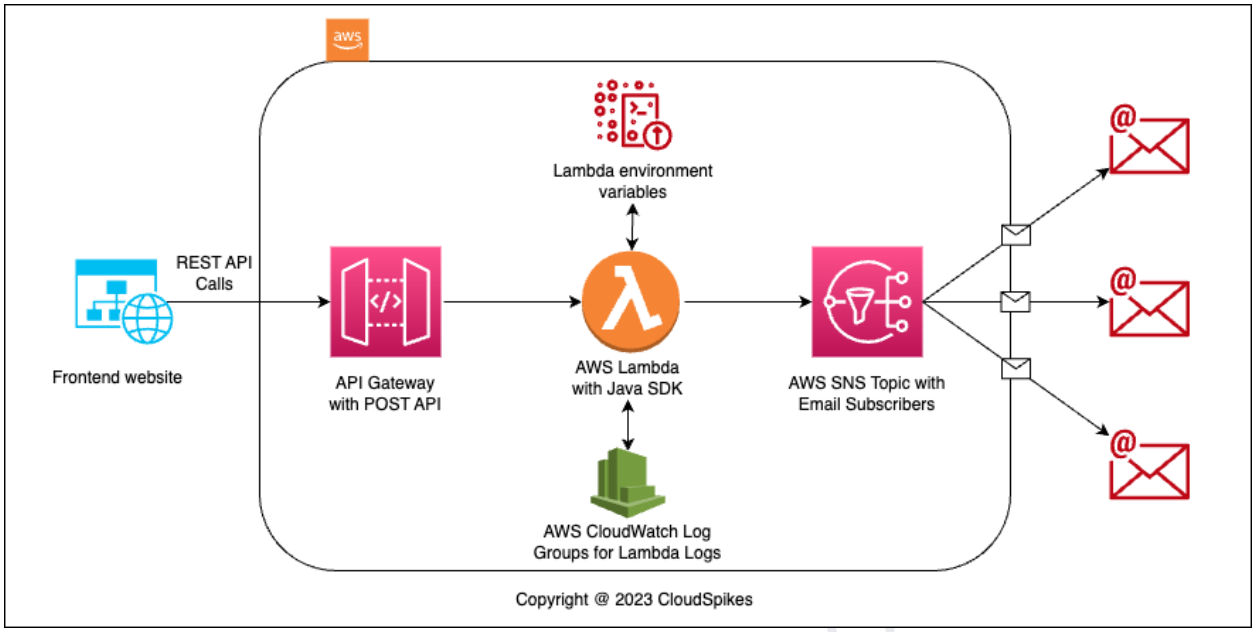
#9 Task: Prepare a Serverless Model Architecture that has Lambda with Java Code, API Gw integration, and SNS notifications using Email protocol.

1. **Here, the Lambda Java function will collect email adresses to send an email notification from the API Gw POST method.**
2. **Once, data is received, Lambda will publish email data to the SNS Topic and SNS will publish messages to the subscribed users via email protocol.**
3. **Configure IAM Roles for the AWS services as and where needed.**
4. **Use Lambda environment variables to inject an application credential (AK, SK, Region, Account ID, etc. as needed) to the running Java App on AWS Lambda Func.**

Monitor Logs from the CloudWatch Lambda-specific Log Group & Log Streams.



1. Java App Code - Create a Lambda Function with Java SDK.

- Open Eclipse and install AWS Toolkit to prepare AWS Lambda specific Java code using Maven build tool configurations using this AWS Official document.
- Now, from the menu bar, click on New → Project → Seach for "AWS Lambda" → Select **AWS Lambda Java Project** → Provide the name of the project and select the Input Type as **Stream Request Handler** → Hit Finish.
- This will create a new project in the project explorer section on left hand side with the project name folder you provided in the previous step.
- Now, refer to the source code from this GitHub Repository and prepare Java class files as well as pom.xml dependency file accordingly.
  Note: To skip the above steps, you can also simply import this GitHub Repository in to Eclipse IDE and compare the code with the GitHub code base to verify the Java and pom.xml files.
- FYI, this Java code will capture the input JSON data from the API Gateway POST API, process the input data as well as Create an SNS topic if does not exist and add subscribers. Also, this Java code will also publish the message received from the API Gateway POST API call on the configured SNS topic.
- Finally, to prepare the executable JAR build for this project, go to the terminal and navigate to the location of the AWS Lambda Java Project. Then. execute "**mvn clean install**" command. This will prepare the JAR file under **target** directory which will be uploaded to the AWS Lambda Function while deployment process will be executed. ***Note***: You can download the Java source code for this Lab work from this public GitHub Repo.

1. Infra Setup - Create a Lambda Function with Java SDK.

- Create a Lambda function and provide the basic details like Function name, Runtime as Java 11 (Corretto), Architecture type as x86_64 and Change default execution role option as Create a new role with basic Lambda permissions.
- Once, the Lambda is created with the basic details, go to Runtime settings and set Handler value as **com.amazonaws.lambda.demo.LambdaFunctionHandler::handleRequest**
- Here, **com.amazonaws.lambda.demo** is the Package value for the Java App, **LambdaFunctionHandler** is the Java Class name where Handler Function (App Java Code Entry Point from where Lambda execution code will start reading Java code from Line 0 to Line N - Last line of code) is defined and to define the Lambda Handler Function is distingueshed using :: (Double column). So, here, **handleRequest** is the Java Lambda Handler Function name.
- Furthermore, configure the environment variables needed for the Lambda function such as AWS_AK, AWS_SK, AWS_DEFAULT_REGION_VALUE and AWS_ACCOUNT_ID by navigating to **Configuration tab** → **Environment Variables** section.
- Deploy the JAR build of the Java App on the prepared AWS Lambda function by navigating to **Code tab** → **Code Source** section and selecting **Upload from** → **.zip or .jar file** and upload the built JAR file using Maven build tool from the build server/machine. In this case, we prepared JAR file on local env, so upload it from the local machine.

2. Infra Setup - Create a REST API with required Resource and Method.

- Create a REST API by going to the API Gateway AWS service.
- Define an API Resource like **test-send-email-notification** by clicking on Actions button and selecting the root / Resource.
- Once the Resource is defined, click on the created **test-send-email-notification** Resource and click on Create Method button via Actions button.
- Select a POST method for this API resource and then select integration type Lambda and map it with the created Lambda function in step 1.
- Finally, select the root / Resource and click on Actions button. Select Deploy API option and create/use the Stage to deploy the APIs defined.

3. Observe Email notifications on the subscribed users as well CloudWatch Logs from the Lambda specific Log Group and Log Stream.

- If there is a new Subcriber email address sent from the POST REST API, that email needs to be confirmed from the email inbox via email sent by Amazon SNS Service.
- Once, the subscribers are confirmed, they can start receiving email notification upon the POST API calls via a frontend website or a web API testing client like Postman tool.
- Also. Lambda execution logs can be observed by navigating to the respective Lambda function → Monitor tab → View CloudWatch Logs. This will open up a CloudWatch Log Group specific to the Lambda function you selected. Now, open up the Log Stream which you want to observe for the Java App Logs on Cloud Environment. FYI, you can go through the required App Logs by referring to the Log Stream's **Log event time** stamp value by comparing it with the time stamp when the API was called.