Troubleshooting Node.js Deploys

1. clean heroku cache

heroku config:set NODE\_MODULES\_CACHE=false

1. clean npm cache

$npm cache clean <package name>@version

1. download latest version node.js and npm, reinstall it
2. don’t check in node\_modules
3. See, A folder failed to build, B folder was built successfully. Try put js folder and partial folder, and index.html of A folder into B folder, replace those same name folders in B.

*Last Updated: 01 May 2015*

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

Table of Contents

* [Check your buildpack](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#check-your-buildpack)
* [Compare Node and Npm Versions](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#compare-node-and-npm-versions)
* [Ensure you aren’t relying on untracked dependencies](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#ensure-you-aren-t-relying-on-untracked-dependencies)
* [Enable verbose logging](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#enable-verbose-logging)
* [Start with a blank slate](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#start-with-a-blank-slate)
* [Open a ticket](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#open-a-ticket)

Your Node.js deploy failed - now what? Start with these simple steps to troubleshoot a build issue.

[Check your buildpack](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#check-your-buildpack)

Are you using the officially supported and maintained buildpack, or something else? Most of the time, the standard buildpack is the best choice - either alone, or paired with other official buildpacks (like Ruby) through the [multi buildpack](https://github.com/heroku/heroku-buildpack-multi).

Find out by running:

$ heroku buildpacks

To use the official buildpack:

$ heroku buildpacks:set https://github.com/heroku/heroku-buildpack-nodejs

[Compare Node and Npm Versions](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#compare-node-and-npm-versions)

Your production environment should mirror your development environment, especially in the case of important binaries. First, check your local versions:

$ node --version

$ npm --version

Then, compare the results with your package.json engines section. You*are* [specifying a node version](https://devcenter.heroku.com/articles/nodejs-support#specifying-a-node-js-version), right?

You can see which binaries Heroku is using on each deploy in the build logs, which look something like this:

remote: -----> Installing binaries

remote: Resolving node version 0.10.x via semver.io...

remote: Downloading and installing node 0.10.35...

remote: Using default npm version: 1.4.28

They should match up with the same versions you saw locally. If they don’t, you should [specify the matching versions](https://devcenter.heroku.com/articles/nodejs-support#specifying-a-node-js-version) in your package.json.

[Ensure you aren’t relying on untracked dependencies](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#ensure-you-aren-t-relying-on-untracked-dependencies)

When you npm install a module, that module is saved to your project’snode\_modules directory *but not to the package.json file that describes your project’s dependencies.* This means that your project will work on your machine, but as soon as you try to share it elsewhere it will fail with missing dependencies.

Instead, install locally with npm install --save foobar. That will automatically record the installation to your package.json dependencies list.

For the same reason, you should avoid installing global dependencies (npm install -g foobar). Use --save instead to put binaries inmyproject/node\_modules/.bin, keeping them local and trackable.

If you run binaries (like grunt or bower) from an npm script, the node\_modules/.bin directory is automatically added to thepath, so you don’t need to include it in your script.

Here’s how to check whether or not you’re relying on commonly-installed global modules:

$ which bower

$ which grunt

$ which gulp

If any of these show a path, your project may depend on a binary outside of its local directory, leading to a case of, *“but it works on my machine!”*

[Enable verbose logging](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#enable-verbose-logging)

By default, Heroku installs dependencies quietly. Sometimes, it can be helpful for debugging to get verbose output:

heroku config:set NPM\_CONFIG\_LOGLEVEL=verbose

On your next push, you’ll see more details:

git commit -am 'verbose logging' --allow-empty

git push heroku master

[Start with a blank slate](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#start-with-a-blank-slate)

Each time you run npm install, npm leaves packages that meet your semver requirements untouched. That’s why an npm install today may lead to a different tree than the npm install you ran yesterday, even if your package.json didn’t change.

Therefore, it’s a good practice to periodically clear node\_modules and reinstall from scratch to ensure that your package.json dependencies are still valid:

$ rm -rf node\_modules

$ npm install --quiet --production

$ npm start

In fact, those are essentially the commands that Heroku runs when we build and launch your project. If they work locally, you’re likely to be cloud-ready.

Alternatively, you can add [npm-shrinkwrap](https://docs.npmjs.com/cli/shrinkwrap) to your workflow to lock down dependency versions so that they are consistent from install to install. This is the best practice for production environments.

[Open a ticket](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#open-a-ticket)

If none of these solutions work for you, [open a ticket](https://help.heroku.com/) with Heroku so we can help.

[Keep reading](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#keep-reading)

* Category: [Node.js](https://devcenter.heroku.com/categories/nodejs)
* [Heroku Node.js Support](https://devcenter.heroku.com/articles/nodejs-support)

[Feedback](https://devcenter.heroku.com/articles/troubleshooting-node-deploys#feedback)

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