Report 1 [IoT]

Student ID: 240581505

Student Name: Shobika Rajeskanna

Introduction:

A pipeline to process Internet of Things (IoT) sensor data in an edge-cloud environment has been created. The operation to be performed are pulling and running EMQX, data injection, data preprocessing, pulling and running RABBITMQ, machine learning model and visualization. Firstly, raw data has been fetched from Urban Observatory and from many metrics, need to store and analysis PM2.5 data with timestamp and values only. Then the filtered data has been sent to EMQX service of Azure lab (edge). Further, collect all PM2.5 data published from EMQX service, next filter out outliers which value is greater than 50. Then, the results (averaged data) have been taken into RabbitMQ service on Azure lab(cloud). After performing data preprocessing, collected daily average PM2.5 data. Furthermore, converted timestamp to date time format and using matplotlib displayed line chart of daily average of PM2.5 data. Finally, using feed averaged PM2.5 data to machine learning model to predict the trend of PM2.5 for next 15 days and visualized the predicted result machine learning classifier model.

Task-1

Design a data injector component by leveraging Newcastle Urban Observatory IoT data streams.

Step 1: Pull Docker image emqx/emqx from Dc:

docker pull emqx/emqx

Step 2: Run Docker image emqx/emqx from Docker Hub:

docker run -d --name emqx\_broker -p 1883:1883 emqx/emqx

EMQX helps in filtering relevant data (like pm2.5 data) by acting as a local broker for distributing data to data preprocessing operator.

Step 3: Development of data injector code using python in Azure Lab (Edge):

Functions in the code: collect data from Urban Observatory platform by sending HTTP request to the given API. Then fetch the raw data and printed them in the console.

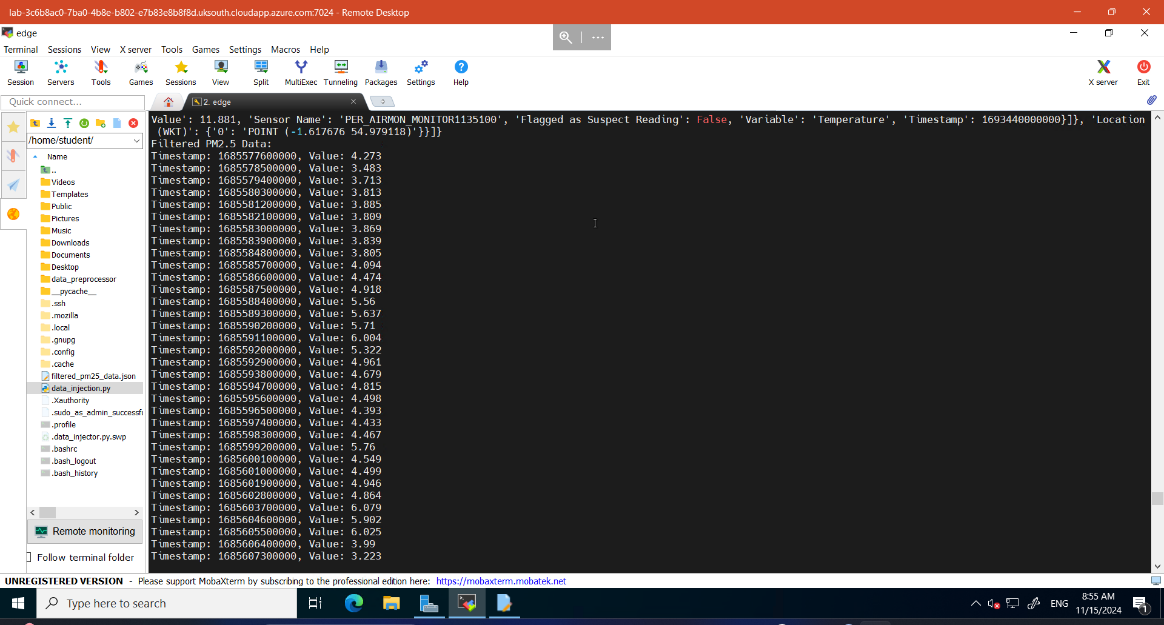
Output:

A screenshot of a computer

Description automatically generated

Step 4: Filtering PM2.5 data:

In Urban Observatory API, there are many air quality sensor data available. We need to print PM2.5 data and in their meta-data, we need to print Timestamp and Values alone.



Step 5: Sending all the data to EMQX service of Azure lab (Edge):

Firstly, need to setup MQTT broker information like Host’s IP, Port number and channel topic (in my case the topic name is sensor/pm2.5) where we send our data. Then, connecting to MQTT broker:

client.connect(BROKER\_HOST, BROKER\_PORT)

Finally sending data to EMQX:

Code snippet for performing the operation:

for data\_point in pm25\_data:

message = json.dumps(data\_point)

client.publish(MQTT\_TOPIC, message)

print(f"Sent data to EMQX: {message}")

time.sleep(0)

Output:

A screenshot of a computer

Description automatically generated

In below screenshot I have added the python program that performs all the functions I mentioned earlier.

data\_injector.py

A screenshot of a computer

Description automatically generatedA white background with black dots

Description automatically generated

Task 2

Data preprocessing operator design

Step1:

In cloud pull and run the RabbitMQ, receive data from the edge (via EMQX) and queue it for further processing and visualizing.

Command to pull RabbitMQ:

docker pull rabbitmq:management

Command to run RabbitMQ:

docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management

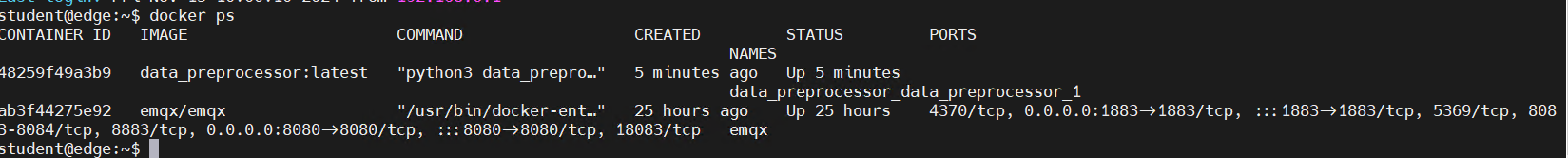
Step2:

Now subscribe all the PM2.5 data that is been sent from task 1 from EMQX service. Containerize this operator using Docker container, so the logs can be seen in the docker logs.

For containerization we need docker-compose, which will help us to start a container in correct order.

Command for docker-compose: docker-compose up

Containers in edge:



Docker log in edge:

A screen shot of a computer

Description automatically generated

The above output is fetching data from EMQX service and as I containerized my data preprocessor it is been logged and logs can be seen automatically.

Step 3: Outliers

Perform a function in the code to find the values greater than 50.

Step 4:

Average value of PM2.5 data in daily basis has been created and the result has been printed in docker logs console.

Output:

A screenshot of a computer

Description automatically generated

Step 5:

Finally, the averaged PM2.5 data has been published into RabbitMQ service on Azure lab (Cloud). Connection has been made, for the connection we need RabbitMQ broker information like Host’s IP, and rabbitmq \_queue (in my code processed\_pm25\_data is queue).

Connection code snippet:

def publish\_to\_rabbitmq(data):

connection = pika.BlockingConnection(pika.ConnectionParameters(host=RABBITMQ\_HOST))

channel = connection.channel()

channel.queue\_declare(queue=RABBITMQ\_QUEUE)

for item in data:

message = json.dumps(item)

channel.basic\_publish(exchange='', routing\_key=RABBITMQ\_QUEUE, body=message)

print(f"[Published] to RabbitMQ - Timestamp: {item['timestamp']}, PM2.5 Average: {item['value']}")

connection.close()

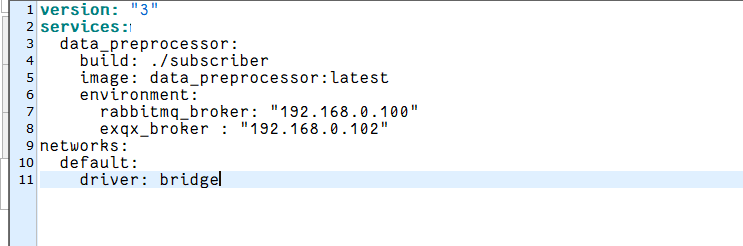
Output:

A screenshot of a computer

Description automatically generated

So, in this task we have created Dockerfile to migrate data preprocessor and docker-compose to run the container.

docker-compose:



Dockerfile:

A screenshot of a computer

Description automatically generated

Code:

Now, the screenshot of entire code that performs task 2 has been added below.

data\_preprocessor.py

A screenshot of a computer

Description automatically generatedA white background with many small colored lines

Description automatically generated with medium confidenceA white background with blue and red text

Description automatically generated

Task 3

Time series data prediction and visualization

Step 1:

Firstly, I added pre-defined machine learning engine code in cloud according to my prediction code.

ml\_engine.py

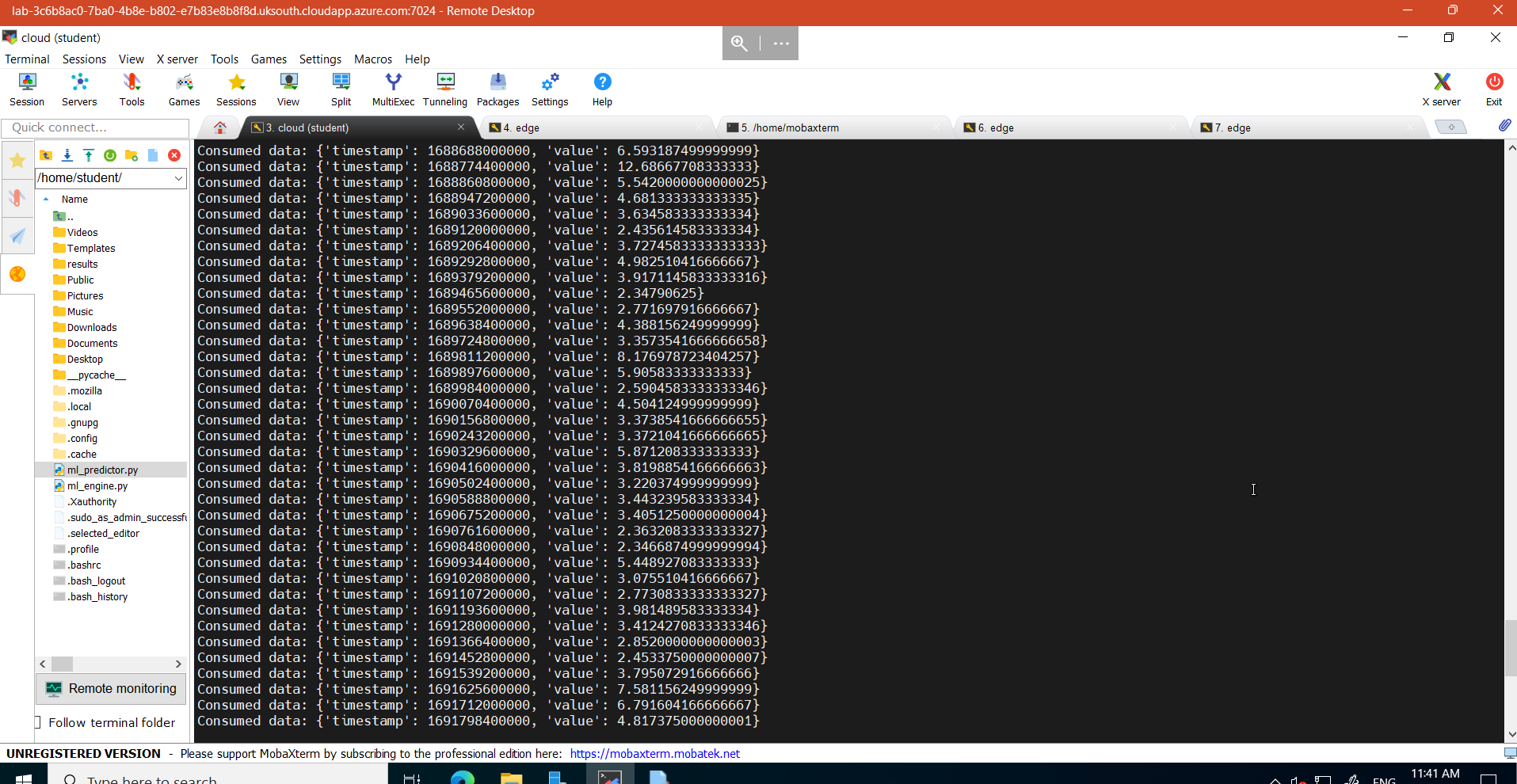
A screenshot of a computer

Description automatically generated

Step 2:

Consume all averaged daily PM2.5 data that has been published by RabbitMQ service in task 2.

Output:



Step 3:

Convert timestamp to date time format.

Output:

A black background with white letters

Description automatically generated

Step 4:

Using matplotlib, visualize the averaged PM2.5 daily data.

Line chart:

A graph showing a number of numbers

Description automatically generated with medium confidence

Step 5:

Finally, I feed the averaged data to machine learning model to predict the trend of PM2.5 data for next 15 days. Then, the prediction has been visualized and stored in the directory.

Figure:

A graph showing a line

Description automatically generated

Code for task 3:

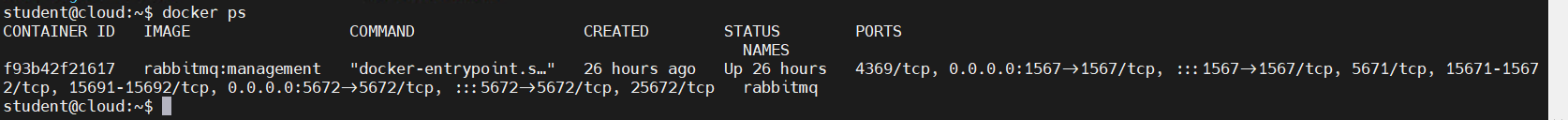
ml\_predictor.py

A computer screen shot of a computer

Description automatically generatedA white background with black text

Description automatically generated

Cloud Container:



Analytics and Conclusion:

In this IOT system, we work with three tiers, they are IoT tier, edge tier and cloud tier. From IoT tier we are accessing the real time data. Then in edge tier we are fetching the real time data and extracting the set of data be processed and, we are publishing the data to preprocessor and then the preprocessor will subscribe the data and removed outlier and finds the average and push the data to cloud tier. Now the cloud tier displays the averaged data into line chart and then using machine learning it predicts for next 15 days, then the predicted image has been stored.