**产生背景：**

很多人都知道，Linus在1991年创建了开源的Linux，从此，Linux系统不断发展，已经成为最大的服务器系统软件了。

　　Linus虽然创建了Linux，但Linux的壮大是靠全世界热心的志愿者参与的，这么多人在世界各地为Linux编写代码，那Linux的代码是如何管理的呢？

　　事实是，在2002年以前，世界各地的志愿者把源代码文件通过diff的方式发给Linus，然后由Linus本人通过手工方式合并代码！

　　你也许会想，为什么Linus不把Linux代码放到版本控制系统里呢？不是有CVS、SVN这些免费的版本控制系统吗？因为Linus坚定地反对CVS和SVN，这些集中式的版本控制系统不但速度慢，而且必须联网才能使用。有一些商用的版本控制系统，虽然比CVS、SVN好用，但那是付费的，和Linux的开源精神不符。

　　不过，到了2002年，Linux系统已经发展了十年了，代码库之大让Linus很难继续通过手工方式管理了，社区的弟兄们也对这种方式表达了强烈不满，于是Linus选择了一个商业的版本控制系统BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神，授权Linux社区免费使用这个版本控制系统。

　　安定团结的大好局面在2005年就被打破了，原因是Linux社区牛人聚集，不免沾染了一些梁山好汉的江湖习气。开发Samba的Andrew试图破解BitKeeper的协议（这么干的其实也不只他一个），被BitMover公司发现了（监控工作做得不错！），于是BitMover公司怒了，要收回Linux社区的免费使用权。

　　Linus可以向BitMover公司道个歉，保证以后严格管教弟兄们，嗯，这是不可能的。实际情况是这样的：

　　Linus花了两周时间自己用C写了一个分布式版本控制系统，这就是Git！一个月之内，Linux系统的源码已经由Git管理了！牛是怎么定义的呢？大家可以体会一下。

**Git迅速成为最流行的分布式版本控制系统，尤其是2008年，GitHub网站上线了，它为开源项目免费提供Git存储，无数开源项目开始迁移至GitHub，包括jQuery，PHP，Ruby等等。**

　　历史就是这么偶然，如果不是当年BitMover公司威胁Linux社区，可能现在我们就没有免费而超级好用的Git了。

工作空间 暂存区 本地代码仓库 远程仓库

在文件夹里输入git.init这个文件夹就是git软件的工作区(git版本控制工具开始管理这个空间）

注册信息

git config --global user.name "Huiyuanguzhi"

git config --global user.email "635471495@qq.com"

Git status 检测当前文件夹中文件状态

三种状态的变化

1红色：新增的文件或者修改的文件或删除的文件

2绿色：git已经管理起来，还没提交到远程仓库

3白色：表示远程仓库和本地文件版本一致

不显示中更改配置：git config --global core.quotepath false

git status -s 简短的输出结果

1 git config --global user.name "tuhooo"// 如果后面没有用户名就是查看, 有的话就是设置

2 git config --global user.email tuhooo@163.com // 如果后面没有邮箱就是查看, 有的话就是设置

Git add . 表示没有被管理的文件一起被管理

Git commit -m ‘备注’;提交到git远程仓库

git rm --cached 撤出暂存区

git rm -f 同时删除工作目录和暂存区的文件

git MV old-filename new-filename 给文件改名

git log -p 显示每次提交具体的变化

git tag –a v1.0.0打标签

git tag -d v1.0.0删除标签

git show v1.0 查看此版本所修改的内容

gitk 看分支图

git push origin <tagname>推送某个标签到远程  
 git push origin testtag

回滚代码

Git reset --hard HEAD(表示回到head版本，应该是反馈操作）

Git reset --hard HEAD^(表示回到head上个一版本，以此类推);

Git reset --hard 版本号（可以用git reflog 查看所有版本id）

git checkout 【版本id】 【FileName】（回退指定文件到某个版本）

回退到指定版本：git reset a4e215234aa4927c85693dca7b68e9976948a35e MainActivity.java

Git log --pretty=oneline （简明的列表显示提交的版本）

git diff --cached 比较暂存区和本地仓库比较

git diff 默认对比工作目录和暂存区有什么不同

如果某个文件已经被仓库管理，如果在更改此文件，直接需要一条命令提交即可

git commit #相当于虚拟机的镜像，任何操作都被做了一次快照，可以恢复到任何位置

首先在做撤销之前，我们可以先用 git status 查看下当前的状态。如下所示：显示已经修改，但是还没有添加到暂存区

git checkout -- readme.txt （注意文件名前边必须有空格）,撤销修改的内容

Git remote -v 查看远程仓库地址

创建分支

Git branch 分支名

查看分支

Git branch

切换分支

Git checkout 分支名

合并分支

Git merge 分支名（注意：要在master分支下合并）

删除分支

Git branch -d 分支名

查看分支指向

git log --online --decorate

git checkout -b 分支名 创建并切换分支

面试题

线上代码出bug怎么办？

答：拉分支修复，在合并到代码主干，分支的作用之一就是生产环境代码隔离

**4.比较差异**

git diff：比较工作区和暂存区的**所有文件**差异

git diff <file name>：比较工作区和暂存区的**指定文件**的差异

git diff HEAD|HEAD^|HEAD~|哈希索引值 <file name>：比较工作区跟本地库的某个版本的**指定文件**的差异

**二、本地库跟远程库交互：**

git clone <远程库地址>：克隆远程库

　　功能：①完整的克隆远程库为本地库，②为本地库新建origin别名，③初始化本地库

git remote -v：查看远程库地址别名

git remote add <别名><远程库地址>：新建远程库地址别名

git remote rm <别名>：删除本地中远程库别名

git push <别名><分支名>：本地库某个分支推送到远程库，分支必须指定

git pull <别名><分支名>：把远程库的修改拉取到本地

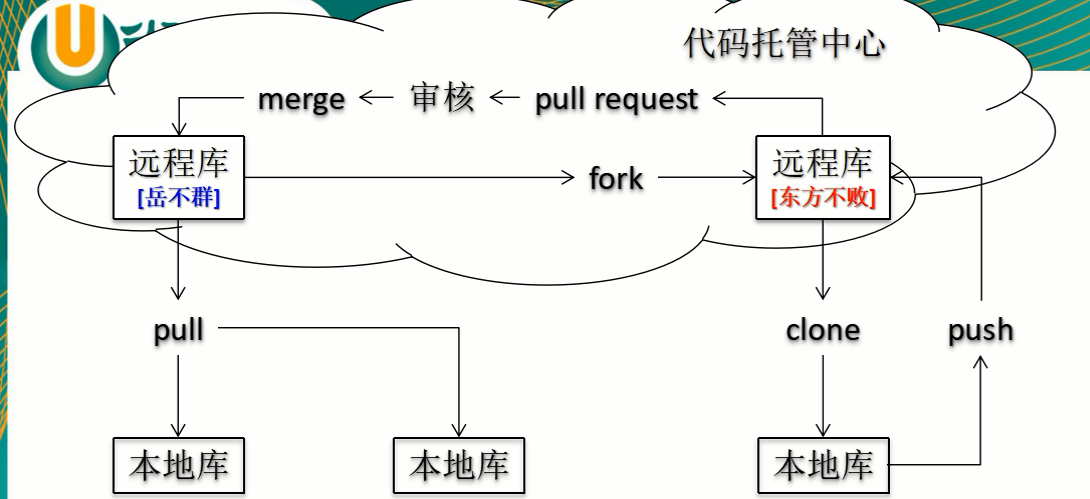
　　tip：该命令包括git fetch，git merge

git fetch <远程库别名><远程库分支名>：抓取远程库的指定分支到本地，但没有合并

git merge <远程库别名/远程库分支名>：将抓取下来的远程的分支，跟当前所在分支进行合并

git fork：复制远程库

　　tip：一般是外面团队的开发人员fork本团队项目，然后进行开发，之后外面团队发起pull request，然后本团队进行审核，如无问题本团队进行merge（合并）到团队自己的远程库，整个流程就是本团队跟外面团队的协同开发流程，Linux的团队开发成员即为这种工作方式。



-------------------------------------------------------------------------------------------git remote -v: 查看远程仓库详细信息，可以看到仓库名称

git remote remove 【分支名】：删除orign仓库（如果把origin拼写成orign，删除错误名称仓库）

生产秘钥和共匙

$ ssh-keygen -t rsa

git remote add origin https://github.com/Huiyuanguzhi/file.git

git push -u origin master（向远程仓库推送代码）

git clone 【仓库地址】 从远程仓库下载代码

git pull origin 【分支名】(前提是代码已经克隆)

记录图形展示

git log --graph --pretty-formart:"%h %s"

brbase（变基） 很重要，面试经常考

git rebase -i

git log --graph 图形显示日志

git rebase --continue

免密登陆

方法一：URL中体现

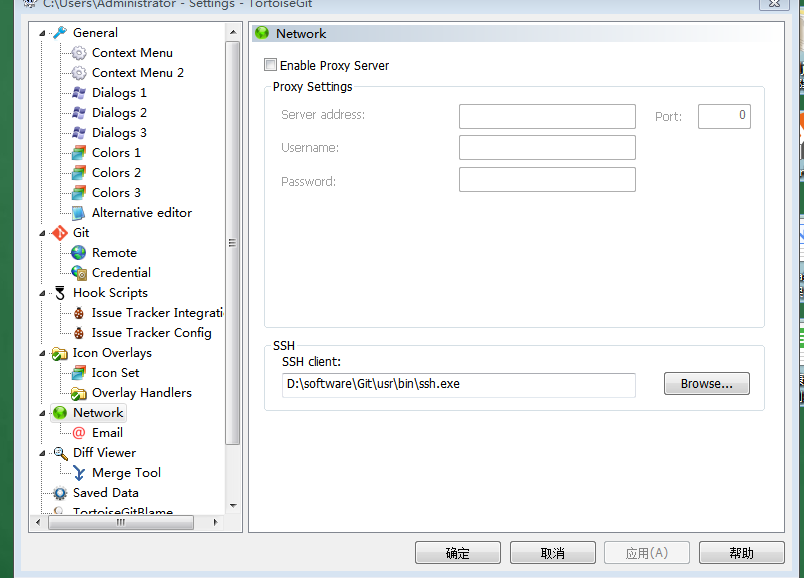
原来的网址：https://github.com/Huiyuanguzhi/file.git

修改地址：https://用户名：密码@github.com/Huiyuanguzhi/file.git

git push origin master

方法二：SSH实现

如果用偷偷四git，还要配置下图的git安装路径



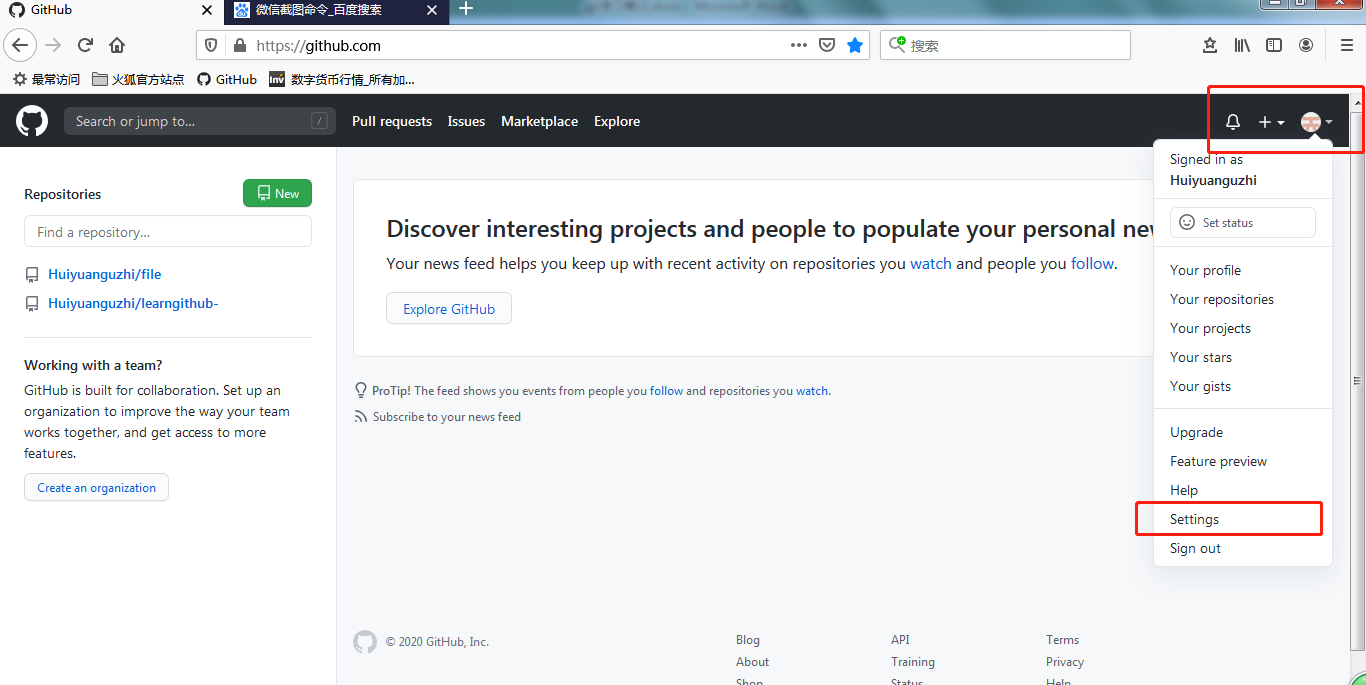
生产秘钥和共匙

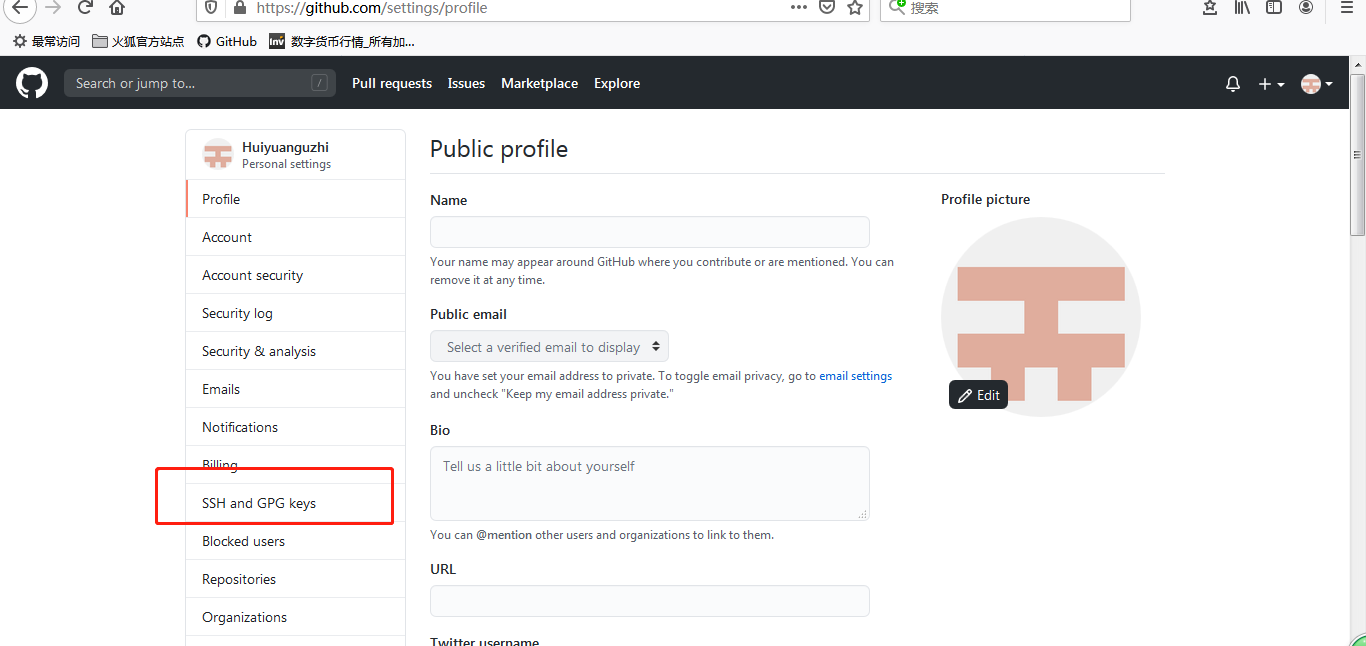
$ ssh-keygen -t rsa

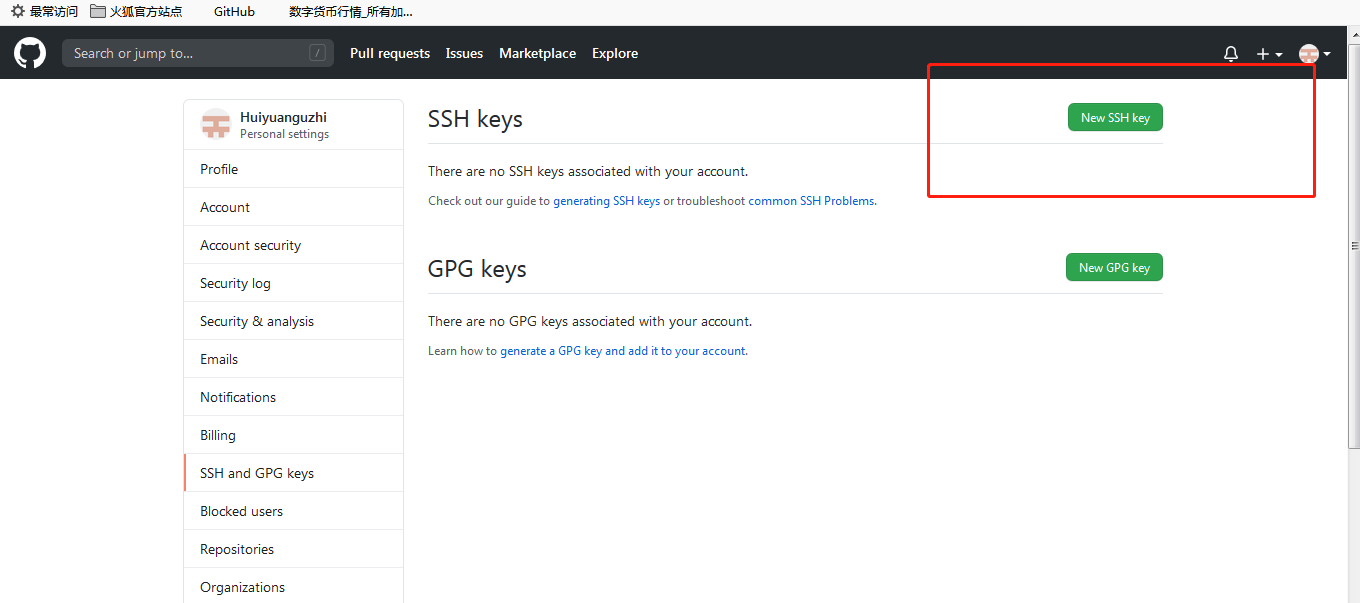
1生成公钥和私钥（默认放在-/.ssh目录下，id\_rsa.pub公钥、id\_rsa私钥）

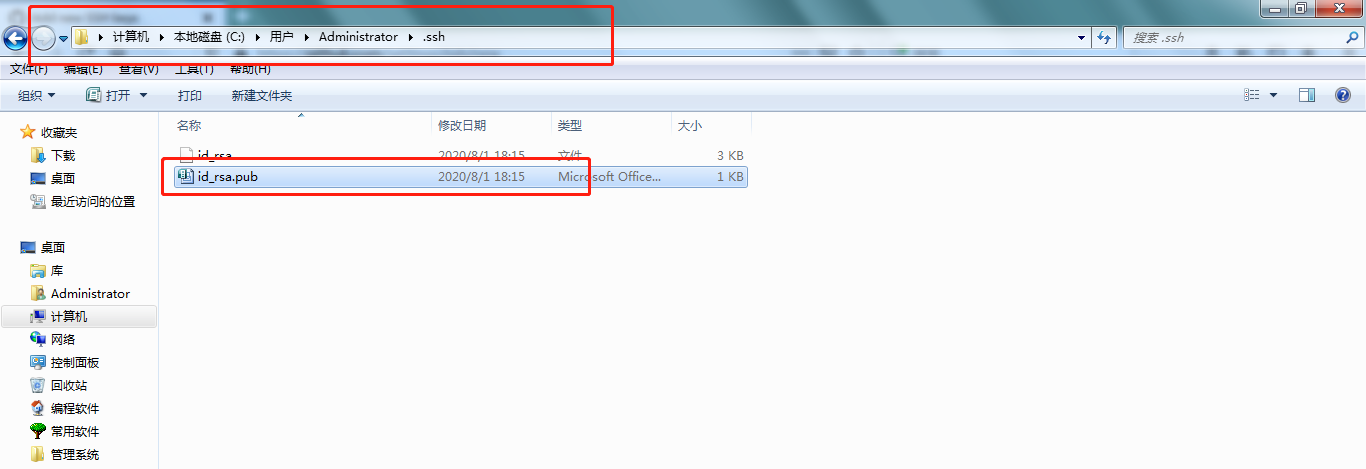
ssh-keygen

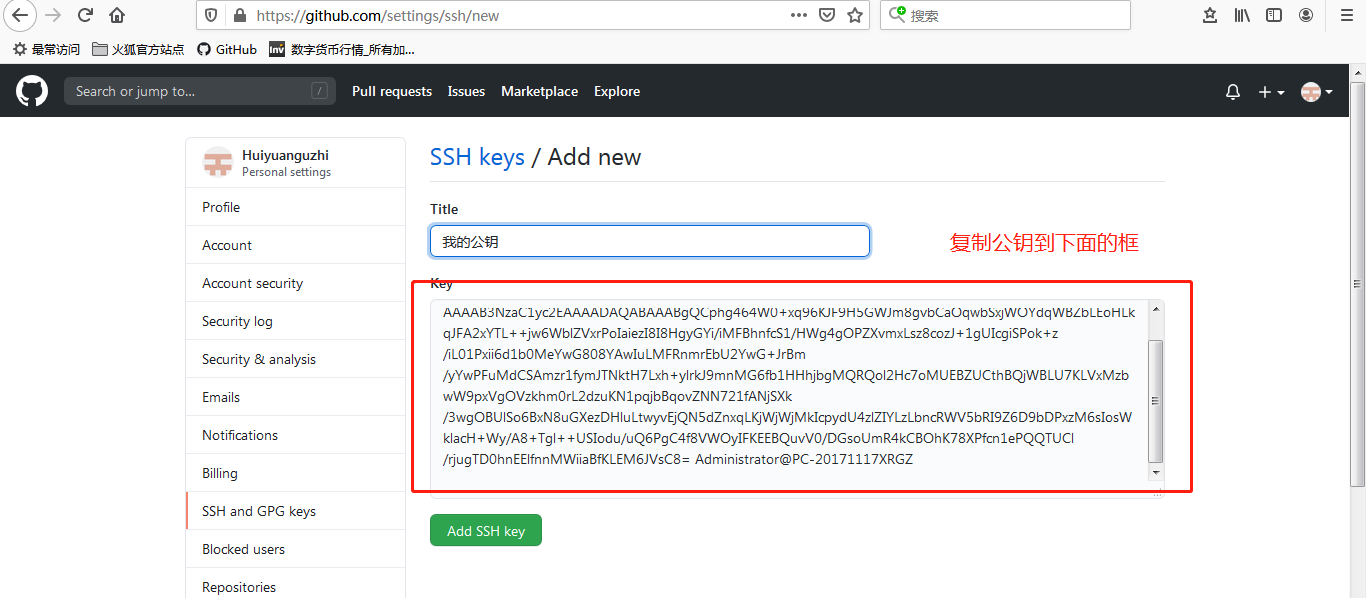
2、拷贝公钥的内容，设置到github中

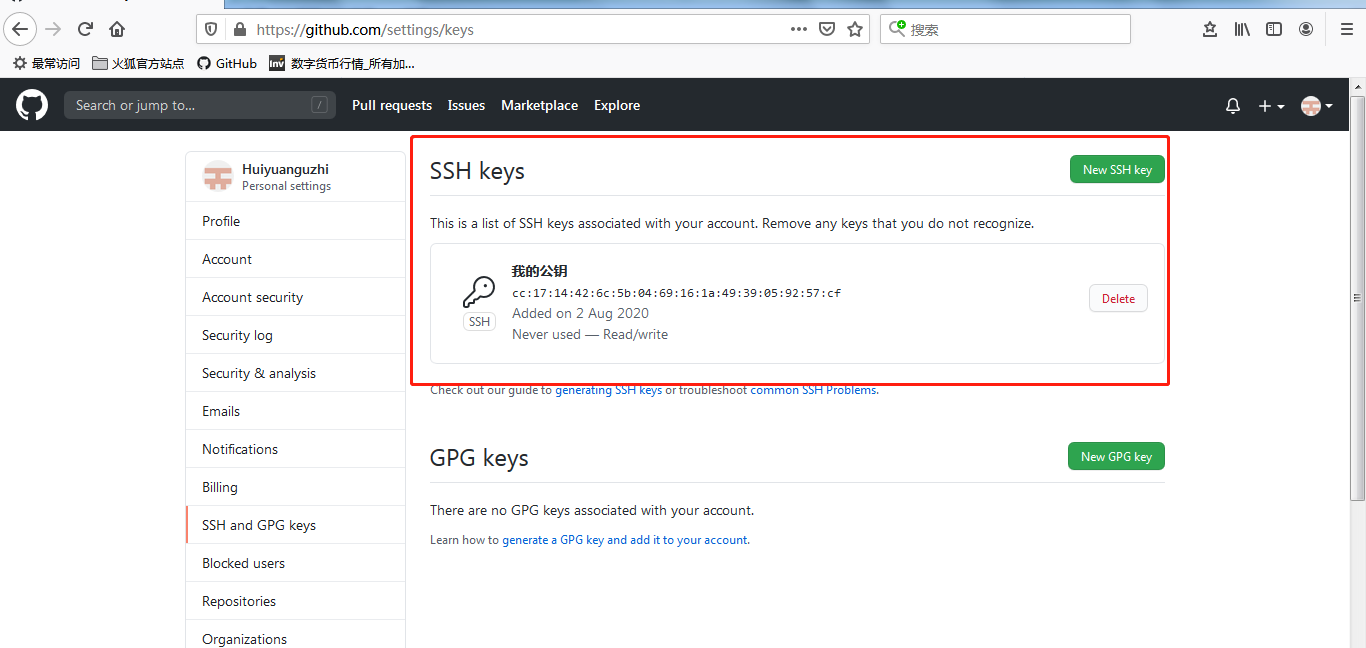






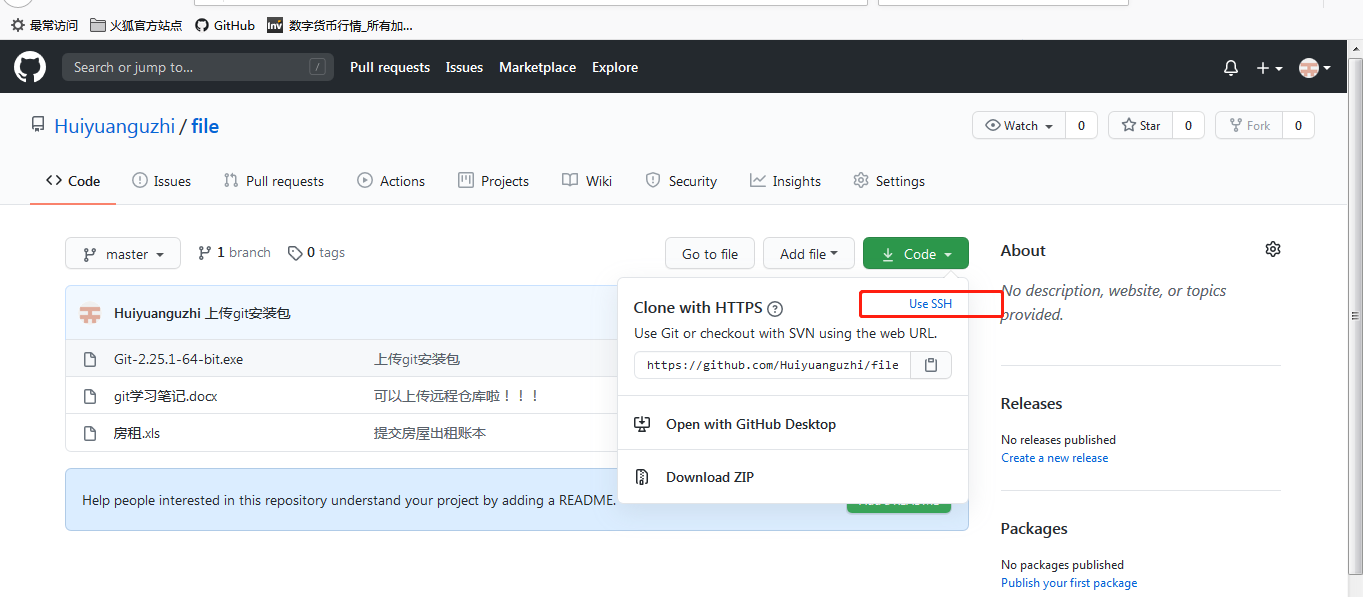


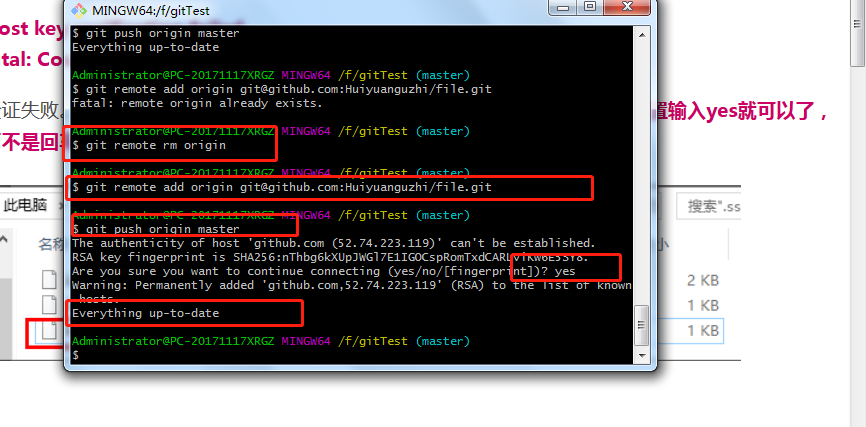




公钥添加完成

3.在git本地中配置ssh地址





开始用的不是免密登陆，改了之后要删除原来的origin

在输入图中命令

git remote add origin git@github.com:Huiyuanguzhi/file.git

4、以后使用

git push origin master不用输入密码了

git忽略的文件

让git不在管理当下目录下的某些文件

gitignore

例如：a.h表示不管a.h这个文件

\*.h表示以h结尾的都不管

files/文件夹里所有的文件也不管

\*.py[c|a|d]表示pyc 和pya和pyd都不管了

**github任务管理相关**

issues 文档以及任务管理

wiki 相关文档

**git clean**

git clean命令用来从你的工作目录中删除所有没有tracked过的文件

git clean经常和git reset --hard一起结合使用. 记住reset只影响被track过的文件, 所以需要clean来删除没有track过的文件. 结合使用这两个命令能让你的工作目录完全回到一个指定的<commit>的状态

用法

git clean -n

是一次clean的演习, 告诉你哪些文件会被删除. 记住他不会真正的删除文件, 只是一个提醒

git clean -f

删除当前目录下所有没有track过的文件. 他不会删除.gitignore文件里面指定的文件夹和文件, 不管这些文件有没有被track过

git clean -f <path>

删除指定路径下的没有被track过的文件

git clean -df

删除当前目录下没有被track过的文件和文件夹

git clean -xf

删除当前目录下所有没有track过的文件. 不管他是否是.gitignore文件里面指定的文件夹和文件

git reset --hard和git clean -f是一对好基友. 结合使用他们能让你的工作目录完全回退到最近一次commit的时候

git clean对于刚编译过的项目也非常有用. 如, 他能轻易删除掉编译后生成的.o和.exe等文件. 这个在打包要发布一个release的时候非常有用

下面的例子要删除所有工作目录下面的修改, 包括新添加的文件. 假设你已经提交了一些快照了, 而且做了一些新的开发

git reset --hard

git clean -df

运行后, 工作目录和缓存区回到最近一次commit时候一摸一样的状态，git status会告诉你这是一个干净的工作目录, 又是一个新的开始了！

强制提交(是个恨命令)

$ git push –f 由于这个命令执行没有成功所以换了一个

git push -f origin master下次试试这个

---------------------------------------------------------------------------

gitremote-v： 查看远程仓库详细信息，可以看到仓库名称

git remote remove orign：   删除orign仓库（如果把origin拼写成orign，删除错误名称仓库）

git remote add origin 仓库地址：  重新添加远程仓库地址

gti push -u origin master：   提交到远程仓库的master主干

tag的使用

发布一个版本时，我们通常先在版本库中打一个标签（tag)，方便今后回退到更新前的版本

查看标签

Git tag -l

打标签

git tag -a v1.0 -m '第一次打标签'

删除标签

git tag -d 'v1.0'

查看当前没有add 的内容修改；

git diff

查看已经add 没有commit 的改动

git diff --cached

查看当前没有add和commit的改动：

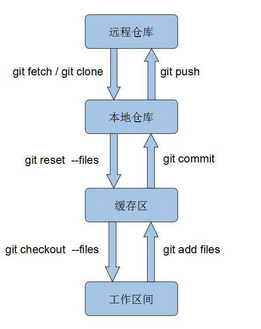
git diff HEAD或者git status

查看任意两个版本之间的改动：

git diff 版本号码1 版本号码2

比较两个版本号码的src 文件夹的差异

git diff 版本号码1 版本号码2 src



2020年11月26日14:48:46

git push origin master --force 远程仓库回退代码

通过网页修改分支权限级别（梁冠勇）

git 远程分支回滚

1、本地代码回滚到上一版本（或者指定版本）

git reset --hard HEAD~1

2、加入-f参数，强制提交，远程端将强制跟新到reset版本

git push -f origin 分支名称