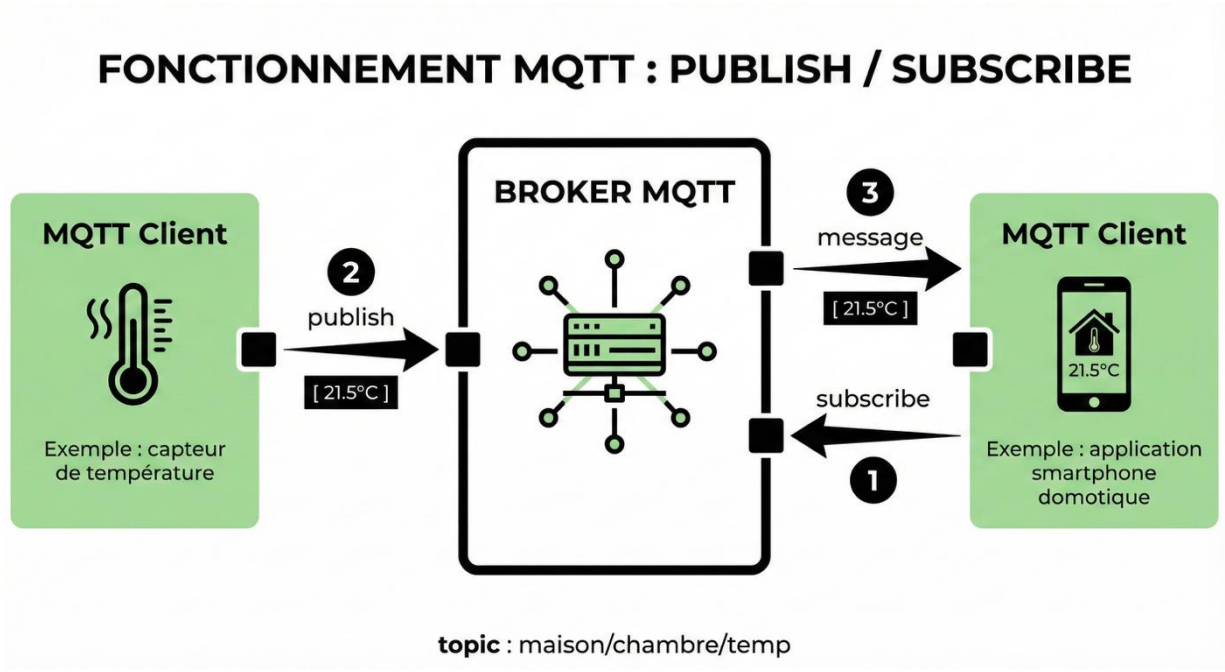
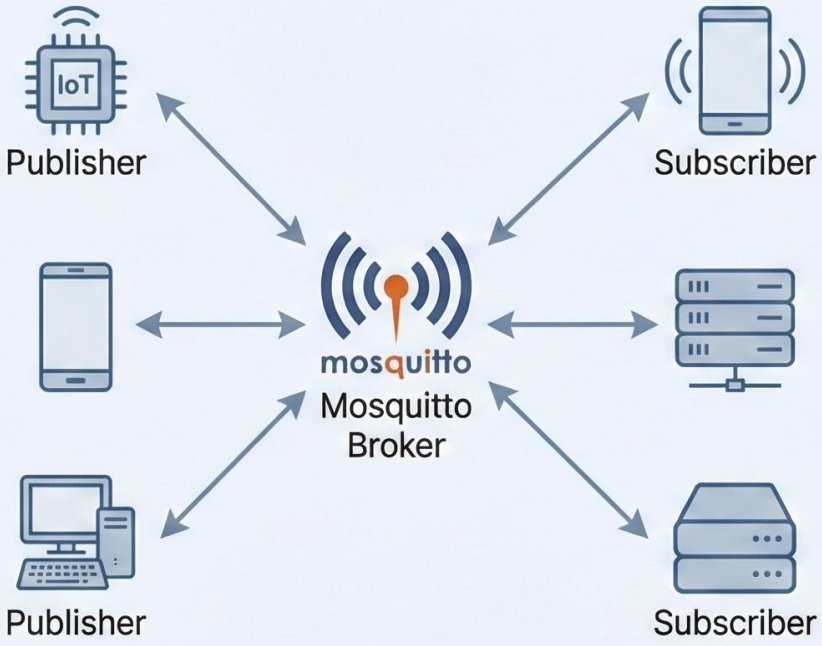
PROTOCOLE MQTT



Le protocole MQTT est assez basique dans son fonctionnement, c'est d'ailleurs sa force. Il s'agit d'un protocole de messagerie machine to machine.

Concrètement les équipements connectés publient et/ou s'abonnent à un **topic** qui référence les messages et les communique aux abonnés.

MOSQUITTO MQTT: EXPLICATION & INSTALLATION



QU'EST-CE QUE MOSQUITTO ?

Un courtier MQTT léger et open-source.
Permet la communication machine-à-machine (IoT) via des messages publiés/souscrits.

COMMENT L'INSTALLER (Linux/apt) ?

```
sudo apt install -y mosquitto mosquitto-clients
```

Installe le serveur Mosquitto (broker) et les outils clients en ligne de commande.

COMMENT LE LANCER ?

```
systemctl start mosquitto
```

Démarre le service Mosquitto en arrière-plan.

1. LA COMMANDE SUBSCRIBE

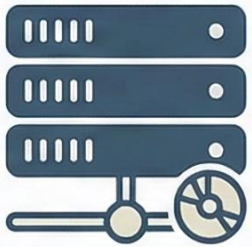
```
mosquitto_sub -h localhost -t your/topic
```

2. LA COMMANDE PUBLISH

```
mosquitto_pub -h localhost -t your/topic
-m "Hello MQTT"
```

Subscribe to “your/topic”

Subscription Confirmed



Publish to “your/topic”

Hello MQTT

Client Abonnés
(Subscriber)

MQTT Broker
(localhost)

Client Éditeur
(Publisher)

1. Client s'abonne pour recevoir des messages sur un topic.

Hello MQTT

Broadcast Message

3. LE MESSAGE

3. Le broker diffuse le message à tous les abonnés du topic.

2. Client publie un message sur un topic spécifique.

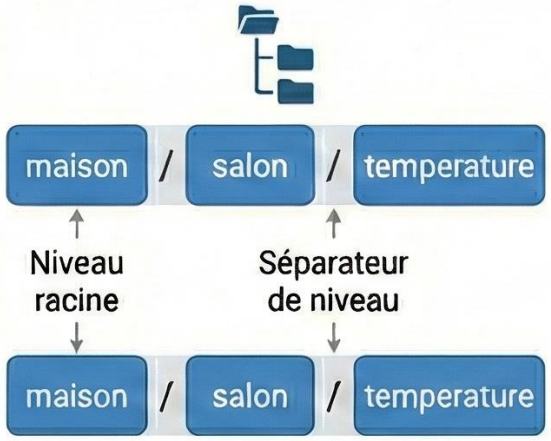
LES TOPICS

Les topics sont structurés de manière hiérarchique avec des niveaux séparés par des barres obliques ("/"), ce qui permet une organisation similaire à celle des chemins dans un système de fichiers.
Par exemple : "maison/salon/temperature"

COMPRENDRE LES WILDCARDS MQTT : + ET

Maîtriser les abonnements (Subscribe) flexibles dans l'IoT

1 RAPPEL : STRUCTURE D'UN TOPIC



2 LE WILDCARD MONO-NIVEAU : +

Remplace exactement UN niveau de l'arborescence.

Abonnement (Subscribe)

```
$ sub maison/+/temperature
```

maison/ + /temperature

Résultats de capture

✅ **Ça matche :**

- maison/**salon**/**temperature**
- maison/**cuisine**/**temperature**

❌ **Ça ne matche pas :**

- maison/**salon/humidite**
  (La fin ne correspond pas)
- maison/**salon/etagere1/temperature**
  (Erreur : 2 niveaux au lieu d'un seul)

3 LE WILDCARD MULTI-NIVEAUX :

Capture TOUT le reste de l'arborescence. Doit être à la fin.

Abonnement (Subscribe)

```
$ sub maison/salon/#
```

Résultats de capture

✅ **Ça matche :**

- maison/**salon**/**temperature**
- maison/**salon/lumieres/etat**
- maison/**salon/lumieres/plafond/couleur**

❌ **Ça ne matche pas :**

- maison/**cuisine/temperature**
  (Le début ne correspond pas)

4 PIÈGES À ÉVITER ⚠

Erreur 1 : Le # mal placé (Syntaxe Invalide)

```
$ sub maison/#/temperature
```

INTERDIT. Le # doit obligatoirement être le dernier caractère du topic.

Erreur 2 : L'abonnement racine (Performance)

```
$ sub #
```

DANGEREUX. S'abonne à tous les messages du broker.
Risque de surcharge (le "firehose").

Astuce : Soyez aussi spécifique que possible pour économiser la bande passante !

PROTOCOLE MQTT

C’est un protocole applicatif. Il fonctionne sur TCP. Il utilise le port 1883 et 8883 pour le mqtts.

ANALYSE MQTT DANS WIRESHARK

No.	Time	Source	Destination	Protocol	Info
44	1…	192.168.1.39	3.68.58.135	MQTT	Connect Command
46	1…	3.68.58.135	192.168.1.39	MQTT	Connect Ack
48	1…	192.168.1.39	3.68.58.135	MQTT	Subscribe Request
50	1…	3.68.58.135	192.168.1.39	MQTT	Subscribe Ack

Le **client MQTT** initie la connexion TCP au broker.
Frame CONNECT = ouverture de session MQTT.
Ports : client (64301) -> 1883

Le broker accepte la connexion MQTT.
Frame CONNACK = authentification MQTT validée

Le client demande l’abonnement à un topic MQTT.
SUBSCRIBE = enregistrement du client au flux choisi

Le broker confirme l’abonnement.
SUBACK = confirmation d’abonnement au topic

ANALYSE TECHNIQUE DES COUCHES RÉSEAU

```
> Frame 44: 62 bytes on with dith, 128 bytes (03 vensit on 0
> Ethernet II, Src: Ethernet I (04:00:0f:0) .adv12:02:06:58), ports:204(imef:61:08:1b)
> IPv4, Foct (0vf8.1024:23:0), src: 192.168.1.39
> TCP, TCP transforn (192.168.1.39, 3.68.58.135 .1883)
> MQTT
```

PROTOCOLE MQTT

ANALYSE MQTT : PING KEEPALIVE DANS WIRESHARK

No.	Time	Source	Destination	Protocol	Info
234	40.759562	192.168.1.39	3.66.104.219	MQTT	Ping Request
238	40.775686	3.66.104.219	192.168.1.39	MQTT	Ping Response

- **PINGREQ** envoyé par le client MQTT.
- Message de keepalive pour vérifier si le broker est actif.
- Empêche la session TCP/MQTT d'être fermée pour inactivité.

Ping Request

- **PINGRESP** envoyé par le broker MQTT.
- Accuse réception du keepalive.
- Connexion MQTT confirmée comme active.

Échange automatique, transparent pour l'utilisateur.

DÉTAILS TECHNIQUES (EXTRAITS DE FRAME 238)

Ethernet II	IPv4	TCP	MQTT Layer
Src: 20:66:cf:68:f2:a0 Dst: 78:4f:43:6c:f5:bb	Src: 3.66.104.219 Dst: 192.168.1.39	Src Port: mqtt (1883) Dst Port: 64204 | Seq: 1, Ack: 3	Infos Ping Request et Ping Response

PROTOCOLE MQTT

Analyse MQTT : Connexion + Publication + Déconnexion

No.	Time	Source	Destination	Protocol	Info
79	1…	192.168.1.39	3.68.58.135	MQTT	Connect Command
81	1…	3.68.58.135	192.168.1.39	MQTT	Connect Ack
83	1…	192.168.1.39	3.68.58.135	MQTT	Publish Message
84	1…	192.168.1.39	3.68.58.135	MQTT	Disconnect Req

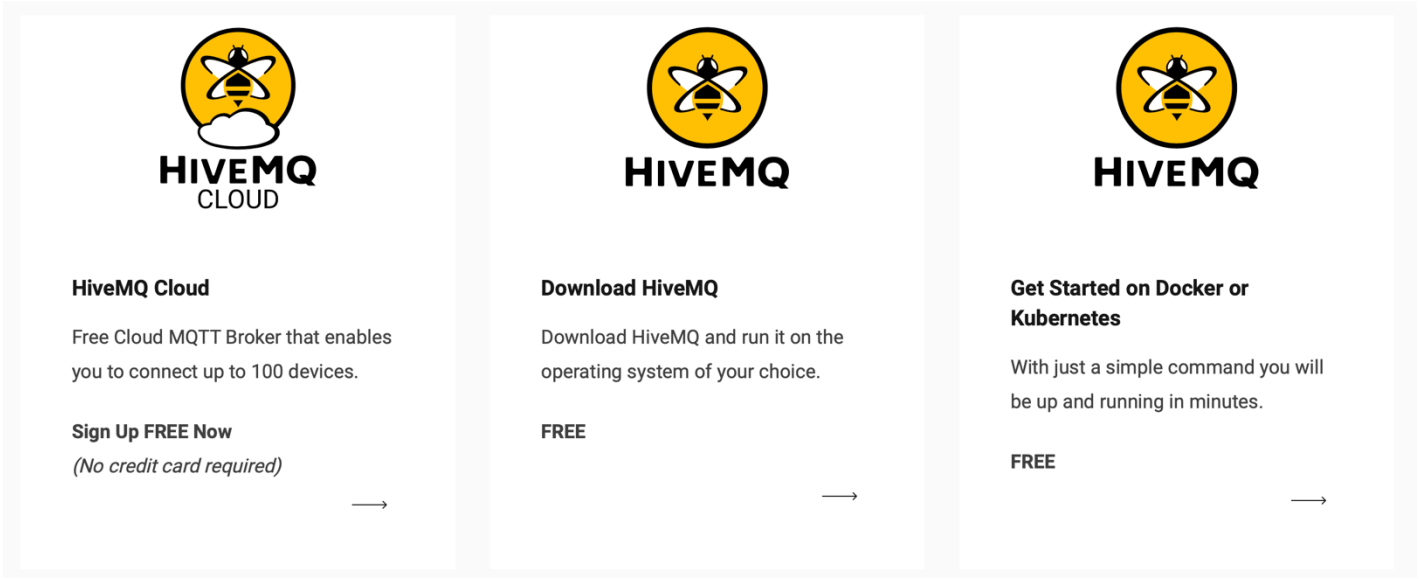
Connexion		Publication	Déconnexion
			
1 Frame 79 — Connect Command	**2 Frame 81 — Connect Ack**	**3 Frame 83 — Publish Message**	**4 Frame 84 — Disconnect Req**
Le client ouvre une session MQTT avec le broker. **CONNECT** initie l'établissement de session. **Ports TCP** : 64538 → 1883	Le broker accepte la connexion MQTT. **CONNACK** = authentification + session validée	Le client publie un message MQTT. **Frame PUBLISH** : envoi vers un topic du broker. **Sens** : client → broker	Client met fin proprement à la session MQTT. **DISCONNECT** = fermeture volontaire, sans erreur

CLIENT → **CONNECT** → **CONNACK** → **PUBLISH** → **DISCONNECT** → **BROKER**

- **Ethernet II**: Src: 78:4f:43:6c:f5:bb, Dst: 20:66:cf:68:f2:a0
- **IPv4**: Src: 192.168.1.39, Dst: 3.68.58.135
- **TCP**: Src Port: 64538, Dst Port: mqtt (1883), Seq: 1, Ack: 1, Len: 14
- **MQTT Layer: CONNECT, CONNACK, PUBLISH, DISCONNECT**

HIVEMQ

HiveMQ est une plateforme de messagerie MQTT conçue pour faciliter la communication entre appareils dans les applications Internet des Objets (IoT). Elle fournit un broker MQTT hautement évolutif et performant, capable de gérer des millions de messages et des milliers de clients simultanément.



Plusieurs solutions possibles :

- Utiliser **broker.hivemq.com** pour faire des tests (c'est sans inscription).
- S'inscrire sur le cloud
- Télécharger l'application et l'installer sur une machine.
- Utiliser une image Docker.

AUTRES OPTIONS MOSQUITTO

mosquitto_pub –h <broker> -t your/topic –m "message" -r

-r : Cette option est utilisée pour retenir le message. Lorsqu'un message est retenu, le dernier message publié sur un topic sera stocké par le broker et immédiatement envoyé à tout client qui s'abonne à ce topic.

mosquitto_rr –h <broker> -t your/topic –e other/topic –m "message"

La commande mosquitto_rr est une commande du client MQTT Mosquitto qui est utilisée pour envoyer un message MQTT et attendre une réponse, ce qui est un modèle de communication request-response (requête-réponse).

-e other/topic : L'option -e spécifie le topic d'écoute pour la réponse.

UN PEU DE SÉCURITÉ

Authentification avec mot de passe :

```
mosquitto_pub –h <broker> -u <user> -P <password> -t your/topic –m "message"
```

Connexion chiffrée avec le mqtts (port 8883) avec certificat signé par CA :

```
mosquitto_pub –h <broker>  -p 8883  -t your/topic –m "message"
```

Connexion chiffrée avec le mqtts (port 8883) avec certificat auto-signé:

```
mosquitto_pub -h <broker> -p 8883 --cafile /ca.crt -t your/topic -m "message"
```

Connexion chiffrée avec le mqtts (port 8883) avec certificat auto-signé et authentification avec certificat client :

```
mosquitto_pub -h 172.27.0.2 -p 8883 --cafile /ca.crt --cert /client.crt --key /client.key  -t "your/topic" -m "message"
```