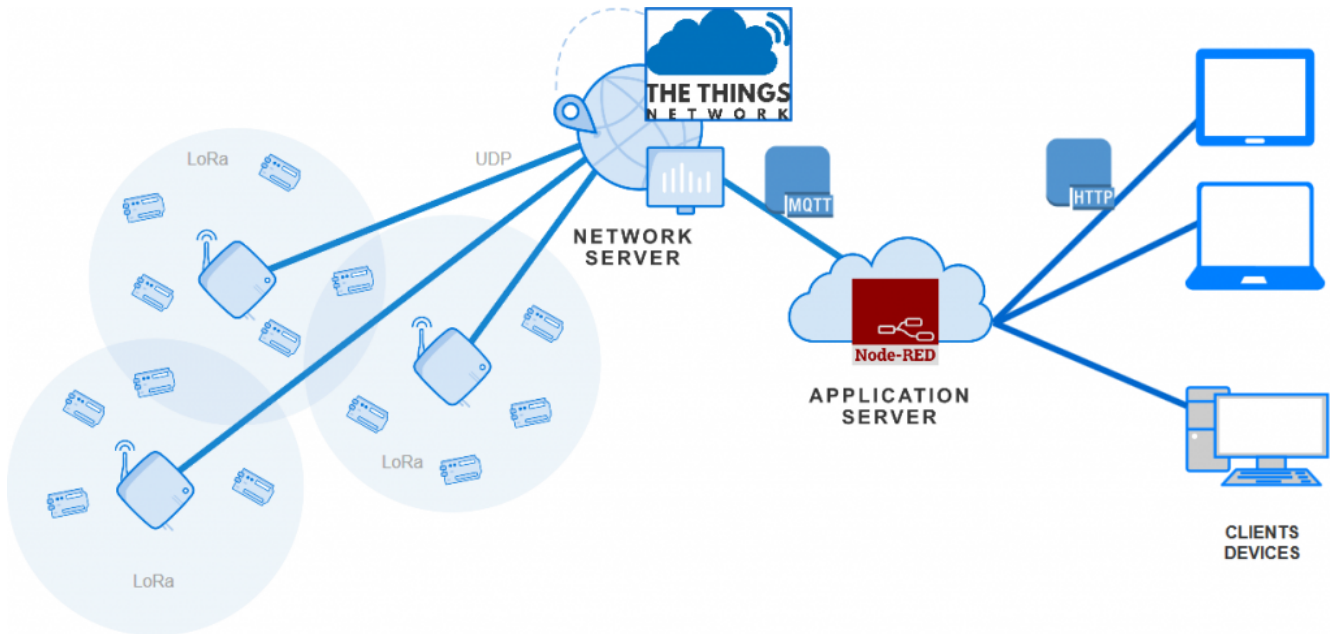


# Développement d'un objet connecté - Node-Red

## Sommaire

1. Introduction .....	2
2. Objectif de l'activité .....	3
3. Node-red .....	3
3.1. Installation .....	4
3.2. démarrage et arrêt .....	4
4. interface .....	5
4.1. Ajout d'une catégorie de nodes à la palette .....	6
4.2. Principe de la programmation .....	7
5. Réalisation de l'applications .....	10
5.1. souscription MQTT .....	11
5.2. Extraction des informations .....	14
5.3. Production du Dashboard .....	17
6. Enregistrement des données .....	18
6.1. Fichier csv .....	18
6.2. Base de données MySQL .....	20
6.3. Base de données InfluxDB .....	22
7. Allez plus loin... .....	24

# 1. Introduction



Les objets connectés ou l'**Internet des Objets** ( *IoT : Internet of Things* ) sont des termes très utilisés de nos jours même si bien des personnes ont encore du mal les définir.

Au sens large, un objet connecté est un objet communicant. Ils sont à la fois des émetteurs, des capteurs d'informations qui émettent, mais également qui peuvent interagir avec d'autres machines (M2M) comme des serveurs ou d'autres objets connectés. Les données générées par ces objets sont capitalisées dans des centres de données (*data centers*) et de nouveaux usages de cette donnée apparaissent autour de la santé, du sport, des produits financiers par exemple.

Les objets connectés connaissent une croissance exponentielle. Des estimations indiquent qu'ils seraient près de 15 milliards en circulation dans le monde actuel. Ils pourraient être plus de 50 milliards d'ici 2020.

Pour connecter les objets, les technologies actuelles peuvent être utilisées (*RFID, Wi-Fi, GSM, Bluetooth, Z-Wave, ZigBee, ...*) mais de nouveaux réseaux se développent au niveau mondial. Ils doivent permettre d'envoyer (mais aussi recevoir dans certains cas) **des très petits messages sur des longues portées** sans passer par des systèmes coûteux de réseaux mobile et **en consommant peu d'énergie**.

Au cours de ce TP, vous allez mettre en oeuvre **node-red**, pour acquérir les données d'un objet connecté au travers du réseau **The Things Network**, via le protocole **MQTT**, et réaliser un serveur d'application dont le **front-end** consistera en un site web appelé **dashboard**.

## 2. Objectif de l'activité

- Installer **node-red**.
- Connecter **node-red** aux serveurs **The Things Network** via **MQTT**.
- Réaliser un **dashboard** pour présenter les données à l'utilisateur final.
- Enregistrer les données dans une base de données.

## 3. Node-red



Figure 1. Node-red logo

**Node-red** est un outil puissant pour construire des applications de l'**Internet des Objets** en mettant l'accent sur la simplification de la programmation qui se fait grâce à des blocs de code prédéfinis, appelés **nodes** pour effectuer des tâches. Il utilise une approche de programmation visuelle qui permet aux développeurs de connecter les blocs de code ensemble.

Les noeuds connectés, généralement une combinaison de noeuds d'entrée, de noeuds de traitement et de noeuds de sortie, lorsqu'ils sont câblés ensemble, constituent un **flow**.

**Node-red** est construit sur **Node.js**, tirant pleinement parti de son modèle non bloquant piloté par les événements. Cela le rend idéal pour fonctionner en périphérie du réseau sur un serveur d'application qui peut être un matériel à faible coût tel que le **Raspberry Pi** ou un serveur dans le **cloud**.

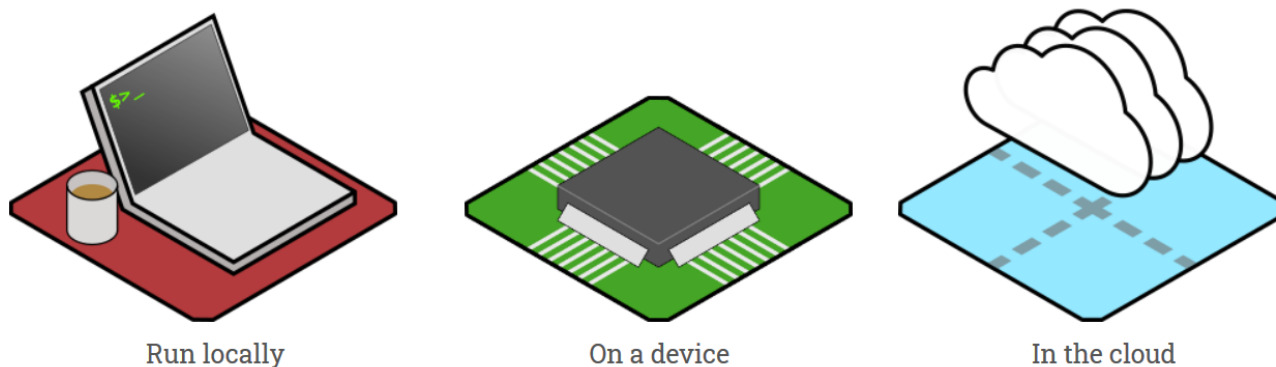


Figure 2. Node-red fonctionne sur tout type de systèmes

## 3.1. Installation

Consultez le guide de démarrage de node-red pour procéder à l'installation qui correspond à votre cas.

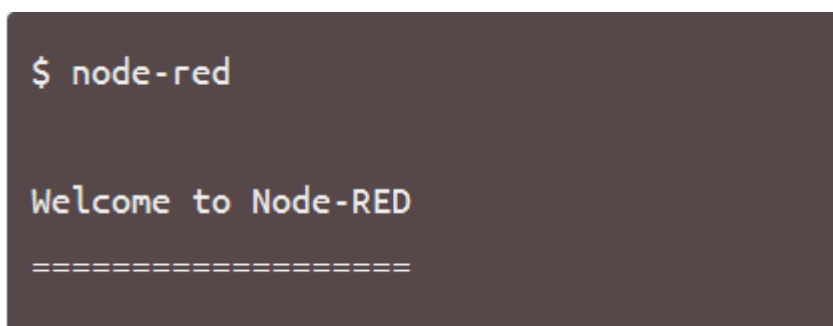
<https://nodered.org/docs/getting-started/>

Nous utiliserons ici **node-red** localement sur notre ordinateur. L'installation consiste donc à installer préalablement **node.js** :

- [Installation de `node.js`](#)
- [Installation de \*\*node-red\*\*](#)

## 3.2. démarrage et arrêt

- Pour démarrer **node-red**, utilisez la commande **node-red** dans un terminal.

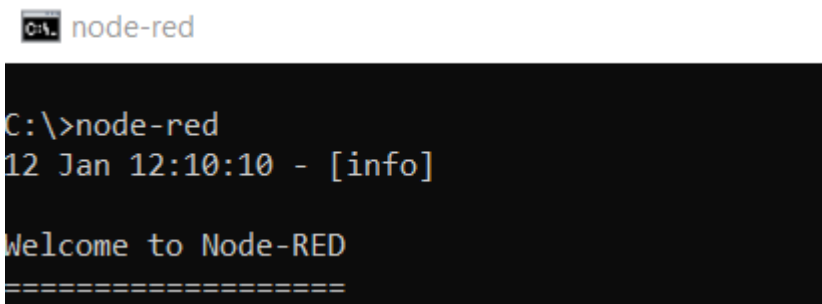


```
$ node-red

Welcome to Node-RED

=====
```

Figure 3. Sous linux



```
node-red

C:\>node-red
12 Jan 12:10:10 - [info]

Welcome to Node-RED

=====
```

Figure 4. Sous Windows

- Pour arrêter **node-red**, utilisez la combinaison de touche `kbd:[Ctrl] + kbd:[C]`

## 4. interface

On accède à l'interface de développement par un navigateur en utilisant l'URL : <http://127.0.0.1:1880> comme l'indique **node-red** dans le terminal :

```
C:\> node-red

Welcome to Node-RED
=====
12 Jan 12:10:10 - [info] Node-RED version: v1.0.3
12 Jan 12:10:10 - [info] Node.js version: v12.14.1
12 Jan 12:10:10 - [info] Windows_NT 10.0.18363 x64 LE
12 Jan 12:10:11 - [info] Loading palette nodes
12 Jan 12:10:12 - [info] Dashboard version 2.19.3 started at /ui
12 Jan 12:10:12 - [info] Settings file : \Users\marcs\.node-red\settings.js
12 Jan 12:10:12 - [info] Context store : 'default' [module=memory]
12 Jan 12:10:12 - [info] User directory : \Users\marcs\.node-red
12 Jan 12:10:12 - [warn] Projects disabled : editorTheme.projects.enabled=false
12 Jan 12:10:12 - [info] Flows file : \Users\marcs\.node-red\flows_DESKTOP-G07TA16.json
12 Jan 12:10:12 - [info] Server now running at http://127.0.0.1:1880/
12 Jan 12:10:12 - [warn]
```

Figure 5. Execution de node-red dans le terminal

On y accède également depuis n'importe quel poste du réseau pour peu que le pare-feu soit correctement configuré.

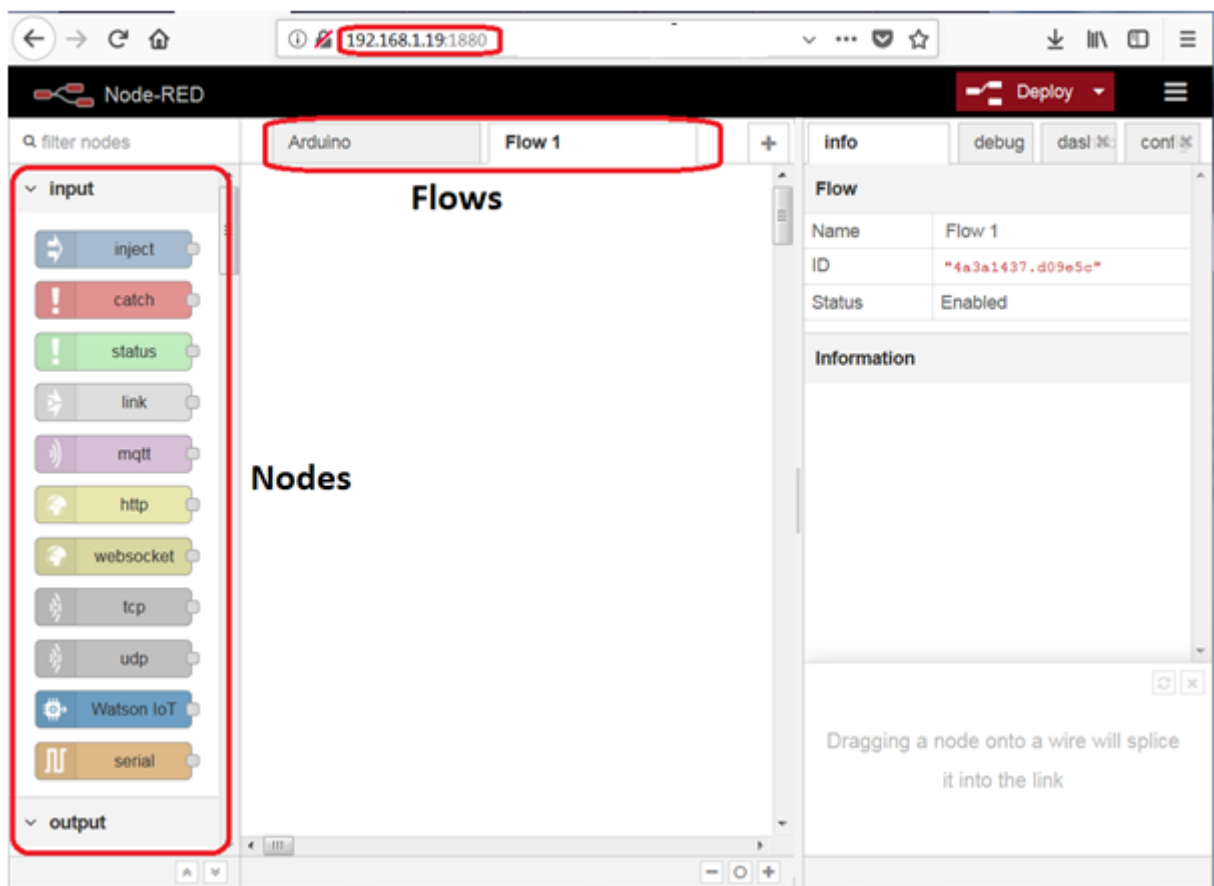


Figure 6. IHM de node-red

## 4.1. Ajout d'une catégorie de nodes à la palette

La palette de **nodes** d'origine est déjà bien fournie mais il manque au moins une catégorie essentielle qui va permettre de réaliser l'interface web. Elle se nomme **Dashboard**.

Dans le menu, cliquez sur **Manage Palette** puis recherchez la catégorie de nodes à installer. Une nouvelle catégorie est ensuite disponible :

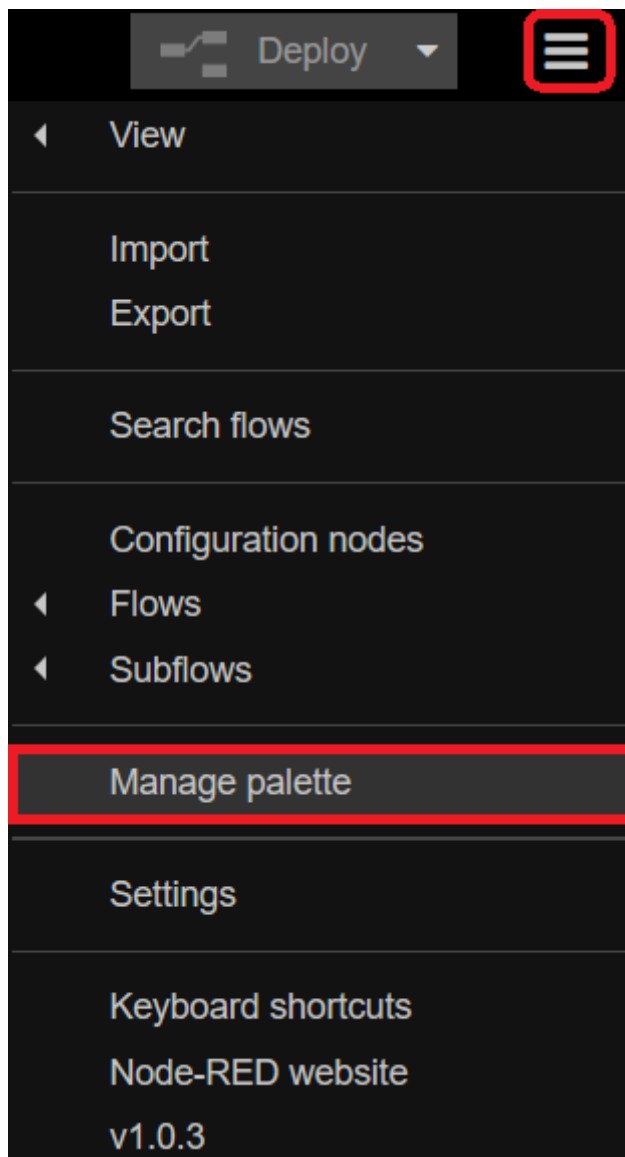


Figure 7. Accéder au gestionnaire de palette

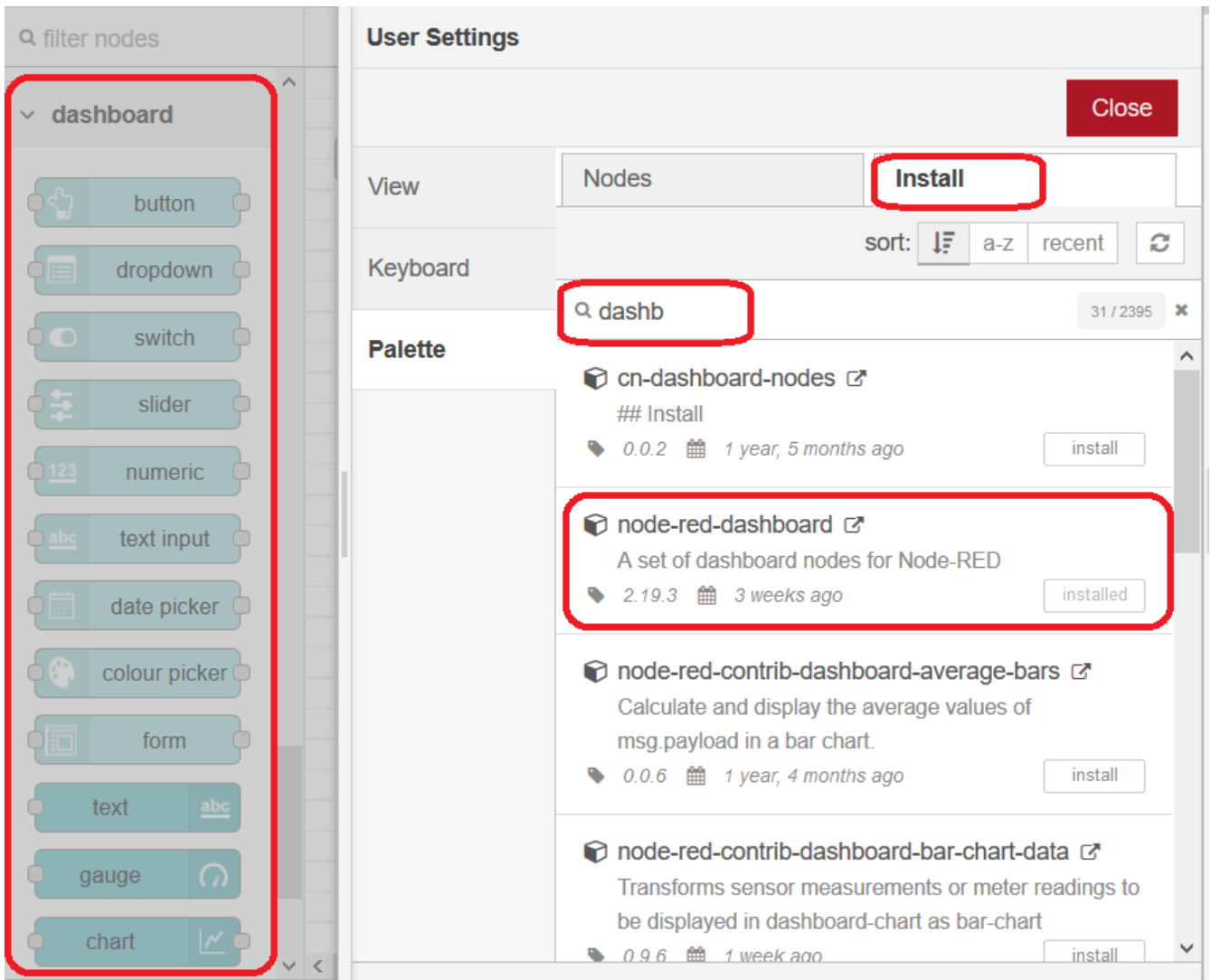


Figure 8. Ajout de la catégorie Dashboard à la Palette

## 4.2. Principe de la programmation

- La programmation consiste à glisser-déplacer les **nodes** sur le **flow** et à les relier entre eux. Les données échangées sont des objets **JSON** et doivent posséder la propriété **payload** (charge utile).
- Le **payload** peut être un nombre, une chaîne de caractères ou un objet **JSON**.
- Certains nodes peuvent avoir plusieurs sorties. Dans ce cas, le node retourne un tableau d'objets **JSON** dont le premier élément sort par la première sortie, le deuxième par la seconde, ...

Commençons par réaliser un **flow** très simple composé de :

- un **Slider**
- une **Gauge**.

Pour modifier la configuration par défaut des **nodes**, il suffit de double-cliquer dessus.

Pour que ces composants soient bien disposés sur la page web, il faut les mettre dans un **Tab**, puis dans un **Groupe** :

- Dans la boîte de sélection du **groupe**, cliquez sur **Add new ui\_group**,

- Créer un **groupe** (ici : "**Le groupe**") et faite de même pour le **Tab** (ici : "**Test**").
- Cliquez ensuite sur le bouton **Done**, puis déployer le **flow** en cliquant sur le bouton **Deploy**.
- On accède au site web généré par l'url : <http://127.0.0.1:1880/ui>

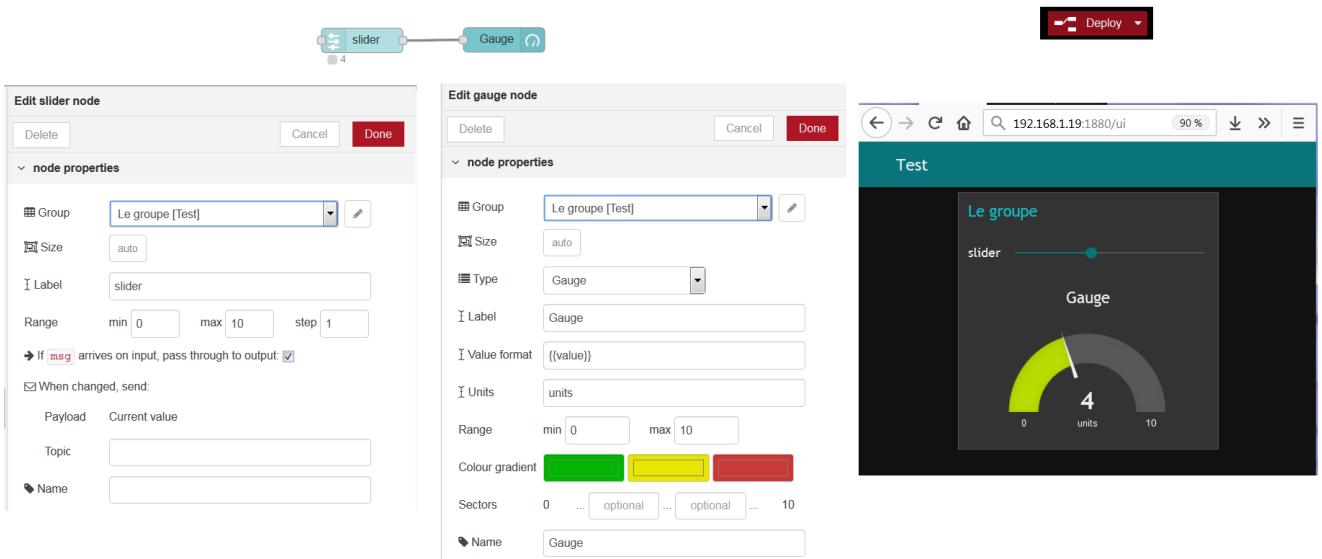


Figure 9. Jauge pilotée par un slider

- Actionnez le **Slider** et constatez l'action sur la jauge (**Gauge**).
- Modifiez le **flow** en ajoutant un node **Numeric**, déployez et constatez le résultat :



Figure 10. Ajout d'un node Numeric



- Pour conserver un historique des valeurs affichées, on peut utiliser un **node Chart** :

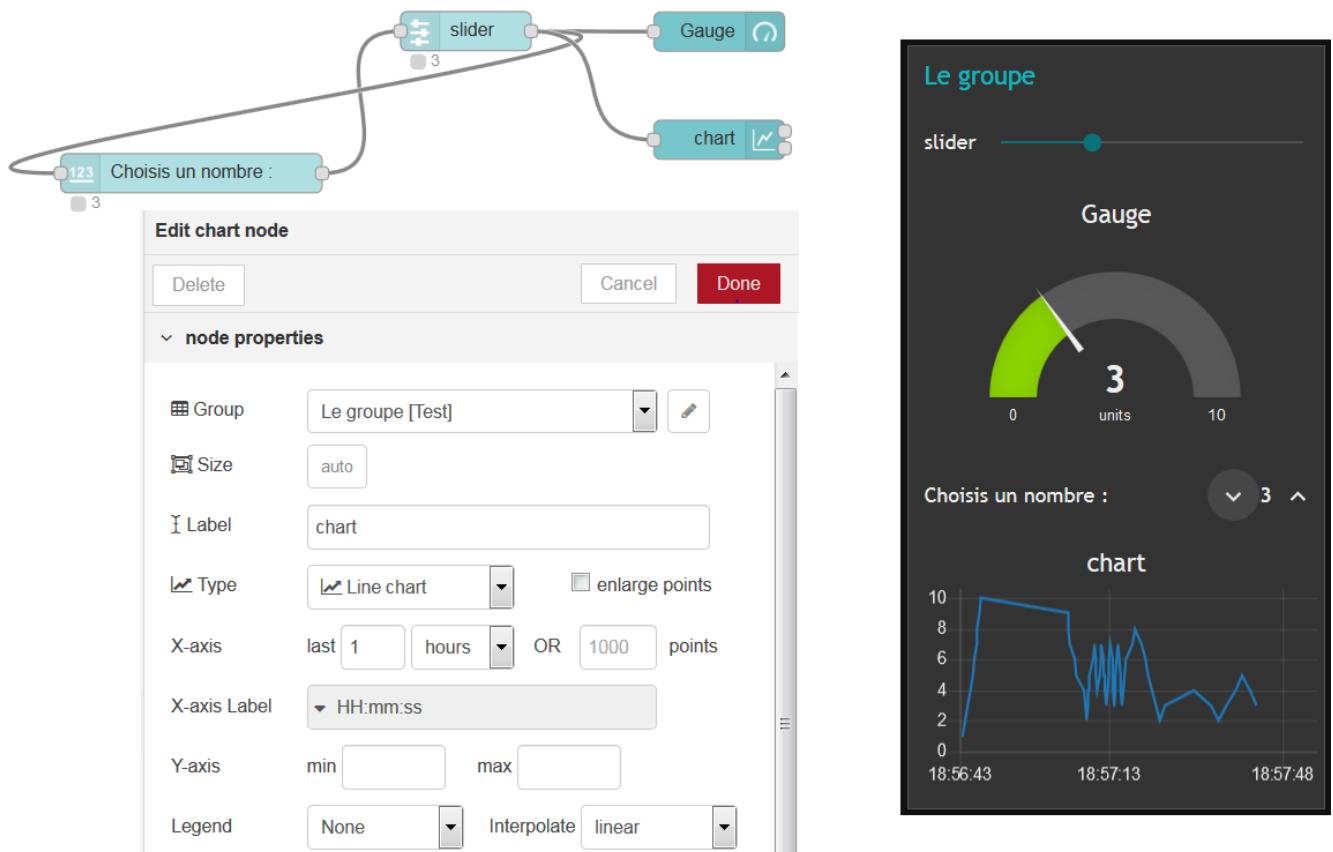


Figure 11. Ajout d'un node Chart pour obtenir un graph de l'historique



Si la propriété **Y-axis** est laissée vide, l'échelle est automatique. L'affichage de la courbe est réactualisé à chaque nouvelle valeur.

Nous allons maintenant rajouter une information textuelle en bas du **groupe**. Cette information contiendra par exemple la valeur du **slider**, la **date et l'heure** de modification.

- Ajoutez un **node Text** qui permet d'ajouter un **label** et le contenu d'un message (**value**).

Si on se contente de connecter la sortie du node slider à l'entrée du node Text, on pourra écrire :



```
Label="Valeur = "
Value Format="{{msg.payload}}"
```

**Ce n'est pas suffisant dans notre cas ! Nous devons ajouter une fonction pour fabriquer notre message.**

- Intercalez un **node Function** pour insérer la date dans le message à afficher.
- Modifiez la propriété **label** du **node Text** : "**Valeur le**"



Le **node Function** permet d'effectuer n'importe quel traitement sur le flux entrant pour fabriquer un ou plusieurs flux sortants. Le langage utilisé est du **JavaScript** :

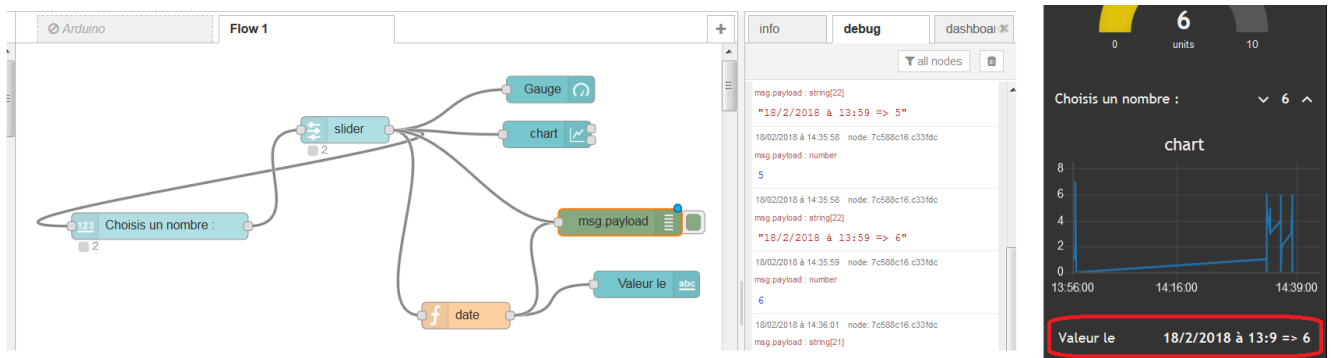


Figure 12. Ajout du node Text et Debug

Code de la fonction `date()`

```
var now      = new Date();
var year    = now.getFullYear();
var month   = now.getMonth()+1;
var day     = now.getDate();
var hour    = now.getHours();
var minute  = now.getMinutes();
var second  = now.getSeconds();
msg.payload =  day+"/"+
               month+"/"+
               year+" à "+
               hour+": "+
               second+" => "+
               msg.payload;

return msg;
```

- Déployez observez le résultat.



**A noter** : Les messages transmis dans les flux d'entrée et de sortie des nodes sont des objets. C'est l'attribut **payload**, soit "**charge utile**" en français qui contient l'information à transmettre. Pour observer cette charge utile de message vous pouvez utiliser un **node Debug**.

## 5. Réalisation de l'applications

Nous avons un objet connecté qui transmet à interval régulier la température et l'humidité relative qu'il mesure aux serveurs de **The Things Network**. Ces derniers diffusent ces données via le protocole **MQTT** aux applications qui y souscrivent. Nous allons donc réaliser un **flow** qui connecte ces serveurs afin d'obtenir les mesures et les afficher dans un **dashboard**.

## 5.1. souscription MQTT

- Saisissez le **flow** suivant :
- Configurez le **node mqtt in** :

La consultation de la [documentation de l'API de The Things Network](#) permet de trouver les éléments suivants nécessaires à la configuration du node :



- Host: `<Region>.thethings.network`, where `<Region>` is last part of the handler you registered your application to, e.g. `eu`.
- Port: `1883`
- Username: `Application ID`
- Password: `Application Access Key`
- Topic: `<AppID>/devices/<DevID>/up`

- Double-cliquez sur le **node mqtt in** pour faire apparaître la boîte de dialogue de configuration.

**Edit mqtt in node**

Delete Cancel Done

node properties

Cliquer ici

Server undefined:1883

Topic Topic

QoS 2

Name Name

Figure 13. Boîte de dialogue de configuration du **node mqtt in**

- Configurez les paramètres du serveur conformément à la documentation de l'API de **TTN** puis cliquer sur **Update**

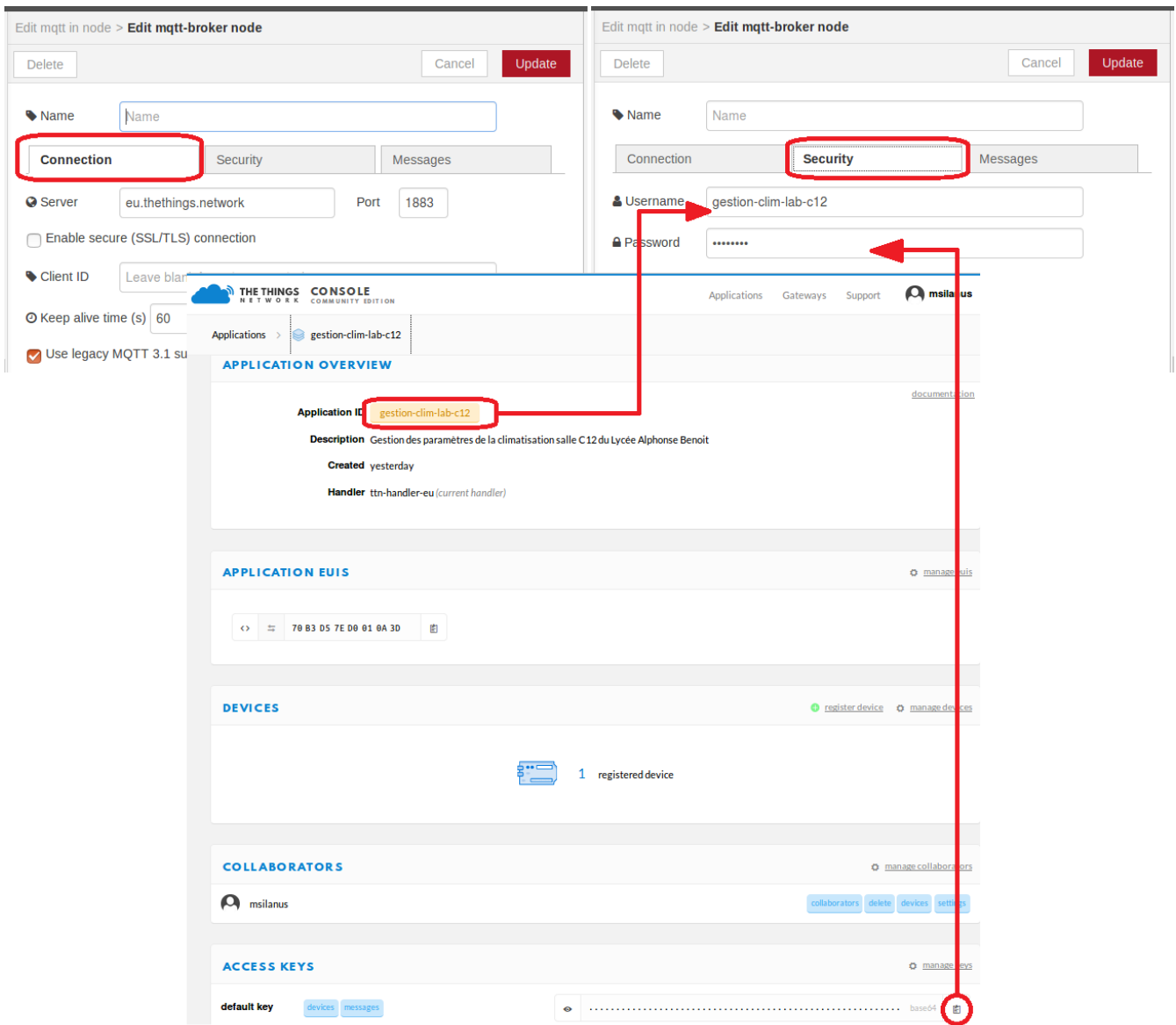


Figure 14. Configuration du node mqtt in pour la connexion aux serveurs TTN

- Configurez le Topic conformément à la documentation de l'API de TTN et cliquer sur Done

**Edit mqtt in node**

Delete Cancel Done

node properties

Server eu.thethings.network:1883

Topic gestion-clim-lab-c12/devices/temp-hum-sensor-1/up

QoS 2

Name mqtt from TTN

---

THE THINGS NETWORK CONSOLE COMMUNITY EDITION

Applications > gestion-clim-lab-c12 > Devices > temp-hum-sensor-1

### DEVICE OVERVIEW

Application ID **gestion-clim-lab-c12**

Device ID **temp-hum-sensor-1**

Activation Method OTAA

Device EUI <> 00 04 A3 0B 00 1E 7D 95

Application EUI <> 70 B3 D5 7E D0 01 0A 3D

Figure 15. Configuration du node mqtt in pour souscrire à un topic

Il est normalement possible de sélectionner les informations souscrites via le topic. Par exemple, pour ne recevoir que la température, on peut souscrire au topic :

`<AppID>/devices/<DevID>/up/temperature_1` (selon la configuration du `payload formats` pour l'application considérée dans la console TTN)

Cependant, TTN met en garde quand l'utilisation des filtres vers les champs de données :



**Warning: not every cluster publishes uplink fields to individual topics. See [status.thethings.network](https://status.thethings.network) for details.**

**Uplink Fields on MQTT Disabled**

**Identified** - For performance reasons we unfortunately had to disable publishing Uplink Fields to MQTT in our EU region. This means that the values from payload decodes are no longer published to individual `[AppID]/devices/[DevID]/up/[Field]` topics. Until this is resolved, you can subscribe to `[AppID]/devices/[DevID]/up` and read the `payload_fields` field from the uplinks you receive there.

Nov 18, 11:00 CET

Figure 16. TTN status page

- Déployez et observez les traces `debug`.

debug

all nodes

```
{"app_id": "formation-lora-sts-sn-2020", "dev_id": "temp-hum-sensor-1", "hardware_serial": "0004A30B001E657C", {"relative_humidity_2": 68, "temperature_1": 20.5, "time": "2020-01-12T15:34:10.898564455Z", [{"gtw_id": "lab-ttn", "gtw_trusted": true, "timestamp": 655454, "snr": 9.25, "rf_chain": 0, "latitude": 43.7614, "longitude": 4.8358}], "payload_fields": [{"field": "temperature_1", "value": 20.5}, {"field": "relative_humidity_2", "value": 68}], "timestamp": "2020-01-12T15:34:10.898564455Z", "gateway_id": "lab-ttn", "gateway_trusted": true, "snr": 9.25, "rf_chain": 0, "latitude": 43.7614, "longitude": 4.8358}
```

12/01/2020 à 16:35:14 node: fbd8b724.734ef8  
formation-lora-sts-sn-2020/devices/temp-hum-sensor-1/up : msg.payload : string[610]

Figure 17. Debug de la réception mqtt

## 5.2. Extraction des informations



Le `payload` du message reçu est une `string`. Pas très pratique pour accéder aux propriétés et aux valeurs qu'elle contient. Nous allons la transformer en objet `JSON` en utilisant la classe javascript `JSON` et sa méthode `parse()`.

- Ajouter un `node Function` avec le code :

```
msg.payload = JSON.parse(msg.payload);
return msg;
```



Remarque : Le `node json` fait également très bien l'affaire ! `.Node convert-json image::images/convert-json.png[align="center"]`

- Déployez et observez les traces `debug`.

```
12/01/2020 à 16:47:40 node: fbd8b724.734ef8
formation-lora-sts-sn-2020/devices/temp-hum-
sensor-1/up : msg.payload : Object
  ▶ { app_id: "formation-lora-sts-
sn-2020", dev_id: "temp-hum-
sensor-1", hardware_serial:
"0004A30B001E657C", port: 1, counter:
51 ... }
```

Figure 18. Debug après conversion en JSON

- Créez les fonctions nécessaires pour accéder à :
  - la température : `getTemp()`
  - l'humidité relative : `getHum()`
  - l'horodatage de la mesure : `getTime()`

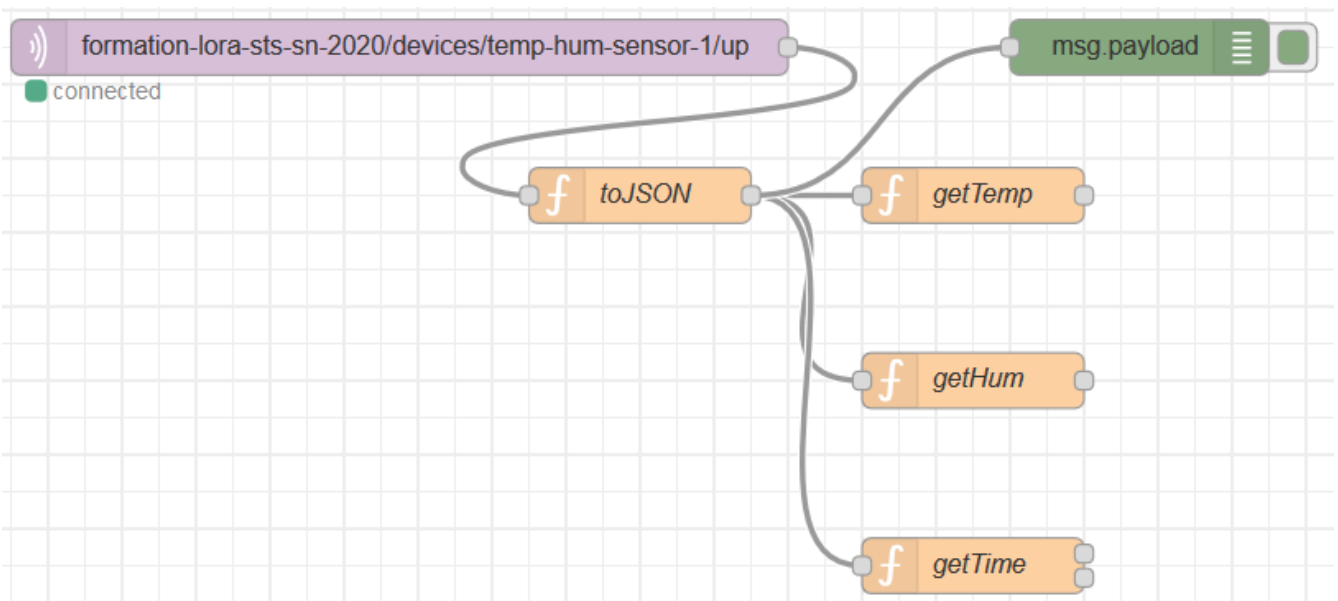


Figure 19. Fonction de découpages des informations

### Code la fonction `getTemp()`

```
var temp = msg.payload.payload_fields.temperature_1;
msg.payload = temp
return msg;
```

### Code la fonction `getHum()`

```
var hum = msg.payload.payload_fields.relative_humidity_2;
msg.payload = hum
return msg;
```

### Code de la fonction `getTime()`

```
var time = msg.payload.metadata.time;
var date = time.split("T")[0];
var heure = time.split("T")[1].split(".")[0];
var msg1 = {};
var msg2 = {};
var out = {"date" : date, "heure" : heure};
msg1.payload = out;
msg2.payload = "Le " + out.date + " à " + out.heure;

return [msg1, msg2];
```



La fonction `getTime()` possède deux sorties. Ne pas oublier de configurer le `node` en conséquence. La sortie 1 fournit un objet qui contient la date et l'heure. La sortie 2 fournit une chaîne de caractères.



## 5.3. Production du Dashboard

Pour le dashboard, nous allons utiliser deux jauges (nodes gauges) et deux nodes chart pour tracer l'évolution des grandeurs mesurées. Nous utiliserons également un node text pour afficher la date et l'heure et de la mesure.

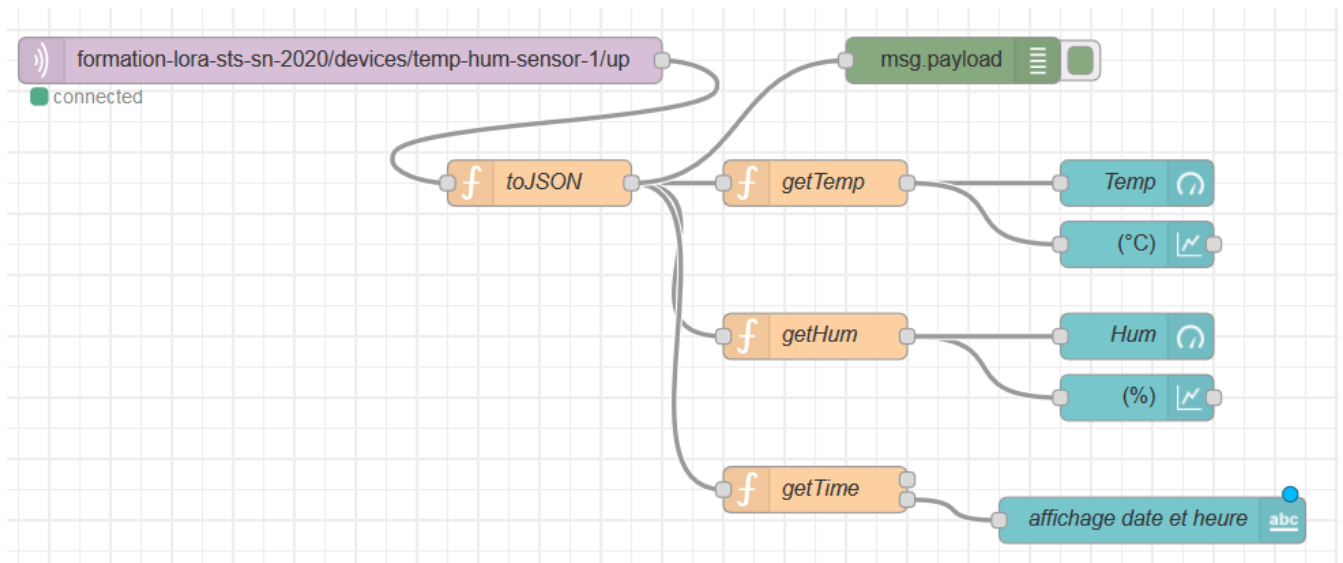


Figure 20. Flow complet de l'applications

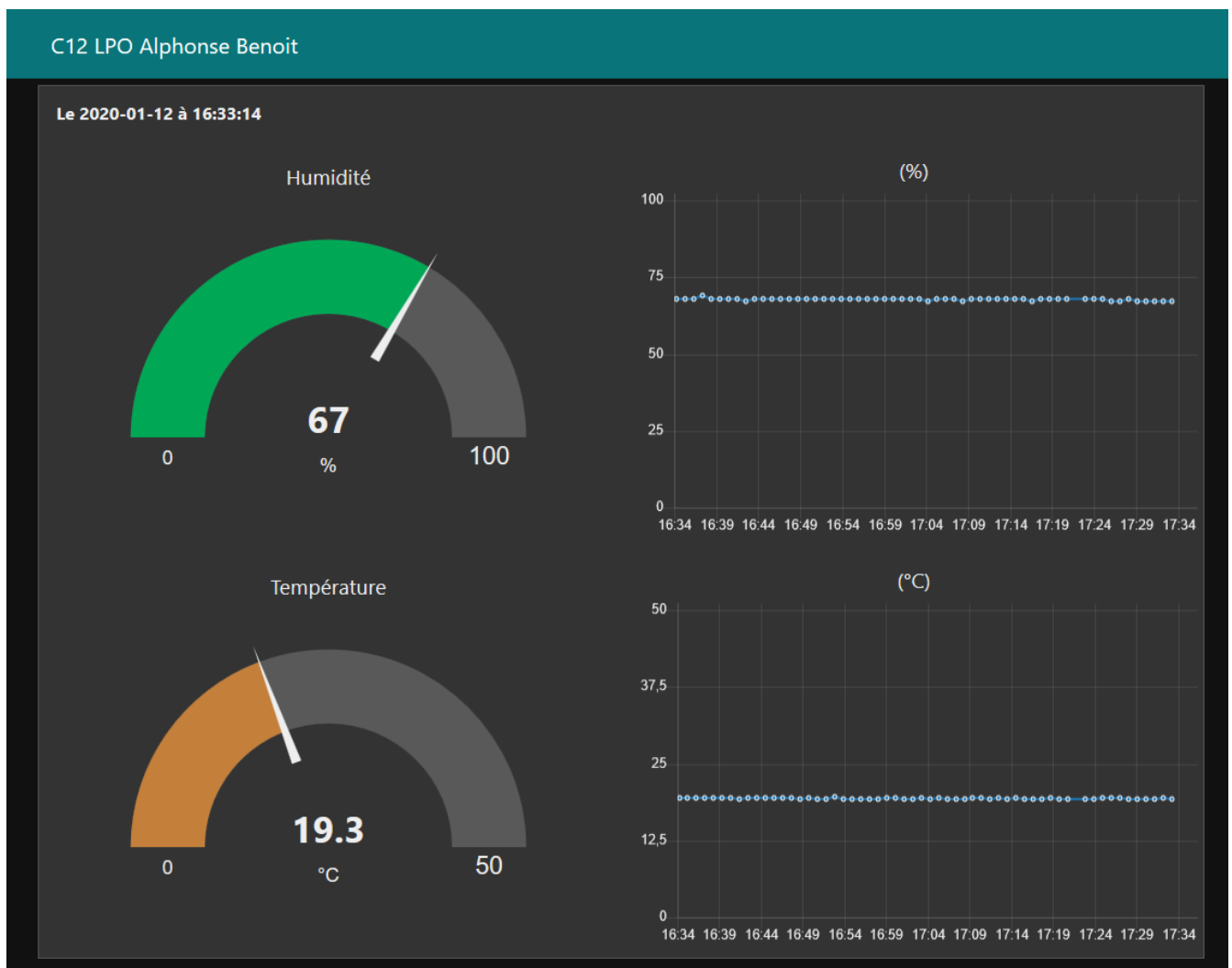


Figure 21. Dashboard complet

# 6. Enregistrement des données

Il est possible d'enregistrer les données dans un **fichier texte** ou dans une **base de données**. Il suffit d'utiliser le node approprié.

## 6.1. Fichier csv

Le **node csv** permet de formater une **string** en **csv** à partir d'un objet **JSON** :

- Glissez le **node csv** sur le **flow** et connectez le à la sortie 1 de la fonction **getTime()**.

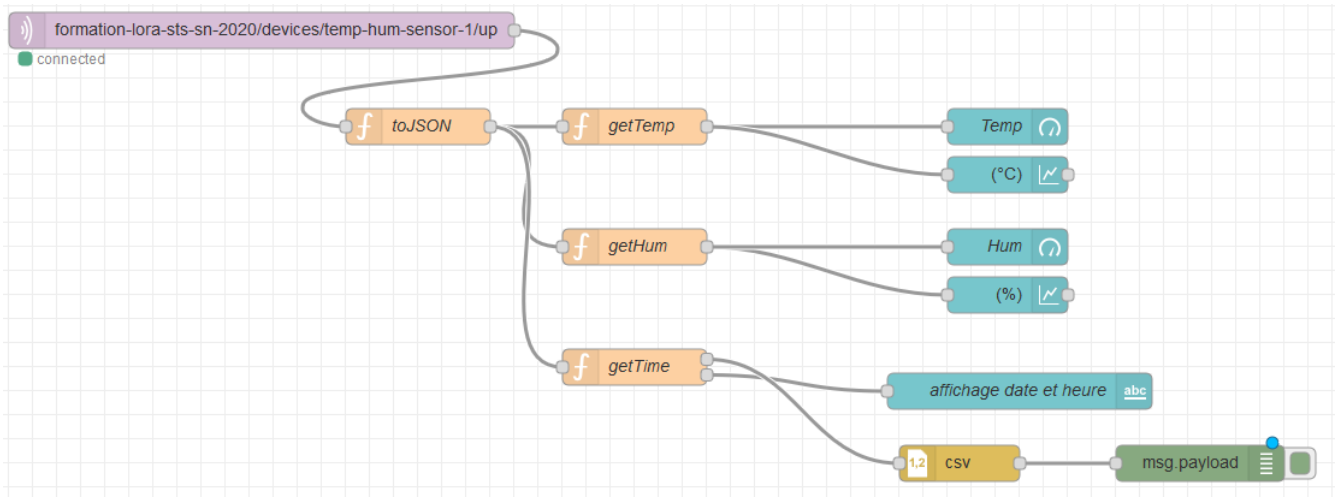


Figure 22. Préparation des données en csv

- Configurez le **node csv** comme suit :

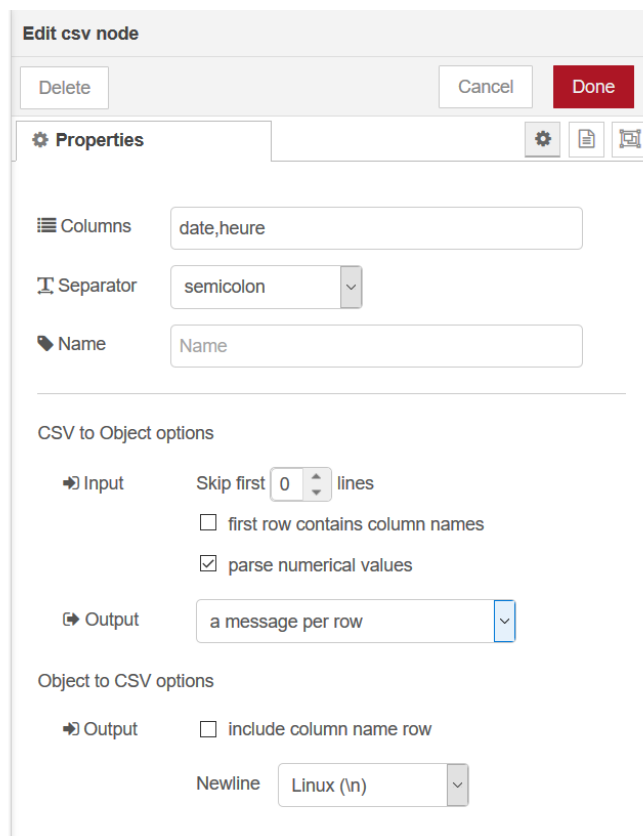


Figure 23. Configuration du node csv



**Remarque :** Les nom des colonnes sont les noms des propriétés de l'objet entrant que l'on désire ajouter au csv.

- Déployez et observez les traces **debug**.

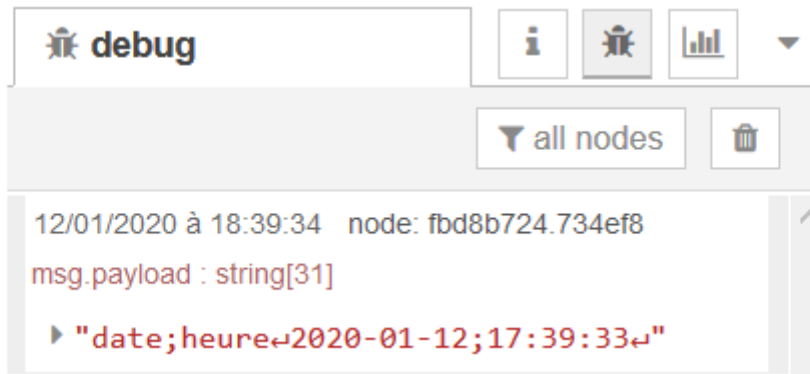


Figure 24. Debug formatage csv

- Ajoutez maintenant un **node file** pour enregistrer les lignes **csv** dans un fichier.

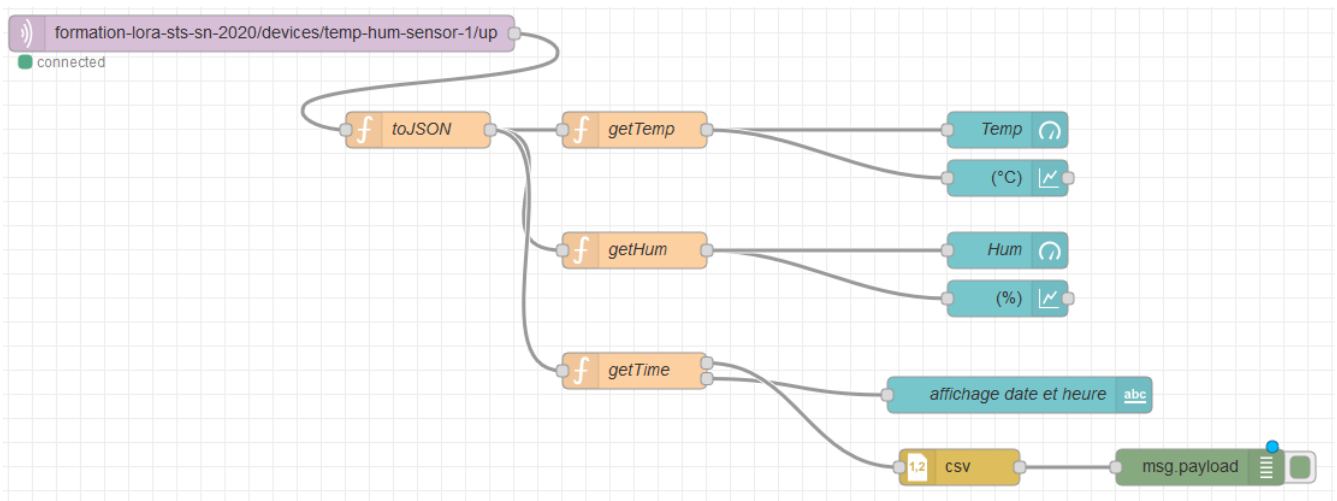


Figure 25. Ecriture dans un fichier csv

- Configurez le **node file** :

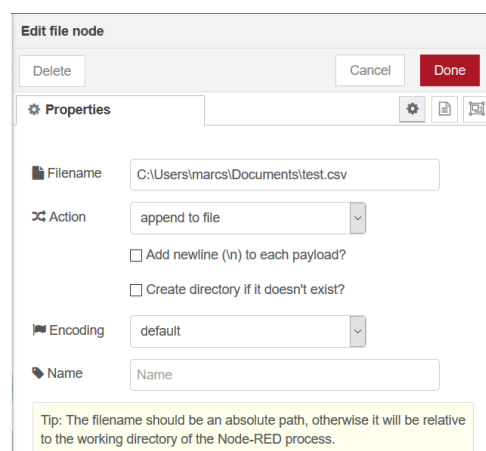


Figure 26. Configuration du node file

- A vous de jouer. A partir de la sortie du **node toJSON**, traitez les données pour ne fournir au **node csv** que l'horodatage, la température et l'humidité relative et les enregistrer dans un fichier csv.

## 6.2. Base de données MySQL

Un node de gestion du classique SGBD MySQL est disponible.

- Commençons par installer le `node mysql` :

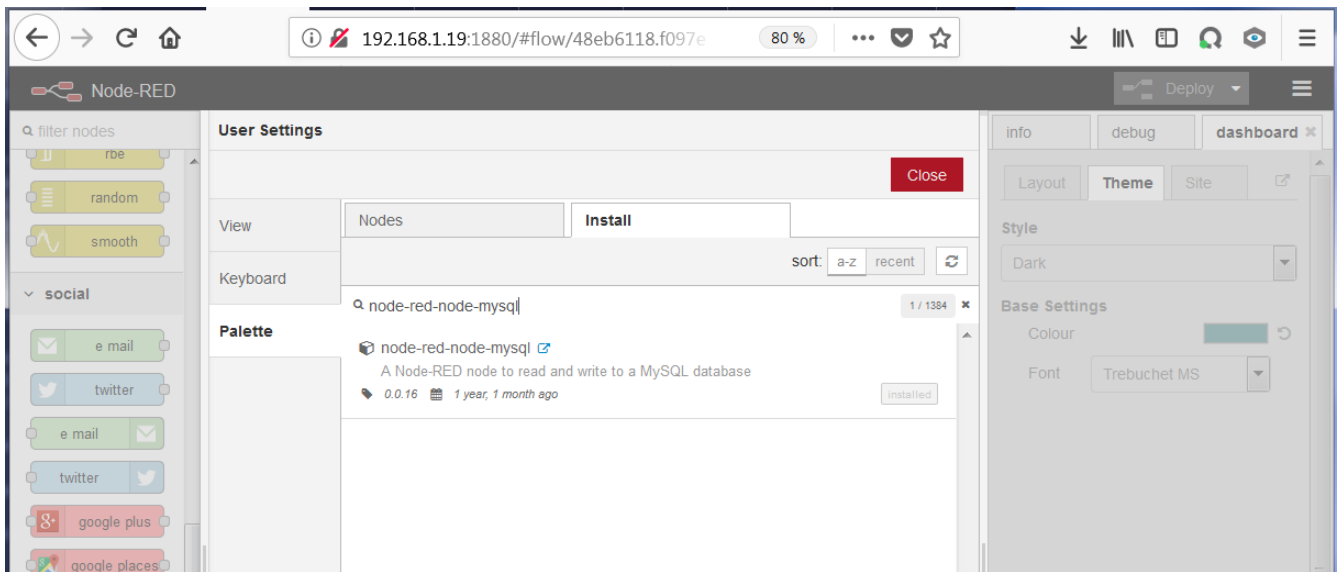


Figure 27. Installations du nodes mysql

- Glissez le `node mysql` sur le `flow` et configurez le pour se connecter à la base de données

Cela sous-entend que nous ayons MySQL installé et démarré avec une base de données accessible.



<https://dev.mysql.com/downloads/mysql/#downloads>

ou plus rapide et plus simple (mais pas sécurisé) : [XAMPP](#)

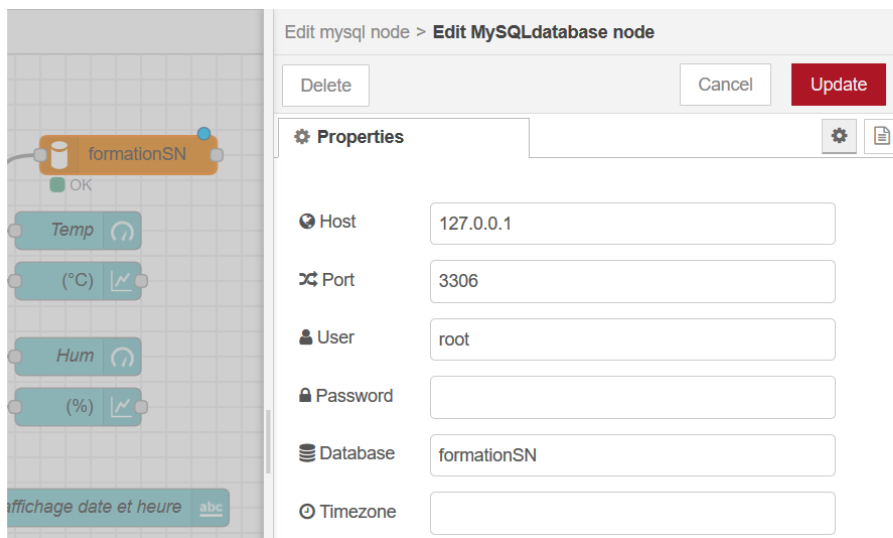


Figure 28. Configuration du node mysql



La requête sql est injectée au `node mysql` sous la forme d'un objet. C'est une chaîne de caractères qui constitue la propriété `topic` de l'objet.

- Ajoutez une fonction entre le **node** `getTemp` et le **node** `mysql`. Saisissez le code ci-dessous qui sous-entend qu'il existe une table `dht11_1` et le champ `temp` dans ma base de données.

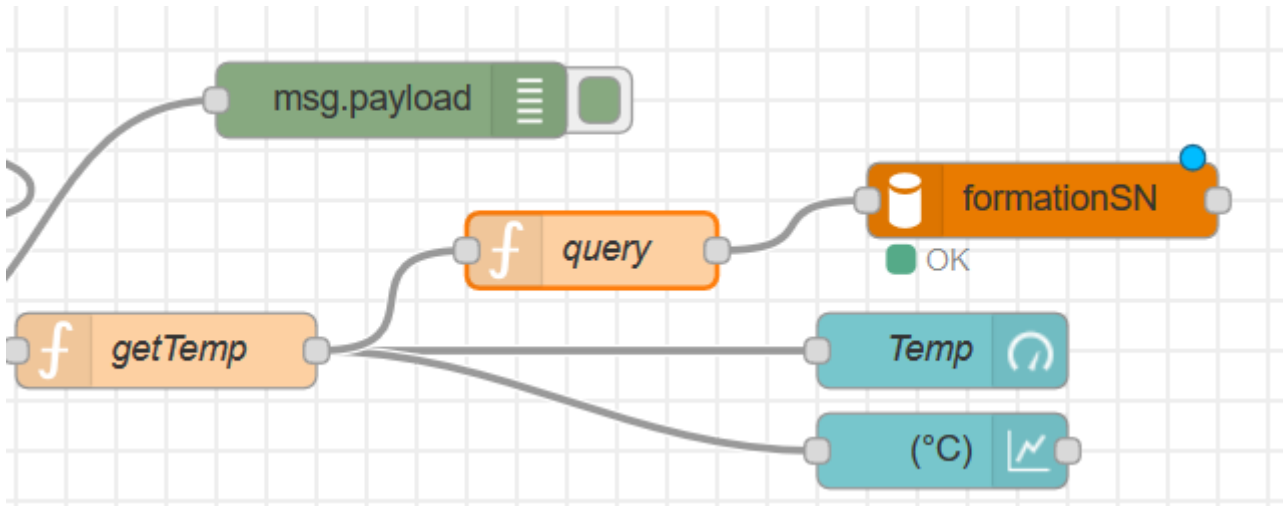


Figure 29. Préparation de la requête d'insertion

#### Code de la fonction query

```
var query = "INSERT INTO `dht11_1` (`temp`) VALUES ('"+msg.payload+"');";
msg.topic = query;
return msg;
```

- Observez dans MySQL l'insertion des données (dans le terminal ou avec phpmyadmin).

The screenshot shows the phpMyAdmin interface. The left sidebar shows the database structure with 'dht11\_1' selected. The main area shows a table with the following data:

temp	hum
19.2	67
19.5	67
19.3	67
19.3	67
19.3	67
19.3	67
19.2	67
19.4	67

The interface also shows a message: "Affichage des lignes 0 - 24 (total de 140, traitement en 0,0007 seconde(s).)" and a SQL query: "SELECT \* FROM `dht11\_1`".

Figure 30. Enregistrement des données dans MySQL

## 6.3. Base de données InfluxDB



**InfluxDB** est une base de données de séries chronologiques. Elle est optimisée pour le stockage et la récupération rapide et à haute disponibilité de données de séries temporelles dans des domaines comme les capteurs de l'**Internet des Objets**.

**InfluxDB** doit être installé sur l'ordinateur ou le serveur.

- Téléchargez et installez **InfluxDB** : <https://portal.influxdata.com/downloads/>
- Dans **node-red**, installez les **nodes** `node-red-contrib-influxdb`.



Figure 31. Installation des nodes `node-red-contrib-influxdb`



InfluxDB utilise le port **8086** par défaut. Pour configurer le **node** `influxdb`, il suffit d'indiquer l'adresse du serveur et le port.

En local, on indiquera donc : `localhost:8086`

**Measurement** indique le nom de la table dans laquelle les données seront introduites.

- Glissez un **node** `influxdb` sur le **flow** et configurez le.
- Ecrivez une fonction `toInfluxDB()` qui fournira en sortie un objet **json** contenant les noms des champs et les valeurs à enregistrer :

```
msg.payload = [{
  "humidity" : msg.payload
}];
return msg;
```

- Déployez et observez dans la console de **InfluxDB** l'insertion des données.

```
> SELECT * FROM historical
name: historical
time                humidity
-----
1549295413210398929 37
1549295415476852641 37
1549295417752839537 37
1549295420027007197 37
1549295422299668872 37
1549295424576904176 37
1549295426851769621 37
1549295429126275721 37
1549295431398993075 37
1549295433674050496 37
```

Figure 32. Observation des données insérées

**InfluxDB** est souvent accompagné de **Telegraf** et **Grafana**. Ce trio forme une solution technique très efficace et open-source pour monitorer un système d'informations

[Lire l'article de Thomas Delannoy de Sup Info](#)

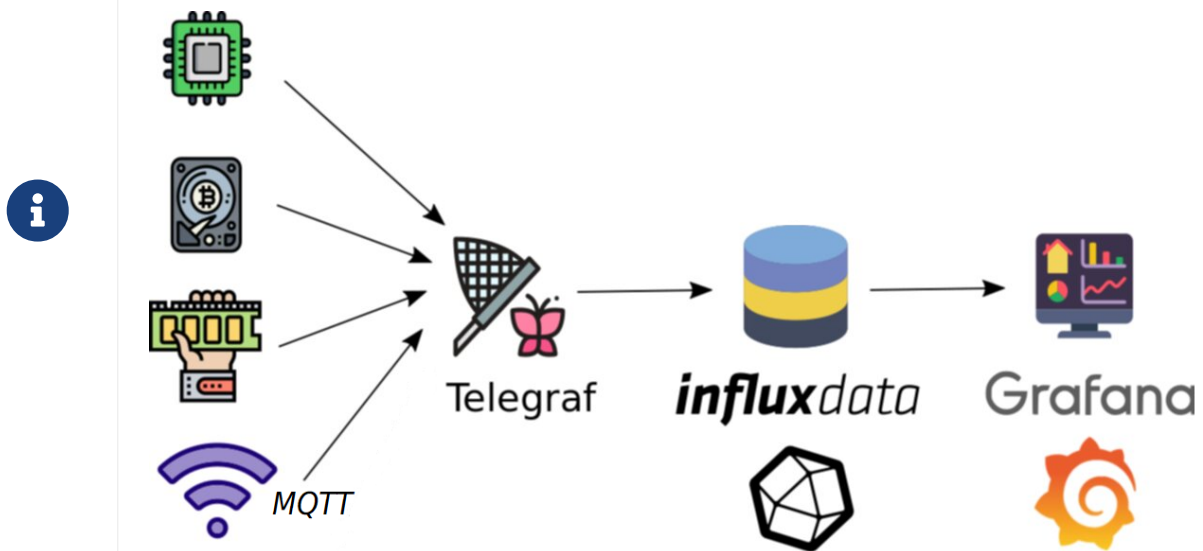


Figure 33. Solution de monitoring Telegraf, InfluxDB et Grafana

## 7. Allez plus loin...

S'il vous reste du temps vous pouvez essayer de mettre en oeuvre une notification automatique à partir de **node-red**...

Utilisez par exemple les **nodes** `node-red-node-email` (<https://flows.nodered.org/node/node-red-node-email>), mais il en existe plein d'autres. A vous de chercher un peu ....



Remarque : Si vous utilisez votre compte google, il doit être configuré pour accorder l'accès aux applications moins sécurisées <https://myaccount.google.com/lesssecureapps>