Use AWS Lambda-SQS-SNS for online Car delivery service.

Step 1: Login to AWS Creds and open SNS to create a Topic named Car_Order and select Standard option and click on create as shown in Fig 100, 101. And similarly, we will create another Topic named order_generation as shown in Fig 102.

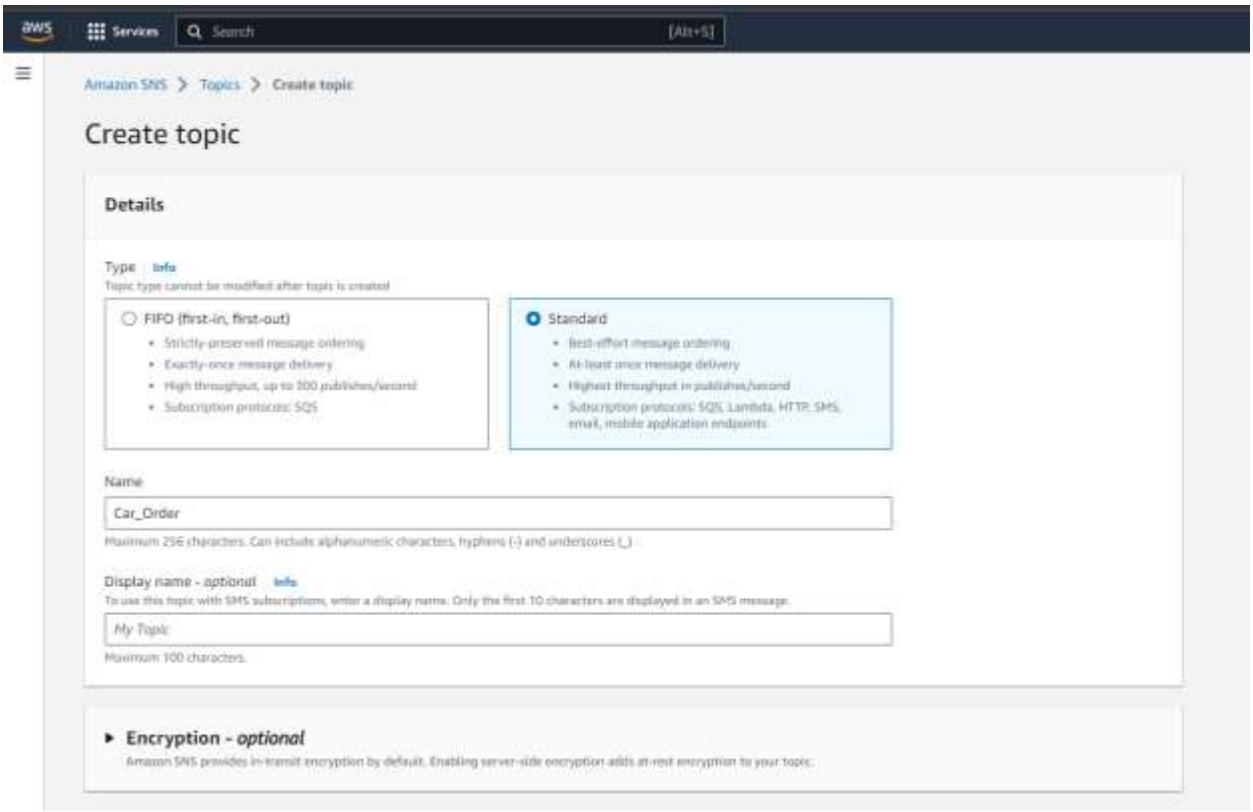

Fig 100

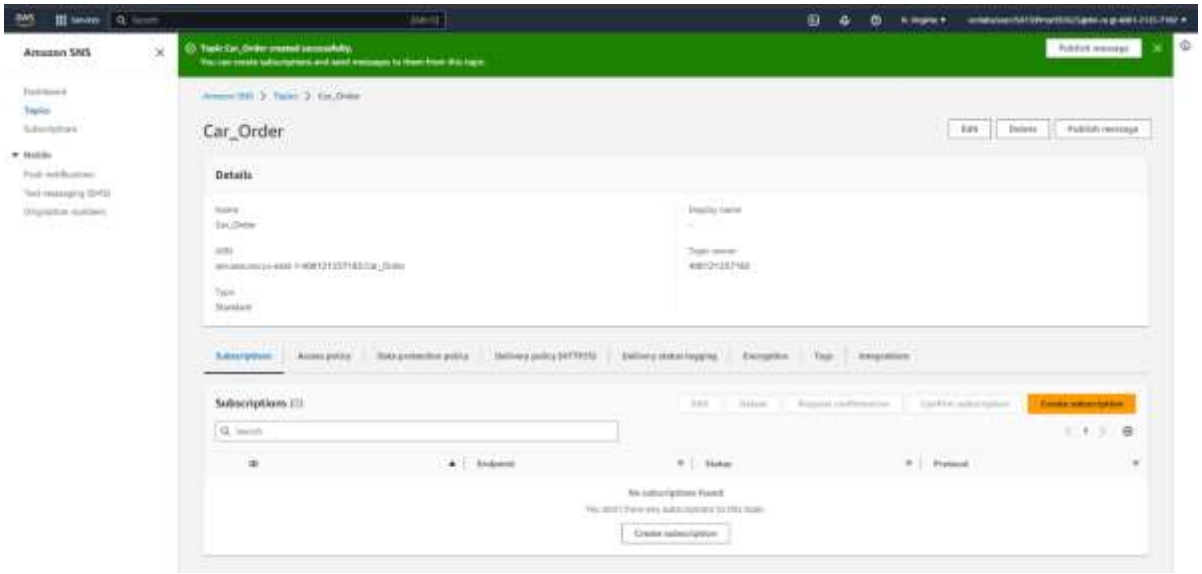

Fig 101

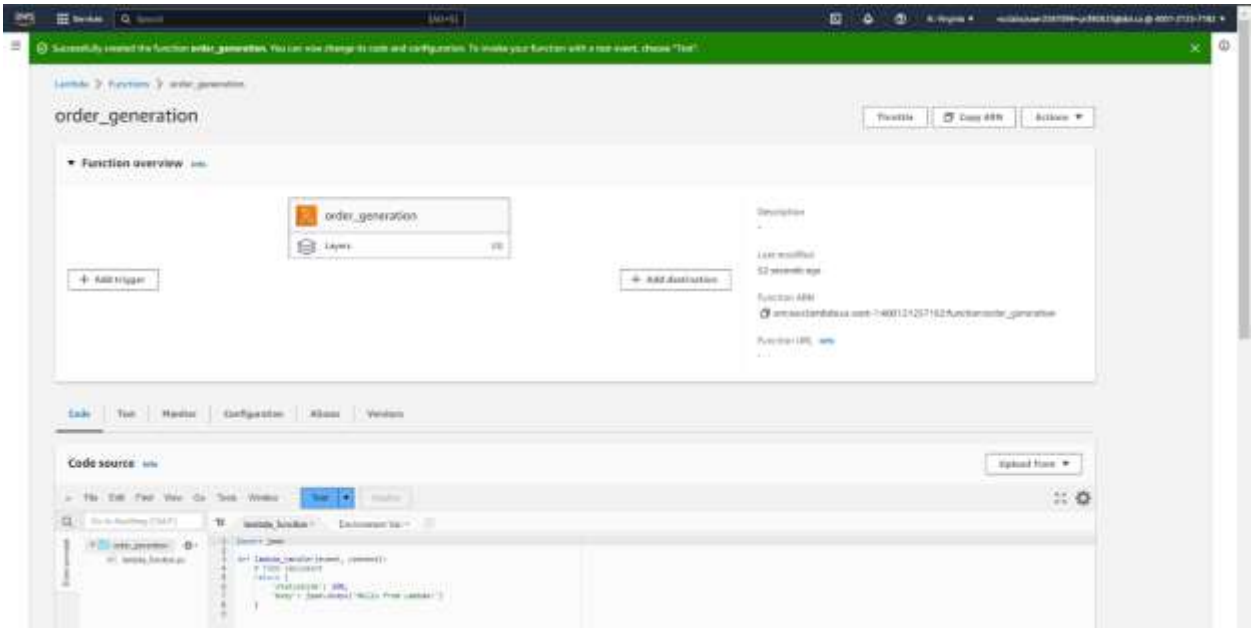

Fig 102

Step 2: Now we will go to SQS and create a Queue named Car_type, Car_Accessories and Client_address with standard option as shown in Fig 103, 104, 105, 106, 107, 108, 109.

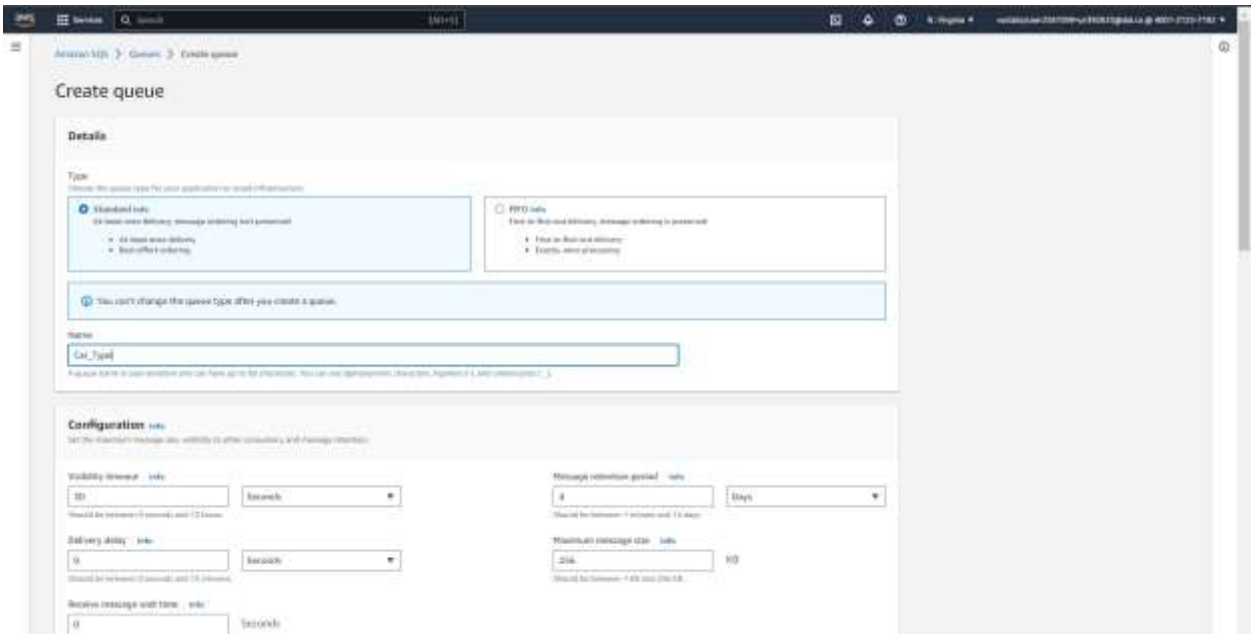

Fig 103

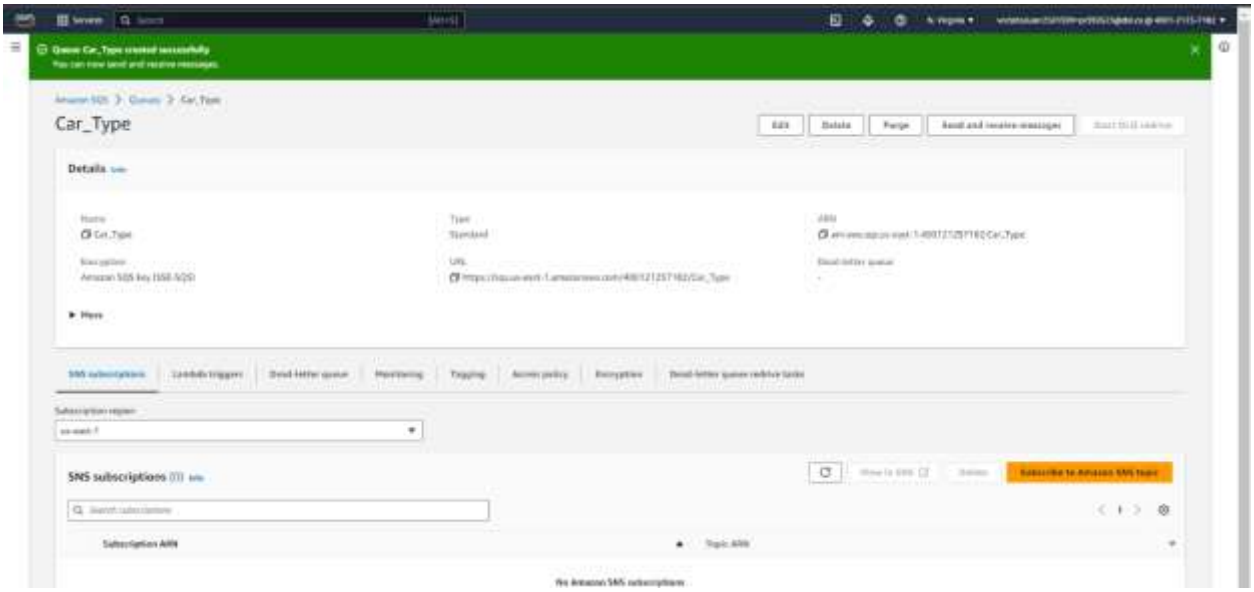

Fig 104

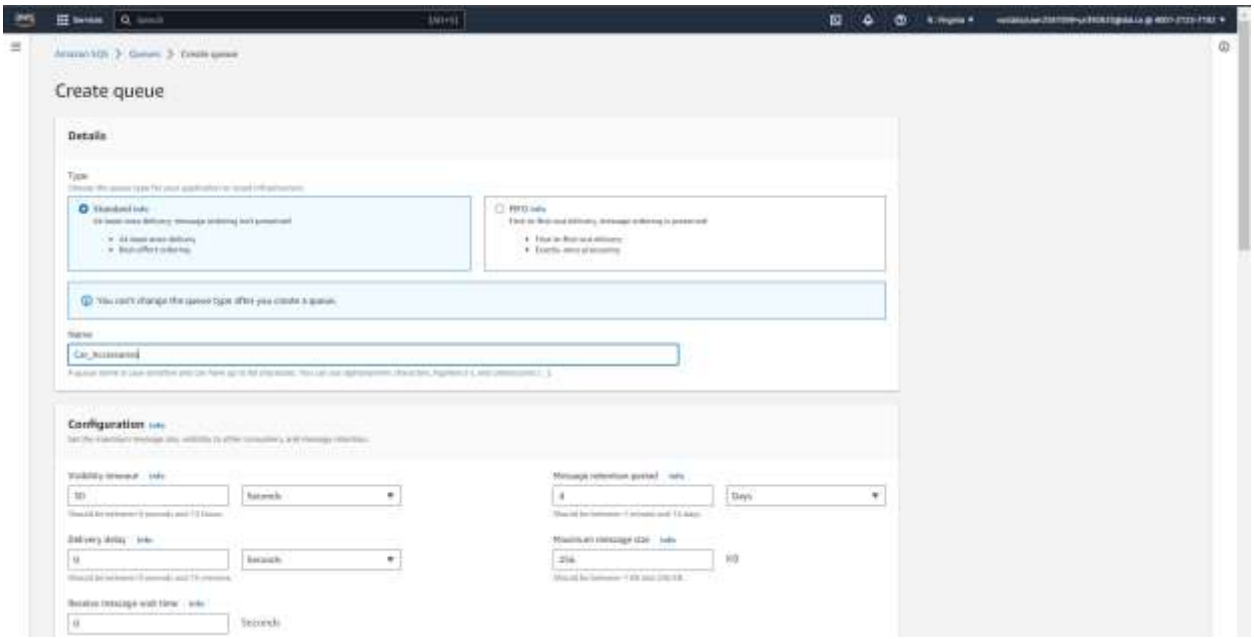

Fig 105

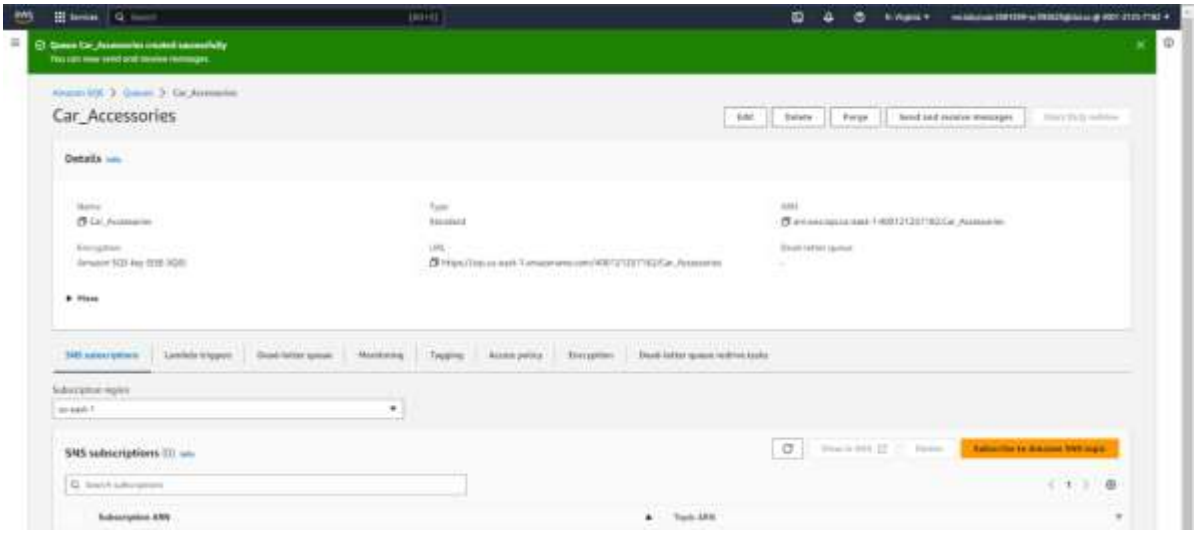

Fig 106

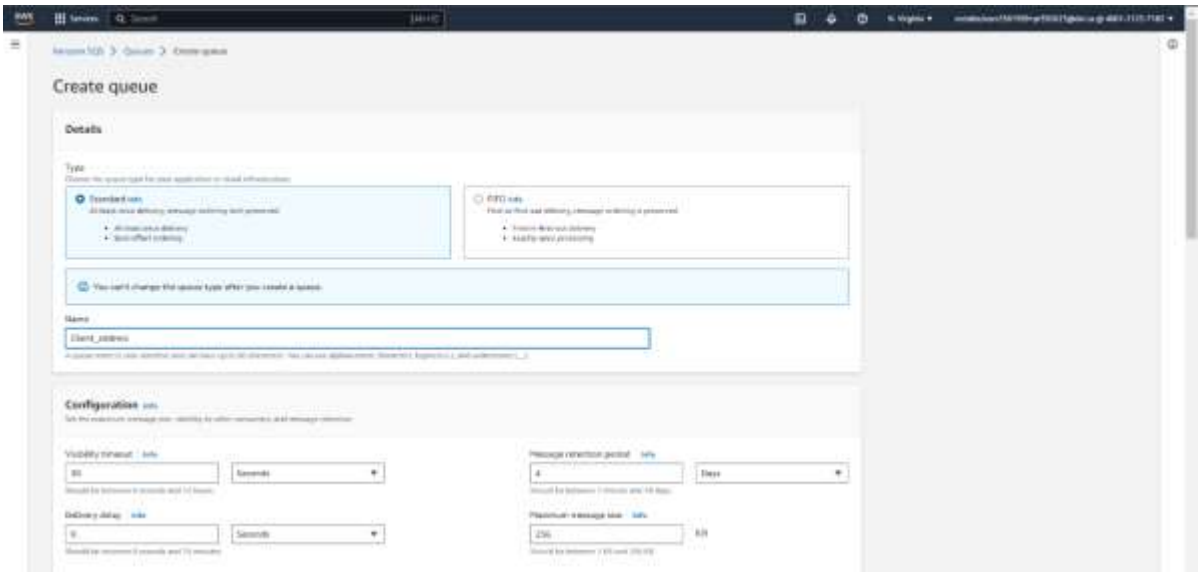

Fig 107

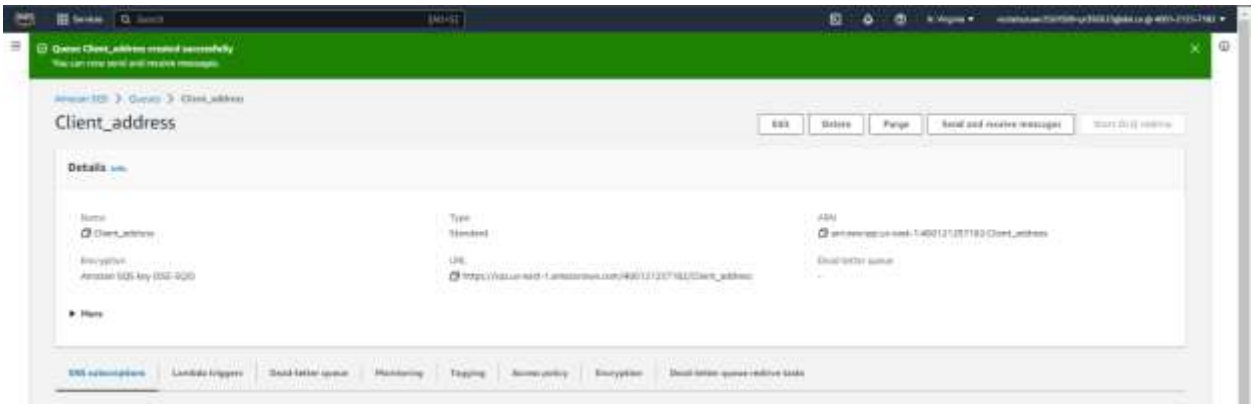

Fig 108

Fig 109

Step 3: Now we will go to Lambda functions and create functions named Car_Updates and create a Trigger with cloud watch events as shown in Fig 110, 111, 112, 113. Now we will add code to this lambda function as shown in Fig 114.

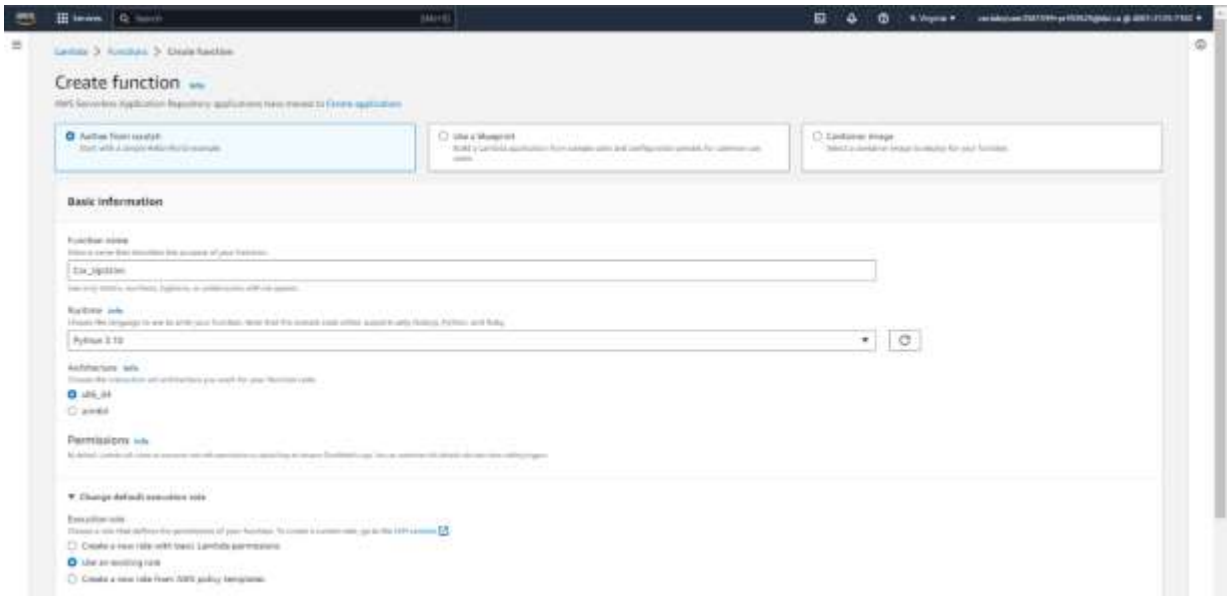

Fig 110

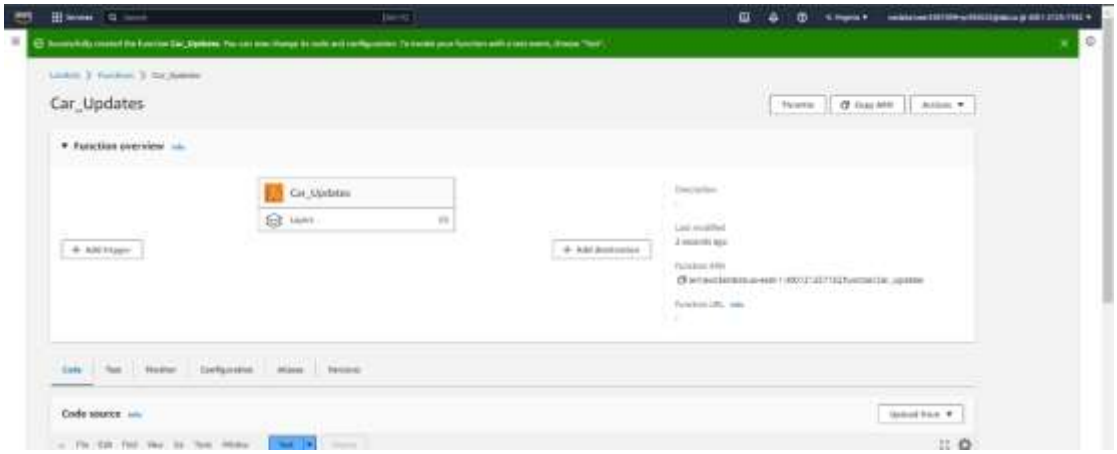

Fig 111

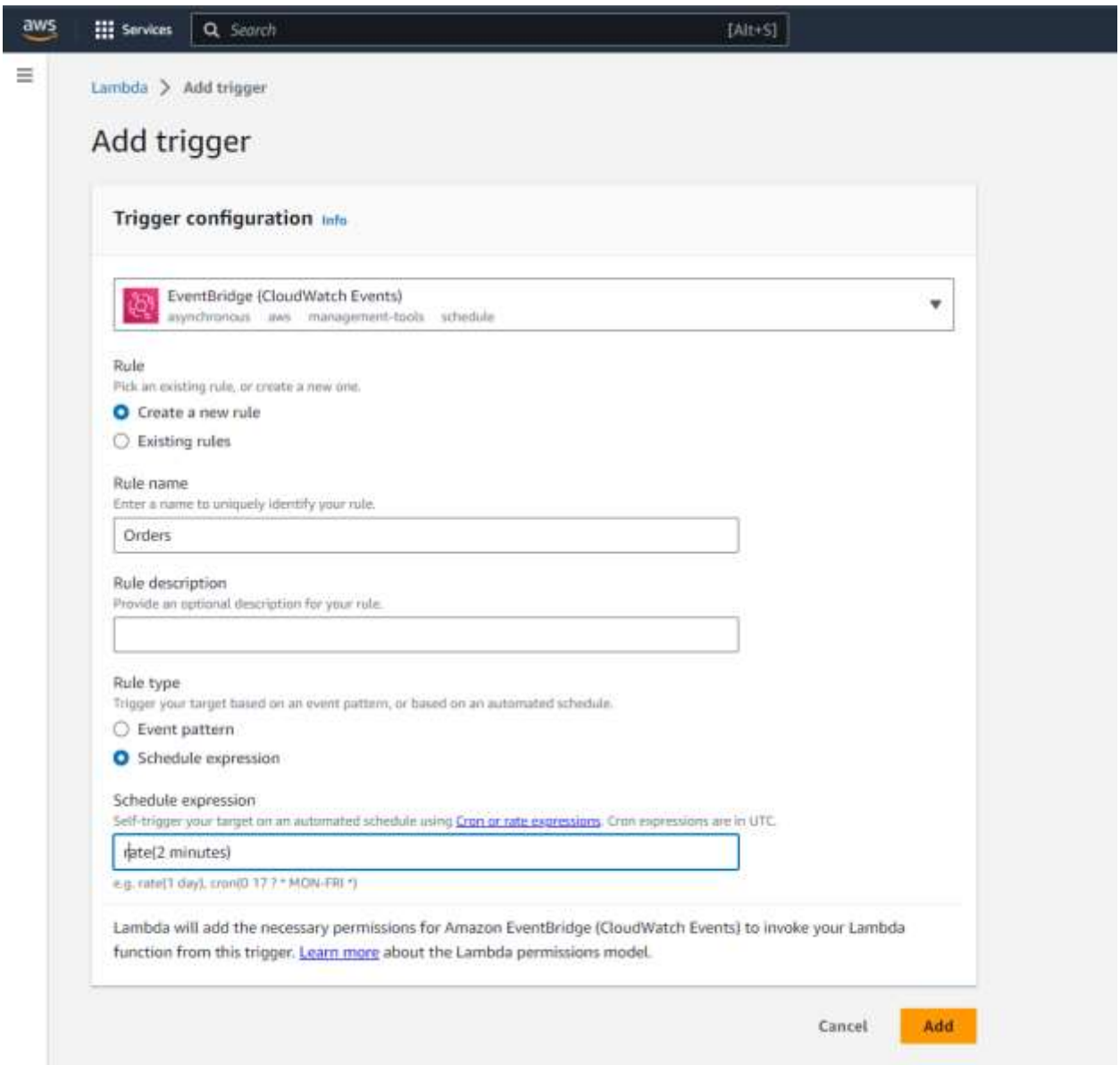
aws
Services
Search
[Alt+S]
Lambda > Add trigger
Add trigger
Trigger configuration Info
EventBridge (CloudWatch Events)
asynchronous aws management-tools schedule
Rule
Pick an existing rule, or create a new one.
Create a new rule
Existing rules
Rule name
Enter a name to uniquely identify your rule.
Orders
Rule description
Provide an optional description for your rule.
Rule type
Trigger your target based on an event pattern, or based on an automated schedule.
Event pattern
Schedule expression
Schedule expression
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.
rate(2 minutes)
e.g. rate(1 day), cron(0 17 ? * MON-FRI *)
Lambda will add the necessary permissions for Amazon EventBridge (CloudWatch Events) to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.
Cancel
Add

Fig 112

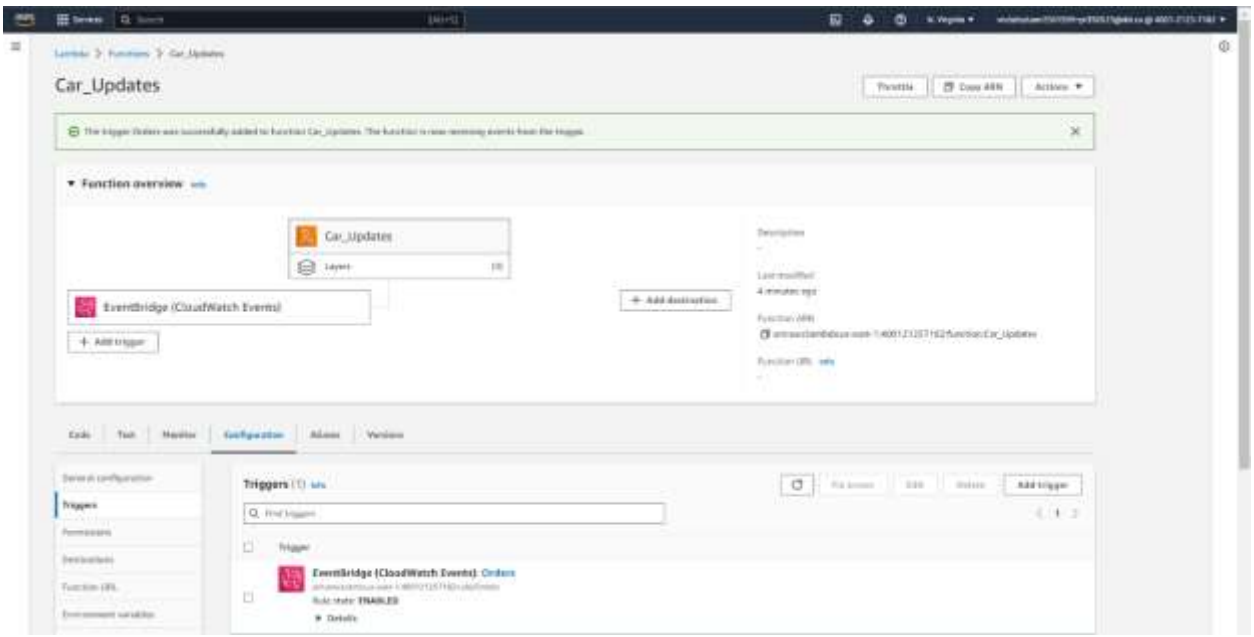

Fig 113

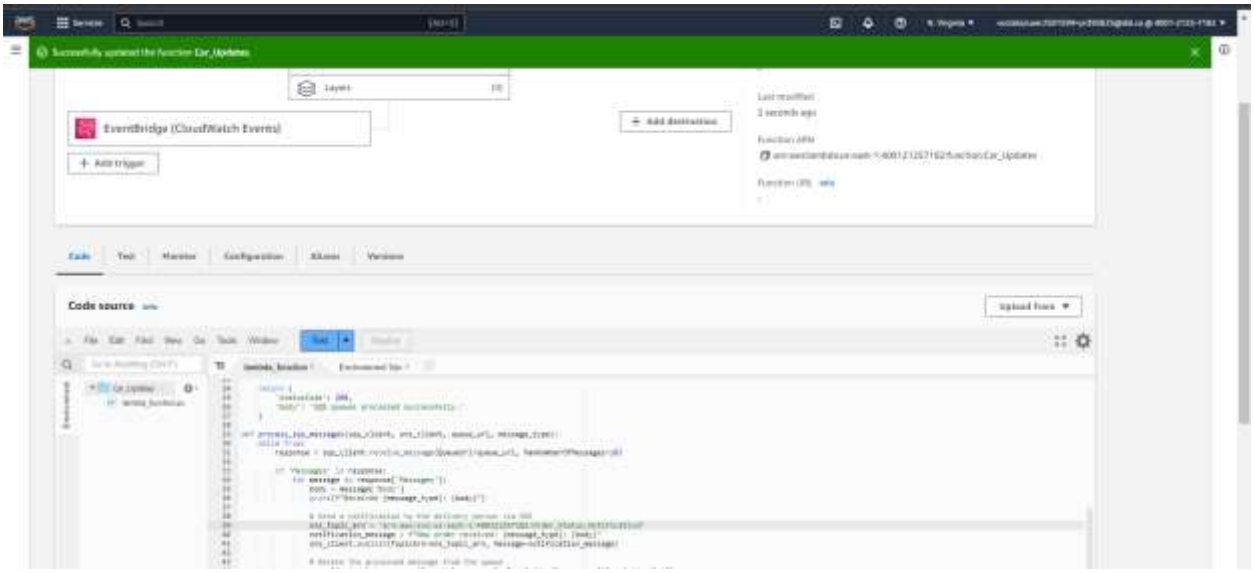

Fig 114

Code added:

```
import boto3

def lambda_handler(event, context):
    # Initialize SQS client
    sqs_client = boto3.client('sqs')

    # Initialize SNS client
    sns_client = boto3.client('sns')
```

```
    # SQS queue URLs
    car_type_queue_url = 'https://sqs.us-east-1.amazonaws.com/400121257182/Car_Type'
    accessory_queue_url = 'https://sqs.us-east-
1.amazonaws.com/400121257182/Car_Accessories'
    address_queue_url = 'https://sqs.us-east-1.amazonaws.com/400121257182/Client_address'

    # Process car types
    process_sqs_messages(sqs_client, sns_client, car_type_queue_url, "Car Type")

    # Process accessories
    process_sqs_messages(sqs_client, sns_client, accessory_queue_url, "Accessory")

    # Process delivery addresses
    process_sqs_messages(sqs_client, sns_client, address_queue_url, "Delivery Address")

    return {
        'statusCode': 200,
        'body': 'SQS queues processed successfully.'
    }

def process_sqs_messages(sqs_client, sns_client, queue_url, message_type):
    while True:
        response = sqs_client.receive_message(QueueUrl=queue_url,
MaxNumberOfMessages=10)

        if 'Messages' in response:
            for message in response['Messages']:
                body = message['Body']
                print(f"Received {message_type}: {body}")

                # Send a notification to the delivery person via SNS
                sns_topic_arn = 'arn:aws:sns:us-east-1:400121257182:Order_Status_Notification'
                notification_message = f"New order received! {message_type}: {body}"
                sns_client.publish(TopicArn=sns_topic_arn, Message=notification_message)

                # Delete the processed message from the queue
                sqs_client.delete_message(QueueUrl=queue_url,
ReceiptHandle=message['ReceiptHandle'])
        else:
            break
```

Step 4: Similarly, we will create another Lambda function named order_generation and with this we will add trigger to SNS Car_order that we created earlier. Also, we will add the code too in the process as shown in Fig 115, 116, 117.

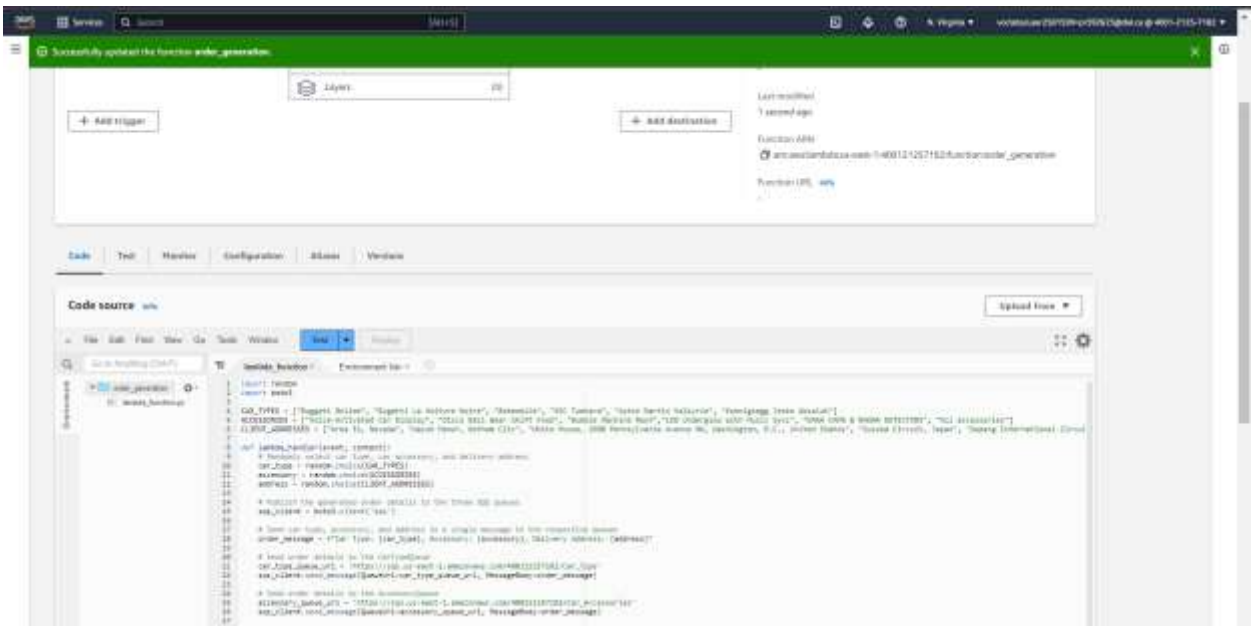

Fig 115

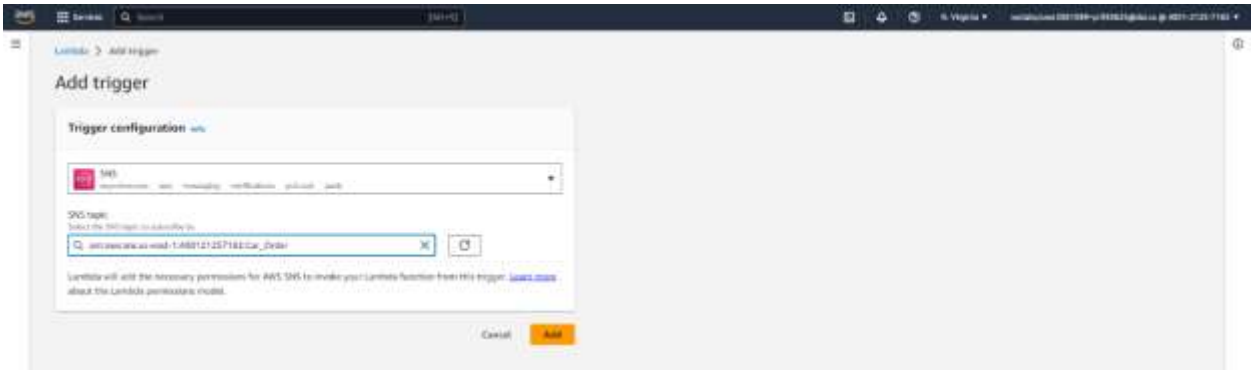

Fig 116

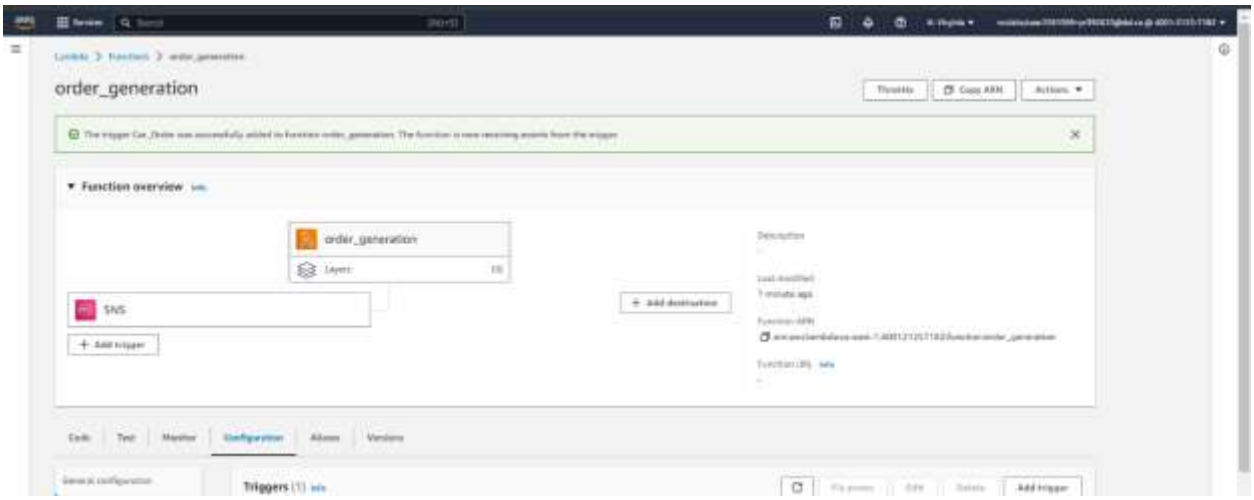

Fig 117

Code added:

```
# Import necessary libraries
import random
import boto3

# Define lists of car types, accessories, and client addresses
CAR_TYPES = ["Buggati Bolide", "Bugatti La Voiture Noire", "Batmobile", "SSC Tuatara",
"Aston Martin Valkyrie", "Koenigsegg Jesko Absolut"]
ACCESSORIES = ["Voice-Activated Car Display", "Disco Ball Gear Shift Knob", "Bubble
Machine Roof", "LED Underglow with Music Sync", "DASH CAMS & RADAR
DETECTORS", "All accessories"]
CLIENT_ADDRESSES = ["Area 51, Nevada", "Wayne Manor, Gotham City", "White House,
1600 Pennsylvania Avenue NW, Washington, D.C., United States", "Suzuka Circuit,
Japan", "Sepang International Circuit, Malaysia", "Circuit Gilles Villeneuve, Canada"]

def lambda_handler(event, context):
    # Randomly select car type, car accessory, and delivery address
    car_type = random.choice(CAR_TYPES)
    accessory = random.choice(ACCESSORIES)
    address = random.choice(CLIENT_ADDRESSES)

    # Publish the generated order details to the three SQS queues
    sqs_client = boto3.client('sqs')

    # Construct the order message with car type, accessory, and delivery address
    order_message = f"Car Type: {car_type}, Accessory: {accessory}, Delivery Address:
{address}"

    # Send order details to the CarTypeQueue
    car_type_queue_url = 'https://sqs.us-east-
1.amazonaws.com/400121257182/Car_Type'
    sqs_client.send_message(QueueUrl=car_type_queue_url,
MessageBody=order_message)

    # Send order details to the AccessoryQueue
    accessory_queue_url = 'https://sqs.us-east-
1.amazonaws.com/400121257182/Car_Accessories'
    sqs_client.send_message(QueueUrl=accessory_queue_url,
MessageBody=order_message)

    # Send order details to the AddressQueue
```

```
    address_queue_url = 'https://sqs.us-east-
1.amazonaws.com/400121257182/Client_address'
    sqs_client.send_message(QueueUrl=address_queue_url,
MessageBody=order_message)

    # Publish the generated order details to the SNS topic (optional)
    sns_client = boto3.client('sns')
    sns_topic_arn = 'arn:aws:sns:us-east-1:400121257182:Car_Order'
    sns_client.publish(TopicArn=sns_topic_arn, Message=order_message)

    return {
        'statusCode': 200,
        'body': 'Order generated and sent to SQS queues.'
    }
```

Step 5: Now we will create another SNS topic named Order_Status_Notification with a display name of Halifax Ultimate Car Services as shown in Fig 118, 119. Now we will click on create Subscription in which we will add an endpoint which is the email of client as shown in 120. This will also send a confirmation email to the client and after we click on confirm we will be subscribed to receive messages from the aws as shown in Fig 121, 122.

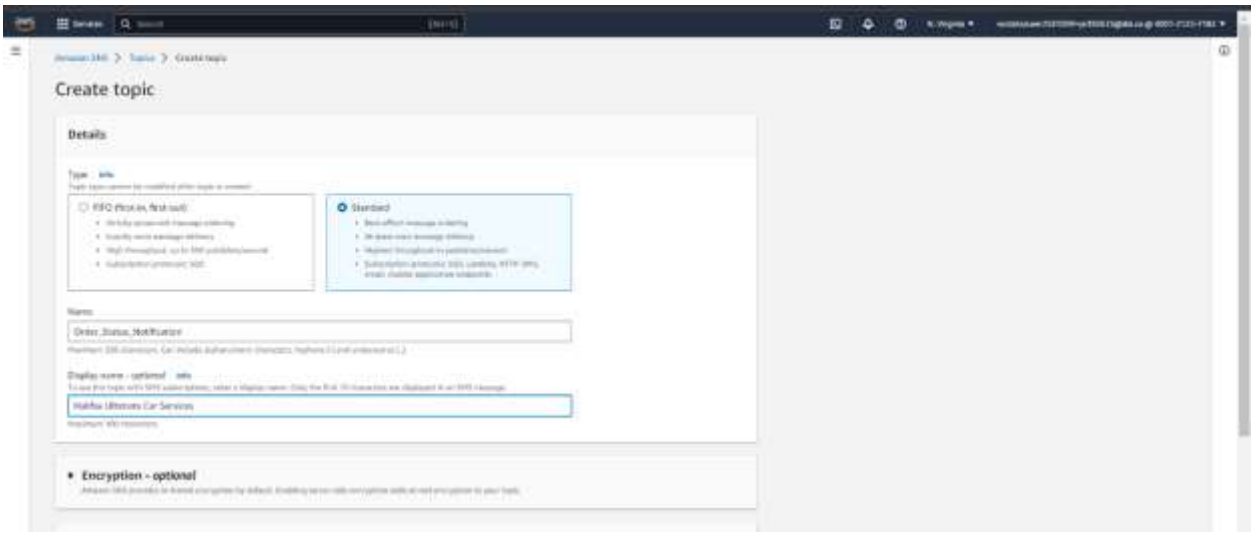

Fig 118

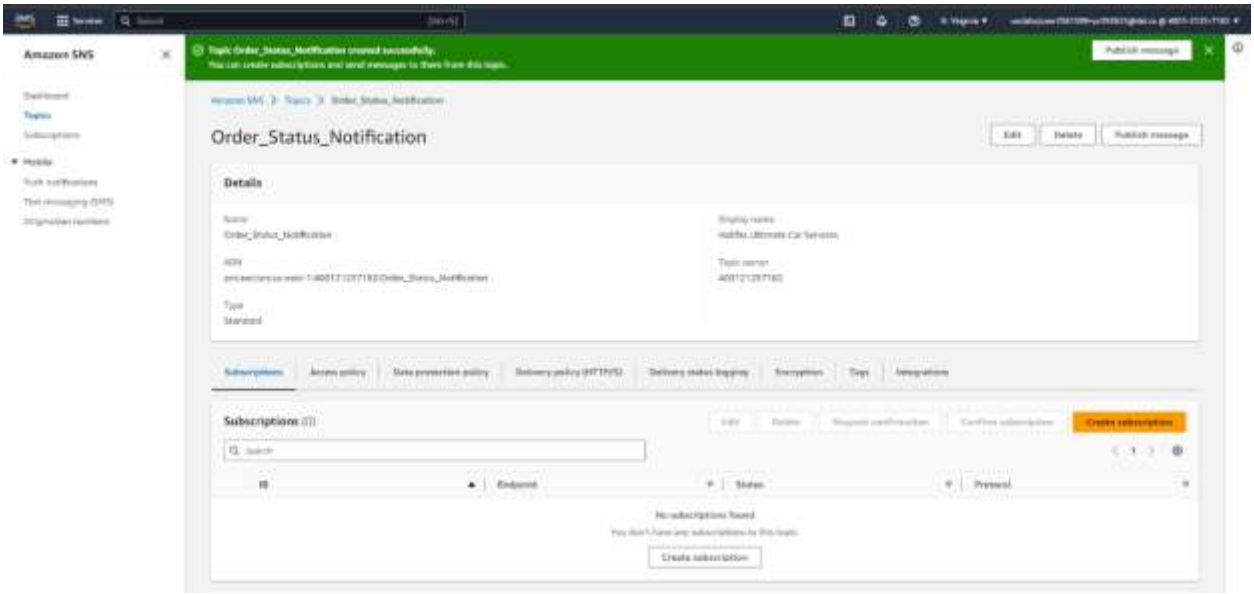

Fig 119

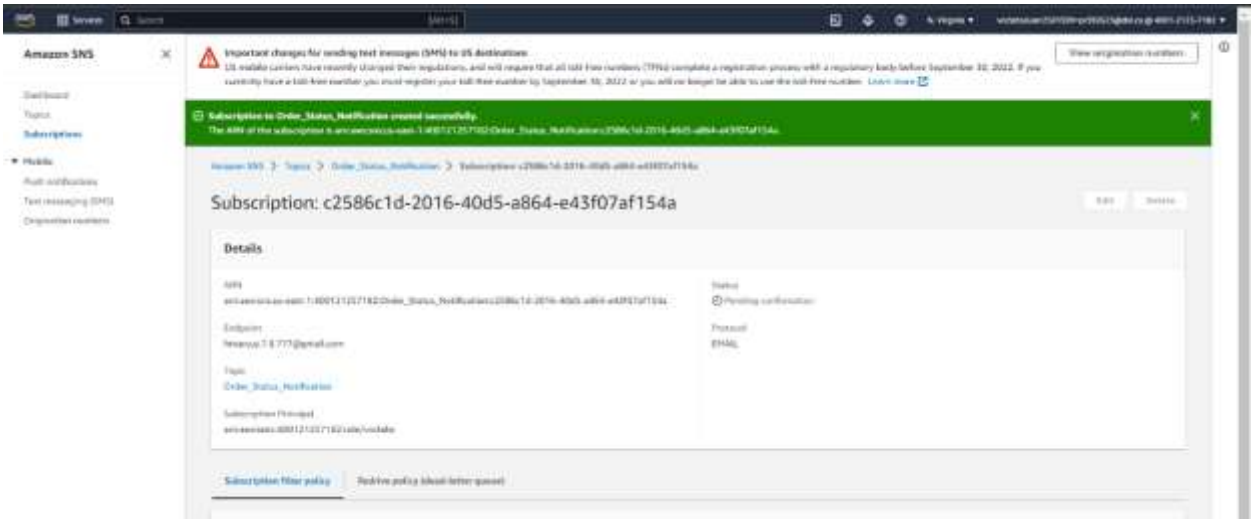

Fig 120

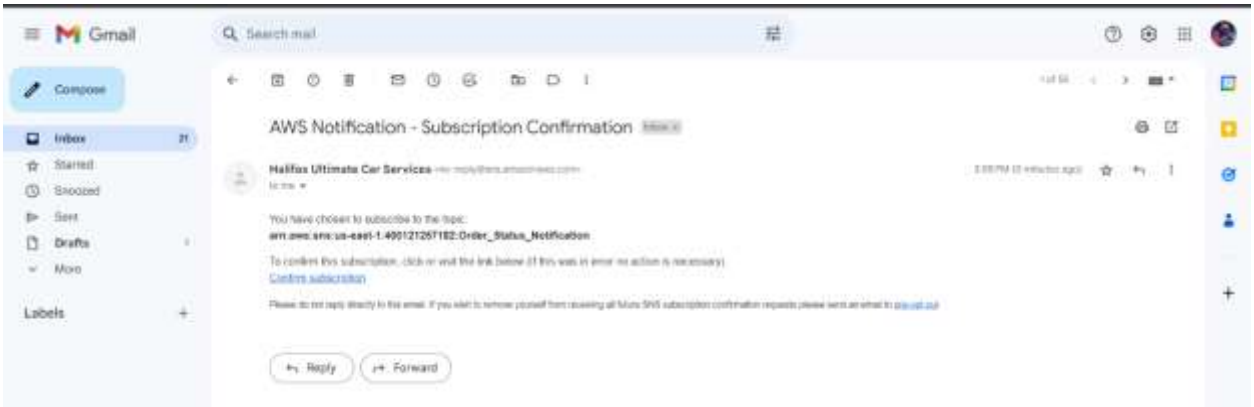

Fig 121

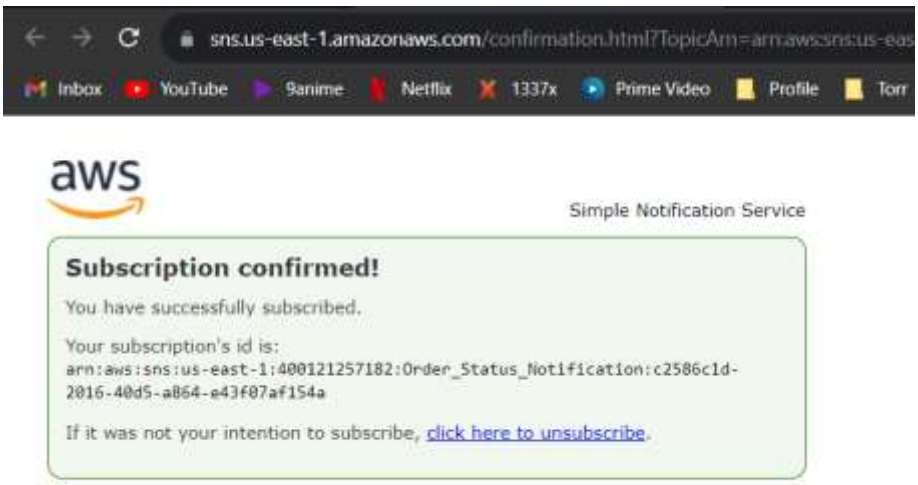

Fig 122

Step 6: Now we will begin the testing of our application by clicking on test button in our lambda function order_generation which will give us response order generated as shown in Fig 123, 124, 125. We can also check logs as shown in Fig 126, 127.

Fig 123

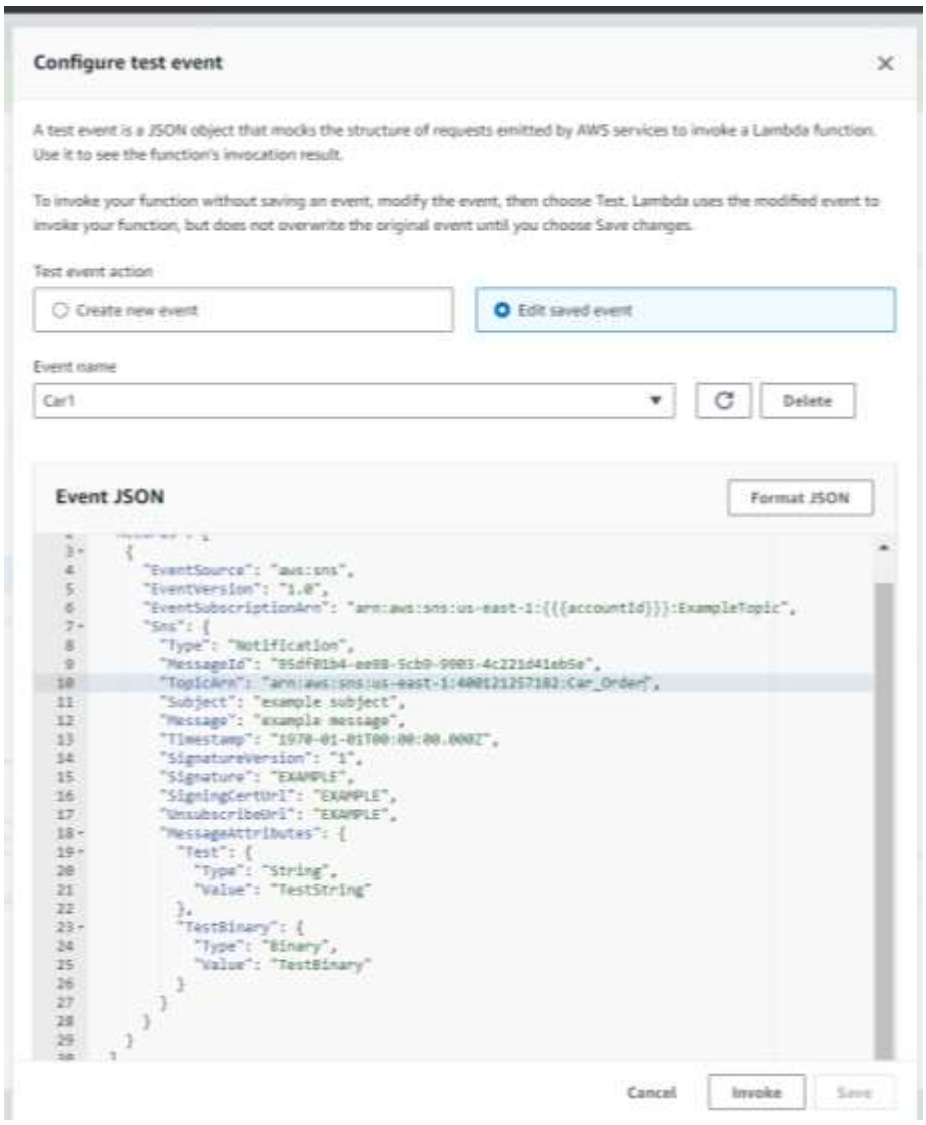
Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.
Test event action
Create new event
Edit saved event
Event name
Car1
Delete
Event JSON
Format JSON
{
"EventSource": "aws:sns",
"EventVersion": "1.0",
"EventSubscriptionArn": "arn:aws:sns:us-east-1:{{{accountId}}}:ExampleTopic",
"Sns": {
"Type": "Notification",
"MessageId": "95df01b4-ee98-5cb9-9903-4c221d41eb5e",
"TopicArn": "arn:aws:sns:us-east-1:400121257182:Car_Order",
"Subject": "example subject",
"Message": "example message",
"Timestamp": "1970-01-01T00:00:00.000Z",
"SignatureVersion": "1",
"Signature": "EXAMPLE",
"SigningCertUrl": "EXAMPLE",
"UnsubscribeUrl": "EXAMPLE",
"MessageAttributes": {
"Test": {
"Type": "String",
"Value": "TestString"
},
"TestBinary": {
"Type": "Binary",
"Value": "TestBinary"
}
}
}
}
}
Cancel
Invoke
Save

Fig 124

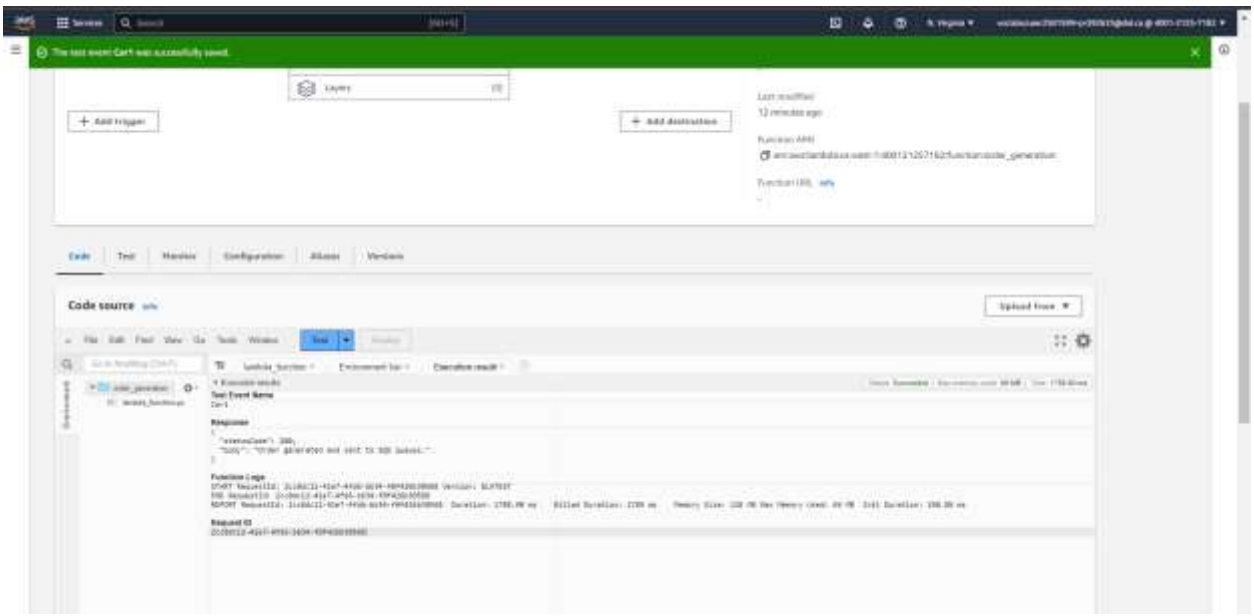

Fig 125

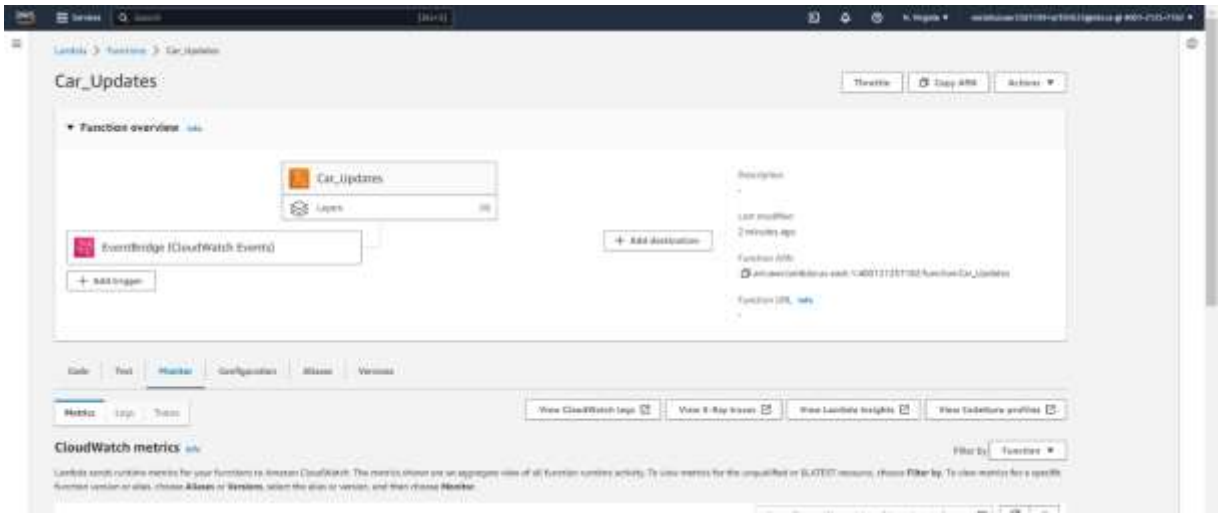

Fig 126

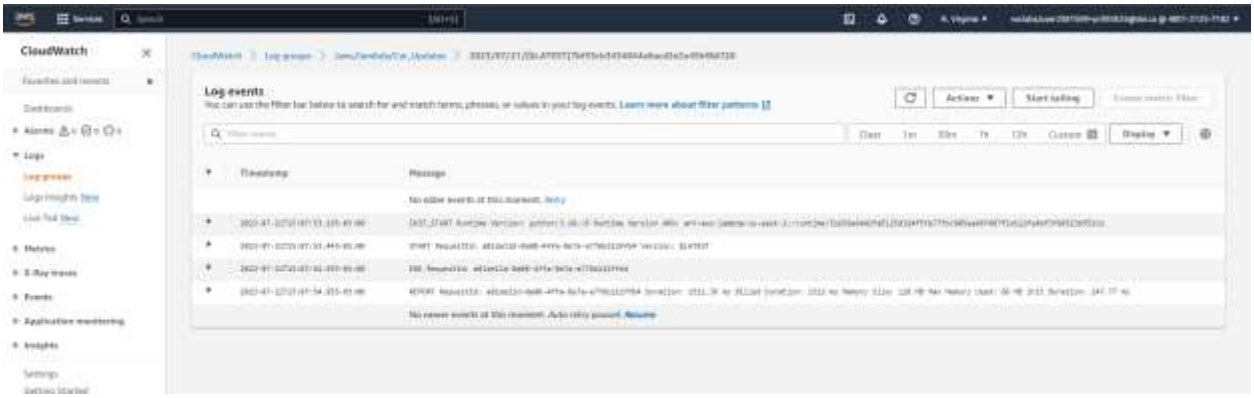

Fig 127

Step 7: We can verify that the notification has been sent by going to SQS which will show messages available 1 as shown in Fig 128. We can also see that we have received an email of the order as shown in Fig 129.

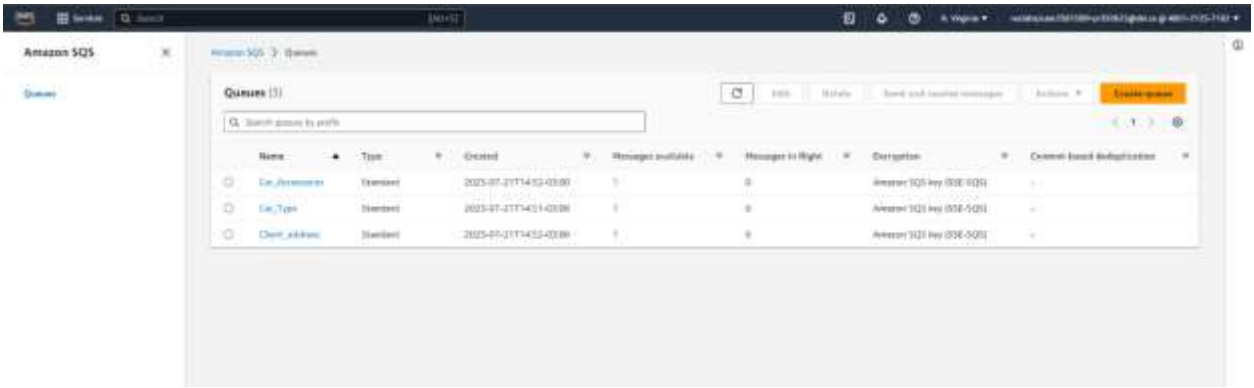

Fig 128

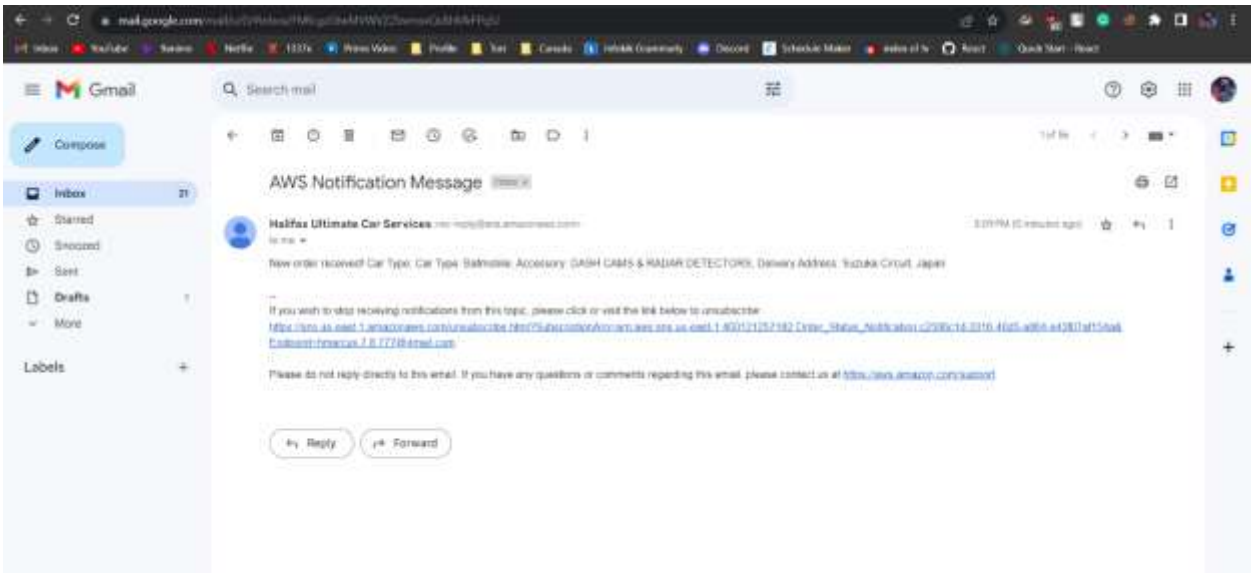

Fig 129

Step 8: I have sent a bunch of Tests like 5 and we can see that we have different results each time as shown in Fig 130, 131.

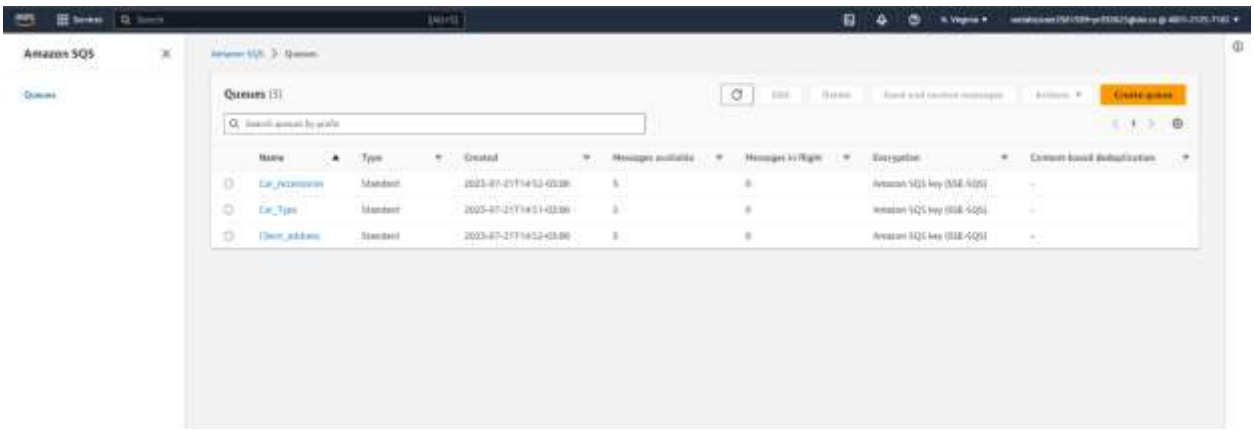

Fig 130

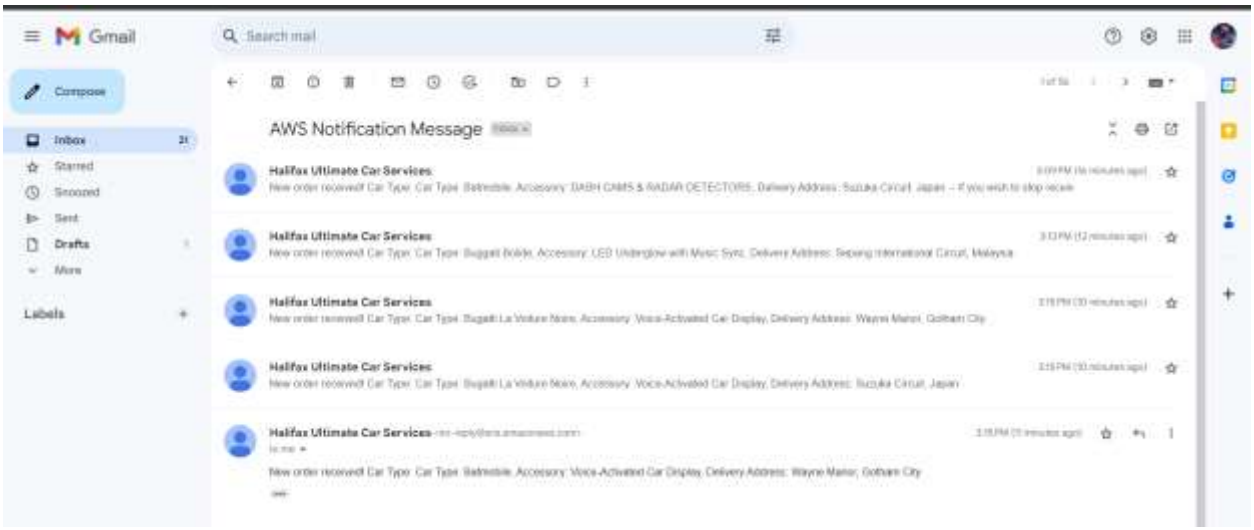

Fig 131