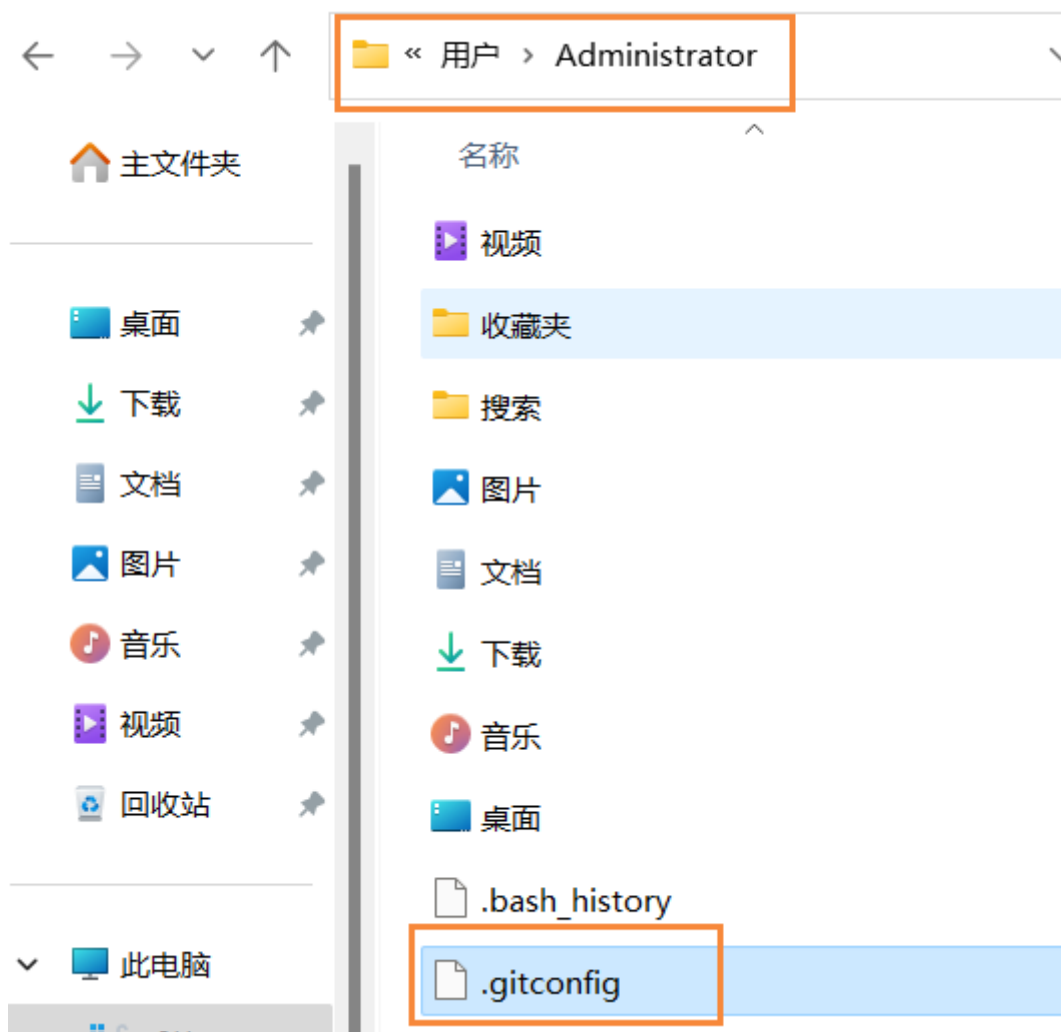


# GitHub使用帮助

## 1 配置代理

由于网络原因，有的时候使用https的方式克隆、获取、拉取、推送代码会出现连接失败的问题，此时可以通过设置代理服务器的方式来解决。

使用代理的时候需要配置你的git，找到你的.gitconfig配置文件，一般在当前用户的根目录，如下图所示：



配置示例如下：

```
[credential "https://gitee.com"]
    provider = generic
[user]
    name = xxxxxx
    email = xxxxxx@xxxx.com
[http "https://github.com"]
    proxy = http://127.0.0.1:7890
[safe]
    directory = *
[core]
    compression = 9
```

```
1 [http "https://github.com"]
2   proxy = http://127.0.0.1:7890
```

将 `http://127.0.0.1:7890` 替换成你自己的代理服务器和端口。

### 补充一下

我们在Windows CMD 里面使用命令行下载资源或安装程序的时候，也可能需要使用代理才能完成，可以通过下面步骤给控制台设置代理。

在执行命令前，首先设置代理，根据你的需要选择性的执行下面的命令：

```
1 # 设置HTTP代理
2 set http_proxy=protocol://proxyserveraddress:port
3 # 设置HTTPS代理
4 set https_proxy=protocol://proxyserveraddress:port
5 # 如果代理服务器需要用户名和密码进行验证，则需要在命令中添加用户名和密码信息
6 set http_proxy=protocol://username:password@proxyserveraddress:port
7 set https_proxy=protocol://username:password@proxyserveraddress:port
```

参数说明：

- `protocol` 是代理服务器协议，根据你代理服务器实际情况来了，通常是`http`或`https`
- `proxyserveraddress` 是代理服务器的地址
- `port` 是代理服务器的端口号
- `username` 和 `password` 分别是代理服务器的用户名和密码

## 2 配置SSH

有时候通过SSH方式也能解决GitHub访问慢的问题。

### 2.1 生成秘钥对

打开Git bash工具，然后输入命令

```
1 ssh-keygen -t rsa -C "github邮箱"
```

遇到输入提示直接回车即可如下图所示

```
MINGW64:/c/Users/Administrator

Administrator@Naaman MINGW64 ~
$ ssh-keygen -t rsa -C "2060682473@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Administrator/.ssh/id_rsa):
Created directory '/c/Users/Administrator/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Administrator/.ssh/id_rsa
Your public key has been saved in /c/Users/Administrator/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:a0+CJdDEWRIGIZuROcABIHvRha6ZHeffDsHMwMxB72I 2060682473@qq.com
The key's randomart image is:
+---[RSA 3072]-----+
|@o+ooBO+.
|=..*oBo+
|..+..*
| . o..*
|  = +E S
| + ...= o
|      ..=..
|      ..=.
|      .o
+---[SHA256]-----+

Administrator@Naaman MINGW64 ~
$ |
```

## 2.2 添加私钥到秘钥管理器

ssh-agent就是一个密钥管理器，运行ssh-agent以后，使用ssh-add将私钥交给ssh-agent保管，其他程序需要身份验证的时候可以将验证申请交给ssh-agent来完成整个认证过程。

确认一下秘钥管理器是否正常运行，执行ssh-agent命令，如下图所示。

```
Administrator@Naaman MINGW64 ~
$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-37i9fhpQZAKY/agent.1972; export SSH_AUTH_SOCK;
SSH_AGENT_PID=1973; export SSH_AGENT_PID;
echo Agent pid 1973;
```

添加私钥到秘钥管理器，执行下列命令

```
1 | ssh-add ~/.ssh/id_rsa
```

执行结果如下图所示，表示出现了错误

```
Administrator@Naaman MINGW64 ~
$ ssh-add ~/.ssh/id_rsa
Could not open a connection to your authentication agent.
```

通过下面的方式解决

- 首先查询进程

```
1 | ps aux | grep ssh
```

- 然后杀死进程

```
1 kill -9 进程号
```

- 进入用户名目录下的.ssh目录，执行如下命令

```
1 cd ~/.ssh
2 exec ssh-agent bash
3 eval ssh-agent -s
```

```
Administrator@Naaman MINGW64 ~
$ ps aux | grep ssh
1973      1    1973      13784  ?        Ss   15:48:39 /usr/bin/ssh-agent
Administrator@Naaman MINGW64 ~
$ kill -9 1973
Administrator@Naaman MINGW64 ~
$ cd ~/.ssh
Administrator@Naaman MINGW64 ~/.ssh
$ exec ssh-agent bash
Administrator@Naaman MINGW64 ~/.ssh
$ eval ssh-agent -s
SSH_AUTH_SOCK=/tmp/ssh-K7mYXG14ZAJY/agent.2010; export SSH_AUTH_SOCK;
SSH_AGENT_PID=2011; export SSH_AGENT_PID;
echo Agent pid 2011;
```

- 再次执行添加命令

```
1 ssh-add ~/.ssh/id_rsa
```

执行结果如下图所示表示添加成功

```
Administrator@Naaman MINGW64 ~/.ssh
$ ssh-add ~/.ssh/id_rsa
Identity added: /c/Users/Administrator/.ssh/id_rsa (20090027...)
```

## 2.3 配置GitHub

查看公钥

```
1 cat ~/.ssh/id_rsa.pub
```

```
Administrator@Naaman MINGW64 ~/.ssh
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDd4Ux009iTqkRg0FM8z67KJgLf37Vvgf205QNYJWoC
KTci9qRWUGzc4n81JYA8dmAUynwsE0Op/v3P7LcrnEv0xADTW9DIO+66zGsNhX2UNEawS94P3GzV8CUL7
11B0u5d3r8ZTWkiS1RpAvRU3CAkckZ3mNnnLsSeLw0ATm1jyir1ArhwXir20MaA0DtZjAQ0IYFBsne5A
dNCHdb0cdme6euQSkcVUa8zjew21xpQFovni33LCUF3odm1xGhCtNnu+PChcICRaATDywg5Jcm9zYt6hs
X/013wM1Uom15HxhXs6IkFTFK505tS23L21yzQ6UeYCefEYfR0FrJoUFbqGZyJysU/utIIkMJVvL/7vn1
4m91yRdCpyk0i3B6unqNpp3fCcczxt4k8omSqwzzGsZVps9qgZC4kI6XVi2mrM2eXrBnGEBya2n+juUir
```

复制公钥备用。

打开GitHub SSH Key新建链接: <https://github.com/settings/ssh/new>

然后在执行下图所示的操作

## SSH keys / Add new

Title

awei-windows

1 名称随意

Key type

Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQd14UxG03nqkRgCFM0-67K1nl f37W4g1205QnyJWoQbe/DmK7cl9qrWVUSz.
4n81JYA8dmAUynwsEOOp/v3P7LcnEvOxADTW9DIO+66zGsNhx2UNEawS94P3GzV8CUL7x/WYU11B0u5d3r8ZTWKiSlRp
AvRU3CAkckZ5minnnLsSelw0ATm1jyidA1hwXir20MdaA0D+ZiAQOIVEP3nc5A1e45yPdNCHdb0cdme6euQSkcVUa8zjeW2lxp
QFovnl33LCUF3odm1xGhCtNnu+PCbclCRaATDyMq5lcm9zYt6bcGcU11yOT3WMIUom15HxhXs6lkFTFK505tS23L21yzQ
6UeYCefEYfR0FrJoUFbqGZyJysU/utllkMJVvL/7vnlPeVYz4m91yRdCpykOi3B0unq1pp5fcccxt4k8omSqwzzGszVps9qgZC4
kl6XVi2mrM2eXrBnGEBYa2n+juUimlElMeEEF9Lg1sBrrr24i356vtAVPDDuBJAcvTyAM=
```

2 粘贴复制过来的公钥

Add SSH key

## 2.4 连接测试

执行命令

```
1 ssh -T git@github.com
```

执行结果如如下图所示

```
Administrator@Naaman MINGW64 ~/.ssh
$ ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqu.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

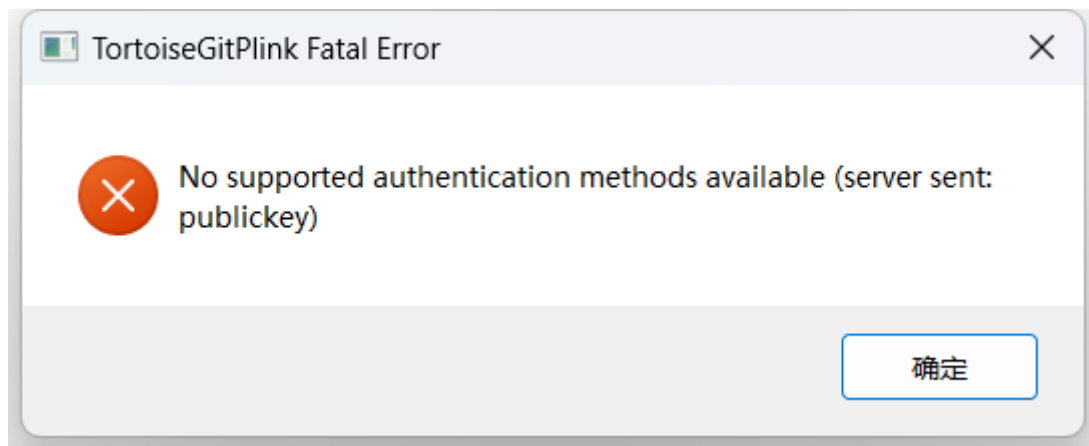
输入yes, 执行结果如下图所示

```
Administrator@Naaman MINGW64 ~/.ssh
$ ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqu.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi zero-awei! You've successfully authenticated, but GitHub does not provide shell access.
Administrator@Naaman MINGW64 ~/.ssh
```

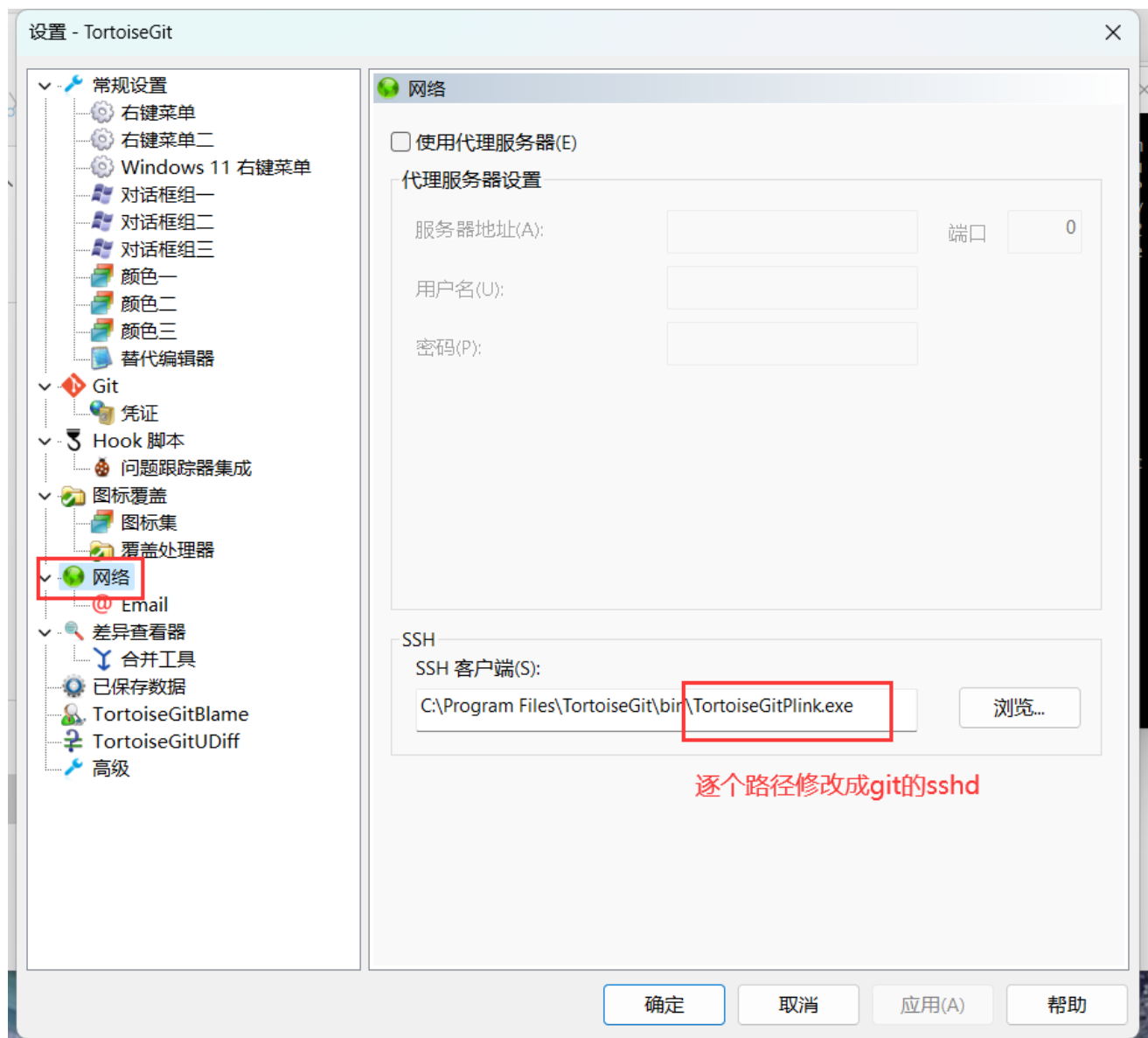
到此说明配置成功了, 你就可以用ssh方式克隆和推送代码到GitHub了

提示:

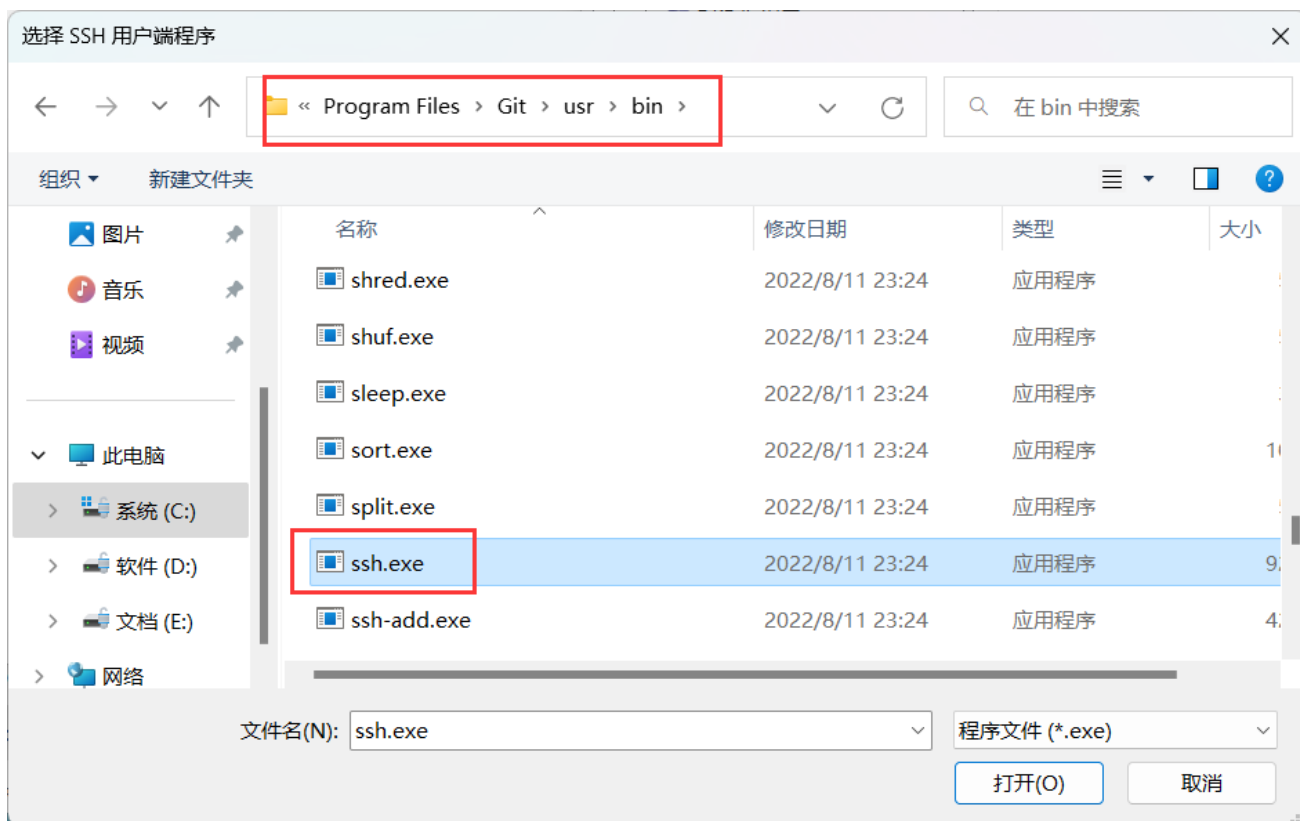
如果我们用乌龟clone项目的时候出现下图的错误提示, 我们需要做以下处理。



我们需要修改TortoiseGit 网络的SSH客户端为 git 服务器的ssh.exe



点击浏览，示例路径如下



再次尝试即可正常clone

## 2.5 扩展

如果需要配置多个凭证，可以在.ssh目录下面建立一个config文件书写配置来对应

下面是一个配置示例：

```
1 #github
2 Host github.com
3 HostName github.com
4 PreferredAuthentications publickey
5 IdentityFile ~/.ssh/id_rsa
6 User zero-awei
```

配置文件说明：

每个账号单独配置一个Host，HostName和IdentityFile三个属性即可。

- **Host**：别名，可以取为自己喜欢的名字，不过这个会影响git相关命令，例如：
  - Host mygithub 这样定义的话，命令如下，即git@后面紧跟的名字改为mygithub，相应的clone命令为：

```
1 git clone git@mygithub:xxxx/xxxxx.git
```
- **HostName**：映射Host对应的真正的域名，如：github.com
- **IdentityFile**：配置私钥文件所在位置，如：~/.ssh/id\_rsa
- **PreferredAuthentications**：配置登录时用什么权限认证
  - 可设置为：publickey,password publickey,keyboard-interactive等

- User: 配置使用用户名

## 3 个人访问令牌

有时候我们在使用 `https` + 用户名密码验证的方式访问私库时，你可能会出现下面的错误：

```
1 remote: Support for password authentication was removed on August 13, 2021. Please
  use a personal access token instead.
2 remote: Please see https://github.blog/2020-12-15-token-authentication-
  requirements-for-git-operations/ for more information.
```

出现这个错误的原因是，2021年8月13日以后，不再支持用户名密码的方式验证了，需要使用个人访问令牌（personal access token），也就是把你的密码需要替换成token，当然你可以不使用token，使用ssh方式操作仓库也是可行的。

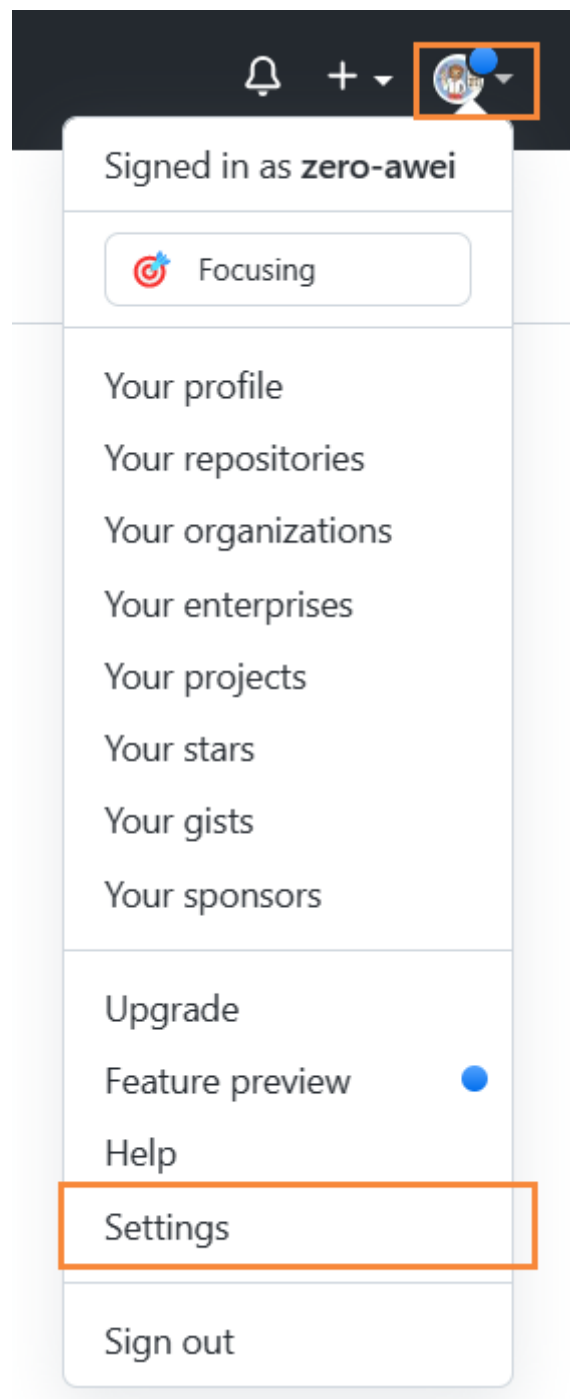
参考链接：<https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/>

下面来看一看生成和使用令牌的步骤。

### 3.1 生成令牌

首先进入个人设置页面





找到并打开开发设置

## Security

🛡️ Code security and analysis

## Integrations

🔌 Applications

🕒 Scheduled reminders

## Archives

📖 Security log

📖 Sponsorship log

🔗 Developer settings

找到个人访问令牌

🔌 GitHub Apps

👤 OAuth Apps

🔑 Personal access tokens 1

Fine-grained tokens Beta

Tokens (classic) 2

### Personal access tokens (classic)

Generate new token ▾

Need an API token for scripts or testing? [Generate a personal access token](#) for quick access to the [GitHub API](#).

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

点击生成令牌

Generate new token ▾ 1

Generate new token Beta

Fine-grained, repo-scoped

Generate new token (classic) 2

For general use

输入相关参数并生成令牌

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

test-token 1 备注

What's this token for?

### Expiration \*

30 days 2 设置有效期 will expire on Tue, Apr 18 2023

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

- ☒ repo 3 分配访问权限，比如我这里分配了对仓库的访问权限
  - ☒ repo:status Access commit status
  - ☒ repo\_deployment Access deployment status
  - ☒ public\_repo Access public repositories
  - ☒ repo:invite Access repository invitations
  - ☒ security\_events Read and write security events

- ☐ admin:ssh\_signing\_key Full control of public user SSH signing keys
- ☐ write:ssh\_signing\_key Write public user SSH signing keys
- ☐ read:ssh\_signing\_key Read public user SSH signing keys

Generate token 4

Cancel

如下是生成的令牌，记得保存，下次就看不到了。

## Personal access tokens (classic)

Generate new token ▾

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_cw8oAw7tDAhns55fVwKuElvKcwnyif3AKg7L 1 点击这里可以复制

Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

## 3.2 使用令牌

### 用法一：直接将token作为密码使用

在需要输入密码的地方，换成token即可。

## 用法二：修改现有项目的url

```
1 git remote set-url origin https://<TOKEN>@github.com/<USERNAME>/<REPO>.git
```

- <TOKEN>：换成你刚生成的令牌
- <USERNAME>：换成你 GitHub 的用户名
- <REPO>：换成你对应的仓库名称

## 用法三：在克隆项目的时候在github.com前面加个令牌

```
1 git clone https://<TOKEN>@github.com/<USERNAME>/<REPO>.git
```

后面两种方式都可以通过修改.git目录下面的config文件实现，只需要要找到remote配置，然后在url链接值里面加上令牌即可

```
1 [remote "origin"]
2   url = https://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@github.com/zero-
   awei/test.git
3   fetch = +refs/heads/*:refs/remotes/origin/*
```