

A Guide to Setup ROS2 (Jetson Nano) Via Windows VM VirtualBox

March 10th 2024

1 Setting up Environment

1.1 Install NDVI and Ubuntu 20.04 for VM VirtualBox

NVIDIA JetPack SDK is a complete solution for building AI applications. It includes operating system images, libraries, APIs, developer tools, and related documentation for Jetson products. Before using the development components, JetPack installation is required. The functionality of Jetpack 5.0, released in 2022, corresponds to Ubuntu 20.04. However, we are still using Jetpack 4.X (Ubuntu 18.04) in the project development process, so we need to upgrade to Ubuntu 20.04 before installing ROS2.

The upgrading steps are described as follows:

1) *Install VM VirtualBox*

The software can be downloaded from the website: <https://www.virtualbox.org/wiki/Downloads>.

Download Ubuntu 18.04 at the website: <https://releases.ubuntu.com/18.04/>.

Name	Last modified	Size	Description
Parent Directory		-	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (standard download)
ubuntu-18.04.6-desktop-amd64.iso.torrent	2021-09-16 21:46	188K	Desktop image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-18.04.6-desktop-amd64.iso.zsync	2021-09-16 21:46	4.7M	Desktop image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-18.04.6-desktop-amd64.list	2021-09-15 20:42	7.8K	Desktop image for 64-bit PC (AMD64) computers (file listing)

Fig. 1 Download the Ubuntu 18.04 ISO file.

2) *Create virtual machine (VM) on local machine (Windows)*

Launch the VirtualBox application in your computer. Click on “New” on the interface to create new VM, then enter the name and select OS for the new VM.

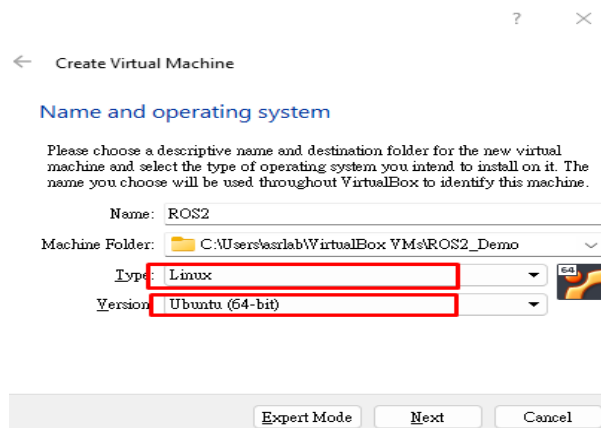


Fig. 2 Create new virtual machine.

Allocate memory (8192 MB as recommendation) and click on “Create a virtual hard disk now”.

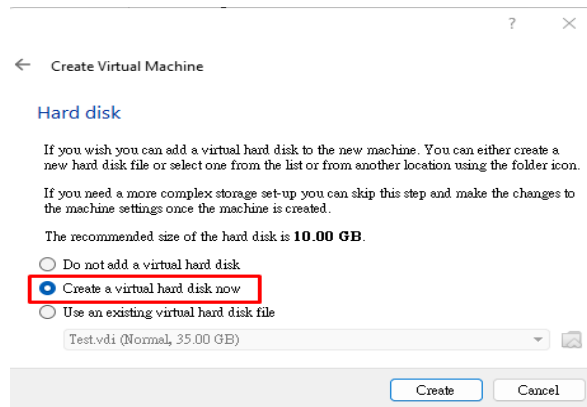


Fig. 3 Create a virtual hard disk.

Click on “Create” button and select “Dynamic allocated” option.

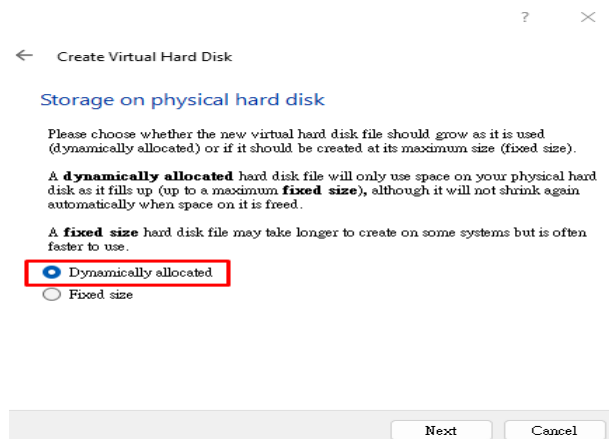


Fig. 4 Allocate the virtual hard disk.

Select the size of virtual hard disk. Finally, click on “Create” to generate new VM.

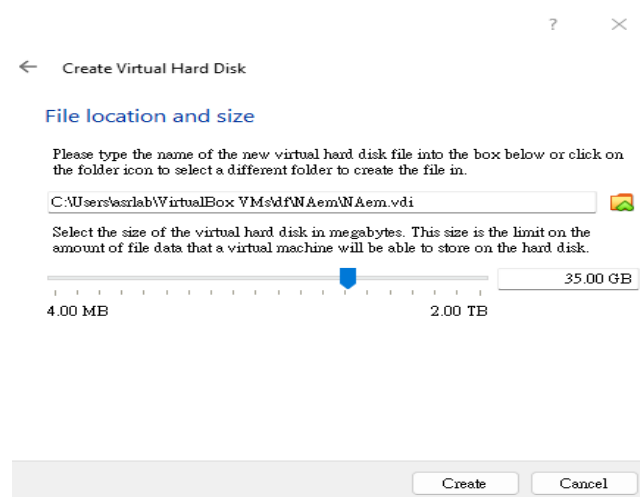


Fig. 5 Select the size of virtual hard disk.

3) VM Configuration

On the new VM, click on “Settings” option, then select “Storage → Empty” choose a Ubuntu 18.04 being downloaded before.

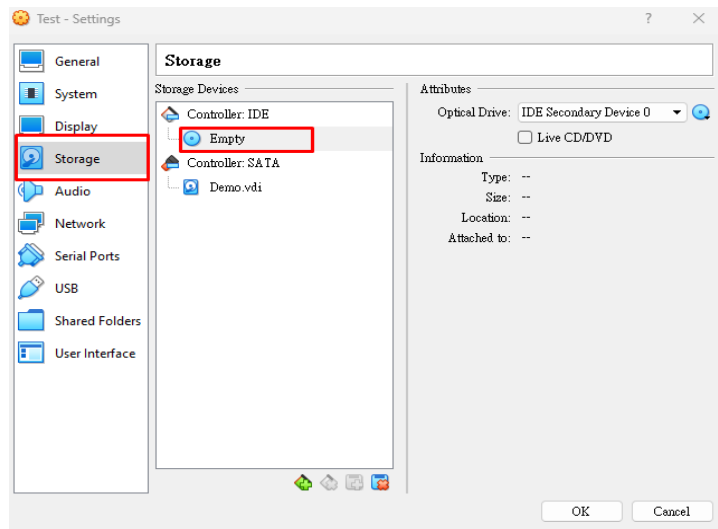


Fig. 6 Insert the Ubuntu 18.04 for IDE.

Click “OK” button and begin to start VM.

4) Setup the Ubuntu for VM

Enter “Try or Install Ubuntu”.

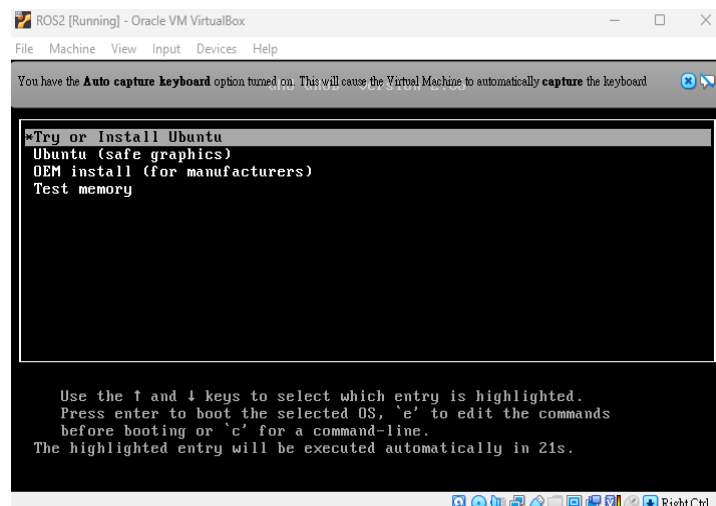


Fig. 7 Install Ubuntu 18.04.

Click on “Install Ubuntu” option (Choose keyboard layout, Continue and Install now).

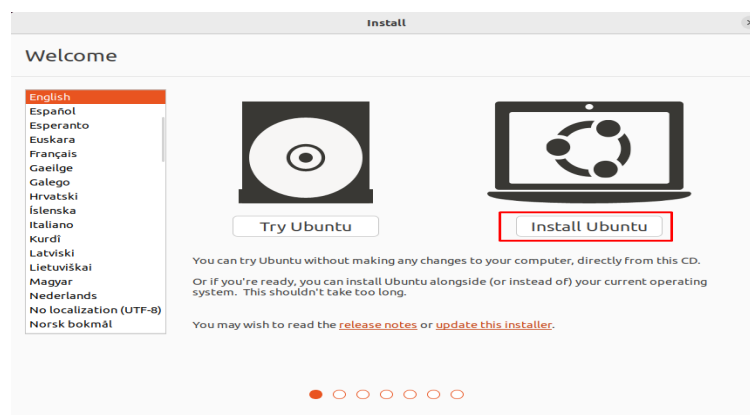


Fig. 8 Install Ubuntu 18.04 option.

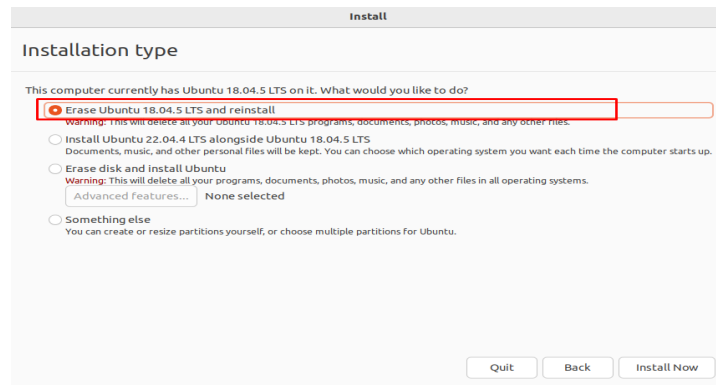


Fig. 9 Select new installation for Ubuntu 18.04.

Select location and datetime and click “Next”.

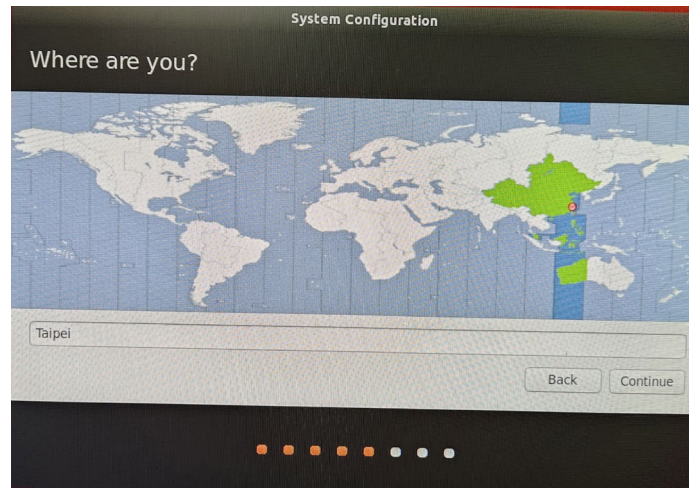


Fig. 10 Configure location.

Fill in the information to create account.

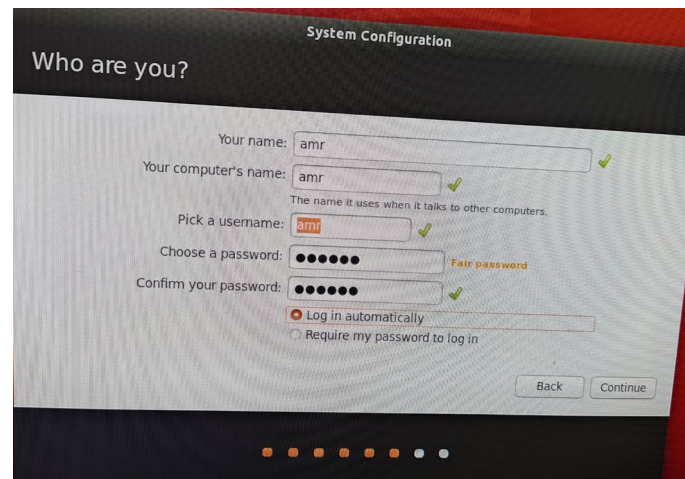


Fig. 11 Create account for VM.

1.2 Install L4T Driver Package (BSP) and Sample Root File System

Create folder “NDVI_SKD”.

```
mkdir ~/NDVI_SKD
```

Download two packages at the website: <https://developer.nvidia.com/embedded/linux-tegra-r3261>.

	Jetson AGX Xavier Series, Xavier NX and TX2 Series	Jetson Nano, Nano 2GB and TX1
DRIVERS	L4T Driver Package (BSP)	L4T Driver Package (BSP)
	Sample Root Filesystem	Sample Root Filesystem
	NVIDIA Hardware Acceleration in the WebRTC Framework	
SOURCES	L4T Driver Package (BSP) Sources	L4T Driver Package (BSP) Sources
	Cboot Sources T186	
	Cboot Sources T194	
	Free RTOS Sources	

Fig. 12 Download packages.

These downloaded files should be relocated in “NDVI_SKD”. Install “qemu” dependencies

```
sudo apt install qemu qemu-user-static -y
```

Extract the packages in “NDVI_SKD” folder.

```
cd ~/NDVI_SKD
tar xf Jetson-210 cd Linux_for_Tegra/rootfs/
sudo tar xpf ../../Tegra_Linux_Sample-Root-Filesystem_R32.6.1_aarch64.tbz
cd .. && sudo ./apply_binaries.sh _Linux_R32.6.1_aarch64.tbz2
```

1.3 Upgrade Ubuntu 20.04

1) Remove unused applications

```
# remove games
sudo apt purge aisleriot gnome-mahjongg gnome-mines gnome-sudoku -y
# remove office
sudo apt purge libreoffice* -y
# remove chrome browser
sudo apt purge chromium-* -y
# remove unused applications
sudo apt purge leafpad thunderbird smplayer gnome-todo -y
# remove unused dependencies
sudo apt autoclean -y && sudo apt autoremove -y
```

2) Upgrade the system

After removing unnecessary software, you can connect the internet and upgrade the system.

```
sudo apt update -y && sudo apt upgrade -y
sudo apt autoremove -y
```

Reboot the system

```
sudo reboot
```

After rebooting, set `Prompt=1ts` in the `/etc/update-manager/release-upgrades` file to enable the “Update Manager” to search for long-term support version updates. The modified file should look like this:

```
sudo nano /etc/update-manager/release-upgrades
```

Then we can upgrade the system into newer distribution:

```
# upgrade to new distribution
sudo apt update -y sudo apt dist-upgrade -y
# reboot system
sudo reboot
```

After rebooting, use the command to see if there are any new releases available:

```
sudo do-release-upgrade -c
```

Once you've confirmed that 20.04 exists, use the command to upgrade:

```
sudo do-release-upgrade
```

During the upgrade process, you will need to confirm certain options in the terminal. Answer all questions with the default values. At the final step, some redundant packages have been removed, and Ubuntu will request a restart. At this step, type 'N' to not restart!

Before restarting, you need to modify some files. First, in the `/etc/gdm3/custom.conf` file, comment out the line `WaylandEnable=false`. Then, in the `/etc/X11/xorg.conf` file, uncomment the line `# Driver "nvidia"`.

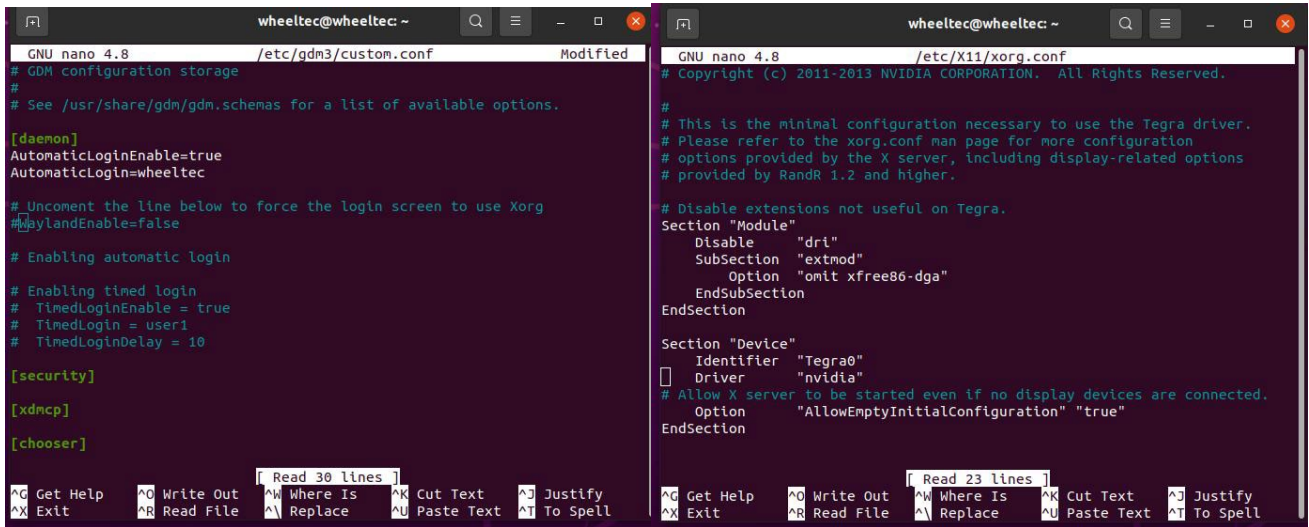


Fig. 13 Edit necessary files.

Finally, in the `/etc/update-manager/release-upgrades` file, reset the upgrade manager to **"never"**.

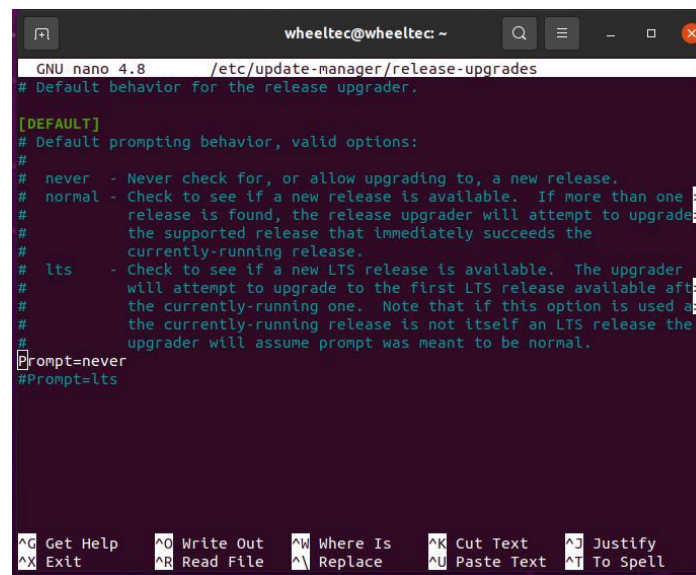


Fig. 14 Deactivate the upgrade.

After restarting, you can enter Ubuntu 20.04. To remove the distorted Nvidia logo in the top bar:

```
cd /usr/share/nvmodel_indicator
sudo mv nv_logo.svg no_logo.svg
```

Note: In Ubuntu 20.04, do not install Chromium as it can interfere with Snap installation. Use the pre-installed Mozilla Firefox instead.

1.4 Installing ROS2-Galactic on Ubuntu 20.04

There are two ways to install ROS2, one is from source and the other is from binaries. This tutorial uses binary installation. The installation steps are described as follows:

1) *Set the system locale*

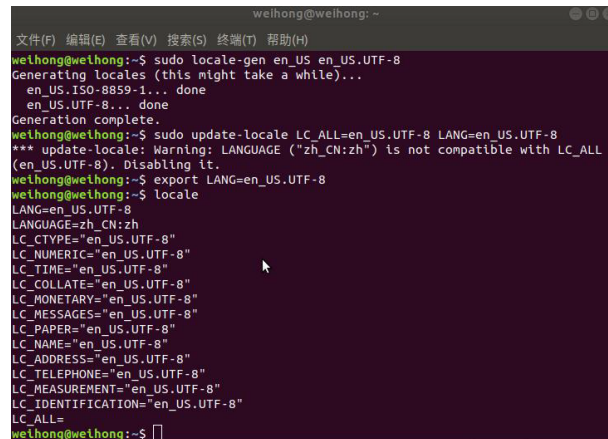
First, ensure that the installation environment supports the UTF-8 format, and use the following command to set:


```

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
locale # verify settings

```

After executing, the terminal displays:



```

weihong@weihong:~$ sudo locale-gen en_US en_US.UTF-8
Generating locales (this might take a while)...
  en_US.ISO-8859-1... done
  en_US.UTF-8... done
Generation complete.
weihong@weihong:~$ sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
*** update-locale: Warning: LANGUAGE ("zh_CN:zh") is not compatible with LC_ALL
(en_US.UTF-8). Disabling it.
weihong@weihong:~$ export LANG=en_US.UTF-8
weihong@weihong:~$ locale
LANG=en_US.UTF-8
LANGUAGE=zh_CN:zh
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
weihong@weihong:~$

```

Fig. 15 Setup locale.

Set up the source by adding the ROS2 apt repository to the system, then enter the command to check the output to ensure that the Ubuntu Universe repository is enabled.

```
apt-cache policy | grep universe
```

The output should be:

```

500 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 Packages
release v=20.04,o=Ubuntu,a=focal,n=focal,l=Ubuntu,c=universe,b=amd64

```

2) Add the ROS2 code repository

Authorize the key using the following command:

```

sudo apt update && sudo apt install curl gnupg lsb-release -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-
archive-keyring.gpg

```

Add the repository to the source list:

```

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) Install ROS2

```

# update the system
sudo apt update -y
# install ROS2 Desktop, including ROS, Rviz, demos, and tutorials
sudo apt install ros-galactic-desktop -y

```

4) Install python3-related tools

```

sudo apt install -y python3-pip
pip3 install -U argcomplete
sudo apt install -y python3-colcon-common-extensions python3-flake8 python3-pytest-cov python3-rosdep python3-
setuptools python3-vcstool

```

5) Install dependencies and ROS tools

```
sudo apt update -y && sudo apt install -y build-essential cmake git wget
```

Install the test package using pip3. Execute the following command:

```

python3 -m pip install -U flake8-blind-except flake8-builtins flake8-class-newline flake8-comprehensions
flake8-deprecated flake8-docstrings flake8-import-order flake8-quotes pytest-repeat pytest-rerunfailures pytest
setuptools

```

Install dependencies by executing the following command:

```
python3 -m pip install -U importlib-metadata importlib-resources
```

1.5 Set up ROS2 environment

After completing the installation of ROS2, if ROS1 is also installed on the system, to avoid having the user execute the source command every time they open a terminal, you can write the relevant commands into the environment configuration file and use the alias command to set them in a bash script. Here are the steps to set up ROS2 environments:

1) *Edit the Bash configuration file*

Open the .bashrc file in your home directory for editing.

```
nano ~/.bashrc
```

2) *Add aliases*

Add the following lines at the end of the .bashrc file to set up aliases for sourcing the ROS2 setup scripts.

```
alias initros2="source /opt/ros/galactic/setup.bash"
```

3) *Source the ROS2 setup scripts*

Source the ROS2 setup scripts once in the current terminal to set up the environments.

```
source ~/.bashrc
```

1.6 Run test

Open two terminals using **ctrl+alt+T** and enter the following commands to initialize the ROS 2 environment on each terminal window:

```
initros2
```

To run the system example nodes for publishing and subscribing to topics in two separate terminals, use the following commands:

```
# terminal 1
ros2 run demo_nodes_cpp talker
# terminal 2
ros2 run demo_nodes_py listener
```

Note the space between 'ros2' and 'run' in the command. After executing the command, the terminal should output information as follows:

In fact, there isn't much difference in the way nodes are run in ROS 1 and ROS 2. The above command can be understood as follows: the package name is **demo_nodes_cpp** and **demo_nodes_py**, the node names are **talker** and **listener**. This can also verify if the C++ and Python APIs are working correctly.

2 Common commands in ROS2

1) *Create ROS2 workspace*

A workspace is a directory that contains ROS2 packages. Create a folder named **wheeltec_robot_ros2** as the workspace:

```
mkdir -p ~/wheeltec_robot_ros2/src
cd ~/wheeltec_robot_ros2/src
```

You can directly copy and place your packages in the **src** directory, or you can use the **git clone** command to download open-source code from GitHub.

2) *Create ROS2 package*

In ROS2, Ament is used as the package building system, with colcon as its building tool. There are two types of packages in ROS2: C++ and Python, and their creation instructions differ. The following tutorial is based on creating a C++ package. The syntax for creating a C++ package is:

```
ros2 pkg create --build-type ament_cmake <package_name>
```

An example of creating wheeltec_tutorial package:

```
# navigate to the 'src' folder in the workspace
cd ~/wheeltec_robot_ros2/src
# create a package named 'wheeltec_tutorial'
ros2 pkg create --build-type ament_cmake wheeltec_tutorial
```

After returning to the parent directory, enter the following command to compile:

```
# compile all packages in the 'src' directory
colcon build
# compile select package
colcon build --packages-select wheeltec_tutorial
```

Then return robot_ros2 directory and run following command:

```
source install/setup.bash
```

The package.xml file contains metadata about the package, while the CMakeLists.txt file describes how to generate code within the package.

3) Create ROS2 message file

To create a custom message type in ROS2, you first need to create a new package:

```
cd wheeltec_robot_ros2/src
ros2 pkg create --build-type ament_cmake wheeltec_interfaces
```

wheeltec_interfaces is the name of the package. Currently, it is not possible to generate a .msg or .srv file in a pure Python package. Create msg and srv directory in the package:

```
cd ~/wheeltec_robot_ros2/src/wheeltec_interfaces && mkdir msg
cd ~/wheeltec_robot_ros2/src/wheeltec_interfaces && mkdir srv
```

Enter the msg directory and create a new file named Num.msg, with the first letter capitalized.

```
cd msg
nano Num.msg
```

Edit the Num.msg file and enter the following on the first line:

```
int64 num
```

Enter the srv directory and create a new file named ThreeInts.srv, with the first letter capitalized.

```
cd srv
nano ThreeInts.srv
```

Edit the ThreeInts.srv file and enter the following content:

```
int64 a
int64 b
int64 c
---
int64 sum
```

The above is a custom service that requests three integers named a, b, and c, and responds with an integer named sum. After creating the files, the package tree looks like this:

Modify the CMakeLists.txt file of the package and add the following content:

```
find_package(rosidl_default_generators REQUIRED)
rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/Num.msg"
  "srv/ThreeInts.srv"
)
```

Modify the package.xml file to declare dependencies. Add the following content:

```
<build_depend>rosidl_default_generators</build_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<member_of_group>rosidl_interface_packages</member_of_group>
```

After making the modifications, return to the workspace directory and start the compilation:

```
colcon build --packages-select wheeltec_interfaces
```

Enter the following command to get the executable files of the workspace:

```
source install/setup.bash
```

In the terminal, enter the command to view the available system messages:

```
ros2 interface list -m
```

4) *Compiling ROS2 Packages*

The compilation and building tool in ROS2 has changed to **colcon**, and the workspace structure is different from ROS1. In ROS 2, there is no **devel** folder in the workspace. The command to build packages in a ROS 2 workspace is:

```
colcon build
```

colcon does not use source code to build. After compilation, the generated folders in the workspace are as shown in the figure below:

build folder: Location for storing intermediate files.

install folder: Location for each package.

log folder: Location for all available log information.

src folder: Location for placing source code.

3 Convert *.vdi to *.img and *.img to *.vdi

To created VM (*.vdi format) needs to be converted into *.img file to flash to Jetson Nano.

Open command line on Windows and navigate to directory “C:\Program Files\Oracle\VirtualBox”

```
cd "C:\Program Files\Oracle\VirtualBox"
```

Convert *.vdi to *.img.

```
VBoxManage clonehd --format RAW input.vdi output.img
```

(Option) Convert *.img to *.vdi.

```
VBoxManage convertdd input.img output.vdi --format VDI
```