


# How to Connect Your Internet of Things with Node-RED on Ubuntu 16.04



Posted September 23, 2016  91.9k

MISCELLANEOUS

NGINX

LET'S ENCRYPT

UBUNTU

UBUNTU 16.04

By: Brian Boucheron

## Introduction

Node-RED is a switchboard for the Internet of Things, a visual tool that helps you connect your favorite apps, websites, and hardware together to do new and useful things. Most often compared to [IFTTT](#) or the late Yahoo Pipes, Node-RED has a much more powerful and flexible interface, and a large open source community creating *nodes* to interact with a wide variety of apps and services.

In this tutorial, we'll install Node.js and Node-RED, get an SSL certificate from Let's Encrypt, and use Nginx to handle secure connections for Node-RED.

## Prerequisites

To follow this tutorial, you will need:

- One Ubuntu 16.04 server with a non-root sudo user and basic firewall set up by following [this Ubuntu 16.04 sever setup tutorial](#). For this tutorial, we'll use a user called **sammy**, but of course you can choose whatever you like and substitute as needed.
- The web server Nginx installed, with the firewall updated to allow traffic on ports 80 and 443 (**Nginx Full**), as explained in [How To Install Nginx on Ubuntu 16.04](#)
- A domain name pointed at your server, as described in [How To Set Up a Host Name with DigitalOcean](#). This tutorial will use `node-red.example.com` throughout.

- Let's Encrypt installed, and a certificate generated for the domain you configured above. [How To Secure Nginx with Let's Encrypt on Ubuntu 16.04](#) will walk you through the necessary steps. You can ignore the steps regarding Nginx configuration (steps 3–5), as we'll cover that here. Just make sure you get a certificate successfully issued, and set up the `cron` job to handle automatic renewals.

## Step 1 — Installing Node.js and npm

Ubuntu 16.04 makes it easy to install the latest long term support (LTS) release of Node.js because it's included in the default repository.

```
$ sudo apt-get install nodejs-legacy
```

The command installs Node.js v4.2.x LTS (long term support), which means the Node.js Foundation will continue to support this version for 30 months from its release date of October 12, 2015.

**Note:** It's important to install the `-legacy` version of the package because Node-RED's startup scripts expect your Node.js binary to be named `node`, but the standard package uses `nodejs` instead. This is due to a naming conflict with a preexisting package.

Verify that the installation was successful by checking the version.

```
$ node -v
```

You'll see Node.js output its version number:

Output

```
v4.2.6
```

Node Package Manager (`npm`) helps you install and manage Node.js software packages, and we'll use it to install Node-RED. Install `npm` using `apt-get`.

```
$ sudo apt-get install npm
```

To verify the install was successful, ask `npm` to print its version information:

```
$ npm -v
```

Output

```
3.5.2
```

If it prints a version number without error, we can continue on to our next step, where we'll use `npm` to install Node-RED itself.

## Step 2 — Installing Node-RED

Use `npm` to install `node-red` and a helper utility called `node-red-admin`.

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

`npm` normally installs its packages into your current directory. Here, we use the `-g` flag to install packages 'globally' so they're placed in standard system locations such as `/usr/local/bin`. The `--unsafe-perm` flag helps us avoid some errors that can pop up when `npm` tries to compile native modules (modules written in a compiled language such as C or C++ vs. JavaScript).

After a bit of downloading and file shuffling, you'll be returned to the normal command line prompt. Let's test our install.

First, we'll need to open up a port on our firewall. Node-RED defaults to using port `1880`, so let's allow that.

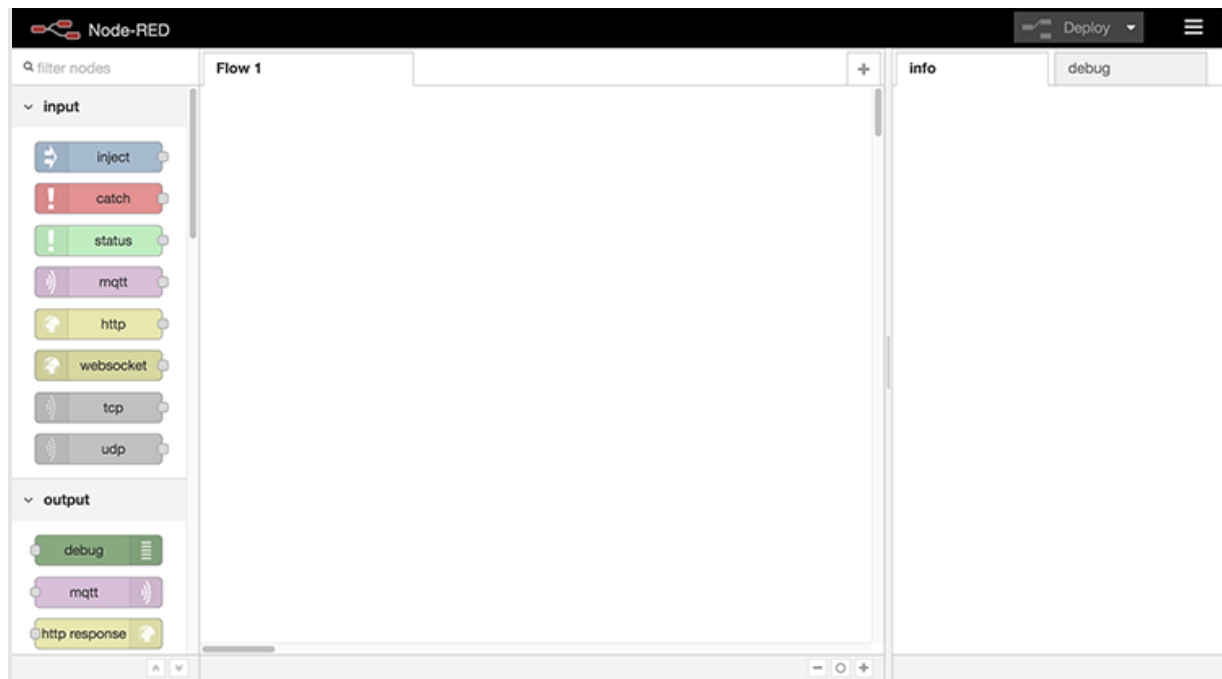
```
$ sudo ufw allow 1880
```

And now launch Node-RED itself. No `sudo` is necessary, as port `1880` is high enough to not require root privileges.

```
$ node-red
```

Some "Welcome to Node-RED" messages will print to the terminal. On your computer, point a web browser to port `1880` of the server. In our example, that's `http://node-red.example.com:1880`. The main admin interface of Node-RED will load.

---



If it worked, you can type `CTRL+C` in your terminal to shut down Node-RED and return to the command prompt. We've installed Node-RED successfully and tested it out, so next, we'll set it up to launch during system startup.

## Step 3 — Launching Node-RED on Startup

In order to start Node-RED automatically on startup, we'll need to install a `node-red.service` file instead of the more traditional init script. This is because Ubuntu 16.04 is the first LTS release that uses `systemd` for its init system. You can find a summary of this and other Ubuntu 16.04 changes in [What's New in Ubuntu 16.04](#).

Open a blank service file called `node-red.service`.

```
$ sudo nano /etc/systemd/system/node-red.service
```

Copy and paste in the following, then save and close the file.

```
/etc/systemd/system/node-red.service
```

```
[Unit]
```

```
Description=Node-RED
```

```
After=syslog.target network.target
```

```
[Service]
```

```
ExecStart=/usr/local/bin/node-red-pi --max-old-space-size=128 -v
```

```
Restart=on-failure
```

```
KillSignal=SIGINT

# log output to syslog as 'node-red'
SyslogIdentifier=node-red
StandardOutput=syslog

# non-root user to run as
WorkingDirectory=/home/sammy/
User=sammy
Group=sammy

[Install]
WantedBy=multi-user.target
```

A full explanation of systemd service files is beyond this tutorial, but you can learn more by reading [Systemd Essentials: Working with Services, Units, and the Journal](#).

That said, let's break down some of the sections in our service file:

```
/etc/systemd/system/node-red.service
```

```
[Unit]
Description=Node-RED
After=syslog.target network.target
```

This describes our service and indicates that it should be started after networking and syslog are functioning.

```
/etc/systemd/system/node-red.service
```

```
[Service]
ExecStart=/usr/local/bin/node-red-pi --max-old-space-size=128 -v
Restart=on-failure
KillSignal=SIGINT
```

`ExecStart` is the command needed to start our service. We call `node-red-pi` instead of plain `node-red` so we can pass some memory-saving options to Node.js. This should allow it to run well on any reasonably sized server, depending of course on how many flows you create in Node-RED (and how complicated they are). `Restart=on-failure` means systemd will try to restart Node-RED if it crashes, and `KillSignal` tells systemd the best way to quit Node-RED when it needs to shut down or restart the process.

```
/etc/systemd/system/node-red.service
```

```
# log output to syslog as 'node-red'  
SyslogIdentifier=node-red  
StandardOutput=syslog
```

This sets the label used when logging, and logs all output to the syslog service.

```
/etc/systemd/system/node-red.service
```

```
# non-root user to run as  
WorkingDirectory=/home/sammy/  
User=sammy  
Group=sammy
```

We want to run Node-RED as our non-root user. The lines above tell systemd to launch Node-RED using our user and group, and from within our home directory.

```
/etc/systemd/system/node-red.service
```

```
[Install]  
WantedBy=multi-user.target
```

WantedBy indicates the targets our service should run under. In this case, when Ubuntu boots into multi-user mode, it will know to also launch our Node-RED service. Multi-user mode is the default startup target.

Now that our service file is installed and understood, we need to enable it. This will enable it to execute on startup.

```
$ sudo systemctl enable node-red
```

Let's manually start the service now to test that it's still working.

```
$ sudo systemctl start node-red
```

Point a browser back at the server's port 1880 and verify that Node-RED is back up. If it is, shut it back down until we secure the install in the next step.

```
$ sudo systemctl stop node-red
```

## Step 4 — Setting Up Nginx

We're going to use Nginx to *proxy* the Node-RED service. This means Nginx will handle all of the SSL connections on port 443 (using the Let's Encrypt certificates you previously set up), and then pass the traffic along to Node-RED.

Open a new Nginx configuration for the site.

```
$ sudo nano /etc/nginx/sites-enabled/node-red.example.com
```

Copy and paste the following, changing the server name and certificate paths:

```
/etc/nginx/sites-enabled/node-red.example.com

server {
    listen 80;
    listen 443 ssl http2;
    server_name node-red.example.com;
    ssl_certificate /etc/letsencrypt/live/node-red.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/node-red.example.com/privkey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3DES;
    ssl_prefer_server_ciphers On;
    ssl_session_cache shared:SSL:128m;
    ssl_stapling on;
    ssl_stapling_verify on;
    resolver 8.8.8.8;

    location / {
        if ($scheme = http) {
            return 301 https://$server_name$request_uri;
        }
        proxy_pass http://localhost:1880;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location '/.well-known/acme-challenge' {
```

```
}  
}
```

Save and close the file. Let's explain what this file does.

The first three lines tell Nginx what ports to listen on, and what domain name to respond to. The `ssl_certificate` and `ssl_certificate_key` lines point to the certificates we retrieved from Let's Encrypt. The remaining `ssl_` lines choose protocols, ciphers, and options that are more secure than the defaults.

`location /` starts the block where we actually define our Node-RED proxy.

```
                /etc/nginx/sites-enabled/node-red.example.com  
  
if ($scheme = http) {  
    return 301 https://$server_name$request_uri;  
}
```

This block will match any plain, non-secure http connections, and redirect them to the https version of the site.

```
                /etc/nginx/sites-enabled/node-red.example.com  
  
proxy_pass http://localhost:1880;
```

We point to our Node-RED service here. It is available on `localhost`, at port `1880`, so we pass connections to it there. The remainder of this block of configuration sets some headers that are important for proper proxy functioning. The `Upgrade` and `Connection` headers are especially important for handling Node-RED's websocket connections.

Finally, we have a block to make sure the Let's Encrypt challenge responses continue to be fetched from Nginx's default web root:

```
                /etc/nginx/sites-enabled/node-red.example.com  
  
location '/.well-known/acme-challenge' {  
    root /var/www/html;  
}
```

Reload Nginx to pick up the new configuration.



```
$ sudo systemctl reload nginx
```

Finally, start Node-RED again.

```
$ sudo systemctl start node-red
```

Once again, navigate to your server: <http://node-red.example.com>. You should be redirected to <https://node-red.example.com> (note the `https`) and see the Node-RED admin interface. This means we're now proxying Node-RED through Nginx. We just have a few more tweaks to lock down Node-RED, and then we'll be finished.

## Step 5 — Securing Node-RED and Wrapping Up

Now that our connection is secure, let's add a password to the Node-RED admin. Instead of putting a bare password right into our settings file, we first make a one-way cryptographic hash of it, and use that instead. We'll use `node-red-admin` to create the hash:

```
$ node-red-admin hash-pw
```

You will be prompted for a password. Type it in, press `ENTER`, and a hash will be printed on screen. Copy that to your clipboard and open the Node-RED settings file.

```
$ nano ~/.node-red/settings.js
```

Scroll down and uncomment the `adminAuth` block (by removing the `///` in front of each line). Change `username` to whatever you like, and paste the hash into the `password` field.

settings.js

```
adminAuth: {  
  type: "credentials",  
  users: [{  
    username: "admin",  
    password: "$2a$08$Ab9prIr1M8a5a1/Zx8.B9.uIOCpe.v90ZGuZc2kAATp6BHJ/WV5KS",  
    permissions: "*"   
  }]  
},
```

While we've got the file open, uncomment the `uihost` line as well by removing the `//` at the front of the line.

```
settings.js
```

```
uiHost: "127.0.0.1",
```

This means Node-RED will only listen on the local interface, and won't be reachable directly by the outside world (it will only be accessed through the Nginx proxy). You can now save and close the file.

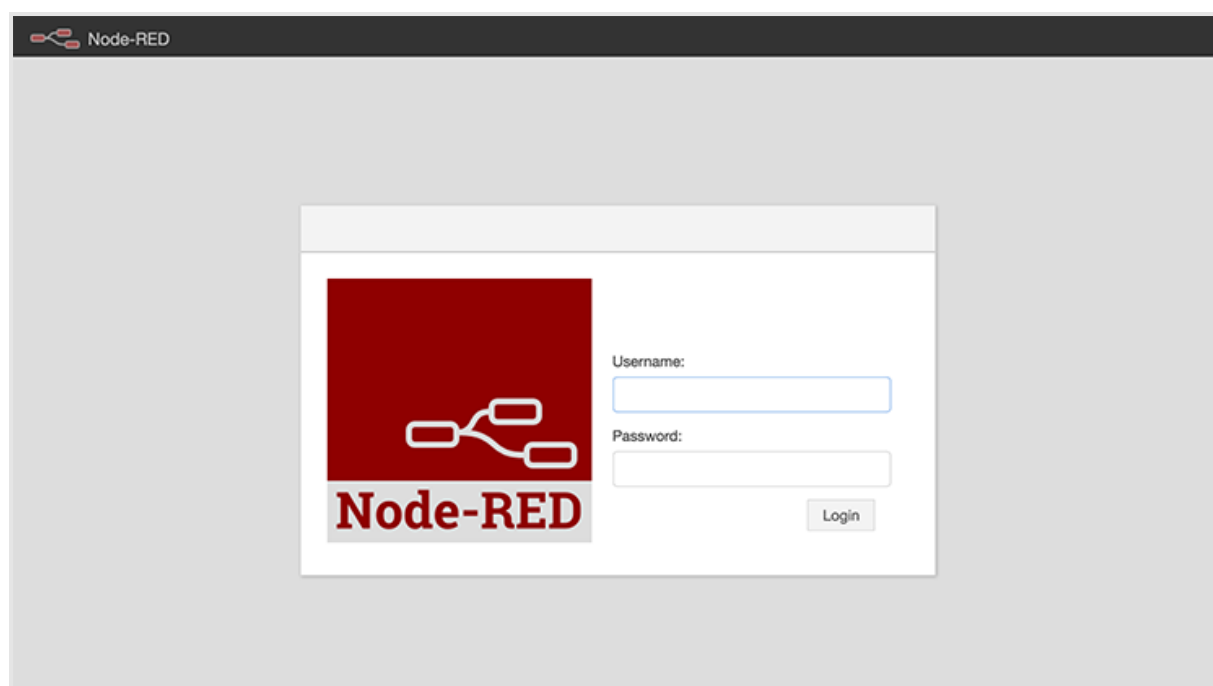
Update the firewall one last time, just to make sure Node-RED is never directly accessible.

```
$ sudo ufw deny 1880
```

Finally, restart Node-RED.

```
$ sudo systemctl restart node-red
```

Navigate to <https://node-red.example.com> and you'll see a login screen instead of the main editing interface.



If your site is showing a login screen and an `https` connection, you've set everything up correctly.

## Conclusion

We now have a reasonably secure installation of Node-RED, being proxied by Nginx using Let's Encrypt for its SSL certificates. Log in and get wiring! There is much more information and project inspiration available at [Node-RED's website](#).

By: Brian Boucheron

♥ Upvote (11)

✚ Subscribe

🔗 Share



Editor:  
Hazel Virdó

---



We just made it easier for you to deploy faster.

[TRY FREE](#)

---

### Related Tutorials

How to Set Up a Remote Desktop with X2Go on Debian 8

How To Find Broken Links on Your Website Using Wget on Debian 7

How To Set Up SETI@home on Ubuntu 14.04 or Debian 7

How To Set Up an OSRM Server on Ubuntu 14.04

## 14 Comments

Leave a comment...

Log In to Comment

^ javicho21 October 9, 2016



0 The installation hangs when I go to install node-red. After I execute the node-red install it stays here and won't proceed.

```
$ sudo npm install -g --unsafe-perm node-red node-red-admin
npm WARN deprecated i18next-client@1.10.3: you can use npm install i18next from
version 2.0.0
npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie
https://nodesecurity.io/advisories/130
/usr/local/bin/node-red -> /usr/local/lib/nodemodules/node-red/red.js
/usr/local/bin/node-red-pi -> /usr/local/lib/nodemodules/node-red/bin/node-red-pi
/usr/local/bin/node-red-admin -> /usr/local/lib/node_modules/node-red-admin/node-red-
admin.js
```

```
bcrypt@0.8.7 install /usr/local/lib/nodemodules/node-red-
admin/nodemodules/bcrypt
node-gyp rebuild
```

```
make: Entering directory '/usr/local/lib/nodemodules/node-red-
admin/nodemodules/bcrypt/build'
CXX(target) Release/obj.target/bcrypt/lib/src/blowfish.o
```

CXX(target) Release/obj.target/bcryptlib/src/bcrypt.o  
CXX(target) Release/obj.target/bcryptlib/src/bcryptnode.o

---

^ digitalocean06a November 1, 2016

- 0 @javicho21 , did you apt-get update before installing node-red?  
Everything went fine for me but at the end the POST of admin username password simply timeout.  
Any idea?

---

^ digitalocean06a November 1, 2016

- 0 I mixed apt-get and npm insall in my previous post :)

---

^ gfmallmann December 12, 2016

- 0 I am having issues installing node red packages using nmp.  
I tried both types, as described here: <https://nodered.org/docs/getting-started/adding-nodes>

```
cd $HOME/.node-red  
npm install <npm-package-name>
```

```
sudo npm install -g <npm-package-name>
```

what am i doing wrong?

Thank you,

---

^ beardicus December 12, 2016

- 0
- Did you restart Node-RED after installing the packages?
  - What output is **npm** giving you... are there any errors? Just wondering if it's successfully installing but not being recognized by Node-RED, or something else.
  - What package are you trying to install, specifically?

---

^ gfmallmann December 12, 2016



◦ I tried to install dashboard.  
now it shows up, that it is already installed.  
sudo npm install -g dashboard  
/usr/local/lib  
└─ dashboard@0.0.1

after installing I restarted node-red, by calling node-red at the console again

---

^ [beardicus](#) December 12, 2016

◦ The **dashboard** package is not a Node-RED package, and seems abandoned. You've installed it properly, but because it has no Node-RED-specific integrations, you're not going to see anything in the Node-RED interface.

You probably want the **node-red-dashboard** package. Generally, any Node-RED-compatible package will begin with **node-red-** or **node-red-contrib-**.

Good luck!

---

^ [gfmallmann](#) December 12, 2016

◦ I see... I did not know the different until now.  
thank you very much for the extremely quick help!

---

^ [ryanmin](#) February 2, 2017

◦ Great tutorial! Thank You.

Only one issue... Node-RED is rejecting my login attempts. After a bit of investigation, if I unblock the 1880 port and connect via http and not https I can login perfectly. It seems that when I try connecting on https Node-Red is unable to issue a token.

Browser output excerpt:

1. Request URL: [https://example.com/settings?\\_=1485942900514](https://example.com/settings?_=1485942900514)
2. Request Method: GET
3. Status Code: 401 I have been unsuccessful in resolving this. Your help is appreciated.

Thanks

---

^ [jpfranca](#) February 25, 2017

0 After I run step 4, nginx stops loading.

```
-- Unit nginx.service has begun starting up.  
Feb 25 00:46:57 nodered nginx[753]: nginx: [emerg] the size 134217728 of s  
Feb 25 00:46:57 nodered nginx[753]: nginx: configuration file /etc/nginx/n  
Feb 25 00:46:57 nodered systemd[1]: nginx.service: Control process exited,  
Feb 25 00:46:57 nodered systemd[1]: Failed to start A high performance web
```



Please help

---

^ [henroritchie](#) November 22, 2017

0 The instructions to launch Node-RED on startup failed in my case. Node-RED was not installed to /usr/local/bin/node-red-pi as stated node-red.service file. To obtain the directory where Node-RED is installed use `$which node-red`. Secondly, the command node-red-pi specifically relates to devices with constrained memory like the Raspberry Pi and BeagleBone.

The instructions [here](#) have been a great help in my case.

---

^ [michaelhertig](#) December 8, 2017

0 Thanks for this tutorial, i followed it and successfully set up a node-red server. However, in the `node-red.service` file, i had to use the line

```
ExecStart=/usr/bin/node-red-pi --max-old-space-size=128 -v
```

instead of

```
ExecStart=/usr/local/bin/node-red-pi --max-old-space-size=128 -v
```

---

^ [bgirardot](#) February 3, 2018



0 I ran into a small problem when asked to initially test by going to the fully qualified host and domain name and port 1880.

I was installing via ssh of course and of course nginx was not set up to listen on that port so it failed. I just skipped down and finished the nginx configs and everything else worked exactly as detailed above.

Thank you for the great tutorial!

---

^ [Demothi](#) October 18, 2018



0 Installed it all with no errors. When launching Node-Red it states that

```
18 Oct 17:35:21 - [info]
```

```
Welcome to Node-RED
```

```
=====
```

```
18 Oct 17:35:21 - [info] Node-RED version: v0.19.4
```

```
18 Oct 17:35:21 - [info] Node.js version: v9.11.2
```

```
18 Oct 17:35:21 - [info] Linux 4.9.0-7-amd64 x64 LE
```

```
18 Oct 17:35:21 - [info] Loading palette nodes
```

```
18 Oct 17:35:21 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
```

```
18 Oct 17:35:21 - [warn] rpi-gpio : Cannot find Pi RPi.GPIO python library
```

```
18 Oct 17:35:22 - [info] Settings file : /root/.node-red/settings.js
```

```
18 Oct 17:35:22 - [info] Context store : 'default' [module=memory]
```

```
18 Oct 17:35:22 - [info] User directory : /root/.node-red
```

```
18 Oct 17:35:22 - [warn] Projects disabled : editorTheme.projects.enabled=false
```

```
18 Oct 17:35:22 - [info] Flows file : /root/.node-red/flows_project.json
```

```
18 Oct 17:35:22 - [info] Creating new flow file
```

```
18 Oct 17:35:22 - [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.
```

```
-----
```



```
18 Oct 17:35:22 - [info] Starting flows
18 Oct 17:35:22 - [info] Started flows
18 Oct 17:35:22 - [info] Server now running at http://127.0.0.1:1880/
```

How do I move the installation from localhost to my server's IP or hosted domain?



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

---

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

---

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#)  
[Write for DOnations](#) [Shop](#)