# Docker interaction



The tutorial mainly introduces the data and hardware interaction between the host and Docker.

## 1. Bash script

Bash (Bourne Again SHell) is a scripting language for automated tasks. Users can combine multiple commands and execute them in sequence!

```
Docker commands provide optional parameters to start containers, but too many
parameters will affect the reading and understanding of the command
```

### 1.1, Basic use (Docker)

> **Basic script**

```
#!/bin/bash

docker run -it \
<image_name>:<tag> /bin/bash
```

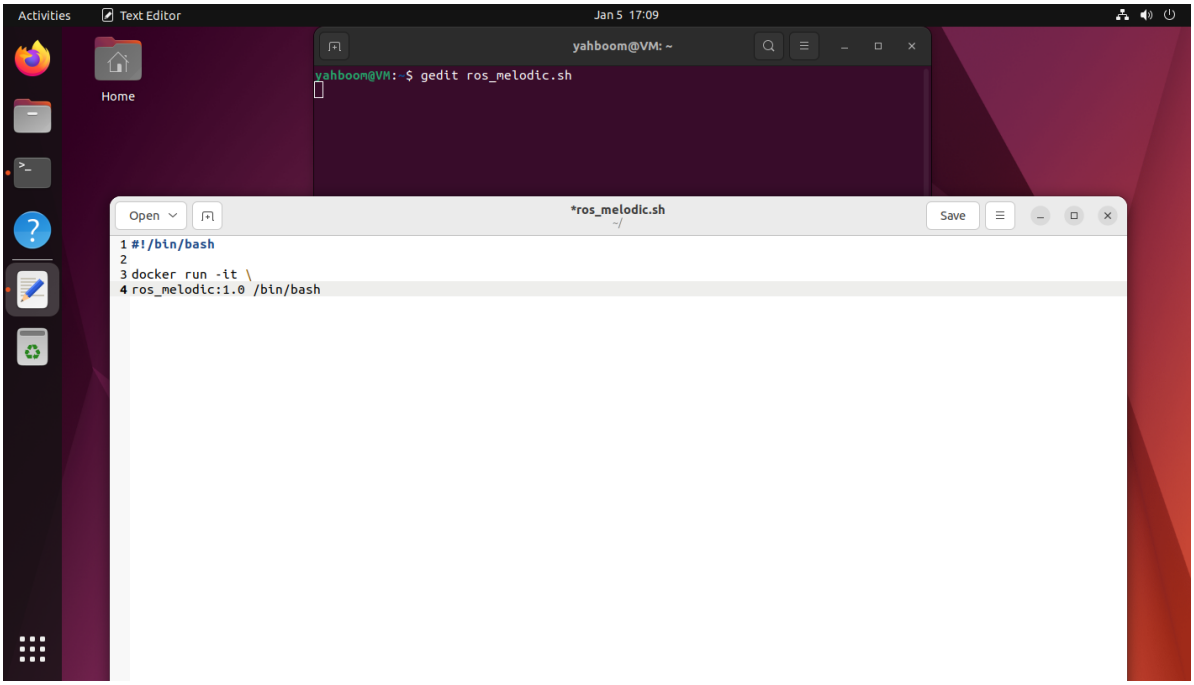
> **Sample script**

Write the example to the `ros_melodic.sh` file:

```
gedit ros_melodic.sh
```

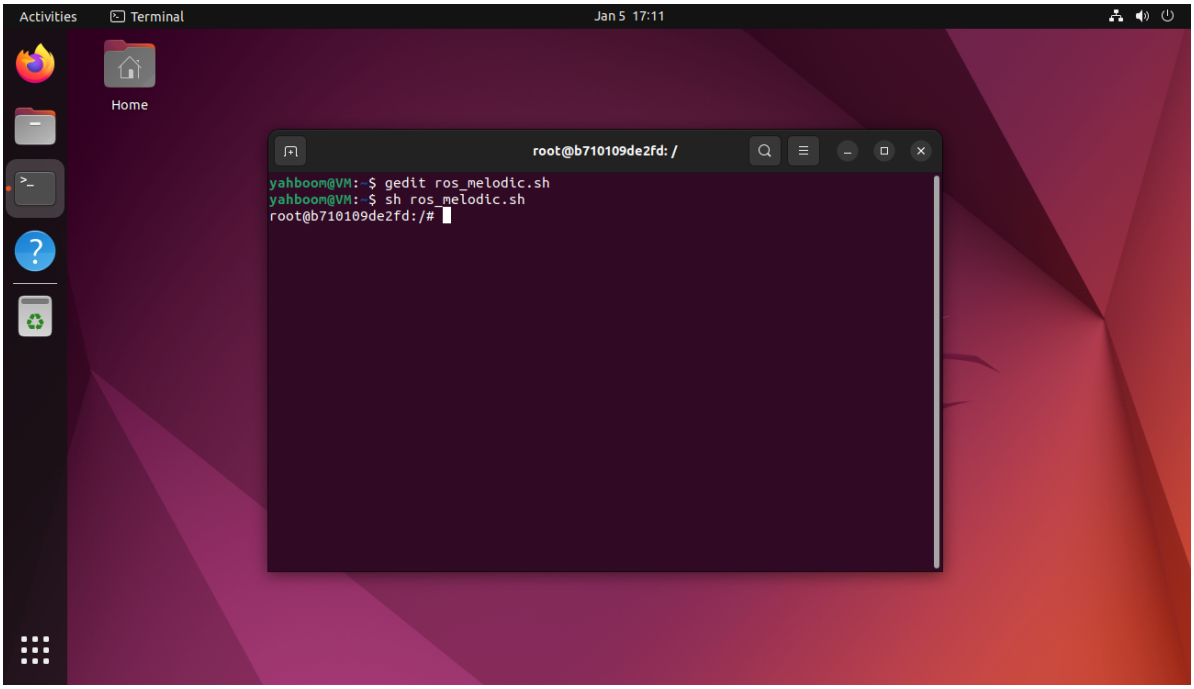
```
#!/bin/bash

docker run -it \
ros_melodic:1.0 /bin/bash
```




## 1.2, Run the script

```
sh ros_melodic.sh
```




# 2. Shared data

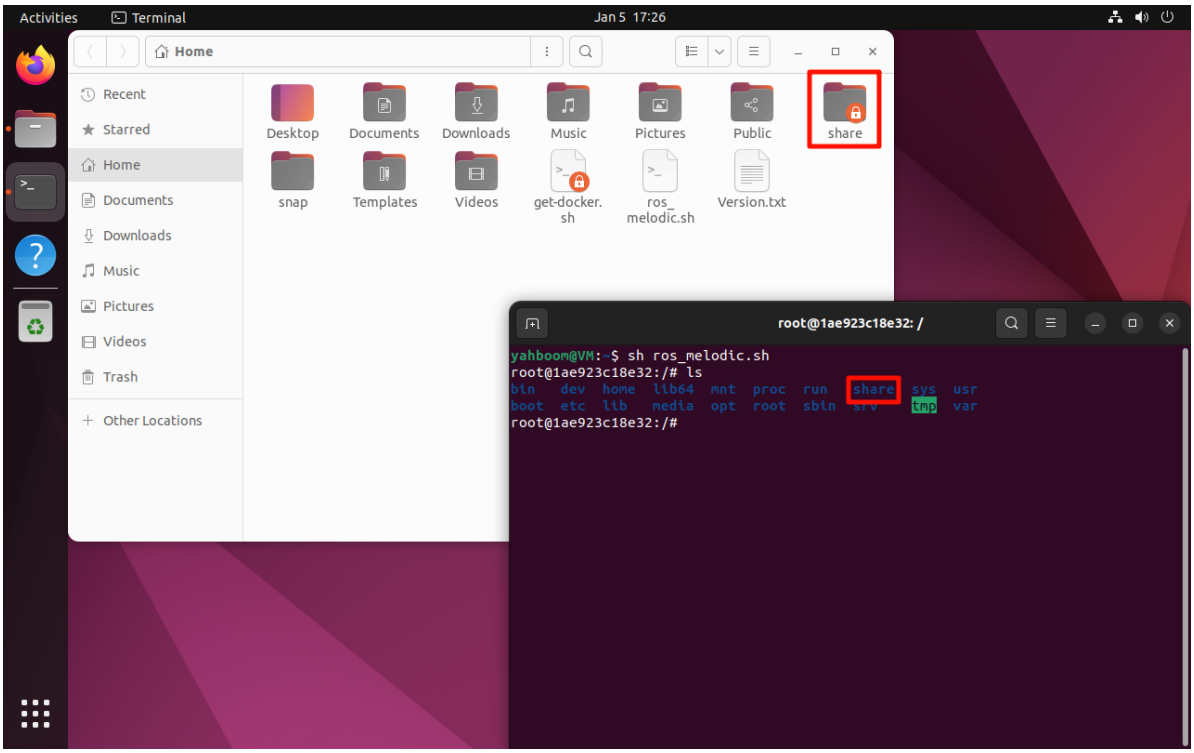
## 2.1. Shared folder

Share the host's `share` folder to Docker.

> **Sample script**

```
#!/bin/bash

docker run -it \
-v /home/yahboom/share:/share \
ros_melodic:1.0 /bin/bash
```




## 2.2. Shared network

Share the host's network to Docker.

> **Sample script**

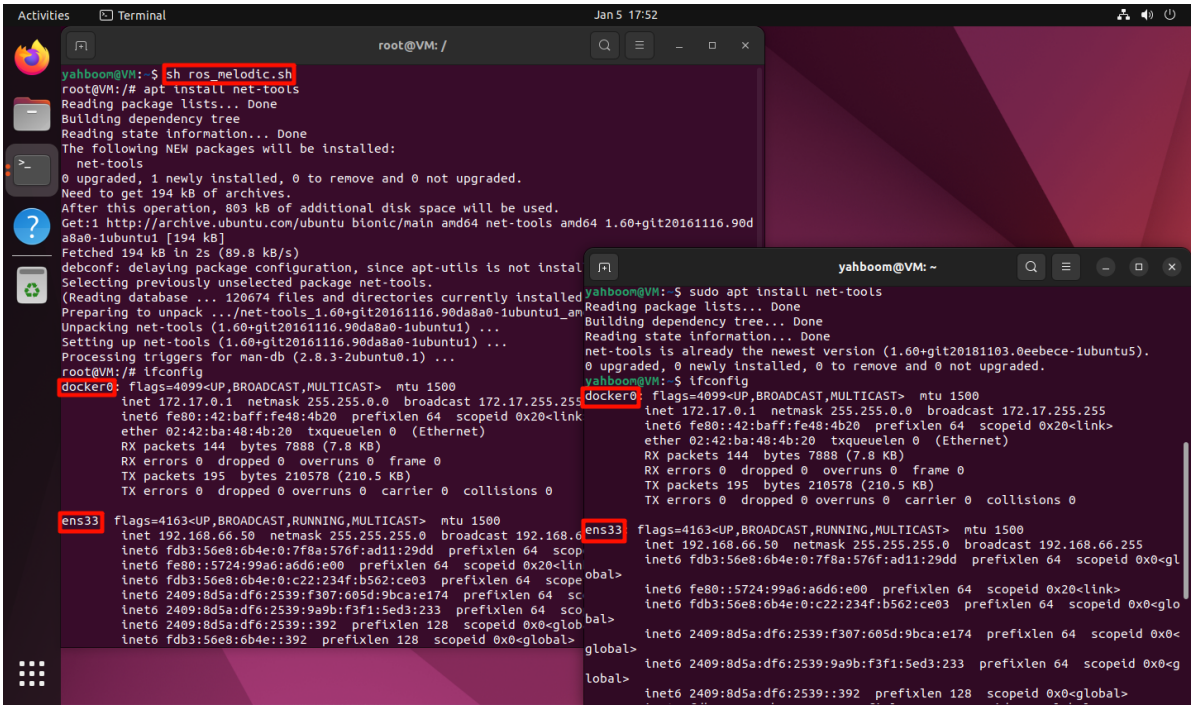
```
#!/bin/bash

docker run -it \
--net=host \
-v /home/yahboom/share:/share \
ros_melodic:1.0 /bin/bash
```

If you cannot use the `ifconfig` command, install `net-tools`:

```
sudo apt install net-tools -y
```

```
ifconfig
```




# 3. Shared hardware

## 3.1. Shared graphical interface

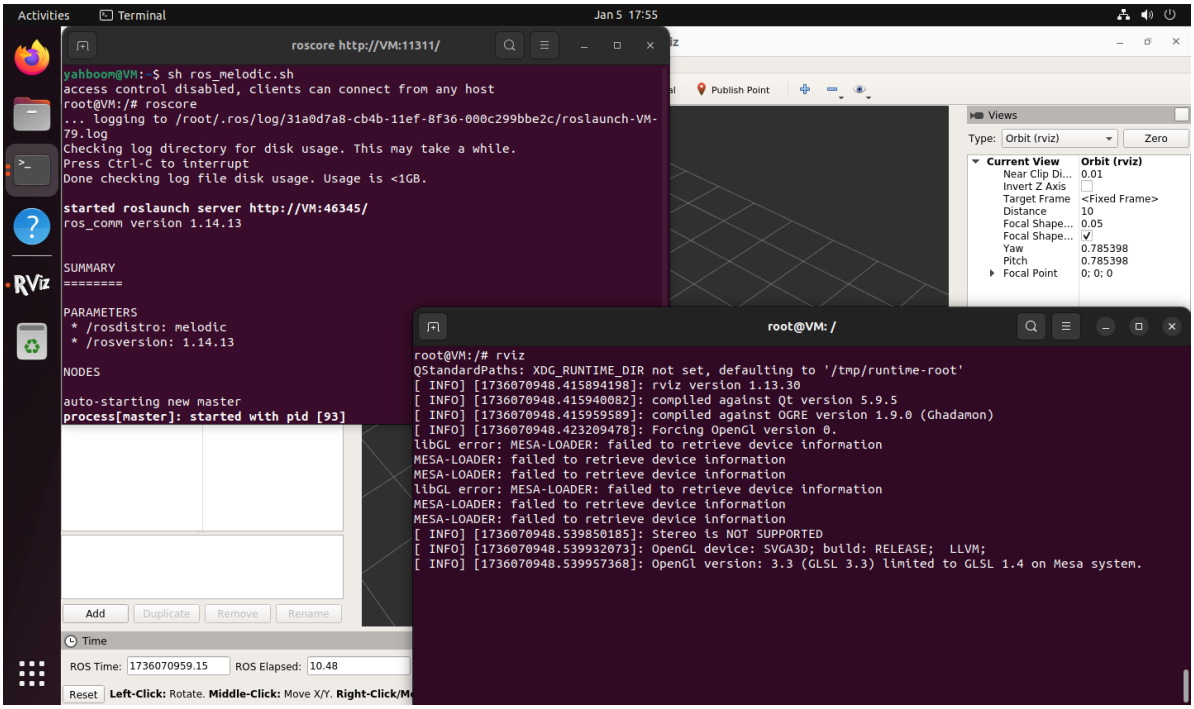
Docker runs graphical interface programs and needs to output to the local display graphical interface.

> **Sample script**

```
#!/bin/bash

xhost +

docker run -it \
--net=host \
-e DISPLAY=$DISPLAY \
-e "QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/yahboom/share:/share \
ros_melodic:1.0 /bin/bash
```




## 3.2, ordinary camera

Ordinary camera mapping to Docker: just need to add the `/dev/video*` device number corresponding to the camera

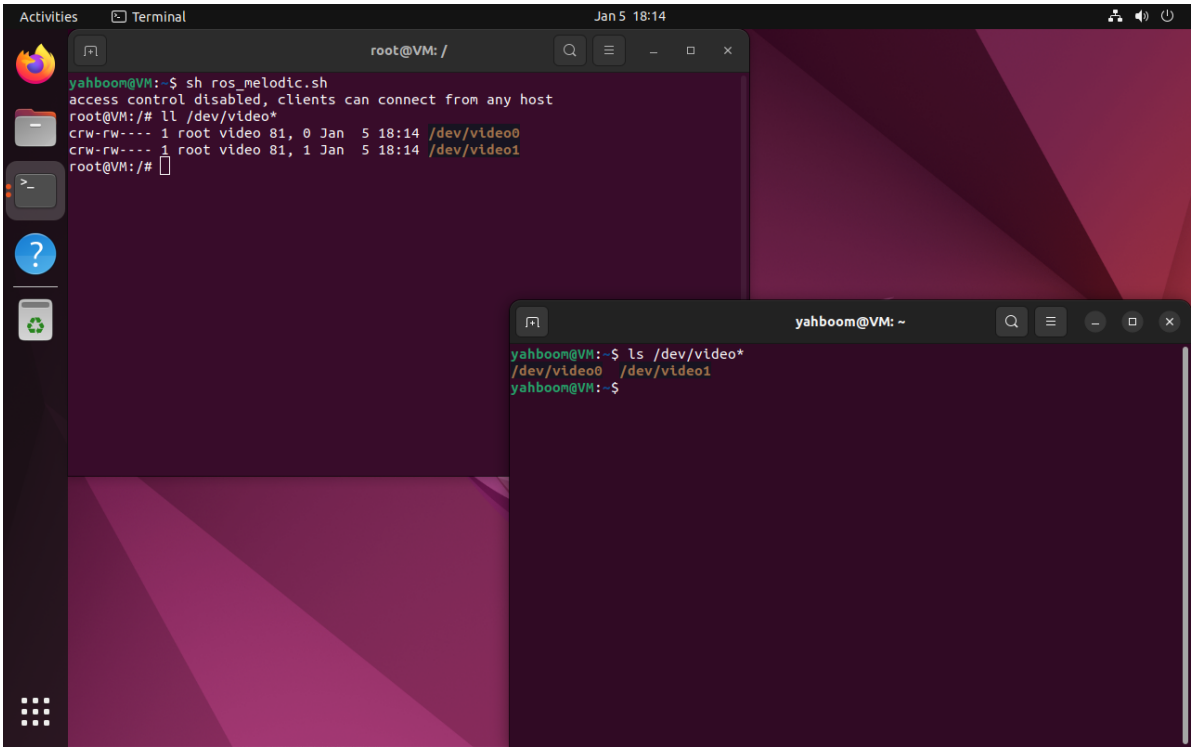
```
If only one camera is connected, it generally corresponds to /dev/video0
```

> **Sample script**

```
#!/bin/bash

xhost +

docker run -it \
--net=host \
-e DISPLAY=$DISPLAY \
-e "QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/yahboom/share:/share \
--device=/dev/video0 \
ros_melodic:1.0 /bin/bash
```

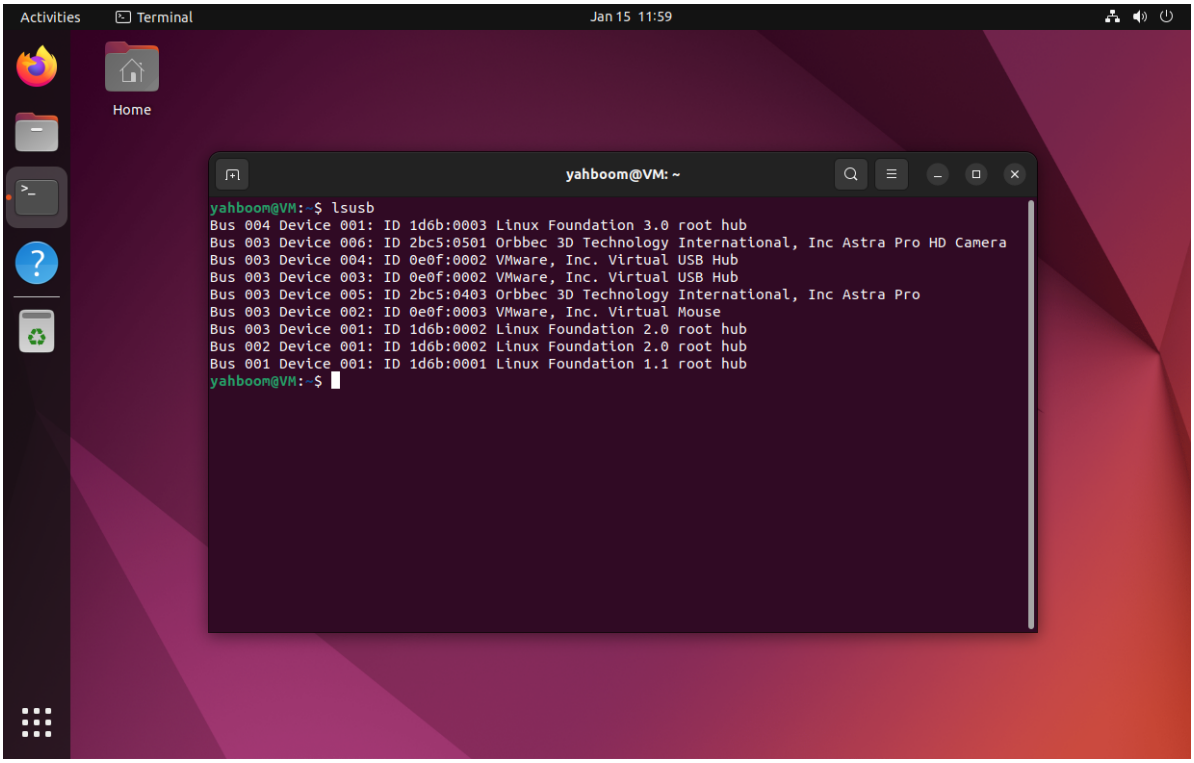



## 3.3, Depth Camera

Compared with ordinary cameras, depth cameras require additional information to call the camera: Take `Astra Pro` as an example

> **View USB device information**

```
lsusb
```




Information we need to pay attention to: USB bus number, device number on the USB bus, device vendor ID, product ID

```
Bus 003 Device 005: ID 2bc5:0403 Orbbec 3D Technology International, Inc Astra
Pro
Bus 003 Device 006: ID 2bc5:0501 Orbbec 3D Technology International, Inc Astra
Pro HD Camera
```

`USB bus number`: Bus 003

`Device number on the USB bus`: Device 005, Device 006

`Vendor ID of the device`: 2bc5

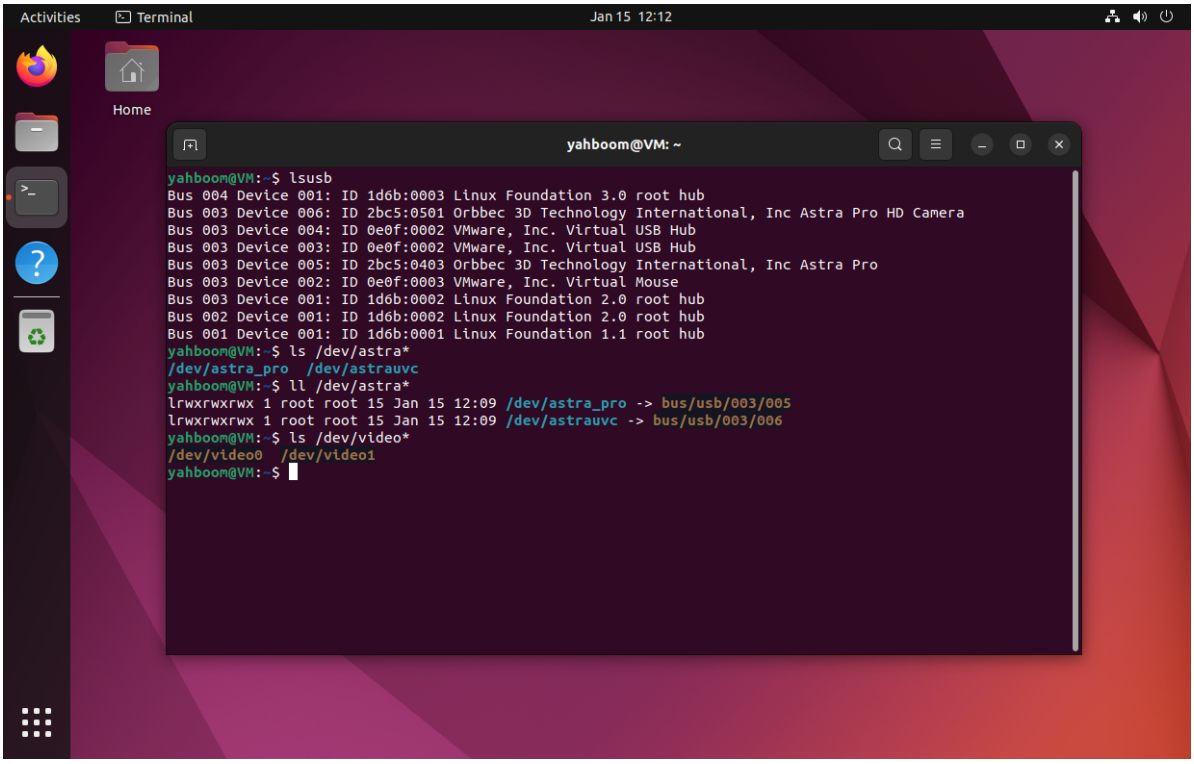
`Product ID`: 0403, 0501

Device can be located and device ID remapped based on vendor ID and product ID, where USB bus number and device number on USB bus affect the device connected in Docker.

> **Device ID remap**

The binding process and method are not introduced here, and the results are directly displayed. Device ID remap is not part of this tutorial.

Every time you start docker, you need to pay attention to the USB bus number and device number on USB bus corresponding to the depth camera. If the two are inconsistent, the camera will always prompt that the device cannot be connected when it is started in Docker.




> **Sample Script**

```
#!/bin/bash

xhost +

docker run -it \
--net=host \
-e DISPLAY=$DISPLAY \
-e "QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/yahboom/share:/share \
```

```
-v /dev/bus/usb/003/005:/dev/bus/usb/003/005 \
-v /dev/bus/usb/003/006:/dev/bus/usb/003/006 \
--device=/dev/astra_pro \
--device=/dev/astrauvc \
--device=/dev/video0 \
--device=/dev/video1 \
ros_melodic:1.0 /bin/bash
```