

How to integrate NodeRED with Home Assistant – both running inside Docker containers

Putting together automations in Home Assistant using YAML scripting can be a challenge for us non-programmers. But automations are a vital part of any successful Home Automation system.

NodeRED is an Open Source graphical User Interface that allows you to create simple as well as complex automations by just dragging, dropping and connecting nodes on a sheet of paper.

NodeRED can be integrated with Home Assistant. This guide shows you how.

This How-To guide is for those of you with a Home Assistant Core installation. If you have the bundled Home Assistant installation (previously known as HassIO), you do not need to read this guide. NodeRED integration is already available as a simple plug-in.

The prerequisites based on my own set-up, is this;

I have Home Assistant version 0.113.3 running in Container Station on a QNAP, and it's been running there for almost two years without any major issues. It is beyond the scope of this guide to explain how to install and run Home Assistant Core. Whether you run your Home Assistant on a NAS, in a container, in a Virtual Machine, or as a standalone app on a Windows PC or an Apple Mac, makes no difference. What's important is that it can be accessed on your LAN using straight forward HTTP to any reachable IP. It is also assumed throughout this guide that the Home Assistant GUI is available on port 8123. Thus, if you can reach your Home Assistant by typing <http://192.168.10.20:8123> then you are in business (the actual IP will of course differ).

Install NodeRED in Container Station (CS) on a QNAP NAS, this is how to do it:

Pull the following image from Docker HUB: [nodered/node-red:latest](https://hub.docker.com/r/nodered/node-red)

When installing, consider whether you want the container to autostart each time you reboot your NAS;

Container Settings

Image : nodered/node-red

Name : Node-Red

Command : --


Entrypoint : npm start --cache /data/.npm --userDir /data

Auto start :

CPU Limit : 100 %

Memory Limit : 15912 MB

The CPU limit must be within 10-100 %. The memory limit must be within 64-15912MB.

 [Advanced Settings >>](#)

Please restart the container to apply these settings

Apply Cancel

You can name the container whatever you like, and you do not have to add an Entrypoint. The image contains that information for you. Next, click on 'Advanced Settings' and fill in the following fields:

Name	Value
TZ	Europe/Oslo
LANG	nb_NO.UTF-8

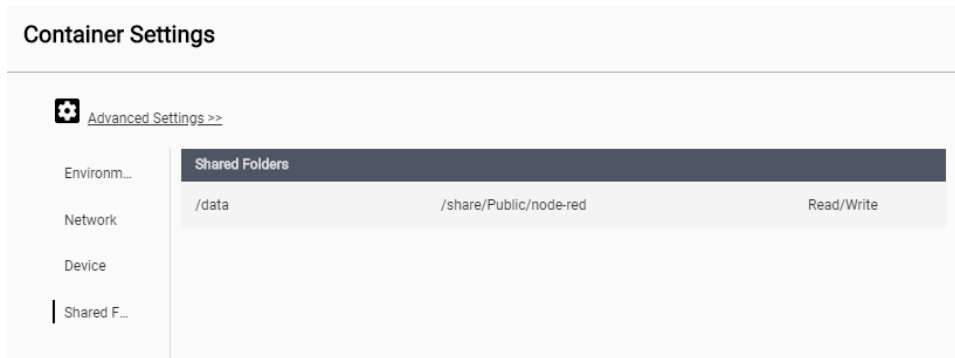
These two fields are not mandatory. If you do not need a locale for your country, then you can skip the LANG variable. Of course, you must insert the right locale for your country. The same goes for TimeZone (TZ). The default setting is UTC time. You only need to change this if your local time is different, i.e. 'Asia/Singapore'

Then, in the Network tab, you can choose between **NAT** or **HOST** as your Network Mode. Both will work, but unless you have a particular reason for choosing **HOST**, you're better off using **NAT**. Then, a random port number will be generated for you which you can change, or just use as-is for access to the NodeRED GUI.

Host	Container	Protocol
32769	1880	TCP

In the above example, I'd use <http://192.168.10.13:32769> to open the NodeRED GUI (of course, your NAS-IP is different from mine). If you chose **HOST** as your network mode, then you would access the NodeRED GUI using this address: <http://192.168.10.13:1880>.

Last, but not least – you should establish a Shared Folder link in order to have all your work, flows, designs and config data stored outside of the container, in case it ever crashes, or you decide to delete it and recreate from DockerHUB. It even makes upgrades and updates much easier, and backing up your data also becomes much easier;



The path shown in the above picture can be anything you choose. Just make sure you create the folder first, otherwise you will not be able to locate it (obviously). I chose to create a folder named *node-red* as a subfolder in my Public Folder. The final path will then automatically be set to */share/Public/node-red*. Add the */data* path as is. This is the folder name used by NodeRED to store all user data and designs and it will now be redirected via this symlink to your external folder automatically.

Now click CREATE and you will be presented with a summary of all settings (those made by you, and those predefined for the NodeRED container), and the container will be created and started.

If the container fails to start (immediate stop), check the output in the console window as well as the error log created in the */data* folder (Shared folder). It should start only on its own basic configuration.

If it runs successfully, then it's time to do the final configuration. Launch the NodeRED GUI as explained above. This should take you to the opening page of NodeRED. There is no login required yet at this point.

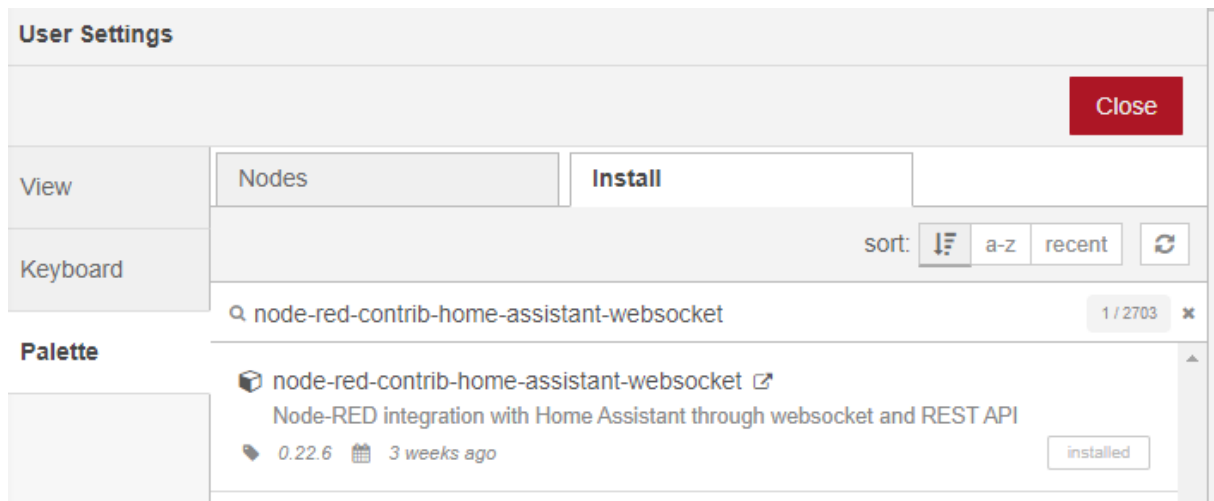
The next step is to connect NodeRED to your Home Assistant instance.

Open your Home Assistant GUI and go to your account page (click on your name at the very bottom on the left side menu). Then, scroll down to the very bottom of your Profile page until you see the option **Long-Lived Access Tokens** and click on *CREATE TOKEN*. A small window will pop up asking you to name your token. Name it *NodeRED*. That's an easy name to remember, and then click OK and the token will automatically be generated for you. Now you need to copy the whole token which is a very long string. Make sure you get the whole string (scroll to the right), and store it somewhere safe, because once you accept it, it will never be shown again and cannot be retrieved again in any way.

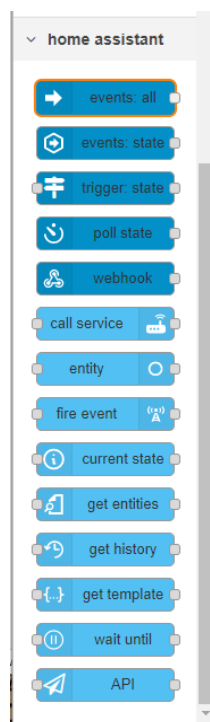
Now turn back to the NodeRED GUI. Now we're going to add the Home Assistant component to NodeRED. Click on the menu icon in the top-right corner (the three horizontal lines) and choose *Manage palette* to open the *User Settings* fly-out. Then click on the *Install* tab. In the search bar, type the following:

```
node-red-contrib-home-assistant-websocket
```

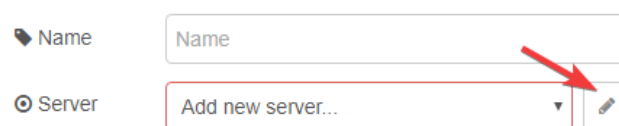
Then click on the *install* button and then click *Close*. This will pull in the node components for Home Assistant and install them in NodeRED.



As a result, you should now be able to find a collection of Home Assistant nodes at the bottom of the node list on the right edge of the NodeRED workspace:



The next step is to configure a connection to your Home Assistant instance so that NodeRED can find it. To do this, just grab and pull an *Events: all* node onto the workspace and then double click on it to open the *Edit events* fly-out. Click on the pencil icon next to the Server field;



This will open the **Edit server node** dialog. Here you should fill in the following three fields: Name, Base URL and Access Token. Leave the rest as-is and *Add/Update* the config and then *Deploy*;

Edit events: all node > **Edit server node**

Delete Cancel Update

Properties

Name Home Assistant

I use the Home Assistant Add-on

Base URL http://192.168.0.8:8123

Access Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc...

Use Legacy API Password

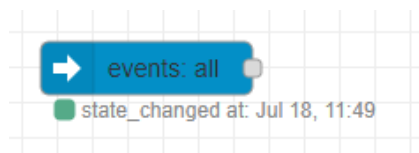
Accept Unauthorized SSL Certificates

State Boolean y|yes|true|on|home|open

Cache Autocomplete Results

The Name is whatever you choose. The Base URL must be the address you use on your LAN to access the Home Assistant GUI. The Access Token is the one you generated and copied as explained earlier in this guide.

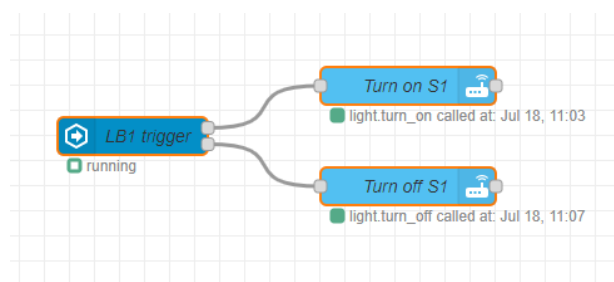
If all goes well, your node symbol should change to:



Meaning that the red triangle is replaced by a blue circle on top and the green box below the node briefly says *Connected* followed by the *state changed at* as seen in the picture above.

What we have done, is to use a node to configure the initial and permanent connection between NodeRED and Home Assistant. Now you can safely delete the *events: all* node. The connection setting is stored and remembered as default for all future use of every Home Assistant node in the node list.

Now you should test your connection by creating a small automation;



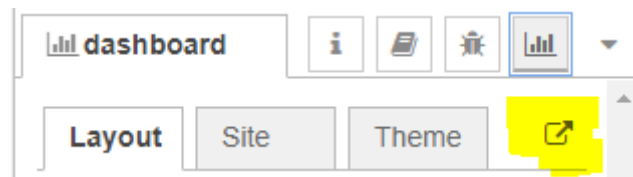
This assumes that you have some entities in your house already configured and available in Home Assistant that you can use. In the above example, I use light bulb named LB1 as a trigger to turn on and off another light bulb named S1. So, when I switch on LB1, the above automation also switches on S1, and vice versa.

You know for sure that the integration between NodeRED and Home Assistant works as it should as soon as you add the first node which in this example is the *events: state* node, and open it to play with the *Entity ID* field. As soon as you start typing into that field, it will look ahead and populate it in real-time with whatever it finds in your Home Assistant list of available entities. If it doesn't, then your connection isn't working and you have to go back and check your setting all over again.

If you decide you need to delete the container and start over again, you should also consider if you need to delete the content of the */share/Public/node-red* folder. Otherwise, incorrect settings may be carried into a new container and you'll find yourself going in loop.

How to build automations using NodeRED is beyond the scope of this guide. Now that you have things running, the rest is up to you. But allow me to make at least one additional recommendation;

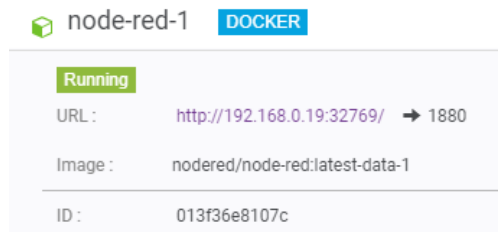
In order to be able to produce a fancy graphical UI from your NodeRED flow projects, you should add the *Dashboard* component to NodeRED. You do this by selecting the *Manage palette* option on the main menu and installing the *node-red-dashboard* component. This will give you a host of new nodes to play with along with the **Dashboard** tab in top menu:



How to update

Normally, the easiest way to update [any] container software to its most recent version, is just to delete the container and recreate it using the “latest” version. But sometimes you may wish to have manual control over the update process, so here is how to do it manually:

NodeRED depend on NPM (Node Package Manager). In order to update NPM to the latest version, you need to log in to your container as root user. To do this, use an SSH client like PuTTY to first log into your NAS, then log into your NodeRED container using its containerID which you can find in CS:



With the ID from the above example, log into the NodeRED container as follows:

```
docker exec -it --user root 013f36e8107c /bin/bash
```

Once inside the container, issue the command:

```
whoami
```

This should return *root* to verify that you indeed are logged in as user root. Then check your current NPM version by typing:

```
npm -v
```

This should give you the current version of your installed Node Package Manager. Then, to update NPM, issue the following command:

```
npm i npm -g
```

This will take a while, and when finished it will report the number of packages added, removed and updated, like in this example:

```
npm@6.14.7
added 16 packages from 2 contributors, removed 17 packages and updated 22
packages in 115.481s
```

It's a good idea to restart the container after a successful update, just to see if everything is ok.

Then, after successfully updating Node Package Manager, you should update NodeRED itself. To do this, jump back into the container as root user (as explained above), and issue the command:

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

This should perform all necessary steps required to update NodeRED. Remember to restart the container after the update.

To see what version is currently out, just visit <https://nodered.org> you'll find the version number in the web heading. To find out which version of NodeRED you currently have installed, just observe the console output when you restart NodeRED. It will display something like this:

Console

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

4 Aug 19:40:21 - [info] Node-RED version: v1.1.2
4 Aug 19:40:21 - [info] Node.js version: v10.21.0
4 Aug 19:40:21 - [info] Linux 4.14.24-qnap x64 LE
4 Aug 19:40:21 - [info] Loading palette nodes
4 Aug 19:40:24 - [info] Dashboard version 2.23.0 started at /ui
4 Aug 19:40:25 - [info] Settings file : /data/settings.js
4 Aug 19:40:25 - [info] Context store : 'default' [module=memory]
4 Aug 19:40:25 - [info] User directory : /data
```

Adding logon security

Adding logon authentication is a must if you plan to access NodeRED remotely. The various options are thoroughly explained here:

<https://nodered.org/docs/user-guide/runtime/securing-node-red>

In short, you have to edit the **adminAuth** setting in your *settings.js* file located in your */data* folder;

```
adminAuth: {
  type: "credentials",
  users: [{
    username: "admin",
    password: "$2a$08$zZWtXTja0fB1pzD4sHcMyOCMyz2Z6dNbgENOMcxWV9DN.",
    permissions: "*"
  }]
},
```

To generate the password hash, jump into the container and type:

```
node -e "console.log(require('bcryptjs').hashSync(process.argv[1], 8));"
your-password-here
```

(all on one line)