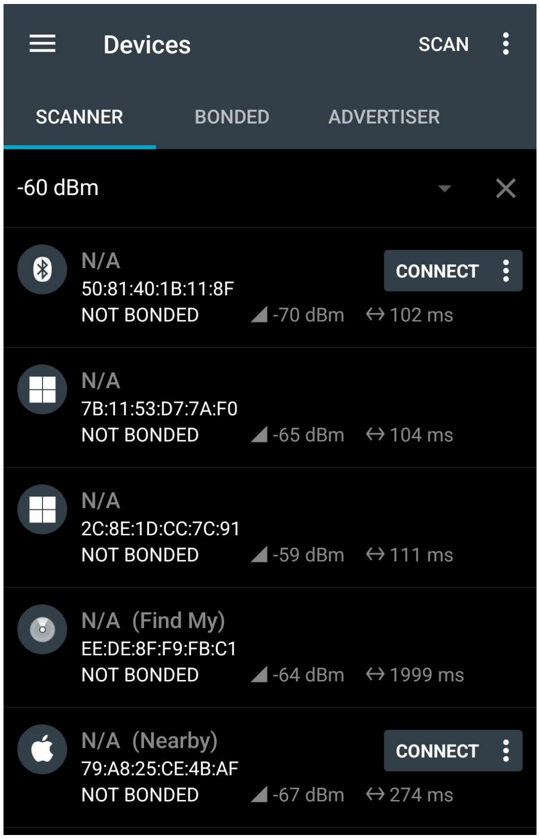
Faculty of Engineering & Applied Science

SOFE4610 - Assignment 2

Oct 19, 2022

Name	Student ID
Ivan Bisol	**100701735**
Tiwaloluwa Ojo	**100700622**

Question 1



-60 dBm

N/A
50:81:40:1B:11:8F
NOT BONDED -70 dBm ↔ 102 ms

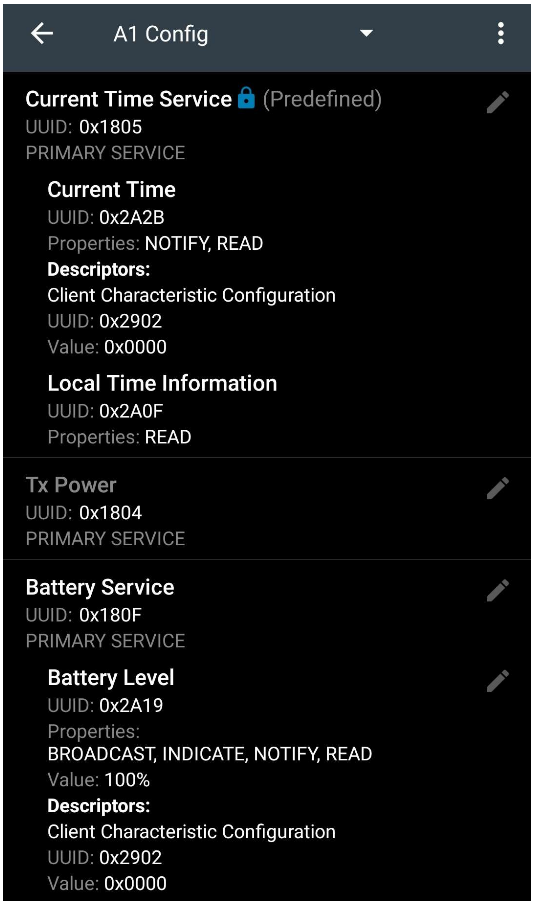
N/A
7B:11:53:D7:7A:F0
NOT BONDED -65 dBm ↔ 104 ms

N/A
2C:8E:1D:CC:7C:91
NOT BONDED -59 dBm ↔ 111 ms

N/A (Find My)
EE:DE:8F:F9:FB:C1
NOT BONDED -64 dBm ↔ 1999 ms

N/A (Nearby)
79:A8:25:CE:4B:AF
NOT BONDED -67 dBm ↔ 274 ms

Screenshot of the scanner device before setting up the GATT server and advertiser on the second device.



Current Time Service (Predefined)
UUID: 0x1805
PRIMARY SERVICE

Current Time
UUID: 0x2A2B
Properties: NOTIFY, READ
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: 0x0000

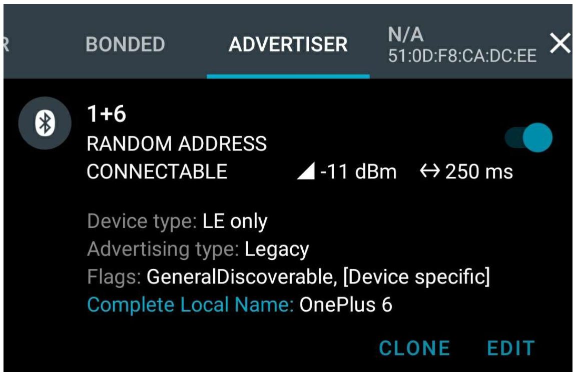
Local Time Information
UUID: 0x2A0F
Properties: READ

Tx Power
UUID: 0x1804
PRIMARY SERVICE

Battery Service
UUID: 0x180F
PRIMARY SERVICE

Battery Level
UUID: 0x2A19
Properties:
BROADCAST, INDICATE, NOTIFY, READ
Value: 100%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: 0x0000

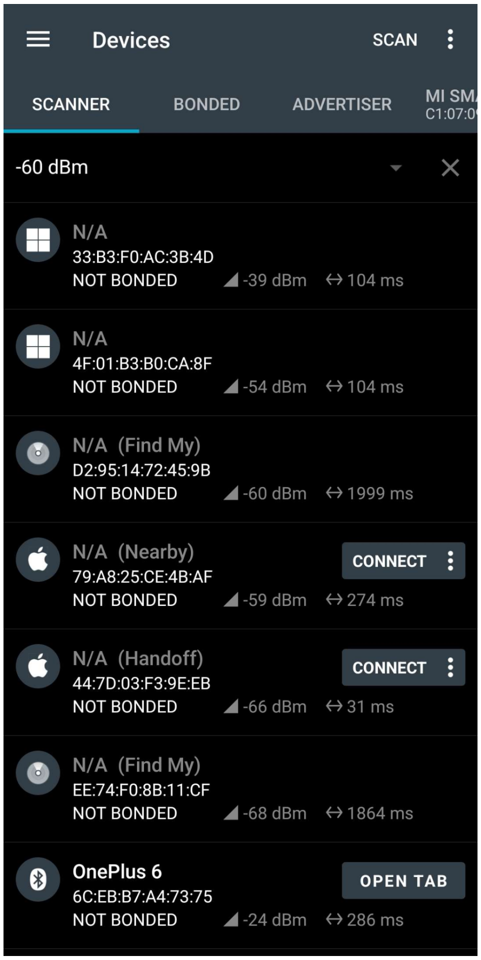
Configuration of the GATT server using a predefined current time service and one custom battery service. (Tx Power is greyed out as it is disabled)



1+6
RANDOM ADDRESS
CONNECTABLE -11 dBm ↔ 250 ms

Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, [Device specific]
Complete Local Name: OnePlus 6

Advertiser settings on server device. Advertiser wasn't required when I initially set up the GATT server during testing, but the scanner device was unable to find the server without an advertiser after the initial test.



-60 dBm

N/A
33:B3:F0:AC:3B:4D
NOT BONDED -39 dBm ↔ 104 ms

N/A
4F:01:B3:B0:CA:8F
NOT BONDED -54 dBm ↔ 104 ms

N/A (Find My)
D2:95:14:72:45:9B
NOT BONDED -60 dBm ↔ 1999 ms

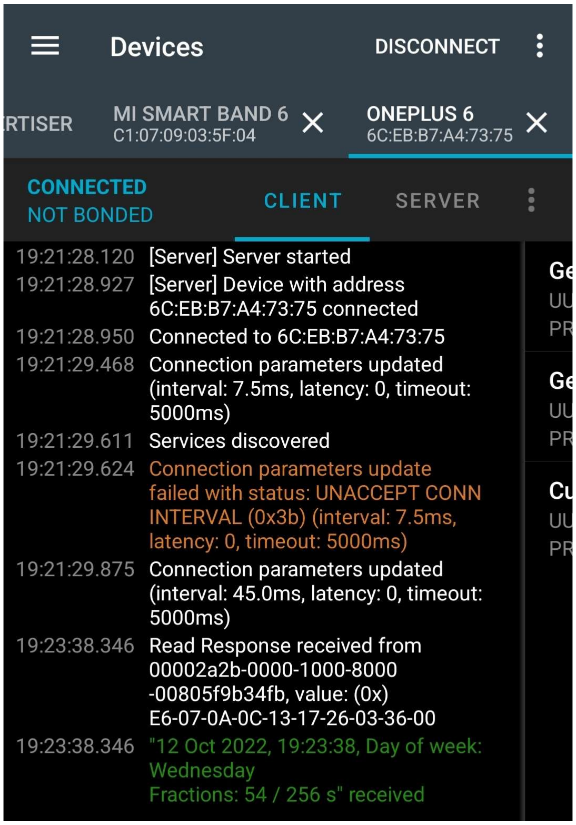
N/A (Nearby)
79:A8:25:CE:4B:AF
NOT BONDED -59 dBm ↔ 274 ms

N/A (Handoff)
44:7D:03:F3:9E:EB
NOT BONDED -66 dBm ↔ 31 ms

N/A (Find My)
EE:74:F0:8B:11:CF
NOT BONDED -68 dBm ↔ 1864 ms

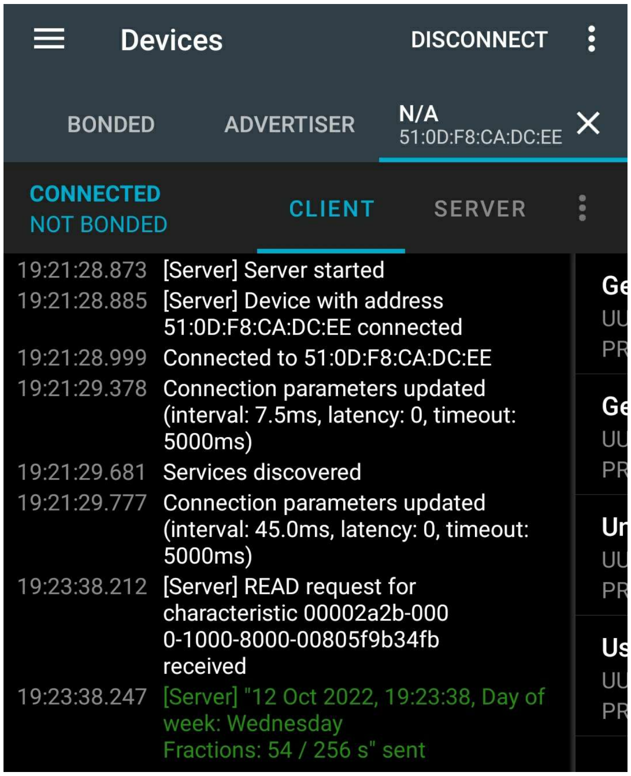
OnePlus 6
6C:EB:B7:A4:73:75
NOT BONDED -24 dBm ↔ 286 ms

List of devices after enabling the GATT server and advertiser on the server device.



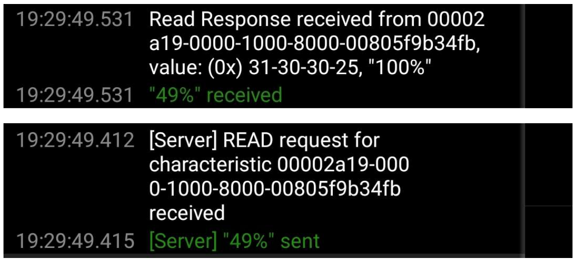
```
19:21:28.120  [Server] Server started
19:21:28.927  [Server] Device with address
              6C:EB:B7:A4:73:75 connected
19:21:28.950  Connected to 6C:EB:B7:A4:73:75
19:21:29.468  Connection parameters updated
              (interval: 7.5ms, latency: 0, timeout:
              5000ms)
19:21:29.611  Services discovered
19:21:29.624  Connection parameters update
              failed with status: UNACCEPT CONN
              INTERVAL (0x3b) (interval: 7.5ms,
              latency: 0, timeout: 5000ms)
19:21:29.875  Connection parameters updated
              (interval: 45.0ms, latency: 0, timeout:
              5000ms)
19:23:38.346  Read Response received from
              00002a2b-0000-1000-8000
              -00805f9b34fb, value: (0x)
              E6-07-0A-0C-13-17-26-03-36-00
19:23:38.346  "12 Oct 2022, 19:23:38, Day of week:
              Wednesday
              Fractions: 54 / 256 s" received
```

Log response upon requesting the current time service from the scanner device.



```
19:21:28.873  [Server] Server started
19:21:28.885  [Server] Device with address
              51:0D:F8:CA:DC:EE connected
19:21:28.999  Connected to 51:0D:F8:CA:DC:EE
19:21:29.378  Connection parameters updated
              (interval: 7.5ms, latency: 0, timeout:
              5000ms)
19:21:29.681  Services discovered
19:21:29.777  Connection parameters updated
              (interval: 45.0ms, latency: 0, timeout:
              5000ms)
19:23:38.212  [Server] READ request for
              characteristic 00002a2b-000
              0-1000-8000-00805f9b34fb
              received
19:23:38.247  [Server] "12 Oct 2022, 19:23:38, Day of
              week: Wednesday
              Fractions: 54 / 256 s" sent
```

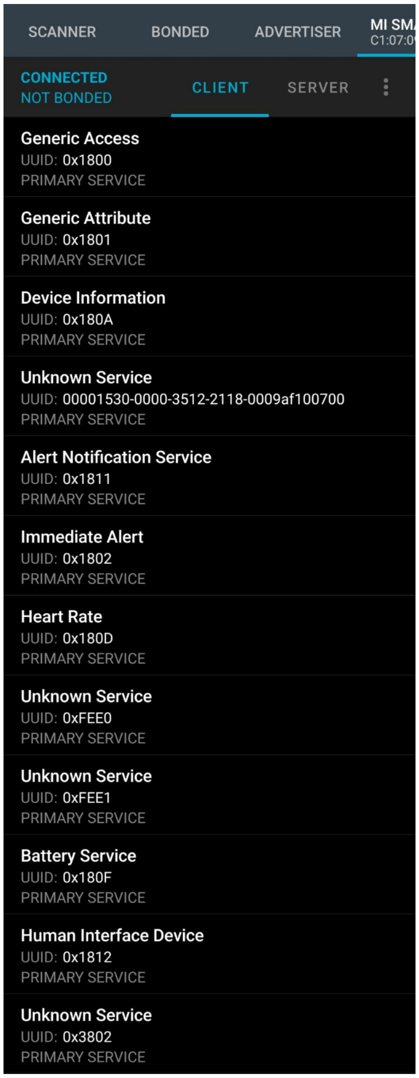
Log response from the server device upon having current time requested.



```
19:29:49.531  Read Response received from 00002
              a19-0000-1000-8000-00805f9b34fb,
              value: (0x) 31-30-30-25, "100%"
19:29:49.531  "49%" received
```

```
19:29:49.412  [Server] READ request for
              characteristic 00002a19-000
              0-1000-8000-00805f9b34fb
              received
19:29:49.415  [Server] "49%" sent
```

Log responses from client and server respectively upon requesting Battery Level. The reading of 49% is way off the device's battery level of 100% which leads me to believe either a certain byte value is required for the default value, or "Battery Level" doesn't return the actual charge of the device.



Generic Access
UUID: 0x1800
PRIMARY SERVICE

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Device Information
UUID: 0x180A
PRIMARY SERVICE

Unknown Service
UUID: 00001530-0000-3512-2118-0009af100700
PRIMARY SERVICE

Alert Notification Service
UUID: 0x1811
PRIMARY SERVICE

Immediate Alert
UUID: 0x1802
PRIMARY SERVICE

Heart Rate
UUID: 0x180D
PRIMARY SERVICE

Unknown Service
UUID: 0xFEE0
PRIMARY SERVICE

Unknown Service
UUID: 0xFEE1
PRIMARY SERVICE

Battery Service
UUID: 0x180F
PRIMARY SERVICE

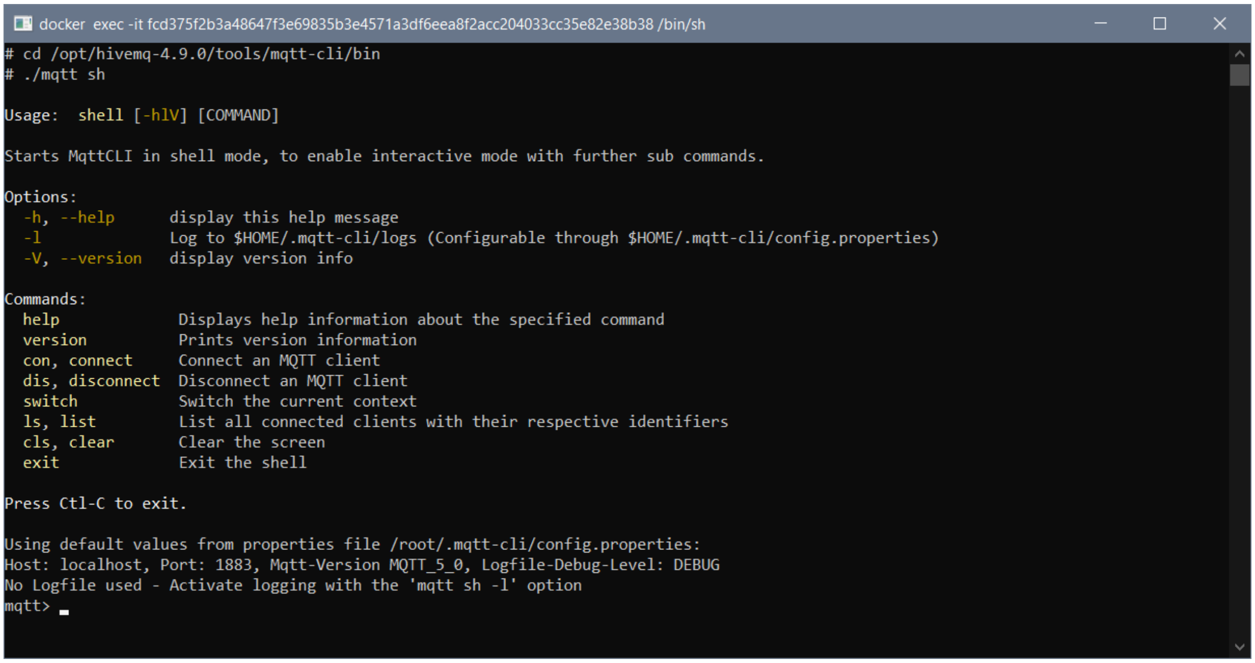
Human Interface Device
UUID: 0x1812
PRIMARY SERVICE

Unknown Service
UUID: 0x3802
PRIMARY SERVICE

Connection to my Mi Smart Band 6 shows a variety of GATT services. With a few of the services showing up as Unknown Service and a few of the descriptors returning invalid data/syntax we can only presume most of the data is sent in a proprietary/nontypical format. This is further supported by the fact that the developer's app is required to do essentially anything with/to the smart watch from my phone.

Question 2

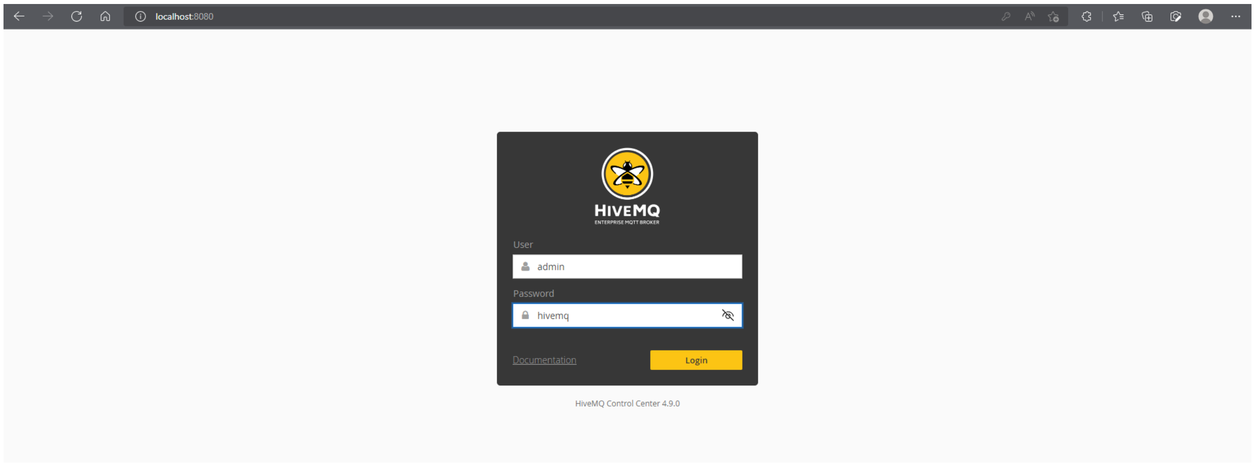
1. Set up a free MQTT broker such as HiveMQ cloud or a broker on your local machine.
    a. We set up HiveMq using the following docker quickstart. The container comes with a premade broker and cluster with a maximum of 25 client connections.
2. Use the HiveMQ MQTT CLI and HiveMQ WebSocket Client to connect to the MQTT broker to test the broker. See the steps in https://www.hivemq.com/docs/hivemq-cloud/introduction.html#guide. Exercise accessing and creating a simple test topic.
    a. Navigate to MQTT CLi tool to enter shell mode



docker exec -it fcd375f2b3a48647f3e69835b3e4571a3df6eea8f2acc204033cc35e82e38b38 /bin/sh

```
# cd /opt/hivemq-4.9.0/tools/mqtt-cli/bin
# ./mqtt sh

Usage:  shell [-hlV] [COMMAND]

Starts MqttCLI in shell mode, to enable interactive mode with further sub commands.

Options:
  -h, --help      display this help message
  -l              Log to $HOME/.mqtt-cli/logs (Configurable through $HOME/.mqtt-cli/config.properties)
  -V, --version   display version info

Commands:
  help             Displays help information about the specified command
  version          Prints version information
  con, connect     Connect an MQTT client
  dis, disconnect  Disconnect an MQTT client
  switch           Switch the current context
  ls, list         List all connected clients with their respective identifiers
  cls, clear       Clear the screen
  exit             Exit the shell

Press Ctl-C to exit.

Using default values from properties file /root/.mqtt-cli/config.properties:
Host: localhost, Port: 1883, Mqtt-Version MQTT_5_0, Logfile-Debug-Level: DEBUG
No Logfile used - Activate logging with the 'mqtt sh -l' option
mqtt> 
```

    b. Access the dashboard to access broker hostname. We can also use localhost



localhost:8080

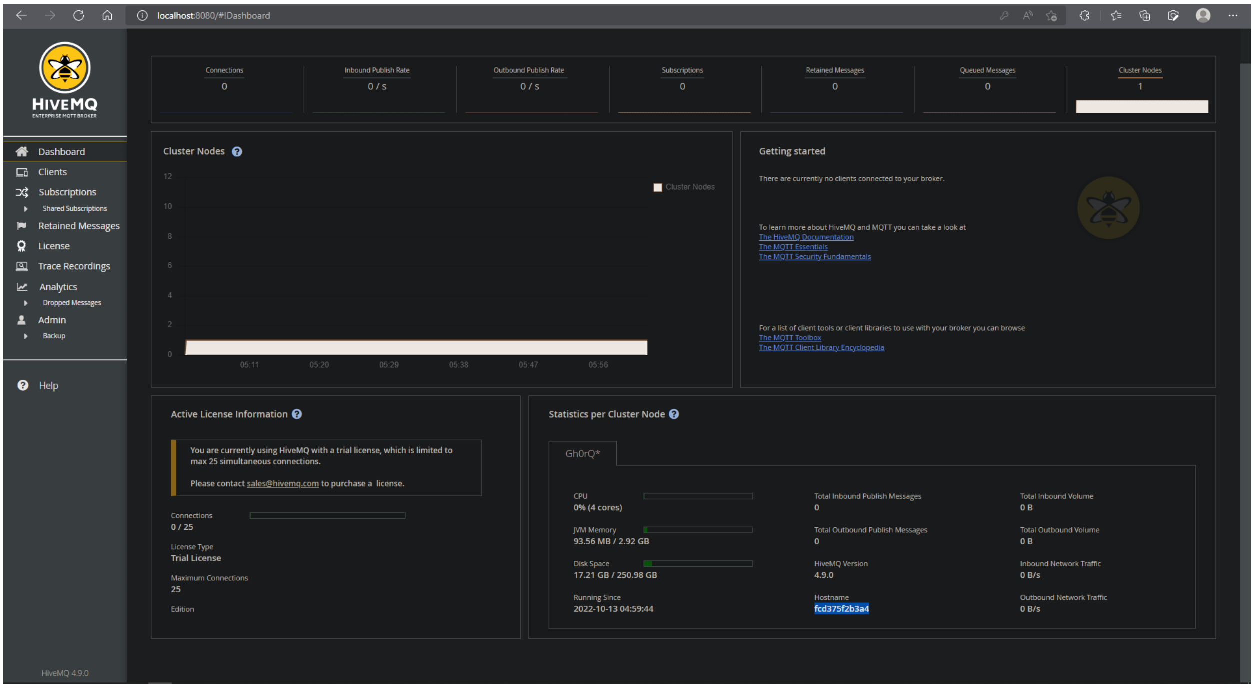
User

admin

Password

hivemq

HiveMQ Control Center 4.9.0



localhost:8080/#!Dashboard

HIVEMQ
ENTERPRISE MQTT BROKER

Dashboard
Clients
Subscriptions
Shared Subscriptions
Retained Messages
License
Trace Recordings
Analytics
Dropped Messages
Admin
Backup

Help

HiveMQ 4.9.0

Connections	Inbound Publish Rate	Outbound Publish Rate	Subscriptions	Retained Messages	Queued Messages	Cluster Nodes
0	0 / s	0 / s	0	0	0	1

Cluster Nodes

Getting started

There are currently no clients connected to your broker.

To learn more about HiveMQ and MQTT you can take a look at
The HiveMQ Documentation
The MQTT Essentials
The MQTT Security Fundamentals

For a list of client tools or client libraries to use with your broker you can browse
The MQTT Toolbox
The MQTT Client Library Encyclopedia

Active License Information

You are currently using HiveMQ with a trial license, which is limited to max 25 simultaneous connections.

Please contact sales@hivemq.com to purchase a license.

Connections
0 / 25

License Type
Trial License

Maximum Connections
25

Edition

Statistics per Cluster Node

Gh0rQ*

CPU
0% (4 cores)

JVM Memory
93.56 MB / 2.92 GB

Disk Space
17.21 GB / 250.98 GB

Running Since
2022-10-13 04:59:44

Total Inbound Publish Messages
0

Total Outbound Publish Messages
0

HiveMQ Version
4.9.0

Hostname
fcd375f2b3a4

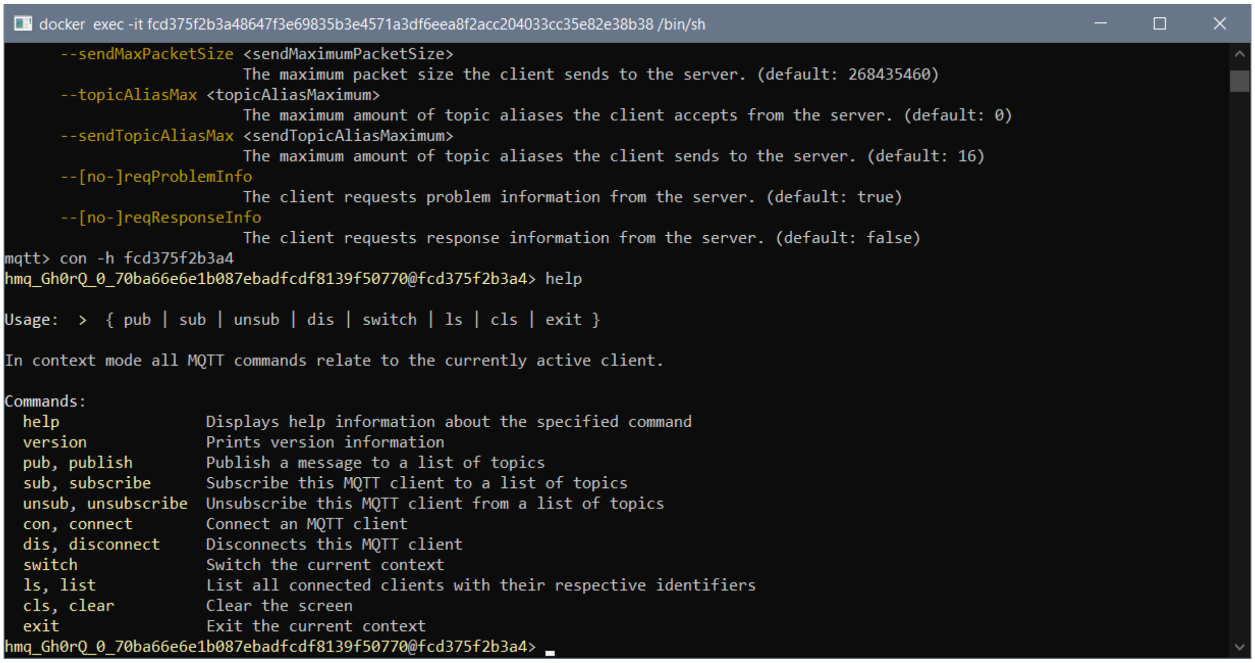
Total Inbound Volume
0 B

Total Outbound Volume
0 B

Inbound Network Traffic
0 B/s

Outbound Network Traffic
0 B/s

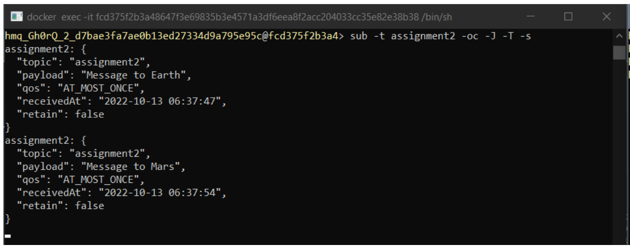
c. Connect to our cluster and enter its context



docker exec -it fcd375f2b3a48647f3e69835b3e4571a3df6eea8f2acc204033cc35e82e38b38 /bin/sh

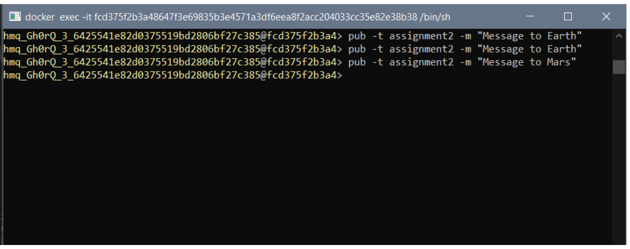
```
      --sendMaxPacketSize <sendMaximumPacketSize>
                          The maximum packet size the client sends to the server. (default: 268435460)
      --topicAliasMax <topicAliasMaximum>
                          The maximum amount of topic aliases the client accepts from the server. (default: 0)
      --sendTopicAliasMax <sendTopicAliasMaximum>
                          The maximum amount of topic aliases the client sends to the server. (default: 16)
      --[no-]reqProblemInfo
                          The client requests problem information from the server. (default: true)
      --[no-]reqResponseInfo
                          The client requests response information from the server. (default: false)
mqtt> con -h fcd375f2b3a4
hmq_Gh0rQ_0_70ba66e6e1b087ebadfcdf8139f50770@fcd375f2b3a4> help

Usage:  >  { pub | sub | unsub | dis | switch | ls | cls | exit }

In context mode all MQTT commands relate to the currently active client.

Commands:
  help                  Displays help information about the specified command
  version               Prints version information
  pub, publish          Publish a message to a list of topics
  sub, subscribe        Subscribe this MQTT client to a list of topics
  unsub, unsubscribe    Unsubscribe this MQTT client from a list of topics
  con, connect          Connect an MQTT client
  dis, disconnect       Disconnects this MQTT client
  switch                Switch the current context
  ls, list              List all connected clients with their respective identifiers
  cls, clear            Clear the screen
  exit                  Exit the current context
hmq_Gh0rQ_0_70ba66e6e1b087ebadfcdf8139f50770@fcd375f2b3a4> _
```

d. Publish & Subscribe to the /assignment2 topic



docker exec -it fcd375f2b3a48647f3e69835b3e4571a3df6eea8f2acc204033cc35e82e38b38 /bin/sh

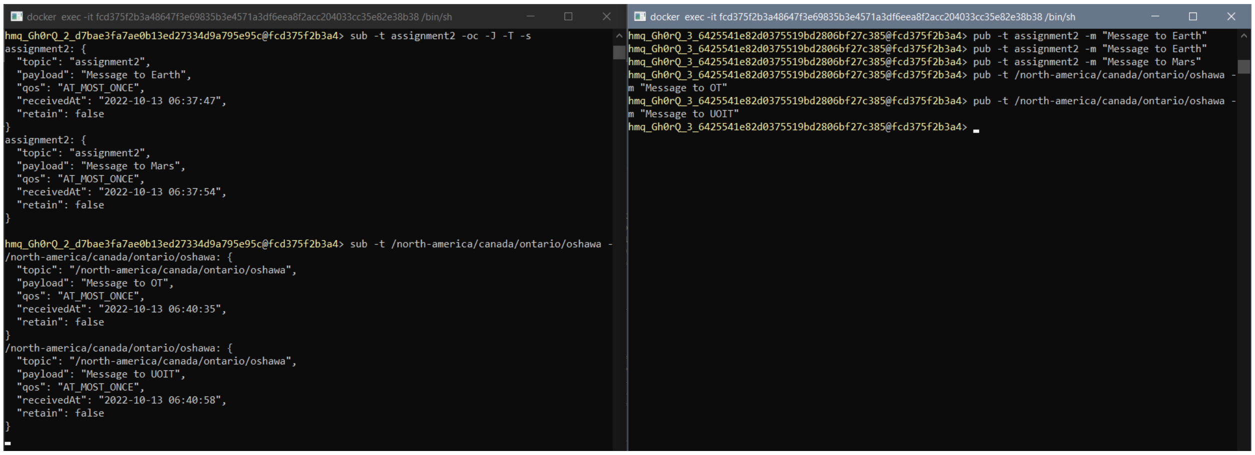
```
hmq_Gh0rQ_2_d7bae3fa7ae0b13ed27334d9a795e95c@fcd375f2b3a4> sub -t assignment2 -oc -J -T -s
assignment2: {
  "topic": "assignment2",
  "payload": "Message to Earth",
  "qos": "AT_MOST_ONCE",
  "receivedAt": "2022-10-13 06:37:47",
  "retain": false
}
assignment2: {
  "topic": "assignment2",
  "payload": "Message to Mars",
  "qos": "AT_MOST_ONCE",
  "receivedAt": "2022-10-13 06:37:54",
  "retain": false
}
_
```



docker exec -it fcd375f2b3a48647f3e69835b3e4571a3df6eea8f2acc204033cc35e82e38b38 /bin/sh

```
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t assignment2 -m "Message to Earth"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t assignment2 -m "Message to Earth"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t assignment2 -m "Message to Mars"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4>
```

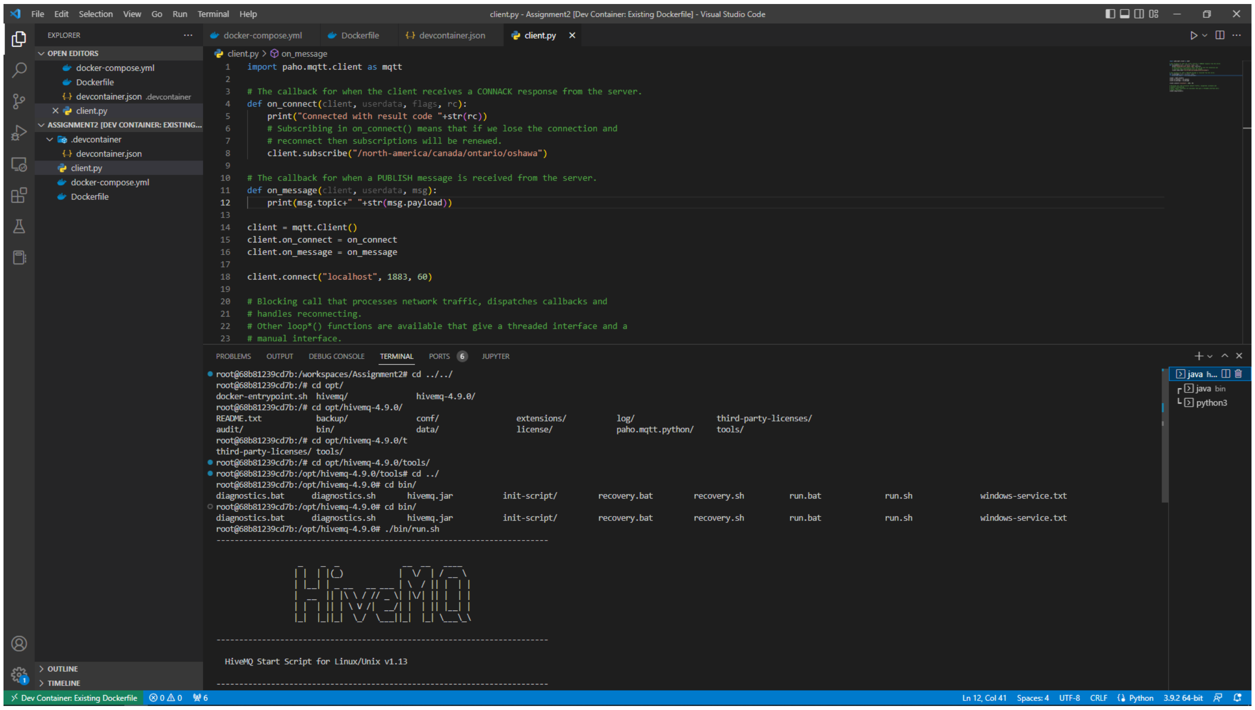
3. Create your own topic ontology (see http://www.steves-internet-guide.com/understanding-mqtt-topics/) and exercise publishing and subscribing to it.



```
docker  exec -it fcd375f2b3a48647f3e69835b3e4571a3df6eea8f2acc204033cc35e82e38b38 /bin/sh
hmq_Gh0rQ_2_d7bae3fa7ae0b13ed27334d9a795e95c@fcd375f2b3a4> sub -t assignment2 -oc -J -T -s
assignment2: {
  "topic": "assignment2",
  "payload": "Message to Earth",
  "qos": "AT_MOST_ONCE",
  "receivedAt": "2022-10-13 06:37:47",
  "retain": false
}
assignment2: {
  "topic": "assignment2",
  "payload": "Message to Mars",
  "qos": "AT_MOST_ONCE",
  "receivedAt": "2022-10-13 06:37:54",
  "retain": false
}

hmq_Gh0rQ_2_d7bae3fa7ae0b13ed27334d9a795e95c@fcd375f2b3a4> sub -t /north-america/canada/ontario/oshawa -
/north-america/canada/ontario/oshawa: {
  "topic": "/north-america/canada/ontario/oshawa",
  "payload": "Message to OT",
  "qos": "AT_MOST_ONCE",
  "receivedAt": "2022-10-13 06:40:35",
  "retain": false
}
/north-america/canada/ontario/oshawa: {
  "topic": "/north-america/canada/ontario/oshawa",
  "payload": "Message to UOIT",
  "qos": "AT_MOST_ONCE",
  "receivedAt": "2022-10-13 06:40:58",
  "retain": false
}
```

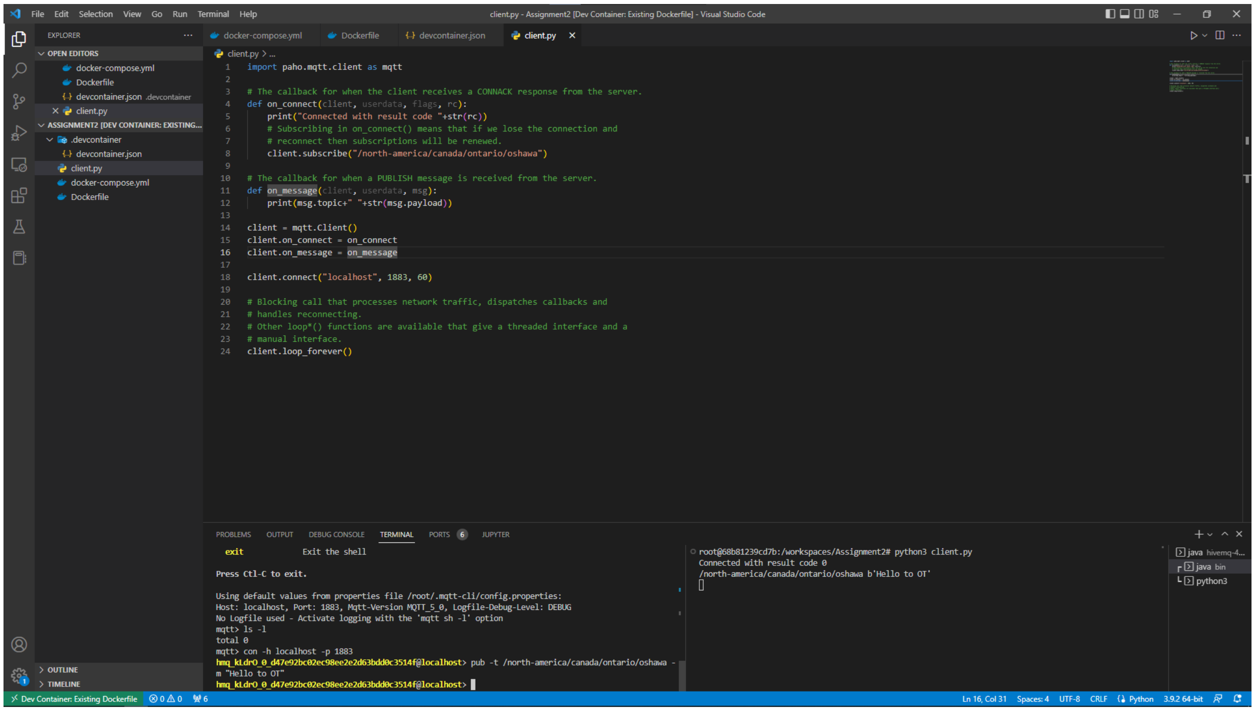
```
docker  exec -it fcd375f2b3a48647f3e69835b3e4571a3df6eea8f2acc204033cc35e82e38b38 /bin/sh
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t assignment2 -m "Message to Earth"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t assignment2 -m "Message to Earth"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t assignment2 -m "Message to Mars"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t /north-america/canada/ontario/oshawa -
m "Message to OT"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4> pub -t /north-america/canada/ontario/oshawa -
m "Message to UOIT"
hmq_Gh0rQ_3_6425541e82d0375519bd2806bf27c385@fcd375f2b3a4>
```

4. Leverage the Eclipse Paho MQTT Python client library (https://www.eclipse.org/paho/index.php?page=clients/python/index.php) to write your own client code to connect and exercise the MQTT broker you installed.


File Edit Selection View Go Run Terminal Help
client.py - Assignment2 [Dev Container: Existing Dockerfile] - Visual Studio Code
EXPLORER
OPEN EDITORS
docker-compose.yml
Dockerfile
devcontainer.json .devcontainer
client.py
ASSIGNMENT2 [DEV CONTAINER: EXISTING...
.devcontainer
devcontainer.json
client.py
docker-compose.yml
Dockerfile
OUTLINE
TIMELINE
client.py > on_message
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 6 JUPYTER
java h...
java bin
python3
Dev Container: Existing Dockerfile
Ln 12, Col 41 Spaces: 4 UTF-8 CRLF Python 3.9.2 64-bit

```python
import paho.mqtt.client as mqtt

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("/north-america/canada/ontario/oshawa")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("localhost", 1883, 60)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
```

```
root@68b81239cd7b:/workspaces/Assignment2# cd ../../
root@68b81239cd7b:/# cd opt/
docker-entrypoint.sh  hivemq/             hivemq-4.9.0/
root@68b81239cd7b:/# cd opt/hivemq-4.9.0/
README.txt            backup/             conf/               extensions/         log/                third-party-licenses/
audit/                bin/                data/               license/            paho.mqtt.python/   tools/
root@68b81239cd7b:/# cd opt/hivemq-4.9.0/t
third-party-licenses/ tools/
root@68b81239cd7b:/# cd opt/hivemq-4.9.0/tools/
root@68b81239cd7b:/opt/hivemq-4.9.0/tools# cd ../
root@68b81239cd7b:/opt/hivemq-4.9.0# cd bin/
diagnostics.bat      diagnostics.sh       hivemq.jar           init-script/         recovery.bat         recovery.sh          run.bat              run.sh               windows-service.txt
root@68b81239cd7b:/opt/hivemq-4.9.0# cd bin/
diagnostics.bat      diagnostics.sh       hivemq.jar           init-script/         recovery.bat         recovery.sh          run.bat              run.sh               windows-service.txt
root@68b81239cd7b:/opt/hivemq-4.9.0# ./bin/run.sh
-------------------------------------------------------------------------

                  _    _  _              __  __   ____
                 | |  | |(_)            |  \/  | / __ \
                 | |__| | _ __   __ ___ | \  / || |  | |
                 |  __  || |\ \ / // _ \| |\/| || |  | |
                 | |  | || | \ V /|  __/| |  | || |__| |
                 |_|  |_||_|  \_/  \___||_|  |_| \___\_\

-------------------------------------------------------------------------

  HiveMQ Start Script for Linux/Unix v1.13

-------------------------------------------------------------------------
```


File Edit Selection View Go Run Terminal Help
client.py - Assignment2 [Dev Container: Existing Dockerfile] - Visual Studio Code
EXPLORER
OPEN EDITORS
docker-compose.yml
Dockerfile
devcontainer.json .devcontainer
client.py
ASSIGNMENT2 [DEV CONTAINER: EXISTING...
.devcontainer
devcontainer.json
client.py
docker-compose.yml
Dockerfile
OUTLINE
TIMELINE
client.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 6 JUPYTER
java hivemq-4...
java bin
python3
Dev Container: Existing Dockerfile
Ln 16, Col 31 Spaces: 4 UTF-8 CRLF Python 3.9.2 64-bit

```python
import paho.mqtt.client as mqtt

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("/north-america/canada/ontario/oshawa")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("localhost", 1883, 60)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

```
  exit             Exit the shell

Press Ctl-C to exit.

Using default values from properties file /root/.mqtt-cli/config.properties:
Host: localhost, Port: 1883, Mqtt-Version MQTT_5_0, Logfile-Debug-Level: DEBUG
No Logfile used - Activate logging with the 'mqtt sh -l' option
mqtt> ls -l
total 0
mqtt> con -h localhost -p 1883
hmq_kLdrO_0_d47e92bc02ec98ee2e2d63bdd0c3514f@localhost> pub -t /north-america/canada/ontario/oshawa -
m "Hello to OT"
hmq_kLdrO_0_d47e92bc02ec98ee2e2d63bdd0c3514f@localhost>
```

```
root@68b81239cd7b:/workspaces/Assignment2# python3 client.py
Connected with result code 0
/north-america/canada/ontario/oshawa b'Hello to OT'
```