

Assignment 3

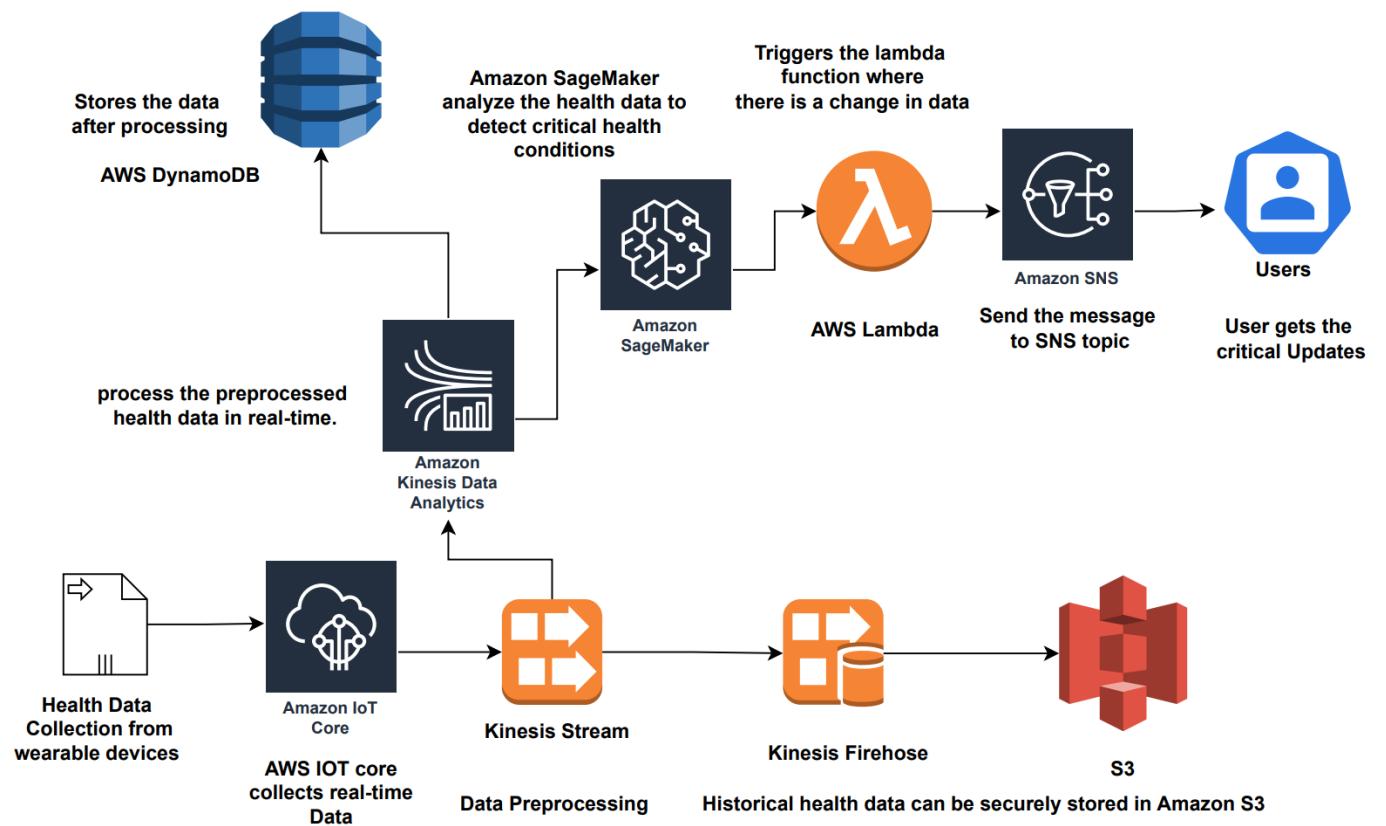
Part A

Scenario: In a remote and disaster-prone region, one organization faces challenges during natural calamities like earthquakes and floods. Communication breakdowns lower their ability to access critical health and location data, making it difficult to provide immediate assistance during emergencies. The organization plans to deploy a real-time health monitoring and emergency response system to tackle these issues. This system will quickly collect important health data from affected individuals using wearable devices and medical sensors, analyze it in real time, and coordinate emergency responses effectively to ensure that urgent medical aid reaches those in urgent need. The main goal of this system is to save lives, enhance the efficiency of disaster relief operations, and improve the overall resilience of the disaster effected community.

Use-Case:

Health Data Collection: Equipped with AWS IoT Core, the system gathers real-time health data, including heart rate, blood pressure, body temperature, and respiratory rate, from individuals affected by the disaster. This data is instantly transmitted through AWS Kinesis Data Streams, allowing immediate processing and analysis[1].

Real-Time Health Analytics: Powered by AWS Kinesis Data Analytics and advanced machine learning models from Amazon SageMaker, the system processes the incoming health data in real time[1]. It can accurately detect critical health conditions like cardiac arrest, respiratory distress, or abnormal vital signs, allowing it to prioritize emergency cases based on severity. When emergencies are seen, AWS Lambda functions, integrated with Amazon SNS, trigger real-time alerts to emergency response teams and medical personnel, enabling them to take swift action and allocate resources efficiently. The system securely stores historical health data in Amazon S3 using Kinesis Data Firehose for future analysis and disaster preparedness, further enhancing the community's resilience during future calamities[1].

Fig 1: Workflow Diagram (Draw.io - <https://app.diagrams.net/>)

Part B:

The screenshot shows the 'Create bucket' wizard in the AWS S3 console. The process is at the 'General configuration' step.

General configuration

- Bucket name:** sampledatab00917151
- AWS Region:** US East (N. Virginia) us-east-1
- Copy settings from existing bucket - optional:** Only the bucket settings in the following configuration are copied.
Choose bucket

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, resource point policies, or all in order to

Fig 2: Creating 1st Bucket sampledatab00917151

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠️ Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

Fig 3: Configuration of the bucket

The screenshot shows the third step of the AWS S3 Bucket Creation Wizard. It includes sections for Tags, Default encryption, Bucket Key, and Advanced settings, with a summary note at the bottom.

Tags (0) - optional
You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket. [Add tag](#)

Default encryption [Info](#)
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)
 Server-side encryption with Amazon S3 managed keys (SSE-S3)
 Server-side encryption with AWS Key Management Service keys (SSE-KMS)
 Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the [Storage tab](#) of the [Amazon S3 pricing page](#).

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

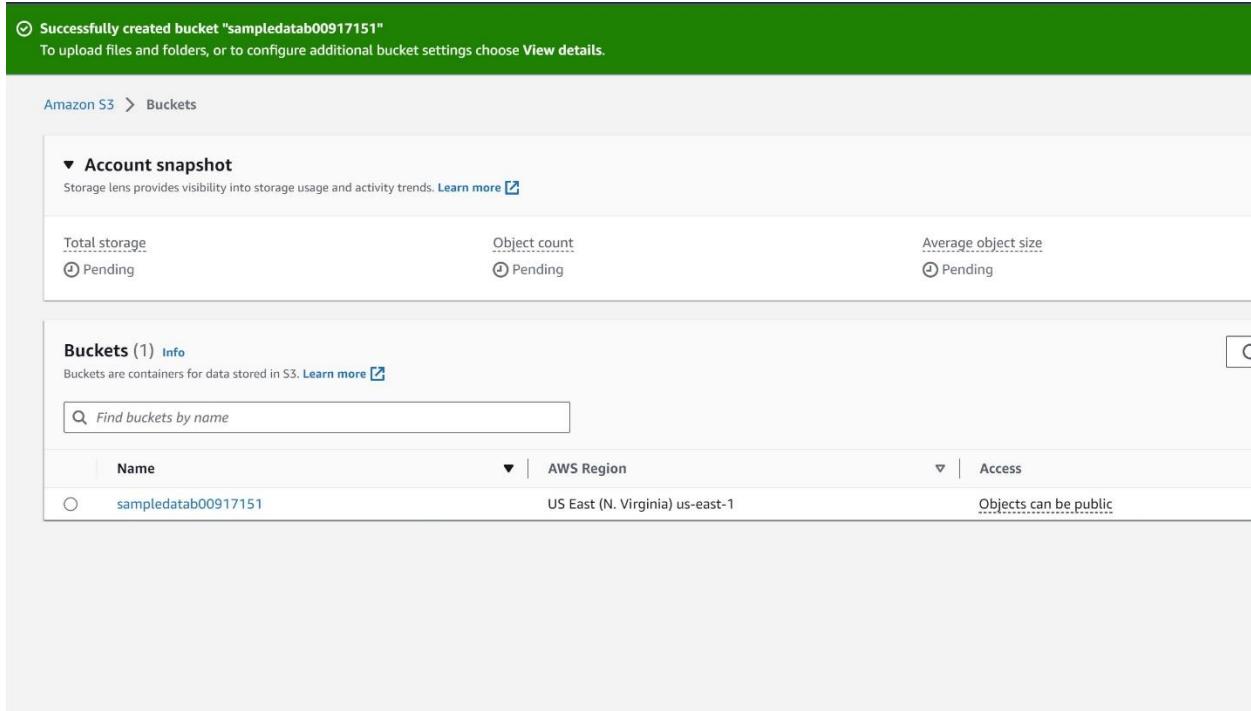
Disable
 Enable

► Advanced settings

Info After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

Fig 4: Creating bucket

Fig 5: 1st bucket Created

⌚ Successfully edited bucket policy.

Objects can be public

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets, turn on Block all public access. This will prevent public access to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly. If you have objects within this bucket, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Edit

Block all public access

⚠ Off

▶ Individual Block Public Access settings for this bucket

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowReadWrite",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": [  
                "s3:GetObject",  
                "s3:PutObject",  
                "s3:DeleteObject"  
            ],  
            "Resource": "arn:aws:s3:::sampleddatab00917151/*"  
        }  
    ]  
}
```

Fig 6: changing the bucket policy of the 1st bucket

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Fig 7: creating 2nd bucket tagsb00917151 with default setting as previous bucket

The screenshot shows the 'Edit bucket policy' page in the AWS S3 console. The navigation path is: Amazon S3 > Buckets > tagsb00917151 > Edit bucket policy. The title is 'Edit bucket policy' with an 'Info' link. Below it is a section titled 'Bucket policy' with a note: 'The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts.' A 'Learn more' link is also present. The 'Bucket ARN' is listed as arn:aws:s3:::tagsb00917151. The 'Policy' section contains a JSON code editor with the following policy:

```
1 ▼ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Sid": "AllowReadWrite",  
6             "Effect": "Allow",  
7             "Principal": "*",  
8             "Action": [  
9                 "s3:GetObject",  
10                "s3:PutObject",  
11                "s3:DeleteObject"  
12            ],  
13            "Resource": "arn:aws:s3:::tagsb00917151/*"  
14        }  
15    ]  
16 }
```

Fig 8: changing the bucket policy of 2nd bucket to allow public access

DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
 entityFrequencyDb

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

entity

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name

1 to 255 characters and case sensitive.

Table settings

Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

Customize settings
Use these advanced features to make DynamoDB work better for your needs.

Fig 9: Creating a DynamoDb table called entityFrequencyDb

Default table settings

These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#) [Create table](#)

Fig 10 : creating table with default config

Lambda > Functions > Create function

Create function Info

AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to do

Basic information

Function name
Enter a name that describes the purpose of your function.
extractFeatures
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.11

Architecture Info
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Fig 11: creating 1st lambda function called extractFeatures

Function name
Enter a name that describes the purpose of your function.
extractFeatures
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.11

Architecture Info
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole
View the LabRole role [on the IAM console](#).

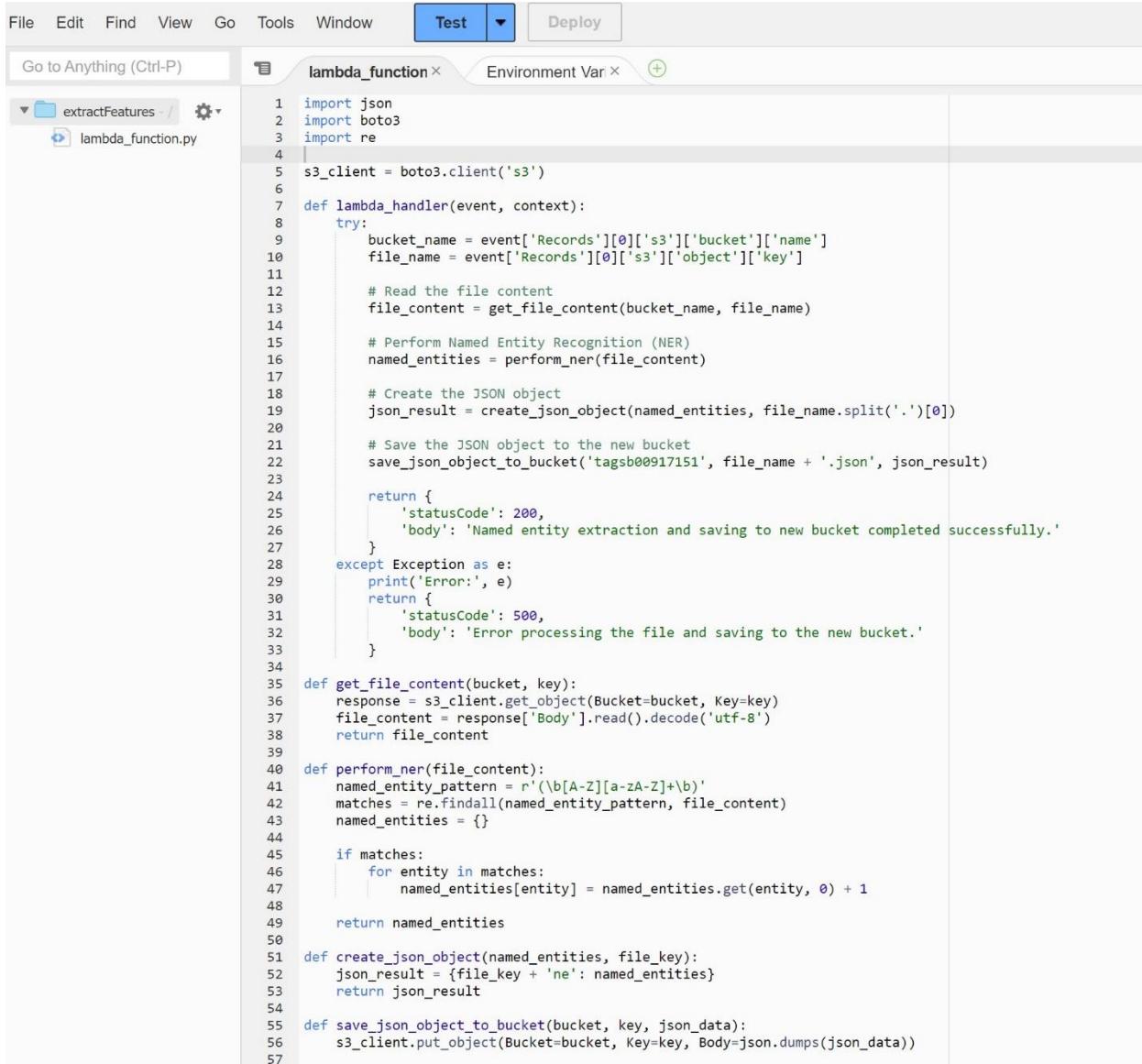
► Advanced settings

[Cancel](#) [Create function](#)

Fig 12: Assigning a default LabRole and language selection

The screenshot shows the AWS Lambda 'Create function' wizard. The first step is 'Function name', where 'accessDb' is entered. The 'Runtime' is set to 'Python 3.11'. Under 'Architecture', 'x86_64' is selected. In the 'Permissions' section, the 'Change default execution role' button is expanded, showing options for creating a new role or using an existing one; 'Use an existing role' is selected with 'LabRole' chosen from the dropdown. The 'Advanced settings' section is collapsed. At the bottom right are 'Cancel' and 'Create function' buttons.

Fig 13: creating a 2nd lambda function called accessDb with default config as 1st function.



```
File Edit Find View Go Tools Window Test Deploy
Go to Anything (Ctrl-P)
lambda_function Environment Var +
1 import json
2 import boto3
3 import re
4
5 s3_client = boto3.client('s3')
6
7 def lambda_handler(event, context):
8     try:
9         bucket_name = event['Records'][0]['s3']['bucket']['name']
10        file_name = event['Records'][0]['s3']['object']['key']
11
12        # Read the file content
13        file_content = get_file_content(bucket_name, file_name)
14
15        # Perform Named Entity Recognition (NER)
16        named_entities = perform_ner(file_content)
17
18        # Create the JSON object
19        json_result = create_json_object(named_entities, file_name.split('.')[0])
20
21        # Save the JSON object to the new bucket
22        save_json_object_to_bucket('tagsb00917151', file_name + '.json', json_result)
23
24        return {
25            'statusCode': 200,
26            'body': 'Named entity extraction and saving to new bucket completed successfully.'
27        }
28    except Exception as e:
29        print('Error:', e)
30        return {
31            'statusCode': 500,
32            'body': 'Error processing the file and saving to the new bucket.'
33        }
34
35    def get_file_content(bucket, key):
36        response = s3_client.get_object(Bucket=bucket, Key=key)
37        file_content = response['Body'].read().decode('utf-8')
38        return file_content
39
40    def perform_ner(file_content):
41        named_entity_pattern = r'(\b[A-Z][a-zA-Z]+\b)'
42        matches = re.findall(named_entity_pattern, file_content)
43        named_entities = {}
44
45        if matches:
46            for entity in matches:
47                named_entities[entity] = named_entities.get(entity, 0) + 1
48
49        return named_entities
50
51    def create_json_object(named_entities, file_key):
52        json_result = {file_key + 'ne': named_entities}
53        return json_result
54
55    def save_json_object_to_bucket(bucket, key, json_data):
56        s3_client.put_object(Bucket=bucket, Key=key, Body=json.dumps(json_data))
57
```

Fig 14: lambda code for extractFeatures

Trigger configuration [Info](#)

 S3
asynchronous aws storage ▾

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

X C

Bucket region: us-east-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events X ▾

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

Fig 14: setting trigger for extractFeatures lambda code with s3 bucket sampleddatab00917151

The screenshot shows a code editor interface with a tab bar at the top. The active tab is labeled "lambda_function". To its right are tabs for "Environment Var" and a "+" icon. On the left, there's a sidebar titled "Go to Anything (Ctrl-P)" with a dropdown menu showing "accessDB /" and a gear icon. Below the sidebar is a file tree with a single item: "lambda_function.py". The main workspace contains the following Python code:

```
1 import json
2 import boto3
3
4 s3_client = boto3.client('s3')
5 dynamodb_client = boto3.client('dynamodb')
6
7 def lambda_handler(event, context):
8     try:
9         for record in event['Records']:
10             bucket_name = record['s3']['bucket']['name']
11             file_name = record['s3']['object']['key']
12
13             # Read the JSON file content
14             file_content = get_file_content(bucket_name, file_name)
15             json_result = json.loads(file_content)
16
17             entities_key = file_name[:3] + "ne"
18             entities = json_result.get(entities_key, {})
19
20             # Update the DynamoDB table with the entities from the current file
21             update_dynamodb(entities)
22
23         return {
24             'statusCode': 200,
25             'body': 'DynamoDB table updated successfully.'
26         }
27     except Exception as e:
28         print('Error:', e)
29         return {
30             'statusCode': 500,
31             'body': 'Error updating the DynamoDB table.'
32         }
33
34 def get_file_content(bucket, key):
35     response = s3_client.get_object(Bucket=bucket, Key=key)
36     file_content = response['Body'].read().decode('utf-8')
37     return file_content
38
39 def update_dynamodb(entities):
40     table_name = 'entityFrequencyDb' # Replace with your actual DynamoDB table name
41
42     # Iterate through the entities object and update the DynamoDB table
43     for entity_name, entity_count in entities.items():
44         params = {
45             'TableName': table_name,
46             'Item': {
47                 'entity': {'S': entity_name},
48                 'count': {'N': str(entity_count)}
49             }
50         }
51
52         dynamodb_client.put_item(**params)
53
54     print('DynamoDB table updated.')
55
```

Fig 15: lambda code for accessDB

Trigger configuration [Info](#)

 S3 asynchronous aws storage ▾

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

X C

Bucket region: us-east-1

Event types
Select the events that you want to trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

▾

All object create events X

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. jpg

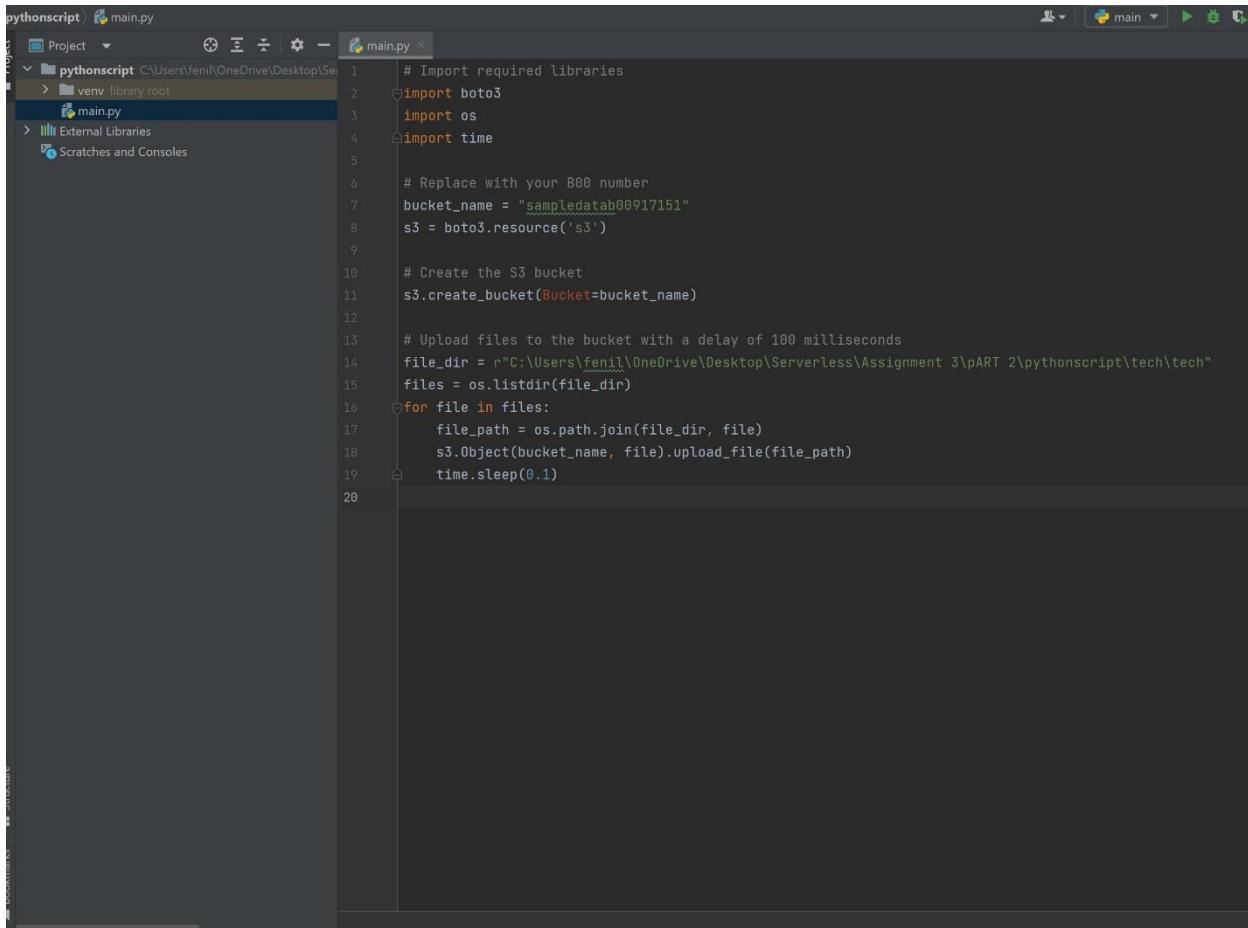
Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

Fig 16: setting trigger for accessDb lambda code with s3 bucket tagsb00917151



The screenshot shows a code editor window with a dark theme. On the left is a project sidebar titled "pythonscript" containing a "venv library root" folder with "main.py". The main editor area displays the following Python script:

```
# Import required libraries
import boto3
import os
import time

# Replace with your B00 number
bucket_name = "sampledatab00917151"
s3 = boto3.resource('s3')

# Create the S3 bucket
s3.create_bucket(Bucket=bucket_name)

# Upload files to the bucket with a delay of 100 milliseconds
file_dir = r"C:\Users\fenil\OneDrive\Desktop\Serverless\Assignment 3\pART 2\pythonscript\tech\tech"
files = os.listdir(file_dir)
for file in files:
    file_path = os.path.join(file_dir, file)
    s3.Object(bucket_name, file).upload_file(file_path)
    time.sleep(0.1)
```

Fig 17: python script for pushing 400 text file of zip to sampledatab00917151 bucket

Objects (401)

Name	Type	Last modified	Size	Storage class
001.txt	txt	July 27, 2023, 21:53:04 (UTC-03:00)	3.9 KB	Standard
002.txt	txt	July 27, 2023, 21:53:04 (UTC-03:00)	2.2 KB	Standard
003.txt	txt	July 27, 2023, 21:53:04 (UTC-03:00)	1.3 KB	Standard
004.txt	txt	July 27, 2023, 21:53:05 (UTC-03:00)	2.5 KB	Standard
005.txt	txt	July 27, 2023, 21:53:05 (UTC-03:00)	4.8 KB	Standard
006.txt	txt	July 27, 2023, 21:53:05 (UTC-03:00)	3.8 KB	Standard
007.txt	txt	July 27, 2023, 21:53:05 (UTC-03:00)	1.7 KB	Standard
008.txt	txt	July 27, 2023, 21:53:05 (UTC-03:00)	1.7 KB	Standard
009.txt	txt	July 27, 2023, 21:53:06 (UTC-03:00)	7.0 KB	Standard
010.txt	txt	July 27, 2023, 21:53:06 (UTC-03:00)	2.9 KB	Standard
011.txt	txt	July 27, 2023, 21:53:06 (UTC-03:00)	3.3 KB	Standard
012.txt	txt	July 27, 2023, 21:53:06 (UTC-03:00)	1.5 KB	Standard

Fig 18 : sampledatab00917151 uploaded text files

Objects (401)

Name	Type	Last modified	Size
001.txt.json	json	July 27, 2023, 21:53:06 (UTC-03:00)	644.0 B
002.txt.json	json	July 27, 2023, 21:53:06 (UTC-03:00)	292.0 B
003.txt.json	json	July 27, 2023, 21:53:06 (UTC-03:00)	167.0 B
004.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	334.0 B
005.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	629.0 B
006.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	536.0 B
007.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	312.0 B
008.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	378.0 B
009.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	1.2 KB
010.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	407.0 B
011.txt.json	json	July 27, 2023, 21:53:07 (UTC-03:00)	527.0 B

Fig 19: converted text files from sampledatab00917151 to tagsb00917151

```
C:\> Users > fenil > Downloads > {} 001.txt.json > ...
1   {"@01ne": {"Ink": 1, "Asia": 1, "The": 15, "Kyrgyz": 4, "Republic": 2, "Soviet": 2, "This": 2, "In": 3, "President": 1, "Askar": 1, "Akayev": 1, "Parliamentary": 1, "Presidential": 1, "Office": 1, "He": 1, "Losing": 1, "Kodak": 1, "File": 1, "Embassy": 1, "(ICA)": 1, "Agriculture": 1, "Trojan": 1, "Republic": 1, "Reseller": 1}}
```

Fig 20: tagsb00917151 s3 bucket's one file structure

entityFrequencyDb			
▶ Scan or query items			
Expand to query or scan items.			
✔ Completed. Read capacity units consumed: 0.5			
Items returned (881)			
□	entity (String)	▼	count
□	Ciphertrust,		1
□	Office		1
□	He		1
□	Losing		1
□	Kodak		1
□	File		1
□	Embassy		1
□	(ICA).		1
□	Agriculture		1
□	Trojan		1
□	Republic,		1
□	Reseller		1

Fig 21:named entity stored in entityFrequencyDb

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event Edit saved event

Event name

TestingLambda

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

s3-put

Event JSON

```
7   "eventTime": "1970-01-01T00:00:00.000Z",
8   "eventName": "ObjectCreated:Put",
9   "userIdentity": {
10     "principalId": "EXAMPLE"
11   },
12   "requestParameters": {
13     "sourceIPAddress": "127.0.0.1"
14   },
15   "responseElements": {
16     "x-amz-request-id": "EXAMPLE123456789",
17     "x-amz-id-2": "EXAMPLE1E123/5678abcdefhijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
18   },
19   "s3": {
20     "s3SchemaVersion": "1.0",
21     "configurationId": "testConfigRule",
22     "bucket": {
23       "name": "example-bucket",
24       "ownerIdentity": {
25         "principalId": "EXAMPLE"
26       },
27       "arn": "arn:aws:s3:::sampledatab00917151"
28     },
29     "object": {
30       "key": "test%2Fkey",
31       "size": 1024,
32       "eTag": "0123456789abcdef0123456789abcdef",
33       "sequencer": "0A1B2C3D4E5F678901"
34     }
35   }
36 }
```

Format JSON

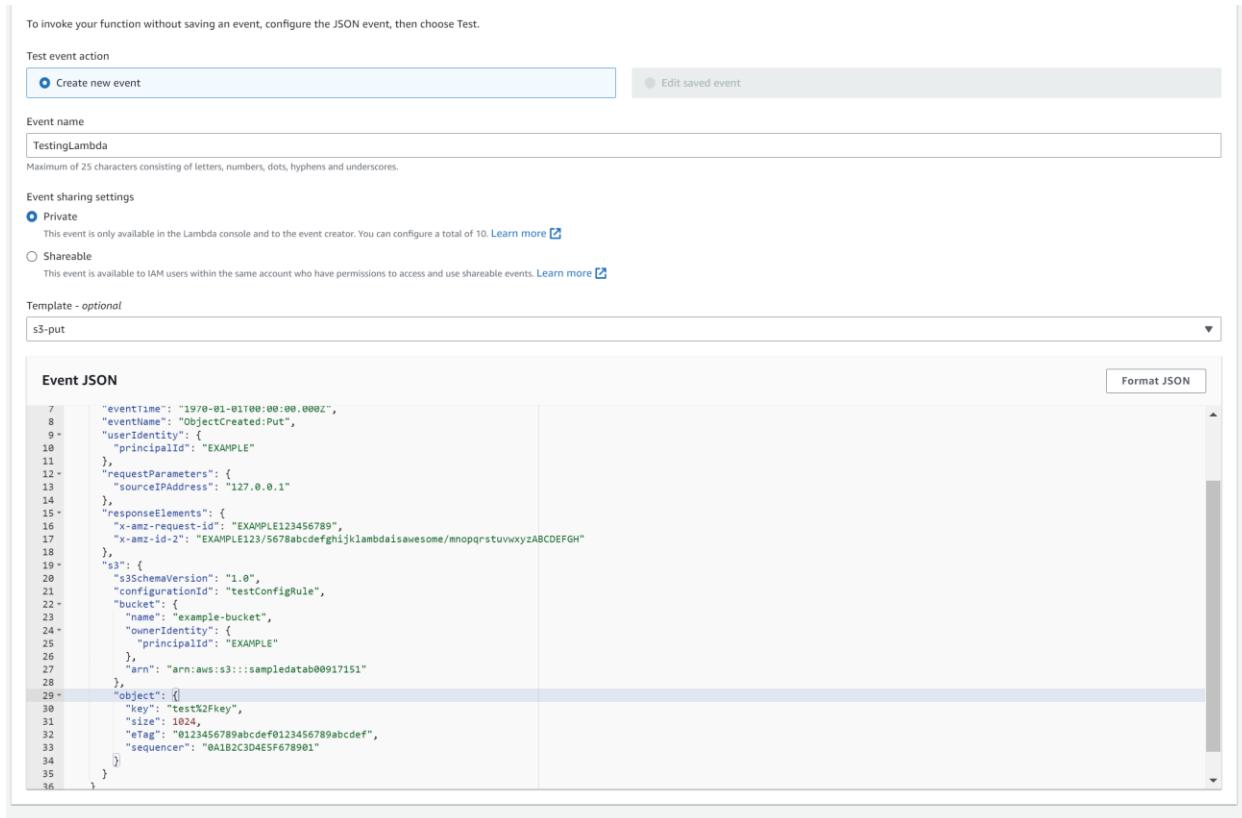


Fig 22: Functional Testing of the lambda function

Part C:

Amazon SQS > Queues > Create queue

Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

Standard Info
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

FIFO Info
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

ⓘ You can't change the queue type after you create a queue.

Name

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Fig 23: Creating a CartypeQueue

Configuration Info
Set the maximum message size, visibility to other consumers, and message retention.

<p>Visibility timeout <small>Info</small> <input type="text" value="30"/> Seconds</p> <p>Should be between 0 seconds and 12 hours.</p>	<p>Message retention period <small>Info</small> <input type="text" value="4"/> Days</p> <p>Should be between 1 minute and 14 days.</p>
<p>Delivery delay <small>Info</small> <input type="text" value="0"/> Seconds</p> <p>Should be between 0 seconds and 15 minutes.</p>	<p>Maximum message size <small>Info</small> <input type="text" value="256"/> KB</p> <p>Should be between 1 KB and 256 KB.</p>
<p>Receive message wait time <small>Info</small> <input type="text" value="0"/> Seconds</p> <p>Should be between 0 and 20 seconds.</p>	

Encryption Info
Amazon SQS provides in-transit encryption by default. To add at-rest encryption to your queue, enable server-side encryption.

Server-side encryption
 Disabled
 Enabled

Fig 24: setting configuration for CartypeQueue

Access policy Info

Define who can access your queue.

Choose method

Basic
Use simple criteria to define a basic access policy.

Advanced
Use a JSON object to define an advanced access policy.

Define who can send messages to the queue

Only the queue owner
Only the owner of the queue can send messages to the queue.

Only the specified AWS accounts, IAM users and roles
Only the specified AWS account IDs, IAM users and roles can send messages to the queue.

Define who can receive messages from the queue

Only the queue owner
Only the owner of the queue can receive messages from the queue.

Only the specified AWS accounts, IAM users and roles
Only the specified AWS account IDs, IAM users and roles can receive messages from the queue.

JSON (read-only)

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__owner_statement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "283622577067"
      },
      "Action": [
        "SQS:*"
      ],
      "Resource": "arn:aws:sqs:us-east-1:283622577067:CartypeQueue"
    }
  ]
}
```

Redrive allow policy - Optional Info

Identify which source queues can use this queue as the dead-letter queue.

Select which source queues can use this queue as the dead-letter queue.

Disabled

Enabled

Fig 25: all the setting of the queue as default

Redrive allow policy - Optional Info

Identify which source queues can use this queue as the dead-letter queue.

Select which source queues can use this queue as the dead-letter queue.

Disabled

Enabled

Dead-letter queue - Optional Info

Send undeliverable messages to a dead-letter queue.

Set this queue to receive undeliverable messages.

Disabled

Enabled

Tags - Optional Info

A tag is a label assigned to an AWS resource. Use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	Remove
<input type="text"/> <small>Enter key</small>	<input type="text"/> <small>Enter value</small>	<input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>		

You can add 49 more tags.

Fig 26: creating a queue with default setting

Amazon SQS > Queues > Create queue

Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

Standard Info
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

FIFO Info
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

Info You can't change the queue type after you create a queue.

Name
AccessoriesQueue

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Fig 27: Creating a 2nd queue called AccessoriesQueue

"SQS:*"
1,
"Resource": "arn:aws:sqs:us-east-1:283622577067:AccessoriesQueue"
}

Redrive allow policy - Optional Info
Identify which source queues can use this queue as the dead-letter queue.

Select which source queues can use this queue as the dead-letter queue.

Disabled
 Enabled

Dead-letter queue - Optional Info
Send undeliverable messages to a dead-letter queue.

Set this queue to receive undeliverable messages.

Disabled
 Enabled

Tags - Optional Info
A tag is a label assigned to an AWS resource. Use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	
<input type="text" value="Enter key"/>	<input type="text" value="Enter value"/>	<input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>		
You can add 49 more tags.		

Cancel

Fig 28: creating queue with default setting

Amazon SQS > Queues > Create queue

Create queue

Details

Type

Choose the queue type for your application or cloud infrastructure.

Standard Info
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

FIFO Info
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

i You can't change the queue type after you create a queue.

Name

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Configuration Info

Set the maximum message size, visibility to other consumers, and message retention.

Visibility timeout Info

Seconds

Should be between 0 seconds and 12 hours.

Message retention period Info

Days

Should be between 1 minute and 14 days.

Delivery delay Info

Seconds

Should be between 0 seconds and 15 minutes.

Maximum message size Info

KB

Should be between 1 KB and 256 KB.

Receiving message wait time Info

Fig 29: Creating a 3rd queue called AddressQueue

Redrive allow policy - Optional Info

Identify which source queues can use this queue as the dead-letter queue.

Select which source queues can use this queue as the dead-letter queue.

Disabled
 Enabled

Dead-letter queue - Optional Info

Send undeliverable messages to a dead-letter queue.

Set this queue to receive undeliverable messages.

Disabled
 Enabled

Tags - Optional Info

A tag is a label assigned to an AWS resource. Use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	
<input type="text" value="Enter key"/>	<input type="text" value="Enter value"/>	<input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>		
You can add 49 more tags.		

Fig 30: Creating queue with other default setting

Amazon SNS > Topics > Create topic

Create topic

Details

Type [Info](#)
Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

Maximum 100 characters.

Fig 31 : Creating a standard SNS topic HalifaxTaxiTopic

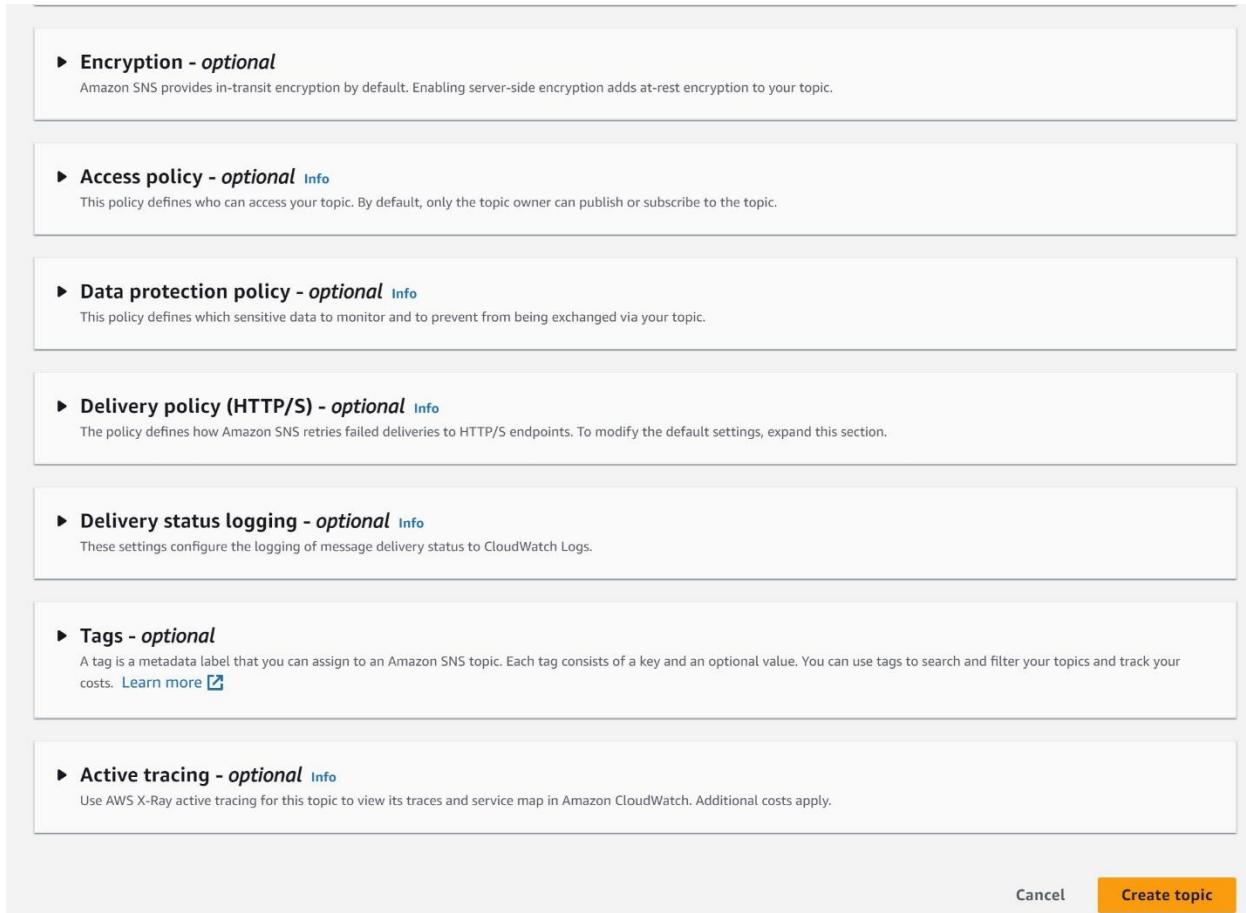


Fig 32: setting it as default setting and creating the topic

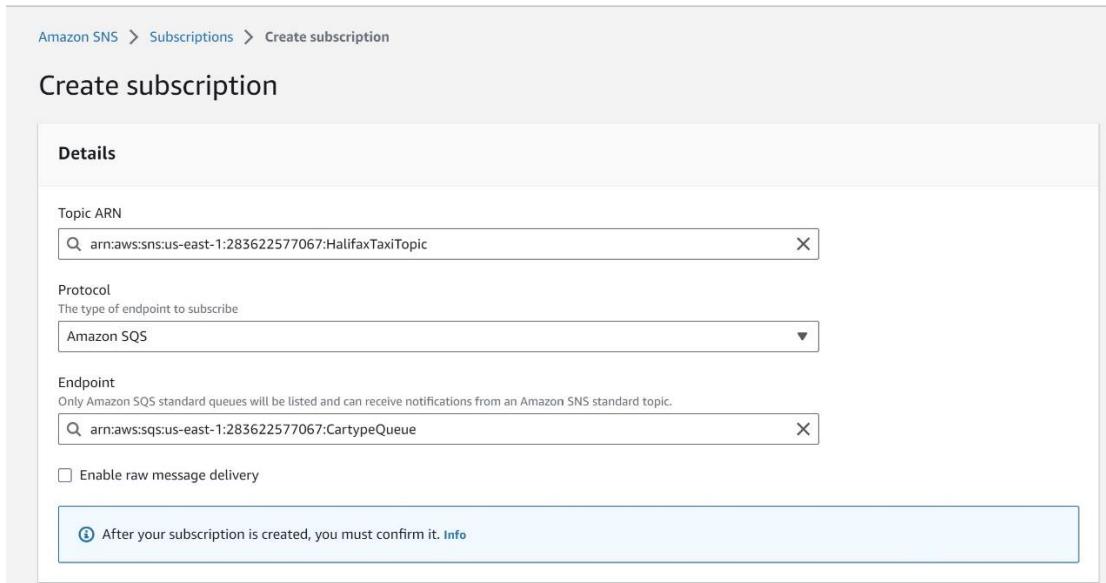


Fig 33: Creating a subscription with sqs queue CartypeQueue

▼ **Subscription filter policy - optional** Info

This policy filters the messages that a subscriber receives.

Subscription filter policy

Filter policy scope Info
Determines how the filter policy will be applied to the message

Message attributes
Filter policy will be applied to the message attributes.

Message body
Filter policy will be applied to the message body.

JSON editor

```
1 {
2   "car_type": ["Compact", "Mid-size Sedan", "SUV", "Luxury"]
3 }
4
```

▼ **Redrive policy (dead-letter queue) - optional** Info

Send undeliverable messages to a dead-letter queue.

Redrive policy (dead-letter queue)

Cancel **Create subscription**

Fig 34: applying filter policy for that subscription

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
 X

Protocol
The type of endpoint to subscribe
 ▼

Endpoint
Only Amazon SQS standard queues will be listed and can receive notifications from an Amazon SNS standard topic.
 X

Enable raw message delivery

Info After your subscription is created, you must confirm it. [Info](#)

Fig 35: Creating a 2nd subscription with sqs queue AccessoriesQueue

▼ **Subscription filter policy - optional** Info

This policy filters the messages that a subscriber receives.

Subscription filter policy

Filter policy scope | Info
Determines how the filter policy will be applied to the message

Message attributes
Filter policy will be applied to the message attributes.

Message body
Filter policy will be applied to the message body.

JSON editor

```
1 {
2   "addons": [
3     "GPS",
4     "Camera"
5   ]
6 }
7
```

▼ **Redrive policy (dead-letter queue) - optional** Info

Send undeliverable messages to a dead-letter queue.

Redrive policy (dead-letter queue)

Cancel Create subscription

Fig 36: applying filter policy for that subscription

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
arn:aws:sns:us-east-1:283622577067:HalifaxTaxiTopic

Protocol
The type of endpoint to subscribe
Amazon SQS

Endpoint
Only Amazon SQS standard queues will be listed and can receive notifications from an Amazon SNS standard topic.
arn:aws:sqs:us-east-1:283622577067:AddressQueue

Enable raw message delivery

Info After your subscription is created, you must confirm it. [Info](#)

► Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

Fig 37: Creating a 3rd subscription with sqs queue AddressQueue

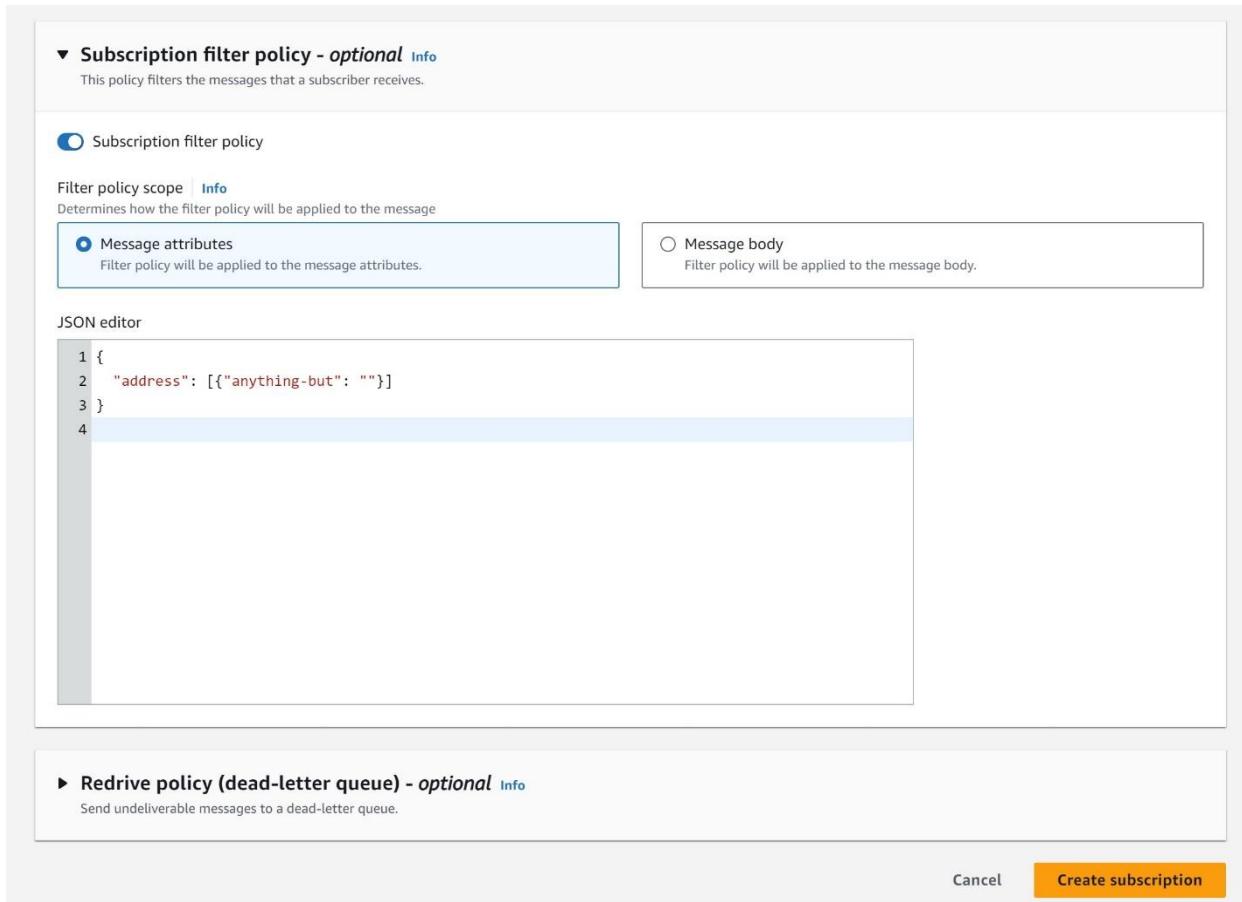


Fig 38: applying filter policy for that subscription

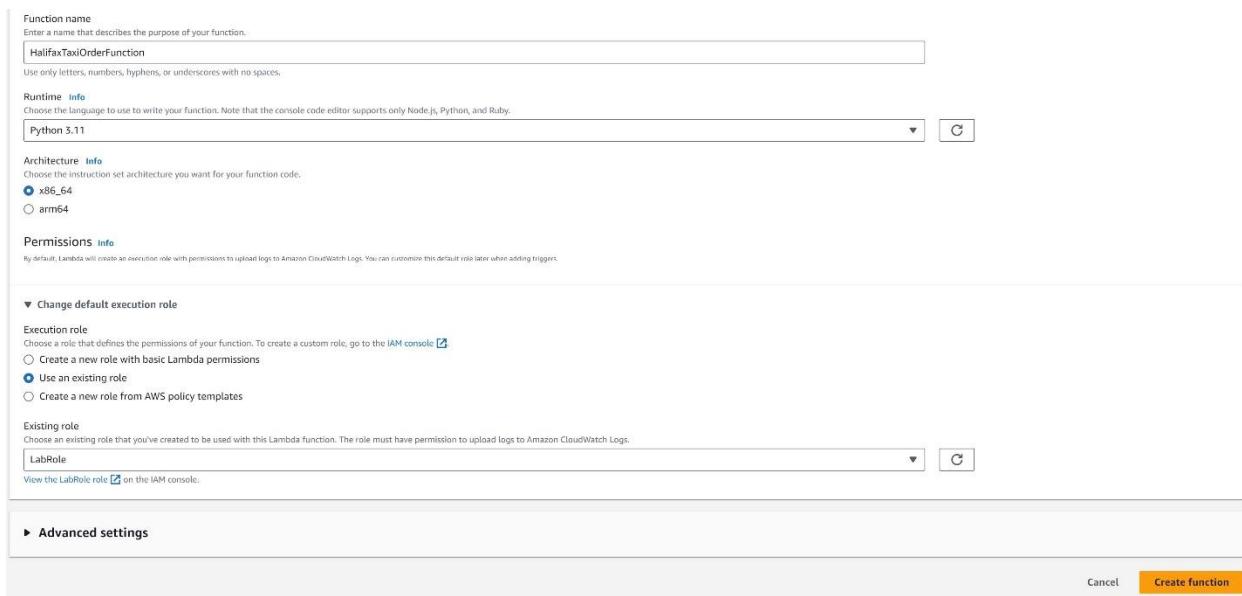


Fig 39: creating HalifaxTaxiOrderFunction with LabRole

The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes File, Edit, Find, View, Go, Tools, Window, Test, Deploy, Environment Var, and Execution results tabs. The main area displays the code for the `lambda_function`. The code is a Python script named `lambda_function.py` located in the `HalifaxTaxiOrderFun` folder. The code defines a `lambda_handler` function that sends three types of messages to an SNS topic: car type, accessories, and address.

```
1 import boto3
2 import json
3 import random
4
5 CAR_TYPES = ["Compact", "Mid-size Sedan", "SUV", "Luxury"]
6 ADDONS = ["GPS", "Camera"]
7 CLIENT_ADDRESSES = ["431 University Avenue", "123 windsor Street", "789 quinpool road"]
8
9 sns = boto3.client('sns')
10 topic_arn = 'arn:aws:sns:us-east-1:283622577067:HalifaxTaxiTopic'
11
12 def lambda_handler(event, context):
13     car_type = random.choice(CAR_TYPES)
14     addons = random.sample(ADDONS, random.randint(0, len(ADDONS)))
15     address = random.choice(CLIENT_ADDRESSES)
16
17     # Send car_type message
18     sns.publish(
19         TopicArn=topic_arn,
20         Message=json.dumps({"car_type": car_type}),
21         MessageAttributes={
22             'car_type': {
23                 'DataType': 'String',
24                 'StringValue': 'CarType'
25             }
26         }
27     )
28
29     # Send accessories message
30     sns.publish(
31         TopicArn=topic_arn,
32         Message=json.dumps({"addons": addons}),
33         MessageAttributes={
34             'addons': {
35                 'DataType': 'String',
36                 'StringValue': 'Accessories'
37             }
38         }
39     )
40
41     # Send address message
42     sns.publish(
43         TopicArn=topic_arn,
44         Message=json.dumps({"address": address}),
45         MessageAttributes={
46             'address': {
47                 'DataType': 'String',
48                 'StringValue': 'Address'
49             }
50         }
51     )
52
53     return {
54         'statusCode': 200,
55         'body': json.dumps('Order messages sent to the SNS topic.')
56     }
57
```

Fig 40: python code for HalifaxTaxiOrderFunction

Lambda > Functions > Create function

Create function Info

AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to dep

Basic information

Function name
Enter a name that describes the purpose of your function.
HalifaxTaxiOrderNotifier

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.11

Architecture Info
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console [\[\]](#).

Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Fig 41: creating HalifaxTaxiOrderNotifier with LabRole

Create topic

Details

Type Info
Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name
HalifaxTopicEmailNotification

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* Info
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.
My Topic

Maximum 100 characters.

▼ Encryption - *optional*
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Encryption [Learn more \[\]](#)
Enabling server side encryption adds at-rest encryption to your topic. Amazon SNS encrypts your message as soon as it is received. The message is decrypted immediately prior to delivery.

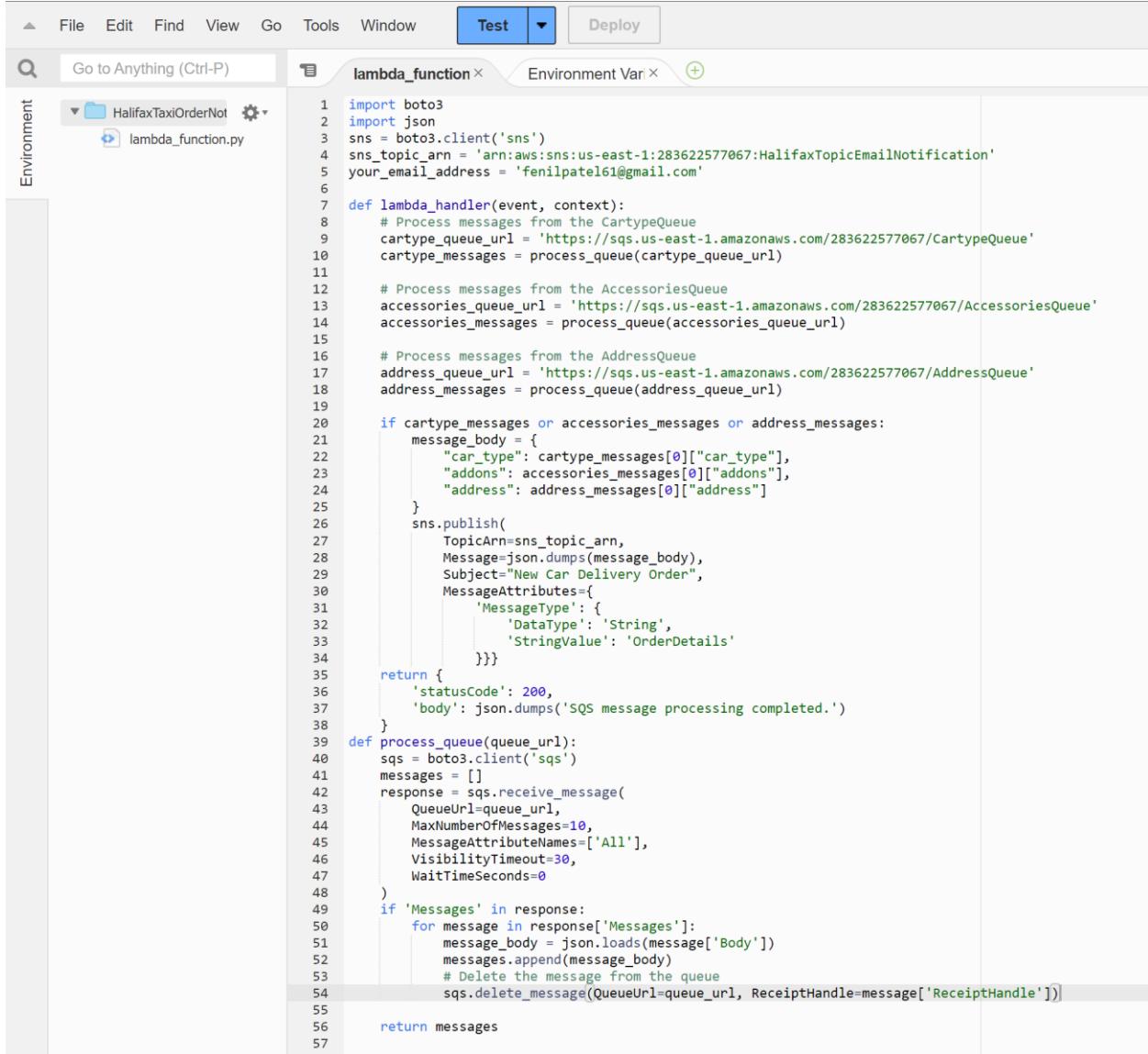
Fig 42: creating a SNS topic for email notification called HalifaxTopicEmailNotification

The screenshot shows the 'Create topic' wizard on the AWS SNS console. It displays several optional configuration sections:

- Encryption - optional**: Describes server-side encryption and provides a link to learn more.
- Access policy - optional**: Describes the access policy and provides a link to learn more.
- Data protection policy - optional**: Describes the data protection policy and provides a link to learn more.
- Delivery policy (HTTP/S) - optional**: Describes the delivery policy and provides a link to learn more.
- Delivery status logging - optional**: Describes delivery status logging and provides a link to learn more.
- Tags - optional**: Describes tags and provides a link to learn more.
- Active tracing - optional**: Describes active tracing and provides a link to learn more.

At the bottom right, there are 'Cancel' and 'Create topic' buttons.

Fig 43: Setting up sns with default settings



The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes File, Edit, Find, View, Go, Tools, Window, Test (highlighted in blue), and Deploy. On the left, the Environment sidebar shows a folder named 'HalifaxTaxiOrderNot' with a sub-item 'lambda_function.py'. The main workspace displays the following Python code:

```

1 import boto3
2 import json
3 sns = boto3.client('sns')
4 sns_topic_arn = 'arn:aws:sns:us-east-1:283622577067:HalifaxTopicEmailNotification'
5 your_email_address = 'fenilpatel61@gmail.com'
6
7 def lambda_handler(event, context):
8     # Process messages from the CartypeQueue
9     cartype_queue_url = 'https://sns.us-east-1.amazonaws.com/283622577067/CartypeQueue'
10    cartype_messages = process_queue(cartype_queue_url)
11
12    # Process messages from the AccessoriesQueue
13    accessories_queue_url = 'https://sns.us-east-1.amazonaws.com/283622577067/AccessoriesQueue'
14    accessories_messages = process_queue(accessories_queue_url)
15
16    # Process messages from the AddressQueue
17    address_queue_url = 'https://sns.us-east-1.amazonaws.com/283622577067/AddressQueue'
18    address_messages = process_queue(address_queue_url)
19
20    if cartype_messages or accessories_messages or address_messages:
21        message_body = {
22            "car_type": cartype_messages[0]["car_type"],
23            "addons": accessories_messages[0]["addons"],
24            "address": address_messages[0]["address"]
25        }
26        sns.publish(
27            TopicArn=sns_topic_arn,
28            Message=json.dumps(message_body),
29            Subject="New Car Delivery Order",
30            MessageAttributes={
31                'MessageType': {
32                    'DataType': 'String',
33                    'StringValue': 'OrderDetails'
34                }
35            }
36        )
37        return {
38            'statusCode': 200,
39            'body': json.dumps('SQS message processing completed.')
40        }
41    def process_queue(queue_url):
42        sqs = boto3.client('sns')
43        messages = []
44        response = sqs.receive_message(
45            QueueUrl=queue_url,
46            MaxNumberOfMessages=10,
47            MessageAttributeNames=['All'],
48            VisibilityTimeout=30,
49            WaitTimeSeconds=0
50        )
51        if 'Messages' in response:
52            for message in response['Messages']:
53                message_body = json.loads(message['Body'])
54                messages.append(message_body)
55                # Delete the message from the queue
56                sqs.delete_message(QueueUrl=queue_url, ReceiptHandle=message['ReceiptHandle'])
57
58    return messages

```

Fig 44: Lambda code of HalifaxTaxiOrderNotification for fetching messages from 3 sqs and sending sns notification to email user

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
 X

Protocol
The type of endpoint to subscribe
 ▼

Endpoint
An email address that can receive notifications from Amazon SNS.

ⓘ After your subscription is created, you must confirm it. [Info](#)

► Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

► Redrive policy (dead-letter queue) - optional [Info](#)
Send undeliverable messages to a dead-letter queue.

[Cancel](#) Create subscription

Fig 45: creating an email subscription for the user

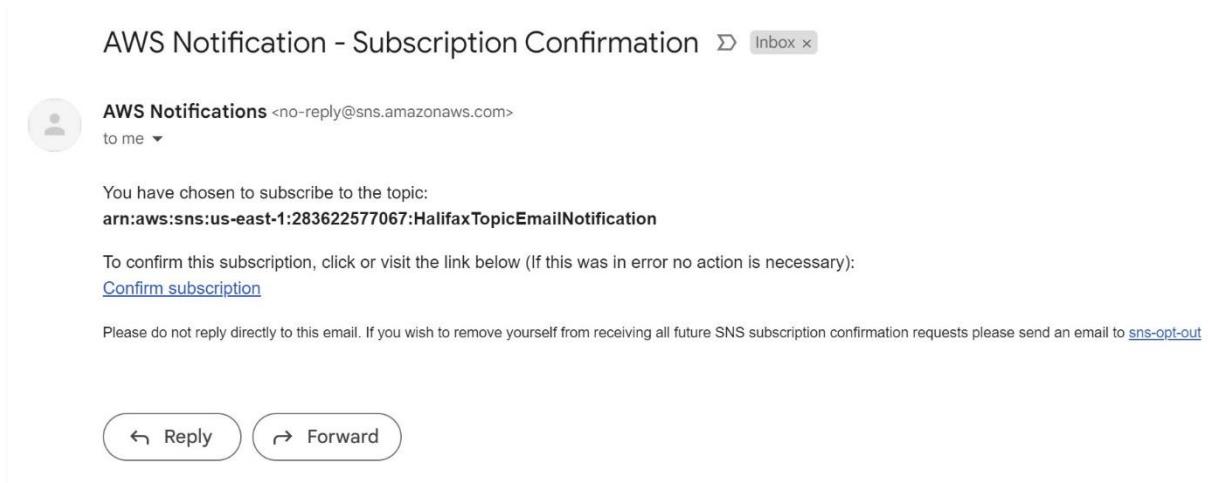


Fig 46: user receives the confirmation through mail

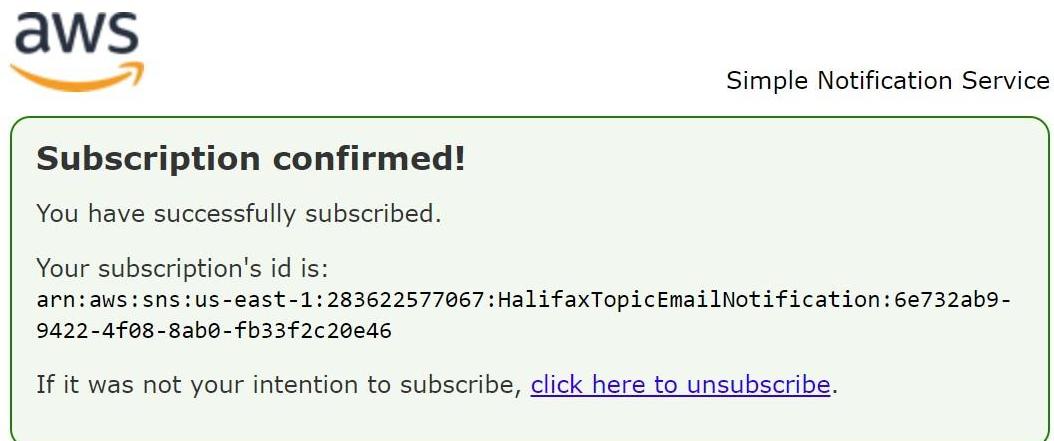


Fig 47: User approve confirmation

The screenshot shows the AWS SNS Topics page with the following details:

Topic Name: HalifaxTopicEmailNotification

Details:

- Name: HalifaxTopicEmailNotification
- ARN: arn:aws:sns:us-east-1:283622577067:HalifaxTopicEmailNotification
- Type: Standard
- Display name: -
- Topic owner: 283622577067

Subscriptions: Subscriptions (1)

ID	Endpoint	Status	Protocol
6e732ab9-9422-4f08-8ab0-fb33f2c20e46	fenilpatel61@gmail.com	Confirmed	EMAIL

Fig 48: Subscription activated

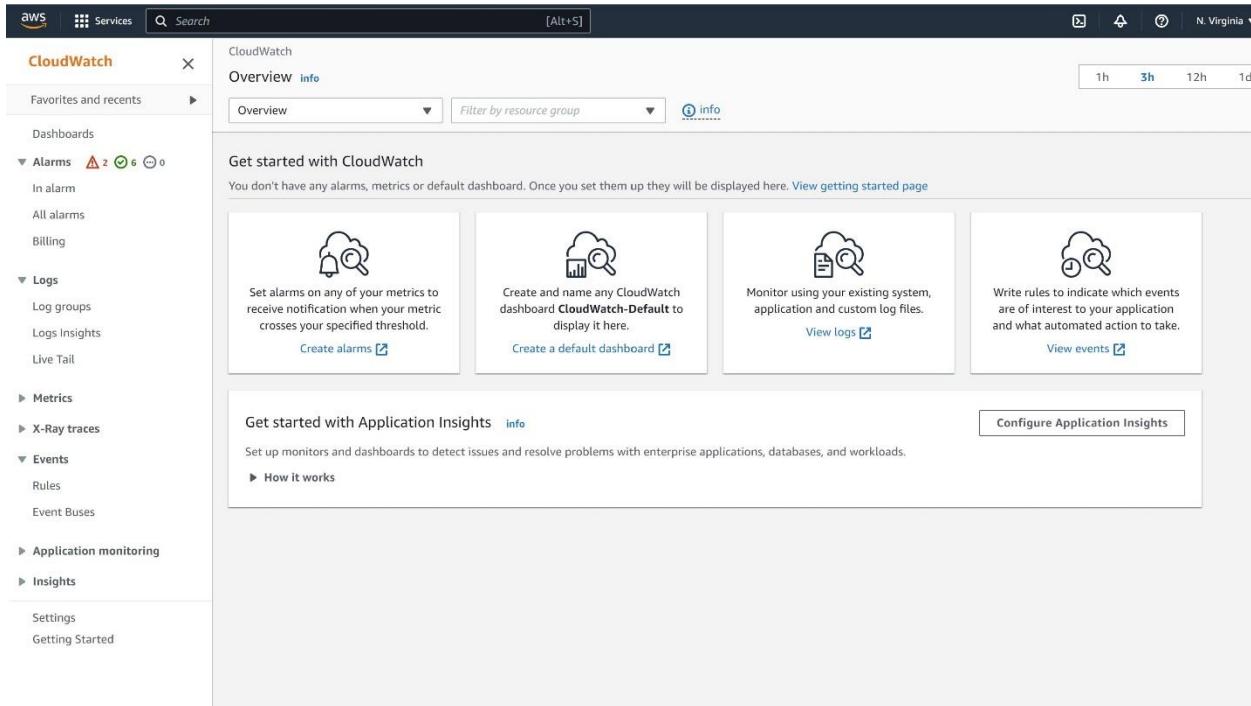


Fig 49: setting an event for lambda trigger every 2 minutes through CloudWatch

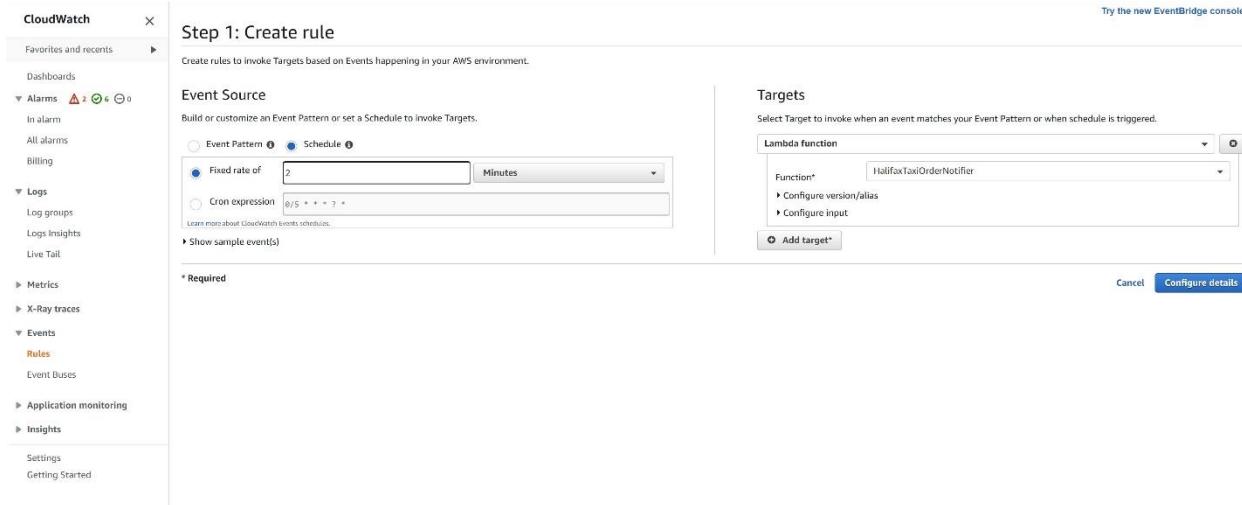


Fig 50: Creating a rule for that lambda trigger in cloudwatch

The screenshot shows the AWS CloudWatch Rules interface. On the left, there's a sidebar with navigation links like CloudWatch, Favorites and recent, Dashboards, Alarms, Logs, Metrics, X-Ray traces, Events, Application monitoring, Insights, Settings, and Getting Started. The main area is titled 'Rules' and contains a success message: 'Success Rule OrderCheck was created.' Below this is a table listing six rules:

Status	Name	Description
Green	MonitoringRule	MonitoringRule
Green	OrderCheck	
Green	resourceFunctionRule	
Green	voc-codebuild-cw-rule	codebuild build state change events
Green	voc-ec2-cw-rule	ec2 state change events
Green	voc-rds-cw-rule	rds all events

Fig 51: OrderCheck Rule has been created

The screenshot shows the AWS Lambda Function overview and code editor for the 'HalifaxTaxiOrderFunction'. In the top section, it shows the function name 'HalifaxTaxiOrderFunction', 'Layers (0)', and a '+ Add destination' button. Below this, there are sections for 'Description', 'Last modified 1 hour ago', 'Function ARN', and 'Function URL'. At the bottom, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected, showing the code source. The code editor displays a Python file named 'lambda_function.py' with the following content:

```

# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows,
# tutorials, examples and documentation.

# import libraries
import json
import boto3
from botocore.exceptions import ClientError
sns = boto3.client('sns')

# Set the SNS topic ARN
topic_arn = "arn:aws:sns:us-east-1:283622577067:OrderTopic"

def lambda_handler(event, context):
    # Create a new message
    response = sns.publish(
        TopicArn=topic_arn,
        Message="Order messages sent to the SNS topic."
    )
    print(response)

```

Fig 52: testing 1st lambda function HalifaxTaxiOrderFunction

Queues (4)							<input type="button" value="Create"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	<input type="button" value="Send and receive"/>
	Name	Type	Created	Messages available	Messages in flight	Encryption				
○	AccessoriesQueue	Standard	2023-07-27T12:57:03:00	1	0	Disabled				
○	AddressQueue	Standard	2023-07-27T12:58:03:00	1	0	Disabled				
○	CartypeQueue	Standard	2023-07-27T12:56:03:00	1	0	Disabled				
○	TriviaGameNotificationAlertsQueue	Standard	2023-07-20T16:34:03:00	0	0	Disabled				

Fig 53: sns topic submitted from 1st lambda and then the message has been received to 3 sqs queues with filter policy applied

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 .	
▶ 2023-07-27T18:50:50.656-03:00	START RequestId: edd63870-5983-4c20-9ce0-6d3e4eaa4e61 Version: \$LATEST
▶ 2023-07-27T18:50:51.372-03:00	END RequestId: edd63870-5983-4c20-9ce0-6d3e4eaa4e61
▶ 2023-07-27T18:50:51.372-03:00	REPORT RequestId: edd63870-5983-4c20-9ce0-6d3e4eaa4e61 Duration: 715.99 ms Billed Duration: 716 ms Memory Size: 1024
▶ 2023-07-27T18:52:50.573-03:00	START RequestId: 99cd14d0-27fd-4eb0-a562-af19b60a8211 Version: \$LATEST
▶ 2023-07-27T18:52:51.313-03:00	END RequestId: 99cd14d0-27fd-4eb0-a562-af19b60a8211
▶ 2023-07-27T18:52:51.313-03:00	REPORT RequestId: 99cd14d0-27fd-4eb0-a562-af19b60a8211 Duration: 739.35 ms Billed Duration: 740 ms Memory Size: 1024
▶ 2023-07-27T18:54:50.817-03:00	START RequestId: 478d2415-47e5-4e66-9446-84843cea8e83 Version: \$LATEST
▶ 2023-07-27T18:54:51.555-03:00	END RequestId: 478d2415-47e5-4e66-9446-84843cea8e83
▶ 2023-07-27T18:54:51.555-03:00	REPORT RequestId: 478d2415-47e5-4e66-9446-84843cea8e83 Duration: 738.35 ms Billed Duration: 739 ms Memory Size: 1024
▶ 2023-07-27T18:56:50.707-03:00	START RequestId: 01ed840e-4f1a-4479-9014-dd7c34e272bf Version: \$LATEST
▶ 2023-07-27T18:56:51.599-03:00	END RequestId: 01ed840e-4f1a-4479-9014-dd7c34e272bf
▶ 2023-07-27T18:56:51.599-03:00	REPORT RequestId: 01ed840e-4f1a-4479-9014-dd7c34e272bf Duration: 891.67 ms Billed Duration: 892 ms Memory Size: 1024
▶ 2023-07-27T18:58:50.728-03:00	START RequestId: f744885e-f411-49bd-98d1-0278d06e5680 Version: \$LATEST
▶ 2023-07-27T18:58:51.435-03:00	END RequestId: f744885e-f411-49bd-98d1-0278d06e5680
▶ 2023-07-27T18:58:51.435-03:00	REPORT RequestId: f744885e-f411-49bd-98d1-0278d06e5680 Duration: 706.52 ms Billed Duration: 707 ms Memory Size: 1024
No newer events at this moment. Auto retry paused . Resume	

Fig 54: Cloud watch logs for the 2nd lambda

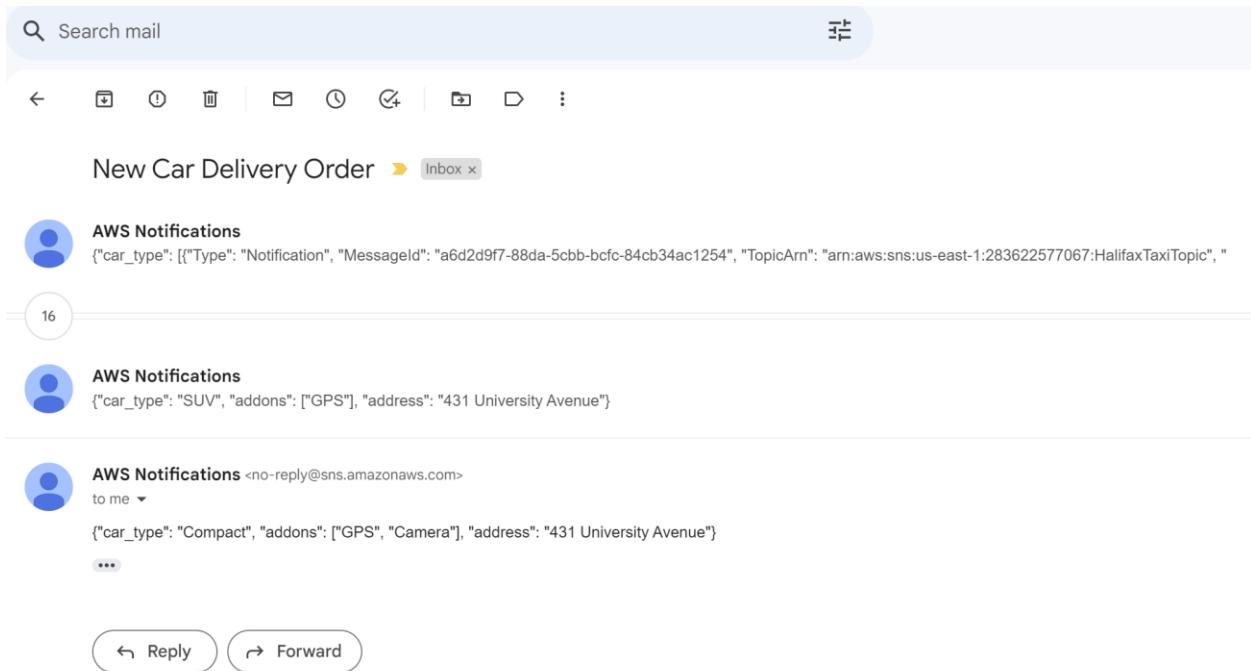
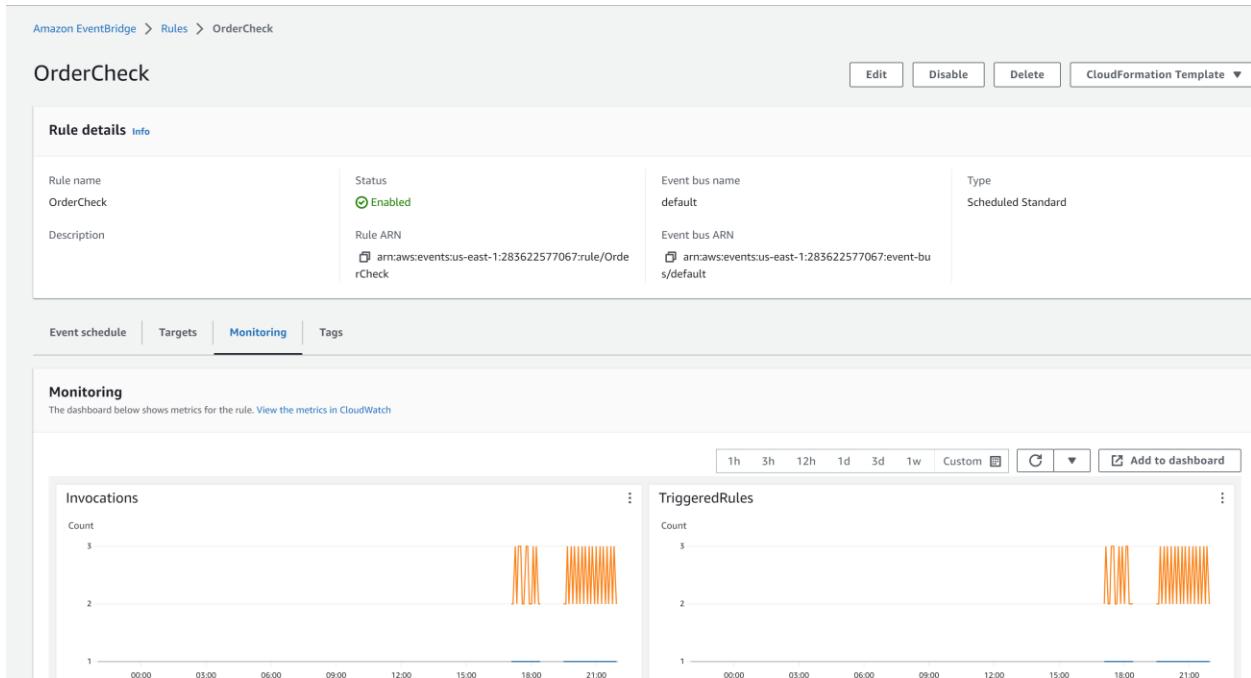


Fig 55: users receive the email for the notification

Fig 56: monitoring of OrderCheck rule for 2nd lambda trigger

References :

- [1] Walmsley, G., & Bie, J. (1983). *Kinesis*. Amazon. <https://aws.amazon.com/kinesis/>