ASSIGNMENT 5: PART B

LINK TO GITLAB: https://git.cs.dal.ca/rtyagi/csci5410_b00872269_ruhirajnish_tyagi/-/tree/main/assignment-5

Step 1: AWS console setup – Attaching required policies to LabRole, creation of SNS[1], SQS[2], and EventBridge[3].

Figure 1-12 demonstrate the setup required on AWS Console for the message sending service including attaching policies to the role: LabRole, creating SNS, SQS, and EventBridge.

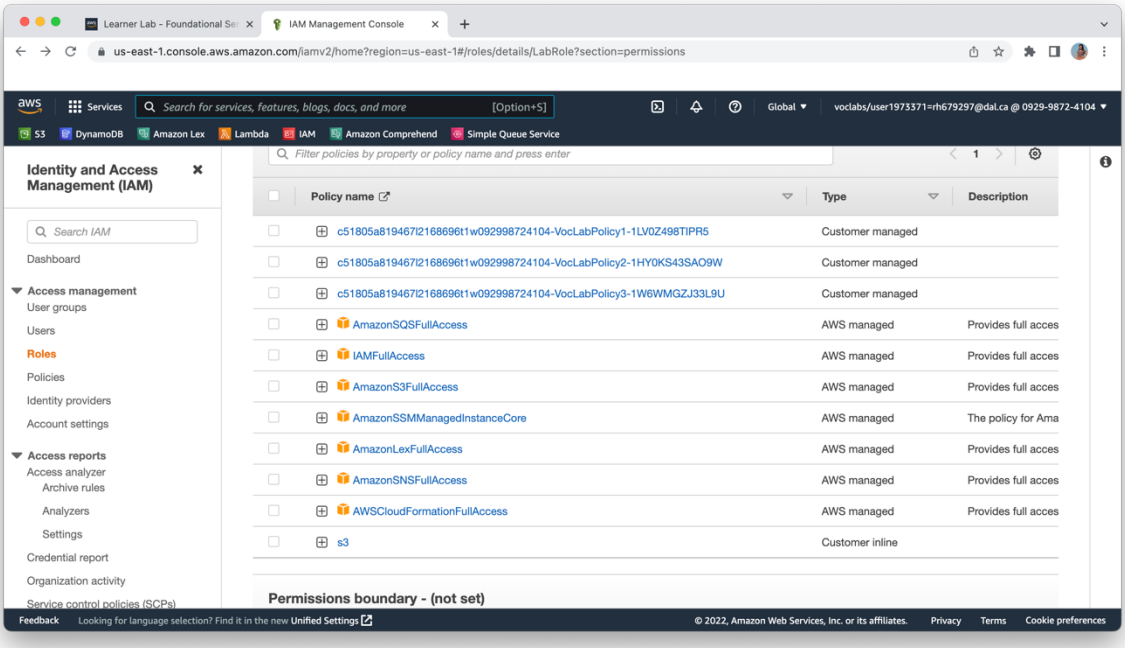

***Figure 1**. Screenshot of attaching policies related to SQS and SNS to LabRole.*

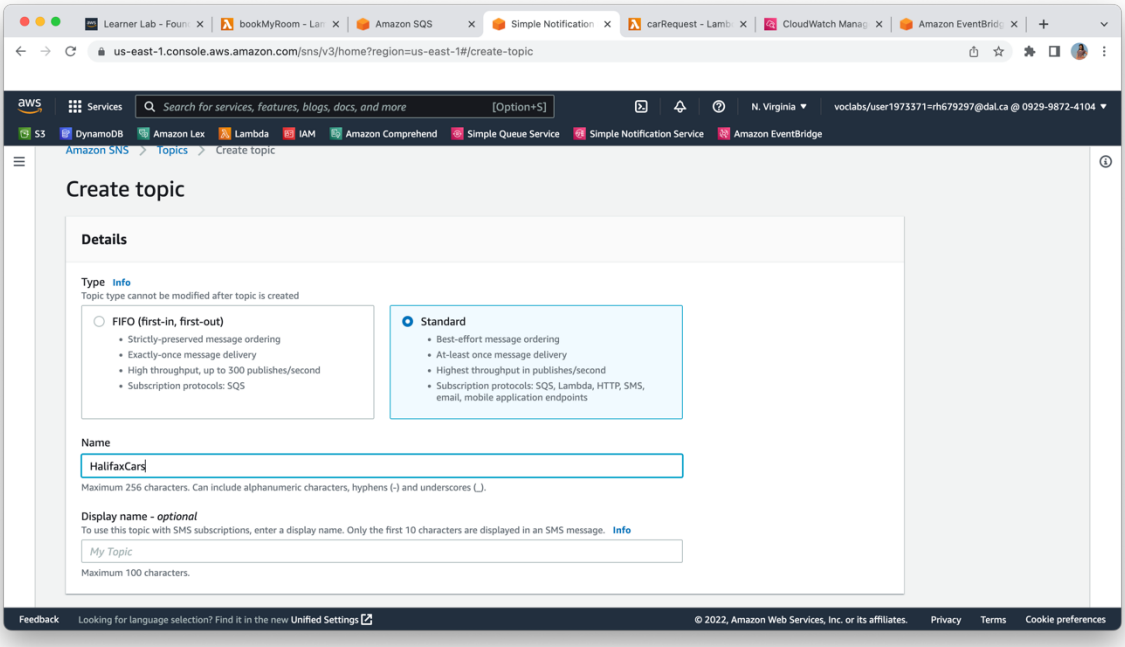

Figure 2(a). *Screenshot of creating topic* ***HalifaxCars*** *on Amazon SNS.*

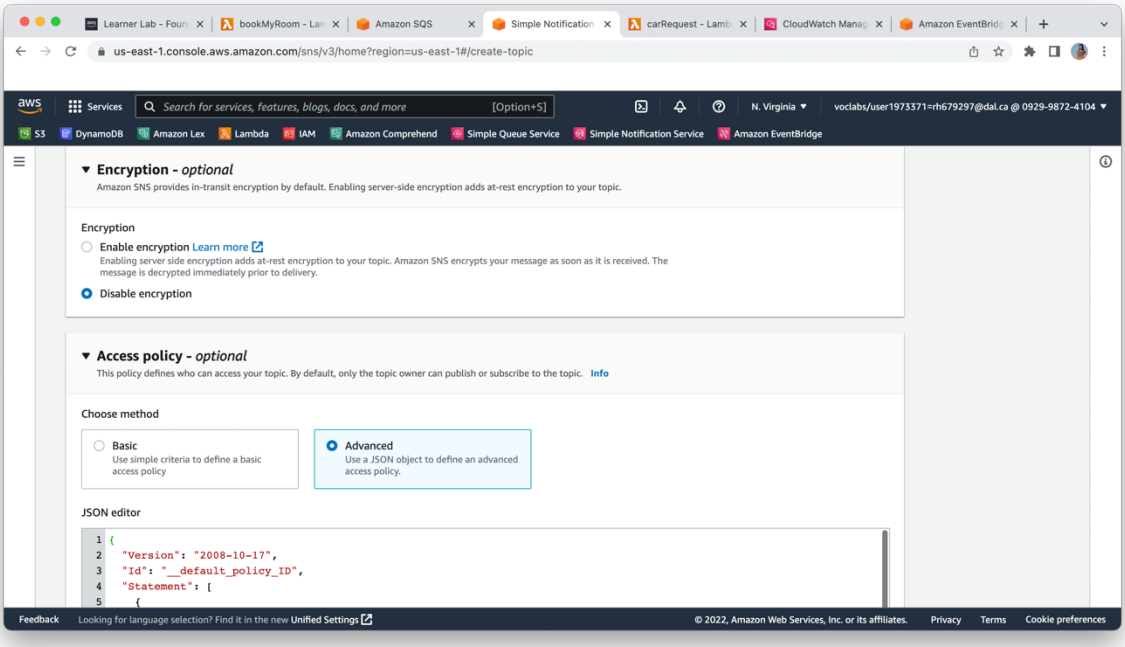

Figure 2(b). *Screenshot of creating topic* ***HalifaxCars*** *on Amazon SNS.*

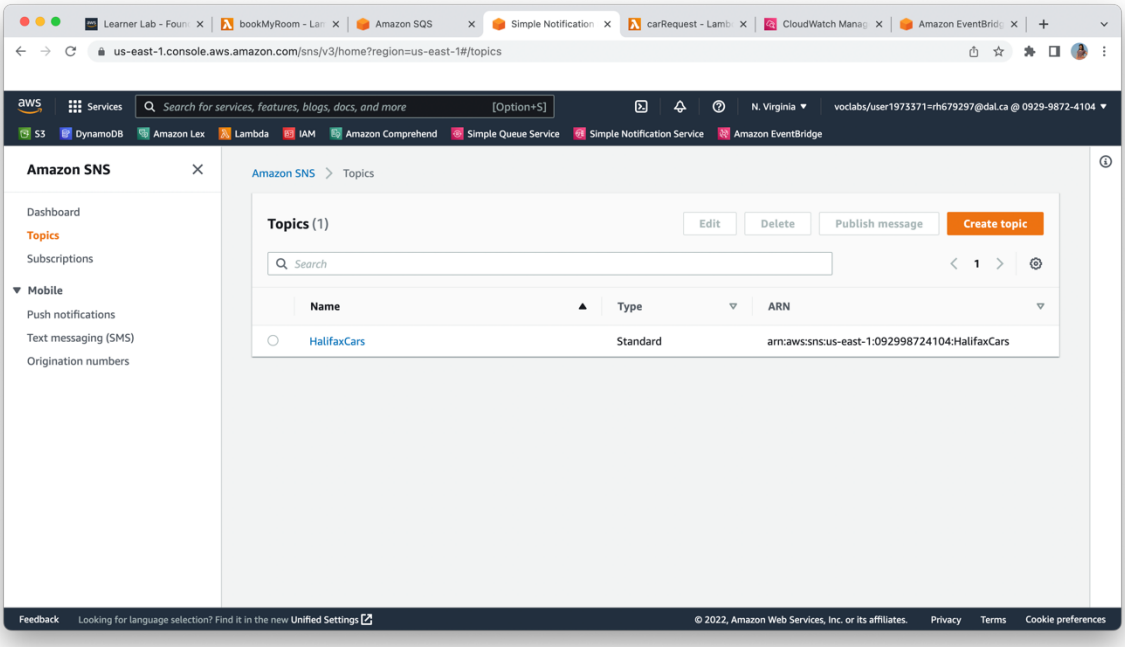

***Figure 3**. Screenshot of created topic **HalifaxCars** on Amazon SNS.*

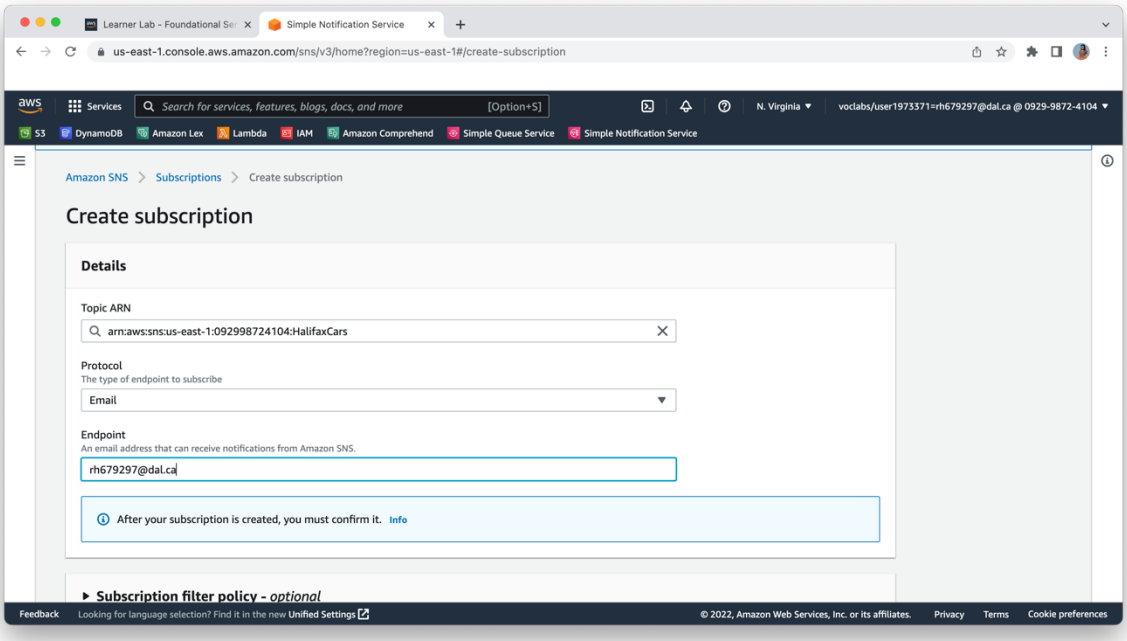

***Figure 4**. Screenshot of creating subscription in Amazon SNS onto dal email.*

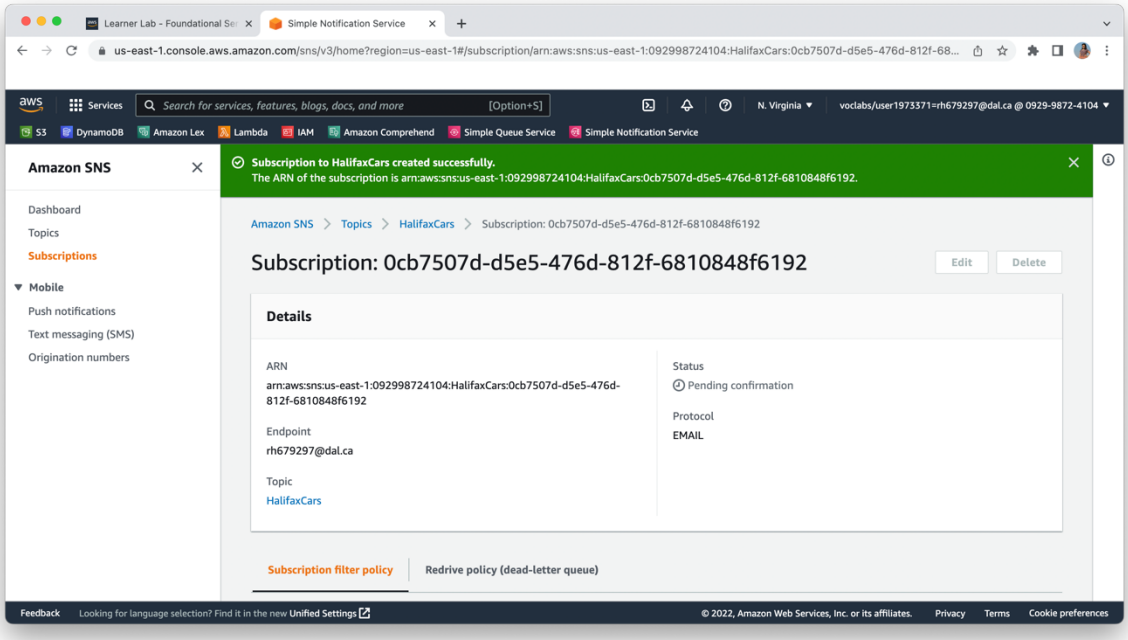

***Figure 5**. Screenshot of PENDING status BEFORE confirming from dal email.*

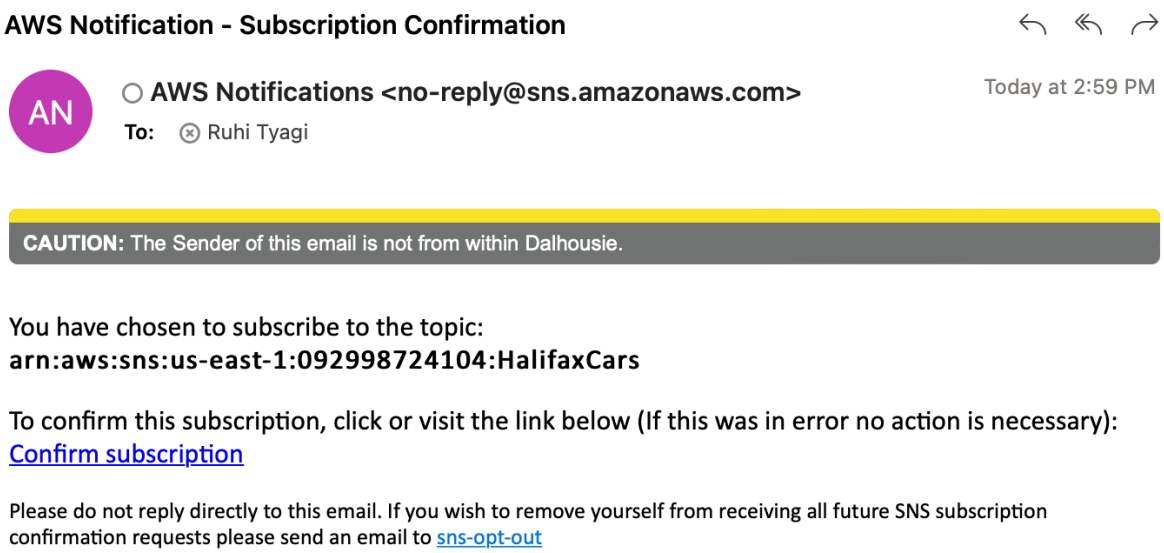

***Figure 6**. Screenshot of 'Confirm subscription' email on my dal email.*

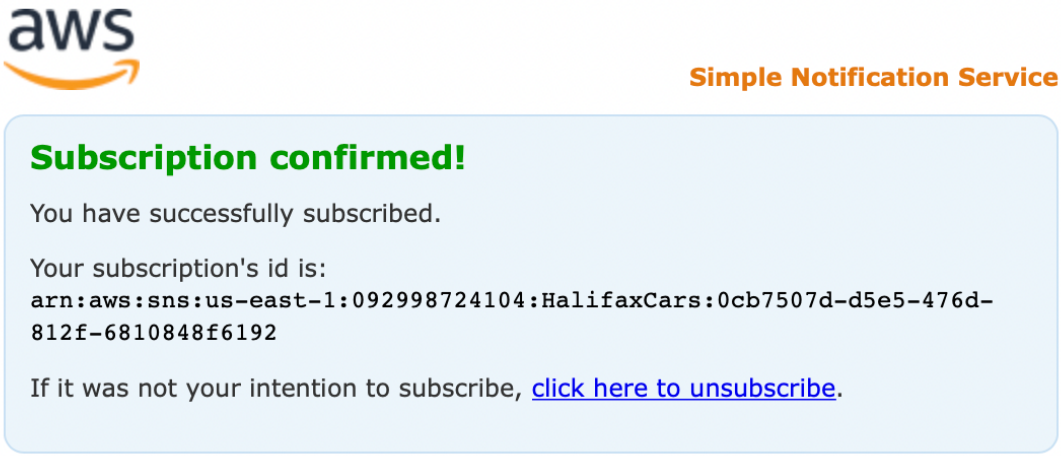

Figure 7. *Screenshot of confirming from dal email.*

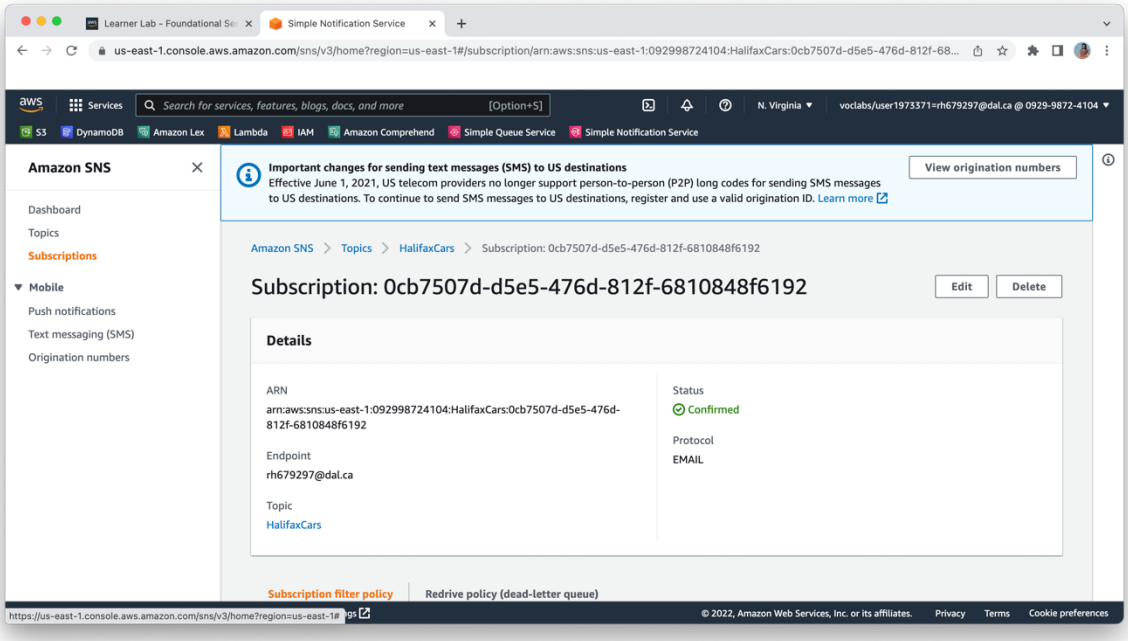

Figure 8(a). *Screenshot of CONFIRMED status AFTER confirming subscription.*

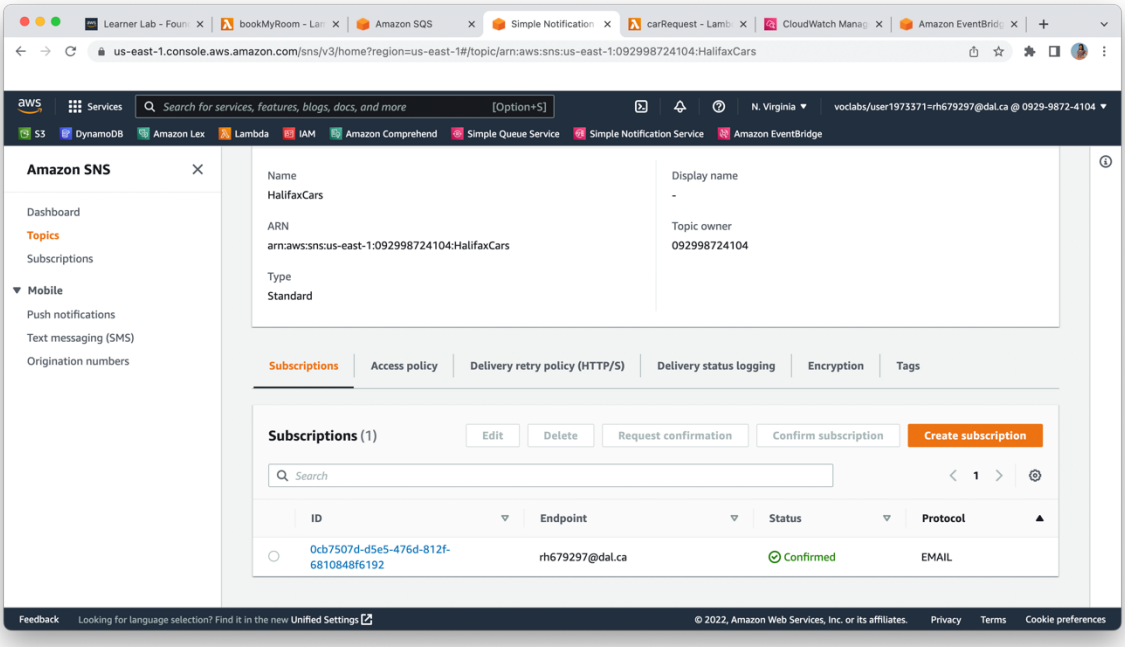

***Figure 8(b)**. Screenshot of CONFIRMED status AFTER confirming subscription.*

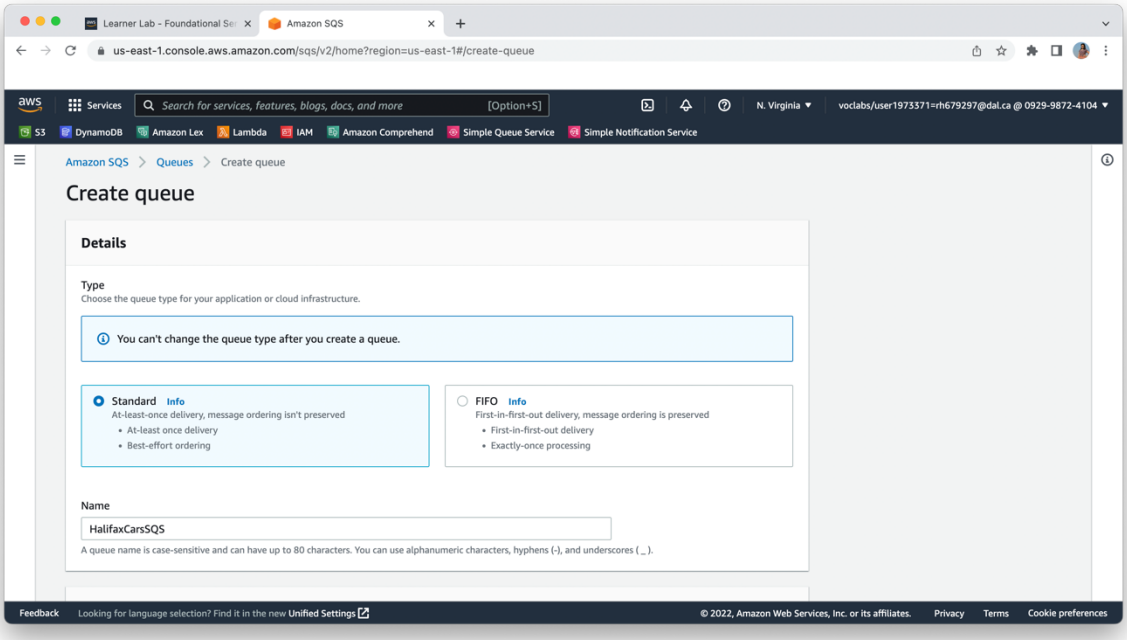

***Figure 9(a)**. Screenshot of creating **HalifaxCarsSQS** in Amazon SQS.*

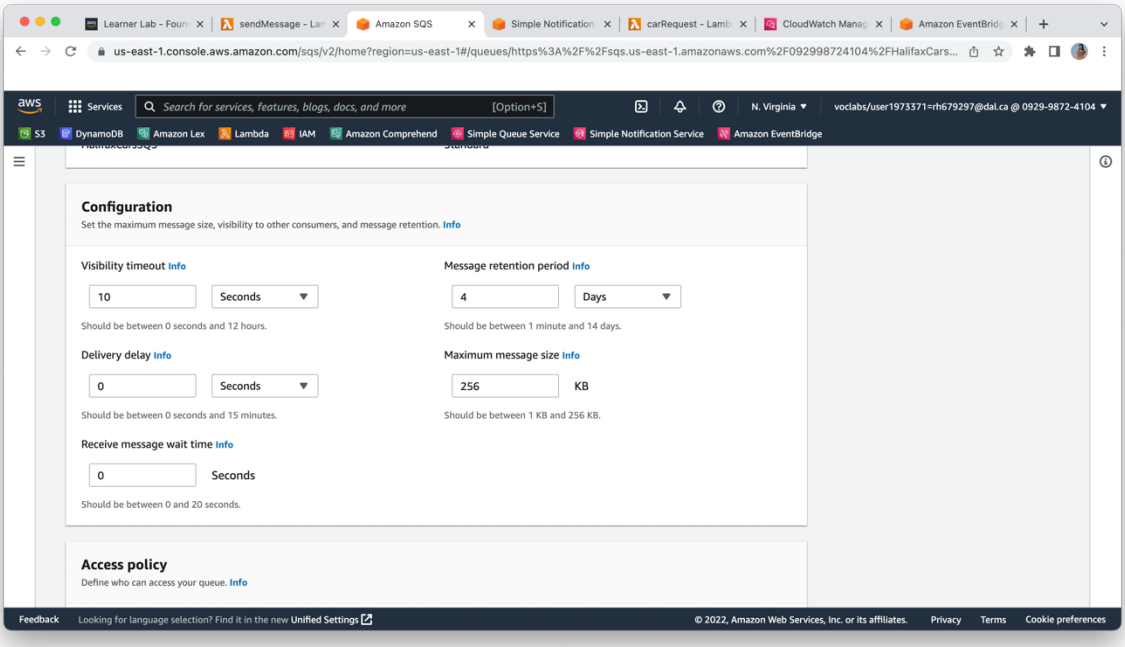

***Figure 9(b)**. Screenshot of creating **HalifaxCarsSQS** in Amazon SQS.*

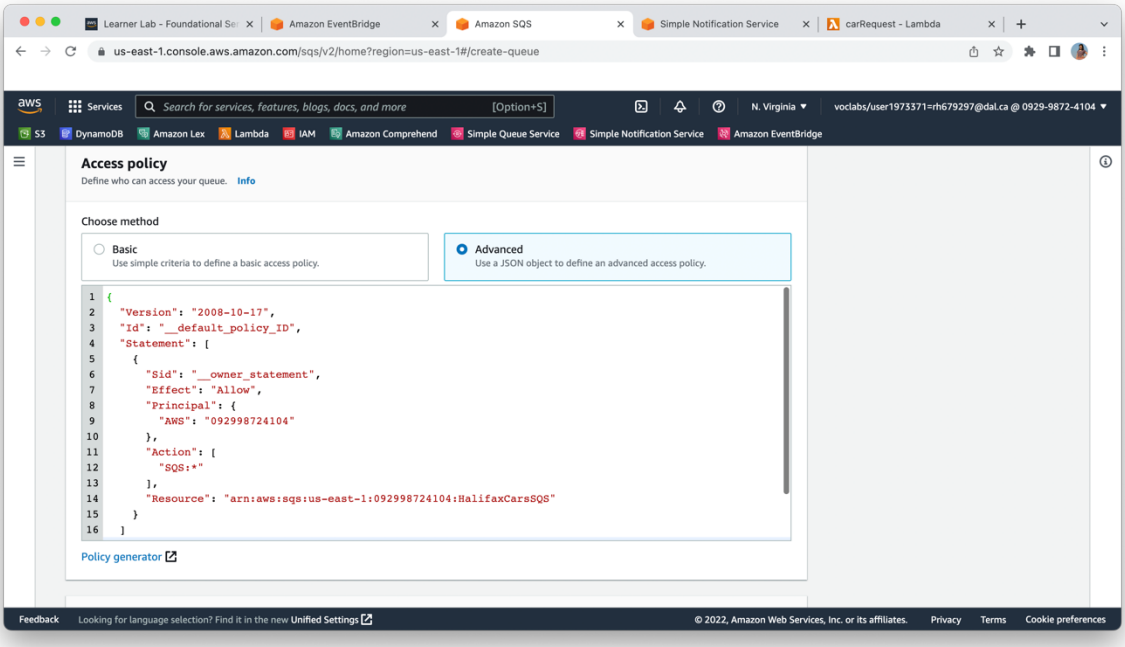

***Figure 9(c)**. Screenshot of creating **HalifaxCarsSQS** in Amazon SQS.*

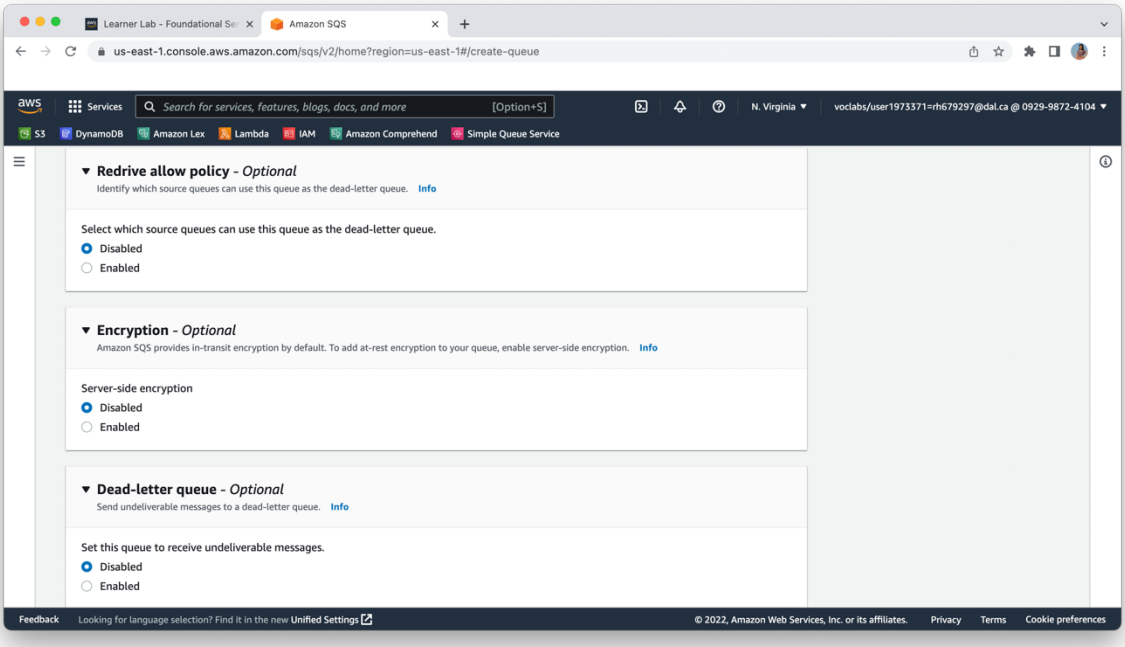

***Figure 9(d)**. Screenshot of creating **HalifaxCarsSQS** in Amazon SQS.*

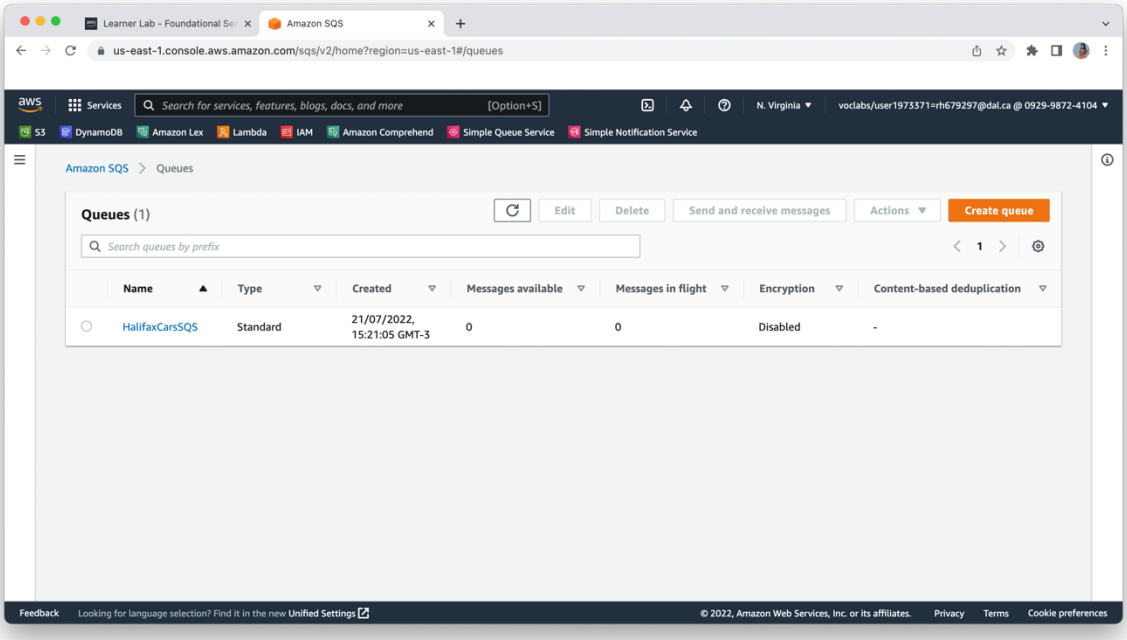

***Figure 10(a)**. Screenshot of SUCCESSFUL creation of **HalifaxCarsSQS** in Amazon SQS.*

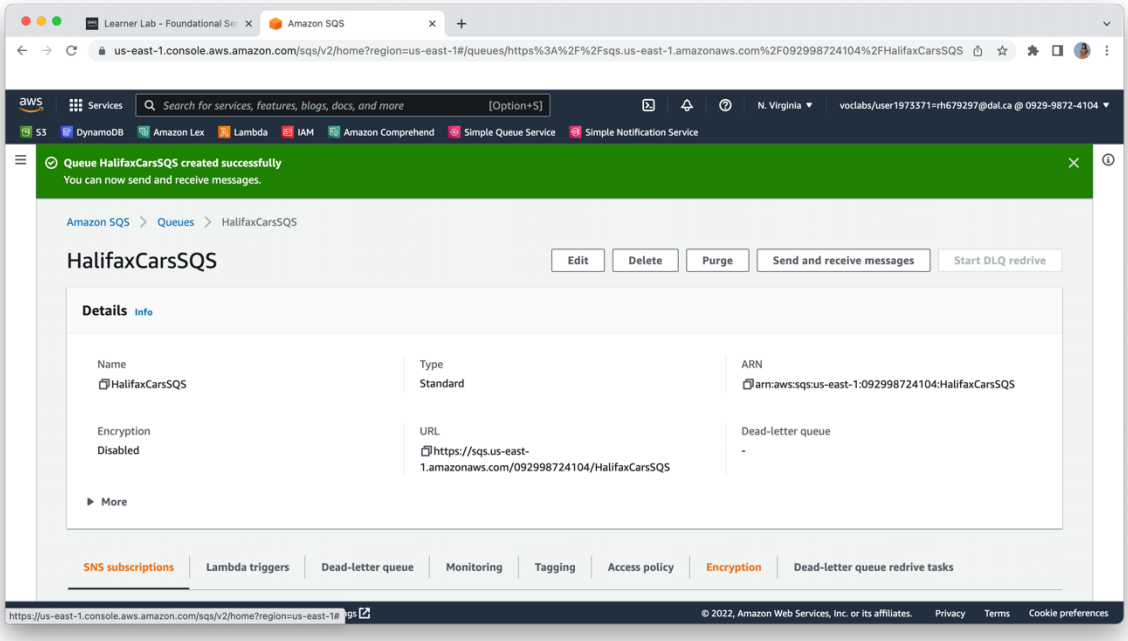

Figure 10(b). *Screenshot of SUCCESSFUL creation of* ***HalifaxCarsSQS*** *in Amazon SQS.*

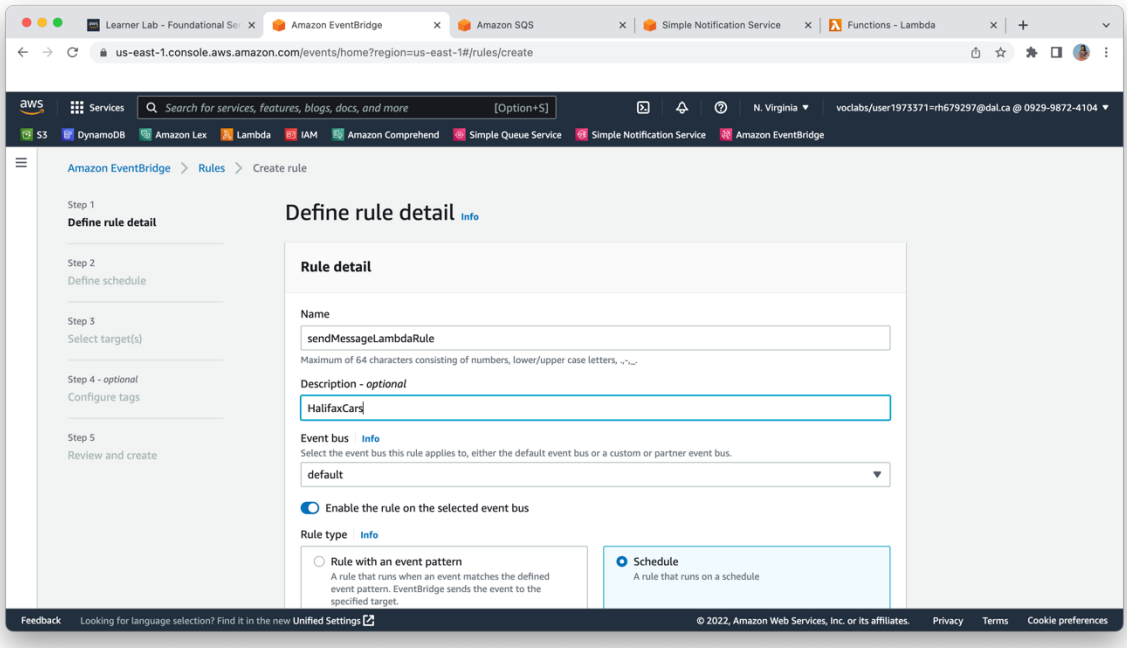

Figure 11(a). *Screenshot of creating of* ***sendMessageLambdaRule*** *rule in Amazon EventBridge.*

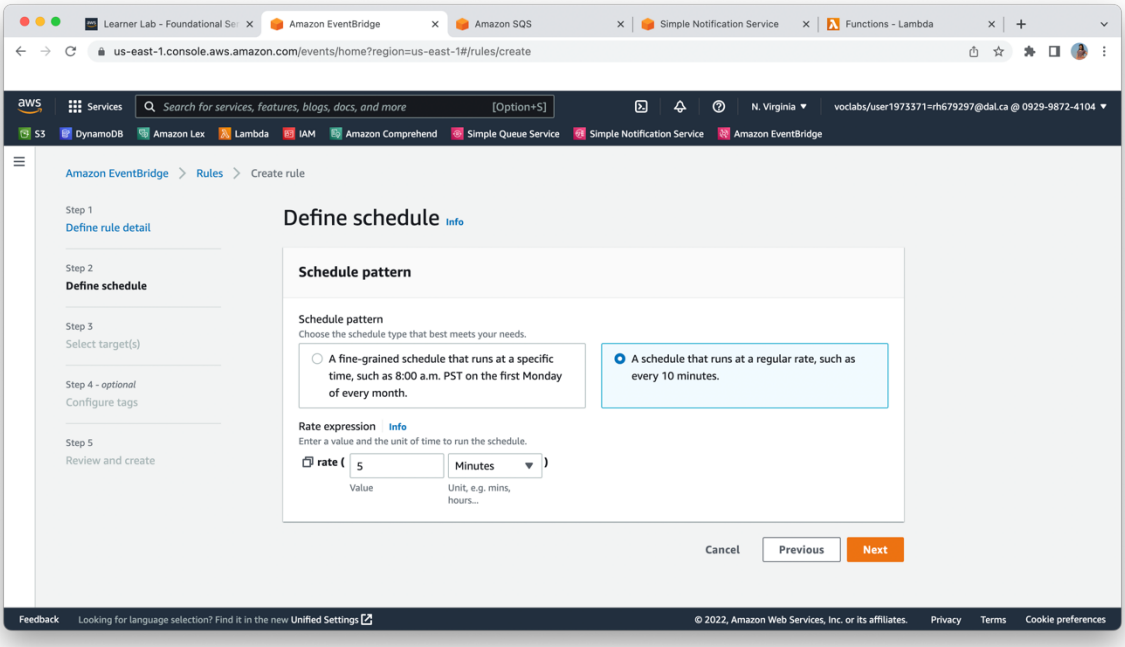

Figure 11(b). *Screenshot of creating of* ***sendMessageLambdaRule*** *rule in Amazon EventBridge.*

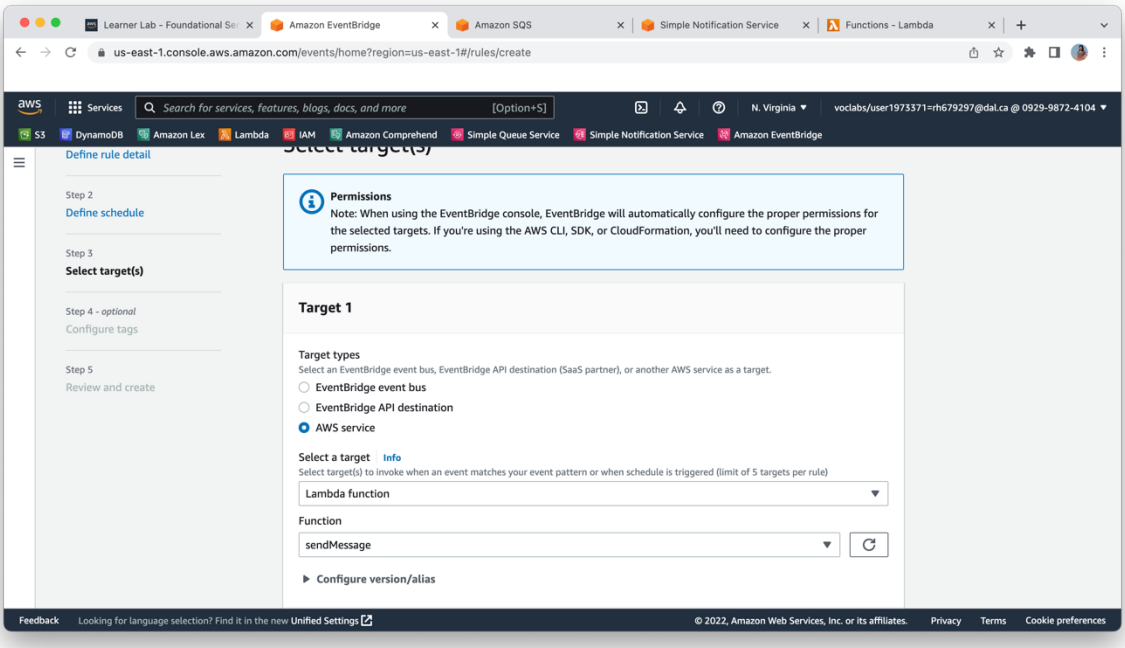

Figure 11(c). *Screenshot of creating of* ***sendMessageLambdaRule*** *rule in Amazon EventBridge.*

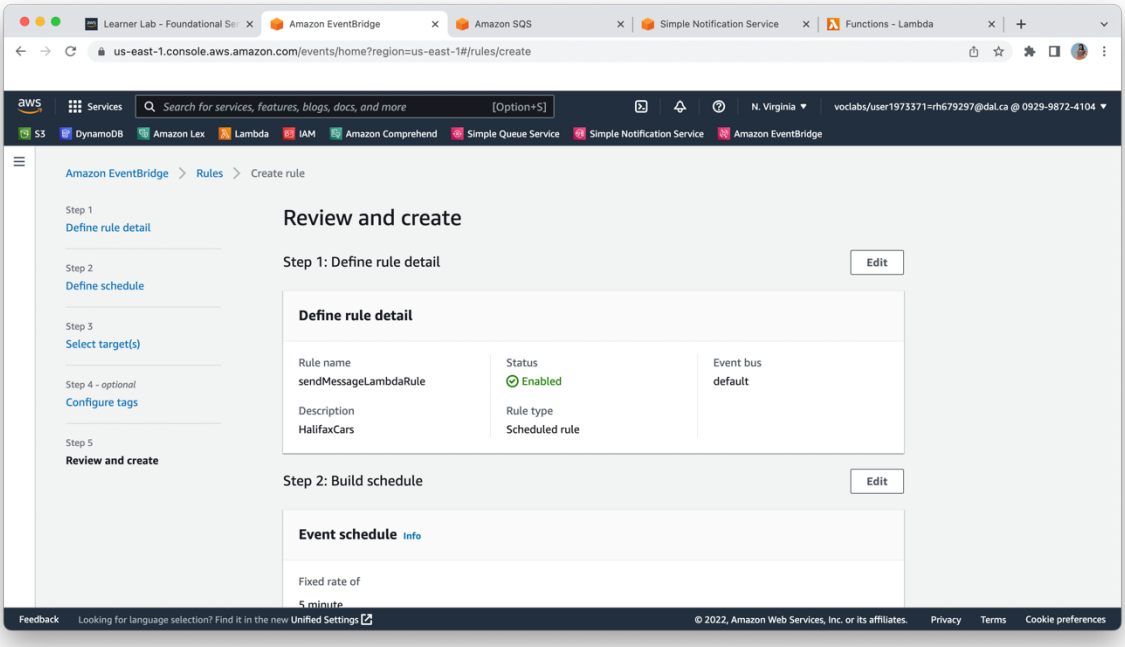

Figure 11(d). *Screenshot of creating of* ***sendMessageLambdaRule*** *rule in Amazon EventBridge.*

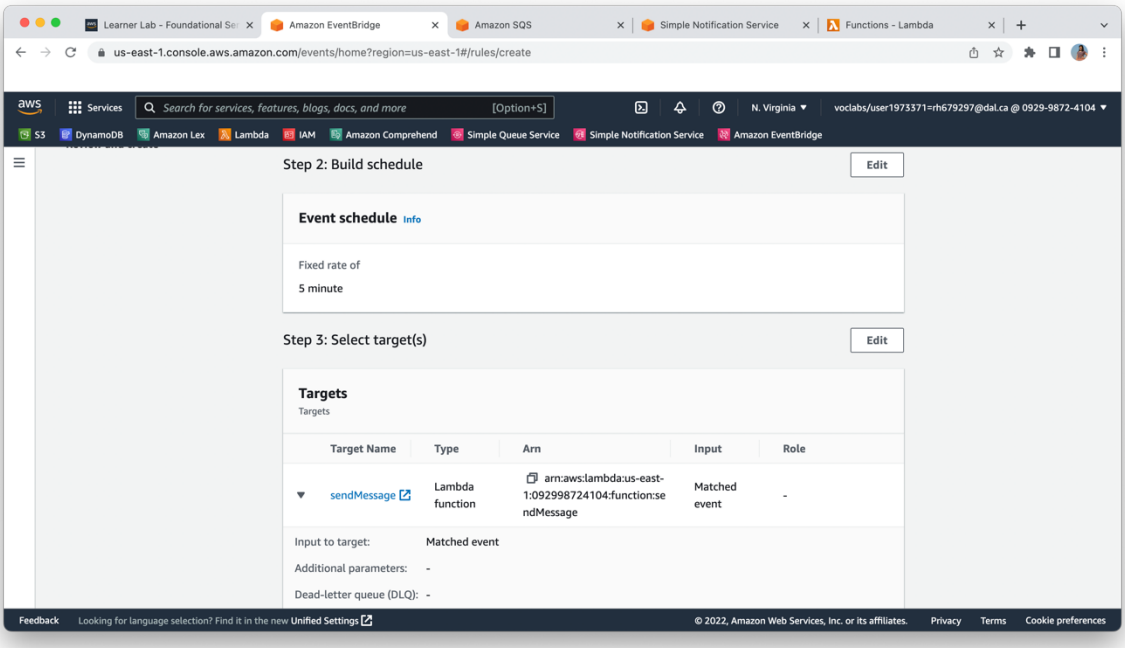

Figure 11(e). *Screenshot of creating of* ***sendMessageLambdaRule*** *rule in Amazon EventBridge.*

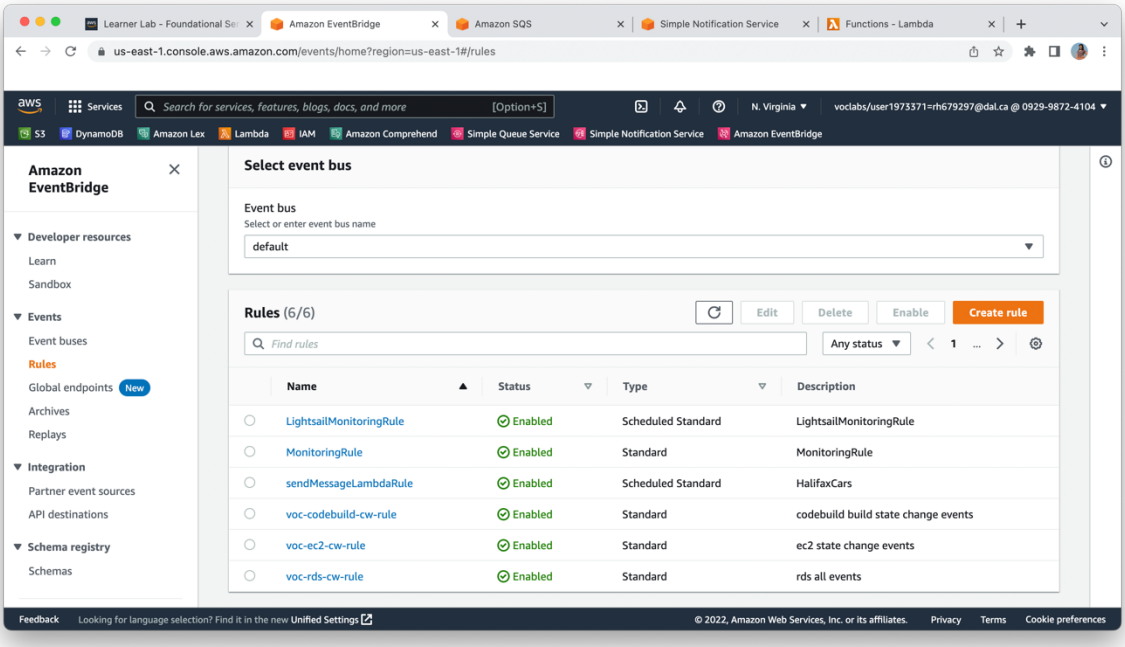

***Figure 12**. Screenshot of SUCCESSFUL creation of **sendMessageLambdaRule** rule in Amazon EventBridge.*

Step 2:Creating Lambda Functions for sending email to the user.

Screenshot 13-20 demonstrate when lambda function carRequest is executed it will send message into HalifaxCarsSQS queue. Then, Bob checks periodically in 5 minutes if there are any messages. After receiving message triggered by sendMessage function, it sends the email to registered email id i.e. my dal email.

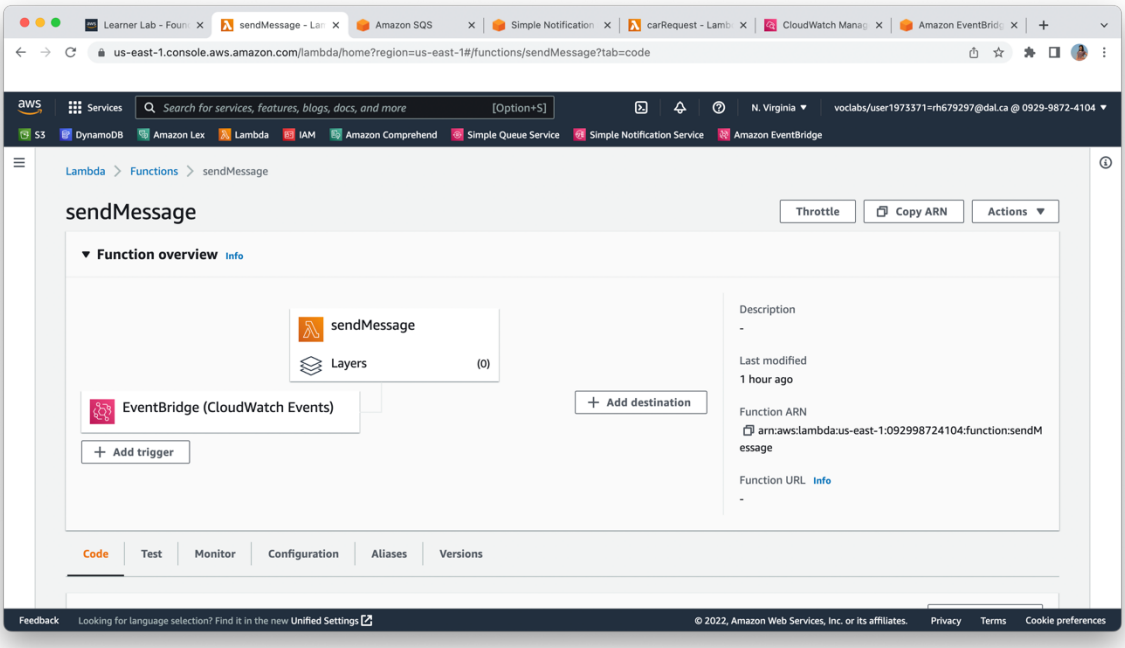

Figure 13(a). *Screenshot of created* ***sendMessage*** *lambda function with trigger set to sendMessagLambdaRule rule of Amazon EventBridge.*

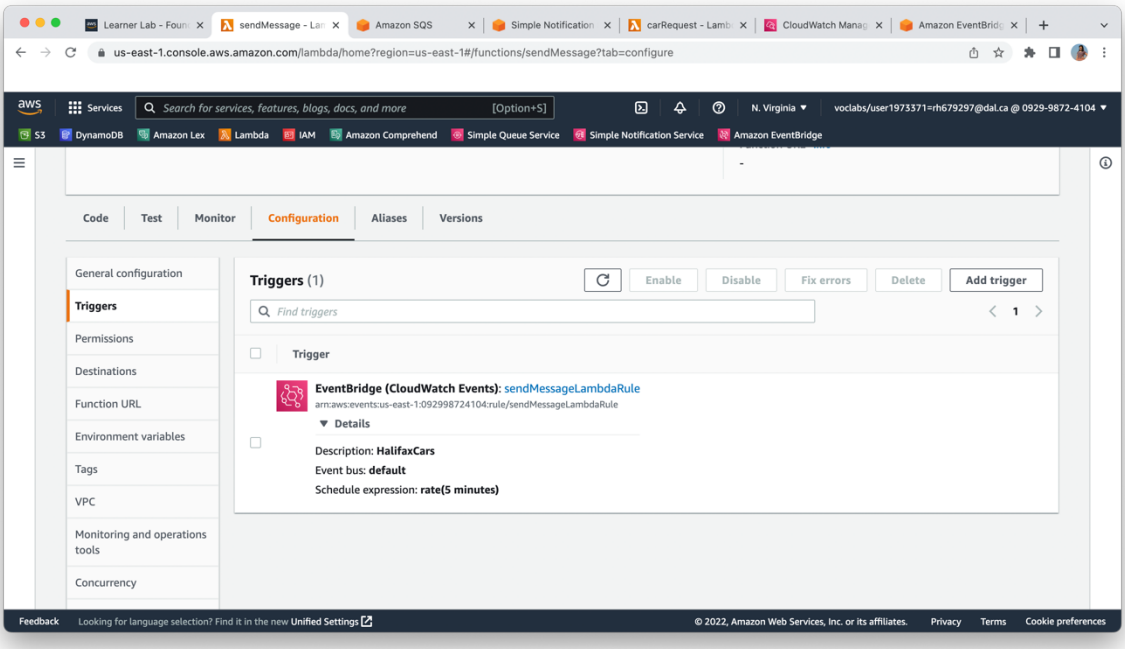

Figure 13(b). *Screenshot of created* ***sendMessage*** *lambda function with trigger set to sendMessagLambdaRule rule of Amazon EventBridge.*

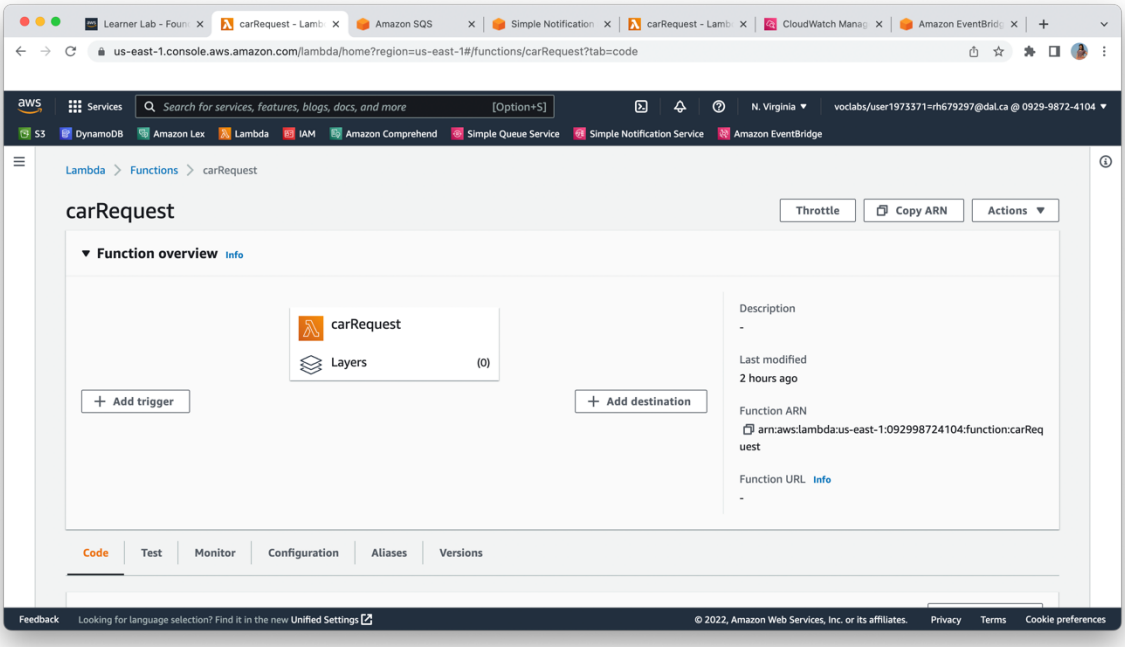

Figure 14. *Screenshot of created of* ***carRequest*** *lambda function.*

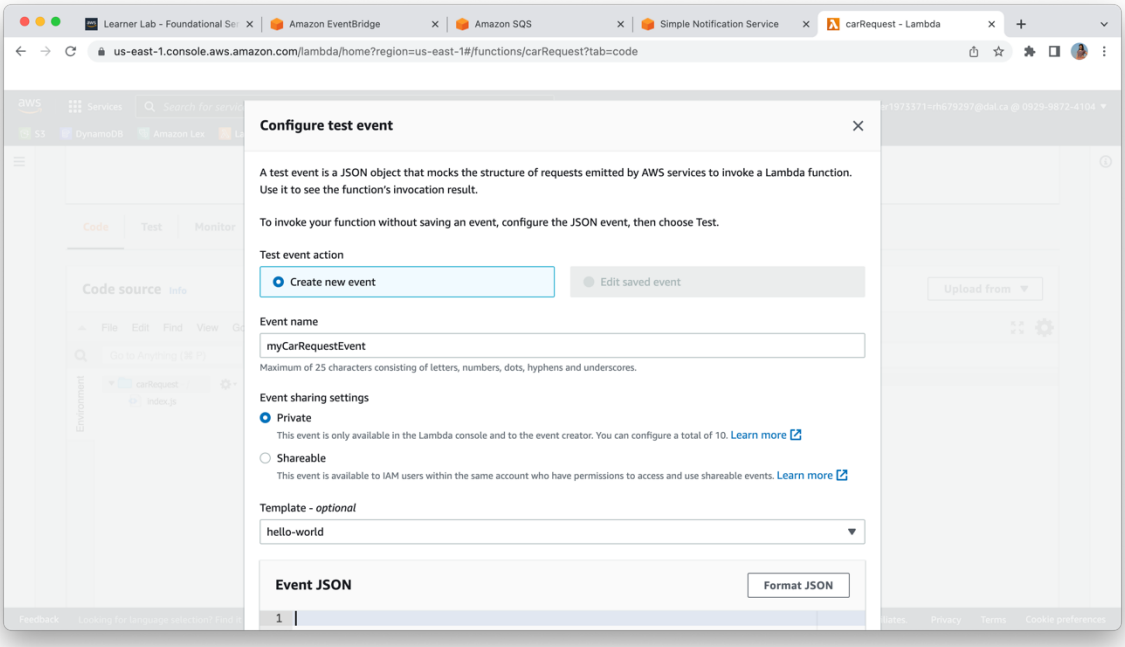

Figure 15(a). *Screenshot of creating test event* ***myCarRequestEvent*** *for* ***carRequest*** *lambda function.*

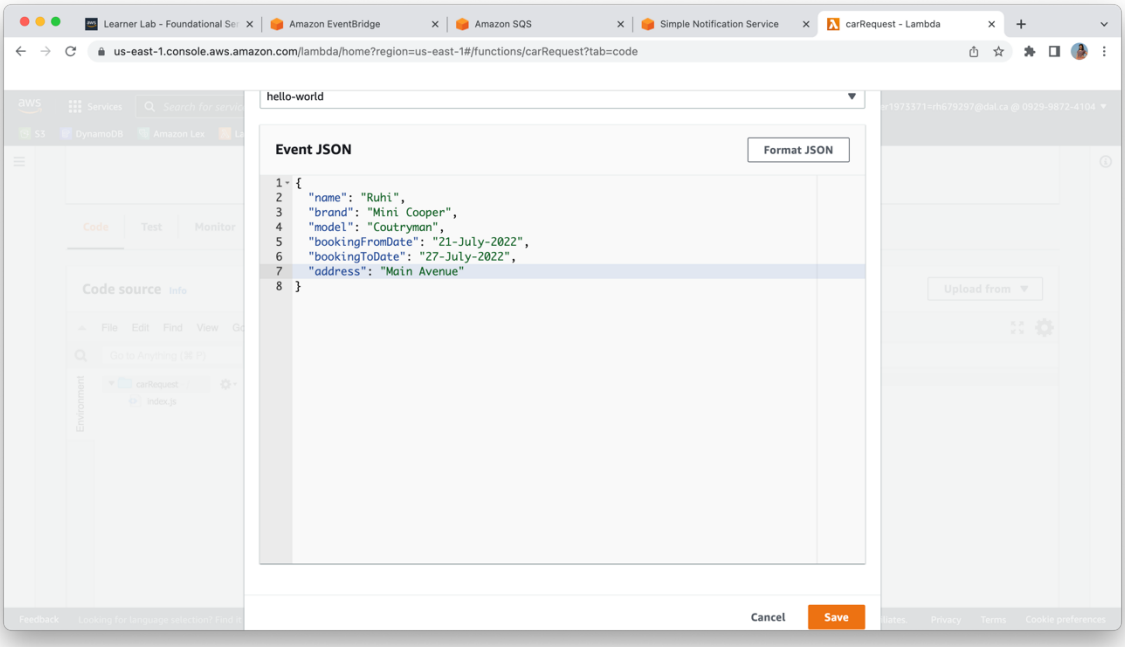

Figure 15(b). *Screenshot of creating test event* ***myCarRequestEvent*** *for* ***carRequest*** *lambda function.*

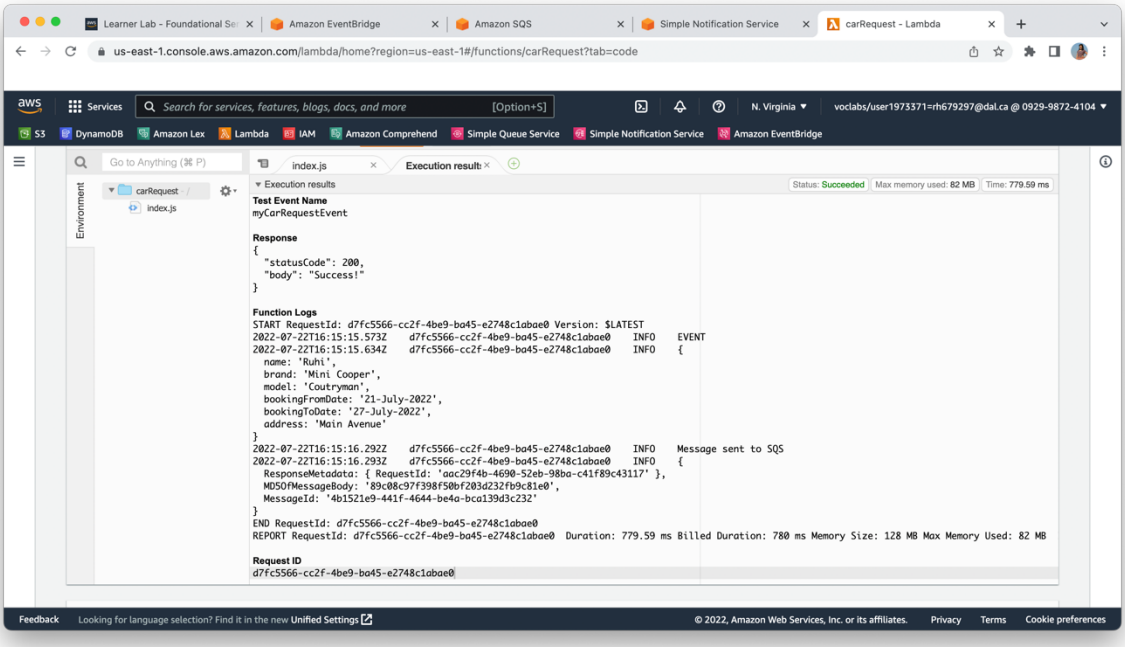

Figure 16. *Screenshot of SUCCESSFUL execution of test event* ***myCarRequestEvent*** *for* ***carRequest*** *lambda function.*

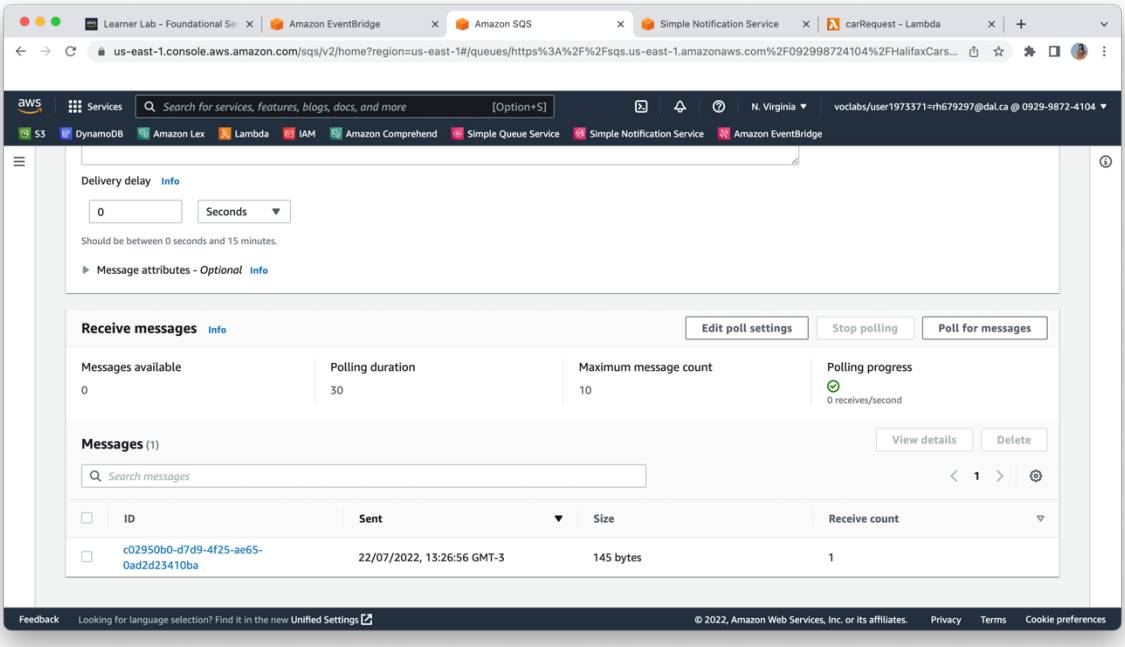

Figure 17*. Screenshot of SUCCESSFUL message received into EventBridge rule.*

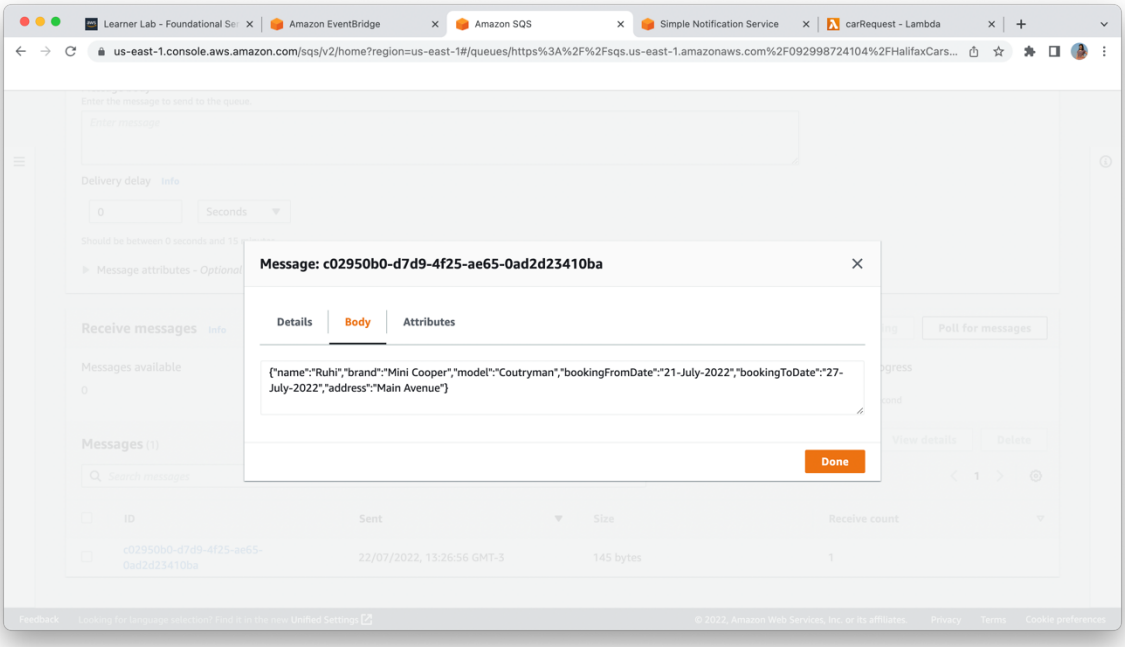

Figure 18*. Screenshot of message body received into EventBridge rule from lambda test event.*

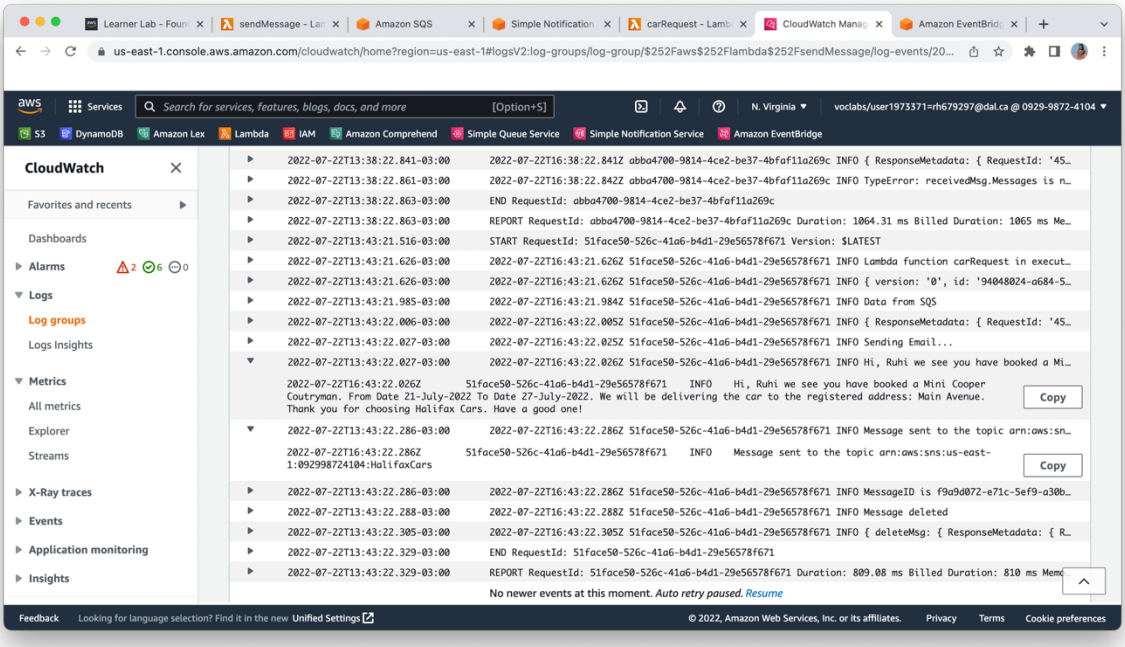

***Figure 19**. Screenshot of SUCCESSFUL event in Cloud Watch Log.*

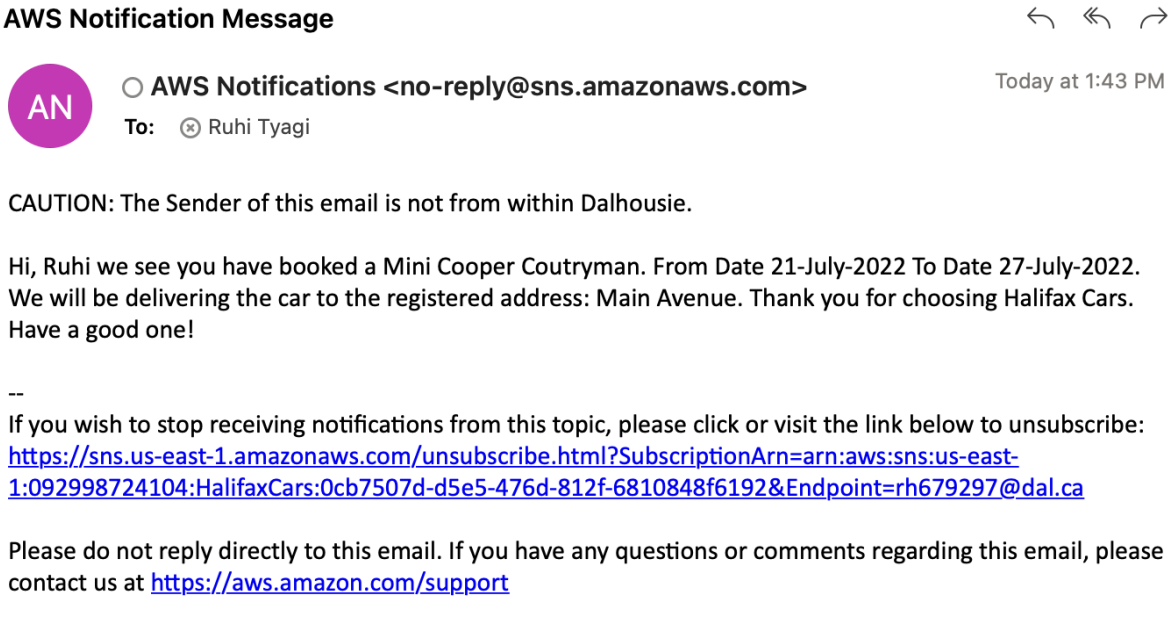

***Figure 20**. Screenshot of email received as per request on dal email.*

Step 3:*Finally, the code of* ***sendMessage*** *and* ***carRequest*** *lambda function.*

sendMessage.js :

```
const AWS = require('aws-sdk');

AWS.config.update({ region: 'us-east-1' });

const sqsVersion = new AWS.SQS({ apiVersion: '2012-11-05' });
const queueURL = "https://sqs.us-east-
1.amazonaws.com/092998724104/HalifaxCarsSQS";
const snsARN = "arn:aws:sns:us-east-1:092998724104:HalifaxCars";

exports.handler = async (event) => {
    console.log('Lambda function carRequest in execution...');
    console.log(event);

    const params = {
        QueueUrl: queueURL
    };

    try {
        const receivedMsg = await
sqsVersion.receiveMessage(params).promise();
        console.log("Data from SQS");
        console.log(receivedMsg);

        for (const message of receivedMsg.Messages) {

            sendEmail(message.Body);
            const deleteParams = {
                QueueUrl: queueURL,
                ReceiptHandle: message.ReceiptHandle,
            }
            const deleteMsg = await
sqsVersion.deleteMessage(deleteParams).promise();
            console.log("Message deleted")
            console.log({ deleteMsg })
        }
    }
    catch (e) {
        console.log(e);
    }

    const response = {
        statusCode: 200,
        body: JSON.stringify("SUCCESS"),
    };
    return response;
};

let sendEmail = (message) => {
    console.log("Sending Email...");
    let body = formatEmailBody(message);
    console.log(body);
    var params = {
```

```
        Message: body,
        TopicArn: snsARN
    };

    var snsVersion = new AWS.SNS({ apiVersion: '2010-03-31'
}).publish(params).promise();
    snsVersion.then(
        function(data) {
            console.log(`Message sent to the topic ${params.TopicArn}`);
            console.log("MessageID is " + data.MessageId);
        }).catch(
        function(err) {
            console.error(err, err.stack);
        });
}

let formatEmailBody = (bodyStr) => {
    let body = JSON.parse(bodyStr);
    let title = "Hi, " + body.name ;
    let carType = " we see you have booked a " + body.brand + " " +
body.model+". ";
    let dateDelivery = "From Date " + body.bookingFromDate + " To Date " +
body.bookingToDate + ". ";
    let addressDelivery = "We will be delivering the car to the registered
address: " + body.address + ". ";
    let endingMsg = "Thank you for choosing Halifax Cars. Have a good one!"

    return title + carType + dateDelivery + addressDelivery + endingMsg;
}
```

carRequest.js :

```
const AWS = require('aws-sdk');

AWS.config.update({ region: 'us-east-1' });

const sqsVersion = new AWS.SQS({ apiVersion: '2012-11-05' });
const queueURL = "https://sqs.us-east-
1.amazonaws.com/092998724104/HalifaxCarsSQS";
exports.handler = async (event) => {
    console.log("EVENT");
    console.log(event);
    try {
        const params = {
            MessageBody: JSON.stringify(event),
            QueueUrl: queueURL
        };
        let sqsResponse = await sqsVersion.sendMessage(params).promise();
        console.log("Message sent to SQS");
        console.log(sqsResponse);
    }
    catch (e) {
        console.log(e);
    }
    const response = {
        statusCode: 200,
        body: 'Success!',
    };
    return response;
};
```

REFERENCES :

[1] "Amazon Simple Notification Service (SNS) | Messaging Service | AWS", *Amazon Web Services, Inc.*, 2022. [Online]. Available: https://aws.amazon.com/sns/. [Accessed: 21- Jul- 2022]

[2] "Amazon SQS | Message Queuing Service | AWS", *Amazon Web Services, Inc.*, 2022. [Online]. Available: https://aws.amazon.com/sqs/. [Accessed: 21- Jul- 2022]

[3] "Amazon EventBridge | Event Bus | Amazon Web Services", *Amazon Web Services, Inc.*, 2022. [Online]. Available: https://aws.amazon.com/eventbridge/. [Accessed: 22- Jul- 2022]