ROS2 安装与配置

Ubuntu虚拟机安装

虚拟机是一个软件，可以在已有系统之上，构建另外一个虚拟的系统，让多个操作环境同时运行。

这里我们采用的虚拟机软件叫做vmware，下载地址如下，安装步骤和其他软件相同，请大家自行下载并安装：

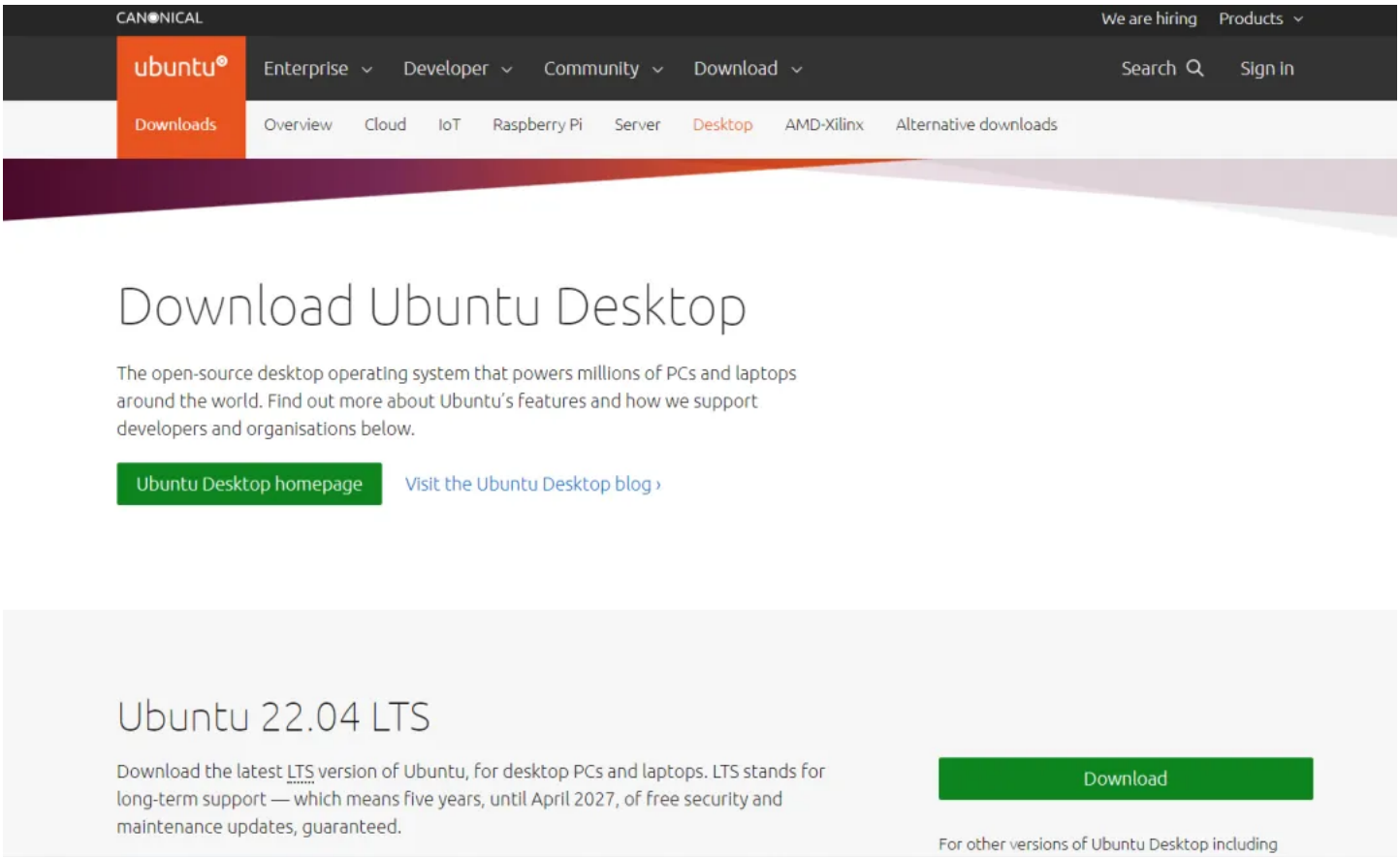
https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

准备工作完成后，就可以开始系统安装啦，安装步骤如下：

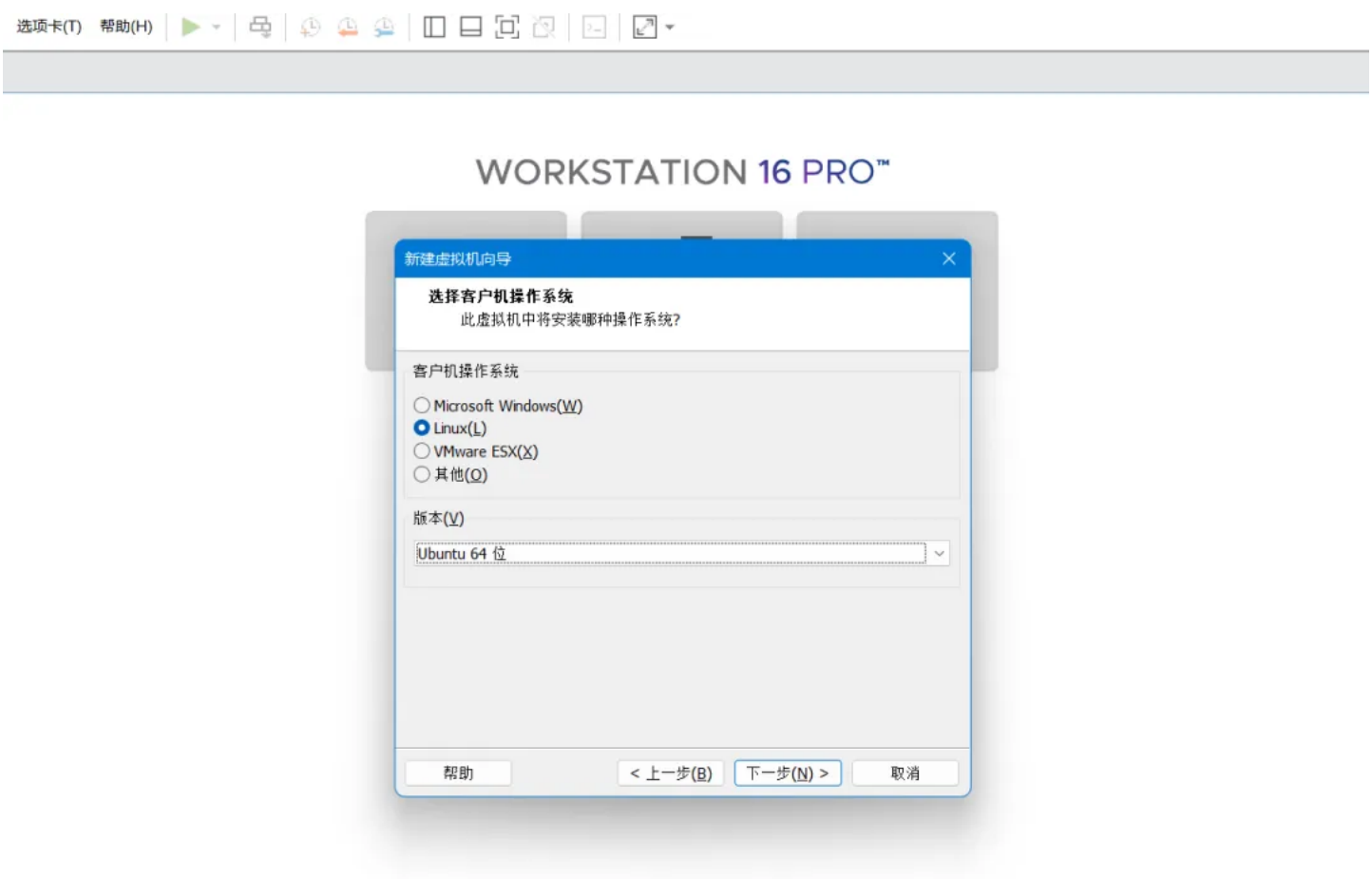
1.下载系统镜像

下载链接：

https://ubuntu.com/download/desktop

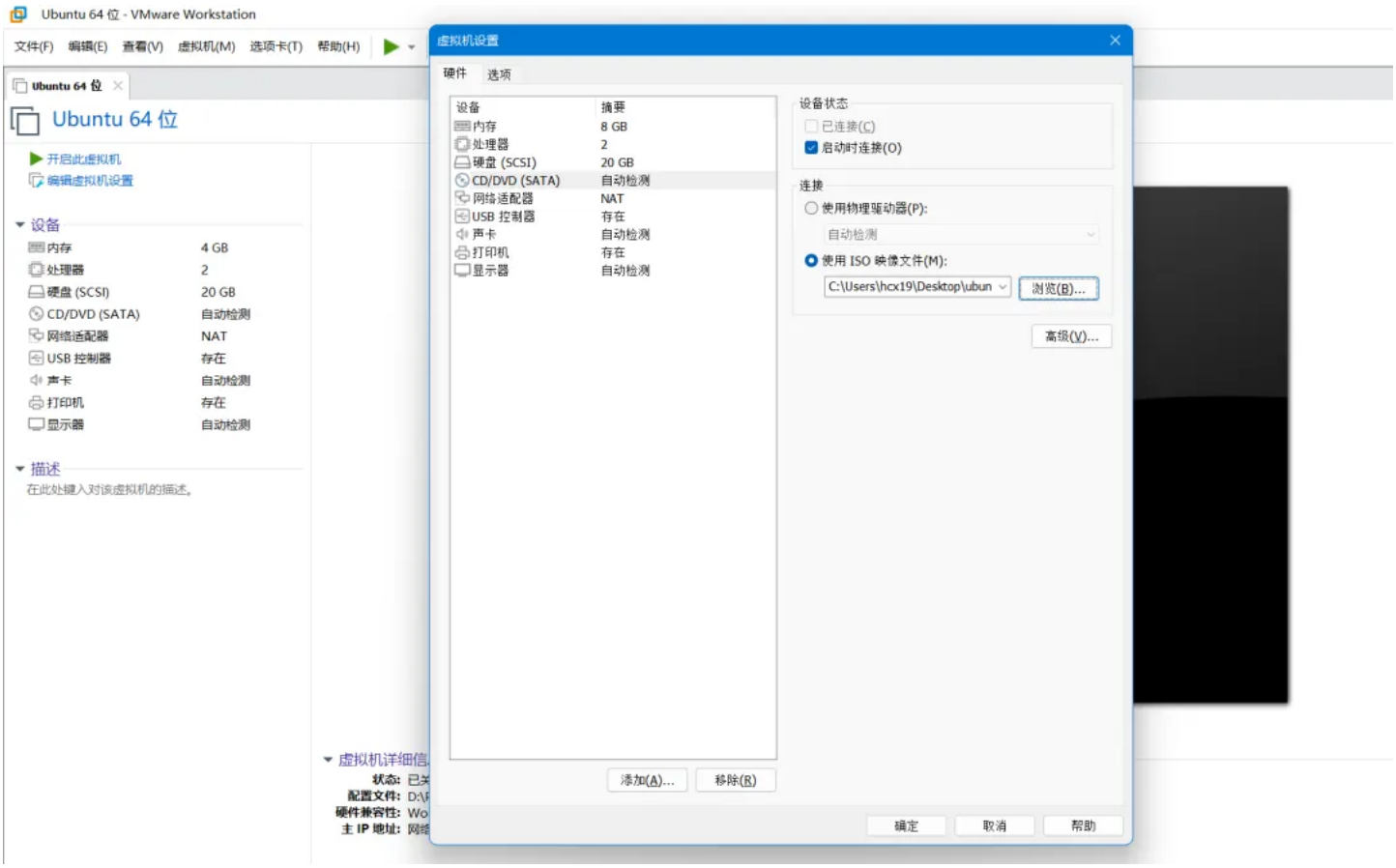


1. **在虚拟机中创建系统**

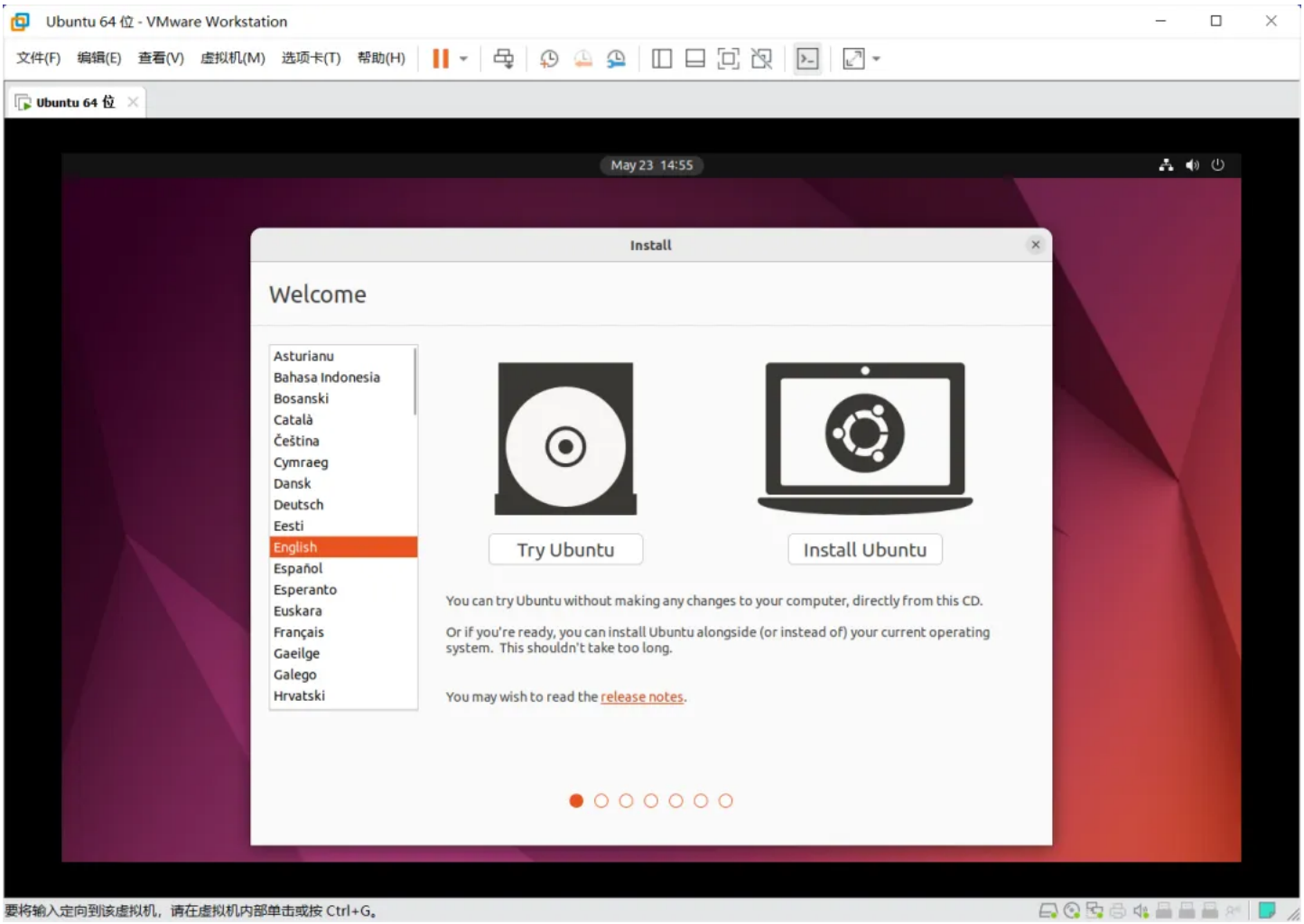


2. 设置虚拟机硬盘大小

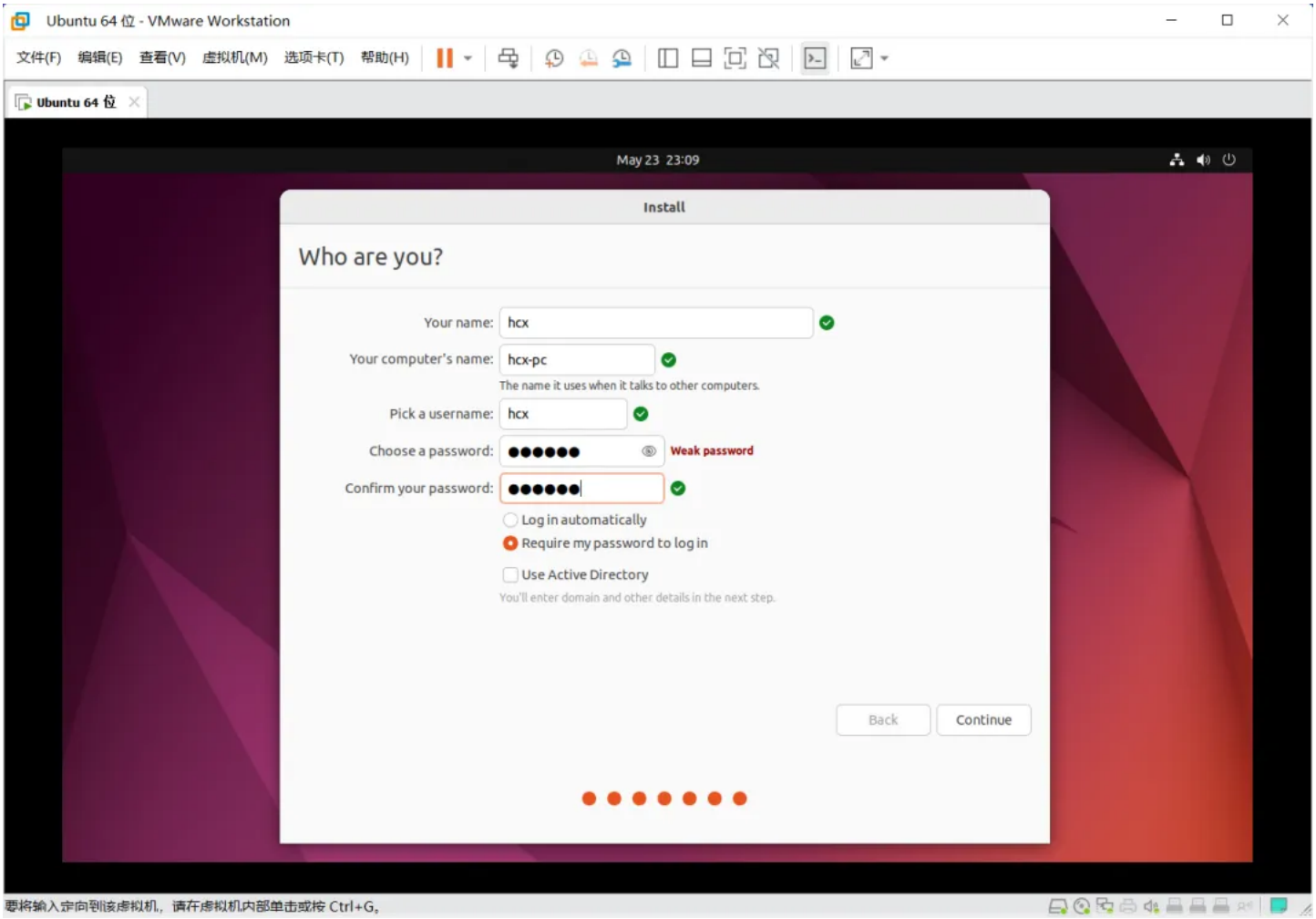
3. 设置Ubuntu镜像路径


Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
Ubuntu 64 位
开启此虚拟机
编辑虚拟机设置
设备
内存 4 GB
处理器 2
硬盘 (SCSI) 20 GB
CD/DVD (SATA) 自动检测
网络适配器 NAT
USB 控制器 存在
声卡 自动检测
打印机 存在
显示器 自动检测
描述
在此处键入对该虚拟机的描述。
虚拟机设置
硬件 选项
设备 摘要
内存 8 GB
处理器 2
硬盘 (SCSI) 20 GB
CD/DVD (SATA) 自动检测
网络适配器 NAT
USB 控制器 存在
声卡 自动检测
打印机 存在
显示器 自动检测
设备状态
已连接(C)
启动时连接(O)
连接
使用物理驱动器(P):
自动检测
使用 ISO 映像文件(M):
C:\Users\hcx19\Desktop\ubun
浏览(B)...
高级(V)...
添加(A)...
移除(R)
确定
取消
帮助

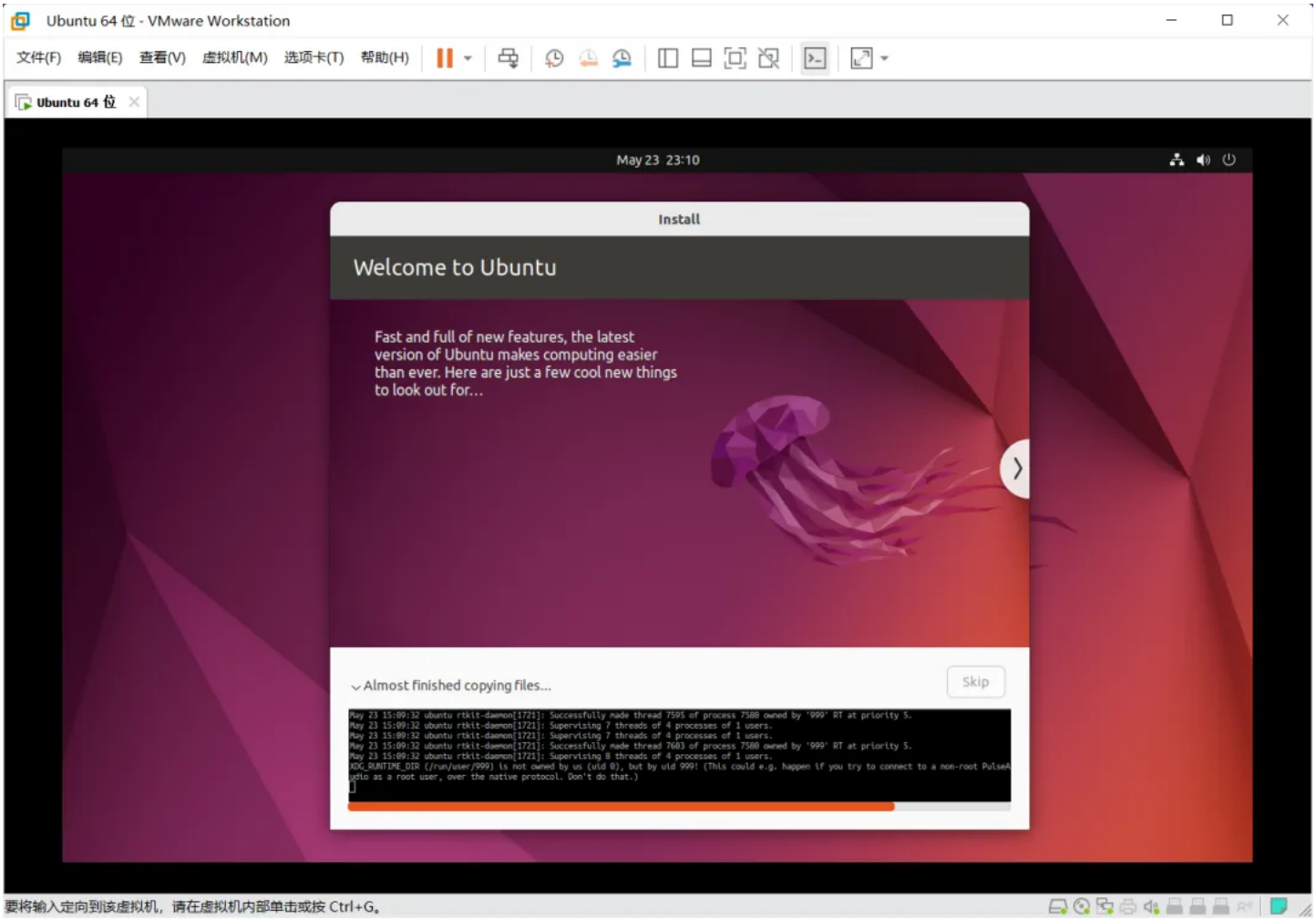
4. 启动虚拟机


Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
Ubuntu 64 位
May 23 14:55
Install
Welcome
Asturianu
Bahasa Indonesia
Bosanski
Català
Čeština
Cymraeg
Dansk
Deutsch
Eesti
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Try Ubuntu
Install Ubuntu
You can try Ubuntu without making any changes to your computer, directly from this CD.
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.
You may wish to read the release notes.
要将输入定向到该虚拟机，请在虚拟机内部单击或按 Ctrl+G。

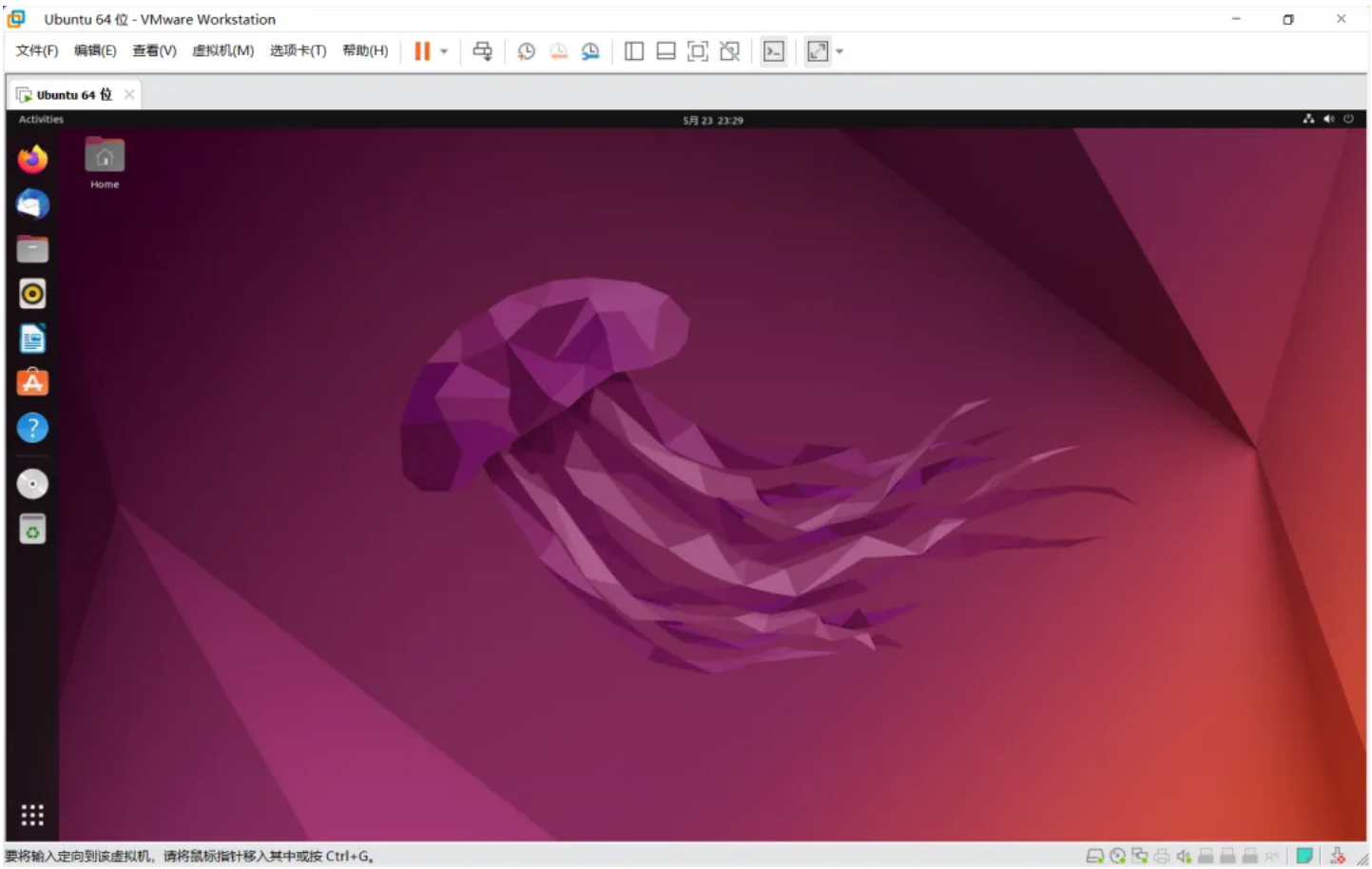
5. 设置用户名和密码



6. 等待系统安装


Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
Ubuntu 64 位
May 23 23:10
Install
Welcome to Ubuntu
Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for...
Almost finished copying files...
Skip
要将输入定向到该虚拟机，请在虚拟机内部单击或按 Ctrl+G。

7. 完成安装


Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
Ubuntu 64 位
Activities
5月 23 23:29
Home
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。

ROS2安装

1. 设置编码

```
sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
```

2. 添加源

```
sudo apt update && sudo apt install curl gnupg lsb-release
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(source /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

3. 安装ROS2

```
sudo apt update
sudo apt upgrade
sudo apt install ros-humble-desktop
```

4. 设置环境变量

```
source /opt/ros/humble/setup.bash
echo " source /opt/ros/humble/setup.bash" >> ~/.bashrc
```

5. 小海龟仿真示例

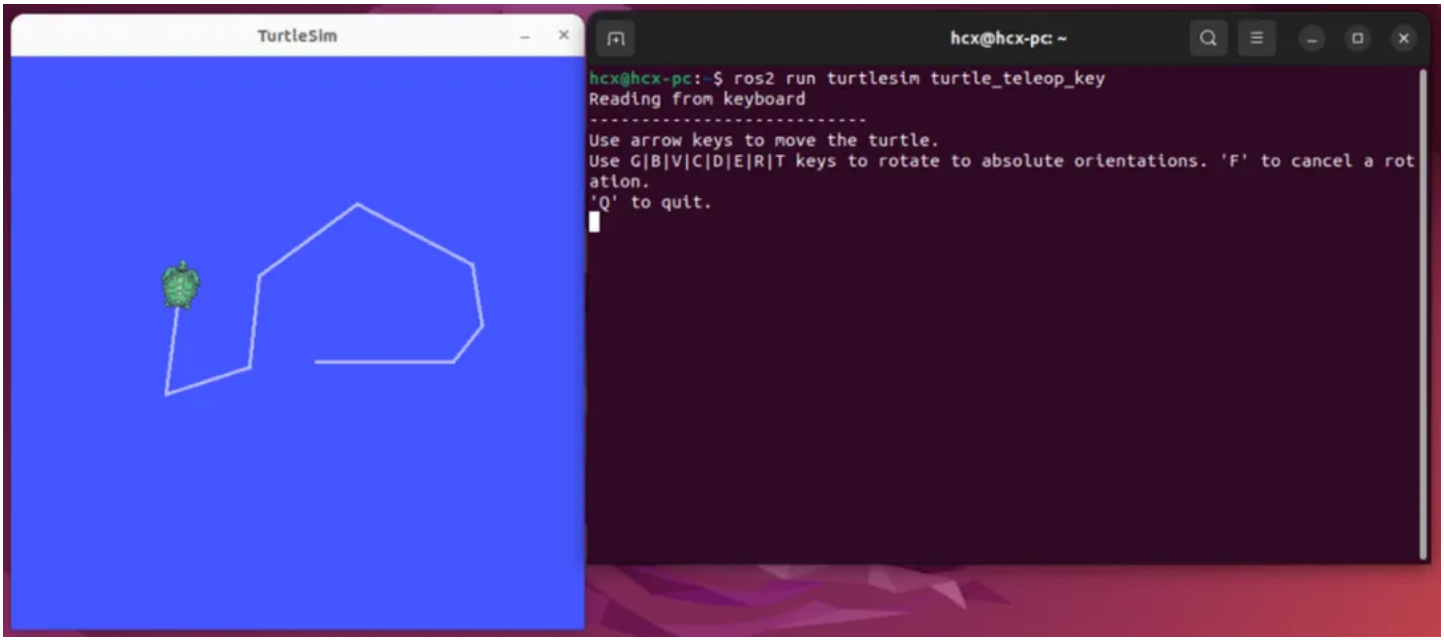
再来试一试ROS中的经典示例——小海龟仿真器。

启动两个终端，分别运行如下指令：

```
ros2 run turtlesim turtlesim_node
```

```
2 ros2 run turtlesim turtle_teleop_key
```

第一句指令将启动一个蓝色背景的海龟仿真器，第二句指令将启动一个键盘控制节点，在该终端中点击键盘上的“上下左右”按键，就可以控制小海龟运动啦。



ROS2安装完成。

gazebo安装：

安装

```
1 sudo apt install ros-humble-gazebo-*
```

环境变量添加

```
1 echo "source /usr/share/gazebo/setup.bash" >> ~/.bashrc
```

运行

```
1 ros2 launch gazebo_ros gazebo.launch.py
```

moveit安装

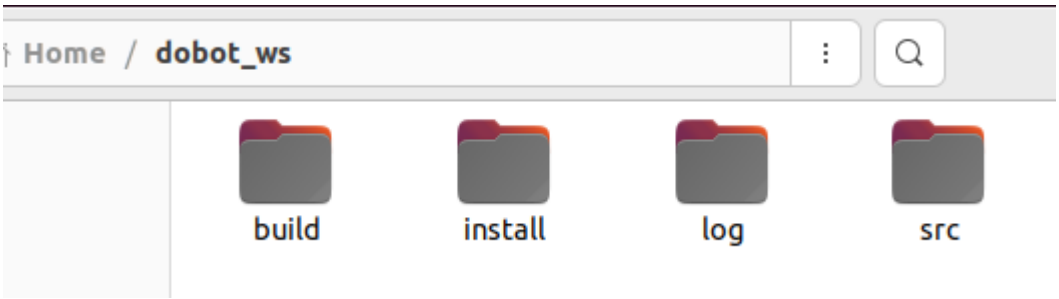
安装

```
sudo apt-get install ros-humble-moveit
```

介绍

工作空间结构

ROS系统中一个典型的工作空间结构如图所示，这个dev_ws就是工作空间的根目录，里边会有四个子目录，或者叫做四个子空间。



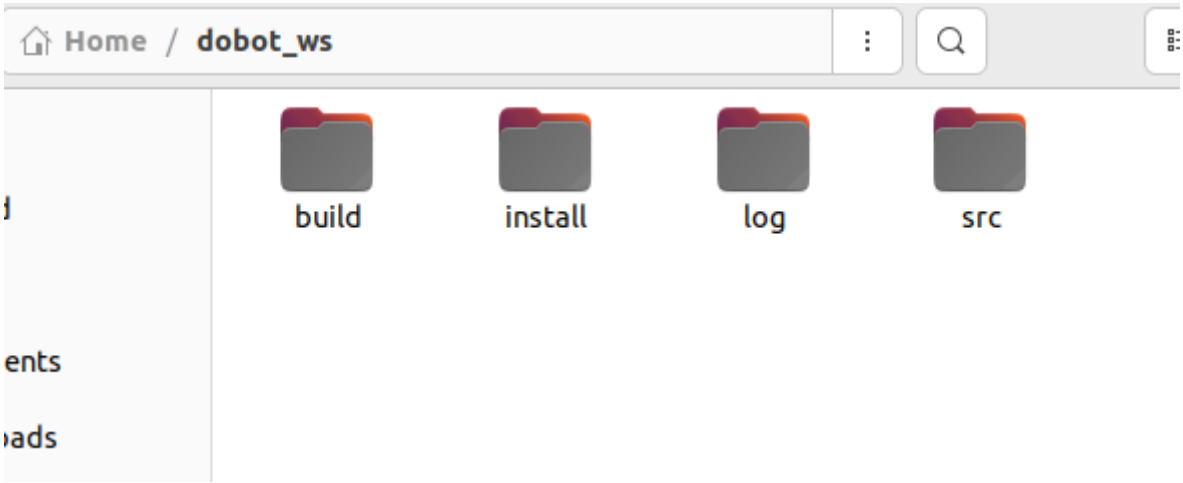
- **src，代码空间**，未来编写的代码、脚本，都需要人为的放置到这里；
- **build，编译空间**，保存编译过程中产生的中间文件；
- **install，安装空间**，放置编译得到的可执行文件和脚本；
- **log，日志空间**，编译和运行过程中，保存各种警告、错误、信息等日志。

编译工作空间

依赖安装完成后，就可以使用如下命令编译工作空间啦，如果有缺少的依赖，或者代码有错误，编译过程中会有报错，否则编译过程应该不会出现任何错误：

```
$ sudo apt install python3-colcon-ros
$ cd ~/dobot_ws/
$ colcon build
source install/local_setup.sh
```

编译成功后，就可以在工作空间中看到自动生产的build、log、install文件夹了。



设置环境变量

编译成功后，为了让系统能够找到我们的功能包和可执行文件，还需要设置环境变量：

```
$ source install/local_setup.sh # 仅在当前终端生效$
 echo " source ~/dobot_ws/install/local_setup.sh"~/.bashrc # 所有终端均生效
```

机械臂配置

下载源码

```
mkdir -p ~/dobot_ws/src
cd ~/dobot_ws/src
git clone https://github.com/Dobot-Arm/DOBOT_6Axis_ROS2.git
cd ~/dobot_ws
```

编译

```
colcon build
source install/local_setup.sh
```

设置环境变量

```
echo "source ~/dobot_ws/install/local_setup.sh" >> ~/.bashrc
```

设置机械臂连接IP

```
echo "export IP_address=192.168.5.1" >> ~/.bashrc
source ~/.bashrc
```

若为 CR3 机械臂，则使用如下命令设置机械臂类型

```
echo "export DOBOT_TYPE=cr3" >> ~/.bashrc
source ~/.bashrc
```

若为 CR5 机械臂，则使用如下命令设置机械臂类型

```
echo "export DOBOT_TYPE=cr5" >> ~/.bashrc
source ~/.bashrc
```

若为 CR10 机械臂，则使用如下命令设置机械臂类型

```
echo "export DOBOT_TYPE=cr10" >> ~/.bashrc
source ~/.bashrc
```

若为 CR16 机械臂，则使用如下命令设置机械臂类型

```
echo "export DOBOT_TYPE=cr16" >> ~/.bashrc
source ~/.bashrc
```

1. **使用演示**

在仿真环境下使用

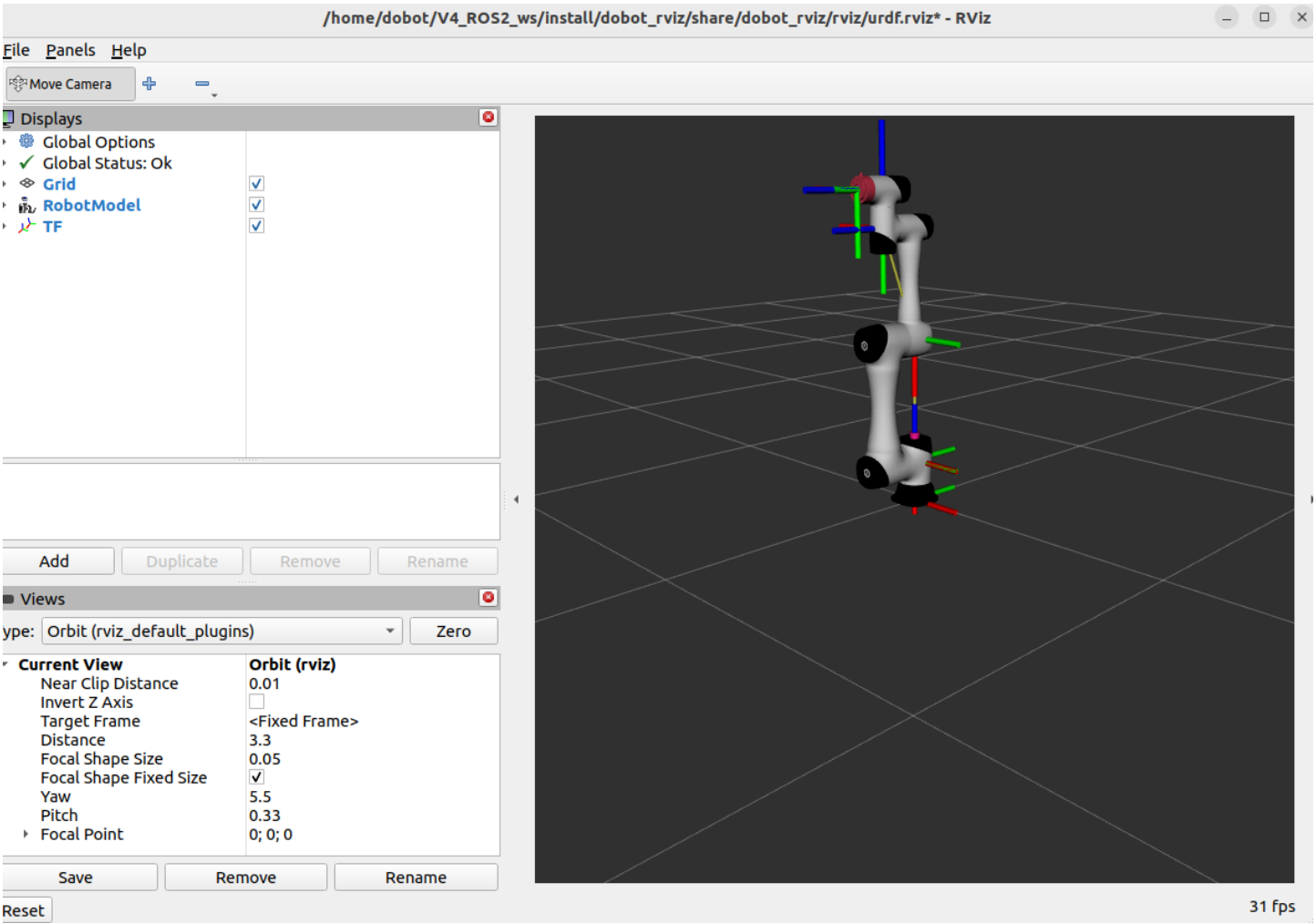
rviz 显示

安装：joint-state-publisher-gui

```
sudo apt-get install ros-humble-joint-state-publisher-gui
```

```
ros2 launch dobot_rviz display.launch
```

可通过加载 joint_state_publisher_gui 调节各关节的角度，在 rviz 上看到其显示效果

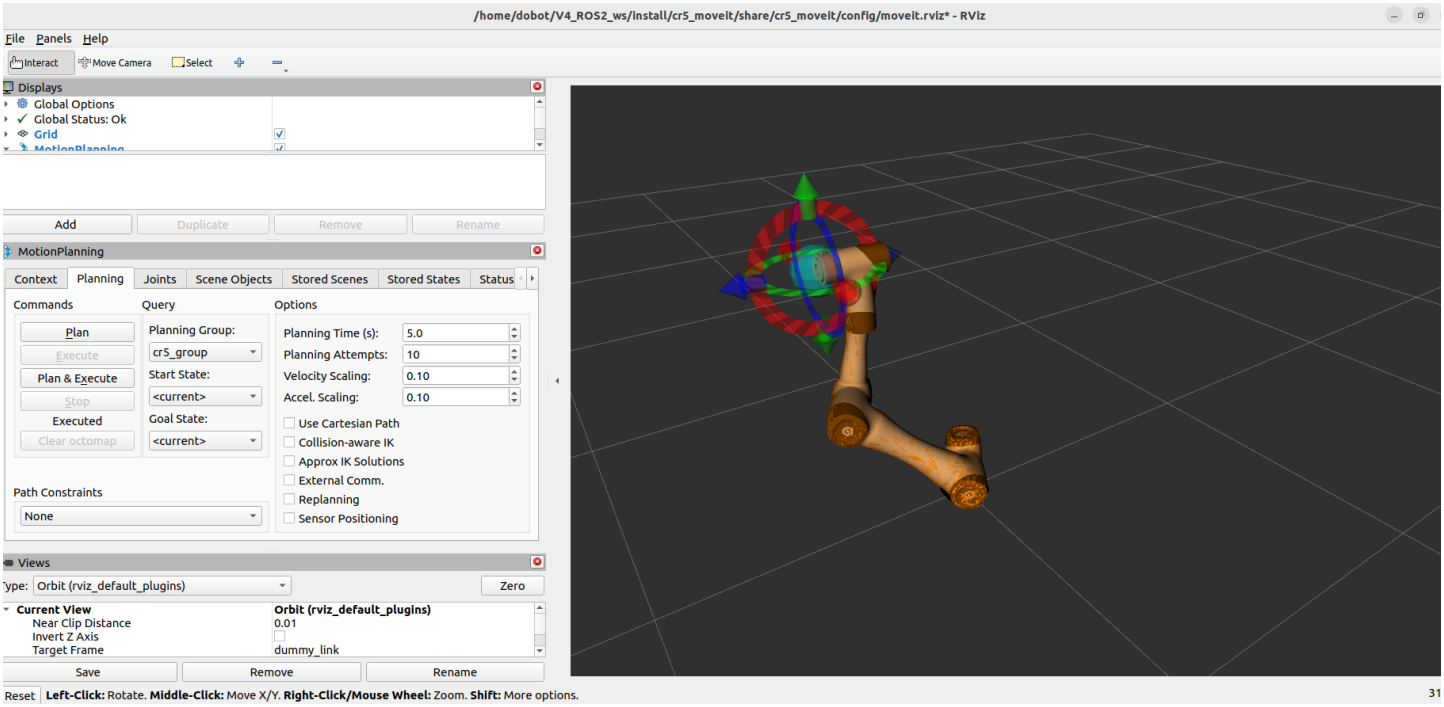


moveit 控制

- 使用如下命令启动 moveit

```
ros2 launch dobot_moveit dobot_moveit.launch.py
```

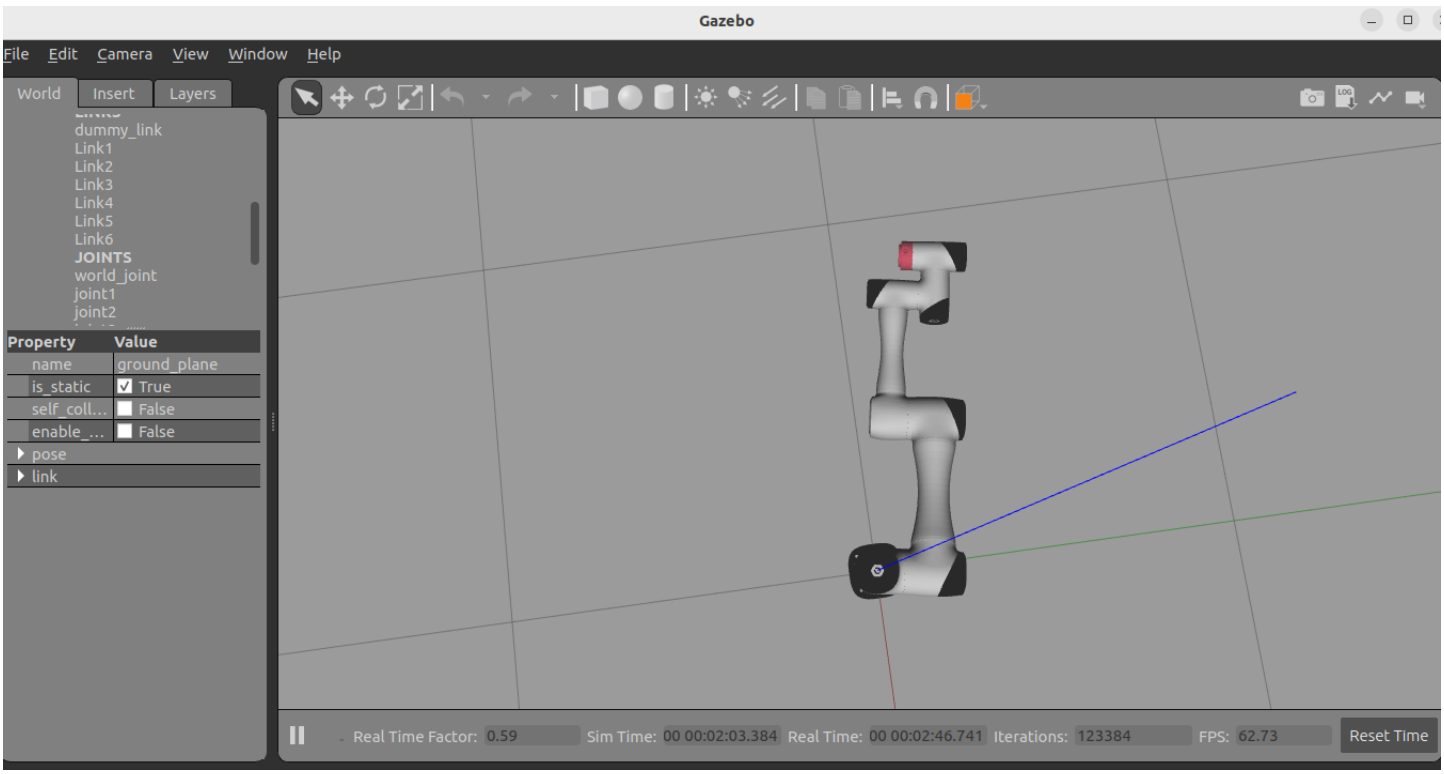
- 鼠标将关节拖到任意的角度，点击 "Plan and Execute" 即可看到运行效果



gazebo 仿真

- 使用如下命令启动 gazebo

```
1 ros2 launch dobot_gazebo dobot_gazebo.launch.py
```



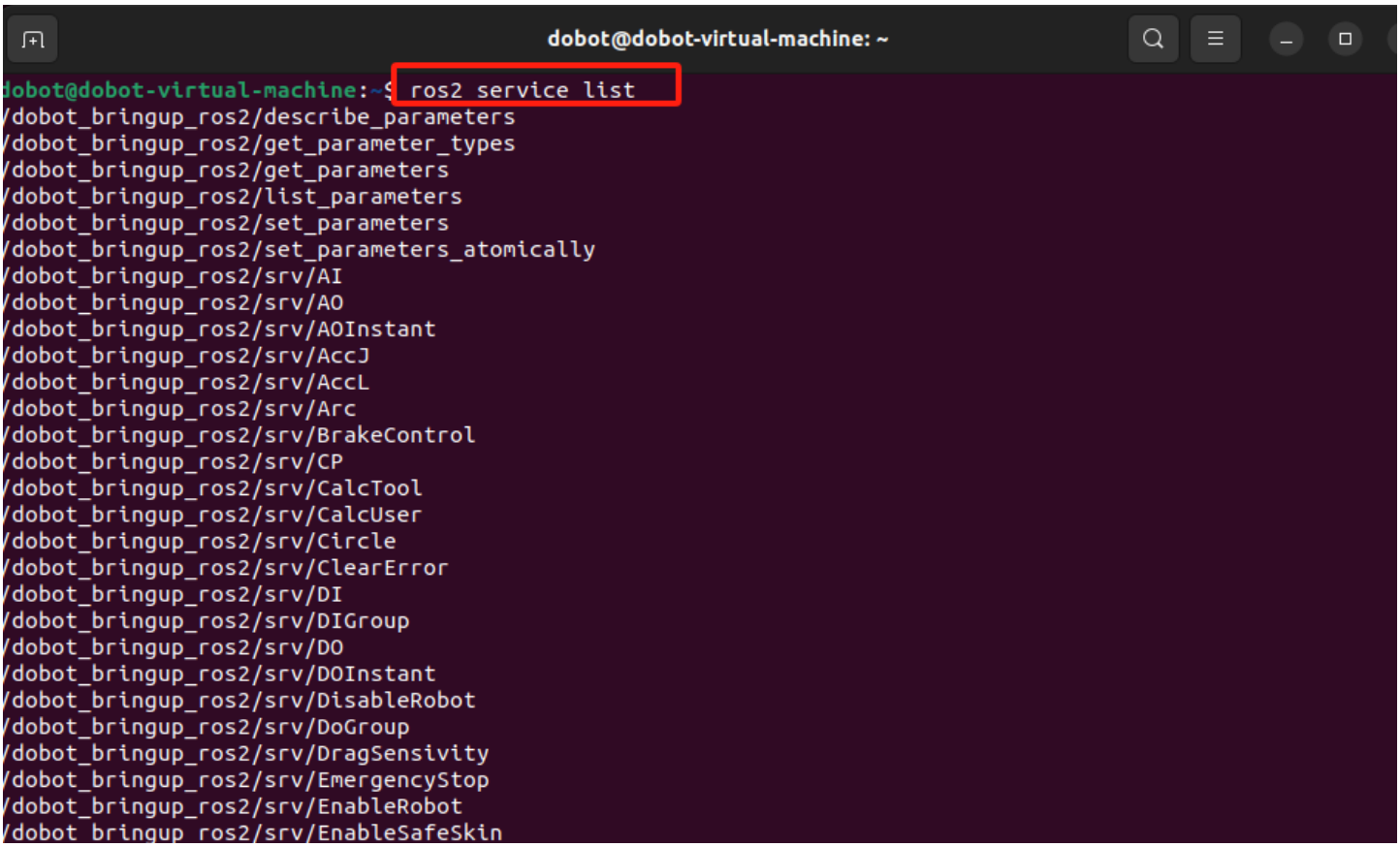
控制真实机械臂

- **使用如下命令连接机械臂, robot_ip 为实际机械臂所对应的IP地址**

```
ros2 launch cr_robot_ros2 dobot_bringup_ros2.launch.py
```

- **使用如下命令查看服务**

```
ros2 service list
```



Demo

完成一次取放动作

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
import rclpy
    # ROS2 Python接口库
from rclpy.node   import Node
    # ROS2 节点类
from dobot_msgs_v4.srv import *    # 自定义的服务接口


```

```python
class adderClient(Node):
    def __init__(self, name):
        super().__init__(name)
    # ROS2节点父类初始化
        self.EnableRobot_l = self.create_client(EnableRobot,'/dobot_bringup_ros2/srv/EnableRobot')
        self.MovJ_l = self.create_client(MovJ,'/dobot_bringup_ros2/srv/MovJ')
        self.SpeedFactor_l = self.create_client(SpeedFactor,'/dobot_bringup_ros2/srv/SpeedFactor')
        self.MovL_l = self.create_client(MovL,'/dobot_bringup_ros2/srv/MovL')
        self.DO_l = self.create_client(DO,'/dobot_bringup_ros2/srv/DO', )
        while not self.EnableRobot_l.wait_for_service(timeout_sec=1.0):
          # 循环等待服务器端成功启动
            self.get_logger().info('service not available, waiting again...')

    def initialization(self):  # 初始化：速度、坐标系、负载、工具偏心等
        response = self.EnableRobot_l.call_async()
        print(response)
        spe = SpeedFactor.Request()
        spe.ratio = 10
        response = self.SpeedFactor_l.call_async(spe)
        print(response)

    def point(self, Move, X_j1, Y_j2, Z_j3, RX_j4, RY_j5, RZ_j6):  # 运动指令
        if Move == "MovJ":
            P1 = MovJ.Request()
            P1.mode = 0
            P1.x = X_j1
            P1.y = Y_j2
            P1.z = Z_j3
            P1.a = RX_j4
            P1.b = RY_j5
            P1.c = RZ_j6
            response = self.MovJ_l.call_async(P1)
            print(response)
        elif Move == "MovL":
            P1 = MovL.Request()
            P1.mode = 0
            P1.x = X_j1
            P1.y = Y_j2
            P1.z = Z_j3
            P1.a = RX_j4
            P1.b = RY_j5
            P1.c = RZ_j6
            response = self.MovL_l.call_async(P1)
            print(response)
        else:
```

```
            print("无该指令")

    def DO(self, index, status):  # IO 控制夹爪/气泵
        DO_V = DO.Request()
        DO_V.index = index
        DO_V.status = status
        response = self.DO_l.call_async(DO_V)
        print(response)


def main(args=None):
    rclpy.init(args=args)
    # ROS2 Python接口初始化
    node = adderClient("service_adder_client")
    # 创建ROS2节点对象并进行初始化
    node.send_request()
    # 发送服务请求
    talker.point("MovJ", 350, -80, 0, 0, 0, 0)
    talker.point("MovL", 350, -80, -40, 0, 0, 0)
    talker.DO(1, 1)
    talker.point("MovJ", 350, -80, 0, 0, 0, 0)
    talker.point("MovJ", 350, 60, 0, 0, 0, 0)
    talker.point("MovL", 350, 60, -40, 0, 0, 0)
    talker.DO(1, 0)
    talker.point("MovJ", 350, 60, 0, 0, 0, 0)
    time.sleep(3)
    node.destroy_node()
    # 销毁节点对象
    rclpy.shutdown()
    # 关闭ROS2 Python接口
```