Git

1.公司如何基于git做的协同开发?

```
1. 初始化项目: git clone 仓库地址(生成仓库文件)
   如果想重命名: git clone url  name(项目名)
2. git pull origin master(dev...)  合并别人推送的新内容
3. rebase可以帮助将提交的记录整成一条直线(可以不做)
4. 创建自己的开发分支，在自己的分支进行开发
5. 将需要和别人协作的内容推送至公开分支，
   git push origin serverfix
   协作者: git fetch origin
        抓取远程跟踪分支，本地不会有这个分支，相当于指针
        git merge origin /serverfix ,合并内容到自己的分支
        git checkout -b serverfix origin/serverfix
        创建并切换到分支，起点位于origin / serverfix
6. 跟踪分支
   git checkout --track origin/serverfix
   跟踪该分支，git pull自动识别这个分支进行合并
   重命本地分支: git checkout -b sf origin/serverfix
7. git fetch 从服务器拉取本地没有的数据
8. 删除远程分支
   git push origin --deletc serverfix

9.远程仓库托管网站:  github，码云 gitlab
```

2. git 常见命令

```
1:git init---------初始化
2:git add .---------从工作区，添加到版本库
3:git commit -m"xxx"----从暂存区，添加到分支
4:git status--------查看状态
5:git log ----------查看版本库的日志
6:git reflog--------查看所有日志
7:git reset –head 版本号--切换
8:git stash---------保存
9:git stash---------将第一个记录从“某个地方”重新拿到工作区（可能有冲突）
git stash list-------------------------查看“某个地方”存储的所有记录
git stash clear-------------------------清空“某个地方”
git stash pop-------------------------将第一个记录从“某个地方”重新拿到工作区（可能有冲突）
git stash apply -------------------------编号,将指定编号记录从“某个地方”重新拿到工作区（可能有冲突）
git stash drop -------------------------编号 ，删除指定编号的记录
10:git branch dev-----创建分支
11:git branch -d dev----删除分支
12:git checkout dev----切换分支
13:git merge dev------合并分支
14:git branch-------查看所有分支
15:git clone https:xxx---克隆
```

```
16:git add origin https:xxx-起个别名
17:git push origin dev ——添加到dev分支
18:git pull origin master—拉代码
19:git fetch origin master-去仓库获取
20:git merge origin/master-和网上下的master分支合并
# ------------------------------------协同开发----------------------------------
默认是master分支----------master
开发的分支-------------dev
做代码review-------------reciew
程序员自己的分支----------…….
1：每个员工创建自己的分支
2：将自己的代码提交的到自己的分支-----xxx,sss,wwww…….
3：由组长或老大做代码的review,-------代码提交的review分支
4：再提交到dev.
5： 再合并到master分支
```

3. stash的作用以及相关命令

```
'git stash'：将当前工作区所有修改过的内容存储到“某个地方”，将工作区还原到当前版本未修改过的状态
'git stash list'：查看“某个地方”存储的所有记录
'git stash clear'：清空“某个地方”
'git stash pop'：将第一个记录从“某个地方”重新拿到工作区（可能有冲突）
'git stash apply'：编号，将指定编号记录从“某个地方”重新拿到工作区（可能有冲突）
'git stash drop'：编号，删除指定编号的记录
```

4. merge和rebase的区别

```
merge:
会将不同分支的提交合并成一个新的节点，之前的提交分开显示，
注重历史信息、可以看出每个分支信息，基于时间点,遇到冲突,手动解决,再次提交
rebase:
将两个分支的提交结果融合成线性，不会产生新的节点；
注重开发过程，遇到冲突，手动解决，继续操作
```

5. 如何基于git实现代码review

```
review的人：  老板/小组长/领导

review的内容：  审查代码的规范，

流程：   master

    dev

    个人分支: san

            si

            review   分支

    个人提交代码到review分支，审查完毕之后由领导合并到dev分支
```

6. git如果实现v1.0 v2.0等的版本管理

```
git tag   //查看标签:
git tag -l "v1.8*"   //查询 1.8开头的所有版本
git tag -a v1.0  -m "one"    //创建标签
git show v1.0     //查看v1.0
git tag v1.1-lw    //轻量级的标签
git tag -a v1.2  "commitID"   给历史提交记录打标签
git push origin v1.5   推送到远程
git push origin --tags    //推送所有标签到远程
git tag -d v1.2    //删除标签
git push origin :refs/tags/v1.2    //删除远程仓库标签
步骤，给commit打好标签，再提交到远程仓库
```

7. 什么是gitlab

```
基于gitd的项目管理软件
代码托管的私有仓库，自动进行代码备份
gitlab是公司自己搭建的项目代码托管平台
```

8. github和gitlab的区别?

```
1、gitHub是一个面向开源及私有软件项目的托管平台
(创建私有的话，需要购买，最低级的付费为每月7刀，支持5个私有项目)
2、gitlab是公司自己搭建的项目托管平台


GitLab是可以部署到自己的服务器上，数据库等一切信息都掌握在自己手上，适合团队内部协作开发，你总不可能把团队内部的智慧总放在别人的服务器上吧？简单来说可把GitLab看作个人版的GitHub
```

9. 如果为github上牛逼的开源项目贡献代码

```
1、fork需要协作项目
2、克隆/关联fork的项目到本地
3、新建分支（branch）并检出（checkout）新分支
4、在新分支上完成代码开发
5、开发完成后将你的代码合并到master分支
6、添加原作者的仓库地址作为一个新的仓库地址
7、合并原作者的master分支到你自己的master分支,用于和作者仓库代码同步
8、push你的本地仓库到GitHub
9、在Github上提交 pull requests
10、等待管理员（你需要贡献的开源项目管理员）处理
```

10. git中.gitignore文件的作用

一般来说每个Git项目中都需要一个“.gitignore”文件，
这个文件的作用就是告诉Git哪些文件不需要添加到版本管理中。
实际项目中，很多文件都是不需要版本管理的，比如Python的.pyc文件和一些包含密码的配置文件等等。