**初始化配置**

git config --global user.name "Your Name Comes Here" #配置使用git仓库的人员姓名

git config --global user.email you@yourdomain.example.com #配置使用git仓库的人员email

git config --global credential.helper cache # 配置到缓存 默认15分钟

git config --global credential.helper 'cache --timeout=3600' # 修改缓存时间

git config --list # 列举所有配置

**增删改查**

git help # 显示帮助

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 revert <$id> # 恢复某次提交的状态，恢复动作本身也创建了一次提交对象

git revert HEAD # 恢复最后一次提交的状态

**查看文件**

git diff <file> # 比较当前文件和暂存区文件差异

git diff # 默认比较也是同上

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

**查看提交记录**

git log #查看提交记录

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

tig # Mac上可以使用tig代替diff和log，brew install tig

**仓库管理**

git init # 初始化一个版本仓库

git clone git@xbc.me:wordpress.git # Clone远程版本库

git remote add origin git@xbc.me:wordpress.git # 添加远程版本库origin，语法为 git remote add [shortname] [url]

git remote -v # 查看远程仓库

git remote set-url origin git@ github.com:robbin/robbin\_site.git # 修改远程仓库地址

git remote rm <repository> # 删除远程仓库

**工作流管理**

git add . # 添加当前修改的文件到暂存区

git add -u # 如果你自动追踪文件，包括你已经手动删除的，状态为Deleted的文件

git commit –m "你的注释" # 提交你的修改

git push origin master # 推送你的更新到远程服务器,语法为 git push [远程名] [本地分支]:[远程分支]

git status # 查看文件状态

git add readme.txt # 跟踪新文件

git rm readme.txt # 从当前跟踪列表移除文件，并完全删除

git rm –cached readme.txt # 仅在暂存区删除，保留文件在当前目录，不再跟踪

git mv reademe.txt readme # 重命名文件

git log # 查看提交的历史记录

git commit --amend # 修改最后一次提交注释的，利用–amend参数

git reset HEAD b # 已使用git add .，将修改过的文件a、b加到暂存区，现只想提交a，不想提交b

git checkout –- readme.txt # 取消对文件的修改

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

**分支管理**

git br -r # 查看远程分支

git br <n\_br> # 创建新的分支

git br -v # 查看各个分支最后提交信息

git merge <br> # 将br分支合并到当前分支

git br --merged # 查看已经被合并到当前分支的分支

git br --no-merged # 查看尚未被合并到当前分支的分支

git co <br> # 切换到某个分支

git co -b <n\_br> # 创建新的分支，并且切换过去

git co -b <n\_br> <brh> # 基于br创建新的n\_br

git co $id # 把某次历史提交记录co出来，无分支信息，切换分支会自动删除

git co $id -b <n\_br> # 把某次历史提交记录co出来，创建成一个分支

git br -d <br> # 删除某个分支

git br -D <br> # 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)

**远程分支**

git pull # 拉取远程仓库所有分支更新并合并到本地

git pull --no-ff # 拉取远程仓库所有分支更新并合并到本地，不快进合并

git fetch or # 拉取远程仓库更新

git merge or/master # 将远程主分支合并到本地当前分支

git co --track or/br # 跟踪某个远程分支创建相应的本地分支

git push # push所有分支

git push or master # 将本地主分支推到远程主分支

git push -u or master # 同上(如无远程主分支则创建，用于初始化远程仓库)

git push origin <lo\_br> # 创建远程分支， origin是远程仓库名

git push origin :<re\_br> # 先删除本地分支(git br -d <br>)，后再push删除远程分支

**补丁管理**

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