Les objets connectés
(MQTT)

LE CREN Anthony

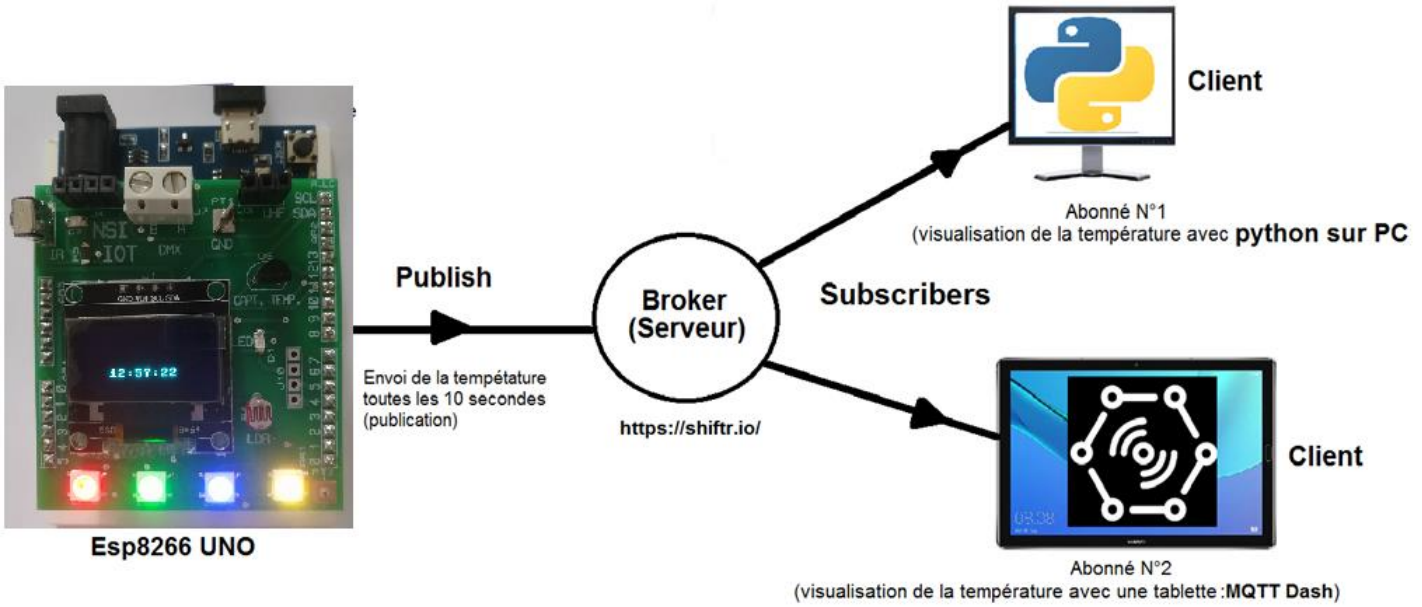
Les logins et les mots de passe décrits dans ce document sont donnés comme exemple.

1 Qu'est-ce que MQTT ?

MQTT (Message Queuing Telemetry Transport) est une messagerie publish-subscribe basé sur le protocole TCP/IP. MQTT est utilisé pour les IOT (Internet of Things / objects connectés). Il est conçu comme un transport de messagerie de publication / abonnement extrêmement léger en termes de ressources.

Mise en situation :

Vous avez réalisé un système à base d'ESP8266 permettant de mesurer la température d'une pièce. Vous voulez connaitre cette température quand vous êtes à l'extérieur de votre maison. A première vue, une solution serait de concevoir une page WEB afin de pouvoir y accéder depuis un navigateur. MQTT va vous permettre de remplir cet objectif plus rapidement en utilisant une bande passante très réduite. Il est aussi possible de déclencher un mécanisme à distance comme une des volets roulants etc... La communication peut ainsi être bidirectionnelle.



Communication en 2 étapes :

- Un objet connecté va publier (**publish**) son message vers le Broker (Serveur)
- Pour recevoir des messages le client va souscrire (**subscribe**) leurs réceptions auprès du Broker.

Pour que les messages ne se mélangent pas, ils sont publiés sur une chaine (topic). Par exemple /capteurs/temperature. Voir la vidéo de présentation.

1 Publication de température

Q1 A l'aide de l'annexe 1, **configurer** votre broker sur le site https://www.shiftr.io/cloud/

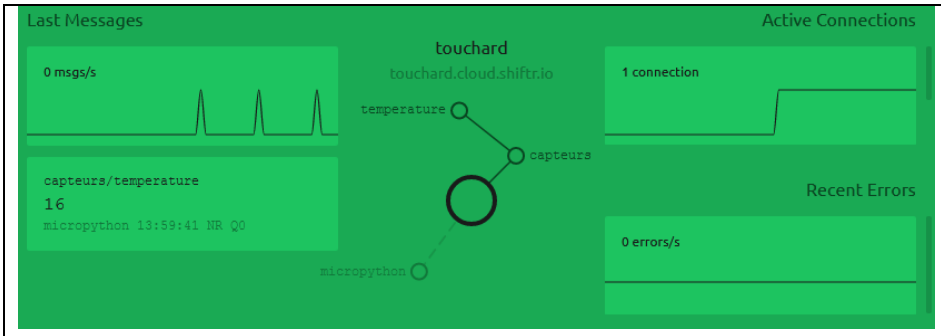
Noter l'adresse, le login et le mot de passe utilisé lors de l'enregistrement dans le tableau ci-dessous :

Adresse du serveur (URL)	touchard.cloud.shiftr.io
Client id (myMqttClient)	Python ou esp8266 ou arduino
Login (USER_ID)	touchard
Mot de passe (MQTT_API_KEY)	MFmD747BIp8YIJYI
Topic ou token	capteurs

Q2 Vérifier la connexion sur le point d'accès internet en utilisant le programme « **`wifi_connexion.py`** »
Configurer et **tester** le programme ci-dessous avec vos propres identifiants conformément à la question **Q1.** (Q2-mqtt-publish.py)

```
from umqtt.robust import MQTTClient
import network
import sys
import time
from time import sleep_ms
from machine import Pin
from onewire import OneWire
from ds18x20 import DS18X20


sta_if = network.WLAN(network.STA_IF)
print(sta_if.active())
print(sta_if.ifconfig())

myMqttClient = b"micropython"

URL = b"touchard.cloud.shiftr.io"
USER_ID = b'touchard'
MQTT_API_KEY = b'MFmD747BIp8YIJYI'
client = MQTTClient(client_id=myMqttClient,
                    server=URL,
                    user=USER_ID,
                    password=MQTT_API_KEY,
                    ssl=False)

try:
    client.connect()
    print("connection ok");
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__, e))
    sys.exit()

bus = OneWire(Pin(12))
ds = DS18X20(bus)
capteur_temperature = ds.scan()

PUBLISH PERIOD IN SEC = 10

while True:
    try:
        ds.convert_temp()
        sleep_ms( 750 )
        temp_celsius = ds.read_temp(capteur_temperature[0])
        print("Température : ",temp_celsius )
        client.publish("/capteurs/temperature", str(int(temp_celsius)))
        print("publish ok");
        time.sleep(PUBLISH_PERIOD_IN_SEC)
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()
        print("exit")
```

Q3 Vérifier l'envoi de données au broker. (https://touchard.cloud.shiftr.io/)



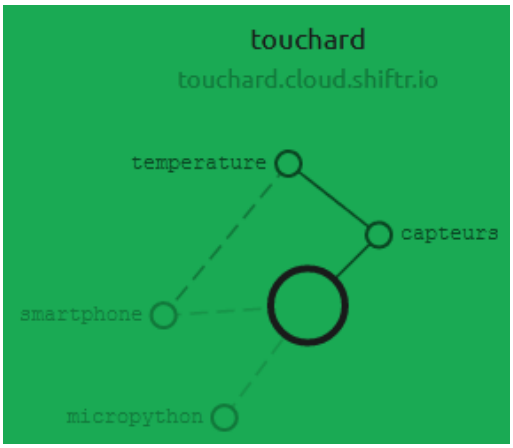

```
True

('192.168.1.104',
'255.255.255.0',
'192.168.1.1',
'192.168.1.1')

connection ok

Température :  16.25

publish ok
```

Q4 A l'aide de l'annexe 2,3 ou 4, **configurer** le client MQTT sur un smartphone. Préférence pour **MQTT panel** : annexe 3)

Noter à nouveau la clé et le mot de passe en lien avec le broker (**Q1**) dans le tableau ci-dessous :

Adresse du serveur	touchard.cloud.shiftr.io
Client id	smartphone
login	touchard
Mot de passe	MFmD747BIp8YIJYI
Topic ou token	capteurs

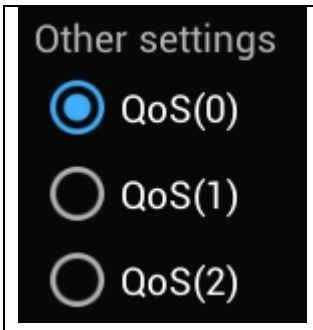
Q5 Vérifier l'envoi de données du broker vers le smartphone.



Q6 Identifier le subscriber et le publisher en complétant le tableau ci-dessous par oui ou non

	Subscriber	Publisher
L'esp8266 est le	Non	Oui
Le smartphone est le	Oui	Non

La qualité de service (QDS) ou quality of service (QoS) est la capacité à véhiculer dans de bonnes conditions un type de trafic donné.



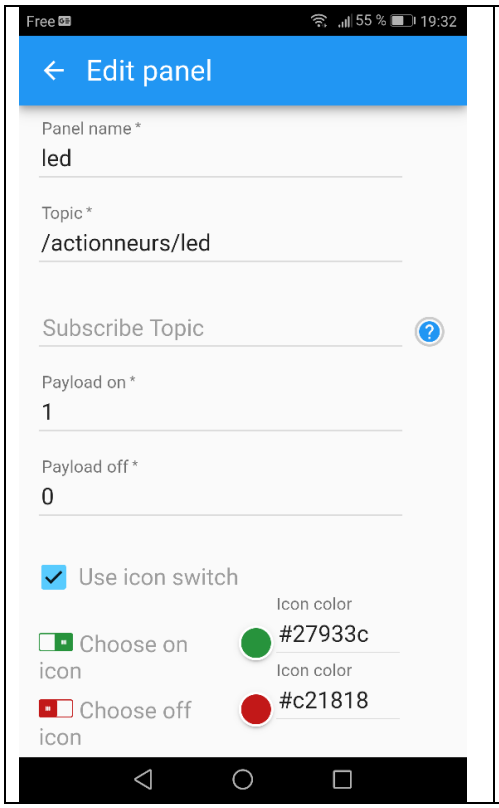
- QoS0. Le message envoyé n'est pas stocké par le Broker. Il n'y a pas d'accusé de réception. Le message sera perdu en cas d'arrêt du serveur ou du client. C'est le mode par défaut
- QoS1. Le message sera livré au moins une fois. Le client renvoie le message jusqu'à ce que le broker envoi en retour un accusé de réception.
- QoS2. Le broker sauvegarde le message et le transmettra jusqu'à ce qu'il ait été réceptionné par tous les souscripteurs connectés

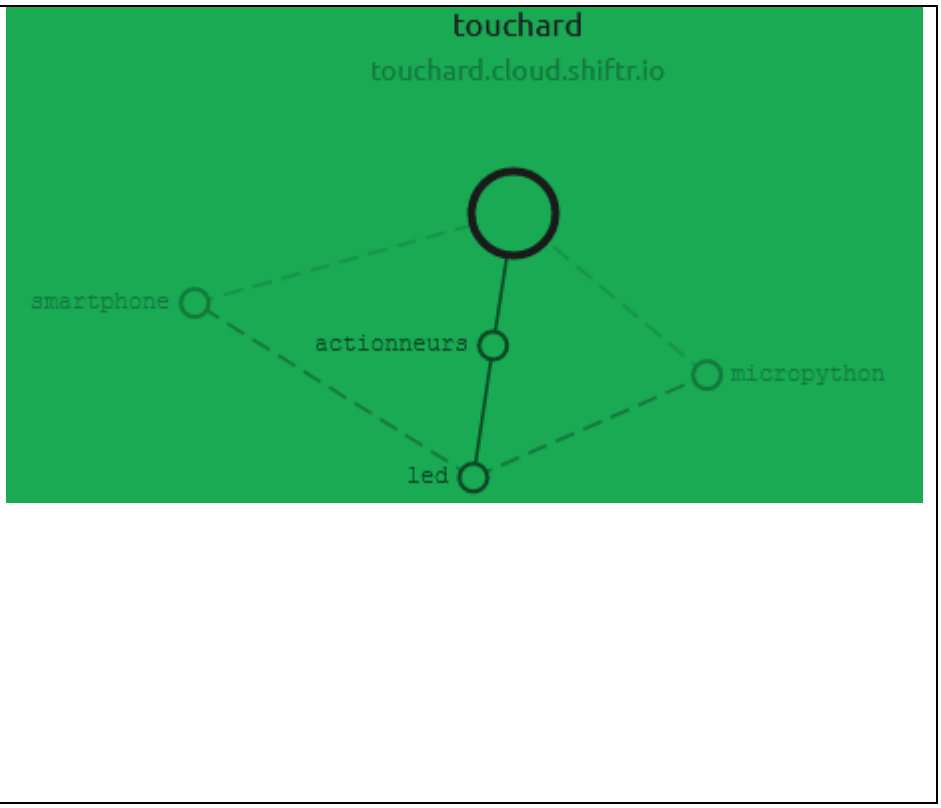
2 Commande de la Led de l'IOT avec le smartphone

Q7 Configurer et tester le programme ci-dessous avec vos propres identifiants conformément à la question **Q1** (Q7-mqtt-subscribe-led.py)

```
from umqtt.robust import MQTTClient
import network
import sys
from machine import Pin

led = Pin(0,Pin.OUT)


def cb(topic, msg):
    print((topic, msg))
    valeur=int(msg.decode("ascii"))
    if valeur==1:
        led.on()
    elif valeur==0:
        led.off()

sta_if = network.WLAN(network.STA_IF)
print(sta_if.active())
print(sta_if.ifconfig())

myMqttClient = b"micropython"

URL = b"touchard.cloud.shiftr.io"
USER_ID = b'touchard'
MQTT_API_KEY = b'MFmD747BIp8YIJYI'
client = MQTTClient(client_id=myMqttClient,
                    server=URL,
                    user=USER_ID,
                    password=MQTT_API_KEY,
                    ssl=False)

client.set_callback(cb)

try:
    client.connect()
    print("connection ok");
    client.subscribe("/actionneurs/led")
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__, e))
    sys.exit()


while True:
    try:
        client.check msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()
        print("exit")
```

Q8 Ajouter un panel appelé « **Switch** »de commande comme le montre l'exemple ci-dessous, puis valider la commande de la led sur le site shiftr.io





Q9 Identifier le subscriber et le publisher en complétant le tableau ci-dessous par oui ou non

	Subscriber	Publisher
L'esp8266 est le	Oui	Non
Le smartphone est le	Non	Oui

3 Analyse du protocole MQTT

http://mqtt.org/documentation

Connect Command (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718028

```
▼ MQ Telemetry Transport Protocol
  ▼ Connect Command
    ▶ 0001 0000 = Header Flags: 0x10 (Connect Command)
      Msg Len: 53
      Protocol Name: MQTT
      Version: 4
    ▶ 1100 0010 = Connect Flags: 0xc2
      Keep Alive: 60
      Client ID: processing
      User Name: weatherSensors
      Password: bme280Sensors
```

```
0000  90 4d 4a a3 0e 00 00 25  22 8a 86 a0 08 00 45 00   .MJ....% ".....E.
0010  00 6b 93 a5 40 00 40 06  96 d9 c0 a8 01 15 36 4c   .k..@.@. ......6L
0020  18 05 b7 ca 07 5b 79 8f  ed 67 52 30 38 88 80 18   .....[y. .gR08...
0030  00 e5 4a 0b 00 00 01 01  08 0a d4 85 d3 f7 07 17   ..J..... ........
0040  39 fa 10 35 00 04 4d 51  54 54 04 c2 00 3c 00 0a   9..5..MQ TT...<..
0050  70 72 6f 63 65 73 73 69  6e 67 00 0e 77 65 61 74   processi ng..weat
0060  68 65 72 53 65 6e 73 6f  72 73 00 0d 62 6d 65 32   herSenso rs..bme2
0070  38 30 53 65 6e 73 6f 72  73                       80Sensor s
```

Connect Command

0001 0000 = Header Flags: 0x10 (Connect Command)

Msg Len: 53

Protocol Name: MQTT

Version: 4

1100 0010 = Connect Flags: 0xc2

Keep Alive: 60

Client ID: processing

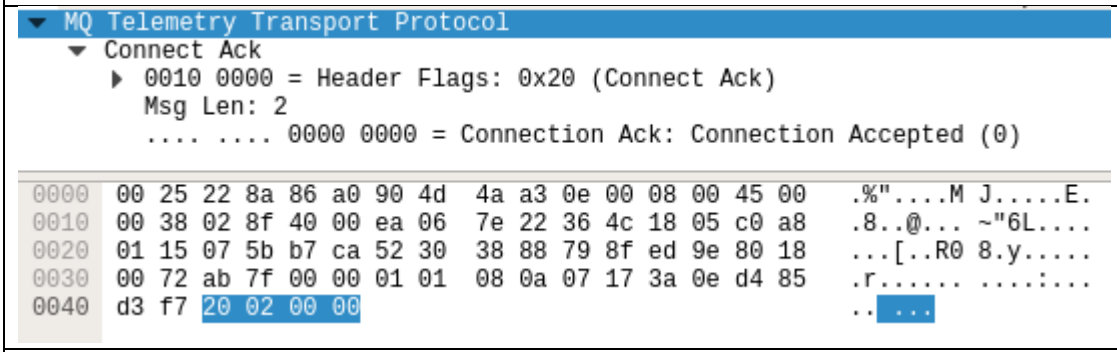
User Name: weatherSensors

Password: bme280Sensors

```
0000   10 35 00 04 4d 51 54 54 04 c2 00 3c 00 0a 70 72  .5..MQTT...<..pr
0010   6f 63 65 73 73 69 6e 67 00 0e 77 65 61 74 68 65  ocessing..weathe
0020   72 53 65 6e 73 6f 72 73 00 0d 62 6d 65 32 38 30  rSensors..bme280
0030   53 65 6e 73 6f 72 73                             Sensors
```

Connect Ack (server to client)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718033

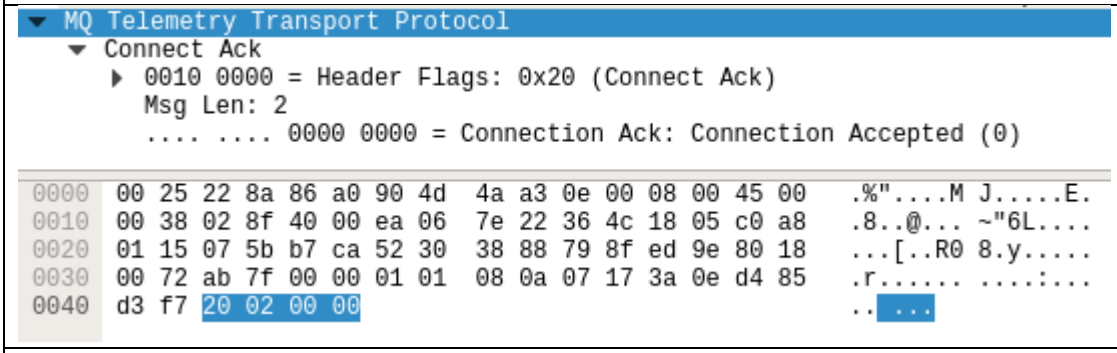


Connect Ack
0010 0000 = Header Flags: 0x20 (Connect Ack)
Msg Len: 2
.... 0000 0000 = Connection Ack: Connection Accepted (0)

0000 20 02 00 00

Subscribe Request (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718063



Subscribe Request
1000 0010 = Header Flags: 0x82 (Subscribe Request)
Msg Len: 25
Message Identifier: 1
Topic: /sensors/temperature
.... ..00 = Granted Qos: Fire and Forget (0)

0000 82 19 00 01 00 14 2f 73 65 6e 73 6f 72 73 2f 74 ....../sensors/t
0010 65 6d 70 65 72 61 74 75 72 65 00 emperature.

Subscribe Ack (server to client)

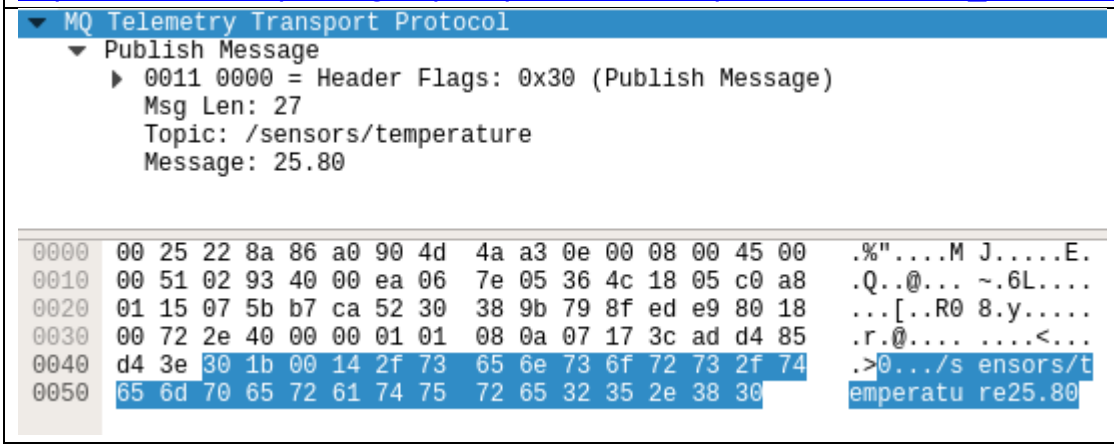
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718068

Subscribe Ack
1001 0000 = Header Flags: 0x90 (Subscribe Ack)
Msg Len: 3
Message Identifier: 1
.... ..00 = Granted Qos: Fire and Forget (0)

A PUBLISH Control Packet is sent **from a Client to a Server** or **from Server to a Client** to transport an Application Message.

Publish Message (server to client)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718037



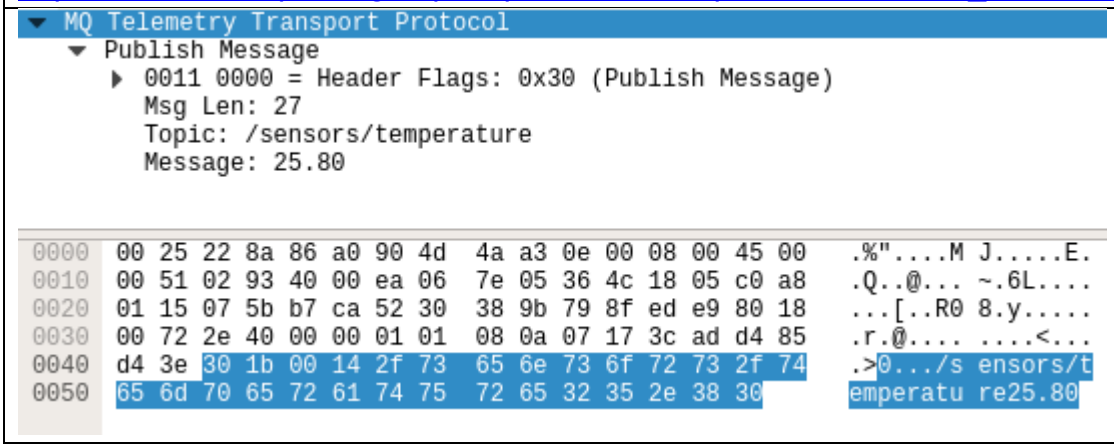
Publish Message **incoming**

0011 0000 = Header Flags: 0x30 (Publish Message)
Msg Len: 27
Topic: /sensors/temperature
Message: 25.80

```
0000   30 1b 00 14 2f 73 65 6e 73 6f 72 73 2f 74 65 6d  0.../sensors/tem
0010   70 65 72 61 74 75 72 65 32 35 2e 38 30           perature25.80
```

Publish Message (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718037

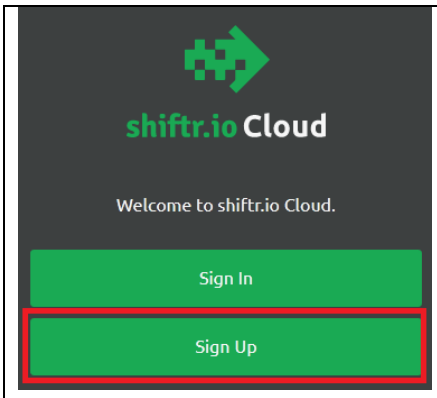
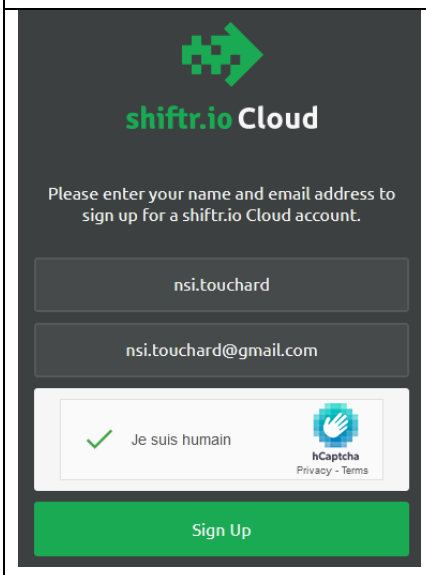
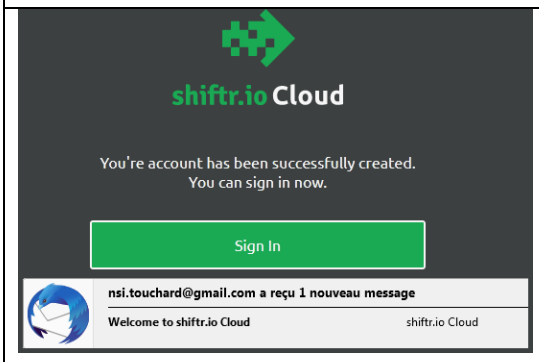


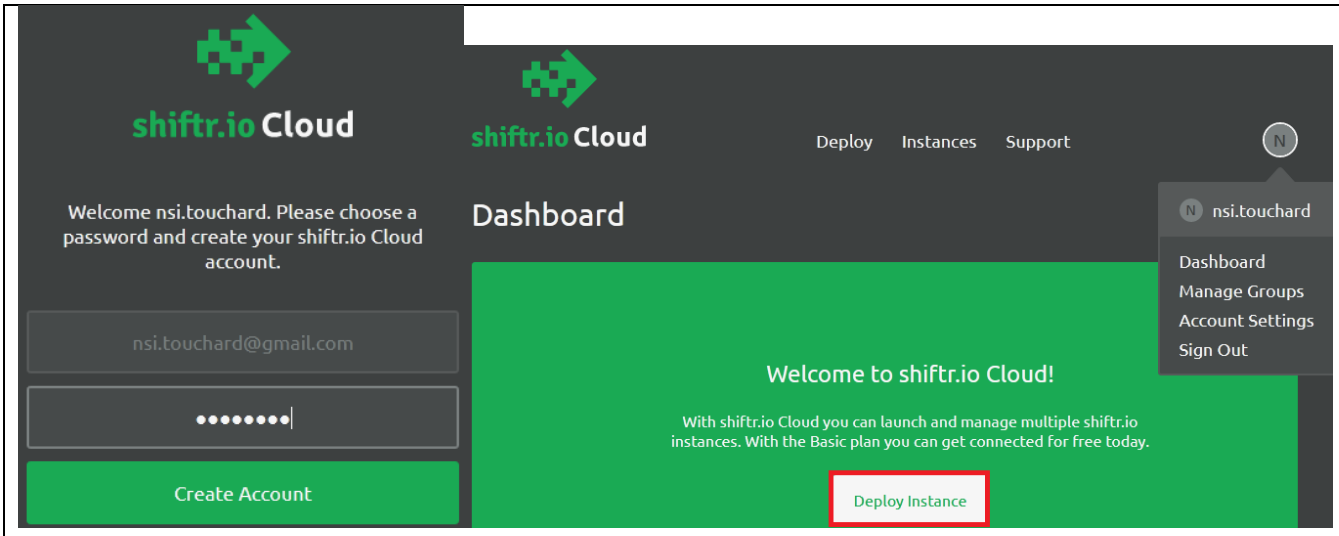
Publish Message **outcoming**

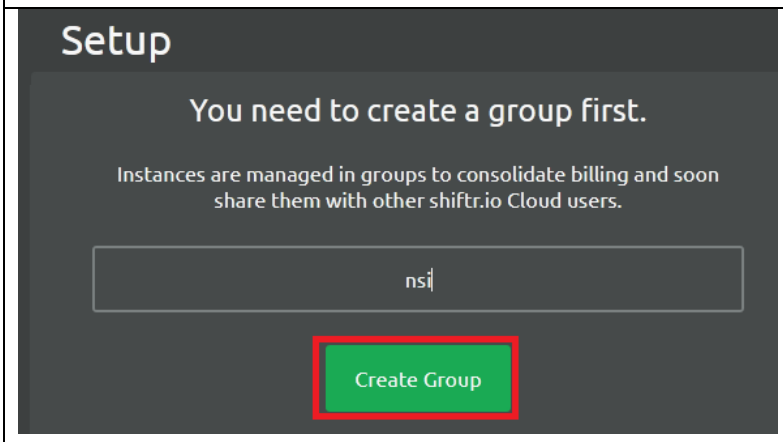
0011 0000 = Header Flags: 0x30 (Publish Message)
Msg Len: 24
Topic: /sensors/temperature
Message: 20

```
0000   30 18 00 14 2f 73 65 6e 73 6f 72 73 2f 74 65 6d  0.../sensors/tem
0010   70 65 72 61 74 75 72 65 32 30                    perature20
```

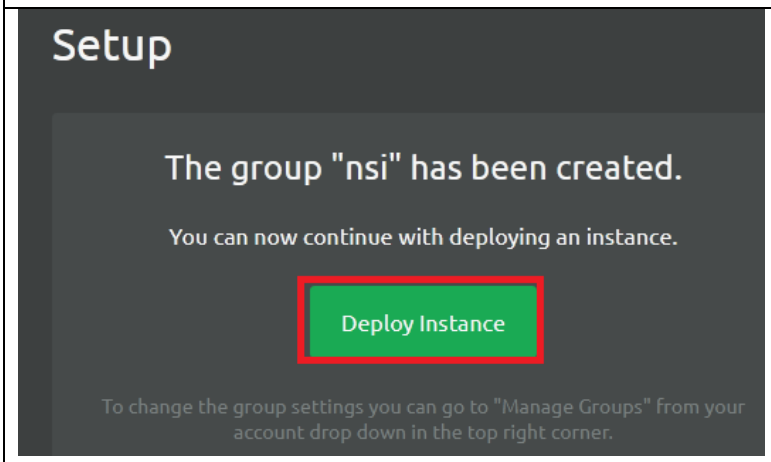
Annexe 1 : Configuration d'un broker (serveur MQTT) en ligne

	Créer un compte sur le site https://www.shiftr.io/cloud/ Cliquer sur Sign Up
	**Indiquer un login et un compte mail** **Puis cliquer sur Sign up**
	Confirmer le compte à partir de votre adresse mail
	Terminer la création du compte et afficher le « Dashboard » Cliquer ensuite sur « Deploy Instance »

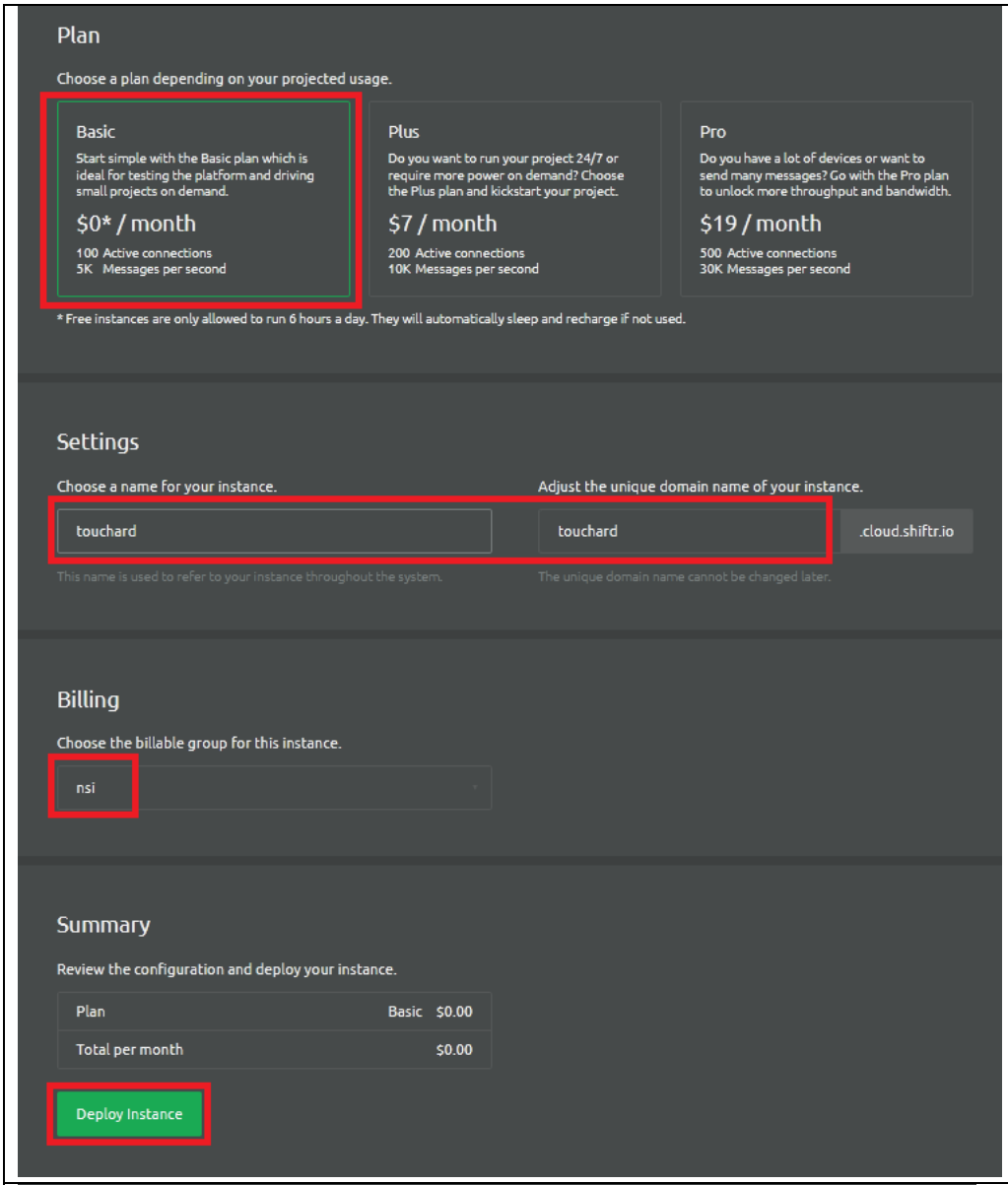




Créer un groupe



Cliquer sur « Deploy Instance »

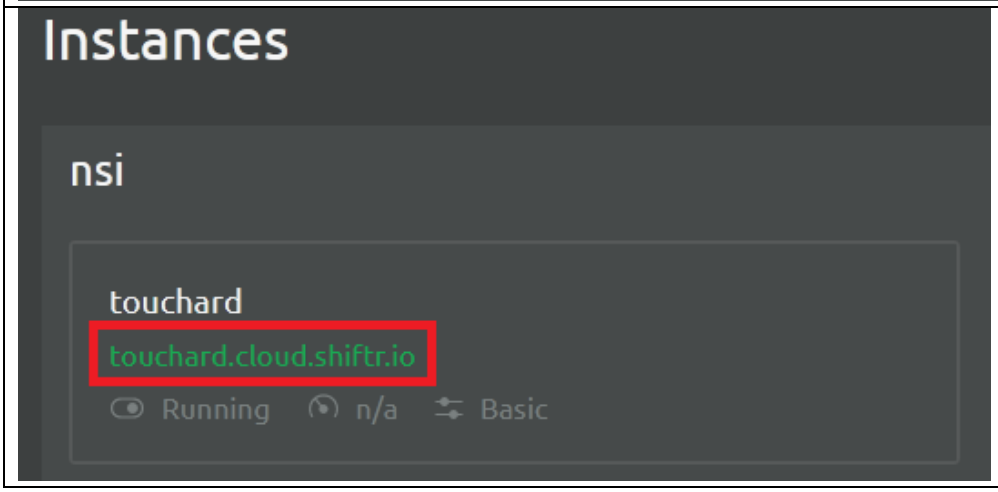


Choisir la version basic « Gratuite »

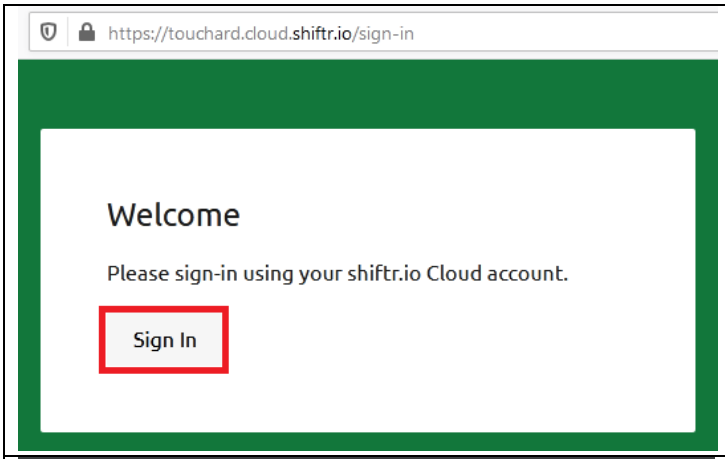
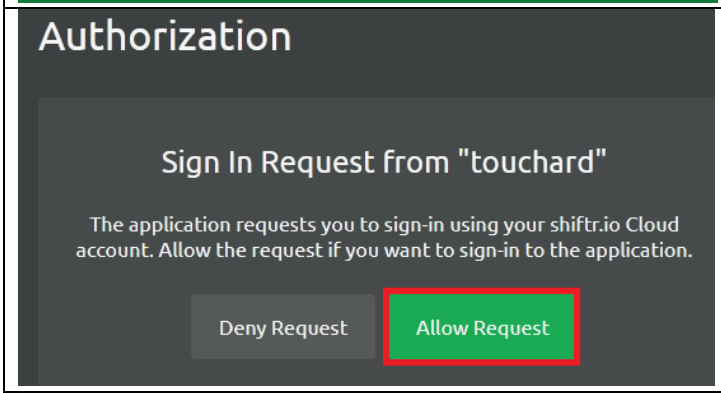
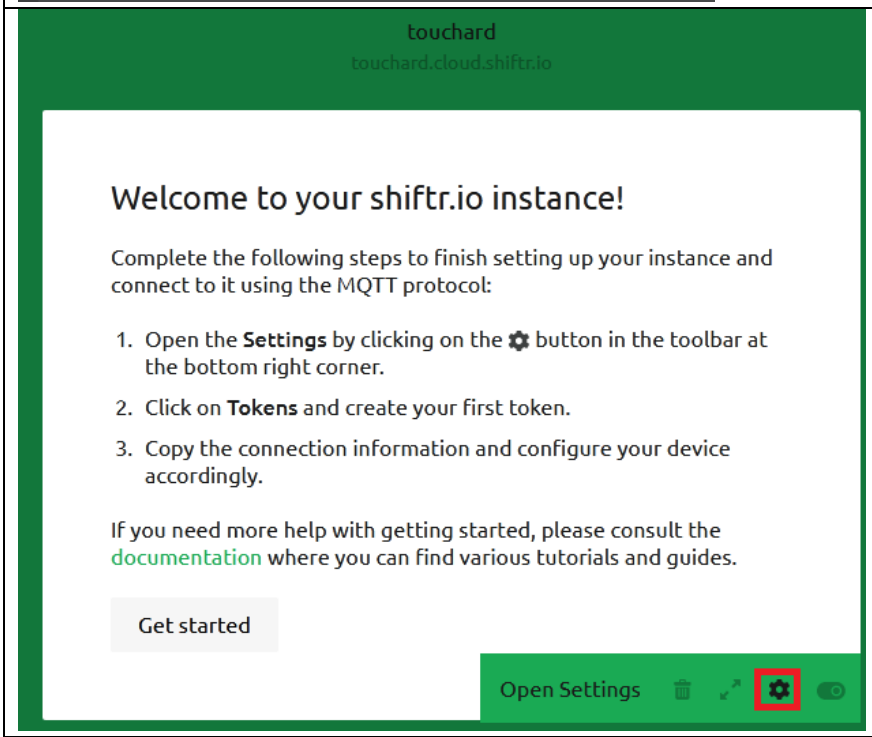
Puis donner un nom pour l'instance

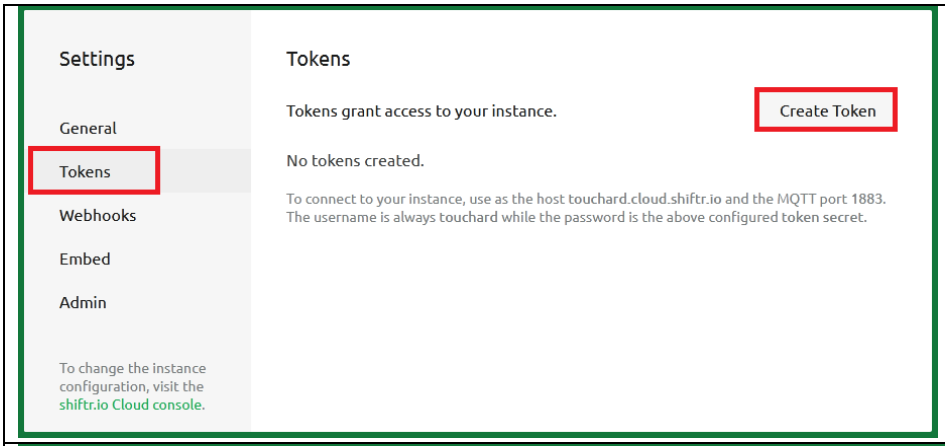
Ainsi que le nom de groupe crée précédemment.

Puis cliquer sur « deploy Instance »

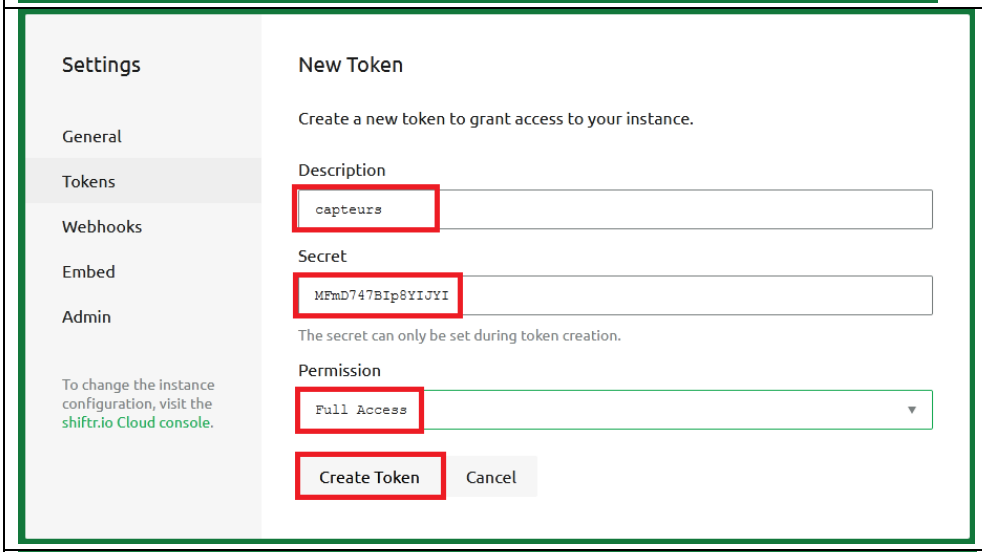


Cliquer sur l'instance (lien en couleur verte)

	Lier l'instance avec votre compte Crée au tout début
	Accepter l'autorisation
	Cliquer sur l'icône des paramètres

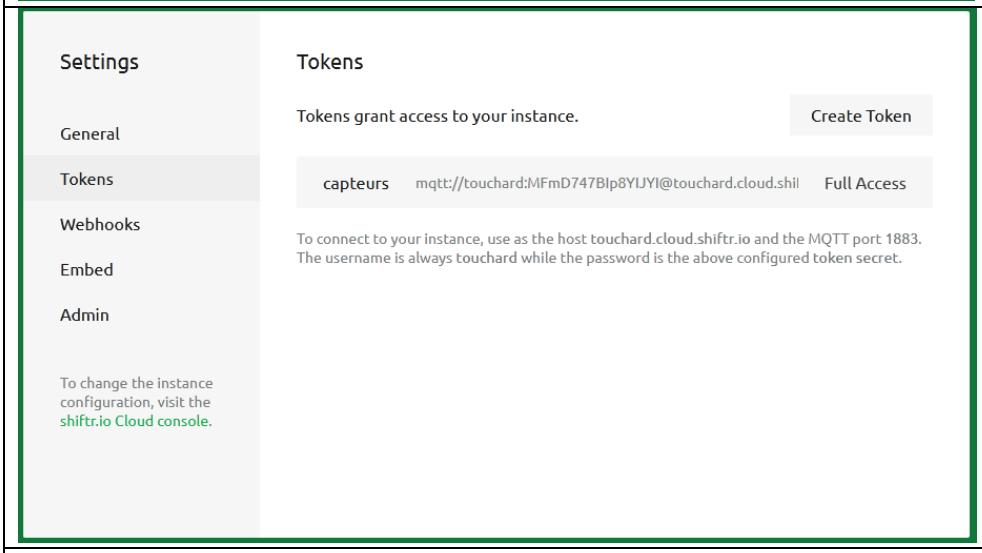


Créer un « token » ou topic.



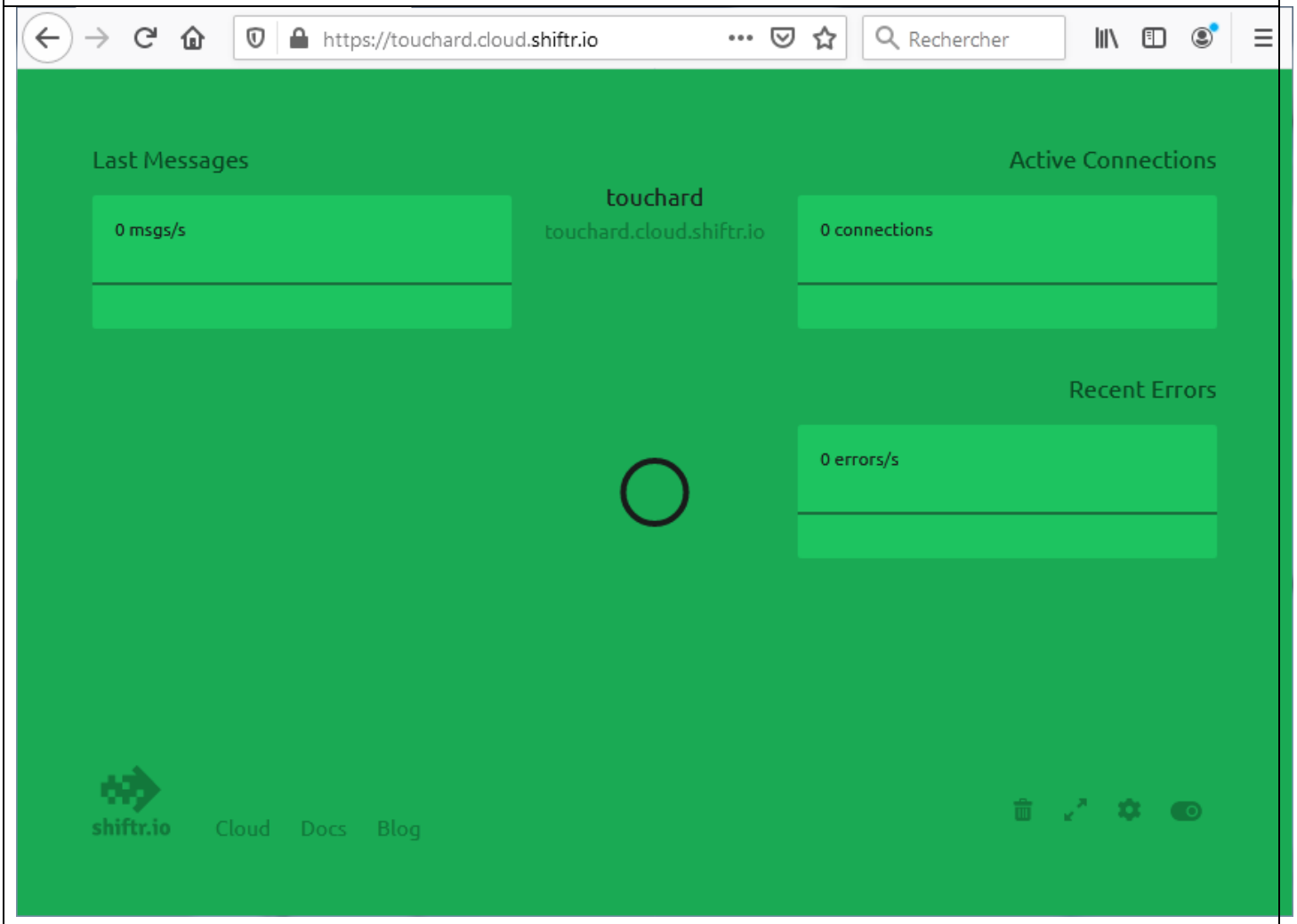
Renseigner la description du nouveau Token.

Par exemple « capteurs »



La création est maintenant terminée

Aller sur la page accueil de l'instance https://touchard.cloud.shiftr.io/



La configuration du Broker (Serveur MQTT) est terminée.

Exemple de configuration

Adresse du serveur	touchard.cloud.shiftr.io
Client id	python ou micropython ou arduino
login	touchard
Mot de passe	MFmD747BIp8YIJYI
Topic ou token	capteurs

Remarque :

Il est possible d'utiliser le broker shitr.io hors ligne et pouvant être utilise sur un réseau local sans avoir de connexion Internet. (Installation du programme shiftr-io-desktop.exe)

Il existe d'autres brokers gratuits comme https://www.maqiatto.com/ ,mais il ne dispose pas comme shiftr.io d'une interface graphique indiquant les connexions en cours.

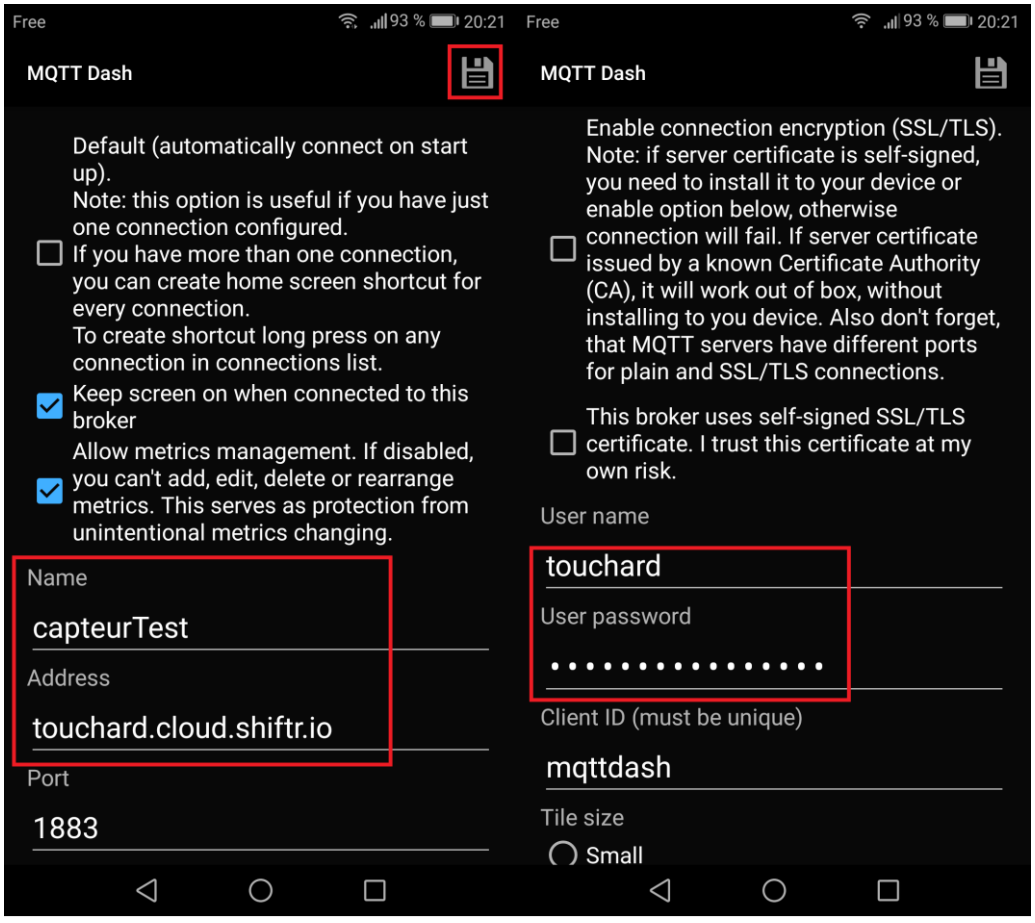
Annexe 2 : Configuration du client sur Android (MQTT Dash)

Installer le programme MQTT dash

https://play.google.com/store/apps/details?id=net.routix.mqttdash

Configuration :

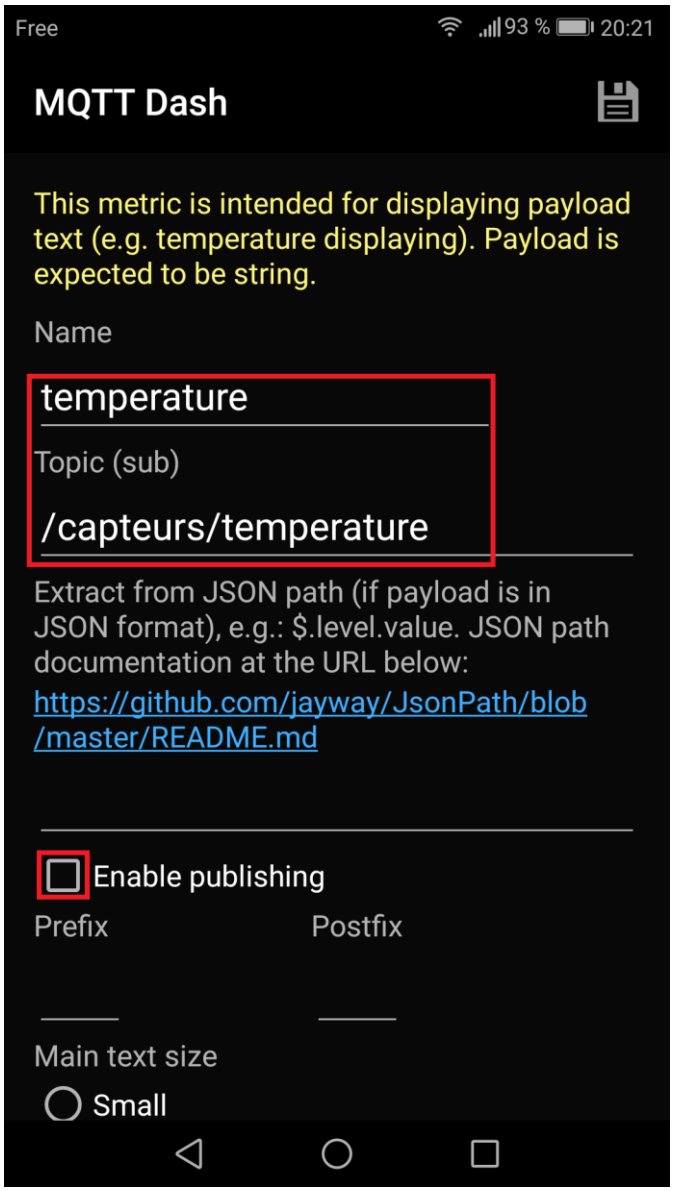
Ajouter une connexion vers le broker shiftr.io



Cliquer sur sensorsTest

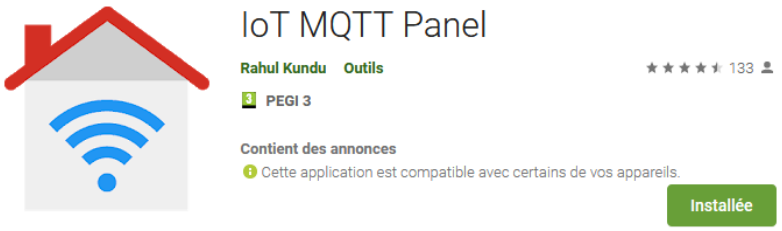
Puis ajouter le(s) topics nécessaires.

Exemple pour le Topic température (Ne pas oublier de **décocher** « Enable publishing »)



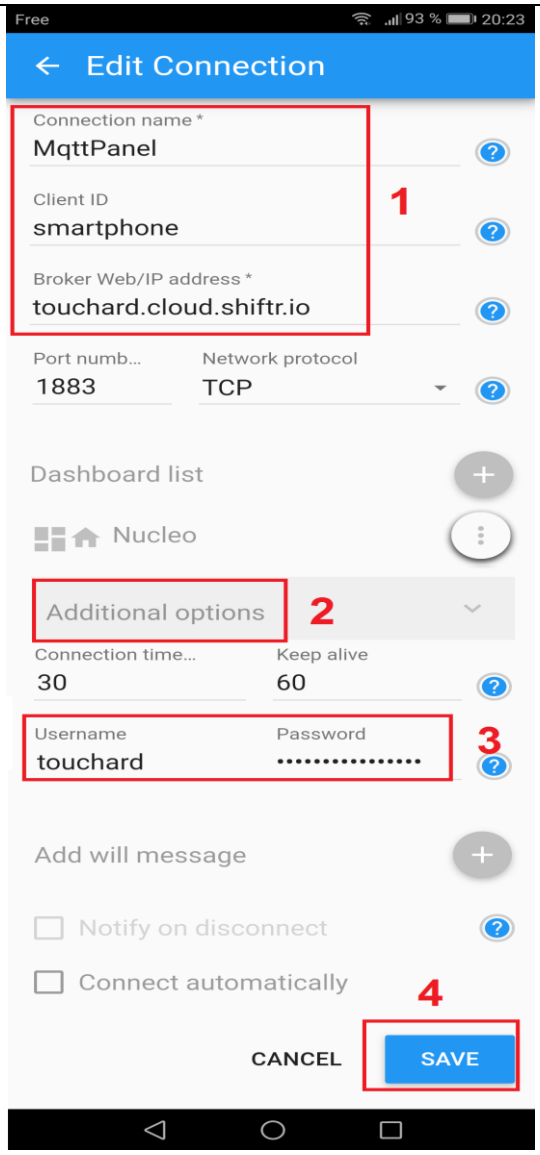
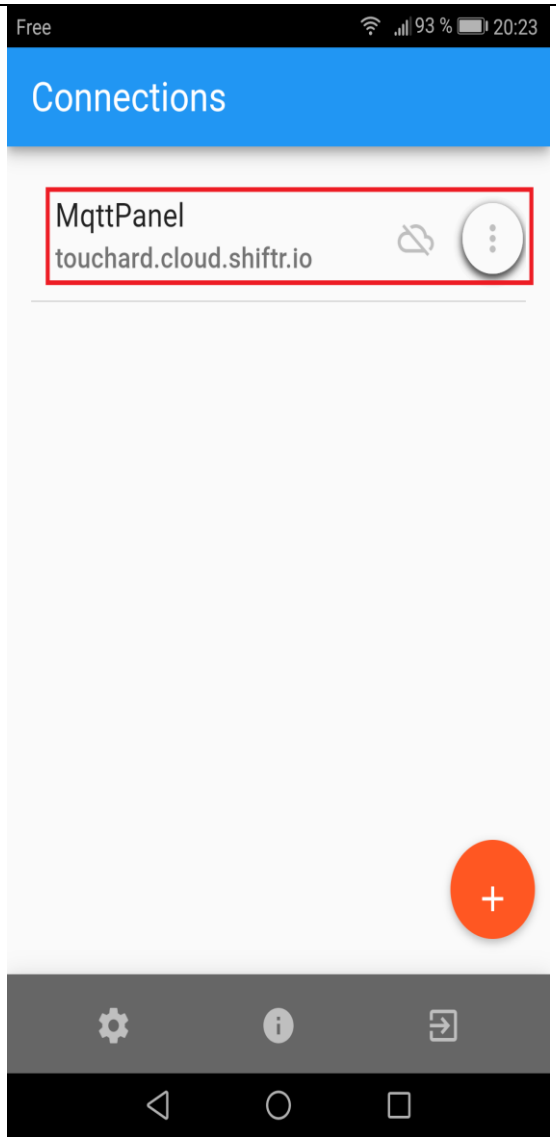
Annexe 3 : Configuration du client sur Android (MQTT panel)

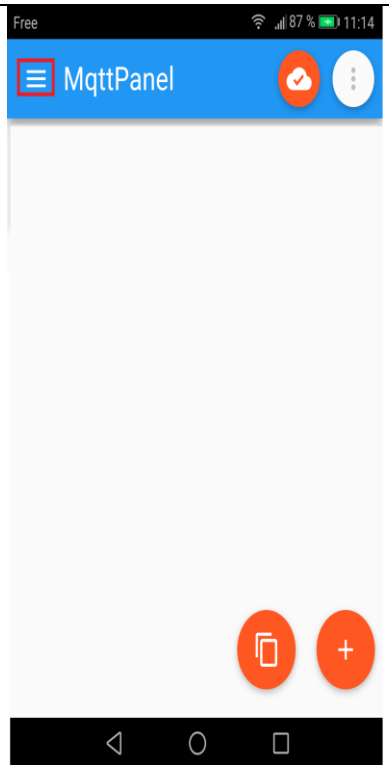
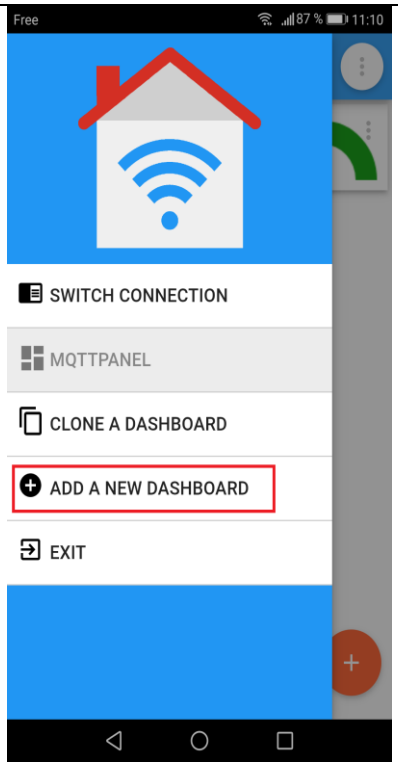
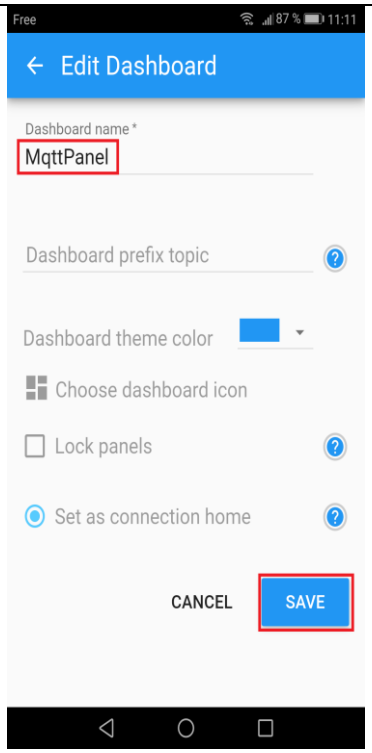
Installer le programme IoT MQTT Panel

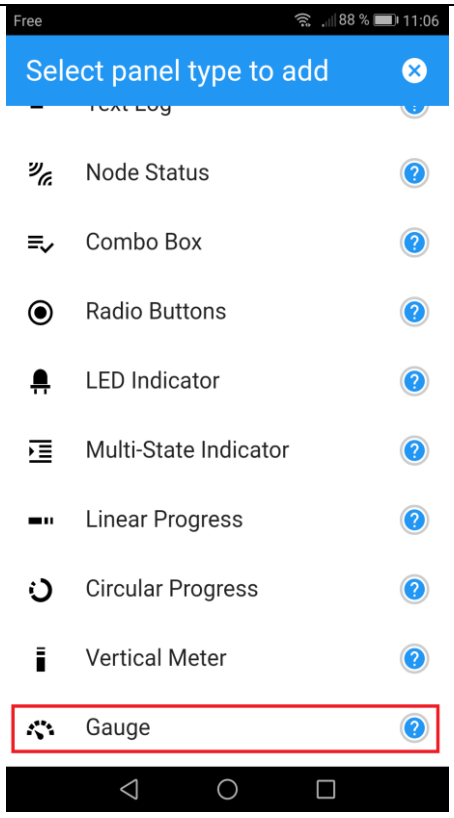
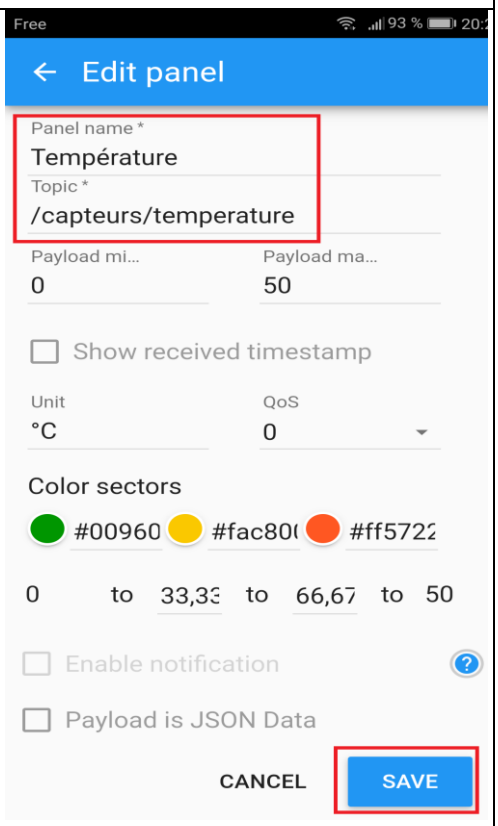


https://play.google.com/store/apps/details?id=snr.lab.iotmqttpanel.prod

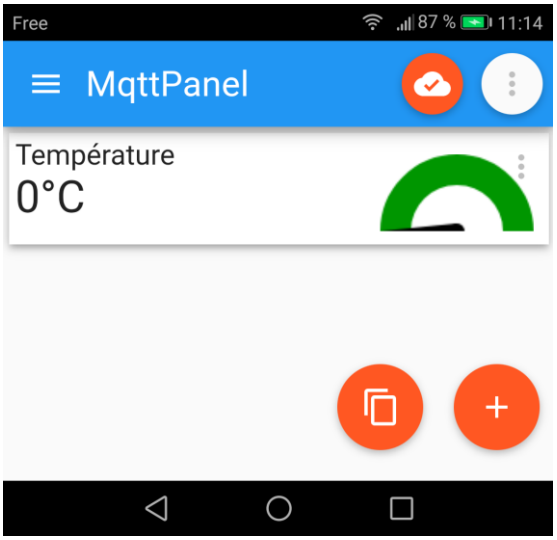
Ce programme un plus convivial que MQTT dash et possède des widgets. (Élément de base de l'interface graphique d'un logiciel : fenêtre, barre d'outils, par exemple). Une fois installé, cliquer sur ajouter une connexion icone  , puis configurer le broker shiftr.io

Configuration de la connexion au Broker	Sélectionner MqttPanel
	

Configuration de la connexion au Broker	Ajouter un nouveau Dashboard	Nommer le Dashboard
		

Ajouter un panel en cliquant sur l'icône rouge +	Sélectionner Gauge	Configurer le topic
		

La configuration du logiciel sur le smartphone est terminée. L'utilisateur est prêt à recevoir l'information de température en provenance de la carte IOT.



Exemple de configuration en lien avec le broker :

Name	MqttPanel
Adresse du serveur	touchard.cloud.shiftr.io
Client id	Smartphone
User name (key)	touchard
User password (password)	MFmD747BIp8YIJYI
Topic name	capteurs/temperature

Port 1883 : communication non sécurisée (données en clair sur le réseau) par défaut dans ce document.

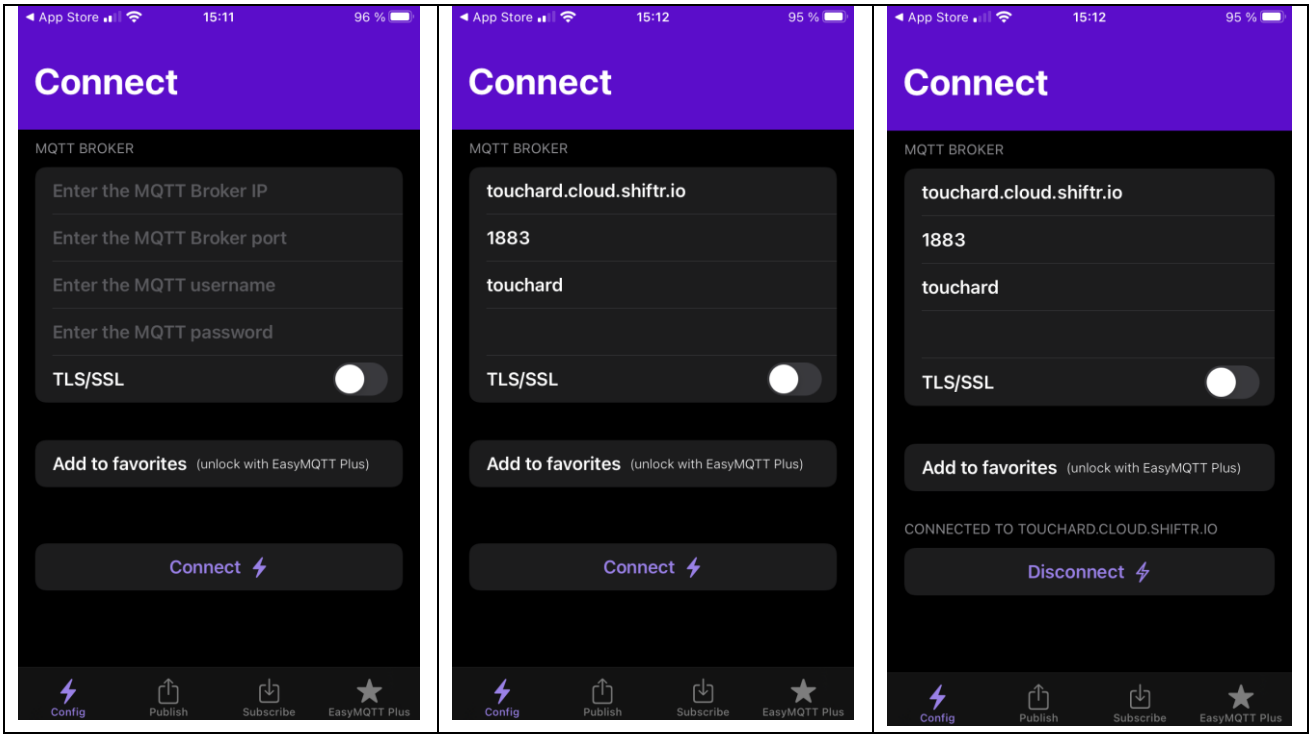
Port 8881 : communication non sécurisé SSL (Secure Socket Layer) / TLS (Transport Layer Security)

Annexe 4 : Configuration du client sur IPhone (EasyMQTT)

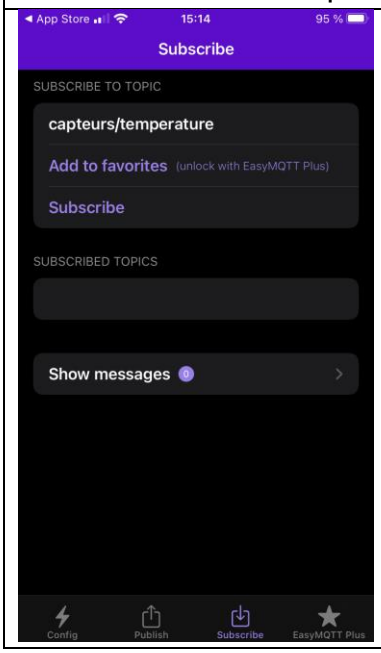
Installer le programme EasyMQTTsur le téléphone

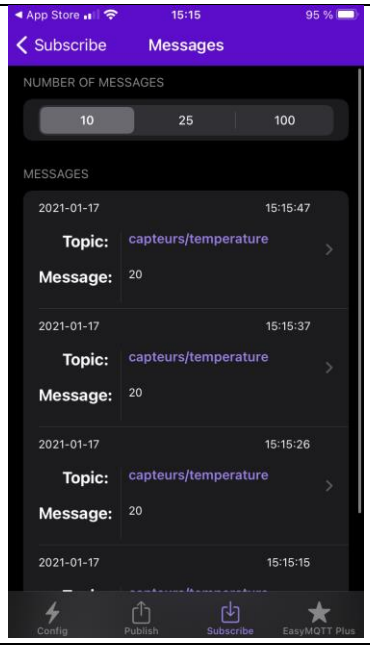
https://apps.apple.com/fr/app/easymqtt/id1523099606

Ajouter une connexion vers le broker shiftr.io, puis tester la connexion

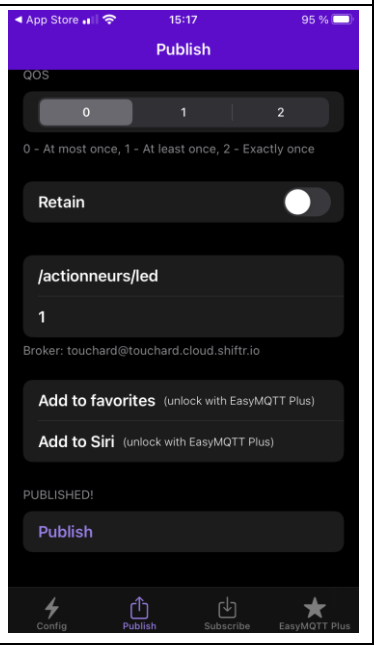


Exemple de souscription





Exemple de publication



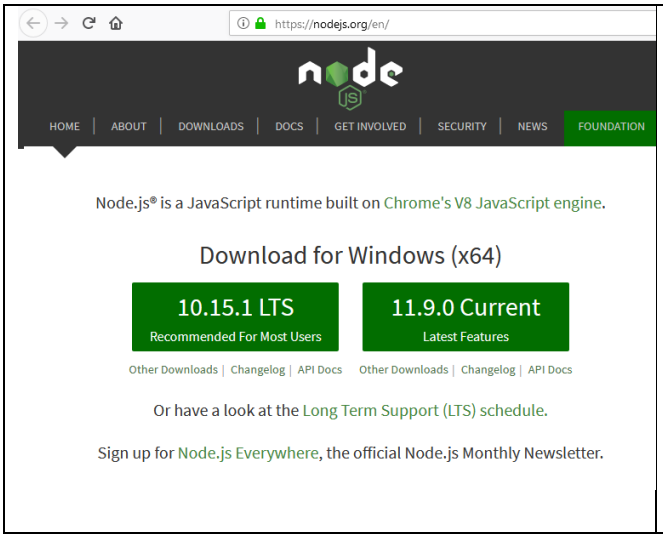
Annexe 5 Utilisation de Node-red

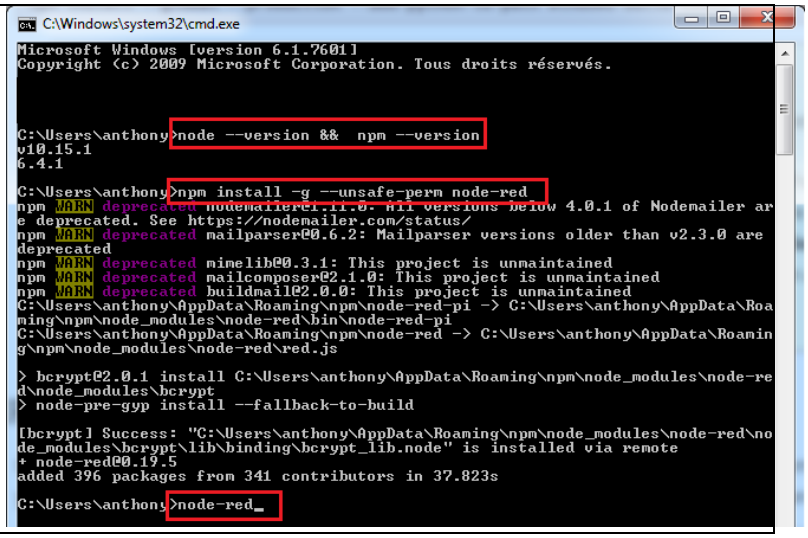
Node-RED est un outil de développement basé sur une programmation visuelle pour connecter un ensemble de périphériques matériels, des API et des services en ligne dans le cadre de l'Internet des objets.

3.1 Installation de Node-red sous Windows :

- Commencer par installer node js

https://nodejs.org/en/



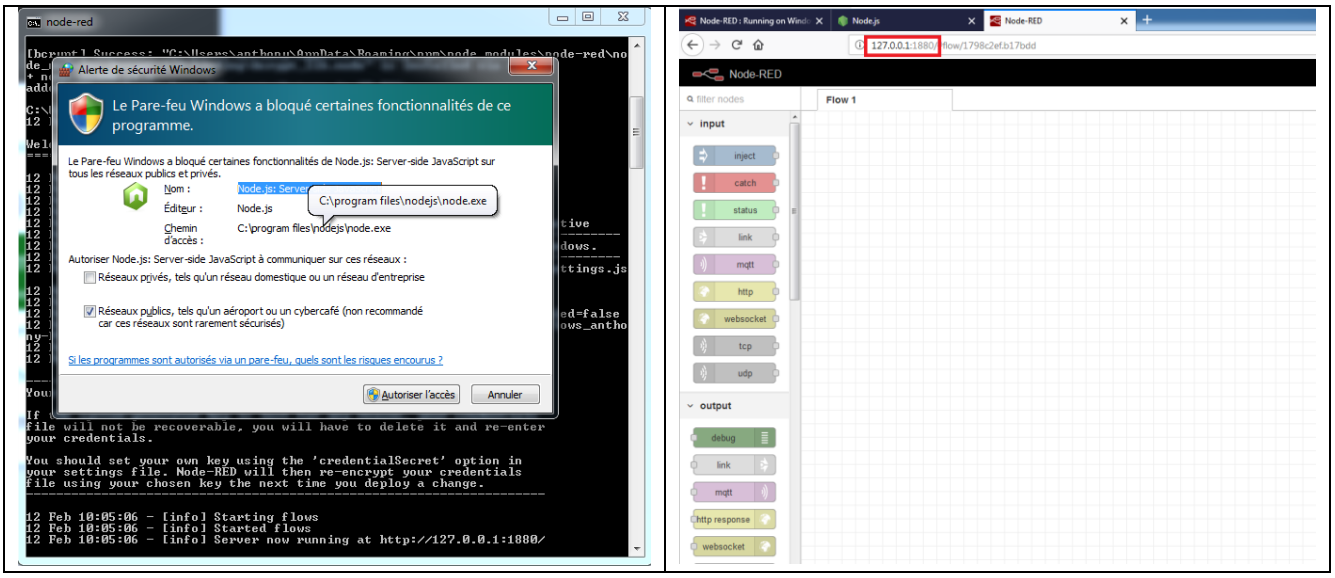


https://nodered.org/docs/platforms/windows

- Puis dans la console saisir les commandes suivantes :

```
node --version && npm –version

npm install -g --unsafe-perm node-red

node-red
```

Ne pas fermer la console



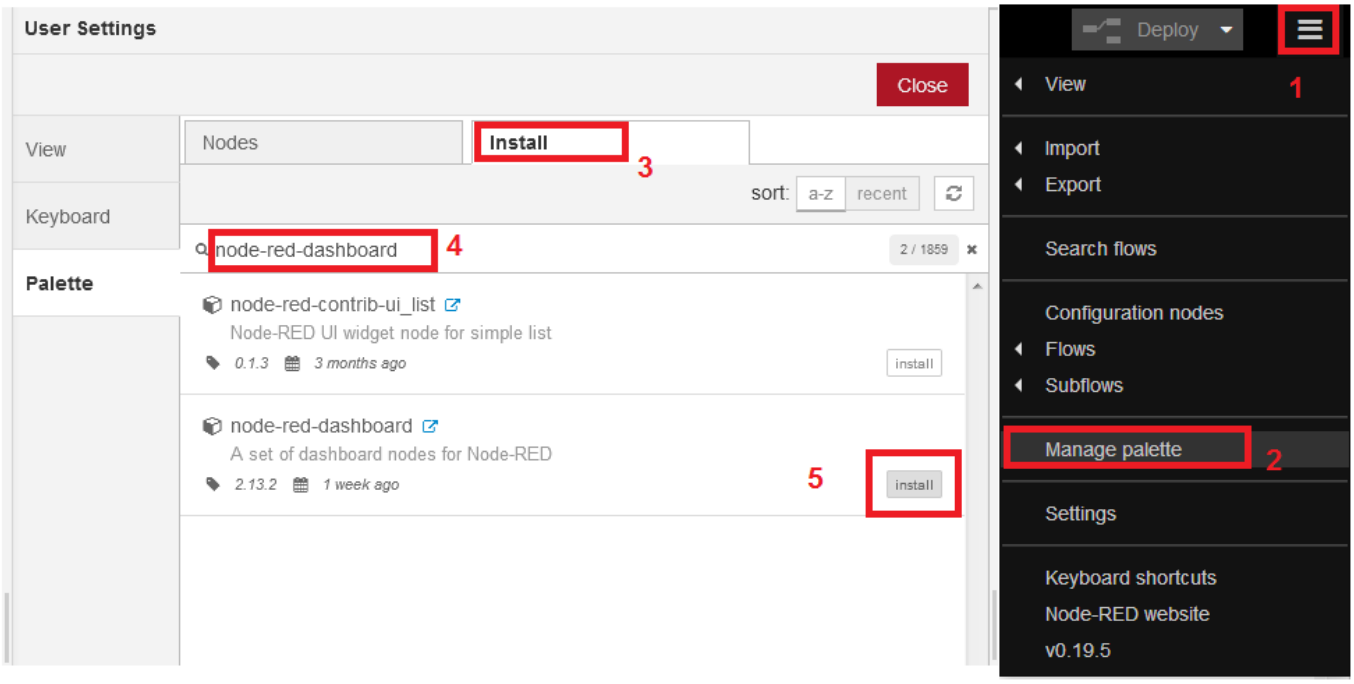
Ouvrir un navigateur puis saisir l'adresse suivante :

http://127.0.0.1:1880

Ne pas fermer la console.

3.2 Add-ons

Ajouts d'outils de gestion graphique.



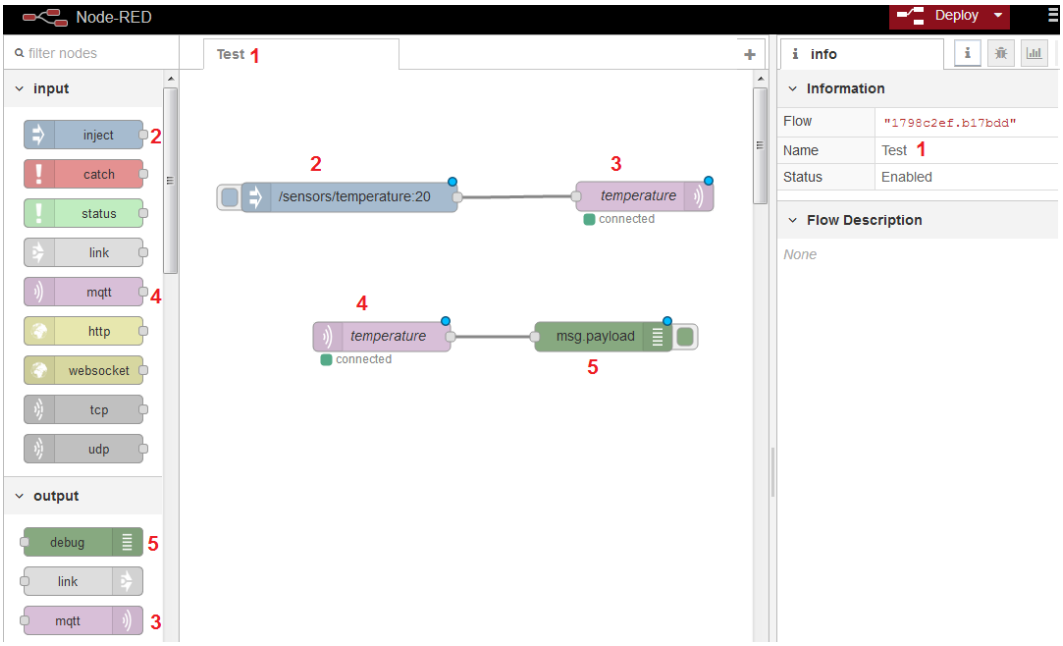
Il en va de même pour d'autres outils en connaissant le nom du fichier à installer.

3.2 Installation sous linux

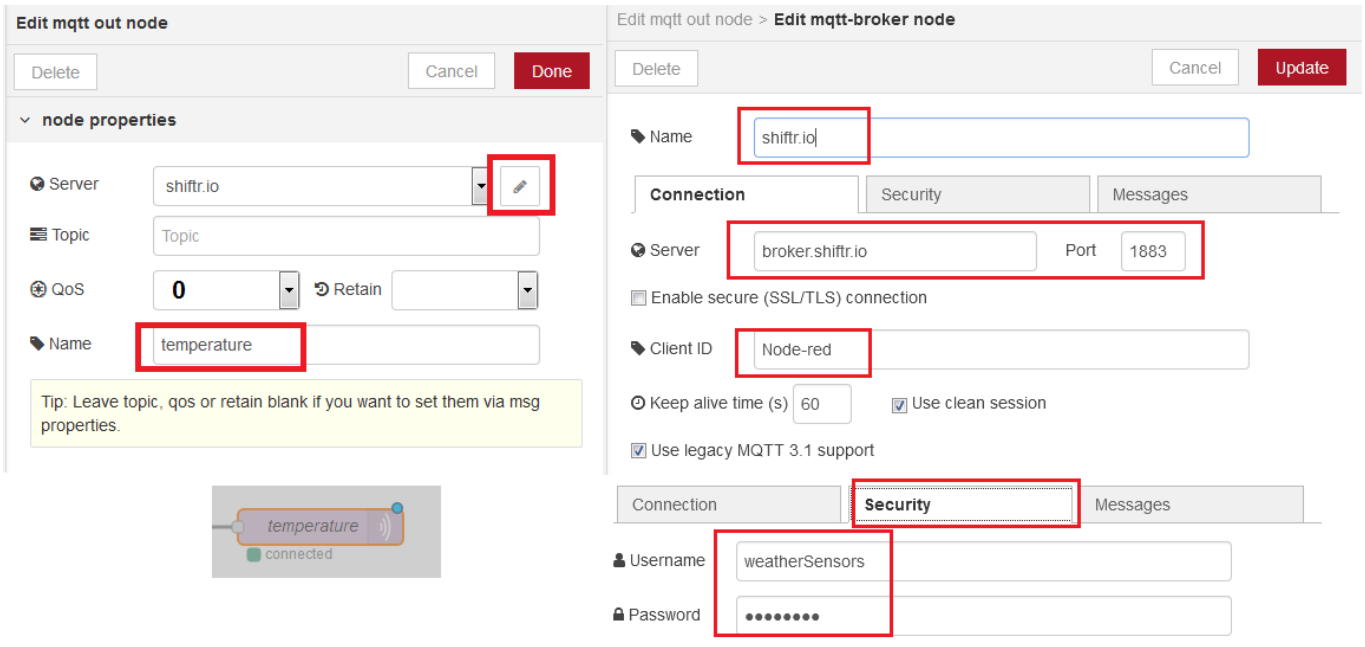
```
curl -sL https://deb.nodesource.com/setup_11.x | sudo -E bash -
sudo apt-get install -y nodejs
node -v
sudo npm install -g --unsafe-perm node-red
node-red
```

3.3 1er pas

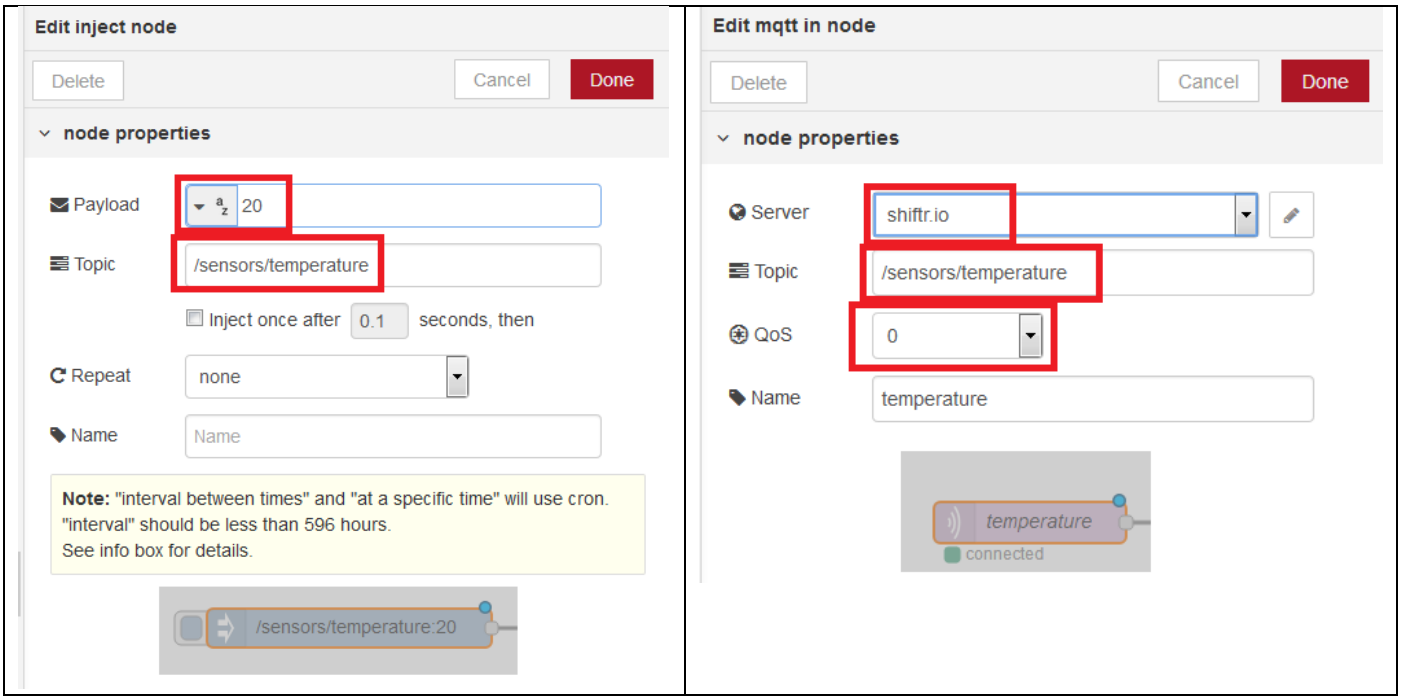
Tracer le diagramme suivant :



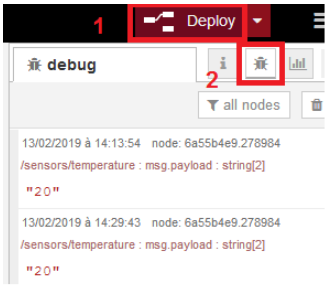
Configuration du broker shiftr.io :

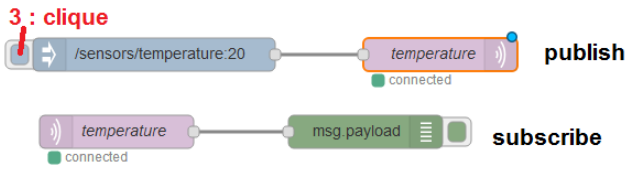


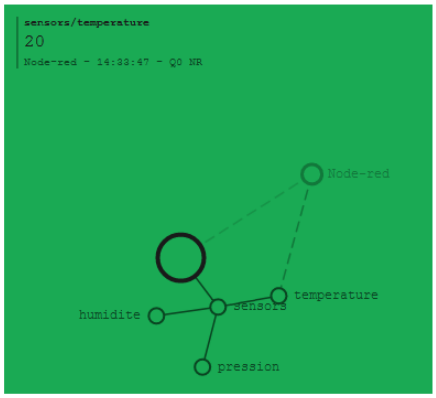
Ici la valeur est fixe, 20° pour tester le fonctionnement.



Test du diagramme :





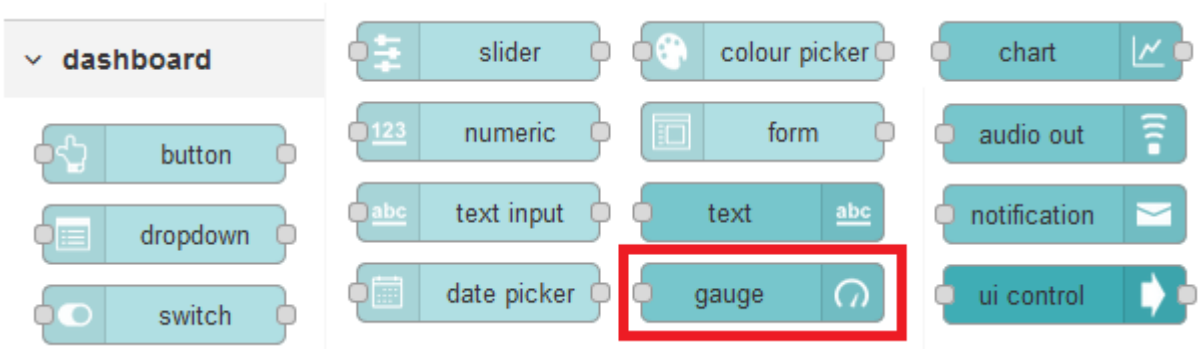


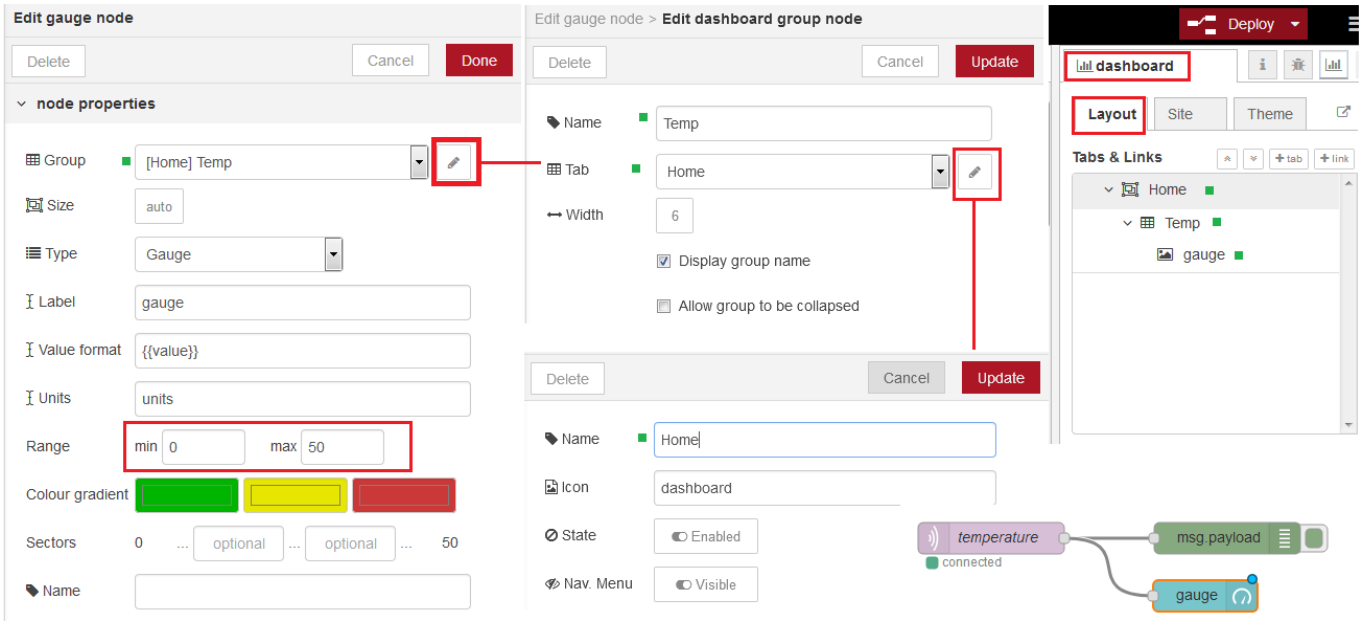
1- Cliquer sur Deploy
2- Puis sur debug pour voir les messages de debug
3- Enfin sur le bouton envoi (symbole inject)

L'information de température 20 ° test envoyé vers le broker, puis revient.

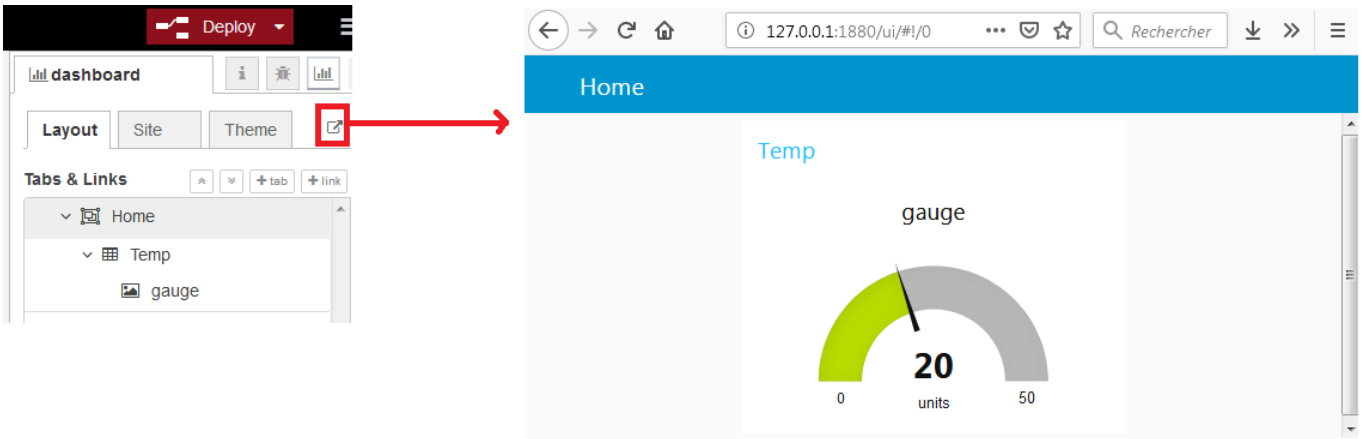
3.4 Gestion graphique

Utilisation d'un vu mètre (gauge)





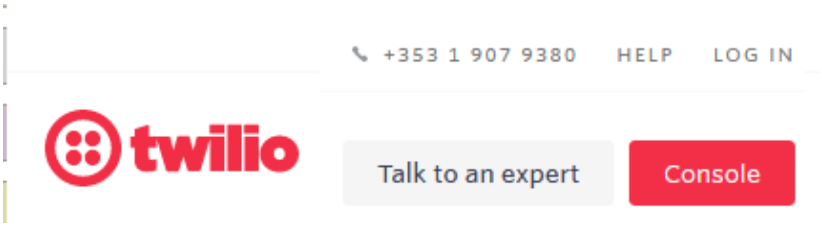
Ou se trouve le vu mètre une fois que l'on clique sur **Deploy** ?



Une nouvelle fenêtre dans le navigateur s'ouvre avec le vu mètre.

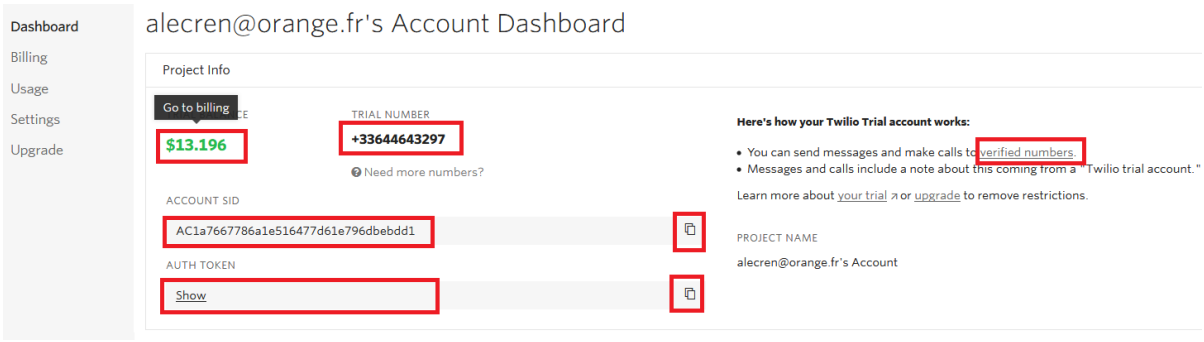
3.5 Envoi de sms

Inscription sur le site https://www.twilio.com/

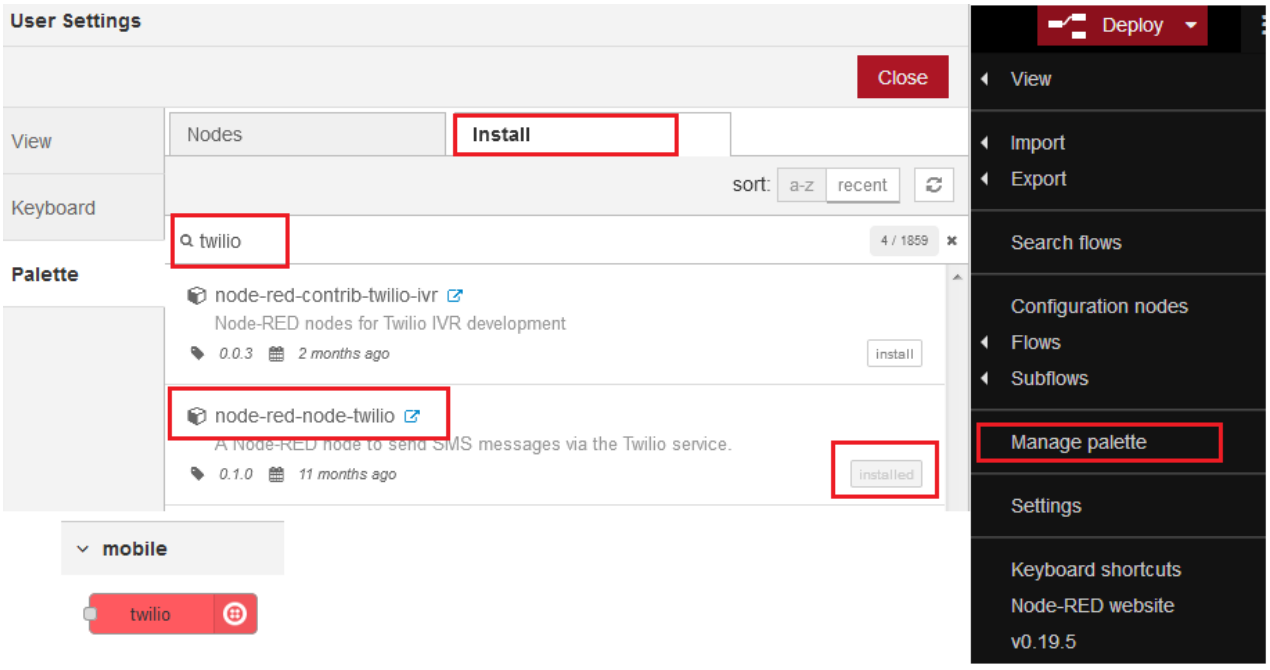


Trial version : 200 sms gratuits, puis payant : 7,6 cts /sms

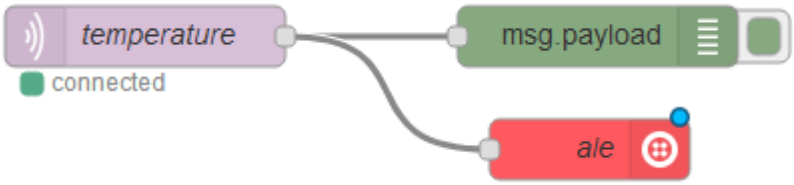
Exemple de dashboard une fois l'inscription terminée
Twilio vous fourni un numéro de téléphone : ici Trial number
Il est possible d'envoyer des sms que sur des numéros de téléphones vérifiés : (le vôtre lors de l'inscription).
En vert le crédit sms restant

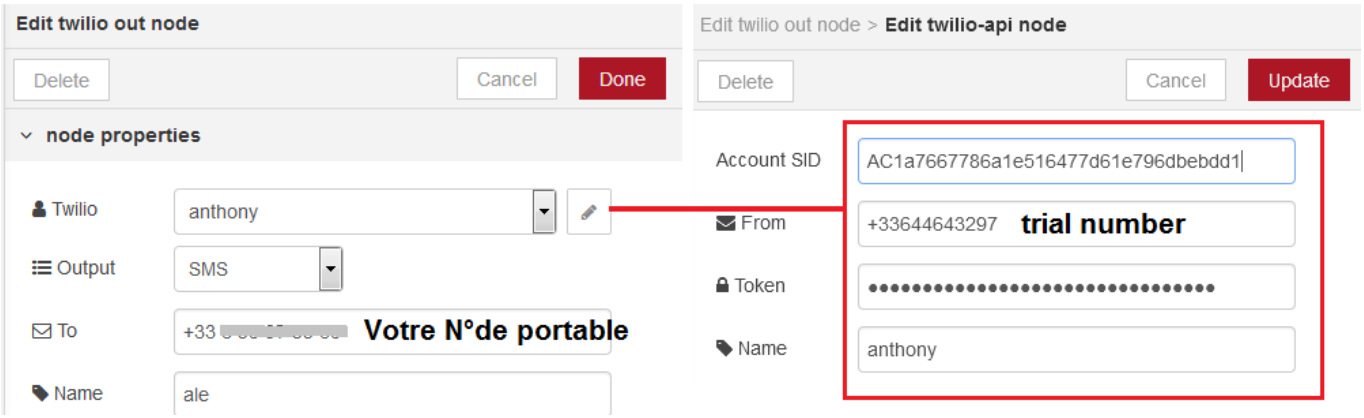


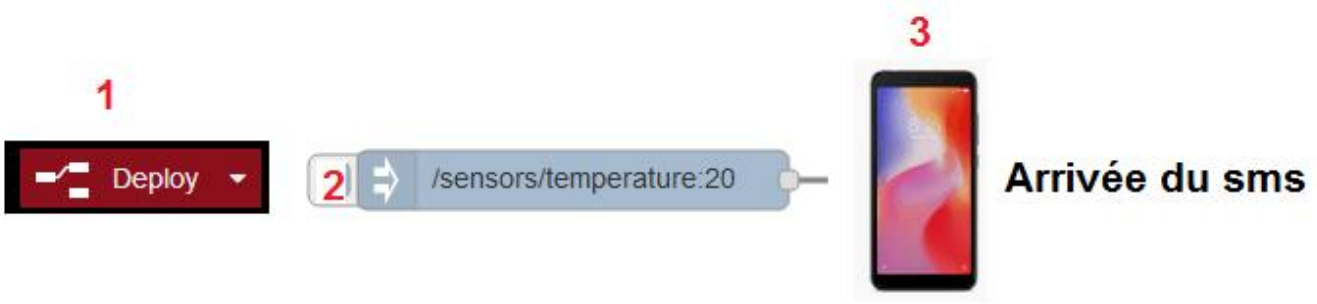
Installation dans Node-red



Configuration de twilio







Il existe de nombreuses possibilités avec Node-red

Ce document n'a pas pour but d'expliquer toutes les possibilités mais de démarrer rapidement

Liens :

http://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/8054/8054-objets-communicants.pdf

https://www.youtube.com/results?search_query=node+red