**[总结自己的Git常用命令](http://www.cnblogs.com/lwzz/archive/2013/02/23/2921426.html)**

使用git也有一段时间了，把自己常用的命令用自己的描述记录起来，方便自己备忘也方便其他人参考。

目录:

**最基本的命令**：

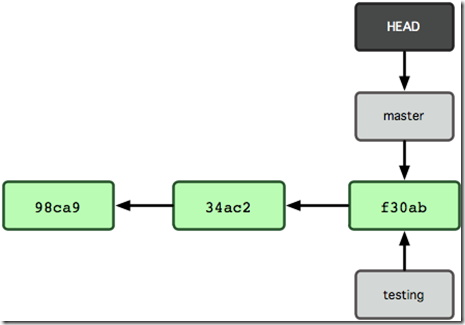
git clone 拷贝并跟踪远程的master分支。跟踪的好处是以后可以直接通过pull和push命令来提交或者获取远程最新的代码，而不需要指定远程分支名字。

git submodule init

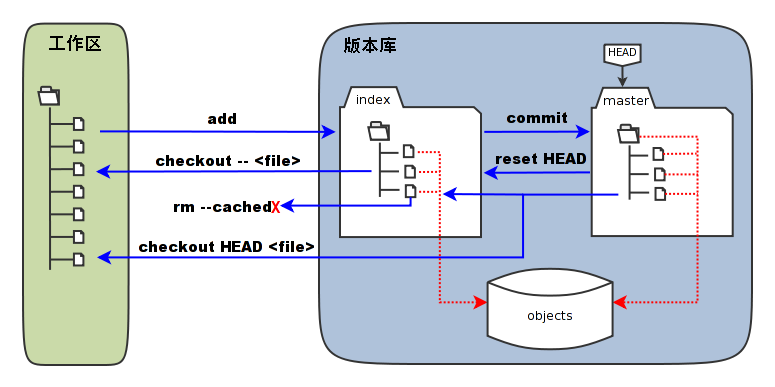
git submodule update

**参考示意图**

HEAD 指向当前的commit 对象(可以想象为当前分支的别名)，同时也用来表明我们在哪个branch上工作。所以当我们使用HEAD来操作指针的时候，其实就是不改变当前的commit的指向。

[](http://images.cnitblog.com/blog/99928/201302/21234140-04d8a98f07df402a856cd7c53942eb62.png)

对照这张图来理解两段提交，工作区(working tree)，暂存区（index）和 branch之间的关系



**显示信息类命令**   
git ls-files -u 显示冲突的文件，-s是显示标记为冲突已解决的文件

git diff 对比工作区和stage文件的差异   
git diff --cached 对比stage和branch之间的差异

git branch 列出当前repository下的所有branch   
git branch --a 列出local 和remote下的所有branch

git ls-files --stage 检查保存在stage的文件

git log 显示到HEAD所指向的commit为止的所有commit记录 。使用reset HEAD~n 命令使HEAD指针向前移动，会导致HEAD之后的commit记录不会被显示。

git log -g则会查询reflog去查看最近做了哪些动作，这样可以配合git branch 恢复之前因为移动HEAD指针所丢弃的commit对象。如果reflog丢失则可以通过git fsck --full来查看没被引用的commit对象。   
git log -p -2 对比最新两次的commit对象   
log -1 HEAD

git log --pretty=oneline

git log --stat 1a410e 查看sha1为1a410e的commit对象的记录

git blame -L 12,22 sth.cs 如果你发现自己代码中 的一个方法存在缺陷,你可以用git blame来标注文件,查看那个方法的每一行分别是由谁 在哪一天修改的。下面这个例子使用了-L选项来限制输出范围在第12至22行

**创建类命令**   
git brach branchName 创建名为branchName的branch   
git checkout branchName 切换到branchName的branch   
git checkout -b 创建并切换，也就是上面两个命令的合并

git brach branchName ef71 从commit ef71创建名为branchName的branch

**撤销类命令**如果是单个文件   
1.use "git reset HEAD <file>..." to unstage   
如果已经用add 命令把文件加入stage了，就先需要从stage中撤销

然后再从工作区撤销   
2.use "git checkout -- <file>..." to discard changes in working directory

git checkout a.txt  撤销a.txt的变动（工作区上的文件）   
如果是多个文件   
git chenkout .

如果已经commit 了，则需要   
git commit --amend 来修改，这个只能修改最近上一次的,也就是用一个新的提交来覆盖上一次的提交。因此如果push以后再做这个动作就会有危险

$ git reset --hard HEAD 放弃工作区和index的改动,HEAD指针仍然指向当前的commit.（参照第一幅图）

这条命令同时还可以用来撤销还没commit的merge,其实原理就是放弃index和工作区的改动，因为没commit的改动只存在于index和工作区中。

$ git reset --hard HEAD^ 用来撤销已经commit的内容(等价于 git reset --hard HEAD~1) 。原理就是放弃工作区和index的改动，同时HEAD指针指向前一个commit对象。

git revert 也是撤销命令，区别在于reset是指向原地或者向前移动指针，git revert是创建一个commit来覆盖当前的commit,指针向后移动

**提交类命令**git add 跟踪新文件或者已有文件的改动，或者用来解决冲突

git commit 把文件从stage提交到branch

git commit -a 把修改的文件先提交到stage,然后再从stash提交到branch

**删除类命令**   
git rm --cached readme.txt 只从stage中删除，保留物理文件

git rm readme.txt 不但从stage中删除，同时删除物理文件

git mv a.txt b.txt 把a.txt改名为b.txt

**Merge类命令**

在冲突状态下，需要解决冲突的文件会从index打回到工作区。

1.用工具或者手工解决冲突   
2.git add 命令来表明冲突已经解决。   
3.再次commit 已解决冲突的文件。

$ git reset --hard ORIG\_HEAD 用来撤销已经commit 的merge.   
$ git reset --hard HEAD 用来撤销还没commit 的merge,其实原理就是放弃index和工作区的改动。

git reset --merge ORIG\_HEAD，注意其中的--hard 换成了 --merge，这样就可以避免在回滚时清除working tree。