

**GEORGIA INSTITUTE OF TECHNOLOGY**  
**SCHOOL of ELECTRICAL and COMPUTER ENGINEERING**

**ECE 4150-A Spring 2021**

**Lab: Air Quality Monitoring Application using AWS IoT, DynamoDB,  
Lambda, CloudWatch and EC2**

---

**References:**

- [1] A. Bahga, V. Madisetti, "Cloud Computing Solutions Architect: A Hands-On Approach", ISBN: 978-0996025591
- [2] <https://aws.amazon.com/documentation/>
- [3] Guidelines for the Reporting of Daily Air Quality - the Air Quality Index (AQI), <https://archive.epa.gov/ttn/ozone/web/pdf/rg701.pdf>

**Due Date:**

The lab report will be **due on March 17, 2021 at 11:59 PM.**

---

In this lab, we will create an Air Quality Monitoring application.

The application uses the following:

1. AWS IoT thing to receive data from IoT devices and send to DynamoDB
2. DynamoDB tables to store raw pollutants data and computed AQI data
3. Lambda function to compute AQI from raw pollutants data.
4. Cloudwatch time-based event to trigger Lambda function to compute AQI every minute or hour
5. A web interface implemented in Django to visualize raw pollutants data and computed AQI

The **Air Quality Monitoring System** uses air quality monitoring nodes (IoT devices) that measure concentrations of various pollutants such as PM2.5, PM10, CO, SO<sub>2</sub>, O<sub>3</sub>, and NO<sub>2</sub>. Each node is known by an identifier called **stationID**. The IoT devices publish the raw data on pollutant concentrations to AWS IoT. Within AWS IoT, a rule to send the data to a DynamoDB table is created. The raw information is stored in a DynamoDB table. A Lambda function is used to compute average concentrations of the pollutants over 8 or 24 hours and then compute the Air Quality Index from average pollutant concentrations using the formula as shown below:

$$I_p = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}}(C_p - BP_{Lo}) + I_{Lo}.$$

Where  $I_p$  = the index for pollutant p

$C_p$  = the rounded concentration of pollutant p

$BP_{Hi}$  = the breakpoint that is greater than or equal to  $C_p$

$BP_{Lo}$  = the breakpoint that is less than or equal to  $C_p$

$BP_{Hi}$  = the breakpoint that is greater than or equal to  $C_p$

$I_{Hi}$  = the AQI value corresponding to  $BP_{Hi}$

$I_{Lo}$  = the AQI value corresponding to  $BP_{Lo}$

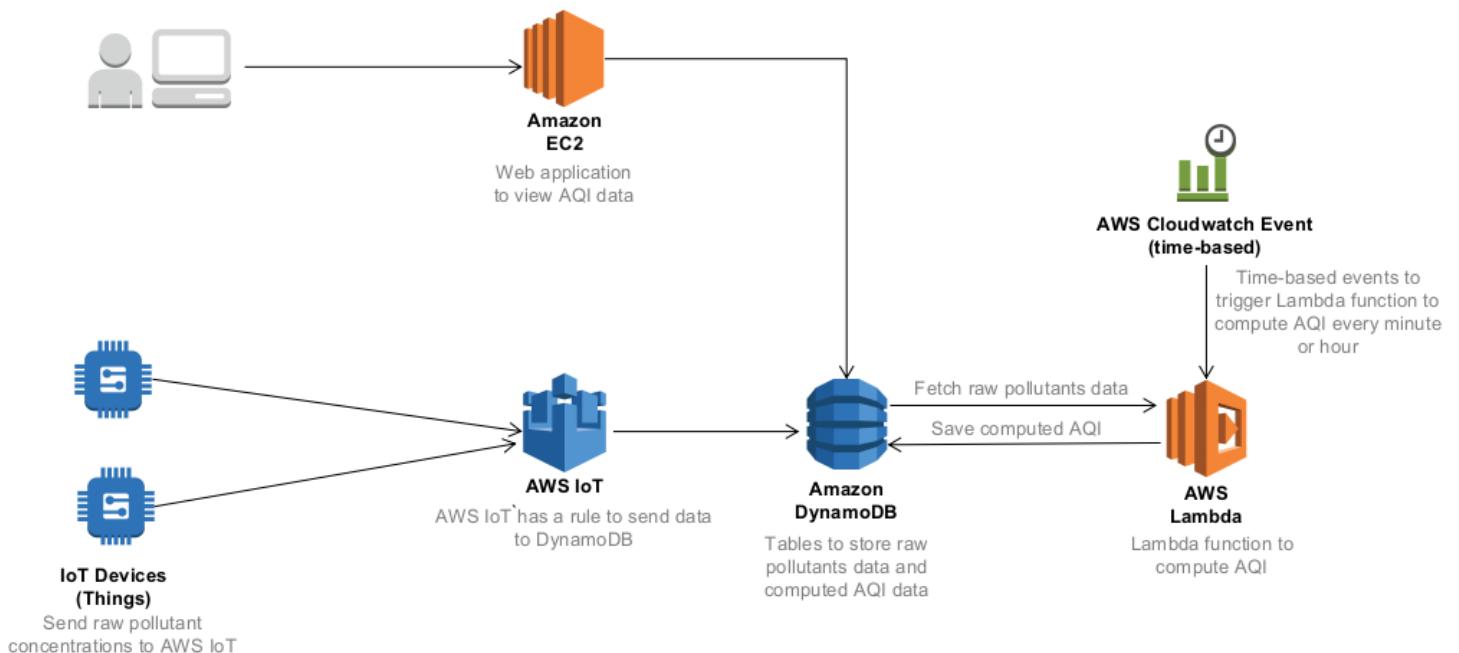


Fig.1 Architecture diagram of the Air Quality Monitoring application showing AWS services used

The breakpoint values used for AQI computation are shown in the table below.

This Breakpoint...						...equal this AQI		...and this category
O <sub>3</sub> (ppm) 8-hour	O <sub>3</sub> (ppm) 1-hour1	PM <sub>10</sub> (µg/m <sup>3</sup> )	PM <sub>2.5</sub> (µg/m <sup>3</sup> )	CO (ppm)	SO <sub>2</sub> (ppm)	NO <sub>2</sub> (ppm)	AQI	
0.000 - 0.064	-	0 - 54	0.0 - 15.4	0.0 - 4.4	0.000 - 0.034	( <sup>2</sup> )	0 - 50	Good
0.065 - 0.084	-	55 - 154	15.5 - 40.4	4.5 - 9.4	0.035 - 0.144	( <sup>2</sup> )	51 - 100	Moderate
0.085 - 0.104	0.125 - 0.164	155 - 254	40.5 - 65.4	9.5 - 12.4	0.145 - 0.224	( <sup>2</sup> )	101 - 150	Unhealthy for Sensitive Groups
0.105 - 0.124	0.165 - 0.204	255 - 354	65.5 - 150.4	12.5 - 15.4	0.225 - 0.304	( <sup>2</sup> )	151 - 200	Unhealthy
0.125 - 0.374 (0.155 - 0.404) <sup>4</sup>	0.205 - 0.404	355 - 424	150.5 - 250.4	15.5 - 30.4	0.305 - 0.604	0.65 - 1.24	201 - 300	Very unhealthy
( <sup>3</sup> )	0.405 - 0.504	425 - 504	250.5 - 350.4	30.5 - 40.4	0.605 - 0.804	1.25 - 1.64	301 - 400	Hazardous
( <sup>3</sup> )	0.505 - 0.604	505 - 604	350.5 - 500.4	40.5 - 50.4	0.805 - 1.004	1.65 - 2.04	401 - 500	Hazardous

Table.1 Breakpoint values used for AQI computation

The AQI is determined by the pollutant with the highest index. For example, if the PM2.5 AQI is 135, the PM10 AQI is 40, SO2 is 30, CO is 50, and all other pollutants are less than 135, then the AQI is 135, determined only by the concentration of PM2.5.

Follow the steps below to setup the air quality monitoring application in your AWS account.

## 1. Create DynamoDB Tables (5 points)

- Create two DynamoDB tables named **AirQualityData** and **AirQualityDataOutput** to store raw pollutant concentration data and computed AQI values.
- For these tables, use **stationID**, type *String*, as partition key, and **timestamp**, type *Number*, as sort key.

## 2. Create IAM Role (5 points)

- Create a new IAM role for a **Lambda Function** named **lambda\_dynamodb\_access**. Attach the policy that allows you full access to your DynamoDB.

## 3. Create IAM User (5 points)

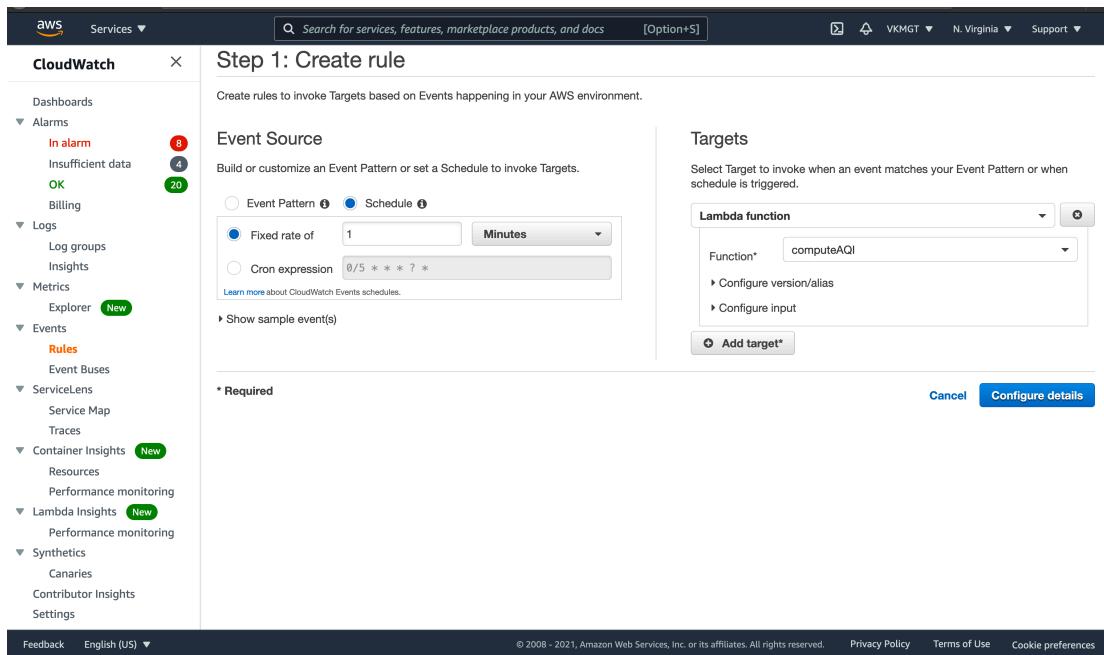
- Create a new IAM User for a for the Django dashboard in the EC2 instance named **Django\_user**. Attach the policy that allows you full access to your DynamoDB. Save the **Access key ID** and **Secret access key**.

## 4. Create Lambda function (5 points)

- Create a new **Lambda Function** for computing AQI from raw pollutant concentrations and saving the AQI values to DynamoDB.
- Create a function for **Python 2.7** and call it **computeAQI** using the code provided as `lambda_function.py`, attach the **role** created in the previous step.

## 5. Create CloudWatch Event (5 points)

- From **CloudWatch** console, go to **Events**, and create a new **event rule** to trigger the Lambda function created in the previous step in a **schedule** pattern at a rate of 1 minute. Assign the lambda function to the event and name it **computeAQI\_rule**. An example is shown below.



- Navigate back your lambda function, you should be able to see a CloudWatch event been added.

## 6. Create IoT Core (10 points)

- Navigate to AWS IoT core console and provision a thing using the Connect to AWS IoT wizard

The screenshot shows the AWS search results for 'iot core'. The search bar at the top contains 'iot core'. Below it, there are two main sections: 'Services' and 'Features'. In the 'Services' section, 'IoT Core' is highlighted with a blue border, showing its description: 'Connect Devices to the Cloud'. Other services listed include 'AWS IoT Core for LoRaWAN', 'IoT Analytics', and 'IoT Events'. In the 'Features' section, there are three items: 'Quick start', 'License configurations', and 'SiteWise Monitor'. A sidebar on the left lists categories like 'Services (17)', 'Features (3)', 'Documentation (187,438)', and 'Marketplace (31)'. A right-hand sidebar provides information about additional regions and mobile app support.

The screenshot shows the new AWS IoT console experience. The left sidebar has a tree view with 'AWS IoT' selected. Under 'Onboard', the 'Get started' option is circled in red. The main content area features a large 'AWS IoT' logo and a descriptive paragraph: 'AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices.' Below this are three sections with icons: 'Connect and manage your devices' (a car and a windmill), 'Process and act upon device data' (a car and a bar chart), and 'Read and set device state at any time' (a car and a smartphone). A note at the bottom states: 'AWS IoT stores the latest state of a device'.

Screenshot of the AWS IoT Connect to AWS IoT page. The left sidebar shows navigation options like Monitor, Activity, Onboard, Manage, Fleet Hub, Greengrass, Wireless connectivity, Secure, Defend, Act, Test, Software, Settings, Learn, and Documentation. A banner at the top says "Introducing the new AWS IoT console experience" and "We're updating the console experience for you. Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu." The main content area is titled "Connect to AWS IoT" and describes the three steps: Register a device, Download a connection kit, and Configure and test your device. Each step has an icon and a brief description. At the bottom, there's a "Get started" button and a link to "Try the interactive overview".

- Select Linux/OSX if you are using a MacOS or Linux based computer, otherwise select Windows.

Screenshot of the AWS IoT How are you connecting to AWS IoT? page. The left sidebar is identical to the previous screenshot. The main content area asks "How are you connecting to AWS IoT?" and "Select the platform and SDK that best suits how you are connecting to AWS IoT." It shows two main sections: "Choose a platform" with "Linux/OSX" selected and "Windows" as an option, and "Choose a AWS IoT Device SDK" with "Python" selected and "Node.js" and "Java" as other options. Below these, it says "Some prerequisites to consider: the device should have Python and Git installed and a TCP connection to the public internet on port 8883." At the bottom, it says "Looking for AWS IoT Device SDKs and documentation? View AWS IoT Device SDKs" and has a "Next" button.

- Name the IoT Thing **AQI-IoT-Thing**

AWS IoT Services Search for services, features, marketplace products, and docs [Option+S] VKMGT N. Virginia Support

**AWS IoT**

CONNECT TO AWS IOT

**Register a thing**

STEP 1/3

A thing is the representation and record of your physical device in the cloud. Any physical device needs a thing to work with AWS IoT. Creating a thing will also create a thing shadow.

Choose an existing thing instead?

Name

Show optional configuration (this can be done later) ▾

Back Next step

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

- Download the connection kit generated and run the start script to test the device.

AWS IoT Services Search for services, features, marketplace products, and docs [Option+S] VKMGT N. Virginia Support

**AWS IoT**

CONNECT TO AWS IOT

**Download a connection kit**

STEP 2/3

The following AWS IoT resources will be created:

A thing in the AWS IoT registry	AQI-IoT-Thing
A policy to send and receive messages	AQI-IoT-Thing-Policy

Preview policy

The connection kit contains:

A certificate and private key	AQI-IoT-Thing.cert.pem, AQI-IoT-Thing.private.key
AWS IoT Device SDK	Python SDK
A script to send and receive messages	start.sh

Before your device can connect and publish messages, you will need to download the connection kit.

Download connection kit for **Linux/OSX**

Back Next step

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

- For MacOS or Linux users

The screenshot shows the AWS IoT Connect and Test Your Device wizard, Step 3/3. The left sidebar includes sections like Monitor, Activity, Onboard (Get started, Fleet provisioning templates), Manage (Fleet Hub, Greengrass, Wireless connectivity, Secure, Defend, Act, Test), Software, Settings, Learn, Documentation, and a New console experience feedback link. The main content area displays three steps: Step 1: Unzip the connection kit on the device (unzip connect\_device\_package.zip), Step 2: Add execution permissions (chmod +x start.sh), and Step 3: Run the start script. A message box says "Waiting for messages from your device". At the bottom are Back and Done buttons.

- For Windows users

The screenshot shows the same AWS IoT Connect and Test Your Device wizard as the previous one, but for Windows users. The left sidebar includes additional sections under Manage: Things, Types, Thing groups, Billing groups, Jobs, Tunnels. The main content area shows the same three steps: Step 1: Unzip the connection kit on the device (unzip connect\_device\_package.zip), Step 2: Add execution permissions (Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process), and Step 3: Run the start script. A message box says "Waiting for messages from your device". At the bottom are Back and Done buttons.

Screenshot of the AWS IoT Connected successfully page.

The page displays a summary of successful tasks:

- Registered a thing to represent a device in AWS IoT
- Set up security for the device using a certificate and policy
- Used a device SDK to connect a device to AWS IoT
- Received messages from the device

A "Done" button is present at the bottom right.

Screenshot of the AWS IoT Things page.

The page lists a single thing named "AQI-IoT-Thing".

Name	Type	...
AQI-IoT-Thing	NO TYPE	...

AWS Services ▾

Search for services, features, marketplace products, and docs [Option+S]

VKMG N. Virginia Support ▾

**AWS IoT** X ⓘ

Introducing the new AWS IoT console experience  
We're updating the console experience for you. Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

AWS IoT > Things > AQI-IoT-Thing

THING AQI-IoT-Thing NO TYPE Actions ▾

**Details** Thing ARN Edit

Security A thing Amazon Resource Name uniquely identifies this thing.

Thing groups arn:aws:iot:us-east-1:797770618644:thing/AQI-IoT-Thing

Billing Groups

Shadows

Interact

Activity

Type Q No type ...

Jobs

Violations

Defender metrics

Feedback English (US) https://console.aws.amazon.com/iot/home?region=us-east-1 © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

AWS Services ▾

Search for services, features, marketplace products, and docs [Option+S]

VKMG N. Virginia Support ▾

**AWS IoT** X ⓘ

Introducing the new AWS IoT console experience  
We're updating the console experience for you. Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

AWS IoT > Things > AQI-IoT-Thing

THING AQI-IoT-Thing NO TYPE Actions ▾

Details This thing already appears to be connected. Connect a device

Security

Thing groups

Billing Groups

Shadows

Interact

Activity

Jobs

Violations

Defender metrics

HTTPS

Update your Thing Shadow using this Rest API Endpoint. Learn more ahketedwn49fr7-ats.iot.us-east-1.amazonaws.com

MQTT

Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow). Learn more

Feedback English (US) https://console.aws.amazon.com/iot/home?region=us-east-1 © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

## 7. Create IoT Rule (10 points)

- Create an IoT rule to send data to DynamoDB. In the AWS IoT Core console, select the **Act** dropdown button on the left panel, and then click **Rules**.

The screenshot shows the AWS IoT Core Rules page. On the left sidebar, under the 'Act' section, the 'Rules' option is selected. The main content area displays a message: 'You don't have any rules yet'. Below this message, a brief description states: 'Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.' At the bottom of the content area are two buttons: 'Learn more' and 'Create a rule'.

- In rule query statement enter: **SELECT \* FROM '#'** as shown below.

The screenshot shows the 'Create a rule' wizard. In the 'Name' field, the value 'SendToDynamoDBTable' is entered. In the 'Rule query statement' section, the text 'SELECT \* FROM "#';' is typed into the input field. The rest of the wizard interface includes fields for 'Description' and a dropdown for 'Using SQL version' set to '2016-03-23'.

- Insert a new action to insert data into the **AirQualityData** table in DynamoDB. Configure the action as shown below

Rule query statement  
Using SQL version 2016-03-23  
Rule query statement  
SELECT <attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

Set one or more actions  
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\* required)

Add action

Error action  
Optional set an action that will be executed when something goes wrong with processing your rule.

Add action

Tags

Select an action

Select an action.

<input type="radio"/>  Insert a message into a DynamoDB table DYNAMODB
<input type="radio"/>  Split message into multiple columns of a DynamoDB table (DynamoDBv2) DYNAMODBV2
<input type="radio"/>  Send a message to a Lambda function LAMBDA
<input type="radio"/>  Send a message as an SNS push notification SNS
<input type="radio"/>  Send a message to an SQS queue SQS
<input type="radio"/>  Send a message to an Apache Kafka cluster APACHE KAFKA
<input type="radio"/>  Send a message to an Amazon Kinesis Stream AMAZON KINESIS
<input type="radio"/>  Republish a message to an AWS IoT topic AWS IoT REPUBLISH
<input type="radio"/>  Store a message in an Amazon S3 bucket AMAZON S3

- Create a new role by clicking **Create Role** to allow access to insert the data into DynamoDB. Name it **aws\_iot\_dynamodb\_airquality**, as shown below

The table must contain Partition and Sort keys.

\*Table name: AirQualityData

\*Partition key: stationID

\*Partition key type: STRING

\*Partition key value: \${stationID}

Sort key: timestamp

Sort key type: NUMBER

Sort key value: \${timestamp}

Write message data to this column: data

Operation: Info

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

**Create Role** (highlighted with a red circle)

Select

Cancel

Add action

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name: aws\_iot\_dynamodb\_airquality

\*Partition key: stationID

\*Partition key type: STRING

\*Partition key value: \${stationID}

Sort key: timestamp

Sort key type: NUMBER

Sort key value: \${timestamp}

Write message data to this column: data

Operation: Info

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

**Create Role** (highlighted with a red circle)

Select

Cancel

Add action

## 8. Run the synthetic data generator to send data to AWS IoT Core (5 points)

- A synthetic data generator is provided with this lab (**myPub.py**). This data generator generates data on raw pollutant concentrations for two stations.
- Update the connection strings in this file to match the AWS IoT thing you created previously.
- Run the data generator to publish synthetic data on raw pollutant measurements to AWS IoT.

```
➜ ~ ~/Downloads/Lab-3/lab-3-files python3 myPub.py
Published topic sdk/test/Python: {"stationID": "ST105", "latitude": 33.933337, "longitude": -84.35729, "timestamp": 1614540974, "pm2_5": 45.8, "pm10": 501.8, "so2": 0.898, "co": 2.56}
Published topic sdk/test/Python: {"stationID": "ST102", "latitude": 33.71795, "longitude": -84.45454, "timestamp": 1614540975, "pm2_5": 391.3, "pm10": 65.5, "so2": 0.675, "co": 3.7}
Published topic sdk/test/Python: {"stationID": "ST102", "latitude": 33.71795, "longitude": -84.45454, "timestamp": 1614540977, "pm2_5": 96.8, "pm10": 121.5, "so2": 0.11, "co": 3.94}
Published topic sdk/test/Python: {"stationID": "ST105", "latitude": 33.933337, "longitude": -84.35729, "timestamp": 1614540978, "pm2_5": 78.0, "pm10": 433.6, "so2": 0.366, "co": 0.09}
Published topic sdk/test/Python: {"stationID": "ST105", "latitude": 33.933337, "longitude": -84.35729, "timestamp": 1614540979, "pm2_5": 355.0, "pm10": 526.4, "so2": 0.185, "co": 1.06}
Published topic sdk/test/Python: {"stationID": "ST105", "latitude": 33.933337, "longitude": -84.35729, "timestamp": 1614540980, "pm2_5": 1.2, "pm10": 265.3, "so2": 0.769, "co": 4.45}
Published topic sdk/test/Python: {"stationID": "ST105", "latitude": 33.933337, "longitude": -84.35729, "timestamp": 1614540981, "pm2_5": 71.4, "pm10": 69.1, "so2": 0.186, "co": 1.87}
Published topic sdk/test/Python: {"stationID": "ST105", "latitude": 33.933337, "longitude": -84.35729, "timestamp": 1614540982, "pm2_5": 183.4, "pm10": 276.0, "so2": 0.728, "co": 1.98}
```

## 9. Verify Data in DynamoDB (5 points)

- After running the synthetic data generator for a minute, navigate back to the DynamoDB created previously, see if the raw air quality data sent by the publisher python program and the computed AQI data are saved in the corresponding databases.

## 10. Deploy the air quality monitoring Django application on EC2 (5 points)

- Deploy the air quality monitoring Django application provided with this lab on EC2.
- Update your EC2 instance by running this command once you are inside the instance:

**sudo apt update**

- Ensure you have the package installer pip3 and Python3 installed on the instance. You can use this command to instance these dependencies:

**sudo apt install python3-pip**

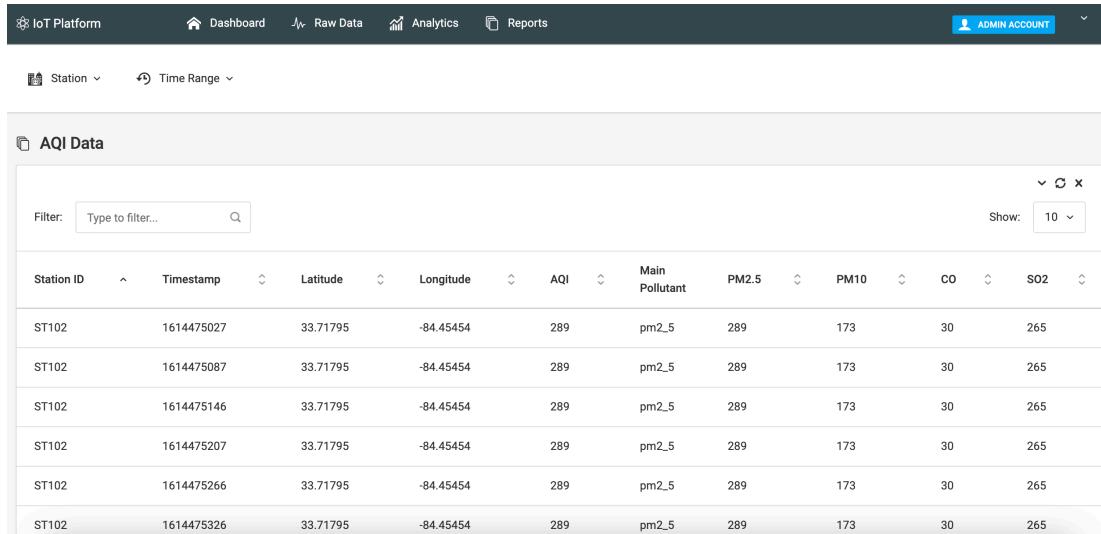
- Inside the **aqi-django-app** directory, run the following commands to install all the dependecies.

**pip3 install -r requirements.txt**

- After installing all required packages, run the Django application and view it within the browser.

### **Python3 manage.py runserver**

- You should be able to view the IoT platform as:



The screenshot shows a web-based IoT platform interface. At the top, there's a navigation bar with links for 'IoT Platform', 'Dashboard', 'Raw Data', 'Analytics', 'Reports', and 'ADMIN ACCOUNT'. Below the navigation bar, there are dropdown menus for 'Station' and 'Time Range'. The main content area is titled 'AQI Data' and displays a table of data. The table has columns for Station ID, Timestamp, Latitude, Longitude, AQI, Main Pollutant, PM2.5, PM10, CO, and SO2. There are 10 rows of data, all corresponding to Station ID ST102 at coordinates 33.71795, -84.45454, with an AQI of 289 and PM2.5 as the main pollutant.

Station ID	Timestamp	Latitude	Longitude	AQI	Main Pollutant	PM2.5	PM10	CO	SO2
ST102	1614475027	33.71795	-84.45454	289	pm2_5	289	173	30	265
ST102	1614475087	33.71795	-84.45454	289	pm2_5	289	173	30	265
ST102	1614475146	33.71795	-84.45454	289	pm2_5	289	173	30	265
ST102	1614475207	33.71795	-84.45454	289	pm2_5	289	173	30	265
ST102	1614475266	33.71795	-84.45454	289	pm2_5	289	173	30	265
ST102	1614475326	33.71795	-84.45454	289	pm2_5	289	173	30	265

## 11.Exercise (40)

- Create a new method in the Django application that allows a user to download the raw data in a CSV format.
- Using the programming language of preference, create a script to plot the data in a graphical line chart for each of the measurements and stations using the CSV with the raw data.
- Modify the script above to get the data from DynamoDB and display the graphical line chart changing in real-time. **\*\* You can run it locally on your computer\*\***
- Create a new Lambda Function with CloudWatch event rule that allows you to send notifications via email when the measurements are outside a particular threshold (See the table 1) .

### **Deliverables:**

1. A video showing all the full functionalities of your Air Quality Monitoring Application, including the ones in the final exercise. Please show your AWS username clearly to indicate that you are using your account. You may briefly explain the functions that you created for the last activity, **but please make sure to limit the total length to 10 minutes.**
2. The complete code with the modifications needed to complete each exercise, including the new lambda function.