**# 设置提交代码时的用户信息**

**$ git config [--global] user.name "[name]"**

**$ git config [--global] user.email "[email address]"**

**# 设定你喜欢的编辑软件**

**$ git config --global core.editor sublime\_text.exe**

**# 下载一个项目和它的整个代码历史**

**$ git clone git://github.com/schacon/grit.git mygrit**

**# 在当前目录新建一个Git代码库**

**$ git init**

**# 显示当前的Git配置**

**$ git config –list**

**# 添加指定文件到暂存区**

**$ git add README.txt**

**$ git add .**

**# 删除工作区文件，并且将这次删除放入暂存区**

**$ rm test.txt**

**# 改名**

**$ git mv first.txt second.txt**

**# 移动文件**

**$ git mv first.txt myFolder**

**# 移动文件并改名**

**$ git mv first.txt myFolder/second.txt**

**# 提交暂存区到仓库区**

**$ git commit**

**$ git commit -m "Story 182: Fix benchmarks for speed"**

**# 列出所有本地分支**

分支前的 \* 字符：它表示当前所在的分支

**$ git branch**

**# 新建一个分支，但依然停留在当前分支**

**$ git branch testing**

**# 切换到指定分支，并更新工作区**

**$ git checkout testing**

**# 合并指定分支到当前分支**

**$ git merge [branch]**

**# 删除分支**

**$ git branch -d hotfix**

**# 列出所有已有的tag**

**$ git tag**

**# 新建一个tag在当前commit**

**$ git tag [tag]**

**# 新建一个tag在指定commit**

**$ git tag [tag] [commit]**

**# 含附注的标签**

-a （译注：取 annotated 的首字母）指定标签名字

**$ git tag -a v1.4 -m 'my version 1.4'**

**# 签署标签**

**$ git tag -s v1.5 -m 'my signed 1.5 tag'**

**# 轻量级标签**

**$ git tag v1.4-lw**

**# 验证标签**

**$ git tag -v v1.4.2.1**

**# 查看tag信息**

**$ git show [tag]**

**# 提交指定tag**

**$ git push origin v1.5**

**$ git push origin --tags**

**# 检查当前文件状态：仅仅列出了修改过的文件**

**$ git status**

**# 显示当前分支的版本历史**

**$ git log**

**# 显示commit历史，以及每次commit发生变更的文件**

**$ git log --stat**

**$ git log -p -2**

**# 显示暂存区和工作区的差异**

**$ git diff**

**# 显示暂存区域快照 和 仓库区的差异**

**$ git diff --staged**

**# 显示两次提交之间的差异**

**$ git diff [first-branch]...[second-branch]**

**# 显示某次提交的元数据和内容变化**

**$ git show [commit]**

**# 显示当前分支的最近几次提交**

**$ git reflog**

**# 从远程仓库抓取数据/ 下载远程仓库的所有变动**

**$ git fetch [remote]**

**# 显示所有远程仓库**

-v 选项（译注：此为 --verbose 的简写，取首字母），显示对应的克隆地址

**$ git remote -v**

**# 远程仓库的删除和重命名**

**$ git remote rename pb paul**

**# 显示某个远程仓库的信息**

**$ git remote show origin**

# 增加一个新的远程仓库，并命名

**$ git remote add pb git://github.com/paulboone/ticgit.git**

**# 取回远程仓库的变化，并与本地分支合并**

**$ git pull [remote] [branch]**

**# 推送数据到远程仓库/ 上传本地指定分支到远程仓库**

**$ git push [remote] [branch]**

**# 推送所有分支到远程仓库**

**$ git push [remote] –all**

**# 修改最后一次提交：撤消刚才的提交操作**

**$ git commit --amend**

**# 取消对文件的修改（把未添加到暂存区的内容撤销掉）**

**$ git checkout -- benchmarks.rb**

**# 恢复文件**

**$ git checkout – test.txt**

**# 恢复上一个commit的所有文件到工作区**

**$ git checkout .**

**# 恢复某个commit的指定文件到工作区**

**$ git checkout efs001 -– text.txt**

**# 重置暂存区的指定文件，与上一次commit保持一致，但工作区不变**

**$ git reset [file]**

**# 重置暂存区与工作区，与上一次commit保持一致**

**$ git reset --hard**

**# 重置当前分支的指针为指定commit，同时重置暂存区，但工作区不变**

**$ git reset [commit]**

**# 重置当前分支的HEAD为指定commit，同时重置暂存区和工作区，与指定commit一致**

**$ git reset --hard [commit]**

**# 回退到上一个版本**

**$ git reset –hard HEAD^**

**# 重置当前HEAD为指定commit，但保持暂存区和工作区不变**

**$ git reset --keep [commit]**

**# 取消已经暂存的文件**

**$ git reset HEAD benchmarks.rb**

**# 显示所有文件**

**ls –a**

**# 删除 .git**

**rm –rf .git**

**# 删除本地仓库目录**

**rm –rf ml\_eeg**

**# 将当前的工作现场隐藏起来**

**git stash**