

ASSIGNMENT 3: PART A

TASK A: Build, deploy, and run a Containerized Application using GCP.

LINK TO GITLAB:

https://git.cs.dal.ca/rtyagi/csci5410_b00872269_ruhirajnish_tyagi-/tree/main/assignment-3

Language used: Python 3.8.9

Library used for AWS SDK: boto3 [1].

Step 1: Buckets creation and uploading files to each programmatically .

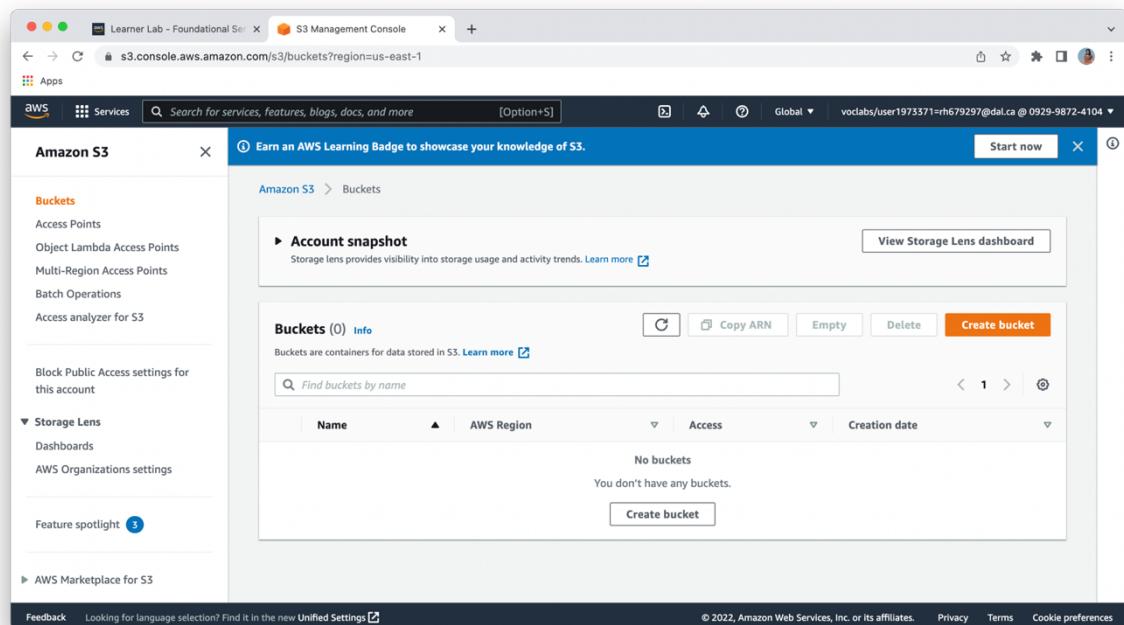


Figure 1. Screenshot of Amazon S3 console BEFORE creating buckets.

```

bucketS3 - main.py
Project  main.py  createBucket.py  uploadFile.py  extractFeatures.py  accessDB.py
bucketS3 -> /PycharmProjects/bucketS3
  tech
    001.txt
    002.txt
    003.txt
    004.txt
    005.txt
    006.txt
  venv
    accessDB.py
    createBucket.py
    extractFeatures.py
    main.py
    uploadFile.py
External Libraries
Scratches and Consoles

buckets3 - main.py
1 from createBucket import create_bucket
2 from uploadFile import upload_file
3
4 if __name__ == '__main__':
5     print('Assignment 3.....')
6     create_bucket('us-east-1')
7     upload_file('source1b00872269')

Run: main
/main
/Users/ruhityagi/PycharmProjects/bucketS3/venv/bin/python /Users/ruhityagi/PycharmProjects/bucketS3/main.py
Assignment 3.....  
Created the bucket successfully!
Process finished with exit code 0

```

Figure 2. Screenshot of main.py for creating bucket.

```

bucketS3 - createBucket.py
Project  main.py  createBucket.py  uploadFile.py  extractFeatures.py  accessDB.py
bucketS3 -> /PycharmProjects/bucketS3
  tech
    001.txt
    002.txt
    003.txt
    004.txt
    005.txt
    006.txt
  venv
    accessDB.py
    createBucket.py
    extractFeatures.py
    main.py
    uploadFile.py
External Libraries
Scratches and Consoles

buckets3 - createBucket.py
1 import logging
2 import boto3
3 from botocore.exceptions import ClientError
4
5 def create_bucket(region):
6     # Create bucket
7     try:
8         s3_client = boto3.client('s3', region_name=region,
9             aws_access_key_id='ASIAJLZHM5IE6S2M65R',
10            aws_secret_access_key='BTj4iACYx70ighVGt7Ig5FcQYQ4b1xyyRuEcJgf',
11            aws_session_token='FwoGZXIvYXdzENX//////////wEaOKHv2stjLZhAHN1SLAAV4M1JXy0uZn1ISS9YI'
12            )
13
14         #location = {'LocationConstraint': region}
15         s3_client.create_bucket(Bucket='source1b00872269')
16         print('Created the bucket successfully!')
17     except ClientError as e:
18         logging.error(e)
19         return False
20     return True
21
create_bucket() > try

Run: main
/main
/Users/ruhityagi/PycharmProjects/bucketS3/venv/bin/python /Users/ruhityagi/PycharmProjects/bucketS3/main.py
Assignment 3.....  
Created the bucket successfully!
Process finished with exit code 0

```

Figure 3. Screenshot of successful creation of bucket **source1b00872269**.

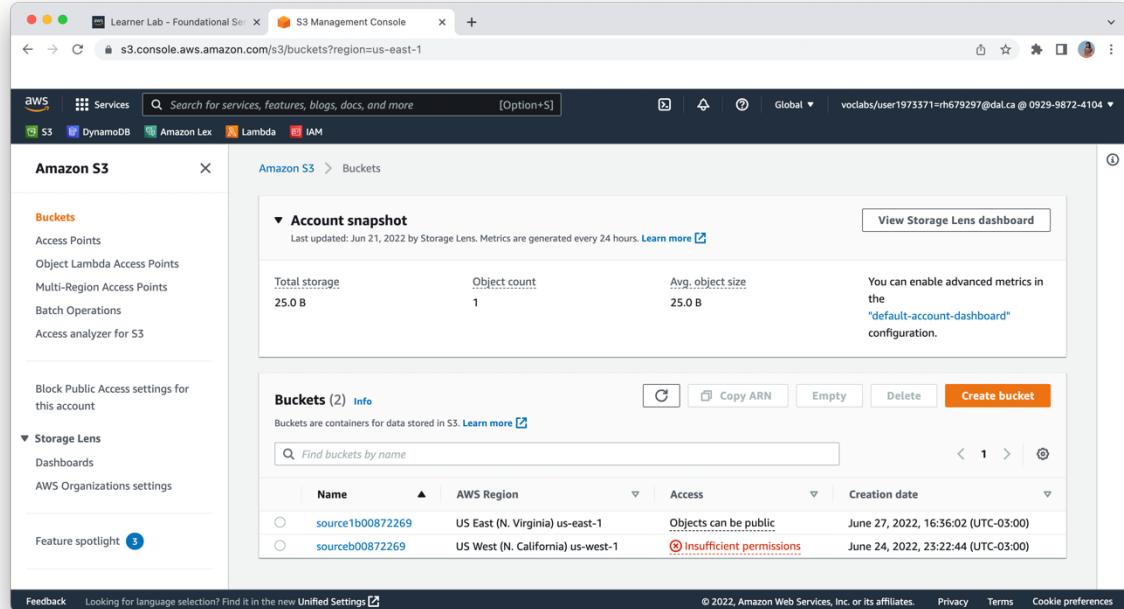


Figure 4. Screenshot of successful creation of bucket **source1b00872269** in Amazon S3 console .

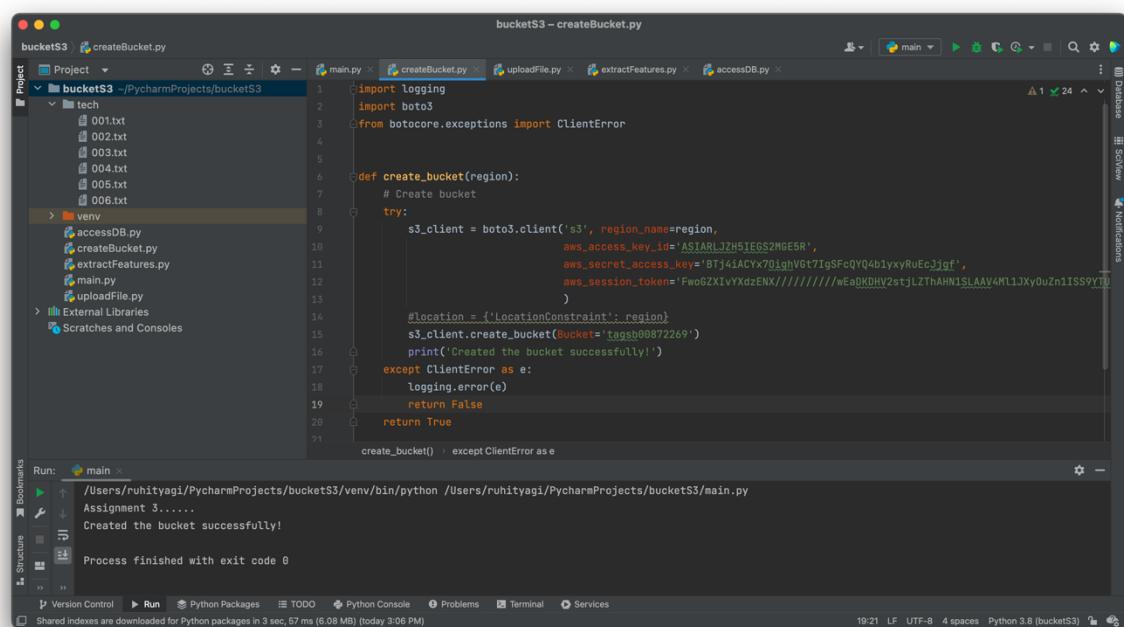


Figure 5. Screenshot of successful creation of bucket **tagsb00872269** .

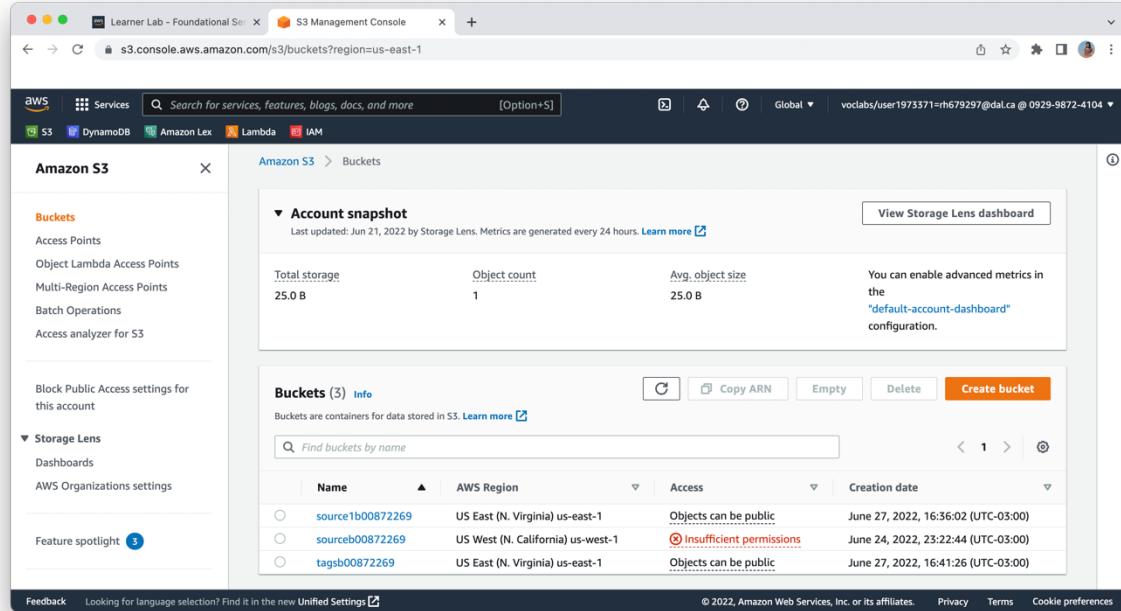


Figure 6. Screenshot of successful creation of bucket **source1b00872269** in Amazon S3 console .

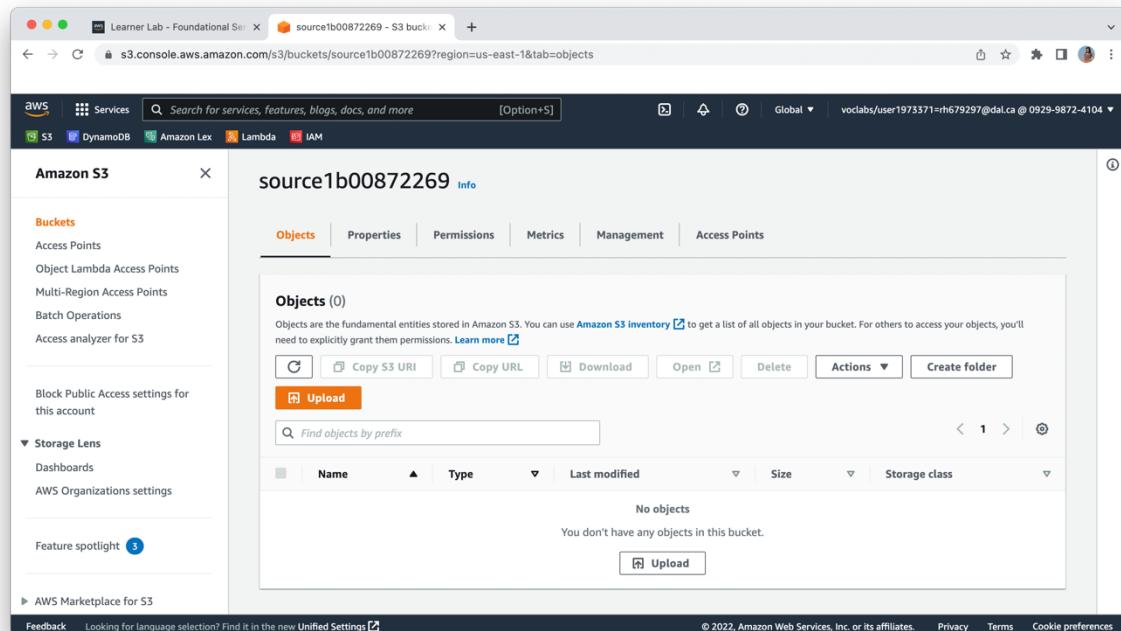
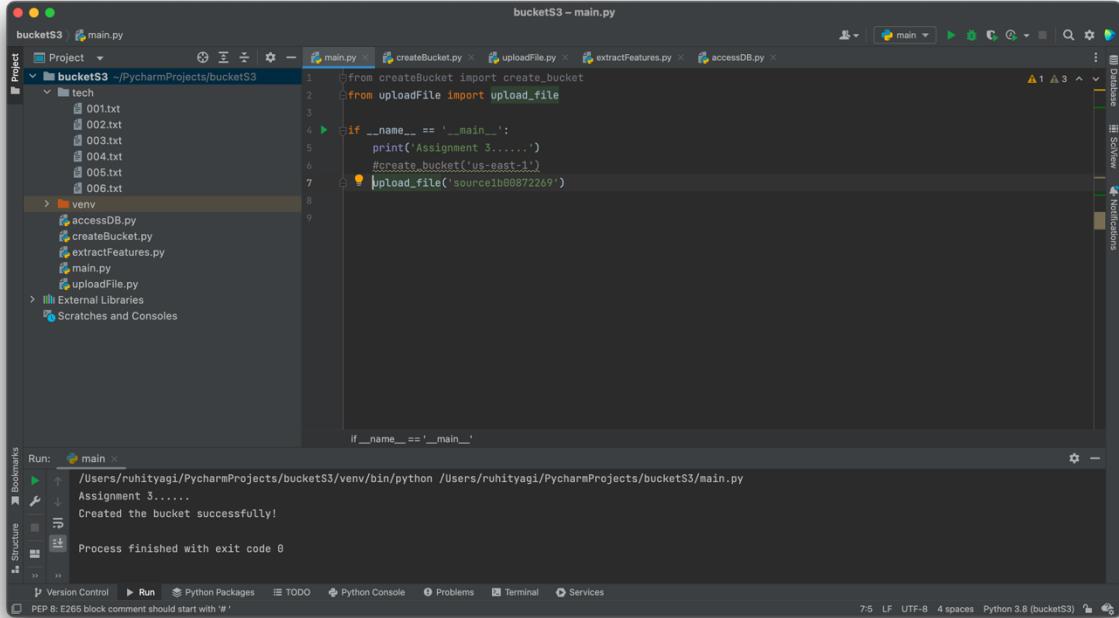


Figure 7. Screenshot of empty bucket **source1b00872269** in Amazon S3 console .



The screenshot shows the PyCharm IDE interface with the project 'bucketS3' open. The 'main.py' file is the active editor, containing the following code:

```

from createBucket import create_bucket
from uploadFile import upload_file

if __name__ == '__main__':
    print('Assignment 3.....')
    #create_bucket('us-east-1')
    upload_file('source1b00872269')

```

The 'Run' tab at the bottom shows the output of running the script:

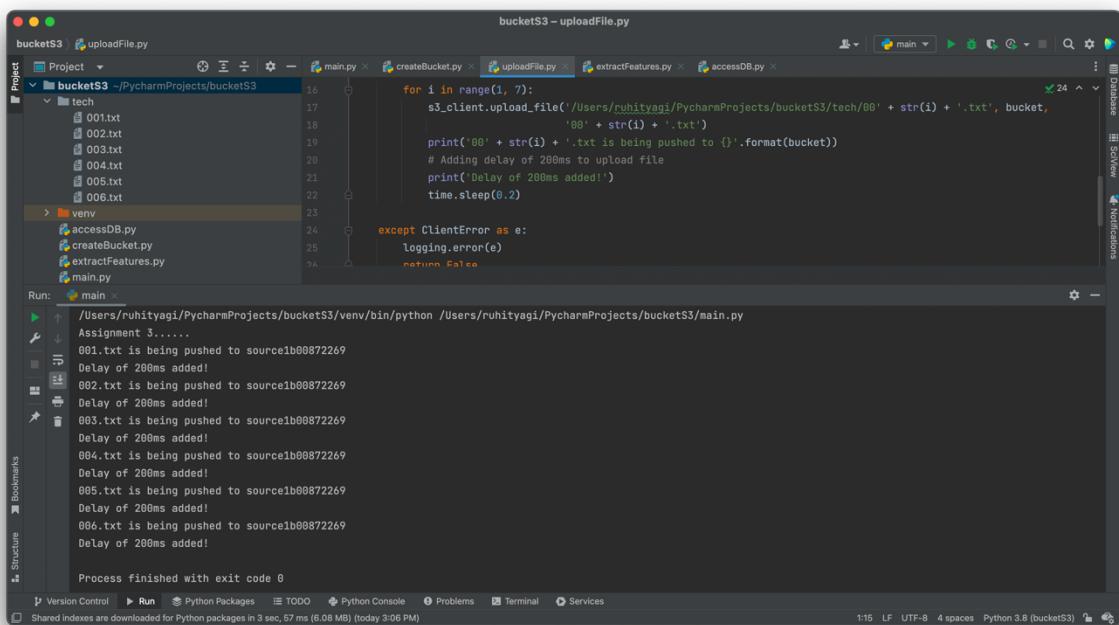
```

Assignment 3.....  

Created the bucket successfully!  

Process finished with exit code 0

```

Figure 8. Screenshot of main.py for uploading files to bucket.


The screenshot shows the PyCharm IDE interface with the project 'bucketS3' open. The 'uploadFile.py' file is the active editor, containing the following code:

```

for i in range(1, 7):
    s3_client.upload_file('/Users/ruhityagi/PycharmProjects/bucketS3/tech/00' + str(i) + '.txt', bucket,
                          '00' + str(i) + '.txt')
    print('00' + str(i) + '.txt is being pushed to {}'.format(bucket))
    # Adding delay of 200ms to upload file
    print('Delay of 200ms added!')
    time.sleep(0.2)

except ClientError as e:
    logging.error(e)
    return False

```

The 'Run' tab at the bottom shows the output of running the script, indicating the upload of each file and a 200ms delay between them:

```

001.txt is being pushed to source1b00872269  

Delay of 200ms added!  

002.txt is being pushed to source1b00872269  

Delay of 200ms added!  

003.txt is being pushed to source1b00872269  

Delay of 200ms added!  

004.txt is being pushed to source1b00872269  

Delay of 200ms added!  

005.txt is being pushed to source1b00872269  

Delay of 200ms added!  

006.txt is being pushed to source1b00872269  

Delay of 200ms added!

Process finished with exit code 0

```

Figure 9. Screenshot of uploading file to bucket **source1b00872269** programmatically.

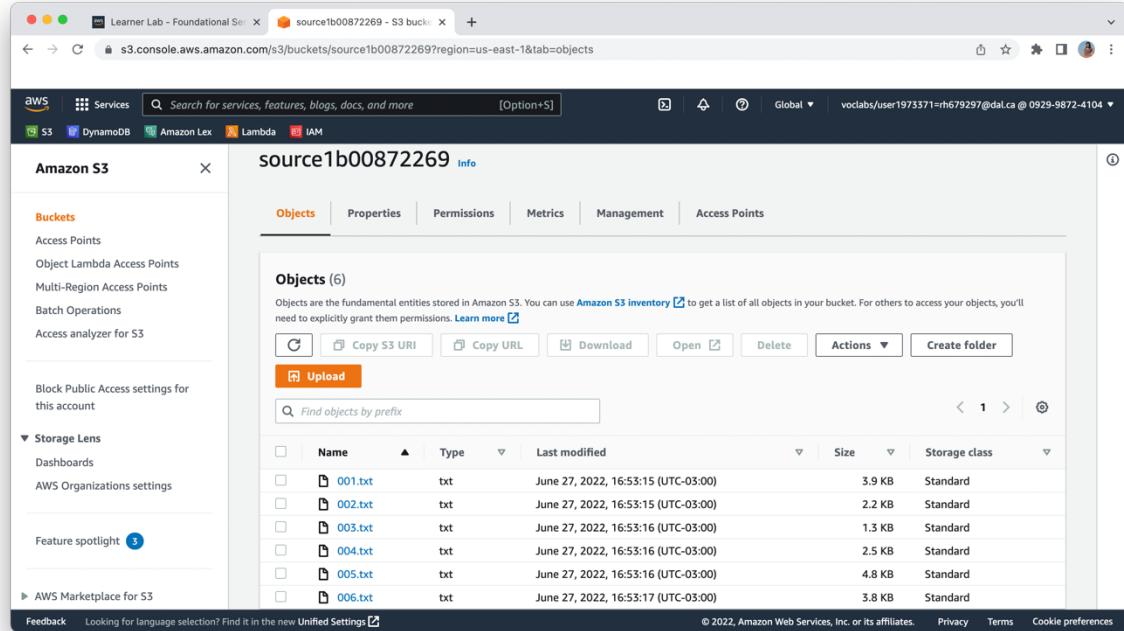


Figure 10. Screenshot of SUCCESSFUL upload of .txt's to bucket **source1b00872269**.

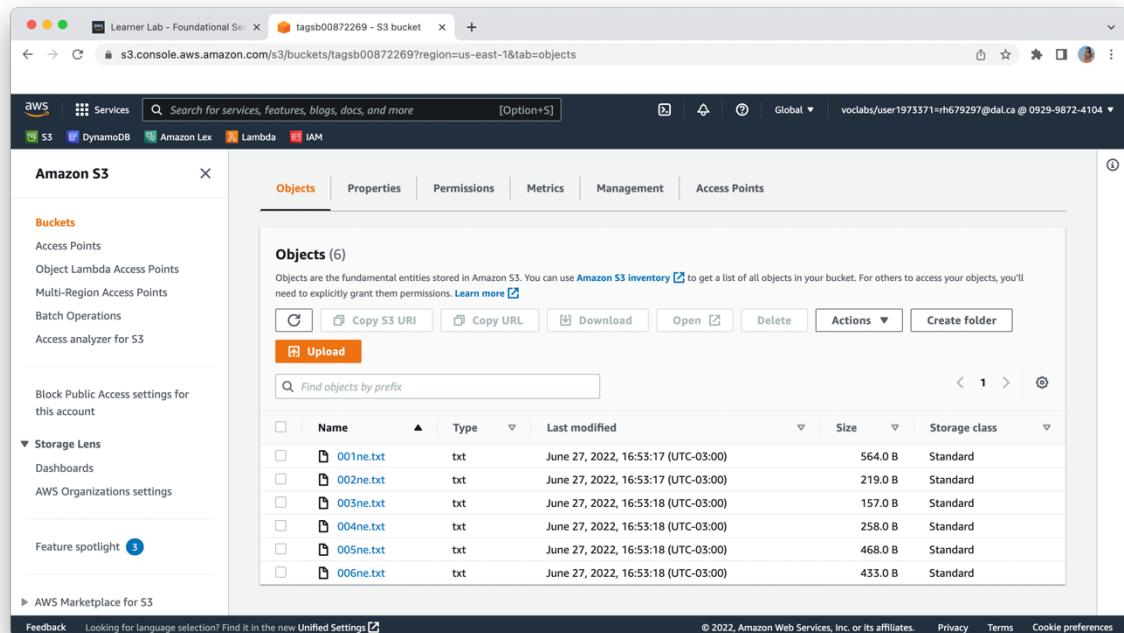


Figure 11. Screenshot of SUCCESSFUL upload of ***ne.txt's to bucket **tagsb00872269**.

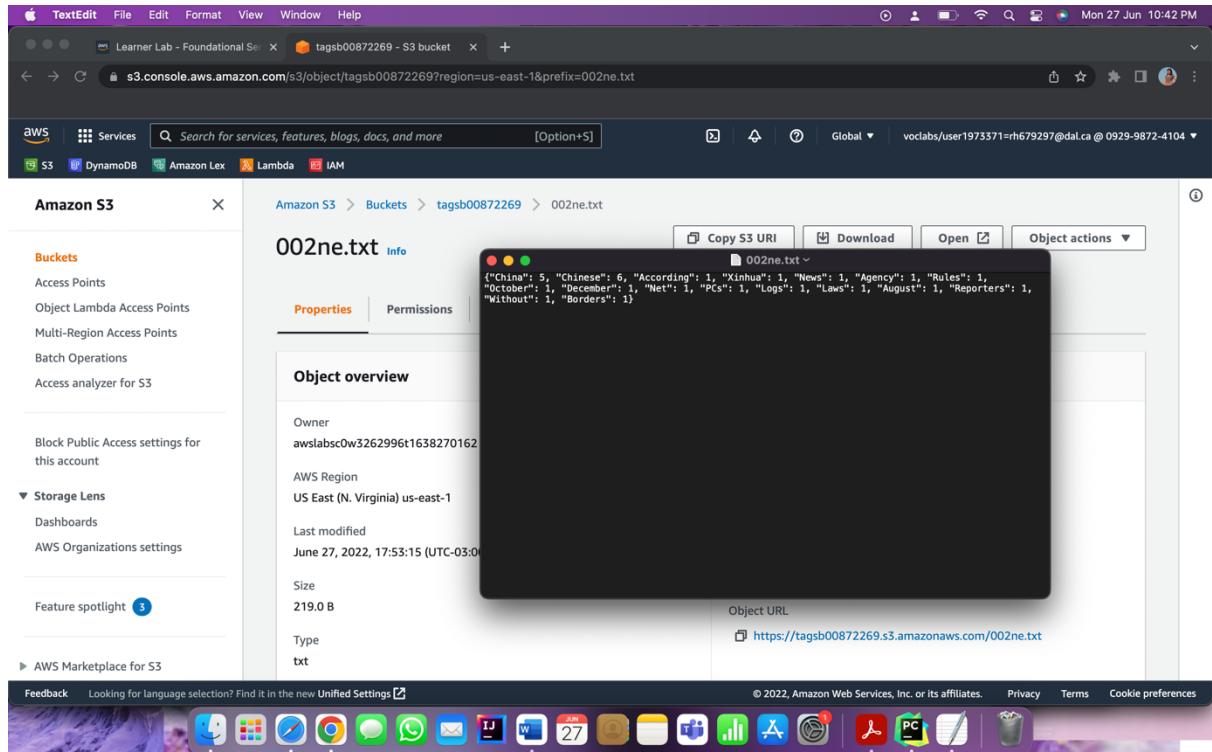


Figure 12. Screenshot of contents of 002ne.txt's.

Step 2: Rendering Lambda functions and updating DynamoDB table.

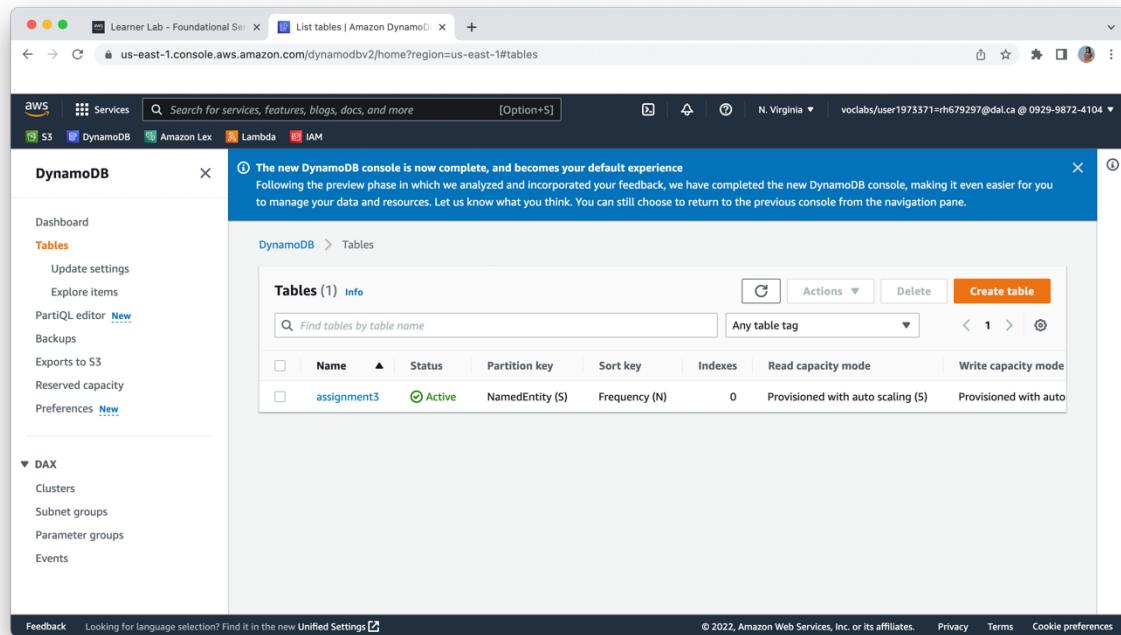


Figure 13. Screenshot of table assignment3 in DynamoDB console.

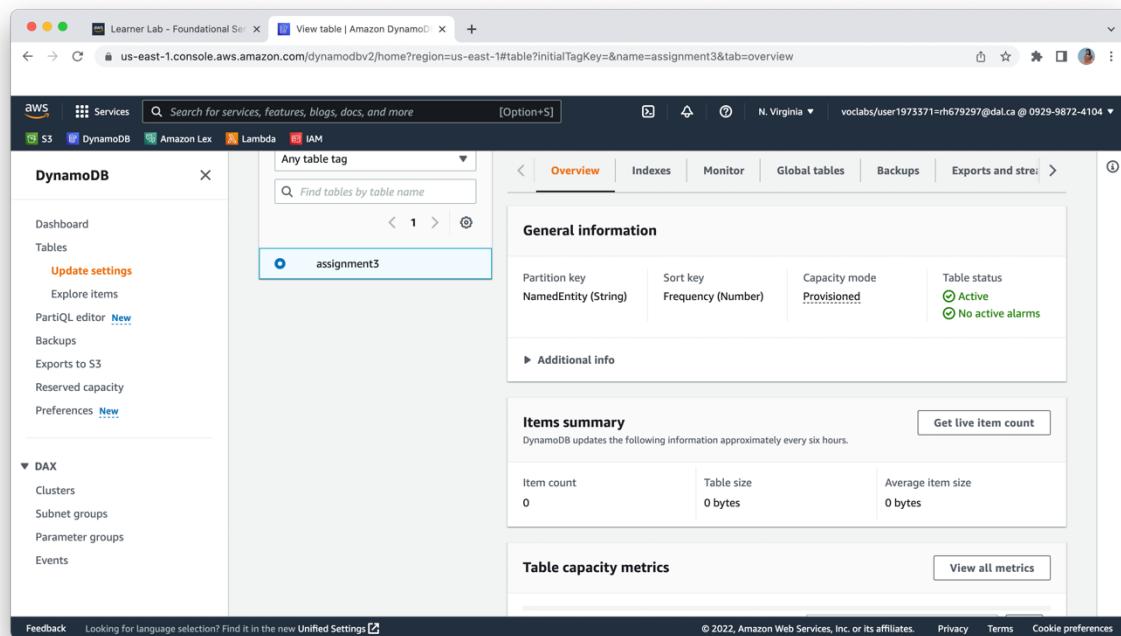


Figure 14. Screenshot of EMPTY table assignment3 in DynamoDB console.

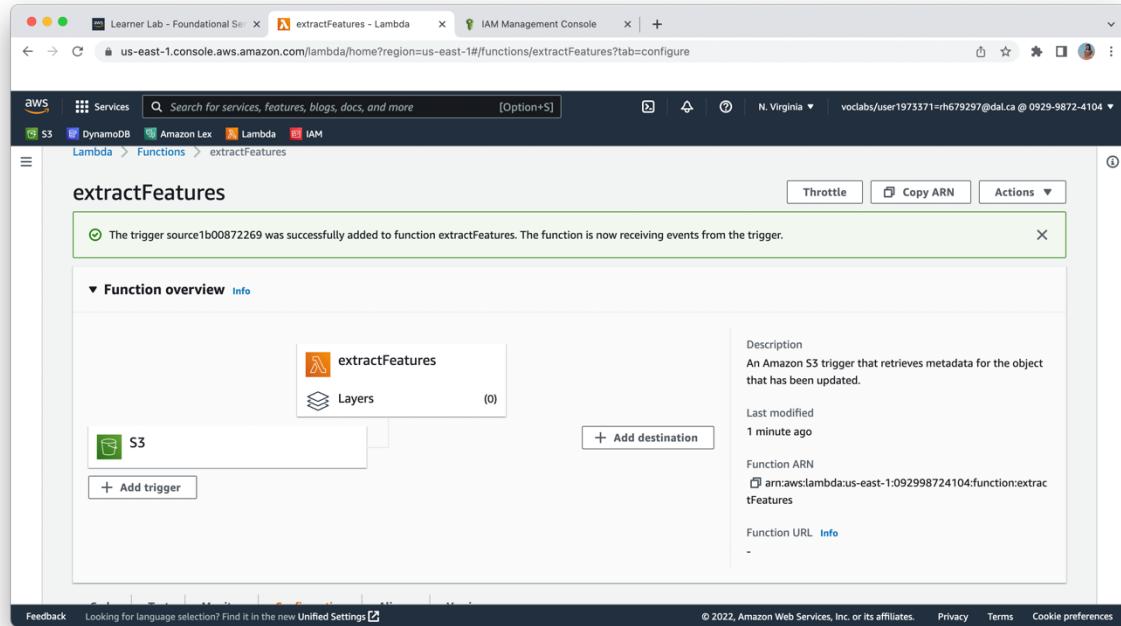


Figure 15. Screenshot of *SUCCESSFUL* creation of ***extractFeatures*** lambda function with trigger set to *S3*.

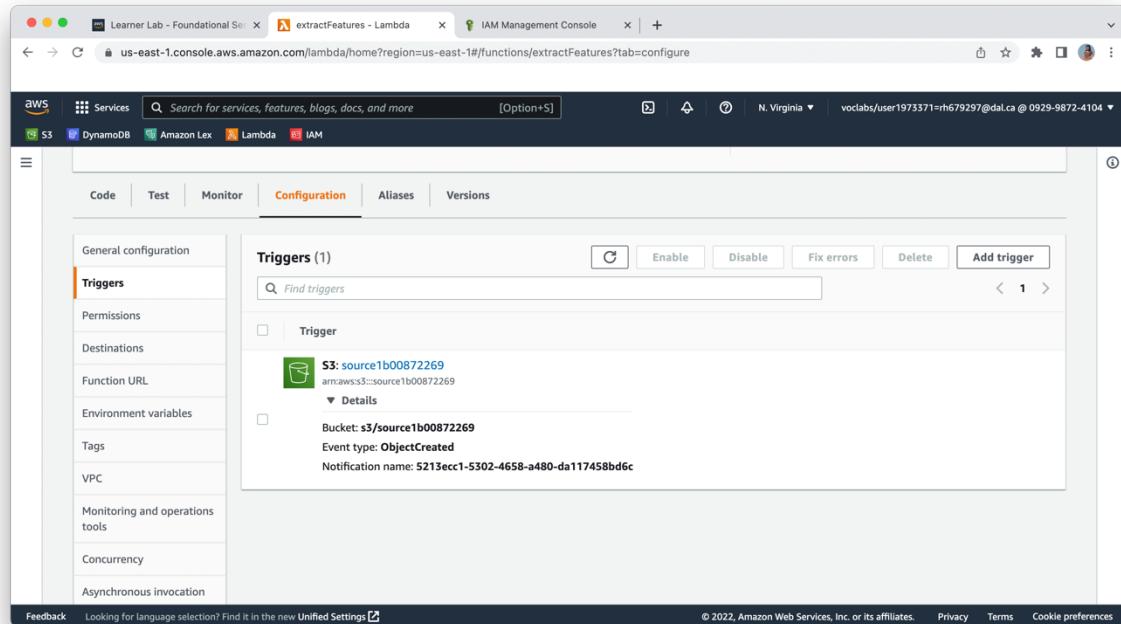


Figure 16. Screenshot of *SUCCESSFUL* creation of ***extractFeatures*** lambda function with *S3* resource set to ***source1b00872269***.

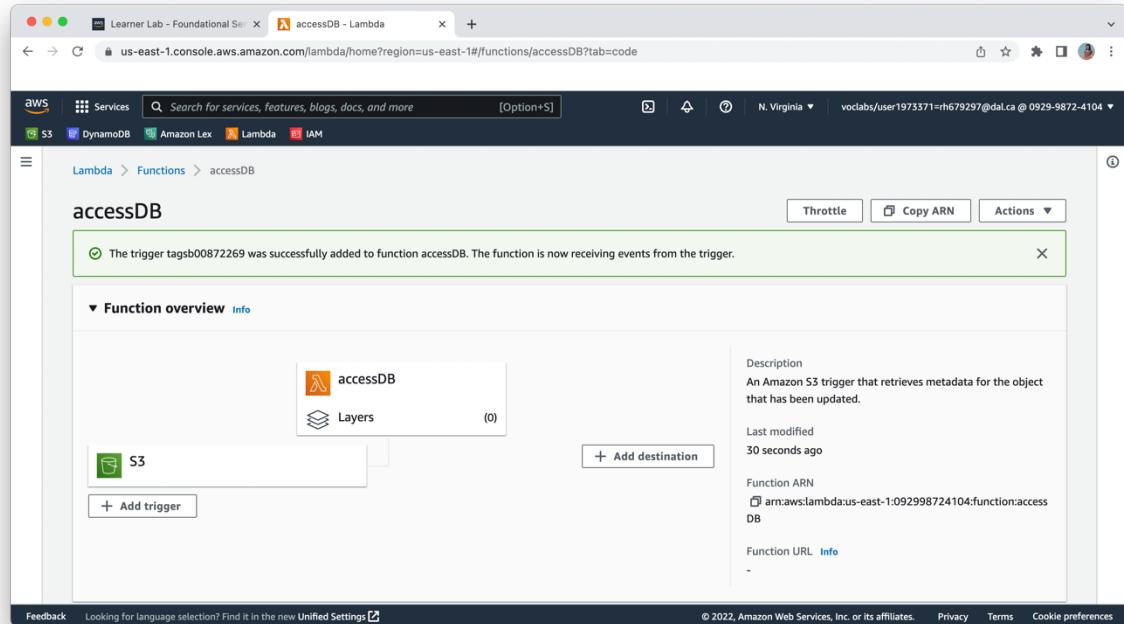


Figure 17. Screenshot of SUCCESSFUL creation of **accessDB** lambda function with trigger set to S3.

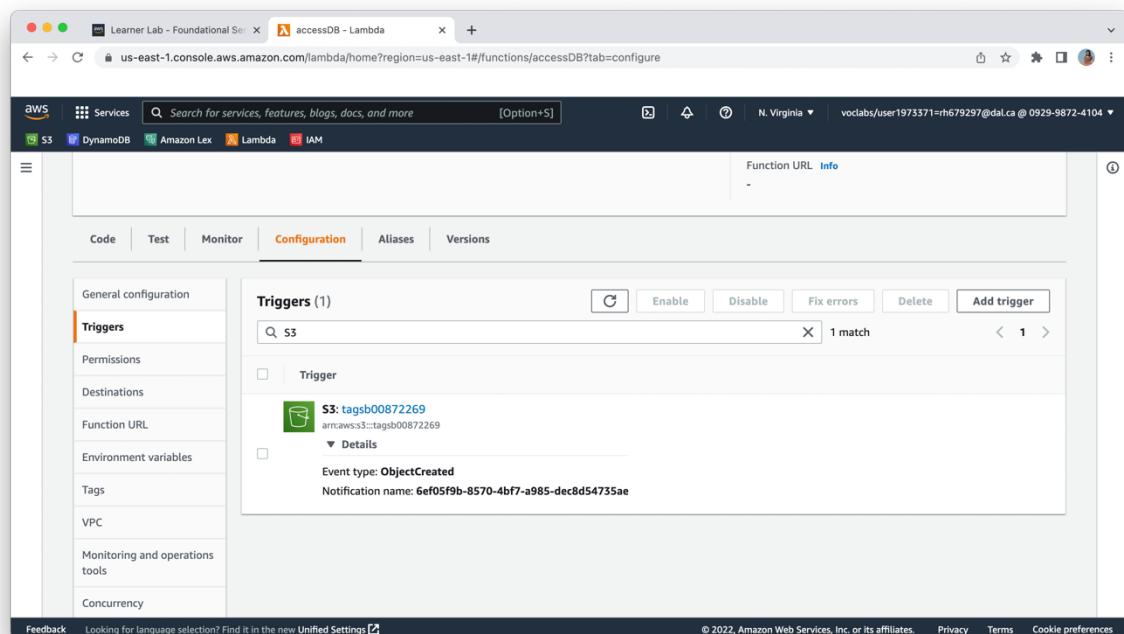


Figure 18. Screenshot of SUCCESSFUL creation of **accessDB** lambda function with S3 resource set to **tagsb00872269**.

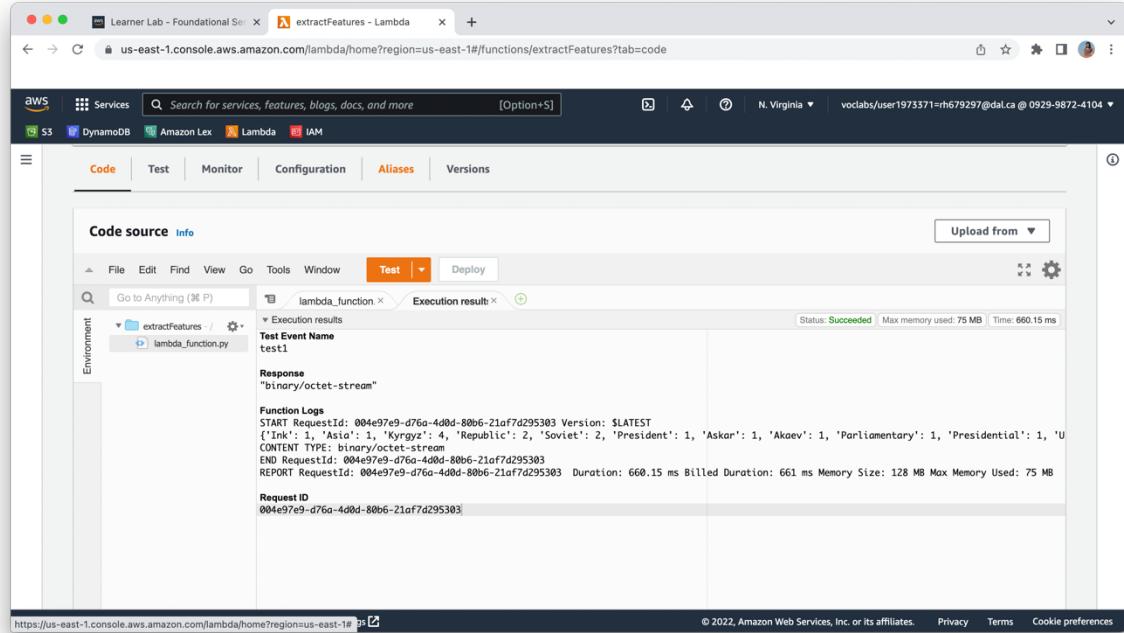


Figure 19. Screenshot of SUCCESSFUL test run of **extractFeatures** lambda function.

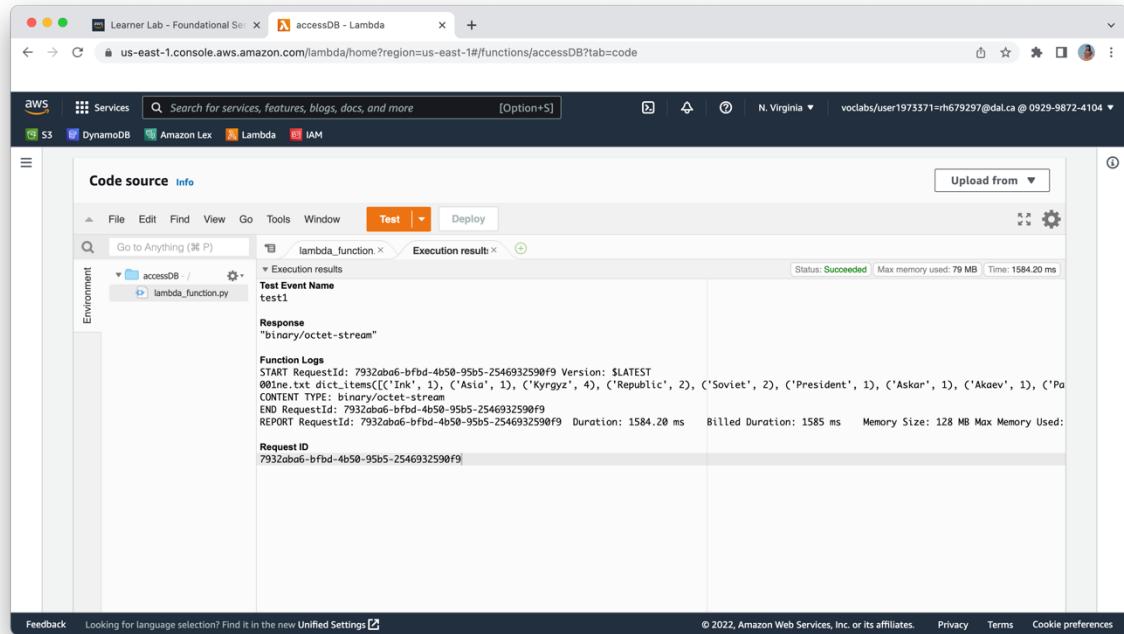


Figure 20. Screenshot of SUCCESSFUL test run of **accessDB** lambda function.

The screenshot shows the AWS DynamoDB console interface. On the left, the navigation sidebar is visible with options like Dashboard, Tables, Update settings, Explore items, PartQL editor, Backups, Exports to S3, Reserved capacity, and Preferences. The 'Tables' section is currently selected. In the main content area, there is a search bar at the top labeled 'Search for services, features, blogs, docs, and more [Option+S]'. Below it, a dropdown menu says 'Any table tag' and a search input field says 'Find tables by table name'. A table titled 'assignment3' is listed under the search results. The table has three columns: 'NamedEntity', 'Frequency', and 'TimeStamp'. The data returned is as follows:

NamedEntity	Frequency	TimeStamp
Georgia	1	2022-06-27 20:53:16.726686
Often	1	2022-06-27 20:53:17.400749
System	1	2022-06-27 20:53:18.075938
UV	2	2022-06-27 20:53:16.566678
US	1	2022-06-27 20:53:17.637799
According	1	2022-06-27 20:53:17.388443
Wi-fi	1	2022-06-27 20:53:17.165038
Logs	1	2022-06-27 20:53:17.668515

Figure 21. Screenshot of SUCCESSFUL update to DynamoDB table-1 .

This screenshot is identical to Figure 21, showing the AWS DynamoDB console with the same navigation sidebar and search interface. The table 'assignment3' is displayed with the same data as Figure 21. The data returned is as follows:

NamedEntity	Frequency	TimeStamp
Georgia	1	2022-06-27 20:53:16.726686
Often	1	2022-06-27 20:53:17.400749
System	1	2022-06-27 20:53:18.075938
UV	2	2022-06-27 20:53:16.566678
US	1	2022-06-27 20:53:17.637799
According	1	2022-06-27 20:53:17.388443
Wi-fi	1	2022-06-27 20:53:17.165038
Logs	1	2022-06-27 20:53:17.668515

Figure 22. Screenshot of SUCCESSFUL update to DynamoDB table-2.

The screenshot shows the AWS DynamoDB console interface. On the left, the navigation pane is open with the 'DynamoDB' section selected. Under 'Tables', there is one entry: 'assignment3'. The main panel displays the contents of the 'assignment3' table. A heading 'Scan/Query items' is followed by a table titled 'Items returned (149)'. The table has columns: 'NamedEntity', 'Frequency', and 'TimeStamp'. The data rows are:

NamedEntity	Frequency	TimeStamp
Mikosz	1	2022-06-27 20:53:17.246671
Centre	1	2022-06-27 20:53:17.671006
Linux-based	1	2022-06-27 20:53:17.577792
Quake	1	2022-06-27 20:53:18.073811
Nicholas	1	2022-06-27 20:53:17.297839
Expressionist	1	2022-06-27 20:53:18.333770
River	1	2022-06-27 20:53:18.013782
...

Figure 23. Screenshot of SUCCESSFUL update to DynamoDB table-3 .

Step 3: extractFeatures.py lambda function script [2][3][4]:

```

import json
import urllib.parse
import boto3
import re

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    # print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key =
urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        var1 = response["Body"].read().decode('utf-8')
        dictionary = {}
        stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours',
'ourselves', 'you', "you're", "you've", "you'll",
            "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's",
            'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their', 'theirs',
            'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', 'am',
            'is', 'are', 'was', 'were', 'be', 'been', 'being',
'have', 'has', 'had', 'having', 'do', 'does',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if',
'or', 'because', 'as', 'until', 'while',
            'of', 'at', 'by', 'for', 'with', 'about', 'against',
'between', 'into', 'through', 'during',
            'before', 'after', 'above', 'below', 'to', 'from',
'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all',
            'any', 'both', 'each', 'few', 'more', 'most', 'other',
'some', 'such', 'no', 'nor', 'not', 'only',
            'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't",
            'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't",
            'couldn', "couldn't", 'didn', "didn't", 'doesn',
"doesn't", 'hadn', "hadn't", 'hasn', "hasn't",
            'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn', "mustn't", 'needn',
            'needn't', 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won',
            'won't', 'wouldn', "wouldn't"]

        for word in var1.split():
            if word.lower() not in stopwords and word[0].isupper():
                w = re.sub("[\)\.,]", "", word)

                if w in dictionary:

```

```
        dictionary[w] += 1
    else:
        dictionary[w] = 1
print(dictionary)

actual_file_name = key.split(".") [0]
file_name = f"/tmp/{actual_file_name}ne.txt"
file = open(file_name, 'w+')
file.write(json.dumps(dictionary))
file.close()

s3.upload_file(file_name, 'tagsb00872269',
f'{actual_file_name}ne.txt')
print("CONTENT TYPE: " + response['ContentType'])
return response['ContentType']
except Exception as e:
    print(e)
    print(
        'Error getting object {} from bucket {}. Make sure they exist
and your bucket is in the same region as this function.'.format(
            key, bucket))
    raise e
```

Step 4: accessDB.py lambda function script [2][3][4][5]:

```

import json
import urllib.parse
import boto3
from datetime import datetime

print('Loading function')

s3 = boto3.client('s3')
dynamodb = boto3.client('dynamodb')

def lambda_handler(event, context):
    # print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key =
    urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        var1 = response["Body"].read().decode('utf-8')
        json_data = json.loads(var1)
        key1 = key.split(":") [0]
        print(key1, json_data.items())
        for k, v in json_data.items():
            dynamodb.put_item(TableName='assignment3', Item={ 'NamedEntity':
{'$': k}, 'Frequency': {'$': str(v)}},
                               'TimeStamp':
{'$': str(datetime.now())})
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print(
            'Error getting object {} from bucket {}. Make sure they exist
and your bucket is in the same region as this function.'.format(
                key, bucket))
        raise e

```

REFERENCES:

- [1] "Amazon S3 buckets — Boto3 Docs 1.24.17 documentation", *Boto3.amazonaws.com*, 2022. [Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-example-creating-buckets.html>. [Accessed: 24- Jun- 2022]
- [2] "S3 — Boto3 Docs 1.24.16 documentation", *Boto3.amazonaws.com*, 2022. [Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>. [Accessed: 25- Jun- 2022]
- [3] "Tutorial: Using an Amazon S3 trigger to invoke a Lambda function", *docs.aws.amazon.com*, 2022. [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html>. [Accessed: 25-Jun- 2022]
- [4] "NLTK :: Natural Language Toolkit", *Nltk.org*, 2022. [Online]. Available: <https://www.nltk.org/>. [Accessed: 25- Jun- 2022]
- [5] "DynamoDB — Boto3 Docs 1.24.16 documentation", *Boto3.amazonaws.com*, 2022.[Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html>. [Accessed: 25- Jun- 2022]