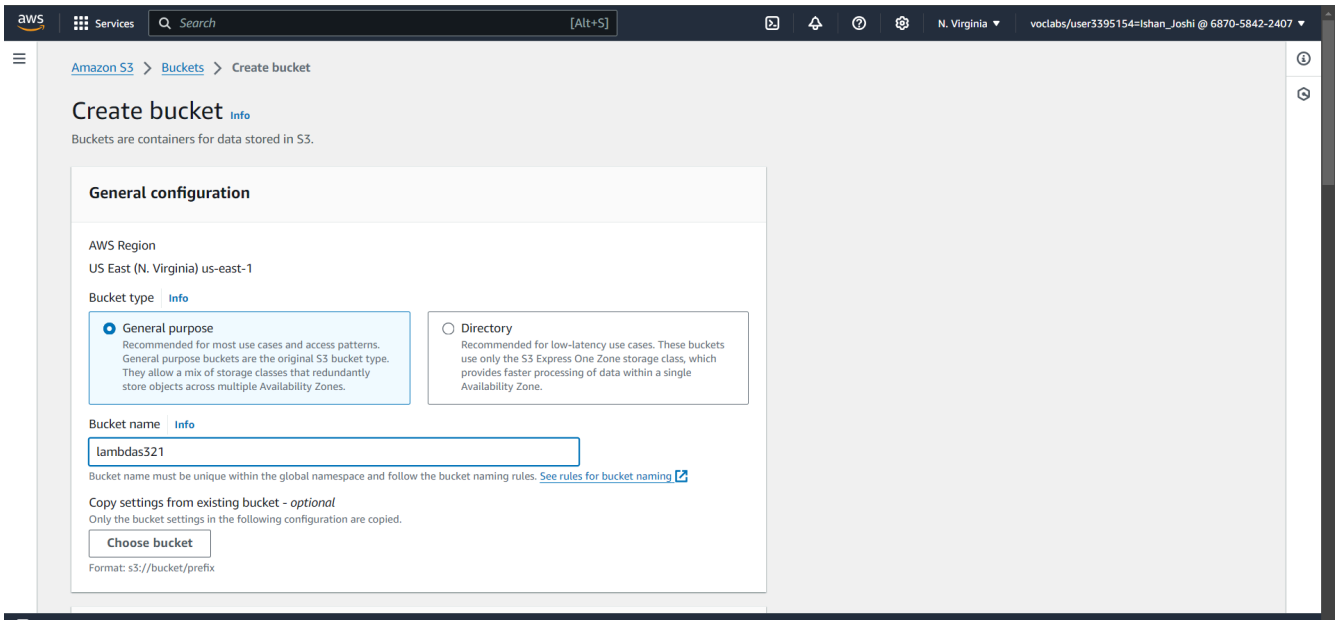
Name-Ishan Kiran Joshi Div-D15C Roll No-21 A.Y.-2024-25

Experiment 12

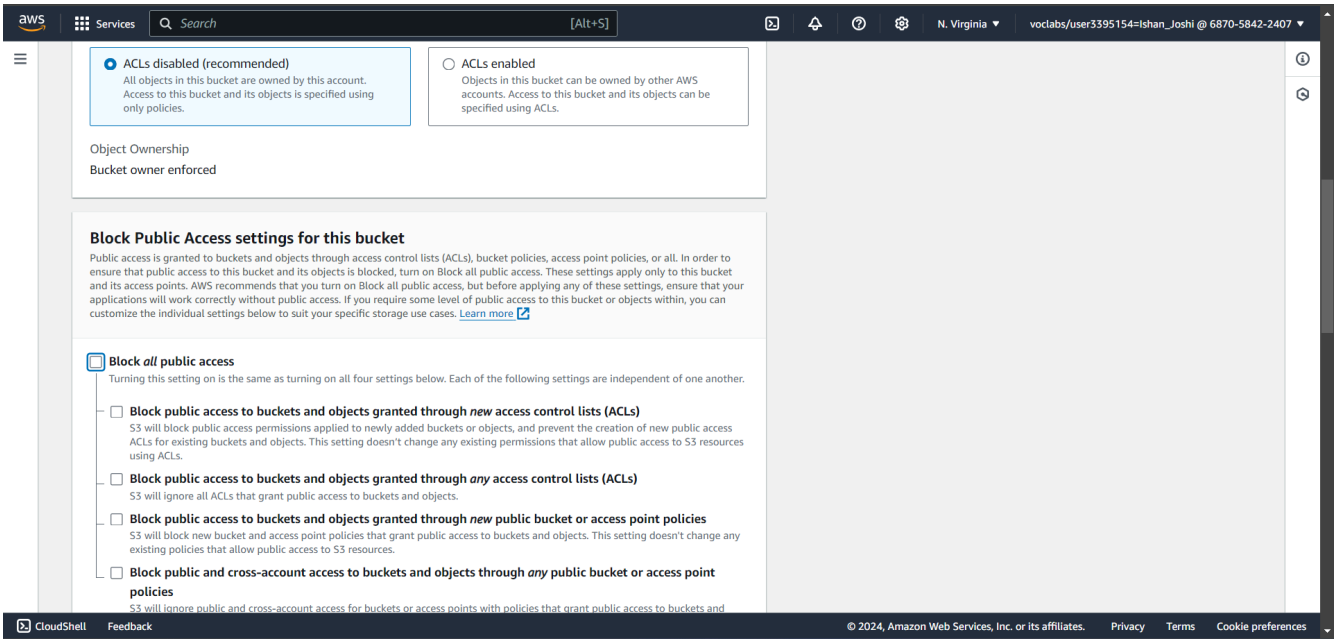
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Steps:

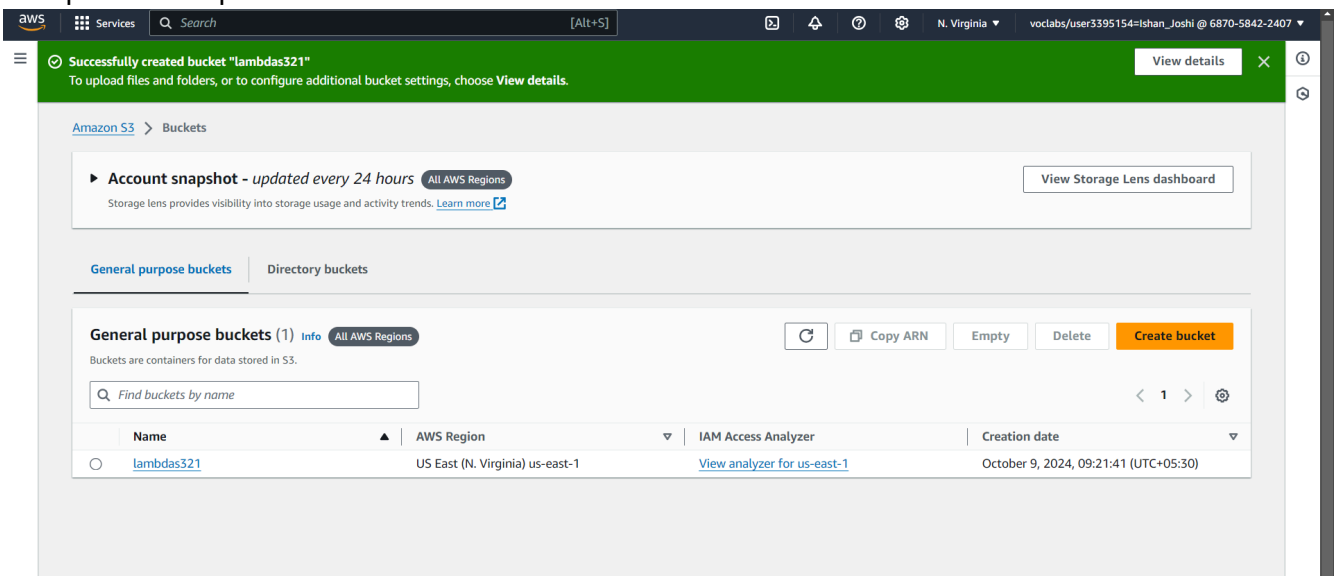
Step 1: On your AWS console, click on ‘S3’ in the services section and click on ‘Create bucket’. Give your bucket a name.



Uncheck the ‘Block all public access’ box.

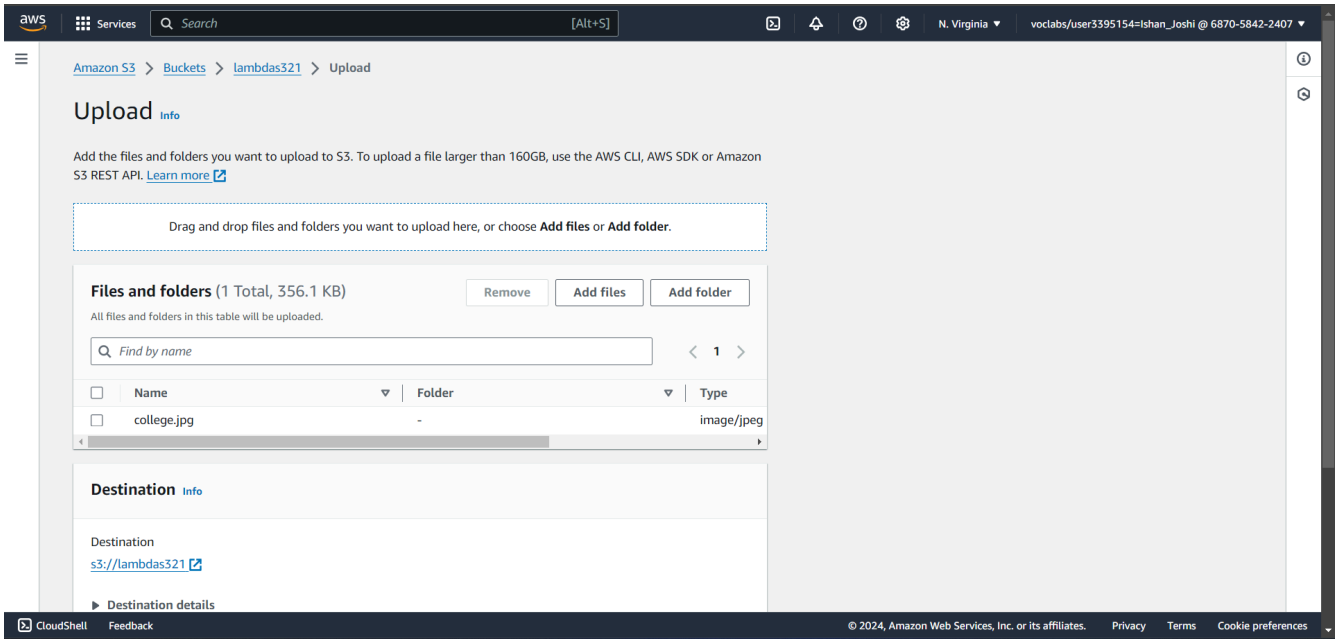


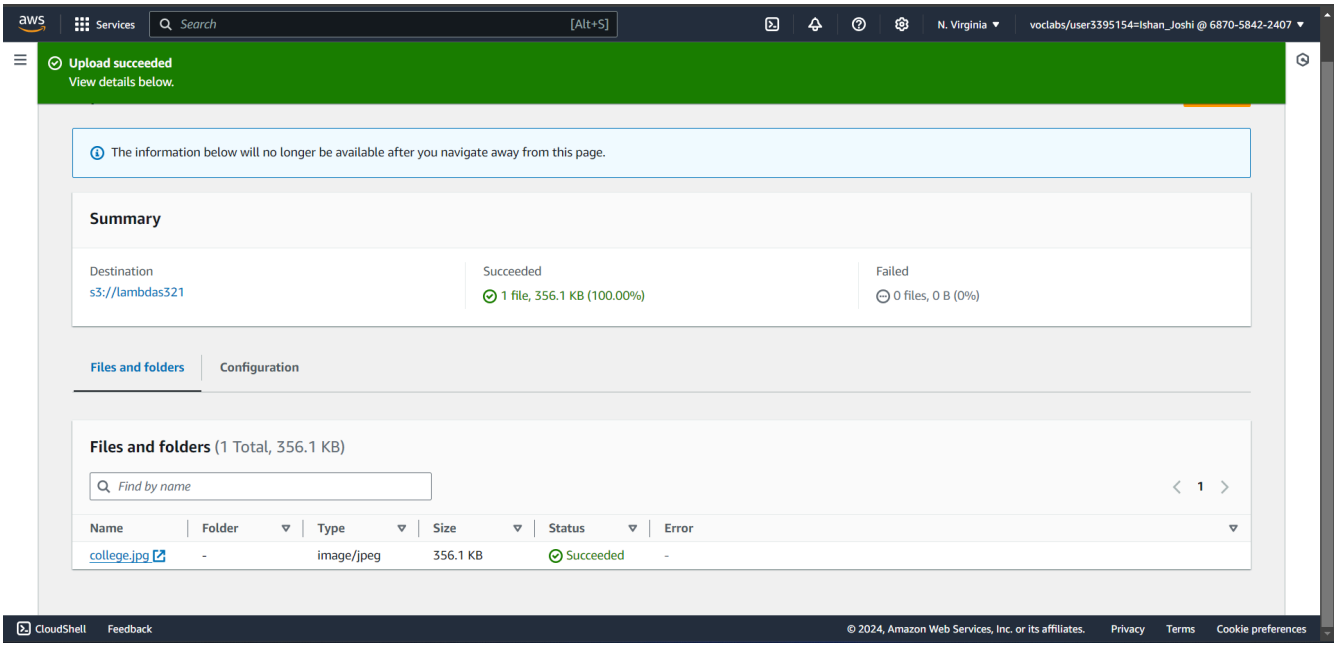
Keep all other options as default and click on ‘Create bucket’.



Your bucket is created.

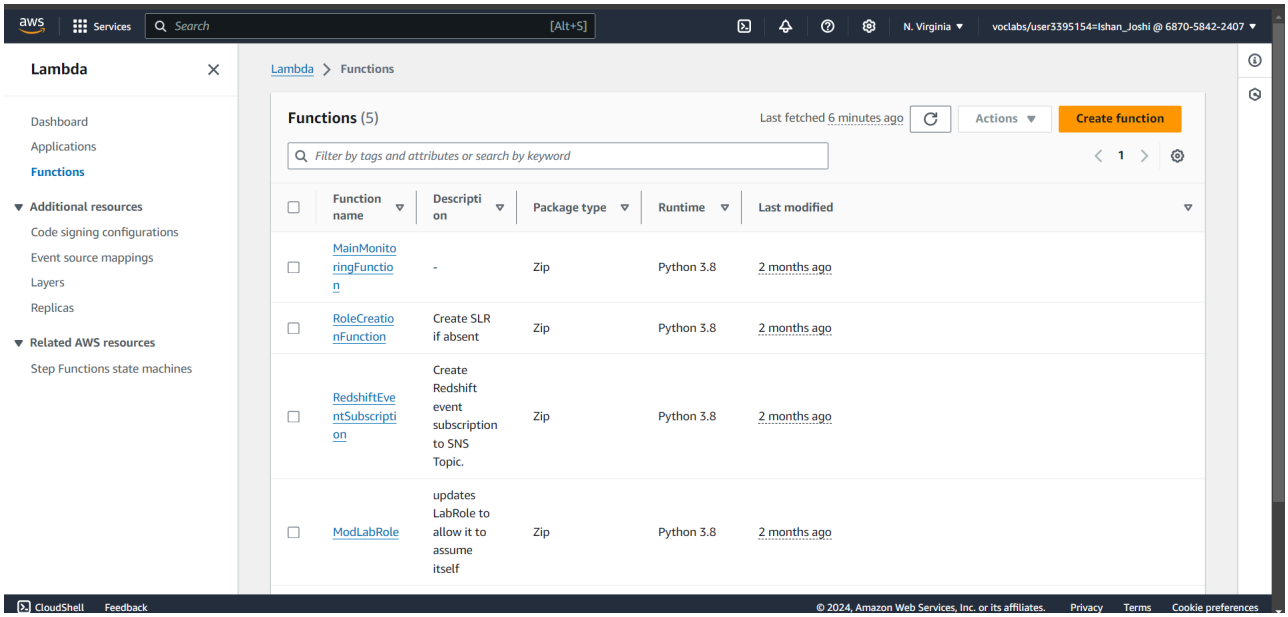
Step 2: Upload an image onto your S3 bucket by clicking on your S3 bucket, clicking on ‘Upload’, clicking on ‘Add files’, navigating to your image and selecting it.



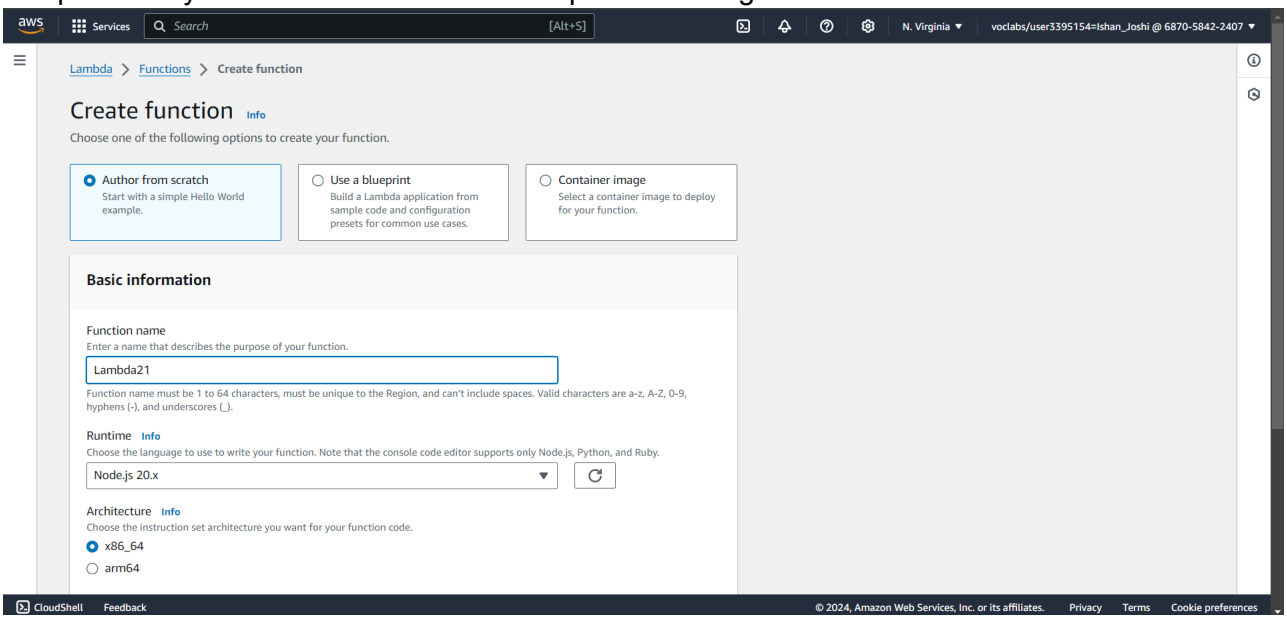


Your image gets uploaded onto the S3 bucket.

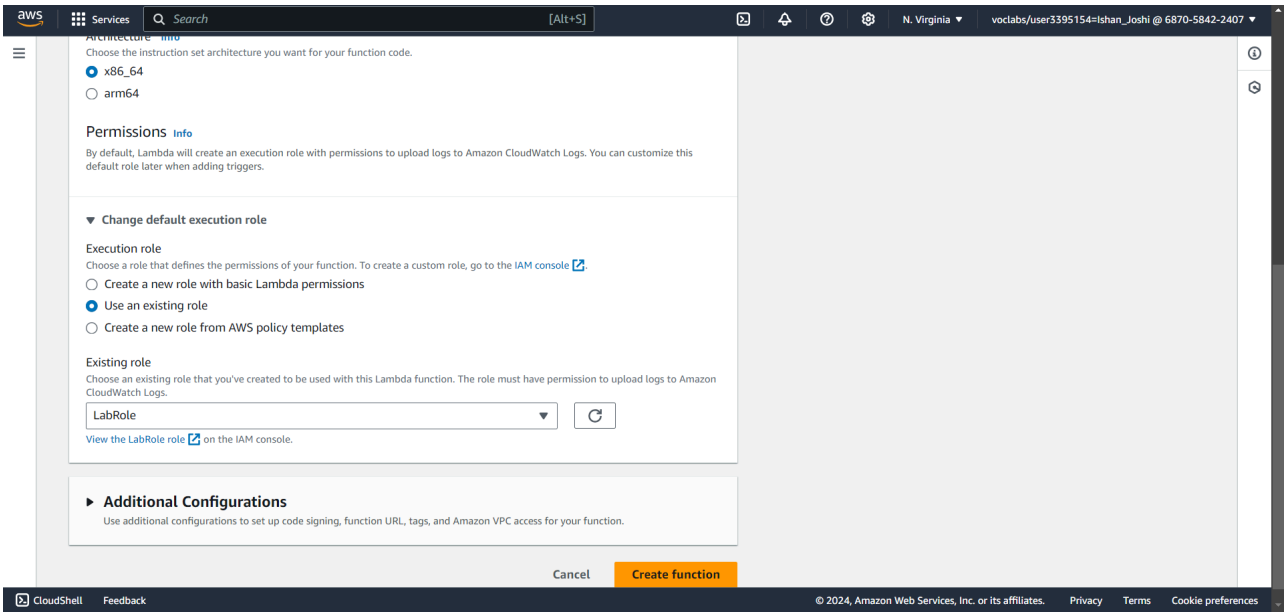
Step 3: Navigate to the AWS Lambda console using the ‘Services’ section. Click on ‘Create function’.



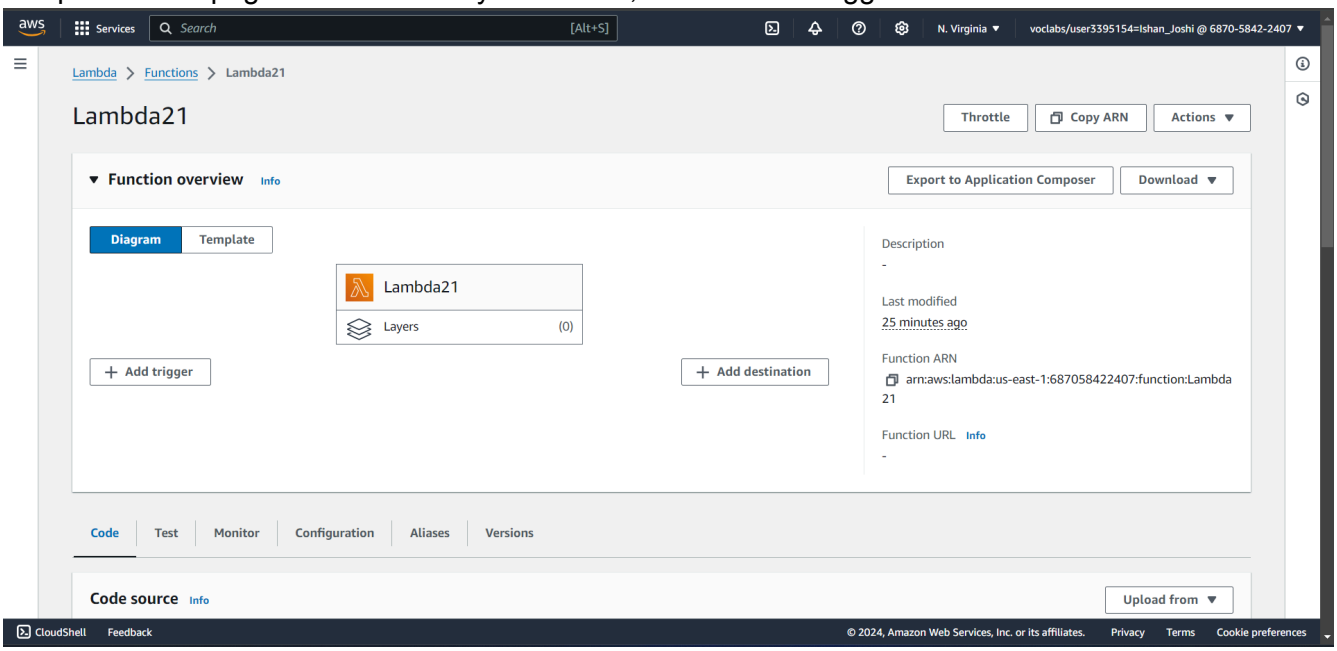
Step 4: Give your function a name and keep other settings as default.



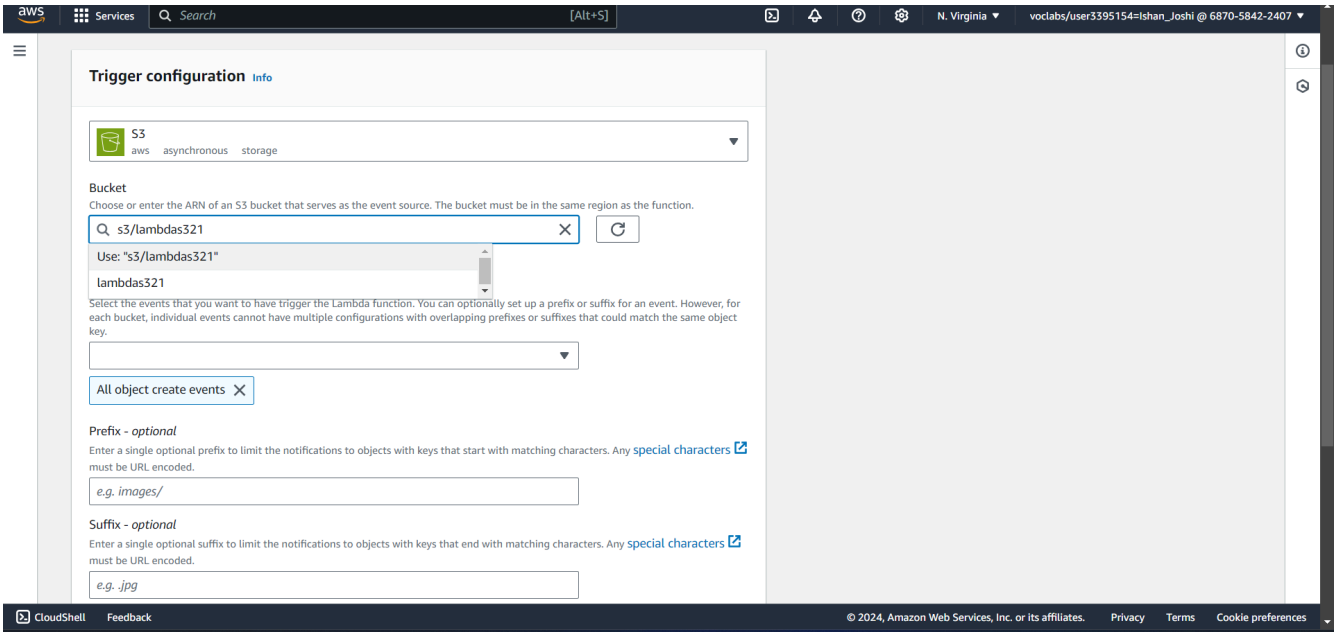
Under ‘Execution role’, choose ‘Use an existing role’ and in the dropdown box below, choose ‘LabRole’. Then, click on ‘Create function’. Your function gets created.

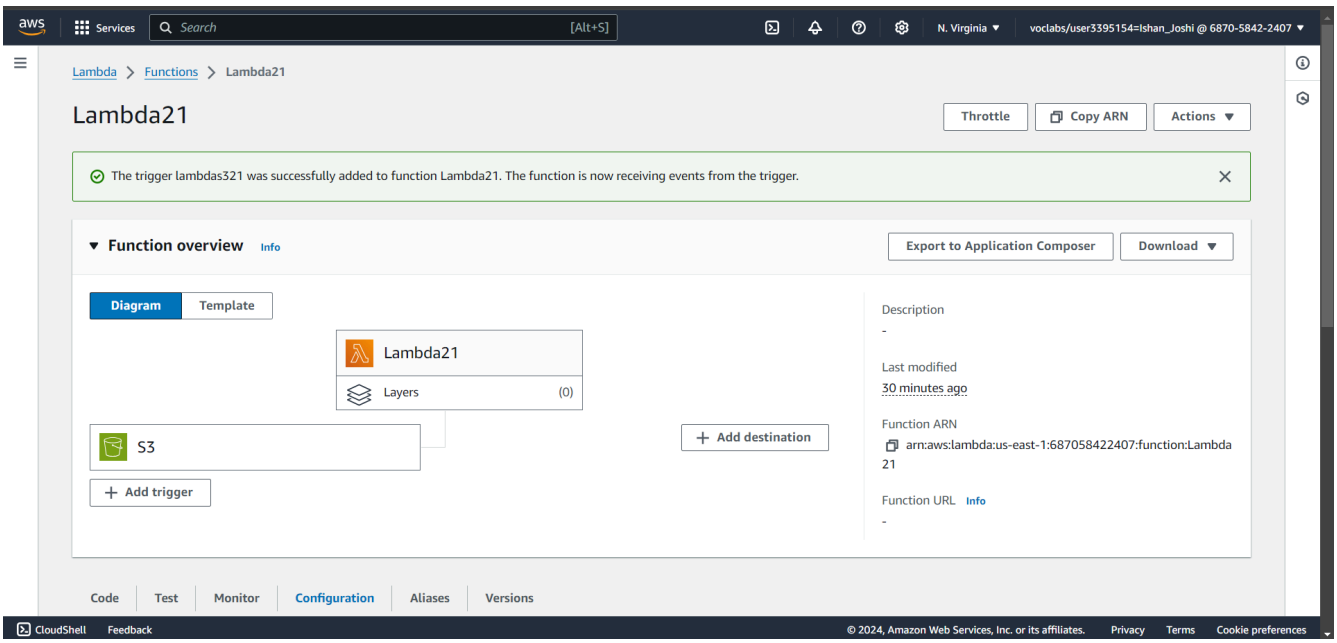


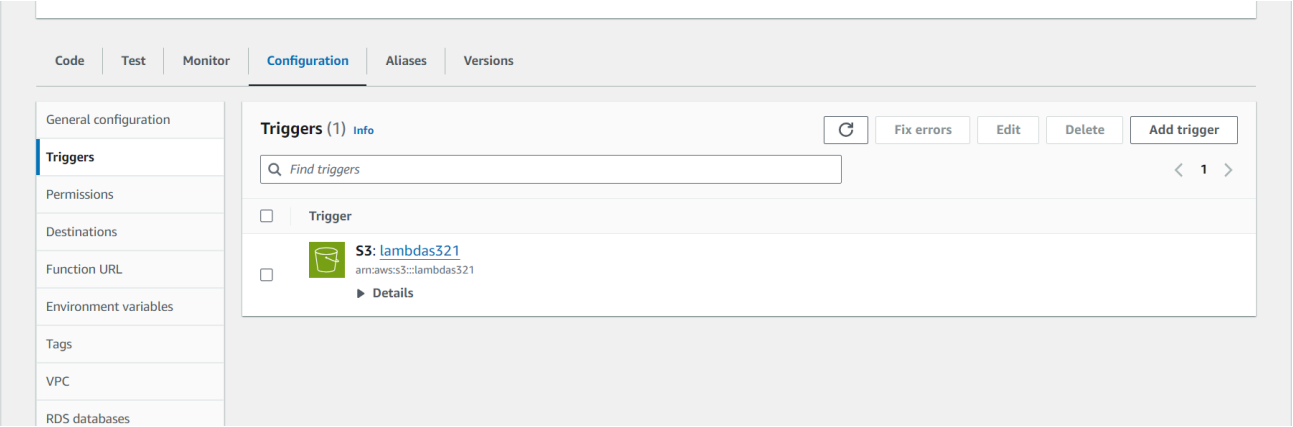
Step 5: On the page of the function you created, click on ‘Add trigger’.



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

Event name

LambdaEvent21S

Delete

Event JSON

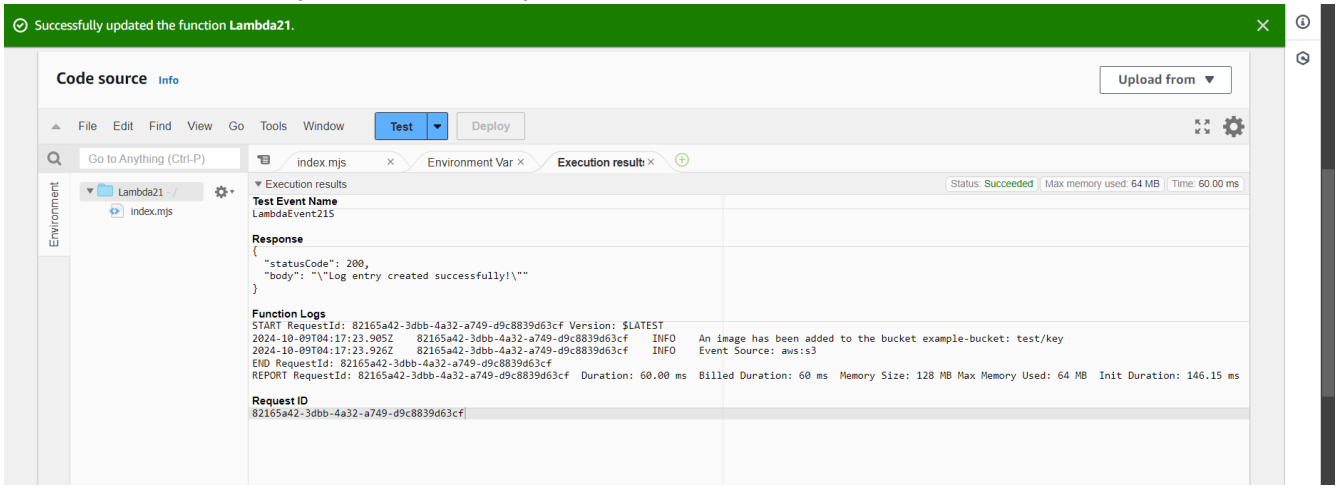
Format JSON


```
{
  "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",
```

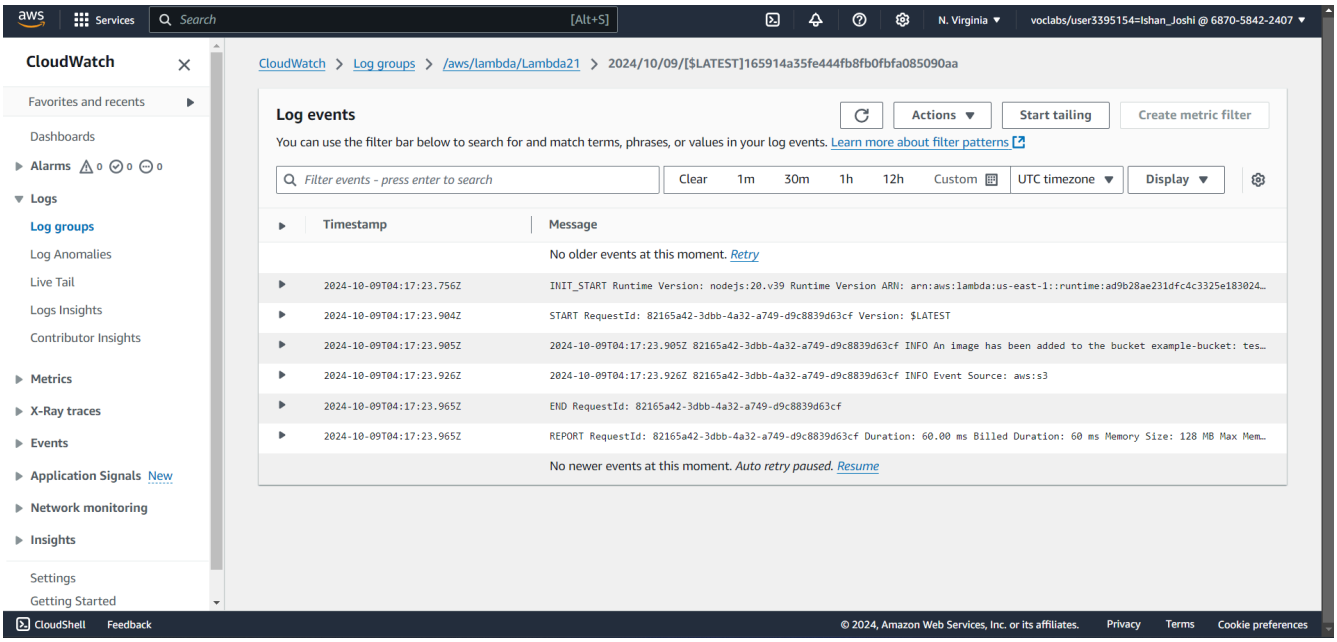

Cancel Invoke Save

Then, click on ‘Save’. Your function gets successfully updated.

Step 9: Click on ‘Deploy’ and after deployment is successful, click on ‘Test’.



Running the test gives the above output which displays that ‘An Image has been added to the bucket’ and that the log entry was successfully created.



Conclusion: In this experiment, we demonstrated how to set up a Lambda function that logs the message "An image has been added" whenever an image is uploaded to a specific S3 bucket. We started by creating an S3 bucket and uploading an image to it. Next, we set up a Lambda function, attached an S3 trigger, and selected the S3 bucket we created. After that, we configured the Lambda function's code and set up a test event. Upon running the test, the function logged key event details, such as the bucket name and object key, confirming the image upload. Additionally, Lambda logs confirmed that the image was successfully added to the bucket.