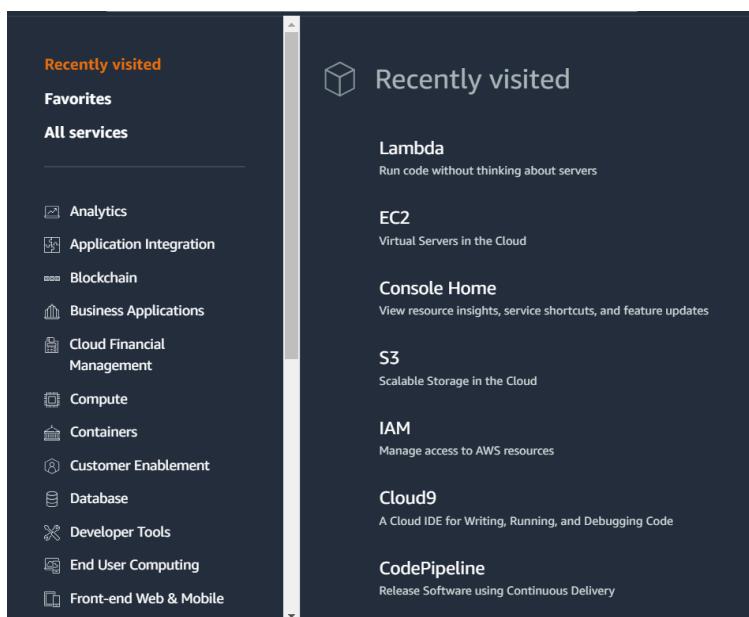


Experiment No :12

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

CREATING LAMBDA FUNCTION :

Step 1: Log in to your AWS Personal account. Then go to S3 in the services menu and click on "Create S3 Bucket."



The screenshot shows the Amazon S3 console with the 'General purpose buckets' tab selected. There are four buckets listed:

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-380557944473			August 5, 2024, 14:37:53 (UTC+05:30)
my-video-stream			September 28, 2024, 20:18:57 (UTC+05:30)
www.ingmaker.com			August 4, 2024, 17:55:21 (UTC+05:30)
www.mywebsite.com			July 28, 2024, 18:09:18 (UTC+05:30)

Step 2: Give your bucket a name, select "General purpose project," then uncheck "Block public access." Keep the other settings as they are.

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](#)
shivam41

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

Given proper bucket Name

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](#)
Lab12_Shivam

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

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

Selected Appropriate object Ownership

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)**
- Block public access to buckets and objects granted through any access control lists (ACLs)**
- Block public access to buckets and objects granted through new public bucket or access point policies**
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**

⚠️ 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.

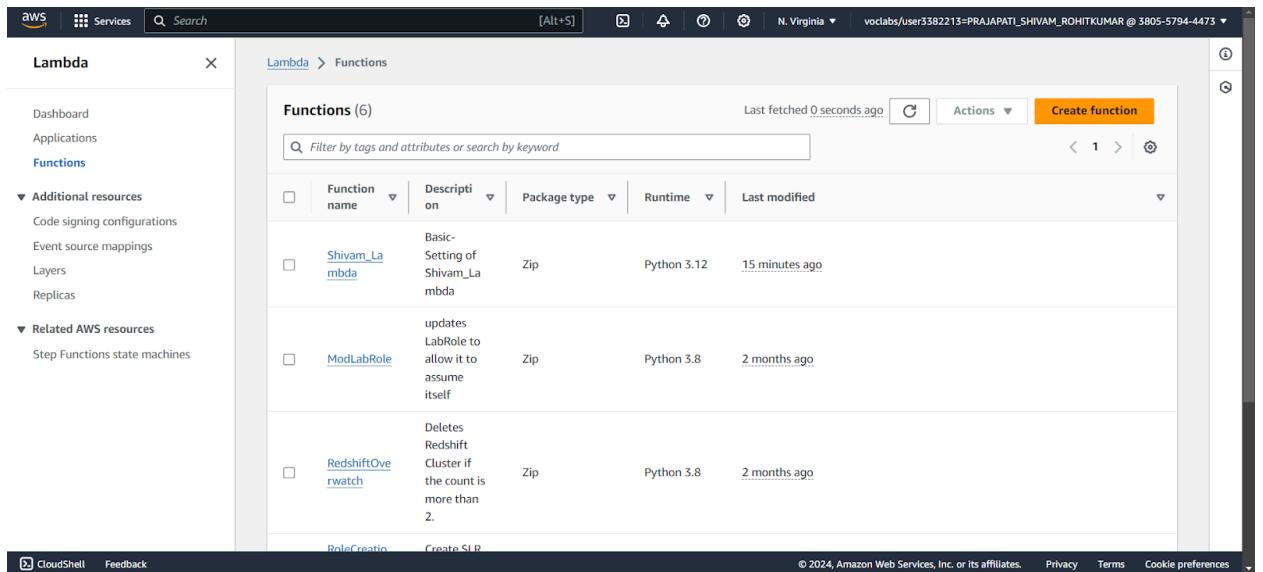
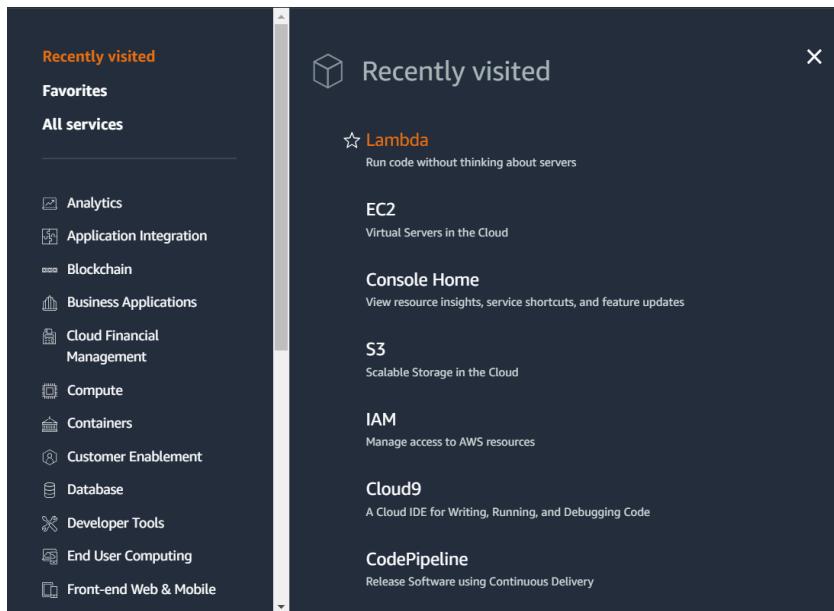
I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Unchecked the block all public access checkbox and checked the lower checkbox.

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-38055794473	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 5, 2024, 14:37:53 (UTC+05:30)
my-video-stream	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 28, 2024, 20:18:57 (UTC+05:30)
shivam41	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 30, 2024, 14:49:48 (UTC+05:30)
www.kingmaker.com	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 4, 2024, 17:55:21 (UTC+05:30)
www.mywebsite.com	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 28, 2024, 18:09:18 (UTC+05:30)

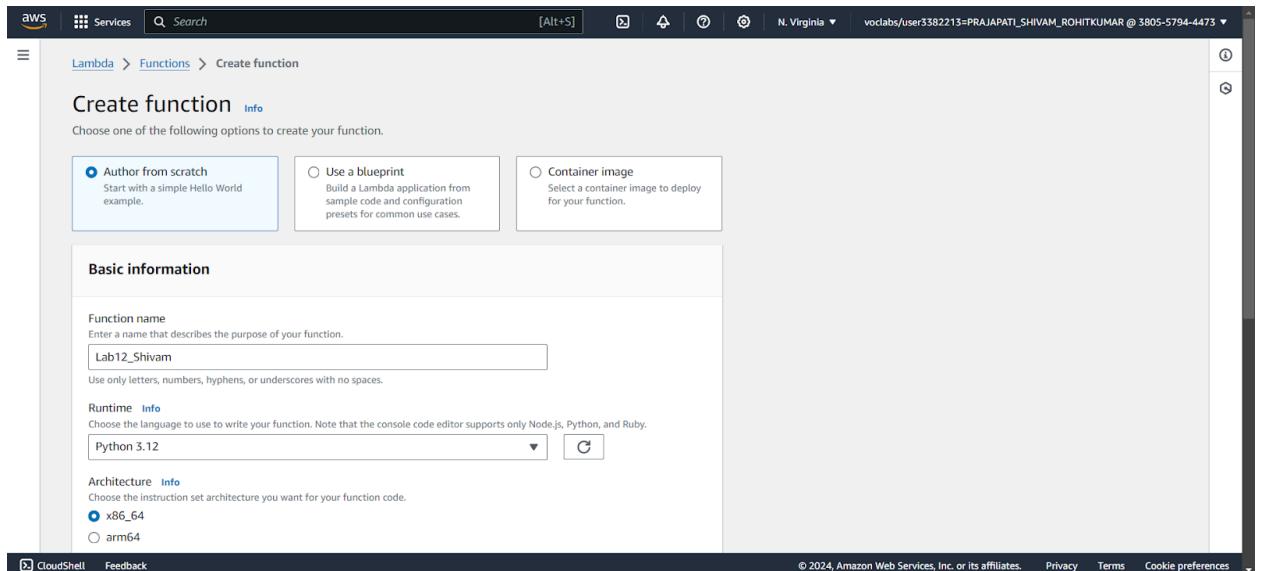
Successfully created the bucket

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

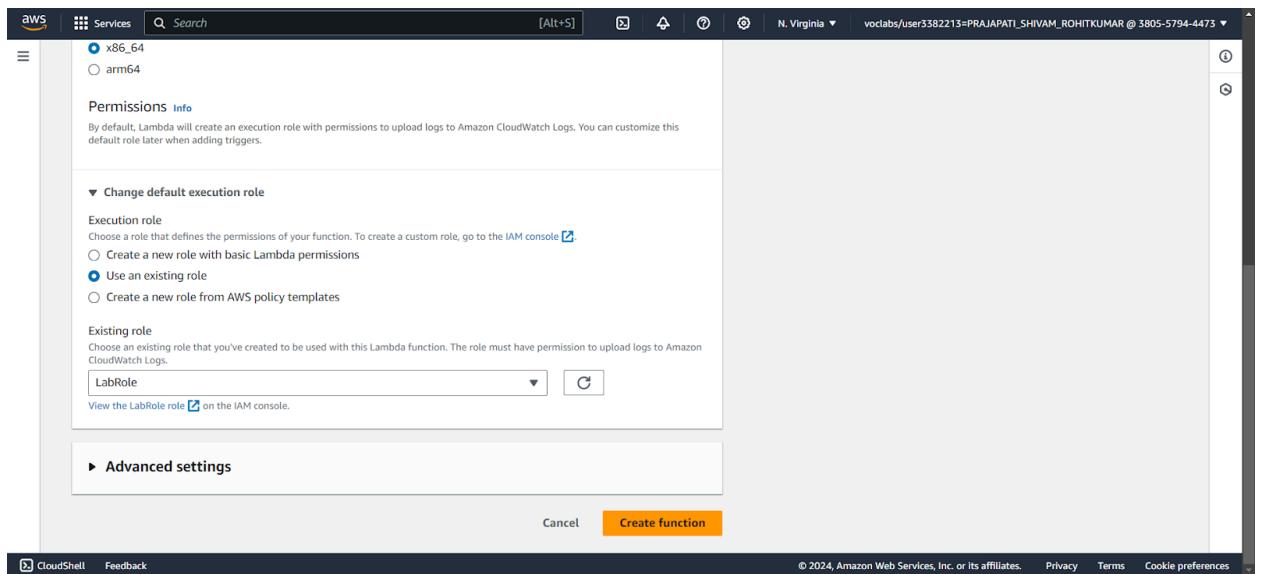


Step 4: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen **Python 3.12**. Set the **architecture to x86**. For the execution role, select '**Use an existing role**', then pick '**Lab role**' from the dropdown menu under existing roles .

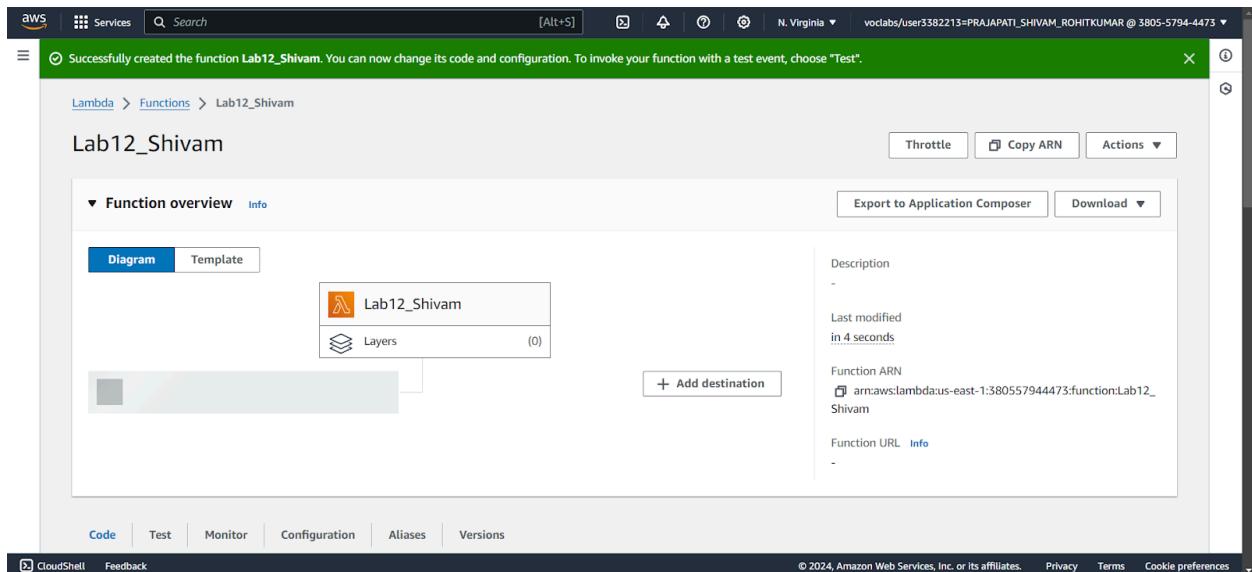
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)



Given proper function name and selected language for Lambda function

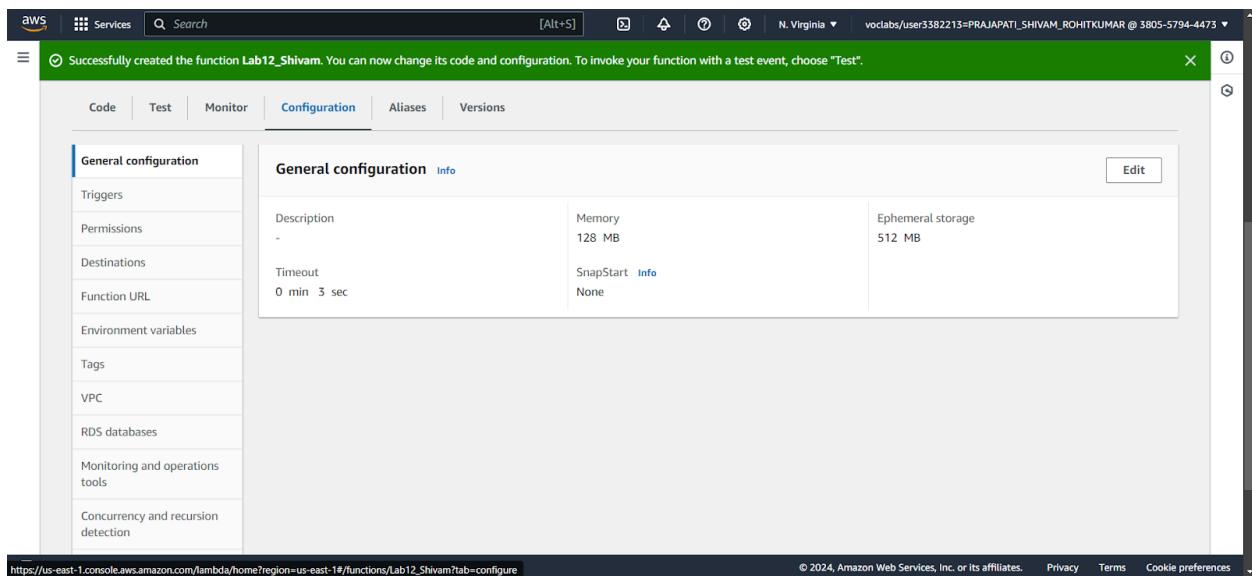


Selected Appropriate Execution role



Successfully created the Lambda function.

Step 5: To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)



You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.

Name : Shivam.R.Prajapati

Div: D15C

Roll No: 41

Academic Year: 2024-25

Basic settings

Description - optional
Basic Settings of Lab12_Shivam

Memory Info
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage Info
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 Info
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.

Given some description for your settings

Ephemeral storage Info
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 Info
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 1 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console
 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

Click on Save.

Successfully updated the function Lab12_Shivam.

Lab12_Shivam

Function overview

Diagram Template

Lab12_Shivam

Layers (0)

+ Add trigger + Add destination

Throttle Copy ARN Actions

Description
Basic Settings of Lab12_Shivam

Last modified
in 7 seconds

Function ARN
arn:aws:lambda:us-east-1:380557944473:function:Lab12_Shivam

Step 6: Click on the "Test" tab, then select "Create a new event." Give the event a name, set "Event Sharing" to private, and choose the "S3 Put" template. S3 (Simple Storage Service) template allows you to test your Lambda function specifically for events related to uploading files to an S3 bucket.

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

Event41_Lab12

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

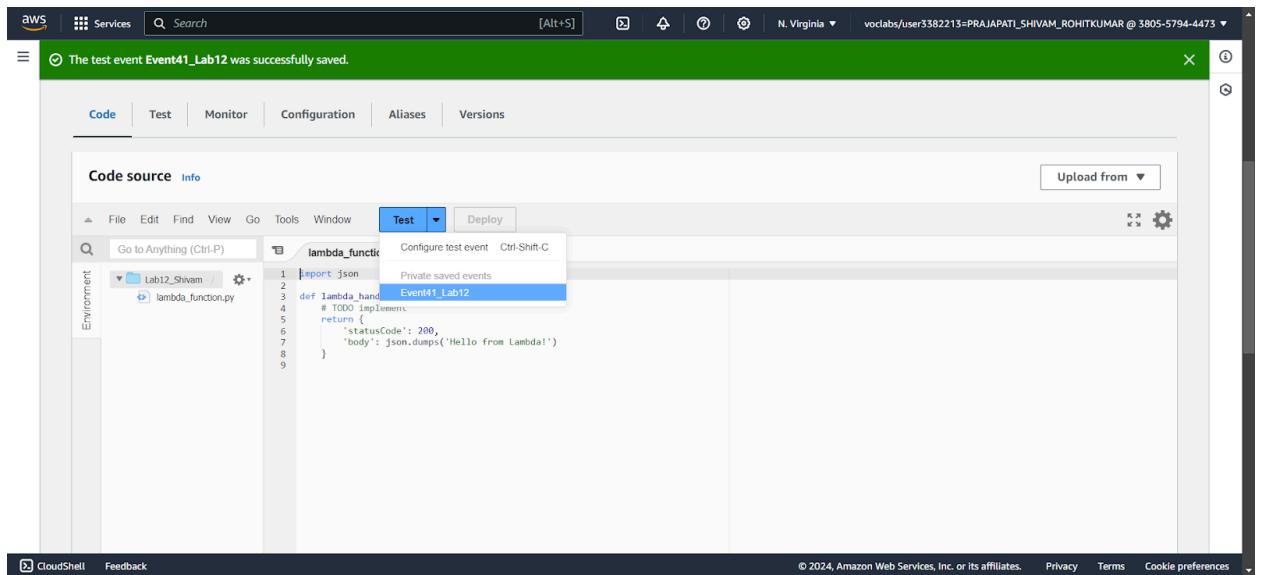
```

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 *       }
30 *     }
31 *   ]
32 *
33 * }

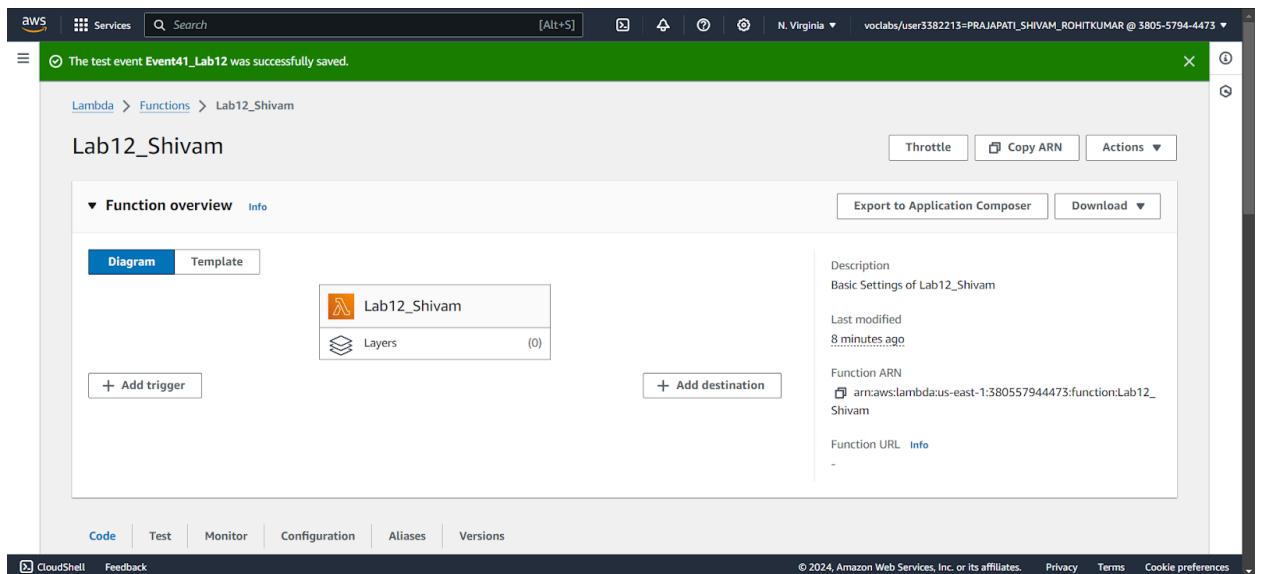
```

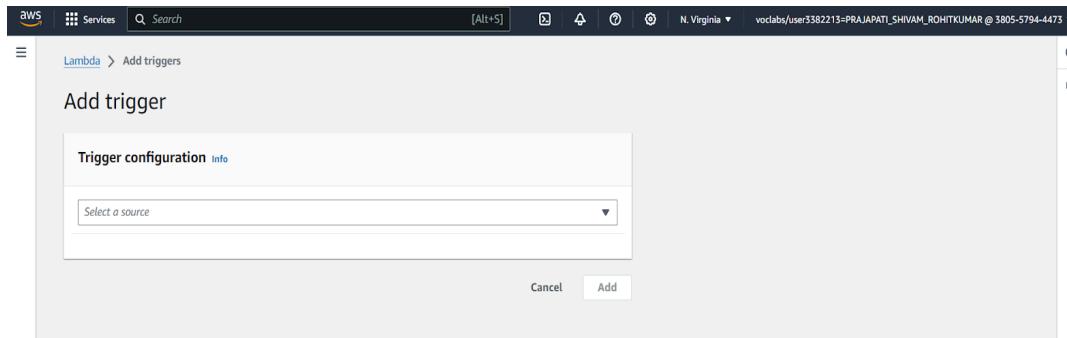
Event Jason code will be automatically generated once S3 -put is selected.

Step 7: Now In the Code section select the created event from the dropdown .

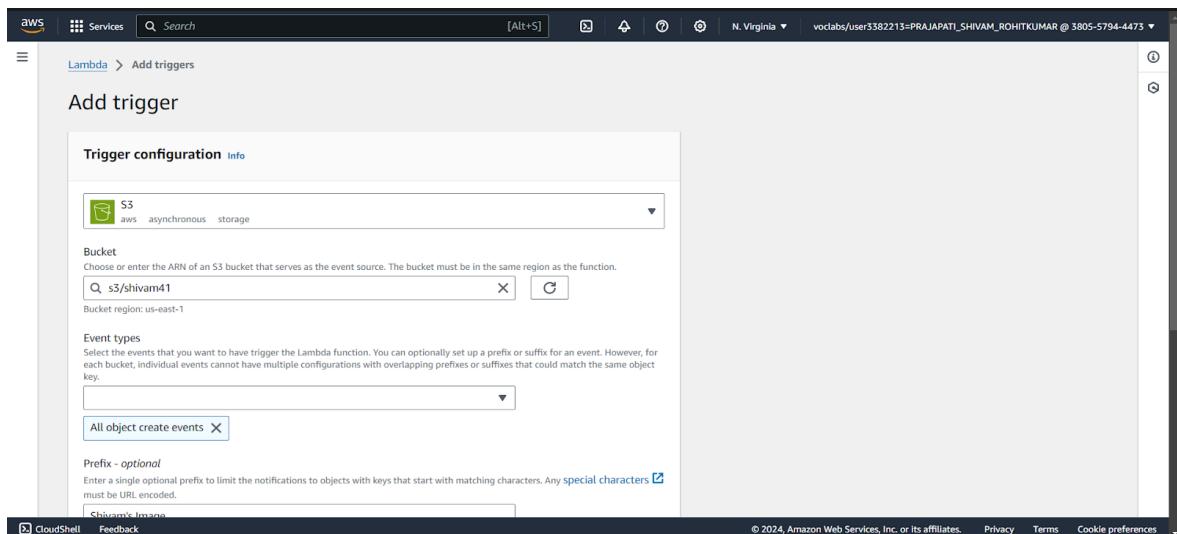
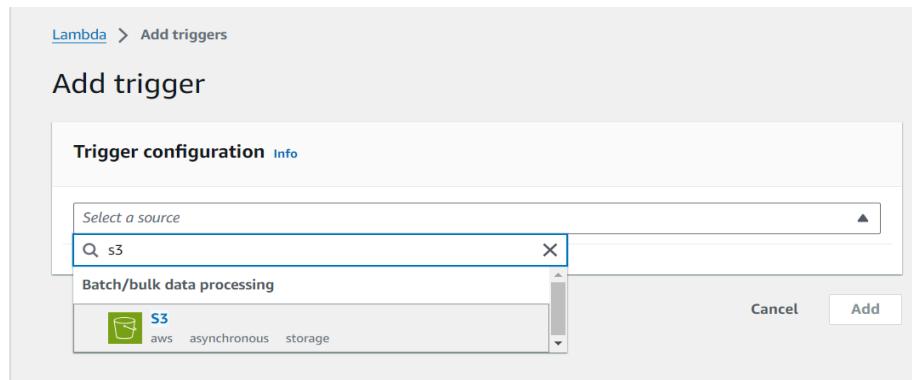


Step 8: In the Lambda function, click on "Add Trigger." Adding a trigger allows your Lambda function to automatically run in response to specific events such as uploads to an S3 bucket





Now select the source as S3, then choose the bucket name from the dropdown menu. Keep the other settings as default, and you can also add a prefix for the image if you want. A prefix for an image (or any file) in S3 is a string that you can use to organize or filter files within a bucket. It acts like a folder name, helping to categorize your files.

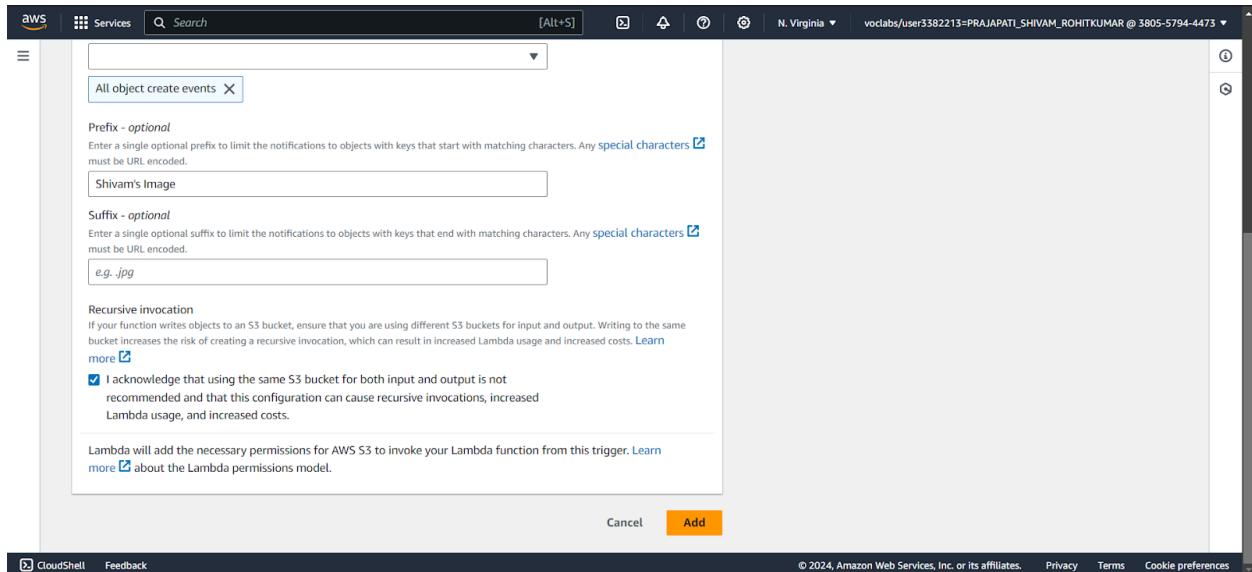


Name : Shivam.R.Prajapati

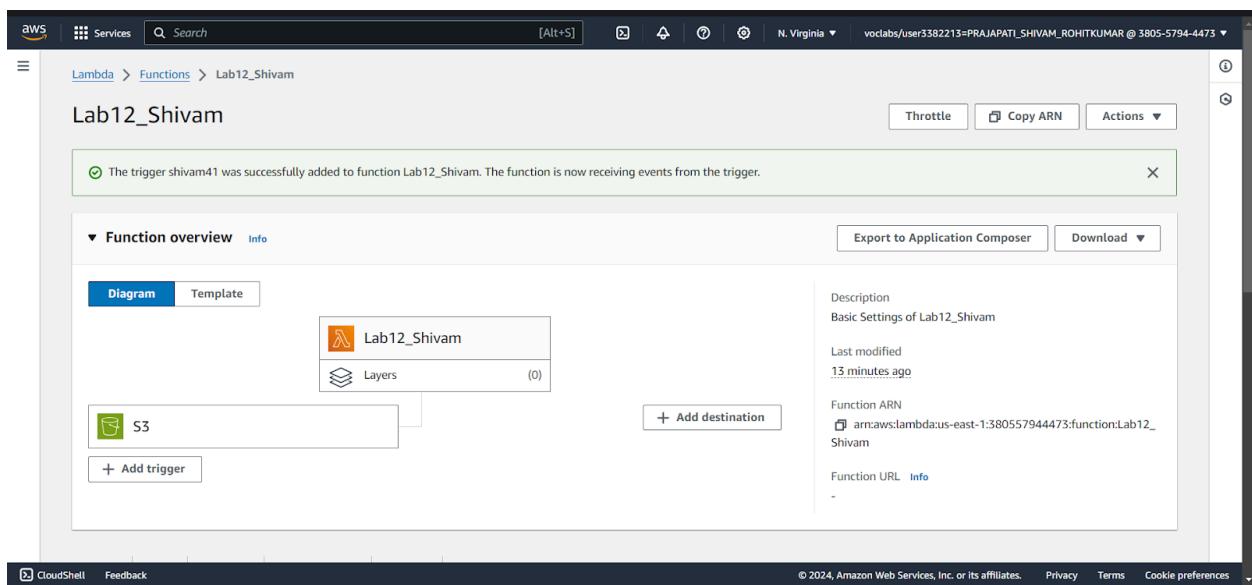
Div: D15C

Roll No: 41

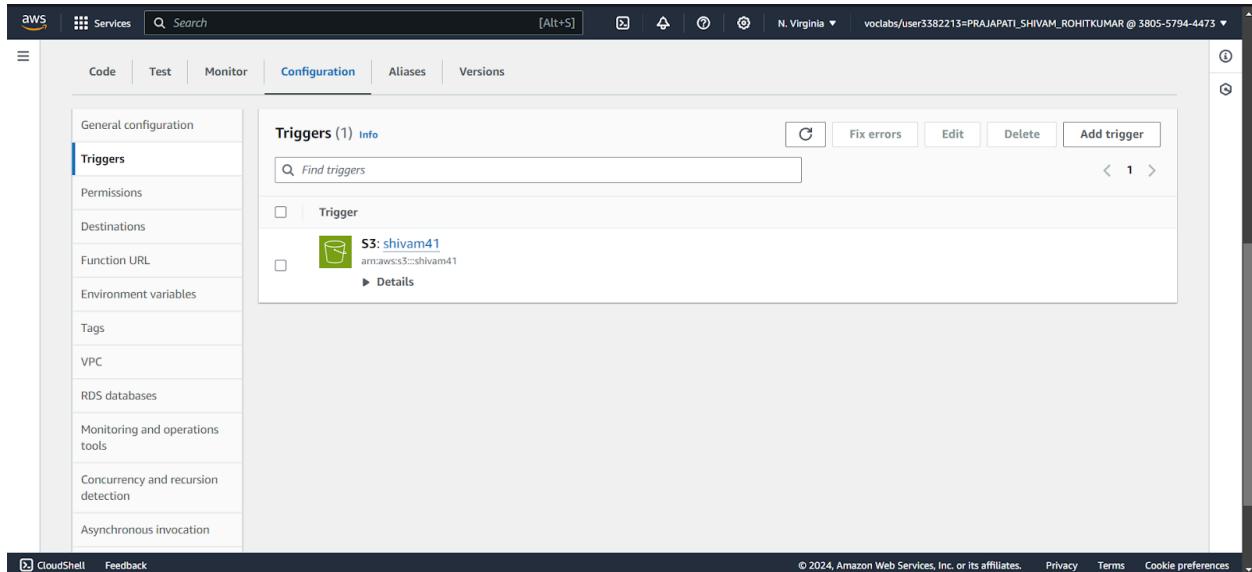
Academic Year: 2024-25



Click on save



Successfully triggered the event



Step 9: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

```
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']

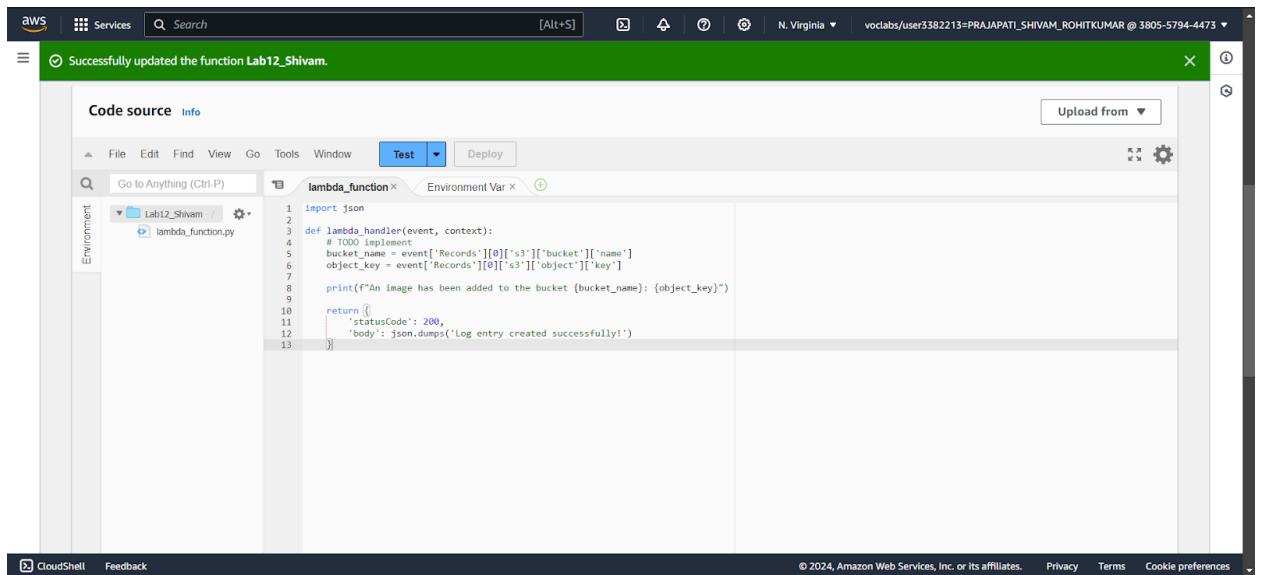
    print(f"An image has been added to the bucket {bucket_name}: {object_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```

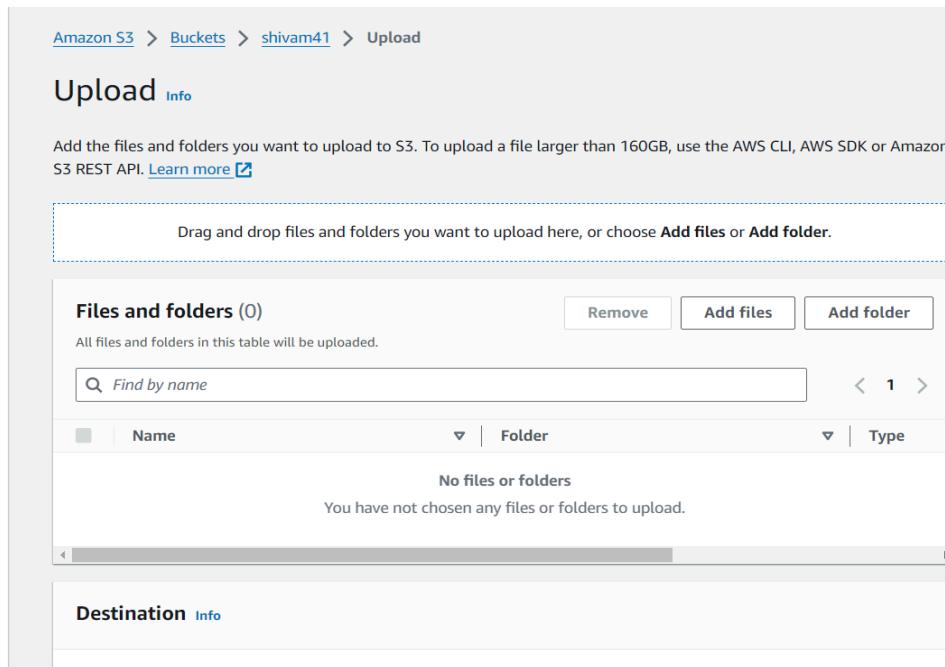
The screenshot shows the AWS Lambda function editor. The tab bar at the top has 'lambda_function' and 'Environment Var' tabs, with a '+' button to add new environment variables. The code editor contains the Python code for the lambda function, which is identical to the code provided in the previous step. The code uses the AWS Lambda event structure to extract the bucket name and object key from the S3 event, prints a log message, and returns a successful response with a log entry.

```
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
8     print(f"An image has been added to the bucket {bucket_name}: {object_key}")
9
10    return {
11        'statusCode': 200,
12        'body': json.dumps('Log entry created successfully!')
13    }
```

Save it by Control + S and deploy



Step 10: Now we will upload any image to the bucket



Click on add file where you can upload any image of your choice in your bucket

The screenshot shows the AWS S3 'Upload' interface. At the top, the navigation bar includes 'Amazon S3 > Buckets > shivam41 > Upload'. Below this, the 'Upload' section has a sub-header 'Upload [Info](#)'. A note states: '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](#)'.

In the main area, there's a large dashed box labeled 'Drag and drop files and folders you want to upload here, or choose Add files or Add folder.' Below this is a table titled 'Files and folders (1 Total, 255.5 KB)'. It contains one item: 'back_ground.png' (image/png). There are 'Remove', 'Add files', and 'Add folder' buttons above the table. A search bar 'Find by name' is also present.

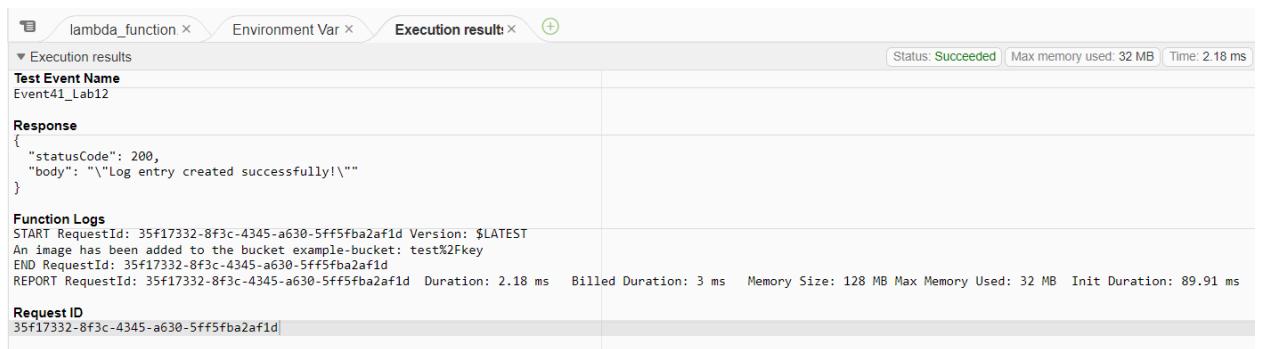
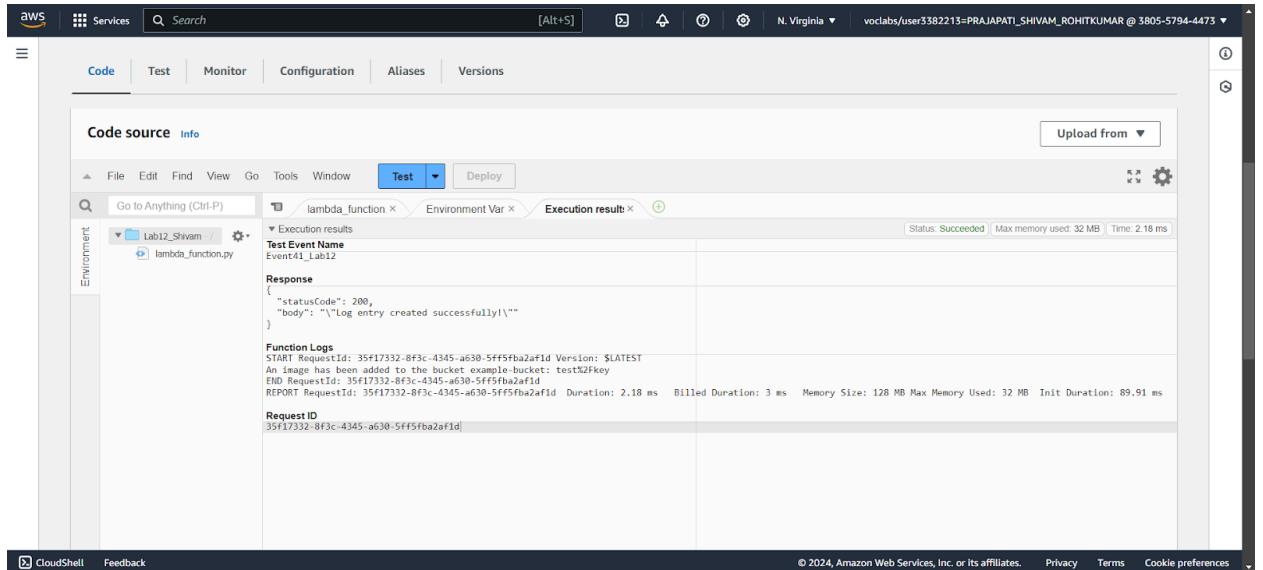
The 'Destination' section shows 'Destination s3://shivam41' with a link icon. At the bottom of the page, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

This screenshot shows the same AWS S3 upload interface as the previous one, but with additional expanded sections. The 'Destination' section now includes a 'Destination details' section with the note: 'Bucket settings that impact new objects stored in the specified destination.' Below this are sections for 'Permissions' (with the note: 'Grant public access and access to other AWS accounts.') and 'Properties' (with the note: 'Specify storage class, encryption settings, tags, and more.').

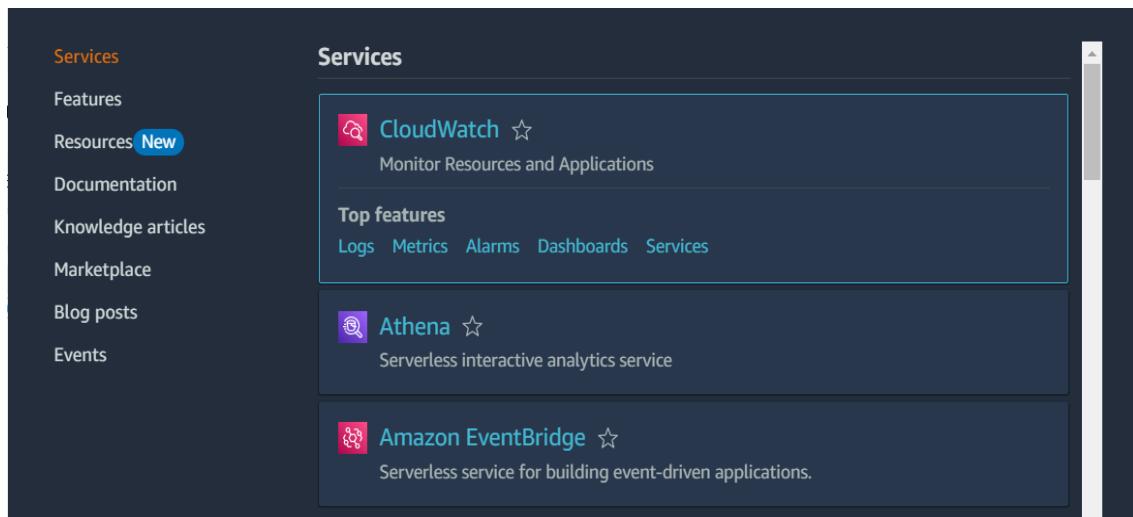
At the bottom right of the main content area, there are 'Cancel' and 'Upload' buttons. The 'Upload' button is highlighted with a yellow background. The footer links at the bottom of the page remain the same as in the first screenshot.

Click on Upload

Step 11: Now click on "Test" in Lambda to see if it logs the activity when an image is added to S3.



Step 12: Now let's check the logs on CloudWatch. Go to the "Monitor" section and click on "View CloudWatch Logs".



Click on the CloudWatch:

The screenshot shows the AWS CloudWatch Overview page. The left sidebar has 'Logs' selected under 'CloudWatch'. The main area features a 'Get started with CloudWatch' section with four cards: 'Create alarms' (Set alarms on any of your metrics to receive notification when your metric crosses your specified threshold), 'Create a default dashboard' (Create and name any CloudWatch dashboard CloudWatch-Default to display it here), 'View logs' (Monitor using your existing system, application and custom log files), and 'View events' (Write rules to indicate which events are of interest to your application and what automated action to take). Below this is a 'Get started with Application Insights' section with a 'Configure Application Insights' button.

Click on the logs:

The screenshot shows the AWS CloudWatch Log groups page. The left sidebar has 'Logs' selected under 'CloudWatch'. The main area shows details for a log group named '/aws/lambda/Lab12_Shivam'. The 'Log group details' table includes columns for Log class (Info, Standard), ARN (arn:aws:logs:us-east-1:380557944473:log-group:/aws/lambda/Lab12_Shivam:*), Metric filters (0), Subscription filters (0), Creation time (2 minutes ago), Retention (Never expire), KMS key ID (-), Anomaly detection (Configure), Data protection (-), and Contributor Insights rules (-). Below the table are tabs for Log streams, Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, and Data protection. A 'Log streams (1)' section at the bottom contains a 'Create log stream' button.

Click on the log group:

CloudWatch Log group details

Log class: Info Standard

ARN: arn:aws:logs:us-east-1:380557944473:log-group:/aws/lambda/Lab12_Shivam:*

Creation time: 3 minutes ago

Retention: Never expire

Metric filters: 0

Subscription filters: 0

KMS key ID: -

Anomaly detection: Configure

Data protection: -

Sensitive data count: -

Log streams

Log streams (1)

Filter log streams or try prefix search

Last event time: 2024-09-30 [\\$LATEST]c5ed8859ae7e475da99c216683024613 2024-09-30 09:51:15 (UTC)

Scrolled down and click on the log stream:

CloudWatch > Log groups > /aws/lambda/Lab12_Shivam > 2024/09/30/[\\$LATEST]c5ed8859ae7e475da99c216683024613

Log events

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
2024-09-30T09:51:15.367Z	INIT_START Runtime Version: python:3.12.v30 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:acd06500d0e3f6a085fb07933e...
2024-09-30T09:51:15.460Z	START RequestId: 35f17332-8f3c-4345-a630-5ff5fba2af1d Version: \$LATEST
2024-09-30T09:51:15.460Z	An image has been added to the bucket example-bucket: test%2Fkey
2024-09-30T09:51:15.471Z	END RequestId: 35f17332-8f3c-4345-a630-5ff5fba2af1d
2024-09-30T09:51:15.471Z	REPORT RequestId: 35f17332-8f3c-4345-a630-5ff5fba2af1d Duration: 2.18 ms Billed Duration: 3 ms Memory Size: 128 MB Max Mem...

CONCLUSION:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It's important to choose the S3 Put template for the event; otherwise, the code will give an error. The function was triggered correctly when files were uploaded to S3, showing that Lambda's event-driven design works well. This experiment showed how Lambda can respond to S3 events and how to fix common problems with the event setup.