**创建**

克隆一个已存在的远程仓库

$ git clone git://user@domain.com/repo.git

**分支&标签**

列出所有已存在的分支

$ git branch -av

**合并&变基**

合并当前分支到HEAD指向的分支

$ git merge <branch>

创建一个新的本地仓库

$ git init

转换HEAD 指向的分支

$ git checkout <branch>

将当前HEAD指向的分支设为<branch>分支

切勿rebase 已推送的提交!

$ git rebase <branch>

基于当前HEAD指向的分支创建一个新的分支

$ git branch <new-branch>

**本地更改**

显示工作目录和暂存区的状态

$ git status

基于远程分支创建一个新的跟踪分支

$ git checkout --track <remote/bran-

ch>

删除本地一个分支

$ git branch -d <branch>

使用标记标识当前提交

$ git tag <tag-name>

**更新&发布**

列出当前所有已配置的远程仓库

$ git remote -v

**撤销**

丢弃工作目录中的所有本地更改

$ git reset --hard HEAD

显示关于指定<remote>远程仓库的信息

$ git remote show <remote>

添加新的远程仓库，名为remote

$ git remote add <shortname> <url>

**提交历史**

显示所有提交，从最近的开始

$ git log

从远程仓库拉取所有更改，但不整合到HEAD 分支

$ git fetch <remote>

从远程仓库拉取所有更改，并直接整合到HEAD 分支

$ git pull <remote> <branch>

谁在<file>中更改了什么

$ git blame <file>

推送本地所有更改到指定<remote>远程仓库

$ git push <remote> <branch>

并将之前所有更改保存为**unstaged**的更改

$ git reset <commit>

并保留未提交的本地更改

$ git reset --keep <commit>

**Git常用命令参考手册**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

created by lingb

https://github.com/a-yin

水平有限，有误请指出！

将HEAD重置为以前的提交，并丢弃此后的所有更改

$ git reset --hard <commit>

推送所有标签

$ git push --tags

在远程仓库删除一个分支

$ git branch -dr <remote/branch>

还原提交(通过生成具有相反更改的新提交)

$ git revert <commit>

丢弃特定文件中的本地更改

$ git checkout HEAD <file>

手动解决冲突后将文件添加为已解决的标记

$ git add <resolved-file>

中止一个变基

$ git rebase --abort

在解决冲突后继续重新部署基地

$ git rebase --continue

使用配置的合并工具来解决冲突

$ git mergetool

删除标记已解决的文件

$ git rm <resolved-file>

显示特定文件随时间的变化

$ git log -p <file>

更改最后一次提交

切勿修改已推送的提交!

$ git commit --amend

提交之前暂存(staged)的更改

$ git commit

将<file>中的一些更改添加到下一次提交中

$ git add -p <file>

提交跟踪文件中的所有本地更改

$ git commit -a

将所有当前更改添加到下一次提交

$ git add .

跟踪文件的更改

$ git diff