Ivn Cote

Follow 80 Followers About

Version Freeze with NPM Done Right

Ivn Cote Mar 28, 2017 · 3 min read

UDPATE April 2019: *I see some incoming traffic for this article, so I have to update it with the important remark that the examples here are outdated. NPM team introduced* `package-lock.json` *. And working with it is much more easier than before.*
If you still use NPM v4 for some reason, or if you want to have a broader context, please continue reading. Happy coding!

I promised to consider NPM shrinkwrap file creation and maintenance. So I suppose it would be a good start to understand why we need such a file at all.

As soon as your project depends on many other open source frameworks and libraries, this dependency has a dark side in form of breaking changes. Usually we use semver agreement to handle modifications in node modules. So we can describe in package.json what degree of changes we can admit. But there is no guarantee that authors of the pointed library won't make a mistake with semver some day. Especially this could be an issue with WIP projects (e.g. alpha versions, and refactoring to a new API).

Even if all packages have very concrete versions in package description file, there is a chance that dependencies can be not so specified inside these modules. That means you would have situation when all is working fine on your local machine, but a production service is down.

Shrinkwrap json file consist of references for each dependencies you need for the project, with version specified. If such file is present in the root folder of your project, `npm install` will use it for sure, and not package.json. So far this file itself is a great solution for CI/CD for node based projects! One common advice here to use `dev` flag so that dev packages will be included as well, and all CI tests would be still possible.

Sometimes the technique of avoiding undesired changes in dependant modules using specified versions is called version freeze. Lets consider VF for your app from maintenance perspective, because there are a lot of articles which describe using this file creation and update as a nightmare. Contrarily, my experience is bright. Some discipline and good understanding of NPM is all you need!

First of all, in order to create shrinkwrap json you have to get all packages already installed (and placed in `node_module` folder). My advice is to don't install optional packages which are often presented in development utilities, such as `fsevents`. The risk that your CI would be broken is high when an optional module is pointed in schrikwrap file, since it is OS dependant, and therefore failed on NIX machines. There is a special flag for omitting these dependencies during install process: `no-optional`, and it is valid for several NPM commands.

Wrapping up, to create the first version of shrinkwrap file do

```
npm i --no-optional --no-progress --quiet
npm shrinkwrap
# or npm shrinkwrap --dev
```

npmshrink.md hosted with ❤ by **GitHub** **view raw**

BTW don't forget to add the created file for changes to commit.

Then you can check that all is working fine by deleting folder node_modules, and running `npm i` again.

Fine, now there is a question how to update some modules for the time being. If you are wondering which packages should be updated, run `npm updated`. The result is a table with package available for promoting their versions. Or maybe you would like to add a new module. In any case you already know the version you want. So simply add it locally

```
npm i --save --no-optional [module]@[semver]
# or npm i --save-dev --no-optional [module]@[semver]
```

adddep.md hosted with ❤ by **GitHub** **view raw**

That's all: you could check that both `package.json` and `npm-shrinkwrap.json` have proper modifications concerning the new (or updated) module.

Using shrinkwrap you avoid issues with incompatibilities on different machines over time as well as speed up modules downloading process. Both stories are very crucial for continuous integration. So, happy coding and shipping to production! All the information is valid for NPM v4.1.2

Thanks to Tanya Mulkidzhanova.

JavaScript NPM Continuous Integration