# Internet of Things – CSC8112 Coursework 2023-24 Project Report

Ronil Rodrigues
MSc Cloud Computing
230112209

# Table of Contents

## AIM:

The aim of this coursework is to process sensor data in the edge and cloud environment and able to implement machine learning model to create IOT data processing pipeline. The pipeline will be able to do operations like data collection, data pre -processing, data visualization and data prediction in edge cloud setting, and be able to use a lightweight virtualization technology stack, such as Docker, to implement IOT data processing pipeline in the edge cloud setting.

## Introduction:

The coursework focuses on creating an IoT pipeline by sourcing input from the Urban Observatory and subsequently channeling the data through various layers, including Azure Edge and the cloud, to generate the desired results. Figure 1 below illustrates the flow of the IoT pipeline.

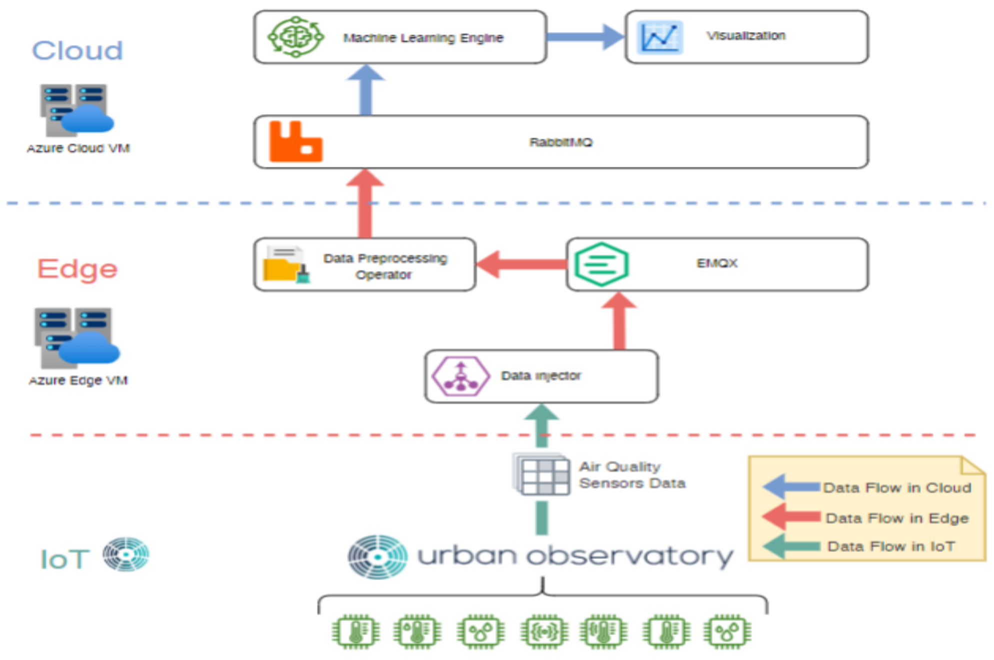


Figure 1: Overview

## IoT tier:

• Newcastle Urban Observatory (NCL UO) [https://urbanobservatory.ac.uk/]: The largest set of publicly available real-time urban data in the UK. NCL UO sensors are gathering data across Newcastle city. With over 50 data types and counting, there are lots of live data for you to access.

## Edge tier:

• Data Injector: This will be a software component that you will design and implement in

Task 1, focusing on (i) reading data from Urban Observatory API and (ii) transmitting data to the machine learning pipeline.
• EMQX: A broker of MQTT protocol, a message queuing system given to you as a Docker image, which forms the basis for enabling asynchronous service-to-service communication in a complex Machine Learning (ML)-based IoT data processing pipeline.
• Data Pre-processing Operator: A software component that you will develop in Task 2, Responsible for preparing training data of Machine Learning model.

## Cloud tier:

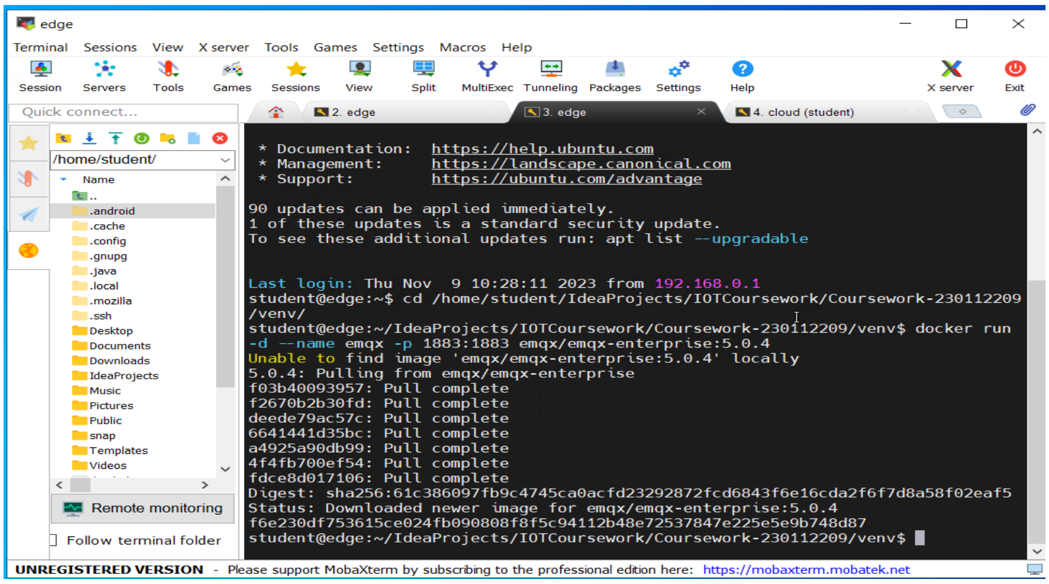
• RabbitMQ: A cloud-based message queuing system.
• Machine Learning Model/Classifier/Engine: A software component that can be trained to predict particular types of future events.
• Visualization: A component that will visualize the trend of raw time-series data and the prediction results (input from the Machine Learning Model/Classifier/Engine)

## IMPLEMENTATION:

### Task 1: Design a data injector component by leveraging Newcastle Urban Observatory IoT data streams

#### 1) Pull and run the Docker image "emqx/emqx" from Docker Hub in the virtual machine running on Azure lab (Edge). Perform this task first using the command line interface (CLI).

We execute the Docker image emqx/emqx with the command “docker run -d --name emqx -p 1883:1883 emqx/emqx-enterprise:5.0.4”. The image is fetched from Docker Hub and operates on the Edge layer. The Emqx broker is currently operational on the edge layer.



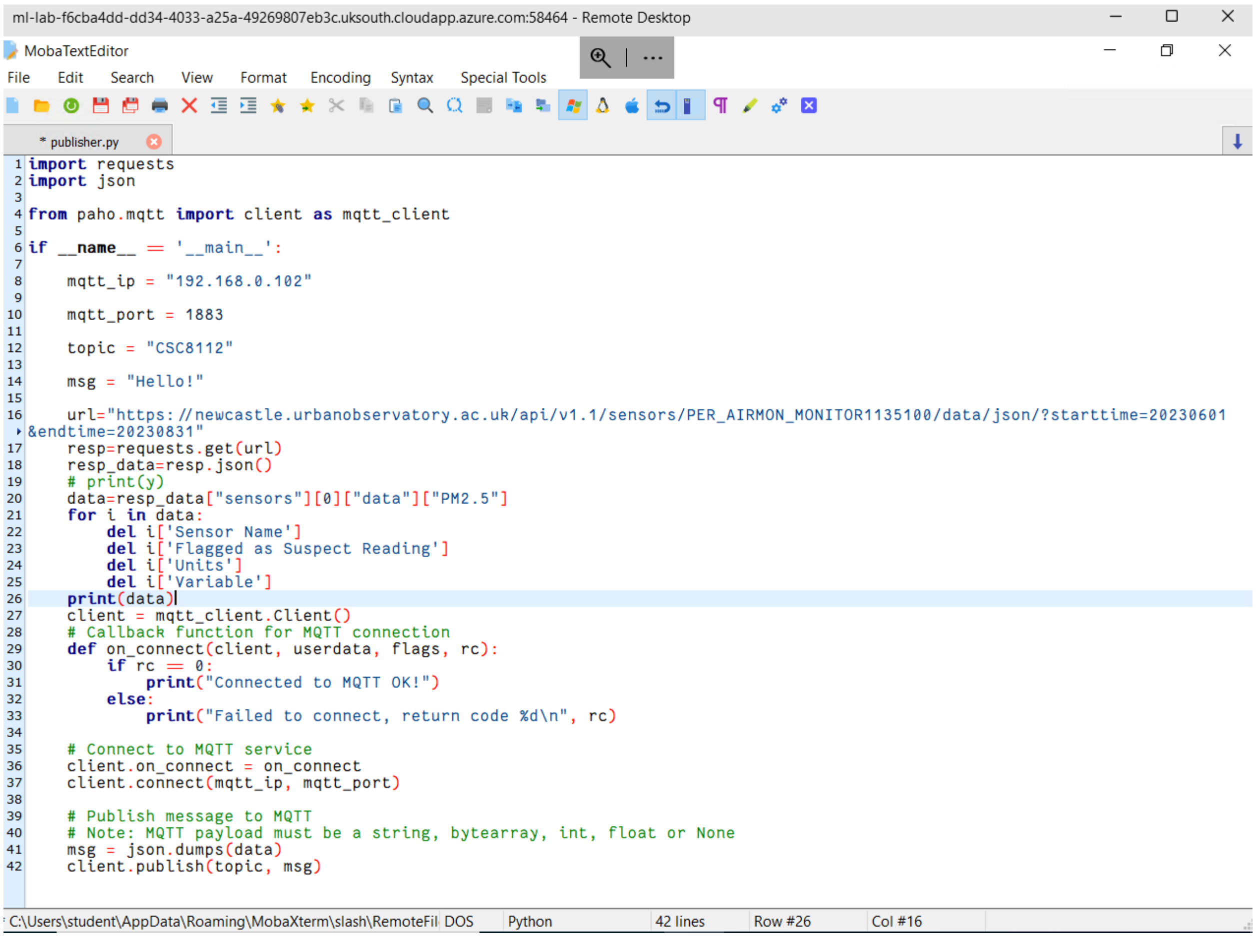
```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

90 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable


Last login: Thu Nov  9 10:28:11 2023 from 192.168.0.1
student@edge:~$ cd /home/student/IdeaProjects/IOTCoursework/Coursework-230112209
/venv/
student@edge:~/IdeaProjects/IOTCoursework/Coursework-230112209/venv$ docker run
-d --name emqx -p 1883:1883 emqx/emqx-enterprise:5.0.4
Unable to find image 'emqx/emqx-enterprise:5.0.4' locally
5.0.4: Pulling from emqx/emqx-enterprise
f03b40093957: Pull complete
f2670b2b30fd: Pull complete
deede79ac57c: Pull complete
6641441d35bc: Pull complete
a4925a90db99: Pull complete
4f4fb700ef54: Pull complete
fdce8d017106: Pull complete
Digest: sha256:61c386097fb9c4745ca0acfd23292872fcd6843f6e16cda2f6f7d8a58f02eaf5
Status: Downloaded newer image for emqx/emqx-enterprise:5.0.4
f6e230df753615ce024fb090808f8f5c94112b48e72537847e225e5e9b748d87
student@edge:~/IdeaProjects/IOTCoursework/Coursework-230112209/venv$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

## 2) Develop a data injector component with the following functions (Code) in Azure Lab (Edge) or the Azure Lab Localhost:

The Following code is developed for MQTT Publisher. In this code we receive data from Urban Observatory using the provided URL, which is then filtered out to get 2.5PM data. This 2.5PM data is then send to MQTT Subscriber using the Emqx broker operating in the edge layer.

## MQTT PUBLISHER CODE:

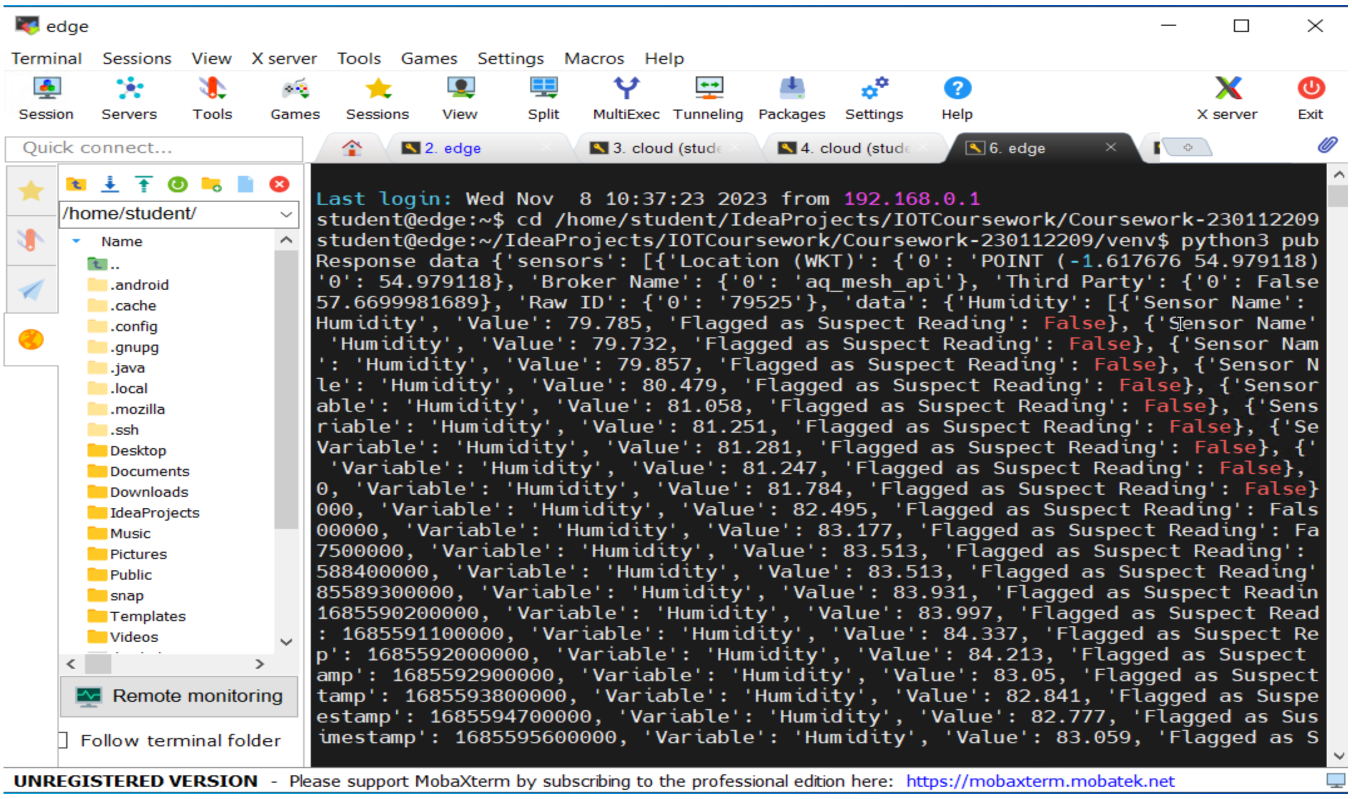

ml-lab-f6cba4dd-dd34-4033-a25a-49269807eb3c.uksouth.cloudapp.azure.com:58464 - Remote Desktop
MobaTextEditor
File Edit Search View Format Encoding Syntax Special Tools
* publisher.py

```python
import requests
import json

from paho.mqtt import client as mqtt_client

if __name__ == '__main__':

    mqtt_ip = "192.168.0.102"

    mqtt_port = 1883

    topic = "CSC8112"

    msg = "Hello!"

    url="https://newcastle.urbanobservatory.ac.uk/api/v1.1/sensors/PER_AIRMON_MONITOR1135100/data/json/?starttime=20230601
&endtime=20230831"
    resp=requests.get(url)
    resp_data=resp.json()
    # print(y)
    data=resp_data["sensors"][0]["data"]["PM2.5"]
    for i in data:
        del i['Sensor Name']
        del i['Flagged as Suspect Reading']
        del i['Units']
        del i['Variable']
    print(data)
    client = mqtt_client.Client()
    # Callback function for MQTT connection
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT OK!")
        else:
            print("Failed to connect, return code %d\n", rc)

    # Connect to MQTT service
    client.on_connect = on_connect
    client.connect(mqtt_ip, mqtt_port)

    # Publish message to MQTT
    # Note: MQTT payload must be a string, bytearray, int, float or None
    msg = json.dumps(data)
    client.publish(topic, msg)
```

C:\Users\student\AppData\Roaming\MobaXterm\slash\RemoteFil DOS
Python
42 lines
Row #26
Col #16

## A) Collect data from Urban Observatory platform by sending HTTP request to the following url ([http://uoweb3.ncl.ac.uk/api/v1.1/sensors/PER_AIRMON_MONITOR1135100/data/json/?starttime=20230601&endtime=20230831]). Following that, please print out the raw data streams that you collected on the console.

The below figure shows output after running publisher.py, we are able to get raw data from Newcastle urban observatory.

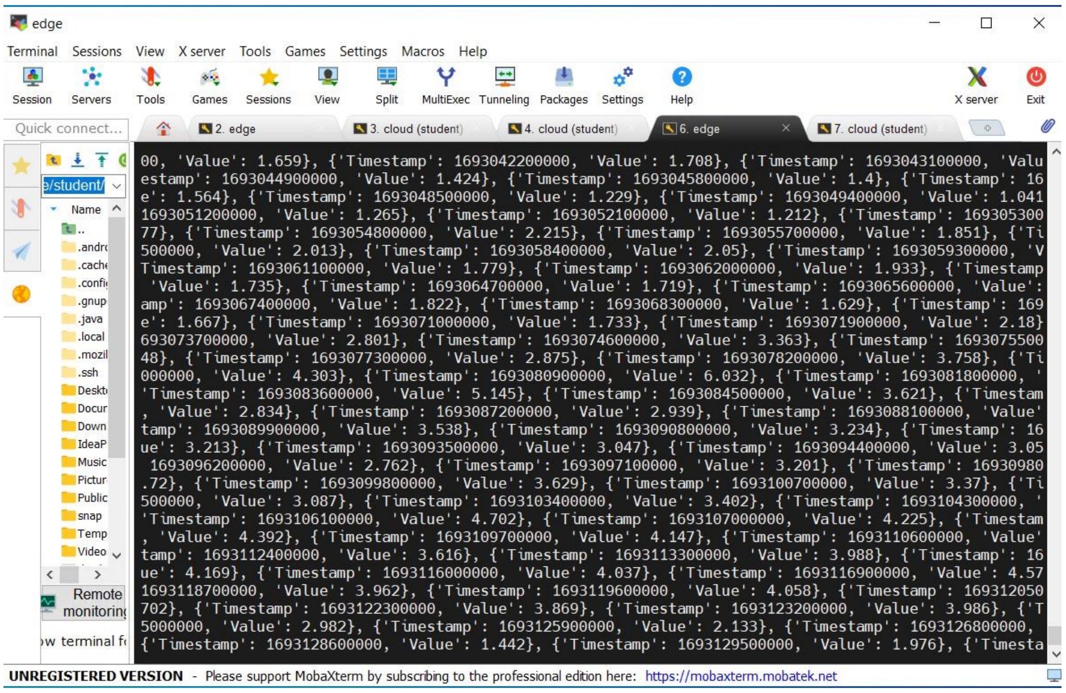


```
Last login: Wed Nov  8 10:37:23 2023 from 192.168.0.1
student@edge:~$ cd /home/student/IdeaProjects/IOTCoursework/Coursework-230112209
student@edge:~/IdeaProjects/IOTCoursework/Coursework-230112209/venv$ python3 pub
Response data {'sensors': [{'Location (WKT)': {'0': 'POINT (-1.617676 54.979118)
'0': 54.979118}, 'Broker Name': {'0': 'aq_mesh_api'}, 'Third Party': {'0': False
57.6699981689}, 'Raw ID': {'0': '79525'}, 'data': {'Humidity': [{'Sensor Name':
Humidity', 'Value': 79.785, 'Flagged as Suspect Reading': False}, {'Sensor Name'
 'Humidity', 'Value': 79.732, 'Flagged as Suspect Reading': False}, {'Sensor Nam
': 'Humidity', 'Value': 79.857, 'Flagged as Suspect Reading': False}, {'Sensor N
le': 'Humidity', 'Value': 80.479, 'Flagged as Suspect Reading': False}, {'Sensor
able': 'Humidity', 'Value': 81.058, 'Flagged as Suspect Reading': False}, {'Sens
riable': 'Humidity', 'Value': 81.251, 'Flagged as Suspect Reading': False}, {'Se
Variable': 'Humidity', 'Value': 81.281, 'Flagged as Suspect Reading': False}, {'
 'Variable': 'Humidity', 'Value': 81.247, 'Flagged as Suspect Reading': False},
0, 'Variable': 'Humidity', 'Value': 81.784, 'Flagged as Suspect Reading': False}
000, 'Variable': 'Humidity', 'Value': 82.495, 'Flagged as Suspect Reading': Fals
00000, 'Variable': 'Humidity', 'Value': 83.177, 'Flagged as Suspect Reading': Fa
7500000, 'Variable': 'Humidity', 'Value': 83.513, 'Flagged as Suspect Reading':
588400000, 'Variable': 'Humidity', 'Value': 83.513, 'Flagged as Suspect Reading'
85589300000, 'Variable': 'Humidity', 'Value': 83.931, 'Flagged as Suspect Readin
1685590200000, 'Variable': 'Humidity', 'Value': 83.997, 'Flagged as Suspect Read
: 1685591100000, 'Variable': 'Humidity', 'Value': 84.337, 'Flagged as Suspect Re
p': 1685592000000, 'Variable': 'Humidity', 'Value': 84.213, 'Flagged as Suspect
amp': 1685592900000, 'Variable': 'Humidity', 'Value': 83.05, 'Flagged as Suspect
tamp': 1685593800000, 'Variable': 'Humidity', 'Value': 82.841, 'Flagged as Suspe
estamp': 1685594700000, 'Variable': 'Humidity', 'Value': 82.777, 'Flagged as Sus
imestamp': 1685595600000, 'Variable': 'Humidity', 'Value': 83.059, 'Flagged as S
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

(b) **Although the raw air quality data you collected from the Urban Observatory API contains many metrics including NO2, NO, CO2, PM2.5, and PM10, among others, for the purpose of this coursework you only need to store and analyze PM2.5 data. While many meta-data are available for PM2.5 data, such as sensor name, timestamp, value, and location, you only need to store the metrics related to the Timestamp and Value meta-data fields.**

We filter out the data obtained from Urban Observatory API to include only the Time stamp and Value meta-data fields and then print out the same on console.

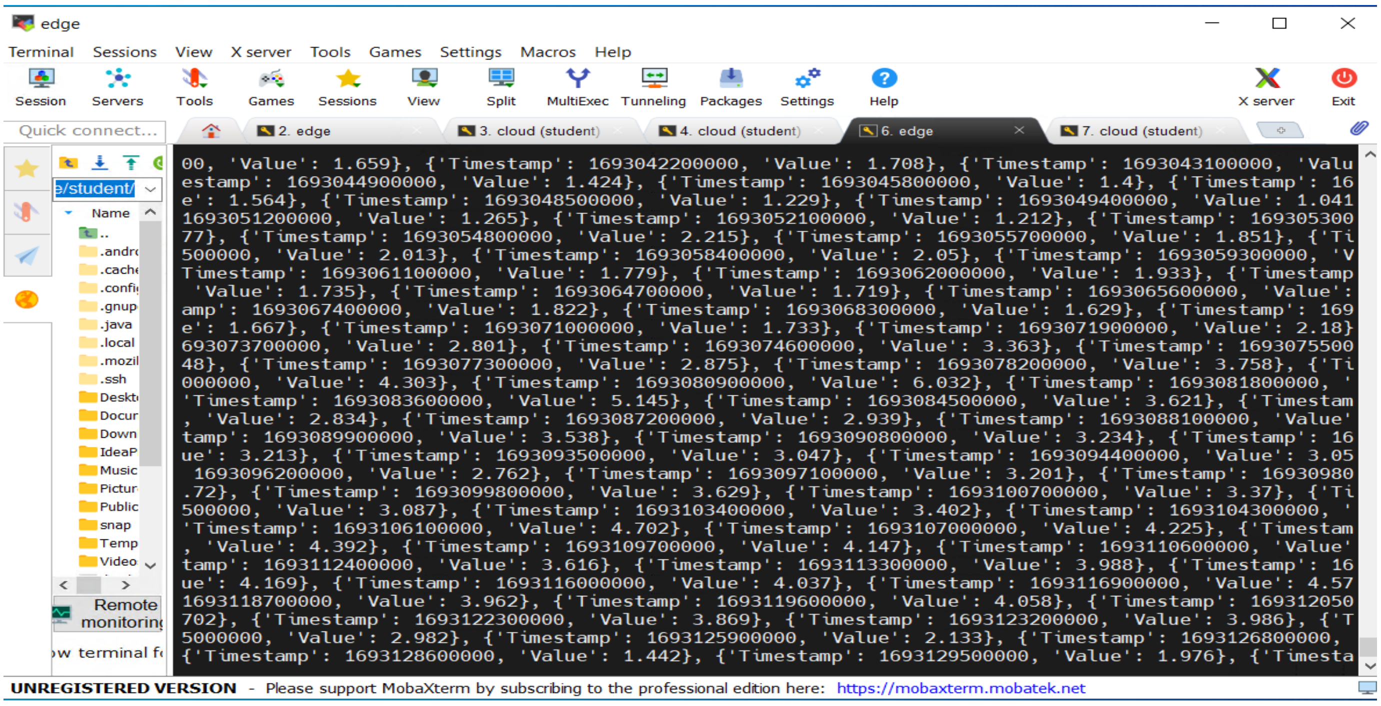

edge
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. edge
3. cloud (student)
4. cloud (student)
6. edge
7. cloud (student)
e/student/
Name
Remote monitoring

```
00, 'Value': 1.659}, {'Timestamp': 1693042200000, 'Value': 1.708}, {'Timestamp': 1693043100000, 'Valu
estamp': 1693044900000, 'Value': 1.424}, {'Timestamp': 1693045800000, 'Value': 1.4}, {'Timestamp': 16
e': 1.564}, {'Timestamp': 1693048500000, 'Value': 1.229}, {'Timestamp': 1693049400000, 'Value': 1.041
1693051200000, 'Value': 1.265}, {'Timestamp': 1693052100000, 'Value': 1.212}, {'Timestamp': 169305300
77}, {'Timestamp': 1693054800000, 'Value': 2.215}, {'Timestamp': 1693055700000, 'Value': 1.851}, {'Ti
500000, 'Value': 2.013}, {'Timestamp': 1693058400000, 'Value': 2.05}, {'Timestamp': 1693059300000, 'V
Timestamp': 1693061100000, 'Value': 1.779}, {'Timestamp': 1693062000000, 'Value': 1.933}, {'Timestamp
 'Value': 1.735}, {'Timestamp': 1693064700000, 'Value': 1.719}, {'Timestamp': 1693065600000, 'Value':
amp': 1693067400000, 'Value': 1.822}, {'Timestamp': 1693068300000, 'Value': 1.629}, {'Timestamp': 169
e': 1.667}, {'Timestamp': 1693071000000, 'Value': 1.733}, {'Timestamp': 1693071900000, 'Value': 2.18}
693073700000, 'Value': 2.801}, {'Timestamp': 1693074600000, 'Value': 3.363}, {'Timestamp': 1693075500
48}, {'Timestamp': 1693077300000, 'Value': 2.875}, {'Timestamp': 1693078200000, 'Value': 3.758}, {'Ti
000000, 'Value': 4.303}, {'Timestamp': 1693080900000, 'Value': 6.032}, {'Timestamp': 1693081800000, '
'Timestamp': 1693083600000, 'Value': 5.145}, {'Timestamp': 1693084500000, 'Value': 3.621}, {'Timestam
, 'Value': 2.834}, {'Timestamp': 1693087200000, 'Value': 2.939}, {'Timestamp': 1693088100000, 'Value'
tamp': 1693089900000, 'Value': 3.538}, {'Timestamp': 1693090800000, 'Value': 3.234}, {'Timestamp': 16
ue': 3.213}, {'Timestamp': 1693093500000, 'Value': 3.047}, {'Timestamp': 1693094400000, 'Value': 3.05
 1693096200000, 'Value': 2.762}, {'Timestamp': 1693097100000, 'Value': 3.201}, {'Timestamp': 16930980
.72}, {'Timestamp': 1693099800000, 'Value': 3.629}, {'Timestamp': 1693100700000, 'Value': 3.37}, {'Ti
500000, 'Value': 3.087}, {'Timestamp': 1693103400000, 'Value': 3.402}, {'Timestamp': 1693104300000, '
'Timestamp': 1693106100000, 'Value': 4.702}, {'Timestamp': 1693107000000, 'Value': 4.225}, {'Timestam
, 'Value': 4.392}, {'Timestamp': 1693109700000, 'Value': 4.147}, {'Timestamp': 1693110600000, 'Value'
tamp': 1693112400000, 'Value': 3.616}, {'Timestamp': 1693113300000, 'Value': 3.988}, {'Timestamp': 16
ue': 4.169}, {'Timestamp': 1693116000000, 'Value': 4.037}, {'Timestamp': 1693116900000, 'Value': 4.57
1693118700000, 'Value': 3.962}, {'Timestamp': 1693119600000, 'Value': 4.058}, {'Timestamp': 169312050
702}, {'Timestamp': 1693122300000, 'Value': 3.869}, {'Timestamp': 1693123200000, 'Value': 3.986}, {'T
5000000, 'Value': 2.982}, {'Timestamp': 1693125900000, 'Value': 2.133}, {'Timestamp': 1693126800000,
{'Timestamp': 1693128600000, 'Value': 1.442}, {'Timestamp': 1693129500000, 'Value': 1.976}, {'Timesta
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

(c) Send all PM2.5 data to be used by Task 2.2 (a) to EMQX service of Azure lab (Edge).

In the following code we filtered out data obtained from urban observatory and store metrics related to Timestamp and Value fields. The same 2.5PM data is shared to mqtt subscriber through emqx broker running in edge layer.

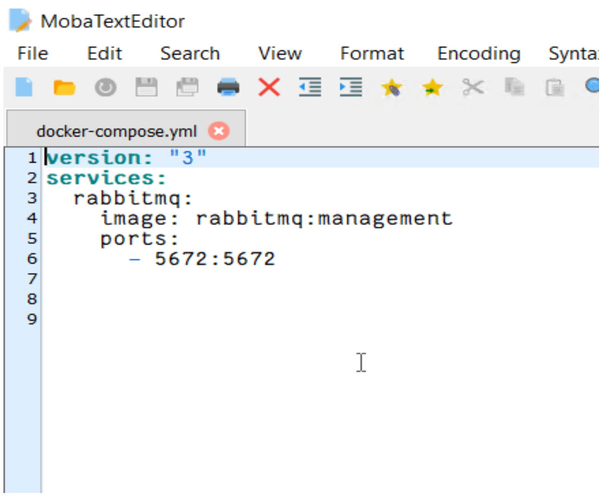

edge
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. edge
3. cloud (student)
4. cloud (student)
6. edge
7. cloud (student)
/student/
Name
..
.andro
.cache
.confi
.gnup
.java
.local
.mozil
.ssh
Deskt
Docur
Down
IdeaP
Music
Pictur
Public
snap
Temp
Video
Remote monitoring
ow terminal f
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
00, 'Value': 1.659}, {'Timestamp': 1693042200000, 'Value': 1.708}, {'Timestamp': 1693043100000, 'Valu
estamp': 1693044900000, 'Value': 1.424}, {'Timestamp': 1693045800000, 'Value': 1.4}, {'Timestamp': 16
e': 1.564}, {'Timestamp': 1693048500000, 'Value': 1.229}, {'Timestamp': 1693049400000, 'Value': 1.041
1693051200000, 'Value': 1.265}, {'Timestamp': 1693052100000, 'Value': 1.212}, {'Timestamp': 169305300
77}, {'Timestamp': 1693054800000, 'Value': 2.215}, {'Timestamp': 1693055700000, 'Value': 1.851}, {'Ti
500000, 'Value': 2.013}, {'Timestamp': 1693058400000, 'Value': 2.05}, {'Timestamp': 1693059300000, 'V
Timestamp': 1693061100000, 'Value': 1.779}, {'Timestamp': 1693062000000, 'Value': 1.933}, {'Timestamp
 'Value': 1.735}, {'Timestamp': 1693064700000, 'Value': 1.719}, {'Timestamp': 1693065600000, 'Value':
amp': 1693067400000, 'Value': 1.822}, {'Timestamp': 1693068300000, 'Value': 1.629}, {'Timestamp': 169
e': 1.667}, {'Timestamp': 1693071000000, 'Value': 1.733}, {'Timestamp': 1693071900000, 'Value': 2.18}
693073700000, 'Value': 2.801}, {'Timestamp': 1693074600000, 'Value': 3.363}, {'Timestamp': 1693075500
48}, {'Timestamp': 1693077300000, 'Value': 2.875}, {'Timestamp': 1693078200000, 'Value': 3.758}, {'Ti
000000, 'Value': 4.303}, {'Timestamp': 1693080900000, 'Value': 6.032}, {'Timestamp': 1693081800000, '
'Timestamp': 1693083600000, 'Value': 5.145}, {'Timestamp': 1693084500000, 'Value': 3.621}, {'Timestam
, 'Value': 2.834}, {'Timestamp': 1693087200000, 'Value': 2.939}, {'Timestamp': 1693088100000, 'Value'
tamp': 1693089900000, 'Value': 3.538}, {'Timestamp': 1693090800000, 'Value': 3.234}, {'Timestamp': 16
ue': 3.213}, {'Timestamp': 1693093500000, 'Value': 3.047}, {'Timestamp': 1693094400000, 'Value': 3.05
 1693096200000, 'Value': 2.762}, {'Timestamp': 1693097100000, 'Value': 3.201}, {'Timestamp': 16930980
.72}, {'Timestamp': 1693099800000, 'Value': 3.629}, {'Timestamp': 1693100700000, 'Value': 3.37}, {'Ti
500000, 'Value': 3.087}, {'Timestamp': 1693103400000, 'Value': 3.402}, {'Timestamp': 1693104300000, '
'Timestamp': 1693106100000, 'Value': 4.702}, {'Timestamp': 1693107000000, 'Value': 4.225}, {'Timestam
, 'Value': 4.392}, {'Timestamp': 1693109700000, 'Value': 4.147}, {'Timestamp': 1693110600000, 'Value'
tamp': 1693112400000, 'Value': 3.616}, {'Timestamp': 1693113300000, 'Value': 3.988}, {'Timestamp': 16
ue': 4.169}, {'Timestamp': 1693116000000, 'Value': 4.037}, {'Timestamp': 1693116900000, 'Value': 4.57
1693118700000, 'Value': 3.962}, {'Timestamp': 1693119600000, 'Value': 4.058}, {'Timestamp': 169312050
702}, {'Timestamp': 1693122300000, 'Value': 3.869}, {'Timestamp': 1693123200000, 'Value': 3.986}, {'T
5000000, 'Value': 2.982}, {'Timestamp': 1693125900000, 'Value': 2.133}, {'Timestamp': 1693126800000,
{'Timestamp': 1693128600000, 'Value': 1.442}, {'Timestamp': 1693129500000, 'Value': 1.976}, {'Timesta
```

## TASK 2: DATA PRE PROCESSING OPERATOR DESIGN:

1) Define a Docker compose file which contains the following necessary configurations and instructions for deploying and instantiating the following set of Docker images on Azure lab (Cloud):

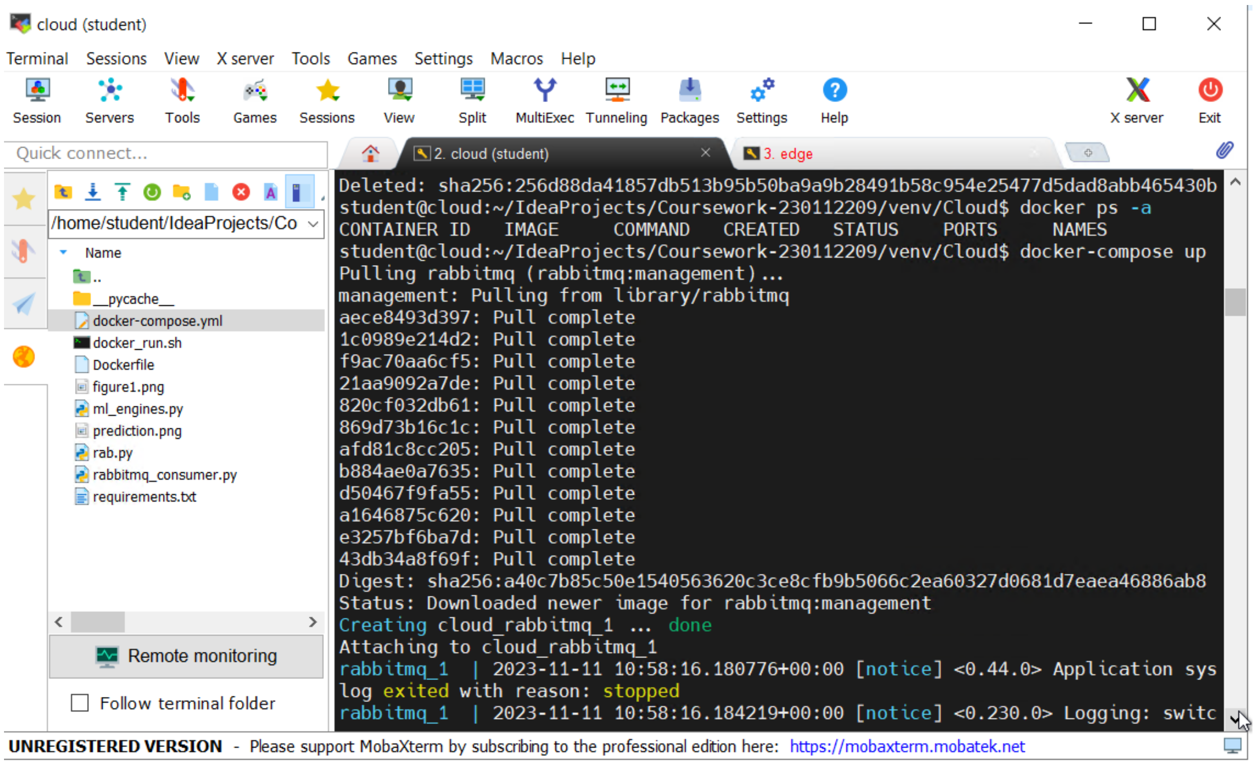
The below figure shows the Docker file created to run the docker image rabbitmq on the specified ports.



```
version: "3"
services:
  rabbitmq:
    image: rabbitmq:management
    ports:
      - 5672:5672
```

(a) Download and run RabbitMQ image (rabbitmq:management);

The below image shows RabbitMQ image running on Azure Lab (Cloud) after running docker-compose up command on cloud.



```
Deleted: sha256:256d88da41857db513b95b50ba9a9b28491b58c954e25477d5dad8abb465430b
student@cloud:~/IdeaProjects/Coursework-230112209/venv/Cloud$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
student@cloud:~/IdeaProjects/Coursework-230112209/venv/Cloud$ docker-compose up
Pulling rabbitmq (rabbitmq:management)...
management: Pulling from library/rabbitmq
aece8493d397: Pull complete
1c0989e214d2: Pull complete
f9ac70aa6cf5: Pull complete
21aa9092a7de: Pull complete
820cf032db61: Pull complete
869d73b16c1c: Pull complete
afd81c8cc205: Pull complete
b884ae0a7635: Pull complete
d50467f9fa55: Pull complete
a1646875c620: Pull complete
e3257bf6ba7d: Pull complete
43db34a8f69f: Pull complete
Digest: sha256:a40c7b85c50e1540563620c3ce8cfb9b5066c2ea60327d0681d7eaea46886ab8
Status: Downloaded newer image for rabbitmq:management
Creating cloud_rabbitmq_1 ... done
Attaching to cloud_rabbitmq_1
rabbitmq_1  | 2023-11-11 10:58:16.180776+00:00 [notice] <0.44.0> Application sys
log exited with reason: stopped
rabbitmq_1  | 2023-11-11 10:58:16.184219+00:00 [notice] <0.230.0> Logging: switc
```

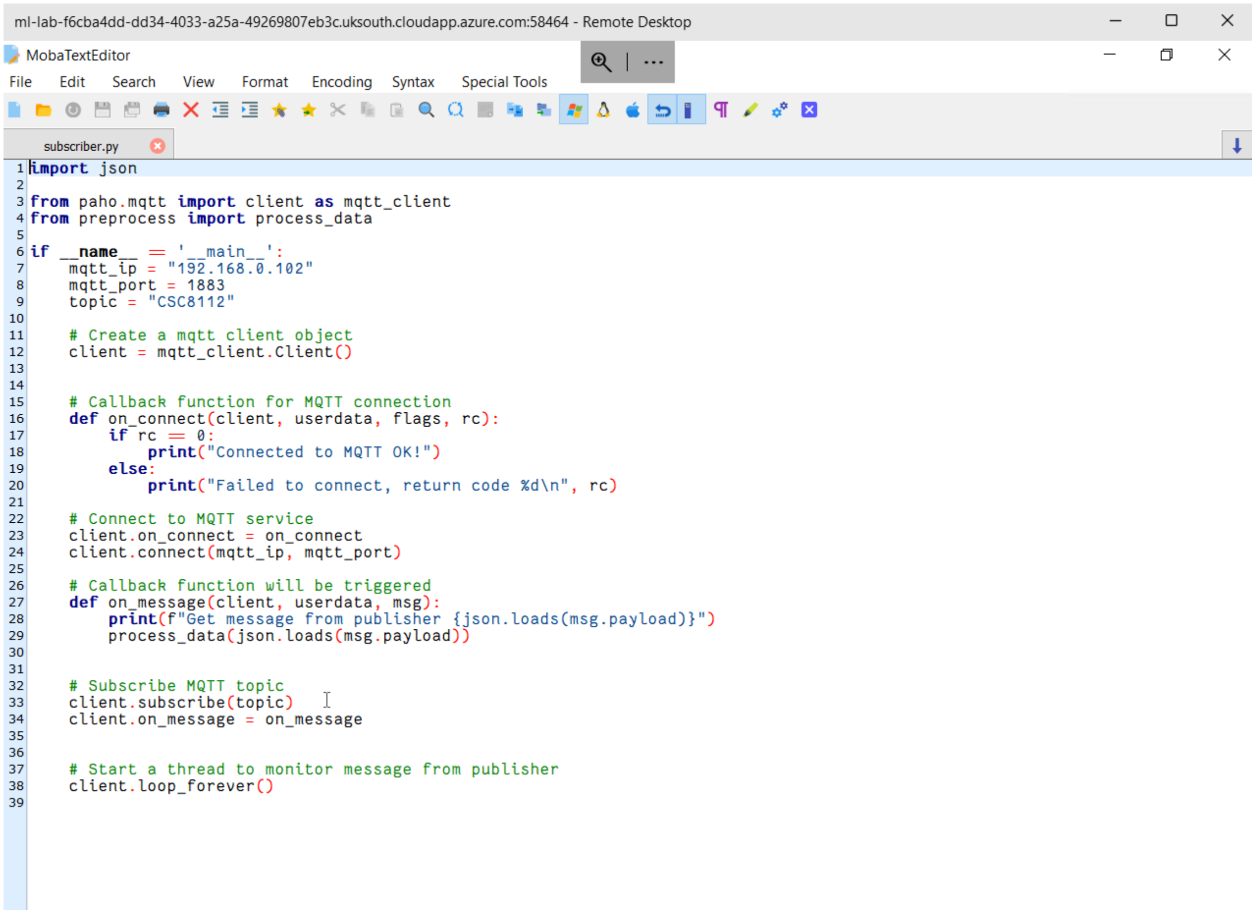
**UNREGISTERED VERSION** - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

## *2. DESIGN A DATA PREPROCESSING OPERATOR WITH THE FOLLOWING FUNCTIONS (CODE) IN AZURE LAB (EDGE):*

(a) Collect all PM2.5 data published by Task 1.2 (c) from EMQX service, and please print out the PM2.5 data to the console (this operator will run as a Docker container, so the logs can be seen in the Docker logs console automatically

The following code is written for MQTT Subscriber to collect PM2.5 data published in Task1 from EMQX service. The output obtained after running MQTT Subscriber is displayed below the code.

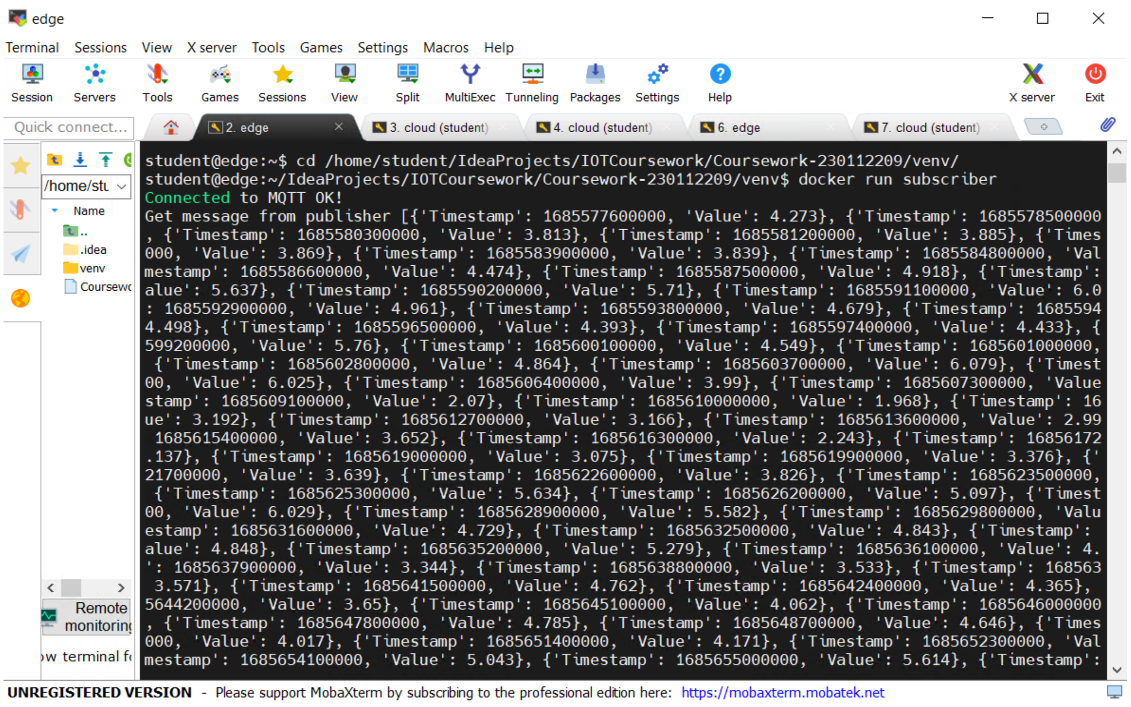
## MQTT_SUBSCRIBER CODE:



ml-lab-f6cba4dd-dd34-4033-a25a-49269807eb3c.uksouth.cloudapp.azure.com:58464 - Remote Desktop
MobaTextEditor
File Edit Search View Format Encoding Syntax Special Tools
subscriber.py

```
import json

from paho.mqtt import client as mqtt_client
from preprocess import process_data

if __name__ == '__main__':
    mqtt_ip = "192.168.0.102"
    mqtt_port = 1883
    topic = "CSC8112"

    # Create a mqtt client object
    client = mqtt_client.Client()


    # Callback function for MQTT connection
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT OK!")
        else:
            print("Failed to connect, return code %d\n", rc)

    # Connect to MQTT service
    client.on_connect = on_connect
    client.connect(mqtt_ip, mqtt_port)

    # Callback function will be triggered
    def on_message(client, userdata, msg):
        print(f"Get message from publisher {json.loads(msg.payload)}")
        process_data(json.loads(msg.payload))


    # Subscribe MQTT topic
    client.subscribe(topic)
    client.on_message = on_message


    # Start a thread to monitor message from publisher
    client.loop_forever()
```

Below is the PM2.5 data received in the edge layer by running the command “docker-compose up”.

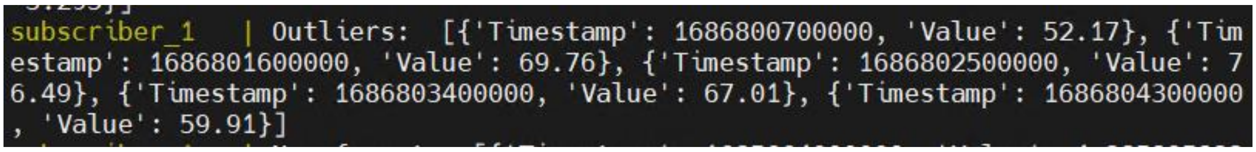


edge
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. edge
3. cloud (student)
4. cloud (student)
6. edge
7. cloud (student)
/home/stu
Name
..
.idea
venv
Coursewo
Remote monitoring
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
student@edge:~$ cd /home/student/IdeaProjects/IOTCoursework/Coursework-230112209/venv/
student@edge:~/IdeaProjects/IOTCoursework/Coursework-230112209/venv$ docker run subscriber
Connected to MQTT OK!
Get message from publisher [{'Timestamp': 1685577600000, 'Value': 4.273}, {'Timestamp': 1685578500000
, {'Timestamp': 1685580300000, 'Value': 3.813}, {'Timestamp': 1685581200000, 'Value': 3.885}, {'Times
000, 'Value': 3.869}, {'Timestamp': 1685583900000, 'Value': 3.839}, {'Timestamp': 1685584800000, 'Val
mestamp': 1685586600000, 'Value': 4.474}, {'Timestamp': 1685587500000, 'Value': 4.918}, {'Timestamp':
alue': 5.637}, {'Timestamp': 1685590200000, 'Value': 5.71}, {'Timestamp': 1685591100000, 'Value': 6.0
: 1685592900000, 'Value': 4.961}, {'Timestamp': 1685593800000, 'Value': 4.679}, {'Timestamp': 1685594
4.498}, {'Timestamp': 1685596500000, 'Value': 4.393}, {'Timestamp': 1685597400000, 'Value': 4.433}, {
599200000, 'Value': 5.76}, {'Timestamp': 1685600100000, 'Value': 4.549}, {'Timestamp': 1685601000000,
 {'Timestamp': 1685602800000, 'Value': 4.864}, {'Timestamp': 1685603700000, 'Value': 6.079}, {'Timest
00, 'Value': 6.025}, {'Timestamp': 1685606400000, 'Value': 3.99}, {'Timestamp': 1685607300000, 'Value
stamp': 1685609100000, 'Value': 2.07}, {'Timestamp': 1685610000000, 'Value': 1.968}, {'Timestamp': 16
ue': 3.192}, {'Timestamp': 1685612700000, 'Value': 3.166}, {'Timestamp': 1685613600000, 'Value': 2.99
 1685615400000, 'Value': 3.652}, {'Timestamp': 1685616300000, 'Value': 2.243}, {'Timestamp': 16856172
.137}, {'Timestamp': 1685619000000, 'Value': 3.075}, {'Timestamp': 1685619900000, 'Value': 3.376}, {'
21700000, 'Value': 3.639}, {'Timestamp': 1685622600000, 'Value': 3.826}, {'Timestamp': 1685623500000,
 {'Timestamp': 1685625300000, 'Value': 5.634}, {'Timestamp': 1685626200000, 'Value': 5.097}, {'Timest
00, 'Value': 6.029}, {'Timestamp': 1685628900000, 'Value': 5.582}, {'Timestamp': 1685629800000, 'Valu
estamp': 1685631600000, 'Value': 4.729}, {'Timestamp': 1685632500000, 'Value': 4.843}, {'Timestamp':
alue': 4.848}, {'Timestamp': 1685635200000, 'Value': 5.279}, {'Timestamp': 1685636100000, 'Value': 4.
': 1685637900000, 'Value': 3.344}, {'Timestamp': 1685638800000, 'Value': 3.533}, {'Timestamp': 168563
 3.571}, {'Timestamp': 1685641500000, 'Value': 4.762}, {'Timestamp': 1685642400000, 'Value': 4.365},
5644200000, 'Value': 3.65}, {'Timestamp': 1685645100000, 'Value': 4.062}, {'Timestamp': 1685646000000
, {'Timestamp': 1685647800000, 'Value': 4.785}, {'Timestamp': 1685648700000, 'Value': 4.646}, {'Times
000, 'Value': 4.017}, {'Timestamp': 1685651400000, 'Value': 4.171}, {'Timestamp': 1685652300000, 'Val
mestamp': 1685654100000, 'Value': 5.043}, {'Timestamp': 1685655000000, 'Value': 5.614}, {'Timestamp':
```

(b) Filter out outliers (the value greater than 50), and please print out outliers to the console (docker logs console).

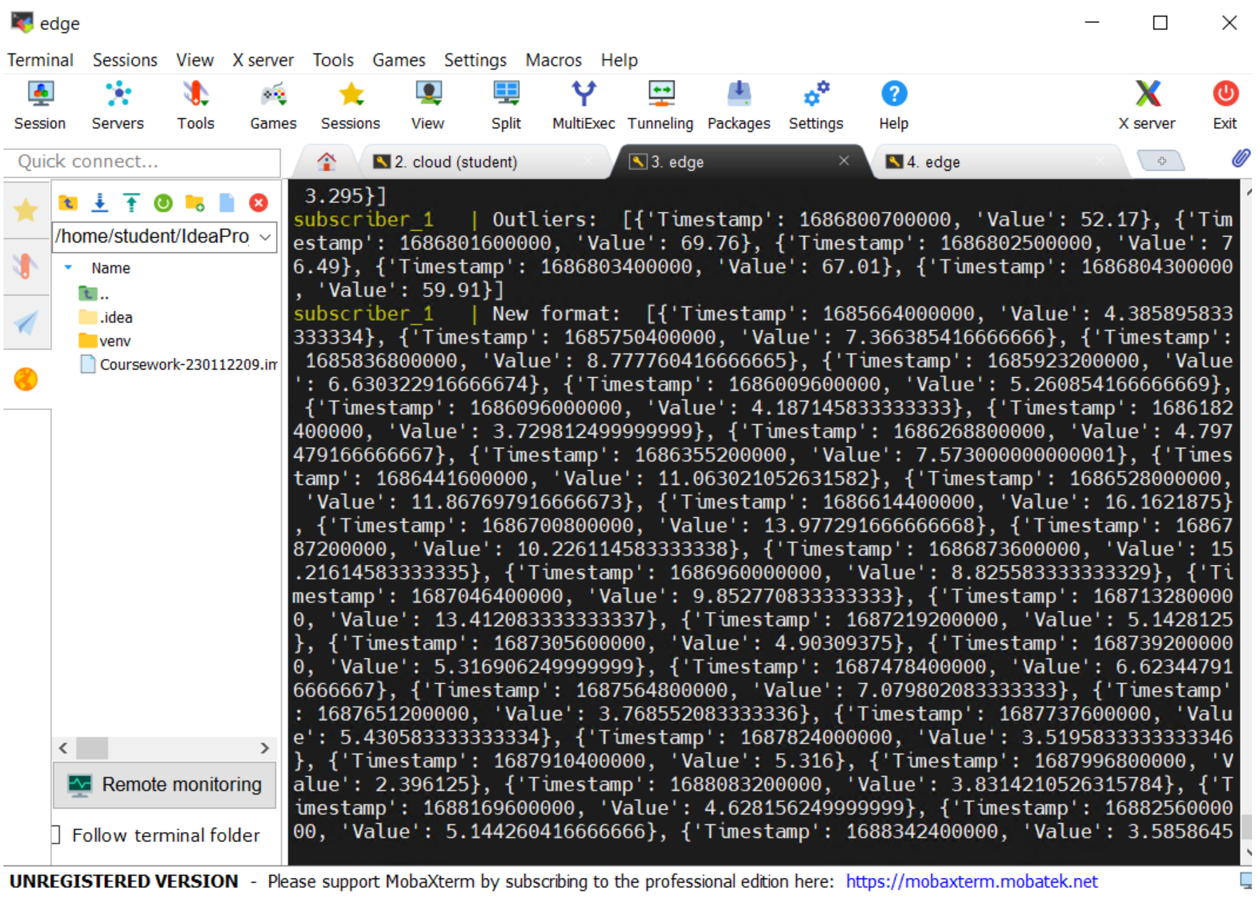
We filter out outliers (the value greater than 50) value from the PM2.5 data obtained and print them to docker logs console.



```
subscriber_1   | Outliers:  [{'Timestamp': 1686800700000, 'Value': 52.17}, {'Tim
estamp': 1686801600000, 'Value': 69.76}, {'Timestamp': 1686802500000, 'Value': 7
6.49}, {'Timestamp': 1686803400000, 'Value': 67.01}, {'Timestamp': 1686804300000
, 'Value': 59.91}]
```

(c) Since the original PM2.5 data readings are collected every 15 mins, so please implement a python code to calculate the averaging value of PM2.5 data on daily basis (every 24 hours), please pick the first start date of a day or a 24 hours interval as the new timestamp of averaged PM2.5 data, and please print out the result to the console (docker logs console).

The Obtained data is then averaged over a 24 hour interval as the new timestamp of PM2.5 data and the output is printed on docker console.



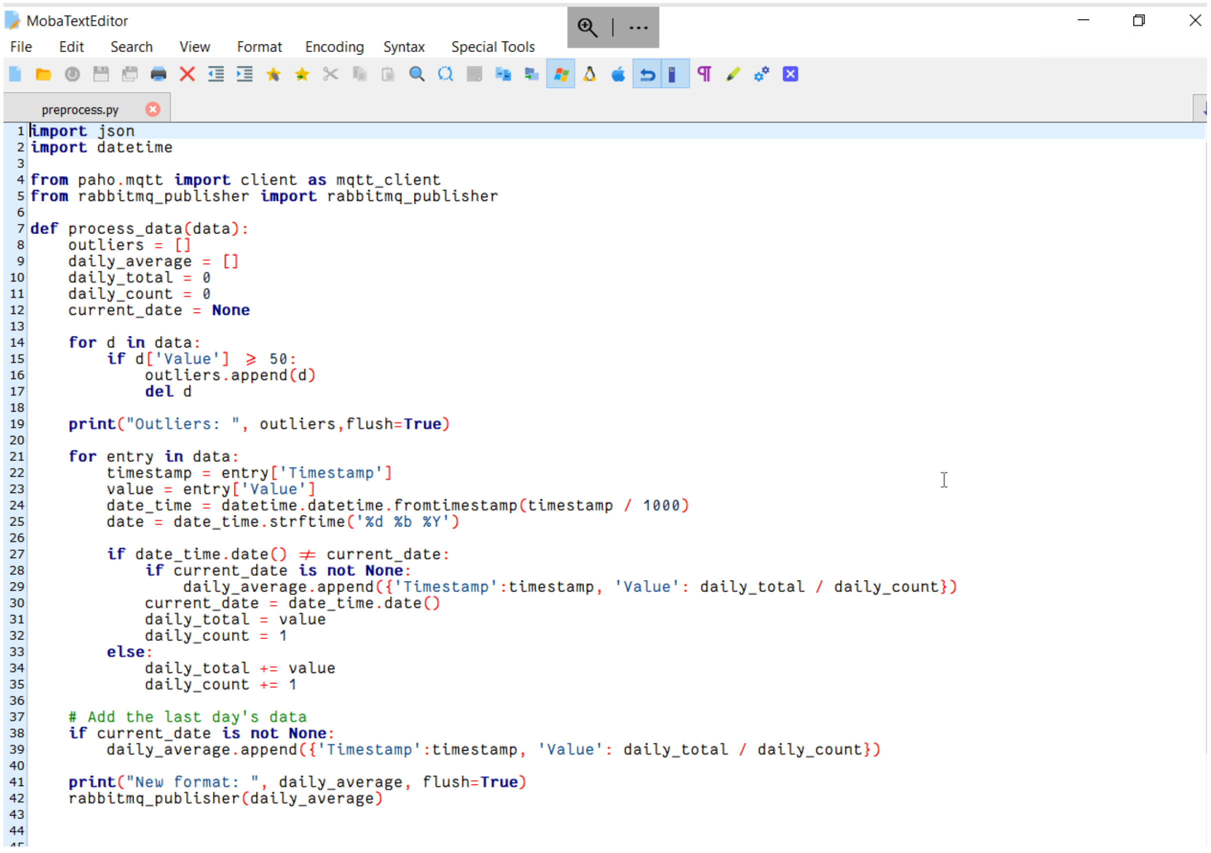
edge
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. cloud (student)
3. edge
4. edge
/home/student/IdeaPro
Name
..
.idea
venv
Coursework-230112209.im
Remote monitoring
Follow terminal folder

```
 3.295}]
subscriber_1   | Outliers:  [{'Timestamp': 1686800700000, 'Value': 52.17}, {'Tim
estamp': 1686801600000, 'Value': 69.76}, {'Timestamp': 1686802500000, 'Value': 7
6.49}, {'Timestamp': 1686803400000, 'Value': 67.01}, {'Timestamp': 1686804300000
, 'Value': 59.91}]
subscriber_1   | New format:  [{'Timestamp': 1685664000000, 'Value': 4.385895833
333334}, {'Timestamp': 1685750400000, 'Value': 7.366385416666666}, {'Timestamp':
 1685836800000, 'Value': 8.777604166666665}, {'Timestamp': 1685923200000, 'Value
': 6.630322916666674}, {'Timestamp': 1686009600000, 'Value': 5.260854166666669},
 {'Timestamp': 1686096000000, 'Value': 4.187145833333333}, {'Timestamp': 1686182
400000, 'Value': 3.729812499999999}, {'Timestamp': 1686268800000, 'Value': 4.797
479166666667}, {'Timestamp': 1686355200000, 'Value': 7.573000000000001}, {'Times
tamp': 1686441600000, 'Value': 11.063021052631582}, {'Timestamp': 1686528000000,
 'Value': 11.867697916666673}, {'Timestamp': 1686614400000, 'Value': 16.1621875}
, {'Timestamp': 1686700800000, 'Value': 13.977291666666668}, {'Timestamp': 16867
87200000, 'Value': 10.226114583333338}, {'Timestamp': 1686873600000, 'Value': 15
.216145833333335}, {'Timestamp': 1686960000000, 'Value': 8.825583333333329}, {'Ti
mestamp': 1687046400000, 'Value': 9.852770833333333}, {'Timestamp': 168713280000
0, 'Value': 13.412083333333337}, {'Timestamp': 1687219200000, 'Value': 5.1428125
}, {'Timestamp': 1687305600000, 'Value': 4.90309375}, {'Timestamp': 168739200000
0, 'Value': 5.316906249999999}, {'Timestamp': 1687478400000, 'Value': 6.62344791
6666667}, {'Timestamp': 1687564800000, 'Value': 7.079802083333333}, {'Timestamp'
: 1687651200000, 'Value': 3.768552083333336}, {'Timestamp': 1687737600000, 'Valu
e': 5.430583333333334}, {'Timestamp': 1687824000000, 'Value': 3.519583333333346
}, {'Timestamp': 1687910400000, 'Value': 5.316}, {'Timestamp': 1687996800000, 'V
alue': 2.396125}, {'Timestamp': 1688083200000, 'Value': 3.8314210526315784}, {'T
imestamp': 1688169600000, 'Value': 4.628156249999999}, {'Timestamp': 16882560000
00, 'Value': 5.144260416666666}, {'Timestamp': 1688342400000, 'Value': 3.5858645
```

**UNREGISTERED VERSION** - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

The below shows pre-processor code written to find outliers and calculate average PM2.5 data which contains process_data function, where data obtained in the previous step is passed to filter out the outliers and then the filtered data is processed to get average daily PM2.5 data. The final averaged Pm2.5 data is passed to rabbitMQ_publisher function to send it over to cloud layer using rabbitmq.

## PREPROCESSOR CODE:

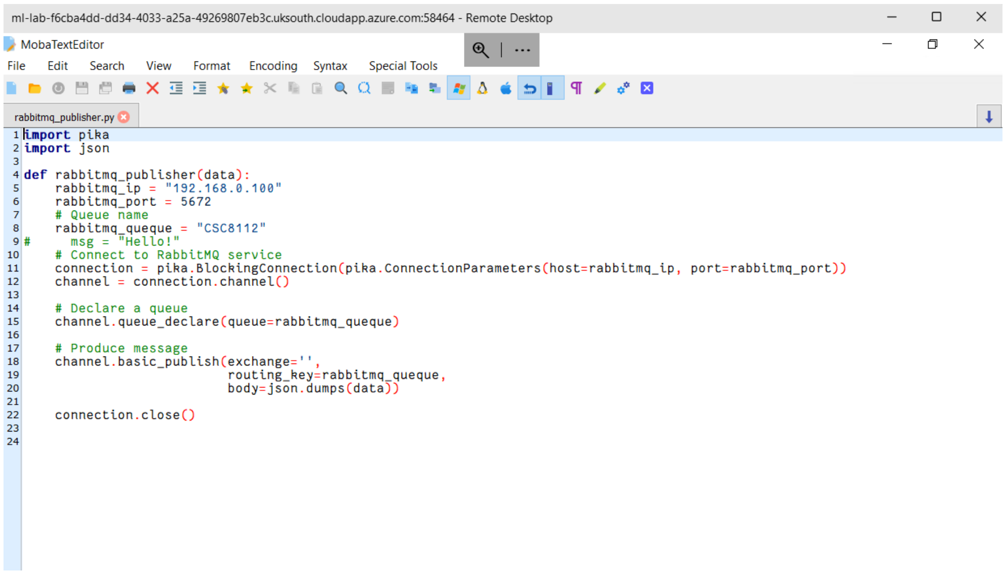


```python
import json
import datetime

from paho.mqtt import client as mqtt_client
from rabbitmq_publisher import rabbitmq_publisher

def process_data(data):
    outliers = []
    daily_average = []
    daily_total = 0
    daily_count = 0
    current_date = None

    for d in data:
        if d['Value'] ≥ 50:
            outliers.append(d)
            del d

    print("Outliers: ", outliers,flush=True)

    for entry in data:
        timestamp = entry['Timestamp']
        value = entry['Value']
        date_time = datetime.datetime.fromtimestamp(timestamp / 1000)
        date = date_time.strftime('%d %b %Y')

        if date_time.date() ≠ current_date:
            if current_date is not None:
                daily_average.append({'Timestamp':timestamp, 'Value': daily_total / daily_count})
            current_date = date_time.date()
            daily_total = value
            daily_count = 1
        else:
            daily_total += value
            daily_count += 1

    # Add the last day's data
    if current_date is not None:
        daily_average.append({'Timestamp':timestamp, 'Value': daily_total / daily_count})

    print("New format: ", daily_average, flush=True)
    rabbitmq_publisher(daily_average)
```

(d) Transfer all results (averaged PM2.5 data) to be used by Task 3.2 (a) into RabbitMQ service on Azure lab (Cloud).

The output obtained from pre-processor code is then sent over to Azure lab (Cloud) through RabbitMQ publisher. The following displays the code crafted to transmit the data from the Azure Edge layer to the cloud.

## RABBITMQ_PUBLISHER CODE:

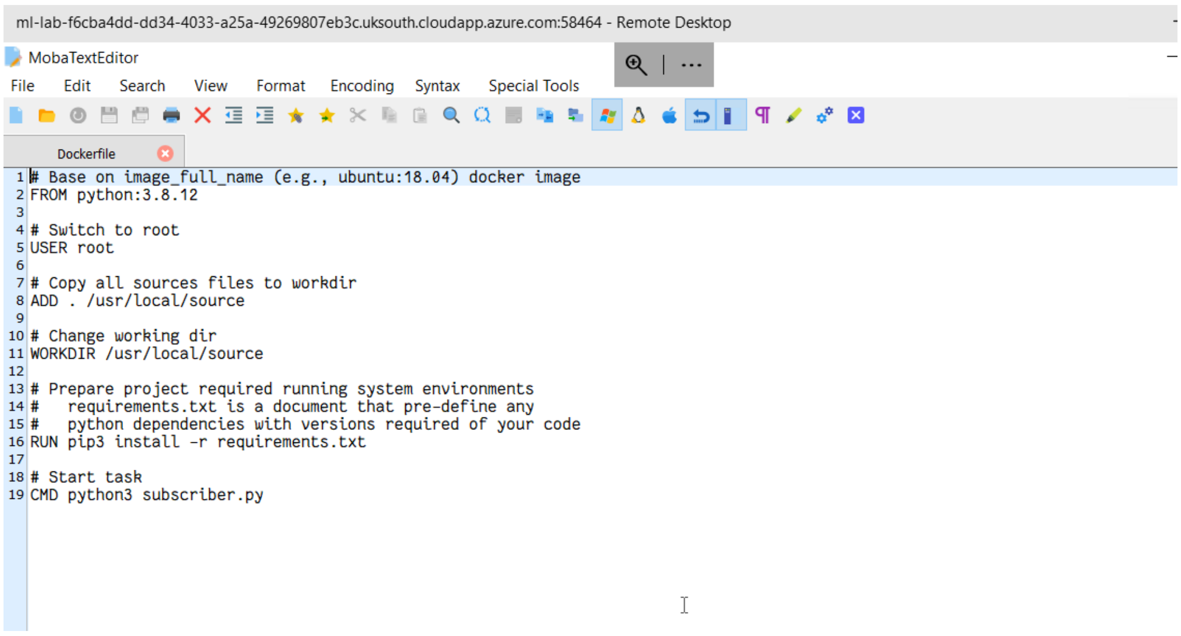

ml-lab-f6cba4dd-dd34-4033-a25a-49269807eb3c.uksouth.cloudapp.azure.com:58464 - Remote Desktop
MobaTextEditor
File Edit Search View Format Encoding Syntax Special Tools
rabbitmq_publisher.py

```python
import pika
import json

def rabbitmq_publisher(data):
    rabbitmq_ip = "192.168.0.100"
    rabbitmq_port = 5672
    # Queue name
    rabbitmq_queque = "CSC8112"
#     msg = "Hello!"
    # Connect to RabbitMQ service
    connection = pika.BlockingConnection(pika.ConnectionParameters(host=rabbitmq_ip, port=rabbitmq_port))
    channel = connection.channel()

    # Declare a queue
    channel.queue_declare(queue=rabbitmq_queque)

    # Produce message
    channel.basic_publish(exchange='',
                          routing_key=rabbitmq_queque,
                          body=json.dumps(data))

    connection.close()
```

3. Define a Dockerfile to migrate your "data preprocessing operator" source code into a Docker image and then define a docker-compose file to run it as a container locally on the Azure lab (Edge).

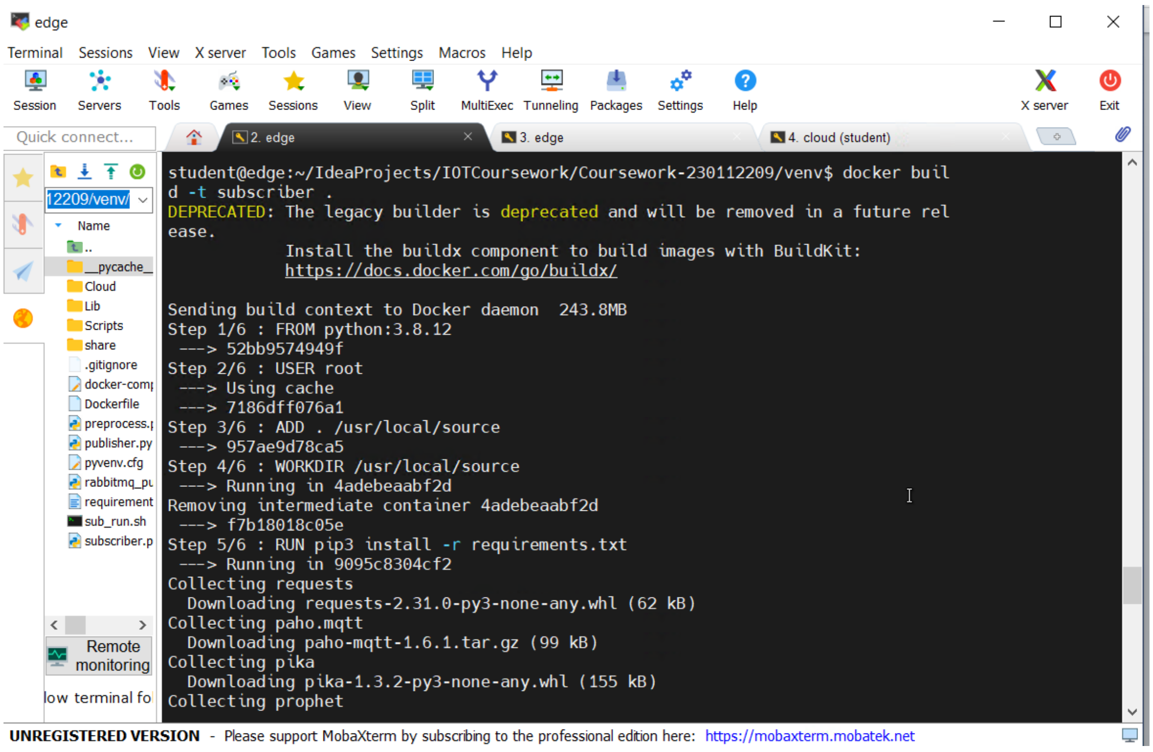
We created a Docker file to install all of our dependencies required to build our Docker image which are specified in "requirements.txt". We build the image using the command "docker build –t subscriber . " . Where subscriber refers to the image name and "." refers to the current directory.

## DOCKERFILE:


ml-lab-f6cba4dd-dd34-4033-a25a-49269807eb3c.uksouth.cloudapp.azure.com:58464 - Remote Desktop
MobaTextEditor
File Edit Search View Format Encoding Syntax Special Tools
Dockerfile

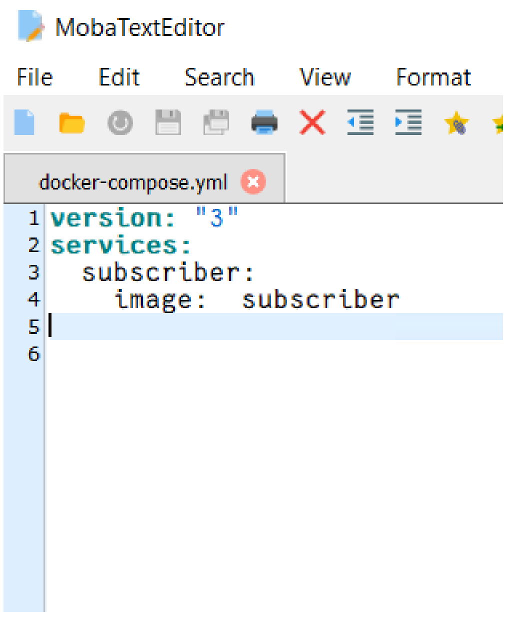
```
# Base on image_full_name (e.g., ubuntu:18.04) docker image
FROM python:3.8.12

# Switch to root
USER root

# Copy all sources files to workdir
ADD . /usr/local/source

# Change working dir
WORKDIR /usr/local/source

# Prepare project required running system environments
#   requirements.txt is a document that pre-define any
#   python dependencies with versions required of your code
RUN pip3 install -r requirements.txt

# Start task
CMD python3 subscriber.py
```

We use the "docker build –t subscriber ." command to build subscriber image.


edge
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. edge
3. edge
4. cloud (student)
12209/venv/
Name
..
__pycache__
Cloud
Lib
Scripts
share
.gitignore
docker-comp
Dockerfile
preprocess.
publisher.py
pyvenv.cfg
rabbitmq_pu
requirement
sub_run.sh
subscriber.p
Remote monitoring
low terminal fo
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

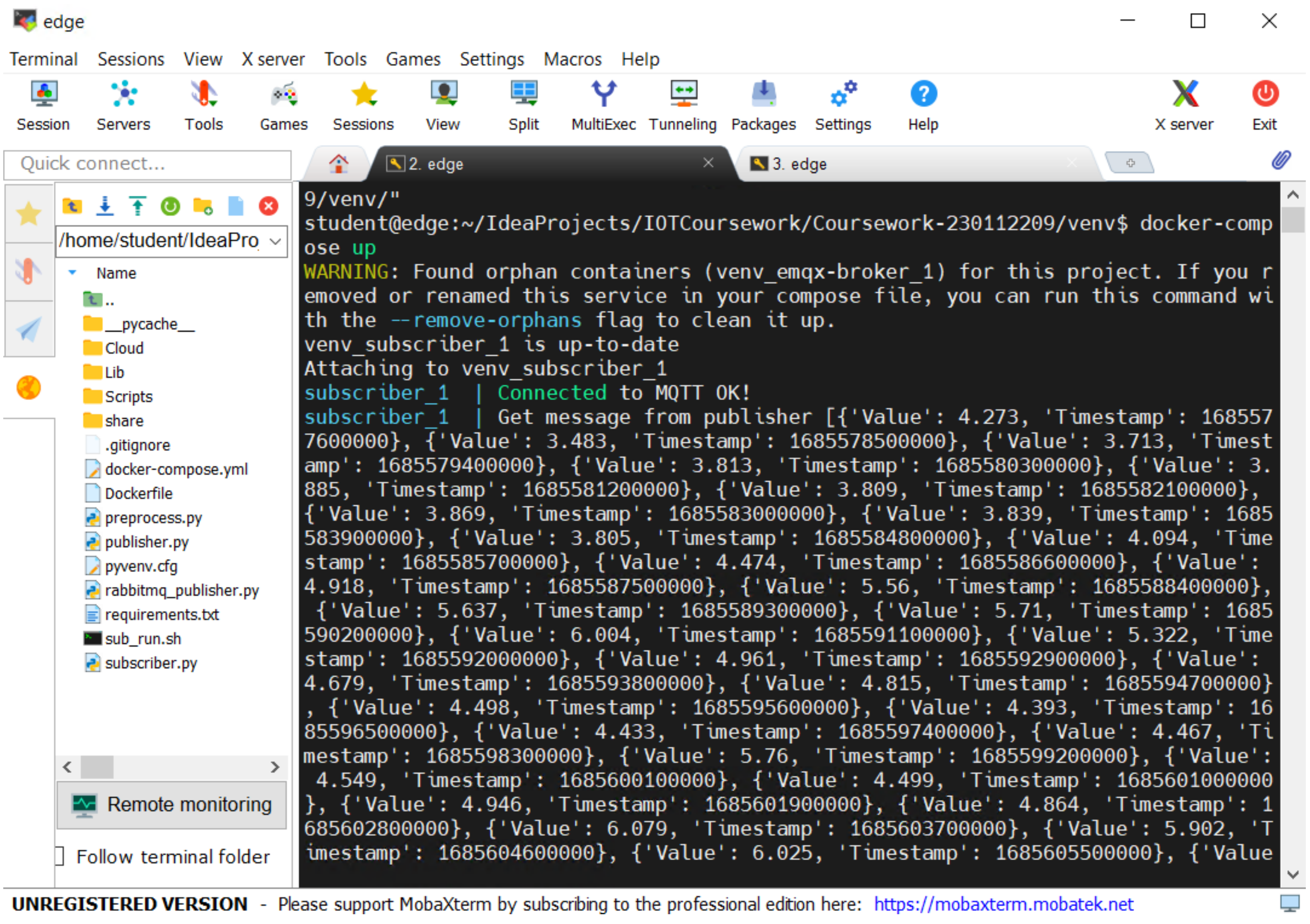
```
student@edge:~/IdeaProjects/IOTCoursework/Coursework-230112209/venv$ docker buil
d -t subscriber .
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  243.8MB
Step 1/6 : FROM python:3.8.12
 ---> 52bb9574949f
Step 2/6 : USER root
 ---> Using cache
 ---> 7186dff076a1
Step 3/6 : ADD . /usr/local/source
 ---> 957ae9d78ca5
Step 4/6 : WORKDIR /usr/local/source
 ---> Running in 4adebeaabf2d
Removing intermediate container 4adebeaabf2d
 ---> f7b18018c05e
Step 5/6 : RUN pip3 install -r requirements.txt
 ---> Running in 9095c8304cf2
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 prophet
```

We created a Docker compose file to run the subscriber image.



```
version: "3"
services:
  subscriber:
    image:  subscriber

```

We use the “docker-compose up” command to run the Docker container locally on the Azure lab (Edge).

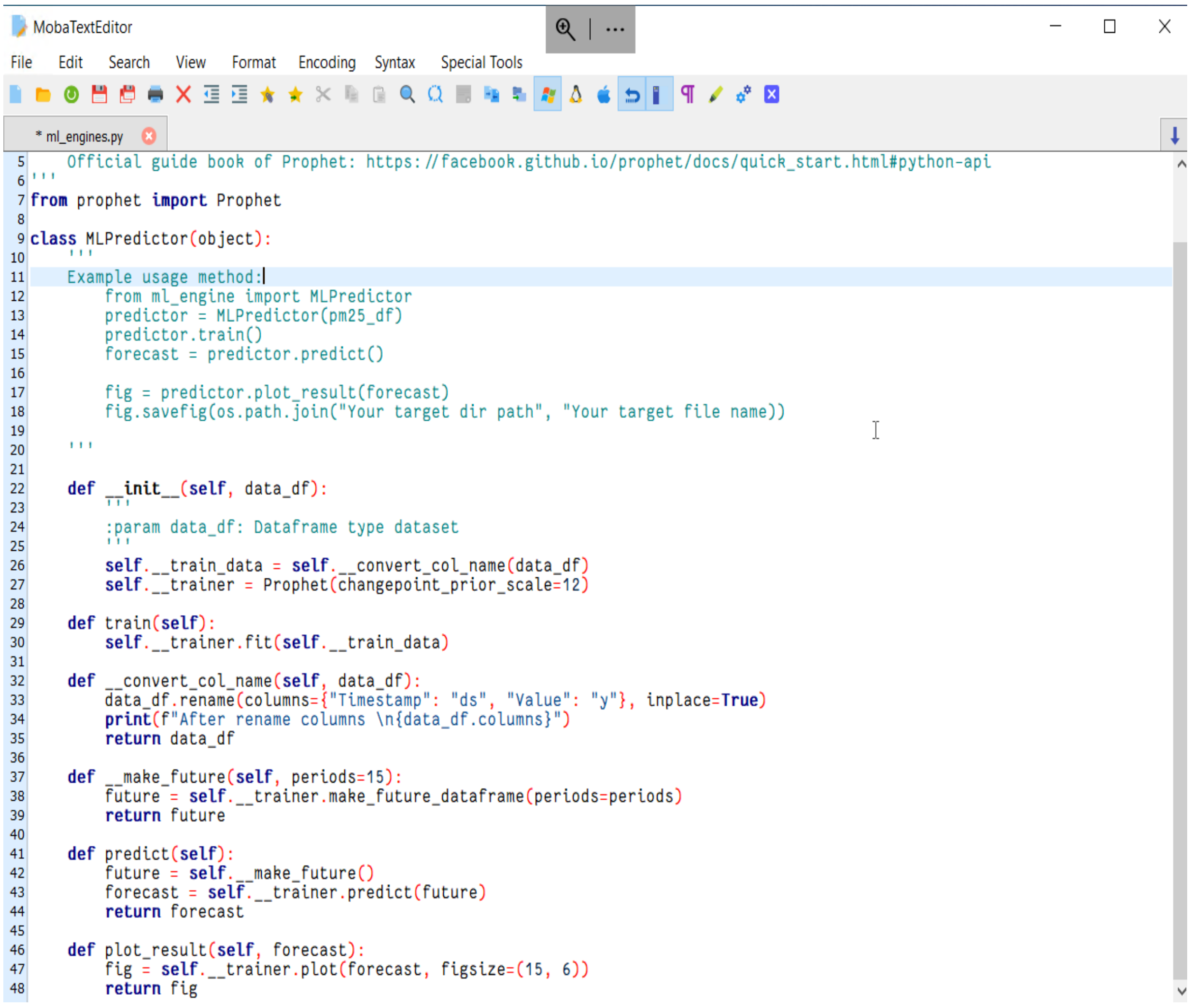

edge
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. edge
3. edge
/home/student/IdeaPro
Name
..
__pycache__
Cloud
Lib
Scripts
share
.gitignore
docker-compose.yml
Dockerfile
preprocess.py
publisher.py
pyvenv.cfg
rabbitmq_publisher.py
requirements.txt
sub_run.sh
subscriber.py
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
9/venv/"
student@edge:~/IdeaProjects/IOTCoursework/Coursework-230112209/venv$ docker-comp
ose up
WARNING: Found orphan containers (venv_emqx-broker_1) for this project. If you r
emoved or renamed this service in your compose file, you can run this command wi
th the --remove-orphans flag to clean it up.
venv_subscriber_1 is up-to-date
Attaching to venv_subscriber_1
subscriber_1  | Connected to MQTT OK!
subscriber_1  | Get message from publisher [{'Value': 4.273, 'Timestamp': 168557
7600000}, {'Value': 3.483, 'Timestamp': 1685578500000}, {'Value': 3.713, 'Timest
amp': 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': 1685
583900000}, {'Value': 3.805, 'Timestamp': 1685584800000}, {'Value': 4.094, 'Time
stamp': 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': 1685
590200000}, {'Value': 6.004, 'Timestamp': 1685591100000}, {'Value': 5.322, 'Time
stamp': 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': 16
85596500000}, {'Value': 4.433, 'Timestamp': 1685597400000}, {'Value': 4.467, 'Ti
mestamp': 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': 1
685602800000}, {'Value': 6.079, 'Timestamp': 1685603700000}, {'Value': 5.902, 'T
imestamp': 1685604600000}, {'Value': 6.025, 'Timestamp': 1685605500000}, {'Value
```

## Task 3: Time-series data prediction and visualization\

1. Download a pre-defined Machine Learning (ML) engine code from [https://github.com/ ncl-iot-team/CSC8112_MLEngine].

   We download and use a pre-defined machine learning code to predict data.

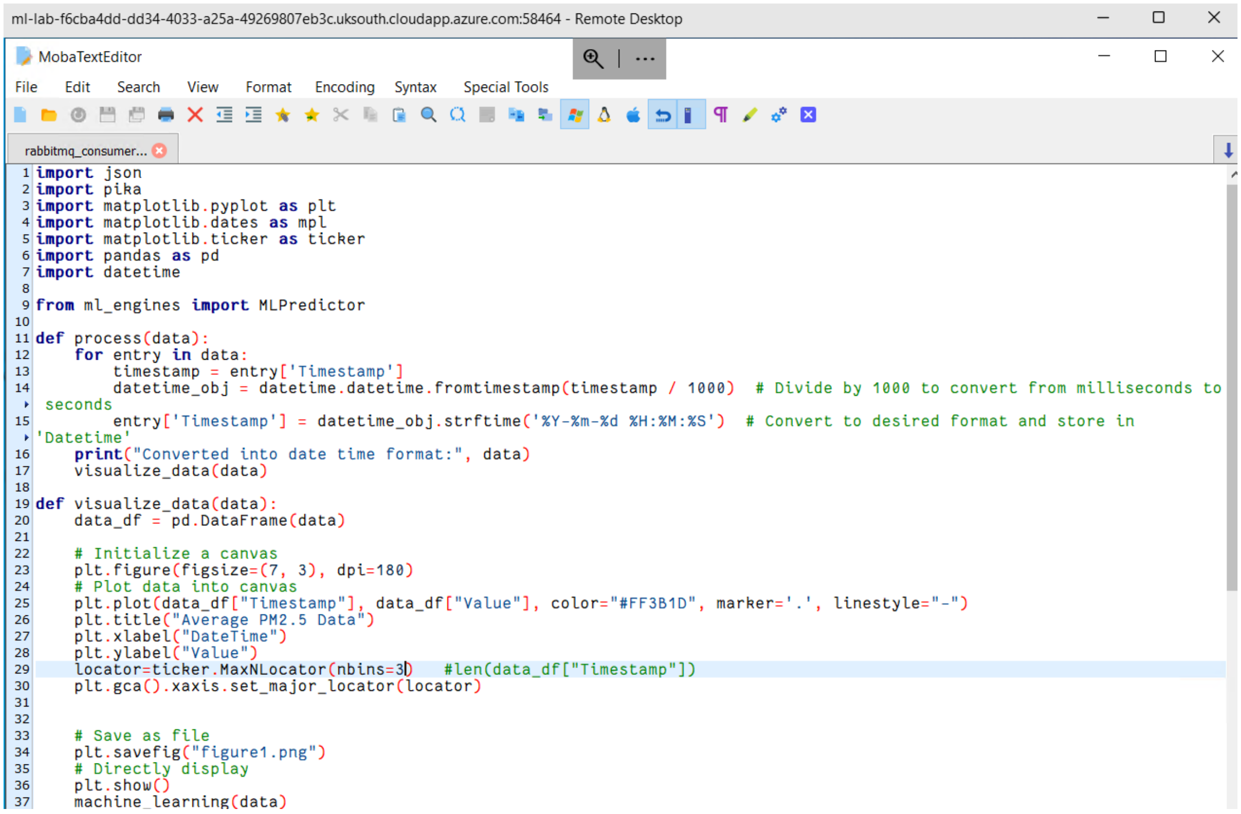


```python
    Official guide book of Prophet: https://facebook.github.io/prophet/docs/quick_start.html#python-api
'''
from prophet import Prophet

class MLPredictor(object):
    '''
    Example usage method:
        from ml_engine import MLPredictor
        predictor = MLPredictor(pm25_df)
        predictor.train()
        forecast = predictor.predict()

        fig = predictor.plot_result(forecast)
        fig.savefig(os.path.join("Your target dir path", "Your target file name))

    '''

    def __init__(self, data_df):
        '''
        :param data_df: Dataframe type dataset
        '''
        self.__train_data = self.__convert_col_name(data_df)
        self.__trainer = Prophet(changepoint_prior_scale=12)

    def train(self):
        self.__trainer.fit(self.__train_data)

    def __convert_col_name(self, data_df):
        data_df.rename(columns={"Timestamp": "ds", "Value": "y"}, inplace=True)
        print(f"After rename columns \n{data_df.columns}")
        return data_df

    def __make_future(self, periods=15):
        future = self.__trainer.make_future_dataframe(periods=periods)
        return future

    def predict(self):
        future = self.__make_future()
        forecast = self.__trainer.predict(future)
        return forecast

    def plot_result(self, forecast):
        fig = self.__trainer.plot(forecast, figsize=(15, 6))
        return fig
```

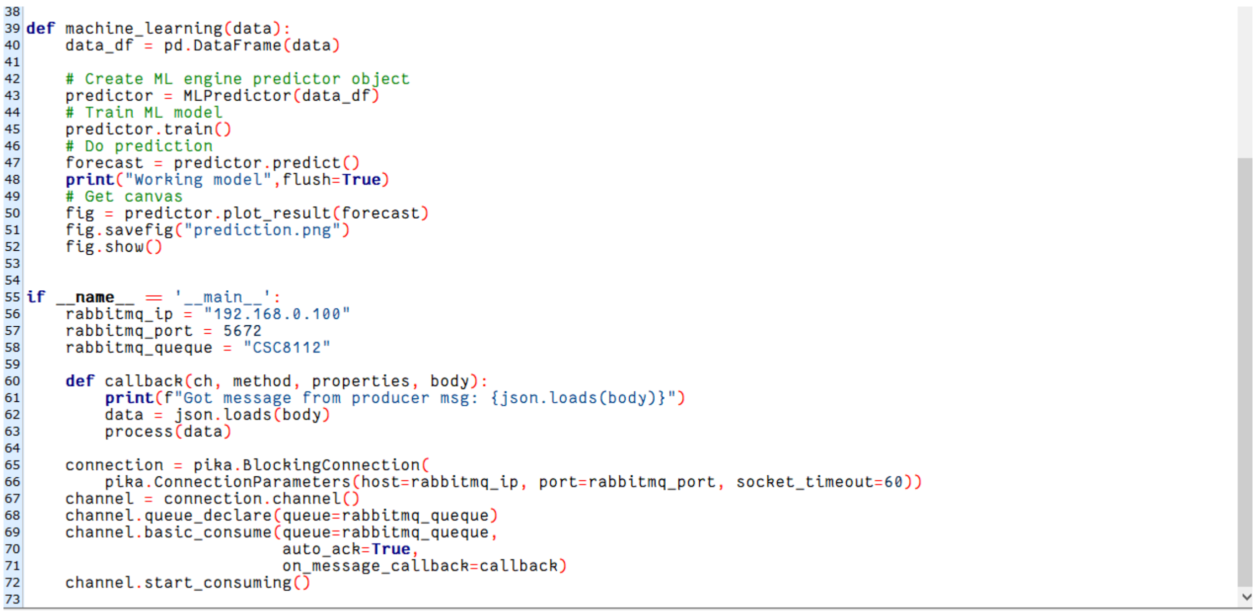
2. Design a PM2.5 prediction operator with the following functions (code) in Azure Lab (Cloud)

We have implemented a function to receive the code sent through RabbitMQ. This function then channels the data to the processing function, where it undergoes conversion into a date-time format. Subsequently, the processed data is transferred to the visualize data function for a graphical representation of the given dataset. In the final step, the data is forwarded to a machine learning model to generate predictions and facilitate visualizations.

## RABBITMQ_CONSUMER:


ml-lab-f6cba4dd-dd34-4033-a25a-49269807eb3c.uksouth.cloudapp.azure.com:58464 - Remote Desktop
MobaTextEditor
File
Edit
Search
View
Format
Encoding
Syntax
Special Tools
rabbitmq_consumer...

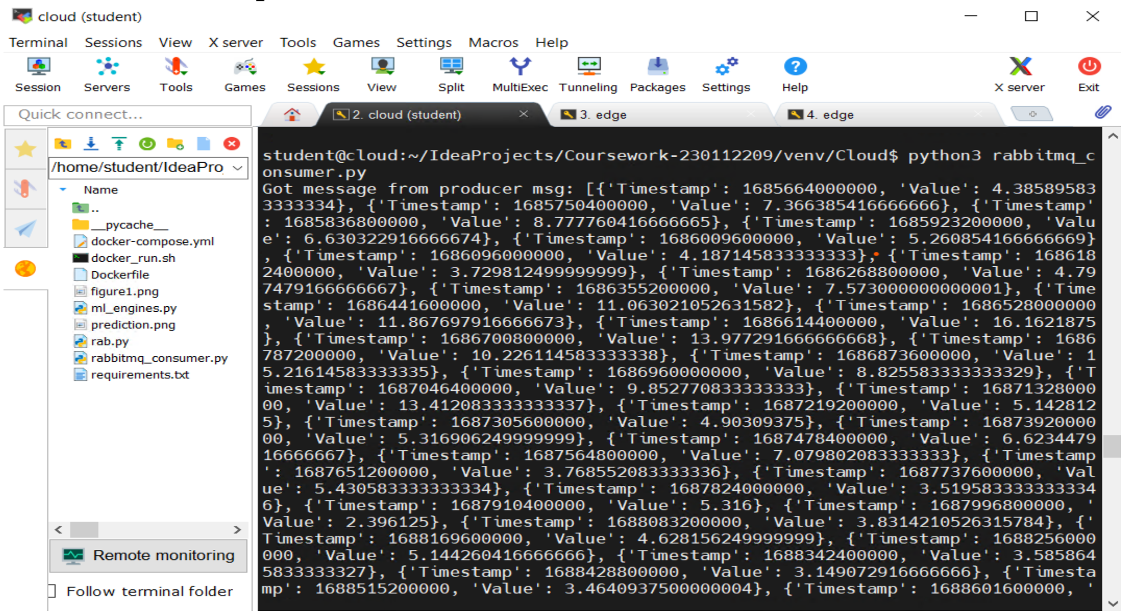
```python
import json
import pika
import matplotlib.pyplot as plt
import matplotlib.dates as mpl
import matplotlib.ticker as ticker
import pandas as pd
import datetime

from ml_engines import MLPredictor

def process(data):
    for entry in data:
        timestamp = entry['Timestamp']
        datetime_obj = datetime.datetime.fromtimestamp(timestamp / 1000)  # Divide by 1000 to convert from milliseconds to seconds
        entry['Timestamp'] = datetime_obj.strftime('%Y-%m-%d %H:%M:%S')  # Convert to desired format and store in 'Datetime'
    print("Converted into date time format:", data)
    visualize_data(data)

def visualize_data(data):
    data_df = pd.DataFrame(data)

    # Initialize a canvas
    plt.figure(figsize=(7, 3), dpi=180)
    # Plot data into canvas
    plt.plot(data_df["Timestamp"], data_df["Value"], color="#FF3B1D", marker='.', linestyle="-")
    plt.title("Average PM2.5 Data")
    plt.xlabel("DateTime")
    plt.ylabel("Value")
    locator=ticker.MaxNLocator(nbins=3)    #len(data_df["Timestamp"])
    plt.gca().xaxis.set_major_locator(locator)


    # Save as file
    plt.savefig("figure1.png")
    # Directly display
    plt.show()
    machine_learning(data)
```



```python

def machine_learning(data):
    data_df = pd.DataFrame(data)

    # Create ML engine predictor object
    predictor = MLPredictor(data_df)
    # Train ML model
    predictor.train()
    # Do prediction
    forecast = predictor.predict()
    print("Working model",flush=True)
    # Get canvas
    fig = predictor.plot_result(forecast)
    fig.savefig("prediction.png")
    fig.show()


if __name__ == '__main__':
    rabbitmq_ip = "192.168.0.100"
    rabbitmq_port = 5672
    rabbitmq_queque = "CSC8112"

    def callback(ch, method, properties, body):
        print(f"Got message from producer msg: {json.loads(body)}")
        data = json.loads(body)
        process(data)

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host=rabbitmq_ip, port=rabbitmq_port, socket_timeout=60))
    channel = connection.channel()
    channel.queue_declare(queue=rabbitmq_queque)
    channel.basic_consume(queue=rabbitmq_queque,
                          auto_ack=True,
                          on_message_callback=callback)
    channel.start_consuming()

```

(a) Collect all averaged daily PM2.5 data computed by Task 2.2 (d) from RabbitMQ service, and please print out them to the console.

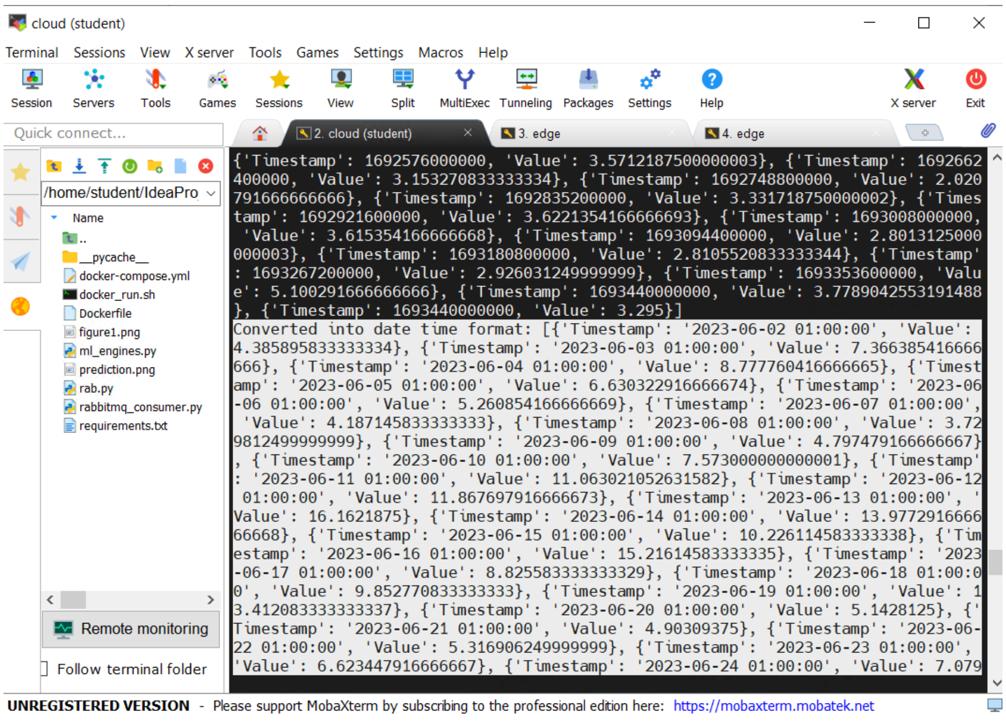
We run rabbitmq_consumer.py code which collects all the data published by RabbitMQ service and then prints out the result to console.



```
student@cloud:~/IdeaProjects/Coursework-230112209/venv/Cloud$ python3 rabbitmq_c
onsumer.py
Got message from producer msg: [{'Timestamp': 1685664000000, 'Value': 4.38589583
3333334}, {'Timestamp': 1685750400000, 'Value': 7.366385416666666}, {'Timestamp'
: 1685836800000, 'Value': 8.777760416666665}, {'Timestamp': 1685923200000, 'Valu
e': 6.630322916666674}, {'Timestamp': 1686009600000, 'Value': 5.260854166666669}
, {'Timestamp': 1686096000000, 'Value': 4.187145833333333}, {'Timestamp': 168618
2400000, 'Value': 3.729812499999999}, {'Timestamp': 1686268800000, 'Value': 4.79
7479166666667}, {'Timestamp': 1686355200000, 'Value': 7.573000000000001}, {'Time
stamp': 1686441600000, 'Value': 11.063021052631582}, {'Timestamp': 1686528000000
, 'Value': 11.867697916666673}, {'Timestamp': 1686614400000, 'Value': 16.1621875
}, {'Timestamp': 1686700800000, 'Value': 13.977291666666668}, {'Timestamp': 1686
787200000, 'Value': 10.226114583333338}, {'Timestamp': 1686873600000, 'Value': 1
5.216145833333335}, {'Timestamp': 1686960000000, 'Value': 8.825583333333329}, {'T
imestamp': 1687046400000, 'Value': 9.852770833333333}, {'Timestamp': 16871328000
00, 'Value': 13.412083333333337}, {'Timestamp': 1687219200000, 'Value': 5.142812
5}, {'Timestamp': 1687305600000, 'Value': 4.90309375}, {'Timestamp': 16873920000
00, 'Value': 5.316906249999999}, {'Timestamp': 1687478400000, 'Value': 6.6234479
16666667}, {'Timestamp': 1687564800000, 'Value': 7.079802083333333}, {'Timestamp
': 1687651200000, 'Value': 3.768552083333336}, {'Timestamp': 1687737600000, 'Val
ue': 5.430583333333334}, {'Timestamp': 1687824000000, 'Value': 3.519583333333334
6}, {'Timestamp': 1687910400000, 'Value': 5.316}, {'Timestamp': 1687996800000, '
Value': 2.396125}, {'Timestamp': 1688083200000, 'Value': 3.831421052631578 4}, {'
Timestamp': 1688169600000, 'Value': 4.628156249999999}, {'Timestamp': 1688256000
000, 'Value': 5.144260416666666}, {'Timestamp': 1688342400000, 'Value': 3.585864
5833333327}, {'Timestamp': 1688428800000, 'Value': 3.149072916666666}, {'Timesta
mp': 1688515200000, 'Value': 3.4640937500000004}, {'Timestamp': 1688601600000, '
```

(b) Convert timestamp to date time format (year-month-day hour: minute: second), and please print out the PM2.5 data with the reformatted timestamp to the console.

We convert the obtained into date time format by passing it to the process function as specified in the above code.

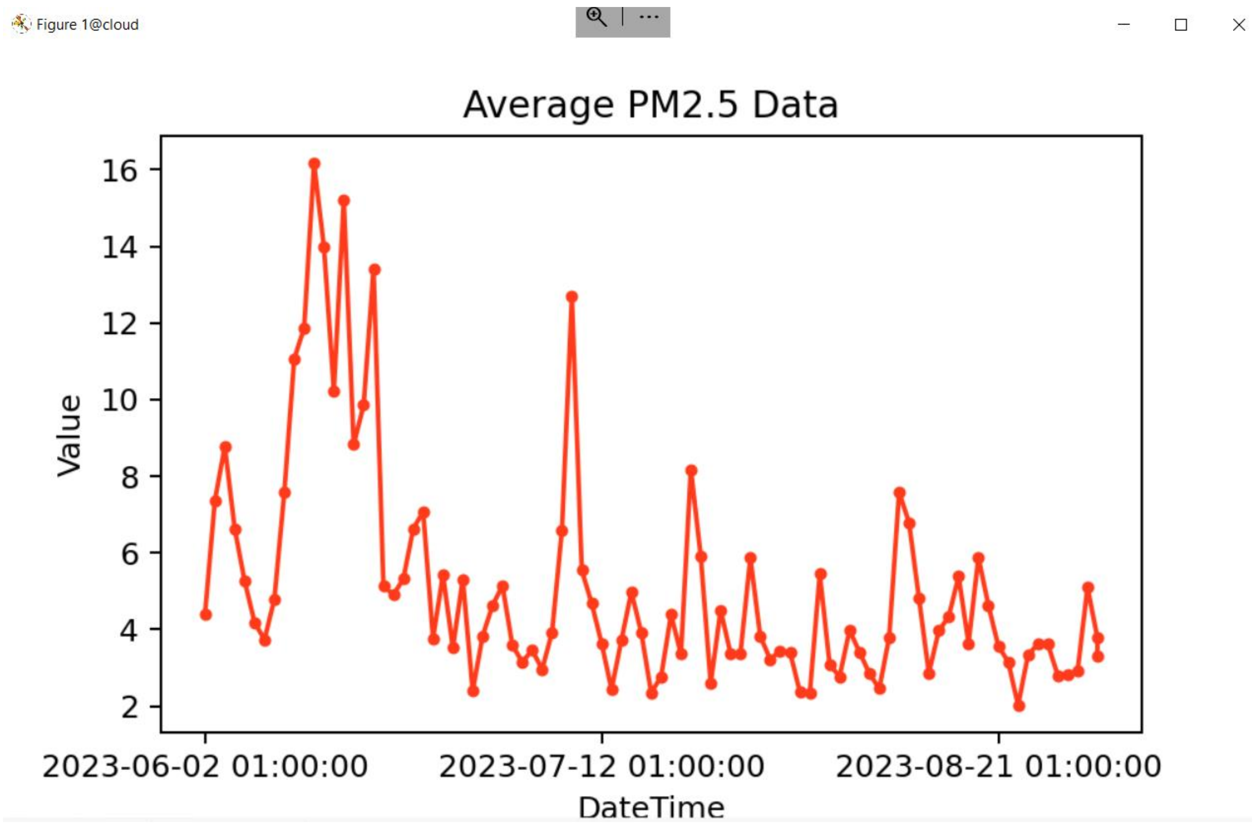


```
{'Timestamp': 1692576000000, 'Value': 3.5712187500000003}, {'Timestamp': 1692662
400000, 'Value': 3.153270833333334}, {'Timestamp': 1692748800000, 'Value': 2.020
791666666666}, {'Timestamp': 1692835200000, 'Value': 3.3317187500000002}, {'Times
tamp': 1692921600000, 'Value': 3.6221354166666693}, {'Timestamp': 1693008000000,
 'Value': 3.615354166666668}, {'Timestamp': 1693094400000, 'Value': 2.8013125000
000003}, {'Timestamp': 1693180800000, 'Value': 2.810552083333344}, {'Timestamp'
: 1693267200000, 'Value': 2.926031249999999}, {'Timestamp': 1693353600000, 'Valu
e': 5.100291666666666}, {'Timestamp': 1693440000000, 'Value': 3.7789042553191488
}, {'Timestamp': 1693440000000, 'Value': 3.295}]
Converted into date time format: [{'Timestamp': '2023-06-02 01:00:00', 'Value':
4.385895833333334}, {'Timestamp': '2023-06-03 01:00:00', 'Value': 7.366385416666
666}, {'Timestamp': '2023-06-04 01:00:00', 'Value': 8.777760416666665}, {'Timest
amp': '2023-06-05 01:00:00', 'Value': 6.630322916666674}, {'Timestamp': '2023-06
-06 01:00:00', 'Value': 5.260854166666669}, {'Timestamp': '2023-06-07 01:00:00',
 'Value': 4.187145833333333}, {'Timestamp': '2023-06-08 01:00:00', 'Value': 3.72
9812499999999}, {'Timestamp': '2023-06-09 01:00:00', 'Value': 4.797479166666667}
, {'Timestamp': '2023-06-10 01:00:00', 'Value': 7.573000000000001}, {'Timestamp'
: '2023-06-11 01:00:00', 'Value': 11.063021052631582}, {'Timestamp': '2023-06-12
 01:00:00', 'Value': 11.867697916666673}, {'Timestamp': '2023-06-13 01:00:00', '
Value': 16.1621875}, {'Timestamp': '2023-06-14 01:00:00', 'Value': 13.9772916666
66668}, {'Timestamp': '2023-06-15 01:00:00', 'Value': 10.226114583333338}, {'Tim
estamp': '2023-06-16 01:00:00', 'Value': 15.216145833333335}, {'Timestamp': '2023
-06-17 01:00:00', 'Value': 8.825583333333329}, {'Timestamp': '2023-06-18 01:00:0
0', 'Value': 9.852770833333333}, {'Timestamp': '2023-06-19 01:00:00', 'Value': 1
3.412083333333337}, {'Timestamp': '2023-06-20 01:00:00', 'Value': 5.1428125}, {'
Timestamp': '2023-06-21 01:00:00', 'Value': 4.90309375}, {'Timestamp': '2023-06-
22 01:00:00', 'Value': 5.316906249999999}, {'Timestamp': '2023-06-23 01:00:00',
'Value': 6.623447916666667}, {'Timestamp': '2023-06-24 01:00:00', 'Value': 7.079
```

**UNREGISTERED VERSION** - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

(c) Use the line chart component of matplotlib to visualize averaged PM2.5 daily data, directly display the figure or save it as a file.

We pass the obtained averaged PM2.5 daily data into visualize_data function to create a visual representation using matplotlib.

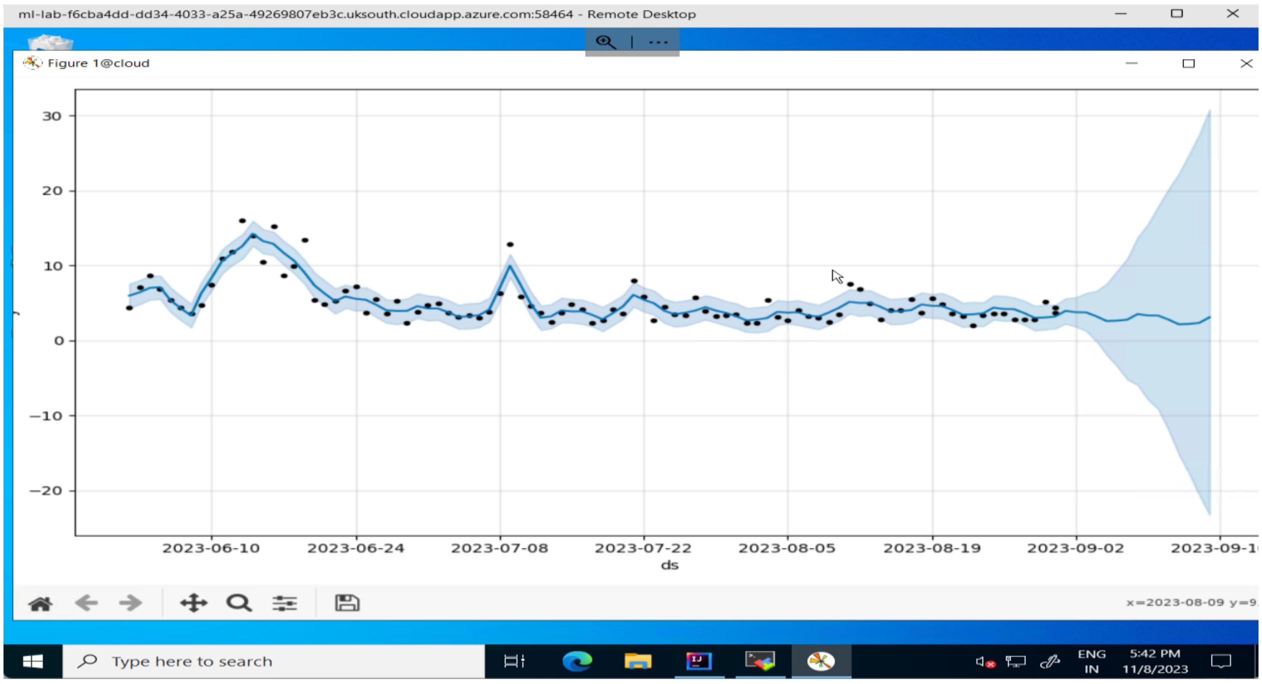


(d) Feed averaged PM2.5 data to machine learning model to predict the trend of PM2.5 for the next 15 days (this predicted time period is a default setting of provided machine learning predictor/classifier model).

We feed the average daily PM2.5 data into the provided machine learning model (provided above in task3 part 1) which helps us predict the PM2.5 data for the next 15 days.

(e) Visualize predicted results from Machine Learning predictor/classifier model, directly display the figure or save as it a file (pre-defined in the provided Machine Learning code)

After implementing the machine learning model we are able to visualize the predicted PM2.5data for the next 15 days.



## Analytical Discussion:

In the given project we have obtained some insight about the PM2.5 data obtained over a period of time and also have got a predicted value of the same data which shows the decrease in its value over the period of next 15 days. The PM2.5 data graph shows that the value of PM2.5 was maximum in the month of June and had a decline over time from June till November.

## Conclusion:

In this project we were able to implement an IOT pipeline using two different message (EMQX and RABBITMQ) queuing systems operating in two different layers i.e. Azure Edge and Azure Cloud layer. This project helped me to understand the use of IOT principles using various tools such as Docker to containerize the code into lightweight Docker file and run the same on the edge layer to reduce computing. The project has equipped with knowledge pertaining to various modules and techniques to implement a basic IOT pipeline.

## REFERENCES:

1) https://stackoverflow.com/ : For various doubts related to implementation.
2) https://docker-curriculum.com/ : Docker documentation.