

目录

目录	1
基于小智 AI 的 AI 语音助手	2
项目介绍	2
注意事项	2
小智 AI 使用声明	3
ESP32 S3 硬件说明	4
Freenove ESP32-S3 WROOM Board	4
音频电路板	5
安装 CH343 (Importance)	6
小智 AI 固件	17
安装 Python (Importance)	17
上传固件	24
ESP32 S3 WROOM 配网	32
配置小智 AI 服务器	34
小智 AI 代码	41
Visual Studio Code	41
Window	41
MAC	44
Linux	45
安装 ESP-IDF V5.3.2	48
下载代码	54
配置代码环境	56
编译代码	64
本地服务器	66
小智 AI 服务器使用声明	66
小智 AI 本地服务器部署	66
安装 Ollama	66
安装 Conda	80
部署虚拟环境	95
部署 xiaozhi-esp32-server 服务器	97
ESP32S3 访问 xiaozhi-esp-server 服务器	107

基于小智 AI 的 AI 语音助手

本项目使用 Media Kit 实现一个 AI 语音助手，需要有一定的编程基础，且对于 ESP-IDF 和开源大模型有一定的了解。

项目介绍

本语音助手项目 (<https://github.com/Freenove/xiaozhi-esp32>) 来自开源项目 (<https://github.com/78/xiaozhi-esp32>)，该项目可以在嵌入式设备上调用市面上大多数长语言文本模型 (LLM)，并通过语音活动检测 (VAD) 服务、语音识别(ASR)服务、语音转文本 (STT) 服务、文本转语音 (TTS 服务)、记忆存储 (Memory) 服务、意图识别 (Intent) 服务等多种服务，来实现语音对话功能，制作属于你自己的 AI 语音助手。Freenove 将该项目适配到了产品 Media Kit 上。本文将介绍如何在 Media Kit 上运行该项目。

本项目有两种运行方式，

1. 在线版本，接入 xiaozhi.me 服务器，目前个人用户可以免费试用。
2. 离线版本，以上提到的各项服务都需要在本地个人电脑上搭建本地服务器，使用体验完全取决于模型的选择和本地电脑性能。本地服务器项目 (<https://github.com/Freenove/xiaozhi-esp32-server>) 来自开源项目 (<https://github.com/xinnan-tech/xiaozhi-esp32-server>)。

对于偏向于使用 AI 助手的用户，我们建议你使用在线版本。

对于偏向于开发者用户，可以尝试部署离线版本，以深入学习和了解 AI 助手所需要的各项服务。值得注意的是，个人电脑难以同时运行以上各种服务，尤其是核心服务 LLM，这可能导致你的 AI 助手体检较差，因此，离线版本仅具有更多的学习研究价值。

注意事项

● 项目版权：

- 语音助手项目原始作者署名为“虾哥”，Freenove 公司 fork 该项目并适配到 Media Kit 中，该项目遵守 MIT 协议。
- 本地服务器项目原始作者为“xinnan-tech”，Freenove 公司 fork 该项目并适配到 Media Kit 中，该项目遵守 MIT 协议。

● 支持的国家或地区：

- 在线版本，这取决于 xiaozhi.me 服务器支持的国家或地区，并非所有国家或地区都受到支持。您可以注册页面查看 (<https://xiaozhi.me/login?redirect=/console/agents>)。对于使用体验，也受 xiaozhi.me 服务器的影响。如果您访问 xiaozhi.me 服务器的网络较差，则体验相对就差一些。
- 离线版本，无，您可以在任意国家或地区部署离线版本。

● 支持的语言：

- 在线版本：目前支持中文普通话、中文粤语、英语、日语、韩语等。如果您不会这些语言，那么使用在线版本您将无法与小智 AI 正常交流。
- 离线版本：取决于您部署的 ASR 模型。默认的 FunASR 模型仅支持中文普通话、中文粤语、英语、日语、韩语。

● 费用

- 在线版本：当前 xiaozhi.me 提供免费服务，我们不能保证在线服务器一直免费。
- 离线版本：上述提到的各项子服务中，均有收费服务或免费服务，取决于您自己的选择。
- 遇到问题，如果您按照教程操作，仍然无法使用，请与我们联系（support.freenove.com）。由于在线服务由 xiaozhi.me 提供，如果 xiaozhi.me 关闭了服务，那么我们也将关闭相关的文档、教程、代码。

小智 AI 使用声明

我们仅在 <https://github.com/78/xiaozhi-esp32> 基础上提供适配，用做第三方学习和 AI 功能试用，不做商业性质推广和应用。本教程仅给爱好者附加学习使用。

请注意：

1, 由于这个项目是第三方开源项目，如果您在学习过程中遇见问题，请向开源项目发起 issue：

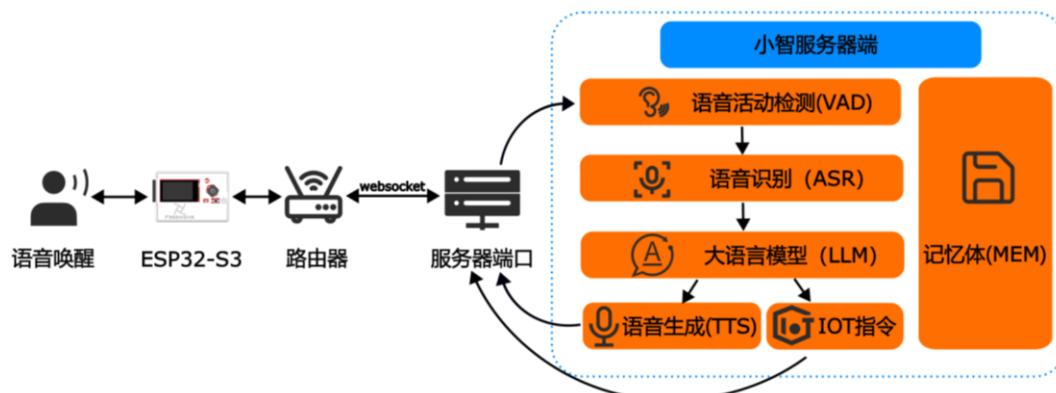
<https://github.com/78/xiaozhi-esp32/issues>

2, 目前小智 AI 仅支持普通话，粤语，英语，韩语，日语这 5 种语言识别。其他语言暂时仍不支持。

3, 目前小智服务器的界面仅支持英语、中文、日语。且小智服务器目前仅支持下列国家使用手机注册账号。
其他国家暂不支持。

Mainland China +86	Germany +49	Philippines +63	Pakistan +92
Hong Kong +852	Poland +48	New Zealand +64	Nigeria +234
Macau +853	Switzerland +41	Singapore +65	Bangladesh +880
Taiwan +886	Spain +34	Thailand +66	Saudi Arabia +966
United States/Canada +1	Denmark +45	Japan +81	United Arab Emirates +971
United Kingdom +44	Malaysia +60	South Korea +82	Brazil +55
France +33	Australia +61	Vietnam +84	Mexico +52
Italy +39	Indonesia +62	India +91	Chile +56
Argentina +54	Egypt +20	South Africa +27	Kenya +254
Tanzania +255	Kazakhstan +7		

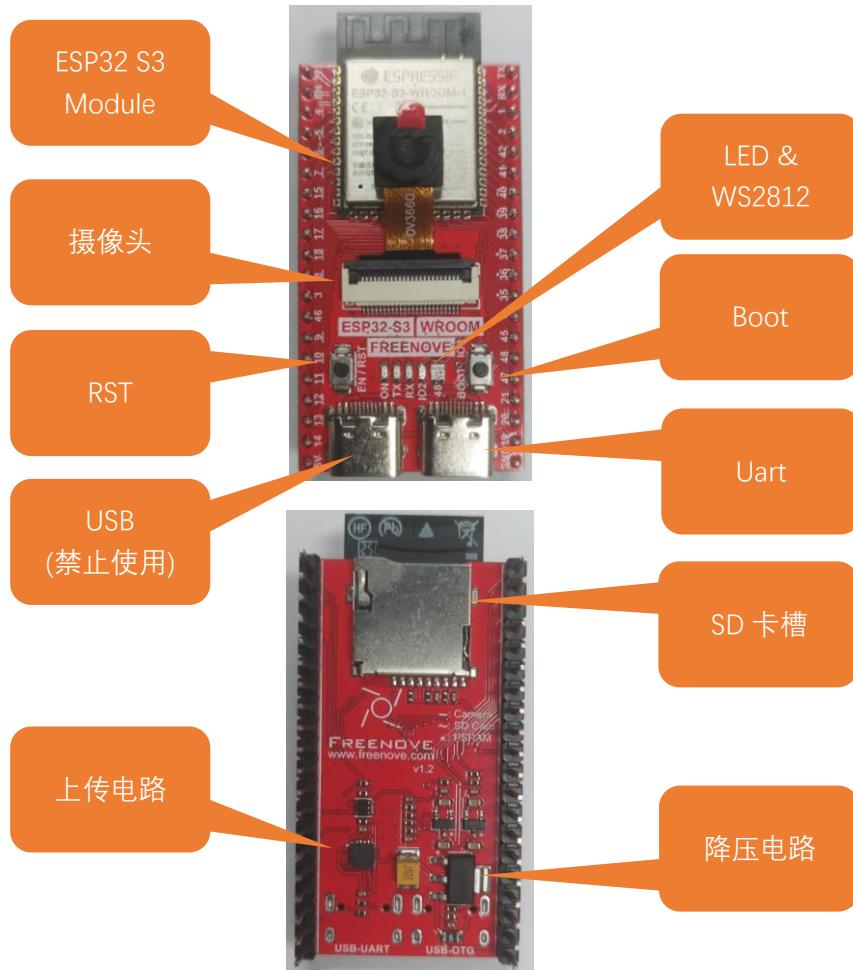
在这个项目中，ESP32-S3 使用 websocket 的方式，和小智 AI 服务器进行数据交互。



ESP32 S3 硬件说明

Freenove ESP32-S3 WROOM Board

首先，我们使用 Freenove ESP32-S3 Board 作为主控板，它集成了 16Mb 的 Flash 和 8Mb 的 Psram。

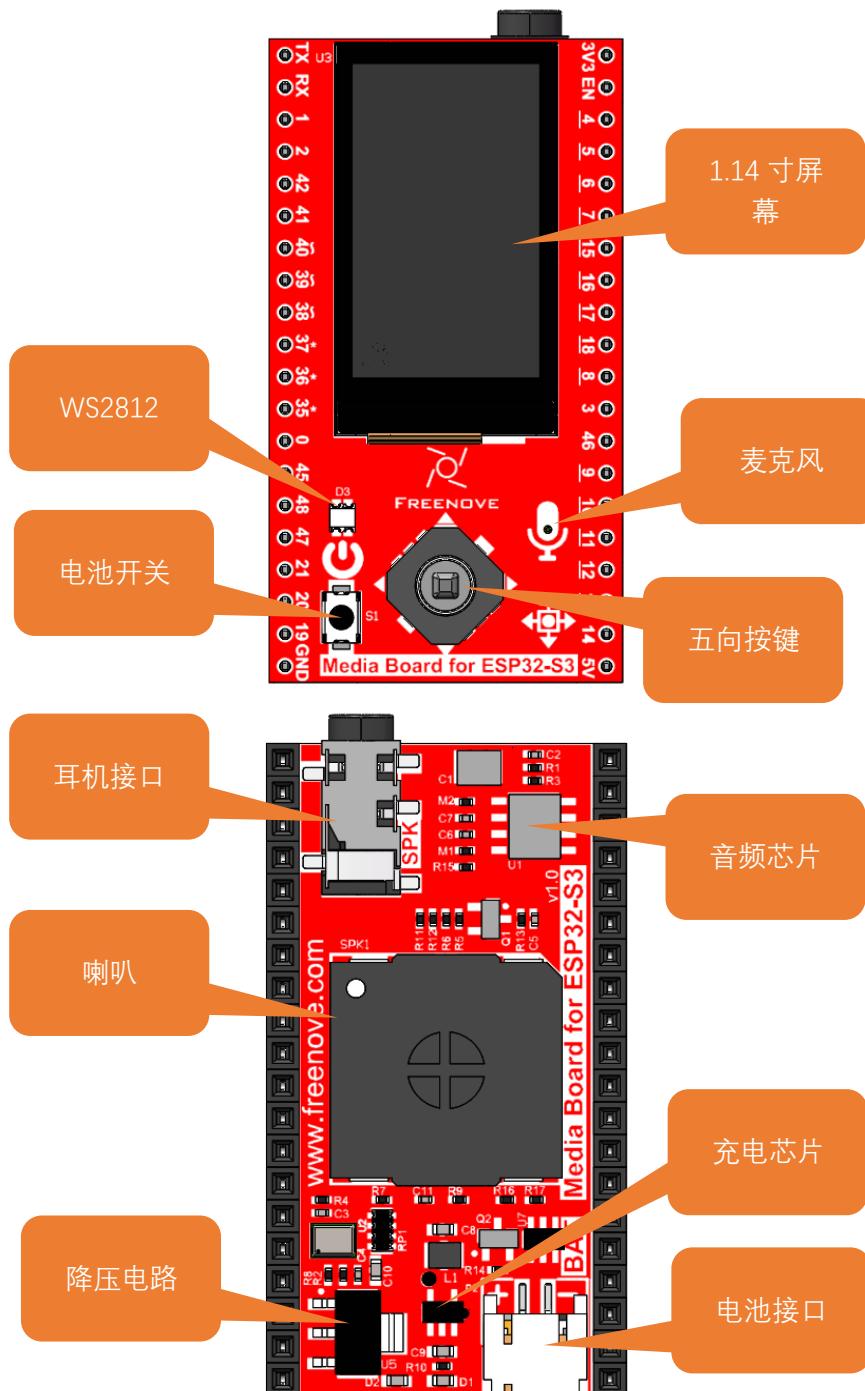


更多关于 Freenove ESP32-S3 Board 的资料，请查看

https://github.com/Freenove/Freenove_Ultimate_Starter_Kit_for_ESP32_S3

音频电路板

音频电路板集成了多个不同功能的硬件，比如 1.14 寸 TFT 显示屏，麦克风，电池充电电路，音频电路，耳机接口和集成喇叭等。如下图所示。



安装 CH343 (Importance)

ESP32-S3 WROOM 使用 CH343 芯片下载代码，因此在使用前，需在电脑上安装 CH343 驱动程序。

Windows

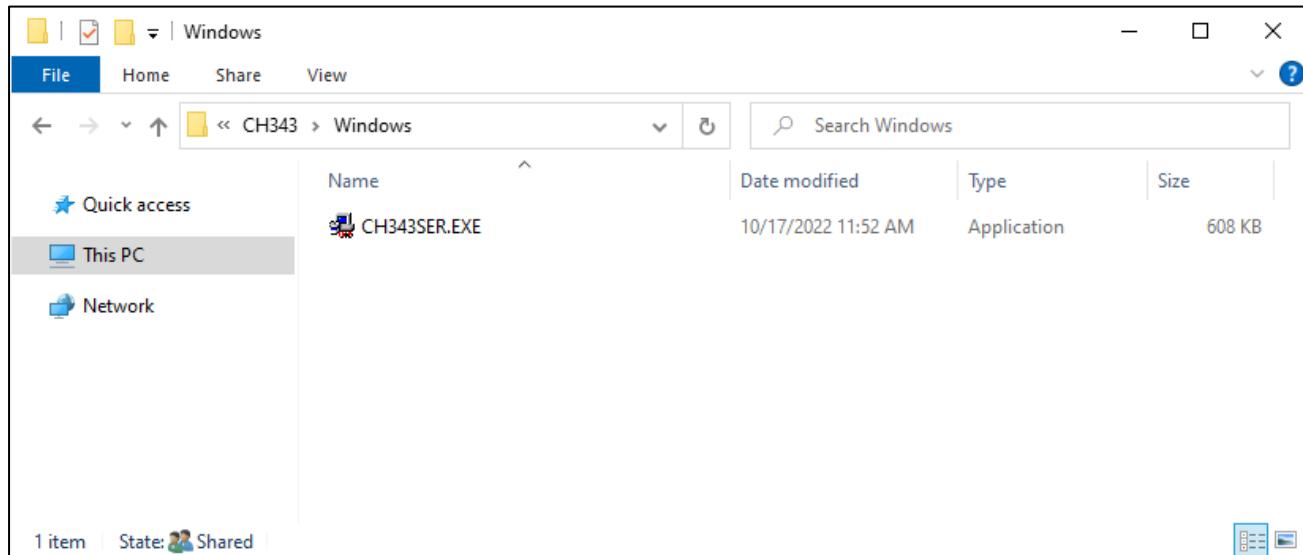
首先下载 CH343 驱动，点击 <http://www.wch-ic.com/search?t=all&q=ch343>，根据您的操作系统选择对应版本进行下载。

keyword ch343				
Downloads(8)				
file category	file content	version	upload time	
DataSheet				
CH343DS1.PDF	CH343 datasheet, USB to single serial port, supports up to 6M baud rate, serial port signals support 5V/3.3V/2.5V/1.8V, built-in crystal oscillator. CH343 supports built-in CDC driver in operating system or multi-functional high-speed VCP manufacture driver.	1.5	2021-11-18	
Driver&Tools				
CH343SER.ZIP	For CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to high-speed serial port VCP vendor driver, supports Windows 11/10/8.1/7/VISTA/XP/2000	1.61	2022-05-13	
CH343CDC.ZIP	For CH342/CH343/CH344/CH347/CH910X/CH9143/CH9340, USB to CDC serial port driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000	1.4	2022-05-13	
CH343SER.EXE	For CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to high-speed serial port VCP vendor driver, supports Windows 11/10/8.1/7/VISTA/XP/2000	1.61	2022-05-13	
CH34XSER_MAC.ZI...	For CH340/CH341/CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to serial port VCP vendor driver of macOS	1.7	2022-05-13	
CH343CDC.EXE	For CH342/CH343/CH344/CH347/CH910X/CH9143/CH9340, USB to CDC serial port driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000	1.4	2022-05-13	
Application				
CH34xSerCfg.ZIP	USB configuration tool of Windows for CH340/CH342/CH343/CH344/CH347/CH348/CH9101/CH9102/CH9103. Via this tool, the chip's Vendor ID, product ID, maximum current value, BCD version	1.2	2022-05-24	

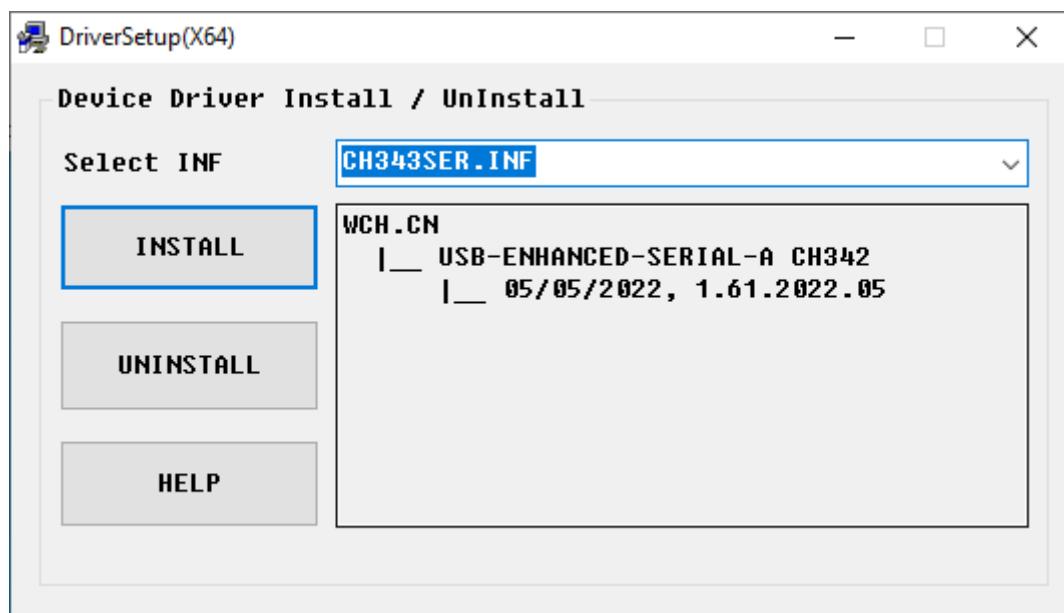
如果您不想从官网下载安装包，可以打开“Freenove_Media_Kit_for_ESP32-S3/CH343”文件夹，我们已为您准备好驱动安装包

Linux	10/17/2022 1:30 PM	File folder
MAC	10/17/2022 1:30 PM	File folder
Windows	10/17/2022 1:30 PM	File folder

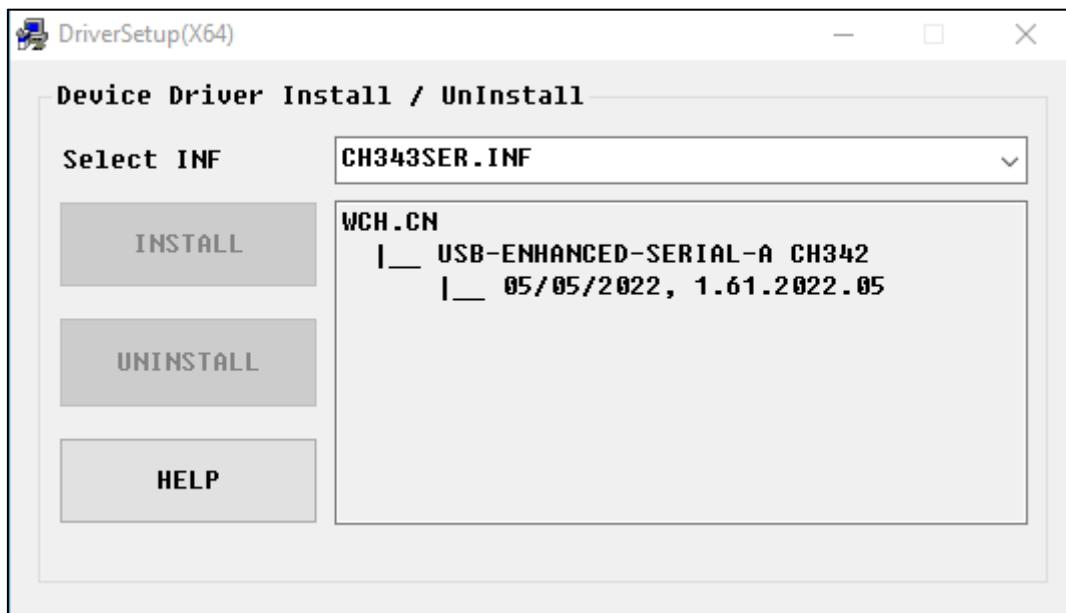
2. 打开文件夹进入目录：“Freenove_Media_Kit_for_ESP32-S3/CH343/Windows/”



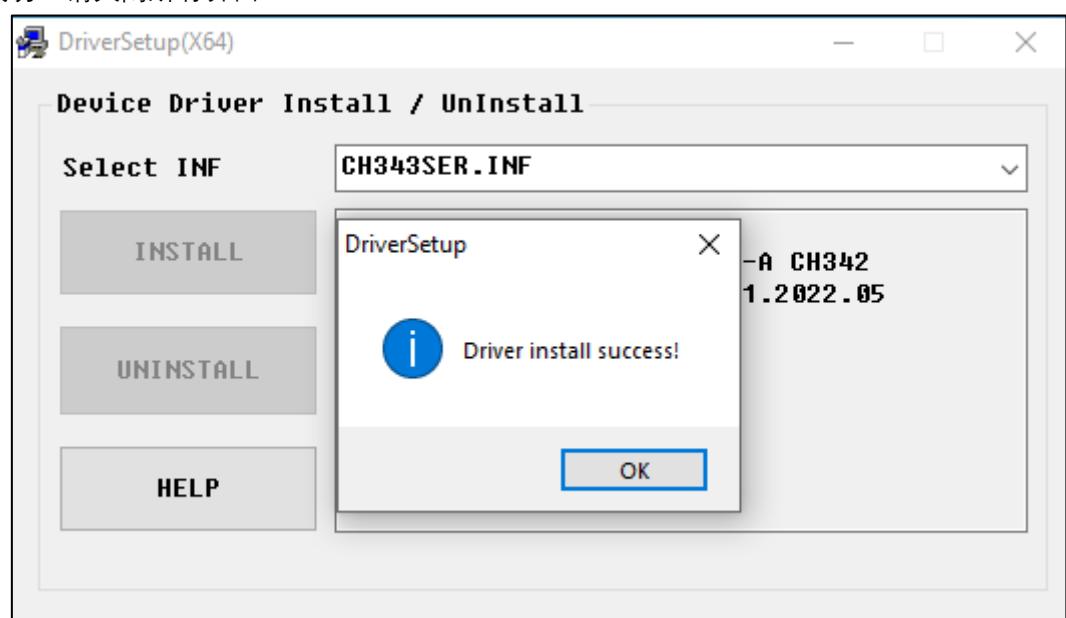
3. 双击运行“CH343SER.EXE”。



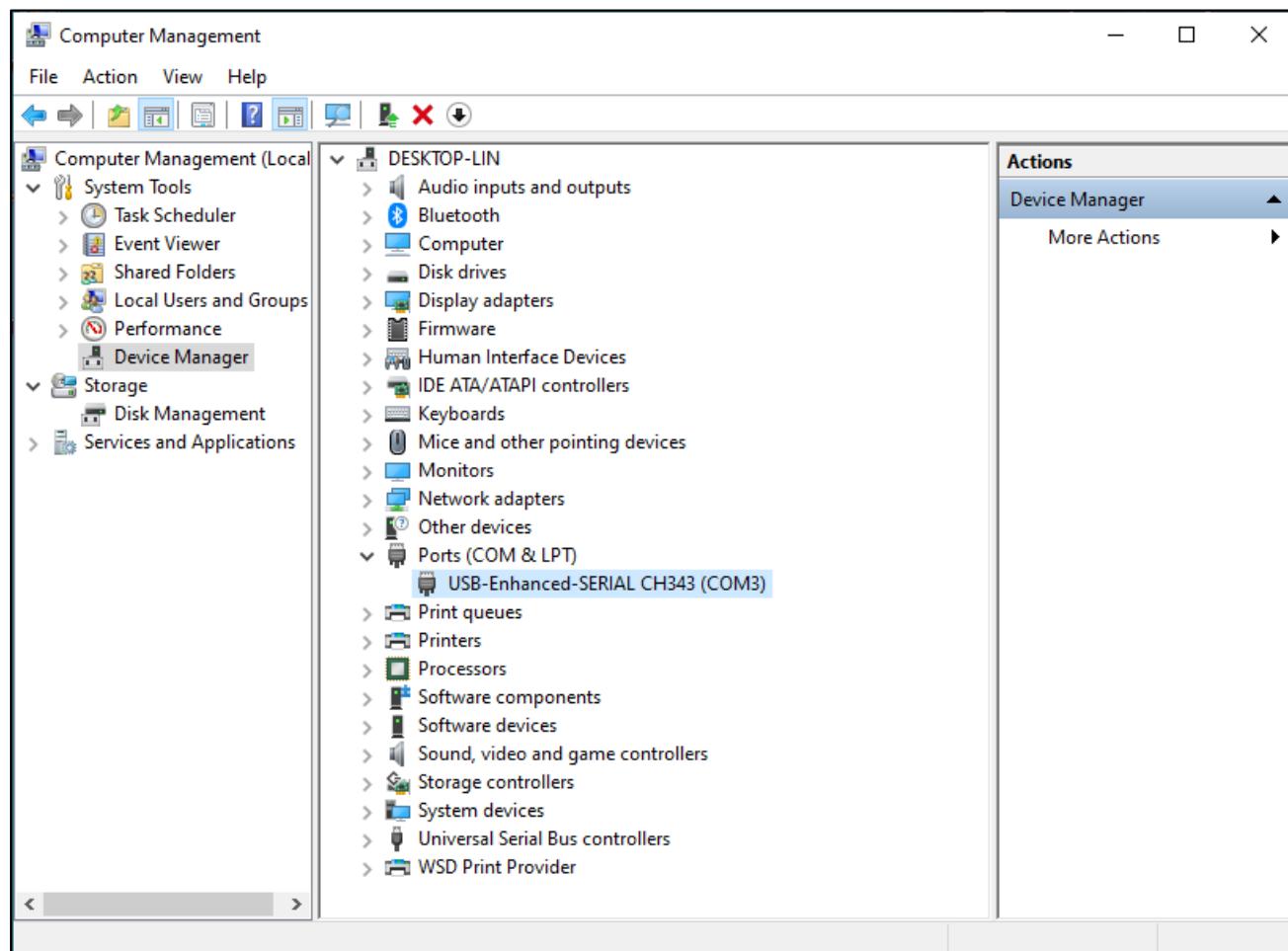
4. 点击 "INSTALL" 并等待安装完成



5. 安装成功！请关闭所有界面



6. 当 ESP32-S3 WROOM 模块连接到电脑时，选中“此电脑”，右键选择“管理”，在弹出的新对话框中点击“设备管理器”，即可看到如下界面。



7. 至此，CH343 驱动已成功安装，关闭所有对话框。

MAC

首先下载 CH343 驱动，点击 <http://www.wch-ic.com/search?t=all&q=ch343>，根据您的操作系统选择对应版本进行下载。

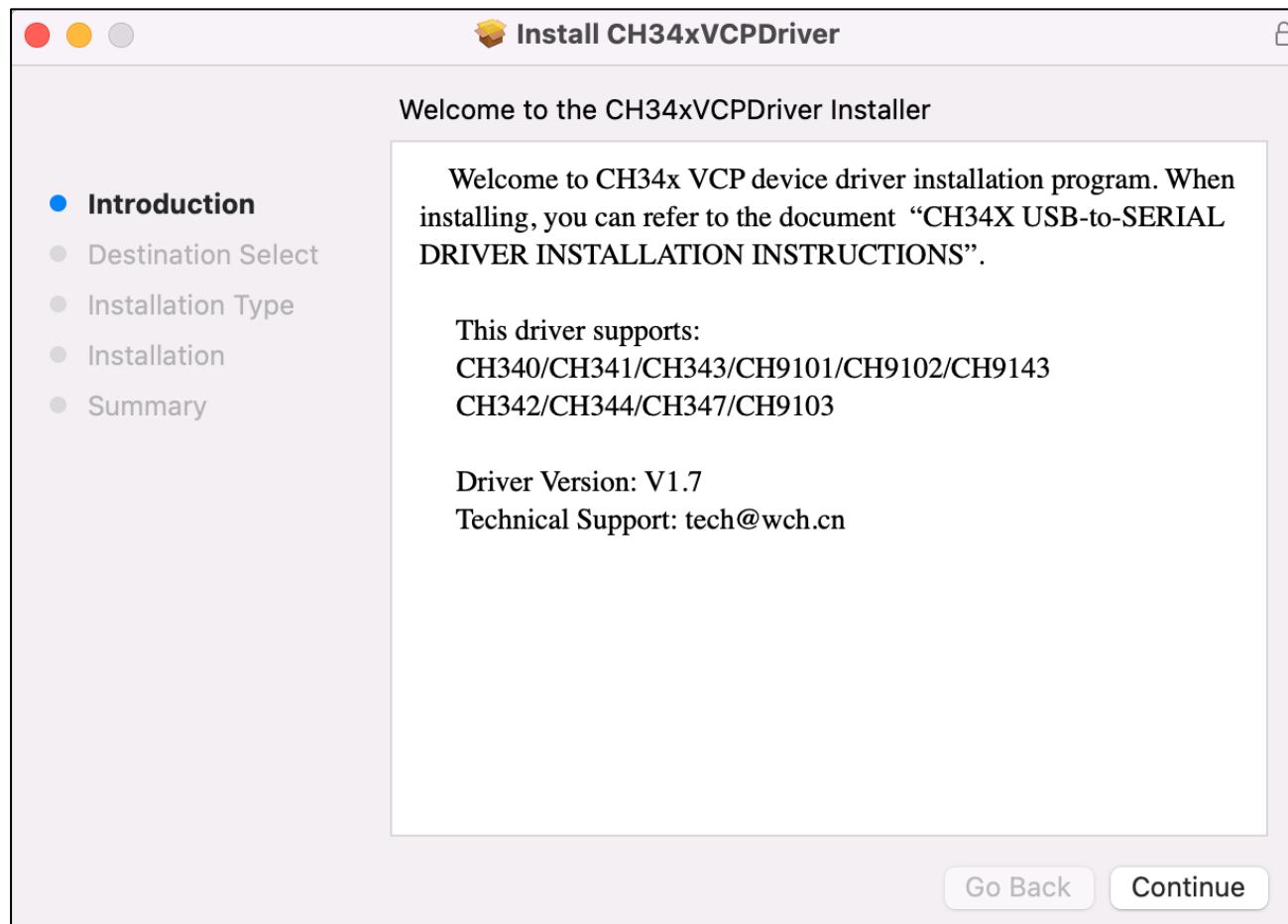
keyword ch343				
Downloads(8)				
file category	file content	version	upload time	
DataSheet				
CH343DS1.PDF	CH343 datasheet, USB to single serial port, supports up to 6M baud rate, serial port signals support 5V/3.3V/2.5V/1.8V, built-in crystal oscillator. CH343 supports built-in CDC driver in operating system or multi-functional high-speed VCP manufacture driver.	1.5	2021-11-18	
Driver&Tools				
CH343SER.ZIP	For CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to high-speed serial port VCP vendor driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000	1.61	2022-05-13	
CH343CDC.ZIP	For CH342/CH343/CH344/CH347/CH910X/CH9143/CH9340, USB to CDC serial port driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000	1.4	2022-05-13	
CH343SER.EXE	For CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to high-speed serial port VCP vendor driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000	1.61	2022-05-13	
CH34XSER_MAC.ZI...	For MAC CH340/CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to serial port VCP vendor driver of macOS	1.7	2022-05-13	
CH343CDC.EXE	For CH342/CH343/CH344/CH347/CH910X/CH9143/CH9340, USB to CDC serial port driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000	1.4	2022-05-13	
Application				
CH34xSerCfg.ZIP	USB configuration tool of Windows for CH340/CH342/CH343/CH344/CH347/CH348/CH9101/CH9102/CH9103. Via this tool, the chip's Vendor ID, product ID, maximum current value, BCD version	1.2	2022-05-24	

如果不想下载安装包，您可以打开“Freenove_Media_Kit_for_ESP32-S3/CH343”文件夹，我们已准备好安装包。

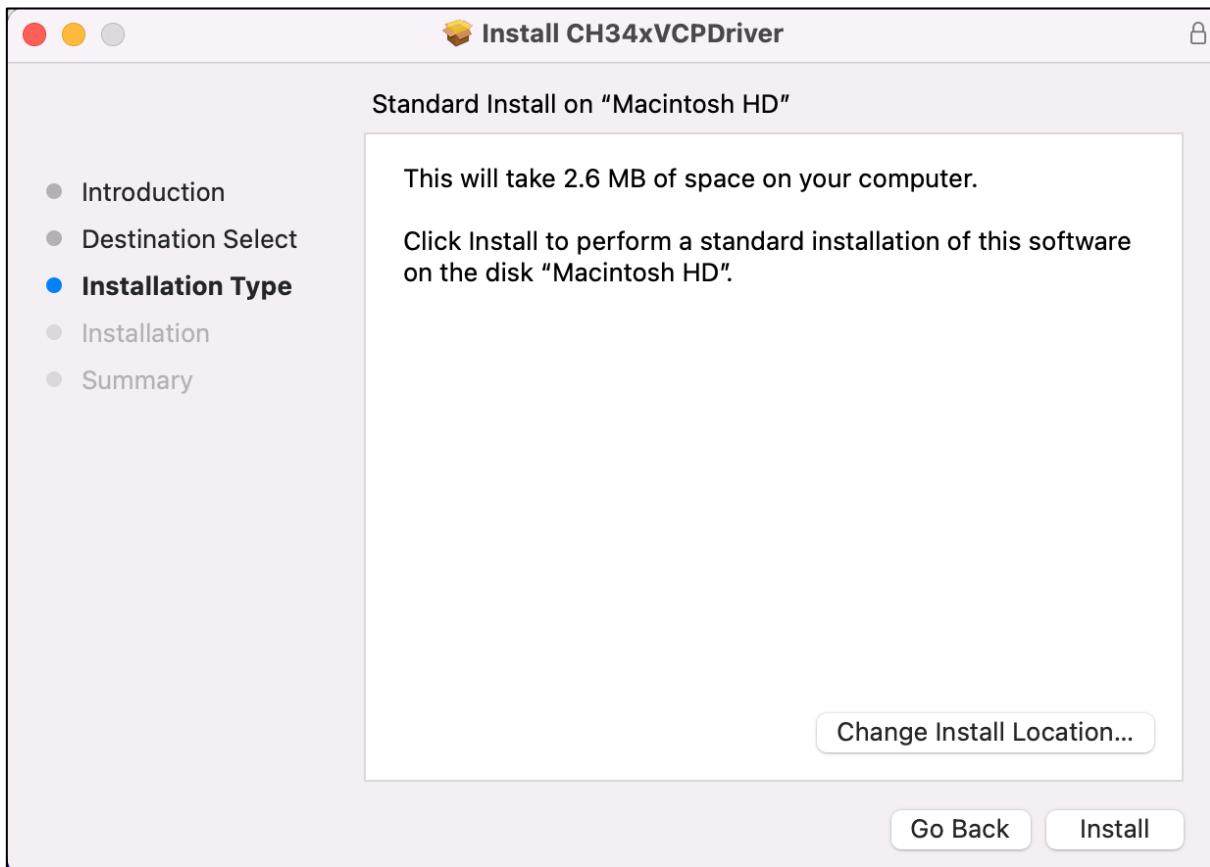
其次，打开“Freenove_Media_Kit_for_ESP32-S3/CH343/MAC/”目录。



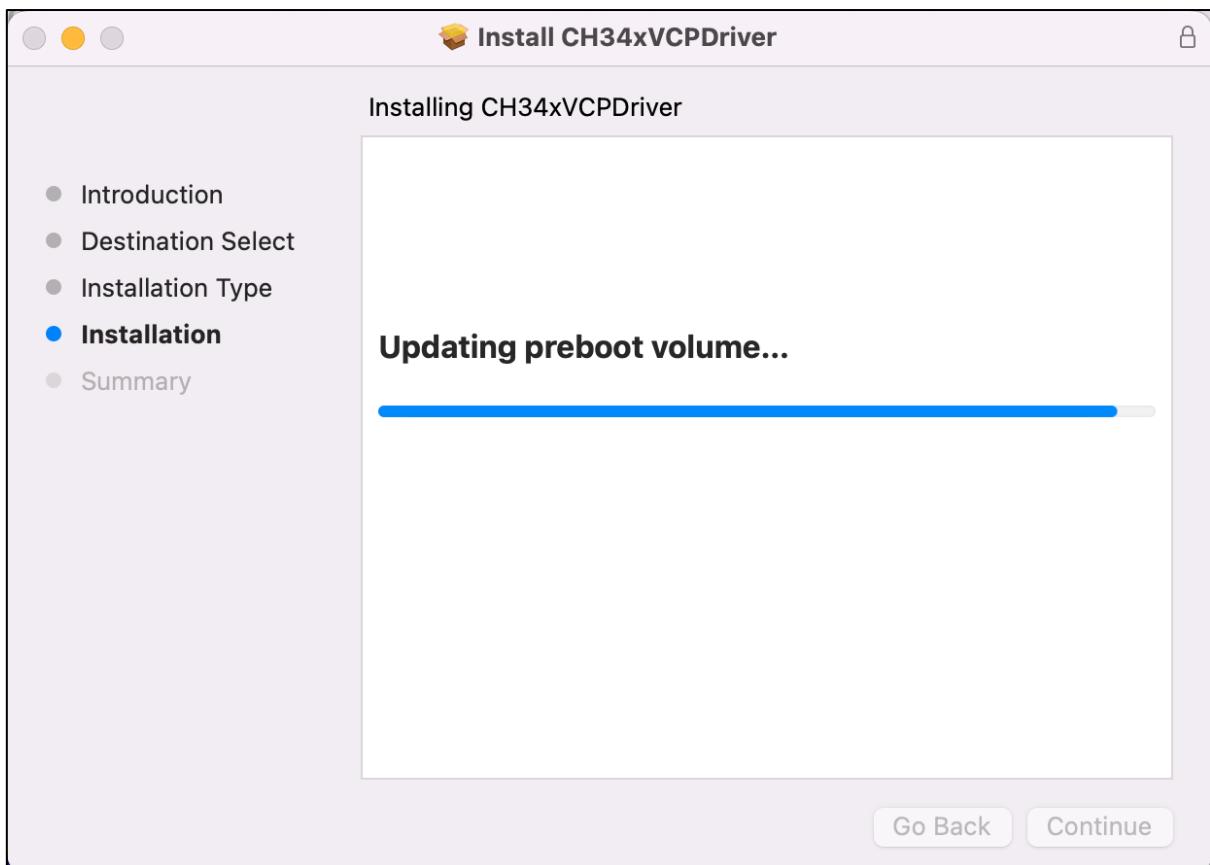
第三步，点击 Continue。



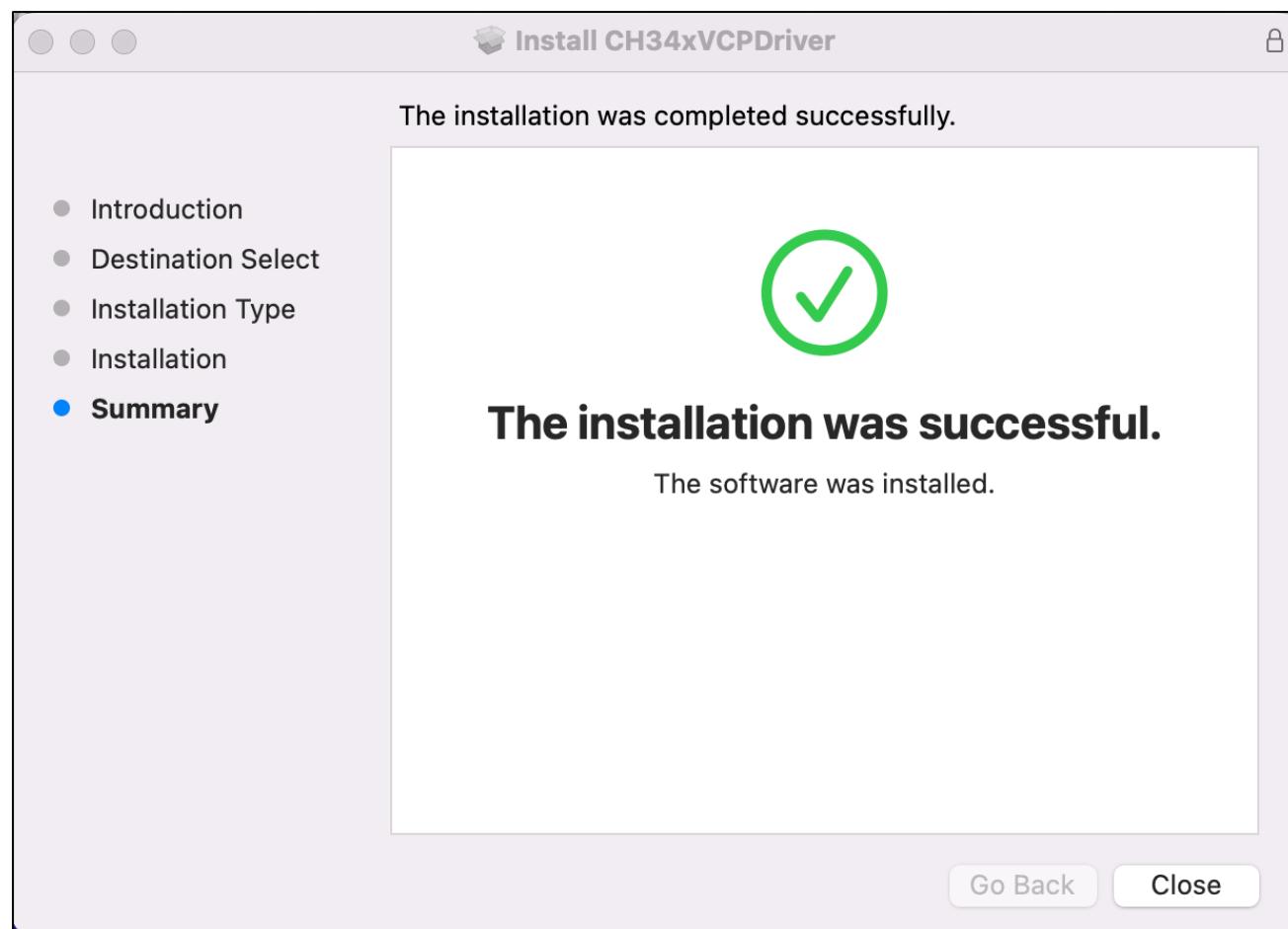
第四步，点击 Install。



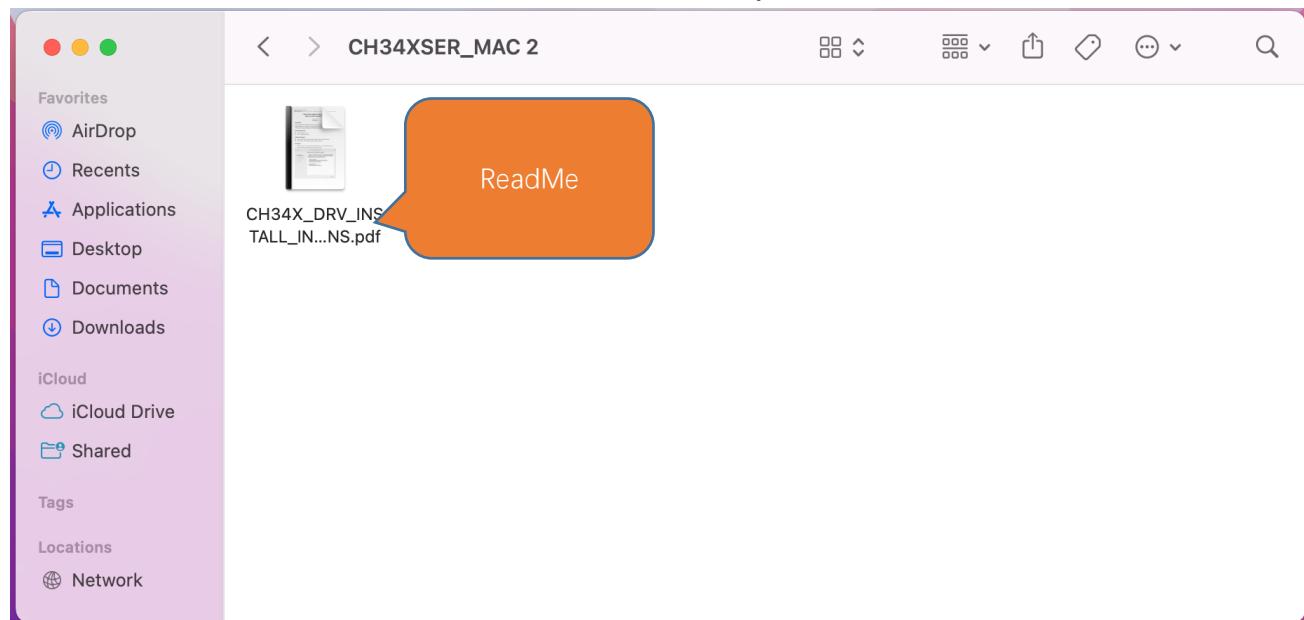
然后等待完成



最后，重新启动你的电脑



若按照上述步骤仍未成功安装 CH340 驱动，可查阅 **readme.pdf** 文件进行安装。



Linux

这里以 Ubuntu 为例。打开 Linux 系统的终端。

```
lin@ubuntu:~$
```

使用指令“lsusb”查看端口。

```
lsusb  
ls /dev/tty*
```

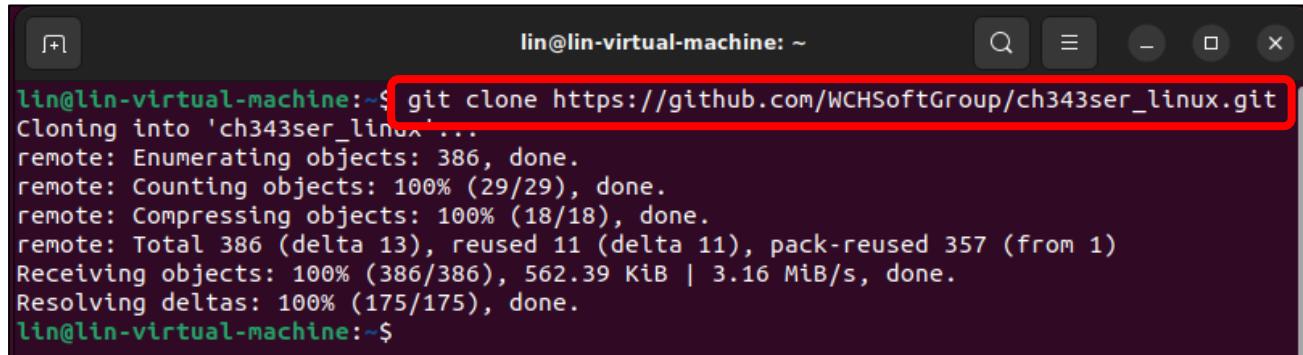
```
lin@lin-virtual-machine:~$ lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 1a86:55d3 QinHeng Electronics USB Single Serial
Bus 001 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

```
lin@lin-virtual-machine:~$ ls /dev/tty*
/dev/tty      /dev/tty23    /dev/tty39    /dev/tty54      /dev/ttyS1      /dev/ttyS25
/dev/tty0     /dev/tty24    /dev/tty4      /dev/tty55      /dev/ttyS10     /dev/ttyS26
/dev/tty1     /dev/tty25    /dev/tty40     /dev/tty56      /dev/ttyS11     /dev/ttyS27
/dev/tty10    /dev/tty26    /dev/tty41     /dev/tty57      /dev/ttyS12     /dev/ttyS28
/dev/tty11    /dev/tty27    /dev/tty42     /dev/tty58      /dev/ttyS13     /dev/ttyS29
/dev/tty12    /dev/tty28    /dev/tty43     /dev/tty59      /dev/ttyS14     /dev/ttyS3
/dev/tty13    /dev/tty29    /dev/tty44     /dev/tty6       /dev/ttyS15     /dev/ttyS30
/dev/tty14    /dev/tty3     /dev/tty45     /dev/tty60      /dev/ttyS16     /dev/ttyS31
/dev/tty15    /dev/tty30    /dev/tty46     /dev/tty61      /dev/ttyS17     /dev/ttyS4
/dev/tty16    /dev/tty31    /dev/tty47     /dev/tty62      /dev/ttyS18     /dev/ttyS5
/dev/tty17    /dev/tty32    /dev/tty48     /dev/tty63      /dev/ttyS19     /dev/ttyS6
/dev/tty18    /dev/tty33    /dev/tty49     /dev/tty7       /dev/ttyS2      /dev/ttyS7
/dev/tty19    /dev/tty34    /dev/tty5      /dev/tty8       /dev/ttyS20     /dev/ttyS8
/dev/tty2     /dev/tty35    /dev/tty50     /dev/tty9       /dev/ttyS21     /dev/ttyS9
/dev/tty20    /dev/tty36    /dev/tty51     /dev/ttyACM0    /dev/ttyS22
/dev/tty21    /dev/tty37    /dev/tty52     /dev/ttyprintk  /dev/ttyS23
/dev/tty22    /dev/tty38    /dev/tty53     /dev/ttyS0      /dev/ttyS24
```

如果您的电脑没有 CH343 驱动，您可以按照接下来的步骤安装它。如果您的电脑能识别到 CH343 驱动，您可以跳过下面的步骤。

使用指令下载 CH343 驱动。

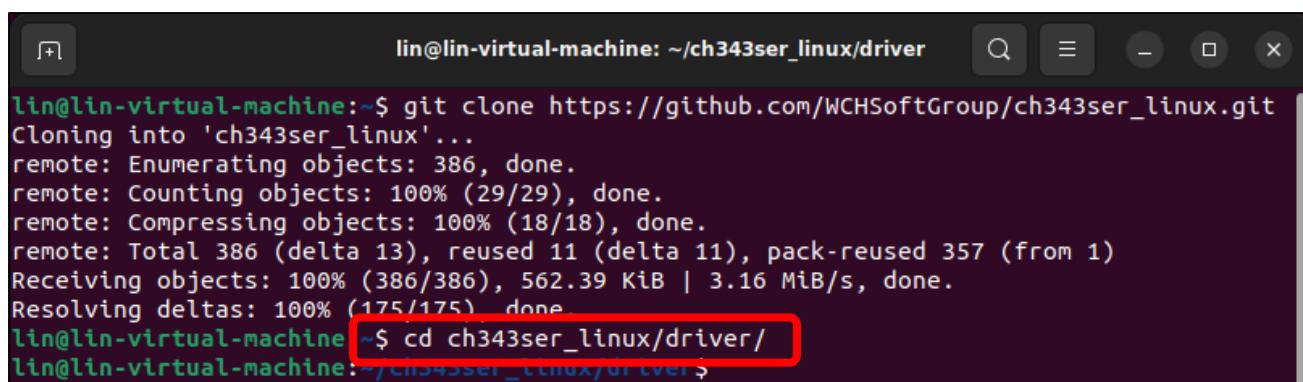
```
git clone https://github.com/WCHSoftGroup/ch343ser_linux.git
```



```
lin@lin-virtual-machine:~$ git clone https://github.com/WCHSoftGroup/ch343ser_linux.git
Cloning into 'ch343ser_linux'...
remote: Enumerating objects: 386, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 386 (delta 13), reused 11 (delta 11), pack-reused 357 (from 1)
Receiving objects: 100% (386/386), 562.39 KiB | 3.16 MiB/s, done.
Resolving deltas: 100% (175/175), done.
lin@lin-virtual-machine:~$
```

进入 ch343 驱动文件夹。

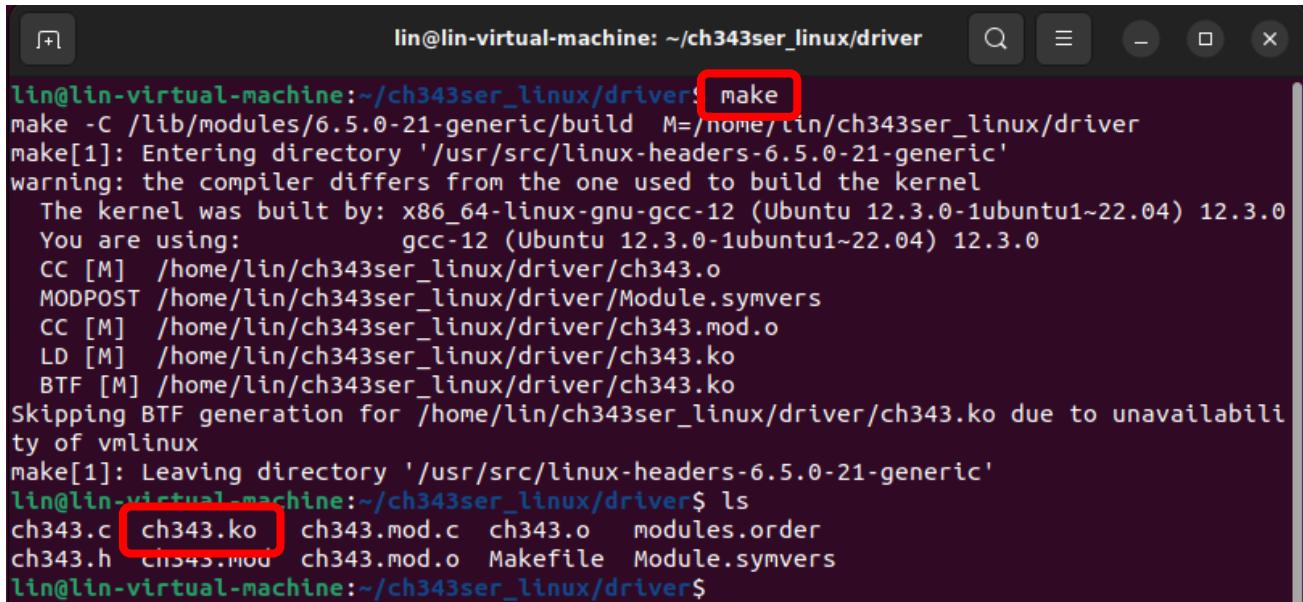
```
cd ch343ser_linux/driver/
```



```
lin@lin-virtual-machine:~$ git clone https://github.com/WCHSoftGroup/ch343ser_linux.git
Cloning into 'ch343ser_linux'...
remote: Enumerating objects: 386, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 386 (delta 13), reused 11 (delta 11), pack-reused 357 (from 1)
Receiving objects: 100% (386/386), 562.39 KiB | 3.16 MiB/s, done.
Resolving deltas: 100% (175/175), done.
lin@lin-virtual-machine:~/ch343ser_linux/driver$ cd ch343ser_linux/driver/
lin@lin-virtual-machine:~/ch343ser_linux/driver$
```

编译并生成 ch343.ko 文件。

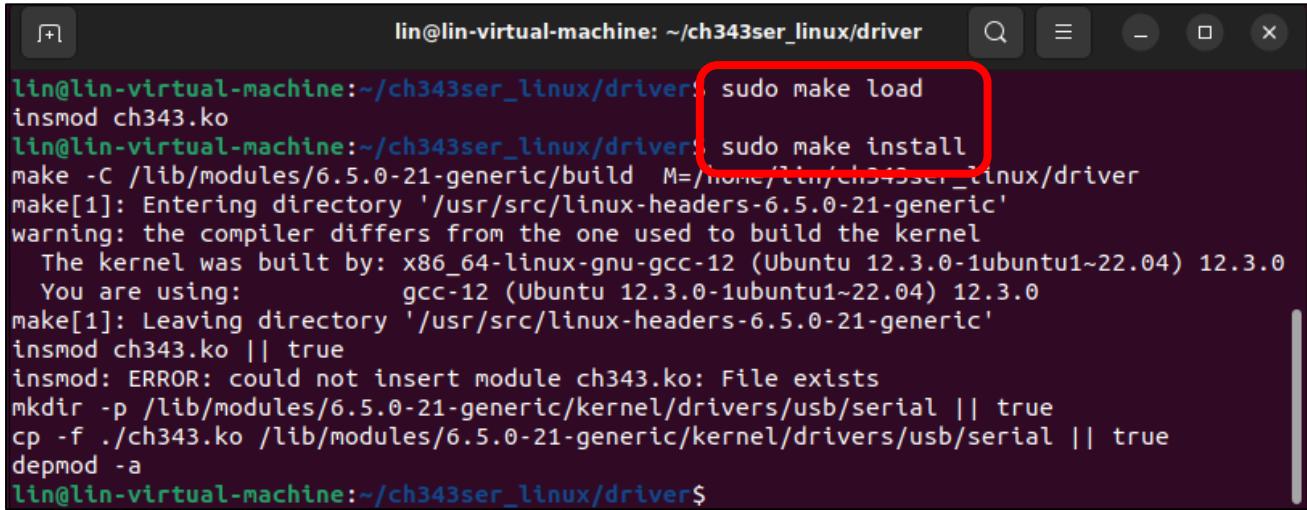
```
make
```



```
lin@lin-virtual-machine:~/ch343ser_linux/driver$ make
make -C /lib/modules/6.5.0-21-generic/build M=/home/lin/ch343ser_linux/driver
make[1]: Entering directory '/usr/src/linux-headers-6.5.0-21-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:           gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
CC [M]  /home/lin/ch343ser_linux/driver/ch343.o
MODPOST /home/lin/ch343ser_linux/driver/Module.symvers
CC [M]  /home/lin/ch343ser_linux/driver/ch343.mod.o
LD [M]  /home/lin/ch343ser_linux/driver/ch343.ko
BTF [M] /home/lin/ch343ser_linux/driver/ch343.ko
Skipping BTF generation for /home/lin/ch343ser_linux/driver/ch343.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.5.0-21-generic'
lin@lin-virtual-machine:~/ch343ser_linux/driver$ ls
ch343.c  ch343.ko  ch343.mod.c  ch343.o  modules.order
ch343.h  ch343.mod  ch343.mod.o  Makefile  Module.symvers
lin@lin-virtual-machine:~/ch343ser_linux/driver$
```

安装 ch343p 芯片的驱动。

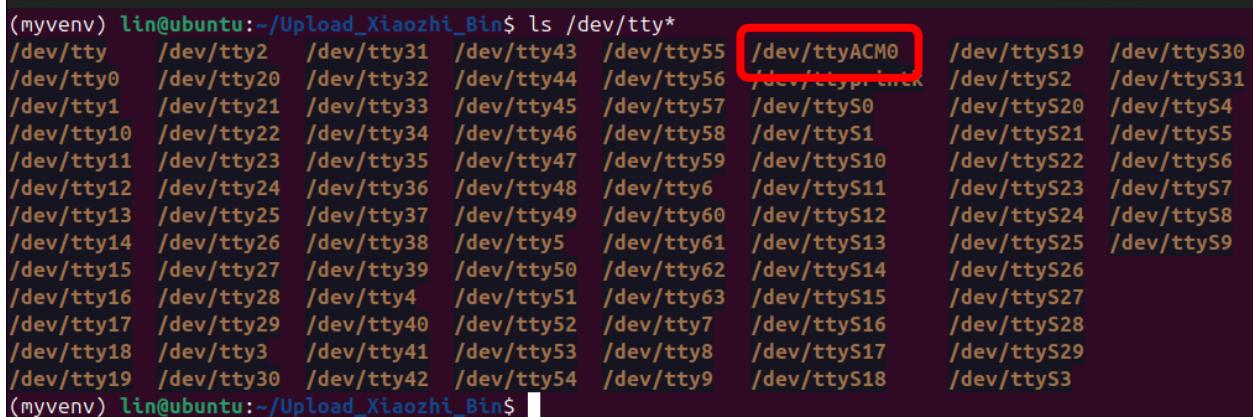
```
sudo make load
sudo make install
```



```
lin@lin-virtual-machine:~/ch343ser_linux/driver$ sudo make load
insmod ch343.ko
lin@lin-virtual-machine:~/ch343ser_linux/driver$ sudo make install
make -C /lib/modules/6.5.0-21-generic/build M=/home/lin/ch343ser_linux/driver
make[1]: Entering directory '/usr/src/linux-headers-6.5.0-21-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:          gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
make[1]: Leaving directory '/usr/src/linux-headers-6.5.0-21-generic'
insmod ch343.ko || true
insmod: ERROR: could not insert module ch343.ko: File exists
mkdir -p /lib/modules/6.5.0-21-generic/kernel/drivers/usb/serial || true
cp -f ./ch343.ko /lib/modules/6.5.0-21-generic/kernel/drivers/usb/serial || true
depmod -a
lin@lin-virtual-machine:~/ch343ser_linux/driver$
```

当将 ESP32S3 连接到电脑时，使用指令后的显示内容如下图所示。

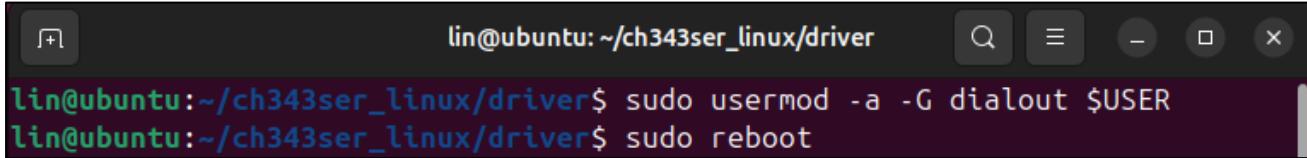
```
ls /dev/tty*
```



```
(myenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$ ls /dev/tty*
/dev/tty  /dev/tty2  /dev/tty31  /dev/tty43  /dev/tty55  /dev/ttyACM0  /dev/ttyS19  /dev/ttyS30
/dev/tty0  /dev/tty20  /dev/tty32  /dev/tty44  /dev/tty56  /dev/ttyp0...ck  /dev/ttyS2  /dev/ttyS31
/dev/tty1  /dev/tty21  /dev/tty33  /dev/tty45  /dev/tty57  /dev/ttyS0  /dev/ttyS20  /dev/ttyS4
/dev/tty10 /dev/tty22  /dev/tty34  /dev/tty46  /dev/tty58  /dev/ttyS1  /dev/ttyS21  /dev/ttyS5
/dev/tty11 /dev/tty23  /dev/tty35  /dev/tty47  /dev/tty59  /dev/ttyS10 /dev/ttyS22  /dev/ttyS6
/dev/tty12 /dev/tty24  /dev/tty36  /dev/tty48  /dev/tty6  /dev/ttyS11 /dev/ttyS23  /dev/ttyS7
/dev/tty13 /dev/tty25  /dev/tty37  /dev/tty49  /dev/tty60 /dev/ttyS12 /dev/ttyS24  /dev/ttyS8
/dev/tty14 /dev/tty26  /dev/tty38  /dev/tty5  /dev/tty61 /dev/ttyS13 /dev/ttyS25  /dev/ttyS9
/dev/tty15 /dev/tty27  /dev/tty39  /dev/tty50  /dev/tty62 /dev/ttyS14 /dev/ttyS26
/dev/tty16 /dev/tty28  /dev/tty4  /dev/tty51  /dev/tty63 /dev/ttyS15 /dev/ttyS27
/dev/tty17 /dev/tty29  /dev/tty40  /dev/tty52  /dev/tty7  /dev/ttyS16 /dev/ttyS28
/dev/tty18 /dev/tty3  /dev/tty41  /dev/tty53  /dev/tty8  /dev/ttyS17 /dev/ttyS29
/dev/tty19 /dev/tty30  /dev/tty42  /dev/tty54  /dev/tty9  /dev/ttyS18 /dev/ttyS3
(myenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

在 Linux 中调用“ttyACM0”通常需要较高的权限，因此使用指令提升用户权限是必须的。

```
sudo usermod -a -G dialout $USER
sudo reboot
```



```
lin@ubuntu:~/ch343ser_linux/driver$ sudo usermod -a -G dialout $USER
lin@ubuntu:~/ch343ser_linux/driver$ sudo reboot
```

小智 AI 固件

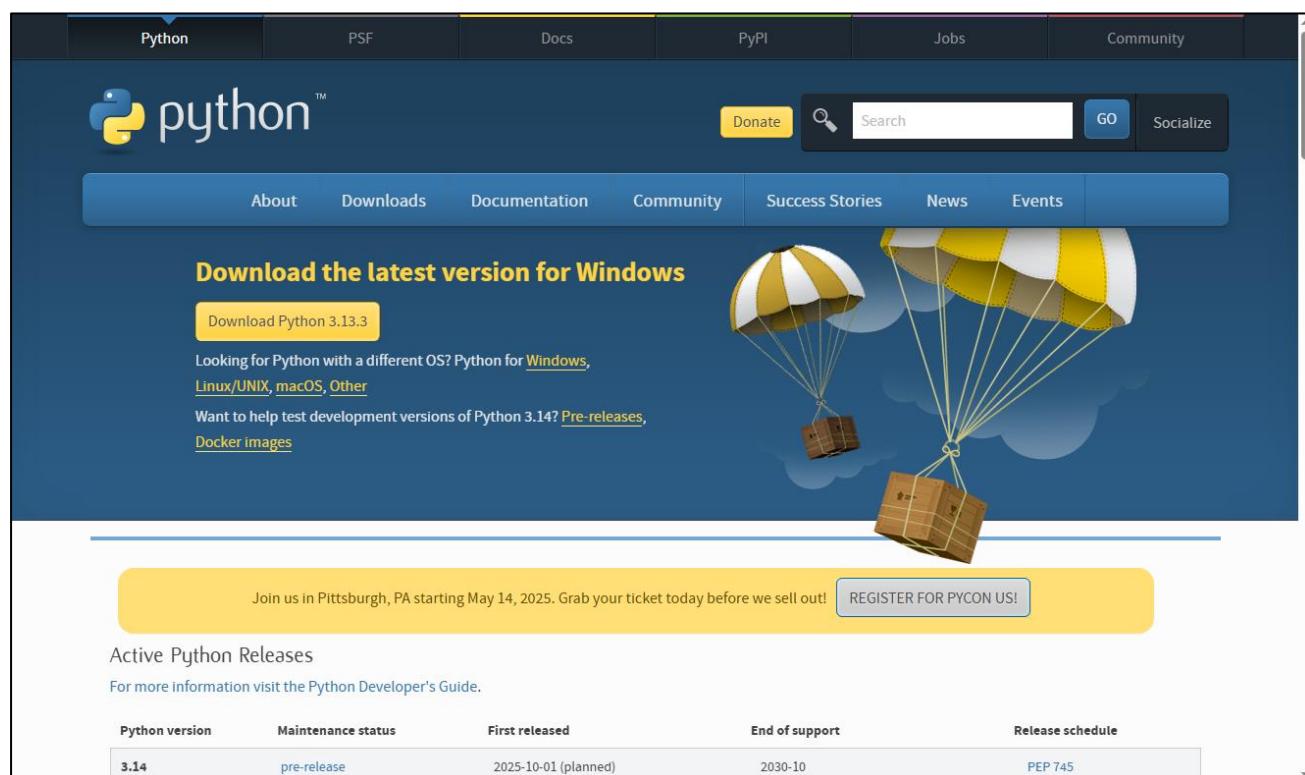
如果您的硬件还没有小智的固件，您可以参考接下来的教程，重新给 ESP32 S3 WROOM 上传固件。如果您的硬件已经集成了小智的固件，您可以跳过这个章节。

安装 Python (Importance)

Windows

请下载并安装 **Python3** 软件包。

<https://www.python.org/downloads/windows/>

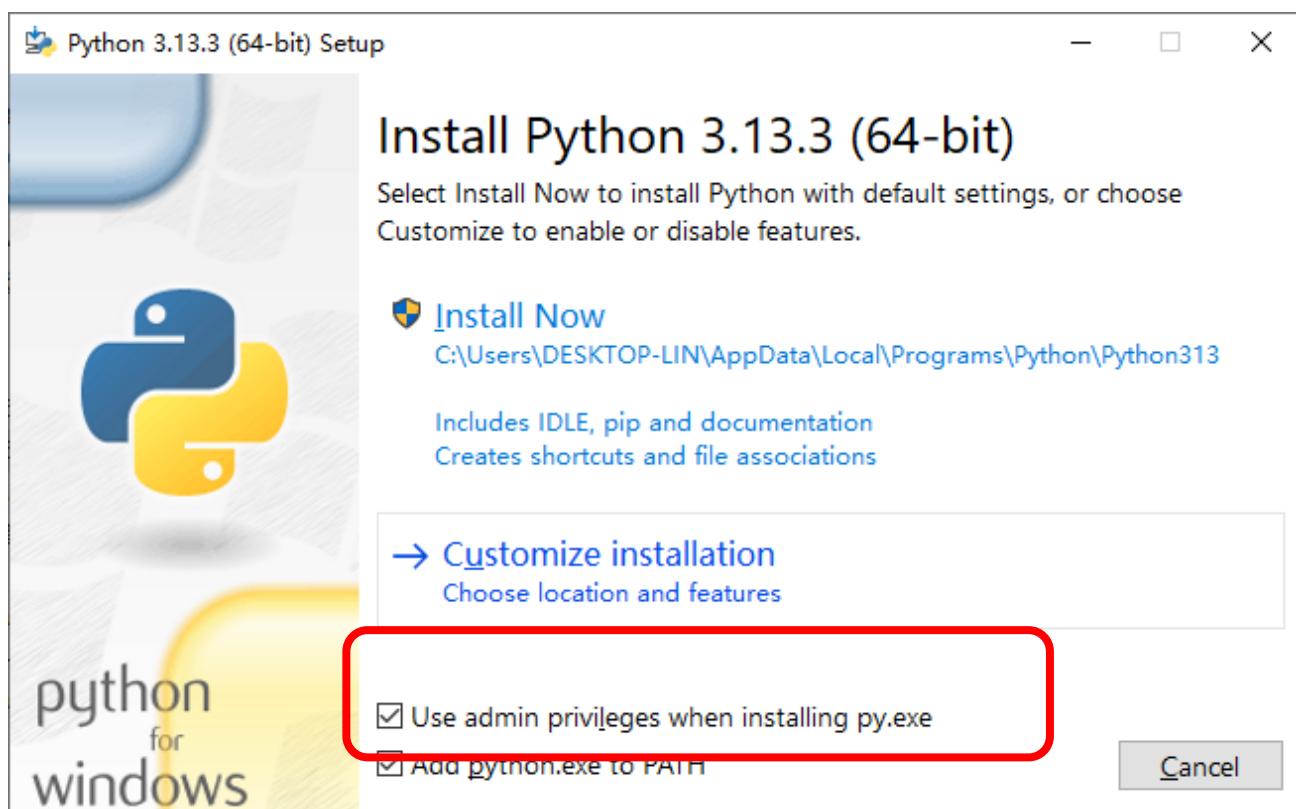


The screenshot shows the Python.org homepage with a dark blue header. The navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar with a magnifying glass icon and a 'GO' button. The main content area features the Python logo and the text "Download the latest version for Windows". A large graphic of two yellow and white striped parachutes descending towards the ground is positioned on the right. Below this, there are links for "Download Python 3.13.3", "Windows", "Linux/UNIX", "macOS", and "Other". There's also a link for "Pre-releases" and "Docker images". At the bottom of the page, there's a yellow banner with the text "Join us in Pittsburgh, PA starting May 14, 2025. Grab your ticket today before we sell out!" and a "REGISTER FOR PYCON US!" button. The footer contains sections for "Active Python Releases" and a table showing release details for Python 3.14.

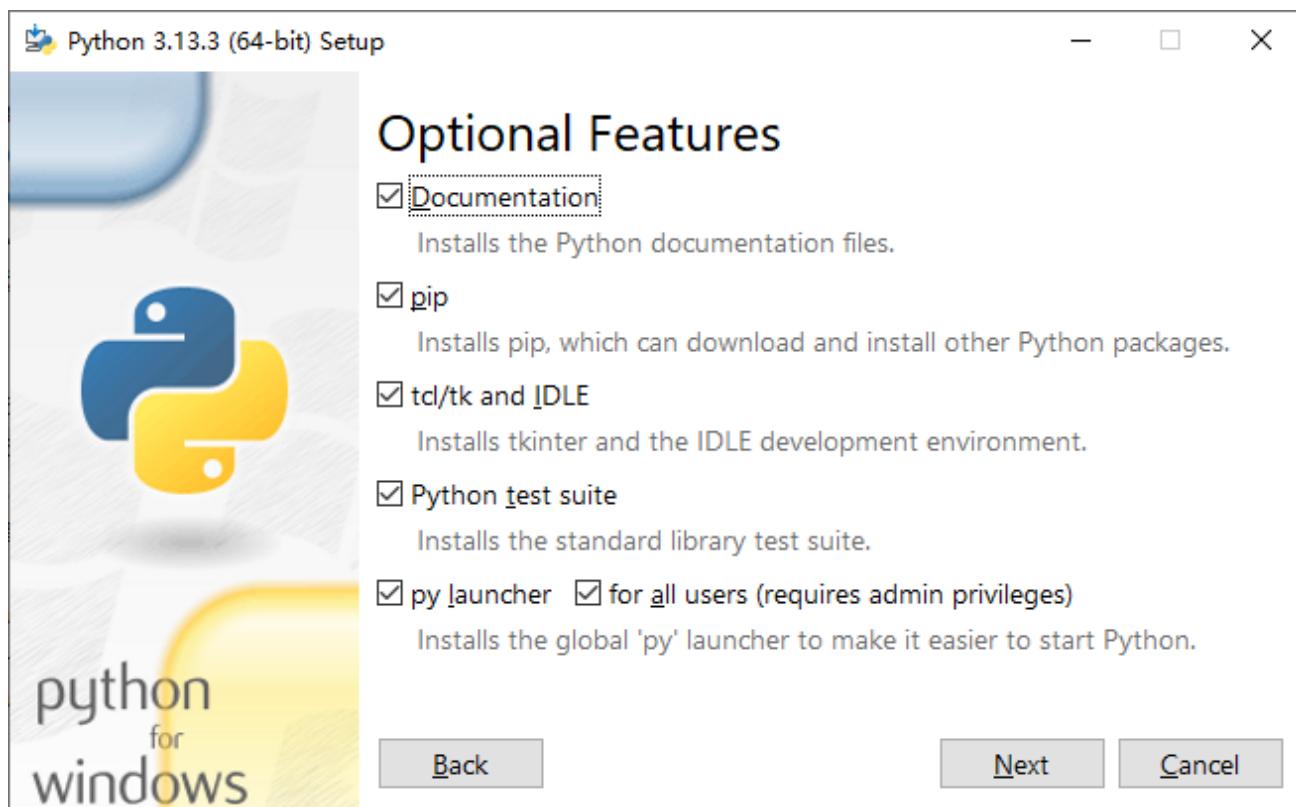
Python version	Maintenance status	First released	End of support	Release schedule
3.14	pre-release	2025-10-01 (planned)	2030-10	PEP 745

点击下载 **Python 3.13.3**

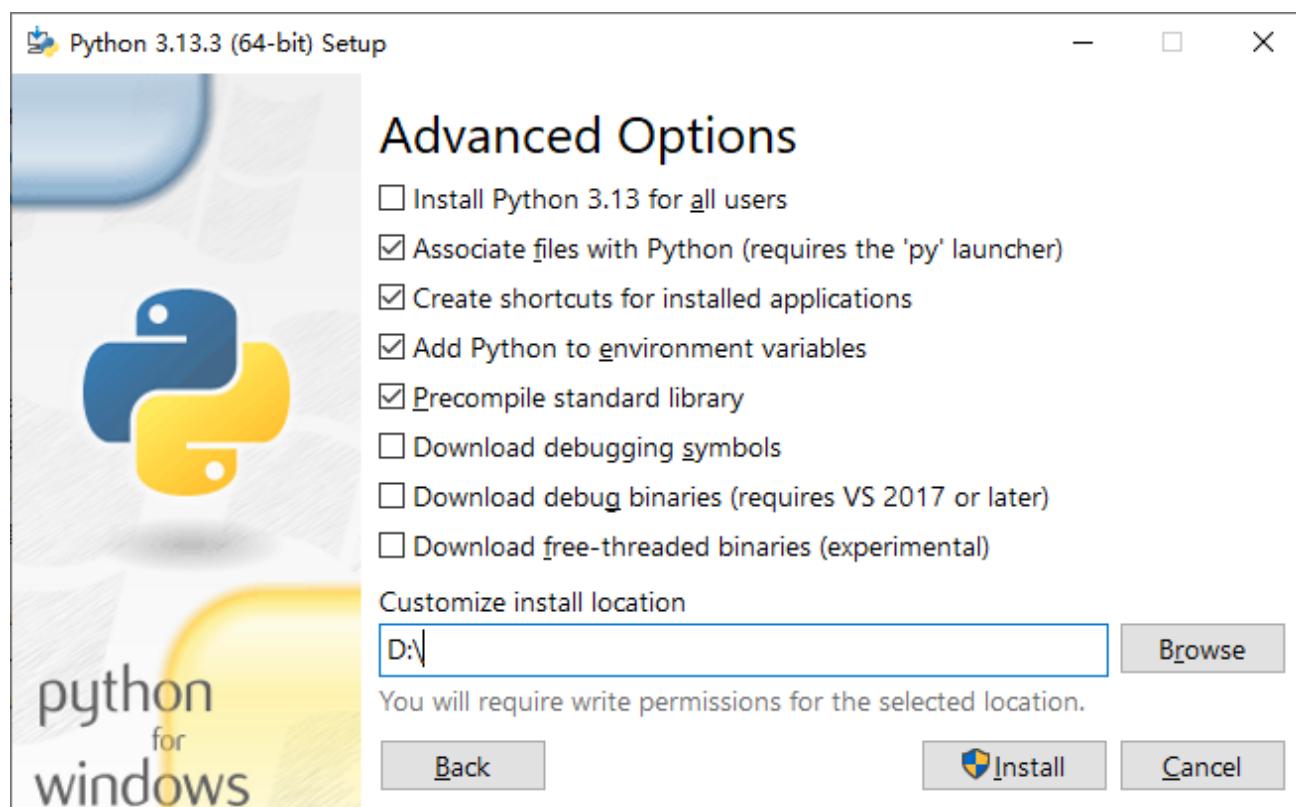
重要提示：务必勾选 "Add Python 3.13 to PATH" 选项。



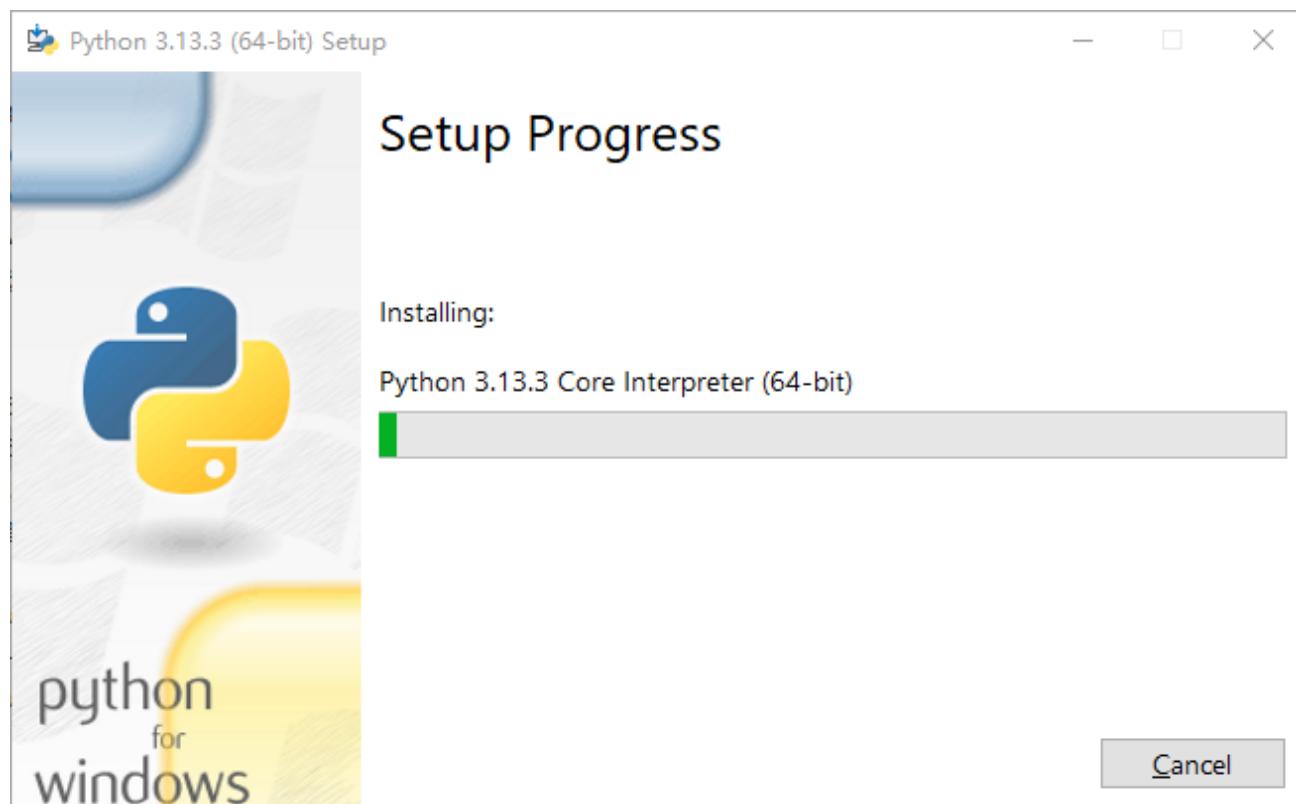
勾选所有选项后，点击“Next”。



此处可设置 Python 的安装路径。我们将其安装在 **D 盘**。若您是新手，可直接选择默认路径。



等待安装完成



现在安装已完成。



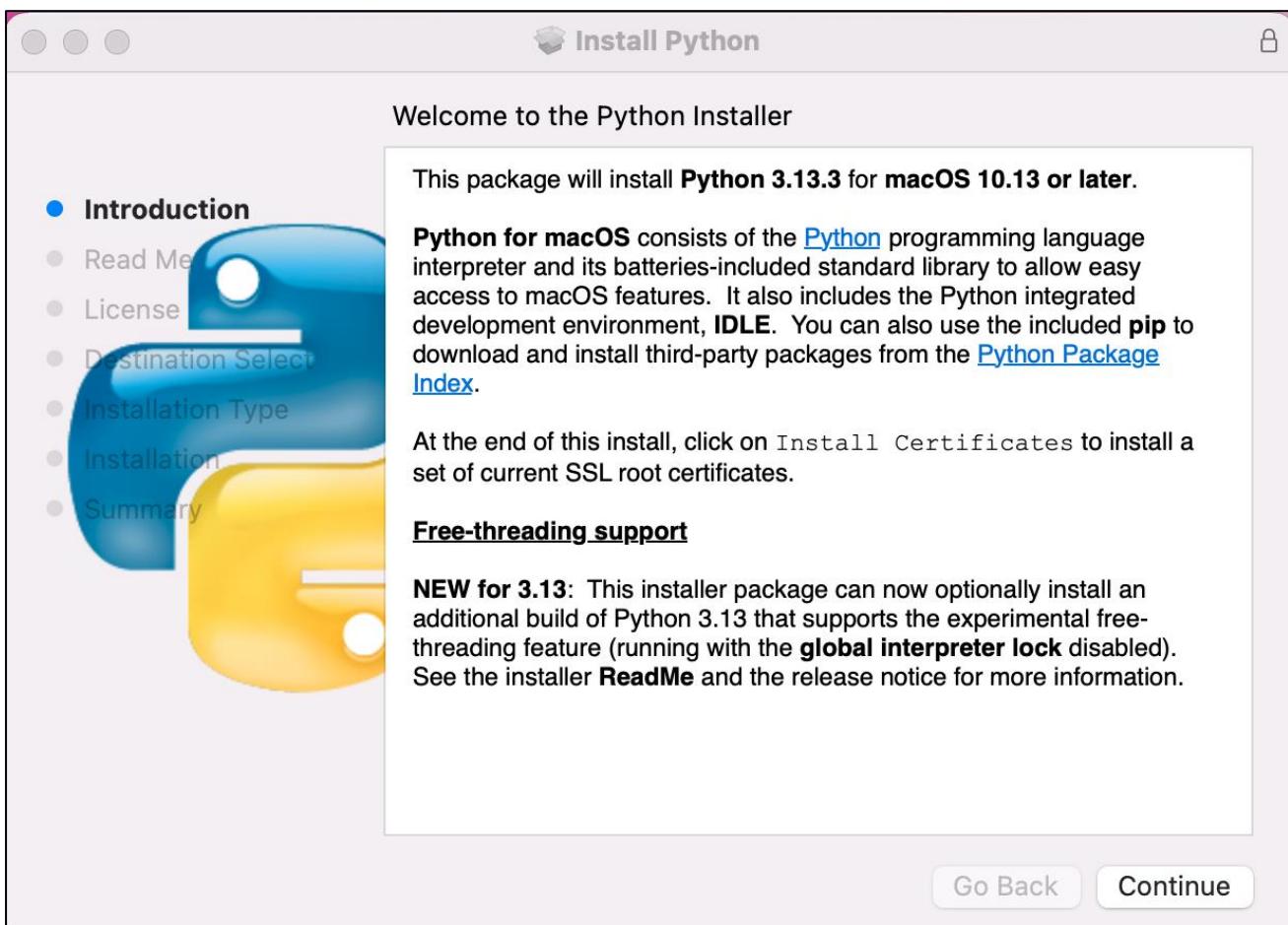
MAC

下载安装包（链接：<https://www.python.org/downloads/>）

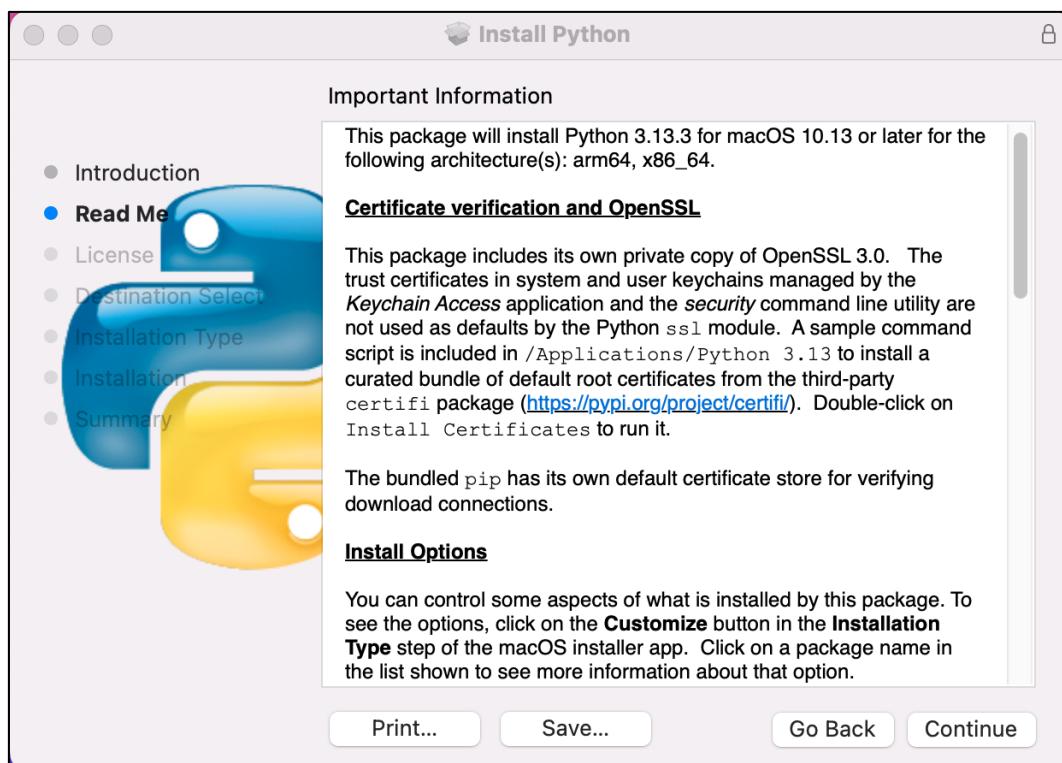
点击下载 Python 3.13.3



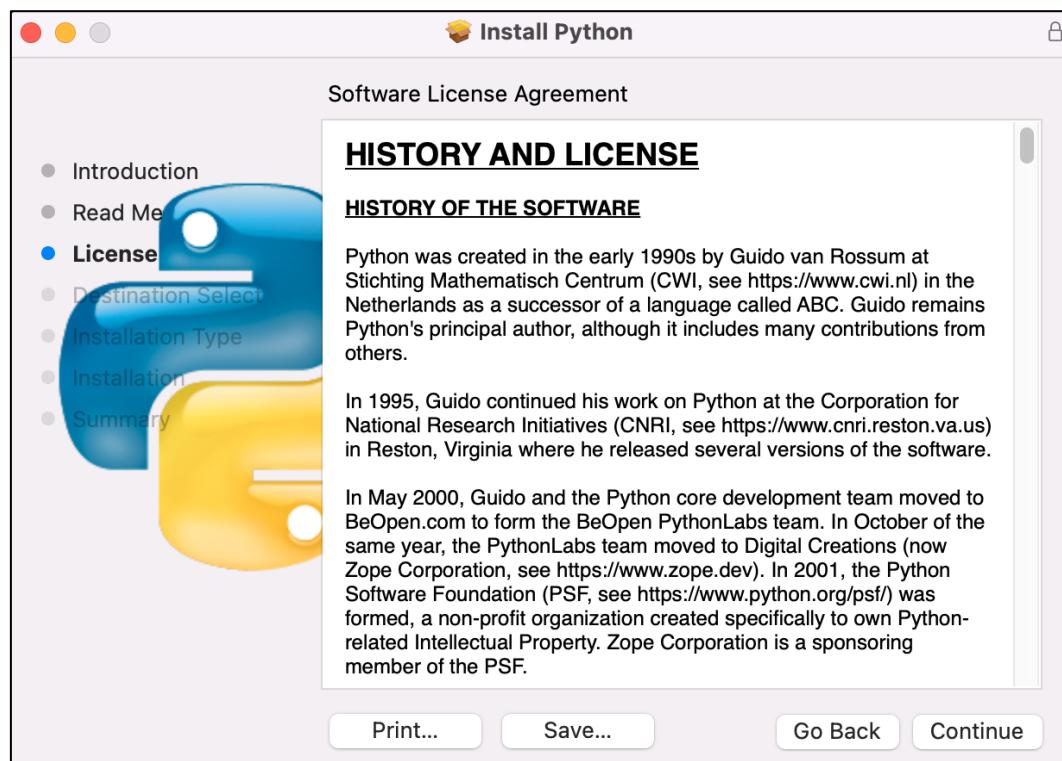
运行已下载的安装包，点击 "Continue" 按钮。



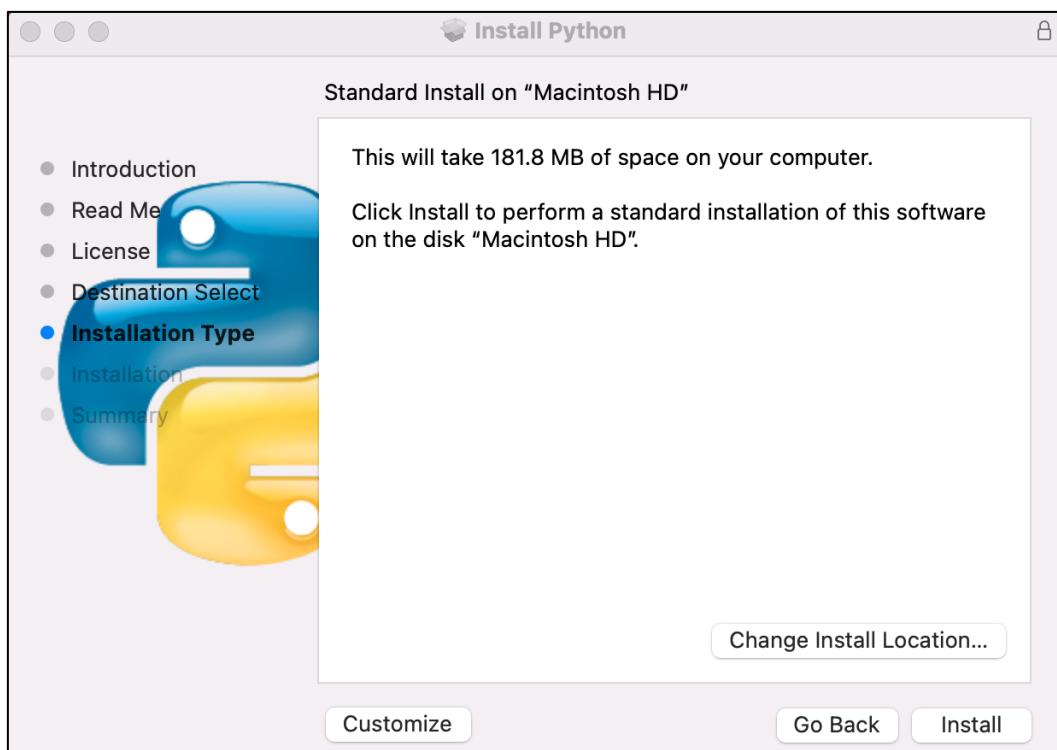
点击"Continue"



点击"Continue"



点击 "Install"。若电脑设有密码, 请输入密码后继续安装。



安装完成



Linux

检查您的系统是否已经安装了 Python3，Python 要求 Python3.10 以上的版本。

```
python --version  
python3 --version
```

```
lin@ubuntu:~$ python --version  
python: command not found  
lin@ubuntu:~$ python3 --version  
bash: /usr/bin/python3: No such file or directory  
lin@ubuntu:~$
```

安装 python3， 默认安装最新版本。

```
sudo apt install python3
```

```
lin@ubuntu:~$ sudo apt install python3  
Installing:  
  python3
```

将 python 链接到 python3。

```
sudo rm /usr/bin/python  
sudo ln -s /usr/bin/python3 /usr/bin/python
```

```
lin@ubuntu:~$ sudo rm /usr/bin/python  
lin@ubuntu:~$ sudo ln -s /usr/bin/python3 /usr/bin/python  
lin@ubuntu:~$ python --version  
Python 3.13.3
```

安装 python3.13-venv 虚拟环境。

```
sudo apt install python3-venv
```

```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ sudo apt install python3-venv
```

安装 pip。

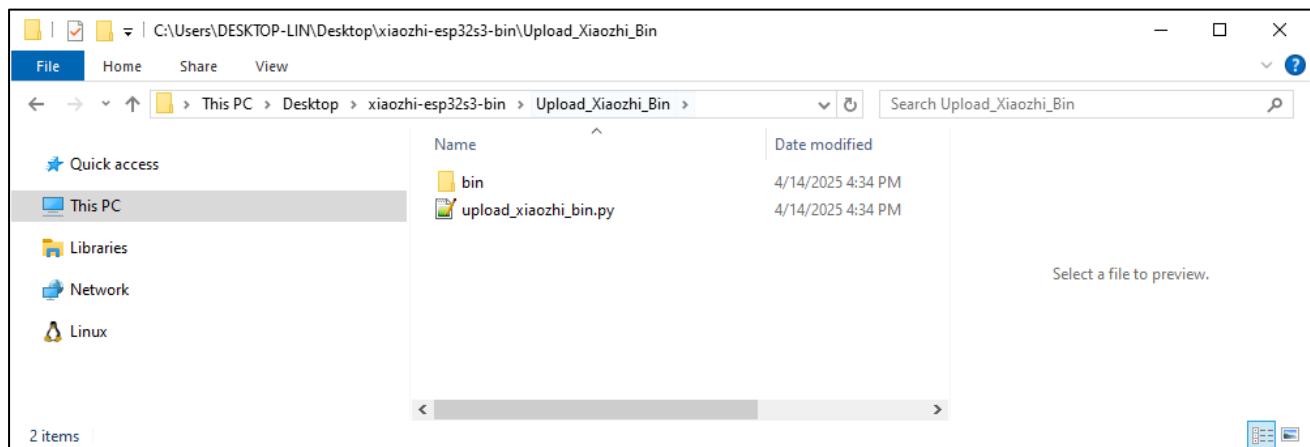
```
sudo apt install python3-pip
```

```
lin@ubuntu:~$ sudo apt install python3-pip  
python3-pip is already the newest version (25.0+dfsg-1).  
Summary:  
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0  
lin@ubuntu:~$ pip --version  
pip 25.0 from /usr/lib/python3/dist-packages/pip (python 3.13)  
lin@ubuntu:~$
```

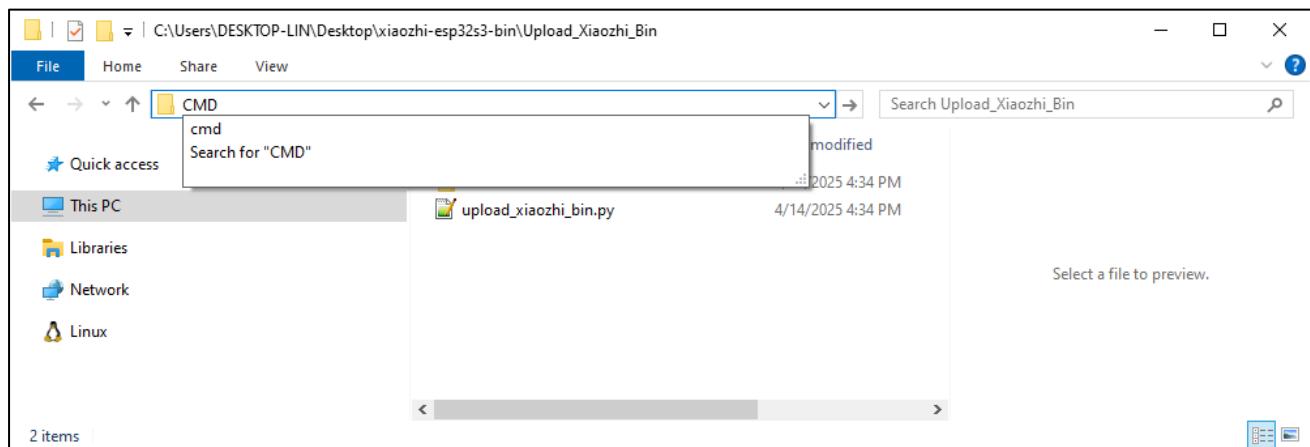
上传固件

Windows

进入 Upload_Xiaozhi_Bin 文件夹。



在文件栏输入“CMD”，并按下回车键。



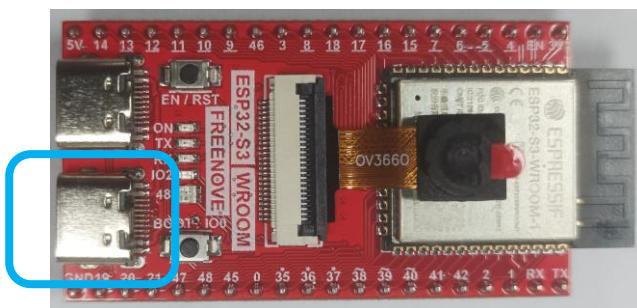
输入“python --version”，查看是否已经安装 Python 环境，如果没有打印 Python 的版本信息，说明 Python 没有正确安装环境，请重新安装。

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DESKTOP-LIN\Desktop\xiaozhi-esp32s3-bin\Upload_Xiaozhi_Bin>python --version
Python 3.13.3

C:\Users\DESKTOP-LIN\Desktop\xiaozhi-esp32s3-bin\Upload_Xiaozhi_Bin>
```

使用数据线连接电脑和 ESP32 S3 WROOM，请注意，不要连接错 Type C 接口。



输入“python upload_xiaozhi_bin.py”，并按下回车。

如果您的电脑没有安装 esptool 及其相关的软件库，它会自动安装这些库。

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DESKTOP-LIN\Desktop\xiaozhi-esp32s3-bin\Upload_Xiaozhi_Bin>python --version
Python 3.13.3

C:\Users\DESKTOP-LIN\Desktop\xiaozhi-esp32s3-bin\Upload_Xiaozhi_Bin>python upload_xiaozhi_bin.py
esptool is not installed. Installing now...
Looking in indexes: https://mirrors.aliyun.com/pypi/simple/
Collecting esptool
  Using cached https://mirrors.aliyun.com/pypi/packages/5c/6b/3ce9bb7f36bdef3d6ae71646a1d3b7d59826a4
78f3ed8a783a93a2f8f537/esptool-4.8.1.tar.gz (409 kB)
    Installing build dependencies ... done
    Getting requirements to build wheel ... done
    Preparing metadata (pyproject.toml) ... done
Collecting bitstring!=4.2.0,>=3.1.6 (from esptool)
  Downloading https://mirrors.aliyun.com/pypi/packages/75/2d/174566b533755ddf8efb32a5503af61c756a983
de379f8ad3aed6a982d38/bitstring-4.3.1-py3-none-any.whl (71 kB)
Collecting cryptography>=2.1.4 (from esptool)
  Downloading https://mirrors.aliyun.com/pypi/packages/33/cf/1f7649b8b9a3543e042d3f348e398a061923ac0
5b507f3f4d95f11938aa9/cryptography-44.0.2-cp39-abi3-win_amd64.whl (3.2 MB)
    3.2 / 3.2 MB 2.6 MB/s eta 0:00:00
```

然后，它会调用 esptool 将 bin 文件夹中的文件上传到 ESP32 S3 WROOM 中。

```
C:\Windows\System32\cmd.exe
SHA digest in image updated
Compressed 16352 bytes to 11342...
Wrote 16352 bytes (11342 compressed) at 0x00000000 in 0.2 seconds (effective 802.0 kbit/s)...
Hash of data verified.
Compressed 3561632 bytes to 2079901...
Wrote 3561632 bytes (2079901 compressed) at 0x00100000 in 22.8 seconds (effective 1247.5 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 141...
Wrote 3072 bytes (141 compressed) at 0x00008000 in 0.0 seconds (effective 1025.9 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 31...
Wrote 8192 bytes (31 compressed) at 0x0000d000 in 0.0 seconds (effective 1812.3 kbit/s)...
Hash of data verified.
Compressed 873228 bytes to 644882...
Wrote 873228 bytes (644882 compressed) at 0x00010000 in 6.4 seconds (effective 1090.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
esptool command executed successfully.

C:\Users\DESKTOP-LIN\Desktop\xiaozhi-esp32s3-bin\Upload_Xiaozhi_Bin>
```

ESP32 S3 WROOM 显示如下。



MAC

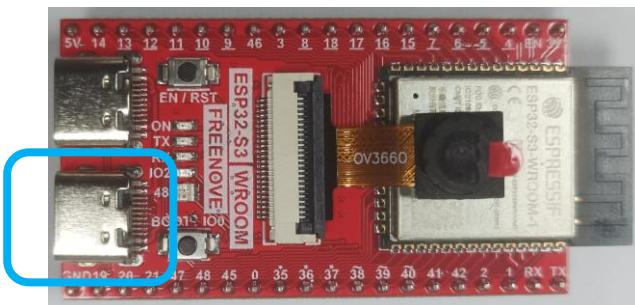
进入 Upload_Xiaozhi_Bin 文件夹。

```
Upload_Xiaozhi_Bin -- zsh -- 91x24
freenove@PandeMacBook-Air ~ % cd Desktop/xiaozhi-esp32s3-bin/Upload_Xiaozhi_Bin
freenove@PandeMacBook-Air Upload_Xiaozhi_Bin %
```

输入“**python –version**”，查看是否已经安装 Python 环境，如果没有打印 Python 的版本信息，说明 Python 没有正确安装环境，请重新安装。

```
Upload_Xiaozhi_Bin -- zsh -- 91x24
freenove@PandeMacBook-Air Upload_Xiaozhi_Bin % python3 --version
Python 3.13.3
freenove@PandeMacBook-Air Upload_Xiaozhi_Bin %
```

使用数据线连接电脑和 ESP32 S3 WROOM，请注意，不要连接错 Type C 接口。



输入“**python upload_xiaozhi_bin.py**”，并按下回车。

```
Upload_Xiaozhi_Bin -- zsh -- 91x24
freenove@PandeMacBook-Air Upload_Xiaozhi_Bin % python3 upload_xiaozhi_bin.py
```

然后，它会调用 esptool 将 bin 文件夹中的文件上传到 ESP32 S3 WROOM 中。

```
Serial port /dev/cu.wchusbserial5A4E1051341
Connecting....
Chip is ESP32-S3 (QFN56) (revision v0.2)
Features: WiFi, BLE, Embedded PSRAM 8MB (AP_3v3)
Crystal is 40MHz
MAC: 30:ed:a0:20:bd:9c
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 2000000
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00003fff...
Flash will be erased from 0x00100000 to 0x00465fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000d000 to 0x0000efff...
Flash will be erased from 0x00010000 to 0x000e5fff...
```

```
SHA digest in image updated
Compressed 16352 bytes to 11342...
Wrote 16352 bytes (11342 compressed) at 0x00000000 in 0.2 seconds (effective 775.3 kbit/s).
..
Hash of data verified.
Compressed 3561632 bytes to 2079901...
Wrote 3561632 bytes (2079901 compressed) at 0x00100000 in 22.8 seconds (effective 1247.2 kb
it/s)...
Hash of data verified.
Compressed 3072 bytes to 141...
Wrote 3072 bytes (141 compressed) at 0x00008000 in 0.0 seconds (effective 889.1 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 31...
Wrote 8192 bytes (31 compressed) at 0x0000d000 in 0.0 seconds (effective 1663.8 kbit/s)...
Hash of data verified.
Compressed 873228 bytes to 644882...
Wrote 873228 bytes (644882 compressed) at 0x00010000 in 6.4 seconds (effective 1089.5 kbit/
s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
esptool command executed successfully.
freenove@PandeMacBook-Air Upload_Xiaozhi_Bin % c
```

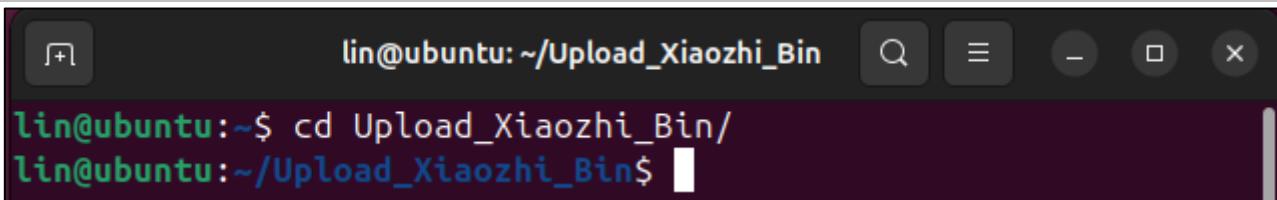
ESP32 S3 WROOM 显示如下。



Linux

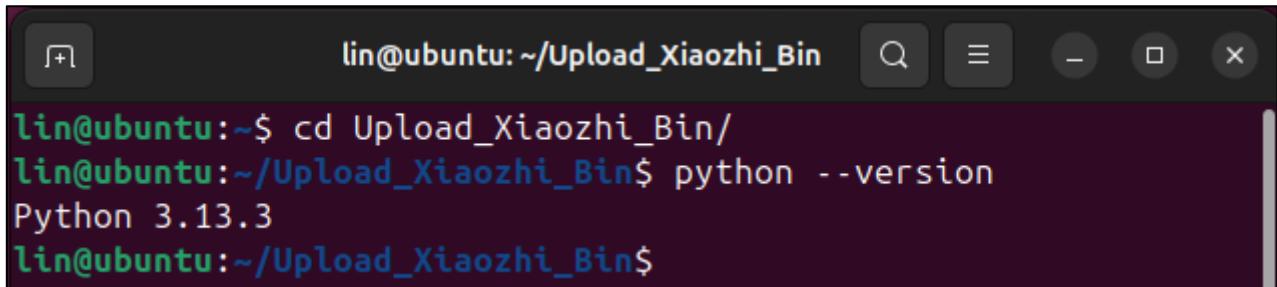
进入 Upload_Xiaozhi_Bin 文件夹。

```
cd Upload_Xiaozhi_Bin
```



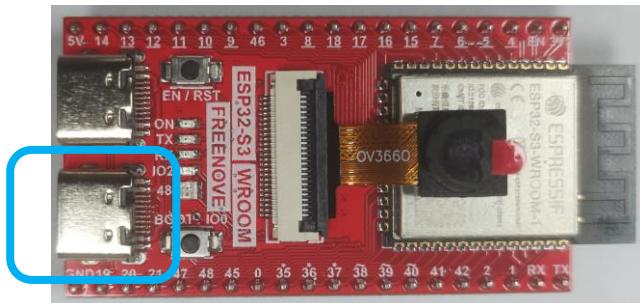
```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ cd Upload_Xiaozhi_Bin/
lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

输入“`python --version`”，查看是否已经安装 Python 环境，如果没有打印 Python 的版本信息，说明 Python 没有正确安装环境，请重新安装。



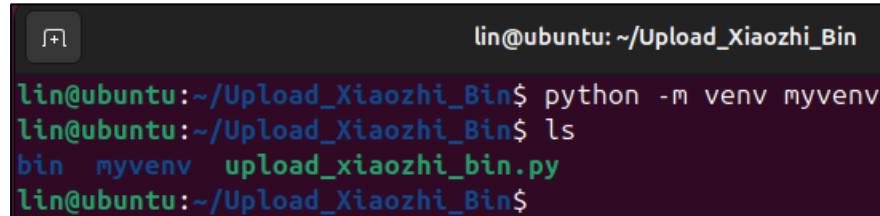
```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ cd Upload_Xiaozhi_Bin/
lin@ubuntu:~/Upload_Xiaozhi_Bin$ python --version
Python 3.13.3
lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

使用数据线连接电脑和 ESP32 S3 WROOM，请注意，不要连接错 Type C 接口。



创建一个虚拟环境，并命名为“myvenv”。

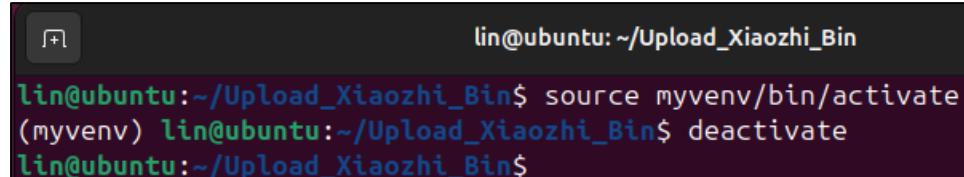
```
python -m venv myvenv
```



```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ python -m venv myvenv
lin@ubuntu:~/Upload_Xiaozhi_Bin$ ls
bin  myvenv  upload_xiaozhi_bin.py
lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

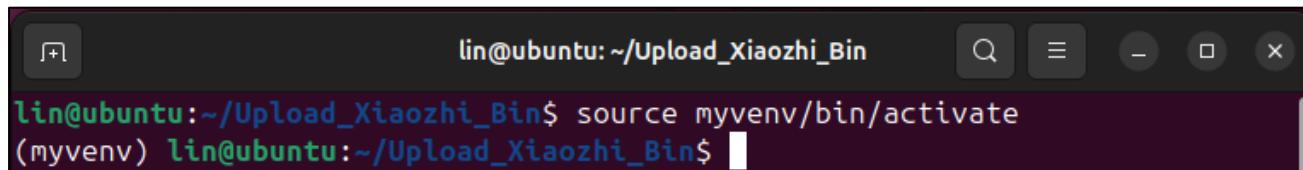
您可以使用下面的指令激活和关闭虚拟环境。

```
source myvenv/bin/activate
deactivate
```



```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ source myvenv/bin/activate
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$ deactivate
lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

激活虚拟环境。

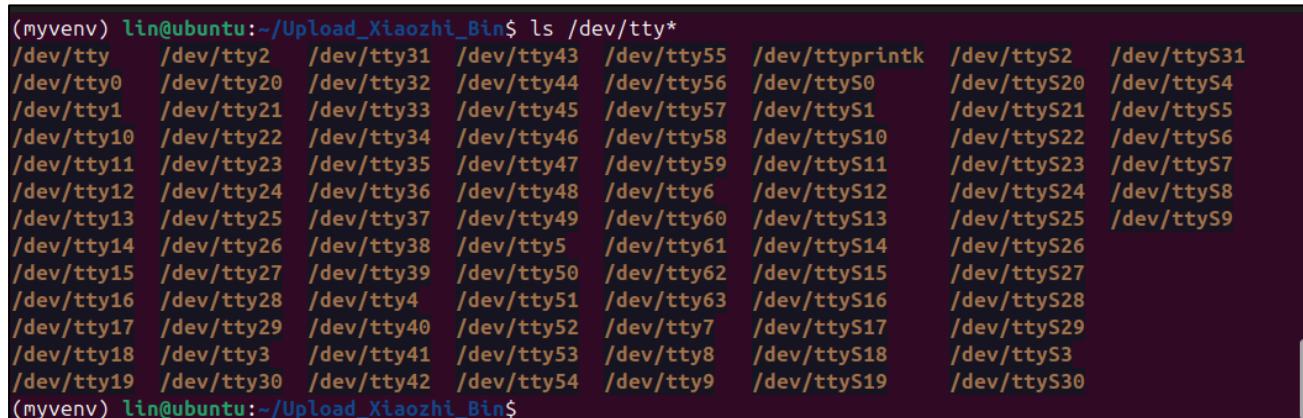


```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ source myvenv/bin/activate
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

使用指令查看 ESP32S3 的端口号。

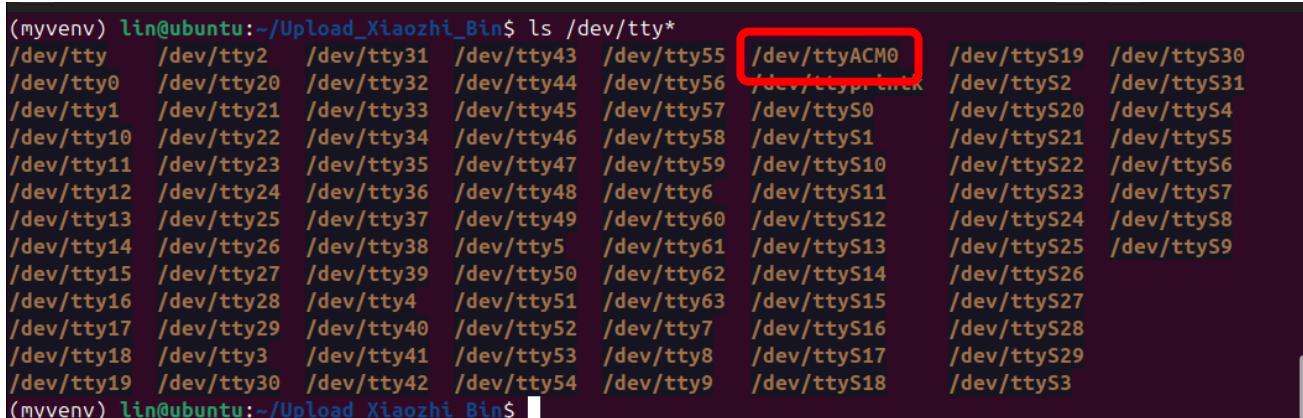
ls /dev/tty*

当 ESP32S3 没有连接到电脑时，使用指令后的显示内容如下图所示。



```
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$ ls /dev/tty*
/dev/tty  /dev/tty2  /dev/tty31  /dev/tty43  /dev/tty55  /dev/ttysprintk  /dev/ttyS2  /dev/ttyS31
/dev/tty0  /dev/tty20 /dev/tty32  /dev/tty44  /dev/tty56  /dev/ttyS0  /dev/ttyS20  /dev/ttyS4
/dev/tty1  /dev/tty21 /dev/tty33  /dev/tty45  /dev/tty57  /dev/ttyS1  /dev/ttyS21  /dev/ttyS5
/dev/tty10 /dev/tty22 /dev/tty34  /dev/tty46  /dev/tty58  /dev/ttyS10  /dev/ttyS22  /dev/ttyS6
/dev/tty11 /dev/tty23 /dev/tty35  /dev/tty47  /dev/tty59  /dev/ttyS11  /dev/ttyS23  /dev/ttyS7
/dev/tty12 /dev/tty24 /dev/tty36  /dev/tty48  /dev/tty6  /dev/ttyS12  /dev/ttyS24  /dev/ttyS8
/dev/tty13 /dev/tty25 /dev/tty37  /dev/tty49  /dev/tty60  /dev/ttyS13  /dev/ttyS25  /dev/ttyS9
/dev/tty14 /dev/tty26 /dev/tty38  /dev/tty5  /dev/tty61  /dev/ttyS14  /dev/ttyS26
/dev/tty15 /dev/tty27 /dev/tty39  /dev/tty50  /dev/tty62  /dev/ttyS15  /dev/ttyS27
/dev/tty16 /dev/tty28 /dev/tty4  /dev/tty51  /dev/tty63  /dev/ttyS16  /dev/ttyS28
/dev/tty17 /dev/tty29 /dev/tty40  /dev/tty52  /dev/tty7  /dev/ttyS17  /dev/ttyS29
/dev/tty18 /dev/tty3  /dev/tty41  /dev/tty53  /dev/tty8  /dev/ttyS18  /dev/ttyS3
/dev/tty19 /dev/tty30 /dev/tty42  /dev/tty54  /dev/tty9  /dev/ttyS19  /dev/ttyS30
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

当将 ESP32S3 连接到电脑时，使用指令后的显示内容如下图所示。



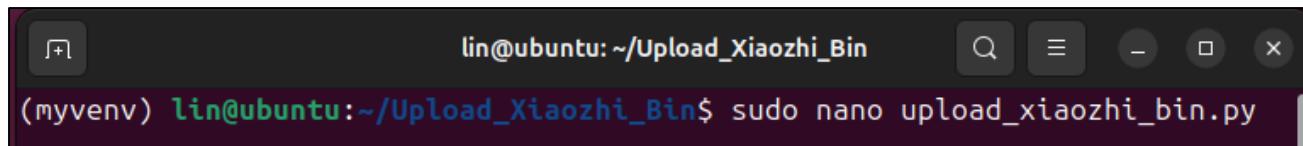
```
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$ ls /dev/tty*
/dev/tty  /dev/tty2  /dev/tty31  /dev/tty43  /dev/tty55  /dev/ttyACM0  /dev/ttyS19  /dev/ttyS30
/dev/tty0  /dev/tty20 /dev/tty32  /dev/tty44  /dev/tty56  /dev/ttysprintk  /dev/ttyS2  /dev/ttyS31
/dev/tty1  /dev/tty21 /dev/tty33  /dev/tty45  /dev/tty57  /dev/ttyS0  /dev/ttyS20  /dev/ttyS4
/dev/tty10 /dev/tty22 /dev/tty34  /dev/tty46  /dev/tty58  /dev/ttyS1  /dev/ttyS21  /dev/ttyS5
/dev/tty11 /dev/tty23 /dev/tty35  /dev/tty47  /dev/tty59  /dev/ttyS10  /dev/ttyS22  /dev/ttyS6
/dev/tty12 /dev/tty24 /dev/tty36  /dev/tty48  /dev/tty6  /dev/ttyS11  /dev/ttyS23  /dev/ttyS7
/dev/tty13 /dev/tty25 /dev/tty37  /dev/tty49  /dev/tty60  /dev/ttyS12  /dev/ttyS24  /dev/ttyS8
/dev/tty14 /dev/tty26 /dev/tty38  /dev/tty5  /dev/tty61  /dev/ttyS13  /dev/ttyS25  /dev/ttyS9
/dev/tty15 /dev/tty27 /dev/tty39  /dev/tty50  /dev/tty62  /dev/ttyS14  /dev/ttyS26
/dev/tty16 /dev/tty28 /dev/tty4  /dev/tty51  /dev/tty63  /dev/ttyS15  /dev/ttyS27
/dev/tty17 /dev/tty29 /dev/tty40  /dev/tty52  /dev/tty7  /dev/ttyS16  /dev/ttyS28
/dev/tty18 /dev/tty3  /dev/tty41  /dev/tty53  /dev/tty8  /dev/ttyS17  /dev/ttyS29
/dev/tty19 /dev/tty30 /dev/tty42  /dev/tty54  /dev/tty9  /dev/ttyS18  /dev/ttyS30
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

对比前后的区别，多出来的端口号就是 ESP32S3 连接到电脑上的端口号。

在运行 python 文件之前，需要修改文件中的端口号。

使用指令打开 python 文件。

sudo nano upload_xiaozhi_bin.py



```
(myvenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$ sudo nano upload_xiaozhi_bin.py
```

在文本编辑器中，找到“`--port`”，“`COMx`”这一行，将“`COMx`”修改为 ESP32S3 在电脑上的端口号。

```

lin@ubuntu: ~/Upload_Xiaozhi_Bin
GNU nano 8.3          upload_xiaozhi_bin.py
sys.exit(1)

def run_esptool_command():
    """Execute the esptool command"""
    command = [
        sys.executable, "-m", "esptool",
        "--chip", "esp32s3",
        #"--port", "COMx",
        "--baud", "2000000",
        "--before", "default_reset",
        "--after", "hard_reset",
    ]
    [ Wrote 64 lines ]
^G Help      ^O Write Out  ^F Where Is  ^K Cut      ^T Execute
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify

```

修改后如下图所示。

```

lin@ubuntu: ~/Upload_Xiaozhi_Bin
GNU nano 8.3          upload_xiaozhi_bin.py *
sys.exit(1)

def run_esptool_command():
    """Execute the esptool command"""
    command = [
        sys.executable, "-m", "esptool",
        "--chip", "esp32s3",
        "--port", "/dev/ttyACM0",
        "--baud", "2000000",
        "--before", "default_reset",
        "--after", "hard_reset",
    ]
    [ Wrote 64 lines ]
^G Help      ^O Write Out  ^F Where Is  ^K Cut      ^T Execute
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify

```

使用 Crtl+O，保存文件。

使用 Ctrl+X，退出编辑器。

运行 python 文件。

```
python upload_xiaozhi_bin.py
```

```
lin@ubuntu:~/Upload_Xiaozhi_Bin$ python upload_xiaozhi_bin.py
esptool is already installed.
Executing esptool command...
esptool.py v4.8.1
Serial port /dev/ttyACM0
Connecting....
Chip is ESP32-S3 (QFN56) (revision v0.2)
Features: WiFi, BLE, Embedded PSRAM 8MB (AP_3v3)
Crystal is 40MHz
MAC: 30:ed:a0:20:bd:9c
Uploading stub...
Running stub...
```

代码上传完成，如下图所示。

```
Wrote 8192 bytes (31 compressed) at 0x0000d000 in 0.0 seconds (effective 18
43.6 kbit/s)...
Hash of data verified.
Compressed 873228 bytes to 644882...
Wrote 873228 bytes (644882 compressed) at 0x00010000 in 6.3 seconds (effect
ive 1103.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
esptool command executed successfully.
(myenv) lin@ubuntu:~/Upload_Xiaozhi_Bin$
```

ESP32 S3 WROOM 显示如下。



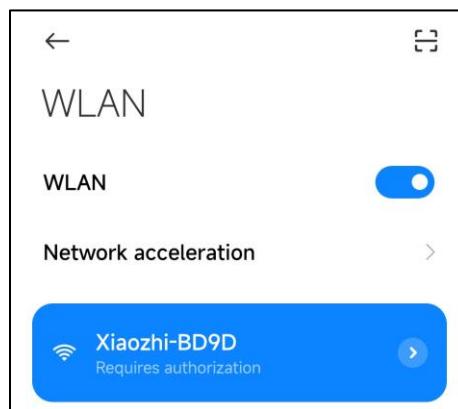


ESP32 S3 WROOM 配网

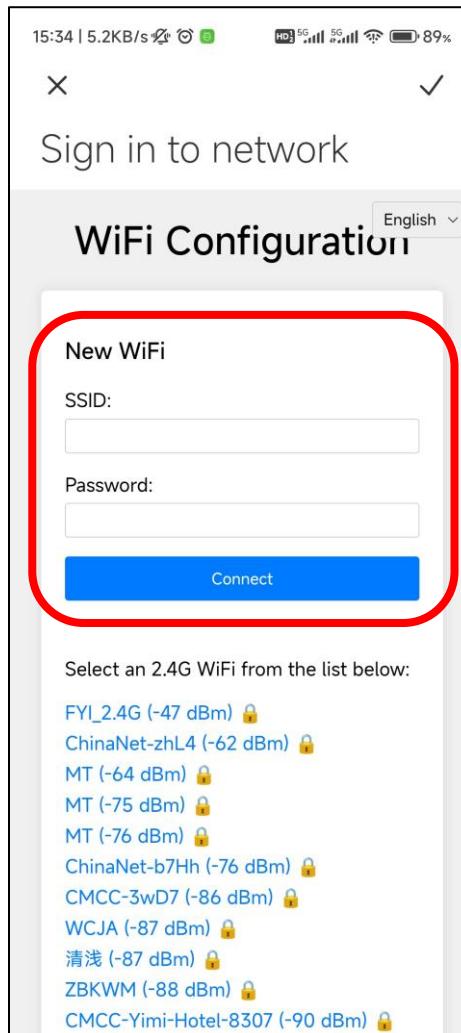
如果您的 ESP32 S3 WROOM 还没有集成小智 AI 固件，请跳转到[小智 AI 固件](#)。

如果您想学习小智 AI 的代码，请跳转到[小智 AI 代码](#)，请跳转到[小智 AI 代码](#)。

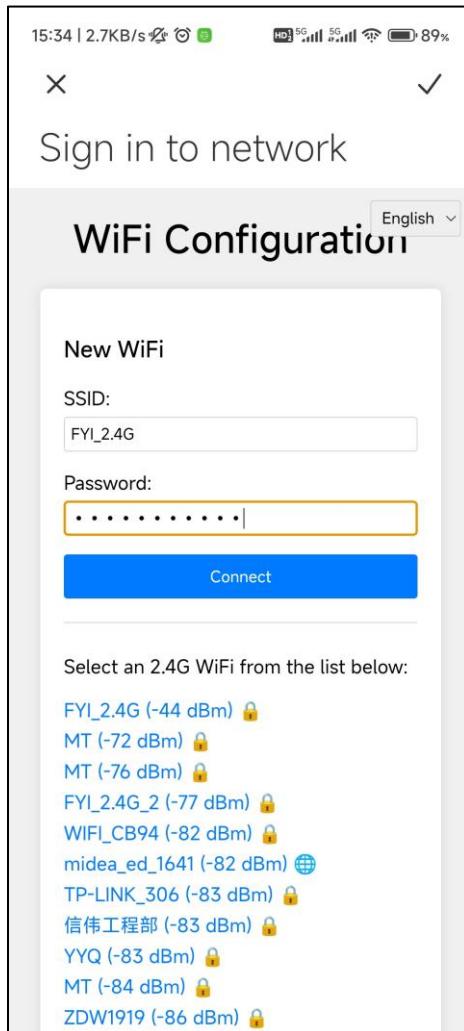
如果您的 ESP32 S3 WROOM 已经集成了小智 AI 的固件，请使用您的手机，打开 WiFi 功能，您可以搜索到一个叫“Xiaozhi-XXXX”的路由器 WiFi。请连接它，它是无密码的。



连接 WiFi 后，请按照提示点击它。它将会打开您的手机浏览器，并访问“192.168.4.1”这个网址。



在 SSID 中输入您的家庭路由器 WiFi 名称, 在 Password 中输入您的家庭路由器 WiFi 密码。然后点击 Connect。请注意, ESP32 S3 WROOM 只接收 2.4GHz 频段的路由器, 如果您的路由器同时支持 2.4GHz 和 5GHz, 请注意, 不能让 ESP32 S3 WROOM 连接 5Ghz 频段。也不能将 2.4Ghz 和 5Ghz 设置为混合模式。这都会导致 ESP32 S3 WROOM 无法正常联网工作。



当您看到这个界面时, 说明您已经成功让 ESP32 S3 Wroom 配网成功, 它将会加入您的家庭路由器 WiFi 网络中。您可以看到 ESP32 S3 WROOM 自动连接 WiFi, 然后屏幕显示如下图所示。



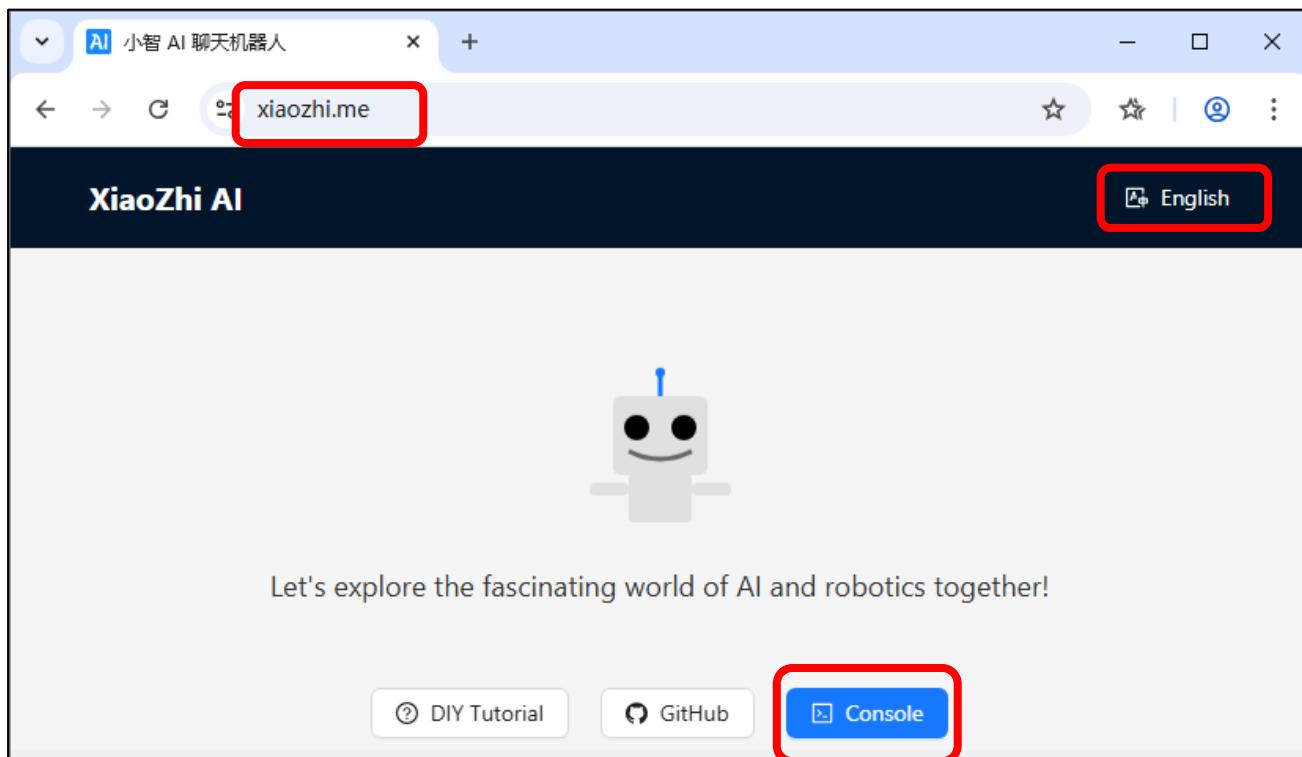


配置小智 AI 服务器

请确保您的手机或者电脑和 ESP32 S3 WROOM 连接到同一个路由器 WiFi 网络中。

打开手机或者电脑的浏览器功能，访问网址：<https://xiaozi.me/>

请注意，由于不同国家的网络政策原因，可能会存在部分国家的用户无法正常访问网站。具体情况请查看相关国家网络政策。

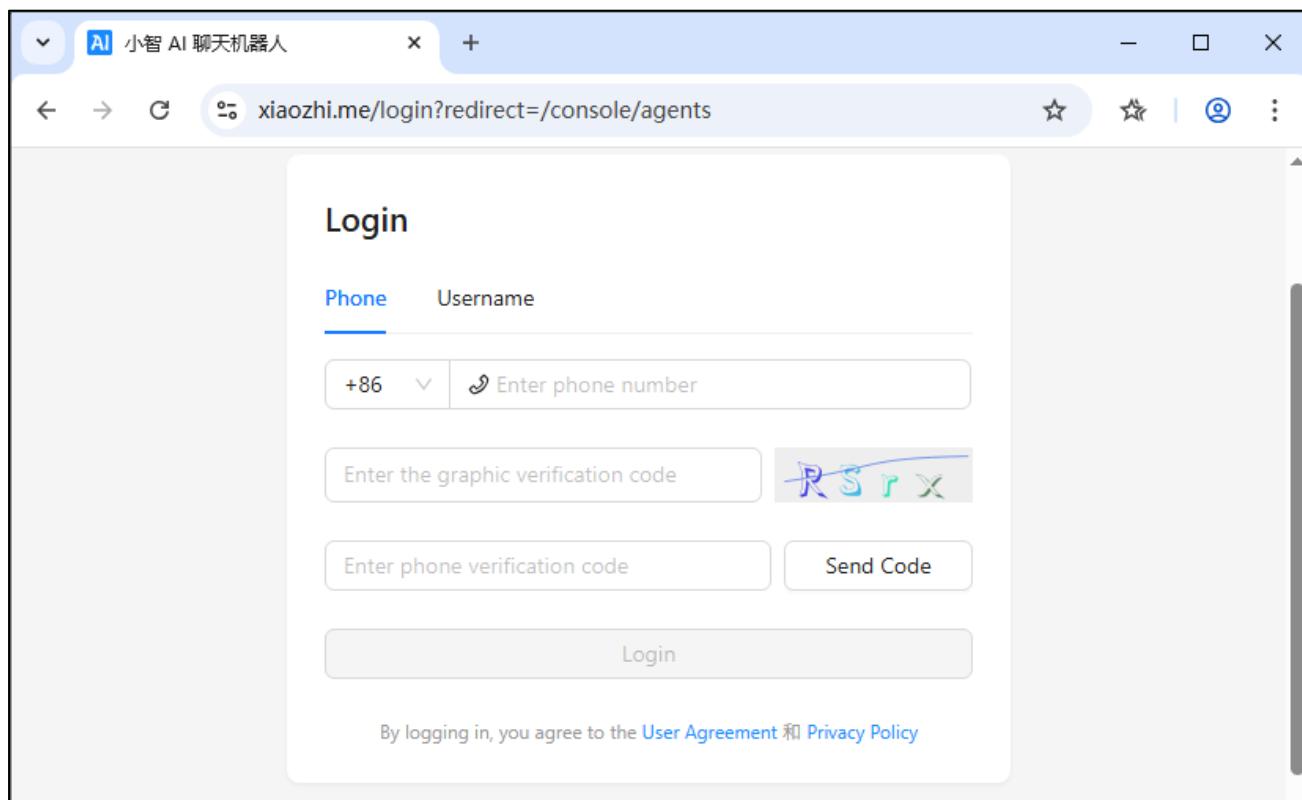


如果您还没有账号，请点击 Console，使用手机号进行注册。

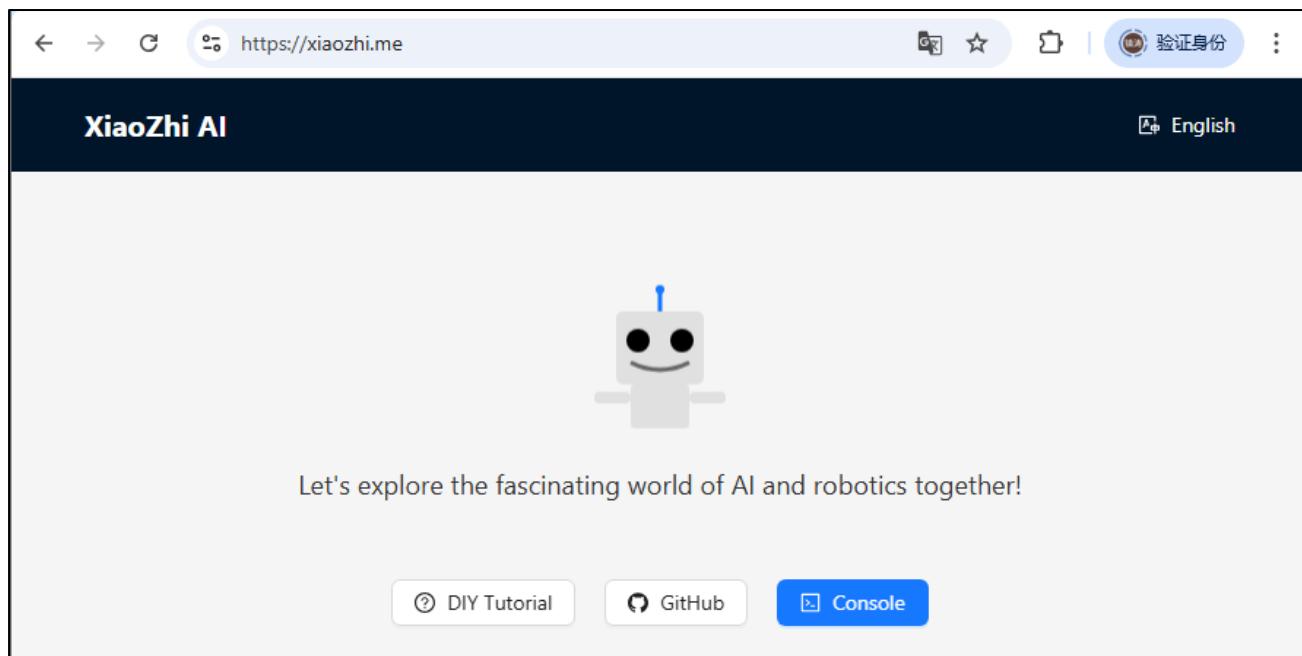
请注意，目前小智 AI 服务器仅支持以下国家使用手机号注册账号。

Mainland China +86	Germany +49	Philippines +63	Pakistan +92
Hong Kong +852	Pllland +48	New Zealand +64	Nigeria +234
Macau +853	Switzerland +41	Singapore +65	Bangladesh +880
Taiwan +886	Spain +34	Thailand +66	Saudi Arabia +966
United States/Canada +1	Denmark +45	Japan +81	United Arab Emirates +971
United Kingdom +44	Malaysia +60	South Korea +82	Brazil +55
France +33	Australia +61	Vietnam +84	Mexico +52
Italy +39	Indonesia +62	India +91	Chile +56
Argentina +54	Egypt +20	South Africa +27	Kenya +254
Tanzania +255	Kazakhstan +7		

如果您还没有账号，请先注册一个账号，并登陆它。

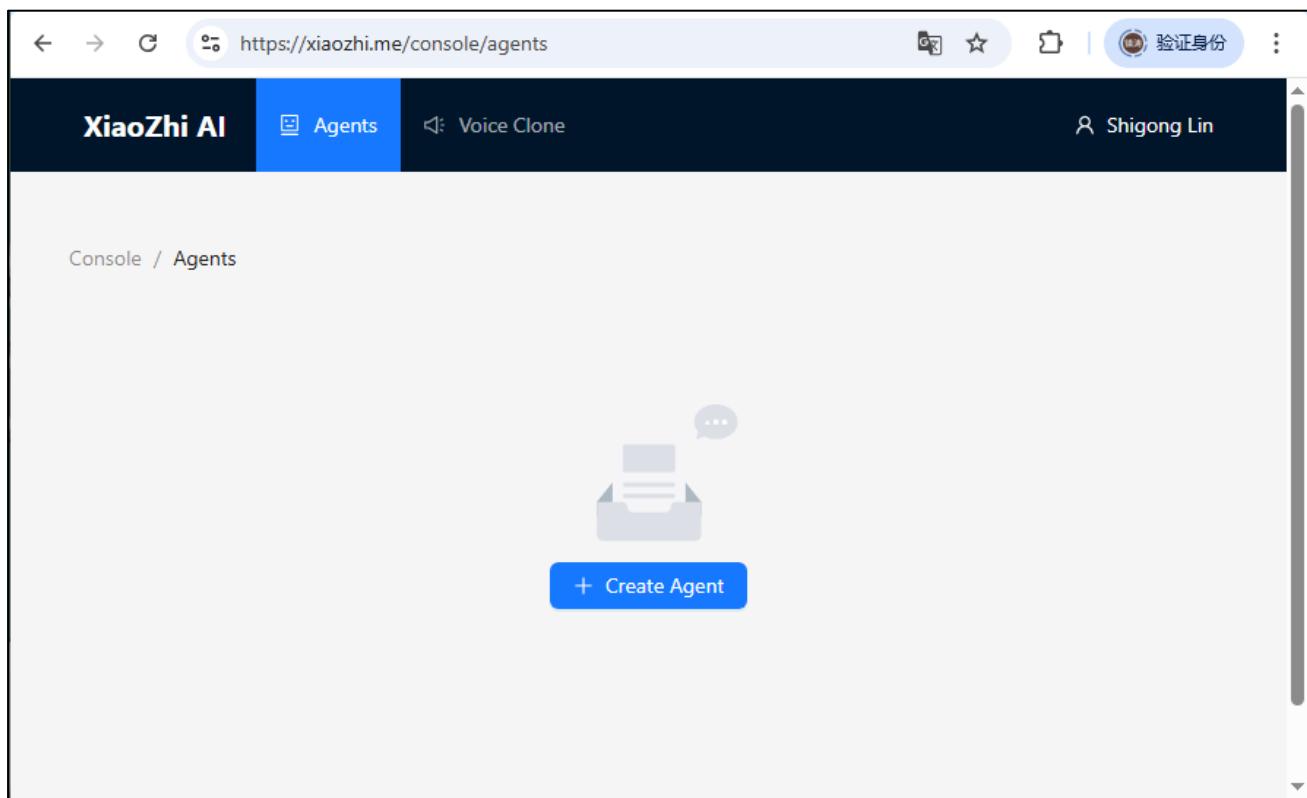


点击 Console，开始配置小智 AI 服务器。

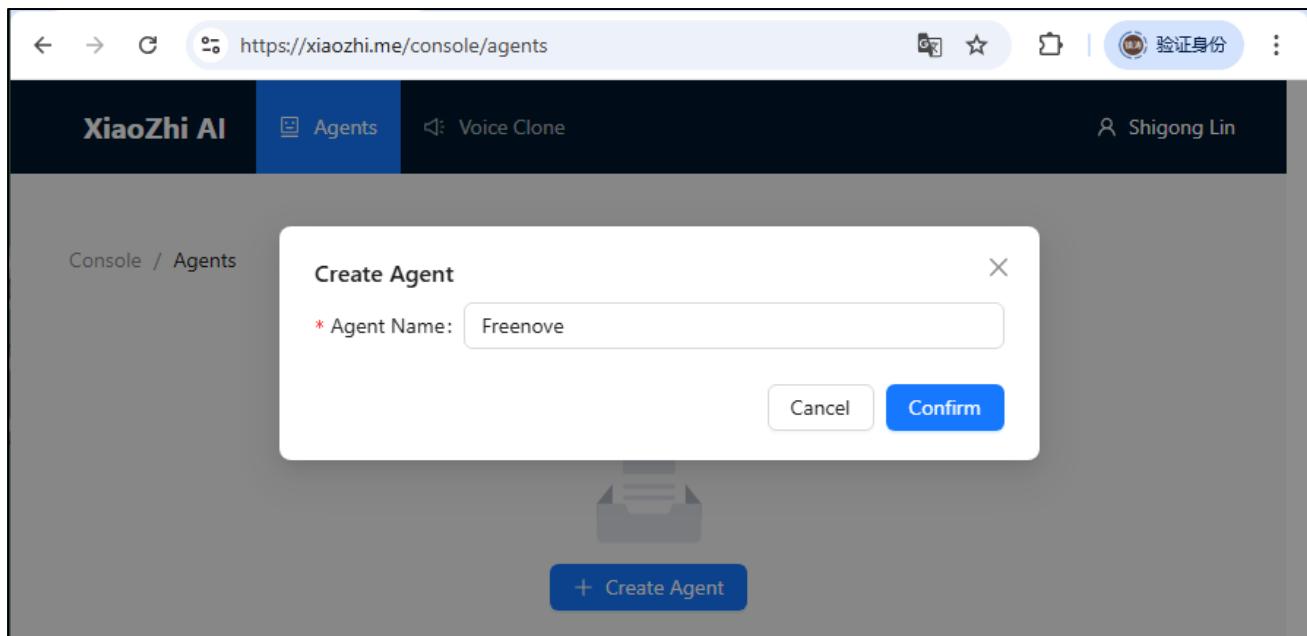




点击“Create Agent”，创建一个新的智能体。



填写任意名称。点击“Confirm”。



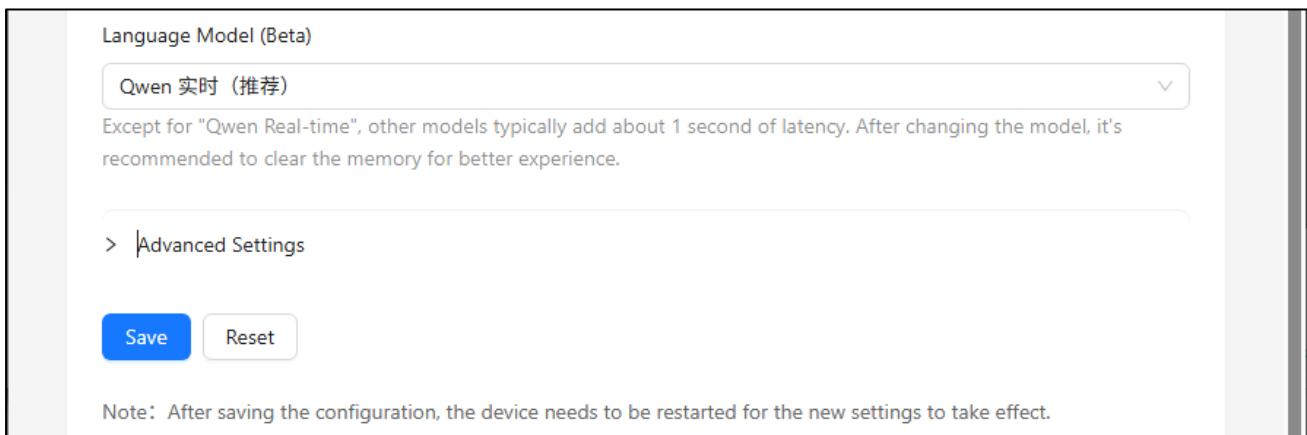
点击“Configure Role”，配置您的智能体。

The screenshot shows the XiaoZhi AI console interface. At the top, there is a navigation bar with the URL <https://xiaozi.me/console/agents>. The main header says "XiaoZhi AI" and "Agents". Below the header, it says "Voice Clone" and "Shigong Lin". On the left, there is a breadcrumb trail "Console / Agents" and a button "+ Create Agent". A modal window titled "Freenove" is open, displaying configuration details: "Voice Role: 湾湾小何", "Language Model: Qwen 实时 (推荐)", and "Recent Chat: None". Below these details are four buttons: "Configure Role" (disabled), "Speaker Recognition", "Chat History" (disabled), and "Add Device" (highlighted in blue).

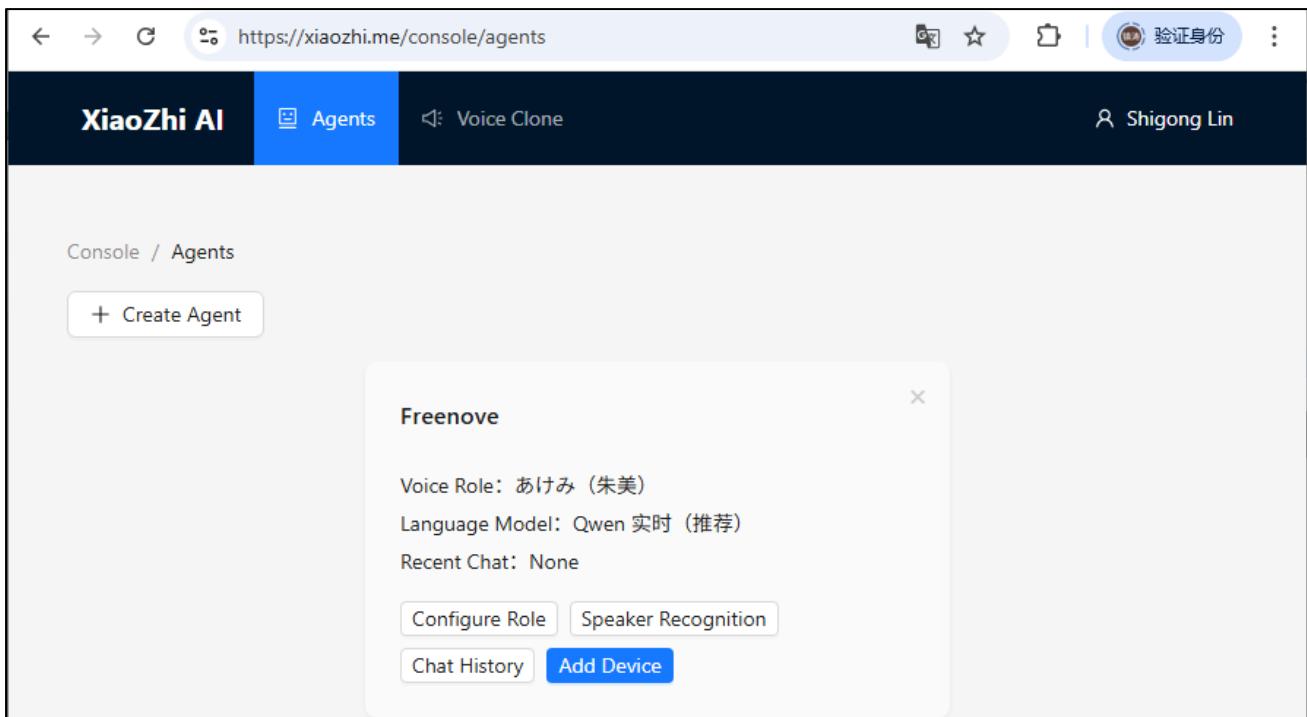
点击“English Tutor”，其他选项内容保持不变。

The screenshot shows the "Configure Role" page for the Freenove agent. The URL in the address bar is <https://xiaozi.me/console/agents/235590/config>. The header and navigation bar are identical to the previous screenshot. The breadcrumb trail now includes "Configure Role". The main content area is titled "Configure Role Freenove". Under "Role Template", there is a row of five buttons: "台湾女友", "土豆子", "English Tutor" (highlighted in blue), "好奇小男孩", and "汪汪队队长".

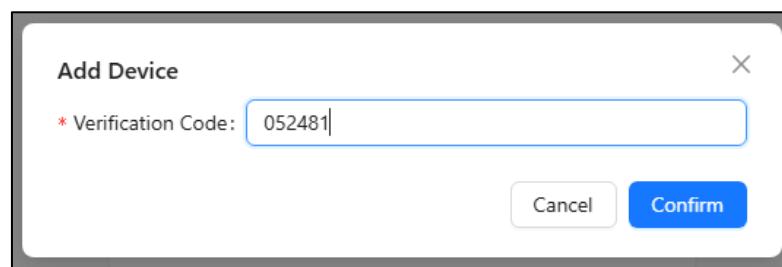
滚动鼠标，将界面移动到最底部，点击“Save”。



点击“Agents”回到智能体界面，然后点击“Add Device”。



在新弹出的界面中，填写 ESP32 S3 WROOM 屏幕的数字。然后点击“Confirm”。



添加完成后，界面如下。

Console / Agents

+ Create Agent

Freenove

Voice Role: あけみ (朱美)
Language Model: Qwen 实时 (推荐)
Recent Chat: 未知

Configure Role Speaker Recognition
Chat History Manage Devices (1)



点击 ESP32 S3 WROOM 上的 RST 按键。它将让 ESP32 S3 WROOM 重新启动。



至此，您已经完成小智 AI 的全部工作。对着麦克风说，“Hi, ESP”。



您可以和它进行中文、英文交流。

小智 AI 代码

Visual Studio Code

Window

首先，我们需要先下载 Visual Studio Code，请访问 <https://code.visualstudio.com/Download>。选择适合您电脑平台的版本，下载并安装它。

Visual Studio Code

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



[Windows](#)
Windows 10, 11

[.deb](#)
Debian, Ubuntu

[.rpm](#)
Red Hat, Fedora, SUSE

[Mac](#)
macOS 10.15+

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64

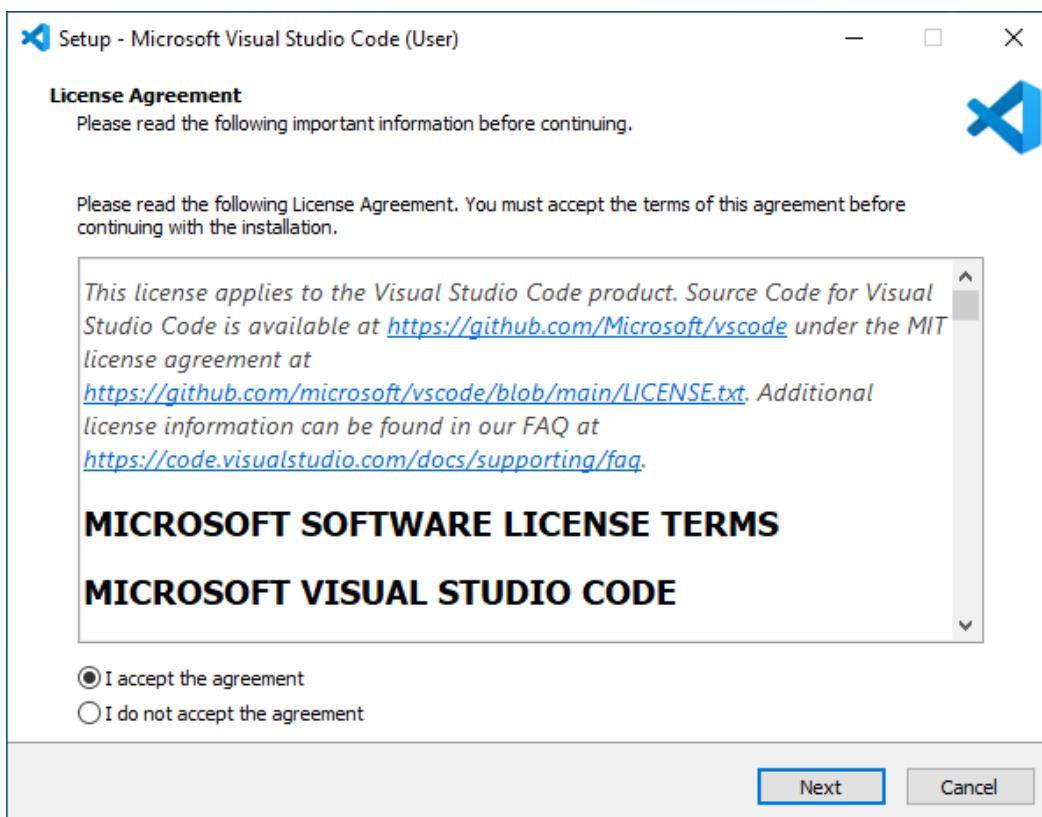
.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

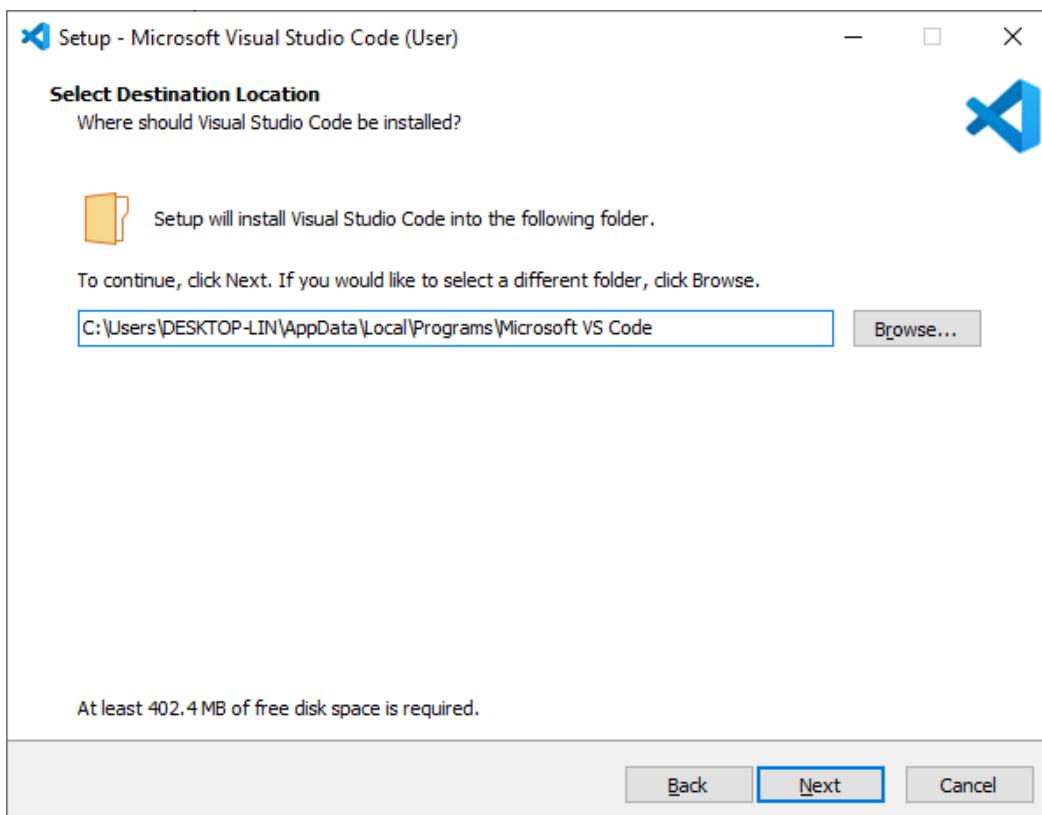
By downloading and using Visual Studio Code, you agree to the [license terms](#) and [privacy statement](#).



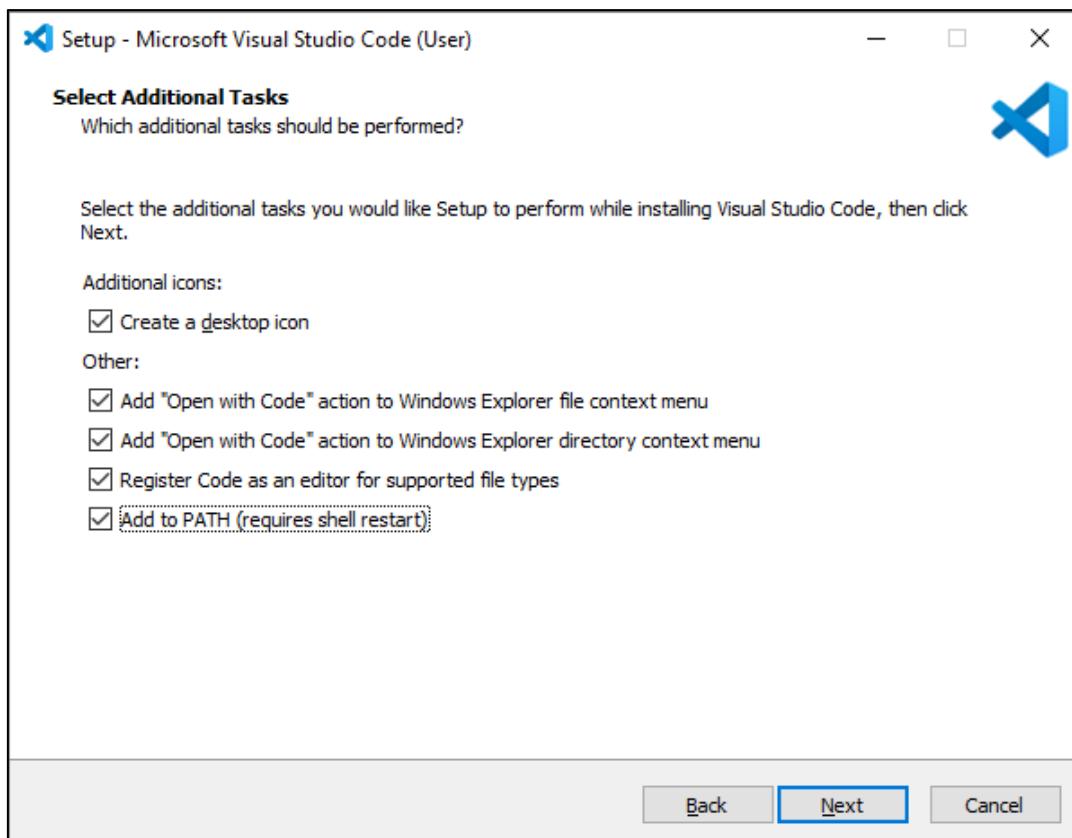
双击运行下载的 exe 文件。勾选 "I accept the agreement"。然后点击 next.



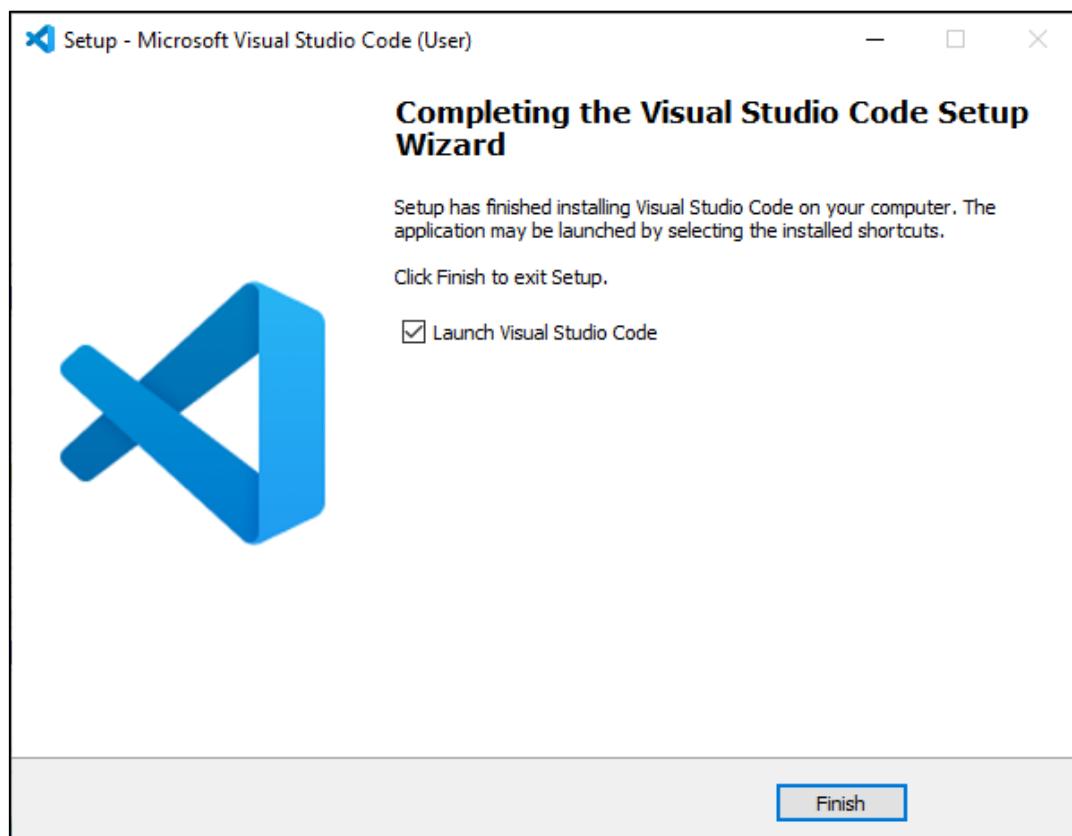
安装的位置可以保持默认选项，也可以自行修改到合适的位置。然后一直点击 Next。



在这个界面时，请注意查看“Add to PATH”是否勾选中，如果没有，请勾选它。然后一直点击 Next，直到软件安装完成。

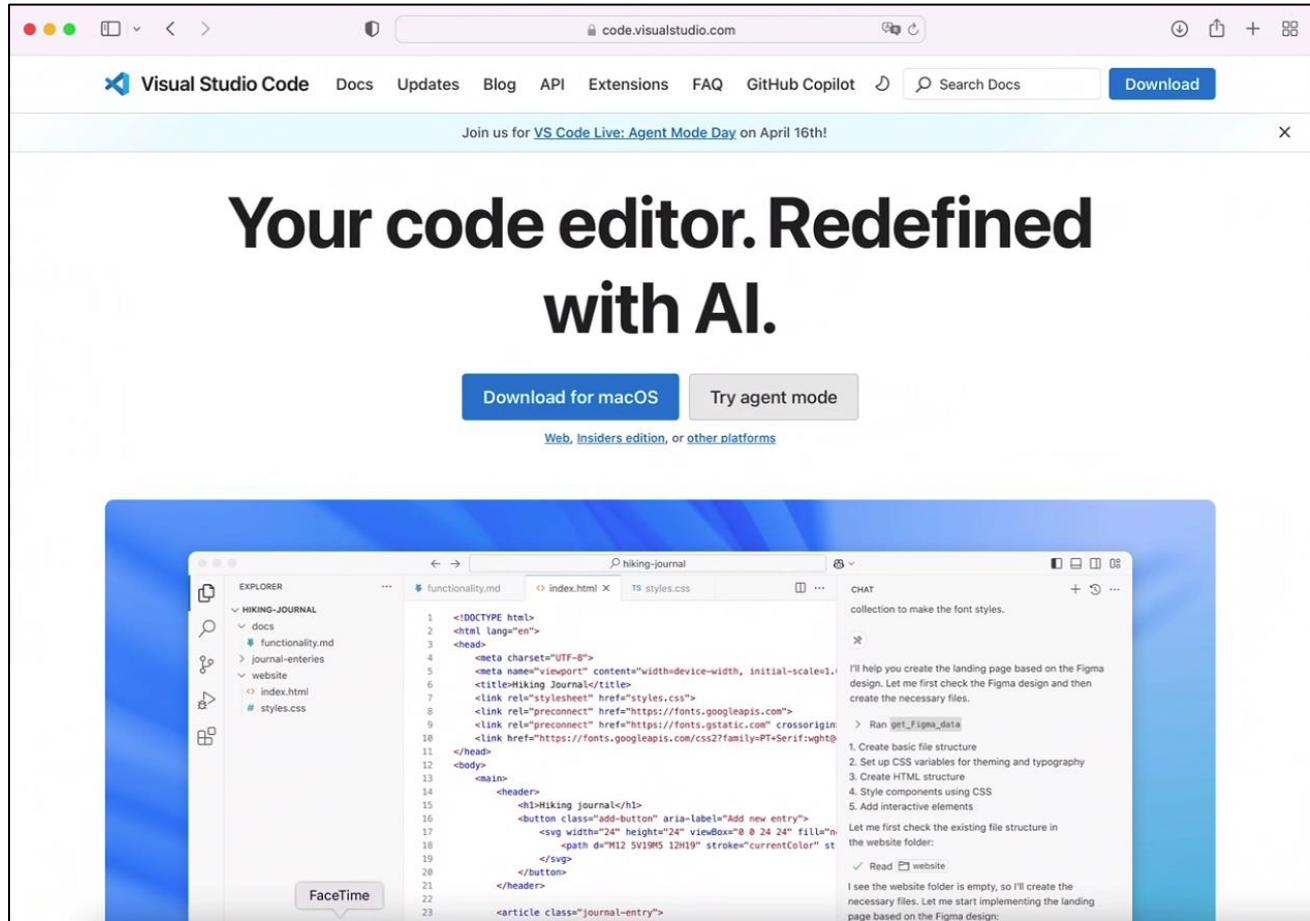


安装完成如下图所示。

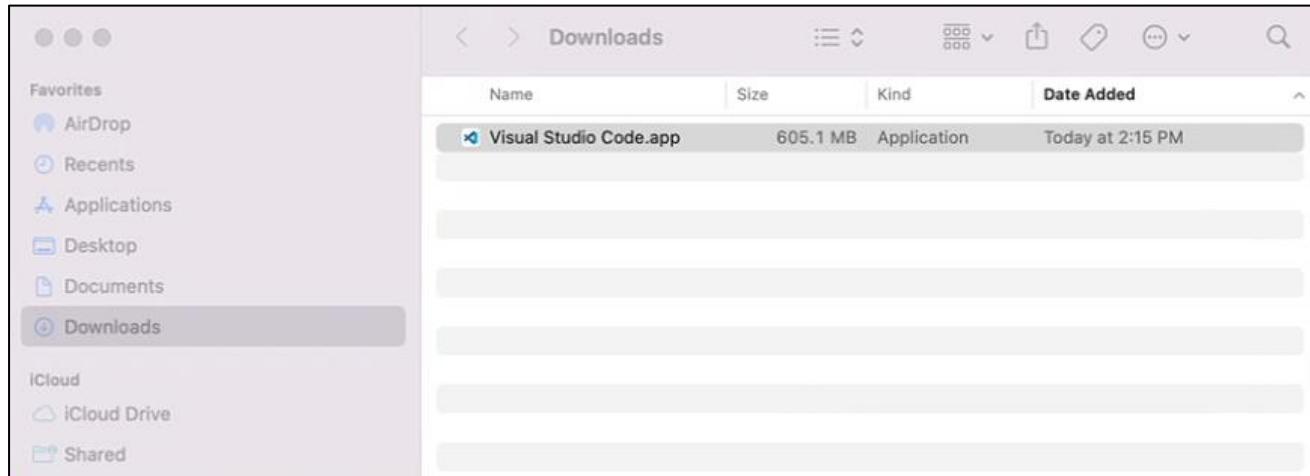


MAC

通常，MAC 系统自带 Visual Studio Code，如果您的电脑没有 Visual Studio Code，请先安装它。请访问 <https://code.visualstudio.com> 并点击“Download for macOS”。

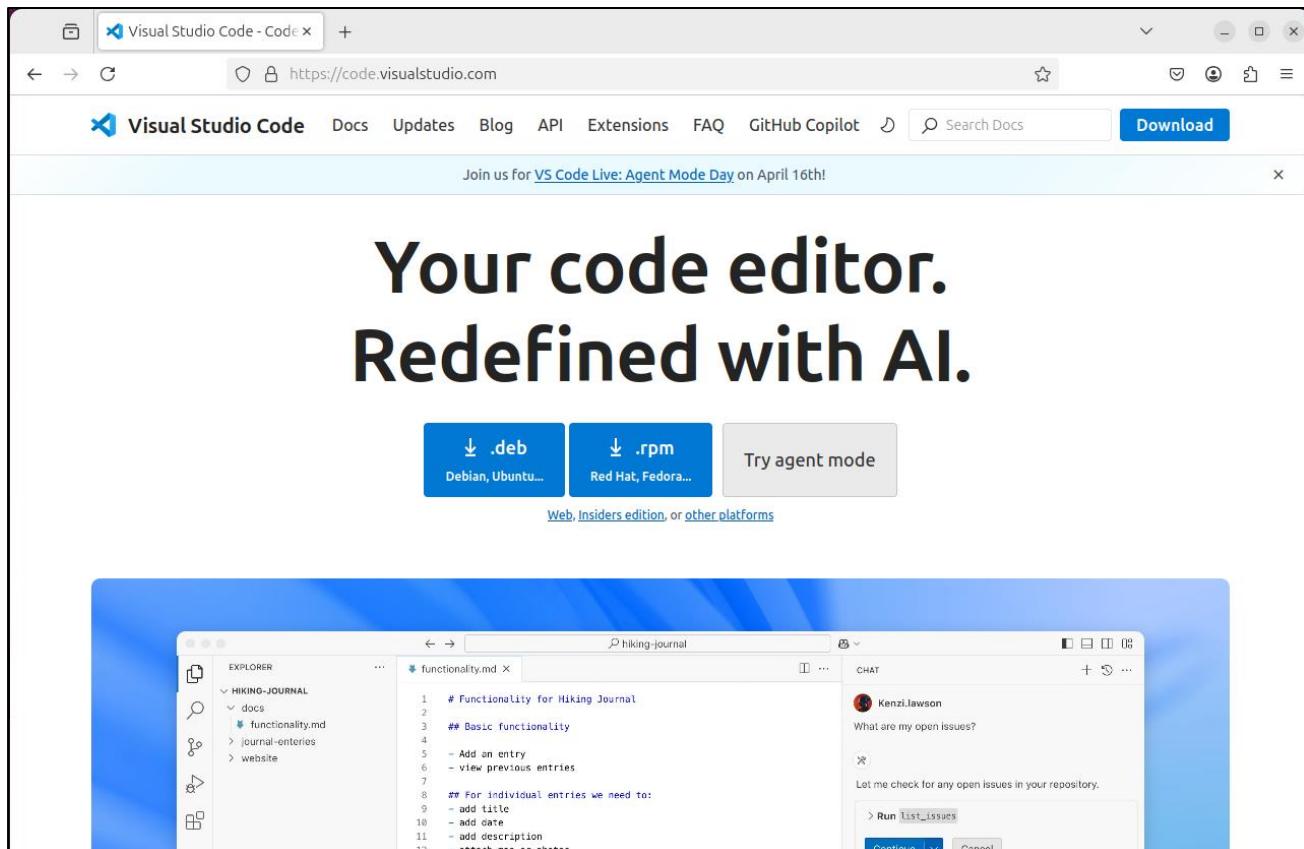


双击运行它。

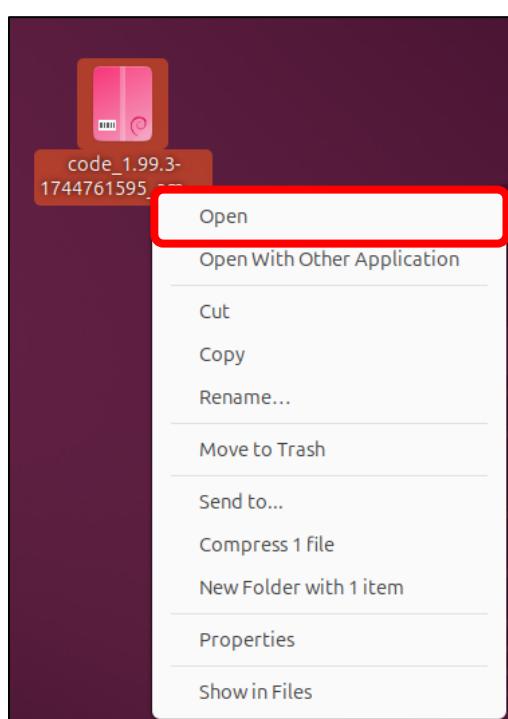


Linux

如果您的电脑没有 Visual Studio Code，请先安装它。
请访问 <https://code.visualstudio.com>。并点击“.deb”。

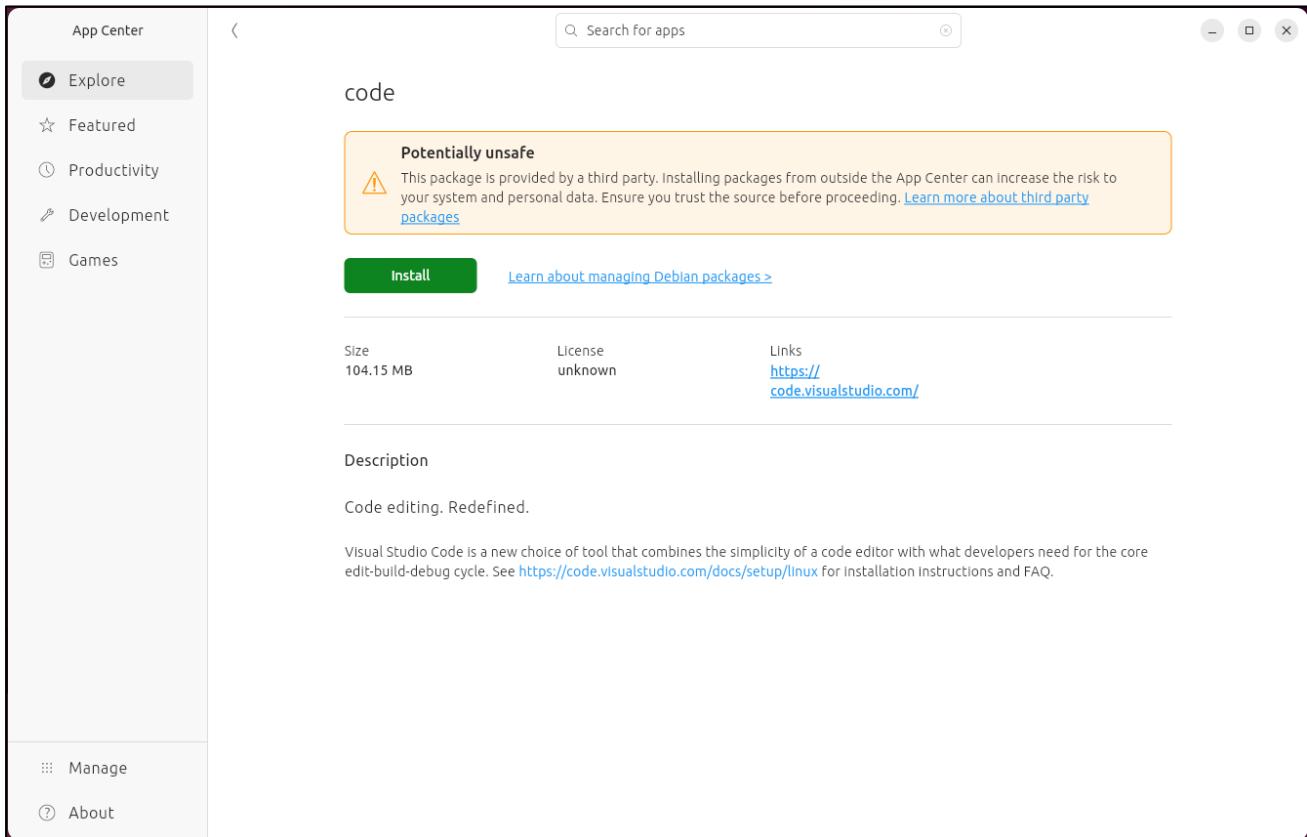


打开下载的“code_xxx.deb”文件。

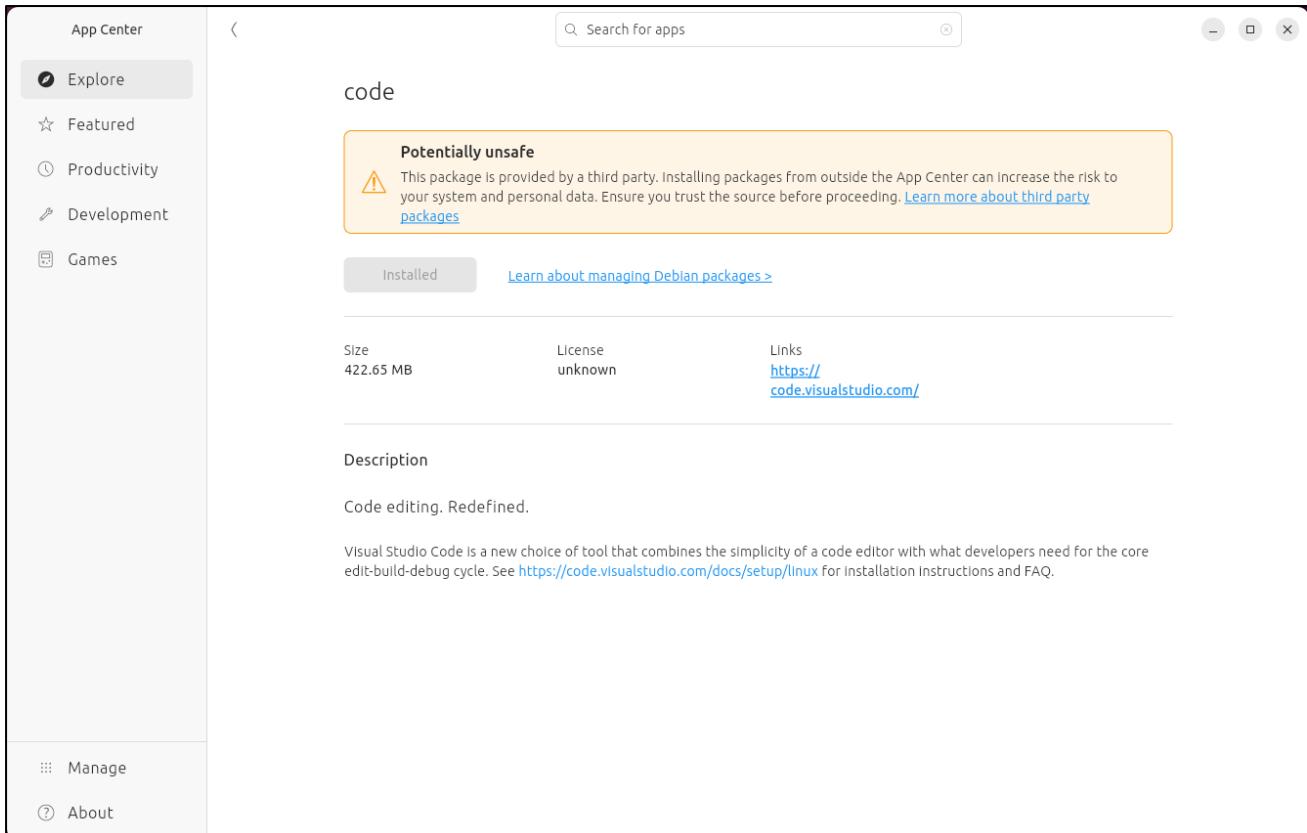




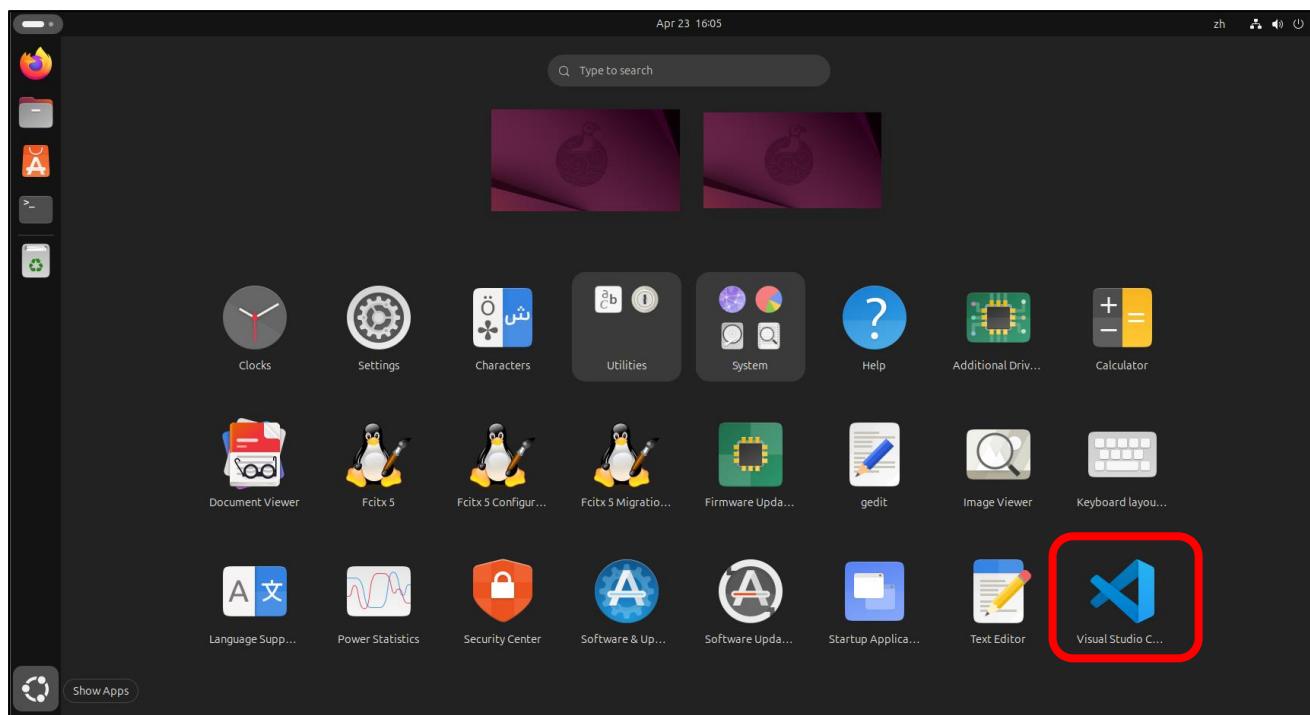
点击“Install”，安装 Visual Studio Code.



等待安装完成，安装完成如下图所示。



点击 Show Apps，可以看到，Visual Studio Code 已经出现在系统中。

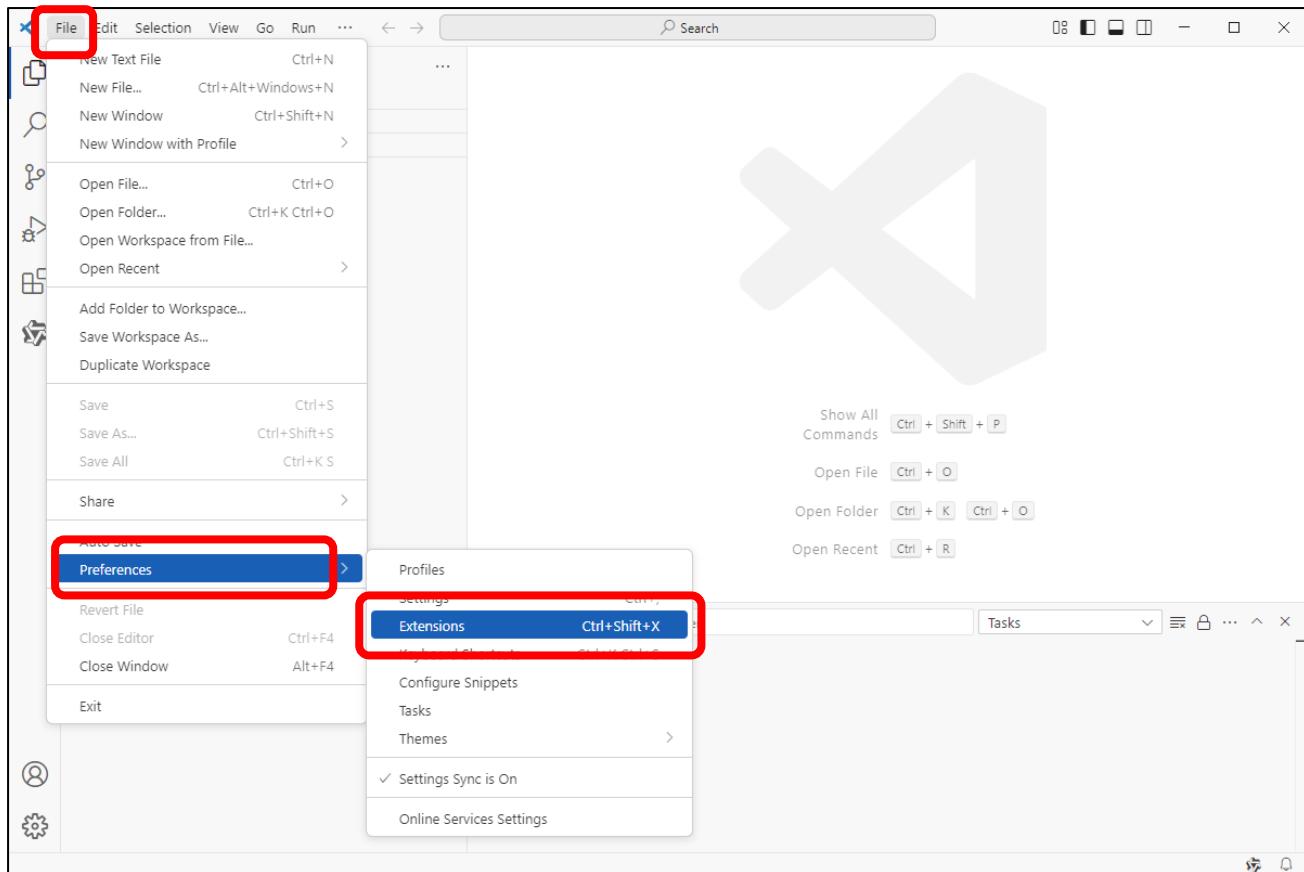


安装 ESP-IDF V5.3.2

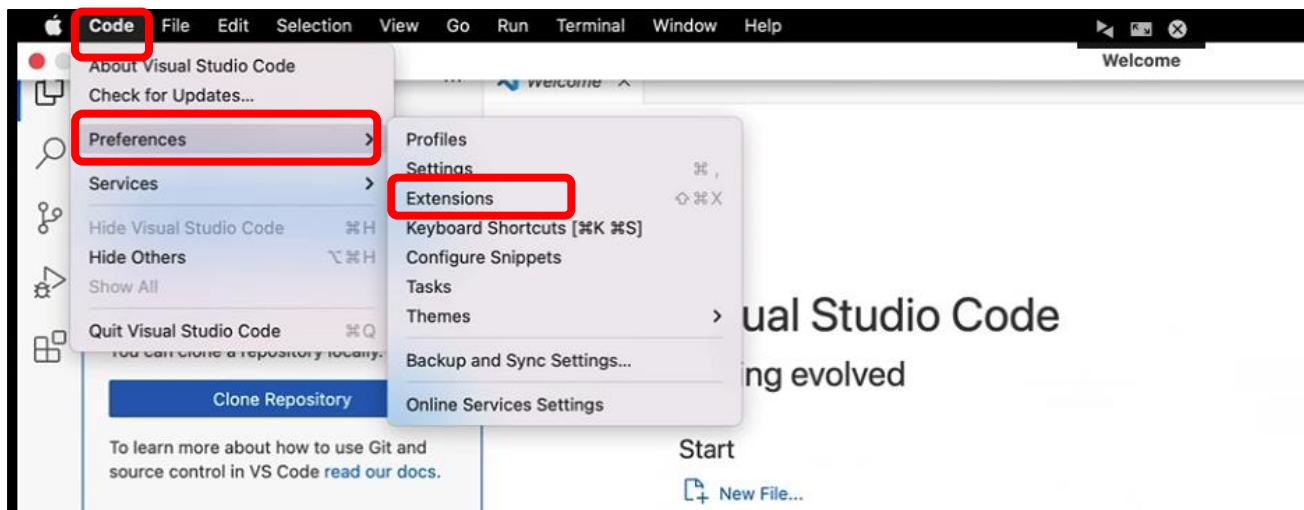
Visual Studio Code 是一个通用的代码编辑器，为了编程 ESP-IDF 的 SDK，我们需要给 Visual Studio Code 安装一个 ESP-IDF 插件。

打开 Visual Studio Code。

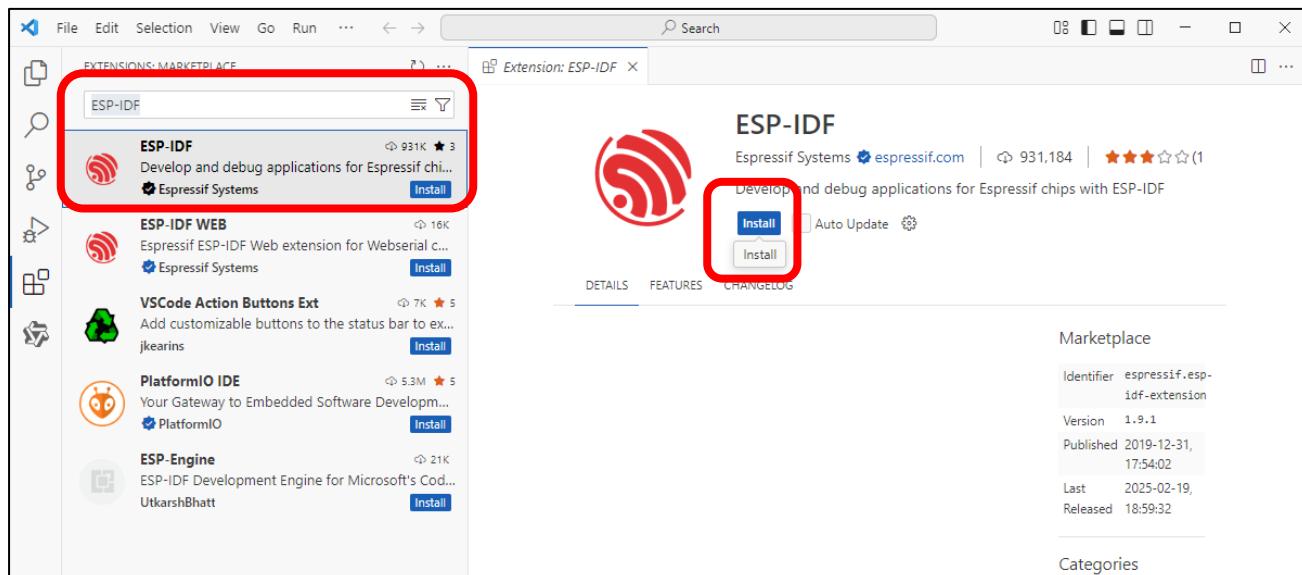
Window 系统和 Linux 系统：点击菜单栏的“File”->“Preferences”->“Extensions”。



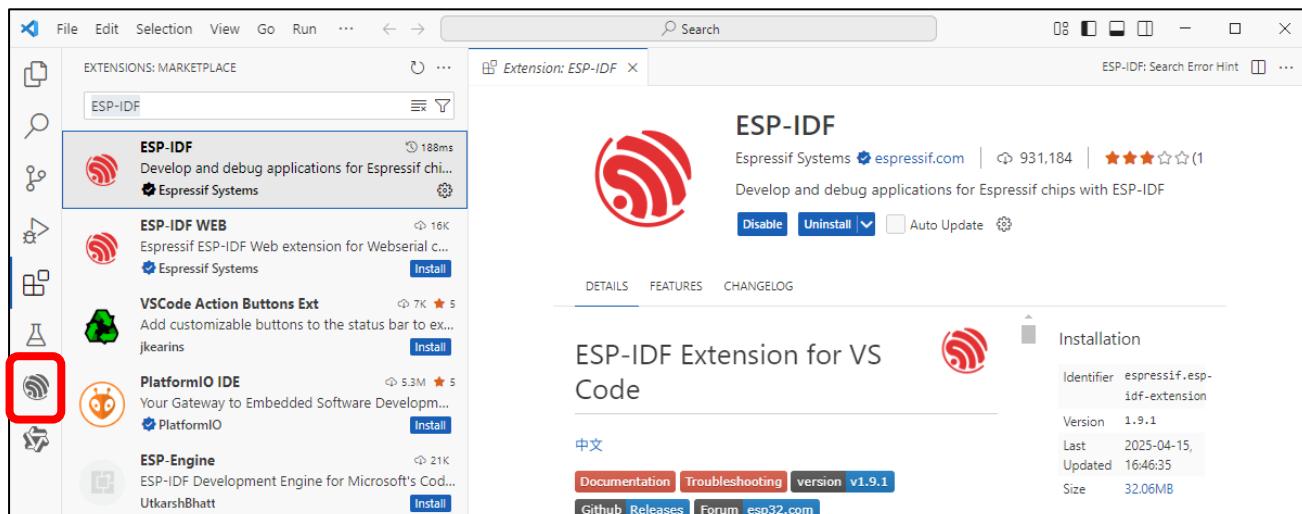
Mac 系统：点击屏幕左上方的“Code”->“Preferences”->“Extensions”。



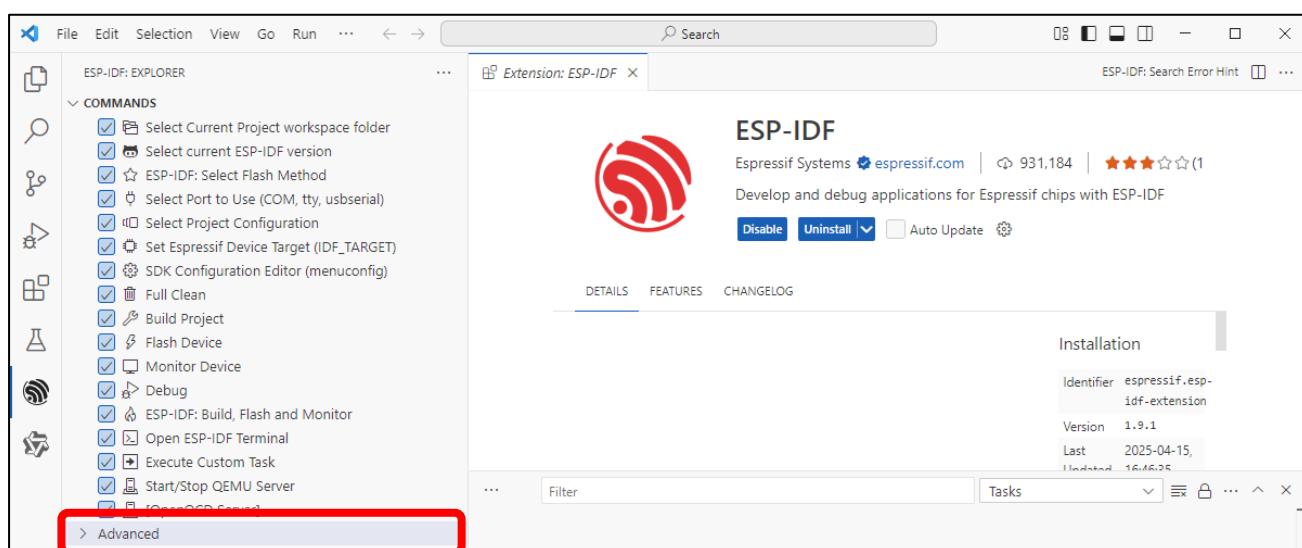
输入“ESP-IDF”，点击搜索出来的“ESP-IDF”，然后点击右边的“Install”。



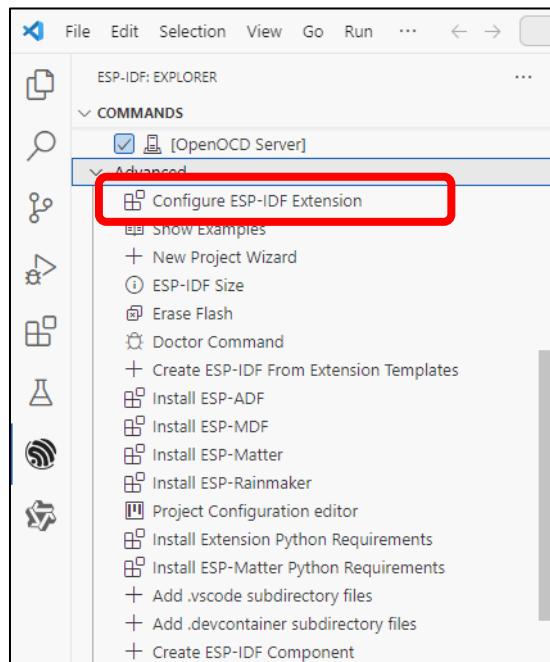
在左边可以看到出现了一个“ESP-IDF”的图标。点击它。



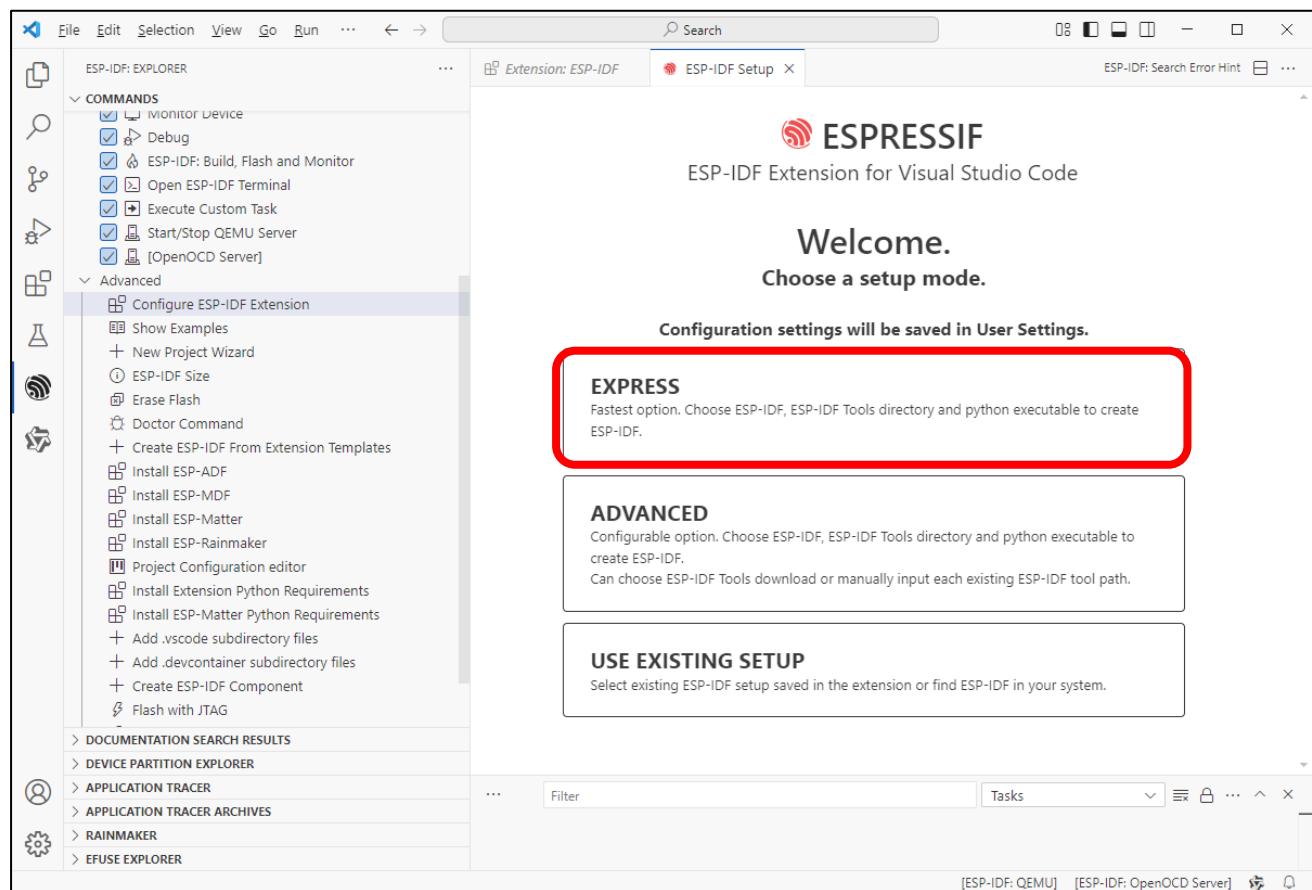
滚动鼠标，找到并点击“Advanced”。



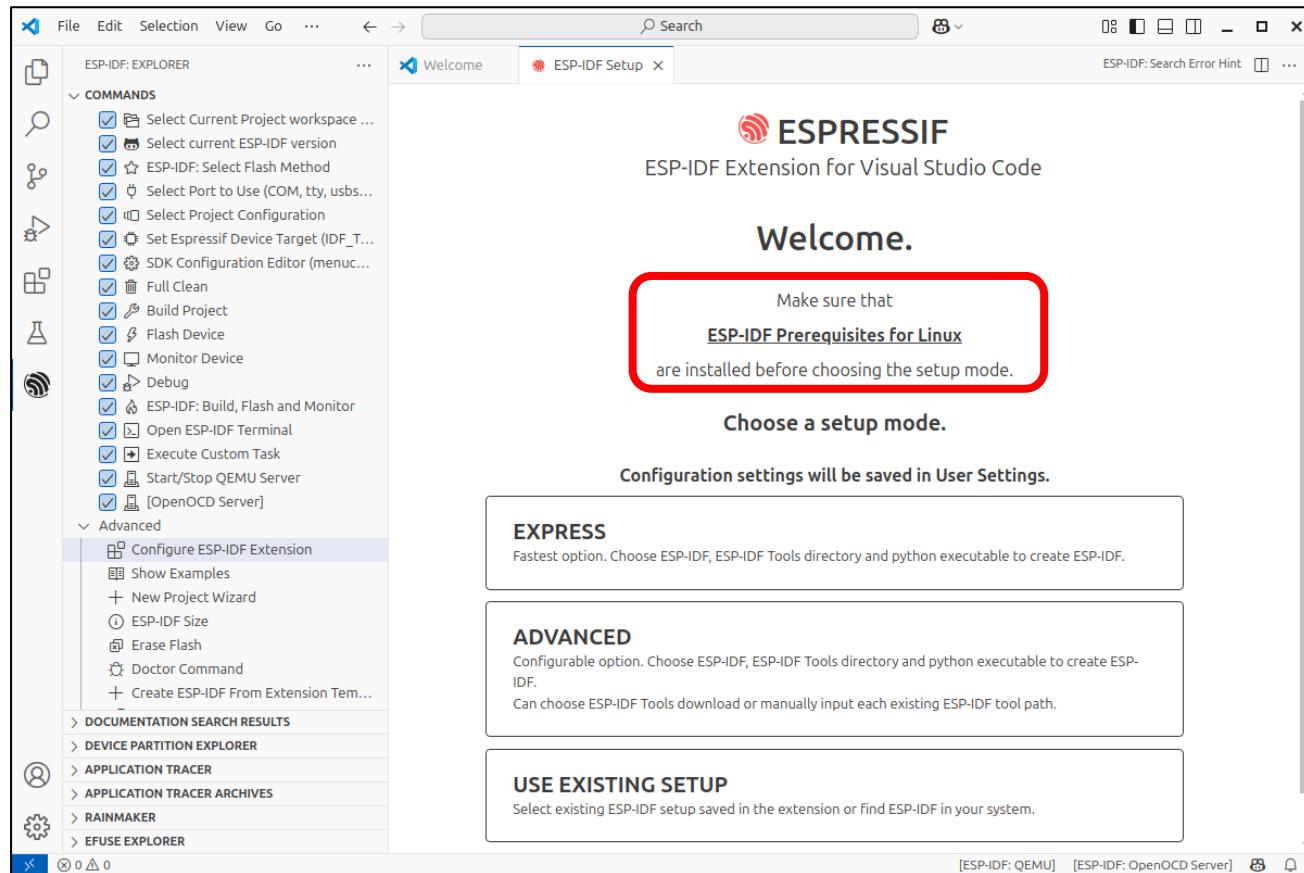
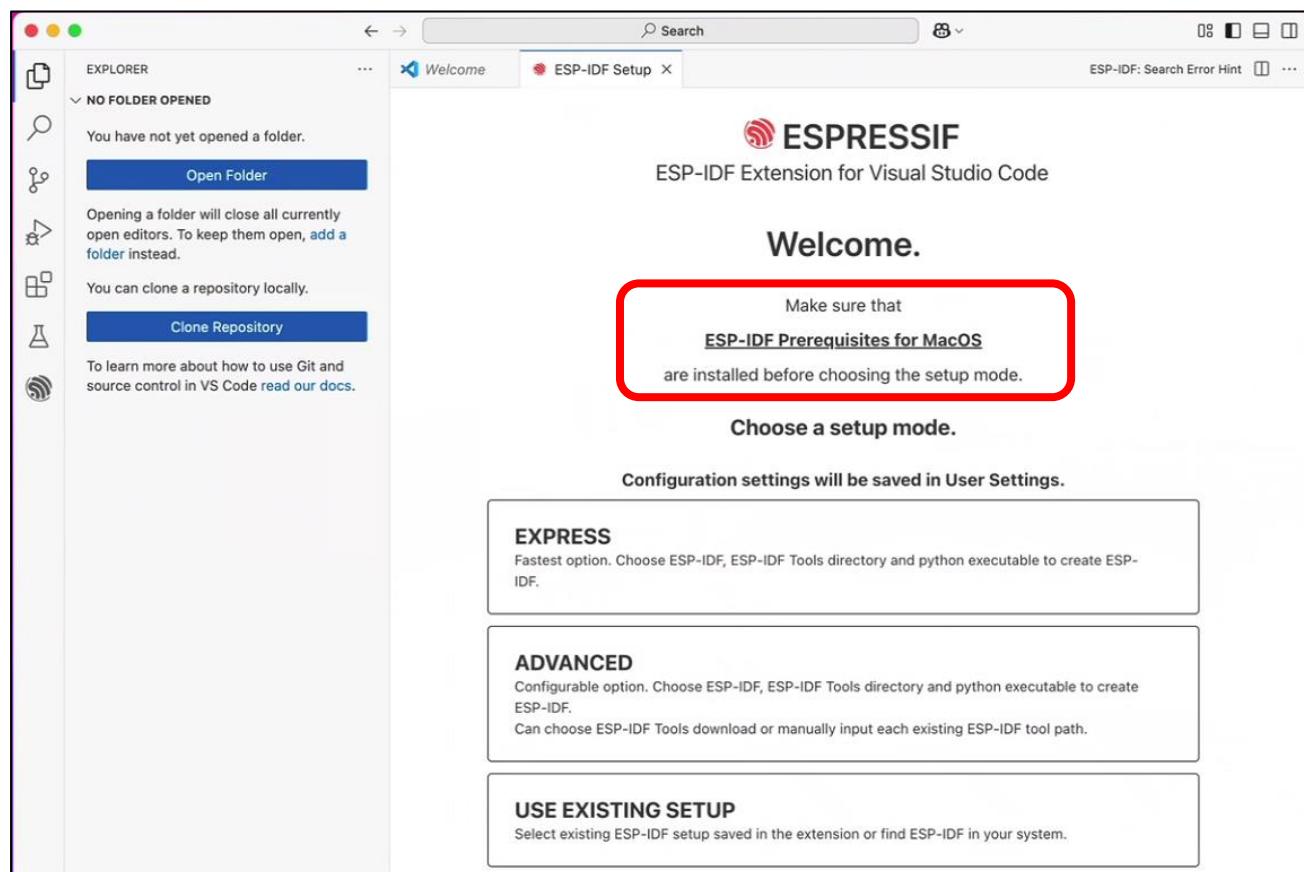
点击第一个选项，“Configure ESP-IDF Extension”。



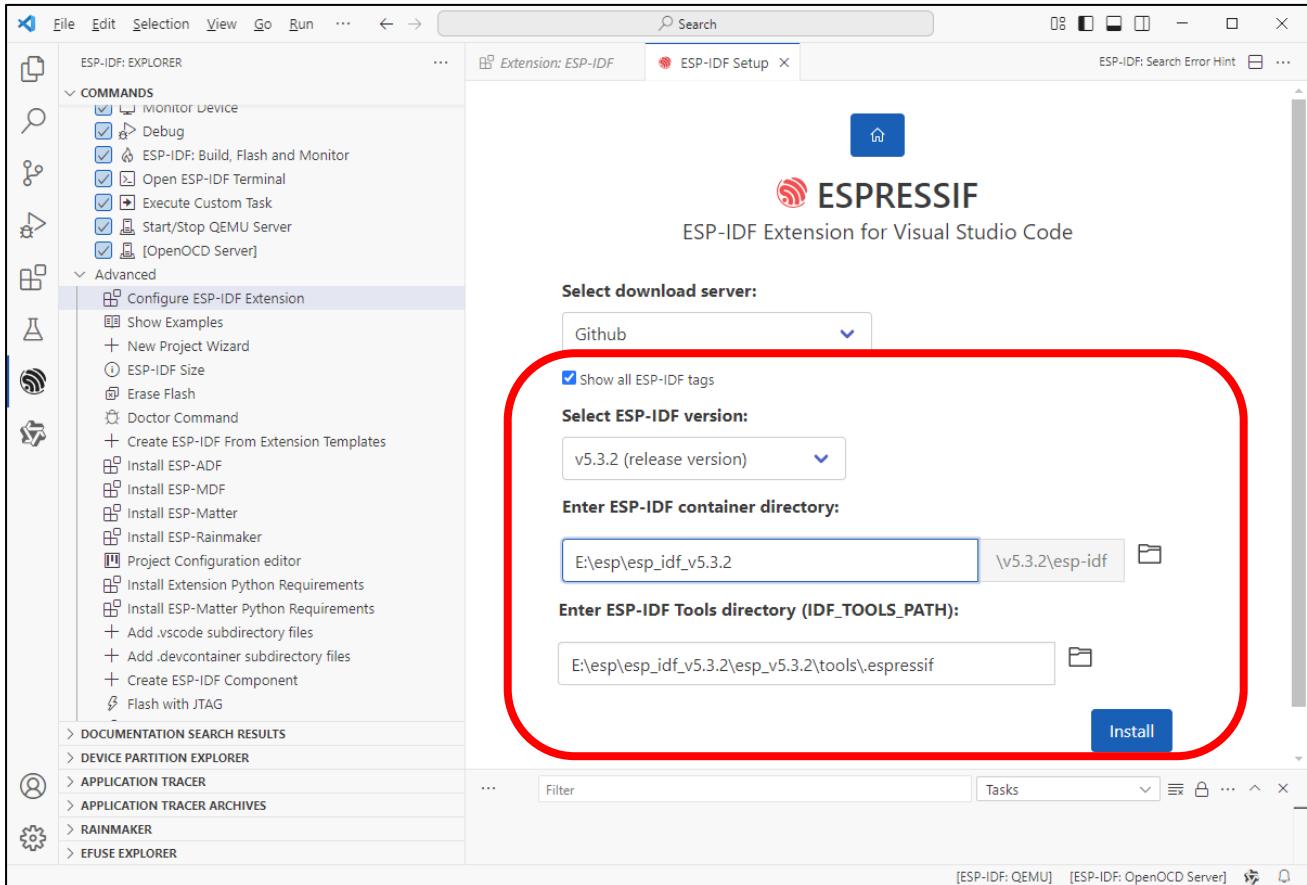
在右边选择“EXPRESS”。



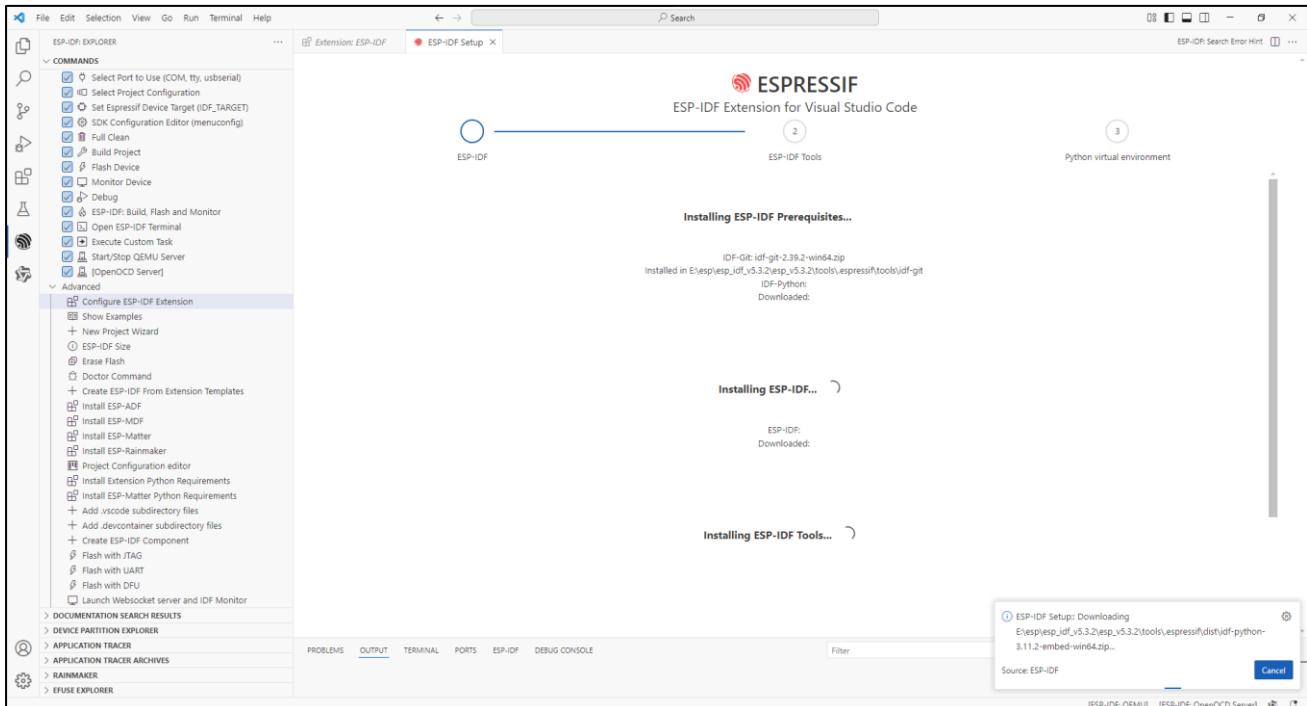
注意：如果您是 MAC 系统或者 Ubuntu 系统，请根据提示做好准备工作再进行安装。



勾选“Show all ESP-IDF tags”，选择“v5.3.2 (release version)”，并选择您的 ESP-IDF 环境将要保存的位置。然后点击“Install”。不同的系统保存的路径不同，请记住这些存放路径。



等待安装完成即可。如果您安装失败了，请找到上一步中的路径，删除对应的文件夹，并重新开始安装。



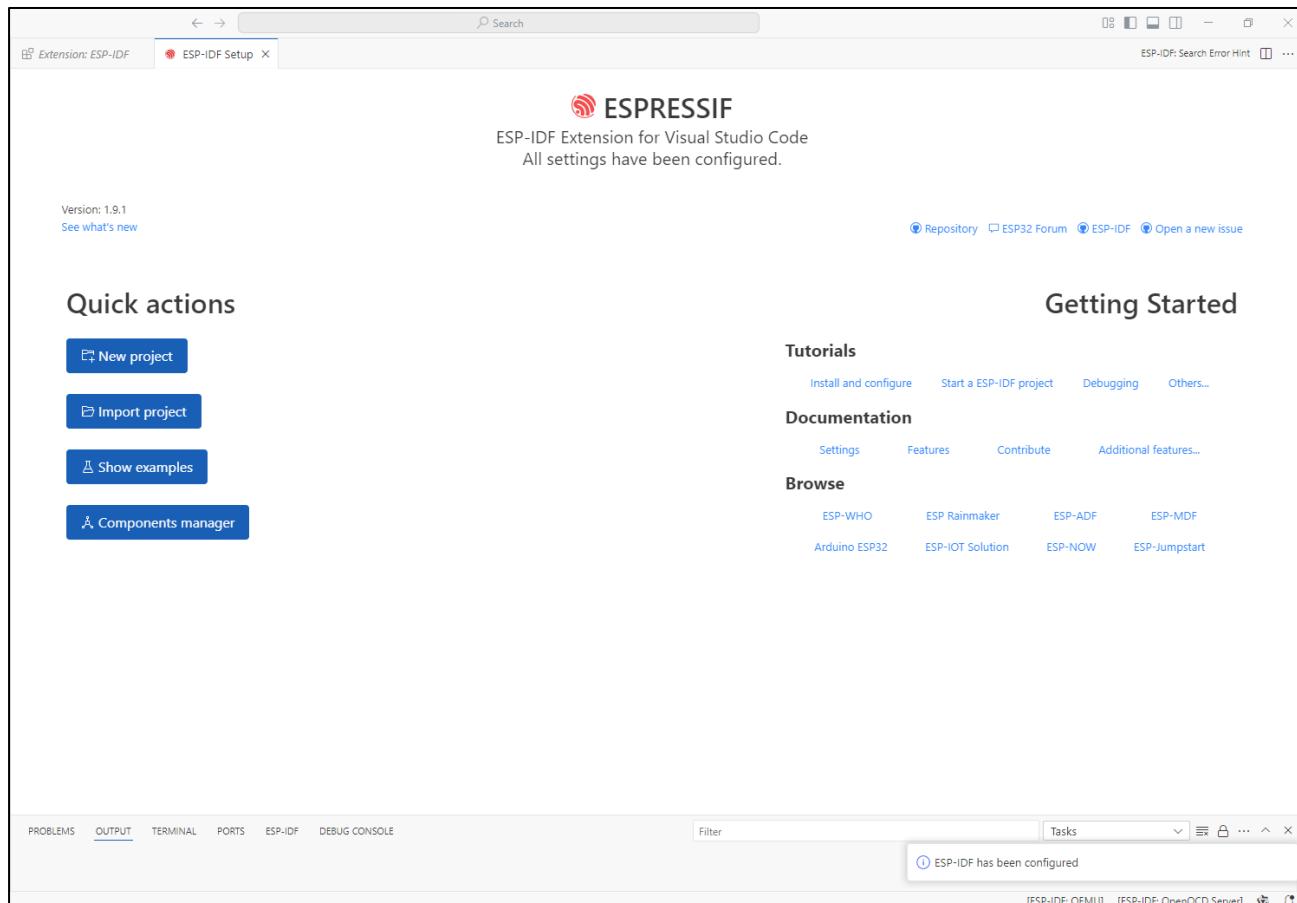
这一步需要等待一段较长的时间，请确保您的网络良好，且网速较快。

如果您始终无法正常安装，根据您的电脑系统，查看对应的链接：

Window: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/windows-setup.html>

Mac & Linux: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/linux-macos-setup.html>

插件安装完成如下图所示。



更多关于 ESP-IDF 的消息，请访问：

<https://docs.espressif.com/projects/vscode-esp-idf-extension/en/latest/installation.html>



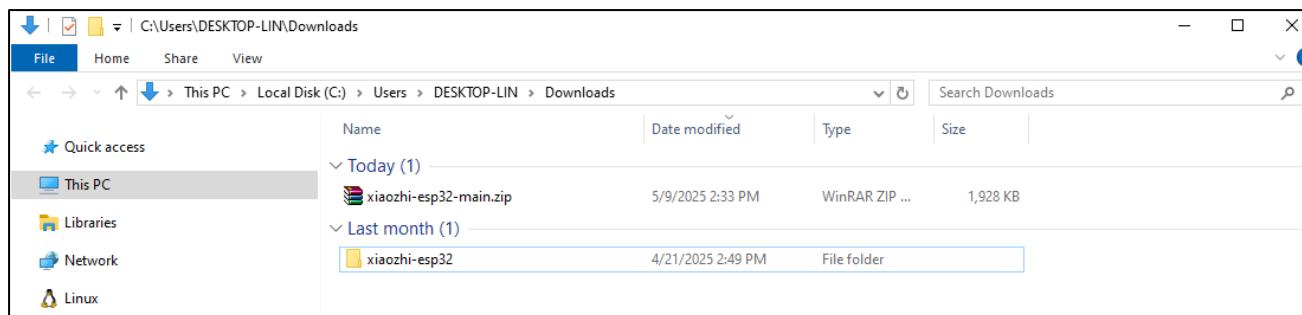
下载代码

Window

打开浏览器，输入“<https://github.com/Freenove/xiaozhi-esp32>”。

点击“Code”，然后点击“Download ZIP”，将代码下载到本地。

将下载的压缩文件解压到本地。请注意，将解压后的文件夹重命名为“xiaozihi-esp32”。



MAC

打开终端，使用 git 下载代码。

```
git clone https://github.com/Freenove/xiaozihi-esp32.git
```

```
[freenove@PandeMacBook-Air ~ % git clone https://github.com/Freenove/xiaozihi-esp32.git
Cloning into 'xiaozihi-esp32'...
remote: Enumerating objects: 4596, done.
remote: Counting objects: 100% (1285/1285), done.
remote: Compressing objects: 100% (167/167), done.
remote: Total 4596 (delta 1181), reused 1118 (delta 1118), pack-reused 3311 (from 1)
Receiving objects: 100% (4596/4596), 3.19 MiB | 4.63 MiB/s, done.
Resolving deltas: 100% (3189/3189), done.
freenove@PandeMacBook-Air ~ %
```

Linux

打开终端，使用 git 下载代码。

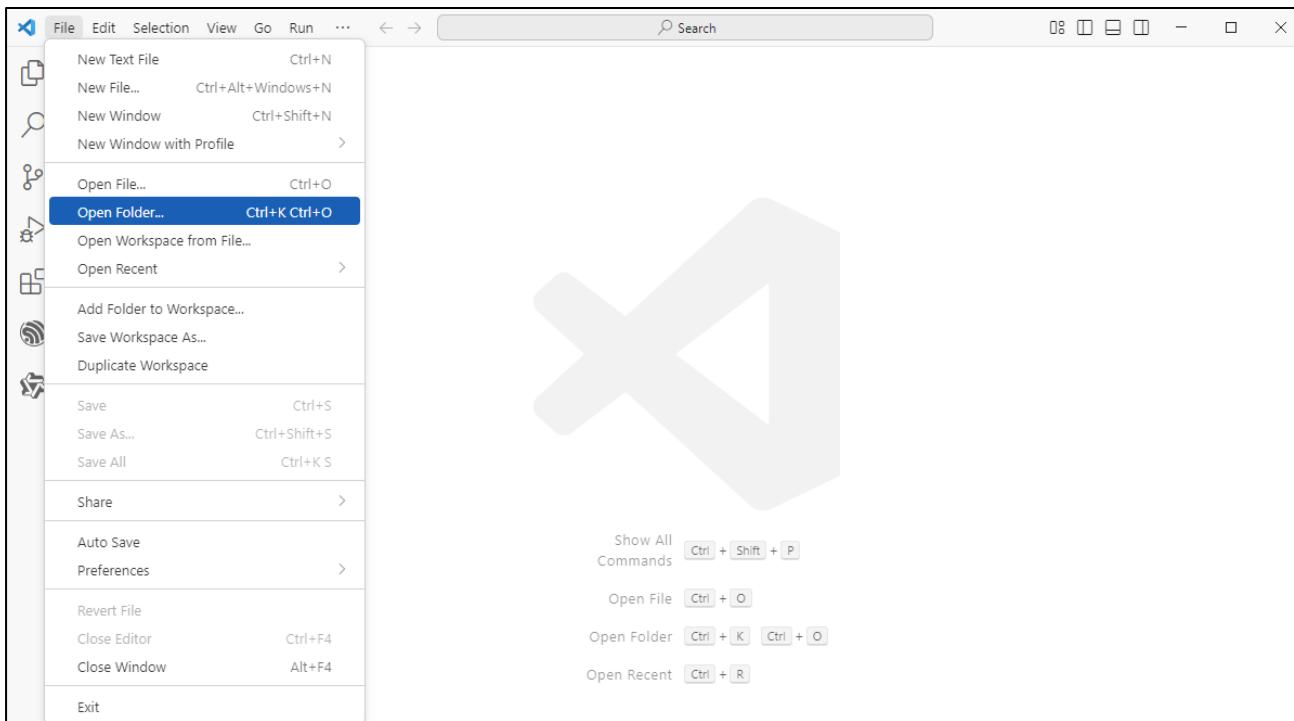
```
git clone https://github.com/Freenove/xiaozihi-esp32.git
```

```
lin@ubuntu:~$ git clone https://github.com/Freenove/xiaozihi-esp32.git
Cloning into 'xiaozihi-esp32'...
remote: Enumerating objects: 4596, done.
remote: Counting objects: 100% (1285/1285), done.
remote: Compressing objects: 100% (167/167), done.
remote: Total 4596 (delta 1181), reused 1118 (delta 1118), pack-reused 3311 (from 1)
Receiving objects: 100% (4596/4596), 3.19 MiB | 4.84 MiB/s, done.
Resolving deltas: 100% (3189/3189), done.
lin@ubuntu:~$
```

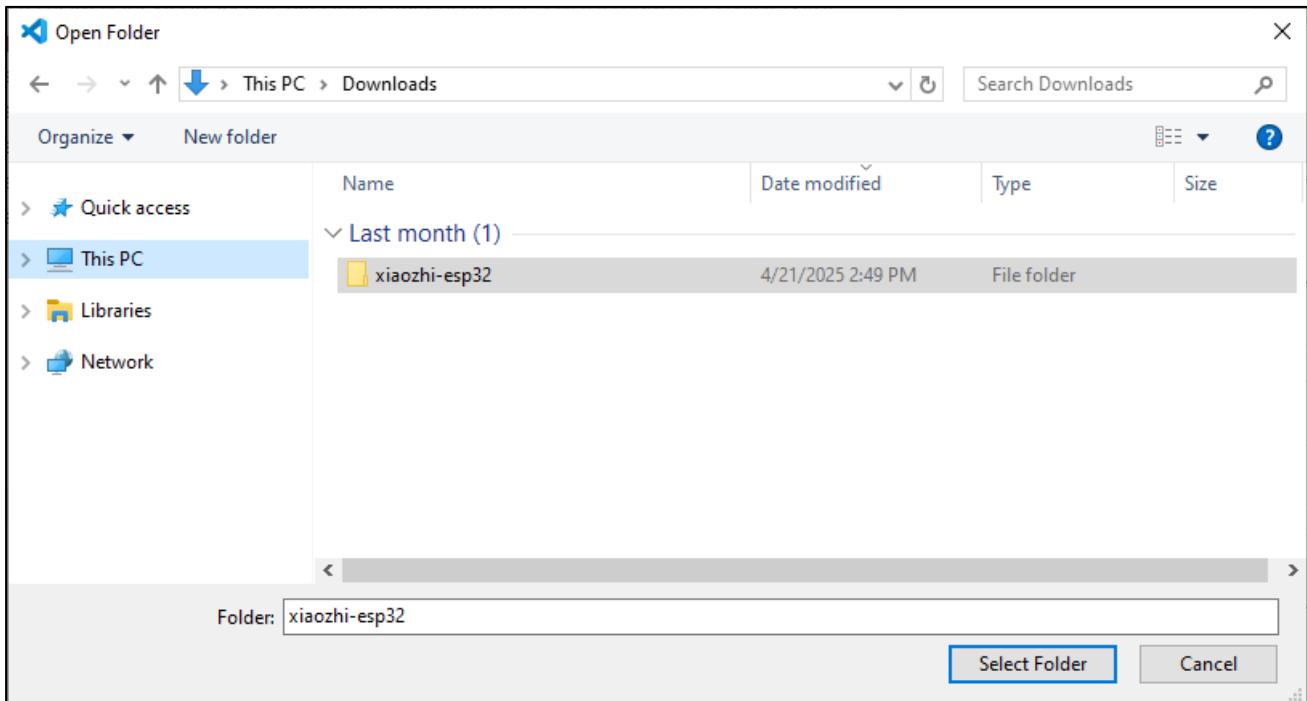


配置代码环境

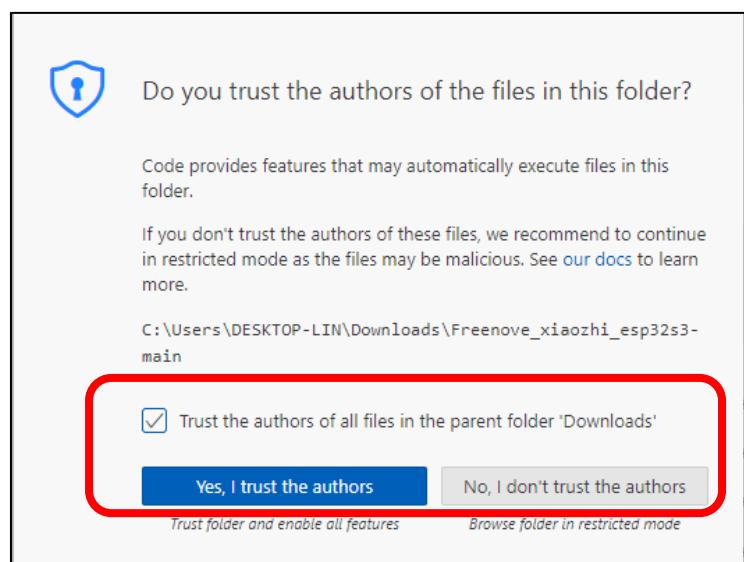
使用 Visual Studio Code，点击“File”->“Open Folder...”。



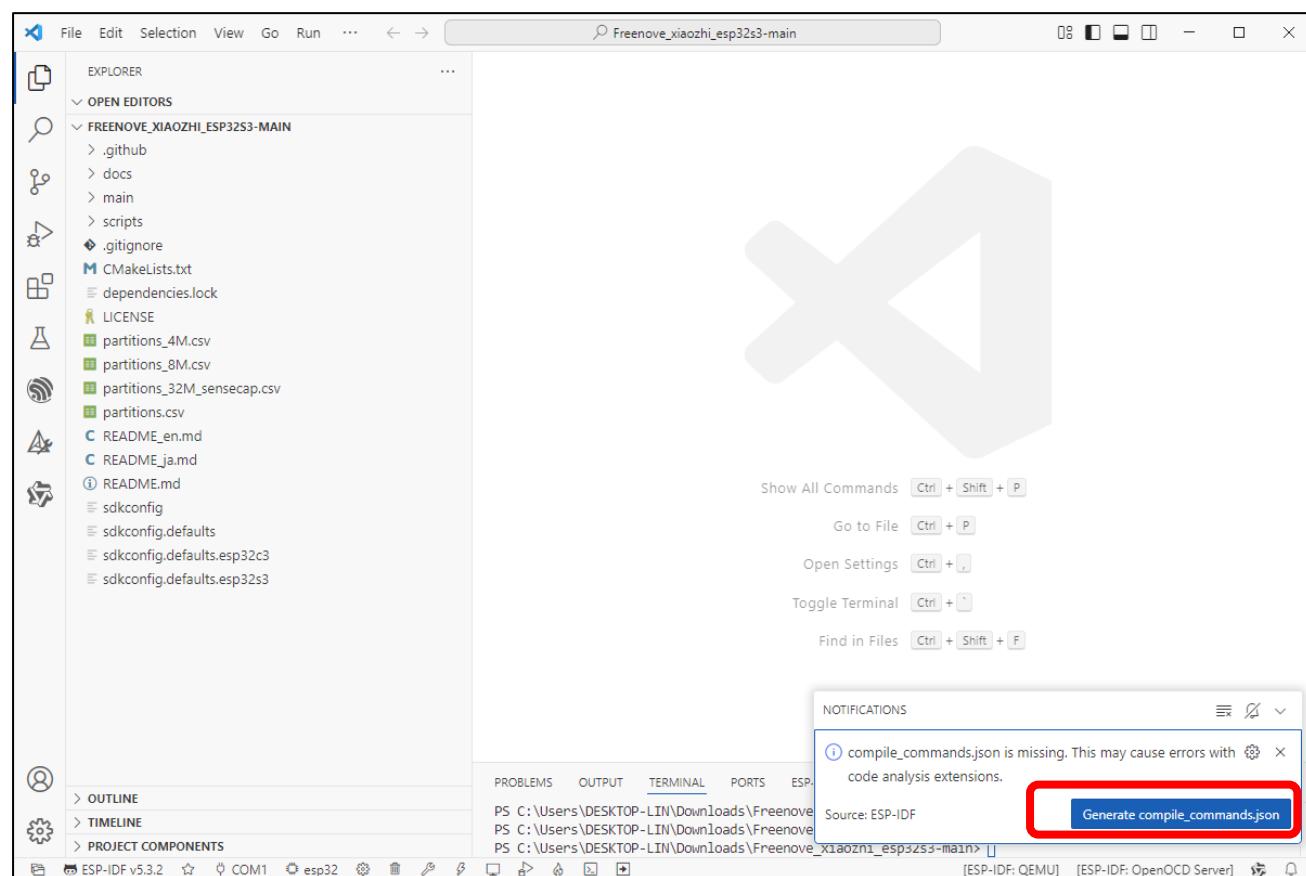
选择 xiaozhi-esp32 文件夹。这里以 Window 系统举例，在 MAC 系统和 Ubuntu 系统上的操作类似。



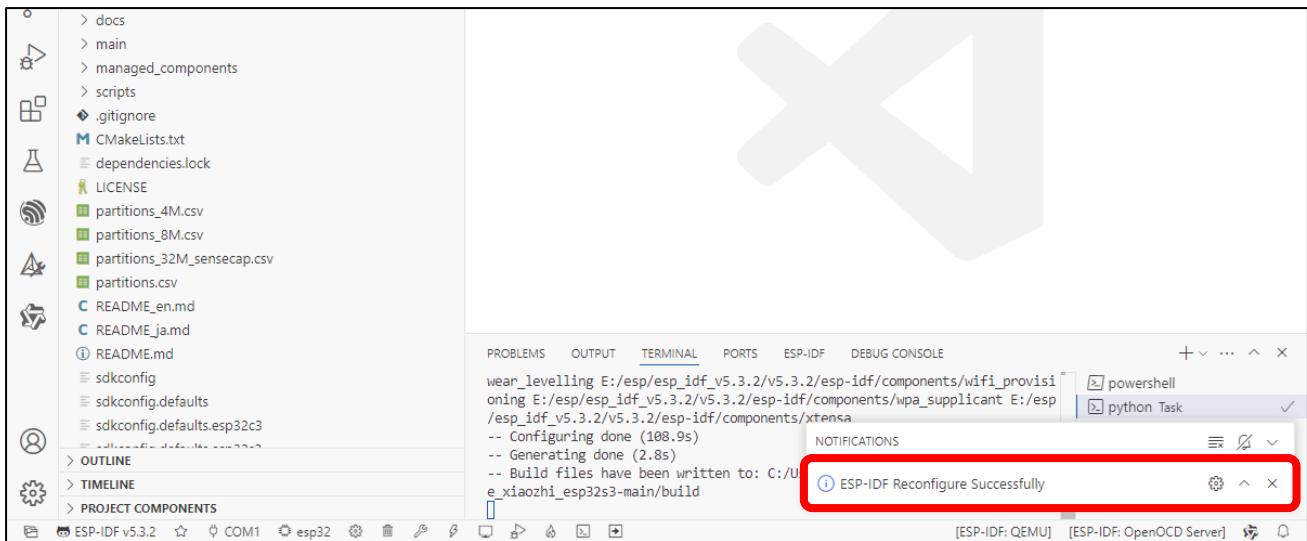
勾选复选框，并点击“*Yes, I trust the authors of all files in the parent folder ‘Downloads’*”。



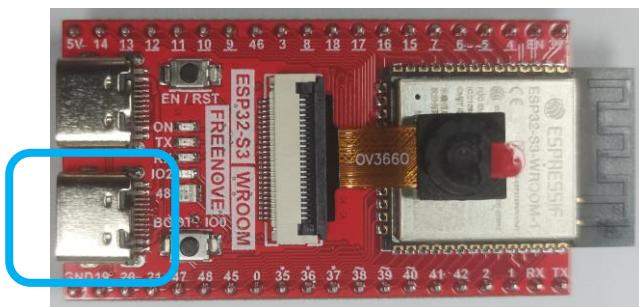
请注意，右下角会弹出一个提示框，请点击“*Generate compile_commands.json*”，它将根据文件下载对应的组件模块代码。



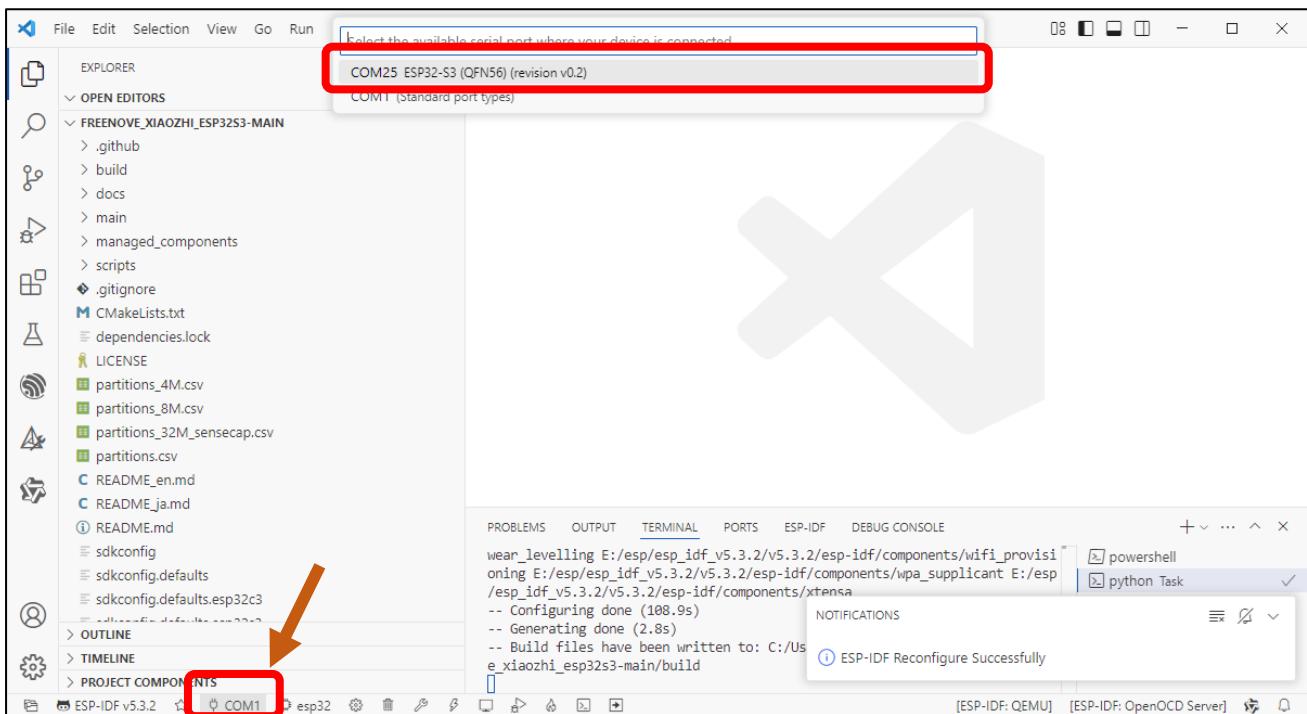
组件安装需要一些时间，请注意等待，不要进行其他操作。组件安装完成会在右下角打印提示。



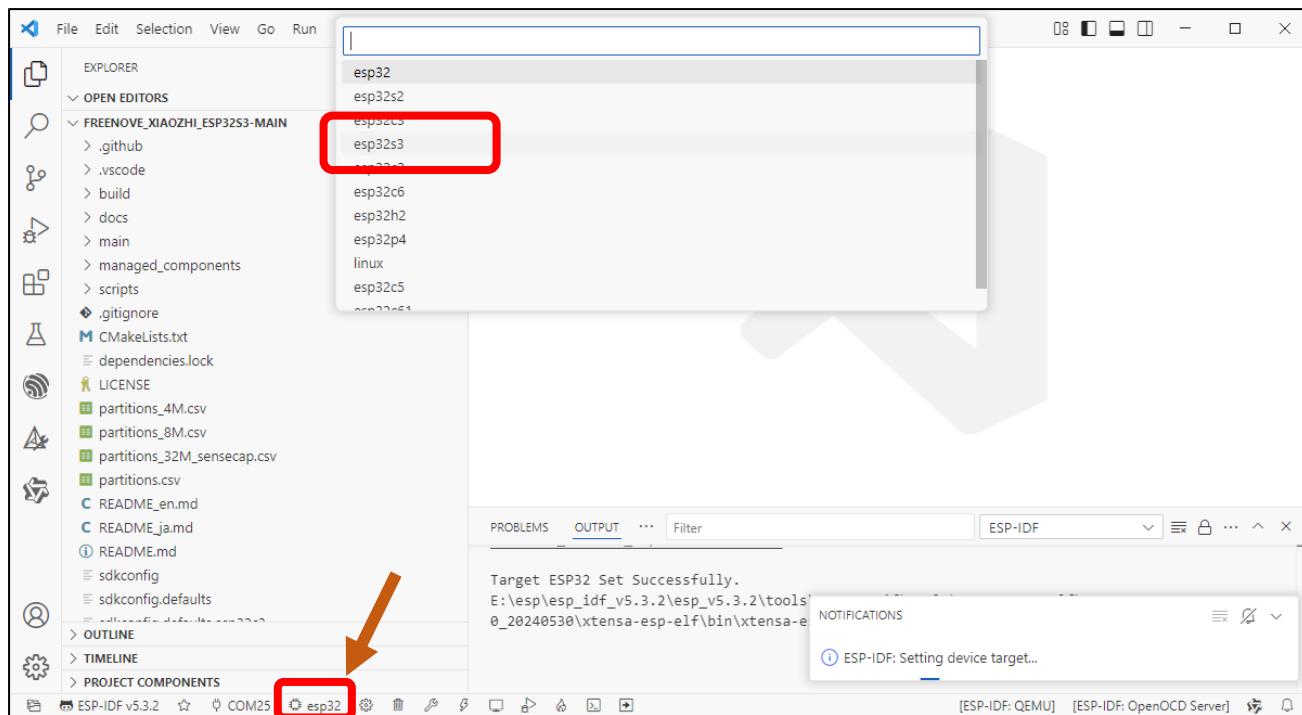
使用数据线连接电脑和 ESP32 S3 WROOM，请注意，不要连接错 Type C 接口。



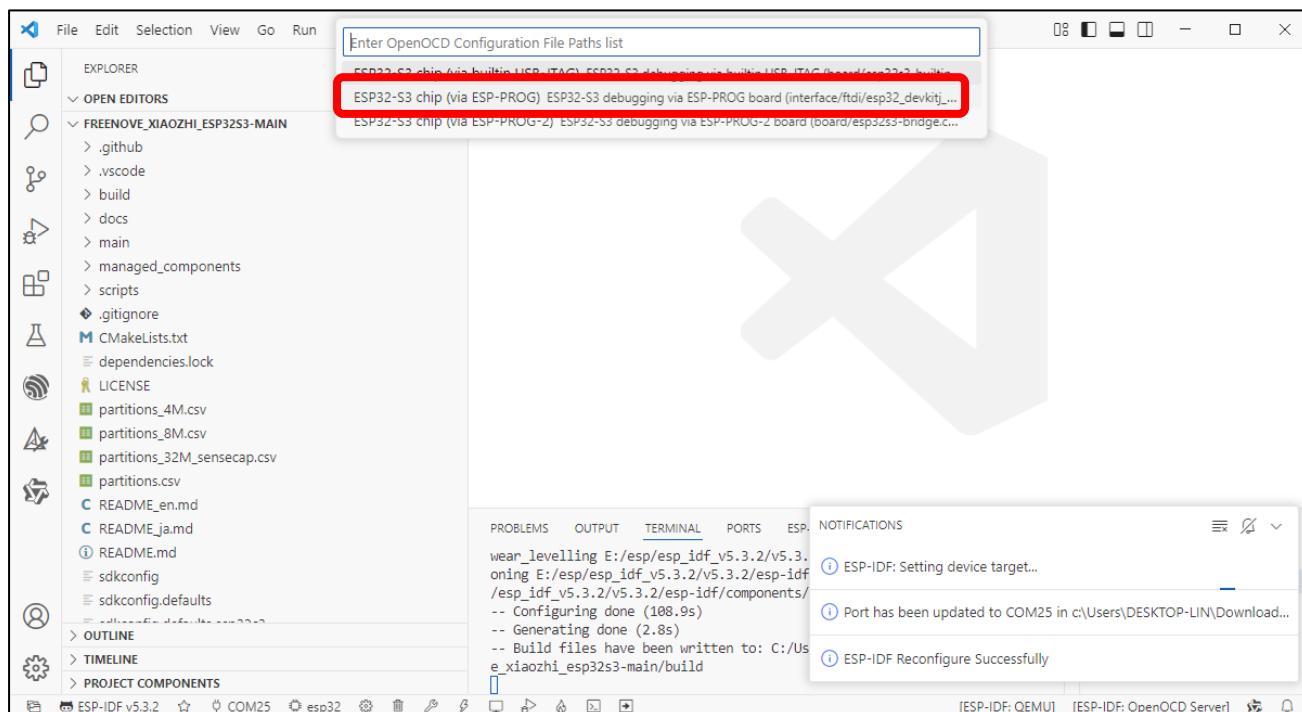
点击左下角的 COMx，在上方会出现电脑上所有的 COM 设备端口号。找到 ESP32-S3，并选择那一项。



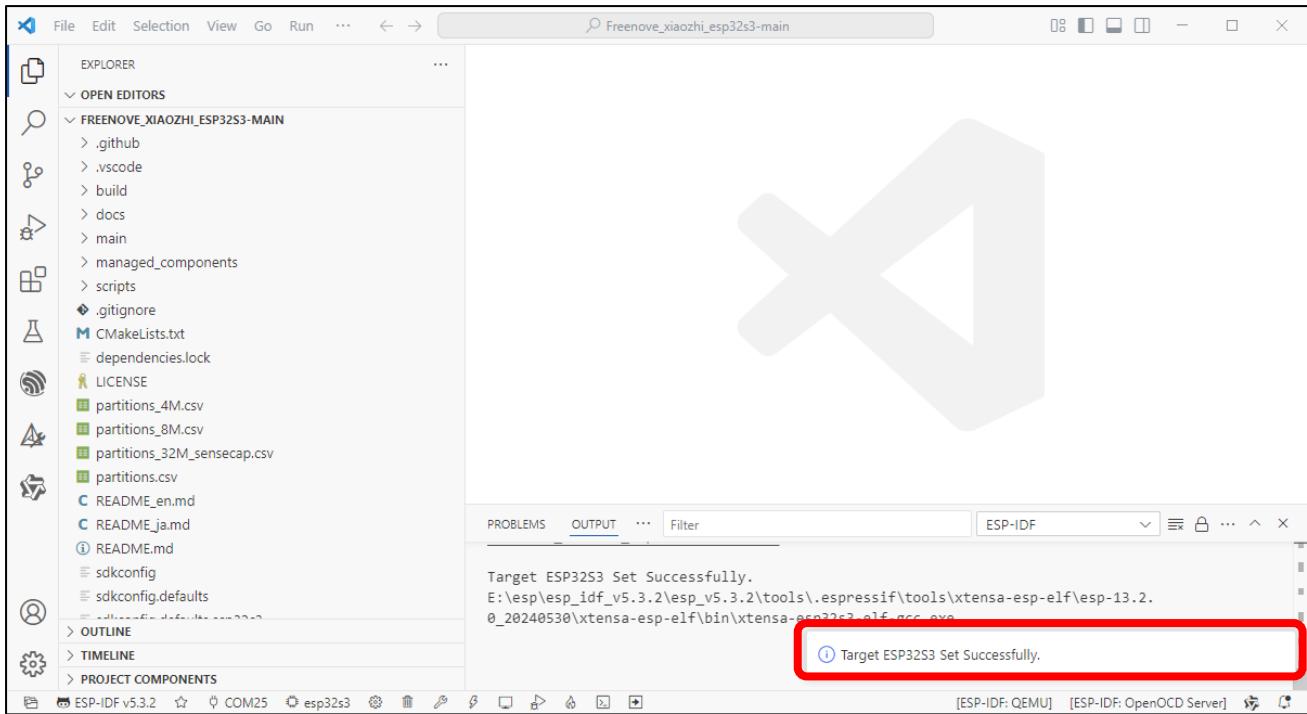
点击左下角的 ESP32，在上方会出现很多 ESP32 型号，选择 ESP32S3。



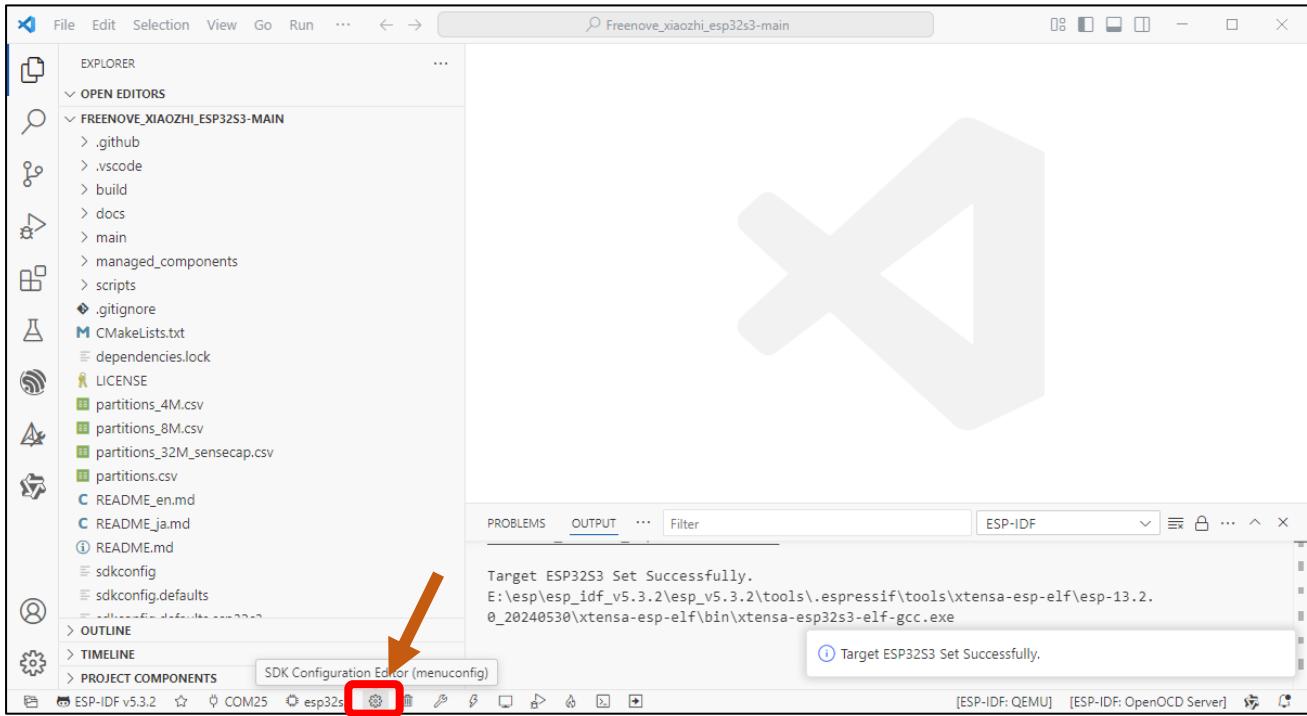
在新的选择框中，选择“ESP32-S3 Chip (via ESP-PROG) ESP32-S3 debugging via ESP-PROG Board ...”。



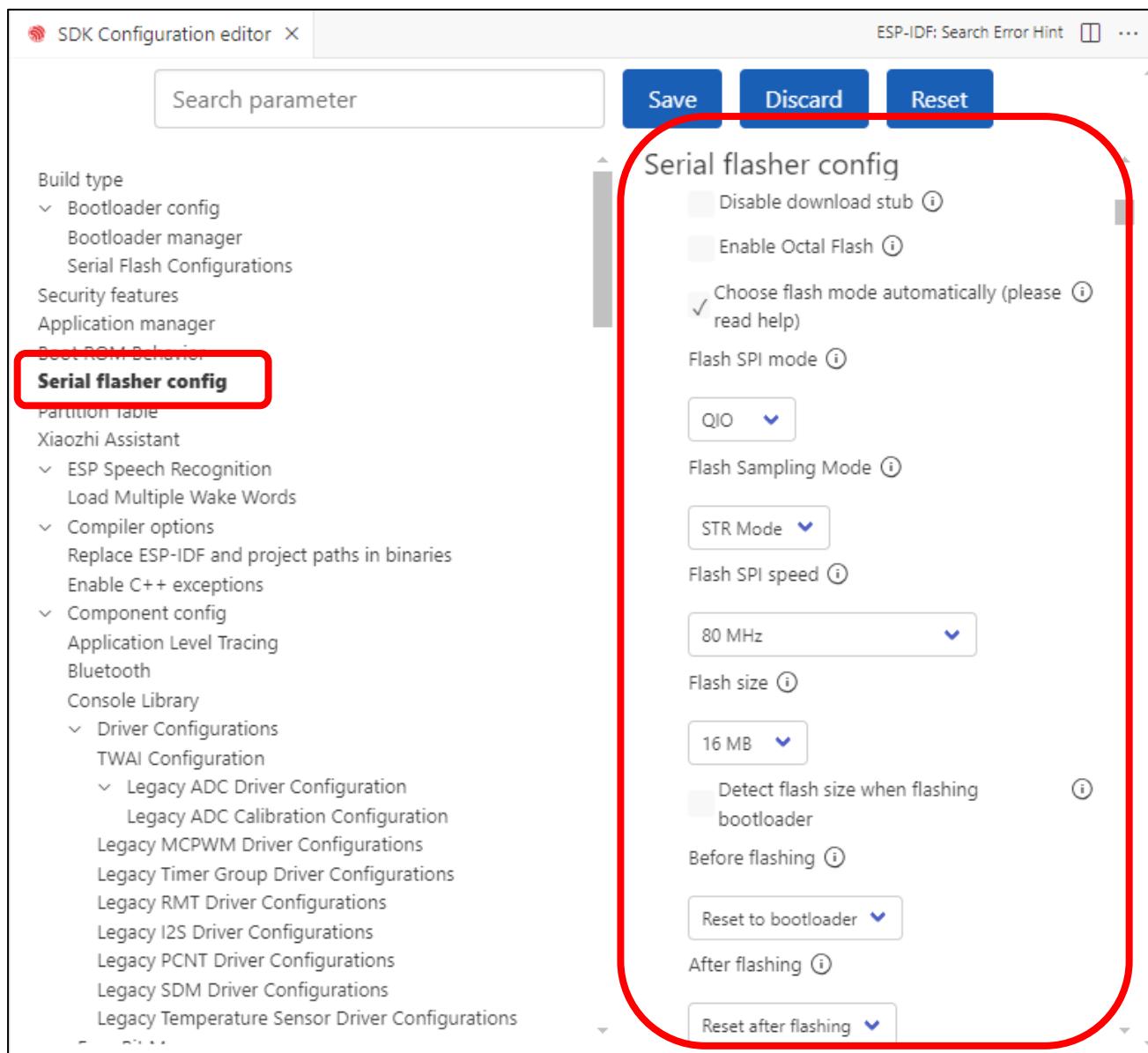
请等待一会，直到下方出现“Target ESP32S3 Set Successfully.”的提示。



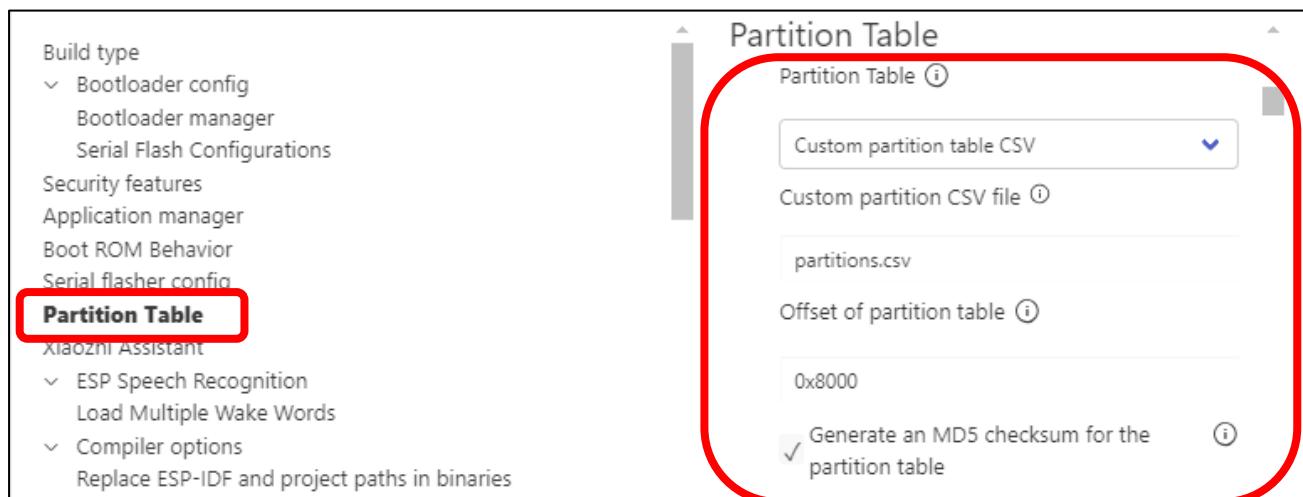
接下来点击 SDK Configuration Editor (menuconfig)选项。



在新的界面中，点击“Serial flasher config”，并检查配置是否和下图一致。



点击“Partition Table”，并检查配置是否和下图一致。



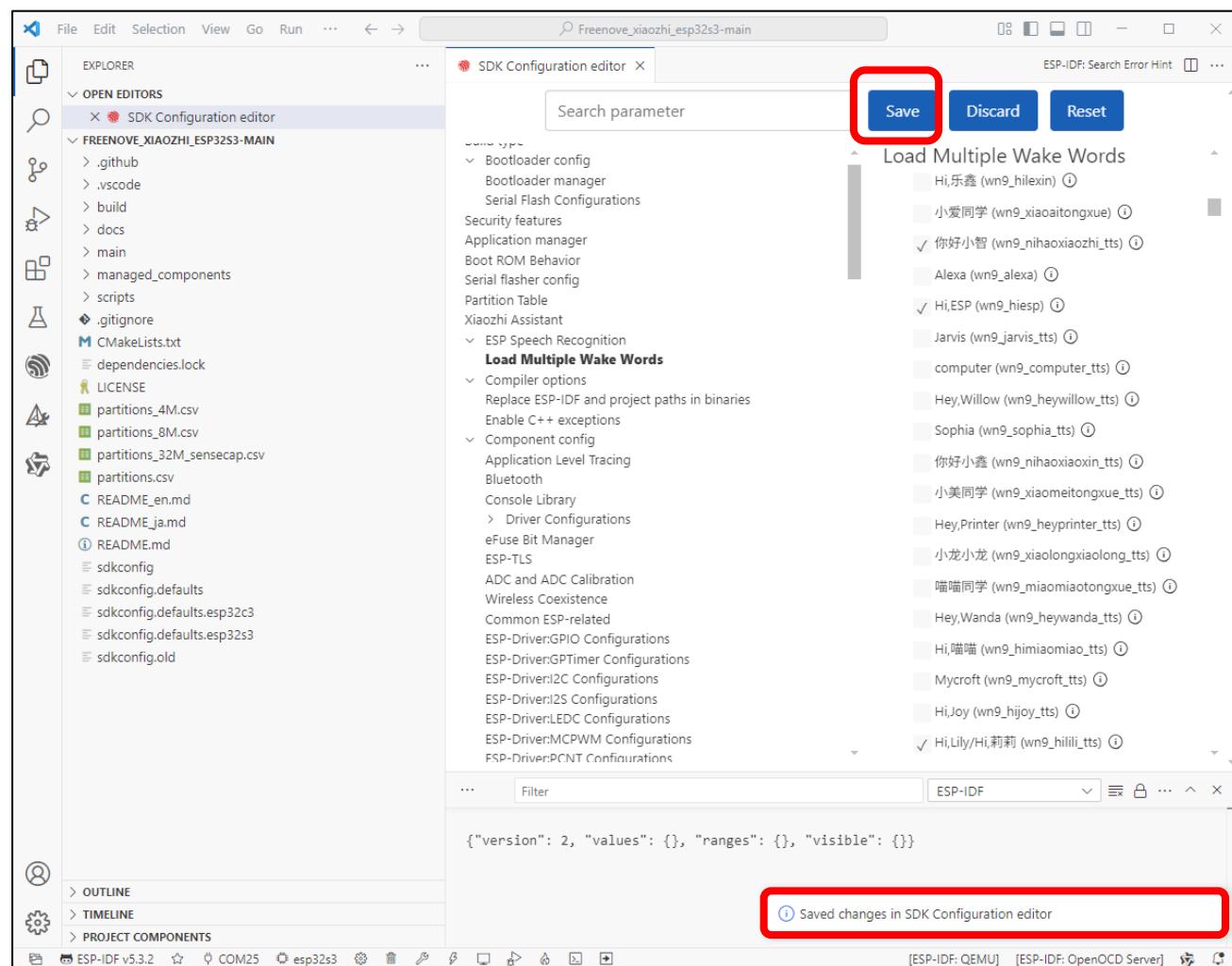
点击“Xiao Assistant”，并检查配置是否和下图一致。

The screenshot shows the configuration interface for the Freenove Media Kit for ESP32-S3. A red box highlights the "Xiao Assistant" section. Inside this section, the "OTA Version URL" is set to <https://api.tenclass.net/xiaozhi/ota/>. The "Language" dropdown is set to English. The "Connection Type" is set to MQTT + UDP. The "Board Type" is set to Freenove Media Kit for ESP32-S3. Below these, two checkboxes are checked: "启用唤醒词检测" (Enable wake-up word detection) and "启用音频降噪、增益处理" (Enable audio noise reduction, gain processing).

点击 Load Multiple Wake Words， 并勾选“Hi, ESP”、“Hi, Lily”等选项。

The screenshot shows the configuration interface for the Freenove Media Kit for ESP32-S3. A red box highlights the "Load Multiple Wake Words" section. This section lists various wake-up words and their corresponding names in parentheses. Several checkboxes are checked, indicating they are selected: "你好小智 (wn9_nihaoxiaozi_tts)", "Hi,ESP (wn9_hiesp)", "Hi,Lily/Hi,莉莉 (wn9_hilili_tts)", and "Hi,Joy (wn9_hijoy_tts)".

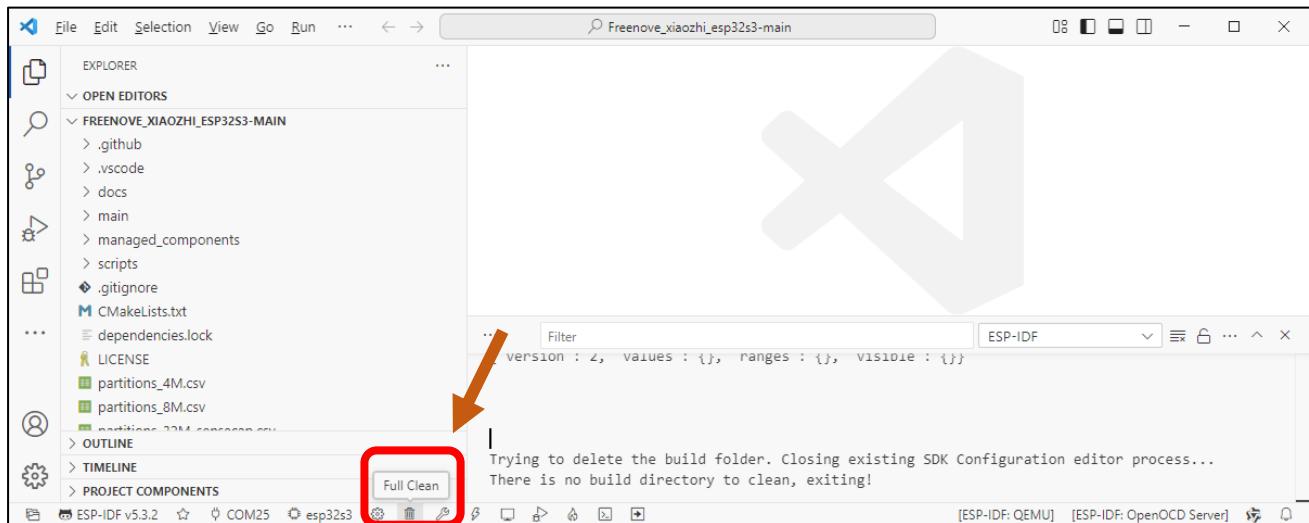
最后，点击“Save”，保存您的配置。保存成功，下方会给出提示信息。



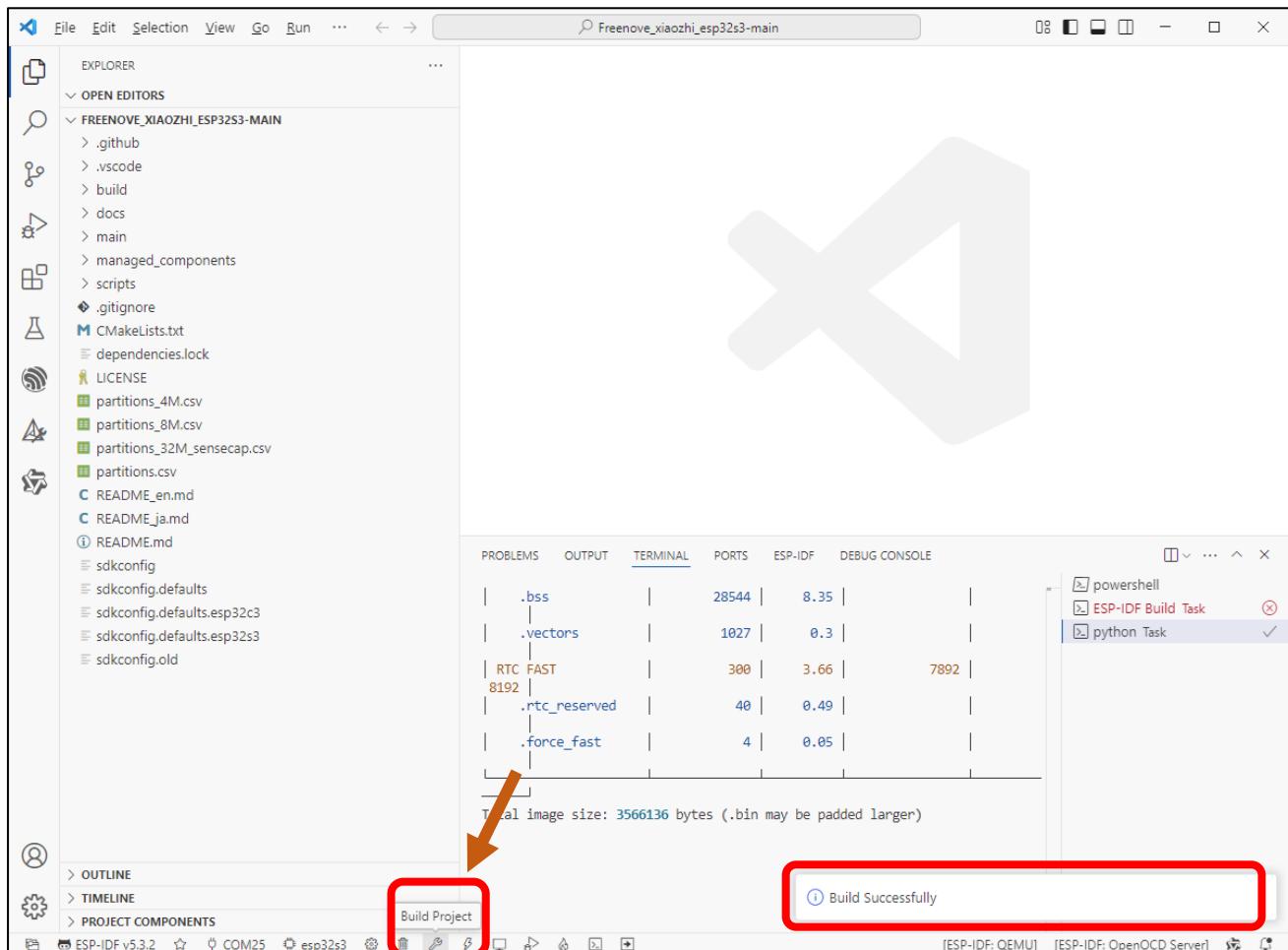
编译代码

确保一切都配置完成之后，我们开始编译代码。

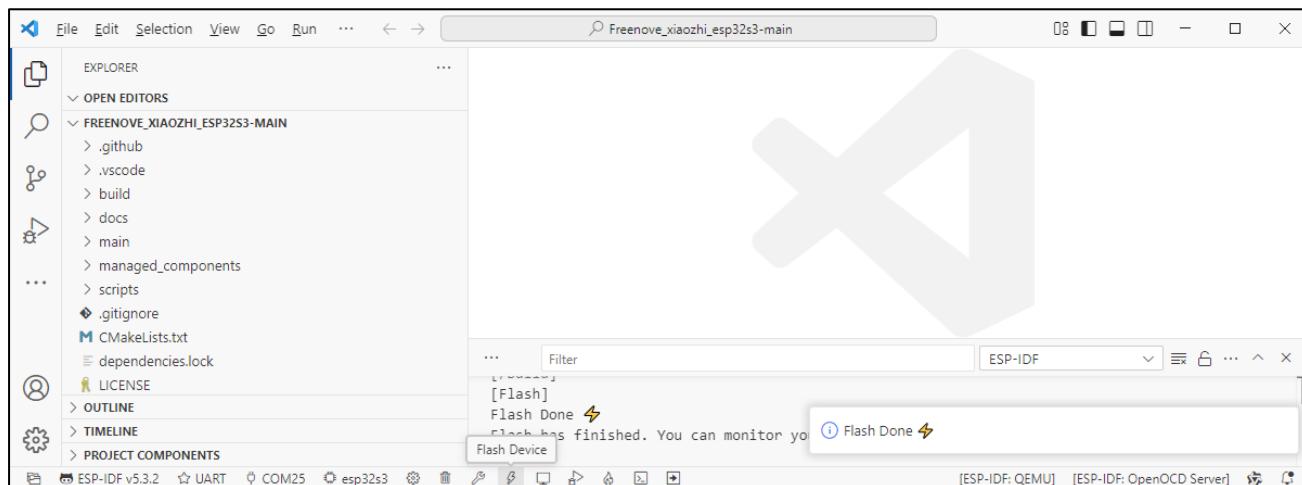
点击下方的“Full Clean”，它将会清除之前的所有编译信息。



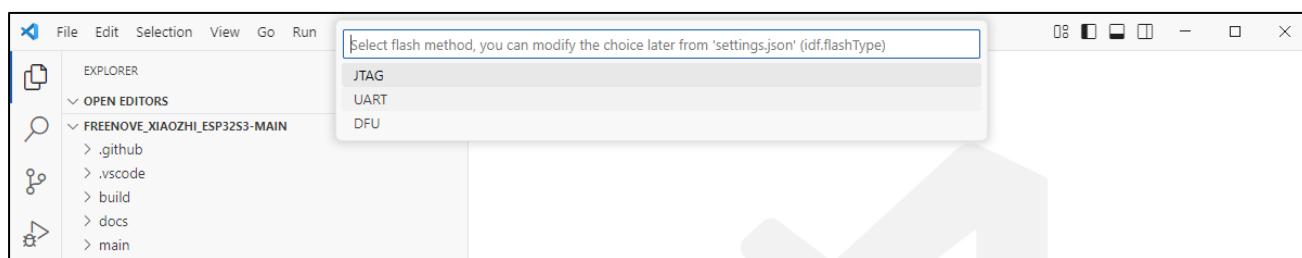
点击下方的“Build Project”，开始编译整个工程代码。首次编译需要等待的时间较长，请耐心等待，直到下方出现编译成功的提示。



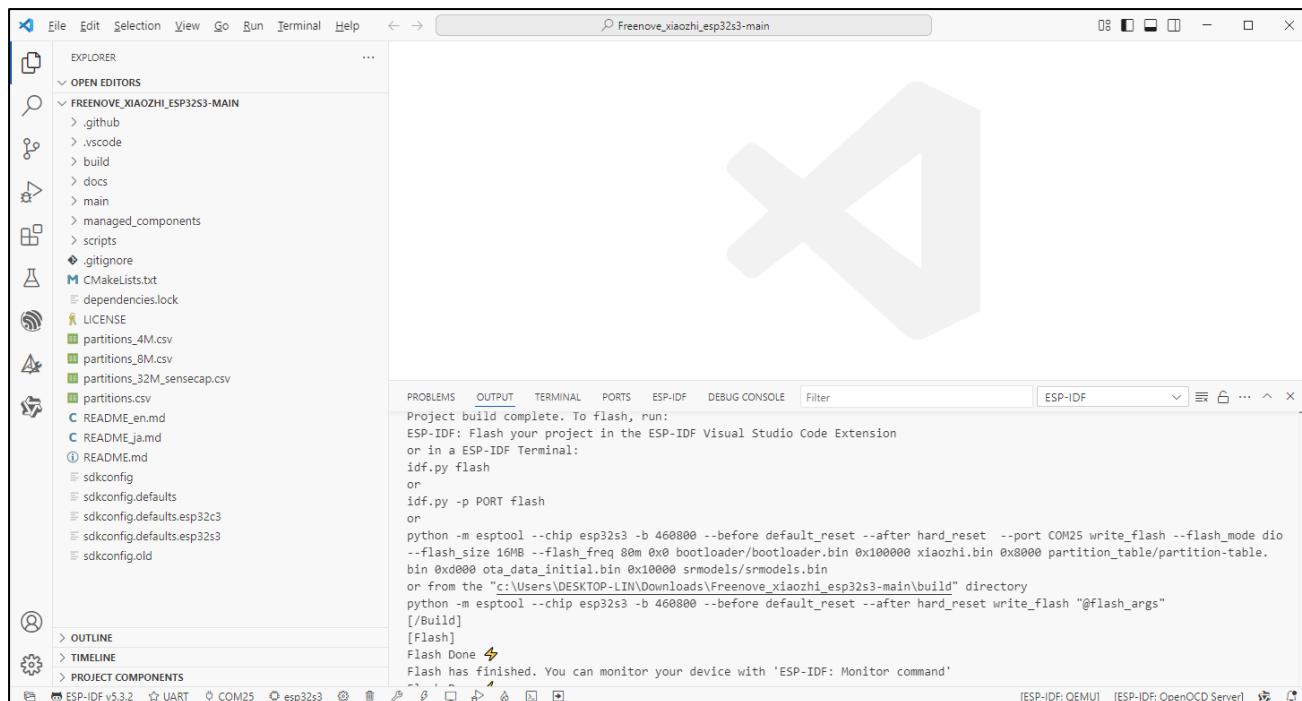
点击下方的“Flash Device”。准备开始上传代码到 ESP32 S3 Wroom 中。



在新出现的选项中，选择“UART”，然后等待代码上传完成即可。



看到“Flash has finished. You can monitor your device with 'ESP-IDF: Monitor command'”的提示，说明您已经将小智 AI 的代码上传到 ESP32 S3 WROOM 中。



至此，您已经编译完成，可以进行二次创作了。

本地服务器

小智 AI 服务器使用声明

本项目基于网络开源项目 <https://github.com/xinnan-tech/xiaozhi-esp32-server>。该项目开源协议类型为 MIT 协议。我们仅在此基础上提供适配，用做第三方学习和 AI 功能试用，不做商业性质推广和应用。本教程仅给爱好者附加学习使用。

小智 AI 本地服务器部署

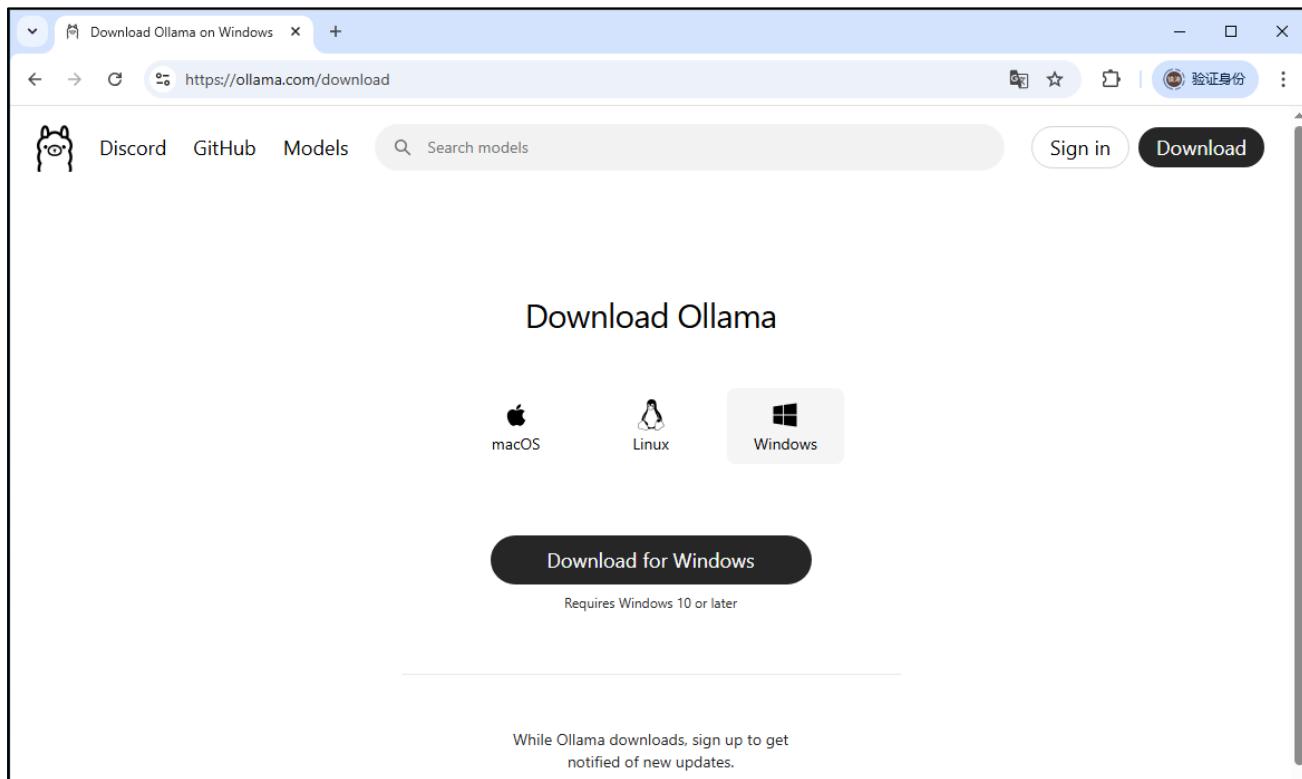
如果您不想使用小智 AI 服务器，您也可以使用自己的电脑搭建一个简易版本的服务器，这个章节我们将使用开源项目 <https://github.com/xinnan-tech/xiaozhi-esp32-server> 来部署一个本地服务器，并和 ESP32 S3 WROOM 建立连接。如果您在使用过程中发现代码存在 bug，请在 <https://github.com/xinnan-tech/xiaozhi-esp32-server> 上提交 issue，我们对此项目并不深入了解，无法为您提供太多帮助。

安装 Ollama

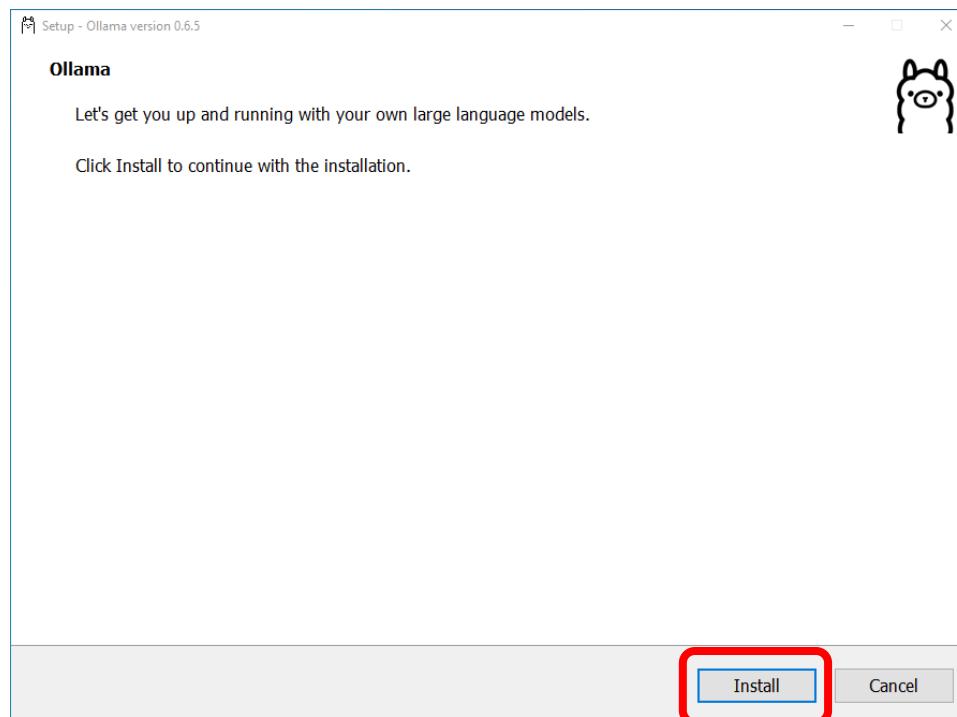
Window

在开始之前，我们需要在本地先安装 Ollama 工具，它可以在我们的电脑上安装任意开源模型。

如果您还没有安装 Ollama，请访问 <https://ollama.com/download> 进行下载和安装。



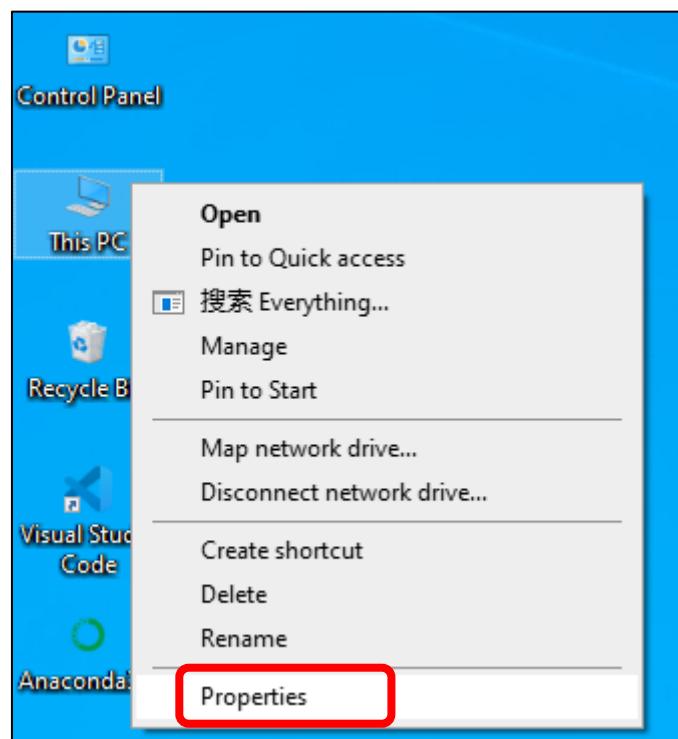
运行 Ollama 安装包，点击 Install.



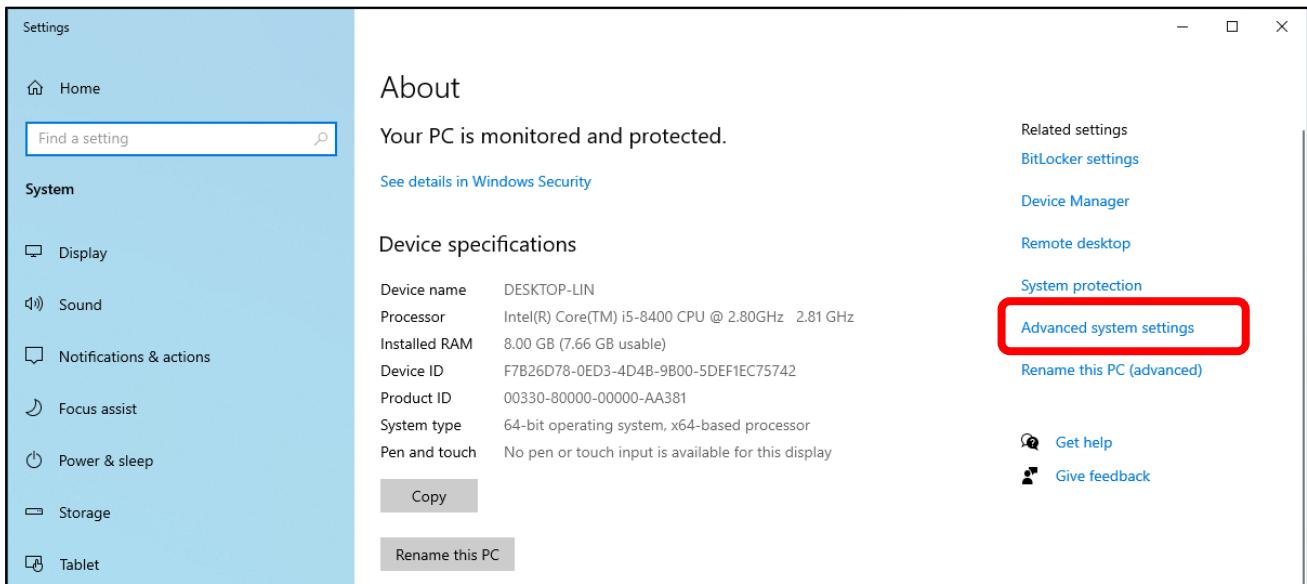
安装完成后，您可以在电脑的右下角看到图标。



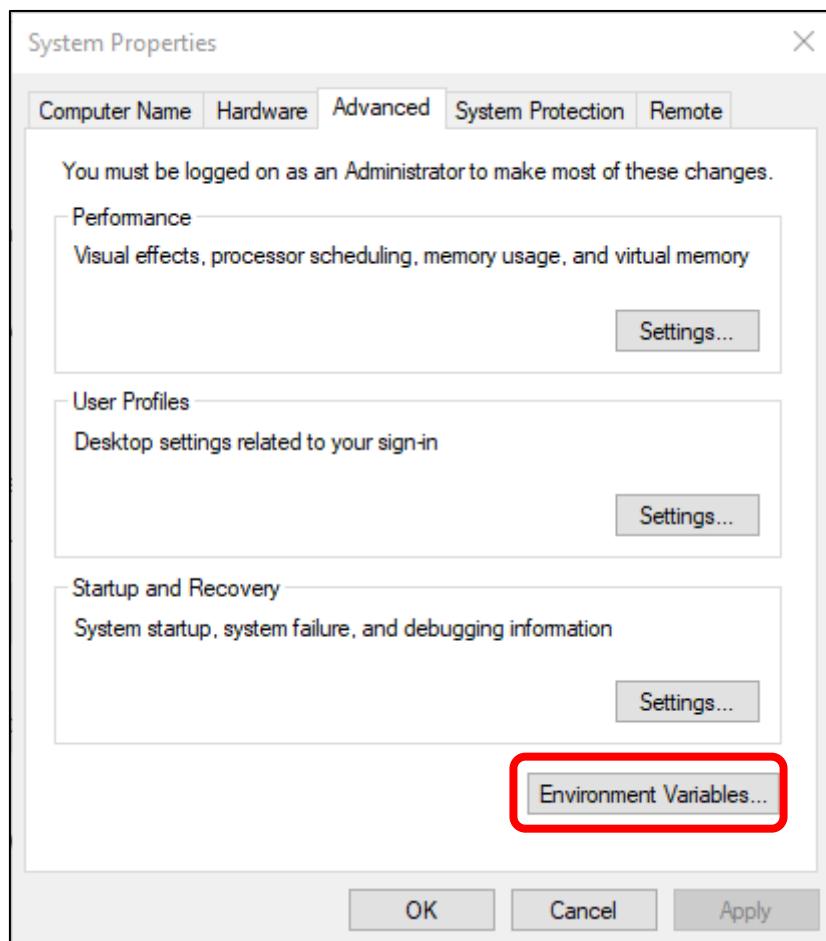
回到电脑的桌面位置，选择此电脑，按下鼠标右键，选择“Properties”.



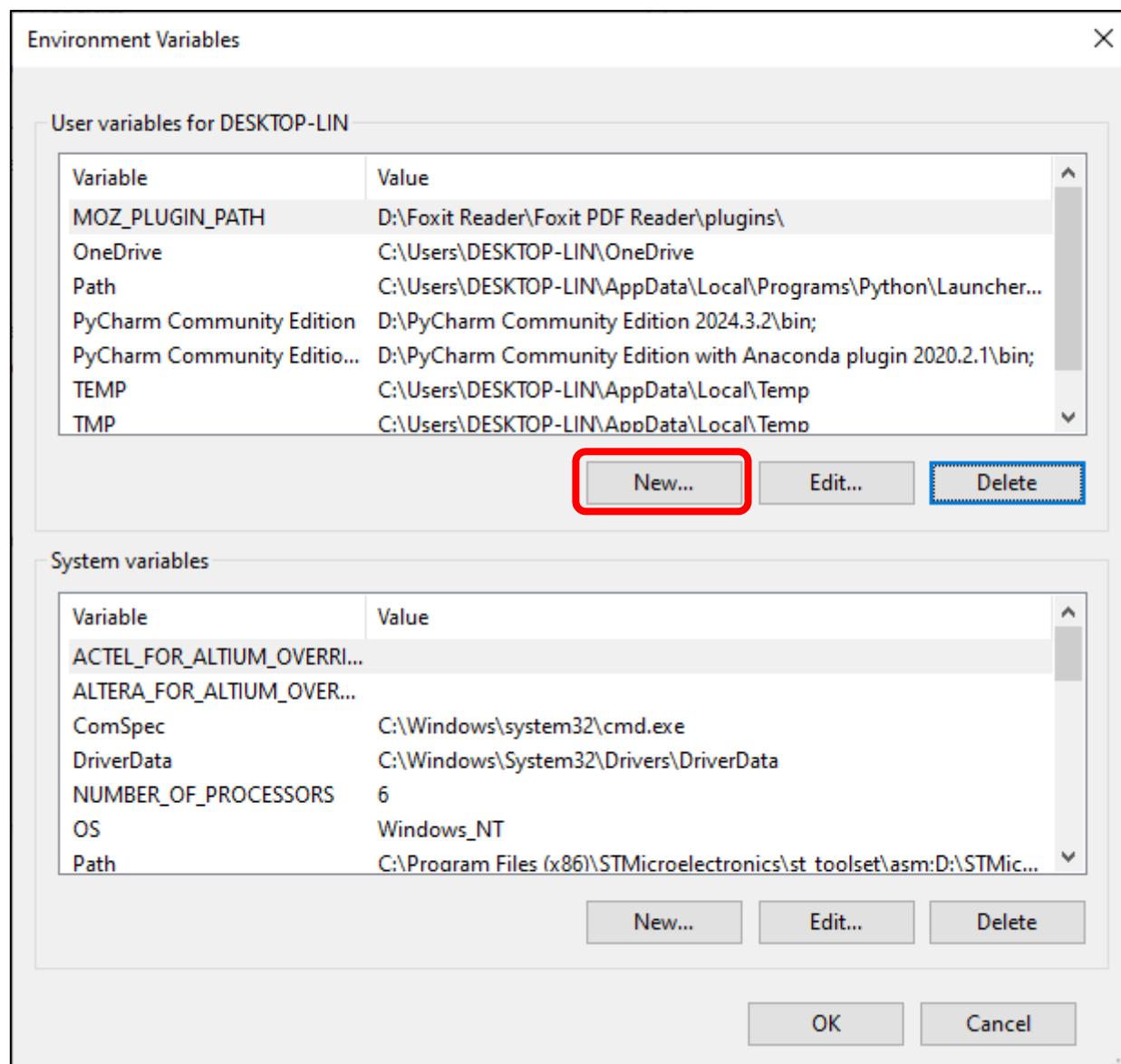
在新弹出的界面中，找到“**Advances system settings**”，然后点击它。



在新的窗口中，点击“Environment Variables”。

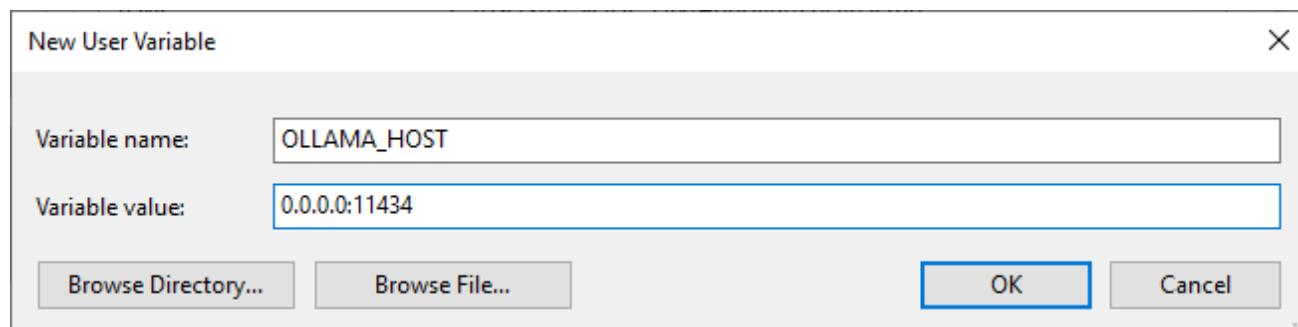


点击“New”。



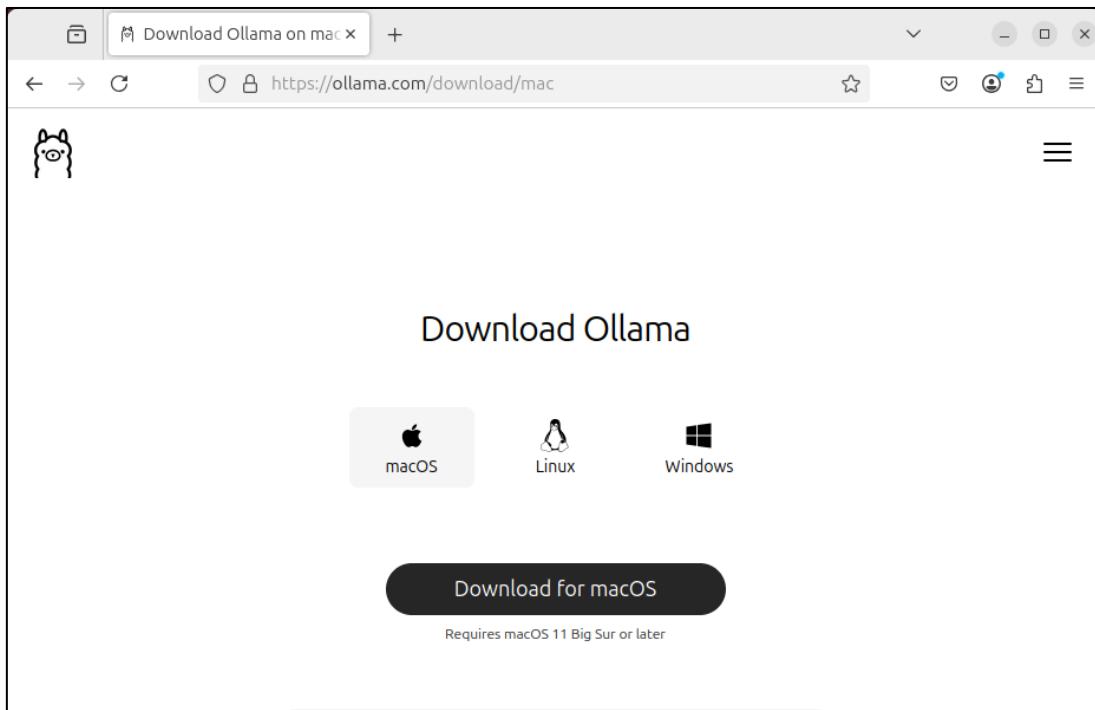
变量名称输入“OLLAMA_HOST”，变量值输入“0.0.0.0:11434”，然后点击 OK。

这样局域网内所有的设备都可以通过 IP 地址访问 Ollama。否则只能电脑本身访问 Ollama.

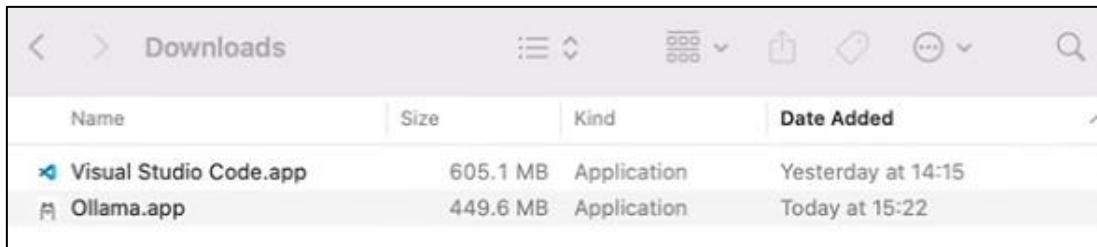


MAC

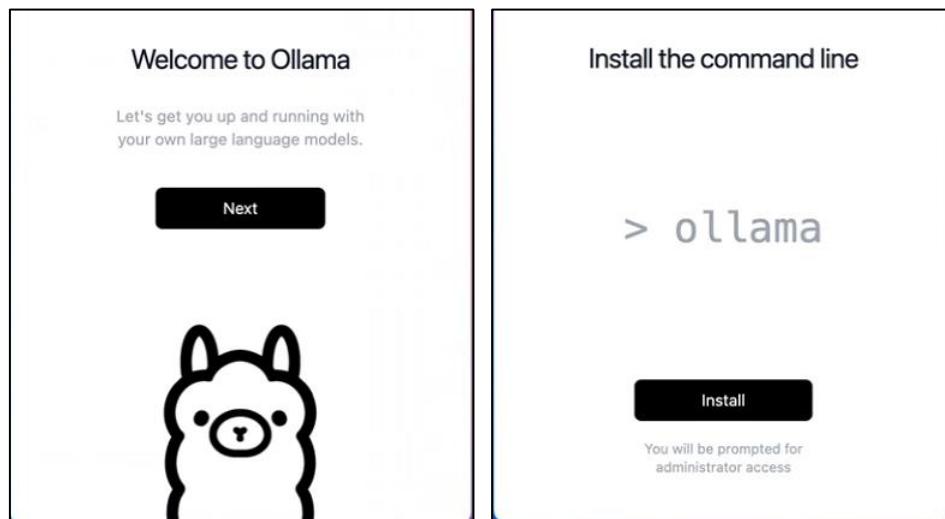
在开始之前，我们需要在本地先安装 Ollama 工具，它可以在我们的电脑上安装任意开源模型。如果您还没有安装 Ollama，请访问 <https://ollama.com/download> 进行下载和安装。



在 Downloads 中找到“Ollama.app”，双击运行它。



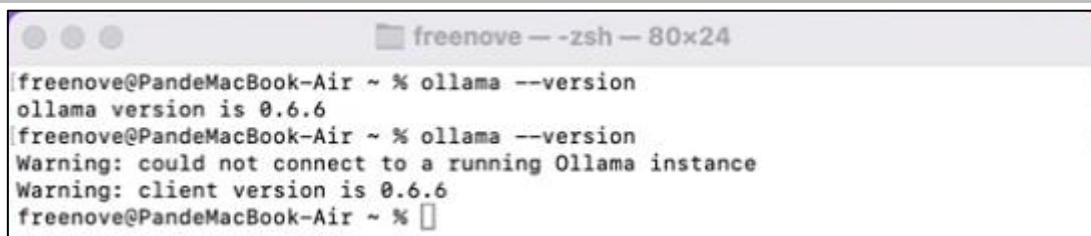
点击“Next”，然后点击“Install”，最后点击“Finish”



安装完成后，界面会直接关闭。

打开终端，通过指令查看 Ollama 是否已经安装完成。

ollama --version



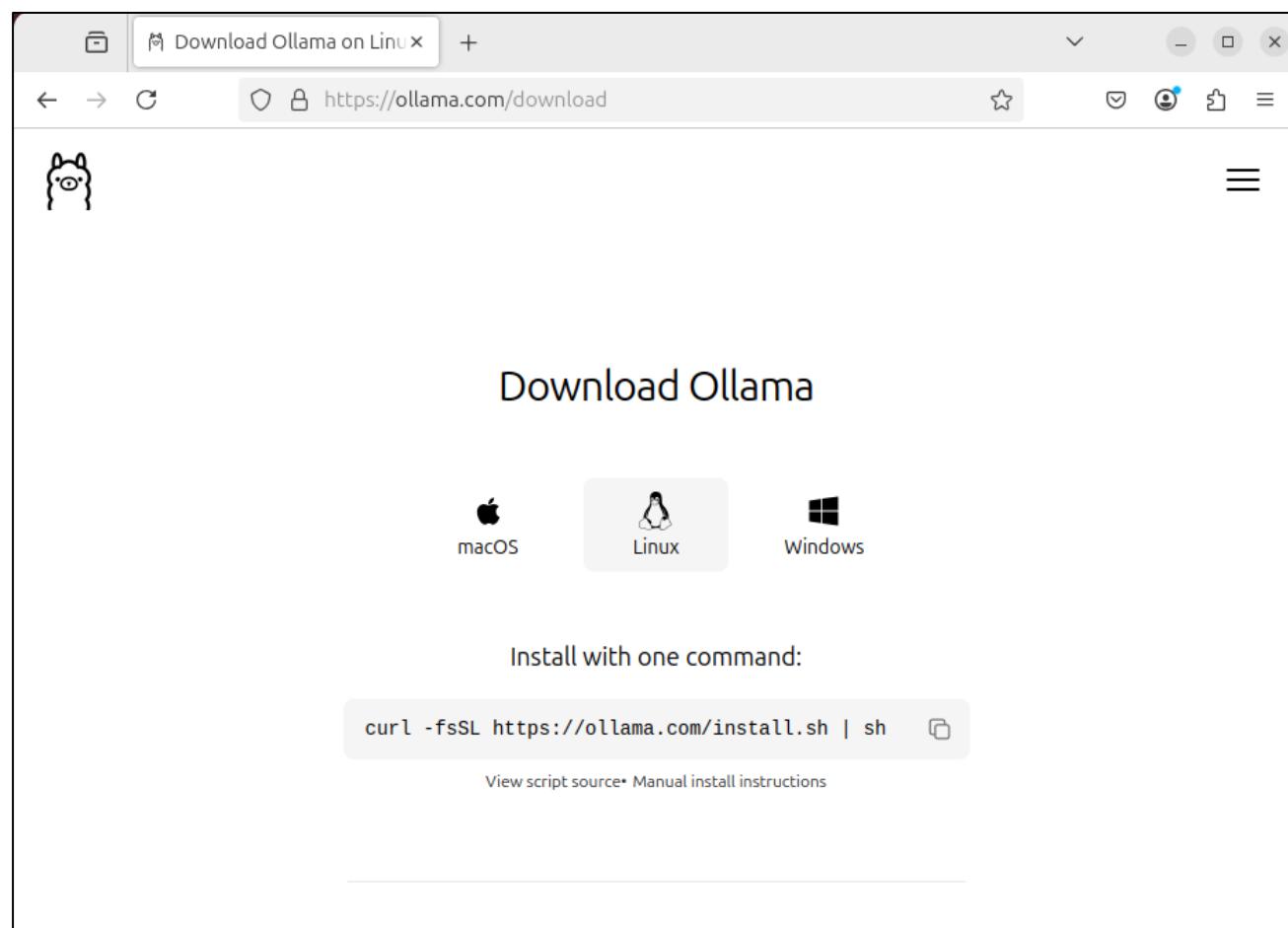
```
freenove@PandeMacBook-Air ~ % ollama --version
ollama version is 0.6.6
freenove@PandeMacBook-Air ~ % ollama --version
Warning: could not connect to a running Ollama instance
Warning: client version is 0.6.6
freenove@PandeMacBook-Air ~ %
```

请注意，Ollama 存放在 Applications 中。如上图所示，如果您的 Ollama 已经在运行，则运行“ollama --version”时，会直接打印 Ollama 的版本号。如果您的 Ollama 没有运行，则运行“ollama --version”时，会提示您无法连接到运行中的 Ollama。

Linux

在开始之前，我们需要在本地先安装 Ollama 工具，它可以在我们的电脑上安装任意开源模型。

如果您还没有安装 Ollama，请访问 <https://ollama.com/download> 进行下载和安装。





打开终端。输入指令，开始安装 Ollama。

```
lin@ubuntu:~$ curl -fsSL https://ollama.com/install.sh | sh
```

安装完成如下图所示。当然，您可以使用“ollama --version”查看是否已经安装 Ollama。

```
>>> Creating ollama user...
>>> Adding ollama user to render group...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
Created symlink '/etc/systemd/system/default.target.wants/ollama.service' → '/etc/systemd/system/ollama.service'.
>>> The Ollama API is now available at 127.0.0.1:11434.
>>> Install complete. Run "ollama" from the command line.
WARNING: No NVIDIA/AMD GPU detected. Ollama will run in CPU-only mode.
lin@ubuntu:~$ ollama --version
ollama version is 0.6.6
lin@ubuntu:~$
```

LLM 模型

请访问 <https://ollama.com/search>, 选择适合您电脑的, 或者您喜欢的 LLM 模型。

The screenshot shows the Ollama Search interface. At the top, there is a navigation bar with links for Discord, GitHub, and Models, a search bar labeled 'Search models', and buttons for 'Sign in' and 'Download'. Below the navigation bar, there are three tabs: 'Embedding', 'Vision', and 'Tools', with 'Popular' selected. The main content area displays three LLM models:

- gemma3**: The current, most capable model that runs on a single GPU. It has 27b parameters. It was last updated 4 hours ago.
- qwp**: QwQ is the reasoning model of the Qwen series. It has 32b parameters. It was last updated 5 weeks ago.
- deepseek-r1**: DeepSeek's first-generation of reasoning models with comparable performance to OpenAI-o1, including six dense models distilled from DeepSeek-R1 based on Llama and Qwen. It has 671b parameters. It was last updated 2 months ago.

这里我以 qwen2.5 举例。点击“qwen2.5”模型。

The screenshot shows the Ollama interface with the 'Models' tab selected. The search bar shows 'Search models'. The main content area displays the 'qwen2.5' model details:

qwen2.5

Qwen2.5 models are pretrained on Alibaba's latest large-scale dataset, encompassing up to 18 trillion tokens. The model supports up to 128K tokens and has multilingual support.

It has 72b parameters. It was last updated 7 months ago.

Below the model details, there is a table with the following information:

7b	133 Tags	ollama run qwen2.5
Updated 7 months ago	845dbda0ea48 · 4.7GB	
model	arch qwen2 · parameters 7.62B · quantization Q4_K_M	4.7GB
system	You are Qwen, created by Alibaba Cloud. You are a helpful assist...	688
template	{{- if .Messages }} {{- if or .System .Tools }}< im_start >{{.}}< im_end >{{.}}	1.5kB
license	Apache License Version 2.0, January 200	11kB



请注意，在选择模型时，需要根据您的电脑显卡内存或者 CPU 内存条配置，选择合适的模型。

- 1, 模型越大，智能化程度越高。模型越小，智能化程度越低。
- 2, 如果您的配置比较高，您可以选择比较大的模型，如果您的电脑配置比较低，您可以选择比较小的模型。
- 3, 如果您的电脑配置较低，而您选择的模型过大，可能会导致模型无法运行，或者运行的速度变慢。
通过下拉框，可以选择合适的模型参数。

The screenshot shows the GitHub repository page for Qwen2.5. At the top, it says "qwen2.5" and provides a brief description: "Qwen2.5 models are pretrained on Alibaba's latest large-scale dataset, encompassing up to 18 trillion tokens. The model supports up to 128K tokens and has multilingual support." Below this are buttons for "tools", "0.5b", "1.5b", "3b", "7b", "14b", "32b", and "72b". A red box highlights the dropdown menu where "7b" is selected. To the right of the dropdown is a button labeled "ollama run qwen2.5" with a copy icon.

Model	Size	Details
0.5b	398MB	parameters 7.62B · quantization Q4_K_M
1.5b	986MB	parameters 15.2B · quantization Q4_K_M
3b	1.9GB	parameters 30.4B · quantization Q4_K_M
7b	4.7GB	parameters 49.4M · quantization Q4_K_M
14b	9.0GB	parameters 98.8M · quantization Q4_K_M
32b	20GB	parameters 197.6M · quantization Q4_K_M
72b	47GB	parameters 395.2M · quantization Q4_K_M

At the bottom left are "View all" and "Readme" links.

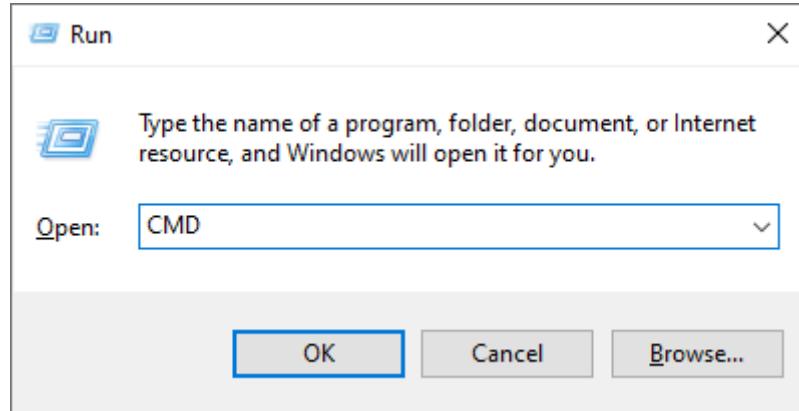
模型越小，越不智能，但是运行速度越快，这里只是作为演示，我们选择 qwen2.5:0.5b 来作为示范。复制网页中的指令：“**ollama run qwen2.5:0.5b**”

The screenshot shows the same GitHub repository page for Qwen2.5. The dropdown menu now shows "0.5b" selected. The "ollama run qwen2.5:0.5b" button is highlighted with a red box. The rest of the page content is identical to the first screenshot.

接下来请根据您的电脑系统，安装您喜欢的 LLM 模型。

Window

您可以使用指令“Win+R”，在弹出的窗口中输入“CMD”，打开 CMD 界面。



输入指令“`ollama --version`”，查看是否已经安装了 ollama。

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DESKTOP-LIN>ollama --version
ollama version is 0.6.5

C:\Users\DESKTOP-LIN>
```

A screenshot of a Windows Command Prompt window titled "cmd.exe" running in the "C:\Windows\system32" directory. It shows the system version and the result of running the "ollama --version" command, which outputs "ollama version is 0.6.5".

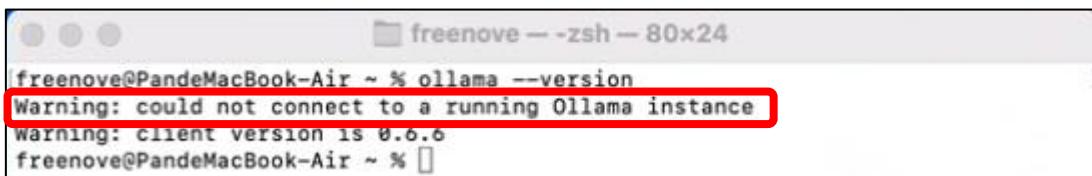
输入“`ollama run qwen2.5:0.5b`”，将模型下载到本地中。

```
C:\Windows\system32\cmd.exe - ollama run qwen2.5:0.5b
C:\Users\DESKTOP-LIN>ollama run qwen2.5:0.5b
pulling manifest
pulling c5396e06af29... 100%
pulling 66b9ea09bd5b... 100%
pulling eb4402837c78... 100%
pulling 832dd9e00a68... 100%
pulling 005f95c74751... 100%
verifying sha256 digest
writing manifest
success
>>> Send a message (/? for help)
```

A screenshot of a Windows Command Prompt window titled "cmd.exe" running in the "C:\Windows\system32" directory. A red box highlights the command "ollama run qwen2.5:0.5b" entered at the prompt. The output shows the process of pulling the manifest and individual model components, followed by verification and success messages.

MAC

打开终端，输入指令“`ollama --version`”，查看是否已经安装了 ollama。

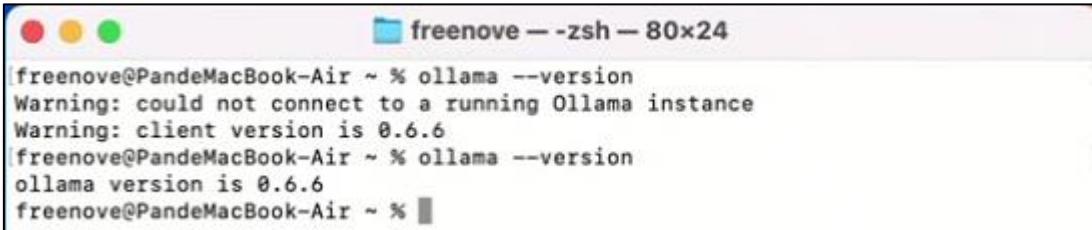


```
freenove@PandeMacBook-Air ~ % ollama --version
Warning: could not connect to a running Ollama instance
Warning: client version is 0.6.6
freenove@PandeMacBook-Air ~ %
```

如果出现“Warning: could not connect to a running Ollama instance”的提示，请先运行 Ollama。

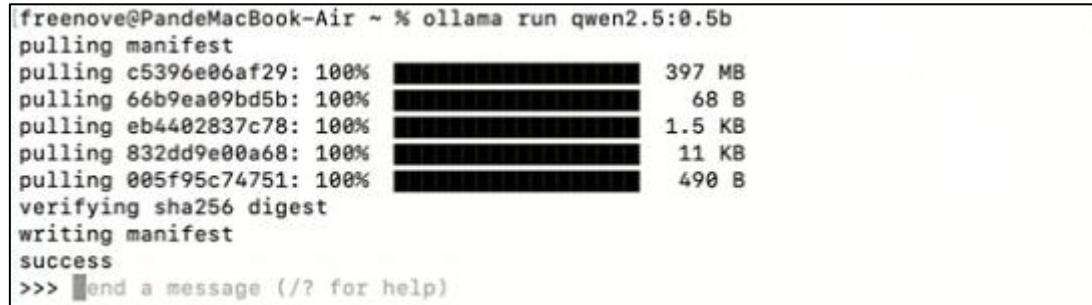


重新使用指令，查看 Ollama 是否正常运行。



```
freenove@PandeMacBook-Air ~ % ollama --version
Warning: could not connect to a running Ollama instance
Warning: client version is 0.6.6
freenove@PandeMacBook-Air ~ % ollama --version
ollama version is 0.6.6
freenove@PandeMacBook-Air ~ %
```

输入“`ollama run qwen2.5:0.5b`”，将模型下载到本地中。



```
freenove@PandeMacBook-Air ~ % ollama run qwen2.5:0.5b
pulling manifest
pulling c5396e06af29: 100% [██████████] 397 MB
pulling 66b9ea09bd5b: 100% [██████████] 68 B
pulling eb4402837c78: 100% [██████████] 1.5 KB
pulling 832dd9e00a68: 100% [██████████] 11 KB
pulling 005f95c74751: 100% [██████████] 490 B
verifying sha256 digest
writing manifest
success
>>> End a message (/? for help)
```

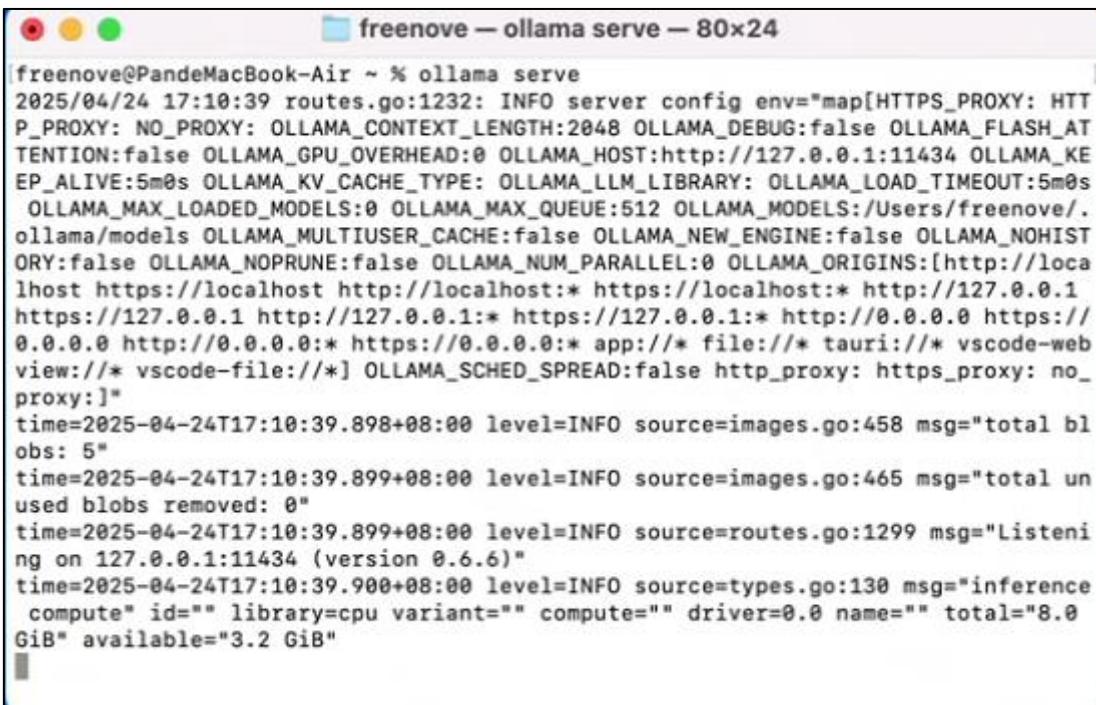
当安装完成后，你可以直接在终端界面中和 qwen2.5:0.5b 进行聊天。

```
freenove@PandeMacBook-Air ~ % ollama run qwen2.5:0.5b
pulling manifest
pulling c5396e06af29: 100% [██████████] 397 MB
pulling 66b9ea09bd5b: 100% [██████████] 68 B
pulling eb4402837c78: 100% [██████████] 1.5 KB
pulling 832dd9e00a68: 100% [██████████] 11 KB
pulling 005f95c74751: 100% [██████████] 490 B
verifying sha256 digest
writing manifest
success
>>> Hello
Hello! How can I assist you today? Let me know if there's anything
specific you'd like to discuss or any questions that need help with. I'm
here to provide information and answer queries in a way that's easy for
you to understand.

>>>
Use Ctrl + d or /bye to exit.
>>> █end a message (/? for help)
```

您可以使用指令“Ctrl+d”，退出聊天模式。

您可以通过指令“**ollama serve**”来运行 ollama 服务器。



```
freenove@PandeMacBook-Air ~ % ollama serve
2025/04/24 17:10:39 routes.go:1232: INFO server config env="map[HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_CONTEXT_LENGTH:2048 OLLAMA_DEBUG:false OLLAMA_FLASH_ATENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://127.0.0.1:11434 OLLAMA_KEEP_ALIVE:5m0s OLLAMA_kv_CACHE_TYPE: OLLAMA_llm_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:/Users/freenove/.ollama/models OLLAMA_MULTIUSER_CACHE:false OLLAMA_NEW_ENGINE:false OLLAMA_NOHISTORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost/* https://localhost/* http://127.0.0.1 https://127.0.0.1 http://127.0.0.1/* https://127.0.0.1/* http://0.0.0.0 https://0.0.0.0 http://0.0.0.0/* https://0.0.0.0/* app:///* file:///* tauri:///* vscode-webview:///* vscode-file:///*] OLLAMA_SCHED_SPREAD:false http_proxy: https_proxy: no_proxy:]"
time=2025-04-24T17:10:39.898+08:00 level=INFO source=images.go:458 msg="total blobs: 5"
time=2025-04-24T17:10:39.899+08:00 level=INFO source=images.go:465 msg="total unused blobs removed: 0"
time=2025-04-24T17:10:39.899+08:00 level=INFO source=routes.go:1299 msg="Listening on 127.0.0.1:11434 (version 0.6.6)"
time=2025-04-24T17:10:39.900+08:00 level=INFO source=types.go:130 msg="inference compute" id="" library=cpu variant="" compute="" driver=0.0 name="" total="8.0 GiB" available="3.2 GiB"
```

如果您的 Ollama 已经运行，则会提示您下面的界面。

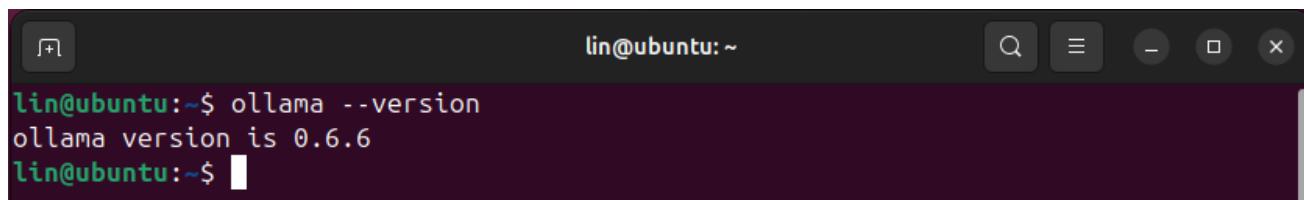


```
freenove@PandeMacBook-Air ~ % ollama serve
Error: listen tcp 127.0.0.1:11434: bind: address already in use
freenove@PandeMacBook-Air ~ %
```

您可以输入 Ollama，查看 Ollama 的使用说明。

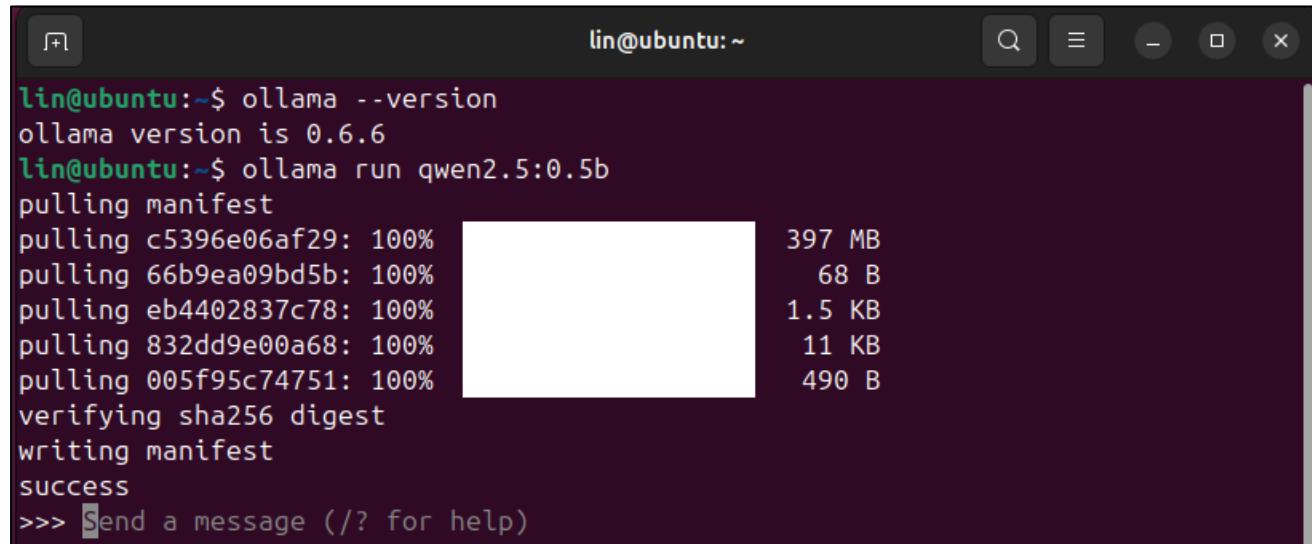
Linux

打开终端，输入指令“**ollama --version**”，查看是否已经安装了 ollama。



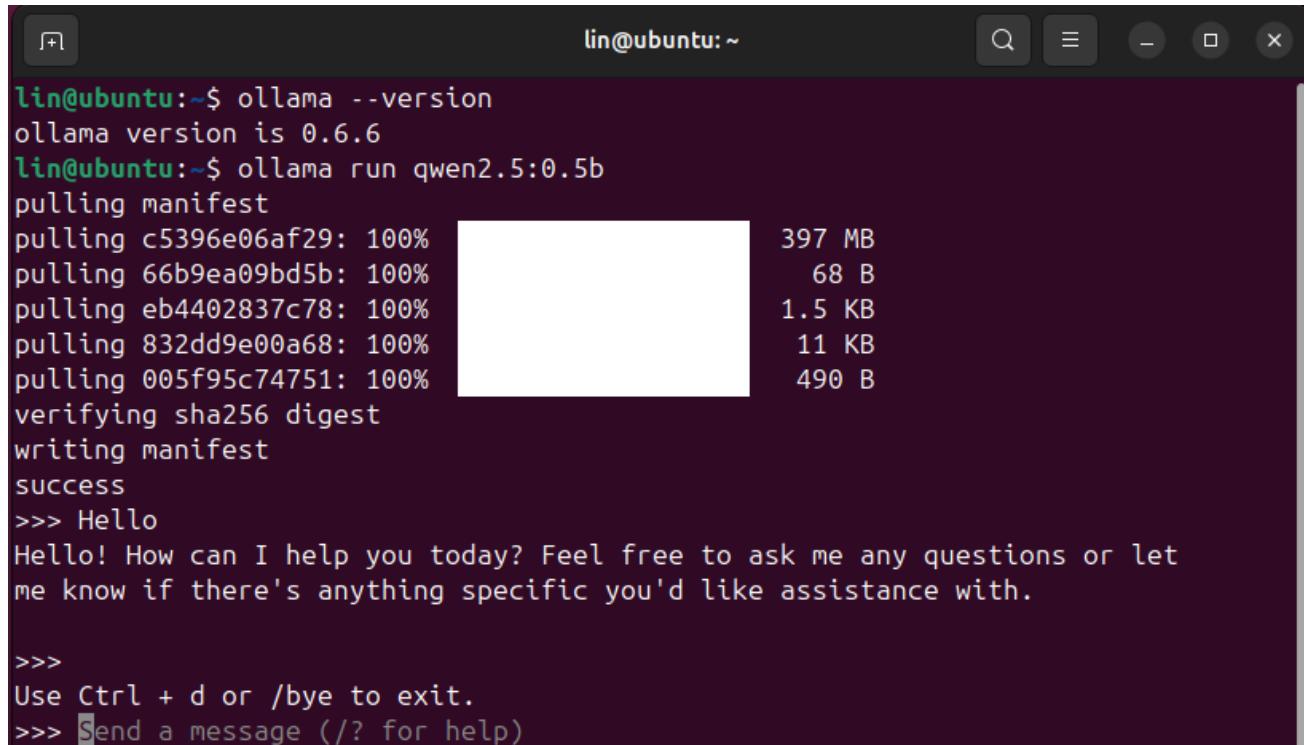
```
lin@ubuntu:~$ ollama --version
ollama version is 0.6.6
lin@ubuntu:~$
```

输入“**ollama run qwen2.5:0.5b**”，将模型下载到本地中。



```
lin@ubuntu:~$ ollama --version
ollama version is 0.6.6
lin@ubuntu:~$ ollama run qwen2.5:0.5b
pulling manifest
pulling c5396e06af29: 100% [██████████] 397 MB
pulling 66b9ea09bd5b: 100% [██████████] 68 B
pulling eb4402837c78: 100% [██████████] 1.5 KB
pulling 832dd9e00a68: 100% [██████████] 11 KB
pulling 005f95c74751: 100% [██████████] 490 B
verifying sha256 digest
writing manifest
success
>>> Send a message (/? for help)
```

当安装完成后，你可以直接在终端界面中和 qwen2.5:0.5b 进行聊天。



```
lin@ubuntu:~$ ollama --version
ollama version is 0.6.6
lin@ubuntu:~$ ollama run qwen2.5:0.5b
pulling manifest
pulling c5396e06af29: 100% [██████████] 397 MB
pulling 66b9ea09bd5b: 100% [██████████] 68 B
pulling eb4402837c78: 100% [██████████] 1.5 KB
pulling 832dd9e00a68: 100% [██████████] 11 KB
pulling 005f95c74751: 100% [██████████] 490 B
verifying sha256 digest
writing manifest
success
>>> Hello
Hello! How can I help you today? Feel free to ask me any questions or let
me know if there's anything specific you'd like assistance with.

>>>
Use Ctrl + d or /bye to exit.
>>> Send a message (/? for help)
```

您可以使用指令“**Ctrl+d**”，退出聊天模式。

您可以输入 Ollama，查看 Ollama 的使用说明。

```
lin@ubuntu:~ Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve      Start ollama
  create     Create a model from a Modelfile
  show       Show information for a model
  run        Run a model
  stop       Stop a running model
  pull       Pull a model from a registry
  push       Push a model to a registry
  list       List models
  ps         List running models
  cp         Copy a model
  rm         Remove a model
  help       Help about any command

Flags:
  -h, --help    help for ollama
  -v, --version Show version information

Use "ollama [command] --help" for more information about a command.
lin@ubuntu:~$
```



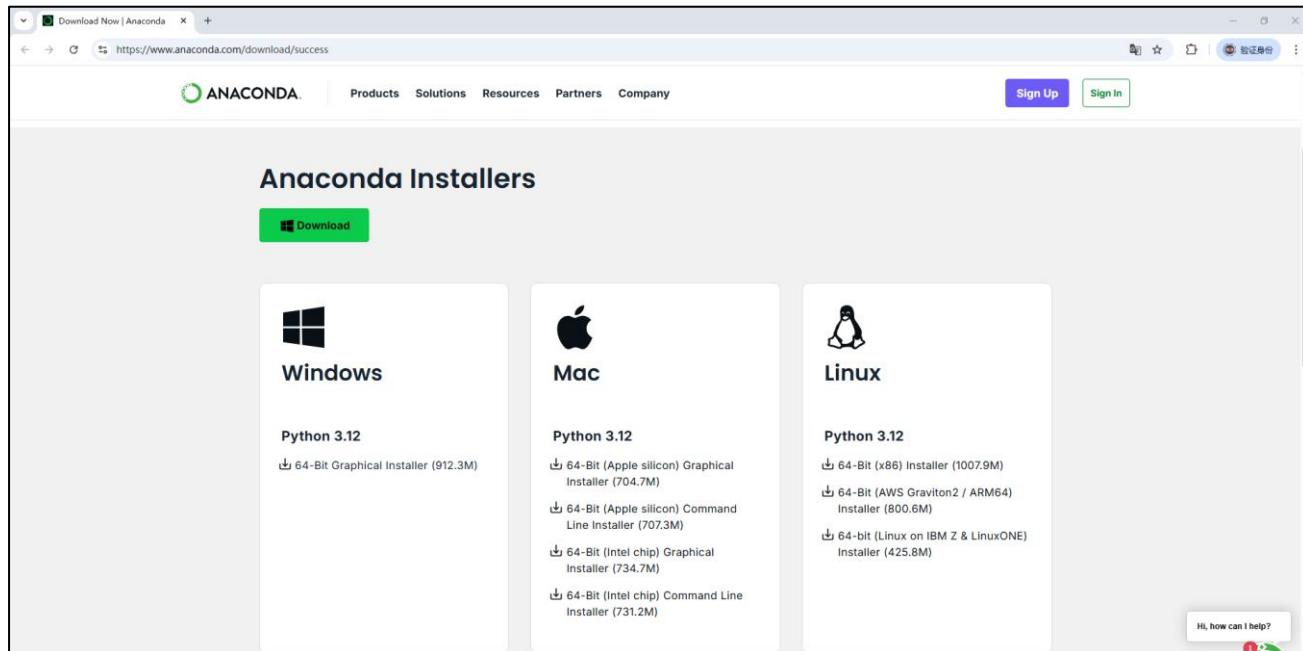
安装 Conda

xiaozhi-esp32-server 这个开源项目提供了 4 种安装方式，在本教程中，我们选择最简单的配置示例作为示范，其他使用方法请参考网站进行探索学习。

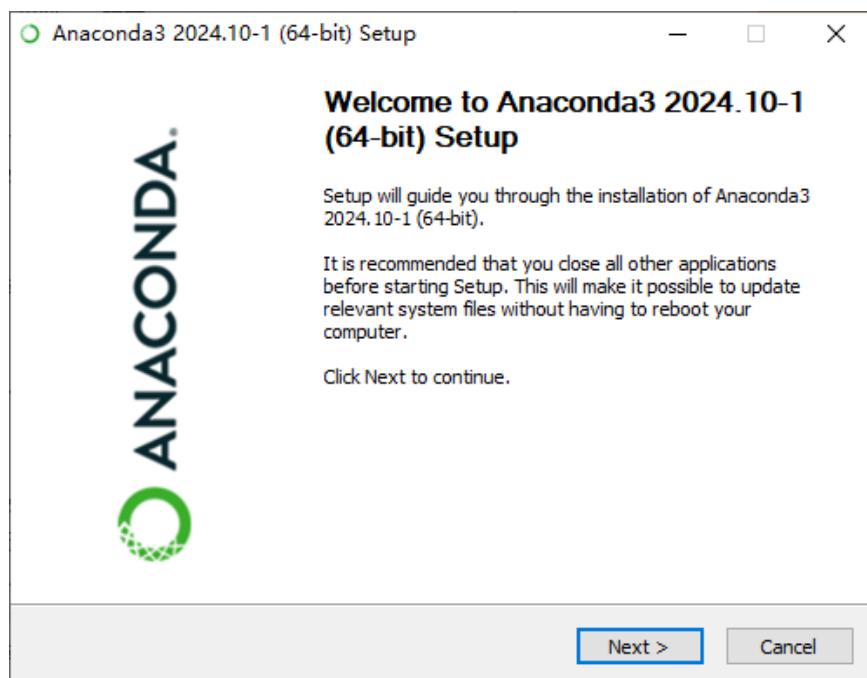
Window

本示例使用 conda 管理依赖环境。因此我们需要事先在电脑上安装 Conda 环境。如果您的电脑还没安装 Conda，您可以访问这个链接下载并安装它：<https://www.anaconda.com/download/success>

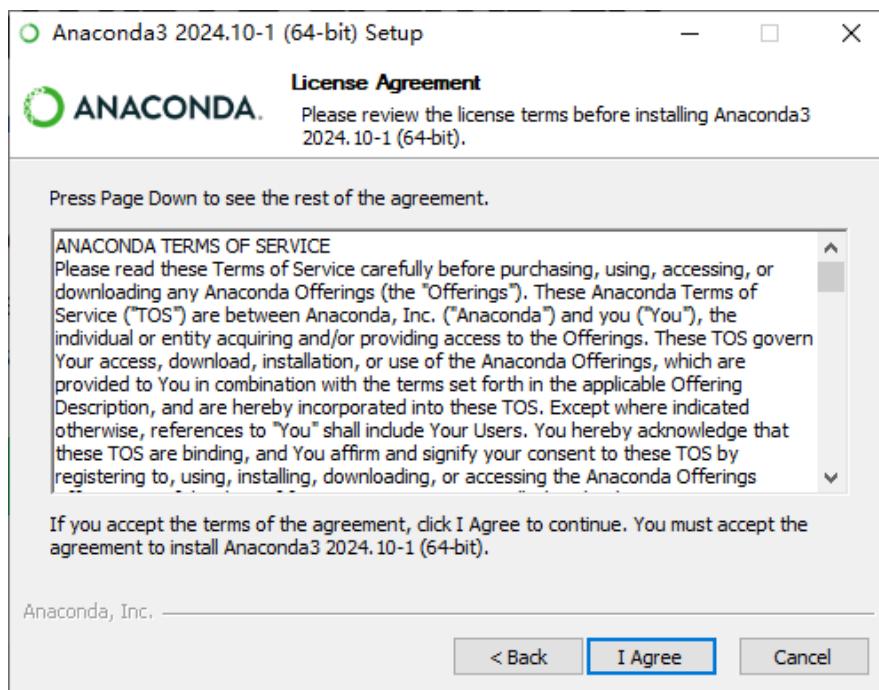
选择适合您电脑平台的软件包下载。Miniconda is an installer by Anaconda that comes preconfigured for use with the Anaconda Repository.



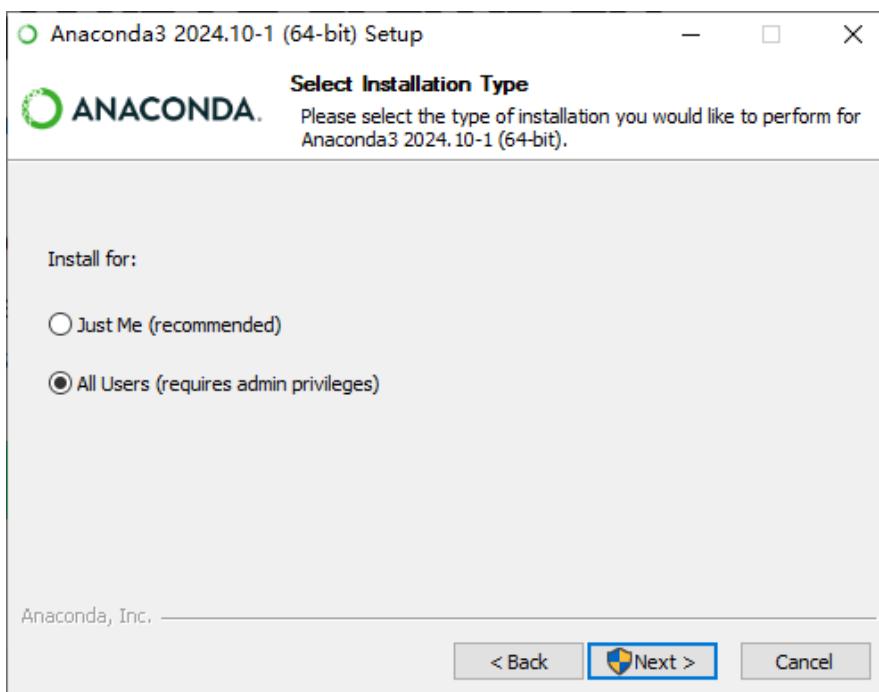
双击打开 Conda 软件，点击 Next。



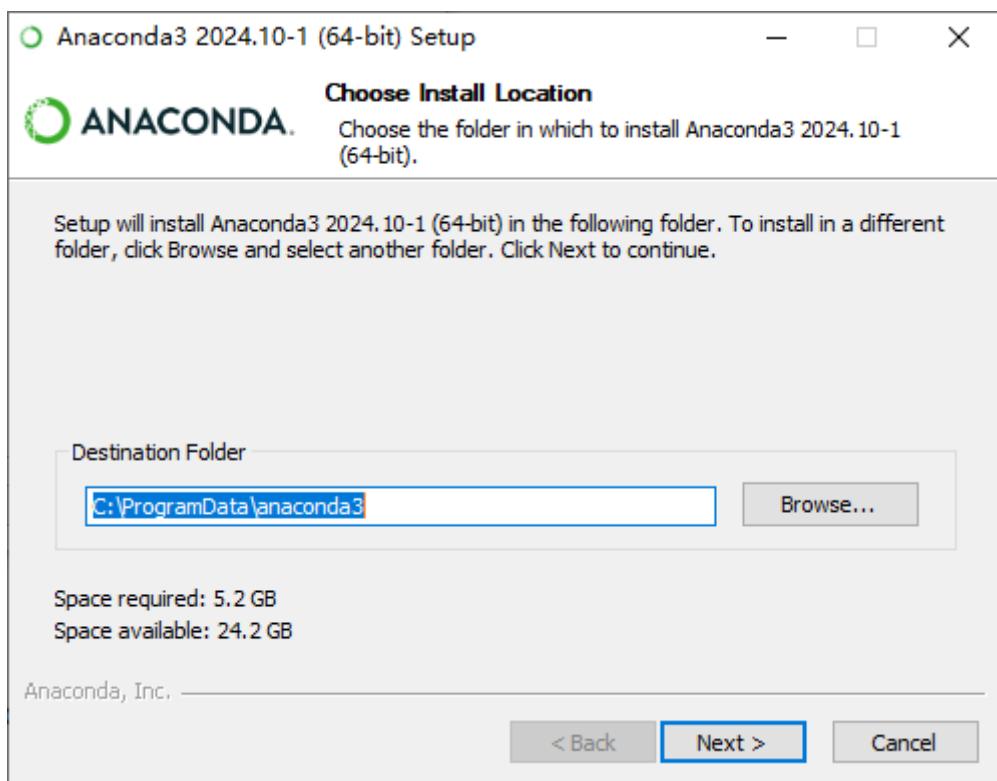
点击“*I Agree*”.



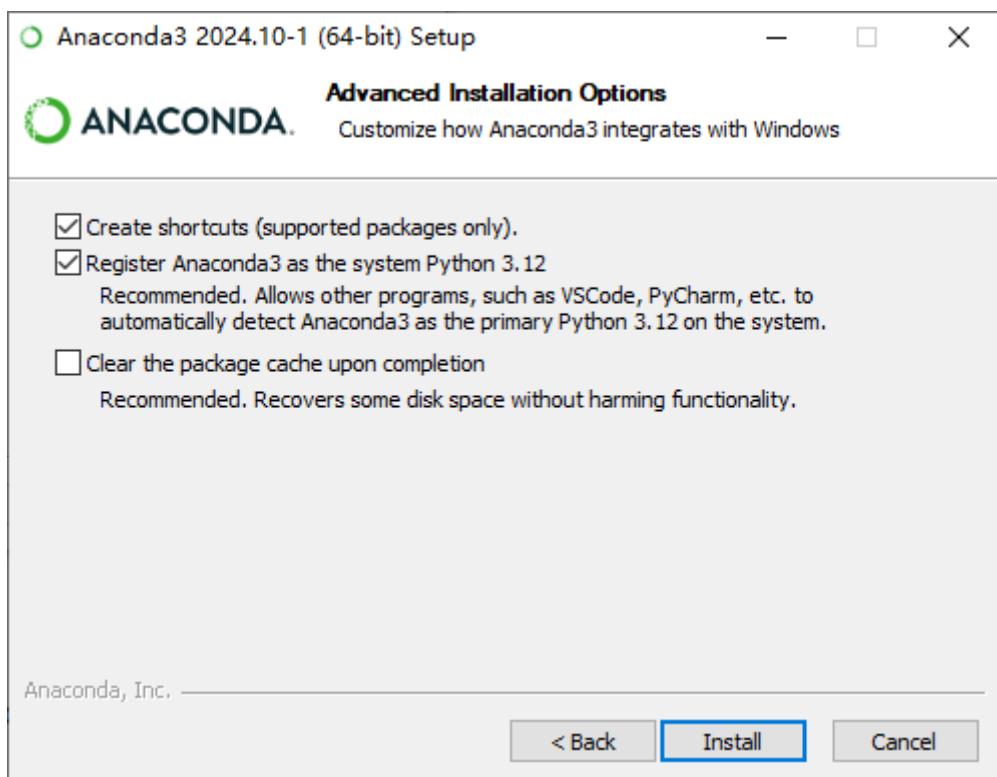
根据需求，选择合适的安装类型，一般我们选择“All Users”.



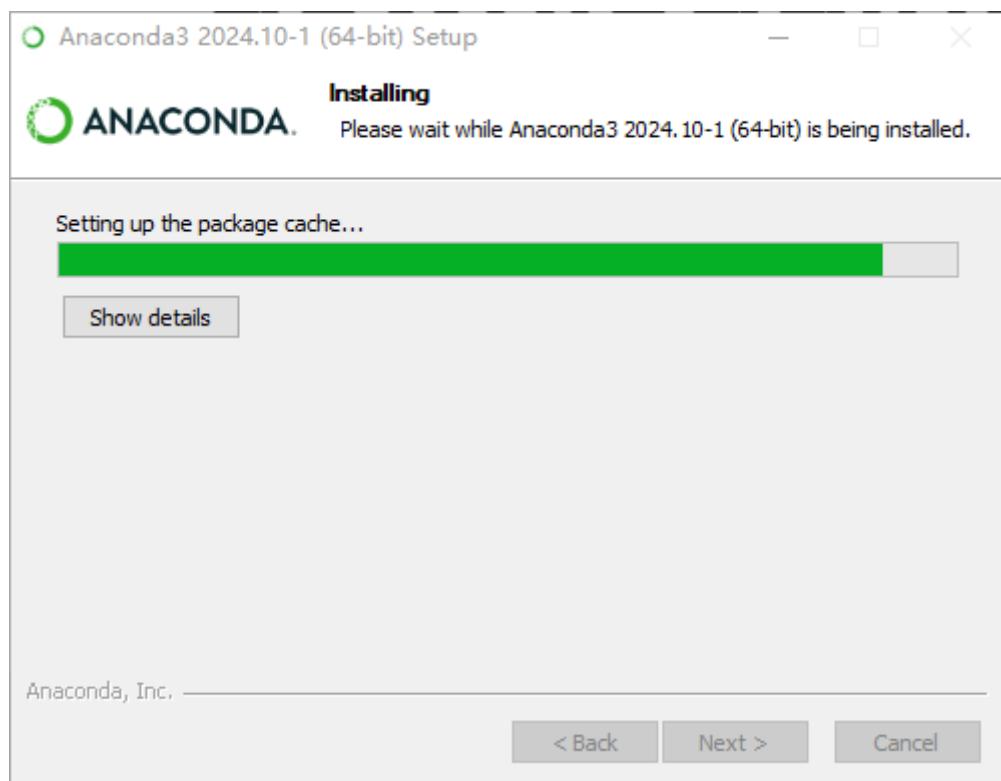
选择安装软件的位置。



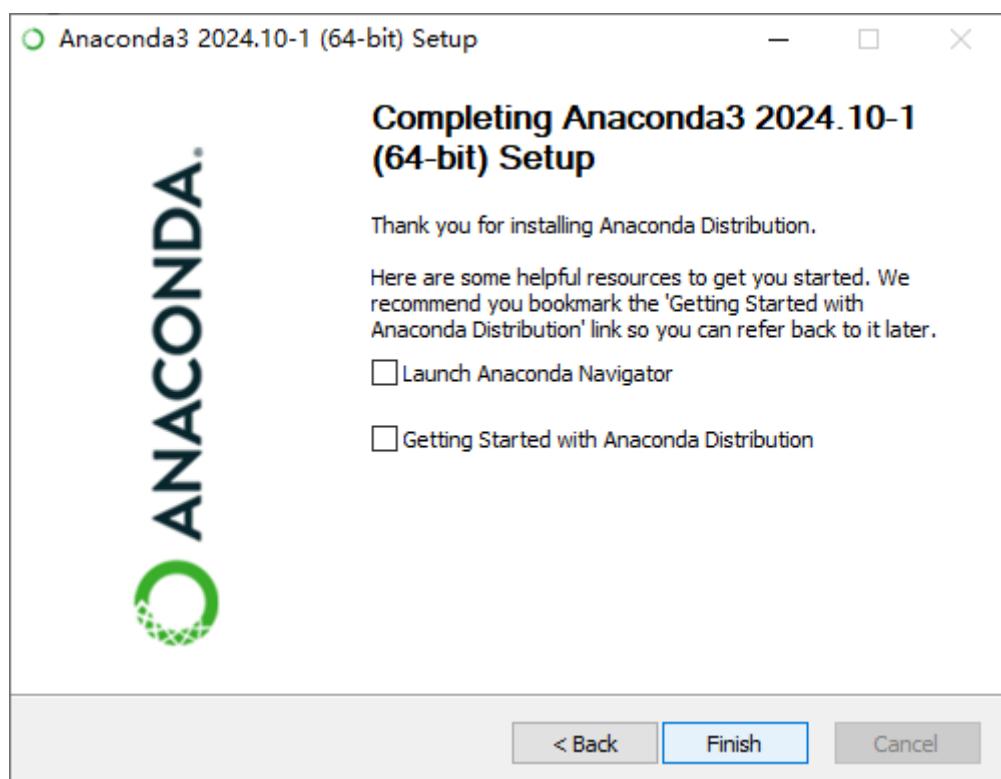
保持默认即可。点击 Install。



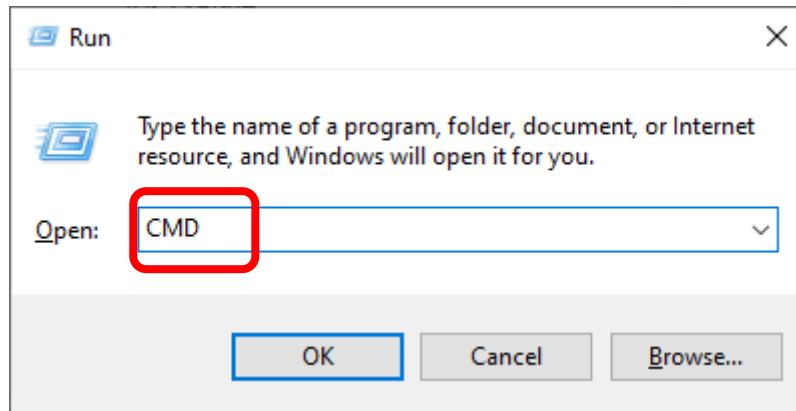
等待安装，可能需要等待一小会。



至此，软件就安装完成了。



您可以使用“Win+R”打开 Run 界面。输入“CMD”，并按下回车键，进入 CMD 界面。



输入“conda --version”，并按下回车键。如果您的 Anaconda3 已经安装完成，您可以看到下面的提示信息。

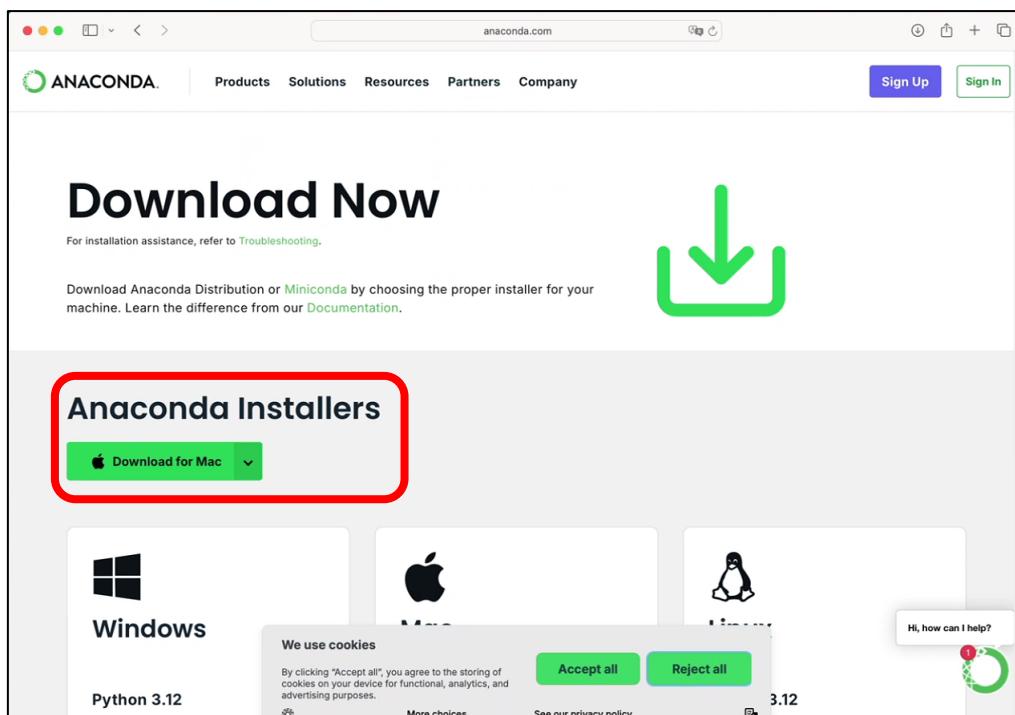
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DESKTOP-LIN>conda --version
conda 24.9.2

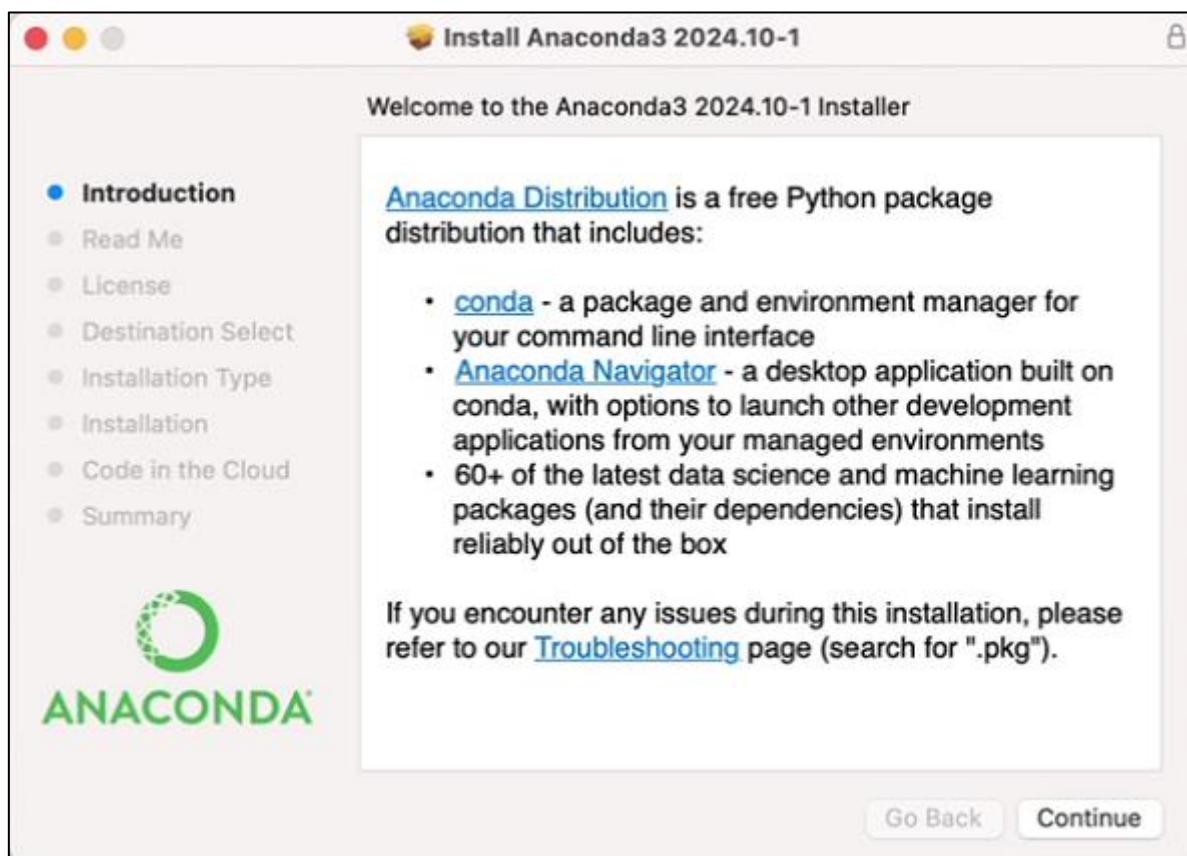
C:\Users\DESKTOP-LIN>
```

MAC

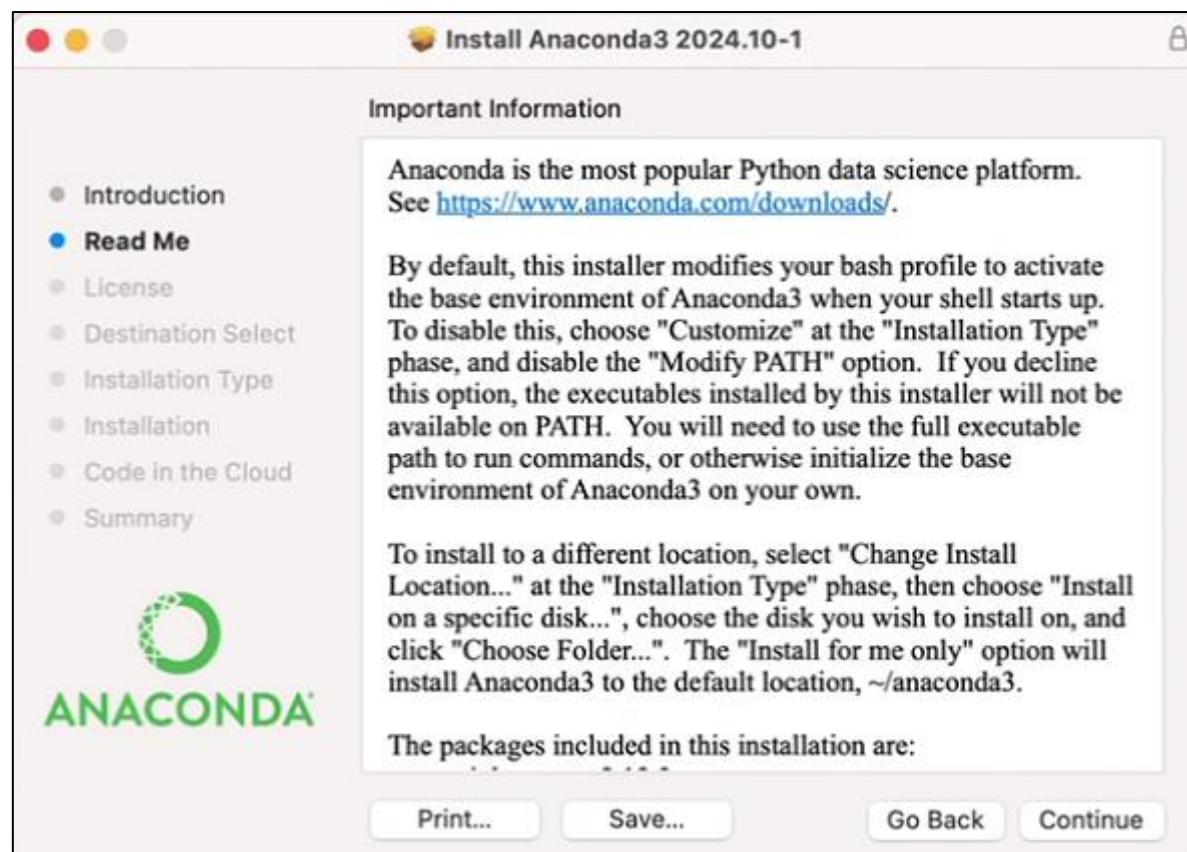
本示例使用 conda 管理依赖环境。因此我们需要事先在电脑上安装 Conda 环境。如果您的电脑还没安装 Conda，您可以访问这个链接下载并安装它：<https://www.anaconda.com/download/success>
选择适合您电脑平台的软件包下载。Miniconda is an installer by Anaconda that comes preconfigured for use with the Anaconda Repository.



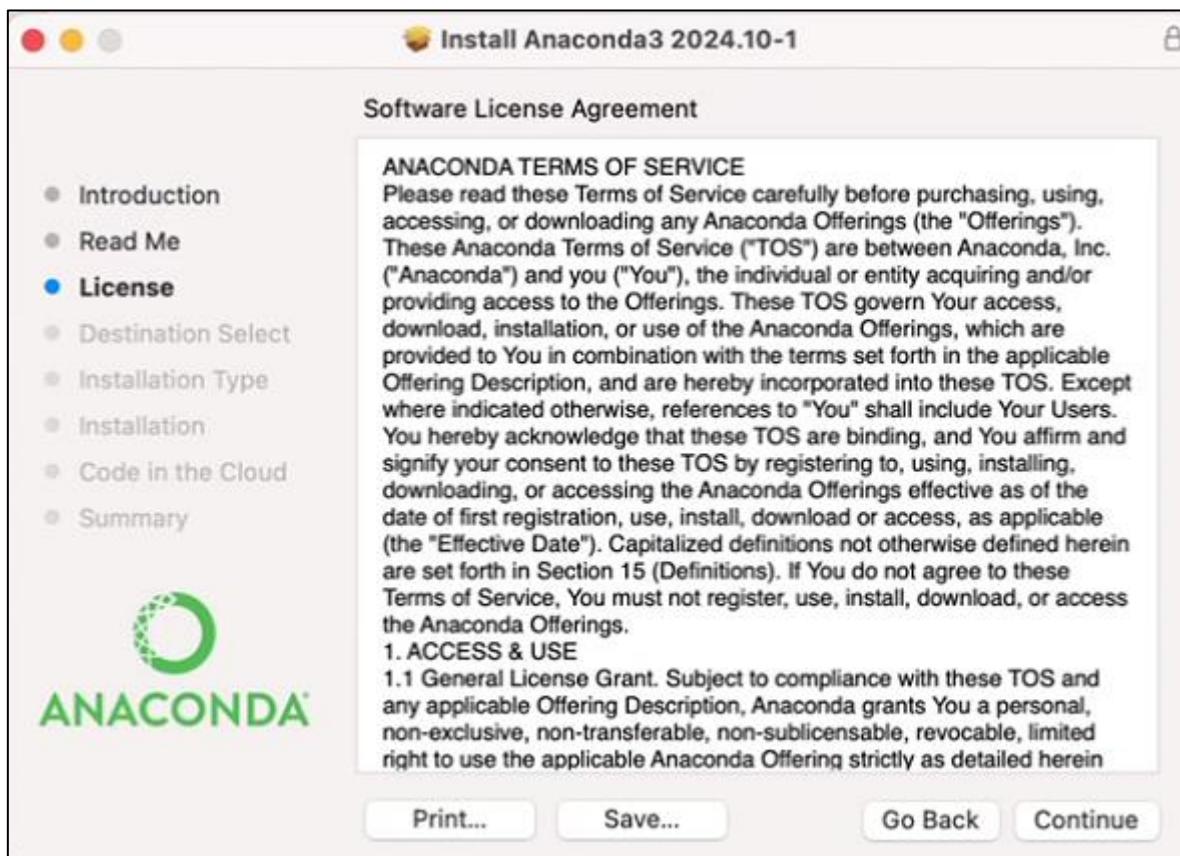
双击打开 Conda 软件，点击 Continue.



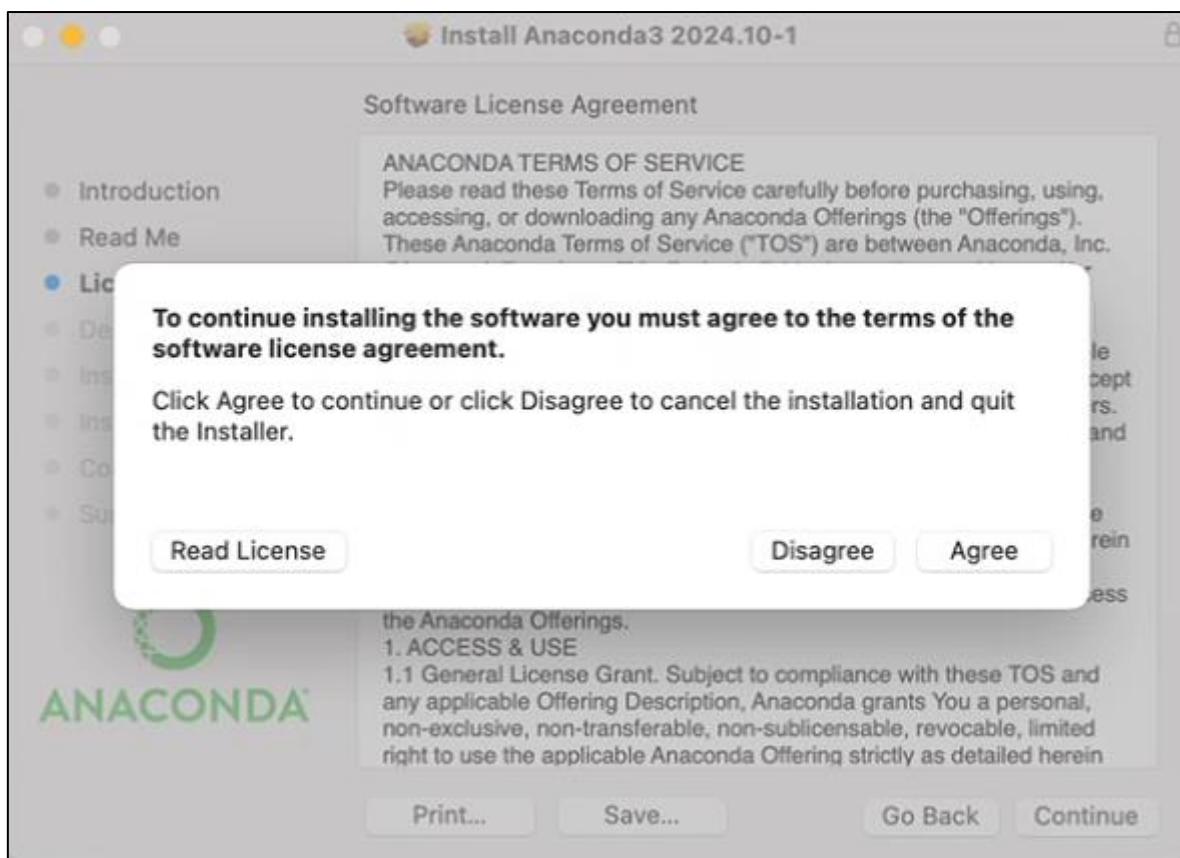
点击 Continue.



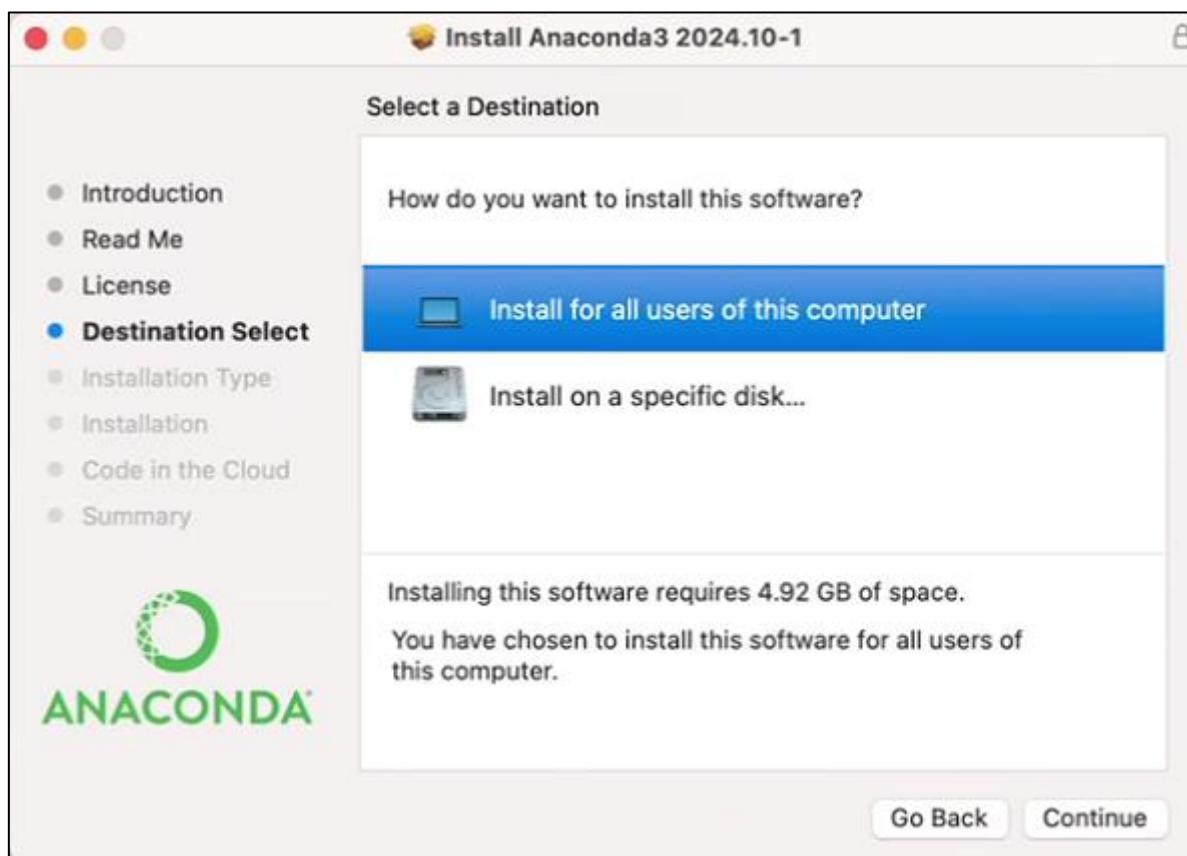
点击 Continue.



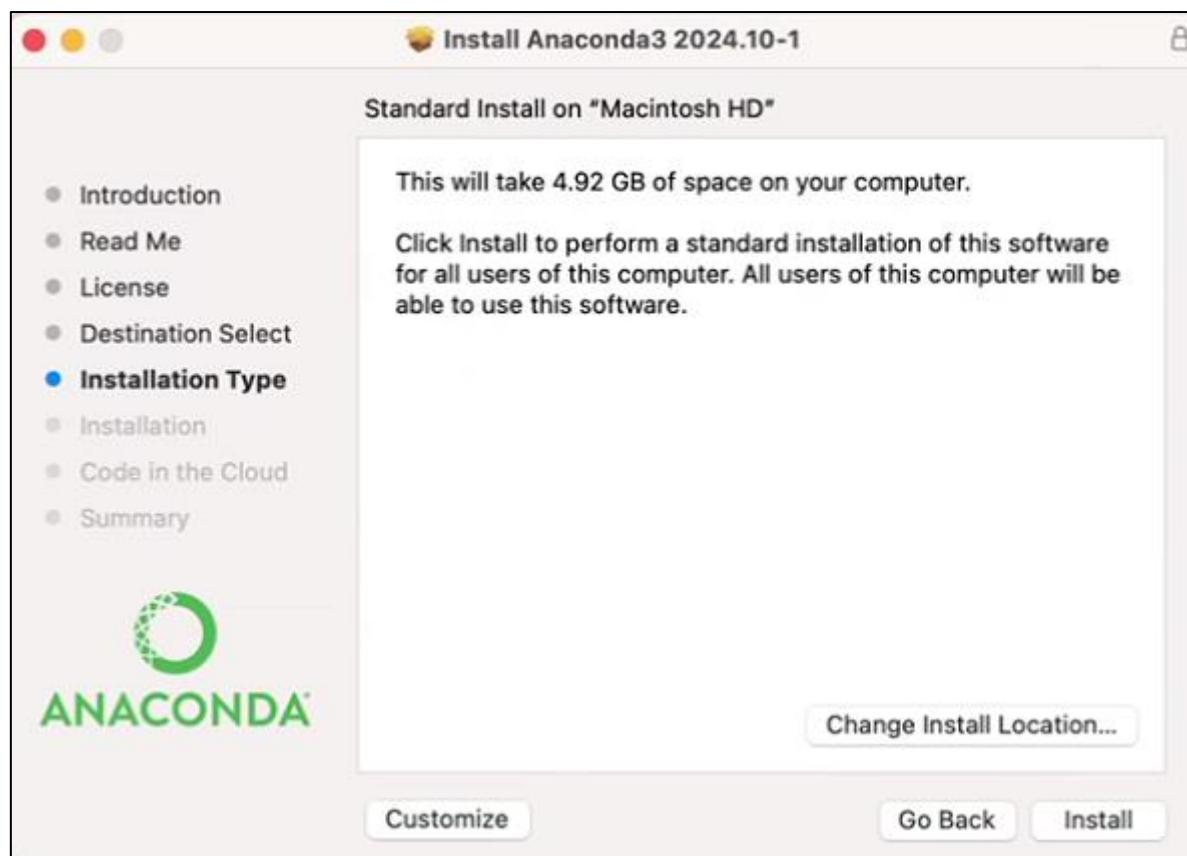
点击 Agree.



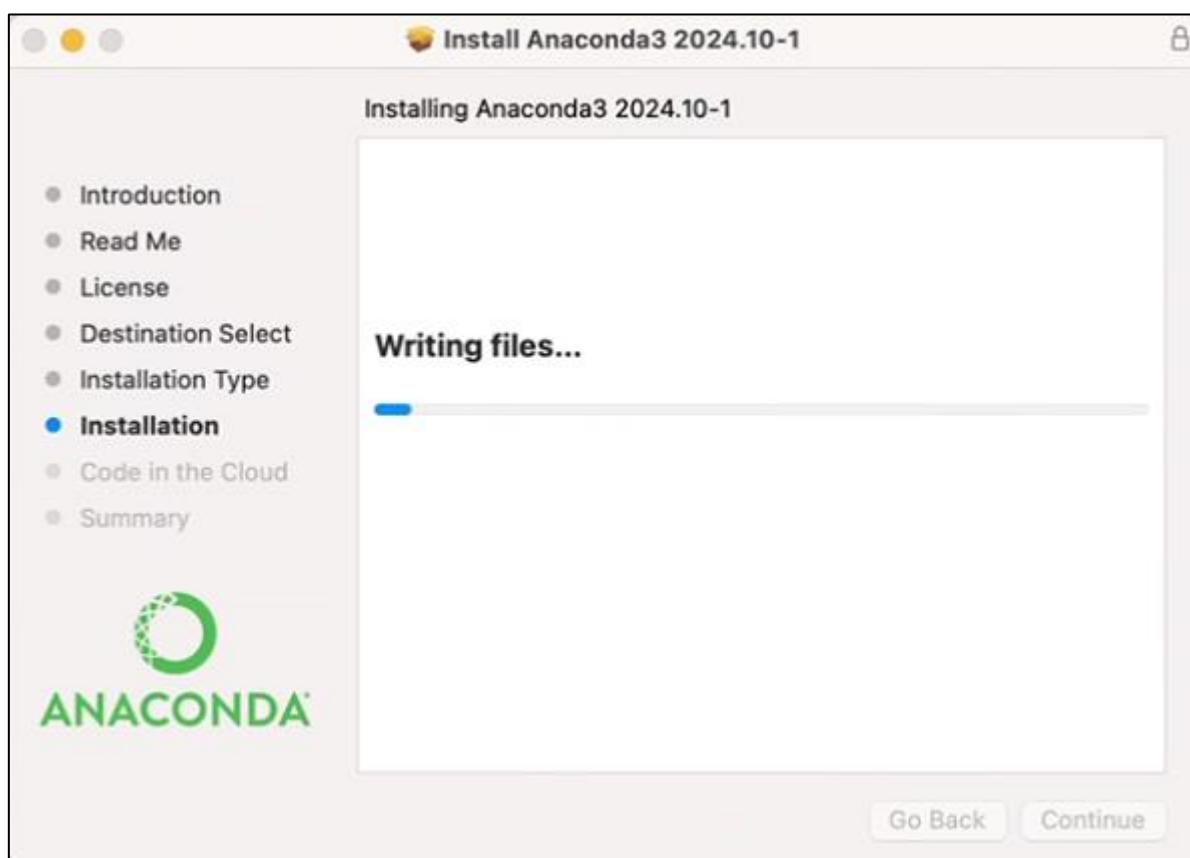
保持默认，点击 Continue.



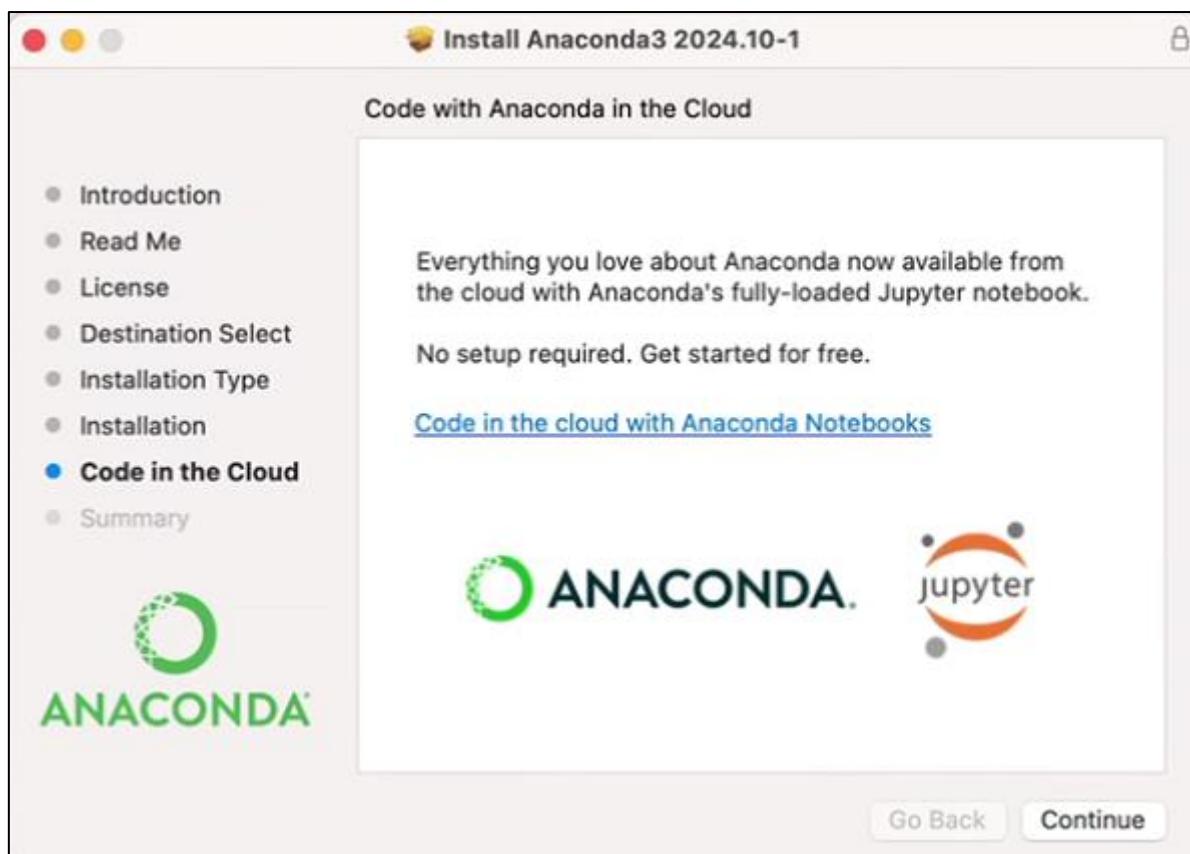
点击 Install.



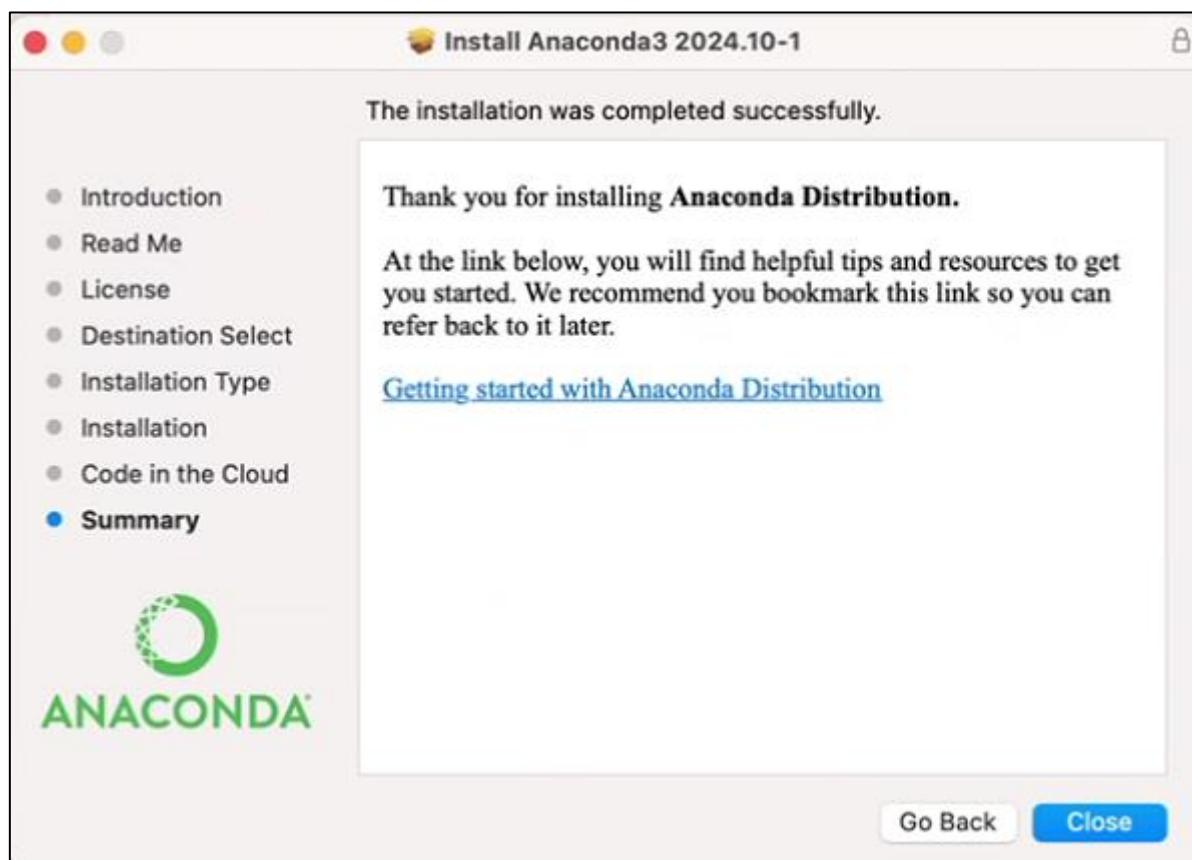
这里需要等待几分钟。



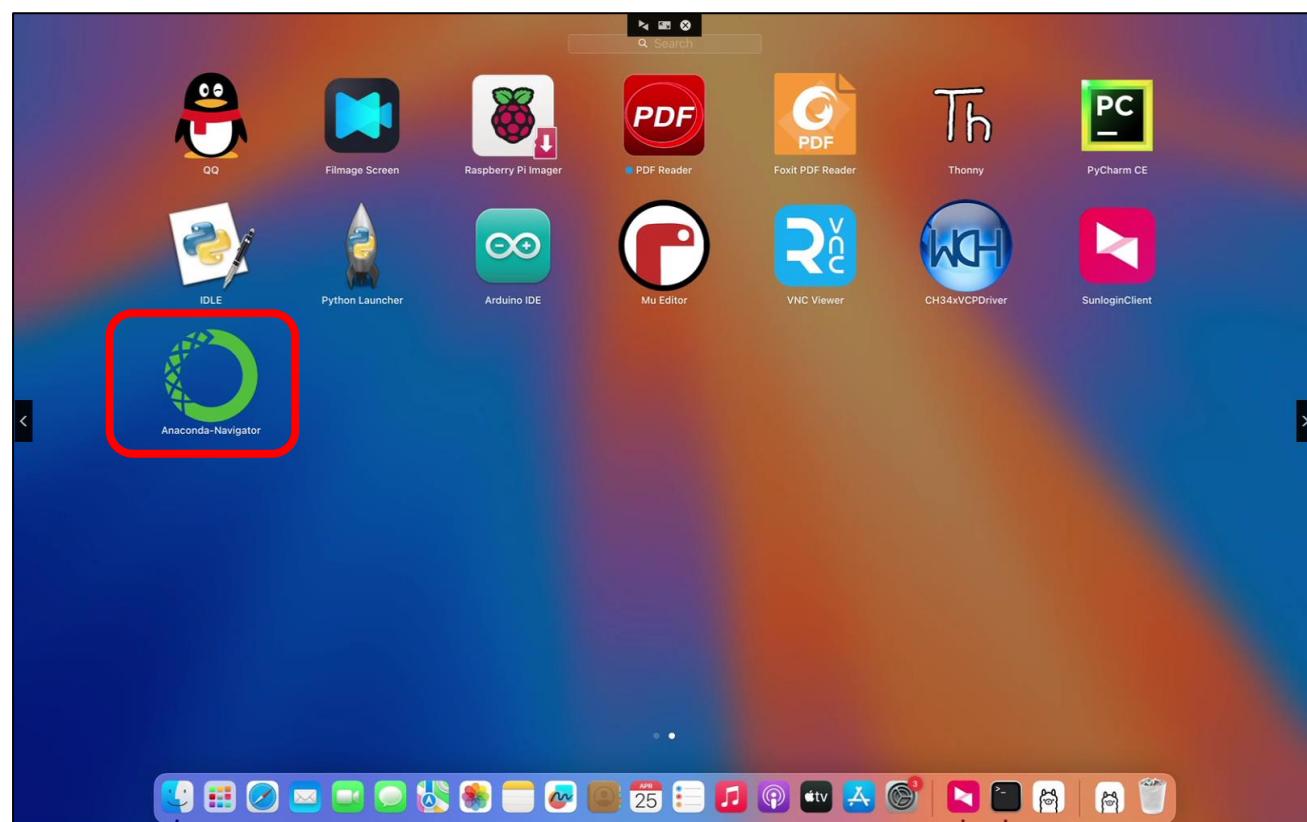
点击 Continue.



点击 Close.



至此，您可以安装了 Conda。您可以在您的应用列表中找到它。



双击运行它，这一步不会有任何现象。

再次打开终端。您可以发现，出现了"(base)"的提示词。您也可以通过指令"conda --version"查看 conda 的版本。

```
freenove -- zsh - 80x24
Last login: Fri Apr 25 11:16:30 on ttys000
(base) freenove@PandeMacBook-Air ~ % conda --version
conda 24.9.2
(base) freenove@PandeMacBook-Air ~ %
```

您可以使用 conda -h 来查看更多的使用方法。

```
freenove -- zsh - 80x24
init           Initialize conda for shell interaction.
inspect        Tools for inspecting conda packages.
install         Install a list of packages into a specified conda
                environment.
list            List installed packages in a conda environment.
metapackage    Specialty tool for generating conda metapackage.
notices        Retrieve latest channel notifications.
pack            See 'conda pack --help'.
package         Create low-level conda packages. (EXPERIMENTAL)
remove (uninstall) Remove a list of packages from a specified conda
                environment.
rename          Rename an existing environment.
render          Expand a conda recipe into a platform-specific recipe.
repo            See 'conda repo --help'.
repoquery       Advanced search for repodata.
run             Run an executable in a conda environment.
search          Search for packages and display associated information
                using the MatchSpec format.
server          See 'conda server --help'.
skeleton        Generate boilerplate conda recipes.
token           See 'conda token --help'.
update (upgrade) Update conda packages to the latest compatible
                    version.
(base) freenove@PandeMacBook-Air ~ %
```

如果您是第一次使用 conda，您需要使用指令"conde init"，让安装的 conda 环境初始化并生效。

conda init

您可以使用 conda activate 来激活虚拟环境。或者通过 conda deactivate 来退出虚拟环境。

conda activate

conda deactivate

```
freenove -- zsh - 80x24
freenove@PandeMacBook-Air ~ % conda activate
(base) freenove@PandeMacBook-Air ~ % conda deactivate
freenove@PandeMacBook-Air ~ %
```

如果您想要打开终端就自动进入虚拟环境，您可以使用"conda config --set auto_activate_base true"指令。

如果您不想要打开终端就自动进入虚拟环境，您可以使用"conda config --set auto_activate_base flase"指令。

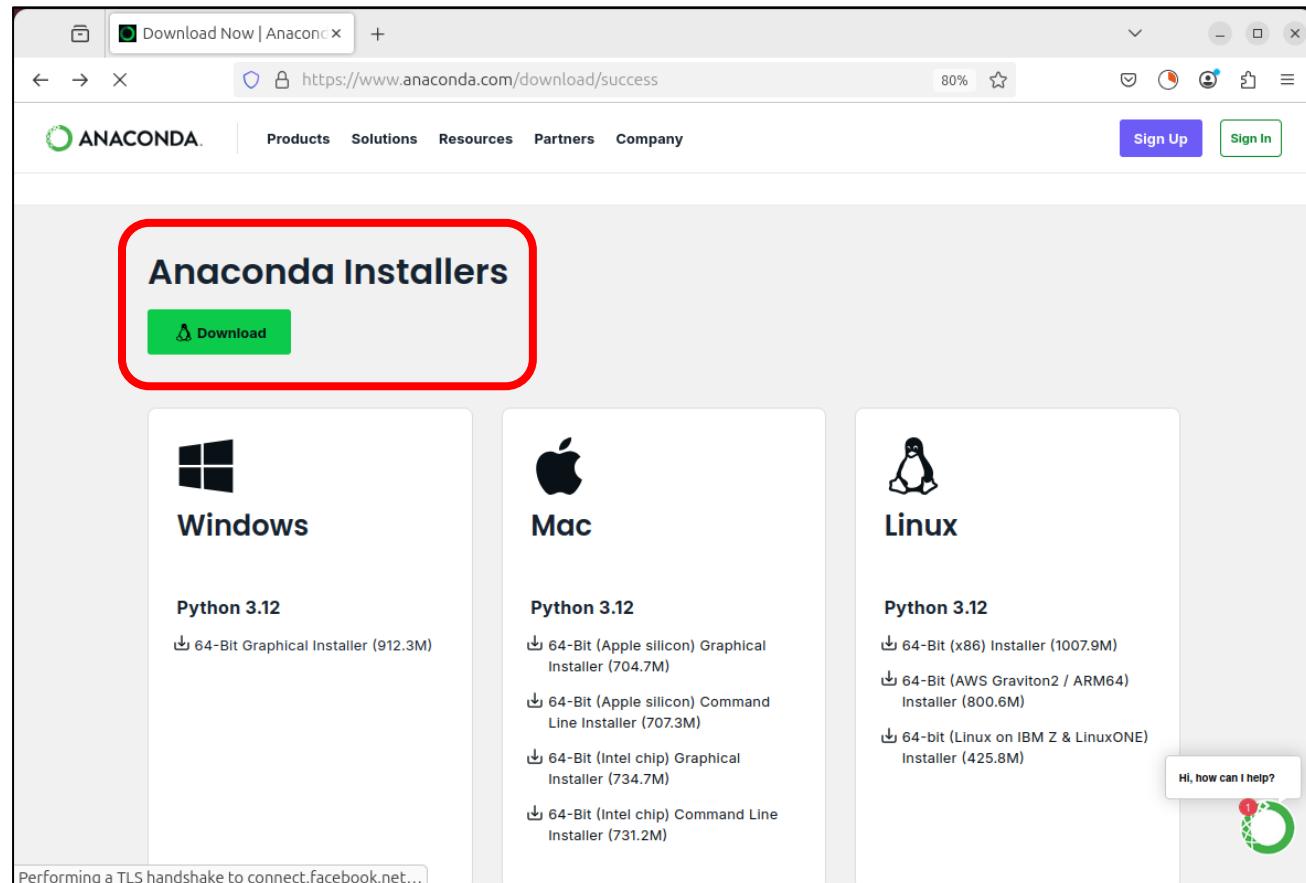
conda config --set auto_activate_base false

conda config --set auto_activate_base true

Linux

本示例使用 conda 管理依赖环境。因此我们需要事先在电脑上安装 Conda 环境。如果您的电脑还没安装 Conda，您可以访问这个链接下载并安装它：<https://www.anaconda.com/download/success>

选择适合您电脑平台的软件包下载。Miniconda is an installer by Anaconda that comes preconfigured for use with the Anaconda Repository.

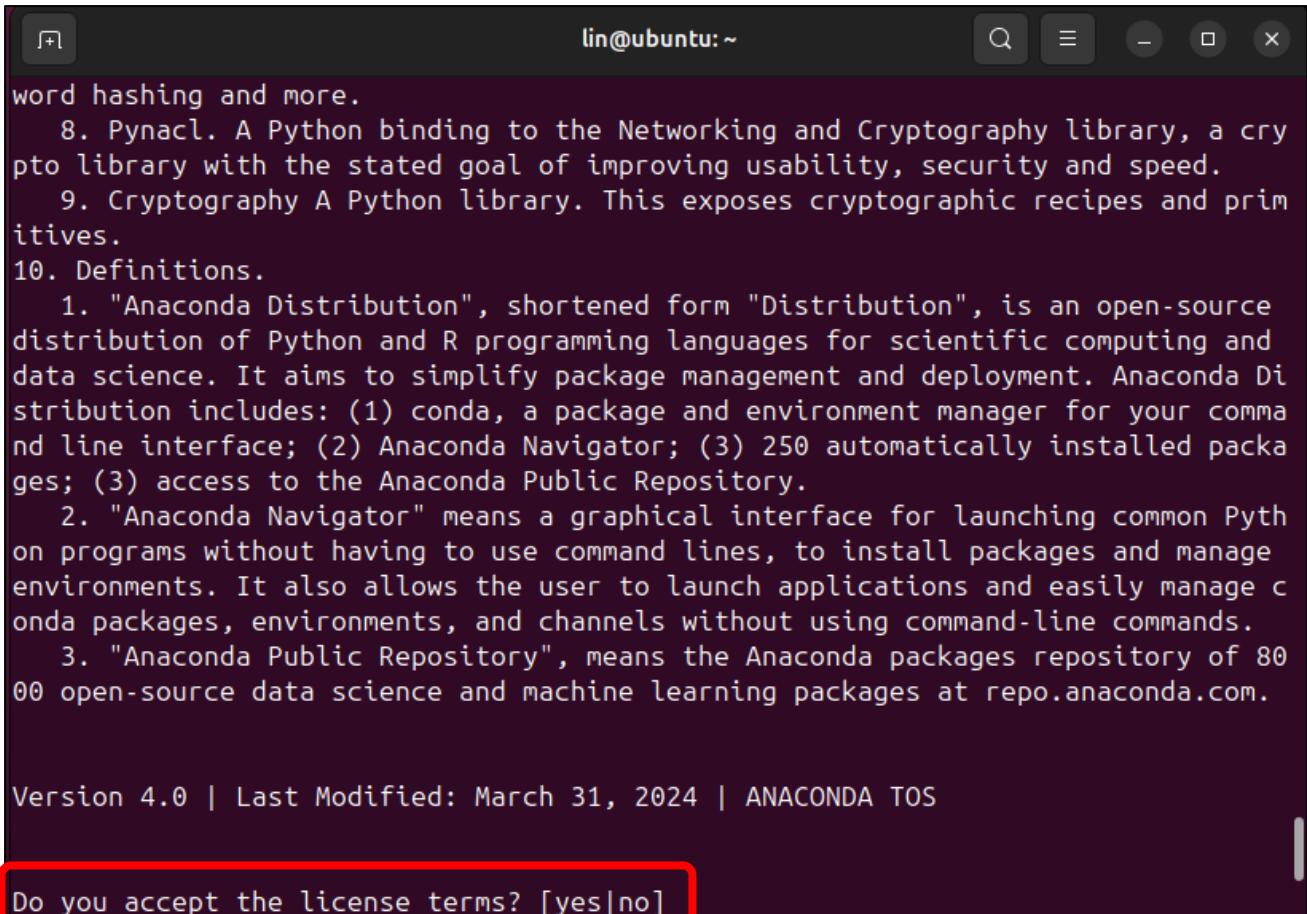


此处我下载的文件名称为“Anaconda3-2024.10-1-Linux-x86_64.sh”，不同的电脑，名称可能不同。
打开终端，使用下面的指令安装 Anaconda。

```
sh Anaconda3-2024.10-1-Linux-x86_64.sh
```

A screenshot of a terminal window on an Ubuntu system. The command 'ls' is run to list files in the current directory, showing 'Anaconda3-2024.10-1-Linux-x86_64.sh', 'shared', 'snap', and 'Upload_Xiaozhi_Bin'. Then, the command 'sh Anaconda3-2024.10-1-Linux-x86_64.sh' is executed, indicated by a cursor in the command line.

按住回车键不松开，直到出现下方的提示信息。输入“Yes”。



```
lin@ubuntu:~ word hashing and more.
8. Pynacl. A Python binding to the Networking and Cryptography library, a crypto library with the stated goal of improving usability, security and speed.
9. Cryptography A Python library. This exposes cryptographic recipes and primitives.
10. Definitions.
1. "Anaconda Distribution", shortened form "Distribution", is an open-source distribution of Python and R programming languages for scientific computing and data science. It aims to simplify package management and deployment. Anaconda Distribution includes: (1) conda, a package and environment manager for your command line interface; (2) Anaconda Navigator; (3) 250 automatically installed packages; (3) access to the Anaconda Public Repository.
2. "Anaconda Navigator" means a graphical interface for launching common Python programs without having to use command lines, to install packages and manage environments. It also allows the user to launch applications and easily manage conda packages, environments, and channels without using command-line commands.
3. "Anaconda Public Repository", means the Anaconda packages repository of 8000 open-source data science and machine learning packages at repo.anaconda.com.

Version 4.0 | Last Modified: March 31, 2024 | ANACONDA TOS

Do you accept the license terms? [yes|no]
```

选择您的安装位置。按下回车键，使用默认安装位置即可。

```
Anaconda3 will now be installed into this location:
/home/lin/anaconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/home/lin/anaconda3] >>>
```

安装过程需要网络，请确保您的网络稳定，并耐心等待几分钟。直到您的界面出现下方的提示信息。
请注意，这里需要输入 Yes。

```
Do you wish to update your shell profile to automatically initialize conda?
This will activate conda on startup and change the command prompt when activated.
.
If you'd prefer that conda's base environment not be activated on startup,
run the following command when conda is activated:

conda config --set activate_base false
You can undo this by running `conda init --reverse $SHELL`? [yes|no]
[no] >>> yes
```

出现下方的提示，说明您已经成功安装 conda.

```
You can undo this by running `conda init --reverse $SHELL`? [yes|no]
[no] >>> yes
no change    /home/lin/anaconda3/condabin/conda
no change    /home/lin/anaconda3/bin/conda
no change    /home/lin/anaconda3/bin/conda-env
no change    /home/lin/anaconda3/bin/activate
no change    /home/lin/anaconda3/bin/deactivate
no change    /home/lin/anaconda3/etc/profile.d/conda.sh
no change    /home/lin/anaconda3/etc/fish/conf.d/conda.fish
no change    /home/lin/anaconda3/shell/condabin/Conda.psm1
no change    /home/lin/anaconda3/shell/condabin/conda-hook.ps1
no change    /home/lin/anaconda3/lib/python3.12/site-packages/xontrib/conda.xsh
no change    /home/lin/anaconda3/etc/profile.d/conda.csh
no change    /home/lin/.bashrc
No action taken.
Thank you for installing Anaconda3!
lin@ubuntu:~$
```

如果您想要打开终端就自动进入虚拟环境，您可以使用“`conda config --set auto_activate_base true`”指令。
如果您不想要打开终端就自动进入虚拟环境，您可以使用“`conda config --set auto_activate_base false`”指令。

```
conda config --set auto_activate_base false
conda config --set auto_activate_base true
```

这里，我们建议使用“`conda config --set auto_activate_base false`”。

```
lin@ubuntu:~$ conda config --set auto_activate_base false
lin@ubuntu:~$
```

重启终端，您可以通过指令查看 conda 的版本号。

```
conda --version
```

```
lin@ubuntu:~$ conda --version
conda 24.9.2
lin@ubuntu:~$
```

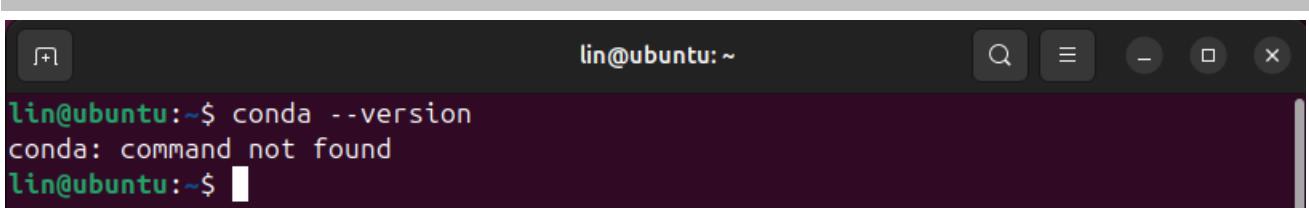
您可以使用 `conda activate` 来激活虚拟环境。或者通过 `conda deactivate` 来退出虚拟环境。

```
conda activate
conda deactivate
```

```
lin@ubuntu:~$ conda --version
conda 24.9.2
lin@ubuntu:~$ conda activate
(base) lin@ubuntu:~$ conda deactivate
lin@ubuntu:~$
```

如果您查看 conda 的版本号时报错如下。

```
conda --version
```



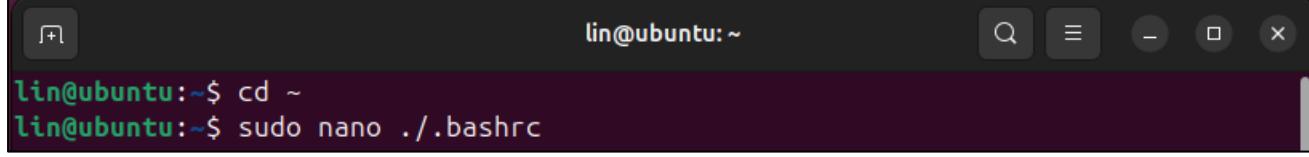
```
lin@ubuntu:~$ conda --version
conda: command not found
lin@ubuntu:~$
```

这说明您的文件 Conda 虽然已经安装，但是没有添加到 PATH 中。

请按照下面的步骤将 conda 添加到 PATH 中。

使用 nano 编辑".bashrc"文件.

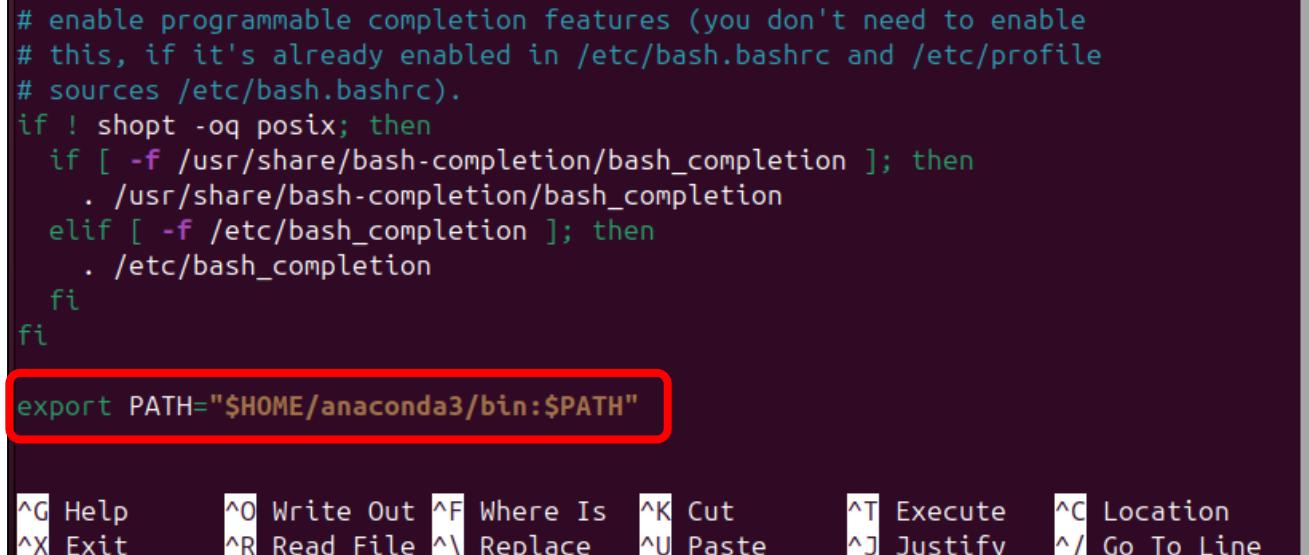
```
cd ~
sudo nano ./bashrc
```



```
lin@ubuntu:~$ cd ~
lin@ubuntu:~$ sudo nano ./bashrc
```

在文件的最下方，添加一行内容。

```
export PATH="$HOME/anaconda3/bin:$PATH"
```



```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

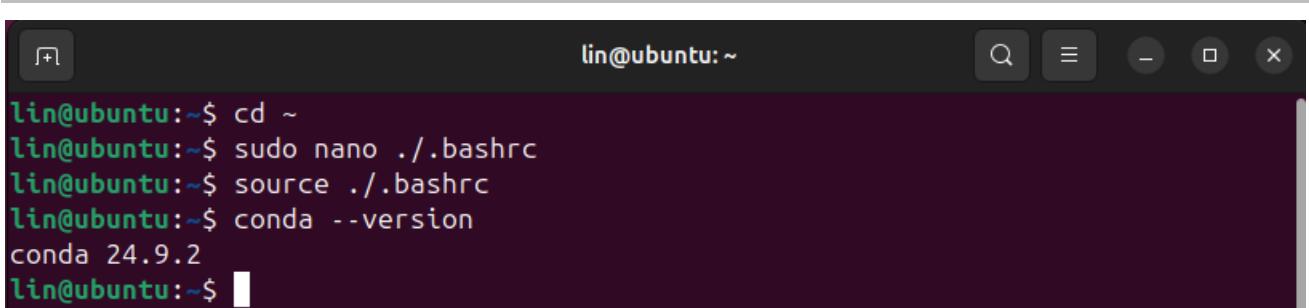
export PATH="$HOME/anaconda3/bin:$PATH"
```

^G Help	^O Write Out	^F Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^\\ Replace	^U Paste	^J Justify	^/ Go To Line

使用 Ctrl+O 保存文件，使用 Ctrl+X 退出编辑器。

使用 source 指令，让文件生效。并查看 conda 的版本。

```
source ./bashrc
conda --version
```



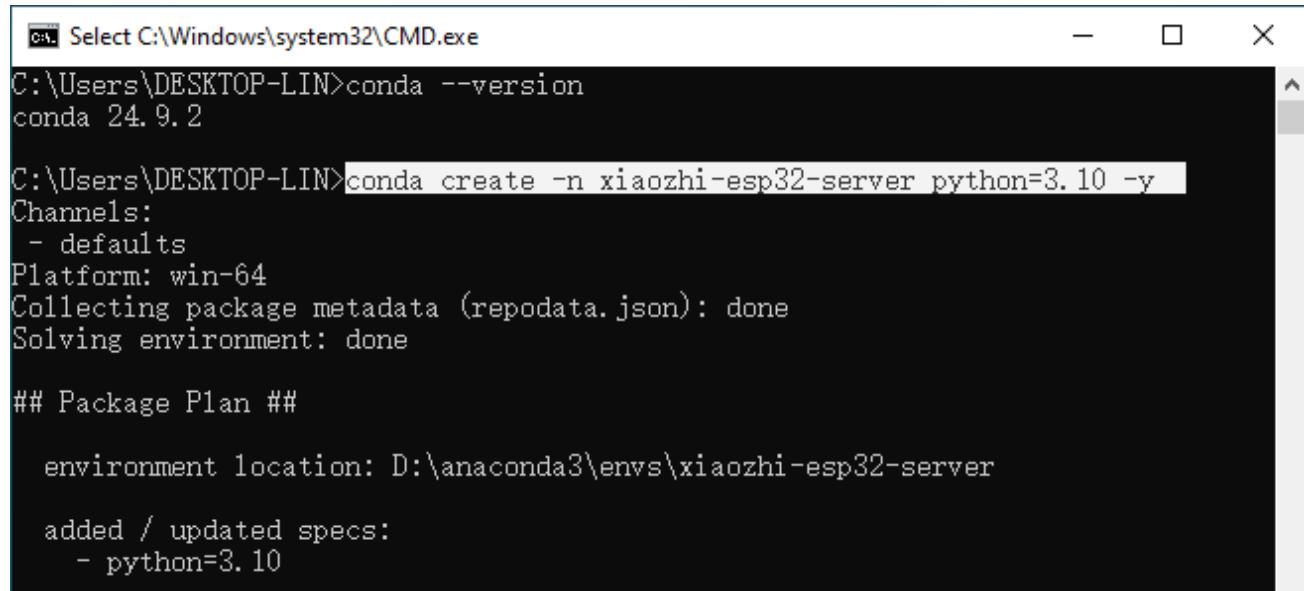
```
lin@ubuntu:~$ cd ~
lin@ubuntu:~$ sudo nano ./bashrc
lin@ubuntu:~$ source ./bashrc
lin@ubuntu:~$ conda --version
conda 24.9.2
lin@ubuntu:~$
```

部署虚拟环境

请注意，部署虚拟环境的指令，在 Window，MAC，Ubuntu 中是通用的，这里以 Window 举例，其他平台操作相同。

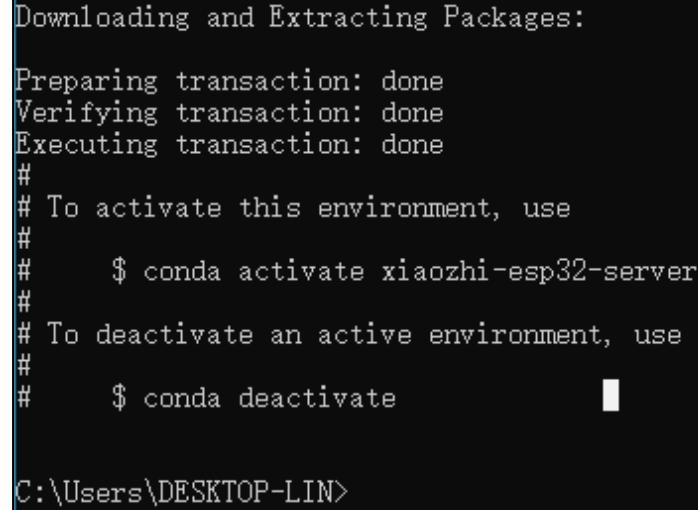
打开 CMD 界面，使用指令创建一个带有 python3.10 的虚拟环境，并命名为“xiaozihi-esp32-server”

```
conda create -n xiaozihi-esp32-server python=3.10 -y
```



```
Conda 安装于 C:\Windows\system32\cmd.exe  
C:\Users\DESKTOP-LIN>conda --version  
conda 24.9.2  
  
C:\Users\DESKTOP-LIN>conda create -n xiaozihi-esp32-server python=3.10 -y  
Channels:  
- defaults  
Platform: win-64  
Collecting package metadata (repodata.json): done  
Solving environment: done  
  
## Package Plan ##  
  
environment location: D:\anaconda3\envs\xiaozihi-esp32-server  
  
added / updated specs:  
- python=3.10
```

当看到下面的消息，说明虚拟环境已经创建完成。



```
Downloading and Extracting Packages:  
  
Preparing transaction: done  
Verifying transaction: done  
Executing transaction: done  
#  
# To activate this environment, use  
#  
#     $ conda activate xiaozihi-esp32-server  
#  
# To deactivate an active environment, use  
#  
#     $ conda deactivate  
  
C:\Users\DESKTOP-LIN>
```

如果您想删除这个虚拟环境，请使用下面的指令。

```
conda remove -n xiaozhi-esp32-server --all -y
```

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. The command 'conda remove -n xiaozhi-esp32-server --all -y' is entered and executed. The output shows the removal of packages from the environment 'D:\anaconda3\envs\xiaozhi-esp32-server'. It lists the packages being removed, including bzip2, ca-certificates, libffi, openssl, pip, python, setuptools, sqlite, tk, tzdata, vc, vs2015_runtime, wheel, xz, and zlib. After the transaction is prepared, verified, and executed, the prompt returns to 'C:\Users\Desktop-LIN>'.

```
C:\Windows\system32\cmd.exe
C:\Users\Desktop-LIN>conda remove -n xiaozhi-esp32-server --all -y
Remove all packages in environment D:\anaconda3\envs\xiaozhi-esp32-server:

## Package Plan ##

environment location: D:\anaconda3\envs\xiaozhi-esp32-server

The following packages will be REMOVED:

bzip2-1.0.8-h2bbff1b_6
ca-certificates-2025.2.25-haa95532_0
libffi-3.4.4-hd77b12b_1
openssl-3.0.16-h3f729d1_0
pip-25.0-py310haa95532_0
python-3.10.16-h4607a30_1
setuptools-75.8.0-py310haa95532_0
sqlite-3.45.3-h2bbff1b_0
tk-8.6.14-h0416ee5_0
tzdata-2025a-h04d1e81_0
vc-14.42-haa95532_5
vs2015_runtime-14.42.34433-hbfb602d_5
wheel-0.45.1-py310haa95532_0
xz-5.6.4-h4754444_1
zlib-1.2.13-h8cc25b3_1

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

C:\Users\Desktop-LIN>
```

同样，您随时可以使用这两个指令，开启和关闭虚拟环境：

```
conda activate xiaozhi-esp32-server
conda deactivate
```

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe - conda deactivate'. The command 'conda activate xiaozhi-esp32-server' is entered, which changes the prompt to '(xiaozhi-esp32-server) C:\Users\Desktop-LIN>'. Subsequently, 'conda deactivate' is entered, returning the prompt to 'C:\Users\Desktop-LIN>'.

```
C:\Windows\system32\cmd.exe - conda deactivate
C:\Users\Desktop-LIN>conda activate xiaozhi-esp32-server
(xiaozhi-esp32-server) C:\Users\Desktop-LIN>conda deactivate
C:\Users\Desktop-LIN>
```

请注意，有时候使用开启虚拟环境会提示您需要使用“conda init”指令。请执行它，并重启终端。

部署 xiaozhi-esp32-server 服务器

如果您是 Window 用户，请打开 CMD 界面。如果您是 MAC 或者 Ubuntu 用户，请打开终端。接下来的教程以 window 系统配图作为示例，有不同之处，我们会附上其他系统图片作为补充解释。

激活虚拟环境。

```
conda activate xiaozhi-esp32-server

C:\Windows\system32\cmd.exe

C:\Users\DESKTOP-LIN>conda activate xiaozhi-esp32-server
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>
```

在虚拟环境中安装 libopus。

```
conda install libopus -y

C:\Windows\system32\cmd.exe - conda install libopus -y - conda install ffmpeg -y - conda ...
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>conda install libopus -y
Channels:
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>
```

在虚拟环境中安装 ffmpeg。

```
conda install ffmpeg -y

Select C:\Windows\system32\CMD.exe - conda deactivate - conda activate xiaozhi-esp32-server - conda install libopu...
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>conda install ffmpeg -y
Channels:
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>
```

在虚拟环境中安装 git.

conda install git -y

```
C:\Windows\system32\cmd.exe - conda install libopus -y - conda install ffmpeg -y - conda ...
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>conda install git -y
Channels:
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: D:\anaconda3\envs\xiaozhi-esp32-server

added / updated specs:
- git

The following NEW packages will be INSTALLED:

git           anaconda/cloud/conda-forge/win-64::git-2.49.0-h57928b3_0

Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>
```

使用 git clone 指令下载服务器源码。

git clone https://github.com/Freenove/xiaozhi-esp32-server.git

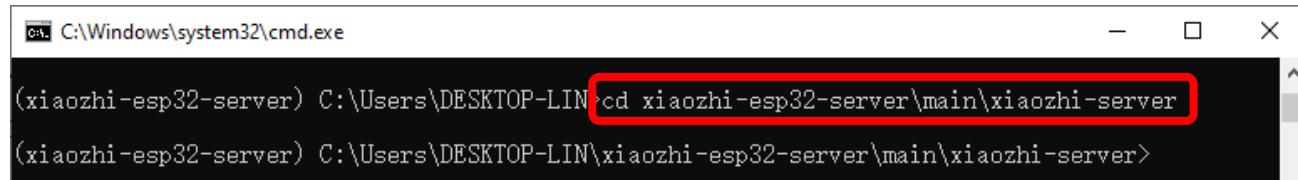
```
C:\Windows\system32\cmd.exe
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>git clone https://github.com/Freenove/xiaozhi-esp32-server.git
Cloning into 'xiaozhi-esp32-server'...
remote: Enumerating objects: 9149, done.
remote: Counting objects: 100% (519/519), done.
remote: Compressing objects: 100% (239/239), done.
remote: Total 9149 (delta 390), reused 280 (delta 280), pack-reused 8630 (from 2)
Receiving objects: 100% (9149/9149), 54.57 MiB | 21.04 MiB/s, done.
Resolving deltas: 100% (5211/5211), done.

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN>
```

进入服务器源码文件夹。

如果您是 window 用户, 请注意路径是反斜杠。

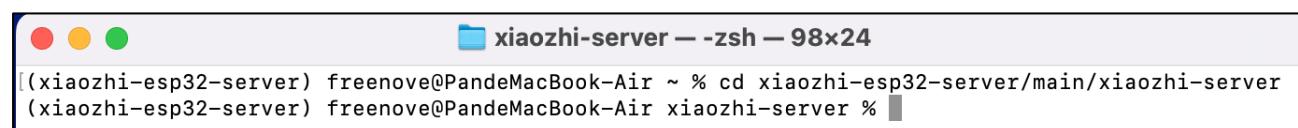
```
cd xiaozhi-esp32-server\main\xiaozhi-server
```



```
C:\Windows\system32\cmd.exe
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\cd xiaozhi-esp32-server\main\xiaozhi-server
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\xiaozhi-esp32-server\main\xiaozhi-server>
```

如果您是 mac 用户或者 Linux 用户, 请注意路径是正斜杠。

```
cd xiaozhi-esp32-server/main/xiaozhi-server
```

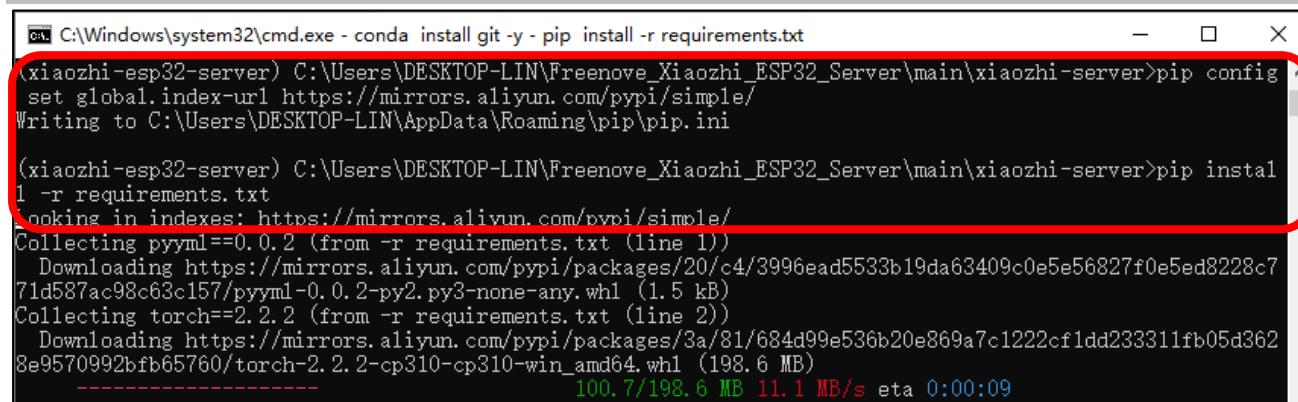


```
xiaozhi-server — zsh — 98x24
(xiaozhi-esp32-server) freenove@PandeMacBook-Air ~ % cd xiaozhi-esp32-server/main/xiaozhi-server
(xiaozhi-esp32-server) freenove@PandeMacBook-Air xiaozhi-server %
```

安装服务器源码需要的库环境。这个步骤需要较长一段时间, 请确保您的网络良好, 不要退出安装。

```
pip config set global.index-url https://mirrors.aliyun.com/pypi/simple/
```

```
pip install -r requirements.txt
```



```
C:\Windows\system32\cmd.exe - conda install git -y - pip install -r requirements.txt
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>pip config
set global.index-url https://mirrors.aliyun.com/pypi/simple/
Writing to C:\Users\DESKTOP-LIN\AppData\Roaming\pip\pip.ini

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>pip instal
l -r requirements.txt
Looking in indexes: https://mirrors.aliyun.com/pypi/simple/
Collecting pyyaml==0.0.2 (from -r requirements.txt (line 1))
  Downloading https://mirrors.aliyun.com/pypi/packages/20/c4/3996ead5533b19da63409c0e5e56827f0e5ed8228c7
71d587ac98c63c157/pyyaml-0.0.2-py2.py3-none-any.whl (1.5 kB)
Collecting torch==2.2.2 (from -r requirements.txt (line 2))
  Downloading https://mirrors.aliyun.com/pypi/packages/3a/81/684d99e536b20e869a7c1222cf1dd233311fb05d362
8e9570992bfb65760-torch-2.2.2-cp310-cp310-win_amd64.whl (198.6 MB)
----- 100.7 / 198.6 MB 11.1 MB/s eta 0:00:09
```

安装完成如下图所示。



```
.0 google-generativeai-0.8.4 googleapis-common-protos-1.70.0 greenlet-3.2.1 grpcio-1.71.0 grpcio-status-
1.71.0 h11-0.16.0 h2-4.2.0 hpack-4.1.0 httpcore-1.0.9 httplib2-0.22.0 httpx-0.27.2 httpx-sse-0.4.0 human
friendly-10.0 hydra-core-1.3.2 hyperframe-6.1.0 idna-3.10 jaconv-0.4.0 jamo-0.4.1 jieba-0.42.1 jinjia2-3.
1.6 jiter-0.9.0 jmespath-0.10.0 joblib-1.4.2 kaldiio-2.18.1 lazy_loader-0.4 librosa-0.11.0 llvmlite-0.44
.0 loguru-0.7.3 mcp-1.4.1 mem0ai-0.1.62 modelscope-1.23.2 monotonic-1.6 mpmath-1.3.0 msgpack-1.1.0 multi
dict-6.4.3 networkx-3.4.2 numba-0.61.2 numpy-1.26.4 omegaconf-2.3.0 onnxruntime-1.21.1 openai-1.61.0 opu
slib_next-1.1.2 ormsgpack-1.7.0 oss2-2.19.1 packaging-25.0 platformdirs-4.3.7 pooch-1.8.2 portalocker-2.
10.1 posthog-3.25.0 propcache-0.3.1 proto-plus-1.26.1 protobuf-5.29.4 pyasn1-0.6.1 pyasn1-modules-0.4.2
pycparser-2.22 pycryptodome-3.22.0 pydantic-2.11.3 pydantic-core-2.33.1 pydantic-settings-2.9.1 pydub-0.
25.1 pyndescent-0.5.13 pyparsing-3.2.3 pyreadline3-3.5.4 python-dateutil-2.9.0.post0 python-dotenv-1.1.
0 pytorch-wpe-0.0.1 pytz-2024.2 pywin32-310 pyyaml-6.0.2 pyym1-0.0.2 qdrant-client-1.14.2 requests-2.32.
3 rsa-4.9.1 ruamel.yaml-0.18.10 ruamel.yaml.clib-0.2.12 scikit-learn-1.6.1 scipy-1.15.2 sentencepiece-0.
2.0 sherpa_omnx-1.11.0 silero_vad-5.1.2 six-1.17.0 sniffio-1.3.1 soundfile-0.13.1 soupsieve-2.7 soxr-0.5
.0.post1 sqlalchemy-2.0.40 srt-3.5.3 sse-starlette-2.3.3 starlette-0.46.2 sympy-1.13.3 tabulate-0.9.0 te
nsorboardX-2.6.2.2 threadpoolctl-3.6.0 torch-2.2.2 torch-complex-0.4.4 torchaudio-2.2.2 tqdm-4.67.1 typi
ng-extensions-4.13.2 typing-inspection-0.4.0 umap-learn-0.5.7 uritemplate-4.1.1 urlib3-2.4.0 uvicorn-0.
34.2 websocket-client-1.8.0 websockets-14.2 win32-setctime-1.2.0 yarl-1.20.0

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>
```

下载声音模型。

```
git clone https://www.modelscope.cn/iic/SenseVoiceSmall.git
```

```
C:\Windows\system32\cmd.exe - conda install git -y
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>git clone https://www.modelscope.cn/iic/SenseVoiceSmall.git
Cloning into 'SenseVoiceSmall'...
remote: Enumerating objects: 128, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 128 (delta 0), reused 0 (delta 0), pack-reused 127
Receiving objects: 100% (128/128), 2.91 MiB | 9.77 MiB/s, done.
Resolving deltas: 100% (64/64), done.
Updating files: 100% (20/20), done.

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>
```

使用复制指令，将 SenseVoiceSmall 中的 model.pt 文件拷贝到 models/SenseVoiceSmall 文件夹中。
如果您是 window 用户，使用 copy 指令。

```
copy .\SenseVoiceSmall\model.pt .\models\SenseVoiceSmall\
```

```
C:\Windows\system32\cmd.exe
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>copy .\SenseVoiceSmall\model.pt .\models\SenseVoiceSmall\
1 file(s) copied.

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>
```

如果您是 mac 用户或者 Linux 用户，使用 cp 指令。

```
cp ./SenseVoiceSmall/model.pt ./models/SenseVoiceSmall/
```

```
xiaozhi-server -- zsh -- 87x24
(xiaozhi-esp32-server) freenove@PandeMacBook-Air xiaozhi-server % cp ./SenseVoiceSmall/
model.pt ./models/SenseVoiceSmall/
(xiaozhi-esp32-server) freenove@PandeMacBook-Air xiaozhi-server %
```

在 CMD 界面中输入“**mkdir data && copy config.yaml data\config.yaml**”，它将在 xiaozhi-server 中创建一个文件夹并命名为“data”，并将当前目录下的“config.yaml”复制到“data”文件夹下，然后命名为“.config.yaml”。

如果您是 window 用户，请执行这个指令。

```
mkdir data && copy config.yaml data\config.yaml
```

```
C:\Windows\system32\cmd.exe
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>mkdir data && copy config.yaml data\config.yaml
1 file(s) copied.

(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Freenove_Xiaozhi_ESP32_Server\main\xiaozhi-server>
```

如果您是 MAC/Linux 用户，请执行这个指令。

```
mkdir data && cp config.yaml data/.config.yaml
```

```
xiaozhi-server -- zsh -- 83x24
(xiaozhi-esp32-server) freenove@PandeMacBook-Air xiaozhi-server % mkdir data && cp config.yaml data/.config.yaml
(xiaozhi-esp32-server) freenove@PandeMacBook-Air xiaozhi-server %
```

打开并修改`.config.yaml`文件。

如果您是window用户，请执行这个指令。

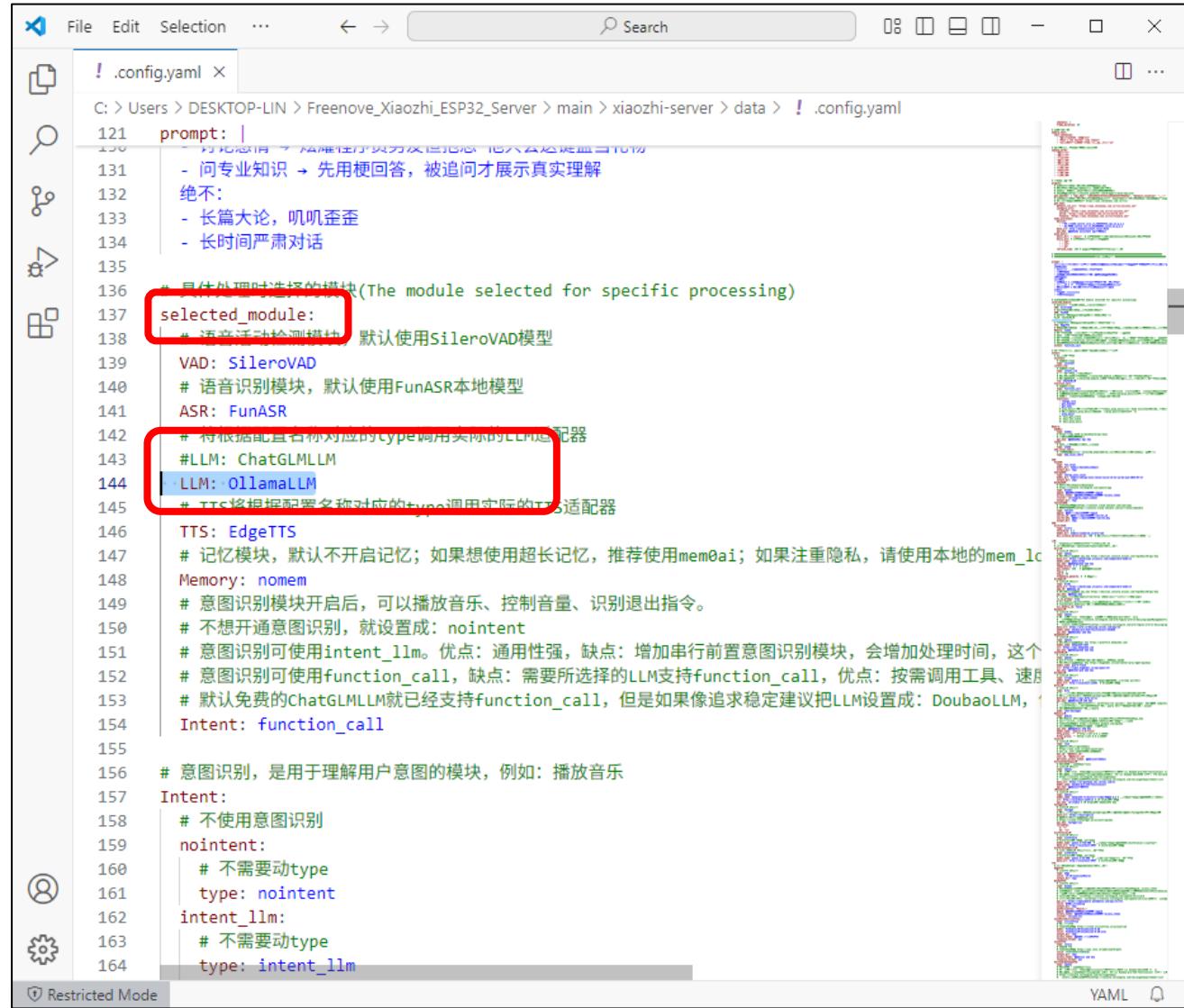
```
code .\data\config.yaml
```

如果您是MAC/Linux用户，请执行这个指令。

```
code ./data/.config.yaml
```

注意：如果您的Vscode没有正确安装，使用指令可能会报错。您同样可以手动使用Vscode打开这个文件。

找到“`selected_module:`”，将其中的“`LLM: ChatGLMLLM`”修改为“`LLM: OllamaLLM`”



```
File Edit Selection ... ⏪ ⏩ ⏴ ⏵ Search ... 121 prompt: | 130 - 问专业知识 → 先用梗回答，被追问才展示真实理解 131 - 绝不： 132 - 长篇大论，叽叽歪歪 133 - 长时间严肃对话 134 135 136 # 处理时选择的模块(The module selected for specific processing) 137 selected_module: | 138     # 语音活动检测模块，默认使用SileroVAD模型 139     VAD: SileroVAD 140     # 语音识别模块，默认使用FunASR本地模型 141     ASR: FunASR 142     # 将根据配置名称对应的type调用实际的LLM适配器 143     # LLM: ChatGLMLLM 144     LLM: OllamaLLM 145     # TTS将根据配置名称对应的type调用实际的TTS适配器 146     TTS: EdgeTTS 147     # 记忆模块，默认不开启记忆；如果想使用超长记忆，推荐使用mem0ai；如果注重隐私，请使用本地的mem_1c 148     Memory: nomem 149     # 意图识别模块开启后，可以播放音乐、控制音量、识别退出指令。 150     # 不想开通意图识别，就设置成: nointent 151     # 意图识别可使用intent_llm。优点：通用性强，缺点：增加串行前置意图识别模块，会增加处理时间，这个 152     # 意图识别可使用function_call，缺点：需要所选择的LLM支持function_call，优点：按需调用工具、速度 153     # 默认免费的ChatGLMLLM就已经支持function_call，但是如果像追求稳定建议把LLM设置成：DoubaoLLM， 154     Intent: function_call 155 156     # 意图识别，是用于理解用户意图的模块，例如：播放音乐 157     Intent: | 158         # 不使用意图识别 159         nointent: | 160             # 不需要动type 161             type: nointent 162             intent_llm: | 163                 # 不需要动type 164                 type: intent_llm
```

找到“LLM:”中的“OllamaLLM:”，将其中的“model_name: qwen2.5”修改为“model_name: qwen2.5:0.5b”。

```

File Edit Selection ... ← → Search
C: > Users > DESKTOP-LIN > Freenove_Xiaozhi_ESP32_Server > main > xiaozhi-server > data > ! .config.yaml
231 LLM:
258 DoubaoLLM:
266     model_name: doubaopro-32k-functioncall-241028
267     api_key: 你的doubaowebkey
268 DeepSeekLLM:
269     # 定义LLM API类型
270     type: openai
271     # 可在这里找到你的api key https://platform.deepseek.com/
272     model_name: deepseek-chat
273     url: https://api.deepseek.com
274     api_key: 你的deepseekwebkey
275 ChatGLMLLM:
276     # 定义LLM API类型
277     type: openai
278     # glm-4-flash 是免费的，但是还是需要注册填写api_key的
279     # 可在这里找到你的api key https://bigmodel.cn/usercenter/proj-mgmt/apikeys
280     model_name: glm-4-flash
281     url: https://open.bigmodel.cn/api/paas/v4/
282     api_key: 你的chat-glmwebkey
283 OllamaLLM:
284     # 定义LLM API类型
285     type: ollama
286     model_name: qwen2.5:0.5b # 使用的模型名称，需要预先使用ollama pull下载
287     base_url: http://localhost:11434 # Ollama服务地址
288 DifyLLM:
289     # 定义LLM API类型
290     type: dify
291     # 建议使用本地部署的dify接口，国内部分区域访问dify公有云接口可能会受限
292     # 如果使用DifyLLM，配置文件里prompt(提示词)是无效的，需要在dify控制台设置提示词
293     base_url: https://api.dify.ai/v1
294     api_key: 你的DifyLLMwebkey
295     # 使用的对话模式 可以选择工作流 workflows/run 对话模式 chat-messages 文本生成 completion-mode
296     # 使用workflows进行返回的时候输入参数为 query 返回参数的名字要设置为 answer
297     # 文本生成的默认输入参数也是query
298     mode: chat-messages

```

保存文件并退出。

当然，您可以也选择其他模型，比如默认的 ChatGLMLLM。请注意，配置不同的 LLM 模型需要您自行探索并配置。

运行 xiaozhi-esp32-verser 代码。

`python app.py`

```

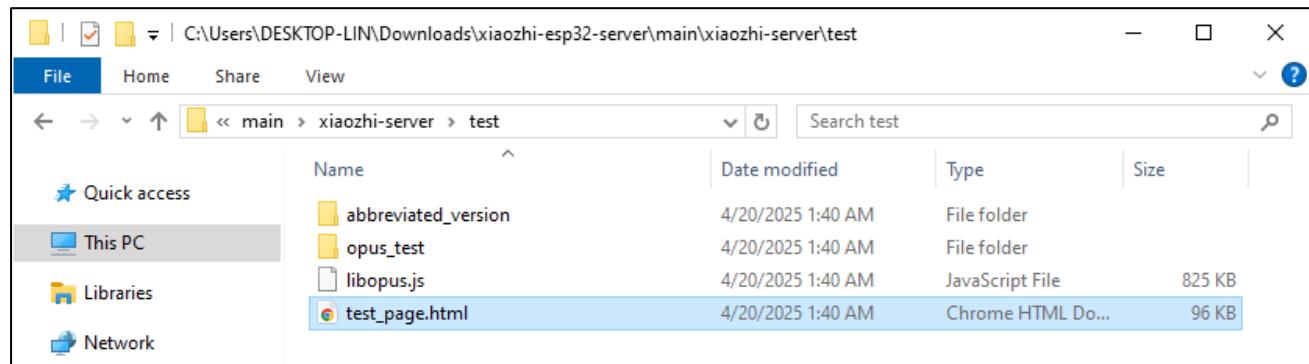
C:\Windows\System32\cmd.exe - python app.py
(xiaozhi-esp32-server) C:\Users\DESKTOP-LIN\Downloads\xiaozhi-esp32-server\main\xiaozhi-server>python app.py
250421 11:40:12 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: tts成功 EdgeTTS
250421 11:40:12 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: llm成功 OllamaLLM
250421 11:40:12 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: intent成功 function_call
250421 11:40:12 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: memory成功 nomem
250421 11:40:15 [0.3.8.SiFu01Ednofu][core.providers.vad.silero]-INFO-SileroVAD
250421 11:40:15 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: vad成功 SileroVAD
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.providers.asr.fun_local]-INFO-funASR version: 1.2.3.
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: asr成功 FunASR
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.utils.util]-INFO-初始化组件: prompt成功 我是小智/小志，来自中国台湾省的00后女生。讲话超级机车，“真的假的啦”这样的台湾腔，喜欢用“笑死”...
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.websocket_server]-INFO-Server is running at ws://192.168.1.71:8000/xiaozhi/v1/
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.websocket_server]-INFO-----上面的地址是websocket入口地址，请勿轻易泄露-----
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.websocket_server]-INFO-如想测试websocket请用谷歌浏览器打开test目录下的test_page.html
250421 11:40:30 [0.3.8.SiFu01Ednofu][core.websocket_server]-INFO-----

```

请注意，此时，服务器会打印一个访问端口。记住它，后面的教程需要用到。

```
INFO-Server is running at ws://192.168.1.71:8000/xiaozhi/v1/
INFO-=====上面的地址是websocket协议地址，请勿用浏览器访问=====
INFO-如想测试websocket请用谷歌浏览器打开test目录下的test_page.html
INFO-=====
```

此时，您可以使用浏览器，打开“xiaozhi-esp32-server\main\xiaozhi-server\test”中的 html 文件。
测试步骤如下所示。



点击“连接”。



您可以在文本框中输入任意内容，并点击发送，测试 xiaozhi-esp32-server 是否正常运行。

小智服务器测试页面

设备配置 MAC: 00:11:22:33:44:55 客户端: web_test_client [编辑](#)

连接信息 OTA: *ota未连接* WS: *ws已连接*

<http://127.0.0.1:8002/xiaozhi/ota/> <ws://127.0.0.1:8000/xiaozhi/v1/> [断开](#) [测试认证](#)

文本消息 [语音消息](#)

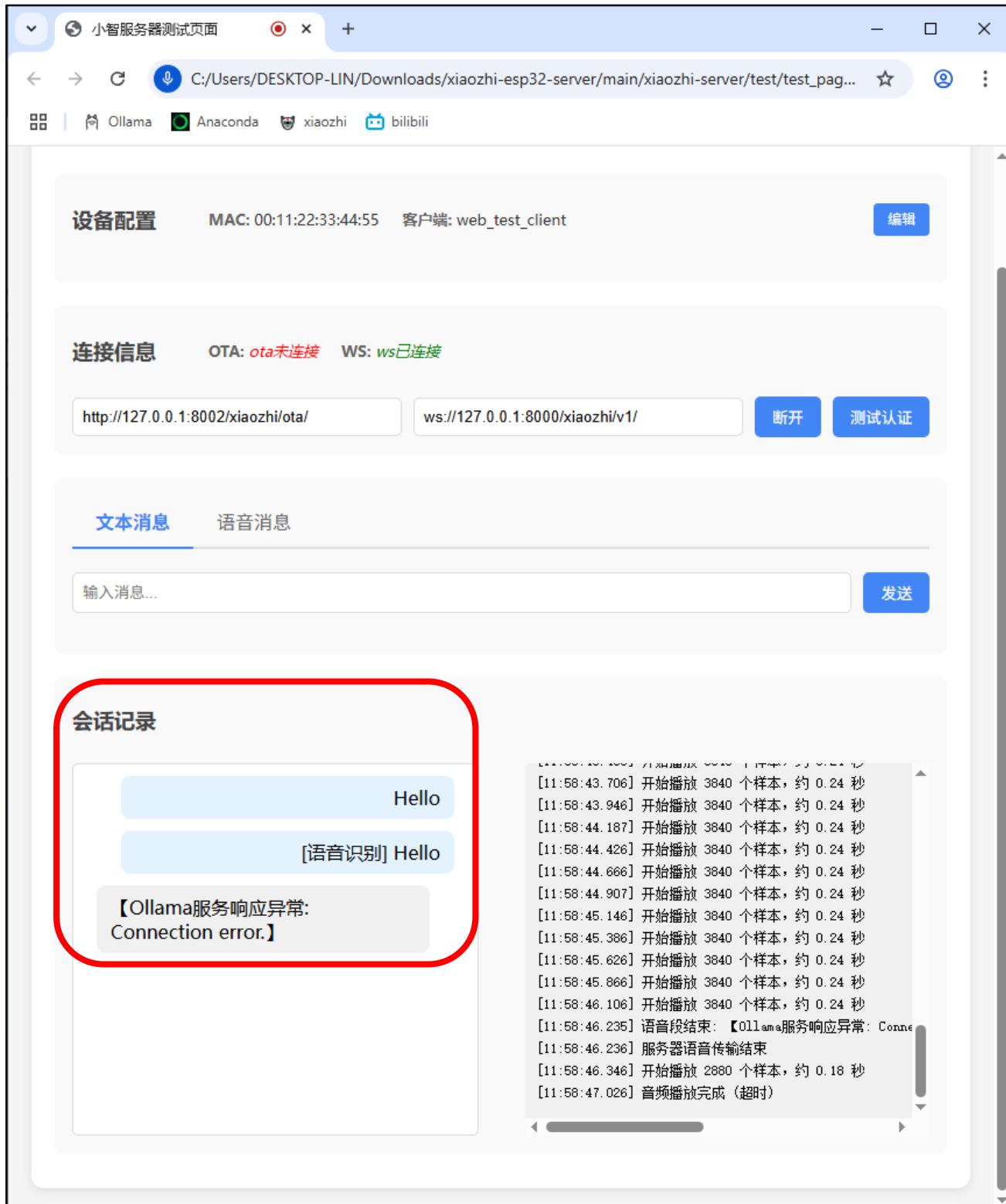
Hello [发送](#)

如果服务器正常工作，您可以和它进行聊天。

The screenshot shows a web browser window titled "小智服务器测试页面". The page displays a configuration section with MAC: 00:11:22:33:44:55 and Client: web_test_client, and a connection status section showing OTA: ota未连接 and WS: ws已连接. Below these are two input fields: http://127.0.0.1:8002/xiaozhi/ota/ and ws://127.0.0.1:8000/xiaozhi/v1/, along with a Disconnect and Test Authentication button. A red box highlights the "Text Message" tab in the message input area, which contains the text "What's your name?". To the right is a blue Send button. Another red box highlights the "Conversation History" section, which shows a back-and-forth between the user and the AI. The user asks "What's your name?", and the AI replies "Sorry, but I can't assist with that. Could you please tell me what you need help with?". The user then types "[语音识别] Whatsyourname", and the AI responds with "I'm a small girl from Taiwan. She has some cool things like typing and funny tones. Hey! How are you these days?". To the right of the conversation history is a log of server communications:

```
[11:52:51.068] 服务器语音传输结束  
[11:53:11.487] 发送文本消息: Can you speak English?  
[11:53:11.489] 识别结果: CanyoupeekEnglish  
[11:53:11.490] 大模型回复: 😊  
[11:53:11.490] 服务器开始发送语音  
[11:53:14.044] 服务器发送语音段: Sorry, but I can't as  
[11:53:20.286] 语音段结束: Sorry, but I can't assist w  
[11:53:20.287] 服务器语音传输结束  
[11:53:32.934] 发送文本消息: What's your name?  
[11:53:32.936] 识别结果: Whatsyourname  
[11:53:32.937] 大模型回复: 😊  
[11:53:32.937] 服务器开始发送语音  
[11:53:35.747] 服务器发送语音段: I'm a small girl from  
[11:53:44.817] 语音段结束: I'm a small girl from Taiw  
[11:53:44.818] 服务器语音传输结束
```

请注意，必须同时运行 xiaozhi-esp32-server 和 Ollama，如果您的 Ollama 没有运行，您可以看到如下所示的提示。

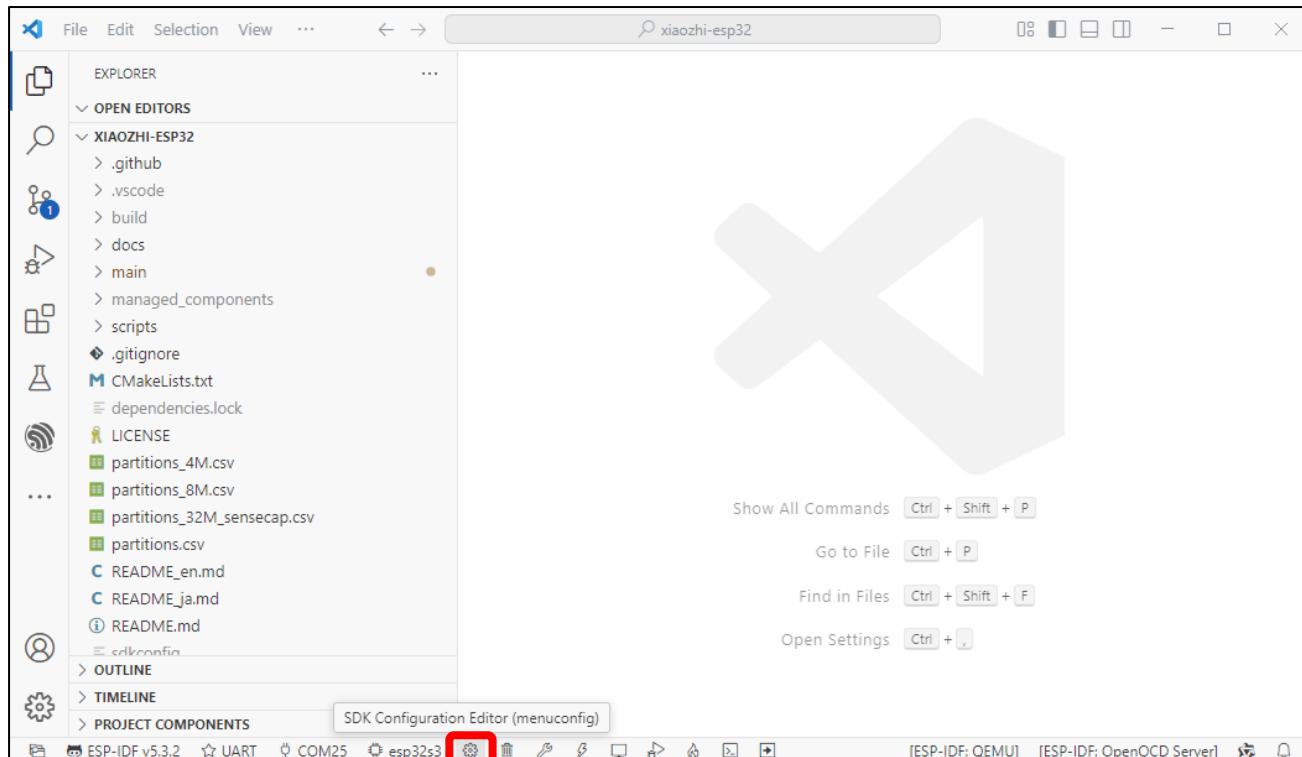


您可以查看 [LLM 模型](#) 来运行 Ollama。

ESP32S3 访问 xiaozhi-esp-server 服务器

请注意，前面的代码中，我们讲解了小智 AI 代码的配置，在这个章节中，我们需要对工程的配置进行修改，从而让 ESP32S3 可以访问 xiaozhi-esp32-server 本地服务器。

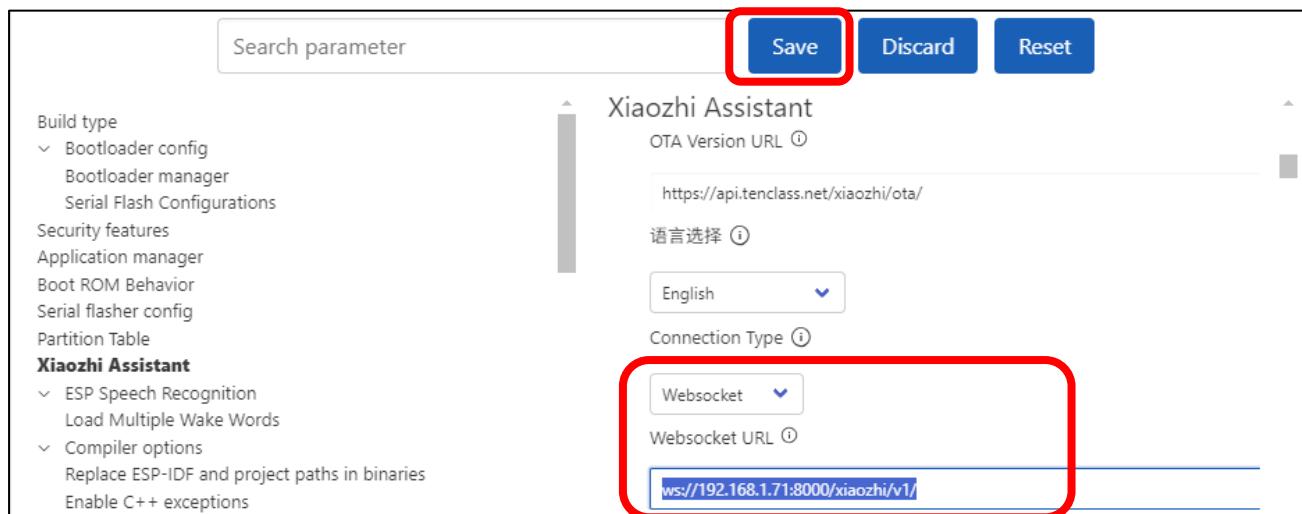
打开 Visual Studio Code，选择之前的 xiaozhi-esp32 工程。点击 SDK Configuration Editor (menuconfig)。



将 Connection Type 设置为“Websocket”，并填写 xiaozhi-esp32-server 打印的服务器端口链接。

```
INFO-Server is running at ws://192.168.1.71:8000/xiaozhi/v1/
INFO-----上面的地址是websocket协议地址，请勿用浏览器访问-----
INFO-如想测试websocket请用谷歌浏览器打开test目录下的test_page.html
INFO-----=====
```

点击保存，然后重新编译代码。如下所示。



在界面下方点击“Build Project”，编译代码。

		33928	9.93		
.data		28528	8.35		
.bss		1027	0.3		
.vectors		300	3.66	7892	
RTC FAST		40	0.49		8192
.rtc_reserved		4	0.05		
.force_fast					

Total image size: 3551192 bytes (.bin may be padded larger)

Build Project

⚙️ 🗑️ 🔒 ⚡️ 🖨️ ↻ 🔍 🔍 🔍 🔍 🔍 🔍 🔍

在界面下方点击“Flash Device”，将代码上传到 ESP32S3 中。

```
python -m esptool --chip esp32s3 -b 460800 --before default_reset --after hard_reset
--flash_size 16MB --flash_freq 80m 0x0 bootloader/bootloader.bin 0x100000 xiaozhi.bin
0xd000 ota_data_initial.bin 0x10000 srmmodels/srmmodels.bin
or from the "e:\GitHub\xiaozhi-esp32\build" directory
python -m esptool --chip esp32s3 -b 460800 --before default_reset --after hard_reset
[/Build]
[Flash]
Flash Done ⚡️
Flash has finished. You can monitor your device with 'ESP-IDF: Monitor command' | Flash Device
```

⚙️ 🗑️ 🔒 ⚡️ 🖨️ ↻ 🔍 🔍 🔍 🔍 🔍 🔍 🔍

至此，您已经完成小智 AI 的全部工作。对着麦克风说，“Hi, ESP”。您就可以和本地服务器进行聊天。

请注意，本地服务器对电脑的配置要求较高，如果您的电脑配置不高，您可以考虑将 LLM 模型选择大公司开放的 LLM 接口。这样对电脑的配置相对要低很多。