#### Git安装

#### 2. 本地创建版本库

##### 2.1 新建空目录

什么是版本库呢？版本库又名仓库，英文名repository，你可以简单理解成一个目录，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可以“还原”。所以，创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录，方式有如下两种：

（1）通过手动新建文件夹

（2）通过命令行的形式创建文件夹（D:\GitRepositories）

1. 切换到主目录

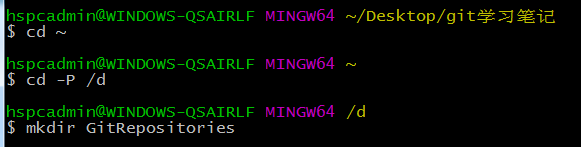
cd / （或者 cd ~）

2. 切换到D盘

cd –P /d

3. 新建GitRepositories 文件夹

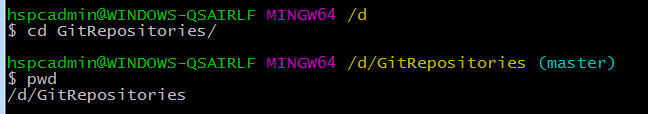
mkdir GitRepositories



4. 进入GitRepositories目录并查看当前目录

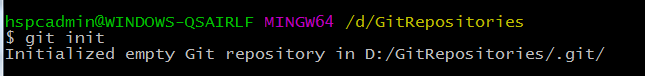
cd GitRepositories

pwd （pwd命令用于显示当前目录）

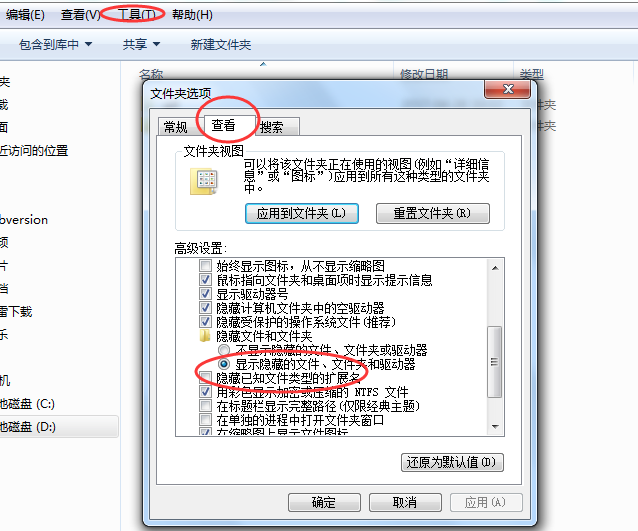


##### 2.2 将目录初始化成一个空仓库

通过 git init 命令初始化一个空仓库



瞬间Git就把仓库建好了，而且告诉你是一个空的仓库（empty Git repository），细心的读者可以发现当前目录下多了一个.git的目录，这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。Windows下可以通过修改 工具 -> 查看 -> 显示隐藏的文件、文件夹和驱动器 显示隐藏的文件。

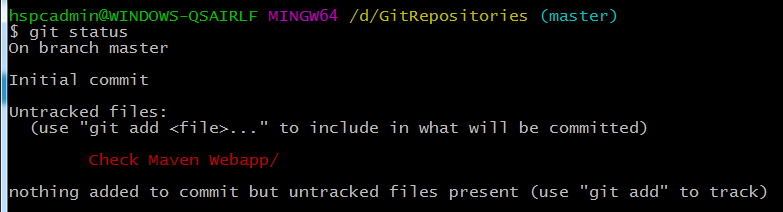


#### 3. 将资源添加并提交

将需要进行版本控制的资源文件拷贝至GitRepositories 目录下，等待后续的操作：

##### 3.1 查看资源的状态

git status



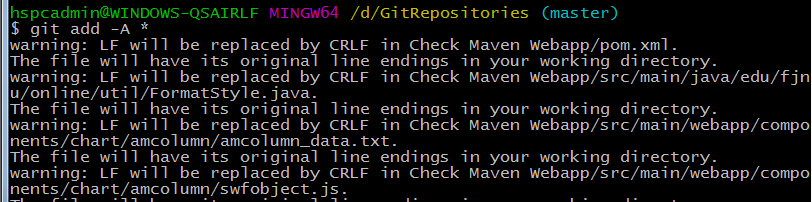
##### 3.2 将资源文件添加到仓库

（1）添加单个资源文件

git add swfobject.js

（2）批量添加所有的文件

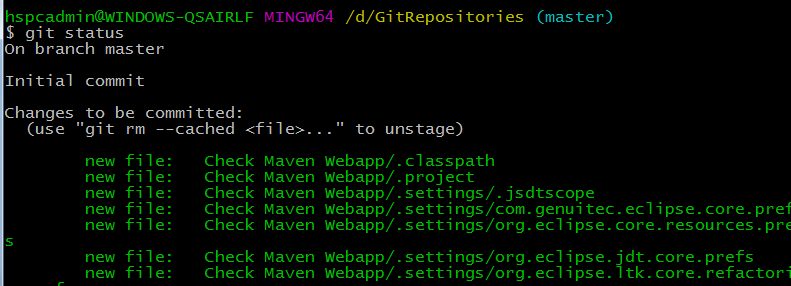
git add –A \*



##### 3.3 将资源文件提交到仓库

（1）查看资源状态

git status



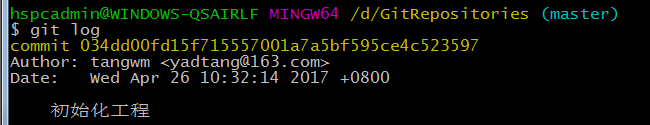
（2）提交到仓库

git commit –m “初始化工程”

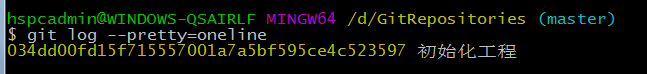


（3）查看提交日志

1. git log



2. git log –pretty=oneline



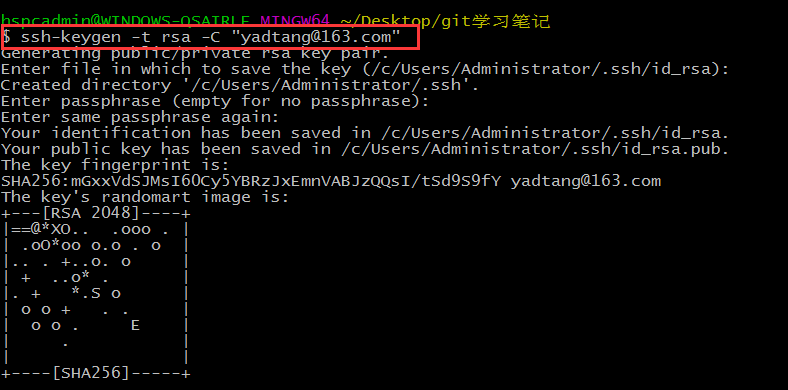
#### 4. 版本回退

#### 5. 远程仓库（GitHub）

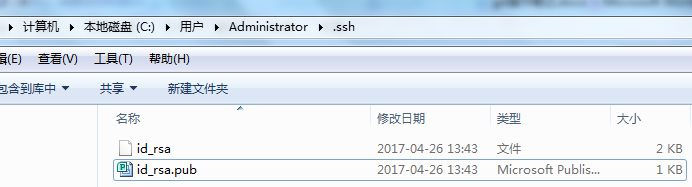
##### 5.1 创建SSH Key

在用户主目录下，看看有没有.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，如果已经有了，可直接跳到下一步。如果没有，打开 Shell（Windows 下打开 Git Bash），创建 SSH Key：

ssh-keygen –t rsa –C “yadtang@163.com”



你需要把邮件地址换成你自己的邮件地址且该邮箱为GitHub中注册的邮箱，然后一路回车，使用默认值即可，由于这个Key 也不是用于军事目的，所以也无需设置密码。如果一切顺利的话，可以在用户主目录（C:\Users\Administrator\.ssh）里找到 .ssh 目录，里面有 id\_rsa 和 id\_rsa.pub两个文件，这两个就是 SSH Key 的秘钥对， 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.



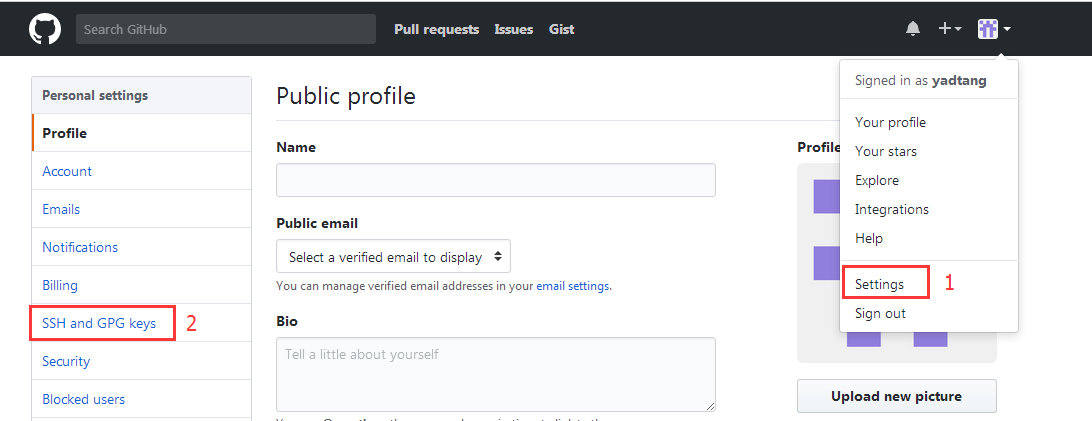
##### 5.2 添加SSH Key

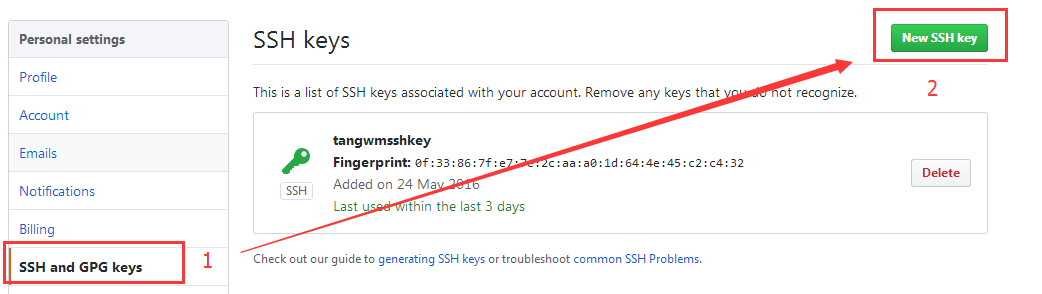
登陆 GitHub，打开“Settings”，“SSH Keys”页面,然后，点“Add SSH Key”，填上任意 Title，在 Key 文本框里粘贴 id\_rsa.pub 文件的内容：

1. **．登录GitHub**

访问<https://github.com/> , 如果有账号则登录，无则注册一个账号

1. **．选择 Settings -> SSH and GPG keys –> New SSH Key 添加一个SSH Keys**

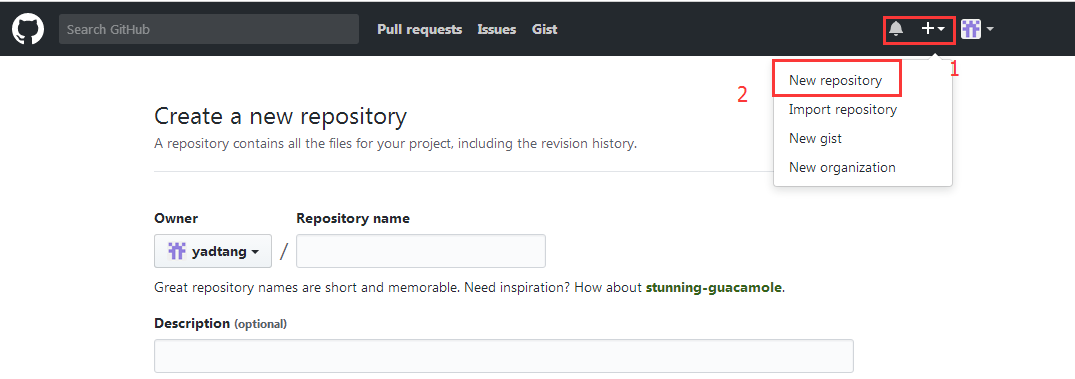




##### 5.3 添加远程仓库（GitHub）

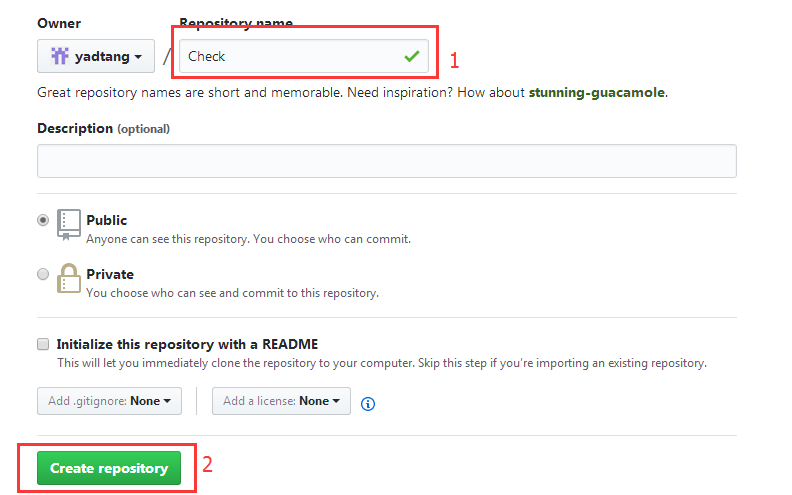
在本地创建了一个Git仓库后，又想在GitHub创建一个Git仓库，并且让这两个仓库进行远程同步，这样，GitHub上的仓库既可以作为备份，又可以让其他人通过该仓库来协作，需要如下操作:

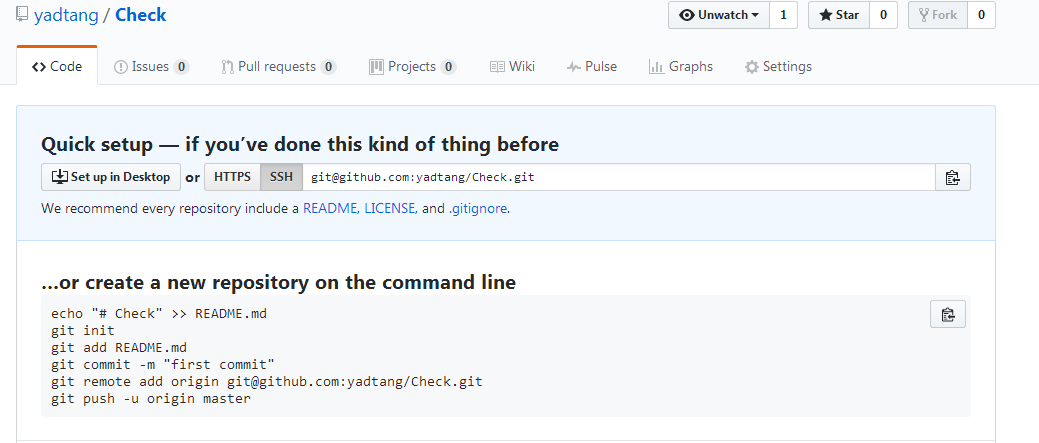
**（1）登陆GitHub，然后，在右上角找到“New repository”按钮，创建一个新的仓库：**



**（2）填写 Repository name**

在Repository name填入Check，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：





目前，在GitHub上的这个Check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。

##### 5.4 将本地仓库与远程仓库关联

**（1）仓库关联**

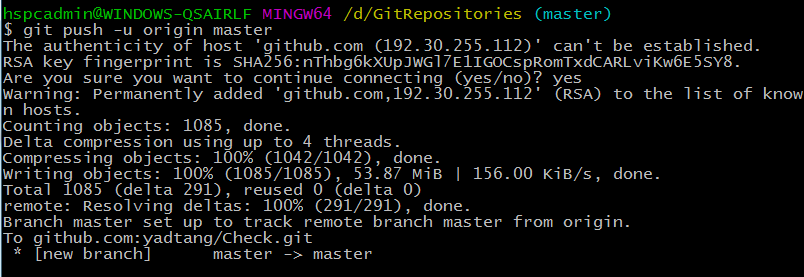
$ git remote add origin git@github.com:yadtang/Check.git



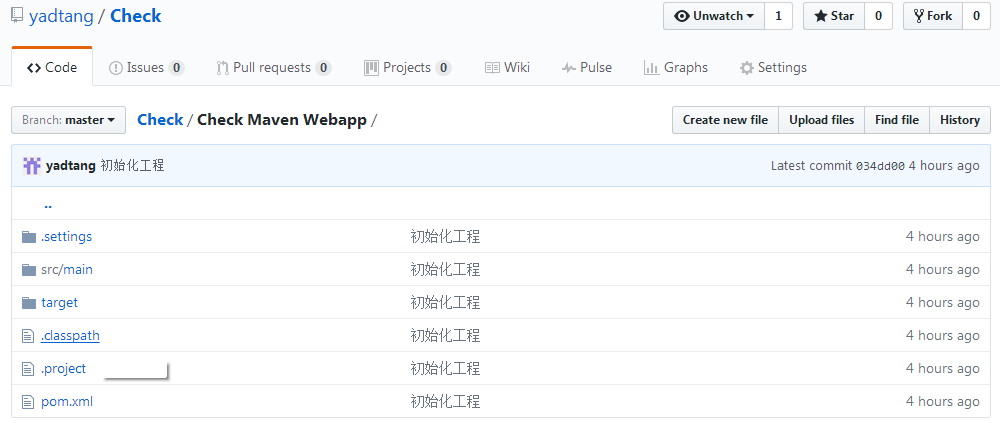
请千万注意，把上面的yadtang替换成你自己的GitHub账户名，否则，你在本地关联的就是我的远程库，关联没有问题，但是你以后推送是推不上去的，因为你的SSH Key公钥不在我的账户列表中。添加后，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的，但是origin这个名字一看就知道是远程库。

**（2）本地库的所有内容推送到远程库上**

$ git push -u origin master



把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。推送成功后，可以立刻在GitHub页面中看到远程库的内容已经和本地一模一样：



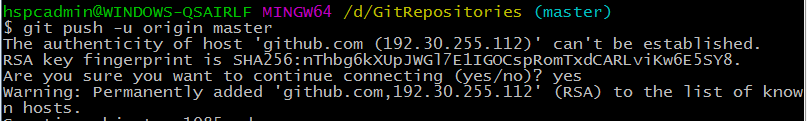
**（3）SSH警告**

当你第一次使用Git的clone或者push命令连接GitHub时，会得到一个警告：

The authenticity of host 'github.com (192.30.255.112)' can't be established.

RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

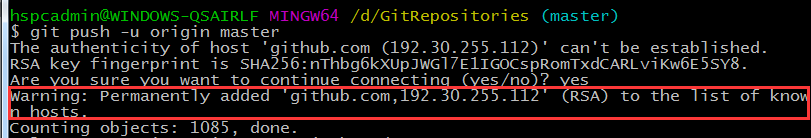
Are you sure you want to continue connecting (yes/no)? yes



这是因为Git使用SSH连接，而SSH连接在第一次验证GitHub服务器的Key时，需要你确认GitHub的Key的指纹信息是否真的来自GitHub的服务器，**输入yes回车即可**。

Git会输出一个警告，告诉你已经把GitHub的Key添加到本机的一个信任列表里了：

Warning: Permanently added 'github.com,192.30.255.112' (RSA) to the list of known hosts.



#### 5. 远程仓库