How to install Node.js applications, if you're not a Node.js developer

> This article was originally published at https://gist.github.com/joepie91/24f4e70174d10325a9af743a381d5ec6.

While installing a Node.js application isn't difficult *in principle*, it may still be confusing if you're not used to how the Node.js ecosystem works. This post will tell you how to get the application going, what to expect, and what to do if it doesn't work.

Occasionally an application may have custom installation steps, such as installing special system-wide dependencies; in those cases, you'll want to have a look at the install documentation of the application itself as well. However, *most of the time* it's safe to assume that the instructions below will work fine.

If the application you want to install is available in your distribution's repositories, then install it through there instead and skip this entire guide; your distribution's package manager will take care of all the dependencies.

Checklist

Before installing a Node.js application, check the following things:

1. **You're running a maintained version of Node.js.** You can find a list of current maintained versions here. For minimal upgrade headaches, ensure that you're running an LTS version. If your system is running an *unsupported* version, you should install Node.js from the Nodesource repositories instead.
2. **Your version of Node.js is a standard one.** In particular Debian and some Debian-based distributions have a habit of modifying the way Node.js works, leading to a lot of things breaking. Try running `node --version` - if that works, you're running a standard-enough version. If you can only do `nodejs --version`, you should install Node.js from the Nodesource repositories instead.

3. **You have build tools installed.** In particular, you'll want to make sure that `make`, `pkgconfig`, GCC and Python exist on your system. If you don't have build tools or you're unsure, you'll want to install a package like `build-essential` (on Linux) or look here for further instructions (on other platforms, or unusual Linux distributions).
4. ***npm* works.** Run `npm --version` to check this. If the `npm` command doesn't exist, your distribution is probably shipping a weird non-standard version of Node.js; use the Nodesource repositories instead. **Do not** install npm as a separate package, this will lead to headaches down the road.

No root/administrator access, no repositories exist for your distro, can't change your system-wide Node.js version, need a really specific Node.js version to make the application work, or have some other sort of edge case? Then nvm can be a useful solution, although keep in mind that it *will not* automatically update your Node.js installation.

How packages work in Node.js

Packages work a little differently in Node.js from most languages and distributions. In particular, **dependencies are *not* installed system-wide**. Every project has its own (nested) set of dependencies. This solves a lot of package management problems, but it can take a little getting used to if you're used to other systems.

In practice, this means that you should almost always do a regular `npm install` - that is, installing the dependencies locally into the project. The only time you need to do a 'global installation' (using `npm install -g packagename`) is when you're installing an *application* that is *itself* published on npm, and you want it to be available globally on your system.

This also means that **you should *not* run npm as root** by default. This is a really important thing to internalize, or you'll run into trouble down the line.

To recap:

- Run npm under your own, unprivileged user - unless instructions *specifically* state that you should run it as root.
- Run npm in 'local' mode, installing dependencies into the project folder - unless instructions *specifically* state that you should do a global installation.

If you're curious about the details of packages in Node.js, here is a developer-focused article about them.

Installing an application from the npm registry

Is the application published on the npm registry, ie. does it have a page on `npmjs.org`? Great! That means that installation is a single command.

If you've installed Node.js through your distribution's package manager: `sudo npm install -g packagename`, where `packagename` is the name of the package on npm.

If you've installed Node.js through `nvm` or a similar tool: `npm install -g packagename`, where `packagename` is the name of the package on npm.

You'll notice that you need to run the command as root (eg. through `sudo`) when installing Node.js through your distribution's package manager, but not when installing it through `nvm`.

This is because by default, Node.js will use a system-wide folder for globally installed packages; but under `nvm`, your entire Node.js installation exists in a subdirectory of your unprivileged user's home directory - including the 'global packages' folder.

After following these steps, some new binaries will probably be available for you to use system-wide. If the application's documentation doesn't tell you what binaries are available, then you should find its code repository, and look at the `"bin"` key in its `package.json`; that will contain a list of all the binaries it provides. Running them with `--help` will probably give you documentation.

You're done!

> **If you run into a problem:** Scroll down to the 'troubleshooting' section.

Installing an application from a repository

Some applications are not published to the npm registry, and instead you're expected to install it from the code (eg. Git) repository. In those cases, start by looking at the application's install instructions to see if there are special requirements for cloning the repository, like eg. checking out submodules.

If there are no special instructions, then a simple `git clone http://example.com/path/to/repository` should work, replacing the URL with the cloning URL of the repository.

Making it available globally (like when installing from the npm registry)

Enter the cloned folder, and then run:

- If you installed Node.js from your distribution's repositories: `sudo npm install -g`, with no other arguments.
- If you installed Node.js through `nvm` or a similar tool: `npm install -g`, with no other arguments.

You're done!

> **If you run into a problem:** Scroll down to the 'troubleshooting' section.

Keeping it in the repository

Sometimes you don't want to really install the application onto your system, but you rather just want to get it running locally from the repository.

In that case, enter the cloned folder, and run: `npm install`, with no other arguments.

You're done!

> **If you run into a problem:** Scroll down to the 'troubleshooting' section.

Troubleshooting

Sometimes, things still won't work. In most cases it'll be a matter of missing some sort of undocumented external dependency, ie. a dependency that npm can't manage for you and that's typically provided by the OS. Sometimes it's a version compatibility issue. Occasionally applications are just outright broken.

When running into trouble with npm, try entering your installation output into this tool first. It's able to (fully automatically!) recognize the most common issues that people tend to run into with npm.

If the tool can't find your issue and it still doesn't work, then drop by the IRC channel (#Node.js on Libera, an online chat can be found here) and we'll be happy to help you get things going! You do need to register your username to talk in the channel; you can get help with this in the #libera channel.

Revision #1
Created 11 December 2024 16:42:32 by joepie91
Updated 11 December 2024 16:50:35 by joepie91