

SlOT →HW 2 →MQTT

Parisa Toumari

99101857

### **Mosquitto:**

Mosquitto is an open-source message broker that implements the MQTT (Message Queuing Telemetry Transport) protocol. MQTT is a lightweight and efficient protocol designed for low-bandwidth, high-latency, or unreliable networks, making it well-suited for communication between devices in the Internet of Things (IoT) and other scenarios where minimal overhead is crucial.

### **MQTT Protocol:**

- Publish-Subscribe Model: MQTT follows a publish-subscribe messaging pattern. Devices can publish messages to specific topics, and other devices can subscribe to those topics to receive the messages.
- Quality of Service (QoS): MQTT supports three levels of QoS (0, 1, and 2) to ensure message delivery reliability according to the application's requirements.

### **Broker:**

Mosquitto acts as a broker, which is a server that facilitates communication between devices using the MQTT protocol. It receives messages published by devices and routes them to the appropriate subscribers.

### **Topics:**

Topics are used to categorize and route messages within the MQTT system. Devices can publish messages on specific topics, and other devices can subscribe to these topics to receive relevant messages.

### **Client:**

Devices that communicate using MQTT are referred to as clients. Clients can act as either publishers, subscribers, or both. Mosquitto manages the connection and communication between these clients.

### **Security:**

Mosquitto provides security features, including support for TLS/SSL encryption to secure the communication between clients and the broker. Additionally, authentication

mechanisms, such as username/password or client certificates, can be configured to control access to the broker.

### Configuration:

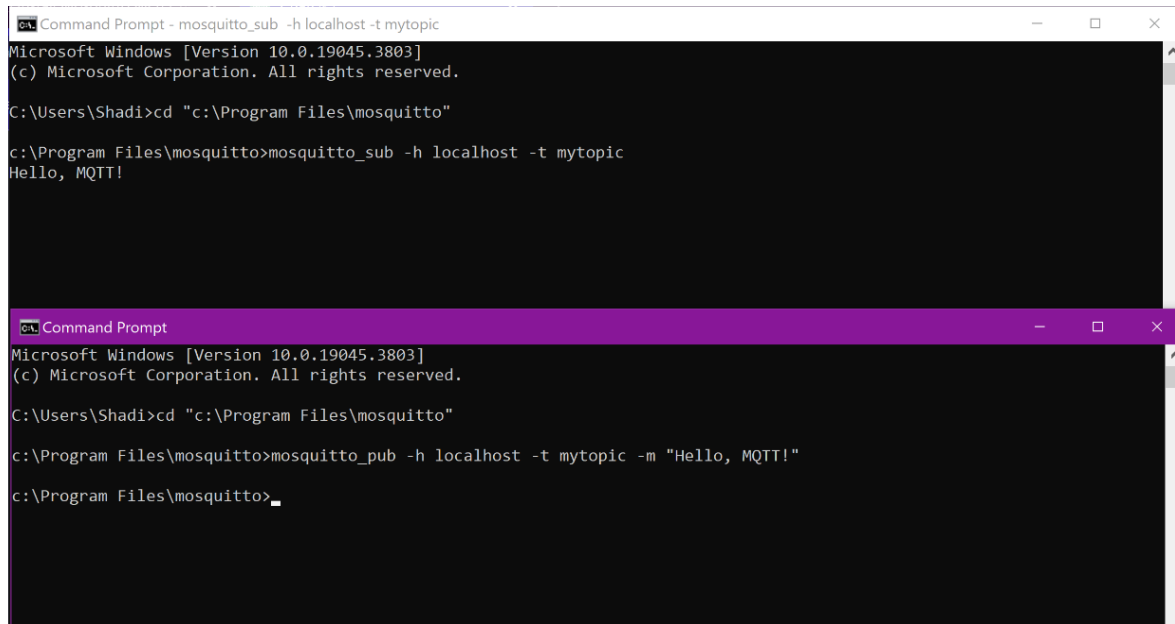
Mosquitto's configuration is typically done through a configuration file, where settings such as port numbers, security options, and logging parameters can be adjusted according to the specific requirements of the deployment.

## 1.

In Mosquitto, the commands for subscribing and publishing are performed using the `mosquitto_sub` and `mosquitto_pub` command-line tools, respectively. These tools come with the Mosquitto broker installation and allow you to interact with the MQTT broker from the command line.

```
mosquitto_sub -h <broker_address> -t <topic>
```

```
mosquitto_pub -h <broker_address> -t <topic> -m <message>
```



The image shows two screenshots of a Windows Command Prompt window. The top screenshot shows the execution of the `mosquitto_sub` command. The bottom screenshot shows the execution of the `mosquitto_pub` command.

```
ca Command Prompt - mosquitto_sub -h localhost -t mytopic
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shadi>cd "c:\Program Files\mosquitto"

c:\Program Files\mosquitto>mosquitto_sub -h localhost -t mytopic
Hello, MQTT!
```

```
ca Command Prompt
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shadi>cd "c:\Program Files\mosquitto"

c:\Program Files\mosquitto>mosquitto_pub -h localhost -t mytopic -m "Hello, MQTT!"

c:\Program Files\mosquitto>_
```

## 2.

### ● Authentication

To secure your Mosquitto broker and require authentication, we need to configure it with usernames and passwords.

First, we need to create the file, containing the usernames and passwords, using the command below:

```
ni passwd
```

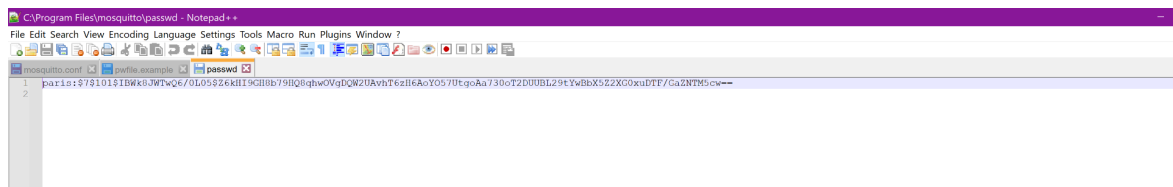
Then, we can add users and their passwords by the following command:

```
.\mosquitto_passwd.exe -c passwd username  
password
```

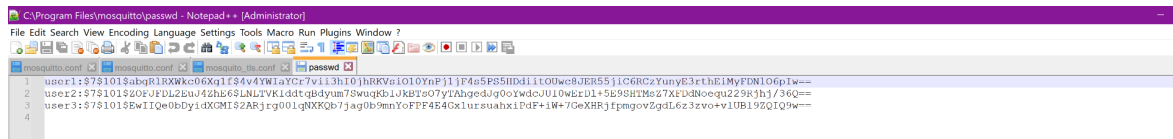
Example:

Username = paris

Password = parispass



We can add more users as well:

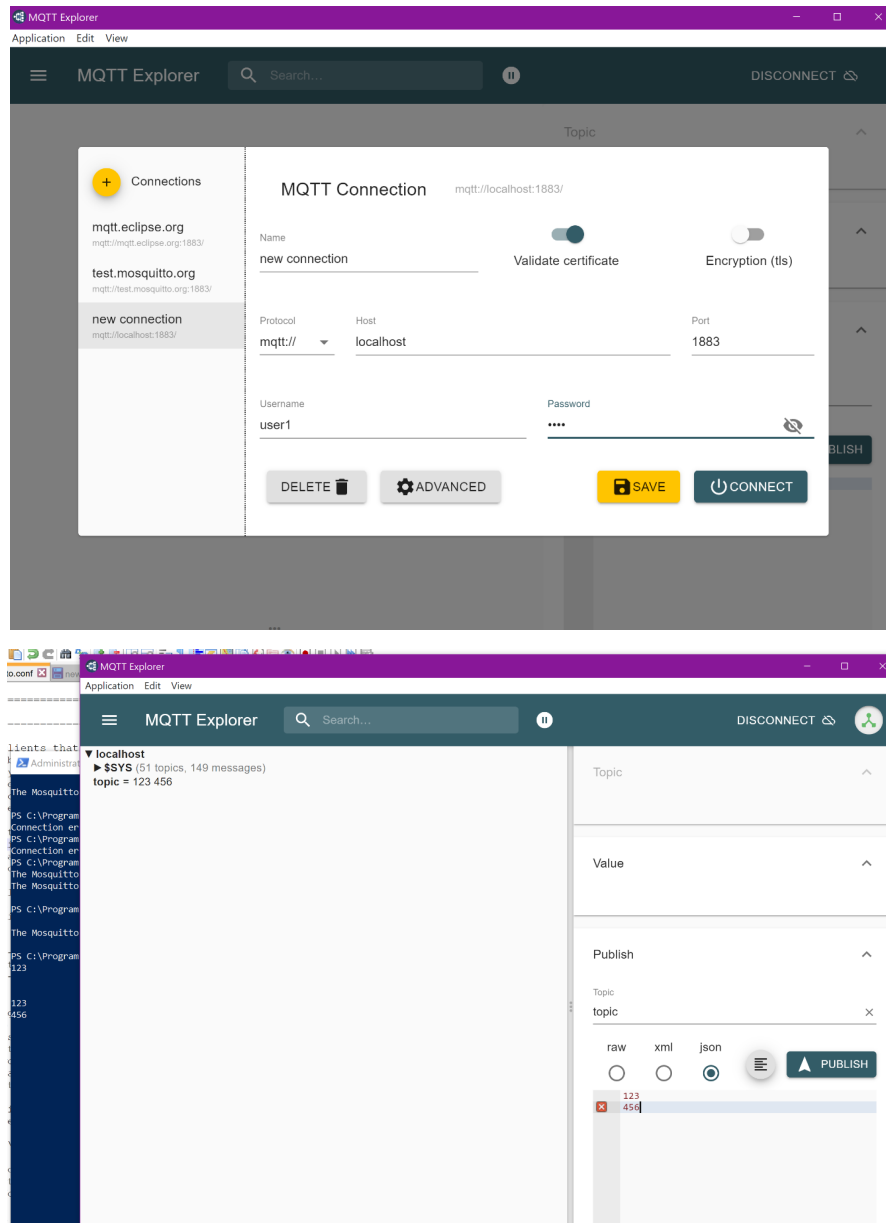


Then, we should edit some configurations in the config file.

1. Allow\_anonymus false
2. Listener 1883
3. password\_file C:\Program Files\mosquitto\passwd

Then, we need to restart the mosquitto server.

And then we can use a tool (mqtt explorer) to publish and also subscribe.



## • MQTTS

Various servers support TLS/SSL, including web servers like Apache and Nginx, which are not provided in this tutorial. These servers use TLS to secure HTTP protocol and finally communicate with browsers with HTTPS protocol. MQTT brokers like Mosquitto can also be secured with TLS (because MQTT protocol is based on TCP). In this part, we are going to make a secure broker and then test this secure broker by a MQTT client. Here is our work in brief:

Overview of steps:

1. Create a CA key pair : `openssl genrsa -des3 -out ca.key 2048`

2. Create CA certificate and use the CA key from step 1 to sign it:

```
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
```

3. Create a broker key pair:

```
openssl genrsa -out server.key 2048
```

4. Create a broker certificate request using key from step 3:

```
openssl req -new -out server.csr -key server.key
```

5. Use the CA certificate to sign the broker certificate request from step 4:

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key  
-CAcreateserial -out server.crt -days 360
```

6. Now we have a CA key file, a CA certificate file, a broker key file, and a broker certificate file.

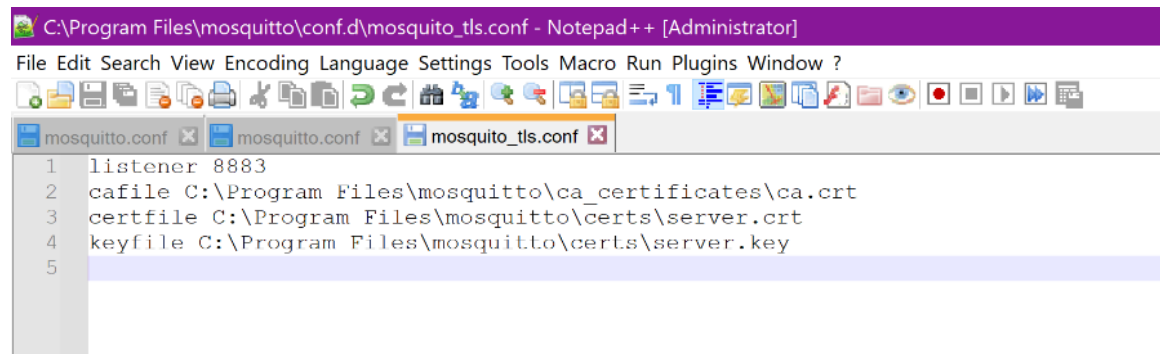
7. Place all files in a directory on the broker e.g. certs.

8. Copy the CA certificate file to the client folder.

9. Edit the Mosquitto conf file to use the files.

These are my config files:

```
90  
91 # =====  
92 # External config files  
93 # =====  
94  
95 # External configuration files may be included by using the  
96 # include_dir option. This defines a directory that will be searched  
97 # for config files. All files that end in '.conf' will be loaded as  
98 # a configuration file. It is best to have this as the last option  
99 # in the main file. This option will only be processed from the main  
100 # configuration file. The directory specified must not contain the  
101 # main configuration file.  
102 # Files within include_dir will be loaded sorted in case-sensitive  
103 # alphabetical order, with capital letters ordered first. If this option is  
104 # given multiple times, all of the files from the first instance will be  
105 # processed before the next instance. See the man page for examples.  
106 include_dir C:/Program Files/mosquitto/conf.d  
107
```



C:\Program Files\mosquitto\conf.d\mosquitto\_tls.conf - Notepad++ [Administrator]

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

mosquitto.conf x mosquitto.conf x mosquitto\_tls.conf x

```
1 listener 8883  
2 cafile C:\Program Files\mosquitto\ca_certificates\ca.crt  
3 certfile C:\Program Files\mosquitto\certs\server.crt  
4 keyfile C:\Program Files\mosquitto\certs\server.key  
5
```

And when i try to check everything, i got this error and i did EVERYTHINGGGGGGGGG to fix it but nothing worked and everytime, i got this same exact error: 😭

```
PS C:\Program Files\mosquitto> net stop mosquitto
The Mosquitto Broker service is stopping.
The Mosquitto Broker service was stopped successfully.

PS C:\Program Files\mosquitto> net start mosquitto
The Mosquitto Broker service was started successfully.

PS C:\Program Files\mosquitto> mosquitto -c mosquitto.conf -v
1704116779: Loading config file C:/Program Files/mosquitto/conf.d/mosquitto_tls.conf
1704116779: mosquitto version 2.0.18 starting
1704116779: Config loaded from mosquitto.conf.
1704116779: Opening ipv6 listen socket on port 8883.
1704116779: Error: Only one usage of each socket address (protocol/network address/port) is normally permitted
```

## ● ACL

First, we create the .acl file:

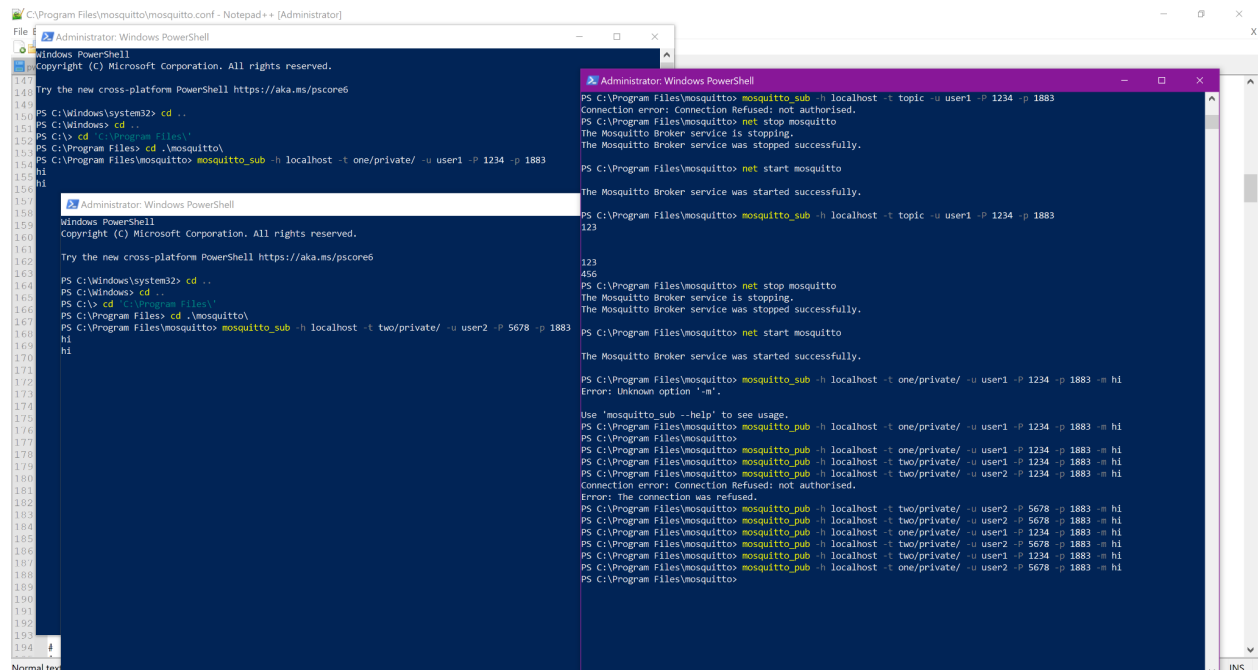
```
C:\Program Files\mosquitto\acl.acl - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
mosquitto.conf x mosquitto.conf x mosquitto_tls.conf x acl.acl x
1 user user1
2 topic readwrite one/private/#
3 topic readwrite public/#
4
5 user user2
6 topic readwrite two/private/#
7 topic readwrite public/#
8
9 user user3
10 topic readwrite three/private/#
11 topic readwrite public/#
12 topic readwrite two/private/#
13
14 pattern read two/private/#
15 pattern read public/#
16 pattern read two/private/#
17 pattern read three/private/#
```

And then we edit the config file:

```
612 #
613 # If an plugin is used as well as acl_file, the plugin check will be
614 # made first.
615 acl_file C:\Program Files\mosquitto\acl.acl
616
```

Now, we start the test:

First, we open a window for each user (as their room) and then we open an other window to publish the data:



The image shows two overlapping Windows PowerShell windows. The left window is titled 'Administrator: Windows PowerShell' and shows the following commands and output:

```
PS C:\Windows\system32> cd ..
PS C:\Windows> cd ..
PS C:\> cd 'C:\Program Files\'
PS C:\Program Files> cd .\mosquitto\
PS C:\Program Files\mosquitto> mosquitto_sub -h localhost -t one/private/ -u user1 -p 1234 -p 1883
hi
hi
```

The right window is also titled 'Administrator: Windows PowerShell' and shows the following commands and output:

```
PS C:\Program Files\mosquitto> mosquitto_sub -h localhost -t topic -u user1 -p 1234 -p 1883
Connection error: Connection Refused: not authorised.
PS C:\Program Files\mosquitto> net stop mosquitto
The Mosquitto Broker service is stopping.
The Mosquitto Broker service was stopped successfully.
PS C:\Program Files\mosquitto> net start mosquitto
The Mosquitto Broker service was started successfully.
PS C:\Program Files\mosquitto> mosquitto_sub -h localhost -t topic -u user1 -p 1234 -p 1883
123
456
PS C:\Program Files\mosquitto> net stop mosquitto
The Mosquitto Broker service is stopping.
The Mosquitto Broker service was stopped successfully.
PS C:\Program Files\mosquitto> net start mosquitto
The Mosquitto Broker service was started successfully.
PS C:\Program Files\mosquitto> mosquitto_sub -h localhost -t one/private/ -u user1 -p 1234 -p 1883 -m hi
Error: Unknown option '-m'.
Use 'mosquitto_sub --help' to see usage.
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t one/private/ -u user1 -p 1234 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t one/private/ -u user1 -p 1234 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t two/private/ -u user1 -p 1234 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t two/private/ -u user2 -p 1234 -p 1883 -m hi
Connection error: Connection Refused: not authorised.
Error: The connection was refused.
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t two/private/ -u user2 -p 5678 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t two/private/ -u user2 -p 5678 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t one/private/ -u user1 -p 1234 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t one/private/ -u user2 -p 5678 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t two/private/ -u user1 -p 1234 -p 1883 -m hi
PS C:\Program Files\mosquitto> mosquitto_pub -h localhost -t two/private/ -u user2 -p 5678 -p 1883 -m hi
```

Well, as you can see, when we publish a message by user1 in the “one” topic, we can see the message in the room “one”; but when the user2 tries to publish a message on the topic “one”, we can see anything in that room. (the opposite is also true for the topic “two”)

Also, user3, can send a message on all topics, and every user can send a message on the topic “three”.