PART – 1 REPORT

Real-time Underwater Exploration and Monitoring

In this use case, an innovative project called "MarineGuard" is dedicated to the real-time monitoring and conservation of marine ecosystems. MarineGuard aims to deploy a network of advanced sensors and devices in various marine locations to collect crucial environmental data, ensuring the protection and sustainable management of ocean resources.

MarineGuard plants a range of underwater sensors, including underwater cameras, hydrophones, and water quality probes, distributed strategically across different marine environments across the globe. These sensors continuously monitor and gather real-time data on water quality, marine species behavior, and environmental conditions. The underwater sensor data will be securely transmitted to AWS IoT Core. Collected data is seamlessly transmitted as records to AWS Kinesis Data Streams, which acts as the central data gathering service.

To optimize data processing and analysis, AWS Kinesis Data Streams partition the incoming data into multiple shards based on unique partition keys derived from identifiers associated with each marine location or specific aspects of marine life. This ensures even data distribution and streamlines subsequent analysis and decision-making processes for each targeted area or marine species. The partitioned data undergoes parallel processing and efficient distribution, maximizing the value of the collected information.

The streaming data is further processed and automatically transformed by AWS Kinesis Data Firehose, a service that prepares the data before delivery. It handles essential data transformations such as compression, encoding, and conversion to various formats. AWS Kinesis Data Analytics plays a pivotal role in MarineGuard's real-time data analysis. It ingests data from AWS Kinesis Data Firehose enabling comprehensive and in-depth analytics. The prepared data is stored in Amazon S3 for long-term archival, facilitating historical analysis and data retention. Additionally, the data can be seamlessly transferred to Amazon Redshift for data warehousing and advanced analytics.

Following the data analytics process, AWS QuickSight steps in, providing an interactive and user-friendly platform for data visualization and business intelligence. By creating interactive dashboards, charts, graphs, and maps from the processed data, AWS QuickSight empowers our team to communicate findings, identify trends, and make informed decisions. The seamless integration of AWS QuickSight into the data pipeline enhances our ability to optimize marine conservation efforts, safeguard marine ecosystems, and promote sustainable management practices globally.

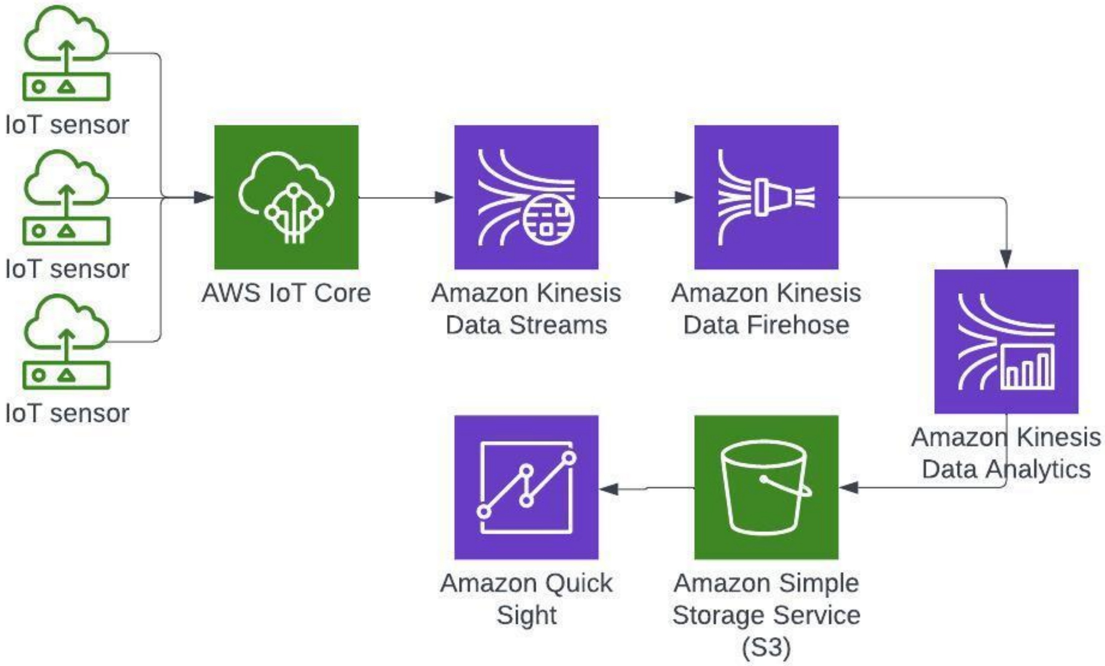
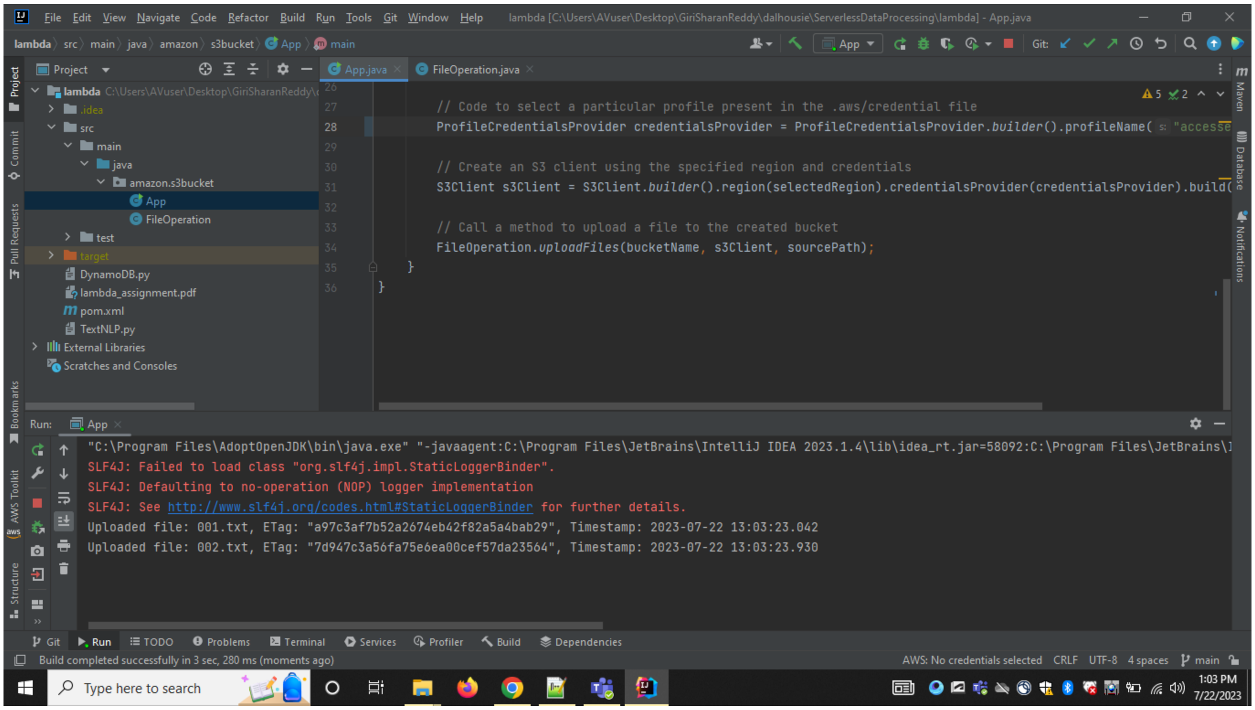

IoT sensor
IoT sensor
IoT sensor
AWS IoT Core
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon Kinesis Data Analytics
Amazon Simple Storage Service (S3)
Amazon Quick Sight

Figure 1:Architecture for developing ocean data analysis using amazon cloud services [12]

PART – 2 REPORT

Github Link : https://git.cs.dal.ca/pusuluru/csci-5410-summer-23-b00913674.git

1. My application uploading files to S3 bucket (sampledata-b00913674) for every 100 milli-seconds



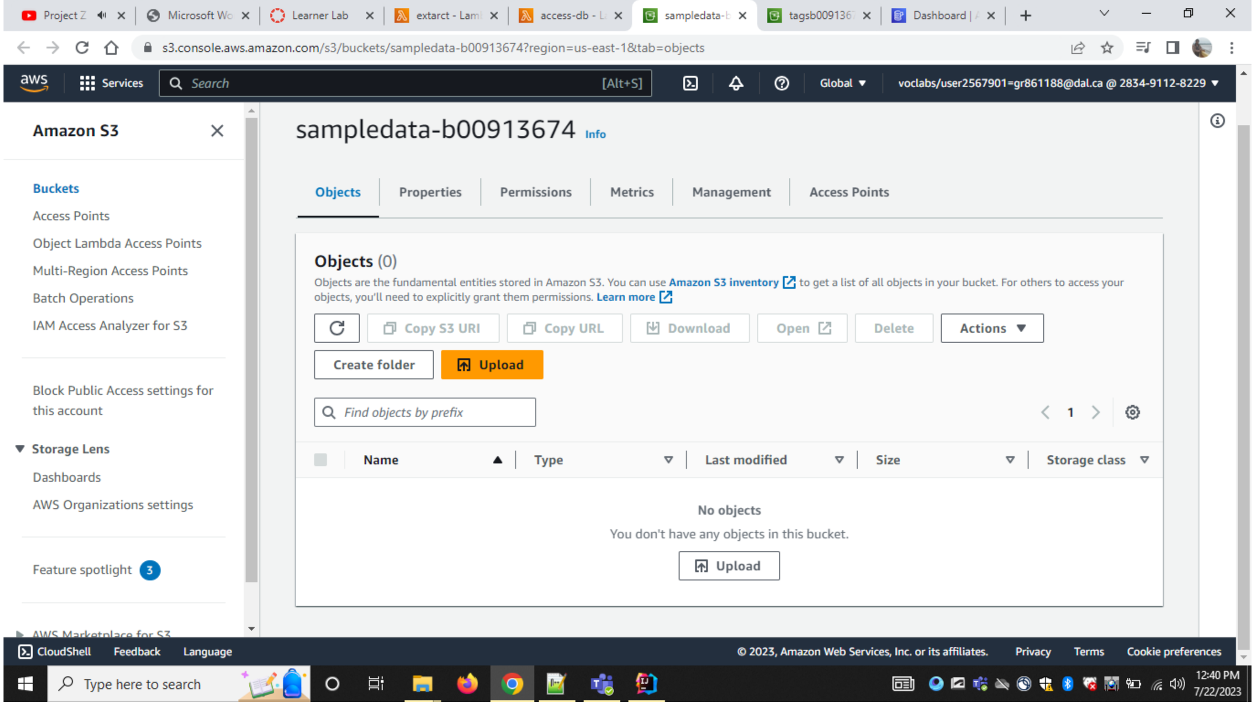
```
        // Code to select a particular profile present in the .aws/credential file
        ProfileCredentialsProvider credentialsProvider = ProfileCredentialsProvider.builder().profileName( s: "accesse

        // Create an S3 client using the specified region and credentials
        S3Client s3Client = S3Client.builder().region(selectedRegion).credentialsProvider(credentialsProvider).build(

        // Call a method to upload a file to the created bucket
        FileOperation.uploadFiles(bucketName, s3Client, sourcePath);
    }
}
```

```
"C:\Program Files\AdoptOpenJDK\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1.4\lib\idea_rt.jar=58092:C:\Program Files\JetBrains\I
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Uploaded file: 001.txt, ETag: "a97c3af7b52a2674eb42f82a5a4bab29", Timestamp: 2023-07-22 13:03:23.042
Uploaded file: 002.txt, ETag: "7d947c3a56fa75e6ea00cef57da23564", Timestamp: 2023-07-22 13:03:23.930
```

Build completed successfully in 3 sec, 280 ms (moments ago)

Figure 2:Application uploading files to s3 bucket

2. Initially there were no files in my bucket (sampledata-b00913674)



s3.console.aws.amazon.com/s3/buckets/sampledata-b00913674?region=us-east-1&tab=objects

voclabs/user2567901=gr861188@dal.ca @ 2834-9112-8229

sampledata-b00913674

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

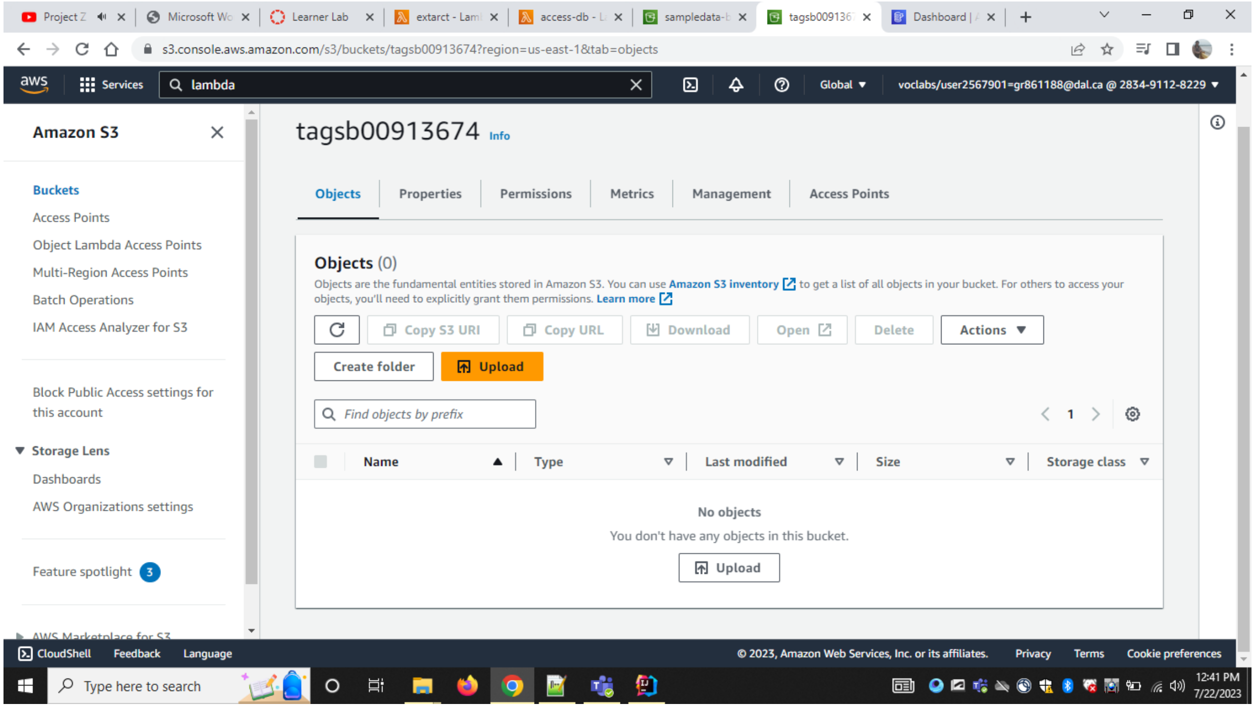
Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.

Figure 3:No objects present in s3 bucket initially

3. Initially there were no files in S3 Bucket (tagsb00913674)



s3.console.aws.amazon.com/s3/buckets/tagsb00913674?region=us-east-1&tab=objects

tagsb00913674 Info

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

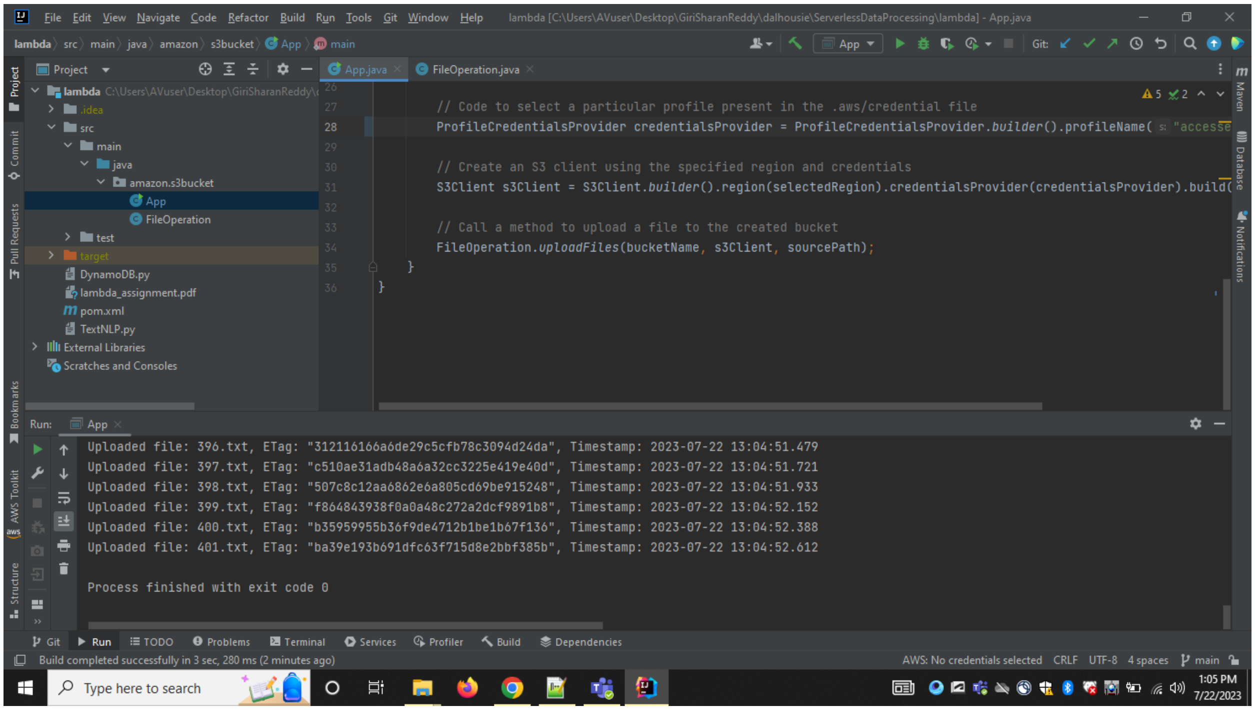
Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.

Figure 4:No objects present in s3 bucket initially

4. My application code successfully uploaded 401 files to S3 bucket (sampledata-b00913674)



lambda [C:\Users\AVuser\Desktop\GiriSharanReddy\dalhousie\ServerlessDataProcessing\lambda] - App.java

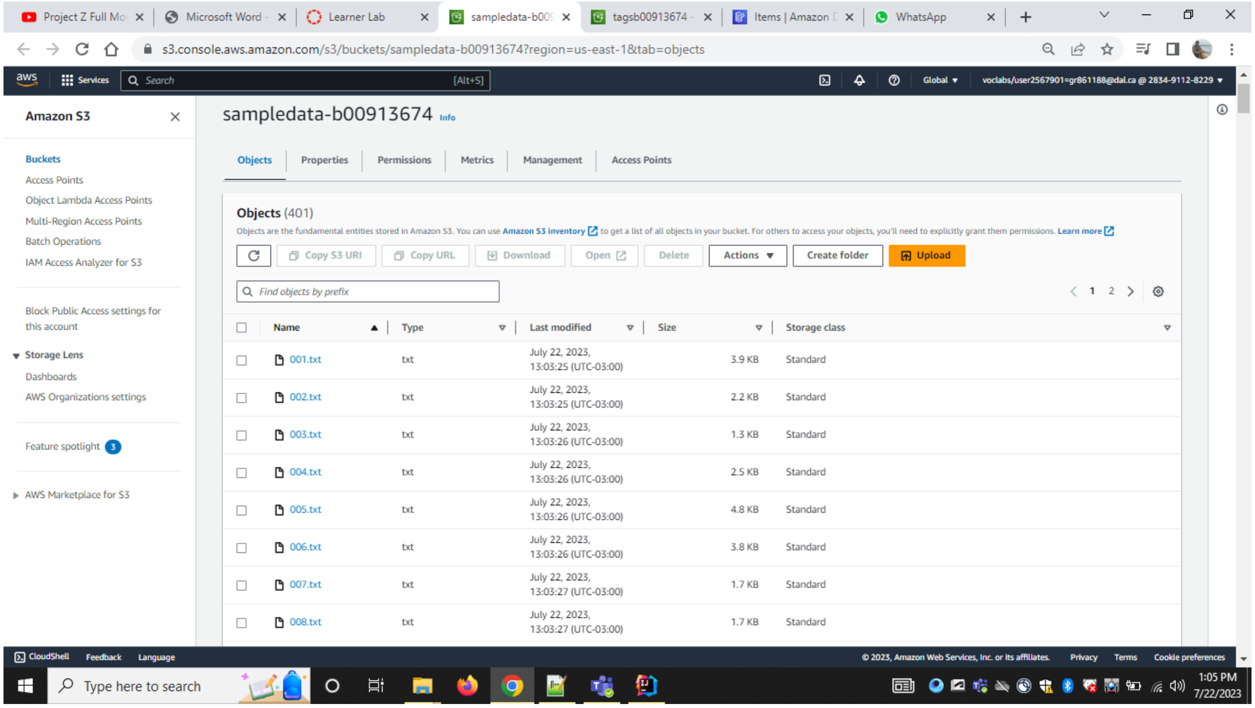
```
        // Code to select a particular profile present in the .aws/credential file
        ProfileCredentialsProvider credentialsProvider = ProfileCredentialsProvider.builder().profileName( s: "accesse

        // Create an S3 client using the specified region and credentials
        S3Client s3Client = S3Client.builder().region(selectedRegion).credentialsProvider(credentialsProvider).build(

        // Call a method to upload a file to the created bucket
        FileOperation.uploadFiles(bucketName, s3Client, sourcePath);
    }
}
```

```
Uploaded file: 396.txt, ETag: "312116166a6de29c5cfb78c3094d24da", Timestamp: 2023-07-22 13:04:51.479
Uploaded file: 397.txt, ETag: "c510ae31adb48a6a32cc3225e419e40d", Timestamp: 2023-07-22 13:04:51.721
Uploaded file: 398.txt, ETag: "507c8c12aa6862e6a805cd69be915248", Timestamp: 2023-07-22 13:04:51.933
Uploaded file: 399.txt, ETag: "f864843938f0a0a48c272a2dcf9891b8", Timestamp: 2023-07-22 13:04:52.152
Uploaded file: 400.txt, ETag: "b35959955b36f9de4712b1be1b67f136", Timestamp: 2023-07-22 13:04:52.388
Uploaded file: 401.txt, ETag: "ba39e193b691dfc63f715d8e2bbf385b", Timestamp: 2023-07-22 13:04:52.612

Process finished with exit code 0
```

Build completed successfully in 3 sec, 280 ms (2 minutes ago)

Figure 5:Application successfully uploaded files to s3 bucket

5. Total 401 files uploaded to sampledata-b00913674 S3 bucket



sampledata-b00913674 Info

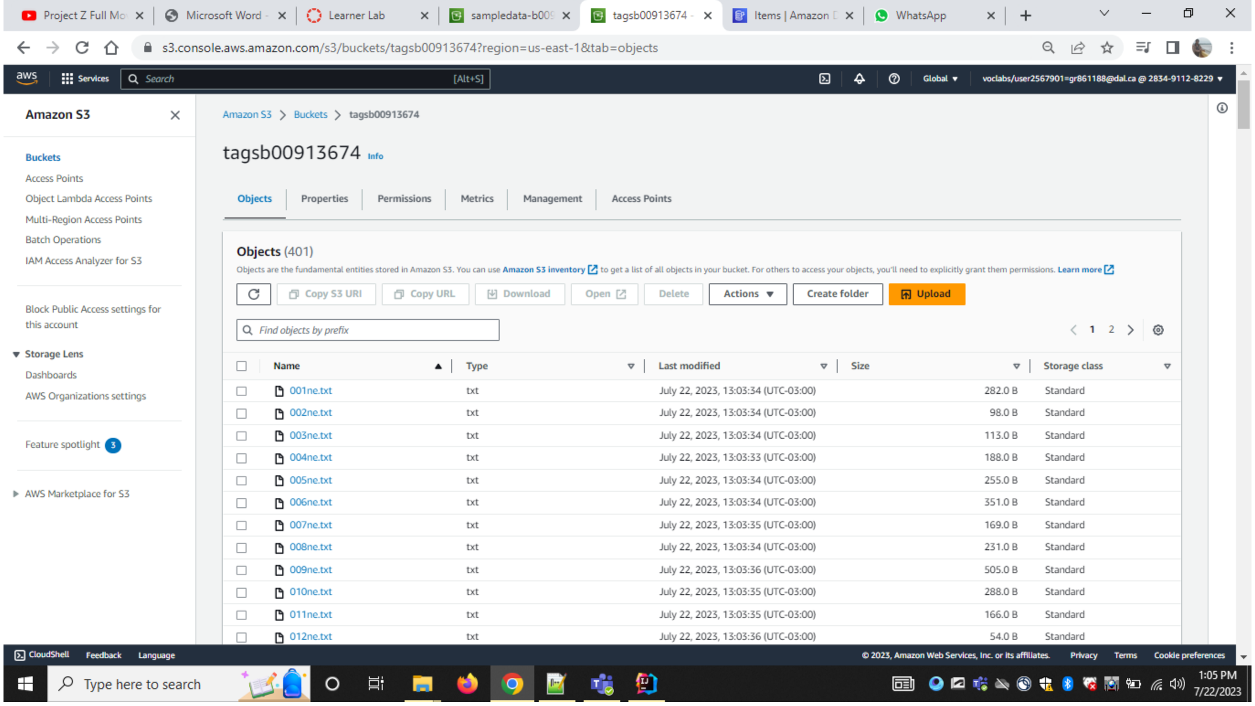
Objects (401)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	001.txt	txt	July 22, 2023, 13:03:25 (UTC-03:00)	3.9 KB	Standard
□	002.txt	txt	July 22, 2023, 13:03:25 (UTC-03:00)	2.2 KB	Standard
□	003.txt	txt	July 22, 2023, 13:03:26 (UTC-03:00)	1.3 KB	Standard
□	004.txt	txt	July 22, 2023, 13:03:26 (UTC-03:00)	2.5 KB	Standard
□	005.txt	txt	July 22, 2023, 13:03:26 (UTC-03:00)	4.8 KB	Standard
□	006.txt	txt	July 22, 2023, 13:03:26 (UTC-03:00)	3.8 KB	Standard
□	007.txt	txt	July 22, 2023, 13:03:27 (UTC-03:00)	1.7 KB	Standard
□	008.txt	txt	July 22, 2023, 13:03:27 (UTC-03:00)	1.7 KB	Standard

Figure 6:401 objects present in s3 bucket

6. 401 new files created by “extarct” lambda function and uploaded to tagsb00913674 S3 bucket



tagsb00913674 Info

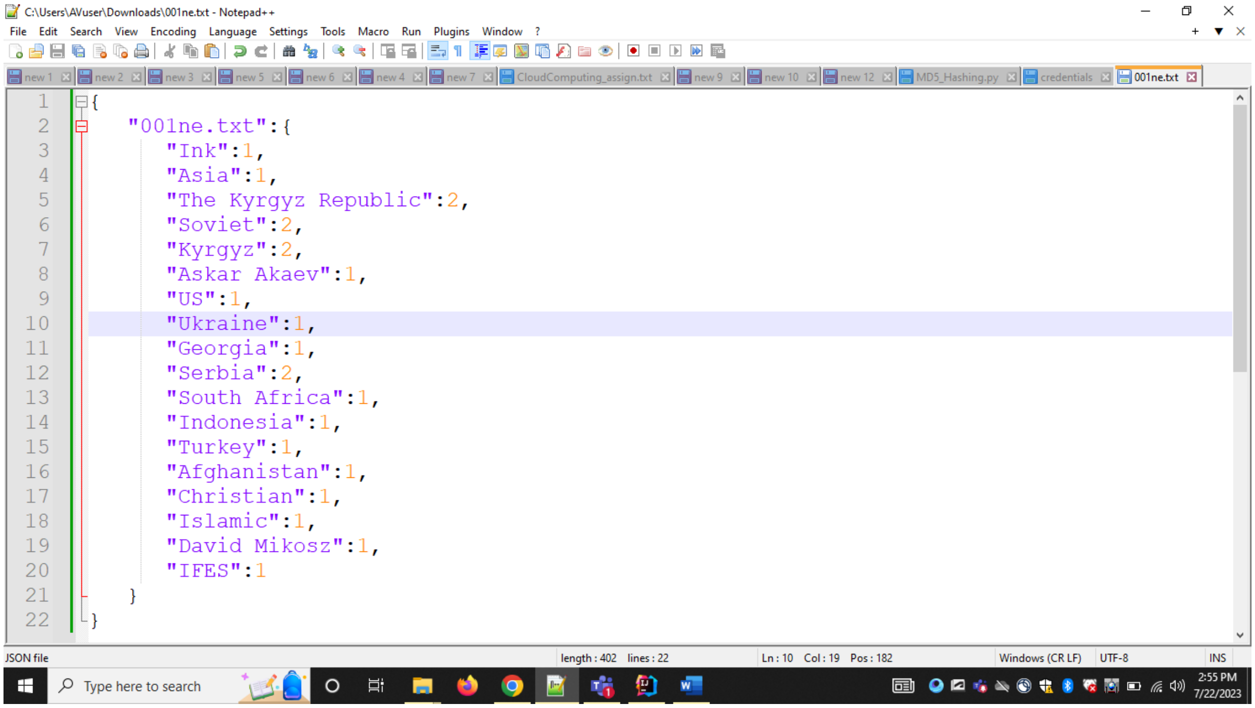
Objects (401)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	001ne.txt	txt	July 22, 2023, 13:03:34 (UTC-03:00)	282.0 B	Standard
□	002ne.txt	txt	July 22, 2023, 13:03:34 (UTC-03:00)	98.0 B	Standard
□	003ne.txt	txt	July 22, 2023, 13:03:34 (UTC-03:00)	113.0 B	Standard
□	004ne.txt	txt	July 22, 2023, 13:03:33 (UTC-03:00)	188.0 B	Standard
□	005ne.txt	txt	July 22, 2023, 13:03:34 (UTC-03:00)	255.0 B	Standard
□	006ne.txt	txt	July 22, 2023, 13:03:34 (UTC-03:00)	351.0 B	Standard
□	007ne.txt	txt	July 22, 2023, 13:03:35 (UTC-03:00)	169.0 B	Standard
□	008ne.txt	txt	July 22, 2023, 13:03:34 (UTC-03:00)	231.0 B	Standard
□	009ne.txt	txt	July 22, 2023, 13:03:36 (UTC-03:00)	505.0 B	Standard
□	010ne.txt	txt	July 22, 2023, 13:03:35 (UTC-03:00)	288.0 B	Standard
□	011ne.txt	txt	July 22, 2023, 13:03:35 (UTC-03:00)	166.0 B	Standard
□	012ne.txt	txt	July 22, 2023, 13:03:36 (UTC-03:00)	54.0 B	Standard

Figure 7:401 new objects present in s3 bucket

7. Data present in newly created files (e.g., data present in 001ne.txt)



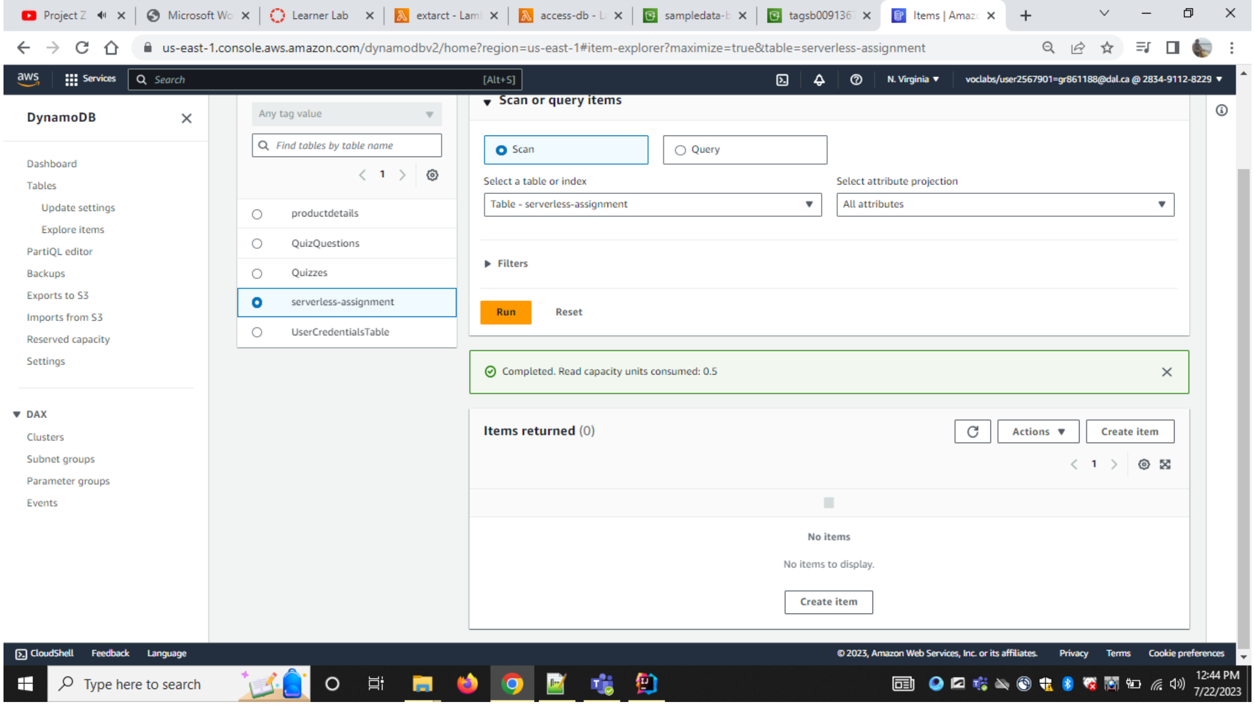
C:\Users\AVuser\Downloads\001ne.txt - Notepad++

```
{
    "001ne.txt":{
        "Ink":1,
        "Asia":1,
        "The Kyrgyz Republic":2,
        "Soviet":2,
        "Kyrgyz":2,
        "Askar Akaev":1,
        "US":1,
        "Ukraine":1,
        "Georgia":1,
        "Serbia":2,
        "South Africa":1,
        "Indonesia":1,
        "Turkey":1,
        "Afghanistan":1,
        "Christian":1,
        "Islamic":1,
        "David Mikosz":1,
        "IFES":1
    }
}
```

JSON file length : 402 lines : 22 Ln : 10 Col : 19 Pos : 182 Windows (CR LF) UTF-8 INS

Figure 8:Sample data present in new objects created

8. Initally there are no items in Dynamodb (serverless-assignment)



us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#item-explorer?maximize=true&table=serverless-assignment

N. Virginia ▾ voclabs/user2567901=gr861188@dal.ca @ 2834-9112-8229

Scan or query items

● Scan ○ Query

Select a table or index: Table - serverless-assignment

Select attribute projection: All attributes

Completed. Read capacity units consumed: 0.5

Items returned (0)

No items

No items to display.

Figure 9:No items present in dynamodb

Extarct Lambda code to extract Important entities like location, Nouns and in this code, I have removed some unwanted entities like percentages, time, date formats and moreover I removed the entities which are not starting with capital letters even though they belong to entities like location, nouns. Once the important entities are fetched from the file, then a new file is created with name like 001ne.txt and kept the entities in that file as json format

```
/*
[1] “SpaCy · industrial-strength Natural Language Processing in Python,”
Spacy.io. [Online]. Available: https://github.com/minway/NlpTest. [Accessed:
22-Jul-2023].
[2] NlpTest: Test CoreNLP on AWS Lambda. Available: https://spacy.io/.
[Accessed: 22-Jul-2023].
 */

import json
import spacy
import boto3
from collections import defaultdict

def lambda_handler(event, context):

nlp = spacy.load('/opt/en_core_web_sm-2.1.0')

# Get the S3 bucket name and object key from the event
bucket_name = event['Records'][0]['s3']['bucket']['name']
object_key = event['Records'][0]['s3']['object']['key']

print('Bucket Name: {} - Object Name: {}'.format(bucket_name, object_key))

# Create an S3 client
s3_client = boto3.client('s3')

# Read the content of the object
response = s3_client.get_object(Bucket=bucket_name, Key=object_key)
content = response['Body'].read().decode('utf-8')

word_frequencies = {}
my_set = {'apple'}  # Existing set

nlp_test = nlp(content)

doc = nlp(content)

# for entity in doc.ents:
# print(entity.text, entity.label_)


unwanted_entity_types = ["CARDINAL", "PERCENT", "QUANTITY", "TIME", "DATE",
"ORDINAL", "MONEY"]
for entity in nlp_test.ents:
if entity.label_ in unwanted_entity_types:
my_set.add(str(entity))
continue
# checking if first letter is capital or not
if entity.text[0].isupper() or entity.text.isupper():
if str(entity) in word_frequencies:
```

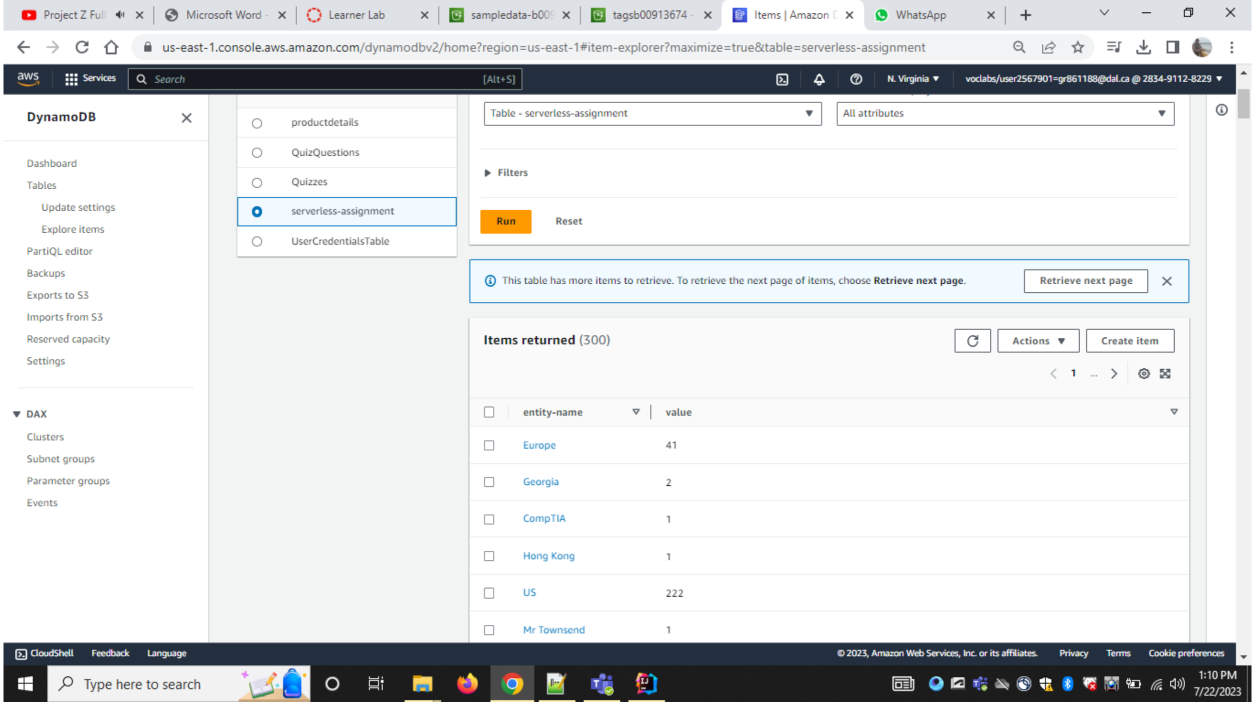
```
word_frequencies[str(entity)] += 1
else:
word_frequencies[str(entity)] = 1

print(word_frequencies)
print(my_set)

newFileName = object_key.split(".")[0] + 'ne.txt'


# Prepare data in the required format
data = {newFileName: word_frequencies}


file_path = f"/tmp/{newFileName}"  # Use /tmp directory for writable storage

with open(file_path, "w") as f:
json.dump(data, f)

new_bucket_name = 'tagsb00913674'

try:
response = s3_client.upload_file(file_path, new_bucket_name, newFileName)
return {
'statusCode': 200
}
except ClientError as e:
logging.error(e)
```

Accessdb lambda code, this code will extract the keywords from the files stored in tagsb00913674 bucket and keep in dynamodb along with their repeatable frequency count

```
/*
Amazon.com. [Online]. Available: https://docs.aws.amazon.com/code-library/latest/ug/python_3_dynamodb_code_examples.html. [Accessed: 22-Jul-2023].
 */

        import json
        import boto3

        def lambda_handler(event, context):
        # Get the S3 bucket name and object key from the event
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']

        # Create an S3 client
        s3_client = boto3.client('s3')

        # Read the content of the object
        response = s3_client.get_object(Bucket=bucket_name, Key=object_key)
        content = response['Body'].read().decode('utf-8')
```

```
# Parse the content and extract the required data
content_data = json.loads(content)[object_key]

# Create or get the DynamoDB resource
dynamodb_resource = boto3.resource('dynamodb')
table_name = 'serverless-assignment'
table = dynamodb_resource.Table(table_name)

# Update the DynamoDB table with the content data
for key, value in content_data.items():
response = table.update_item(
Key={'entity-name': key},
UpdateExpression='ADD #attrName :value',
ExpressionAttributeNames={'#attrName': 'value'},
ExpressionAttributeValues={':value': value},
ReturnValues='UPDATED_NEW'
)

print("Data written to DynamoDB table:", table_name)
```

9. Processed data of files from tagsb00913674 bucket, stored in dynamodb along with their count



○ productdetails
○ QuizQuestions
○ Quizzes
● serverless-assignment
○ UserCredentialsTable

Table - serverless-assignment ▼ | All attributes ▼

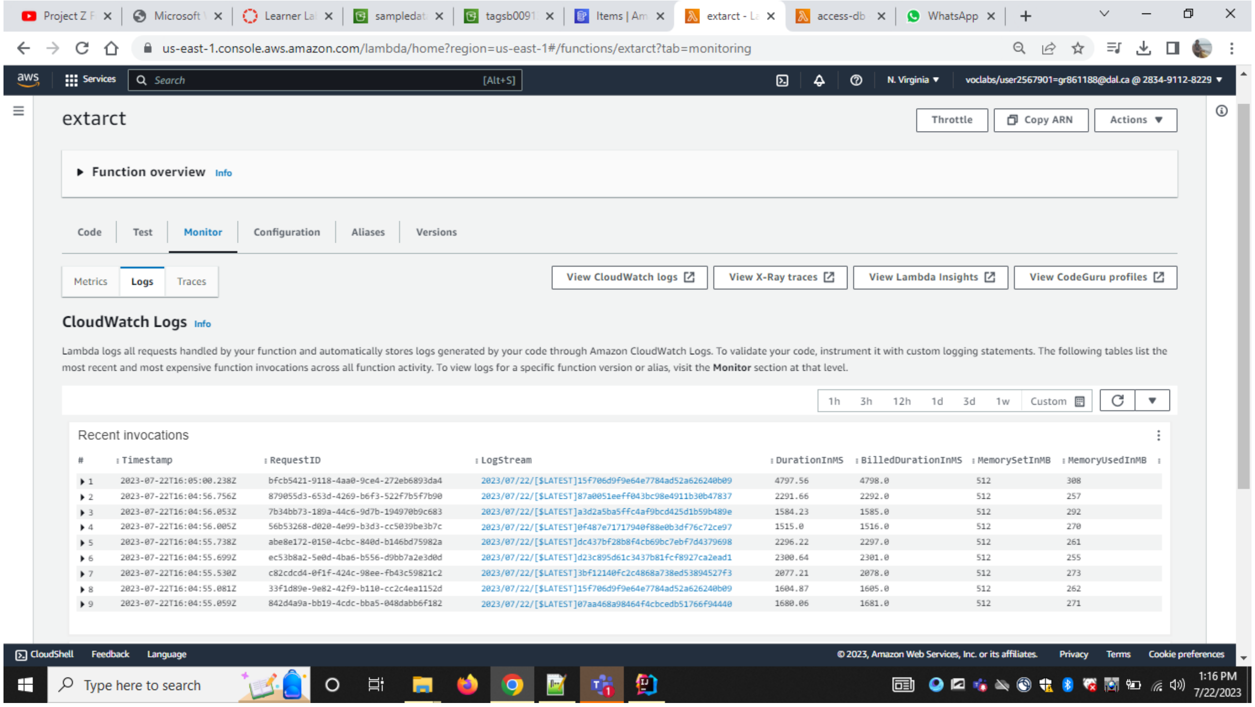
This table has more items to retrieve. To retrieve the next page of items, choose **Retrieve next page**.

Items returned (300)

□	entity-name	value
□	Europe	41
□	Georgia	2
□	CompTIA	1
□	Hong Kong	1
□	US	222
□	Mr Townsend	1

Figure 10:Items and their count in dynamodb

10. Extarct Lambda execution Logs



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/extarct?tab=monitoring

extarct

CloudWatch Logs

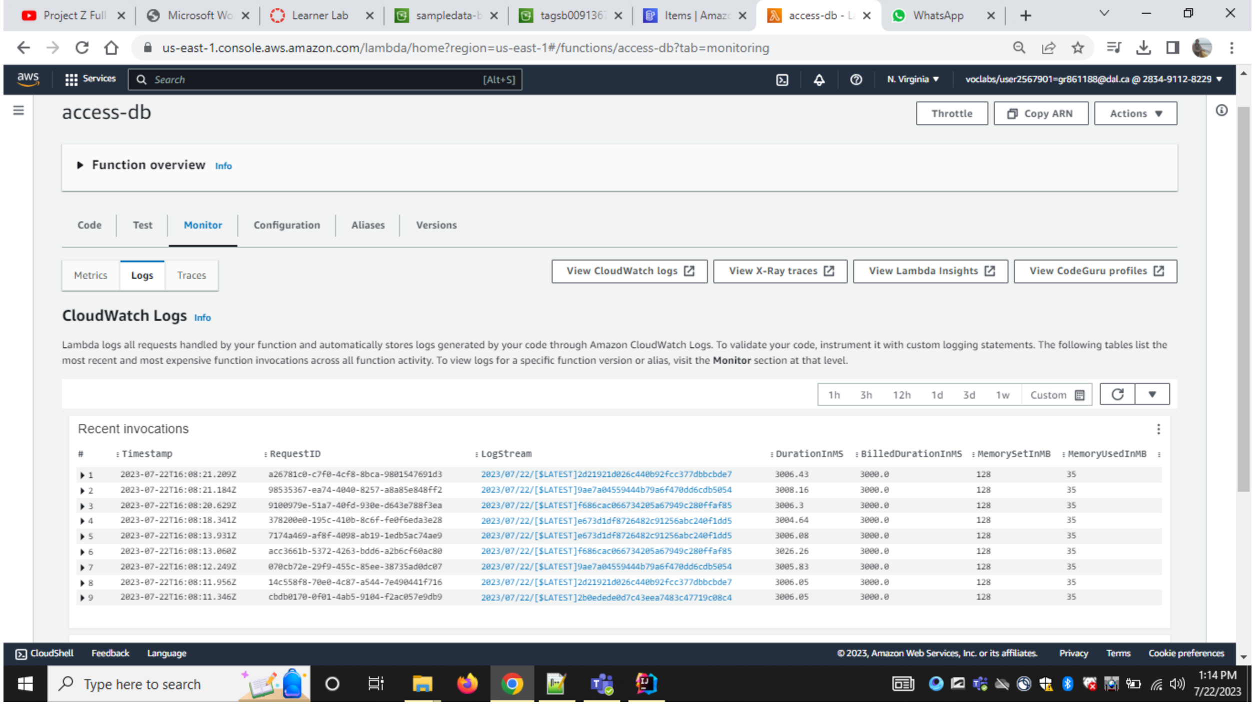
Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
1	2023-07-22T16:05:00.238Z	bfcb5421-9118-4aa0-9ce4-272eb6893da4	2023/07/22/[$LATEST]15f706d9f9e64e7784ad52a626240b09	4797.56	4798.0	512	308
2	2023-07-22T16:04:56.756Z	879055d3-653d-4269-b6f3-522f7b5f7b90	2023/07/22/[$LATEST]87a0051eeff043bc98e4911b30b47837	2291.66	2292.0	512	257
3	2023-07-22T16:04:56.053Z	7b34bb73-189a-44c6-9d7b-194970b9c683	2023/07/22/[$LATEST]a3d2a5ba5ffc4af9bcd425d1b59b489e	1584.23	1585.0	512	292
4	2023-07-22T16:04:56.005Z	56b53268-d020-4e99-b3d3-cc5039be3b7c	2023/07/22/[$LATEST]0f487e71717940f88e0b3df76c72ce97	1515.0	1516.0	512	270
5	2023-07-22T16:04:55.738Z	abe8e172-0150-4cbc-840d-b146bd75982a	2023/07/22/[$LATEST]dc437bf28b8f4cb69bc7ebf7d4379698	2296.22	2297.0	512	261
6	2023-07-22T16:04:55.699Z	ec53b8a2-5e0d-4ba6-b556-d9bb7a2e3d0d	2023/07/22/[$LATEST]d23c895d61c3437b81fcf8927ca2ead1	2300.64	2301.0	512	255
7	2023-07-22T16:04:55.530Z	c82cdcd4-0f1f-424c-98ee-fb43c59821c2	2023/07/22/[$LATEST]3bf12140fc2c4868a738ed53894527f3	2077.21	2078.0	512	273
8	2023-07-22T16:04:55.081Z	33f1d89e-9e82-42f9-b110-cc2c4ea1152d	2023/07/22/[$LATEST]15f706d9f9e64e7784ad52a626240b09	1604.87	1605.0	512	262
9	2023-07-22T16:04:55.059Z	842d4a9a-bb19-4cdc-bba5-048dabb6f182	2023/07/22/[$LATEST]07aa468a98464f4cbcedb51766f94440	1680.06	1681.0	512	271

Figure 11:Extarct lambda execution logs

11. Accessdb lambda execution Logs



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/access-db?tab=monitoring

access-db

CloudWatch Logs

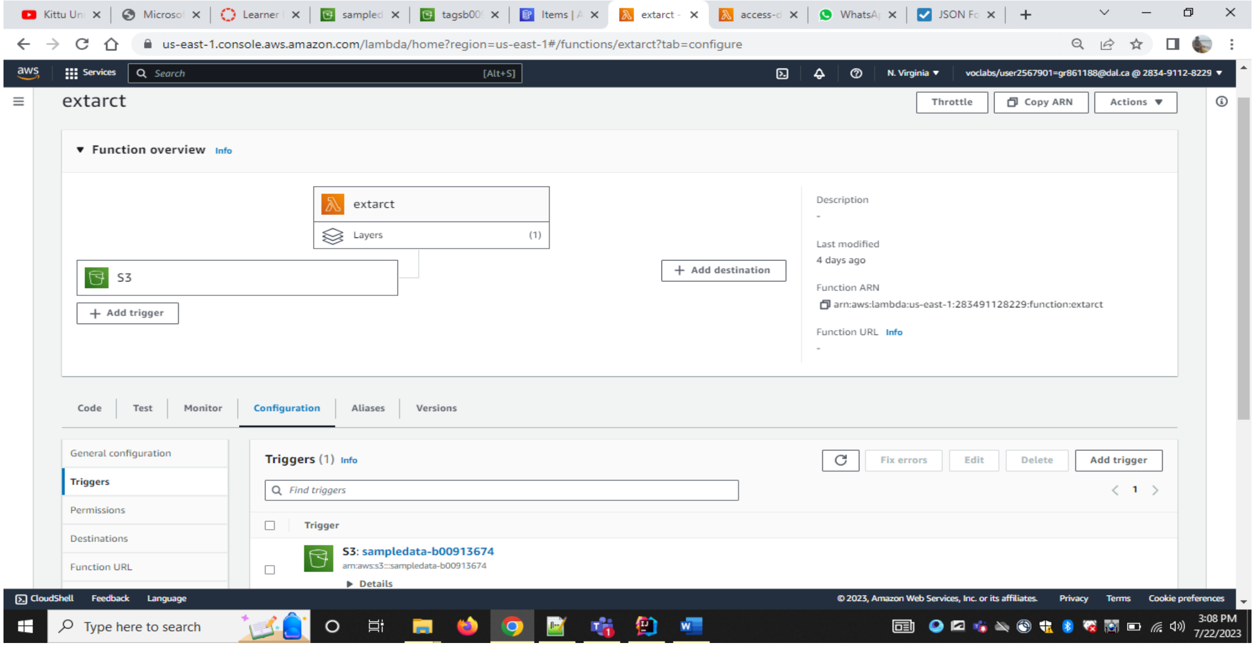
Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
1	2023-07-22T16:08:21.209Z	a26781c0-c7f0-4cf8-8bca-9801547691d3	2023/07/22/[$LATEST]2d21921d026c440b92fcc377dbbcbde7	3006.43	3000.0	128	35
2	2023-07-22T16:08:21.184Z	98535367-ea74-4040-8257-a8a85e848ff2	2023/07/22/[$LATEST]9ae7a04559444b79a6f470dd6cdb5054	3008.16	3000.0	128	35
3	2023-07-22T16:08:20.629Z	9100979e-51a7-40fd-930e-d643e788f3ea	2023/07/22/[$LATEST]f686cac066734205a67949c280ffaf85	3006.3	3000.0	128	35
4	2023-07-22T16:08:18.341Z	378200e0-195c-410b-8c6f-fe0f6eda3e28	2023/07/22/[$LATEST]e673d1df8726482c91256abc240f1dd5	3004.64	3000.0	128	35
5	2023-07-22T16:08:13.931Z	7174a469-af8f-4098-ab19-1edb5ac74ae9	2023/07/22/[$LATEST]e673d1df8726482c91256abc240f1dd5	3006.08	3000.0	128	35
6	2023-07-22T16:08:13.060Z	acc3661b-5372-4263-bdd6-a2b6cf60ac80	2023/07/22/[$LATEST]f686cac066734205a67949c280ffaf85	3026.26	3000.0	128	35
7	2023-07-22T16:08:12.249Z	070cb72e-29f9-455c-85ee-38735ad0dc07	2023/07/22/[$LATEST]9ae7a04559444b79a6f470dd6cdb5054	3005.83	3000.0	128	35
8	2023-07-22T16:08:11.956Z	14c558f8-70e0-4c87-a544-7e490441f716	2023/07/22/[$LATEST]2d21921d026c440b92fcc377dbbcbde7	3006.05	3000.0	128	35
9	2023-07-22T16:08:11.346Z	cbdb0170-0f01-4ab5-9104-f2ac057e9db9	2023/07/22/[$LATEST]2b0edede0d7c43eea7483c47719c08c4	3006.05	3000.0	128	35

Figure 12:Accessdb lambda execution logs

12. When an object is uploaded to S3 bucket (sampledata-b00913674) lambda function (extarct) will get triggered automatically



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/extarct?tab=configure

extarct

Function overview

extarct — Layers (1)

S3

Description: -

Last modified: 4 days ago

Function ARN: arn:aws:lambda:us-east-1:283491128229:function:extarct

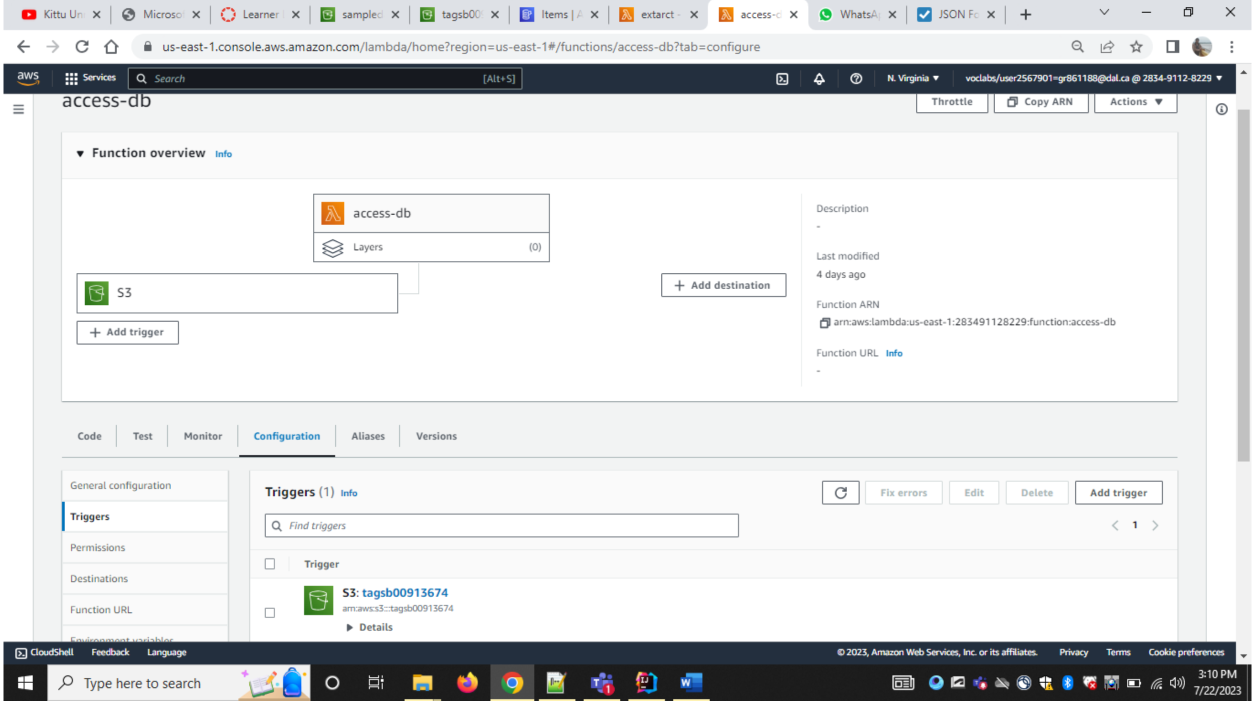
Function URL: -

Triggers (1)

S3: sampledata-b00913674
arn:aws:s3:::sampledata-b00913674

Figure 13:Trigger between s3 bucket and lambda

13. When an object is uploaded to S3 bucket (tagsb00913674) lambda function (accessdb) will get triggered automatically



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/access-db?tab=configure

access-db

Function overview

access-db — Layers (0)

S3

Description: -

Last modified: 4 days ago

Function ARN: arn:aws:lambda:us-east-1:283491128229:function:access-db

Function URL: -

Triggers (1)

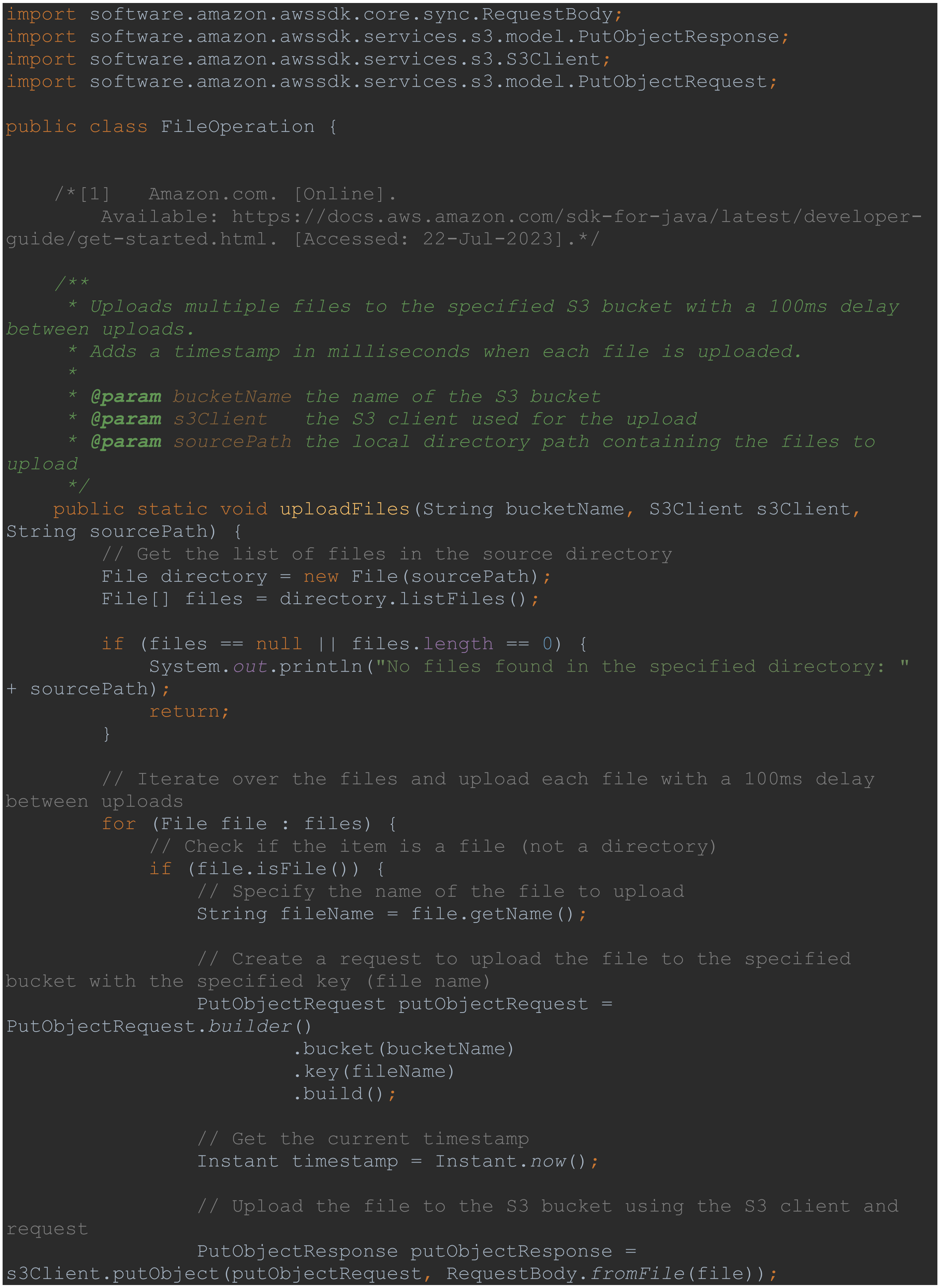
S3: tagsb00913674
arn:aws:s3:::tagsb00913674

Figure 14:Trigger between s3 bucket and lambda

My application code to upload files to s3 bucket (sampledata-b00913674)

```
package amazon.s3bucket;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketAlreadyExistsException;
import
software.amazon.awssdk.services.s3.model.BucketAlreadyOwnedByYouException;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;

public class App {

    /*[1]   Amazon.com. [Online].
        Available: https://docs.aws.amazon.com/sdk-for-java/latest/developer-
guide/get-started.html. [Accessed: 22-Jul-2023].*/

    public static void main(String[] args) {

        final String sourcePath = "C:\\Users\\AVuser\\Downloads\\tech\\";

        // Specify the name of the bucket
        final String bucketName = "sampledata-b00913674";

        // Specify the region for the bucket
        Region selectedRegion = Region.US_EAST_1;

        // Code to select a particular profile present in the .aws/credential
file
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.builder().profileName("accessed-June-22-
2023").build();

        // Create an S3 client using the specified region and credentials
        S3Client s3Client =
S3Client.builder().region(selectedRegion).credentialsProvider(credentialsProv
ider).build();

        // Call a method to upload a file to the created bucket
        FileOperation.uploadFiles(bucketName, s3Client, sourcePath);
    }
}
```

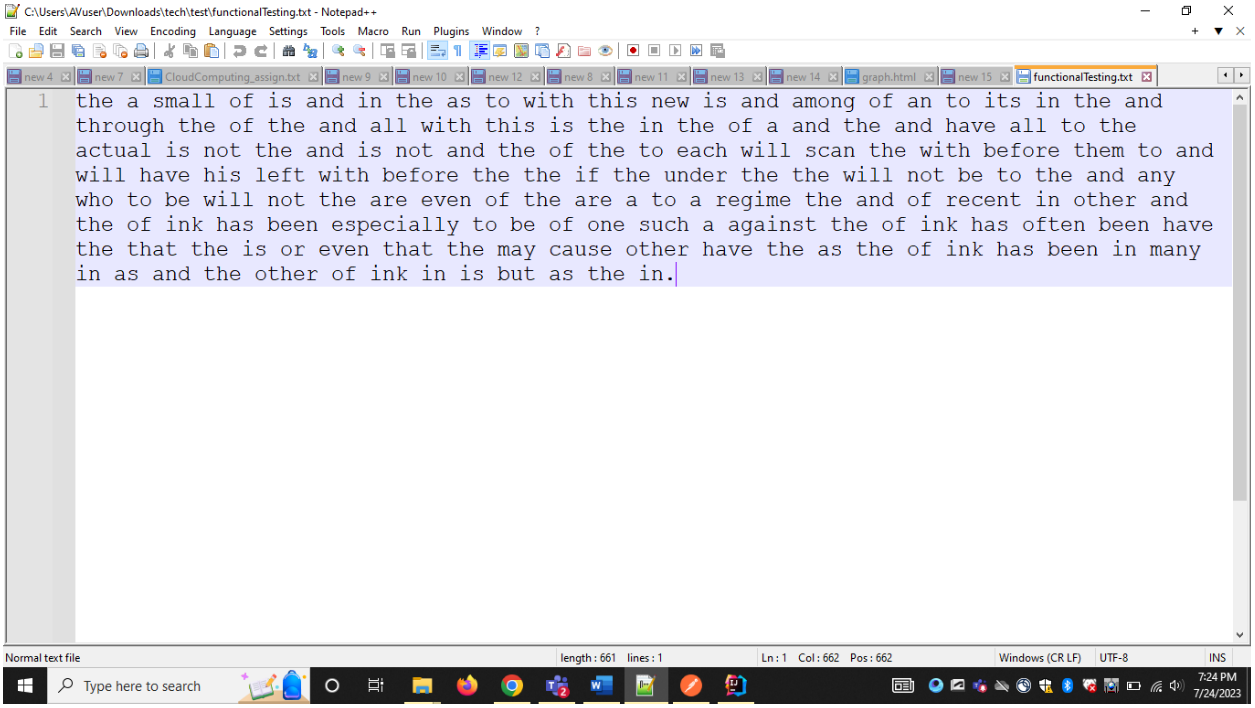
```
package amazon.s3bucket;

import java.io.File;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
```



```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;

public class FileOperation {


    /*[1]   Amazon.com. [Online].
        Available: https://docs.aws.amazon.com/sdk-for-java/latest/developer-
guide/get-started.html. [Accessed: 22-Jul-2023].*/

    /**
     * Uploads multiple files to the specified S3 bucket with a 100ms delay
between uploads.
     * Adds a timestamp in milliseconds when each file is uploaded.
     *
     * @param bucketName the name of the S3 bucket
     * @param s3Client   the S3 client used for the upload
     * @param sourcePath the local directory path containing the files to
upload
     */
    public static void uploadFiles(String bucketName, S3Client s3Client,
String sourcePath) {
        // Get the list of files in the source directory
        File directory = new File(sourcePath);
        File[] files = directory.listFiles();

        if (files == null || files.length == 0) {
            System.out.println("No files found in the specified directory: "
+ sourcePath);
            return;
        }

        // Iterate over the files and upload each file with a 100ms delay
between uploads
        for (File file : files) {
            // Check if the item is a file (not a directory)
            if (file.isFile()) {
                // Specify the name of the file to upload
                String fileName = file.getName();

                // Create a request to upload the file to the specified
bucket with the specified key (file name)
                PutObjectRequest putObjectRequest =
PutObjectRequest.builder()
                        .bucket(bucketName)
                        .key(fileName)
                        .build();

                // Get the current timestamp
                Instant timestamp = Instant.now();

                // Upload the file to the S3 bucket using the S3 client and
request
                PutObjectResponse putObjectResponse =
s3Client.putObject(putObjectRequest, RequestBody.fromFile(file));
```

```
                // Format the timestamp using the desired format
                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS");
                String formattedTimestamp =
LocalDateTime.ofInstant(timestamp, ZoneId.systemDefault())
                        .format(formatter);

                // Print the response from the upload operation along with
the formatted timestamp
                System.out.println("Uploaded file: " + fileName + ", ETag: "
+ putObjectResponse.eTag()
                        + ", Timestamp: " + formattedTimestamp);

                try {
                    // Sleep for 100ms before uploading the next file
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Functional Testing

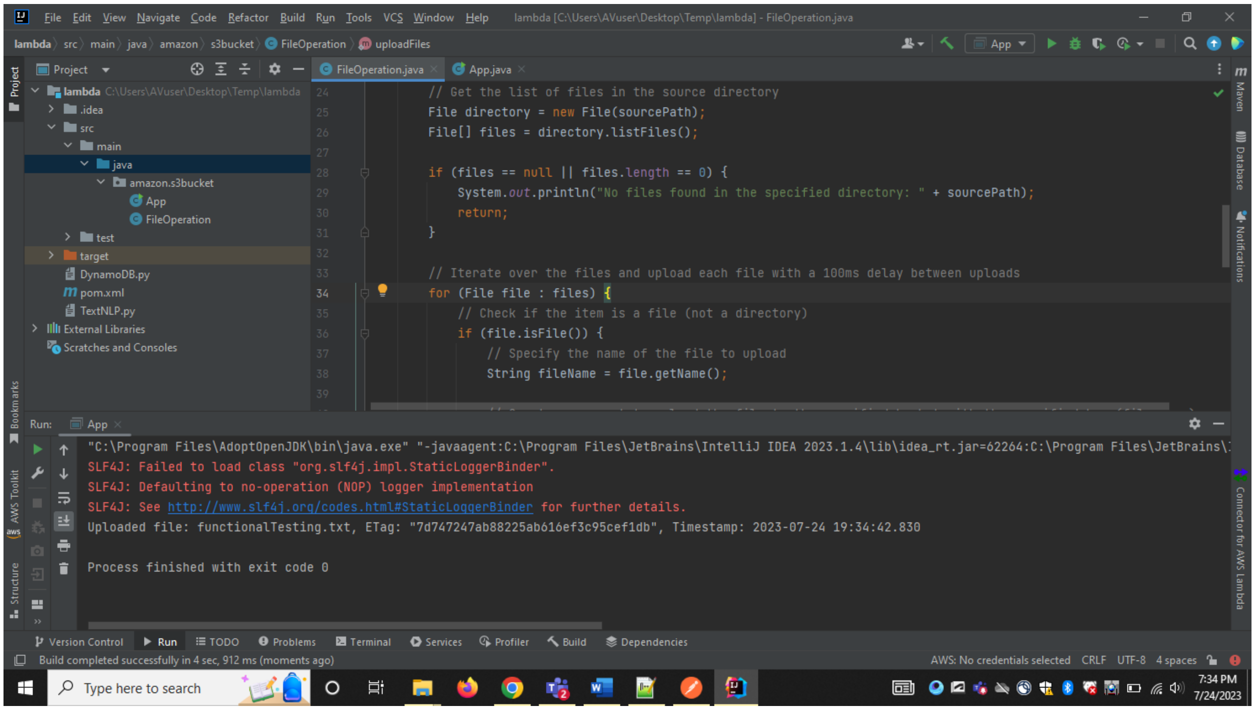


C:\Users\AVuser\Downloads\tech\test\functionalTesting.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 4
new 7
CloudComputing_assign.txt
new 9
new 10
new 12
new 8
new 11
new 13
new 14
graph.html
new 15
functionalTesting.txt
Normal text file
length : 661 lines : 1
Ln : 1 Col : 662 Pos : 662
Windows (CR LF) UTF-8 INS
Type here to search
7:24 PM
7/24/2023

the a small of is and in the as to with this new is and among of an to its in the and through the of the and all with this is the in the of a and the and have all to the actual is not the and is not and the of the to each will scan the with before them to and will have his left with before the the if the under the the will not be to the and any who to be will not the are even of the are a to a regime the and of recent in other and the of ink has been especially to be of one such a against the of ink has often been have the that the is or even that the may cause other have the as the of ink has been in many in as and the other of ink in is but as the in.

Figure 15: Text file to test functionality

In the above photo functionalTesting.txt file (which I am going to upload to the bucket), there are no words that are starting with capital letters. So, my total process should not keep any of the words in my DynamoDB table. I have deleted all the elements present in the table prior to testing.

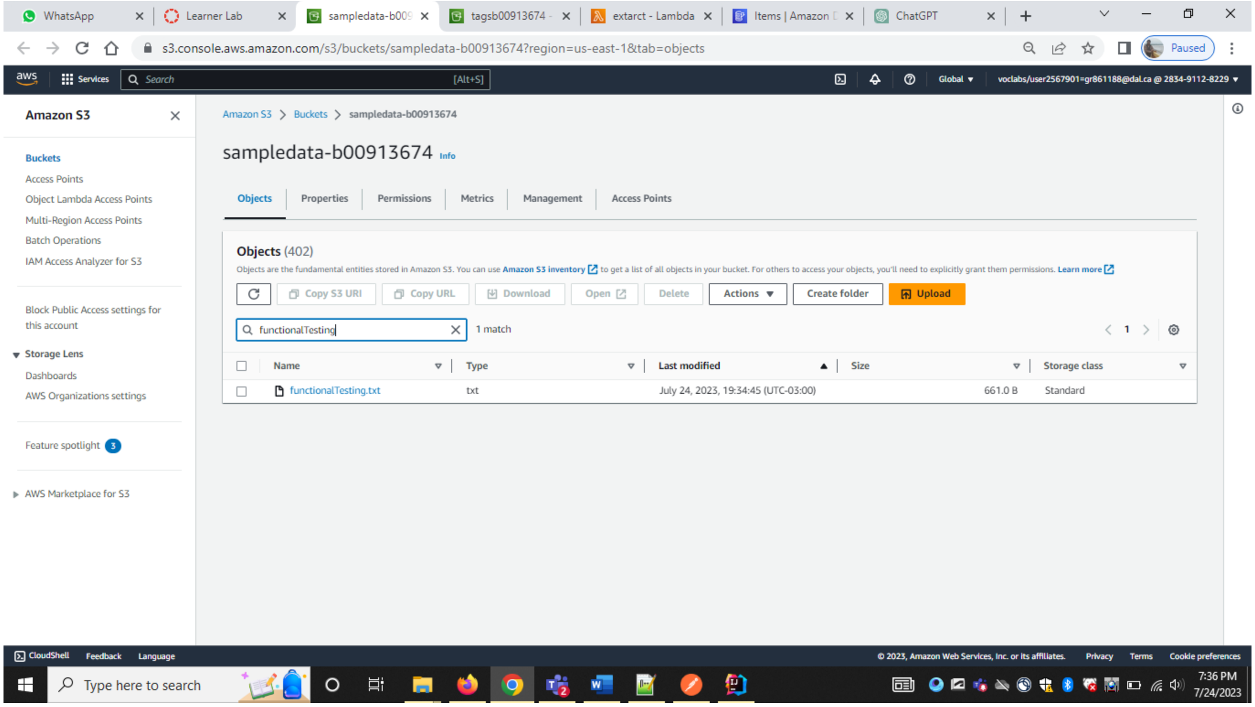


```
        // Get the list of files in the source directory
        File directory = new File(sourcePath);
        File[] files = directory.listFiles();

        if (files == null || files.length == 0) {
            System.out.println("No files found in the specified directory: " + sourcePath);
            return;
        }

        // Iterate over the files and upload each file with a 100ms delay between uploads
        for (File file : files) {
            // Check if the item is a file (not a directory)
            if (file.isFile()) {
                // Specify the name of the file to upload
                String fileName = file.getName();
```

```
"C:\Program Files\AdoptOpenJDK\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1.4\lib\idea_rt.jar=62264:C:\Program Files\JetBrains\
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Uploaded file: functionalTesting.txt, ETag: "7d747247ab88225ab616ef3c95cef1db", Timestamp: 2023-07-24 19:34:42.830

Process finished with exit code 0
```

Build completed successfully in 4 sec, 912 ms (moments ago)

Figure 16:Testing file uploaded to s3 bucket successfully from code



s3.console.aws.amazon.com/s3/buckets/sampledata-b00913674?region=us-east-1&tab=objects

Amazon S3 > Buckets > sampledata-b00913674

sampledata-b00913674 Info

Objects (402)

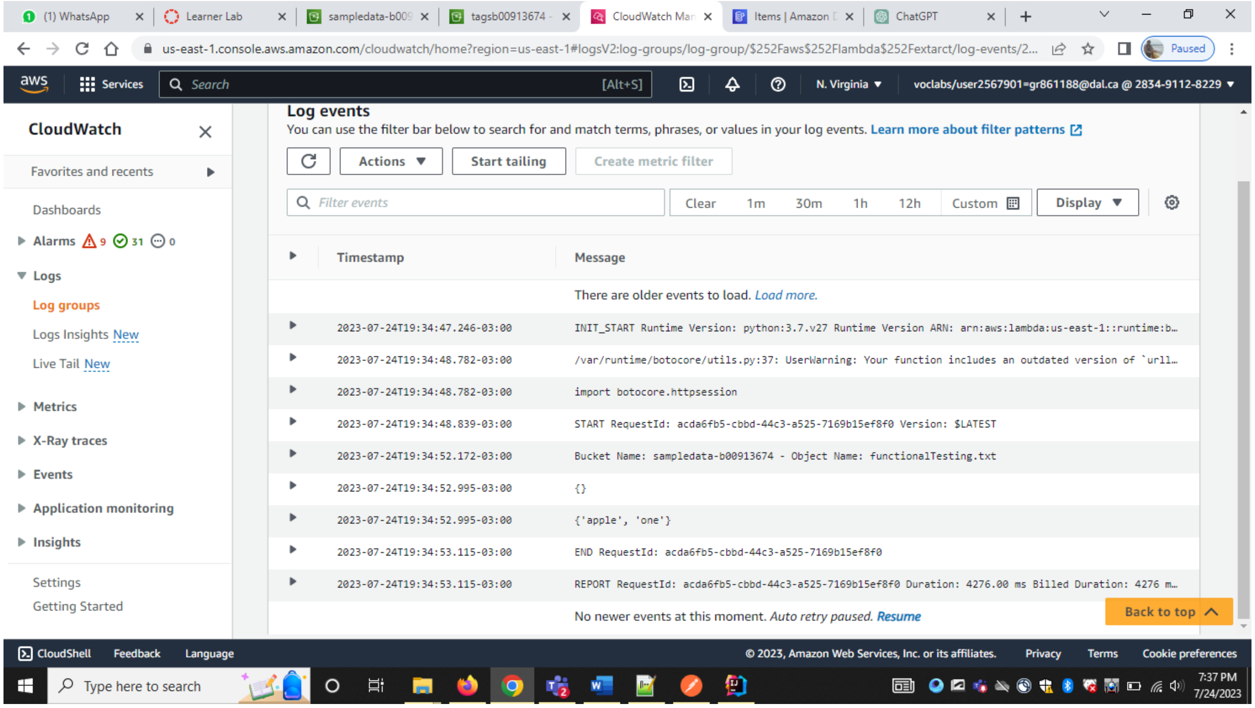
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

functionalTesting — 1 match

Name	Type	Last modified	Size	Storage class
functionalTesting.txt	txt	July 24, 2023, 19:34:45 (UTC-03:00)	661.0 B	Standard

Figure 17: Testing file in s3 bucket

Extract lambda log while testing



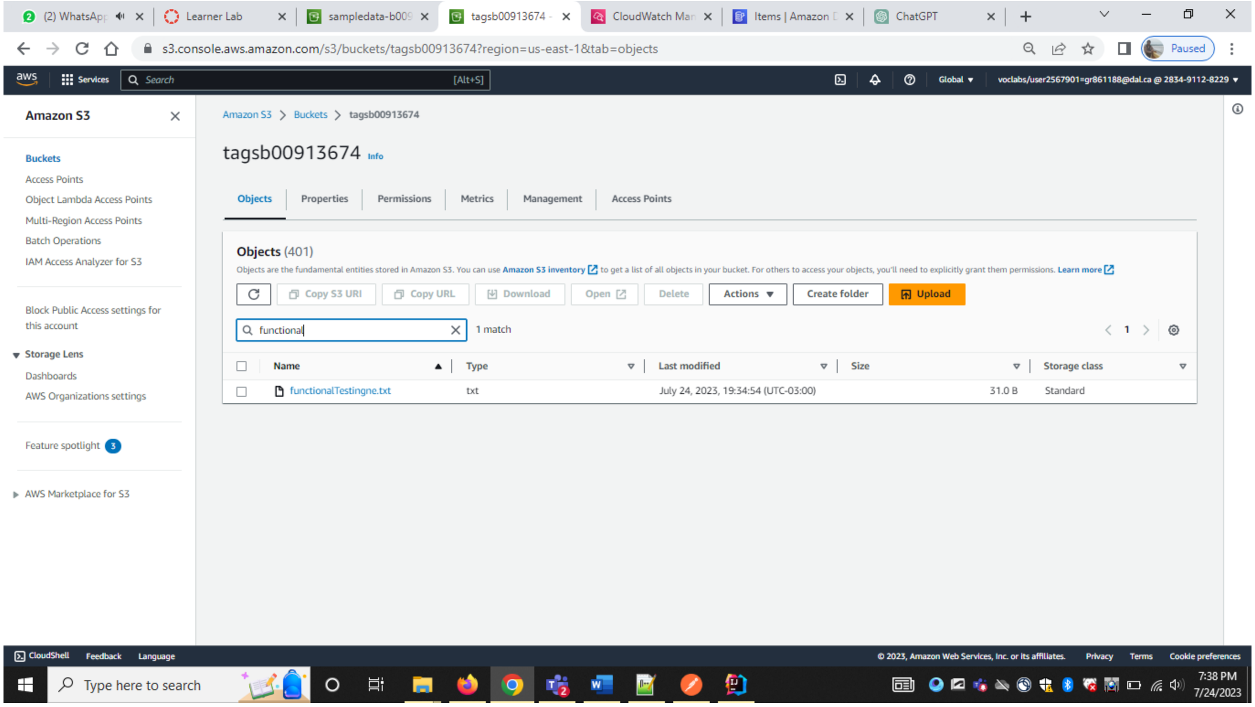
us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#logsV2:log-groups/log-group/$252Faws$252Flambda$252Fextarct/log-events/2...

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
                                There are older events to load. Load more.
2023-07-24T19:34:47.246-03:00   INIT_START Runtime Version: python:3.7.v27 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:b…
2023-07-24T19:34:48.782-03:00   /var/runtime/botocore/utils.py:37: UserWarning: Your function includes an outdated version of `urll…
2023-07-24T19:34:48.782-03:00   import botocore.httpsession
2023-07-24T19:34:48.839-03:00   START RequestId: acda6fb5-cbbd-44c3-a525-7169b15ef8f0 Version: $LATEST
2023-07-24T19:34:52.172-03:00   Bucket Name: sampledata-b00913674 - Object Name: functionalTesting.txt
2023-07-24T19:34:52.995-03:00   {}
2023-07-24T19:34:52.995-03:00   {'apple', 'one'}
2023-07-24T19:34:53.115-03:00   END RequestId: acda6fb5-cbbd-44c3-a525-7169b15ef8f0
2023-07-24T19:34:53.115-03:00   REPORT RequestId: acda6fb5-cbbd-44c3-a525-7169b15ef8f0 Duration: 4276.00 ms Billed Duration: 4276 m…
                                No newer events at this moment. Auto retry paused. Resume
```

Figure 18:extarct lambda log



s3.console.aws.amazon.com/s3/buckets/tagsb00913674?region=us-east-1&tab=objects

Amazon S3 > Buckets > tagsb00913674

tagsb00913674 Info

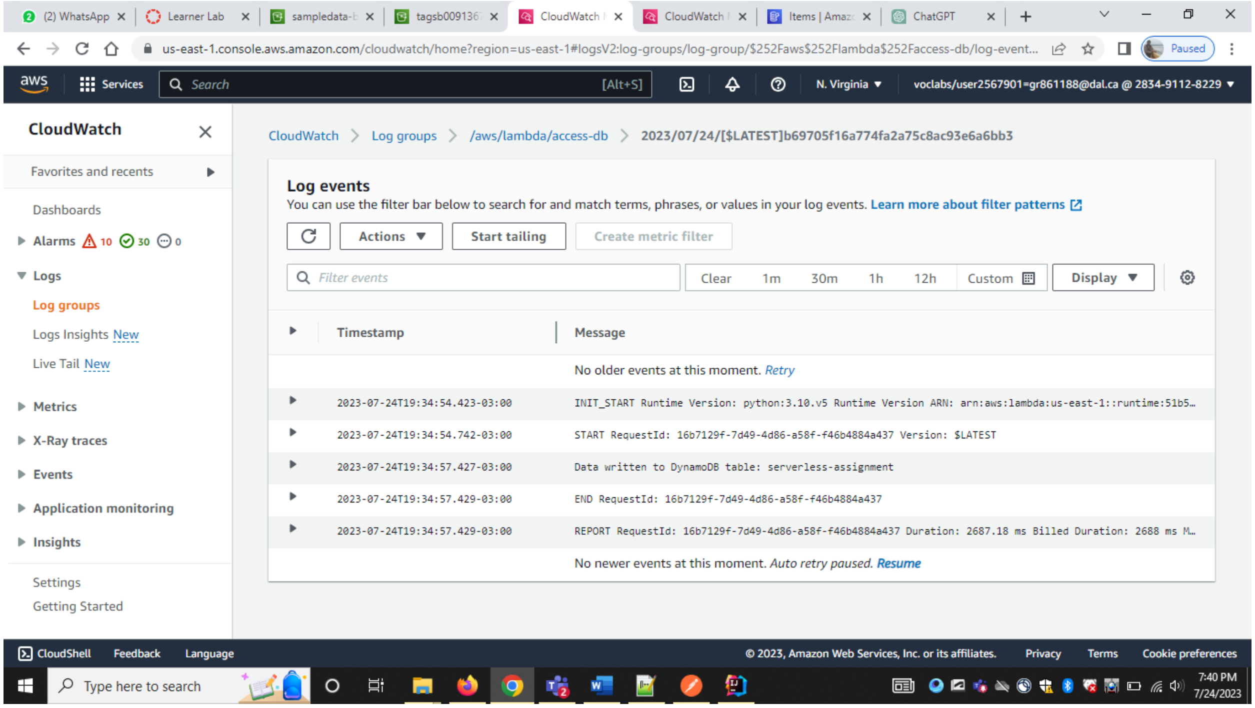
Objects (401)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

functional — 1 match

Name	Type	Last modified	Size	Storage class
functionalTestingne.txt	txt	July 24, 2023, 19:34:54 (UTC-03:00)	31.0 B	Standard

Figure 19: new file created in tags s3 bucket with extension 'ne'



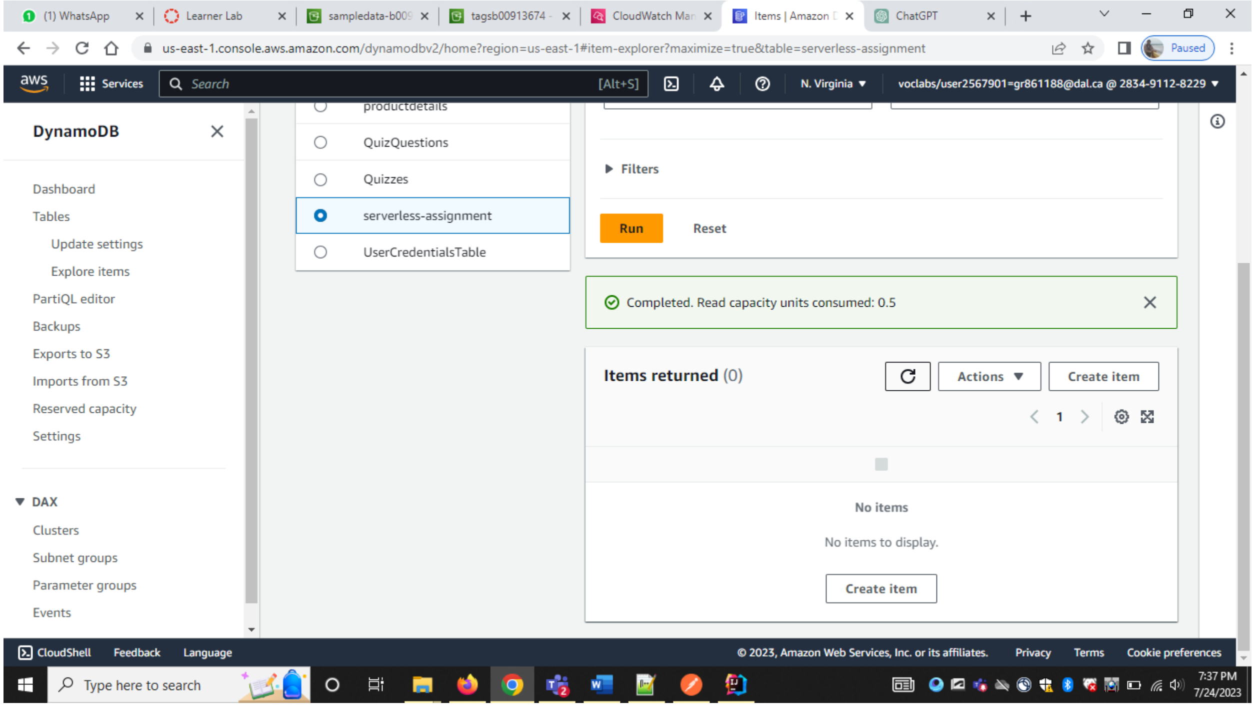
CloudWatch > Log groups > /aws/lambda/access-db > 2023/07/24/[$LATEST]b69705f16a774fa2a75c8ac93e6a6bb3

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
	No older events at this moment. Retry
2023-07-24T19:34:54.423-03:00	INIT_START Runtime Version: python:3.10.v5 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:51b5…
2023-07-24T19:34:54.742-03:00	START RequestId: 16b7129f-7d49-4d86-a58f-f46b4884a437 Version: $LATEST
2023-07-24T19:34:57.427-03:00	Data written to DynamoDB table: serverless-assignment
2023-07-24T19:34:57.429-03:00	END RequestId: 16b7129f-7d49-4d86-a58f-f46b4884a437
2023-07-24T19:34:57.429-03:00	REPORT RequestId: 16b7129f-7d49-4d86-a58f-f46b4884a437 Duration: 2687.18 ms Billed Duration: 2688 ms M…
	No newer events at this moment. Auto retry paused. Resume

Figure 20:AccessDB lambda function log



○ productdetails
○ QuizQuestions
○ Quizzes
● serverless-assignment
○ UserCredentialsTable

Completed. Read capacity units consumed: 0.5

Items returned (0)

No items

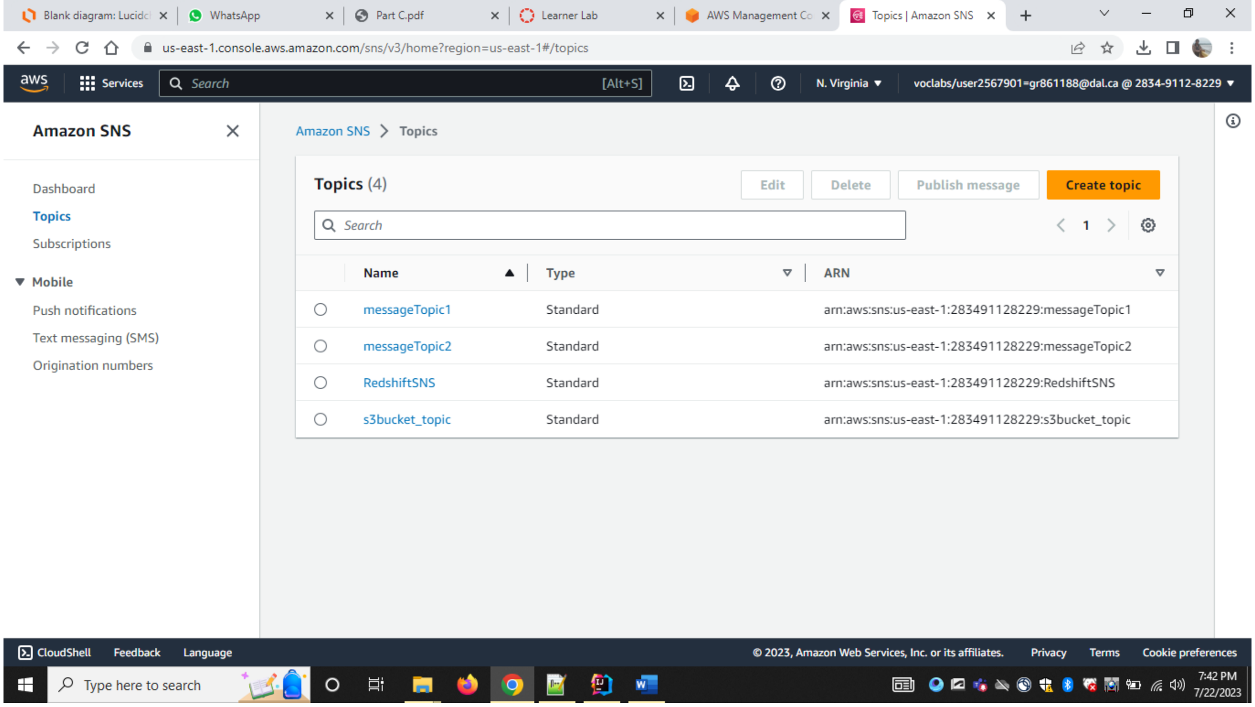
No items to display.

Figure 21: DynamoDB with no items created

In the above testing, we can see that I tried to upload a file with no word starting with capital letter and the extract lambda invocked once the file got uploaded to sampledata-b00913674 bucket. The lambda function processed the file and created a new file with extension ‘ne’ to filename and uploaded the file to tagsb00913674 bucket. Once file is successfully uploaded to tagsb00913674, accessdb lambda function will access that file and get the contents that are in json format and keep the entities in DynamoDB along with their updated count.

PART – 3 REPORT

1. List of SNS Topics available are (messageTopic1, messageTopic2)



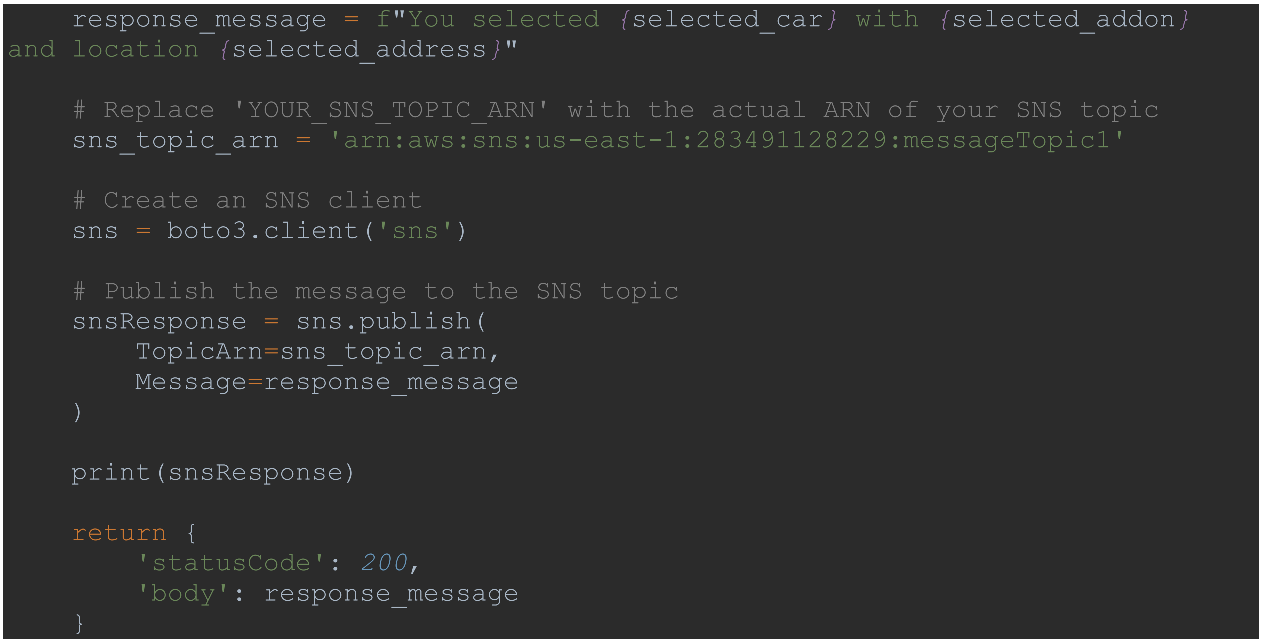
Topics (4)

	Name ▲	Type	ARN
○	messageTopic1	Standard	arn:aws:sns:us-east-1:283491128229:messageTopic1
○	messageTopic2	Standard	arn:aws:sns:us-east-1:283491128229:messageTopic2
○	RedshiftSNS	Standard	arn:aws:sns:us-east-1:283491128229:RedshiftSNS
○	s3bucket_topic	Standard	arn:aws:sns:us-east-1:283491128229:s3bucket_topic

Figure 22:List of SNS Topics available

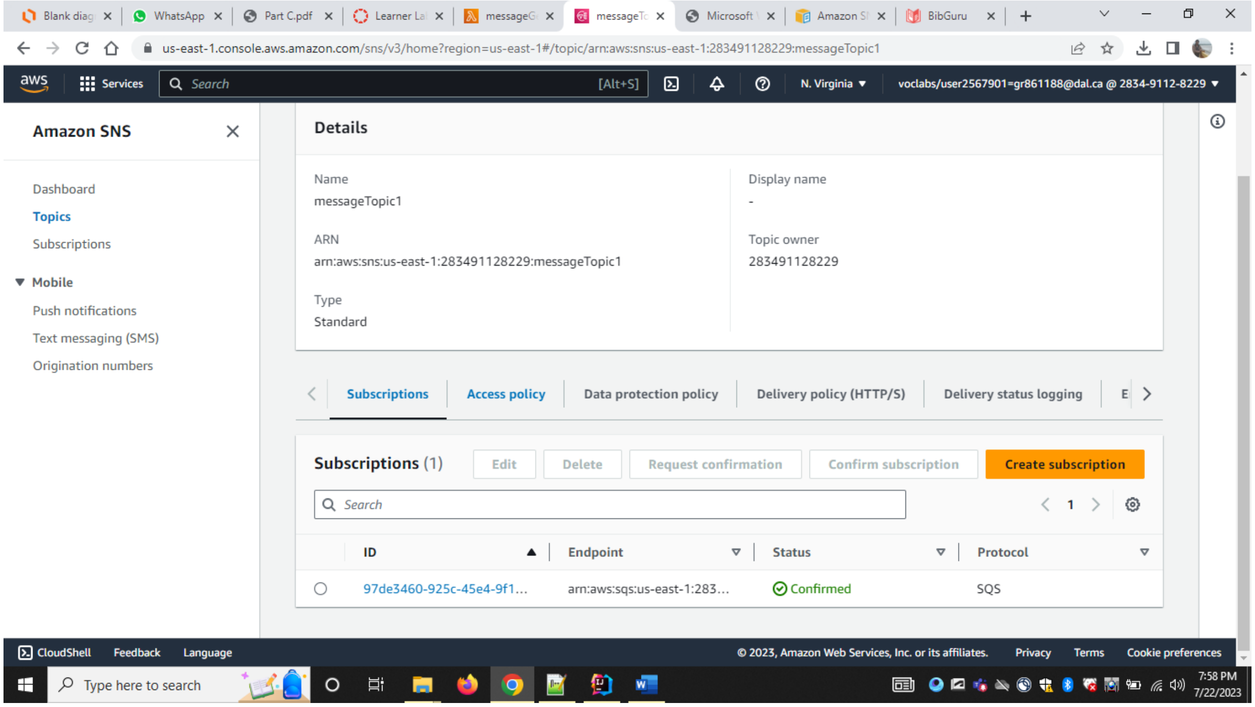
SNS topic messageTopic1 will be triggered by lambda function “mesageGenerator” where this lambda function will create a random string containing vehicle car type, addon, address. This lambda function will send publish message to SNS topic messageTopic1.

```
""" References:
[1] Amazon.com. [Online]. Available:
    https://docs.aws.amazon.com/code-
library/latest/ug/python_3_sns_code_examples.html.
    [Accessed: 22-Jul-2023]. """


import json
import random
import boto3

def lambda_handler(event, context):
    # TODO implement
    CAR = ["Compact", "mid-size Sedan", "SUV", "Luxury"]
    ADDON = ["GPS", "Camera"]
    ADDRESSES = ["6050 University Avenue", "1333 South Park St.", "5543 Clyde
St."]

    selected_car = random.choice(CAR)
    selected_addon = random.choice(ADDON)
    selected_address = random.choice(ADDRESSES)
```



```
    response_message = f"You selected {selected_car} with {selected_addon} and location {selected_address}"

    # Replace 'YOUR_SNS_TOPIC_ARN' with the actual ARN of your SNS topic
    sns_topic_arn = 'arn:aws:sns:us-east-1:283491128229:messageTopic1'

    # Create an SNS client
    sns = boto3.client('sns')

    # Publish the message to the SNS topic
    snsResponse = sns.publish(
        TopicArn=sns_topic_arn,
        Message=response_message
    )

    print(snsResponse)

    return {
        'statusCode': 200,
        'body': response_message
    }
```

SNS topic “messageTopic1” is created and SQS “firstQueue” is subscribed to it



Details

Name
messageTopic1

Display name
-

ARN
arn:aws:sns:us-east-1:283491128229:messageTopic1

Topic owner
283491128229

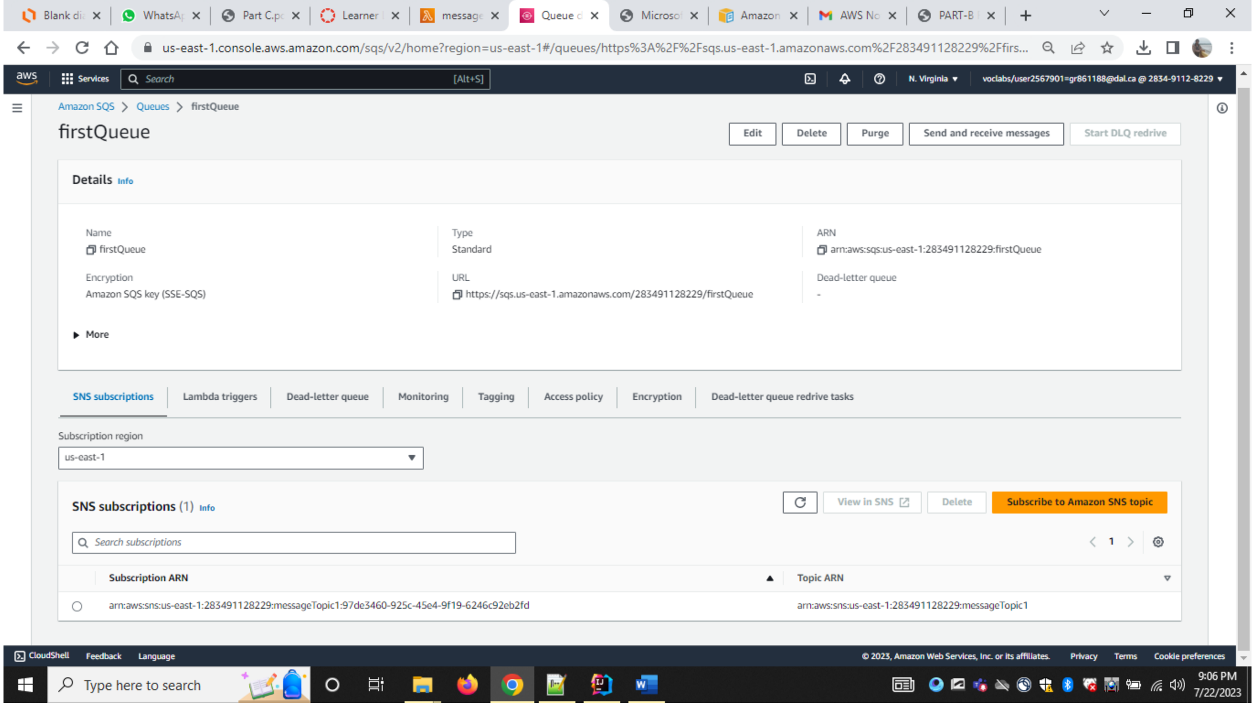
Type
Standard

Subscriptions (1)

ID	Endpoint	Status	Protocol
97de3460-925c-45e4-9f1...	arn:aws:sqs:us-east-1:283...	Confirmed	SQS

Figure 23:SNS Topic messageTopic1

SQS “firstQueue” is created and we can see that it is subscribed to SNS Topic “messageTopic1”



us-east-1.console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues/https%3A%2F%2Fsqs.us-east-1.amazonaws.com%2F283491128229%2Ffirs...

firstQueue

Details

Name
firstQueue

Type
Standard

ARN
arn:aws:sqs:us-east-1:283491128229:firstQueue

Encryption
Amazon SQS key (SSE-SQS)

URL
https://sqs.us-east-1.amazonaws.com/283491128229/firstQueue

Dead-letter queue
-

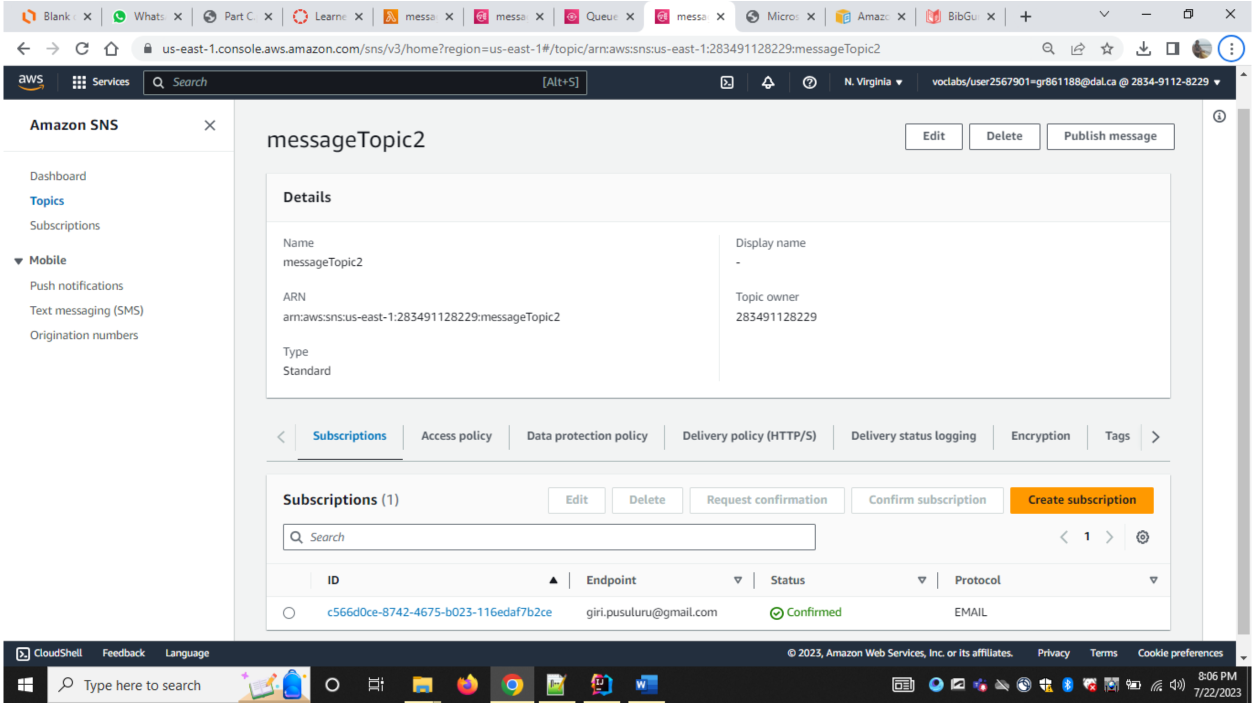
Subscription region
us-east-1

SNS subscriptions (1)

Subscription ARN	Topic ARN
arn:aws:sns:us-east-1:283491128229:messageTopic1:97de3460-925c-45e4-9f19-6246c92eb2fd	arn:aws:sns:us-east-1:283491128229:messageTopic1

Figure 24:SQS firstQueue and its subscription

Creating a one more SNS Topic “messageTopic2” and subscribing it to my email id to receive the emails



us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/topic/arn:aws:sns:us-east-1:283491128229:messageTopic2

messageTopic2

Details

Name
messageTopic2

Display name
-

ARN
arn:aws:sns:us-east-1:283491128229:messageTopic2

Topic owner
283491128229

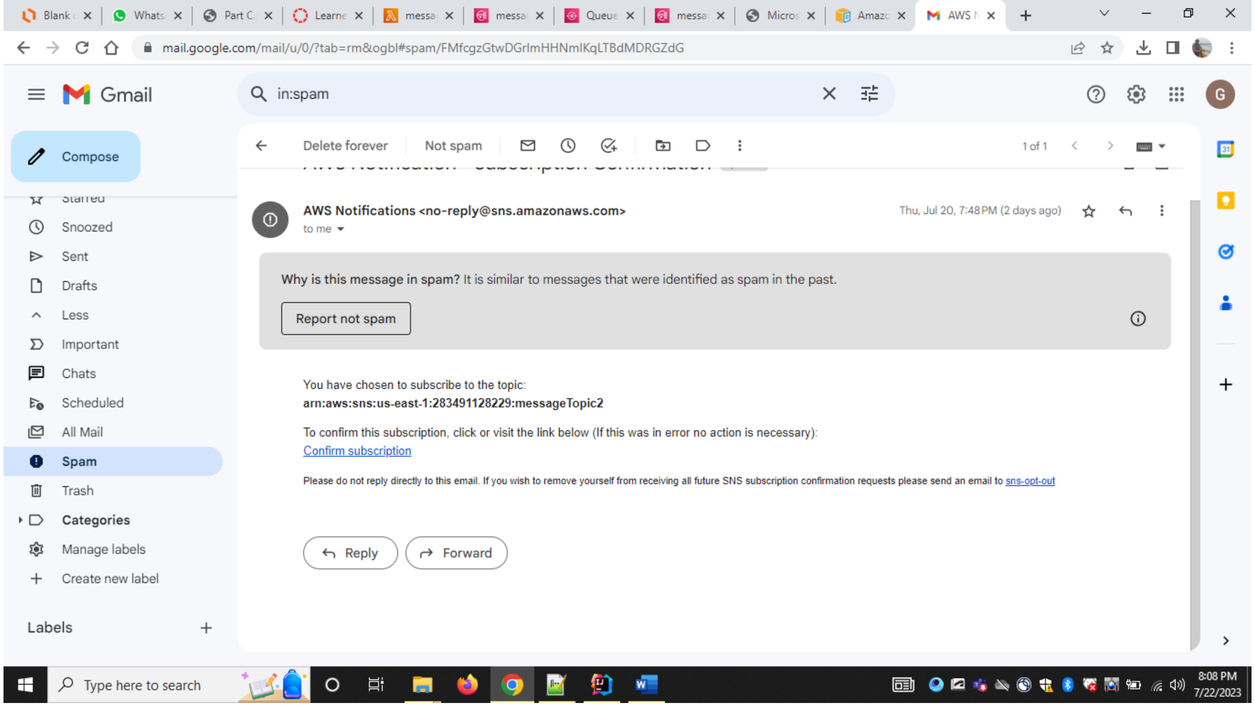
Type
Standard

Subscriptions (1)

ID	Endpoint	Status	Protocol
c566d0ce-8742-4675-b023-116edaf7b2ce	giri.pusuluru@gmail.com	Confirmed	EMAIL

Figure 25:SNS Topic messageTopic2 and its subscription

Email I got to subscribe for SNS Topic messageTopic2 and then confirming by clicking the link provided



AWS Notifications <no-reply@sns.amazonaws.com>
to me
Thu, Jul 20, 7:48 PM (2 days ago)

Why is this message in spam? It is similar to messages that were identified as spam in the past.

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:283491128229:messageTopic2

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

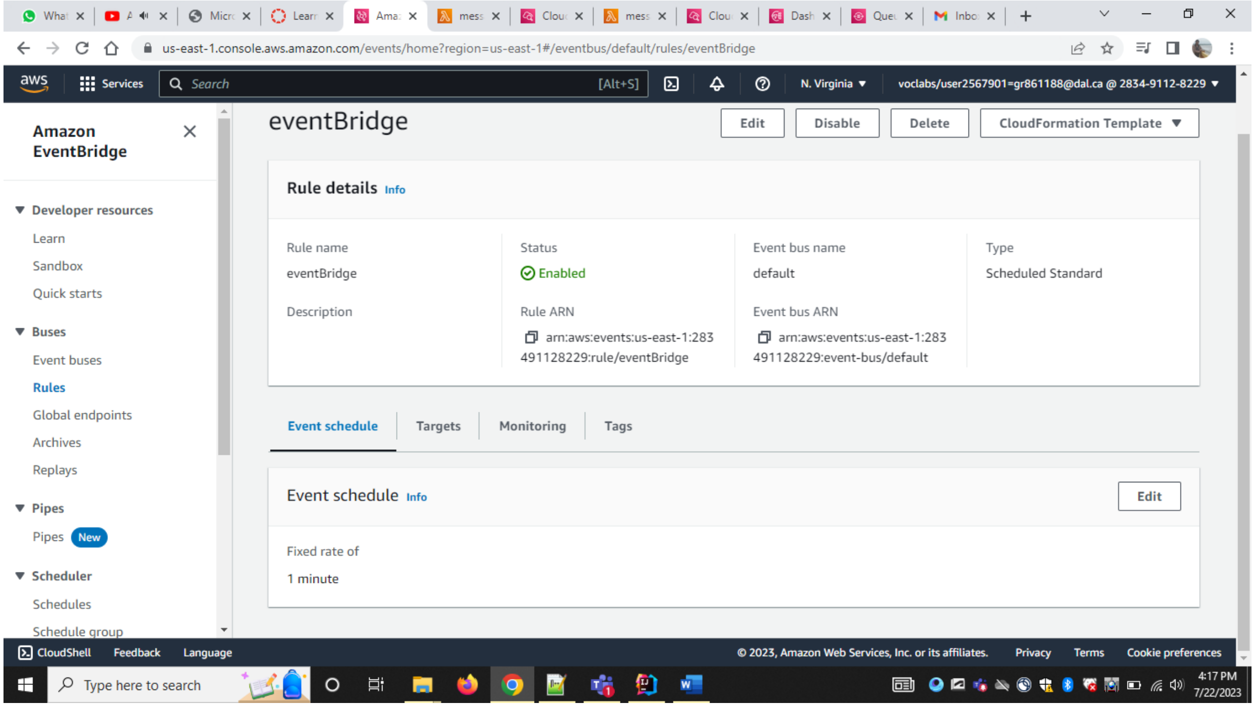
Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

Figure 26:Email notification to subscribe

Lambda “messageGetter” code that will poll on SQS “firstQueue” to get the message from the queue based on queue url and then publish it to SNS Topic “messageTopic2” and after that deleting the message from the queue.

```
""" References:
[1] Amazon.com. [Online]. Available:
    https://docs.aws.amazon.com/code-
library/latest/ug/python_3_sns_code_examples.html.
    [Accessed: 22-Jul-2023].
[2] “Sending and receiving messages in Amazon SQS - Boto3 1.28.9
documentation,” Amazonaws.com. [Online].
    Available:
https://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs-example-
sending-receiving-msgs.html.
    [Accessed: 22-Jul-2023].
"""

import json
import boto3

def lambda_handler(event, context):

    sns = boto3.client('sns')

    sns_topic_arn = 'arn:aws:sns:us-east-1:283491128229:messageTopic2'
    # Create an SQS client
    sqs = boto3.client('sqs')

    # Receive messages from the queue
    response = sqs.receive_message(
        QueueUrl=
'https://sqs.us-east-1.amazonaws.com/283491128229/firstQueue',
        MaxNumberOfMessages=1,  # Number of messages to retrieve at once
        WaitTimeSeconds=10  # Wait time for receiving messages
    )

    print(response)

    # Check if there are messages in the response
    if 'Messages' in response:
        message = response['Messages'][0]  # Get the first message

        # Extract the body of the message
        body = json.loads(message['Body'])

        # Getting the message sent by messageGenerator Code
        messageReceived = body['Message']

        print(messageReceived)

        snsResponse = sns.publish(
            TopicArn='arn:aws:sns:us-east-1:283491128229:messageTopic2',
            Message=messageReceived
            )

        print(snsResponse)

        # Delete the message from the queue after processing (optional,
depending on your use case)
        receipt_handle = message['ReceiptHandle']
        sqs.delete_message(
            QueueUrl="https://sqs.us-east-
1.amazonaws.com/283491128229/firstQueue",
            ReceiptHandle=receipt_handle
            )
        print('Message Deleted Successfully')

    return {
        'statusCode': 200,
        'body': json.dumps('SQS message processed successfully!')
    }
```

Amazon Event Bridge rule “eventBridge” is created where it will invoke the lambda function “messageGetter” for every 1 minutes



eventBridge

Rule details

Rule name	Status	Event bus name	Type
eventBridge	Enabled	default	Scheduled Standard
Description	Rule ARN	Event bus ARN	
	arn:aws:events:us-east-1:283491128229:rule/eventBridge	arn:aws:events:us-east-1:283491128229:event-bus/default	

Event schedule

Fixed rate of

1 minute

Figure 27:EventBridge to schedule

Procedure I am following to do this task

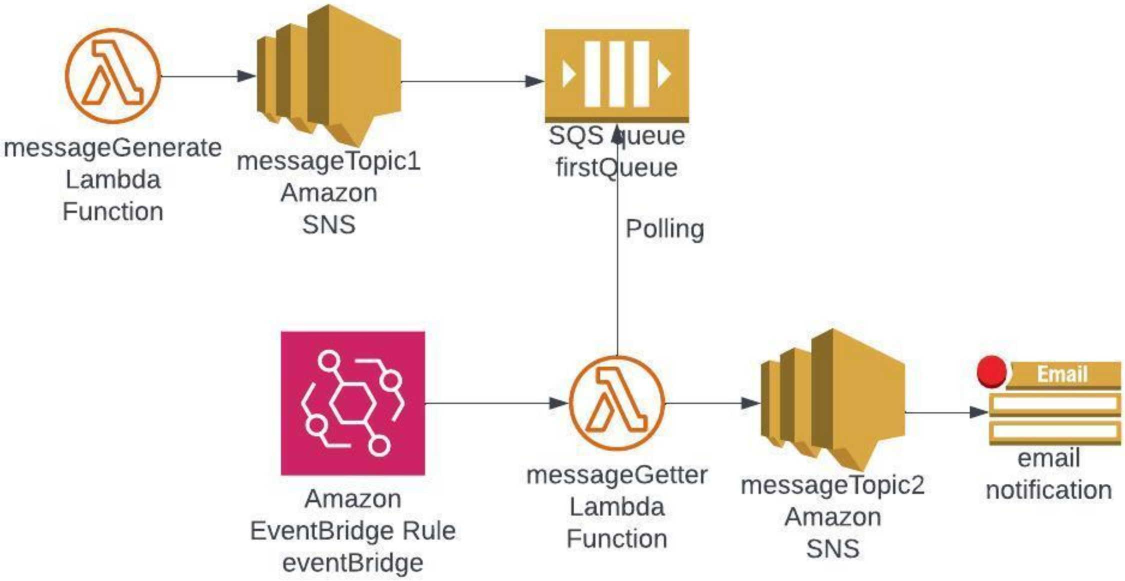
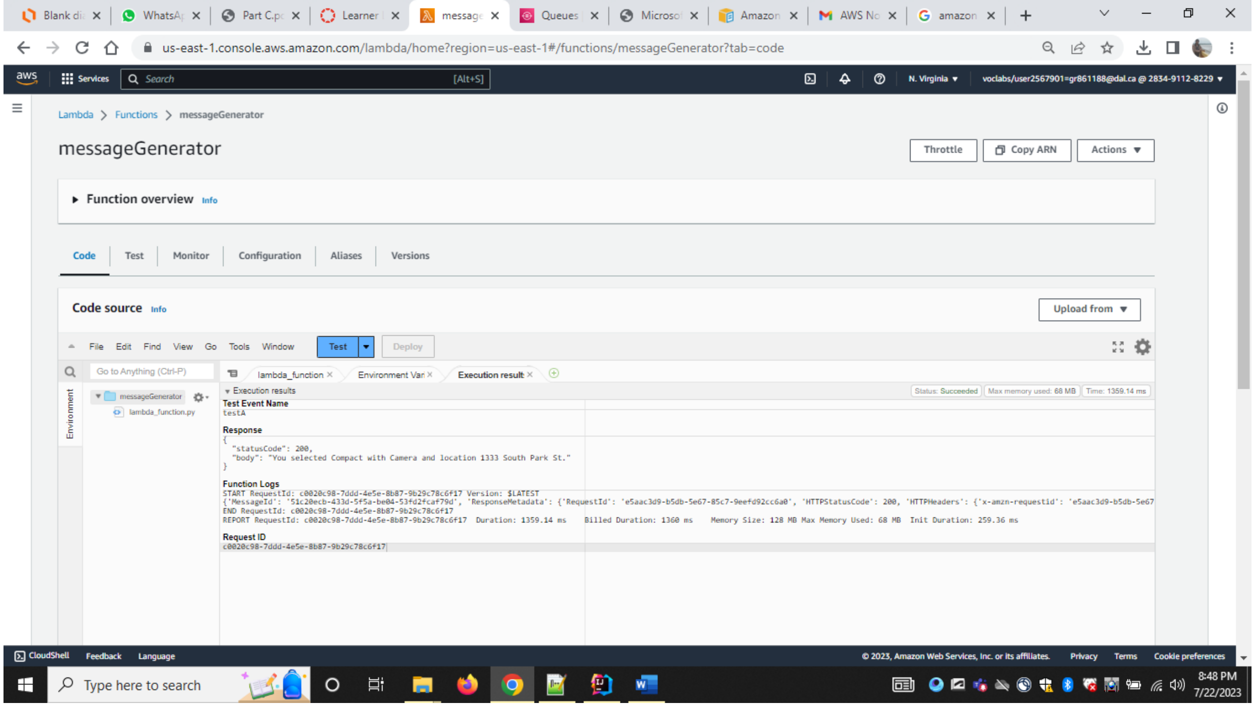


Figure 28: Architecture followed to send an email [12]

I have invoked messageGenerator lambda function



Execution results

Test Event Name

```
testA
```

Response

```
{
  "statusCode": 200,
  "body": "You selected Compact with Camera and location 1333 South Park St."
}
```

Function Logs

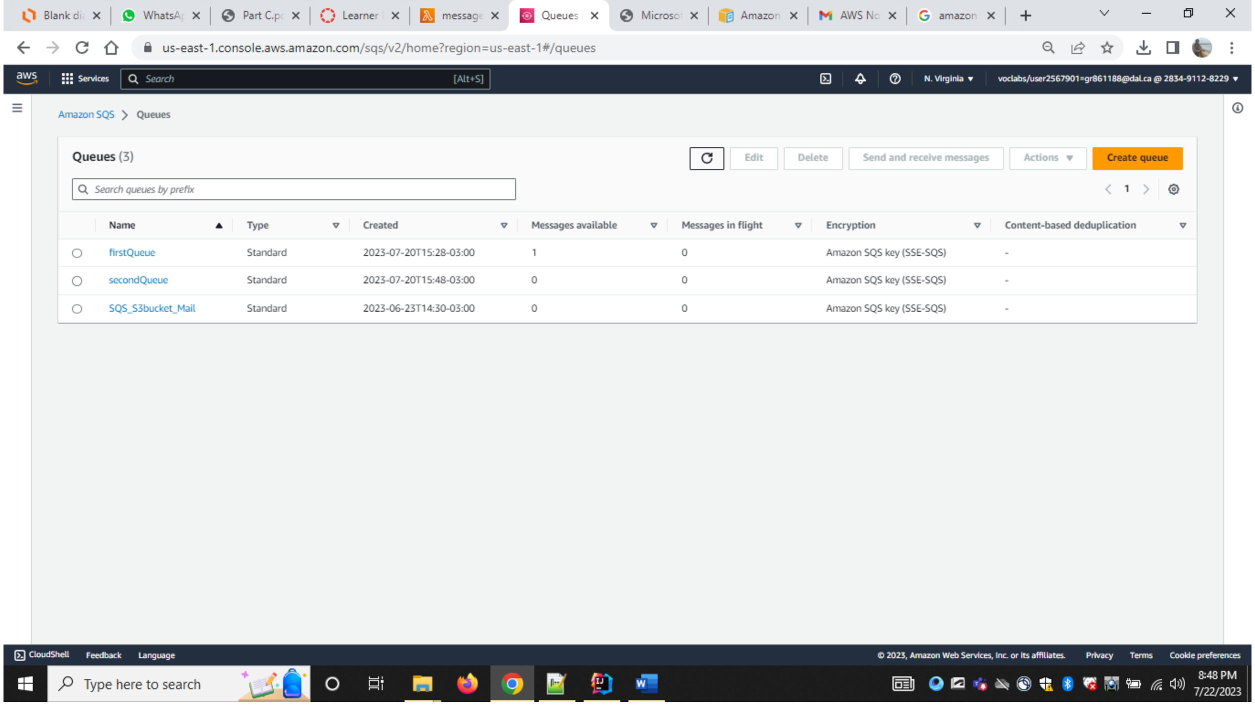
```
START RequestId: c0020c98-7ddd-4e5e-8b87-9b29c78c6f17 Version: $LATEST
{'MessageId': '51c20ecb-433d-5f5a-be04-53fd2fcaf79d', 'ResponseMetadata': {'RequestId': 'e5aac3d9-b5db-5e67-85c7-9eefd92cc6a0', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'e5aac3d9-b5db-5e67
END RequestId: c0020c98-7ddd-4e5e-8b87-9b29c78c6f17
REPORT RequestId: c0020c98-7ddd-4e5e-8b87-9b29c78c6f17  Duration: 1359.14 ms    Billed Duration: 1360 ms    Memory Size: 128 MB Max Memory Used: 68 MB  Init Duration: 259.36 ms
```

Request ID

```
c0020c98-7ddd-4e5e-8b87-9b29c78c6f17
```

Figure 29: Invoked lambda to generate a random message

The message is in queue now

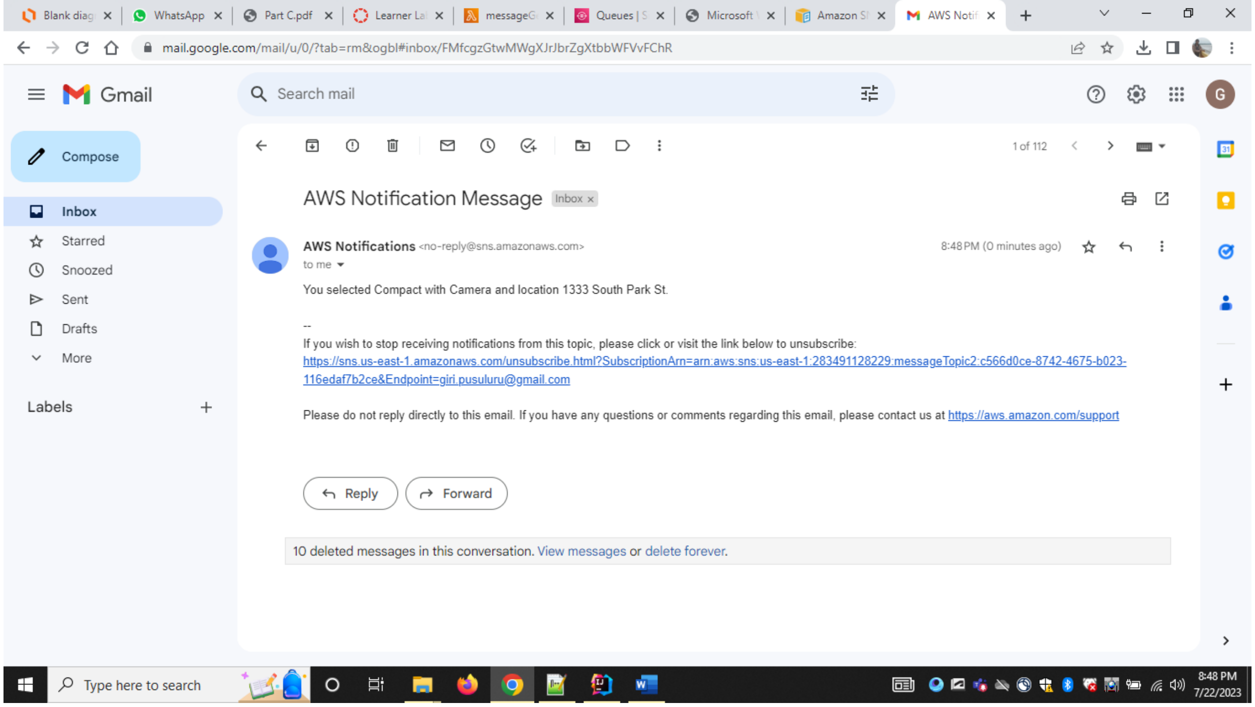

us-east-1.console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Amazon SQS > Queues
Edit
Delete
Send and receive messages
Actions
Create queue
Search queues by prefix

Queues (3)

	Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
○	firstQueue	Standard	2023-07-20T15:28-03:00	1	0	Amazon SQS key (SSE-SQS)	-
○	secondQueue	Standard	2023-07-20T15:48-03:00	0	0	Amazon SQS key (SSE-SQS)	-
○	SQS_S3bucket_Mail	Standard	2023-06-23T14:30-03:00	0	0	Amazon SQS key (SSE-SQS)	-

Figure 30: Message arrived into queue

Email Notification received



AWS Notification Message Inbox ×

AWS Notifications <no-reply@sns.amazonaws.com>
to me

8:48 PM (0 minutes ago)

You selected Compact with Camera and location 1333 South Park St.

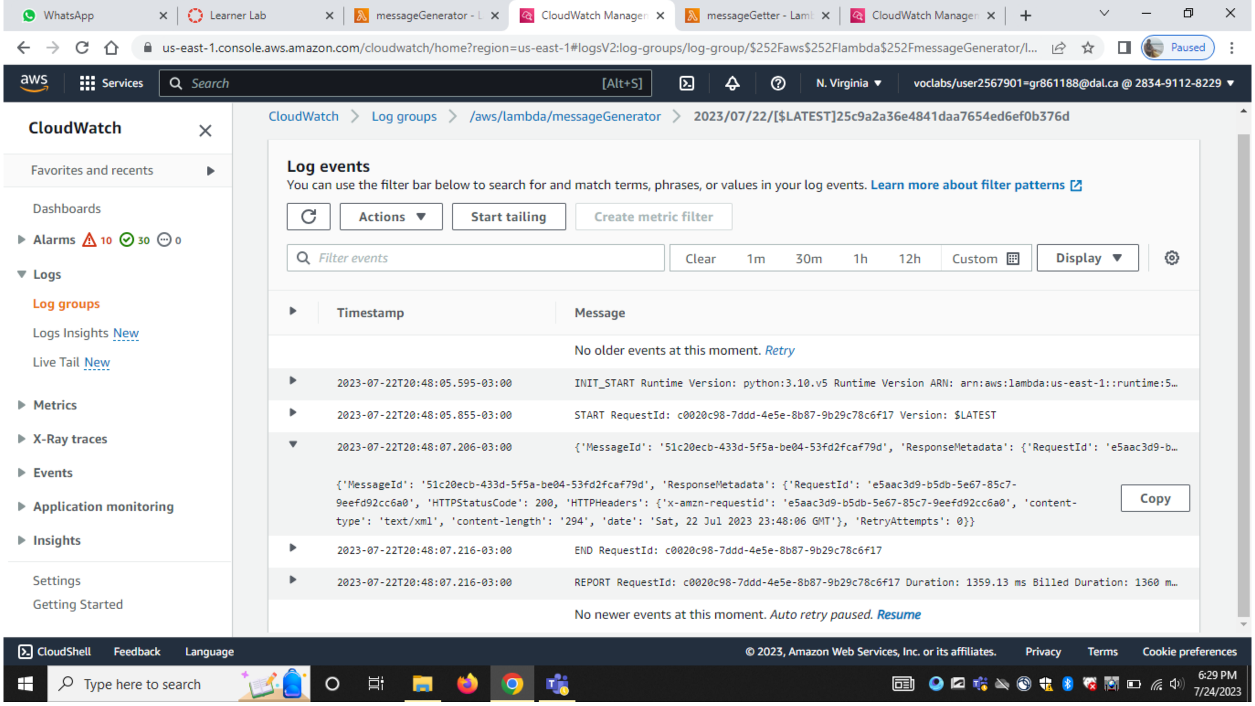
--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:283491128229:messageTopic2:c566d0ce-8742-4675-b023-116edaf7b2ce&Endpoint=giri.pusuluru@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

10 deleted messages in this conversation. View messages or delete forever.

Figure 31: Email notification received

messageGenerator Lambda function Log



CloudWatch > Log groups > /aws/lambda/messageGenerator > 2023/07/22/[$LATEST]25c9a2a36e4841daa7654ed6ef0b376d

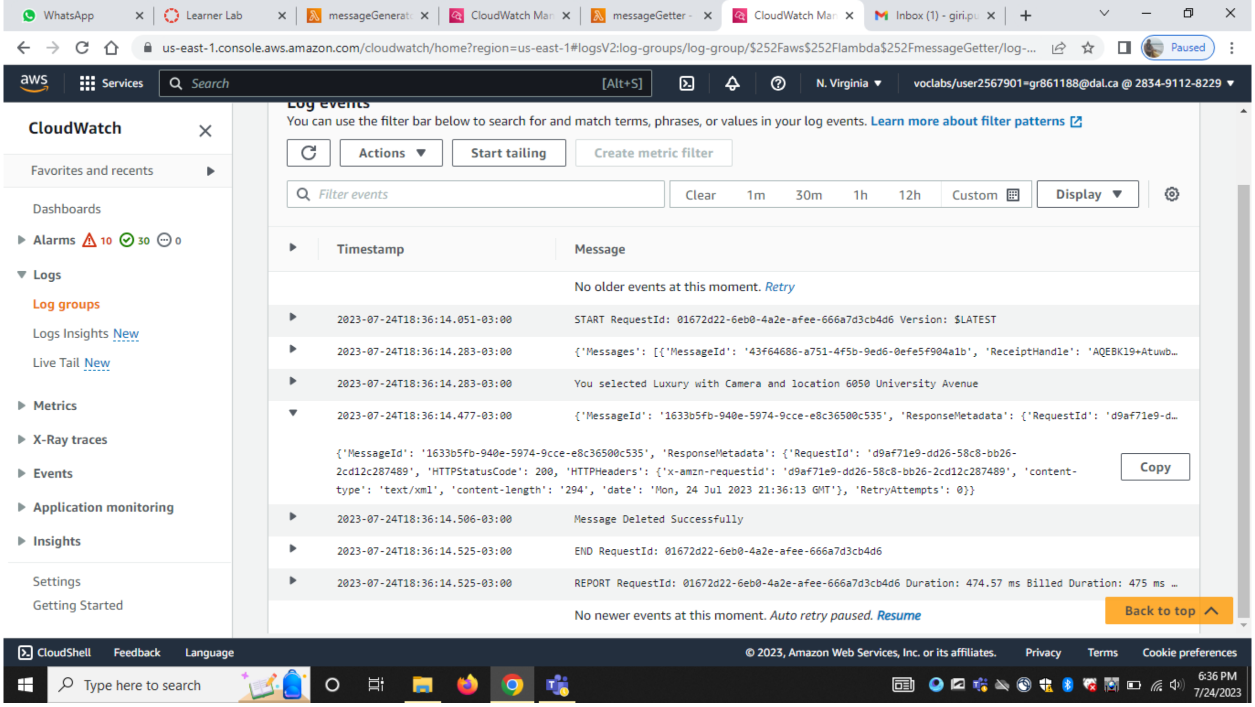
Log events

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
                                 No older events at this moment. Retry
2023-07-22T20:48:05.595-03:00    INIT_START Runtime Version: python:3.10.v5 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:5…
2023-07-22T20:48:05.855-03:00    START RequestId: c0020c98-7ddd-4e5e-8b87-9b29c78c6f17 Version: $LATEST
2023-07-22T20:48:07.206-03:00    {'MessageId': '51c20ecb-433d-5f5a-be04-53fd2fcaf79d', 'ResponseMetadata': {'RequestId': 'e5aac3d9-b…

{'MessageId': '51c20ecb-433d-5f5a-be04-53fd2fcaf79d', 'ResponseMetadata': {'RequestId': 'e5aac3d9-b5db-5e67-85c7-
9eefd92cc6a0', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'e5aac3d9-b5db-5e67-85c7-9eefd92cc6a0', 'content-
type': 'text/xml', 'content-length': '294', 'date': 'Sat, 22 Jul 2023 23:48:06 GMT'}, 'RetryAttempts': 0}}

2023-07-22T20:48:07.216-03:00    END RequestId: c0020c98-7ddd-4e5e-8b87-9b29c78c6f17
2023-07-22T20:48:07.216-03:00    REPORT RequestId: c0020c98-7ddd-4e5e-8b87-9b29c78c6f17 Duration: 1359.13 ms Billed Duration: 1360 m…
                                 No newer events at this moment. Auto retry paused. Resume
```

Figure 32:Log of messageGenerator lambda

messageGetter Lambda function Log



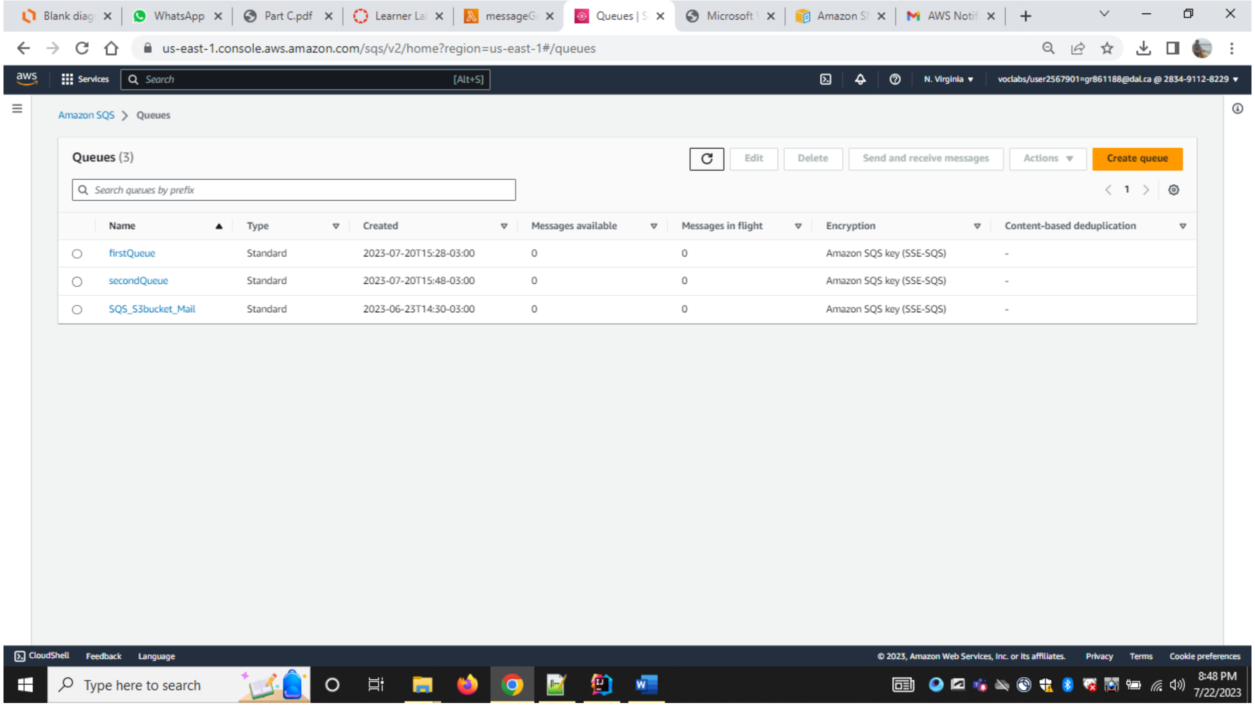
Log events

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
                                  No older events at this moment. Retry
2023-07-24T18:36:14.051-03:00     START RequestId: 01672d22-6eb0-4a2e-afee-666a7d3cb4d6 Version: $LATEST
2023-07-24T18:36:14.283-03:00     {'Messages': [{'MessageId': '43f64686-a751-4f5b-9ed6-0efe5f904a1b', 'ReceiptHandle': 'AQEBK19+Atuwb…
2023-07-24T18:36:14.283-03:00     You selected Luxury with Camera and location 6050 University Avenue
2023-07-24T18:36:14.477-03:00     {'MessageId': '1633b5fb-940e-5974-9cce-e8c36500c535', 'ResponseMetadata': {'RequestId': 'd9af71e9-d…

{'MessageId': '1633b5fb-940e-5974-9cce-e8c36500c535', 'ResponseMetadata': {'RequestId': 'd9af71e9-dd26-58c8-bb26-
2cd12c287489', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'd9af71e9-dd26-58c8-bb26-2cd12c287489', 'content-
type': 'text/xml', 'content-length': '294', 'date': 'Mon, 24 Jul 2023 21:36:13 GMT'}, 'RetryAttempts': 0}}

2023-07-24T18:36:14.506-03:00     Message Deleted Successfully
2023-07-24T18:36:14.525-03:00     END RequestId: 01672d22-6eb0-4a2e-afee-666a7d3cb4d6
2023-07-24T18:36:14.525-03:00     REPORT RequestId: 01672d22-6eb0-4a2e-afee-666a7d3cb4d6 Duration: 474.57 ms Billed Duration: 475 ms …
                                  No newer events at this moment. Auto retry paused. Resume
```

Figure 33:Log of messageGetter lambda function

Message got deleted in the queue once it is processed



Queues (3)

	Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
○	firstQueue	Standard	2023-07-20T15:28-03:00	0	0	Amazon SQS key (SSE-SQS)	-
○	secondQueue	Standard	2023-07-20T15:48-03:00	0	0	Amazon SQS key (SSE-SQS)	-
○	SQS_S3bucket_Mail	Standard	2023-06-23T14:30-03:00	0	0	Amazon SQS key (SSE-SQS)	-

Figure 34: Message got deleted in the queue

References

[1] "Sending and receiving messages in Amazon SQS - Boto3 1.28.9 documentation," *Amazonaws.com*. [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs-example-sending-receiving-msgs.html. [Accessed: 22-Jul-2023].

[2] "SNS - Boto3 1.28.9 documentation," *Amazonaws.com*. [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sns.html. [Accessed: 22-Jul-2023].

[3] M. Ozkaya, "Amazon SQS Queue Polling From AWS Lambda with Event Source Mapping Invocations," *AWS Lambda & Serverless — Developer Guide with Hands-on Labs*, 06-Oct-2022. [Online]. Available: https://medium.com/aws-lambda-serverless-developer-guide-with-hands/amazon-sqs-queue-polling-from-aws-lambda-with-event-source-mapping-invocations-2bf34108cef0. [Accessed: 22-Jul-2023].

[4] A. Maksimov, "Boto3 SQS - complete tutorial 2023 2023," *Hands-On.Cloud*, 02-Sep-2021. [Online]. Available: https://hands-on.cloud/boto3-sqs-tutorial/. [Accessed: 22-Jul-2023].

[5] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-create-rule.html. [Accessed: 22-Jul-2023].

[6] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/step-create-queue.html. [Accessed: 22-Jul-2023].

[7] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html. [Accessed: 22-Jul-2023].

[8] R. Python, "Natural language processing with spaCy in Python," *Realpython.com*, 02-Jan-2023. [Online]. Available: https://realpython.com/natural-language-processing-spacy-python/. [Accessed: 22-Jul-2023].

[9] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/code-library/latest/ug/python_3_dynamodb_code_examples.html. [Accessed: 22-Jul-2023].

[10] "SpaCy · industrial-strength Natural Language Processing in Python," *Spacy.io*. [Online]. Available: https://github.com/minway/NlpTest. [Accessed: 22-Jul-2023].

[11] *NlpTest: Test CoreNLP on AWS Lambda*. Available: https://spacy.io/. [Accessed: 22-Jul-2023].

[12] "*Lucid visual collaboration suite: Log in*," *Lucid.app*. [Online]. Available: https://lucid.app/documents#/documents?folder_id=recent. [Accessed: 22-Jul-2023].