# SCI 5410
# Serverless Data Processing

# Assignment 3
# Part C
# Use AWS Lambda-SNS-SQS

**Ritva Katrodiya**
**B00930131**

Git Repository Link: [Files · A3 · Ritva Ashvinbhai Katrodiya / CSCI5410-SUMMER-23-B00930131 · GitLab (dal.ca)]

# Table of Contents

Table of figures ..... 3
Creation of SQS ..... 5
Creation of SNS Topic for publishing message from lambda ..... 9
Creation of SNS Topic for sending the Order Details ..... 15
Creation of lambda function to publish messages to SNS ..... 22
Creation of lambda function to get the message from queue and place the order details using SNS ..... 26
Lambda functions Scripts ..... 35
References ..... 39

# Table of figures

Figure 1 AWS SQS service dashboard [6] ..... 5
Figure 2 Creation of SQS named **CSCI5410_PArtC_SNS1** [6] ..... 6
Figure 3 Creation of SQS named **CSCI5410_PArtC_SNS1** [6] ..... 6
Figure 4 Creation of SQS named **CSCI5410_PArtC_SNS1** [6] ..... 7
Figure 5 Creation of SQS named **CSCI5410_PArtC_SNS1** [6] ..... 7
Figure 6 SQS named **CSCI5410_PArtC_SNS1** successfully created [6] ..... 8
Figure 7 Created SQS named **CSCI5410_PArtC_SNS1** is listed on Queues [6] ..... 8
Figure 8 Creation of SNS topic named **CSCI5410_PartC_Email** [5] ..... 9
Figure 9 Creation of SNS topic named **CSCI5410_PartC_Email** [5] ..... 10
Figure 10 SNS topic named **CSCI5410_PartC_Email** created successfully [5] ..... 10
Figure 11 No subscription in SNS topic named **CSCI5410_PartC_Email** [5] ..... 11
Figure 12 Create subscription with SQS named **CSCI5410_PArtC_SNS1** [5] ..... 11
Figure 13 Create subscription with SQS named **CSCI5410_PArtC_SNS1** [5] ..... 12
Figure 14 Subscription created successfully with SQS named **CSCI5410_PArtC_SNS1** [5] ..... 12
Figure 15 Subscribe SQS named **CSCI5410_PArtC_SNS1** to SNS topic **CSCI5410_PartC_Email** [6] 13
Figure 16 Subscribe SQS named **CSCI5410_PArtC_SNS**1 to SNS topic **CSCI5410_PartC_Email** [6] 14
Figure 17 AWS SNS service dashboard [5] ..... 15
Figure 18 SNS with existing Topics [5] ..... 15
Figure 19 Creation of SNS topic named **CSCI5410_Partc_SNS** [5] ..... 16
Figure 20 Creation of SNS topic named **CSCI5410_Partc_SNS** [5] ..... 16
Figure 21 SNS topic named **CSCI5410_Partc_SNS** created successfully [5] ..... 17
Figure 22 Newly created SNS topic named **CSCI5410_Partc_SNS** [5] ..... 17
Figure 23 No subscription in **CSCI5410_Partc_SNS** topic [5] ..... 18
Figure 24 Subscription of email to get order details [5] ..... 18
Figure 25 Subscription to **CSCI5410_Partc_SNS** created successfully [5] ..... 19
Figure 26 Subscription confirmation mail on **rt664810@dal.ca** [8] ..... 20
Figure 27 Subscription is confirmed by **rt664810@dal.ca** [5] ..... 20
Figure 28 Subscription confirmed with email and ARN [5] ..... 21
Figure 29 Creation of lambda function named **CSCI5410_PartC_SendMessages** [3] ..... 22
Figure 30 Creation of lambda function named **CSCI5410_PartC_SendMessages** [3] ..... 23
Figure 31 Successfully created lambda function named **CSCI5410_PartC_SendMessages** [3] ..... 23
Figure 32 Lambda function named CSCI5410_PartC_SendMessages is listed [3] ..... 24
Figure 33 Written lambda function in python to publish messages in SNS [3] ..... 24
Figure 34 Configured the test event named test to **test** the lambda fucntion [3] ..... 25
Figure 35 Creation of lambda fucntion named **CSCI5410_PartC_processOrder** [3] ..... 26
Figure 36 Creation of lambda fucntion named **CSCI5410_PartC_processOrder** [3] ..... 27
Figure 37 Successfully created lambda function named **CSCI5410_PartC_processOrder** [3] ..... 27
Figure 38 Lambda function named **CSCI5410_PartC_processOrder** is listed [3] ..... 28
Figure 39 Written lambda function in python to get the available messages from SQS and send the order details by SNS [3] ..... 28
Figure 40 Trigger Configuration between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [7] ..... 29

Figure 41 Trigger Configuration between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [7] ..... 30
Figure 42 Trigger Configured successfully between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [3] ..... 30
Figure 43 Trigger Configured successfully between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [3] ..... 31
Figure 44 Lambda named **CSCI5410_PartC_SendMessages** tested to publish messages in SNS [3] ..... 31
Figure 45 **4 Messages** are available in SQS named **CSCI5410_PArtC_SNS1** [6] ..... 32
Figure 46 **First time** lambda triggered, and email is available with order details [8] ..... 33
Figure 47 After 2 minutes, second time lambda triggered, and email is available with order details [8] ... 33
Figure 48 After 2 minutes, third time lambda triggered, and email is available with order details [8] ..... 34
Figure 49 After 2 minutes, second time lambda triggered, and email is available with order details [8] ... 34

# Creation of SQS

In Assignment 3, PartC, I have created one SQS named **CSCI5410_PArtC_SNS1.** When the lambda named **CSCI5410_PartC_SendMessages** will be tested, it will publish message to SNS, and Message will be queued in SQS by SNS.
Figures 2,3,4 and 5 show the steps require to create queue in AWS SQS service. Figure 6 shows that SQS named **CSCI5410_PArtC_SNS1** created successfully and figure 7 defines that SQS **CSCI5410_PArtC_SNS1** is listed in queues.

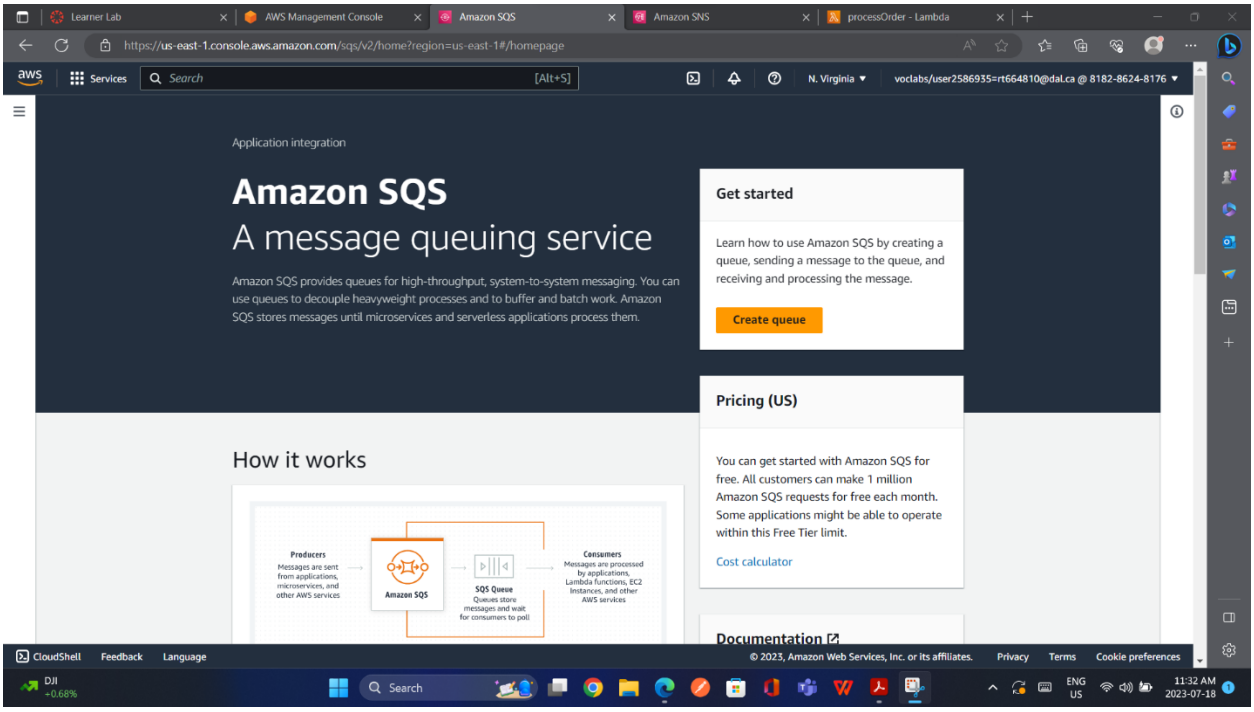

Figure 1 AWS SQS service dashboard [6]

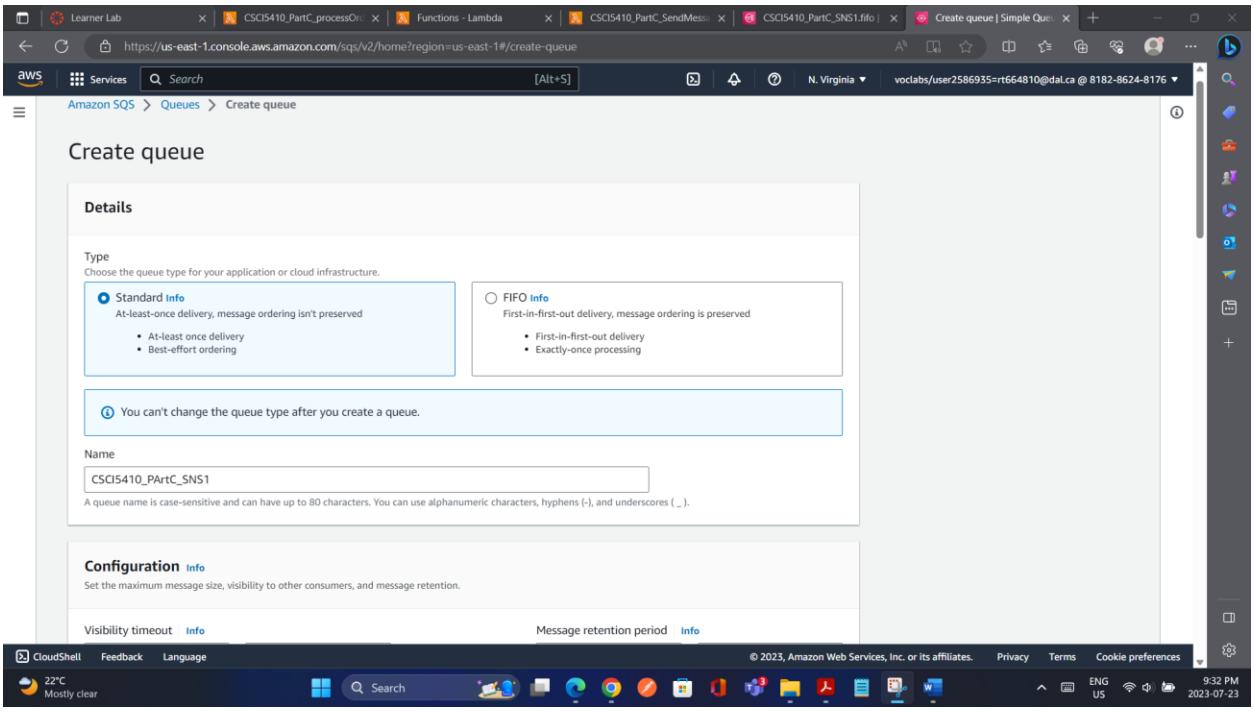


Figure 2 Creation of SQS named **CSCI5410_PArtC_SNS1** [6]

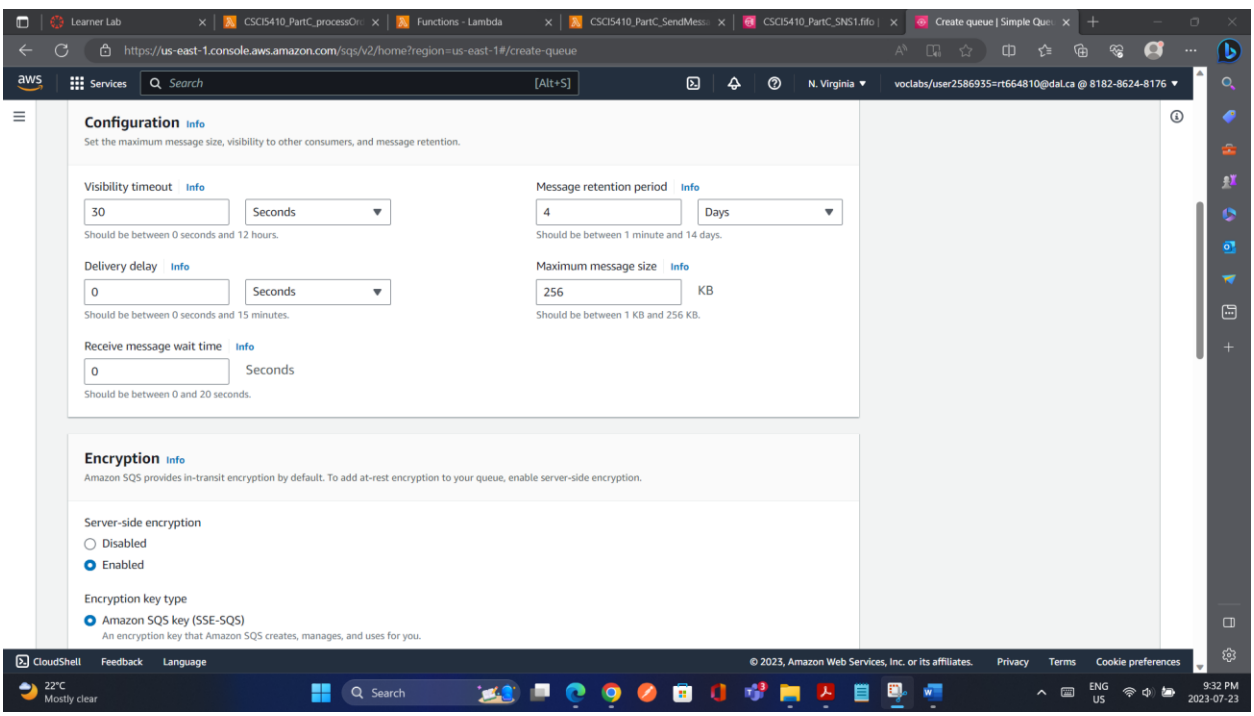


Figure 3 Creation of SQS named **CSCI5410_PArtC_SNS1** [6]

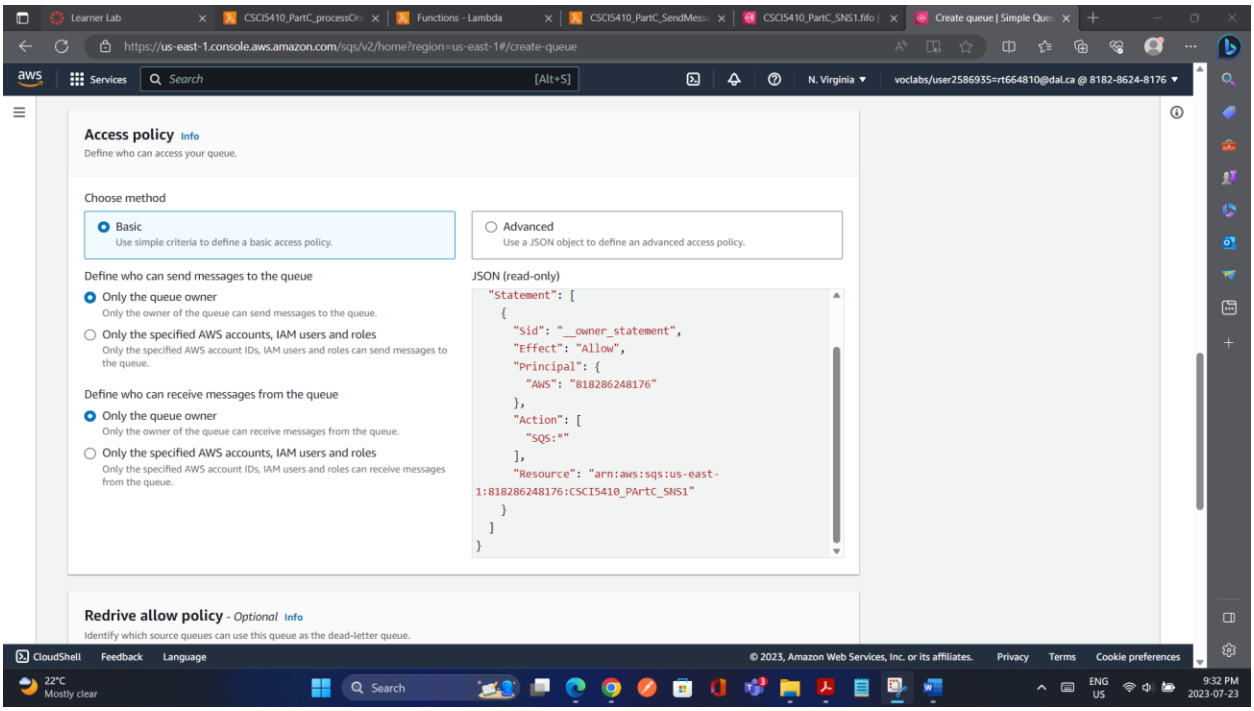


Figure 4 Creation of SQS named **CSCI5410_PArtC_SNS1** [6]

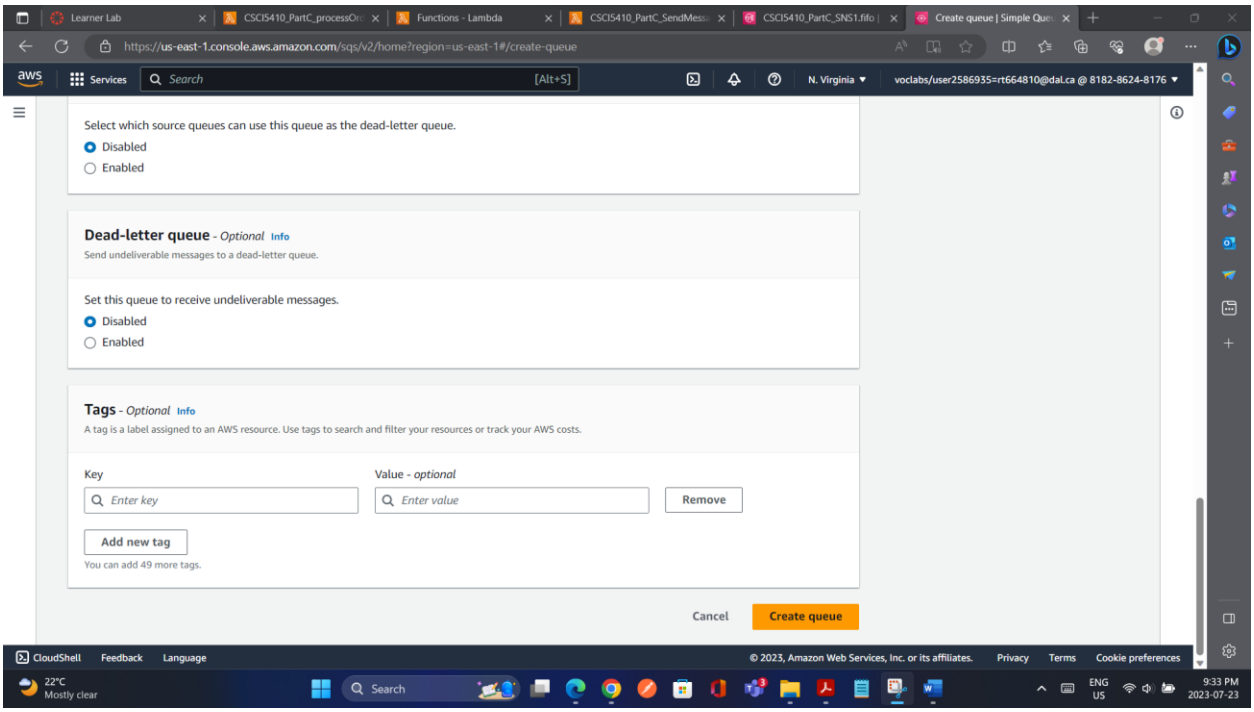


Figure 5 Creation of SQS named **CSCI5410_PArtC_SNS1** [6]

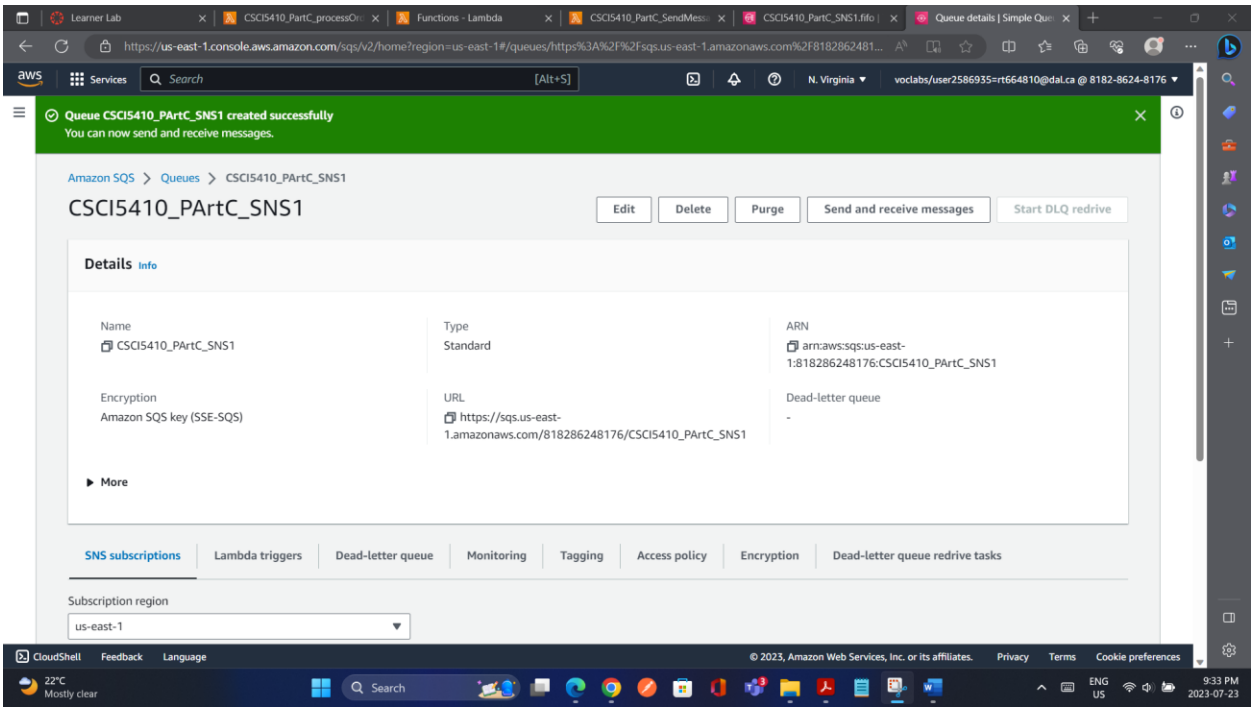


Figure 6 SQS named **CSCI5410_PArtC_SNS1** successfully created [6]

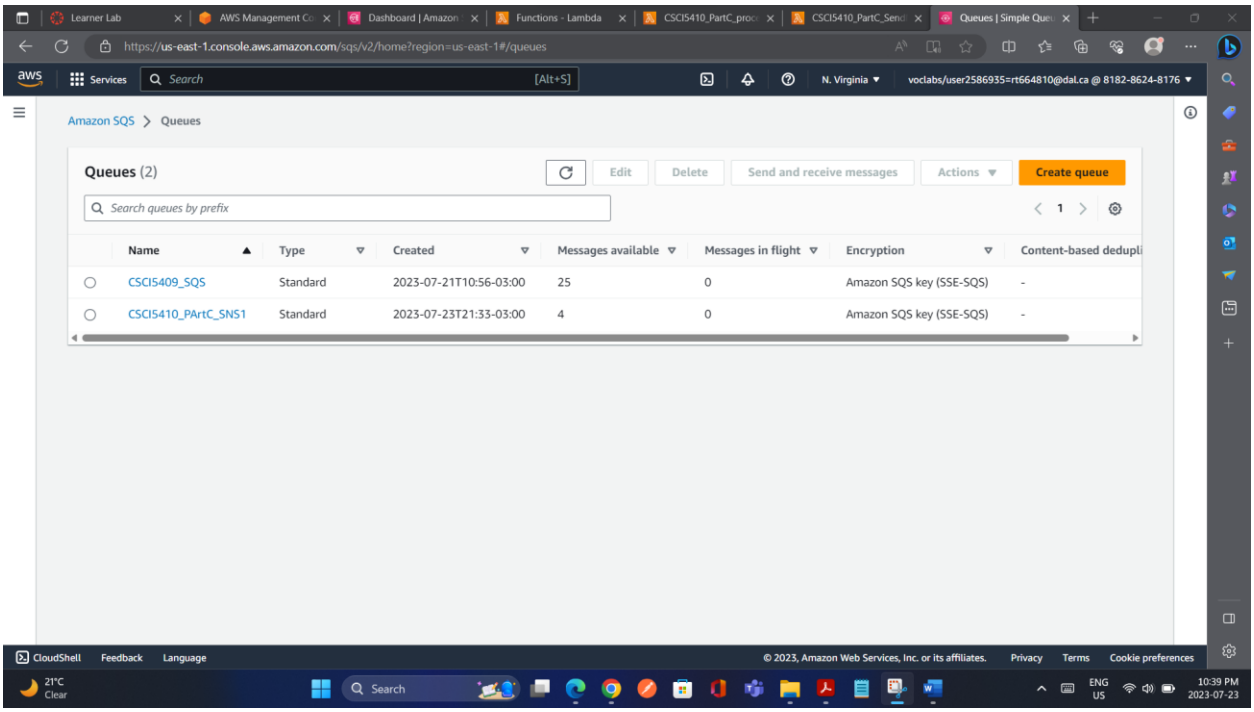


Figure 7 Created SQS named **CSCI5410_PArtC_SNS1** is listed on Queues [6]

## Creation of SNS Topic for publishing message from lambda

Figures 8 and 9 show the steps required to create SNS topic named **CSCI5410_PartC_Email**. Figure 10 shows that SNS topic named **CSCI5410_PartC_Email** created successfully. This SNS topic is created because when the lambda fucntion named **CSCI5410_PartC_SendMessages** will be tested, lambda fucntion will first publish message in this SNS topic and then SNS topic will queue message in SQS **CSCI5410_PArtC_SNS1.** Figure 11 shows that there is not any subscription in SNS topic **CSCI5410_PartC_Email**.

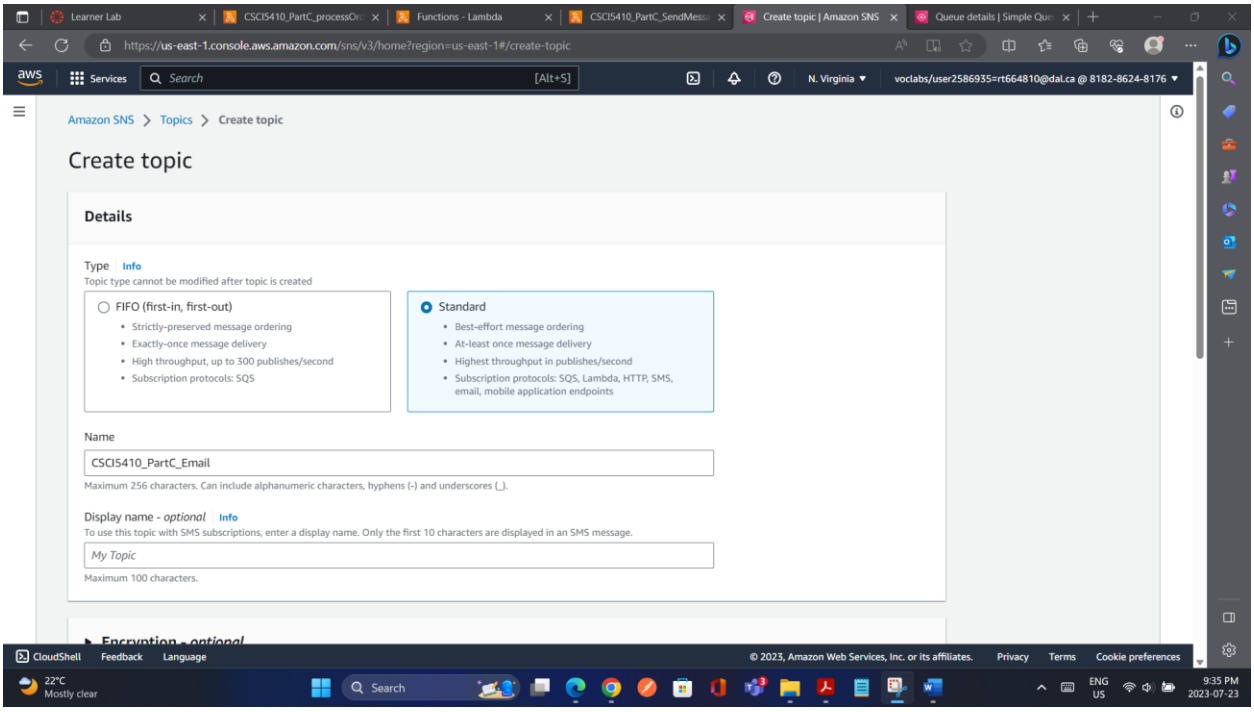


Figure 8 Creation of SNS topic named **CSCI5410_PartC_Email** [5]

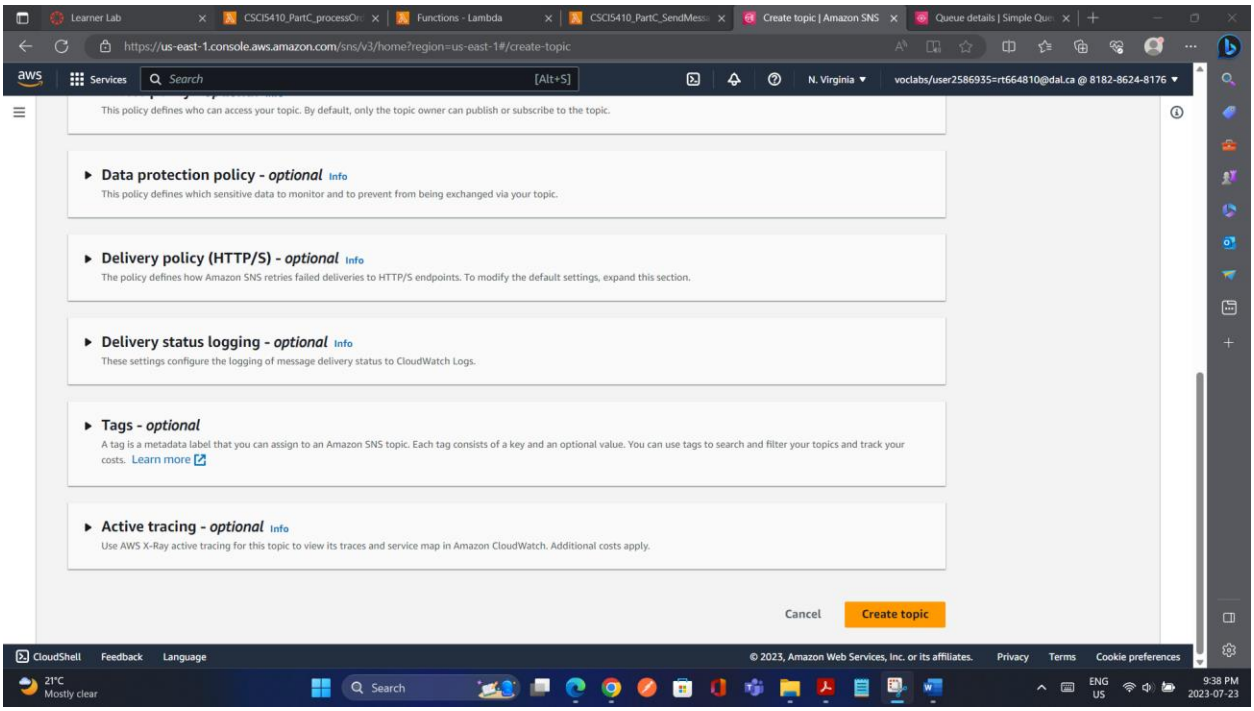


Figure 9 Creation of SNS topic named **CSCI5410_PartC_Email** [5]

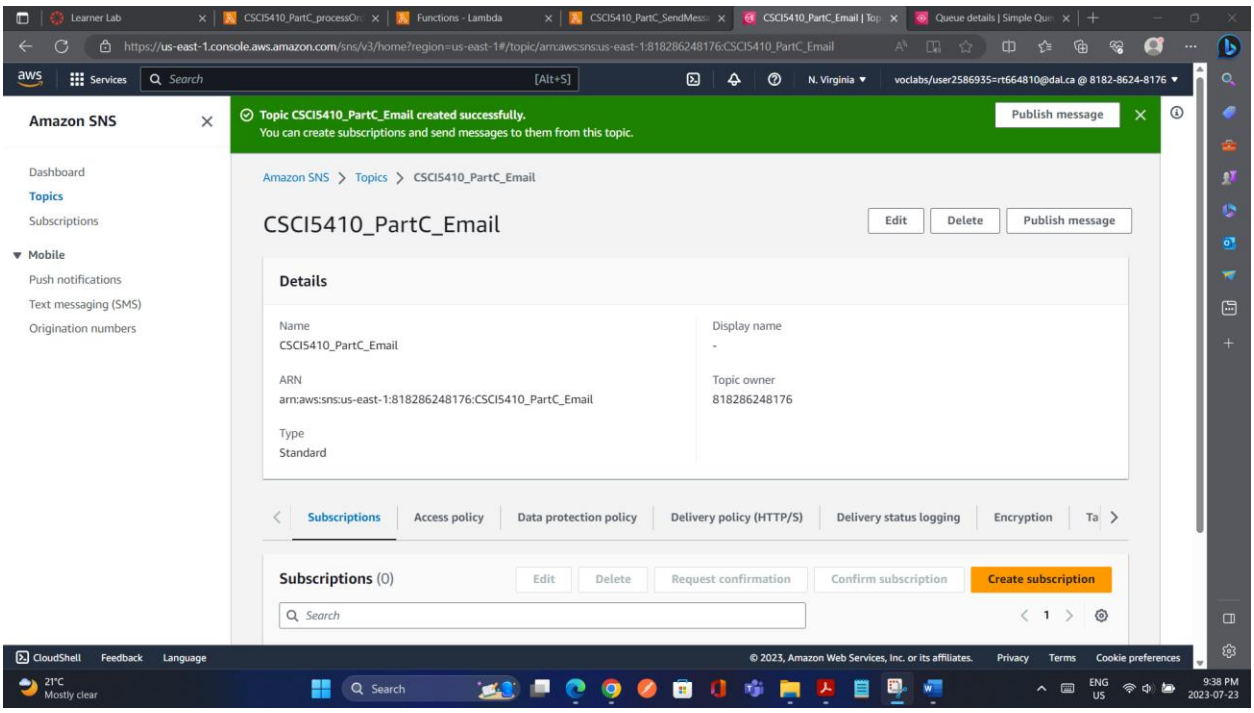


Figure 10 SNS topic named **CSCI5410_PartC_Email** created successfully [5]

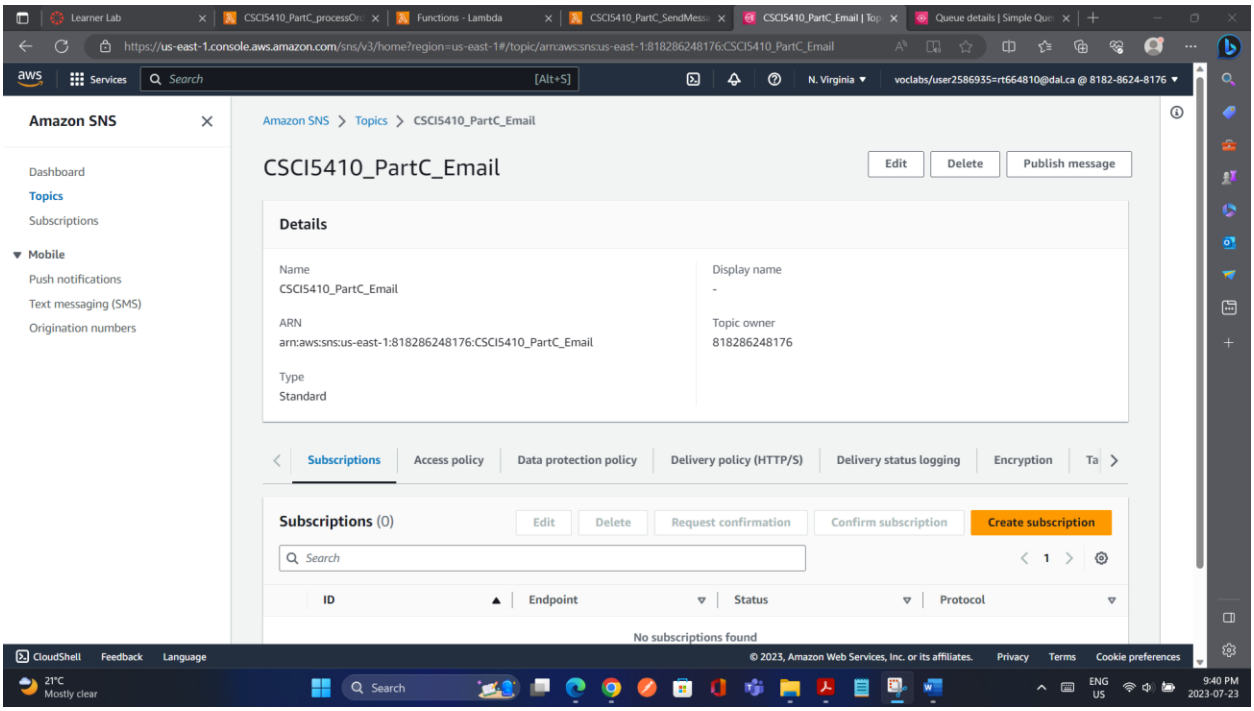


Figure 11 No subscription in SNS topic named **CSCI5410_PartC_Email** [5]

Figures 12 and 13 shows the creation of subscription in SNS topic named **CSCI5410_PartC_Email** with SQS named **CSCI5410_PArtC_SNS1**. Figure 14 shows that subscription created successfully with **CSCI5410_PArtC_SNS1**.

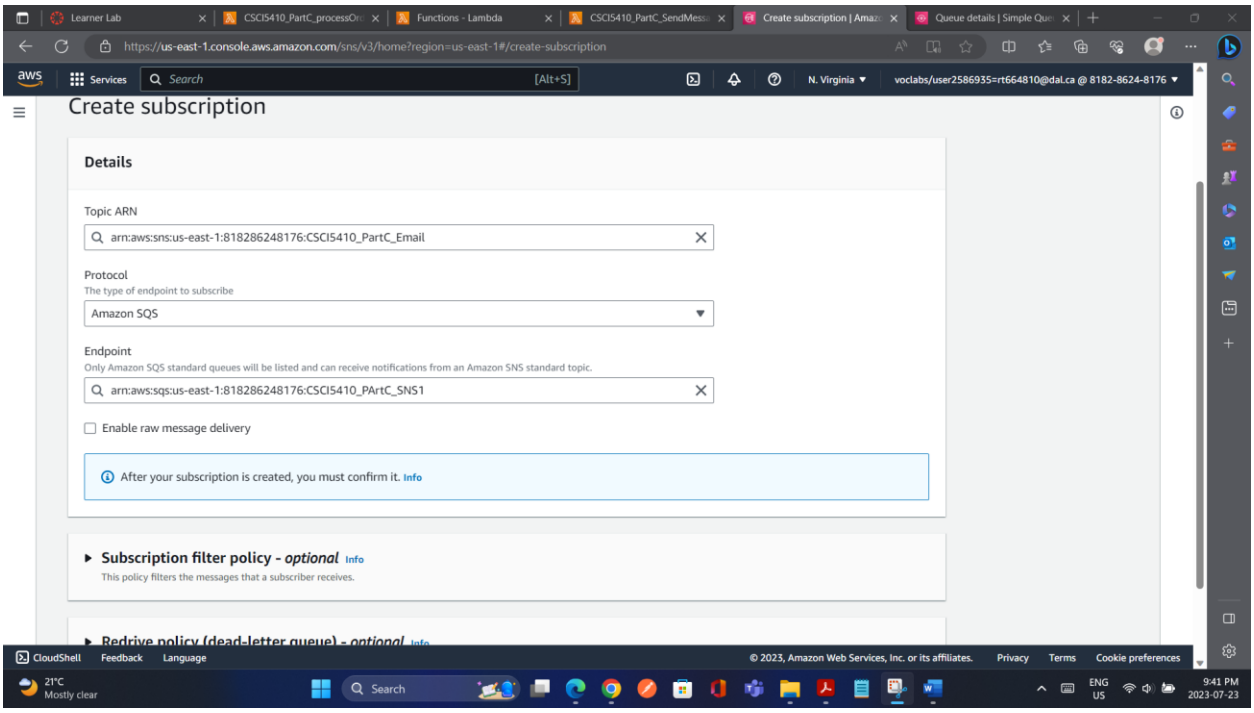


Figure 12 Create subscription with SQS named **CSCI5410_PArtC_SNS1** [5]

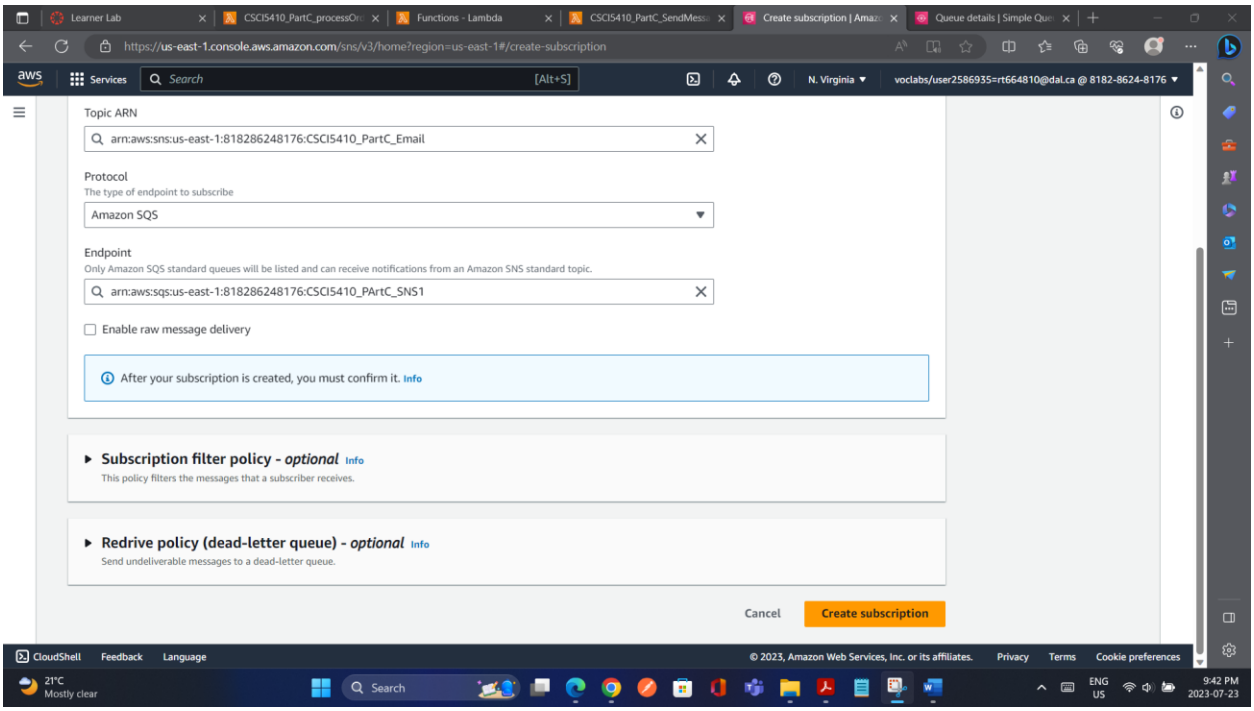


Figure 13 Create subscription with SQS named **CSCI5410_PArtC_SNS1** [5]

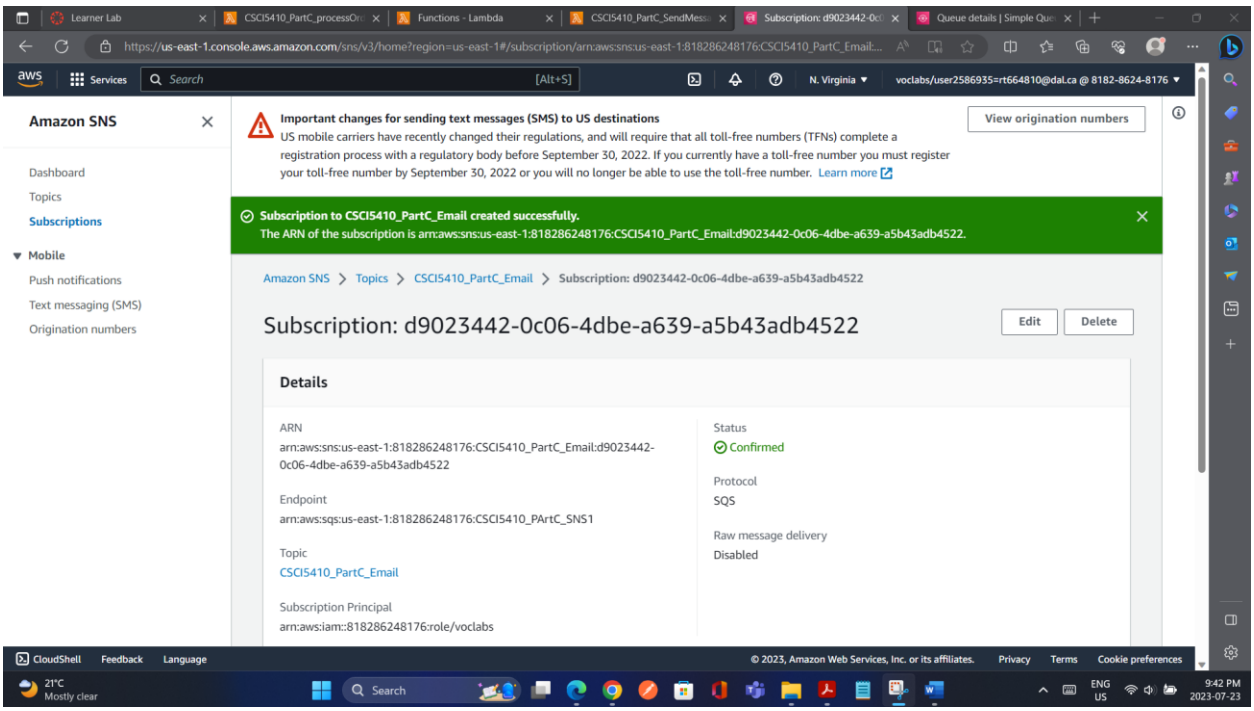


Figure 14 Subscription created successfully with SQS named **CSCI5410_PArtC_SNS1** [5]

Figure 15 shows that SQS named **CSCI5410_PArtC_SNS1** is subscribed to SNS topic named **CSCI5410_PartC_Email**. Figure 16 defines that SNS topic **CSCI5410_PartC_Email** is available in the queue **CSCI5410_PArtC_SNS1.**

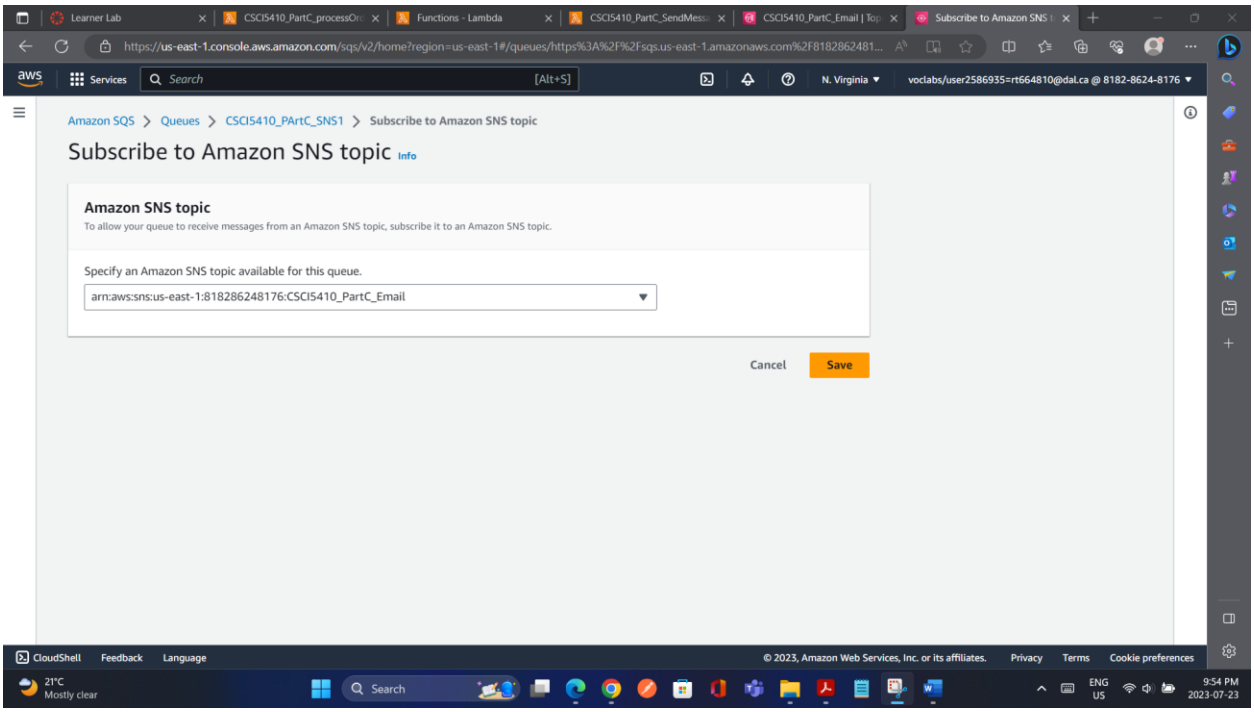


Figure 15 Subscribe SQS named **CSCI5410_PArtC_SNS1** to SNS topic **CSCI5410_PartC_Email** [6]

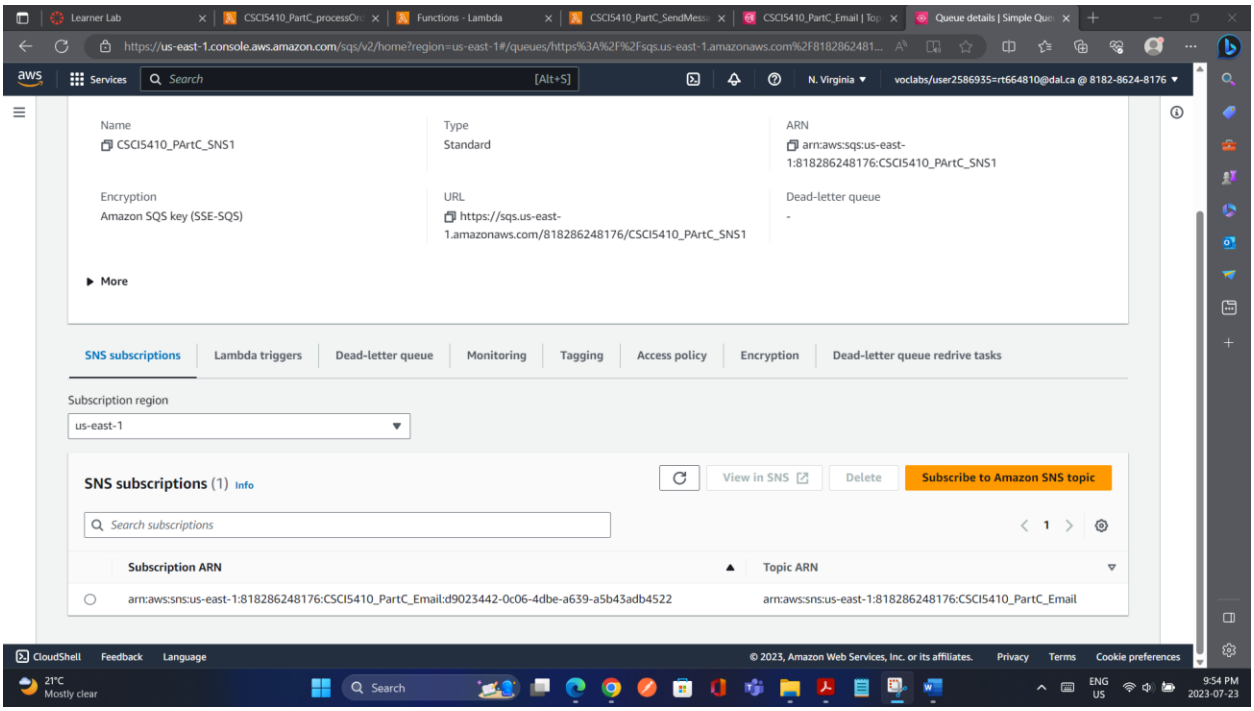


Figure 16 Subscribe SQS named **CSCI5410_PArtC_SNS**1 to SNS topic **CSCI5410_PartC_Email** [6]

## Creation of SNS Topic for sending the Order Details

Figure 17 shows Amazon SNS service dashboard and figure 18 show the existing SNS topics.

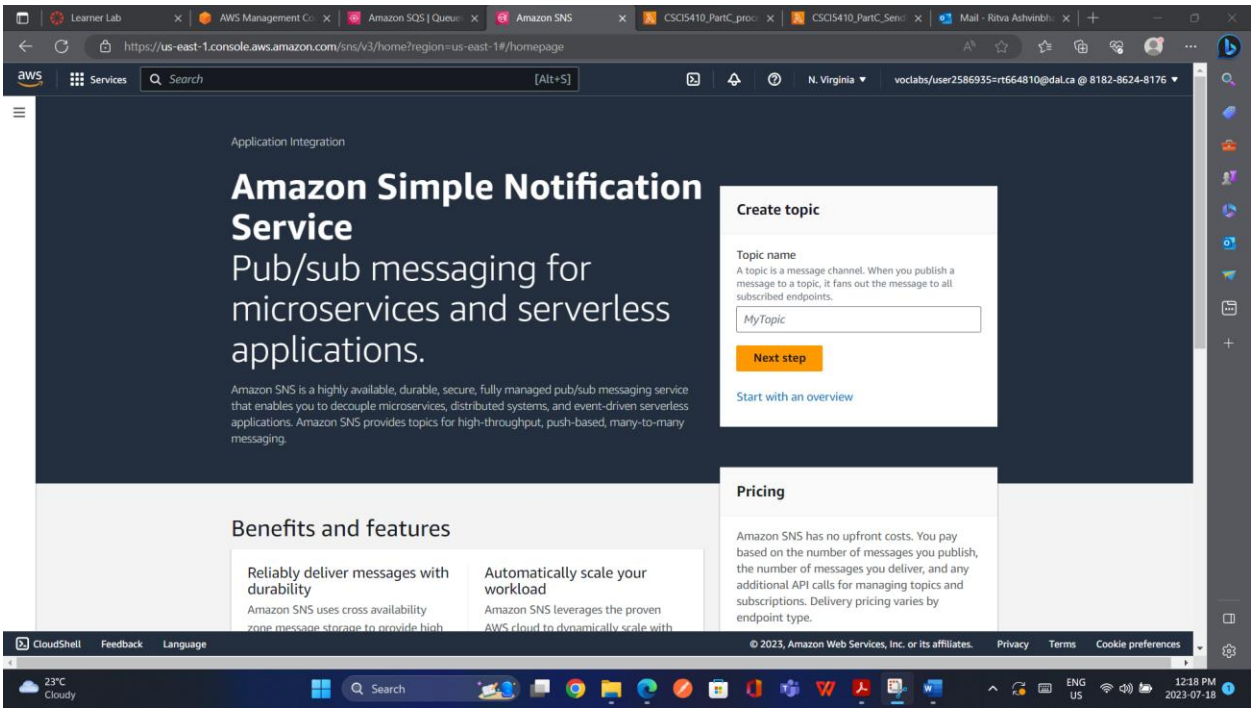


Figure 17 AWS SNS service dashboard [5]

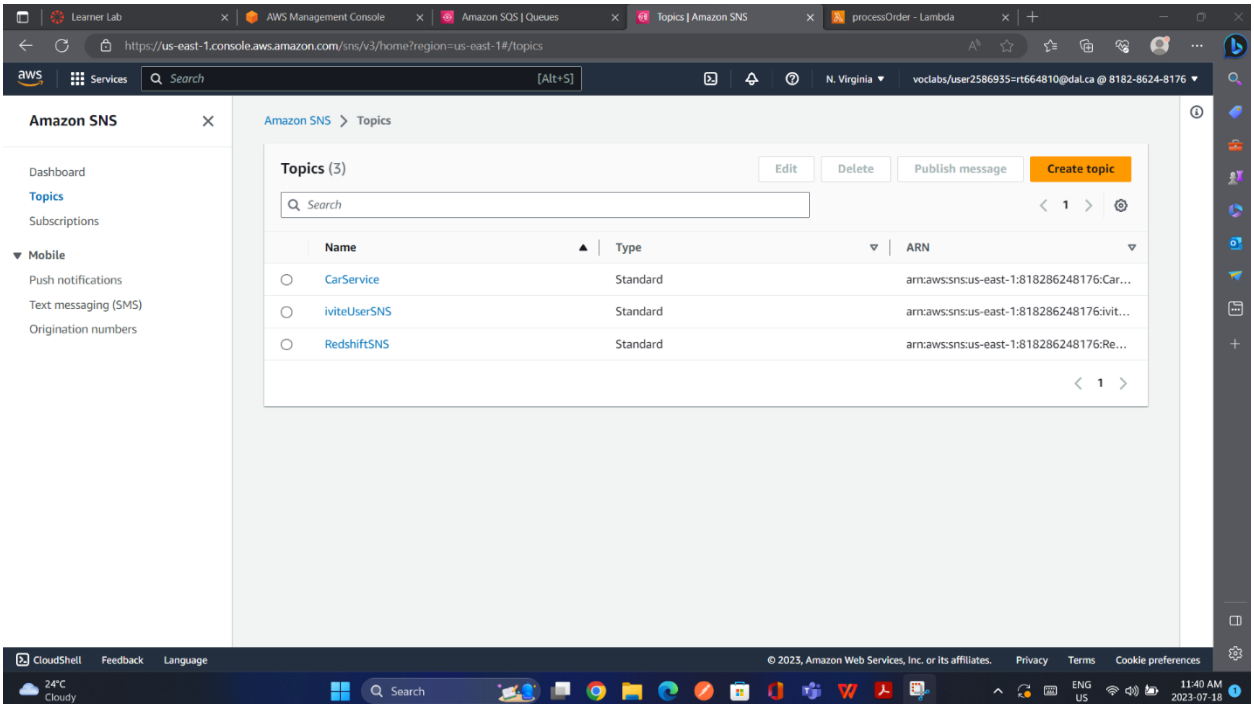


Figure 18 SNS with existing Topics [5]

Figures 19,20 and 21 show the steps require to create SNS topic named **CSCI5410_Partc_SNS**. Figure 21 defines that SNS topic named **CSCI5410_Partc_SNS** created successfully.

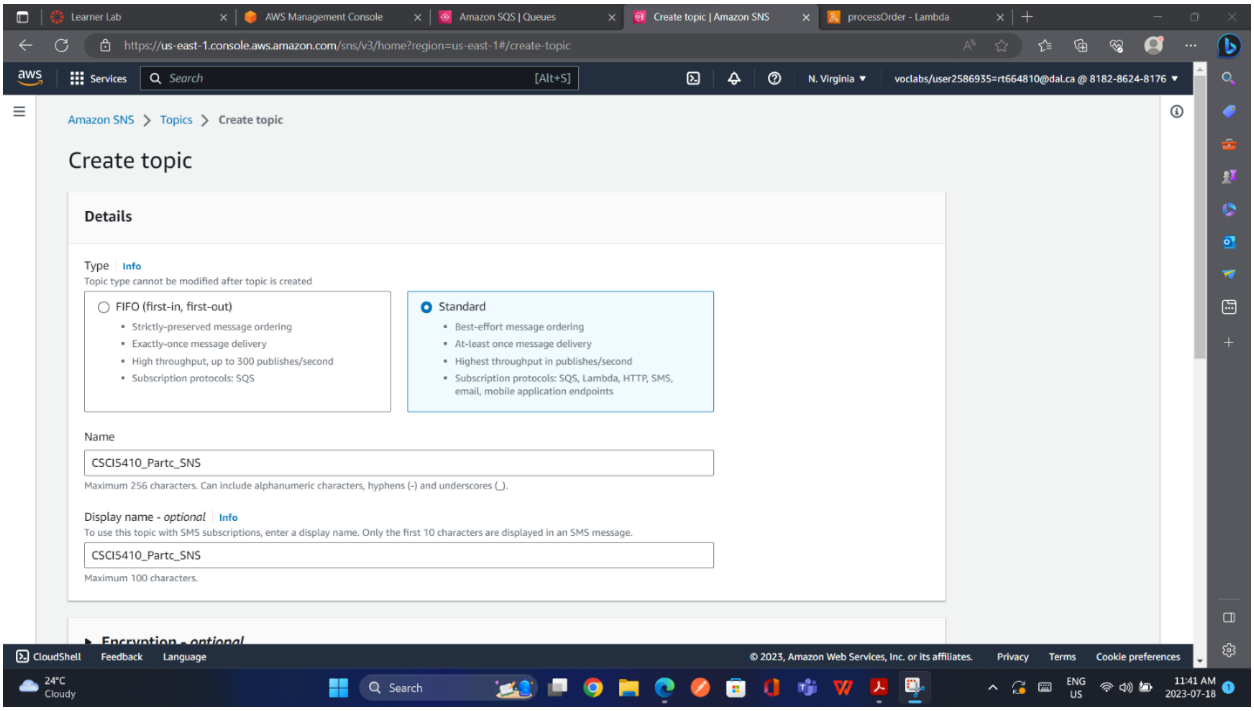


Figure 19 Creation of SNS topic named **CSCI5410_Partc_SNS** [5]

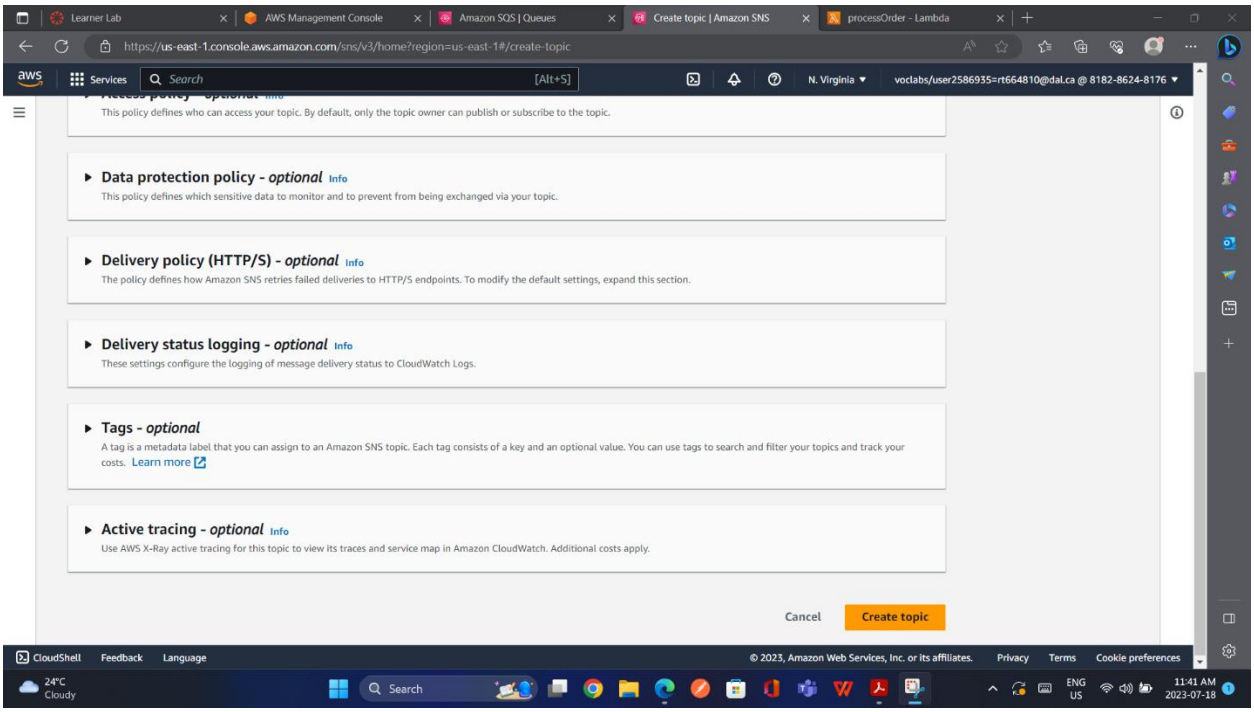


Figure 20 Creation of SNS topic named **CSCI5410_Partc_SNS** [5]

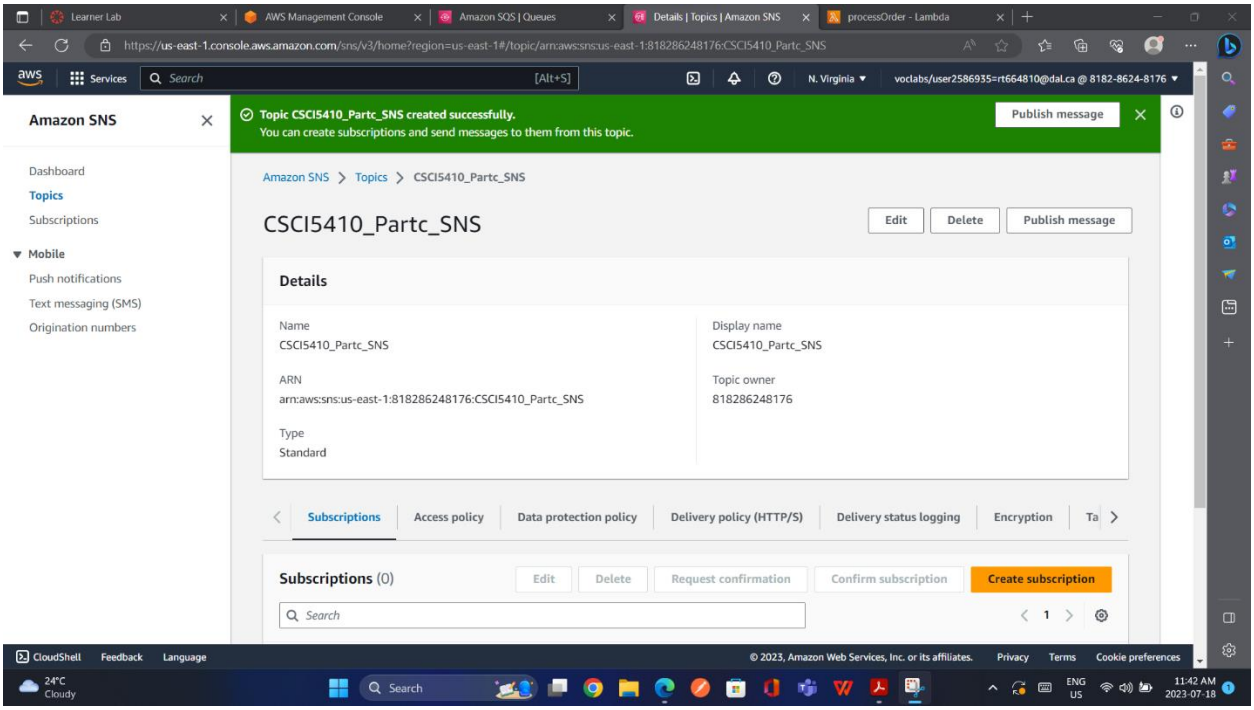


Figure 21 SNS topic named **CSCI5410_Partc_SNS** created successfully [5]

Figure 22 shows that the SNS topic named **CSCI5410_Partc_SNS is** successfully created and listed in Amazon SNS topics.

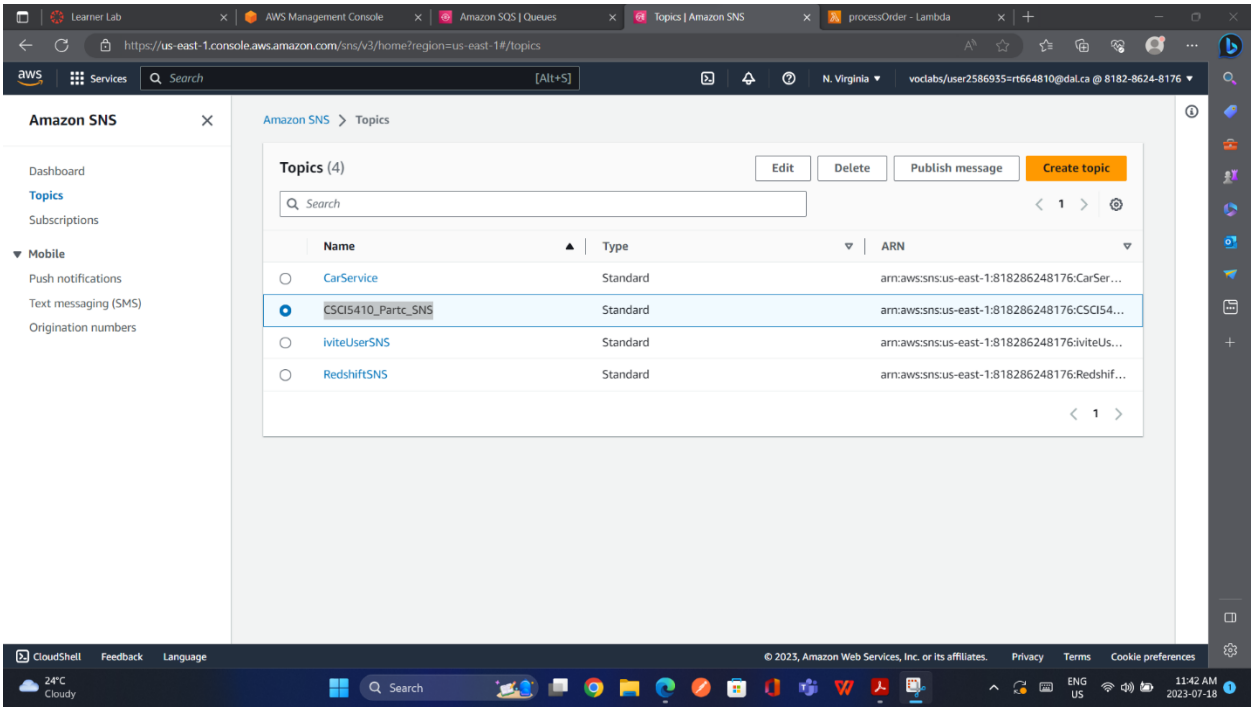


Figure 22 Newly created SNS topic named **CSCI5410_Partc_SNS** [5]

Figure 23 shows that there is no subscription in SNS topic named **CACI5410_Partc_SNS**.

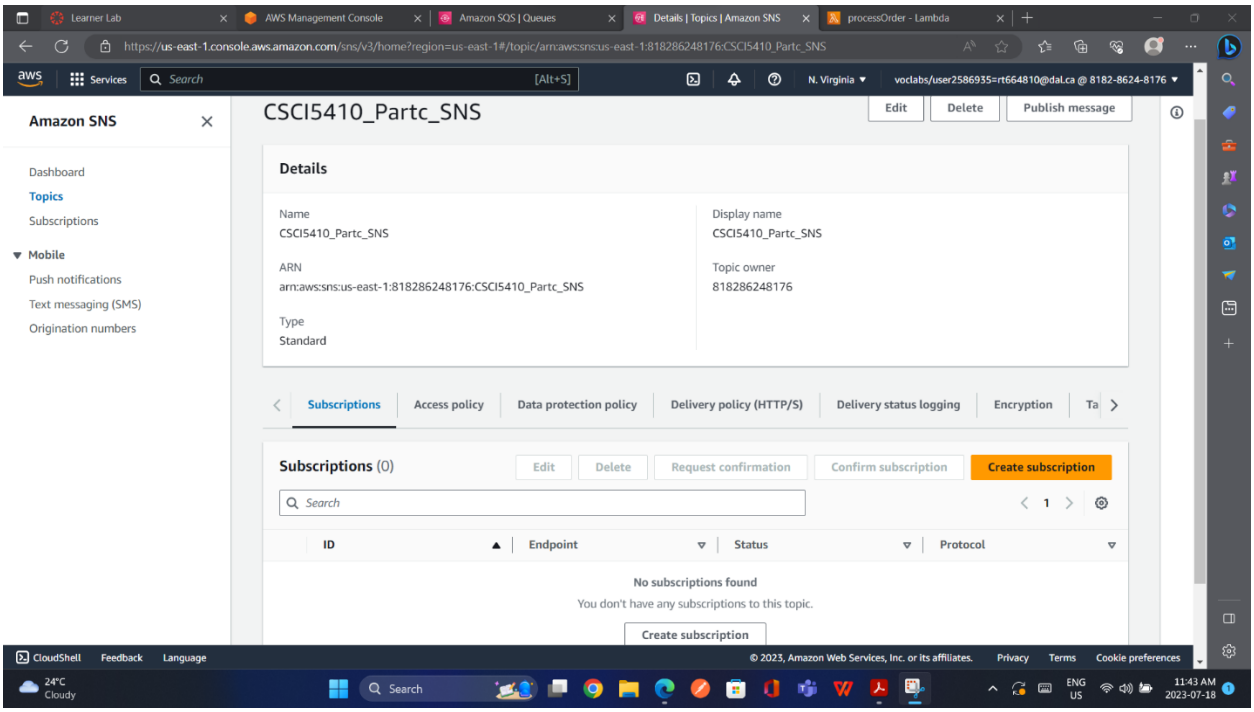


Figure 23 No subscription in **CSCI5410_Partc_SNS** topic [5]

Figure 24 shows the creation of subscription in **CSCI5410_Partc_SNS** topic. In subscription, selected the Topic ARN for the topic, selected email and given the email address in endpoint. Which means that, SNS will send subscription mail to the given email address.

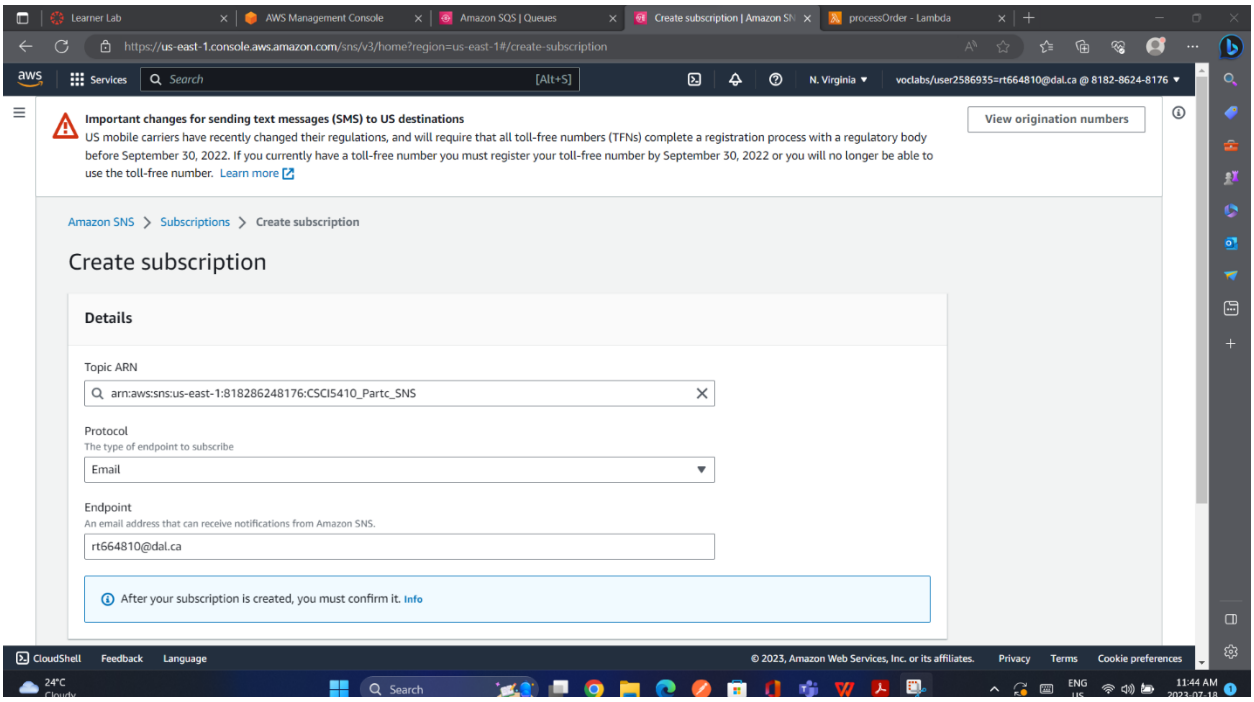


Figure 24 Subscription of email to get order details [5]

Figure 25 shows the successful creation of subscription. On subscription creation, SNS topic sends the subscription email. As of now, status shows the **Pending confirmation** it means SNS sent the subscription email, but recipient has not confirmed the subscription yet.

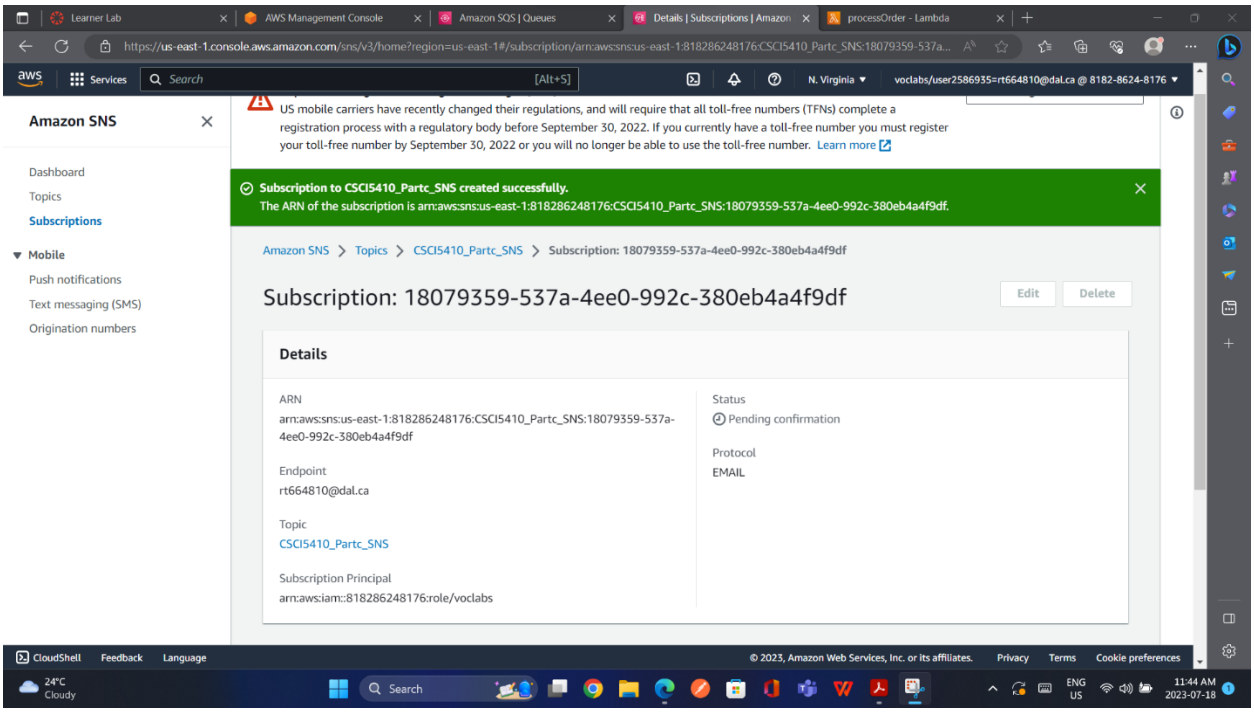


Figure 25 Subscription to **CSCI5410_Partc_SNS** created successfully [5]

Figure 26 show that recipient received the subscription email on creation of subscription in SNS topic. Figure 27 shows that Subscription has been confirmed by **rt664810@dal.ca**

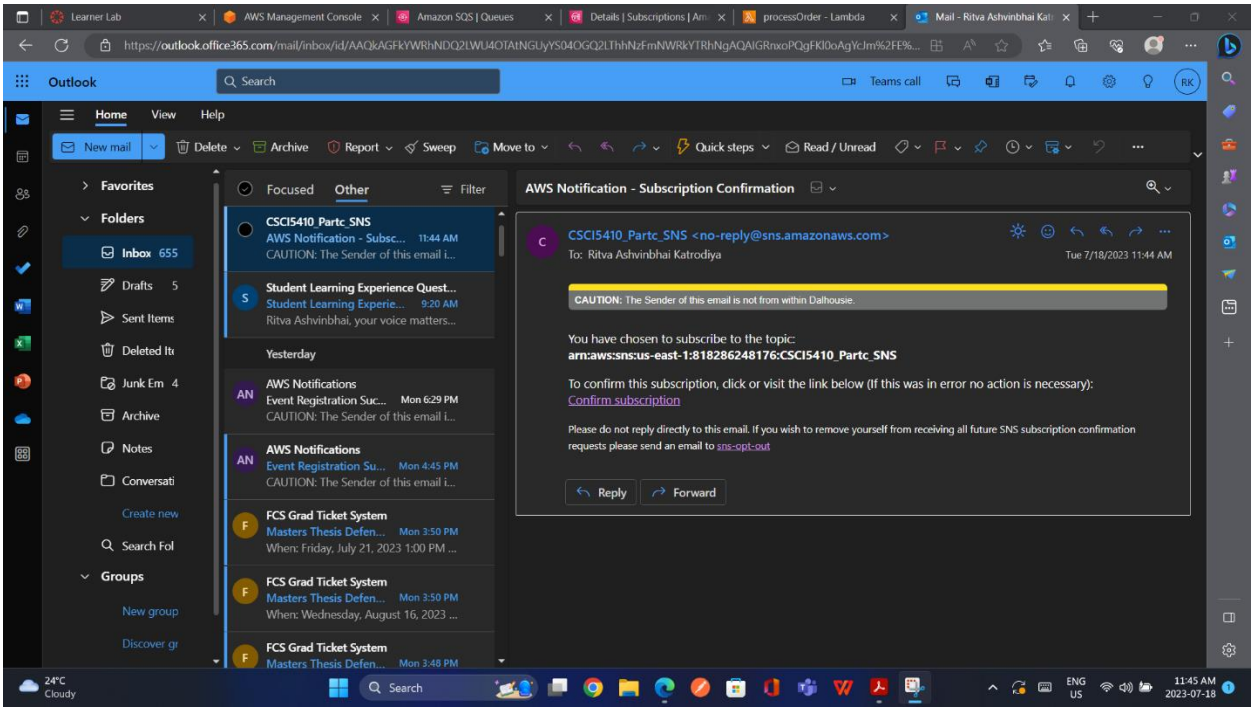


Figure 26 Subscription confirmation mail on **rt664810@dal.ca** [8]

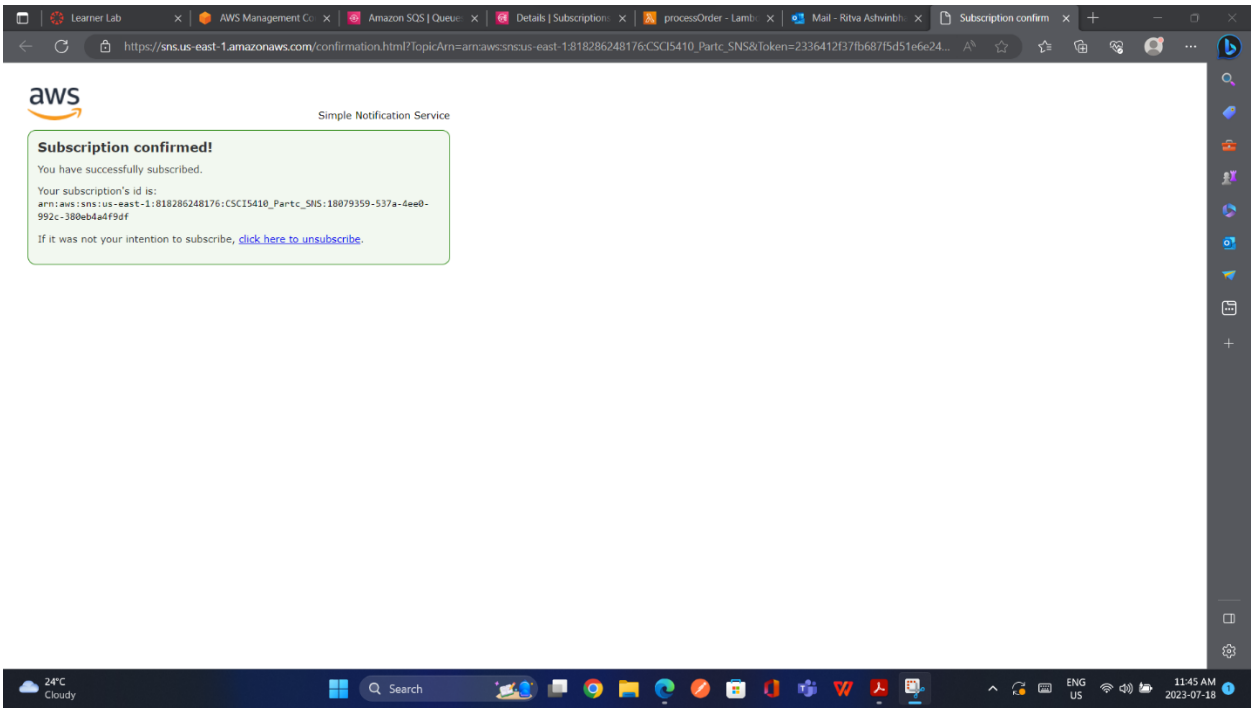


Figure 27 Subscription is confirmed by **rt664810@dal.ca** [5]

Figure 28 illustrates status is **Confirmed** it means recipient with rt664810@dal.ca email address has successfully subscribed for the **CSCI5410_Partc_SNS** topic.

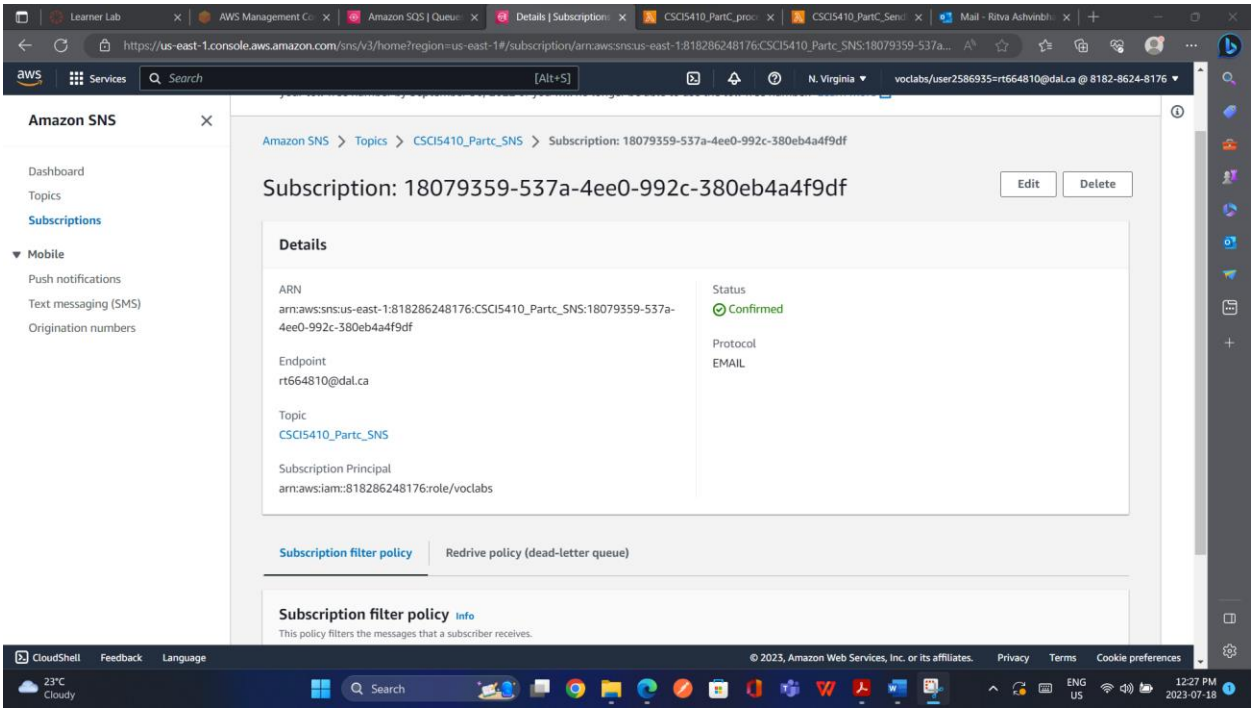


Figure 28 Subscription confirmed with email and ARN [5]

## Creation of lambda function to publish messages to SNS

Figures 29 and 30 show the steps required to create lambda fucntion named **CSCI5410_PartC_SendMessages**. The lambda fucntion is created because it will randomly select car type, accessories and address form the list given in code and publish that as a message in SNS topic named **CSCI5410_PartC_Email**. After that, SNS will queue that message in SQS named **CSCI5410_PArtC_SNS1**.

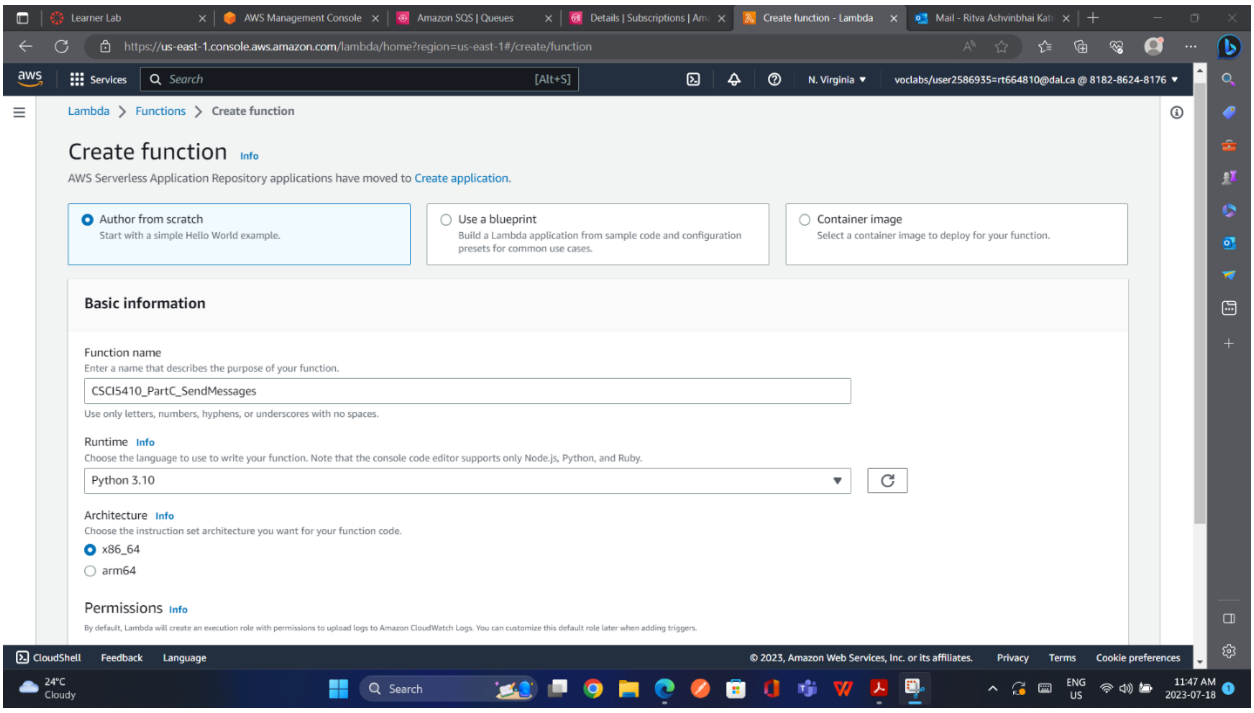


Figure 29 Creation of lambda function named **CSCI5410_PartC_SendMessages** [3]

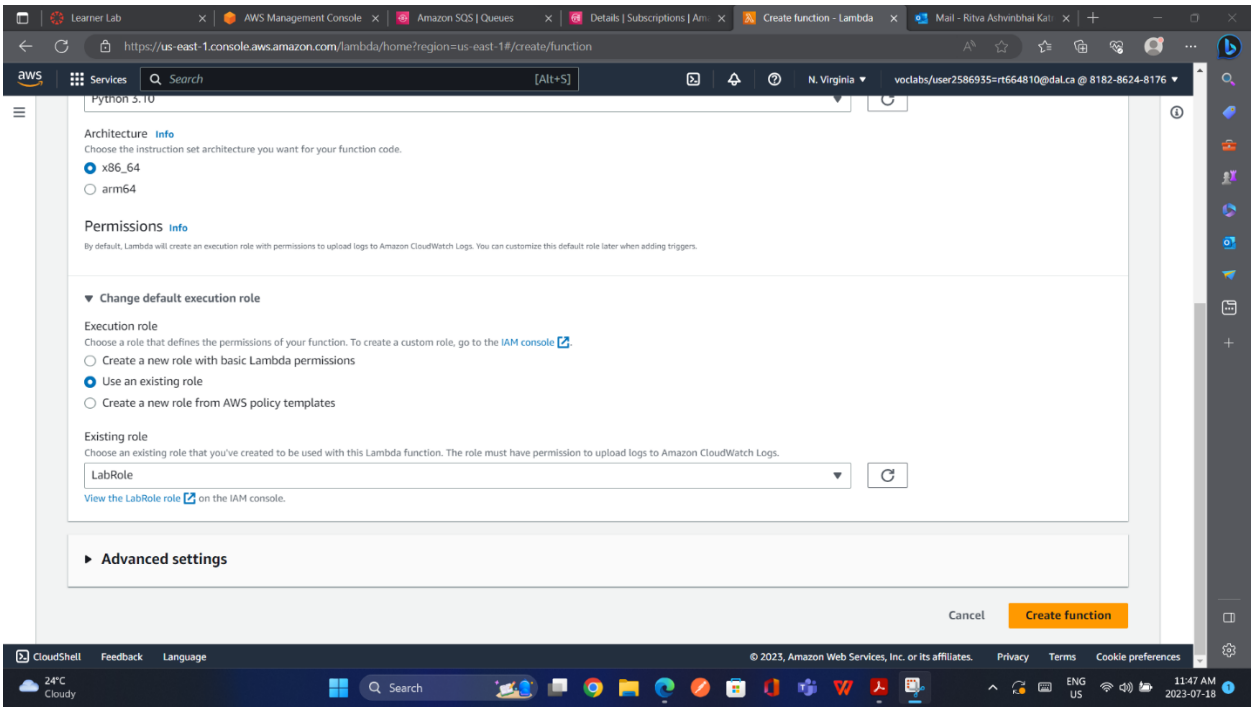


Figure 30 Creation of lambda function named **CSCI5410_PartC_SendMessages** [3]

Figure 31 shows the successfully creation of lambda named **CSCI5410_PartC_SendMessages** and figure 32 shows that fucntion is listed on Amazon lambda functions service.

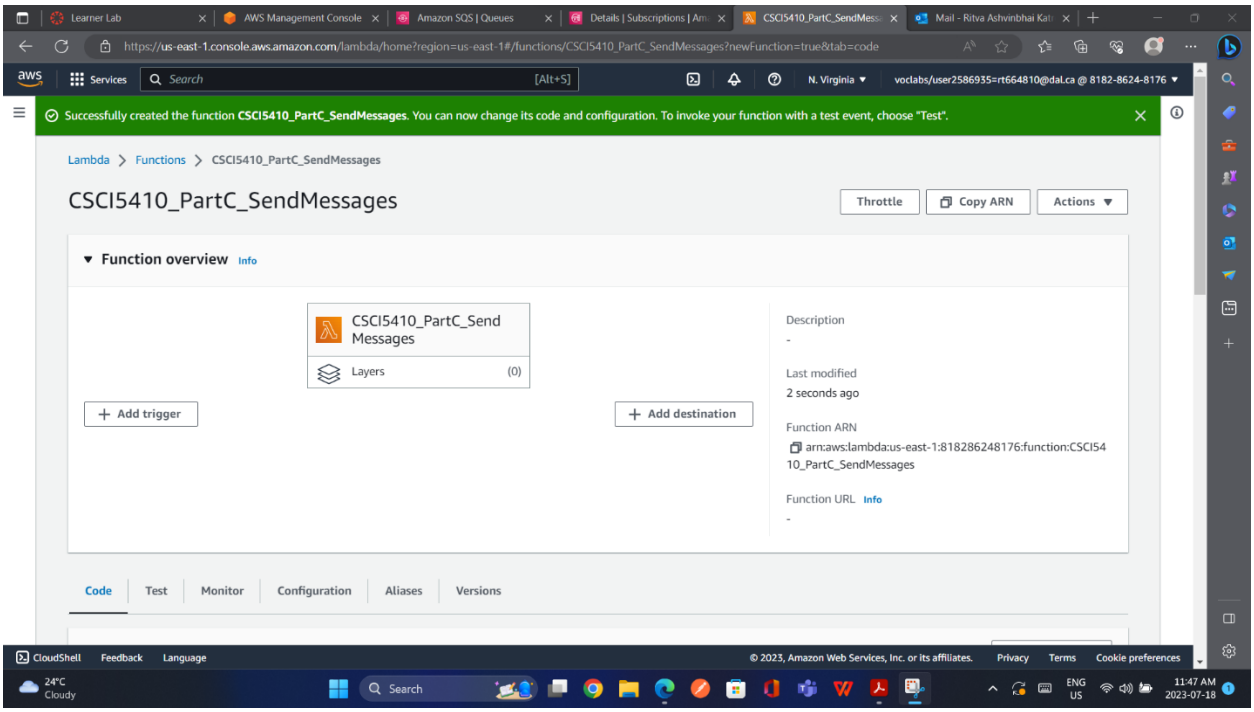


Figure 31 Successfully created lambda function named **CSCI5410_PartC_SendMessages** [3]

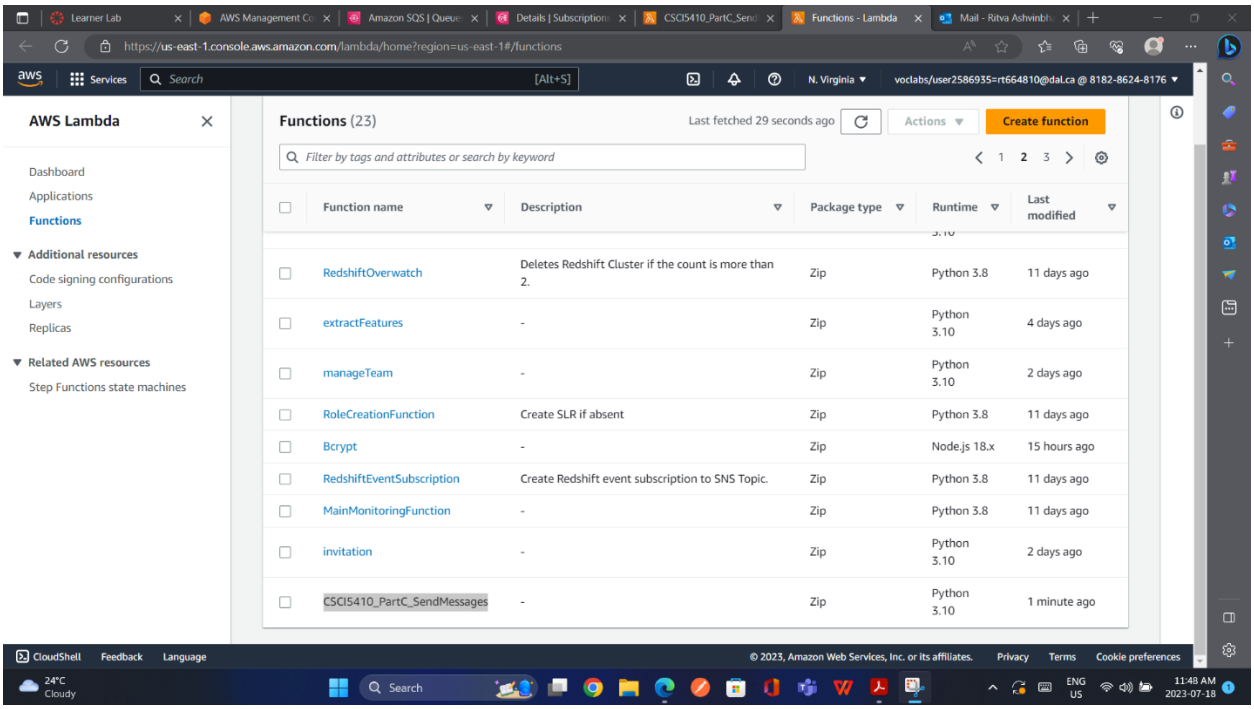


Figure 32 Lambda function named CSCI5410_PartC_SendMessages is listed [3]

Figure 33 shows the lambda function code is written in python which is used to select car type value, accessories value and address value randomly from list and publish those value as a message in SNS topic names **CSCI5410_PartC_Email**.

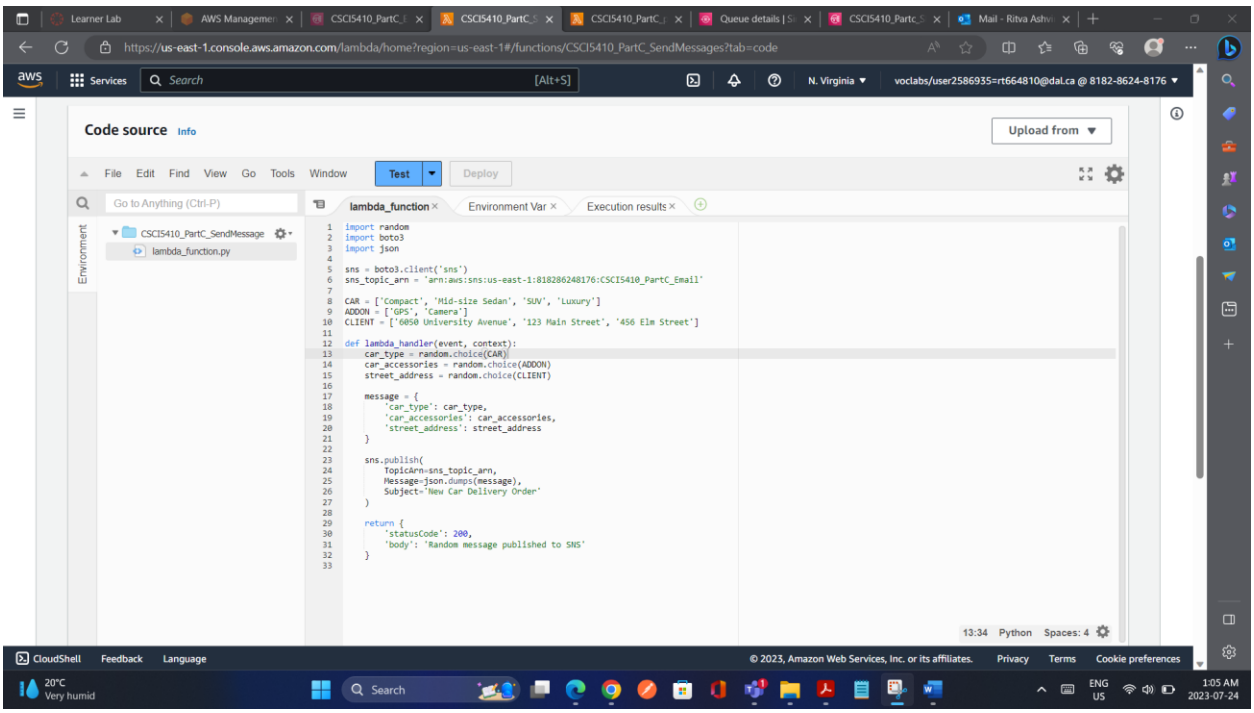


Figure 33 Written lambda function in python to publish messages in SNS [3]

Figure 34 shows the configuration of test event named **test** which is used to test the lambda function.

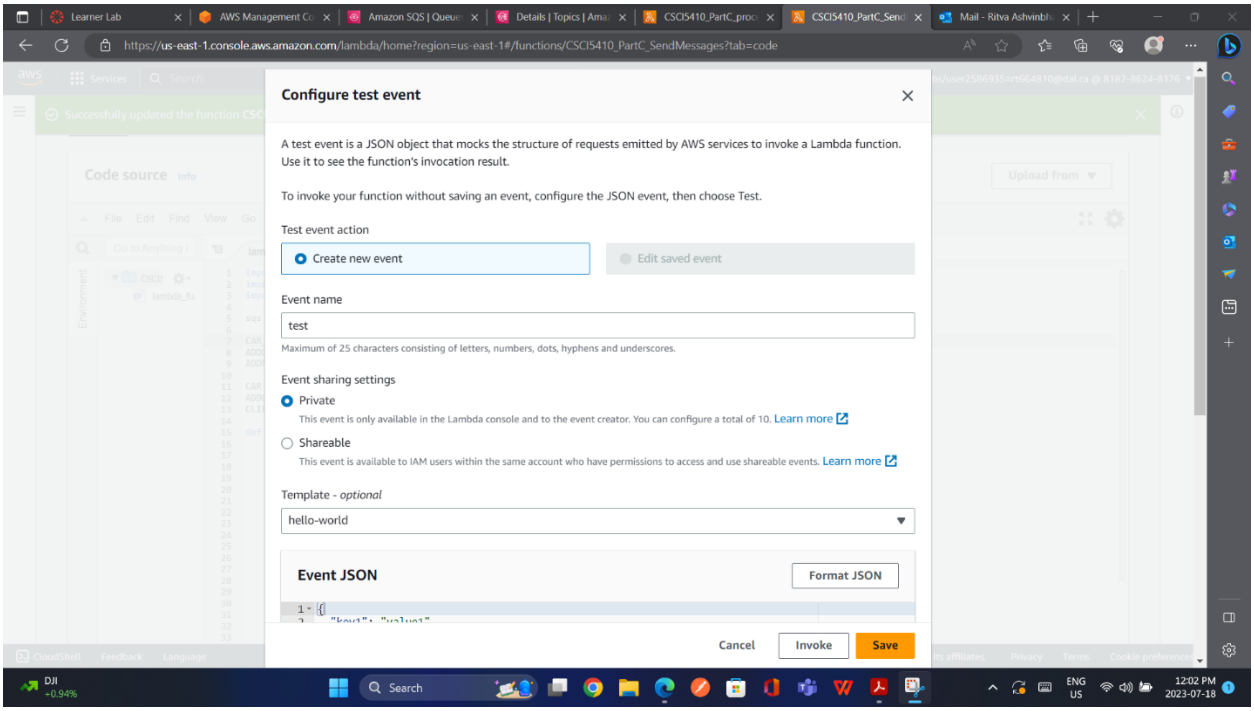


Figure 34 Configured the test event named test to **test** the lambda fucntion [3]

## Creation of lambda function to get the message from queue and place the order details using SNS

Figures 35 and 36 show the steps required to create lambda fucntion named **CSCI5410_PartC_processOrder**. The lambda fucntion is created because it will get the value of car type, accessories and address from the queue as a message **CSCI5410_PArtC_SNS1.** After getting all the value form the queue, lambda fucntion will generate the order details and send the email using SNS topic named **CSCI5410_Partc_SNS** to rt664810@dal.ca email address. This email address is already subscribed to the topic **CSCI5410_Partc_SNS.**

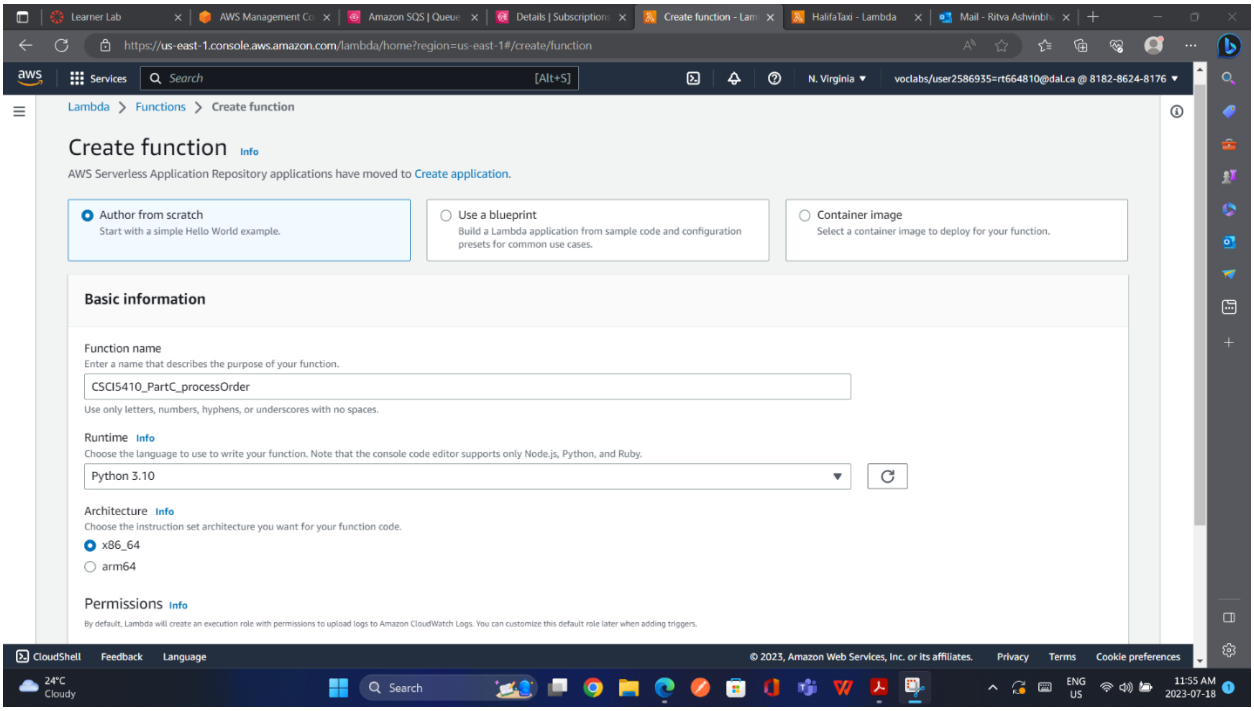


Figure 35 Creation of lambda fucntion named **CSCI5410_PartC_processOrder** [3]

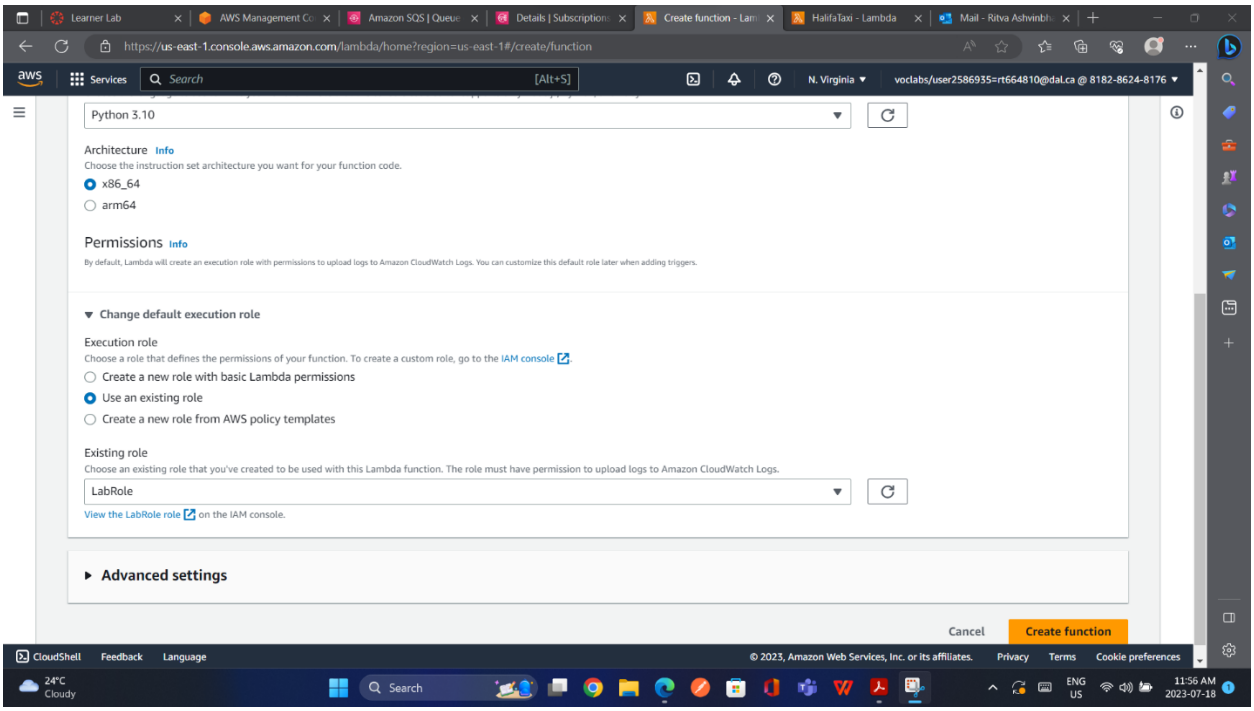


Figure 36 Creation of lambda fucntion named **CSCI5410_PartC_processOrder** [3]

Figure 37 shows the successfully creation of lambda named **CSCI5410_PartC_processOrder** and figure 38 shows that fucntion is listed on Amazon lambda functions service.

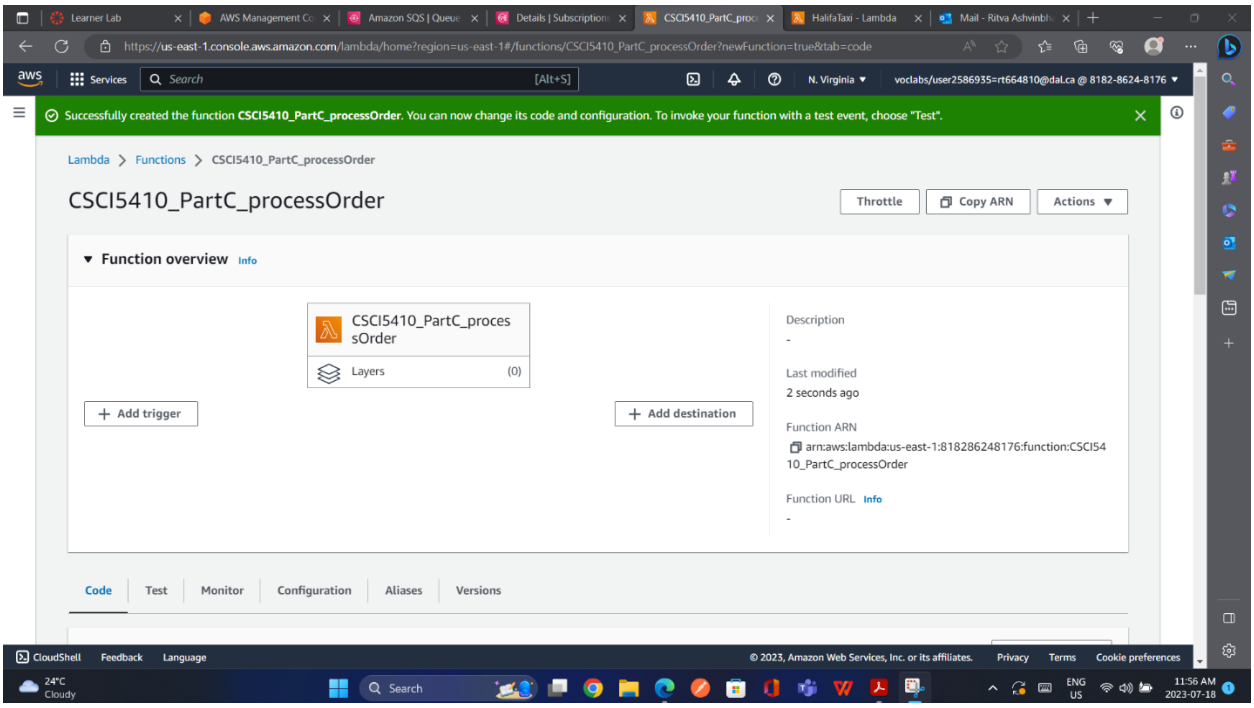


Figure 37 Successfully created lambda function named **CSCI5410_PartC_processOrder** [3]

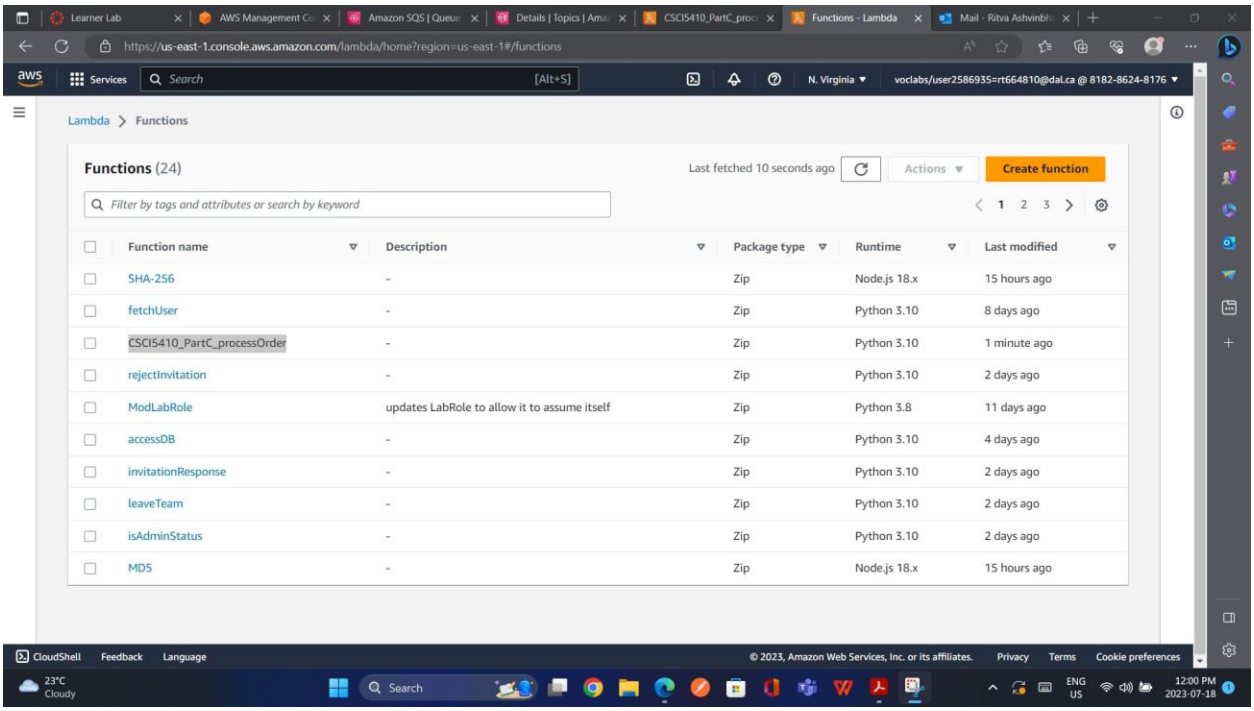


Figure 38 Lambda function named **CSCI5410_PartC_processOrder** is listed [3]

Figure 39 shows the lambda function code is written in python which is used to get the car type, accessories, and address value from the queue, generate order details and send an email using SNS topic to the subscribed email address.

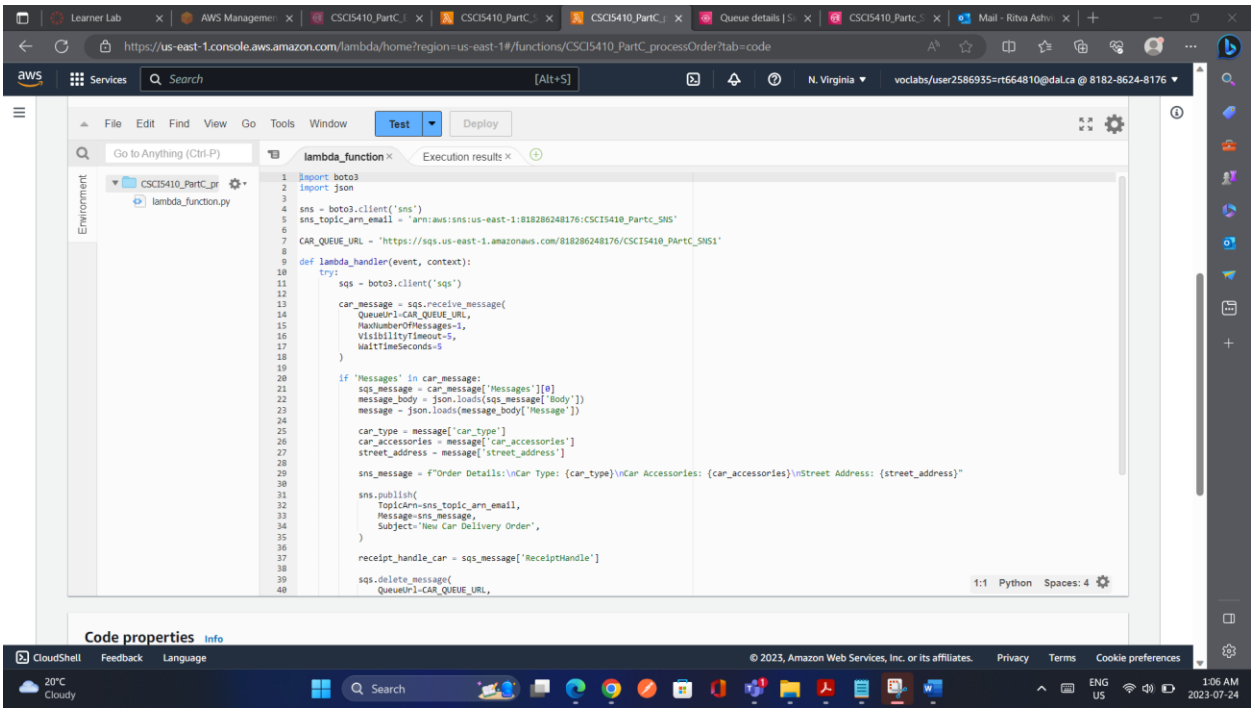


Figure 39 Written lambda function in python to get the available messages from SQS and send the order details by SNS [3]

Figures 40 and 41 show the trigger configuration between **EventBridge** and Lambda named **CSCI5410_PartC_processOrders**. **EventBridge** is added, given rule2 as Rule name and schedule expression for **2 minutes**. It means it will invoke lambda fucntion after **2 minutes**. Figure 42 and 43 show that trigger configured successfully between lambda fucntion and EventBridge.

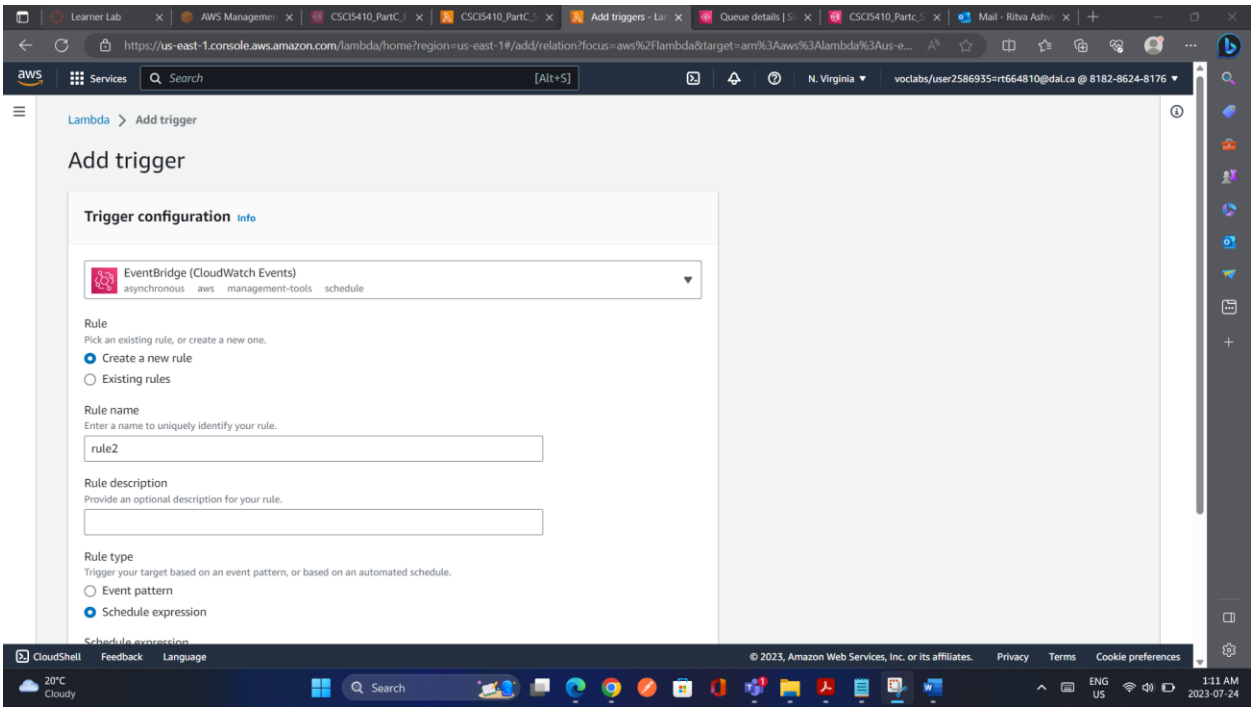


Figure 40 Trigger Configuration between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [7]

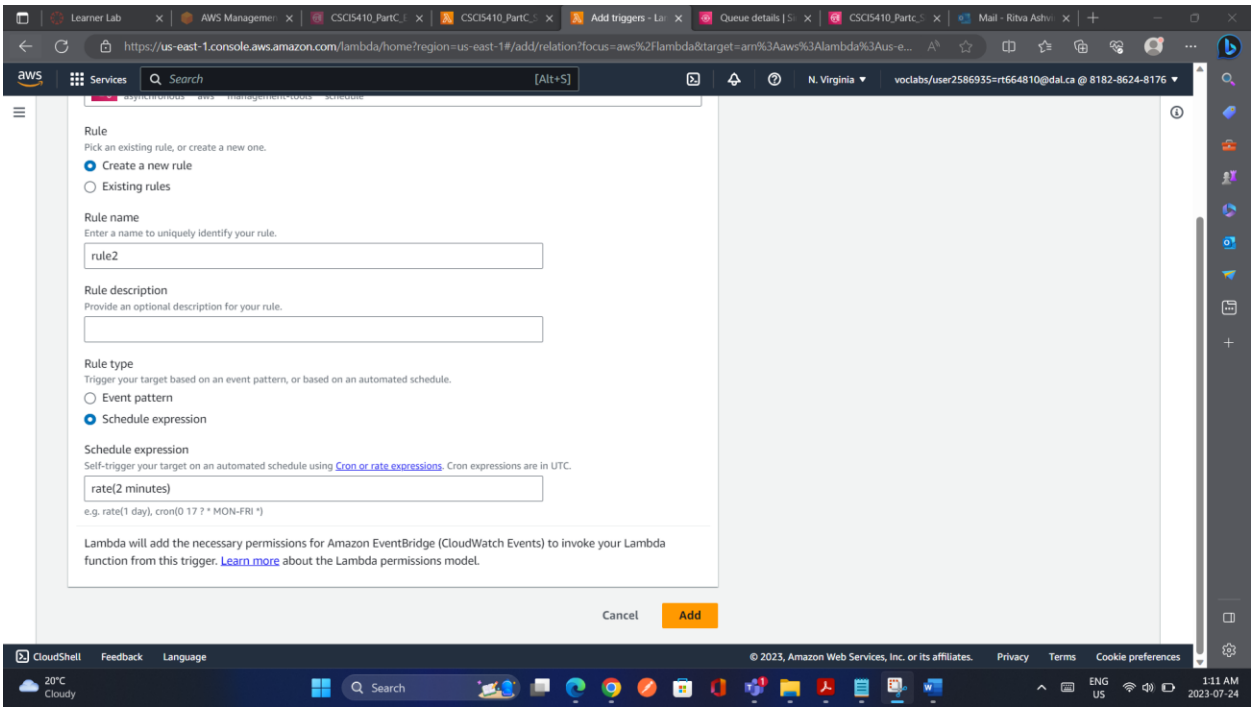


Figure 41 Trigger Configuration between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [7]

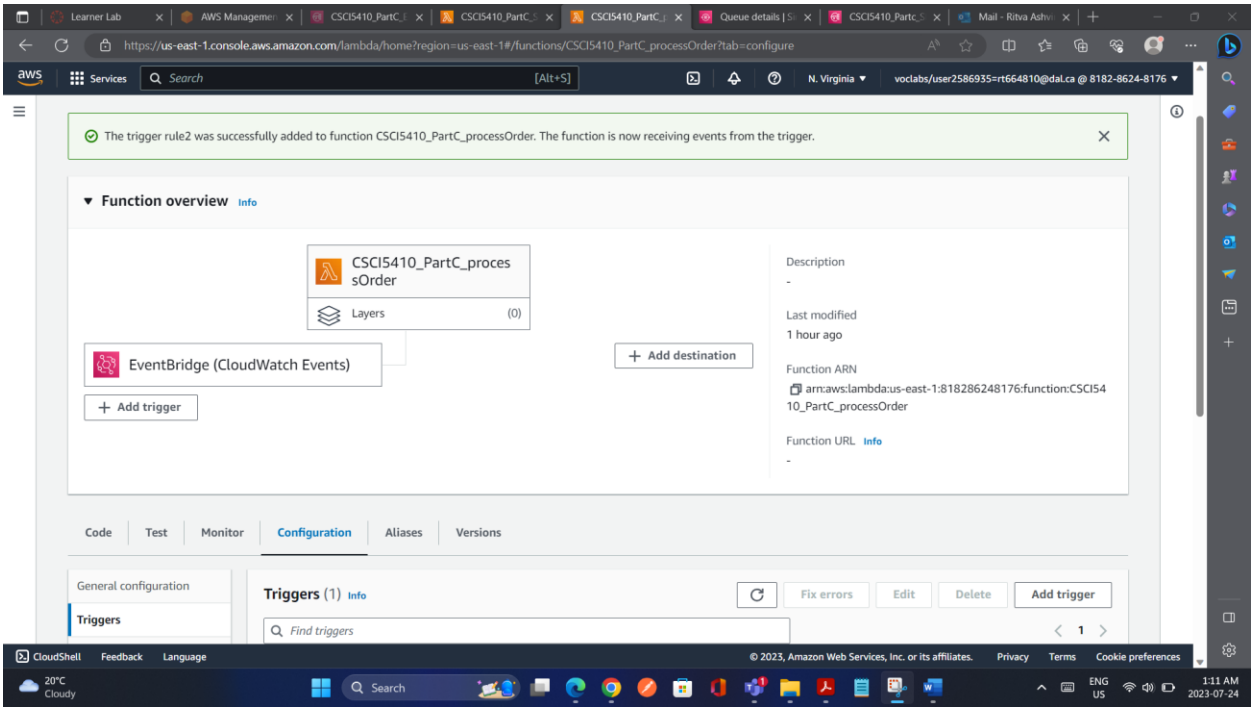


Figure 42 Trigger Configured successfully between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [3]

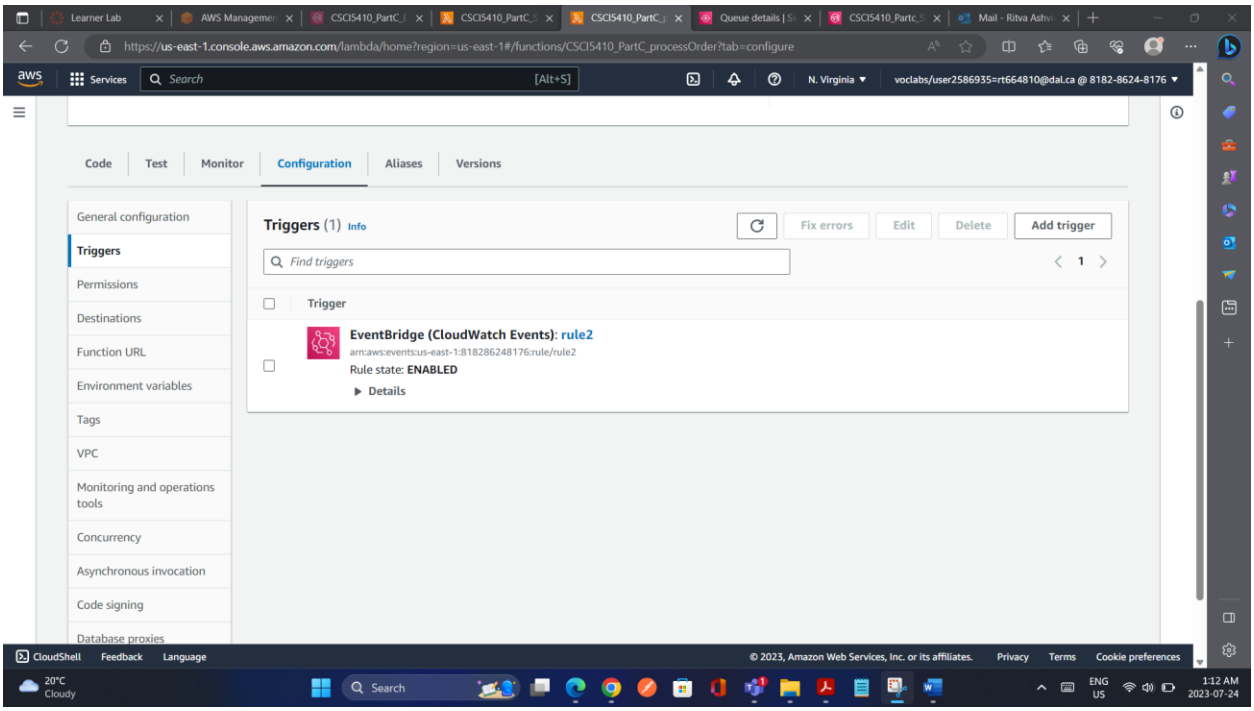


Figure 43 Trigger Configured successfully between **EventBridge** and lambda named **CSCI5410_PartC_processOrder** [3]

Figure 44 show that lambda has been tested successfully with statusCode 200. It means message has been published in SNS and now messages are available in queue **CSCI5410_PArtC_SNS1**.

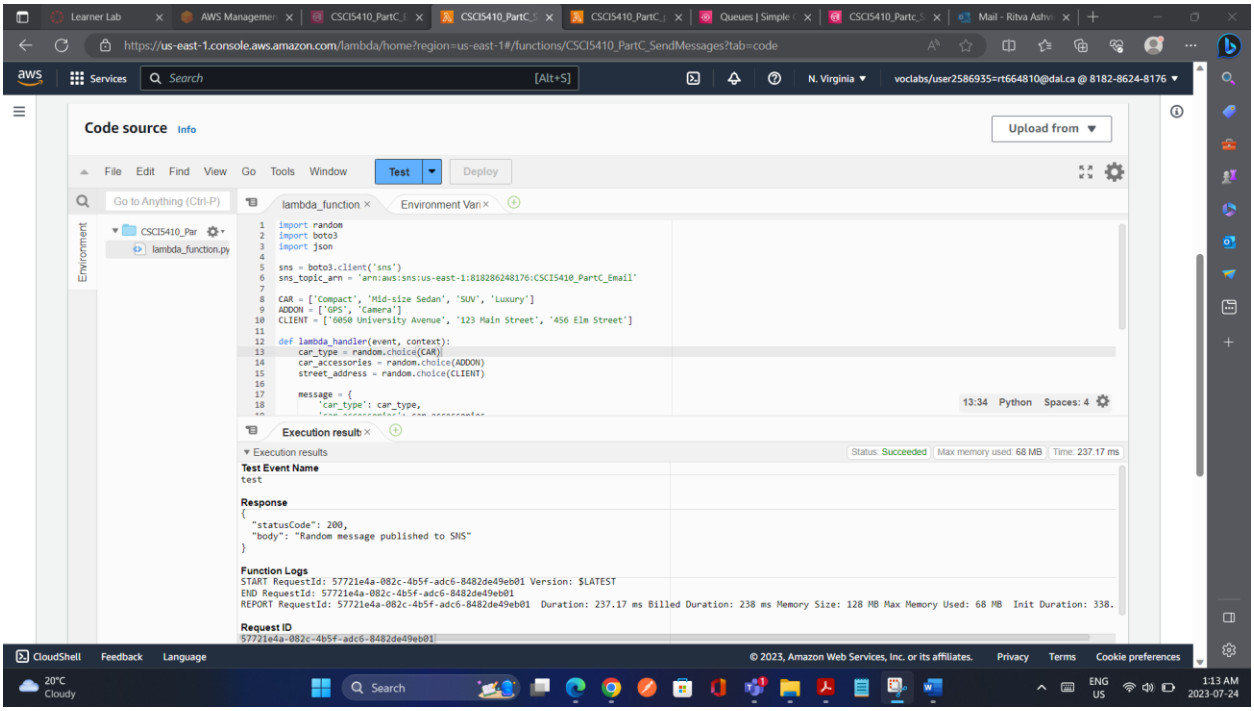


Figure 44 Lambda named **CSCI5410_PartC_SendMessages** tested to publish messages in SNS [3]

Figure 45 shows that 4 messages are available in queue. As per 4 messages , whenever lambda will be triggered after 2 minutes , lambda will send the email with order details to the subscribed email address using SNS.

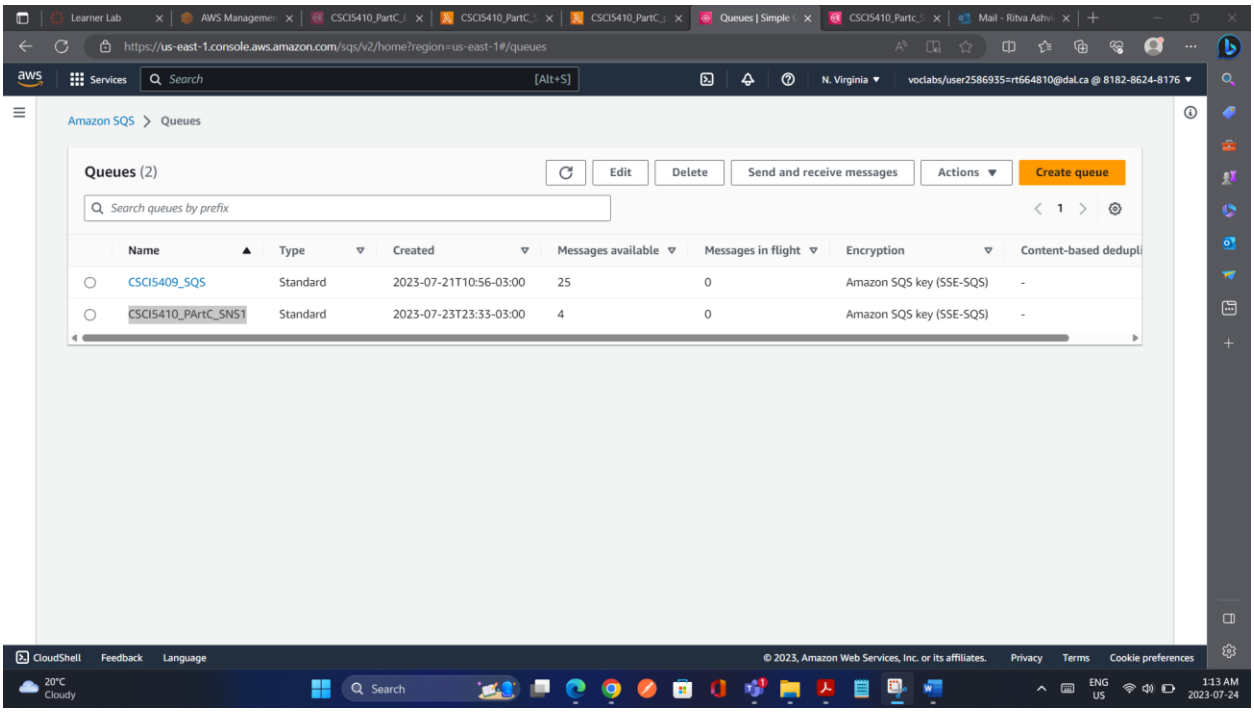


Figure 45 **4 Messages** are available in SQS named **CSCI5410_PArtC_SNS1** [6]

Figure 46,47,48 and 48 show that 4 emails have been received. Figure 46,47,48 and 48 show that 4 emails have been received in after two minutes time.

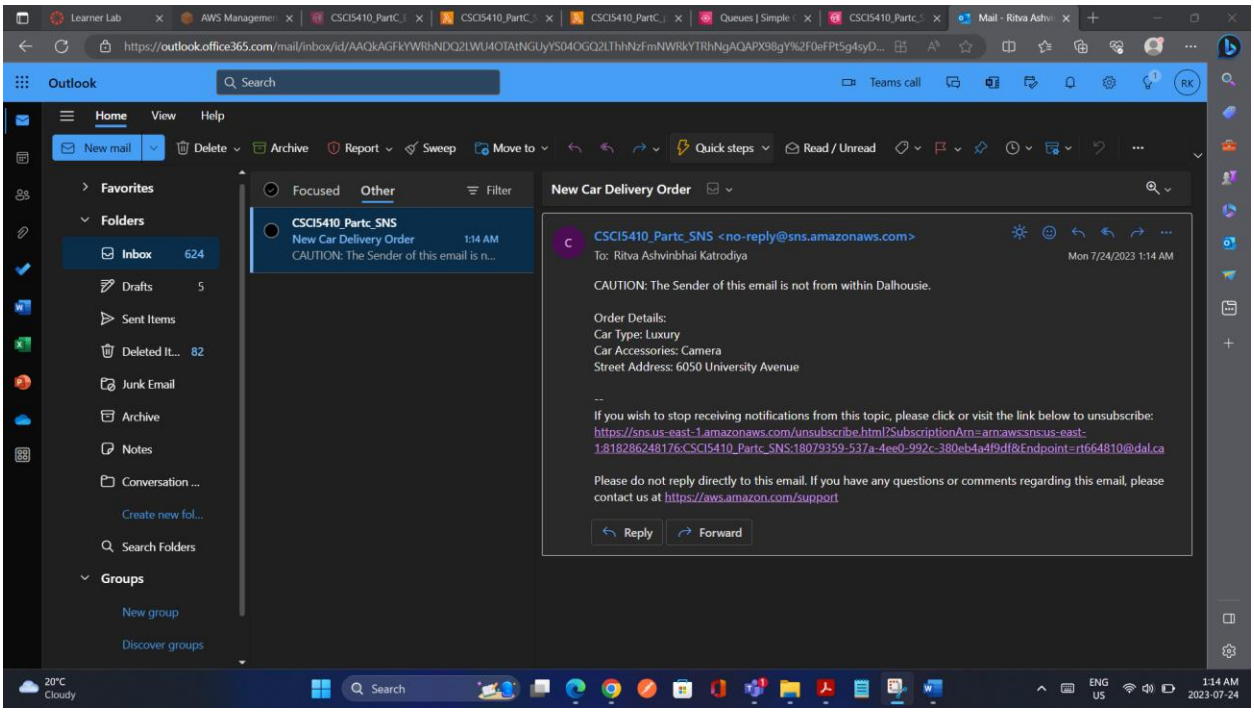


Figure 46 **First time** lambda triggered, and email is available with order details [8]

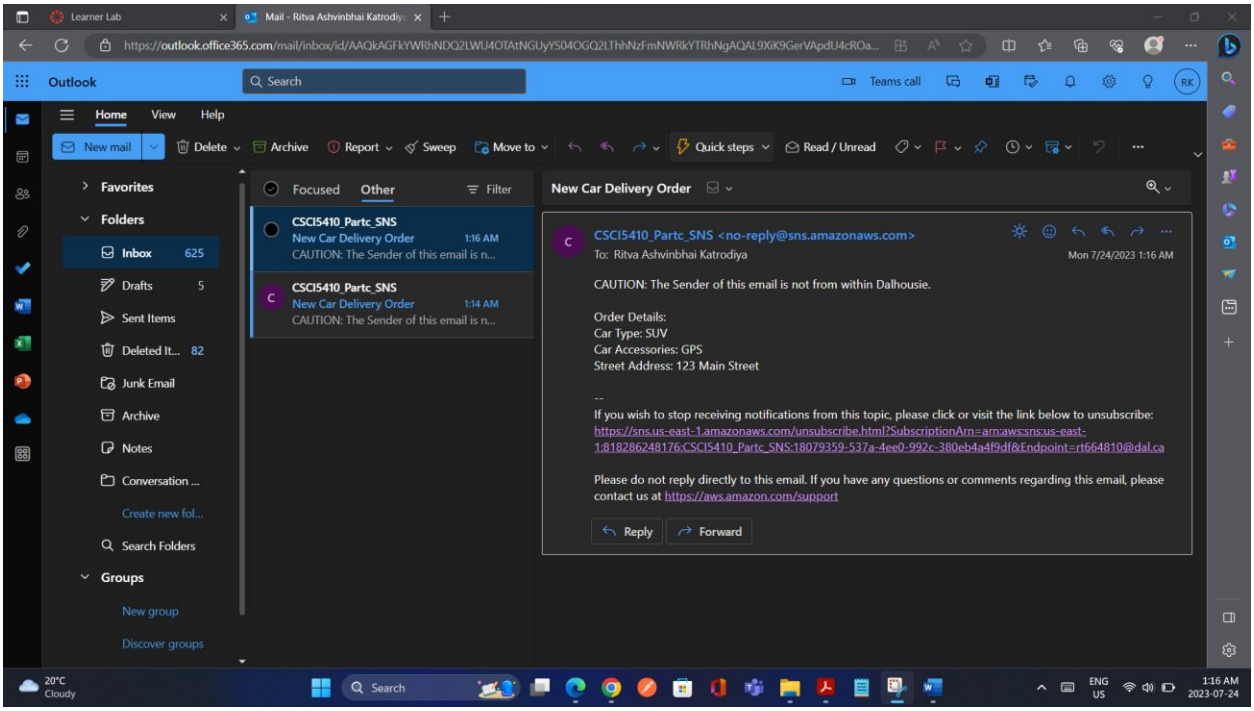


Figure 47 After 2 minutes, second time lambda triggered, and email is available with order details [8]

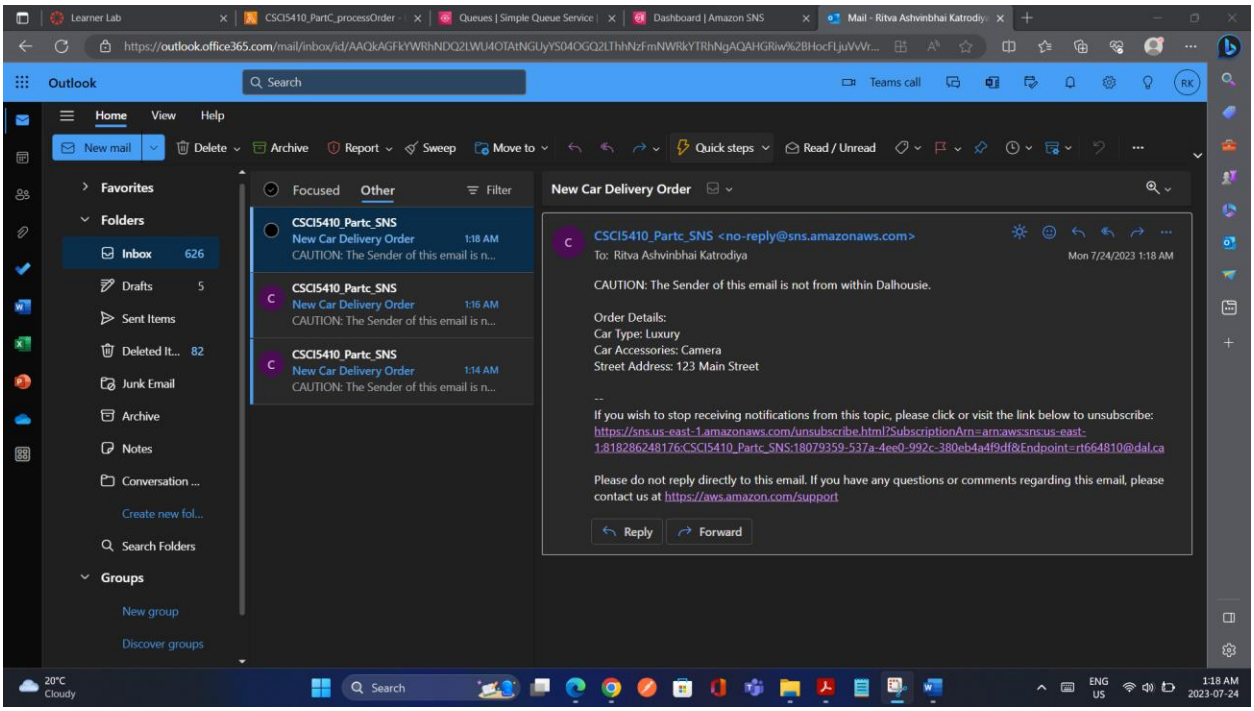


Figure 48 After 2 minutes, third time lambda triggered, and email is available with order details [8]

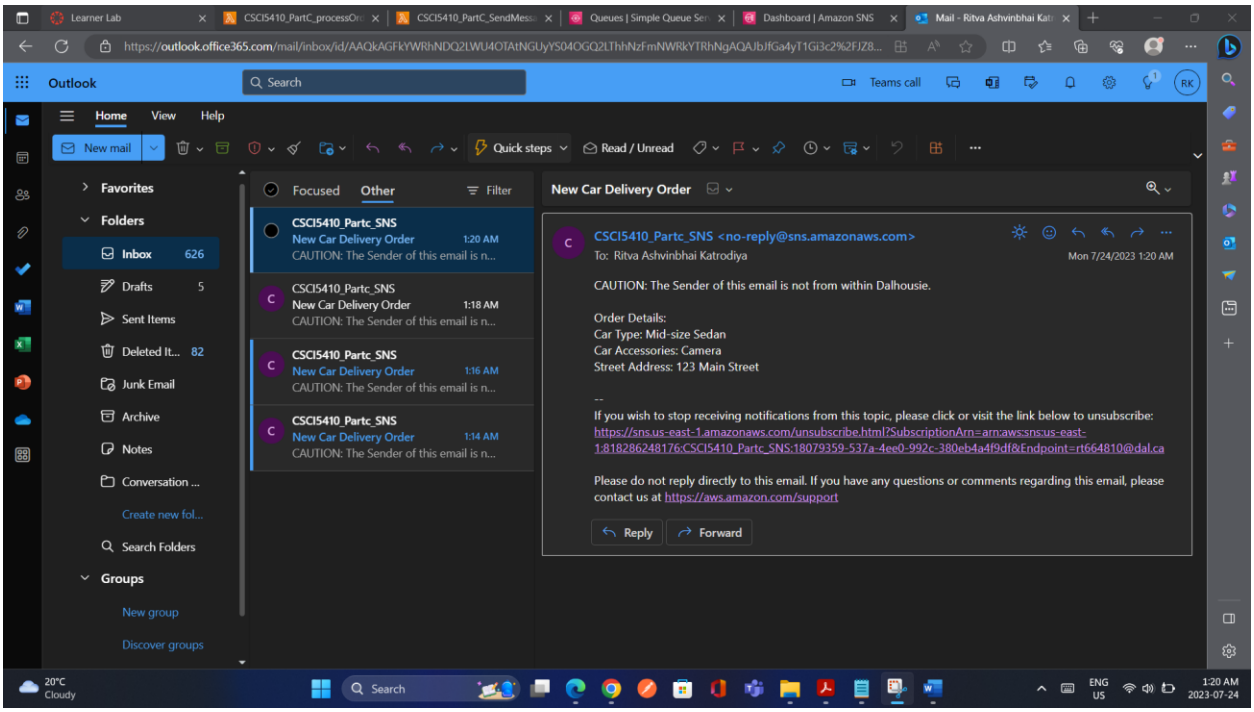


Figure 49 After 2 minutes, second time lambda triggered, and email is available with order details [8]

## Lambda functions Scripts

Pyhton Script of Lambda function named **CSCI5410_PartC_SendMessages** is given below:

lambda_functiopn.py [3]

```
import random
import boto3
import json

# Initialize the SNS client
sns = boto3.client('sns')
# Initialize SNS tipic ARN to publish the message
sns_topic_arn = 'arn:aws:sns:us-east-1:818286248176:CSCI5410_PartC_Email'

# get thew random value from the list for cartype, accessories and address
CarType = ['Compact', 'Mid-size Sedan', 'SUV', 'Luxury']
Accessories = ['GPS', 'Camera','Bluetooth']
Address = ['6050 University Avenue', '123 Main Street', '456 Elm Street']

# this fucntion will select the car type , car accessories and
def lambda_handler(event, context):
    type = random.choice(CarType)
    accessories = random.choice(Accessories)
    address = random.choice(Address)

    message = {
        'car_type': type,
        'car_accessories': accessories,
        'street_address': address
    }

    #  this will publish the message in SNS topic
    sns.publish(
        TopicArn=sns_topic_arn,
        Message=json.dumps(message),
        Subject='New Car Delivery Order'
    )

    return {
        'statusCode': 200,
        'body': 'message published to SNS'
    }
```

```
# References:

# [1] “Building Lambda functions with Python - AWS Lambda,” docs.aws.amazon.com.
# [Online]. Available:  https://docs.aws.amazon.com/lambda/latest/dg/lambda-
python.html. [Accessed: 23 July 2023].
# [2] “Sample function code - AWS Lambda,” Amazon.com, 2023.
# [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/with-sns-
create-package.html. [Accessed: 23 July 2023].
# [3] “Sample Amazon SQS function code - AWS Lambda,” Amazon.com, 2023.
# [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/with-sqs-
create-package.html#with-sqs-example-deployment-pkg-python. [Accessed: 23 July
2023].
```

Pyhton Script of Lambda function named **CSCI5410_PartC_processOrder** is given below:

lambda_functiopn.py [3]

```
import boto3
import json

# Intialize the SNS client
sns = boto3.client('sns')

# Initialize the SNS topic ARN
sns_topic_arn_email = 'arn:aws:sns:us-east-1:818286248176:CSCI5410_Partc_SNS'

# Initialize the SQS URL
QUEUE_URL = 'https://sqs.us-east-
1.amazonaws.com/818286248176/CSCI5410_PArtC_SNS1'

# This function is used to get the message from the queue
def lambda_handler(event, context):
    try:
        sqs = boto3.client('sqs')

        car_message = sqs.receive_message(
            QueueUrl=QUEUE_URL,
            MaxNumberOfMessages=1,
            VisibilityTimeout=5,
            WaitTimeSeconds=5
        )

        # Extract the order details from the message and publish it in SNS
        if 'Messages' in car_message:
```

```
            sqs_message = car_message['Messages'][0]
            message_body = json.loads(sqs_message['Body'])
            message = json.loads(message_body['Message'])

            car_type = message['car_type']
            car_accessories = message['car_accessories']
            street_address = message['street_address']

            sns_message = f"Order Details:\nCar Type: {car_type}\nCar
Accessories: {car_accessories}\nStreet Address: {street_address}"

            # Publish the Order details
            sns.publish(
                TopicArn=sns_topic_arn_email,
                Message=sns_message,
                Subject='New Car Delivery Order',
            )

            receipt_handle_car = sqs_message['ReceiptHandle']

            sqs.delete_message(
                QueueUrl=QUEUE_URL,
                ReceiptHandle=receipt_handle_car
            )

            print("Successfully processed the message.")
        else:
            print("No messages in the SQS queue.")

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        raise

    return {
        'statusCode': 200,
        'body': 'Processing complete'
    }


# References:
# [1] “Building Lambda functions with Python - AWS Lambda,” docs.aws.amazon.com.
# [Online]. Available:  https://docs.aws.amazon.com/lambda/latest/dg/lambda-
python.html. [Accessed: 23 July 2023].
# [2] “Sample function code - AWS Lambda,” Amazon.com, 2023.
```

```
# [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/with-sns-
create-package.html. [Accessed: 23 July 2023].
# [3] “Sample Amazon SQS function code - AWS Lambda,” Amazon.com, 2023.
# [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/with-sqs-
create-package.html#with-sqs-example-deployment-pkg-python. [Accessed: 23 July
2023].
```

# References

[1] "Sample function code - AWS Lambda," Amazon.com, 2023.[Online].Available: https://docs.aws.amazon.com/lambda/latest/dg/with-sns-create-package.html. [Accessed: 23 July 2023].

[2] "Sample Amazon SQS function code - AWS Lambda," Amazon.com, 2023. [Online].Available: https://docs.aws.amazon.com/lambda/latest/dg/with-sqs-create-package.html#with-sqs-example-deployment-pkg-python. [Accessed: 23 July 2023].

[3] Amazon, "AWS Lambda – Serverless Compute - Amazon Web Services," Amazon Web Services, Inc., 2019. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 23 July 2023].

[4] "Building Lambda functions with Python - AWS Lambda," docs.aws.amazon.com. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html. [Accessed: 23 July 2023].

[5] AWS, "Amazon Simple Notification Service (SNS) | AWS," Amazon Web Services, Inc., 2019. [Online]. Available: https://aws.amazon.com/sns/. [Accessed: 23 July 2023].

[6] AWS, "Amazon Simple Queue Service (SQS) | Message Queuing for Messaging Applications | AWS," Amazon Web Services, Inc., 2018. [Online]. Available: https://aws.amazon.com/sqs/. [Accessed: 23 July 2023].

[7] "Amazon EventBridge - Amazon Web Services," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/eventbridge/. [Accessed: 23 July 2023].

[8] "Outlook.com - Microsoft free personal email," Live.com, 2019. [Online]. Available: https://outlook.live.com/owa/. [Accessed: 23 July 2023].