# 环境：

1. Node.js v10.24.1

npm v6.14.12

1. vue-cli 相关gitHooks的文档地址：[https://cli.vuejs.org/zh/guide/cli-service.html#git-hook](https://cli.vuejs.org/zh/guide/cli-service.html" \l "git-hook)
2. 由于 vue-cli 内置了 yorkie ( 上面的官网文档说：yorkie fork 自 husky 并且与后者不兼容。)

也就是说我们按照官网这个内置的软件程序的使用流程来使用就能得到git hook 的效果。

Yorkie的github地址： https://github.com/yyx990803/yorkie

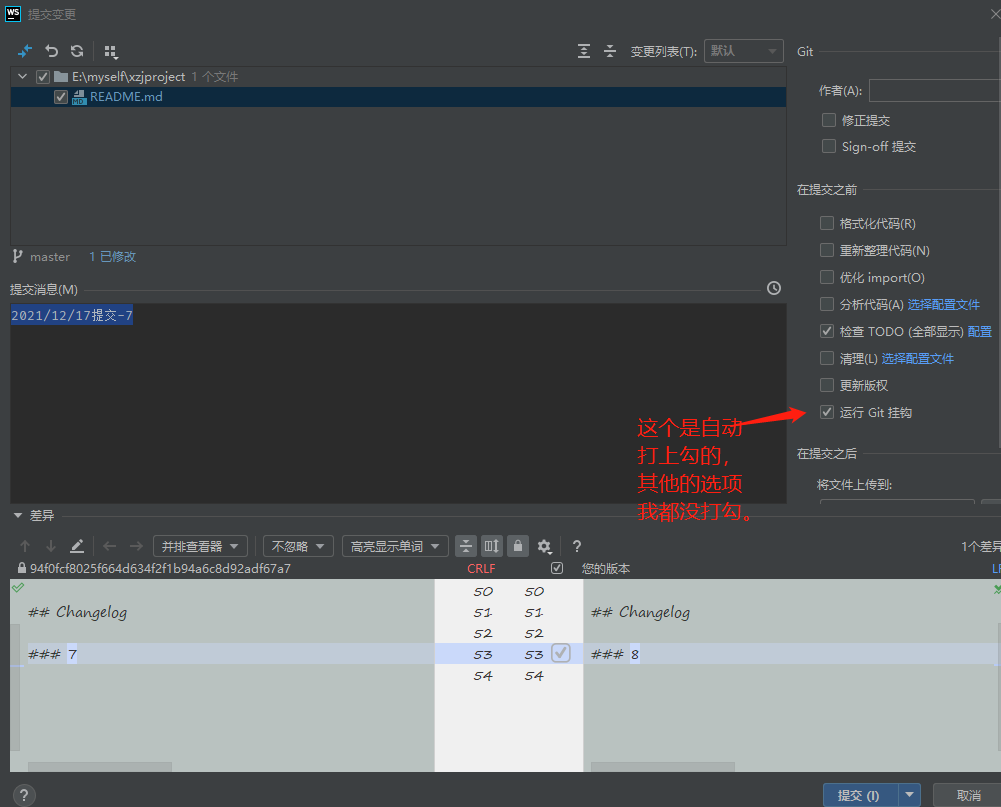
1. Git hook 的git相关地址是：<https://www.git-scm.com/book/zh/v2/%E8%87%AA%E5%AE%9A%E4%B9%89-Git-Git-%E9%92%A9%E5%AD%90>
2. Husky 的官网地址：[https://typicode.github.io/husky/#/](https://typicode.github.io/husky/" \l "/)

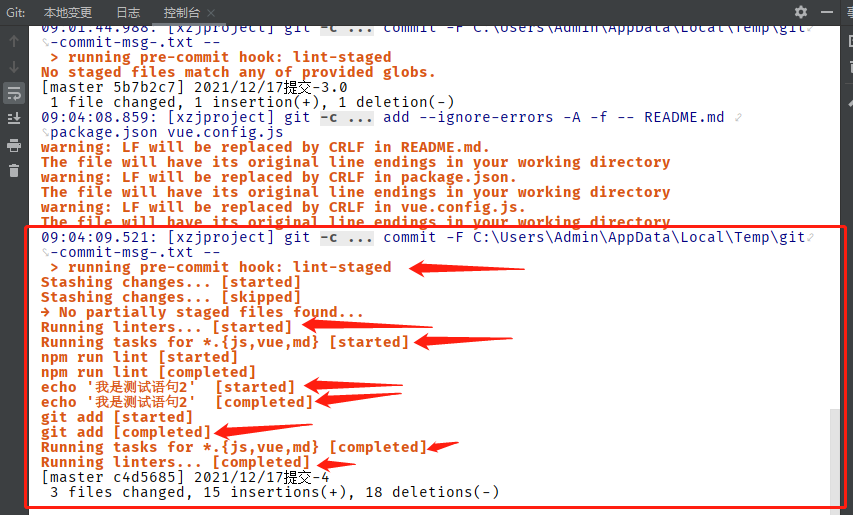
# 过程：

按照vue-cli 官网的介绍，我安装了lint-staged 版本^12.x ,但是发现这版的lint-staged 用了 import fs from ‘fs’ 语法，但是在Node.js v10版本中是不支持该 import 语法的，



所以，我把 lint-staged的版本降到了8.x, 然后经过测试发现：





git-hook的效果出现了，pre-commit钩子生效。后测试在node v10.x下lint-staged可使用的最高版本为 ^10.5.4。

Lint-staged的github地址： https://github.com/okonet/lint-staged

之后：

出于对Node.js升级的预防，我必须去测试在Node.js ^12.x 版本下，对应多少版本的lint-staged, 才能实现 git hook 效果。

过程如下： 升级node.js 到12.x (我安装的是Node.js v12.22.8);

分别测试 lint-staged 包在^8.x , ^10.x , ^12.x版本下是否能完整达到git hook(pre-commit)的效果：

|  |  |
| --- | --- |
| Lint-staged版本 | Git hook 效果是否达到 |
| lint-staged^8.x ：  "lint-staged": "^8.2.1", | 成功， |
| lint-staged^10.x  "lint-staged": "^10.5.4", | 成功， |
| lint-staged^12.x  lint-staged@12.1.3 | 成功， |

结论，lint-staged v12.x 的也可以用。Git hook 功能也能实现。

主体流程完毕。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分割线\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Lint-staged github地址： https://github.com/okonet/lint-staged

# 以下对于lint-staged的常用功能做出介绍：

1. Configuration

Starting with v3.1 you can now use different ways of configuring lint-staged:

1. lint-staged object in your package.json （我们使用的方式）
2. ……其他方式请参阅上面的地址，这里不再赘述，反正我们也不用。

Configuration should be an object where each value is a command to run and its key is a glob pattern to use for this command. This package uses [micromatch](https://github.com/micromatch/micromatch) for glob patterns. JavaScript files can also export advanced configuration as a function. See [Using JS configuration files](https://github.com/okonet/lint-staged#using-js-configuration-files) for more info.

2.Ignoring files

The concept of lint-staged is to run configured linter tasks (or other tasks) on files that are staged in git. lint-staged will always pass a list of all staged files to the task, and ignoring any files should be configured in the task itself.

Consider a project that uses [prettier](https://prettier.io/) to keep code format consistent across all files. The project also stores minified 3rd-party vendor libraries in the vendor/ directory. To keep prettier from throwing errors on these files, the vendor directory should be added to prettier's ignore configuration, the .prettierignore file. Running npx prettier . will ignore the entire vendor directory, throwing no errors. When lint-staged is added to the project and configured to run prettier, all modified and staged files in the vendor directory will be ignored by prettier, even though it receives them as input.

In advanced scenarios, where it is impossible to configure the linter task itself to ignore files, but some staged files should still be ignored by lint-staged, it is possible to filter filepaths before passing them to tasks by using the function syntax. See [Example: Ignore files from match](https://github.com/okonet/lint-staged#example-ignore-files-from-match).

3. Reformatting the code

Tools like [Prettier](https://prettier.io/), ESLint/TSLint, or stylelint can reformat your code according to an appropriate config by running prettier --write/eslint --fix/tslint --fix/stylelint --fix. Lint-staged will automatically add any modifications to the commit as long as there are no errors.

{

"\*.js": "prettier --write"

}

Prior to version 10, tasks had to manually include git add as the final step. This behavior has been integrated into lint-staged itself in order to prevent race conditions with multiple tasks editing the same files. If lint-staged detects git add in task configurations, it will show a warning in the console. Please remove git add from your configuration after upgrading.

4.What commands are supported?

Supported are any executables installed locally or globally via npm as well as any executable from your $PATH.

5.Running multiple commands in a sequence

You can run multiple commands in a sequence on every glob. To do so, pass an array of commands instead of a single one. This is useful for running autoformatting tools like eslint --fix or stylefmt but can be used for any arbitrary sequences.

For example:

{

"\*.js": ["eslint", "prettier --write"]

}

going to execute eslint and if it exits with 0 code, it will execute prettier --write on all staged \*.js files.