How To Install Node.js on Ubuntu 16.04

Updated November 1, 2016 478k NODE.JS UBUNTU 16.04

Introduction

Node.js is a JavaScript platform for general-purpose programming that allows users to build network applications quickly. By leveraging JavaScript on both the front-end and the back-end, development can be more consistent and be designed within the same system.

In this guide, we'll show you how to get started with Node.js on an Ubuntu 16.04 server.

If you are looking to set up a production Node.js environment, check out this link: How To Set Up a Node.js Application for Production.

Prerequisites

This guide assumes that you are using Ubuntu 16.04. Before you begin, you should have a non-root user account with `sudo` privileges set up on your system. You can learn how to do this by completing steps 1-4 in the initial server setup for Ubuntu 16.04.

How To Install the Distro-Stable Version for Ubuntu

Ubuntu 16.04 contains a version of Node.js in its default repositories that can be used to easily provide a consistent experience across multiple systems. At the time of writing, the version in the repositories is v4.2.6. This will not be the latest version, but it should be quite stable, and should be sufficient for quick experimentation with the language.

In order to get this version, we just have to use the `apt` package manager. We should refresh our local package index first, and then install from the repositories:

```
sudo apt-get update
sudo apt-get install nodejs
```

If the package in the repositories suits your needs, this is all that you need to do to get set up with Node.js. In most cases, you'll also want to also install `npm`, which is the Node.js package manager. You can do this by typing:

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

This will allow you to easily install modules and packages to use with Node.js.

Because of a conflict with another package, the executable from the Ubuntu repositories is called `nodejs` instead of `node`. Keep this in mind as you are running software.

Next, we'll discuss more flexible and robust methods of installation.

How To Install Using a PPA

An alternative that can get you a more recent version of Node.js is to add a PPA (personal package archive) maintained by NodeSource. This will have more up-to-date versions of Node.js than the official Ubuntu repositories, and allows you to choose between Node.js v4.x (the older long-term support version, supported until April of 2017), v6.x (the more recent LTS version, which will be supported until April of 2018), and Node.js v7.x (the current actively developed version).

First, you need to install the PPA in order to get access to its contents. Make sure you're in your home directory, and use `curl` to retrieve the installation script for your preferred version, making sure to replace `6.x` with the correct version string:

```
cd ~
curl -sL https://deb.nodesource.com/setup_6.x -o nodesource_setup.sh
```

You can inspect the contents of this script with `nano` (or your preferred text editor):

```
nano nodesource_setup.sh
```

And run the script under `sudo`:

```
sudo bash nodesource_setup.sh
```

The PPA will be added to your configuration and your local package cache will be updated automatically. After running the setup script from nodesource, you can install the Node.js package in the same way that you did above:

```
• sudo apt-get install nodejs
```

The `nodejs` package contains the `nodejs` binary as well as `npm`, so you don't need to install `npm` separately. However, in order for some `npm` packages to work (such as those that require compiling code from source), you will need to install the `build-essential` package:

```
• sudo apt-get install build-essential
```

How To Install Using NVM

An alternative to installing Node.js through `apt` is to use a specially designed tool called `nvm`, which stands for "Node.js version manager".

Using `nvm`, you can install multiple, self-contained versions of Node.js which will allow you to control your environment easier. It will give you on-demand access to the newest versions of Node.js, but will also allow you to target previous releases that your app may depend on.

To start off, we'll need to get the software packages from our Ubuntu repositories that will allow us to build source packages. The nvm script will leverage these tools to build the necessary components:

```
• sudo apt-get update
• sudo apt-get install build-essential libssl-dev
```

Once the prerequisite packages are installed, you can pull down the nvm installation script from the project's GitHub page. The version number may be different, but in general, you can download it with `curl`:

```
• curl -sL https://raw.githubusercontent.com/creationix/nvm/v0.31.0/install.sh -o install_r
```

And inspect the installation script with `nano`:

```
• nano install_nvm.sh
```

Run the script with `bash`:

```
• bash install_nvm.sh
```

It will install the software into a subdirectory of your home directory at `~/.nvm`. It will also add the necessary lines to your `~/.profile` file to use the file.

To gain access to the nvm functionality, you'll need to log out and log back in again, or you can source the `~/.profile` file so that your current session knows about the changes:

```
• source ~/.profile
```

Now that you have nvm installed, you can install isolated Node.js versions.

To find out the versions of Node.js that are available for installation, you can type:

```
• nvm ls-remote
```

```
Output
...
         v5.8.0
         v5.9.0
         v5.9.1
        v5.10.0
        v5.10.1
        v5.11.0
         v6.0.0
```

As you can see, the newest version at the time of this writing is v6.0.0. You can install that by typing:

typing:

```
• nvm install 6.0.0
```

Usually, nvm will switch to use the most recently installed version. You can explicitly tell nvm to use the version we just downloaded by typing:

```
• nvm use 6.0.0
```

When you install Node.js using nvm, the executable is called `node`. You can see the version currently being used by the shell by typing:

```
• node -v
```

```
Output
v6.0.0
```

If you have multiple Node.js versions, you can see what is installed by typing:

```
• nvm ls
```

If you wish to default one of the versions, you can type:

```
• nvm alias default 6.0.0
```

This version will be automatically selected when a new session spawns. You can also reference it by the alias like this:

```
• nvm use default
```

Each version of Node.js will keep track of its own packages and has `npm` available to manage these.

You can have `npm` install packages to the Node.js project's `./node_modules` directory by using the normal format. For example, for the `express` module:

```
npm install express
```

If you'd like to install it globally (making it available to the other projects using the same Node.js version), you can add the `-g` flag:

```
npm install -g express
```

This will install the package in:

```
~/.nvm/node_version/lib/node_modules/package_name
```

Installing globally will let you run the commands from the command line, but you'll have to link the package into your local sphere to require it from within a program:

```
npm link express
```

You can learn more about the options available to you with nvm by typing:

```
nvm help
```

Conclusion

As you can see, there are a quite a few ways to get up and running with Node.js on your Ubuntu 16.04 server. Your circumstances will dictate which of the above methods is the best idea for your circumstance. While the packaged version in Ubuntu's repository is the easiest, the `nvm` method is definitely much more flexible.