Must read before use

After getting the car, you need to set the IP and ROS_DOMAIN_ID. The former is to connect to the agent, and the latter is to interact with the virtual machine. Distributed multi-machine communication

1、Set up the board to connect to wifi

Modify the config_robot.py code in the /home/yahboom directory and find the following parts,

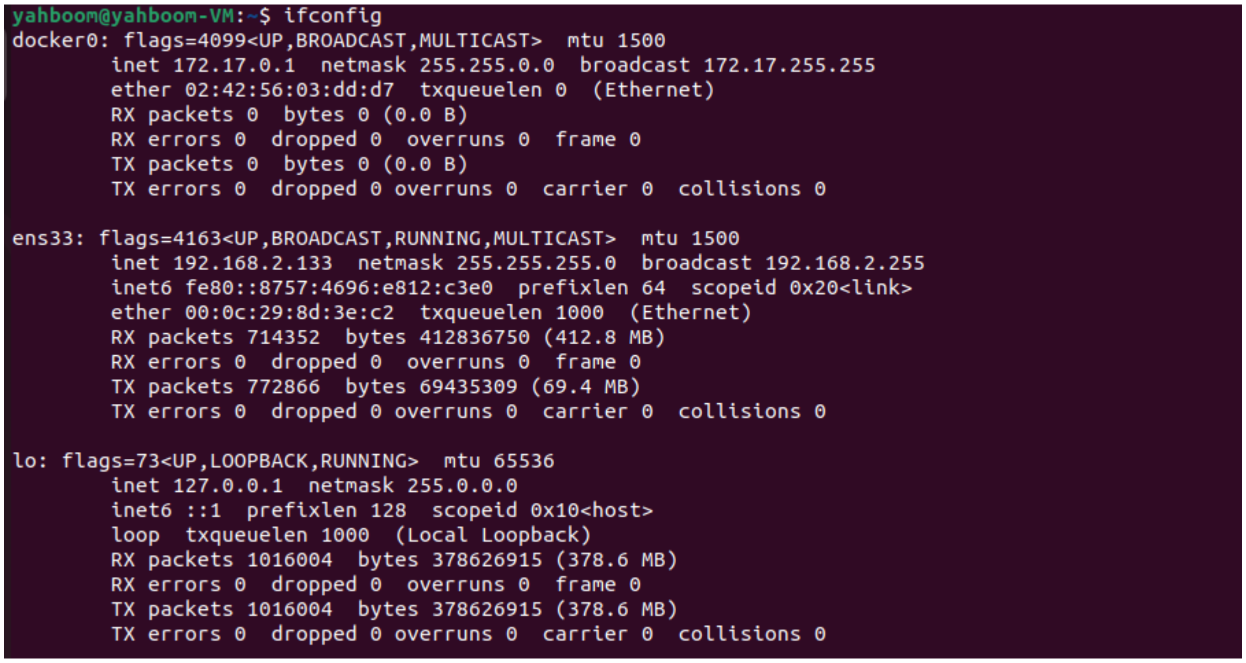
```
robot.set_wifi_config("ssid123", "passwd123")
```

Change ssid123 to your own WiFi name, and change passws123 to the password corresponding to your own WiFi. Save and exit after modification.

2、Set IP

Taking the virtual machine provided by yahboom as an example, query the current IP of the virtual machine and enter in the terminal

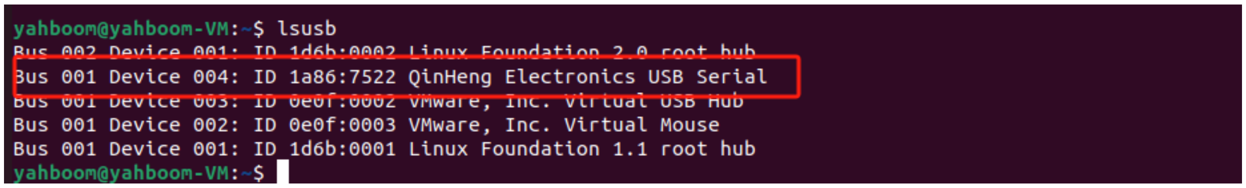
```
ifconfig
```



```
yahboom@yahboom-VM:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:56:03:dd:d7  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.133  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::8757:4696:e812:c3e0  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8d:3e:c2  txqueuelen 1000  (Ethernet)
        RX packets 714352  bytes 412836750 (412.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 772866  bytes 69435309 (69.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1016004  bytes 378626915 (378.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1016004  bytes 378626915 (378.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

As can be seen from the picture, the query here shows that my IP is 192.168.2.133. This is the IP when we start the agent, so we need to set the board to this. IP can connect to the proxy

Use a USB to Type-C cable to connect the board and the virtual machine, and make sure the virtual machine is connected to the board.



```
yahboom@yahboom-VM:~$ lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 1a86:7522 QinHeng Electronics USB 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
yahboom@yahboom-VM:~$
```

Then modify the test_microros.py code in the /home/yahboom directory and find the following parts,

```
robot.set_udp_config([192, 168, 2, 133], 8090)
```

Change [192, 168, 2, 133] here to the ip you queried

Assume that robot.set_ros_domain_id(20) is set to 20, then modify the config_robot.py file as follows,

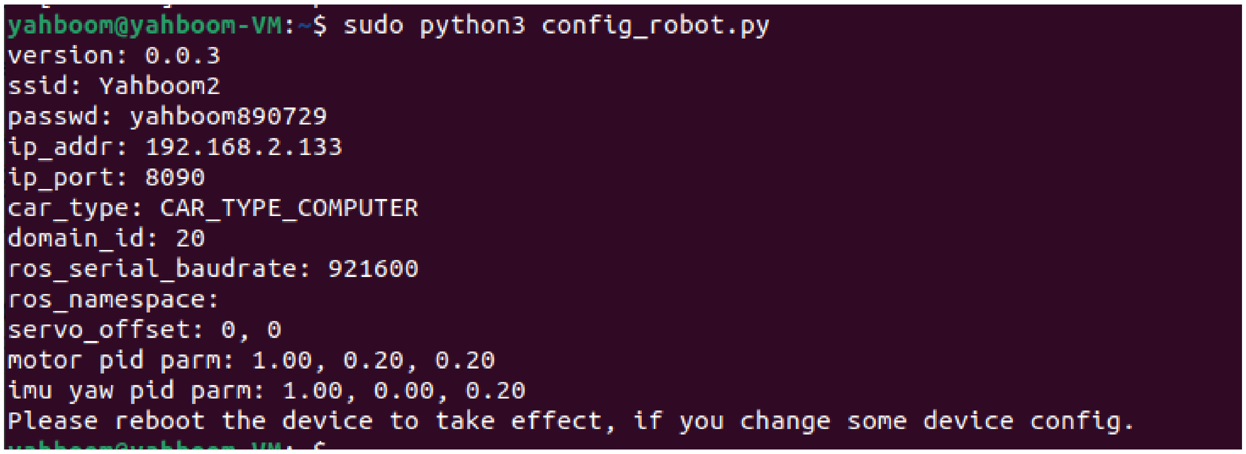
```
robot.set_ros_domain_id(20)
```

Change the 20 inside to a custom number, which cannot exceed 100.

After the modification is completed, save and exit, and run it in the terminal.

```
cd ~
sudo python3 config_robot.py
```

If the following screen appears, it means the modification is successful. The content in the picture shall be subject to the actual modification. Then you need to restart the board to take effect.



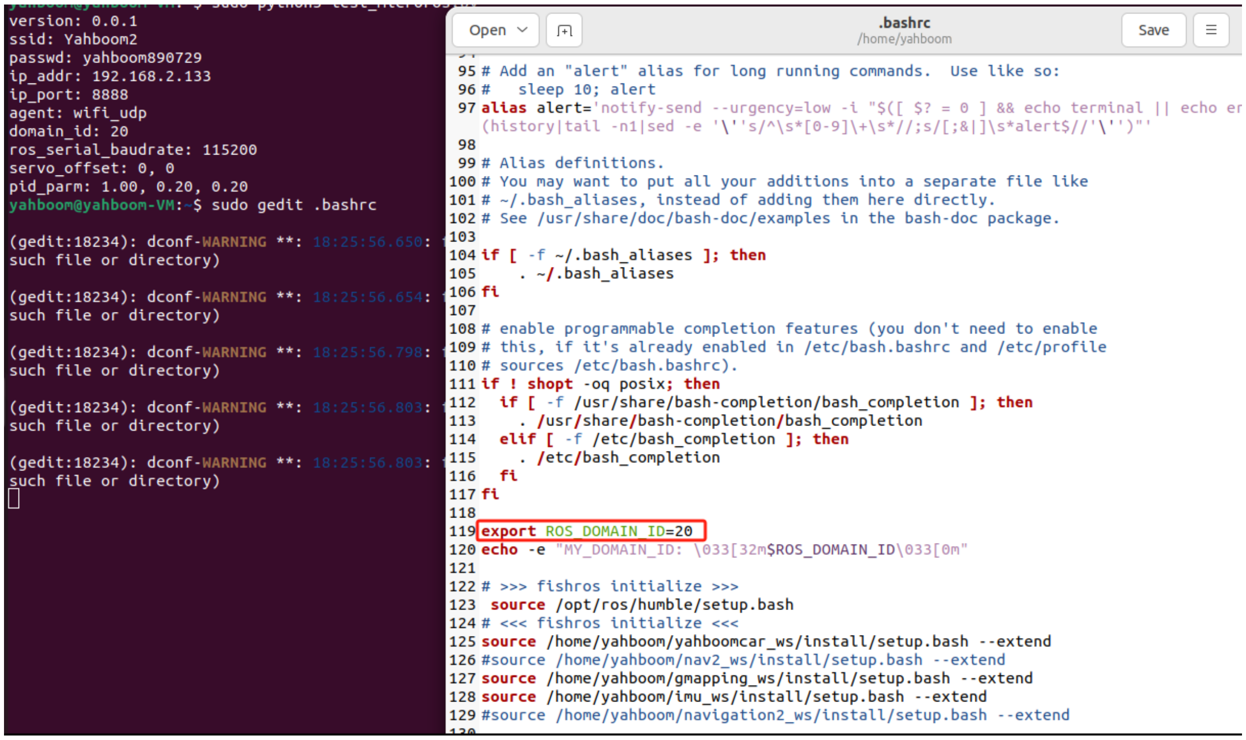
```
yahboom@yahboom-VM:~$ sudo python3 config_robot.py
version: 0.0.3
ssid: Yahboom2
passwd: yahboom890729
ip_addr: 192.168.2.133
ip_port: 8090
car_type: CAR_TYPE_COMPUTER
domain_id: 20
ros_serial_baudrate: 921600
ros_namespace:
servo_offset: 0, 0
motor pid parm: 1.00, 0.20, 0.20
imu yaw pid parm: 1.00, 0.00, 0.20
Please reboot the device to take effect, if you change some device config.
```

2、Virtual machine settings ROS_DOMAIN_ID

Then you also need to set the same ROS_DOMAIN_ID in the ~/.bashrc file of the virtual machine to achieve distributed multi-machine communication between the board and the virtual machine.

Modify the ~/.bashrc file of the virtual machine and enter it in the terminal.

```
sudo gedit .bashrc
```

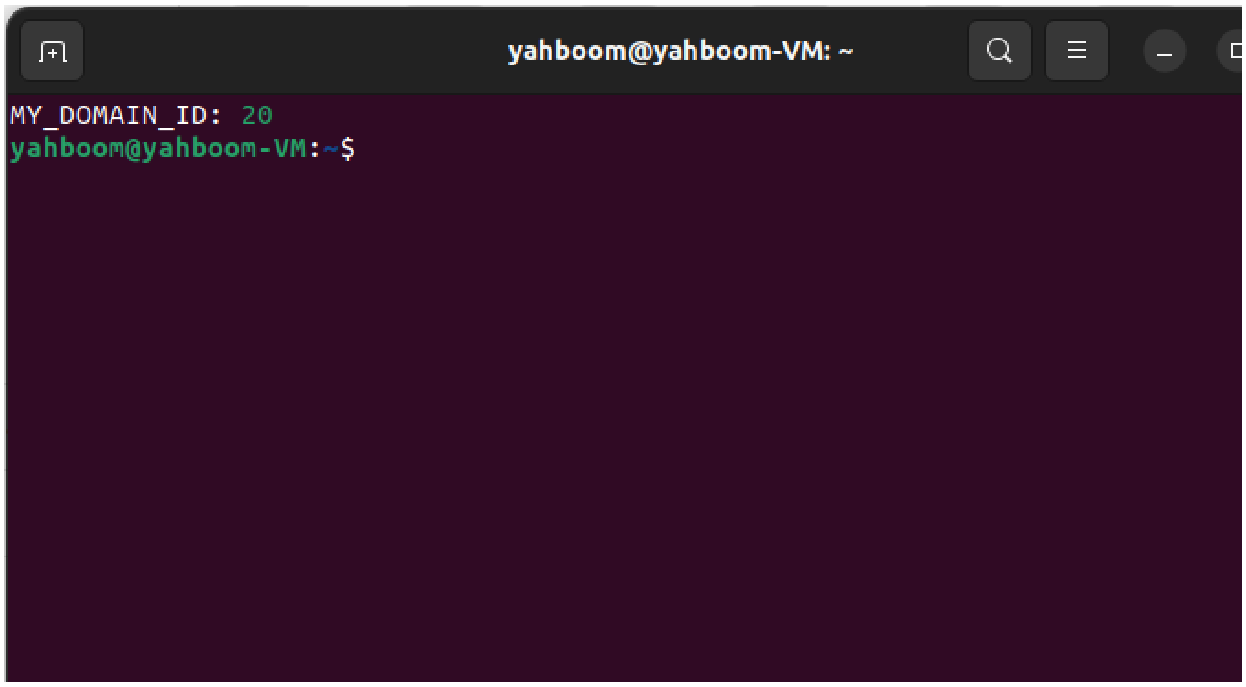


```
version: 0.0.1
ssid: Yahboom2
passwd: yahboom890729
ip_addr: 192.168.2.133
ip_port: 8888
agent: wifi_udp
domain_id: 20
ros_serial_baudrate: 115200
servo_offset: 0, 0
pid_parm: 1.00, 0.20, 0.20
yahboom@yahboom-VM:~$ sudo gedit .bashrc

(gedit:18234): dconf-WARNING **: 18:25:56.650:
such file or directory)

(gedit:18234): dconf-WARNING **: 18:25:56.654:
such file or directory)

(gedit:18234): dconf-WARNING **: 18:25:56.798:
such file or directory)

(gedit:18234): dconf-WARNING **: 18:25:56.803:
such file or directory)

(gedit:18234): dconf-WARNING **: 18:25:56.803:
such file or directory)
```

```
# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo er
(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# 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 ROS_DOMAIN_ID=20
echo -e "MY_DOMAIN_ID: \033[32m$ROS_DOMAIN_ID\033[0m"

# >>> fishros initialize >>>
 source /opt/ros/humble/setup.bash
# <<< fishros initialize <<<
source /home/yahboom/yahboomcar_ws/install/setup.bash --extend
#source /home/yahboom/nav2_ws/install/setup.bash --extend
source /home/yahboom/gmapping_ws/install/setup.bash --extend
source /home/yahboom/imu_ws/install/setup.bash --extend
#source /home/yahboom/navigation2_ws/install/setup.bash --extend
```

Find ROS_DOMAIN_ID and set it to be consistent with robot.set_ros_domain_id(20) set in 1,

Assume that the setting is 20, fill in 20 here, then save and exit, reopen the terminal, the terminal will display the set ROS_DOMAIN_ID value,

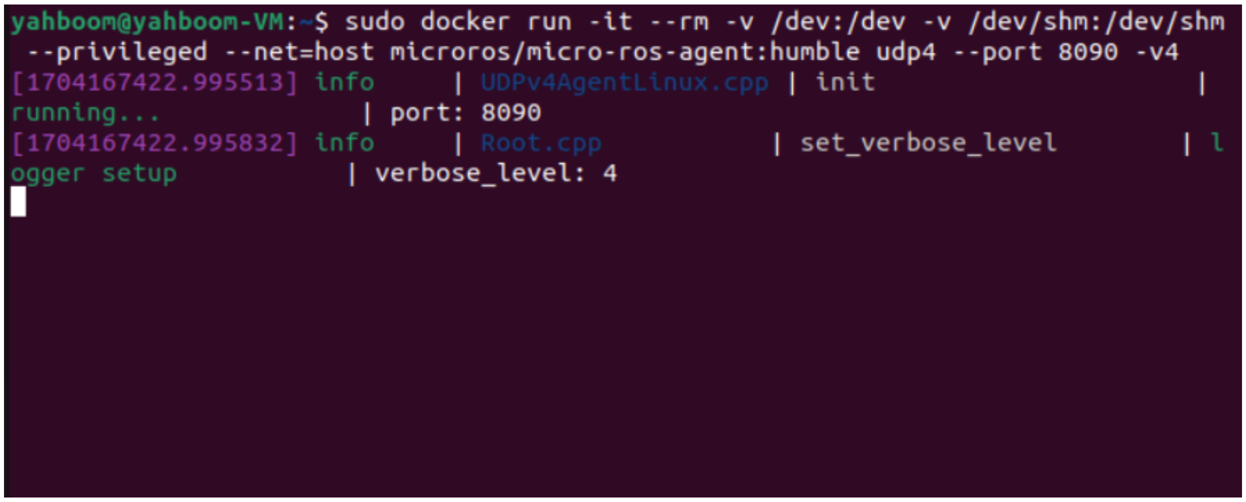


```
MY_DOMAIN_ID: 20
yahboom@yahboom-VM:~$
```

3. Test whether the modification is completed

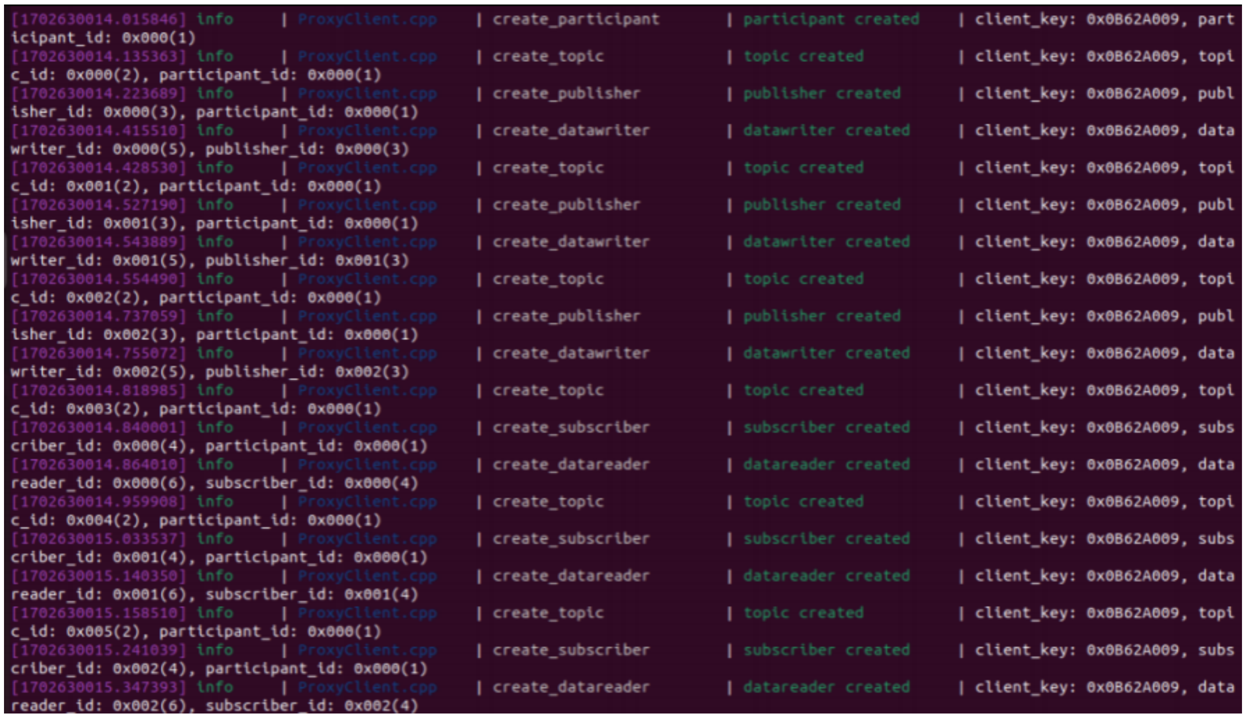
Enter the following command in the virtual machine terminal to enable the agent:

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host
microros/micro-ros-agent:humble udp4 --port 8090 -v4
```



```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8090 -v4
[1704167422.995513] info     | UDPv4AgentLinux.cpp | init                     |
running...               | port: 8090
[1704167422.995832] info     | Root.cpp            | set_verbose_level        | l
ogger setup              | verbose_level: 4
```

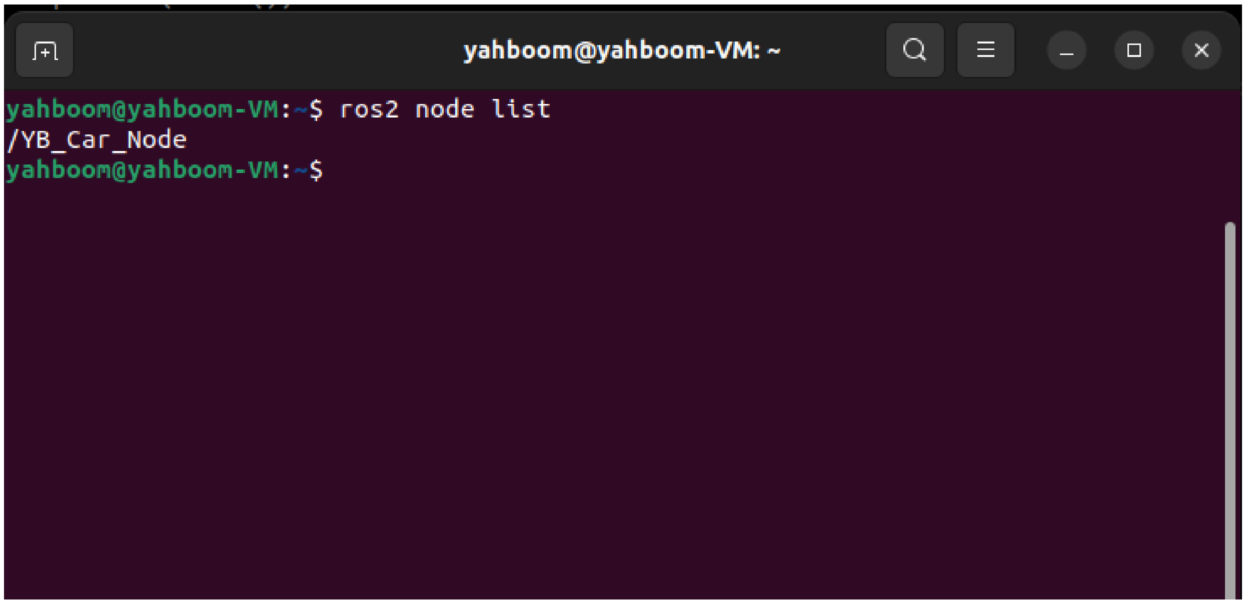
Then, turn on the car switch and wait for the car to connect to the agent. The connection is successful, as shown in the figure below.



```
[1702630014.015846] info     | ProxyClient.cpp    | create_participant       | participant created      | client_key: 0x0B62A009, part
icipant_id: 0x000(1)
[1702630014.135363] info     | ProxyClient.cpp    | create_topic             | topic created            | client_key: 0x0B62A009, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info     | ProxyClient.cpp    | create_publisher         | publisher created        | client_key: 0x0B62A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info     | ProxyClient.cpp    | create_datawriter        | datawriter created       | client_key: 0x0B62A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info     | ProxyClient.cpp    | create_topic             | topic created            | client_key: 0x0B62A009, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info     | ProxyClient.cpp    | create_publisher         | publisher created        | client_key: 0x0B62A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info     | ProxyClient.cpp    | create_datawriter        | datawriter created       | client_key: 0x0B62A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info     | ProxyClient.cpp    | create_topic             | topic created            | client_key: 0x0B62A009, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info     | ProxyClient.cpp    | create_publisher         | publisher created        | client_key: 0x0B62A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info     | ProxyClient.cpp    | create_datawriter        | datawriter created       | client_key: 0x0B62A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818985] info     | ProxyClient.cpp    | create_topic             | topic created            | client_key: 0x0B62A009, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info     | ProxyClient.cpp    | create_subscriber        | subscriber created       | client_key: 0x0B62A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info     | ProxyClient.cpp    | create_datareader        | datareader created       | client_key: 0x0B62A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info     | ProxyClient.cpp    | create_topic             | topic created            | client_key: 0x0B62A009, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info     | ProxyClient.cpp    | create_subscriber        | subscriber created       | client_key: 0x0B62A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info     | ProxyClient.cpp    | create_datareader        | datareader created       | client_key: 0x0B62A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info     | ProxyClient.cpp    | create_topic             | topic created            | client_key: 0x0B62A009, topi
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info     | ProxyClient.cpp    | create_subscriber        | subscriber created       | client_key: 0x0B62A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info     | ProxyClient.cpp    | create_datareader        | datareader created       | client_key: 0x0B62A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

Then reopen a terminal and enter the following command to query the currently running node:

```
ros2 node list
```



```
yahboom@yahboom-VM:~$ ros2 node list
/YB_Car_Node
yahboom@yahboom-VM:~$
```

When the car is connected to the agent, a node program will be run. If the node can be queried on the virtual machine, it means that the two have achieved distributed multi-machine communication.

Note: When running any routine, the agent only needs to be started normally once. Repeatedly starting the agent will cause the agent to report an error. If you need to restart the agent

The manager needs to ensure that the last agent used is shut down before starting again.