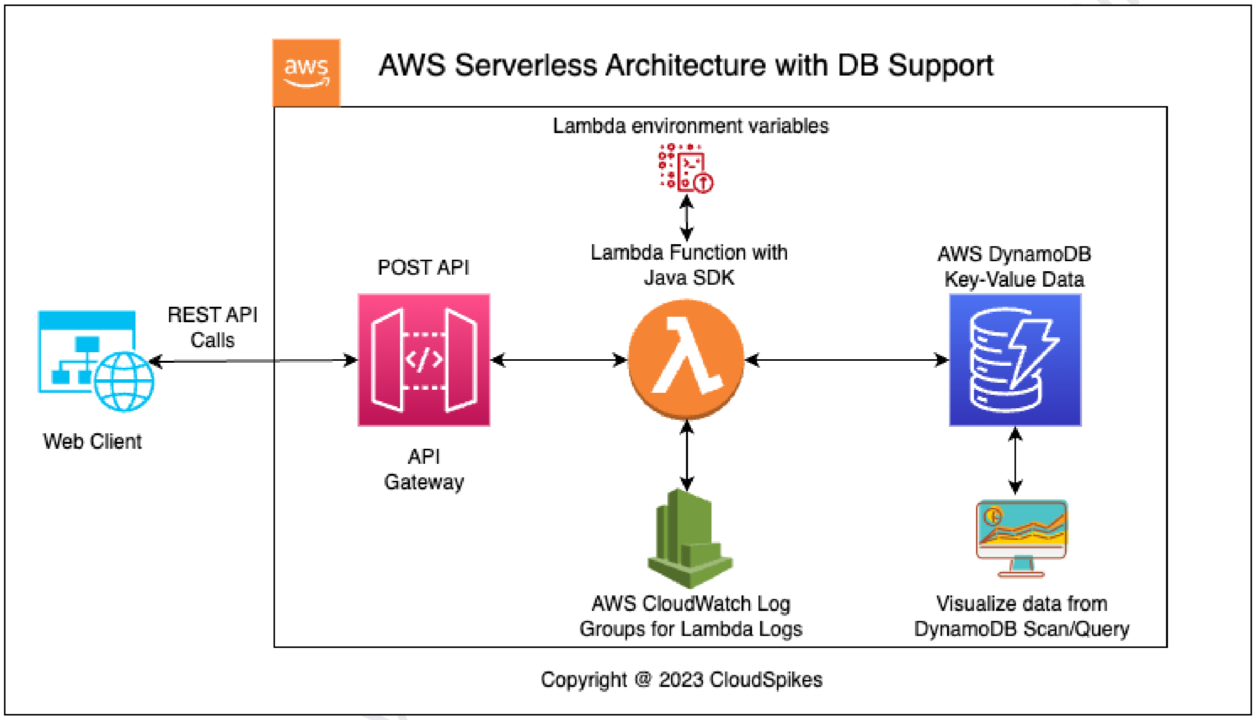
#11 Task: Create a POST API Gw API with Lambda that has a Java function to store data directly to the DynamoDB Instance with the latest stable version. Visualize the data stored in the DB using DynamoDB Dashboard.


AWS Serverless Architecture with DB Support
Lambda environment variables
POST API
Lambda Function with Java SDK
AWS DynamoDB Key-Value Data
REST API Calls
Web Client
API Gateway
AWS CloudWatch Log Groups for Lambda Logs
Visualize data from DynamoDB Scan/Query
Copyright @ 2023 CloudSpikes

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 DynamoDB Table if does not exist and add data inputs received from API Gateway POST API call.
- 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 and AWS_SK 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-dyanmodb-ops** by clicking on Actions button and selecting the root / Resource.
- Once the Resource is defined, click on the created **test-dyanmodb-ops** 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.
- You can test this deployed Stage Invoke API via a web client tool such as Postman.