使用 git subtree 管理项目中的子库

随着项目越来越多，很多项目依赖同一个模板或是配置文件想同一管理，又不想分开维护，所以只能互相引用，或是各自维护，导致了后续的很多麻烦。怎么办呢？

1 背景

最近工作中遇到了一个问题：随着项目越来越多，很多项目依赖同一个模板或是配置文件想同一管理，又不想分开维护，所以只能互相引用，或是各自维护，导致了后续的很多麻烦。

场景一： 很多公司手机端和 pc 同时开发，引用同一套模板，或者通用的组件库。

场景二：用 gulp、webpack 来打包，或是用 node 来开发会有一些通用的配置文件需要统一管理。

1.可以使用 git submodule 方法，来建立一个子模块，方法见
http://toimc.com/2016/11/git-submodule/

2.使用 git subtree 方法：

可以在 git bash 中使用 git subtree --help 来看官方的说明。

语法：

#从提交中，创建本地目录
git subtree add   -P <prefix> <commit>

#从仓库中，创建本地目录
git subtree add   -P <prefix> <repository> <ref>

#更新与推送

```
git subtree pull  -P <prefix> <repository> <ref>
git subtree push  -P <prefix> <repository> <ref>

#快速拆分目录代码
git subtree split -P <prefix> [OPTIONS] [<commit>]

#与指定提交进行合并
git subtree merge -P <prefix> <commit>
```

合并指定提交中的代码到 subtree 中来。

如：

```
#使用 git subtree add 新建了目录 lib
git subtree add -P lib <repository> <ref>

#此后 lib 中的文件发生了多次的改变
#可以使用 git subtree merge 到某次的提交
git subtree merge -p lib <commit>
```

使用 git log 或者 git reflog 来查看提交代码，取前 6 位数

2 情景：项目 A 中已经有 lib 库，需要在其他地方使用。

方法一：把 lib 目录（lib 分支）中的代码，使用以下命令进行常规提交到另一个库

1. 使用 split 方法

```
#语法
git subtree split -P <prefix> [OPTIONS] [<commit>]

#实际使用
git subtree split -P lib -b newLib
```

git push origin newLib

#在其他项目中，拉取分支内容
git clone --branch=newLib <repository> <ref>

此命令会把目录下的 lib 目录，新建一个分支为 newLib,里面会包含所有与 lib 目录下文件相关的 commit。

以上命令，相当于是建立了一个新的分支去管理 lib 目录中的文件，

2. 或者在分支中初始化一个 git 仓库，推送代码到远程分支，对分支进行管理（不过 这样就与 submodule 没有区别了，而且逻辑更复杂不便管理）

git init
git remote add lib <reponame> <repourl>
git add .
git commit -m "first commit
git push origin master

方法二：使用 git subtree push 方法

语法：

git subtree push -P lib <repository> <ref>
git subtree push --prefix=<子目录名> <子仓库名> <分支> --squash

3 情景：项目 A 中需要 newLib 项目中的代码作为 lib 库

1. 使用 git subtree add 命令新建目录

git subtree add --prefix=<子目录名> <子仓库名> <分支> --squash

--squash 会把 subtree 上的改动合并成一次 commit。

2. 使用 pull & push 操作更新代码 ##

pull：git subtree pull --prefix=<子目录名> <子仓库名> <分支> --squash

push：git subtree push --prefix=<子目录名> <子仓库名> <分支> --squash

4 总结

1. 在新员工加入团队时：一次性 clone 项目，submodule 可以一起 clone 出来，只需添加--recursive 递归参数就可以了，而 subtree 并不行，只能手动添加，不过可以借助神器 Yeoman(一个自动生成项目脚手架的工具)来实现。

2. subtree 适合像配置文件这种需要跟着项目走的情况。

3. submodule 适合在开发阶段时引用，到了生产环境会被打包到指定文件内，而本身并不用跟着版本走的情况。

5 参考文献

[1] "Git submodule 的坑"

[2] 使用 git subtree & submodule 管理多个子项目

[3] Mastering Git subtrees

[4] 用 Git Subtree 在多个 Git 项目间双向同步子项目，附简明使用手册