Git（分布式版本管理系统）

一、本地的git库

1、创建本地的Git版本库

选择一个合适的地方，创建一个文件夹在这个文件夹内打开git bash(在该文件夹下右击鼠标——单击“Git Bash Here”);

pwd命令 : 用于显示当前目录；

git init命令：初始化一个代码仓库把这个文件夹变成git可以管理的仓库；

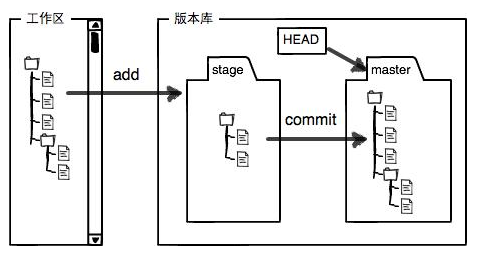
执行完以上命令会发现当前目录下多了一个.git的目录，这个目录是Git来跟踪管理版本库的，请勿手动修改该目录里面的文件，如修改，会把Git仓库给破坏。

如果你没有看到.git目录，那是因为这个目录默认是隐藏的，用ls -ah命令就可以看见。

2、工作区和暂存区以及版本库

工作区：就是你在电脑里能看到的目录（仓库文件夹里除.git目录以外的内容）

版本库：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的版本库里存了很多东西，其中最重要的就是称为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。



3、Git中文件的提交

第一步 把工作区的文件添加到暂存区

git add filename

filename为文件的名称

第二步 把暂存区的文件添加到版本库

git commit -m "instructions "

简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录，注：此处-m是必填内容。

4、查看版本和回退到某一版本

git status 命令用于查看版本库的状态

1）查看版本

git log //查看当前有效版本历史

git log --pretty=oneline //查看的版本信息在一行显示，易于阅读

gitreflog //查看所有版本

2）回退到某一版本

git reset --hard HEAD^// Git必须知道当前版本是哪个版本，在Git中，用HEAD表示当前版本，上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

git reset --hard commit-id //回退到特定id版本，参数也可以是id的全部或一部分

3）撤销修改的工作区的内容

git checkout – filename

命令git checkout -- filename意思是，把filename文件在工作区的修改全部撤销，这里有两种情况：一种是filename自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；另一种是filename已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。总之，就是让这个文件回到最近一次git commit或git add时的状态。

4）撤销暂存区的内容

git reset HEAD filename命令可以把暂存区的修改撤销掉（unstage），重新放回工作区

5）删除文件

rm filename命令用于删除文件

二、链接GitHub远程库

GitHub是一个提供Git仓库托管服务平台，请自行注册GitHub账号

1、准备工作

1）创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

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

你需要把邮件地址换成你自己的邮件地址，然后一路回车，使用默认值即可，由于这个Key也不是用于军事目的，所以也无需设置密码。

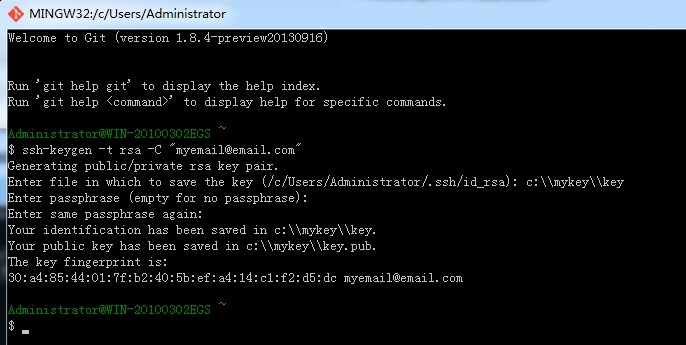
如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

如果想在指定的文件夹内建立钥匙输入如下代码

在msysgit命令行窗口输入ssh-keygen -t rsa -C '邮箱'

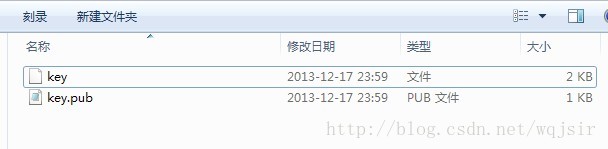
回车后，会叫输入密钥名字

这里输入：c:\\mykey\\key   其中key为密钥文件名



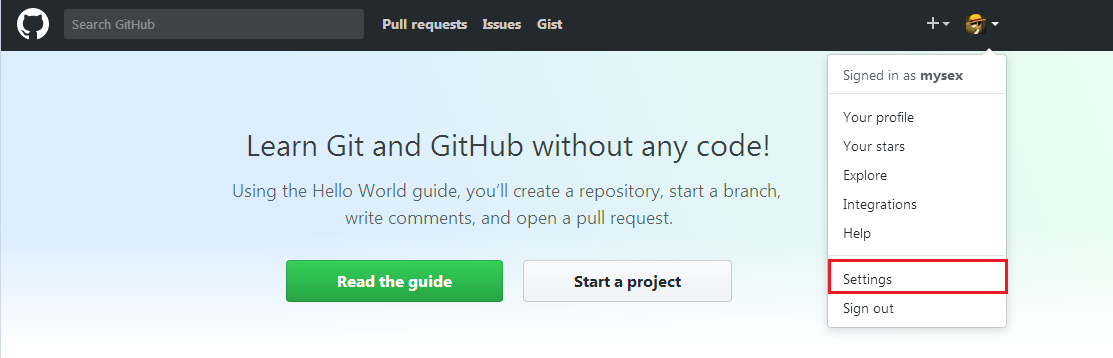
接着会叫输入密码，输入密码和确认密码后，我们的密钥就生成了。

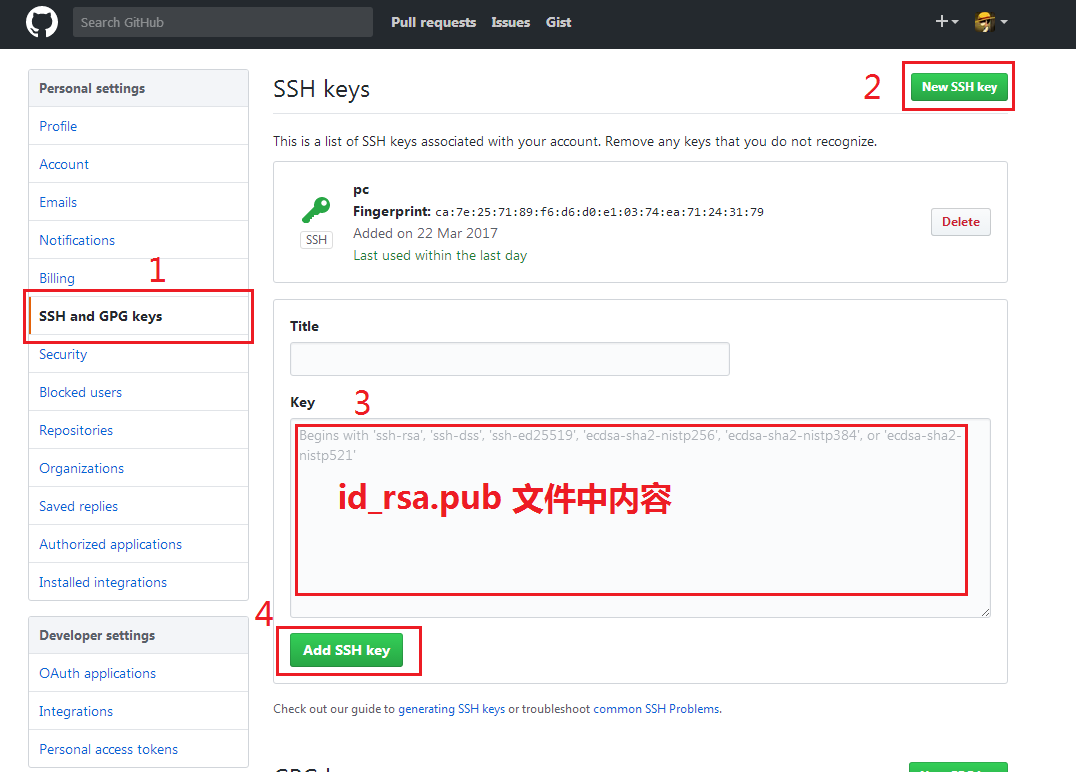
此时我们在c:\\mykey目录下就可以看到我们刚生成的密钥了：



这时候就可以将公钥上传至github

2）登陆GitHub，打开“settings”中“Add SSH Key”页面

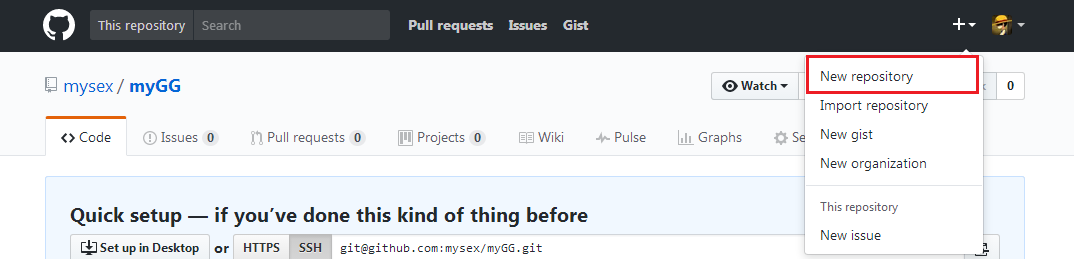


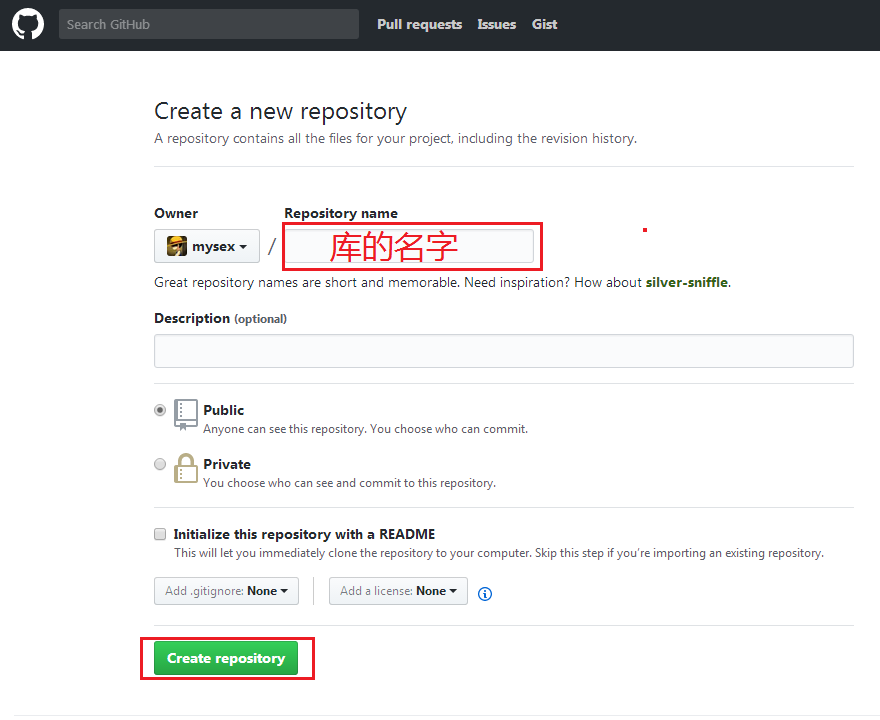


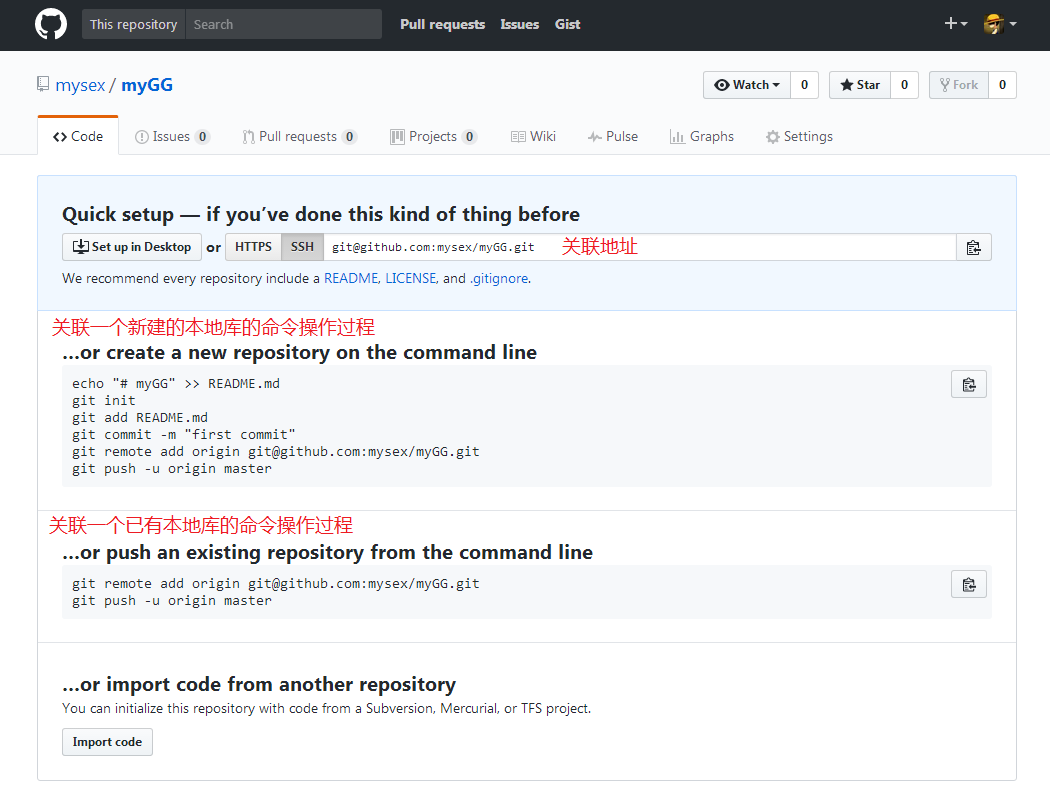
填上任意title，在Key文本框里粘贴id\_rsa.pub文件的内容。点击 Add Key后可以看到已经添加的key。

3）在GitHub上创建一个新的仓库

后在右上角找到“New repository”创建一个新的仓库。







4）添加远程的仓库(把一个已有的本地仓库与远程仓库关联)

$ git remote add origin git@github.com:mysex/myGG.git

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

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。

从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

5）克隆远程的仓库

$ gitclone url

如果远程仓库有东西

先执行 git pull - -rebase origin master

// git pull命令的作用是，取回远程主机某个分支的更新，再与本地的指定分支合并。

再执行 git push -u origin master 如果是空仓库直接执行此句即可。

//实际上是把当前分支master推送到远程

如果是从远程仓库克隆到本地可执行入如下命令

[git clone](http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013752340242354807e192f02a44359908df8a5643103a000) <https://github.com/guowenqiangde/testclone.git>

git 支持多种协议，默认的git://使用ssh，但也可以使用heeps等其他协议

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只使用https。

三、分支管理

在[版本回退](http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000)里，你已经知道，每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

一开始的时候，master分支是一条线，Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点：



当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev上：



分支管理

创建分支并切换到当前分支dev

git branch -b dev

相当于两句命令：git branch dev//创建分支 git checkout dev//切换到分支dev

git branch //列出所有分支。

然后add commit就在当前分支了。

git checkout master//切换回主分支，

git merge dev //快速合并分支dev的内容到当前分支

git branch -d dev //删除分支dev

解决冲突

git checkout -b feature1 //新建分支

add commit

git checkout master //切换到主分支

add commit

git merge feature1 //主分支快速合并feature1分支

失败有冲突

处理冲突文件 <<<<<< ===== >>>>>>

add commit

git log --graph --pretty=oneline --abbrev-commit

git log --graph //查看分支合并图

分支管理策略：

使用git merge file 后用git log --graph是看不到分支修改的

使用 git merge --no-ff -m"message" file 命令进行合并可以看到分支的修改

- - no-ff 表示禁用 Fast forward模式。

bug分支：

未完成的分支储藏起来，再创建bug分支，add commit 处理完bug 之后再继续开发未完成的分支

git stash 将当前分支dev储藏

git checkout master 切换到主分支

git checkout -b "isseue-101" 创建bug分支

add commit

git checkout master

git merge --no-ff -m"合并bug分支到主分支" issue-101

git branch -d issue-101 //删除bug分支

git checkout dev //回到dev分支

git stash list //查看储藏室

git stash pop //恢复同时删掉储藏相当于两个命令 ： git stash apply git stash drop

强行删除未合并的分支

git branch -D file

git clone实际上是把本地的master分支和远程的master分支对应，远程仓库默认名称是origin

git remote //查看远程库的信息

git remote -v //显示远程库的详细信息 如果没有推送权限，就看不到push地址

如果要推送其他分支，就用下边的命令

git push origin dev

415023

多人协作的工作模式：

1 git push origin branch-name

2 如果推送失败，需要git pull

3 如有冲突，解决冲突再提交

4 git push origin branch-name

如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

标签，tag是一个让人容易记住的名字，它跟某个commit绑在一起

1 切换到要打标签的分支

2 git tag v1.0//给当前分支打一个标签

git tag //查看所有标签

git tag v0.9 commit id//给某次提交打个标签

git show v0.9//查看标签信息

git tag -a v0.1 -m "说明"

操作标签：

git push origin <tagname>可以推送一个本地标签

git push origin - -tags可以推送全部未推送过的本地标签

git tag -d <tagname>可以删除一个本地标签

git push origin :refs/tags/<tagname> //删除一个远程标签（删之前先把本地的删了）

在GitHub上，可以任意Fork开源仓库

自己拥有Fork后的仓库的读写权限

可以推送pull request 给官方仓库来贡献代码

git config - -global color.ui true //为git配置简单的颜色

.gitignore要放到版本库里并且能被管理，只能忽略未被track的文件

git add -f App.class //强制添加被忽略的文件

git check-ignore -v App.class //查看ignore文件与App.class对应的忽略规则

git reset HEAD file //可以把add的修改撤销掉

gitconfig --global alias.st status //给status配置别名为st 给整段命令添加别名时要用引号包裹如：

gitconfig --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

- - global参数的意思是对当前用户起作用，不加是对当前仓库起作用

每个仓库的配置文件都放在git/config文件中，别名就在[alias]后面

当前用户的git配置文件放在用户主目录下的一个隐藏文件.gitconfig中

四、标签管理

发布一个版本时，我们通常先在版本库中打一个标签（tag），这样，就唯一确定了打标签时刻的版本。将来无论什么时候，取某个标签的版本，就是把那个打标签的时刻的历史版本取出来。所以，标签也是版本库的一个快照。

1、操作标签

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。

2、管理标签

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。