

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

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

- First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

- Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.
- Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:

- Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

- Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

5. Test the Setup:

- Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Prerequisites: AWS Personal Account

Name: Bhushan Mukund Kor

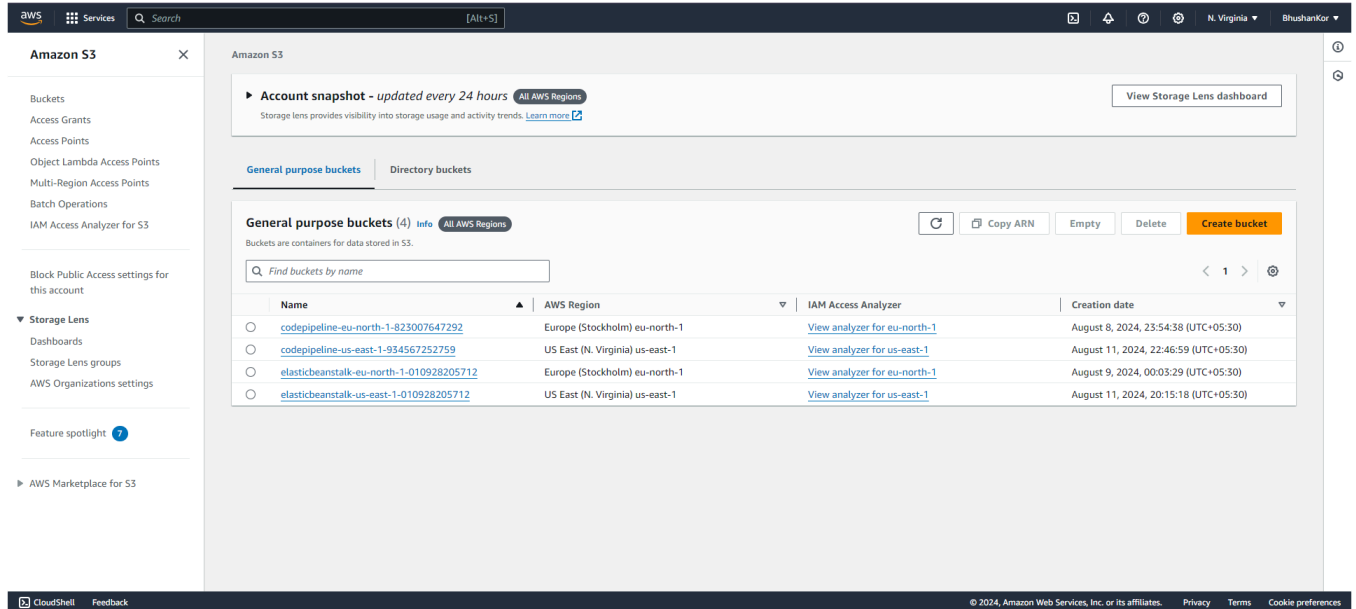
Academic Year: 2024-2025

Division: D15C

Roll No: 28

Steps To create the lambda function:

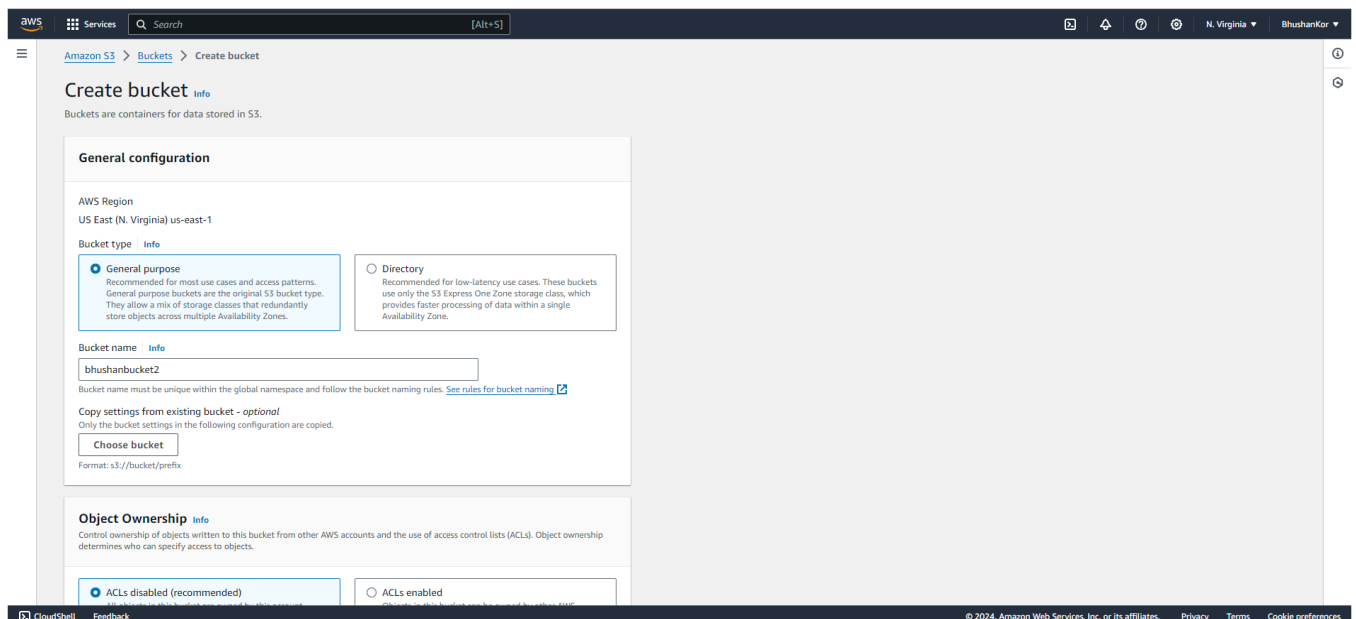
Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



The screenshot shows the Amazon S3 console interface. On the left, there's a navigation pane with 'Amazon S3' selected. The main area displays 'General purpose buckets (4)' with a table listing existing buckets. The table has columns for Name, AWS Region, IAM Access Analyzer, and Creation date. The buckets listed are:

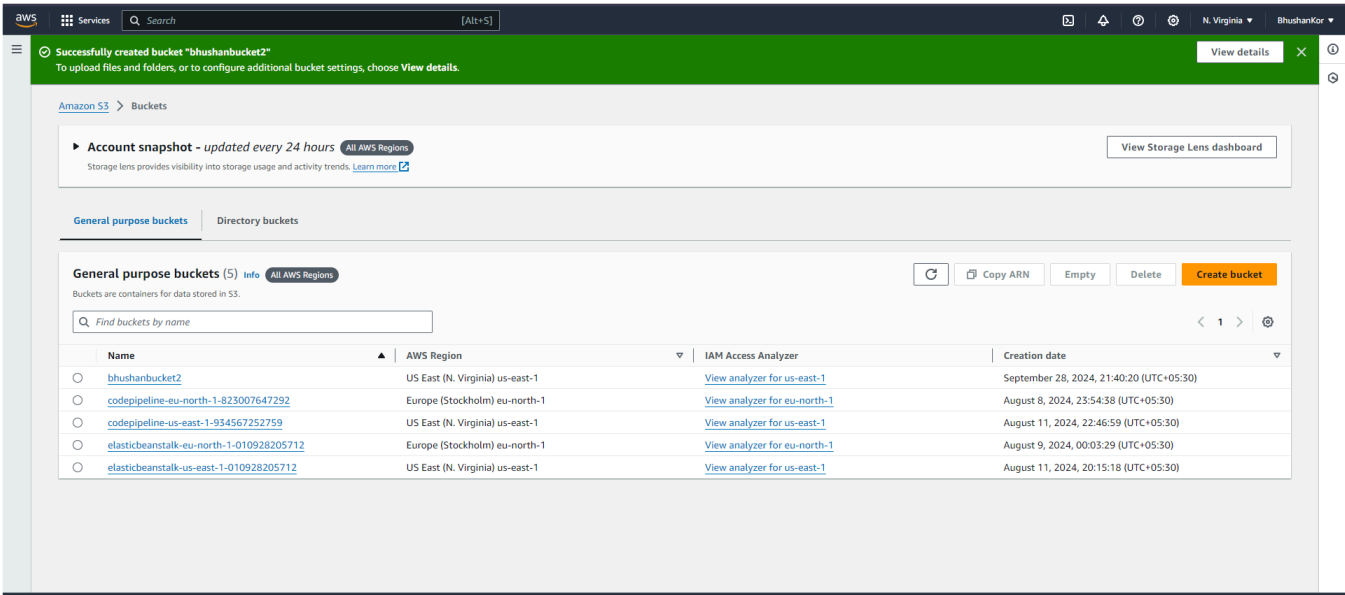
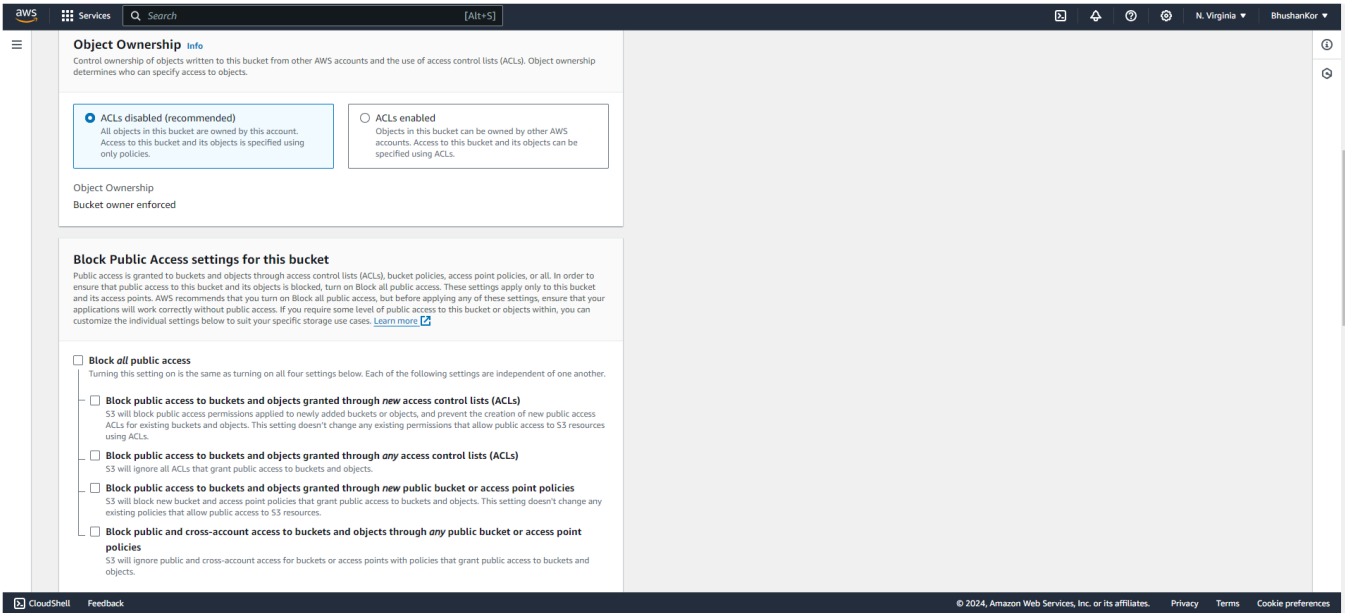
Name	AWS Region	IAM Access Analyzer	Creation date
codepipeline-eu-north-1-823007647292	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 8, 2024, 23:54:38 (UTC+05:30)
codepipeline-us-east-1-934567252759	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 11, 2024, 22:46:59 (UTC+05:30)
elasticbeanstalk-eu-north-1-010928205712	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 9, 2024, 00:03:29 (UTC+05:30)
elasticbeanstalk-us-east-1-010928205712	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 11, 2024, 20:15:18 (UTC+05:30)

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



The screenshot shows the 'Create bucket' wizard in the Amazon S3 console. The 'General configuration' section is active, showing the following settings:

- AWS Region:** US East (N. Virginia) us-east-1
- Bucket type:** General purpose (selected), Directory (deselected)
- Bucket name:** bhushanbucket2
- Copy settings from existing bucket - optional:** Choose bucket (button)
- Object Ownership:** ACLs disabled (recommended) (selected), ACLs enabled (deselected)



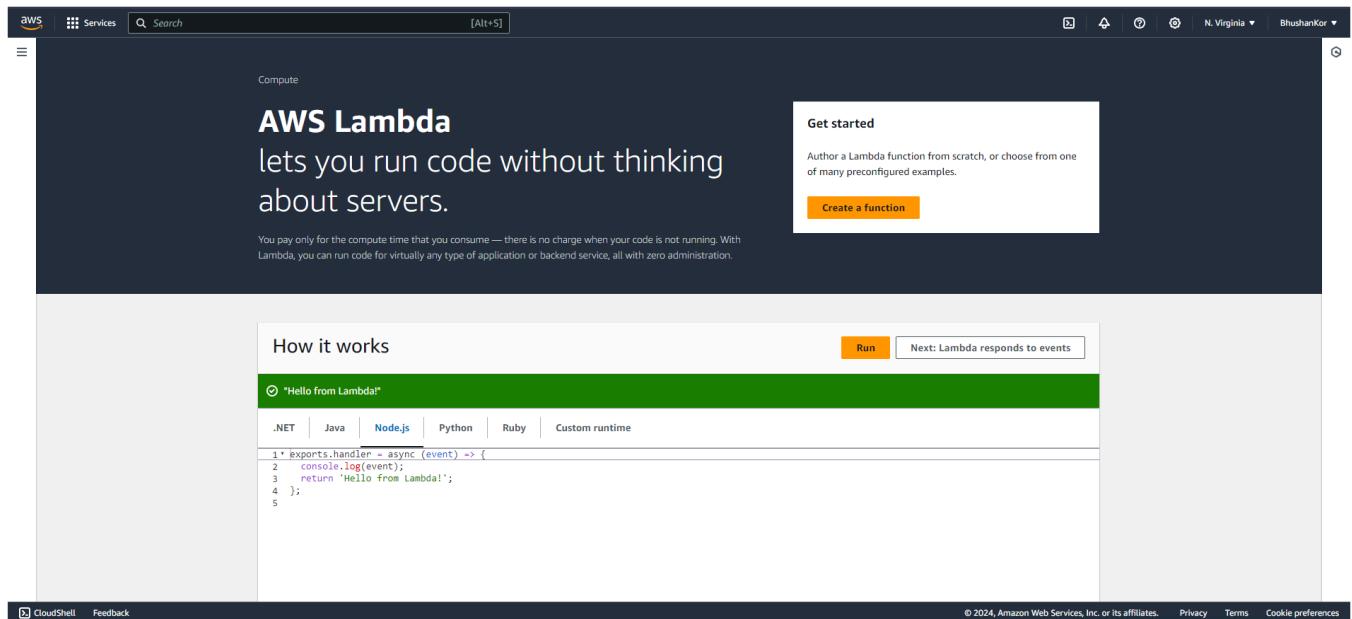
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

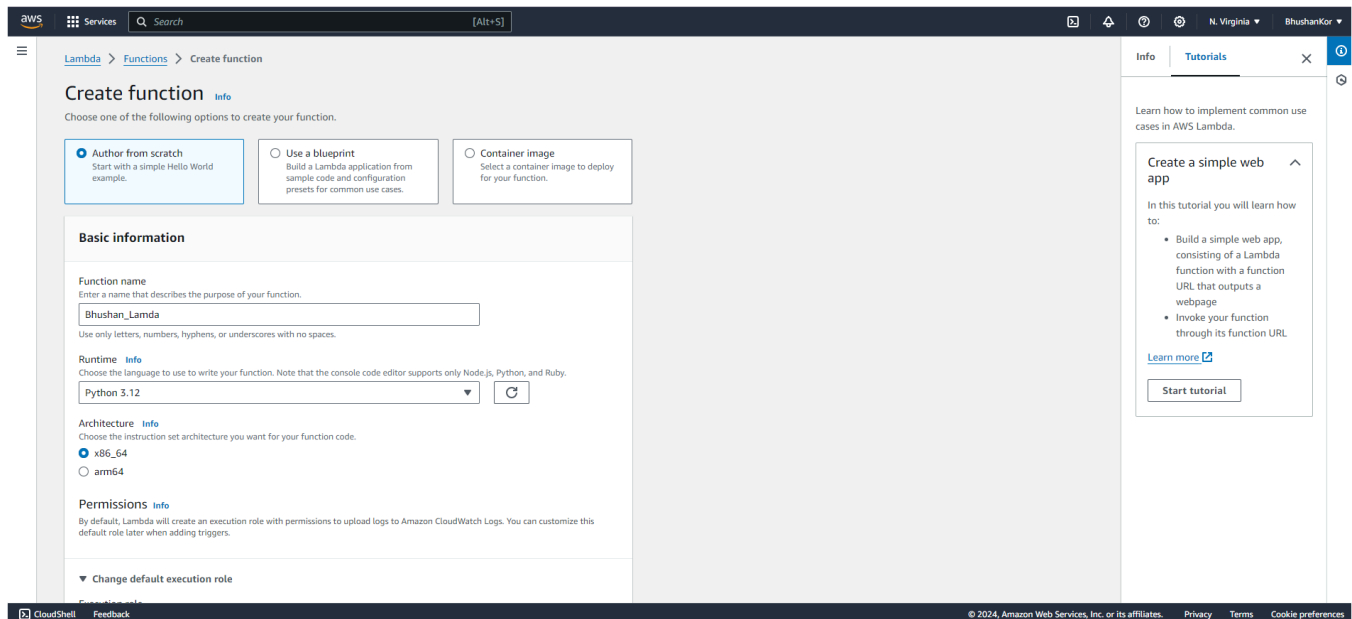
Division: D15C

Roll No: 28

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

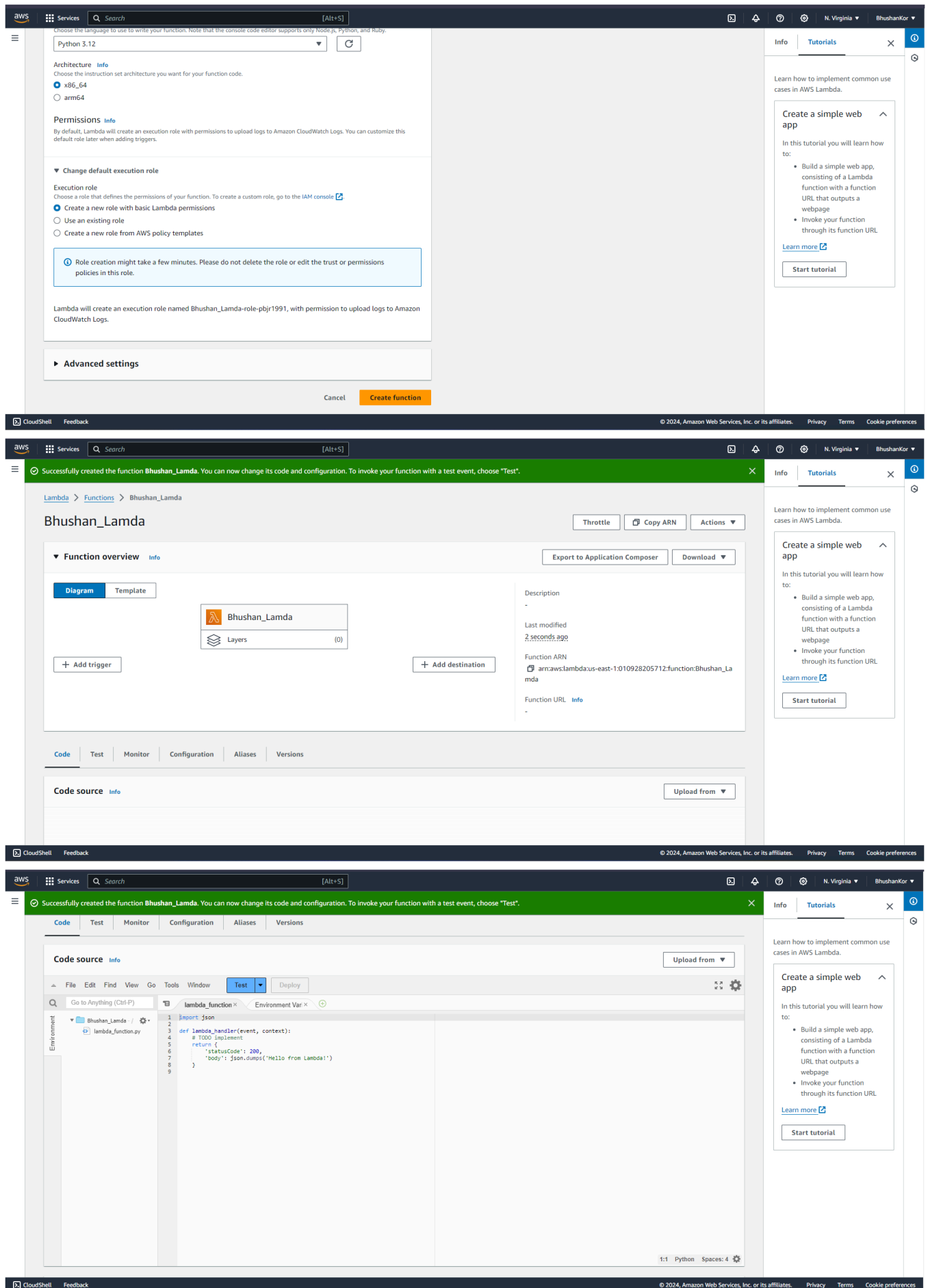


Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.



Academic Year:2024-2025

Roll No: 28



Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

So See or Edit the basic settings go to configuration then click on edit general setting.

The screenshot shows the 'Configuration' tab in the AWS Lambda console. On the left, a sidebar lists various configuration options: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, and RDS databases. The 'General configuration' option is selected and highlighted. The main area displays the 'General configuration' for the function. It includes an 'Edit' button in the top right corner. The configuration details are as follows:

Property	Value
Description	-
Memory	128 MB
Ephemeral storage	512 MB
Timeout	0 min 3 sec
SnapStart	None

Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

The screenshot shows the 'Edit basic settings' page in the AWS Lambda console. The page title is 'Edit basic settings'. The left sidebar shows the 'Basic settings' section. The main area displays the configuration details for the function 'Bhushan_Lambda'. The configuration details are as follows:

Property	Value
Description - optional	Basic Settings
Memory	128 MB
Ephemeral storage	512 MB
SnapStart	None
Timeout	0 min 1 sec

Below the configuration details, there is an 'Execution role' section. It includes a radio button to 'Use an existing role' and a radio button to 'Create a new role from AWS policy templates'. The 'Use an existing role' option is selected.

Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

The screenshot shows the AWS Lambda console's 'Test' tab for a function. At the top, a green banner indicates 'Executing function: succeeded'. Below this, the 'Test event' section has a 'Save' button and a 'Test' button. The 'Test event action' is set to 'Create new event'. The 'Event name' is 'Bhushan_Bucket'. Under 'Event sharing settings', 'Private' is selected. The 'Template - optional' dropdown is set to 's3-put'. At the bottom, there is an 'Event JSON' section with a 'Format JSON' button.

Test event Info

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

Bhushan_Bucket

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

Format JSON

This screenshot shows the 'Event JSON' section of the AWS Lambda console. The JSON is formatted and shows details for an s3-put event, including event version, source, region, time, name, identity, request parameters, response elements, and bucket information.

Template - optional

s3-put

Event JSON

Format JSON

```
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",  
31 *       "size": 1024,  
32 *     }  
33 *   }  
34 * }  
35 * ]
```

1,1 JSON Spaces: 2

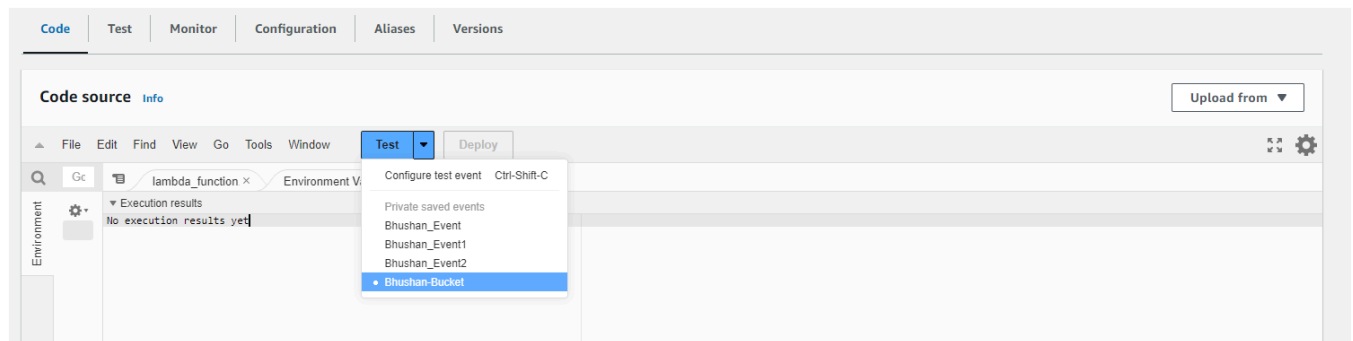
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

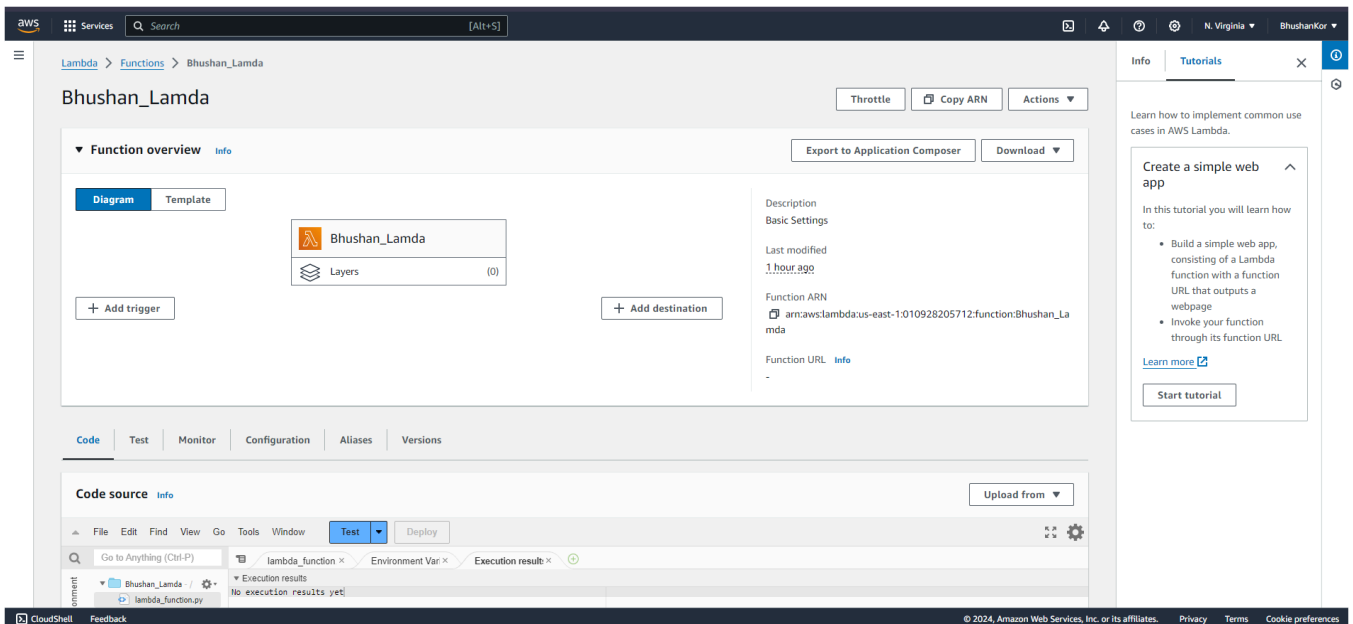
Division: D15C

Roll No: 28

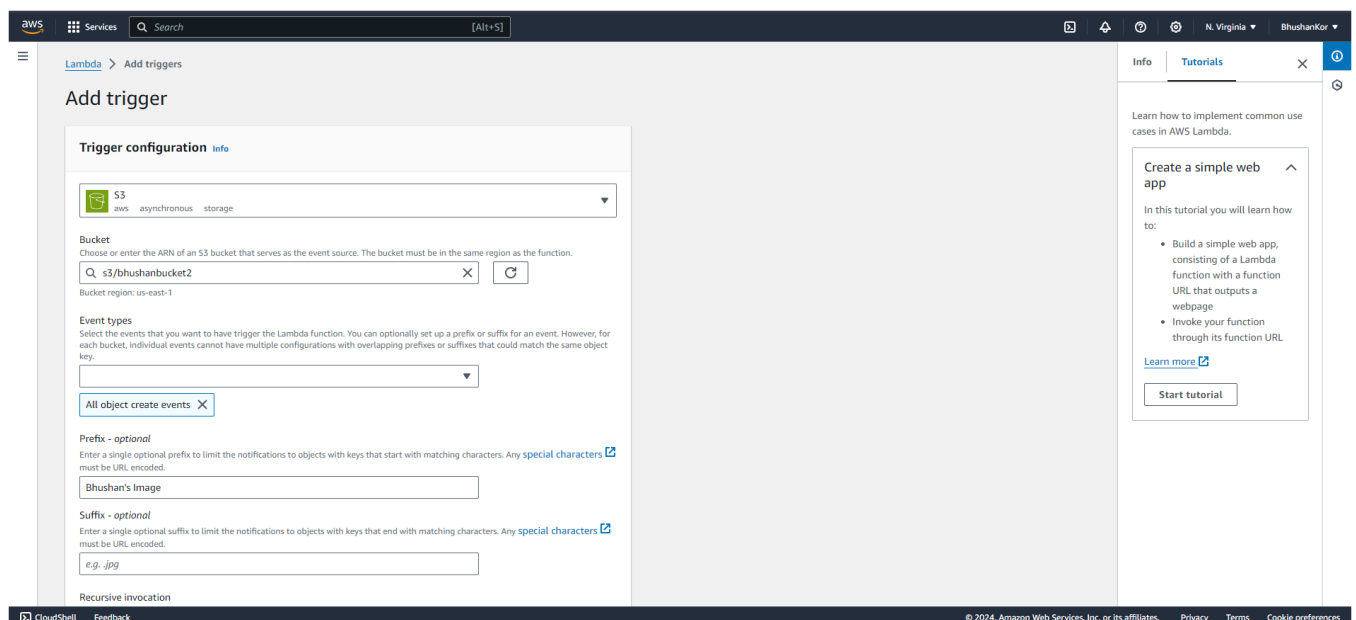
Step 6: Now In Code section select the created event from the dropdown .



Step 7: Now In the Lambda function click on add trigger.



Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.



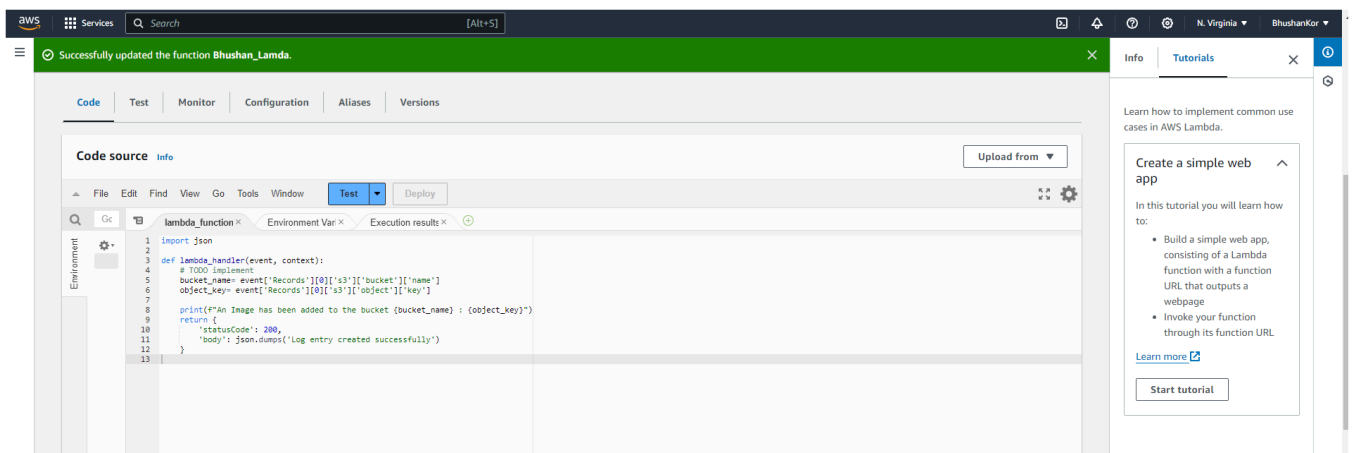
The first screenshot shows the 'Bhushan_Lambda' function overview in the AWS Lambda console. A green notification bar at the top states: 'The trigger bhushanbucket2 was successfully added to function Bhushan_Lambda. The function is now receiving events from the trigger.' The 'Function overview' section includes a diagram showing the function connected to an S3 bucket. The 'Triggers' tab is selected, showing a list of triggers with one entry: 'S3: bhushanbucket2'. The second screenshot shows the 'Configuration' tab for the same function. The 'Triggers' section is expanded, showing the details of the 'S3: bhushanbucket2' trigger, including its ARN and details link. The left sidebar shows various configuration options like General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, RDS databases, Monitoring and operations tools, Concurrency and recursion detection, Asynchronous invocation, Code signing, File systems, and State machines.

Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

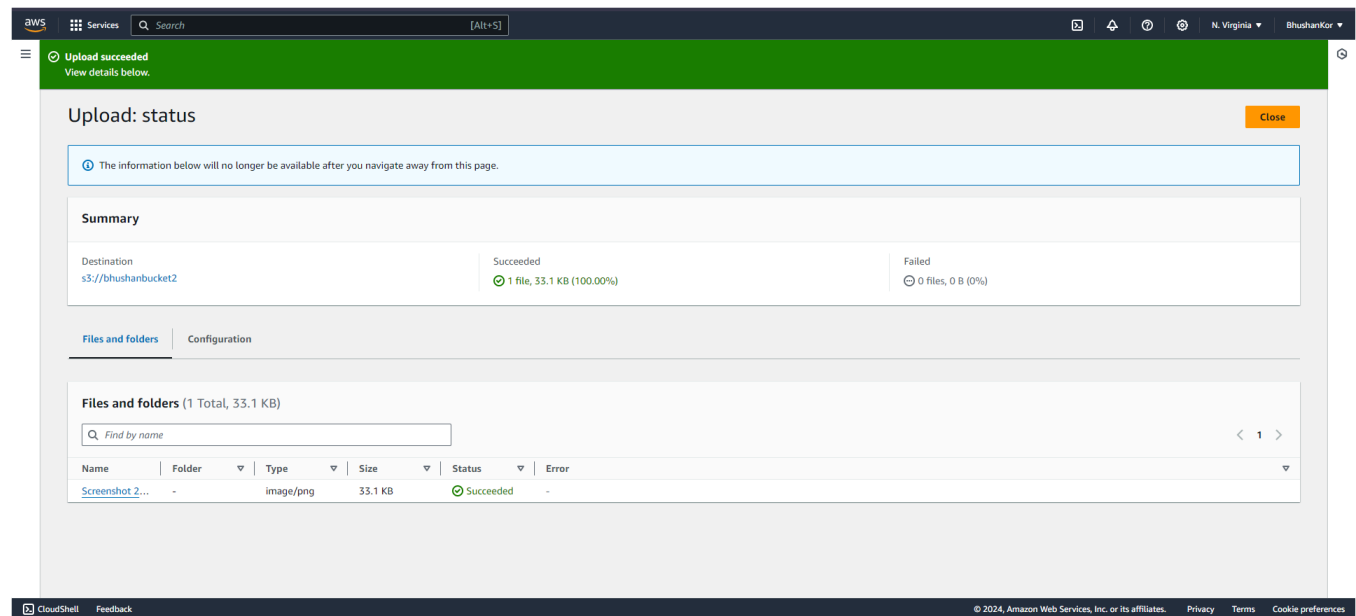
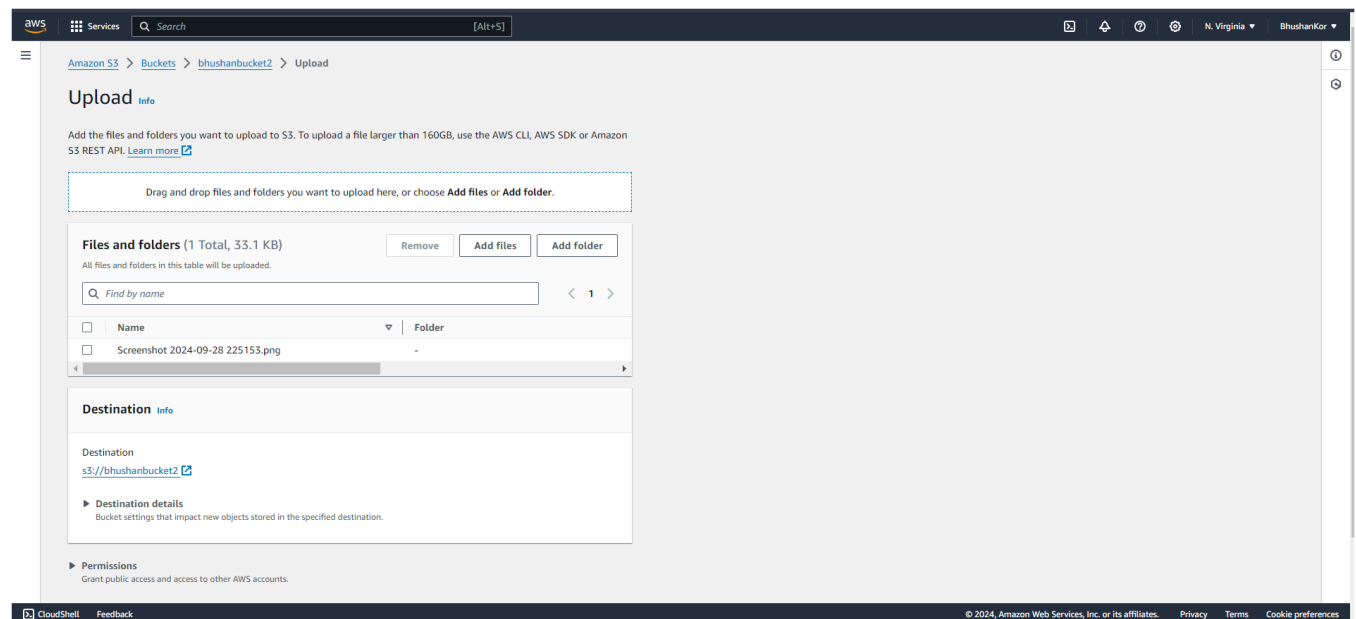
The screenshot shows the 'Code source' tab in the AWS Lambda console for the 'lambda_function'. The code is written in Python and is as follows:

```
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     return {
10         'statusCode': 200,
11         'body': json.dumps('Log entry created successfully')
12     }
13
```

The interface includes a 'Deploy' button and a status message 'Changes not deployed'. The left sidebar shows the 'Code' tab selected, with options for File, Edit, Find, View, Go, Tools, Window, and Environment.



Step 9: Now upload any image to the bucket.



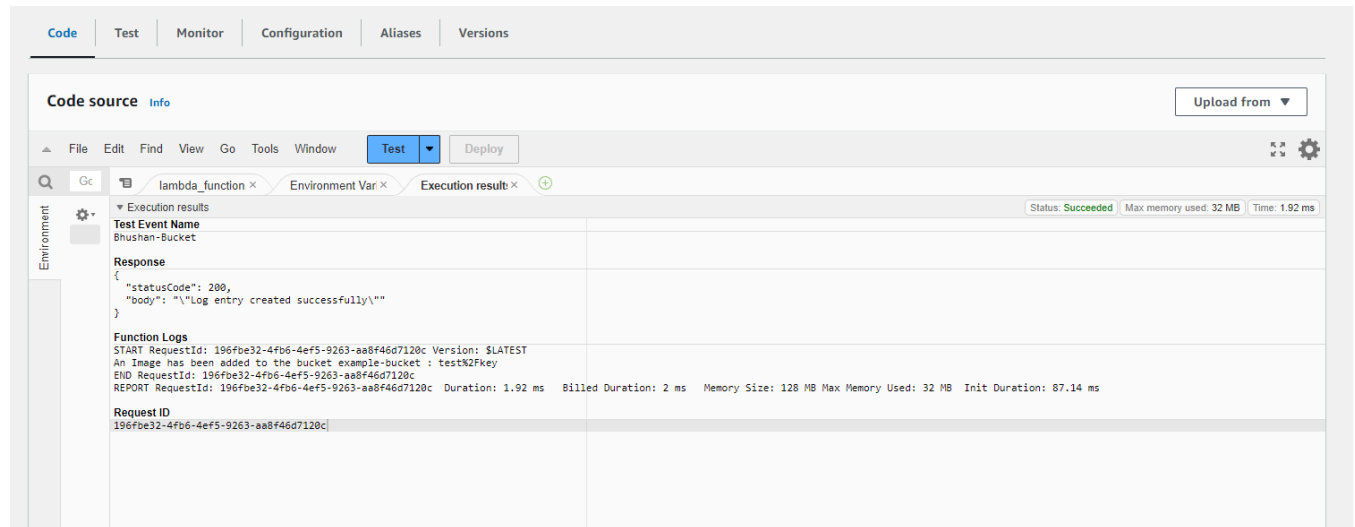
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

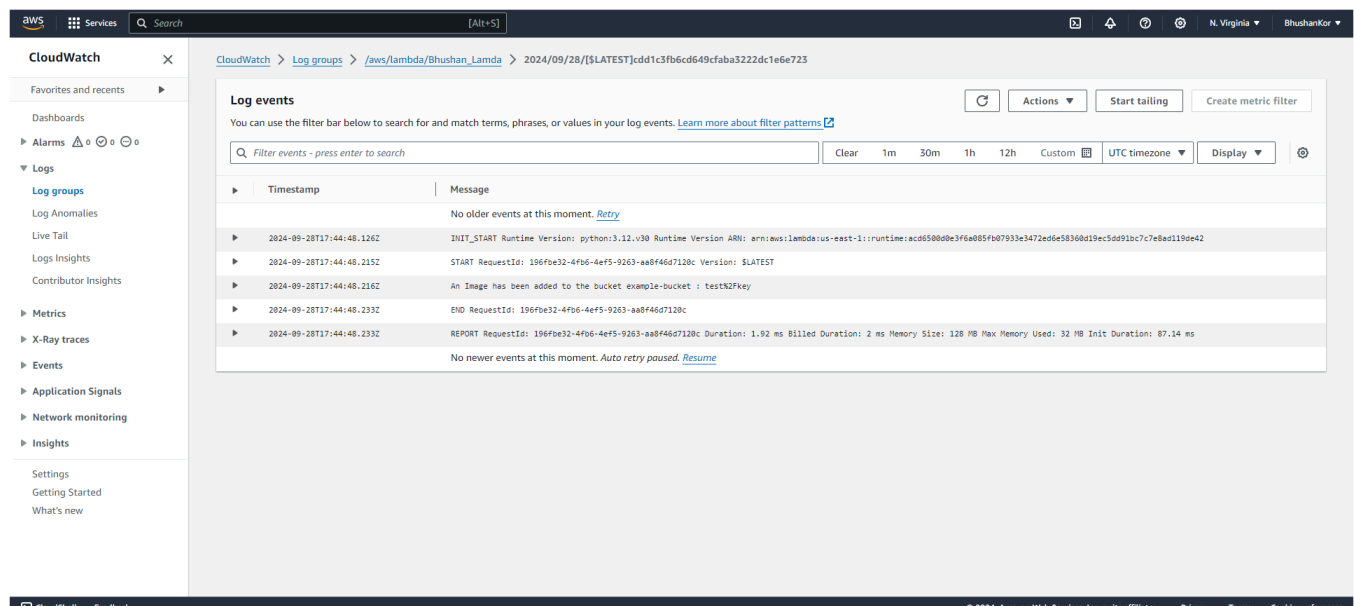
Division: D15C

Roll No: 28

Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.



Step 11: Now Lets see the log on Cloud watch. To see it go to monitor section and then click on view cloudwatch logs.



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 is important to note that we have to select S3-put template in event other wise code will give an error.** The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.