Git clone url

Cd ‘进入的项目名’

工：工作区 暂：暂存区 版：版本区 >：到

Git status 查状态

Git add 工>暂

Git add . 工作区所有文件>暂

Git commit -m “修改取名 ” 暂>版（里面的所有文件都会提交）

Git commit -a -m “修改取名” 工>暂>版

对比

Git diff 工和暂文件差异的对比

Git diff --cached(--staged) 暂和版对比

Git diff master 工和版对比

撤销

Git reset HEAD <file.name> 暂文件 撤销到 工

Git checkout -- <file.name> 工的文件状态还原到版的文件的状态

Git commit -m “修改取名” --amend 误提交后，撤销上一次的提交，并再次提交

删除

工作区的文件可以直接在文件夹中删除

Git rm <file.name> 删除暂中的文件（前提是工对应的文件已经删除）

Git rm -f <file.name> 删除暂中的文件（包括工对应的文件）

Git rm --cached <file.name>删除暂中的文件（不包括对应的文件）

恢复

commit\_id就是git log 命令下出来的commit后面的一大串数字，可以是全部，可以是一部分

git checkout commit\_id <file.name> 恢复commit\_id版本的文件

git reset --hard commit\_id 恢复到commit\_id版本

git reset --hard HEAD^ 恢复到上一个版本

git reset --hard HEAD~<num> 恢复到第num个版本

git reflog 记录每一次的操作行为

同步到远程仓库

git remote 先获取到仓库名

-v

Origin 默认的仓库名

git push origin master 将本地的操作同步到github上面

多人协作解决冲突

git fetch 先查看不合并（推荐）

Git diff master origin/master 查看之间的不同

Git merge orgin/master 通过手动的合并

git pull 查看后直接合并

Git分支

git branch 查看分支 \*在哪个分支上就指当前在哪个分支上

Git bramch ‘分支名称’ 生成这个分支

Git bramch -d ‘分支名称’ 删除这个分支(这个分支要是在当前分支下的)

-D 强制把没有合并的分支删除

--merged 查看分支下面所合并的的分支

--no-merged 查看分支下面没有合并的的分支

git checkout ‘分支名称’ 指向这个分支

Git checkout -b ‘分支名称’ 生成一个分支，并指向它

-b

[当前状态的分支] git merge ‘分支名称’ 将这个分支合并到当前状态的分支上

github上的分支

git push origin “分支名称” 将这个分支同步到github

github上直接创建

github上的标签

git tag “版本名，随便起 例如v1.0”

github上直接创建