1 查看、添加、提交、删除、找回，重置修改文件

git help <command> # 显示 command 的 help
git show # 显示某次提交的内容 git show $id
git status # 查看文件的修改状态
git checkout -- <file> # 抛弃工作区修改
git checkout. # 抛弃工作区修改
git add <file> # 将工作文件修改提交到本地暂存区
git add . # 将所有修改过的工作文件提交暂存区
git rm <file> # 从版本库中删除文件
git rm <file> --cached # 从版本库中删除文件，但不删除文件
git reset <file> # 从暂存区恢复到工作文件
git reset -- . # 从暂存区恢复到工作文件
git reset --hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改
git commit <file> git commit. git commit -a # 将 git add, git rm 和 git commit 等操作都合并在一起做 git commit -am "some comments"
git commit --amend # 修改最后一次提交记录
git revert <$id> # 恢复某次提交的状态，恢复动作本身也创建次提交对象
git revert HEAD # 恢复最后一次提交的状态

2 查看文件 diff

git diff <file> # 比较当前文件和暂存区文件差异
git diff <$id1> <$id2> # 比较两次提交之间的差异
git diff <branch1>..<branch2> # 在两个分支之间比较
git diff --staged # 比较暂存区和版本库差异
git diff --cached # 比较暂存区和版本库差异
git diff --stat # 仅仅比较统计信息

3 查看提交记录

git log # 查看所有提交记录
git log <file> # 查看该文件每次提交记录
git log -p <file> # 查看每次详细修改内容的 diff
git log -p -2 # 查看最近两次详细修改内容的 diff
git log --stat #查看提交统计信息

4 Git 本地分支管理

4.1 查看、切换、创建和删除分支

git branch #查看当前分支
git branch -r # 查看远程分支
git branch <new_branch> # 创建新的分支
git branch -v # 查看各个分支最后提交信息
git branch --merged # 查看已经被合并到当前分支的分支
git branch --no-merged # 查看尚未被合并到当前分支的分支
git checkout <branch> # 切换到某个分支
git checkout -b <new_branch> # 创建新的分支，并且切换过去
git checkout -b <new_branch> <branch> # 基于 branch 创建新的 new_branch
git checkout $id # 把某次历史提交记录 checkout 出来，但无分支信息，切换到其他分支会自动删除
git checkout $id -b <new_branch> # 把某次历史提交记录 checkout 出来，创建成一个分支
git branch -d <branch> # 删除某个分支
git branch -D <branch> # 强制删除某个分支 (未被合并的分支被删除的时候需要强制)

4.2 分支合并和 rebase

git merge <branch> # 将 branch 分支合并到当前分支
git merge origin/master --no-ff # 不要 Fast-Foward 合并，这样可以生成 merge 提交
git rebase master <branch> # 将 master rebase 到 branch，相当于： git checkout <branch> && git rebase master && git co master && git merge <branch>

5 Git 补丁管理(方便在多台机器上开发同步时用)

git diff > ../sync.patch # 生成补丁
git apply ../sync.patch # 打补丁
git apply --check ../sync.patch #测试补丁能否成功

6 Git 暂存管理

git stash # 暂存
git stash list # 列所有 stash
git stash apply # 恢复暂存的内容
git stash drop # 删除暂存区

7 Git 远程分支管理

git pull # 抓取远程仓库所有分支更新并合并到本地
git pull --no-ff # 抓取远程仓库所有分支更新并合并到本地，不要快进合并
git fetch origin # 抓取远程仓库更新
git merge origin/master # 将远程主分支合并到本地当前分支
git checkout --track origin/branch # 跟踪某个远程分支创建相应的本地分支
git checkout -b <local_branch> origin/<remote_branch> # 基于远程分支创建本地分支，功能同上
git push # push 所有分支
git push origin master # 将本地主分支推到远程主分支
git push -u origin master # 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)
git push origin <local_branch> # 创建远程分支， origin 是远程仓库名
git push origin <local_branch>:<remote_branch> # 创建远程分支
git push origin :<remote_branch> #先删除本地分支(git br -d <branch>)，然后再 push 删除远程分支

8 Git 远程仓库管理

GitHub
git remote -v # 查看远程服务器地址和仓库名称
git remote show origin # 查看远程服务器仓库状态
git remote add origin git@ github:robbin/robbin_site.git # 添加远程仓库地址
git remote set-url origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址(用于修改远程仓库地址) git remote rm <repository> # 删除远程仓库
创建远程仓库
git clone --bare robbin_site robbin_site.git # 用带版本的项目创建纯版本仓库
scp -r my_project.git git@ git.csdn.net:~ # 将纯仓库上传到服务器上
mkdir robbin_site.git && cd robbin_site.git && git --bare init # 在服务器创建纯仓库
git remote add origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址
git push -u origin master # 客户端首次提交
git push -u origin develop # 首次将本地 develop 分支提交到远程 develop 分支，并且 track
git remote set-head origin master # 设置远程仓库的 HEAD 指向 master 分支
也可以命令设置跟踪远程库和本地库
git branch --set-upstream master origin/master
git branch --set-upstream develop origin/develop