

# Git 基本使用方法

Document Version Control			
Version	Date	Author	Changes
1.0	2022/12/26	王雨霄	Git 安装方法、基础使用方法

## 目录

一、安装 Git: .....	4
二、在本地电脑使用 Git: .....	5
三、将本地仓库提交至 GitHub 服务器: .....	9

# 一、安装 Git:

## 1、下载 Git 安装包——<https://git-scm.com/>

The image shows two screenshots from the Git website. The top screenshot is the homepage, which features the Git logo and tagline "--fast-version-control". It describes Git as a free and open source distributed version control system. A diagram on the right shows a branching model with stacks of code. Below this are links for "About", "Documentation", "Downloads", and "Community". A red box highlights a monitor icon displaying the latest source release "2.39.0" and a "Download for Windows" button. The bottom screenshot is the "Download for Windows" page, which provides instructions on how to download and install Git. It lists options for standalone installers (32-bit and 64-bit), portable versions, and using the winget tool. A red box highlights the "64-bit Git for Windows Setup" link. The page also includes a command to install winget and information about the current source code release version 2.39.0.

**Download for Windows**

Click [here to download](#) the latest (**2.39.0**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **5 days ago**, on 2022-12-21.

**Other Git for Windows downloads**

**Standalone Installer**

[32-bit Git for Windows Setup.](#)

**64-bit Git for Windows Setup.**

**Portable ("thumbdrive edition")**

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

**Using winget tool**

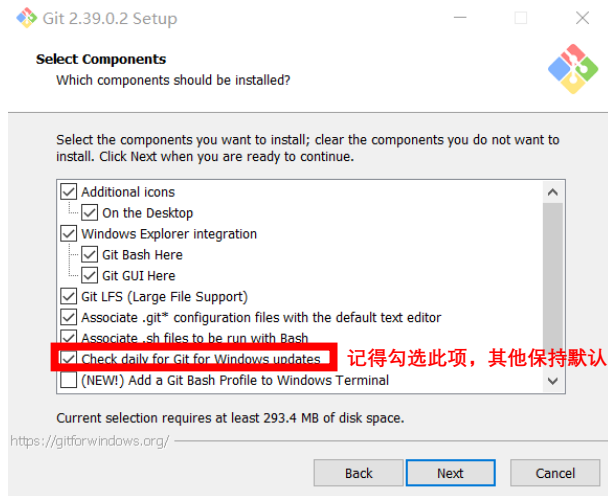
Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.39.0**. If you want the newer version, you can build it from [the source code](#).

如果打不开官网，也可以在下面这个链接下载：[https://pan.baidu.com/s/1c-lv-ZTyO4ytG\\_WK2X6flw?pwd=2qnm](https://pan.baidu.com/s/1c-lv-ZTyO4ytG_WK2X6flw?pwd=2qnm)

## 2、软件安装（除下图外，其他选项均保持默认）：



## 3、安装成功后，Git 出现在菜单栏中：

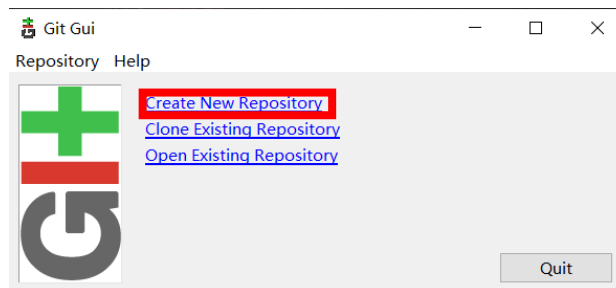


## 二、在本地电脑使用 Git：

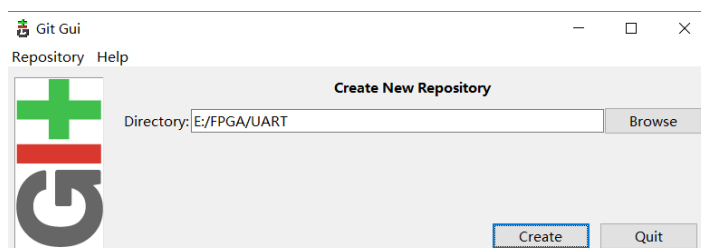
Git 中将一个工程的存储空间称为“Repository”，用以记录工程的具体内容、修改记录等信息；“Repository”可分为存储在本地的“Repository”，以及存放在 GitHub/GitLab 等托管服务器上的远程“Repository”。

Git 可通过 GUI 界面或命令行的方式进行操作，下面将介绍如何使用 GUI 界面建立本地仓库，并在第三部分中介绍如何通过命令行实现本地仓库与远程仓库的交互。

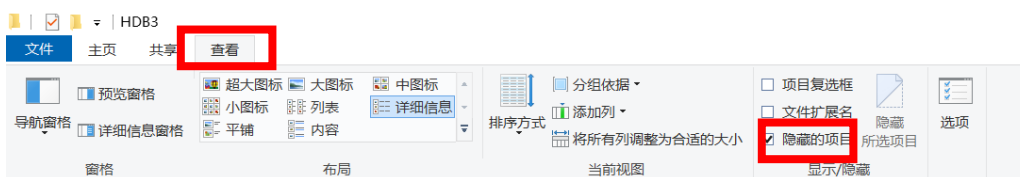
### 1、建立本地新仓库：



## 2、选择需存入仓库中的项目所在的路径：



## 3、进入项目路径，点击“查看”，将隐藏项目设置为可见：

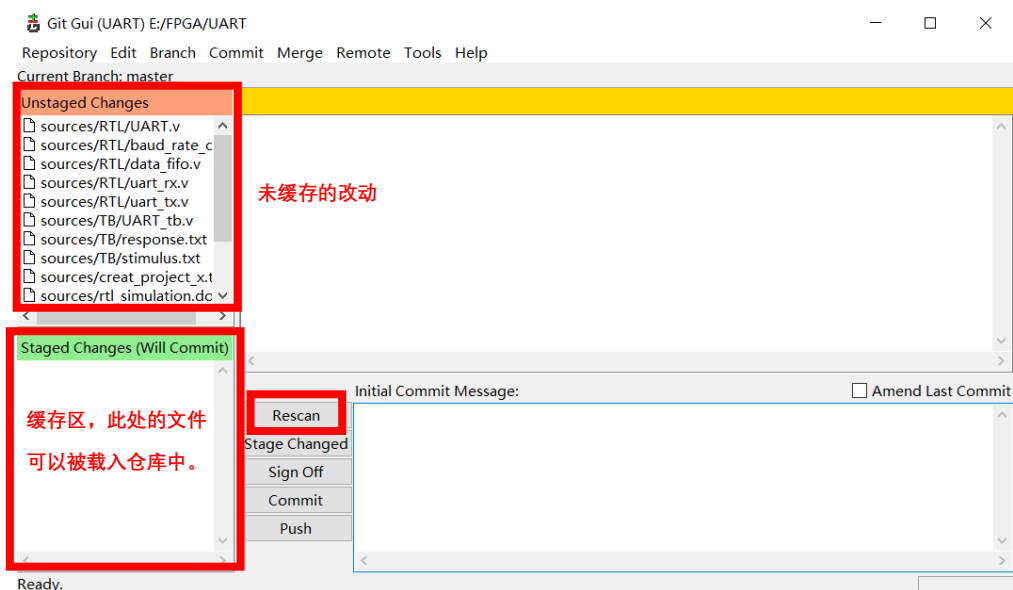


## 4、在项目路径下能看到".git"文件夹，说明仓库建立成功：

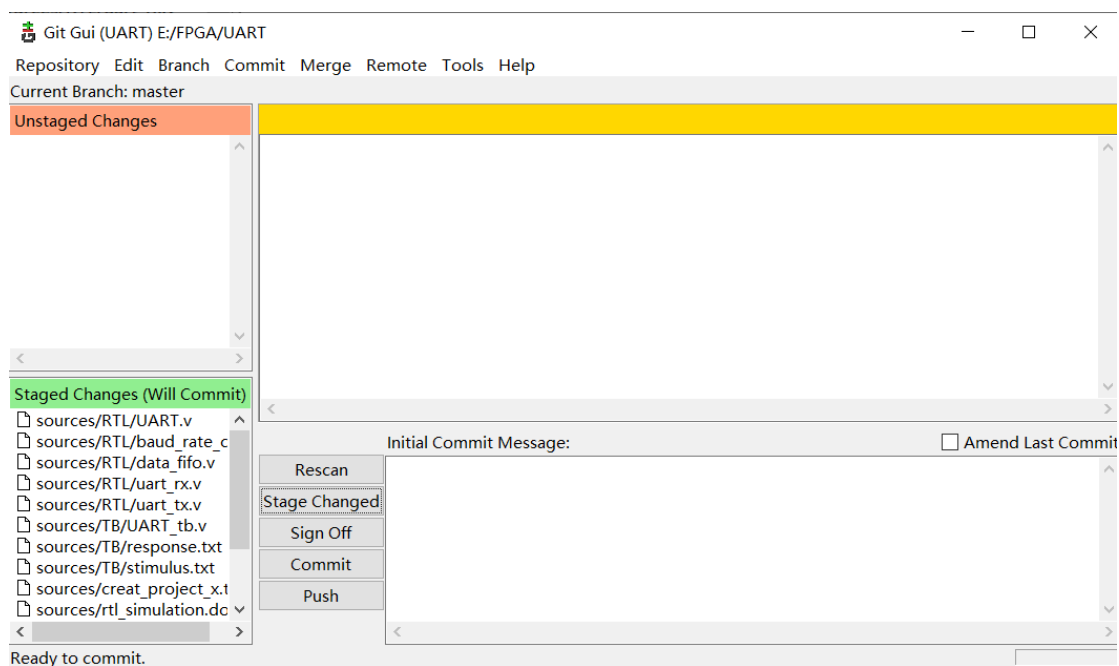
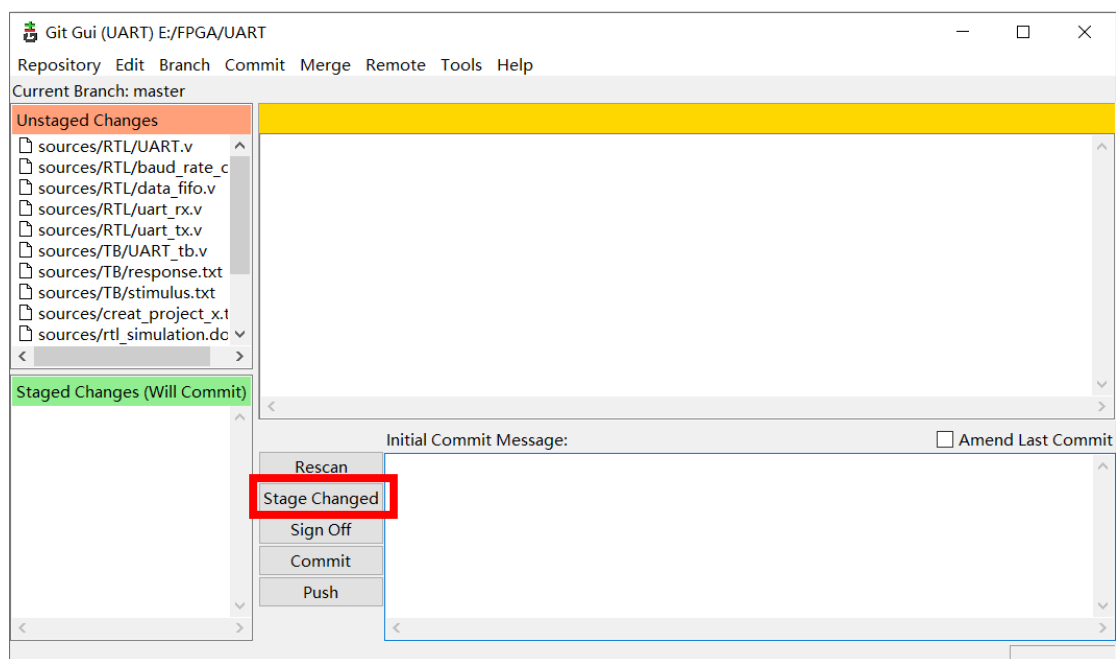


## 5、将项目存入仓库：

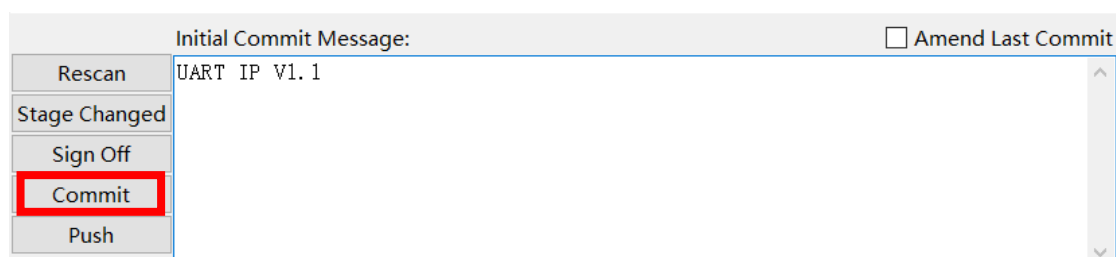
点击"Rescan"，扫描得到相对于上次的存储内容而言发生改动的文件（新建立仓库的时候，项目中所有的文件对于仓库而言都是新文件）：



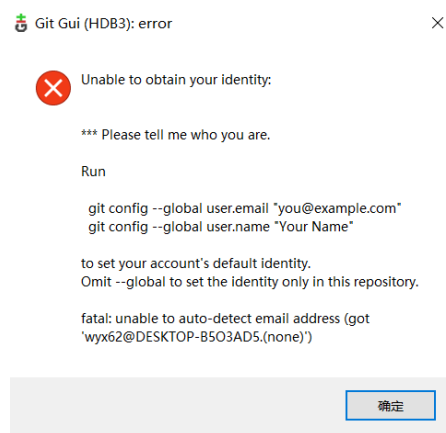
点击"Stage Changed", 将发生改动的文件移入缓存区:



输入本次提交的项目描述, 点击"Commit", 将项目提交至仓库中:



第一次使用时，点击"Commit"后通常会出现如下错误提示，意思是没有为本机的 Git 软件设定全局身份验证（在复杂项目开发时，常需要多人协作，Git 不仅会记录每次提交的项目内容，还需要记录是谁提交的）：



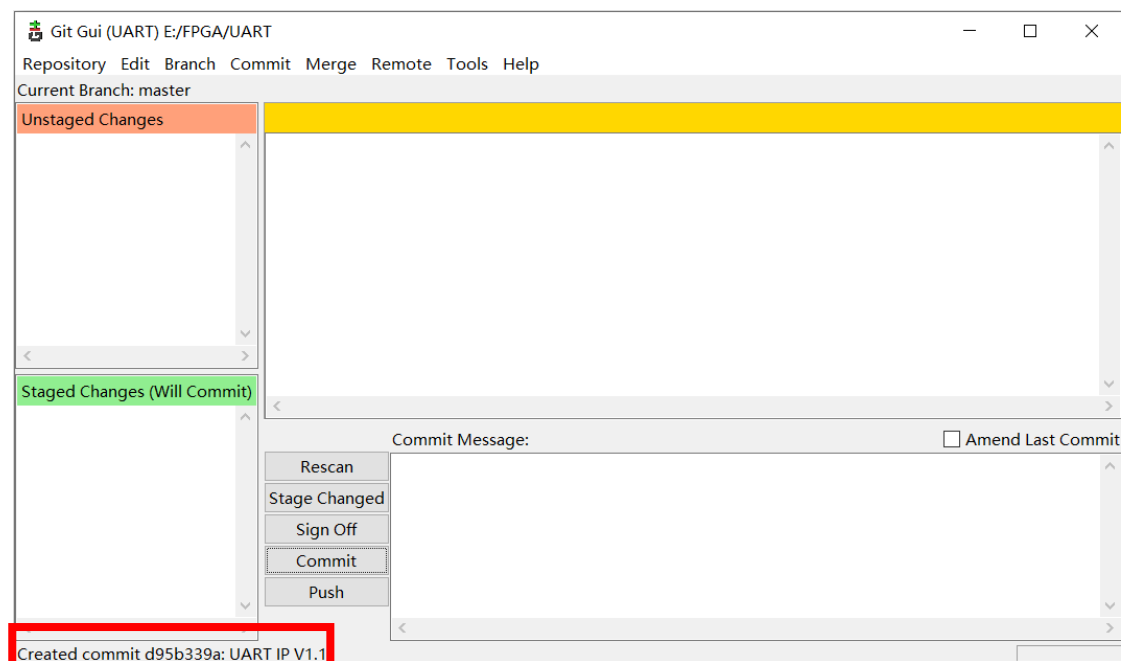
为解决这一问题，需要在系统命令窗输入如下指令：

**git config --global user.email "邮箱(最好是注册 GitHub 的邮箱)"**  
**git config --global user.name "昵称(最好写自己名字的拼音)"**

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.2364]
(c) Microsoft Corporation。保留所有权利。

C:\Users\wyx62>git config --global user.email "wyxee2000@163.com"
C:\Users\wyx62>git config --global user.name "wangyuxiao"
```

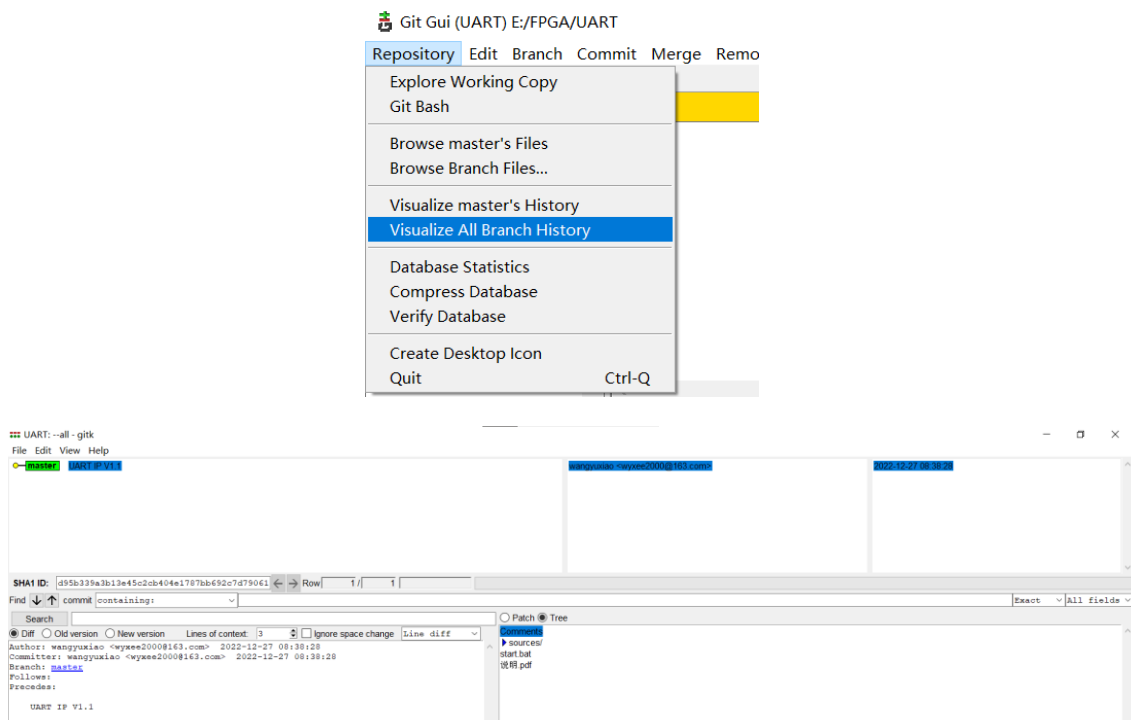
完成全局身份验证后，重新点击“Commit”：



底部出现此提示，说明项目提交成功；“d95b339a”是本次提交的项目的哈希值的前八位，被称为高位哈希值。



点击“Repository——Visualize All Branch History”，可查看提交记录：



### 三、将本地仓库提交至 GitHub 服务器：

#### 1、初次配置 VPN：

国内网络需要借助 VPN 才可以访问 GitHub，而大部分 VPN 只对浏览器进行代理，Git 软件仍通过国内网络访问 GitHub，造成文件传输不稳定。

为使 Git 经由代理服务器访问 GitHub，需在“Git Bash”中输入以下命令：

```
git config --global http.proxy socks5://127.0.0.1:10808
```

```
git config --global https.proxy socks5://127.0.0.1:10808
```

10808 是 VPN 软件在我电脑上的本地监听端口号，需要在具体的 VPN 软件中查看：

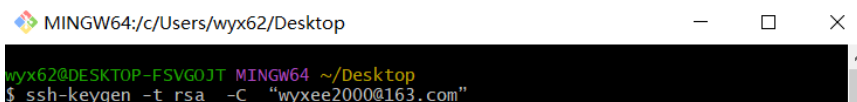


#### 2、初次配置 SSH 密钥：

Git 和 GitHub 之间可建立 SSH 连接或 Http 连接，使用 SSH 连接将 Git 接入 GitHub 时，需要依赖 SSH 密钥进行身份验证。

为生成 SSH 密钥，需要在 Git 的命令行窗口“Git Bash”中输入以下命令：

```
ssh-keygen -t rsa -C "邮箱(设定全局身份验证时所用的邮箱)"
```

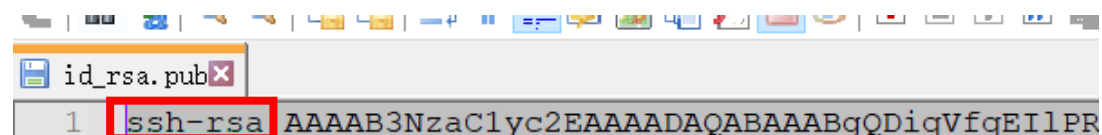


对出现的三个“Enter”输入提示依次敲击回车，直到 SSH 密钥生成完成即可，红框中的地址即为 SSH 密钥存放的地址：

```
wyx62@DESKTOP-FSVG0JT MINGW64 ~/Desktop
$ ssh-keygen -t rsa -C "wyxee2000@163.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/wyx62/.ssh/id_rsa):
Created directory '/c/Users/wyx62/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/wyx62/.ssh/id_rsa
Your public key has been saved in /c/Users/wyx62/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:yxG2HUej+bDRexrIjaFvmIoLW/JBthKYEWHdhNqGv8 "wyxee2000@163.com"
The key's randomart image is:
+---[RSA 3072]-----+
  . . . = .
  +Bo.
  +oX.o.
  .o@ % .
  .S.Bo* .
  . +o+o+
  o...E= .
  . +oB
  ..*o
+---[SHA256]-----+
```

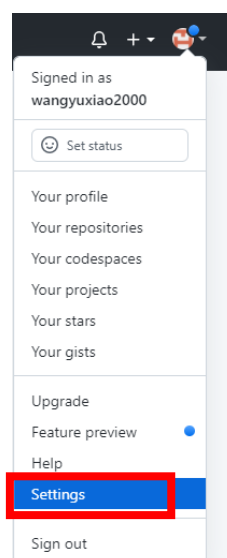
在上述地址找到“id\_rsa.pub”文件，用记事本打开，直接复制其中的全部内容（开头的“ssh-rsa”也要复制，不能仅复制后面的密钥）：

id_rsa	2022/12/26 21:19	文件	3 KB
id_rsa.pub	2022/12/26 21:19	PUB 文件	1 KB



```
1 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDiqVfqEILPR
```

登录 GitHub 网站，进入如下界面：



Account settings

Profile

Account

Appearance

Accessibility

Account security

Billing & plans

Security log

Security & analysis

Sponsorship log

Emails

Notifications

Scheduled reminders

SSH and GPG keys

Repositories

Packages

Pages

Organizations

Saved replies

Applications

## SSH keys

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

## GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

## Vigilant mode

☐ Flag unsigned commits as unverified

This will include any commit attributed to your account but not signed with your GPG or S/MIME key. Note that this will include your existing unsigned commits.

[Learn about vigilant mode.](#)

将之前复制的"id\_rsa.pub"文件中的内容粘贴进“Key”中，点击“Add SSH Key”：

[SSH keys](#) / Add new

Title

DELLG3

Key type

Authentication Key

Key

ssh-rsa  
AAAAB3NzaC1yc2EAAAADAQABAAQBAQD7h-ENfCh4m0-7-VQ1/OLV:9TA'ttqnLsQHkiiANgheGEmyFCkgnjFmk6ruql28  
oyfP/eBc5ooQmtpMThM2RIFGGCY+Rv5zRRfGcpxrtkgJLO7u3ufCsJletla6dN4vW  
WNwFEYs8xNbQGd0Jk7PqEFm+4qscHtE0Y7hyidppsUqcdOK+puil+Nlxcmqm3kLCS  
AREJVCsnDhOeUM35jOnPyFzQ4g3hlvr+iqen109WxAlSZ96UWl0+zUvmnK1kXYAs  
Br8A+91fypkqGK5w03S8mibSfc+jqcBZl2hNLNJdAkTcA3CFtyV5JTkWncvU9ym/PY  
DroQb6UjiZ2XUzzhbhg75EUGAk+NbOK+otWfyMoV2B87reXINbXBj2Hbix72eu8V8="wyxee2000@163.com"

Add SSH key


添加成功后，可以在 SSH Keys 列表中看到刚添加的密钥：

SSH keys

New SSH key

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

### Authentication Keys

 **DELLG3**  
SHA256:yxG2H#  
Added on Dec 26, 2022  
Never used — Read/write

iLW/7BthKYEWdhHlqGv8

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

在"Git Bash"中输入命令 `ssh -T git@github.com`：

```
MINGW64:/c/Users/wyx62/Desktop
wyx62@DESKTOP-FSVG0JT MINGW64 ~/Desktop
$ 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])?
```

继续输入 yes：

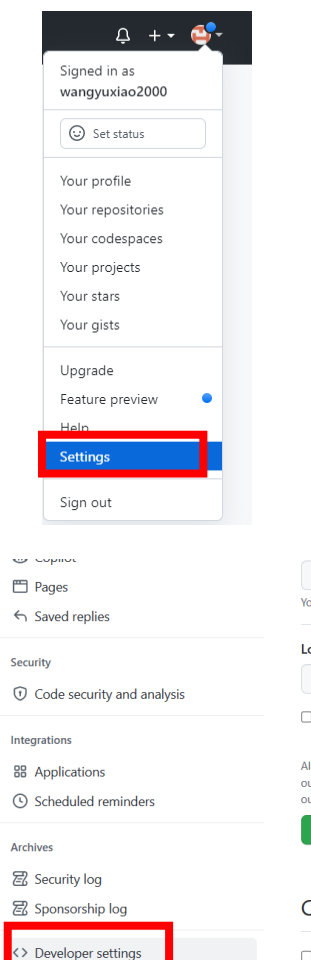
```
MINGW64:/c/Users/wyx62/Desktop
wyx62@DESKTOP-FSVG0JT MINGW64 ~/Desktop
$ ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

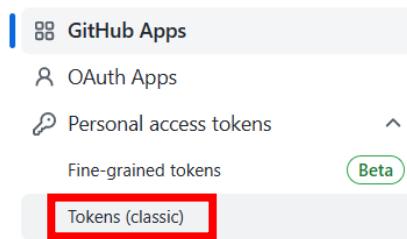
出现如下提示，代表 GitHub 与本机电脑成功建立联系：

```
Hi wangyuxiao2000! You've successfully authenticated, but GitHub does not provide shell access.
```

### 3、初次设置 tokens：

Git 和 GitHub 之间可建立 SSH 连接或 Http 连接，使用 Http 连接将 Git 接入 GitHub 时，需要通过"账号-密码"的方式或者 tokens 的方式进行身份验证；2021 年 8 月 13 日后 GitHub 不再支持"账号-密码"的方式，因此，若使用 Http 连接，需要用 tokens 来验证身份。





## Personal access tokens (classic)

Generate new token ▾

Need an API token for scripts or testing? [Generate a personal access token](#)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used to [authenticate to the API over Basic Authentication](#).

Generate new token (Beta)  
Fine-grained, repo-scoped

Generate new token (classic)  
For general use

生成 token 时仅需按下图设置，其余各项保持默认：

## 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

WYX

What's this token for?

### Expiration \*

No expiration ▾

The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

### Select scopes

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

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

生成的 token 只会显示这一次，之后不能查看，故需记录下来以备使用：

## 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\_I9JMjgZ

ibLyXpP3U5P7k

Delete

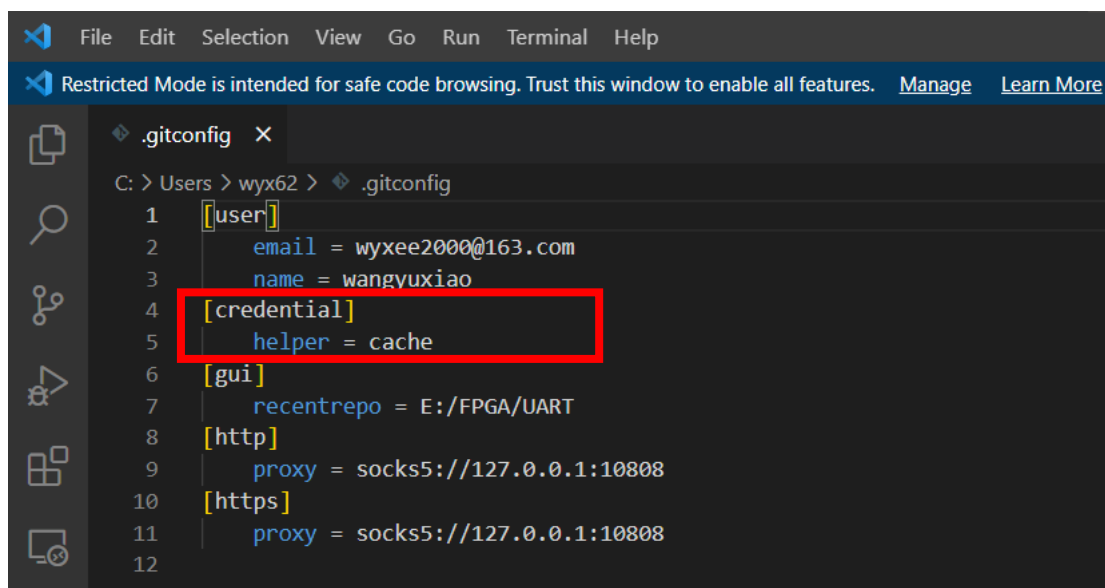
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

#### 4、初次修改 Git 配置文件：

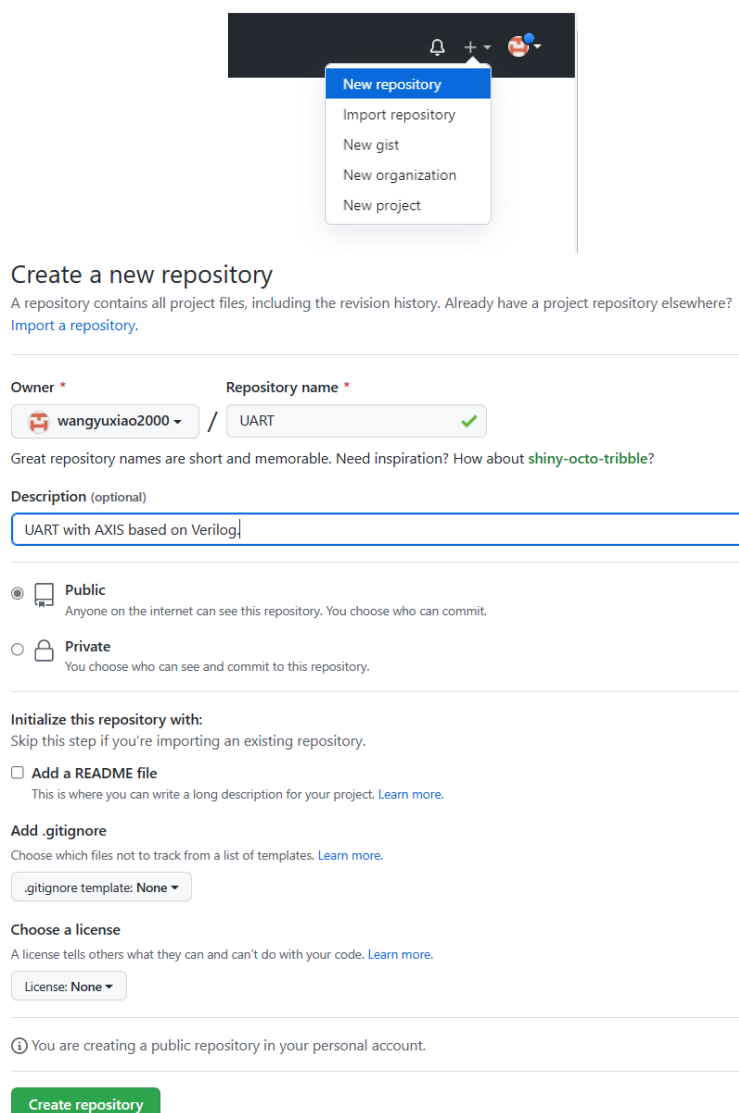
在"Git Bash"中输入命令 `git config -l --show-origin`，找到含“credential.helper=cache”配置选项的配置文件（如下图红框所示）：

```
wyx62@DESKTOP-FSVG0JT MINGW64 ~/Desktop
$ git config -l --show-origin
file:D:/Git/etc/gitconfig diff.astextplain.textconv=astextplain
file:D:/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f
file:D:/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:D:/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process
file:D:/Git/etc/gitconfig filter.lfs.required=true
file:D:/Git/etc/gitconfig http.sslbackend=openssl
file:D:/Git/etc/gitconfig http.sslcainfo=D:/Git/mingw64/ssl/certs/ca-bundle.crt
file:D:/Git/etc/gitconfig core.autocrlf=true
file:D:/Git/etc/gitconfig core.fscache=true
file:D:/Git/etc/gitconfig core.symlinks=false
file:D:/Git/etc/gitconfig pull.rebase=false
file:D:/Git/etc/gitconfig credential.helper=manager
file:D:/Git/etc/gitconfig credential.https://dev.azure.com.usehttppath=true
file:D:/Git/etc/gitconfig init.defaultbranch=master
file:C:/Users/wyx62/.gitconfig user.email=wyxee2000@163.com
file:C:/Users/wyx62/.gitconfig user.name=wangyuxiao
file:C:/Users/wyx62/.gitconfig credential.helper=cache
file:C:/Users/wyx62/.gitconfig gui.recentrepo=E:/FPGA/UART
file:C:/Users/wyx62/.gitconfig http.proxy=socks5://127.0.0.1:10808
file:C:/Users/wyx62/.gitconfig https.proxy=socks5://127.0.0.1:10808
```

打开此配置文件，删除红框中的两行，保存并退出：



## 5、在 GitHub 建立远程仓库：



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* wangyuxiao2000 / Repository name \* UART ✓

Great repository names are short and memorable. Need inspiration? How about [shiny-octo-tribble](#)?

Description (optional)  
UART with AXIS based on Verilog

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)  
.gitignore template: None

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)  
License: None

① You are creating a public repository in your personal account.

**Create repository**

## 6、将本地仓库与 GitHub 上的远程仓库相关联：

在本地仓库的文件夹内单击鼠标右键打开"Git Bash"（这样可以使 Git 软件的执行路径指向这一仓库，除此方法外，也可以使用 cd 命令手动进入相应路径；下图红框中的路径为当前 Git 软件的执行路径），输入以下命令，建立本地仓库与远程仓库间的 SSH 连接：

**git remote add origin 网址(Github 远程仓库的 SSH 地址)**

```
MINGW64:/e/FPGA/UART
wyx62@DESKTOP-FSVG0JT MINGW64 /e/FPGA/UART (master)
$ git remote add origin git@github.com:wangyuxiao2000/UART.git
```

输入命令 **git remote -v** 可以查看本地仓库与远程仓库的关联情况：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/FPGA/UART (master)
$ git remote -v
origin git@github.com:wangyuxiao2000/UART.git (fetch)
origin git@github.com:wangyuxiao2000/UART.git (push)
```

## 7、将本地仓库推送至 GitHub 上的远程仓库：

在 UART 文件夹下打开"Git Bash"界面，输入以下命令：

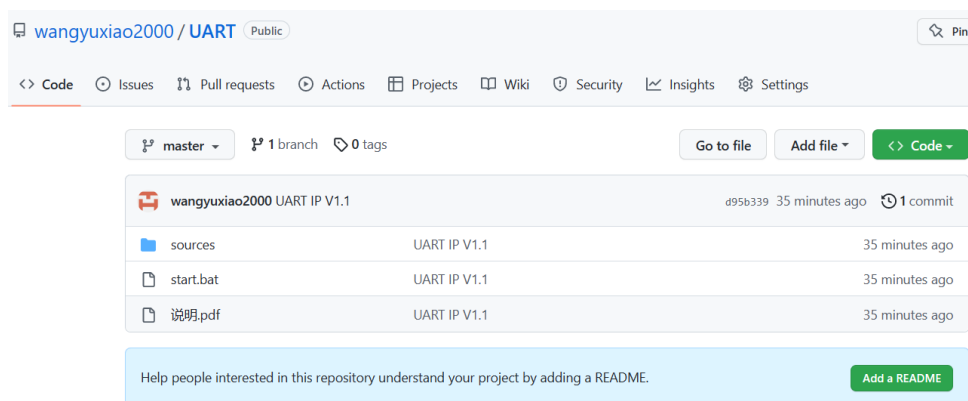
**git push origin master**

```
MINGW64:/e/FPGA/UART

wyx62@DESKTOP-FSVG0JT MINGW64 /e/FPGA/UART (master)
$ git push origin master

wyx62@DESKTOP-FSVG0JT MINGW64 /e/FPGA/UART (master)
$ git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (17/17), 212.00 KiB | 798.00 KiB/s, done.
Total 17 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
```

传输完成后，项目出现在 GitHub 中的远程仓库中，至此，成功完成所有操作：



## 8、克隆远程仓库到本地仓库，修改后再推送至自己的远程仓库：

在 GitHub 上有很多开源代码，我们可以克隆其他人的仓库到本机上，按需修改后，再传入自己的远程仓库中。

首先，按照 5 中的方式，在 GitHub 新建一个空仓库，用于演示：

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner \*  / Repository name \*

Great repository names are short test is available. Need inspiration? How about [fuzzy-pancake?](#)

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.



之后，在"Git Bash"界面，输入以下命令，将某个远程仓库克隆到本地：

**git clone 网址(Github 远程仓库的 SSH 地址)**

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e
$ git clone git@github.com:wangyuxiao2000/UART.git
```

克隆完成后，在本地出现了相应文件夹，文件夹中的内容即为远程仓库中的内容：

本地磁盘 (E:) >

名称	修改日期	类型	大小
UART	2022/12/27 11:01	文件夹	

> 本地磁盘 (E:) > UART >

名称	修改日期	类型	大小
.git	2022/12/27 11:01	文件夹	
sources	2022/12/27 11:01	文件夹	
start.bat	2022/12/27 11:01	Windows 批处理...	3 KB
说明.pdf	2022/12/27 11:01	Microsoft Edge ...	227 KB

对文件夹中的内容按需修改（这里我随便添加了一个 txt）：

本地磁盘 (E:) > UART >				
名称	修改日期	类型	大小	
.git	2022/12/27 11:01	文件夹		
sources	2022/12/27 11:01	文件夹		
A.txt	2022/12/27 11:03	文本文档	1 KB	
start.bat	2022/12/27 11:01	Windows 批处理...	3 KB	
说明.pdf	2022/12/27 11:01	Microsoft Edge ...	227 KB	

之后，可以通过 Git 的 GUI 界面打开这个本地仓库，按照第一部分中介绍的操作方法将文件的改动载入本地仓库中；也可按照如下的命令行操作方式达到这一目的（每次修改项目文件后需要重新提交时，也推荐使用以下命令行操作方式）。

输入命令 **git status**，扫描得到相对于上次的存储内容而言发生改动的文件，相当于 GUI 界面的"Rescan"按钮功能：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  A.txt

nothing added to commit but untracked files present (use "git add" to track)
```

图中标红的文件是发生改动的文件。

输入命令 **git add .**，将发生改动的文件移入缓存区，相当于 GUI 界面的"Stage Changed"按钮功能：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git add .
```

再次输入命令 **git status**，可以看到，发生改动的文件已被移入缓存区：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   A.txt
```

输入命令 **git commit -m “项目描述”**，将缓存区内的文件提交至本地仓库：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git commit -m "test"
[master 19fec22] test
1 file changed, 1 insertion(+)
create mode 100644 A.txt
```

再次输入命令 **git status**，可以看到，此时所有的改动文件已被提交：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

至此，对本地仓库的操作完成，可以将本地仓库提交至远程仓库中。但需要注意的是，在第 6 步操作中，我们的本地仓库是自己创建的，在初始阶段没有与任何远程仓库相关联，故可以直接用 **git remote add origin SSH 地址** 的命令将其关联到远程仓库；而当我们 clone 已有的远程仓库到本地后，本地仓库与远程仓库的关联关系已经存在，此时若直接使用上述命令，会出现如下报错：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git remote add origin git@github.com:wangyuxiao2000/test.git
error: remote origin already exists.
```

输入 **git remote -v** 命令，可见此时已存在图中的关联关系：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git remote -v
origin git@github.com:wangyuxiao2000/UART.git (fetch)
origin git@github.com:wangyuxiao2000/UART.git (push)
```

我们需要先使用 **git remote rm origin** 命令删除这一关联关系后，才能将本地仓库关联到我们自己的远程仓库上：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git remote rm origin
```

再次输入 **git remote -v** 命令，可见此时已不存在关联关系：

```
wyx62@DESKTOP-FSVG0JT MINGW64 /e/UART (master)
$ git remote -v
```

此时再次使用 `git remote add origin SSH 地址命令`，即可实现本地仓库与我们之前新建立的远程仓库间的关联：

```
wyx62@DESKTOP-FSVGOJT MINGW64 /e/UART (master)
$ git remote add origin git@github.com:wangyuxiao2000/test.git

wyx62@DESKTOP-FSVGOJT MINGW64 /e/UART (master)
$ git remote -v
origin  git@github.com:wangyuxiao2000/test.git (fetch)
origin  git@github.com:wangyuxiao2000/test.git (push)
```

使用 `git push origin master` 命令，将本地仓库推送到远程仓库中：

```
wyx62@DESKTOP-FSVGOJT MINGW64 /e/UART (master)
$ git push origin master
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 12 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (20/20), 212.30 KiB | 1.20 MiB/s, done.
Total 20 (delta 2), reused 17 (delta 2), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To github.com:wangyuxiao2000/test.git
 * [new branch]      master -> master
```

wangyuxiao2000 / test Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file <> Code

wangyuxiao2000 test		19fec22 46 minutes ago 2 commits
sources	UART IP V1.1	3 hours ago
A.txt	test	46 minutes ago
start.bat	UART IP V1.1	3 hours ago
说明.pdf	UART IP V1.1	3 hours ago