BASES de données

TD3 nodered

https://www.youtube.com/playlist?list=PLYCQztVKrSmDaZG0bwnbd0PfgdXGe9Q2c

https://noderedguide.com/

https://nodered.org/docs/tutorials/

https://www.youtube.com/watch?v=kHtzghrNzAE

1 Node-red : premiers pas

1.1 Introduction

Node-RED est un outil de programmation visuelle open-source, basé sur **Node.js**, qui permet de créer rapidement des applications en reliant des blocs fonctionnels appelés **nœuds**.
Il est particulièrement utilisé dans les domaines de l'**IoT**, de l'**automatisation**, de la **supervision**, et de l'**intégration de services**.

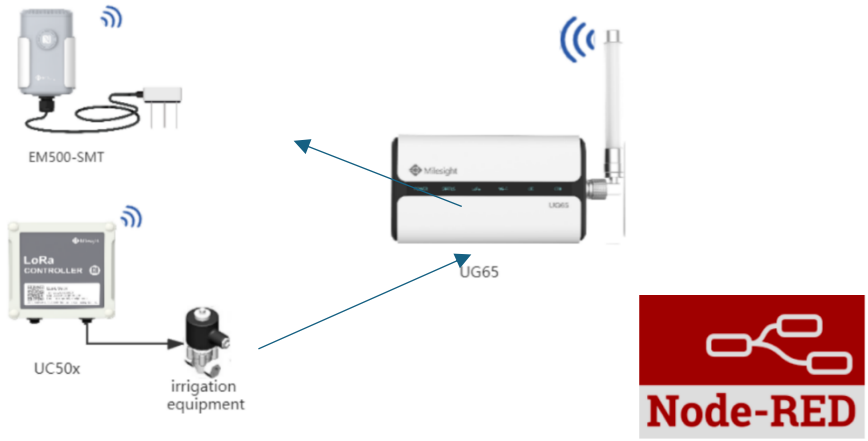
L'idée centrale de Node-RED est simple :

👉 *au lieu d'écrire beaucoup de code, on construit des flux logiques en reliant graphiquement des nœuds entre eux.*

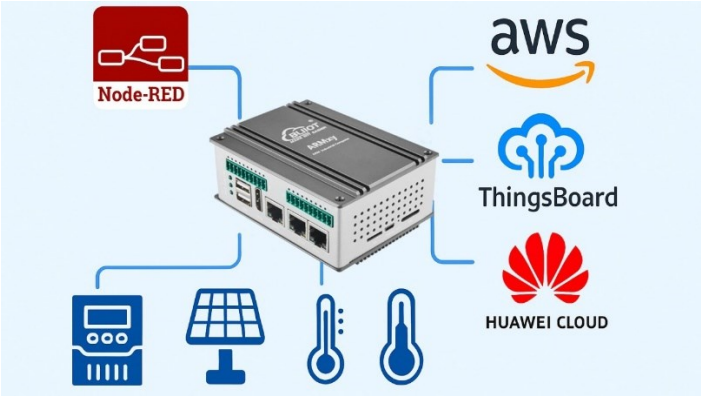
Node-RED est utilisé dans de nombreux contextes (industriels et tertiaires)

- ✓ automatisation industrielle et domotique
- ✓ supervision de systèmes (température, énergie, états machines)
- ✓ communication MQTT, HTTP, Modbus, OPC UA, BACNET
- ✓ prototypage rapide d'applications

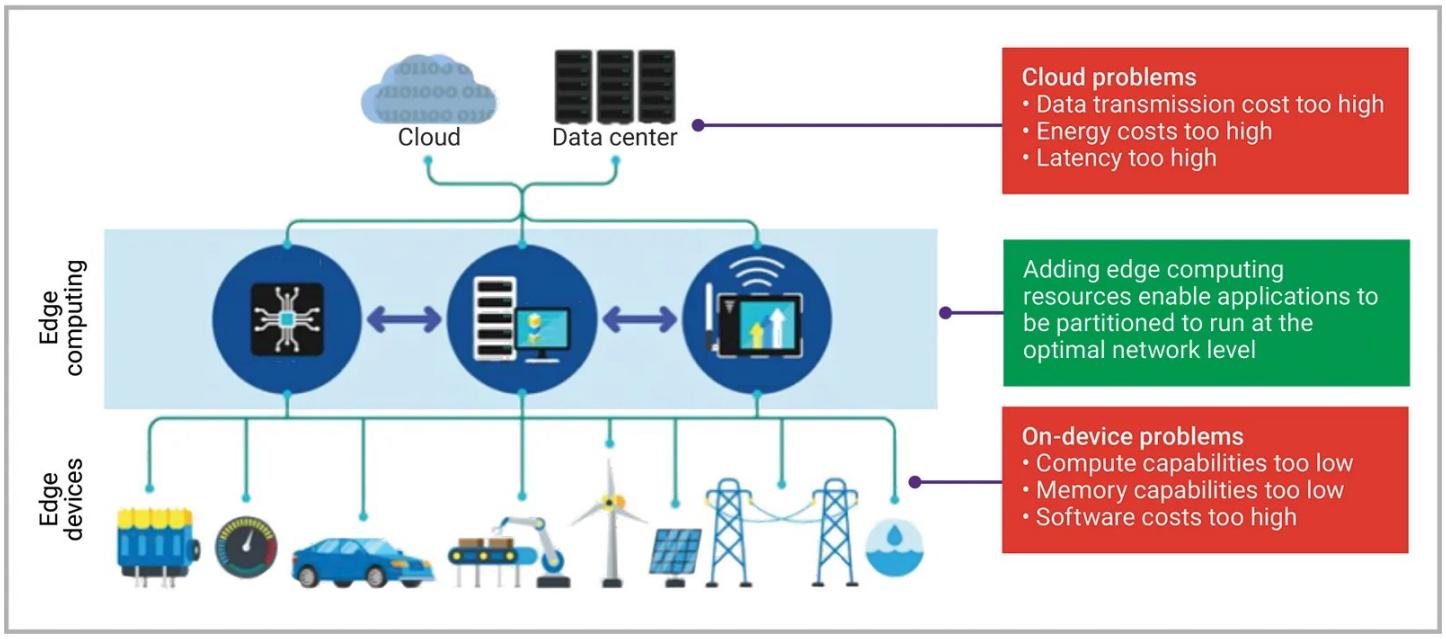
Cas d'usage n1 : traitement d'informations capteurs Lorawan au travers d'une passerelle avec paramétrage et programmation via nodered.



Cas d'usage n°2 : edge computer pour le traitement et/ou la remontée d'informations sur le cloud (PV, capteurs domotiques, industriel...) avec de nombreuses solutions (wago, sick, Siemens, ...)



Cas général industriel : intégrer des edge computer (en général des modules fanless) intégrant souvent nodered et qui permettent de garder l'information importante en local.



Cas numéro 4 : sans oublier les makers et le do it yourself permettant de domotiser à moindre cout. Solution que l'on peut retrouver aussi au sein des entreprises innovantes.

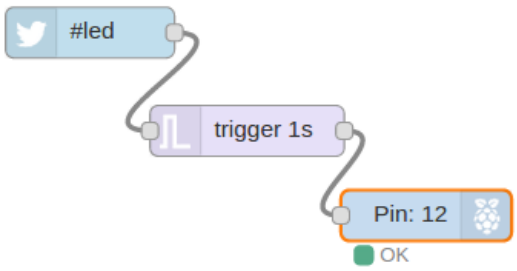
Quelques idées de projets node-red :

https://monraspberry.com/10-idees-de-projets-a-realiser-avec-node-red-sur-votre-raspberry-pi/

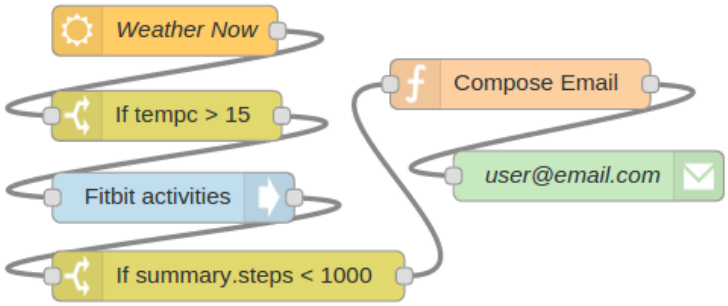
1.2 Des exemples node-red

Avant de commencer, voyons ce qu'est une programmation node-red au tranvers 2 exemples de programmation :

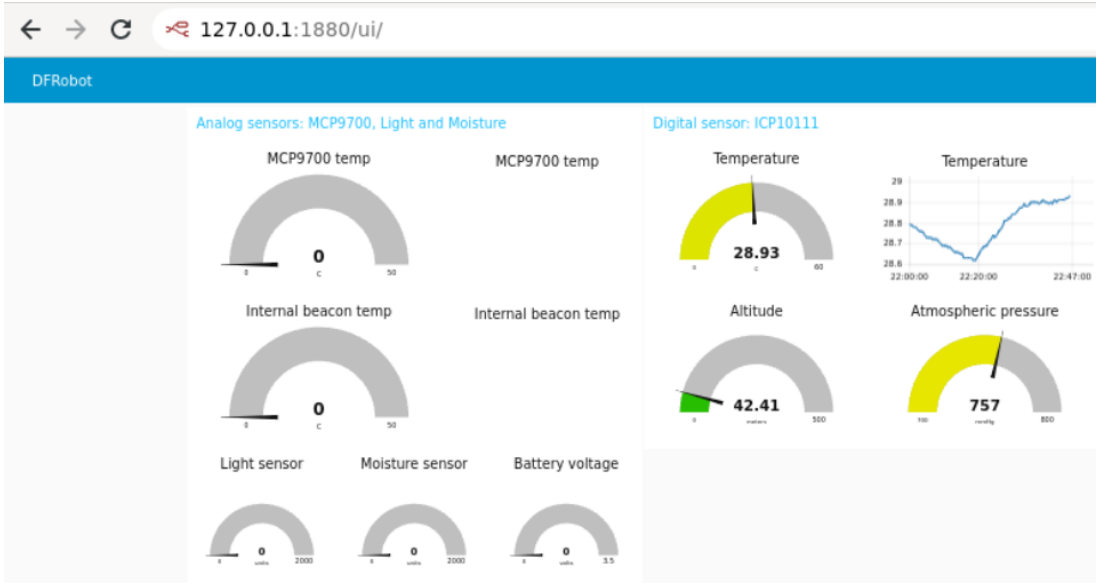
Exemple 1 : A chaque réception d'un tweet sur un sujet précis, allumer une led sur le raspberry pi pendant 1s



Exemple 2 : cette mini-application vient lire les informations de la météo (site web Openweather) sur Nice par exemple et si la température extérieure est supérieure à 15°c et que le nombre de pas sur la montre connectée est inférieur à 1000 m'envoie un mail pour me dire d'aller marcher (pourquoi faire simple quant on peut faire compliqué 😉)

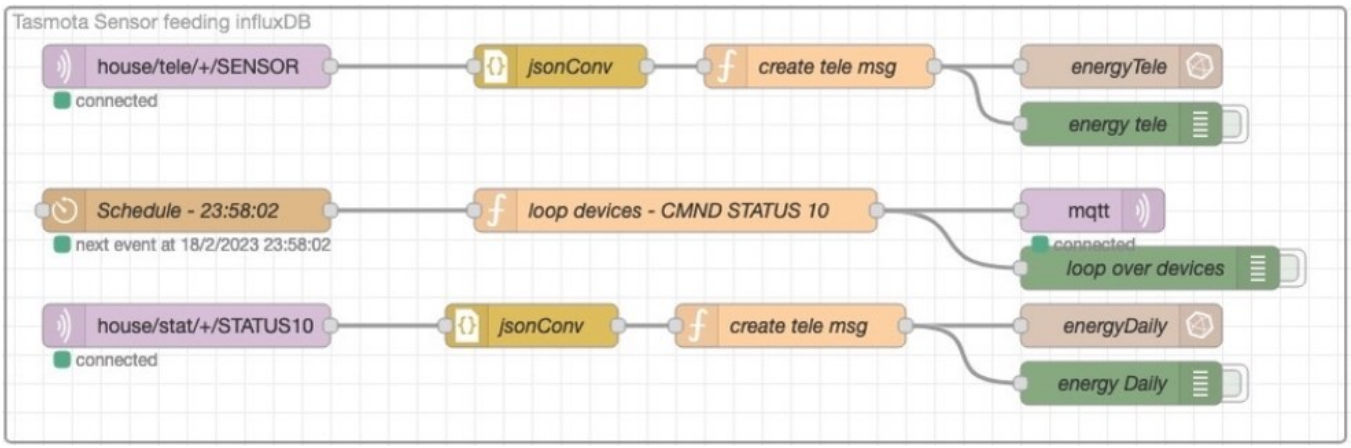


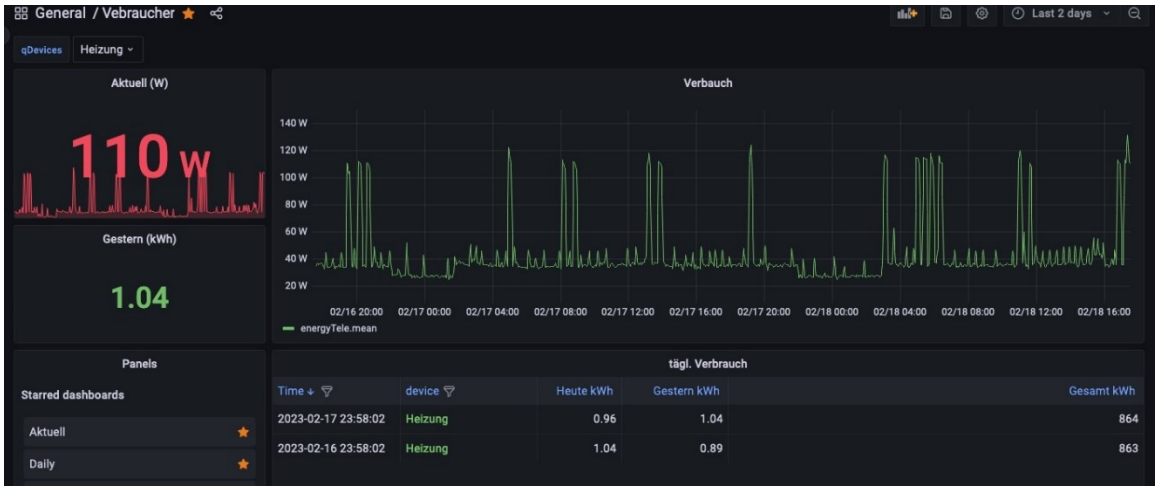
Nodered c'est de la programmation graphique (low code) mais c'est aussi la possibilité de générer une interface graphique, voici un exemple :



La visualisation des données peut aussi être réalisée par grafana avec un transfert de données vers influxdb (base de données temporelle) mais qui pourrait être aussi un base de données mariadb. Voici un exemple de code venant lire des données capteurs et les envoyer vers influxdb (grafana s'occupe alors d'afficher les données enregistrées dans influxdb).

https://flows.nodered.org/flow/9e223d4f816dd15385687bb6bc05a2f6/in/b4yLAVuxd6if





1.3 Lancement de node-red (217.154.16.99)

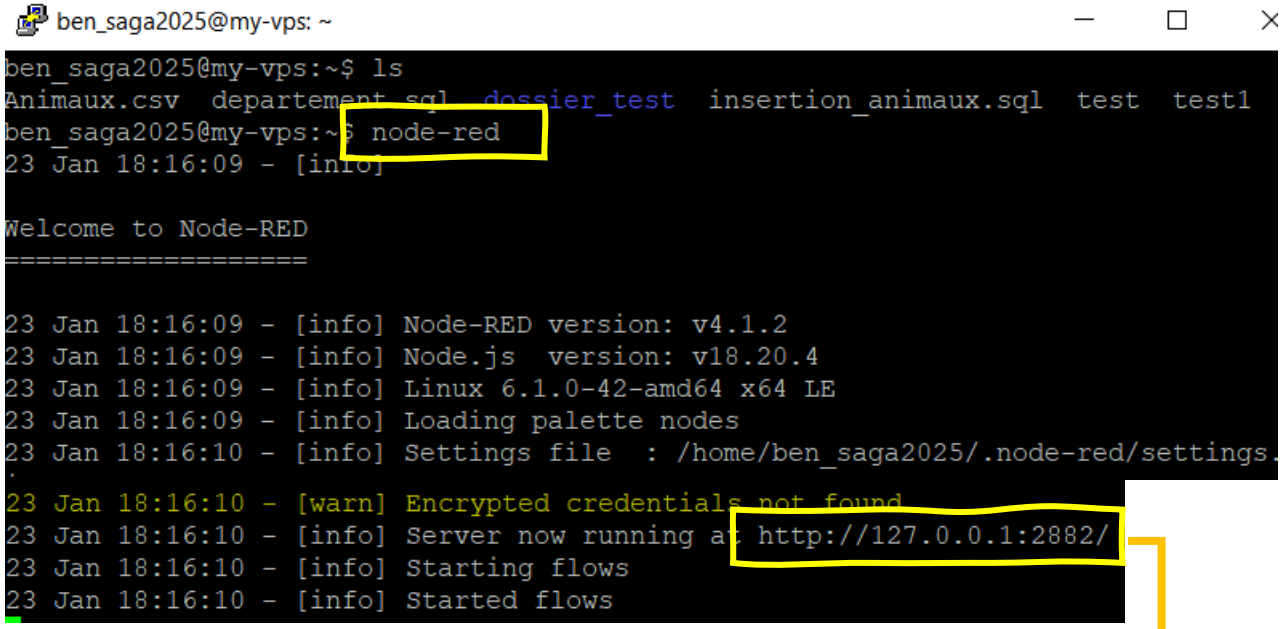
node-red peut s'installer sur n'importe quelle machine. Nous aurions pu utiliser windows : https://nodered.org/docs/getting-started/windows .

Ce n'est pas le choix que nous allons faire : nous avons déjà utilisé notre machine virtuelle ionos à l'adresse **217.154.16.99**, pour tester mariadb, il aurait été dommage de s'arrêter la, donc node-red a été installé sur cette machine et vous avez chacun la possibilité de travailler sur votre compte /home/votre-login. C'est ce que nous allons faire .

Pour lancer node-red, vous allez vous connecter sur 217.154.16.99 avec votre login et mot de passe (exemple pour ben_saga2025)

Travail à faire : ouvrir un terminal putty sur votre compte, lancer node-red puis se connecter via un navigateur

Pour lancer nodered, il suffit de taper la commande **node-red**



Ouvrir le navigateur et aller sur le port de votre compte :

Login : votre login (ici barrat2025) et mot de passe **1234**

Information sur le mot de passe utilisateur 1234 et sa modification éventuelle :

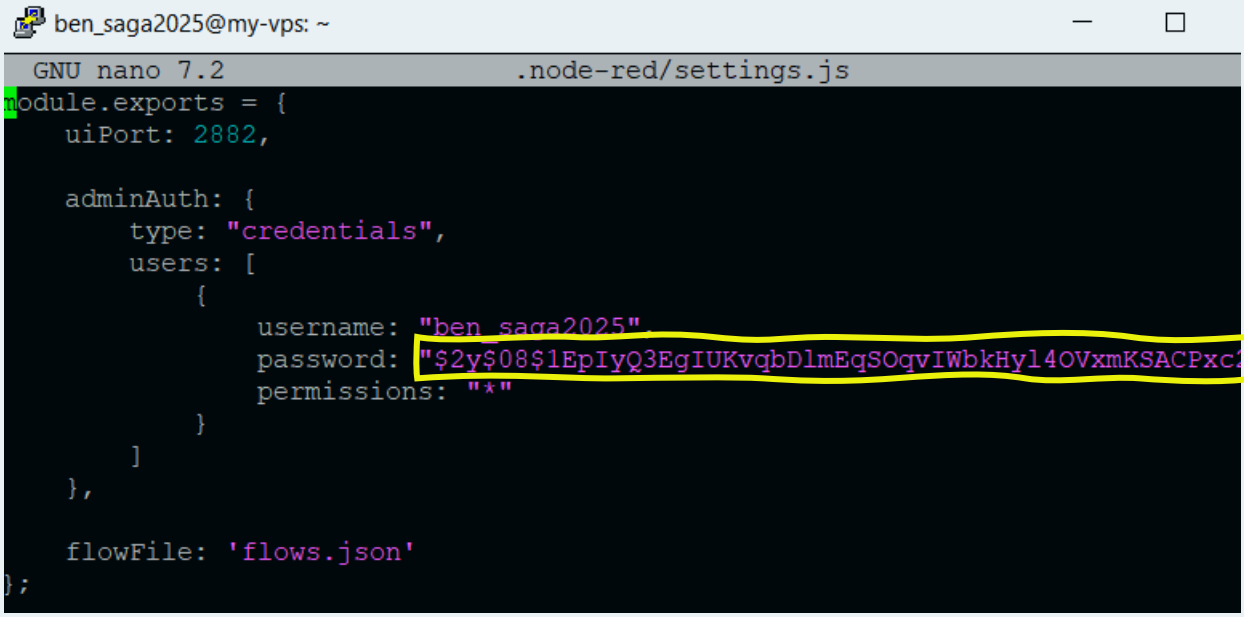
Si vous désirez changer le mot de passe (pour information), il faut lancer la commande node-red admin hash-pw (mot de passe géré par node-red) comme l'exemple ci-dessous (un mot de passe vous est demandé, ici XXXX mais le mot de passe ne s'affiche pas à l'écran) le résultat du hashage est ensuite affiché

```
ben_saga2025@my-vps:~$ node-red admin hash-pw
Password: XXXX
$2y$08$vzWj6NAo16oQ0vVDTLcokuJurPWgEHCGUhCxMOUhxZTlrWoEqi4Ey
```

Pour mettre à jour le mot de passe node-red, aller sur ./node-red et ouvrir settings.js et copier le nouveau hash

```
ben_saga2025@my-vps:~$ cd ~/.node-red
ben_saga2025@my-vps:~/.node-red$ nano settings.js
```

Vous pouvez alors copier (vous pouvez aussi utiliser winscp) le nouveau mot de passe haché).

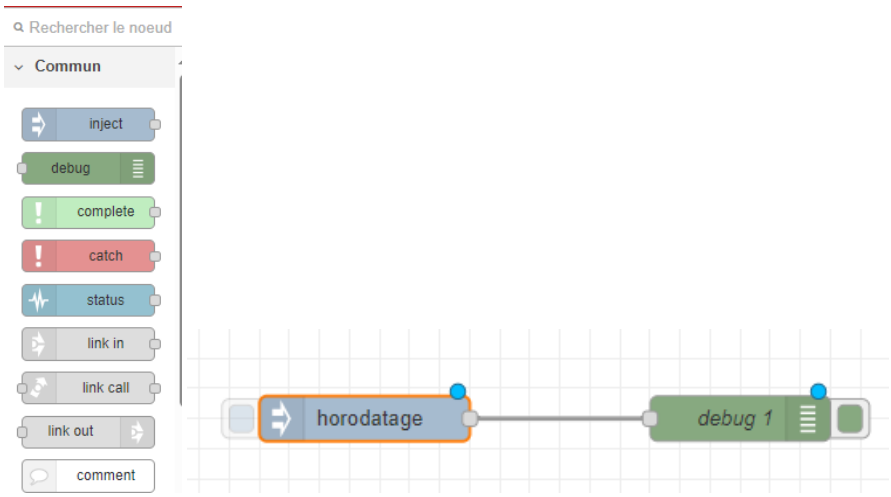


1.4 Premiers tests

Ce premier programme va nous permettre de nous familiariser avec l'environnement node-red.

1. Modifier le nom du flux (double clic sur flux1)

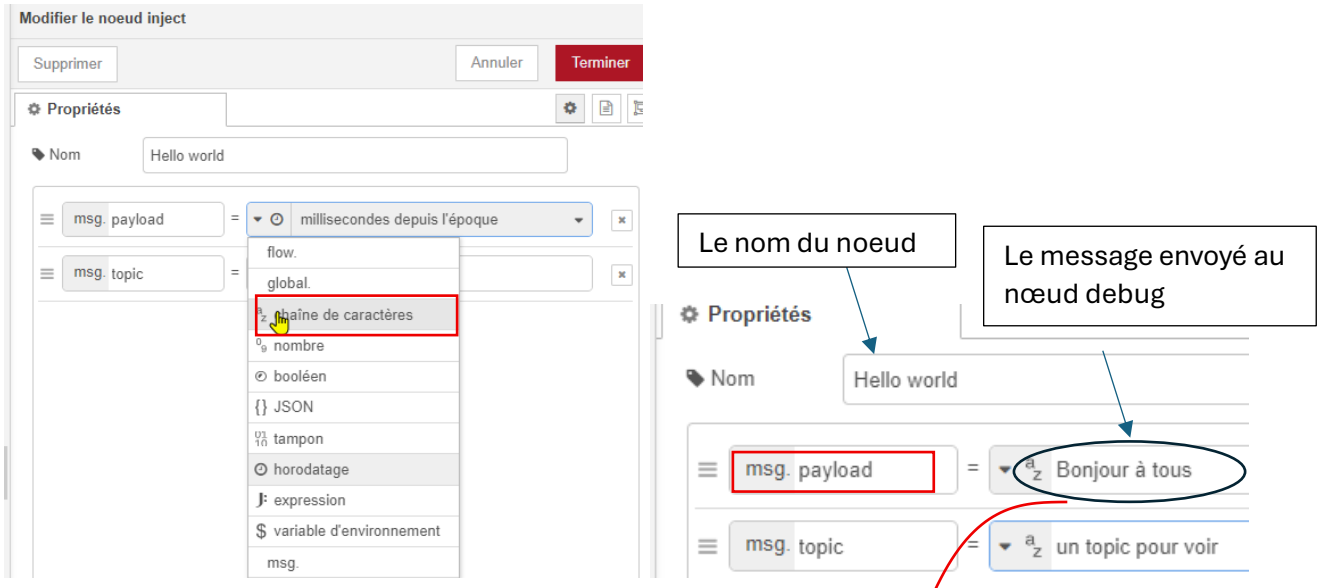
2. Ajouter inject et debug (glisser/déposer) qui sont les 2 premiers nœuds que nous allons utilisés et qui sont très souvent utilisés pour créer le premier programme Hello world ! (un classique)



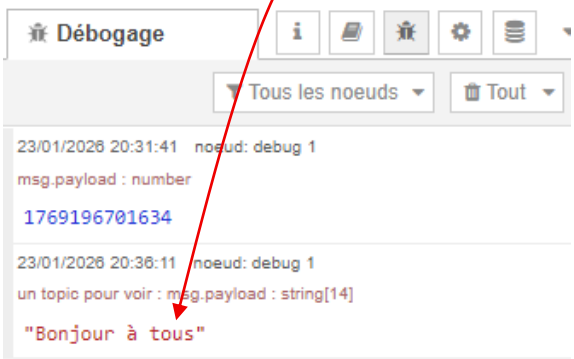
3. Déployer (il faudra à chaque modification (point bleu) déployer le programme pour le tester (équivalent à une compilation en C)

4. Cliquer sur Horodatage

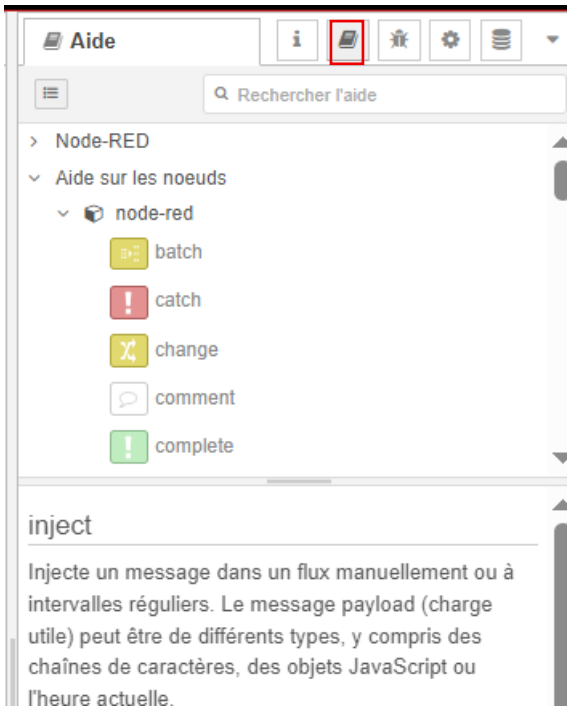
5. Nous pouvons modifier le type de message envoyé par horodatage et choisir chaine de caractère pour notre premier message Hello world

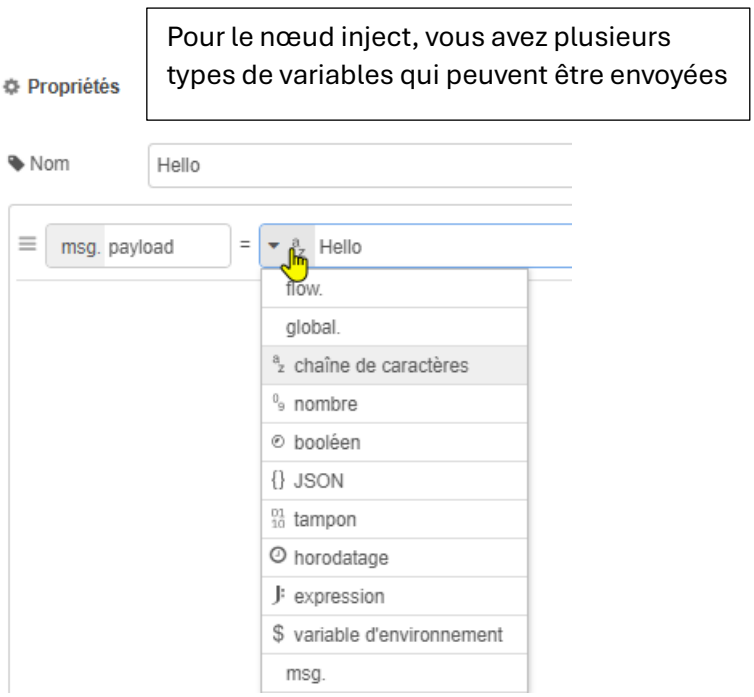


6. Déployer

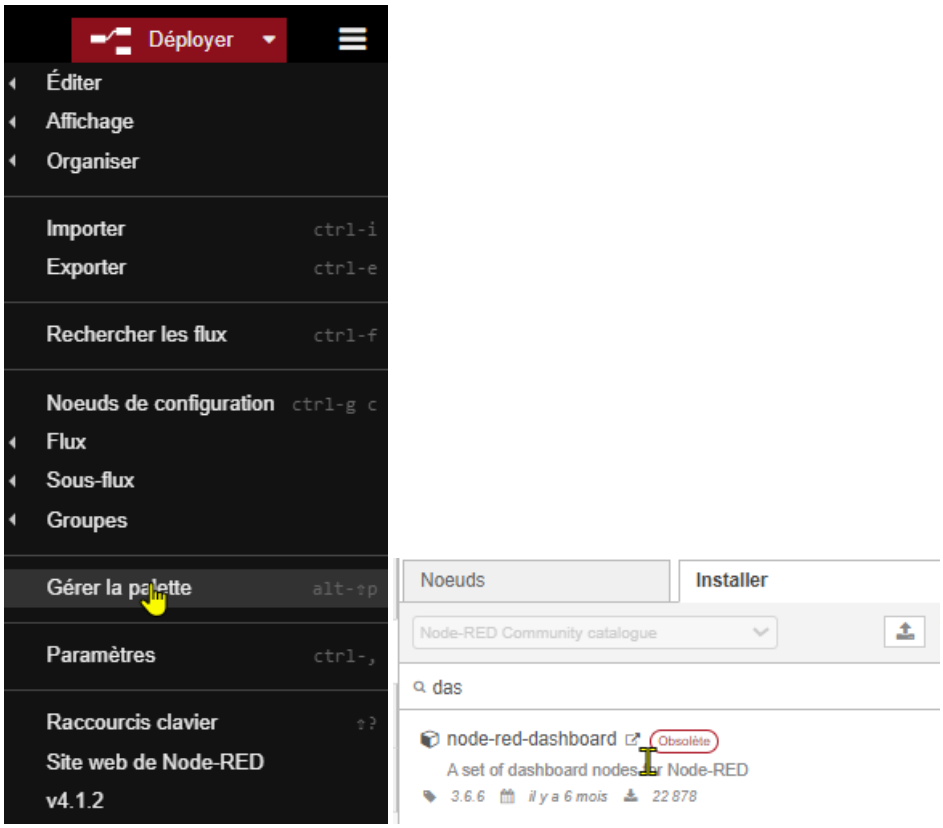


Pour avoir de l'information sur les différents nœuds, on pourra aller sur l'aide



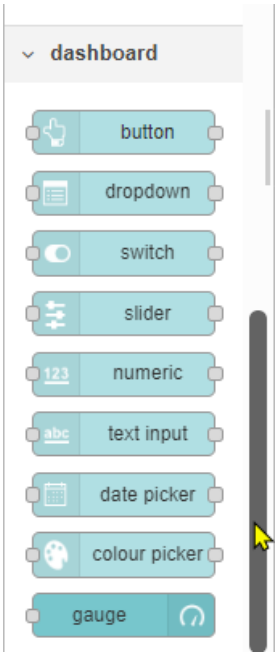


Node-red c'est aussi la possibilité d'ajouter des nœuds, commençons par le nœud dashboard pour pouvoir créer une interface graphique

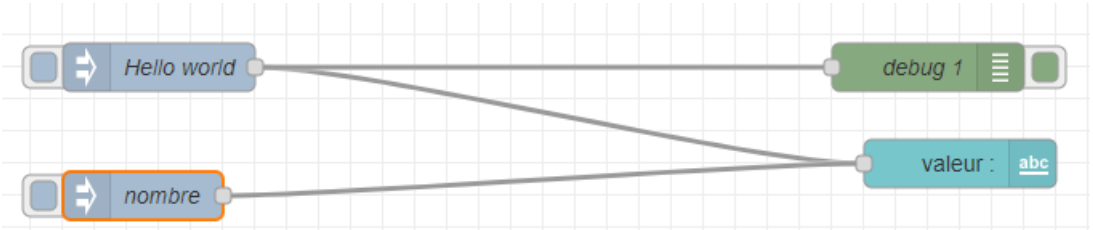


Il existe une nouvelle interface flowfuse dashboard (Dashboard 2) qui a l'inconvénient de prendre plus de place mémoire (et comme nous avons des ressources limitées nous restons avec l'ancienne interface).

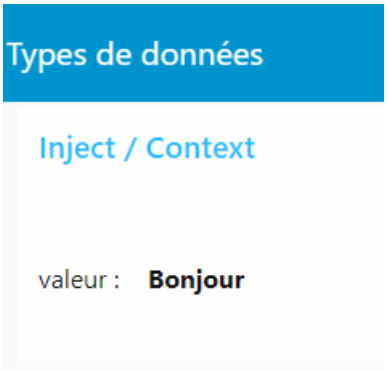
Une nouvelle palette a été installée :



Notre nouveau programme : le nœud inject noté nombre envoie la valeur 10, le nœud Hello World envoie le texte bonjour à tous. Tester le clic sur l'un ou l'autre sur la fenêtre debugs puis sur la page web 217.154.16.99 :1882/ui



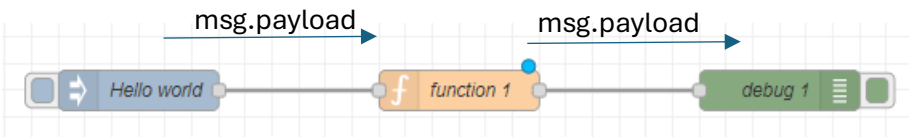
Cliquer sur nombre et hello world



1.5 Fonctionnement de node-red

Nous avons affaire à une programmation évènementielle. Un evènement déclenche l'exécution d'un flux.

Chaque nœud envoie la donnée msg.payload (mais plus généralement msg) de nœud en nœud. Cette variable est modifiée par chaque nœud.



La fonction peut modifier le message envoyé :

```
msg.payload += " world";
return msg;
```

ou bien créer un nouveau message

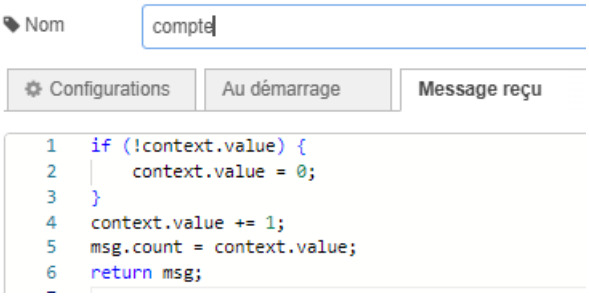
```
let newMsg = { payload: "Hello World" };
return newMsg;
```

(https://noderedguide.com/node-red-lecture-5-the-node-red-programming-model/)

Dans le cas ou l'on veut par exemple avoir un compteur, ce type de programmation ne peut pas fonctionner (un nœud se reboucle sur lui-même -> impossible). Il existe donc une solution pour contourner ce problème : les variables de contexte

Contexte : c'est l'équivalent d'une variable globale pour le flux. Ce qui permet de créer par exemple un compteur

```
if (!context.value) {
  context.value = 0;
}
context.value +=1;
msg.count = context.value;
return msg;
```



Dans cet exemple le payload n'a pas été changé (Hello), on a par contre ajouter un champs count au message.

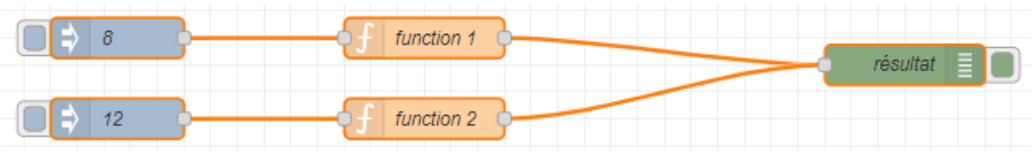
Contexte.global, c'est la même chose mais cette fois, c'est une variable globale partagées par tous les flux

Function 1 :

```
context.global.startTime = new Date().getTime();
return msg;
```

Function 2 :

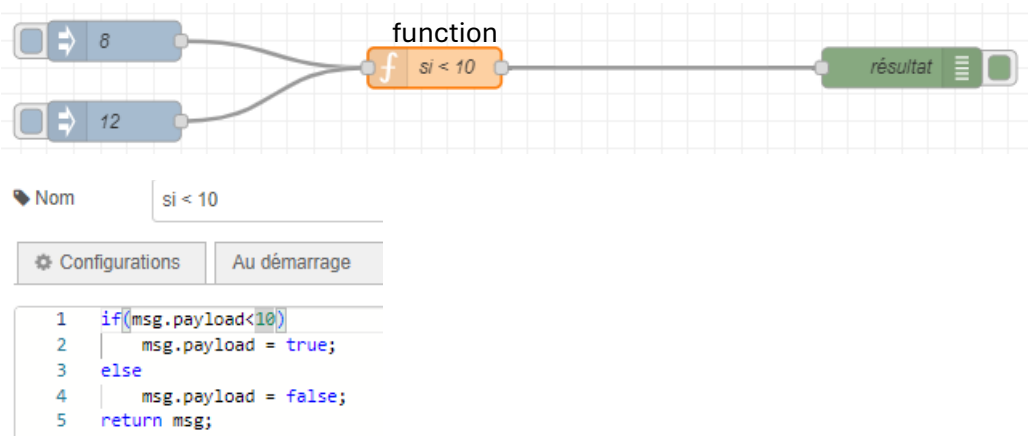
```
let currentTime = new Date().getTime();
let timeElapsed = (currentTime - context.global.startTime) / 1000;
msg.payload = "Time elapsed is: " + timeElapsed + " s";
return msg;
```



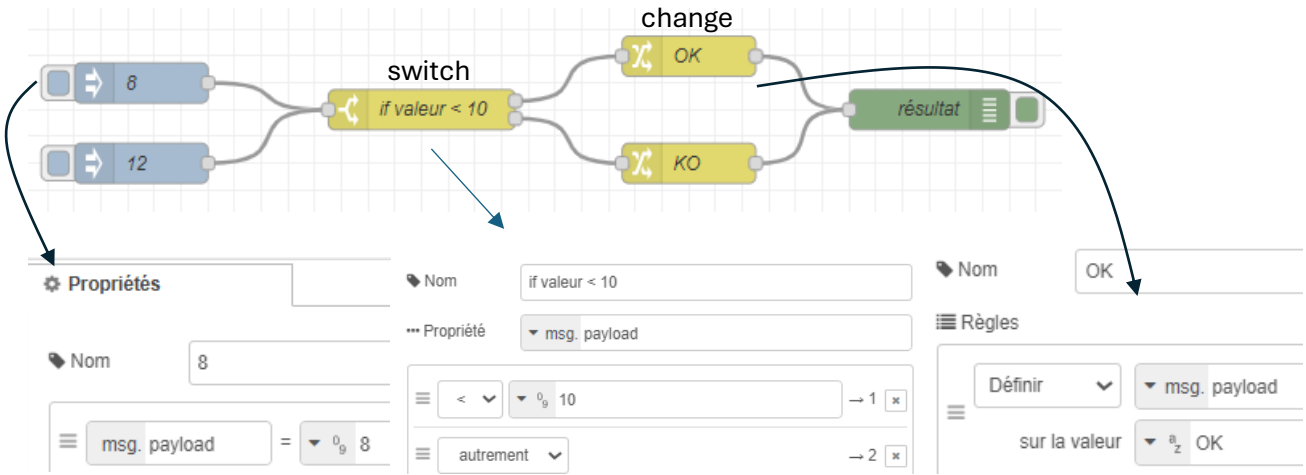
La mesure de temps se fait entre l'appui sur 8 et l'appui sur 12.

1.6 Exemple 1 : faire un if else (test de seuil)

solution 1 : avec function

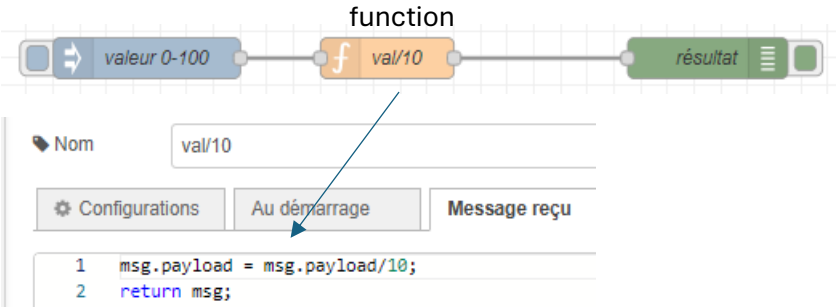


Solution 2 : programmation tout graphique

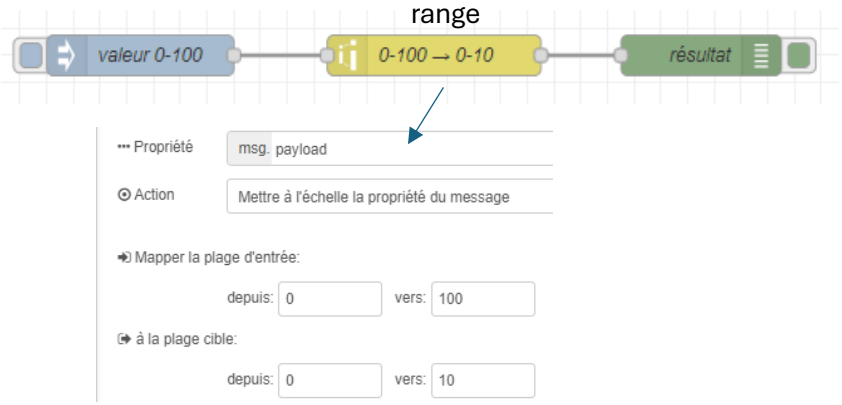


1.7 Faire une mise à l'échelle

solution 1 : avec function

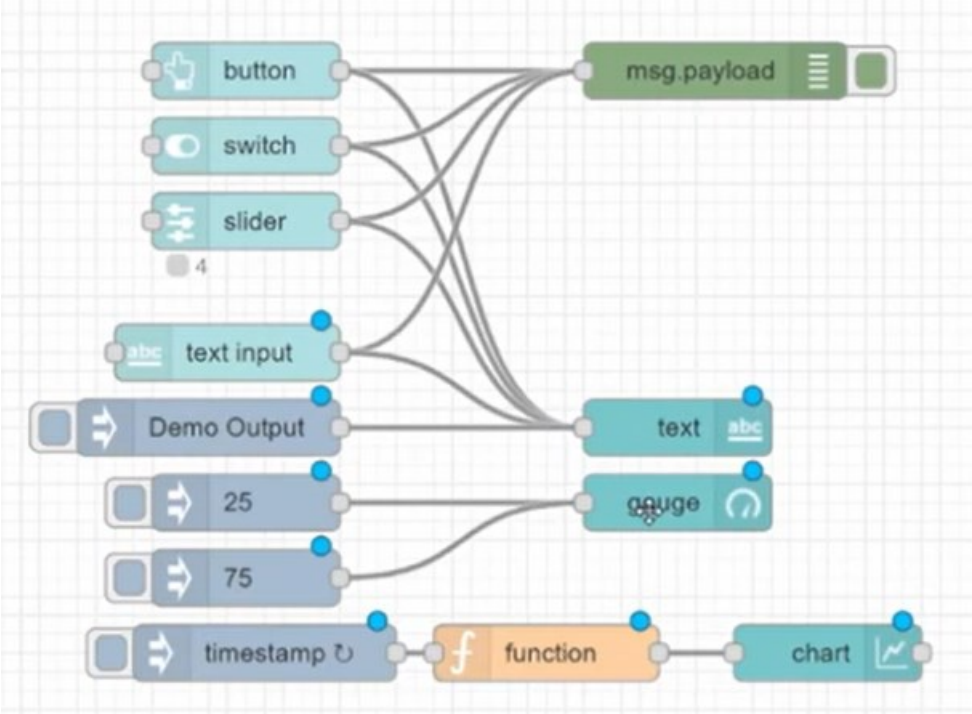


Solution 2 : tout graphique



1.8 Auto-formation dashboard

- Suivre la formation au dashboard sur https://www.youtube.com/watch?v=C0hCQuj1BM4

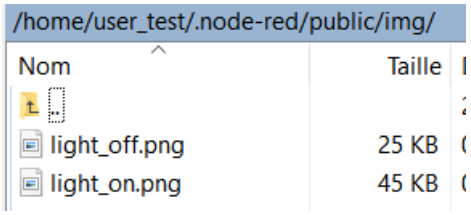


- Tester l'ajout d'un icone sur le button: https://klarsys.github.io/angular-material-icons/

Montrer le résultat à l'enseignant

Nous allons maintenant ajouter 2 images dans le répertoire public.

- **Créer le répertoire public/img dans votre répertoire .nodered puis copier les 2 images du répertoire/mnt/data vers votre répertoire .node-red/public/img**



Sur votre compte dans le répertoire .nodered/settings.js ajouter la ligne ci-dessous, ce qui va permettre à node-red de pouvoir afficher les fichiers dans ce répertoire.

```
    httpStatic: '/home/votre-login/.node-red/public',
```

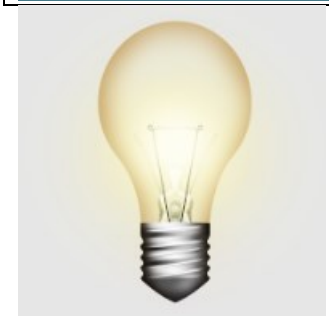
/home/user_test/.node-red/settings.js – ionos user_test – Éditeur – WinSCP

Encodage

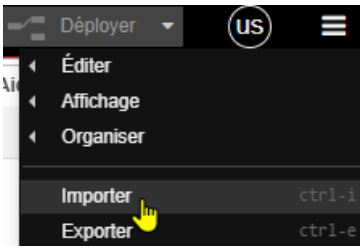
```
module.exports = {
    uiPort: 1880,
    httpStatic: '/home/user_test/.node-red/public',
    adminAuth: {
        type: "credentials",
        users: [
            {
                username: "user_test",
                password: "$2y$08$1EpIyQ3EgIUKvqbDlmEqSOqvIWbkH
                permissions: "*"
            }
        ]
    },

    flowFile: 'flows.json'
};
```

- Relancer node-red (CTR+C puis node-red) dans la console putty.
- Tester que vous pouvez accéder aux 2 images sur le navigateur

```
http://217.154.16.99:1880/img/light_on.png
```



Vous allez maintenant importer le fichier json




```
[{"id":"44dd4a206010148c","type":"ui_template","z":"78b49e2e67736c6c","group":"group_lampe","name":"Lampe","order":7,"width":6,"height":6,"format":"<img ng-if=\"msg.payload === true\" src=\"/img/light_on.png\" width=\"120\">\n<img ng-if=\"msg.payload === false\" src=\"/img/light_off.png\" width=\"120\">\n","storeOutMessages":true,"fwdInMessages":true,"resendOnRefresh":true,"templateScope":"local","className":"","x":1590,"y":420,"wires":[[]]},{"id":"dbfb2ca4404ea1da","type":"ui_switch","z":"78b49e2e67736c6c","name":"","label":"switch","tooltip":"","group":"group_lampe","order":7,"width":"2","height":"2","passthru":true,"decouple":"false","topic":"topic","topicType":"msg","style":"","onvalue":"true","onvalueType":"bool","onicon":"","oncolor":"","offvalue":"false","offvalueType":"bool","officon":"","offcolor":"","animate":false,"className":"btn-center","x":1370,"y":420,"wires":[["44dd4a206010148c"]]},{"id":"group_lampe","type":"ui_group","name":"Commande lampe","tab":"tab_demo","order":1,"disp":true,"width":"6","collapse":false},{"id":"tab_demo","type":"ui_tab","name":"TP Lampe","icon":"lightbulb_outline","order":1},{"id":"262470571c48d887","type":"global-config","env":[],"modules":{"node-red-dashboard":"3.6.6"}}]
```

- Tester le programme

Qu'est-ce que le format json?

JSON = **JavaScript Object Notation, c**'est un **format de données** utilisé pour **échanger des informations** entre ordinateurs ou applications.

- **Lisible par les humains**
- **Facile à traiter par les machines**

C'est très utilisé pour :

- ✓ API web (ex. récupérer des données météo)
- ✓ Configurations (ex. Node-RED flow)
- ✓ Stockage léger (ex. fichiers .json)

JSON est basé sur **deux structures principales** :

a) **Objet {}** → collection de paires clé/valeur

```
{
  "nom": "Alice",
  "age": 25,
  "ville": "Paris"
}
```

- nom, age, ville → **clés**
- "Alice", 25, "Paris" → **valeurs**

⚠ Les **clés doivent toujours être entre guillemets** " "
Les valeurs peuvent être :

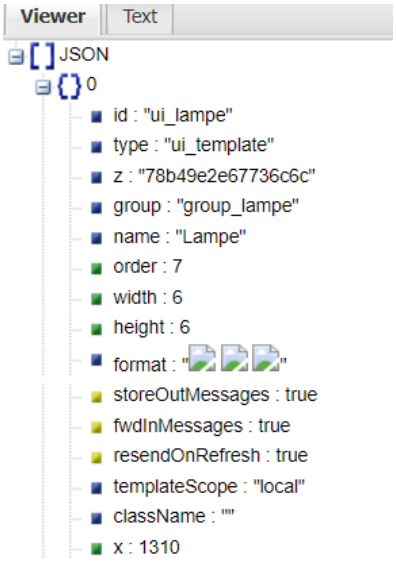
- ✓ chaîne "texte"
- ✓ nombre 123
- ✓ booléen true / false
- ✓ null
- ✓ tableau []
- ✓ objet {}

b) **Tableau []** → liste de valeurs ou objets

```
[
  {"nom": "Alice", "age": 25},
  {"nom": "Bob", "age": 30}
]
```

- Une **liste de deux objets**
- Chaque objet a ses propres clés et valeurs

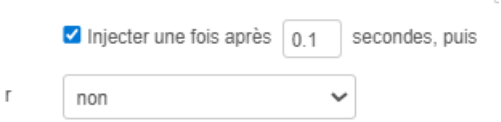
Pour mieux comprendre ce qu'est un fichier json, aller sur https://jsonviewer.stack.hu/ et copier le code json du programme node-red que vous venez d'importer.



Dans node-red, expliquer le problème à l'initialisation.

- ➢ Ajouter la ligne <img ng-if="msg.payload === 'init'" src="/img/light_off.png" width="120">

Quel doit être le message à envoyer à l'initialisation pour charger l'image ?



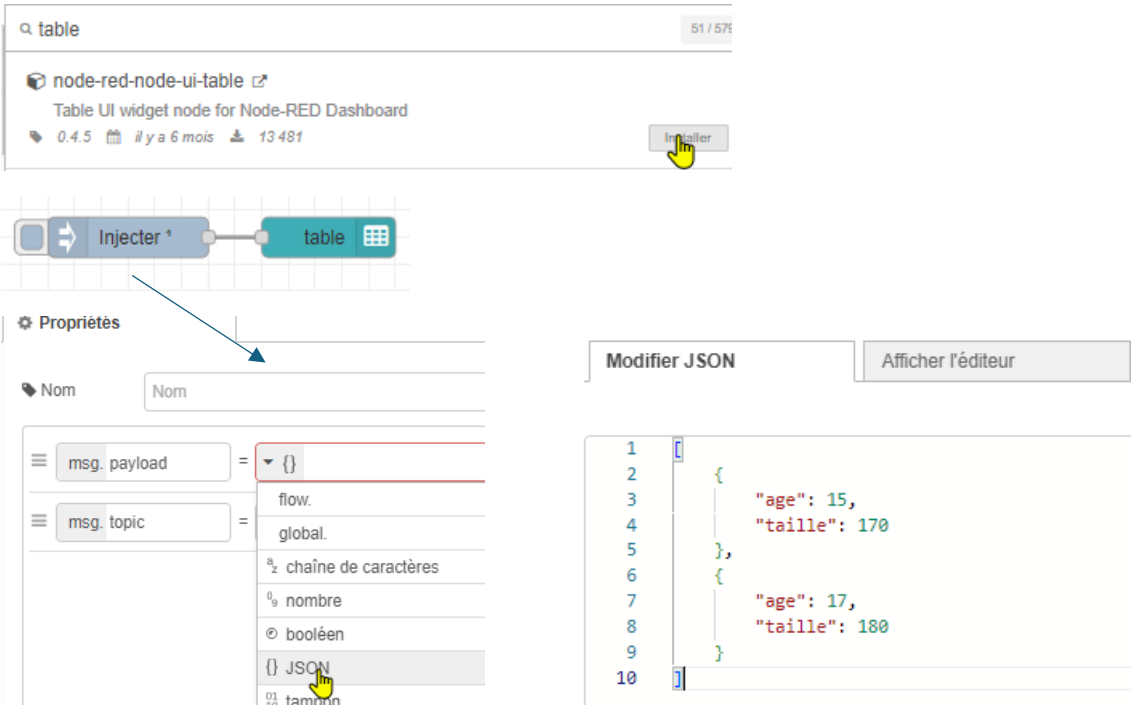
Ajouter et configurer un nœud inject pour répondre au problème

Montrer le résultat à l'enseignant

Tester et analyser le programme ci-dessous :

```
[{"id":"457bd919e61792c0","type":"inject","z":"78b49e2e67736c6c","name":"Envoyer
Alice","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":"","to
pic":"","payload":"Alice","payloadType":"str","x":850,"y":160,"wires":[["43f13c57d5ea55e2
"]]},{"id":"10e4c17bb5acfa92","type":"inject","z":"78b49e2e67736c6c","name":"Envoyer
Bob","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":"","topi
c":"","payload":"Bob","payloadType":"str","x":850,"y":220,"wires":[["43f13c57d5ea55e2"]]}
,{"id":"43f13c57d5ea55e2","type":"switch","z":"78b49e2e67736c6c","name":"Déterminer
couleur","property":"payload","propertyType":"msg","rules":[{"t":"eq","v":"Alice","vt":"s
tr"},{"t":"else"}],"checkall":"true","repair":false,"outputs":2,"x":1080,"y":190,"wires":
[["854831517fe5d91e"],["86f7c25b3954023b"]]},{"id":"854831517fe5d91e","type":"change","z"
:"78b49e2e67736c6c","name":"Texte
vert","rules":[{"t":"set","p":"color","pt":"msg","to":"green","tot":"str"},{"t":"set","p"
:"payload","pt":"msg","to":"{{msg.payload}} est
OK","tot":"str"}],"x":1300,"y":160,"wires":[["c8927b8134a0d668"]]},{"id":"86f7c25b3954023
b","type":"change","z":"78b49e2e67736c6c","name":"Texte
rouge","rules":[{"t":"set","p":"color","pt":"msg","to":"red","tot":"str"},{"t":"set","p":
"payload","pt":"msg","to":"{{msg.payload}} est
KO","tot":"str"}],"x":1300,"y":220,"wires":[["c8927b8134a0d668"]]},{"id":"c8927b8134a0d66
8","type":"ui_text","z":"78b49e2e67736c6c","group":"grp1","order":1,"width":6,"height":1,
"name":"Affichage dynamique","label":"Statut","format":"<span style='color:
{{msg.color}}; font-
size:24px'>{{msg.payload}}</span>","layout":"","className":"","style":false,"font":"","fo
ntSize":"","color":"#000000","x":1520,"y":190,"wires":[]},{"id":"grp1","type":"ui_group",
"name":"Commande","tab":"tab1","order":1,"disp":true,"width":6},{"id":"tab1","type":"ui_t
ab","name":"Démo ON/OFF","icon":"toggle_on"},{"id":"0189921c619fb143","type":"global-
config","env":[],"modules":{"node-red-dashboard":"3.6.6"}}]
```

Pour terminer insérer une table (qui prend en paramètre une entrée inject json), mais avant installer dans palette



Montrer le résultat à l'enseignant

1.9 Exercices :

1. En utilisant 2 inject et un debug, afficher 1 ou 0 si vous appuyez sur l'un ou l'autre inject
2. En utilisant seulement inject, afficher « ceci est un test » toutes les secondes (case Repeter)
3. En utilisant trigger faire afficher « bonjour » puis au bout de 2s « au revoir »
4. Injecter une température en °C , la transformer en fareineit et afficher sur debug
5. Ajouter le slider (valeur entre 0 et 100) et tester maintenant l'affichage debug puis afficher le résultat dans un text ui
6. Utiliser un filter pour n'afficher la température que pour les valeurs comprises entre 20 et 50
7. A chaque clic sur inject le programme incrémente un compteur qui est affiché dans le debug et dans l'ui.
8. Ajouter un bouton pour incrémenter le compteur
9. A chaque clic sur un bouton la couleur et le nom s'inverse
10. Si la valeur entrée est supérieure à 50, le debug affiche trop chaud, sinon il affiche trop froid
11. Créer un nœud qui génère une variable aléatoire comprise entre 0 et 100 et affiche cette valeur sur un camembert et sur un graphe (utiliser Math.random()*100 et .toFixed(2) pour la partie affichage
12. Reprendre le programme de la lampe, mais changer le fond et le texte d'un ui_text (allumé, eteind) et changer aussi la couleur (red, green).

Créer un fichier doc dans lequel vous placer le json et la visu de la réponse à chaque question (à mettre dans la boite de dépôt).

2 communication avec le monde (API Rest)

2.1 Introduction

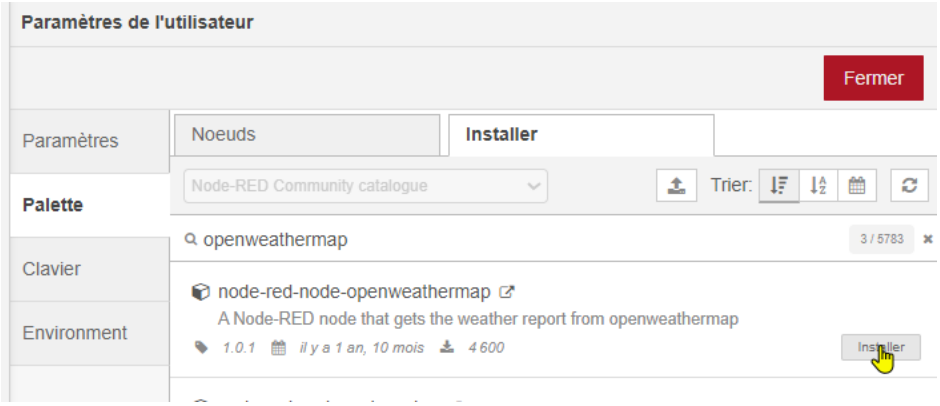
Qui dit bases de données dit données évidemment. Les bases de données ne sont pas seulement utilisées dans les sites web (moodle, google, twiter, whattsapp, ..) mais aussi dans la récoltes de données capteurs (PV, energie, températures, humidité,...) et de leur affichage et traitement.

Afin de bien comprendre l'intérêt de node-red nous allons prendre des exemples de lectures de données. Ces données pourraient venir de capteurs Lorawan, wifi, zigbee, bluetooth,Z-wave, KNX, bacnet,modbus,... ou bien directement d'un site web qui agrège ces données.

2.2 Openweathermap

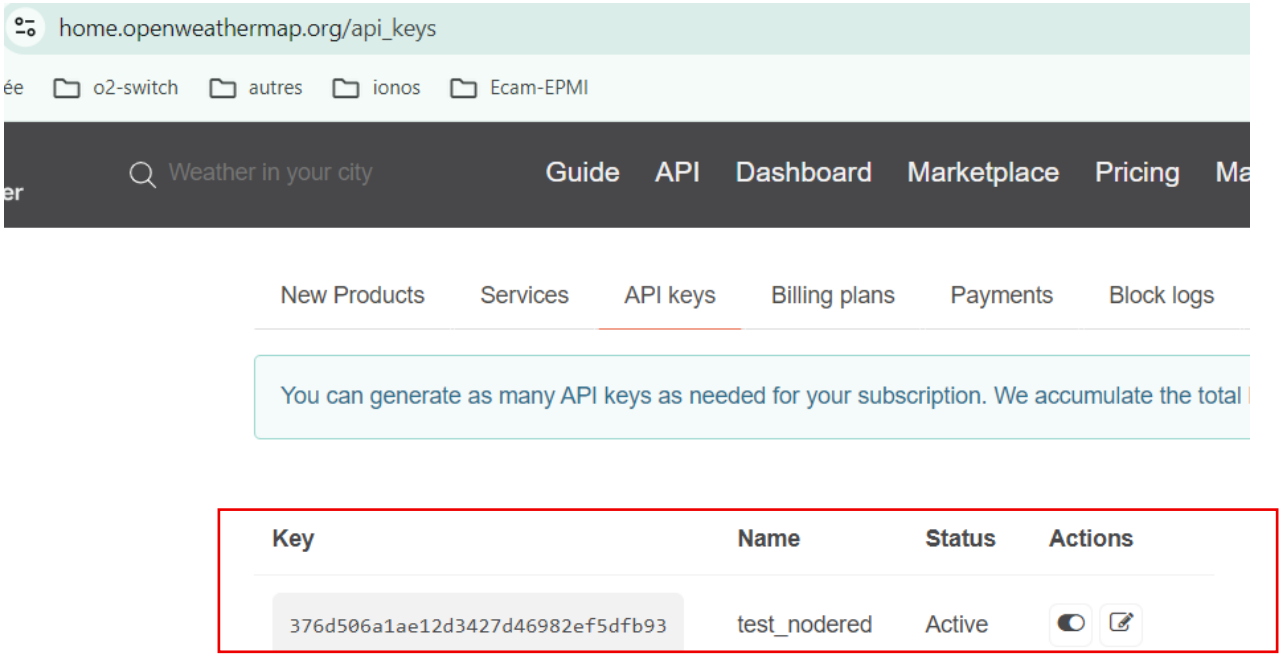
Commençons par un premier exemple classique : openwheather qui est un site permettant d'obtenir des informations météos dans le monde https://openweathermap.org/ , ce site propose une API disponible permettant d'interroger la base de données au travers de requêtes web prédéfinies. Nodered a l'avantage d'être open source et donc il est possible de partager les nœuds déjà programmé, ici le node openweathermap.

- Installer openweathermap



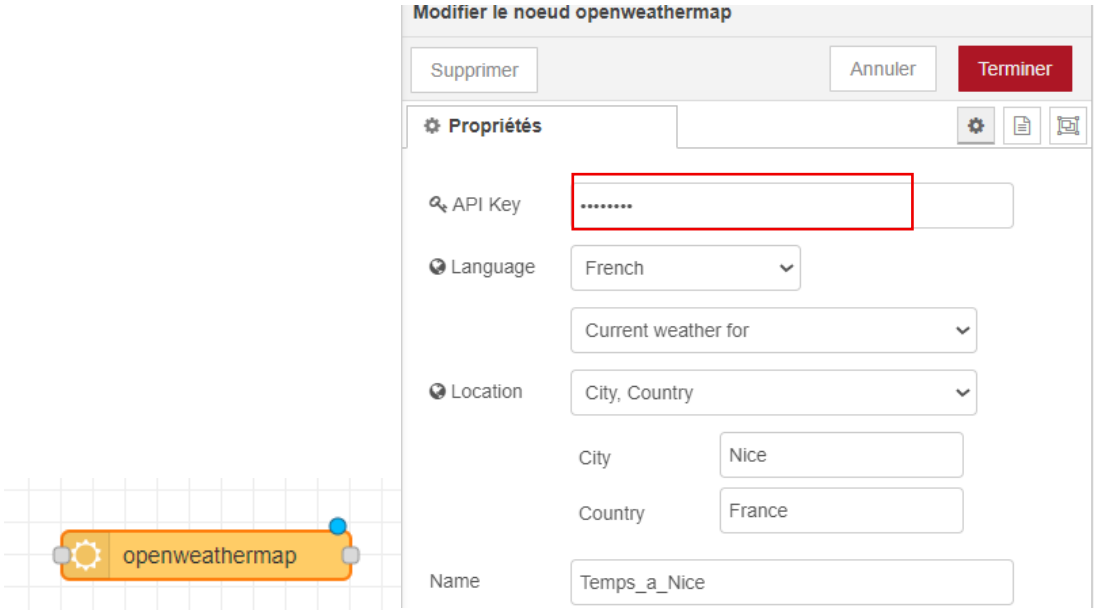
Avant d'aller plus loin, vous aurez besoin de créer un compte afin d'obtenir un token permettant d'interroger la base de données.

C'est ce que j'ai déjà fait, vous pouvez utiliser cette clé (à tester)…

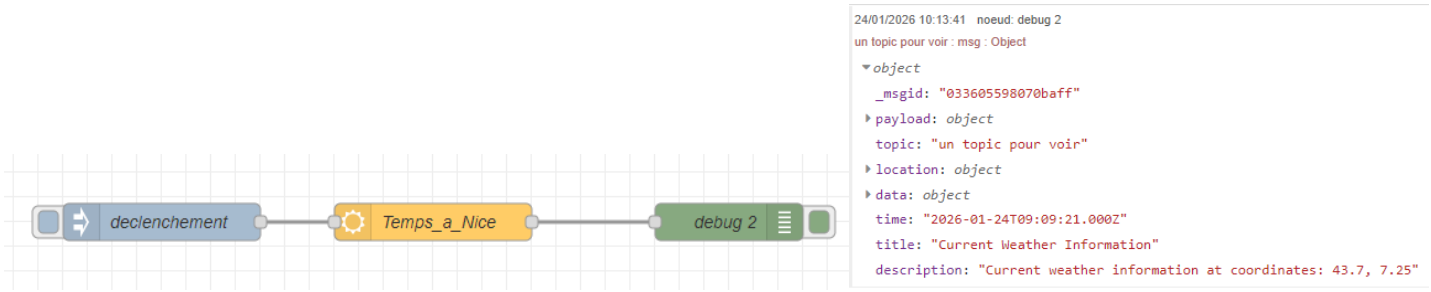


Voici ma clé : 376d506a1ae12d3427d46982ef5dfb93

- Configurer le nœud openweathermap



Il ne reste plus qu'à tester la connexion.



Pour bien comprendre ce qu'est une API copier/coller la ligne ci-dessous dans un navigateur :

```
https://api.openweathermap.org/data/2.5/weather?q=Paris&appid=376d506a1ae12d3427d46982ef5dfb93&units=metric&lang=fr
```

Quel est le résultat généré. Expliquer

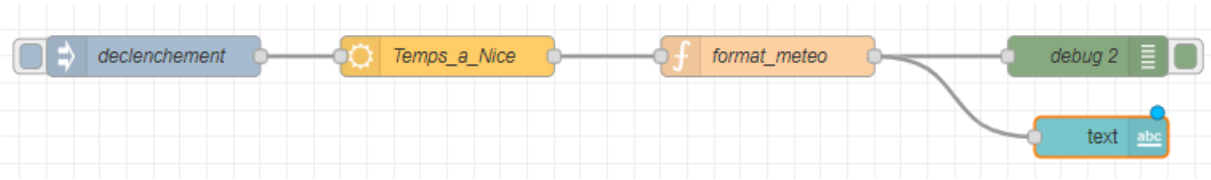
- Tester cette deuxième commande en remplaçant {lat} et {lon} par les coordonnées de Grasse et ajouter la clé à la place de {API key}

```
https://api.openweathermap.org/data/3.0/onecall?lat={lat}&lon={lon}&appid={API key}
```

En utilisant fonction, il est possible de venir extraire dans le fichier json les informations pertinentes.

```
msg.payload = {
    tempc: +(msg.data.main.temp - 273.15).toFixed(1),
} ;
return msg ;
```

- Ajouter les informations humitité et ville et les afficher dans une boite de texte

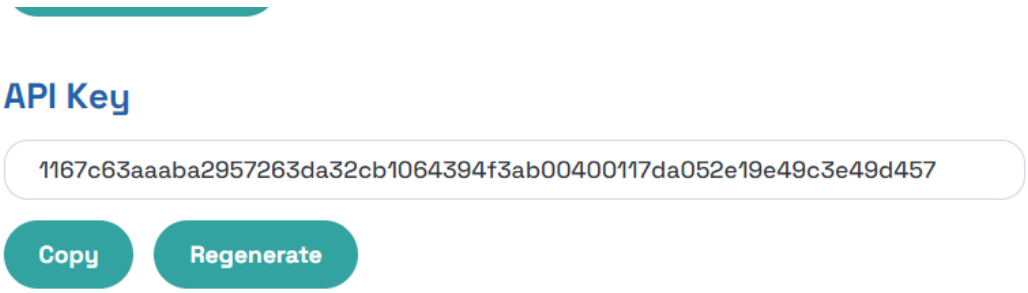


Montrer à l'enseignant

2.3 Extraction de données openaq.org

Prenons un deuxième exemple qui cette fois récolte les informations de pollution aux particules et ozone. Comme tous les sites publics, il faut créer un compte pour obtenir une clé de cryptage.

https://explore.openaq.org/account

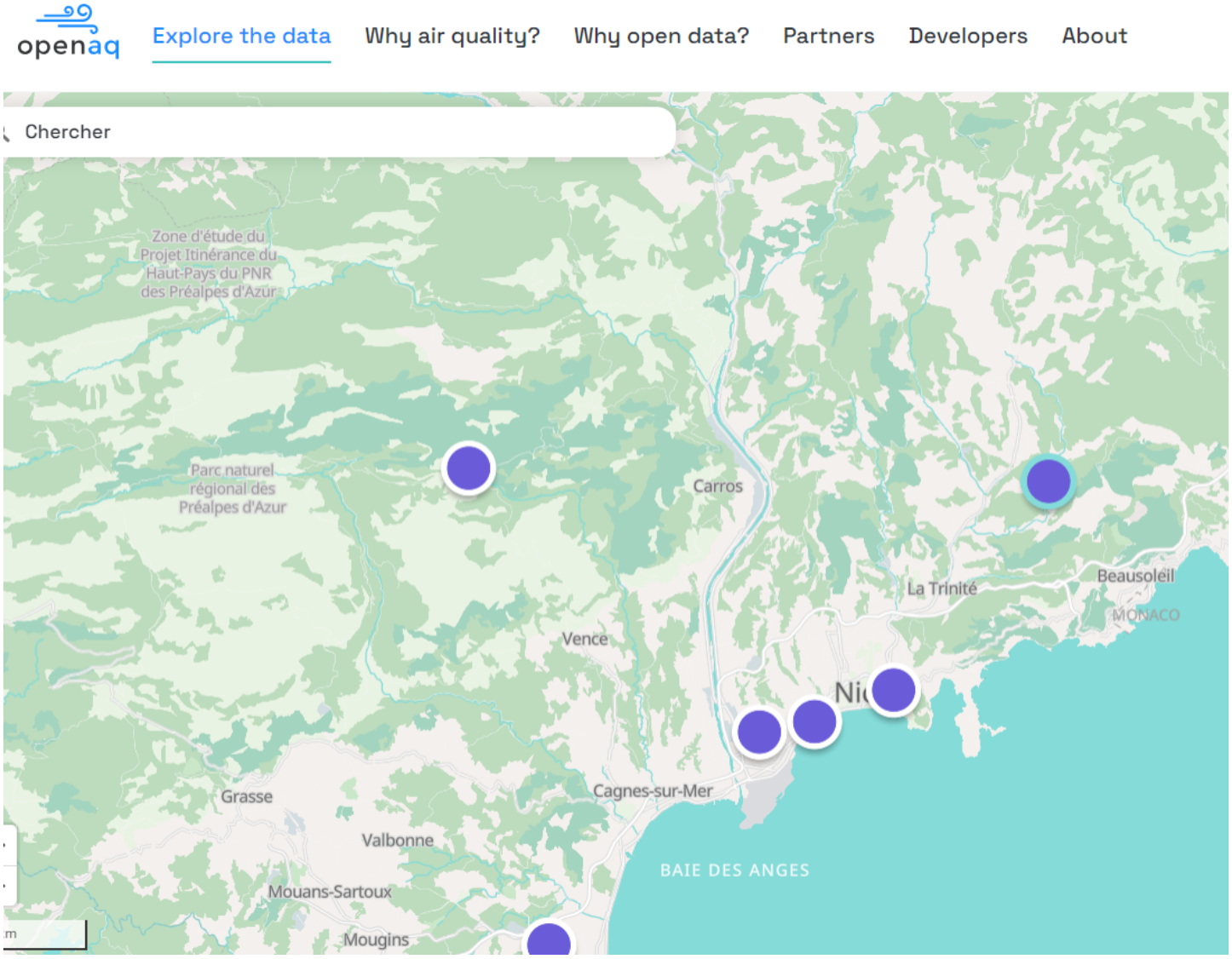


Voic la clé utilisé, vous pouvez créer votre compte pour obtenir votre propre clé .

1167c63aaaba2957263da32cb1064394f3ab00400117da052e19e49c3e49d457

Comme pour le site web openwheatermap, il est possible de visualiser les données sur le site ou bien de venir au travers de requêtes et d'une clé récupérer des informations sous forme de format json permettant le traitement de ces données (insertion dans un base de données, affichage, calculs, ...).

https://explore.openaq.org/



En cliquant sur une station on peut voir son numéro apparaitre dans l'URL, on pourra alors faire une demande de données sur cette station.

- Quel est le numéro de station pour le site du PEILLON et de COURSEGOULE ?

- Tester toutes les stations et notez les stations qui sont en fonctionnement

Exemple de requête pour la station 2162694 (par exemple).

https://api.openaq.org/v3/locations/2162694/latest

- Quel est le résultat affiché, Expliquer

L'envoie de la clé pour cette requête ne se fait pas directement dans l'URL, mais au travers du header qui fait partie de la requête envoyée au serveur mais qui n'est pas visible dans l'URL.

Dans sous linux la commande curl qui permet de générer une requête https mais en ajoutant le header (-H) qui intègre la clé de requête.

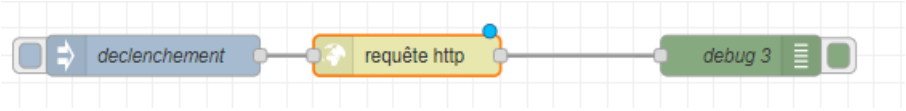
- Tester le résultat renvoyé par :

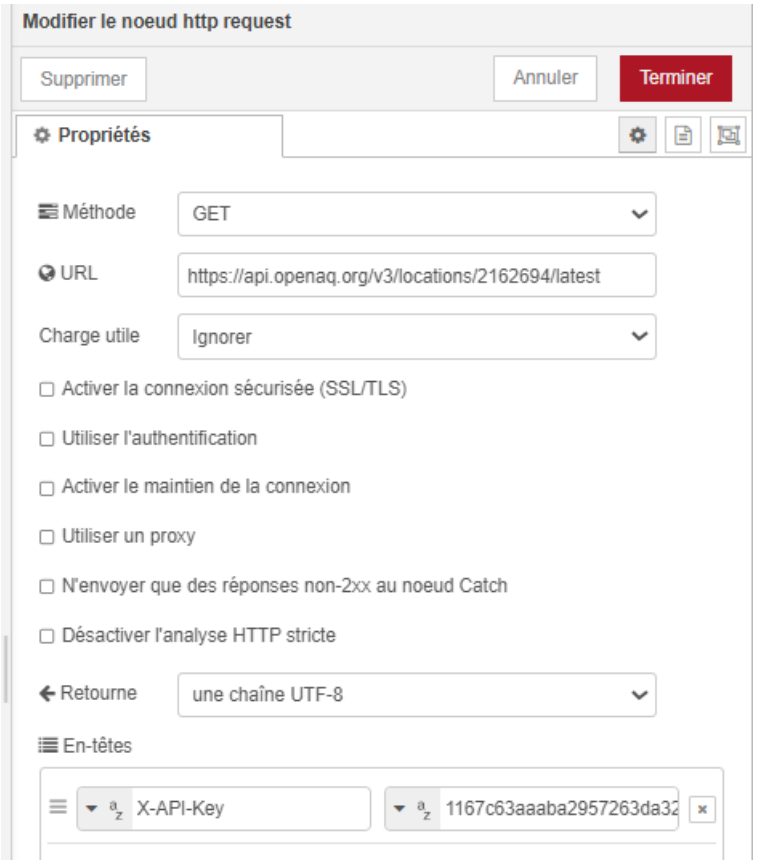
```
curl -i \
-H "X-API-Key: 92e53d49fbcd93bfce18a64ae99ca1fcab48f1107172ad299ba2fd4c227cf0d8" \
"https://api.openaq.org/v3/locations/2162694/latest"
```

- Tester le résultat renvoyé par cette deuxième requête (on demande l'affichage du type de capteur, chaque capteur a un id unique)

```
curl -i \
-H "X-API-Key: 92e53d49fbcd93bfce18a64ae99ca1fcab48f1107172ad299ba2fd4c227cf0d8" \
"https://api.openaq.org/v3/sensors/1290017"
```

Nous allons faire la même requête vers openaq.org dans nodered, en prenant soin d'ajouter la clé en paramètre d'en-tête.

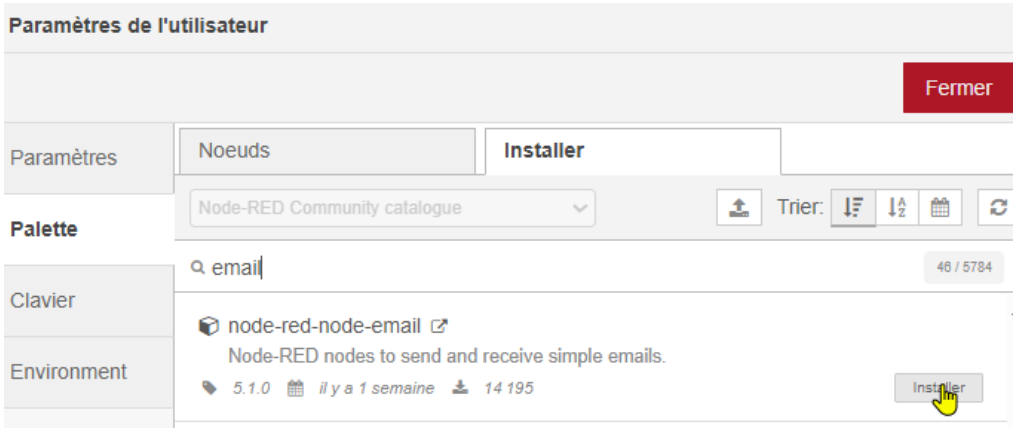


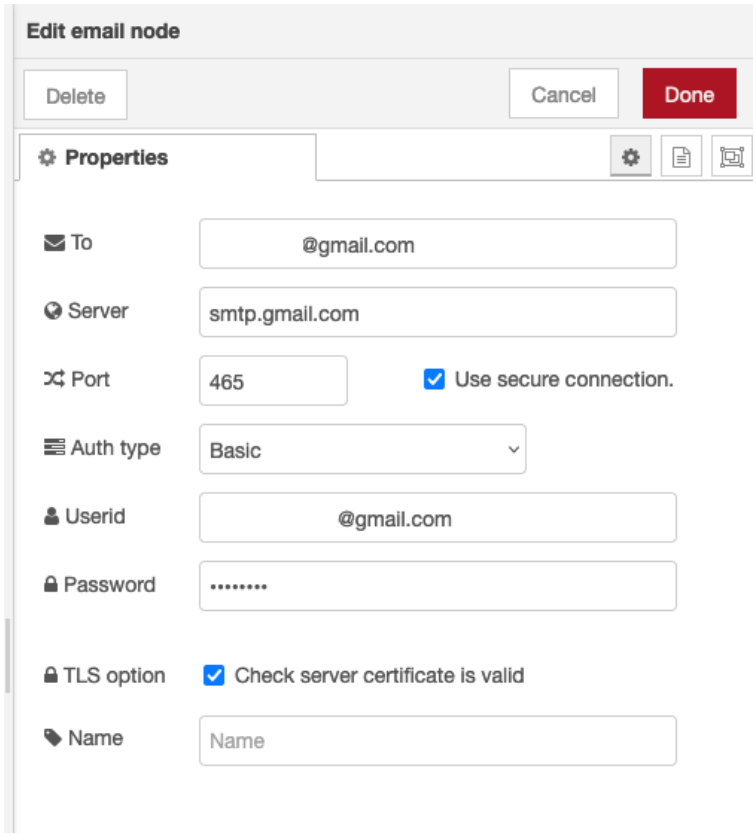


- Tester le programme.

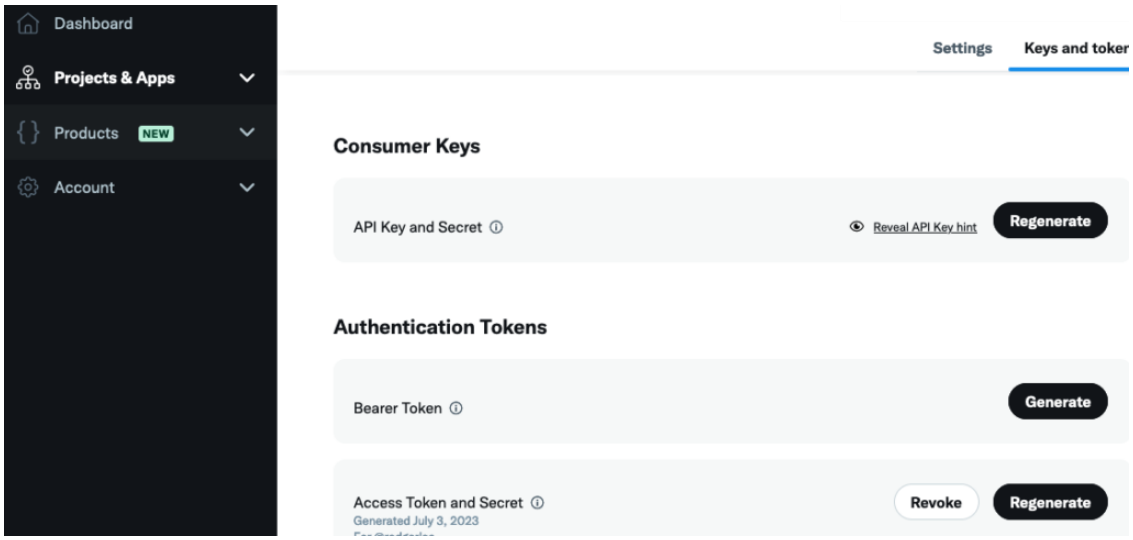
Travail à faire : vous devez utiliser un nœud dropdown dans lequel vous placerez les stations disponibles. En utilisant {{msg.payload}} vous enverrez le choix à httprequest pour aller chercher les informations de la station. Les informations pertinentes seront affichés dans un ui_text mais aussi dans un graphe et un donut.

En cas de dépassement d'un seuil d'alerte vous devrez vous envoyer un mail

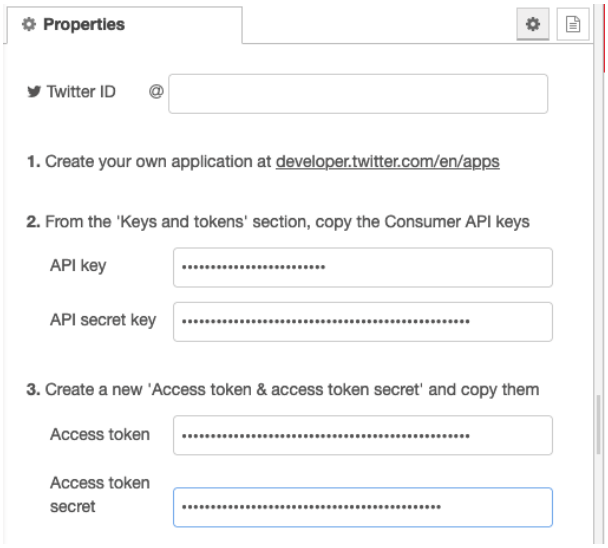




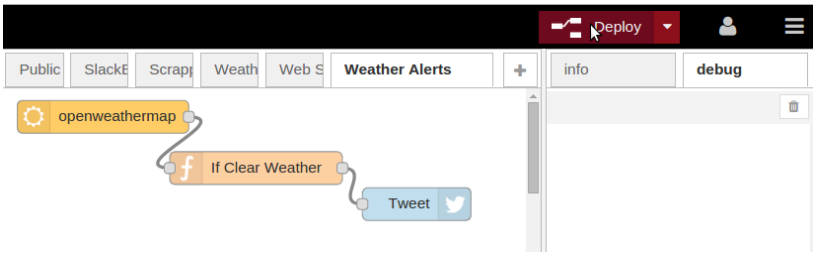
Ainsi qu'un tweet à condition d'avoir un token sur developer.twitter.com/en/apps



Vous pourrez alors configurer le tweet



Exemple avec openwheatermap



Travail à faire : tester l'envoi de mail et de tweet

3 communication avec le monde (mqtt)

3.1 introduction

MQTT = Message Queuing Telemetry Transport, c'est un protocole léger de communication conçu pour l'IoT, les capteurs et les objets connectés. Il est simple, rapide et peu gourmand en bande passante. C'est une autre solution simple et rapide pour faire communiquer des objets. Il fonctionne comme un **service de messagerie entre machines**. Les appareils ne parlent pas directement entre eux, mais via un **serveur central appelé Broker**.

Les rôles dans MQTT

1. **Broker (serveur) , dans notre cas mosquitto (installé sur la machine 217.154.16.99)**
   - ✓ Centre de communication
   - ✓ Reçoit les messages et les distribue aux bons clients
   - ✓ Exemples : Mosquitto, HiveMQ, EMQX
2. **Client (nodered,mqtt,...)**
   - ✓ Tout appareil qui envoie ou reçoit des messages
   - ✓ Peut être un capteur, un serveur, un ordinateur, ou Node-RED
   - ✓ Chaque client se **connecte au broker**

Concept clé : Topic

Les messages sont envoyés sur des **topics** (sujets)

- ✓ Exemple de topic : **maison/salon/temperature**
- ✓ **Topic hiérarchique** (avec des /) pour organiser les données
- ✓ Les clients s'abonnent aux topics qu'ils veulent lire et publient sur ceux qu'ils veulent envoyer

Pub/Sub : Publish / Subscribe

MQTT utilise le modèle **Publish/Subscribe** au lieu de client-serveur classique

- ✓ **Publisher** : publie un message sur un topic
- ✓ **Subscriber** : s'abonne à un topic pour recevoir les messages
- ✓ **Broker** : distribue le message à tous les abonnés

Exemple concret :

- ✓ Capteur température → publie sur maison/salon/temperature
- ✓ Dashboard Node-RED → s'abonne à maison/salon/temperature → affiche la température en temps réel

QoS (Quality of Service)

MQTT peut garantir la livraison des messages selon 3 niveaux :

QoS	Signification
0	Au maximum une fois (pas de confirmation)
1	Au moins une fois (confirmation mais duplication possible)
2	Exactement une fois (plus sûr mais plus lourd)

3.2 Premiers tests

Le serveur mosquitto est normalement déjà lancé sur la machine **217.154.16.99, il est présent sur le port 3883**

- ➢ Ouvrir putty et vérifier que le service fonctionne

```
ps -ef | grep mosquitto (coté user)
```

- ➢ Expliquer le résultat

Information ,Coté root : la commande pour avoir les informations du broket ss -tulpn | grep mosquitto

Vous allez tester mosquitto en local dans 2 fenêtres putty différentes et envoyer un message d'une fenêtre à l'autre, pour des raisons de sécurité vous devez ajouter un user (user_test) et un mot de passe (ici 1234) qui ont été configuré dans le broker

```
mosquitto_sub -h localhost -p 3883 -t "test" -u "user_test" -P "1234" -v
```

Et pour la deuxième fenêtre

```
mosquitto_pub -h localhost -p 3883 -t "test" -u "user_test" -P "1234" -m "Hello MQTT"
```

- ➢ Créer un nouveau message avec le topic /maison/temperature et envoyer la température sur la deuxième fenêtre. Placer la copie d'écran ci-dessous pour chaque fenêtre

Fenêtre 1

Fenêtre 2

Un programme a été écrit en python et il envoie des informations issues du CPU et mémoire ainsi qu'une rampe et une température simulée et ceci toutes les 5s.

MQTT_TOPIC_CPU = "vps/cpu"

MQTT_TOPIC_MEM = "vps/mem"

MQTT_TOPIC_RAMP = "vps/ramp"

MQTT_TOPIC_TEMP = "vps/temp"

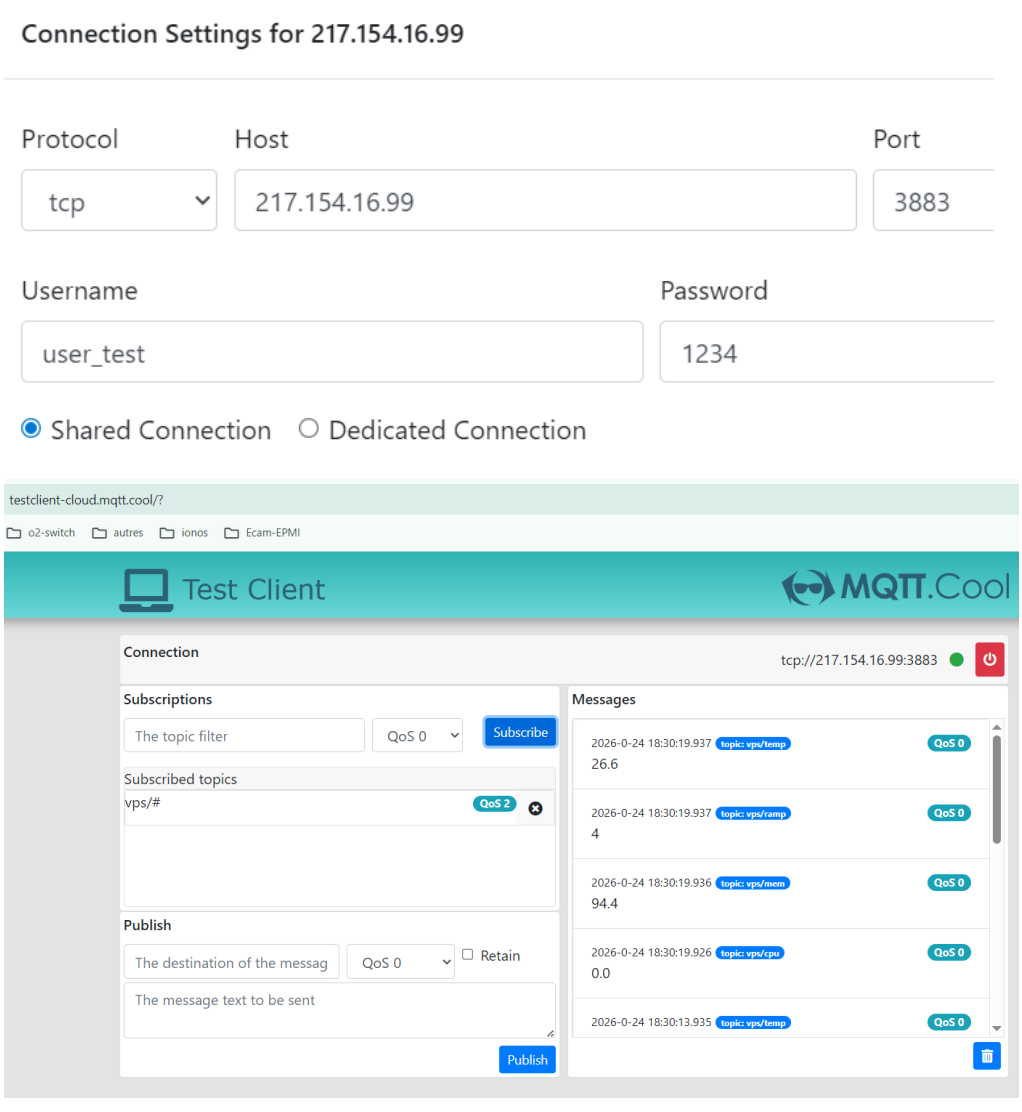
- Tester l'abonnement à un topic sur putty

```
mosquitto_sub -h localhost -p 3883 -t "vps/ramp" -u "user_test" -P "1234" -v
```

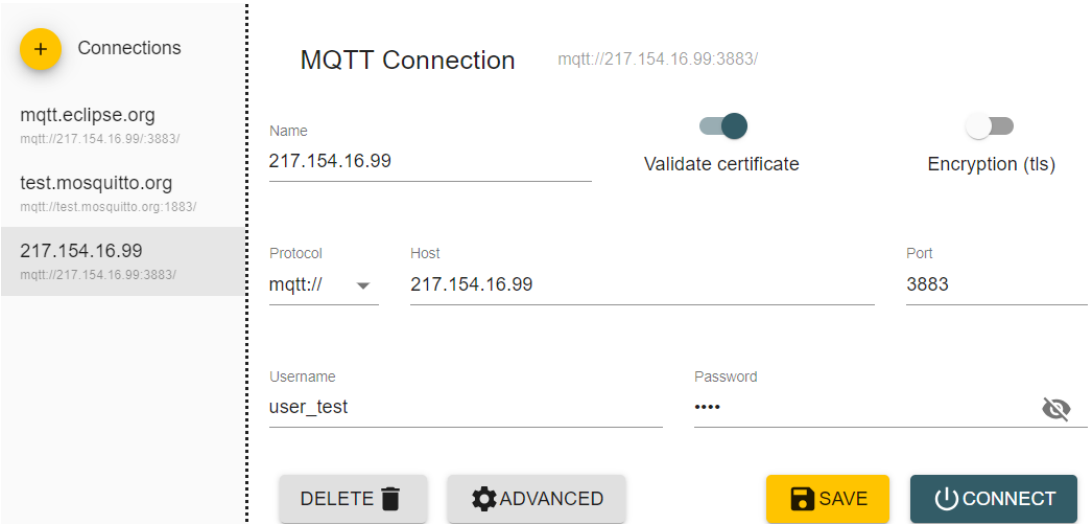
On peut visualiser tous les topics avec #, tester la commande pour voir les 4 topics apparaitre

3.3 Tests mqtt avec un client sur le cloud

Aller sur https://testclient-cloud.mqtt.cool/ et tester la connexion vers notre serveur mqtt 217.154.16.99, n'oubliez pas d'ajouter le username et le password permettant d'accéder au broker.

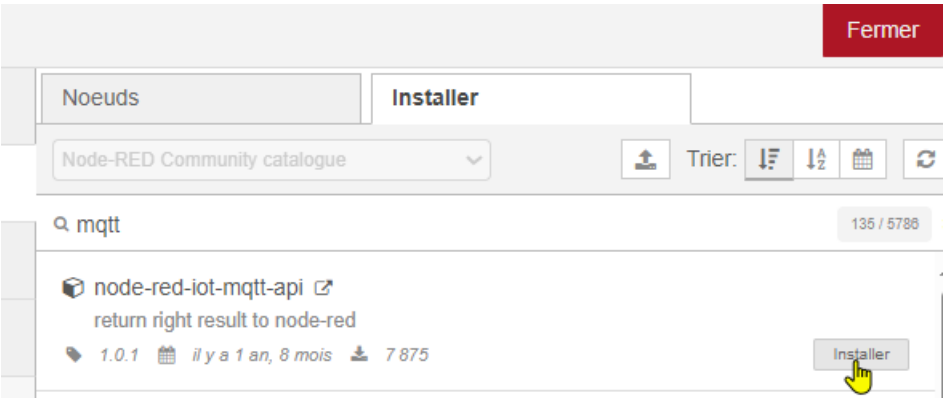


On aurait pu aussi utiliser MQTT explorer (client windows)

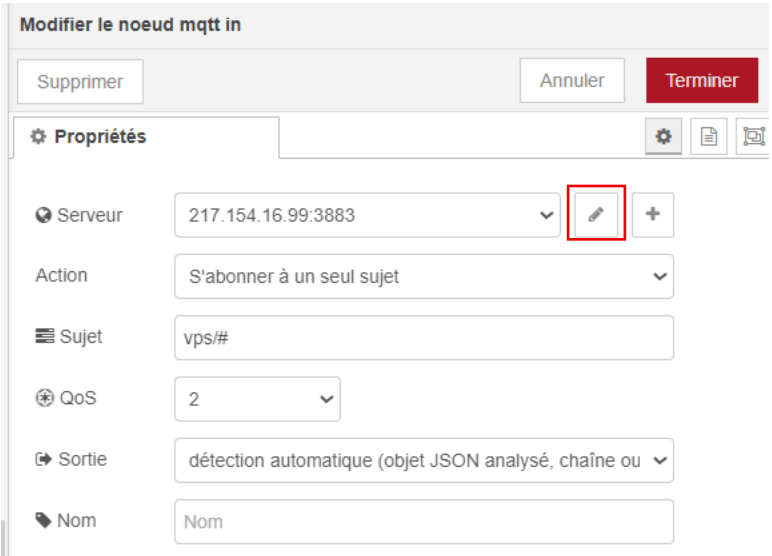


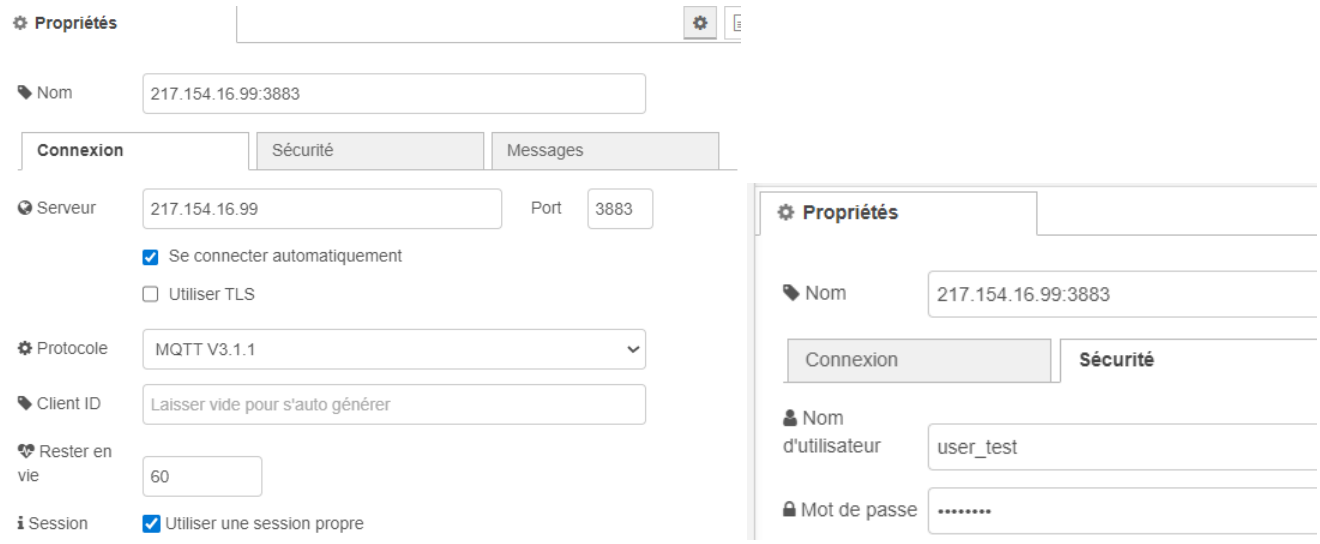
3.4 Utilisation de node-red

- Installer le nœud mqtt

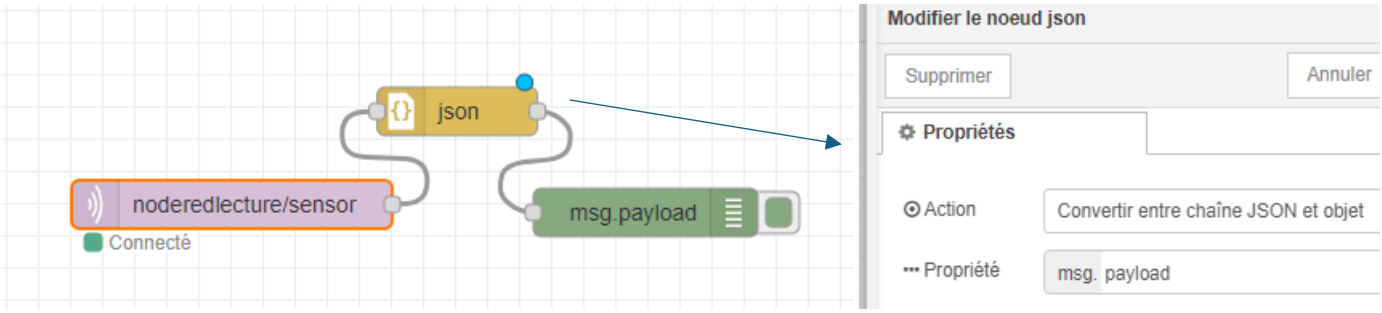


- Insérer le nœud mqtt in et le configurer





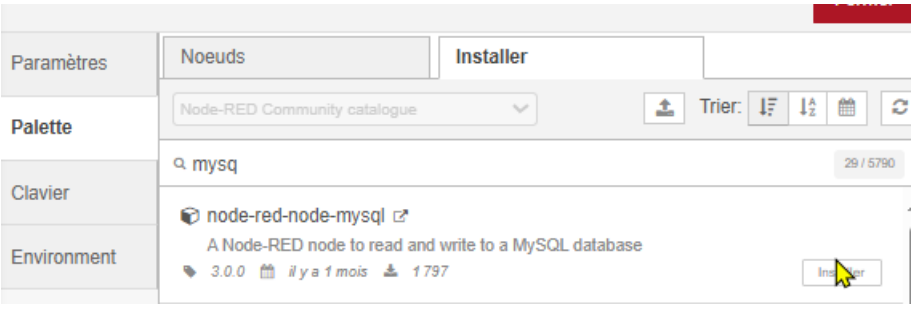
Il faudra de plus ajouter convertisseur string vers json pour remettre en forme la chaine de caractère dans un objet de type msg.



4 Travail avec mysql (mariadb)

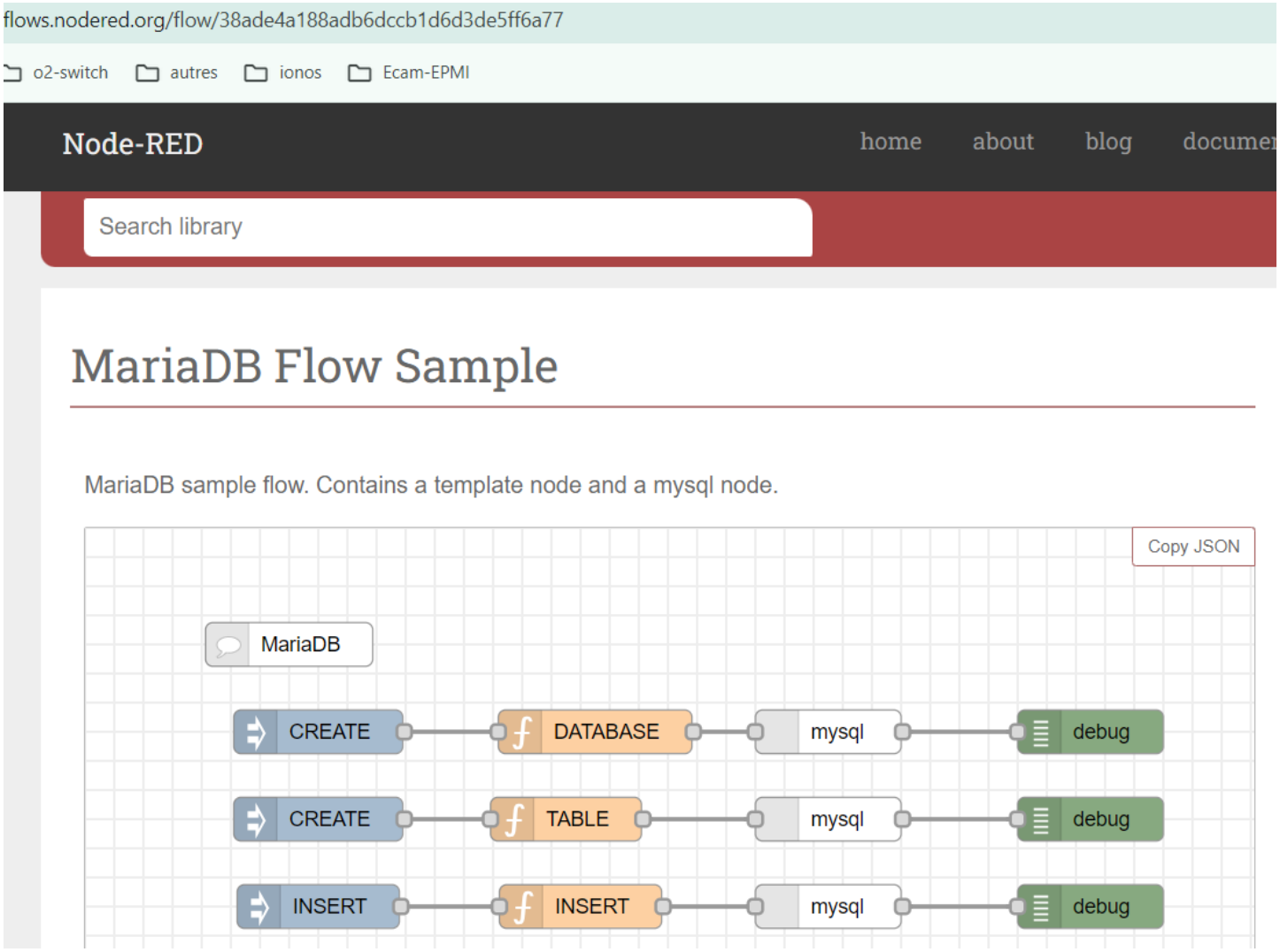
Nous allons dans cette dernière partie prendre en main le nœud mysql dans nodered.

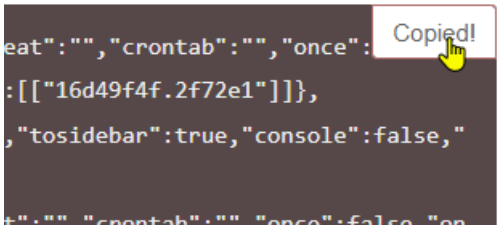
- Installer le nœud mysql



Vous avez de nombreux exemples sur le site de node-red, nous allons prendre un flow déjà créé et le modifier

- Allez sur https://flows.nodered.org/
- Cherche le flow mariadb





Pour l'accés à la base de données : dans le td1, chaque étudiant se connecte à la base de données avec son login et le mot de passe 1234.

Regarder la chaine https://youtu.be/KPhUmvSShSo et

Travail à faire : configurer la base, faite les premiers tests, vérifier avec putty la création de la base l'insertion de données et une requête sql

Travail à faire : utiliser un form pour lire des données et les placer dans une table, afficher cette table

Insérer les données venant de mqtt et les placer dans la base de données mariadb, afficher ces valeurs sur un graphe