**The use of Python MQTT**

**Connect to the MQTT broker**

This article will use [the free public MQTT broker](https://www.emqx.com/en/mqtt/public-mqtt5-broker) provided by EMQX. This service is based on [MQTT cloud service - EMQX Cloud](https://www.emqx.com/en/cloud) to create. The accessing information of the broker is as follows:

* Broker: **broker.emqx.io**
* TCP Port: **1883**
* Websocket Port: **8083**

**Import the Paho MQTT client**

from paho.mqtt import client as mqtt\_client

**Set the parameter of the MQTT Broker connection**

Set the address, port and topic of the 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 = 1883

topic = "python/mqtt"

client\_id = f'python-mqtt-{random.randint(0, 1000)}'

# username = 'emqx'

# password = 'public'

**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.username\_pw\_set(username, password)

client.on\_connect = on\_connect

client.connect(broker, port)

return client

**Publish messages**

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**

Write the message callback function on\_message. This function will be called after the client receives 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 full code**

**The code of publishing messages**

# python 3.6

import random

import time

from paho.mqtt import client as mqtt\_client

broker = 'broker.emqx.io'

port = 1883

topic = "python/mqtt"

# generate client ID with pub prefix randomly

client\_id = f'python-mqtt-{random.randint(0, 1000)}'

# username = 'emqx'

# password = 'public'

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.username\_pw\_set(username, password)

client.on\_connect = on\_connect

client.connect(broker, port)

return client

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

def run():

client = connect\_mqtt()

client.loop\_start()

publish(client)

if \_\_name\_\_ == '\_\_main\_\_':

run()

**The code of subscribing**

# python3.6

import random

from paho.mqtt import client as mqtt\_client

broker = 'broker.emqx.io'

port = 1883

topic = "python/mqtt"

# generate client ID with pub prefix randomly

client\_id = f'python-mqtt-{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()