Name:Sahil Ramrakhyani Div:D15C Roll No:42

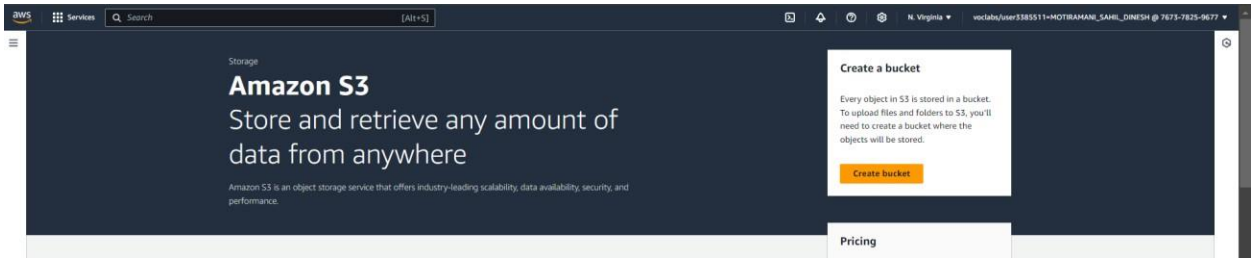
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Prerequisites:

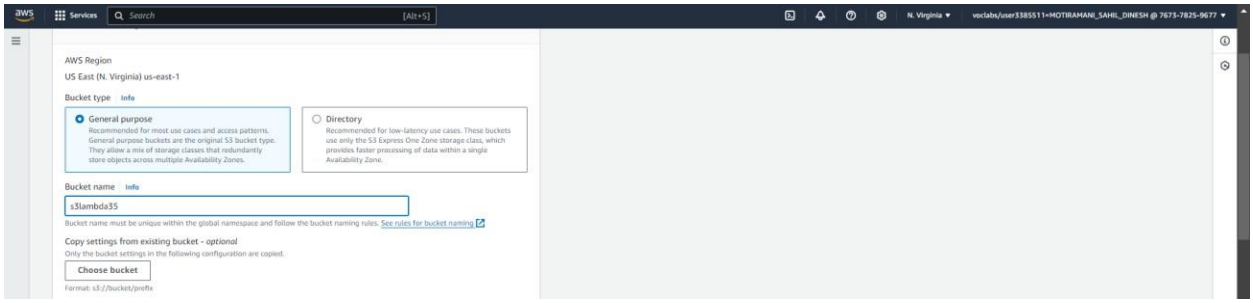
1) AWS account (acacdemy preferable)
2) Lambda function (created in the previous experiment).

Step 1: Create a s3 bucket.

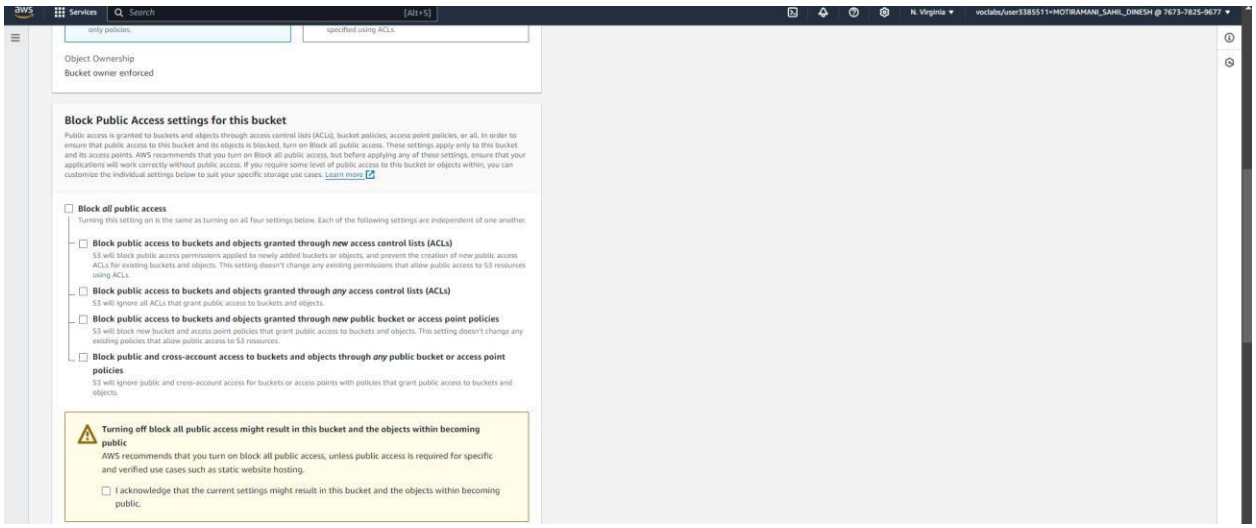
1) Search for S3 bucket in the services search. Then click on create bucket.



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

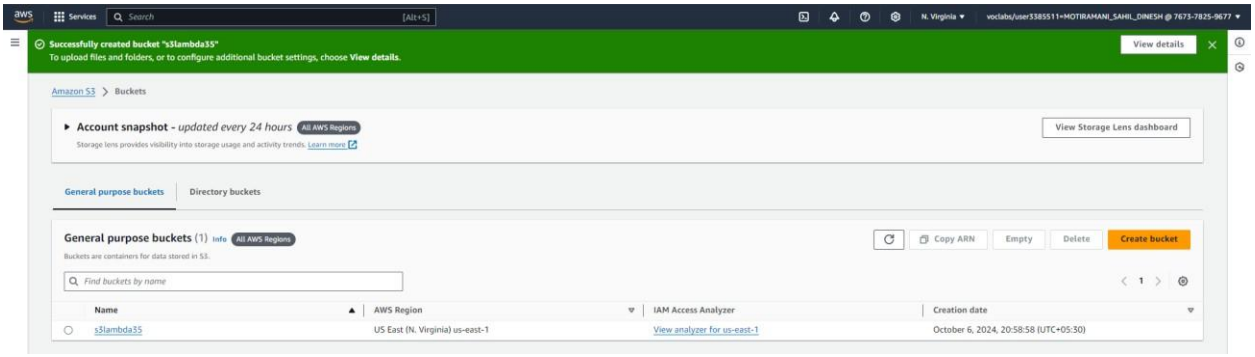


3) Uncheck block all public access.

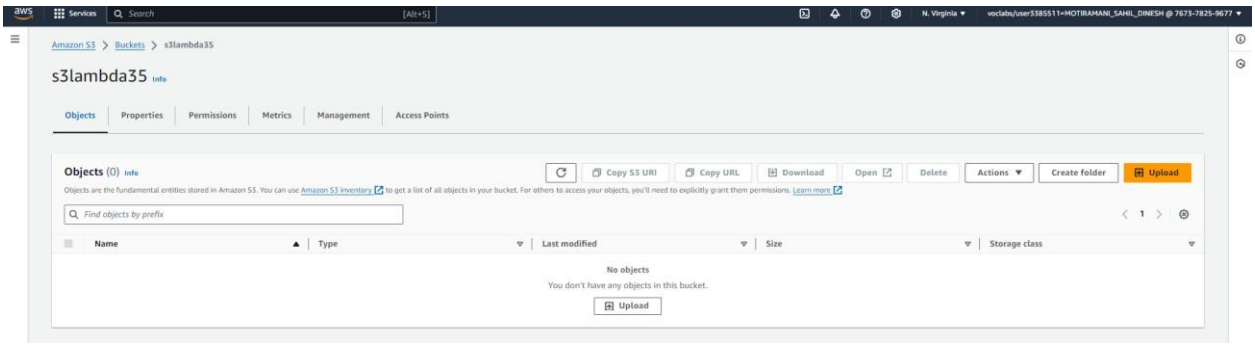


Name:Sahil Ramrakhyani Div:D15C Roll No:42

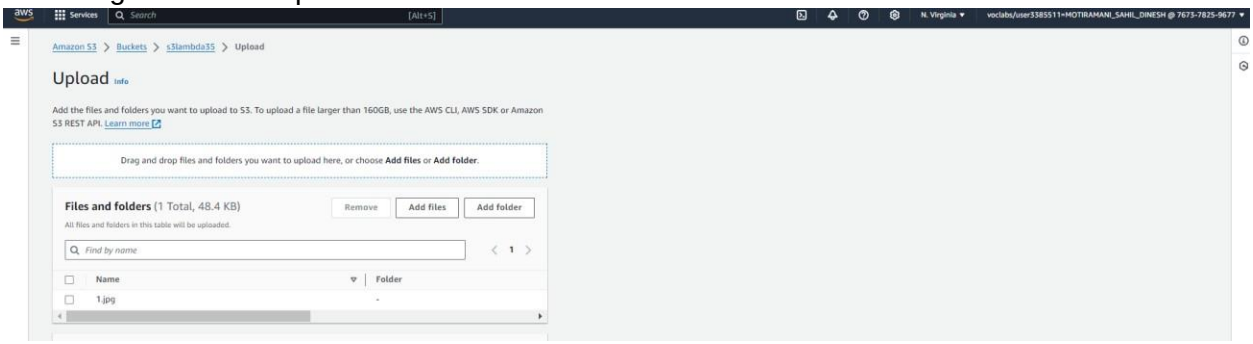
4) Keeping all other options same, click on create. This would create your bucket. Now click on the name of the bucket.



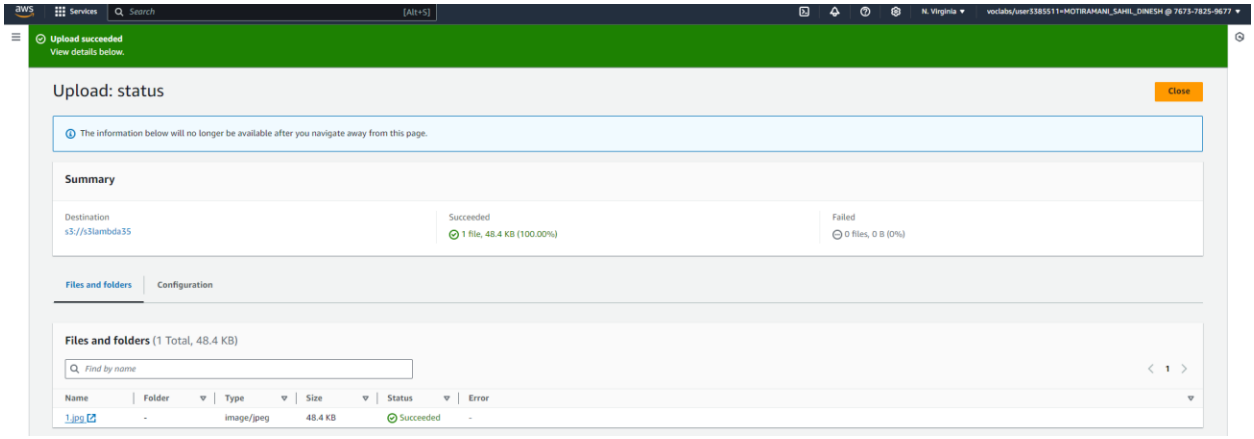
5) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload.



6) The image has been uploaded to the bucket.

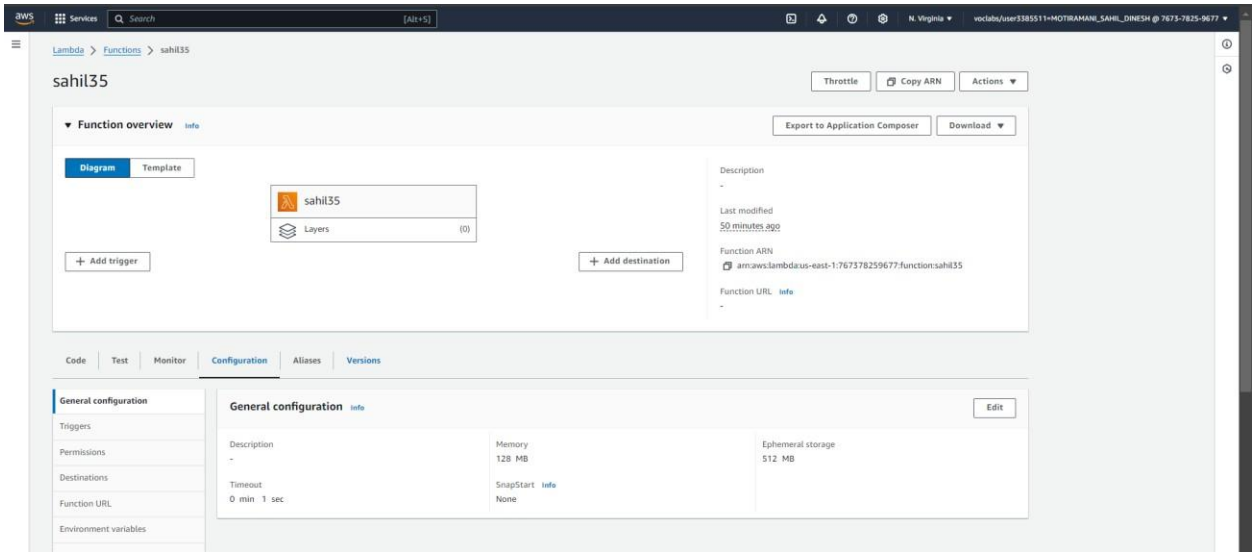


Name:Sahil Ramrakhyani Div:D15C Roll No:42

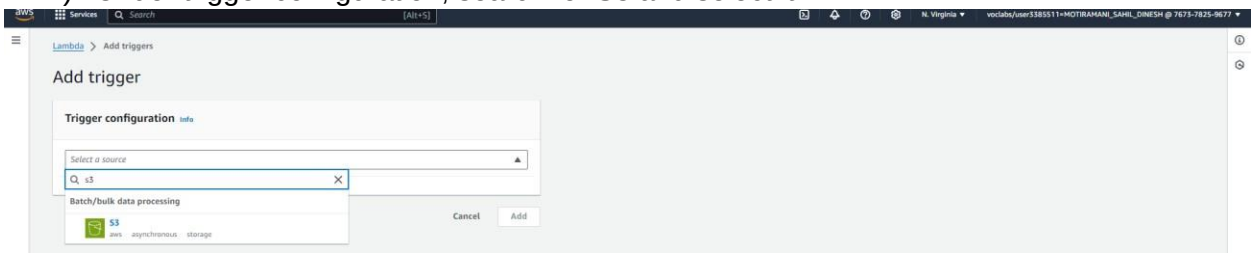


Step 2: Configure Lambda function

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

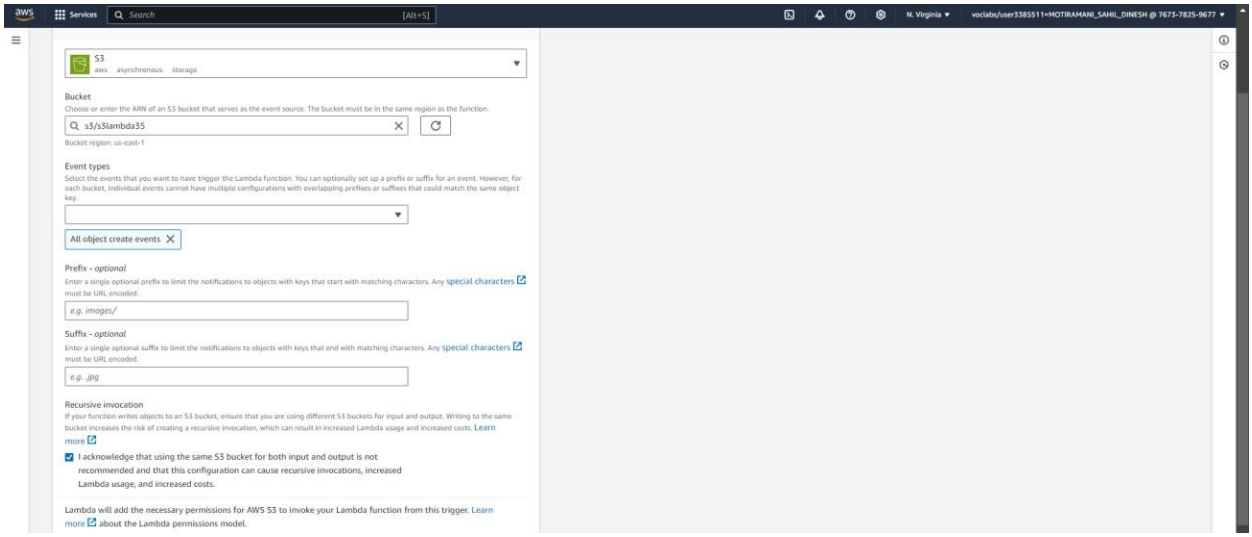


2) Under trigger configuration, search for S3 and select it.



Name:Sahil Ramrakhyani Div :D15C Roll no:42

Here, select teh 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.

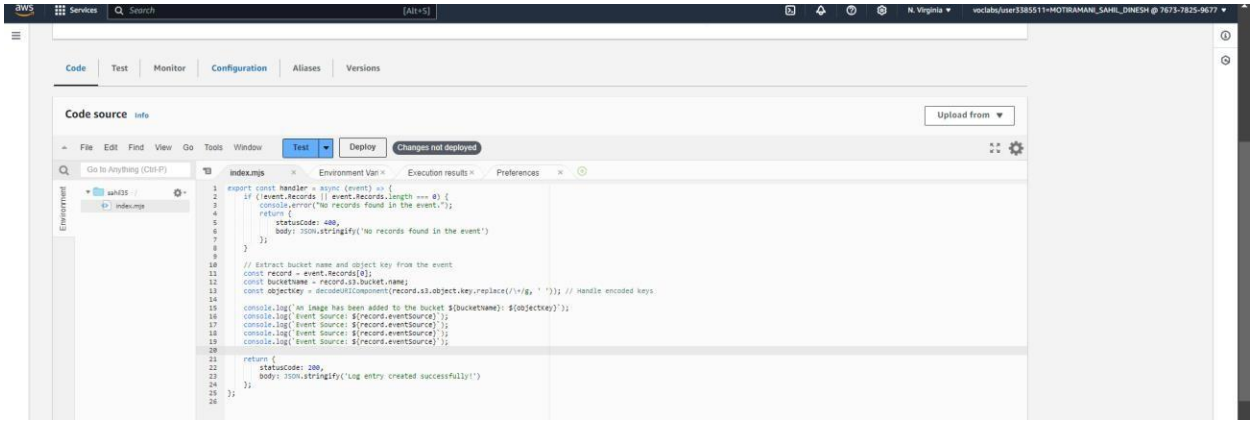


Scroll down to the code section of the function. Add the following javascript code to the code area by replacingthe existing code.

```
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 code checks for records in the event, extracts the bucket name and object key, logs the details, and returns a success message if an image is added to the bucket.



Now, click on the dropdown near test, then click on configure test event.

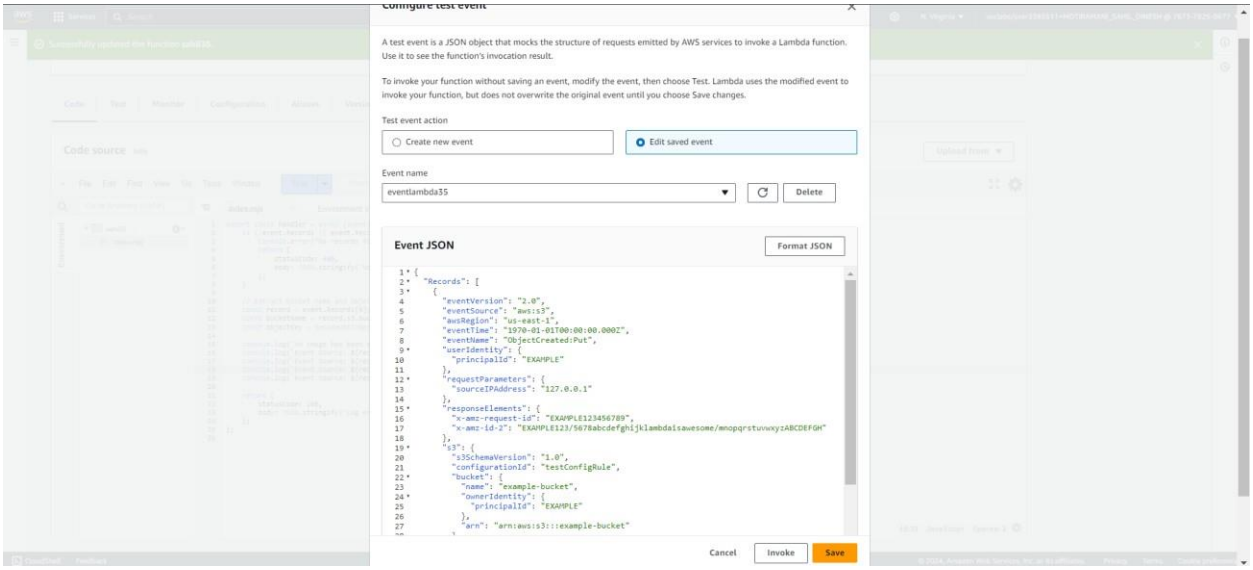
6) Here, select edit saved event. Select the event taht you had created before. Under Event JSON, paste the following code.

```
{
  "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"
        }
      }
    }
  ]
}
```


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

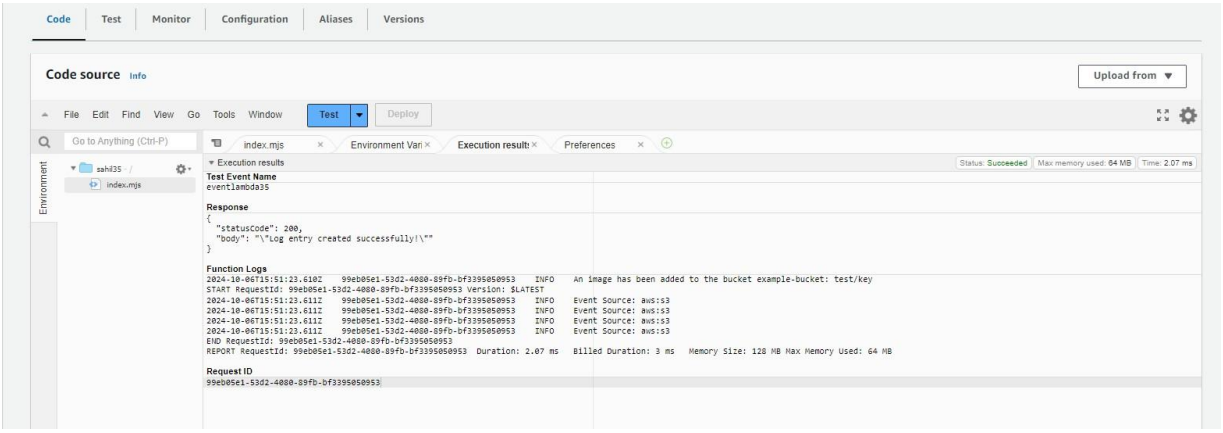
event time.



Save the changes. Then deploy the code changes by clicking on deploy.

7) After deploying, click on Test. The console output shows that ‘an image has been added to the bucket’

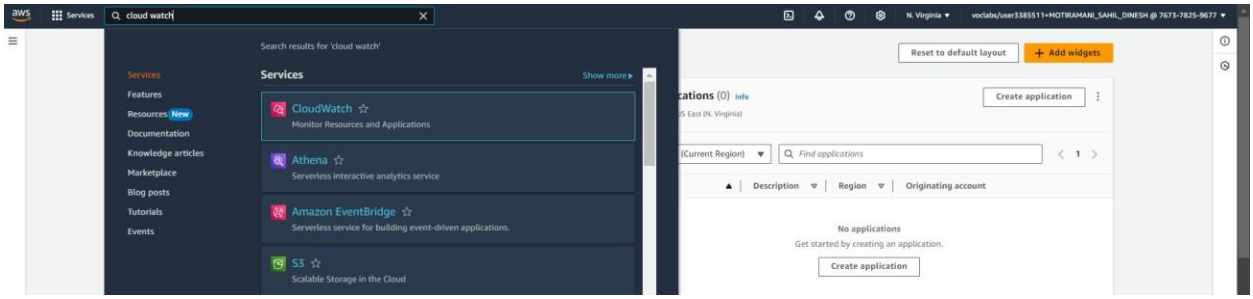
   The JSON response shows that the log entry was created successfully.



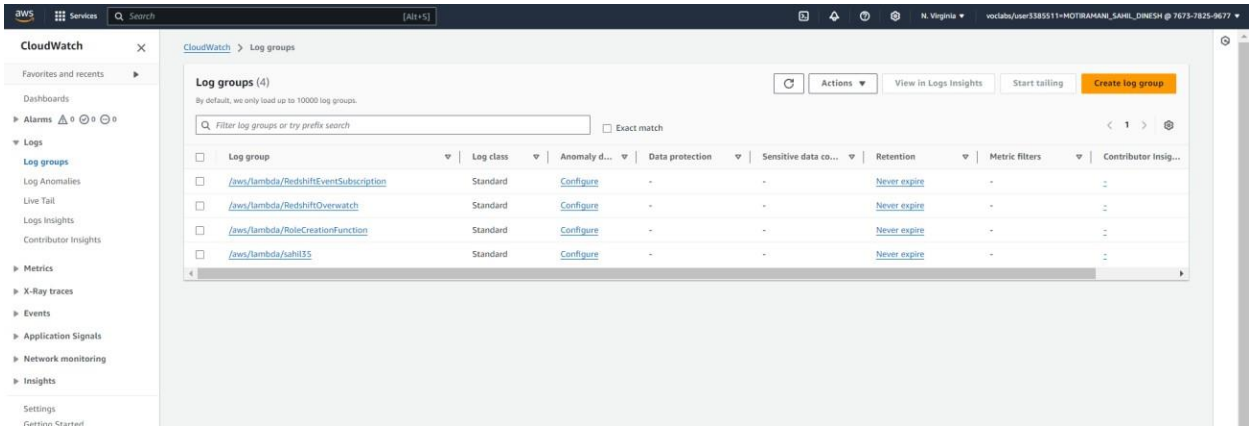
Step 3: Check the logs

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

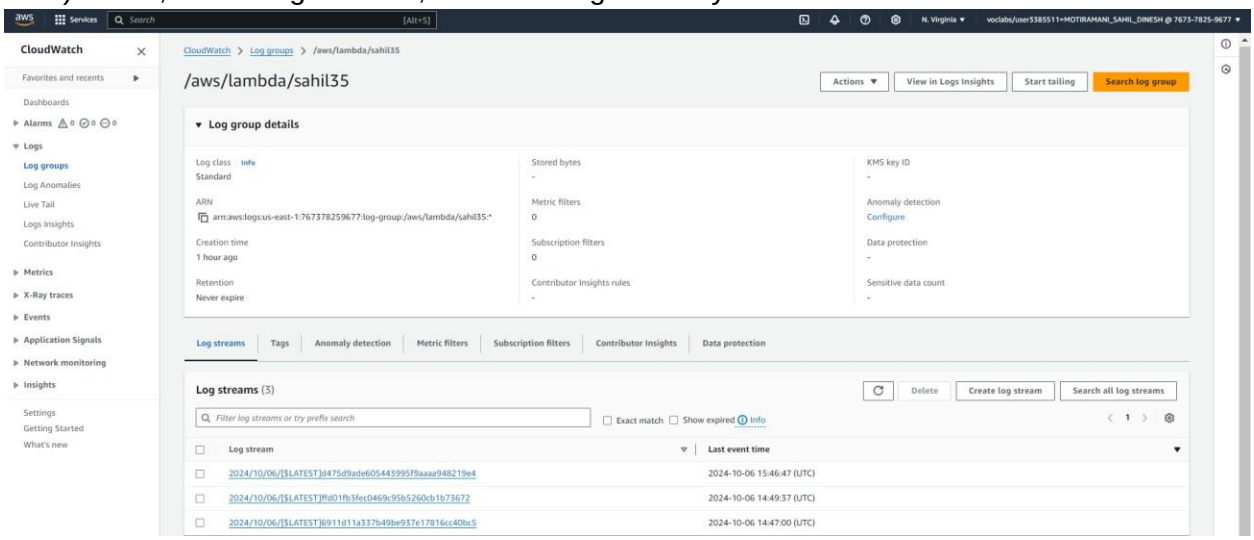
Name:Sahil Ramrakhyani Div :D15C Roll no:42



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

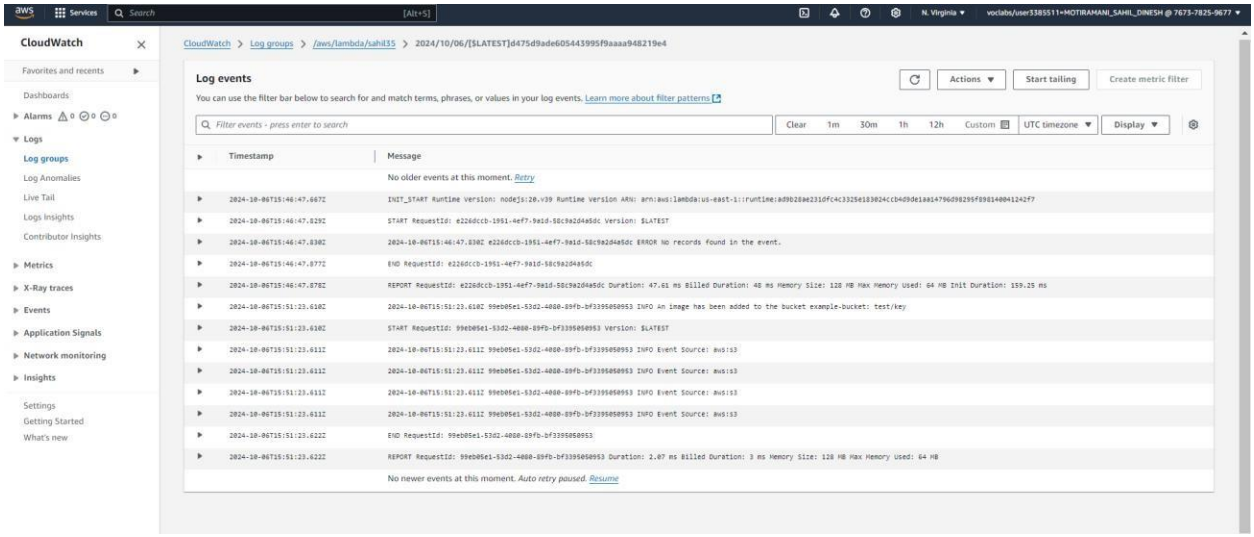


3) Here, under Log streams, select the log stream you want to check.



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

Name:Sahil Ramrakhyani Div :D15C Roll no:42



Conclusion:

In this experiment, we developed and deployed a Lambda function designed to respond to file uploads in an S3 bucket. The function was triggered automatically whenever a new object was added to the bucket, illustrating how AWS services can efficiently automate workflows. The Lambda function extracted and logged key details from the event, such as the bucket's name and the object's key. We tested this by uploading a sample file, and upon reviewing the logs in CloudWatch, we confirmed that the function executed successfully, capturing the upload event. This experiment demonstrated the powerful synergy between AWS Lambda and S3, enabling seamless, event-driven automation.