AWS Lambda Exercise

Re-submit Assignment

Due Feb 12 by 11:59pm   **Points** 2   **Submitting** a file upload   **File Types** java

---

Complete the exercise in Java.

In this exercise you will:

1. Write an AWS Lambda function that sends email messages using the AWS Simple Email Service (SES)
2. Create a JAR file containing your compiled Lambda function
3. Create an AWS IAM role that provides your Lambda function the security permissions it requires.
4. Deploy your Lambda function using your JAR file and IAM role
5. Test your Lambda function using a test event

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

- By default, to send emails using the Simple Email Service (SES), you must first "verify" the email addresses you intend to send emails to and from (verifying an email address means proving that you actually own the addresses).
    1. Login to the AWS console
    2. Navigate to the Simple Email Service
    3. Select "Email Addresses" on the left side of the SES screen
    4. Verify each of the email addresses you want to send SES emails to and from. Do this by clicking the "Verify a New Email Address" button.
- When you deploy an AWS Lambda, you must assign it an IAM role that specifies what security permissions the Lambda should have when it runs. For this purpose, create an IAM role that defines the security permissions your Lambda function should have when it executes.
    1. Login to the AWS console
    2. Navigate to the IAM service
    3. On the left of the IAM service screen, select "Roles"
    4. Click the "Create role" button

5. When asked to "Select type of trusted entity", select "AWS Services", and under "Choose the service that will use this role", select "Lambda" and click the Next button
6. On the "Attach permissions policies" screen, attach the AmazonSESFullAccess policy to your role. This will allow your Lambda function to send emails using the SES service. You can find the AmazonSESFullAccess role by entering "SES" in the search field.
7. Also attach the CloudWatchLogsFullAccess policy to your role. This will allow your Lambda function to log messages using the AWS CloudWatch service. After doing that, click the Next button.
8. Skip the "Add tags" screen by clicking the Next button
9. On the "Review" screen, enter a name for your role (e.g., cs340lambda or whatever you like), and click the "Create role" button

- Create a directory that will contain the code for this exercise
- In the new directory, create a new Intellij project and select Maven on the left hand side.

  Maven

  1. Do not select an archetype and click "Next".
  2. Name your project and select where it will save to.
  3. Click the drop down labeled "Artifact Coordinates". Set the group id to whatever you want your package structure to be. For example "edu.byu.cs340" will tell Maven where your main package is.
  4. Select "Finish".

- In the new project, add dependencies on the AWS SDK Java library
  1. Go to **https://mvnrepository.com/** **(https://mvnrepository.com/)** and search for the following:
     - aws-lambda-java-core
     - aws-java-sdk-ses
  2. Find the latest version of each dependency
  3. Copy the dependency for Maven.

  Maven

  1. Open pom.xml and add the following below the version tag:

```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

  2. Create a dependencies tag:

```
<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-lambda-java-core</artifactId>
        <version>1.2.0</version>
    </dependency>
```

```
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-ses</artifactId>
            <version>1.11.720</version>
        </dependency>
    </dependencies>
```

3. In between the dependencies tag paste what you copied for each of the dependencies.
4. On the right hand side, click the Maven tab, and then click the refresh button. Maven will now download and import the dependencies and everything else it needs. NOTE: This will take some time, depending on your internet connection it can take 5-10 minutes.

- In your project, create a Java package named "ses".
- In the "ses" package, create a class named "EmailRequest". In this class, create public properties for the different parts of an email message, including: "to", "from", "subject", "textBody", "htmlBody". All of these should be strings.
- In the "ses" package, create a class named "EmailResult". In this class, create a "message" property for returning the status of an email send operation, and a "timestamp" property for returning the date/time at which the email was sent. Both of these should be strings.
- In the "ses" package, create a class named EmailSender that will contain the "handler" for your Lambda function. Add the following code to this file. You might need to change the AWS region value if you're using a region other than us-west-2.

```java
package ses;

// these are the imports for SDK v1
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.simpleemail.*;
import com.amazonaws.services.simpleemail.model.*;
import com.amazonaws.regions.Regions;

public class EmailSender {
    public EmailResult handleRequest(EmailRequest request, Context context) {

        LambdaLogger logger = context.getLogger();
        logger.log("Entering send_email");

        try {
                AmazonSimpleEmailService client =
                AmazonSimpleEmailServiceClientBuilder.standard()

                // Replace US_WEST_2 with the AWS Region you're using for
                // Amazon SES.
                .withRegion(Regions.US_WEST_2).build();

                // TODO:
                // Use the AmazonSimpleEmailService object to send an email message
                // using the values in the EmailRequest parameter object
```

```
            logger.log("Email sent!");
        } catch (Exception ex) {
            logger.log("The email was not sent. Error message: " + ex.getMessage());
            throw new RuntimeException(ex);
        }
        finally {
            logger.log("Leaving send_email");
        }

        // TODO:
        // Return EmailResult
    }

}
```

- Fill in the code that actually creates and sends an email message using the values contained in the
  EmailRequest parameter object. Return an EmailResult containing a message indicating the result
  of the operation and the date/time at which the email was sent. The online documentation for the
  AWS Java SDK can be found here:

  1. **https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html**
     (https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html)

- Once your program compiles, add an "artifact" to your Intellij project that creates a JAR file containing
  your compiled code.

  1. Select the File -> Project Structure menu
  2. Select "Artifacts" on the left side of the "Project Structure" dialog
  3. To add a new artifact, click the "plus" button at the top of the dialog
  4. Select JAR -> From modules with dependencies
  5. In the "Create JAR from Modules" dialog, keep the default options, and click OK
  6. Check the "Include in project build" check box
  7. 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

***Note: AWS supports Java 8. You may be using a later versioned java compiler to build your jar. To
change your compiler version in Intellij, go to Preferences -> Build, Execution, Deployment -> Java
Compiler and then change "Project bytecode version" to 8.

- Next, deploy your Lambda function.

  1. Login to the AWS console

2. Navigate to the Lambda service

3. Select "Functions" on the left side of the Lambda screen

4. Click the "Create Function" button in the top-right corner

5. Give your function a name (e.g., send_email)

6. As your function's Runtime select "Java 8"

7. Under Permissions, click "Choose or create an execution role"

8. In the "Execution role" drop-down, select "Use an existing role"

9. In the "Existing role" drop-down, select the name of the IAM role that you created in a previous step (e.g., cs340lambda)

10. Click the "Create function" button. This will actually create the function, and take you to a screen that lets you further configure the function.

11. Scroll down to the "Function code" section of the function configuration screen.

12. In the "Handler" field, specify the name of the method that contains your Lambda function handler (e.g., ses.EmailSender::handleRequest)

13. Click the Function package "Upload" button and select the JAR file created in the previous step

14. Scroll to the top, and click the "Save" button. This will deploy your Lambda function, which means it can now be called

- Now, test your Lambda function. This can be done as follows:

  1. Scroll to the top and open the "Select a test event" drop-down

  2. Select "Configure test events". A test event is simply a JSON object that you will send to your Lambda function to see if it works.

  3. Specify a name for your test event (e.g., SendEmail)

  4. Fill in the test JSON object. Its attribute names should match the field names in the Java EmailRequest class created in a previous step. The JSON object's attributes should start with lower-case letters. (AWS will automatically deserialize the JSON object into an EmailRequest Java object)

  5. Click the "Create" button at the bottom. This will take you back to the Lambda function configuration screen.

  6. Execute your test event. Scroll to the top, select the name of your test event in the drop-down next to the "Test" button, and click the "Test" button. This will execute your Lambda function with the test event JSON object.

  7. The output of your function will be displayed in the console. For more details, you can click the "logs" link to view the complete CloudWatch logs for your Lambda function

Submission

- Submit the following .java files. DO NOT SUBMIT A .zip FILE.
  - File containing created "EmailRequest" class
  - File containing created "EmailResult" class
  - File containing created "EmailSender" class

- Submit your Java code **here**.