# Git配置

* Git --version 查看git版本号
* Git config 配置或读取环境变量
* Git config --system user.name xxx

系统所有用户默认配置，无以下两个默认用这个，在/etc/gitconfig文件中查看

* Git config --global user.name xxx

本用户的默认配置，无以下一个默认用这个，在~/.gitconfig文件中查看

* Git config user.name xxx

当前项目git目录中默认配置，优先使用这个，在.git/config中查看

* 常用需要配置的选项：

Git config --global user.name xxx 用户名

Git config --global user.email xxx 用户邮件地址

Git config --global core.editor vim 系统默认编辑器

Git config --list 查看默认配置

# Git使用

* Git init 在当前目录创建git仓库，生成.git目录
* Git status 查看当前目录的文件状态，包含：
* 已提交到git仓库的
* 位于工作区未track（追踪）的
* 位于工作区已追踪待提交的
* Git add和git rm在git init命令中有介绍
* Git commit
* Git commit xxx 将已track待提交的文件提交，此命令会进vi编辑提交信息，
* git commit -a 提交所有已stage的文件，
* git commit -m”ooo” xxx其中-m”ooo”参数可以直接在””中输入提交信息，不用进入vi编辑器
* Git log
* Git log完整显示版本信息
* Git log --pretty=oneline 简洁一点显示（放在一行显示）
* Git log --oneline 比上面更简洁显示（缩短了哈希值）
* Git reflog 在oneline基础上多现实了到某一个版本需要几步，且能显示之前之后的所有版本（显示的最全）
* Git reset
* Git reset --hard 824921c 跳转到缩短哈希值82..对应的版本
* Git reset --hard HEAD^^ 跳转到所在版本的上两个版本（取决于^个数）
* Git reset --hard HEAD~2 等价于上面，直接用数值表示比用^个数表示方便

注：以上两个只能后退，第一个跳转可以往前可以往后

Git reset --hard xxx 仅仅在本地库移动HEAD指针

Git reset --mixed xxx 移动本地库HEAD指针并重置暂存区

Git reset --hard xxx 本地库移动HEAD指针，充值暂存区，重置工作区

* Git diff用于比较文件差异，具体百度
* Git branch分支操作
* Git branch -v 查看分支
* Git branch hot\_fix 创建hot\_fix分支
* Git checkout hot\_fix 切换到hot\_fix分支
* Git merge hot\_fix 将hot\_fix分支中的修改同步到本分支中（现在已经切换到master）
* 远程库操作

Git remote add mygithub <http://xxxxxx> 将远程库链接设置为mygithub，方便调用

Git remote -v 显示已经设置好的远程库链接，也可在config中查看，跟name一起

Git remote remove mygithub 删除设置好的mygithub

Git push mygithub master 将本地库的master分支推送到mygithub对应的远程库

Git clone <http://xxxx> 将远程库克隆到本地目录

Git fetch mygithub master 将mygithub远程库的master分支抓取到本地但不同步

Git checkout mygithub/master 切换到抓取到的mygithub，master分支

Git checkout master 切换到本地库master分支

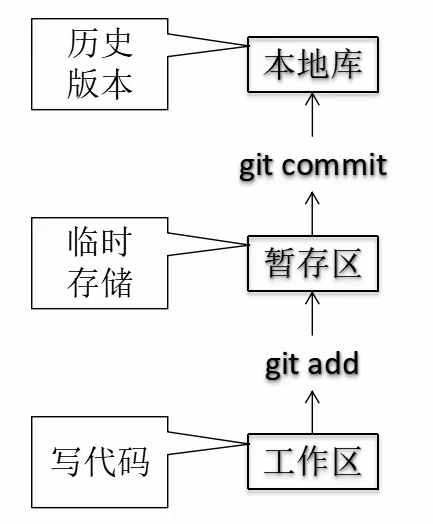
以上两步用于对比文件信息，考虑要不要合并

Git merge mygithub/master 将抓取到的mygithub/master合并到本地库master库

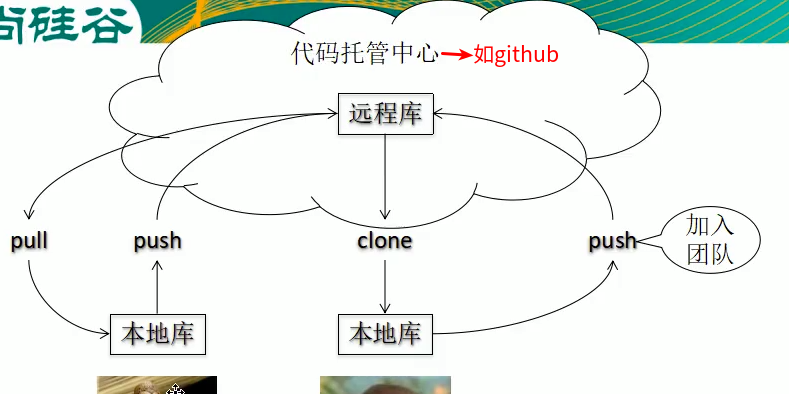
git pull mygithub master 此命令相当于前面的先fetch再merge，一步达成同步

# Git原理

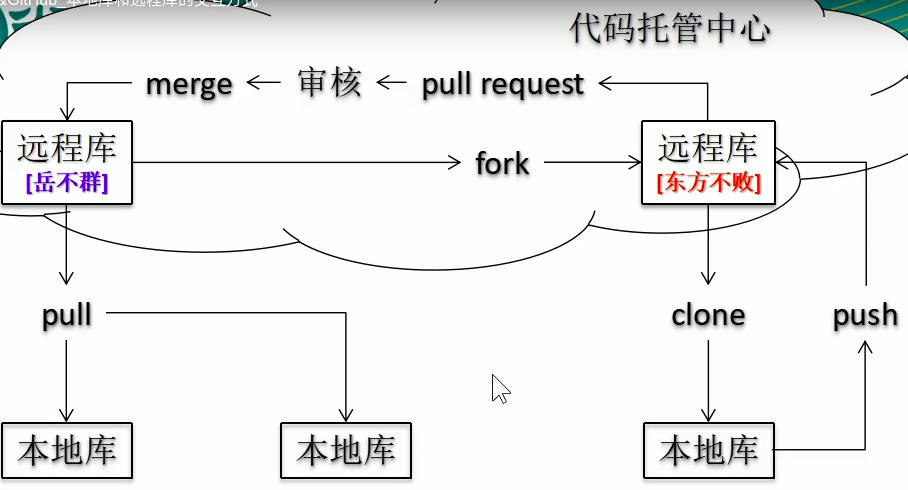
* Git在本地结构



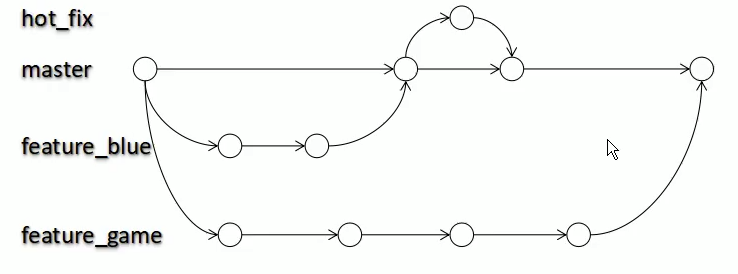
* 本地库和远程库之间团队内部协作



* 本地库和远程库跨团队协作



* Git分支管理



并行开发项目，各分支开发过程中失败对主分支无影响

当两条分支中的同一个位置都更改了，然后将两分支同步，就会产生冲突