**工作区暂存区的理解：**

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除都可以“一键还原”。

分支管理 。你创建了一个属于你自己的分支，别人看不到

首先，我们创建dev分支，然后切换到dev分支：

$ git checkout -b dev

git merge命令用于合并指定分支到当前分支。

初始化仓库：$git init

把文件添加到仓库 $git add new.txt

提交 $git commit -m “说明内容”

查看状态 $git status

查看修改 $git diff new.txt

查看日志 $git log

回退到上一个版本 $git reset –hard HEAD^

回退到指定ID$git reset –hard 33434534 (3344534为commit id )

把文件在工作区的修改撤销掉 $git checkout – new.txt

把文件在暂存区的修改撤销掉,重新放回工作区 $git reset HEAD new.txt

删除文件 $git rm new.txt

**与GitHub连接**

1.创建SSH Key$ ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)

2.关联本地仓库

$ git remote add origin [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git)

3.把本地库的所有内容推送到远程库上：$ git push -u origin master

4.提交到GitHub上 $ git push origin master

***分支管理***

创建+切换分支 $git checkout -b dev -b表示创建并切换到dev

创建$git branch dev 切换$git checkout dev

查看当前分支(列出所有分支) $git branch

合并指定分支到当前分支 $ git merge dev

删除分支 $git branch -d dev

解决冲突：当分支与主分支内容不同不能快速合并时发生，打开冲突文件回看到两个版本之间的区别，修改后提交即可。

**分支管理策略**

Bug分支: 修复bug时，我们会通过创建新的bug分进行修复，然后合并，最后删除；当手头工作没有完成时，先把工作现场git stash⼀下，然后去修复bug，修复后，再回到工作现场。

Feature分支

$git stash 把当前工作现场“储藏”起来，等以后恢复现场后继续工作

查看stash $git stash list

恢复stash到工作区 $git stash apply $git stash drop

$git stash pop

**标签**：相当于版本库的快照，实际上是一个指向某个commit的指针，与分支的区别是分支可以移动，标签不能移动。标签指存在与本地，不会自动推送到远程。

打标签 $git tag v1.0 或 $git tag v1.0 commitid

查看所有标签 $git tag

查看某个标签的详细信息 $git show v1.0

删除标签 $git tag -d v1.0

推送标签到远程 $git pudh origin v1.0

**多人协作**

推送分支 $git push orign master

$git branch –set-upstream dev origin/dev

$git pull

那么，哪些分支需要推送，哪些不需要呢？

• master分支是主分支，因此要时刻与远程同步；

• dev分支是开发分支，团队所有成员都需要在上面工作，所以也需要与远程同步；

• bug分支只用于在本地修复bug，就没必要推到远程了，除非老板要看看你每周到底修复了几个bug；

• feature分支是否推到远程，取决于你是否和你的小伙伴合作在上面开发。

因此，多人协作的工作模式通常是这样：

1. 首先，可以试图用git push origin branch-name推送自己的修改；

2. 如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；

3. 如果合并有冲突，则解决冲突，并在本地提交；

4. 没有冲突或者解决掉冲突后，再用git push origin branch-name推送就能成功！

如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没

有创建，用命令git branch --set-upstream branch-name origin/branch-name。