Learn GIT

git add fileName 将修改放入暂存区

git add .

git commit -m “备注” 提交暂存区的修改到分支

git log 查看提交历史记录

git log —pretty=oneline 查看提交历史记录显示在一行

git reset —hard HEAD^ 退回到上一个版本

git reset —hard HEAD^^ 退回到上上个版本

git reset —hard HEAD~100 往上100个版本

git reset —hard <commit id> 指定文件回到某个版本 commit id 为当时提交的版本号 没有必要写全前几位就好，GIt会自动去找

git reflog 记录了你的每次命令

cat fileName 查看当前文件内容

现在总结一下：

* HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。
* 穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。
* 要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

git status 查看commit之前那些文件被修改了

git diff HEAD — fileNAme 查看工作区和版本库里面最新版本的区别

每次修改，如果不add到暂存区，那就不会加入到commit中。

git checkout — fileName 让这个文件回到最近一次git commit或git add时的状态 丢弃工作区的修改

git reset HEAD fileName 把暂存区的修改撤销，重新放回工作区

git rm fileName 删除文件

git checkout — fileName 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”。

git pus -u origin master 将本地库的所有内容推送到远程库上 userName:utali password:github1415

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令

git push origin master

git clone  仓库地址（gihub）

要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆

git checkout -b dev 创建并切换分支，相当于下面两条命令(创建了一个dev的分支)

git branch dev

git checkout dev

git branch 查看当前分支

dev分支的工作完成，可以切换到master分支（需要add和commit）

git checkout master 切换回master分支

git merge dev把dev分支的工作成果合并到master分支上

合并完成后可以删除dev分支 git branch -d dev  
Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

用git log --graph命令可以看到分支合并图。

git merge --no-ff -m "merge with no-ff" dev    --no-ff参数，表示禁用Fast forward

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并

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

用git stash list命令查看刚才的工作区

工作现场还在，Git把stash内容存在某个地方了，但是需要恢复一下，有两个办法：

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

另一种方式是用git stash pop，恢复的同时把stash内容也删了

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git stash pop，回到工作现场

如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除

要查看远程库的信息，用git remote 或者，用git remote -v显示更详细的信息

* 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
* 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
* 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
* 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。
* 命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；
* git tag -a <tagname> -m "blablabla..."可以指定标签信息；
* git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；
* 命令git tag可以查看所有标签。
* 命令git push origin <tagname>可以推送一个本地标签；
* 命令git push origin --tags可以推送全部未推送过的本地标签；
* 命令git tag -d <tagname>可以删除一个本地标签；
* 命令git push origin :refs/tags/<tagname>可以删除一个远程标签。