

Git 配置

环境: Ubuntu 16.04

1. 安装 Git

```
sudo apt-get install git
```

2. 安装安装 ssh, 因为 git 是基于 ssh 协议的, 所以必须先装 ssh

```
sudo apt-get install openssh-server openssh-client
```

安装好 ssh 后, 启动 ssh 服务: `sudo /etc/init.d/ssh restart`

3. 查看 git 版本号

```
git --version
```

```
weihanlin@ubuntu:~$ git --version  
git version 2.7.4
```

4. 配置邮箱和用户名

```
git config --global user.name "你的 github 用户名"
```

```
git config --global user.email "你的 github 用户邮箱"
```

5. 配置公钥

```
ssh-keygen -C '你的 github 用户邮箱' -t rsa
```

保存公钥文件的位置选择默认 (按回车键) 即可。见红色框。

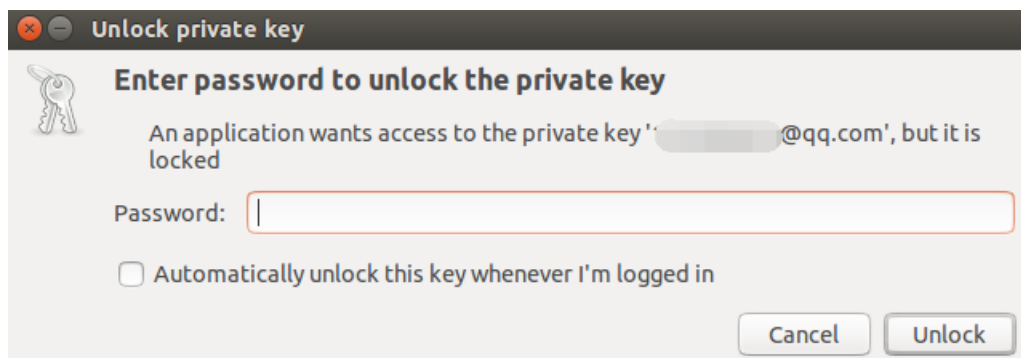
这里需要输入一个字符串作为种子来生成公钥, 见黄色框。随便输或者输一个自己记得住的字符串即可。

```

weihanlin@ubuntu:~$ ssh-keygen -C 'you email address@gmail.com' -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/weihanlin/.ssh/id_rsa):
Created directory '/home/weihanlin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/weihanlin/.ssh/id_rsa.
Your public key has been saved in /home/weihanlin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:qrnVmGjsr6fpXpsegGEU0390vaAxH7cof9U9KYWSByo you email address@gmail.com
The key's randomart image is:
+---[RSA 2048]---+
|  +0      oo . |
|  . .    + =ooo . |
|  o  . E B =oo..o |

```

这个字符串和公钥是对应的，如果字符串换了，会生成不同的公钥。字符串本身是什么并不重要。但是最好记住这个字符串，有时候系统弹出窗口让你输入这个字符串。



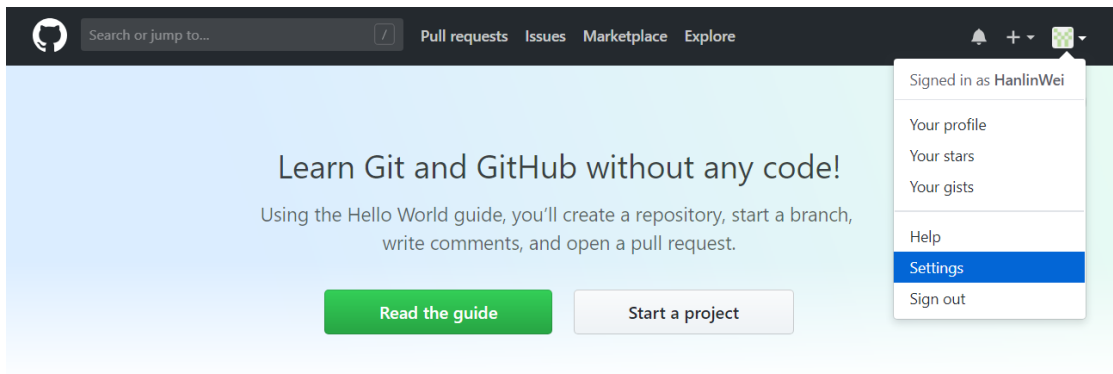
6. 进入刚才保存公钥的目录，找到 `id_rsa.pub` 文件。打开后复制其中的内容。

```
cd ~/.ssh
```

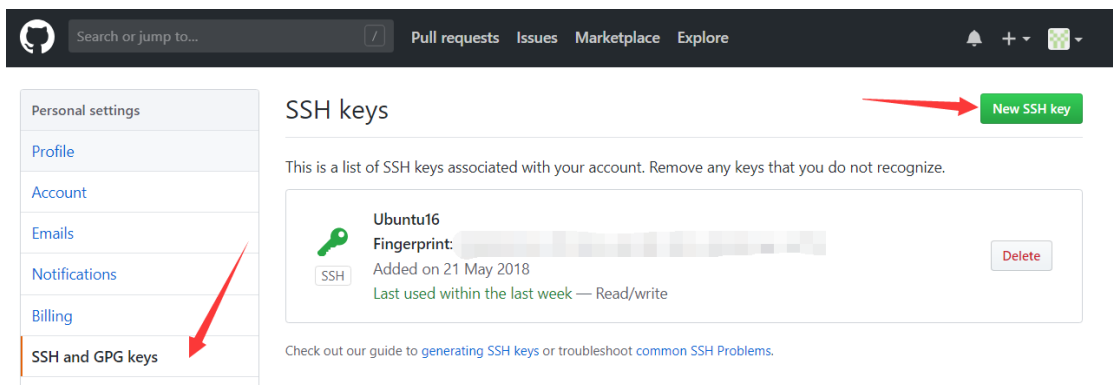
```
gedit id_rsa.pub
```

7. 登录 github，找到 setting

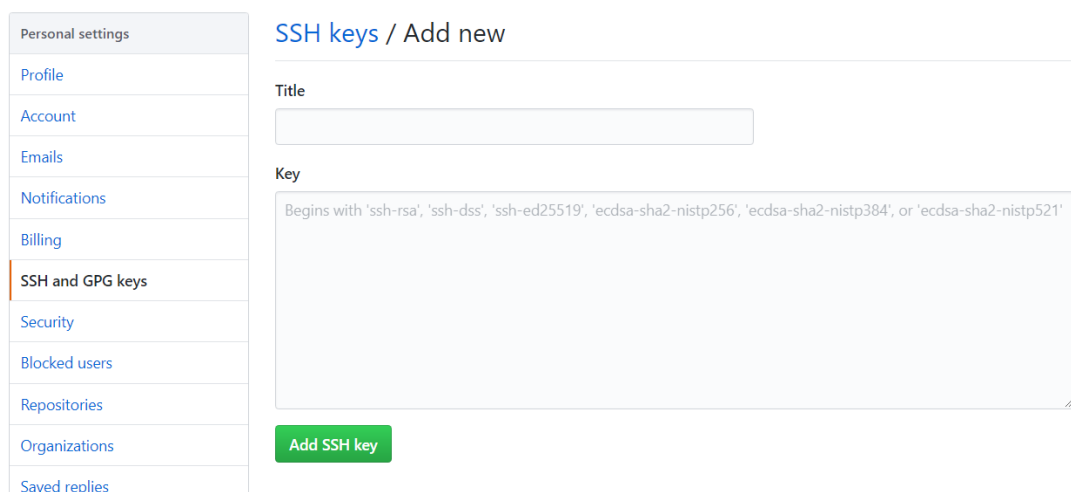
<https://github.com/>



8. 进入设置后，左边找到 SSH and GPG keys，然后右上角点击 New SSH keys。




9. 粘贴刚才复制的公钥，并给该公钥起个你喜欢的名字。结束后你就可以看到 7 中的图片上我已经注册好的公钥。



10. Ubuntu 尝试连接 git。

ssh -v [git@github.com](https://github.com)

连接成功后网页上钥匙会变成绿色，下面会出现一行小字。命令行也可以看到有建立连接和收发数据的记录。

**Ubuntu_14**
Fingerprint: b9:14:d5:96:7e:11:06:43:de:35:54:a8:c6:24:5c:52
Added on 1 Jun 2018
[Last used within the last week — Read/write](#)
[Delete](#)

```
debug1: channel 0: free: client-session, nchannels 1
Connection to github.com closed.
Transferred: sent 3632, received 1808 bytes, in 1.6 seconds
Bytes per second: sent 2260.8, received 1125.4
debug1: Exit status 1
```

以上参考：

<https://www.linuxidc.com/Linux/2016-09/135527.htm>

https://blog.csdn.net/yadong_word/article/details/52628836

Git 建立远程仓库

这一部分基本上参考百度经验，我只是大自然的搬运工，另外加入了一些我自己做的过程中的体会。直接看原贴也差不多。

<https://jingyan.baidu.com/album/2fb0ba4091a21c00f2ec5fbf.html?picindex=13>

明确一下概念：

https://blog.csdn.net/qq_37311616/article/details/80497604

工作区，暂存区

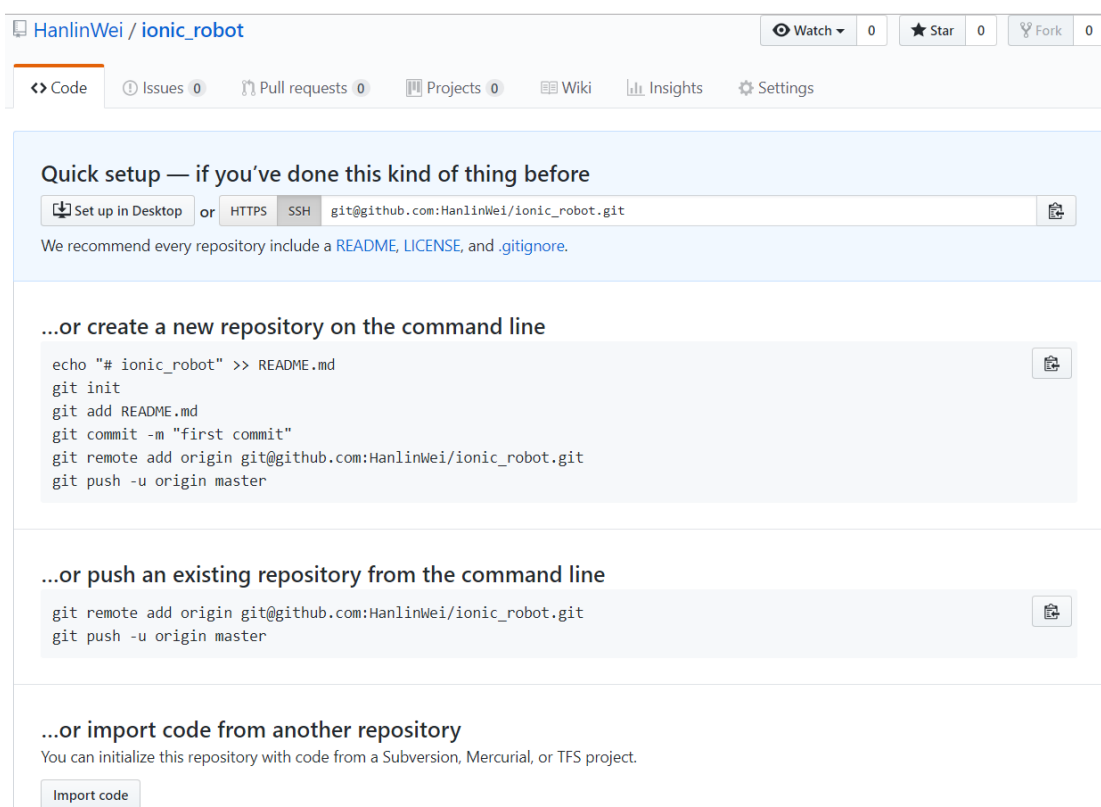
- 工作区 Workspace：就是你在电脑里能看到的目录，即你代码放的那个文件夹。即时性强，对文件的所有更改都会立刻提现在这里。
- 版本库：工作区有一个隐藏目录.git，这个不算工作区，而是 Git 的版本库。
- 暂存区 Index / Stage：git add 以后，当前对文件的更改会保存到这个区

- 本地仓库 Repository : git commit 以后 , 当前暂存区里对文件的更改会提交到本地仓库

- 远程仓库 Remote : 远程仓库名一般叫 origin。git push 以后 , 本地仓库里优先于远程仓库的 commit 会被 push 到远程仓库

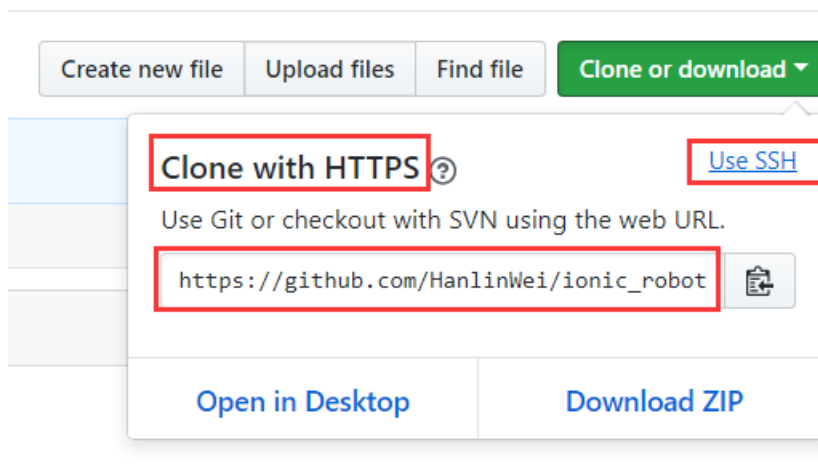
11.添加一个远程仓库。点击 GitHub 页面上的 New repository 按钮。然后设置一下仓库的名字和其他属性。

设置完成后会出现一个简单的教程 ,接下来的操作基本上和这个教程差不多。



HTTP 或 SSH 的链接是用来连接本地仓库 (本地文件系统) 和 git 仓库的。

日后在项目的右上角有一个绿色的 clone or download 按钮 ,可以找到项目的 HTTP 或 SSH 的链接。



12. 在本地任意位置建立一个文件夹，我建的文件夹名称为 MyProject，然后在终端中切换到该目录。

13. 在终端中输入：

git init

回车之后，可以看到 MyProject 目录中出现了一个 .git 的文件夹（默认是隐藏的。Ubuntu 下可以 ctrl+H 显示）。这个文件夹含有你初始化的 Git 仓库中所有的必须文件。初始化完毕后，我们就可以添加跟踪我们的文件了。

14. 建立 Read.md 文件

echo "# 这是一个 git 简单使用的经验介绍项目" >> README.md

接下来，我们将 README.md 加入到我们的跟踪列表里，这样 git 就会跟踪该文件的变化。输入命令：

git add README.md

回车后，再输入：

git commit -m "添加了 README 文件"

这样就将 README.md 提交到本地仓库了。

不过此时，我们的 github 上的远程仓库依然是空的。

15. 下面，我们将远程仓库添加进来：

```
git remote add origin https://github.com/HanlinWei/ionic_robot.git
```

上面这条命令的格式后面的链接就是之前我们刚创建的项目的时候 quick setup 里面给的链接。HTTP 和 SSH 都可以。链接的格式如下，知道格式的话就算日后忘记了链接也可以重新找到。

SSH: git@github.com:用户名/项目名称.git

HTTP: https://github.com/用户名/项目名称.git

origin 就是我们给远程仓库起的名字。

然后，我们就可以将本地提交推送到远程仓库了。

输入命令：git push -u origin master

回车后，终端会询问你用户名和密码，这个是我们刚才注册的 github 的用户名和密码，在输入密码时不会有任何显示。

master 是本地仓库，在上图中可以看到。

16. 提交后我们再看看 github 上的仓库发生了什么。我们可以看到，远程仓库已经有了 README.md 文件，而且我们的注释也有显示。还能看出，这个 README.md 文件的内容是显示在仓库主页面的。



17. 总结一下 ,修改项目后如何提交 ? 假如修改了 main.cpp 文件输入下面的命令 :

(提交三连)

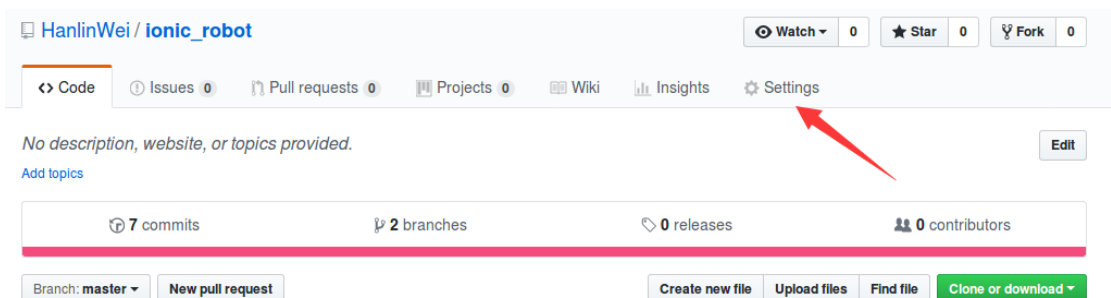
git add main.cpp

git commit -m "给 main.cpp 添加注释"

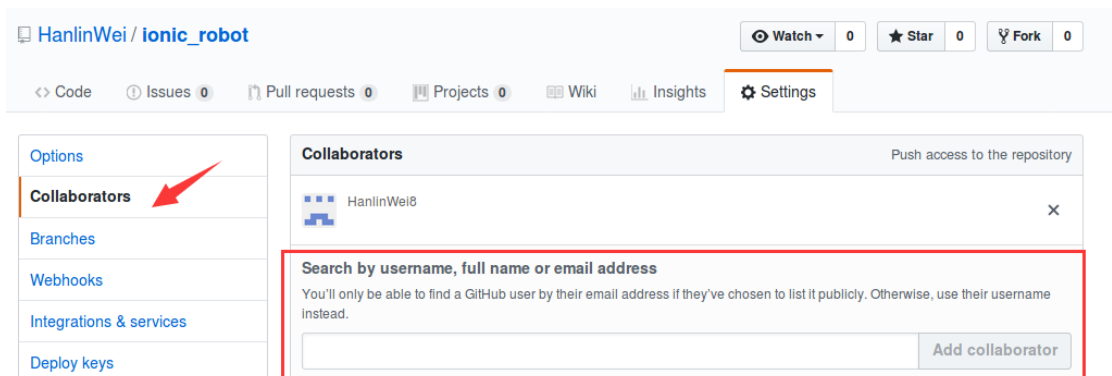
git push -u origin master

Git 添加其他用户一起开发

18. 打开仓库页面 , 找的仓库的 Setting



19.找到 Collaborators，搜索用户，然后会出现一个邀请的链接，把链接发给被邀请人就可以了。



Git 分支管理

20.在 GitHub 网站上使用得最多的就是这种工作流。人们可以复制（fork 亦即克隆）某个项目到自己的列表中，成为自己的公共仓库。随后将自己的更新提交到这个仓库，所有人都可以看到你的每次更新。这么做最主要的优点在于，你可以按照自己的节奏继续工作，而不必等待维护者处理你提交的更新；而维护者也可以按照自己的节奏，任何时候都可以过来处理接纳你的贡献。

同样，开发者有自己的公共仓库，实现方法就是创建 branch。切换到某个分支后，push 和 commit 只会记录在这个分支下。分支相当于一个时间线，一条 if 剧情线。

参考：https://blog.csdn.net/v_king_/article/details/14052039

21.查看分支

git branch

22.创建分支

```
git branch mybranch
```

23.切换到某个分支

```
git checkout mybranch
```

24.删除分支（一般别干这种事，留着不碍事）

```
git branch -d mybranch //如果该分支没有合并到主分支会报错  
或者  
git branch -D mybranch //强制删除
```

25.分支合并

例如，我们之前建立了 mybranch 进行我们的开发。测试通过后，我们想要把 mybranch 合并到主分支 master 里，进行代码的同步。

首先切换到 master：`git checkout master`

合并：`git merge mybranch`

查看分支之间的区别：`git diff master mybranch`

此时可能什么都不会输出，表示两者已经同步了。

但是这是假象，只是在本地的两条 branch 已经同步了，远程仓库里两条 branch 还是不一样的。可以在 Github 网页端确认。

此时需要把我们合并后的 master push 到远程仓库。

```
git push -u origin master
```

直接 push 可能会报错,如下图。

```
weihanlin@ubuntu:~/Documents/ionic_app$ git push -u origin master
To git@github.com:HanlinWei/ionic_robot.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:HanlinWei/ionic_robot.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push -help' for details.
```

这是因为如果两个开发者从中心仓库克隆代码下来，同时作了一些修订，那么只有第一个开发者可以顺利地把数据推送到共享服务器。第二个开发者在提交他的修订之前，必须先下载合并服务器上的数据，解决冲突之后才能推送数据到共享服务器上。

此时应该先 pull，再 push。

git pull

git push -u origin master

以上参考：

https://blog.csdn.net/qq_35332692/article/details/78964785

Git 回退

26. 参考：<https://blog.csdn.net/grace666/article/details/44704291>

27.

