

echo "# GitNote" >> README.md 是添加一个README文件

git init 是创建一个本地仓库

git add 文件名 把该文件添加到仓库缓存区

git add . 可以把所有文件添加到仓库缓存区（能匹配所有文件）

git commit -m "更新信息" 带更新信息的一次版本更新

git remote add origin xx 添加一个为xx的远程仓库

git push -u origin master 把本地master分支上传到远程origin仓库

-u指定该远程仓库为默认的

//以上就是刚刚在GitHub上面创建远程仓库时他给的命令的作用

git pull 拉取远程仓库：

\*如果一开始远程仓库有README或者在上传之后再直接修改（有内容且与本地不同）会报错

需要先执行这条语句

我寒假考核就遇到了这个问题，不过不是直接git pull解决的，还加了些参数（运行得非常慢）

这个经尝试快得多（不是很懂

git re**set** --hard id 重置到id版本 HEAD是当前版本，后面加上几个^就指前几个版本，或者~加数字

git log 可以查看版本记录，而git reflog可以查看reset的记录

git status 查看当前状态

git rm删除文件

感觉目前这些东西就比较有用

（1）**git stash** save "save message"  : 执行存储时，添加备注，方便查找，只有git stash 也要可以的，但查找时不方便识别。

（2）**git stash list** ：查看stash了哪些存储

（3）**git stash show** ：显示做了哪些改动，默认show第一个存储,如果要显示其他存贮，后面加stash@{$num}，比如第二个 git stash show stash@{1}

（4）**git stash show -p** : 显示第一个存储的改动，如果想显示其他存存储，命令：git stash show  stash@{$num}  -p ，比如第二个：git stash show  stash@{1}  -p

（5）**git stash apply** :应用某个存储,但不会把存储从存储列表中删除，默认使用第一个存储,即stash@{0}，如果要使用其他个，git stash apply stash@{$num} ， 比如第二个：git stash apply stash@{1}

（6）**git stash pop** ：命令恢复之前缓存的工作目录，将缓存堆栈中的对应stash删除，并将对应修改应用到当前的工作目录下,默认为第一个stash,即stash@{0}，如果要应用并删除其他stash，命令：git stash pop stash@{$num} ，比如应用并删除第二个：git stash pop stash@{1}

（7）**git stash drop** stash@{$num} ：丢弃stash@{$num}存储，从列表中删除这个存储

（8）**git stash clear** ：删除所有缓存的stash