git config -– global user.name “your name”

git config –- global user.email “[email@example.com](mailto:email@example.com)”

从本地提交一个全新的项目到github:

1.切换到项目根目录，创建本地仓库，git init （此时会对应创建.git目录，里面包含git版本控制信息）

2.将目录下所有文件添加到本地仓库： git add .

3.提交到本地仓库，添加注释：git commit -m ‘first commit’

4.添加远程仓库地址(需要先在github上创建好，复制地址到这里)，命名为origin:

git remote add origin <https://github.com/Xiaobai0419/minexiaobai0419.git>

5.推送到远程仓库： git push -u origin master

创建版本库：

git init: 把一个目录变成git可以管理的仓库

git status: 查看哪些文件修改过，但还没有提交

git add readme.txt 把文件添加到仓库

git commit -m ‘aaa’ 把文件提交到仓库

版本回退：

git log

418e787abc94d63a7af1315bfed0abea8077cf50 git的commit id

回退到上一个版本： git reset –-hard HEAD^

git reset –-hard commitid 回退到某个版本

git reflog

一旦commitid找不到了，无法回退到某个版本号了。可以使用git reflog:它记录了用户的每一次命令(查看历史命令)。

工作区和暂存区：

工作区：就是在电脑中能看到的目录

版本库(repository)：工作区有一个隐藏目录”.git”, 这个不算工作区，而是git的版本库。Git版本库中存放很多东西，其中最重要的就是stage(或者叫index)的暂存区，还有git为我们自动创建的第一个分支master,以及指向master的一个指针叫HEAD。

git add: 把文件添加到暂存区。 git commit: 把暂存区内的所有内容提交到当前分支。每次修改，如果不add到暂存区，那就不会加入到commit中。

撤销修改(git restore):撤销工作区文件和暂存区文件的修改

git status后发现可以使用命令git restore filename来撤销工作区文件的修改。

撤销暂存区的文件修改：根据git status的提示，先git restore –stage filename(从暂存区撤销修改)，再git restore filename(从工作区撤销修改)

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。

git reset HEAD filename

删除文件：

当不小心删除了暂存区中的一个文件时，需要恢复，此时可以用git restore filename

远程仓库:github

添加远程库:在本地创建了一个git仓库后，又在github上创建一个git仓库，并且让这两个仓库进行远程同步。把本地仓库的内容推送到github仓库。

git remote add origin <https://github.com/Xiaobai0419/minexiaobai0419.git>(本地和远程关联)

添加后，远程仓库的名字就是origin,这是git默认 的叫法，也可以改成别的。

下一步，就是把本地库的所有内容推送到远程库上：git push -u origin master(第一次推送master分支时，加-u参数)。Git 不仅把本地的master分支推送到远程新的master分支，还会把本地master分支和远程master分支关联起来，在以后推送或者拉取可以简化命令。

分支管理：

创建与合并分支：

创建分支: git checkout -b dev (创建dev分支) 或者git branch dev git checkout dev

创建分支: git branch dev

查看分支: git branch -a (本地和远程分支)

创建+切换分支: git checkout -b dev

合并某分支到当前分支: git merge dev (先切换到当前分支，在当前分支中git merge dev.把dev分支合并到当前分支)

删除分支: git branch -d dev 最后删除没用的分支

解决冲突：手动解决。当无法自动合并时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

用带参数的git log查看分支的合并情况：git log –-graph –-pretty=oneline (git log -–graph命令可以看到分支合并图)

分支管理策略：

在实际开发中，分支管理的基本原则：

1.master分支应该非常稳定，仅用来发布新版本，平时不能在上面干活。

2. 干活都在dev分支上，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本。

3.每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

Bug分支：每个bug都可以通过一个新的临时分支来修复，修复后，合并分支，然后将临时分支删除。

git stach功能：把现场保存起来，等以后恢复现场后继续工作。 使用git stash，工作区干净，此时可以创建分支来修复bug

bug分支管理：当手头工作没完成时，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。

在master分支上创建分支修复bug,首先需要切换到master分支上(git checkout master)。然后创建修复分支(git branch issue-101),切换到该分支上(git checkout issue-101)

当git stash后，使用git status发现工作区是干净的。此时, git stash list。发现：stash@{0}:WIP on ……。工作现场还在，git把stash内容存放起来了，需要恢复一下。使用git sash pop恢复现场(此时发现git stash list内容为空)。

Feature分支：

每添加一个新功能，最好新建一个feature分支，在上面开发，完成后，合并，最后，删除该feature分支。 如果要丢弃一个没被合并过的分支，可以通过git branch -D featurename强行删除。

多人协作：当从远程仓库克隆时，实际上git自动把本地的master分支和远程的master分支对应起来，并且，远程仓库的默认名称是origin。查看远程仓库的信息：git remote (git remote -v详细信息)

推送分支：