Exercise: SNS - Typescript

In this exercise you will:

1. Create an SNS topic
2. Write a Typescript program that sends messages to your topic
3. Write a Typescript Lambda function that processes messages sent to your topic

Assumptions

You have already configured your AWS credentials on your computer. This should have been done when you installed the AWS CLI on your computer.

Node.js and ts-node are installed on your computer.

Steps

1. Go the the AWS Simple Notification Service (SNS) web console

2. Create an SNS topic
    a. On the left side, select "Topics"
    b. Click the "Create topic" button
    c. Enter a name for your queue
    d. At the bottom, click the "Create topic" button

3. Create an email subscription to your topic. All messages published to your topic will be emailed to the address you enter.
    a. Select the "Subscriptions" tab
    b. Click the "Create subscription" button
    c. For Protocol select "Email"
    d. For Endpoint enter the email addresses messages should be sent to
    e. Click the "Create subscription" button. This will cause a confirmation email to be sent to the specified email address. Open the confirmation email and click the "Confirm subscription" link

4. Send a message to your topic through the SNS web console
    a. On the main SNS console page, select "Topics" on the left side, and click the name of your topic
    b. Click the "Publish message" button
    c. Enter a subject for your message
    d. Enter a body for your message
    e. Click the "Publish message" button at the bottom
    f. Verify that you received an email containing the published message

5. Next, you will write a Typescript program that sends messages to your queue.

6. Create and initialize a new Node.js Typescript project

7. Install the AWS SDK

```
> npm install --save aws-sdk
```

8. Install AWS Lambda Typescript types

```
> npm install --save @types/aws-lambda
```

9. Create a Typescript file containing the following code, and update the region, message body, and topic ARN as needed. The topic ARN can be found in the SNS console.

```
// import AWS types
import * as AWS from 'aws-sdk';

// Configure AWS region.  *** YOU MIGHT NEED TO CHANGE THE REGION ***
AWS.config.update({region: 'us-west-2'});

// Create SNS object for doing SNS operations
let sns = new AWS.SNS({apiVersion: '2010-03-31'});

// Create publish parameters
let params = {
        Message: "*** PUT YOUR MESSAGE BODY HERE ***",
        TopicArn: "*** PUT YOUR TOPIC ARN HERE ***"
};

// Call publish
let publishPromise = sns.publish(params).promise();

publishPromise
        .then((data: AWS.SNS.Types.PublishResponse) => {
                console.log("Message ID: " + data.MessageId);
        })
        .catch((err: AWS.AWSError) => {
                console.error(err, err.stack);
        });
```

10. More information on accessing SNS from Javascript can be found here:
a. https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/sns-examples-publishing-messages.html

11. Run your program a few times, and verify that each published message is received by the email subscriber created earlier.

12. Next you will write a Typescript Lambda function that processes messages sent to your topic.

13. Your Lambda function needs to run with an IAM role that has SNS permissions. Modify the IAM role you created in the "Lambda / IAM" exercise to allow SNS access:
    a. Go to the IAM web console
    b. On the left, select "Roles"
    c. Goto the role you created previously for the "Lambda / IAM" exercise by clicking on its name (e.g., cs340lambda)
    d. Click the "Attach Policies" button
    e. On the "Attach Permissions" screen, attach the AmazonSNSFullAccess policy to your role. This will allow your Lambda function to access your SNS topics.

14. In your Typescript project, create a file named index.ts and add the following code to this file:

```
import * as AWS from 'aws-sdk';
import * as Lambda from 'aws-lambda';

export const handler = async (event: Lambda.SNSEvent) => {

        console.log('Entering topic_subscriber');

        event.Records.forEach((record: Lambda.SNSEventRecord) => {

                let msg: string = msg.Sns.Message;

                console.log(msg);
        });

        console.log('Leaving topic_subscriber');
}
```

15. Compile and bundle your Lambda code:
    a. Compile your code by running the "tsc" command. The output Javascript files should be in the dist/ directory. If they are not, add "outDir": "./dist/" to your tsconfig.json.
    ```
    > cd <project-directory>
    > tsc
    ```
    b. Copy your project's node_modules/ directory into the dist/ directory
    ```
    > cp -r node_modules/ dist/
    ```
    c. Create a ZIP file containing the contents of the dist/ directory. (It is important that the ZIP file contain the contents of the dist/ directory, but not the dist/ directory itself):
    ```
    > cd dist/
    > zip -r dist.zip *
    ```

16. Next, deploy your Lambda function.
    a. Login to the AWS console
    b. Navigate to the Lambda service
    c. Select "Functions" on the left side of the Lambda screen
    d. Click the "Create Function" button in the top-right corner
    e. Give your function a name (e.g., topic_subscriber)
    f. As your function's Runtime select "Node.js 8.10"
    g. Under Permissions, click "Choose or create an execution role"
    h. In the "Execution role" drop-down, select "Use an existing role"
    i. In the "Existing role" drop-down, select the name of the IAM role that you created in a previous step (e.g., cs340lambda)
    j. Click the "Create function" button. This will actually create the function, and take you to a screen that lets you further configure the function.
    k. Scroll down to the "Function code" section of the function configuration screen.
    l. In the "Handler" field, specify the name of the file and function that contain your Lambda function handler (e.g., index.handler)
    m. Click the Function package "Upload" button and select the ZIP file created in the previous step (e.g., dist.zip)
    n. Scroll to the top, and click the "Save" button. This will deploy your Lambda function, which means it can now be called

17. Next, subscribe your Lambda function to your SNS topic, as follows:

18. Go back to the SNS web console

19. Select "Topics" on the left side

20. Click on the name of your topic

21. Select the “Subscriptions” tab

22. Click the “Create subscription” button

23. For Protocol select “AWS Lambda”

24. For Endpoint select the Lambda function you previously created (e.g., topic_subscriber)

25. Click the “Create subscription” button

26. Now, whenever a message is published to your SNS topic, the message will be sent to your Lambda function for processing, as well as to the email subscriber created previously. Verify that this is happening by running your Typescript program that publishes messages to your topic, or publish messages manually through the SNS console.

27. Submit your Typescript code through Canvas.

More Information

The full AWS SDK documentation can be found here in the SDKs section:
https://aws.amazon.com/tools/