

PART – C

Strategy for the Problem

- Create 3 SNS and 3 SQS architecture to ease the complexity by removing the usage of filter policy.
- Using very basic configurations even for the security to keep it simple and achieve goal.

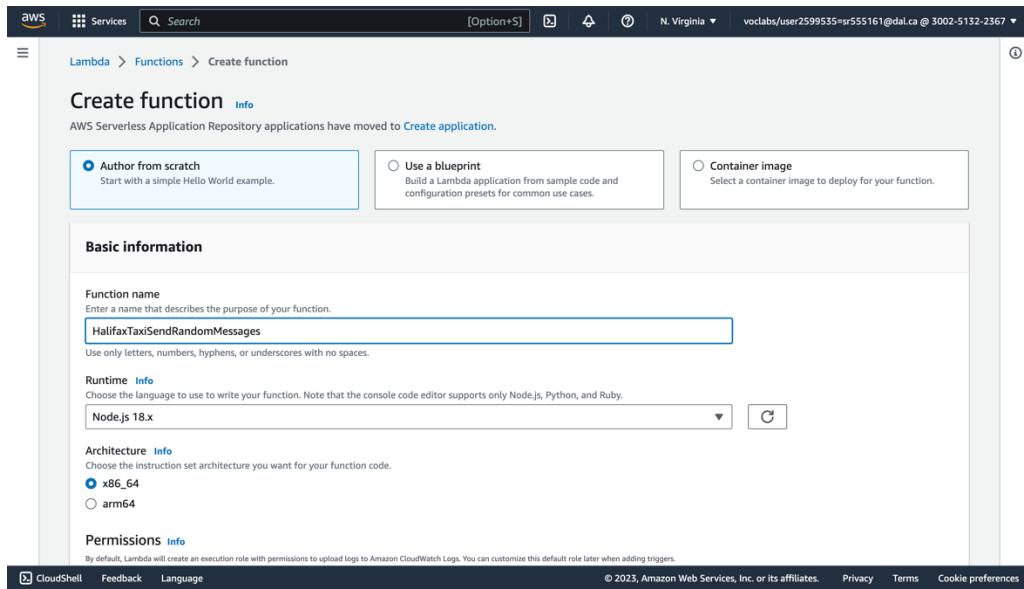


Fig. 1: Creating Lambda Function to send message to SNS.

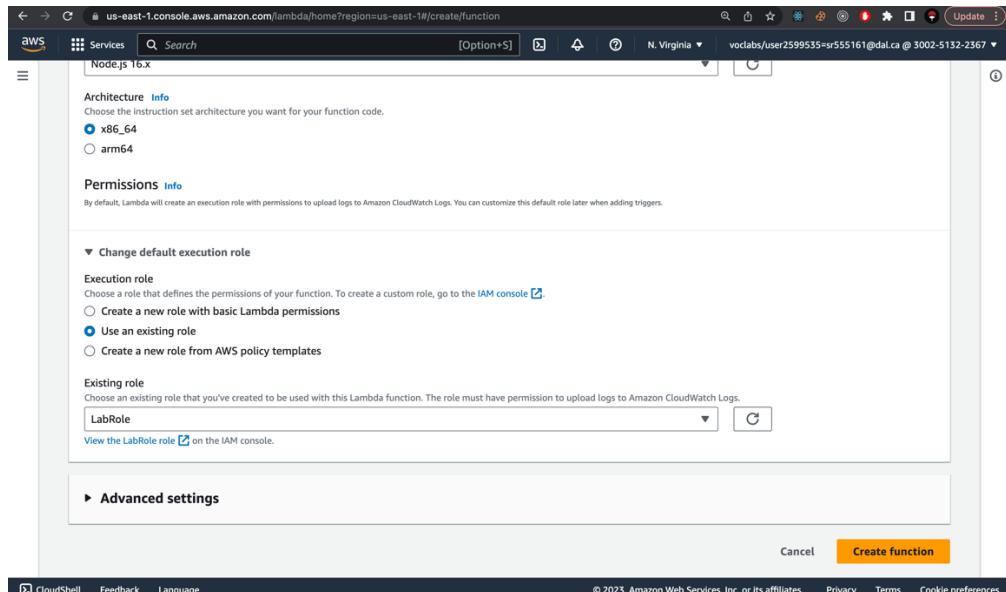


Fig. 2: Selecting LabRole as Existing Role

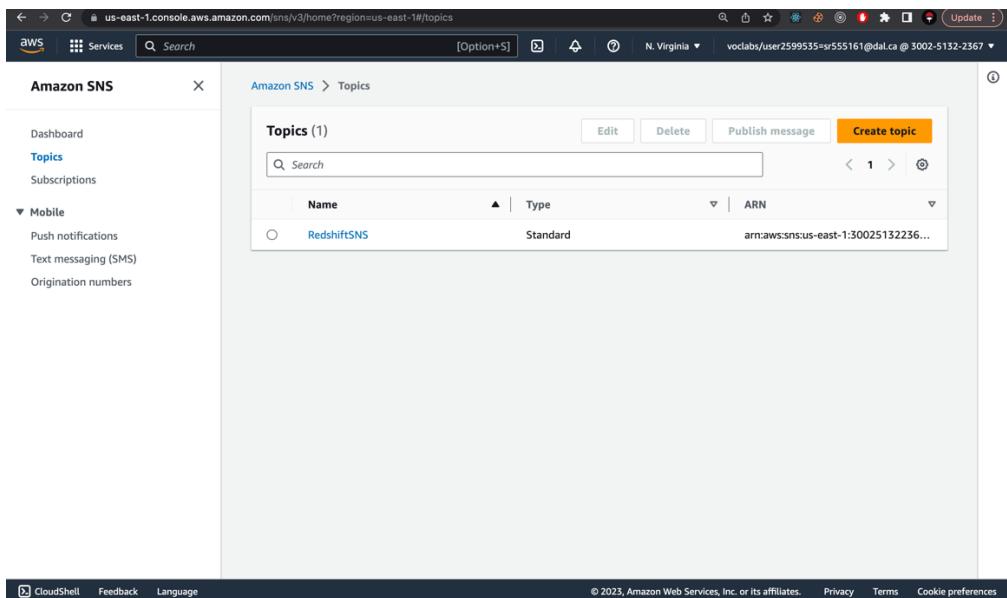


Fig. 3: SNS Dashboard

The screenshot shows the 'Create topic' wizard in the 'Details' step. It compares two topic types:

- FIFO (first-in, first-out)** (radio button not selected):
 - Strictly-preserved message ordering
 - Exactly-once message delivery
 - High throughput, up to 300 publishes/second
 - Subscription protocols: SQS
- Standard** (radio button selected):
 - Best-effort message ordering
 - At-least once message delivery
 - Highest throughput in publishes/second
 - Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Fields for 'Name' (Car) and 'Display name - optional' (My Topic) are filled.

Fig. 4: Creating Standard SNS for Car

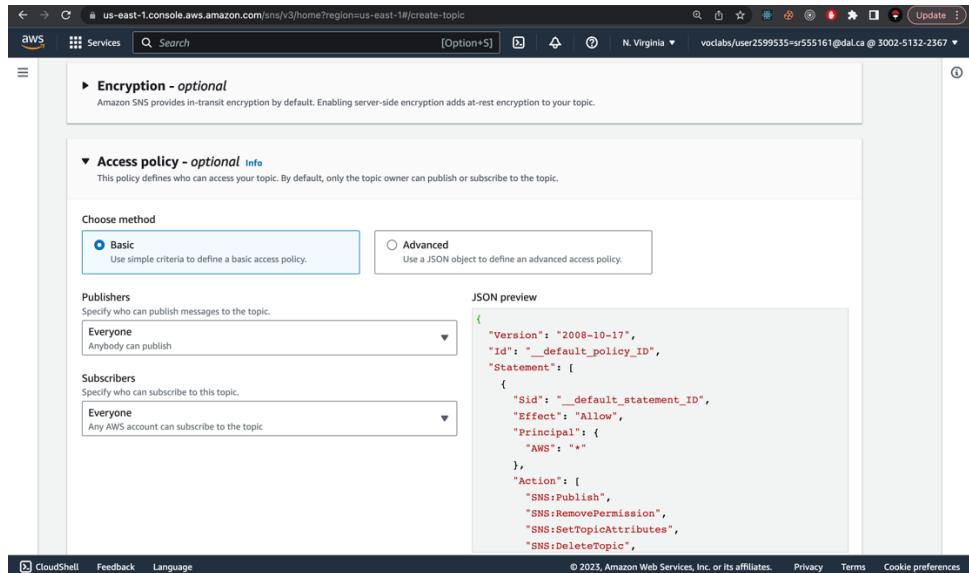


Fig. 5: Using very basic Access Policy for Car SNS.

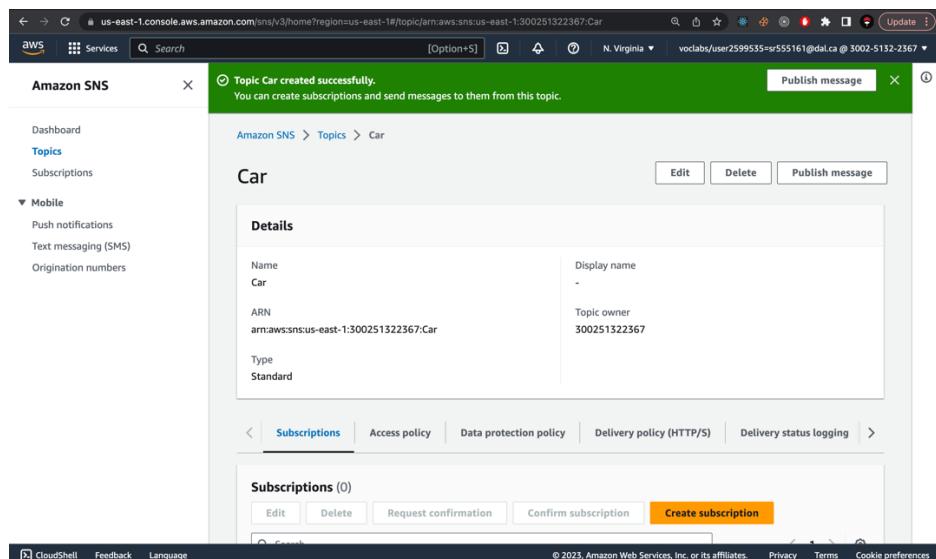


Fig. 6: SNS Car Created.

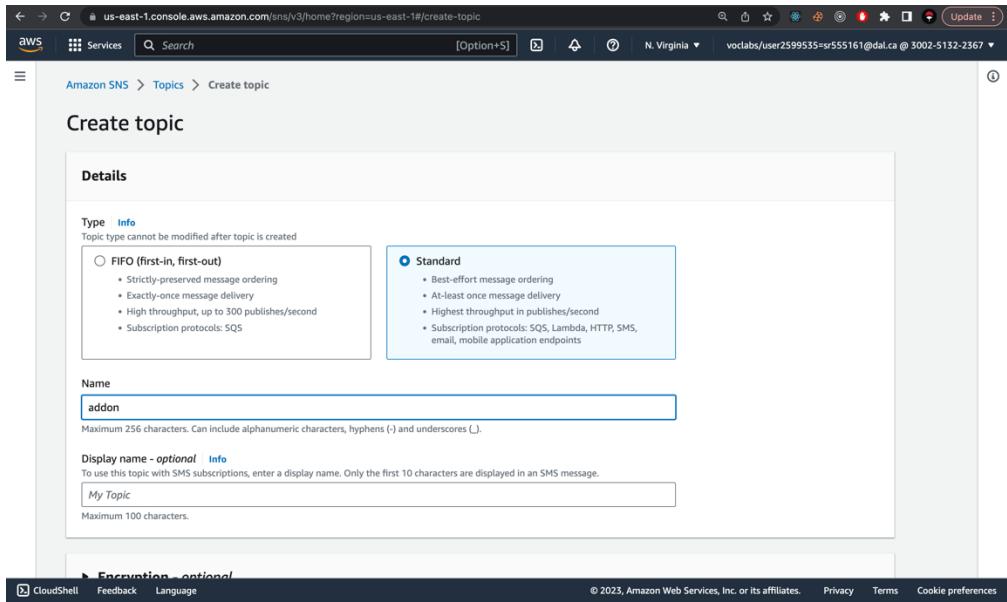


Fig. 7: Creating Standard SNS for addon

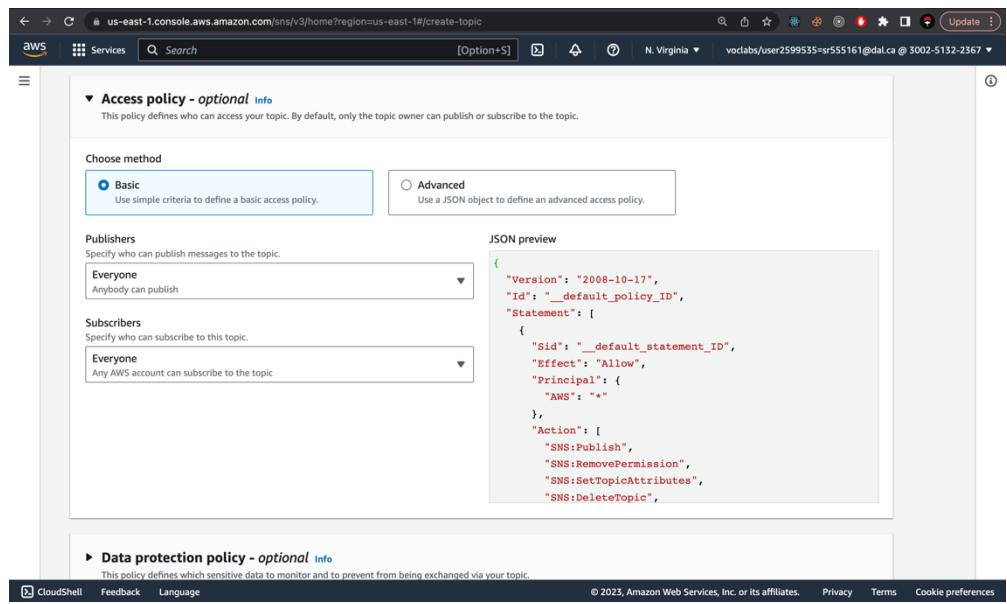


Fig. 8: Creating basic access policy for addon SNS.

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

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.

[Encryption - optional](#)

CloudShell Feedback Language © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig. 9: Creating Standard SNS for client.

▼ Access policy - optional [Info](#)
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

Choose method
 Basic Use simple criteria to define a basic access policy.
 Advanced Use a JSON object to define an advanced access policy.

Publishers
Specify who can publish messages to the topic.
 Anybody can publish

Subscribers
Specify who can subscribe to this topic.
 Any AWS account can subscribe to the topic

JSON preview

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:Publish",
        "SNS:RemovePermission",
        "SNS:SetTopicAttributes",
        "SNS:DeleteTopic"
      ]
    }
  ]
}
```

CloudShell Feedback Language © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig. 10: Creating basic access policy for SNS client.

The screenshot shows the 'Create queue' wizard in the AWS SQS console. In the 'Details' step, the 'Type' section is open, showing two options: 'Standard Info' (selected) and 'FIFO Info' (selected). The 'FIFO Info' section is highlighted with a blue border. Below it, a note says 'You can't change the queue type after you create a queue.' The 'Name' field contains 'addon'. The 'Configuration' section is collapsed.

Fig. 11: Creating Standard SQS queue for addon.

The screenshot shows the 'Queues' list page in the AWS SQS console. It displays three queues: 'addon', 'Car', and 'client', all of which are Standard type. The 'Create queue' button is visible at the top right of the table header.

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based
addon	Standard	2023-07-27T19:48-03:00	0	0	Disabled	-
Car	Standard	2023-07-27T19:50-03:00	0	0	Disabled	-
client	Standard	2023-07-27T19:50-03:00	0	0	Disabled	-

Fig. 12: Similarly create three SQS for three entities.

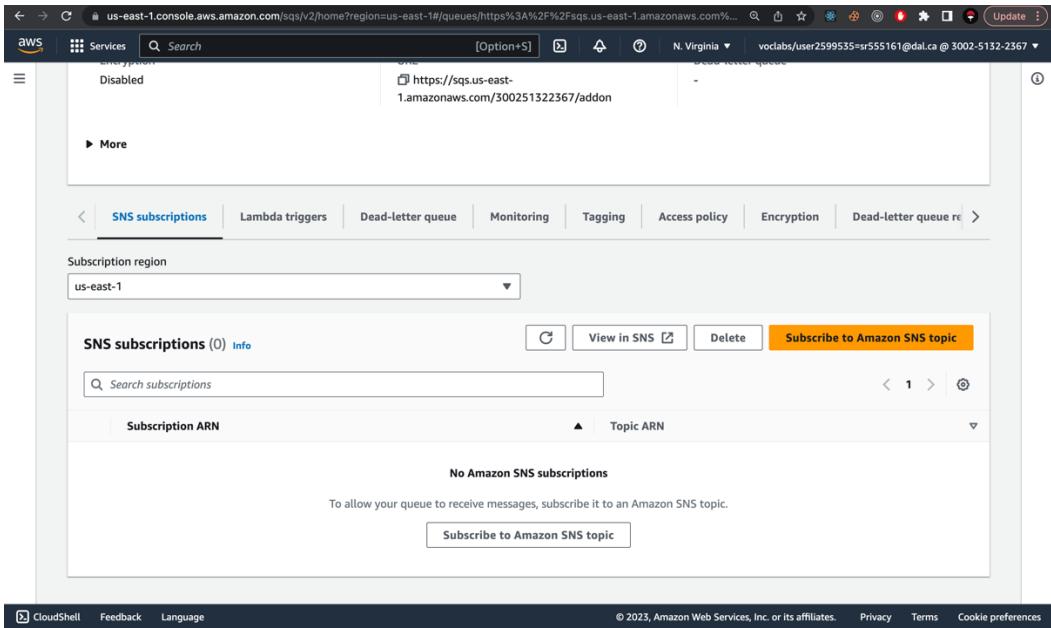


Fig. 13: Creating a SNS subscription for addon.

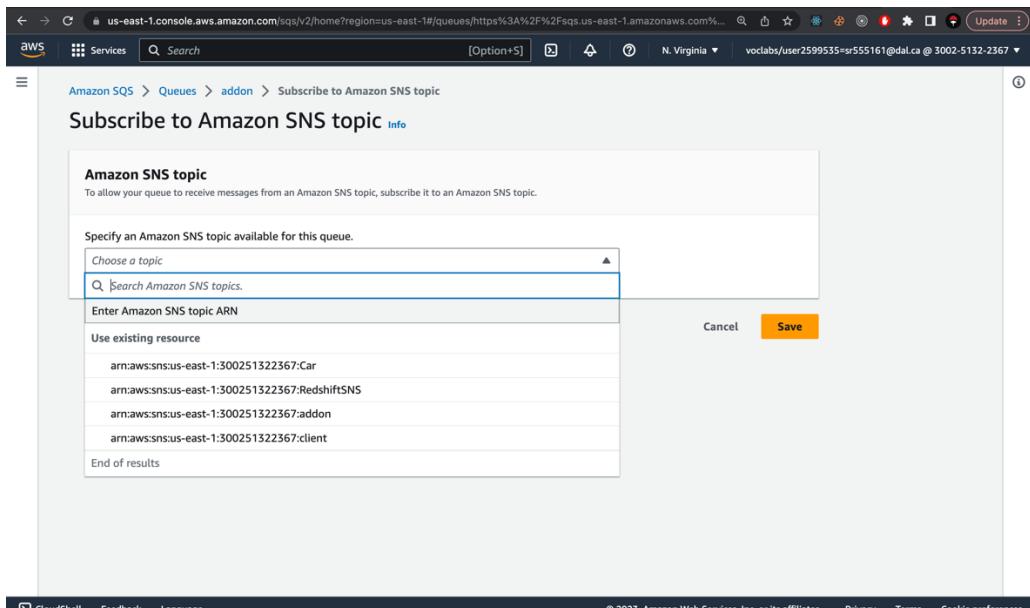


Fig. 14: Choosing addon SNS.

The screenshot shows the AWS SQS console for the 'addon' queue. The queue details are as follows:

Name	Type	ARN
addon	Standard	arn:aws:sqs:us-east-1:300251322367:addon
Encryption	Disabled	Dead-letter queue
		-

Below the queue details, there is a 'More' button. The navigation bar at the bottom includes tabs for 'SNS subscriptions', 'Lambda triggers', 'Dead-letter queue', 'Monitoring', 'Tagging', 'Access policy', 'Encryption', and 'Dead-letter queue rules'.

The 'SNS subscriptions' tab is selected, showing one subscription:

Subscription ARN	Topic ARN
arn:aws:sns:us-east-1:300251322367:addon:a3dd06aa-1f61-469e-bb9e-358786e5f4e9	arn:aws:sns:us-east-1:300251322367:addon

Buttons for 'View in SNS', 'Delete', and 'Subscribe to Amazon SNS topic' are present.

Fig. 15: Subscription for addon SQS created with addon SNS.

The screenshot shows the AWS SQS console for the 'Car' queue. The queue details are as follows:

Name	Type	ARN
Car	Disabled	-

Below the queue details, there is a 'More' button. The navigation bar at the bottom includes tabs for 'SNS subscriptions', 'Lambda triggers', 'Dead-letter queue', 'Monitoring', 'Tagging', 'Access policy', 'Encryption', and 'Dead-letter queue rules'.

The 'SNS subscriptions' tab is selected, showing zero subscriptions:

Subscription ARN	Topic ARN

A message states: "To allow your queue to receive messages, subscribe it to an Amazon SNS topic." A 'Subscribe to Amazon SNS topic' button is available.

Fig. 16: Subscription for Car SQS.

The screenshot shows the AWS SQS console with a single queue named 'Car'. The queue details are as follows:

Name	Type	ARN
Car	Standard	arn:aws:sqs:us-east-1:300251322367:Car
Encryption	Disabled	Dead-letter queue
		-

Below the queue details, there is a navigation bar with tabs: SNS subscriptions, Lambda triggers, Dead-letter queue, Monitoring, Tagging, Access policy, Encryption, and Dead-letter queue re. The 'SNS subscriptions' tab is selected. A dropdown menu for 'Subscription region' is open, showing 'us-east-1'.

The 'SNS subscriptions' section displays one entry:

Subscription ARN	Topic ARN
arn:aws:sns:us-east-1:300251322367:Car:039ae899-bcc5-4781-9140-19018107796f	arn:aws:sns:us-east-1:300251322367:Car

At the bottom of the page, there are links for CloudShell, Feedback, Language, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

Fig. 17: Subscription for Car SQS created with Car SNS.

The screenshot shows the AWS SQS console with a queue named 'Disabled'. The queue details are as follows:

Name	Type	ARN
Disabled		-

Below the queue details, there is a navigation bar with tabs: SNS subscriptions, Lambda triggers, Dead-letter queue, Monitoring, Tagging, Access policy, Encryption, and Dead-letter queue re. The 'SNS subscriptions' tab is selected. A dropdown menu for 'Subscription region' is open, showing 'us-east-1'.

The 'SNS subscriptions' section displays no entries:

Subscription ARN	Topic ARN
No Amazon SNS subscriptions	

To allow your queue to receive messages, subscribe it to an Amazon SNS topic. There is a 'Subscribe to Amazon SNS topic' button at the bottom of the list.

At the bottom of the page, there are links for CloudShell, Feedback, Language, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

Fig. 18: Subscription for client SQS.

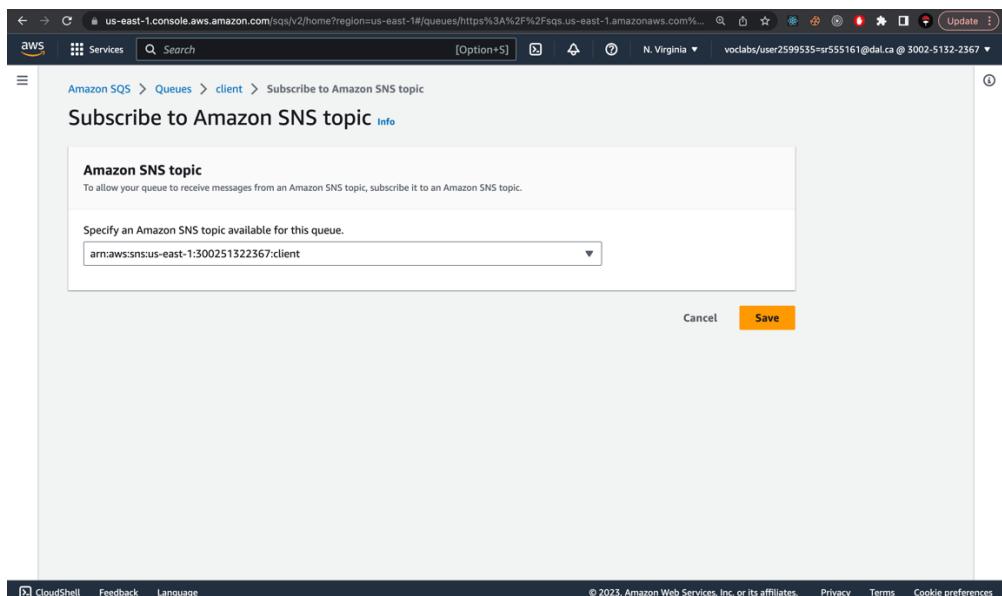


Fig. 19: Subscription for client SQS created with client SNS.

The screenshot shows the AWS SNS 'Topics' page for the 'Car' topic. The ARN 'arn:aws:sns:us-east-1:300251322367:Car' is copied from the 'Endpoint' column of the 'Subscriptions' table.

ID	Endpoint	Status	Protocol
	arn:aws:sns:us-east-1:300251322367:Car		

Fig. 20: Copying Car SNS ARN.

```

const response = {
  statusCode: 200,
  body: JSON.stringify({
    randomCar,
    randomAddon,
    randomClient
  })
};

callback(null, response);

```

```

function snsCar() {
  return new Promise((resolve, reject) => {
    const car = ["Compact", "mid-size", "Sedan", "SUV", "Luxury"];
    const randomCar = car[Math.floor(Math.random() * car.length)];
    var sns = new AWS.SNS();
    var params = {
      Message: randomCar,
      Subject: "Random Car type from Lambda",
      TopicArn: "arn:aws:sns:us-east-1:300251322367:Car"
    };
    sns.publish(params, (err, data) => {
      if (err) {
        reject(err);
      } else {
        resolve(randomCar);
      }
    });
  });
}

```

Fig. 21: Pasting the Car SNS ARN in lambda.

ID	Endpoint	Status	Protocol
1			

Fig. 22: Copying the addon SNS ARN in lambda.

The screenshot shows the AWS Lambda function editor for the 'HalifaxTaxiSendRandomMessages' function. The code source tab is selected, displaying the following JavaScript code:

```

30     sns.publish(params, (err, data) => {
31       if (err) {
32         reject(err);
33       } else {
34         resolve(randomCar);
35       }
36     });
37   });
38 }
39
40 function snsAddonO{
41   return new Promise((resolve, reject) => {
42     const addoon = ["GPS", "Camera"];
43     const randomAddon = addoon[Math.floor(Math.random() * addoon.length)];
44     var sns = new AWS.SNS();
45
46     var params = {
47       Message: randomAddon,
48       Subject: "Random Addon from Lambda",
49       TopicArn: "arn:aws:sns:us-east-1:300251322367:addon"
50     };
51     sns.publish(params, (err, data) => {
52       if (err) {
53         reject(err);
54       } else {
55         resolve(randomAddon);
56       }
57     });
58   });
59 }

```

The browser address bar shows the URL: us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/HalifaxTaxiSendRandomMessages?newFunction=...

Fig. 23: Pasting the addon SNS ARN in lambda.

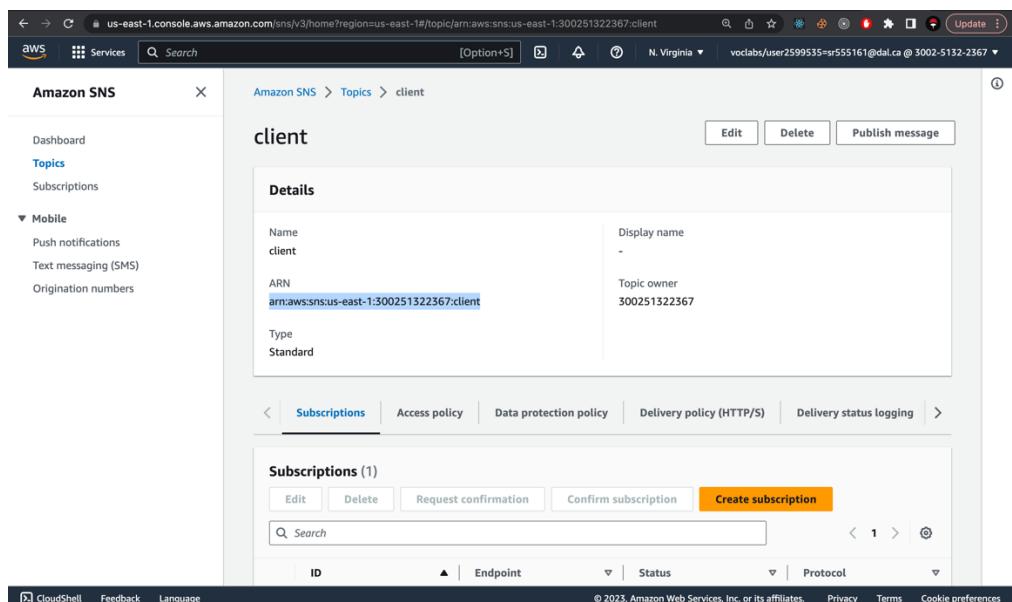


Fig. 24: Copying the client SNS ARN in lambda.

```

    42     return new Promise((resolve, reject) => {
43       const client = ["6050 University Avenue", "6967 Boyers Road", "2327 Brunswick Street"];
44       const randomClient = client[Math.floor(Math.random() * client.length)];
45       var sns = new AWS.SNS();
46
47       var params = {
48         Message: randomClient,
49         Subject: "Random Client from Lambda",
50         TopicArn: "arn:aws:sns:us-east-1:300251322367:client"
51     };
52     sns.publish(params, (err, data) => {
53       if (err) {
54         reject(err);
55       } else {
56         resolve(randomClient);
57       }
58     });
59   });
60 }

```

Fig. 25: Pasting the client SNS ARN in lambda.

Amazon SNS > Topics > client > Subscription: 0b0fb65a-0bd9-4c80-a1fd-a7af1cd9e7b5 > Edit subscription

Edit 0b0fb65a-0bd9-4c80-a1fd-a7af1cd9e7b5

Details

Topic
arn:aws:sns:us-east-1:300251322367:client

Protocol
Amazon SQS

Endpoint
arn:aws:sqs:us-east-1:300251322367:client

Enable raw message delivery

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.

Fig. 26: Enabling Raw Message Delivery for client SNS.

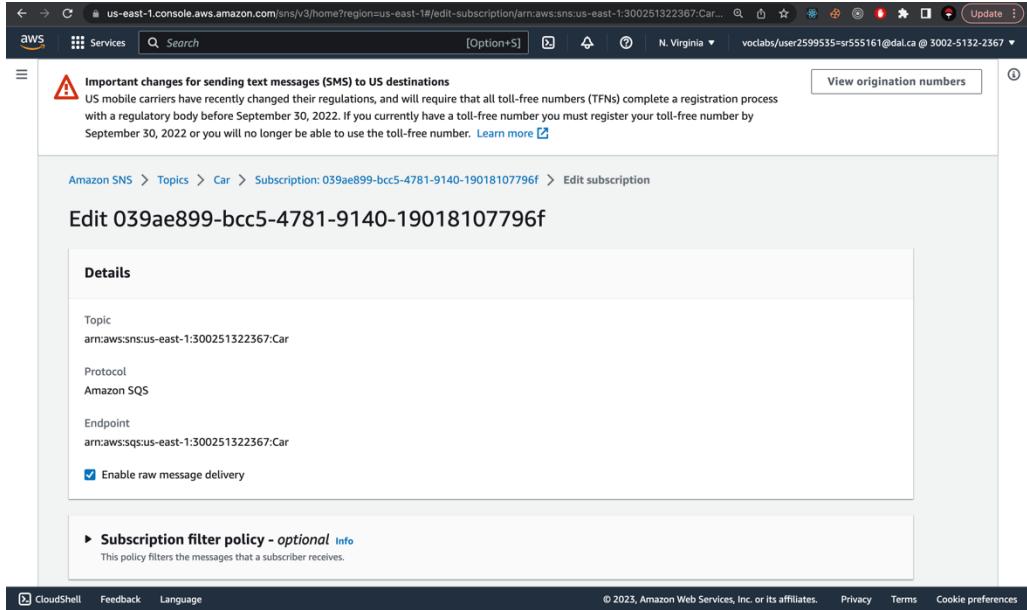


Fig. 27: Enabling Raw Message Delivery for Car SNS.

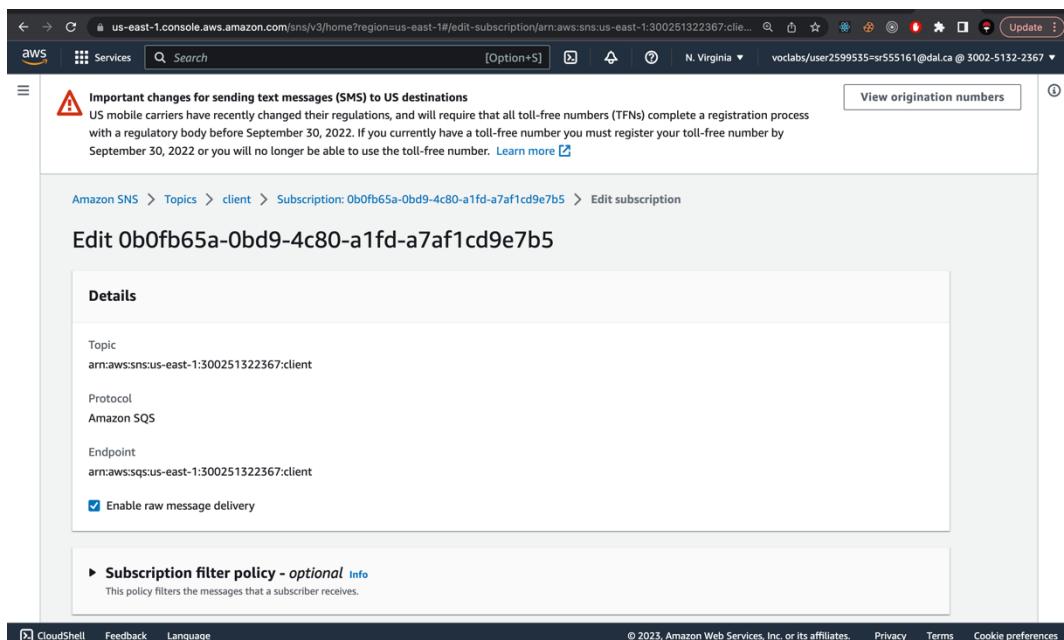


Fig. 28: Enabling Raw Message Delivery for addon.

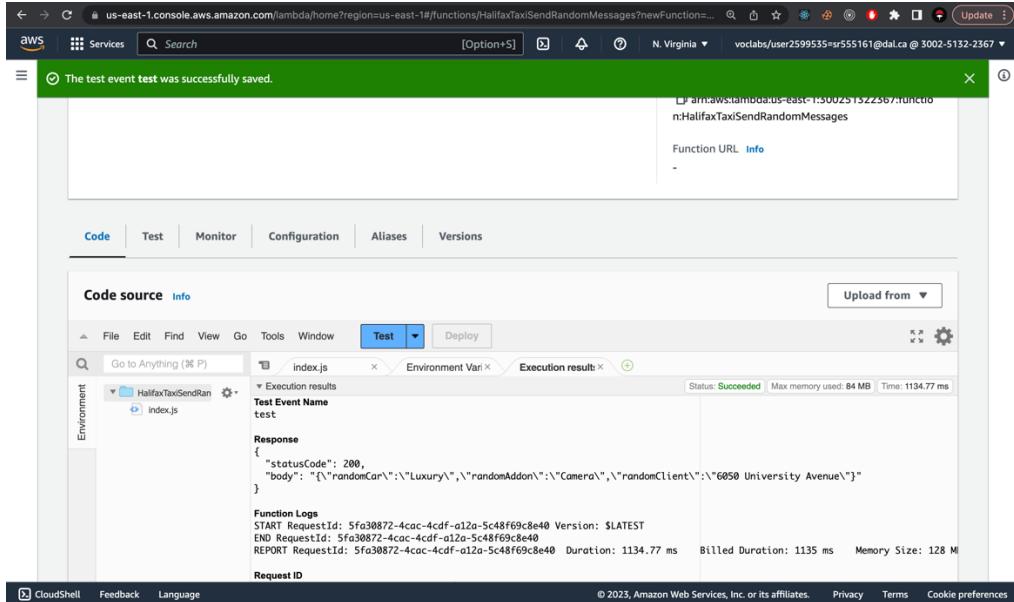


Fig. 29: Random Messages sent to AWS SQS through SNS.

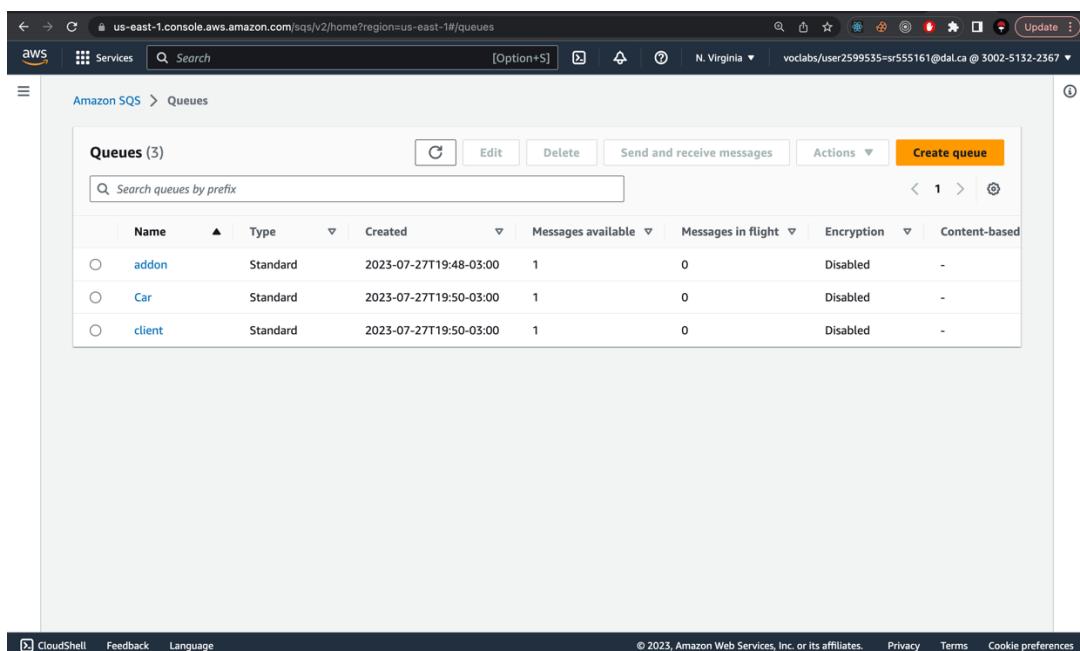


Fig. 30: Messages available in SQS Queue.

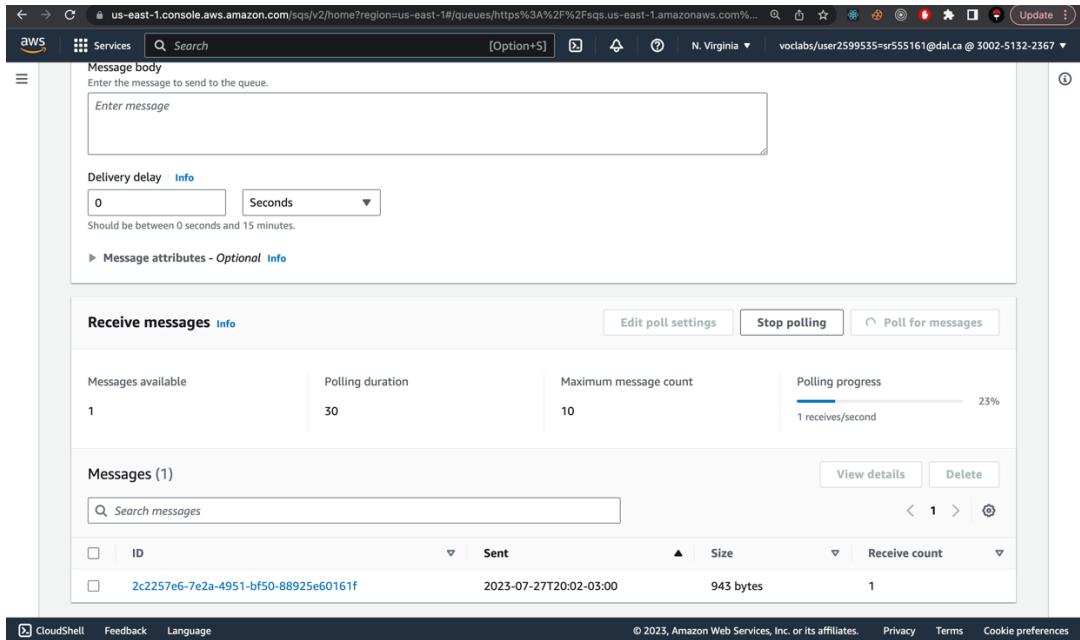


Fig. 31: Polling addon SQS for messages.

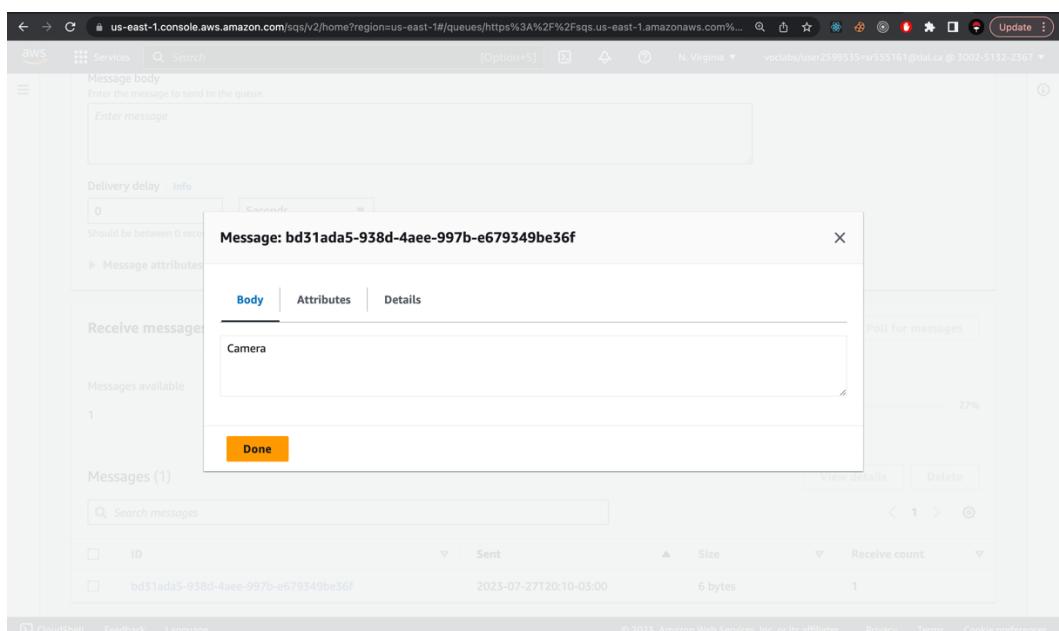


Fig. 32: Message from lambda to SNS to SQS.

The screenshot shows the 'Basic information' step of the Lambda creation wizard. The function name is set to 'HalifaxTaxiMessageFromSQS'. The runtime is chosen as 'Node.js 16.x'. The architecture is set to 'x86_64'. Under 'Permissions', it's noted that a default execution role will be created. The 'Change default execution role' section is expanded, showing options for creating a new role with basic permissions or using an existing one. The 'Use an existing role' option is selected. The 'Existing role' dropdown is currently empty.

Fig. 33: Another lambda to retrieve all the messages from Queue.

The screenshot shows the 'Create topic' step of the SNS wizard. The 'Type' is set to 'Standard'. The 'Name' field contains 'SendEmail'. The 'Display name - optional' field contains 'My Topic'. The 'Encryption - optional' section is collapsed.

Fig. 34: Creating standard SNS to send email.

The screenshot shows the AWS SNS console with the 'Topics' section selected. A new topic named 'SendEmail' is being configured. The 'Subscriptions' tab is active, showing a table with one row: 'No subscriptions found'. Below the table is a 'Create subscription' button. The URL in the browser is us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/topic/arm-aws:sns:us-east-1:300251322367:SendEmail.

Fig. 35: Creating subscription for SendEmail SNS.

The screenshot shows the 'Create subscription' wizard. In the 'Details' step, the 'Topic ARN' is set to 'arn:aws:sns:us-east-1:300251322367:SendEmail', the 'Protocol' is set to 'Email', and the 'Endpoint' is 'sarthak3136@gmail.com'. A note says 'After your subscription is created, you must confirm it.' In the 'Subscription filter policy - optional' step, there is a note about filtering messages. In the 'Redrive policy (dead-letter queue) - optional' step, there is a note about sending undeliverable messages to a dead-letter queue. The URL in the browser is us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/create-subscription.

Fig. 36: Choosing Protocol as email and providing email ID.

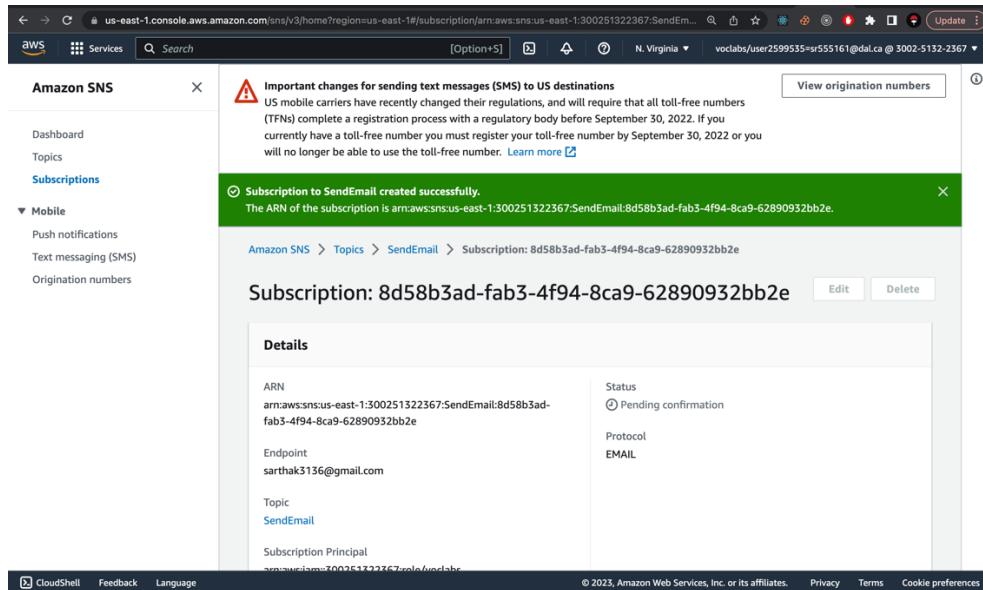


Fig. 37: Subscription for email created.

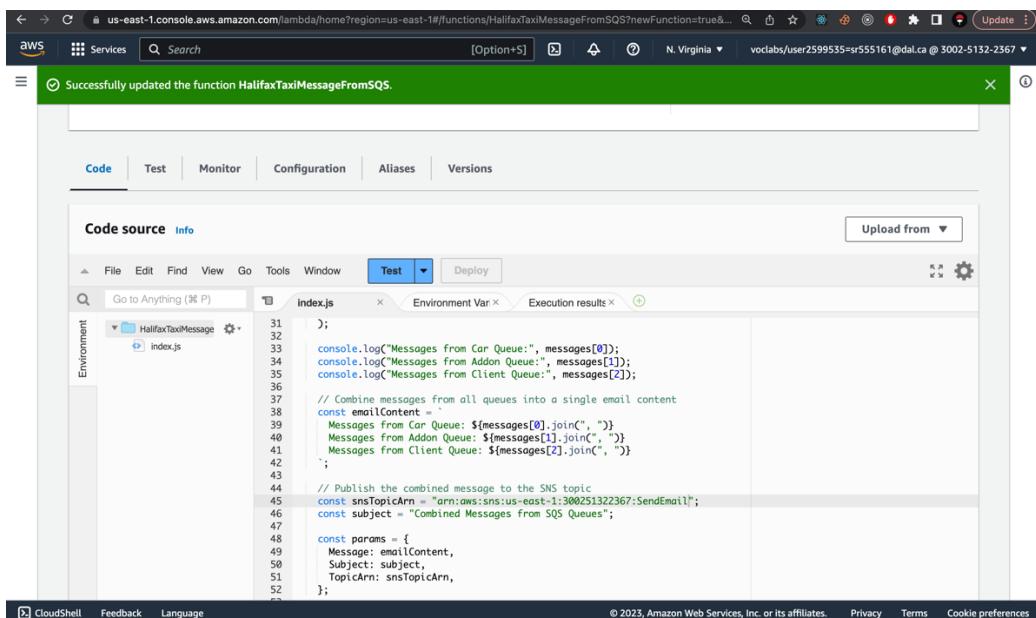


Fig. 38: Putting the SendEmail ARN in the second lambda.

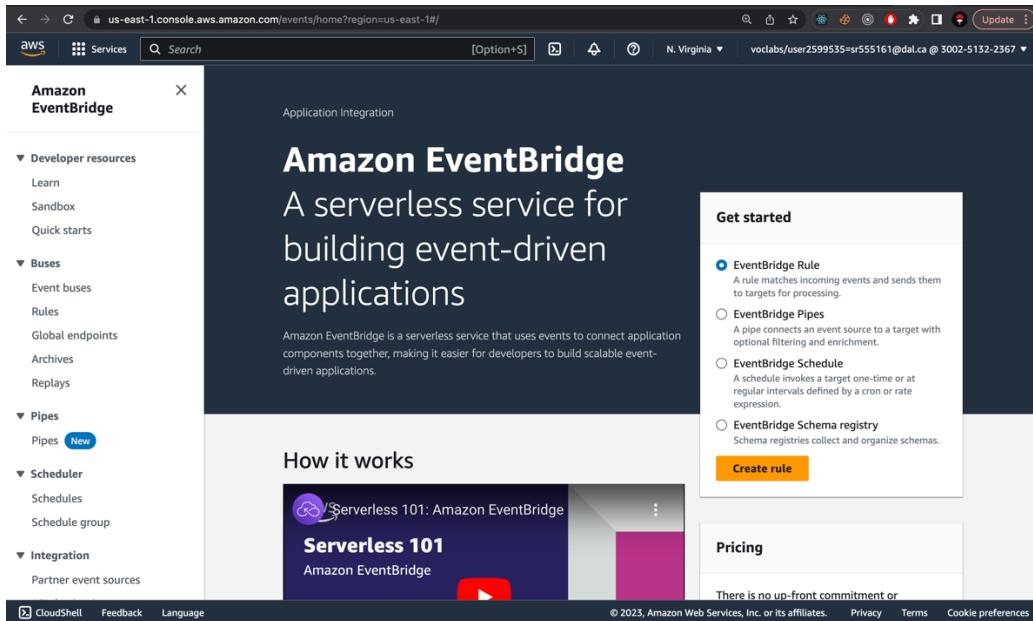


Fig. 39: Creating an event bridge to trigger lambda every 2 minutes.

Fig. 40: Naming Event Bridge Schedule.

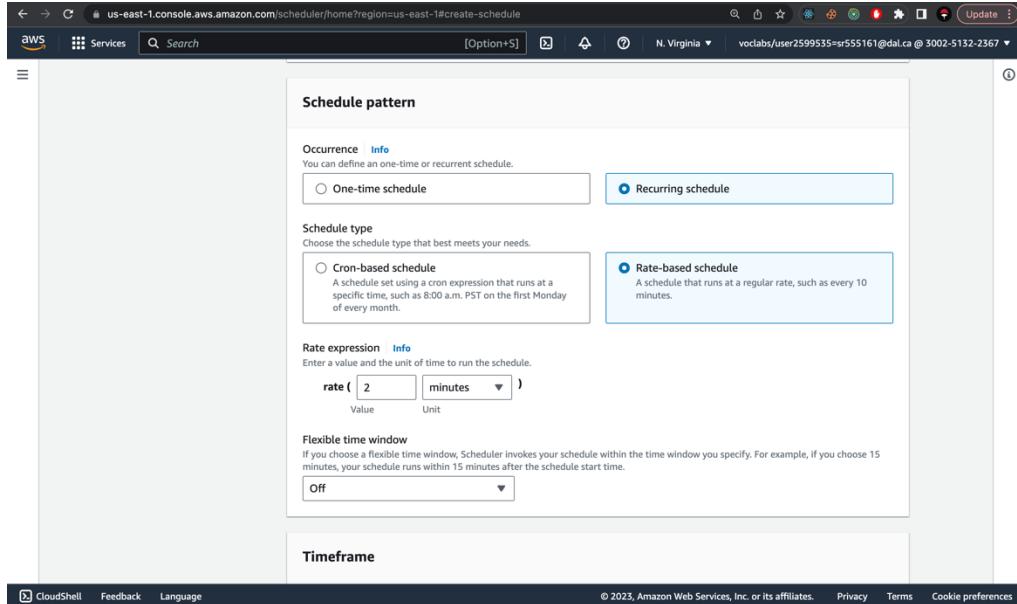


Fig. 41: Recurring schedule selected for every 2 minutes invocation.

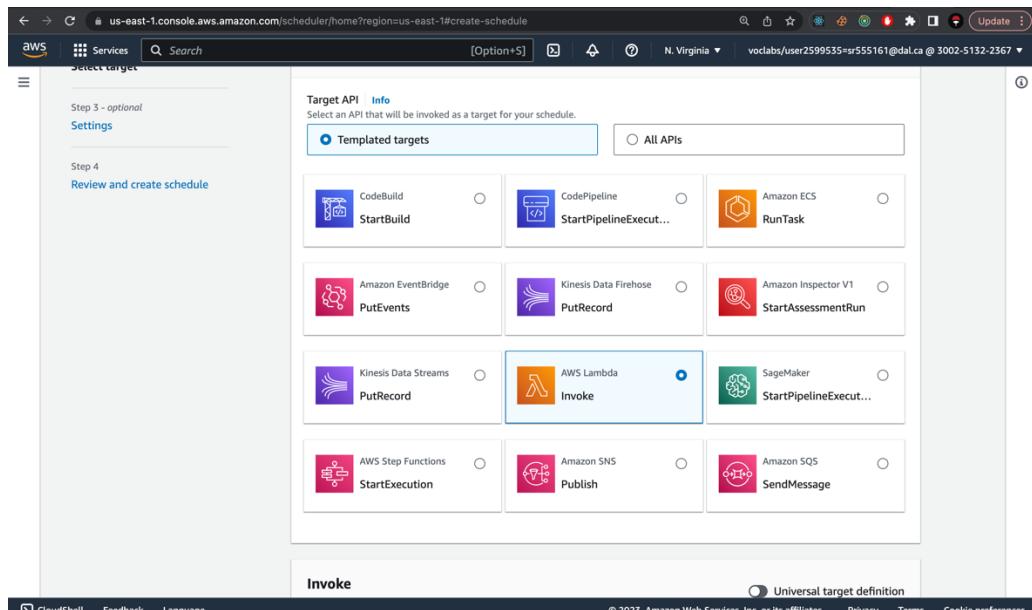


Fig. 42: Lambda selected for invocation.

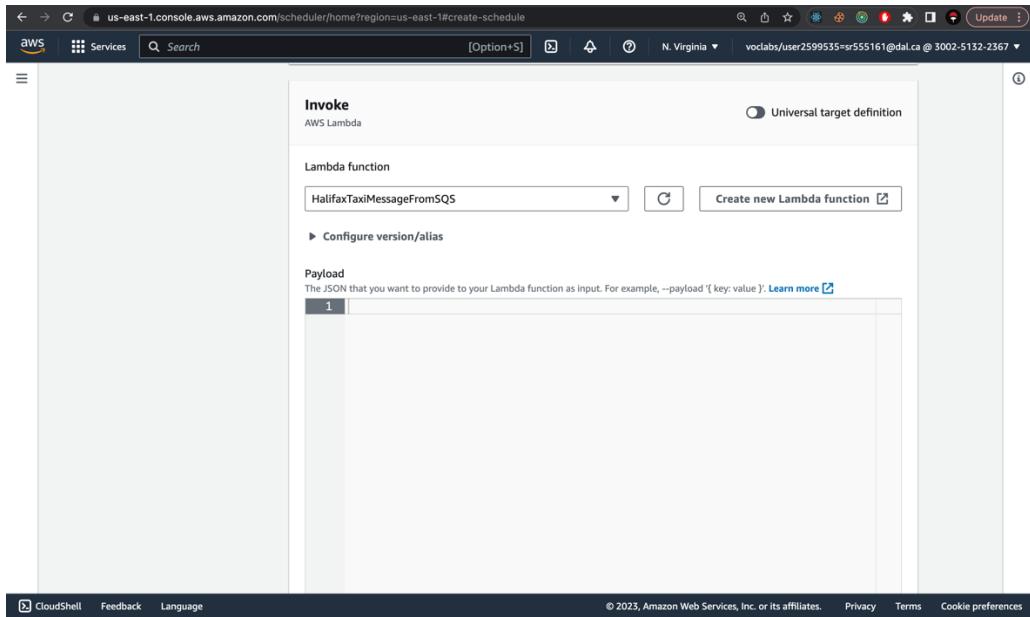


Fig. 43: Selecting the Lambda function to invoke.

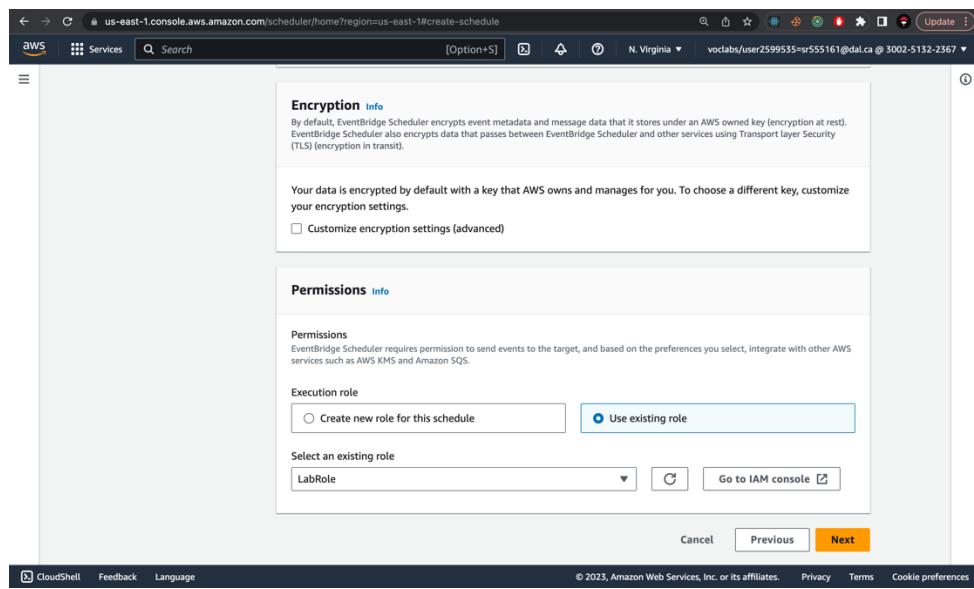


Fig. 44: Keeping the same IAM role as Lambda.

The screenshot shows the AWS EventBridge console with a green header bar indicating 'Your schedule TriggerLambdaForA3 is being created.' The main area displays the 'TriggerLambdaForA3' schedule detail table:

Schedule name	Status	Schedule start time	Flexible time window
TriggerLambdaForA3	Enabled	-	-
Description	Schedule ARN	Schedule end time	Created date
-	arn:aws:scheduler:us-east-1:300251322367:schedule/default/TriggerLambdaForA3	-	Jul 27, 2023, 20:22:14 (UTC-03:00)
Schedule group name	Execution timezone	Last modified date	
default	America/Halifax	Jul 27, 2023, 20:22:14 (UTC-03:00)	

Below the table are tabs for Schedule, Target, Retry policy, Dead-letter queue, and Encryption. The Schedule tab is selected.

Fig. 45: Event bridge for lambda created.

The screenshot shows a Gmail inbox with 8,387 messages. An email from 'AWS Notifications <no-reply@sns.amazonaws.com>' titled 'AWS Notification - Subscription Confirmation' is selected. The email body contains the following text:

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:300251322367:SendEmail
To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#).

At the bottom of the email are 'Reply' and 'Forward' buttons.

Fig. 46: Subscription permission email received.



Fig. 47: Subscription accepted.

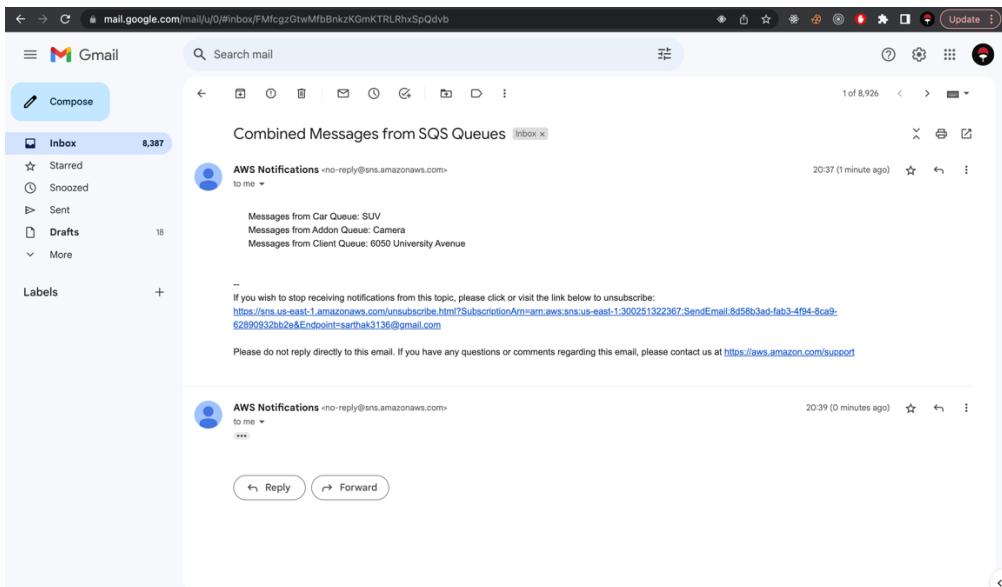


Fig. 48: After 2 minutes event bridge invoked lambda which sent email.