禁用 package-lock

NPM Node.js 缓存 版本

npm（Node Package Manager）是由 JavaScript 编写的 Node.js 默认的包管理工具， 会随 Node 一起安装。NPM 是伟大的工具，在它的基础上构建了现在的整个 JavaScript 生态。 这些模块有每周数十亿的下载量，可以用来构建 Web 服务， 命令行工具，IoT 节点， 桌面应用，甚至操作系统。

npm 5.0 开始会自动生成 package-lock.json，解决 npm 无法递归锁定版本的问题（类似 yarn）同时使用该文件作为缓存来加速依赖解析。 但现在看来 package-lock 制造的问题比解决的问题还要多，有些争议性的处理细节仍在讨论。

Harttle 不建议现在使用 package-lock，文章尾部给出了禁用 package-lock 的方式。

所以什么是版本锁定?

npm 的依赖定义在 `package.json` 中，它的语法叫做 语义版本。 即只声明接受哪些版本（更新）而不是写死某个版本，借此可以方便地获得最新的 Bugfix 与新特性。 但只使用语义版本来约束依赖也会造成问题，比如：

开发完成后，但部署之前某个依赖发布了新版。这时部署时安装的版本是未经测试的，存在风险。 写死版本无法解决问题。因为你的依赖不一定写死了版本，深层的依赖仍然可能会更新。

因此 NPM 社区提出了很多锁定版本的机制，比如 shrinkwrap.json 和 yarn。 npm 5.0 便提供了 `package-lock.json` 试图解决这个问题。

package-lock 是如何工作的?

同样是用来锁定递归依赖的机制，我们先把 `package-lock.json` 与 npm 很早就支持的 shrinkwrap 机制做一个对比：

- `package-lock.json` 尝试反应 `node_modules` 的真实状态，每次 npm 操作都会更新；`npm-shrinkwrap.json` 只有执行 `npm shrinkwrap` 时才生成。
- `package-lock.json` 只作用于当前项目不会发布到 npm。即使 `node_modules` 下真的有 `pac lock.json` 也不起作用；而 `npm-shrinkwrap.json` 可以发布到 npm 并在所有安装它的地方都生效。

因此，如果你在开发 library，可以用 `npm-shrinkwrap.json` 让使用者完全复现你发布时的依赖树。 如果你在开发一个顶层项目（不作为 library 被别人使用），可以用 `package-lock.json` 来锁定版本， 使得每次 `npm install` 都能得到同样的依赖树。

package-lock 的问题

resolved 字段

`package-lock.json` 中 `dependencies.<package name>.resolved` 字段保存了包的 `.tgz` 所在的 URL。众所周知每个人的 Registry 配置都不同，尤其是国内有很强的需要使用 `registry.npm.taobao.org`。 所以这一地址是环境相关的，不应进入代码仓库。否则会有数不清的麻烦：

- 不同 Registry 的代码产生 Merge diff。
- CI 工具所在环境可能无法（或很慢）访问 `package-lock.json` 中的源。
- 不小心暴露内网地址。

测试时依赖树不够新

如果你的项目中存在 `package-lock.json`，所有开发者 `install` 得到的依赖树都是锁定版本。 而用户安装你的库后，得到的依赖树却不是锁定的版本。（还记得吗？`package-lock.json` 不可发布） 类似地你的 CI 工具 使用的也是锁定的版本， 不再反映安装最新依赖后能否正确运行。

因此如果使用 `package-lock.json`，建议更改 CI 工具的配置，在执行 `install` 前先删除 `package-lock.json`。如果你在使用 Travis CI，可以在 `.travis.yml` 中加一个 `before_script`：

```
before_script:
  - rm package-lock.json
```

当然这可能会让你的本地与 CI 工具跑出不同的测试结果，anyway。

缓存失效问题

由于 npm 5.0 的这一特性不向后兼容，如果同一个项目的开发者中有人使用新版有人使用旧版， 这个问题会变得很麻烦。

除不必要的 git diff 之外，还可能发生安装失败的问题。 因为 `package-lock.json` 中的 `depend` `.<package name>.integrity` 记录了文件的哈希。

与 package.json 的关系

如果 `package.json` 有所更新，安装时便不能只按照 `package-lock.json` 来。 那么这时进行 `npm install` 就可能导致 `package-lock.json` 的变化。 这一行为一听就很让人费解。

在这里有非常集中的讨论： https://github.com/npm/npm/issues/16866

如何禁用 package-lock

因为 `package-lock.json` 是自动生成的，可以配置 npm 来避免经常需要手动删除这个文件。 在当前项目禁用 `package-lock.json`：

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

在（当前机器的当前用户的）所有项目禁用 `package-lock.json`：

```
npm config set package-lock false
```


> 上一篇：恢复 TMUX 工作区
>
> 下一篇：合理地设计 URL