

开源项目阅读与管理

四、开源项目阅读与管理-远程仓库

开源项目阅读与管理-远程仓库

步骤 1：如何科学上网访问 Github

步骤 2：远程仓库

步骤 3：添加远程库

步骤 4：从远程库克隆

步骤 1：如何科学上网访问 GITHUB

<https://github.com> 为什么会访问失败或者速度很慢？

国内网络访问 Github 速度过慢的原因有许多，但其中最直接和原因是其 CND 域名遭到 DNS 污染，导致我们无法连接使用 GitHub 的加速服务，因此访问速度缓慢。

简单理解：CDN [Content Delivery Network]，即内容分发网络，依靠部署在各地的边缘服务器，平衡中心服务器的负荷，就近提供用户所需内容，提高响应速度和命中率。DNS 污染，是指一些刻意或无意制造出来的数据包，把域名指向不正确的 IP 地址，阻碍了网络访问。我们默认从目标网址的最近 CDN 节点获取内容，但当节点过远或 DNS 指向错误时，就会造成访问速度过慢或无法访问的问题。

解决方案：

1、购买 VPN，科学上网

2、修改 host 文件(用绕过 dns 解析，在本地直接绑定 host)

使用查询工具网站：<https://tool.chinaz.com/dns>

查询：assets-cdn.github.com 和 github.global.ssl.fastly.net

当前位置: 站长工具 > Dns查询

广告

超高分成播放器亏钱收量

Ping检测

国内测速

国际测速

网站速度对比

DNS查询

路由器追踪

NEW
DNS污染检测

A类型

assets-cdn.github.com

检测

查询记录

选填:如果要针对固定DNS服务器可填此项(限IP地址)

*(选填限IP地址)

DNS所在地	响应IP	TTL值
广东[电信]	-	-
贵州[电信]	185.199.111.153 [美国加利福尼亚旧金山]	452
	185.199.109.153 [美国加利福尼亚旧金山]	452
	185.199.108.153 [美国加利福尼亚旧金山]	452
	185.199.110.153 [美国加利福尼亚旧金山]	452
安徽[电信]	185.199.108.153 [美国加利福尼亚旧金山]	900
	185.199.109.153 [美国加利福尼亚旧金山]	900
	185.199.111.153 [美国加利福尼亚旧金山]	900
	185.199.110.153 [美国加利福尼亚旧金山]	900

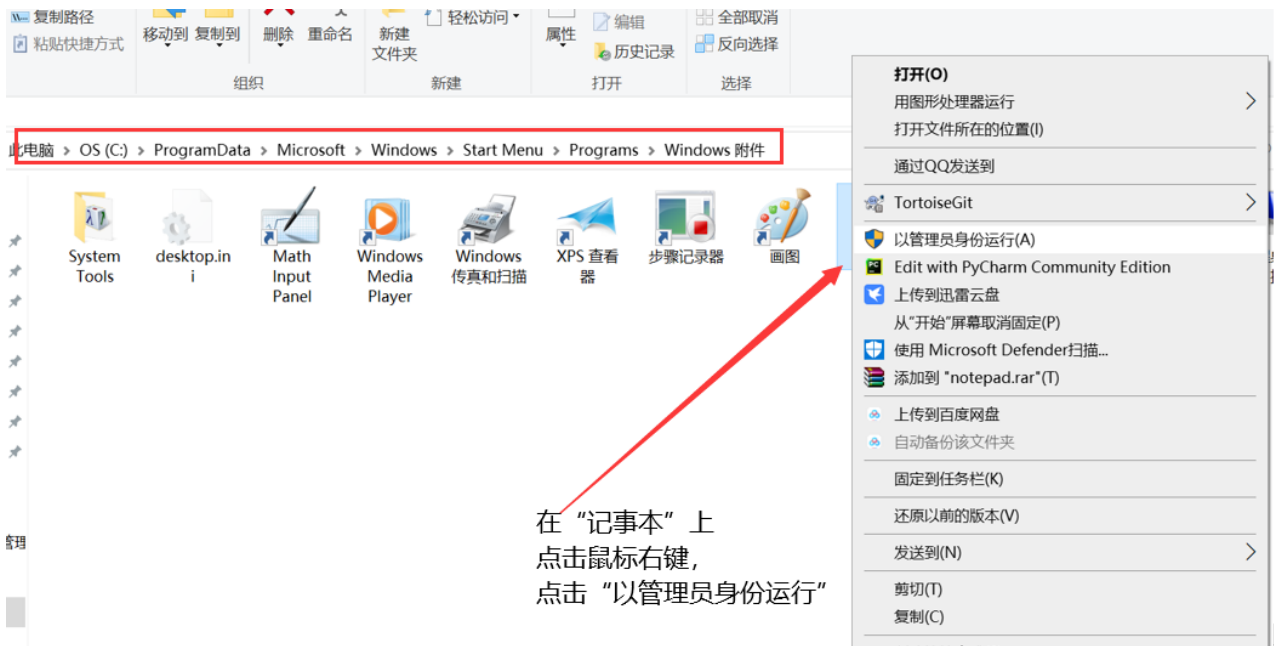
g检测 国内测速 国际测速 网站速度对比 **DNS查询** 路由器追踪 DNS污染检测

A类型 github.global.ssl.fastly.net 检测

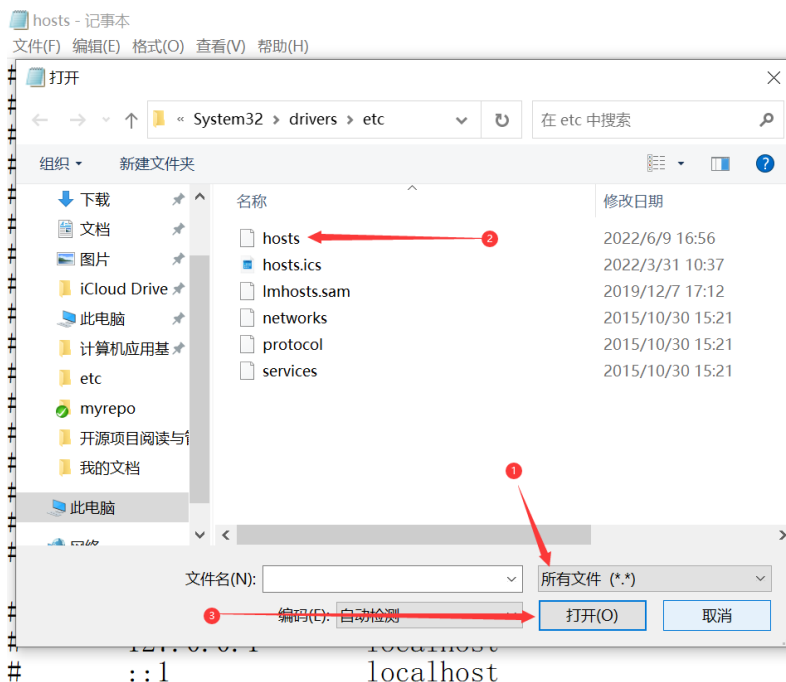
选填:如果要针对固定DNS服务器可填此项(限IP地址) *(选填限IP地址)

DNS所在地	响应IP	TTL值
广东[电信]	74.86.151.162 [美国德克萨斯达拉斯 SoftLayer]	600
贵州[电信]	103.252.115.169 [日本东京 Twitter]	60

找到“记事本”，以管理员身份运行（只有管理员才有权限改 hosts）：



在本地 host 文件中添加映射：路径是在 C:\Windows\System32\drivers\etc 选择菜单“文件”——“打开”——在“文件名”输入框输入 C:\Windows\System32\drivers\etc 点击打开，选择“所有文件”，选择“hosts”文件，点击“打开”：



P for Windows.

host names. Each
address should
responding host name.
ed by at least one

ected on individual
' symbol.

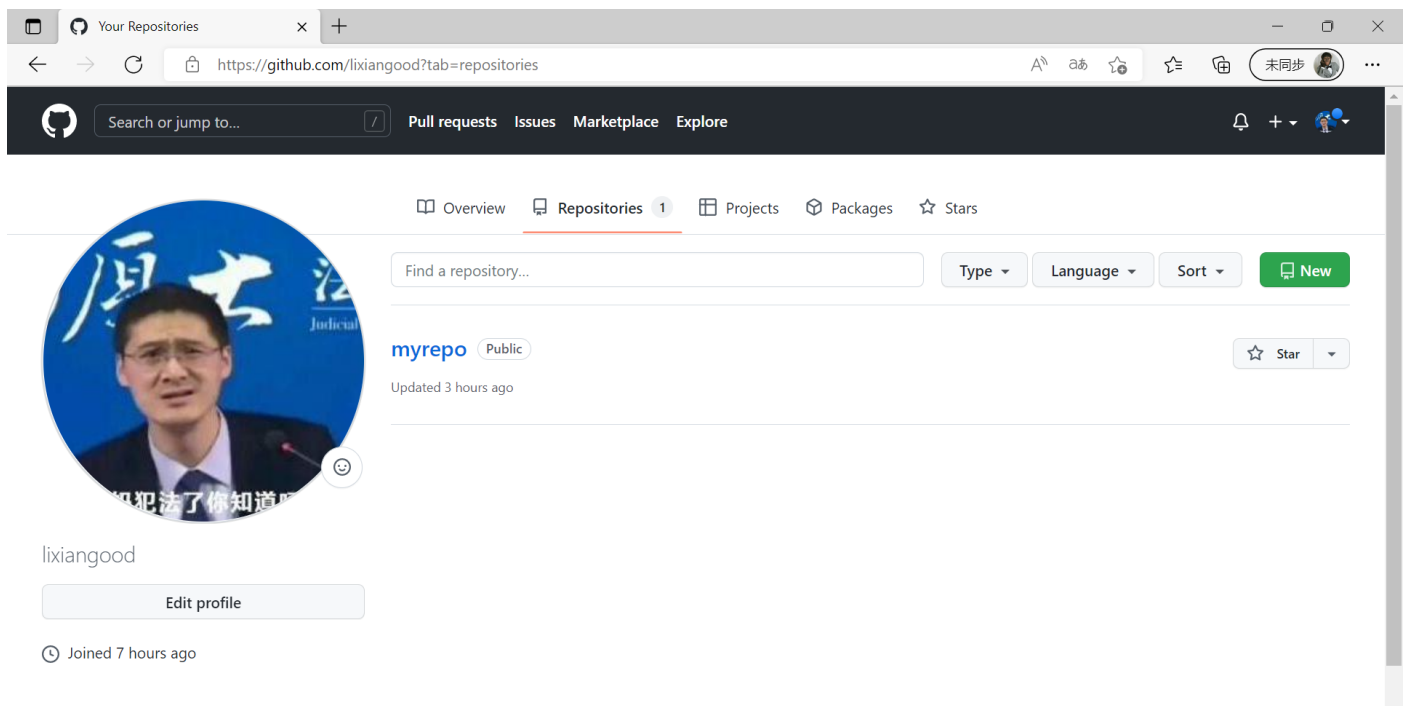
orce server
lient host

elf.

修改完成，记得保存文件：

```
hosts - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by a single
# space.
#
# Additionally, comments (such as these) may be inserted on empty
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com   # source host
#       38.25.63.10       x.acme.com       # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1               localhost
185.199.111.153 assets-cdn.github.com
185.199.111.153 github.global.ssl.fastly.net
```

让 hosts 生效，Windows：开始 -> 运行 -> 输入 cmd -> 在 CMD 窗口输入：ipconfig /flushdns
当然，断网重连或者重启电脑最为简单粗暴~
到了这一步，就可以顺利的访问 Github 啦~!!!



步骤 2：远程仓库

我们来回顾一下我们了解的，目前市面上优秀的版本管理器有两个：

①集中式的 SVN

②分布式的 Git

版本管理器既然是帮我们做备份的，那么问题来了，备份的文件放在哪？SVN 既然是集中式的，那肯定就是中央集权，有一个统一的文件服务器存放这些文件，每个人单独与之做沟通，但集中式的注定了当作为核心的 SVN 服务器挂掉之后，所有人都没法干活。

而 Git，它高明之处在于，人人平等，每个人都有一个完全属于自己的独立仓库，尽管它也有一个中间的交互服务器，但那仅仅只是作为一个中间媒介，当中间节点挂了，你本机有一整个的库，不会对你有过大的影响。

如果只是在一个仓库里管理文件历史，Git 和 SVN 区别不大，本章开始介绍 Git 重要功能之一：远程仓库。

1、远程仓库介绍

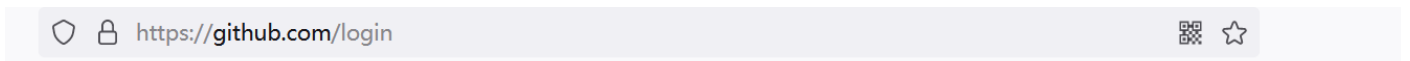
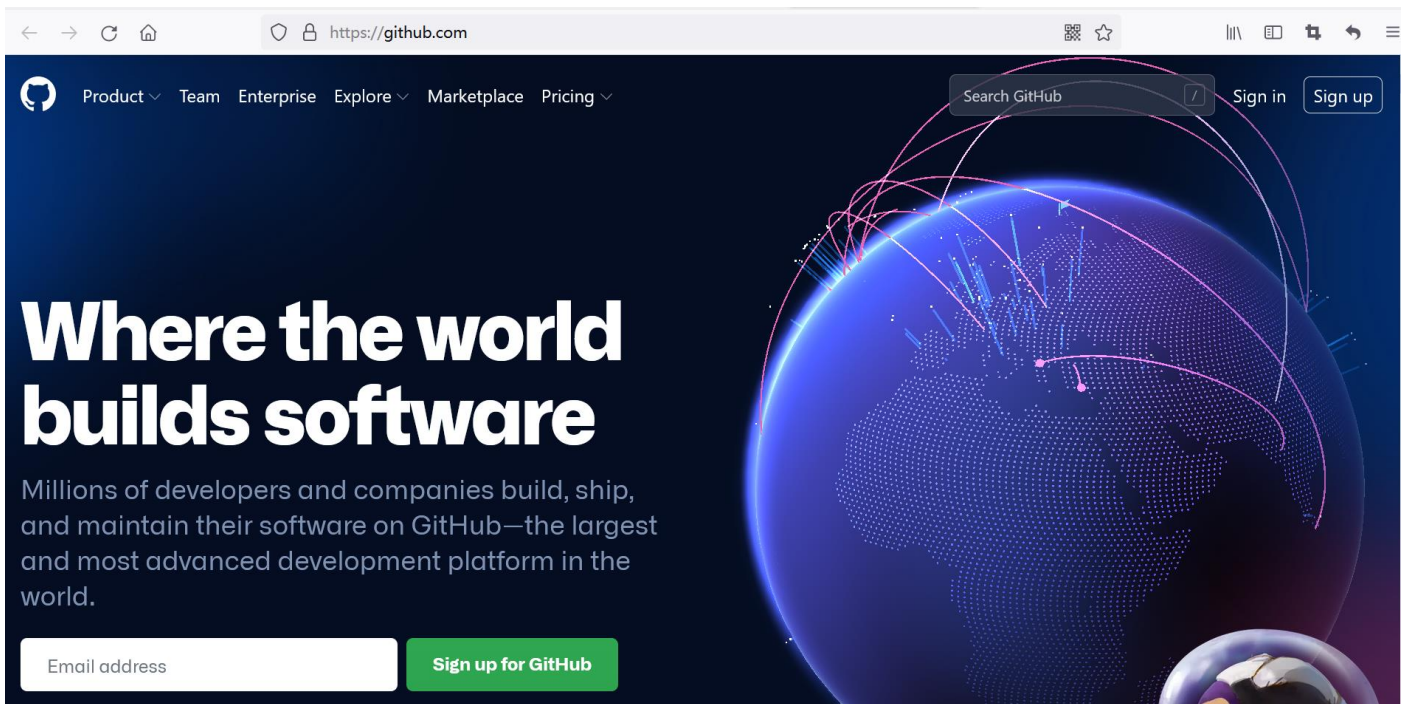
Git 是分布式版本控制系统，同一个 Git 仓库，可以分布到不同的机器上。怎么分布呢？最早，肯定只有一台机器有一个原始版本库，此后，别的机器可以“克隆”这个原始版本库，而且每台机器的版本库其实都是一样的，并没有主次之分。

你肯定会想，至少需要两台机器才能玩远程库不是？但是我只有一台电脑，怎么玩？其实一台电脑上也是可以克隆多个版本库的，只要不在同一个目录下。不过，现实生活中是不会有人这么傻的在一台电脑上搞几个远程库玩，因为一台电脑上搞几个远程库完全没有意义，因为硬盘挂了会导致所有库都挂掉。

实际情况往往是这样，找一台电脑充当服务器的角色，每天 24 小时开机，其他每个人都从这个“服务器”仓库克隆一份到自己的电脑上，并且各自把各自的提交推送到服务器仓库里，也从服务器仓库中拉取别人的提交。

完全可以自己搭建一台运行 Git 的服务器，不过现阶段，为了学 Git 先搭个服务器绝对是小题大作。好在这个世界上有个叫 **GitHub** 的神奇网站，从名字就可以看出，这个网站就是提供 Git 仓库托管服务的，所以，只要注册一个 GitHub 账号，就可以免费获得 Git 远程仓库。

请访问：<https://github.com/> 注册 GitHub 账号。



输入邮箱、密码、：

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ lixiangood@qq.com

Create a password
✓ ●●●●●●●●

Enter a username
✓ lixiangood


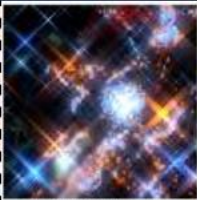




Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no

→ y

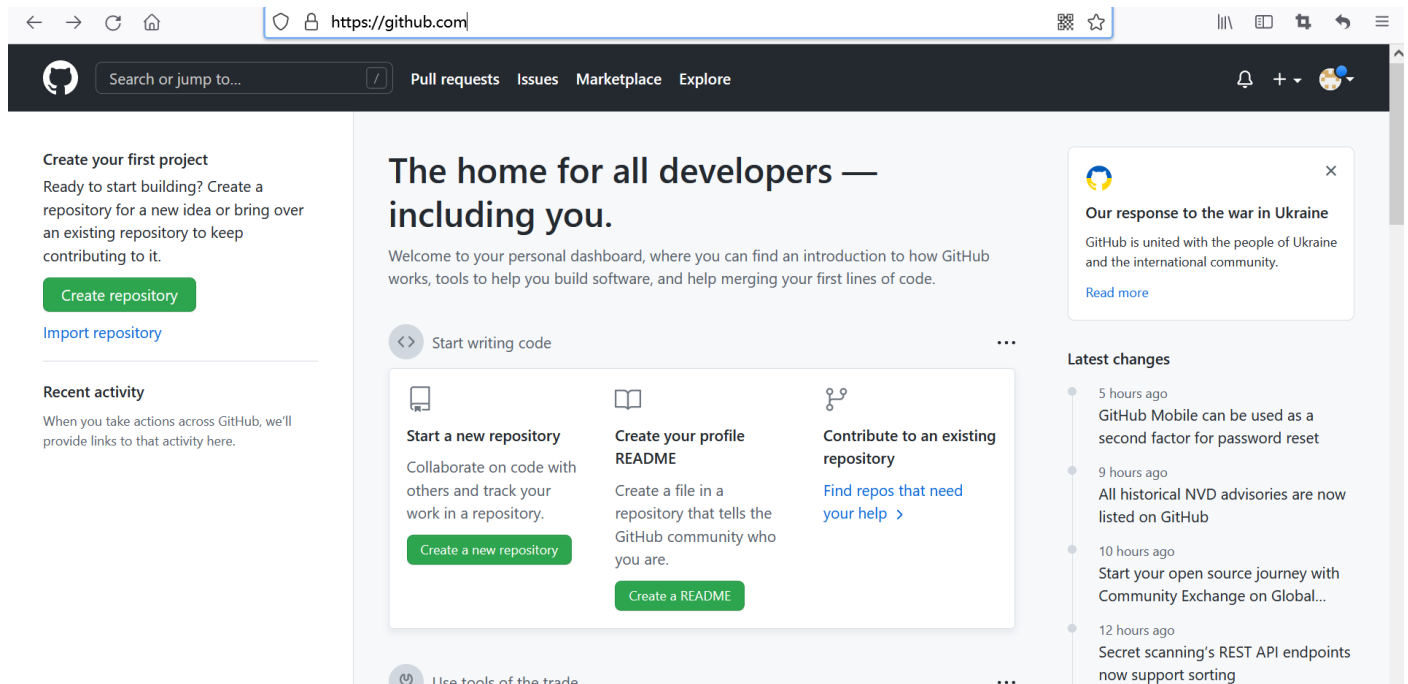
验证你是不是真人：

Verify your account

选出旋涡星系

输入邮箱注册码，如下图显示注册完成：



2、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，所以进行设置：

第 1 步：创建 SSH Key。在本地仓库 myrepo 目录下，打开 Git Bash，创建 SSH Key：

```
$ ssh-keygen -t rsa -C "lixiangood@qq.com"
```

把邮箱换成你的邮箱名，接下来输入密钥文件名：id_rsa，然后一直按回车，不懂英文的看下面的翻译。

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ ssh-keygen -t rsa -C "lixiangood@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/李响/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:Dgme7F4qc4V2ZsuFP1wIjYX9U7tSHBYMcK/UUYiL7uw lixiangood@qq.com
The key's randomart image is:
+---[RSA 3072]---+
|           +0.0+0+. |
|          0 0..0*.  |
|         . 0 0.=00   |
|        0 0....+.+   |
|       +.0+S..0 .    |
|      .0 *0+ 0 .     |
|     ..*.B.. .       |
|    0..00 *          |
|    +0 .E.           |
+---[SHA256]-----+
```

Generating public/private rsa key pair.

生成公共/私有 rsa 密钥对。

Enter file in which to save the key

输入保存密钥的文件

Enter passphrase (empty for no passphrase)

输入密码短语（无密码短语为空）

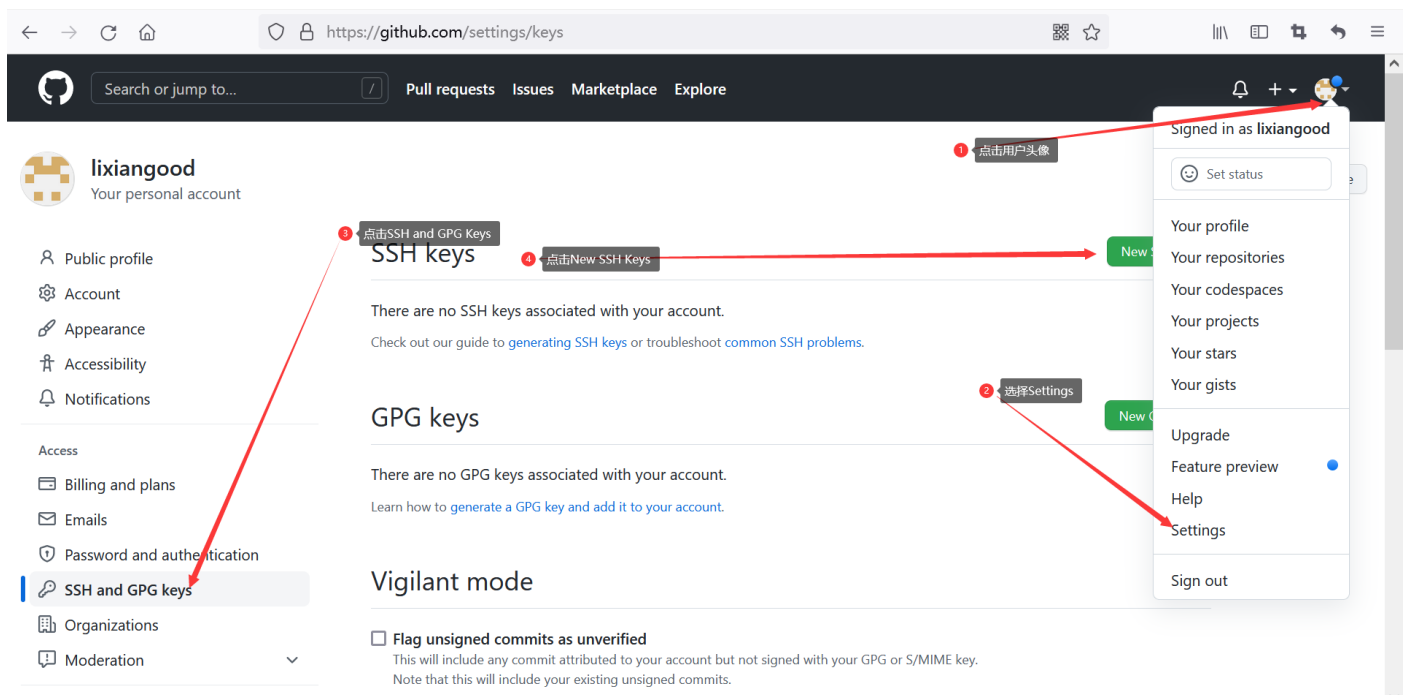
在 myrepo 目录下找到公共/私有 rsa 密钥对：

id_rsa
id_rsa.pub

id_rsa 和 id_rsa.pub，这两个就是 SSH Key 的密钥对，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。使用记事本打开：id_rsa.pub

```
id_rsa.pub - 记事本
文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDJH3Ynm
+hPd2LJQ1KAJzQNYSSedaE48jSuHjT1BuMcRZz0jorNvm4Mdz/BcW2pjgqxr0x/zrkD5J2Ix8
ATuuSR8/bXBA5x21Gv3XAKOb0VHvNyoYeRrOHTcBJmUYhEQZthr6+kOCnRzFegCFY0LYB51Cn
r09aVvcs6LcBqmMH696rbPIikU90b/w5WYT6BDHcE9XvinP4xVb8Ncgin36ez9kpfIOGphN9
AFCbxHPsRUJ6X84Y0IHP2d0sY8JqWyVuWkWjnh8PHadp0Af7v7RSa0xucDItd8gtvw0nB15wL
EZzsh9faL7EE0tSfXPby6VM0K0oCzFwAVx9Y7hrn8NFtCHU1FtJR6NnFwhQs/y3ecAYHLaUif
dUwMiBYjxQSt72mGUqFv0990HEhpMBT839e/HRrYysdRFecJ08J+rfgI6/HNxOc
+2pldZAuwkso8bMXBS4m1WRwIW3GGDung9gHZOE20GHgJ8wnbCW
+SoLSQPHkyfK5WmZtZXzgdC0pX0= lixiangood@qq.com
```

进入 GitHub 网站，按照如下的步骤，填入 ssh keys:



复制上面记事本打开的 id_rsa.pub 中的文本，粘贴填入 Key，Title 随便写：

https://github.com/settings/ssh/new

SSH keys / Add new

Title

my ssh key

Key

AAAAB3NzaC1yc2EAAAADAQABAAQgQCXQop5cmRTI3iZ05ttcfZjhcWDRUCVcOUjLTav2OyPdsmAf9t1PAS5DDYnBaR
tqw9cab3MrSov595CitJ8vg8ZYa7qS5+dTHaKV49wF3Nd7hNOgEWI4Bp1tHMm/E77TJ0Gt6M4Cvmrr5A3Qusx2s
/yOxTuTtfqu37+CQ0GivMHKD
//W5ZZzOLMgNfV1mghMFb5LvJ92mUlfG2wBaL4zVCqNxx7oss59RnV4drSASQJnNDBnTTsRpzCMMtaZUcvsqdoc7PN
HjLKYSAlBfMmy
/wxnpJa8UxAmO4fzPKuQzDKPYfXv2JtQQ8lqGo7sTsHTXqTWXWNJHcPug+TTQ6rzlKv4tBuHOzq29G+28+uTIU140rYw
jCQ19JknUaHsIVZDwLzIF3vXeqo+i+gyS4v8Ows8s34hZBKa5xA4DC2aZdX441OA9WtCnGRPww5uBvRkXlwoVgWZ2hkn
TbDASRdCSGc0P8QiyLy0xqCF0l/ktOBfsL9xOJcN+xgSKSrAsPrk= lixiangood@qq.com


Add SSH key

显示填入的 SSH Key:

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



my ssh key

SHA256:Dgme7F4qc4V2ZsuFP1wIJYX9U7tSHBYMcK/UUYiL7uw

Added on 9 Jun 2022

Never used — Read/write

SSH

Delete

GitHub 需要 SSH Key 来识别出你推送的提交确实是你推送的，而不是别人冒充的，而 Git 支持 SSH 协议和 Https 协议，所以，GitHub 只要知道了你的公钥，就可以确认只有你自己才能推送，GitHub 允许你添加多个 Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的 Key 都添加到 GitHub，就可以在每台电脑上往 GitHub 推送了。

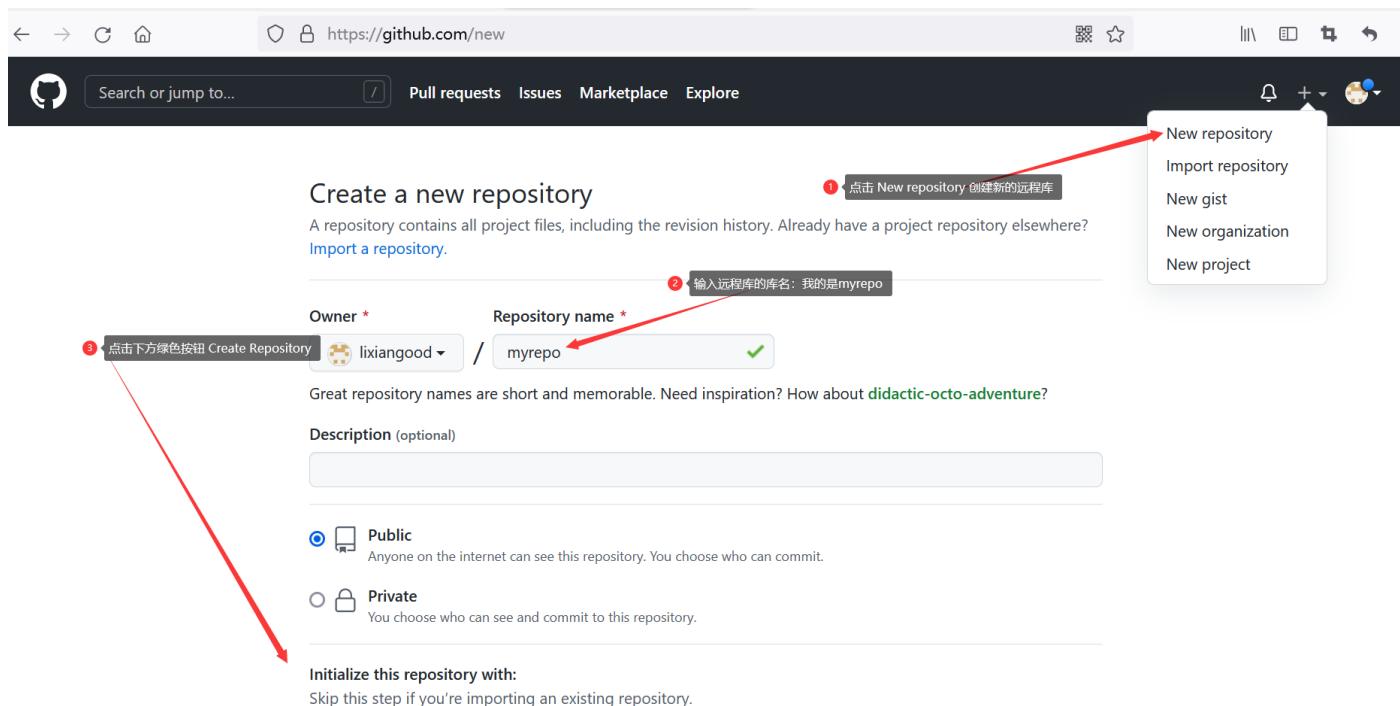
在 GitHub 上免费托管的 Git 仓库，任何人都可以看到喔（但只有你自己才能改）。所以，不要把敏感信息放进去。

如果你不想让别人看到 Git 库，有两个办法，让 GitHub 把公开的 public 仓库变成私有的 private，这样别人就看不见了（不可读更不可写）。另一个办法是自己动手，搭一个 Git 服务器，因为是你自己的 Git 服务器，所以别人也是看不见的。这个方法我们后面会讲到的，相当简单，公司内部开发必备。确保你拥有一个 GitHub 账号后，我们就即将开始远程仓库的学习。

步骤 3: 添加远程库

1、现在的情景是，你已经在本地创建了一个 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，真是一举多得。

首先，登陆 GitHub，然后，在右上角找到“New repository”链接，按照下面的步骤，创建一个新的远程仓库：**myrepo**



目前，在 GitHub 上的这个 myrepo 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。

现在，我们根据 GitHub 的提示，在本地的 myrepo 仓库下运行命令：

```
$ git remote add origin https://github.com/lixiangood/myrepo.git
```

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git remote add origin https://github.com/lixiangood/myrepo.git
```

请千万注意，把上面的 lixiangood 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。下一步，就可以把本地库的所有内容推送到远程库上：

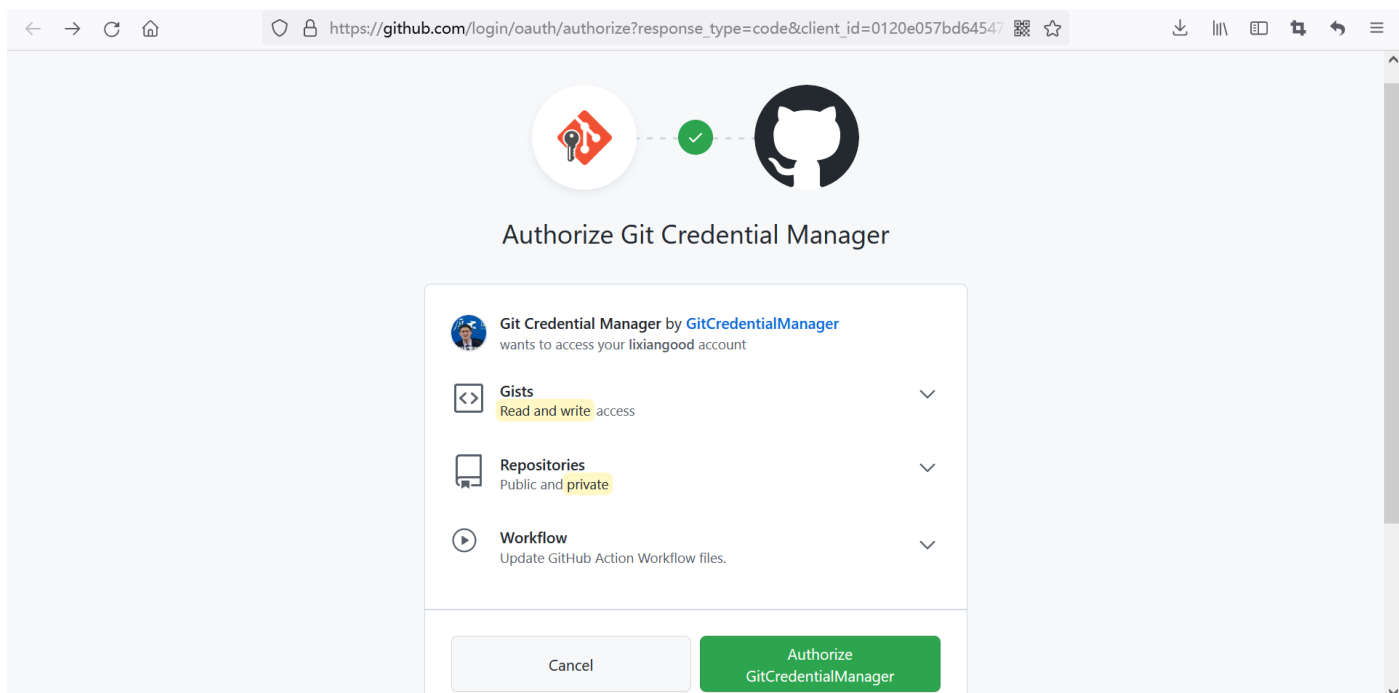
```
$ git push -u origin master
```

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。

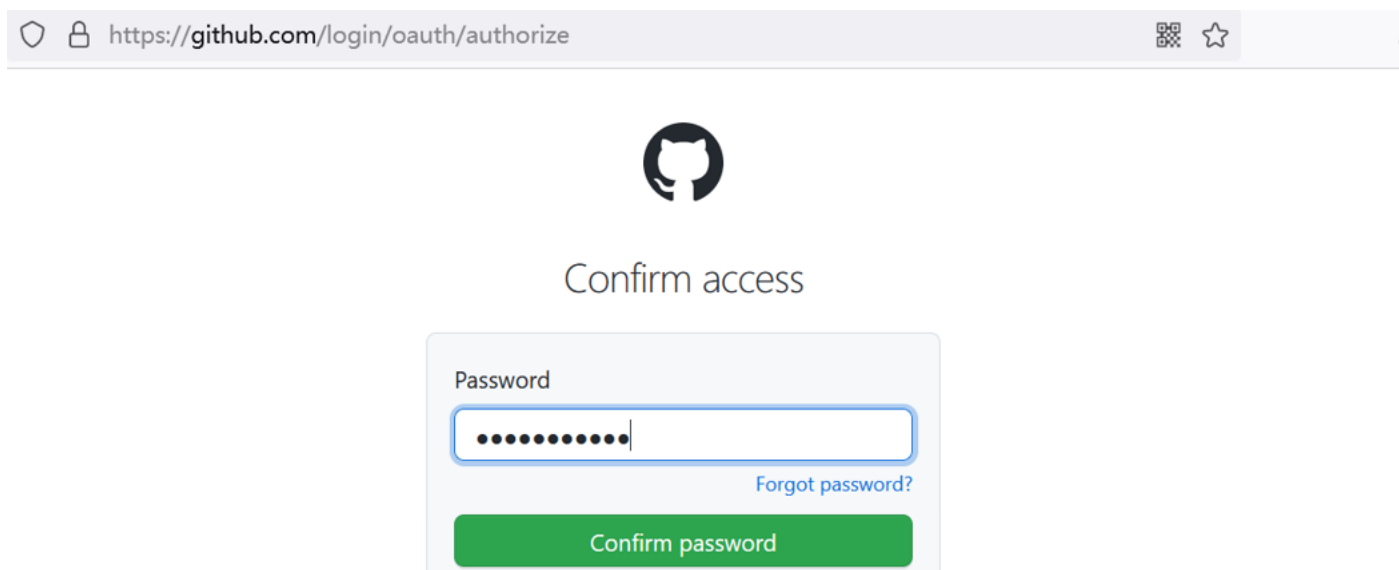
```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git push -u origin master
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 4 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (27/27), 36.53 KiB | 2.28 MiB/s, done.
Total 27 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
To https://github.com/lixiangood/myrepo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

敲完命令回车会卡起来，
弹出一个网页选择授权方式
选第一个，网页授权。
选择授权，输入密码，回车

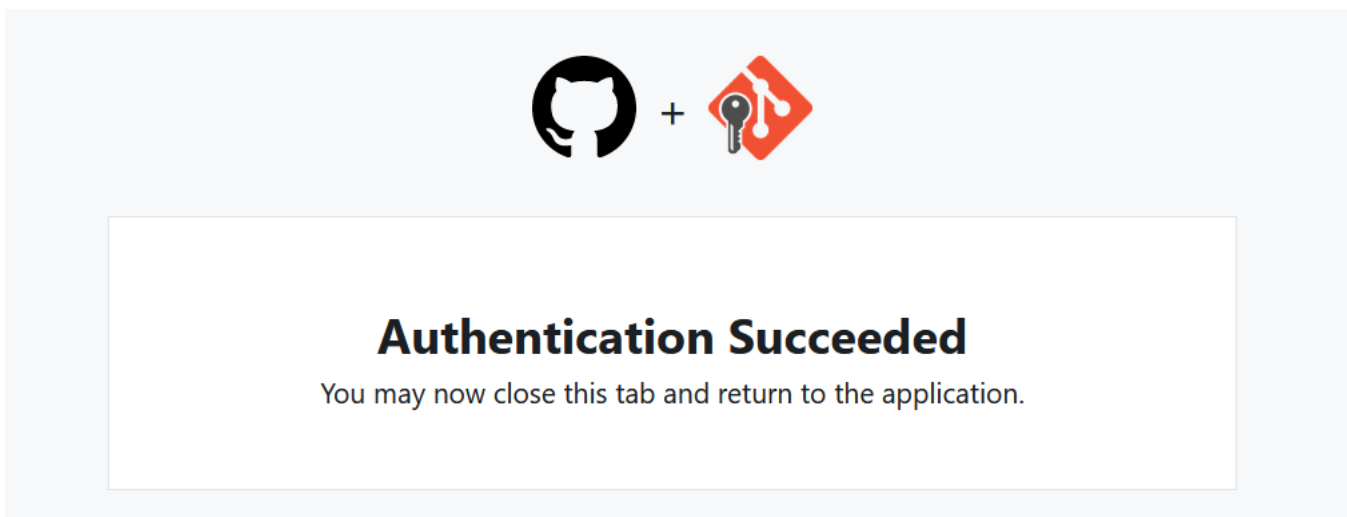
授权登录：



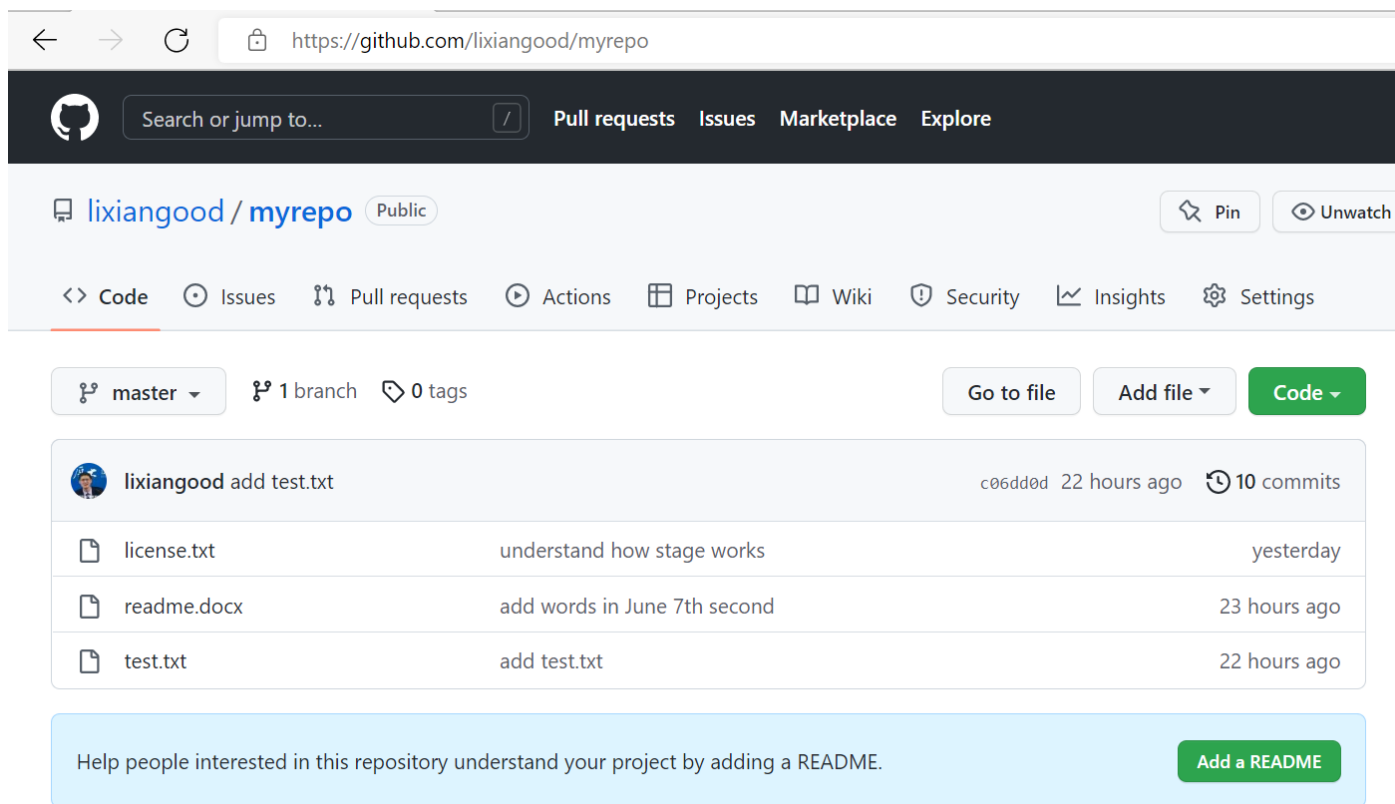
输入密码：



提示授权登录成功：



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：



从现在起，只要本地作了提交，就可以通过命令：

\$ git push origin master

把本地 **master** 分支的最新修改推送至 **GitHub**，现在，你就拥有了真正的分布式版本库！

现在自己实验一下，在 **readme.docx** 里增加一些文字，再次 **add** 和 **commit** 到本地库，再添加到远程库。

■ 删除远程库（这只是告诉你可以删除，你别手贱把可以用的远程库删了就不能提交了啦）

如果添加的时候地址写错了，或者就是想删除远程库，可以用 `git remote rm <name>` 命令删除远程库。使用前，建议先用 `git remote -v` 查看远程库信息：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git remote -v
origin https://github.com/lixiangood/myrepo.git (fetch)
origin https://github.com/lixiangood/myrepo.git (push)
```

删除命令：\$ `git remote rm origin`

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git remote rm origin

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git remote -v
```

此处的“删除”其实是解除了本地和远程的绑定关系，并不是物理上删除了远程库。远程库本身并没有任何改动。要真正删除远程库，需要登录到 **GitHub**，在后台页面找到删除按钮再删除。

当然如果删了还是可以按上面的步骤再次添加~！

■ 总结

要关联一个远程库，使用命令

```
$ git remote add origin https://github.com/lixiangood/myrepo.git
```

关联一个远程库时必须给远程库指定一个名字，`origin` 是默认习惯命名；

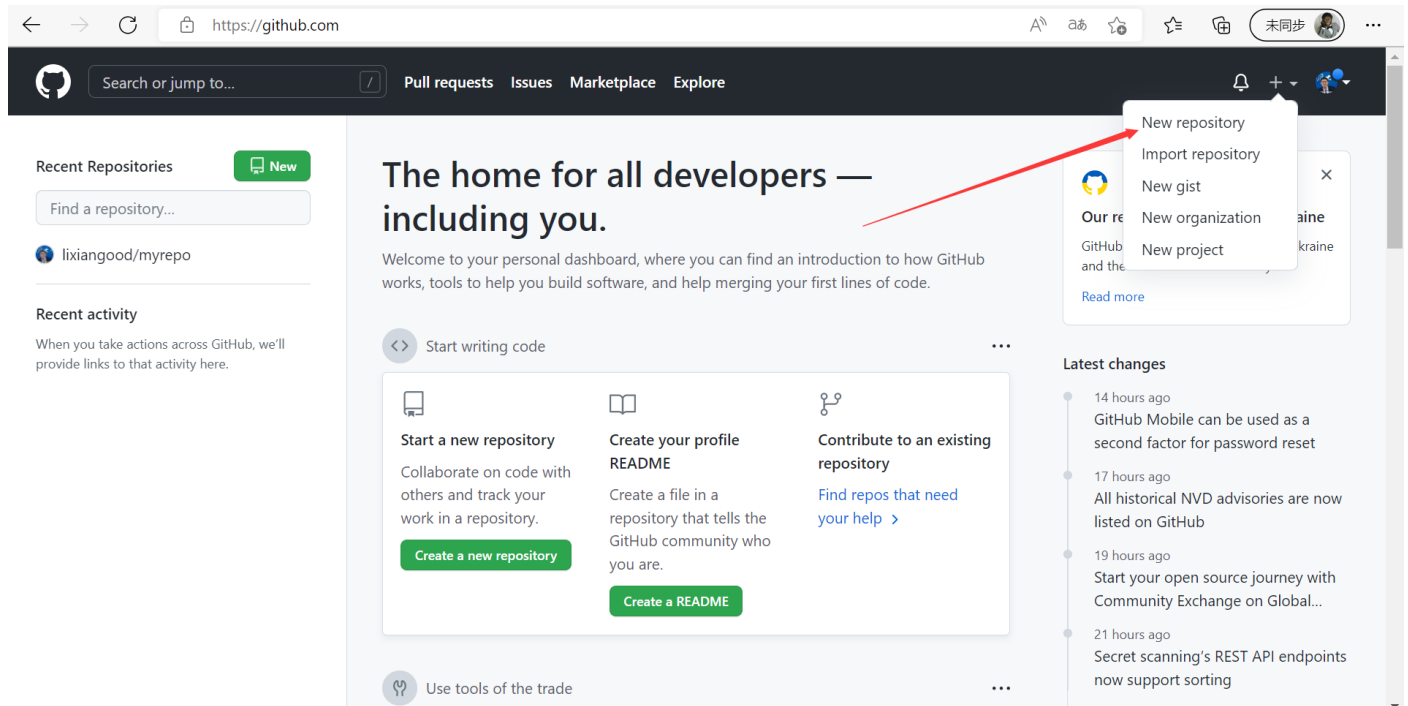
关联后，使用命令 `git push -u origin master` 第一次推送 `master` 分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令 `git push origin master` 推送最新修改；

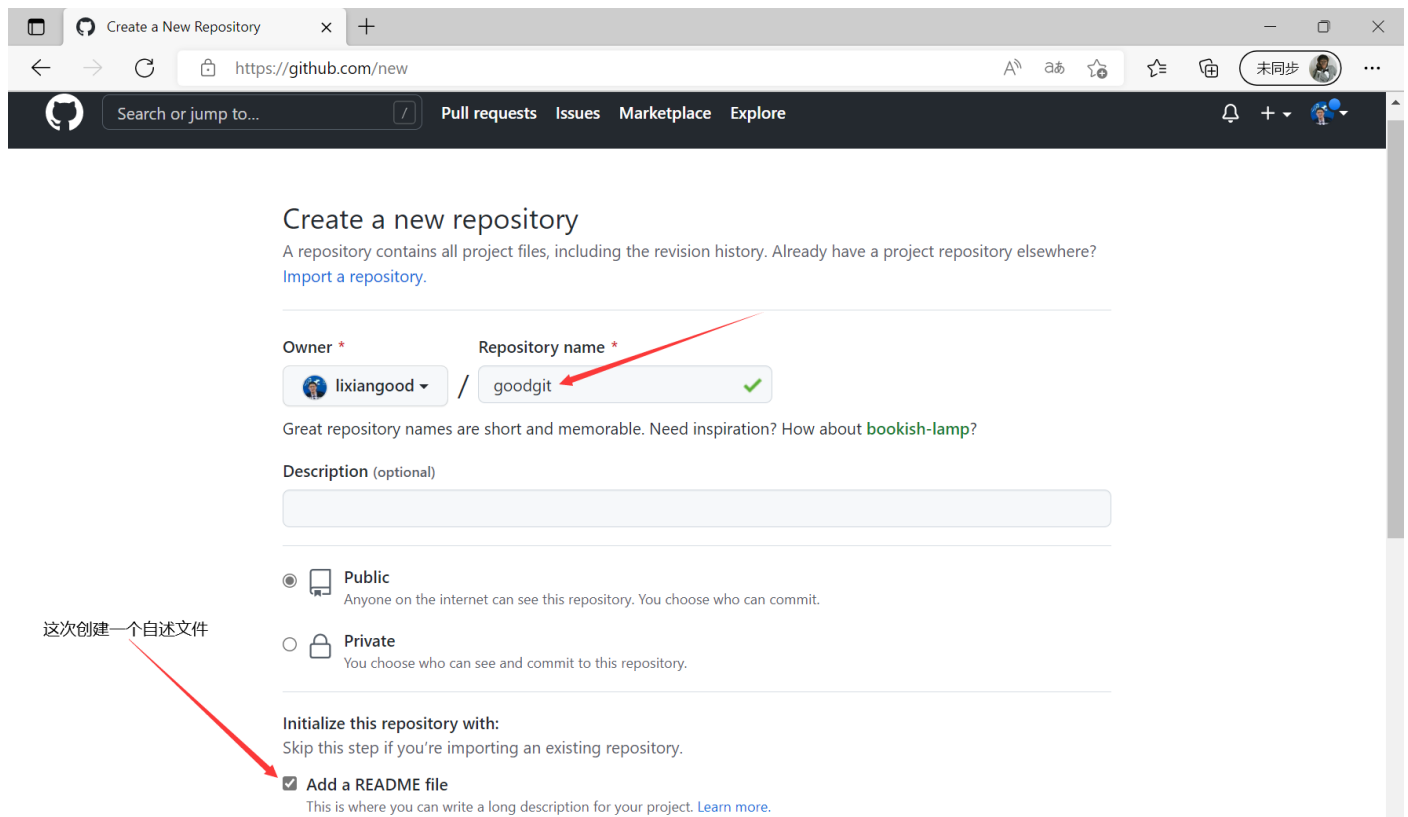
分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而 **SVN** 在没有联网的时候是拒绝干活的！当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了！

步骤 4：从远程库克隆

上次我们讲了先有本地库，后有远程库的时候，如何关联远程库。现在，假设我们从零开发，那么最好的方式是先创建远程库，然后，从远程库克隆。首先，登陆 **GitHub**，

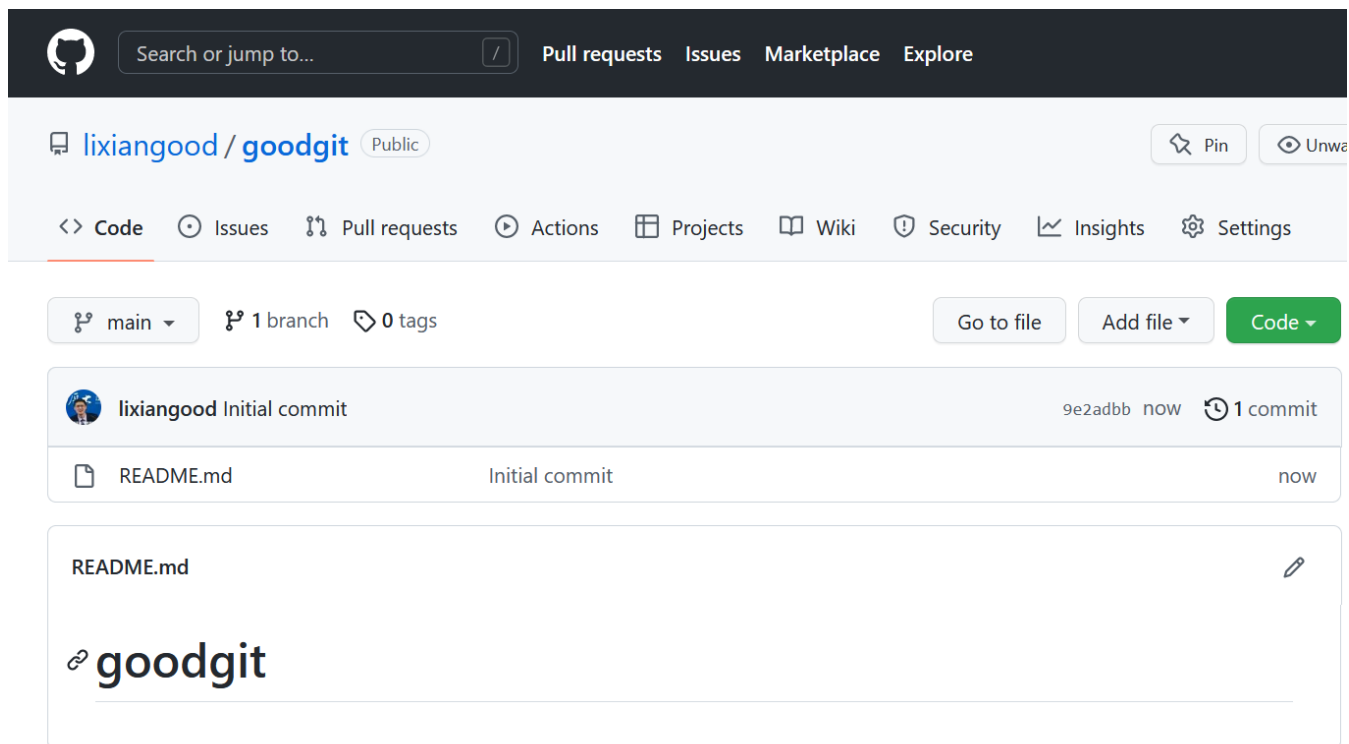


创建一个新的仓库，名字叫 **goodgit**：



这次创建一个自述文件

现在，远程库已经准备好了



下一步是用命令 `git clone` 克隆一个本地库：

在 Git Bash 中先把当前目录切换到 D 盘

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ cd D:
```

输入命令：\$ `git clone https://github.com/lixiangood/goodgit.git`

```
李响1@LiXiang MINGW64 /d
$ git clone https://github.com/lixiangood/goodgit.git
Cloning into 'goodgit'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

注意把 Git 库的地址换成你自己的，然后进入 `goodgit` 目录看看，已经有 `README.md` 文件了：

电脑 > 新加卷 (D:) > `goodgit`



如果有多个人协作开发，那么每个人各自从远程克隆一份就可以了。

GitHub 给出的地址不止一个，我们可以用 `https://github.com/michaelliao/gitskills.git` 这样的地址。Git 支持多种协议，默认的 `git://` 使用 `ssh`，但也可以使用 `https` 等其他协议。

使用 `https` 除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放 `http` 端口的公司内部就无法使用 `ssh` 协议而只能用 `https`。

总结

要克隆一个仓库，首先必须知道仓库的地址，然后使用 `git clone` 命令克隆。Git 支持多种协议，包括 `https`，但 `ssh` 协议速度最快。