['bucket']['name']
6     object_key=event['Records'][0]['s3']['object']['key']
7     print(f"An image has been added to the bucket {bucket_name} : {object_key}")
8     return {
9         'statusCode': 200,
10        'body': json.dumps('Log entry created successfully!')}
11
12
```

19. Go to S3 bucket,and there upload any image to the bucket.

**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, 1.5 MB) Remove Add files Add folder

All files and folders in this table will be uploaded.

< 1 >

<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	abigail-lynn-9Jr8iphz0e0-unsplash.jpg	-

**Destination** Info

Destination  
[s3://anshi-sadneya-lambda-bucket](#)

**Destination details**  
Bucket settings that impact new objects stored in the specified destination.

**Permissions**  
Grant public access and access to other AWS accounts.

**Properties**  
Specify storage class, encryption settings, tags, and more.

Cancel Upload

20. Thus image uploaded successfully.

**Upload succeeded**  
View details below.

**Upload: status** Close

The information below will no longer be available after you navigate away from this page.

**Summary**

Destination <a href="#">s3://anshi-sadneya-lambda-bucket</a>	Succeeded 1 file, 1.5 MB (100.00%)	Failed 0 files, 0 B (0%)
---	---------------------------------------	-----------------------------

**Files and folders** | Configuration

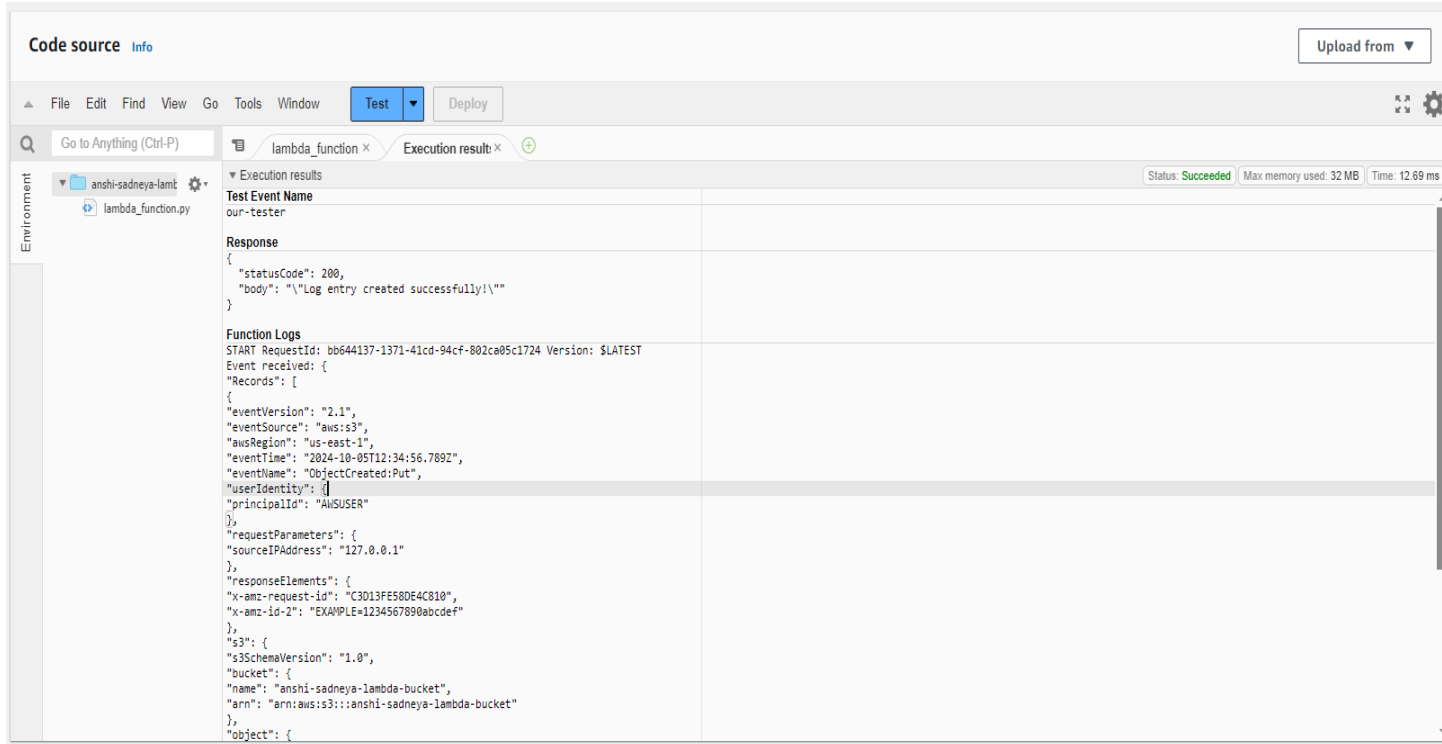
**Files and folders** (1 Total, 1.5 MB)

< 1 >

Name	Folder	Type	Size	Status	Error
<a href="#">abigail-lynn-...</a>	-	image/jpeg	1.5 MB	Succeeded	-

21. Now goto lambda function. Then click on test. This will give you log about the image that we have uploaded in S3 bucket.





The screenshot displays the AWS Lambda console interface. At the top, there's a 'Code source' tab and an 'Info' link. Below this is a navigation bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window' menus, along with 'Test' and 'Deploy' buttons. The main area is divided into two tabs: 'lambda\_function' and 'Execution result'. The 'Execution result' tab is active, showing the following details:

- Test Event Name:** our-tester
- Response:**

```
{
  "statusCode": 200,
  "body": "\\Log entry created successfully!\\n"
}
```
- Function Logs:**

```
START RequestId: bb644137-1371-41cd-94cf-802ca05c1724 Version: $LATEST
Event received: {
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-10-05T12:34:56.789Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "AWSUSER"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "C3D13FE58DE4C810",
        "x-amz-id-2": "EXAMPLE1234567890abcdef"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "bucket": {
          "name": "anshi-sadneya-lambda-bucket",
          "arn": "arn:aws:s3:::anshi-sadneya-lambda-bucket"
        },
        "object": {
```

The status bar at the bottom right indicates 'Status: Succeeded', 'Max memory used: 32 MB', and 'Time: 12.69 ms'.

**In response,It gives status 200 and also the message “Log entry created successfully” and also contains function Logs.**

22. Now go to cloudwatch. Then go into log groups. Inside that you will get the lambda function name that we have created click on it. Here, you will get a detailed log of events.

The screenshot shows the AWS CloudWatch 'Log events' page for a Lambda function named 'anshi-sadneya-lambda'. The breadcrumb trail is: CloudWatch > Log groups > /aws/lambda/anshi-sadneya-lambda > 2024/10/05/[\$LATEST]7c6cf8d57a964810b5ef202abaa432aa. The page has a search bar with the placeholder 'Filter events - press enter to search' and buttons for 'Clear', '1m', '30m', '1h', '12h', 'Custom', 'UTC timezone', 'Display', and a settings icon. Below the search bar, there is a table with two columns: 'Timestamp' and 'Message'. The table contains several log entries, including 'INIT\_START', 'START', 'Event received', and 'END'. The first entry is 'No older events at this moment. [Retry](#)'. The second entry is 'INIT\_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:18809ce2714ff5637bd2bbe06ce081ec3bc408a9f277dab104c14cd814b081'. The third entry is 'START RequestId: 95f7832d-4bb2-45c3-99ae-64f848bc719e Version: \$LATEST'. The fourth entry is 'Event received: { "key1": "value1", "key2": "value2", "key3": "value3" }'. The fifth entry is 'END RequestId: 95f7832d-4bb2-45c3-99ae-64f848bc719e'. The sixth entry is 'REPORT RequestId: 95f7832d-4bb2-45c3-99ae-64f848bc719e Duration: 1.71 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 71.52 ms'. The seventh entry is 'START RequestId: bb644137-1371-41cd-94cf-802ca05c1724 Version: \$LATEST'. The eighth entry is 'Event received: { "Records": [ { "eventVersion": "2.1",

Timestamp	Message
	No older events at this moment. <a href="#">Retry</a>
2024-10-05T06:45:53.784Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:18809ce2714ff5637bd2bbe06ce081ec3bc408a9f277dab104c14cd814b081
2024-10-05T06:45:53.778Z	START RequestId: 95f7832d-4bb2-45c3-99ae-64f848bc719e Version: \$LATEST
2024-10-05T06:45:53.778Z	Event received: {
2024-10-05T06:45:53.778Z	"key1": "value1",
2024-10-05T06:45:53.778Z	"key2": "value2",
2024-10-05T06:45:53.778Z	"key3": "value3"
2024-10-05T06:45:53.778Z	}
2024-10-05T06:45:53.780Z	END RequestId: 95f7832d-4bb2-45c3-99ae-64f848bc719e
2024-10-05T06:45:53.780Z	REPORT RequestId: 95f7832d-4bb2-45c3-99ae-64f848bc719e Duration: 1.71 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 71.52 ms
2024-10-05T06:47:52.671Z	START RequestId: bb644137-1371-41cd-94cf-802ca05c1724 Version: \$LATEST
2024-10-05T06:47:52.671Z	Event received: {
2024-10-05T06:47:52.671Z	"Records": {
2024-10-05T06:47:52.671Z	{
2024-10-05T06:47:52.671Z	"eventVersion": "2.1",

**Conclusion:** In this, we have created lambda function successfully. Then we have also created s3 bucket successfully. Then I have edited the setting by setting timeout for 1 sec and adding a description. Then created an event name 'our-tester'. Then selected that event for test. Then deployed it. Thus deployed successfully. Then we have added a trigger in which we added an s3 bucket which we created. Then we have added a print message. Then again deployed the code after uploading an image in an s3 bucket. This given a status code 200 in response with that message which we added in code. Then in CloudWatch it given detailed logs of it.