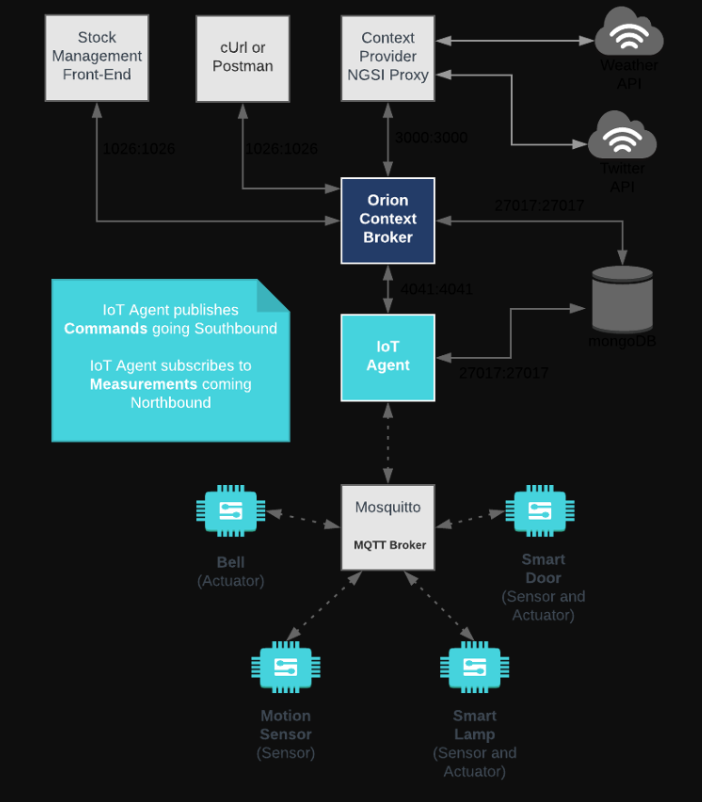
**IOT over MQTT**

* !!!!! Ce document contient une réalisation pratique des cmds et les résultats obtenus
* RQ : Ces étapes sont disponibles sur le site suivant :

<https://github.com/Fiware/tutorials.IoT-over-MQTT>

* Les parties principales de ce doc sont :
  + Définition de MQTT et IOT Agent
  + [Mise en service an IoT Agent](https://github.com/Fiware/tutorials.IoT-over-MQTT#provisioning-an-iot-agent-ultralight-over-mqtt)
  + [Connection des IoT Devices](https://github.com/Fiware/tutorials.IoT-over-MQTT#connecting-iot-devices)
  + [Activation des commandes de Context Broker](https://github.com/Fiware/tutorials.IoT-over-MQTT#enabling-context-broker-commands)

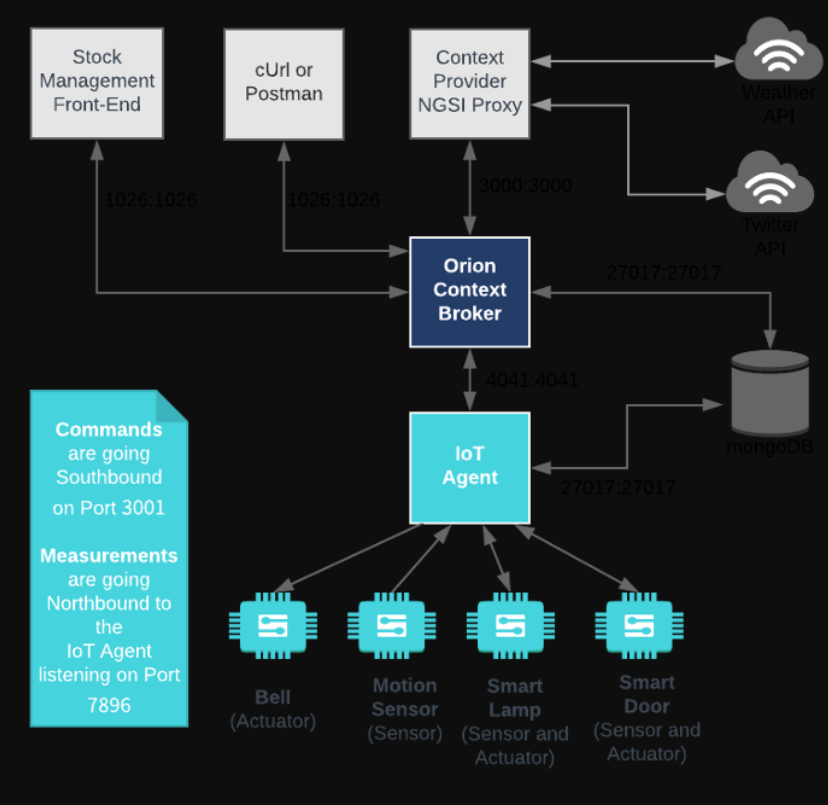
# **Qu'est ce que le MQTT?**



MQTT est un protocole de messagerie basé sur un abonnement de publication utilisé dans l'Internet des objets. Il fonctionne au dessus du protocole TCP / IP et est conçu pour les connexions avec des sites distants où une "empreinte de code réduite" est requise ou la bande passante du réseau est limitée. L'objectif est de fournir un protocole efficace en termes de bande passante et utilisant peu de batterie.

|  |
| --- |
| <https://camo.githubusercontent.com/fcf9dfb149b48b5860c0187f04f6e15f31d3c7d6/68747470733a2f2f6669776172652e6769746875622e696f2f7475746f7269616c732e496f542d6f7665722d4d5154542f696d672f6d7174742e706e67> |
| IoT Agent communique **indirectement** avec les périphériques IoT via un courtier MQTT |
| Paradigme de [publication / abonnement](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) |
| IoT Devices choisit quand recevoir des communications |
| Faible puissance requise |

# **Qu'est-ce qu'un IoT agent?**



Un agent IoT est un composant qui permet à un groupe d’appareils d’envoyer leurs données vers un courtier Context et de les gérer à l’aide de leurs propres protocoles natifs. Les agents IoT devraient également être en mesure de gérer les aspects de sécurité de la plate-forme FIWARE (authentification et autorisation du canal) et fournir d'autres services communs au programmeur de périphérique. Il permet de :

* Offrir un emplacement standard pour écouter les mises à jour de l'appareil
* Offrir un emplacement standard pour écouter les mises à jour de données contextuelles
* Conserver une liste de périphériques et mapper des attributs de données de contexte à la syntaxe de périphérique
* Autorisation de sécurité

# **Les étapes d’installation**

1ére étape : **installation docker(déjà décrite dans le wiki installation Fiware)**

2éme étape : **vérifier l’installation**

( docker-compose -v

docker version )

3éme étape : **création des images nécessaires**

git clone git@github.com:Fiware/tutorials.IoT-over-MQTT.git

cd tutorials.IoT-over-MQTT

./services create

Initialiser les services

./services start

Device monitor

<http://localhost:3000/device/monitor>

### lancer nouveau terminal et commencer un MQTT Subscriber

docker run -it --rm --name mqtt-subscriber \

--network fiware\_default efrecon/mqtt-client sub -h mosquitto -t "/#"

### Ouvrir nouveau terminal ey lancer MQTT Publisher

docker run -it --rm --name mqtt-publisher \

--network fiware\_default efrecon/mqtt-client pub -h mosquitto -m "HELLO WORLD" -t "/test"

Voir le log de conteneur Docker Mosquitto pour montrer que la communication a eu lieu via le MQTT Broker

docker logs --tail 10 mosquitto

vérifier si l'agent IoT est en cours d'exécution en effectuant une requête HTTP sur le port exposé

curl -X GET \

'http://localhost:4041/iot/about'

## 4éme étape  **Connecter les Devices IOT**

1. Mise en service d'un groupe de services pour MQTT

=> Pour la communication MQTT, l'approvisionnement fournit la clé d'authentification afin que l'agent IoT sache à quel sujet il doit s'abonner. (ici topic est 4jggokgpepnvsb2uv4s40d59ov)

curl -iX POST \

'http://localhost:4041/iot/services' \

-H 'Content-Type: application/json' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /' \

-d '{

"services": [

{

"apikey": "4jggokgpepnvsb2uv4s40d59ov",

"cbroker": "http://orion:1026",

"entity\_type": "Thing",

"resource": ""

}

]

}'

1. Mise en service d'un capteur

Request :

curl -iX POST \

'http://localhost:4041/iot/devices' \

-H 'Content-Type: application/json' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /' \

-d '{

"devices": [

{

"device\_id": "motion001",

"entity\_name": "urn:ngsd-ld:Motion:001",

"entity\_type": "Motion",

"protocol": "PDI-IoTA-UltraLight",

"transport": "MQTT",

"timezone": "Europe/Berlin",

"attributes": [

{ "object\_id": "c", "name": "count", "type": "Integer" }

],

"static\_attributes": [

{ "name":"refStore", "type": "Relationship", "value": "urn:ngsi-ld:Store:001"}

]

}

]

}

'

MQTT Request

docker run -it --rm --name mqtt-publisher --network \

fiware\_default efrecon/mqtt-client pub -h mosquitto -m "c|1" \

-t "/4jggokgpepnvsb2uv4s40d59ov/motion001/attrs"

Request

curl -X GET \

'http://localhost:1026/v2/entities/urn:ngsd-ld:Motion:001?type=Motion' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /'

1. Provisionner un actionneur

Request :

curl -iX POST \

'http://localhost:4041/iot/devices' \

-H 'Content-Type: application/json' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /' \

-d '{

"devices": [

{

"device\_id": "bell001",

"entity\_name": "urn:ngsi-ld:Bell:001",

"entity\_type": "Bell",

"protocol": "PDI-IoTA-UltraLight",

"transport": "MQTT",

"commands": [

{ "name": "ring", "type": "command" }

],

"static\_attributes": [

{"name":"refStore", "type": "Relationship","value": "urn:ngsi-ld:Store:001"}

]

}

]

}

'

Request

curl -iX POST \

'http://localhost:4041/v1/updateContext' \

-H 'Content-Type: application/json' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /' \

-d '{

"contextElements": [

{

"type": "Bell",

"isPattern": "false",

"id": "urn:ngsi-ld:Bell:001",

"attributes": [

{ "name": "ring", "type": "command", "value": "" }

],

"static\_attributes": [

{"name":"refStore", "type": "Relationship","value": "urn:ngsi-ld:Store:001"}

]

}

],

"updateAction": "UPDATE"

}'

* Changement d’etat de la cloche

Request

curl -X GET \

'http://localhost:1026/v2/entities/urn:ngsi-ld:Bell:001?type=Bell&options=keyValues' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /'

1. Provisionner une porte intelligente

curl -iX POST \

'http://localhost:4041/iot/devices' \

-H 'Content-Type: application/json' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /' \

-d '{

"devices": [

{

"device\_id": "door001",

"entity\_name": "urn:ngsi-ld:Door:001",

"entity\_type": "Door",

"protocol": "PDI-IoTA-UltraLight",

"transport": "MQTT",

"commands": [

{"name": "unlock","type": "command"},

{"name": "open","type": "command"},

{"name": "close","type": "command"},

{"name": "lock","type": "command"}

],

"attributes": [

{"object\_id": "s", "name": "state", "type":"Text"}

],

"static\_attributes": [

{"name":"refStore", "type": "Relationship","value": "urn:ngsi-ld:Store:001"}

]

}

]

}

'

1. Provisionner une lampe intelligente

curl -iX POST \

'http://localhost:4041/iot/devices' \

-H 'Content-Type: application/json' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /' \

-d '{

"devices": [

{

"device\_id": "lamp001",

"entity\_name": "urn:ngsi-ld:Lamp:001",

"entity\_type": "Lamp",

"protocol": "PDI-IoTA-UltraLight",

"transport": "MQTT",

"commands": [

{"name": "on","type": "command"},

{"name": "off","type": "command"}

],

"attributes": [

{"object\_id": "s", "name": "state", "type":"Text"},

{"object\_id": "l", "name": "luminosity", "type":"Integer"}

],

"static\_attributes": [

{"name":"refStore", "type": "Relationship","value": "urn:ngsi-ld:Store:001"}

]

}

]

}

'

* RQ : Pour savoir la liste des devices provisionnés

curl -X GET \

'http://localhost:4041/iot/devices' \

-H 'fiware-service: openiot' \

-H 'fiware-servicepath: /'

5 ème étape **ACTIVATION DES CMD DE CONTEXT BROKER**

# **Pratique :**

