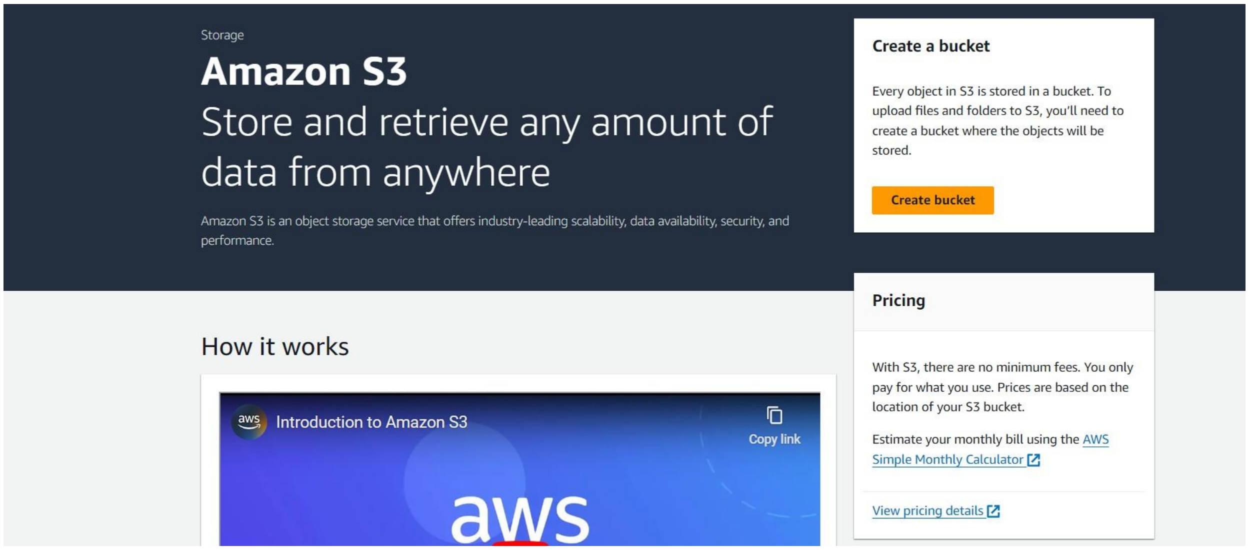
Adv_devops _12 **Advik Hegde** **D15C-15**

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



Storage

Amazon S3

Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create bucket

How it works

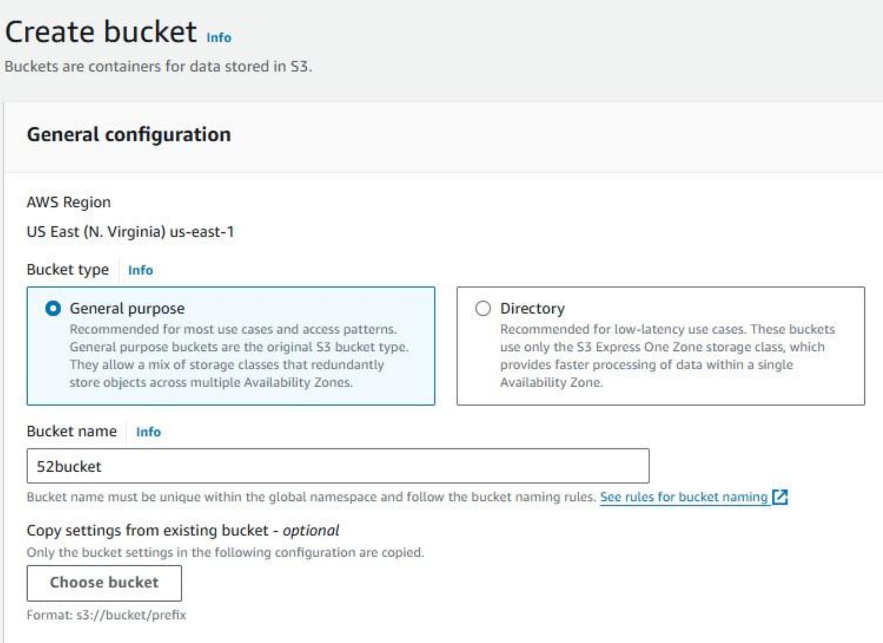
Pricing

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the AWS Simple Monthly Calculator

View pricing details

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



Create bucket Info

Buckets are containers for data stored in S3.

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

52bucket

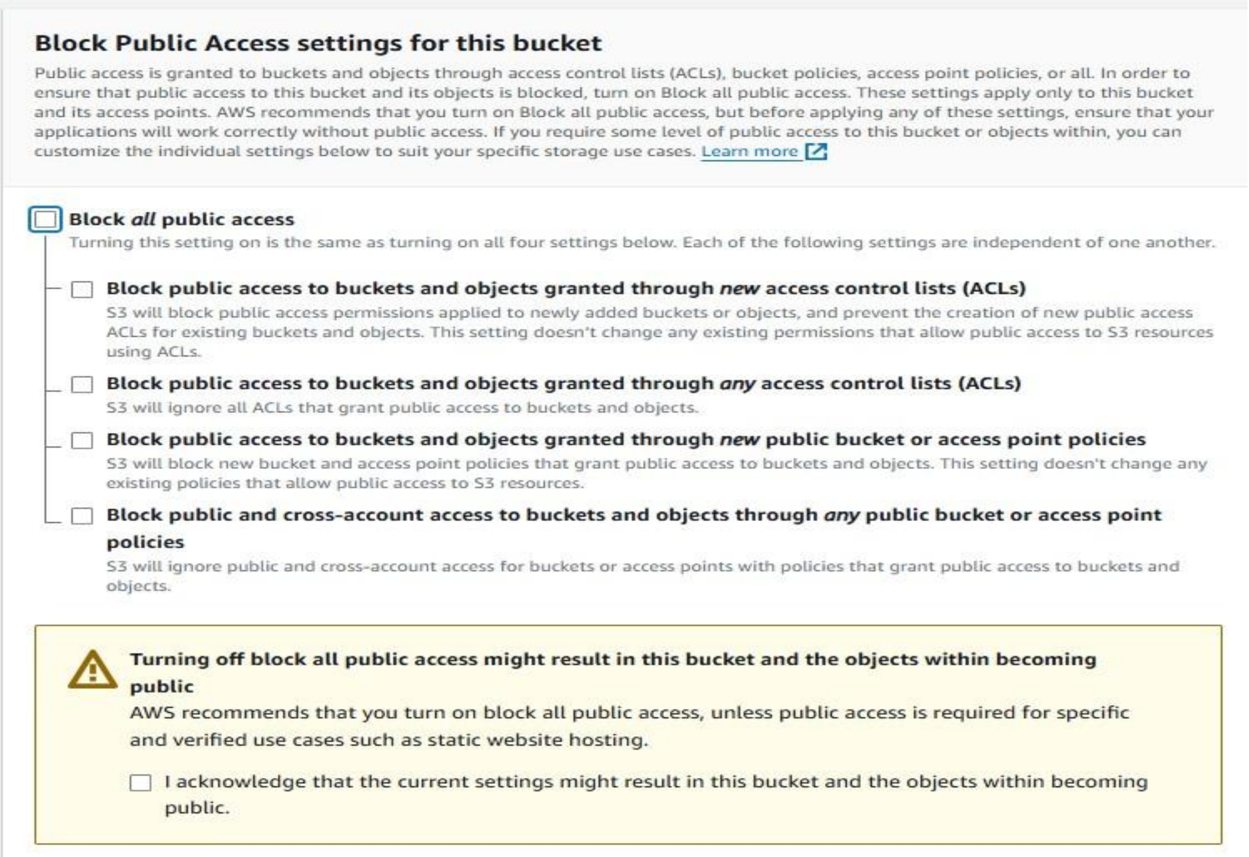
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix



Block Public Access settings for this bucket

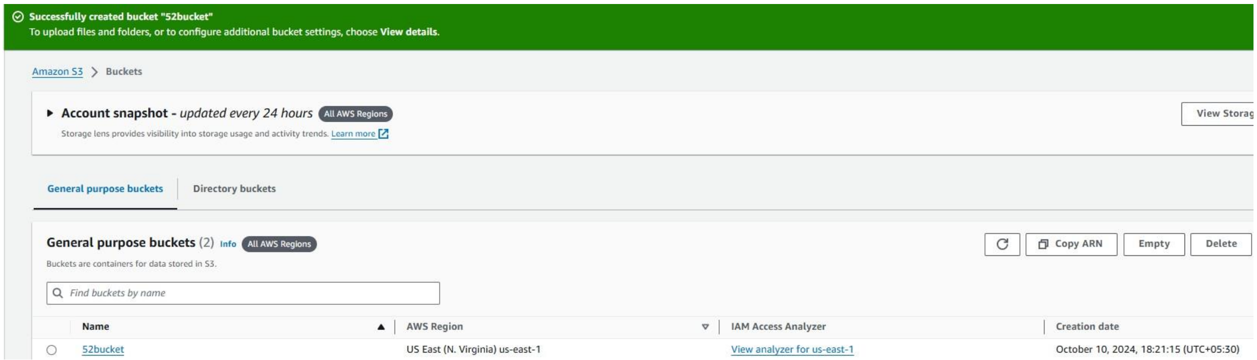
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access 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 "52bucket"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

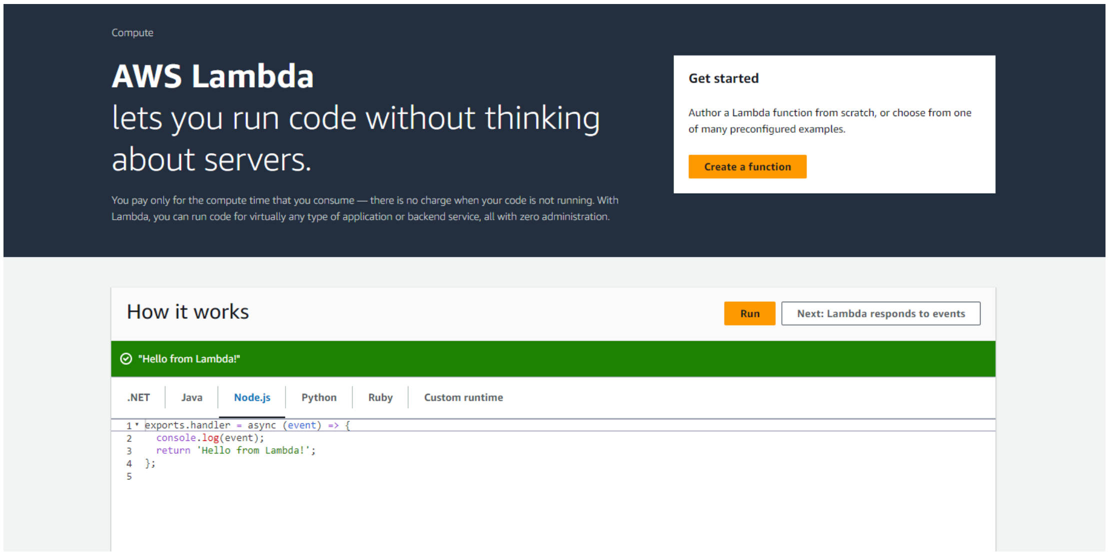
Amazon S3 > Buckets

▸ **Account snapshot** - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (2) Info All AWS Regions
Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
○	52bucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 10, 2024, 18:21:15 (UTC+05:30)

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



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

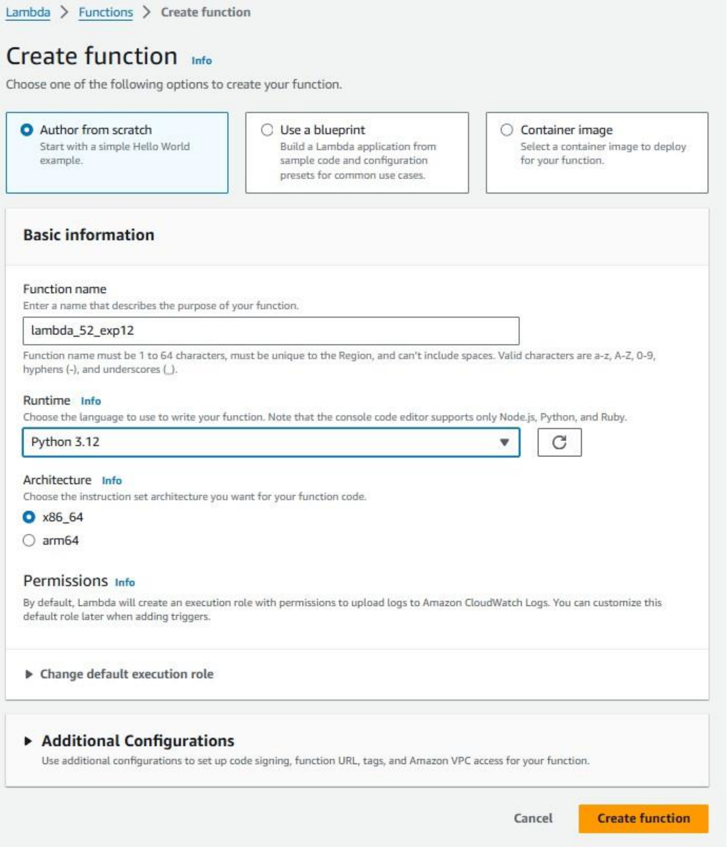
Author a Lambda function from scratch, or choose from one of many preconfigured examples.

How it works

"Hello from Lambda!"

```
exports.handler = async (event) => {
  console.log(event);
  return 'Hello from Lambda!';
};
```

4. 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.

lambda_52_exp12

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

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

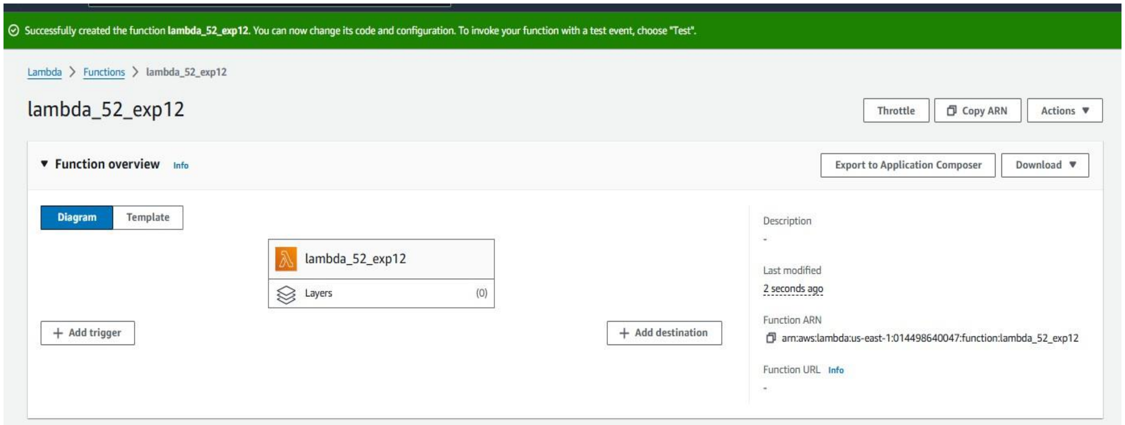
● x86_64
○ arm64

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

▶ **Additional Configurations**
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

5. The Lambda function was created successfully.



Successfully created the function **lambda_52_exp12**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > lambda_52_exp12

lambda_52_exp12

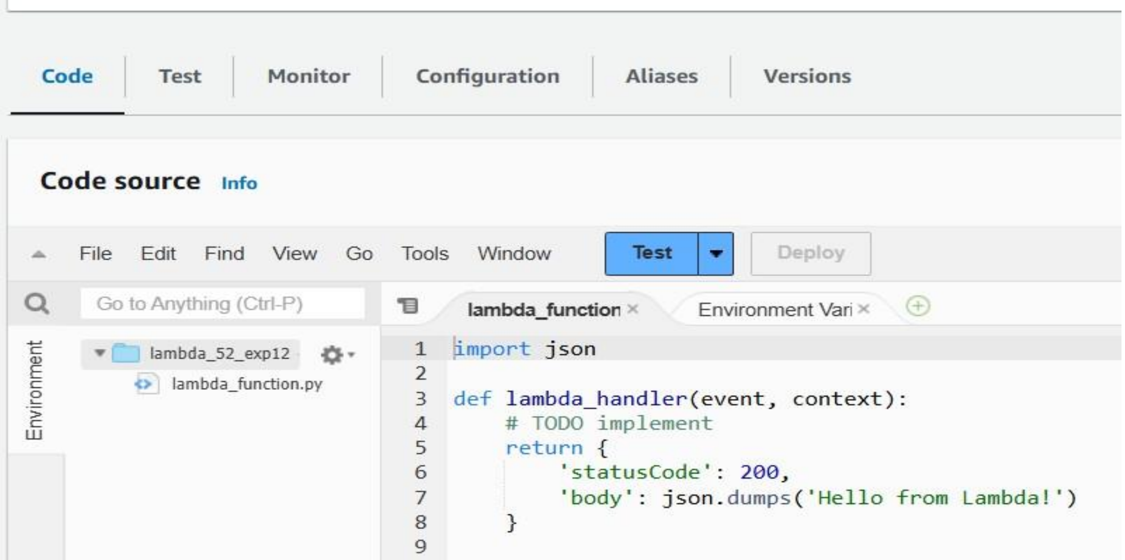
Description
-

Last modified
2 seconds ago

Function ARN
arn:aws:lambda:us-east-1:014498640047:function:lambda_52_exp12

Function URL Info
-

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

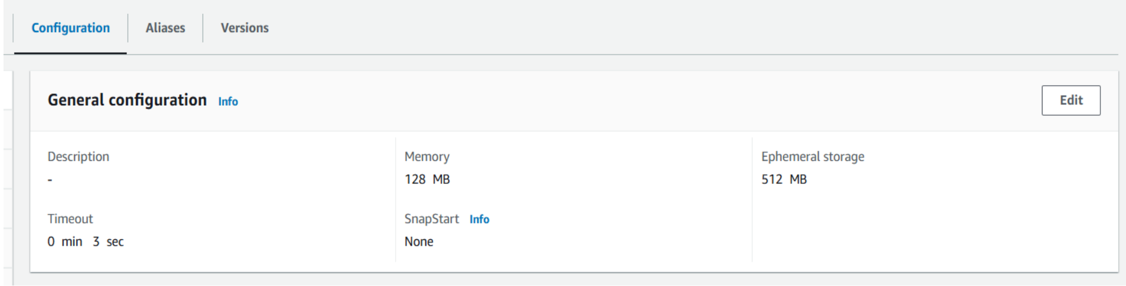


```python
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

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



Description
-

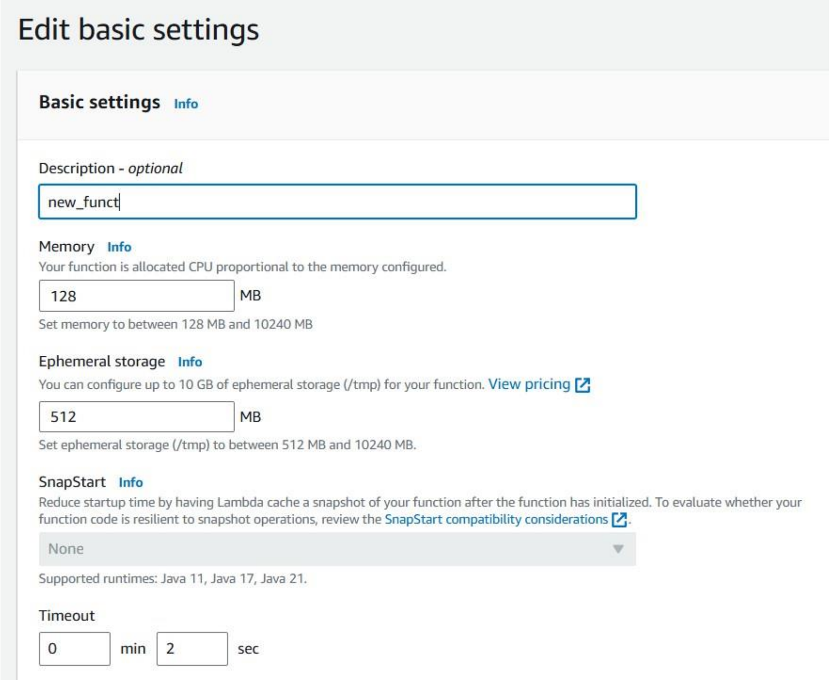
Memory
128 MB

Ephemeral storage
512 MB

Timeout
0 min 3 sec

SnapStart Info
None

8. Here, enter a description which is optional and change Memory and Timeout. I've changed the Timeout period to 2 sec.



Edit basic settings

Basic settings

Description - *optional*

new_funct

Memory

Your function is allocated CPU proportional to the memory configured.

128 MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage

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

512 MB

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

SnapStart

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.

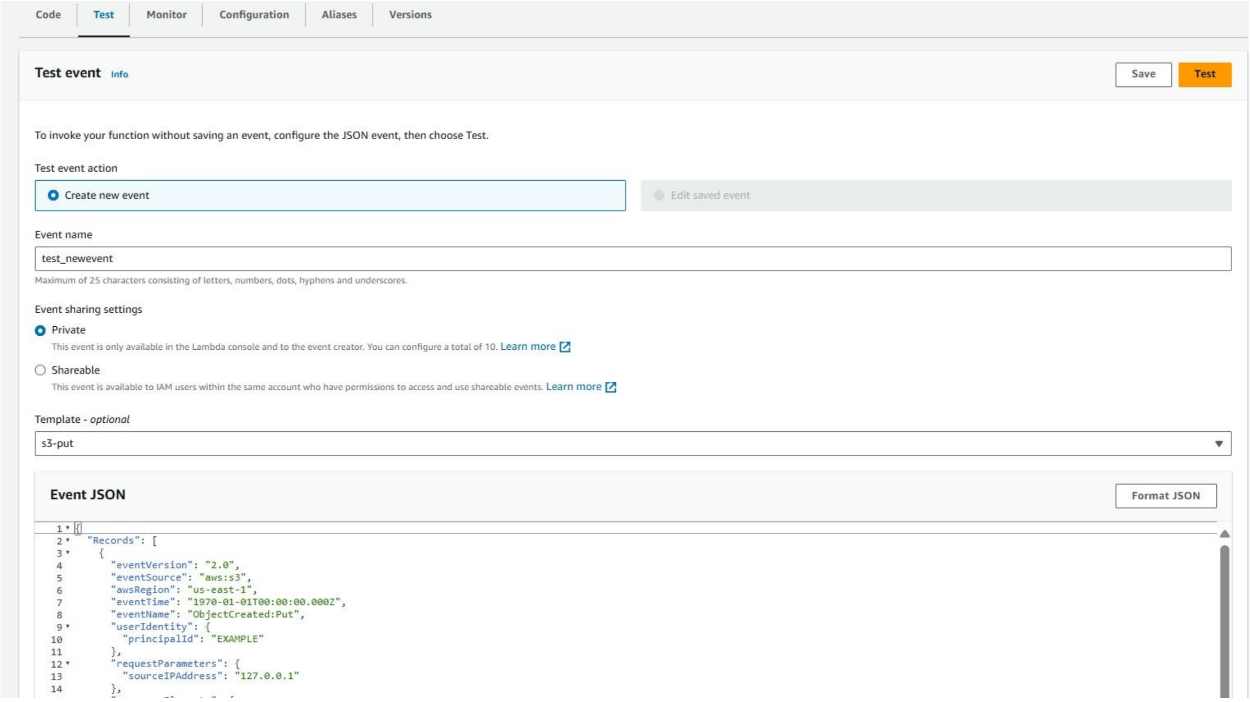
None

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

Timeout

0 min 2 sec

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



Test event

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

test_newevent

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*

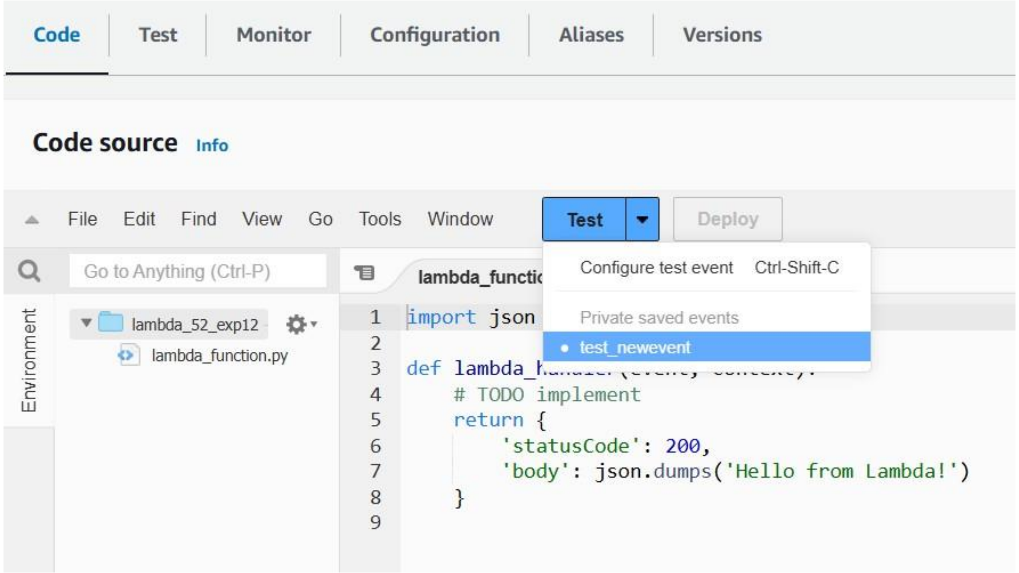
s3-put

Event JSON

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
```

10. Now go to the Code section. Then click on the Test dropdown icon and select the event which

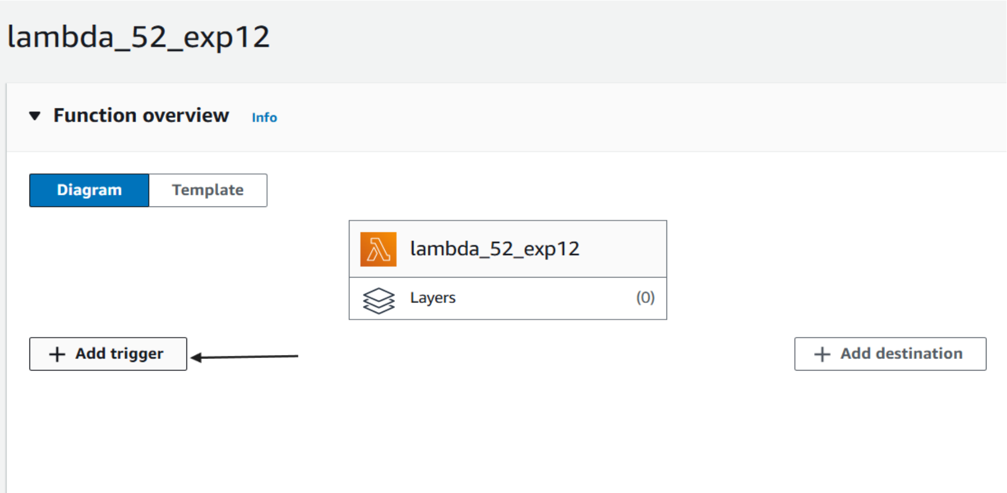
we have created now('test_newevent').



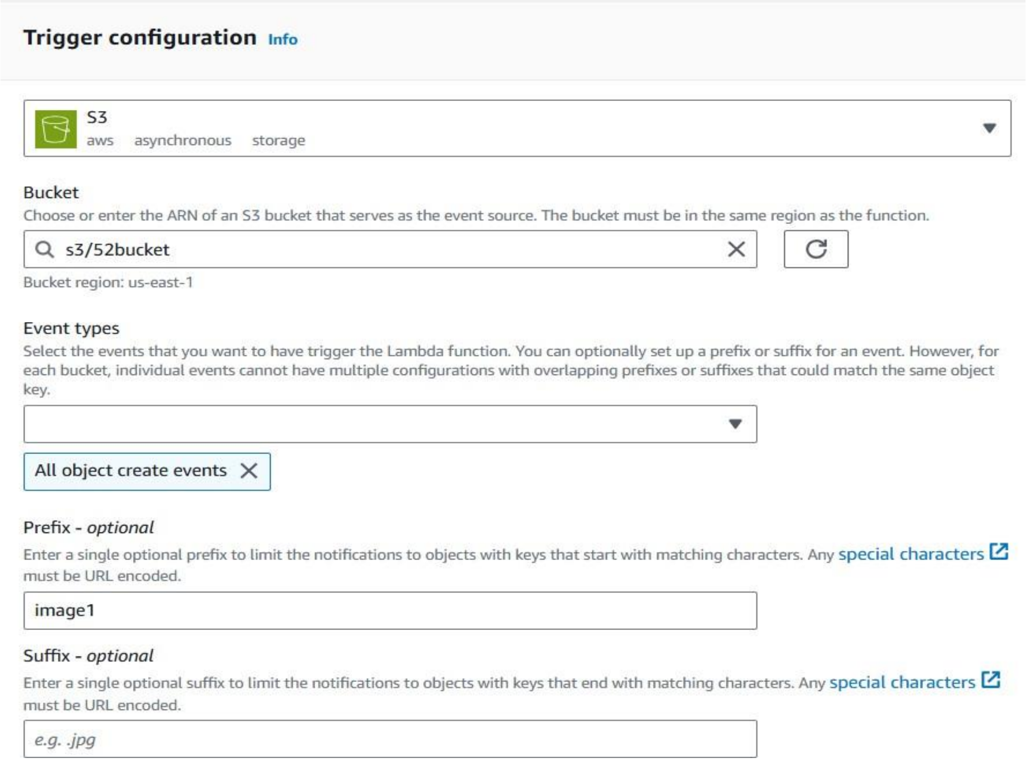
```python
import json

def lambda_h...
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

11. Now go into the Lambda function and then click on add tigger.



12. Now in the Trigger information. Select the source as S3.Then select the bucket which we have created now (52bucket{in my case}), keep other things default and also you can add prefix to image.



Trigger configuration Info

S3
aws asynchronous storage

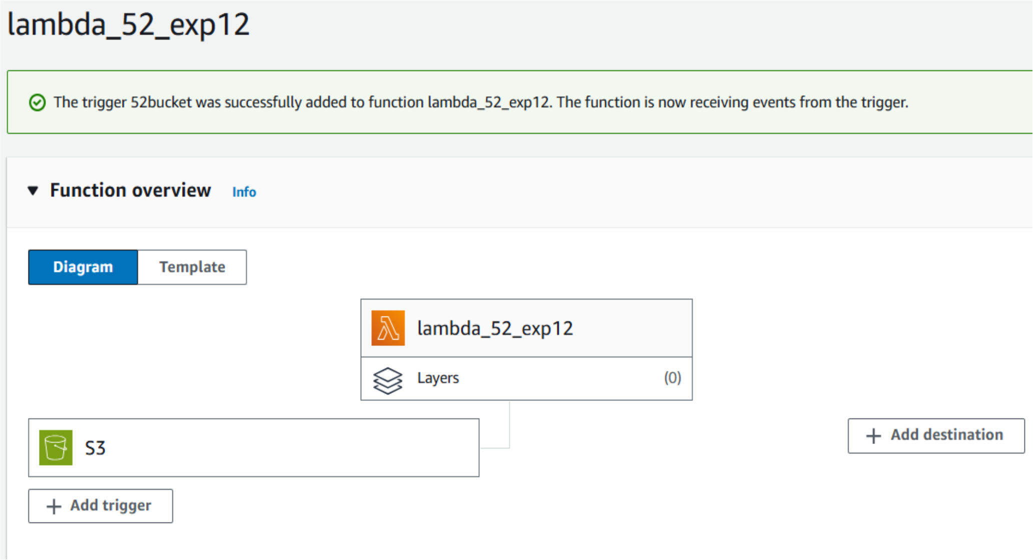
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/52bucket
Bucket region: us-east-1

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.
image1

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.
e.g. .jpg

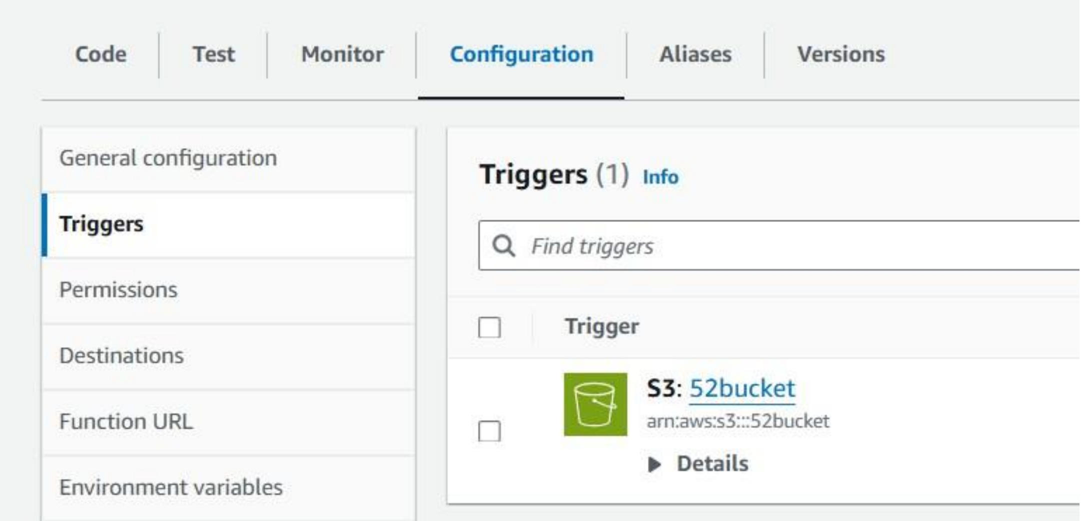
13. Thus, **trigger** is created successfully.



lambda_52_exp12

The trigger 52bucket was successfully added to function lambda_52_exp12. The function is now receiving events from the trigger.

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

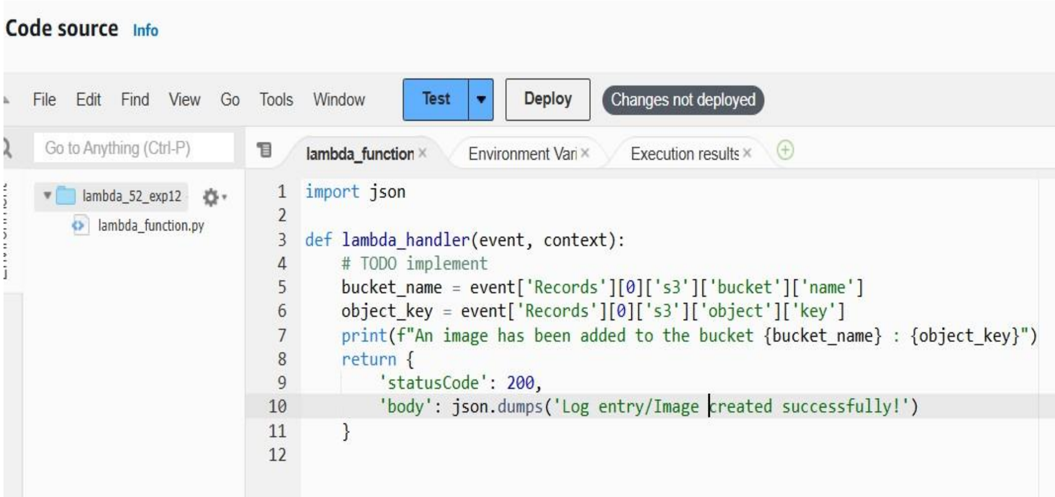


Triggers (1)

S3: 52bucket

arn:aws:s3:::52bucket

15. Now write a code which logs a message “Log entry/image created successfully” when triggered.

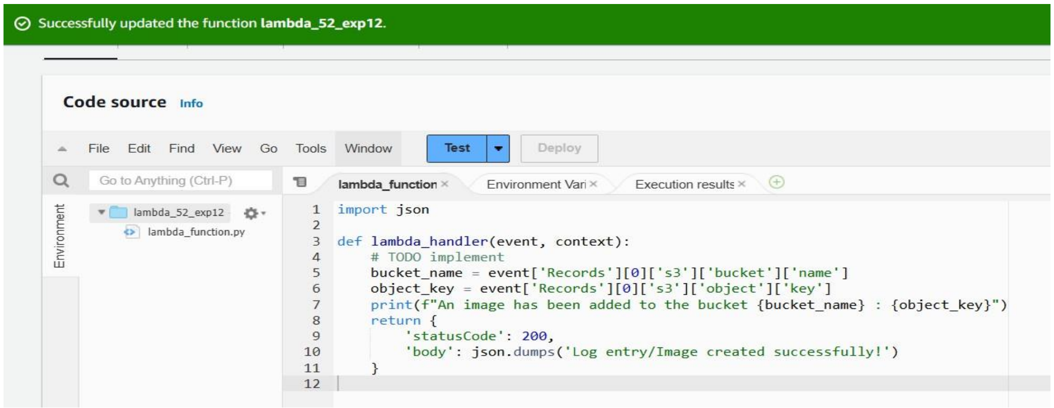


```
import json

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/Image created successfully!')
    }
```

Here changes are not deployed.

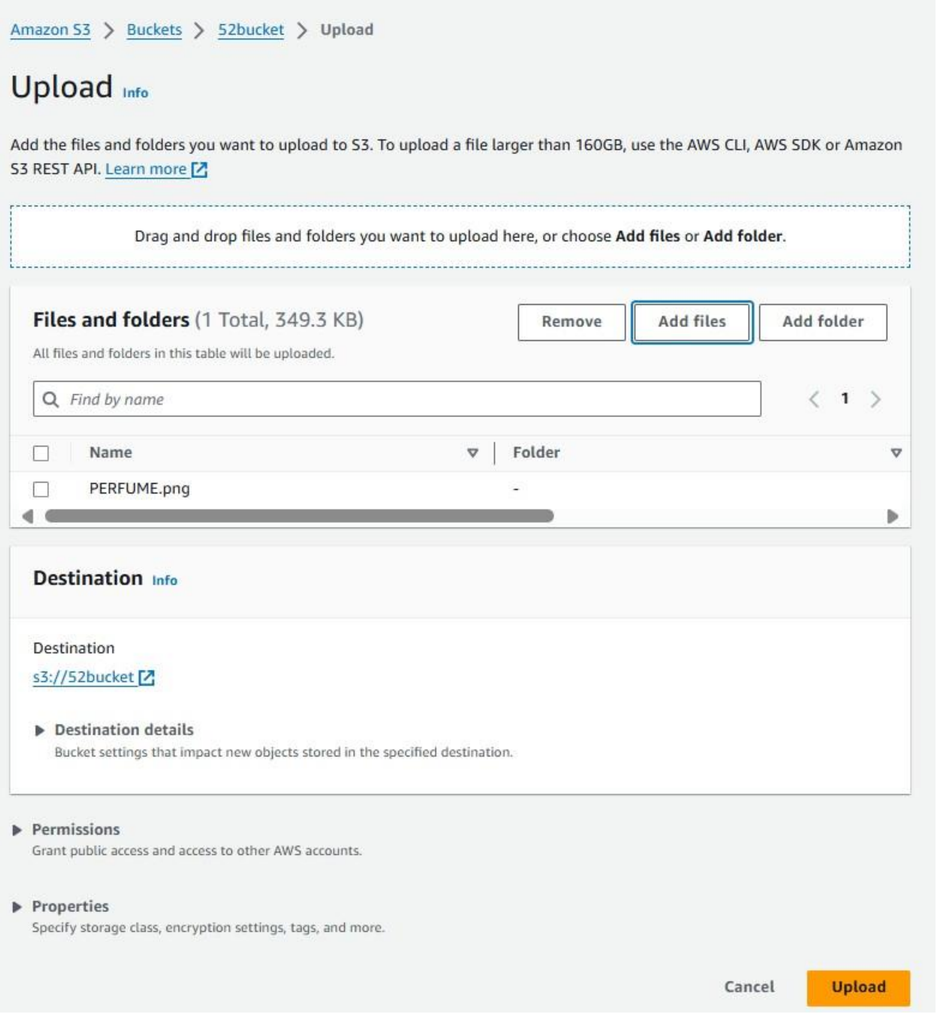
16. So now, Save the file by ctrl+s and then click on deploy.


Successfully updated the function **lambda_52_exp12**.

```
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/Image created successfully!')
    }
```

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



Amazon S3 > Buckets > 52bucket > 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, 349.3 KB)

All files and folders in this table will be uploaded.

□	Name	Folder
□	PERFUME.png	-

Destination Info

Destination

s3://52bucket

▶ **Destination details**

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

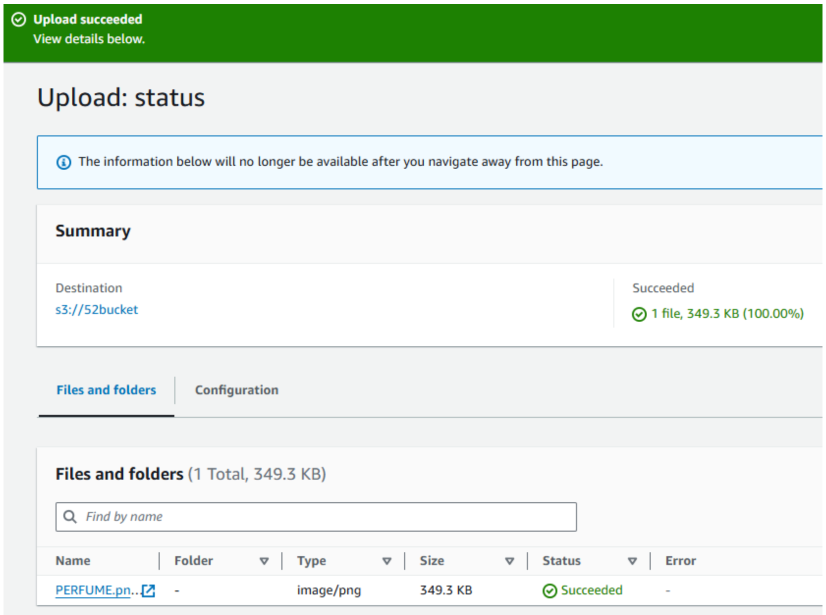
▶ **Permissions**

Grant public access and access to other AWS accounts.

▶ **Properties**

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

18. Thus the image was uploaded successfully



Upload succeeded
View details below.

Upload: status

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

Summary

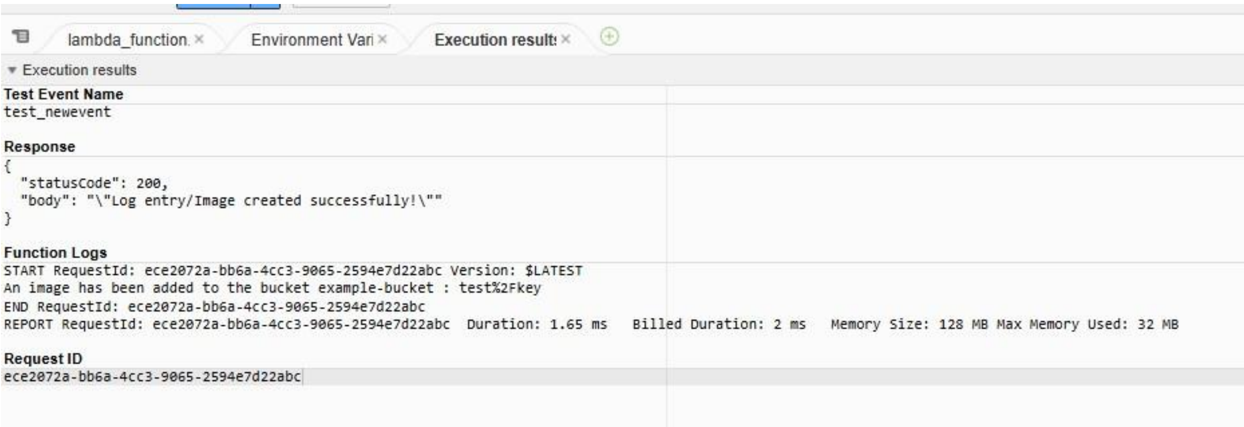
Destination
s3://52bucket

Succeeded
1 file, 349.3 KB (100.00%)

Files and folders (1 Total, 349.3 KB)

Name	Folder	Type	Size	Status	Error
PERFUME.pn..	-	image/png	349.3 KB	Succeeded	-

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



Test Event Name
test_newevent

Response

```
{
  "statusCode": 200,
  "body": "\"Log entry/Image created successfully!\""
}
```

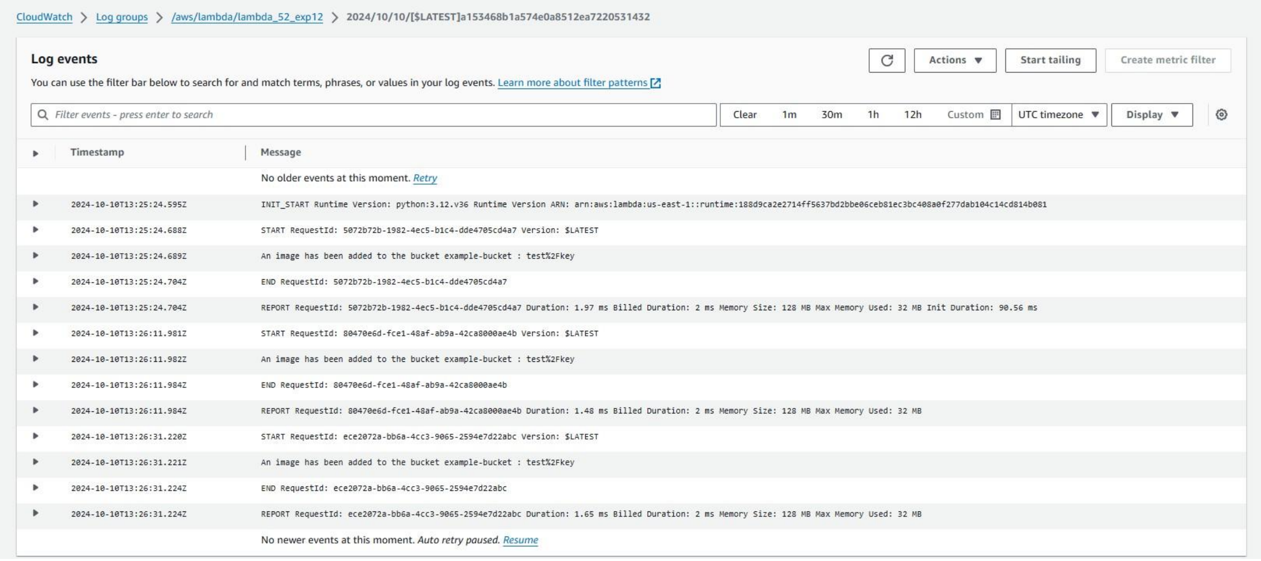
Function Logs

```
START RequestId: ece2072a-bb6a-4cc3-9065-2594e7d22abc Version: $LATEST
An image has been added to the bucket example-bucket : test%2Fkey
END RequestId: ece2072a-bb6a-4cc3-9065-2594e7d22abc
REPORT RequestId: ece2072a-bb6a-4cc3-9065-2594e7d22abc  Duration: 1.65 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 32 MB
```

Request ID
ece2072a-bb6a-4cc3-9065-2594e7d22abc

(In response, It gives status 200 and also the message "Log entry/image created successfully" and also contains function Logs)

20. 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.



CloudWatch > Log groups > /aws/lambda/lambda_52_exp12 > 2024/10/10/[$LATEST]a153468b1a574e0a8512ea7220531432
Log events
Actions
Start tailing
Create metric filter
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display

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

```
Timestamp                   Message
                            No older events at this moment. Retry
2024-10-10T13:25:24.595Z    INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2bbe06ceb81ec3bc408a0f277dab104c14cd814b081
2024-10-10T13:25:24.688Z    START RequestId: 5072b72b-1982-4ec5-b1c4-dde4705cd4a7 Version: $LATEST
2024-10-10T13:25:24.689Z    An image has been added to the bucket example-bucket : test%2Fkey
2024-10-10T13:25:24.704Z    END RequestId: 5072b72b-1982-4ec5-b1c4-dde4705cd4a7
2024-10-10T13:25:24.704Z    REPORT RequestId: 5072b72b-1982-4ec5-b1c4-dde4705cd4a7 Duration: 1.97 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 90.56 ms
2024-10-10T13:26:11.981Z    START RequestId: 80470e6d-fce1-48af-ab9a-42ca8000ae4b Version: $LATEST
2024-10-10T13:26:11.982Z    An image has been added to the bucket example-bucket : test%2Fkey
2024-10-10T13:26:11.984Z    END RequestId: 80470e6d-fce1-48af-ab9a-42ca8000ae4b
2024-10-10T13:26:11.984Z    REPORT RequestId: 80470e6d-fce1-48af-ab9a-42ca8000ae4b Duration: 1.48 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB
2024-10-10T13:26:31.220Z    START RequestId: ece2072a-bb6a-4cc3-9065-2594e7d22abc Version: $LATEST
2024-10-10T13:26:31.221Z    An image has been added to the bucket example-bucket : test%2Fkey
2024-10-10T13:26:31.224Z    END RequestId: ece2072a-bb6a-4cc3-9065-2594e7d22abc
2024-10-10T13:26:31.224Z    REPORT RequestId: ece2072a-bb6a-4cc3-9065-2594e7d22abc Duration: 1.65 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB
                            No newer events at this moment. Auto retry paused. Resume
```

Conclusion:

Through this project, I successfully set up a Lambda function and an S3 bucket. After configuring the settings, like adding a description and setting a 1-second timeout, I created a test event named 'test_newevent' and deployed the function without any issues. I also linked the Lambda function to the S3 bucket via a trigger and added a print statement in the code. After uploading an image to the bucket and redeploying, the function returned a status code of 200 with the expected message, and CloudWatch logs captured the entire process.This practical taught me how to effectively manage AWS Lambda settings, set up and test triggers, and monitor logs using CloudWatch. I also gained a deeper understanding of integrating S3 with Lambda to automate tasks.