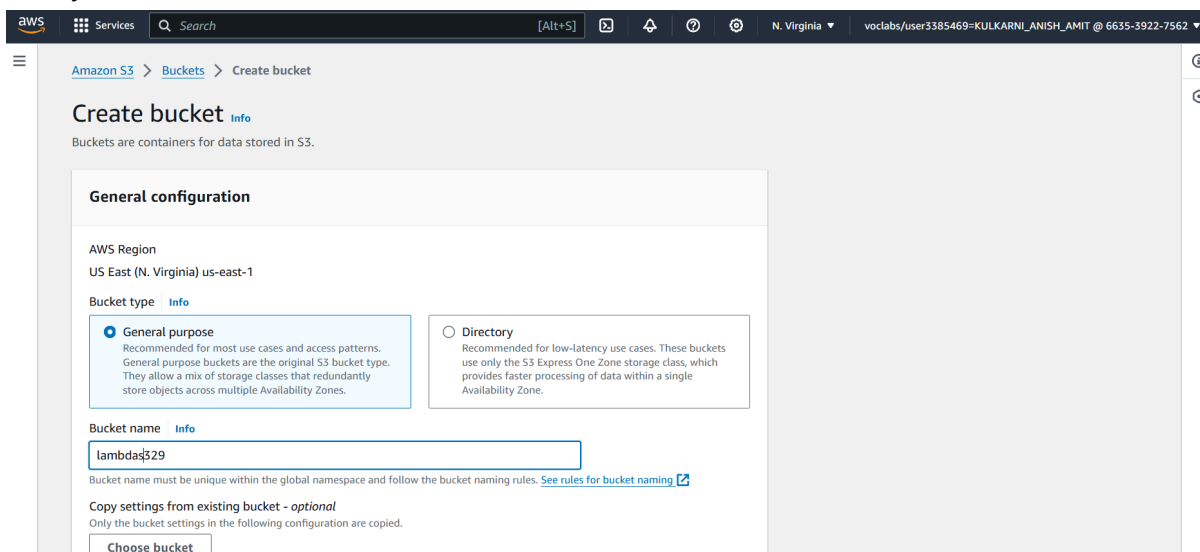


Experiment 12

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.



aws Services Search [Alt+S] N. Virginia voclabs/user3385469=KULKARNI_ANISH_AMIT @ 6635-3922-7562

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.

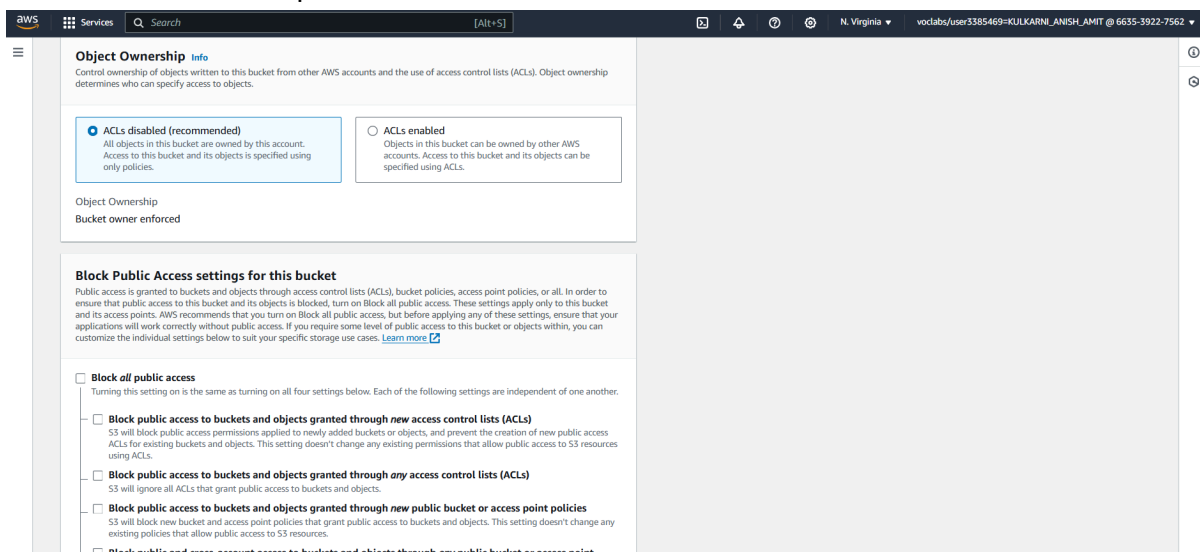
Bucket name [Info](#)
lambda\$29

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](#)

Uncheck the ‘Block all public access’ box.



aws Services Search [Alt+S] N. Virginia voclabs/user3385469=KULKARNI_ANISH_AMIT @ 6635-3922-7562

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.

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**

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

Successfully created bucket "lambda329"
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](#) [Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3.

Find buckets by name

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	elasticbeanstalk-us-east-1-663539227562	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 5, 2024, 13:54:08 (UTC+05:30)
<input type="radio"/>	lambda329	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 30, 2024, 14:41:52 (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 > lambda329 > 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, 75.1 KB) [Remove](#) [Add files](#) [Add folder](#)
All files and folders in this table will be uploaded.

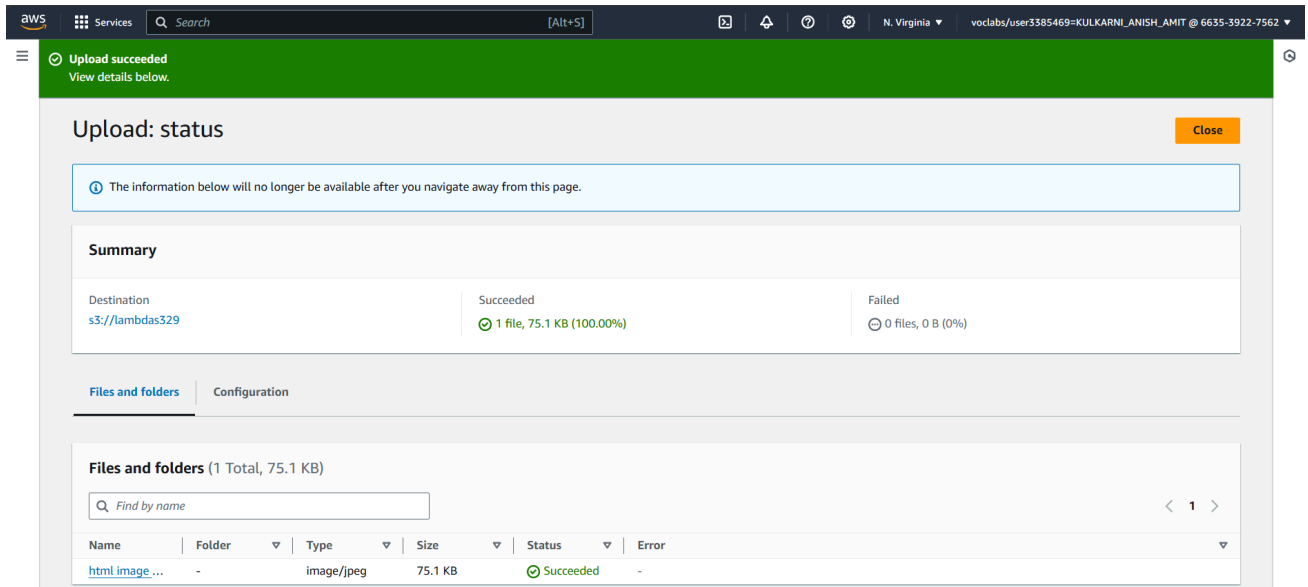
Find by name

<input type="checkbox"/>	Name	Folder	Type
<input type="checkbox"/>	html image 1.jpg	-	image/jpeg

Destination [Info](#)

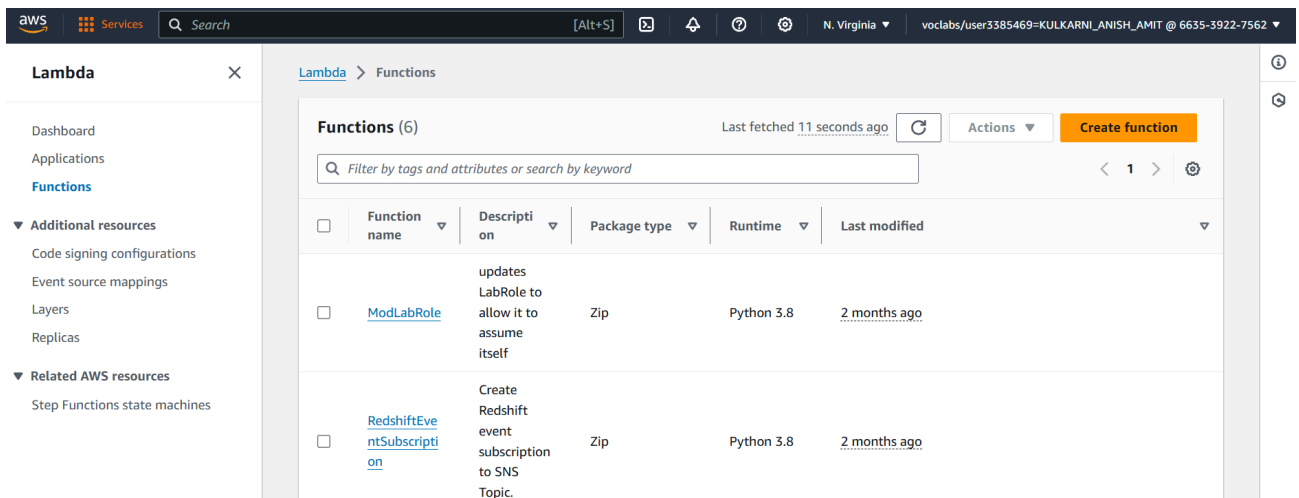
Destination
[s3://lambda329](#)

[Destination details](#)



Your image gets uploaded onto the S3 bucket.

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



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

The screenshot shows the 'Create function' page in the AWS Lambda console. The 'Author from scratch' option is selected. The 'Basic information' section is expanded, showing the function name 'Lambda29', runtime 'Node.js 20.x', and architecture 'x86_64'. The 'Permissions' section is also visible.

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.

Use only letters, numbers, hyphens, or underscores with no spaces.

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

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

Permissions [Info](#)

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.

The screenshot shows the 'Permissions' section of the 'Create function' page. The 'Change default execution role' section is expanded, showing the 'Execution role' options. The 'Use an existing role' option is selected, and 'LabRole' is chosen in the dropdown.

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

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.
 [Refresh](#)
[View the LabRole role](#) on the IAM console.

► Advanced settings

[Cancel](#) [Create function](#)

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

The screenshot shows the 'Lambda29' function overview page in the AWS Lambda console. The 'Function overview' section is expanded, showing the function name, layers, and triggers. The 'Add trigger' button is visible.

Lambda29 [Throttle](#) [Copy ARN](#) [Actions](#)

▼ Function overview [Info](#)

[Export to Application Composer](#) [Download](#)

Diagram **Template**

Lambda29

Layers (0)

[+ Add trigger](#) [+ Add destination](#)

Description
-

Last modified
27 minutes ago

Function ARN
 arn:aws:lambda:us-east-1:663539227562:function:Lambda29

Function URL [Info](#)
-

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

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'.

The screenshot shows the 'Add trigger' configuration page in the AWS Lambda console. The 'Trigger configuration' section is active, showing 'S3' as the event source. The 'Bucket' field contains 's3/lambda329'. The 'Event types' dropdown is set to 'All object create events'. The 'Prefix - optional' field is empty. The 'Suffix - optional' field is also empty.

The screenshot shows the 'Lambda29' function overview page. A green notification bar at the top states: 'The trigger lambda329 was successfully added to function Lambda29. The function is now receiving events from the trigger.' The 'Function overview' section shows a diagram with 'Lambda29' and 'Layers' connected to 'S3'. The 'Add destination' button is visible. The 'Function overview' section also includes a description, last modified time (31 minutes ago), function ARN, and function URL.

The screenshot shows the 'Configuration' page for the 'Lambda29' function. The 'Triggers' tab is selected, showing a list of triggers. The 'S3: lambda329' trigger is listed with its details. The 'Add trigger' button is visible at the bottom right of the triggers list.

The trigger gets successfully added to your function.

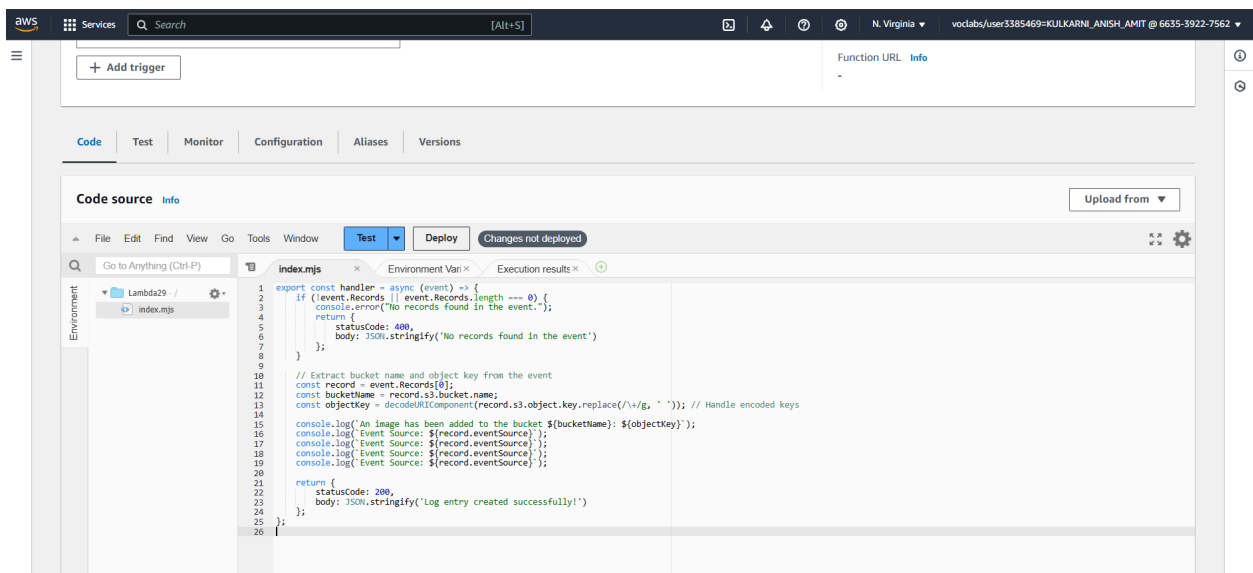
Step 7: In the 'Code source' section of your function, paste the following javascript code instead of the existing code:-

```
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!')
  };
};
```



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":
"EXAMPLE123/5678abcdefghijklmbdaisawesome/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"
        }
      }
    }
  ]
}
```


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				
Filter events - press enter to search		Clear	1m	30m
		1h	12h	Custom
		UTC timezone	Display	
Timestamp	Message			
	There are older events to load. Load more			
2024-09-30T09:24:48.172Z	INIT_START Runtime Version: nodejs128.v25 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:da382f678644e0d7b5702cf2208cc3dc09906126442717fccc4267107033			
2024-09-30T09:24:48.322Z	START RequestID: 01723939-7200-4210-8850-432105575405 Version: SLATEST			
2024-09-30T09:24:48.324Z	2024-09-30T09:24:48.324Z 01723939-7200-4210-8850-432105575405 INFO An image has been added to the bucket example-bucket: test/key			
2024-09-30T09:24:48.334Z	2024-09-30T09:24:48.334Z 01723939-7200-4210-8850-432105575405 INFO Event Source: aws:s3			
2024-09-30T09:24:48.334Z	2024-09-30T09:24:48.334Z 01723939-7200-4210-8850-432105575405 INFO Event Source: aws:s3			
2024-09-30T09:24:48.334Z	2024-09-30T09:24:48.334Z 01723939-7200-4210-8850-432105575405 INFO Event Source: aws:s3			
2024-09-30T09:24:48.334Z	2024-09-30T09:24:48.334Z 01723939-7200-4210-8850-432105575405 INFO Event Source: aws:s3			
2024-09-30T09:24:48.352Z	END RequestID: 01723939-7200-4210-8850-432105575405			
2024-09-30T09:24:48.352Z	REPORT RequestID: 01723939-7200-4210-8850-432105575405 Duration: 29.56 ms Billed Duration: 30 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 147.32 ms			
No newer events at this moment. Auto retry paused. Resume				

Conclusion: In this experiment, we learned how to create a Lambda function which logs “An image has been added” once we add an image to our specific S3 bucket. We first created an S3 bucket and uploaded an image to it. We then created a Lambda function and added an S3 trigger to it and selected the S3 bucket we created. Then, we configured the ‘Code section’ of our Lambda function and a test event for our Lambda function. On running the test event, we observed that it logged important information about the event such as the bucket name and object key and also verified that an image had been added to the S3 bucket. Also, a log of our Lambda function was also created which confirmed that the image had been successfully added to the bucket.