# Exercise: SNS - Java

In this exercise you will:

1. Create an SNS topic
2. Write a Java program that sends messages to your topic
3. Write a Java 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.

Intellij is 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 Java program that sends messages to your queue.

6. Create a directory for this exercise

7. Create a new Intellij project

8. In the new project, add dependencies on the AWS SDK Java library

a. Select the File -> Project Structure menu
b. Select “Modules” on the left side of the “Project Structure” dialog
c. Select the “Dependencies” tab
d. Click the “plus” icon in the top-right
e. Select “Library”
f. Select “From Maven”
g. Enter “com.amazonaws:aws-java-sdk-core:1.11.547”, and click OK. This will add a dependency on the core AWS Java SDK library.
h. Using the same process, add a Maven dependency on “com.amazonaws:aws-java-sdk-sns:1.11.547”. This will add a dependency on the SNS portion of the AWS Java SDK library.

9. Create a class named SnsPublisher containing the following code. The topic ARN can be found in the SNS console.

```
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.PublishResult;

public class SnsPublisher {
  public static void main(String[] args) {

    String messageBody = "*** PUT YOUR MESSAGE BODY HERE ***";
    String topicArn = "*** PUT YOUR TOPIC ARN HERE ***";

    PublishRequest publishRequest = new PublishRequest()
            .withTopicArn(topicArn)
            .withMessage(messageBody);

    AmazonSNS sns = AmazonSNSClientBuilder.defaultClient();
    PublishResult publishResponse = sns.publish(publishRequest);

    String msgId = publishResponse.getMessageId();
    System.out.println("Message ID: " + msgId);
  }
}
```

10. More information on accessing SNS from Java can be found here:

a. https://docs.aws.amazon.com/sns/latest/dg/sns-tutorial-publish-message-to-topic.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 Java 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. To the IntelliJ project created earlier, add dependencies on the AWS SDK Java library
    a. Select the File -> Project Structure menu
    b. Select “Modules” on the left side of the “Project Structure” dialog
    c. Select the “Dependencies” tab
    d. Click the “plus” icon in the top-right
    e. Select “Library”
    f. Select “From Maven”
    g. Enter “com.amazonaws:aws-lambda-java-core:1.2.0”, and click OK. This will add a dependency on the Lambda portion of the AWS Java SDK library.
    h. Using the same process, add a Maven dependency on “com.amazonaws:aws-lambda-java-events:2.2.5”. This library contains the SNSEvent class, which will be used by your Lambda function.

15. In your project, create a Java package named “sns”.

16. In the “sns” package, create a class named TopicSubscriber that will contain the “handler” for your Lambda function. Add the following code to this file.

```
package sns;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;

public class TopicSubscriber implements RequestHandler<SNSEvent, Void> {

  @Override
  public Void handleRequest(SNSEvent event, Context context) {

     for (SNSEvent.SNSRecord record : event.getRecords()) {

        String msg = record.getSNS().getMessage();

        System.out.println(msg);
     }

     return null;
  }

}
```

17. Once your program compiles, add an “artifact” to your Intellij project that creates a JAR file containing your compiled code.
    a. Select the File -> Project Structure menu
    b. Select “Artifacts” on the left side of the “Project Structure” dialog
    c. To add a new artifact, click the “plus” button at the top of the dialog
    d. Select JAR -> From modules with dependencies
    e. In the “Create JAR from Modules” dialog, keep the default options, and click OK
    f. Check the “Include in project build” check box
    g. Rebuild the project, which will result in a JAR file being created in the “out/artifacts/<DIR>/ folder. This JAR file can be used to deploy your Lambda function to AWS

18. Next, deploy your Lambda function.
    a. Login to the AWS console
    b. Navigate to the Lambda service

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 “Java 8”
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 configured earlier (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 method that contains your Lambda function handler (e.g., sns.TopicSubscriber::handleRequest)
m. Click the Function package “Upload” button and select the JAR file created in the previous step
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, subscribe your Lambda function to your SNS topic, as follows:

20. Go back to the SNS web console

21. Select “Topics” on the left side

22. Click on the name of your topic

23. Select the “Subscriptions” tab

24. Click the “Create subscription” button

25. For Protocol select “AWS Lambda”

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

27. Click the “Create subscription” button

28. 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 Java program that publishes messages to your topic, or publish messages manually through the SNS console.
29. Submit your Java code through Canvas.

# More In formation

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