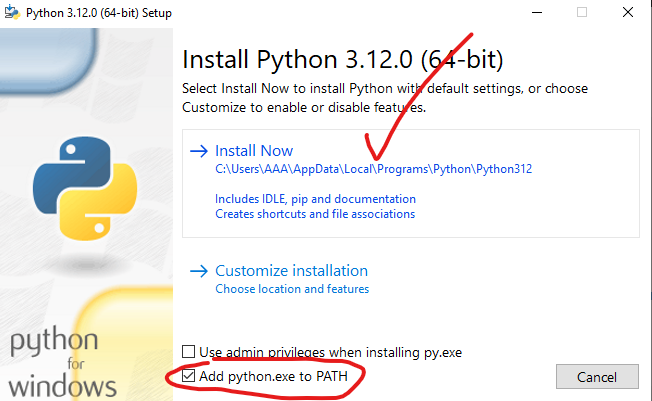
# Conexión desde Python a MQTT

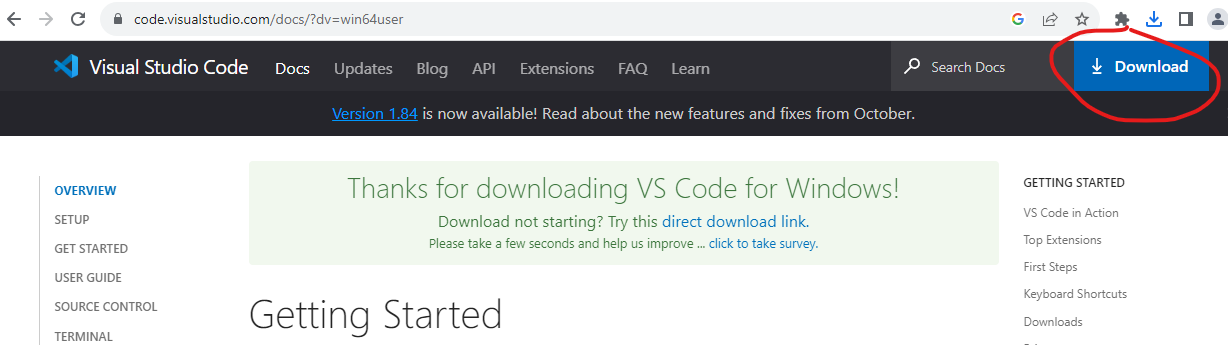
<https://www.hivemq.com/article/mqtt-client-library-paho-python/>

1.- Descarga e instala Python, desde Python.org

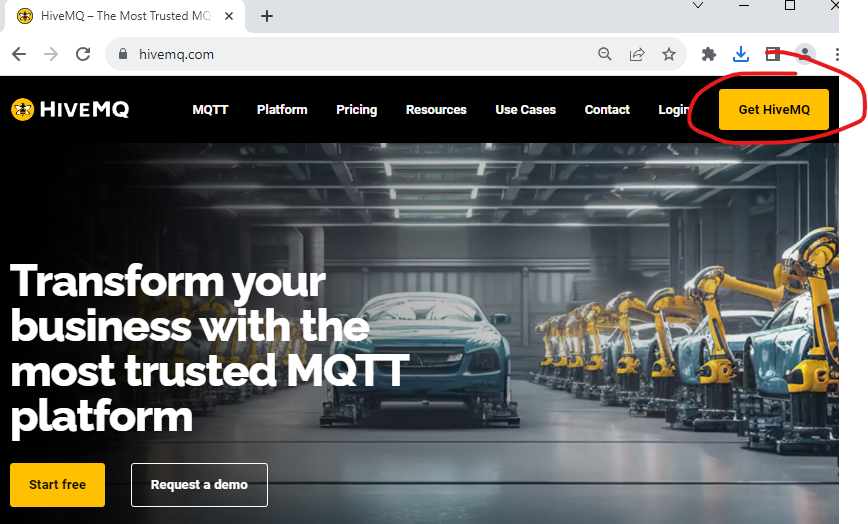


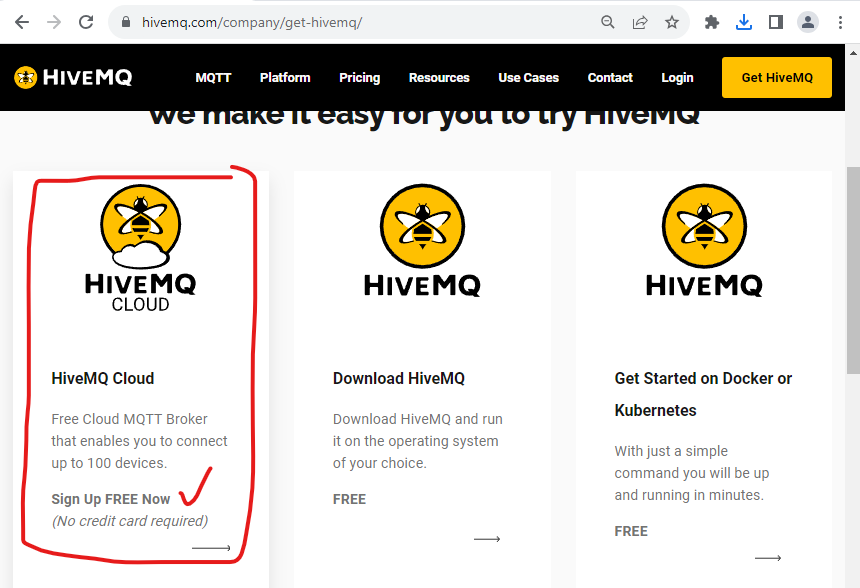


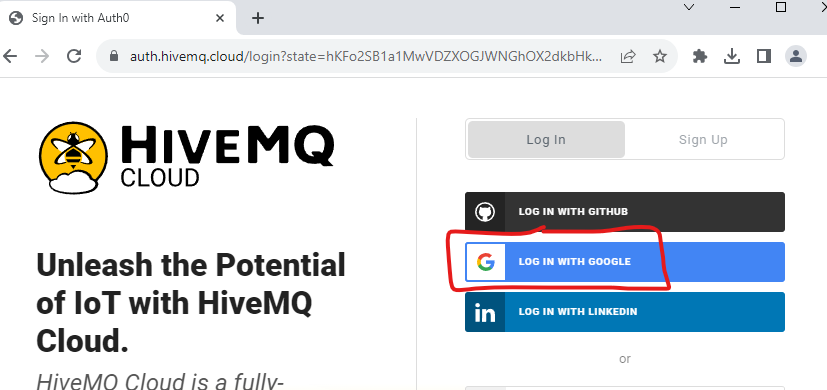
2.- Descarga e instala VSCode desde <https://code.visualstudio.com/>

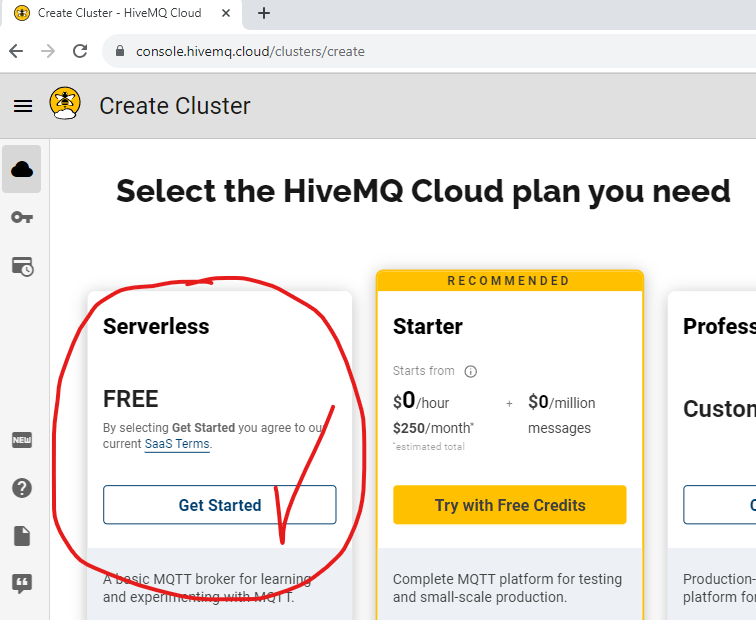


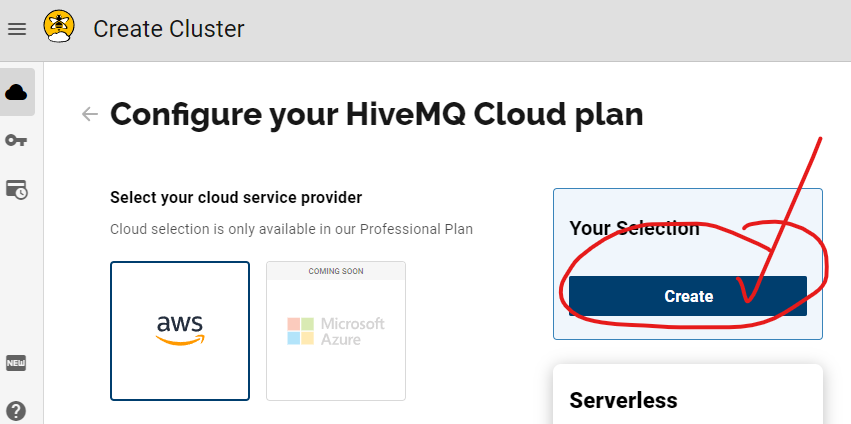
3.- Registra una cuenta en hivemq.com

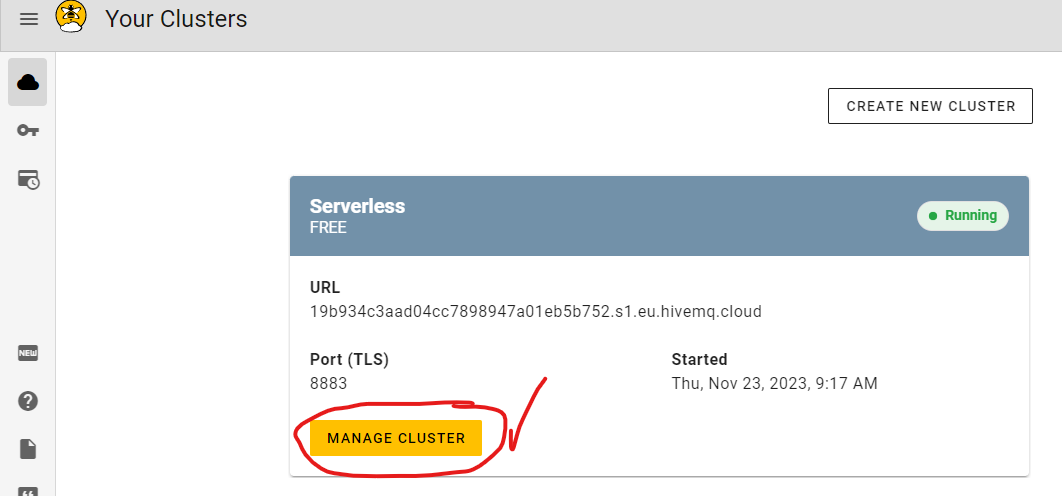


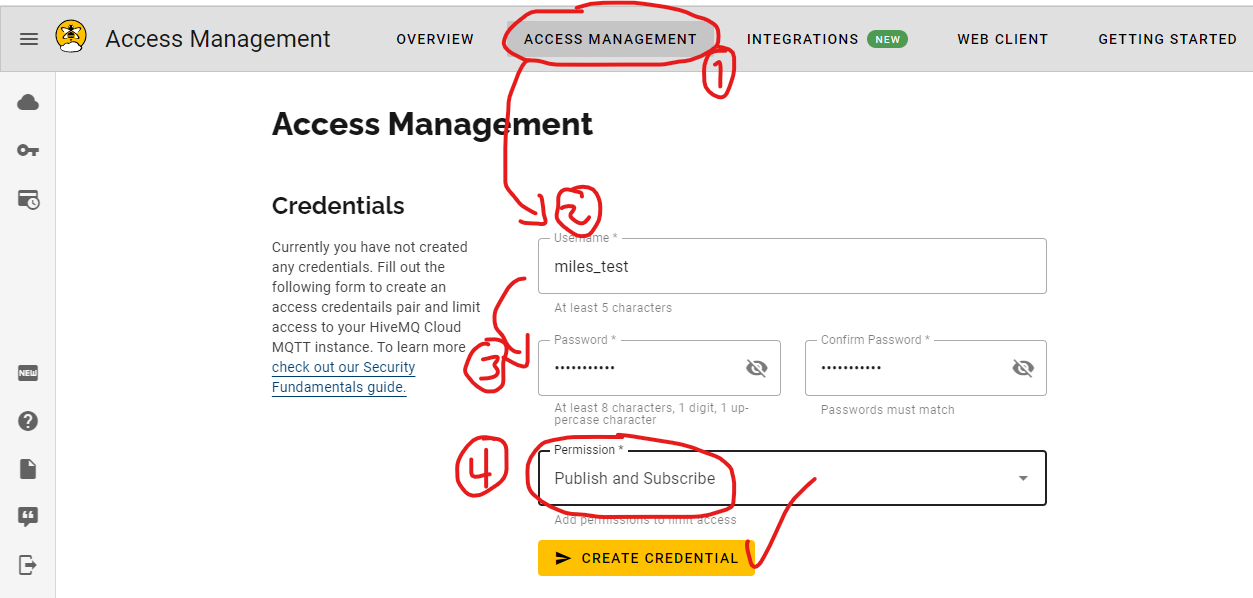




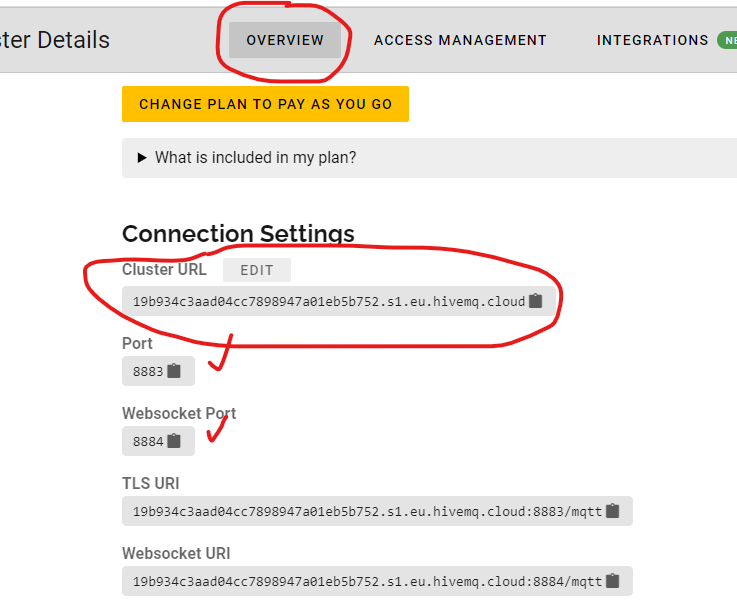






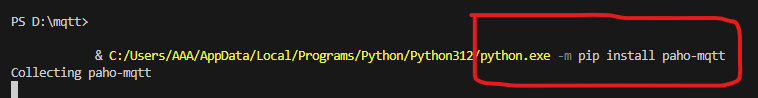


Datos de conexión



4.- Genera un programa en Python, que publique un dato en el servidor hivemq.com, por medio de “paho mqtt client”, para Python.

Desde un terminal, instalar “pip install paho-mqtt” o bien, desde la ruta en donde se encuenta Python instalado, “Python -m pip install paho-mqtt”



Obtenido desde: https://github.com/hivemq-cloud/paho-mqtt-client-example/tree/master

import time

import paho.mqtt.client as paho

from paho import mqtt

# setting callbacks for different events to see if it works, print the message etc.

def on\_connect(client, userdata, flags, rc, properties=None):

    """

        Prints the result of the connection with a reasoncode to stdout ( used as callback for connect )

        :param client: the client itself

        :param userdata: userdata is set when initiating the client, here it is userdata=None

        :param flags: these are response flags sent by the broker

        :param rc: stands for reasonCode, which is a code for the connection result

        :param properties: can be used in MQTTv5, but is optional

    """

    print("CONNACK received with code %s." % rc)

    print("Conectando a MQTT")

# with this callback you can see if your publish was successful

def on\_publish(client, userdata, mid, properties=None):

    """

        Prints mid to stdout to reassure a successful publish ( used as callback for publish )

        :param client: the client itself

        :param userdata: userdata is set when initiating the client, here it is userdata=None

        :param mid: variable returned from the corresponding publish() call, to allow outgoing messages to be tracked

        :param properties: can be used in MQTTv5, but is optional

    """

    print("mid: " + str(mid))

# print which topic was subscribed to

def on\_subscribe(client, userdata, mid, granted\_qos, properties=None):

    """

        Prints a reassurance for successfully subscribing

        :param client: the client itself

        :param userdata: userdata is set when initiating the client, here it is userdata=None

        :param mid: variable returned from the corresponding publish() call, to allow outgoing messages to be tracked

        :param granted\_qos: this is the qos that you declare when subscribing, use the same one for publishing

        :param properties: can be used in MQTTv5, but is optional

    """

    print("Subscribed: " + str(mid) + " " + str(granted\_qos))

# print message, useful for checking if it was successful

def on\_message(client, userdata, msg):

    """

        Prints a mqtt message to stdout ( used as callback for subscribe )

        :param client: the client itself

        :param userdata: userdata is set when initiating the client, here it is userdata=None

        :param msg: the message with topic and payload

    """

    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))

# using MQTT version 5 here, for 3.1.1: MQTTv311, 3.1: MQTTv31

# userdata is user defined data of any type, updated by user\_data\_set()

# client\_id is the given name of the client

client = paho.Client(client\_id="19b934c3aad04cc7898947a01eb5b752.s1.eu.hivemq.cloud", userdata=None, protocol=paho.MQTTv5)

client.on\_connect = on\_connect

# enable TLS for secure connection

client.tls\_set(tls\_version=mqtt.client.ssl.PROTOCOL\_TLS)

# set username and password

client.username\_pw\_set("miles\_test", ".Test.2023.")

# connect to HiveMQ Cloud on port 8883 (default for MQTT)

client.connect("19b934c3aad04cc7898947a01eb5b752.s1.eu.hivemq.cloud", 8883)

# setting callbacks, use separate functions like above for better visibility

client.on\_subscribe = on\_subscribe

client.on\_message = on\_message

client.on\_publish = on\_publish

# subscribe to all topics of encyclopedia by using the wildcard "#"

client.subscribe("sensores/#", qos=1)

# a single publish, this can also be done in loops, etc.

client.publish("sensores/temperatura", payload="cold", qos=1)

# loop\_forever for simplicity, here you need to stop the loop manually

# you can also use loop\_start and loop\_stop

client.loop\_forever()