

**专有名词**

workspace：工作区

index/stage：暂存区

repository：仓库区（或本地仓库）

remote：远程仓库

**安装git**

yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel

<http://git-scm.com/download>

tar -xf git-1.7.2.2.tar.gz

cd git-1.7.2.2

./configure

make prefix=/usr/local all

make install

在linux安装：yum -y install git

**初次运行前配置**

/etc/gitconfig文件：系统中对所有用户都普遍适用的配置。若使用git config时用--system选项，就是读这个文件

~/.gitconfig文件：用户目录下的配置文件只适用于该用户。若使用git config时用--global选项，读写的就是这个文件

当前项目的git目录中的配置文件（也就是工作目录中的.git/config文件）：这里的配置仅仅针对当前项目有效。每一个级别都会覆盖上层的相同配置

用户信息：

git config --global user.name “aa”

git config --global user.email [aa@qq.com](mailto:aa@qq.com)

文本编辑器：

差异分析工具：

查看配置信息：

git config --list

**获取帮助**

git help

**新建代码库**

在当前目录新建一个git仓库：git init

新建一个目录，将其初始化为git代码库：git init [project-name]

下载一个项目和它的整个代码历史：git clone [url]

**配置**

git的设置文件为.gitconfig，它可以在用户主目录下（全局配置），也可以在项目目录下（项目配置）。

显示当前的git配置：git config --list

编辑git配置文件：git config [--global] user.name “[name]”

git config [--global] user.email “[email address]”

**增加/删除文件**

添加指定文件到暂存区：git add [file1] [file2] ...

添加指定目录到暂存区，包括子目录：git add [dir]

添加当前目录的所有文件到暂存区：git add .

添加每个变化前，都会要求确认，对于同一个文件的多出变化，可以实现分次提交：git add -p

删除工作区文件爱你，并且将这次删除放入暂存区：git rm [fire1] [fire2]...

停止追踪指定文件，但该文件会保留在工作区：git rm --cached [file]

**代码提交**

提交暂存区到仓库区：git commit -m [message]

提交暂存区的指定文件到仓库区：git commit [file1] [file2] ... -m [message]

提交工作区自上次commit之后的变化，直接到仓库区：git commit -a

提交时显示所有diff信息：git commit -v

使用一次新的commit，替代上一次提交，如果代码没有任何新的变化，则用来改写上一次commit的提交信息：git commit --amend -m [message]

重做上一次commit，并包括指定文件的新变化：git commit --amend [file1] [file2] ...

**分支**

列出本地所有分支：git branch

列出所有远程分支：git branch -r

列出所有本地分支和远程分支：git branch -a

新建一个分支，但依然停留在当前分支：git branch [branch-name]

新建一个分支，并切换到该分支：git checkout -b [branch]

新建一个分支，指定指定commit：git branch [branch] [commit]

新建一个分支，与指定的远程分支建立追踪关系：git branch --track [branch] [remote-branch]

切换到指定分支，并更新工作去：git checkout [branch-name]

切换到上一个分类：git checkout -

建立追踪关系，在现有分支与指定的远程分支之间：git branch --set-upstream [branch] [remote-branch]

合并指定分支到当前分支：git merge [branch]

选择一个commit，合并进当前分支：git cherry0pick [commit]

删除分支：git branch -d [branch-name

删除远程分支：git push origin --delete [branch-name]

git branch -dr [remote/branch]

**标签**

列出所有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]

删除本地tag：git tag -d [tag]

删除远程tag：git push origin :refs/tags/[tagname]

查看tag信息：git show [tag]

提交指定tag：git push [remote] [tag]

提交所有tag：git push [remote] --tags

新建一个分支，指向某个tag：git checkout -b [branch] [tag]

**查看信息**

显示所有变更文件：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

搜索提交历史，根据关键词：git log -S [keyword]

显示某个commit之后的所有变动，每个commit占据一行：git log [tag] HEAD --pretty=format:%s

显示某个commit之后的所有变动，其“提交说明”必须符合搜索条件：git log [tag] HEAD --grep feature

显示某个文件的历史版本，包括文件改名：git log --follow [file]

git whatchanged [file]

显示指定文件相关的每一次diff：git log -p [file]

显示过去5次提交：git log -5 --pretty --oneline

显示所有提交过的用户，按提交次数排序：git shortlog -sn

显示指定文件是什么人在什么时间修改过：git blame [file]

显示暂存区和工作区的代码差异：git diff

显示暂存区和上一个commit的差异：git diff HEAD

显示两次提交之间的差异：git diff [first-branch]...[second-branch]

显示今天写了多少行代码：git diff --shortstat “@{0 day ago}”

显示某次提交的元数据和内容变化：git show [commit]

显示某次提交时，某个文件的内容：git show [commit]:[filename]

显示当前分支的最近几次提交：git reflog

从本地master拉取代码更新当前分支：branch一般为master：git rebase [branch]

**远程同步**

更新远程仓库：git remote update

下载远程仓库的所有变动：git fetch [remote]

显示所有远程仓库：git remote -v

显示某个远程仓库的信息：git remote show [remote]

增加一个新的远程仓库，并命名：git remote add [shortname] [url]

取回远程仓库的变化，并与本地分支合并：git pull [remote] [branch]

上传本地指定分支到远程仓库：git push [remote] [branch]

强行推送当前分支到远程仓库，即使有冲突：git push [remote] --force

推送所有分支到远程仓库：git push [remote] --all

**撤销**

恢复暂存区的指定文件到工作区：git checkout [file]

恢复某个commit的指定文件到暂存区和工作区：git checkout [commit] [file]

恢复暂存区的所有文件到工作区：git checkout .

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，遇上一次commit保持一致：git reset --hard

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]

重置当前HEAD为指定commit，但保持暂存区和工作区不变：git rest --keeo [commit]

新建一个commit，用来撤销指定的commit，后者的所有变化都将被前者抵消，并且应用到当前分支：git revert [commit]

暂时将未提交的变化移除，稍后再移入：git stash

**多人合作**

gir clone

git branch [分支名] 创建分支

git branch 查看本地所有分支

git checkout [分支名称] 切换分支

---写代码---

git status（查看文件改变记录）

git diff（查看代码改变）

git add（添加）

git commit -m提交注释（提交当前分支到工作区）

git push [远程分支：origin] [本地分支名称]

去Git官网创建merge request

等待合并

----管理员合并所有人的merge quesy---

checkout master（切换至master）

git pull（从远程master更新至本地master）

checkout [branch]（切换至本地分区）

git rebase master（从本地master更新当前分支）

---是否有冲突---

---有---

---如何解决---

1. 操作代码文件并解决冲突
2. git add .加入待提交
3. git rebase --continue

---如果仍有冲突，重复1、2、3步

1. git rebase --skip
2. git push -f origin [branch] 强推

---去网站重新创建merge request---

结束，等待合并，重复上述对应步骤

**其他**

生成一个可发布的压缩包：git archive

**下载代码**

git stash

git pull --rebase origin master

**上传大文件**

git lfs install

git lfs track "\*.pdf"

git add .

git commit -m “add”

git push