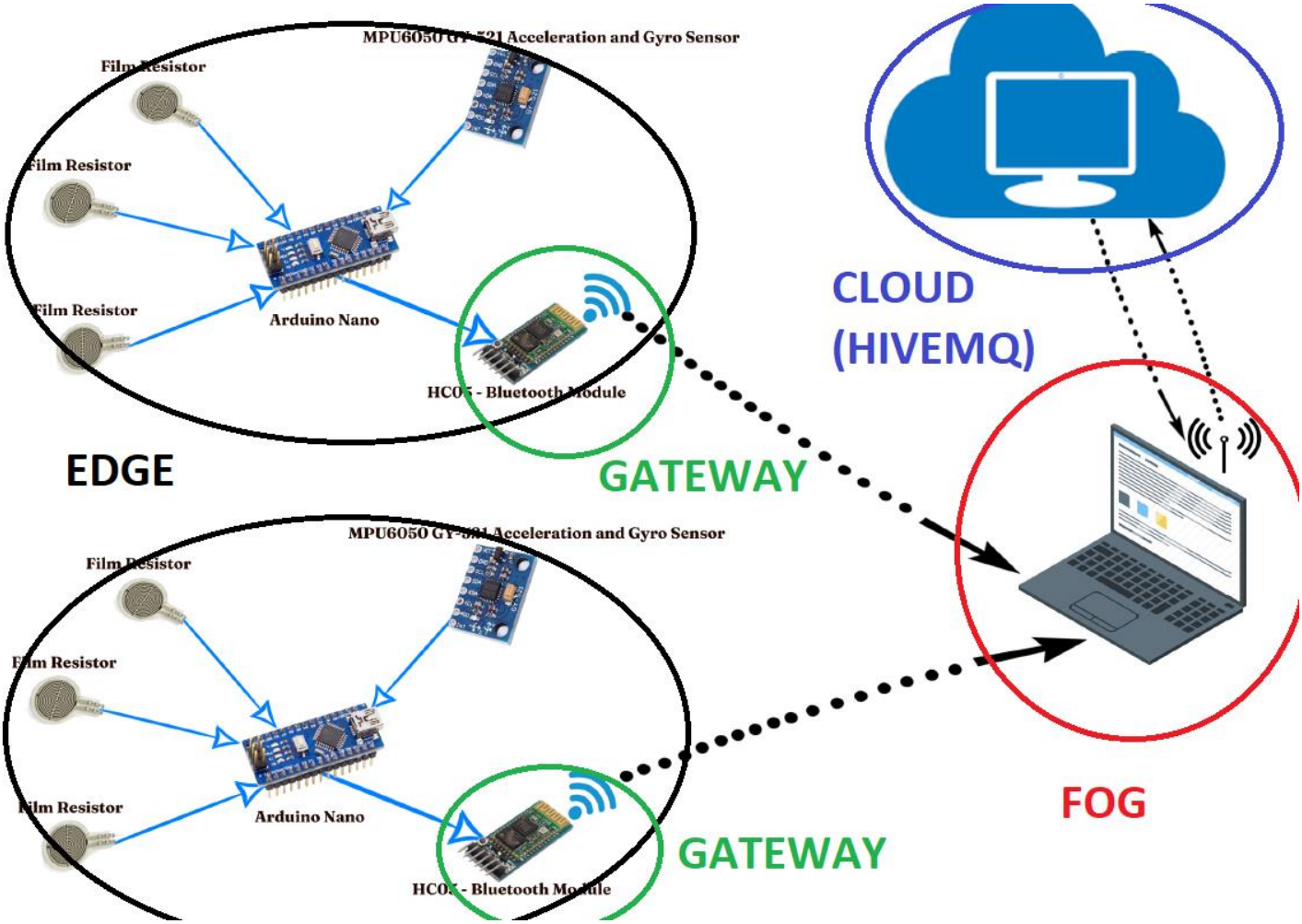
FOG COMPUTING



- Our sensors and arduinos are in the Edge part of our project.

In Fog part of our system:

1) We firstly import necessary libraries for serial connection to our computer.

```
import serial
import serial.tools.list_ports
import time
import pandas as pd
import csv
```

2) We take list of all connected serial ports in our computer

```
################################################################################
########## This function is to finding All Ports on the device. ################
def get_ports():
    ports = serial.tools.list_ports.comports()

    return ports
```

3) In our list of ports, we take those which has COM in their names

```
#############################################################################################################
########## This function is  to finding the port which the Aduino was connected to the device. ###############
def findArduino(portsFound):
    commPort = []
    numConnection = len(portsFound)

    for i in range(0, numConnection):
        port = portsFound[i]
        strPort = str(port)

        if 'COM' in strPort:
            splitPort = strPort.split(" ")
            commPort.append(splitPort[0])

    return commPort
```

4) Call the upper functions and print the results

```
#############################################################################################
######################### The Main Code ###################################################

foundPorts = get_ports()
ArduinoPort = findArduino(foundPorts)
print(ArduinoPort)
```

5) We open serial connection for each of our connected COM ports

```
ser = []

for i in ArduinoPort:
    ser.append(serial.Serial(i, baudrate= 115200, timeout=(0.5)))

for i in range(len(ArduinoPort)):
    ser[i].flushInput()
    ser[i].close()
    time.sleep(1)
    ser[i].open()
    time.sleep(2)
```

6) We create a dataframe named data.csv for save the datas after

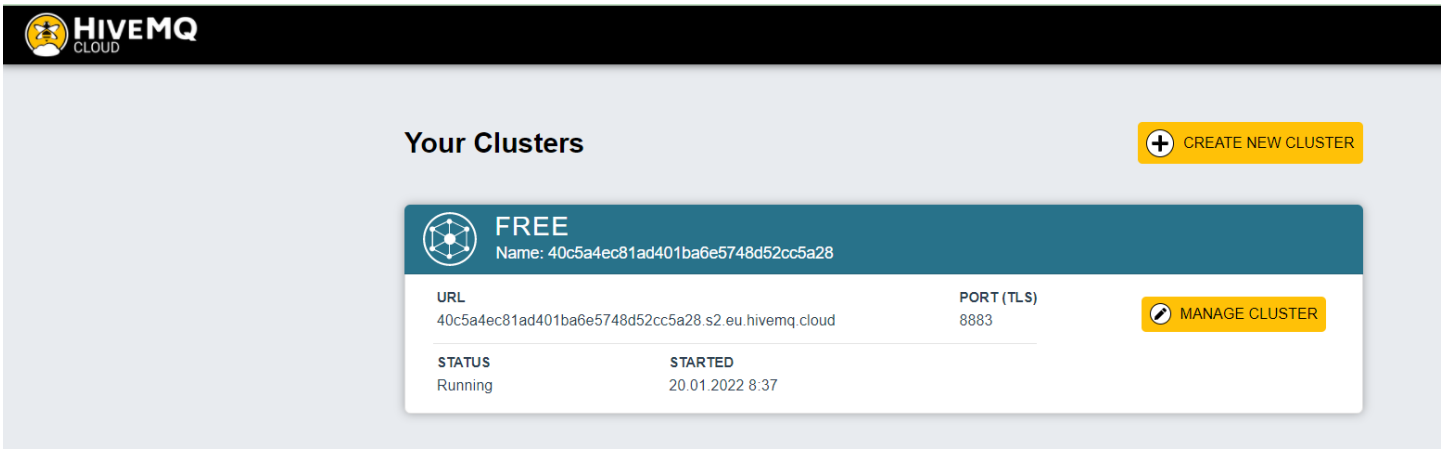
```
fields = ['ivmeX','ivmeY','ivmeZ','tmp','GyroX','GyroY','GyroZ','A0','A1','A2','ivmeX2','ivmeY2','ivmeZ2','tmp2','GyroX2','GyroY

with open('data.csv', 'w',newline='') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)
    # writing the fields
    csvwriter.writerow(fields)
```

7) Lastly we take data from the two Bluetooth serial connections and both save them to csv and publish them on cloud

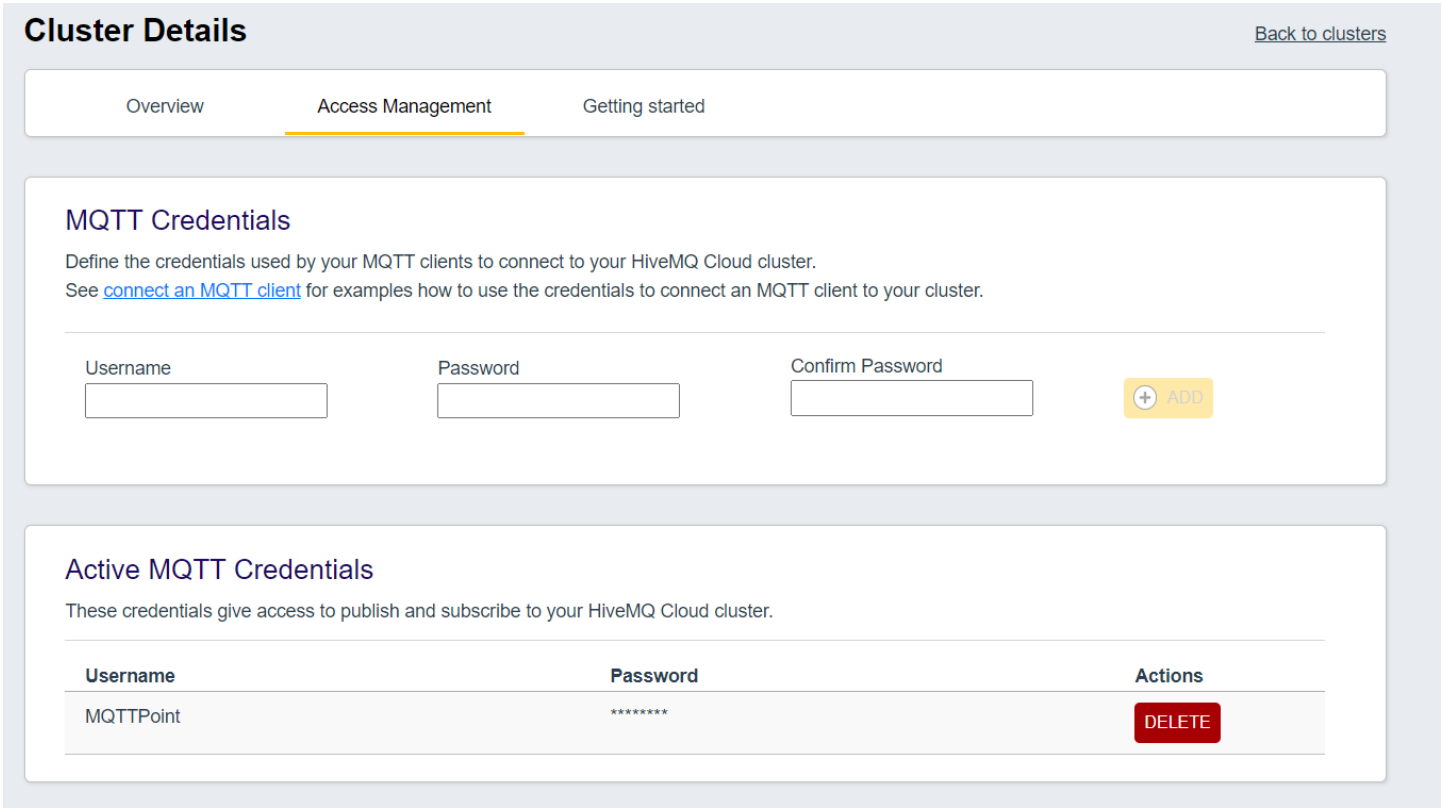
```
data = []
i = 0

while True:
    line = str(ser[0].readline().decode('ascii').strip())
    line2 = str(ser[1].readline().decode('ascii').strip())

    try:
        info=line.split(" ")
        infox=line2.split(" ")
        info2=[info[0],info[1],info[2],info[3],info[4],info[5],info[6],info[7],info[8],info[9],infox[0],infox[1],infox[2],infox[
        print(info2)

        #Save the data to csv
        data.append(info2)
        #Send the data to Cloud
        client.publish("IoT/FeetSensors", info2)

    except:
        print("Reading Error")

    with open('data.csv', 'a',newline='') as csvfile:
        # creating a csv writer object
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(data[i])

    i = i + 1
```

On cloud part of our project:

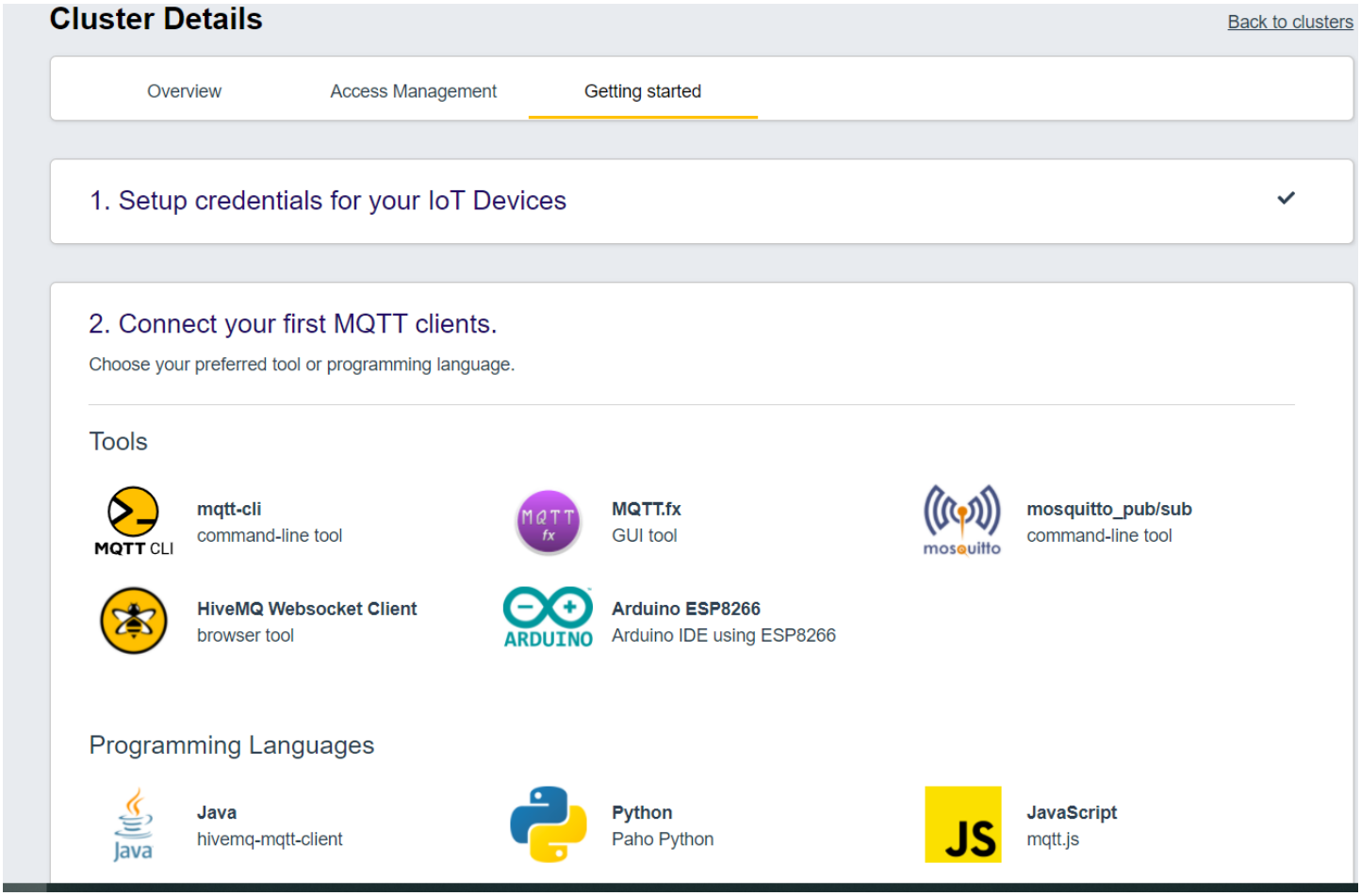
1) We opened a new MQTT server account on the HiveMQ website and created a cluster



2) We set Username and Password for connection



3) We used website's own python connection instructions to connect the server



4) We set connection settings in our code

```
##############################################################################
############## Connection to the Cloud by MQTT ###############################

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

# with this callback you can see if your publish was successful
def on_publish(client, userdata, mid, properties=None):
    print("mid: " + str(mid))

# print which topic was subscribed to
def on_subscribe(client, userdata, mid, granted_qos, properties=None):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

# print message, useful for checking if it was successful
def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload.decode("utf-8")))

# 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="", 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("MQTTPoint", "asD12^zxc")
# connect to HiveMQ Cloud on port 8883 (default for MQTT)
client.connect("40c5a4ec81ad401ba6e5748d52cc5a28.s2.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("IoT/#", qos=1)
```

5) We open infinite connection loop for sending data taken from sensors:

```
#Cloud Connection Start
client.loop_forever()
```

6) For the listener part for our cloud connection, we set connection settings

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

# with this callback you can see if your publish was successful
def on_publish(client, userdata, mid, properties=None):
    print("mid: " + str(mid))

# print which topic was subscribed to
def on_subscribe(client, userdata, mid, granted_qos, properties=None):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

# print message, useful for checking if it was successful
def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload.decode("utf-8")))

# 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="", 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("MQTTPoint", "asD12^zxc")
# connect to HiveMQ Cloud on port 8883 (default for MQTT)
client.connect("40c5a4ec81ad401ba6e5748d52cc5a28.s2.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
```

7) We subscribe to the topic and wait for released informations, so that we take any data sent to the cloud

```
# subscribe to all topics of encyclopedia by using the wildcard "#"
client.subscribe("IoT/#", 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()
```