

CSCI 5410: Assignment 3

Part A: Build an event-driven serverless application using AWS Lambda

GitLab Repository Link: https://git.cs.dal.ca/drshah/dhrumilrakeshshah_csci5410.git

Screenshots of the Part A:

```

Main ×
C:\Users\shahd\.jdks\corretto-1.8.0_292\bin\java.exe ...
Choose any of the below tasks.
1. Create two new buckets.
2. Upload all files in tech folder to the firstb00870600 bucket.
3. Exit.
1
A new bucket 'firstb00870600' is created.
A new bucket 'secondb0870600' is created.
Choose any of the below tasks.
1. Create two new buckets.
2. Upload all files in tech folder to the firstb00870600 bucket.
3. Exit.

```

Figure 1: Two S3 Buckets Created 1

The screenshot shows the AWS S3 Management Console interface. On the left, there is a sidebar with navigation links for Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, Access analyzer for S3, Block Public Access settings for this account, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3.

The main content area displays the "Account snapshot" which includes a storage lens dashboard. Below it is a table titled "Buckets (2) Info" showing the details of two buckets:

Name	AWS Region	Access	Creation date
firstb00870600	US East (N. Virginia) us-east-1	Objects can be public	October 26, 2021, 13:49:30 (UTC-03:00)
secondb0870600	US East (N. Virginia) us-east-1	Objects can be public	October 26, 2021, 13:49:30 (UTC-03:00)

At the bottom of the page, there are links for Feedback, English (US), Privacy Policy, Terms of Use, Cookie preferences, and a timestamp indicating the page was last updated at 02:06 PM on 26-10-2021.

Figure 2: Two S3 Buckets Created 2

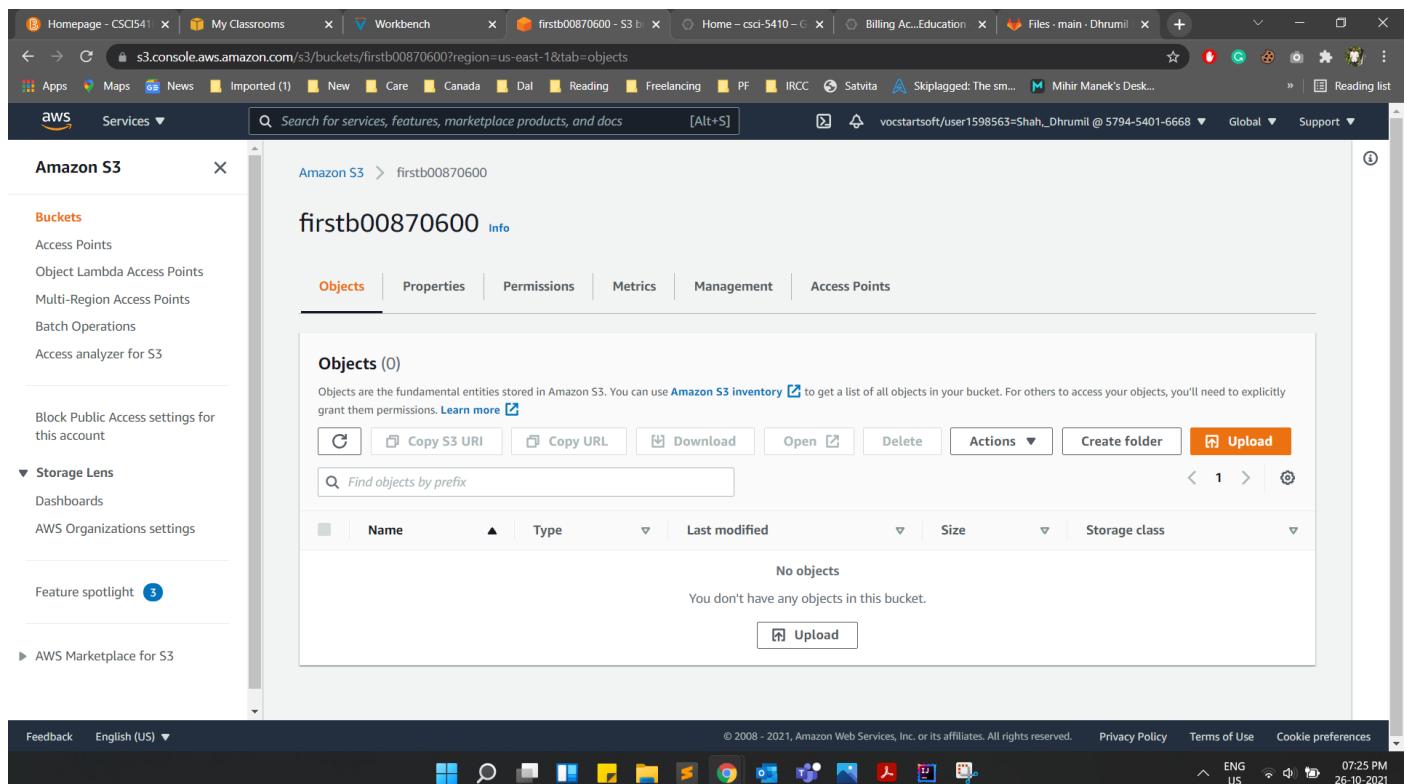


Figure 3: First Empty Bucket (firstb00870600)

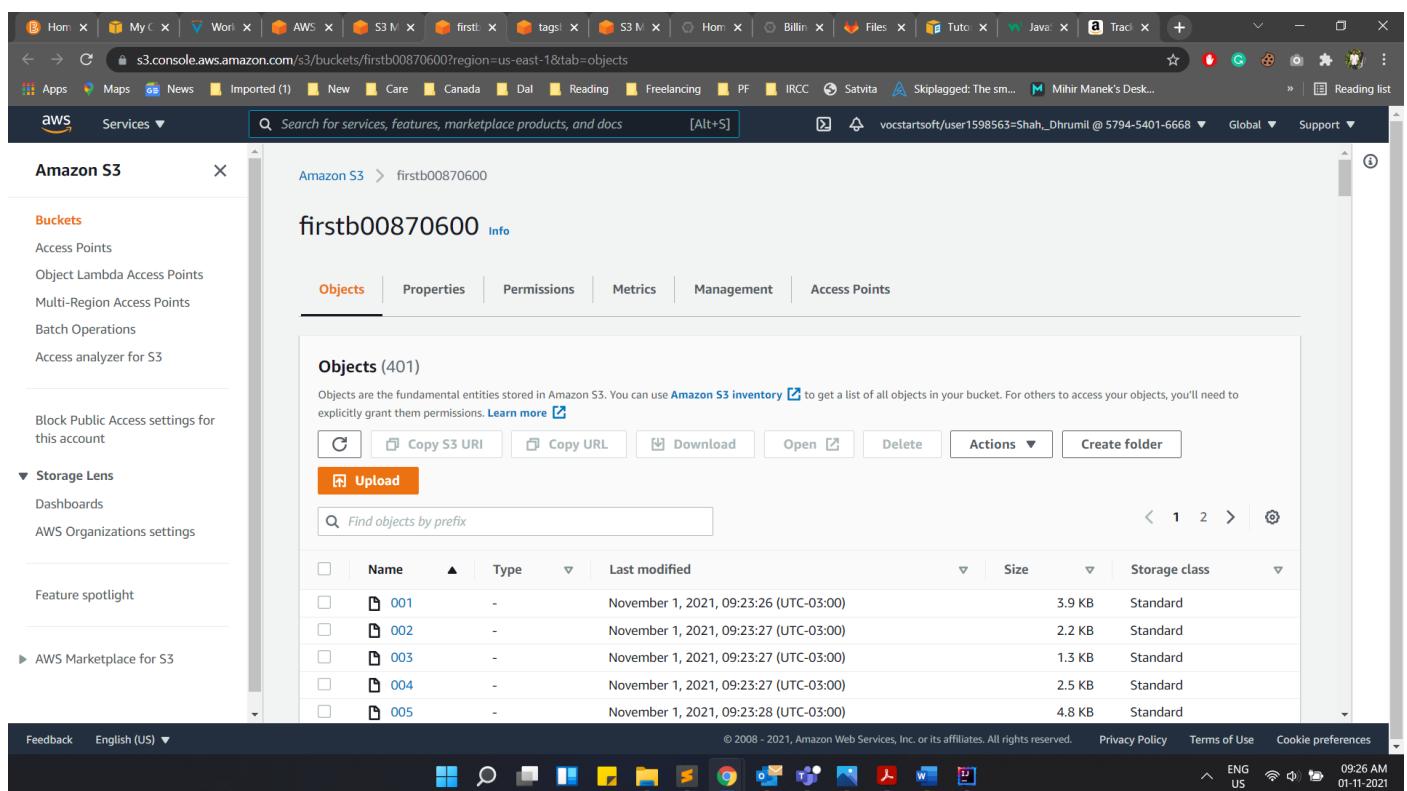


Figure 4: First Filled Bucket (firstb00870600)

The screenshot shows the AWS S3 console interface. On the left, a sidebar menu includes 'Buckets', 'Storage Lens', 'Feature spotlight', and 'AWS Marketplace for S3'. The main content area displays the 'secondb0870600' bucket. At the top, there are tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. Below this is a section titled 'Objects (0)' with a message stating 'No objects'. There is a large 'Upload' button at the bottom of this section. The browser's address bar shows the URL `s3.console.aws.amazon.com/s3/buckets/secondb0870600?region=us-east-1&tab=objects`. The status bar at the bottom right indicates the date as 26-10-2021 and the time as 07:25 PM.

Figure 5: Second Empty Bucket (secondb0870600)

The screenshot shows the AWS S3 console interface, similar to Figure 5 but for a different bucket. The main content area displays the 'tagsb0870600' bucket. The 'Objects' tab is selected, showing 'Objects (0)' with a message 'No objects'. There is a large 'Upload' button at the bottom. The browser's address bar shows the URL `s3.console.aws.amazon.com/s3/buckets/tagsb0870600?region=us-east-1&tab=objects`. The status bar at the bottom right indicates the date as 29-10-2021 and the time as 12:25 PM.

Figure 6: Third Empty Tags Bucket (tagsb0870600)

Buckets

- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

Feedback English (US) ▾

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

ENG US 09:27 AM 01-11-2021

Figure 7: Third Filled Tags Bucket (tagsb00870600)

```

public void fileUploadToS3Bucket(AmazonS3 s3Object, String bucketName) {
    try {
        // Looping through the files in tech folder
        for (int i = 1; i <= 401; i++) {
            // Declaring the filePrefix and fileName
            String filePrefix = "";
            String fileName = "";

            // Conditioning through the files in tech folder
            if (i < 10) {
                filePrefix = "00" + i;
                fileName = "tech/00" + i + ".txt";
            } else if (i > 9 && i < 100) {
                filePrefix = "0" + i;
                fileName = "tech/0" + i + ".txt";
            } else {
                filePrefix = "" + i;
                fileName = "tech/" + i + ".txt";
            }

            // Instantiating the PutObjectRequest class
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
                filePrefix, new File(fileName));

            // Uploading the file to the S3 bucket
            s3Object.putObject(putObjectRequest);

            // Printing to the console
        }
    } catch (AmazonServiceException e) {
        System.out.println("Error occurred while uploading file");
        e.printStackTrace();
    }
}

```

Figure 8: S3Task Code to Upload the Files to the S3 Bucket 1

The screenshot shows the IntelliJ IDEA interface with the project 'dhrumila3 [A3PartA]' open. The 'Main.java' file is the current editor tab. The code implements a task to upload files from a local directory to an S3 bucket. It uses the AWS SDK's PutObjectRequest class to upload files with specific prefixes and handles exceptions.

```

    75 } else {
    76     filePrefix = "" + i;
    77     fileName = "tech/" + i + ".txt";
    78 }
    79
    80 // Instantiating the PutObjectRequest class
    81 PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
    82     filePrefix, new File(fileName));
    83
    84 // Uploading the file to the S3 bucket
    85 s3Object.putObject(putObjectRequest);
    86
    87 // Printing to the console
    88 System.out.format("The file '%s' is pushed to the '%s' bucket\n",
    89     filePrefix, bucketName);
    90
    91 // Adding a delay of 200 milliseconds after adding a new file to the bucket
    92 Thread.sleep( millis: 200);
    93
    94 // Printing that 200 milliseconds sleep is added
    95 System.out.println("Added sleep of 200 millis");
    96 }
    97 // Catching the exception
    98 } catch (Exception e) {
    99     System.err.println(e.getMessage());
    100 }
    101 }
    102 }
    103

```

Figure 9: S3Task Code to Upload the Files to the S3 Bucket 2

The screenshot shows the IntelliJ IDEA interface with the 'Run' tool window active. The log output shows multiple entries indicating files are being uploaded to an S3 bucket named 'firstb00870600'. The log also includes instructions for the user to choose tasks related to bucket creation and file upload.

```

    Run: Main
    Added sleep of 200 millis
    The file '390.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '391.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '392.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '393.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '394.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '395.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '396.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '397.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '398.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '399.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '400.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    The file '401.txt' is pushed to the 'firstb00870600' bucket
    Added sleep of 200 millis
    Choose any of the below tasks.
    1. Create two new buckets.
    2. Upload all files in tech folder to the firstb00870600 bucket.
    3. Exit.

```

Figure 10: S3Task Code to Upload the Files to the S3 Bucket 3

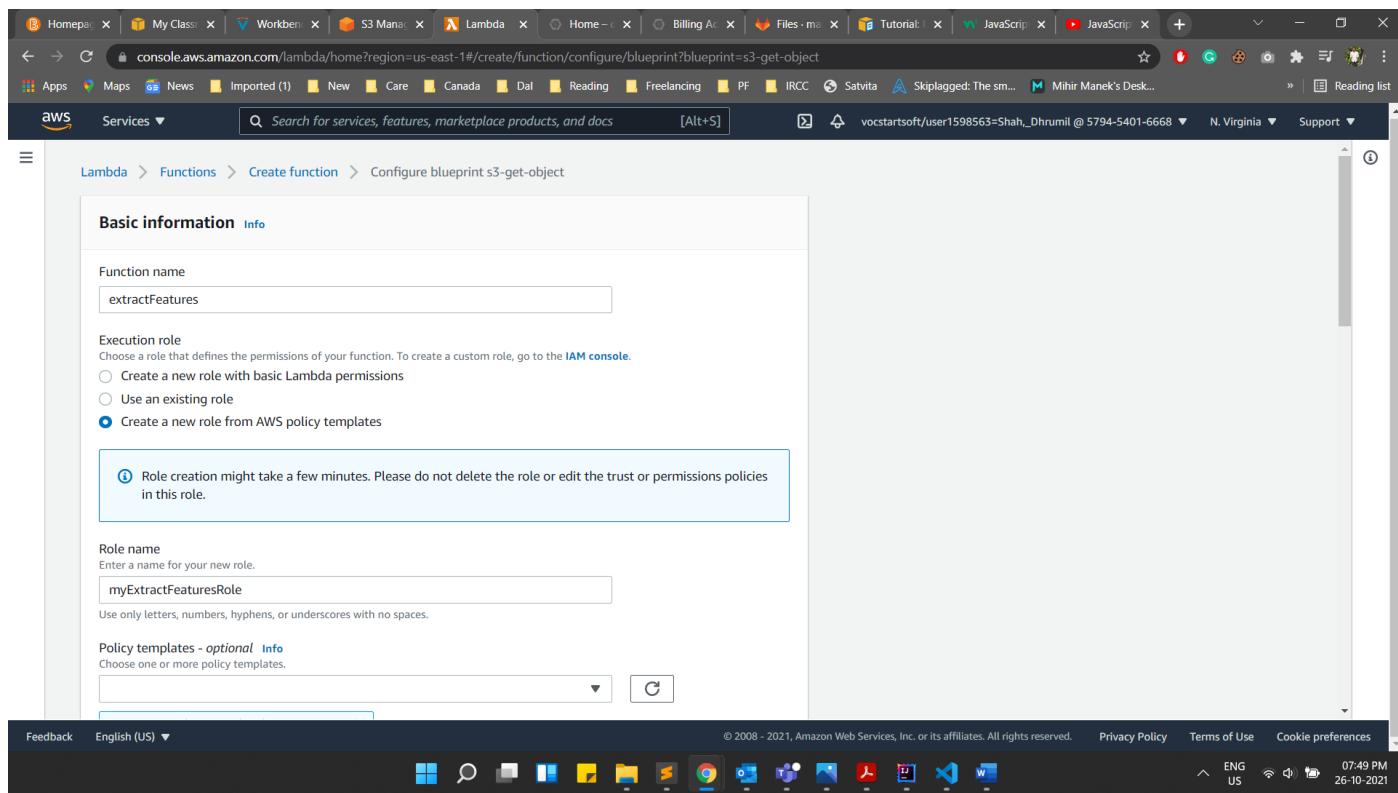


Figure 11: First Lambda Function (extractFeatures) 1

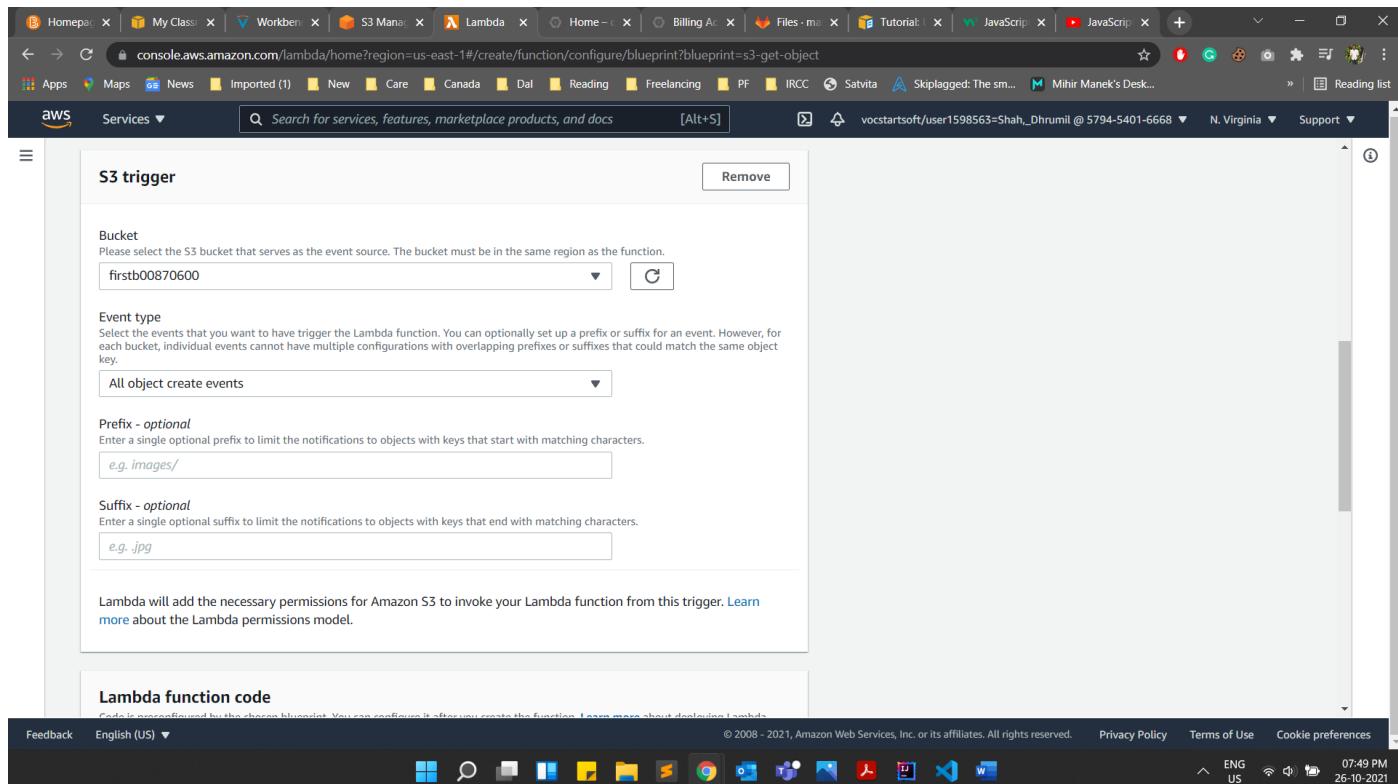


Figure 12: First Lambda Function (extractFeatures) 2

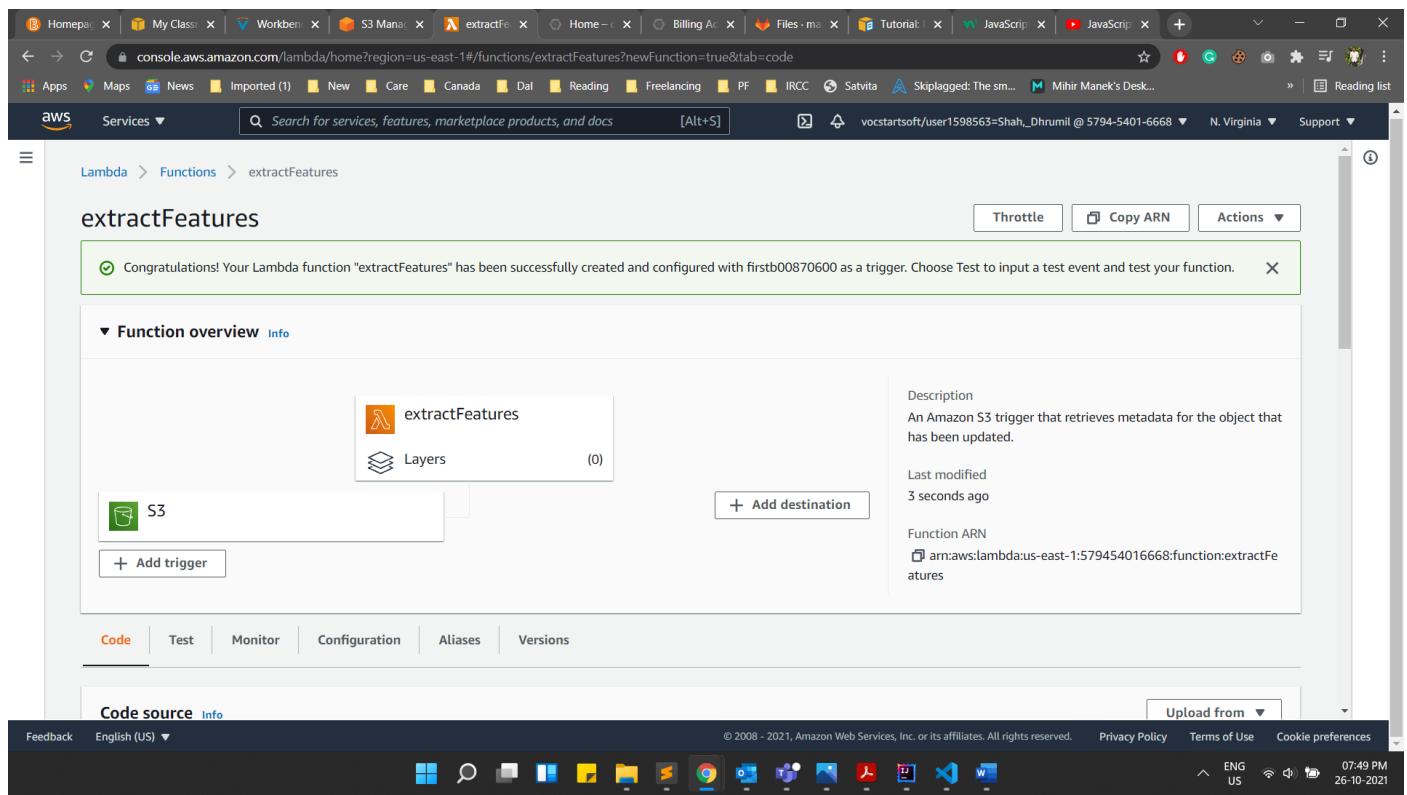


Figure 13: First Lambda Function (extractFeatures) 3

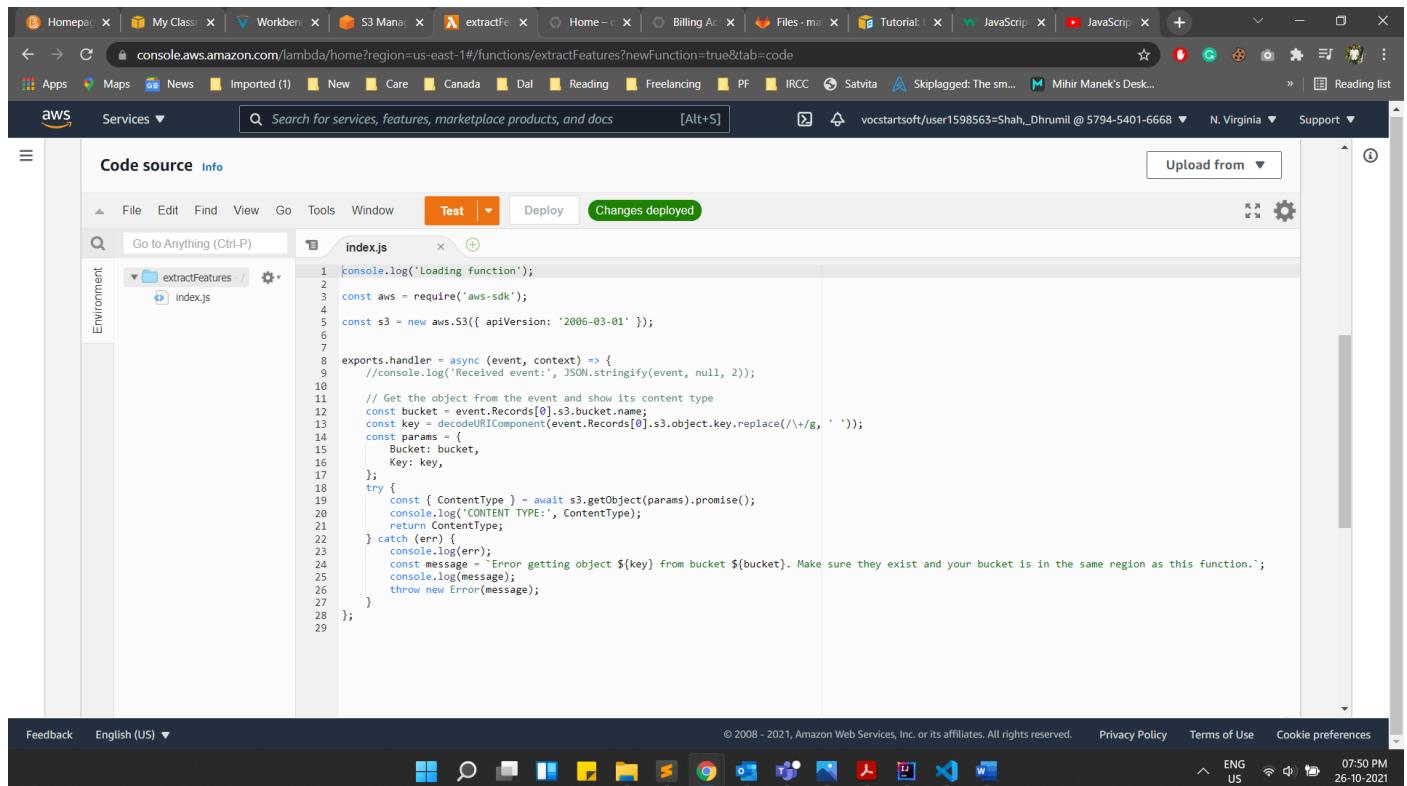


Figure 14: First Lambda Function (extractFeatures) 4

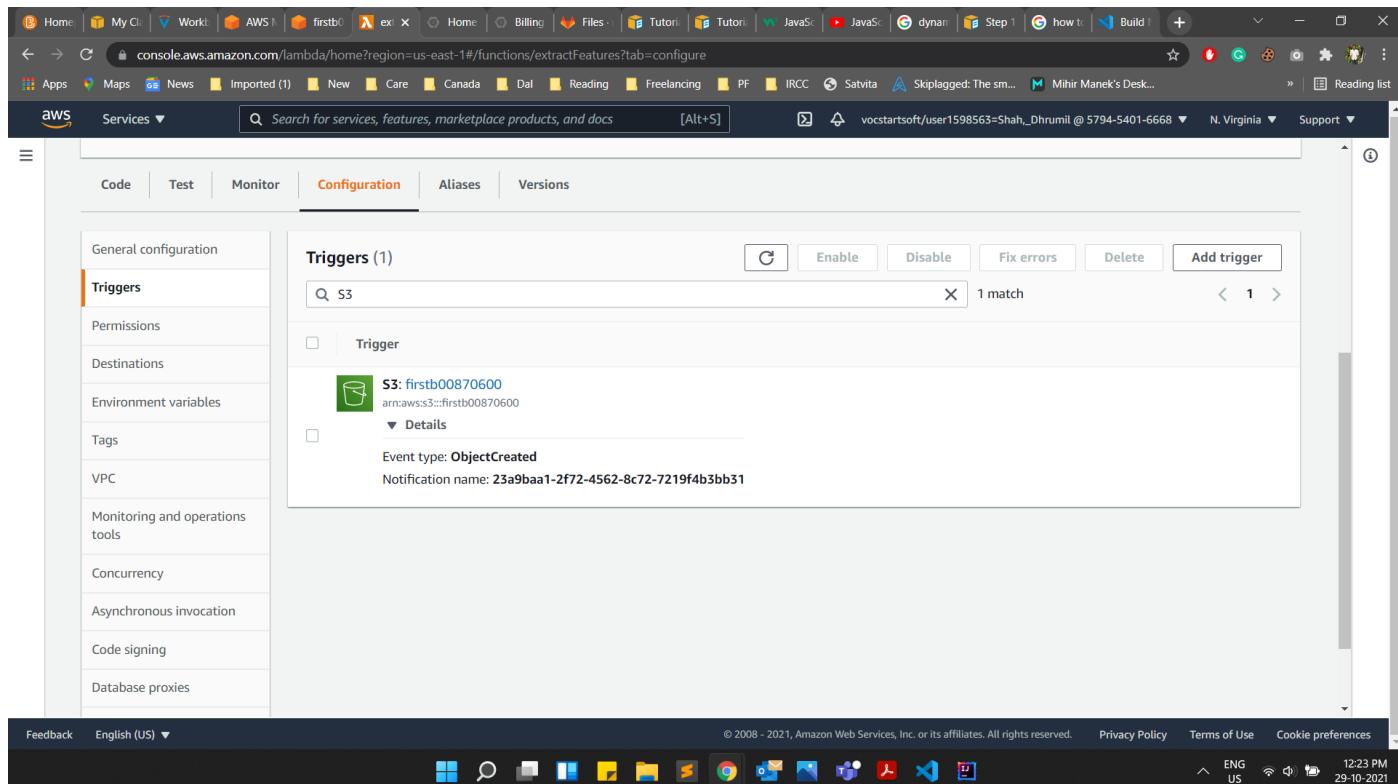


Figure 15: S3 Trigger added to the extractFeatures Lambda Function

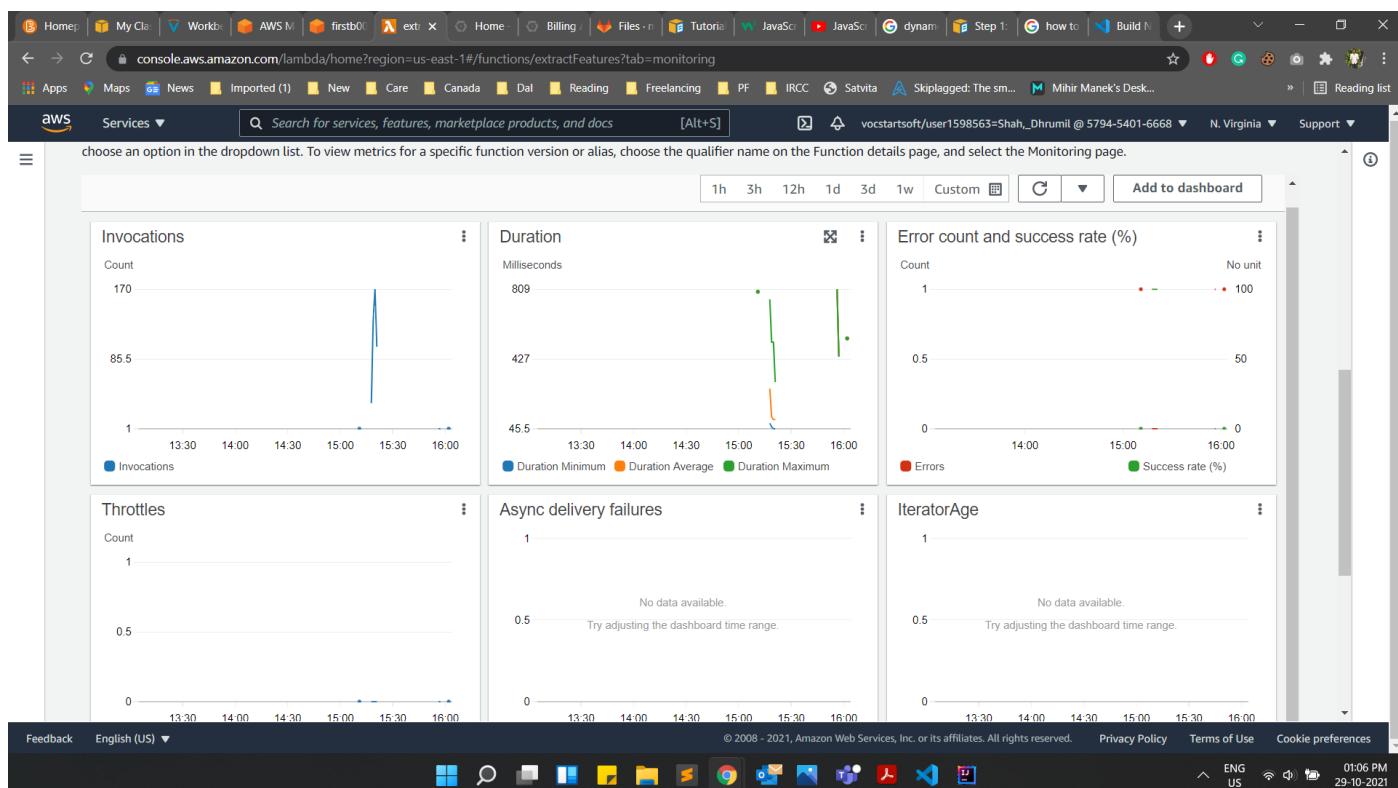


Figure 16: First Lambda Function (extractFeatures) – CloudWatch Logs Insights I

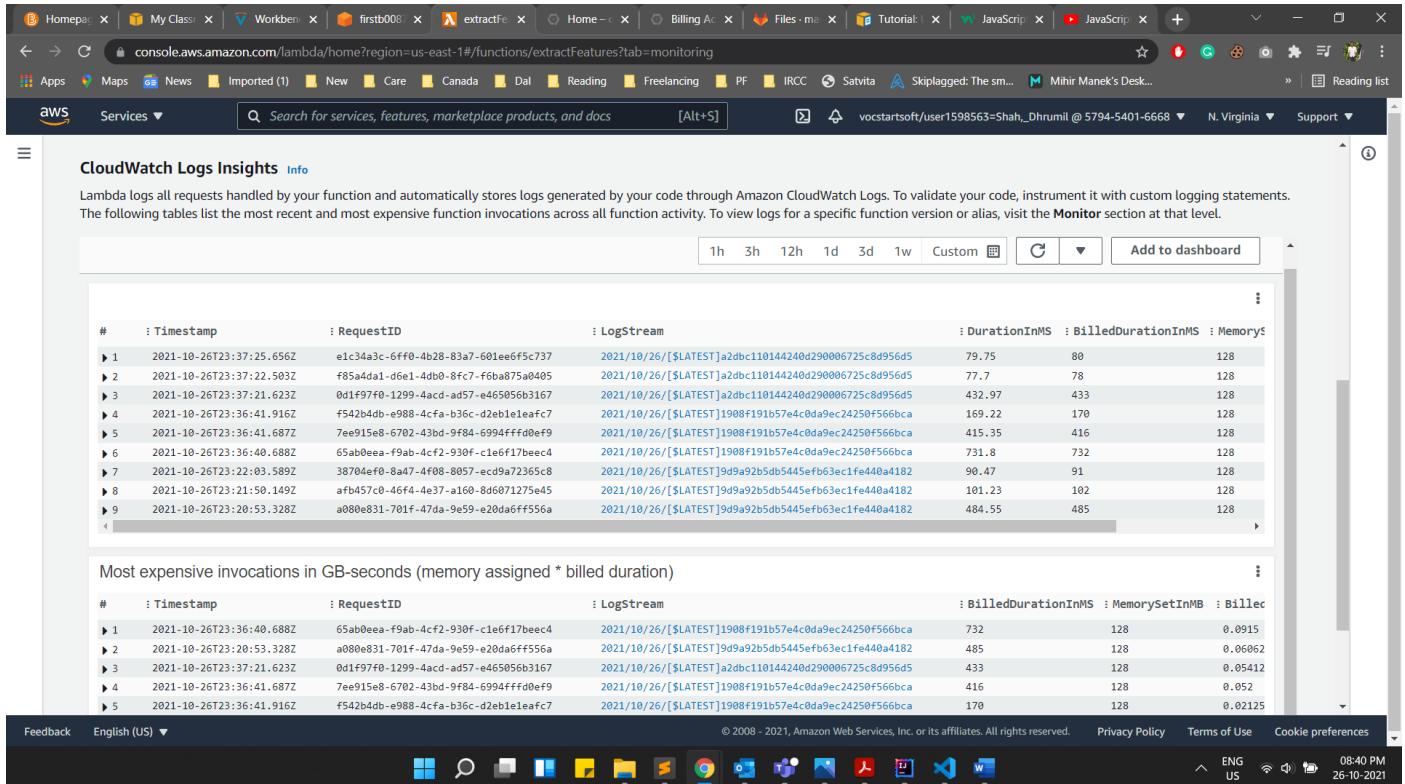


Figure 17: First Lambda Function (extractFeatures) – CloudWatch Logs Insights 2

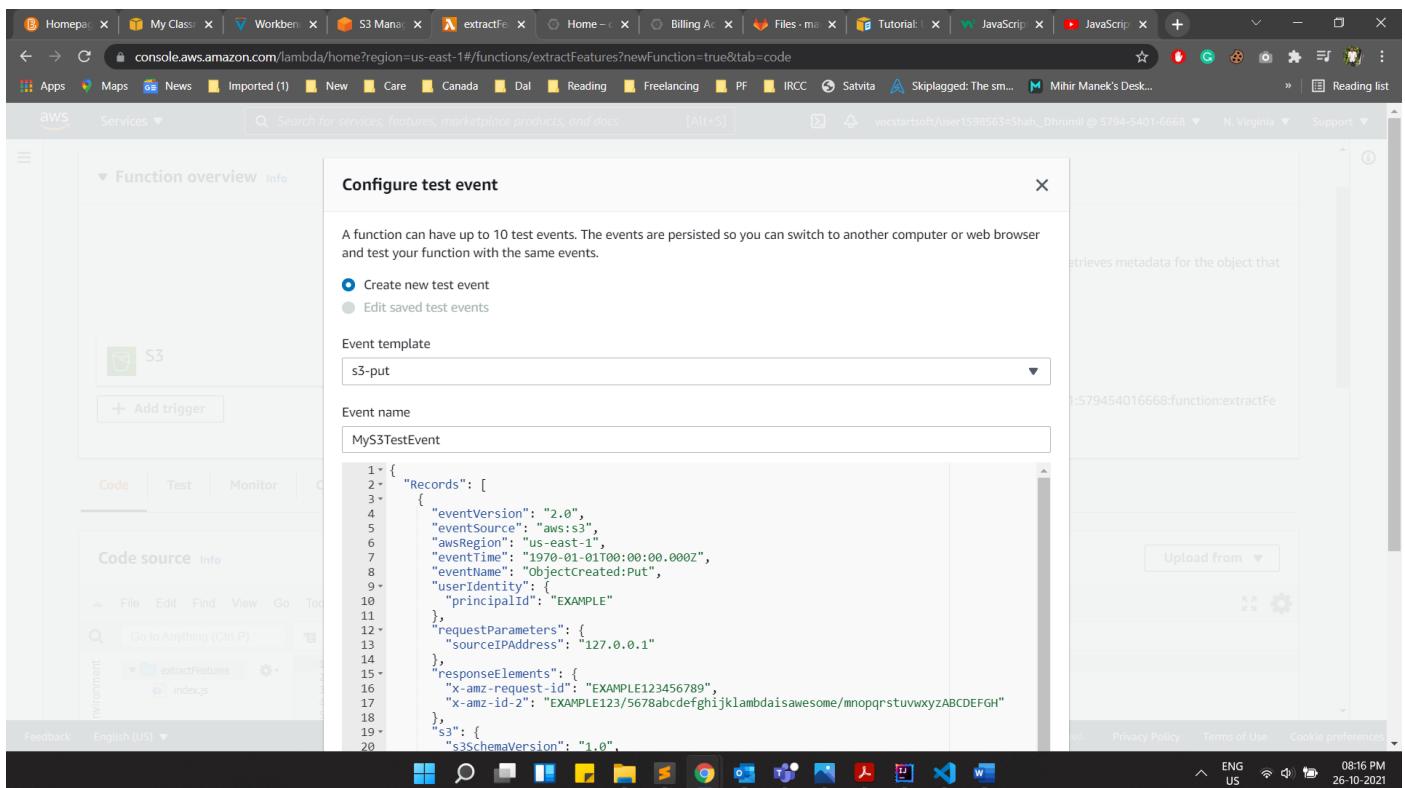


Figure 18: extractFeatures Lambda Function Configure Test Event 1

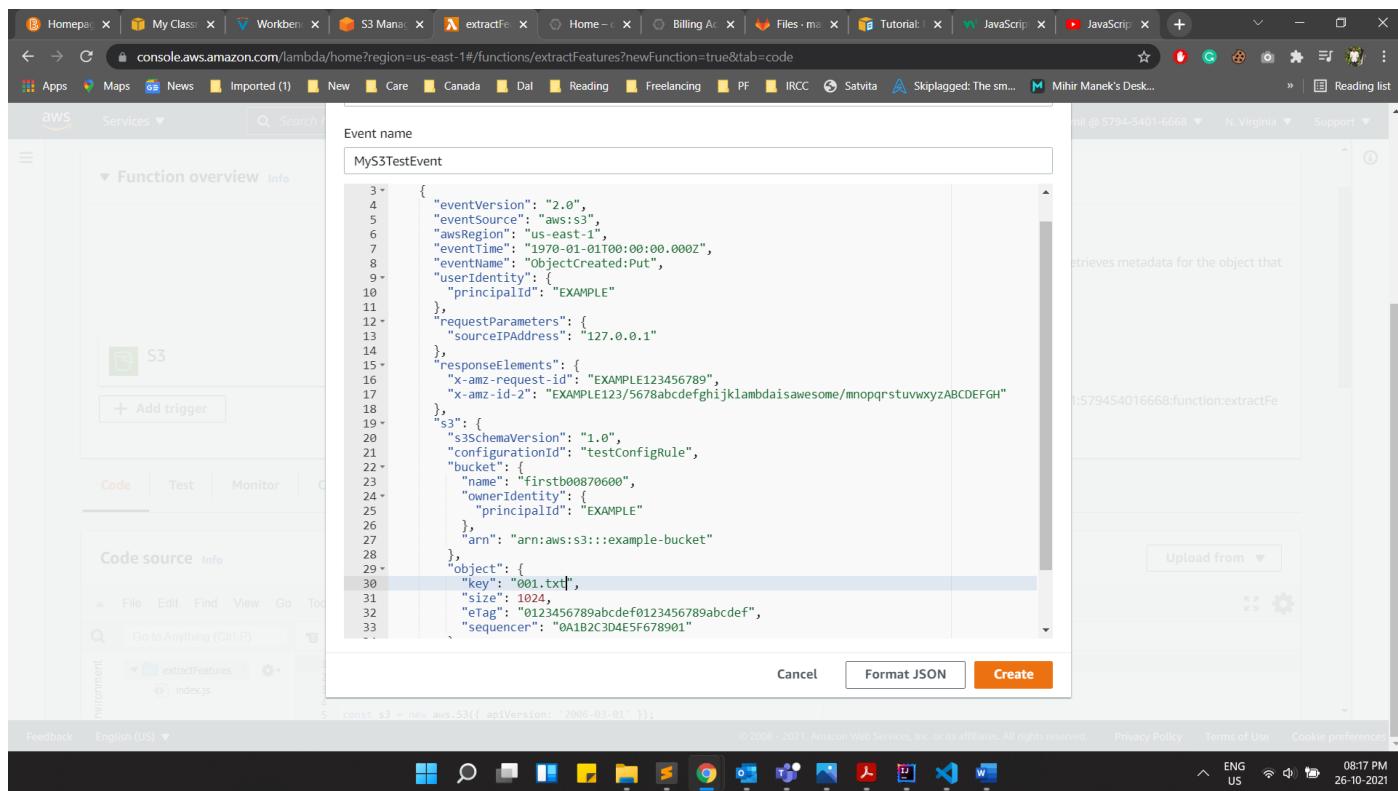


Figure 19: extractFeatures Lambda Function Configure Test Event 2

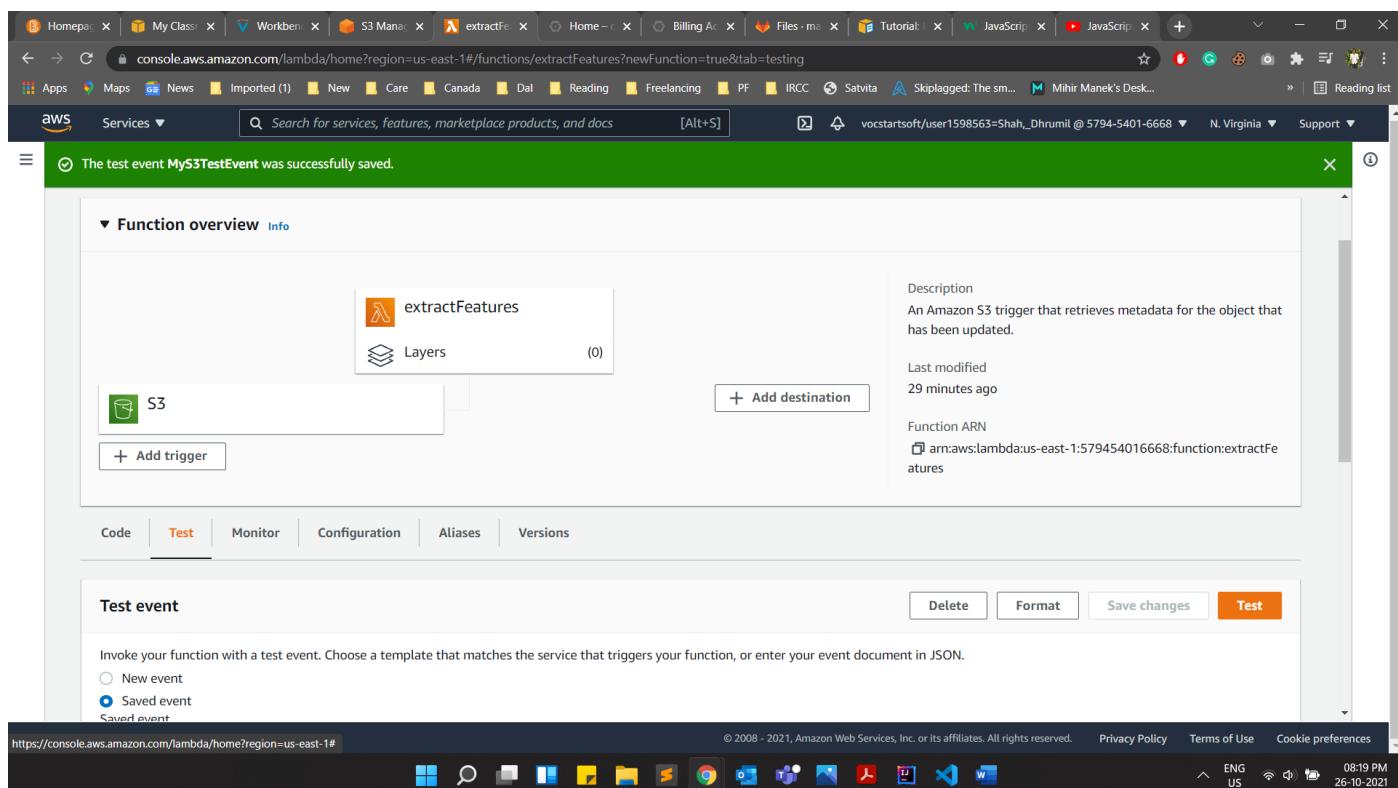


Figure 20: extractFeatures Lambda Function Configure Test Event 3

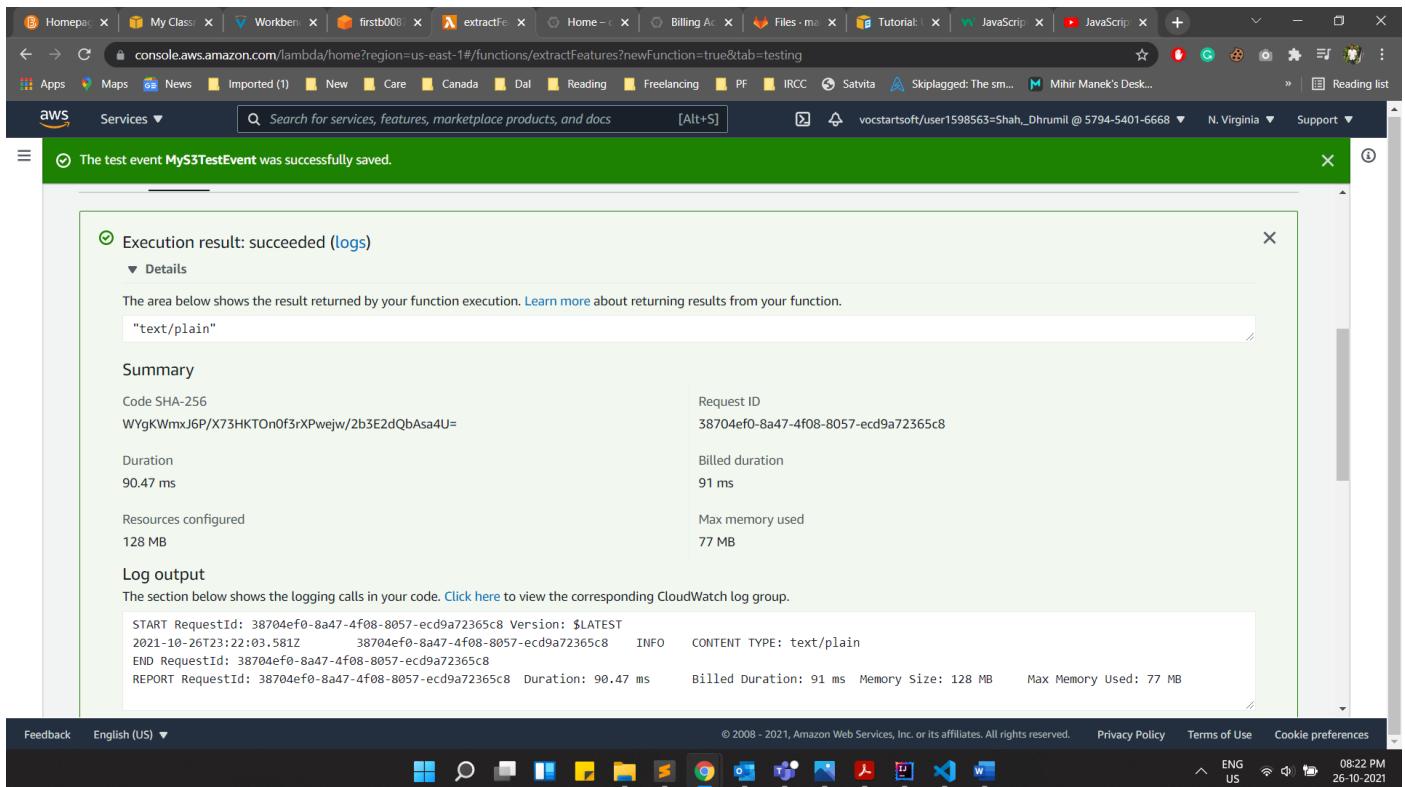


Figure 21: extractFeatures Lambda Function Configure Test Event 4

```

1 finding new homes for old phones
2
3 Re-using old mobile phones is not just good for the environment, it has social benefits too.
4
5 Research has found that in some developing nations old mobile phones can help close the digital divide. The Forum for the Future research found that the low cost of these recycled handsets means they can have a very useful second life in poorer nations. But the Forum found that more needed to be done to collect old phones rather than let them rot in landfill sites.
6
7 The report reveals that approximately 15 million mobile phones go out of use every year in the UK. Of the 15 million that are swapped for newer models each year, only 25% get returned to mobile phone firms for recycling or re-use. The slowly growing mass of unrecycled, discarded phones has now reached 90 million handsets, the equivalent of 9,000 tonnes of waste, estimates James Goodman, report author and a senior adviser at the Forum for the Future. "It's quite common for people to have two or three phones just lying around," said Mr Goodman.
8
9 Many of these older phones could end up in landfill sites leaking the potentially toxic materials they are made of into the wider world, said Mr Goodman. Far better, he said, to hand the phone back to an operator who can send it overseas where it can enjoy a second lease of life. "We've heard the environmental argument for handing a phone back," said Mr Goodman, "but there's a strong social argument too." Older mobile phones are proving particularly useful in poorer nations where people want to use a mobile and keep in touch with friends and family but do not have the income to buy the most up to date model. The Forum for the Future report took an in-depth look at Romania where reconditioned mobile phones were proving very popular. "It's an interesting country because it has a really crap fixed line network," said Mr Goodman, "and there's a real desire for people to get mobile phones." But the relatively low wages in Romania, which is one of the poorest countries in Europe, mean few people can afford a shiny new phone. "The affordability of the handsets is a real barrier to getting one," he said. Reconditioned handsets have boosted take-up of mobiles as the report revealed that almost one-third of Romanian pre-pay mobile phone users were using reconditioned handsets. The re-used handsets tend to be about one-third of the price of a new handset. Georgeta Minciuc, a Romanian part-time cleaner, said: "Normally a mobile phone would not be possible on my wages. I am a single parent - keeping in touch with my daughter is important to me." "This is the only way I can afford to have a phone," she said. Mr Goodman said phone operators and consumers needed to do more to ensure that more of Britain's mobile mountain made it overseas. But, he added, those keen to use a mobile will not accept any old handset. "If its more than a few years old people are not going to want it," he said.
10
  
```

Figure 22: Original File Content

```

1  [
2   "017ne" : [
3     { "Finding" : 1 } ,
4     { "Reusing" : 1 } ,
5     { "Research" : 1 } ,
6     { "The" : 1 } ,
7     { "Forum" : 1 } ,
8     { "Future" : 1 } ,
9     { "But" : 1 } ,
10    { "Forum" : 1 } ,
11    { "The" : 1 } ,
12    { "UK" : 1 } ,
13    { "Of" : 1 } ,
14    { "The" : 1 } ,
15    { "James" : 1 } ,
16    { "Goodman" : 1 } ,
17    { "Forum" : 1 } ,
18    { "Future" : 1 } ,
19    { "Its" : 1 } ,
20    { "Mr" : 1 } ,
21    { "Goodman" : 1 } ,
22    { "Many" : 1 } ,
23    { "Mr" : 1 } ,
24    { "Goodman" : 1 } ,
25    { "Fan" : 1 } ,
26    { "Weve" : 1 } ,
27    { "Mr" : 1 } ,
28    { "Goodman" : 1 } ,
29    { "Older" : 1 } ,
30    { "The" : 1 } ,
31    { "Forum" : 1 } ,
32    { "Future" : 1 } ,
33    { "Romania" : 1 } ,
34    { "Its" : 1 } ,
35    { "Mr" : 1 } ,
36    { "Goodman" : 1 } ,
37    { "But" : 1 } ,
38    { "Romania" : 1 } ,
39    { "Europe" : 1 } ,
40    { "The" : 1 } ,

```

Line 1. Column 1 Tab Size: 4 Plain Text ENG US 09:24 PM 29-10-2021

Figure 23: JSON File Created by Extracting Named Entities by extractFeatures Lambda Function

Figure 24: JSON File in tagsb00870600 S3 Bucket

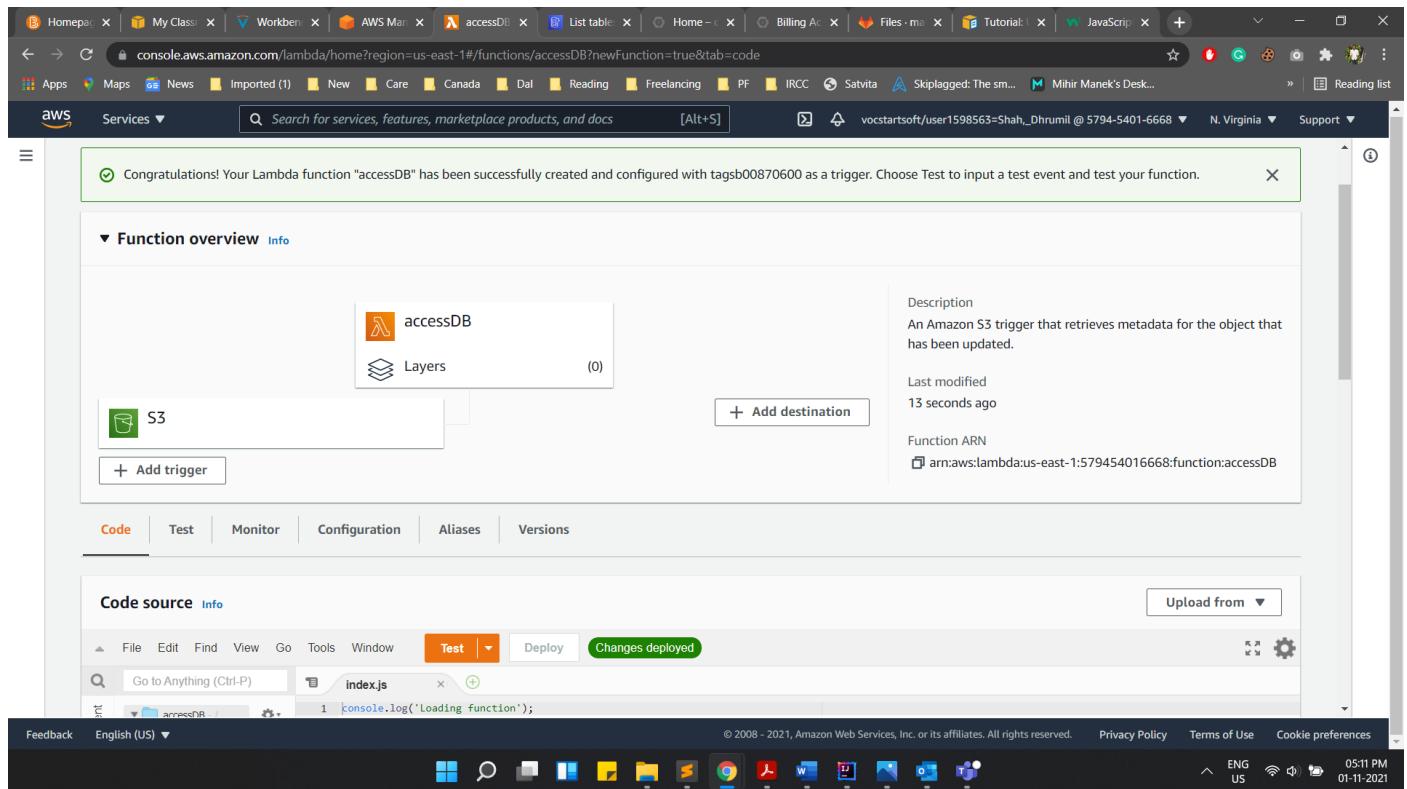


Figure 25: accessDB Lambda Function

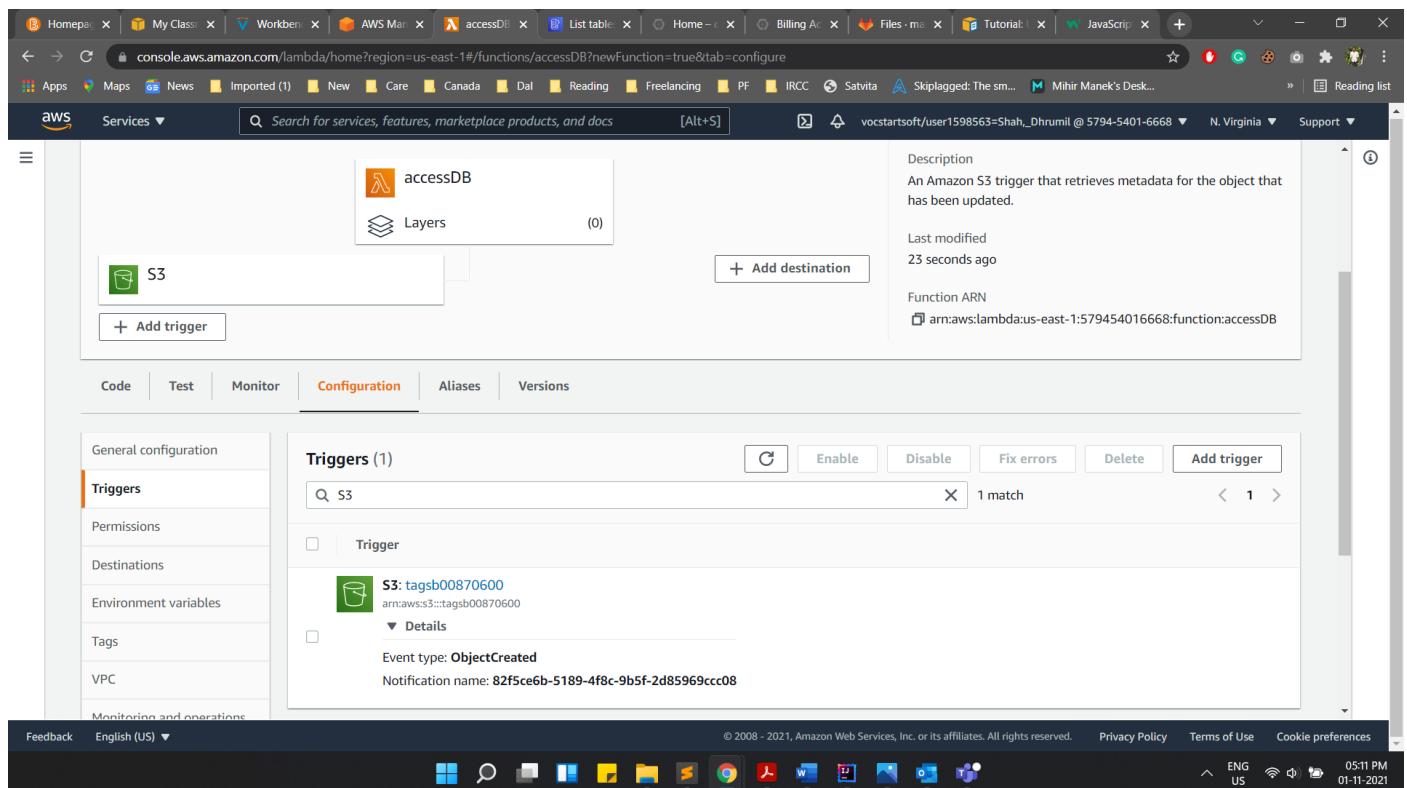
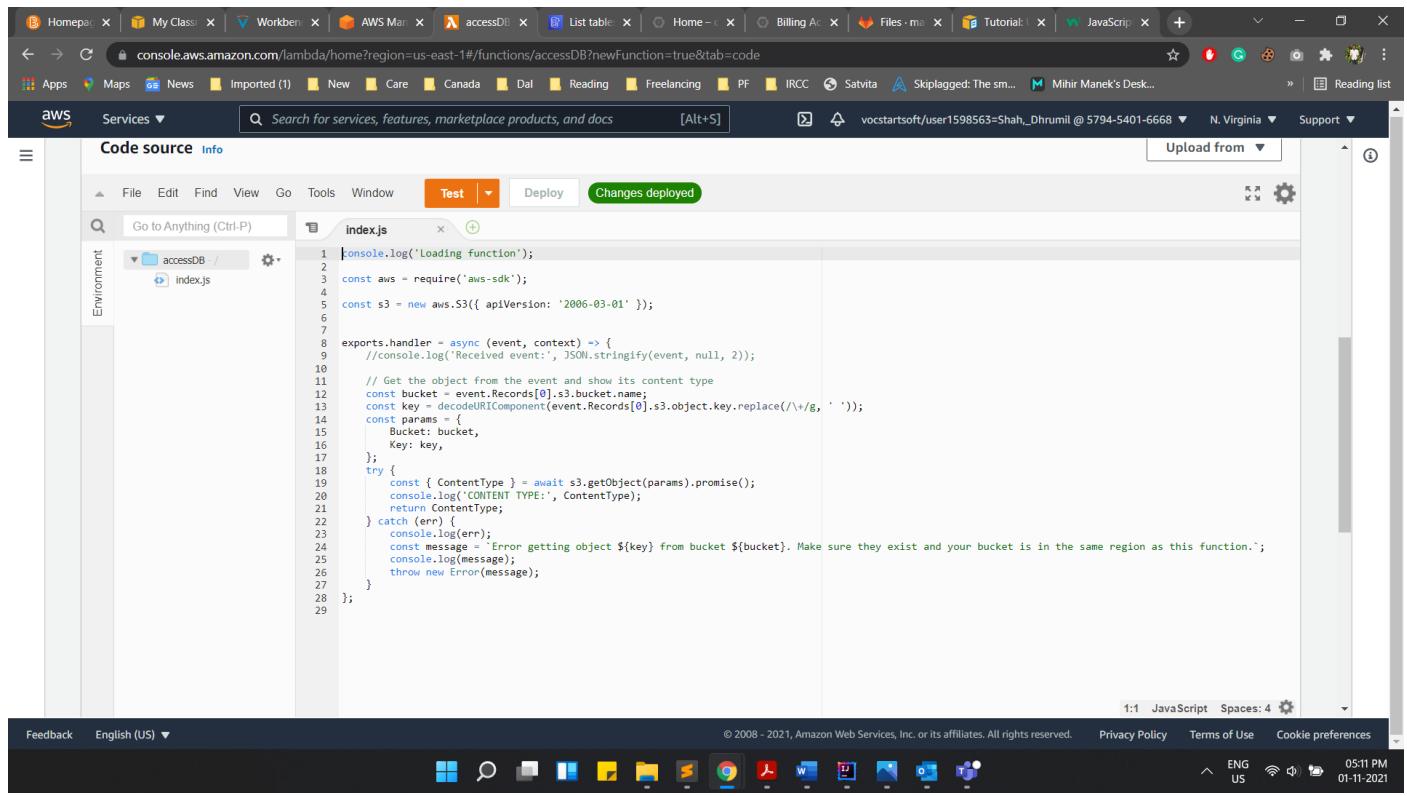


Figure 26: accessDB Lambda Function with S3 Bucket Trigger (tagsb00870600)

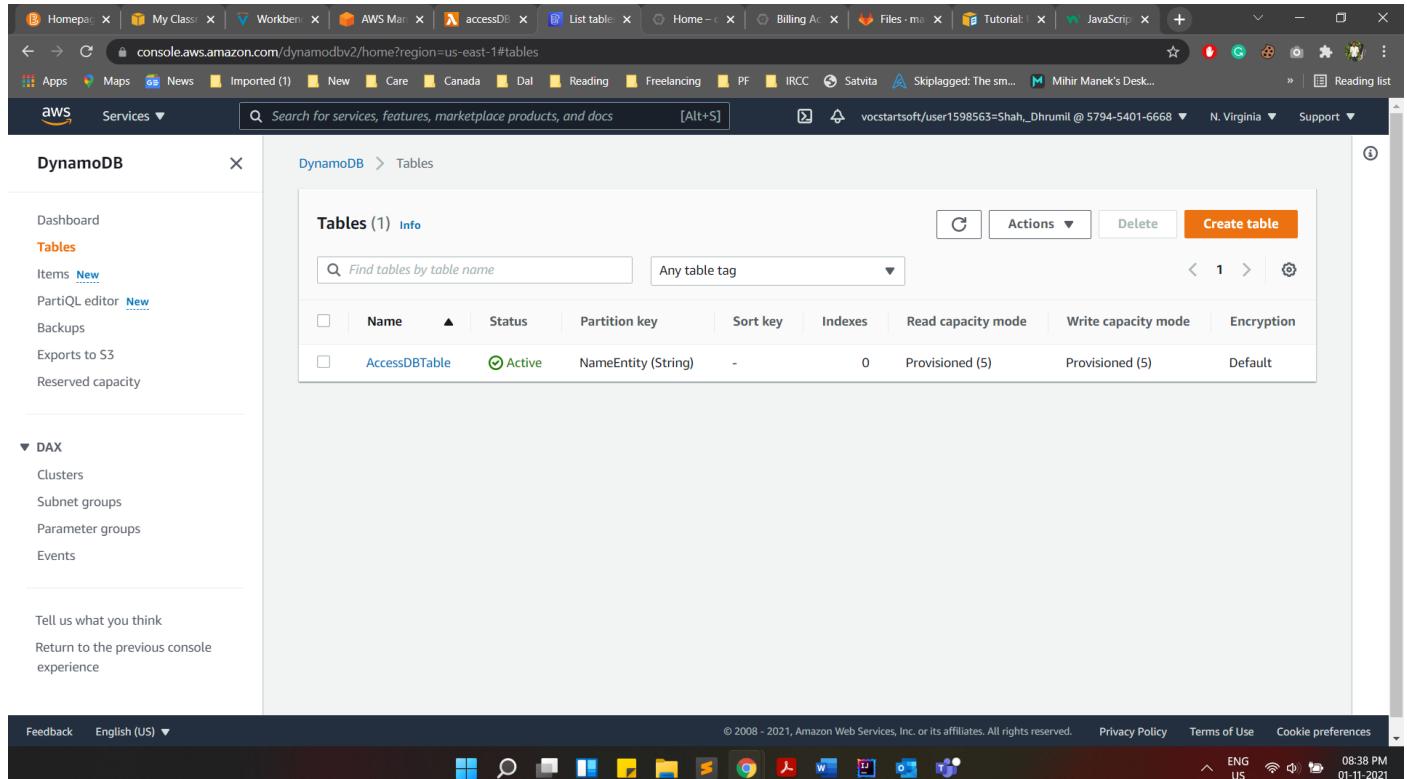


```

1  console.log('Loading function');
2
3  const aws = require('aws-sdk');
4
5  const s3 = new aws.S3({ apiVersion: '2006-03-01' });
6
7
8  exports.handler = async (event, context) => {
9      //console.log("Received event:", JSON.stringify(event, null, 2));
10
11     // Get the object from the event and show its content type
12     const bucket = event.Records[0].s3.bucket.name;
13     const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
14     const params = {
15         Bucket: bucket,
16         Key: key,
17     };
18     try {
19         const { ContentType } = await s3.getObject(params).promise();
20         console.log('CONTENT TYPE:', ContentType);
21         return ContentType;
22     } catch (err) {
23         const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
24         console.log(message);
25         throw new Error(message);
26     }
27 }
28
29

```

Figure 27:accessDB Lambda Function Boilerplate Code



	Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Encryption
<input type="checkbox"/>	AccessDBTable	Active	NameEntity (String)	-	0	Provisioned (5)	Provisioned (5)	Default

Figure 28: DynamoDB Table (AccessDBTable) I

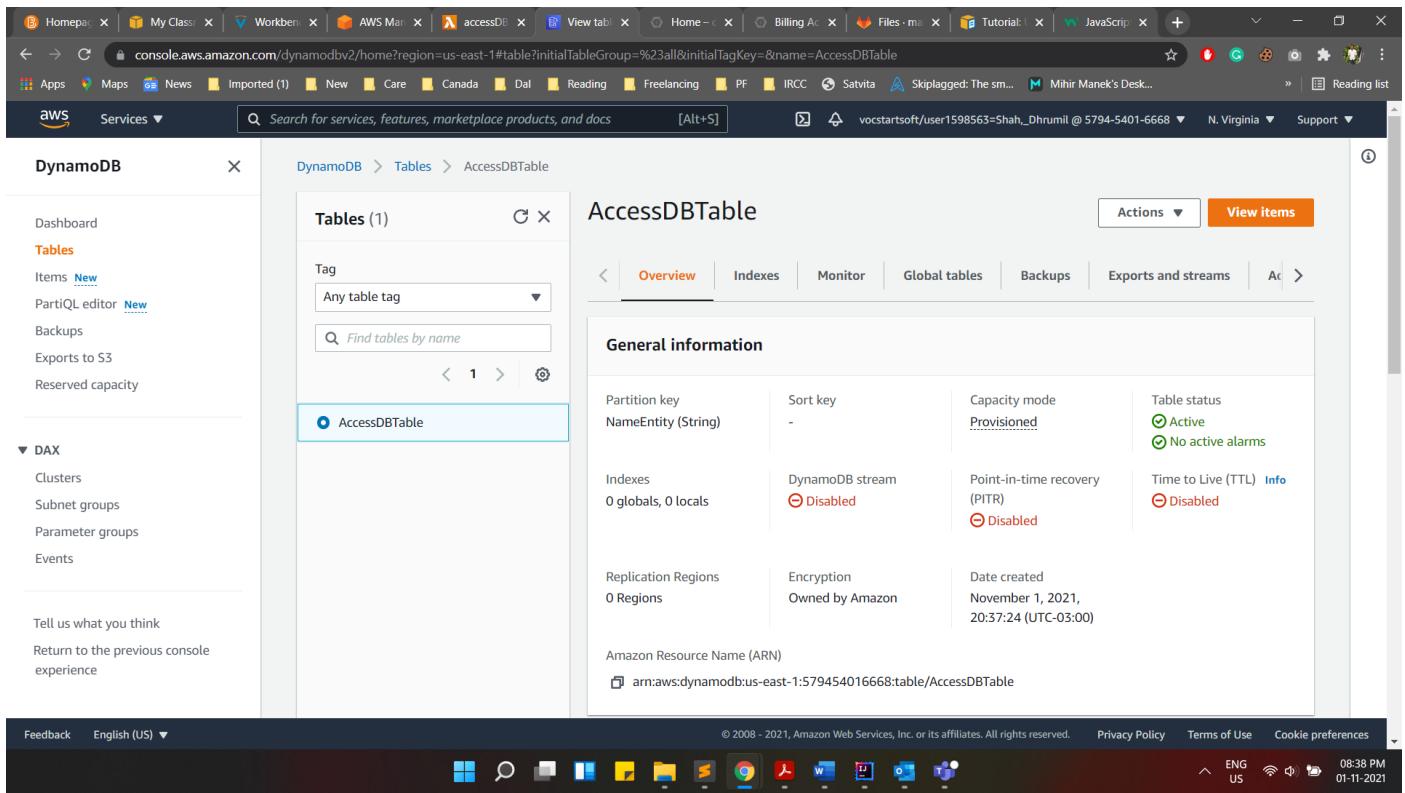


Figure 29: DynamoDB Table (AccessDBTable) 2

Code Snippets of the Part A:

Main.java:

```

import com.amazonaws.services.s3.AmazonS3;

import java.util.Scanner;

/**
 * Author: Dhrumil Rakesh Shah
 * Version: 1.0
 * Class: The Main class containing the boilerplate code of AWS SDK for Java
 * for Assignment 3
 */
public class Main {
    // The driver method
    public static void main(String[] args) {

        try {

            // Instantiating the Scanner object to read user input
            Scanner sc = new Scanner(System.in);

            // Declaring & Initializing the choice to 0 that stores the
            // choice made by the user
            int choice = 0;

            S3Task s3TaskObject = new S3Task();

            AmazonS3 s3Object = s3TaskObject.getConnection();
        }
    }
}

```

```
// Looping through the menu
while (choice != -1) {
    System.out.println("Choose any of the below tasks.");
    System.out.println("1. Create two new buckets.");
    System.out.println("2. Upload all files in tech folder to the " +
        "firstb00870600 bucket.");
    System.out.println("3. Testing the extractFeatures Lambda function.");
    System.out.println("4. Exit.");

    // Reading the user entered choice
    choice = sc.nextInt();

    // Switching through the different choices
    switch (choice) {
        // Creating a new bucket
        case 1:
            s3TaskObject.createBucket("firstb00870600", s3Object);
            s3TaskObject.createBucket("secondb00870600", s3Object);
            break;
        case 2:
            // Uploading the file to the created bucket
            s3TaskObject.fileUploadToS3Bucket(s3Object,
                "firstb00870600");
            break;
        case 3:
            // Testing the extractFeatures Lambda function
            s3TaskObject.testLambda(s3Object);
        case 4:
            // Exiting the application
            System.exit(0);
        default:
            // Default switch case
            System.out.println("Enter a valid option.");
            break;
    }
}
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}
```

S3Task.java

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;

/**
 * Author: Dhrumil Rakesh Shah
 * Version: 1.0
 * Class: The AWS SDK for Java helper class
 */
public class S3Task {

    // The method to get connection object of the AWS account
    public AmazonS3 getConnection() {
```

```

// Storing the AWS credentials to establish the connection
BasicSessionCredentials sessionCredentials = new BasicSessionCredentials(
    "ASIAYN2RGTSOPUQ5DBV",
    "4vxZQggC1hSF7Qb1DgErSuMJ0e0Ab6GfknBvjNUg",
    "FwoGZXivYXdzEIn//////////wEaDD+7t6vcfZOA5QcwKCK9AdS25Y1T" +
        "jDdeanTXWoWVa8zMqIdPeqwgbpsfSVptCU+2bi0DBGkcFv5czXVtsyg9Wv615wt" +
        "c4mOKjYGkh9Uh6bhn6qLYshVu4EtXloqrD+Ye7x08r3wUXrt2SydHJtmv1oWBcCr" +
        "9XoxY64vs9jJByICeL930Ozybwsn7hzZYeQ1weumC5zE2SL8UukXJrM/n0MVI9Pz" +
        "pPa57z1kmeOYIDOkjF82PgUxrGOUeH1Nnh73Z8Rjt5aW/sPeg9dN5MCjv8IGMBjI" +
        "tHprRFHAYt+KUKGnwPxf5JzDD12MX2BYaNDPVFrM4roZzpXDAR5gOAGSRJpGm");

// Establishing connection with the Amazon S3 service
// and storing it in the s3Object
AmazonS3 s3Object = AmazonS3ClientBuilder.standard().withCredentials(
    new AWSStaticCredentialsProvider(sessionCredentials))
    .withRegion(Regions.US_EAST_1)
    .build();

// Returning the s3Object
return s3Object;
}

// The public method to create a new S3 bucket
public void createBucket(String bucketName, AmazonS3 s3Object) {
    try {
        // Creating a new S3 bucket using the s3Object
        s3Object.createBucket(bucketName);

        // Printing to the console
        System.out.format("A new bucket '%s' is created.\n", bucketName);

        // Catching the exception
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}

// The public method to upload multiple files to a S3 bucket
public void fileUploadToS3Bucket(AmazonS3 s3Object,
                                  String bucketName) {
    try {
        // Looping through the files in tech folder
        for (int i = 1; i <= 401; i++) {

            // Declaring the filePrefix and fileName
            String filePrefix = "";
            String fileName = "";

            // Conditioning through the files in tech folder
            if (i < 10) {
                filePrefix = "00" + i;
                fileName = "tech/00" + i + ".txt";
            } else if (i > 9 && i < 100) {
                filePrefix = "0" + i;
                fileName = "tech/0" + i + ".txt";
            } else {
                filePrefix = "" + i;
                fileName = "tech/" + i + ".txt";
            }

            // Instantiating the PutObjectRequest class
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,

```

```

        filePrefix, new File(fileName));

        // Uploading the file to the S3 bucket
        s3Object.putObject(putObjectRequest);

        // Printing to the console
        System.out.format("The file '%s.txt' is pushed to the '%s' bucket \n",
            filePrefix, bucketName);

        // Adding a delay of 200 milliseconds after adding a new file to the bucket
        Thread.sleep(200);

        // Printing that 200 milliseconds sleep is added
        System.out.println("Added delay of 200 millis");
    }
    // Catching the exception
} catch (Exception e) {
    System.err.println(e.getMessage());
}
}

// Method to test the working of extractFeatures Lambda function
public void testLambda(AmazonS3 s3Client) {
    String bucketName = "firstb00870600";
    String filePrefix = "001";
    String fileName = "tech/001.txt";
    try {
        PutObjectRequest request = new PutObjectRequest(bucketName, filePrefix, new
File(fileName));
        s3Client.putObject(request);

        // Printing to the console
        System.out.format("The file '%s.txt' is pushed to the '%s' bucket \n",
            filePrefix, bucketName);
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}
}

```

extractFeatures.js

```

console.log('Loading function');

const aws = require('aws-sdk');

const s3 = new aws.S3({ apiVersion: '2006-03-01' });

exports.handler = async (event, context) => {
    //console.log('Received event:', JSON.stringify(event, null, 2));

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
    const params = {
        Bucket: bucket,

```

```

    Key: key,
};

console.log(params);
extractFeatures(params);
try {
  const { ContentType } = await s3.getObject(params).promise();
  console.log('CONTENT TYPE:', ContentType);
  return ContentType;
} catch (err) {
  console.log(err);
  const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your
bucket is in the same region as this function.`;
  console.log(message);
  throw new Error(message);
}
};

const extractFeatures = (params) => {

let uploadParams = {
  Bucket: 'tagsb00870600',
  Key: ""
};

let wordCount = '{ \n "' + params.Key + 'ne" : [ ';

s3.getObject(params, function (err, metadata) {
  if (err && err.code === 'Not found') {
    console.log("Object not found");
  } else {
    let filedata = metadata.Body.toString('utf-8');
    filedata = filedata.split("\n");
    for (let i = 0; i < filedata.length; i++) {

      let eachwords = filedata[i].split(' ');
      for (let k = 0; k < eachwords.length; k++) {
        if (eachwords[k] === "") {
          continue;
        }
        eachwords[k] = eachwords[k].replace(/[^a-zA-Z ]/g, "");
        if (eachwords[k].charAt(0) === eachwords[k].charAt(0).toUpperCase()) {
          if (eachwords[k] === "") {
            continue;
          }
          let wordformat = "\n\t{ ";
          wordformat = wordformat + "" + eachwords[k] + "" : 1 } ,";
          wordCount = wordCount + wordformat;
        }
      }
    }
  }
  wordCount = wordCount.slice(0, -1);
}

```

```

wordCount = wordCount + " \n] }";
let filename = params.Key + "ne";
uploadParams.Key = filename;
uploadParams.Body = wordCount;
s3.upload(uploadParams, function (s3Err, data) {
  if (s3Err) {
    console.log(s3Err);
  }
  console.log(`File uploaded successfully at ${data.Location}`);
});
console.log(wordCount);
let countjsonarray = JSON.parse(wordCount);
}
});
};

```

accessDB.js

```

console.log('Loading function');

const aws = require('aws-sdk');

const s3 = new aws.S3({ apiVersion: '2006-03-01' });

var docClient = new aws.DynamoDB.DocumentClient();

const ddb = new aws.DynamoDB({ apiVersion: '2012-08-10' });

exports.handler = async (event, context) => {
  //console.log('Received event:', JSON.stringify(event, null, 2));

  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  let filedata;
  s3.getObject(params, function (err, metadata) {
    if (err && err.code === 'Not found') {
      console.log("Object not found");
    } else {
      filedata = metadata.Body.toString('utf-8');
    }
  })
  accessDB(filedata, params.Key);
  try {

```

```

const { ContentType } = await s3.getObject(params).promise();
console.log('CONTENT TYPE:', ContentType);
return ContentType;
} catch (err) {
  console.log(err);
  const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your
bucket is in the same region as this function.`;
  console.log(message);
  throw new Error(message);
}
};


```

```

const accessDB = async (wordsarray, key1) => {

  console.log(wordsarray[key1]);
  for (let a = 0; a < wordsarray[key1].length; a++) {
    for (var key in wordsarray[key1][a]) {
      docClient.put(nameEntityCol, function(err, data) {
        if (err) {
          console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
        } else {
          console.log("Added item:", JSON.stringify(data, null, 2));
        }
      });
      docClient.put(frequencyCol, function(err, data) {
        if (err) {
          console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
        } else {
          console.log("Added item:", JSON.stringify(data, null, 2));
        }
      });
    }
    docClient.put(timestampOfEntryCol, new AttributeValue(new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSSSSS")
      .format(new Date()))), function(err, data) {
      if (err) {
        console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
      } else {
        console.log("Added item:", JSON.stringify(data, null, 2));
      }
    });
  }
};


```

References:

1. <https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html>
2. https://www.w3schools.com/js/js_json.asp
3. <https://aws.amazon.com/education/awseducate/>