Exercise 9.2 - Use IoT Analytics to Enhance and Look at the Data

In this exercise, you will use the AWS IoT Analytics service to visualize the data sent by the cars.

To do so, you will create a Data store, a Channel and a Pipeline. You will create an IoT Rule with the following query statement: *SELECT* FROM 'lab/greengrass/telemetry'*. You will instruct this Rule that selects everything from the* lab/greengrass/telemetry* IoT Topic to send the data to the Channel. So the data will be first sent into the Channel, the Pipeline will process the data and store it in the Data store.

One of the requirements from your Data Analysts is that they would also like to get the weather information and the ZIP Code from the location of the cars. Instead of doing a code modification to the cars, you have decided to do that using IoT Analytics.

To solve that requirement, you will create a Lambda function that takes the data as an input and adds that information. You will then modify the Pipeline to add the Lambda function as a step to execute with the streaming data.

To validate that the enrichment is there, you will create a Data set to analyze the streamed data using a SQL query.

Finally, you will act as the Data Analyst to execute different queries on the telemetry data.

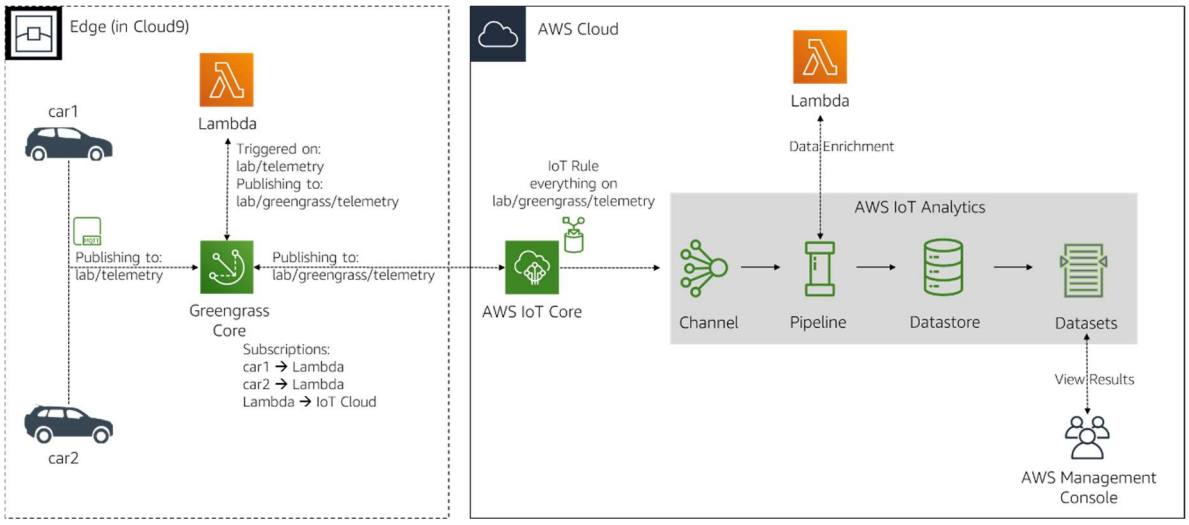

Ilustración 1 The diagram below shows the resources and data flow that you will create in this exercise.

1. Create the Lambda function for enrichment

In this section, you will create the Lambda function to enrich the data that will be used by the IoT Analytics Pipeline later in this exercise.

1.1 Create the Lambda function

1. In the AWS Management Console, click **Services**, and then click **Lambda** to go to the Lambda console.
2. Make sure you are in the same **Region** as the one you used in Exercise 1.1. It should be **Frankfurt, Ireland, N. Virginia, Ohio, Oregon or Tokyo**. You can validate that by going to the Cloud9 service and looking for the IoTOnAWS environment. If you don't see it, you aren't in the right region.
3. Click **Create function**.
4. For **Function name**, enter `labIoTAnalyticsLambda`.
5. For **Runtime**, select **Node.js 12.x**.
6. Expand **Choose or create an execution role** under **Permissions**.
7. Select the radio button next to **Create a new role from AWS policy templates**.

8. For **Role name**, enter `labIoTAnalyticsLambdaRole`. Leave the *Policy templates* empty as you are creating a Role with no specific permissions other than logging.
9. Click **Create function**.
10. Download the code of the Lambda function to your computer by clicking here.

    https://aws-tc-largeobjects.s3.amazonaws.com/OTP-AWS_D5-2019/v1.0/code/exercise-4.1-lambda.zip

11. In the **Function code** section, click **Actions** and select **Upload a .zip file**.
12. Click **Upload**, select the **exercise-4.1-lambda.zip** you just downloaded and click **Save**.
13. Scroll down to the **Basic settings** section and click **Edit**.
14. Under **Timeout**, input **30** sec and click **Save**.

    The code has been uploaded and Lambda will show the code in the browser. Feel free to review the it. In summary, it uses a library to get the ZIP Code from the longitude and latitude and uses that ZIP Code to get the weather for that location. It then adds the data into the input payload and sends it back.

1.2 Start Cloud9

Your Cloud9 environment may have shut down at this point as it's supposed to automatically shutdown after 30 minutes. To restart it, follow these steps:

1. In the AWS Management Console, click **Services**, and then click **Cloud9** to go to the Cloud9 console.
2. You should see a list of *environments*. If you don't, click on the hamburger menu icon (the three parallel lines) near the top left of the screen and click on **Your environments**.
3. Click the **Open IDE** button in the **IoTOnAWS** card.
4. It may take a minute for your environment to start.

1.3 Set Function Policy for IoT Analytics

1. In the Cloud9 **terminal** use the following command to allow IoT Analytics to invoke your Lambda function.

```
aws lambda add-permission --function-name labIoTAnalyticsLambda --
statement-id iot --principal iotanalytics.amazonaws.com --action
lambda:InvokeFunction
```

You should see an output similar to:

```
{
    "Statement": "{\"Sid\":\"iot\",\"Effect\":\"Allow\",\"Principal\":{\"
Service\":\"iotanalytics.amazonaws.com\"}...
}
```

2. Start Greengrass and the cars

Data needs to flow into IoT Core to make it easier on the creation of the different elements in the next section. You will start you Greengrass Core first and the car right after.

2.1 Start Greengrass

1. Start Greengrass by executing the following commands in the Cloud9 **terminal**:

```
cd /greengrass/ggc/core/
```

```
sudo ./greengrassd start
```

2.2 Start the cars

1. If you **don't** have 2 Cloud9 **terminal**. click the **circled +** icon that is next to your current terminal and select **New Terminal**. You now have 2 different terminals.
2. In the left terminal, execute the following commands to start the code for car1.

```
cd ~/environment/car1
```

```
node exercise-3.2.js
```

3. In the right terminal, execute the following commands to start the code for car2.

```
cd ~/environment/car2
```

```
node exercise-3.2.js
```

3. Create the stream Analytics Pipeline

3.1 Create the Channel

This channel will receive the data from IoT Core.

1. In the AWS Management Console, click **Services**, and then click **IoT Analytics** to go to the IoT Analytics console.
2. Click **Channels** in the left menu.
3. Click **Create a channel**.
4. Under **Channel ID**, enter `labIoTChannel`.
5. Under **Storage type**, select **Service-managed store**.
6. Click **Next**.
7. Under **IoT Core topic filter**, enter `lab/greengrass/telemetry`. This will automatically create an IoT Rule with the action to send the data it finds with the query statement `SELECT * FROM 'lab/greengrass/telemetry'` to the *labIoTChannel*.
8. Under **IAM role name**, click **Create new**.
9. Under **Name**, enter `labIoTAnalyticsRole`.
10. Click **Create role**.
11. Click **Create Channel**.

3.2 Create the Data store

The Data store will store the data received by the Pipeline once it's processed.

1. Click **Data stores** in the left menu.
2. Click **Create a data store**.
3. Under **ID**, enter `labIoTDatastore`.

4. Under **Storage type**, select **Service-managed store**.
5. Click **Create data store**.

3.3 Create the Pipeline

The Pipeline will take the data arriving in the Channel, process it and send it into the Data store. Although, the Pipeline won't do much processing as of yet.

1. Click **Pipelines** in the left menu.
2. Click **Create a pipeline**.
3. Under **Pipeline ID**, enter `labIoTPipeline`.
4. Under **Pipeline source**, click the **Edit** link and select **labiotchannel**.
5. Click **Next**.
6. In this *Set attributes of your messages* menu, you see a list of attributes that were found from the data that the cars are currently sending. Click **Next**.
7. In the *Enrich, transform, and filter messages*, click the **Add activity** link.
8. Click **Transform message with Lambda function**.
9. Under **Lambda function**, select **labIoTAnalyticsLambda**.
10. Click **Update preview**. You will see the same attributes as earlier with 2 more attributes that you can see at the bottom of the list: weather and zipcode. These are added by the Lambda function.
11. Click **Next**.
12. Under **Pipeline output**, click **Edit** and select **labiotdatastore**.
13. Click **Create pipeline**.

4. Create your first Data set and look at results

In this section, you will first create an initial IoT Analytics Data set with a SQL Query that will select everything in the data store. Then, you will run the query to see the results.

4.1 Create the initial AWS IoT Analytics Data set

1. In the AWS Management Console, click **Services**, and then click **IoT Analytics** to go to the IoT Analytics console.
2. Click **Data sets** in the left menu.
3. Click **Create a data set**.
4. Click **Create SQL** to create a SQL Data Set so you can issue SQL Queries.
5. Under **ID**, enter `labIoTDataset`.
6. Under **Select data store source**, click **Edit** and select **labiotdatastore**.
7. Click **Next**.
8. Leave the SQL Query to `SELECT * FROM labiotdatastore` to show all of the data and click **Next**.
9. Click **Next** under the **Configure data selection filter** page.
10. Click **Next** under the **Set query schedule** page.
11. In the years and days fields, enter `0` and `1` respectively. It should read: For 0 year(s) and 1 day(s).
12. Click **Next**.
13. Click **Create data set**.

4.2 Run the AWS IoT Analytics Data set

1. Click the **labiotdataset** link.
2. Click **Actions > Run now**.
3. Under the **labiotdataset** title, you will see the word **CREATING**. Wait until it changes to **SUCCEEDED** which may take a few minutes. You will see some data in the **Result preview**. You can find the weather and zipcode attributes that were added via the Pipeline that uses Lambda.
4. Click **Content**.
5. You should see a green light with the word *Succeeded* next to the only query you ran so far. You can also download the *csv* file by clicking on the *Download* link.

5. Analyze Time Series data with IoT Analytics

In this section, you will act as a Data Analyst to issue other SQL queries to find information about the data.

5.1 Look at the mean speed of cars that exceeds the limit in a specific location and weather

1. Click the **Details** in the labiotdataset left menu.
2. Click **Edit** next to **SQL query**.
3. Replace the content of the **Query** with

```
SELECT device, weather, engine_speed_mean, zipcode FROM labiotdatastore
WHERE engine_speed_mean > 60
```

4. Click **Save**.
5. Click **Actions > Run now**.
6. Wait for the query to succeed. The status is visible under the *labiotdataset* title. It will go from CREATING to SUCCEEDED. You will see the result in the **Result preview** section which will be for only the device, weather, engine_speed_mean and zipcode columns for the cars with an engine_speed_mean above 60.

5.2 Lookup the number of trips per vehicle.

1. In the **Details** menu of the labiotdataset, click **Edit** next to **SQL query**.
2. Replace the content of the **Query** with

```
SELECT device, count(*) AS count FROM labiotdatastore GROUP BY device
```

3. Click **Save**.
4. Click **Actions > Run now**.
5. Wait for the query to succeed and you will see how many data entries per car there are at this point in the **Result preview** section.

Feel free to try other SQL statements.