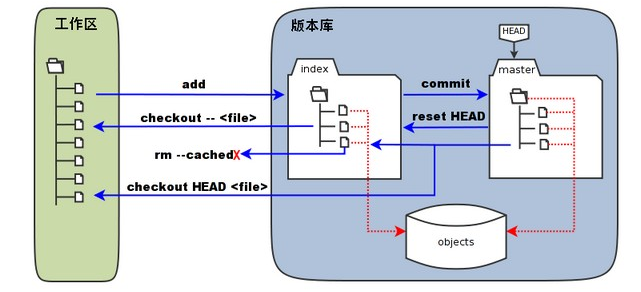
一、git工作区、缓存区和版本库

git是目前流行的分布式版本管理系统。它拥有两套版本库，本地库和远程库，在不进行合并和删除之类的操作时这两套版本库互不影响，也因此其近乎所有的操作都是本地执行。

* **工作区：**就是你在电脑里能看到的目录。
* **缓存区：**一般存放在 ".git目录下" 下的index文件（.git/index）中，缓存区有时也叫作索引（index）。
* **版本库：**本地代码仓库。

工作区、版本库中的缓存去以及版本库之间的关系：



当对工作区修改（或新增）的文件执行 "git add" 命令时，缓存区的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的ID被记录在缓存区的文件索引中。

当执行提交操作（git commit）时，缓存区的目录树写到版本库（对象库）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时缓存区的目录树。

当执行 "git reset HEAD" 命令时，缓存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。

当执行 "git rm --cached <file>" 命令时，会直接从缓存区删除文件，工作区则不做出改变。

当执行 "git checkout ." 或者 "git checkout -- <file>" 命令时，会用缓存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到缓存区的改动。

当执行 "git checkout HEAD ." 或者 "git checkout HEAD <file>" 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换缓存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除缓存区中未提交的改动。

二、git操作流程：

1. 首先利用命令把远程仓库的代码克隆到本地：git clone xxx.git，该仓库包含有多个分支。
2. 使用命令git branch可以查看本地分支，利用命令 git checkout xx可以切换分支开发，进行添加或者修改代码。
3. 如果是增加了文件，需使用命令git add更新缓存区的代码。
4. 利用命令git commit -am ‘description’把缓存区增加或者改变的代码更新本地仓库中。
5. 再利用命令：比如git push origin ztc-goliah\_1-0-0\_BRANCH:refs/for/ztc-goliah\_1-0-0\_BRANCH 将本地仓库中的代码提交到远程仓库。用冒号分隔本地分支名称和远程分支名称。
6. 如果有其他人修改了远程仓库中的代码，可以使用命令git pull来拉代码。（注意如果此时项目中的代码已经修改了，必须先使用git commit 命令向本地仓库提交已经修改的代码。）
7. 删除本地分支git branch -D xxx; 删除本地的远程分支：git branch -r -D origin/xxx
8. git pull和git fetch的共同点在于都可以从远程仓库来更新本地。其区别简单来说是：从远程获取最新版本到本地时，git pull会自动和本地分支merge,而git fetch不会。所以，如果需要有选择的合并，git fetch是更好的选择。
9. origin/HEAD就像一个指针，表示默认分支，指向origin/master，那么origin/master是默认分支。

origin/HEAD -> origin/master

10. git push -u origin master，用于在github上创建仓库后，把项目放到本地仓库后，向github推送本地项目。

11. git push origin branch2，向远程分支（和当前分支name一样）提交。

12. git push origin branch2:branch,向远程分支（和当前分支name不一样）提交。

13. refs/for/mybranch需要经过code review之后才可以提交；而refs/heads/mybranch不需要code review。

三、常用git操作

向远程分支提交本地分支的修改

git push origin ztc-nadab\_1-0-2-1\_BRANCH:refs/for/ztc-nadab\_1-0-2-1\_BRANCH

从本地仓库恢复工作区的文件，即是还原修改。

git checkout xxx

还原当前目录的所有更改

git checkout .

拉取远程分支并创建本地分支：

git checkout origin/ztc-goliah\_1-0-2\_BRANCH -b ztc-goliah\_1-0-2\_BRANCH

（远程分支） （本地分支）

git pull origin ztc-goliah\_1-0-2\_BRANCH:ztc-goliah\_1-0-2\_BRANCH

建立本地分支和远程分支的关联联系：

git branch --set-upstream-to=origin/ztc-nadab\_1-0-3\_BRANCH ztc-nadab\_1-0-3\_BRANCH

查看本地分支和远程分支的关联关系

git branch -vv

创建一个和当前分支内容一样的新分支

git checkout -b ztc-nadab\_1-0-2-1\_BRANCH

向远程推送一个和本地分支同名的分支

（本地分支） （远程分支）

git push origin ztc-nadab\_1-0-2-1\_BRANCH:ztc-nadab\_1-0-2-1\_BRANCH

删除远程分支

git push origin --delete branch\_2\_0