Running under Docker

Container versions

We publish three tagged versions of the container to [DockerHub](https://hub.docker.com/r/nodered/node-red-docker/):

* latest - uses [official Node.JS v4 base image](https://hub.docker.com/_/node/).
* slim - uses [Alpine Linux base image](https://hub.docker.com/r/mhart/alpine-node/).
* rpi - uses [RPi-compatible base image](https://hub.docker.com/r/hypriot/rpi-node/).

Using Alpine Linux reduces the built image size (~100MB vs ~700MB) but removes standard dependencies that are required for native module compilation. If you want to add modules with native dependencies, use the standard image or extend the slim image with the missing packages.

Additional images using a newer Node.js v8 base image are now available with the following tags.

* **latest-v8**
* **slim-v8**
* **rpi-v8**

Node-RED releases are also tagged with a version label, allowing you to fix on a specific version: latest:X.Y.Z, slim:X.Y.Z, rpi:X.Y.Z.

Quick start

To run the latest container:

docker run -it -p 1880:1880 --name mynodered nodered/node-red-docker

*Note:* on a Raspberry Pi you must use a rpi-tagged image: nodered/node-red-docker:rpi.

This command will download the nodered/node-red-docker container from DockerHub and run an instance of it with the name of mynodered and with port 1880 exposed. In the terminal window you will see Node-RED start. Once started you can then browse to http://{host-ip}:1880 to access the editor.

Hit Ctrl-p Ctrl-q to detach from the container. This leaves it running in the background.

To reattach to the container:

docker attach mynodered

To stop the container:

docker stop mynodered

To start the container:

docker start mynodered

*Note* : your flows will be stored in the file called flows.json within the container. This can be customised by setting the FLOWS environment parameter:

docker run -it -p 1880:1880 -e FLOWS=my\_flows.json nodered/node-red-docker

Node.js runtime arguments can be passed to the container using an environment parameter (NODE\_OPTIONS). For example, to fix the heap size used by the Node.js garbage collector you would use the following command:

docker run -it -p 1880:1880 -e NODE\_OPTIONS="--max\_old\_space\_size=128" nodered/node-red-docker

Customising

The container uses the directory /data as the user configuration directory. To add additional nodes you can open shell into the container and run the appropriate npm install commands:

# Open a shell in the container

docker exec -it mynodered /bin/bash

# Once inside the container, npm install the nodes in /data

cd /data

npm install node-red-node-smooth

exit

# Restart the container to load the new nodes

docker stop mynodered

docker start mynodered

Storing data outside of the container

It is possible to mount the /data path on an external volume:

docker run -it -p 1880:1880 -v ~/node-red-data:/data --name mynodered nodered/node-red-docker

This command mounts the host’s ~/node-red-data directory as the user configuration directory inside the container.

Adding extra nodes to the container can then be accomplished by running npm install on the host machine:

cd ~/node-red-data

npm install node-red-node-smooth

docker stop mynodered

docker start mynodered

*Note* : Modules with a native dependencies will be compiled on the host machine's architecture. These modules will not work inside the Node-RED container unless the architecture matches the container's base image. For native modules, it is recommended to install using a local shell or update the package.json and re-build.

Building the container from source

The Dockerfiles for these containers are maintained [here](https://github.com/node-red/node-red-docker), each under its own branch.

To build your own version:

git clone https://github.com/node-red/node-red-docker.git

cd node-red-docker

# Build it with the desired tag

docker build -f <version>/Dockerfile -t mynodered:<tag> .

Building a custom image

Creating a new Docker image, using the public Node-RED images as the base image, allows you to install extra nodes during the build process.

1. Create a file called Dockerfile with the content:
2. FROM nodered/node-red-docker
3. RUN npm install node-red-node-wordpos
4. Run the following command to build the image:
5. docker build -t mynodered:<tag> .

That will create a Node-RED image that includes the wordpos nodes.

Updating

Updating the base container image is as simple as

docker pull nodered/node-red-docker

docker stop mynodered

docker start mynodered

Linking Containers

You can link containers “internally” within the Docker runtime by using the --link option.

For example, if you have a container that provides an MQTT broker container called mybroker, you can run the Node-RED container with the link parameter to join the two:

docker run -it -p 1880:1880 --name mynodered --link mybroker:broker nodered/node-red-docker

This will make broker a known hostname within the Node-RED container that can be used to access the service within a flow, without having to expose it outside of the Docker host.

[{"id":"190c0df7.e6f3f2","type":"mqtt-broker","broker":"broker","port":"1883","clientid":""},{"id":"37963300.c869cc","type":"mqtt in","name":"","topic":"test","broker":"190c0df7.e6f3f2","x":226,"y":244,"z":"f34f9922.0cb068","wires":[["802d92f9.7fd27"]]},{"id":"edad4162.1252c","type":"mqtt out","name":"","topic":"test","qos":"","retain":"","broker":"190c0df7.e6f3f2","x":453,"y":135,"z":"f34f9922.0cb068","wires":[]},{"id":"13d1cf31.ec2e31","type":"inject","name":"","topic":"","payload":"","payloadType":"date","repeat":"","crontab":"","once":false,"x":226,"y":157,"z":"f34f9922.0cb068","wires":[["edad4162.1252c"]]},{"id":"802d92f9.7fd27","type":"debug","name":"","active":true,"console":"false","complete":"false","x":441,"y":261,"z":"f34f9922.0cb068","wires":[]}]