https://devcenter.heroku.com/articles/nodejs-support#node-js-runtimes

Heroku Node.js Support

*Last Updated: 07 May 2015*

[**cedar**](https://devcenter.heroku.com/tags/cedar) [**node**](https://devcenter.heroku.com/tags/node)

Table of Contents

* [Activation](https://devcenter.heroku.com/articles/nodejs-support#activation)
* [Node.js runtimes](https://devcenter.heroku.com/articles/nodejs-support#node-js-runtimes)
* [Specifying a Node.js Version](https://devcenter.heroku.com/articles/nodejs-support#specifying-a-node-js-version)
* [Specifying an Npm Version](https://devcenter.heroku.com/articles/nodejs-support#specifying-an-npm-version)
* [Build Behavior](https://devcenter.heroku.com/articles/nodejs-support#build-behavior)
* [Customizing the Build Process](https://devcenter.heroku.com/articles/nodejs-support#customizing-the-build-process)
* [Cache Behavior](https://devcenter.heroku.com/articles/nodejs-support#cache-behavior)
* [Runtime Behavior](https://devcenter.heroku.com/articles/nodejs-support#runtime-behavior)
* [Default Web Process Type](https://devcenter.heroku.com/articles/nodejs-support#default-web-process-type)
* [Warnings](https://devcenter.heroku.com/articles/nodejs-support#warnings)
* [Add-ons](https://devcenter.heroku.com/articles/nodejs-support#add-ons)
* [Multi-buildpack Behavior](https://devcenter.heroku.com/articles/nodejs-support#multi-buildpack-behavior)
* [Going further](https://devcenter.heroku.com/articles/nodejs-support#going-further)

This document describes the general behavior of the Cedar stack as it relates to the recognition and execution of Node.js applications. For a more detailed explanation of how to deploy an application, see [Getting Started with Node.js on Heroku](https://devcenter.heroku.com/articles/getting-started-with-nodejs).

[Activation](https://devcenter.heroku.com/articles/nodejs-support#activation)

The Heroku Node.js buildpack is employed when the application has either a package.json file or a server.js file in the root directory.

[Node.js runtimes](https://devcenter.heroku.com/articles/nodejs-support#node-js-runtimes)

Node versions adhere to [SemVer](http://semver.org/), the semantic versioning convention popularized by GitHub. SemVer uses a version scheme in the formMAJOR.MINOR.PATCH.

* MAJOR denotes incompatible API changes
* MINOR denotes added functionality in a backwards-compatible manner
* PATCH denotes backwards-compatible bug fixes

Node’s versioning strategy is [borrowed from Linux](http://en.wikipedia.org/wiki/Software_versioning#Odd-numbered_versions_for_development_releases), where odd MINORversion numbers denote unstable development releases, and evenMINOR version numbers denote stable releases. Here are some examples for node:

* 0.8.x: stable
* 0.9.x: unstable
* 0.10.x: stable
* 0.11.x: unstable

[Supported runtimes](https://devcenter.heroku.com/articles/nodejs-support#supported-runtimes)

Heroku’s node support extends to the latest stable MINOR version and the previous MINOR stable version that still receives security updates.

Currently, those versions are 0.10.x and 0.12.x.

[Other available runtimes](https://devcenter.heroku.com/articles/nodejs-support#other-available-runtimes)

While there are limits to the Node versions Heroku officially supports, it is possible to run any available version of Node beyond 0.8.5, including unstable pre-release versions like 0.11.x. To see which versions of node are currently available for use on Heroku, visit[semver.io/node.json](http://semver.io/node.json) or [what-is-the-latest-version-of-node.com](http://what-is-the-latest-version-of-node.com/).

Unstable versions 0.11.15 and greater are not compatible with the legacy cedar stack. You can upgrade to cedar-14 (heroku stack:set cedar-14) or lock the version at 0.11.14.

Additionally, Heroku supports the use of the [io.js](https://iojs.org/) Node fork. Support for iojs is a beta feature and may change.

[Specifying a Node.js Version](https://devcenter.heroku.com/articles/nodejs-support#specifying-a-node-js-version)

Use the engines section of your package.json to specify the version of Node.js to use on Heroku:

{

"name": "myapp",

"description": "a really cool app",

"version": "0.0.1",

"engines": {

"node": "0.10.x"

}

}

To try the io.js beta, replace ‘node’ with ‘iojs’:

{

"engines": {

"iojs": "1.0.x"

}

}

You should always specify a node version, but if you don’t the latest stable version will be used.

[Specifying an Npm Version](https://devcenter.heroku.com/articles/nodejs-support#specifying-an-npm-version)

Use the engines section of your package.json to specify the version of Npm to use on Heroku:

{

"name": "myapp",

"description": "a really cool app",

"version": "0.0.1",

"engines": {

"npm": "2.1.x"

}

}

If you don’t specify a version of Npm, the default version bundled with Node will be used. We recommend specifying a version greater than or equal to 2.1.x since that branch fixes many common Npm issues.

[Build Behavior](https://devcenter.heroku.com/articles/nodejs-support#build-behavior)

npm install is run on every build, even if the node\_modules directory is already present, to ensure execution of any [npm script hooks](https://npmjs.org/doc/misc/npm-scripts.html) defined in your package.json.

npm prune is run after restoring cached modules to ensure cleanup of any unused dependencies. You must specify all of your application’s dependencies in package.json, else they will be removed by npm prune.

On each build, the node runtime version is checked against the version in the previous build. If the version has changed, npm rebuild is run automatically to recompile any binary dependencies. This ensures your app’s dependencies are compatible with the installed node version.

[Customizing the Build Process](https://devcenter.heroku.com/articles/nodejs-support#customizing-the-build-process)

If your app has a build step that you’d like to run when you deploy, you can use an npm postinstall script, which will be executed automatically after the buildpack runs npm install. Here’s an example:

"scripts": {

"start": "node index.js",

"test": "mocha",

"postinstall": "bower install && grunt build"

}

Your app’s [environment is available](https://devcenter.heroku.com/changelog-items/416) during the build, allowing you to adjust build behavior based on the values of environment variables. For instance:

NPM\_CONFIG\_PRODUCTION=true

Npm reads configuration from any environment variables beginning with[NPM\_CONFIG](https://docs.npmjs.com/misc/config). We set production=true by default to install dependenciesonly. If you would like to install additional devDependencies, you can disable this behavior:

heroku config:set NPM\_CONFIG\_PRODUCTION=false

Since you usually don’t want all devDependencies in your production builds, it’s preferable to move only the dependencies you actually need for a build into dependencies (bower, grunt, gulp, etc).

You can also control npm’s behavior via a [.npmrc](https://docs.npmjs.com/files/npmrc) file in your project’s root.

[Cache Behavior](https://devcenter.heroku.com/articles/nodejs-support#cache-behavior)

Heroku maintains a [cache directory](https://devcenter.heroku.com/articles/buildpack-api#caching) that is persisted between builds. This cache is used to store resolved dependencies so they don’t have to be downloaded and installed every time you deploy.

NODE\_MODULES\_CACHE=true

This variable determines whether or not the Node buildpack uses cached node\_modules from previous builds. It defaults to true, but you can disable caching (and force clean builds) by overriding it:

heroku config:set NODE\_MODULES\_CACHE=false

git commit -am 'disable node\_modules cache' --allow-empty

git push heroku master

If you check your node\_modules directory into source control, the build cache is **not** used. We do not recommend [checking node\_modules into git.](https://docs.npmjs.com/misc/faq#should-i-check-my-node_modules-folder-into-git-)

While we cache node\_modules by default, some projects can benefit from fine-grained control over caching. For instance, you may want to cachebower\_components or the node\_modules of “client” and “server” sub-directories. To specify directories that should be cached between builds, provide a cacheDirectories array in your top-level package.json:

"cacheDirectories": ["node\_modules", "bower\_components"]

[Runtime Behavior](https://devcenter.heroku.com/articles/nodejs-support#runtime-behavior)

The buildpack puts node and npm on the PATH so they can be executed with [heroku run](https://devcenter.heroku.com/articles/one-off-dynos" \l "an-example-one-off-dyno) or used directly in a Procfile:

$ cat Procfile

web: npm start

The NODE\_ENV environment variable is unset by default. If you would like to set it (to specify staging, production, etc):

heroku config:set NODE\_ENV=production

[Default Web Process Type](https://devcenter.heroku.com/articles/nodejs-support#default-web-process-type)

A Procfile is not required to run a Node.js app on Heroku. If noProcfile is present in the root directory of your app during the build process, we will check for a scripts.start entry in your package.jsonfile. If a start script entry is present, a default Procfile is generated automatically:

$ cat Procfile

web: npm start

The default npm start script is node server.js. If no scripts.start entry is present but a server.js file is found, the default Procfile will be generated as:

$ cat Procfile

web: node server.js

Read more about npm script behavior at [npmjs.org](https://npmjs.org/doc/misc/npm-scripts.html).

[Warnings](https://devcenter.heroku.com/articles/nodejs-support#warnings)

During builds, the Node.js Buildpack identifies common issues in Node applications and provides warnings with best-practice recommendations. If you’re experiencing Node.js build issues, this is a good place to look for guidance.

[Add-ons](https://devcenter.heroku.com/articles/nodejs-support#add-ons)

No add-ons are provisioned by default. If you need a SQL database for your app, add one explicitly:

$ heroku addons:create heroku-postgresql

[Multi-buildpack Behavior](https://devcenter.heroku.com/articles/nodejs-support#multi-buildpack-behavior)

When using the Node.js Buildpack with [the Multi Buildpack](https://github.com/heroku/heroku-buildpack-multi), it automatically exports Node, Npm, and node\_modules binaries onto the path for subsequent buildpacks to consume.

[Going further](https://devcenter.heroku.com/articles/nodejs-support#going-further)

The Heroku Node.js buildpack is open source. For a better technical understanding of how the buildpack works, check out the source code at [github.com/heroku/heroku-buildpack-nodejs](https://github.com/heroku/heroku-buildpack-nodejs#readme).

[Feedback](https://devcenter.heroku.com/articles/nodejs-support#feedback)

[Log in to submit feedback](https://devcenter.heroku.com/login?back_to=%2Farticles%2Fnodejs-support&utm_campaign=login&utm_medium=feedback&utm_source=web).