AWS LAMBDA-SQS-SNS

CSCI5410 – Assignment 5 – Part B

Dhrumil Amish Shah (B00857606)
dh416386@dal.ca

GitLab URL

The code for the Part B of Assignment 5 can be accessed at URL - https://git.cs.dal.ca/dashah/csci-5410-f2021-b00857606-dhrumil-amish-shah/-/tree/main/assignment_5_code/part_b/part_b_code

AWS Lambda-SQS-SNS

Amazon Web Services (AWS) is a cloud platform provided by Amazon that is highly accepted among developers, organizations, and emerging start-ups. AWS provides over 200 highly scalable services that allow developers to develop applications seamlessly. Services by AWS are offered in various categories such as compute, storage, analytics, networking, business applications, Machine Learning, Artificial Intelligence, IoT, and Security. For this assignment, I have developed an event-driven serverless application using three AWS services, namely AWS Lambda [1], AWS Simple Queue Service (SQS) [2], and AWS Simple Notification Service (SNS) [3].

The central idea behind this application is to create an online food delivery service – HalifaxDine that is truly serverless in nature. I have created this online order placement by customers at HalifaxDine simulator using the JAVA programming language and IntelliJ IDE. The program selects a random number of food items from the menu with quantity also randomized, prepares an order message body, and place it on a Standard Queue (i.e., on SQS). After every 5 minutes, a preconfigured Lambda function is triggered that polls the Standard Queue to check if an order exists in the queue. If it does exist, it is assumed that Bob, who works at HalifaxDine as a chef, has already prepared the order and is ready to be delivered. A notification is triggered from SNS which sends the order details to Alice's email address. Alice is also an employee at HalifaxDine whose job is to ensure that orders are delivered to customers hot and fresh.

Figure 1 displays the overall architecture of HalifaxDine – an online food ordering and delivery system prepared using AWS services.

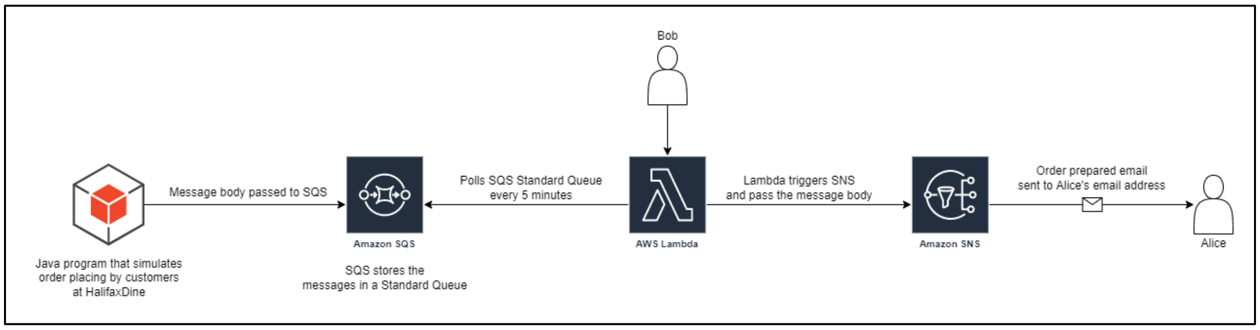
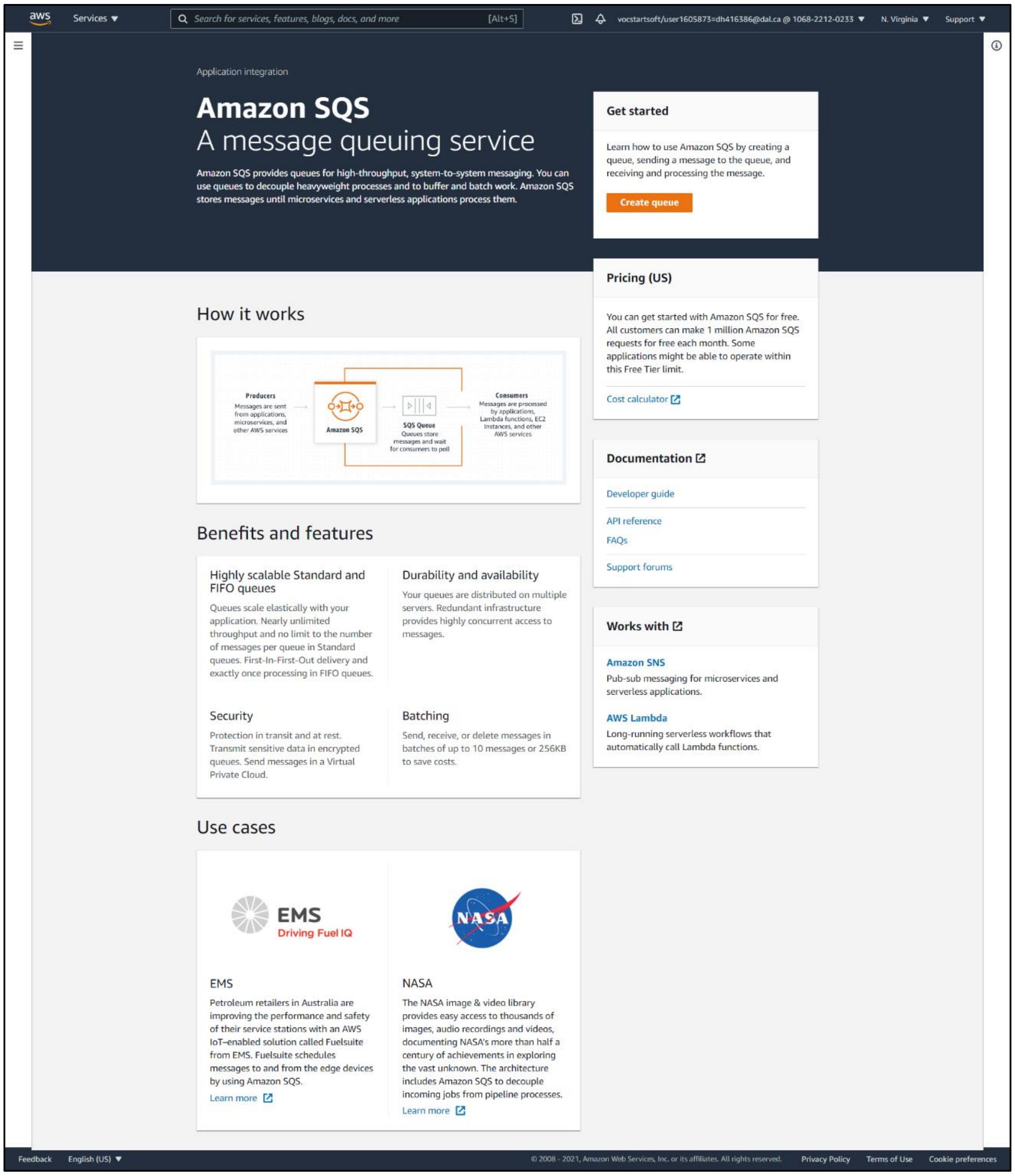

Bob
Message body passed to SQS
Polls SQS Standard Queue every 5 minutes
Lambda triggers SNS and pass the message body
Order prepared email sent to Alice's email address
Amazon SQS
AWS Lambda
Amazon SNS
Alice
Java program that simulates order placing by customers at HalifaxDine
SQS stores the messages in a Standard Queue

Figure 1 – AWS architecture of HalifaxDine [4]

Prerequisite Steps Images

Figures 1 to 22 display screenshots of the initial steps I performed to create this serverless application using AWS Lambda, AWS SNS, and AWS SQS. These steps are required for developing this application and are performed on AWS Console.

Figure 2 displays the screenshot of the Amazon SQS getting started page with a "Create Queue" button. I navigated to this page by typing SQS on the search bar from the navigation menu.


aws
Services ▼
Search for services, features, blogs, docs, and more
[Alt+S]
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233 ▼
N. Virginia ▼
Support ▼
Producers
Messages are sent from applications, microservices, and other AWS services
Amazon SQS
SQS Queue
Queues store messages and wait for consumers to poll
Consumers
Messages are processed by applications, Lambda functions, EC2 Instances, and other AWS services
Create queue
Feedback
English (US) ▼
Privacy Policy
Terms of Use
Cookie preferences

Application integration

Amazon SQS

A message queuing service

Amazon SQS provides queues for high-throughput, system-to-system messaging. You can use queues to decouple heavyweight processes and to buffer and batch work. Amazon SQS stores messages until microservices and serverless applications process them.

Get started

Learn how to use Amazon SQS by creating a queue, sending a message to the queue, and receiving and processing the message.

Pricing (US)

You can get started with Amazon SQS for free. All customers can make 1 million Amazon SQS requests for free each month. Some applications might be able to operate within this Free Tier limit.

Cost calculator

Documentation

Developer guide

API reference

FAQs

Support forums

Works with

Amazon SNS

Pub-sub messaging for microservices and serverless applications.

AWS Lambda

Long-running serverless workflows that automatically call Lambda functions.

How it works

Benefits and features

Highly scalable Standard and FIFO queues

Queues scale elastically with your application. Nearly unlimited throughput and no limit to the number of messages per queue in Standard queues. First-In-First-Out delivery and exactly once processing in FIFO queues.

Durability and availability

Your queues are distributed on multiple servers. Redundant infrastructure provides highly concurrent access to messages.

Security

Protection in transit and at rest. Transmit sensitive data in encrypted queues. Send messages in a Virtual Private Cloud.

Batching

Send, receive, or delete messages in batches of up to 10 messages or 256KB to save costs.

Use cases

EMS

Driving Fuel IQ

EMS

Petroleum retailers in Australia are improving the performance and safety of their service stations with an AWS IoT–enabled solution called Fuelsuite from EMS. Fuelsuite schedules messages to and from the edge devices by using Amazon SQS.

Learn more

NASA

The NASA image & video library provides easy access to thousands of images, audio recordings and videos, documenting NASA's more than half a century of achievements in exploring the vast unknown. The architecture includes Amazon SQS to decouple incoming jobs from pipeline processes.

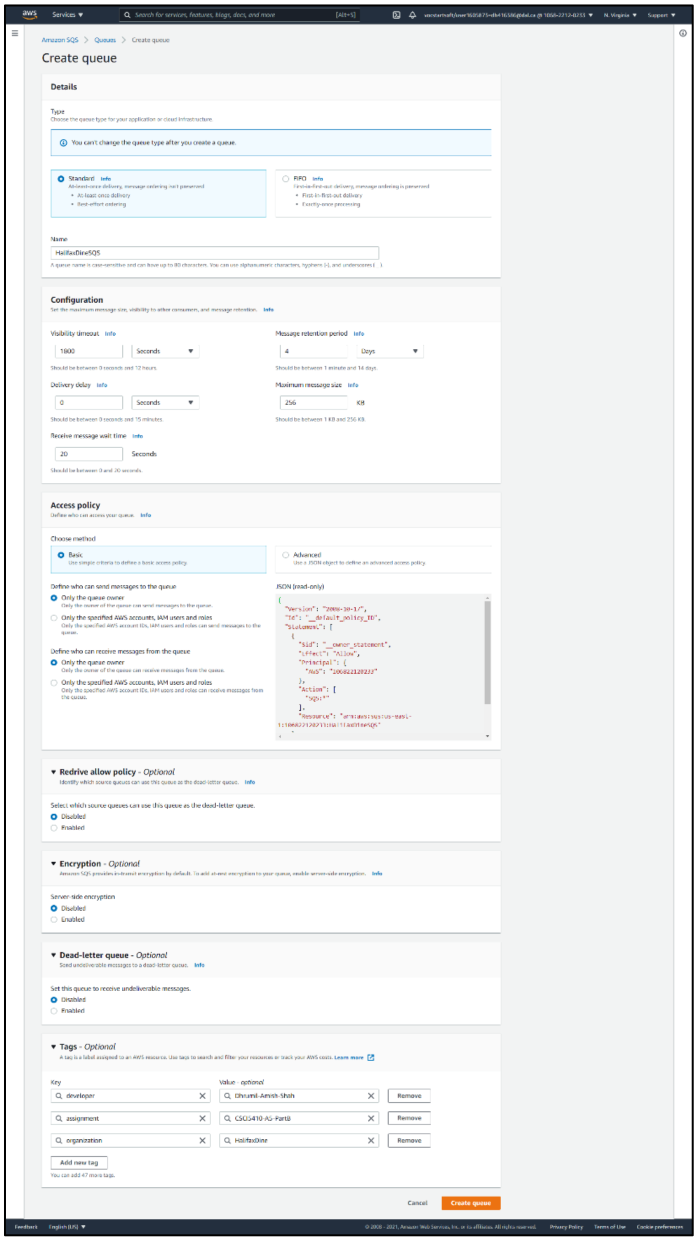
Learn more

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 2 – Amazon SQS Get started page [2]

Figure 3 displays the screenshot of the "Create Queue" page. The page consists of details, configuration, and access policy forms filled with details to configure the queue. Details filled are as follow:

- Queue Type – Standard Queue
- Queue Name – HalifaxDineSQS
- Visibility Timeout – 30 Minutes (Max time set to process & delete a message from a queue)
- Tags
    - assignment – CSCI5410-A5-PartB
    - developer – Dhrumil-Amish-Shah
    - organization – HalifaxDine


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
Support
Amazon SQS > Queues > Create queue
Info
Cancel
Create queue
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

You can't change the queue type after you create a queue.

[✓ Standard] At-least-once delivery, message ordering isn't preserved
- At-least once delivery
- Best-effort ordering

FIFO First-in-first-out delivery, message ordering is preserved
- First-in-first-out delivery
- Exactly-once processing

Name: HalifaxDineSQS
A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Configuration
Set the maximum message size, visibility to other consumers, and message retention.

Visibility timeout: 1800 Seconds	Message retention period: 4 Days
Should be between 0 seconds and 12 hours.	Should be between 1 minute and 14 days.
Delivery delay: 0 Seconds	Maximum message size: 256 KB
Should be between 0 seconds and 15 minutes.	Should be between 1 KB and 256 KB.
Receive message wait time: 20 Seconds	
Should be between 0 and 20 seconds.	

Access policy
Define who can access your queue.

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

Define who can send messages to the queue
[✓ Only the queue owner] Only the owner of the queue can send messages to the queue.
Only the specified AWS accounts, IAM users and roles — Only the specified AWS account IDs, IAM users and roles can send messages to the queue.

Define who can receive messages from the queue
[✓ Only the queue owner] Only the owner of the queue can receive messages from the queue.
Only the specified AWS accounts, IAM users and roles — Only the specified AWS account IDs, IAM users and roles can receive messages from the queue.

JSON (read-only)

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__owner_statement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "106822128233"
      },
      "Action": [
        "SQS:*"
      ],
      "Resource": "arn:aws:sqs:us-east-
1:106822128233:HalifaxDineSQS"
```

Redrive allow policy - *Optional*
Identify which source queues can use this queue as the dead-letter queue.
Select which source queues can use this queue as the dead-letter queue.
[✓ Disabled] Enabled

Encryption - *Optional*
Amazon SQS provides in-transit encryption by default. To add at-rest encryption to your queue, enable server-side encryption.
Server-side encryption
[✓ Disabled] Enabled

Dead-letter queue - *Optional*
Send undeliverable messages to a dead-letter queue.
Set this queue to receive undeliverable messages.
[✓ Disabled] Enabled

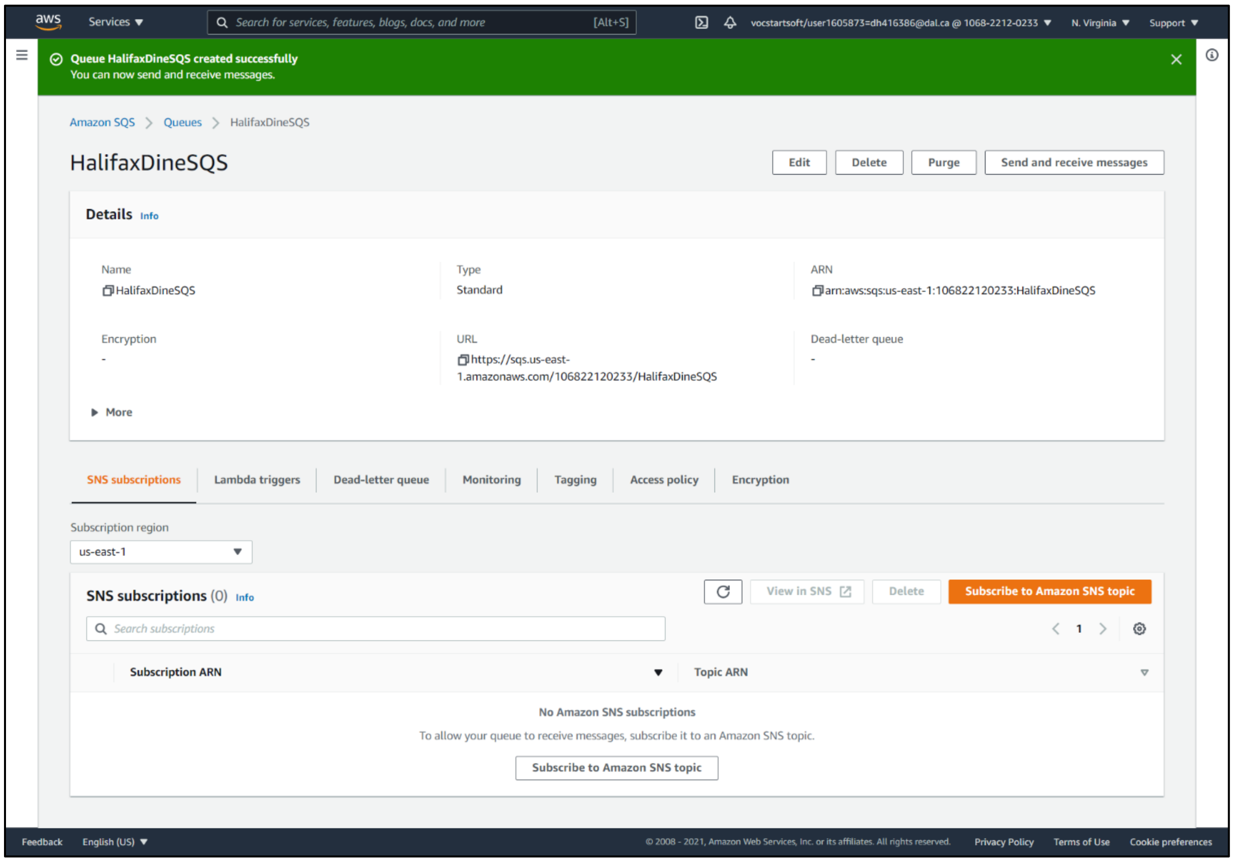
Tags - *Optional*
A tag is a label assigned to an AWS resource. Use tags to search and filter your resources or track your AWS costs. Learn more

Key	Value - optional	
developer	Dhrumil-Amish-Shah	Remove
assignment	CSCI5410-A5-PartB	Remove
organization	HalifaxDine	Remove

Add new tag
You can add 47 more tags.

Figure 3 – Create Queue page with details filled [2]

Figures 4 and 5 display screenshots of the successful creation of the HalifaxDineSQS Standard queue. When an order is created, it will be added to this queue. Successful queue creation displays the topic ARN **arn:aws:sqs:us-east-1:106822120233:HalifaxDineSQS**



Queue HalifaxDineSQS created successfully
You can now send and receive messages.

Amazon SQS > Queues > HalifaxDineSQS

HalifaxDineSQS

Details Info

Name: HalifaxDineSQS
Type: Standard
ARN: arn:aws:sqs:us-east-1:106822120233:HalifaxDineSQS
Encryption: -
URL: https://sqs.us-east-1.amazonaws.com/106822120233/HalifaxDineSQS
Dead-letter queue: -
More

Subscription region: us-east-1

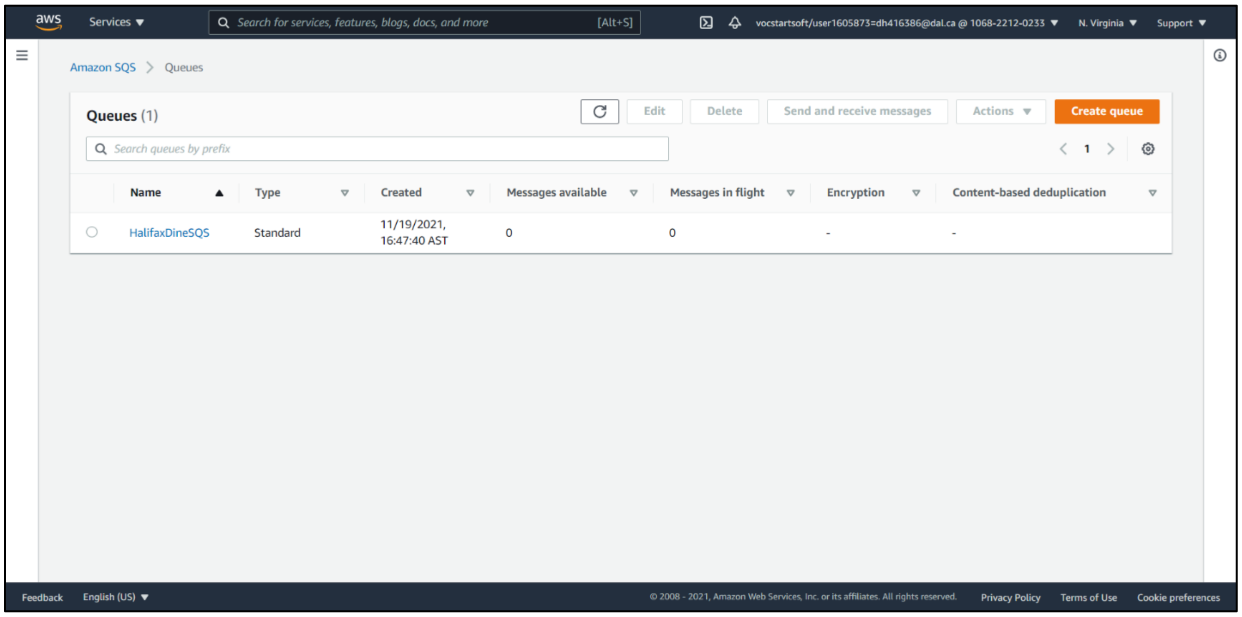
SNS subscriptions (0) Info

Subscription ARN	Topic ARN

No Amazon SNS subscriptions
To allow your queue to receive messages, subscribe it to an Amazon SNS topic.

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 4 – Successful creation of HalifaxDineSQS queue (Dashboard) [2]



Amazon SQS > Queues

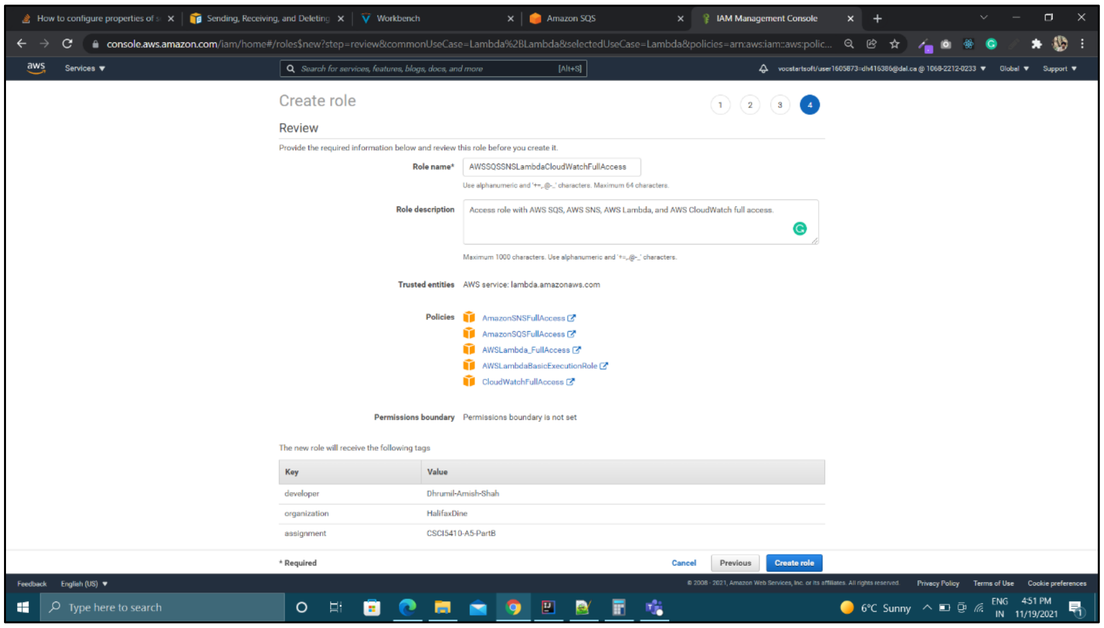
Queues (1)

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
HalifaxDineSQS	Standard	11/19/2021, 16:47:40 AST	0	0	-	-

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 5 – Successful creation of HalifaxDineSQS queue (Listing page) (contd.) [2]

Figures 6 and 7 display screenshots of the successful creation of the IAM role – AWSSQSSNSLambdaCloudWatchFullAccess that allows full access to SQS, SNS, Lambda, and CloudWatch services by AWS.



Create role

Review

Provide the required information below and review this role before you create it.

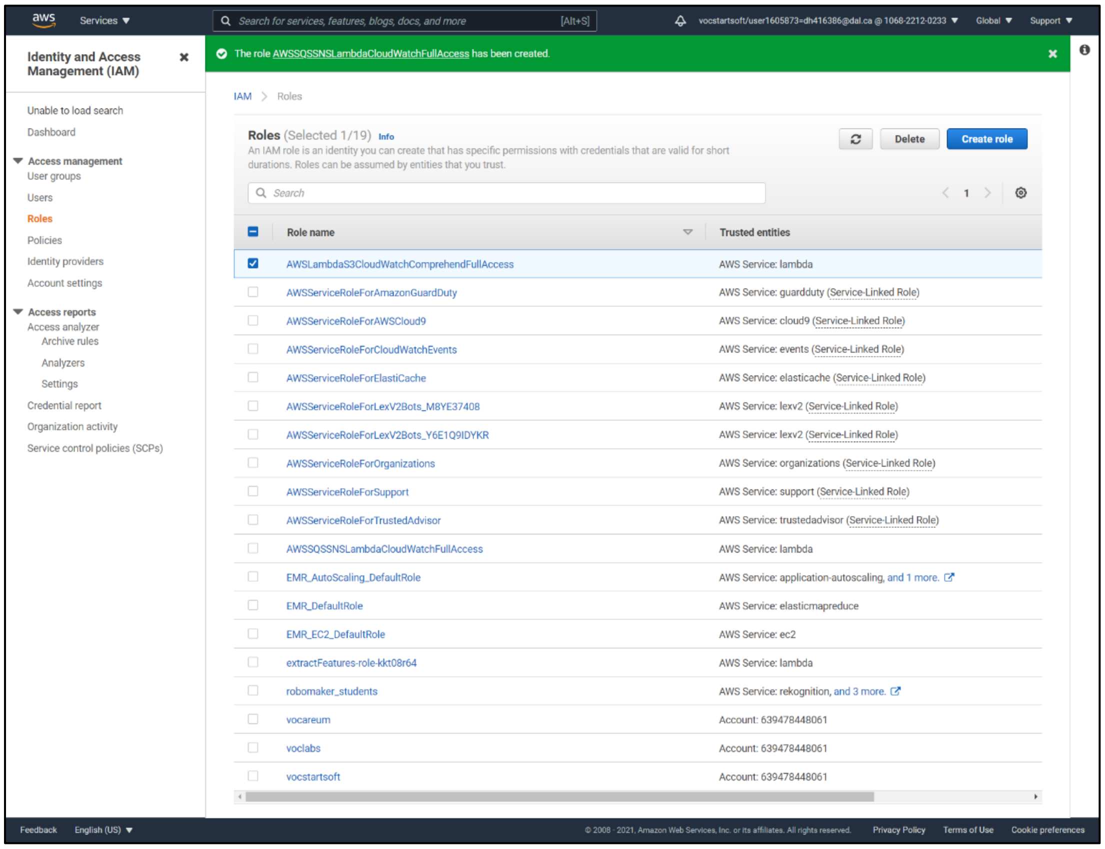
Role name*	AWSSQSSNSLambdaCloudWatchFullAccess
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Access role with AWS SQS, AWS SNS, AWS Lambda, and AWS CloudWatch full access.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: lambda.amazonaws.com
Policies	AmazonSNSFullAccess AmazonSQSFullAccess AWSLambda_FullAccess AWSLambdaBasicExecutionRole CloudWatchFullAccess
Permissions boundary	Permissions boundary is not set

The new role will receive the following tags

Key	Value
developer	Dhrumil-Amish-Shah
organization	HalifaxDine
assignment	CSCI5410-A5-PartB

* Required

Figure 6 – Successful creation of IAM role with full access to SQS, SNS, Lambda and CloudWatch [5]



The role AWSSQSSNSLambdaCloudWatchFullAccess has been created.

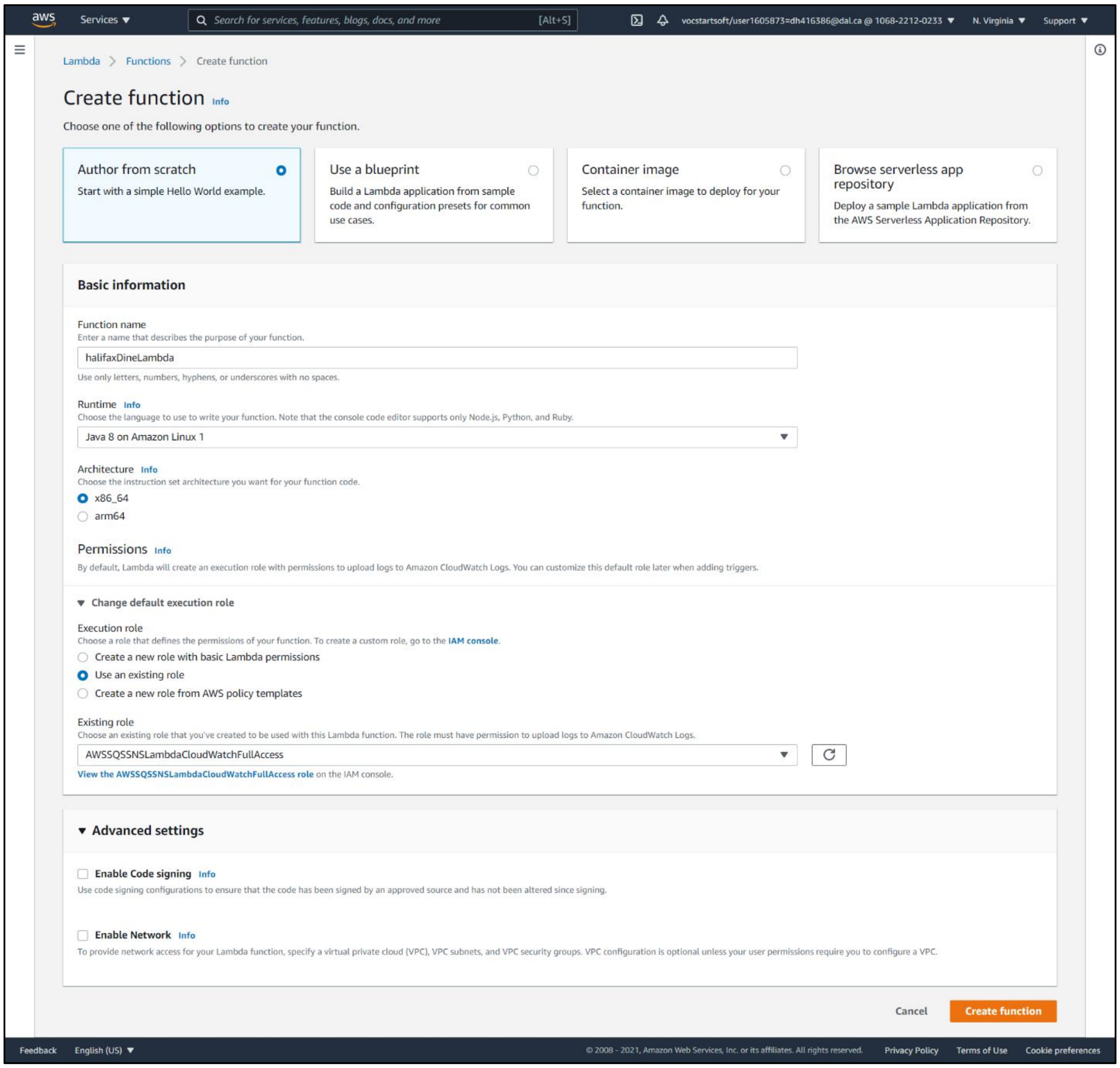
Roles (Selected 1/19)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities
✓ AWSLambdaS3CloudWatchComprehendFullAccess	AWS Service: lambda
AWSServiceRoleForAmazonGuardDuty	AWS Service: guardduty (Service-Linked Role)
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)
AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked Role)
AWSServiceRoleForElastiCache	AWS Service: elasticache (Service-Linked Role)
AWSServiceRoleForLexV2Bots_M8YE37408	AWS Service: lexv2 (Service-Linked Role)
AWSServiceRoleForLexV2Bots_Y6E1Q9IDYKR	AWS Service: lexv2 (Service-Linked Role)
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)
AWSSQSSNSLambdaCloudWatchFullAccess	AWS Service: lambda
EMR_AutoScaling_DefaultRole	AWS Service: application-autoscaling, and 1 more.
EMR_DefaultRole	AWS Service: elasticmapreduce
EMR_EC2_DefaultRole	AWS Service: ec2
extractFeatures-role-kkt08r64	AWS Service: lambda
robomaker_students	AWS Service: rekognition, and 3 more.
vocareum	Account: 639478448061
voclabs	Account: 639478448061
vocstartsoft	Account: 639478448061

Figure 7 – Successful creation of IAM role (Listing page) (contd.) [5]

Figure 8 displays the screenshot of the "Create function" page to create a Lambda function. The name of the Lambda function is halifaxDineLambda. The runtime language is set to "Java 8 on Amazon Linux 1" with x86_64 architecture. This lambda function uses the IAM role – AWSSQSSNSLambdaCloudWatchFullAccess to allow full access to SQS, SNS, Lambda, and CloudWatch services by AWS.



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

- [✓ Author from scratch] Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.
- Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
halifaxDineLambda
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Java 8 on Amazon Linux 1

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
- Create a new role with basic Lambda permissions
- [✓ Use an existing role]
- Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
AWSSQSSNSLambdaCloudWatchFullAccess
View the AWSSQSSNSLambdaCloudWatchFullAccess role on the IAM console.

Advanced settings

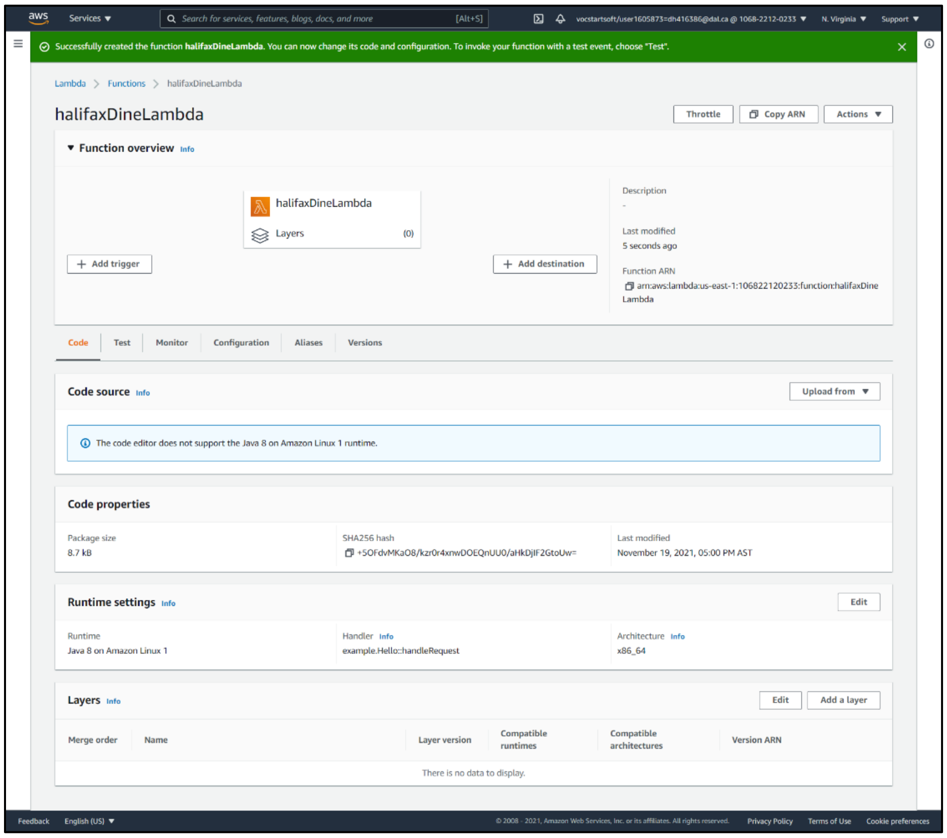
Enable Code signing Info
Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

Enable Network Info
To provide network access for your Lambda function, specify a virtual private cloud (VPC), VPC subnets, and VPC security groups. VPC configuration is optional unless your user permissions require you to configure a VPC.

Services
Search for services, features, blogs, docs, and more [Alt+S]
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
N. Virginia
Support
Cancel
Create function
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Figure 8 – Create function page. [1]

Figure 9 displays the screenshot of the successful creation of the Lambda function – halifaxDineLambda. This Lambda function is responsible to poll messages from the SQS standard queue.


Successfully created the function **halifaxDineLambda**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > halifaxDineLambda

halifaxDineLambda

Function overview

halifaxDineLambda

Layers (0)

Description
-

Last modified
5 seconds ago

Function ARN
arn:aws:lambda:us-east-1:106822120233:function:halifaxDineLambda

Code source

The code editor does not support the Java 8 on Amazon Linux 1 runtime.

Code properties

Package size	SHA256 hash	Last modified
8.7 kB	+5OFdvMKaO8/kzr0r4xnwDOEQnUU0/aHkDjIF2GtoUw=	November 19, 2021, 05:00 PM AST

Runtime settings

Runtime	Handler	Architecture
Java 8 on Amazon Linux 1	example.Hello::handleRequest	x86_64

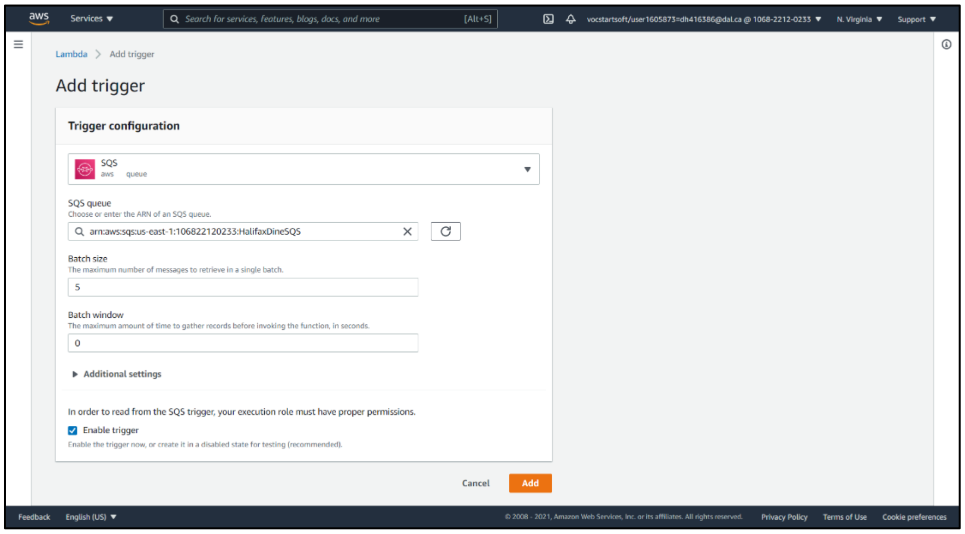
Layers

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN

There is no data to display.

Figure 9 – Successful creation of halifaxDineLambda Lambda function to poll SQS queue [1]

Figure 10 displays the screenshot of the successful configuration of the HalifaxDineSQS SQS trigger set to the halifaxDineLambda Lambda function.


Lambda > Add trigger

Add trigger

Trigger configuration

SQS
aws queue

SQS queue
Choose or enter the ARN of an SQS queue.
arn:aws:sqs:us-east-1:106822120233:HalifaxDineSQS

Batch size
The maximum number of messages to retrieve in a single batch.
5

Batch window
The maximum amount of time to gather records before invoking the function, in seconds.
0

Additional settings

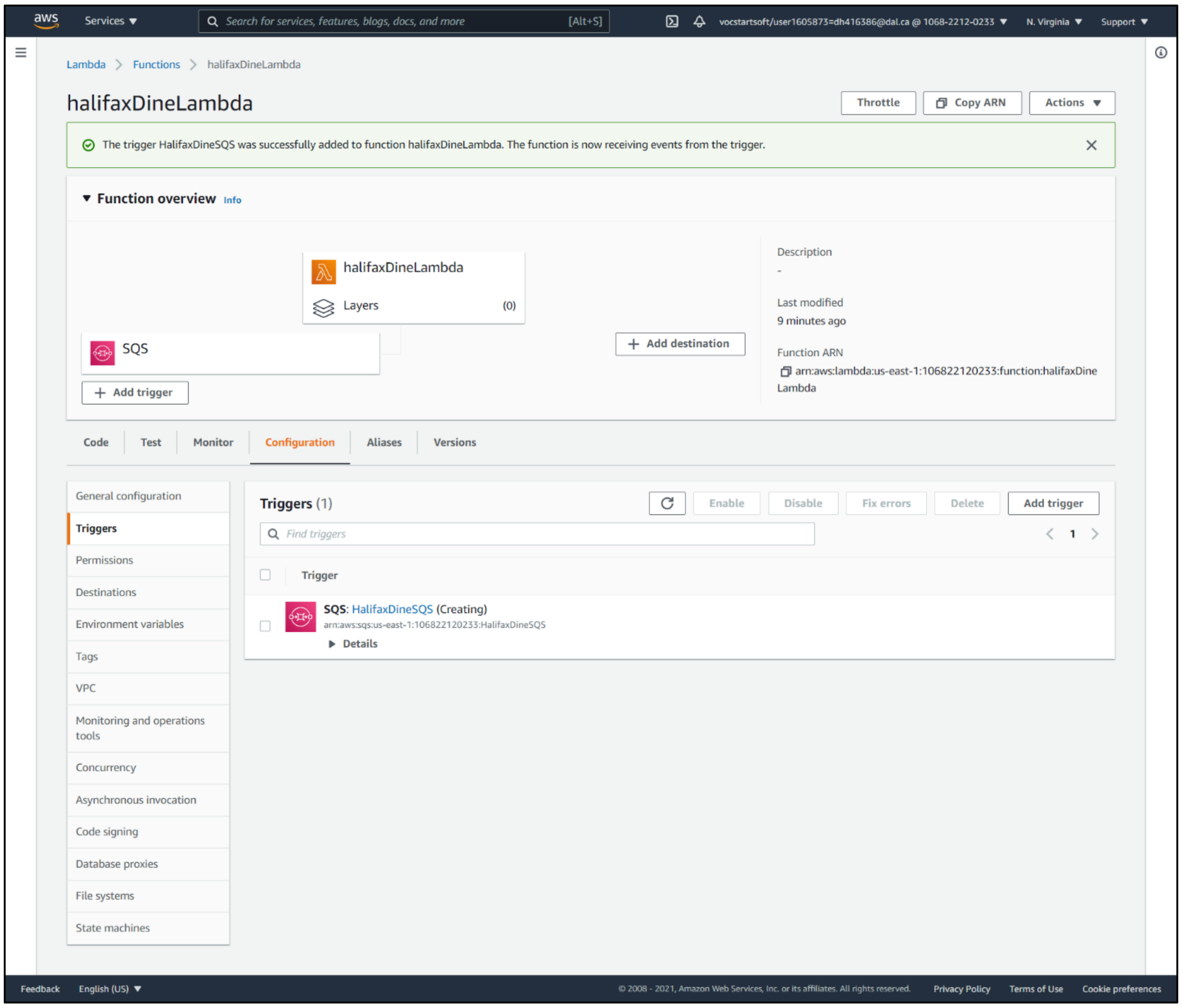
In order to read from the SQS trigger, your execution role must have proper permissions.

☑ Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
N. Virginia
Support
Cancel
Add
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Figure 10 – HalifaxDineSQS SQS trigger set to halifaxDineLambda Lambda function [1]

Figure 11 displays the screenshot of the dashboard of the halifaxDineLambda function along with the HalifaxDineSQS trigger set to it.



halifaxDineLambda

The trigger HalifaxDineSQS was successfully added to function halifaxDineLambda. The function is now receiving events from the trigger.

Description
-

Last modified
9 minutes ago

Function ARN
arn:aws:lambda:us-east-1:106822120233:function:halifaxDineLambda

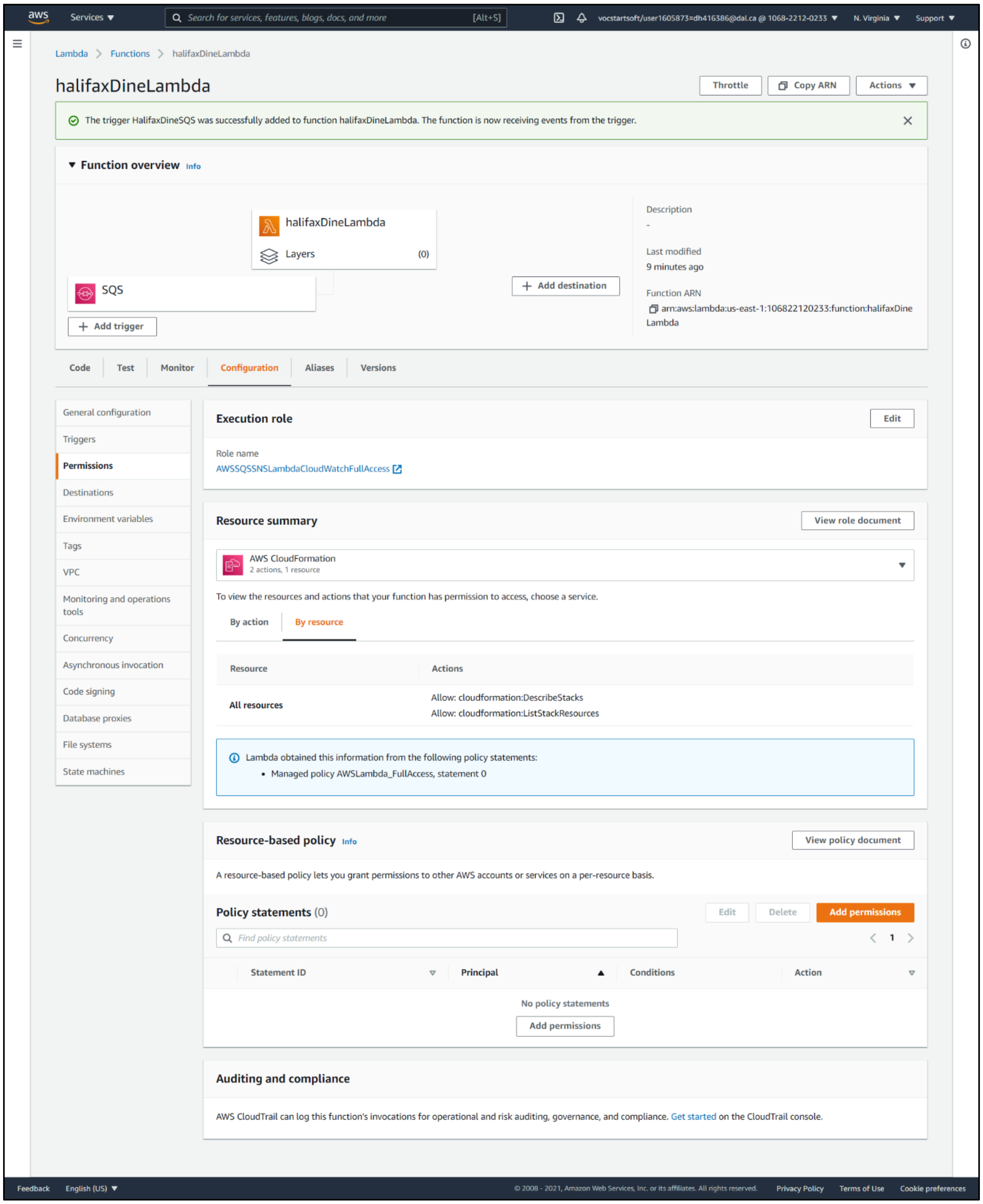
Triggers (1)

Trigger
SQS: HalifaxDineSQS (Creating)
arn:aws:sqs:us-east-1:106822120233:HalifaxDineSQS

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 11 - halifaxDineLambda Lambda function dashboard with HalifaxDineSQS SQS trigger set to it [1]

Figure 12 displays the screenshot of the halifaxDineLambda function's configuration page with SQS trigger and Execution role configured.


aws
Services ▼
Search for services, features, blogs, docs, and more [Alt+S]
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233 ▼
N. Virginia ▼
Support ▼
Lambda > Functions > halifaxDineLambda
Throttle
Copy ARN
Actions ▼

halifaxDineLambda

The trigger HalifaxDineSQS was successfully added to function halifaxDineLambda. The function is now receiving events from the trigger.

Description
-

Last modified
9 minutes ago

Function ARN
arn:aws:lambda:us-east-1:106822120233:function:halifaxDineLambda

Execution role

Role name
AWSSQSSNSLambdaCloudWatchFullAccess

Resource summary

AWS CloudFormation
2 actions, 1 resource

To view the resources and actions that your function has permission to access, choose a service.

Resource	Actions
All resources	Allow: cloudformation:DescribeStacks Allow: cloudformation:ListStackResources

Lambda obtained this information from the following policy statements:
- Managed policy AWSLambda_FullAccess, statement 0

Resource-based policy Info

A resource-based policy lets you grant permissions to other AWS accounts or services on a per-resource basis.

Policy statements (0)

Statement ID	Principal	Conditions	Action

No policy statements

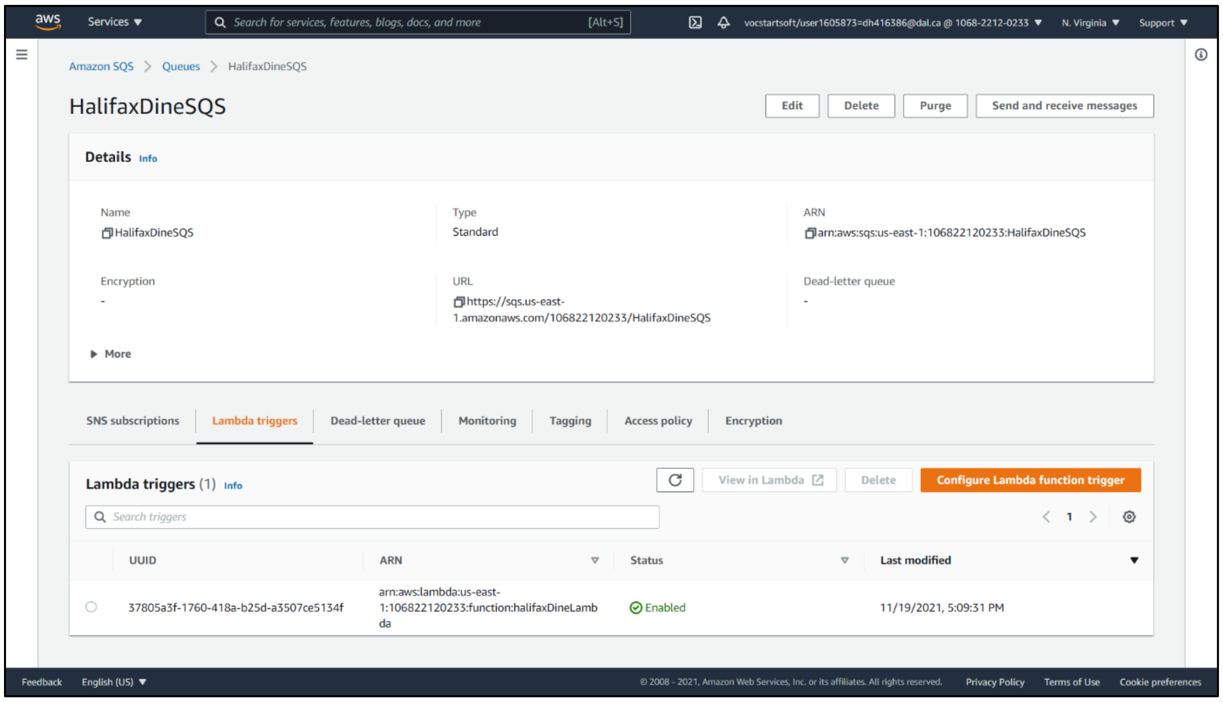
Auditing and compliance

AWS CloudTrail can log this function's invocations for operational and risk auditing, governance, and compliance. Get started on the CloudTrail console.

Feedback
English (US) ▼
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Figure 12 - halifaxDineLambda function with execution role AWSSQSSNSLambdaCloudWatchFullAccess [1]

Figure 13 displays the screenshot of the HalifaxDineSQS dashboard displaying the trigger halifaxDineLambda set to it.


Services
Search for services, features, blogs, docs, and more [Alt+S]
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
N. Virginia
Support
Amazon SQS > Queues > HalifaxDineSQS
Edit
Delete
Purge
Send and receive messages
More
SNS subscriptions
Lambda triggers
Dead-letter queue
Monitoring
Tagging
Access policy
Encryption
View in Lambda
Delete
Configure Lambda function trigger
Search triggers
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

HalifaxDineSQS

Details

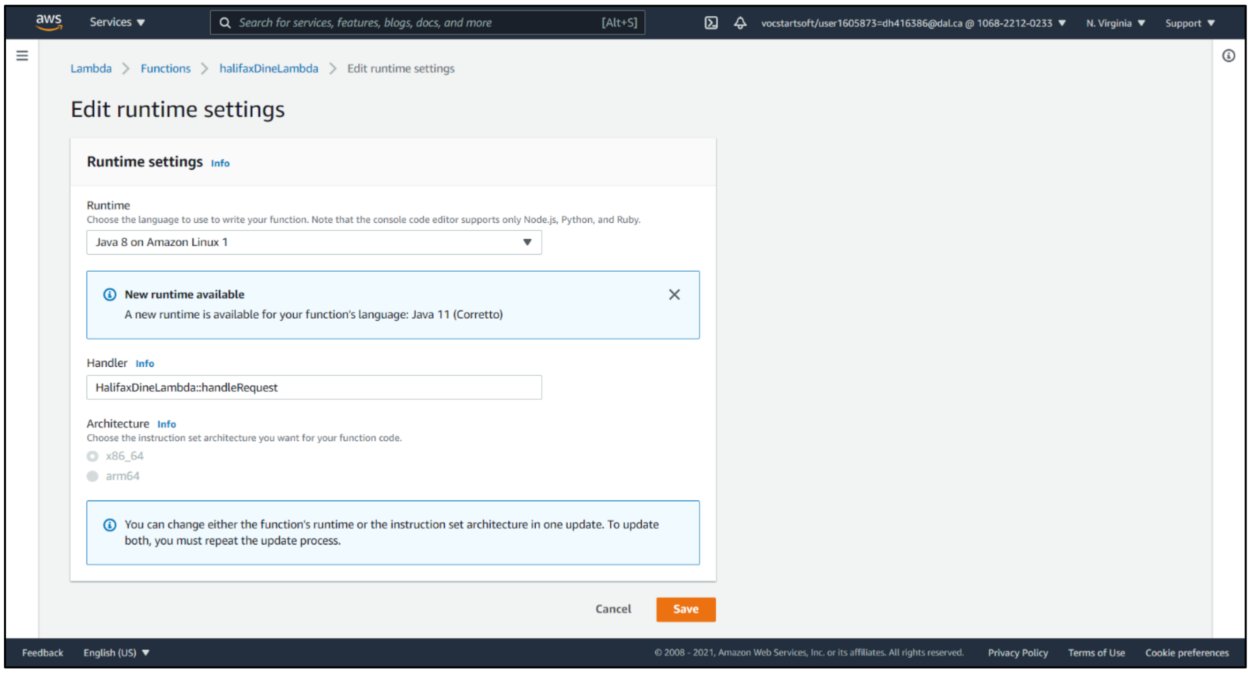
Name: HalifaxDineSQS
Type: Standard
ARN: arn:aws:sqs:us-east-1:106822120233:HalifaxDineSQS
Encryption: -
URL: https://sqs.us-east-1.amazonaws.com/106822120233/HalifaxDineSQS
Dead-letter queue: -

Lambda triggers (1)

UUID	ARN	Status	Last modified
37805a3f-1760-418a-b25d-a3507ce5134f	arn:aws:lambda:us-east-1:106822120233:function:halifaxDineLambda	Enabled	11/19/2021, 5:09:31 PM

Figure 13 - HalifaxDineSQS dashboard displaying the trigger halifaxDineLambda set to it [2]

Figure 14 displays the screenshot of the halifaxDineLambda function's "Edit runtime settings" with the handler set to HalifaxDineLambda::handleRequest.


Services
Search for services, features, blogs, docs, and more [Alt+S]
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
N. Virginia
Support
Lambda > Functions > halifaxDineLambda > Edit runtime settings
Cancel
Save
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Edit runtime settings

Runtime settings

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Java 8 on Amazon Linux 1

New runtime available
A new runtime is available for your function's language: Java 11 (Corretto)

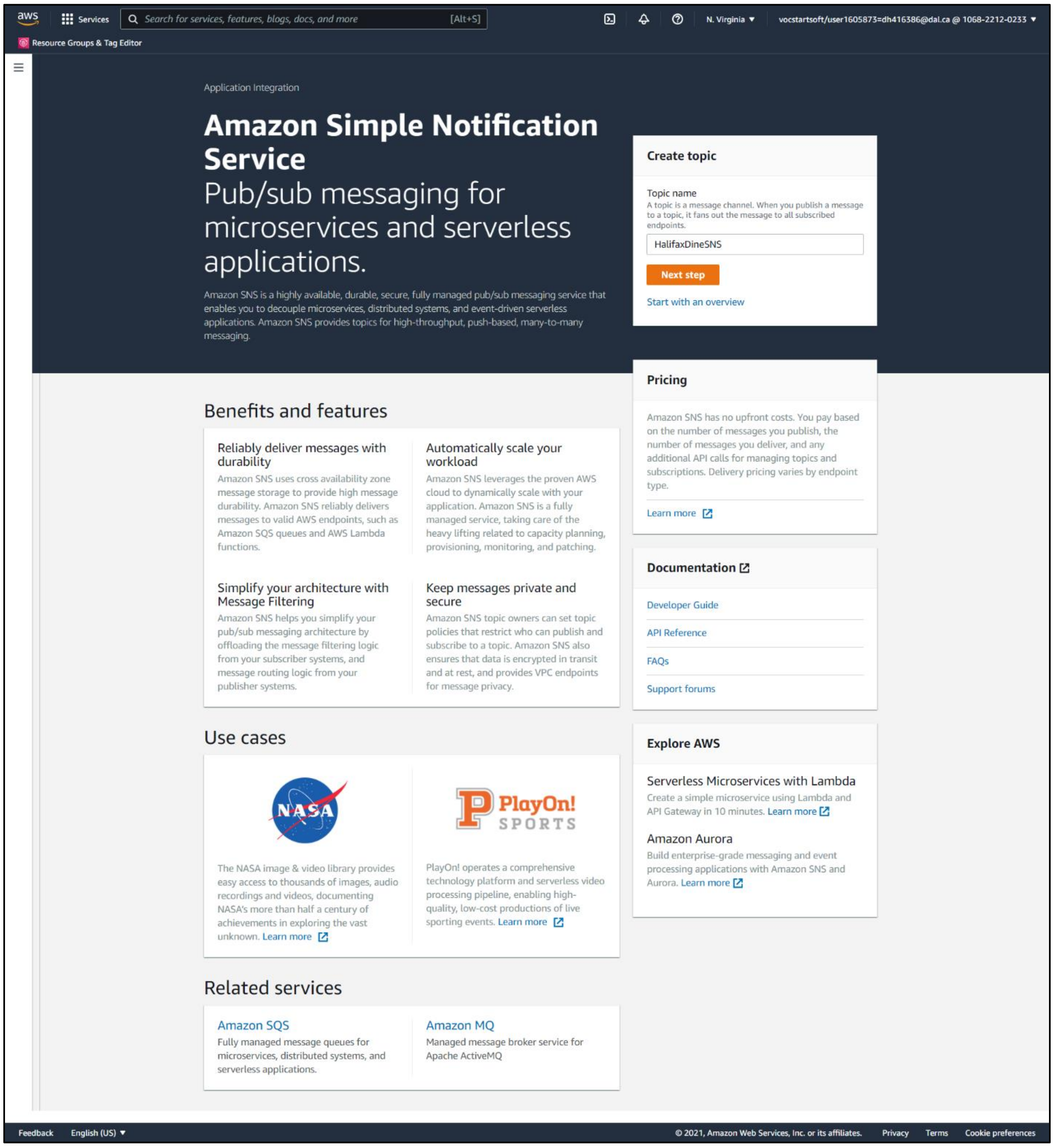
Handler
HalifaxDineLambda::handleRequest

Architecture
Choose the instruction set architecture you want for your function code.
x86_64
arm64

You can change either the function's runtime or the instruction set architecture in one update. To update both, you must repeat the update process.

Figure 14 - Handler HalifaxDineLambda::handleRequest set to the Lambda function which works as a starting point [2]

Figure 15 displays the screenshot of the Amazon SNS “Create Topic” page. I navigated to this page by typing SNS in the search bar from the navigation menu. For this application, I selected the topic name to be HalifaxDineSNS.


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
Resource Groups & Tag Editor
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.
HalifaxDineSNS
Next step
Start with an overview

Pricing

Amazon SNS has no upfront costs. You pay based on the number of messages you publish, the number of messages you deliver, and any additional API calls for managing topics and subscriptions. Delivery pricing varies by endpoint type.
Learn more

Documentation

Developer Guide
API Reference
FAQs
Support forums

Benefits and features

Reliably deliver messages with durability
Amazon SNS uses cross availability zone message storage to provide high message durability. Amazon SNS reliably delivers messages to valid AWS endpoints, such as Amazon SQS queues and AWS Lambda functions.

Automatically scale your workload
Amazon SNS leverages the proven AWS cloud to dynamically scale with your application. Amazon SNS is a fully managed service, taking care of the heavy lifting related to capacity planning, provisioning, monitoring, and patching.

Simplify your architecture with Message Filtering
Amazon SNS helps you simplify your pub/sub messaging architecture by offloading the message filtering logic from your subscriber systems, and message routing logic from your publisher systems.

Keep messages private and secure
Amazon SNS topic owners can set topic policies that restrict who can publish and subscribe to a topic. Amazon SNS also ensures that data is encrypted in transit and at rest, and provides VPC endpoints for message privacy.

Use cases

The NASA image & video library provides easy access to thousands of images, audio recordings and videos, documenting NASA's more than half a century of achievements in exploring the vast unknown. Learn more

PlayOn! operates a comprehensive technology platform and serverless video processing pipeline, enabling high-quality, low-cost productions of live sporting events. Learn more

Explore AWS

Serverless Microservices with Lambda
Create a simple microservice using Lambda and API Gateway in 10 minutes. Learn more

Amazon Aurora
Build enterprise-grade messaging and event processing applications with Amazon SNS and Aurora. Learn more

Related services

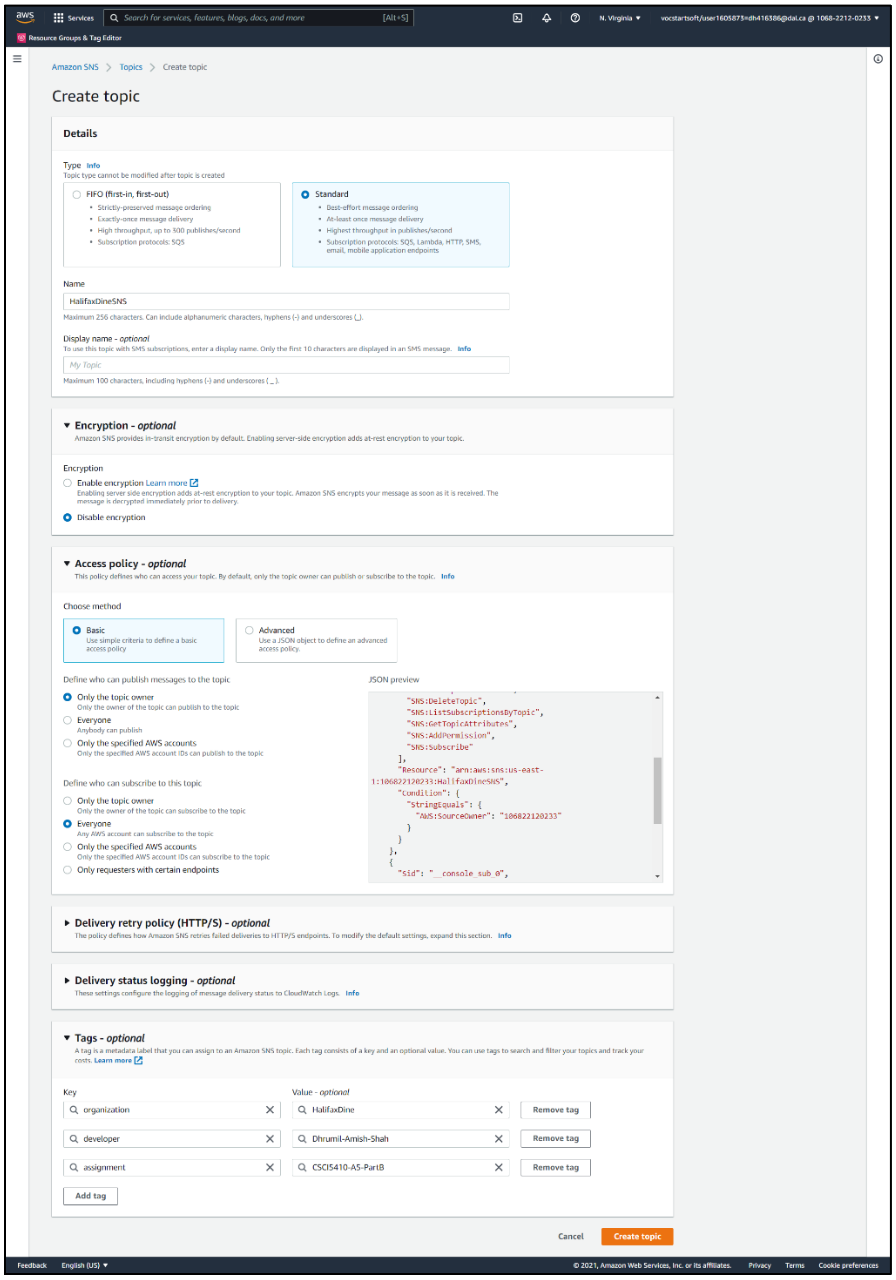
Amazon SQS
Fully managed message queues for microservices, distributed systems, and serverless applications.

Amazon MQ
Managed message broker service for Apache ActiveMQ

Figure 15 - Amazon SNS Create topic page with topic name set to HalifaxDineSNS [3]

Figure 16 displays the screenshot of the Amazon SNS Create topic page with details entered to create an SNS topic with the name HalifaxDineSNS. The type of the SNS topic is set to Standard. Details filled are:

- Access Policy Method – Basic
- Topic Subscriber – Everyone with AWS Account
- Tags
  - assignment – CSCI5410-A5-PartB
  - developer – Dhrumil-Amish-Shah
  - organization – HalifaxDine


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
Resource Groups & Tag Editor
Amazon SNS > Topics > Create topic

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
HalifaxDineSNS
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional*
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. Info
My Topic
Maximum 100 characters, including hyphens (-) and underscores (_).

Encryption - *optional*
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Encryption
- Enable encryption Learn more
  Enabling server side encryption adds at-rest encryption to your topic. Amazon SNS encrypts your message as soon as it is received. The message is decrypted immediately prior to delivery.
- [✓ Disable encryption]

Access policy - *optional*
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. Info

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

Define who can publish messages to the topic
- [✓ Only the topic owner] Only the owner of the topic can publish to the topic
- Everyone Anybody can publish
- Only the specified AWS accounts Only the specified AWS account IDs can publish to the topic

Define who can subscribe to this topic
- Only the topic owner Only the owner of the topic can subscribe to the topic
- [✓ Everyone] Any AWS account can subscribe to the topic
- Only the specified AWS accounts Only the specified AWS account IDs can subscribe to the topic
- Only requesters with certain endpoints

JSON preview
```
        "SNS:DeleteTopic",
        "SNS:ListSubscriptionsByTopic",
        "SNS:GetTopicAttributes",
        "SNS:AddPermission",
        "SNS:Subscribe"
      ],
      "Resource": "arn:aws:sns:us-east-
1:106822120233:HalifaxDineSNS",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "106822120233"
        }
      }
    },
    {
      "Sid": "__console_sub_0",
```

Delivery retry policy (HTTP/S) - *optional*
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. Info

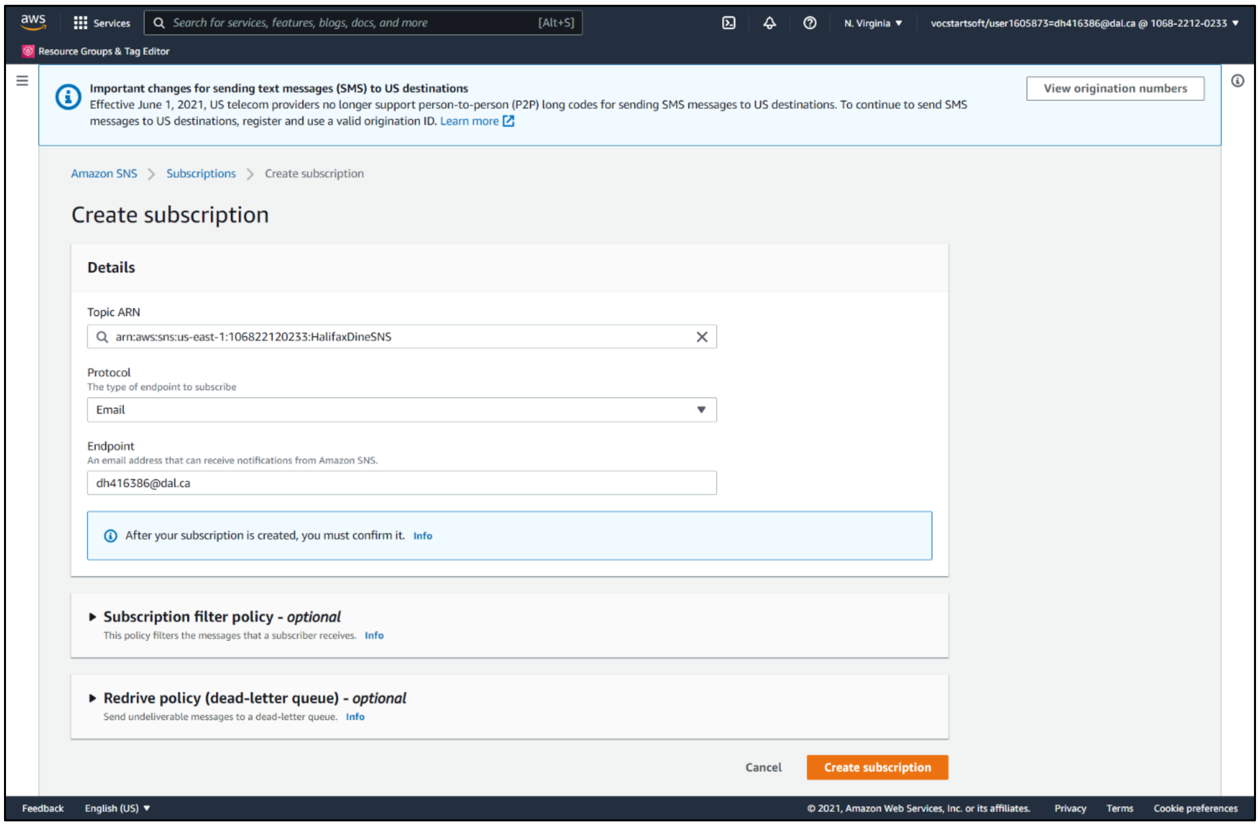
Delivery status logging - *optional*
These settings configure the logging of message delivery status to CloudWatch Logs. Info

Tags - *optional*
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. Learn more

Key	Value - *optional*
organization	HalifaxDine
developer	Dhrumil-Amish-Shah
assignment	CSCI5410-A5-PartB

Figure 16 - SNS Topic HalifaxDineSNS in the create stage [3]

Figure 17 displays the screenshot of the create subscription page for the HalifaxDineSNS topic. The protocol is set to email, and the endpoint is configured to dh416386@dal.ca


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
Resource Groups & Tag Editor
View origination numbers
Cancel
Create subscription
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Important changes for sending text messages (SMS) to US destinations
Effective June 1, 2021, US telecom providers no longer support person-to-person (P2P) long codes for sending SMS messages to US destinations. To continue to send SMS messages to US destinations, register and use a valid origination ID. Learn more

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
dh416386@dal.ca

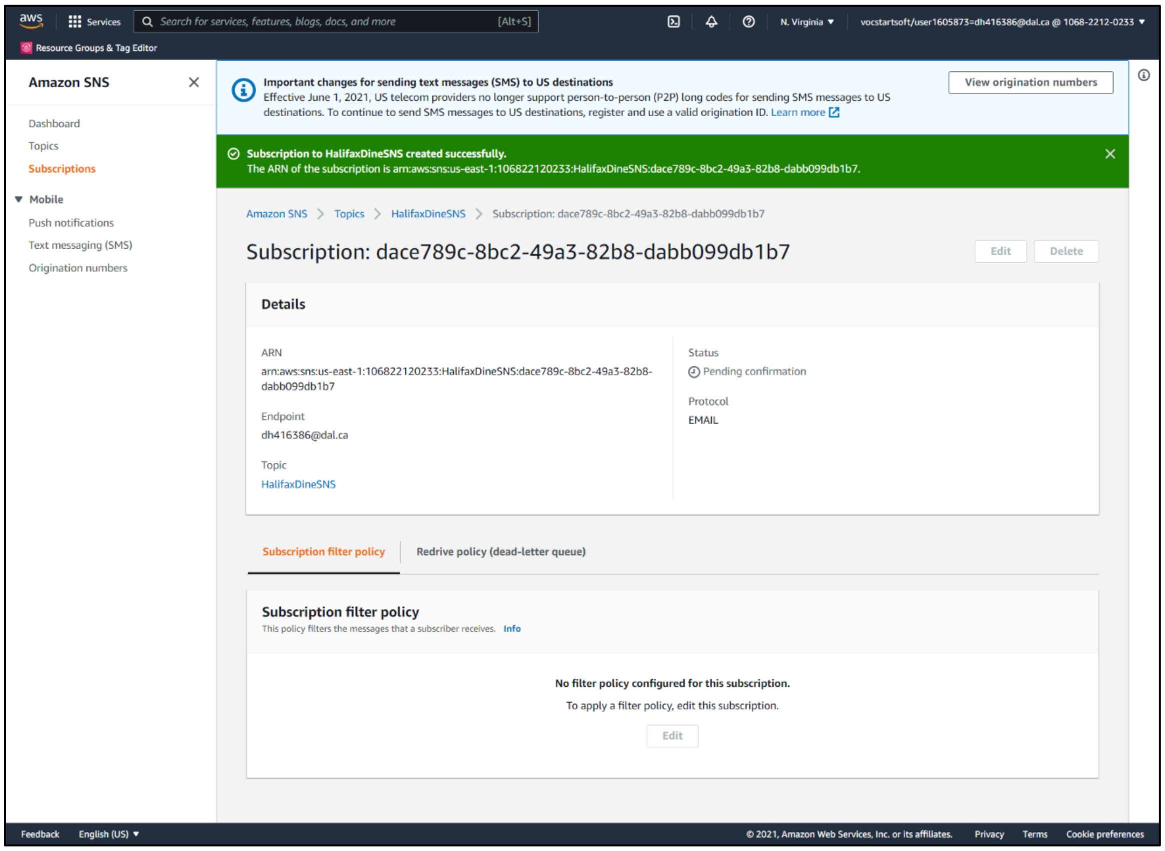
After your subscription is created, you must confirm it. Info

Subscription filter policy - *optional*
This policy filters the messages that a subscriber receives. Info

Redrive policy (dead-letter queue) - *optional*
Send undeliverable messages to a dead-letter queue. Info

Figure 17 - Create subscription page with the protocol set to Email and endpoint set to dh416386@dal.ca [3]

Figures 18 and 19 display the screenshots of the successful creation of a subscription with the status message "Pending Confirmation".


Important changes for sending text messages (SMS) to US destinations
Effective June 1, 2021, US telecom providers no longer support person-to-person (P2P) long codes for sending SMS messages to US destinations. To continue to send SMS messages to US destinations, register and use a valid origination ID. Learn more

Subscription to HalifaxDineSNS created successfully.
The ARN of the subscription is arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS:dace789c-8bc2-49a3-82b8-dabb099db1b7.

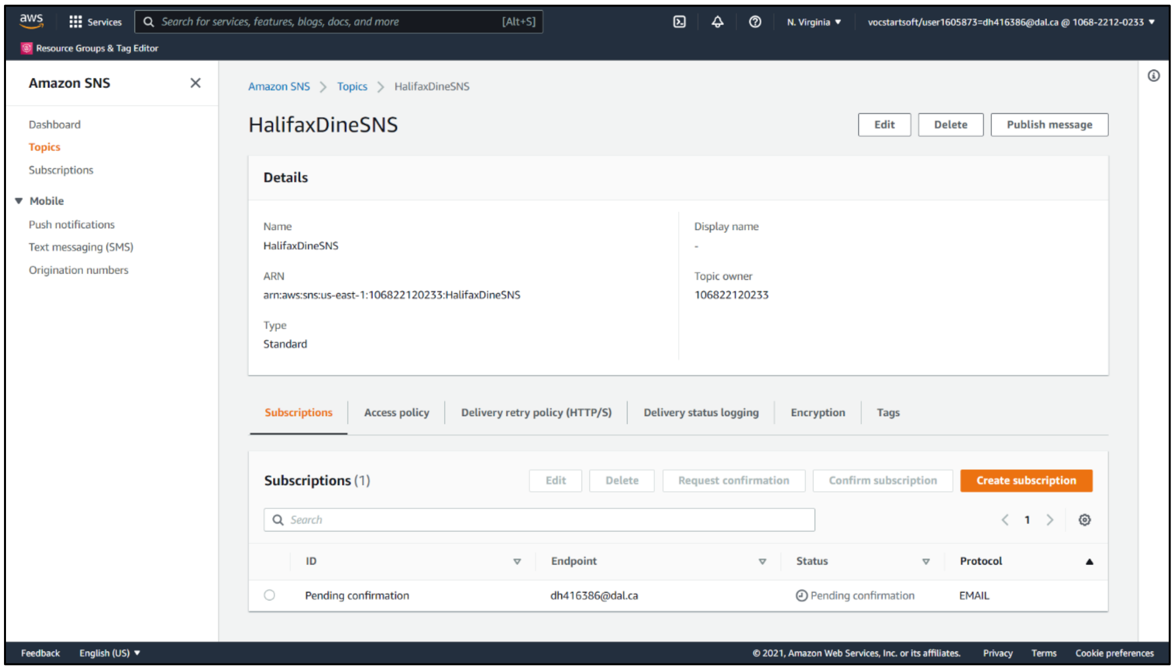
Subscription: dace789c-8bc2-49a3-82b8-dabb099db1b7

Details

ARN: arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS:dace789c-8bc2-49a3-82b8-dabb099db1b7
Endpoint: dh416386@dal.ca
Topic: HalifaxDineSNS
Status: Pending confirmation
Protocol: EMAIL

Subscription filter policy
This policy filters the messages that a subscriber receives.
No filter policy configured for this subscription.
To apply a filter policy, edit this subscription.

Figure 18 - Successful creation of a subscription with status "Pending Confirmation" [3]


HalifaxDineSNS

Details

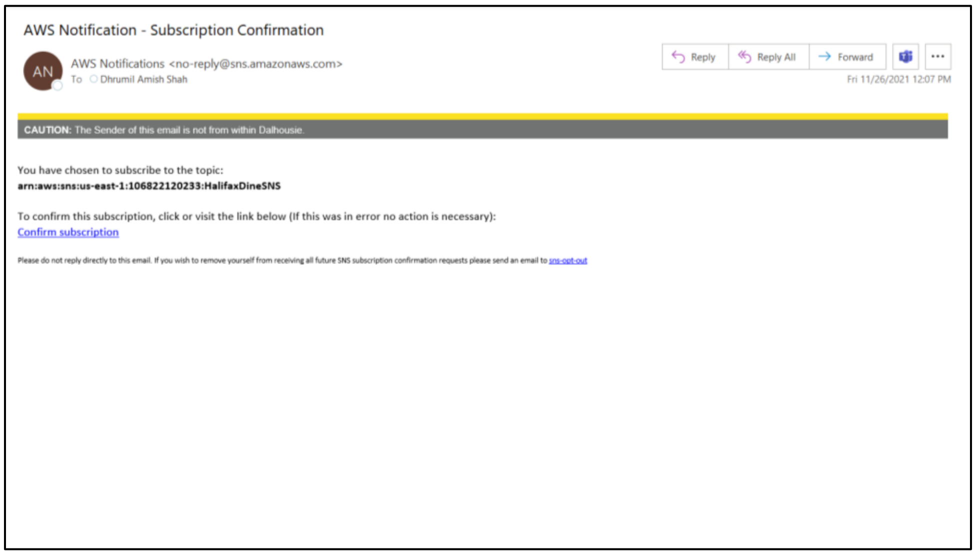
Name: HalifaxDineSNS
ARN: arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS
Type: Standard
Display name: -
Topic owner: 106822120233

Subscriptions (1)

ID	Endpoint	Status	Protocol
Pending confirmation	dh416386@dal.ca	Pending confirmation	EMAIL

Figure 19 - Successful creation of a subscription with status "Pending Confirmation" (contd.) [3]

Figure 20 displays the screenshot of the subscription confirmation email received at the provided email address. (i.e., Email – dh416386@dal.ca)



AWS Notification - Subscription Confirmation

AWS Notifications <no-reply@sns.amazonaws.com>
To Dhrumil Amish Shah

Fri 11/26/2021 12:07 PM

CAUTION: The Sender of this email is not from within Dalhousie.

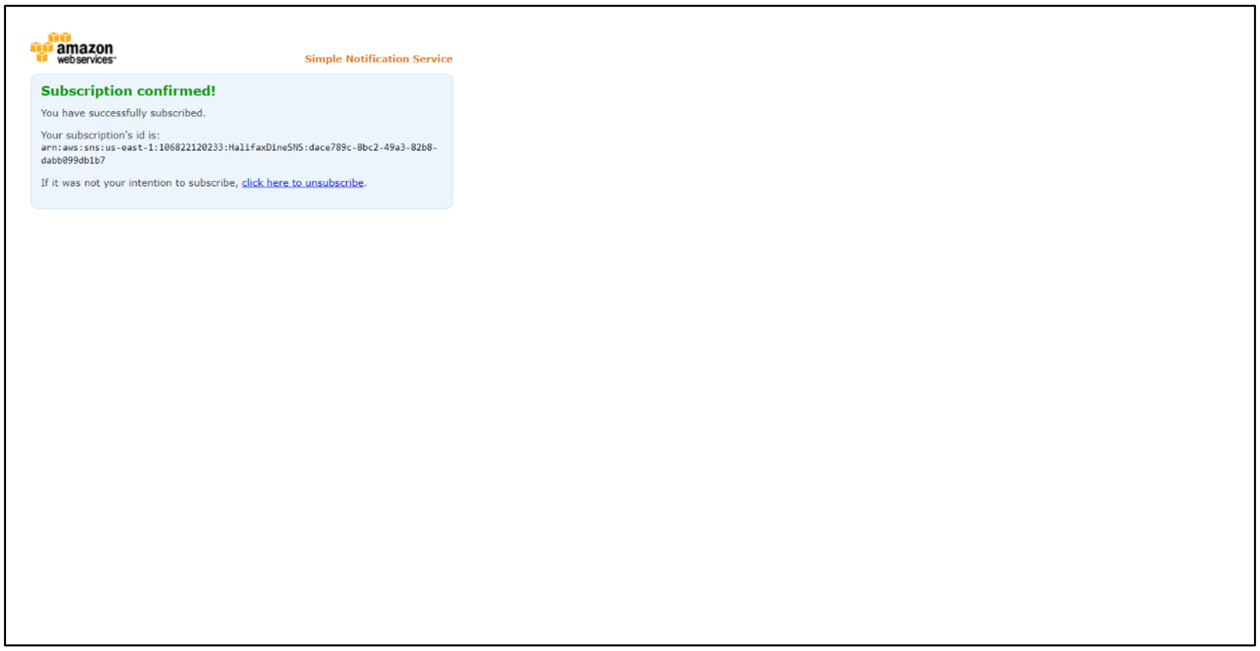
You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS

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

Figure 20 - Successful subscription confirmation message received on Email

Figure 21 displays the screenshot of the subscription confirmed page.



Subscription confirmed!

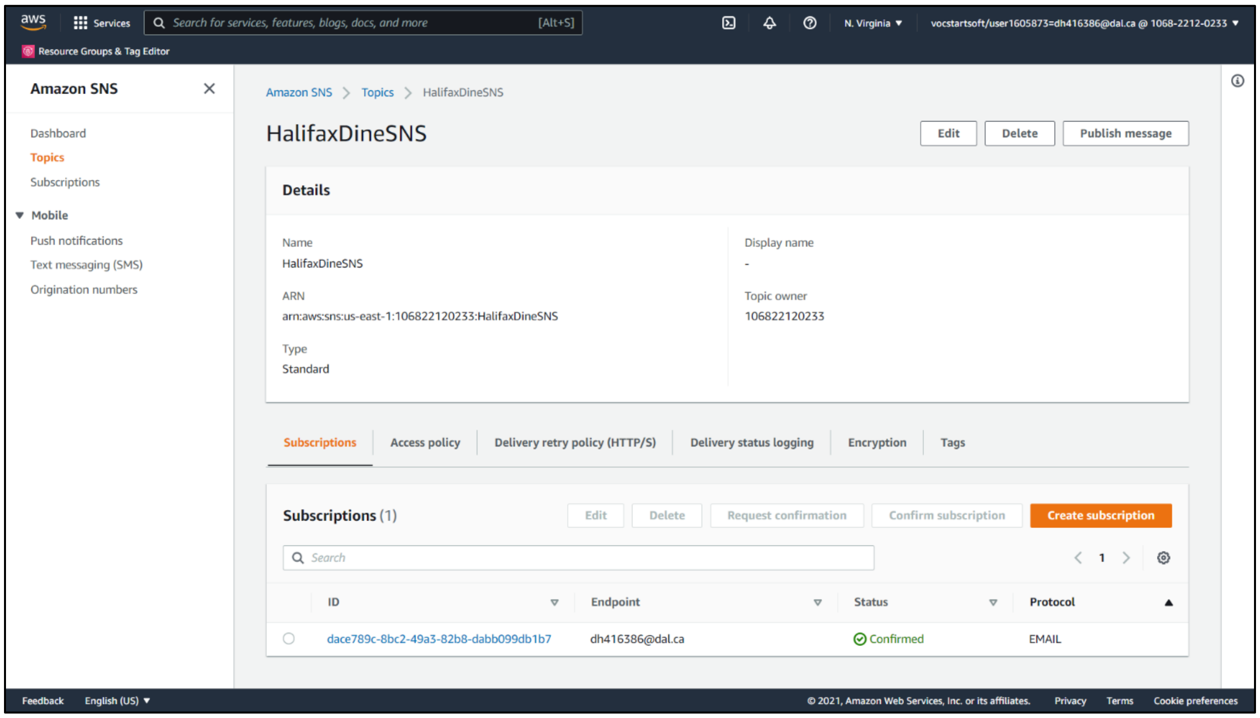
You have successfully subscribed.

Your subscription's id is:
`arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS:dace789c-8bc2-49a3-82b8-dabb099db1b7`

If it was not your intention to subscribe, click here to unsubscribe.

Figure 21 - Screenshot of subscription confirmed page

Figure 22 displays the screenshot of the HalifaxDineSNS dashboard with the subscription status Confirmed for the endpoint dh416386@dal.ca.



HalifaxDineSNS

Details

Name
HalifaxDineSNS

Display name
-

ARN
arn:aws:sns:us-east-1:106822120233:HalifaxDineSNS

Topic owner
106822120233

Type
Standard

Subscriptions (1)

ID	Endpoint	Status	Protocol
dace789c-8bc2-49a3-82b8-dabb099db1b7	dh416386@dal.ca	Confirmed	EMAIL

Figure 22 - Screenshot of HalifaxDineSNS dashboard with subscription status confirmed [3]

Program Execution Output

Figures 23 and 24 display the screenshots of program execution. Orders are processed in batches. Batch 1 consists of 5 orders with different order items. The details about each order are as below:

- Order 1 (Order ID - a05ba993-bd0a-4f75-a9a1-ef7352d14431) contains 3 different dishes with different quantities.
- Order 2 (Order ID - f4e41e47-f55e-46a2-bf57-29a3ec44737b) contains 1 dish.
- Order 3 (Order ID - 61f909a3-6bad-41e8-802f-121c88121837) contains 2 different dishes with different quantities.
- Order 4 (Order ID - 09788cf3-e16f-461d-a22b-8fb05480f268) contains 1 dish.
- Order 5 (Order ID - 56b50866-5d9a-4985-b2f0-518a092216d7) contains 3 different dishes with different quantities.

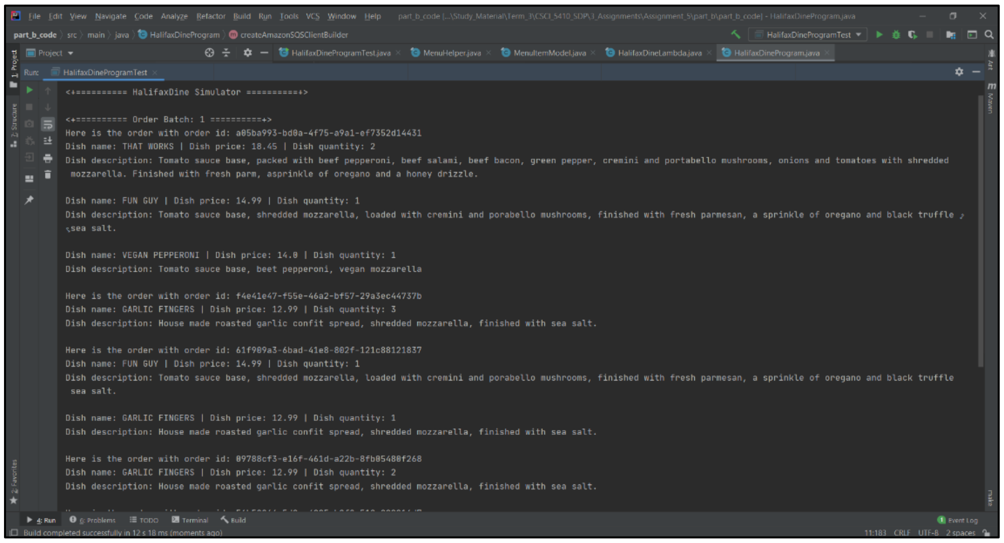

```
<+========== HalifaxDine Simulator ==========+>

<+========== Order Batch: 1 ==========+>
Here is the order with order id: a05ba993-bd0a-4f75-a9a1-ef7352d14431
Dish name: THAT WORKS | Dish price: 18.45 | Dish quantity: 2
Dish description: Tomato sauce base, packed with beef pepperoni, beef salami, beef bacon, green pepper, cremini and portabello mushrooms, onions and tomatoes with shredded
 mozzarella. Finished with fresh parm, asprinkle of oregano and a honey drizzle.

Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1
Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black truffle
 sea salt.

Dish name: VEGAN PEPPERONI | Dish price: 14.0 | Dish quantity: 1
Dish description: Tomato sauce base, beet pepperoni, vegan mozzarella

Here is the order with order id: f4e41e47-f55e-46a2-bf57-29a3ec44737b
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 61f909a3-6bad-41e8-802f-121c88121837
Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1
Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black truffle
 sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 1
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 09788cf3-e16f-461d-a22b-8fb05480f268
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 2
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.
```

Figure 23 - Program execution output

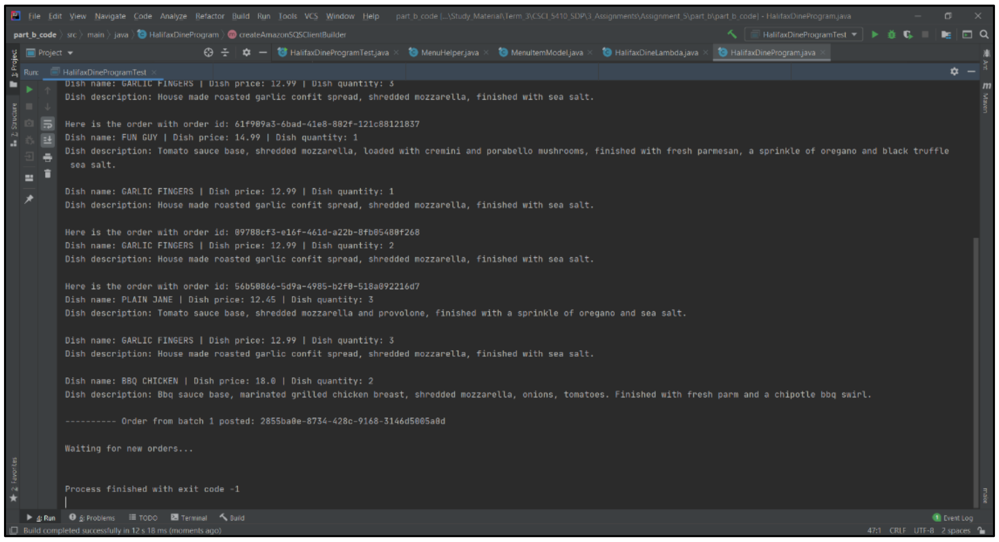
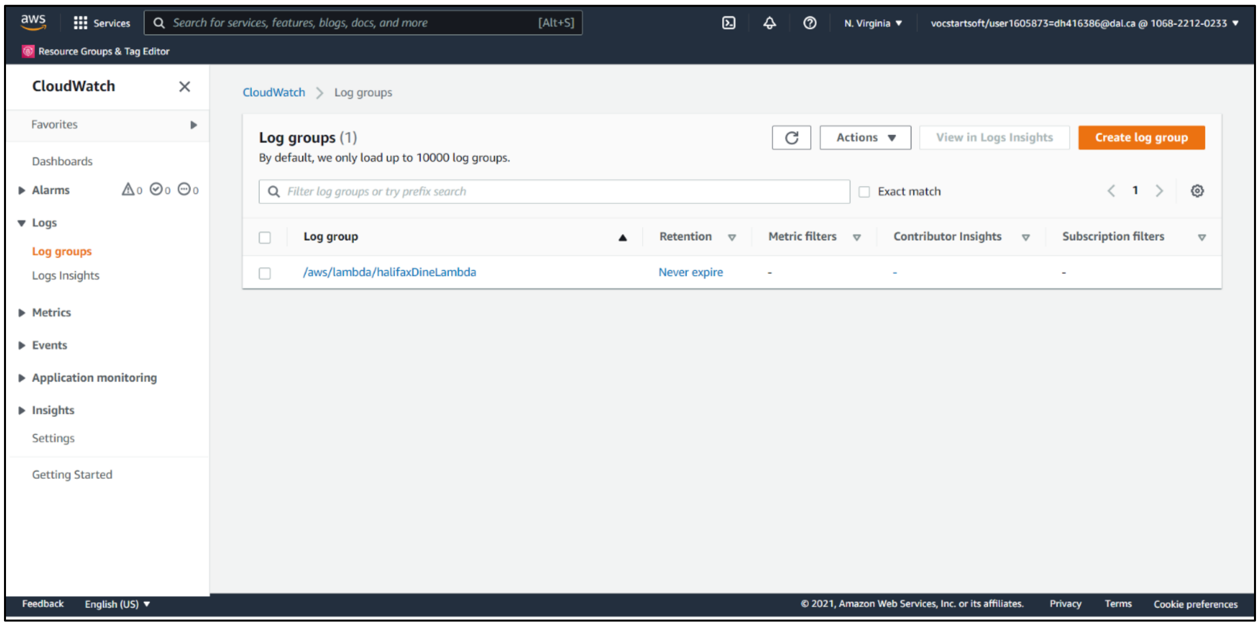

```
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 61f909a3-6bad-41e8-802f-121c88121837
Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1
Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black truffle
 sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 1
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 09788cf3-e16f-461d-a22b-8fb05480f268
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 2
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 56b50866-5d9a-4985-b2f0-518a092216d7
Dish name: PLAIN JANE | Dish price: 12.45 | Dish quantity: 3
Dish description: Tomato sauce base, shredded mozzarella and provolone, finished with a sprinkle of oregano and sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Dish name: BBQ CHICKEN | Dish price: 18.0 | Dish quantity: 2
Dish description: Bbq sauce base, marinated grilled chicken breast, shredded mozzarella, onions, tomatoes. Finished with fresh parm and a chipotle bbq swirl.

---------- Order from batch 1 posted: 2855ba0e-8734-428c-9168-3146d5005a0d

Waiting for new orders...


Process finished with exit code -1
```

Figure 24 - Program execution output (contd.)

Figures 25 and 26 display the screenshots of log groups created by the halifaxDineLambda function on CloudWatch.



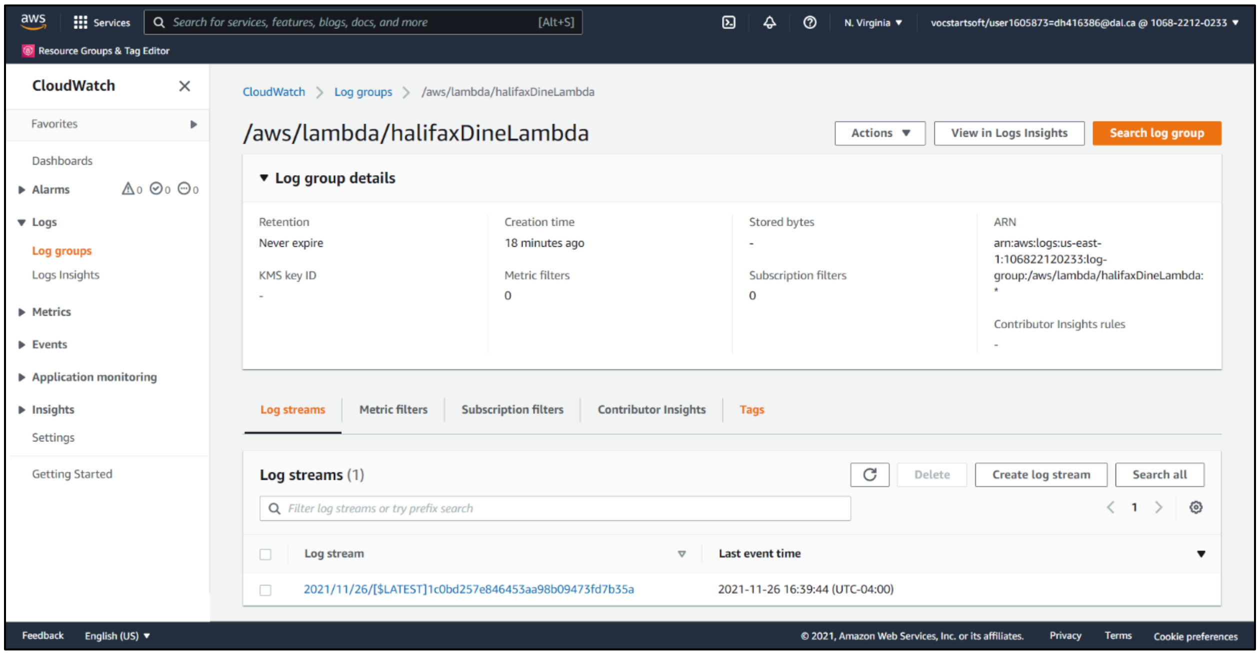
Log groups (1)

By default, we only load up to 10000 log groups.

Filter log groups or try prefix search

Log group	Retention	Metric filters	Contributor Insights	Subscription filters
/aws/lambda/halifaxDineLambda	Never expire	-	-	-

Figure 25 – Log groups for the halifaxDineLambda function on CloudWatch [6]



/aws/lambda/halifaxDineLambda

Log group details

Retention Never expire	Creation time 18 minutes ago	Stored bytes -	ARN arn:aws:logs:us-east-1:106822120233:log-group:/aws/lambda/halifaxDineLambda:*
KMS key ID -	Metric filters 0	Subscription filters 0	Contributor Insights rules -

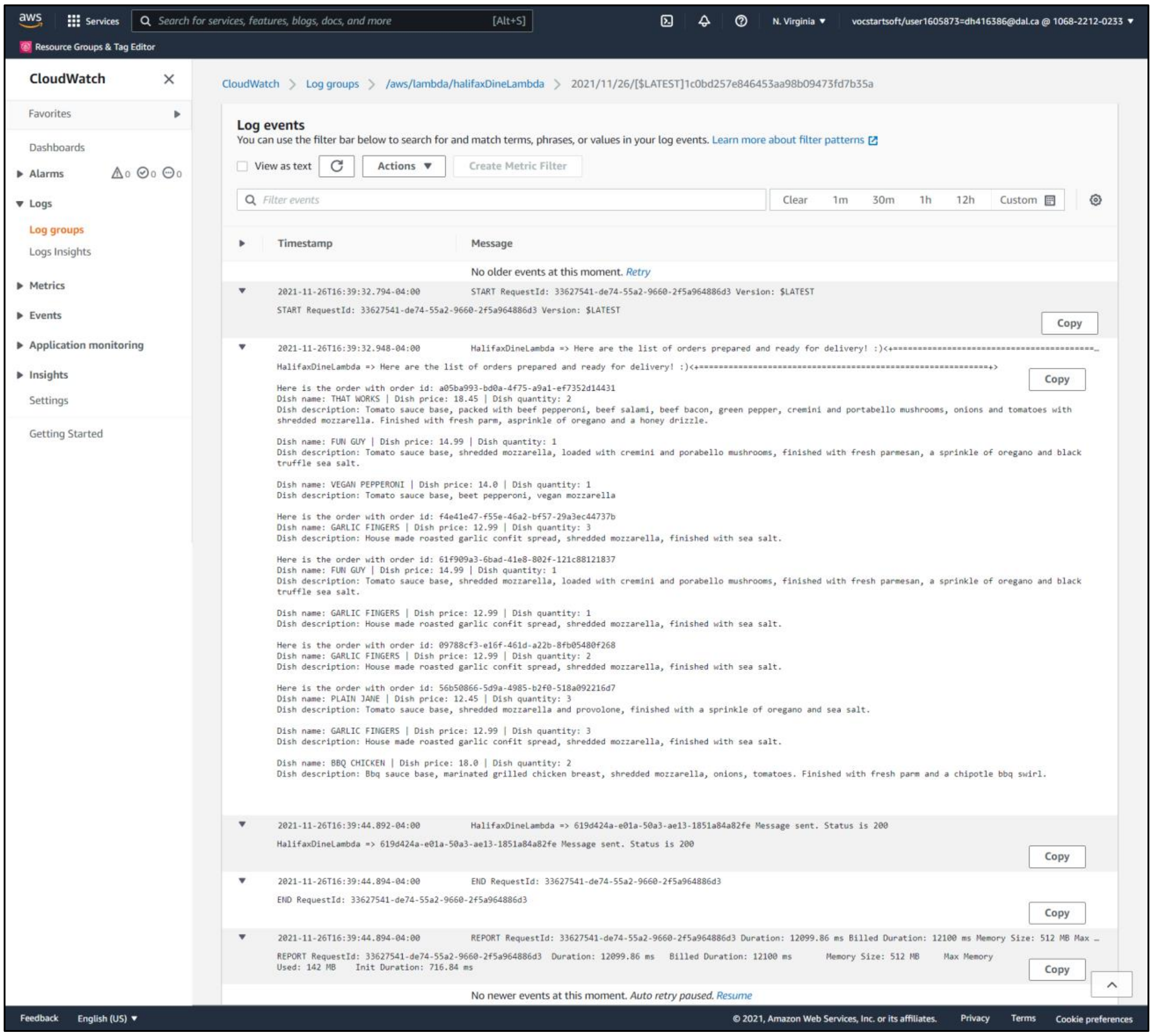
Log streams (1)

Filter log streams or try prefix search

Log stream	Last event time
2021/11/26/[$LATEST]1c0bd257e846453aa98b09473fd7b35a	2021-11-26 16:39:44 (UTC-04:00)

Figure 26 - Log groups for the halifaxDineLambda function on CloudWatch (contd.) [6]

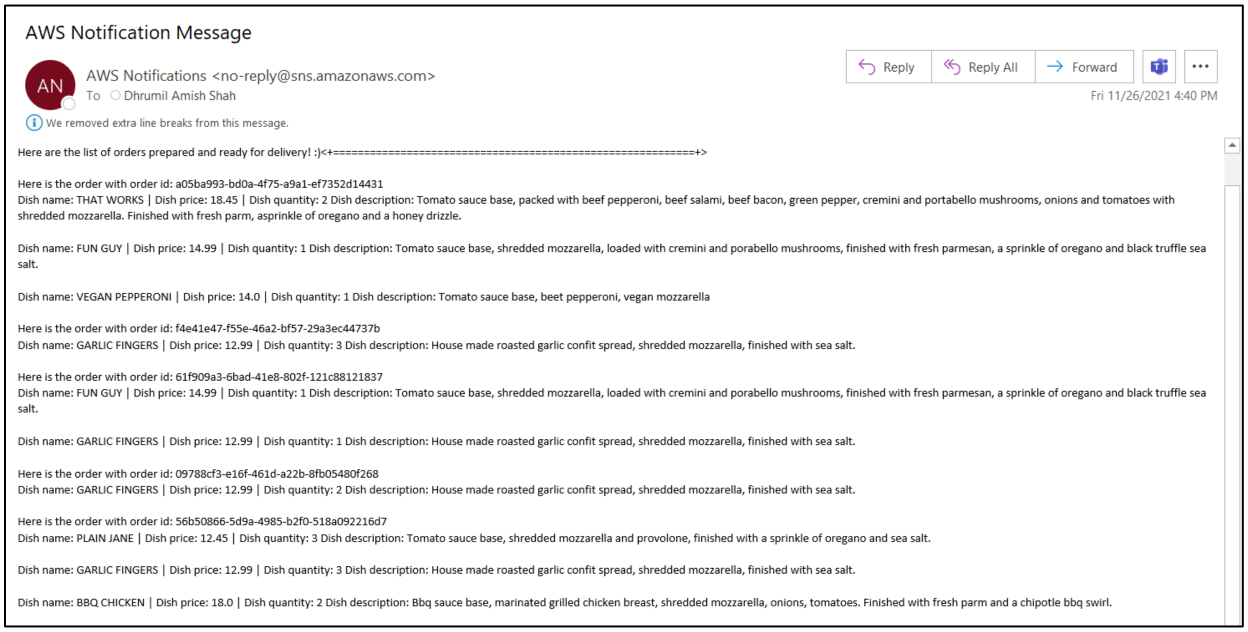
Figure 27 displays the screenshot of logs recorded when the Lambda function halifaxDineLambda was triggered.


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
vocstartsoft/user1605873=dh416386@dal.ca @ 1068-2212-0233
Resource Groups & Tag Editor
CloudWatch
Favorites
Dashboards
Alarms
Logs
Log groups
Logs Insights
Metrics
Events
Application monitoring
Insights
Settings
Getting Started
CloudWatch > Log groups > /aws/lambda/halifaxDineLambda > 2021/11/26/[$LATEST]1c0bd257e846453aa98b09473fd7b35a
Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
View as text
Actions
Create Metric Filter
Filter events
Clear 1m 30m 1h 12h Custom
Timestamp
Message
Copy
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

```
No older events at this moment. Retry

2021-11-26T16:39:32.794-04:00    START RequestId: 33627541-de74-55a2-9660-2f5a964886d3 Version: $LATEST
START RequestId: 33627541-de74-55a2-9660-2f5a964886d3 Version: $LATEST

2021-11-26T16:39:32.948-04:00    HalifaxDineLambda => Here are the list of orders prepared and ready for delivery! :)<+=======================================…
HalifaxDineLambda => Here are the list of orders prepared and ready for delivery! :)<+============================================================+>

Here is the order with order id: a05ba993-bd0a-4f75-a9a1-ef7352d14431
Dish name: THAT WORKS | Dish price: 18.45 | Dish quantity: 2
Dish description: Tomato sauce base, packed with beef pepperoni, beef salami, beef bacon, green pepper, cremini and portabello mushrooms, onions and tomatoes with
shredded mozzarella. Finished with fresh parm, asprinkle of oregano and a honey drizzle.

Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1
Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black
truffle sea salt.

Dish name: VEGAN PEPPERONI | Dish price: 14.0 | Dish quantity: 1
Dish description: Tomato sauce base, beet pepperoni, vegan mozzarella

Here is the order with order id: f4e41e47-f55e-46a2-bf57-29a3ec44737b
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 61f909a3-6bad-41e8-802f-121c88121837
Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1
Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black
truffle sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 1
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 09788cf3-e16f-461d-a22b-8fb05480f268
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 2
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 56b50866-5d9a-4985-b2f0-518a092216d7
Dish name: PLAIN JANE | Dish price: 12.45 | Dish quantity: 3
Dish description: Tomato sauce base, shredded mozzarella and provolone, finished with a sprinkle of oregano and sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3
Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Dish name: BBQ CHICKEN | Dish price: 18.0 | Dish quantity: 2
Dish description: Bbq sauce base, marinated grilled chicken breast, shredded mozzarella, onions, tomatoes. Finished with fresh parm and a chipotle bbq swirl.

2021-11-26T16:39:44.892-04:00    HalifaxDineLambda => 619d424a-e01a-50a3-ae13-1851a84a82fe Message sent. Status is 200
HalifaxDineLambda => 619d424a-e01a-50a3-ae13-1851a84a82fe Message sent. Status is 200

2021-11-26T16:39:44.894-04:00    END RequestId: 33627541-de74-55a2-9660-2f5a964886d3
END RequestId: 33627541-de74-55a2-9660-2f5a964886d3

2021-11-26T16:39:44.894-04:00    REPORT RequestId: 33627541-de74-55a2-9660-2f5a964886d3 Duration: 12099.86 ms Billed Duration: 12100 ms Memory Size: 512 MB Max …
REPORT RequestId: 33627541-de74-55a2-9660-2f5a964886d3  Duration: 12099.86 ms   Billed Duration: 12100 ms      Memory Size: 512 MB     Max Memory
Used: 142 MB    Init Duration: 716.84 ms

No newer events at this moment. Auto retry paused. Resume
```

Figure 27 - Logs recorded when halifaxDineLambda Lambda function was triggered [6]

Figure 28 displays the screenshot of the email received.



AWS Notification Message

AWS Notifications <no-reply@sns.amazonaws.com>
To Dhrumil Amish Shah
Fri 11/26/2021 4:40 PM

We removed extra line breaks from this message.

```
Here are the list of orders prepared and ready for delivery! :)<+==========================================================+>

Here is the order with order id: a05ba993-bd0a-4f75-a9a1-ef7352d14431
Dish name: THAT WORKS | Dish price: 18.45 | Dish quantity: 2 Dish description: Tomato sauce base, packed with beef pepperoni, beef salami, beef bacon, green pepper, cremini and portabello mushrooms, onions and tomatoes with shredded mozzarella. Finished with fresh parm, asprinkle of oregano and a honey drizzle.

Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1 Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black truffle sea salt.

Dish name: VEGAN PEPPERONI | Dish price: 14.0 | Dish quantity: 1 Dish description: Tomato sauce base, beet pepperoni, vegan mozzarella

Here is the order with order id: f4e41e47-f55e-46a2-bf57-29a3ec44737b
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3 Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 61f909a3-6bad-41e8-802f-121c88121837
Dish name: FUN GUY | Dish price: 14.99 | Dish quantity: 1 Dish description: Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, finished with fresh parmesan, a sprinkle of oregano and black truffle sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 1 Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 09788cf3-e16f-461d-a22b-8fb05480f268
Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 2 Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Here is the order with order id: 56b50866-5d9a-4985-b2f0-518a092216d7
Dish name: PLAIN JANE | Dish price: 12.45 | Dish quantity: 3 Dish description: Tomato sauce base, shredded mozzarella and provolone, finished with a sprinkle of oregano and sea salt.

Dish name: GARLIC FINGERS | Dish price: 12.99 | Dish quantity: 3 Dish description: House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.

Dish name: BBQ CHICKEN | Dish price: 18.0 | Dish quantity: 2 Dish description: Bbq sauce base, marinated grilled chicken breast, shredded mozzarella, onions, tomatoes. Finished with fresh parm and a chipotle bbq swirl.
```

Figure 28 - Email received for all the orders

Program Files

1. **MenuItemModel – MenuItemModel** class is a model class that stores the menu items along with their details. Instances of this class can be stored in a data structure that can be accessed while creating random orders. The code explanation can be found in the comments.

```
/**
 * {@code MenuItemModel} class is a Model class that acts as a menu item.
 * {@code MenuItemModel} class acts as a single entity in a list of menu items.
 *
 * @author Dhrumil Amish Shah (B00857606 | dh416386@dal.ca)
 */
public final class MenuItemModel {
  // Dish name.
  private final String dishName;

  // Dish description.
  private final String dishDescription;

  // Dish price.
  private final double dishPrice;

  /**
   * Constructs {@code MenuItemModel} class objects.
   *
   * @param dishName        name of the dish.
   * @param dishDescription description of the dish.
   * @param dishPrice       price of the dish.
   */
  public MenuItemModel(final String dishName,
                       final String dishDescription,
                       final double dishPrice) {
    this.dishName = dishName;
    this.dishDescription = dishDescription;
    this.dishPrice = dishPrice;
  }

  /**
   * Gets the name of the dish.
   *
   * @return name of the dish.
   */
  public String getDishName() {
    return dishName;
  }

  /**
```

```
   * Gets the description of the dish.
   *
   * @return description of the dish.
   */
  public String getDishDescription() {
    return dishDescription;
  }

  /**
   * Gets the price of the dish.
   *
   * @return price of the dish.
   */
  public double getDishPrice() {
    return dishPrice;
  }
}
```

2. **OrderModel** – OrderModel class acts as an individual order that represents an order with multiple dishes and quantities. This class contains a list of MenuItemModel and their quantities. The details about this class can be found in the comment section of the code.

```
import java.util.List;

/**
 * {@code OrderModel} class is a Model class that acts as an order.
 * {@code OrderModel} class contains list of menu items and quantities ordered.
 *
 * @author Dhrumil Amish Shah (B00857606 | dh416386@dal.ca)
 */
public final class OrderModel {
  // Menu items list.
  private final List<MenuItemModel> menuItemModels;

  // Quantities list.
  private final List<Integer> quantities;

  /**
   * Constructs {@code OrderModel} class objects.
   *
   * @param menuItemModels menu items list.
   * @param quantities     quantities list.
   */
  public OrderModel(final List<MenuItemModel> menuItemModels,
                    final List<Integer> quantities) {
    this.menuItemModels = menuItemModels;
    this.quantities = quantities;
  }

  /**
   * Adds the menu item in the menu items list.
   *
   * @param menuItemModel menu item to add in menu items list.
   */
  public void addInMenuItemModels(final MenuItemModel menuItemModel) {
    menuItemModels.add(menuItemModel);
  }

  /**
   * Gets the list of menu items.
   *
   * @return menu items list.
   */
  public List<MenuItemModel> getMenuItemModels() {
    return menuItemModels;
  }

  /**
   * Adds the quantity in the quantity list.
   *
   * @param quantity quantity to add in quantities list.
   */
  public void addInQuantities(final Integer quantity) {
    quantities.add(quantity);
  }

  /**
   * Gets the list of quantities.
   *
   * @return quantities list.
   */
  public List<Integer> getQuantities() {
    return quantities;
  }
}
```

3. **MenuHelper – MenuHelper** class is a helper class that creates a list of menu items. These menu items are added in the list with details like item name, item description and cost. The code explanation can be found in it’s comment section.

```
import java.util.ArrayList;
import java.util.List;
```

```java
/**
 * {@code MenuHelper} class is a helper class that prepares a menu for HalifaxDine.
 * HalifaxDine is a pizza company that serves delicious pizzas.
 *
 * @author Dhrumil Amish Shah (B00857606 | dh416386@dal.ca)
 */
public final class MenuHelper {

  /**
   * Prepares a list of {@code MenuItemModel} dishes with each dish containing dish name, dish
description and dish price.
   * [Citation] Menu item source: https://gstreetpizza.ca/wp-content/uploads/2021/09/Menu-
Digital-Web.pdf
   *
   * @return a list of {@code MenuItemModel} items.
   */
  public static List<MenuItemModel> getMenu() {
    final List<MenuItemModel> menuList = new ArrayList<>();
    menuList.add(new MenuItemModel("G-STREET",
        "Our signature pizza, simple/spicy. Tomato sauce base, beef pepperoni fresh jalapenos, " +
            "shredded mozzarella, topped with freshly grated parmesan and a sprinkle of oregano.",
        15.0));
    menuList.add(new MenuItemModel("MARGHERITA",
        "Tomato sauce base, fior di latte, fresh basil, finished with an extra virgin olive oil drizzle "
+
            "and fresh parmesan.",
        14.0));
    menuList.add(new MenuItemModel("FUN GUY",
        "Tomato sauce base, shredded mozzarella, loaded with cremini and porabello mushrooms, "
+
            "finished with fresh parmesan, a sprinkle of oregano and black truffle sea salt.",
        14.99));
    menuList.add(new MenuItemModel("GARLIC FINGERS",
        "House made roasted garlic confit spread, shredded mozzarella, finished with sea salt.",
        12.99));
    menuList.add(new MenuItemModel("PLAIN JANE",
        "Tomato sauce base, shredded mozzarella and provolone, finished with a sprinkle of
oregano and sea salt.",
        12.45));
    menuList.add(new MenuItemModel("BBQ CHICKEN",
        "Bbq sauce base, marinated grilled chicken breast, shredded mozzarella, " +
            "onions, tomatoes. Finished with fresh parm and a chipotle bbq swirl.",
        18.00));
    menuList.add(new MenuItemModel("THAT WORKS",
        "Tomato sauce base, packed with beef pepperoni, beef salami, beef bacon, green pepper, " +
          "cremini and portabello mushrooms, onions and tomatoes with shredded mozzarella.
Finished with fresh " +
          "parm, asprinkle of oregano and a honey drizzle.",
      18.45));
    menuList.add(new MenuItemModel("VEGAN PEPPERONI",
      "Tomato sauce base, beet pepperoni, vegan mozzarella",
      14.00));
    return menuList;
  }
}
```

4. **HalifaxDineLambda - HalifaxDineLambda** class is a class that polls the SQS queue upon receiving a new order, prepares a new order message, and sends it to a subscriber. The code explanation can be found in its comments section.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.AmazonSNSException;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sns.model.ListTopicsRequest;
import com.amazonaws.services.sns.model.ListTopicsResult;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.PublishResult;
import com.amazonaws.services.sns.model.Topic;

import java.util.List;

/**
 * {@code HalifaxDineLambda} class polls the SQS Queue when a new order is received,
prepares a message
 * for that order, and sends it to the subscriber.
 *
 * @author Dhrumil Amish Shah (B00857606 | dh416386@dal.ca)
 */
public final class HalifaxDineLambda implements RequestHandler<SQSEvent, Void> {
  private static final String LOG_TAG = HalifaxDineLambda.class.getSimpleName();
  private static final String SNS_TOPIC = "HalifaxDineSNS";

  /**
   * Publishes the received message to all the subscribers of the provided {@code topicArn}.
   *
   * @param amazonSNSClient Amazon SNS client.
   * @param message        Message body to be delivered.
   * @param topicArn       SNS Topic ARN (Amazon Resource Name).
   * @param context        Context object.
   *
   * @throws AmazonSNSException if any error occurs while publishing the message.
   */
  private void publishSNSTopicViaEmail(final AmazonSNS amazonSNSClient,
                                       final String message,
                                       final String topicArn,
                                       final Context context) throws AmazonSNSException {
    final PublishRequest request = new PublishRequest();
    request.setMessage(message);
    request.setTopicArn(topicArn);
    final PublishResult result = amazonSNSClient.publish(request);
    context.getLogger().log(LOG_TAG + " => " + result.getMessageId() + " Message sent. " +
        "Status is " + result.getSdkHttpMetadata().getHttpStatusCode());
  }

  /**
   * Creates if not already created, a new SNS Topic {@code SNS_TOPIC} and retrieves it's
Topic ARN.
   *
   * @param amazonSNSClient Amazon SNS Client.
   *
   * @return Created topic's ARN.
   *
   * @throws AmazonSNSException if any error occurs while creating an SNS Topic or
retrieving the topic's ARN.
   */
  private String createSNSTopic(final AmazonSNS amazonSNSClient) throws
AmazonSNSException {
    final ListTopicsResult listTopicsResult = amazonSNSClient.listTopics(new
ListTopicsRequest());
    final List<Topic> topics = listTopicsResult.getTopics();
    for (final Topic topic : topics) {
      final String[] topicARNSplit = topic.getTopicArn().split(":");
      if (topicARNSplit[topicARNSplit.length - 1].equals(SNS_TOPIC)) {
        return topic.getTopicArn();
      }
    }
    final CreateTopicRequest request = new CreateTopicRequest().withName(SNS_TOPIC);
    final CreateTopicResult result = amazonSNSClient.createTopic(request);
    return result.getTopicArn();
  }

  /**
```

```
   * Retrieves the default SNS Client object.
   *
   * @return default SNS Client object.
   */
  private AmazonSNS createAmazonSNSClientBuilder() {
    return AmazonSNSClientBuilder.defaultClient();
  }

  @Override
  public Void handleRequest(final SQSEvent sqsEvent,
                            final Context context) {
    final StringBuilder emailMsgStringBuilder = new StringBuilder();
    emailMsgStringBuilder.append("Here are the list of orders prepared and ready for delivery!
:)");

emailMsgStringBuilder.append("<+==============================================
================+>");
    emailMsgStringBuilder.append("\n\n");
    for (final SQSEvent.SQSMessage sqsMessage : sqsEvent.getRecords()) {
      emailMsgStringBuilder.append(sqsMessage.getBody());
      emailMsgStringBuilder.append("\n\n");
    }
    context.getLogger().log(LOG_TAG + " => " + emailMsgStringBuilder.toString());
    try {
      final AmazonSNS amazonSNSClient = createAmazonSNSClientBuilder();
      final String snsTopicARN = createSNSTopic(amazonSNSClient);
      if (snsTopicARN == null) {
        return null;
      }
      publishSNSTopicViaEmail(amazonSNSClient, emailMsgStringBuilder.toString(),
snsTopicARN, context);
    } catch (final AmazonSNSException e) {
      e.printStackTrace();
      context.getLogger().log(LOG_TAG + " => " + e.getMessage());
    }
    return null;
  }
}
```

5. **HalifaxDineProgram – HalifaxDineProgram** class is a class that performs the order placing at HalifaxDine. The class performs some key tasks of the assignment such as creation of random orders, create food order messages, and process orders. The class design is based on good development practices. The code explanation can be found in its comments section.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.GetQueueUrlRequest;
import com.amazonaws.services.sqs.model.SendMessageRequest;
import com.amazonaws.services.sqs.model.SendMessageResult;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.UUID;

/**
 * {@code HalifaxDineProgram} class simulates the order placing at HalifaxDine.
 *
 * @author Dhrumil Amish Shah (B00857606 | dh416386@dal.ca)
 */
public final class HalifaxDineProgram {
  // AWS Access Key.
  private static final String AWS_ACCESS_KEY;

  // AWS Secret Key.
  private static final String AWS_SECRET_KEY;

  // AWS Session Token.
  private static final String AWS_SESSION_TOKEN;

  // AWS BasicSessionCredentials object prepared using AWS Access Key, AWS Secret Key,
and AWS Session Token.
  private static final BasicSessionCredentials AWS_CREDENTIALS;

  // HalifaxDine SQS URL Name.
  private static final String HALIFAX_DINE_SQS_URL_NAME;

  // Place new order after every interval.
  private static final int ORDERS_INTERVAL_MILLISECONDS;

  static {
    AWS_ACCESS_KEY = "AWS_ACCESS_KEY";
    AWS_SECRET_KEY = "AWS_SECRET_KEY";
    AWS_SESSION_TOKEN = "AWS_SESSION_TOKEN";
    AWS_CREDENTIALS = new BasicSessionCredentials(AWS_ACCESS_KEY,
AWS_SECRET_KEY, AWS_SESSION_TOKEN);
    HALIFAX_DINE_SQS_URL_NAME = "HalifaxDineSQS";
    ORDERS_INTERVAL_MILLISECONDS = 4 * 60 * 1000; // 4 minutes
  }
 /**
  * Prepares an order {@code OrderModel} with each order containing menu items {@code
MenuItemModel} and
  * quantity {@code quantity} for each item. Each order can have minimum of 1 dish and
maximum of 3 dishes
  * if number of items in the menu is greater than or equal to 3 otherwise the order can have
items
  * same as the number of dishes in the menu. Also, each dish can have minimum of 1 quantity
and maximum of 3.
  * Lastly, same dish cannot be repeated in the menu.
  *
  * @return random orders prepared with menu items and quantities.
  */
 private OrderModel getRandomFoodOrder() {
   final List<MenuItemModel> halifaxDineMenu = MenuHelper.getMenu();
   final OrderModel order = new OrderModel(new ArrayList<>(), new ArrayList<>());

   final int minItemsInOrder = 1;
   final int maxItemsInOrder = Math.min(halifaxDineMenu.size(), 3);
   final int totalItemsInOrder = new Random().nextInt(maxItemsInOrder) + minItemsInOrder;

   final int minQuantityOfOrderItem = 1;
   final int maxQuantityOfOrderItem = 3;

   final List<Integer> itemsOrderedAlready = new ArrayList<>();
   for (int i = 1; i <= totalItemsInOrder; ++i) {
     int randomMenuItemIndex = new Random().nextInt(halifaxDineMenu.size());
     if (itemsOrderedAlready.contains(randomMenuItemIndex)) {
       i--;
       continue;
     }
     itemsOrderedAlready.add(randomMenuItemIndex);
     int randomQuantity = new Random().nextInt(maxQuantityOfOrderItem) +
minQuantityOfOrderItem;
     order.addInMenuItemModels(halifaxDineMenu.get(randomMenuItemIndex));
     order.addInQuantities(randomQuantity);
   }

   return order;
 }

 /**
  * Prepares a {@code String} message of the order and assigns an order id {@code UUID} to
it.
  *
```

```
   * @return {@code String} message of the order
   */
  private String getFoodOrderMessage() {
    final OrderModel order = getRandomFoodOrder();
    final StringBuilder orderSB = new StringBuilder();
    orderSB.append("Here is the order with order id:
").append(UUID.randomUUID().toString()).append("\n");
    final List<MenuItemModel> menuItemModels = order.getMenuItemModels();
    final List<Integer> quantities = order.getQuantities();

    for (int i = 0; i < menuItemModels.size(); ++i) {
      orderSB.append("Dish name: ").append(menuItemModels.get(i).getDishName()).append(" |
");
      orderSB.append("Dish price: ").append(menuItemModels.get(i).getDishPrice()).append(" |
");
      orderSB.append("Dish quantity: ").append(quantities.get(i)).append("\n");
      orderSB.append("Dish description:
").append(menuItemModels.get(i).getDishDescription()).append("\n\n");
    }

    return orderSB.toString();
  }

  /**
   * Creates an Amazon SQS client builder with credentials and region provided.
   *
   * @return {@code AmazonSQS} client builder object.
   */
  private AmazonSQS createAmazonSQSClientBuilder() {
    return AmazonSQSClientBuilder.standard()
        .withCredentials(new AWSStaticCredentialsProvider(AWS_CREDENTIALS))
        .withRegion(Regions.US_EAST_1)
        .build();
  }

  /**
   * Creates new orders at duration of every {@code ORDERS_INTERVAL_MILLISECONDS}.
   * At every {@code ORDERS_INTERVAL_MILLISECONDS}, minimum of 1 and maximum of 5
orders are created in SQS.
   */
  public void processOrder() {
    System.out.print("\n<+========== HalifaxDine Simulator ==========+>\n\n");

    final AmazonSQS amazonSQSClientBuilder = createAmazonSQSClientBuilder();
    final GetQueueUrlRequest queueUrlRequest = new
GetQueueUrlRequest().withQueueName(HALIFAX_DINE_SQS_URL_NAME);
```

```
    final String queueUrl =
amazonSQSClientBuilder.getQueueUrl(queueUrlRequest).getQueueUrl();

    int orderBatch = 0;
    while (true) {
      orderBatch++;
      System.out.print("<+========== Order Batch: " + orderBatch + " ==========+>\n");
      final int minOrdersInBatch = 1;
      final int maxOrdersInBatch = 5;
      final int totalOrdersInBatch = new Random().nextInt(maxOrdersInBatch) +
minOrdersInBatch;
      final StringBuilder foodOrderMessageSB = new StringBuilder();
      for (int i = 1; i <= totalOrdersInBatch; ++i) {
        final String foodOrderMessage = getFoodOrderMessage();
        foodOrderMessageSB.append(foodOrderMessage);
      }
      System.out.print(foodOrderMessageSB.toString());
      final SendMessageRequest halifaxDineMsgRequest = new SendMessageRequest()
          .withQueueUrl(queueUrl)
          .withMessageBody(foodOrderMessageSB.toString());
      final SendMessageResult sendMessageResult =
amazonSQSClientBuilder.sendMessage(halifaxDineMsgRequest);
      System.out.print("---------- Order from batch " + orderBatch + " posted: " +
sendMessageResult.getMessageId() + "\n\n");
      try {
        System.out.print("Waiting for new orders...\n\n");
        Thread.sleep(ORDERS_INTERVAL_MILLISECONDS);
      } catch (final InterruptedException e) {
        System.out.print("\nHalifaxDineProgram: " + e.getMessage() + "\n");
        e.printStackTrace();
      }
    }
  }
}
```

6. **HalifaxDineProgramTest - HalifaxDineProgramTest** class tests the entire implementation and is the starting point of the whole functionality at HalifaxDine.

```
/**
 * {@code HalifaxDineProgramTest} class tests the {@code HalifaxDineProgram}.
 *
 * @author Dhrumil Amish Shah (B00857606 | dh416386@dal.ca)
 */
public final class HalifaxDineProgramTest {
  public static void main(String[] args) { new HalifaxDineProgram().processOrder(); }}
```

References

[1] AWS, "AWS Lambda," Amazon, [Online]. Available: https://aws.amazon.com/lambda/. [Accessed 26 November 2021].

[2] AWS, "Amazon Simple Queue Service," Amazon, [Online]. Available: https://aws.amazon.com/sqs/. [Accessed 26 November 2021].

[3] AWS, "Amazon Simple Notification Service," Amazon, [Online]. Available: https://aws.amazon.com/sns/. [Accessed 26 November 2021].

[4] draw.io, "Flowchart Maker & Online Diagram Software," draw.io, [Online]. Available: https://app.diagrams.net/. [Accessed 26 November 2021].

[5] AWS, "AWS Identity and Access Management (IAM)," Amazon, [Online]. Available: https://aws.amazon.com/iam/. [Accessed 26 November 2021].

[6] AWS, "Amazon CloudWatch," Amazon, [Online]. Available: https://aws.amazon.com/cloudwatch/. [Accessed 27 November 2021].