2. ROS2 Common commands and tools

1. Package management tool ros2 pkg

1.1. ros2 pkg create

Function: Create a function package. Specify the package name, compilation mode, and dependencies when creating a function package.

Command format: ros2 pkg create --build-type ament_python pkg_name rclpy std_msgs sensor_msgs

ros2 pkg create: indicates the command to create the package

--build-type: Write ament_cmake if the newly created feature package uses C++ or C, and ament_python if it uses Python

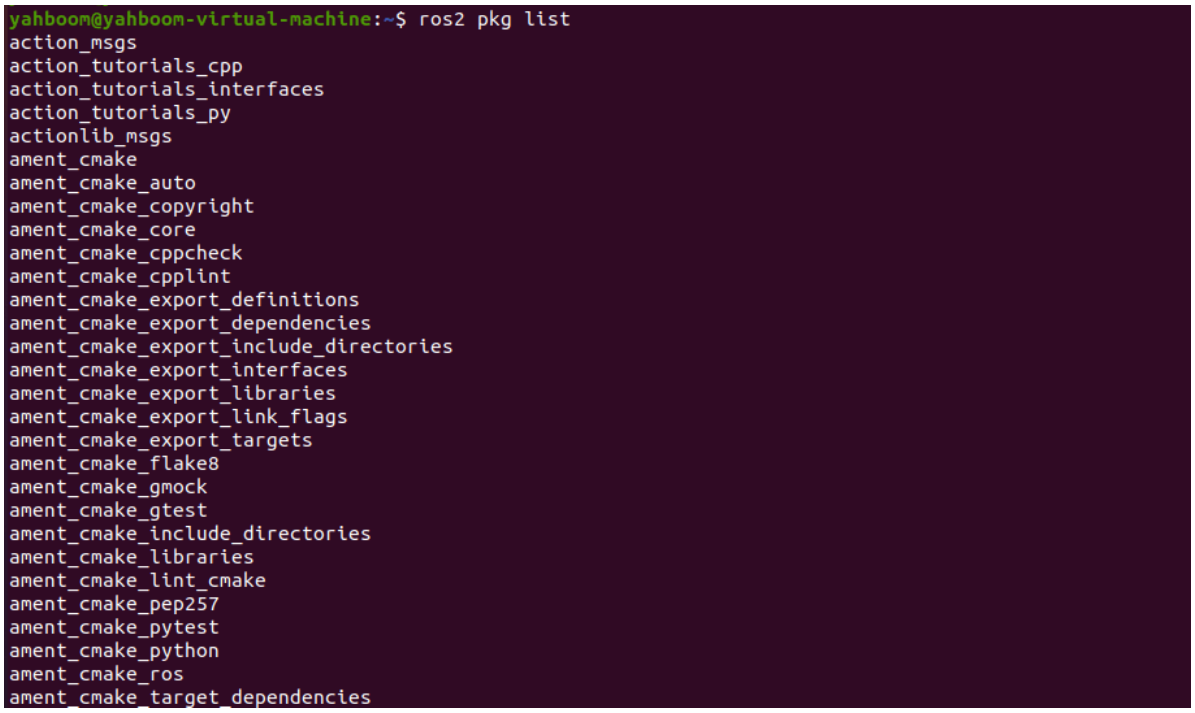
pkg_name: specifies the name of the function package

rclpy std_msgs sensor_msgs: These are compilation dependencies

1.2. ros2 pkg list

Function: View the function pack list in the system

The command format is ros2 pkg list

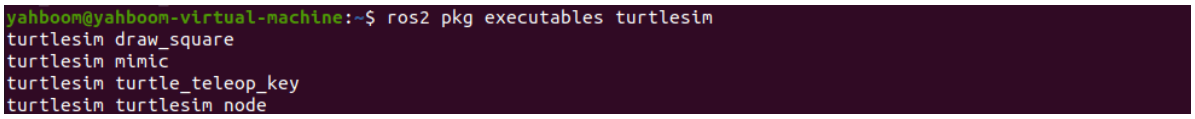


```
yahboom@yahboom-virtual-machine:~$ ros2 pkg list
action_msgs
action_tutorials_cpp
action_tutorials_interfaces
action_tutorials_py
actionlib_msgs
ament_cmake
ament_cmake_auto
ament_cmake_copyright
ament_cmake_core
ament_cmake_cppcheck
ament_cmake_cpplint
ament_cmake_export_definitions
ament_cmake_export_dependencies
ament_cmake_export_include_directories
ament_cmake_export_interfaces
ament_cmake_export_libraries
ament_cmake_export_link_flags
ament_cmake_export_targets
ament_cmake_flake8
ament_cmake_gmock
ament_cmake_gtest
ament_cmake_include_directories
ament_cmake_libraries
ament_cmake_lint_cmake
ament_cmake_pep257
ament_cmake_pytest
ament_cmake_python
ament_cmake_ros
ament_cmake_target_dependencies
```

1.3. ros2 pkg executeables

Command Function: View the list of executable files in the package

The command format is ros2 pkg executables pkg_name



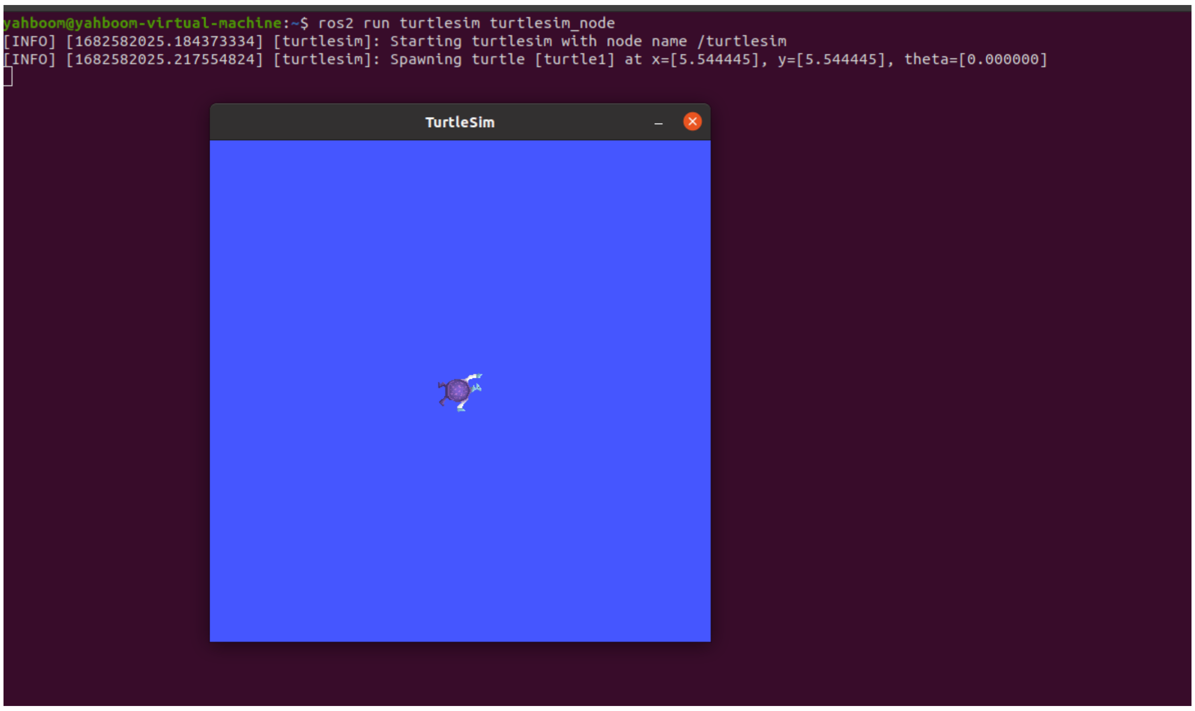
```
yahboom@yahboom-virtual-machine:~$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

2. The node runs ros2 run

Command function: Run the function package node program

The command format is ros2 run pkg_name node_name

- pkg_name: indicates the name of the function package
- node_name: indicates the name of the executable program

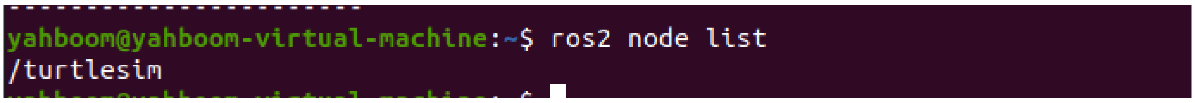


```
yahboom@yahboom-virtual-machine:~$ ros2 run turtlesim turtlesim_node
[INFO] [1682582025.184373334] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1682582025.217554824] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

3. node related tool ros2 node

3.1. ros2 node list

Command function: List all node names in the current domain
Command format: ros2 node list

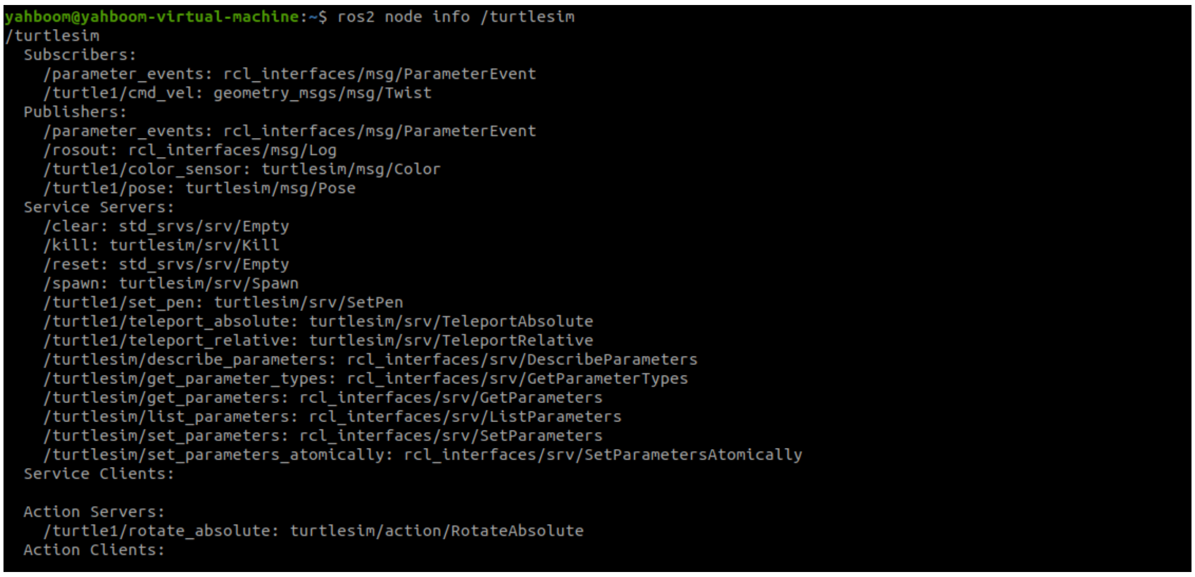


```
yahboom@yahboom-virtual-machine:~$ ros2 node list
/turtlesim
```

3.2. ros2 node info

Command function: View node details, including subscriptions, published messages, opened services and actions
Command format: ros2 node info node_name

- node_name： Specifies the name of the node to be queried



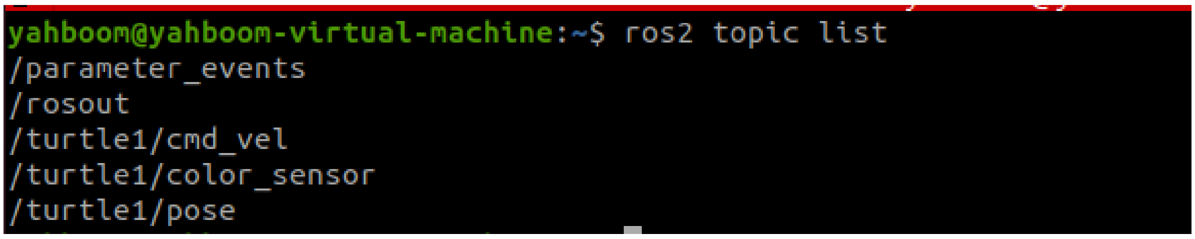
```
yahboom@yahboom-virtual-machine:~$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

4. topic related tool ros2 topic

4.1. ros2 topic list

Command Function: Lists all topics in the current domain

Command format: ros2 topic list



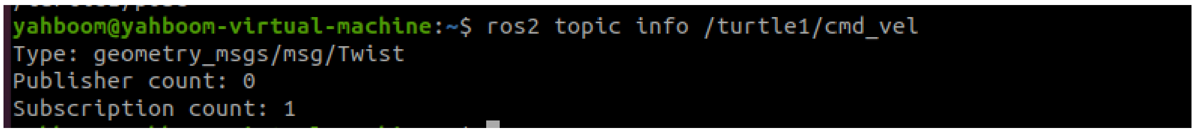
```
yahboom@yahboom-virtual-machine:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

4.2. ros2 topic info

Command function: Display topic message type, number of subscribers/publishers

Command format: ros2 topic info topic_name

- topic_name: specifies the name of the topic to be queried

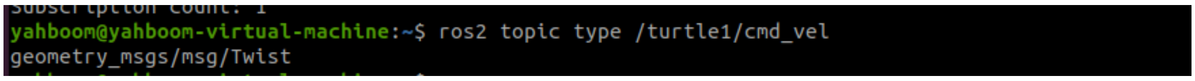


```
yahboom@yahboom-virtual-machine:~$ ros2 topic info /turtle1/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 0
Subscription count: 1
```

4.3. ros2 topic type

Command function: View the message type of the topic
Command format: ros2 topic type topic_name

- topic_name: specifies the name of the topic to query

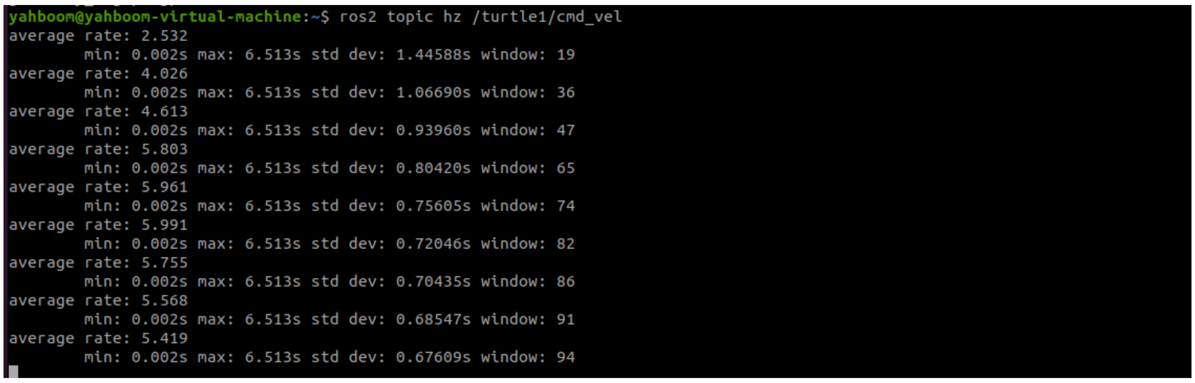


```
yahboom@yahboom-virtual-machine:~$ ros2 topic type /turtle1/cmd_vel
geometry_msgs/msg/Twist
```

4.4. ros2 topic hz

Command function: Show the average publication frequency of topics
Command format: ros2 topic hz topic_name

- topic_name: specifies the name of the topic frequency



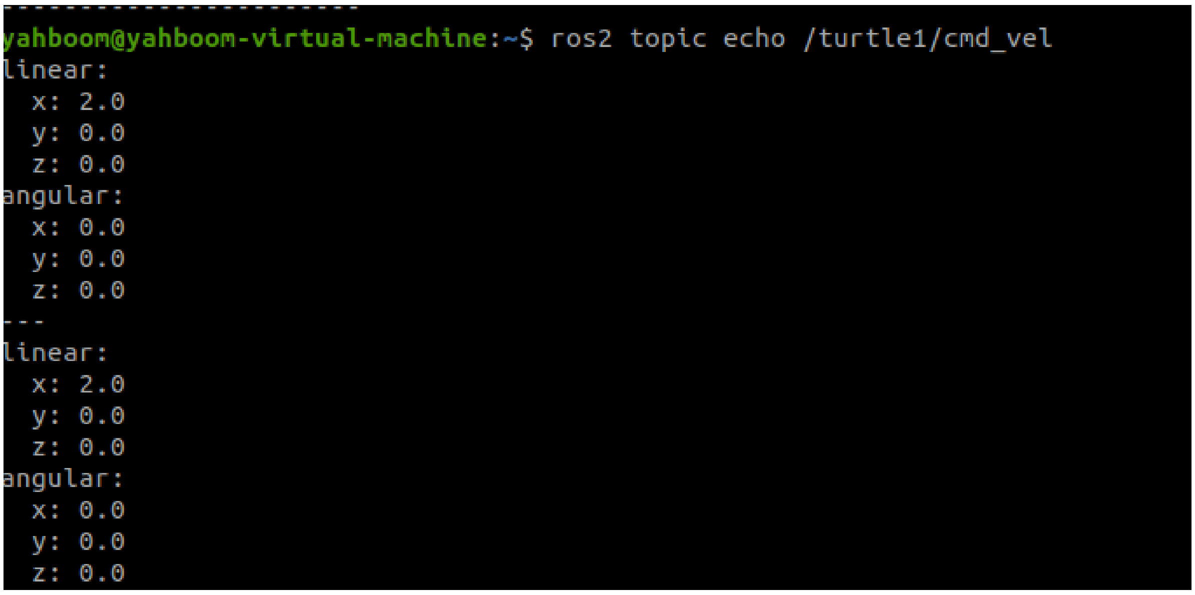
```
yahboom@yahboom-virtual-machine:~$ ros2 topic hz /turtle1/cmd_vel
average rate: 2.532
        min: 0.002s max: 6.513s std dev: 1.44588s window: 19
average rate: 4.026
        min: 0.002s max: 6.513s std dev: 1.06690s window: 36
average rate: 4.613
        min: 0.002s max: 6.513s std dev: 0.93960s window: 47
average rate: 5.803
        min: 0.002s max: 6.513s std dev: 0.80420s window: 65
average rate: 5.961
        min: 0.002s max: 6.513s std dev: 0.75605s window: 74
average rate: 5.991
        min: 0.002s max: 6.513s std dev: 0.72046s window: 82
average rate: 5.755
        min: 0.002s max: 6.513s std dev: 0.70435s window: 86
average rate: 5.568
        min: 0.002s max: 6.513s std dev: 0.68547s window: 91
average rate: 5.419
        min: 0.002s max: 6.513s std dev: 0.67609s window: 94
```

4.5. ros2 topic echo

Command function: Print a subject message on the terminal, similar to a subscriber

Command format: ros2 topic echo topic_name

- topic_name: specifies the name of the topic to be printed



```
yahboom@yahboom-virtual-machine:~$ ros2 topic echo /turtle1/cmd_vel
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
```

4.6. ros2 topic pub

Command function: publish the message of the specified topic in the terminal
Command format: ros2 topic pub topic_name message_type message_content

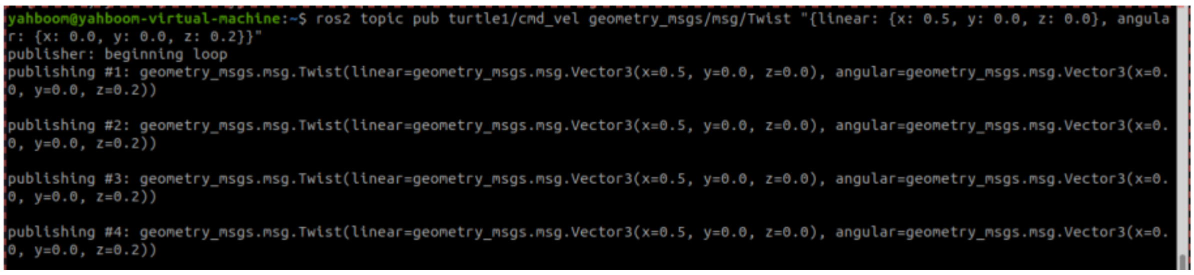
- topic_name: specifies the name of the topic whose topic information you want to publish
- message_type: indicates the topic data type
- message_content: indicates the message content

By default, it is released at the frequency of 1Hz. You can set the following parameters.

- Parameter -1 is published only once. ros2 topic pub-1 topic_name message_type message_content
- Parameter -t count Indicates that the count times the circular advertisement ends. ros2 topic pub -t count topic_name message_type message_content
- Parameter -r count is published at the frequency of count Hz. ros2 topic pub -r count topic_name message_type message_content

```
ros2 topic pub turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.5, y: 0.0,
z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}"
```

The important thing to notice here is that there is a space after the colon.



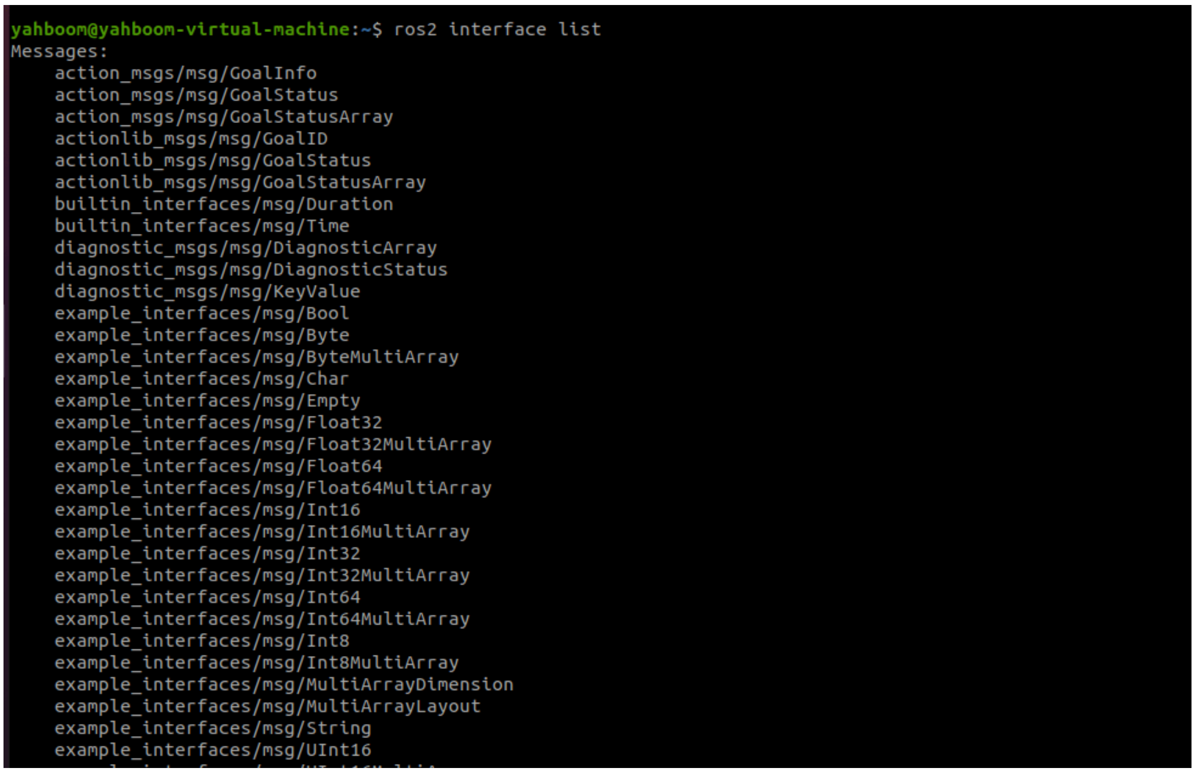
```
yahboom@yahboom-virtual-machine:~$ ros2 topic pub turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.5, y: 0.0, z: 0.0}, angula
r: {x: 0.0, y: 0.0, z: 0.2}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.
0, y=0.0, z=0.2))

publishing #2: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.
0, y=0.0, z=0.2))

publishing #3: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.
0, y=0.0, z=0.2))

publishing #4: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.
0, y=0.0, z=0.2))
```

5. Interface related tool ros2 interface

5.1. ros2 interface list

Command function: Lists all interfaces of the current system, including topics, services, and actions.

Command format: ros2 interface list

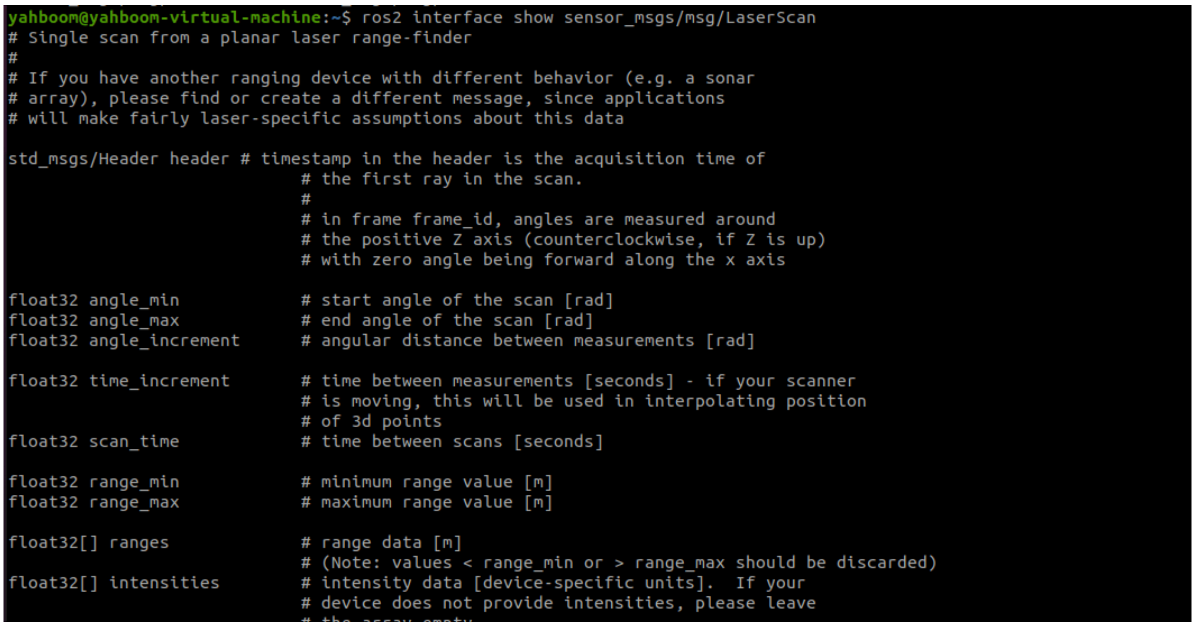


```
yahboom@yahboom-virtual-machine:~$ ros2 interface list
Messages:
    action_msgs/msg/GoalInfo
    action_msgs/msg/GoalStatus
    action_msgs/msg/GoalStatusArray
    actionlib_msgs/msg/GoalID
    actionlib_msgs/msg/GoalStatus
    actionlib_msgs/msg/GoalStatusArray
    builtin_interfaces/msg/Duration
    builtin_interfaces/msg/Time
    diagnostic_msgs/msg/DiagnosticArray
    diagnostic_msgs/msg/DiagnosticStatus
    diagnostic_msgs/msg/KeyValue
    example_interfaces/msg/Bool
    example_interfaces/msg/Byte
    example_interfaces/msg/ByteMultiArray
    example_interfaces/msg/Char
    example_interfaces/msg/Empty
    example_interfaces/msg/Float32
    example_interfaces/msg/Float32MultiArray
    example_interfaces/msg/Float64
    example_interfaces/msg/Float64MultiArray
    example_interfaces/msg/Int16
    example_interfaces/msg/Int16MultiArray
    example_interfaces/msg/Int32
    example_interfaces/msg/Int32MultiArray
    example_interfaces/msg/Int64
    example_interfaces/msg/Int64MultiArray
    example_interfaces/msg/Int8
    example_interfaces/msg/Int8MultiArray
    example_interfaces/msg/MultiArrayDimension
    example_interfaces/msg/MultiArrayLayout
    example_interfaces/msg/String
    example_interfaces/msg/UInt16
```

5.2. ros2 interface show

Command function: Displays the details of the specified interface
Command format: ros2 interface show interface_name

- interface_name: indicates the name of the interface content to be displayed



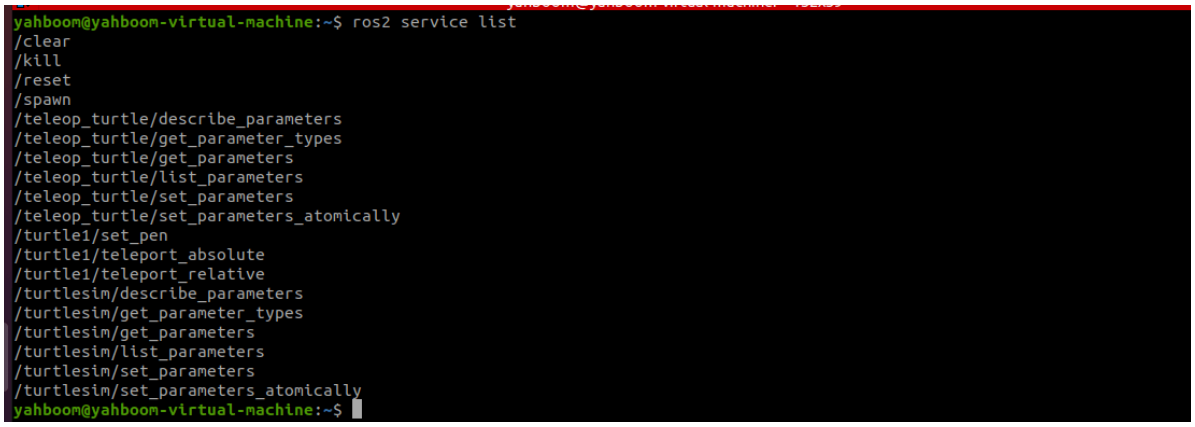
```
yahboom@yahboom-virtual-machine:~$ ros2 interface show sensor_msgs/msg/LaserScan
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

std_msgs/Header header # timestamp in the header is the acquisition time of
                             # the first ray in the scan.
                             #
                             # in frame frame_id, angles are measured around
                             # the positive Z axis (counterclockwise, if Z is up)
                             # with zero angle being forward along the x axis

float32 angle_min            # start angle of the scan [rad]
float32 angle_max            # end angle of the scan [rad]
float32 angle_increment      # angular distance between measurements [rad]

float32 time_increment       # time between measurements [seconds] - if your scanner
                             # is moving, this will be used in interpolating position
                             # of 3d points
float32 scan_time            # time between scans [seconds]

float32 range_min            # minimum range value [m]
float32 range_max            # maximum range value [m]

float32[] ranges             # range data [m]
                             # (Note: values < range_min or > range_max should be discarded)
float32[] intensities        # intensity data [device-specific units].  If your
                             # device does not provide intensities, please leave
```

6. Service related tool ros2 service

6.1. ros2 service list

Command Function: Lists all services in the current domain

Command format: ros2 interface show interface_name



```
yahboom@yahboom-virtual-machine:~$ ros2 service list
/clear
/kill
/reset
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
yahboom@yahboom-virtual-machine:~$
```

6.2. ros2 service call

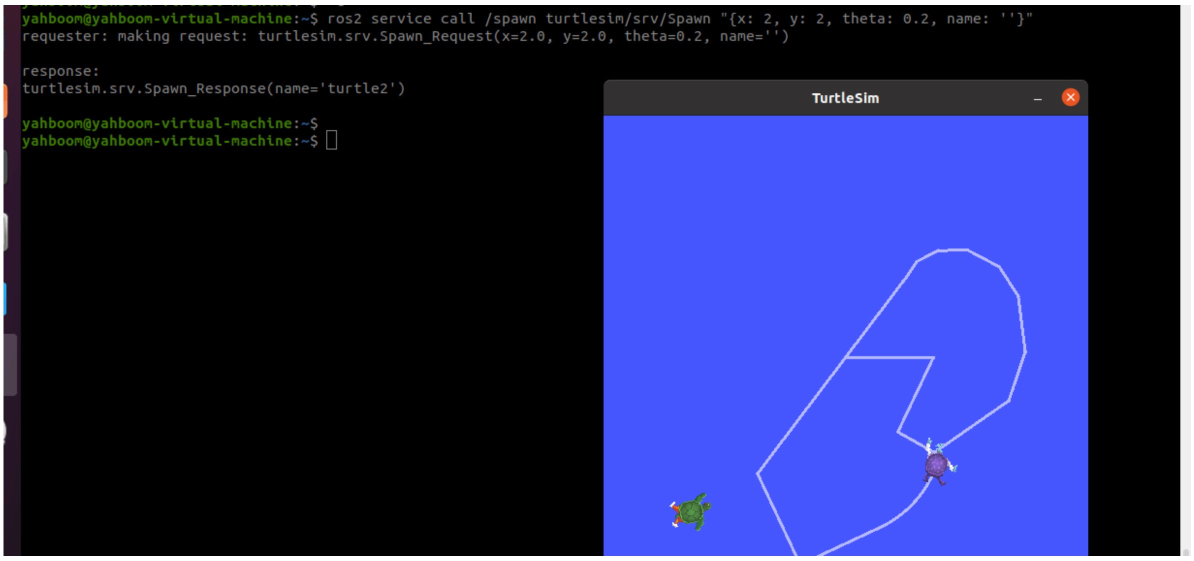
Command function: Invokes the specified service

Command format: ros2 interface call service_name service_Type arguments

- service_name: specifies the service to be invoked
- service_Type: indicates the service data type
- arguments: specifies the parameters required by the service

For example, call the Generate turtle service

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name:
''}"
requester: making request: turtlesim.srv.Spawn_Request(x=2.0, y=2.0, theta=0.2,
name='turtle2')
```



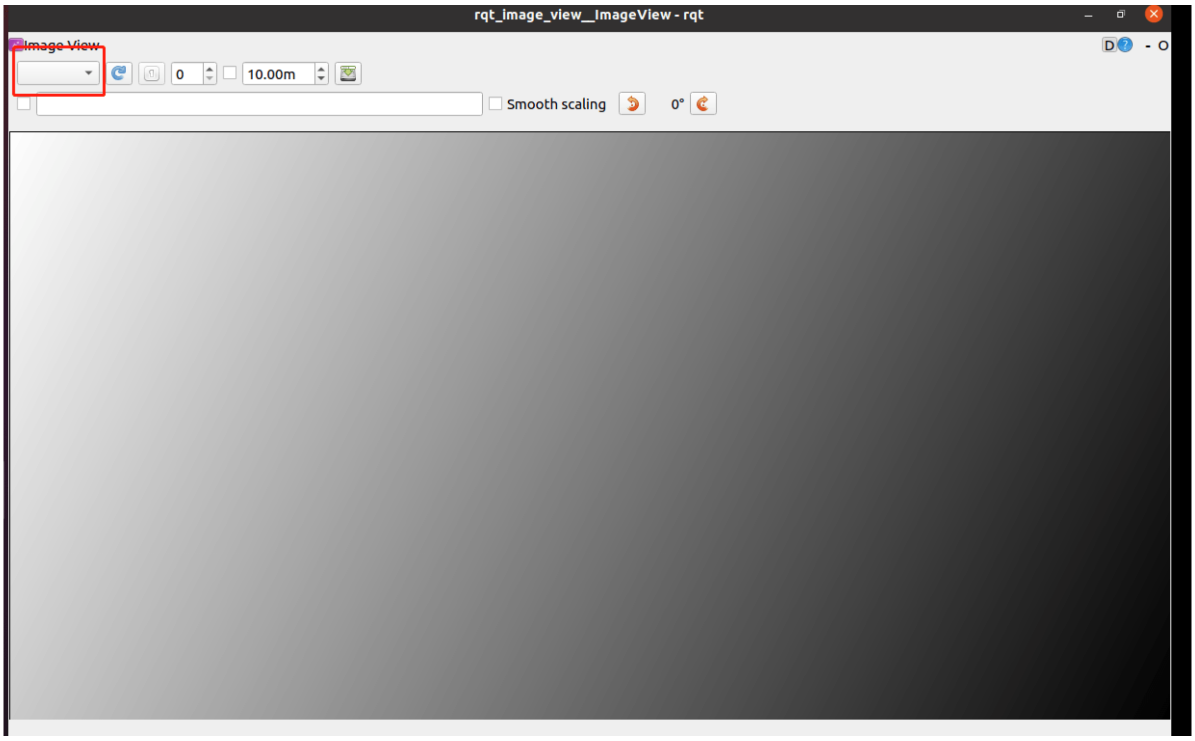
```
yahboom@yahboom-virtual-machine:~$ ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: ''}"
requester: making request: turtlesim.srv.Spawn_Request(x=2.0, y=2.0, theta=0.2, name='')

response:
turtlesim.srv.Spawn_Response(name='turtle2')

yahboom@yahboom-virtual-machine:~$
yahboom@yahboom-virtual-machine:~$
```

7.rqt_image_view

rosrun rqt_image_view rqt_image_view

rqt_image_view can be used to view the image. If the image topic data is published in the current domain, you can use this tool to view the image.

```
ros2 run rqt_image_view rqt_image_view
```

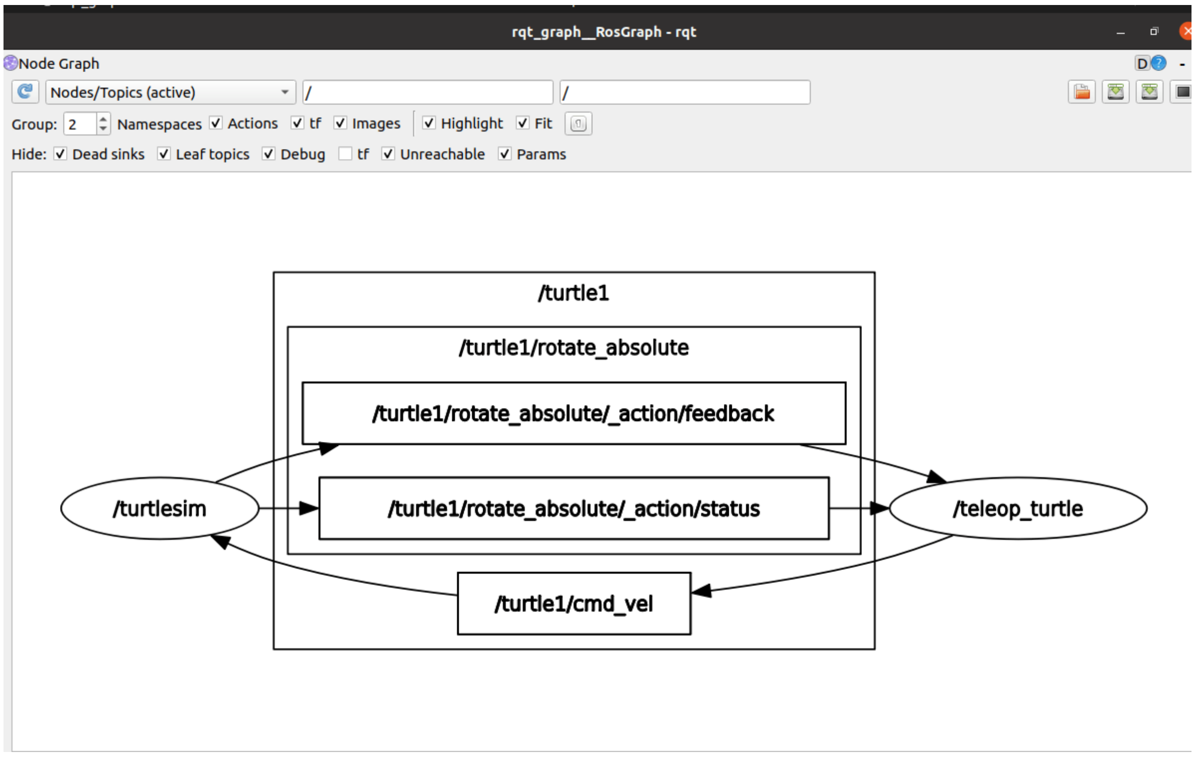


Display the image data through the image topic selected in the upper left corner.

8.rqt_graph

rqt_graph can be used to see which nodes are running in the current domain and the topic communication between nodes, using the following command to enable,

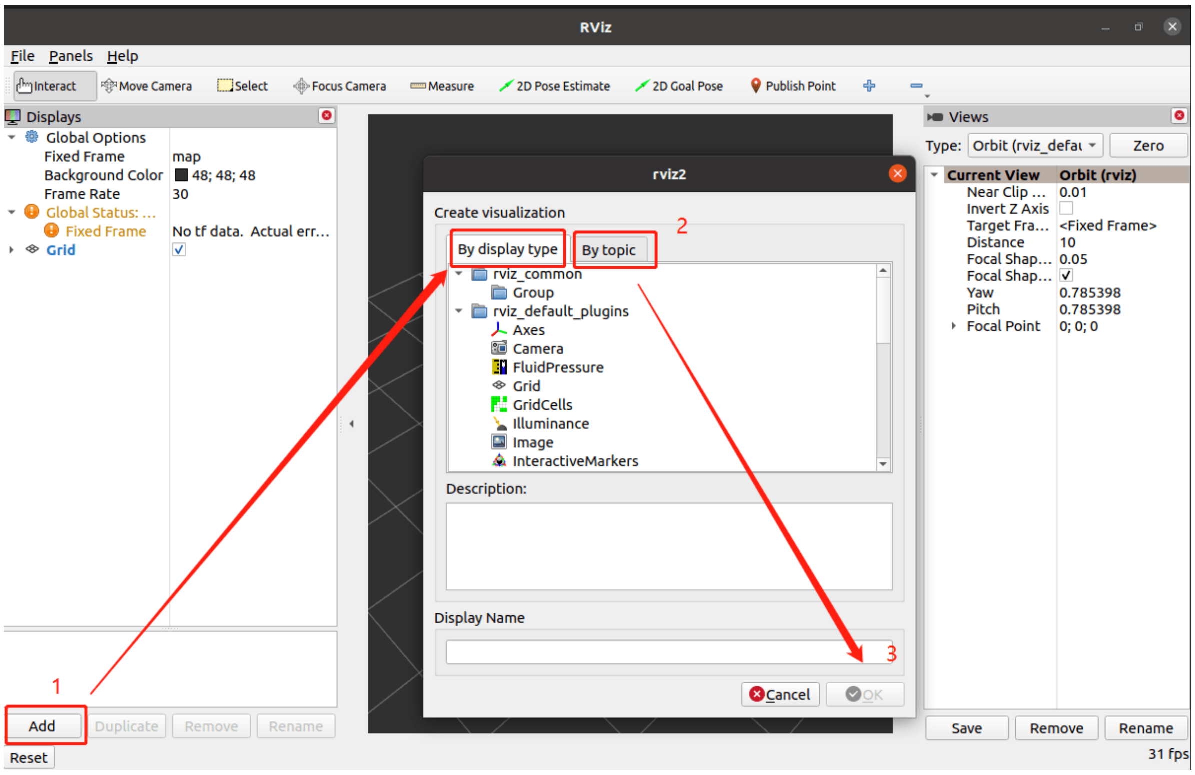
```
ros2 run run rqt_graph rqt_graph
```


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (active)
Group: 2 ✓ Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit
Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug tf ✓ Unreachable ✓ Params
/turtle1
/turtle1/rotate_absolute
/turtle1/rotate_absolute/_action/feedback
/turtlesim
/turtle1/rotate_absolute/_action/status
/teleop_turtle
/turtle1/cmd_vel

9.rviz2

Rviz's core framework is based on the Qt visualization tool to build an open platform, according to the ROS message in the corresponding topic, you can see the graphical effect. In ROS2, use rviz2 to launch the rviz tool.

```
rviz2
```



Through the above steps, you can add visual data through plug-ins or through the topic, the general choice is to add by the topic.

10.tf2_tools

tf2_tools can view the current TF tree and generate a frame.pdf file at the terminal where the command is entered.

```
ros2 run tf2_tools view_frames.py
```