

Name: subhash polisetti

For Internet of Things (IoT) devices, connecting to the Internet is kind of a requirement. Connecting to the Internet allows the devices to work with each other and with backend services. The underlying network protocol of the Internet is **TCP/IP**. **Built on top of the TCP/IP stack, MQTT (Message Queue Telemetry Transport) has become the standard for IoT communications. MQTT can also run on SSL/TLS, which is a secure protocol built on TCP/IP, to ensure that all data communication between devices are encrypted and secure.**

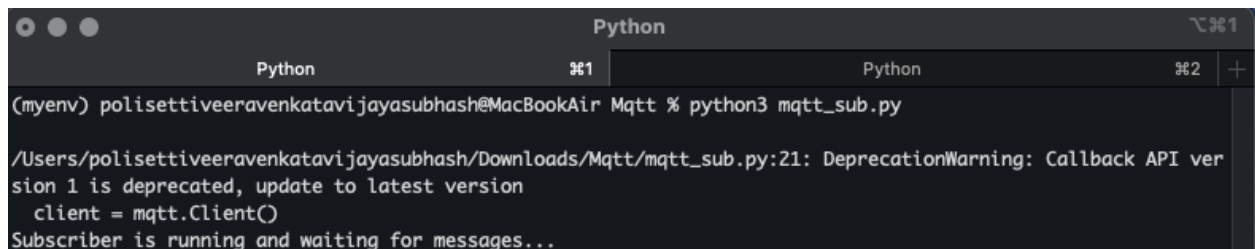
MQTT was **originally invented and developed by IBM in the late 1990's**. Its original application was to link sensors on oil pipelines with satellites. As its name suggests, it is a messaging protocol that supports asynchronous **communication between parties**. **An asynchronous messaging protocol de-couples the message sender and receiver in both space and time**, and hence is scalable in unreliable network environments.

Please install mosquitto and run the Publisher and Subscriber applications 10000 in an **asynchronous manner**. **That is, run producer 1000000 times and capture message count**. Now run consumer and make sure the same count is there!

Implementation:-

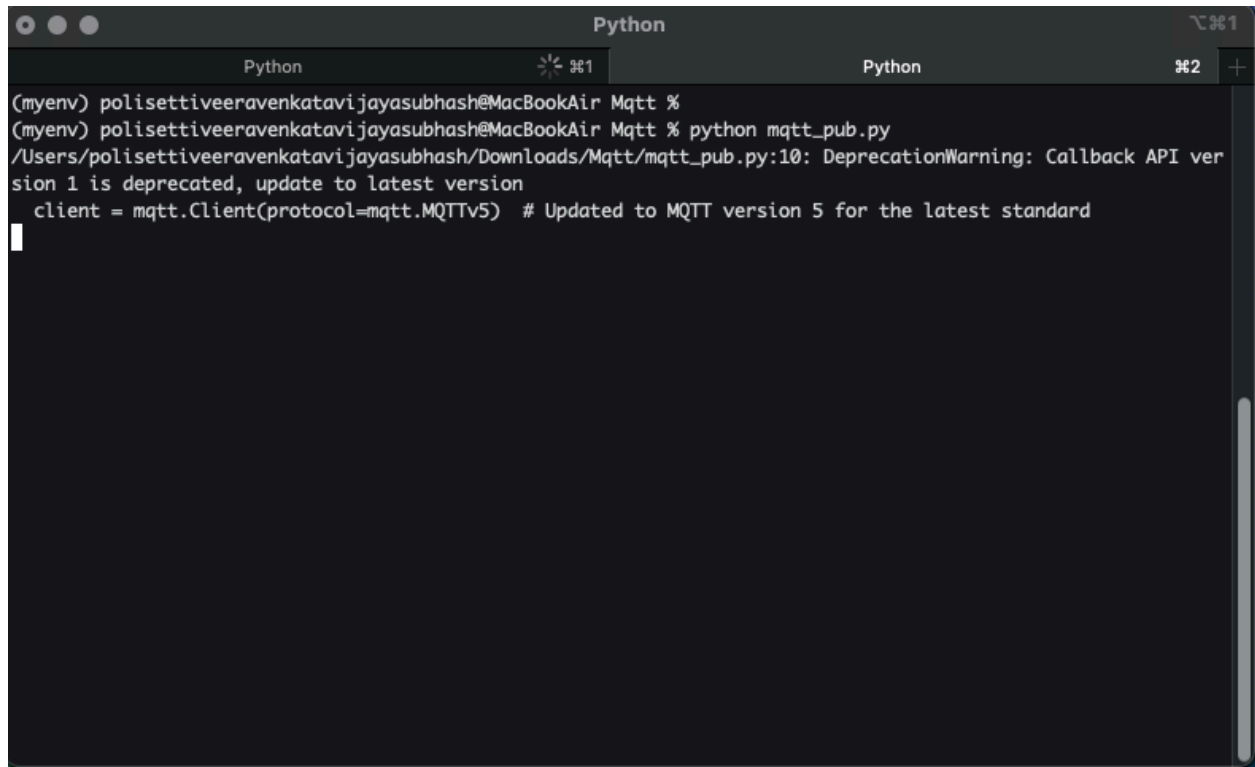
Github repo: <https://github.com/subhashpolisetti/Python-MQTT-Message-Stream>

Step-1: Run the subscriber to start listening for messages:

A screenshot of a macOS terminal window titled "Python". The terminal shows the command `python3 mqtt_sub.py` being executed. The output includes a deprecation warning about the MQTT callback API version 1 and a confirmation that the subscriber is running and waiting for messages.

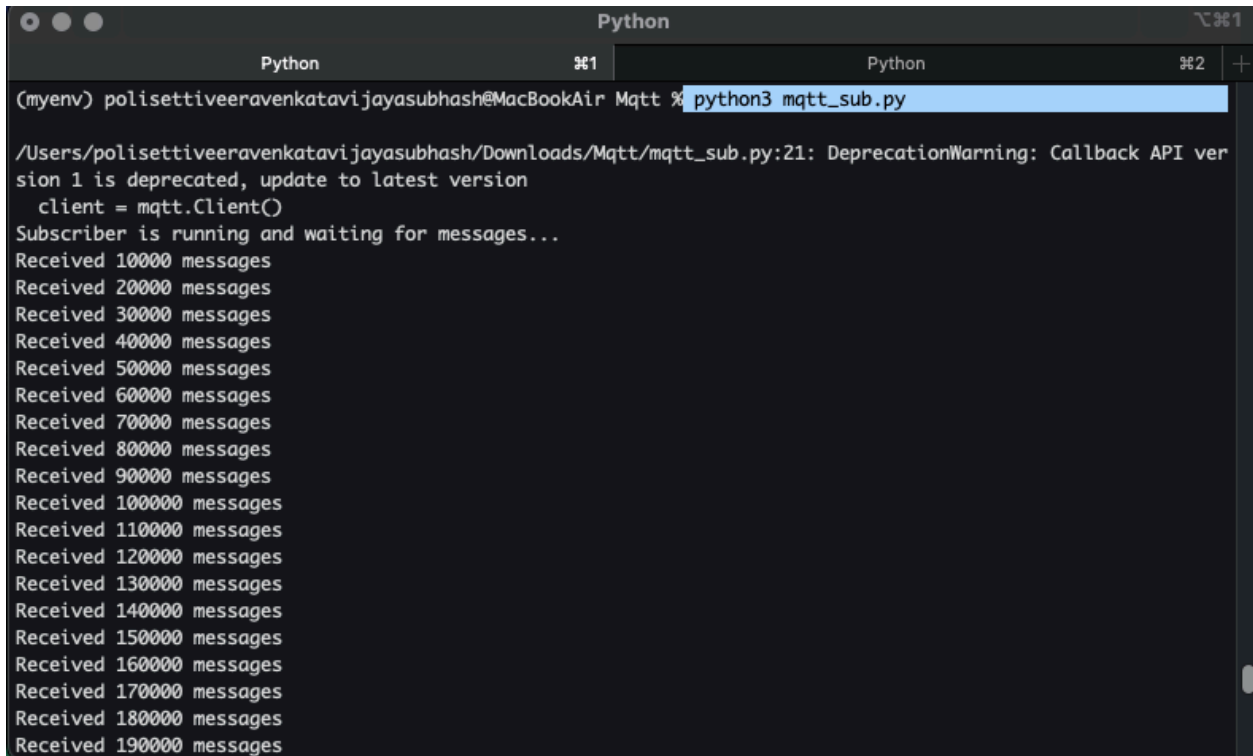
```
(myenv) polisettiveeravenkatavijayasubhash@MacBookAir Mqtt % python3 mqtt_sub.py
/Users/polisettiveeravenkatavijayasubhash/Downloads/Mqtt/mqtt_sub.py:21: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
Subscriber is running and waiting for messages...
```

Step-2: Run the publisher to start sending 1,000,000 messages

A screenshot of a macOS terminal window titled "Python". The window has a dark background and a light gray title bar with standard macOS window controls. The terminal shows a user running a Python script. The prompt is "(myenv) polisettiveeravenkatavijayasubhash@MacBookAir Mqtt %". The command entered is "python mqtt_pub.py". The output shows a deprecation warning: "/Users/polisettiveeravenkatavijayasubhash/Downloads/Mqtt/mqtt_pub.py:10: DeprecationWarning: Callback API version 1 is deprecated, update to latest version". Below the warning, the code "client = mqtt.Client(protocol=mqtt.MQTTv5) # Updated to MQTT version 5 for the latest standard" is visible. The cursor is at the end of the line.

```
(myenv) polisettiveeravenkatavijayasubhash@MacBookAir Mqtt %  
(myenv) polisettiveeravenkatavijayasubhash@MacBookAir Mqtt % python mqtt_pub.py  
/Users/polisettiveeravenkatavijayasubhash/Downloads/Mqtt/mqtt_pub.py:10: DeprecationWarning: Callback API ver  
sion 1 is deprecated, update to latest version  
  client = mqtt.Client(protocol=mqtt.MQTTv5) # Updated to MQTT version 5 for the latest standard
```

The subscriber is receiving the messages.

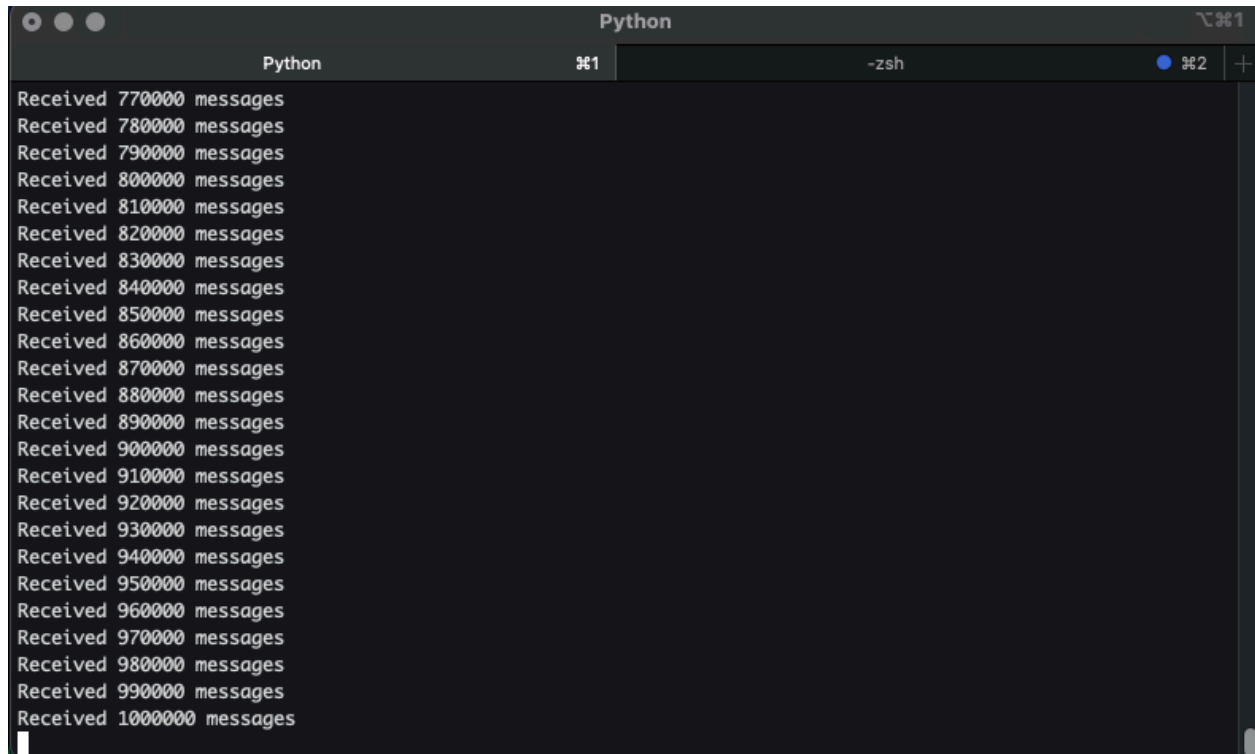
A terminal window titled 'Python' with three tabs. The active tab shows the command 'python3 mqtt_sub.py' being executed. The output displays a deprecation warning followed by a list of received message counts from 10,000 to 190,000 in increments of 10,000.

```
(myenv) polisetiveeravenkatavijayasubhash@MacBookAir Mqtt % python3 mqtt_sub.py

/Users/polisetiveeravenkatavijayasubhash/Downloads/Mqtt/mqtt_sub.py:21: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
Subscriber is running and waiting for messages...
Received 10000 messages
Received 20000 messages
Received 30000 messages
Received 40000 messages
Received 50000 messages
Received 60000 messages
Received 70000 messages
Received 80000 messages
Received 90000 messages
Received 100000 messages
Received 110000 messages
Received 120000 messages
Received 130000 messages
Received 140000 messages
Received 150000 messages
Received 160000 messages
Received 170000 messages
Received 180000 messages
Received 190000 messages
```

Verify the Message Count

- The subscriber terminal should display message counts every 10,000 messages until it reaches 1,000,000.
- This confirms that the subscriber received all published messages.

A terminal window titled "Python" with a dark background and light gray text. The window shows a list of messages received, starting from "Received 770000 messages" and ending with "Received 1000000 messages". The window has a standard macOS-style title bar with three colored buttons (red, yellow, green) on the left. The terminal content is as follows:

```
Received 770000 messages
Received 780000 messages
Received 790000 messages
Received 800000 messages
Received 810000 messages
Received 820000 messages
Received 830000 messages
Received 840000 messages
Received 850000 messages
Received 860000 messages
Received 870000 messages
Received 880000 messages
Received 890000 messages
Received 900000 messages
Received 910000 messages
Received 920000 messages
Received 930000 messages
Received 940000 messages
Received 950000 messages
Received 960000 messages
Received 970000 messages
Received 980000 messages
Received 990000 messages
Received 1000000 messages
```

Successfully Recieved 1000000 messages.

Detailed Steps:

Step 1: Install Mosquitto

Step 2: Install the `paho-mqtt` Python Package

Step 3: Write the Publisher Script

This script will connect to the Mosquitto broker and publish 1,000,000 messages.

```
output.txt  mqtt_pub.py  mqtt_sub.py
1 import paho.mqtt.client as mqtt
2 import time
3
4 # MQTT settings
5 broker = "localhost" # Use 'localhost' if Mosquitto is running on your machine
6 port = 1883 # Default port for MQTT
7 topic = "test/topic" # Topic to publish messages to
8
9 # Create a new MQTT client instance using the latest API
10 client = mqtt.Client(protocol=mqtt.MQTTv5) # Updated to MQTT version 5 for the
11
12 # Define callback for connection
13 def on_connect(client, userdata, flags, rc, properties=None):
14     if rc == 0:
15         print("Connected to the broker successfully.")
16     else:
17         print("Failed to connect, return code %d\n", rc)
18
19 client.on_connect = on_connect
20
21 # Connect to the broker
22 client.connect(broker, port)
23
24 # Publish 1,000,000 messages
25 message_count = 1000000
26
27 for i in range(message_count):
28     message = f"Message {i+1}"
29     client.publish(topic, payload=message, qos=0)
30
31     # Optional: Adjust this sleep time if messages are sent too fast for the s
32     time.sleep(0.001)
33
34 print("Publishing complete!")
35
36 # Disconnect from the broker
37 client.disconnect()
38
```

Step-4: Write the Subscriber Script

This script will connect to the broker and subscribe to the same topic, counting the messages it receives.

```
output.txt  mqtt_pub.py  mqtt_sub.py
1  import paho.mqtt.client as mqtt
2
3  # MQTT settings
4  broker = "localhost"
5  port = 1883
6  topic = "test/topic"
7
8  # Initialize message counter
9  message_count = 0
10
11 # Callback function for when a message is received
12 def on_message(client, userdata, message):
13     global message_count
14     message_count += 1
15
16     # Print count every 10,000 messages to monitor progress
17     if message_count % 10000 == 0:
18         print(f"Received {message_count} messages")
19
20 # Create MQTT client instance
21 client = mqtt.Client()
22
23 # Assign callback function
24 client.on_message = on_message
25
26 # Connect to the broker
27 client.connect(broker, port)
28
29 # Subscribe to the topic
30 client.subscribe(topic)
31
32 # Start the loop to process received messages
33 print("Subscriber is running and waiting for messages...")
34 client.loop_forever()
35
```

Step 5: Run the Mosquitto Broker

```
brew services start mosquitto
```

Step 6: Run the Subscriber Script

```
python3 mqtt_sub.py
```

Step 7: Run the Publisher Script

```
python3 mqtt_pub.py
```

Step 8: Verify the Message Count.

*****end*****