Exercise: SQS - Typescript

In this exercise you will:

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

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 Queue Service (SQS) web console

2. Create an SQS queue
   a. Click the "Create New Queue" button
   b. Enter a name for your queue
   c. Select "Standard Queue"
   d. Click the "Quick-Create Queue" button

3. Send a message to your queue through the SQS web console
   a. On the main SQS console page, select the new queue by checking the box next to its name
   b. In the "Queue Actions" menu, select "Send a Message"
   c. Type some text in the message body
   d. Click the "Send Message" button
   e. Click "Close"

4. Verify that the message was sent
   a. ON the main SQS console page, select your queue
   b. In the "Queue Actions" menu, select "View/Delete Messages"
   c. Click "Start Polling for Messages"
   d. You should see the message you sent in the queue (click More Details to see more details)
   e. Click "Close"

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 queue URL as needed. The queue URL can be found in the SQS console (in the Details tab).

```
// import SQS
import { SQS } from "aws-sdk";

// Create SQS object for doing SQS operations *** YOU MIGHT NEED TO CHANGE
THE REGION ***
let sqs = new SQS({ apiVersion: "2012-11-05", region: "us-west-2" });

// Create sendMessage parameters
let params = {
        MessageBody: "*** PUT YOUR MESSAGE BODY HERE ***",
        QueueUrl: "*** PUT YOUR QUEUE URL HERE ***",
        DelaySeconds: 5
};

// Call sendMessage
sqs.sendMessage(params, (err: AWS.AWSError, data:
AWS.SQS.Types.SendMessageResult) => {
        if (err) {
                console.log("Error", err);
        }
        else {
                console.log("Success", data.MessageId);
        }
});
```

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

11. The online documentation for the AWS Javascript SDK can be found here:
a. https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/
    b. Look at the documentation for the SQS class

12. Run your program a few times, and then go to the SQS console to verify that the messages sent by your program are showing up in the queue.

13. Next you will write a Typescript Lambda function that processes messages sent to your queue.

14. Your Lambda function needs to run with an IAM role that has SQS permissions. Modify the IAM role you created in the "Lambda / IAM" exercise to allow SQS 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 AmazonSQSFullAccess policy to your role. This will allow your Lambda function to access your SQS queues.

15. In your Typescript project, create a file named index.ts and add the following code to this file, and install the imported packages as shown in the example code:

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

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

        console.log('Entering queue_processor');

        event.Records.forEach((msg: Lambda.SQSRecord) => {

                // TODO:
                // Add code to print message "body" to the log

        });

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

16. Fill in the code that logs the message body.

17. 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 *
    ```

18. 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., queue_processor)
    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

19. Next, connect your Lambda function to your SQS queue, as follows:

20. Go back to the SQS web console

21. Select your queue by checking the box next to its name

22. In the "Queue Actions" menu, select "Configure Trigger for Lambda Function"

23. Select the name of the Lambda function you previously created (e.g., queue_processor)

24. Click the "Save" button

25. Now, whenever a message is sent to your SQS queue, the message will be sent to your Lambda function for processing. Verify that this is happening by running your Typescript program that sends messages to your queue, or send messages manually through the SQS console.

26. IMPORTANT: After completing the exercise, remove the lambda trigger from your queue, because SQS constantly polls the queue to detect new messages so it can call the lambda appropriately. All of this polling adds to your AWS charges, so you will want to remove the lambda trigger to avoid this.

27. Submit your Typescript code through Canvas.

More In formation

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