

Internet of Things (IOT) Project

Introduction

This project demonstrates utilizing the data collected from IOT sensors and use it further to complete following tasks. These tasks include collection of data, preprocessing that data, visualizing the data and passing this data to machine learning model for data prediction of next 15 days. The data flows between three layers, namely IOT layer, edge layer and cloud layer. IOT layer comprises of all the sensors and its data is stored in Newcastle Urban Observatory. The edge layer comprises of two tasks wherein task 1 we need to collect data from urban observatory and publish that data to task 2 via EMQX docker image and in task 2, we collect the data published by task 1 and do preprocessing on that data and send it to cloud layer by using RabbitMQ docker image. Now once the data reaches to cloud layer i.e task 3, we use this data for visualization and prediction using the provided machine learning model.

Below figure depicts the overview diagram.

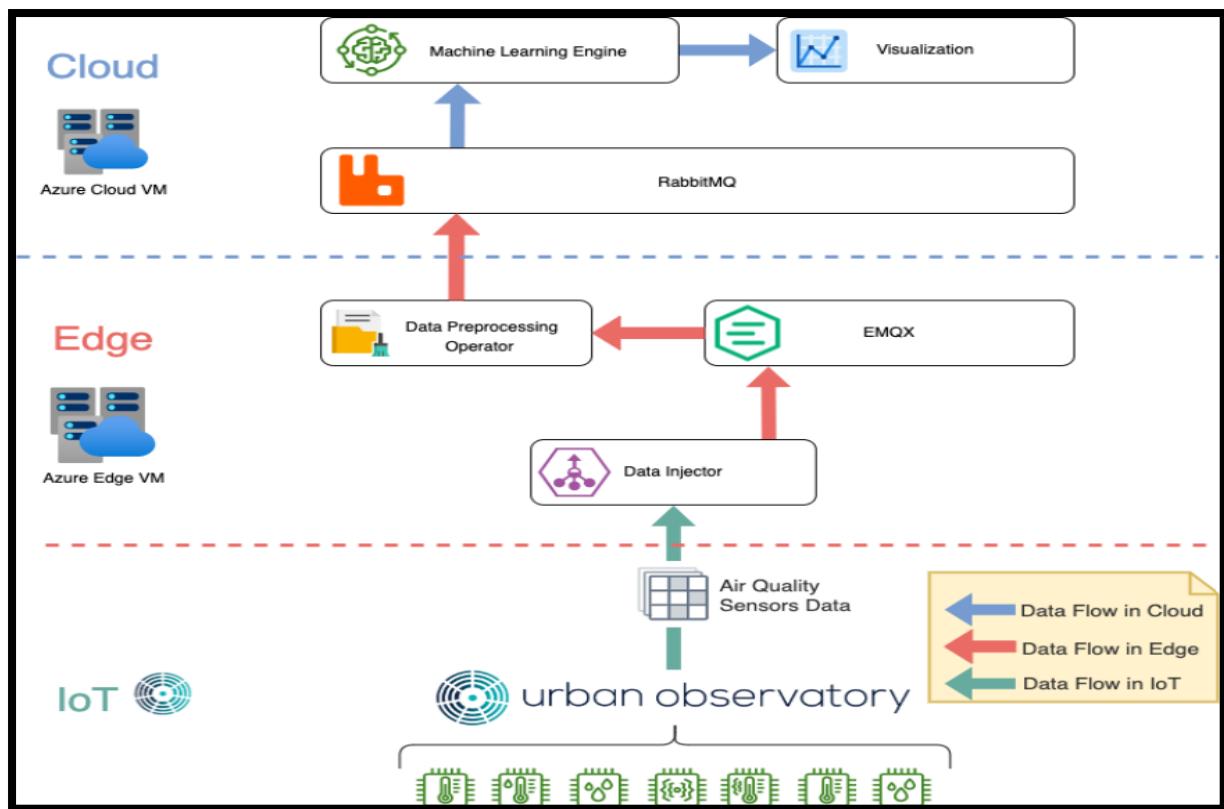


Figure 1: Overview Diagram

Before proceeding with the tasks, we need to turn on both the ubuntu virtual machines, one for edge and another one for cloud. Then connect to those VMs and install *net-tools* to the system by running the following command,

`sudo apt install net-tools` – This package contains important tools for controlling network subsystem.

Then we can run "`ifconfig`" command to check IP addresses of both VMs. Now we need to manually install python3, pip, and docker-compose tool in both ubuntu VMs.

Task 1: Designing a data injector component

In task 1, first of all we need to install “requests” and “paho-mqtt” dependencies in edge virtual machine using the below mentioned commands,

[pip install requests](#)

[pip install paho-mqtt](#)

requests is a simple HTTP library that allows us to perform HTTP based communications.

paho-mqtt enables applications to connect to an MQTT broker and publish message and subscribe the message from topic.

After installing required dependencies, we need to pull and run EMQX docker image on edge virtual machine by executing below command on terminal, here we map port number 1883 of host with port 1883 of container.

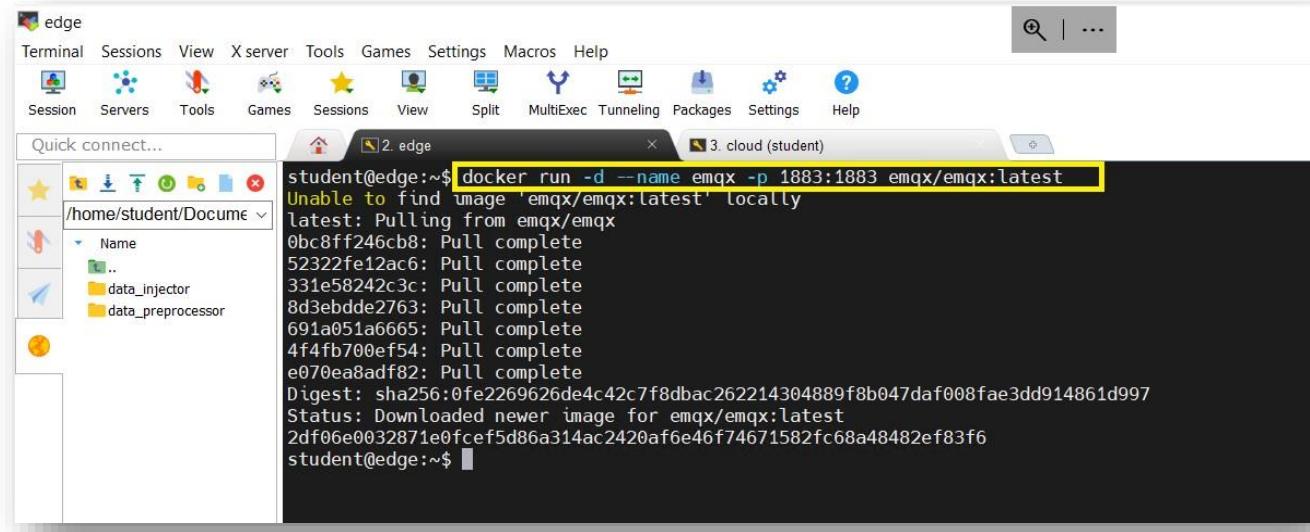


Figure 2: Pulling and running EMQX docker image on edge virtual machine

EMQX docker image is an open source MQTT broker and is used to publish and subscribe messages, to and from topic.

Now we need to write a python code for data injector component.

Code description:

Firstly, we are importing all the required libraries such as json, time, requests and paho mqtt client. Then we create mqtt client object and connect to the mqtt service. We provide IP address of edge virtual machine and port number 1883 to which EMQX image is mapped while connecting to mqtt service. Proceeding further, we have to call Newcastle Urban Observatory API and collect the raw air quality sensors data and convert it from json response to dictionary format. This raw data consists of many metrics such as CO₂, NO₂, NO, PM2.5, PM10 and so on, which we print on the console. However, we only need the data related to PM2.5, so we fetch only PM2.5 data and store it in a list. This PM2.5

data has parameters like Sensor Name, Units, Variable, Timestamp, Value and Flagged as Suspect Reading. Out of these parameters we need only timestamp and value field, as a result we are filtering out other parameters and storing and printing only timestamp and value data of PM2.5. Now we will publish this data on topic which will be further subscribed by task 2.

Code snippet for data injector component:

```
datainjector.py ✘
1 import json
2 import time
3 import requests
4
5 from paho.mqtt import client as mqtt_client
6
7 mqtt_ip = "192.168.0.102"
8 mqtt_port = 1883
9 topic = "CSC8112"
10
11 # Create a mqtt client object
12 client = mqtt_client.Client()
13
14 # Callback function for MQTT connection
15 def on_connect(client, userdata, flags, rc):
16     if rc == 0:
17         print("Connected to MQTT OK!")
18     else:
19         print("Failed to connect, return code %d\n", rc)
20
21 # Connect to MQTT service
22 client.on_connect = on_connect
23 client.connect(mqtt_ip, mqtt_port)
24
25 # Publish message to MQTT
26 # Storing urban observatory url in a variable
27 link = "http://uoweb3.ncl.ac.uk/api/v1.1/sensors/PER_AIRMON_MONITOR1135100/data/json/?starttime=20230601&endtime=20230831"
28
29 # Getting data by calling API
30 getdata = requests.get(link)
31
32 # Converting the response data to dictionary
33 convert = getdata.json()
34
35 # Printing raw data
36 print(convert)
37
38 # Fetching only PM2.5 data and filtering rest and storing in a variable result of type list
39 result = convert["sensors"][0]["data"]["PM2.5"]
40
41 # Keep only Timestamp and Value data, remove remaining data
42 for i in result:
43     del i['Sensor Name']
44     del i['Units']
45     del i['Variable']
46     del i['Flagged as Suspect Reading']
47
48 print(result)
49
50 client.publish(topic, json.dumps(result)) # json.dumps(result) converts it to string and publishes
51 time.sleep(2)
```

Figure 3: Task 1 python code

Output of Task 1 – datainjector component

```

student@edge:~$ cd Documents
student@edge:~/Documents/coursework$ cd coursework
student@edge:~/Documents/coursework$ cd data_injector
student@edge:~/Documents/coursework/data_injector$ python3 datainjector.py
[{"sensors": [{"Sensor Name": {"0": "PER_AIRMON_MONITOR1135100"}, "Raw ID": {"0": "79525"}, "Broker Name": {"0": "aq_mesh_api"}, "Sensor Centroid Latitude": {"0": 54.979118}, "Sensor Height Above Ground": {"0": 2.0}, "Ground Height Above Sea Level": {"0": 57.669981689}, "Location (WKT)": {"0": "POI NT (-1.617676 54.979118)"}, "Sensor Centroid Longitude": {"0": -1.617676}, "data": {"PM 4": [{"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.63, "Flagged as Suspect Reading": false, "Timestamp": 1685577600000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 3.91, "Flagged as Suspect Reading": false, "Timestamp": 1685578500000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.171, "Flagged as Suspect Reading": false, "Timestamp": 1685578490000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.297, "Flagged as Suspect Reading": false, "Timestamp": 1685580300000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.48, "Flagged as Suspect Reading": false, "Timestamp": 1685581200000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.248, "Flagged as Suspect Reading": false, "Timestamp": 1685582100000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.327, "Flagged as Suspect Reading": false, "Timestamp": 1685583000000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.159, "Flagged as Suspect Reading": false, "Timestamp": 1685583900000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.278, "Flagged as Suspect Reading": false, "Timestamp": 1685584800000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.685, "Flagged as Suspect Reading": false, "Timestamp": 1685587000000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.061, "Flagged as Suspect Reading": false, "Timestamp": 1685586000000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.407, "Flagged as Suspect Reading": false, "Timestamp": 1685587500000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 6.059, "Flagged as Suspect Reading": false, "Timestamp": 1685588490000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 6.214, "Flagged as Suspect Reading": false, "Timestamp": 1685589300000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 6.315, "Flagged as Suspect Reading": false, "Timestamp": 1685589200000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 6.519, "Flagged as Suspect Reading": false, "Timestamp": 1685591100000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.861, "Flagged as Suspect Reading": false, "Timestamp": 1685592000000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.516, "Flagged as Suspect Reading": false, "Timestamp": 1685592900000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.281, "Flagged as Suspect Reading": false, "Timestamp": 1685593800000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.46, "Flagged as Suspect Reading": false, "Timestamp": 1685594700000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.968, "Flagged as Suspect Reading": false, "Timestamp": 1685595600000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.063, "Flagged as Suspect Reading": false, "Timestamp": 1685596500000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 4.932, "Flagged as Suspect Reading": false, "Timestamp": 1685597400000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.114, "Flagged as Suspect Reading": false, "Timestamp": 1685598300000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.577, "Flagged as Suspect Reading": false, "Timestamp": 1685599200000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.324, "Flagged as Suspect Reading": false, "Timestamp": 1685600100000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.19, "Flagged as Suspect Reading": false, "Timestamp": 1685601000000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.575, "Flagged as Suspect Reading": false, "Timestamp": 1685601900000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 5.667, "Flagged as Suspect Reading": false, "Timestamp": 1685602800000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 7.253, "Flagged as Suspect Reading": false, "Timestamp": 1685603700000, "Variable": "PM 4", "Units": "ugm -3"}, {"Sensor Name": "PER_AIRMON_MONITOR1135100", "Value": 6.901, "Flagged as Suspect Reading": false, "Timestamp": 1685609100000, "Variable": "PM 4", "Units": "ugm -3"}]

```

Figure 4: Printing raw air quality sensors data collected from urban observatory

```

[{"Value": 4.273, "Timestamp": 1685577600000, {"Value": 3.483, "Timestamp": 1685578500000, {"Value": 3.713, "Timestamp": 1685579400000, {"Value": 3.813, "Timestamp": 1685579300000, {"Value": 3.885, "Timestamp": 1685581200000, {"Value": 3.809, "Timestamp": 1685582100000, {"Value": 3.869, "Timestamp": 1685583000000, {"Value": 3.839, "Timestamp": 1685583900000, {"Value": 3.805, "Timestamp": 1685584800000, {"Value": 4.094, "Timestamp": 1685585700000, {"Value": 4.474, "Timestamp": 1685586600000, {"Value": 4.918, "Timestamp": 1685587500000, {"Value": 5.56, "Timestamp": 1685588400000, {"Value": 5.637, "Timestamp": 1685589300000, {"Value": 5.71, "Timestamp": 1685590200000, {"Value": 6.004, "Timestamp": 1685591100000, {"Value": 5.322, "Timestamp": 1685592000000, {"Value": 4.961, "Timestamp": 1685593900000, {"Value": 4.679, "Timestamp": 1685594800000, {"Value": 4.815, "Timestamp": 1685594700000, {"Value": 4.498, "Timestamp": 1685595600000, {"Value": 4.393, "Timestamp": 1685596500000, {"Value": 4.453, "Timestamp": 1685597400000, {"Value": 4.467, "Timestamp": 1685598300000, {"Value": 5.76, "Timestamp": 1685599200000, {"Value": 4.549, "Timestamp": 1685600100000, {"Value": 4.499, "Timestamp": 1685601000000, {"Value": 4.946, "Timestamp": 1685601900000, {"Value": 4.864, "Timestamp": 1685602800000, {"Value": 6.079, "Timestamp": 1685603700000, {"Value": 5.902, "Timestamp": 1685604600000, {"Value": 6.025, "Timestamp": 1685605500000, {"Value": 3.99, "Timestamp": 1685606400000, {"Value": 3.223, "Timestamp": 1685607300000, {"Value": 2.434, "Timestamp": 1685608200000, {"Value": 2.07, "Timestamp": 1685609100000, {"Value": 1.968, "Timestamp": 1685610000000, {"Value": 2.497, "Timestamp": 1685610900000, {"Value": 3.192, "Timestamp": 1685611800000, {"Value": 3.166, "Timestamp": 1685612700000, {"Value": 2.994, "Timestamp": 1685613600000, {"Value": 3.023, "Timestamp": 1685614500000, {"Value": 3.652, "Timestamp": 1685615400000, {"Value": 2.243, "Timestamp": 1685616300000, {"Value": 2.652, "Timestamp": 1685617200000, {"Value": 3.137, "Timestamp": 1685618100000, {"Value": 3.075, "Timestamp": 1685619000000, {"Value": 3.376, "Timestamp": 1685619900000, {"Value": 3.494, "Timestamp": 1685620800000, {"Value": 3.639, "Timestamp": 1685621700000, {"Value": 3.826, "Timestamp": 1685622600000, {"Value": 3.888, "Timestamp": 1685623500000, {"Value": 4.861, "Timestamp": 1685624400000, {"Value": 5.634, "Timestamp": 1685625300000, {"Value": 5.097, "Timestamp": 1685626200000, {"Value": 5.414, "Timestamp": 1685627100000, {"Value": 6.029, "Timestamp": 1685628000000, {"Value": 5.582, "Timestamp": 1685628900000, {"Value": 5.885, "Timestamp": 1685629800000, {"Value": 4.228, "Timestamp": 1685630700000, {"Value": 4.729, "Timestamp": 1685631600000, {"Value": 4.843, "Timestamp": 1685632500000, {"Value": 5.195, "Timestamp": 1685633400000, {"Value": 4.848, "Timestamp": 1685634300000, {"Value": 5.279, "Timestamp": 1685635200000, {"Value": 4.995, "Timestamp": 1685636100000, {"Value": 3.846, "Timestamp": 1685637000000, {"Value": 3.344, "Timestamp": 1685637900000, {"Value": 3.533, "Timestamp": 1685638800000, {"Value": 3.769, "Timestamp": 1685639700000, {"Value": 3.571, "Timestamp": 1685640600000, {"Value": 4.762, "Timestamp": 1685641500000, {"Value": 4.365, "Timestamp": 1685642400000, {"Value": 3.379, "Timestamp": 1685643300000, {"Value": 3.65, "Timestamp": 1685644200000, {"Value": 4.062, "Timestamp": 1685645100000, {"Value": 4.384, "Timestamp": 1685646000000, {"Value": 4.546, "Timestamp": 1685646900000, {"Value": 4.785, "Timestamp": 1685647800000, {"Value": 4.646, "Timestamp": 1685648700000, {"Value": 3.858, "Timestamp": 1685649600000, {"Value": 4.017, "Timestamp": 1685650500000, {"Value": 4.171, "Timestamp": 1685651400000, {"Value": 4.372, "Timestamp": 1685652300000, {"Value": 4.599, "Timestamp": 1685653200000, {"Value": 5.043, "Timestamp": 1685654100000, {"Value": 5.614, "Timestamp": 1685655000000, {"Value": 5.468, "Timestamp": 1685655900000, {"Value": 5.435, "Timestamp": 1685656800000, {"Value": 5.385, "Timestamp": 1685657700000, {"Value": 4.732, "Timestamp": 1685658600000, {"Value": 5.029, "Timestamp": 1685659500000, {"Value": 5.128, "Timestamp": 1685660400000, {"Value": 5.514, "Timestamp": 1685661300000, {"Value": 5.505, "Timestamp": 1685662200000, {"Value": 5.388, "Timestamp": 1685663100000, {"Value": 5.3, "Timestamp": 1685664000000, {"Value": 5.065, "Timestamp": 1685664900000, {"Value": 4.82, "Timestamp": 1685665800000, {"Value": 4.175, "Timestamp": 1685666700000, {"Value": 4.468, "Timestamp": 1685667600000, {"Value": 4.027, "Timestamp": 1685668500000, {"Value": 4.113, "Timestamp": 1685669400000, {"Value": 4.007, "Timestamp": 1685671200000, {"Value": 4.456, "Timestamp": 1685672100000, {"Value": 4.047, "Timestamp": 1685673000000, {"Value": 4.15, "Timestamp": 1685673900000, {"Value": 4.105, "Timestamp": 1685674800000, {"Value": 4.167, "Timestamp": 1685675700000, {"Value": 4.159, "Timestamp": 1685676600000, {"Value": 4.255, "Timestamp": 1685677500000, {"Value": 4.713, "Timestamp": 1685684000000, {"Value": 4.654, "Timestamp": 1685682000000, {"Value": 4.744, "Timestamp": 1685682900000, {"Value": 5.033, "Timestamp": 1685683800000, {"Value": 5.793, "Timestamp": 1685684700000, {"Value": 7.003, "Timestamp": 1685685600000, {"Value": 5.999, "Timestamp": 1685686500000, {"Value": 6.046, "Timestamp": 1685687400000, {"Value": 6.565, "Timestamp": 1685688300000, {"Value": 6.451, "Timestamp": 1685689200000, {"Value": 7.23, "Timestamp": 1685690100000, {"Value": 7.9, "Timestamp": 1685691000000, {"Value": 6.901, "Timestamp": 1685691900000, {"Value": 7.4, "Timestamp": 1685692800000}]}

```

Figure 5: Printing the metrics timestamp and value of PM2.5 data

Task 2: Data preprocessing operator

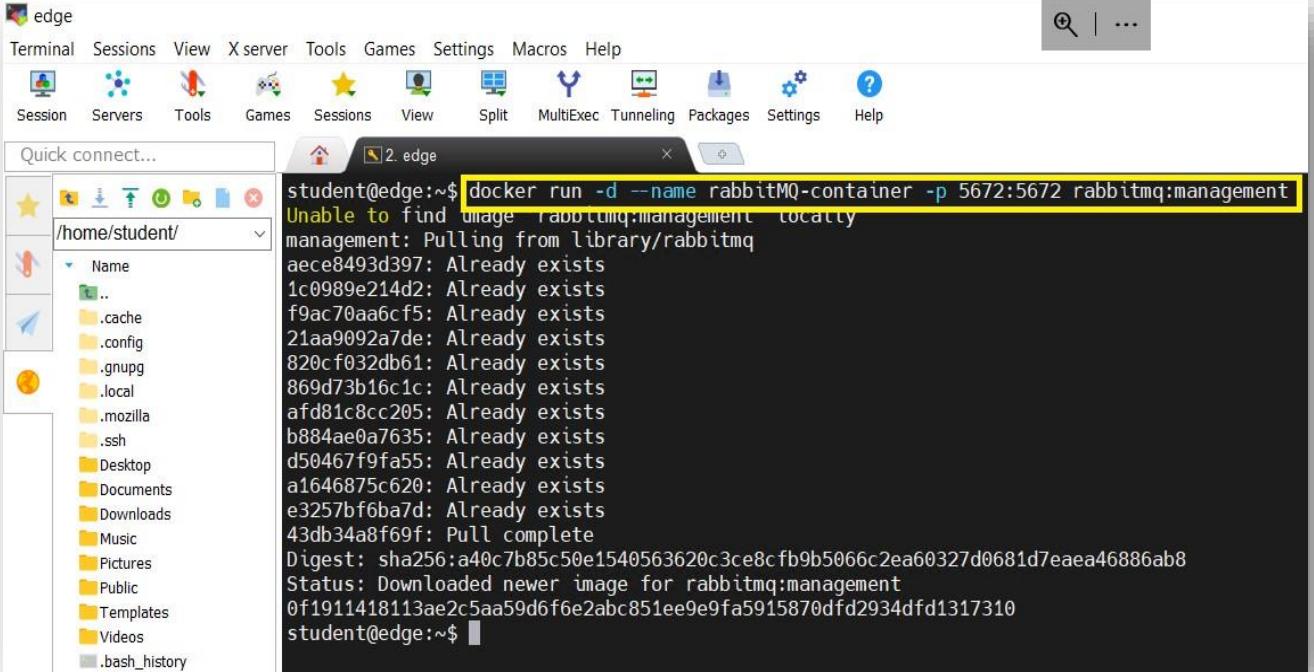
For task 2 we need “paho-mqtt” and “pika” dependencies to be installed in edge virtual machine but as “paho-mqtt” has already been installed in task 1, we will just install pika by executing following command,

[pip install pika](#)

pika is a RabbitMQ client library with implementation of Advance Message Queuing Protocol (AMQP).

We have already downloaded EMQX docker image on edge virtual machine in task 1 which will be used to subscribe data published by task 1. As in task 2, after preprocessing, we have to send that data to cloud layer (task 3) with the help of RabbitMQ docker image, we need to pull and run that image on edge virtual machine.

We execute below command on edge VMs terminal and map the image on port 5672.



The screenshot shows the Edge interface with a terminal window open. The terminal window title is "2. edge". The command entered is:

```
student@edge:~$ docker run -d --name rabbitMQ-container -p 5672:5672 rabbitmq:management
```

The output of the command is displayed in the terminal window:

```
Unable to find image 'rabbitmq:management' locally
management: Pulling from library/rabbitmq
aece8493d397: Already exists
1c0989e214d2: Already exists
f9ac70aa6cf5: Already exists
21aa9092a7de: Already exists
820cf032db61: Already exists
869d73b16c1c: Already exists
afdb81c8cc205: Already exists
b884ae0a7635: Already exists
d50467f9fa55: Already exists
a1646875c620: Already exists
e3257bf6ba7d: Already exists
43db34a8f69f: Pull complete
Digest: sha256:a40c7b85c50e1540563620c3ce8cfb9b5066c2ea60327d0681d7aea46886ab8
Status: Downloaded newer image for rabbitmq:management
0f1911418113ae2c5aa59d6f6e2abc851ee9e9fa5915870dfd2934dfd1317310
student@edge:~$
```

Figure 6: Pulling and running RabbitMQ docker image on edge virtual machine

RabbitMQ image is an open source message broker software which implements AMQP protocol and will be used for data communication/flow between edge layer and cloud layer.

Code Description

Import all the required libraries such as json, pika, datetime and paho-mqtt. Create the mqtt client object and connect to mqtt service in order to get the data from the topic which is published by task 1. In data preprocessing our first task is to filter out the outliers and print them on the screen. Outliers are the data points for which value is greater than 50. After removing all the outliers from the data, next task is to calculate the daily average value. In order to get average value of a particular day, I have made use of dictionaries. One dictionary will be storing the date and total of all the values for that particular date. Another dictionary to store the date and total number of values on that date and finally third dictionary will store the date and average value. Next, we are storing this data in a list with

timestamp and average value which will be sent by RabbitMQ producer to cloud (task 3). As we are sending this data to cloud layer, we have specified IP address of cloud virtual machine and 5672 port number which is mapped with RabbitMQ image while connecting to RabbitMQ service.

Code snippet for data preprocessor operator

```
datapreprocessor.py
1 import json
2 import pika
3 from datetime import datetime
4 from paho.mqtt import client as mqtt_client
5
6 if __name__ == '__main__':
7     mqtt_ip = "192.168.0.102"
8
9     mqtt_port = 1883
10    topic = "CSC8112"
11
12    # Create a mqtt client object
13    client = mqtt_client.Client()
14
15    # Callback function for MQTT connection
16    def on_connect(client, userdata, flags, rc):
17        if rc == 0:
18            print("Connected to MQTT OK!", flush = True)
19        else:
20            print("Failed to connect, return code %d\n", rc, flush = True)
21
22
23    # Connect to MQTT service
24    client.on_connect = on_connect
25    client.connect(mqtt_ip, mqtt_port)
26
27
28    # Callback function will be triggered
29    def on_message(client, userdata, msg):
30        print(f"Printing PM2.5 data collected by emqx subscriber: {json.loads(msg.payload)}", flush = True) #type of json.loads(msg.payload): list
31        emqxcollecteddata = json.loads(msg.payload)
32
33        # Checking outliers and printing them on console
34        outliers = []
35        for d in emqxcollecteddata:
36            if d['Value'] >= 50:
37                outliers.append(d)
38        print("*****", flush = True)
39        print("outliers ", outliers, flush = True)
40        print("*****", flush = True)
41
42        # Filtering out outliers
43        outliersremoved = [] # Data without outliers
44        for d1 in emqxcollecteddata:
45            if d1['Value'] < 50:
46                outliersremoved.append(d1)
47
48
49        # Getting average values logic below
50        a = {}
51        b = {}
52        c = {}
53        preprocessdata = []
54        cnt = 2
55        cnt1 = 1
56        for entry in outliersremoved:
57            timestamp = entry['Timestamp']
58            value = entry['Value']
59            date = datetime.fromtimestamp(timestamp / 1000).strftime('%Y-%m-%d')
60            #print(date)
61            if len(a) == 0:
62                a[date] = value
63            else:
64                if date in a:
65                    a[date] = a[date] + value
66                else:
67                    a[date] = value
68
69
70        for entry1 in outliersremoved:
71            timestamp = entry1['Timestamp']
72            value = entry1['Value']
73            date = datetime.fromtimestamp(timestamp / 1000).strftime('%Y-%m-%d')
74            #print(date)
75            if len(a) == 0:
76                b[date] = 1
77            else:
78                if date in b:
79                    b[date] = b[date] + 1
80                else:
81                    b[date] = 1
82
83        #print(b)
```

```

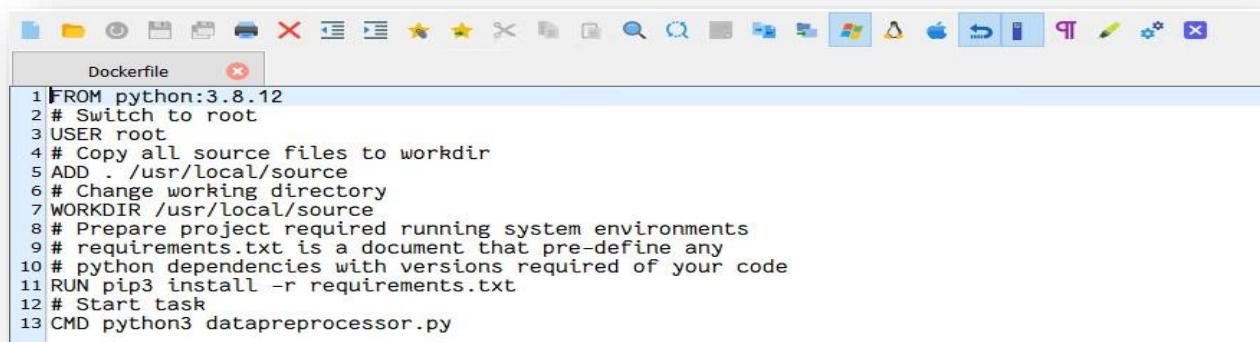
84     #print(b)
85
86     for d in a:
87         c[d] = a[d] / b[d]
88
89     #print(c)
90
91     for i in c:
92         preprocessdata.append({'Timestamp': i, 'Value': c[i]})
93
94     print("Printing daily average value data: ", preprocessdata, flush = True)
95
96     rabbitmq_ip = "192.168.0.100"
97     rabbitmq_port = 5672
98     # Queue name
99     rabbitmq_queue = "CSC8112test"
100    # Connect to RabbitMQ service
101    connection = pika.BlockingConnection(pika.ConnectionParameters(host=rabbitmq_ip, port=rabbitmq_port))
102    channel = connection.channel()
103    # Declare a queue
104    channel.queue_declare(queue=rabbitmq_queue)
105    # Produce message
106    channel.basic_publish(exchange='',
107                           routing_key=rabbitmq_queue,
108                           body=json.dumps(preprocessdata))
109    connection.close()
110
111
112
113
114
115
116
117    # Subscribe MQTT topic
118    client.subscribe(topic)
119    client.on_message = on_message
120
121    # Start a thread to monitor message from publisher
122    client.loop_forever()
123
124

```

Figure 7: Task 2 python code

Another part in task 2 is to build a docker image of data preprocessing operator. For these we need to create three files namely docker file, yaml file and requirements text file.

Docker File



```

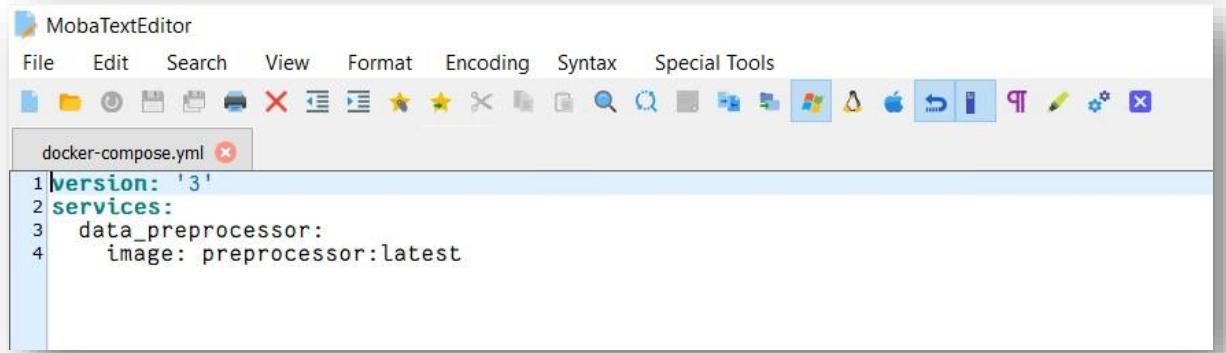
Dockerfile
1 FROM python:3.8.12
2 # Switch to root
3 USER root
4 # Copy all source files to workdir
5 ADD . /usr/local/source
6 # Change working directory
7 WORKDIR /usr/local/source
8 # Prepare project required running system environments
9 # requirements.txt is a document that pre-define any
10 # python dependencies with versions required of your code
11 RUN pip3 install -r requirements.txt
12 # Start task
13 CMD python3 datapreprocessor.py

```

Figure 8: Docker File

Docker file is used to build an image of our data preprocessor (task 2) code. In this file we specify that the language used is python, then we switch to the root and copy all the files from the current directory to source directory and set it as a working directory. Next it installs all the required dependencies for running the code that are mentioned in requirements text file and finally the command to run our data preprocessor (task 2) code.

Docker-compose file



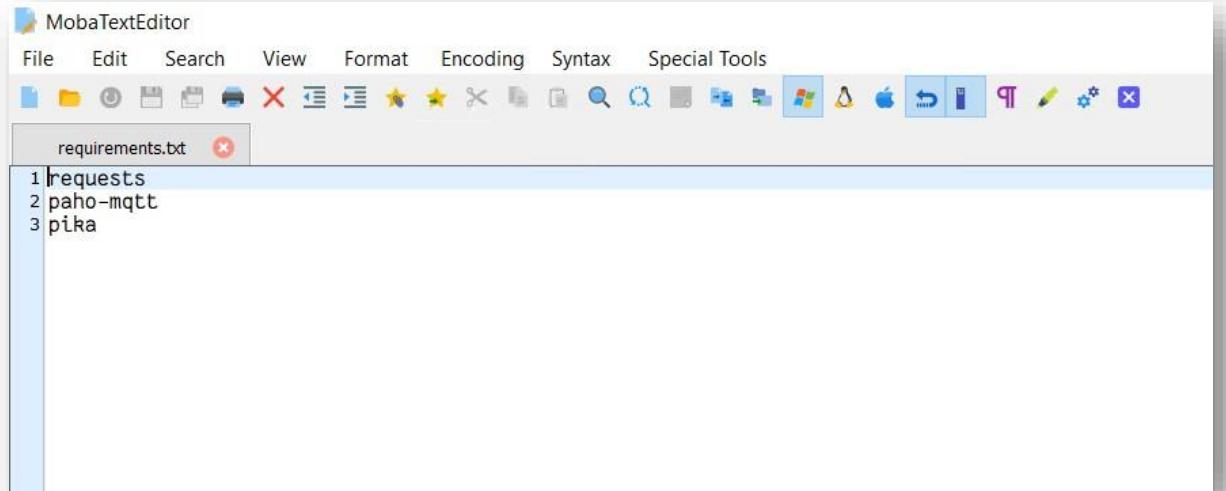
A screenshot of the MobaTextEditor application window. The title bar says "MobaTextEditor". The menu bar includes File, Edit, Search, View, Format, Encoding, Syntax, and Special Tools. Below the menu is a toolbar with various icons. The main editor area shows a file named "docker-compose.yml". The content of the file is:

```
1 version: '3'
2 services:
3   data_preprocessor:
4     image: preprocessor:latest
```

Figure 9: **docker-compose file**

Docker compose file contains the configuration in which we define the name and image of the service that we want to run as a part of our application.

Requirements text file



A screenshot of the MobaTextEditor application window. The title bar says "MobaTextEditor". The menu bar includes File, Edit, Search, View, Format, Encoding, Syntax, and Special Tools. Below the menu is a toolbar with various icons. The main editor area shows a file named "requirements.txt". The content of the file is:

```
1 requests
2 paho-mqtt
3 pika
```

Figure 10: **requirements text file**

Requirements text file contains all the dependencies that we want to install as a part of running our application.

Now we build the image of our data preprocessor (task 2) operator.

```

student@edge:~$ cd Documents
student@edge:~/Documents$ cd coursework
student@edge:~/Documents/coursework$ cd data_preprocessor
student@edge:~/Documents/coursework/data_preprocessor$ docker build -t preprocessor:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 8.704kB
Step 1/6 : FROM python:3.8.12
--> 52bb9574949f
Step 2/6 : USER root
--> Running in 153d988a6916
Removing intermediate container 153d988a6916
--> 05f2ec0290e3
Step 3/6 : ADD . /usr/local/source
--> 52ce52e739d3
Step 4/6 : WORKDIR /usr/local/source
--> Running in 82e96bb0e53c
Removing intermediate container 82e96bb0e53c
--> 1e9912fbf408
Step 5/6 : RUN pip3 install -r requirements.txt
--> Running in a6f0fc592439
Collecting requests
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
Collecting paho-mqtt
  Downloading paho-mqtt-1.6.1.tar.gz (99 kB)
Collecting pika
  Downloading pika-1.3.2-py3-none-any.whl (155 kB)
Collecting certifi>2017.4.17
  Downloading certifi-2023.7.22-py3-none-any.whl (158 kB)
Collecting urllib3<3, >1.21.1
  Downloading urllib3-2.1.0-py3-none-any.whl (104 kB)
Collecting charset-normalizer<4, >2
  Downloading charset_normalizer-3.3.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (141 kB)
Collecting idna<4, >2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)

Step 5/6 : RUN pip3 install -r requirements.txt
--> Running in a6f0fc592439
Collecting requests
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
Collecting paho-mqtt
  Downloading paho-mqtt-1.6.1.tar.gz (99 kB)
Collecting pika
  Downloading pika-1.3.2-py3-none-any.whl (155 kB)
Collecting certifi>2017.4.17
  Downloading certifi-2023.7.22-py3-none-any.whl (158 kB)
Collecting urllib3<3, >1.21.1
  Downloading urllib3-2.1.0-py3-none-any.whl (104 kB)
Collecting charset-normalizer<4, >2
  Downloading charset_normalizer-3.3.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (141 kB)
Collecting idna<4, >2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
Building wheels for collected packages: paho-mqtt
  Building wheel for paho-mqtt (setup.py): started
  Building wheel for paho-mqtt (setup.py): finished with status 'done'
  Created wheel for paho-mqtt: filename=paho_mqtt-1.6.1-py3-none-any.whl size=62133 sha256=8d93e797b26d35e3d6509fe7e2a8a4defef5677076cf828c977c22f761b90b044
  Stored in directory: /root/.cache/pip/wheels/6a/48/01/c895c027e9b9367ec5470fb371ee56e795a49ac6a19aa4c9f
Successfully built paho-mqtt
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests, pika, paho-mqtt
Successfully installed certifi-2023.7.22 charset-normalizer-3.3.2 idna-3.4 paho-mqtt-1.6.1 pika-1.3.2 requests-2.31.0 urllib3-2.1.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.2.4; however, version 23.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
Removing intermediate container a6f0fc592439
--> b924d3c76ca8
Step 6/6 : CMD python3 datapreprocessor.py
--> Running in 12871044c6b2
Removing intermediate container 12871044c6b2
--> f2158c2344f7
Successfully built f2158c2344f7
Successfully tagged preprocessor:latest
student@edge:~/Documents/coursework/data_preprocessor$ 
```

Figure 11: Building an image of “data preprocessing operator” (task 2) python code

Output of Task 2 – data preprocessing operator (Running as a docker image)

```
student@edge:~/Documents/coursework/data_preprocessors$ docker-compose up
Starting data_preprocessor_data_preprocessor_1 ... done
Attaching to data_preprocessor_data_preprocessor_1
data_preprocessor_1 | Connected to MQTT OK!
data_preprocessor_1 | Printing PM2.5 data collected by emqx subscriber: [{"Value": 4.273, "Timestamp": 1685577600000}, {"Value": 3.483, "Timestamp": 1685578500000}, {"Value": 3.713, "Timestamp": 1685579400000}, {"Value": 3.813, "Timestamp": 1685580300000}, {"Value": 3.885, "Timestamp": 1685581200000}, {"Value": 3.809, "Timestamp": 1685582100000}, {"Value": 3.869, "Timestamp": 1685583000000}, {"Value": 3.839, "Timestamp": 1685583900000}, {"Value": 3.805, "Timestamp": 1685584800000}, {"Value": 4.094, "Timestamp": 1685585700000}, {"Value": 4.474, "Timestamp": 1685586600000}, {"Value": 4.918, "Timestamp": 1685587500000}, {"Value": 5.56, "Timestamp": 1685588400000}, {"Value": 5.637, "Timestamp": 1685589300000}, {"Value": 5.71, "Timestamp": 1685590200000}, {"Value": 6.004, "Timestamp": 1685591100000}, {"Value": 5.322, "Timestamp": 1685592000000}, {"Value": 4.961, "Timestamp": 1685592900000}, {"Value": 4.679, "Timestamp": 1685593800000}, {"Value": 4.815, "Timestamp": 1685594700000}, {"Value": 4.498, "Timestamp": 1685595600000}, {"Value": 4.393, "Timestamp": 1685596500000}, {"Value": 4.459, "Timestamp": 1685597400000}, {"Value": 4.467, "Timestamp": 1685598300000}, {"Value": 5.76, "Timestamp": 1685599200000}, {"Value": 4.549, "Timestamp": 1685600100000}, {"Value": 4.499, "Timestamp": 1685601000000}, {"Value": 4.946, "Timestamp": 1685601900000}, {"Value": 4.864, "Timestamp": 1685602800000}, {"Value": 6.079, "Timestamp": 1685603700000}, {"Value": 5.902, "Timestamp": 1685604600000}, {"Value": 6.025, "Timestamp": 1685605500000}, {"Value": 3.99, "Timestamp": 1685606400000}, {"Value": 3.223, "Timestamp": 1685607300000}, {"Value": 2.434, "Timestamp": 1685608200000}, {"Value": 2.07, "Timestamp": 1685609100000}, {"Value": 1.968, "Timestamp": 168561000000}, {"Value": 2.497, "Timestamp": 1685610900000}, {"Value": 3.192, "Timestamp": 1685611800000}, {"Value": 3.166, "Timestamp": 1685612700000}, {"Value": 2.994, "Timestamp": 1685613600000}, {"Value": 3.023, "Timestamp": 1685614500000}, {"Value": 3.652, "Timestamp": 1685615400000}, {"Value": 2.243, "Timestamp": 1685616300000}, {"Value": 2.652, "Timestamp": 1685617200000}, {"Value": 3.137, "Timestamp": 1685618100000}, {"Value": 3.075, "Timestamp": 1685619000000}, {"Value": 3.376, "Timestamp": 1685619900000}, {"Value": 3.494, "Timestamp": 1685620800000}, {"Value": 3.639, "Timestamp": 1685621700000}, {"Value": 3.826, "Timestamp": 1685622600000}, {"Value": 3.888, "Timestamp": 1685623500000}, {"Value": 4.861, "Timestamp": 1685624400000}, {"Value": 5.634, "Timestamp": 1685625300000}, {"Value": 5.097, "Timestamp": 1685626200000}, {"Value": 5.414, "Timestamp": 1685627100000}, {"Value": 6.029, "Timestamp": 1685628000000}, {"Value": 5.582, "Timestamp": 1685628900000}, {"Value": 5.885, "Timestamp": 1685629800000}, {"Value": 4.228, "Timestamp": 1685630700000}, {"Value": 4.729, "Timestamp": 1685631600000}, {"Value": 4.843, "Timestamp": 1685632500000}, {"Value": 5.195, "Timestamp": 1685633400000}, {"Value": 4.848, "Timestamp": 1685634300000}, {"Value": 4.279, "Timestamp": 1685635200000}, {"Value": 4.905, "Timestamp": 1685636100000}, {"Value": 3.846, "Timestamp": 1685637000000}, {"Value": 3.344, "Timestamp": 1685637900000}, {"Value": 3.533, "Timestamp": 1685638800000}, {"Value": 3.769, "Timestamp": 1685639700000}, {"Value": 3.571, "Timestamp": 1685640600000}, {"Value": 4.762, "Timestamp": 1685641500000}, {"Value": 4.365, "Timestamp": 1685642400000}, {"Value": 3.379, "Timestamp": 1685643300000}, {"Value": 3.65, "Timestamp": 1685644200000}, {"Value": 4.062, "Timestamp": 1685645100000}, {"Value": 4.384, "Timestamp": 1685646000000}, {"Value": 4.546, "Timestamp": 1685646900000}, {"Value": 4.785, "Timestamp": 1685647800000}, {"Value": 4.646, "Timestamp": 1685648700000}, {"Value": 5.195, "Timestamp": 1685653400000}, {"Value": 4.017, "Timestamp": 1685650500000}, {"Value": 4.171, "Timestamp": 1685651400000}, {"Value": 4.372, "Timestamp": 1685652300000}, {"Value": 4.599, "Timestamp": 1685653200000}, {"Value": 5.043, "Timestamp": 1685654100000}, {"Value": 5.614, "Timestamp": 1685655000000}, {"Value": 5.468, "Timestamp": 1685655900000}, {"Value": 5.435, "Timestamp": 1685656800000}, {"Value": 5.385, "Timestamp": 1685657700000}, {"Value": 4.732, "Timestamp": 1685658600000}, {"Value": 5.129, "Timestamp": 1685659500000}, {"Value": 5.128, "Timestamp": 1685660400000}, {"Value": 5.514, "Timestamp": 1685661300000}, {"Value": 5.505, "Timestamp": 1685662200000}, {"Value": 5.388, "Timestamp": 1685663100000}, {"Value": 5.3, "Timestamp": 1685664000000}, {"Value": 5.065, "Timestamp": 1685664900000}, {"Value": 4.82, "Timestamp": 1685665800000}, {"Value": 4.175, "Timestamp": 1685666700000}, {"Value": 4.468, "Timestamp": 1685667600000}, {"Value": 4.627, "Timestamp": 1685668500000}, {"Value": 4.113, "Timestamp": 1685669400000}, {"Value": 4.006, "Timestamp": 1685670300000}, {"Value": 3.897, "Timestamp": 1685671200000}, {"Value": 4.007, "Timestamp": 1685672100000}, {"Value": 4.156, "Timestamp": 1685673000000}, {"Value": 4.15, "Timestamp": 1685673900000}, {"Value": 4.105, "Timestamp": 1685674800000}, {"Value": 4.167, "Timestamp": 1685675700000}, {"Value": 4.159, "Timestamp": 1685676600000}, {"Value": 4.255, "Timestamp": 1685677500000}, {"Value": 4.713, "Timestamp": 1685678400000}, {"Value": 4.72, "Timestamp": 1685679300000}, {"Value": 4.92, "Timestamp": 1685680200000}, {"Value": 4.78, "Timestamp": 1685681100000}, {"Value": 4.654
```

Figure 12: Printing PM2.5 data collected by EMQX subscriber

```
data_preprocessor_1 | ****
data_preprocessor_1 | outliers : [{"Value": 52.17, "Timestamp": 1686800700000}, {"Value": 69.76, "Timestamp": 1686801600000}, {"Value": 76.49, "Timestamp": 1686802500000}, {"Value": 67.01, "Timestamp": 1686803400000}, {"Value": 59.91, "Timestamp": 1686804300000}]
data_preprocessor_1 | ****
```

Figure 13: Printing outliers data

```

data_preprocessor_1 | Printing daily average value data: [{"Timestamp": "2023-06-01", "Value": 4.385895833333334}, {"Timestamp": "2023-06-02", "Value": 7.36638541666666}, {"Timestamp": "2023-06-03", "Value": 8.777760416666665}, {"Timestamp": "2023-06-04", "Value": 6.630322916666674}, {"Timestamp": "2023-06-05", "Value": 5.260854166666669}, {"Timestamp": "2023-06-06", "Value": 4.187145833333333}, {"Timestamp": "2023-06-07", "Value": 3.729812499999999}, {"Timestamp": "2023-06-08", "Value": 4.797479166666667}, {"Timestamp": "2023-06-09", "Value": 7.573000000000001}, {"Timestamp": "2023-06-10", "Value": 11.063021052631582}, {"Timestamp": "2023-06-11", "Value": 11.867697916666673}, {"Timestamp": "2023-06-12", "Value": 16.1621875}, {"Timestamp": "2023-06-13", "Value": 13.97729166666668}, {"Timestamp": "2023-06-14", "Value": 10.226114583333338}, {"Timestamp": "2023-06-15", "Value": 12.477032967032972}, {"Timestamp": "2023-06-16", "Value": 8.82558333333329}, {"Timestamp": "2023-06-17", "Value": 9.852770833333333}, {"Timestamp": "2023-06-18", "Value": 13.412083333333337}, {"Timestamp": "2023-06-19", "Value": 5.1428125}, {"Timestamp": "2023-06-20", "Value": 4.90309375}, {"Timestamp": "2023-06-21", "Value": 5.316906249999999}, {"Timestamp": "2023-06-22", "Value": 6.623447916666667}, {"Timestamp": "2023-06-23", "Value": 7.079802083333333}, {"Timestamp": "2023-06-24", "Value": 3.768552083333336}, {"Timestamp": "2023-06-25", "Value": 5.430583333333334}, {"Timestamp": "2023-06-26", "Value": 3.519583333333346}, {"Timestamp": "2023-06-27", "Value": 5.316}, {"Timestamp": "2023-06-28", "Value": 2.396125}, {"Timestamp": "2023-06-29", "Value": 3.8314210526315784}, {"Timestamp": "2023-06-30", "Value": 4.62815624999999}, {"Timestamp": "2023-07-01", "Value": 5.144260416666666}, {"Timestamp": "2023-07-02", "Value": 3.585864583333327}, {"Timestamp": "2023-07-03", "Value": 3.149072916666666}, {"Timestamp": "2023-07-04", "Value": 3.4640937500000004}, {"Timestamp": "2023-07-05", "Value": 2.9548421052631584}, {"Timestamp": "2023-07-06", "Value": 3.925979166666675}, {"Timestamp": "2023-07-07", "Value": 6.59318749999999}, {"Timestamp": "2023-07-08", "Value": 2.9548421052631584}, {"Timestamp": "2023-07-09", "Value": 5.5420000000000025}, {"Timestamp": "2023-07-10", "Value": 4.6813333333335}, {"Timestamp": "2023-07-11", "Value": 3.634583333333334}, {"Timestamp": "2023-07-12", "Value": 2.435614583333334}, {"Timestamp": "2023-07-13", "Value": 3.727458333333333}, {"Timestamp": "2023-07-14", "Value": 4.982510416666667}, {"Timestamp": "2023-07-15", "Value": 3.917114583333316}, {"Timestamp": "2023-07-16", "Value": 2.34790625}, {"Timestamp": "2023-07-17", "Value": 2.771697916666667}, {"Timestamp": "2023-07-18", "Value": 4.388156249999999}, {"Timestamp": "2023-07-19", "Value": 3.357354166666668}, {"Timestamp": "2023-07-20", "Value": 8.176978723404257}, {"Timestamp": "2023-07-21", "Value": 5.90583333333333}, {"Timestamp": "2023-07-22", "Value": 2.590458333333346}, {"Timestamp": "2023-07-23", "Value": 4.50412499999999}, {"Timestamp": "2023-07-24", "Value": 3.373854166666655}, {"Timestamp": "2023-07-25", "Value": 3.372104166666665}, {"Timestamp": "2023-07-26", "Value": 5.87120833333333}, {"Timestamp": "2023-07-27", "Value": 3.819885416666663}, {"Timestamp": "2023-07-28", "Value": 3.220374999999999}, {"Timestamp": "2023-07-29", "Value": 3.443239583333334}, {"Timestamp": "2023-07-30", "Value": 3.405125000000004}, {"Timestamp": "2023-07-31", "Value": 2.3632083333333327}, {"Timestamp": "2023-08-01", "Value": 2.346687499999994}, {"Timestamp": "2023-08-02", "Value": 5.44892708333333}, {"Timestamp": "2023-08-03", "Value": 3.075510416666667}, {"Timestamp": "2023-08-04", "Value": 2.773083333333327}, {"Timestamp": "2023-08-05", "Value": 3.981489583333334}, {"Timestamp": "2023-08-06", "Value": 3.412427083333346}, {"Timestamp": "2023-08-07", "Value": 2.852000000000003}, {"Timestamp": "2023-08-08", "Value": 2.4533750000000007}, {"Timestamp": "2023-08-09", "Value": 3.79507291666666}, {"Timestamp": "2023-08-10", "Value": 7.58115624999999}, {"Timestamp": "2023-08-11", "Value": 6.79160416666667}, {"Timestamp": "2023-08-12", "Value": 4.817375000000001}, {"Timestamp": "2023-08-13", "Value": 2.85765625}, {"Timestamp": "2023-08-14", "Value": 3.96567083333333}, {"Timestamp": "2023-08-15", "Value": 4.33408333333333}, {"Timestamp": "2023-08-16", "Value": 5.386010416666667}, {"Timestamp": "2023-08-17", "Value": 3.63768749999999}, {"Timestamp": "2023-08-18", "Value": 5.8835416666668}, {"Timestamp": "2023-08-19", "Value": 4.635322580645163}, {"Timestamp": "2023-08-20", "Value": 2.0207916666666}, {"Timestamp": "2023-08-21", "Value": 3.571218750000003}, {"Timestamp": "2023-08-22", "Value": 3.153270833333334}, {"Timestamp": "2023-08-23", "Value": 3.331718750000002}, {"Timestamp": "2023-08-24", "Value": 3.622135416666693}, {"Timestamp": "2023-08-25", "Value": 3.6153416666668}, {"Timestamp": "2023-08-26", "Value": 2.801312500000003}, {"Timestamp": "2023-08-27", "Value": 2.810552083333344}, {"Timestamp": "2023-08-28", "Value": 2.92603124999999}, {"Timestamp": "2023-08-29", "Value": 5.10029166666666}, {"Timestamp": "2023-08-30", "Value": 3.7789042553191488}, {"Timestamp": "2023-08-31", "Value": 3.295}]


```

Figure 14: **Printing daily average value data**

Task 3: Data prediction and visualization

This task requires installation of dependencies such as pika, prophet, matplotlib, plotly and pandas on cloud virtual machine. Commands to install these dependencies are as below,

[pip install pika](#) – RabbitMQ client library for python.

[pip install prophet](#) – Python library which is used for forecasting timeseries based data.

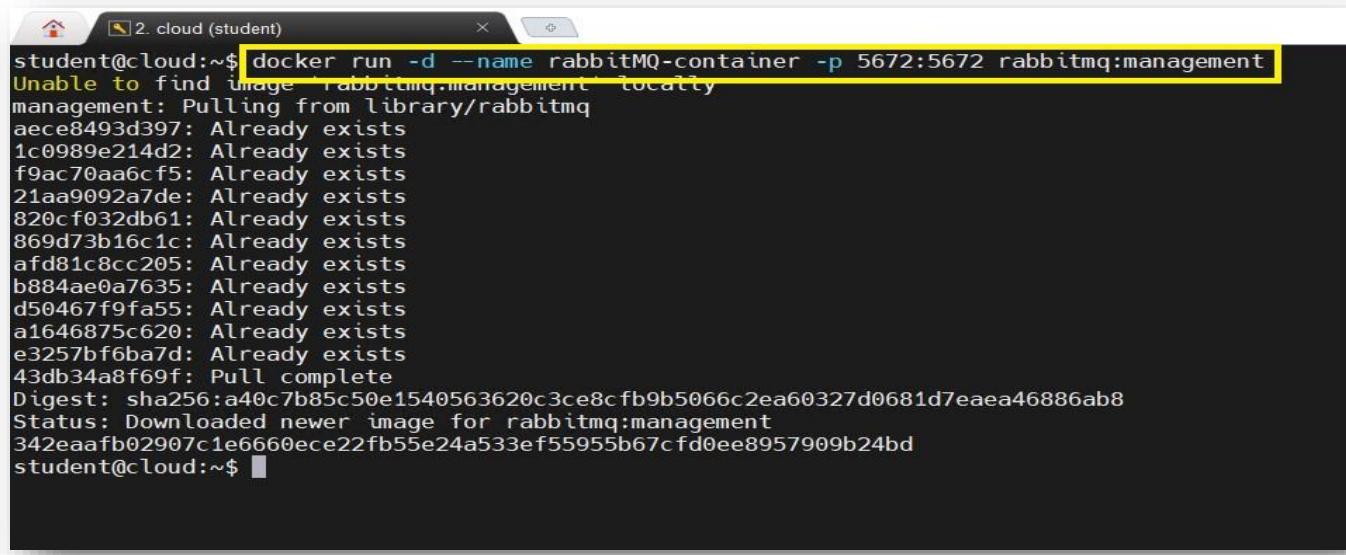
[pip install matplotlib](#) – Python library used for visualization purpose.

[pip install --upgrade plotly](#) – Python plotting library.

[pip install pandas](#) – Python library which is used for data manipulation and data analysis.

As we need to consume data sent by RabbitMQ producer in task 2, we need to pull and run RabbitMQ docker image in cloud virtual machine.

We execute below command on cloud virtual machine to pull and run RabbitMQ image.



```
student@cloud:~$ docker run -d --name rabbitMQ-container -p 5672:5672 rabbitmq:management
Unable to find image "rabbitmq:management" locally
management: Pulling from library/rabbitmq
aece8493d397: Already exists
1c0989e214d2: Already exists
f9ac70aa6cf5: Already exists
21aa9092a7de: Already exists
820cf032db61: Already exists
869d73b16c1c: Already exists
afdb81c8cc205: Already exists
b884ae0a7635: Already exists
d50467f9fa55: Already exists
a1646875c620: Already exists
e3257bf6ba7d: Already exists
43db34a8f69f: Pull complete
Digest: sha256:a40c7b85c50e1540563620c3ce8cfb9b5066c2ea60327d0681d7eaee46886ab8
Status: Downloaded newer image for rabbitmq:management
342eaafb02907c1e6660ece22fb55e24a533ef55955b67cf0ee8957909b24bd
student@cloud:~$
```

Figure 15: Pulling and running RabbitMQ image on cloud virtual machine

Code Description

In task 3 first we are importing all the required libraries such as json, pika, matplotlib, pandas and datetime. Then we are connecting to the RabbitMQ service in order to consume the data that is sent by RabbitMQ producer in task2. Once we receive the data in cloud layer, first we have to reformat the timestamp to date time format and print the reformatted timestamp on console.

There are two functions written with the name “visualize” and “predict” which are used for visualizing average PM2.5 daily data and to predict the trend of PM2.5 for next 15 days respectively. We pass the data with reformatted timestamp to both of these functions as a parameter. For machine learning model to predict the data for next 15 days, it makes use of a predefined ml_engine code. This predefined ml_engine code is downloaded and stored in the same directory where we are storing the python code file for task 3.

Code snippet for task 3

```
datapredictor.py  ×
1 import json
2 import pika
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 from datetime import datetime
6 from ml_engine import MLPredictor
7
8 if __name__ == '__main__':
9     rabbitmq_ip = "192.168.0.100"
10    rabbitmq_port = 5672
11    # Queue name
12    rabbitmq_queue = "CSC8112test"
13
14    # Feeding this time converted data to matplotlib for visualization
15    def visualize(timeconvertdata):
16        data_df = pd.DataFrame(timeconvertdata)
17
18        # Initialize a canvas
19        plt.figure(figsize=(80, 40), dpi=60)
20        # Plot data into canvas
21        plt.plot(data_df["Timestamp"], data_df["Value"], color="#FF3B1D", marker='.', linestyle="-")
22        plt.title("Example data for demonstration")
23        plt.xlabel("DateTime")
24        plt.ylabel("Value")
25        plt.xticks(rotation=90)
26        # Save as file
27        plt.savefig("figure1.png")
28        # Directly display
29        #plt.show()
30
31    # supplying the data to ML engine in order to train and predict values for next 15 days
32    def predict(timeconvertdata):
33
34        data_df = pd.DataFrame(timeconvertdata)
35
36        # Create ML engine predictor object
37        predictor = MLPredictor(data_df)
38        # Train ML model
39        predictor.train()
40        # Do prediction
41        forecast = predictor.predict()
42
43        # Get canvas
44        fig = predictor.plot_result(forecast)
45        fig.savefig("prediction.png")
46        #fig.show()
47
48    def callback(ch, method, properties, body):
49
50        print(f"Printing PM2.5 average value data collected by RabbitMQ consumer: {json.loads(body)}")
51
52        consumeddata = json.loads(body)
53
54        # convert this consumeddata in required timestamp format
55        timeconvertdata = []
56
57        for i in consumeddata:
58            date = datetime.strptime(i['Timestamp'], '%Y-%m-%d').strftime('%Y-%m-%d %H:%M:%S')
59            timeconvertdata.append({'Timestamp': date, 'Value': i['Value']})
60
61        # printing PM2.5 data with reformatted timestamp
62        print("*****PM2.5 data with reformat timestamp: ", timeconvertdata)
63        print("*****PM2.5 data with reformat timestamp: ", timeconvertdata)
64        print("*****PM2.5 data with reformat timestamp: ", timeconvertdata)
65        print("*****PM2.5 data with reformat timestamp: ", timeconvertdata)
66
67        visualize(timeconvertdata)
68        predict(timeconvertdata)
69
70
71
72
73
74    # Connect to RabbitMQ service with timeout 1min
75    connection = pika.BlockingConnection(
76        pika.ConnectionParameters(host=rabbitmq_ip, port=rabbitmq_port, socket_timeout=60))
77
78    channel = connection.channel()
79    # Declare a queue
80    channel.queue_declare(queue=rabbitmq_queue)
81
82    channel.basic_consume(queue=rabbitmq_queue,
83                           auto_ack=True,
84                           on_message_callback=callback)
85
86    channel.start_consuming()
```

Figure 16: Task 3 python code

Code snippet for ml_engine

```
1 """
2     Author: Rui Sun
3     Date: 2022-09-25
4
5     Predict timeseries data
6
7     Official guide book of Prophet: https://facebook.github.io/prophet/docs/quick\_start.html#python-api
8 ...
9
10 from prophet import Prophet
11
12 class MLPredictor(object):
13     """
14         Example usage method:
15     """
16     from ml_engine import MLPredictor
17
18     predictor = MLPredictor(pm25_df)
19     predictor.train()
20     forecast = predictor.predict()
21
22
23     fig = predictor.plot_result(forecast)
24     fig.savefig(os.path.join("Your target dir path", "Your target file name"))
25
26 ...
27
28 def __init__(self, data_df):
29     """
30         :param data_df: Dataframe type dataset
31     """
32     self.__train_data = self.__convert_col_name(data_df)
33     self.__trainer = Prophet(changepoint_prior_scale=12)
34
35     def train(self):
36         self.__trainer.fit(self.__train_data)
37
38     def __convert_col_name(self, data_df):
39         data_df.rename(columns={"Timestamp": "ds", "Value": "y"}, inplace=True)
40         print(f"After rename columns \n{data_df.columns}")
41         return data_df
42
43     def __make_future(self, periods=15):
44         future = self.__trainer.make_future_dataframe(periods=periods)
45         return future
46
47     def predict(self):
48         future = self.__make_future()
49         forecast = self.__trainer.predict(future)
50         return forecast
51
52     def plot_result(self, forecast):
53         fig = self.__trainer.plot(forecast, figsize=(15, 6))
54         return fig
55
```

Figure 17: ml_engine code

Output of Task 3: data prediction and visualization

```
student@cloud:~/Documents/courseworks$ python3 datapredictor.py
Printing PM2.5 average value data collected by RabbitMQ consumer: [{"Timestamp": "2023-06-01", "Value": 4.385895833333334}, {"Timestamp": "2023-06-02", "Value": 7.366385416666666}, {"Timestamp": "2023-06-03", "Value": 8.777604166666665}, {"Timestamp": "2023-06-04", "Value": 6.630322916666674}, {"Timestamp": "2023-06-05", "Value": 5.260854166666669}, {"Timestamp": "2023-06-06", "Value": 4.187145833333333}, {"Timestamp": "2023-06-07", "Value": 3.729812499999999}, {"Timestamp": "2023-06-08", "Value": 4.797479166666667}, {"Timestamp": "2023-06-09", "Value": 5.753000000000001}, {"Timestamp": "2023-06-10", "Value": 11.063021052631582}, {"Timestamp": "2023-06-11", "Value": 11.867697916666673}, {"Timestamp": "2023-06-12", "Value": 16.1621875}, {"Timestamp": "2023-06-13", "Value": 13.977291666666668}, {"Timestamp": "2023-06-14", "Value": 10.226114583333338}, {"Timestamp": "2023-06-15", "Value": 12.477032967032972}, {"Timestamp": "2023-06-16", "Value": 8.825583333333329}, {"Timestamp": "2023-06-17", "Value": 9.852770833333333}, {"Timestamp": "2023-06-18", "Value": 13.412083333333337}, {"Timestamp": "2023-06-19", "Value": 5.1421825}, {"Timestamp": "2023-06-20", "Value": 4.90309375}, {"Timestamp": "2023-06-21", "Value": 5.316906249999999}, {"Timestamp": "2023-06-22", "Value": 6.623447916666667}, {"Timestamp": "2023-06-23", "Value": 7.07982083333333}, {"Timestamp": "2023-06-24", "Value": 3.768552083333336}, {"Timestamp": "2023-06-25", "Value": 5.43058333333334}, {"Timestamp": "2023-06-26", "Value": 3.519583333333346}, {"Timestamp": "2023-06-27", "Value": 5.316}, {"Timestamp": "2023-06-28", "Value": 2.396125}, {"Timestamp": "2023-06-29", "Value": 3.8314210526315784}, {"Timestamp": "2023-06-30", "Value": 4.628156249999999}, {"Timestamp": "2023-07-01", "Value": 5.144260416666666}, {"Timestamp": "2023-07-02", "Value": 3.5858465833333327}, {"Timestamp": "2023-07-03", "Value": 3.149072916666666}, {"Timestamp": "2023-07-04", "Value": 3.4640937500000004}, {"Timestamp": "2023-07-05", "Value": 2.9548421052631584}, {"Timestamp": "2023-07-06", "Value": 3.925979166666675}, {"Timestamp": "2023-07-07", "Value": 6.593187499999999}, {"Timestamp": "2023-07-08", "Value": 12.6866770833333}, {"Timestamp": "2023-07-09", "Value": 5.5420000000000025}, {"Timestamp": "2023-07-10", "Value": 4.681333333333335}, {"Timestamp": "2023-07-11", "Value": 3.63458333333334}, {"Timestamp": "2023-07-12", "Value": 2.435614583333334}, {"Timestamp": "2023-07-13", "Value": 3.727458333333333}, {"Timestamp": "2023-07-14", "Value": 4.982510416666667}, {"Timestamp": "2023-07-15", "Value": 3.917114583333316}, {"Timestamp": "2023-07-16", "Value": 2.34790625}, {"Timestamp": "2023-07-17", "Value": 2.771697916666667}, {"Timestamp": "2023-07-18", "Value": 4.388156249999999}, {"Timestamp": "2023-07-19", "Value": 3.357354166666665}, {"Timestamp": "2023-07-20", "Value": 8.17697823404257}, {"Timestamp": "2023-07-21", "Value": 5.905833333333333}, {"Timestamp": "2023-07-22", "Value": 2.590458333333346}, {"Timestamp": "2023-07-23", "Value": 4.504124999999999}, {"Timestamp": "2023-07-24", "Value": 3.373854166666655}, {"Timestamp": "2023-07-25", "Value": 3.372104166666665}, {"Timestamp": "2023-07-26", "Value": 5.871208333333333}, {"Timestamp": "2023-07-27", "Value": 3.819885416666666}, {"Timestamp": "2023-07-28", "Value": 3.220374999999999}, {"Timestamp": "2023-07-29", "Value": 3.443239583333334}, {"Timestamp": "2023-07-30", "Value": 3.405125000000004}, {"Timestamp": "2023-07-31", "Value": 2.363208333333327}, {"Timestamp": "2023-08-01", "Value": 2.346687499999994}, {"Timestamp": "2023-08-02", "Value": 5.448927083333333}, {"Timestamp": "2023-08-03", "Value": 3.075510416666667}, {"Timestamp": "2023-08-04", "Value": 2.7730833333333327}, {"Timestamp": "2023-08-05", "Value": 3.98148958333334}, {"Timestamp": "2023-08-06", "Value": 3.412427083333346}, {"Timestamp": "2023-08-07", "Value": 2.852000000000003}, {"Timestamp": "2023-08-08", "Value": 2.453375000000007}, {"Timestamp": "2023-08-09", "Value": 3.795072916666667}, {"Timestamp": "2023-08-10", "Value": 7.581156249999999}, {"Timestamp": "2023-08-11", "Value": 6.791604166666667}, {"Timestamp": "2023-08-12", "Value": 4.817375000000001}, {"Timestamp": "2023-08-13", "Value": 2.85765625}, {"Timestamp": "2023-08-14", "Value": 3.965677083333333}, {"Timestamp": "2023-08-15", "Value": 4.334083333333333}, {"Timestamp": "2023-08-16", "Value": 5.386010416666667}, {"Timestamp": "2023-08-17", "Value": 3.637687499999999}, {"Timestamp": "2023-08-18", "Value": 5.8835416666668}, {"Timestamp": "2023-08-19", "Value": 4.63532258064516}, {"Timestamp": "2023-08-20", "Value": 3.571218750000003}, {"Timestamp": "2023-08-21", "Value": 3.153270833333344}, {"Timestamp": "2023-08-22", "Value": 2.020791666666666}, {"Timestamp": "2023-08-23", "Value": 3.33171875000002}, {"Timestamp": "2023-08-24", "Value": 3.622135416666669}, {"Timestamp": "2023-08-25", "Value": 3.615354166666668}, {"Timestamp": "2023-08-26", "Value": 2.801312500000003}, {"Timestamp": "2023-08-27", "Value": 2.810552083333344}, {"Timestamp": "2023-08-28", "Value": 2.926031249999999}, {"Timestamp": "2023-08-29", "Value": 5.10029166666666}, {"Timestamp": "2023-08-30", "Value": 3.778904253191488}, {"Timestamp": "2023-08-31", "Value": 3.295}]]
```

Figure 18: Printing average PM2.5 data collected by RabbitMQ consumer

Figure 19: Printing data with reformatted timestamp

```
*****
After rename columns
Index(['ds', 'y'], dtype='object')
12:18:28 - cmdstanpy - INFO - Chain [1] start processing
12:18:28 - cmdstanpy - INFO - Chain [1] done processing
```

Figure 20: plotting graphs successfully

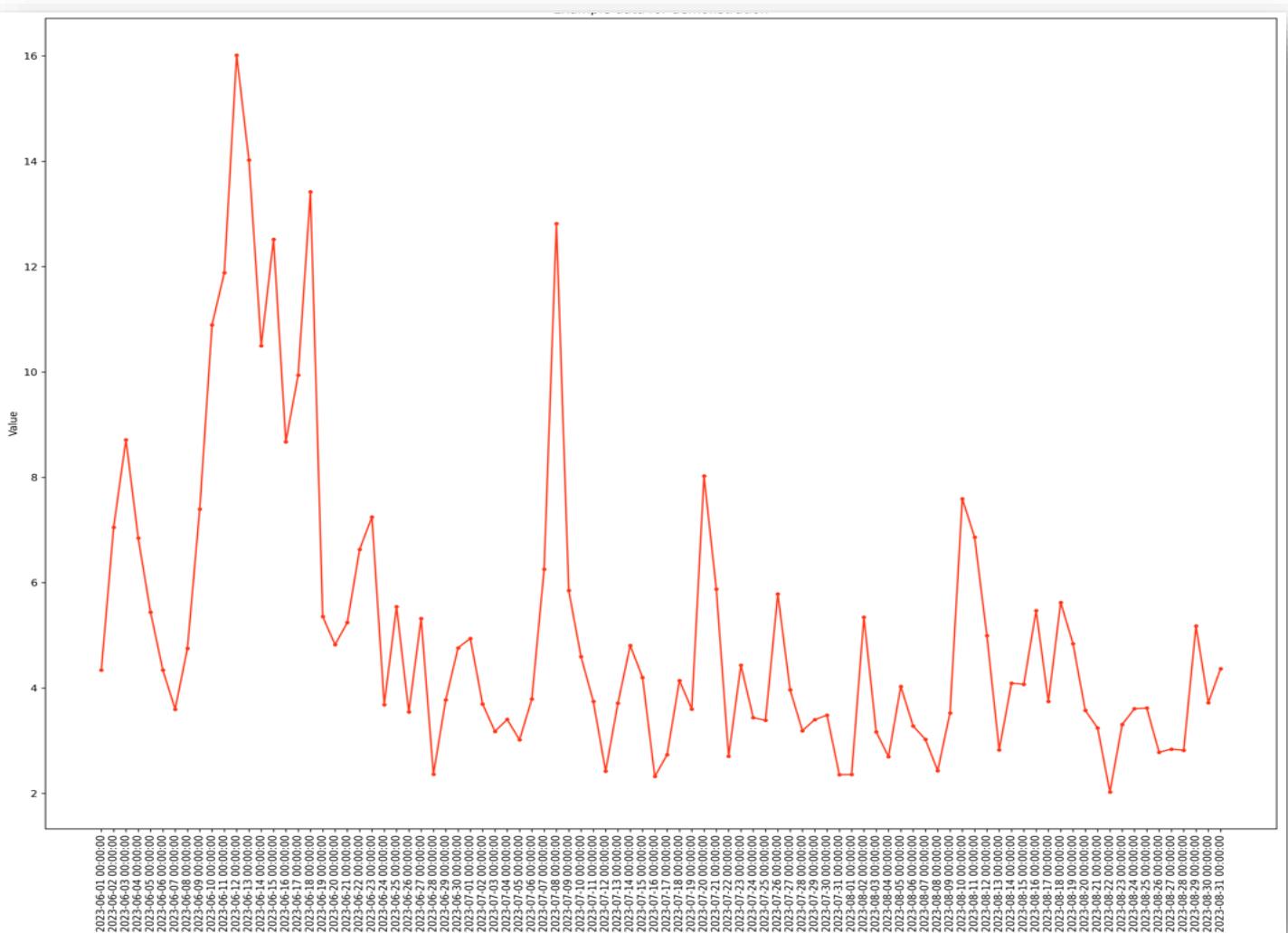


Figure 21: Data Visualization line chart

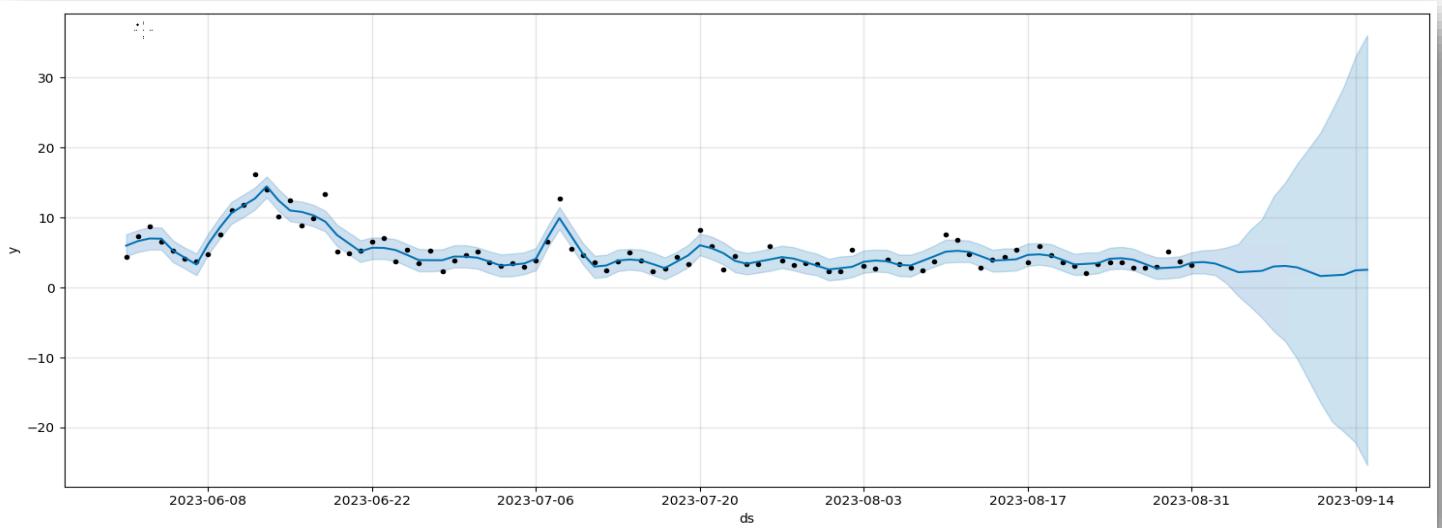


Figure 22: Data prediction graph

Screenshot of running services in docker environment

```
student@edge:~/Documents/coursework/data_injector$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
1b98bc391edb  preprocessor:latest "/bin/sh -c 'python3..." 3 minutes ago Up 3 minutes
data_preprocessor_data_preprocessor_1
0f1911418113  rabbitmq:management "docker-entrypoint.s..." 4 hours ago Up 4 hours 4369/tcp, 5671/tcp, 15671-15672/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp rabbitMQ-container
2df06e003287  emqx/emqx:latest "/usr/bin/docker-ent..." 6 hours ago Up 6 hours 4370/tcp, 5369/tcp, 8083-8084/tcp, 8883/tcp, 11883/tcp, 0.0.0.0:1883->1883/tcp, :::1883->1883/tcp, 18083/tcp emqx
student@edge:~/Documents/coursework/data_injector$
```

Figure 23: Docker services running on edge virtual machine

```
student@cloud:~$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
4fb2063c0b64  rabbitmq:management "docker-entrypoint.s..." 2 days ago Up 2 days 4369/tcp, 5671/tcp, 15671-15672/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp rabbitMQ-container
```

Figure 24: Docker services running on cloud virtual machine

Analytical discussion of results and related conclusion

Looking at the data visualization graph it is observed that for few days in month of June, 2023 the value of PM2.5 was considerably high as compared to rest of the days in selected duration. Also, in month of July we can see that there was a sudden spike in value of PM2.5 between 7th to 9th July, 2023 and then it became normal later. This might have happened due to some one-off activity or some external factors.

Looking at the forecast of first 15 days for the month of September, we can expect stable values of PM2.5 data with no sudden spikes. Overall if we compare starting from June month then we can see a declining trend in values of PM2.5 data over a period of time.

Finally, to conclude, this was an engaging project providing hands on experience and knowledge about collecting and processing data fetched from IOT sensors. Data flow between IOT, edge and cloud layer using MQTT and AMQP protocols. Using docker to pull and run required images and creating docker images of our own code and utilizing it further. Preprocessing and cleaning the raw data before passing it for visualization and prediction using machine learning model.