git submodule 的介绍

常用命令 git submodule add，git submodule init 等。

submodule 子模块的意思，git submodule 的应用场景：经常有这样的事情，当你在一个项目上工作时，你需要在其中使用另外一个项目。也许它是一个第三方开发的库或者是你独立开发和并在多个父项目中使用的。这个场景下一个常见的问题产生了：你想将两个项目单独处理但是又需要在其中一个中使用另外一个。

这里有一个例子。假设你在开发一个网站，为之创建 Atom 源。你不想编写一个自己的 Atom 生成代码，而是决定使用一个库。你可能不得不像 CPAN install 或者 Ruby gem 一样包含来自共享库的代码，或者将代码拷贝到你的项目树中。如果采用包含库的办法，那么不管用什么办法都很难去定制这个库，部署它就更加困难了，因为你必须确保每个客户都拥有那个库。把代码包含到你自己的项目中带来的问题是，当上游被修改时，任何你进行的定制化的修改都很难归并。

Git 通过子模块处理这个问题。子模块允许你将一个 Git 仓库当作另外一个 Git 仓库的子目录。这允许你克隆另外一个仓库到你的项目中并且保持你的提交相对独立。

1 git submodule add 添加子模块

命令：

git submodule add 仓库地址 路径

注意：路径不能以 / 结尾（会造成修改不生效）、不能是现有工程已有的目录（不能顺利 Clone）；仓库地址是指子模块仓库地址，路径指将子模块放置在当前工程下的路径。

2 git submodule init 初始化，update 更新

当 clone 一个项目带子模块的项目。 当你接收到这样一个项目，你将得到了包含子项目的目录，但里面没有文件：

命令：

```
#初始化子模块
git submodule init

#当子模块中的代码更新后，更新项目
git submodule update
```

子模块中，可以正常的使用 git add,git commit,git push 等常见的 git 操作。

3 删除子模块

step1:删除 config 配置

```
git submodule deinit -f docker
```

使用-f 删除文件，使用--cached 保留文件。

step2:删除子模块缓存

```
git rm --cached the_submodule
```

step3:删除子模块工作区文件

```
rm -rf the_submodule
```

step4：删除子模块在仓库中的空文件夹

```
rm -rf .git/modules/the_submodule
```