**github简易使用教程 v0.9**

作者：seerjk

**版本信息**

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 备注 |
| v0.9 | 2013-11-21 | git与github的基本使用方法 |
|  |  |  |
|  |  |  |

**github简介**

github是一个基于git的代码托管平台，付费用户可以建私人仓库，我们一般的免费用户只能使用公共仓库，也就是代码要公开。对于一般人来说公共仓库就已经足够了，而且我们也没多少代码来管理，O(∩\_∩)O~。下面是我总结的一些简单使用方法，供初学者参考。

【初识Github】

首先让我们大家一起喊一句“Hello Github”。YEAH!就是这样。

*原文 http://www.eoeandroid.com/thread-274556-1-1.html*

Git是一个分布式的版本控制系统，最初由Linus Torvalds编写，用作Linux内核代码的管理。在推出后，Git在其它项目中也取得了很大成功，尤其是在Ruby社区中。目前，包括Rubinius和Merb在内的很多知名项目都使用了Git。Git同样可以被诸如Capistrano和Vlad the Deployer这样的部署工具所使用。同样，eoe.cn客户端的源码也托管在github上。

GitHub可以托管各种git库，并提供一个web界面，但与其它像 SourceForge或Google Code这样的服务不同，GitHub的独特卖点在于从另外一个项目进行分支的简易性。为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。已经有人将GitHub称为代码玩家的MySpace。

在GitHub进行分支就像在Myspace（或Facebook…）进行交友一样，在社会关系图的节点中不断的连线。

GitHub 使用 git 分布式版本控制系统，而 git 最初是 Linus Torvalds 为帮助Linux开发而创造的，它针对的是 Linux 平台，因此 git 和 Windows 从来不是最好的朋友，因为它一点也不像 Windows。GitHub 发布了GitHub for Windows，为 Windows 平台开发者提供了一个易于使用的 Git 图形客户端。

**github使用简介**

**1.注册账户以及创建仓库**

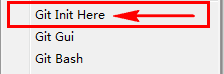
要想使用github第一步当然是注册github账号了。之后就可以创建仓库了（免费用户只能建公共仓库），Create a New Repository，填好名称后Create，之后会出现一些仓库的配置信息，这也是一个git的简单教程。

**2.安装客户端**

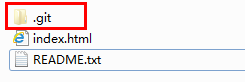
github是服务端，要想在自己电脑上使用git我们还需要一个git客户端，我这里选用**msysgit**，这个只是提供了git的核心功能，而且是基于命令行的。如果想要图形界面的话只要在msysgit的基础上安装TortoiseGit即可。

装完msysgit后右键鼠标会多出一些选项来，**在本地仓库里右键选择Git Init Here**，**会多出来一个.git文件夹**，这就表示本地git创建成功。

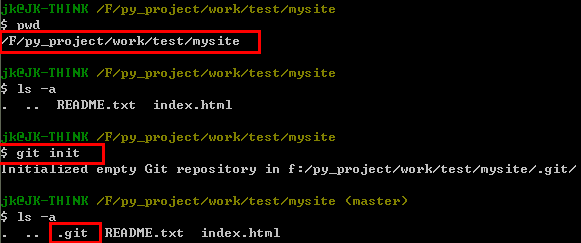
右键选择**Git Init Here：**



**会多出来一个.git文件夹：**



**或者**运行gitbash在git shell中切换到需要建立本地仓库的目录，输入命令: **git init。**

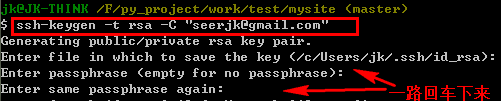


**3.配置Git**

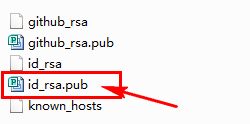
右键Git Bash进入git命令行，为了把本地的仓库传到github，还需要配置ssh key。

首先在本地创建ssh key；

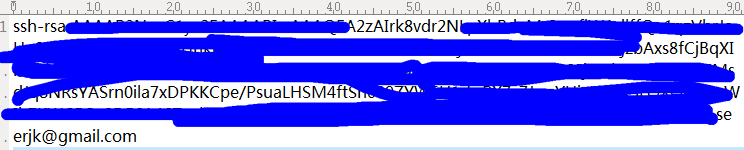
$ ssh-keygen -t rsa -C "your\_email@youremail.com"



后面的your\_email@youremail.com改为你的邮箱，之后会要求确认路径和输入密码，我们这使用默认的一路回车就行。成功的话会在~/下生成.ssh文件夹，进去，打开id\_rsa.pub，复制里面的key。

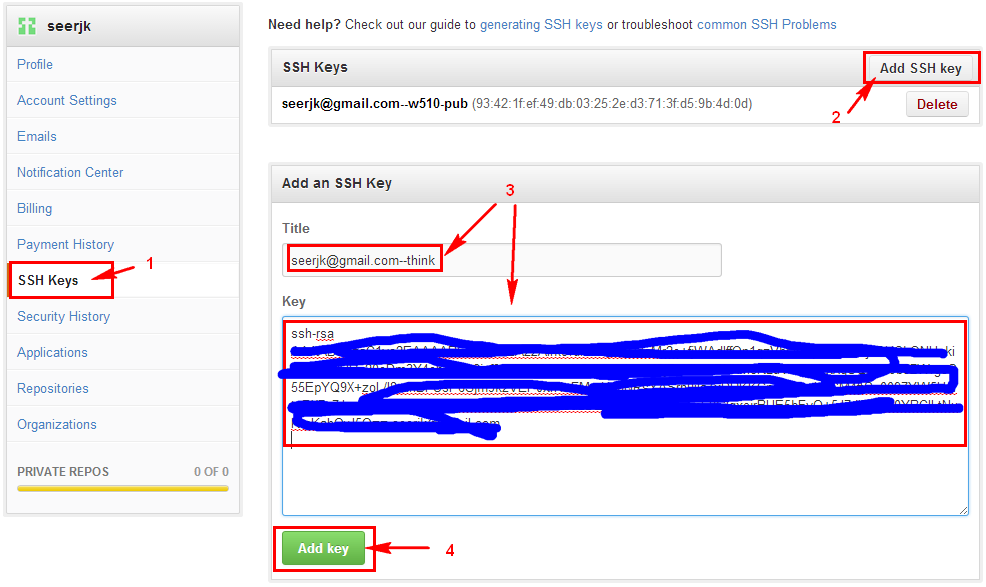


打开id\_rsa.pub文件复制里面的内容



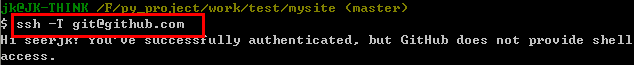
回到github，进入Account Settings，左边选择SSH Keys，Add SSH Key,title随便填，粘贴key。





为了验证是否成功，在git bash下输入：

$ ssh -T [git@github.com](mailto:git@github.com)

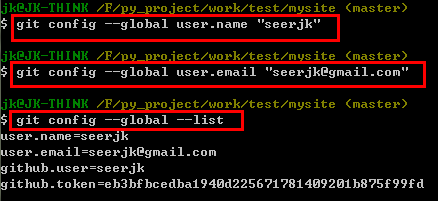


如果是第一次的会提示是否continue，输入yes就会看到：You’ve successfully authenticated, but GitHub does not provide shell access 。这就表示已成功连上github。

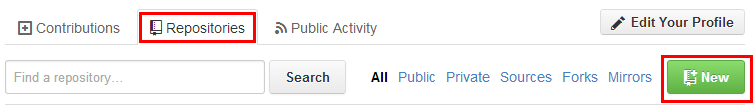
接下来我们要做的就是把本地仓库传到github上去，在此之前还需要设置username和email，因为github每次commit都会记录他们。

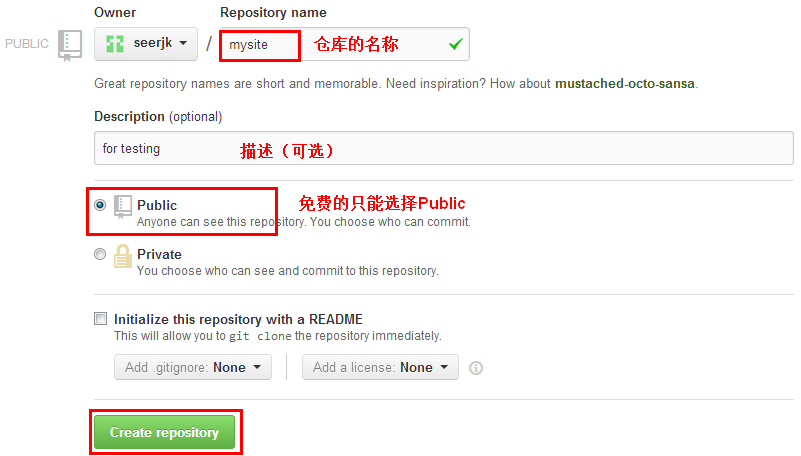
$ git config --global user.name "your name"

$ git config --global user.email "your\_email@youremail.com"



在github上面新建repo（软件仓库）



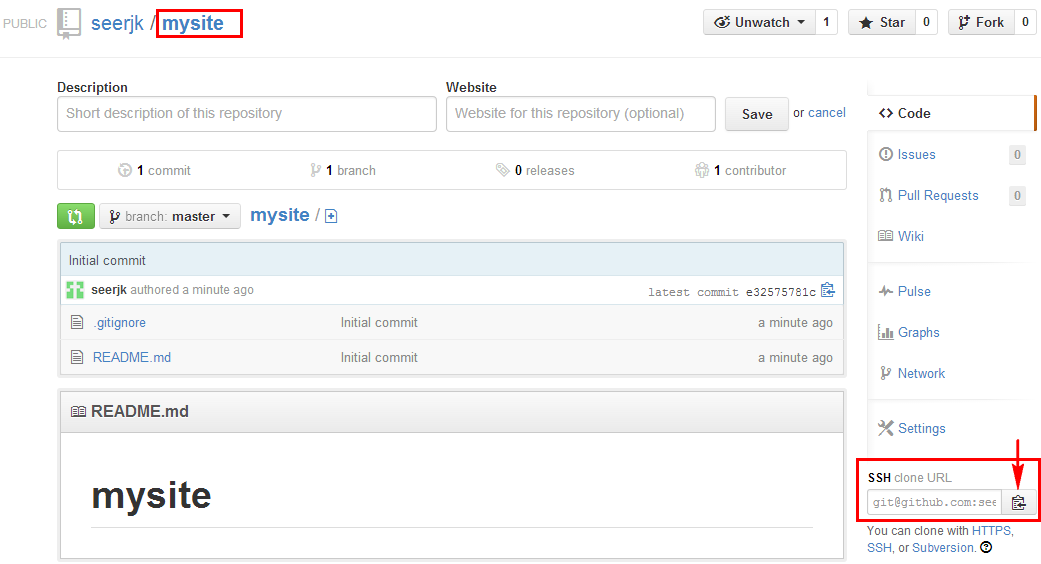


Quick setup 快速配置：



进入刚才创建的mysite仓库，在右侧下方的SSH clone URL里面是mysite仓库的远程地址。

|  |
| --- |
| **git@github.com:seerjk/mysite.git 后面设置远程地址的时候要用到。** |



在本地，进入要上传的仓库，右键git bash，添加远程地址：

$ git remote add origin git@github.com:yourName/yourRepo.git

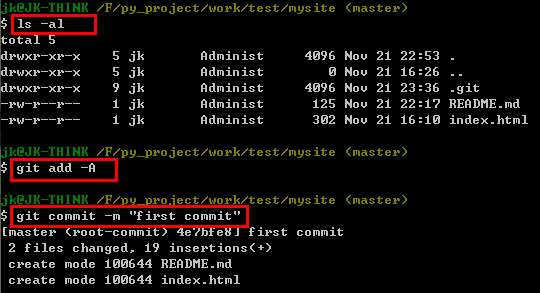
F:\openstack_team\report4op-jk\github简单使用教程--jk修改_files\Image [14].png

后面的yourName和yourRepo表示你再github的用户名和刚才新建的仓库，加完之后进入.git，打开config，这里会多出一个remote “origin”内容，这就是刚才添加的远程地址，也可以直接修改config来配置远程地址。

**4.提交、上传**

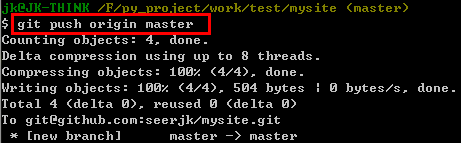
接下来在本地仓库里添加一些文件，比如README，

|  |
| --- |
| $ touch README.md  $ git add README.md  $ git commit -m "first commit" |

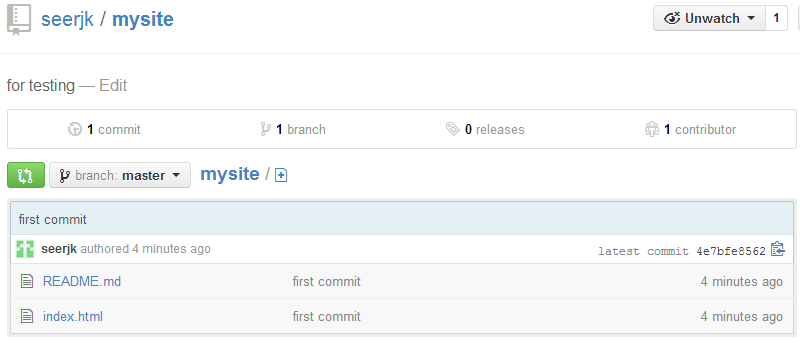


上传到github：

|  |
| --- |
| $ git push origin master |



在github上面可以查看到刚才push的文件。



git push命令会将本地仓库推送到远程服务器。

git pull命令则相反，从远程服务器拉到本地。

修改完代码后，使用git status可以查看文件的差别，使用git add 添加要commit的文件，也可以用git add -i来智能添加文件。

之后git commit提交本次修改，git push上传到github。

**5.  .gitignore文件**

.gitignore顾名思义就是告诉git需要忽略的文件，这是一个很重要并且很实用的文件。一般我们写完代码后会执行编译、调试等操作，这期间会产生很多中间文件和可执行文件，这些都不是代码文件，是不需要git来管理的。我们在git status的时候会看到很多这样的文件，如果用git add -A来添加的话会把他们都加进去，而手动一个个添加的话也太麻烦了。这时我们就需要.gitignore了。比如一般c#的项目我的.gitignore是这样写的：

bin

\*.suo

obj

bin和obj是编译目录，里面都不是源代码，忽略；suo文件是vs2010的配置文件，不需要。这样你在git status的时候就只会看到源代码文件了，就可以放心的git add -A了。

**6.tag**

我们可以创建一个tag来指向软件开发中的一个关键时期，比如版本号更新的时候可以建一个“v2.0”、“v3.1”之类的标签，这样在以后回顾的时候会比较方便。tag的使用很简单，主要操作有：查看tag、创建tag、验证tag以及共享tag。

**6.1查看tag**

列出所有tag：

|  |
| --- |
| git tag |

这样列出的tag是按字母排序的，和创建时间没关系。如果只是想查看某些tag的话，可以加限定：

|  |
| --- |
| git tag -l v1.\* |

这样就只会列出1.几的版本。

**6.2创建tag**

创建轻量级tag：

|  |
| --- |
| git tag v1.0 |

这样创建的tag没有附带其他信息，与之相应的是带信息的tag：

|  |
| --- |
| git tag -a v1.0 -m 'first version' |

-m后面带的就是注释信息，这样在日后查看的时候会很有用，这种是普通tag，还有一种有签名的tag：

|  |
| --- |
| git tag -s v1.0 -m 'first version' |

前提是你有GPG私钥，把上面的a换成s就行了。除了可以为当前的进度添加tag，我们还可以为以前的commit添加tag：

|  |
| --- |
| #首先查看以前的commit  git log --oneline  #假如有这样一个commit：8a5cbc2 updated readme  #这样为他添加tag  git tag -a v1.1 8a5cbc2 |

**6.3删除tag**

很简单，知道tag名称后：

|  |
| --- |
| git tag -d v1.0 |

**6.4验证tag**

如果你有GPG私钥的话就可以验证tag：

|  |
| --- |
| git tag -v v1.0 |

**6.5共享tag**

我们在执行git push的时候，tag是不会上传到服务器的，比如现在的github，创建tag后git push，在github网页上是看不到tag的，为了共享这些tag，你必须这样：

git push origin --tags

**7. 删除github上面的repositories软件仓库**

|  |
| --- |
| 1.到你的个人中心.点击你的个人账号.下图的红色部分 F:\openstack_team\report4op-jk\github简单使用教程--jk修改_files\Image [18].png  2.点击repositories(仓库),选择你要删除的项目 F:\openstack_team\report4op-jk\github简单使用教程--jk修改_files\Image [19].png  3. 点击进入的仓库的右侧工具栏里面的Settings F:\openstack_team\report4op-jk\github简单使用教程--jk修改_files\Image [20].png  4.页面拉到最下方，点击 delete this repository按钮 F:\openstack_team\report4op-jk\github简单使用教程--jk修改_files\Image [21].png  5. 输入一遍要删除的仓库的名称，点击按钮I understand the consequences, delete this repository F:\openstack_team\report4op-jk\github简单使用教程--jk修改_files\Image [22].png |