# [root@room9pc01 aubi]# git remote add origin https://github.com/AuBiSY/aubi\_deku.git

# Git简介

Git是目前世界上最先进的分布式版本控制系统。

# Git与SVN主要的区别

Git是分布式版本控制系统

没有中央服务器的，每个人的电脑就是一个完整的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。

既然每个人的电脑都有一个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件A，其他人也在电脑上改了文件A，这时，你们两之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

SVN是集中式版本控制系统

版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑，所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央服务器。

集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果在互联网下，如果网速慢的话，就纳闷了。

# 下载安装git

# 配置用户名和邮箱

git config --global user.name "xxx"

git config --global user.email "1111@qq.com"

# 配置GitHub

## 注册

https://github.com/

## 添加SSH Key

### 生成SSH Key

ssh-keygen -t rsa -C "1111@qq.com"

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

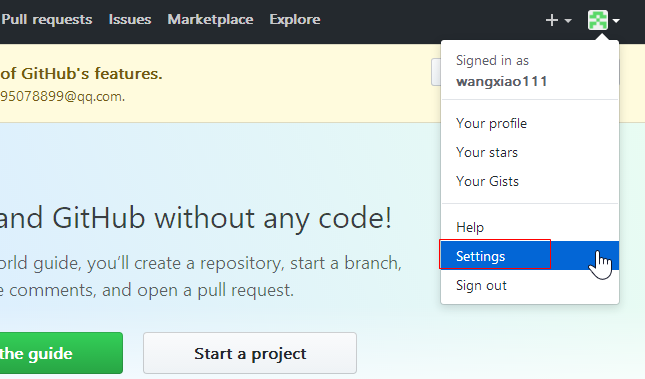
Your public key has been saved in /root/.ssh/id\_rsa.pub.

### 复制里面的 key

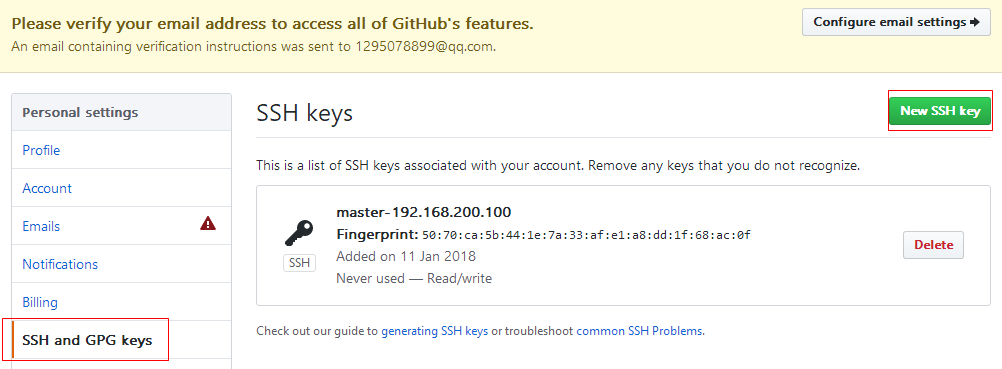
cat /root/.ssh/id\_rsa.pub

### 进入 Account => Settings

https://github.com/



添加SSH key



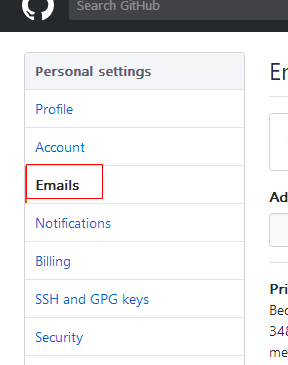
### 验证是否可连接

ssh -T git@github.com

Warning: Permanently added 'github.com,192.30.255.113' (RSA) to the list of known hosts.

Hi xxx! You've successfully authenticated, but GitHub does not provide shell access.

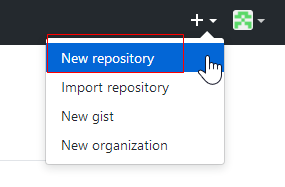
## 验证邮箱

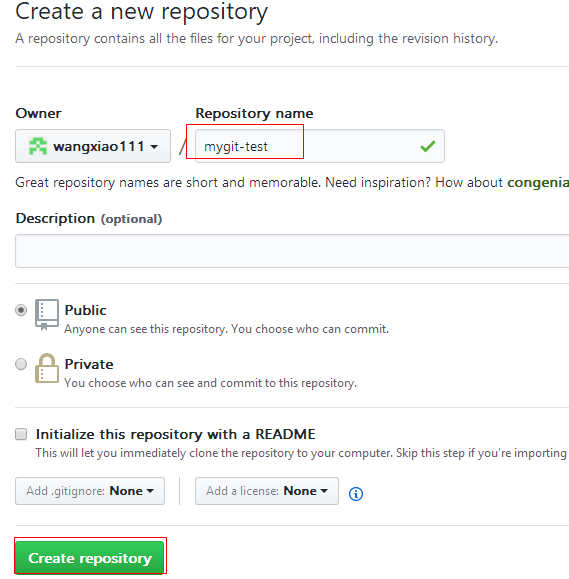


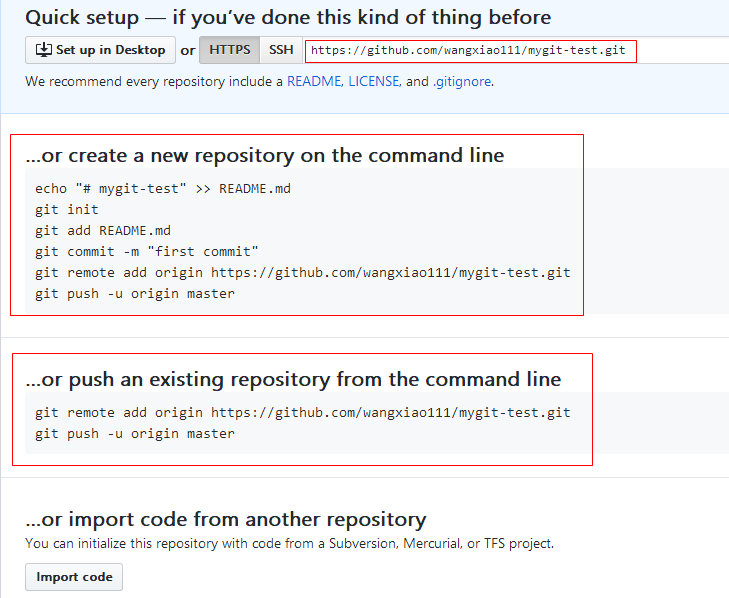
点击发送，然后到邮箱里验证

# 创建仓库

## GitHub上创建库







## 关联一个远程库

git remote add origin https://github.com/xxx/mygit-test.git

第一次推送master分支的所有内容

git push -u origin master

以后推送

git push origin master

# 工作区和暂存区

**工作区：**就是你在电脑里能看到的目录。

**暂存区：**英文叫stage, 或index。一般存放在 ".git目录下" 下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

**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。



添加文件到Git版本库，分两步执行的：

第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

# 创建版本库

## 简介

版本库又名仓库，英文名repository,你可以简单的理解一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改，删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻还可以将文件”还原”。

图片，视频这些二进制文件，虽能也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是知道图片从1kb变成2kb，但是到底改了啥，版本控制也不知道。

## 新建mygit版本库

mkdir mygit

cd mygit

git init # 把这个目录变成git可以管理的仓库

ll -la

drwxr-xr-x 7 root root 119 Jan 11 17:36 **.git**

# 文件基本提交流程

## 创建/修改文件

vim readme.txt

添加一行:

111

222

## 添加文件到暂存区

git add readme.txt

## 提交到当前分支

把暂存区的所有内容提交到当前分支

git commit -m "first commit"

[master (root-commit) 971f0d4] first commit

1 file changed, 2 insertions(+)

create mode 100644 readme.txt

# 查看仓库当前的状态

git status

查看工作区和暂存区的文件区别

git diff readme.txt

diff --git a/readme.txt b/readme.txt

index a30a52a..641d574 100644

--- a/readme.txt

+++ b/readme.txt

@@ -1,2 +1,3 @@

111

222

+333

显示从最近到最远的显示日志(每次提交的内容)

git log

下面的commit id是一个hash值

commit 86324f15d9bca7d6d0e2c2635074593b747481c6

Author: xxx <111@qq.com>

Date: Thu Jan 11 19:37:05 2018 +0800

remove tests.txt

commit eb017c96f497c69932a4477b8da263fa72813583

Author: xxx <111@qq.com>

Date: Thu Jan 11 19:36:18 2018 +0800

add

commit f6b4b04400f04350ba95b38e753a88e6161f2d85

Author: xxx <111@qq.com>

Date: Thu Jan 11 19:18:40 2018 +0800

add 111

# 撤销修改

丢弃工作区的修改

git checkout -- readme.txt

把暂存区的修改撤销掉（unstage），重新放回工作区

git re**set** HEAD readme.txt

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。

用HEAD时，表示最新的版本。

# 删除文件

git rm test.txt

git **commit** -m "remove test.txt"

# 版本回退

## 查看日志

显示从最近到最远的显示日志(每次提交的内容)

git log

下面的commit id是一个hash值

commit 86324f15d9bca7d6d0e2c2635074593b747481c6

Author: xxx <111@qq.com>

Date: Thu Jan 11 19:37:05 2018 +0800

remove tests.txt

commit eb017c96f497c69932a4477b8da263fa72813583

Author: xxx <111@qq.com>

Date: Thu Jan 11 19:36:18 2018 +0800

add

commit f6b4b04400f04350ba95b38e753a88e6161f2d85

Author: xxx <111@qq.com>

Date: Thu Jan 11 19:18:40 2018 +0800

add 111

## 根据日志信息回退

退回上一个版本

git reset --hard HEAD^

退回上上一个版本

git reset --hard HEAD^^

往上100个版本写100个^比较容易数不过来，所以写成HEAD~100

用commit id回退

git reset --hard f6b4b04400f04350ba95b38e753a88e6161f2d85

## 查看历史命令和commit id

git reflog

86324f1 HEAD@{0}: commit: remove tests.txt

eb017c9 HEAD@{1}: commit: add

f6b4b04 HEAD@{2}: commit: add 111

438273a HEAD@{3}: commit: add 111

06e1f62 HEAD@{4}: commit: add 1 line '333'

971f0d4 HEAD@{5}: commit (initial): first commit

用commit id回退

git reset --hard 06e1f62

