**Git及github使用教程**

1. 前言

作者的个人github网址：<https://github.com/La0bALanG> 欢迎各位大佬前来交流！

1. 什么是git

Git(读音为/gɪt/。)是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

分布式相比于集中式的最大区别在于开发者可以提交到本地，每个开发者通过克隆（git clone），在本地机器上拷贝一个完整的Git仓库。

从一般开发者的角度来看，git有以下功能：

1、从服务器上克隆完整的Git仓库（包括代码和版本信息）到单机上。

2、在自己的机器上根据不同的开发目的，创建分支，修改代码。

3、在单机上自己创建的分支上提交代码。

4、在单机上合并分支。

5、把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。

6、生成补丁（patch），把补丁发送给主开发者。

7、看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。

8、一般开发者之间解决冲突的方法，开发者之间可以使用pull 命令解决冲突，解决完冲突之后再向主开发者提交补丁。

从主开发者的角度（假设主开发者不用开发代码）看，git有以下功能：

1、查看邮件或者通过其它方式查看一般开发者的提交状态。

2、打上补丁，解决冲突（可以自己解决，也可以要求开发者之间解决以后再重新提交，如果是开源项目，还要决定哪些补丁有用，哪些不用）。

3、向公共服务器提交结果，然后通知所有开发人员。

Git的优点：

适合[分布式开发](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%BC%80%E5%8F%91" \t "/home/anwc/文档\\x/_blank)，强调个体。

公共服务器压力和数据量都不会太大。

速度快、灵活。

任意两个开发者之间可以很容易的解决冲突。

离线工作。

Git的缺点：

资料少（起码中文资料很少）。

学习周期相对而言比较长。

不符合常规思维。

代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

1. 什么是github

gitHub是一个面向[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90/20720669" \t "/home/anwc/文档\\x/_blank)及私有[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053" \t "/home/anwc/文档\\x/_blank)项目的托管平台，因为只支持git 作为唯一的版本库格式进行托管，故名gitHub。

gitHub于2008年4月10日正式上线，除了git代码仓库托管及基本的 Web管理界面以外，还提供了订阅、讨论组、文本渲染、在线文件编辑器、协作图谱（报表）、代码片段分享（Gist）等功能。目前，其注册用户已经超过350万，托管版本数量也是非常之多，其中不乏知名开源项目 [Ruby](https://baike.baidu.com/item/Ruby/11419" \t "/home/anwc/文档\\x/_blank) on Rails、[jQuery](https://baike.baidu.com/item/jQuery/5385065" \t "/home/anwc/文档\\x/_blank)、[python](https://baike.baidu.com/item/python/407313" \t "/home/anwc/文档\\x/_blank) 等。

GitHub可以托管各种git库，并提供一个web界面，但它与外国的[SourceForge](https://baike.baidu.com/item/SourceForge/6562141" \t "/home/anwc/文档\\x/_blank)、[Google Code](https://baike.baidu.com/item/Google Code" \t "/home/anwc/文档\\x/_blank)或中国的[coding](https://baike.baidu.com/item/coding/8921246" \t "/home/anwc/文档\\x/_blank)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。

1. Git安装及配置

1.讲这个话题之前，先普及git的几个最基本的概念。

工作区：即本地PC可以看到的目录

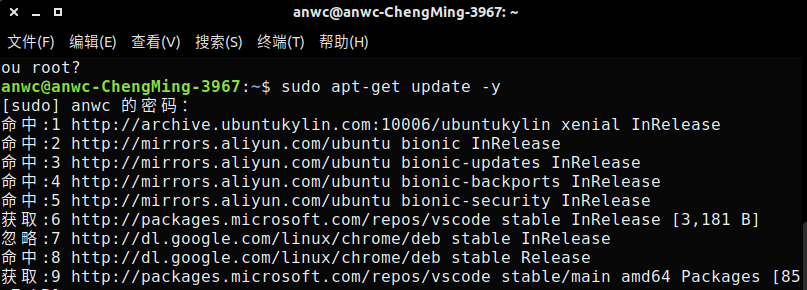
暂存区：一般存放在.git目录下的index文件中

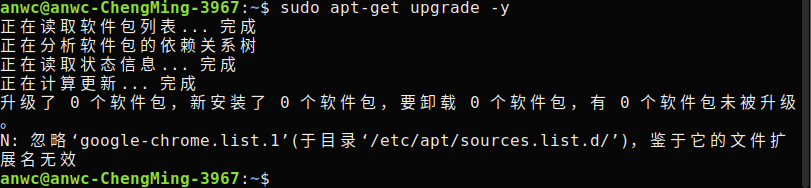
版本库：工作区有一个隐藏的目录.git，此为git的版本库

2.linux下的安装及配置(操作系统：Ubuntu18.10 LTS)

第一步：运行以下命令，确保您的系统和apt包列表完全更新：

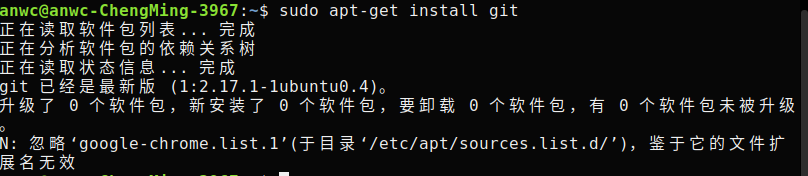
sudo apt-get update -y  
 sudo apt-get upgrade -y





第二步：执行命令，进行安装git(因为我的本机已经安装过，所以提示已经是最新版)：

sudo apt-get install git



安装完毕之后，可以使用以下命令查看当前本机安装的git版本：

Git --version

选区_008

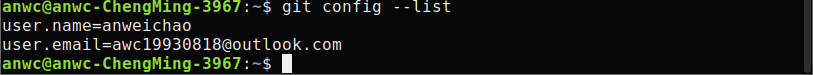
第三步：配置。安装完成后，在git中配置自己的用户名和邮箱地址(我的当前本机已经配置过，所以不再截图做演示了)。使用以下命令实现：

Git config --global user.name ‘your name’

Git config --global user.email ‘your email address’

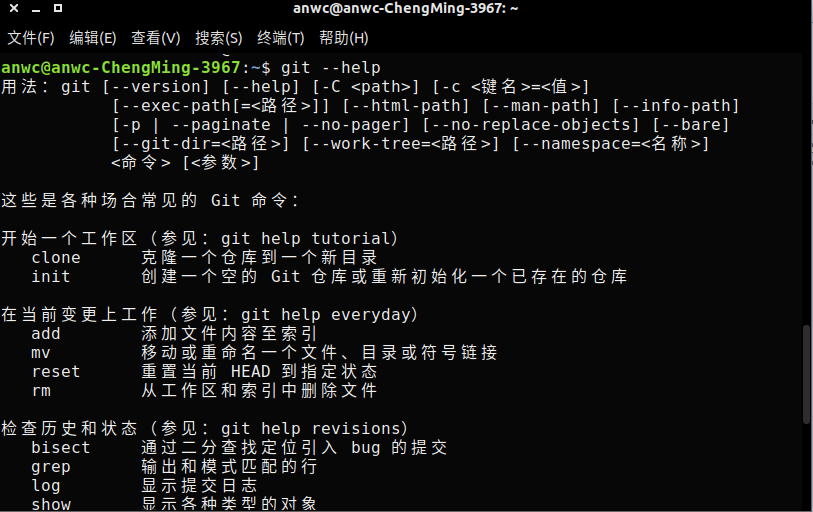
配置完成之后，可以通过以下命令查看配置信息：

Git config --list



有关更多的信息和命令等查看，可以使用以下命令：

Git --help



至此，git的安装及配置已完成。

3.Windows下的安装及配置(操作系统：WIN10 专业版 64位)

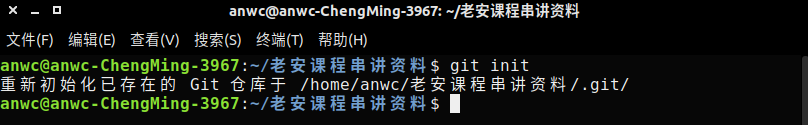
1. Git基本命令操作使用

1.创建仓库

每一次使用git前，都需要先创建仓库。我们可以使用一个已经存在的目录作为git仓库或创建一个空目录。

使用当前目录作为仓库的话，只需要初始化，即可。命令如下：

Git init



2.把文件添加到版本库中

使用命令： git add 文件名/文件夹名

选区_012

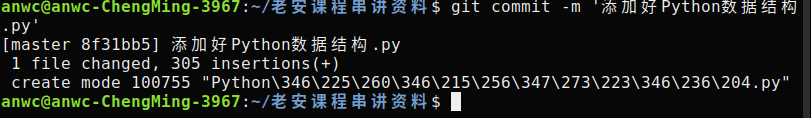
命令运行之后不报任何信息即代表添加到版本库成功

3.把文件提交到仓库

文件添加到版本库之后，需要提交到仓库，才能生成一个新版本。

使用命令：git commit -m ‘此处为注释’

注释的作用是对当前提交的这个版本做一个解释，方便阅读。

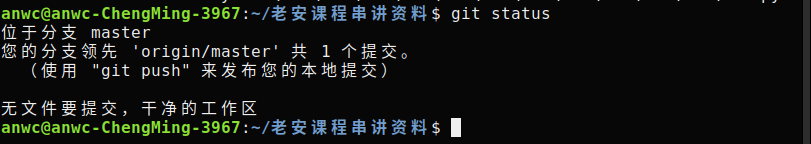


至此，我们已经向仓库中提交了一个文件了(因为我本机有一个现成的仓库，所以这里我举得例子其实是向现有仓库添加新的文件)。

4.git常用操作

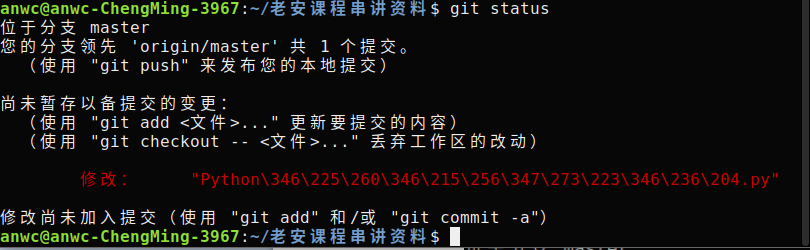
我们可以通过以下命令来查看是否还有文件未提交：

Git status



如果提示没有任何文件未提交，代表成功，否则，代表此文件不支持提交到仓库，比如音视频，二进制文件，特殊加密后的文件等等

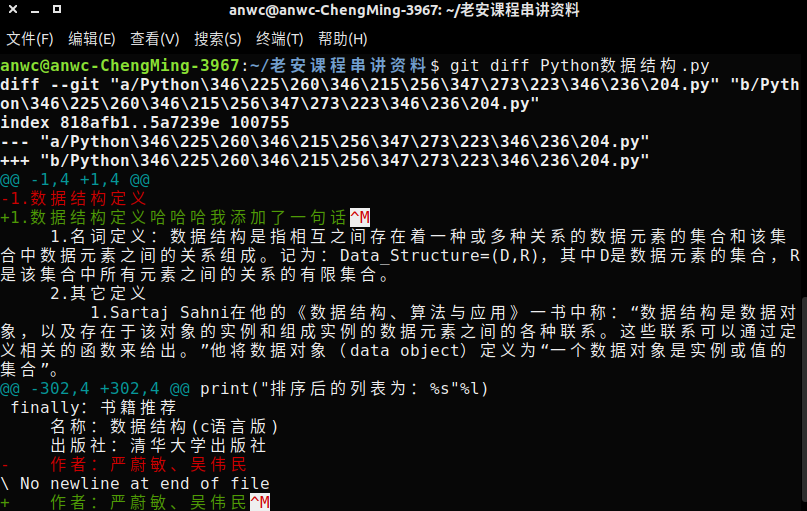
此时，如果我修改一下“Python数据结构.py”这个文件(文件头部添加一个字符串以示修改)，修改完毕保存之后我再使用git status命令查看下结果，结果如下：



如上的结果就告诉我们，这个文件已经被修改，但是是一个未提交的修改。

那么如果我想查看具体修改的是什么内容，可以使用以下命令：

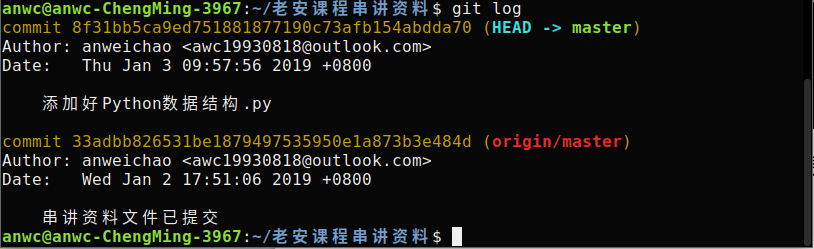
Git diff 文件名



如上结构可以看到，-1那里，是上一个版本，+1那里，就是对上一个版本进行修改的内容。

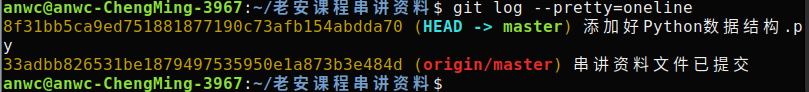
此时，我们对‘Python数据结构.py’这个文件已经做了两次修改了，那么我想查看下历史记录，就可以使用以下命令：

Git log



git log命令显示从最近到最远的显示日志，我们可以看到最近的两次提交，最近的一次是,‘添加好Python数据结构.py’。上一次是‘串讲资料文件已提交’。如果嫌上面显示的信息太多的话，我们可以使用如下命令查看精简信息：

git log --pretty=oneline



那么，如果我想回退到之前的版本，我们可以使用如下两种命令：

git reset --hard HEAD^

这里，一个上尖括号代表回退一个版本，那我想回退两个版本就是两个^^，回退三个版本就是三个^^^

如果想一次性退回到好久之前的某一个版本，可以使用如下命令：

git reset --hard HEAD~你的版本号

例如：

git reset --hard HEAD~23 #回到第23版的版本

5.注意

1.工作区：就是你在电脑上看到的目录，比如目录下的文件(.git隐藏目录版本库除外)。或者以后需要再新建的目录文件等等都属于工作区范畴。  
 2.版本库(Repository)：工作区有一个隐藏目录.git,这个不属于工作区，这是版本库。其中版本库里面存了很多东西，其中最重要的就是stage(暂存区)，还有Git为我们自动创建了第一个分支master,以及指向master的一个指针HEAD。

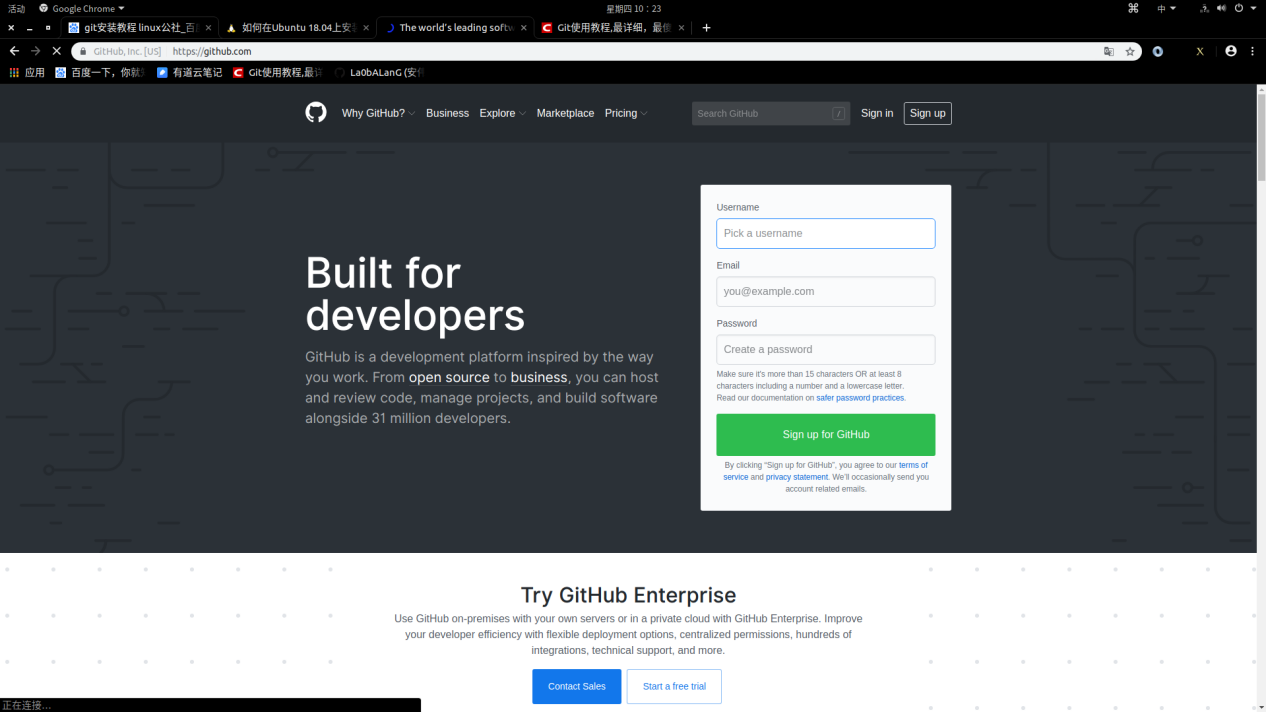
3.文件提交到仓库之后，往后的每一次的对文件的修改，结束之后都要先将文件提交到版本库，再将文件提交到仓库，以将修改后的文件提交到仓库中作为下一个版本。

6.有关git的其他操作，参见资料：

<https://blog.csdn.net/qq_36150631/article/details/81038485>

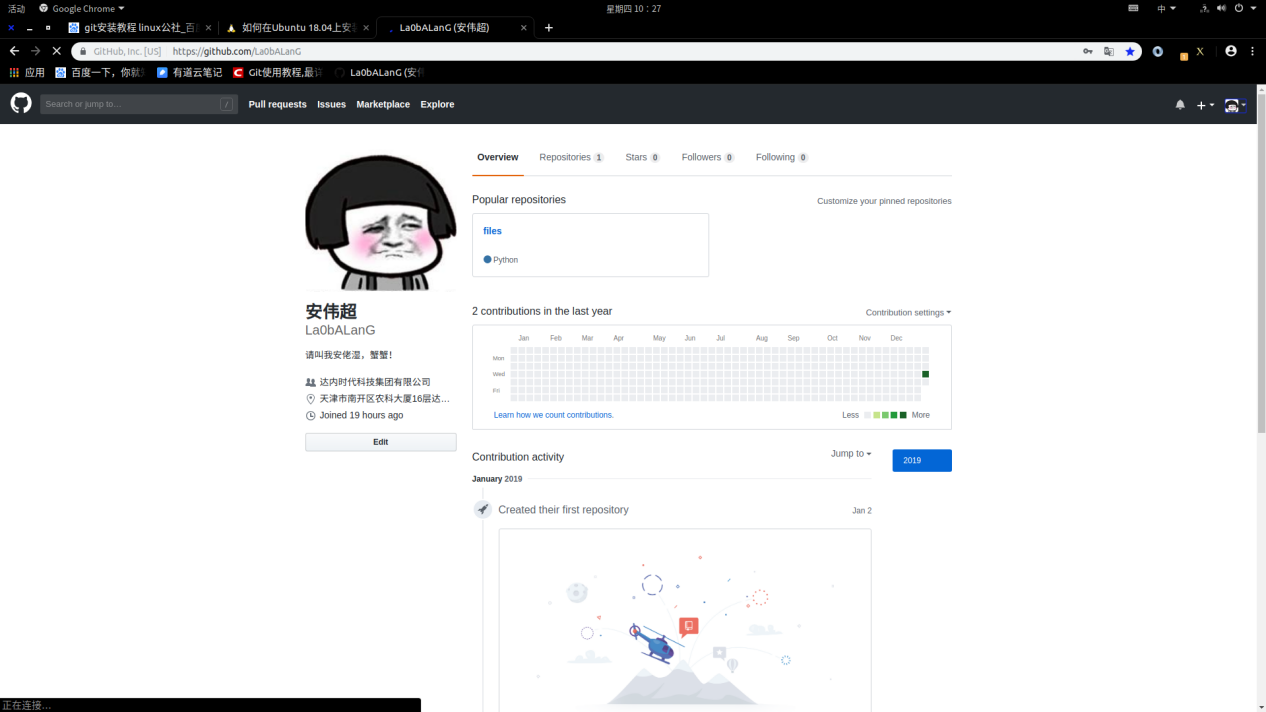
1. Github注册教程

1.浏览器打开github官网：<https://github.com>



2.点击官网首页右上角的’sign up’进行注册。注意：github是全英文网站，虽然看不懂，但尽量不要翻译成中文，看不懂，多用用自然也就熟悉了。注册很简单，输入用户名，邮箱，密码，通过验证，即可注册成功，同时还会让你输入一些信息，如实输入即可。

3.注册完毕，就会进入个人主页页面：



如上，注册完毕。

1. 链接github远程仓库及本地仓库同步远程仓库

1.链接github远程仓库前的设置

由于你的本地Git仓库和github仓库之间的传输是通过SSH加密的，所以需要一点设置。

首先，创建SSH key。回到你的本地PC，打开文件，看看主目录下有没有.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，如果有的话，直接跳过此如下命令，如果没有的话，打开命令行，输入如下命令：

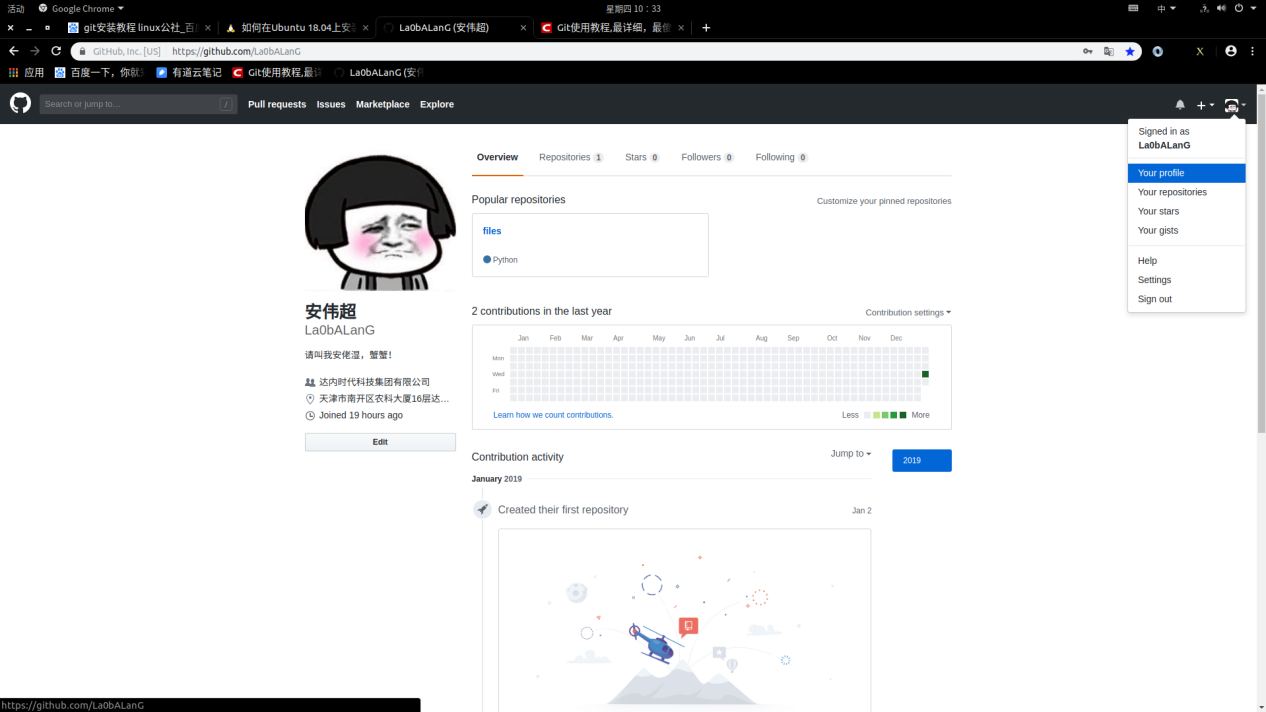
ssh-keygen -t rsa -C [“youremail@example.com”](mailto:\“youremail@example.com\”)

由于我之前运行过一次该命令，所以我的本地PC是有这两个文件的：

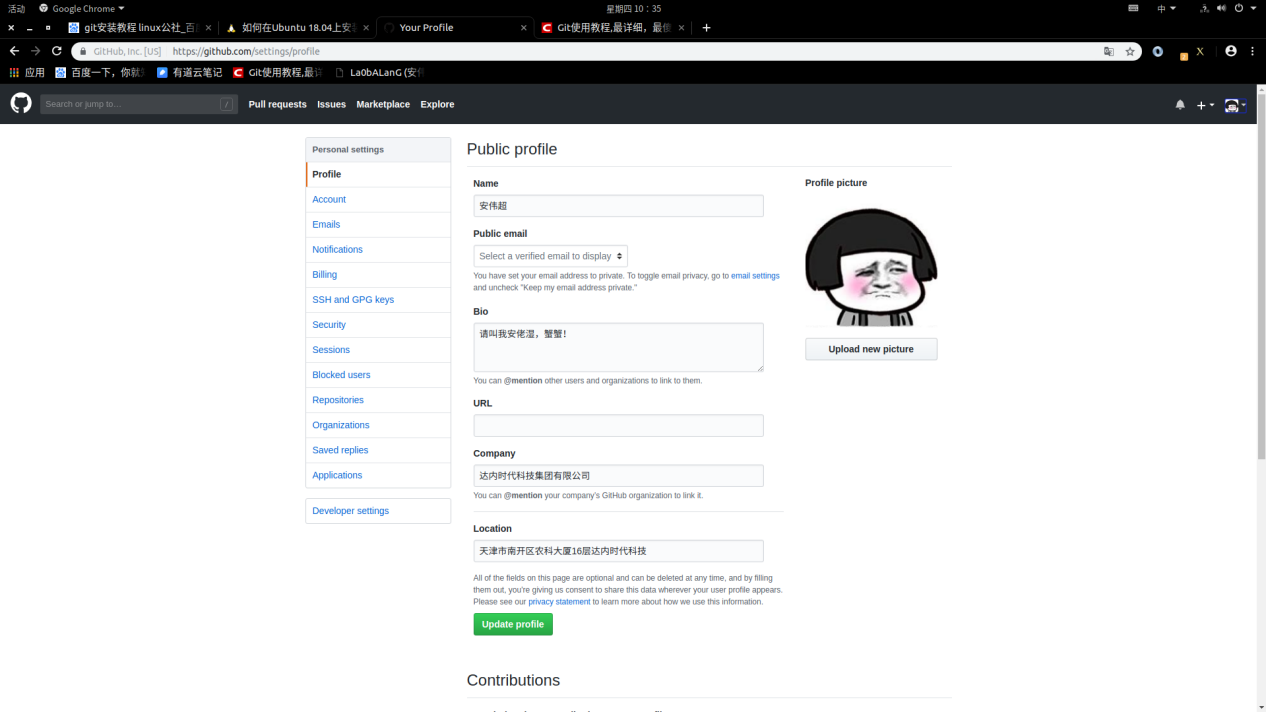


id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

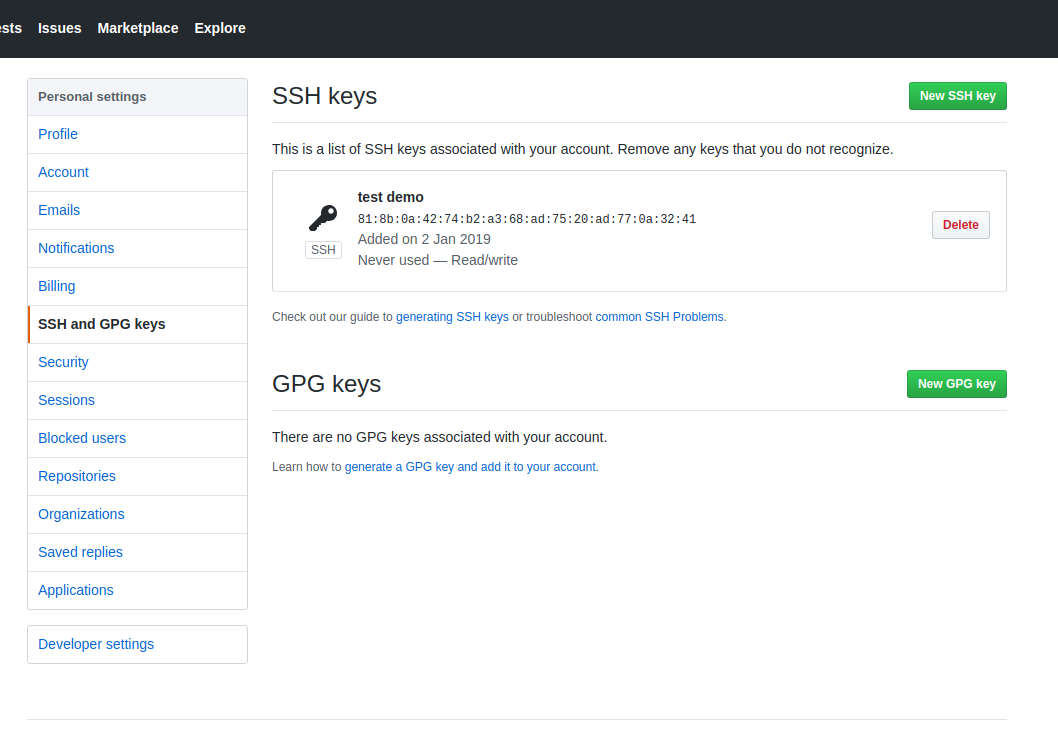
第二步，登录你的github账号，进入个人主页页面，导航右上角有个这个图标：



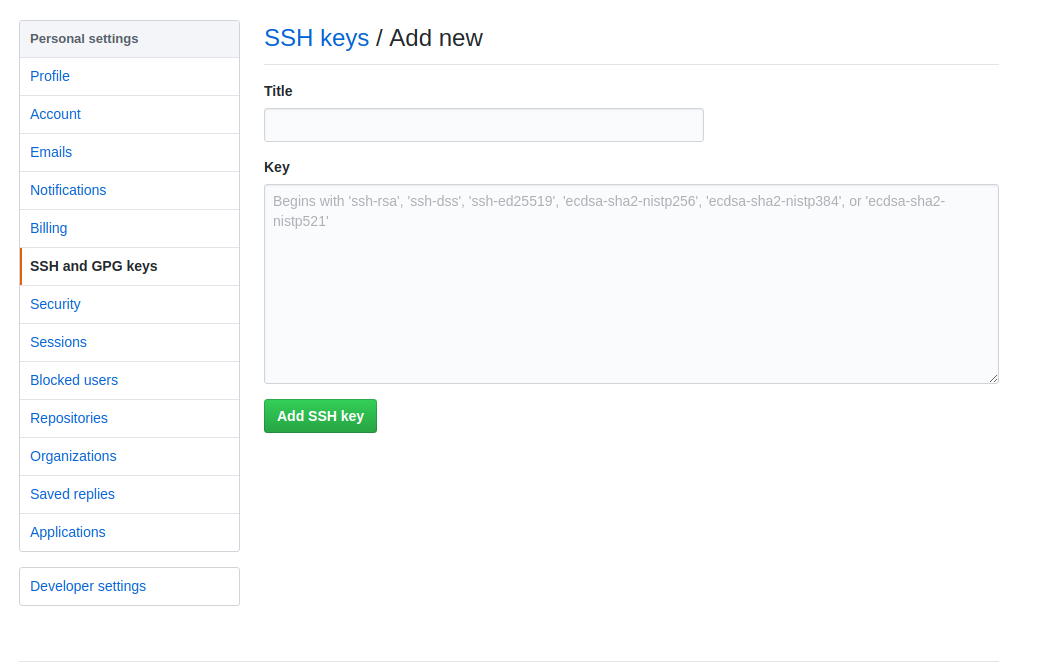
点击’settings’,进入设置页



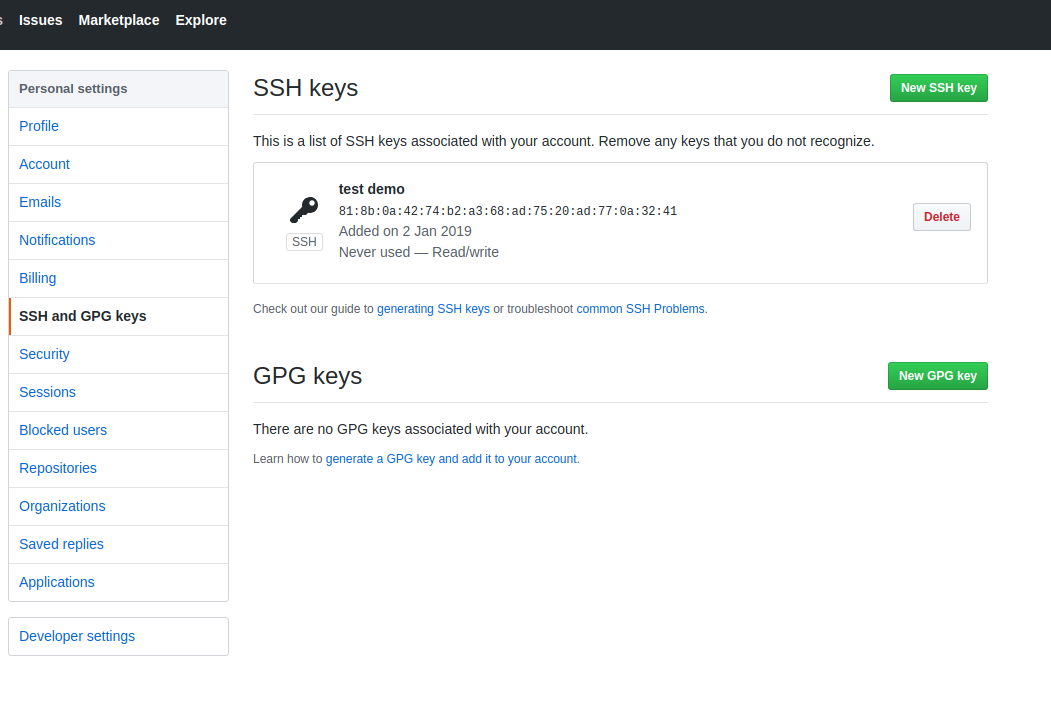
在左边的侧边栏有一个选项叫’ssh and gpg keys’,选择它



点击右上角’new ssh keys’



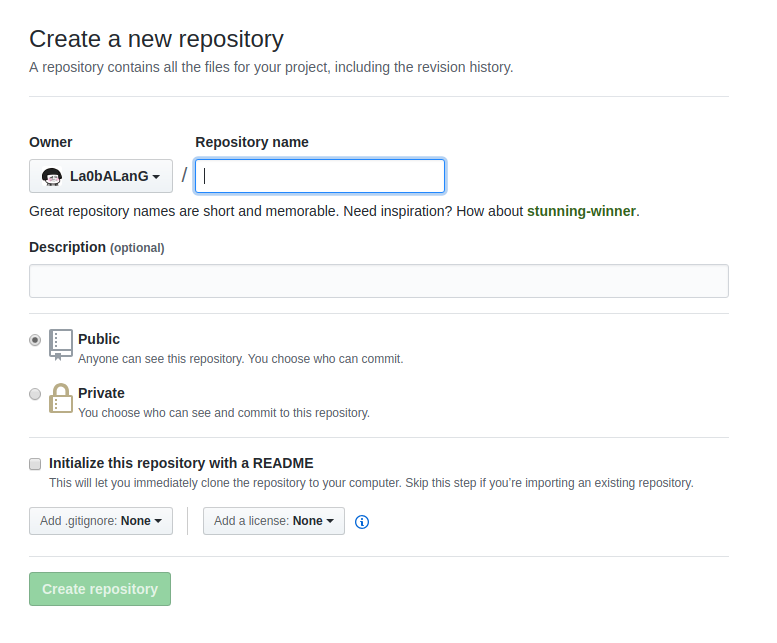
在这一页，标题就随便起名，在key文本框中粘贴id\_rsa.pub这个文件的内容，然后保存，即可。之后，在SSH and GPG keys页面，你就可以看到你刚才添加好的SSH keys了。



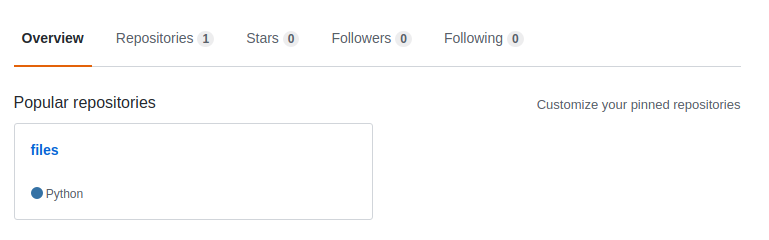
2.添加远程库

比如现在有这样一个情景：我们已经在本地PC创建了一个git仓库，然后我现在想在github上也创建一个git仓库，并且希望这两个仓库进行远程同步，这样github的仓库可以作为备份，又可以其他人通过该仓库来协作。

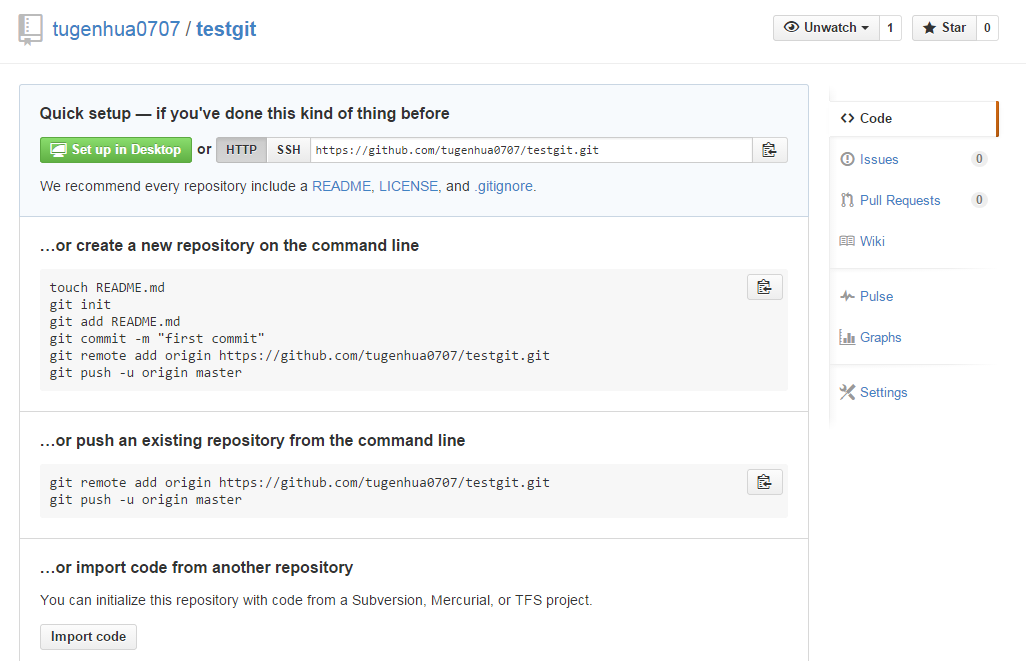
为此，第一步，先在github上创建一个git仓库。进入你的个人github主页，然后在右上角找到“new repository”创建一个新的仓库。如下：



Repository name处输入你的仓库名称，其他全部使用默认选项就行，完事点击create repository即可完成仓库创建。



如上是我之前自己创建好的一个仓库并且已经完成了远程链接。如下是创建出来的空仓库：



目前，在GitHub上的这个仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库。

3.链接远程库

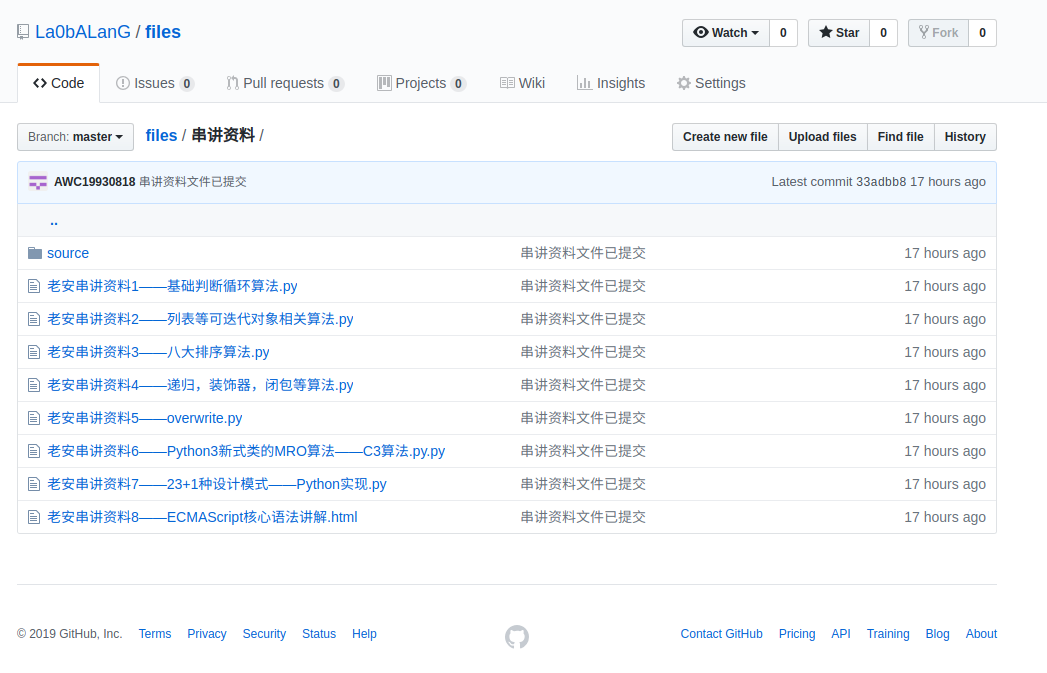
执行如下两条命令：

git remote add origin 你的github的仓库的链接

Git push -u origin master

执行完毕之后，你可以立即在你的github上看到链接远程仓库后同步完成的本地仓库。

这样，你的本地仓库就发布到了github上的远程仓库。你可以把你的个人github主页链接进行分享，其他人就可以通过访问你的链接，查看你当下的所有的仓库的代码或内容了。



注意：从现在起，只要你的本地文件做了修改，就可以通过如下命令：

git push origin master

把本地master分支的最新修改推送到github上了，现在你就拥有了真正的分布式版本库了。

