Node-RED : Introduction

rev1.0 20/04/2020

GOAL

Introduction to Node-RED

PREREQUISITES

Software needed:

- Python
- Node-RED
- NodeJS

Hardware used in this example:

- **none**

What is Node-RED

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model.

Install Node-RED

Node-RED should run on a machine in order to receive and send data via MQTT.

Node-RED can run locally on a machine such as Raspberry Pi or on a server with windows or a linux distro installed.

In our case we will install and run Node-RED in a windows machine.

Before proceding with the installation of Node-RED be shure to have node.js and python installed on your machine

Install Node.js

Installing Node.js is very simple, just go on the download section (link) and download the last stable release for your system.

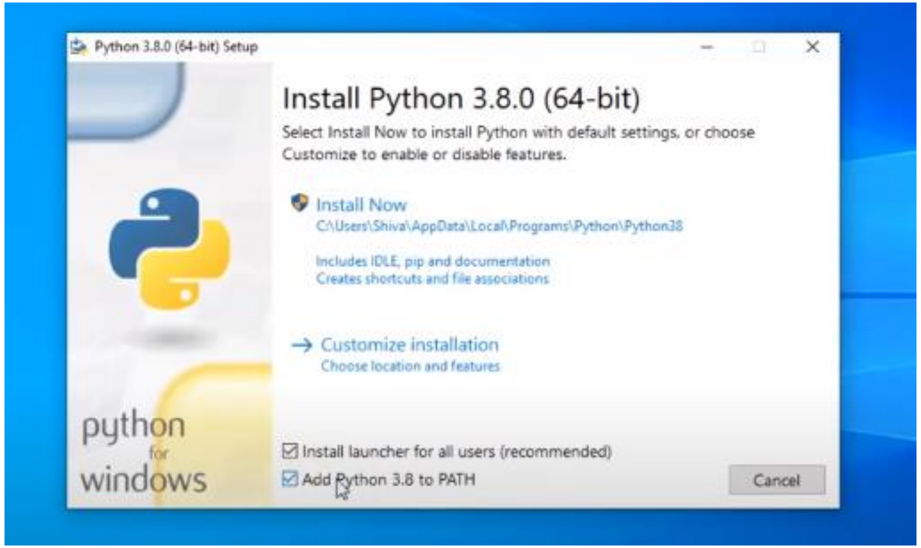
Once downloaded, install it just as a normal application.

Install Python

Python is really simple to install as well: go to the download page (link) and get the last stable release for your system.

Install python as a normal application but **be sure** to add python to **PATH.**

Then check the python installation in the terminal as below.


Python 3.8.0 (64-bit) Setup

Install Python 3.8.0 (64-bit)

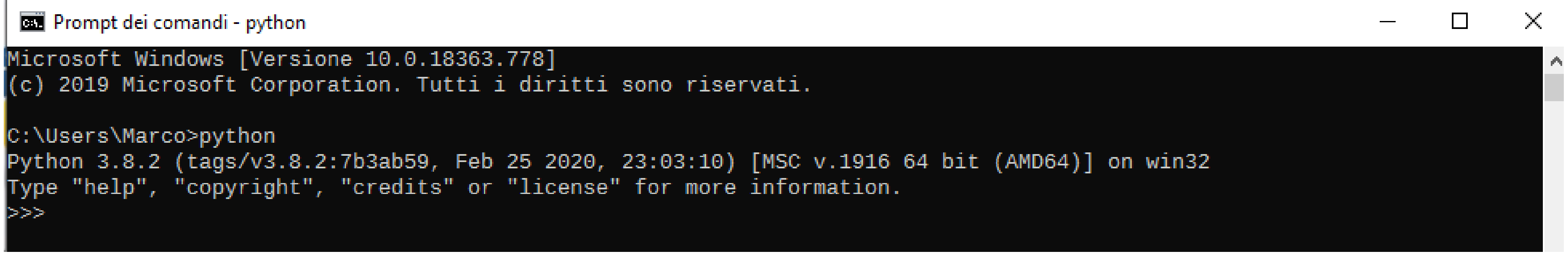
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Shiva\AppData\Local\Programs\Python\Python38

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

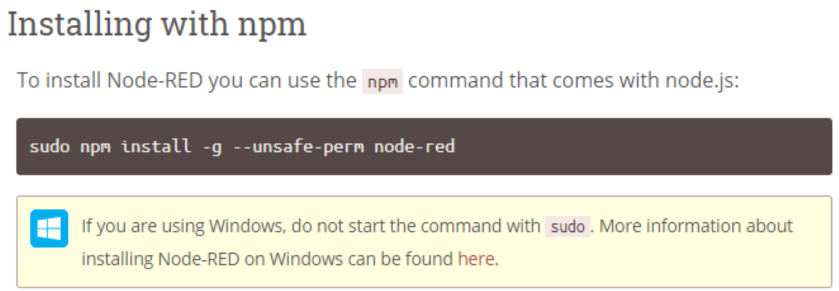
☑ Install launcher for all users (recommended)
☑ Add Python 3.8 to PATH


Prompt dei comandi - python

```
Microsoft Windows [Versione 10.0.18363.778]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Marco>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Install Node-RED

In order to install Node-RED open the terminal use the following command:



Installing with npm

To install Node-RED you can use the `npm` command that comes with node.js:

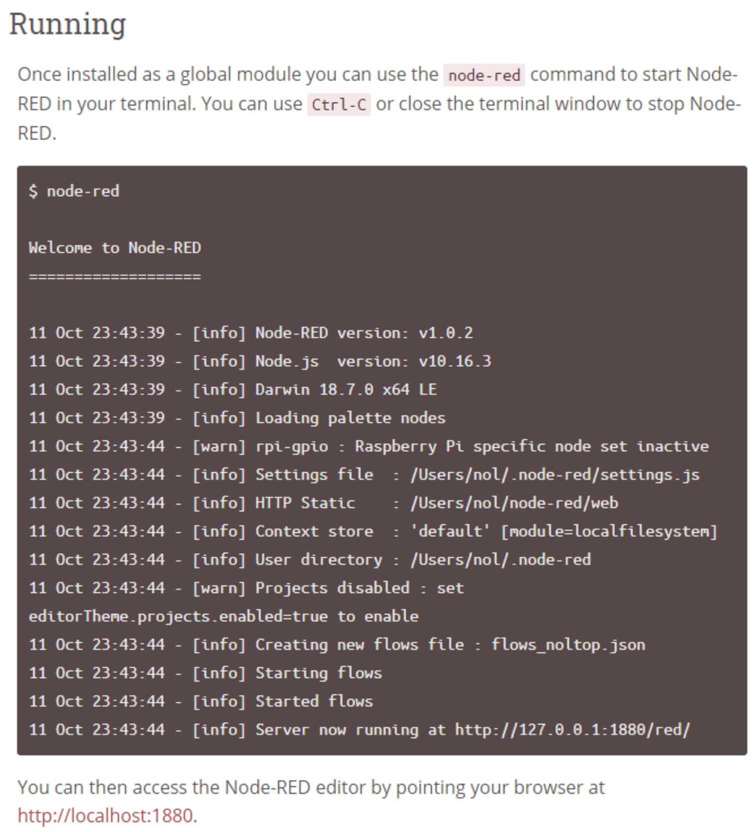
```
sudo npm install -g --unsafe-perm node-red
```

If you are using Windows, do not start the command with `sudo`. More information about installing Node-RED on Windows can be found here.

If you have any trouble here you find the Node-RED installation guide.

To run Node-RED use the command "node-red" on the terminal, then go to

http://localhost:1880 to open Node-Red Console.

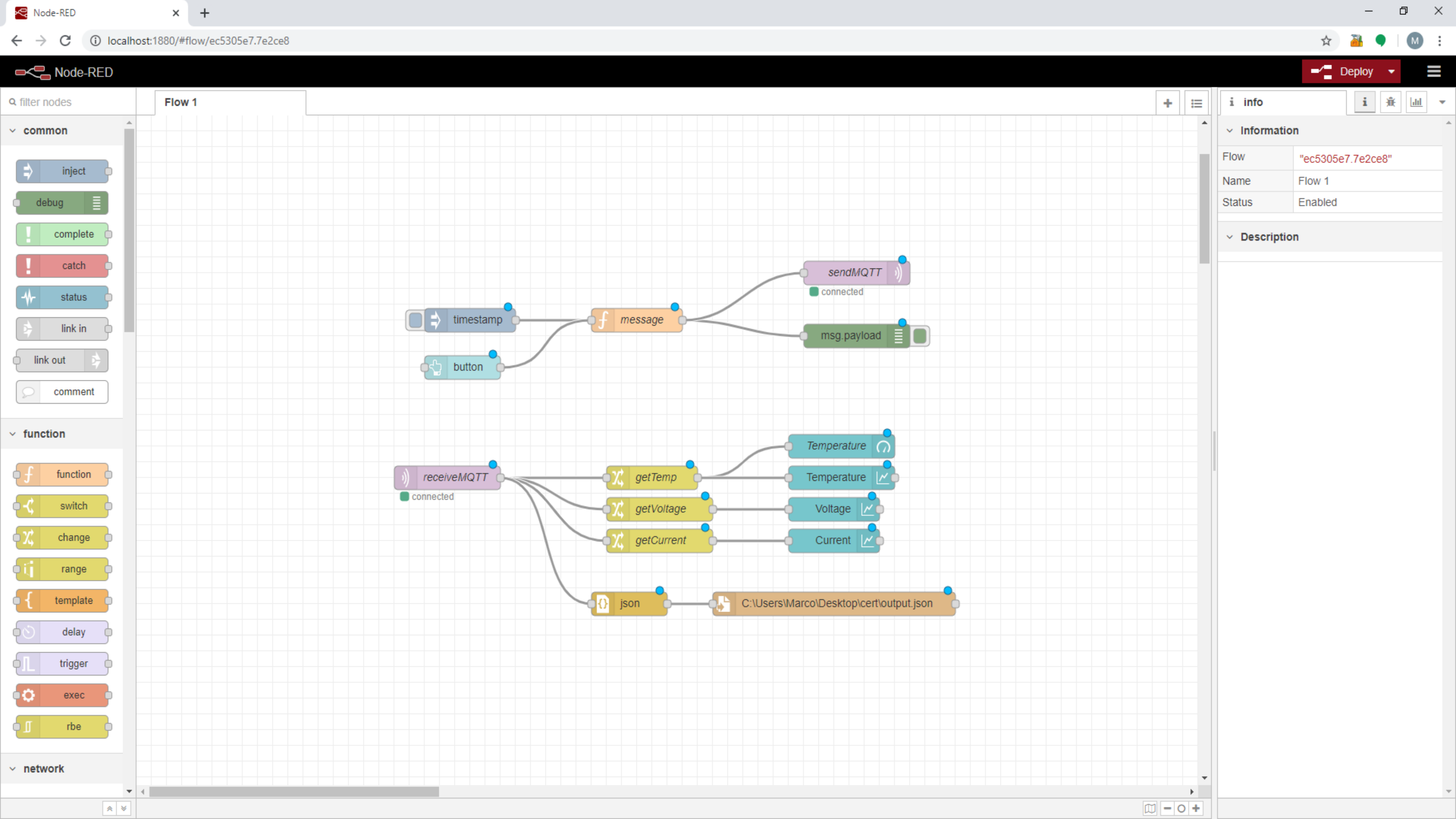


Running

Once installed as a global module you can use the `node-red` command to start Node-RED in your terminal. You can use `Ctrl-C` or close the terminal window to stop Node-RED.

```
$ node-red

Welcome to Node-RED
===================

11 Oct 23:43:39 - [info] Node-RED version: v1.0.2
11 Oct 23:43:39 - [info] Node.js  version: v10.16.3
11 Oct 23:43:39 - [info] Darwin 18.7.0 x64 LE
11 Oct 23:43:39 - [info] Loading palette nodes
11 Oct 23:43:44 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
11 Oct 23:43:44 - [info] Settings file  : /Users/nol/.node-red/settings.js
11 Oct 23:43:44 - [info] HTTP Static    : /Users/nol/node-red/web
11 Oct 23:43:44 - [info] Context store  : 'default' [module=localfilesystem]
11 Oct 23:43:44 - [info] User directory : /Users/nol/.node-red
11 Oct 23:43:44 - [warn] Projects disabled : set
editorTheme.projects.enabled=true to enable
11 Oct 23:43:44 - [info] Creating new flows file : flows_noltop.json
11 Oct 23:43:44 - [info] Starting flows
11 Oct 23:43:44 - [info] Started flows
11 Oct 23:43:44 - [info] Server now running at http://127.0.0.1:1880/red/
```

You can then access the Node-RED editor by pointing your browser at http://localhost:1880.

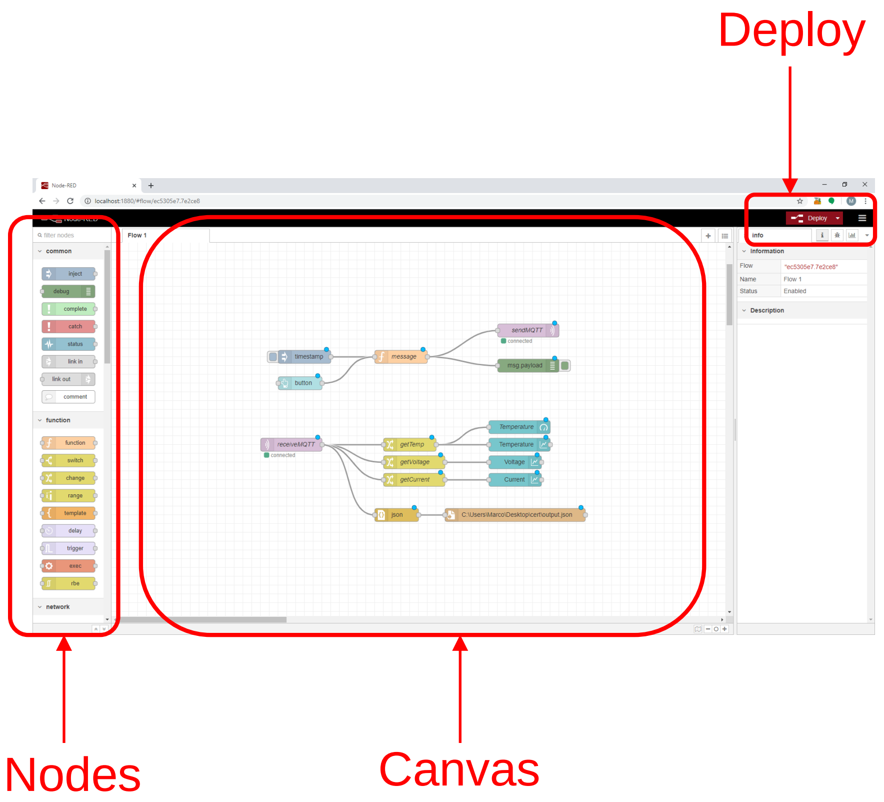


Flow	"ec5305e7.7e2ce8"
Name	Flow 1
Status	Enabled

Node-RED Console

Node-RED have a nice looking GUI interface, see the main components:

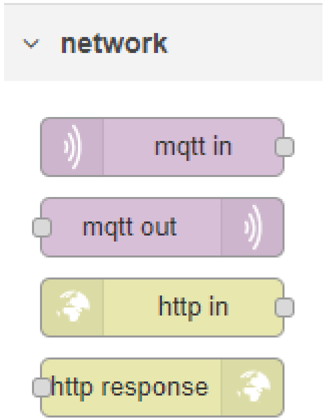
1. *Palette*: here you have the list of all nodes available to build your application
2. *Canvas*: put here the blocks and connect them together, this is your "flow". You can have more than one flow as well.
3. *Deploy*: Save your flow and run it on your machine.

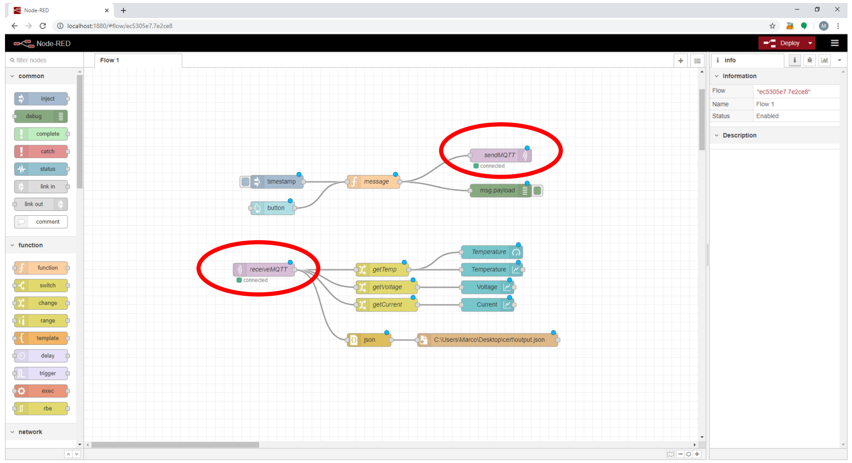


Connect to AWS

The main goal here is to communicate with AWS from Node-RED.

Under the *Network* tab in the palette pick the *mqtt in* and *mqtt out* blocks and put them in the canvas



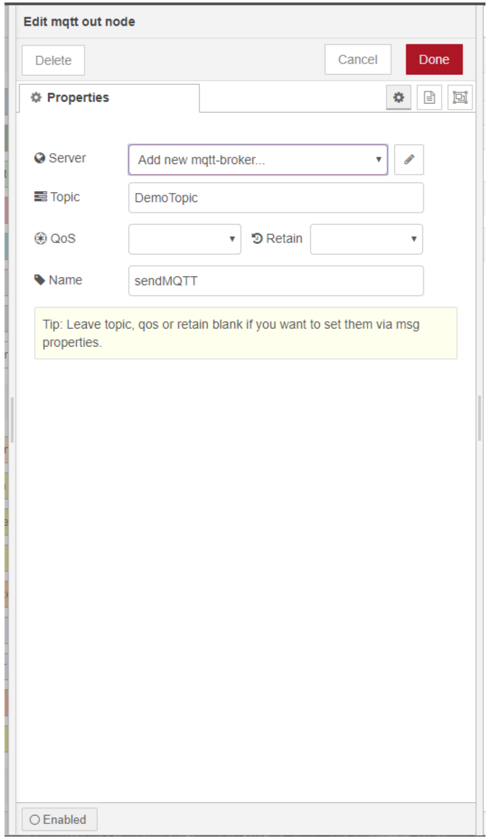


Add new mqtt broker

Double click on the *mqtt out* node, a windows will appear with some infos:

- *Server*: address of the mqtt broker
- *Topic*: subscribe or publish to a topic
- *QoS*: quality of service
- *Name*: name of the node, in this example we called it “sendMQTT”

Click on the *pencil* next to the server field.



Server: Add new mqtt-broker...
Topic: DemoTopic
QoS
Retain
Name: sendMQTT

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

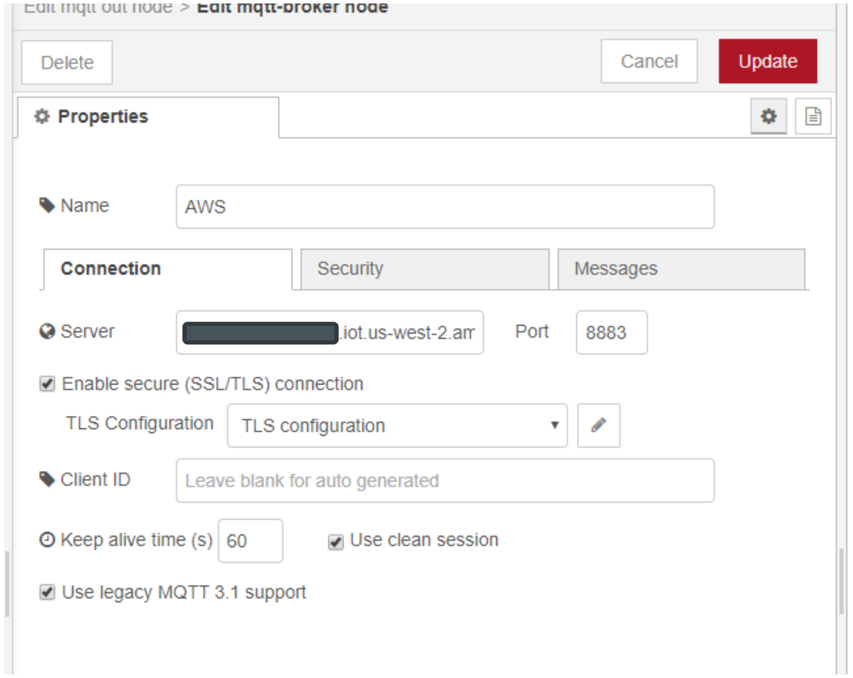
Connect to AWS

Follow the configuration on the picture.

The most important field here is the **server** field: here you have to put your personal endpoint of AWS.

To find the AWS personal endpoint, log in to AWS and go to

AWS IoT Core → Settings


Edit mqtt out node > **Edit mqtt-broker node**

Delete | Cancel | Update

Properties

Name: AWS

Connection | Security | Messages

Server: [redacted].iot.us-west-2.am   Port: 8883

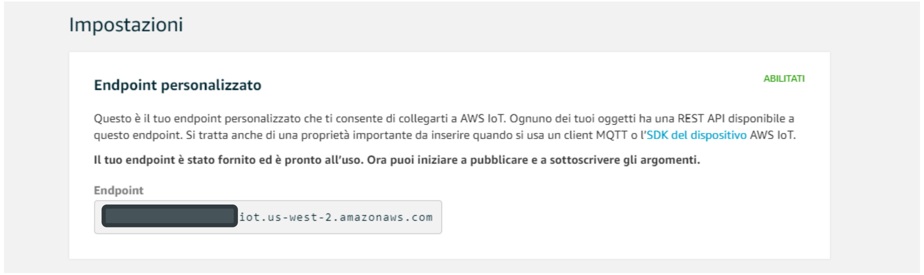
☑ Enable secure (SSL/TLS) connection

TLS Configuration: TLS configuration

Client ID: Leave blank for auto generated

Keep alive time (s): 60   ☑ Use clean session

☑ Use legacy MQTT 3.1 support


Impostazioni

Endpoint personalizzato   ABILITATI

Questo è il tuo endpoint personalizzato che ti consente di collegarti a AWS IoT. Ognuno dei tuoi oggetti ha una REST API disponibile a questo endpoint. Si tratta anche di una proprietà importante da inserire quando si usa un client MQTT o l'SDK del dispositivo AWS IoT.

Il tuo endpoint è stato fornito ed è pronto all'uso. Ora puoi iniziare a pubblicare e a sottoscrivere gli argomenti.

Endpoint

[redacted]iot.us-west-2.amazonaws.com

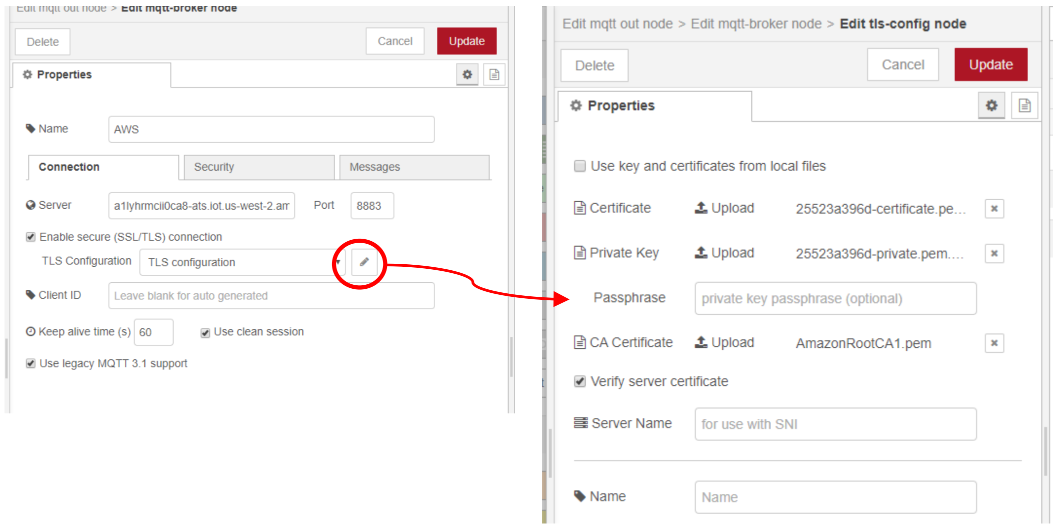
Connect to AWS

The next step is to enable the SSL/TLS connection and upload the certificates.

Click on the pencil and add a new TLS configuration.

Now upload the certificates and the private key, you can choose to upload the files or put a path to your directory.

Now update all setting and **deploy** your *flow*.



Edit mqtt out node > **Edit mqtt-broker node**

Delete | Cancel | Update

Properties

Name	AWS

Connection | Security | Messages

Server	a1lyhrmcii0ca8-ats.iot.us-west-2.am
Port	8883

☑ Enable secure (SSL/TLS) connection

TLS Configuration	TLS configuration
Client ID	Leave blank for auto generated
Keep alive time (s)	60

☑ Use clean session

☑ Use legacy MQTT 3.1 support

Edit mqtt out node > Edit mqtt-broker node > **Edit tls-config node**

Delete | Cancel | Update

Properties

☐ Use key and certificates from local files

Certificate	Upload	25523a396d-certificate.pe...
Private Key	Upload	25523a396d-private.pem....
Passphrase		private key passphrase (optional)
CA Certificate	Upload	AmazonRootCA1.pem

☑ Verify server certificate

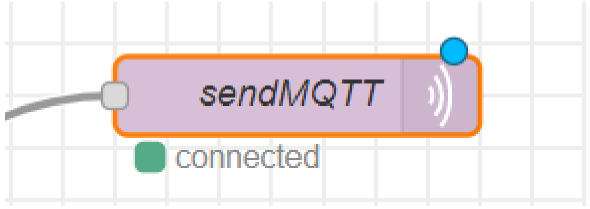
Server Name	for use with SNI
Name	Name

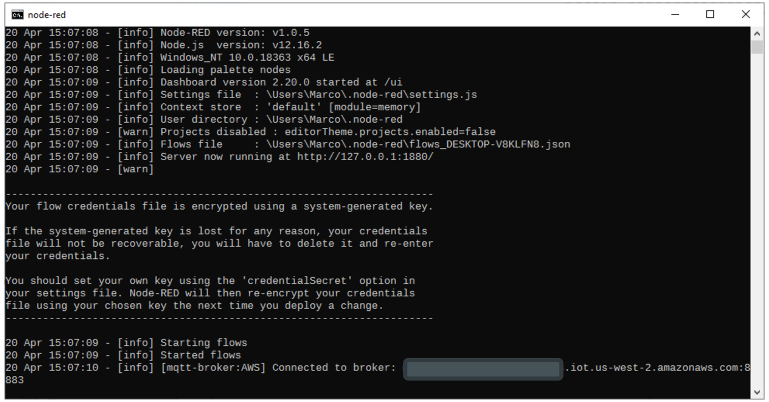
Test connection

If you done all correct, Node-RED should connect with no problems.

After Deploy, Node-RED will connect to AWS and a green square will appear under the node and a confirmation message will appear in the terminal.

Now we are connected to the AWS endpoint.





```
20 Apr 15:07:08 - [info] Node-RED version: v1.0.5
20 Apr 15:07:08 - [info] Node.js  version: v12.16.2
20 Apr 15:07:08 - [info] Windows_NT 10.0.18363 x64 LE
20 Apr 15:07:08 - [info] Loading palette nodes
20 Apr 15:07:09 - [info] Dashboard version 2.20.0 started at /ui
20 Apr 15:07:09 - [info] Settings file  : \Users\Marco\.node-red\settings.js
20 Apr 15:07:09 - [info] Context store  : 'default' [module=memory]
20 Apr 15:07:09 - [info] User directory : \Users\Marco\.node-red
20 Apr 15:07:09 - [warn] Projects disabled : editorTheme.projects.enabled=false
20 Apr 15:07:09 - [info] Flows file     : \Users\Marco\.node-red\flows_DESKTOP-V8KLFN8.json
20 Apr 15:07:09 - [info] Server now running at http://127.0.0.1:1880/
20 Apr 15:07:09 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

20 Apr 15:07:09 - [info] Starting flows
20 Apr 15:07:09 - [info] Started flows
20 Apr 15:07:10 - [info] [mqtt-broker:AWS] Connected to broker: [redacted].iot.us-west-2.amazonaws.com:8
883
```

Send a message

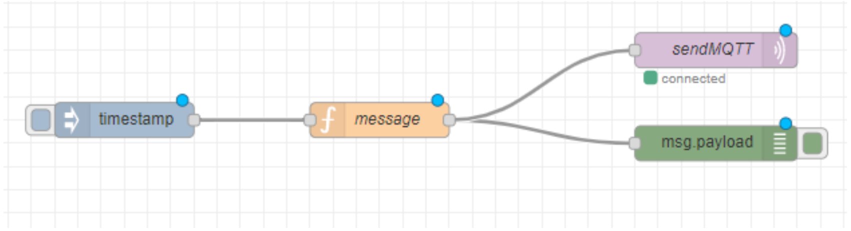
Now try to send a message:

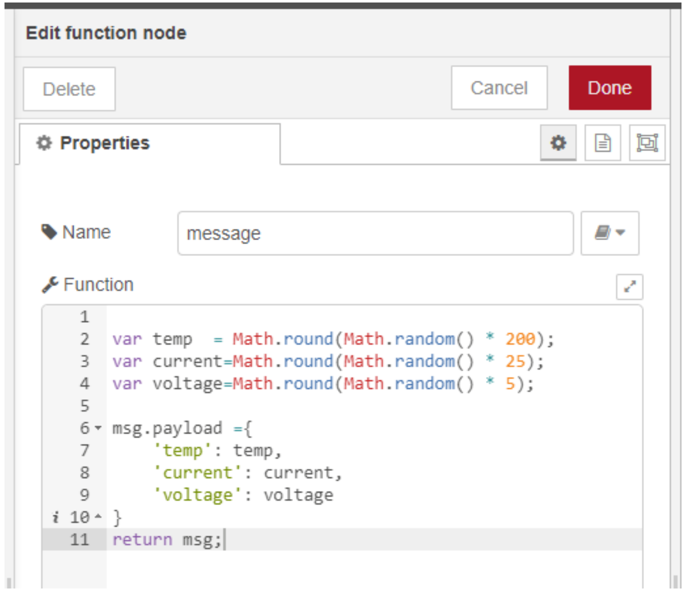
Create a simple flow like the one in figure. The blue node is a inject node: when blue box is pressed, the node activate and send a message to the *mqtt out* node.

The orange node is a *function*: in this example we generate random numbers in a json formatted way.

The green block is a debug block: in this configuration will show the message sent to the AWS broker.

One done deploy your flow.





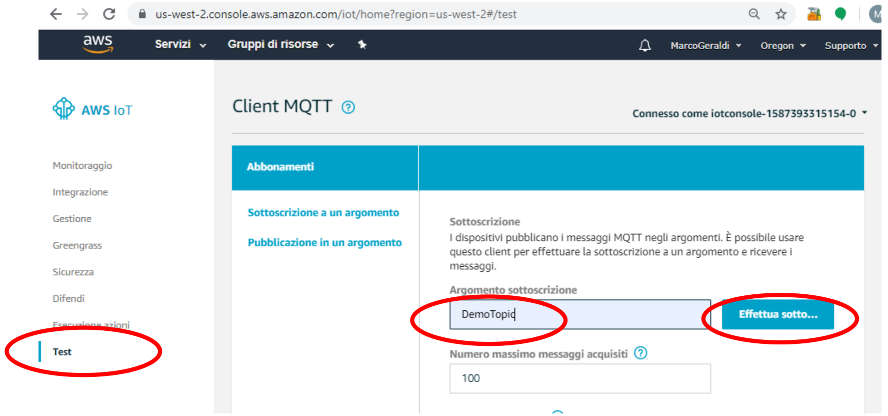
Name: message

Function

```
var temp   = Math.round(Math.random() * 200);
var current=Math.round(Math.random() * 25);
var voltage=Math.round(Math.random() * 5);

msg.payload ={
    'temp': temp,
    'current': current,
    'voltage': voltage
}
return msg;
```

Subscribe to a topic from AWS

Open the AWS IoT Core and go to the **Test** tab end subscribe to the same topic as the Node-RED mqtt node, in our example *DemoTopic*.



us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/test

Connesso come iotconsole-1587393315154-0

Sottoscrizione

I dispositivi pubblicano i messaggi MQTT negli argomenti. È possibile usare questo client per effettuare la sottoscrizione a un argomento e ricevere i messaggi.

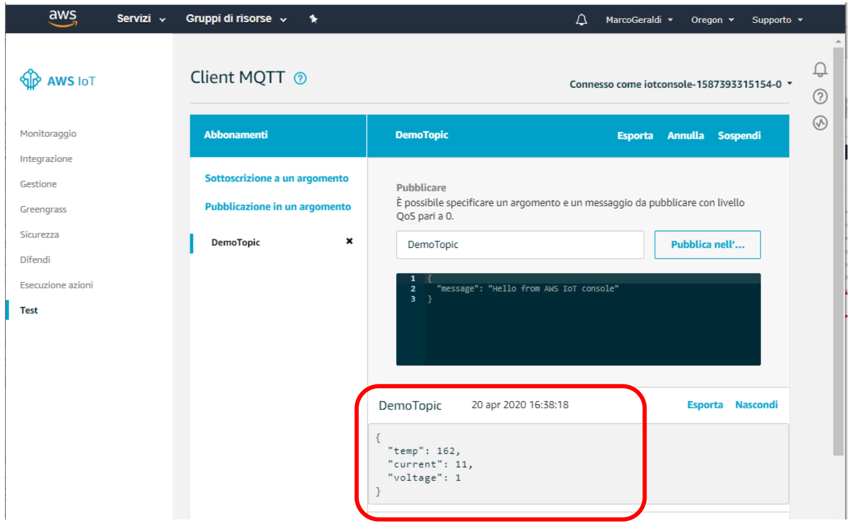
Argomento sottoscrizione: DemoTopic

Numero massimo messaggi acquisiti: 100

Publish to a topic from AWS

From here is possible to monitor all the messages sent and received in the topic. From the console you can publish on the topic by clicking on the **Publish** button.

From Node-RED when the inject node has been activated the message generated in the function will be sent to the *DemoTopic*.



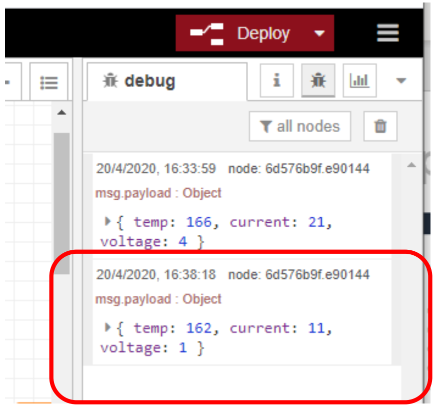
È possibile specificare un argomento e un messaggio da pubblicare con livello QoS pari a 0.

DemoTopic

```
{
  "message": "Hello from AWS IoT console"
}
```

DemoTopic 20 apr 2020 16:38:18

```
{
  "temp": 162,
  "current": 11,
  "voltage": 1
}
```



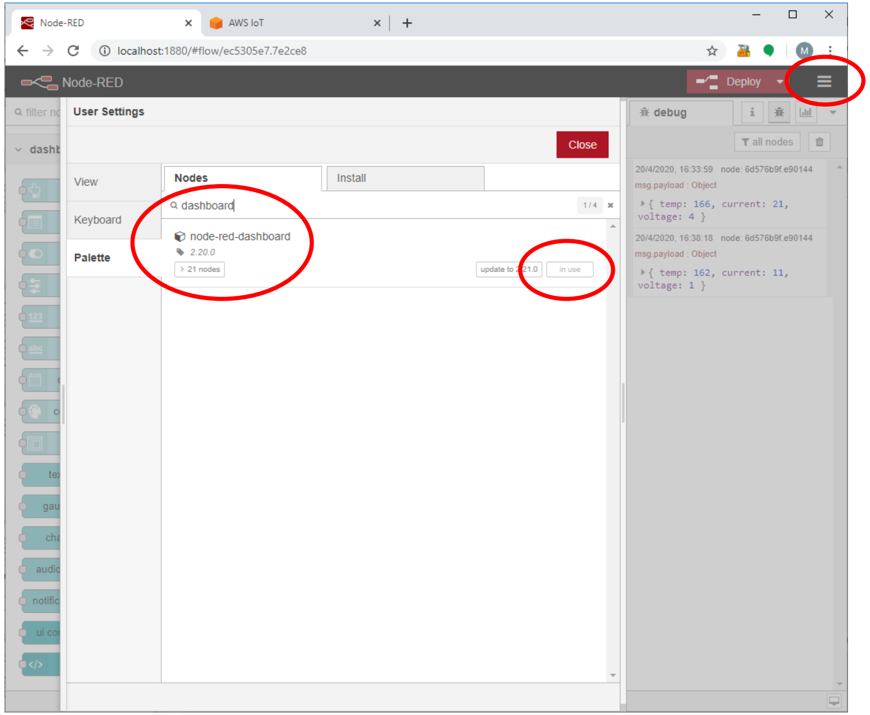
```
20/4/2020, 16:33:59   node: 6d576b9f.e90144
msg.payload : Object
▸{ temp: 166, current: 21,
voltage: 4 }

20/4/2020, 16:38:18   node: 6d576b9f.e90144
msg.payload : Object
▸{ temp: 162, current: 11,
voltage: 1 }
```

Receive a message

From Node-RED it’s possible to subscribe to a topic and receive all the message that other clients have sent to the AWS broker by using the *mqtt in* node.

To display data in a fancy way is possible to use the **dashboard**, a plug-in that includes other nodes such as charts, buttons, gauges, sliders and so, in order to create a great monitor GUI.

By the menu icon, go to **manage palette** and look for the dashboard palette, then install it.



```
20/4/2020, 16:33:59   node: 6d576b9f.e90144
msg.payload : Object
▸{ temp: 166, current: 21,
voltage: 4 }
20/4/2020, 16:38:18   node: 6d576b9f.e90144
msg.payload : Object
▸{ temp: 162, current: 11,
voltage: 1 }
```

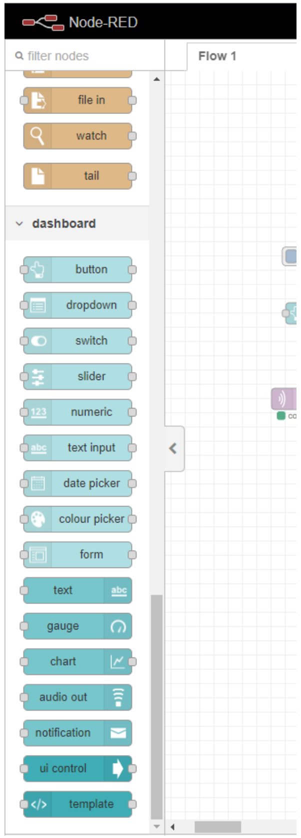
Dashboard plugin

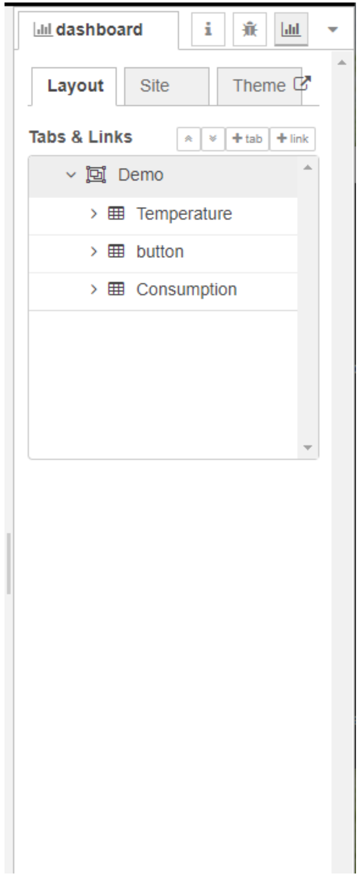
The dashboard palette will be added to your palette of nodes.

From the dashboard menu it’s possible to change themes, layout and so on .

You can access your dashboard from :

http://localhost:1880/ui



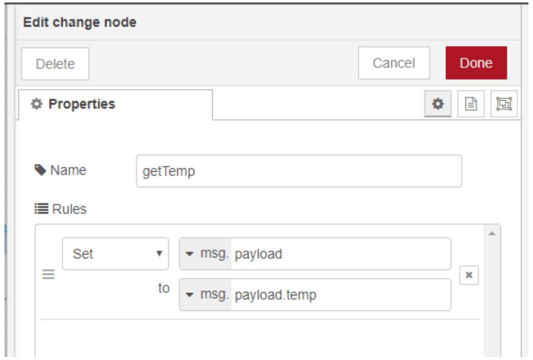


Create a monitoring dashboard

The first step to get the dashboard working is to configure the *mqtt in* node: the configuration is the same as the out node, you should be able to use the same AWS broker configuration.

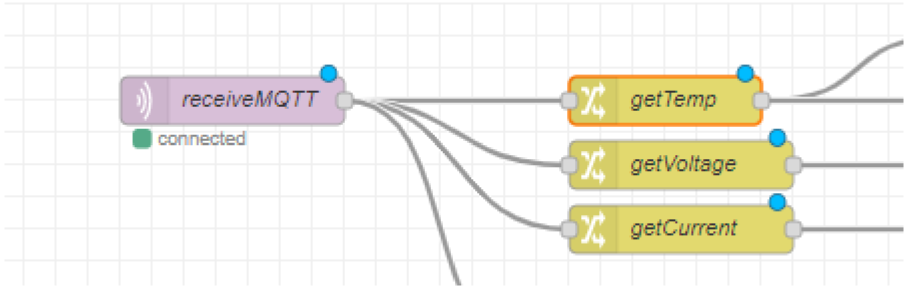
For this example we publish a simulated message that contains 3 main data (*temperature, voltage* and *current*) to the topic and when we receive it from the *mqtt in* node, we save it to a json file for further analysis, and meanwhile, split the message and show it on the dashboard.

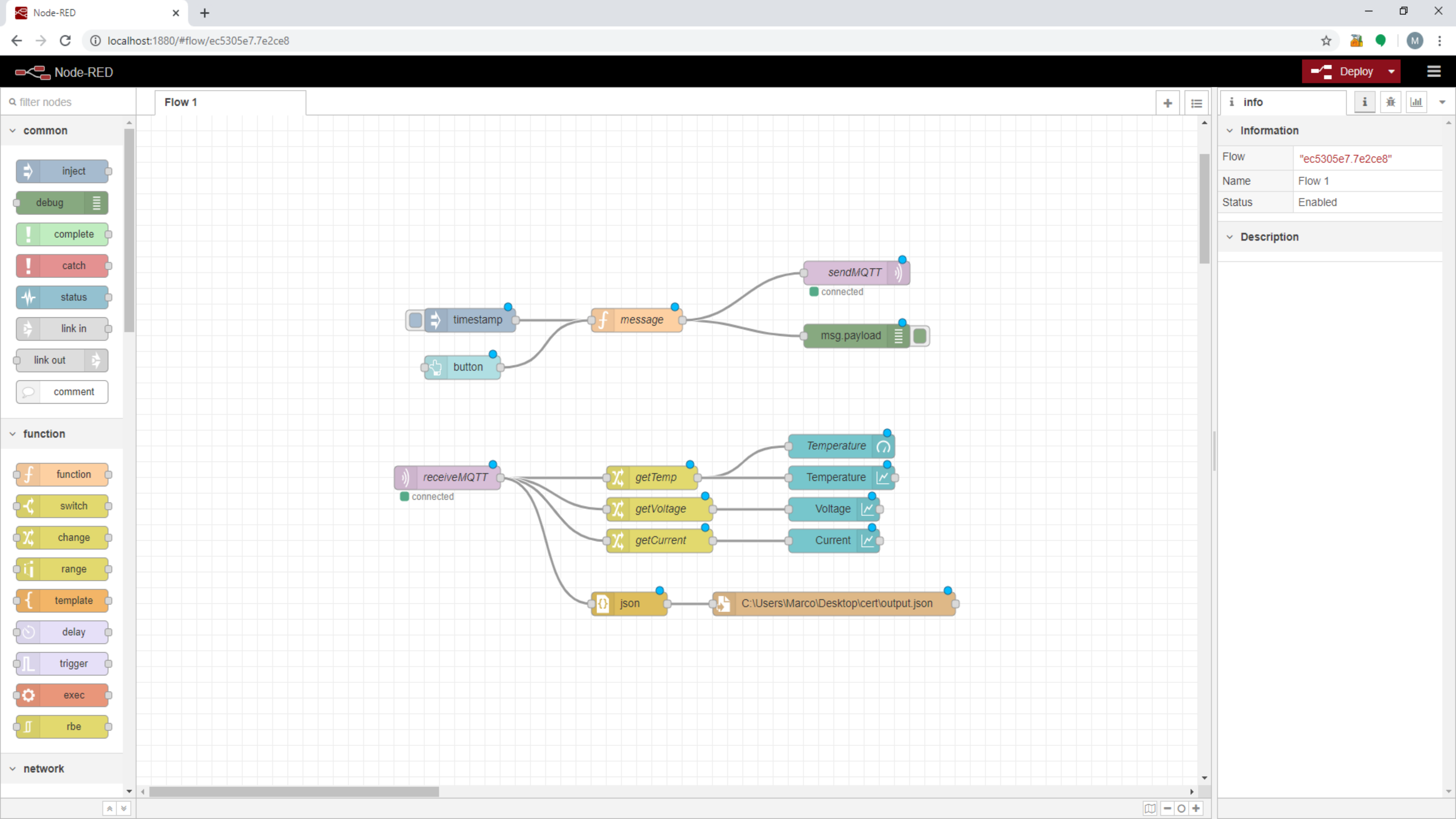
To split the message payload use the *change* node.


Edit change node

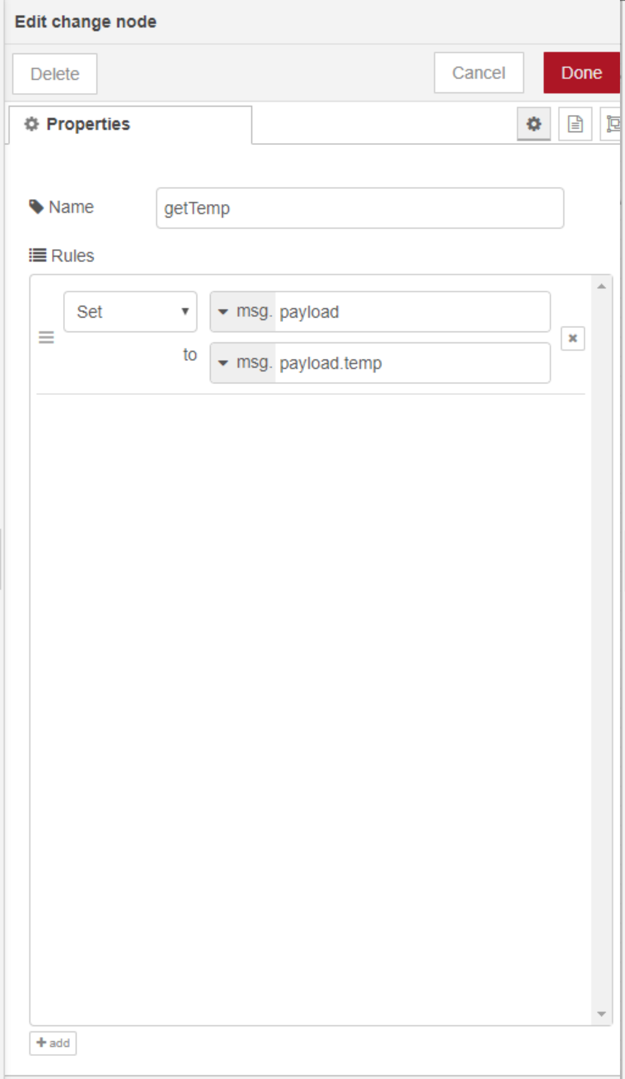
Name: getTemp

Rules: Set msg. payload to msg. payload.temp





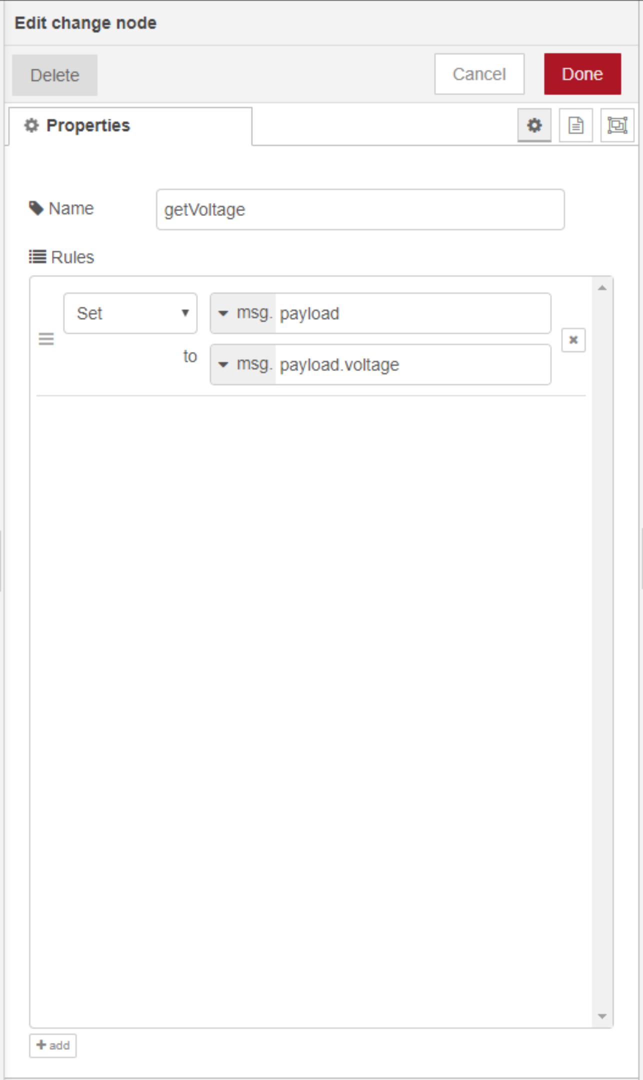
Flow	"ec5305e7.7e2ce8"
Name	Flow 1
Status	Enabled



Edit change node

Name: getTemp

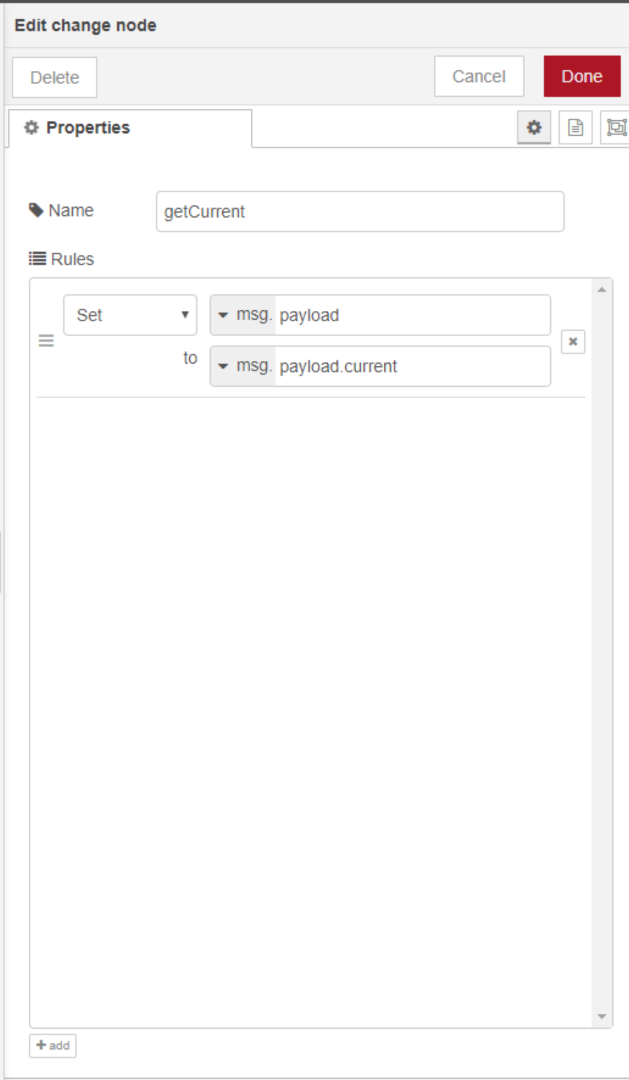
Rules: Set msg. payload to msg. payload.temp



Edit change node

Name: getVoltage

Rules: Set msg. payload to msg. payload.voltage



Edit change node

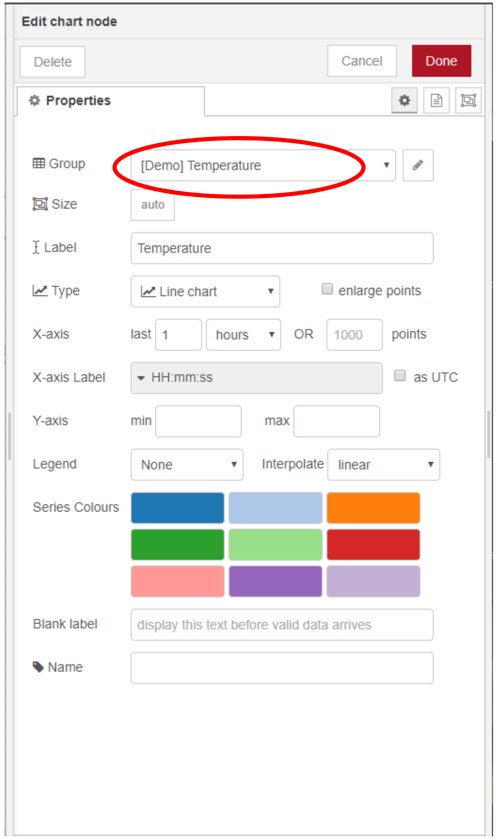
Name: getCurrent

Rules: Set msg. payload to msg. payload.current

Show data with charts and gauges

To display data drag & drop the dashboard nodes as shown.

To organize data in the dashboard is possible to create groups and tabs: for this example we created one tab Demo and each message will be displayed in a different group.


Edit chart node

Group	[Demo] Temperature
Size	auto
Label	Temperature
Type	Line chart ☐ enlarge points
X-axis	last 1 hours OR 1000 points
X-axis Label	HH:mm:ss ☐ as UTC
Y-axis	min max
Legend	None Interpolate linear
Series Colours	🟦 🟦 🟧 🟩 🟩 🟥 🟥 🟪 🟪
Blank label	display this text before valid data arrives
Name	

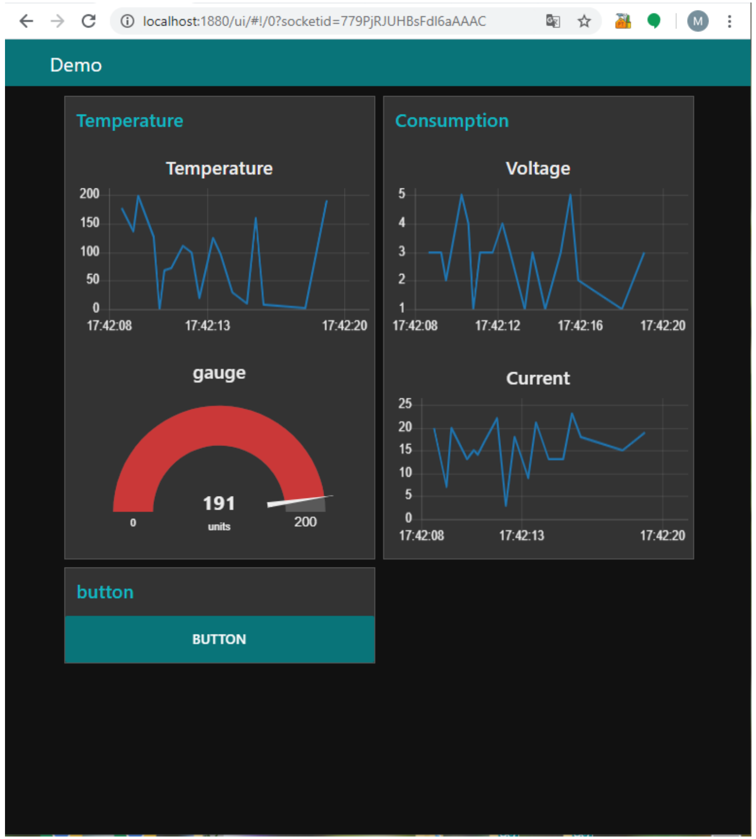
My Dashboard

Open the dashboard
(http://localhost:1880/ui)

When data will be published on the topic, the *mqtt in* node reads the message, split each field and send each data to the respective chart.

In the dashboard of the example we can used a gauge to show the actual «temperature» with a chart that rapresents the temperature over the time. The same for current and voltage.

For testing purposes the button is connected to the *mqtt out* node, so when the button is pressed, the «fake» message has been sent to the AWS broker.



Save data to a file

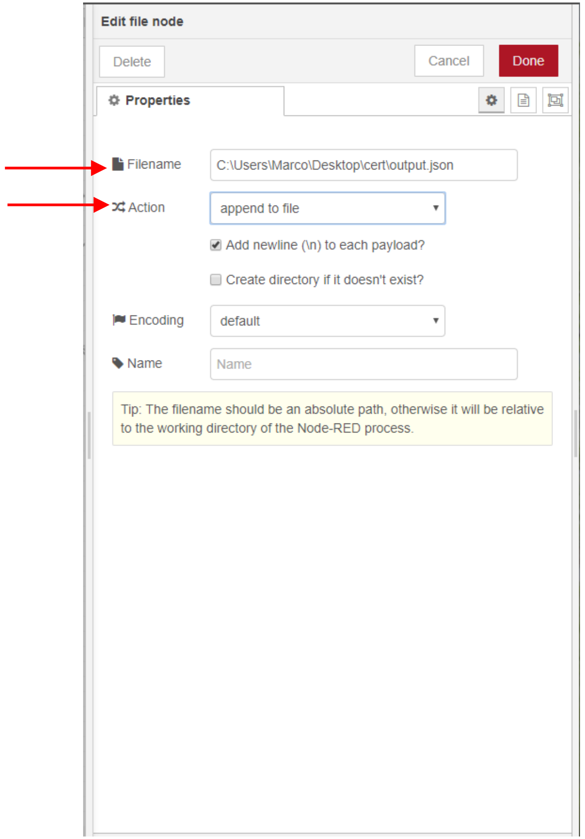
Usually mqtt messages are sent in json format.

The easiest way to store data is to save each stream to a json formatted file and then convert this file to a .csv one.

To save data to a file is very simple, just connect a ***json parser*** to the mqtt in node, and a ***file*** node at the output of the parser.

Double click on the file node, and check the fields:

- *Filename*: is the path where the file is store
- *Action*: behavior of the file when opened



Filename: C:\Users\Marco\Desktop\cert\output.json

Action: append to file

[✓] Add newline (\n) to each payload?

[ ] Create directory if it doesn't exist?

Encoding: default

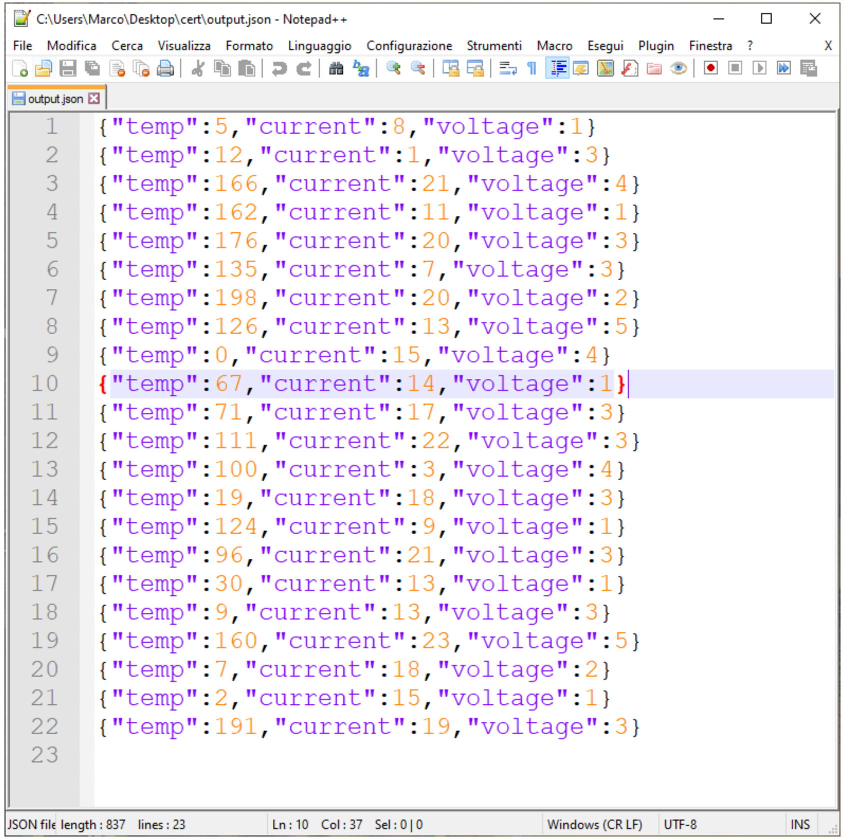
Name: Name

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

Json file

The output file will be something like the one in the picture: you can open with any software, in the example we used notepad++.

In order to convert it to a .csv file for more analysis is possible to use some free tools such as this .


C:\Users\Marco\Desktop\cert\output.json - Notepad++
File Modifica Cerca Visualizza Formato Linguaggio Configurazione Strumenti Macro Esegui Plugin Finestra ?
output.json
JSON file length : 837 lines : 23 Ln : 10 Col : 37 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

```
{"temp":5,"current":8,"voltage":1}
{"temp":12,"current":1,"voltage":3}
{"temp":166,"current":21,"voltage":4}
{"temp":162,"current":11,"voltage":1}
{"temp":176,"current":20,"voltage":3}
{"temp":135,"current":7,"voltage":3}
{"temp":198,"current":20,"voltage":2}
{"temp":126,"current":13,"voltage":5}
{"temp":0,"current":15,"voltage":4}
{"temp":67,"current":14,"voltage":1}
{"temp":71,"current":17,"voltage":3}
{"temp":111,"current":22,"voltage":3}
{"temp":100,"current":3,"voltage":4}
{"temp":19,"current":18,"voltage":3}
{"temp":124,"current":9,"voltage":1}
{"temp":96,"current":21,"voltage":3}
{"temp":30,"current":13,"voltage":1}
{"temp":9,"current":13,"voltage":3}
{"temp":160,"current":23,"voltage":5}
{"temp":7,"current":18,"voltage":2}
{"temp":2,"current":15,"voltage":1}
{"temp":191,"current":19,"voltage":3}
```