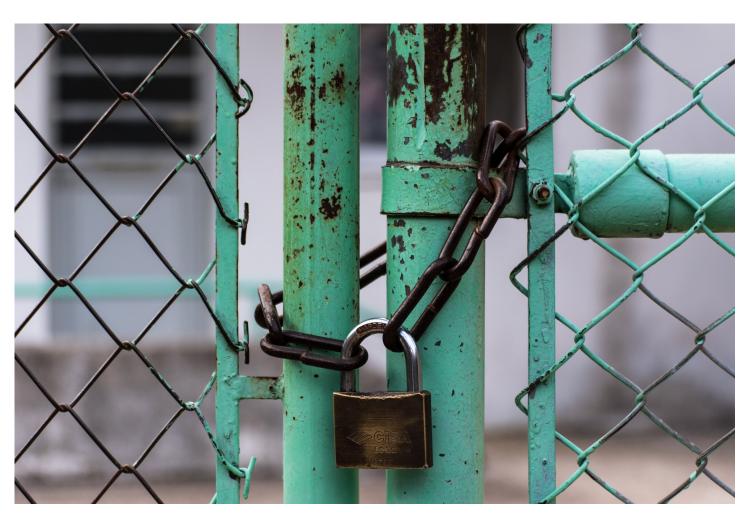


## Disabling package-lock.json

I love to stay up to date with Node.js and npm, but sometimes change can be confusing. I'm talking about package—lock.json, which was introduced in npm v5.

npm notice created a lockfile as package-lock.json. You should commit this file.

... but maybe not. 😲



Lock via unsplash.com

### **Disabling package-lock.json Locally**

To tell npm not to create a package-lock.json lock file for your current project, create a file called .npmrc at the root of the project and add package-lock=false to it.

\*nix users may use:

```
echo 'package-lock=false' >> .npmrc
echo 'package-lock.json' >> .gitignore
```

#### Disabling package-lock.json Globally

If you want completely disable package—lock.json creation on your machine, simply set the config globally.

```
npm config set package—lock false
```

# Installing without creating the lock (one time)

I recommend one of the above approaches instead of this.

```
rm -f package-lock.json && \
npm install lodash --save && \
rm -f package-lock.json
```

#### **But why?**

I've run into multiple instances where dependencies do not install as expected due to package-lock.json existing. The lock file is created every time a dependency is installed ( npm install lodash ) or npm install is run in npm v5. The lock file can easily get out of date if packge-lock.json exists before the install or if a co-worker forgets to update packge-lock.json before pushing to a repository.

Others are hitting this situation as well. If you are interested in diving deeper into the issue, there's a good discussion going on in the npm

issue tracker (#16866).

I'm not saying you *should* disable <code>packge-lock.json</code>, but doing so has enabled me to keep my workflow with npm v5, as I used with npm v4. I expect these usability issues will be ironed out in future versions.

P.S. Please don't forget to <u>shrinkwrap your dependencies</u> for production projects!