Git的主要区域如下，如图所示。

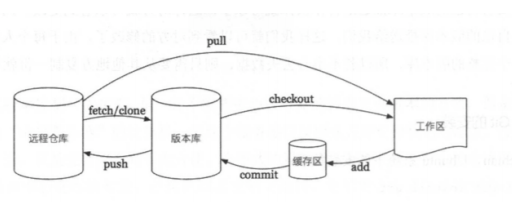
· ⼯作区( Workspace): 在计算机 中能看到的 ⽬录，它持有实际⽂件。

· 缓存区( Index/Stage):临时保存我们 的改动。

· 版本库( Repository): ⼯作区有⼀个隐藏⽬录 .git，是 Git 的版本库。

· 远程仓库( Remote): 托管在因特⽹ 或其他 ⽹络中的项⽬的版本库，可

供多⼈分布式开发 。



(1) Git配置项:

#显示当前的 Git 配置

$ git config --list git config -l

#查看全局配置信息

$ git config --global --list或者git config --global -l

#编辑 Git 配置⽂件

$ git config -e --global

#配置全局信息：用户名和邮箱

$ git config --global user.name ‘[name]’

$ git config --global user.email ‘[email address]’

清除命令：clear

(2)新建仓库:

#在当前⽬录新建⼀个 Git 仓库

$ git init

//win7 工具，文件夹选项，查看，显示隐藏的文件

#新建 ⼀个⽬录，将其初始化为 Git 仓库

$ git init [project-name]

//=>会生成一个隐藏文件夹“.git”(这个文件夹不要删)，因为暂存区和历史区还有一些其他的信息都在这里，删了就不是一个完整的git仓库。

#克隆 ⼀个仓库

$ git clone [url]

#查看工作区状态

$ git status （红色代表在工作区，绿色代表在暂存区，看不见东西证明所有修改的信息都已经提交到历史区）

(3)增加、删除⽂件:

#添加指定⽂件到暂存区

$ git add [filel] [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缓存区

$ git add .

#删除⼯作区⽂件，并且将这次删除放⼊缓存区

$ git rm [filel] [file2]

#改名⽂件，并且将这个改名放⼊缓存区

$ git mv [file-original] [file-renamed]

(4)提交⽂件:

#提交缓存区到仓库区

$ git commit –m “描述信息：本次提交内容的描述”

#提交缓存区的指定⽂件到仓库区

$ git commit [file1] [file2] ... –m “信息”

$git log 查看提交记录

$git reflog 查看所有的历史记录（包括历史区回滚后）

从工作区提交到暂存区，从暂存区提交到历史区：是把内容复制一份传过去的，本区域中依然存在这些信息（只有这样才能对比出哪些文件在哪个区）

------------------------------------------------------------------------------------

推到远程：

$ssh-keygen -t rsa -C "3231063918@qq.com"

连续三次回车,找到

Your public key has been saved in /c/Users/Administrator/.ssh/id\_rsa.pub. 用笔记本打开，复制信息，用于在Github 里面创建项目SSH密钥

复制SSH协议，用在输入命令时候粘贴

$git remote add orign

<https://github.com/13723415509/Learning_FrontEnd.git>

(//项目SSH协议)

$git push origin master (//推上GitHub)

长期储存推送至Gittub的命令语句

git config --global credntial.helper store

------------------------------------------------------------------------------------

克隆GitHub到本地

$ssh-keygen –t rsa –C “邮箱地址”

$git clone<https://github.com/13723415509/Learning_FrontEnd.git>

(//项目SSH协议)

$git push orign master

#提交⼯作区⾃上次 commit 之后的变化，直接提交到仓库区

$ git commit -a

#提交时显示所有 diff 的信息

$ 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变化，则⽤来改写上⼀次 commit 的提交信息

$ git commit --amend -m [message]

(5) Git分⽀:

#列出所有本地分⽀

$ 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

$ g it 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，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，并更新⼯作区

$ git checkout [branch-name)

#切换到上⼀个分⽀

$ 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

$ g it merge [branch]

#选择⼀个 commit，合并在当前分⽀

$ git cherry-pick [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)

(6) Git的标签:

#列出所有 tag

$ git tag

#在当前 commit 上新建⼀个 tag

$ git tag [tag]

#在指定 commit 上新建 ⼀个 tag

$ git tag [tag] [commit)

#删除本地 tag

$ git tag -d [tag]

#删除远程 tag

$ git push origin

#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]

(7)查看信息 :

#显示有变更的⽂件

$ git status

#显示当前分⽀的版本历史

$ git log

#根据关键词搜索提交历史

$ git log -S [keyword]

#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差异

$ git diff

#显示暂存区和上⼀个 commit 的差异

$ git diff --cached [file]

#显示⼯作区与当前分⽀最新 commit 之间的差异

$ git diff HEAD

#显示两次提交之间的差异

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

#显示某次提交的元数据和内容变化

$ git show [commt]

#显示某次提交发⽣变化的⽂件

$ git show --name-only [commit]

(8)远程同步 :

#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

(9)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，与上⼀次 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，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]

#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]

#新建⼀个 commit，⽤来撤销指定 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

$ git stash pop