

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.

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.

Amazon S3

► Account snapshot - updated every 24 hours All AWS Regions View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

General purpose buckets | Directory buckets

General purpose buckets (1) All AWS Regions

Buckets are containers for data stored in S3.

Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-708398963195	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 3, 2024, 22:25:17 (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.

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

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.

General purpose buckets

Directory buckets

General purpose buckets (2) Info All AWS Regions

Refresh

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 > ⚙

	Name ▲	AWS Region ▼	IAM Access Analyzer	Creation date ▼
<input type="radio"/>	anshi-sadneya-lambda-bucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 5, 2024, 11:28:16 (UTC+05:30)
<input type="radio"/>	elasticbeanstalk-us-east-1-708398963195	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 3, 2024, 22:25:17 (UTC+05:30)

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

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo and a search bar. The main heading reads 'AWS Lambda lets you run code without thinking about servers.' Below this, a 'Get started' box contains a 'Create a function' button. Further down, the 'How it works' section displays a sample code editor for a 'Hello from Lambda!' function, with tabs for different runtimes like .NET, Java, Node.js, Python, Ruby, and Custom runtime. The code editor shows a simple JavaScript function that logs an event and returns a string.

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 Exceution role to Create a new role with basic Lambda permissions.

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.

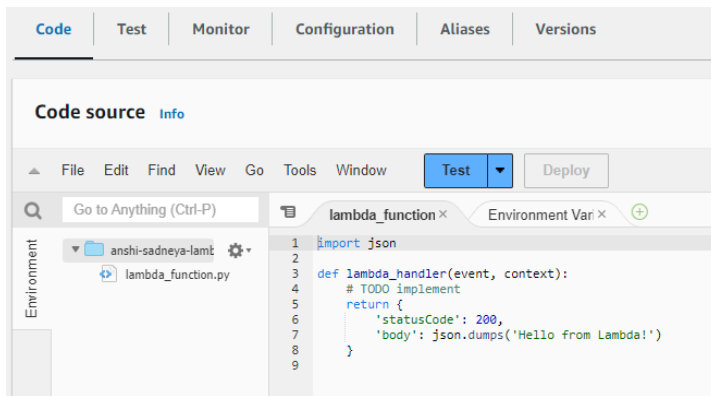
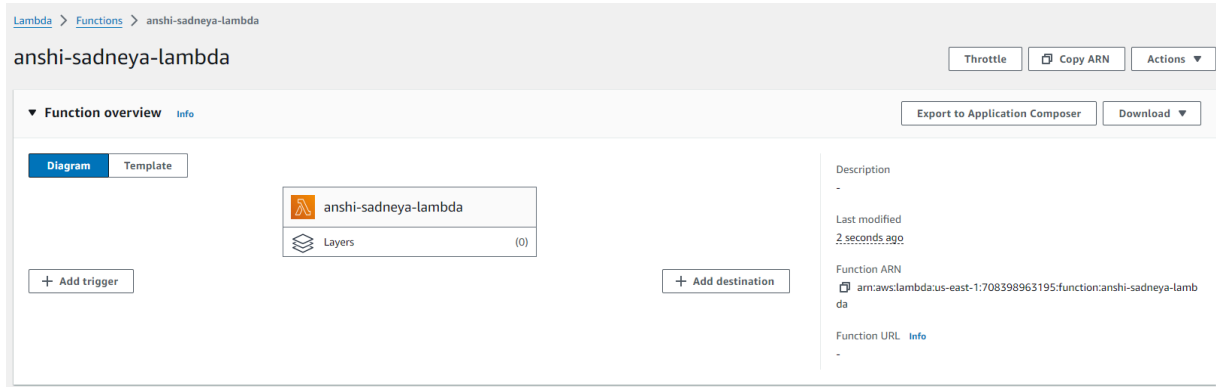
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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

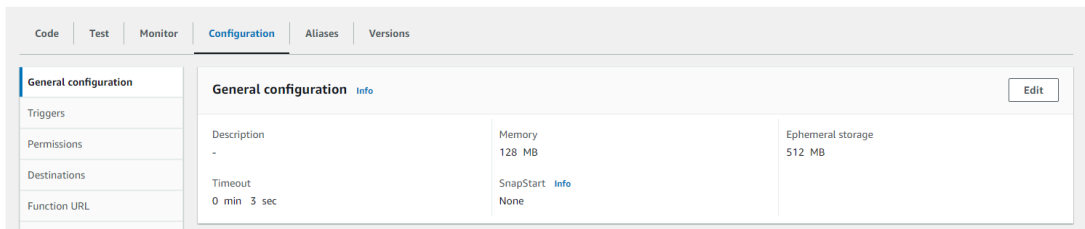
Python 3.12
[Refresh](#)

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

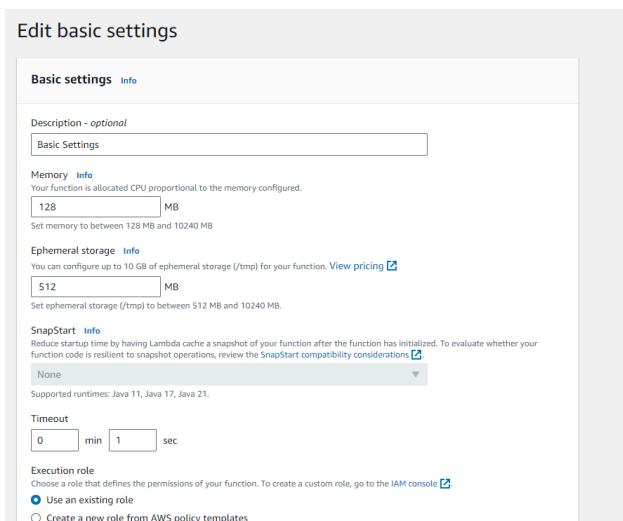
☒ x86_64
☐ arm64



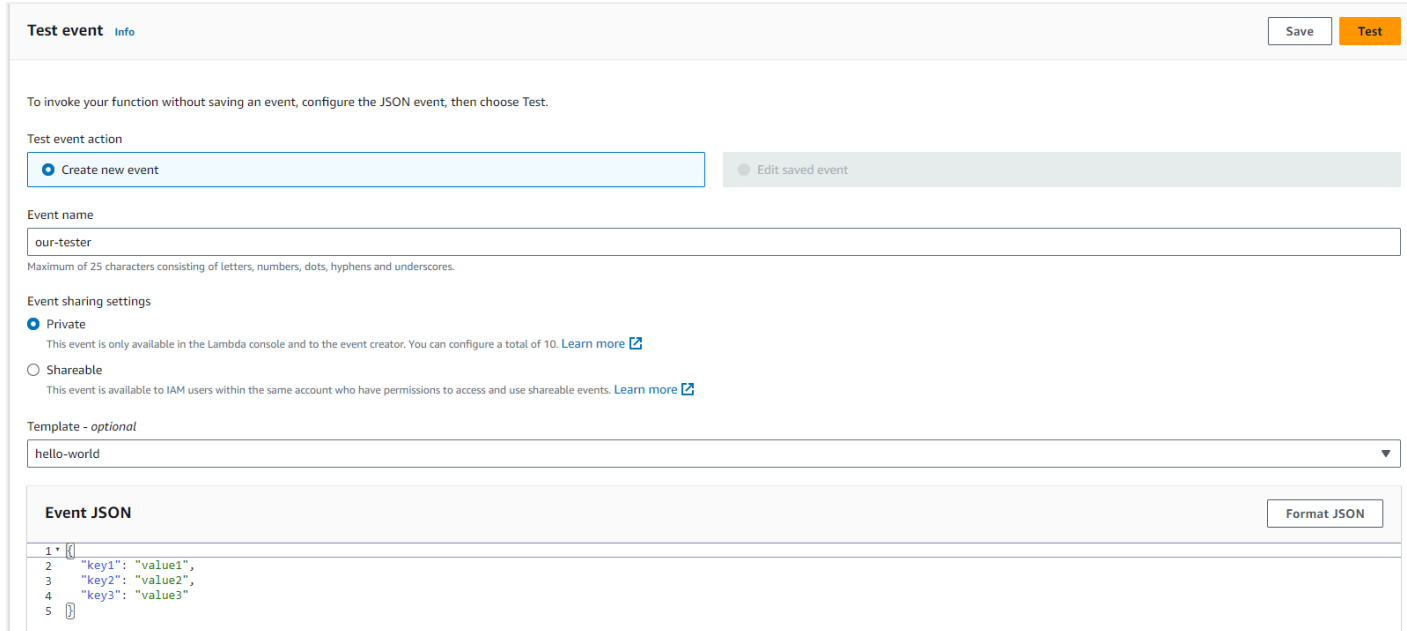
So See or Edit the basic settings go to configuration then click on edit general setting.



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

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

our-tester

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](#)

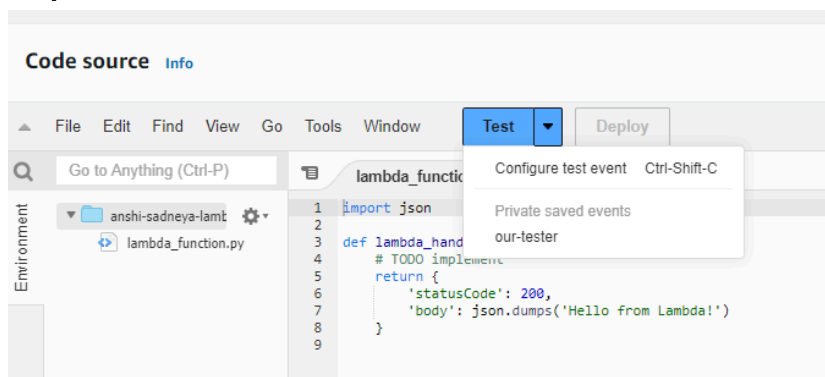
Template - optional

hello-world

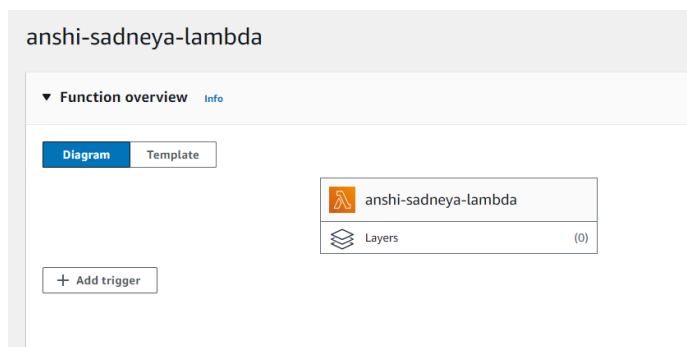
Event JSON Format JSON

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

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.


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.

anshi-sadneya-lambda

☒ The trigger anshi-sadneya-lambda-bucket was successfully added to function anshi-sadneya-lambda. The function is r

Function overview [Info](#)


anshi-sadneya-lambda
Layers (0)



S3

Code
Test
Monitor
Configuration
Aliases
Versions

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables

Triggers (1) [Info](#)

☐ Trigger


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

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

Code source [Info](#)

File
Edit
Find
View
Go
Tools
Window

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

Code source [Info](#)

File
Edit
Find
View
Go
Tools
Window

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

Step 9: Now 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)
All files and folders in this table will be uploaded.

☐

Name

▼

Folder

☐

abigail-lynn-9Jr8lphz0eO-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**

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

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)

Name	Folder	Type	Size	Status	Error
abigail-lynn-...	-	image/jpeg	1.5 MB	Succeeded	-

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

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

File Edit Find View Go **Test** Deploy

Environment
anshi-sadneya-lam
lambda_function.py

lambda_function x

Execution results x

Status: Succeeded

Max memory used: 32 MB

Time: 12.69 ms

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": "ANSUSER"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "C3013F58D0E4C810",
        "x-amz-id-2": "EXAMPLE1234567890abcdeff"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "bucket": {
          "name": "anshi-sadneya-lambda-bucket",
          "arn": "arn:aws:s3:::anshi-sadneya-lambda-bucket"
        },
        "object": {
```

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

CloudWatch > Log groups > /aws/lambda/anshi-sadneya-lambda > 2024/10/05/[LATEST]7c6cf8d57a964810b5ef202abaa432aa

Log events  Actions  Start tailing Create metric filter

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 30m 1h 12h Custom  UTC timezone  Display  

▶	Timestamp	Message
		No older events at this moment. Retry
▶	2024-10-05T06:45:53.780Z	INIT_START Runtime Version: python3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2bbe06ceb81ec3bc488e9f277dab184c14cd814b081
▶	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-802ca85c1724 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 experiment, we successfully created an AWS Lambda function that logs a message whenever an image is uploaded to an S3 bucket. One key takeaway is that selecting the correct event template, specifically the S3 Put event, is essential. Using an incorrect event format initially caused errors due to the Lambda function not receiving the expected event structure.

After resolving these issues, the function was successfully triggered by S3 object uploads, confirming the effectiveness of Lambda's event-driven architecture. This experiment not only showcased Lambda's ability to respond to S3 events efficiently but also highlighted the importance of understanding and troubleshooting common event structure issues.