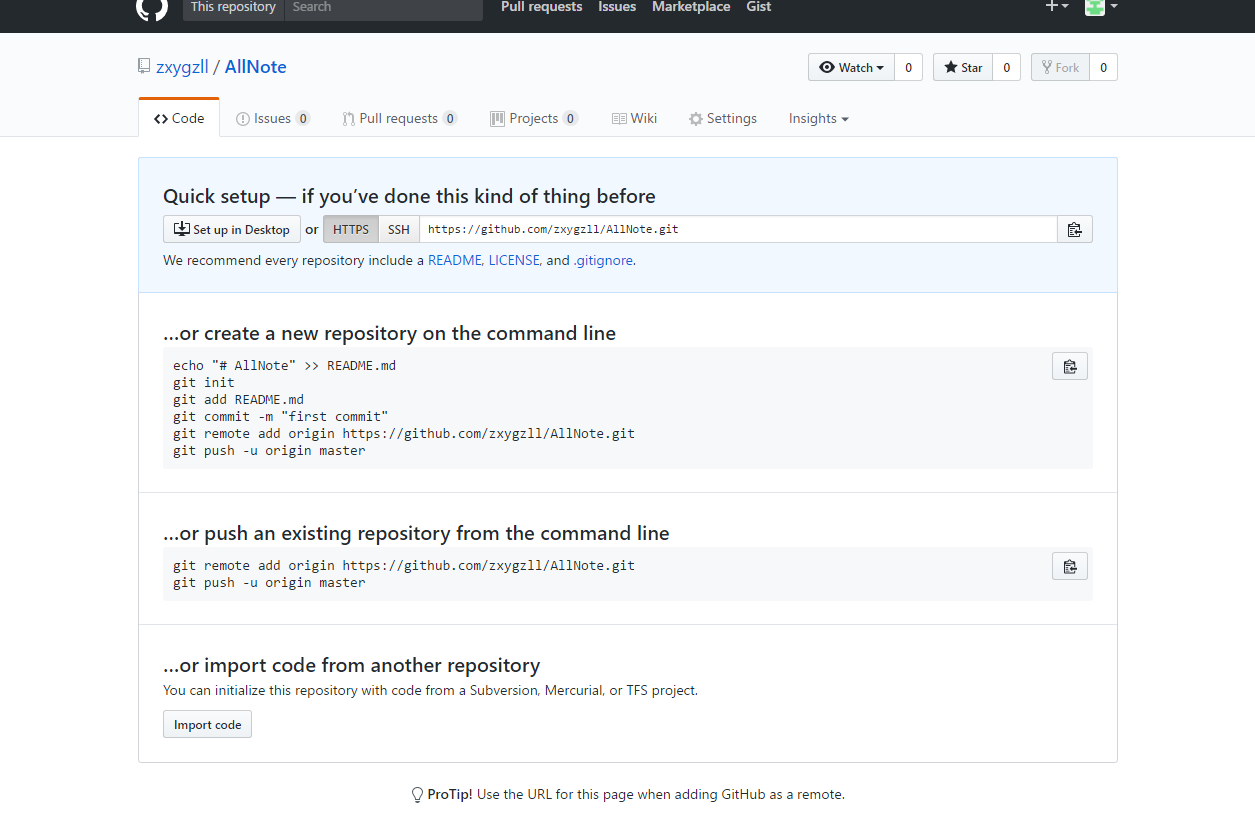
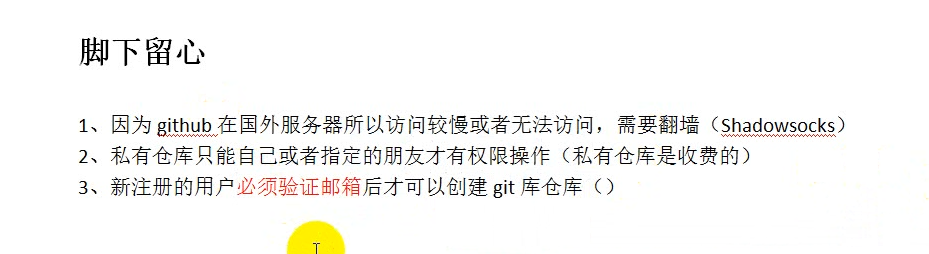
# Git项目管理工具

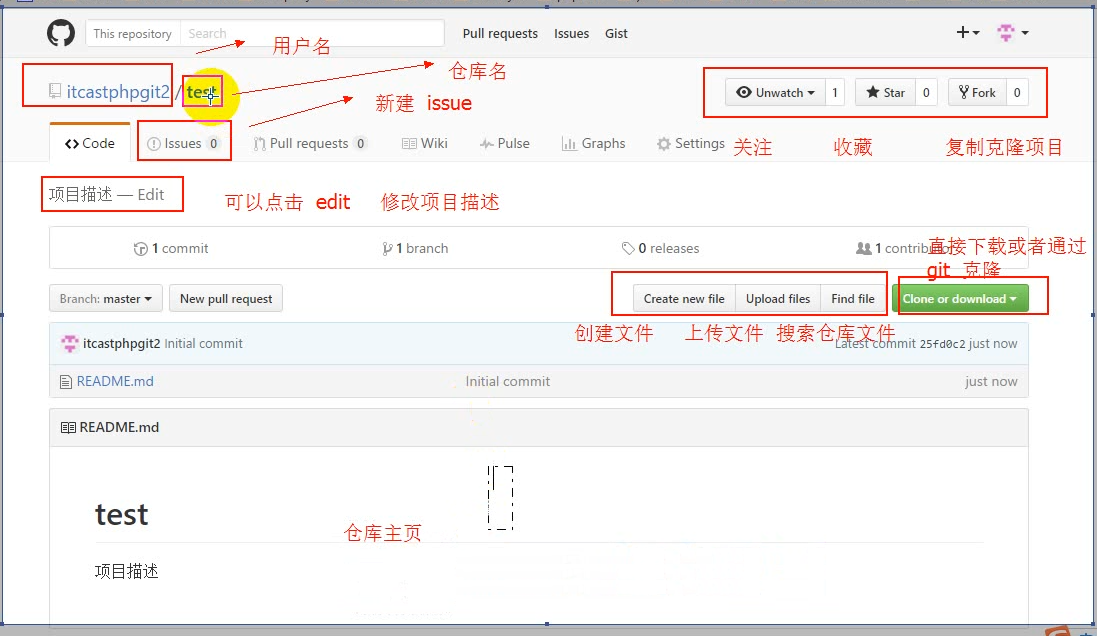




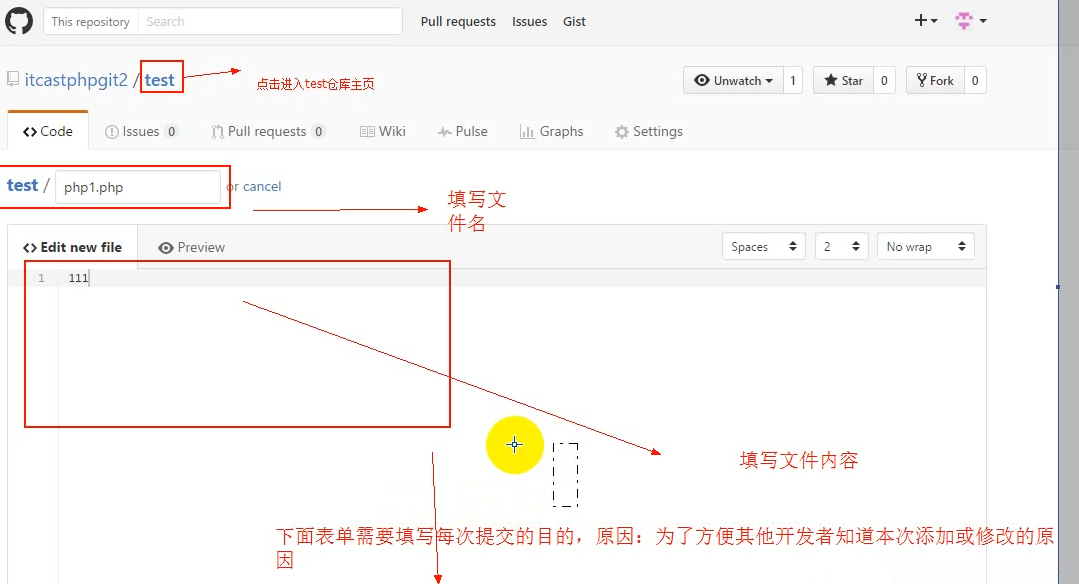
<https://github.com/>

sign in(登录) sign up(注册)，建立公共的仓库，别人能够访问，私有的要收钱，邮箱进行激活

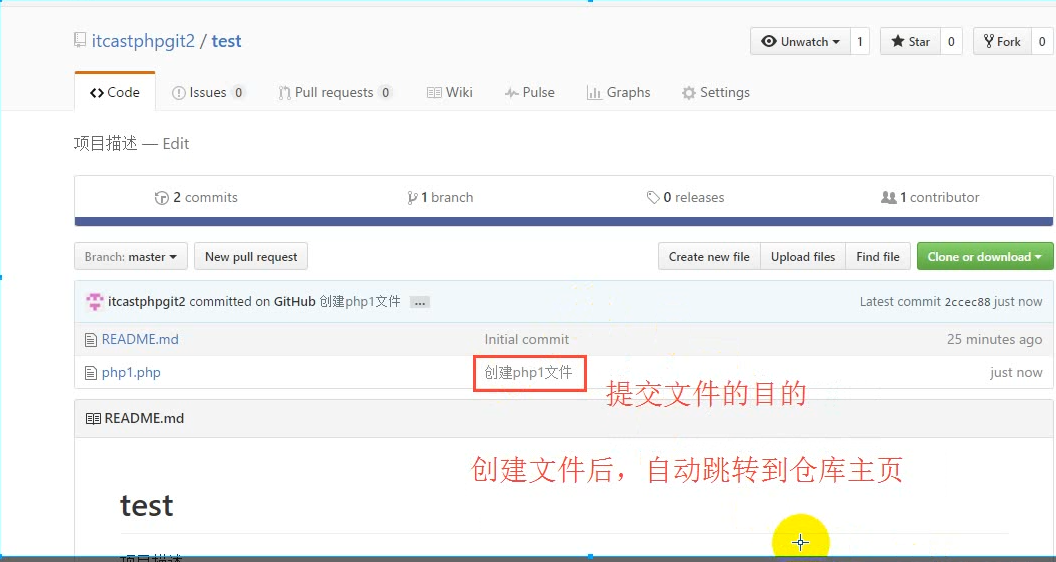
# 仓库主页说明

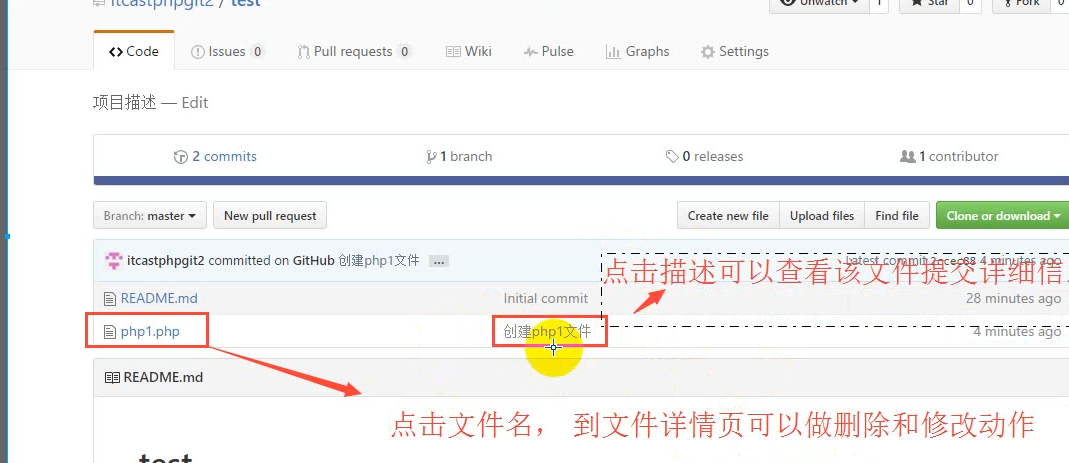


创建文件









## Dithub Issues

作用：发现代码BUG,但是目前没有成型代码，需要讨论时用（通俗的来讲就是，看到别人的错误，给他说，他登录的时候看到了，可以解决之后关闭issues）

# 开源项目贡献流程

## 新建Issue

提交使用问题或者建议或者想法

## Pull Request

步骤：

1. fork项目
2. 修改自己仓库的项目代码
3. 新建pull request
4. 等待作者操作审核

# Git安装和使用

## 目的

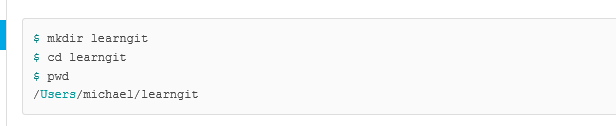
通过git管理github托管项目代码

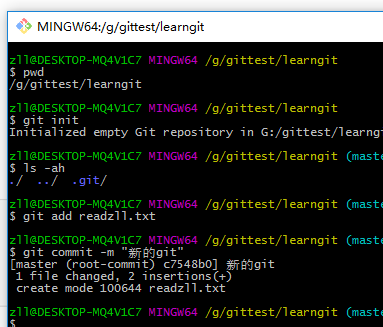
下载安装

1. Git官网下载安装



输入配置的参数，你的用户名和密码

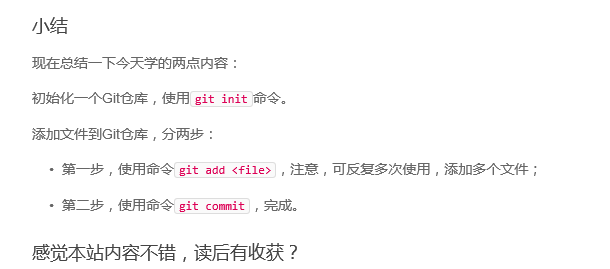




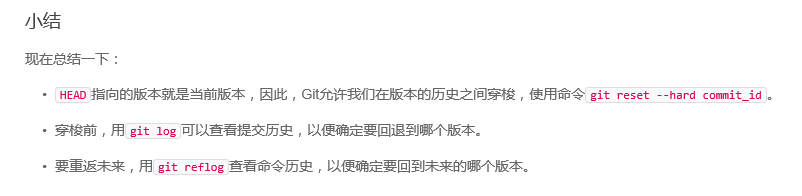
Mkdir:创建文件夹

Pwd:显示当前的文件

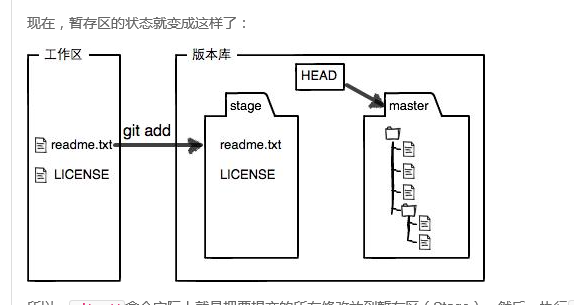
Init:初始化仓库（即现在我们的仓库名叫learngit,添加了readzll.txt）



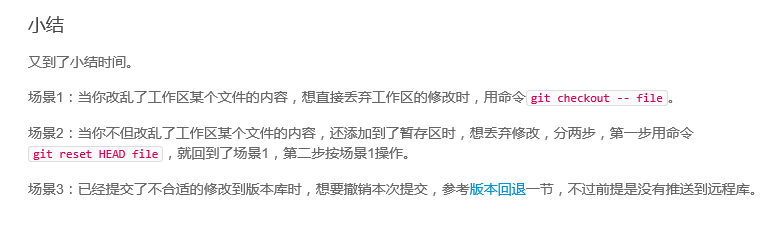


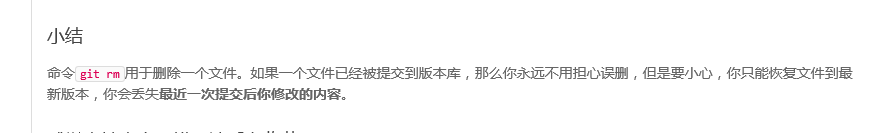


不想看复杂的日志信息可以用git log –pretty=onegitline



Unstaged:未缓存，未载入





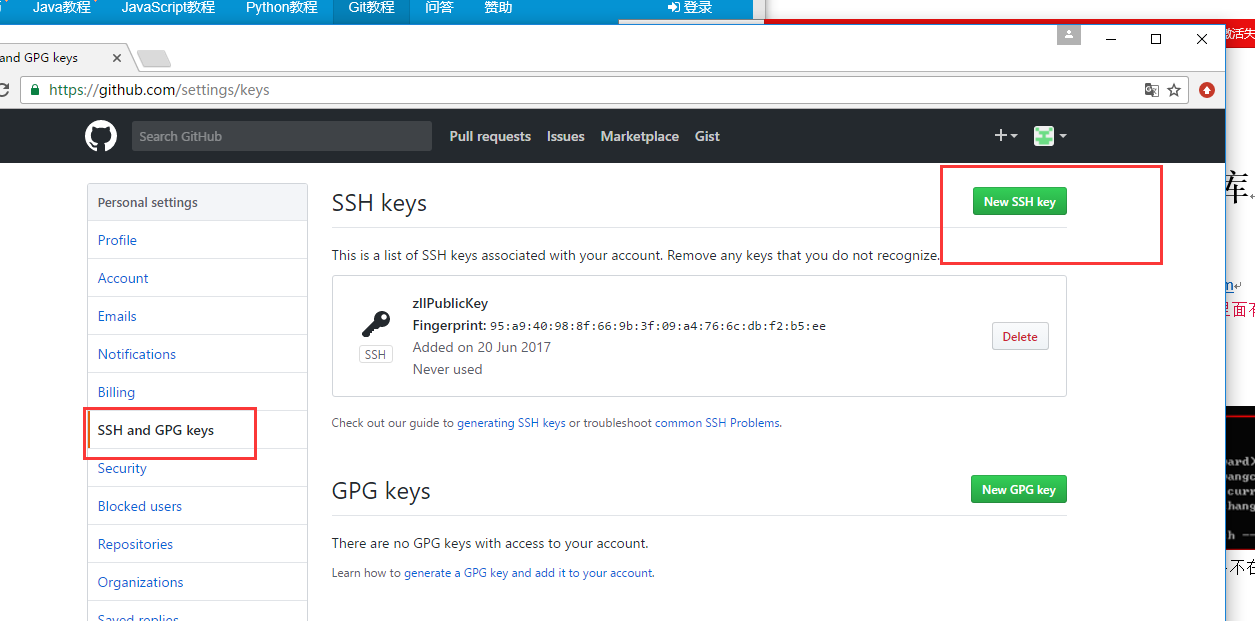
# 远程仓库

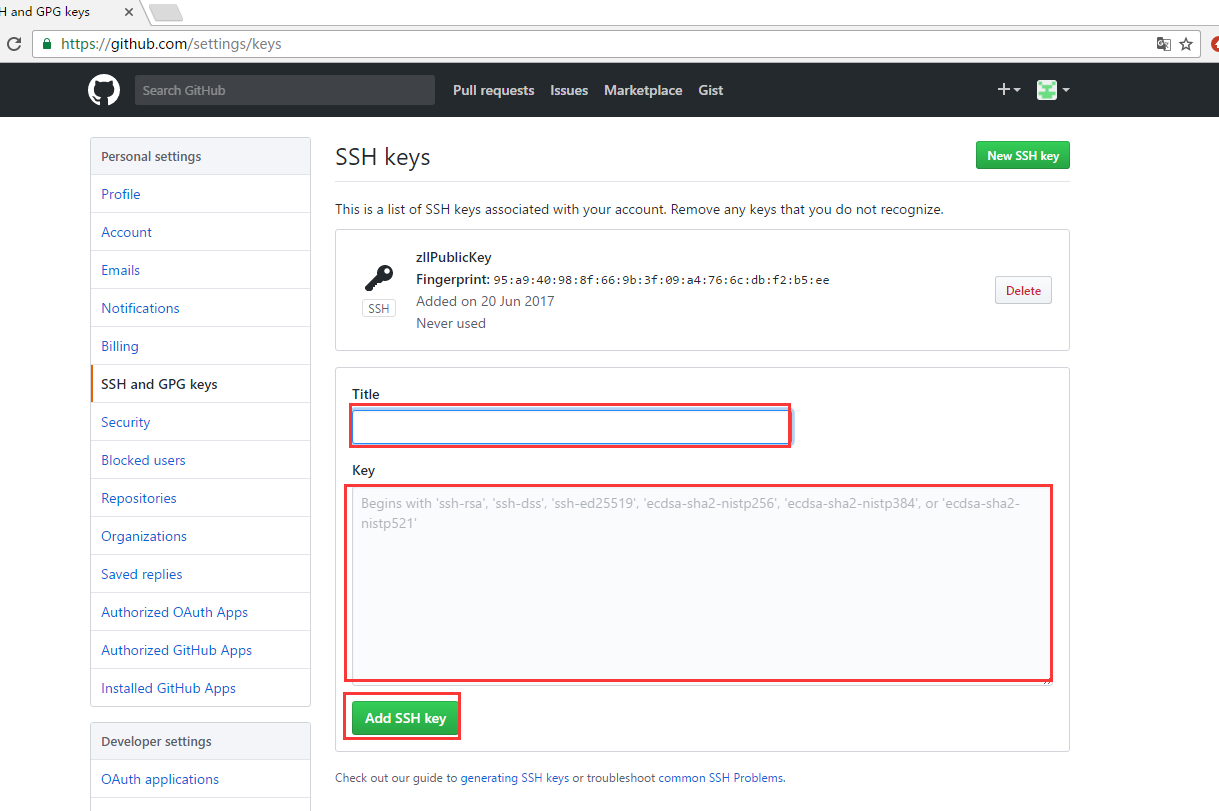
第一步：创建SSHKey

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

一切顺利的话，可以在用户目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub，分别是私钥和公钥

第二步：登录GitHub





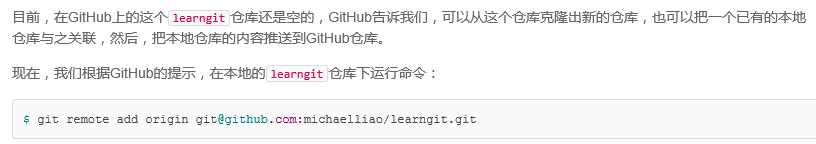
Title随意输入

Key里面填写id\_rsa.pub文件的内容

为什么GitHub需要SSHKey呢？因为GitHub需要识别出你推送的提交确实是你推送的，而不是别人冒充的，而Git支持SSH协议，所以，Git只要知道了你的公钥，就可以确定只有你自己才能推送

注意：

第三步、本地仓库和远程的仓库相关联



$ git remote add origin git@github.com:michaelliao/learngit.git

第四步、把本地仓库中所有内容推送到远成仓库中

git push –u origin master



出现错误的主要原因是github中的README.md文件不在本地代码目录中

git pull --rebase origin master

执行上面代码后可以看到本地代码库中多了README.md文件

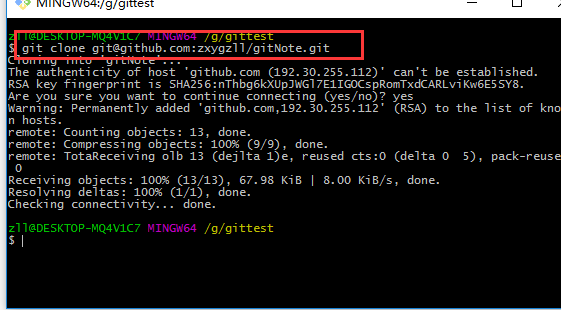
此时再执行语句 git push -u origin master即可完成代码上传到github

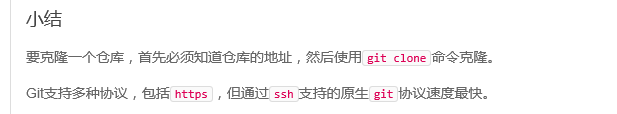
由于远程仓库是空的，我们第一次推送master分支的时候，加上了-u参数，Git不但会把本地内容推送到远程新的master分支上，还会把本地master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令

git push origin master



# 克隆仓库

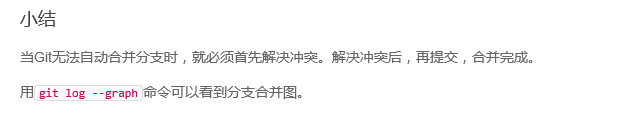




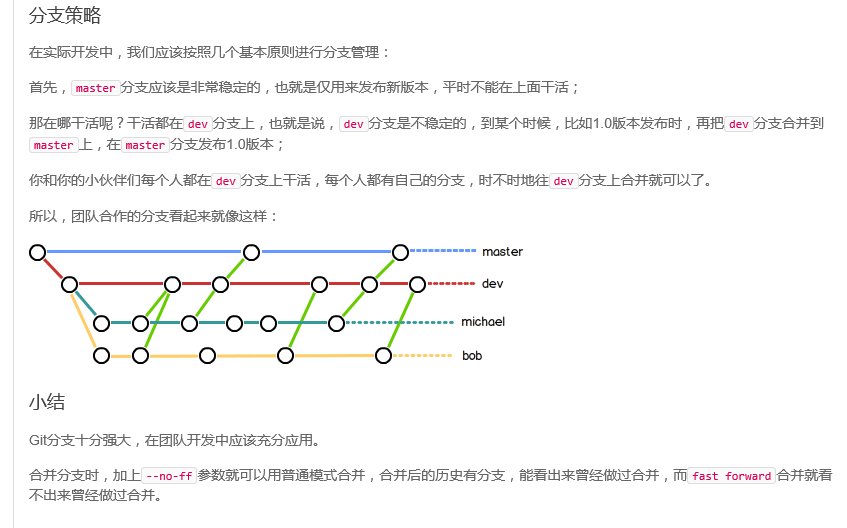
创建和合并分支



## 解决冲突



## 分支管理策略



# 常见单词

Auto-merging:自动混合

Conflict:冲突

Recursive:影藏的

Strategy:策略

Untracked:没有轨迹

## Bug分支

修复bug时，我们会通过创建新的bug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git stash pop，回到工作现场

需求：当你接到一个修复代号101代码的bug任务时，很自然，你想创建一个分支issue-101来修复它，但是，等等，当前正在dev上进行的工作还没有提交

第一步：首先确定要在哪个分支上修复bug，假定需要在master分支上修复，就从master创建临时分支

git stash （保存当前工作区的内容）

git checkout master

git checkout –b issue-101

git add readme.txt

git commit –m “fix bug 101”

第二步修复完成之后，切换到master分支，并且合并，最后删除issue-101分支

git checkout master

git merge –no-ff –m “merged bug fix 101” issue-101

git branch –d issue-101

第三步回到开始工作的分支上（dev）

git checkout dev

git status

此时工作区是干净的

git stash list

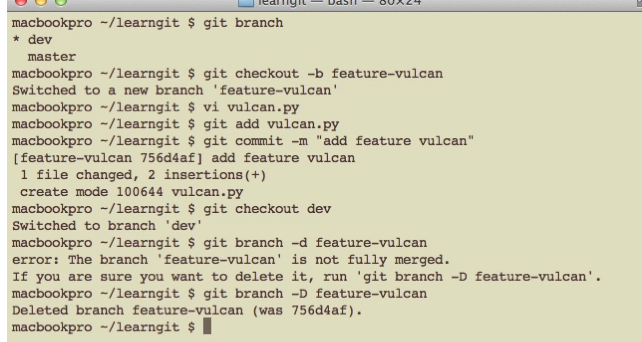
工作现场还在，git 把stash内容存到某个地方，需要恢复有两个办法

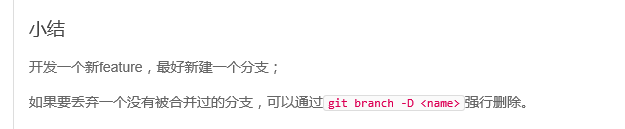
1. git stash apply (恢复但是stash内容并不删除)

git stash drop (用它来删除)

2、git stash pop （恢复的同时把stash内容也删除）

## Feature分支





## 多人协作

多人协作的工作模式通常是这样的：

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. 没有冲突或者解决完冲突后，再用git push origin branch-name推送就能成功

如果git pull 提示”no tracking information”，则说明则说明本地分支和远程分支的连接关系没有创建，用命令git branch –set-upstream branch-name origin/branch-name



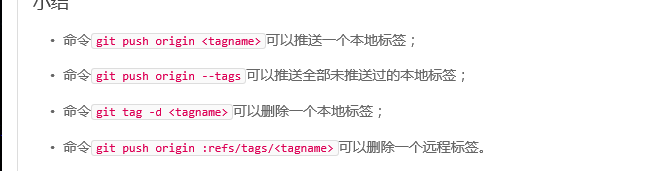
## 创建标签

为了快速定位到某一个版本

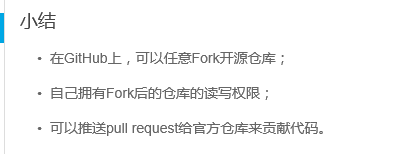


注意：建立了标签就跟发行了一个版本一样

## 操作标签



## 使用GitHub

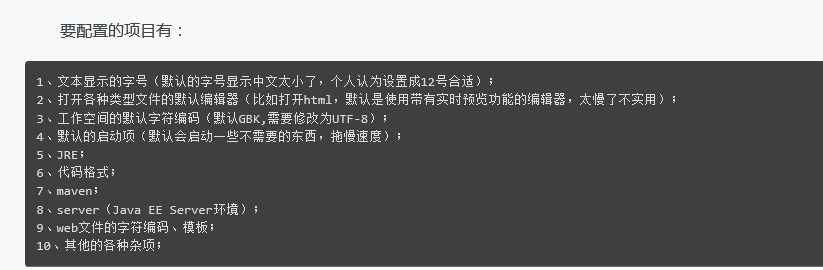


## 忽略特殊文件



<https://my.oschina.net/songxinqiang/blog/474530>

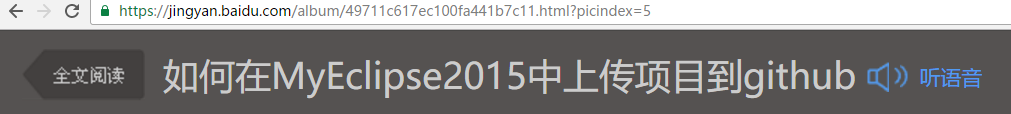




<http://yunchow.iteye.com/blog/2120001>

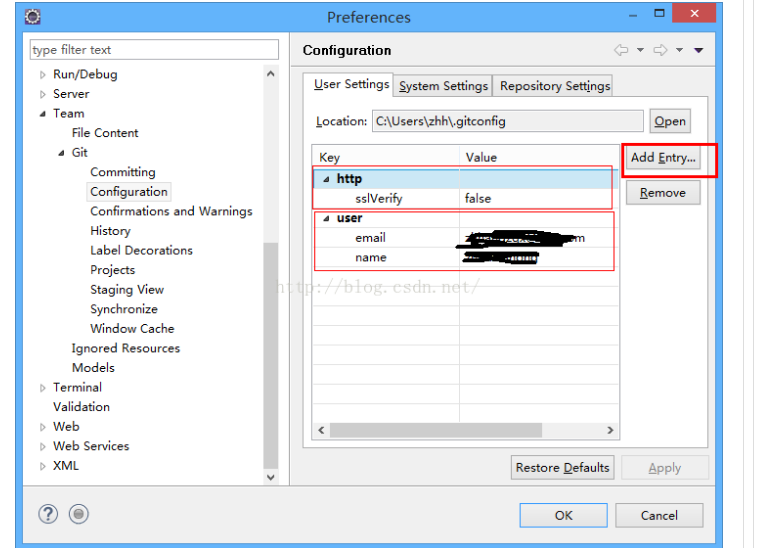


<https://jingyan.baidu.com/album/49711c617ec100fa441b7c11.html?picindex=5>



注意配置

http://blog.csdn.net/huangcsdnjava/article/details/73176213





<http://www.zuidaima.com/share/2251532046601216.htm>

注意：git和svn做版本控制的时候“问号”表示没有提交到版本库中的文件“黄色小圆柱”表示文件版本没有变化。