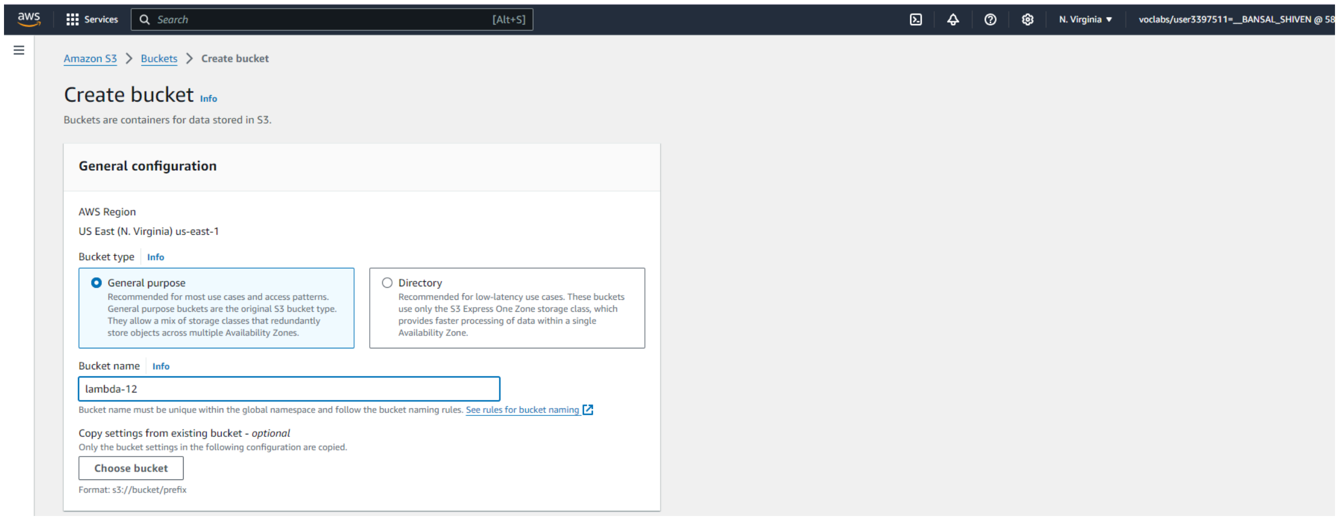
Name: Shiven Bansal   Roll No: 03   Class: D15C

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.

Steps:

Step 1: On your AWS console, click on ‘S3’ in the services section and click on ‘Create bucket’. Give your bucket a name.



Amazon S3 > Buckets > Create bucket

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

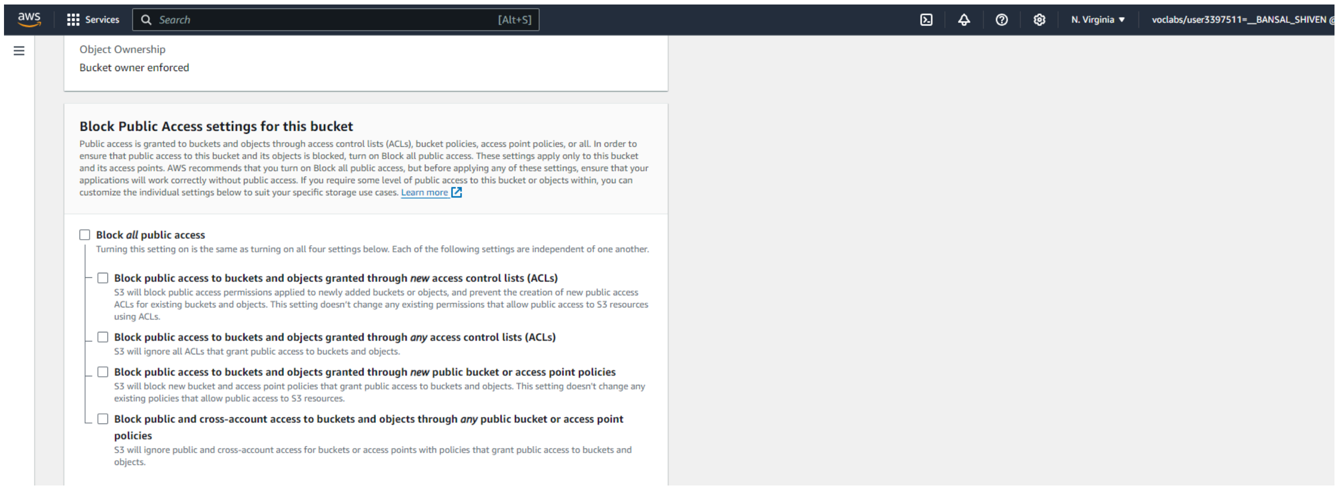
[✓ General purpose] Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

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
lambda-12
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

Uncheck the ‘Block all public access’ box.



Object Ownership
Bucket owner enforced

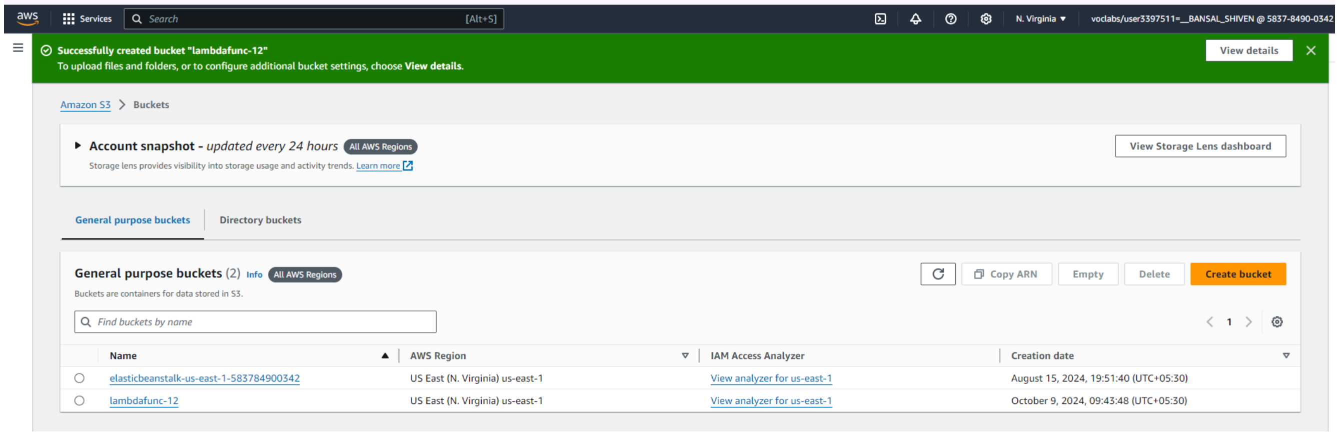
Block Public Access settings for this bucket

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.

Keep all other options as default and click on ‘Create bucket’.



Successfully created bucket "lambdafunc-12"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

Amazon S3 > Buckets

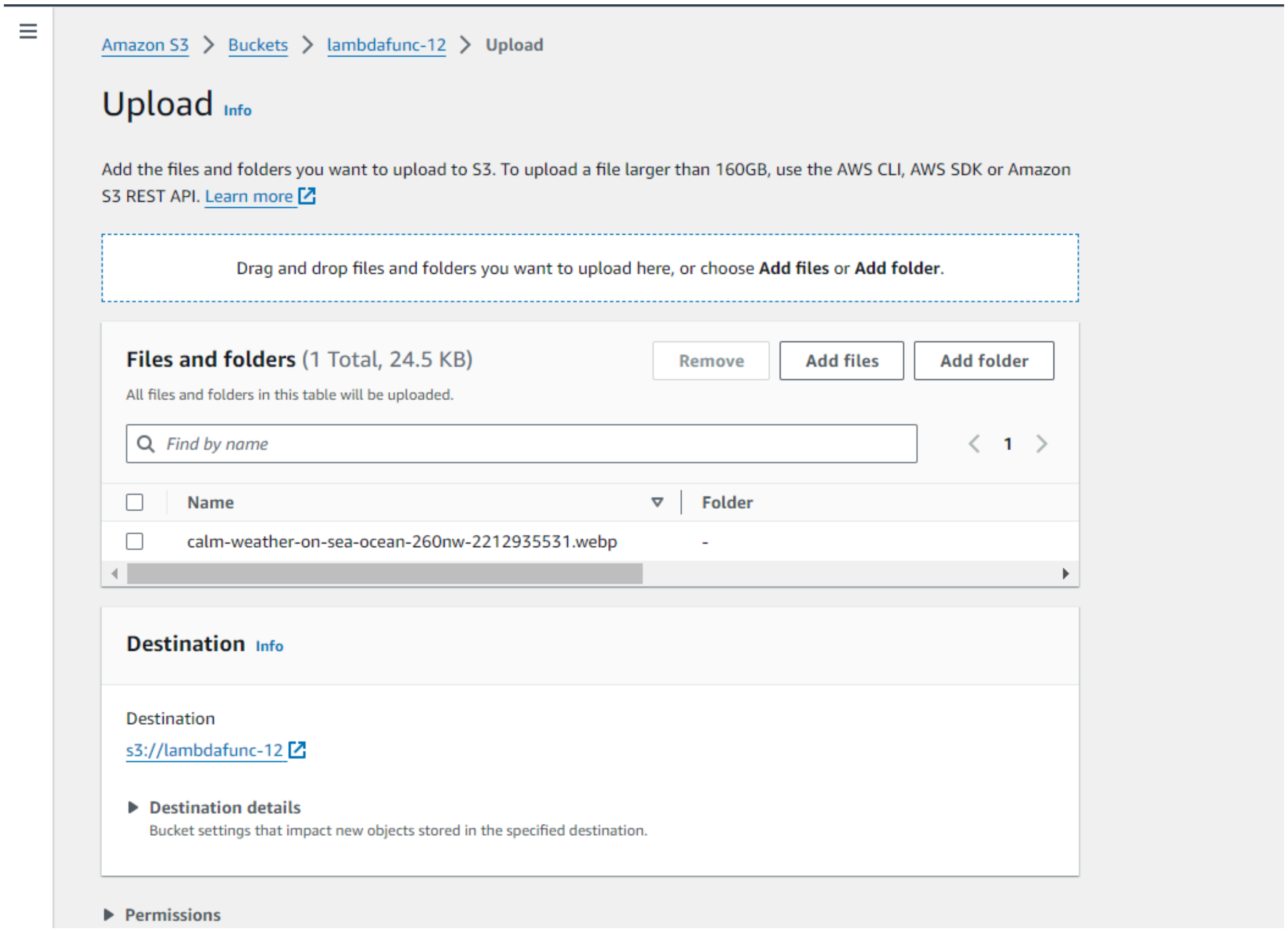
Account snapshot - *updated every 24 hours*
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (2)
Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-583784900342	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 15, 2024, 19:51:40 (UTC+05:30)
lambdafunc-12	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 9, 2024, 09:43:48 (UTC+05:30)

Your bucket is created.

Step 2: Upload an image onto your S3 bucket by clicking on your S3 bucket, clicking on ‘Upload’, clicking on ‘Add files’, navigating to your image and selecting it.



Amazon S3 > Buckets > lambdafunc-12 > 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, 24.5 KB)
All files and folders in this table will be uploaded.

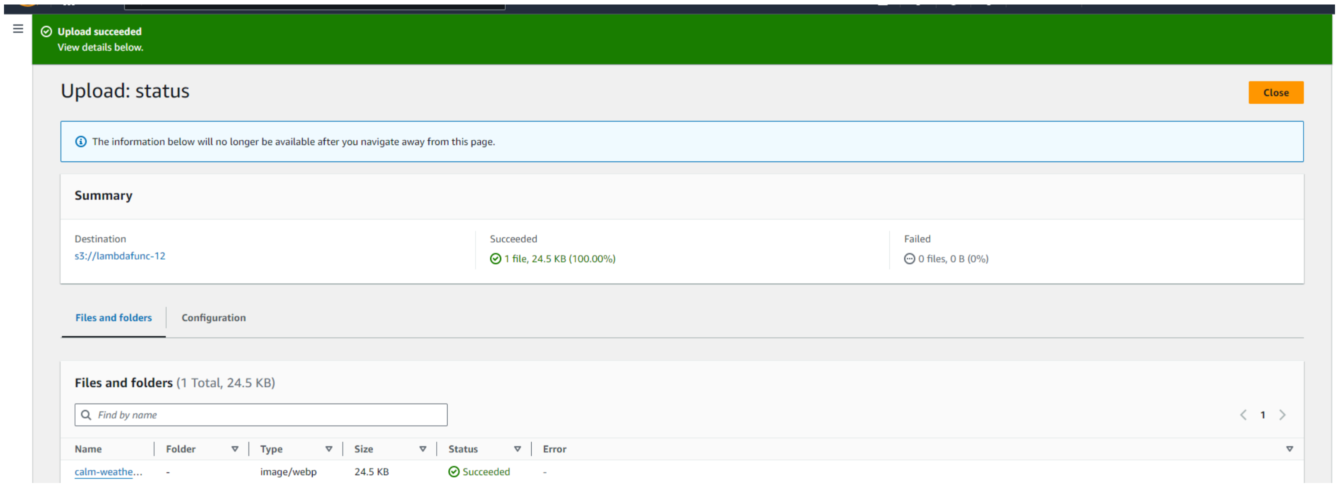
Name	Folder
calm-weather-on-sea-ocean-260nw-2212935531.webp	-

Destination

Destination
s3://lambdafunc-12

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

Permissions


Upload succeeded
View details below.

Upload: status

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

Summary

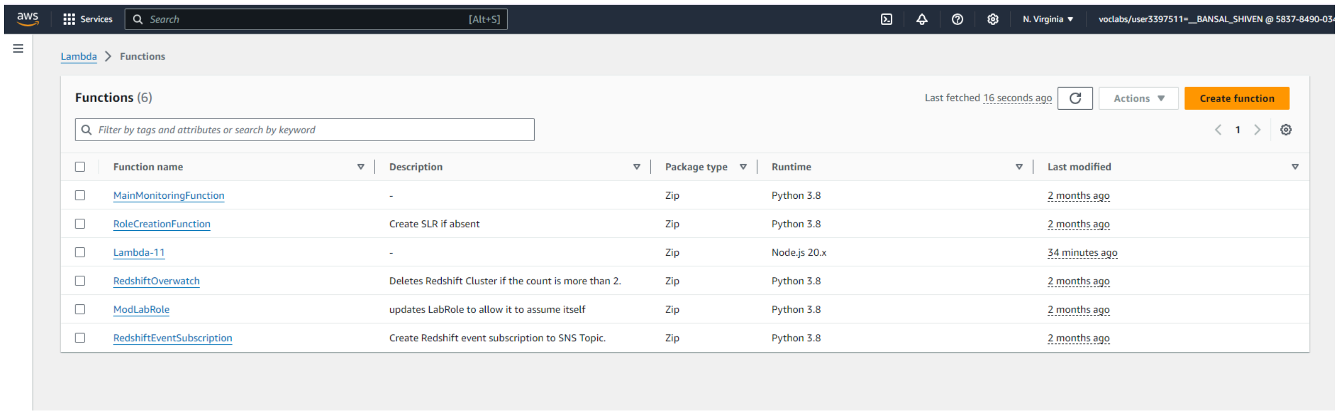
Destination	Succeeded	Failed
s3://lambdafunc-12	1 file, 24.5 KB (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 24.5 KB)

Name	Folder	Type	Size	Status	Error
calm-weathe...	-	image/webp	24.5 KB	Succeeded	-

Your image gets uploaded onto the S3 bucket.

Step 3: Navigate to the AWS Lambda console using the ‘Services’ section. Click on ‘Create function’.

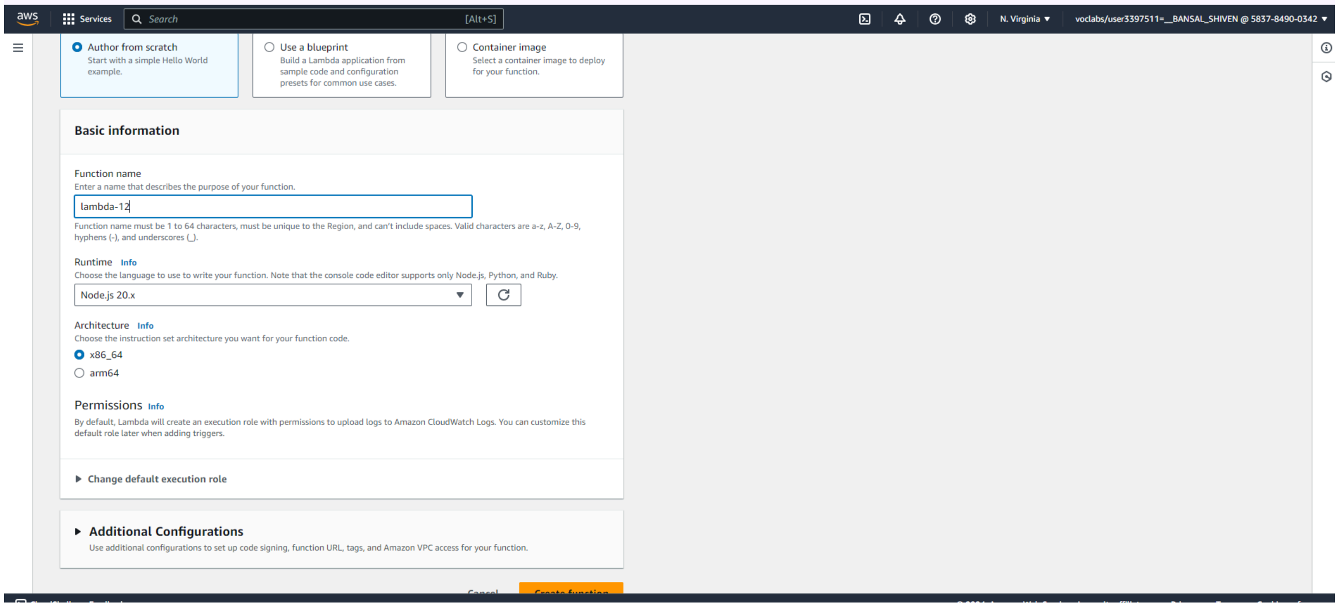

Lambda > Functions

Functions (6)

Last fetched 16 seconds ago

Function name	Description	Package type	Runtime	Last modified
MainMonitoringFunction	-	Zip	Python 3.8	2 months ago
RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	2 months ago
Lambda-11	-	Zip	Node.js 20.x	34 minutes ago
RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	2 months ago
ModLabRole	updates LabRole to allow it to assume itself	Zip	Python 3.8	2 months ago
RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	2 months ago

Step 4: Give your function a name and keep other settings as default.



Author from scratch — Start with a simple Hello World example. (selected)

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-12
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.
Node.js 20.x

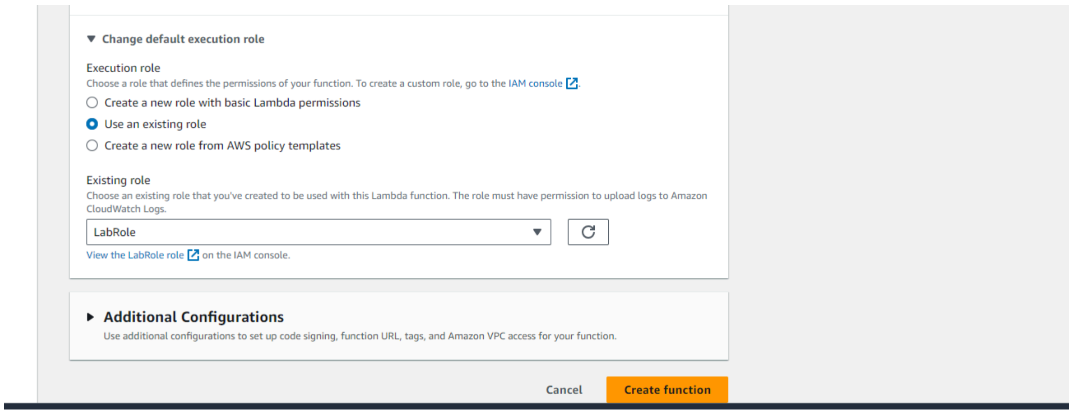
Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

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.

Under 'Execution role', choose 'Use an existing role' and in the dropdown box below, choose 'LabRole'. Then, click on 'Create function'. Your function gets created.



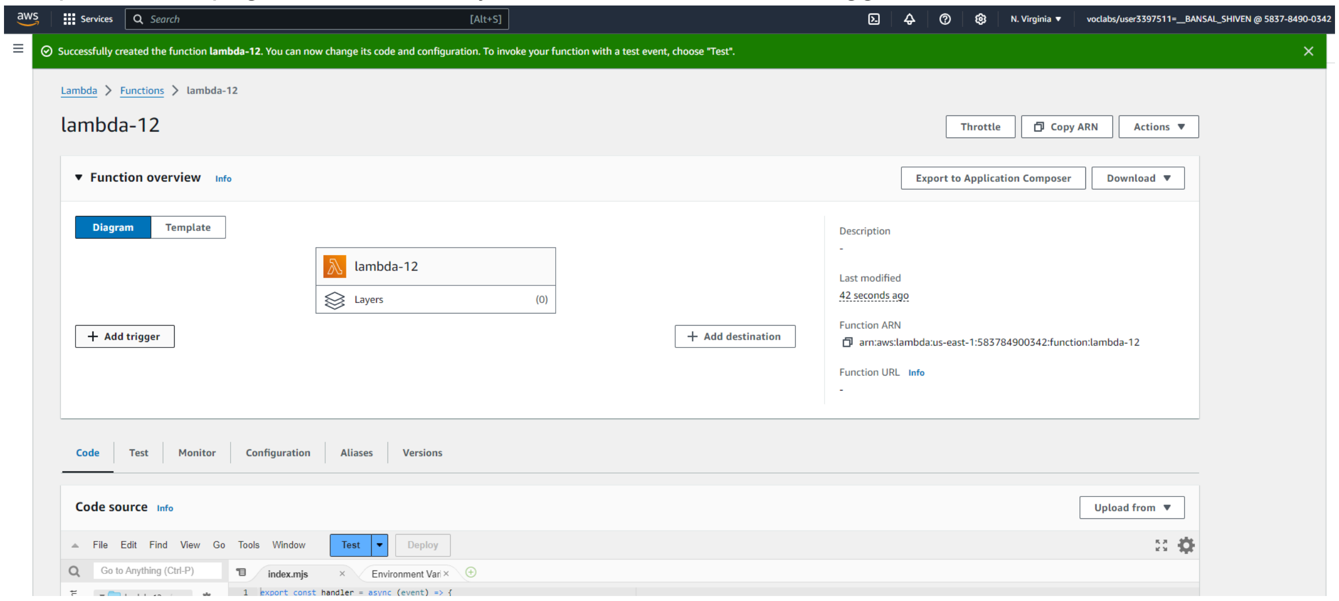
Change default execution role

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.

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

Step 5: On the page of the function you created, click on 'Add trigger'.



Successfully created the function **lambda-12**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > lambda-12

lambda-12

Description
-

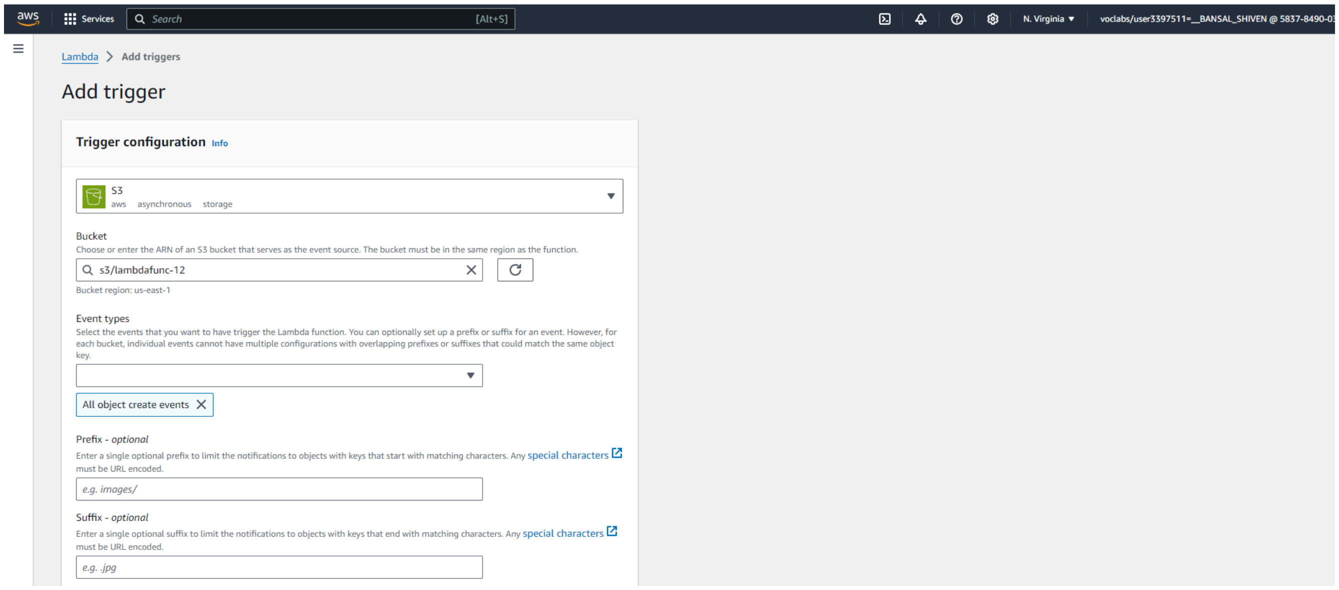
Last modified
42 seconds ago

Function ARN
arn:aws:lambda:us-east-1:583784900342:function:lambda-12

Function URL Info
-

```
export const handler = async (event) => {
```

Step 6: Choose ‘Trigger configuration’ as S3 and select the name of your bucket in the dropdown box below it. Keep other options as default and click on ‘Add’.



Lambda > Add triggers

Add trigger

Trigger configuration Info

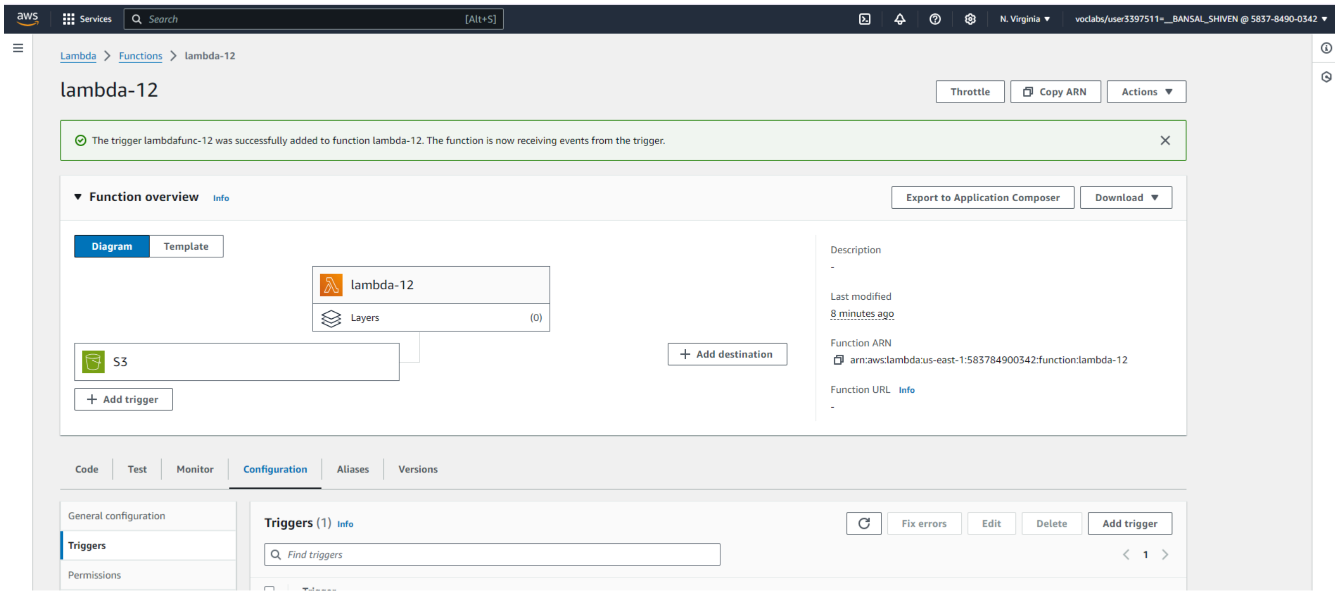
S3
aws asynchronous storage

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.
s3/lambdafunc-12
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.
e.g. images/

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



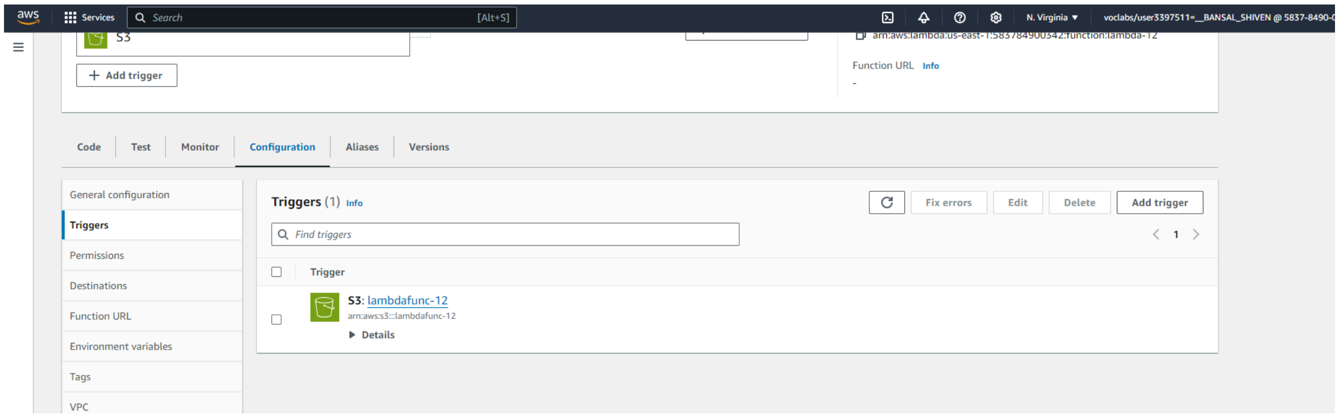
Lambda > Functions > lambda-12

lambda-12

The trigger lambdafunc-12 was successfully added to function lambda-12. The function is now receiving events from the trigger.

Function overview Info

Description
-
Last modified
8 minutes ago
Function ARN
arn:aws:lambda:us-east-1:583784900342:function:lambda-12
Function URL Info
-



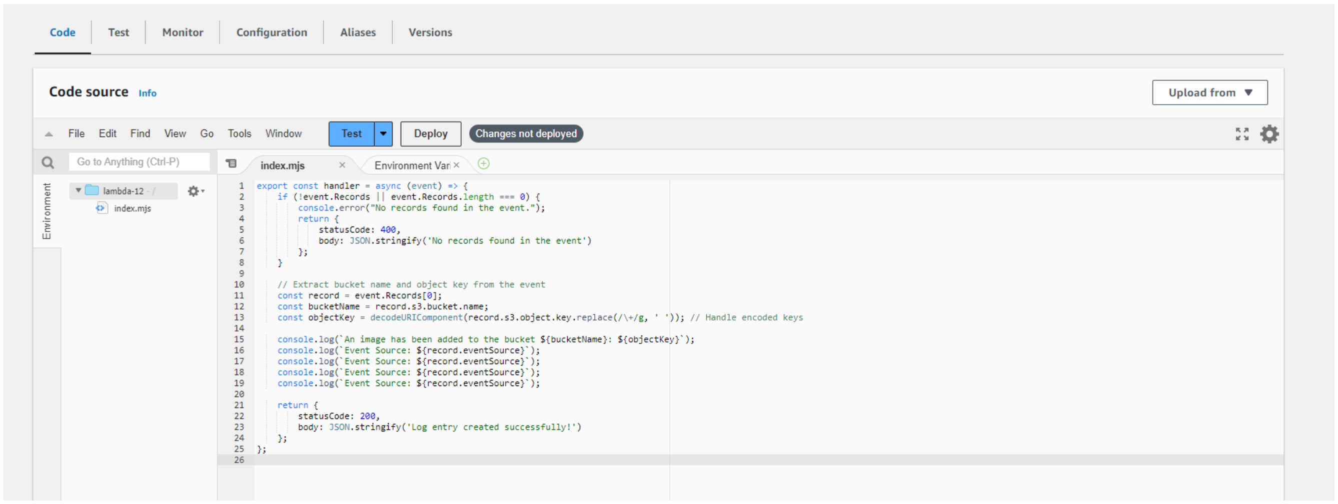
Triggers (1) Info

Trigger
S3: lambdafunc-12
arn:aws:s3:::lambdafunc-12
Details

The trigger gets successfully added to your function.

Step 7: In the ‘Code source’ section of your function, paste the following javascript code instead of the existing code:-

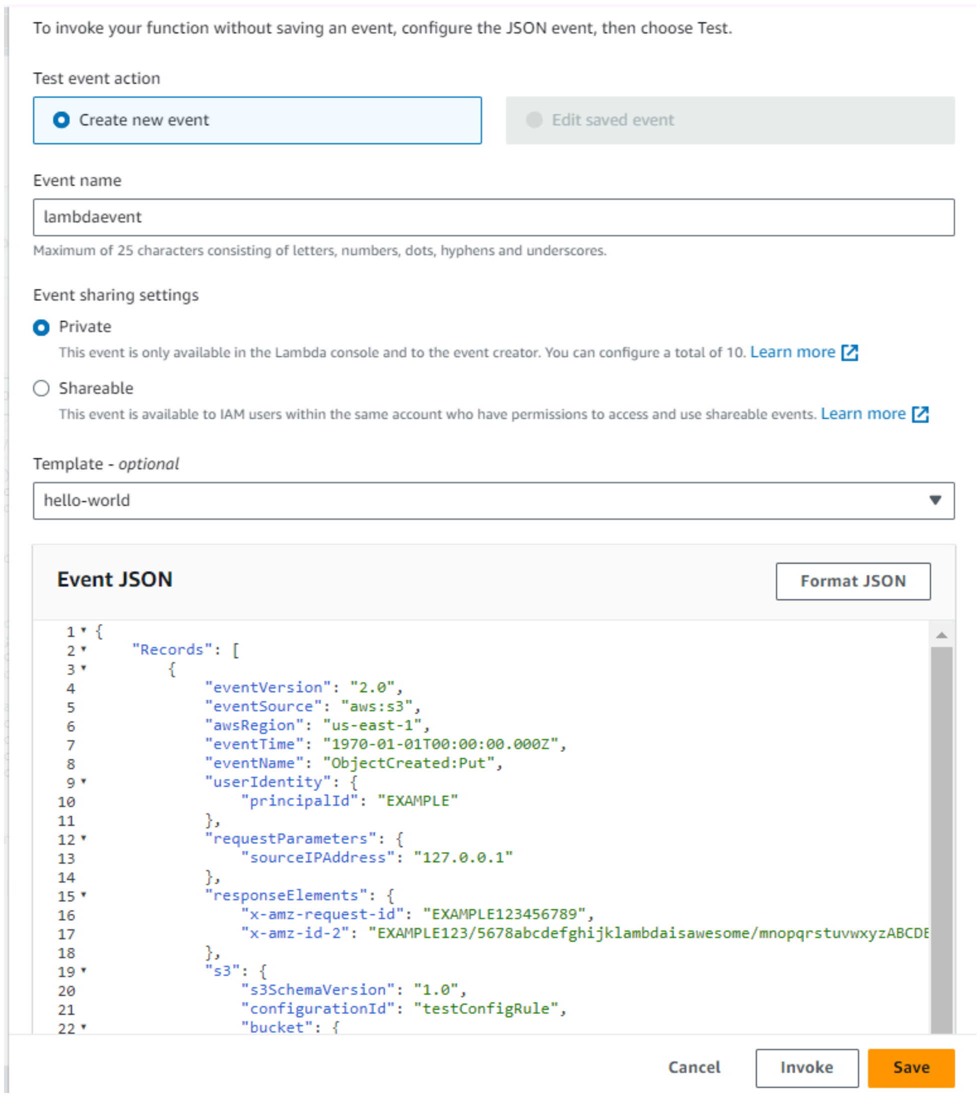
```
export const handler = async (event) => {
   if (!event.Records || event.Records.length === 0) {
      console.error("No records found in the event.");
      return {
         statusCode: 400,
         body: JSON.stringify('No records found in the event')
      };
   }

   // Extract bucket name and object key from the event
   const record = event.Records[0];
   const bucketName = record.s3.bucket.name;
   const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle
encoded keys

   console.log(`An image has been added to the bucket ${bucketName}:
   ${objectKey}`); console.log(`Event Source: ${record.eventSource}`);
   console.log(`Event Source: ${record.eventSource}`);
   console.log(`Event Source: ${record.eventSource}`);
   console.log(`Event Source: ${record.eventSource}`);

   return {
      statusCode: 200,
      body: JSON.stringify('Log entry created successfully!')
   };
};
```



```
export const handler = async (event) => {
    if (!event.Records || event.Records.length === 0) {
        console.error("No records found in the event.");
        return {
            statusCode: 400,
            body: JSON.stringify('No records found in the event')
        };
    }

    // Extract bucket name and object key from the event
    const record = event.Records[0];
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```

Step 8: Click on the arrow next to the ‘Test’ button and click on ‘Configure test event’. In the popup box that appears, if you have an existing event, enter the name of your event or create a new event and in the ‘Event JSON’ section, paste the following code:-

```
{
  "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"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
          "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "test%2Fkey",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```


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

lambdaevent

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*

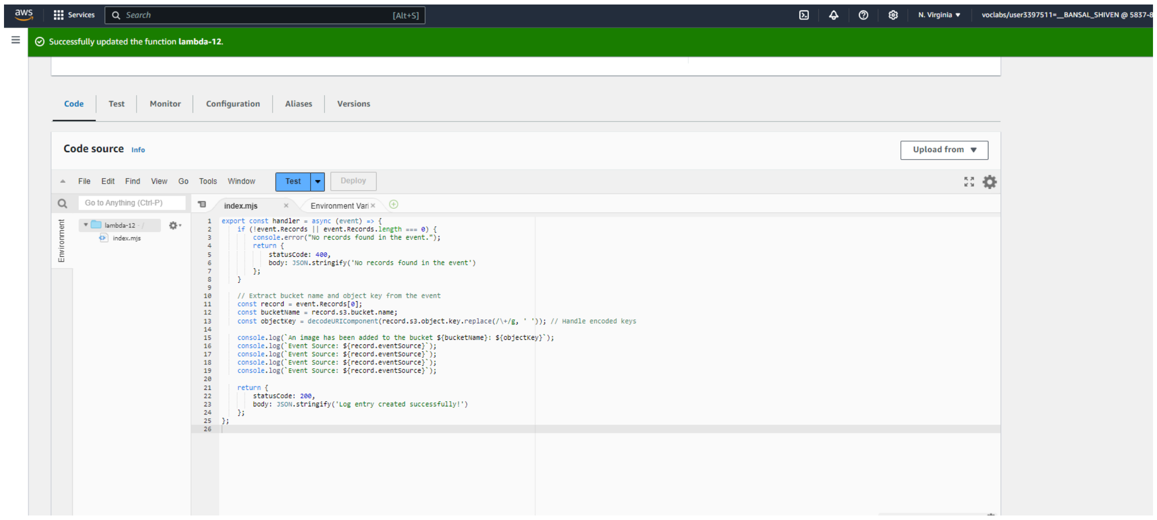
hello-world

Event JSON

```
{
    "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"
            },
            "responseElements": {
                "x-amz-request-id": "EXAMPLE123456789",
                "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDE
            },
            "s3": {
                "s3SchemaVersion": "1.0",
                "configurationId": "testConfigRule",
                "bucket": {
```

Then, click on ‘Save’. Your function gets successfully updated.

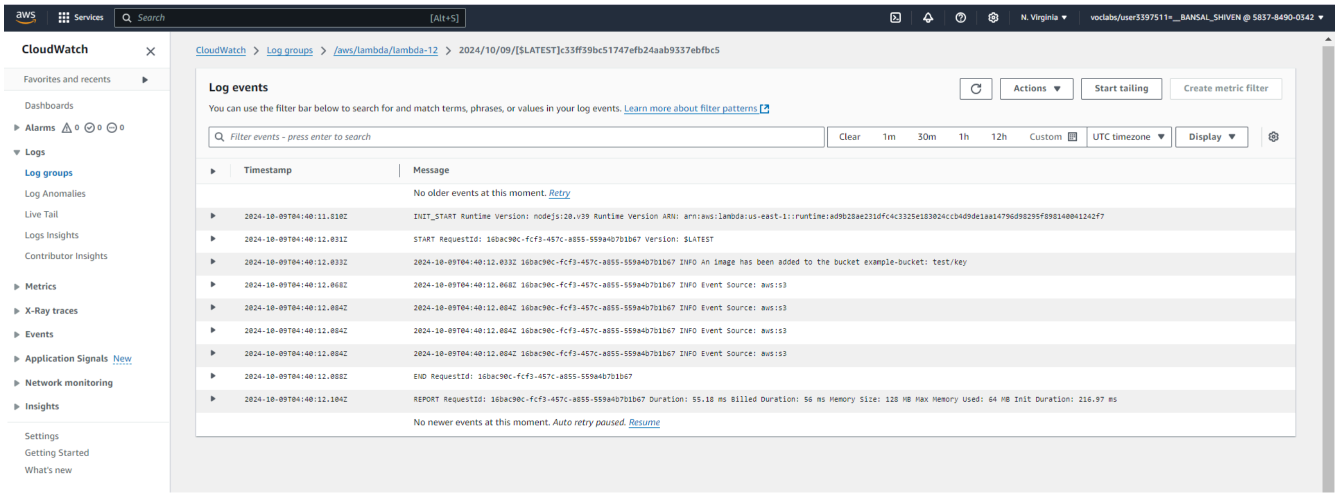
Step 9: Click on ‘Deploy’ and after deployment is successful, click on ‘Test’.



Successfully updated the function **lambda-12**.

```
export const handler = async (event) => {
    if (!event.Records || event.Records.length === 0) {
        console.error("No records found in the event.");
        return {
            statusCode: 400,
            body: JSON.stringify('No records found in the event')
        };
    }

    // Extract bucket name and object key from the event
    const record = event.Records[0];
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```

Running the test gives the above output which displays that ‘An Image has been added to the bucket’ and that the log entry was successfully created.



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
2024-10-09T04:40:11.810Z   INIT_START Runtime Version: nodejs:20.v39 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:ad9b28ae231dfc4c3325e183024ccb4d9de1aa14796d98295f898140041242f7
2024-10-09T04:40:12.031Z   START RequestId: 16bac90c-fcf3-457c-a855-559a4b7b1b67 Version: $LATEST
2024-10-09T04:40:12.033Z   2024-10-09T04:40:12.033Z 16bac90c-fcf3-457c-a855-559a4b7b1b67 INFO An image has been added to the bucket example-bucket: test/key
2024-10-09T04:40:12.068Z   2024-10-09T04:40:12.068Z 16bac90c-fcf3-457c-a855-559a4b7b1b67 INFO Event Source: aws:s3
2024-10-09T04:40:12.084Z   2024-10-09T04:40:12.084Z 16bac90c-fcf3-457c-a855-559a4b7b1b67 INFO Event Source: aws:s3
2024-10-09T04:40:12.084Z   2024-10-09T04:40:12.084Z 16bac90c-fcf3-457c-a855-559a4b7b1b67 INFO Event Source: aws:s3
2024-10-09T04:40:12.084Z   2024-10-09T04:40:12.084Z 16bac90c-fcf3-457c-a855-559a4b7b1b67 INFO Event Source: aws:s3
2024-10-09T04:40:12.088Z   END RequestId: 16bac90c-fcf3-457c-a855-559a4b7b1b67
2024-10-09T04:40:12.104Z   REPORT RequestId: 16bac90c-fcf3-457c-a855-559a4b7b1b67 Duration: 55.18 ms Billed Duration: 56 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 216.97 ms
                           No newer events at this moment. Auto retry paused. Resume
```

Conclusion:

In this experiment, I successfully created an AWS Lambda function that logs "An Image has been added" when an object is uploaded to a specific S3 bucket. I learned how to set up an S3 bucket, configure a Lambda function, and trigger it with S3 events. The function was tested with a simulated event, and it generated the expected log entry, confirming that the function worked as intended. This experiment helped me understand the integration between AWS Lambda and S3 and how to handle real-time event-based processing in AWS.