Table des matières

1) Installation basique....1
2) Utilisation élémentaire....2
3) Captures de trames....3
4) QoS....6
5) Négociation de QoS....9
6) Retain....12
7) Last Will....14
8) Clean Session....14
9) Authentification....15

1) Installation basique

Je commencer par installer le broker MQTT :

```
test@202-15:~$ sudo apt install mosquitto
```

Installation du client :

```
test@202-15:/etc/mosquitto$ sudo apt install mosquitto-clients
```

Grâce au client je vais pouvoir passer les commandes comme mosquitto_sub ou mosquitto_pub

Il faut maintenant que je le configure dans /etc/mosquitto/ :

```
pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
```

Mon fichier de configuration inclus plusieurs fichiers situés dans plusieurs dossiers. Si je veux apporter d'autres modifications à mon broker je dois juste aller placer des fichiers .conf dans le dossier conf.d (on le voit en lisant le README) :

```
Any files placed in this directory that have a .conf ending will be loaded as
config files by the broker. Use this to make your local config.
```

Les logs du broker se situent dans /var/log/mosquitto/mosquitto.log

Les librairies du broker sont placées dans /var/lib/mosquitto/

```
test@202-15:~$ mosquitto_sub -h localhost -t yeet-piano.sh
test
test2
```

Dans un autre terminal :

```
test@202-15:~$ mosquitto_pub -h localhost -t yeet-piano.sh -m test
test@202-15:~$ mosquitto_pub -h localhost -t yeet-piano.sh -m test2
```

2) Utilisation élémentaire

Les joker permettent de remplacer un dossier dans un chemin.

Voici un exemple d'utilisation avec le joker + :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/+/dconf/
bonjour
^C
```

Dans un autre terminal :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/toto/dconf/ -m bonjour
```

Si après /toto je mettais autre chose que /dconf je ne verrai pas le message

Voici un exemple d'utilisation avec le joker # :

```
test@202-15:~$ mosquitto_sub -h localhost -t "usr/test/#"
bonjour
```

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/coucou/ -m bonjour
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/coucou/test/test -m
bonjour
```

Ici n'importe quel chemin qu'on va mettre derrière usr/test/ sera pris par le sub

3) Captures de trames

Connexion d'un client :

```
No.     Time           Source                Destination           Protocol Length Info
      4 0.000094435    ::1                   ::1                   MQTT     119    Connect
Command

MQ Telemetry Transport Protocol, Connect Command
    Header Flags: 0x10, Message Type: Connect Command
    Msg Len: 31
    Protocol Name Length: 4
    Protocol Name: MQTT
    Version: MQTT v3.1.1 (4)
    Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget),
Clean Session Flag
    Keep Alive: 60
    Client ID Length: 19
    Client ID: mosqsub|1996-202-15

No.     Time           Source                Destination           Protocol Length Info
      6 0.000272717    ::1                   ::1                   MQTT     90     Connect Ack


MQ Telemetry Transport Protocol, Connect Ack
    Header Flags: 0x20, Message Type: Connect Ack
    Msg Len: 2
    Acknowledge Flags: 0x00
    Return Code: Connection Accepted (0)
```

Sur ces deux trames de connexion d'un client on retrouve pas mal d'infos comme le client.id ou le type de QoS mais aussi le return code qui permet de dire que la connexion a bien été accepté

```
No.     Time           Source                Destination           Protocol Length Info
      8 0.000381990    ::1                   ::1                   MQTT     103    Subscribe
Request (id=1) [usr/test/#]

MQ Telemetry Transport Protocol, Subscribe Request
    Header Flags: 0x82, Message Type: Subscribe Request
    Msg Len: 15
    Message Identifier: 1
    Topic Length: 10
    Topic: usr/test/#
    Requested QoS: At most once delivery (Fire and Forget) (0)

No.     Time           Source                Destination           Protocol Length Info
```

```
    9 0.000448579    ::1              ::1              MQTT     91     Subscribe Ack
(id=1)

MQ Telemetry Transport Protocol, Subscribe Ack
   Header Flags: 0x90, Message Type: Subscribe Ack
   Msg Len: 3
   Message Identifier: 1
   Granted QoS: At most once delivery (Fire and Forget) (0)
```

Publication d'une information :

```
No.     Time           Source                Destination           Protocol Length Info
     4 0.000094951    ::1              ::1              MQTT     119    Connect
Command

MQ Telemetry Transport Protocol, Connect Command
   Header Flags: 0x10, Message Type: Connect Command
   Msg Len: 31
   Protocol Name Length: 4
   Protocol Name: MQTT
   Version: MQTT v3.1.1 (4)
   Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget),
Clean Session Flag
   Keep Alive: 60
   Client ID Length: 19
   Client ID: mosqpub|4769-202-15

No.     Time           Source                Destination           Protocol Length Info
     6 0.000312447    ::1              ::1              MQTT     90     Connect Ack

MQ Telemetry Transport Protocol, Connect Ack
   Header Flags: 0x20, Message Type: Connect Ack
   Msg Len: 2
   Acknowledge Flags: 0x00
   Return Code: Connection Accepted (0)

No.     Time           Source                Destination           Protocol Length Info
     8 0.000421702    ::1              ::1              MQTT     122    Publish
Message [usr/test/coucou/test/test]
MQ Telemetry Transport Protocol, Publish Message
   Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most
once delivery (Fire and Forget)
   Msg Len: 34
   Topic Length: 25
   Topic: usr/test/coucou/test/test
   Message: bonjour
```

```
No.     Time           Source                Destination           Protocol Length Info
     9 0.000461513    ::1                   ::1                   MQTT     88     Disconnect
Req

MQ Telemetry Transport Protocol, Disconnect Req
    Header Flags: 0xe0, Message Type: Disconnect Req
    Msg Len: 0
```

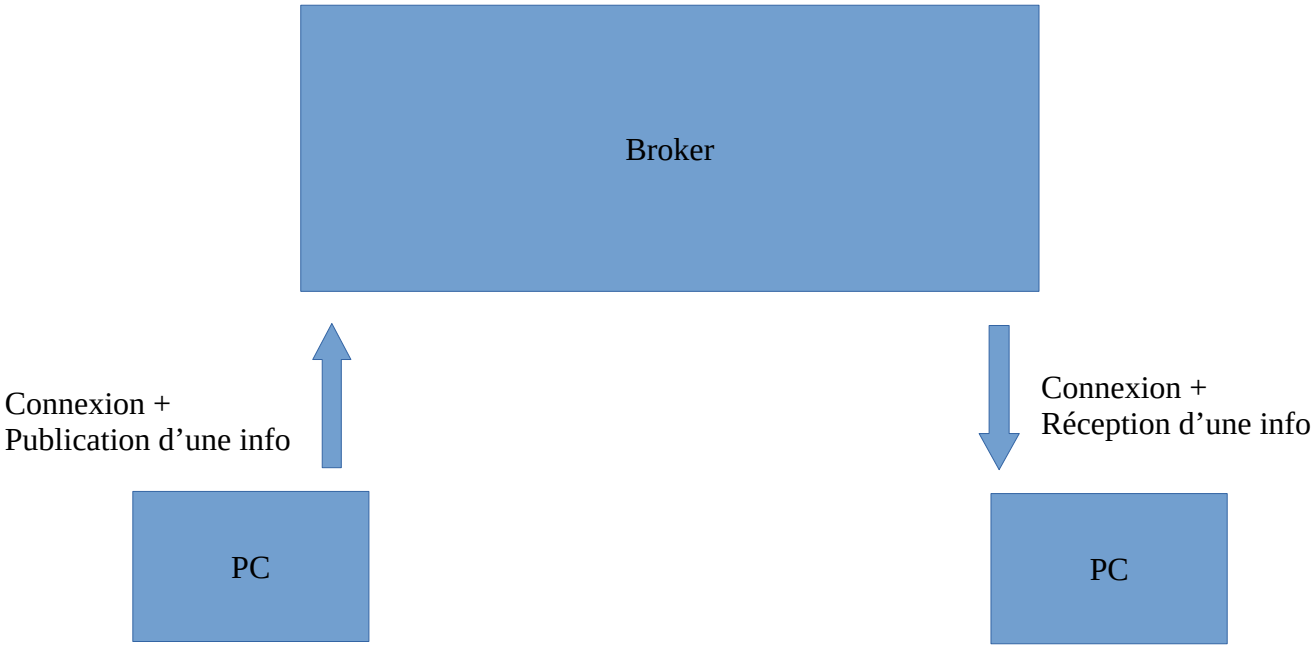
Lors d'une publication d'une information on retrouve la connexion du client vers le broker puis une fois accepté il réceptionne la donnée dans le chemin indiqué

Réception d'une information :

```
No.     Time           Source                Destination           Protocol Length Info
    10 0.000522331    ::1                   ::1                   MQTT     122    Publish
Message [usr/test/coucou/test/test]

MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most
once delivery (Fire and Forget)
    Msg Len: 34
    Topic Length: 25
    Topic: usr/test/coucou/test/test
    Message: bonjour
```

Pour la réception c'est tout simplement l'envoi de la donnée du broker vers un autre client toujours pareil avec une connexion au début

Le schéma est donc le suivant :



4) QoS

Publication d'une info dans le niveau de QoS 0 :

Dans le terminal je fais la commande suivante :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/test/
```

Maintenant dans un autre terminal je vais lancer plusieurs publish avec différentes QoS :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/ -m QoS1 -q 0
```

Pour les analyses de trames je ne met pas les trames de connexion car c'est toujours la même chose dans chaque QoS :

```
No.     Time           Source                Destination           Protocol Length Info
     8 0.000393919    ::1                   ::1                   MQTT     103    Publish
Message [usr/test/]

MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most
once delivery (Fire and Forget)
    Msg Len: 15
```

```
    Topic Length: 9
    Topic: usr/test/
    Message: QoS0
```

Ici en QoS 0 on a juste une trame d'envoi de donnée et on ne regarde même pas si le broker a reçu les infos

Publication d'une info dans le niveau de QoS 1 :

Dans le terminal :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/ -m QoS1 -q 1
```

En analyse de trames :

```
No.     Time           Source                Destination           Protocol Length Info
      8 0.000356442    ::1                   ::1                   MQTT     105    Publish
Message (id=1) [usr/test/]

MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x32, Message Type: Publish Message, QoS Level: At least
once delivery (Acknowledged deliver)
    Msg Len: 17
    Topic Length: 9
    Topic: usr/test/
    Message Identifier: 1
    Message: QoS1

No.     Time           Source                Destination           Protocol Length Info
      9 0.000384677    ::1                   ::1                   MQTT     90     Publish Ack
(id=1)

MQ Telemetry Transport Protocol, Publish Ack
    Header Flags: 0x40, Message Type: Publish Ack
    Msg Len: 2
    Message Identifier: 1
```

Cette fois ci pour la QoS 1 on a une réponse du broker qui dit qu'il a reçu l'info au moins une fois (surligné en rouge). Le message identifier permet d'identifier la donnée envoyée

Publication d'une info dans le niveau de QoS 2 :

Dans le terminal :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/ -m QoS2 -q 2
```

En analyse de trames :

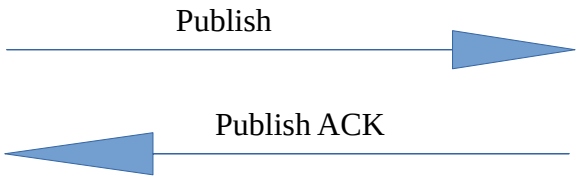
```
No.     Time           Source                Destination           Protocol Length Info
      8 0.000395668    ::1                   ::1                   MQTT     105    Publish
Message (id=1) [usr/test/]

MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x34, Message Type: Publish Message, QoS Level: Exactly
once delivery (Assured Delivery)
    Msg Len: 17
    Topic Length: 9
    Topic: usr/test/
    Message Identifier: 1
    Message: QoS2

No.     Time           Source                Destination           Protocol Length Info
      9 0.000485146    ::1                   ::1                   MQTT     90     Publish
Received (id=1)

MQ Telemetry Transport Protocol, Publish Received
    Header Flags: 0x50, Message Type: Publish Received
    Msg Len: 2
    Message Identifier: 1

No.     Time           Source                Destination           Protocol Length Info
     10 0.000578293    ::1                   ::1                   MQTT     90     Publish
Release (id=1)


MQ Telemetry Transport Protocol, Publish Release
    Header Flags: 0x62, Message Type: Publish Release
    Msg Len: 2
    Message Identifier: 1

No.     Time           Source                Destination           Protocol Length Info
     11 0.000696167    ::1                   ::1                   MQTT     90     Publish
Complete (id=1)

MQ Telemetry Transport Protocol, Publish Complete
    Header Flags: 0x70, Message Type: Publish Complete
    Msg Len: 2
    Message Identifier: 1
```

Cette fois ci pour la QoS 2 on a une réponse du broker qui dit qu'il a reçu l'info une fois et une seule (surligné en rouge). Le message identifier permet d'identifier la donnée envoyée

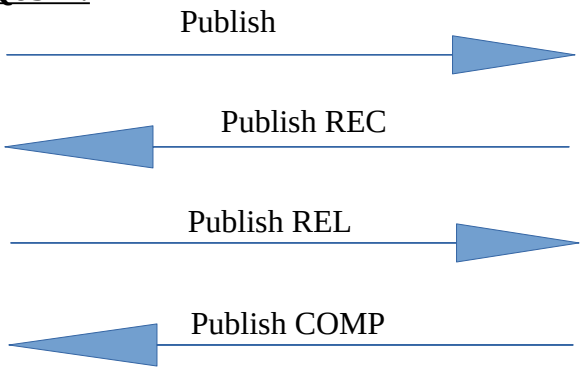
Voici donc les différences entre les trames de QoS 0, 1 ou 2 :

QoS 0 :

QoS 1 :



QoS 2 :



5) Négociation de QoS

Dans un terminal je m'abonne à un topic avec un niveau de QoS 0 :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/test/ -q 0
```

Dans un autre terminal je publie une information dans un niveau de QoS 2 sur le même topic :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/ -m QoS2 -q 2
```

Voici ce qui se passe en capture de trames :

```
No.     Time           Source                Destination           Protocol Length Info
      4 0.000094815    ::1                   ::1                   MQTT     120    Connect
Command

MQ Telemetry Transport Protocol, Connect Command
    Header Flags: 0x10, Message Type: Connect Command
```

```
    Msg Len: 32
    Protocol Name Length: 4
    Protocol Name: MQTT
    Version: MQTT v3.1.1 (4)
    Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget),
Clean Session Flag
    Keep Alive: 60
    Client ID Length: 20
    Client ID: mosqsub|10332-202-15

No.     Time           Source                Destination           Protocol Length Info
      6 0.000271805    ::1                   ::1                   MQTT     90     Connect Ack

MQ Telemetry Transport Protocol, Connect Ack
    Header Flags: 0x20, Message Type: Connect Ack
    Msg Len: 2
    Acknowledge Flags: 0x00
    Return Code: Connection Accepted (0)

No.     Time           Source                Destination           Protocol Length Info
      8 0.000380133    ::1                   ::1                   MQTT     102    Subscribe
Request (id=1) [usr/test/]

MQ Telemetry Transport Protocol, Subscribe Request
    Header Flags: 0x82, Message Type: Subscribe Request
    Msg Len: 14
    Message Identifier: 1
    Topic Length: 9
    Topic: usr/test/
    Requested QoS: At most once delivery (Fire and Forget) (0)

No.     Time           Source                Destination           Protocol Length Info
      9 0.000478251    ::1                   ::1                   MQTT     91     Subscribe Ack
(id=1)

MQ Telemetry Transport Protocol, Subscribe Ack
    Header Flags: 0x90, Message Type: Subscribe Ack
    Msg Len: 3
    Message Identifier: 1
    Granted QoS: At most once delivery (Fire and Forget) (0)
```

Pour le subscribe on retrouve bien le QoS de niveau 0

Maintenant si on fait un publish avec un QoS de niveau 2 on retrouve les trames suivantes :

```
No.     Time           Source                Destination           Protocol Length Info
```

```
    14 51.231918710   ::1             ::1             MQTT     120    Connect
Command

MQ Telemetry Transport Protocol, Connect Command
    Header Flags: 0x10, Message Type: Connect Command
    Msg Len: 32
    Protocol Name Length: 4
    Protocol Name: MQTT
    Version: MQTT v3.1.1 (4)
    Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget),
Clean Session Flag
    Keep Alive: 60
    Client ID Length: 20
    Client ID: mosqpub|10571-202-15

No.     Time        Source              Destination         Protocol Length Info
    16 51.232099625   ::1             ::1             MQTT     90     Connect Ack

MQ Telemetry Transport Protocol, Connect Ack
    Header Flags: 0x20, Message Type: Connect Ack
    Msg Len: 2
    Acknowledge Flags: 0x00
    Return Code: Connection Accepted (0)

No.     Time        Source              Destination         Protocol Length Info
    18 51.232211068   ::1             ::1             MQTT     105    Publish
Message (id=1) [usr/test/]

MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x34, Message Type: Publish Message, QoS Level: Exactly
once delivery (Assured Delivery)
    Msg Len: 17
    Topic Length: 9
    Topic: usr/test/
    Message Identifier: 1
    Message: QoS2

No.     Time        Source              Destination         Protocol Length Info
    19 51.232300201   ::1             ::1             MQTT     90     Publish
Received (id=1)

MQ Telemetry Transport Protocol, Publish Received
    Header Flags: 0x50, Message Type: Publish Received
    Msg Len: 2
    Message Identifier: 1

No.     Time        Source              Destination         Protocol Length Info
```

```
    20 51.232393047   ::1             ::1             MQTT    90    Publish
Release (id=1)

MQ Telemetry Transport Protocol, Publish Release
   Header Flags: 0x62, Message Type: Publish Release
   Msg Len: 2
   Message Identifier: 1

No.     Time          Source          Destination     Protocol Length Info
    21 51.232490375   ::1             ::1             MQTT    90    Publish
Complete (id=1)

MQ Telemetry Transport Protocol, Publish Complete
   Header Flags: 0x70, Message Type: Publish Complete
   Msg Len: 2
   Message Identifier: 1

No.     Time          Source          Destination     Protocol Length Info
    22 51.232525357   ::1             ::1             MQTT    103   Publish
Message [usr/test/]

MQ Telemetry Transport Protocol, Publish Message
   Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most
once delivery (Fire and Forget)
   Msg Len: 15
   Topic Length: 9
   Topic: usr/test/
   Message: QoS2
```

En faisant un publish avec un QoS de niveau 2 on voit qu'on retrouve un QoS de niveau 0 (il s'est mis au même niveau que le sub)

6) Retain

Le retain permet de récupérer la dernière valeur contenu dans le retain lors de la connexion d'un nouveau client. Si une donnée n'est envoyée que toutes les heures (en publish) et qu'un client se connecte au bout de 30min (en subscribe) alors il va prendre la dernière donnée enregistrée dans le retain.

Exemple avec des captures de trames :

J'envoie une donnée avec un client dans le retain :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/test/ -m retain -q 2 -r
```

```
No.     Time           Source                Destination           Protocol Length Info
      8 0.000408162    ::1                   ::1                   MQTT     103    Publish
Message [usr/test/]

MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x31, Message Type: Publish Message, QoS Level: At most
once delivery (Fire and Forget), Retain
    Msg Len: 15
    Topic Length: 9
    Topic: usr/test/
    Message: retain
```

On voit bien que mon publish a été mis dans le retain

Maintenant si avec un autre client je me subscribe je dois voir mon retain :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/test/ -q 0
retain
```

On voit bien le retain apparaître :

```
No.     Time           Source                Destination           Protocol Length Info
      8 0.000384719    ::1                   ::1                   MQTT     102    Subscribe
Request (id=1) [usr/test/]

MQ Telemetry Transport Protocol, Subscribe Request
    Header Flags: 0x82, Message Type: Subscribe Request
    Msg Len: 14
    Message Identifier: 1
    Topic Length: 9
    Topic: usr/test/
    Requested QoS: At most once delivery (Fire and Forget) (0)

No.     Time           Source                Destination           Protocol Length Info
      9 0.000483620    ::1                   ::1                   MQTT     91     Subscribe Ack
(id=1)

MQ Telemetry Transport Protocol, Subscribe Ack
    Header Flags: 0x90, Message Type: Subscribe Ack
    Msg Len: 3
    Message Identifier: 1
    Granted QoS: At most once delivery (Fire and Forget) (0)

No.     Time           Source                Destination           Protocol Length Info
     11 0.043631259    ::1                   ::1                   MQTT     105    Publish
Message [usr/test/]
```

```
MQ Telemetry Transport Protocol, Publish Message
    Header Flags: 0x31, Message Type: Publish Message, QoS Level: At most
once delivery (Fire and Forget), Retain
    Msg Len: 17
    Topic Length: 9
    Topic: usr/test/
    Message: retain
```

7) Last Will

Le LastWill permet d'envoyer les dernières volontés d'un client.

Les champs "LastWill" contiennent :

- Un topic

- Le niveau de QoS

- Un message

- Un retain

On peut le voir en faisant les commandes suivantes :

Dans un terminal :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/test/mort/ -q 2
```

Dans un autre terminal (le sub avec ses dernières volontés) :

```
test@202-15:~$ mosquitto_sub -h localhost --will-payload "mort" --will-qos 2 --
will-topic usr/test/mort/ -t usr/test/
```

Si je ferme cette connexion je verrais dans l'autre terminal mon message de mort :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/test/mort/ -q 2
mort
```

8) Clean Session

Le clean session permet de stocker les messages qui arrivent pendant un publish quand le client (avec son id) n'est pas connecté et ainsi les voir quand il se connecte :

Je me connecte une fois à un topic avec un id que je me donne (ici test) :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/id/ -q 1 -c --id test
^C
```

Maintenant que ma connexion est arrêtée je vais envoyer plusieurs messages dans ce même topic mais avec un client différent :

```
test@202-15:~$ mosquitto_pub -h localhost -t usr/id/ -q 1 -m test
test@202-15:~$ mosquitto_pub -h localhost -t usr/id/ -q 1 -m bonjour
```

Maintenant si je me reconnecte avec mon client qui a l'id test je dois voir tous les messages envoyés pendant que je n'étais pas connecté :

```
test@202-15:~$ mosquitto_sub -h localhost -t usr/id/ -q 1 -c --id test
bonjour
test
^C
```

9) Authentification

Je commence par me créer un fichier pour contenir les mots de passe avec le mot de pass pour un utilisateur :

```
test@202-15:~$ sudo mosquitto_passwd -c /etc/mosquitto/passwd test
Password:
Reenter password:
```

Maintenant dans le fichier de configuration de mosquitto je dois donner le chemin vers le fichier de mot passe et accepter la connexion :

```
password_file /etc/mosquitto/passwd
allow_anonymous false
```

Je relance mosquitto :

```
test@202-15:~$ sudo systemctl restart mosquitto
```

Maintenant je n'ai plus qu'à me connecter avec un client :

```
test@202-15:~$ mosquitto_sub -u test -P test -h localhost -t usr/id/
```

-u pour donner le user et -P pour donner le mot de passe