

|  |  |
| --- | --- |
| git init | 创建初始git仓库 |
| git status | 显示当前git状态 |
| git status -u | 显示所有untracked的文件 |
| git clone https://github.com/tusonggao/CTP\_Simplest\_Demo.git | 克隆github终端某个代码库 |
| git reset --hard | 强制放弃本地的修改，把文件还原到原始状态 |
| git add README | 开始跟踪README文件 |
| git add --all | 增加所有文件到trunk |
| git add new\_dir/\* | 增加新的文件夹到trunk |
| git rm file111 | 从 Git 中移除file111文件，本地也会同时删除 |
| git rm --cached file111 | 从 Git 中移除file111文件，本地保留 |
| git rm -rf some\_directory | 从 Git 中强制移除some\_directory文件夹，本地也会一并删除 |
| git reset –hard HEAD some\_file | 撤销some\_file的修改,回退到上一次提交的状态 |
| git commit -m "first commit" | 提交已经暂存的修改到git repo中（在windows中要使用双引号） |
| git commit -am "second commit" | 会将新的内容先stage 然后commit（在windows中要使用双引号） |
| git help config | 获取config命令的帮助信息 |
| git diff readme.txt | 对比文件的修改内容 |
| git log | 显示从最近到最远的提交日志 |
| git log -3 | 显示从最近到最远的3条提交日志 |
| git log --since=yesterday | 显示从昨天以来的提交日志 |
| git push --force | 强制推送变更到trunk |
| git pull | 把github上的变更同步到本地 |
| git ls-files | 显示有哪些文件是在版本控制下 |
| git blame <filename> | 帮你发现文件中的每一行是在什么时间以及为什么被加上或更改的。 |

关于分支使用，Git鼓励大量使用分支：

|  |  |
| --- | --- |
| git branch | 查看分支 |
| git branch <name> | 创建分支 |
| git checkout <name> | 切换分支 |
| git checkout -b <name> | 创建+切换分支 |
| git merge <name> | 合并某分支到当前分支 |
| git branch -d <name> | 删除分支 |
| git merge --abort | 当发现merge出现冲突时，可以使用该命令终止merge |
| git merge --no-ff -m "merge with no-ff" dev | 不使用fast-forward模型进行merge |
| git log --graph --pretty=oneline --abbrev-commit | 简明的展示所有的commit历史 |
| git stash list | 显示当前分支中国所有的stash内容 |
| git stash apply stash@{1} | 恢复当前分支中某项stash的内容 |
| git stash clear | 删除当前分支中所有stash的内容 |

创建一个全新repository的方法：

0. cd mblog

1. git init

2. git add . 该命令将所有文件（包括子文件夹中的文件加入git控制系统中）

3. git commit - m "first commit"

4. git remote add origin https://github.com/tusonggao/mblog.git

https://github.com/tusonggao/mblog.git 这个文件是要自己在github中设定的

5. git push -u origin master

参考学习资料

|  |  |
| --- | --- |
| Pro git | https://progit.bootcss.com/#\_pro\_git |
| 廖雪峰git教程 | https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/ |
|  |  |

git diff使用

git diff 查看当前工作树与暂存区的差别。如果暂存区是空的，那么它将显示工作树与最新提交状态之间的差别。“+”号标出的是新添加的行，被删除的行用“-”表示。

git diff --staged 查看暂存区与最新一次提交之间的差别

git diff HEAD 查看本次修改(不管是否stage)与上次提交之间的区别。这里的HEAD是指向当前分支中最新一次提交的指针。在git commit前查看两次提交之间的差异，是一个良好的编程习惯。一旦commit之后，立即使用该命令，该命令的输出内容会变为空。

git push有-u这个参数，加了参数-u后，运行git push -u origin master，

以后即可直接用git push 代替git push origin master