常用命令

* 查询当前状态：
  + git status 文件（可选，不写则查询当前目录下所有的文件状态）
* 增加一个文件到git版本：
  + git add index.php
  + git commit -m “增加一个index.php的文件”
* 查询文件哪里被修改了
  + git diff 文件（可选，不写则查询当前目录下所有的文件状态）
* 查询提交的记录log
  + 查看所有log：git log
  + 查看最近一次、上一次、最早的一次（用的会比较多）：git log --pretty=oneline
* 版本回退
  + git reset --hard HEAD^ （一个^表示上一个版本，多个就是一次类推，实在太多就使用HEAD~100）
* 版本前退：
  + git reset --hard 1094a （只能通过版本库的id号前退，id不必全不写全）
* 查询所有的commit和reset记录：
  + reflog：查询所有分支的所有操作记录（包括commit和reset的操作）包括已经被删除的commit记录，git log则不能察看已经删除了的commit记录

git操作过程

1. git clone：从裸服务器将版本库克隆至本地
2. git pull：先从远程仓库拉取最新版本
3. git status：先查询下所有状态
4. git diff：在提交前要查看是否是本次所修改的地方
5. git add index1.php index2.php：添加文件，注意\*（如果修改了多个文件但是只想提交其中几个）必须要在执行一次命令，再提交  
   当这个文件是被修改的文件时，如果想省略这一步，除非是只有一个文件的修改，或者是在提交的时候提交多个文件
6. 提交本次修改且要提交的文件，git commit –m “this is test”

分支操作

1. git init --bare：创建一个git裸服务器
2. git branch：查看分支（前边带有\*号的证明当前在此分支上）
3. git branch dev：创建一个分支
4. git branch -d dev：删除分支
5. git branch --D dev：删除一个还没有合并的分支
6. git checkout dev：切换到某个分支
7. git checkout --b dev：创建dev分支，并切换到dev分支（相当于2和3的组合）
8. git merge dev：合并dev分支到当前分支
9. git checkout master + git merge dev：切换到master分支，合并dev分支，注意必须切换到主分支（不一定是master，只要是上层分支就可以叫做要合并的分支的主分支）后才能合并分支
10. git log --graph --pretty=oneline --abbrev-commit：用带参数的git log可以查看分支的合并情况
11. git remote + git push origin maste：推送版本至服务器

bug分支

当遇到紧急bug修复时手中还有未完成工作：

1. git stash list：查询工作现场列表
2. git stash：保存当前工作现场
3. 然后去修复bug
4. git stash pop（git stash apply和git stash drop的缩写）：恢复上次保存的工作现场并把stash内容也删除

多人协作

1. git remote -v：查看远程仓库的信息（-v为可选，显示更详细的信息）
2. git push origin master：推送分支，就是把该分支上的所有本地提交推送到远程仓库  
   如果要推送其他分支比如dev就改成：git push origin dev

因此，多人协作的工作模式通常是这样：

1. 本地新建的分支如果不推送到远程，对其他人就是不可见的；（本地和远程分支的名称最好一致）
2. 首先，可以试图用git push origin <branch-name>推送自己的修改；
3. 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；
4. 如果合并有冲突，则解决冲突，并在本地提交；
5. 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！
6. 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-to <branch-name> origin/<branch-name>。这就是多人协作的工作模式，一旦熟悉了，就非常简单。

标签（版本号）

1. git tag v1.0：添加一个标签到当前分支
2. git tag：查询所有标签
3. 默认标签是打在最新提交的commit上的。有时候，如果忘了打标签，比如，现在已经是周五了，但应该在周一打的标签没有打，怎么办？方法是找到历史提交的commit id，然后打上就可以了
4. git show v1.0：查看标签信息
5. 创建带有说明的标签，用-a指定标签名，-m指定说明文字：git tag -a v0.1 -m "version 0.1 released" 1094adb  
   注意：标签总是和某个commit挂钩。如果这个commit既出现在master分支，又出现在dev分支，那么在这两个分支上都可以看到这个标签。
6. git tag --d v1.0：删除标签
7. git push origin v1.0：推送某个标签到远程
8. git push origin --tags：一次性推送全部标签到远程
9. 如果标签已经推送到远程，要删除远程标签则需两部：  
   git tag –d v1.0：先删除本地标签  
   git push origin :refs/tags/v1.0：再删除远程标签

工作区、暂存区、分支

1. 工作区：就是你本地的实实在在的文件，比如你修改了一个文件，在没有git add 之前一直属于工作区
2. 暂存区（stage）：当这个被修改的文件被git add后就提交到了暂存区
3. 分支（可能是主分支，也可能是其他子分支）：当使用git commit 提交后就是把暂存区的东西提交到了分支中了  
   