# Creating Node-Red Service

## 1. Install Node.js

Download the latest 14.x LTS version of Node.js from the official Node.js home page. It will offer you the best version for your system.

Run the downloaded MSI file. Installing Node.js requires local administrator rights; if you are not a local administrator, you will be prompted for an administrator password on install. Accept the defaults when installing. After installation completes, close any open command prompts and re-open to ensure new environment variables are picked up.

Once installed, open a command prompt and run the following command to ensure Node.js and npm are installed correctly.

Using Powershell: `node --version; npm --version`

Using cmd: `node --version && npm --version`

You should receive back output that looks similar to:

```
v14.17.2.0
6.14.13
```

## 2. Install Node-RED

Installing Node-RED as a global module adds the command `node-red` to your system path. Execute the following at the command prompt:

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

## 3. Run Node-RED

Once installed, you are ready to run Node-RED.

## Alternative Installations on Windows

In this section, we provide you with information on alternative ways to install Node.js, npm and the Windows Build Tools needed to install some Nodes for Node-RED on Windows.

> *Note* : You should *not* use an administrative (a.k.a. "elevated") command prompt unless specifically instructed to. You will very likely need to be quite familiar with command prompts as you learn about Node-RED and Node.js and it will be worth while reading some of the Microsoft articles on PowerShell. the PowerShell Tutorial and PowerShell One-Liners sites may also be helpful.

Standard installations of Node.js on Windows require local administrator rights. Download the appropriate version from the official Node.js home page. It will offer you the best version. While you can use either 32 bit or 64 bit versions on 64 bit Windows, it is recommended to use the 64bit version of Node. If for some reason, you need a different installation, you can use the Downloads Page.

There are two potentially useful alternatives to installing Node.js with the MSI installer.

1. Using the Chocolatey package manager

   Chocolatey is a package manager for Windows similar to APT or yum on Linux and brew on the Macintosh platforms. If you are already using Chocolatey, you may want to use this tool to install Node.js (e.g. using the `nodejs-lts` package). Note however, that many packages have uncertain management and that these packages may use different folder locations than those mentioned above.

2. Using a Node version manager

   Using a Node.js version manager such as nvm-windows can be very helpful if you are doing Node.js development and need to test against different versions. Keep in mind that you will need to reinstall global packages and may need to re-install local packages when when you switch the version of Node you are using.

*Note* : Microsoft maintain a parallel version of Node that uses the Microsoft Chakra Core JavaScript engine instead of V8. This is not recommended for Node-RED as it has not been tested.

## *npm on Windows*

When you install Node.js, you are also installing the npm package manager. You may see some instructions on the web that recommend installing later releases of npm than the one that comes with the Node.js release. This is **not** recommended as it is too easy to later end up with an incompatible version. Node.js releases are very regular and that is sufficient to keep npm updated.

## *Sharing Node-RED between Users*

Node.js is installed into the `Program Files` folder as you would expect. However, if you install a global *package* like Node-RED using `npm -g`, it is installed into the `$env:APPDATA\npm` folder (`%APPDATA%\npm` using cmd) for the **current** user. This is less than helpful if you are installing on a PC with multiple user logins or on a server and installing using an admin login rather than the login of the user that will run Node applications like Node-RED.

*Note* : To see what a folder name like `%APPDATA%` translates to, you can simply type it into the address bar of the Windows File Explorer. Alternatively, in PowerShell, type the command `cd $Env:APPDATA`(`cd %APPDATA%` using cmd).

To fix this, either give permissions to the folder to other users and make sure that the folder is included in their `path` user environment variable.

Alternatively, change the global file location to somewhere accessible by other users. Make sure that you use the user that will be running Node-RED to make these changes. For example, to change the location to `$env:ALLUSERSPROFILE\npmglobal` using PowerShell:

```
mkdir $env:ALLUSERSPROFILE\npmglobal
npm config set prefix $env:ALLUSERSPROFILE\npmglobal
```

You will then want to change the npm cache folder as well:

```
mkdir $env:ALLUSERSPROFILE\npmglobal-cache
npm config set cache $env:ALLUSERSPROFILE\npmglobal-cache --global
```

If using the above changes, you can add the new *prefix* folder to the *PATH* System variable and remove the old folder from the user's Path variable. To change the PATH Environment variable, type `environment` into the start menu or Cortana and choose *Edit Environment Variables*.

For each of the users running Node-RED, check that the above settings for the other users are correct.

### *Installing Node.js Windows Build Tools*

Many Node.js modules used by Node-RED or installed nodes have binary components that will need compiling before they will work on Windows. To enable npm to compile binaries on the Windows platform, install the windows-build-tools module using the command prompt as an Administrator:

```
npm install --global --production windows-build-tools
```

If you wish to have the built-in Python v2.7 install exposed for use, use the command:

```
npm install --global --production --add-python-to-path windows-build-tools
```

*Notes*:

- Not all Node.js modules will work under Windows, check the install output carefully for any errors.
- During the install some errors may be reported by the `node-gyp` command. These are typically *non-fatal* errors and are related to optional dependencies that require a compiler in order to build them. **Node-RED will work without these optional dependencies**. If you get fatal errors, first check that you installed the `windows-build-tools` module and that you have closed and opened your command prompt window.

## Running on Windows

Once installed, the simple way to run Node-RED is to use the `node-red` command in a command prompt: If you have installed Node-RED as a global npm package, you can use the node-red command:

```
C:>node-red
```

This will output the Node-RED log to the terminal. You must keep the terminal open in order to keep Node-RED running.

Note that running Node-RED will create a new folder in your `%HOMEPATH%` folder called `.node-red`. This is your `userDir` folder, think of it as the home folder for Node-RED configuration for the current user. You will often see this referred to as `~/.node-red` in documentation. `~` is shorthand for the user home folder on Unix-like systems. You can use the same reference if using PowerShell as your command line as recommended. If you are using the older `cmd` shell, that won't work.

You can now create your first flow.

## *Using PM2*

If you are using Windows to develop Node-RED flows or nodes, you may find it helpful to use PM2 to run Node-RED. This can be configured to automatically restart when files change, always keep Node-RED running and manage log output.

## *Run Node-RED on Startup*

If you want to use Windows as a production platform for Node-RED, you will want to have a Windows Task Scheduler job set up. To do so:

1. Go to the start menu and type "task scheduler" and click on the result.
2. Click on "Create Task..." in the right-hand menu. Follow the steps to create a new task.

Make sure that you use the user login that you've used to set up and do the initial run of Node-RED. You can use an "At startup" trigger to always run Node-RED at system startup. Use the Action "Start a program" with details set to `C:\Users\<user>\AppData\Roaming\npm\node-red.cmd` (replacing `<user>` with your actual user name).

You may wish to make sure that it only starts if the network is available. You may also wish to restart if the job fails. Perhaps restarting every minute but only 3 times - if it won't start by then, the error is fatal and will need some other intervention. You can check for failures by looking in the event log. If you want to access to the logs when running this way, you should amend the node-red.cmd file to redirect std and error outputs to a file (creating an alternative startup file would be better so that it isn't overwritten on updates).