

CSC398 introduction to Autonomous Robots

- Introduction into ROS2 (1) -

Ubbo Visser

Department of Computer Science
College of Arts and Sciences
University of Miami

September 2026

- ▶ ROS 2 architecture & philosophy
- ▶ DDS discovery, nodes, and topics
- ▶ ROS 2 CLI commands
- ▶ ament/colcon workspace and build system
- ▶ Launch-files
- ▶ Gazebo / Isaac Sim



WHAT IS ROS?

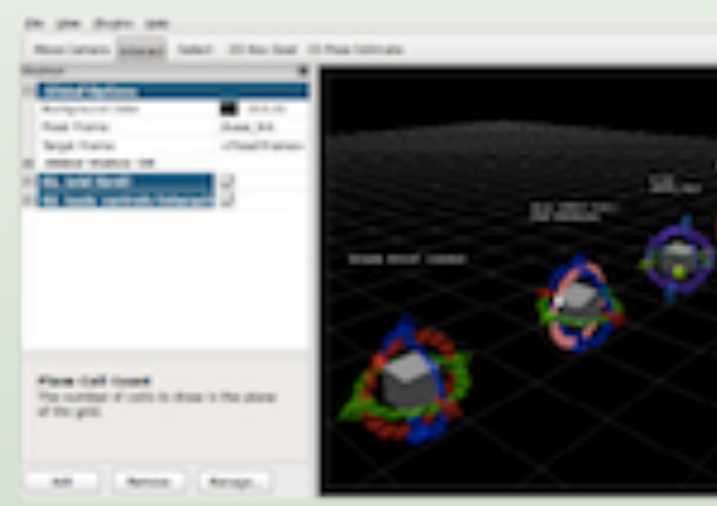
► ROS 2: Robot Operating System 2



=



+



+



+



Plumbing

- Process management
- Inter-process communication
- Device drivers

Tools

- Simulation
- Visualization
- Graphical user interface
- Data logging

Capabilities

- Control
- Planning
- Perception
- Mapping
- Manipulation

Ecosystem

- Package organization
- Software distribution
- Documentation
- Tutorials

- ▶ Originally developed in 2007 at Stanford (AI Lab) and Willow Garage
- ▶ ROS 2 was designed for production robotics: distributed discovery, QoS, security, and real-time use
- ▶ Developed and maintained by the Open Source Robotics community
- ▶ Widely used in research and industry



Source: ros.org

- ▶ Distributed / peer-to-peer

Nodes discover each other through DDS; no central ROS master is required.

- ▶ Standard communication

Topics, services, actions, parameters, and ROS interfaces.

Quality of Service (QoS)

Communication behavior can be tuned for reliability, durability, history, and deadlines.

- ▶ Multi-language

Primary client libraries are rclcpp (C++) and rclpy (Python).

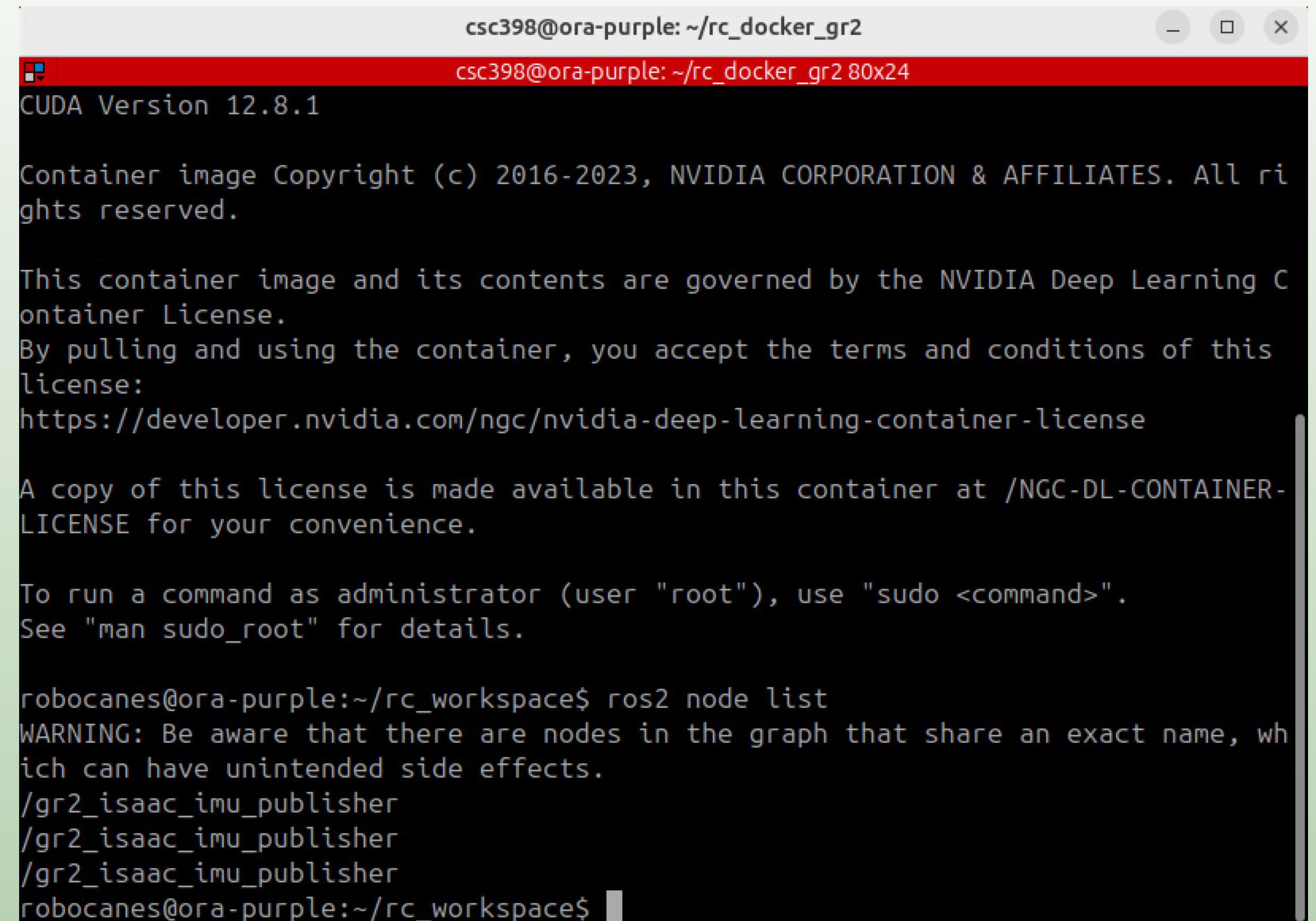
- ▶ Open source

ROS 2 and most ecosystem software are open source.

- ▶ DDS
- ▶ Discovery
- ▶ No ROS core/Master!
- ▶ Better QoS policies

No Master/Core needs to be started

```
~$ros2 node list
```



```
csc398@ora-purple: ~/rc_docker_gr2
csc398@ora-purple: ~/rc_docker_gr2 80x24
CUDA Version 12.8.1

Container image Copyright (c) 2016-2023, NVIDIA CORPORATION & AFFILIATES. All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

A copy of this license is made available in this container at /NGC-DL-CONTAINER-LICENSE for your convenience.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

robocanes@ora-purple:~/rc_workspace$ ros2 node list
WARNING: Be aware that there are nodes in the graph that share an exact name, which can have unintended side effects.
/gr2_isaac_imu_publisher
/gr2_isaac_imu_publisher
/gr2_isaac_imu_publisher
robocanes@ora-purple:~/rc_workspace$
```

- ▶ Executable program with a single purpose
- ▶ Needs to be compiled, can be individually compiled, executed and also managed (e.g. a program showing joint states)
- ▶ Organized in *packages*

Run node with

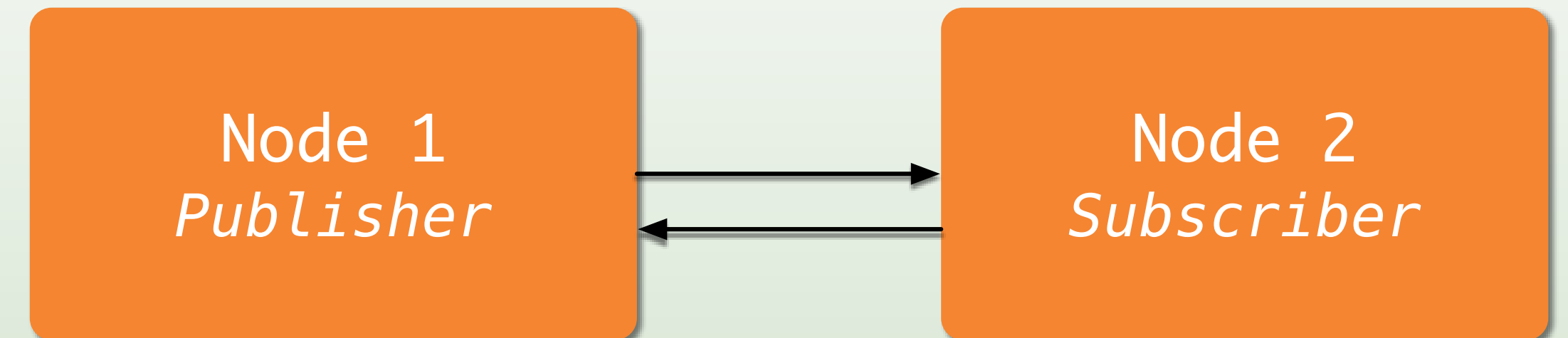
```
~$ros2 run package_name node_name
```

List active nodes

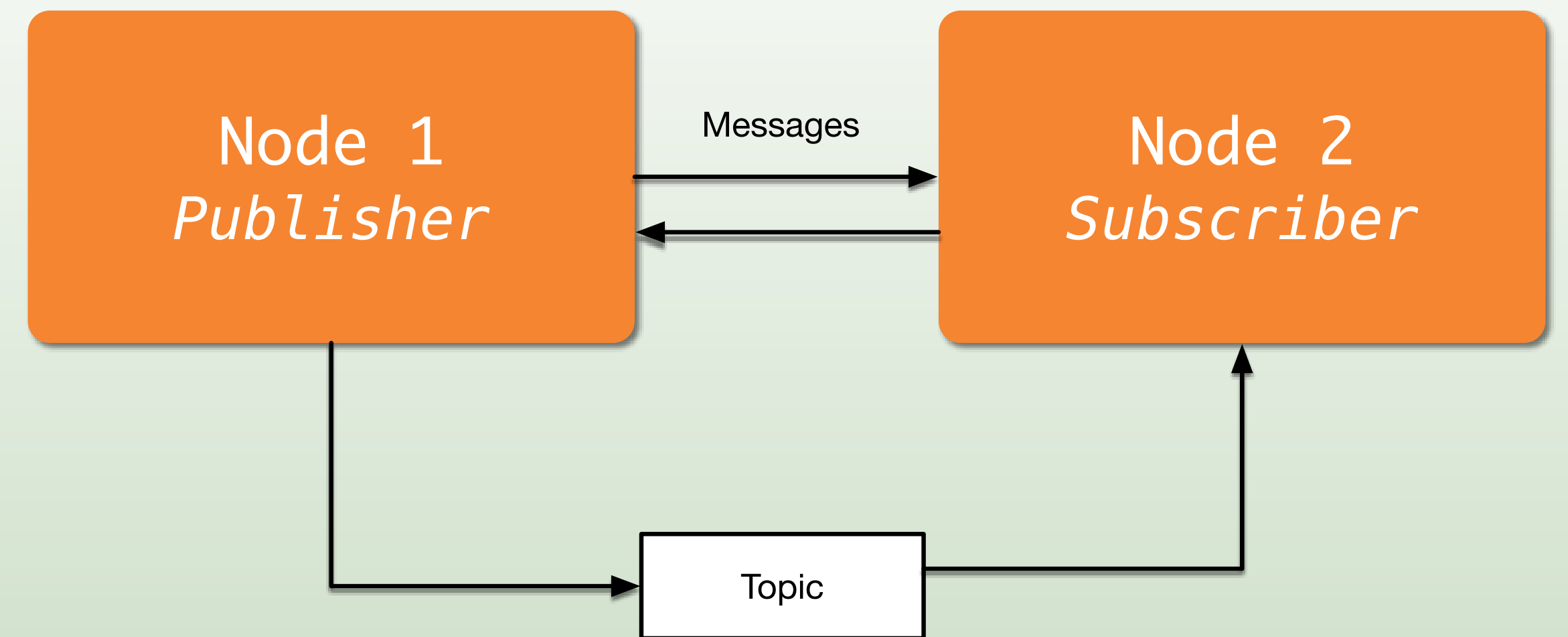
```
~$ros2 node list
```

Node information

```
~$ros2 node info node_name
```



- ▶ Nodes communicate via *topics*
 - ▶ *Publish* or *subscribe* to topics
 - ▶ Common: *1* publisher, *n* subscribers
- ▶ A topic stands for a flow or stream of data, *messages* in ROS language



List active topics

```
~$ros2 topic list
```

Subscribe and print the contents of a topic with

```
~$ros2 topic echo /topic
```

Show information about a topic with

```
~$ros2 topic info /topic
```

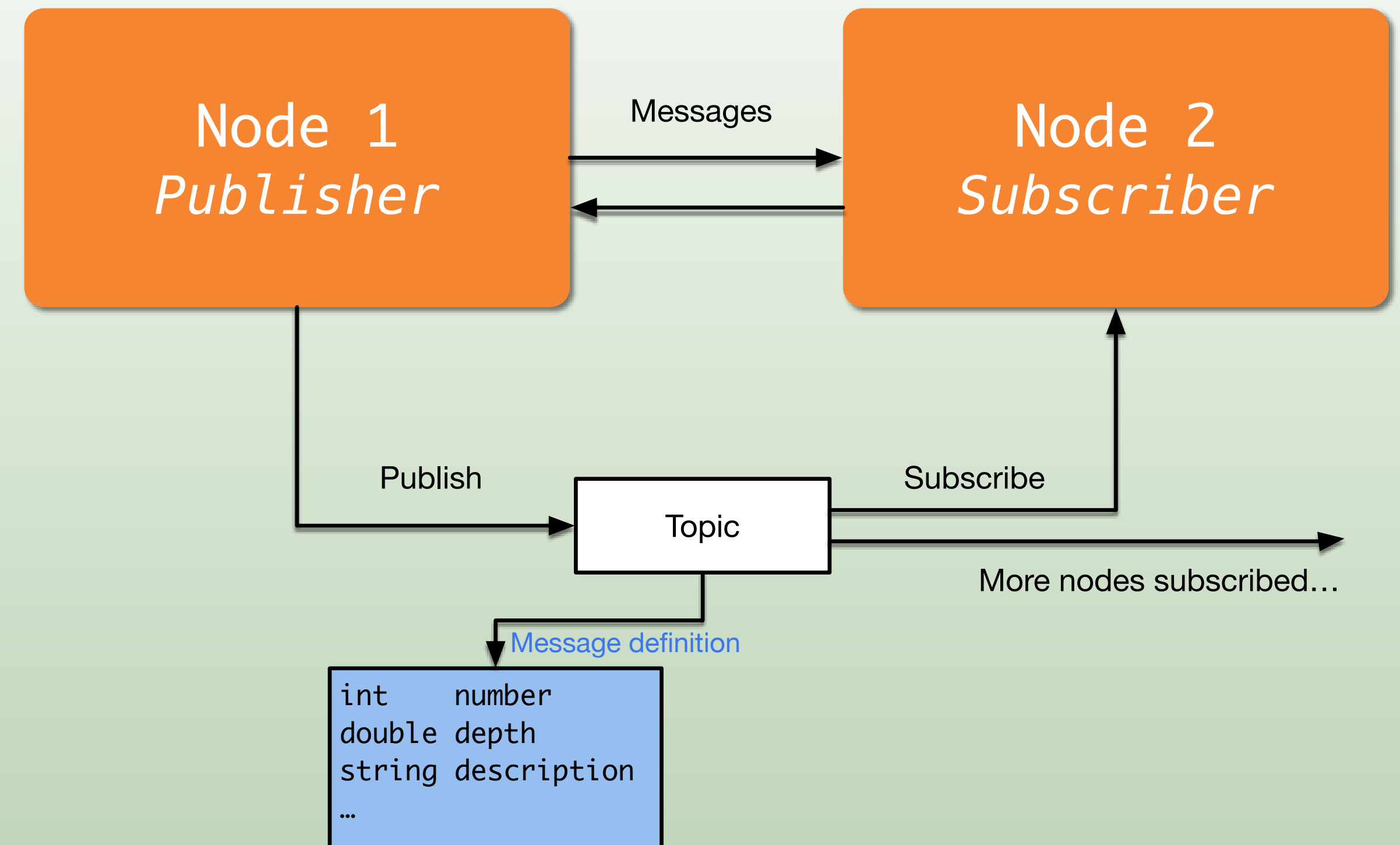
- ▶ Data structure defines *type* of topic
- ▶ Nested structure of primitive data types and arrays
- ▶ Defined in .msg files

List type of topic

```
~$ros2 topic type /topic
```

Publish a message to a topic

```
~$ros2 topic pub /topic pkg/msg/Type "{field: value}"
```



Details at
<http://wiki.ros.org/Messages>

► Pose Stamped Example

geometry_msgs/Point Message

```
# This contains the position of a
point in free space
float64 x
float64 y
float64 z
```

sensor_msgs/Image Message

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
Geometry msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion
orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

EXAMPLE

- ▶ Run and analyse *talker* node

Run talker node

```
~$ros2 run demo_nodes_cpp talker
```

Info about node in another terminal

```
~$ros2 node info /talker
```

Info about topic in a 3rd terminal

```
~$ros2 topic info /chatter
```

```
csc398@ora-purple: ~/rc_docker_gr2
csc398@ora-purple: ~/rc_docker_gr2 80x5
[INFO] [1787834992.521498617] [talker]: Publishing: 'Hello World: 3'
[INFO] [1787834993.521562372] [talker]: Publishing: 'Hello World: 4'
[INFO] [1787834994.521495060] [talker]: Publishing: 'Hello World: 5'
^C[INFO] [1787834994.789666008] [rclcpp]: signal_handler(SIGINT/SIGTERM)
robocanes@ora-purple:~/rc_workspace$

csc398@ora-purple: ~/rc_docker_gr2 80x22
robocanes@ora-purple:~/rc_workspace$ ros2 node info /talker
There are 2 nodes in the graph with the exact name "/talker". You are seeing information about only one of them.
/talker
Subscribers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
  /chatter: std_msgs/msg/String
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
Service Servers:
  /talker/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /talker/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /talker/get_parameters: rcl_interfaces/srv/GetParameters
  /talker/list_parameters: rcl_interfaces/srv/ListParameters
  /talker/set_parameters: rcl_interfaces/srv/SetParameters
  /talker/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:

Action Servers:

Action Clients:

csc398@ora-purple: ~/rc_docker_gr2 80x9
robocanes@ora-purple:~/rc_workspace$ ros2 topic info /chatter
Type: std_msgs/msg/String
Publisher count: 2
Subscription count: 0
robocanes@ora-purple:~/rc_workspace$
```

EXAMPLE

► Analyse *chatter* topic

Type of topic

```
~$ros2 topic type /chatter
```

```
visser@ubuntu: ~ 62x24
visser@ubuntu:~$ rostopic type /chatter
std_msgs/String
```

Message contents

```
~$ros2 topic echo /chatter
```

```
visser@ubuntu: ~ 62x24
---
data: "hello world 4480"
---
data: "hello world 4481"
---
data: "hello world 4482"
---
data: "hello world 4483"
---
data: "hello world 4484"
```

Frequency of publishing

```
~$ros2 topic hz /chatter
```

```
visser@ubuntu: ~ 62x24
min: 0.097s max: 0.102s std dev: 0.00069s window: 220
average rate: 10.000
min: 0.097s max: 0.102s std dev: 0.00069s window: 230
average rate: 10.000
min: 0.097s max: 0.102s std dev: 0.00068s window: 240
average rate: 10.000
min: 0.097s max: 0.102s std dev: 0.00068s window: 250
average rate: 10.000
min: 0.097s max: 0.102s std dev: 0.00068s window: 260
average rate: 10.000
min: 0.097s max: 0.102s std dev: 0.00070s window: 270
```

EXAMPLE

- ▶ Starting new *listener* node

Type of topic

```
~$ros2 run demo_nodes_cpp listener
```

```
visser@ubuntu: ~ 62x24
[ INFO] [1597956202.798404867]: I heard: [hello world 7480]
[ INFO] [1597956202.898415809]: I heard: [hello world 7481]
[ INFO] [1597956202.999082841]: I heard: [hello world 7482]
[ INFO] [1597956203.098911090]: I heard: [hello world 7483]
[ INFO] [1597956203.199337780]: I heard: [hello world 7484]
[ INFO] [1597956203.298707656]: I heard: [hello world 7485]
[ INFO] [1597956203.398969426]: I heard: [hello world 7486]
[ INFO] [1597956203.498160491]: I heard: [hello world 7487]
[ INFO] [1597956203.599366522]: I heard: [hello world 7488]
[ INFO] [1597956203.698226134]: I heard: [hello world 7489]
[ INFO] [1597956203.798259573]: I heard: [hello world 7490]
[ INFO] [1597956203.898956470]: I heard: [hello world 7491]
[ INFO] [1597956203.999509924]: I heard: [hello world 7492]
[ INFO] [1597956204.099068110]: I heard: [hello world 7493]
[ INFO] [1597956204.199388667]: I heard: [hello world 7494]
[ INFO] [1597956204.298301272]: I heard: [hello world 7495]
[ INFO] [1597956204.397761967]: I heard: [hello world 7496]
[ INFO] [1597956204.497986417]: I heard: [hello world 7497]
[ INFO] [1597956204.598001241]: I heard: [hello world 7498]
[ INFO] [1597956204.698816535]: I heard: [hello world 7499]
[ INFO] [1597956204.798441475]: I heard: [hello world 7500]
[ INFO] [1597956204.898351199]: I heard: [hello world 7501]
[ INFO] [1597956204.998361126]: I heard: [hello world 7502]
```

EXAMPLE

- ▶ Starting new *listener* node

Type of topic

```
~$ros2 run demo_nodes_cpp listener
```

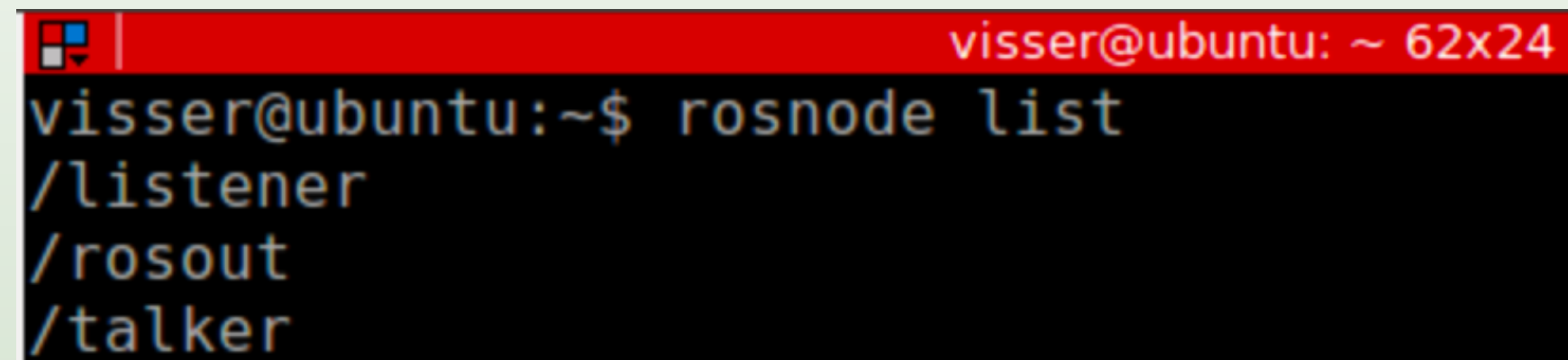
```
csc398@ora-purple: ~/rc_docker_gr2 80x22
[INFO] [1787835347.084881090] [listener]: I heard: [Hello World: 153]
[INFO] [1787835347.563617892] [listener]: I heard: [Hello World: 703]
[INFO] [1787835348.084906081] [listener]: I heard: [Hello World: 154]
[INFO] [1787835348.563851402] [listener]: I heard: [Hello World: 704]
[INFO] [1787835349.084795200] [listener]: I heard: [Hello World: 155]
[INFO] [1787835349.563529245] [listener]: I heard: [Hello World: 705]
[INFO] [1787835350.084751497] [listener]: I heard: [Hello World: 156]
[INFO] [1787835350.563565019] [listener]: I heard: [Hello World: 706]
[INFO] [1787835351.084838129] [listener]: I heard: [Hello World: 157]
[INFO] [1787835351.563750349] [listener]: I heard: [Hello World: 707]
[INFO] [1787835352.084672534] [listener]: I heard: [Hello World: 158]
[INFO] [1787835352.563733882] [listener]: I heard: [Hello World: 708]
[INFO] [1787835353.084755272] [listener]: I heard: [Hello World: 159]
[INFO] [1787835353.563755966] [listener]: I heard: [Hello World: 709]
[INFO] [1787835354.084960568] [listener]: I heard: [Hello World: 160]
[INFO] [1787835354.563822364] [listener]: I heard: [Hello World: 710]
[INFO] [1787835355.085008606] [listener]: I heard: [Hello World: 161]
[INFO] [1787835355.563723373] [listener]: I heard: [Hello World: 711]
[INFO] [1787835356.084899932] [listener]: I heard: [Hello World: 162]
[INFO] [1787835356.563534211] [listener]: I heard: [Hello World: 712]
[INFO] [1787835357.084826154] [listener]: I heard: [Hello World: 163]
```

EXAMPLE

- ▶ Back to Terminal 3: analyse

New listener node visible

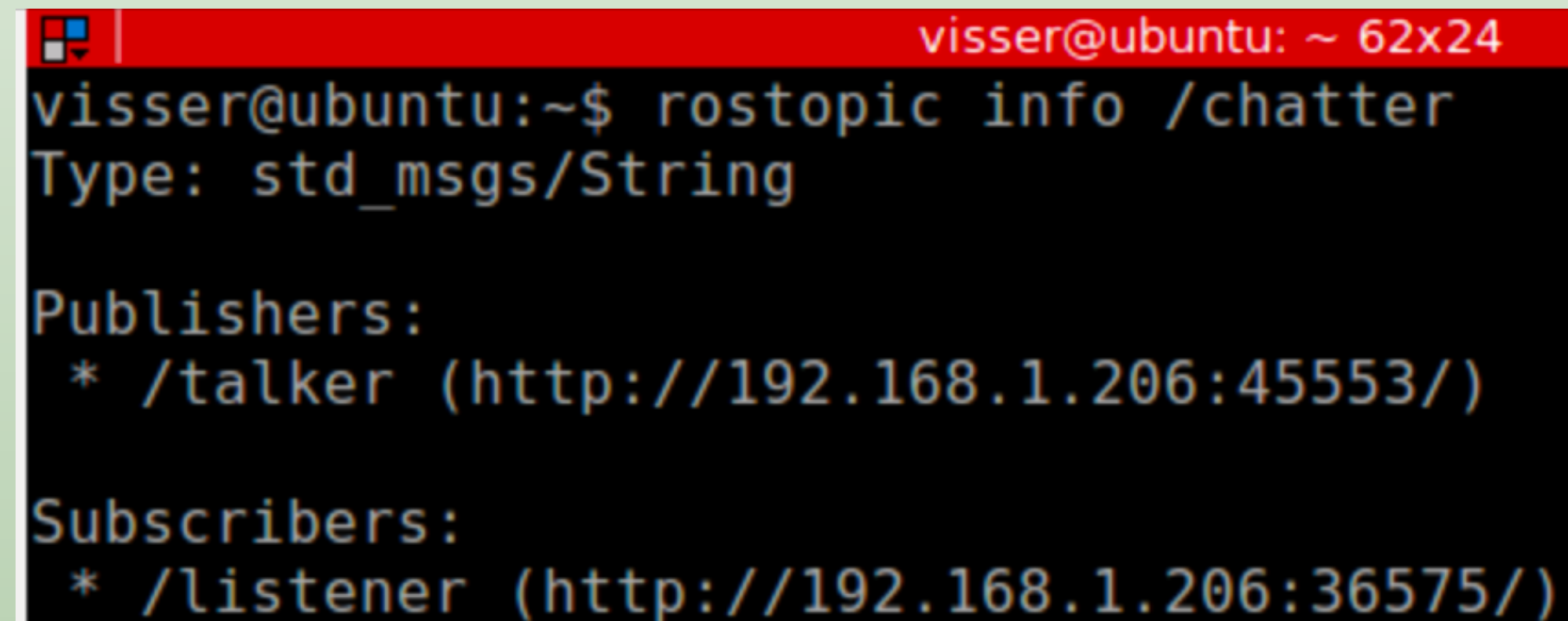
```
~$ros2 node list
```



```
visser@ubuntu: ~ 62x24  
visser@ubuntu:~$ rosnode list  
/listener  
/rosout  
/talker
```

Show the connection of the nodes over the chatter topic with

```
~$ros2 topic info /chatter
```



```
visser@ubuntu: ~ 62x24  
visser@ubuntu:~$ rostopic info /chatter  
Type: std_msgs/String  
  
Publishers:  
* /talker (http://192.168.1.206:45553/)  
  
Subscribers:  
* /listener (http://192.168.1.206:36575/)
```

EXAMPLE

- ▶ Terminal 3: publish own message in terminal

Close talker node in T2, Ctrl-C

Publish own message

```
~$ros2 topic pub /chatter  
std_msgs/String "data: 'RoboCanes  
greet's CSC398 students'"
```

Message shows up in T4 (listener)

```
csc398@ora-purple: ~/rc_docker_gr2 80x22  
[INFO] [1787835599.083876907] [listener]: I heard: [Hello World: 405]  
[INFO] [1787835599.562167799] [listener]: I heard: [Hello World: 955]  
[INFO] [1787835600.083783840] [listener]: I heard: [Hello World: 406]  
[INFO] [1787835600.562629558] [listener]: I heard: [Hello World: 956]  
[INFO] [1787835601.083808003] [listener]: I heard: [Hello World: 407]  
[INFO] [1787835601.562332587] [listener]: I heard: [Hello World: 957]  
[INFO] [1787835602.083776042] [listener]: I heard: [Hello World: 408]  
[INFO] [1787835602.562510474] [listener]: I heard: [Hello World: 958]  
[INFO] [1787835603.083955026] [listener]: I heard: [Hello World: 409]  
[INFO] [1787835603.562660043] [listener]: I heard: [Hello World: 959]  
[INFO] [1787835604.083854432] [listener]: I heard: [Hello World: 410]  
[INFO] [1787835604.562629015] [listener]: I heard: [Hello World: 960]  
[INFO] [1787835605.083928092] [listener]: I heard: [Hello World: 411]  
[INFO] [1787835605.562547531] [listener]: I heard: [Hello World: 961]  
[INFO] [1787835606.083795477] [listener]: I heard: [Hello World: 412]  
[INFO] [1787835606.562541500] [listener]: I heard: [Hello World: 962]  
[INFO] [1787835607.083897952] [listener]: I heard: [Hello World: 413]  
[INFO] [1787835607.562515799] [listener]: I heard: [Hello World: 963]  
[INFO] [1787835608.083847759] [listener]: I heard: [Hello World: 414]  
[INFO] [1787835608.562400360] [listener]: I heard: [Hello World: 964]  
[INFO] [1787835609.083835785] [listener]: I heard: [Hello World: 415]  
[INFO] [1787835609.214909387] [listener]: I heard: [RoboCanes greet's CSC398 stud  
ents]
```

- ▶ Defines context for the current workspace
- ▶ Default workspace loaded with

```
~$source /opt/ros/humble/setup.bash
```

Overlay your catkin workspace with

```
~$cd [ROS2_WS]  
~$source install/setup.bash
```

Check your workspace with

```
~$echo $AMENT_PREFIX_PATH
```

- ▶ *ROS 2 packages use ament as the build system and colcon to build workspaces.*
- ▶ Navigate to your ROS 2 workspace, e.g.:

- ▶ Build the workspace or selected packages

```
~$cd ~/ [ROS2_WS]
```

- ▶ Update your environment after a new package is build

```
~$colcon build --packages-select [package_name]
```

```
~$source install/setup.bash
```

- ▶ The catkin workspace contains the following spaces



src

The *source space* contains the source code. This is where you can clone, create, and edit source code for the packages you want to build.



build

The *build space* is where CMake is invoked to build the packages in the source space. Cache information and other intermediate files are kept here.



install

The *development (install) space* is where built targets are placed (prior to being installed).

Clean entire workspace with

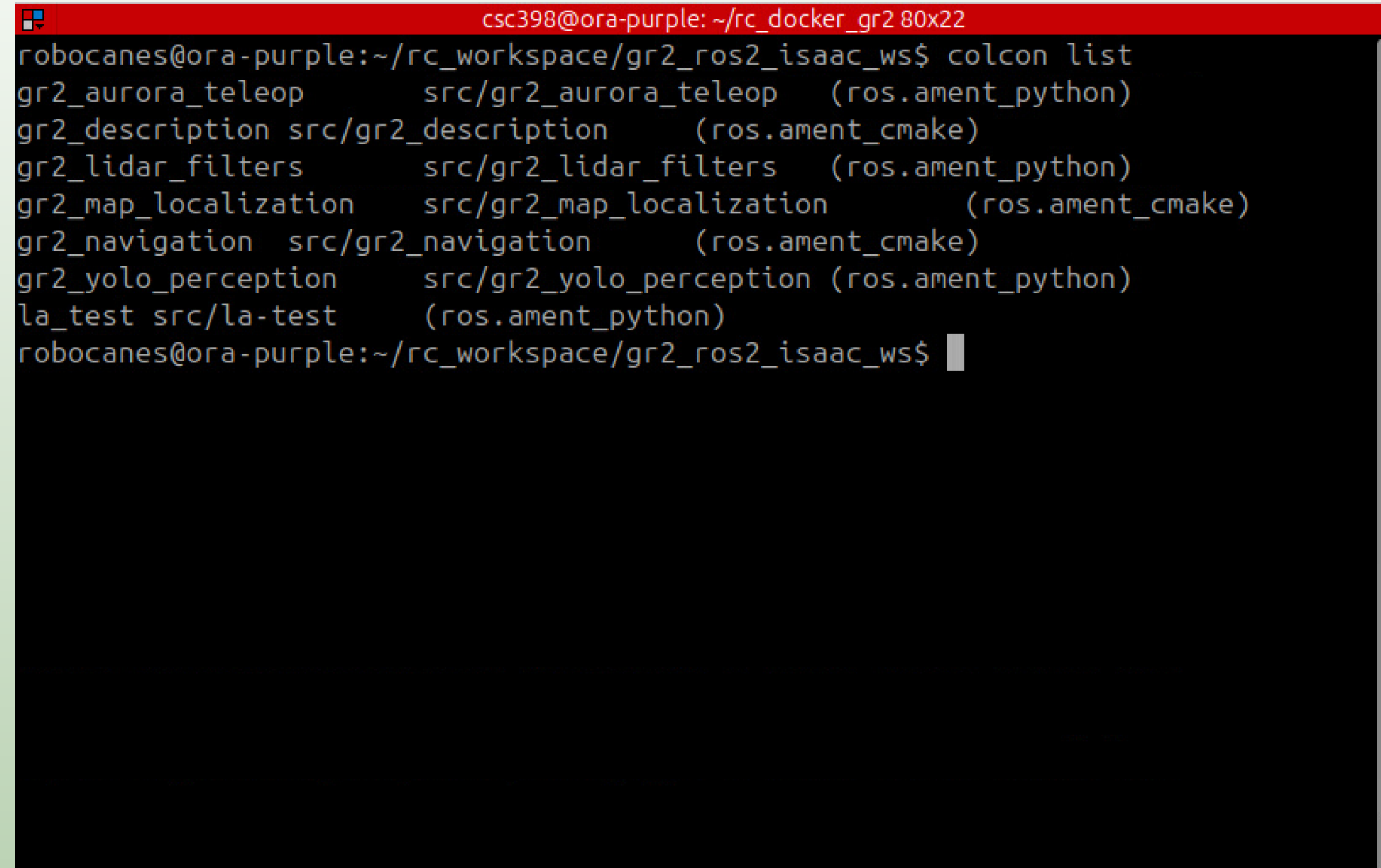
```
~$rm -rf build install log
```

- ▶ colcon provides workspace-wide and package-selective builds.

```
~$colcon list
```

- ▶ Example: set CMake build type to Release

```
~$colcon build --cmake-args  
-DCMAKE_BUILD_TYPE=Release
```

A terminal window with a red title bar containing the text 'csc398@ora-purple: ~/rc_docker_gr2 80x22'. The terminal shows the command 'colcon list' being executed in a directory '~/.rc_workspace/gr2_ros2_isaac_ws'. The output lists several packages with their source directories and build systems: 'gr2_aurora_teleop' (src/gr2_aurora_teleop, ros.ament_python), 'gr2_description' (src/gr2_description, ros.ament_cmake), 'gr2_lidar_filters' (src/gr2_lidar_filters, ros.ament_python), 'gr2_map_localization' (src/gr2_map_localization, ros.ament_cmake), 'gr2_navigation' (src/gr2_navigation, ros.ament_cmake), 'gr2_yolo_perception' (src/gr2_yolo_perception, ros.ament_python), and 'la_test' (src/la_test, ros.ament_python). The prompt then changes to '~/.rc_workspace/gr2_ros2_isaac_ws\$'.

```
csc398@ora-purple: ~/rc_docker_gr2 80x22  
robocanes@ora-purple:~/rc_workspace/gr2_ros2_isaac_ws$ colcon list  
gr2_aurora_teleop      src/gr2_aurora_teleop  (ros.ament_python)  
gr2_description        src/gr2_description    (ros.ament_cmake)  
gr2_lidar_filters      src/gr2_lidar_filters  (ros.ament_python)  
gr2_map_localization   src/gr2_map_localization (ros.ament_cmake)  
gr2_navigation         src/gr2_navigation     (ros.ament_cmake)  
gr2_yolo_perception    src/gr2_yolo_perception (ros.ament_python)  
la_test               src/la_test            (ros.ament_python)  
robocanes@ora-purple:~/rc_workspace/gr2_ros2_isaac_ws$
```

EXAMPLE

- ▶ In your ROS2 workspace

```
~$cd [ROS2_WS]
```

- ▶ Build packages

```
~$colcon build
```

- ▶ Re-source your workspace

```
~$source install/setup.bash
```

- ▶ Launch node

```
~$ros2 launch demo_nodes_cpp talker_listener.launch.py
```

```
visser@ubuntu: ~/hsr 90x14
visser@ubuntu:~/hsr$ catkin_make
Base path: /home/visser/hsr
Source space: /home/visser/hsr/src
Build space: /home/visser/hsr/build
Devel space: /home/visser/hsr/devel
Install space: /home/visser/hsr/install
####
#### Running command: "make cmake_check_build_system" in "/home/visser/hsr/build"
####
####
#### Running command: "make -j10 -l10" in "/home/visser/hsr/build"
####
[ 0%] Built target std_msgs_generate_messages_lisp
[ 0%] Built target std_msgs_generate_messages_cpp
[ 0%] Built target std_msgs_generate_messages_eus
```

```
visser@ubuntu: ~/hsr 90x35
visser@ubuntu:~$ cd hsr
visser@ubuntu:~/hsr$ s
visser@ubuntu:~/hsr$ roslaunch rospy_tutorials talker_listener.launch
... logging to /home/visser/.ros/log/f1a525ac-69ff-11ef-bc5c-6d024d5b10c6/roslaunch-ubuntu
-85293.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.1.200:41115/

SUMMARY
=====

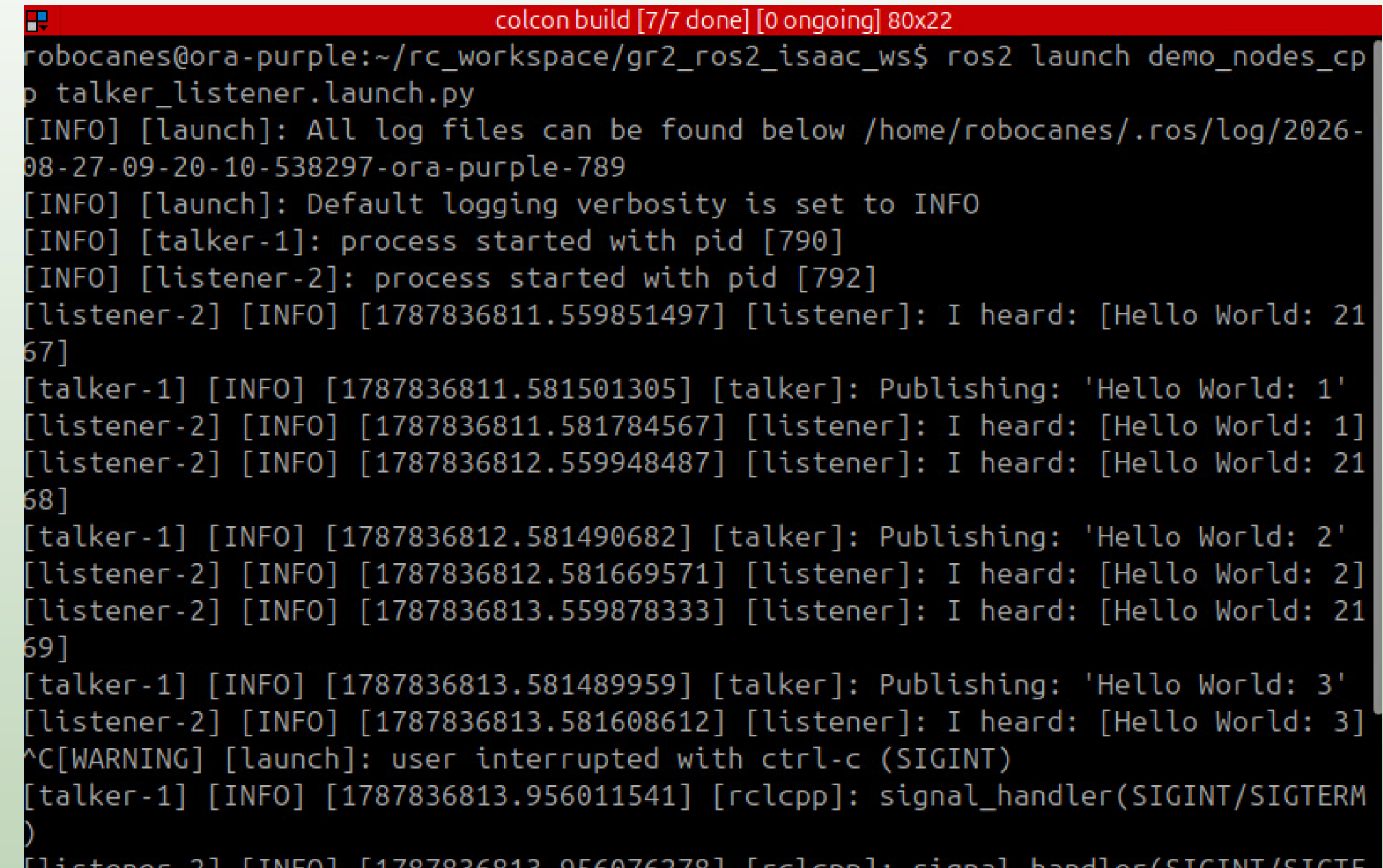
PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES
/
  listener (rospy_tutorials/listener.py)
  talker (rospy_tutorials/talker.py)

auto-starting new master
process[master]: started with pid [85310]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to f1a525ac-69ff-11ef-bc5c-6d024d5b10c6
process[rosout-1]: started with pid [85332]
started core service [/rosout]
process[listener-2]: started with pid [85335]
process[talker-3]: started with pid [85340]
[INFO] [1725373346.950134]: hello world 1725373346.9500668
[INFO] [1725373347.051922]: hello world 1725373347.051781
[INFO] [1725373347.053315]: /listenerI heard hello world 1725373347.051781
```

- ▶ *ROS 2 launch starts and configures multiple nodes.* A launch file also acts like a config file
- ▶ Launch descriptions are commonly written in Python (.launch.py); XML and YAML are also supported.

A terminal window with a red title bar that reads "colcon build [7/7 done] [0 ongoing] 80x22". The terminal shows the command "ros2 launch demo_nodes_cpp talker_listener.launch.py" being executed. The output includes log messages from the launch system and the nodes. The talker node publishes "Hello World" three times, and the listener node receives and prints these messages. The process is interrupted with Ctrl-C, resulting in a warning message and a signal handler call.

```
robocanes@ora-purple:~/rc_workspace/gr2_ros2_isaac_ws$ ros2 launch demo_nodes_cpp talker_listener.launch.py
[INFO] [launch]: All log files can be found below /home/robocanes/.ros/log/2026-08-27-09-20-10-538297-ora-purple-789
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [talker-1]: process started with pid [790]
[INFO] [listener-2]: process started with pid [792]
[listener-2] [INFO] [1787836811.559851497] [listener]: I heard: [Hello World: 2167]
[talker-1] [INFO] [1787836811.581501305] [talker]: Publishing: 'Hello World: 1'
[listener-2] [INFO] [1787836811.581784567] [listener]: I heard: [Hello World: 1]
[listener-2] [INFO] [1787836812.559948487] [listener]: I heard: [Hello World: 2168]
[talker-1] [INFO] [1787836812.581490682] [talker]: Publishing: 'Hello World: 2'
[listener-2] [INFO] [1787836812.581669571] [listener]: I heard: [Hello World: 2]
[listener-2] [INFO] [1787836813.559878333] [listener]: I heard: [Hello World: 2169]
[talker-1] [INFO] [1787836813.581489959] [talker]: Publishing: 'Hello World: 3'
[listener-2] [INFO] [1787836813.581608612] [listener]: I heard: [Hello World: 3]
^C[WARNING] [launch]: user interrupted with ctrl-c (SIGINT)
[talker-1] [INFO] [1787836813.956011541] [rclcpp]: signal_handler(SIGINT/SIGTERM)
[listener-2] [INFO] [1787836813.956076278] [rclcpp]: signal_handler(SIGINT/SIGTERM)
```

ros2 launch demo_nodes_cpp talker_listener.launch.py

Start launch file from package

```
~$ros2 launch package_name file_name.launch.py
```

talker_listener.launch.xml

```
talker_listener.launch.xml
1  <launch>
2    <node pkg="demo_nodes_cpp" exec="talker" output="screen" />
3    <node pkg="demo_nodes_cpp" exec="listener" output="screen" />
4  </launch>
5  |
```

- ▶ **LaunchDescription:** collection of launch actions
- ▶ **node:** action that starts a ROS 2 node
- ▶ **package:** package containing the executable
- ▶ **executable:** executable to run
- ▶ **name:** node name
- ▶ **parameters/remappings:** node configuration
- ▶ **output='screen':** print node output to the terminal

- ▶ Create reusable launch files with DeclareLaunchArgument

```
DeclareLaunchArgument('arg_name', default_value='default_value')
```

- ▶ Read arguments with LaunchConfiguration

```
LaunchConfiguration('arg_name')
```

- ▶ Set arguments on the command line

```
Ros2 launch pkg_launch_file.launch.py arg_name:=value
```

```
<?xml version="1.0"?>
<launch>

  <!-- these are the arguments you can pass this launch file, for example paused:=true -->
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="extra_gazebo_args" default=""/>
  <arg name="gui" default="true"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>
  <arg name="verbose" default="true"/>
  <arg name="output" default="screen"/>
  <arg name="world" default="gazebo_ros_range"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/test/test_worlds/$(arg world).world"/>
    <arg name="paused" value="$(arg paused)"/>
    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
    <arg name="extra_gazebo_args" value="$(arg extra_gazebo_args)"/>
    <arg name="gui" value="$(arg gui)"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
    <arg name="verbose" value="$(arg verbose)"/>
    <arg name="output" value="$(arg output)"/>
  </include>

</launch>
```

ROS 2 NESTED LAUNCH FILES

- Use IncludeLaunchDescription for modular launch files

```
IncludeLaunchDescription(...)
```

- Find package share directories with ament_index_python

```
get_package_share_directory('package_name')
```

- Pass launch arguments to included descriptions

```
launch_arguments={'arg_name': 'value'}.items()
```

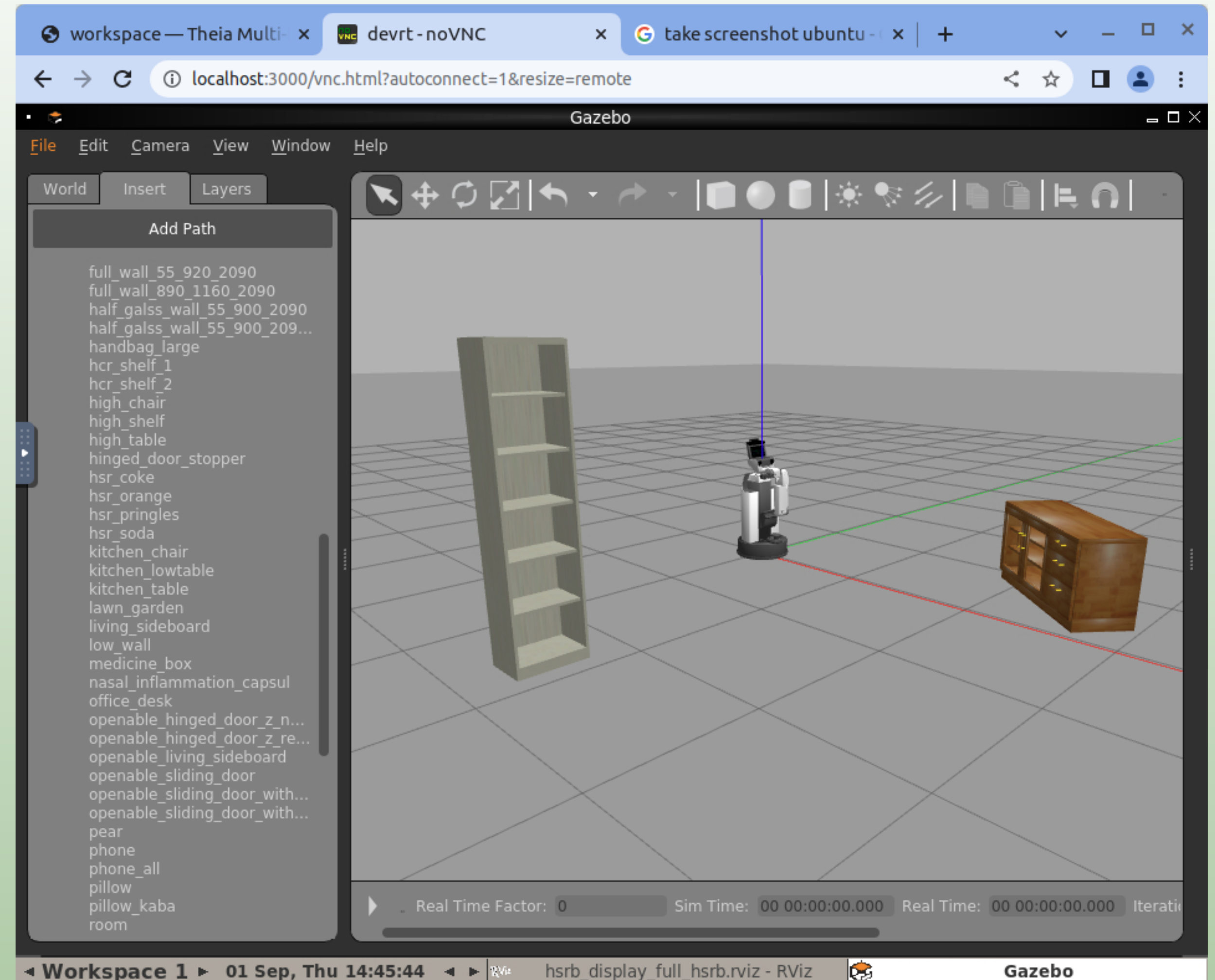
```
def generate_launch_description():
    package_path = get_package_share_directory("gr2_navigation")
    nav2_bringup_path = get_package_share_directory("nav2_bringup")
    default_params_file = os.path.join(package_path, "config", "gr2_nav2_params.yaml")

    params_file = LaunchConfiguration("params_file")
    use_sim_time = LaunchConfiguration("use_sim_time")
    autostart = LaunchConfiguration("autostart")

    return LaunchDescription(
        [
            DeclareLaunchArgument("params_file", default_value=default_params_file),
            DeclareLaunchArgument("use_sim_time", default_value="true"),
            DeclareLaunchArgument("autostart", default_value="true"),
            IncludeLaunchDescription(
                PythonLaunchDescriptionSource(
                    os.path.join(nav2_bringup_path, "launch", "navigation_launch.py")
                ),
                launch_arguments={
                    "params_file": params_file,
                    "use_sim_time": use_sim_time,
                    "autostart": autostart,
                }.items(),
            ),
        ]
    )
```

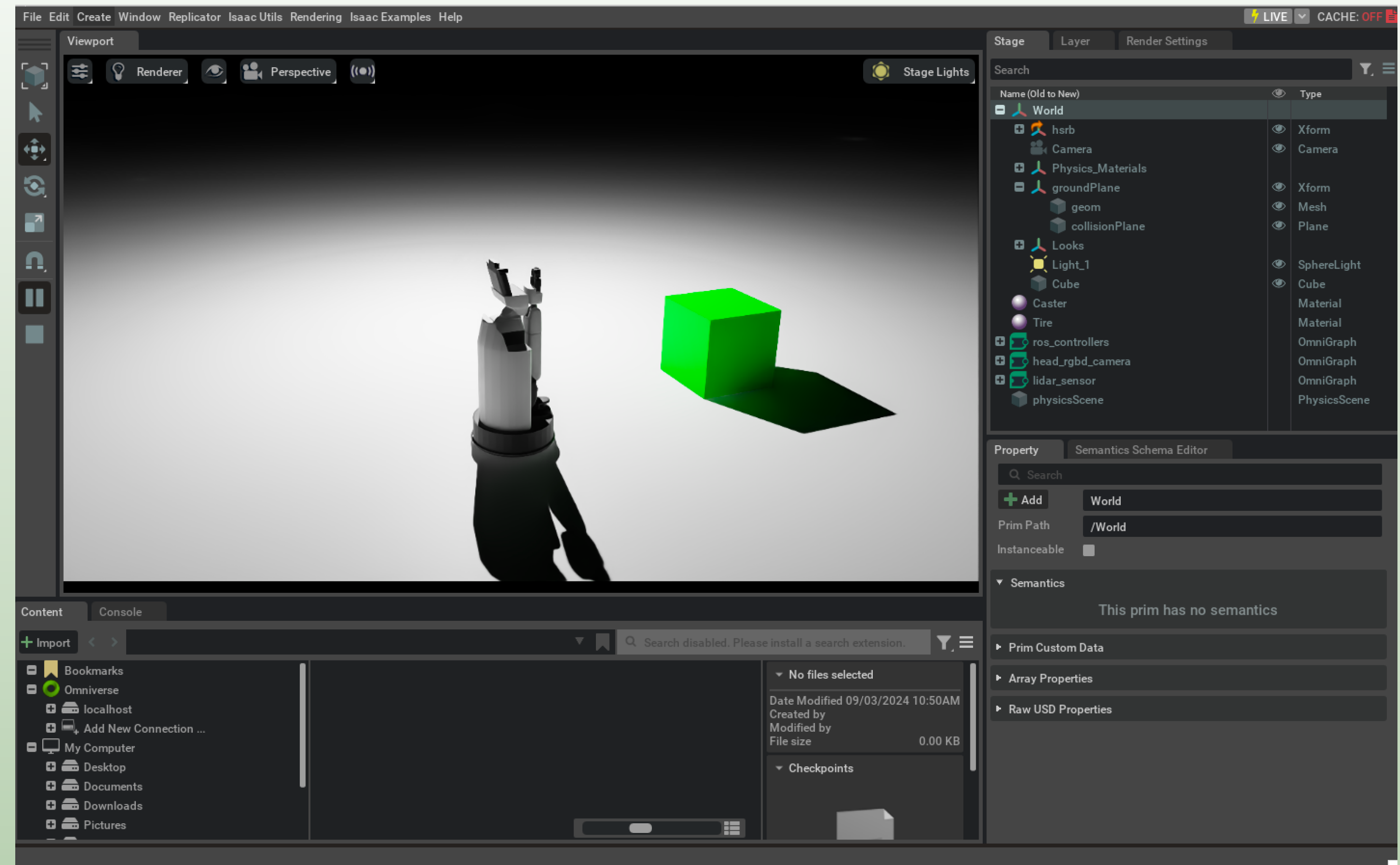
GAZEBO SIMULATOR

- ▶ Simulate 3D rigid-body dynamics
- ▶ Simulate a variety of sensors including noise
- ▶ 3D visualization and user interaction
- ▶ Includes a database of many robots and environments (Gazebo worlds)
- ▶ Provides a ROS interface
- ▶ Extensible with plugins



```
~$ros2 launch ros_gz_sim gz_sim.launch.py
```

- ▶ Simulate 3D rigid-body dynamics
- ▶ Simulate a variety of sensors including noise
- ▶ 3D visualization and user interaction
- ▶ Includes a database of many robots and environments
- ▶ Provides a ROS interface
- ▶ Extensible with plugins



```
$ python run_isaac_gr2_aurora.py
```

- ▶ ROS 2 Documentation

- ▶ <https://docs.ros.org/en/humble/>

- ▶ Installation

- ▶ <https://docs.ros.org/en/humble/Installation.html>

- ▶ Tutorials

- ▶ <https://docs.ros.org/en/humble/Tutorials.html>

- ▶ ROS Index

- ▶ <https://index.ros.org/>

- ▶ ROS 2 Concepts

- ▶ <https://docs.ros.org/en/humble/Concepts.html>

- ▶ ROS 2 CLI

- ▶ <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools.html>

- ▶ colcon

- ▶ <https://colcon.readthedocs.io/>

- ▶ ROS 2 Design

- ▶ <https://design.ros2.org/>

Material adapted from ROS 2 documentation and the original ETH Zürich ROS introduction (<https://rsl.ethz.ch/>)