

ROS2 Brief Introduction

- Features

 - API

- Language

 - CPP

- Node

- Launch

- Communication

- Differences in specific use

ROS2 foxy installation

- Install ROS 2 package

- Environment configuration

 - Configuration Environment

- Install Gazebo

- Compile

- Software package configuration

ROS2 Brief Introduction

ROS2 is a huge update of ROS1.

ROS1 was born in 2007. With the support of an active open source community, its functions are constantly enriched and the number of codes continues to grow. However, its overall design is not very scientific, lacks security, real-time, and robustness, and is not very suitable for industry and specific industry applications.









The ROS leadership team is aware of this problem, but ROS1 has been hard to come back from. Some important underlying modifications will make ROS1 more unstable, and will inevitably encounter a large number of ROS1 package code compatibility issues. Rather than patching things up, it's better to recreate a more scientific and stable ROS2.

Currently, Noetic of ROS1 will stop supporting it in 2025. For the large existing code base of ROS1, it is feasible for existing projects to remain in ROS1, but migrating to ROS2 in the future is undoubtedly inevitable, so it is necessary to learn to migrate and use ROS2 now.

Like ROS1, ROS2 is released once a year. Since Humble and Ubuntu 22.04 are not stable enough after testing, Galactic will stop supporting it, so the foxy version + ubuntu20.04 is selected. At present, the core functions of ROS2 have been improved, and some third-party packages are still being adapted.

List of Distributions

Below is a list of current and historic ROS 2 distributions. Rows in the table marked in green are the currently supported distributions.

Distro	Release date	Logo	EOL date
Iron Irwini	May 23rd, 2023	TBD	November 2024
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		November 2022
Foxy Fitzroy	June 5th, 2020		May 2023
Eloquent Elusor	November 22nd, 2019		November 2020
Dashing Diademata	May 31st, 2019		May 2021
Crystal Clemmys	December 14th, 2018		December 2019
Bouncy Bolson	July 2nd, 2018		July 2019
Ardent Apalone	December 8th, 2017		December 2018
beta3	September 13th, 2017		December 2017
beta2	July 5th, 2017		September 2017
beta1	December 19th, 2016		Jul 2017
alpha1 - alpha8	August 31th, 2015		December 2016

Features

The update of ROS2 is comprehensive, so here I will only briefly describe the situation that the author currently understands.

API

The bottom layer of ROS2 is written in C language, and its name is rcl library. The rclcpp and rclpy called when writing the upper-layer program are packaged on the rcl library. The advantage of this is that the calling API of cpp and python programs is more convenient. They are unified and similar. When updating the functions of ros2 at the same time, directly update the rcl library and add support for cpp and python.

Language

CPP

The cpp 11, cpp1A4, and cpp17 standards are used by default, which are more modern than the cpp98 standard used by ROS1 by default. Support more security and efficiency features.

###python

Completely abandon python2 and fully embrace python3.

Node

For a more standardized Node writing standard, you must create a class that inherits from the Node object (for example: rclcpp::Node in cpp, rclpy.node.Node in Python). Convenient for team development.

Launch

Different from ROS1's extensive xml format launch file, ROS2 recommends using python files to write launch files, providing more flexibility in startup.

Communication

ROS2 no longer has the master node in ROS1. The child nodes in ROS1 need to register and communicate with the master node before they can communicate with each other. It is a centralized architecture. ROS2 uses a distributed node method, and nodes can discover each other without any master node. This avoids the collapse of the entire system due to the crash of the master node, and at the same time makes ROS2 more flexible in multi-machine distributed deployment.

At the same time, service requests in ROS1 are blocked by Block, and the client will be in stop the world (stuck) state before getting a response.

The service design of ROS2 is Asyn asynchronous, and the client will not be stuck in receiving the server response. But you can also use blocking mode if you want.

Differences in specific use

Only common parts are listed

ROS1	ROS2	introduction
catkin_make	colcon build	compile
roslaunch	ros2 launch	launch
rostopic list	ros2 topic list	
rostopic echo	ros2 topic echo	
roslaunch	ros2 run	
roslaunch rqt_graph rqt_graph	rqt_graph	
roslaunch rqt_tf_tree rqt_tf_tree	ros2 run tf2_tools view_frames.py	ros2 will save a pdf file in the terminal path and cannot be viewed directly
rviz	rviz2	

ROS2 foxy installation

Note: To download the humble version, you need to change foxy to humble

The installation process can be found on the official website: <https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html>

Install ROS 2 package

Update software libraries

```
sudo apt update
```

ROS 2 packages are built on frequently updated Ubuntu systems. Before installing new packages, it is always recommended to ensure that your system is up to date.

```
sudo apt upgrade
```

It is recommended to install the full version of ROS including ROS base rviz tutorial and other software.

```
sudo apt install ros-foxy-desktop
```

- If you only need the ROS core package

```
sudo apt install ros-foxy-ros-base
```

Packages that may be needed for subsequent development:

```
sudo apt install ros-foxy-turtlesim
sudo apt install ros-foxy-xacro
sudo apt install python3-pip
sudo apt install python3-colcon-common-extensions
sudo apt install python3-vcstool
sudo apt-get install ros-foxy-joint-state-publisher-gui
```

Environment configuration

Configuration Environment

Execute the following command to configure the terminal environment. This command must be executed every time you open the terminal.

```
source /opt/ros/foxy/setup.bash
```

Or use once and for all, configure automatically every time you open the terminal

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

Install Gazebo

gazebo is a powerful simulation platform provided by ROS

Just install Gazebo 11 and related ROS support packages:

```
sudo apt install gazebo11 ros-foxy-gazebo-ros-pkgs
```

For navigation, the following packages can be installed

```
sudo apt install ros-foxy-cartographer
sudo apt install ros-foxy-cartographer-ros
sudo apt install ros-foxy-navigation2
sudo apt install ros-foxy-nav2-bringup
sudo apt install ros-foxy-gazebo-ros2-control
sudo apt install ros-foxy-ros2-control ros-foxy-ros2-controllers
```

Compile

Enter the source package

```
cd ~/yahboomcar_ws
colcon build --symlink-install
echo 'source ~/yahboomcar_ws/install/setup.bash' >> ~/.bashrc
```

colcon provides many parameter options

Commonly used parameters:

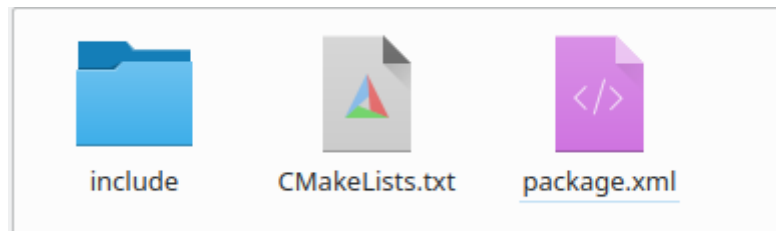
1. `--symlink-install`: Use symbolic links instead of copying files, such as Taking the dynamic link library as an example, a symbolic link will be used in the install directory to point to the library file (such as *.so) generated in the build directory. Without this option, both directories will have the library file.

2. --packages-select: Only compile specified packages, such as
colcon build --packages-select autoware_map_msgs vector_map_msgs
3. --packages-ignore: Ignore the specified package, same as above
4. --continue-on-error: Continue to compile other modules after a compilation error
5. --cmake-args, --ament-cmake-args, --catkin-cmake-args: pass parameters to the corresponding package

Different from ROS1, after modifying any files in ROS2, please recompile colcon build

Software package configuration

Visible in the created software package



Modify the package name in project() of CMakeLists.txt and the name of package.xml to modify the name of the software package.

At the same time, in order to use the files in the folder in the software package, you need to install it in cmakefile.

