CSCI 5410

Serverless Data Processing (Summer 2023)

MASTER OF APPLIED COMPUTER SCIENCE

Assignment-3 (Part C):

Name: **Jainil Sevalia** | Banner Id : **B00925445** |
Email:**jn498899@dal.ca**

Table of figures:

Halifax Taxi:

Architecture:

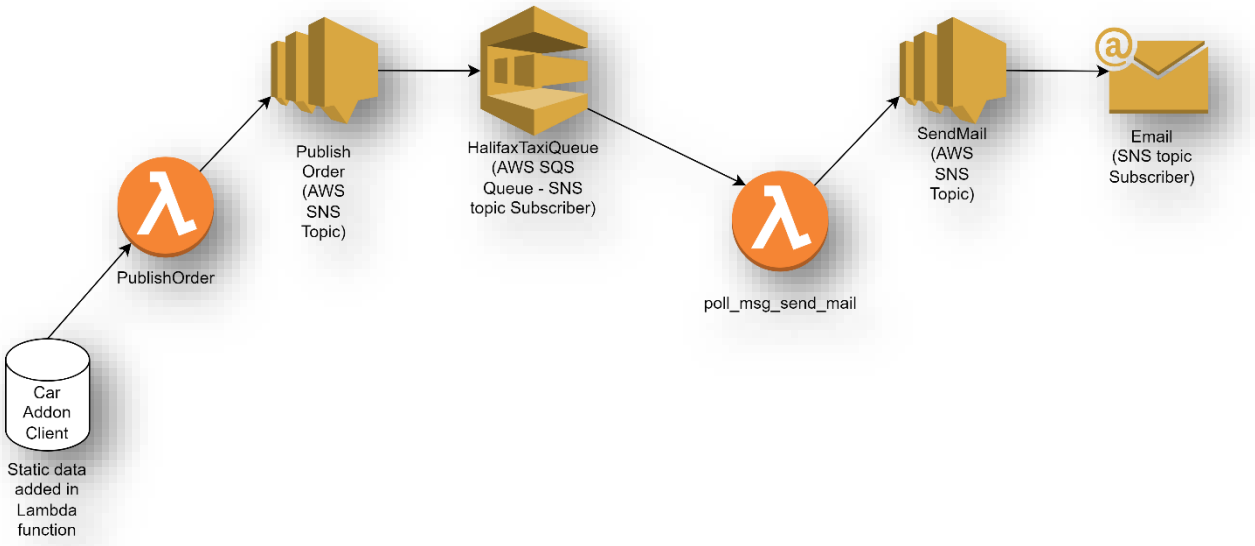

Figure 1 : Overall Architecture of Halifax Taxi

- Created a HalifaxTaxiQueue using AWS SQS service. This Queue will store all messages(Orders Information).

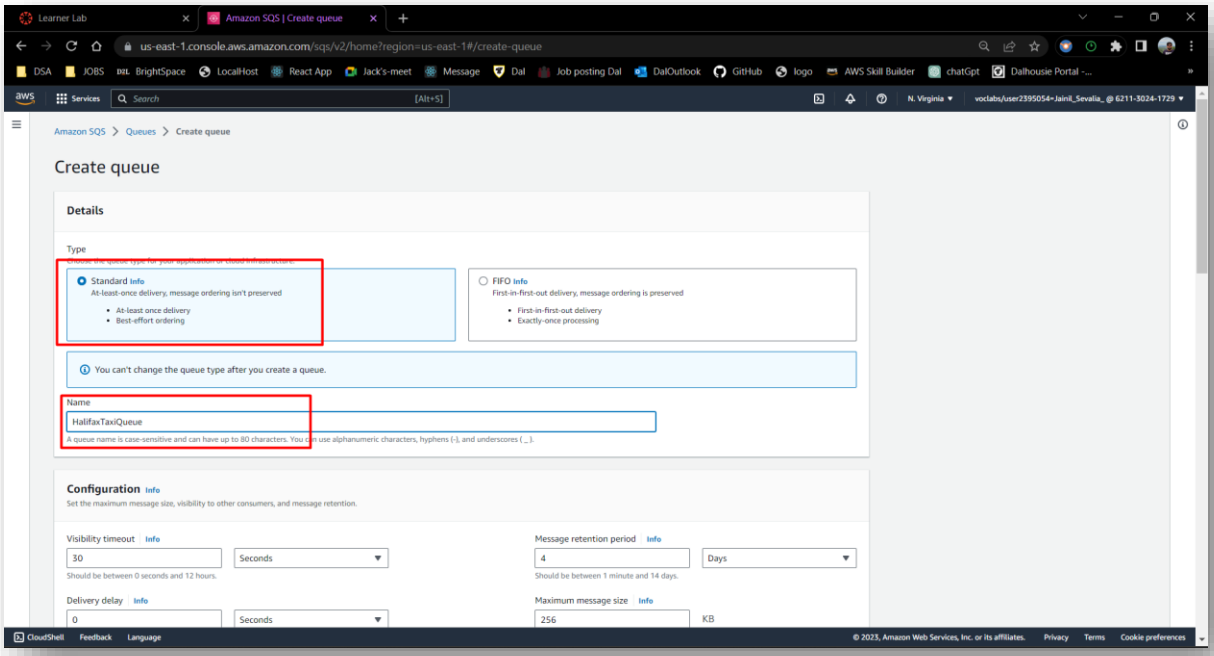

Figure 2 : HalifaxTaxiQueue created (AWS SQS).

- Queue Created successfully.

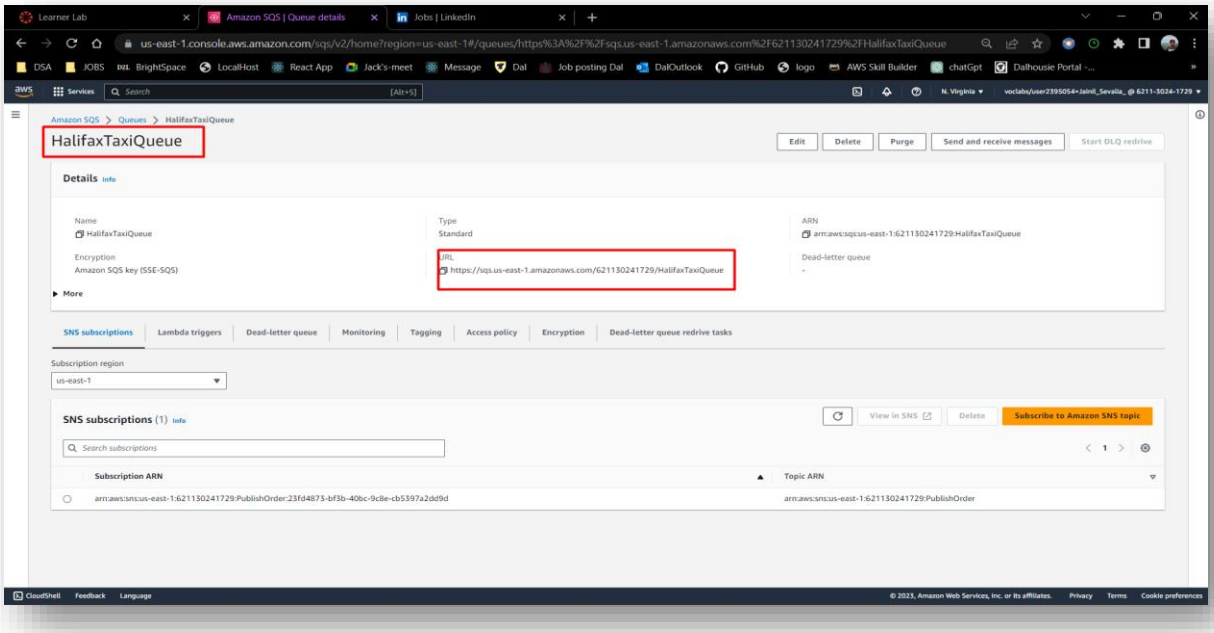

Figure 3 : Queue created and got Queue URL.

- Create AWS SNS topic. First AWS Lambda(Which randomly generate orders) will send order message to this SNS topic.

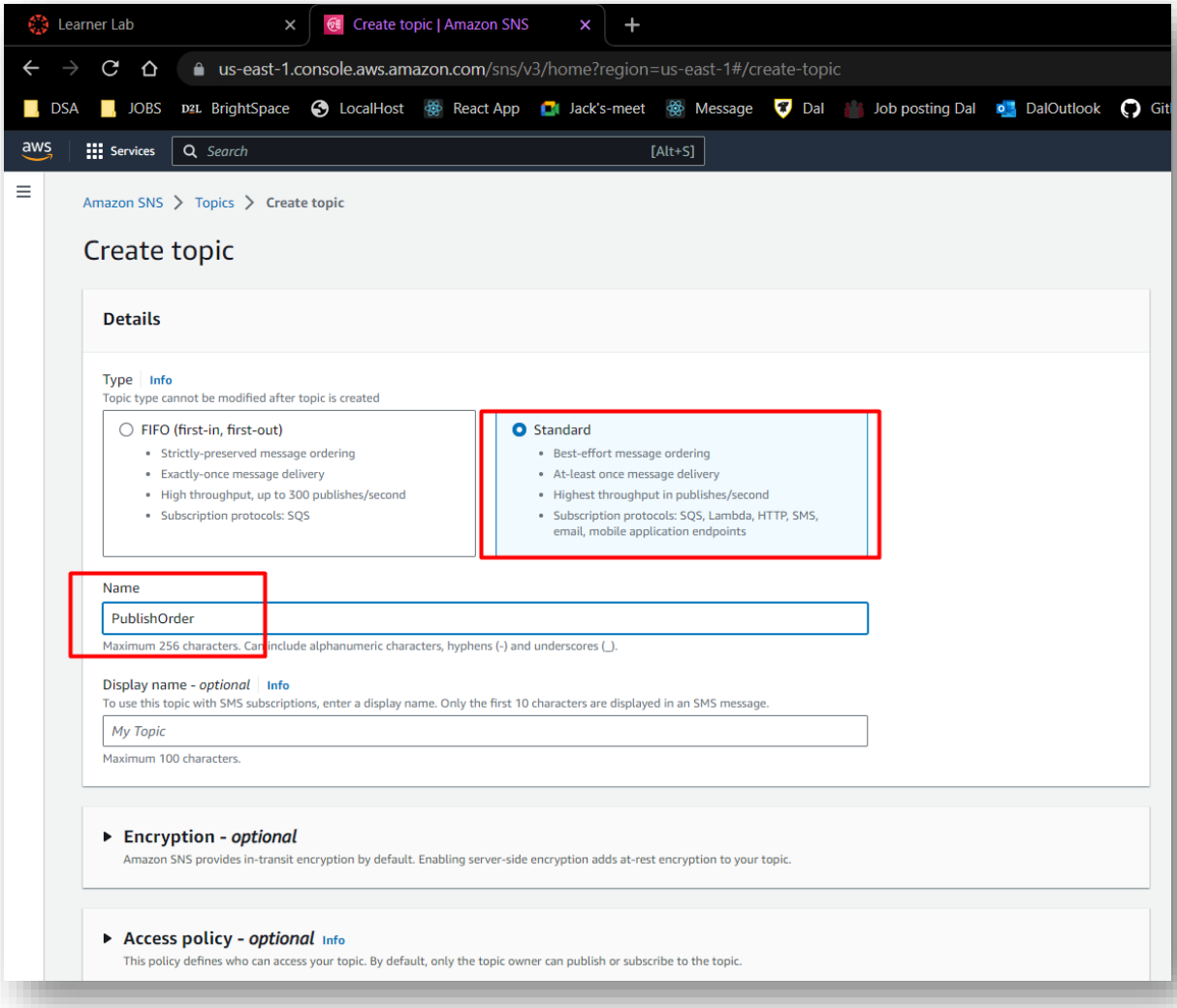

Figure 4 : Creating Publish Order SNS topic.

- Successfully created PublishOrder SNS topic.

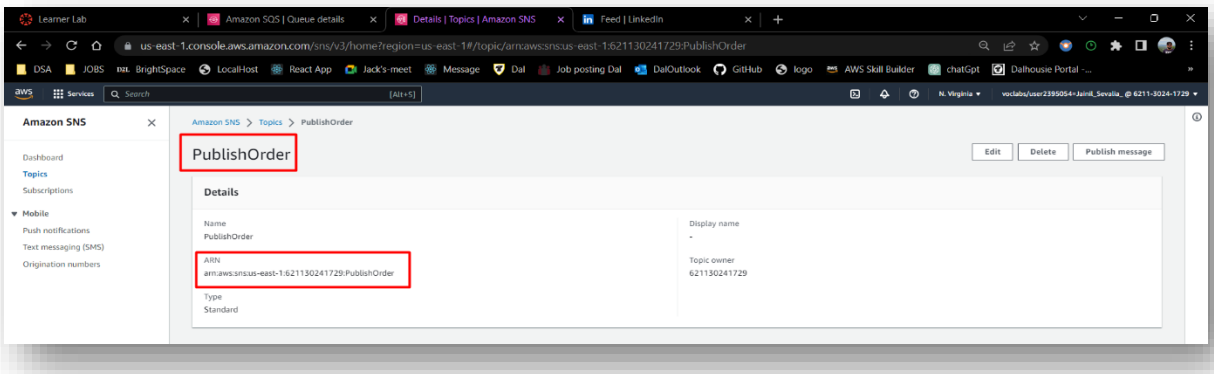

Figure 5: Successfully created PublishOrder SNS topic.

- SQS Queue is Subscribed to SNS topic. So, message will publish to SQS.

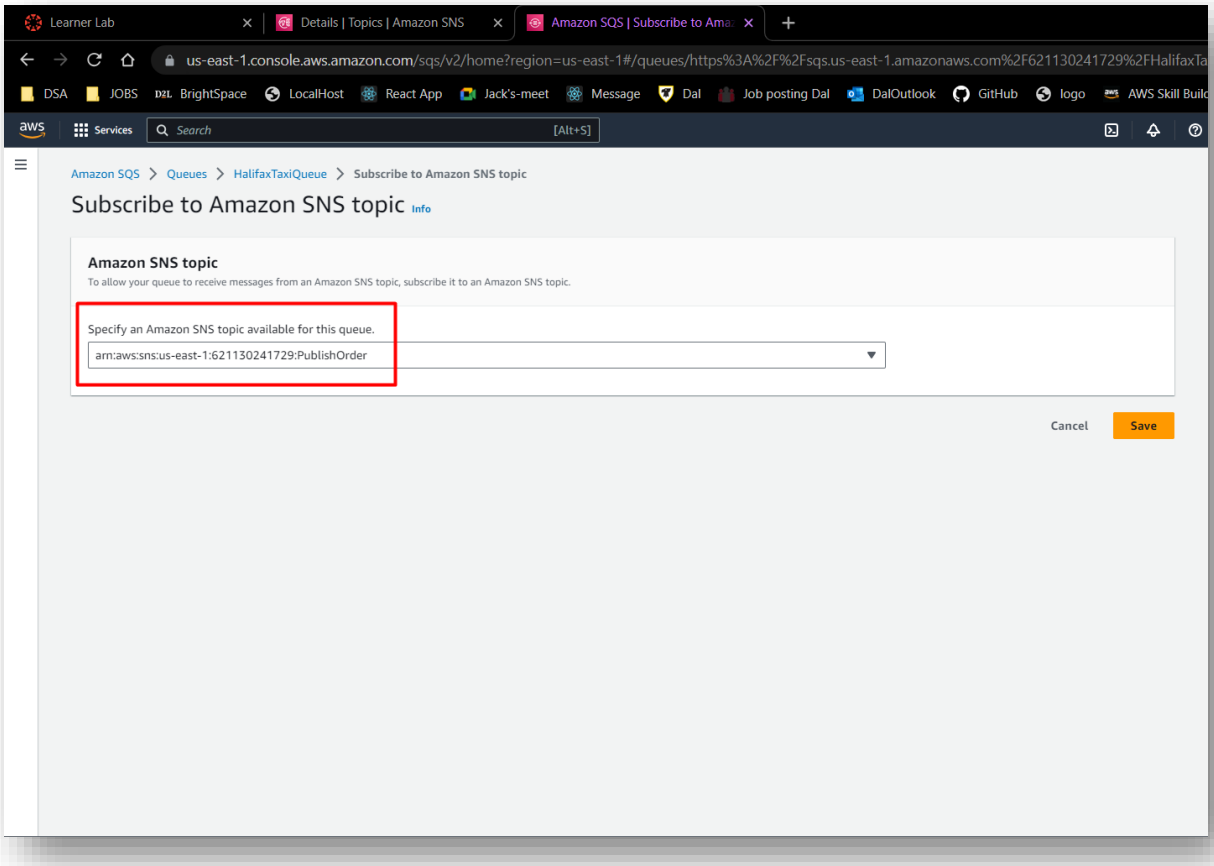

Figure 6 : SQS subscribed to SNS topic.

- Create ECR Repository for Lambda Docker Image.

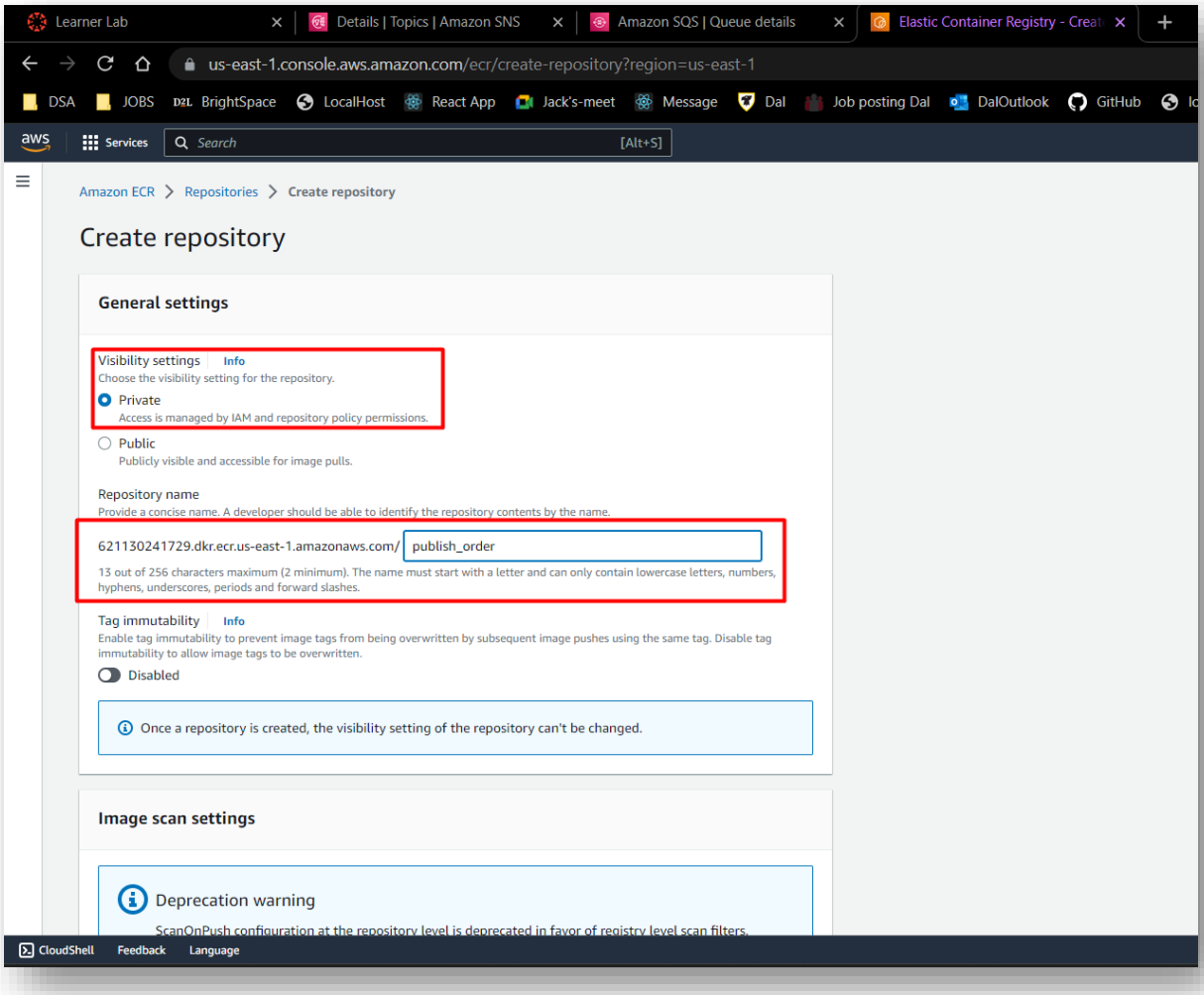

Figure 7 : Create ECR Repository for Lambda Docker Image.

- Created ECR Repository for Docker Image.

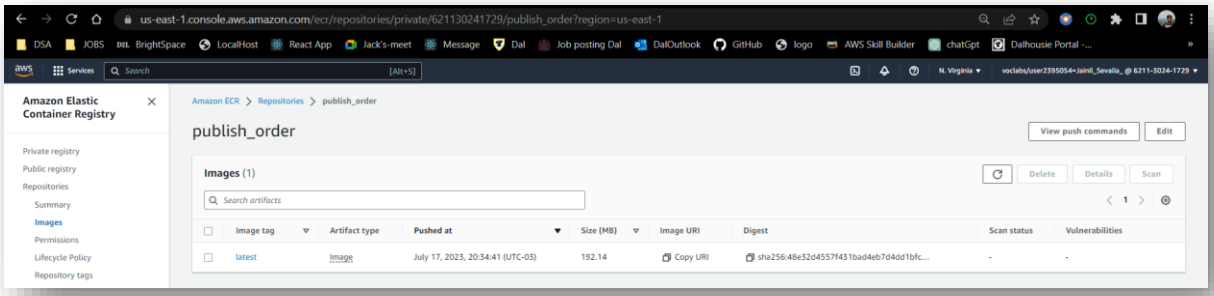

Figure 8 : Successfully created ECR Repository.

- Lambda function code is mentioned below, which will generate Taxi Order message and send it to SNS Topic(PublishOrder).

```python
import boto3
import random

# Instance created for AWS SNS service using boto3 package.
sns_client = boto3.client('sns')

# Static data Lists of car types and accessories and addresses for generating Taxi orders.
STATIC_CAR_TYPES = ['Compact', 'Mid-size Sedan', 'SUV', 'Luxury', 'Hatchback', 'Convertible', 'Coupe',
STATIC_CAR_ACCESSORIES = ['GPS', 'Camera', 'Bluetooth Hands-Free Kit', 'Seat Covers', 'Floor Mats', 'R
STATIC_CLIENT_ADDRESSES = ['6050 University Avenue', '123 Main Street', '456 Elm Street', '789 Oak Ave

# SNS Topic ARN for sending Taxi order messages.
SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:621130241729:PublishOrder'

def lambda_handler(event, context):
    # Randomly select car type, accessory, and client address using pyhton inbuilt finction random
    selected_car_type = random.choice(CAR_TYPES)
    selected_accessory = random.choice(CAR_ACCESSORIES)
    client_address = random.choice(CLIENT_ADDRESSES)

    # Create Taxi order message dictonary body
    order_details = {
        'car_type': selected_car_type,
        'accessory': selected_accessory,
        'address': client_address
    }

    # Send created Taxi order message to SNS topic
    response = sns_client.publish(
        TopicArn=SNS_TOPIC_ARN,
        Message=str(order_details)
    )

    return {
        'statusCode': 200,
        'body': 'Halifax Taxi Order sent successfully to SNS Topic'
    }
```

Figure 9 : PublishOrder lambda function Code.

- Docker image build, tagged and pushed to ECR newly created repository using custom deploy.sh script.

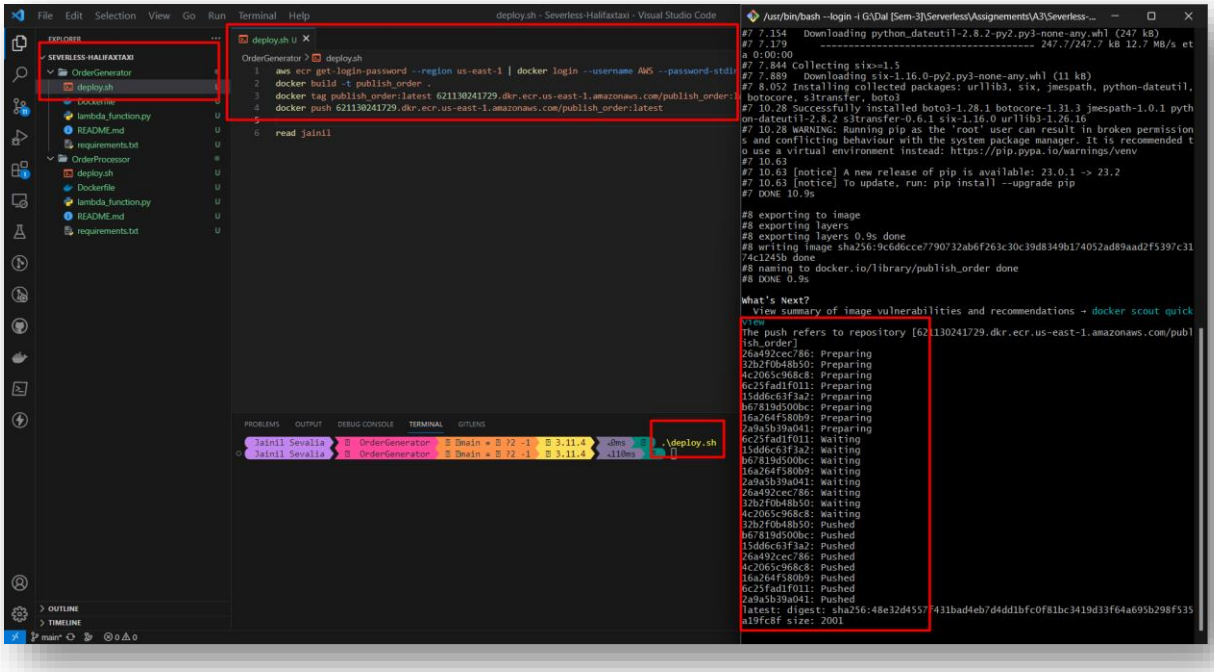

Figure 10 : Docker Image of Lambda code build, tagged and Pushed to ECR.

- Docker image is visible in the ECR repository in AWS Console.

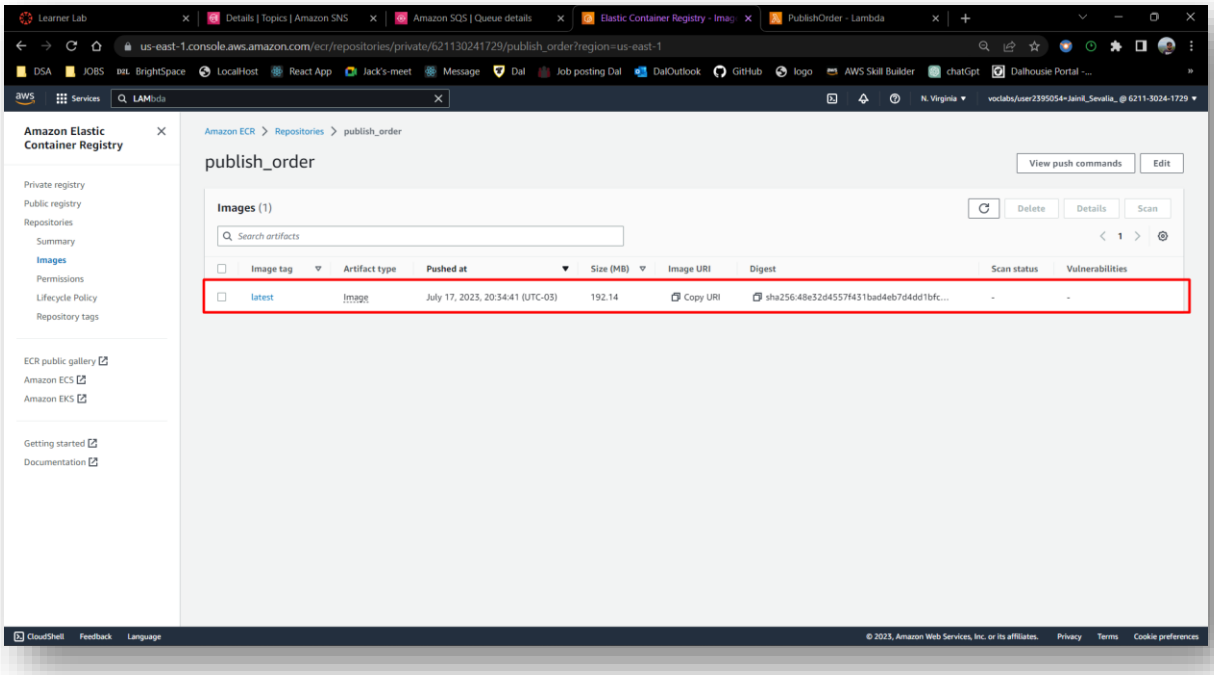

Figure 11 : ECR repository AWS console.

- Created AWS Lambda function using newly pushed docker image on ECR publish_order Repository.

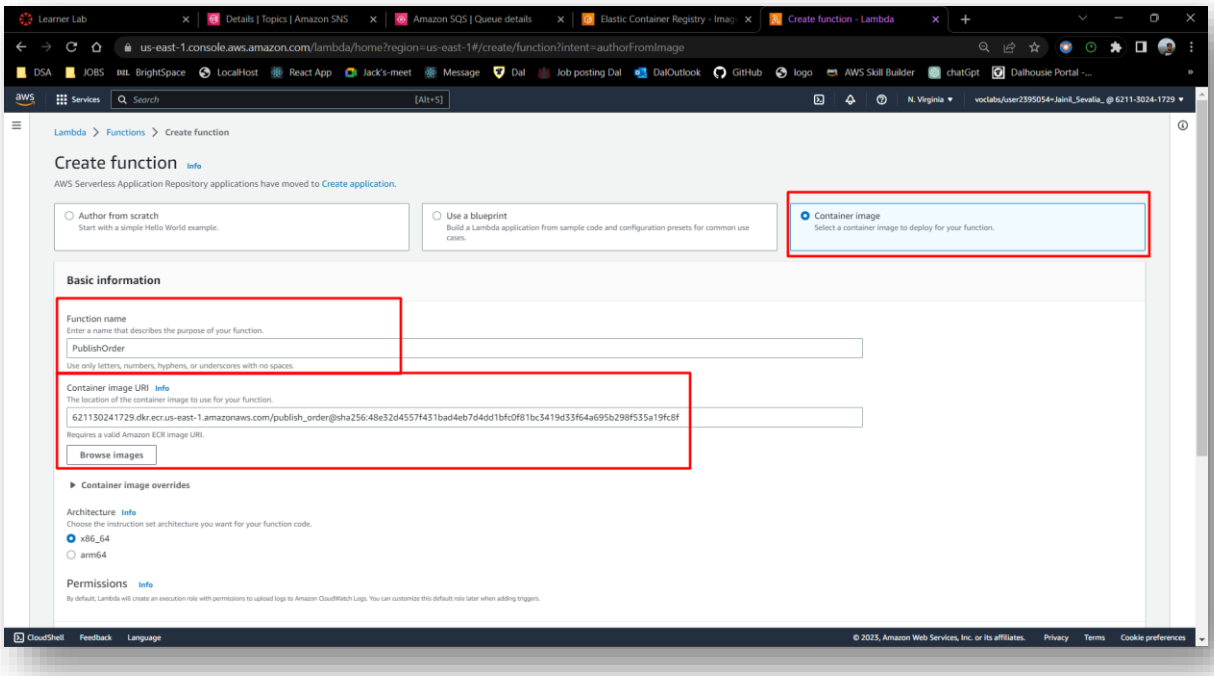

Figure 12 : Creating Lambda function using docker image pushed in ECR.

- Successfully created Lambda function.

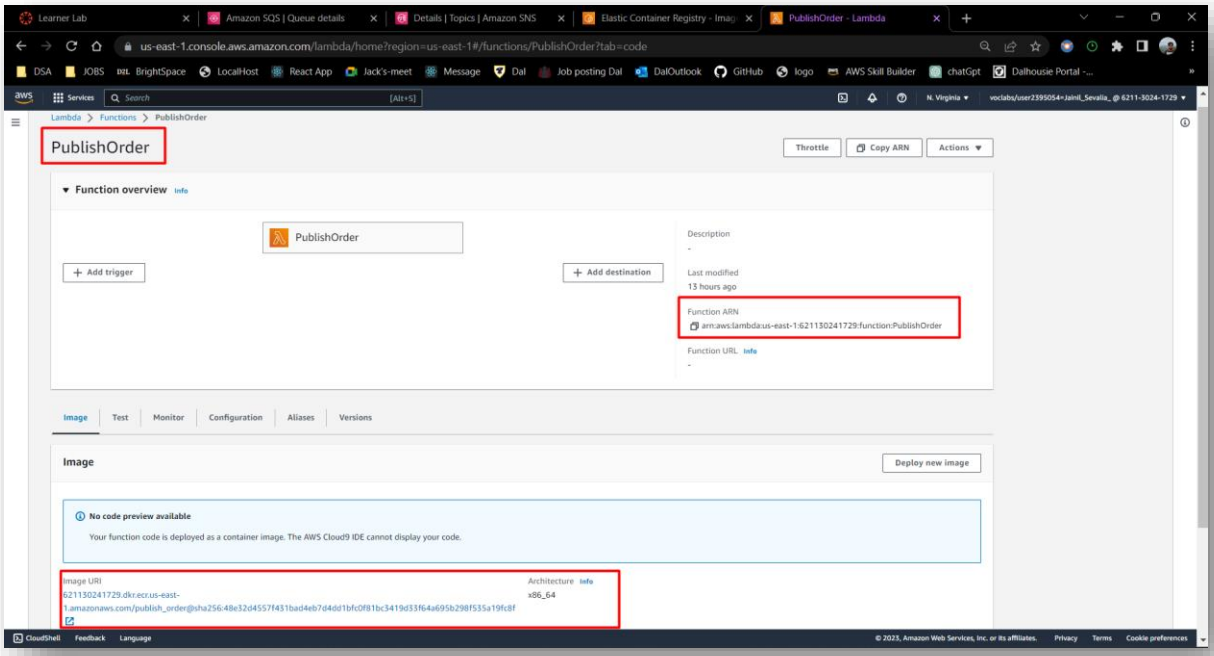

Figure 13 : Successfully created Lambda Function.

- Create a new SNS topic for sending mail. Second Lambda will poll message from the SQS and send it to this SNS topic.

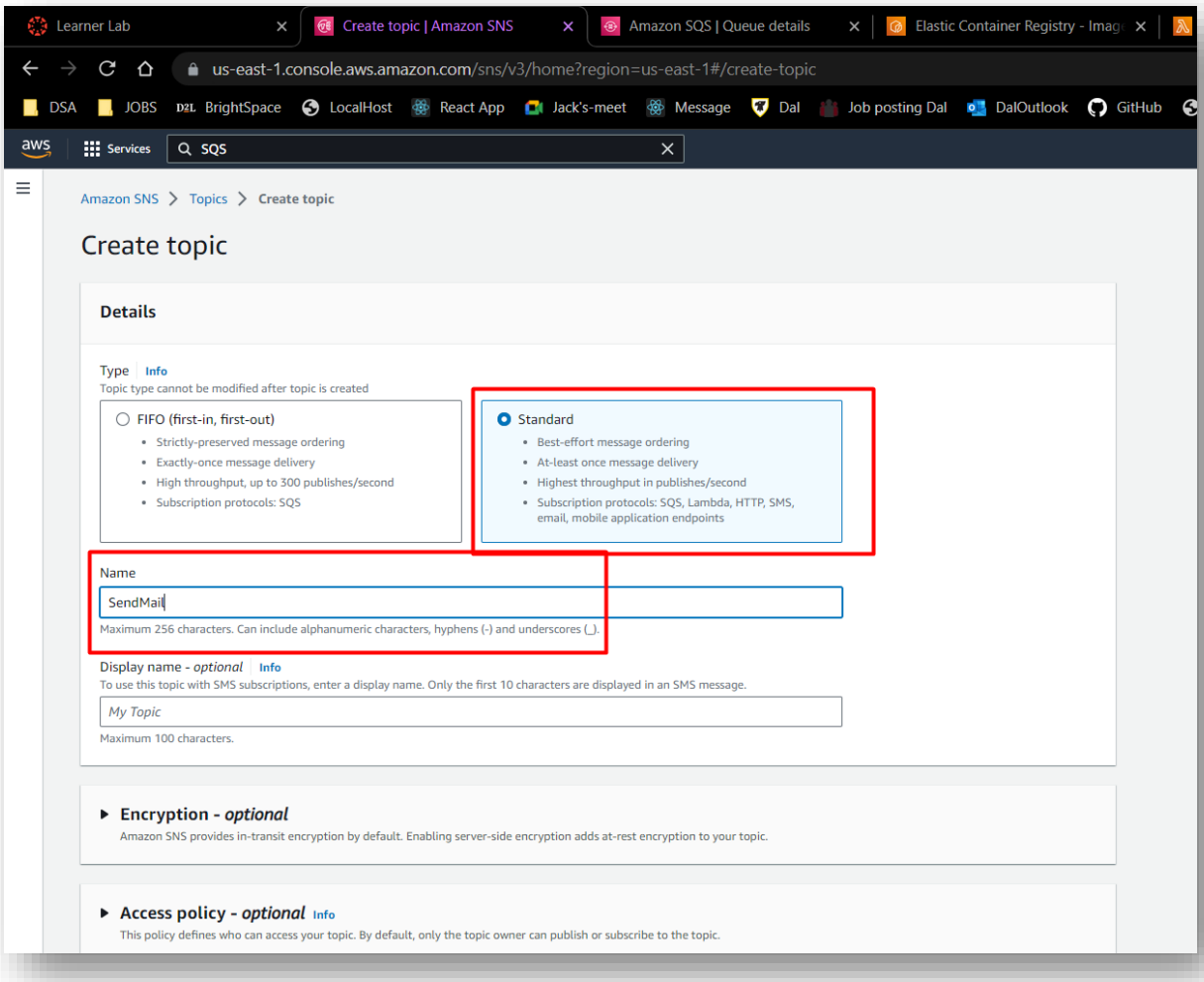

Figure 14 : Send Mail SNS topic.

- Successfully SNS topic created for sending mail for car delivery.

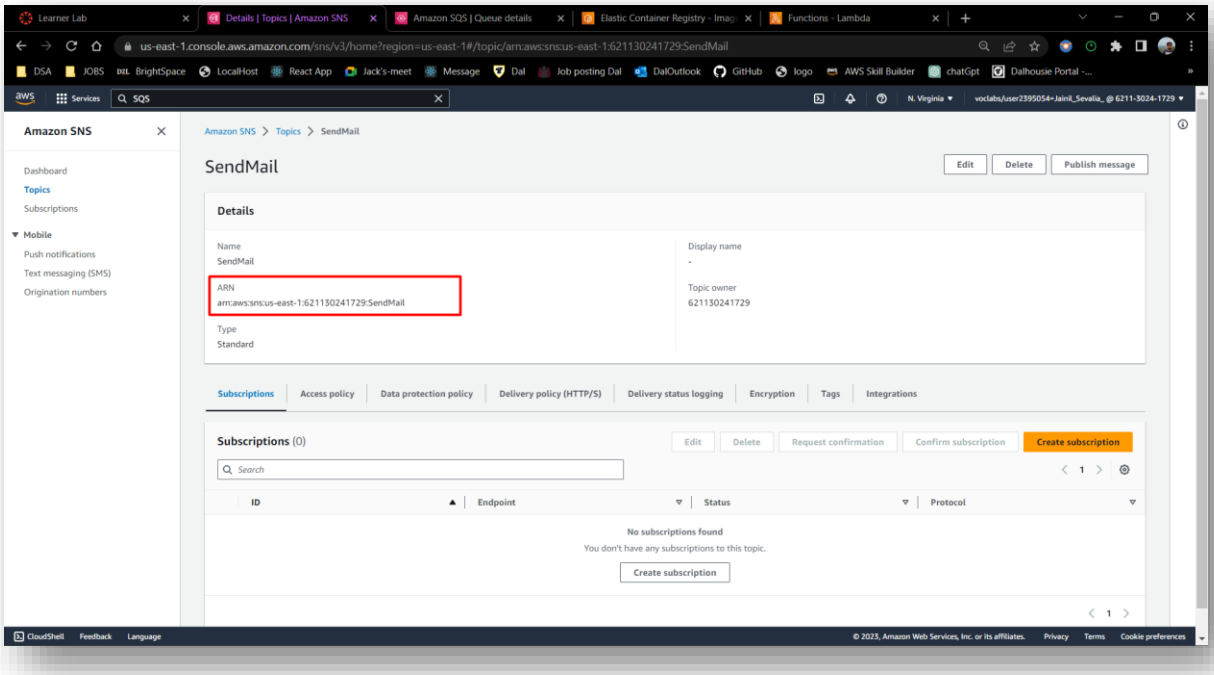

Figure 15 : Successfully created SNS topic. (AWS Console).

- Subscribe to email using SNS Subscription functionality.

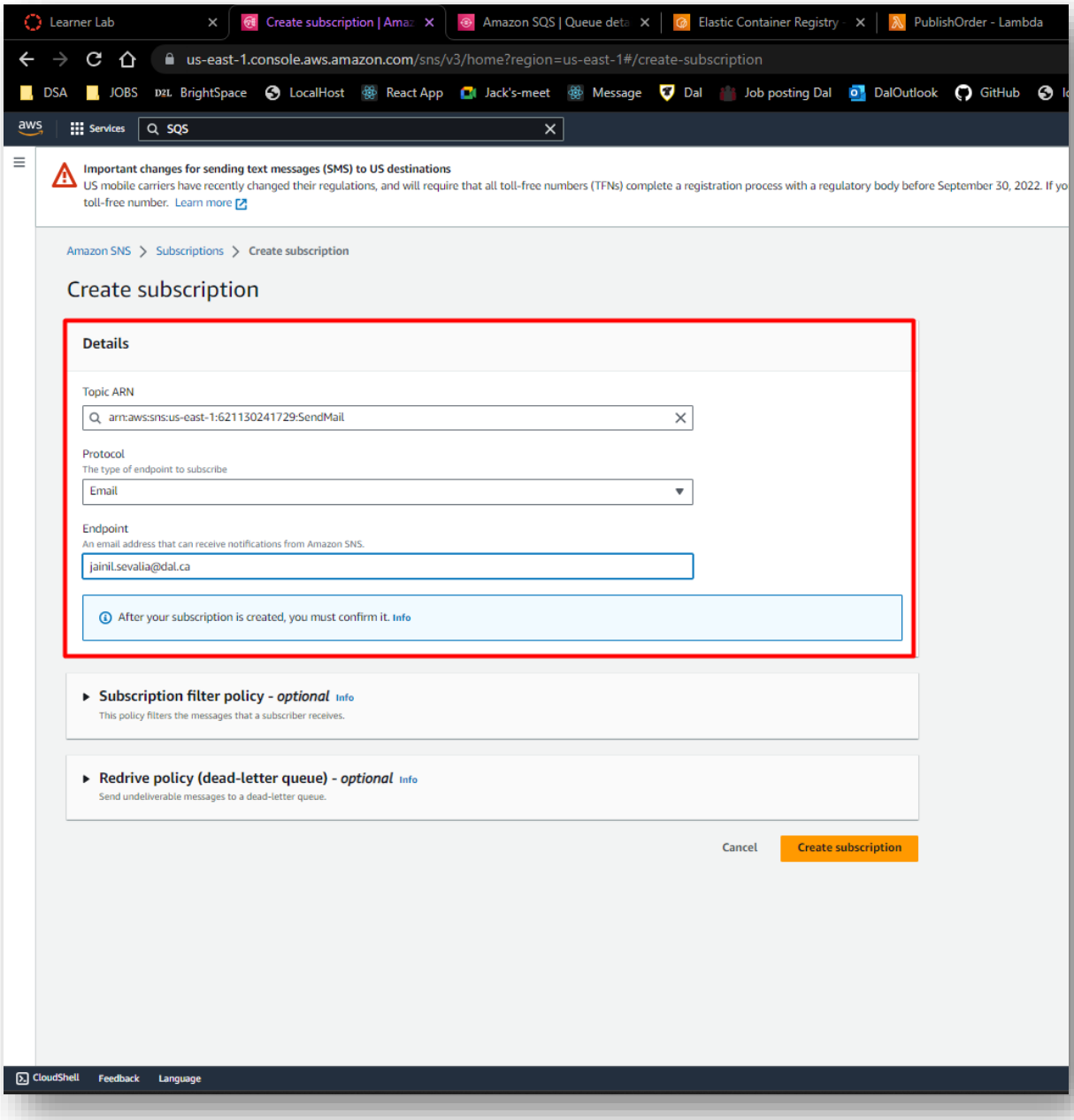

Figure 16 : Subscribe to email in SendMail SNS topic.

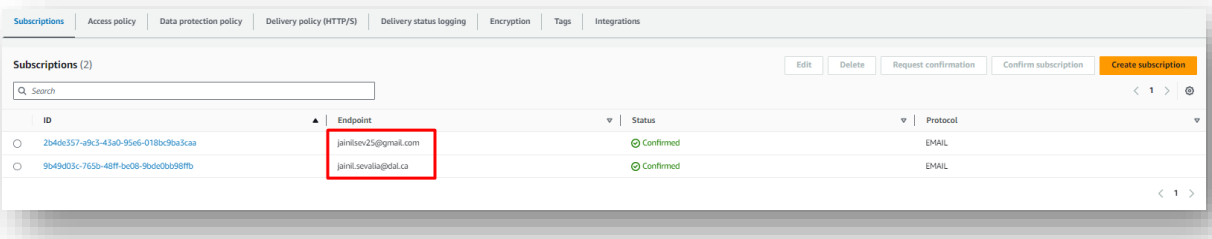

Figure 17 : Subscription list of SNS topic.

- Create ECR repository for second lambda function. This function will poll the message from SQS. This function will trigger every 2 min.

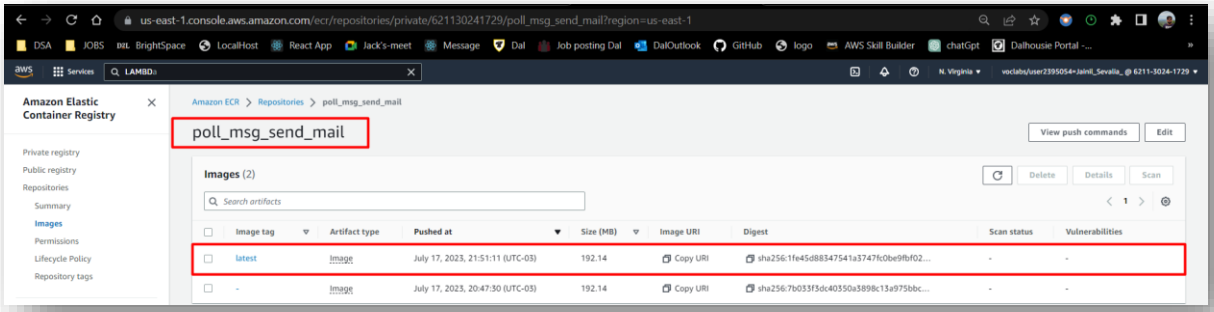

Figure 18 : ECR Repository for second Lambda Docker Image.

- Lambda function code is mentioned below, which will poll Taxi Order message from SQS(Every 2 min) and send it to SNS Topic(SendMail).

```python
import boto3
import json

# Instance created for AWS SNS and AWS SQS service using boto3 package.
sqs_client = boto3.client('sqs')
sns_client = boto3.client('sns')

# SQS Queue URL for polling Taxi orders messages.
SQS_QUEUE_URL = 'https://sqs.us-east-1.amazonaws.com/621130241729/HalifaxTaxiQueue'

# SNS Topic ARN for sending Taxi order messages to subscribed emails..
SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:621130241729:SendMail'

def lambda_handler(event, context):

    # polling a single message from the SQS queue.
    # This will eliminate use of for loop as I'm polling only a message at time.
    response = sqs_client.receive_message(
        QueueUrl=QUEUE_URL,
        MaxNumberOfMessages=1,
        WaitTimeSeconds=20
    )

    # Check if any message was received
    if 'Messages' in response:
        message = response['Messages'][0]

        # get only order taxi message from the SQS order message body.
        taxi_order_message = json.loads(message['Body'])['Message']

        # Send the order message to SNS for email notification
        sns_client.publish(
            TopicArn=TOPIC_ARN,
            Message=taxi_order_message,
            Subject='New Car Delivery Order'
        )

        # Delete the message from the SQS queue after processing
        sqs_client.delete_message(
            QueueUrl=QUEUE_URL,
            ReceiptHandle=message['ReceiptHandle']
        )

        print('Order sent to SNS for email notification:', taxi_order_message)

    return {
        'statusCode': 200,
        'body': 'SQS order taxi order message sent SNS successfully for email notifications.'
    }
```

Figure 19 : Code for (poll_msg_send_mail) lambda function.

- Docker image for second lambda build, tagged and pushed to ECR newly created repository using custom deploy.sh script.

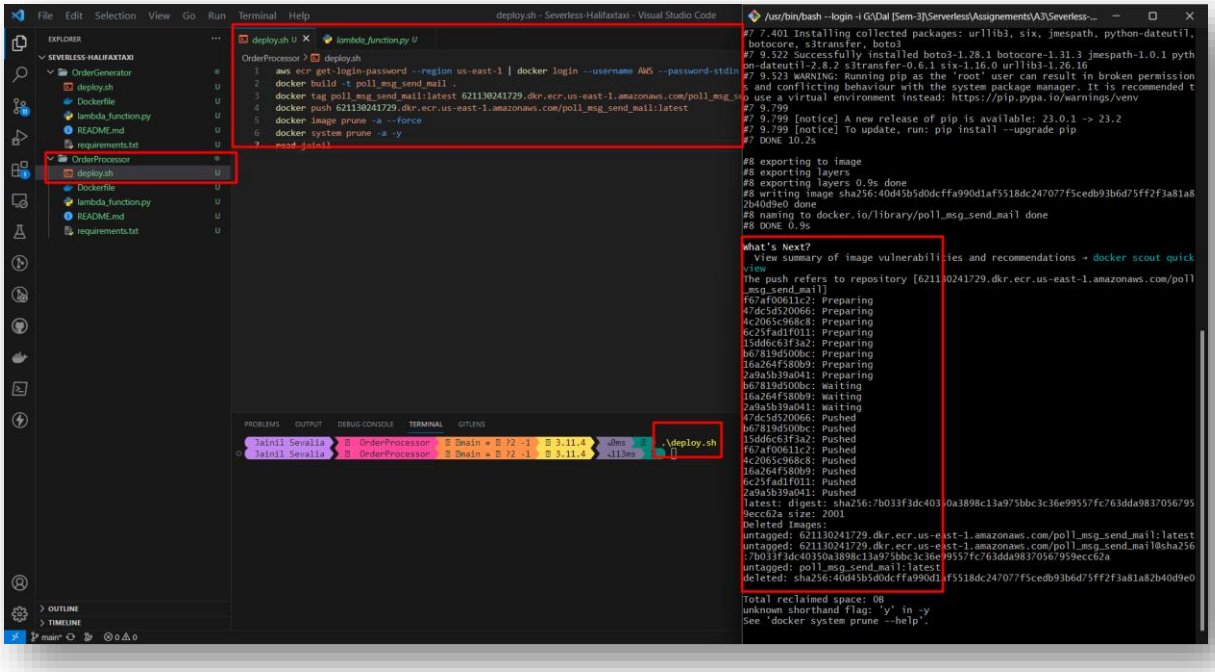

Figure 20 : Docker image for second Lambda is build, tagged, and pushed to ECR repository.

- ECR Repository AWS console, second lambda Docker image is successfully pushed.

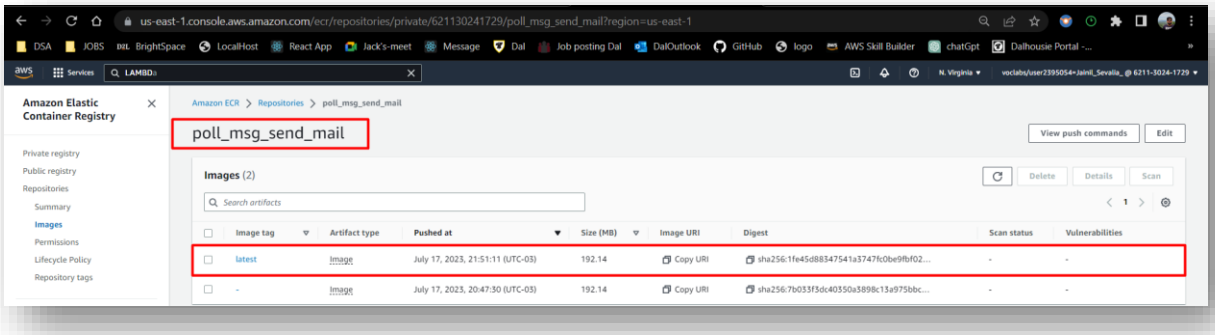

Figure 21 : Successfully Second lambda docker image is pushed. (AWS console).

- Create second lambda function which will poll the message from SQS queue and pass it to SNS Topic(SendMail).

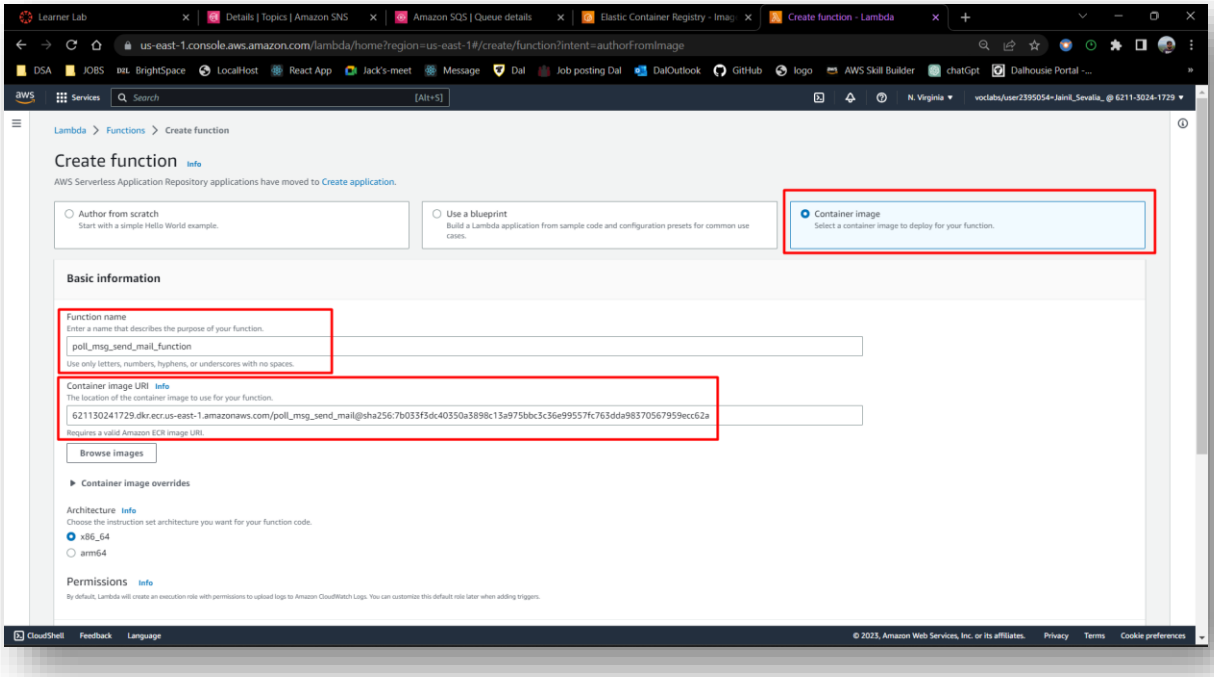

Figure 22 : Creating Second Lambda function poll_msg_send_mail. (AWS console).

- Configure Event Bridge which will invoke this lambda function every 2 min to check the SQS Queue for message. Expression cron(*/2 * * * ? *) will invoke lambda every 2 min.

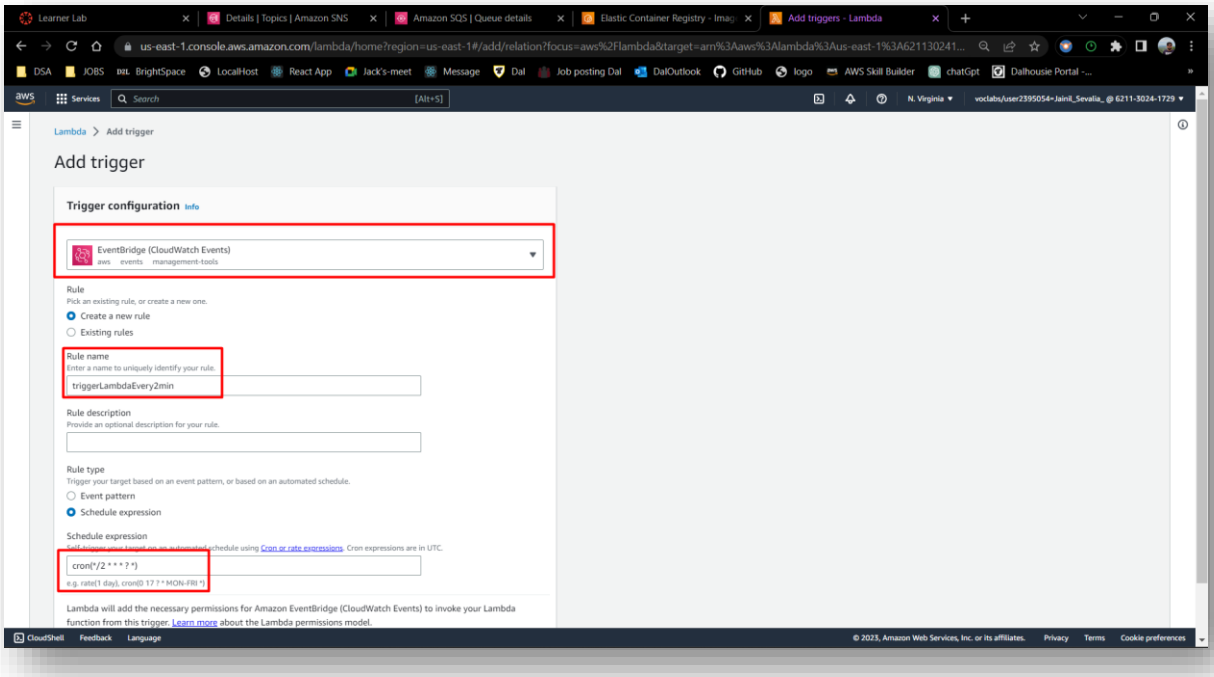

Figure 23 : Configure AWS Event bridge for invoking lambda every 2 min.

- Final Architecture for second lambda(poll_msg_send_mail) is mentioned below.

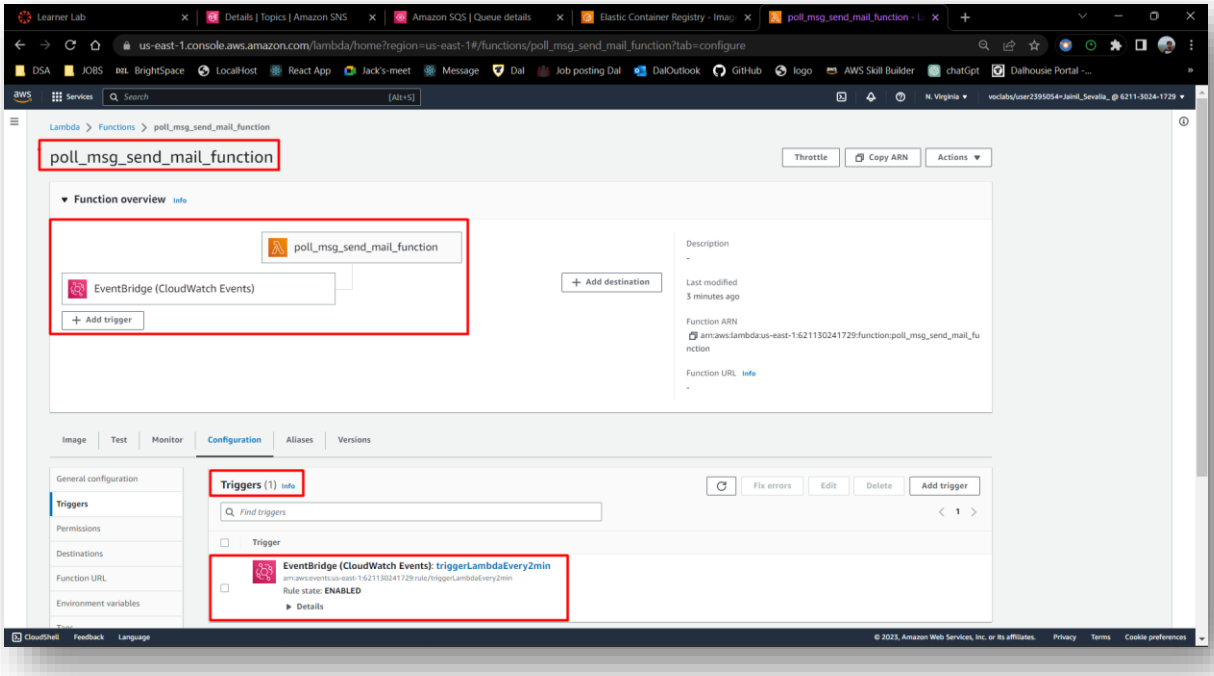

Figure 24 : Second lambda(poll_msg_send_mail) AWS console.

- Everything is set up as mentioned in the assignment guidelines. It's time for testing this HalifaxTaxi cloud serverless architecture.
- Let's hit the or trigger manually first lambda function(PublishOrder). This function will randomly generate order for taxi. This order will send to SNS topic(PublishOrder) and lead to SQS(HalifaxTaxiQueue) as that queue is subscribed to SNS topic.

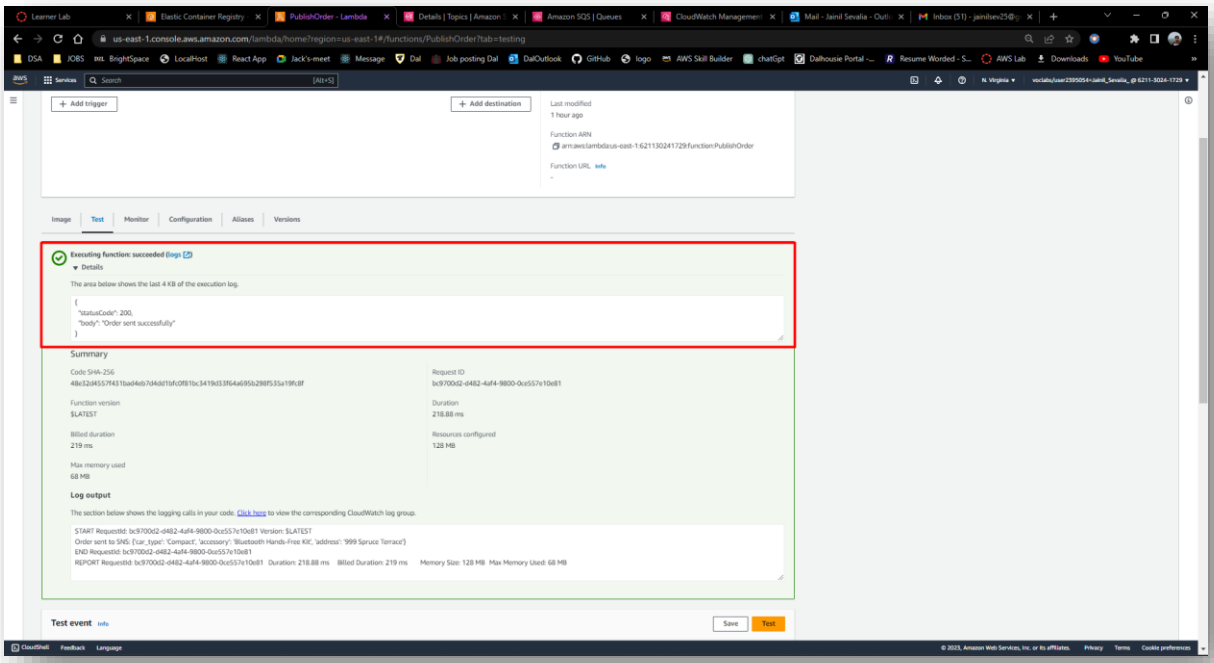

Figure 25: Testing - Triggering first Lambda function to generate order and send to SNS to SQS.

- At this point SQS(halifaxTaxiQueue) will receive random order message in the queue.
- After every 2min, Second Lambda(poll_msg_send_mail) will trigger. This function will poll message form SQS(HalifaxTaxiQueue) and Send that Message to SNS topic(SendMail). As 2 emails are added as subscriber, that emails will receive taxi order details.
- Here, I'm adding cloud watch log image and received email images as a part of testing.

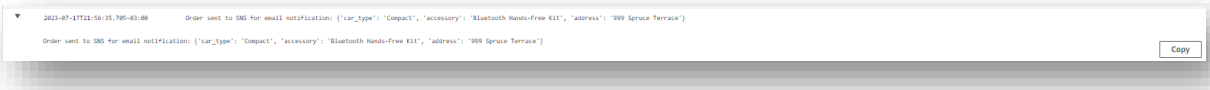

Figure 26: Message that sent to SNS topic. (Cloud watch AWS console).

- Mail got into my account(Subscribed to SNS topic).

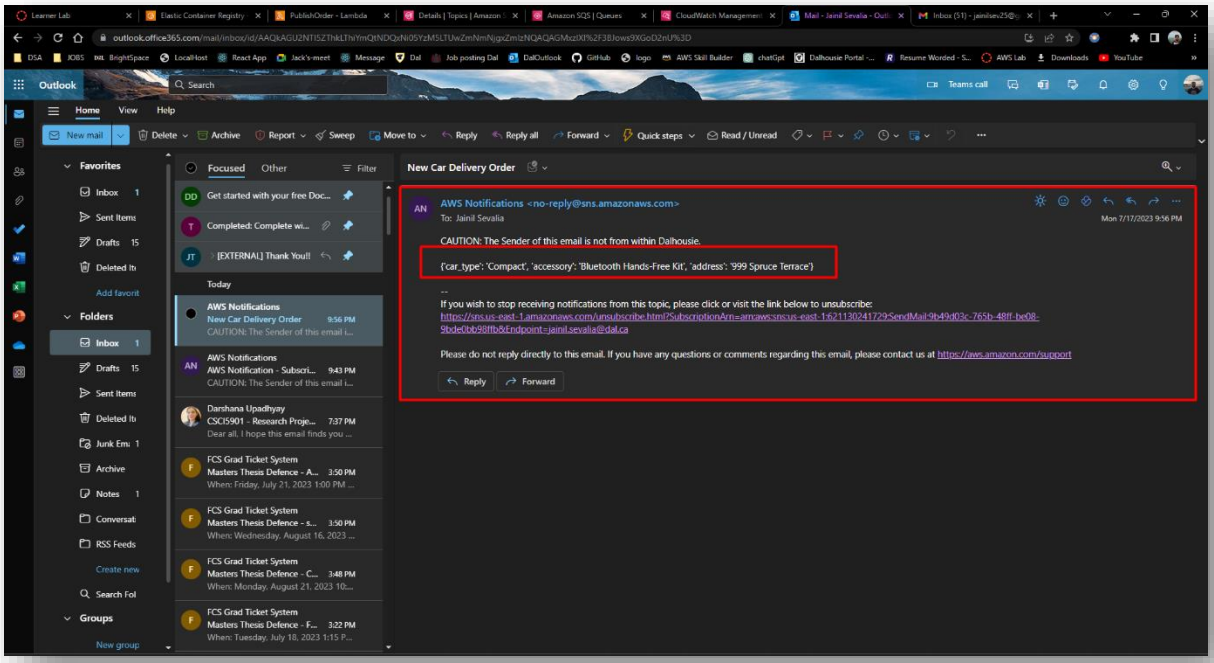

Figure 27 : Got email of with Taxi order Information.

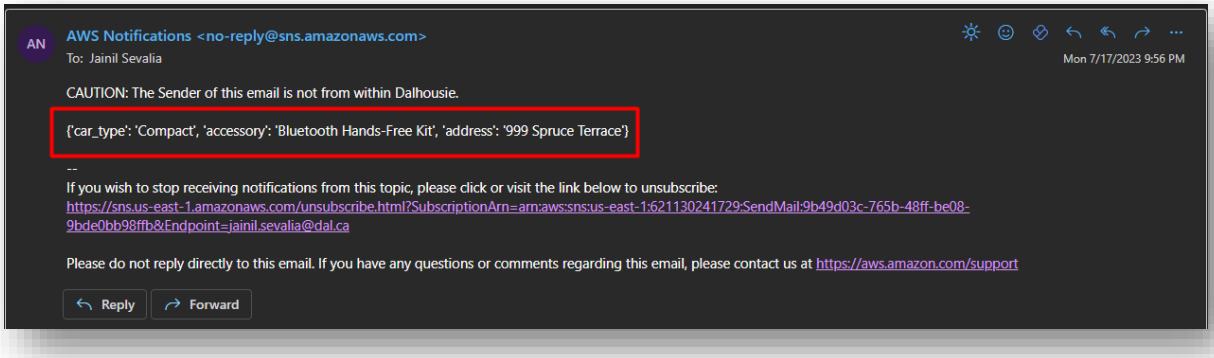

Figure 28 : Taxi Order Information got in mail.