
CSCI 5410 Serverless Data Processing

Assignment 3 Report

Prepared by:
Vikram Venkatapathi - B00936916

Master of Applied Computer Science (Summer'23)
Faculty of Computer Science
Dalhousie University

GitLab Repo link : <https://git.cs.dal.ca/vikramv/csci5410-summer-23-b00936916/-/tree/A3>

Contents

Table of Contents	2
I. PART A	1
1. Introduction	3
2. Hypothetical Scenario and the Use Case	3
2.1. AWS Architecture diagram	4
II. PART B	6
3. NOTE	8
3.1. Usage of AWS account	8
4. Procedure followed for the given experiment	8
4.1. AWS Architecture diagram	8
4.2. Steps	8
4.3. Lamda code	14
4.3.1. extractFeatures	14
4.3.2. accessDB	15
4.3.3. Screenshots	17
4.4. Testing	31
4.4.1. Procedure	31
4.4.2. Screenshots	32
III. PART C	37
5. Car Service	39
5.1. AWS Architecture diagram of my implementation	39
5.2. Steps	39
5.3. Lambda Code	44
5.3.1. sendOrderDetailsToSNS	44
5.3.2. sendOrderDetailsToHalifaxTaxi	45
5.4. Testing	47
5.5. Screenshots	48

Part I.

PART A

Explore & Build a Use Case

1. Introduction

As a former cashier at the Atlantic Superstore[2], I marveled at the complexities of inventory management. This inspired me to envision a scenario where AWS Kinesis[1] revolutionizes the process, ensuring precise stock numbers and minimizing wastage. In this hypothetical scenario, real-time data streams from IoT devices and point-of-sale systems are leveraged to optimize inventory, prompt reordering, and deliver an exceptional shopping experience for customers.

2. Hypothetical Scenario and the Use Case

Hypothetical Scenario: Smart Inventory Management at Atlantic Superstore

In this hypothetical scenario, the Atlantic Superstore[2], a large retail chain, faces challenges with manual inventory management, leading to inefficiencies and occasional stockouts. The store envisions implementing an intelligent inventory management system using AWS Kinesis[1] and other AWS services to address these issues.

Use case:

The Atlantic Superstore decides to leverage AWS Kinesis[1] for real-time data streaming to capture continuous updates from IoT devices and point-of-sale systems placed strategically across the store. These devices will transmit product movement, stock levels, and sales data in real-time to AWS Kinesis Data Streams[15]. With AWS Kinesis Data Analytics[16], the store processes and analyzes the streaming data on the fly. Advanced analytics and machine learning algorithms are applied to forecast demand, identify trending products, and detect potential stock shortages. The system can instantly update inventory records and provide a comprehensive view of stock levels. To ensure efficient stock management, the Atlantic Superstore integrates AWS Lambda functions[17] into the system. These functions automatically trigger reordering processes for low-stock items, streamlining inventory replenishment and preventing stockouts. Additionally, the store incorporates AWS Comprehend[18] for sentiment analysis to understand customer preferences and feedback. By integrating customer sentiment with sales data, the store can optimize its inventory based on real-time demand and adapt to changing customer needs. The smart inventory management system using AWS Kinesis allows the Atlantic Superstore to optimize inventory levels, reduce wastage, and deliver a seamless shopping experience for its customers. With real-time insights and automated processes, the store can maintain accurate stock numbers, minimize manual efforts, and ensure products are readily available for customers, ultimately driving increased customer satisfaction and operational efficiency.

2.1. AWS Architecture diagram

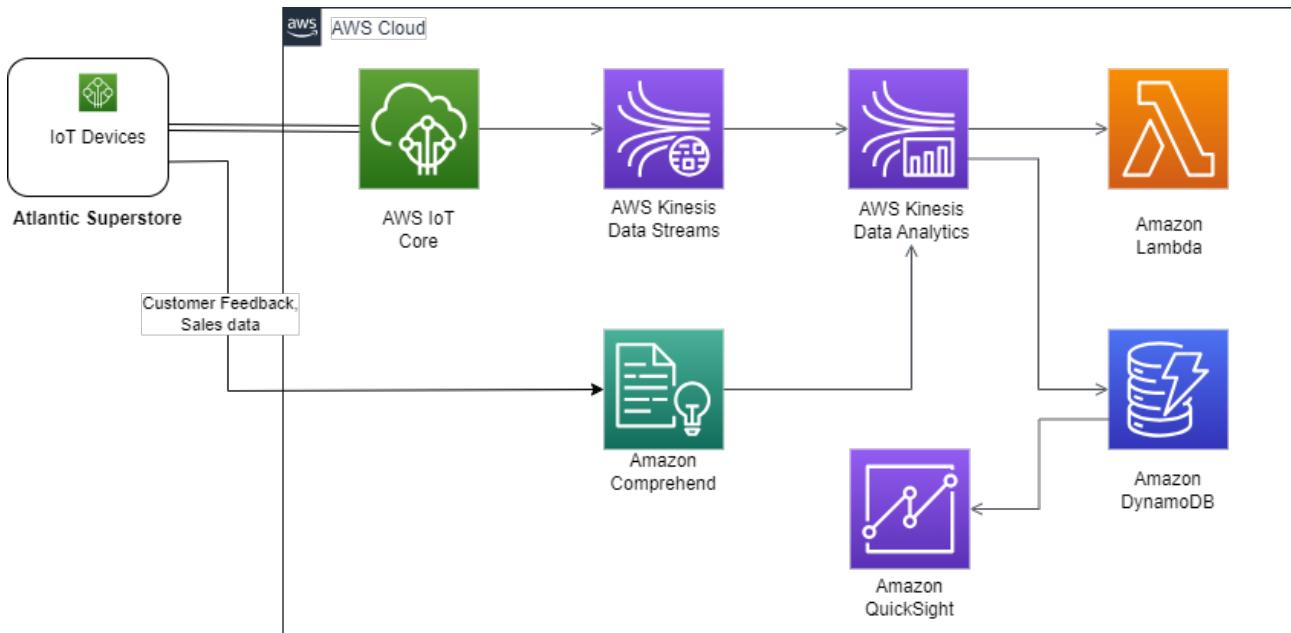


Figure 2.1.: *Smart Inventory Management Architecture at Atlantic Superstore using AWS Services* (Created using: draw.io [23])

1. **IoT Devices:** These devices are strategically placed on shelves and equipped with sensors to detect product movement and measure stock levels.
2. **AWS IoT Core:** The IoT devices securely connect and transmit data to AWS Kinesis Data Streams, ensuring smooth communication with the cloud.[14]
3. **AWS Kinesis Data Streams:** This service captures and streams real-time data from the IoT devices, including product IDs, timestamps, stock levels, and sales data.[15]
4. **AWS Kinesis Data Analytics:** The streaming data is processed and analyzed in real-time to forecast demand, identify trending products, and detect potential stock shortages. It also triggers Amazon Lambda functions for automatic reordering and anomaly detection.[16]
5. **Amazon Lambda:** Lambda functions automatically trigger reordering processes for low-stock items and flag any irregularities in stock movements.[17]
6. **Amazon Comprehend:** This service is used for sentiment analysis, integrating customer feedback with sales data to understand customer preferences and optimize inventory accordingly.[18]
7. **Amazon DynamoDB:** The analyzed data is stored in DynamoDB, a NoSQL database, offering fast and scalable access to historical inventory data.[19]

8. **Amazon QuickSight:** The data in DynamoDB is visualized and analyzed through QuickSight, enabling the store to create interactive dashboards and gain real-time insights into stock levels, popular products, and inventory performance.[20]

Part II.

PART B

Build an event-driven serverless application using AWS Lambda

3. NOTE

3.1. Usage of AWS account

On the account of Novelty, I attempted to create all the services required for this assignment, using the Serverless Framework[3]. Running this YAML file creates an IAM role for logging purposes. Since I don't have permission to create an IAM role in the AWS academy account, I have used my personal AWS account to deploy the services for this entire assignment.

4. Procedure followed for the given experiment

4.1. AWS Architecture diagram

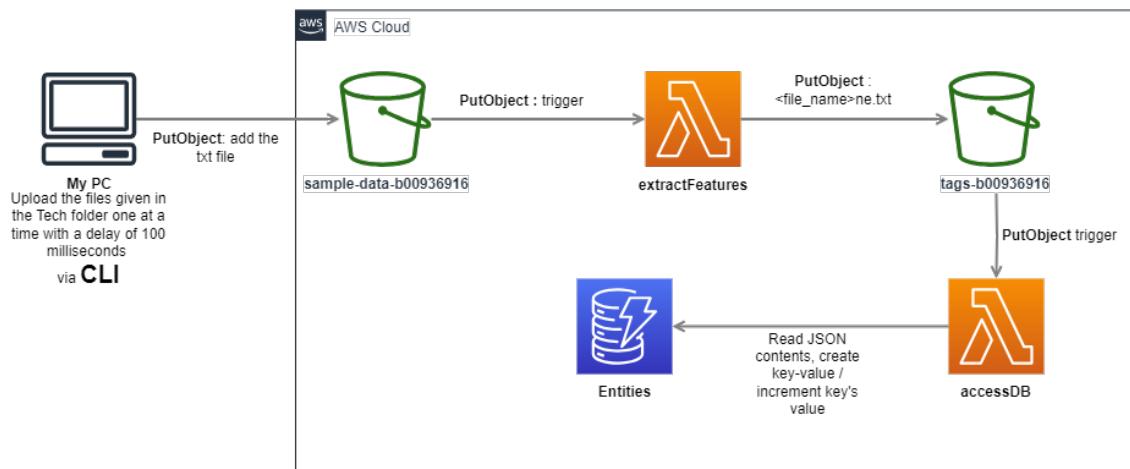


Figure 4.1.: *AWS Architecture diagram for the given work flow (Created using: draw.io [23])*

4.2. Steps

1. Configure the AWS credentials in the CLI using the command "**aws configure**".

2. Create an IAM role with the following permission provided by AWS, Role_Name = **lambdaRole**: [21]
 - a) AWSLambdaExecute: Full access to CloudWatch Logs; Read, Write access to S3
3. Create an IAM role with the following permissions provided by AWS, Role_Name = **lambdaRole_accessDB**: [21]
 - a) AWSLambdaExecute: Full access to CloudWatch Logs; Read, Write access to S3
 - b) AmazonDynamoDBFullAccess (as the name suggests)
4. Role = lambdaRole is for lambda extractFeatures
5. Role = lambdaRole_accessDB is for lambda accessDB
6. Run the following command to create a serverless-framework project

```
# Create a new AWS Python3 Serverless project at path 'event-driven-serverless-app-a3'.
serverless create --template aws-python3 --path event-driven-serverless-app-a3
```

7. Create the required resources using the following serverless framework script[3] (serverless.yml)[4]

```
service: event-driven-serverless-app-a3

provider:
  name: aws
  runtime: python3.8
  stage: dev
  region: us-east-1

resources:
  Resources:
    # 1st S3 Bucket
    SampleDataBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: sample-data-b00936916 # Actual Bucket Name :
          SampleDataB00936916 -> Bucket name should not contain
          uppercase characters

    # 2nd S3 Bucket
    TagsBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: tags-b00936916 # Actual Bucket Name : TagsB00936916
          -> Bucket name should not contain uppercase characters
```

```

# Lambda Function to Extract Named Entities
ExtractFeaturesLambda:
  Type: AWS::Lambda::Function
  Properties:
    FunctionName: extractFeatures
    Code:
      ZipFile: | #sample code
      def lambda_handler(event, context):
        # TODO implement
        return {
          'statusCode': 200,
          'body': json.dumps('Hello from Lambda!')
        }
    Handler: app.lambda_handler
    Runtime: python3.8    # Specify the runtime for the Lambda
                           function
    MemorySize: 512
    Timeout: 10
    Role: arn:aws:iam::000966082997:role/lambdaRole #role with
                                                       permissions : AWSLambdaExecute, AmazonDynamoDBFullAccess

# Lambda Function to Access DynamoDB and Update
AccessDBLambda:
  Type: AWS::Lambda::Function
  Properties:
    FunctionName: accessDB
    Code:
      ZipFile: | #sample code
      def lambda_handler(event, context):
        # TODO implement
        return {
          'statusCode': 200,
          'body': json.dumps('Hello from Lambda!')
        }
    Handler: app.lambda_handler
    Runtime: python3.8    # Specify the runtime for the Lambda
                           function
    MemorySize: 512
    Timeout: 10
    Role: arn:aws:iam::000966082997:role/lambdaRole_accessDB #role
                                                               with permissions : AWSLambdaExecute, AmazonDynamoDBFullAccess

# DynamoDB Table
DynamoDBTable:
  Type: AWS::DynamoDB::Table
  Properties:
    TableName: Entities

```

```

AttributeDefinitions:
  - AttributeName: entity_name
    AttributeType: S # Set the AttributeType to String for the
      partition key "entity_name"
KeySchema:
  - AttributeName: entity_name # Use file_name as the partition
    key
    KeyType: HASH # Set the KeyType to HASH for the partition
      key "entity_name"
ProvisionedThroughput:
  ReadCapacityUnits: 1
  WriteCapacityUnits: 1

```

8. The above script does the following
 - a) Create 2 - S3 buckets: sample-data-b00936916, tags-b00936916
 - b) Create 2 - lambdas: extractFeatures(with IAM role: lambdaRole), accessDB(with IAM role: lambdaRole_accessDB)
 - c) Create 1 - DynamoDB table: Entities (with Partition key: entity_name)
9. Run the command **serverless deploy** to deploy the .yml file to create the resources in CloudFormation
10. Run the following commands to **add the InvokeFunction permission** to both the lambdas via **CLI** [5]

```

Lambda_1:

Command = aws lambda add-permission --function-name arn:aws:lambda:us-
  east-1:000966082997:function:extractFeatures --action lambda:
    InvokeFunction --principal 000966082997 --source-arn arn:aws:s3:::
      sample-data-b00936916 --statement-id s3_trigger

Lambda_2:

Command = aws lambda add-permission --function-name arn:aws:lambda:us-
  east-1:000966082997:function:accessDB --action lambda:InvokeFunction
    --principal 000966082997 --source-arn arn:aws:s3:::tags-b00936916 --
      statement-id s3_trigger

```

11. Now, create separate JSON files with the below contents, to specify the LambdaFunctionConfigurations, to add S3 event notification.[7]
 - a) **sample-data-b00936916** bucket (trigger_configuration_sample-data-b00936916.json)

```
{
  "LambdaFunctionConfigurations": [
    {
      "LambdaFunctionArn": "arn:aws:lambda:us-east-1:000966082997:
        function:extractFeatures",
      "Events": ["s3:ObjectCreated:*"]
    }
  ]
}
```

- b) tags-b00936916 bucket (trigger_configuration_tags-b00936916.json)

```
{
  "LambdaFunctionConfigurations": [
    {
      "LambdaFunctionArn": "arn:aws:lambda:us-east-1:000966082997:
        function:accessDB",
      "Events": ["s3:ObjectCreated:*"]
    }
  ]
}
```

12. Now, run the following commands to add the S3 event notification to each bucket[11], **specifying the path** to the above created json file using `file://</path/to/file_name>.json` [6]

Bucket1:

```
Command = aws s3api put-bucket-notification-configuration --bucket
sample-data-b00936916 --notification-configuration file://
trigger_configuration_sample-data-b00936916.json
```

Bucket2:

```
Command = aws s3api put-bucket-notification-configuration --bucket tags-
b00936916 --notification-configuration file://
trigger_configuration_tags-b00936916.json
```

13. Download the tech.zip file, extract it, change directory to folder 'tech' [22], open the terminal, run the following script[12]

```
foreach ($file in Get-ChildItem -File) {
    aws s3 cp "$file" s3:/sample-data-b00936916/
    Start-Sleep -Milliseconds 100
}
```

14. Now, all the 401 files will be uploaded with a 100 milliseconds delay.
15. Upon each upload, the flow of events mentioned in the architecture will be triggered, and all the names entities will be added to the 'Entities' dynamoDB databases, with the **key as the 'entity_name'** and **value as its frequency**.

4.3. Lambda code

4.3.1. extractFeatures

```
import json
import re
import boto3

def lambda_handler(event, context):
    # Retrieve the S3 event information
    records = event['Records']
    for record in records:
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']

    """
        Boto3 - Amazon Simple Storage Service (S3) API Documentation
        Reference: `Boto3 - Amazon S3 API <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>`_
    """

    # Read the contents of the file from S3
    s3_client = boto3.client('s3')
    response = s3_client.get_object(Bucket=bucket, Key=key)
    file_content = response['Body'].read().decode('utf-8')

    """
        Python Standard Library - re (Regular Expression) module
        Reference: `Python re (Regular Expression) module <https://docs.python.org/3/library/re.html>`_
    """

    # Extract words starting with a capital letter
    capital_words = re.findall(r'\b[A-Z][a-zA-Z]*\b', file_content)

    # Extract all caps words
    all_caps_words = re.findall(r'\b[A-Z]+\b', file_content)

    # Create a dictionary to store the named entities and their counts
    named_entities = {}

    # Count the occurrences of each word
    for word in capital_words + all_caps_words:
        named_entities[word] = named_entities.get(word, 0) + 1

    # Remove the file extension from the key
    file_name = key.split('.')[0]
```

```

# Create the JSON object with the key as the file name and the named
# entities dictionary as the value
result = {f"{file_name}ne": named_entities}

# Convert the result to a JSON string
named_entities_json = json.dumps(result)

destination_s3_bucket = "tags-b00936916"

# Upload the JSON to the tags-b00936916 bucket
s3_client.put_object(Bucket=destination_s3_bucket, Key=f"{file_name}ne.txt", Body=named_entities_json)

return {
    'statusCode': 200,
    'body': json.dumps('Named entities extracted successfully.')
}

"""

All references:
Python Standard Library - re (Regular Expression) module
Reference: `Python re (Regular Expression) module <https://docs.python.org/3/library/re.html>`_

Boto3 - Amazon Simple Storage Service (S3) API Documentation
Reference: `Boto3 - Amazon S3 API <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>`_

"""

```

4.3.2. accessDB

```

import json
import boto3

def lambda_handler(event, context):
    # Retrieve the S3 event information
    records = event['Records']
    for record in records:
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']

        # Check if the file is in the format "001ne.txt"

```

```

if key.endswith("ne.txt"):

    """
        Boto3 - Amazon Simple Storage Service (S3) API Documentation
        Reference: `Boto3 - Amazon S3 API <https://boto3.amazonaws.
                    com/v1/documentation/api/latest/reference/services/s3.
                    html>`_
    """

    # Read the contents of the file from S3
    s3_client = boto3.client('s3')

    response = s3_client.get_object(Bucket=bucket, Key=key)
    file_content = response['Body'].read().decode('utf-8')

    # Convert the JSON content to a dictionary
    named_entities = json.loads(file_content)

    # Get the file name from the key (e.g., "001ne.txt" -> "001ne")
    file_name = key.split('.')[0]

    """
        Boto3 - Amazon DynamoDB API Documentation
        Reference: `Boto3 - Amazon DynamoDB API <https://boto3.
                    amazonaws.com/v1/documentation/api/latest/reference/
                    services/dynamodb.html>`_
    """

    # Update the DynamoDB table with the named entities
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('Entities')

    for entity, count in named_entities[file_name].items():
        # Convert the count to an integer before updating the table
        count = int(count)
        # Update the table with the named entity and its count
        table.update_item(
            Key={'entity_name': entity},
            UpdateExpression='SET frequency = if_not_exists(
                frequency, :zero) + :val',
            ExpressionAttributeValues={':zero': 0, ':val': count}
        )

    return {
        'statusCode': 200,
        'body': json.dumps('DynamoDB table updated successfully.')
    }

"""

```

All references:

Boto3 - Amazon Simple Storage Service (S3) API Documentation

Reference: `Boto3 - Amazon S3 API <<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>>`_

Boto3 - Amazon DynamoDB API Documentation

Reference: `Boto3 - Amazon DynamoDB API <<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html>>`_

"""

4.3.3. Screenshots

The screenshot shows the Visual Studio Code interface. The left sidebar displays a project structure with files like serverless.yml, serverless-app.yaml, tech (1).zip, A3_S23_Summer.pdf, and README.md. The main editor area shows the content of serverless.yml:

```
# Lambda Function to Access DynamoDB and Update
AccessDBLambda:
  Type: AWS::Lambda::Function
  Properties:
    FunctionName: accessDB
    Code:
      Zipfile: |
        def lambda_handler(event, context):
          # TODO implement
          return {
            'statusCode': 200,
            'body': json.dumps('Hello from Lambda!')
          }
    Handler: app.handler
    Runtime: python3.8 # Specify the runtime for the Lambda function
    MemorySize: 512
    Timeout: 10
```

The terminal tab at the bottom shows the command being run:

```
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csc15410-summer-23-b00936916\Part B\serverless_framework> serverless deploy --stage dev --region us-east-1
```

Output from the terminal shows the deployment status:

```
Deploying event-driven-serverless-app-a3 to stage dev (us-east-1)
✓ Service deployed to stack event-driven-serverless-app-a3-dev (43s)
```

A green arrow points to the terminal output line "✓ Service deployed to stack event-driven-serverless-app-a3-dev (43s)".

Figure 4.2.: *Running Serverless-Framework script in CLI*

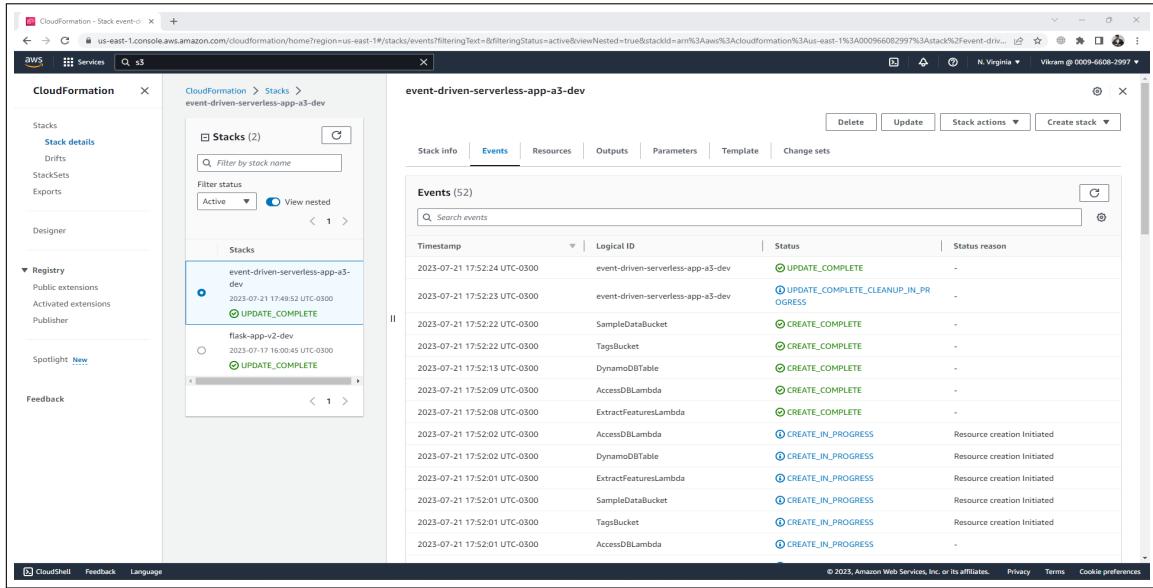


Figure 4.3.: *Resources stack created - CloudFormation in Console*

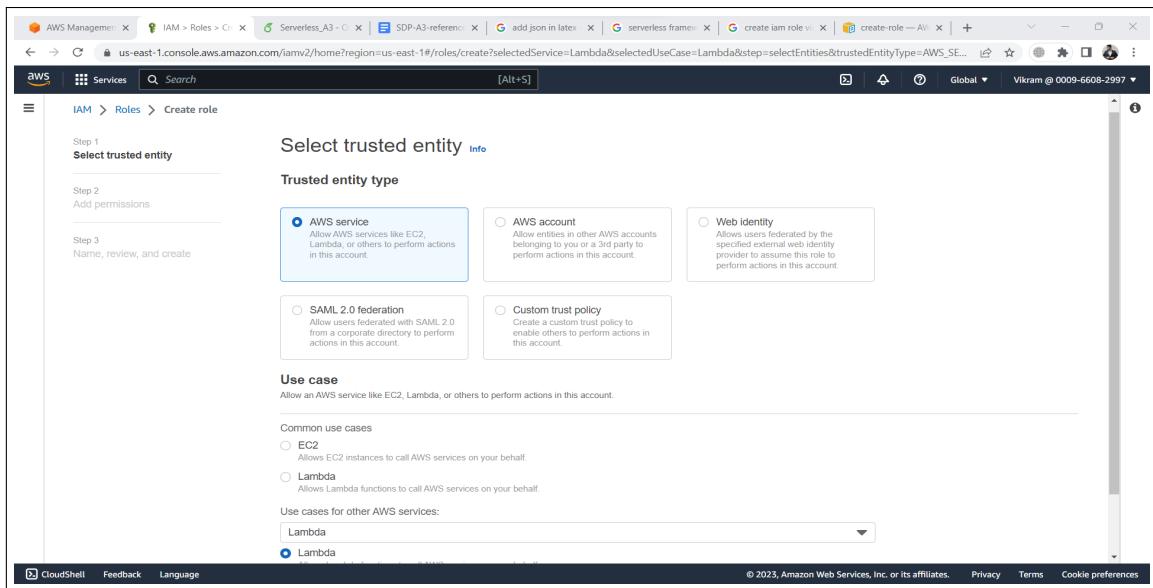


Figure 4.4.: *Create IAM role for lambda extractFeatures*

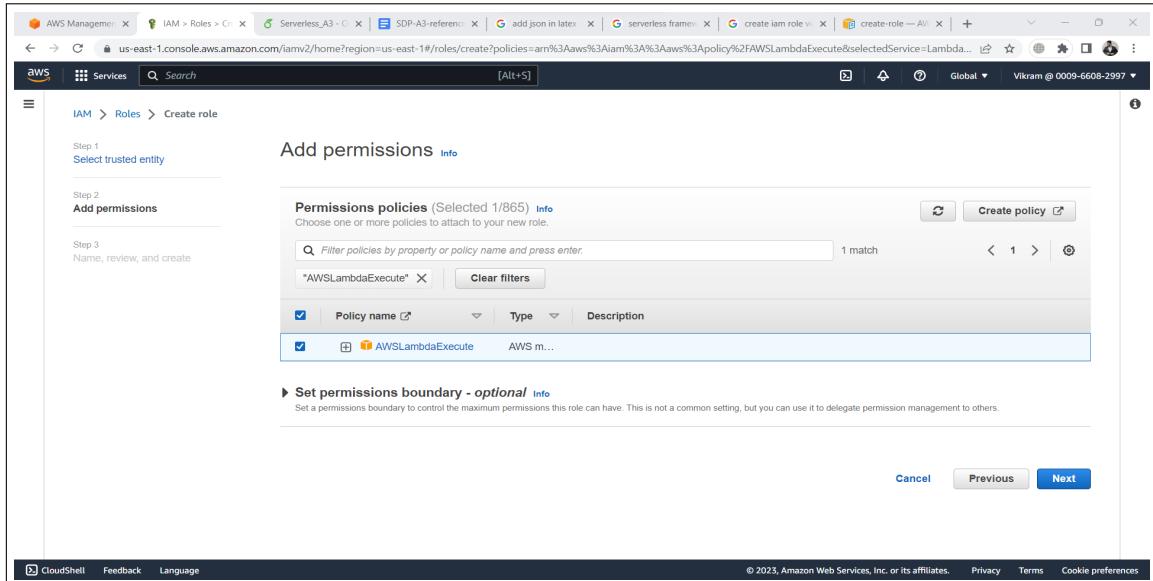


Figure 4.5.: *Create IAM role for lambda extractFeatures*

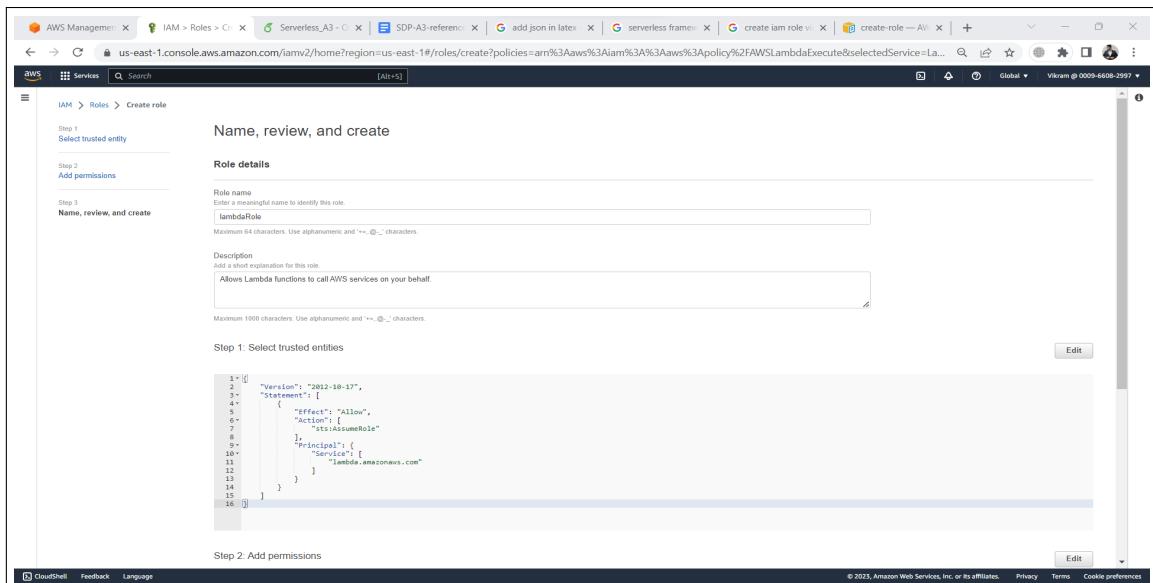


Figure 4.6.: *Create IAM role for lambda extractFeatures*

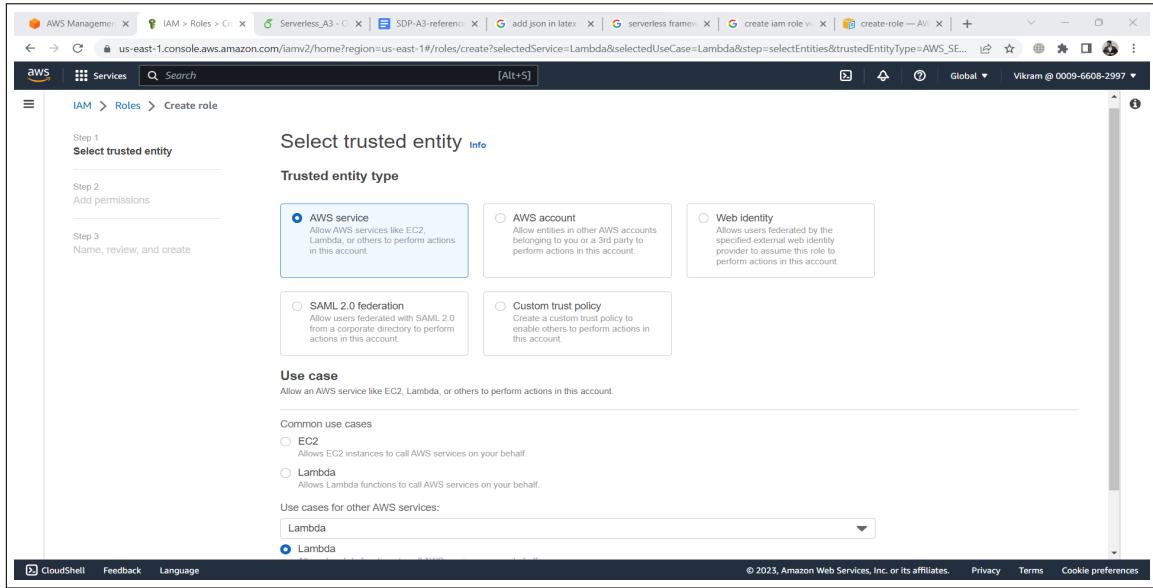


Figure 4.7.: *Create IAM role for lambda accessDB*

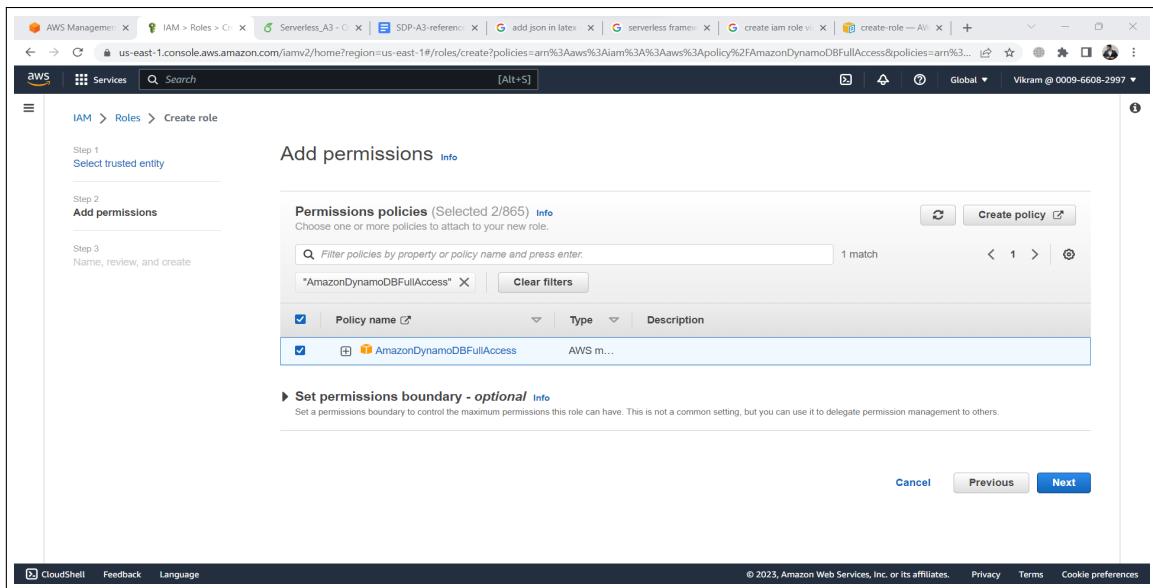


Figure 4.8.: *Create IAM role for lambda accessDB*

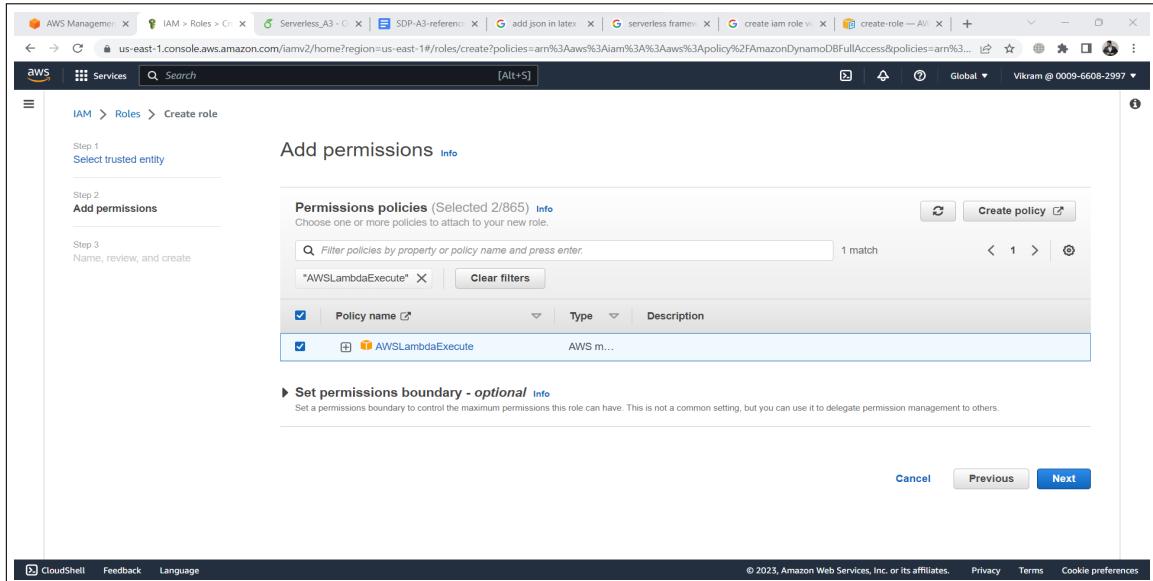


Figure 4.9.: *Create IAM role for lambda accessDB*

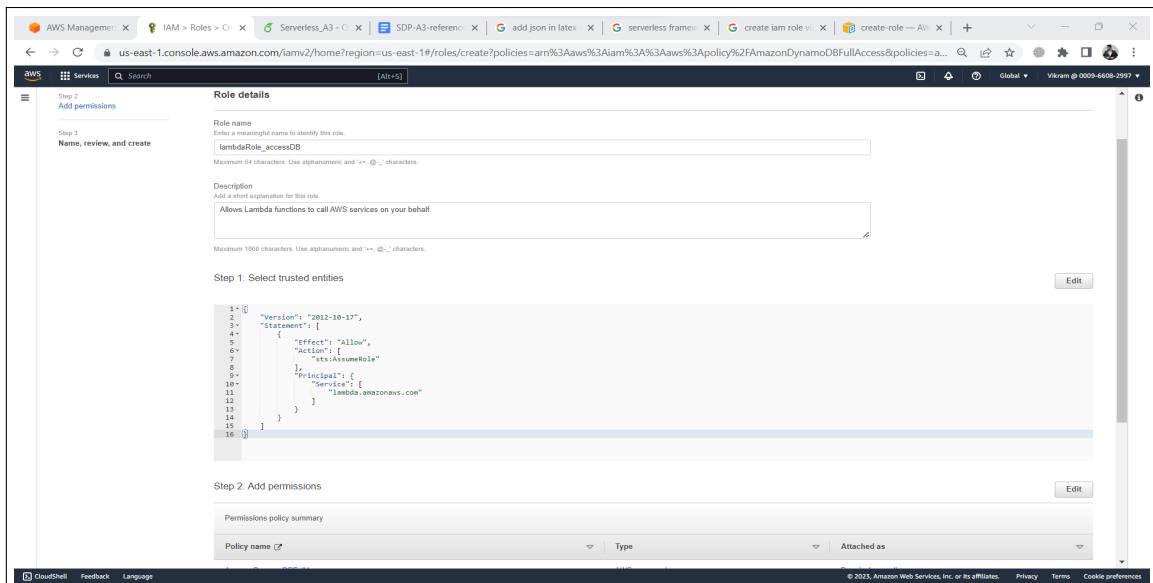


Figure 4.10.: *Create IAM role for lambda accessDB*

The screenshot shows the AWS Management Console with the IAM > Roles page open. The left sidebar shows navigation options like Dashboard, Access management, and Access reports. The main content area lists two roles:

Role Name	Description	Last Updated
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	-
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	1 hour ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
ConfirmSubscriptionTrigger-role-pzwm21p9	AWS Service: lambda	21 minutes ago
flask-app-v2-dev-EnterpriseLogAccessIamRole-1QH7DT97HDXEX	Account: 802587217904	-
flask-app-v2-dev-us-east-1-lambdaRole	AWS Service: lambda	2 days ago
lambdaRole	AWS Service: lambda	1 hour ago
lambdaRole_accessDB	AWS Service: lambda	-
snsDeliveryStatus	AWS Service: sns	-
snsDeliveryStatusFailure	AWS Service: sns	-
SNSsendMsgToSQS-role-68u17om1	AWS Service: lambda	3 days ago

Figure 4.11.: *Both the Roles created*

```

PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B\tech> foreach ($file in Get-ChildItem -File) {
>> aws s3 cp "$file" s3://sample-data-b00936916/
>> Start-Sleep -Milliseconds 100
>>
upload: \001.txt to s3://sample-data-b00936916/001.txt
upload: \002.txt to s3://sample-data-b00936916/002.txt
upload: \003.txt to s3://sample-data-b00936916/003.txt
upload: \004.txt to s3://sample-data-b00936916/004.txt
upload: \005.txt to s3://sample-data-b00936916/005.txt
upload: \006.txt to s3://sample-data-b00936916/006.txt
upload: \007.txt to s3://sample-data-b00936916/007.txt
upload: \008.txt to s3://sample-data-b00936916/008.txt
upload: \009.txt to s3://sample-data-b00936916/009.txt
upload: \010.txt to s3://sample-data-b00936916/010.txt
upload: \011.txt to s3://sample-data-b00936916/011.txt
upload: \012.txt to s3://sample-data-b00936916/012.txt
upload: \013.txt to s3://sample-data-b00936916/013.txt
upload: \014.txt to s3://sample-data-b00936916/014.txt
upload: \015.txt to s3://sample-data-b00936916/015.txt
upload: \016.txt to s3://sample-data-b00936916/016.txt
upload: \017.txt to s3://sample-data-b00936916/017.txt
upload: \018.txt to s3://sample-data-b00936916/018.txt
upload: \019.txt to s3://sample-data-b00936916/019.txt
upload: \020.txt to s3://sample-data-b00936916/020.txt
upload: \021.txt to s3://sample-data-b00936916/021.txt
upload: \022.txt to s3://sample-data-b00936916/022.txt
upload: \023.txt to s3://sample-data-b00936916/023.txt
upload: \024.txt to s3://sample-data-b00936916/024.txt
upload: \025.txt to s3://sample-data-b00936916/025.txt
upload: \026.txt to s3://sample-data-b00936916/026.txt
upload: \027.txt to s3://sample-data-b00936916/027.txt
upload: \028.txt to s3://sample-data-b00936916/028.txt
upload: \029.txt to s3://sample-data-b00936916/029.txt
upload: \030.txt to s3://sample-data-b00936916/030.txt
upload: \031.txt to s3://sample-data-b00936916/031.txt
upload: \032.txt to s3://sample-data-b00936916/032.txt
upload: \033.txt to s3://sample-data-b00936916/033.txt
upload: \034.txt to s3://sample-data-b00936916/034.txt
upload: \035.txt to s3://sample-data-b00936916/035.txt

```

Figure 4.12.: *Upload the files in folder 'tech' to bucket sample-data-b00936916 via CLI*

```

Windows PowerShell
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B\tech> aws s3 sync . s3://sample-data-b00936916/tech --no-progress
upload: .\363.txt to s3://sample-data-b00936916/363.txt
upload: .\364.txt to s3://sample-data-b00936916/364.txt
upload: .\365.txt to s3://sample-data-b00936916/365.txt
upload: .\366.txt to s3://sample-data-b00936916/366.txt
upload: .\367.txt to s3://sample-data-b00936916/367.txt
upload: .\368.txt to s3://sample-data-b00936916/368.txt
upload: .\369.txt to s3://sample-data-b00936916/369.txt
upload: .\370.txt to s3://sample-data-b00936916/370.txt
upload: .\371.txt to s3://sample-data-b00936916/371.txt
upload: .\372.txt to s3://sample-data-b00936916/372.txt
upload: .\373.txt to s3://sample-data-b00936916/373.txt
upload: .\374.txt to s3://sample-data-b00936916/374.txt
upload: .\375.txt to s3://sample-data-b00936916/375.txt
upload: .\376.txt to s3://sample-data-b00936916/376.txt
upload: .\377.txt to s3://sample-data-b00936916/377.txt
upload: .\378.txt to s3://sample-data-b00936916/378.txt
upload: .\379.txt to s3://sample-data-b00936916/379.txt
upload: .\380.txt to s3://sample-data-b00936916/380.txt
upload: .\381.txt to s3://sample-data-b00936916/381.txt
upload: .\382.txt to s3://sample-data-b00936916/382.txt
upload: .\383.txt to s3://sample-data-b00936916/383.txt
upload: .\384.txt to s3://sample-data-b00936916/384.txt
upload: .\385.txt to s3://sample-data-b00936916/385.txt
upload: .\386.txt to s3://sample-data-b00936916/386.txt
upload: .\387.txt to s3://sample-data-b00936916/387.txt
upload: .\388.txt to s3://sample-data-b00936916/388.txt
upload: .\389.txt to s3://sample-data-b00936916/389.txt
upload: .\390.txt to s3://sample-data-b00936916/390.txt
upload: .\391.txt to s3://sample-data-b00936916/391.txt
upload: .\392.txt to s3://sample-data-b00936916/392.txt
upload: .\393.txt to s3://sample-data-b00936916/393.txt
upload: .\394.txt to s3://sample-data-b00936916/394.txt
upload: .\395.txt to s3://sample-data-b00936916/395.txt
upload: .\396.txt to s3://sample-data-b00936916/396.txt
upload: .\397.txt to s3://sample-data-b00936916/397.txt
upload: .\398.txt to s3://sample-data-b00936916/398.txt
upload: .\399.txt to s3://sample-data-b00936916/399.txt
upload: .\400.txt to s3://sample-data-b00936916/400.txt
upload: .\401.txt to s3://sample-data-b00936916/401.txt

```

Figure 4.13.: *Upload the files in folder 'tech' to bucket sample-data-b00936916 via CLI - completed*

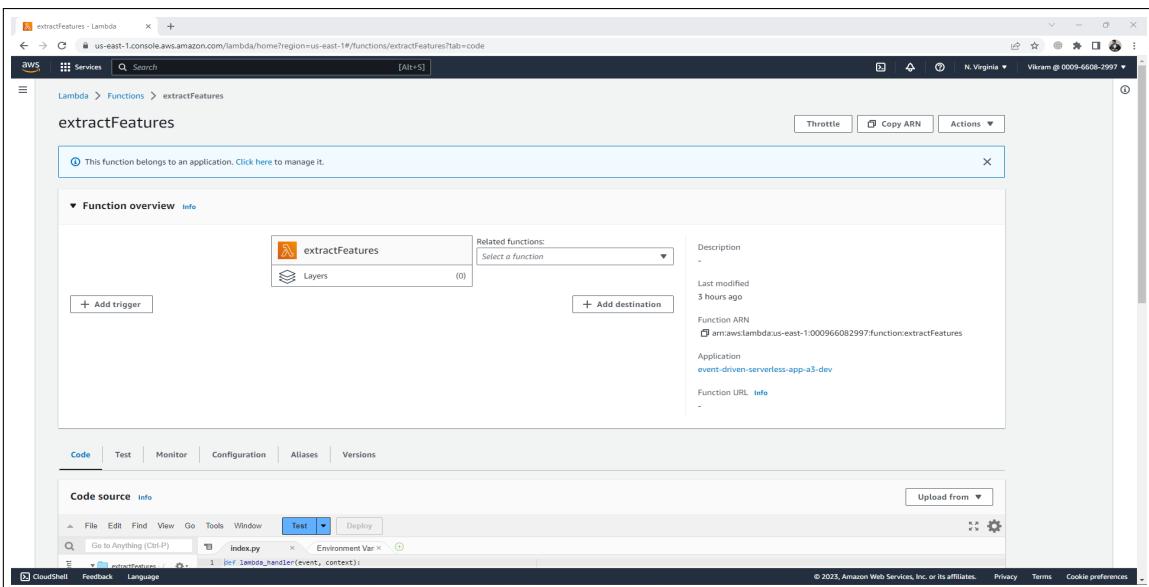


Figure 4.14.: *extractFeatures lambda before adding trigger*

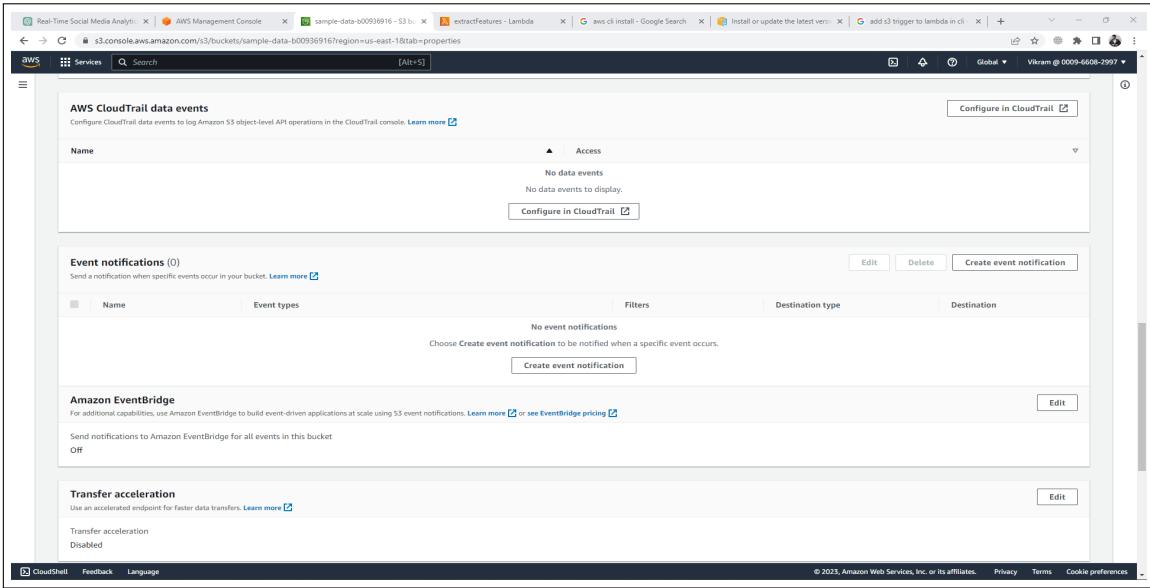


Figure 4.15.: *sample-data-b00936916 before adding event notification*

```
Windows PowerShell
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B> aws lambda add-permission --function-name arn:aws:lambda:us-east-1:000966082997:function:extractFeatures --action Lambda:InvokeFunction --principal 000966082997 --source-arn arn:aws:s3:::sample-data-b00936916 --statement-id s3_trigger
{
    "Statement": "[{"Sid": "s3_trigger", "Effect": "Allow", "Principal": {"AWS": "arn:aws:iam::000966082997:root"}, "Action": "lambda:InvokeFunction", "Resource": "arn:aws:lambda:us-east-1:000966082997:function:extractFeatures", "Condition": {"ArnLike": {"AWS:SourceArn": "arn:aws:s3:::sample-data-b00936916/*"}}}]"
}
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B>
```

Figure 4.16.: *add InvokeFunction permission for lambda sample-data-b00936916 via CLI*

```

Windows PowerShell
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B> aws s3api put-bucket-notification-configuration --bucket sample-data-b00936916 --notification-configuration file:/trigger_configuration.json
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B>

```

Figure 4.17.: *Add s3 trigger via CLI for extractFeatures lambda*

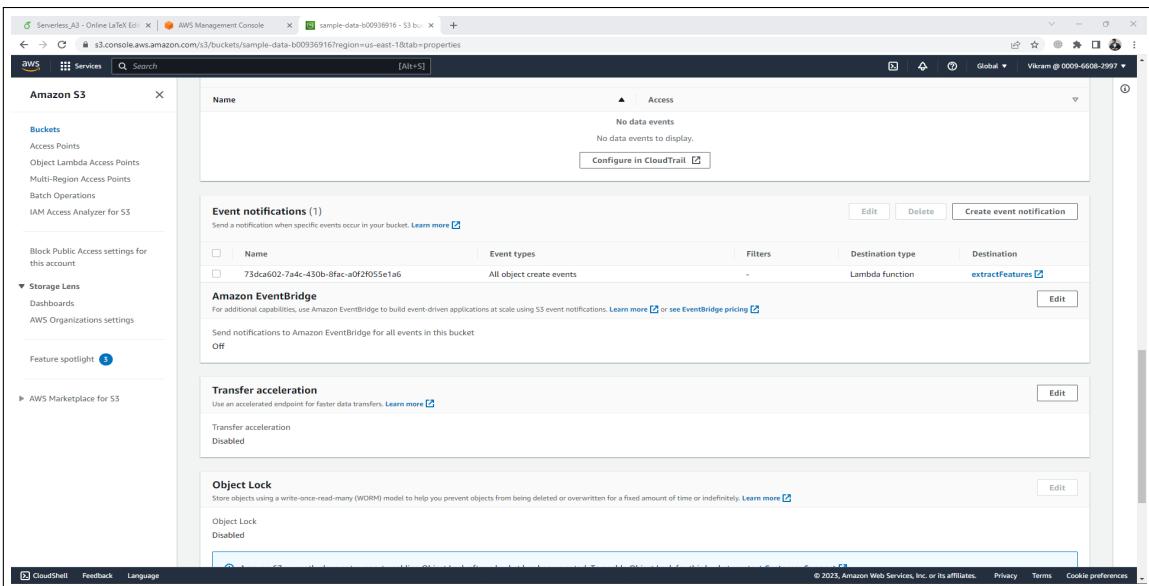


Figure 4.18.: *S3 event notification - bucket sample-data-b00936916*

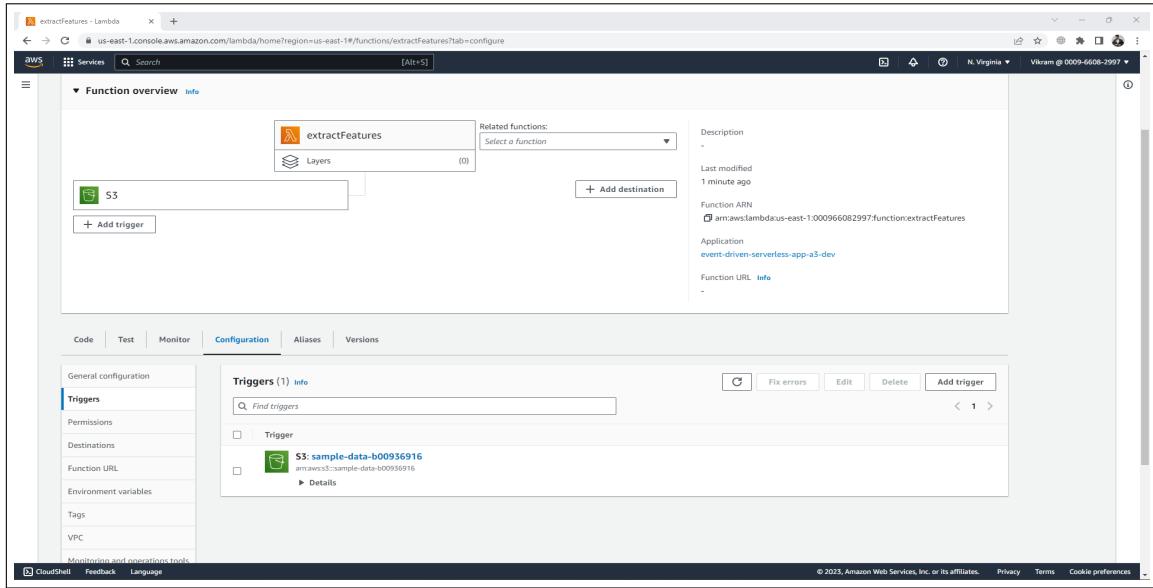


Figure 4.19.: *S3 trigger added for lambda extractFeatures*

```

PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B> aws lambda add-permission --function-name arn:aws:lambda:us-east-1:000966082997:function:accessDB --action lambda:InvokeFunction --principal 000966082997 --source-arn arn:aws:s3:::tags-b00936916 --statement-id s3_trigger
{
    "Statement": "{\"Sid\":\"s3_trigger\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"arn:aws:iam::000966082997:root\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"arn:aws:lambda:us-east-1:000966082997:function:accessDB\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:s3:::tags-b00936916\"}}}"
}
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B>

```

Figure 4.20.: *add InvokeFunction permission for lambda tags-b00936916 via CLI*

```

Windows PowerShell
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B> aws s3api put-bucket-notification-configuration --bucket tags-b00936916 --notification-configuration file://trigger_configuration_tags-b00936916.json
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part B>

```

Figure 4.21.: *Add s3 trigger via CLI for accessDB lambda*

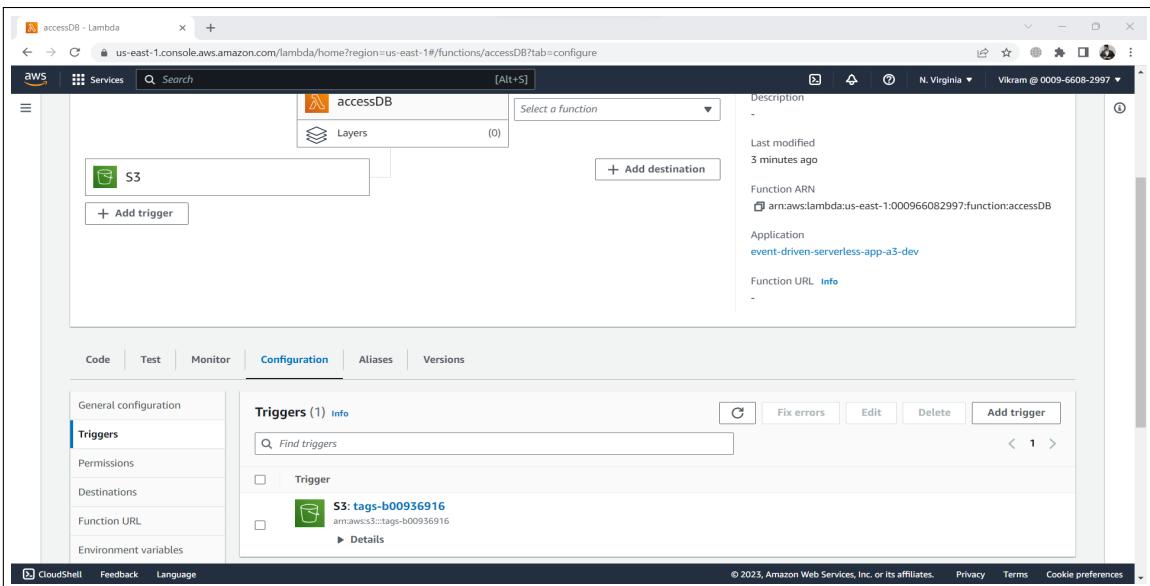


Figure 4.22.: *S3 trigger added for lambda accessDB*

The screenshot shows the AWS Management Console interface for managing an S3 bucket named 'tags-b00936916'. The left sidebar navigation includes 'Amazon S3', 'Buckets', 'Access Points', 'Multi-Region Access Points', 'Batch Operations', 'IAM 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 'AWS CloudTrail data events' with a table showing one entry: 'Event notifications (1)'. The table columns include 'Name', 'Event types', 'Filters', 'Destination type', and 'Destination'. The entry shows 'MWYxMzg2NjIzMjIyMjAwNzE3MDhOWFl' under 'Name', 'All object create events' under 'Event types', 'Lambda function' under 'Destination type', and 'accessDB' under 'Destination'. Below this, there is a section for 'Amazon EventBridge' with a note about additional capabilities and a link to EventBridge pricing. At the bottom, there is a 'Transfer acceleration' section with a note about using an accelerated endpoint for faster data transfers.

Figure 4.23.: *S3 event notification for bucket tags-b00936916*

The screenshot shows the AWS Management Console interface for managing a DynamoDB table named 'Entities'. The left sidebar navigation includes 'DynamoDB', 'Dashboard', 'Tables', 'Update settings', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Reserved capacity', 'Settings', 'DAX', 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main content area shows the 'Entities' table with a table header for 'Scan or query items'. Below the table, it says 'Items returned (0)' and 'No items'. There is a 'Create item' button at the bottom. The top navigation bar shows multiple open tabs for various AWS services like AWS Management, CloudFormation, IAM, S3 Management, and CloudWatch Metrics.

Figure 4.24.: *Empty DynamoDB table 'Entities'*

entity_name	frequency
Georgia	1
Often	2
System	1
UV	4
Executive	1
US	34
Explorer	1
Sun	22
IT	2
According	1
Concern	1
Startac	2

Figure 4.25.: *Final DynamoDB table 'Entities' after all entities are extracted*

Code SHA-256	Request ID
ZxSiIWXOlyK+h4wtGn1hmz2SpaGEonwNlFIS6fbSxhms=	52651e8a-7cad-4657-b99b-ccae17f7472f
Function version	Init duration
\$LATEST	261.57 ms
Duration	Billed duration
699.54 ms	700 ms
Resources configured	Max memory used
512 MB	75 MB

Figure 4.26.: *extractFeatures lambda unit test*

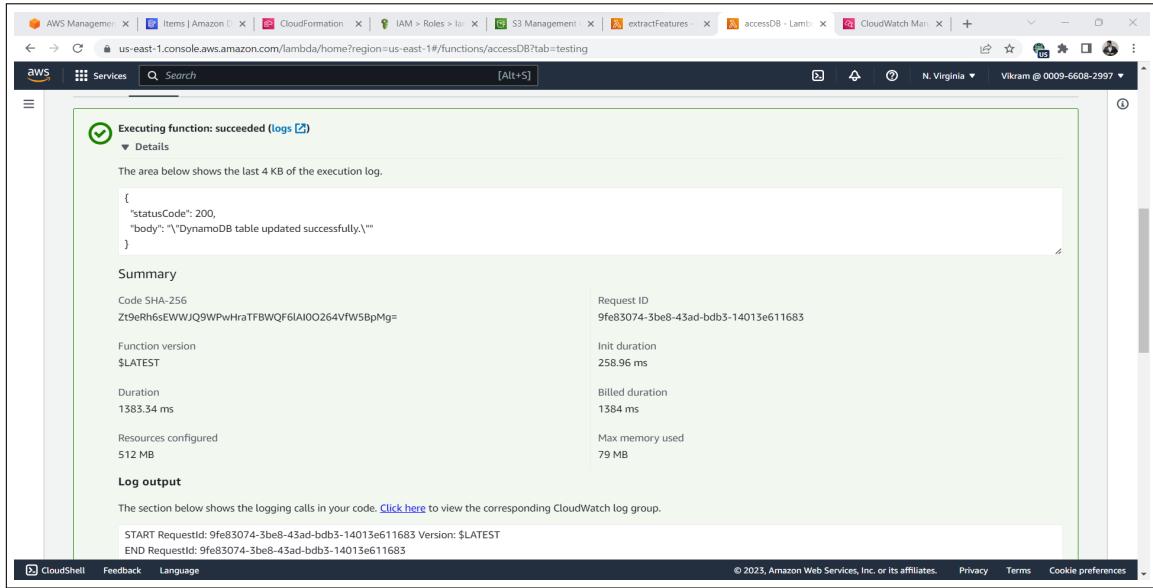


Figure 4.27.: *accessDB lambda unit test*

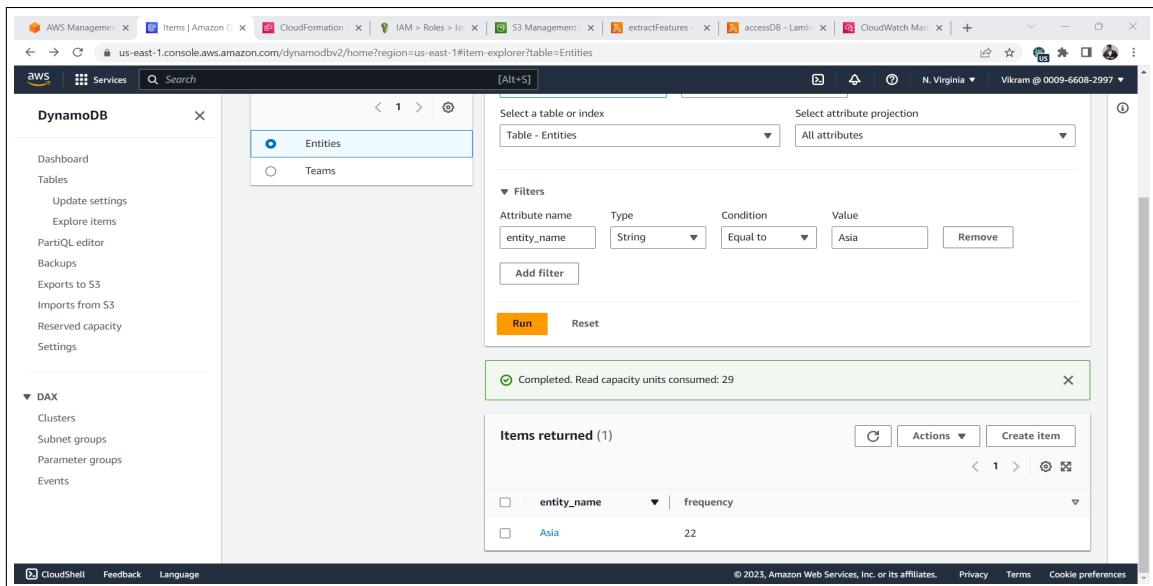


Figure 4.28.: *Sample query with entity_name="Asia"*

4.4. Testing

4.4.1. Procedure

1. Created a new file named '000.txt' with the following content:

```
sample txt file for testing purposes:  
  
all lowercase  
      : ggggggg  
all uppercase  
      : GGGGGGG  
Starting uppercase,  
other lowercase  
      : Ggggggg  
numbers  
      : 0000000
```

2. Named entities in this file are: GGGGGGG, Starting, Ggggggg
3. Uploaded the file via CLI.
4. Verified the file upload in bucket sample-data-b00936916 on console.
5. Verified the triggering of events by checking the logs of lambda functions: extractFeatures, accessDB.
6. Verified the file contents of generated '000ne.txt' file in bucket tags-b00936916 on console, by downloading the file.
7. Queried the extracted entities in DynamoDB and verified their frequency.

4.4.2. Screenshots

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'Part B' which contains a 'testing' folder. Inside 'testing', there is a file named '000.txt'. The content of '000.txt' is a sample text file for testing purposes, containing lowercase, uppercase, and numeric characters. The terminal tab shows the command 'aws s3 cp 000.txt s3://sample-data-b00936916/'. The output of this command shows the file being uploaded to the specified S3 bucket.

```
PS C:\Users\vikra\Desktop\AWS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csc15410-summer-23-b00936916\Part B\testing> aws s3 cp 000.txt s3://sample-data-b00936916/
upload: './000.txt' to s3://sample-data-b00936916/000.txt
PS C:\Users\vikra\Desktop\AWS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csc15410-summer-23-b00936916\Part B\testing> []
```

Figure 4.29.: *Upload 000.txt via CLI*

The screenshot shows the AWS Management Console with the 'Amazon S3' service selected. The left sidebar shows various AWS services like Lambda, CloudWatch, and CloudFront. The main area displays the contents of the 'sample-data-b00936916' bucket. Under the 'Objects' tab, a list of files is shown, including '000.txt' and other files from 001.txt to 014.txt. Each file entry includes its name, type (txt), last modified date, size, and storage class (Standard). The 'Actions' dropdown menu is visible above the list.

Name	Type	Last modified	Size	Storage class
000.txt	txt	July 24, 2023, 11:21:05 (UTC-05:00)	241.0 B	Standard
001.txt	txt	July 22, 2023, 01:14:27 (UTC-05:00)	3.9 KB	Standard
002.txt	txt	July 22, 2023, 01:14:28 (UTC-05:00)	2.2 KB	Standard
003.txt	txt	July 22, 2023, 01:14:29 (UTC-05:00)	1.3 KB	Standard
004.txt	txt	July 22, 2023, 01:14:30 (UTC-05:00)	2.5 KB	Standard
005.txt	txt	July 22, 2023, 01:14:31 (UTC-05:00)	4.8 KB	Standard
006.txt	txt	July 22, 2023, 01:14:32 (UTC-05:00)	3.8 KB	Standard
007.txt	txt	July 22, 2023, 01:14:33 (UTC-05:00)	1.7 KB	Standard
008.txt	txt	July 22, 2023, 01:14:34 (UTC-05:00)	1.7 KB	Standard
009.txt	txt	July 22, 2023, 01:14:35 (UTC-05:00)	7.0 KB	Standard
010.txt	txt	July 22, 2023, 01:14:36 (UTC-05:00)	2.9 KB	Standard
011.txt	txt	July 22, 2023, 01:14:37 (UTC-05:00)	3.3 KB	Standard
012.txt	txt	July 22, 2023, 01:14:38 (UTC-05:00)	1.5 KB	Standard
013.txt	txt	July 22, 2023, 01:14:39 (UTC-05:00)	3.0 KB	Standard
014.txt	txt	July 22, 2023, 01:14:40 (UTC-05:00)	1.7 KB	Standard

Figure 4.30.: *File uploaded in bucket sample-data-b00936916 - console*

The screenshot shows the AWS CloudWatch Log groups interface. On the left, a sidebar navigation includes CloudWatch, Favorites and recent, Dashboards, Alarms, Logs, Log groups, Log Insights, Metrics, X-Ray traces, Events, Application monitoring, and Insights. The main area displays a list of log entries for the 'extractFeatures' lambda function. The first entry is timestamped '2023-07-24T11:21:05,649+03:00' and contains the message 'INIT_START Runtime Version: python3.8.v28 Lambda Version ARN: arn:aws:lambda:us-east-1::runtime:73e44fb4eb122850ae23c2f747dccc0b09db03c04086f8acefc37800a73e0'. Subsequent entries show 'START RequestId: 6667b71a-8f76-4819-b435-a0638c2c26cd Version: \$LATEST', 'END RequestId: 6667b71a-8f76-4819-b435-a0638c2c26cd', and 'REPORT RequestId: 6667b71a-8f76-4819-b435-a0638c2c26cd Duration: 579.40 ms Billed Duration: 580 ms Memory Size: 512 MB Max Memory Used: 75 MB Init Duration: 257.25 ms'. A note at the bottom states 'No newer events at this moment. Auto retry paused. Resume'.

Figure 4.31.: *Logs in extractFeatures lambda - notice the timestamp*

The screenshot shows the AWS S3 Buckets interface. The left sidebar includes Buckets, Storage Lens, and AWS Marketplace for S3. The main area shows the 'tags-b00936916' bucket. It contains 402 objects, all named '000ne.txt'. A table lists these objects with columns for Name, Type, Last modified, Size, and Storage class. The last modified column shows various dates from July 22, 2023, to July 24, 2023, and the size column shows sizes ranging from 1.2 KB to 544.0 B. All objects are stored in the Standard storage class.

Name	Type	Last modified	Size	Storage class
000ne.txt	txt	July 24, 2023, 11:21:07 (UTC-03:00)	54.0 B	Standard
001ne.txt	txt	July 22, 2023, 05:56:04 (UTC-03:00)	644.0 B	Standard
002ne.txt	txt	July 22, 2023, 01:14:30 (UTC-03:00)	292.0 B	Standard
003ne.txt	txt	July 22, 2023, 01:14:31 (UTC-03:00)	175.0 B	Standard
004ne.txt	txt	July 22, 2023, 01:14:32 (UTC-03:00)	334.0 B	Standard
005ne.txt	txt	July 22, 2023, 01:14:32 (UTC-03:00)	630.0 B	Standard
006ne.txt	txt	July 22, 2023, 01:14:33 (UTC-03:00)	544.0 B	Standard
007ne.txt	txt	July 22, 2023, 01:14:34 (UTC-03:00)	320.0 B	Standard
008ne.txt	txt	July 22, 2023, 01:14:36 (UTC-03:00)	402.0 B	Standard
009ne.txt	txt	July 22, 2023, 01:14:37 (UTC-03:00)	1.2 kB	Standard
010ne.txt	txt	July 22, 2023, 01:14:37 (UTC-03:00)	415.0 B	Standard
011ne.txt	txt	July 22, 2023, 01:14:39 (UTC-03:00)	537.0 B	Standard
012ne.txt	txt	July 22, 2023, 01:14:39 (UTC-03:00)	102.0 B	Standard
013ne.txt	txt	July 22, 2023, 01:14:40 (UTC-03:00)	456.0 B	Standard
014ne.txt	txt	July 22, 2023, 01:14:42 (UTC-03:00)	281.0 B	Standard

Figure 4.32.: *000ne.txt file in bucket tags-b00936916*

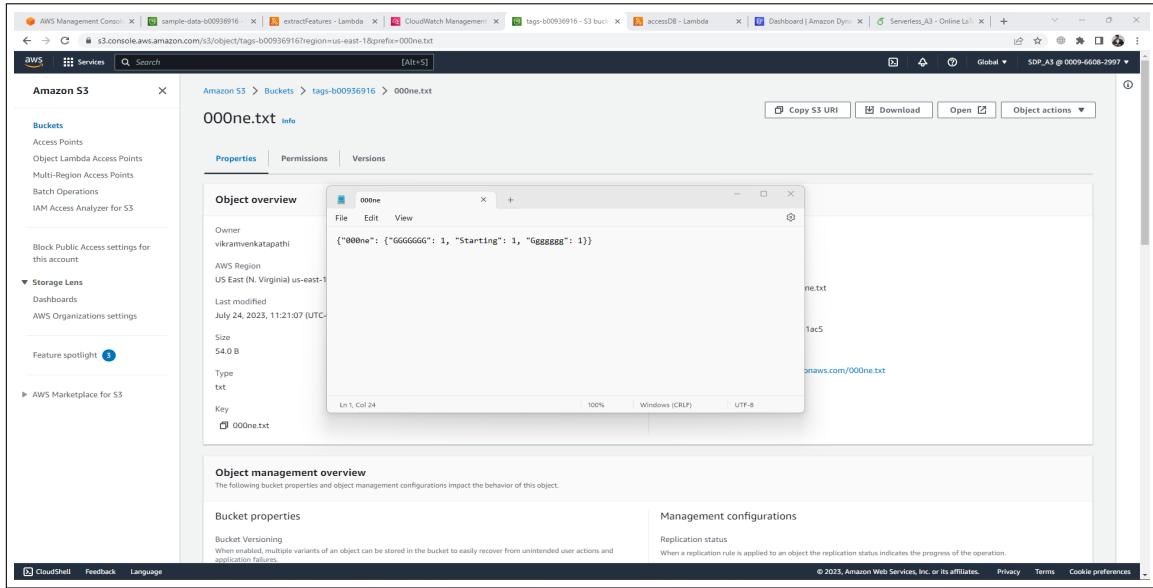


Figure 4.33.: *000ne.txt file contents - with named entities*

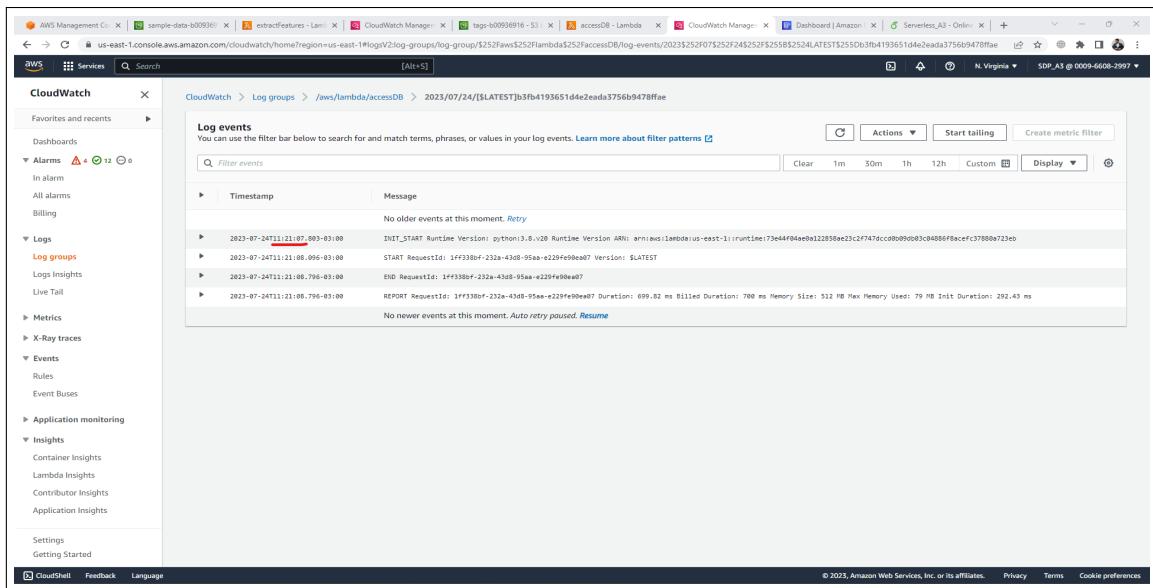


Figure 4.34.: *Logs in accessDB lambda - notice the timestamp*

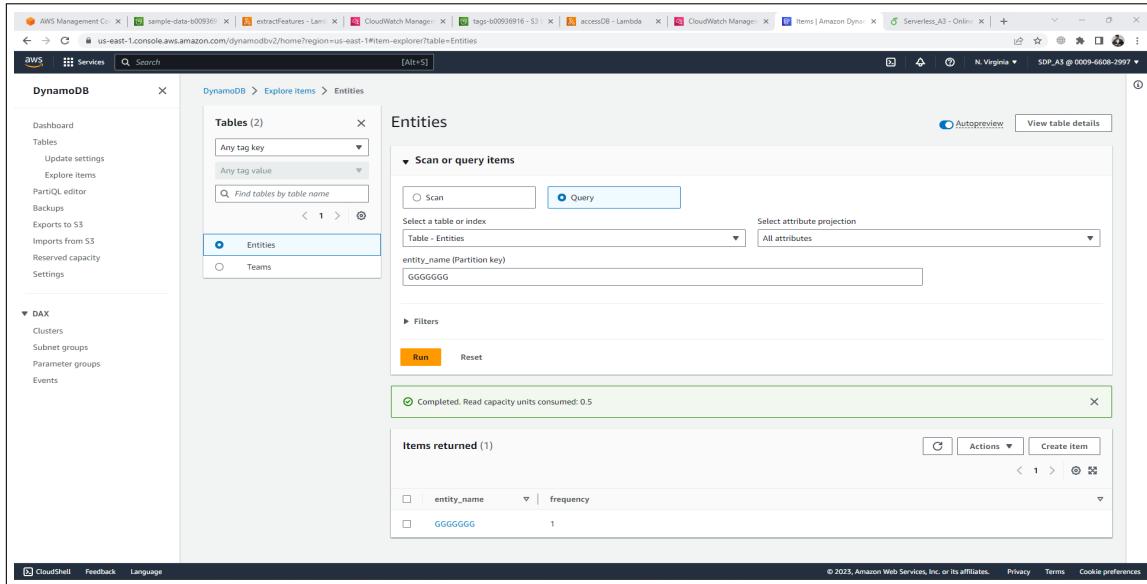


Figure 4.35.: *Query : entity_name=GGGGGGGG, frequency=1 in table*

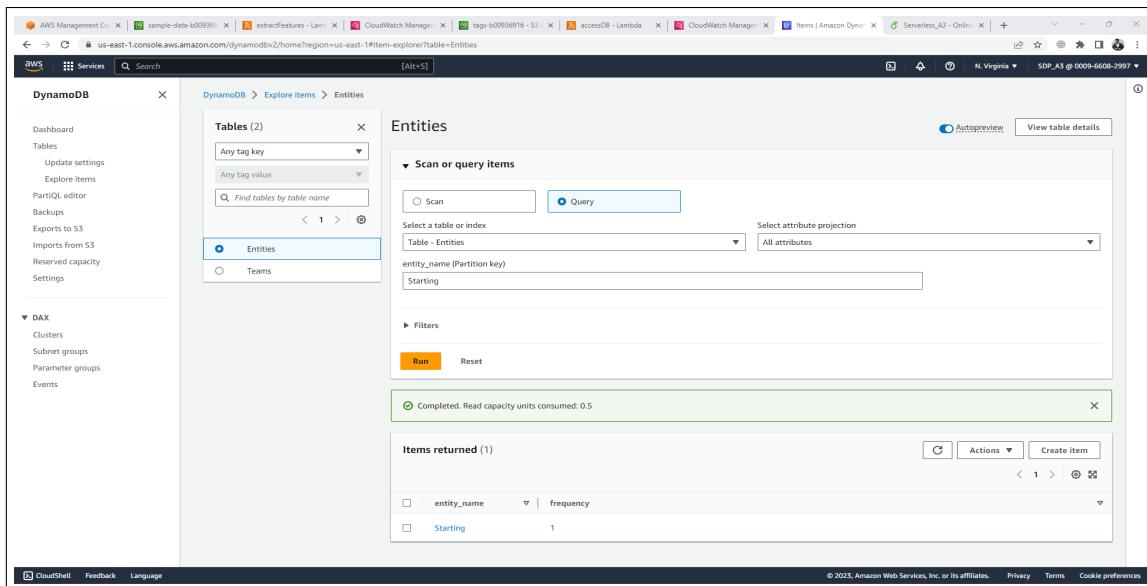


Figure 4.36.: *Query : entity_name=Starting, frequency=1 in table*

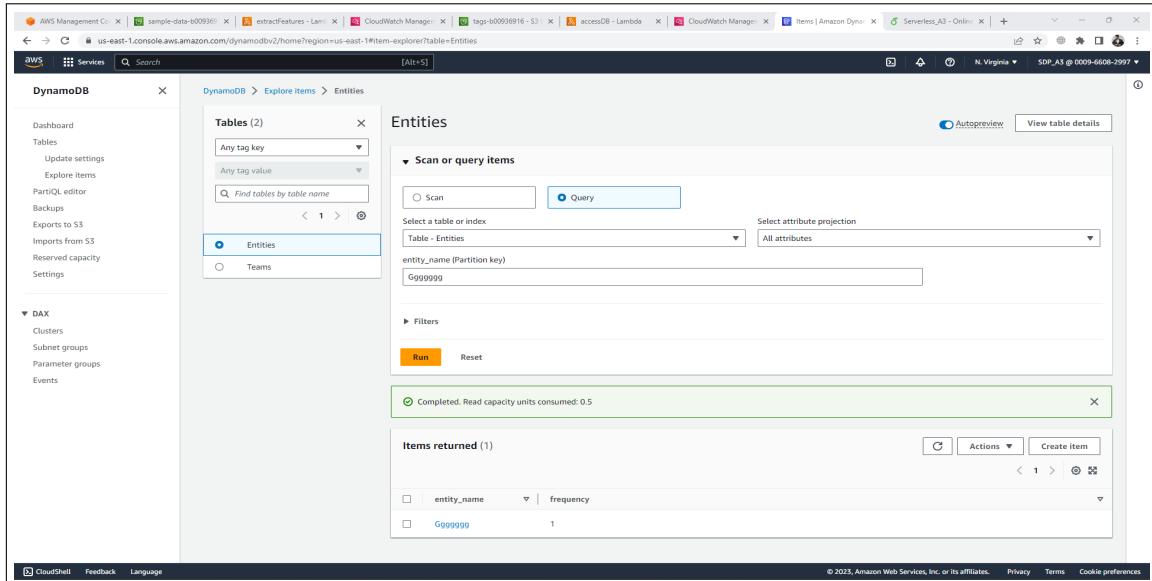


Figure 4.37.: *Query : entity_name=Ggggggg, frequency=1 in table*

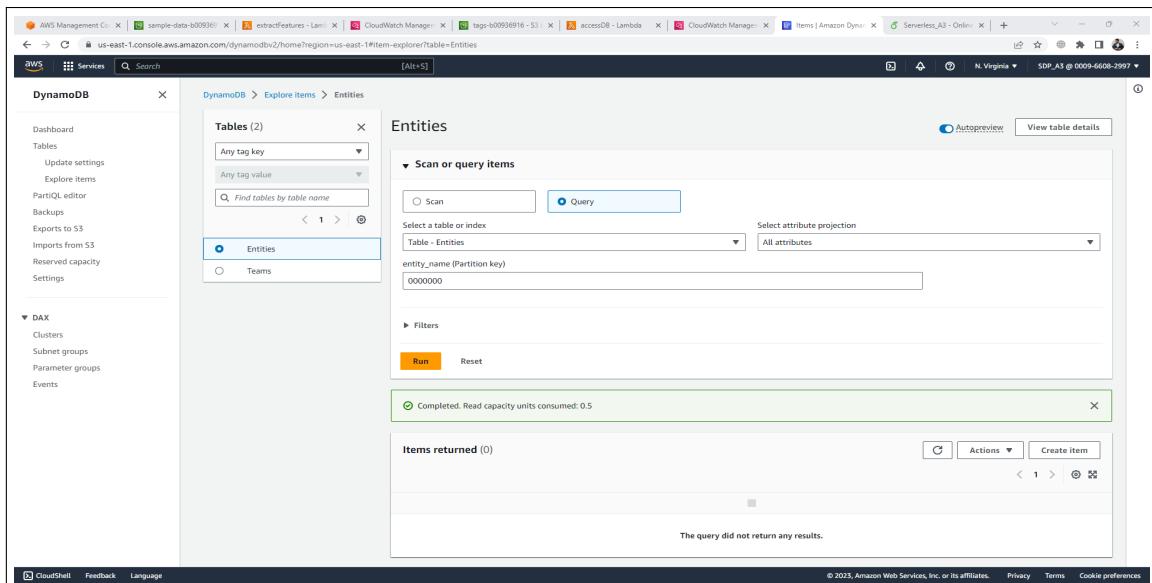


Figure 4.38.: *Query : entity_name=0000000, frequency=0 in table*

Part III.

PART C

Use AWS Lambda-SQS-SNS

5. Car Service

5.1. AWS Architecture diagram of my implementation

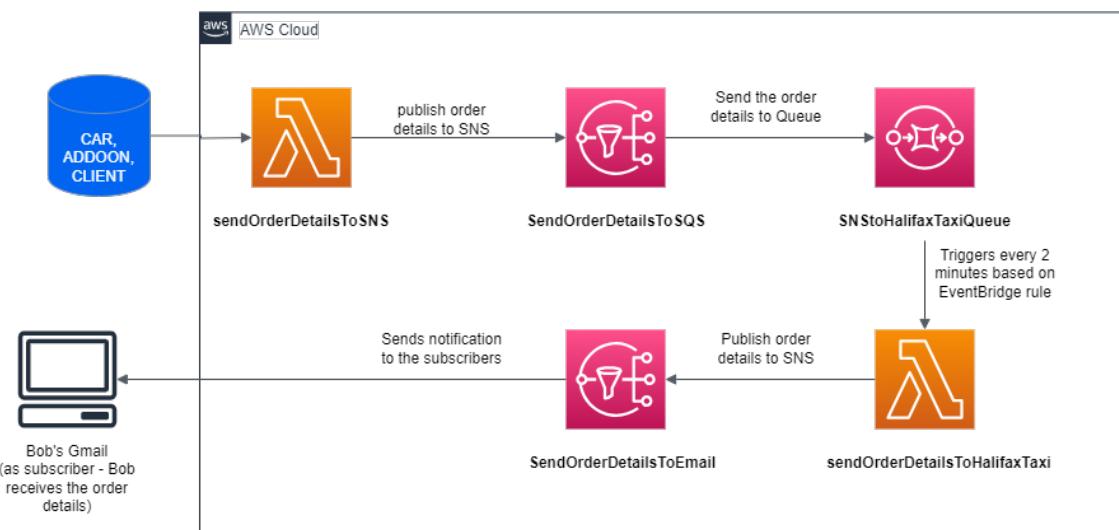


Figure 5.1.: *AWS Architecture diagram for the given work flow (Created using: draw.io [23])*

5.2. Steps

1. Create an IAM role with permissions to be : invoked, publish messages to SNS, push messages to SQS and attach to the lambda functions.
2. Create a JSON file: **AssumeRolePolicyDocument.json** with the following contents (IAM role creation) [24]

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {
```

```

        "Service": "lambda.amazonaws.com"
        ,
        "Action": "sts:AssumeRole"
    }
]
}

```

3. Create another JSON file: **invokeFunction.json** with the following contents (add invokeFunction permission)

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "*"
        }
    ]
}

```

4. Now, run the following commands to create the IAM role: **lambdaRole-Part-C** specifying the path to the above-created json file using **file://</path/to/file_name>.json** [6]

```

aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json

aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json

```

5. Run the following commands to add the permissions to the created lambda role **lambdaRole-Part-C** [25]

```

# AWSLambdaExecute role with permissions - Put, Get access to S3 and full access to CloudWatch Logs.
aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arn arn:aws:iam::aws:policy/AWSLambdaExecute

# AmazonSNSFullAccess role with permissions - Full access to SNS
aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arn arn:aws:iam::aws:policy/AmazonSNSFullAccess

# AmazonSQSFullAccess role with permissions - Full access to SQS
aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arn

```

```
arn:aws:iam::aws:policy/AmazonSQSFullAccess
```

6. Run the following command to create a serverless-framework project [3]

```
# Create a new AWS Python3 Serverless project at path 'CarService'.
serverless create --template aws-python3 --path CarService
```

7. Create the required resources using the following serverless framework script[3] (serverless.yml)[4]

```
service: CarService
provider:
  name: aws
  runtime: python3.8
  region: us-east-1
  stage: dev

resources:
  Resources:
    # 1st Lambda Function to Send order details to SNS
    SendOrderDetailsToSNS:
      Type: AWS::Lambda::Function
      Properties:
        FunctionName: sendOrderDetailsToSNS
        Code:
          ZipFile: |
            import json
            def lambda_handler(event, context):
              # TODO implement
              return {
                'statusCode': 200,
                'body': json.dumps('Hello from Lambda!')
              }
      Handler: app.lambda_handler  # Ensure that the handler function
        name matches the code
      Runtime: python3.8
      MemorySize: 512
      Timeout: 10
      Role: arn:aws:iam::000966082997:role/lambdaRole-Part-C

    # 2nd Lambda Function to Send Order Details to Halifax Taxi
    SendOrderDetailsToHalifaxTaxi:
      Type: AWS::Lambda::Function
      Properties:
        FunctionName: sendOrderDetailsToHalifaxTaxi
        Code:
          ZipFile: |
```

```

import json
def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
Handler: app.handler # Ensure that the handler function name
# matches the code
Runtime: python3.8
MemorySize: 512
Timeout: 10
Role: arn:aws:iam::000966082997:role/lambdaRole-Part-C

# 1st SNS Topic - To take order details from Lambda and send to SQS
SnsTopicToSqs:
Type: AWS::SNS::Topic
Properties:
    TopicName: SendOrderDetailsToSQS

# 1st SNS Topic Subscription - Lambda subscribing to the SNS topic
SnsSubscriptionToSqs:
Type: AWS::SNS::Subscription
Properties:
    TopicArn: !Ref SnsTopicToSqs
    Protocol: lambda
    Endpoint:
        Fn::GetAtt:
            - SendOrderDetailsToSNS
            - Arn

# 2nd SNS Topic - To poll order details from SQS and publish to
# email
SnsTopicToEmail:
Type: AWS::SNS::Topic
Properties:
    TopicName: SendOrderDetailsToEmail
    Subscription:
        - Protocol: email
        Endpoint: vikramvenkatapathi@gmail.com

# 2nd SNS Topic Subscription - Lambda subscribing to the SNS topic
SnsSubscriptionToEmail:
Type: AWS::SNS::Subscription
Properties:
    TopicArn: !Ref SnsTopicToEmail
    Protocol: lambda
    Endpoint:

```

```

Fn::GetAtt:
  - SendOrderDetailsToHalifaxTaxi
  - Arn
SNSToHalifaxTaxiQueue:
  Type: AWS::SQS::Queue
  Properties:
    QueueName: SNSToHalifaxTaxiQueue

  # SQS Queue Subscription - Subscribe SQS to SNS topic
  SqssSubscriptionToSns:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref SnsTopicToSqs
      Protocol: sqs
      Endpoint:
        Fn::GetAtt:
          - SNSToHalifaxTaxiQueue
          - Arn

  # CloudWatch Events Rule to trigger Lambda function periodically
  # every 2 minutes
SendOrderDetailsToHalifaxTaxiSchedule:
  Type: AWS::Events::Rule
  Properties:
    ScheduleExpression: rate(2 minutes)
    Targets:
      - Arn: arn:aws:lambda:us-east-1:000966082997:function:
          sendOrderDetailsToHalifaxTaxi
      Id: SendOrderDetailsToHalifaxTaxiScheduleTarget

```

8. The above script does the following
 - a) Create 2 - lambdas: SendOrderDetailsToSNS, SendOrderDetailsToHalifaxTaxi
 - b) Create 2 - SNS topics: SendOrderDetailsToSQS, SendOrderDetailsToEmail
 - c) Create 1 - SQS Queue: SNSToHalifaxTaxiQueue
 - d) Create 2 subscriptions for SNS topic SendOrderDetailsToSQS
 - i. Lambda - SendOrderDetailsToSNS
 - ii. SQS - SNSToHalifaxTaxiQueue
 - e) Create 2 subscriptions for SNS topic SendOrderDetailsToEmail
 - i. Lambda - sendOrderDetailsToHalifaxTaxi
 - ii. Bob's Email - vikramvenkatapathi@gmail.com (my personal email)
 - f) Create a CloudWatch Events Rule - SendOrderDetailsToHalifaxTaxiSchedule, to trigger Lambda function: sendOrderDetailsToHalifaxTaxi, periodically every 2 minutes.

- Run the command `serverless deploy` to deploy the .yml file to create the resources in CloudFormation.

5.3. Lambda Code

5.3.1. sendOrderDetailsToSNS

```

import json
import boto3
import random

# Sample data lists
car_types = ["Compact", "Mid-size Sedan", "SUV", "Luxury"]
car_accessories = ["GPS", "Camera", "Sun shades", "Vehicle Seat Cover"]
client_addresses = ["123 Main St", "456 Elm St", "789 Oak St", "1333
    South park st"]

def lambda_handler(event, context):

    """
        W3schools - Python Random Module Documentation
        Reference: `W3schools - Python Random Module <https://www.
            w3schools.com/python/module_random.asp>`_
    """

    # Generate a single message by combining data from all three lists
    order_details = {
        "type": "ORDER",
        "data": {
            "CAR_TYPE": random.choice(car_types),
            "CAR_ACCESSORY": random.choice(car_accessories),
            "CLIENT_ADDRESS": random.choice(client_addresses)
        }
    }

    # Publish the combined message to the SNS topic
    sns_topic_arn = "arn:aws:sns:us-east-1:000966082997:
        SendOrderDetailsToSQS"
    publish_to_sns(sns_topic_arn, order_details)

    print(order_details)
    return {
        'statusCode': 200,
        'body': json.dumps('Message sent to SNS topic=
            SendOrderDetailsToSQS!')
    }

```

```

def publish_to_sns(topic_arn, message):

    """
        Boto3 - Amazon Amazon Simple Notification Service (SNS)
        Documentation
        Reference: `Boto3 - Amazon SNS API <https://boto3.amazonaws.com/
                    v1/documentation/api/latest/reference/services/sns.html>`_
    """

    sns_client = boto3.client('sns')

    sns_client.publish(
        TopicArn=topic_arn,
        Message=json.dumps(message)
    )

    """
        All references:
        Boto3 - Amazon Amazon Simple Notification Service (SNS)
        Documentation
        Reference: `Boto3 - Amazon SNS API <https://boto3.amazonaws.com/v1/
                    documentation/api/latest/reference/services/sns.html>`_

        W3schools - Python Random Module Documentation
        Reference: `W3schools - Python Random Module <https://www.w3schools.
                    com/python/module_random.asp>`_
    """

"""

```

5.3.2. sendOrderDetailsToHalifaxTaxi

```

import json
import boto3

def handler(event, context):
    # Get the SQS queue URL
    queue_url = "https://sqs.us-east-1.amazonaws.com/000966082997/
                 SNSToHalifaxTaxiQueue"

    """
        Boto3 - Amazon Amazon Simple Queue Service (SQS) Documentation
        Reference: `Boto3 - Amazon SQS API <https://boto3.amazonaws.com/
                    v1/documentation/api/latest/reference/services/sqs.html>`_
    """

    # Create an SQS client
    sqs = boto3.client('sqs')

```

```

# Receive messages from the SQS queue (max 1 message)
response = sqs.receive_message(
    QueueUrl=queue_url,
    MaxNumberOfMessages=1,
    VisibilityTimeout=30,
    WaitTimeSeconds=0 # Set to 0 for non-blocking receive
)

# Check if any messages were received
if 'Messages' in response:
    # Extract the message
    message = response['Messages'][0]
    body = json.loads(message['Body'])
    order_details = body['Message']
    # print(order_details)

    order_details_json = json.loads(order_details)

    # Template for the email body
    email_body_template = """Car Type : {car_type}, Car Accessory :
        {car_accessory}, Client Address : {client_address}"""

    """
    W3schools - Python String format() Method Documentation
    Reference: `W3schools - Python String format() Method <https://www.w3schools.com/python/ref_string_format.asp>`_
    """

    # Replace the placeholders in the email_body template
    email_body = email_body_template.format(
        car_type=order_details_json['data']['CAR_TYPE'],
        car_accessory=order_details_json['data']['CAR_ACCESSORY'],
        client_address=order_details_json['data']['CLIENT_ADDRESS']
    )

    # Publish the message to the SNS topic
    sns_topic_arn = "arn:aws:sns:us-east-1:000966082997:
        SendOrderDetailsToEmail"

    publish_to_sns(sns_topic_arn, email_body)

    # Delete the message from the SQS queue
    sqs.delete_message(
        QueueUrl=queue_url,
        ReceiptHandle=message['ReceiptHandle']
    )

return {

```

```

        'statusCode': 200,
        'body': json.dumps('Message sent to subscriber')
    }

def publish_to_sns(topic_arn, message):
    # Create an SNS client
    sns = boto3.client('sns')

    """
    Boto3 - Amazon Amazon Simple Notification Service (SNS)
    Documentation
    Reference: `Boto3 - Amazon SNS API <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sns.html>`_
    """
    # Publish the message to the SNS topic
    sns.publish(
        TopicArn=topic_arn,
        Subject="Your order details",
        Message=json.dumps(message)
    )

    """
    All references:
    Boto3 - Amazon Amazon Simple Notification Service (SNS)
    Documentation
    Reference: `Boto3 - Amazon SNS API <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sns.html>`_
    Boto3 - Amazon Amazon Simple Queue Service (SQS) Documentation
    Reference: `Boto3 - Amazon SQS API <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sqs.html>`_
    W3schools - Python Random Module Documentation
    Reference: `W3schools - Python Random Module <https://www.w3schools.com/python/module\_random.asp>`_
    """

```

5.4. Testing

I followed the below testing procedure

- Triggered the 1st lambda **SendOrderDetailsToSNS** 2 times.
- Checked it logs - 2 messageBody's were present.
- Check if it published the messages to the SNS topic **SnsTopicToSqs**.

- d) Polled the messages in SQS **SNSToHalifaxTaxiQueue**, and verified that I received the 2 messages.
- e) Checked the logs of 2nd lambda **SendOrderDetailsToHalifaxTaxi**, and verified that messages are received periodically every 2 minutes based on the EventBridge rule.
- f) Verified the receipt of mail in Gmail.

5.5. Screenshots

Role name	Trusted entities	Last activity
ALLReadOnly	AWS Service: ec2	38 days ago
AWSServiceRoleForAPIGateway	AWS Service: cps.apigateway (Service-Linked Role)	-
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	45 minutes ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	33 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	33 days ago
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	-
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	58 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
ConfirmSubscriptionTrigger-role-pzwm21p9	AWS Service: lambda	40 minutes ago
flask-app-v2-dev-EnterpriseLogAccessIamRole-1QH7DT97HDXEX	Account: 802587217904	-
flask-app-v2-dev-us-east-1-lambdaRole	AWS Service: lambda	2 days ago
lambdaRole	AWS Service: lambda	17 hours ago
lambdaRole_accessDB	AWS Service: lambda	-
snsDeliveryStatus	AWS Service: sns	-
snsDeliveryStatusFailure	AWS Service: sns	-
SNSSendMsgToSQS-role-68u17om1	AWS Service: lambda	3 days ago

Figure 5.2.: *Before IAM Role creation*

```

File Edit Selection View Go Run Terminal Help
commands - csci5410-summer-23-b00936916 - Visual Studio Code
EXPLORER commands invokeLambda.json commands U
Part C > commands
1 aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json
2
3 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AWSLambdaExecute
4
5 aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
powershell - Part C + v 🌐 ⚙️ ... ×
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C> aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json
>>>
{
  "Role": {
    "Path": "/",
    "RoleName": "lambdaRole-Part-C",
    "RoleId": "AROAQOWV1GJUH443642C",
    "Arn": "arn:aws:iam::000966082997:role/lambdaRole-Part-C",
    "CreateDate": "2023-07-23T02:21:19+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "lambda.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}

Ln 2, Col 1 (118 selected) Spaces: 2 UTF-8 CRLF YAML ⚙️

```

Figure 5.3.: *Created IAM role lambdaRole-Part-C in CLI*

```

File Edit Selection View Go Run Terminal Help
commands - csci5410-summer-23-b00936916 - Visual Studio Code
EXPLORER commands invokeLambda.json commands U
Part C > commands
1 aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json
2
3 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AWSLambdaExecute
4
5 aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
powershell - Part C + v 🌐 ⚙️ ... ×
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C> aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AWSLambdaExecute
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C> []

Ln 3, Col 1 (110 selected) Spaces: 2 UTF-8 CRLF YAML ⚙️

```

Figure 5.4.: *Attach AWSLambdaExecute permission to role: lambdaRole-Part-C*

```

File Edit Selection View Go Run Terminal Help
commands - csci5410-summer-23-b00936916 - Visual Studio Code
EXPLORER File serverless.yml commands Part C M commands Part B
Part C > commands
  You, 1 second ago | 1 author (You)
  1 aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json
  2
  3 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AWSLambdaExecute
  4
  5 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AmazonSNSFullAccess
  6
  7 aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json
  8
  9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C\carservice> aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AmazonSNSFullAccess
>>
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C\carservice> []

```

Figure 5.5.: *Attach AmazonSNSFullAccess permission to role: lambdaRole-Part-C*

```

File Edit Selection View Go Run Terminal Help
commands - csci5410-summer-23-b00936916 - Visual Studio Code
EXPLORER File serverless.yml commands Part C M commands Part B
Part C > commands
  You, 1 second ago | 1 author (You)
  1 aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json
  2
  3 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AWSLambdaExecute
  4
  5 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AmazonSNSFullAccess
  6
  7 aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AmazonSQSFullAccess
  8
  9 aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json
  10
  11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C\carservice> aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AmazonSQSFullAccess
>>
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C\carservice> []

```

Figure 5.6.: *Attach AmazonSQSFullAccess permission to role: lambdaRole-Part-C*

The screenshot shows a Visual Studio Code terminal window with the following AWS CLI commands:

```

aws iam create-role --role-name lambdaRole-Part-C --assume-role-policy-document file://AssumeRolePolicyDocument.json
aws iam attach-role-policy --role-name lambdaRole-Part-C --policy-arm arn:aws:iam::aws:policy/AWSLambdaExecute
aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json

```

The terminal also displays PowerShell output:

```

PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C> aws iam put-role-policy --role-name lambdaRole-Part-C --policy-name LambdaExecutionPolicy --policy-document file://invokeFunction.json
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C>

```

Figure 5.7.: *Added invokeFunction permission to role: lambdaRole-Part-C*

The screenshot shows the AWS IAM Roles page for the role **lambdaRole-Part-C**. The **Permissions** tab is selected, displaying the following policy details:

Permissions policies (4) Info
You can attach up to 10 managed policies.

Policy name	Type	Description
AmazonSNSFullAccess	AWS managed	Provides full access to Amazon SNS via the AWS Management Console.
AmazonSQSFullAccess	AWS managed	Provides full access to Amazon SQS via the AWS Management Console.
AWSLambdaExecution	AWS managed	Provides Put, Get access to S3 and full access to CloudWatch Logs.
LambdaExecutionPolicy	Customer inline	Customer inline policy content:

```

1 < [ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "Lambda:InvokeFunction", "Resource": "*" } ] ]

```

Figure 5.8.: *Created IAM role lambdaRole-Part-C's permissions shown in console*

```

File Edit Selection View Go Run Terminal Help
serverless.yml - csci5410-summer-23-b00936916 - Visual Studio Code
EXPLORER serverless.yml
Part C > CarService > serverless.yml
  10)      return {
  11)        'statusCode': 200,
  12)        'body': json.dumps('Hello from Lambda!')
  13)    }
  14) Handler: app.handler # Ensure that the handler function name matches your code
  15) Runtime: python3.8
  16) MemorySize: 512
  17) Timeout: 10
  18) Role: arn:aws:iam::000966082997:role/lambdarole-Part-C
  19)
  20) # 1st SNS Topic - To take order details from Lambda and send to SQS
  21) SnsTopicToSqs:
  22)   Type: AWS::SNS::Topic
  23)   Properties:
  24)     TopicName: SenderOrderDetailsToSQS
  25)
  26) # 1st SNS Topic Subscription - Lambda subscribing to the SNS topic
  27)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C\carservice> serverless deploy
Deploying CarService to stage dev (us-east-1)
✓ Service deployed to stack CarService-dev (27s) ←

```

Need a faster logging experience than CloudWatch? Try our Dev Mode in console: run "serverless dev"

PS C:\Users\vikra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\csci5410-summer-23-b00936916\Part C\carservice>

You, 1 second ago Ln 74, Col 51 Spaces: 2 UTF-8 LF YAML ↻

Figure 5.9.: *serverless.yml* file executed in CLI

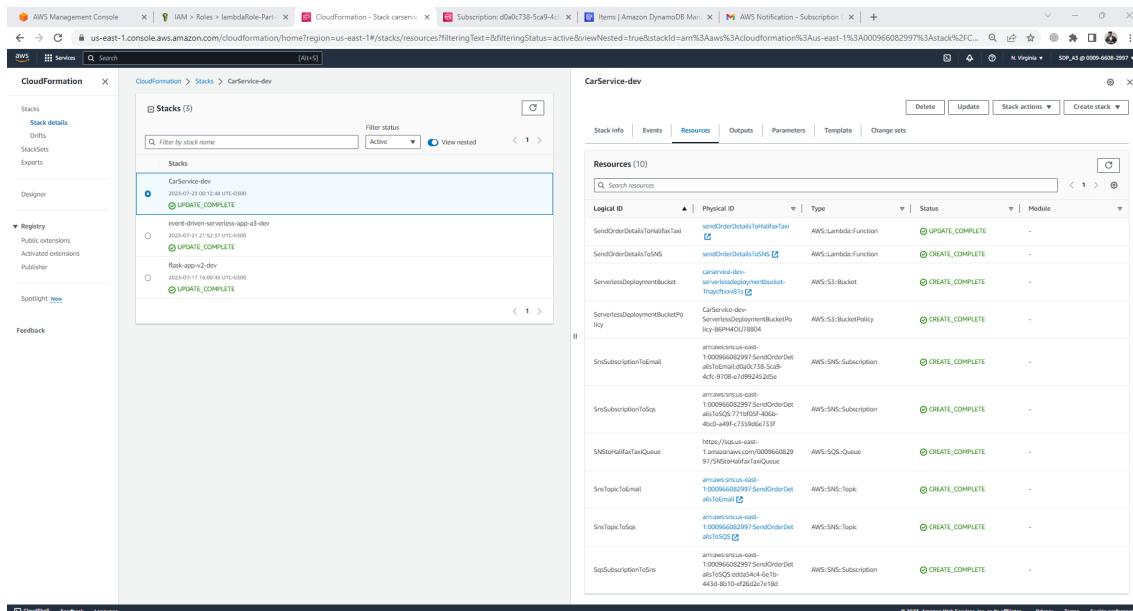


Figure 5.10.: Cloud formation stack created - shown in console

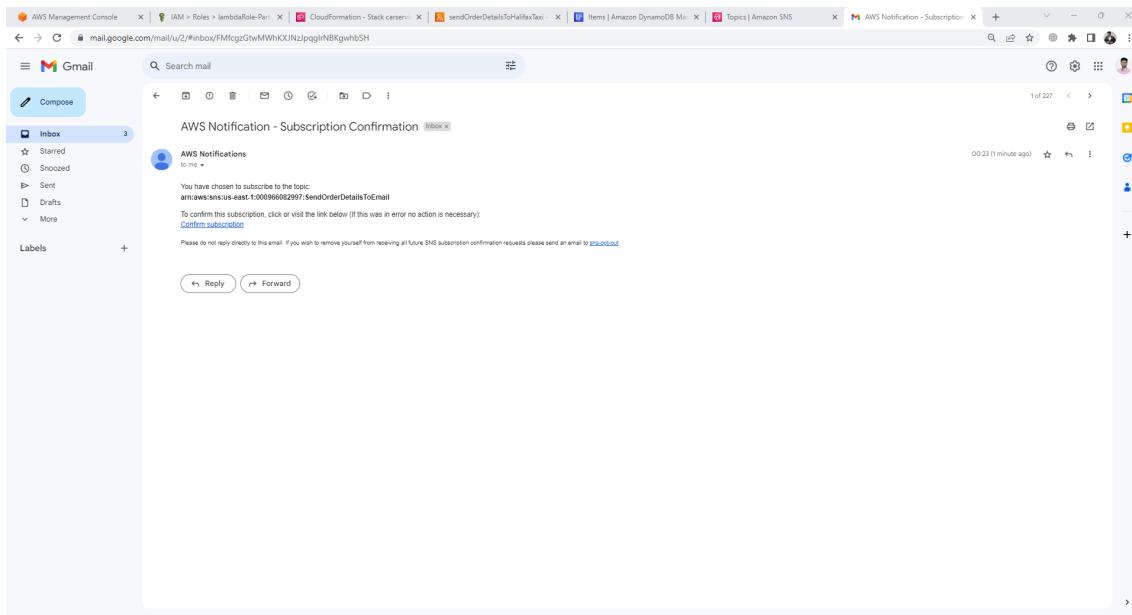


Figure 5.11.: *Subscription mail received*

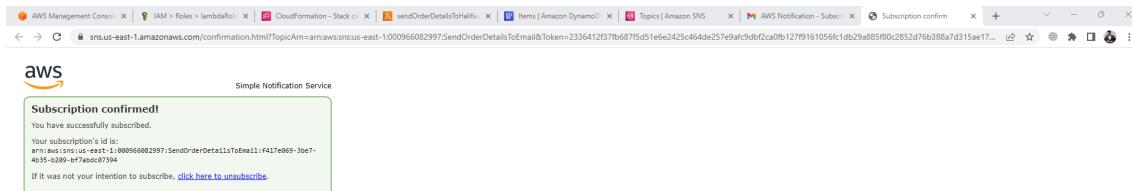


Figure 5.12.: *Confirmed subscription*

The screenshot shows the AWS Management Console interface for the Amazon SNS service. The top navigation bar includes links for AWS Management Console, IAM, CloudFormation, Subscription, Items, Amazon DynamoDB, and AWS Notification. The main title is "Amazon SNS > Topics > SendOrderDetailsToSQS".

Details:

- Name: SendOrderDetailsToSQS
- ARN: arn:aws:sns:us-east-1:000966082997:SendOrderDetailsToSQS
- Type: Standard
- Display name: -
- Topic owner: 000966082997

Subscriptions:

ID	Endpoint	Status	Protocol
771bf05f-406b-4bc0-a49f-c7359d6e733f	arn:aws:lambda:us-east-1:000966082997:function:sendOrderDetailsToSQS	Confirmed	LAMBDA
edda54c4-6e1b-443d-8b10-e1f26d2e7e18d	arn:aws:sqs:us-east-1:000966082997:5NStoHalifaxTaxiQueue	Confirmed	SQS

Figure 5.13.: *Subscriptions of topic=SendOrderDetailsToSQS*

The screenshot shows the AWS Management Console interface for the Amazon SNS service. The top navigation bar includes links for AWS Management Console, IAM, CloudFormation, Subscription, Items, Amazon DynamoDB, and AWS Notification. The main title is "Amazon SNS > Topics > SendOrderDetailsToEmail".

Details:

- Name: SendOrderDetailsToEmail
- ARN: arn:aws:sns:us-east-1:000966082997:SendOrderDetailsToEmail
- Type: Standard
- Display name: -
- Topic owner: 000966082997

Subscriptions:

ID	Endpoint	Status	Protocol
d0a0c738-5ca9-4cf8-9708-e7d992452d5e	arn:aws:lambda:us-east-1:000966082997:function:sendOrderDetailsToHalifaxTaxi	Confirmed	LAMBDA
f417e069-3be7-4b35-b209-bf7abdc07394	vikramvenkatapathi@gmail.com	Confirmed	EMAIL

Figure 5.14.: *Subscriptions of topic=SendOrderDetailsToEmail*

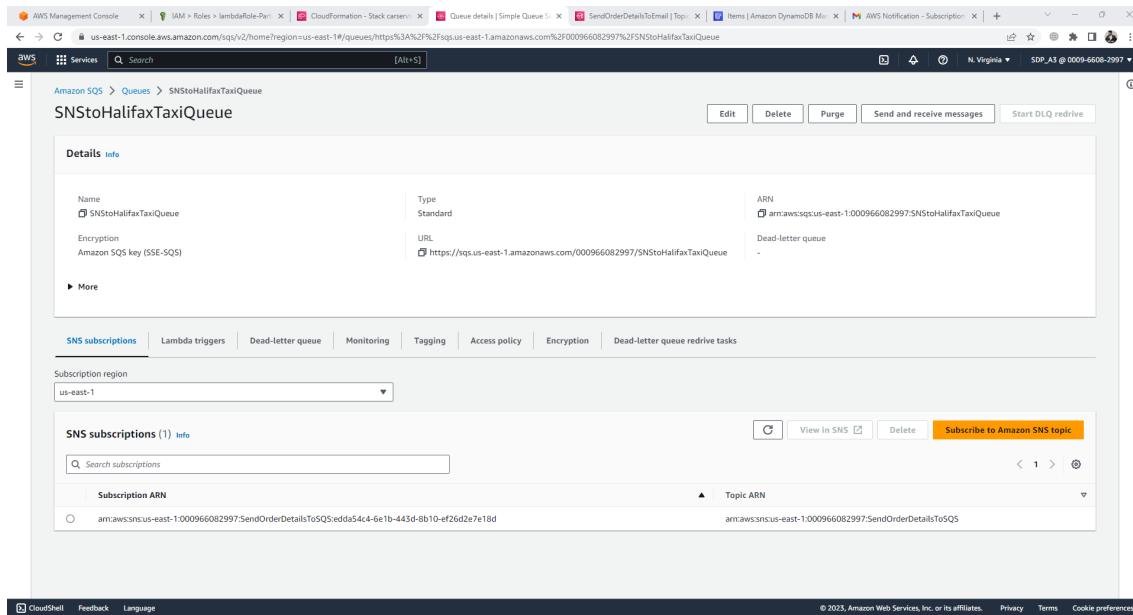


Figure 5.15.: *SNSToHalifaxTaxiQueue - SNS subscription*

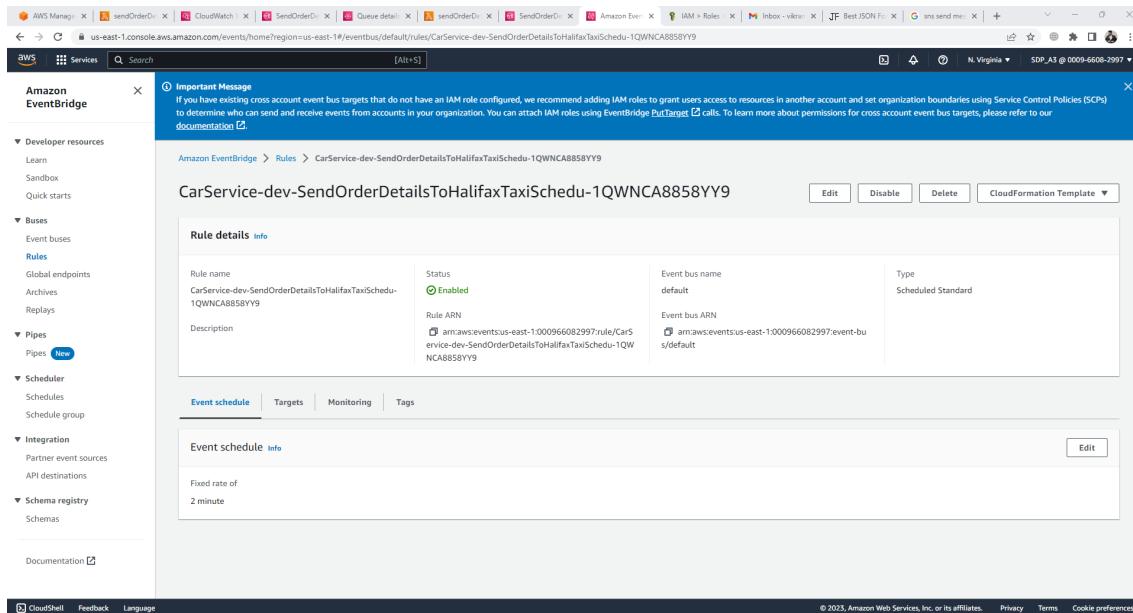


Figure 5.16.: *Event rule "CarService-dev-SendOrderDetailsToHalifaxTaxiSchedu-1QWNCA8858YY9", created for Lambda: SendOrderDetailsToHalifaxTaxi*

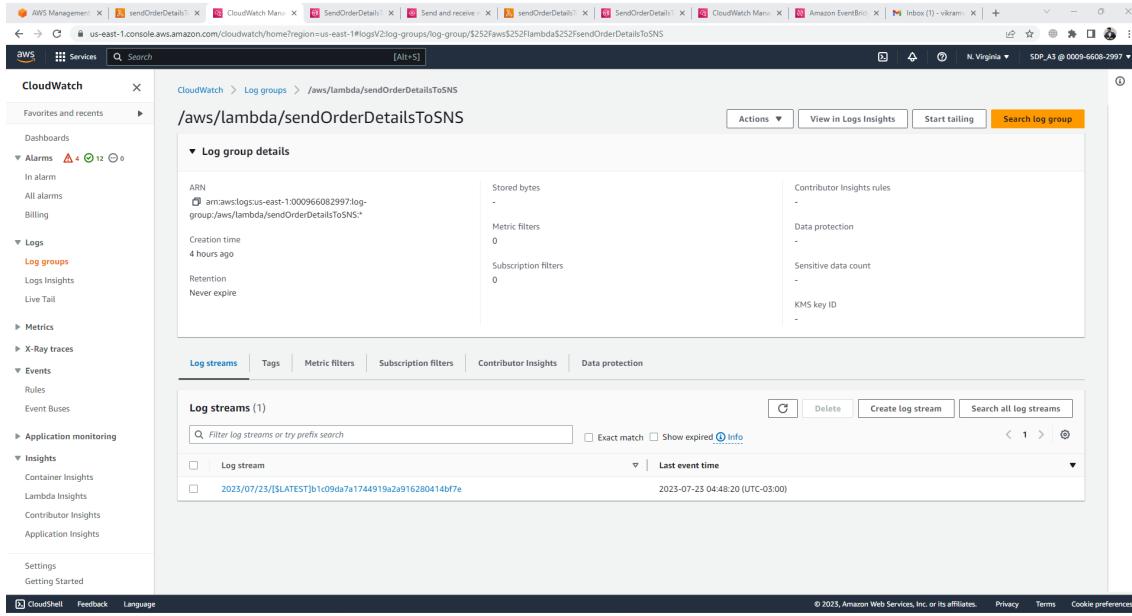


Figure 5.17.: *sendOrderDetailsToSNS lambda - send message logs*

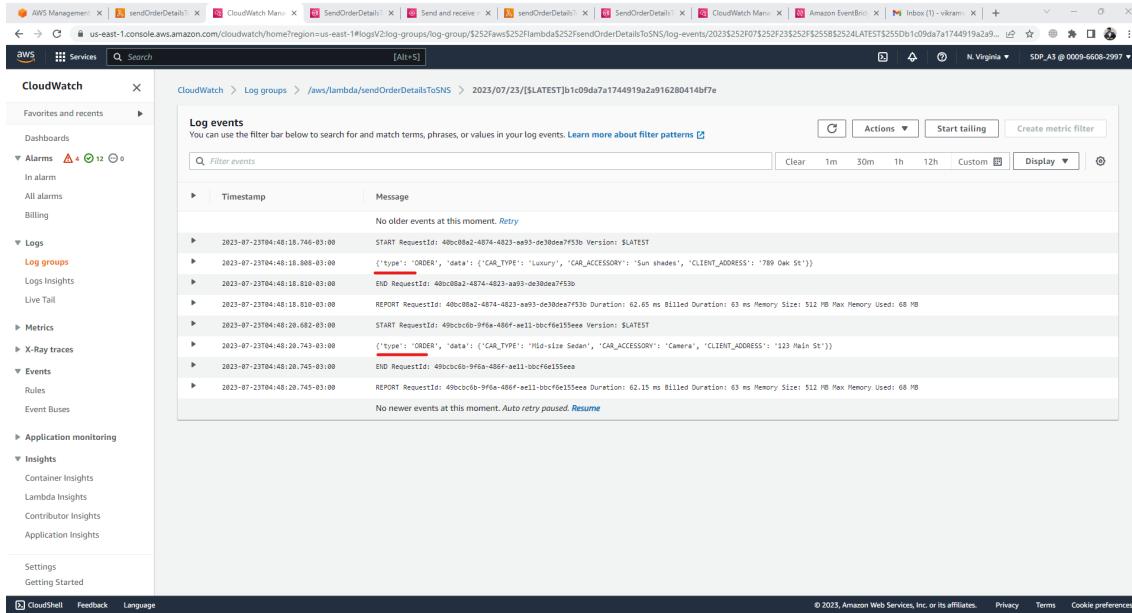


Figure 5.18.: *sendOrderDetailsToSNS lambda - send message logs- 2 messages sent*

Figure 5.19.: *sendOrderDetailsToHalifaxTaxi lambda - receive msgs logs*

Timestamp	Message
2023-07-23T04:40:24.159+01:00	START RequestId: b3236a27-66e7-488d-826e-049a5f5ed3f5 Version: \$LATEST
2023-07-23T04:40:24.236+01:00	{"type": "ORDER", "data": {"CAR_TYPE": "Mid-size Sedan", "CAR_ACCESSORY": "Camera", "CLIENT_ADDRESS": "123 Main St"}}
2023-07-23T04:40:24.314+01:00	END RequestId: b3236a27-66e7-488d-826e-049a5f5ed3f5
2023-07-23T04:40:24.314+01:00	REPORT RequestId: b3236a27-66e7-488d-826e-049a5f5ed3f5 Duration: 115.75 ms Billed Duration: 116 ms Memory Size: 512 MB Max Memory Used: 69 MB
2023-07-23T04:50:23.654+01:00	START RequestId: 9936a58e-669f-42cf-879d-7d0b0553a838 Version: \$LATEST
2023-07-23T04:50:23.654+01:00	{"type": "ORDER", "data": {"CAR_TYPE": "Luxury", "CAR_ACCESSORY": "Sun shade", "CLIENT_ADDRESS": "789 Oak St"}}
2023-07-23T04:50:23.732+01:00	END RequestId: 9936a58e-669f-42cf-879d-7d0b0553a838
2023-07-23T04:50:23.732+01:00	REPORT RequestId: 9936a58e-669f-42cf-879d-7d0b0553a838 Duration: 118.47 ms Billed Duration: 119 ms Memory Size: 512 MB Max Memory Used: 78 MB

Figure 5.20.: *sendOrderDetailsToHalifaxTaxi lambda - receive msgs logs - 2 msgs received*

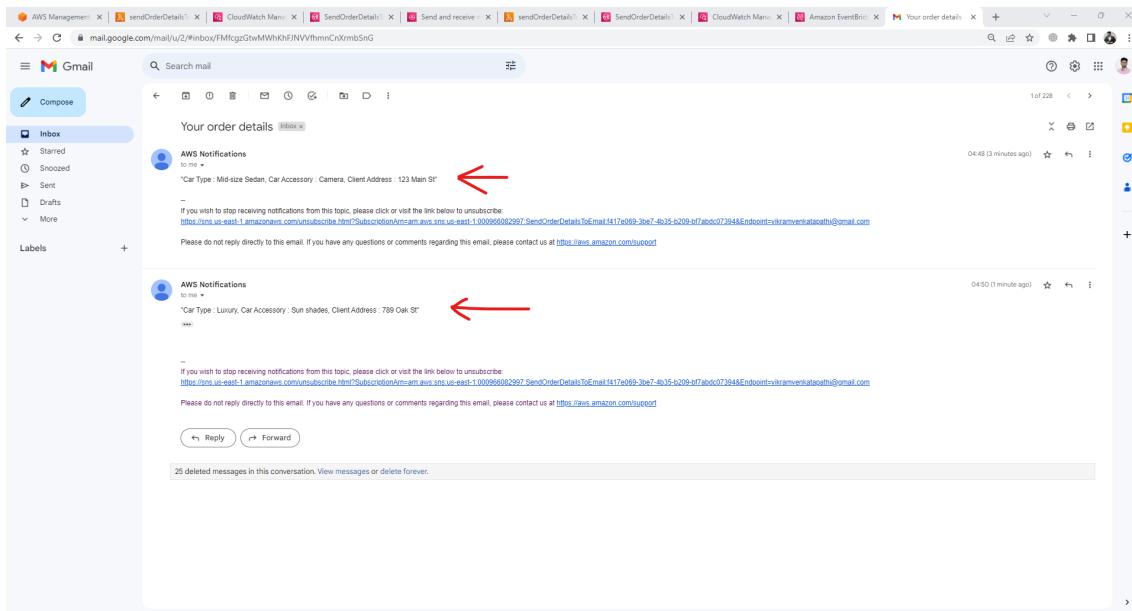


Figure 5.21.: 2 Emails with order details - received

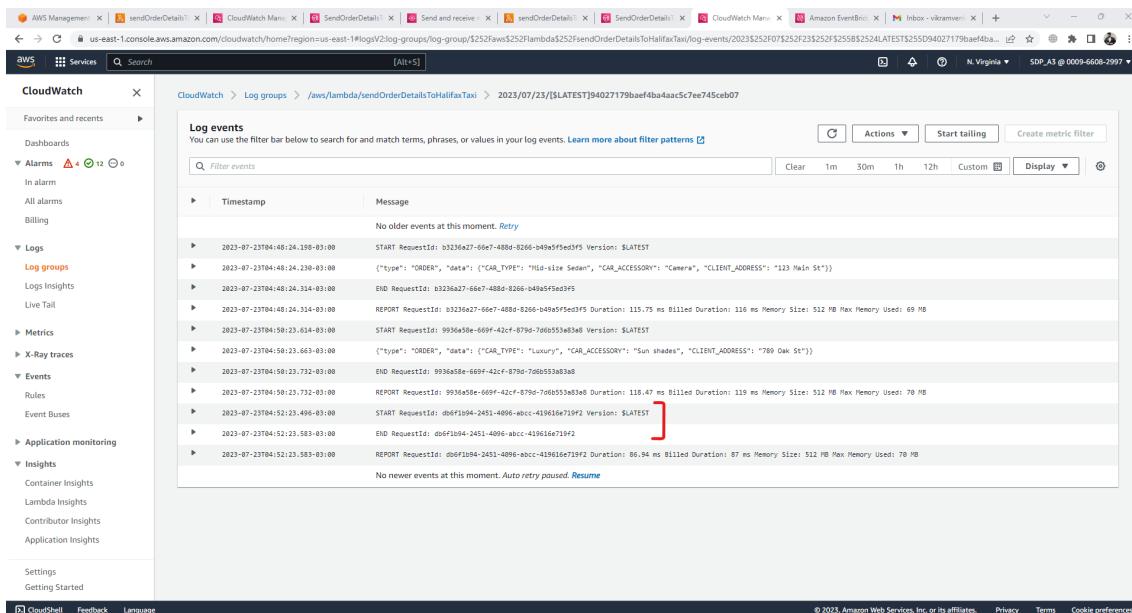


Figure 5.22.: No more msgs in the queue - so empty lambda invoke- didn't send any mail

Part IV.

REFERENCES

References

- [1] "Amazon Kinesis," *Amazon Web Services*, Inc., 2019. Available: <https://aws.amazon.com/kinesis>. [Accessed: 20 July 2023].
- [2] "Atlantic Superstore supermarket: Grocery shop online or instore," *Atlantic Superstore Supermarket — Grocery shop online or instore*, [Online]. Available: <https://www.atlanticsuperstore.ca/> [Accessed: July 22, 2023].
- [3] "Zero-Friction Serverless Apps on AWS," *Serverless*, [Online]. Available: <https://www.serverless.com/> [Accessed: July 22, 2023].
- [4] "Develop & Monitor apps on AWS Lambda," *Serverless*, [Online]. Available: <https://www.serverless.com/> [Accessed: July 22, 2023].
- [5] Abhay Singh, "AWS Lambda S3 trigger using AWS CLI: A step-by-step guide," *Abhay Singh*, [Online]. Available: <https://abhayksingh.com/aws-lambda-s3-trigger-configuration-using-aws-cli-a-step-by-step-guide/> [Accessed: July 22, 2023].
- [6] "AWS CLI Command Reference - Load AWS CLI parameters from a file," *Amazon*, [Online]. Available: <https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html> [Accessed: July 22, 2023].
- [7] D. Musgrave, "Lambda," *AWS*, [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html> [Accessed: July 22, 2023].
- [8] "DynamoDB#," *DynamoDB - Boto3 1.28.9 documentation*, [Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html> [Accessed: July 22, 2023].
- [9] "RE - regular expression operations," *Python documentation*, [Online]. Available: <https://docs.python.org/3/library/re.html> [Accessed: July 22, 2023].
- [10] "Amazon S3 REST API introduction - LambdaFunctionConfiguration," *AWS*, [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html> [Accessed: July 22, 2023].
- [11] "AWS CLI Command Reference - put-bucket-notification-configuration," *AWS*, [Online]. Available: <https://docs.aws.amazon.com/cli/latest/reference/s3api/put-bucket-notification-configuration.html> [Accessed: July 22, 2023].
- [12] "How to use bash for loop in one line - nixcraft," *Nixcraft*, [Online]. Available: <https://www.cyberciti.biz/faq/linux-unix-bash-for-loop-one-line-command/> [Accessed: July 22, 2023].
- [13] "S3#," *S3 - Boto3 1.28.9 documentation*, [Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html> [Accessed: July 22, 2023].

- [14] "AWS IoT Core," *AWS*, [Online]. Available: <https://aws.amazon.com/iot-core/> [Accessed: July 22, 2023].
- [15] "Amazon Kinesis Data Streams", *AWS*, [Online]. Available: <https://aws.amazon.com/kinesis/data-streams/> [Accessed: July 22, 2023].
- [16] "Amazon Kinesis Data Analytics", *AWS*, [Online]. Available: <https://aws.amazon.com/kinesis/data-analytics/> [Accessed: July 22, 2023].
- [17] "AWS Lambda" *AWS*, [Online]. Available: <https://aws.amazon.com/lambda/> [Accessed: July 22, 2023].
- [18] "Amazon Comprehend", *AWS*, [Online]. Available: <https://aws.amazon.com/comprehend/> [Accessed: July 22, 2023].
- [19] "Amazon DynamoDB", *AWS*, [Online]. Available: <https://aws.amazon.com/dynamodb/> [Accessed: July 22, 2023].
- [20] "Amazon QuickSight", *AWS*, [Online]. Available: <https://aws.amazon.com/quicksight/> [Accessed: July 22, 2023].
- [21] "AWS creating IAM roles - javatpoint," *javatpoint*, [Online]. Available: <https://www.javatpoint.com/aws-creating-iam-roles> [Accessed: July 22, 2023].
- [22] D. Greene and P. Cunningham, "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering," *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, Pittsburgh, Pennsylvania, USA, 2006, pp. 377-384. doi: 10.1145/1143844.1143892
- [23] P. Melly, "AWS Architecture and Draw.io, a match made in developer heaven," *draw.io*, <https://drawio-app.com/blog/aws-architecture-and-draw-io-a-match-made-in-developer-heaven/> (Accessed July 24, 2023).
- [24] "AWS Command Line Reference - create-role," Amazon, [Online]. Available: <https://docs.aws.amazon.com/cli/latest/reference/iam/create-role.html> (Accessed: July 24, 2023).
- [25] "AWS Command Line Reference - attach-role-policy," Amazon, [Online]. Available: <https://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html> (Accessed: July 24, 2023).