虽然之前学过 Git，不过用的比较少，很多原理也不是很清楚，所以最近是买了个视频，静下心来从零学习了下 Git，不过 Git 用来用去其实也就那几个命令，今天这篇文章，我把那些命令都分类总结了一下（当然，是参考别人的分类的），这些总结大部分来自于我买的一个 Git 鼠标垫，主要是为了以后自己忘记了方便查找。相信这份 Git 宝典，一定值得你收藏。

一、Git 配置相关

如果你首次使用 Git，那刚开始首先是需要配置各种身份信息的，这样当你提交相关任务的时候，别人才能知道这个 commit 是谁提交的。

Git 最小配置

1、配置全局账户，也就是该账户对所有的 Git 仓库都有效

```
git config --global user.name '你的账户名称'
git config --global user.email '你的 Email'
```

2、配置局部账户，也就是该账户只对当前 Git 仓库有效

```
git config --local user.name '你的账户名称'
git config --local user.email '你的 Email'
```

注意，不同点就是一个参数是 global（全局），一个是 local(本地)

3、查看相关配置情况

配置了之后，显然有时候是需要查看我们当前配置的相关情况的，可以使用如下命令

1、查看 global 类型的配置情况

```
git config --global --list
```

2、查看某个仓库下的配置情况

```
git config --local --list
```

二、本地基本操作

这部分命令有点多，也是使用的最频繁的命令了，待我一一列举出来，建议收藏

1、基本操作

1、查看变更情况

```
git status
```

2、查看当前工作在哪个分支上

```
git branch -v
```

3、切换到指定分支

```
git checkout 指定分支的名称
```

4、把**当前目录及其子目录**下所有变更都加入到**暂存区**

```
git add . // 注意，add 后面是一个 '.';
```

5、把仓库内**所有**变更都假如到暂存区

```
git add -A
```

6、把指定文件添加到暂存区

```
git add 文件1 文件2 ... 文件n
```

7、创建正式的 commit，也就是把当前的数据提交上去

```
git commit
```

2、比较差异

1、比较某文件工作区和暂存区的差异

```
git diff 某文件
```

2、比较某文件暂存区和 HEAD 的差异

```
git diff --cache 某文件
```

3、比较工作区和暂存区的所有差异

```
git diff
```

4、比较暂存区和 HEAD 的所有差异

```
git diff --cache
```

3、暂存区与工作区之间回滚

1、把工作区指定文件恢复成和暂存区一样

```
git checkout 文件1 文件2 ... 文件n
```

2、把暂存区指定文件回复和 HEAD 一样

```
git reset 文件1 文件2 ... 文件n
```

3、把暂存区和工作区所有文件恢复成和 HEAD 一样

```
git reset --hard
```

4、用 difftool 比较任意两个 commit 的差异

```
git difftool commit1 commit2
```

> 注意，从工作区回滚到暂存区则用 checkout ， 否则用 reset

4、其他

查看哪些文件没有被 Git 管控

```
git ls-files --others
```

三、加塞临时任务处理

1、把未处理完的变更先保存到 stash 中

```
git stash
```

2、临时任务处理完后继续之前的工作

```
git stash pop // pop 相当于栈的出栈和入栈一样，把之前的任务弹出来
或者
git stash apply // 和 pop 不同的是， apply 相当于从栈顶把任务取出来，但是不过从栈中把任务移除
```

3、查看所有的 stash

```
git stash list
```

4、取回某次 stash 的变更

```
git stash pop stash @{数字n}
```

四、修改个人分支历史

我们的仓库的内容每次变更执行 commit 的时候，都会生成一个新的 commit，不过有时候，我们不想

产生新的 commit，而是想要通过修改之前的 commit 来变更仓库的内容，那么就可以使用如下命令了

1、修改最后一次 commit

```
1、在工作区中修改文件
2、git add
3、git commit --amend
```

2|、修改中间的 commit(假设代号为 X)

```
1. git rebase -i X前面的一个 commit 的 id
2. 在工作区修改文件
3. git add
4. git rebase --contiue
```

五、查看变更日志等

1、当前分支各个 commit 用一行显示

```
git log --online
```

2、显示最近的 n 个 commit

```
git log -n
```

3、用图示显示所有的分支历史

```
git log --online --graph --all
```

4、查看涉及到某文件变更的所有 commit

```
git log 某文件
```

5、某文件各行最后修改对应的 commit 以及作者

```
git blame 某文件
```

六、分支与标签

1、创建新分支

基于当前分支创建新分支

```
git branch 新分支
```

基于指定分支创建新分支

```
git branch 新分支 已有分支
```

基于某个 commit 创建分支

```
git branch 新分支 某个 commit 的id
```

创建分支并且切换到该分支

```
git chechout -b 新分支
```

2、列出分支

列出本地分支

```
git branch -v
```

列出本地和远端分支

```
git branch -av
```

列出远端所有分支

```
git branch -rv
```

列出名称符号某样式的远端分支

```
git branch -rv -l '某样式'
```

3、删除分支

安全删除本地某分支

```
git branch -d 要删除的分支
```

强行删除本地分支

```
git branch -D 要删除的分支
```

删除已合并到 master 分支的所有本地分支

```
git branch --merged master | grep -v '^\*\| master' | xargs -n 1 git branch -d
```

删除远端 origin 已不存在的所有本地分支

```
git remote prune origin
```

4、打标签

从 commit 打上标签

```
git tag 标签名 commit 的id
```

七、两分支之间的集成

1、把 A 分支合入到当前分支，且为 merge 创建 commit

```
git merge A分支
```

2、把 A 分支合入到 B 分支，且为 Merge 创建 commit

```
git merge A分支 B分支
```

3、把当前分支基于B分支做 rebase，以便把B分支合入到当前分支

```
git rebase B分支
```

4、把A分支基于B分支做rebase，以便把B分支合入到A分支

```
git rebase B分支 A分支
```

5、用 mergetool 解决冲突

```
git mergetool
```

八、和远端交互

1、列出所有 remote

```
git remote -v
```

2、增加 remote

```
git remote add url地址
```

3、删除 remote

```
git remote remove remote的名称
```

4、改变 remote 的name

```
git remote rename 旧名称 新名称
```

5、把远端所有分支和标签的变更都拉到本地

```
git fetch remote
```

6、把远端分支的变更拉倒本地，且 merge 到本地分支

```
git pull remote名称 分支名
```

关于 pull 和 fetch 的区别不懂可以看这篇文章从0学习Git：详解git pull和git fetch的区别

7、把本地分支 push 到远端

```
git push remote名称 分支名
```

8、删除远端分支

```
git push remote --delete 远端分支名
或者
git push remote:远端分支名
```

9、向远端提交指定标签

```
git push remote 标签名
```

10、向远端提交所有标签

```
git push remote --tags
```

总结

如果把这些命令掌握了，那么 git 就顺手拈来了，不过很多命令是比较容易忘的，所有还是比较需要一份 git 手册，以后用到的时候多查几次，多用几次命令，相信就能够记住了。

不过单单会命令不知道原理的话还是不大行，至于原理，可以通过网上找文章慢慢了解，当然，如果有需要，我后面也会分享 git 的一些原理，让大家从本质上读懂 git。