






Getting started with Paho Py


 Clusters

 Billing

 Help

 Feedback

 Logout



Getting started with Paho Python

← BACK TO GETTING STARTED

Install Paho Python



For an already finished project that provides you with all the code in this guide, clone this [example repository](#) and follow the `README.me`. You can also copy the code below into your own project, it is the same as `simple-example.py` on GitHub.


Prerequisites


- [Python](#) installed


Create a new project folder named `python-paho-hivemq-cloud` and open a console inside it.


Paho python can then be installed via the `pip` package manager by running the following command:


 Getting started with Paho Py

 Clusters

 Billing

 Help

 Feedback

 Logout

or for Python 3

```
pip3 install paho-mqtt
```

Connect MQTT clients



Create a file `mqtt_client.py` in the folder `python-paho-hivemq-cloud` and open it with your preferred editor. Add the python code shown below.

Use this code to connect to your HiveMQ Cloud cluster via the Paho Python library. The `host name` of your cluster and your `username` are already inserted automatically. To fully verify your credentials, replace the variable `'YOUR_PASSWORD'` with the value you entered when creating your credentials. As HiveMQ Cloud does not support insecure connections, TLS is required. The code below enables TLS by using:


```
# enable TLS client.tls_set(tls_version=mqtt.cli
```


This enables a secure connection and sets the default security context. The default port used for secure MQTT connections is **8883**.


```
#  
# Copyright 2021 HiveMQ GmbH  
#  
# Licensed under the Apache License, Version 2.0  
# you may not use this file except in compliance  
# You may obtain a copy of the License at
```





Getting started with Paho Py

 Clusters



 Billing

 Help


 Feedback


 Logout


```
#  
  
# Unless required by applicable law or agreed to  
# distributed under the License is distributed o  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,  
# See the License for the specific language gove  
# limitations under the License.  
  
#  
import time  
import paho.mqtt.client as paho  
from paho import mqtt  
  
# setting callbacks for different events to see  
def on_connect(client, userdata, flags, rc, prop  
    print("CONNACK received with code %s." % rc)  
  
# with this callback you can see if your publish  
def on_publish(client, userdata, mid, properties  
    print("mid: " + str(mid))  
  
# print which topic was subscribed to  
def on_subscribe(client, userdata, mid, granted_  
    print("Subscribed: " + str(mid) + " " + str(  
  
# print message, useful for checking if it was s  
def on_message(client, userdata, msg):  
    print(msg.topic + " " + str(msg.qos) + " " +  
  
# using MQTT version 5 here, for 3.1.1: MQTTv311  
# userdata is user defined data of any type, upd  
# client_id is the given name of the client  
client = paho.Client(client_id="", userdata=None  
client.on_connect = on_connect  
  
# enable TLS for secure connection  
client.tls_set(tls_version=mqtt.client.ssl.PROTO  
# set username and password
```





Getting started with Paho Py

 Clusters

 Billing

 Help

 Feedback

 Logout

```
client.connect("6dd678185e194e4ca3c7f1d5fe9ab113")

# setting callbacks, use separate functions like
client.on_subscribe = on_subscribe
client.on_message = on_message
client.on_publish = on_publish

# subscribe to all topics of encyclopedia by using
client.subscribe("encyclopedia/#", qos=1)

# a single publish, this can also be done in loop
client.publish("encyclopedia/temperature", payload)



# loop_forever for simplicity, here you need to
# you can also use loop_start and loop_stop
client.loop_forever()
```

You can run the application from your console by starting it with python:


```
python mqtt_client.py
```


Publish and Subscribe with your MQTT client


This code creates an MQTT client that is capable of publishing and subscribing to topics on your HiveMQ Cloud cluster. First `callbacks` are declared for the events `on_connect`, `on_publish`, `on_subscribe` and `on_message`. They each print a confirmation message and the respective content to the console. Then a MQTT 5 capable Paho client is created. To provide a secure connection to your HiveMQ Cloud cluster, TLS is utilized. Using your username, password, hostname and the standard port `8883`, it





Getting started with Paho Py

 Clusters

 Billing

 Help

 Feedback

 Logout

`encyclopedia/#` with QoS = 1. The symbol `#` is used as a [wildcard](#), which means that the client subscribes to all topics that begin with `"encyclopedia/"`. Then a message with the content `"hot"` and QoS = 1 is published to the topic `"encyclopedia/temperature"`. A confirmation message will be printed for each step, just like it was implemented through the various callbacks. Through `client.loop_forever()` this program will run in a loop until it is stopped.

Next steps

Get familiar with the Paho Python API and build your first application.

Further documentation on Paho Python can be found in the [MQTT Client Library Encyclopedia](#) or on the [Paho python GitHub](#)

Keep in mind that HiveMQ Cloud uses TLS encryption and requires a username and a password to authenticate.

← BACK TO GETTING STARTED