## Lab 6-02: Using AWS Lambda With Amazon Kinesis Data Stream

### Service Introduction

AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers. Amazon Kinesis Data Streams is a service for real-time data streaming, enabling you to continuously capture and store terabytes of data per hour from hundreds of thousands of sources. Combining AWS Lambda with Kinesis Data Streams allows you to process and analyze streaming data in real-time with minimal infrastructure management.

### Problem

You need to process a high volume of streaming data from sources such as IoT devices, application logs, or social media feeds. Your current infrastructure struggles to handle real-time data processing, leading to delays and inefficiencies in data analysis and decision-making.

### Solution

Integrate AWS Lambda with Amazon Kinesis Data Streams to create a real-time data processing pipeline. Kinesis Data Streams will capture and store the streaming data, and AWS Lambda will process this data in real-time as it arrives. This setup allows you to run custom code in response to streaming data, enabling immediate data transformation, analysis, and actions without the need for managing servers or scaling infrastructure.

#### Task 1: Create a CloudFormation Stack

|  |
| --- |
| 1. In the console, navigate to **CloudFormation.**      1. Click **Create Stack** and select With new resources (standard) from the drop down menu.      1. Under Specify template, select **Upload** a template file. 2. Open a new browser tab, navigate to the lab GitHub repo, and download the CFN Build.yaml file. 3. Select **Choose File**, click **Upload,** then navigate to your CFN Build.yaml file, or drag the file into the browser and click **Next.** 4. Name the stack **Kinesislab** and click **Next.**      1. Keep all default settings and click **Next.**      1. Scroll to the bottom, check the three acknowledgment boxes, and click **Submit.** |

#### Task 2: Create an Amazon Kinesis Data Stream

|  |
| --- |
| 1. In the console, navigate to Kinesis.      1. Click the **Create data stream** button.      1. Name the Data Stream input-stream and select **On-demand** capacity mode.      1. Keep all other default settings and click **Create data stream.** |

#### Task 3: Create a DynamoDB Table

|  |
| --- |
| 1. In the console, navigate to **DynamoDB.**      1. Click the **Create table** button.      1. Name the table **kinesisAggs.** 2. Under the Partition key section, add the key name **vendorId** and change the type from **String** to **Number**.      1. Leave all other default settings and click **Create table.** |

#### Task 4: Populate Data on the Cloud9 Instance

|  |
| --- |
| 1. Navigate to the **Cloud9** Section of the console.      1. Next to the KinesisRealTimeStreaming-Kinesislab environment, under Cloud9 IDE, click the Open link. 2. Open a new browser tab and navigate to the lab1.py data creation file in the lab GitHub repo. 3. Select all and copy the contents of the lab1.py file.      1. Navigate back to the Cloud9 IDE tab. From the left file tree, right-click on KinesisRealTimeStreaming-Kinesislab. Select New File and, name it lab.py and hit Enter. Click on the new lab.py file to open it. 2. Paste the content copied from the GitHub repo into the newly created file.      1. Save the new lab.py file using ctrl + s for Windows or command + s for Mac. 2. Click on the command line at the bottom of the IDE screen. 3. Type in pip install boto3 and, hit Enter, and wait for installation to complete. 4. Run the lab.py file by typing in the command line: python lab.py and hit Enter. |

#### Task 5: Create and Configure a Lambda Function

|  |
| --- |
| 1. In the console, search and navigate to Lambda.      1. Click Create function.      1. Name the function kinesisLambdaConsumer01. 2. Under Basic information, set Runtime to Python 3.9.      1. Expand Change default execution role and select Use an existing role and select KinesisLambdaConsumerRole-Kinesislab. 2. Click Create function.      1. Download the lambda-deploy.zip file from the lab GitHub repo.      1. Navigate back in the console, and in the **Code source** section of the Lambda function, click **Upload from** and select the **lambda-deploy.zip** file you downloaded, or drag the file into the browser, then click **Save**.      1. Copy the **dynamoDBTableName** variable. 2. Under the **Configuration** section of the lambda function, go to Environment variables. 3. Click **Edit** and add the environment variable.      1. Paste the **dynamoDBTableName** in the Key section of the variable. 2. Add the **kinesisAggs DynamoDB** table to the Value section and click Save.      1. From the left menu in the Lambda **Configuration** tab, click General configuration and select **Edit.**      1. Increase timeout from 3 seconds to 1 minute and click **Save.**      1. In the Lambda Function overview section, click the **Add trigger** button.      1. Select **Kinesis** for the source and select input-stream for the Kinesis Stream. 2. Increase batch size to 1000.      1. Increase batch window to 120. 2. Under Additional settings, set Retry attempts to 2.      1. Change the tumbling window duration to 30. 2. Click **Add.** |

#### Task 6: Confirm Data is Populating to DynamoDB

|  |
| --- |
| 1. In the console, navigate back to the **DynamoDB** section.      1. Click on the **kinesisAggs** table.      1. Click Explore table items.      1. Under Scan or query items, click Run.      1. Confirm that you have at least two entries in your table.   **Note:** If you do not see data after 5 minutes, your Python script may have timed out. Navigate back to the Cloud9 IDE, rerun the **python lab.py** command, and data should start flowing through in about 3–5 minutes. |