1、首先需要检查电脑是否已经有 SSH key，运行 git Bash 客户端，输入如下代码：

cd ~/.ssh

ls

这两个命令就是检查是否已经存在 id\_rsa.pub 或 id\_dsa.pub 文件，如果文件已经存在，那么可以跳过步骤2，直接进入步骤3。该文件是存储在：~/.ssh/下



2、创建一个 SSH key

ssh-keygen -t rsa -b 4096 -C email@example.com

代码参数含义：

-o 会以比默认格式更能抗暴力破解的格式保存私钥。

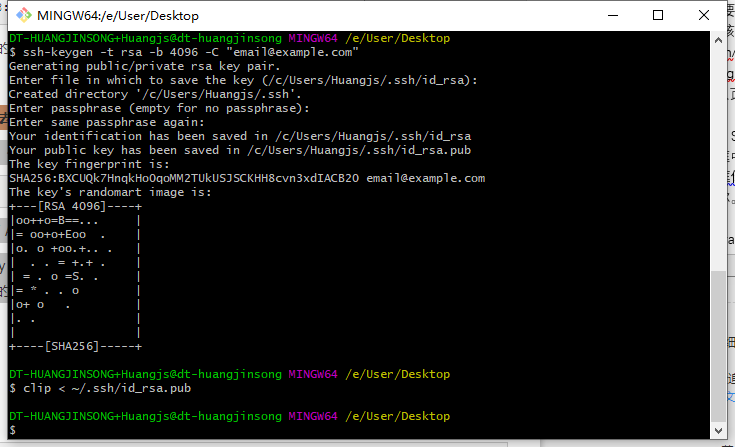
-t 指定密钥类型，默认是 rsa，可以省略。

-b 指定密钥长度。对于RSA密钥，最小要求768位，默认是2048位。

-C 设置注释文字，比如邮箱，是标记SSH密钥的一种可选方式。

-f 指定密钥存储文件名，省略则使用默认文件名，会生成私钥 id\_rsa 和公钥 id\_rsa.pub。

首先 ssh-keygen 会确认密钥的存储位置（默认是~/.ssh/id\_rsa，~在windows系统中代表系统当前登录用户文件夹）直接回车，然后它会要求你输入两次密钥口令，如果你不想在使用密钥时输入密码，将其留空直接回车即可。 然而，如果你使用了密码，那么请确保添加了 -o 选项，它会以比默认格式更能抗暴力破解的格式保存私钥。你也可以用 ssh-agent 工具来避免每次都要输入密码。

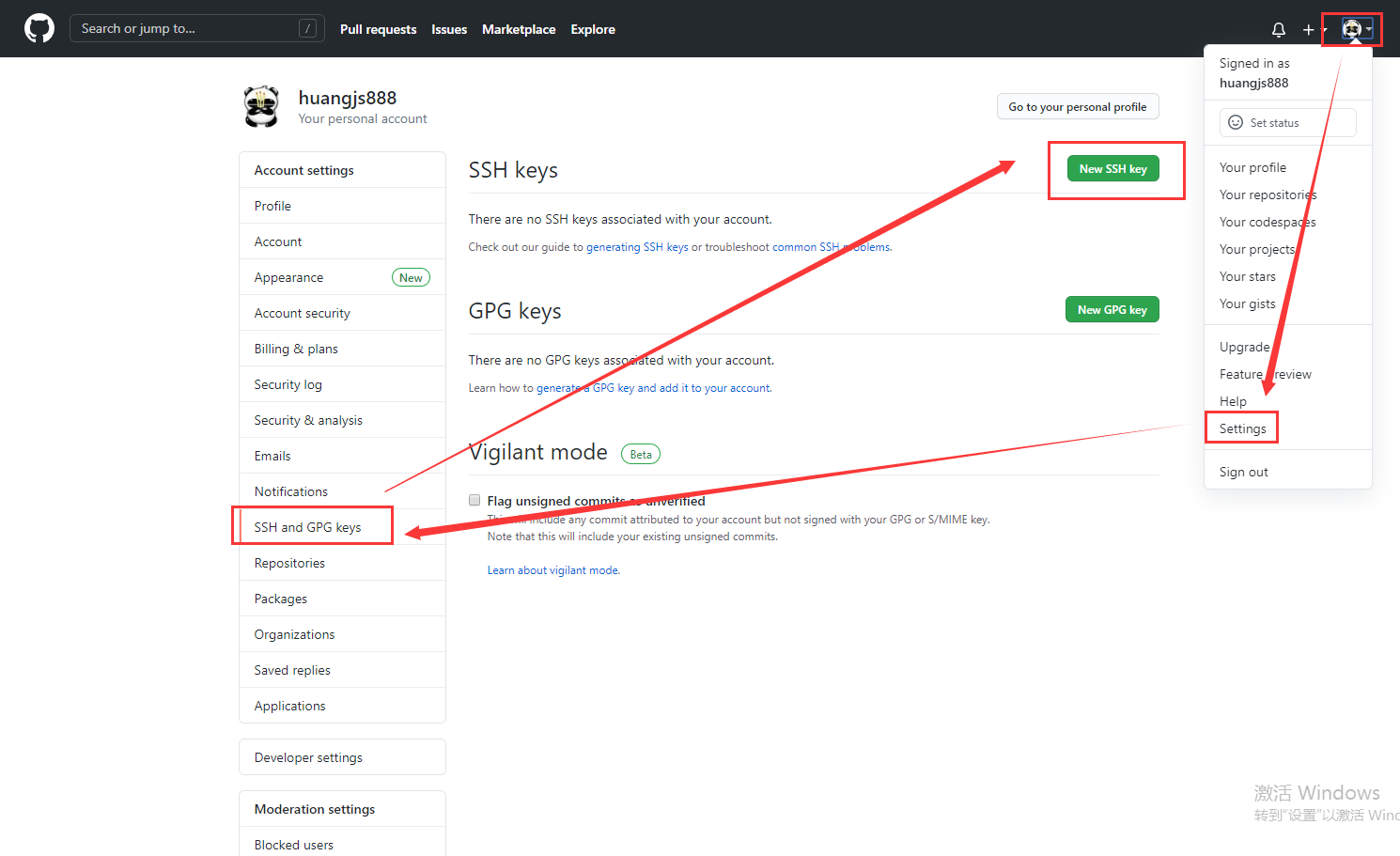


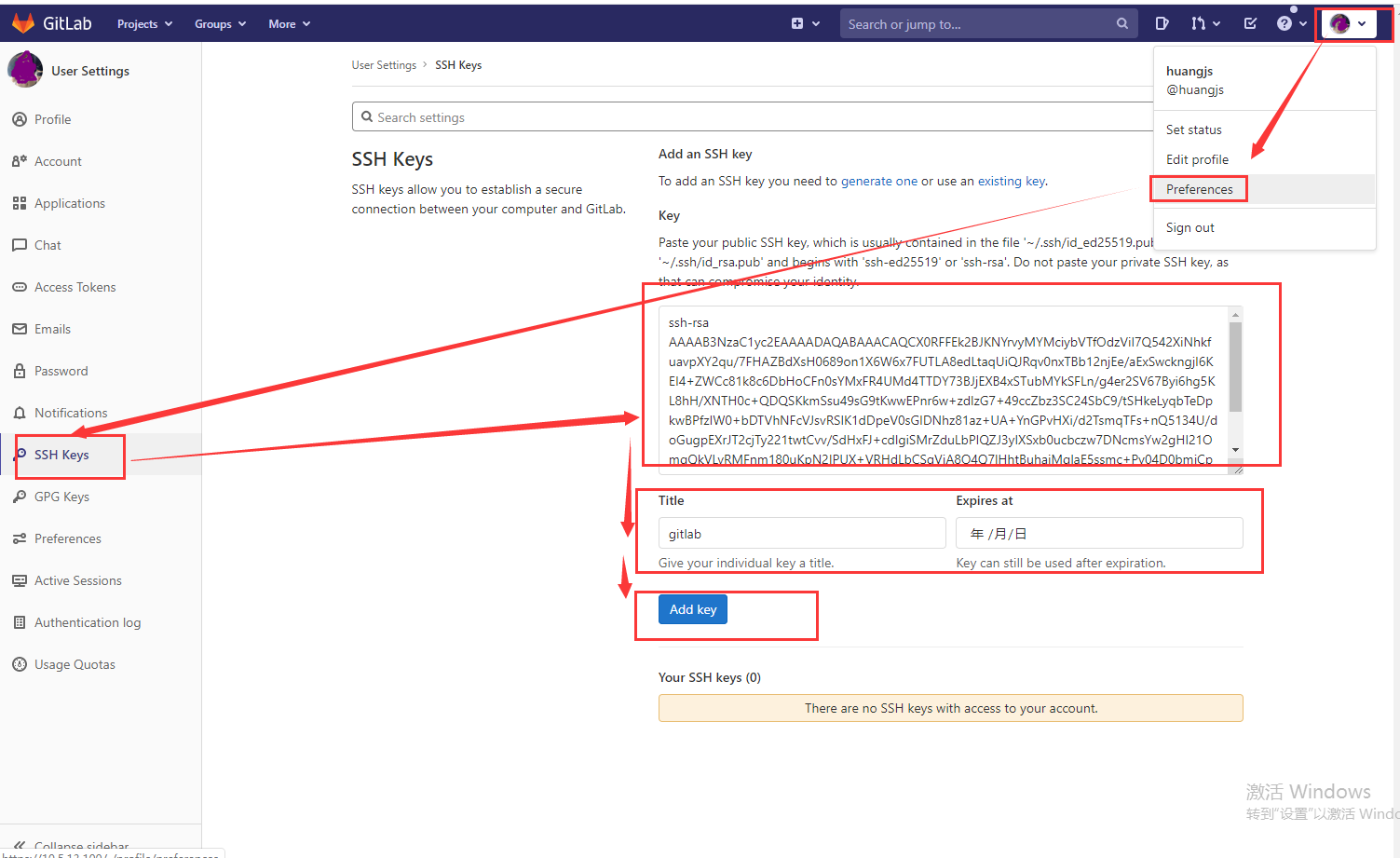
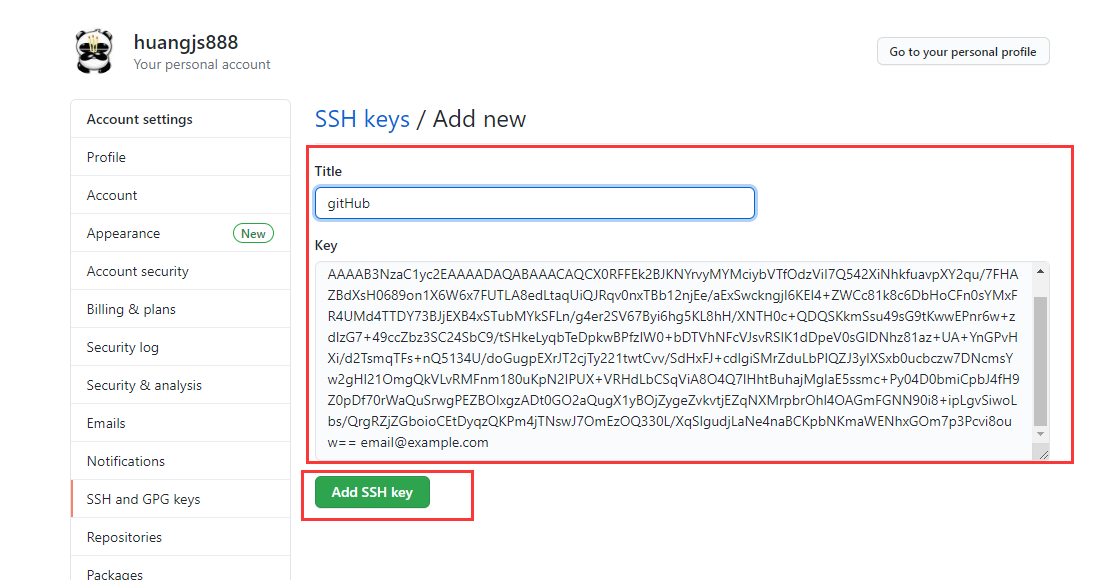
3、添加你的 SSH key 到 github/gitlab上面去

a、首先你需要拷贝公钥id\_rsa.pub 文件的内容，你可以找到文件用编辑器打开复制，也可以用git命令复制该文件的内容，如：

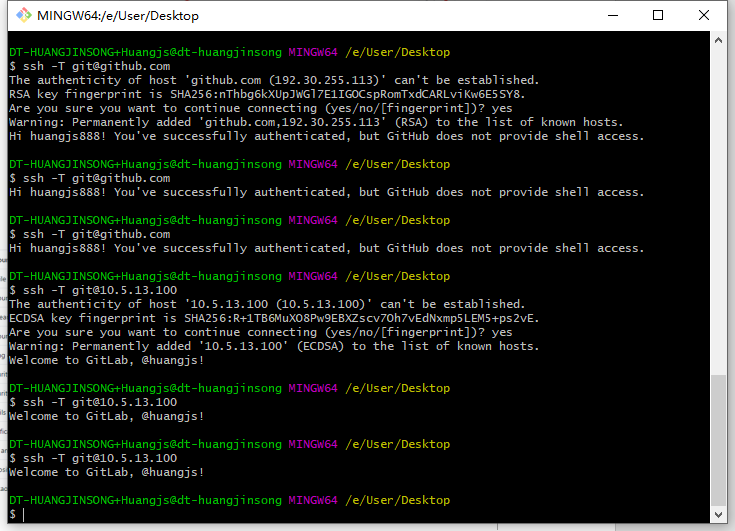


b、登录你的github/gitlab账号，找到对应的填写ssh的地方新建粘贴进去。

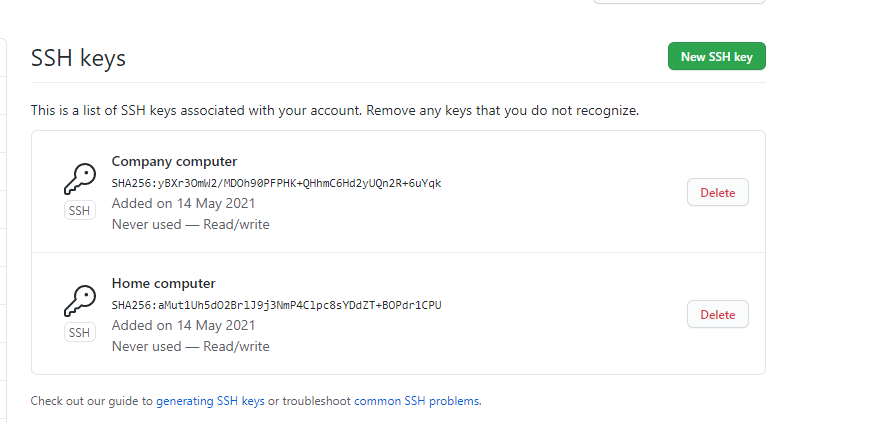




b、使用git bash输入ssh -T [git@github.com](mailto:git@github.com)(或git@gitlab地址)验证ssh是否能用，第一次验证会验证所连接的github/gitlab主机的真实性，回答yes回车即可（如果未出现欢迎字样，则可以输入ssh -Tvvv [git@gitlab.com](mailto:git@gitlab.com)查看详细内容来解决问题）。



注意，如果开发更换了机器，仍然需要使用同一个github或gitlab服务器，则在另一台机器按照同样方式生成ssh-key，然后copy公钥id\_rsa.pub文件的内容，在服务器继续添加新的SSH keys。比如下面的截图我添加了公司电脑和家庭电脑两个SSH Keys。



**以上是单个ssh-key配置，如果既有github又有公司内部的gitlab，并且使用不同的git账号，这时候需要配置多个ssh-key，下面开始多ssh-key配置。**

当在同一电脑下生成多个密钥时，需要对多个密钥进行认证管理，否则密钥将无法正常使用。

1，创建个人GitHub的SSH key

ssh-keygen -t rsa -b 4096 -C email@github.com -f ~/.ssh/github\_id\_rsa

在~/.ssh/目录会生成gitlab\_id\_rsa私钥和gitlab\_id\_rsa.pub公钥。将公钥中的内容粘帖到GitHub服务器的SSH keys的配置中。

2，创建公司GitLab的SSH key

ssh-keygen -t rsa -b 4096 -C email@company.com -f ~/.ssh/company\_id\_rsa

在~/.ssh/目录会生成company\_id\_rsa私钥和company\_id\_rsa.pub公钥。将公钥中的内容粘帖到公司GitLab服务器的SSH keys的配置中。

3，创建个人虚拟机测试GitLab的SSH key（这里不加-f指定路径，使用默认的路径名）

ssh-keygen -t rsa -b 4096 -C email@test.com

在~/.ssh/目录会生成id\_rsa私钥和id\_rsa.pub公钥。将公钥中的内容粘帖到虚拟机GitLab服务器的SSH keys的配置中。

4，在~/.ssh/目录下创建config文件（注意无后缀名），输入这几个密钥的配置信息（config文件主要用于管理密钥信息，并没有实际功能，不添加config文件直接为密钥添加信任也是可以的。但是考虑到密钥信息的管理，强烈建议添加config文件，配置好密钥的信息）。

# 注释

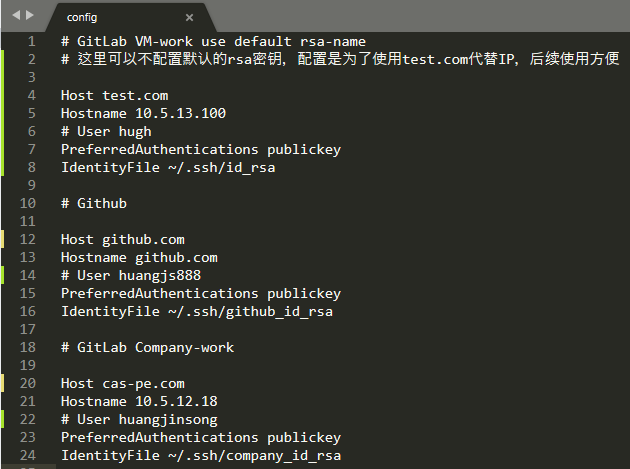
Host 地址映射（可不写此项）例如：github.com，在后续使用仓库地址的地方使用该映射代替（如果不使用，会找不到该主机，后面解密中会详细解释）

Hostname 仓库服务器地址：github.com或10.5.12.18

User 登录仓库的用户名（可不写此项）：huangjs888

PreferredAuthentications 首选认证（公钥）： publickey

IdentityFile 对应的认证文件路径：~/.ssh/company\_id\_rsa



6，最后分别对每个服务输入ssh -T git@github.com(或git@10.5.12.18)验证ssh是否能用。

发现github和默认的能用，但是Company-work不能用，这是为什么？

解密：其实是上面confg中配置的Host的原因，当在config中对ssh-key配置了Host，则该字段将会替代git命令中所有使用该仓库地址中的主机名（IP）部分。比如以下命令：

ssh -T git@10.5.12.18

git clone git@10.5.12.18:xxx/xxx.git

git remote add origin git@10.5.12.18:xxx/xxx.git

如果配置了Host，则该命令失效(会被拒绝或者要求输入密码)，必须换成：

ssh -T git@ cas-pe.com

git clone git@cas-pe.com:xxx/xxx.git

git remote add origin git@cas-pe.com:xxx/xxx.git

因为命令会根据@后面的地址找到对应的密钥，如果配置了Host，命令只会比较Host，而或略Hostname，如果未配置Host，则会比较Hostname，最后比较完，如果匹配，则拿其后的IdentityFile作为密钥，若不匹配，就会拿默认生成的密钥。上述失效的原因就是没有匹配到配置，拿了默认的密钥然后被拒绝或要求输入密码。

github可以使用，是因为配置的Host和Hostname一样，所以能够正确匹配。

也可以对默认的密钥进行配置，使用一个固定Host代替Hostname，好处在于后续git命令remote add地址时，Host映射会被写入项目目录下的.git/config内，假设远程仓库地址迁移更改，IP变化，那么只需要修改~/.ssh/config的Hostname即可，与该远程仓库关联的所有项目内.git/config都无需修改，也无需重新git remote add origin git@{IP地址}:xxx/xxx.git去修改追踪的远程仓库了。默认密钥配置后，使用Host和Hostname都可以成功，因为使用Hostname时配置没找到，直接找的默认密钥，而配置本身也就是默认密钥，所以能成功。

因为配置多个密钥是因为拥有多个git账号，比如个人GitHub账号，公司GitLab账号，个人测试GitLab账号。所以在不同的项目里用git管理项目之前，需要先做个配置，进入项目，右键git bash输入下面命令（-local可以省略不写，根据需求填写对应的用户名及邮箱）：

git config --local user.name huangjinsong

git config --local user.email jinsong\_huang@cas-pe.com

可以把常用的账号（比如GitHub账号）配置为全局的，这样，使用这个账号的项目，就不用再配置当前项目的了。

git config --global user.name huangjs888

git config --global user.email xxxxxxxx@qq.com

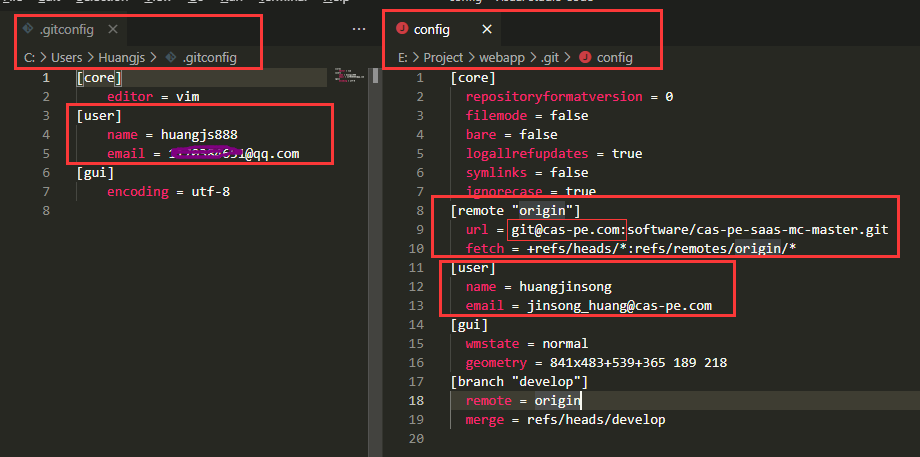
其实还有个--system级别的，设置后不同的电脑用户登录之后也可以用这个git账号。

以上都可以不使用命令，找到对应的config文件进行修改。

--local: 项目目录/.git/config

--global: ~/.gitconfig或 ~/.config/git/config（~在windows中是系统当前登录用户文件夹）

--system: /etc/gitconfig（windows是用户文件夹下的公用文件夹下）



7，对于多个密钥管理，如果不使用config配置，那么需要对每个密钥添加信任，才能使用。打开git bash 输入命令：

eval $(ssh-agent -s)

ssh-add ~/.ssh/github\_id\_rsa

ssh-add ~/.ssh/company\_id\_rsa

eval 代表开启代理认证指令。

ssh-add 代表添加ssh文件认证，是把专用密钥添加到ssh-agent的高速缓存中。

可以执行 ssh-add -d ~/.ssh/id\_xxx.pub 从ssh-agent中删除密钥，也可以执行 ssh-add –l 查看ssh-agent中的密钥列表。

执行以上命令就可以不用配置config，配置了config，也无需再执行上述命令。但是有个问题，就是该操作是一次性的，只会在当前开启的git bash有效，一旦关闭git bash，下次再开启使用，就必须再运行一次。并且ssh-agent开启后不会自己关闭，即使再下次启动git bash也不会关闭，而且也不能再使用，必须重新eval开启一个ssh-agent，如果要关闭它，只能执行命令ps –ef | grep ssh-agent或ps -a找到对应的agent程序pid，用kill -9 pid号来杀死进程。

8，配置了ssh之后，git关联远程仓库直接使用git@开头的ssh地址即可，而且push代码时无需输入密码，如果没有配置ssh，直接使用https方式关联远程仓库，则地址仓库地址直接使用https://开头的，在push代码时会弹出输入Git账号和密码的框框，比较麻烦。

下面方法可以记住密码，后续再push无需输入密码。



如果账户密码不再使用，可以执行命令清除：

git config --global --unset credential.helper

也可以使用Windows自带的 工具凭据管理器 来添加或删除刚才的账号密码



git使用https和ssh方式命令示例：

git clone https://github.com/huangjs888/testCode.git

git remote add origin https://github.com/huangjs888/testCode.git

git clone git@github.com:huangjs888/testCode.git

git remote add origin git@github.com:huangjs888/testCode.git

后续直接使用关键字origin代替了远程仓库地址了，详情查看git命令使用的相关文档。