How to use MQTT in Python (Paho)

medium.com/@emqtt/how-to-use-mqtt-in-python-paho-4e622cb359f9

August 19, 2020

EMQ X

Aug 19, 2020·5 min read

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

MQTT is a kind of **lightweight IoT messaging protocl** based on the publish/subscribe model, which can provide real-time and reliable messaging service for IoT devices, only using very little code and bandwidth. It is suitable for devices with limited hardware resources and the network environment with limited bandwidth. Therefore, MQTT protocol is widely used in IoT, mobile internet, IoV, electricity power, and other industries.

This article mainly introduces how to use the **paho-mqtt** client and implement connection, subscribe, messaging, and other functions between the client and MQTT broker, in the Python project.

Project initialization

This project uses Python 3.6 to develop and test. Readers can use the following command to confirm the Python version.

```
➜  ~ python3 --versionPython 3.6.7
```

Choose the MQTT client

The Paho Python Client provides a client class with support for both MQTT v3.1 and v3.1.1 on Python 2.7 or 3.x. It also provides some helper functions to make publishing one off messages to an MQTT server very straightforward.

Using pip to install the Paho MQTT client

Pip is a management tool for the Python package. This tool provides find, download, install and uninstall functions for Python package.

```
pip3 install paho-mqtt
```

The use of Python MQTT

Connect to the MQTT broker

This article will use the free public MQTT broker provided by EMQ X. This service is based on MQTT IoT cloud platform to create. The accessing information of the broker is as follows:

- Broker:
- TCP Port:
- Websocket Port:

Import the Paho MQTT client

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

Set the parameter of MQTT Broker connection

Set the address, port and topic of MQTT Broker connection. At the same time, we call the Python function `random.randint` to randomly generate the MQTT client id.

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

Write the MQTT connect function

Write the connect callback function `on_connect`. This function will be called after connecting the client, and we can determine whether the client is connected successfully according to `rc` in this function. Usually, we will create an MQTT client at the same time and this client will connect to `broker.emqx.io`.

```
def connect_mqtt():    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)    # Set Connecting Client ID
client = mqtt_client.Client(client_id)    client.on_connect = on_connect
client.connect(broker, port)    return client
```

First, we define a while loop. In this loop, and we will set the MQTT client `publish` function to send messages to the topic `/python/mqtt` every second.

```
def publish(client):     msg_count = 0     while True:         time.sleep(1)
msg = f"messages: {msg_count}"         result = client.publish(topic, msg)
# result: [0, 1]         status = result[0]         if status == 0:
print(f"Send `{msg}` to topic `{topic}`")         else:             print(f"Failed
to send message to topic {topic}")         msg_count += 1
```

Subscribe to messages

Write the message callback function `on_message`. This function will be called after the client received messages from the MQTT Broker. In this function, we will print out the name of subscribed topics and the received messages.

```
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
```

The complete code

The code of publishing messages

```
# python 3.6import randomimport timefrom paho.mqtt import client as
mqtt_clientbroker = 'broker.emqx.io'port = 1883topic = "/python/mqtt"# generate
client ID with pub prefix randomlyclient_id = f'python-mqtt-{random.randint(0,
1000)}'def connect_mqtt():    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.on_connect = on_connect
client.connect(broker, port)    return clientdef publish(client):    msg_count = 0
while True:        time.sleep(1)        msg = f"messages: {msg_count}"
result = client.publish(topic, msg)        # result: [0, 1]        status =
result[0]        if status == 0:            print(f"Send `{msg}` to topic
`{topic}`")        else:            print(f"Failed to send message to topic
{topic}")        msg_count += 1def run():    client = connect_mqtt()
client.loop_start()    publish(client)if __name__ == '__main__':    run()
```

The code of subscribing to messages

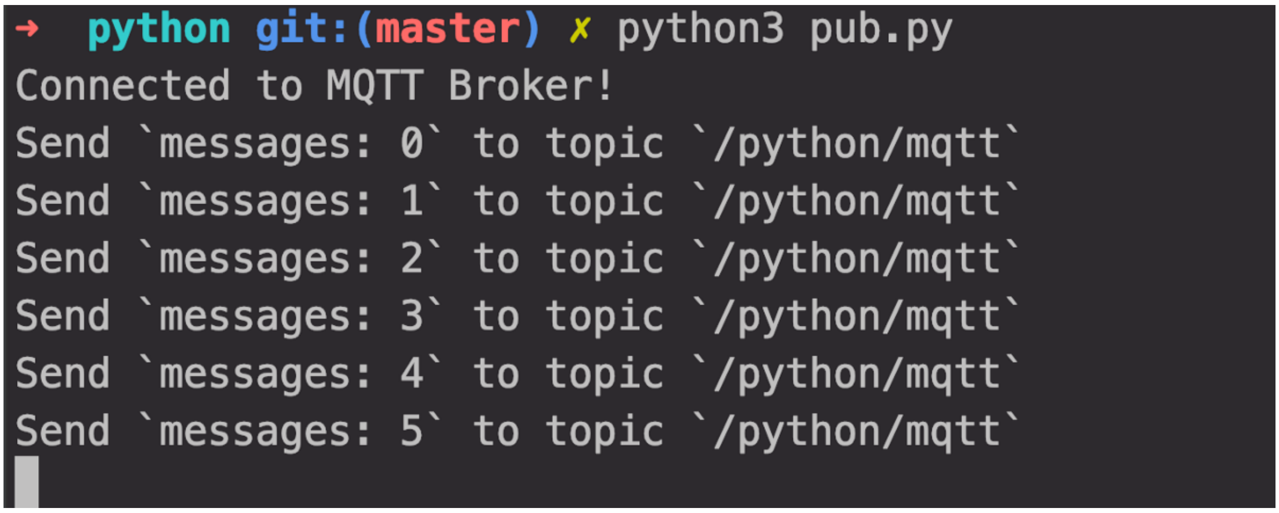
```
# python3.6import randomfrom paho.mqtt import client as mqtt_clientbroker =
'broker.emqx.io'port = 1883topic = "/python/mqtt"# generate client ID with pub
prefix randomlyclient_id = f'python-mqtt-{random.randint(0, 100)}'def
connect_mqtt() -> mqtt_client:    def on_connect(client, userdata, flags, 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.on_connect = on_connect
client.connect(broker, port)    return clientdef 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_messagedef run():    client = connect_mqtt()
subscribe(client)    client.loop_forever()if __name__ == '__main__':    run()
```

Test

Publish messages

Run the code of publishing messages, we will see that the client connects successfully and publishes messages successfully

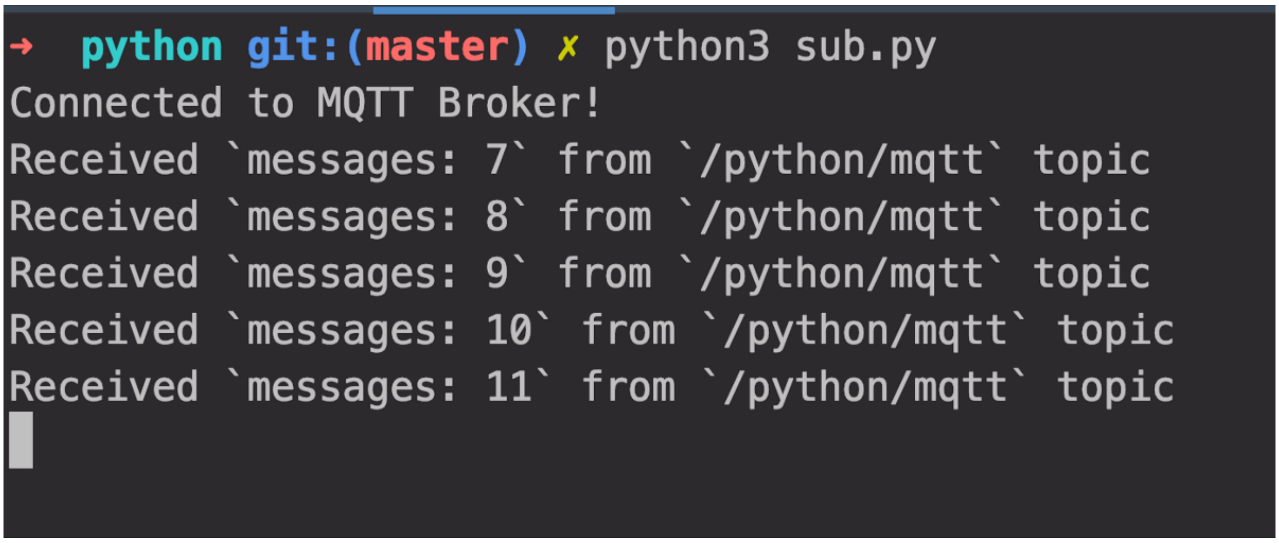
```
python3 pub.py
```


```
➜  python git:(master) ✗ python3 pub.py
Connected to MQTT Broker!
Send `messages: 0` to topic `/python/mqtt`
Send `messages: 1` to topic `/python/mqtt`
Send `messages: 2` to topic `/python/mqtt`
Send `messages: 3` to topic `/python/mqtt`
Send `messages: 4` to topic `/python/mqtt`
Send `messages: 5` to topic `/python/mqtt`
```

Subscribe to messages

Run the code of subscribing to messages, we will see that the client connects successfully and receives the published messages successfully

```
python3 sub.py
```


```
➜  python git:(master) ✗ python3 sub.py
Connected to MQTT Broker!
Received `messages: 7` from `/python/mqtt` topic
Received `messages: 8` from `/python/mqtt` topic
Received `messages: 9` from `/python/mqtt` topic
Received `messages: 10` from `/python/mqtt` topic
Received `messages: 11` from `/python/mqtt` topic
```

Summary

So far, we have finished that use the **paho-mqtt** client to connect to the free public MQTT broker, and implemented the connect, publish messages and subscribe to messages between the test client and MQTT broker.

Python is different from the high-level languages like C++ or Java, it is more suitable for implementing the business logic on the device side. Using Python can reduce the logic complexity of code and reducing the cost of interaction with the device. We believe that Python will have wider applications in the field of IoT.

Next, we will publish more articles about IoT development and Python. Stay tuned.

1 https://en.wikipedia.org/wiki/Python_(programming_language) ↩