

Git and GitHub

Notes on Git and GitHub related operations

目录

Git and GitHub

配置SSH keys

上传本地项目到GitHub仓库

从GitHub上克隆项目到本地

配置SSH keys

(1) 打开Git Bash，检查是否已经生成密钥：

```
cd ~/.ssh //进入文件夹
ls        //列出文件
```

如果有“id_rsa.pub”和“id_rsa”两个文件，则说明已经生成密钥。

打开“此电脑”，在C:\Users\电脑用户名\.ssh里可以找到这两个文件。

(2) 如果没有生成，则运行以下指令生成：

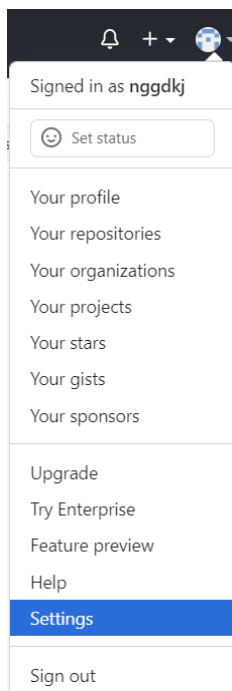
```
ssh-keygen -t rsa -C "xxxxxx@xxx.com"
```

运行后会有“路径确认、是否使用密码、确认密码”等操作，直接默认(按3次回车)。

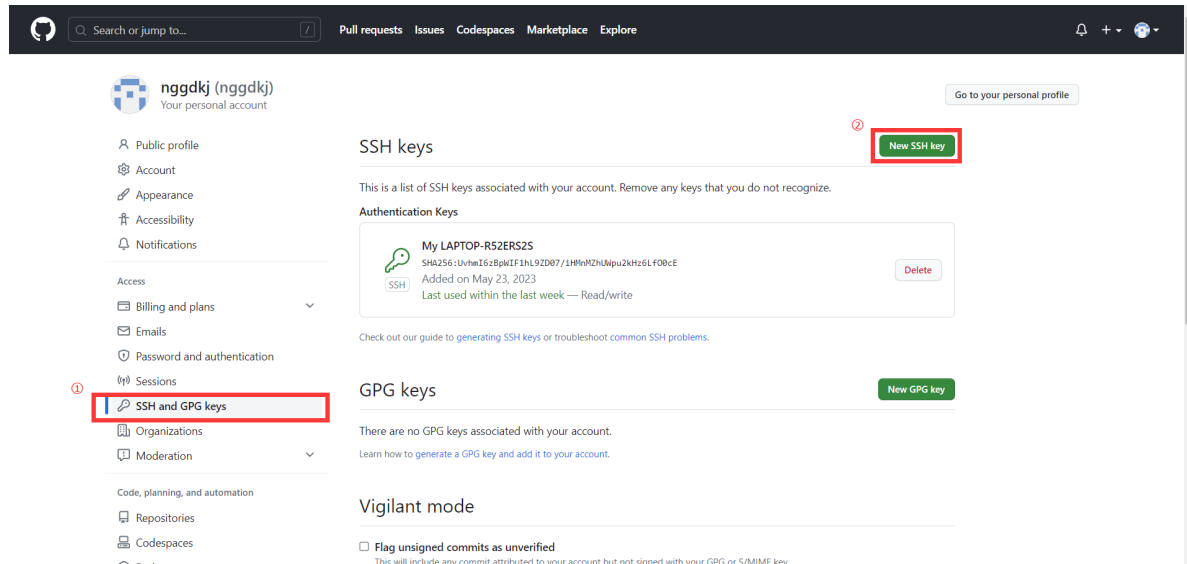
生成成功后，进入C:\Users\电脑用户名\.ssh，用记事本打开“id_rsa.pub”文件，即为ssh key公钥。

(3) 将SSH key与GitHub账号关联

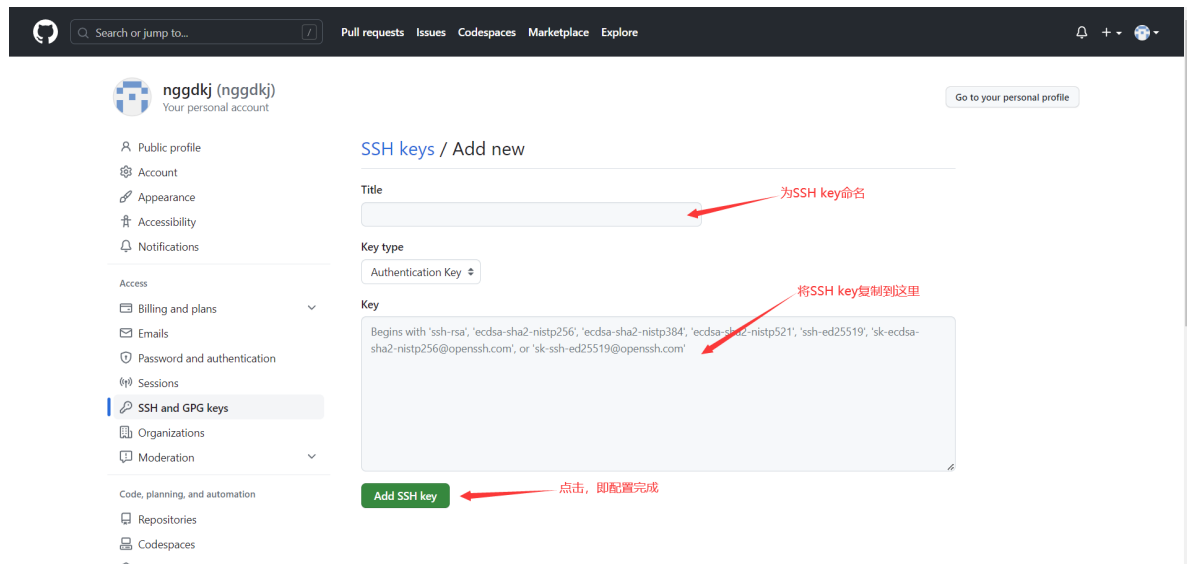
① 登录GitHub账号，点击右上角头像，点击Setting。



② 点击 SSH and GPG keys，点击 New SSH key，新建一个SSH key。



③ 为SSH key命名(建议以主机名字命名，便于区分)；用记事本打开“id_rsa.pub”文件，将**所有内容**复制粘贴到此处，点击 Add SSH key，即配置完成。



至此，本机与GitHub账号使用SSH key关联完成。

1、什么是SSH key?

—— 非对称加密。生成一对密钥，包括一个公钥（即为id_rsa.pub）和一个私钥（即为id_rsa）。公钥放在远程主机（即GitHub上），私钥放在本地主机。通过这对密钥，可以实现本地主机与远程主机的加密传输。

2、为什么要配置SSH key?

—— 配置后GitHub就信任这台主机，之后将本地代码 push 到你的仓库时就不需要输入账号密码了。

3、其他

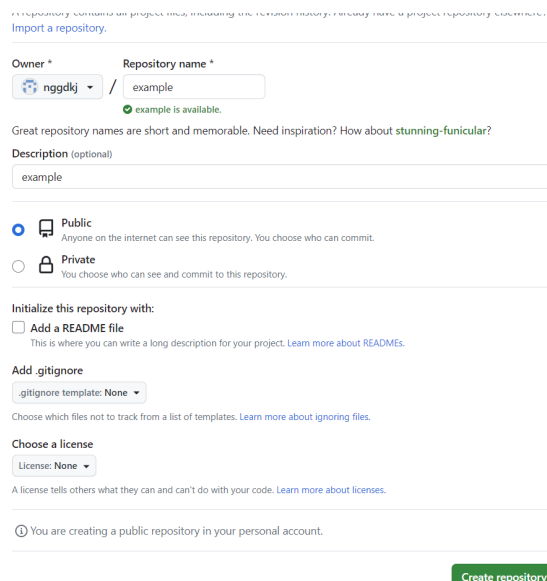
—— 密钥丢失（公钥和私钥丢失任意一个就都不能使用了）：按照上面步骤重新生成一次即可。

—— SSH key 的配置是针对**每台主机**的。

上传本地项目到GitHub仓库

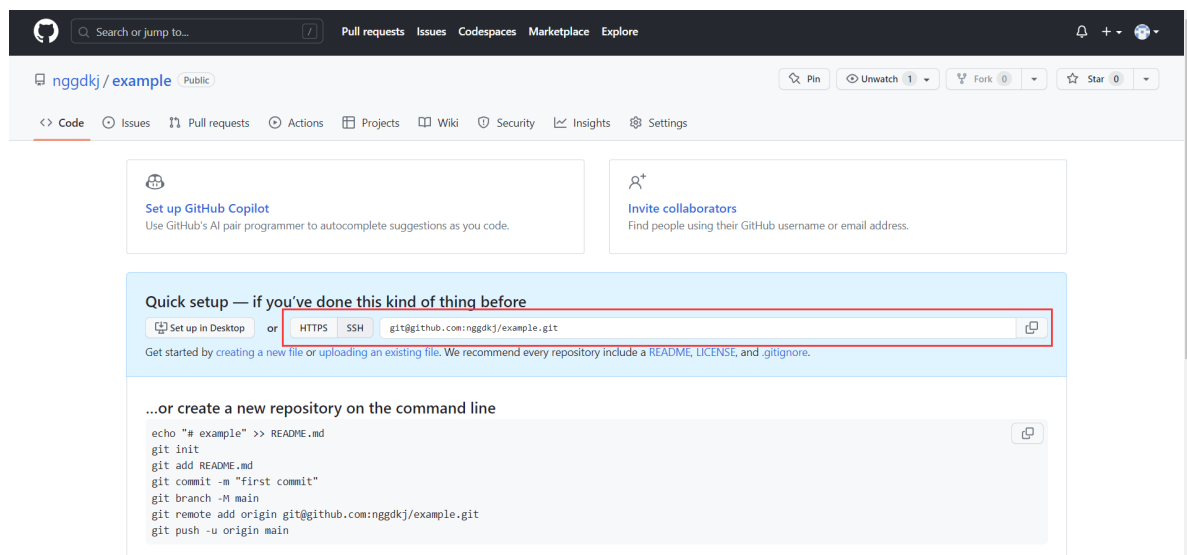
(1) 进入个人的 Repositories 页，点击 New：

设置仓库名称、描述、权限等，点击 Create repository 创建仓库。



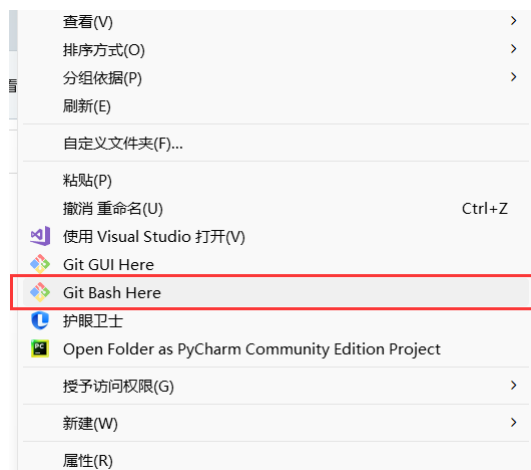
The screenshot shows the 'Create new repository' form on GitHub. It includes fields for 'Owner' (set to 'nggdj'), 'Repository name' (set to 'example'), and 'Description' (optional, set to 'example'). There are radio buttons for 'Public' (selected) and 'Private'. Below these are options to 'Initialize this repository with:' including 'Add a README file' and 'Add .gitignore' (set to 'None'). A 'Choose a license' dropdown is also present. At the bottom right is a green 'Create repository' button.

创建完成后，可以看到仓库的 HTTPS 和 SSH 地址（后面会用到），如下图。



(2) 建立本地仓库：

进入想要上传的项目的文件夹，右击鼠标，点击 Git Bash Here。

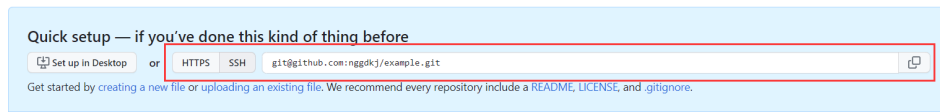


执行以下指令：

```
git init    //初始化, 会生成一个隐藏文件夹.git
git add .   //将所有文件添加到仓库
git commit -m "注释" //把所有文件提交到仓库, 双引号内是提交注释
```

至此, 本地仓库建立完成。

(3) 关联GitHub仓库



复制刚刚创建仓库时得到的仓库地址, 执行以下指令:

```
git remote add origin xxxxxxxx //此处xxxxxxx即为仓库地址
```

(4) 上传本地代码

执行以下指令:

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

如果报错的话执行以下指令:

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

如果还是报错的话执行以下指令:

```
git push -u origin main
```

如果还是报错的话执行以下指令:

```
git push -u origin main-f
```

执行成功后, 中途可能会需要手打一个 yes。

至此, 本地项目成功上传到 GitHub 仓库。

1、如果项目内容有更新, 如何上传更新后的项目?

—— 执行以下命令:

```
git add xxx // xxx为更新的文件名, 也可以直接git add .
git commit -m "注释" //可以在注释中描述更新的内容
git push -u origin master
```

—— 如果不是在原来的项目文件夹里:

```
git init    //初始化, 会生成一个隐藏文件夹.git
git add xxx // xxx为更新的文件名, 也可以直接git add .
git commit -m "注释" //可以在注释中描述更新的内容
git remote rm origin
git remote add origin xxxxxxxx //xxxxxxx即为仓库地址
git push -u origin master
```

—— 如果出现 error: failed to push some refs to "XXXXXXX" :

```
git pull --rebase origin master //把远程库中的跟新合并到本地库中
```

从GitHub上克隆项目到本地

(1) 新建文件夹用于存放克隆的项目：

进入文件夹，右击鼠标，点击 Git Bash Here。

(2) 建立本地仓库并克隆：

执行以下指令：

```
git init //初始化，会生成一个隐藏文件夹.git  
git clone xxxxxxxx //xxxxxxx即为仓库地址
```

至此，则实现了项目克隆。

克隆后对项目有修改，如何更新到远程仓库？

```
git status //确定文件是否已在Git仓库中  
git add xxx // xxx为更新的文件名，也可以直接git add .  
git status //确定文件是否已在Git仓库中  
git commit -m "注释" //可以在注释中描述更新的内容  
git push
```

多个设备同步：

```
git pull //在设备1更新并push到github后，想在设备2同步github内容，则使用git pull  
(git clone 相当于"第一次 git pull"，只需要 clone 一次，后续更新同步用 git pull)
```