Install Node.js Locally with Node Version Manager (nvm)

Using `nvm` (Node.js Version Manager) makes it easier to install and manage multiple versions of Node.js on a single local environment. Even if you only need a single version of Node.js right now, we still recommend using nvm because it allows you to switch between different versions of Node (depending on the requirements of your project) with minimal hassle.

In this tutorial, we'll walk through:

- How to download and install the Node Node Version Manager (nvm) shell script
- Using `nvm` to install the latest LTS version of Node.js
- Switching to a different Node.js version with `nvm`

By the end of this tutorial, you should be able to install the `nvm` command and use it to manage different versions of Node.js on a single environment.

Goal

Install and manage a local installation of `node` using `nvm`.

Prerequisites

- Overview: Manage Node.js Locally

Watch

Install nvm

NVM stands for Node.js Version Manager. The `nvm` command is a POSIX-compliant bash script that makes it easier to manage multiple Node.js versions on a single environment. To use it, you need to first install the bash script, and add it to your shell's `$PATH`.

Learn more about why we recommend using NVM in Overview: Manage Node.js Locally.

Note: We do not recommend using nvm to install Node.js for production environments. If you're installing Node.js on your production environment you should consider using your OS's package manager, or your server tooling of choice, to install and lock the environment to a specific version of Node.js.

The official documentation for how to install nvm, and some common trouble shooting tips, is in the project's README.

Windows users: The process for installing nvm on Windows is different than what's shown below. If you're using Windows check out this Windows-specific version of nvm.

The basic process is as follows:

Download the install script

Using curl, or wget, download the installation script. In the URL below make sure you replace `v0.35.0` with the latest version of nvm.

```
://raw.githubusercontent.com/nvm-sh/nvm/v0.35.0/install.sh
```

It's not a bad idea to open the install script and inspect its contents given that you just downloaded it from the Internet.

Run the install script

Run the install script with `bash`.

```
bash install_nvm.sh
```

This script clones the nvm repository into *~/.nvm*. Then it updates your profile (*~/.bash_profile*, *~/.zshrc*, *~/.profile*, or *~/.bashrc*) to source the *nvm.sh* it contains.

You can confirm that your profile is updated by looking at the install script's output to determine which file it used. Look for something like the following in that file:

```
. "$NVM_DIR/nvm.sh"  # This loads nvm
n" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash
```

Restart your terminal

In order to pick up the changes to your profile either close and reopen the terminal, or manually source your respective *~/.profile*.

Example:

```
source ~/.bash_profile
```

Verify it worked

Finally, you can verify that it's installed with the `command` command:

```
command -v nvm
```

Should return `nvm`. Note: You can't use the `which` command with `nvm` since it's a shell function and not an actual application.

See what it does

Finally, run the `nvm` command to get a list of all the available sub-commands and to further verify that installation worked.

Use nvm to install the latest LTS release of Node.js

Now that you've got nvm installed let's use it to install, and use, the current LTS version of Node.js.

```
nvm install --lts
```

```
# Output
Installing latest LTS version.
Downloading and installing node v10.16.3...
Downloading https://nodejs.org/dist/v10.16.3/node-v10.16.3
######################################################################
Computing checksum with sha256sum
Checksums matched!
Now using node v10.16.3 (npm v6.9.0)
Creating default alias: default -> lts/* (-> v10.16.3)
```

Verify it worked, and that the version is correct:

```
node --version
# => v10.16.3
which node
# => /Users/joe/.nvm/versions/node/v10.16.3/bin/node
```

Note this line `Creating default alias: default -> lts/* (-> v10.16.3)` . This indicates that nvm has set `lts/*` as the default alias. Practically this means that anytime you start a new shell, and the *nvm.sh* script is sourced, it will default that shell to using the installed lts release. You can change this behavior using the `nvm alias` command.

Example to set the default version of node to use when starting a new shell to 10.0.0:

```
nvm alias default 10.0.0
```

Use nvm to install other versions of Node.js

The real benefit of nvm comes when you install different versions of Node.js. You can then switch between them depending on which project you're working on.

List available versions

To see the entire list of Node.js versions available to install, enter the following:

```
nvm ls-remote
```

Install a specific version

Install a specific version:

```
nvm install 8.16.2
```

Install the latest release:

```
nvm install node
```

Install an older LTS release by codename:

```
nvm install carbon
# => Installs v8.16.2 the latest release of the Carbon LTS
```

List installed versions

You can see which versions of Node.js you have installed already, and therefore which ones you can use with the `nvm ls` command:

```
nvm ls
```

This will output a list of installed versions, as well as indicate which version is currently being used by the active shell.

Switch to another version

To switch to another version for the active shell use `nvm use`.

For a specific version provide a version number:

```
nvm use 10.16.3
# => Now using node v10.16.3 (npm v6.9.0)
```

Switch to the latest installed version:

```
nvm use node
```

Use the latest LTS version:

```
nvm use --lts
```

Tip: Use `nvm alias default {VERSION}` to switch the version of Node.js used by default when starting a new shell.

Recap

In this tutorial we walked through installing the `nvm` bash script and making sure it works. Then we used `nvm` to install the latest LTS release of Node.js and set it as our environment's default Node.js version. Then we looked at

how you can use `nvm` to install any number of Node.js versions and switch between them as needed.

You should now be all set to execute and work on your your Node.js project(s) no matter which version of Node.js they are written for.

Further your understanding

- Run the `nvm` command with no arguments and read through the list of sub-commands that we didn't cover in this tutorial. What did you find? What might they be useful for?
- Can you figure out how to switch to the version of Node.js that your OS came with?
- Why would you want to change to a different version of Node.js while doing development?
- What happens when you install an npm package globally (e.g. `npm install -g express`) while using NVM to manage Node.js versions?

Additional resources

- Official NVM repository and documentation (GitHub.com)

Node.js Locally

Variables in Node.js

Data Brokering with Node.js

- Overview of Data Brokering with Node.js
- How the Node Module System Works
- What is NPM?
- What Is package.json?
- Create a package.json File
- What Is package-lock.json?
- How to Install NPM Packages
- How to Uninstall NPM Packages from a Node.js Project
- How to Use Semantic Versioning in NPM
- How to Update a Node Dependency with NPM
- What Are NPM Scripts?
- Restart a Node.js Application upon Changing a File
- Node.js ETL (Extract, Transform, Load) Pipeline: What Are We Building?
- What Is the Node.js ETL (Extract, Transform, Load) Pipeline?
- Understanding Promises in Node.js
- Use Promise.all to Wait for Multiple Promises in Node.js
- Use JavaScript's Async/Await with Promises
- How to Make API Requests with Request-Promise in Node.js
- ETL: Extract Data with Node.js
- ETL: Transform Data with Node.js
- What Is the Node.js fs (File System) Module?
- Read/Write JSON Files with Node.js
- ETL: Load Data to Destination with Node.js
- What Is a Node.js Stream?
- Use Streams to Extract, Transform, and Load CSV Data
- Stream to an HTTP Response with Node.js

- Overview: How to Manage Node.js Locally

> Install Node.js Locally with Node Version Manager (nvm)

- How to Use Environment Variables in Node.js
- Set up and Test a Dot Env (.env) File in Node
- What Is an API Proxy?
- What Is the Express Node.js Framework?
- How to Add a Route to an Express Server in Node.js
- Express Middleware in Node.js
- Use Express to Create an API Proxy Server in Node.js
- How to Set up an Express.js Server in Node.js
- Organize Your Node.js Code into Modules
- Set up Routes for Your API in Node.js
- Optimize an Express Server in Node.js
- Add Compression to Express in Node.js
- Add Response Caching to a Node.js Express Server
- How the Event Loop Works in Node.js
- Explore the Timers Phase of Node's Event Loop
- Explore the I/O Callbacks Phase of the Node.js Event Loop
- Explore the Immediate Callbacks Phase of Node's Event Loop
- What Is the Difference Between Authorization and Authentication?
- What Are Form Validation and Sanitization?
- Process a User Login Form with ExpressJS
- How to Validate and Sanitize an ExpressJS Form
- What Is the Difference Between Sessions and JSON Web Tokens (JWT) Authentication?
- Set Up ExpressJS Session Authentication for Node Applications
- Authenticate Users With Node ExpressJS and Passport.js