Git命令行操作

Git是目前世界上最先进的分布式版本控制系统，很方便公司同事间合作开发，使用Git可以使用可视化的软件，如SourceTree，现在还用中文版本，使用很方便；但是有的公司要求使用命令行，下面我就把一些常用的Git命令整理总结下，方便大家参考。

创建本地仓库

1、创建一个空目录

```
cd 〈路径〉
mkdir 〈仓库名字〉
git init
```

2、添加文件到仓库

```
git add 〈文件〉   eg：git add HelpTool.h
git commit -m "注释说明文件"
```

注：可以一次添加多个文件，统一提交 eg:

```
git add HelpTool.h HelpTool.m
git add HomeViewController.h
git add HomeViewController.m
git commit -m "添加两个文件"
```

3、查看仓库情况

```
git status 查看仓库当前的状态，可以了解仓库中文件状态（哪些被修改，增加，删除等）
git diff 〈文件〉可以查看具体文件被修改了什么地方
git log 显示从最近到最远的提交日志
git log --pretty=oneline 去除多余的信息，只显示版本号（commit id）和注释说明文字
```

4、版本回退

如果要回退版本，Git必须知道当前版本是哪个版本，在Git中用HEAD表示当前版本，上一版本是HEAD，上上版本就是HEAD^，往上50个版本就是HEAD~50。

```
git reset --hard HEAD^  回退到上个版本
```

如果回退到上个版本后又想回退回来，分两种情况：

1）终端命令行窗口还没关闭，在上面找到你想回退回来的那个版本的commit id

```
git reset --hard 〈commit id〉
```

2）终端已关闭退出

```
git reflog 记录了每一次提交命令，找到对应的commit id
git reset --hard <commit id>进行回退
```

5、撤销修改（包括添加、修改、删除等）

1）未添加到暂存区

```
git checkout <文件名> 把工作区的修改全部撤销
```

2）已添加到暂存区

```
git reset HEAD <文件名> 把暂存区的修改撤销掉，重新放回工作区
git checkout <文件名> 把工作区的修改全部撤销
```

6、删除文件

```
git rm <文件名> 从工作区删除
git commit -m "delete one file" 从仓库删除
```

7、克隆远程仓库。关于远程仓库的操作，可以看我的另一篇文章http://www.jianshu.com/p/27264e1bd447

```
git clone <远程仓库地址> 可以是https，也可以是ssh
```

8、创建合并分支

8.1.创建分支

```
git branch 列出所有分支，当前分支前面会有个*
git checkout -b <分支名> 创建并切换到新建的分支，相当于两条命令：
git branch <分支名> 创建分支
git checkout <分支名> 切换分支
```

8.2合并分支

```
git merge <分支名>   合并某分支到当前分支，这是快速合并，没有合并分支信息
git merge --no-ff -m "注释说明文字"<分支名>  合并时生成一个新的commit，这样，从分支历史上可以看出分支信息
```

9、删除分支

```
git branch -d <分支名>
git branch -D <分支名>  丢弃一个没有被合并过的分支，需要强行删除
```

10、暂存分支

```
git stash  把当前工作区的内容暂存起来，等以后恢复继续操作
```

11、恢复暂存分支

```
git stash apply  恢复后，暂存的内容并不会删除，需要用git stash drop 删除
git stash pop     恢复的同时把暂存的内容也删除了
```

12、查看远程库信息

```
git remote
git remote -v显示更详细的信息
```

13、本地创建的分支推送到远程仓库（分两步）

13.1 先抓取远程的新提交

```
git pull  可能有冲突，有则解决
```

13.2 推送本地分支

```
git push origin <分支名字>
```

14、在本地创建和远程分支对应的分支，分两步

14.1 本地和远程分支的名称最好一致

```
git checkout -b <分支名字1> origin/<分支名字1>
```

14.2 建立本地分支和远程分支的关联

```
git branch --set-upstream <分支名字1> origin/<分支名字1>
```

15、标签

15.1 打标签，标签默认是打在最新提交的commit上的，分两步

1）切换到需要打标签的分支上

```
git branch <分支名字>
```

2）打标签

```
git tag <标签名字>  eg：git tag v1.1
```

15.1.1 给特定的commit id 打标签

```
git tag <标签名字><commit id>  eg：  git tag v2.1 5366446
```

15.1.2 带有说明的标签

```
git tag -a <标签名> -m "说明文字" <commit id>  eg：git tag -a v2.1 -m "version 2.1 realeased" 347689321
```

15.2、 推送标签到远程仓库（创建的标签默认都是在本地）

```
git push origin <标签名>
git push origin --tags 一次推送所有本地标签到远程仓库
```

15.3、查看所有标签

```
git tag
```

15.4 查看标签具体信息

```
git show <标签名>
```

15.5 删除标签

15.5.1 删除本地标签

```
git tag -d <标签名>
```

15.5.2 删除远程仓库标签，分两步

1）先删除本地

```
git tag -d <标签名>
```

2）再删除远程

```
git push origin : refs/tags/<标签名>
```

16、配置忽略文件

1）在Git工作区的根目录下创建.gitignore文件，然后把要忽略的文件名填进去，Git就会自动忽略这些文件

2）把.gitignore提交到Git，就OK了！

GitHub上有各种配置文件，只需组合一下就可以使用，可以直接在线浏览https://github.com/github/gitignore。

17、给命令配置别名

有些命令不太好记或者经常混淆，配置别名不但可以解决前面说的问题，还可以提高效率。

配置Git的时候，加上--global是针对当前用户起作用的，如果不加，只针对当前的仓库起作用。

a.当前用户的Git配置文件放在用户主目录下的一个隐藏文件.gitconfig中；

b.每个仓库的Git配置文件都放在.git/config文件中。

配置别名可以直接在上面的配置文件里进行配置

```
git config --global alisa.<别名> <原名>
```

eg:

```
git config --global alias.co checkout->git co fixBug=git checkout fixBug
git config --global alias.ci committ->git ci -m "upload a file"=git commit -m "upload a file"
git config --global alias.br branch->git br fixBug=git branch fixBug
```

如果对两个单词的命令设置别名

```
git config --global alisa.<别名> <'原名'>
```

eg：

```
git config --global alias.cob 'checkout -b'->git cob fixBug=git checkout -b fixBug
```