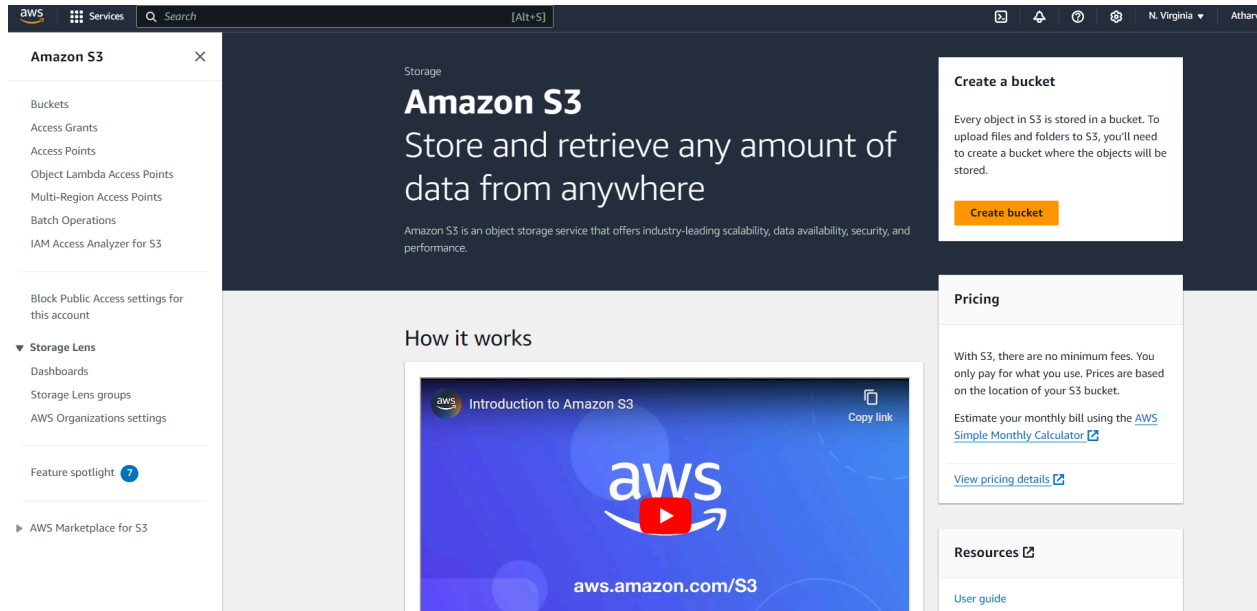


Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Step 1: Create a s3 bucket.

1) Search for S3 bucket in the services search. Then click on create bucket.



2) Keep the bucket as a general purpose bucket. Give a name to your 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](#)

s3lamdaexp11

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

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.

3) Uncheck block all public access

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.

4) Keeping all other options same, click on create. This would create your bucket. Now click on the name of the bucket

Successfully created bucket "s3lamdaexp11"

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](#) [View Storage Lens dashboard](#)

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

General purpose buckets | Directory buckets

General purpose buckets (2) [Info](#) [All AWS Regions](#)

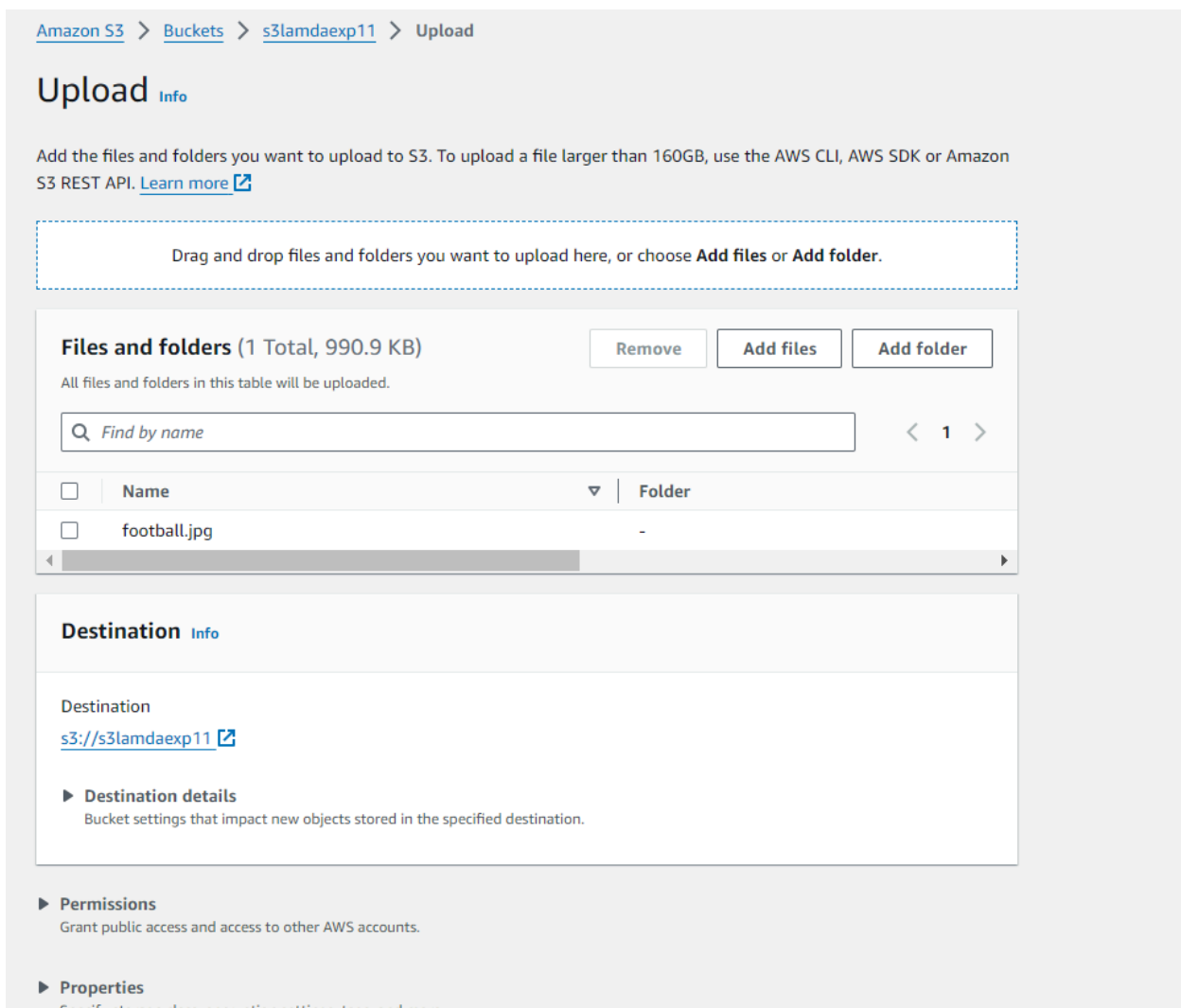
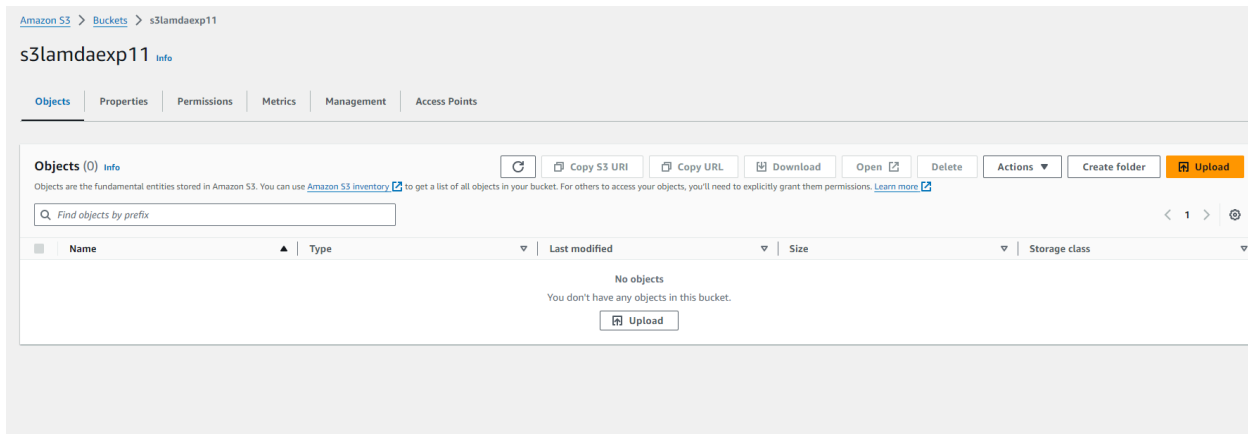
Buckets are containers for data stored in S3.

Find buckets by name

Refresh Copy ARN Empty Delete Create bucket

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	elasticbeanstalk-eu-north-1-010928207735	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 14, 2024, 22:12:26 (UTC+05:30)
<input type="radio"/>	s3lamdaexp11	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 7, 2024, 09:40:50 (UTC+05:30)

5) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload



6) The image has been uploaded to the bucket

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://jslmdaexp11	Succeeded 1 file, 990.9 KB (100.00%)	Failed 0 files, 0 B (0%)
---------------------------------	---	-----------------------------

Files and folders | Configuration

Files and folders (1 Total, 990.9 KB)

Find by name

Name	Folder	Type	Size	Status	Error
football.jpg	-	image/jpeg	990.9 KB	Succeeded	-

Step 2: Configure Lambda function

1) Go to the lambda function you had created berfor. (Services → Lambda → Click on name of function). Here, click on add trigger

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.

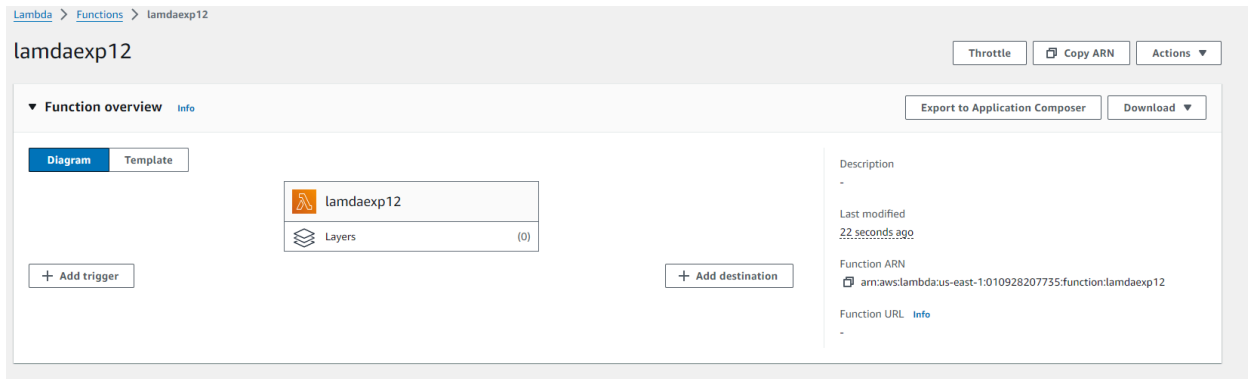
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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

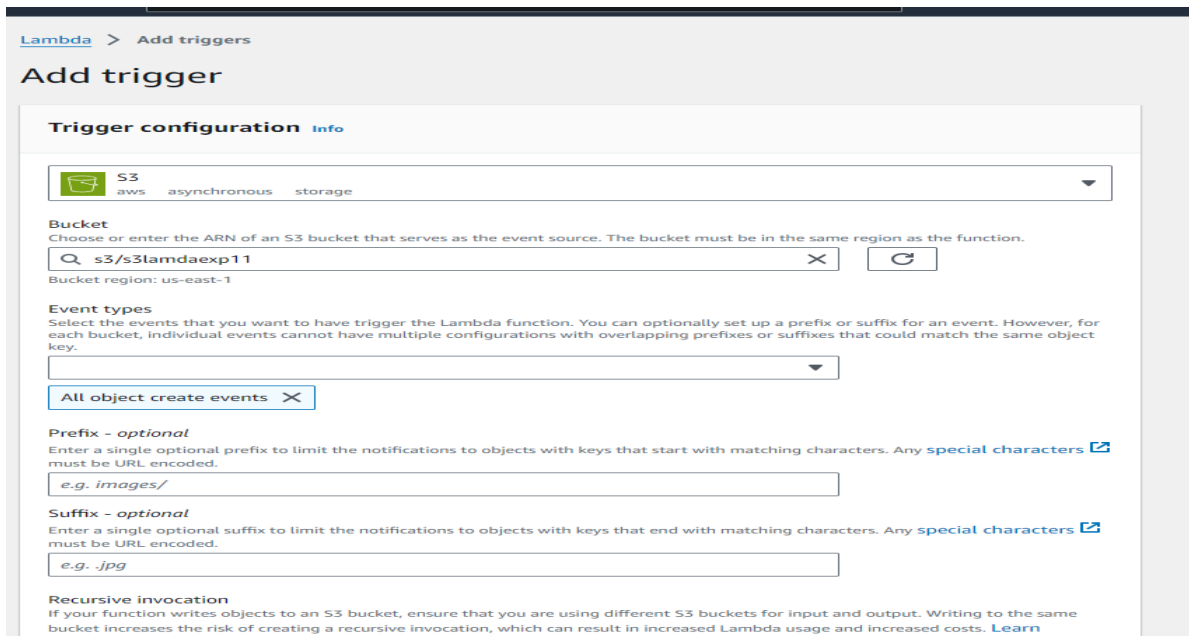
Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

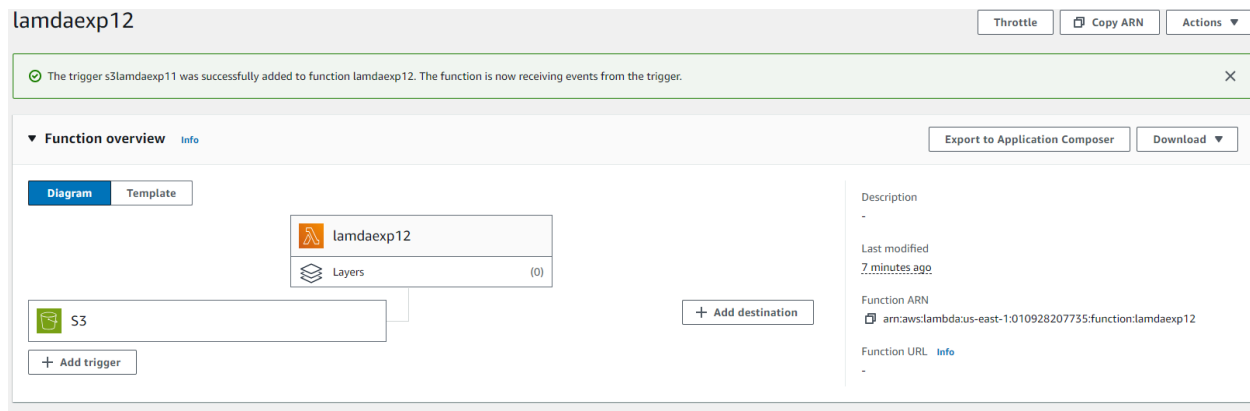
Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
[► Change default execution role](#)

2) Under trigger configuration, search for S3 and select it.



3) Here, select the S3 bucket you created for this experiment. Acknowledge the condition given by AWS. then click on Add. This will add the S3 bucket trigger to your function





4) Scroll down to the code section of the function. Add the following javascript code to the code area by replacing 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!')
  };
};
```

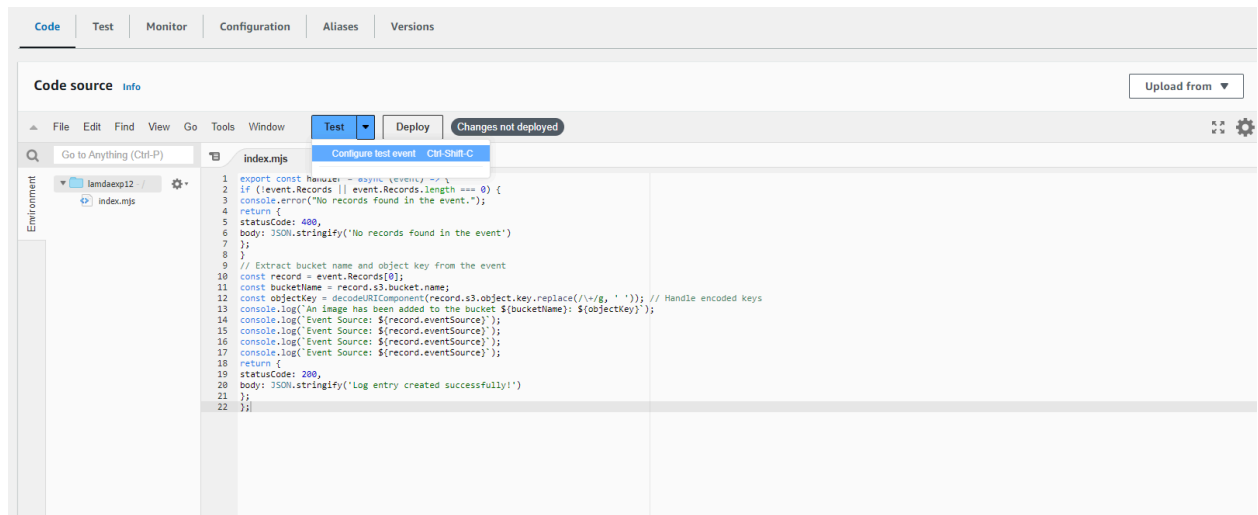
This JSON structure represents an S3 event notification triggered when an object is uploaded to an S3 bucket. It contains details about the event, including the bucket name (example-bucket), the object key (test/key), and metadata like the object's size, the event source (aws:s3), and the event time.



The screenshot shows the AWS Lambda console interface for a function named 'index.mjs'. The 'Code source' tab is selected, displaying the JavaScript code for the handler. The code is a Node.js module that exports an asynchronous handler function. It checks if there are any records in the event, logs the event source, and returns a 200 status code with a success message. The code is as follows:

```
1 export const handler = async (event) => {
2   if (!event.Records || event.Records.length === 0) {
3     console.error("No records found in the event.");
4     return {
5       statusCode: 400,
6       body: JSON.stringify("No records found in the event")
7     };
8   }
9   // Extract bucket name and object key from the event
10  const record = event.Records[0];
11  const bucketName = record.s3.bucket.name;
12  const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys
13  console.log("An image has been added to the bucket ${bucketName}: ${objectKey}");
14  console.log("Event Source: ${record.eventSource}");
15  console.log("Event Source: ${record.eventSource}");
16  console.log("Event Source: ${record.eventSource}");
17  console.log("Event Source: ${record.eventSource}");
18  return {
19    statusCode: 200,
20    body: JSON.stringify("Log entry created successfully!")
21  };
22 };
```

The interface includes a menu bar with File, Edit, Find, View, Go, Tools, and Window. There are buttons for 'Test' and 'Deploy', and a status indicator 'Changes not deployed'. The left sidebar shows the environment 'lamdaexp12' and the file 'index.mjs'. The bottom status bar indicates '22:3 JavaScript Spaces: 2'.



This screenshot is similar to the previous one, but with a context menu open over the 'index.mjs' file in the left sidebar. The menu options are 'Configure test event' and 'Ctrl-Shift-C'. The code source view remains the same, showing the JavaScript handler code. The interface elements like the menu bar, buttons, and status bar are consistent with the previous screenshot.

☒ 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   "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",
```

1:1 JSON Spaces: 2

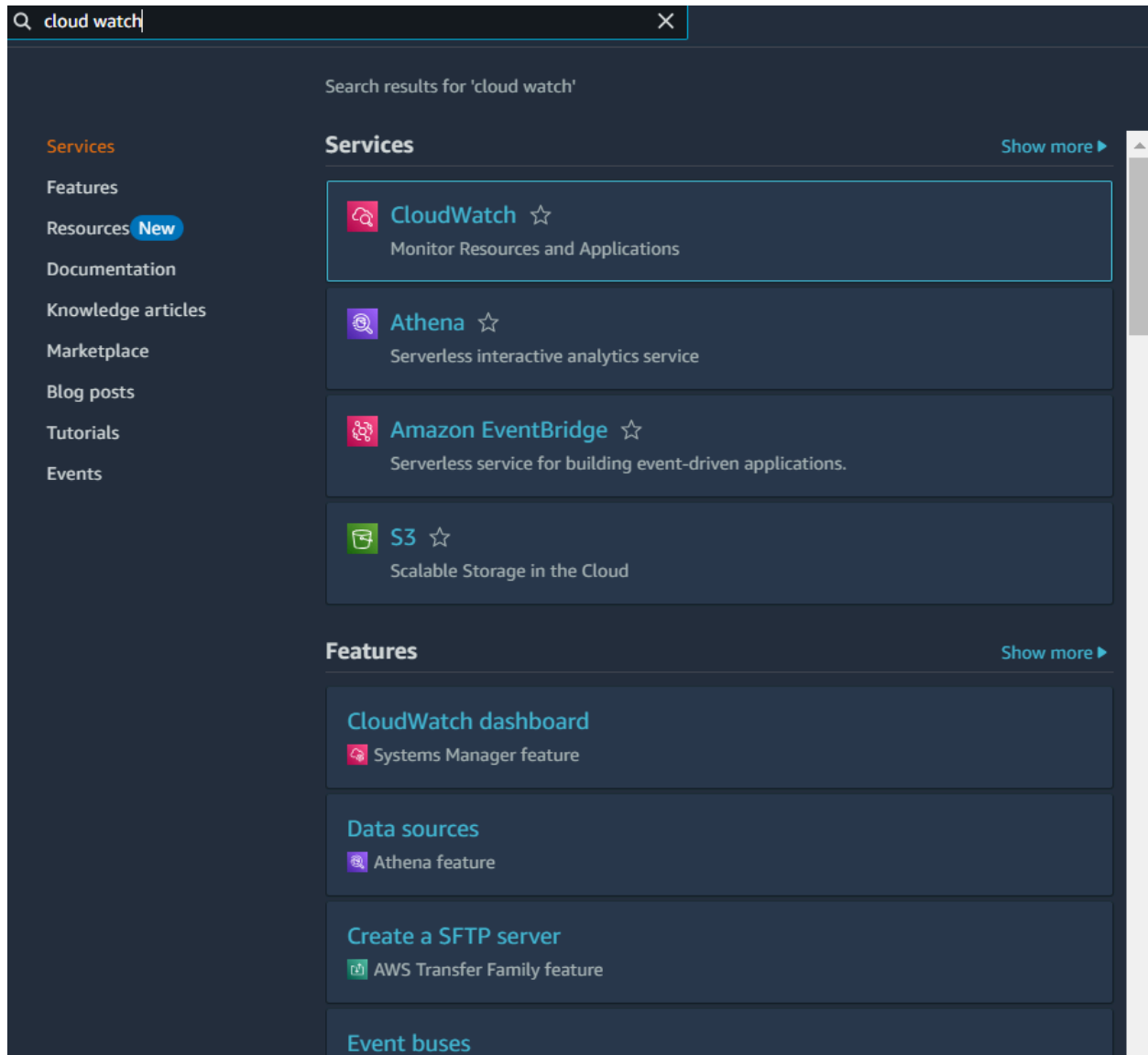
[Cancel](#)[Invoke](#)[Save](#)

© The test event myevent1 was successfully saved.

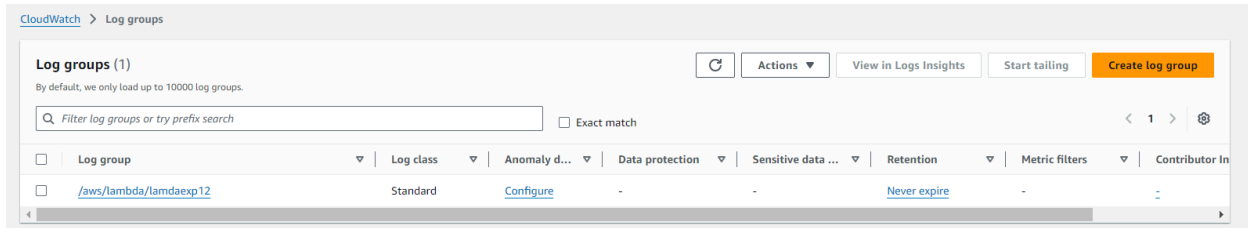
[Add reference](#)[Function URL](#) [Info](#)

Step 3: Check the logs

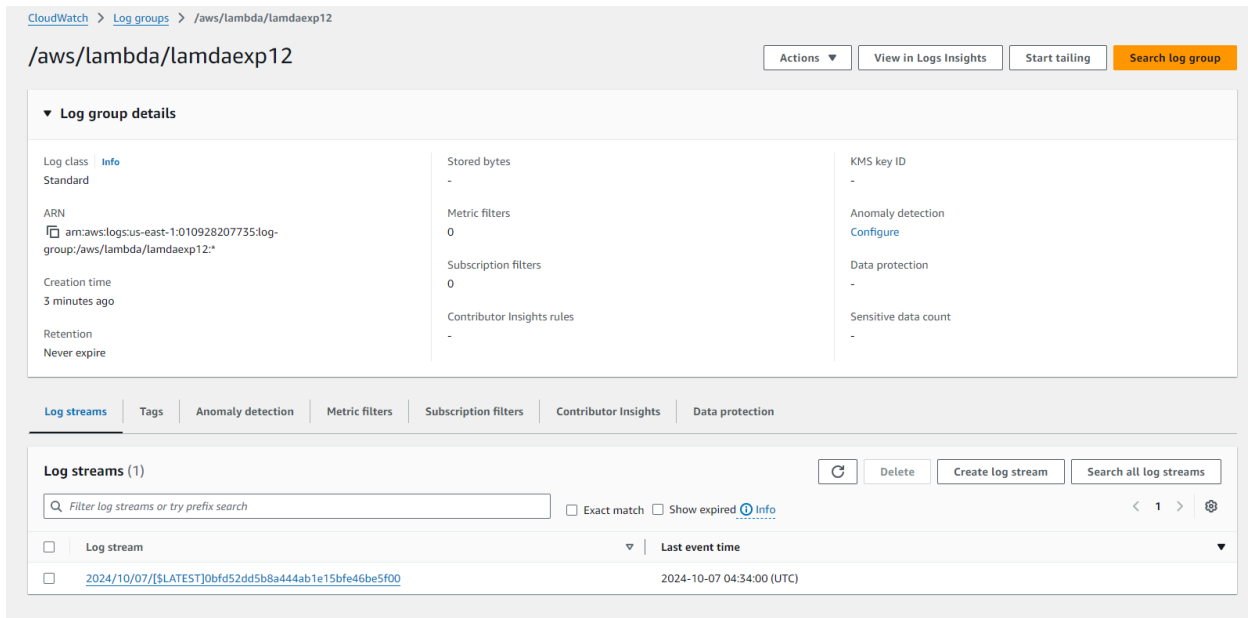
1) To check the logs explicitly, search for CloudWatch on services and open it in a new tab



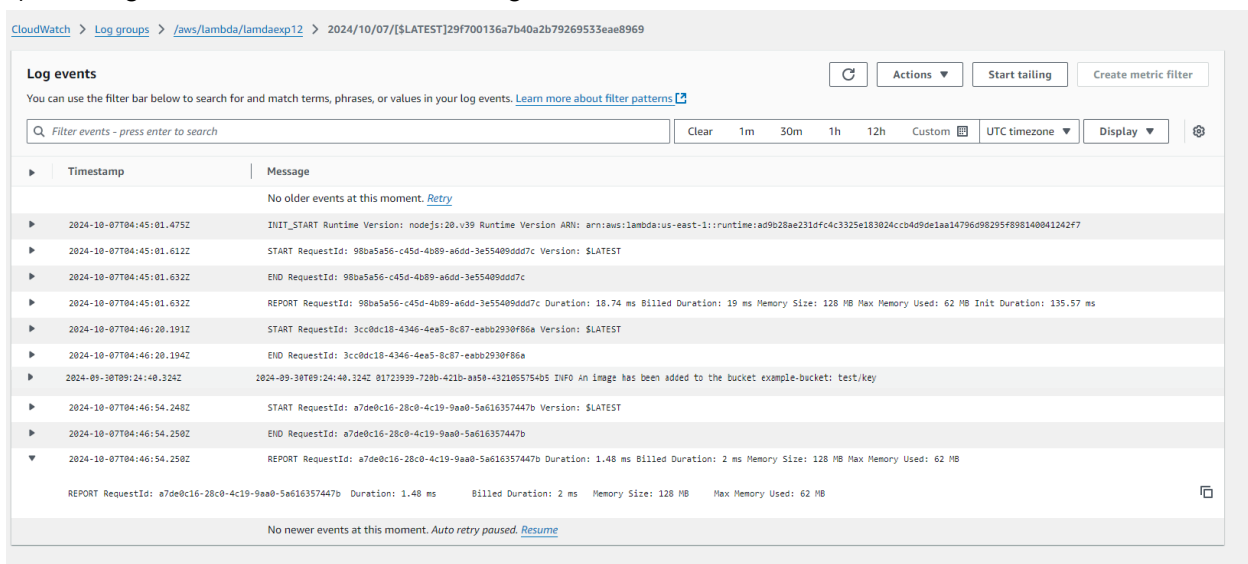
2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.



3) Here, under Log streams, select the log stream you want to check.



4) Here again, we can see that 'An image has been added to the bucket'.



Conclusion: In this experiment, we successfully implemented a Lambda function that automatically logs a message whenever an image is uploaded to a designated S3 bucket. By setting up an S3 bucket trigger, we showcased how AWS services can efficiently interact to automate workflows. The Lambda function captured essential event details, such as the bucket name and object key. After deploying the function, we tested it with a sample image upload and confirmed through CloudWatch logs that the function correctly identified and logged the event. This experiment demonstrated the smooth integration of AWS Lambda with S3 for handling event-driven automation.