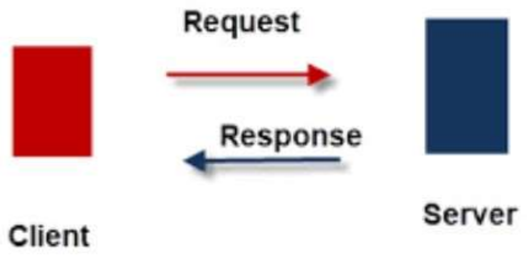
CA - Assignment 3

Aim: **Implement Edge to cloud Protocols (Minimum 3) using a dummy data set.**

Theory:-

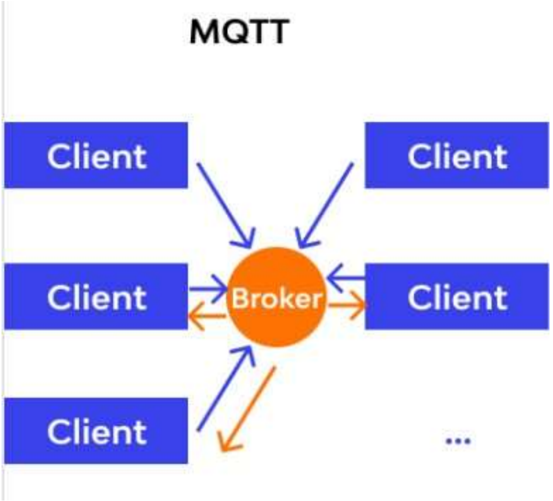
HTTP, MQTT, and WebSocket are three different communication protocols commonly used invarious applications and scenarios.

1. **HTTP (Hypertext Transfer Protocol):**
 - Purpose: HTTP is primarily used for transmitting data between a client (usually a web browser) and a web server. It's the foundation of data communication on the World Wide Web.
 - Communication Style: It follows a request-response model, where a client sends an HTTP request to a server, and the server responds with the requested data or an error message.
 - Use Cases: HTTP is used for web browsing, web services (RESTful APIs), fetching web pages, sending form data, and more. It's a text-based protocol and typically runs over TCP on port 80 (HTTP) or port 443 (HTTPS).



HTTP Protocol Basics

2. **MQTT (Message Queuing Telemetry Transport):**
 - Purpose: MQTT is a lightweight, publish-subscribe messaging protocol designed for constrained devices and low-bandwidth, high-latency, or unreliable networks, such as IoT (Internet of Things) applications.
 - Communication Style: MQTT uses a publish-subscribe model, where clients (publishers) send messages to topics, and other clients (subscribers) receive messages from subscribed topics. It operates on a client-broker architecture.
 - Use Cases: MQTT is commonly used in IoT, home automation, and sensor networks. It's efficient for real-time data streaming, as it minimizes bandwidth usage and supports Quality of Service (QoS) levels for message reliability.

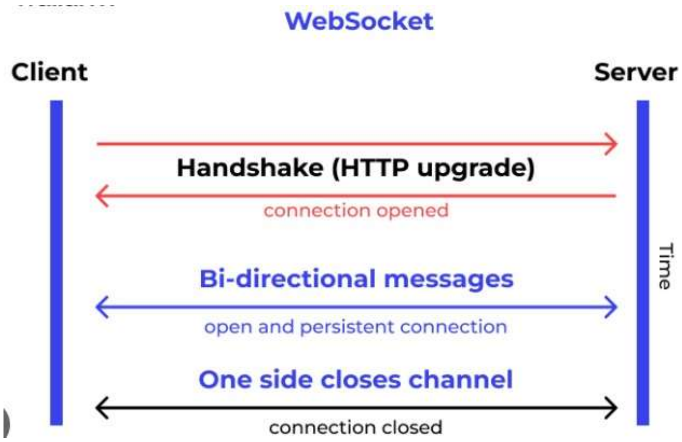


3. WebSocket:

Purpose: WebSocket is a communication protocol that provides full-duplex, bidirectional communication over a single, long-lived connection. It's used for interactive and real-time applications.

Communication Style: WebSocket allows both the client and server to send data to each other at any time without the overhead of traditional HTTP requests. It's initiated with an initial handshake over HTTP and then upgraded to WebSocket.

Use Cases: WebSocket is used in real-time chat applications, online gaming, financial trading platforms, and any application requiring low-latency, bidirectional communication. It's well-suited for interactive and collaborative web applications.



- HTTP is a versatile protocol for general web communication and APIs.
- MQTT is optimized for lightweight, real-time, and publish-subscribe messaging.
- WebSocket provides full-duplex, low-latency communication, making it suitable for interactive, real-time, and collaborative web applications.

Code :

Flask application on local machine (edge application)

1. http_edge.py

```
import requests
import random
import time

while True:
    dummy_data = random.randint(0, 100)
    print("Sending")
    response = requests.post("http://65.2.69.125:8080/receive-data",
   json={"data": dummy_data})
    print(f"Sent: {dummy_data}")
 time.sleep(1)
```

2. mqtt_edge.py

```
 import random
import time
import json

from paho.mqtt import client as mqtt_client

broker = 'broker.emqx.io'
port = 1883
topic = "python/mqtt"
client_id = f'publish-{random.randint(0, 1000)}'

def connect_mqtt():
   def on_connect(client, userdata, flags, rc):
     if rc == 0:
        print("Connected to MQTT Broker!")
     else:
         print(f"Failed to connect, return code {rc}")

     client = mqtt_client.Client(client_id)
    client.on_connect = on_connect

   client.connect(broker, port)
   return client
   def generate_random_data():
   # Modify this function to generate your random data in the desired format.
   # For example, you can generate a dictionary with random values.
   data = {
```

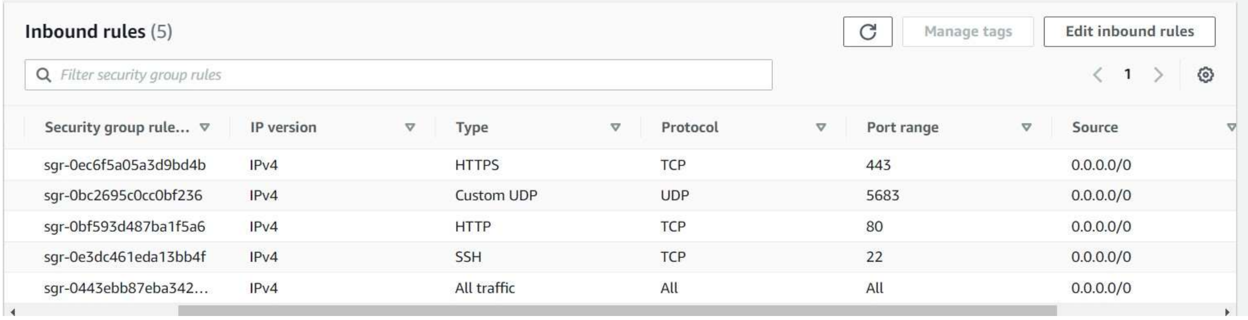
```
   "temperature": random.uniform(20.0, 30.0),
   "humidity": random.uniform(40.0, 60.0)
     }
    return json.dumps(data)

   def publish_random_data(client):
      msg_count = 1
      while True:
          time.sleep(1)
          random_data = generate_random_data()
          result = client.publish(topic, random_data) status = result.rc
   if status == mqtt_client.MQTT_ERR_SUCCESS: print(f"Send `{random_data}` to topic `{topic}`")
   else:
   print(f"Failed to send message to topic {topic}") msg_count += 1
    if msg_count > 5:
         break

   def run():
        client = connect_mqtt()
        client.loop_start()
        publish_random_data(client)
        client.loop_stop()

if name    == ' main ':
  run()
```

3. websocket_edge.py

```
import asyncio
import websockets
import random
import time

async def send_data():
      async with websockets.connect('ws://15.207.116.226:8080') as
 websocket:

    while True:
          dummy_data = random.randint(0, 100)
          await websocket.send(str(dummy_data))
          print(f"Sent: {dummy_data}")
          await asyncio.sleep(1)

if name == ' main ':
    asyncio.get_event_loop().run_until_complete(send_data()
```

Flask application on cloud (AWS EC2)

step 1 :- First create one EC2 instance with inbound security



Inbound rules (5)

Security group rule...	IP version	Type	Protocol	Port range	Source
sgr-0ec6f5a05a3d9bd4b	IPv4	HTTPS	TCP	443	0.0.0.0/0
sgr-0bc2695c0cc0bf236	IPv4	Custom UDP	UDP	5683	0.0.0.0/0
sgr-0bf593d487ba1f5a6	IPv4	HTTP	TCP	80	0.0.0.0/0
sgr-0e3dc461eda13bb4f	IPv4	SSH	TCP	22	0.0.0.0/0
sgr-0443ebb87eba342...	IPv4	All traffic	All	All	0.0.0.0/0

step 2:- Connect to ec2 then sudo su

step 3 :- Then update the ubuntu sudo apt-get update

step 4:- Then install python in ec2 sudo apt-get install python3-venv

step 5:- Now, activate new virtual environment in new directory

step 6 :- create directory mkdir ioe cd ioe

step 7:- Create virtual environment python3 -m venv venv

step 8 :- Activate the virtual environment source venv/bin/activate

step 9 :- install pip install Flask

step 10:- Create a simple flask api sudo nano http_cloud.py

Then copy paste below code and press Ctrl+X then click “y” button then press enter.

Code:-

```
from flask
import Flask, request app = Flask(_name_)
@app.route('/receive-data', methods=['POST']) def receive_data():
data = request.get_json() print(f"Received data: {data['data']}") return "Data received"

if _name_ == '_main_':
app.run(host='0.0.0.0', port=8080)  # Listen on port 80 for HTTP
```

Step 11:- Then run the application by command : python http_cloud.py

then start your local flask application making http request, (above http_edge.py)

Now, let setup mqtt_cloud: sudo nano mqtt_cloud.py

Step 12 Then copy paste below code and press Ctrl+X then click “y” button then press enter.

Code:-

```
import random

from paho.mqtt import client as mqtt_client

broker = 'broker.emqx.io'
port = 1883
topic = "python/mqtt"
# Generate a Client ID with the subscribe prefix.
client_id = f'subscribe-{random.randint(0, 100)}'
# username = 'emqx'
# password = 'public'

def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
      if rc == 0:
        print("Connected to MQTT Broker!")
      else:
        print("Failed to connect, return code %d\n", rc)

client = mqtt_client.Client(client_id)
# client.username_pw_set(username, password)
client.on_connect = on_connect
client.connect(broker, port)
return client

def subscribe(client: mqtt_client):
def on_message(client, userdata, msg):
print(f"Received `{msg.payload.decode()}` from `{msg.topic}`
topic")

client.subscribe(topic) client.on_message = on_message

def run():
```

```
client = connect_mqtt() subscribe(client)
client.loop_forever()

if _name_ == '_main_': run()
```

step 13:- Then run the application by command : `python mqtt_cloud.py`

Then start your local flask application making http request, (above mqtt_edge.py)

Now, let setup websocket_cloud: Create a simple flask api

`sudo nano websocket_cloud.py`

Then copy paste below code and press Ctrl+X then click “y” button then press enter.

Code:-

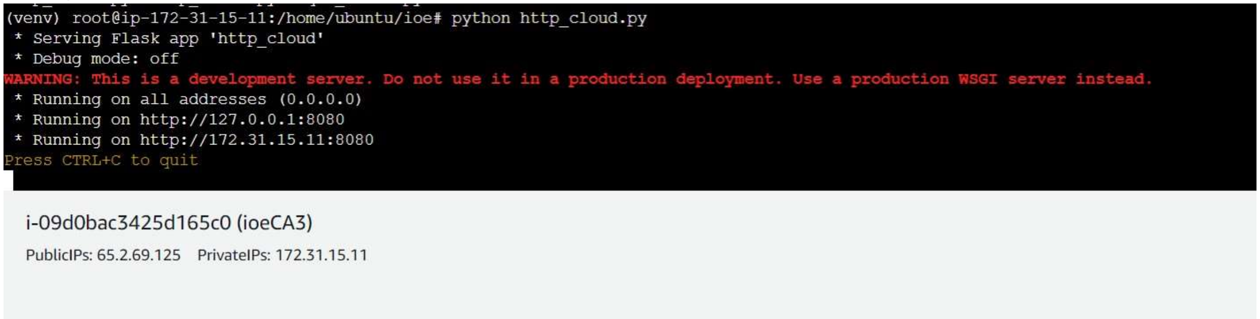
```
import asyncio
import websockets

async def receive_data(websocket, path):
    async for message in websocket:
        print(f"Received: {message}")

if name == ' main  ':
   start_server = websockets.serve(receive_data, '0.0.0.0', 8080)

   asyncio.get_event_loop().run_until_complete(start_server)
   asyncio.get_event_loop().run_forever()
```

step 14:- Then run the application by command : `python websocket_cloud.py`

step 15 :- Then start your local flask application making http request, (above mqtt_edge.py)

Output :

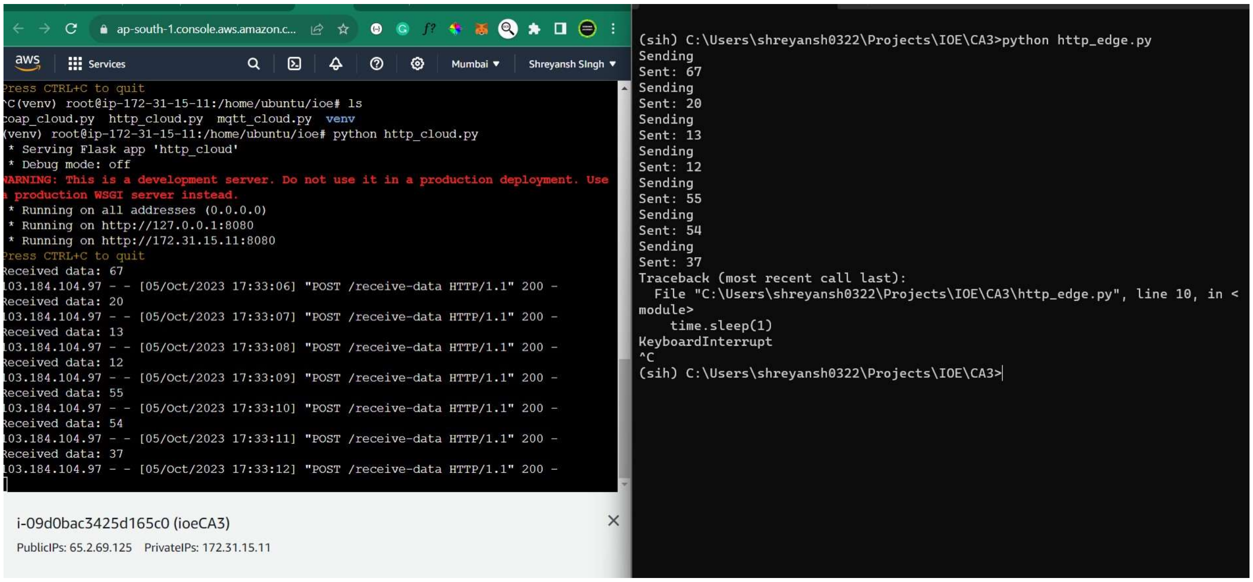
http



```
(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python http_cloud.py
 * Serving Flask app 'http_cloud'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.15.11:8080
Press CTRL+C to quit
```

i-09d0bac3425d165c0 (ioeCA3)

PublicIPs: 65.2.69.125   PrivateIPs: 172.31.15.11

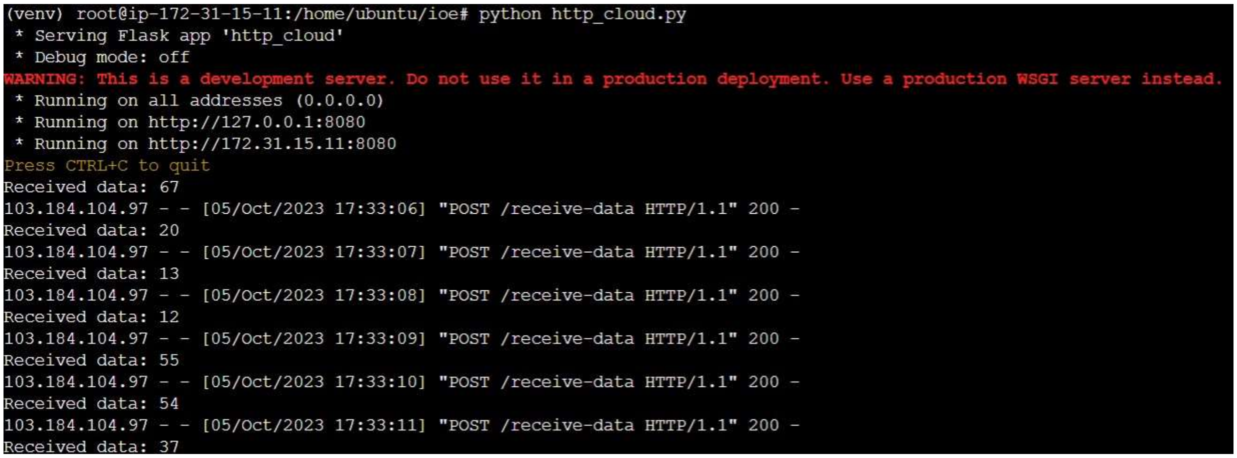


```
Press CTRL+C to quit
^C(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# ls
coap_cloud.py  http_cloud.py  mqtt_cloud.py  venv
(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python http_cloud.py
 * Serving Flask app 'http_cloud'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.15.11:8080
Press CTRL+C to quit
Received data: 67
103.184.104.97 - - [05/Oct/2023 17:33:06] "POST /receive-data HTTP/1.1" 200 -
Received data: 20
103.184.104.97 - - [05/Oct/2023 17:33:07] "POST /receive-data HTTP/1.1" 200 -
Received data: 13
103.184.104.97 - - [05/Oct/2023 17:33:08] "POST /receive-data HTTP/1.1" 200 -
Received data: 12
103.184.104.97 - - [05/Oct/2023 17:33:09] "POST /receive-data HTTP/1.1" 200 -
Received data: 55
103.184.104.97 - - [05/Oct/2023 17:33:10] "POST /receive-data HTTP/1.1" 200 -
Received data: 54
103.184.104.97 - - [05/Oct/2023 17:33:11] "POST /receive-data HTTP/1.1" 200 -
Received data: 37
103.184.104.97 - - [05/Oct/2023 17:33:12] "POST /receive-data HTTP/1.1" 200 -
```

i-09d0bac3425d165c0 (ioeCA3)

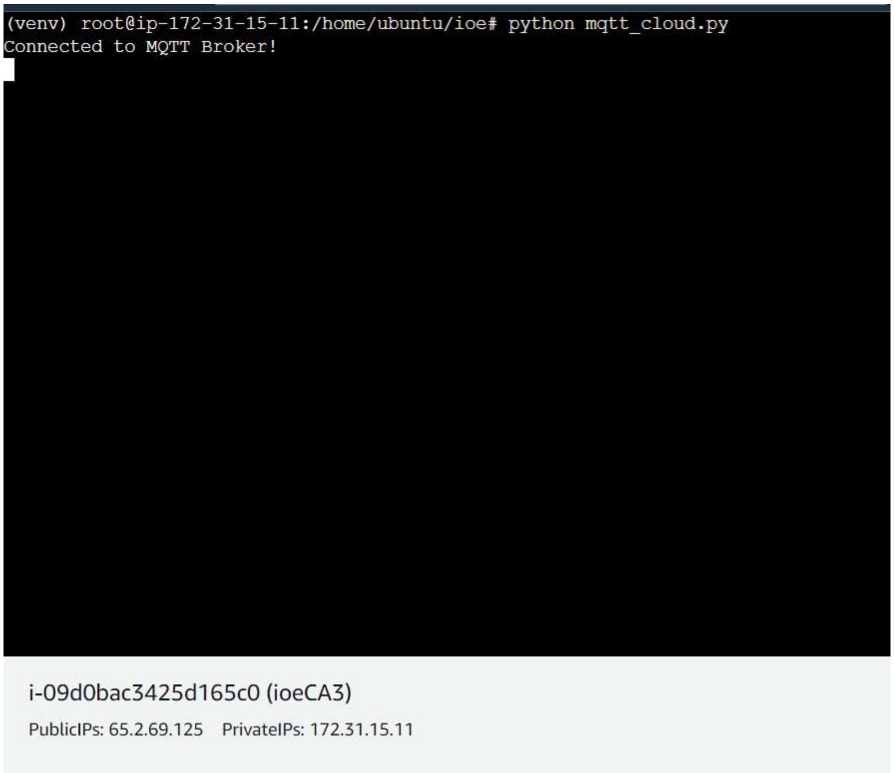
PublicIPs: 65.2.69.125   PrivateIPs: 172.31.15.11

```
(sih) C:\Users\shreyansh0322\Projects\IOE\CA3>python http_edge.py
Sending
Sent:  67
Sending
Sent:  20
Sending
Sent:  13
Sending
Sent:  12
Sending
Sent:  55
Sending
Sent:  54
Sending
Sent:  37
Traceback (most recent call last):
  File "C:\Users\shreyansh0322\Projects\IOE\CA3\http_edge.py", line 10, in <
module>
    time.sleep(1)
KeyboardInterrupt
^C
(sih) C:\Users\shreyansh0322\Projects\IOE\CA3>
```



```
(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python http_cloud.py
 * Serving Flask app 'http_cloud'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.15.11:8080
Press CTRL+C to quit
Received data: 67
103.184.104.97 - - [05/Oct/2023 17:33:06] "POST /receive-data HTTP/1.1" 200 -
Received data: 20
103.184.104.97 - - [05/Oct/2023 17:33:07] "POST /receive-data HTTP/1.1" 200 -
Received data: 13
103.184.104.97 - - [05/Oct/2023 17:33:08] "POST /receive-data HTTP/1.1" 200 -
Received data: 12
103.184.104.97 - - [05/Oct/2023 17:33:09] "POST /receive-data HTTP/1.1" 200 -
Received data: 55
103.184.104.97 - - [05/Oct/2023 17:33:10] "POST /receive-data HTTP/1.1" 200 -
Received data: 54
103.184.104.97 - - [05/Oct/2023 17:33:11] "POST /receive-data HTTP/1.1" 200 -
Received data: 37
```

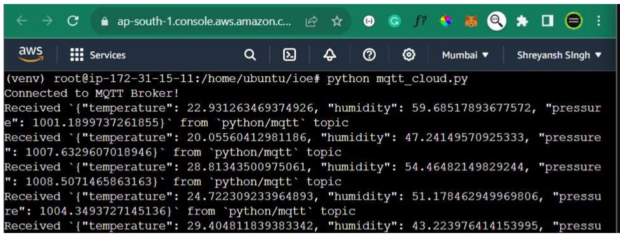
mqtt



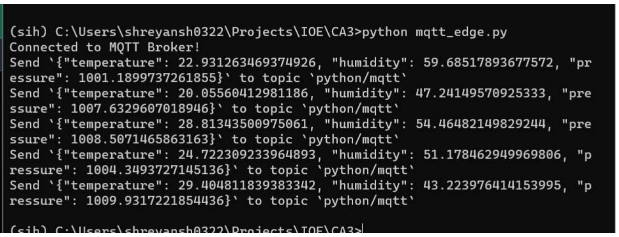
```
(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python mqtt_cloud.py
Connected to MQTT Broker!
```

i-09d0bac3425d165c0 (ioeCA3)

PublicIPs: 65.2.69.125   PrivateIPs: 172.31.15.11

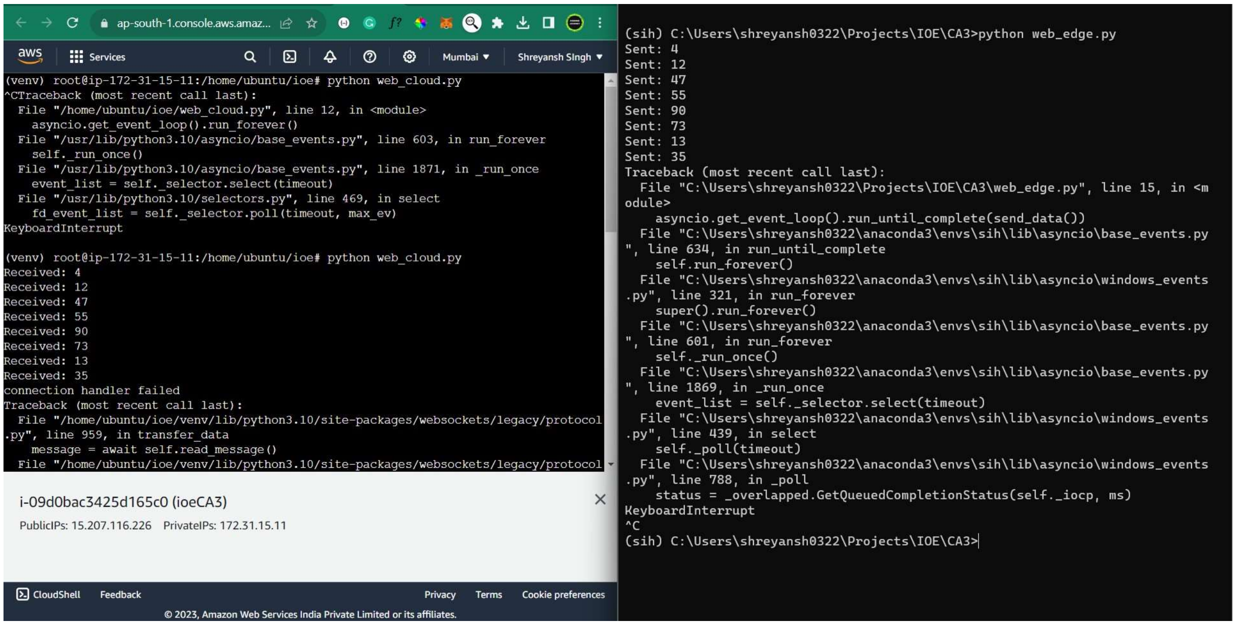


```
(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python mqtt_cloud.py
Connected to MQTT Broker!
Received `{"temperature": 22.931263469374926, "humidity": 59.68517893677572, "pressur
e": 1001.1899737261855}` from `python/mqtt` topic
Received `{"temperature": 20.05560412981186, "humidity": 47.24149570925333, "pressure
": 1007.6329607018946}` from `python/mqtt` topic
Received `{"temperature": 28.81343500975061, "humidity": 54.46482149829244, "pressure
": 1008.5071465863163}` from `python/mqtt` topic
Received `{"temperature": 24.722309233964893, "humidity": 51.178462949969806, "pressu
re": 1004.3493727145136}` from `python/mqtt` topic
Received `{"temperature": 29.404811839383342, "humidity": 43.223976414153995, "pressu
```



```
(sih) C:\Users\shreyansh0322\Projects\IOE\CA3>python mqtt_edge.py
Connected to MQTT Broker!
Send `{"temperature": 22.931263469374926, "humidity": 59.68517893677572, "pr
essure": 1001.1899737261855}` to topic `python/mqtt`
Send `{"temperature": 20.05560412981186, "humidity": 47.24149570925333, "pre
ssure": 1007.6329607018946}` to topic `python/mqtt`
Send `{"temperature": 28.81343500975061, "humidity": 54.46482149829244, "pre
ssure": 1008.5071465863163}` to topic `python/mqtt`
Send `{"temperature": 24.722309233964893, "humidity": 51.178462949969806, "p
ressure": 1004.3493727145136}` to topic `python/mqtt`
Send `{"temperature": 29.404811839383342, "humidity": 43.223976414153995, "p
ressure": 1009.9317221854436}` to topic `python/mqtt`

(sih) C:\Users\shreyansh0322\Projects\IOE\CA3>
```

websocket



```
(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python web_cloud.py
^CTraceback (most recent call last):
  File "/home/ubuntu/ioe/web_cloud.py", line 12, in <module>
    asyncio.get_event_loop().run_forever()
  File "/usr/lib/python3.10/asyncio/base_events.py", line 603, in run_forever
    self._run_once()
  File "/usr/lib/python3.10/asyncio/base_events.py", line 1871, in _run_once
    event_list = self._selector.select(timeout)
  File "/usr/lib/python3.10/selectors.py", line 469, in select
    fd_event_list = self._selector.poll(timeout, max_ev)
KeyboardInterrupt

(venv) root@ip-172-31-15-11:/home/ubuntu/ioe# python web_cloud.py
Received: 4
Received: 12
Received: 47
Received: 55
Received: 90
Received: 73
Received: 13
Received: 35
connection handler failed
Traceback (most recent call last):
  File "/home/ubuntu/ioe/venv/lib/python3.10/site-packages/websockets/legacy/protocol
.py", line 959, in transfer_data
    message = await self.read_message()
  File "/home/ubuntu/ioe/venv/lib/python3.10/site-packages/websockets/legacy/protocol
```

i-09d0bac3425d165c0 (ioeCA3)

PublicIPs: 15.207.116.226 PrivateIPs: 172.31.15.11

CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2023, Amazon Web Services India Private Limited or its affiliates.

```
(sih) C:\Users\shreyansh0322\Projects\IOE\CA3>python web_edge.py
Sent: 4
Sent: 12
Sent: 47
Sent: 55
Sent: 90
Sent: 73
Sent: 13
Sent: 35
Traceback (most recent call last):
  File "C:\Users\shreyansh0322\Projects\IOE\CA3\web_edge.py", line 15, in <m
odule>
    asyncio.get_event_loop().run_until_complete(send_data())
  File "C:\Users\shreyansh0322\anaconda3\envs\sih\lib\asyncio\base_events.py
", line 634, in run_until_complete
    self.run_forever()
  File "C:\Users\shreyansh0322\anaconda3\envs\sih\lib\asyncio\windows_events
.py", line 321, in run_forever
    super().run_forever()
  File "C:\Users\shreyansh0322\anaconda3\envs\sih\lib\asyncio\base_events.py
", line 601, in run_forever
    self._run_once()
  File "C:\Users\shreyansh0322\anaconda3\envs\sih\lib\asyncio\base_events.py
", line 1869, in _run_once
    event_list = self._selector.select(timeout)
  File "C:\Users\shreyansh0322\anaconda3\envs\sih\lib\asyncio\windows_events
.py", line 439, in select
    self._poll(timeout)
  File "C:\Users\shreyansh0322\anaconda3\envs\sih\lib\asyncio\windows_events
.py", line 788, in _poll
    status = _overlapped.GetQueuedCompletionStatus(self._iocp, ms)
KeyboardInterrupt
^C
(sih) C:\Users\shreyansh0322\Projects\IOE\CA3>
```

Conclusion :- we have applied and implemented edge to cloud protocols using a dummy data set successfully.