Course of

Robot Programming with ROS 2

Day 1

2. Hands on!!

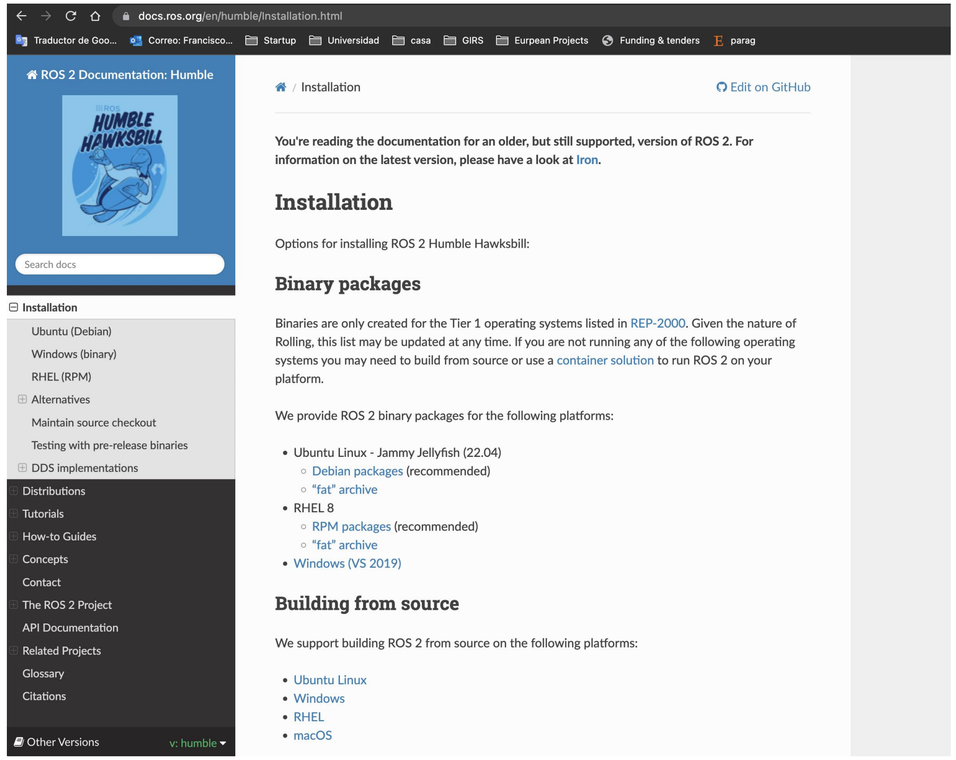
ikerlan

Universidad Rey Juan Carlos

Intelligent Robotics Lab

Installation

- From Binaries
- From sources



https://docs.ros.org/en/humble/Installation.html

First steps in the Terminal

ROS2Cli

```
$ ros2

usage: ros2 [-h] Call ‘ros2 <command> -h‘ for more detailed usage. ...
ros2 is an extensible command-line tool for ROS 2.

...
```

```
ros2 <command> <verb> [<params>|<option>]*
```

```
action      extension_points  node        test
bag         extensions        param       topic
component   interface         pkg         wtf
launch      run               daemon      lifecycle
security    doctor            multicast   service
```

Further readings:

- https://github.com/ros2/ros2cli
- https://github.com/ubuntu-robotics/ros2_cheats_sheet/blob/master/cli/cli_cheats_sheet.pdf

Intelligent Robotics Lab

First steps in the Terminal

Packages

```
$ ros2 pkg list

ackermann_msgs
action_msgs
action_tutorials_cpp
...
```

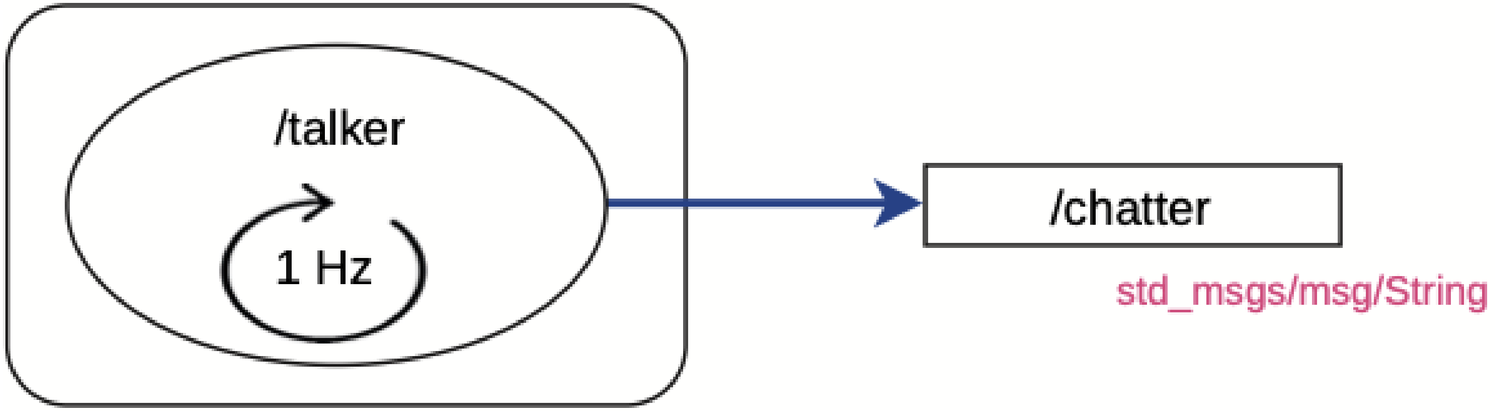
```
$ ros2 pkg executables demo_nodes_cpp

demo_nodes_cpp add_two_ints_client
demo_nodes_cpp add_two_ints_client_async
demo_nodes_cpp add_two_ints_server
demo_nodes_cpp allocator_tutorial
...
demo_nodes_cpp talker
...
```

Intelligent Robotics Lab

First steps in the Terminal

Running a ROS2 program

```
$ ros2 run demo_nodes_cpp talker

[INFO] [1643218362.316869744] [talker]: Publishing: 'Hello World: 1'
[INFO] [1643218363.316915225] [talker]: Publishing: 'Hello World: 2'
[INFO] [1643218364.316907053] [talker]: Publishing: 'Hello World: 3'
...
```

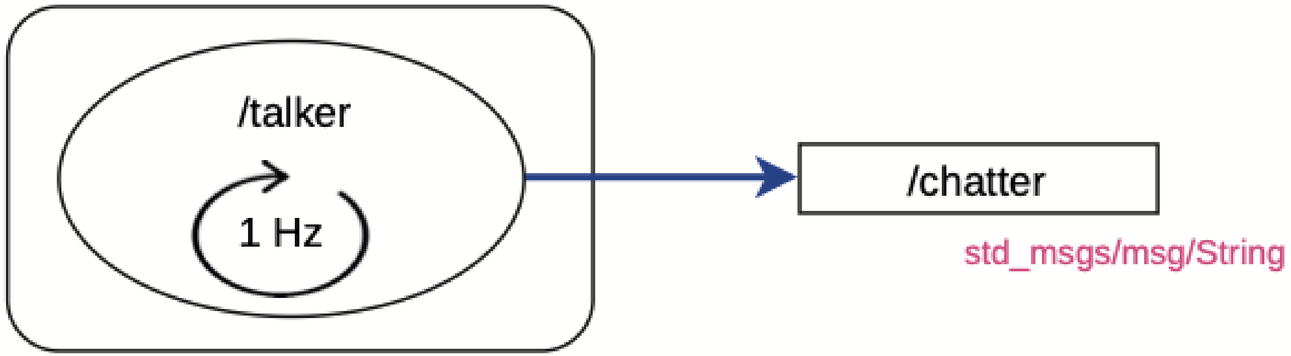


First steps in the Terminal

Running a ROS2 program

```
$ ros2 node list

/talker
```

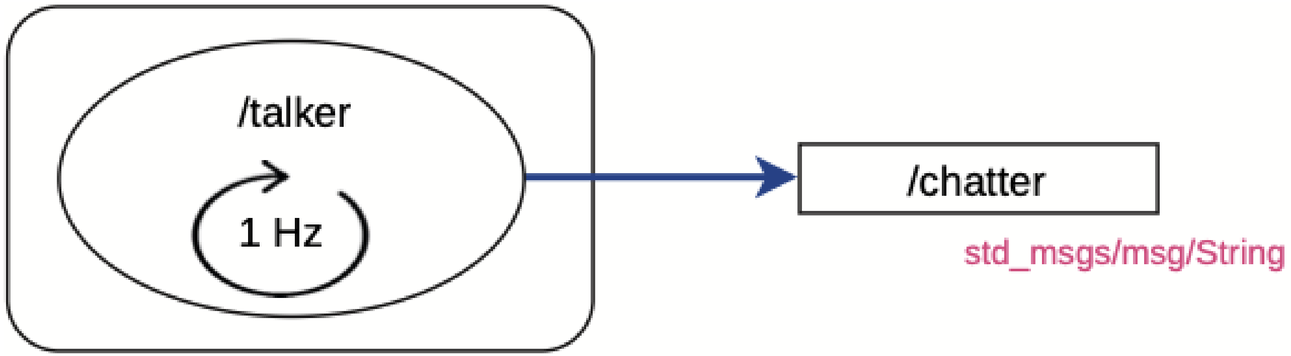
```
$ ros2 topic list

/chatter
/parameter_events
/rosout
```



First steps in the Terminal

Running a ROS2 program

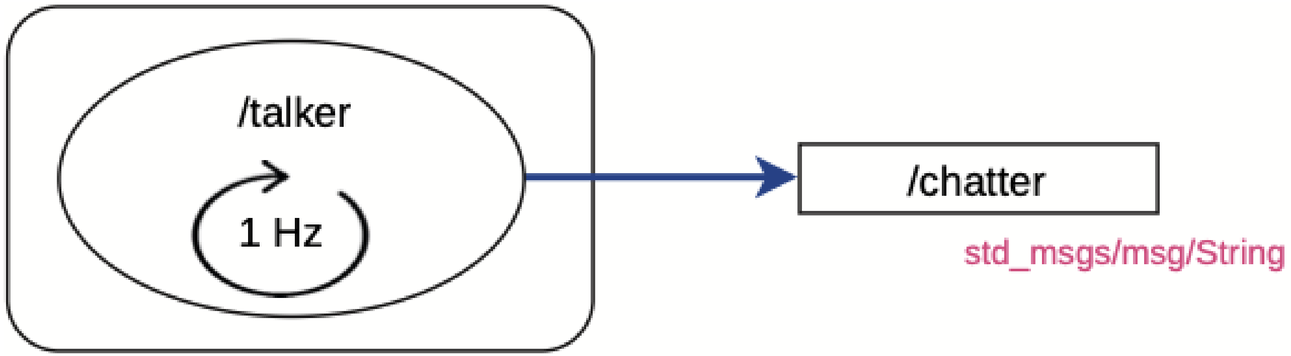
```
$ ros2 node info /talker

/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:

...
```



First steps in the Terminal

Running a ROS2 program

```
$ ros2 topic info /chatter

Type: std_msgs/msg/String
Publisher count: 1

Subscription count: 0
```



First steps in the Terminal

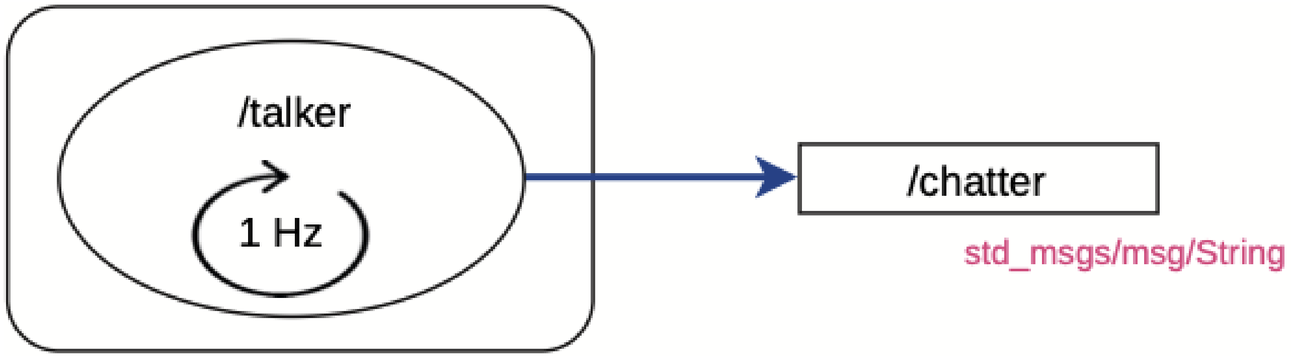
Interfaces

```
$ ros2 interface list

Messages:
    ackermann_msgs/msg/AckermannDrive
    ackermann_msgs/msg/AckermannDriveStamped
    ...
    visualization_msgs/msg/MenuEntry
Services:
    action_msgs/srv/CancelGoal
    ...
    visualization_msgs/srv/GetInteractiveMarkers
Actions:
    action_tutorials_interfaces/action/Fibonacci
    ...
```

```
$ ros2 interface show std_msgs/msg/String

... comments
string data
```

First steps in the Terminal

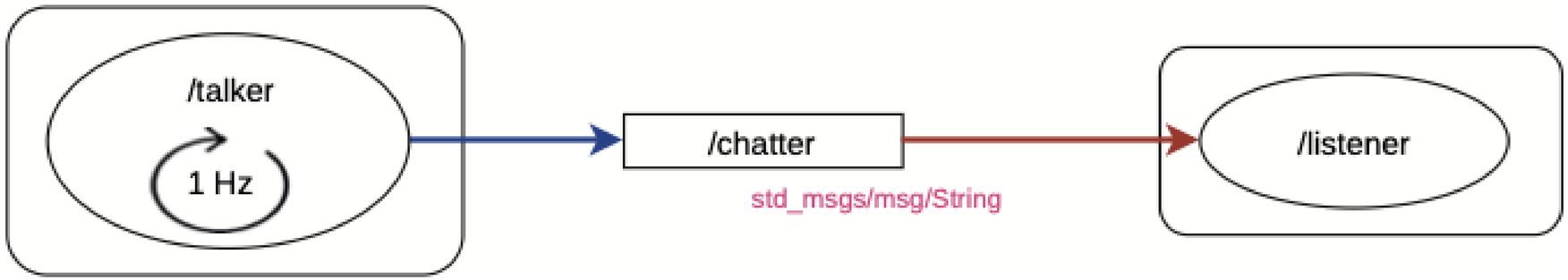
```
$ ros2 topic echo /chatter

data: 'Hello World: 1578'
---
data: 'Hello World: 1579'

...
```



First steps in the Terminal

Running a listener

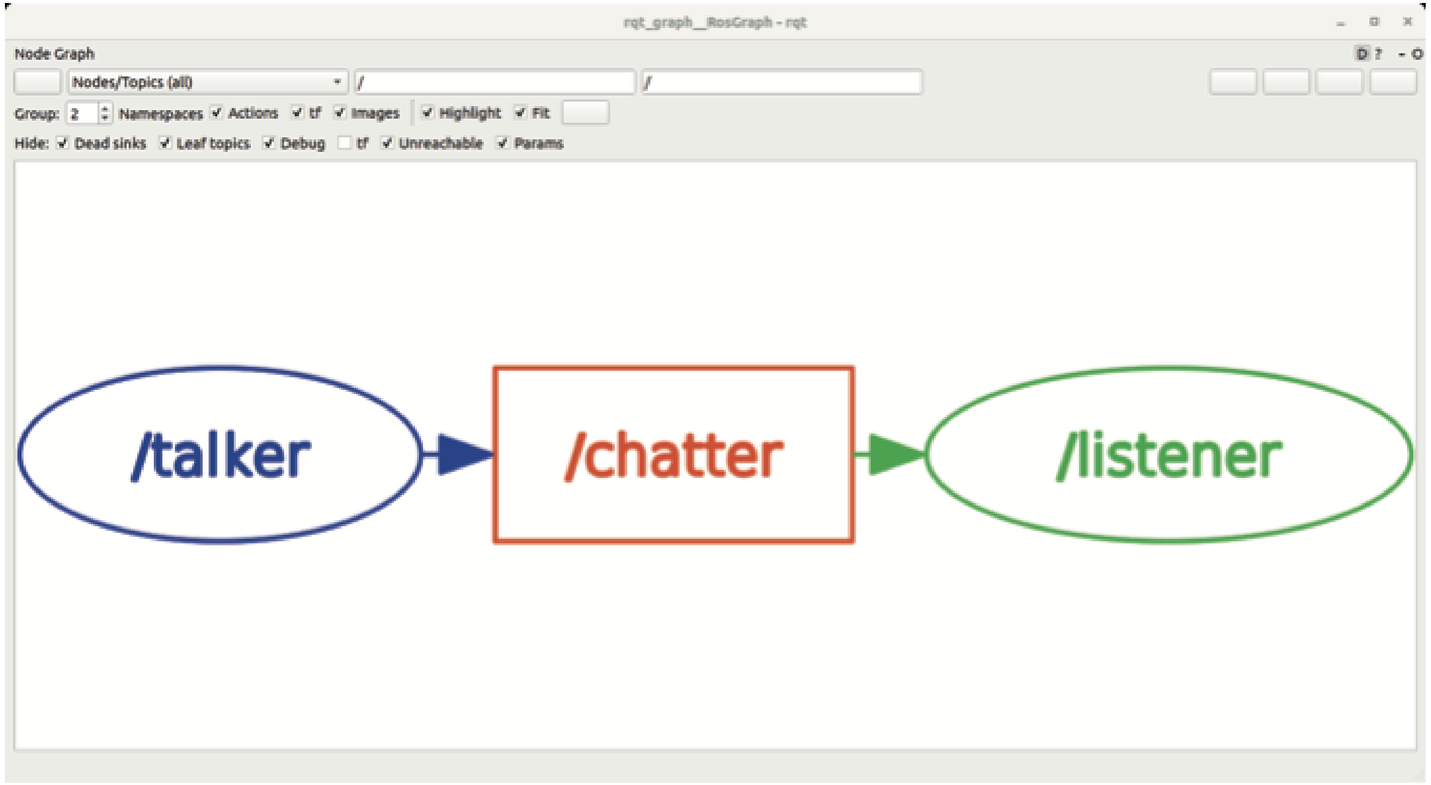
```
$ ros2 run demo_nodes_py listener

[INFO] [1643220136.232617223] [listener]: I heard: [Hello World: 1670]
[INFO] [1643220137.197551366] [listener]: I heard: [Hello World: 1671]
[INFO] [1643220138.198640098] [listener]: I heard: [Hello World: 1672]

...
```



First steps in the Terminal

RQT Tools

```
$ ros2 run rqt_graph rqt_graph
```



Setup a workspace

- Create a workspace with a src folder. All the packages to be compiled will be added to this folder.

```
$ mkdir –p ros2_ws/src
```

- First compilation. Install colcon and compile

```
$ sudo apt install python3-colcon-common-extensions
```

```
$ cd ros2_ws/src && colcon build –symlink-install
```

Setup a workspace

- Clone ikerlan and tiago_simulator repositories

```
$ git clone https://github.com/IntelligentRoboticsLabs/ikerlan.git
```

```
$ git clone https://github.com/jmguerreroh/tiago_simulator.git
```

- Import dependecies

```
$ vcs import –recursive < ikerlan/thirparty.repos
```

```
$ vcs import –recursive < tiago_simulator/thirparty.repos
```

- Install libusb, libftdi & libuvc

```
$ sudo apt install libusb-1.0-0-dev libftdi1-dev libuvc-dev
```

Setup a workspace

- Install udev rules from astra camera, kobuki and rplidar

```
$ cd <workspace-ros>
$ sudo cp src/ThirdParty/ros_astra_camera/astra_camera/scripts/56-orbbec-usb.rules
/etc/udev/rules.d/
$ sudo cp src/ThirdParty/rplidar_ros/scripts/rplidar.rules /etc/udev/rules.d/
$ sudo cp src/ThirdParty/kobuki_ftdi/60-kobuki.rules /etc/udev/rules.d/
$ sudo udevadm control --reload-rules    sudo udevadm trigger
```

- Move xtion calibration

```
$ mkdir -p  /.ros/camera_info
$ cp <workspace>/src/ThirdParty/openni2_camera/openni2_camera/rgb_PS1080_PrimeSens
e.yaml  /.ros/camera_info
```

Intelligent Robotics Lab

Setup a workspace

- Build project

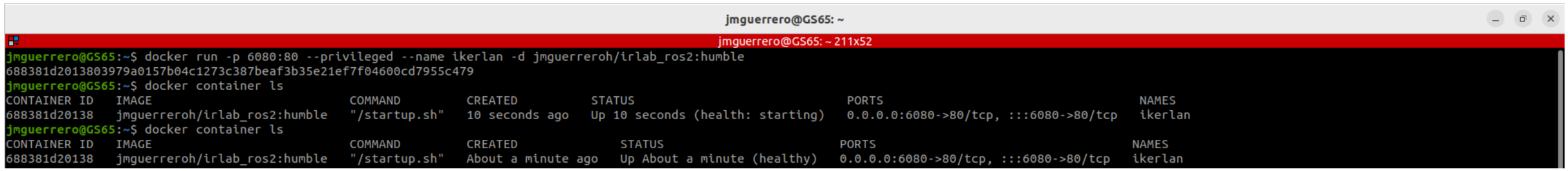
```
$ sudo rosdep init
$ rosdep update
$ rosdep install --from-paths src --ignore-src -r -y
$ colcon build --symlink-install --cmake-args -DBUILD_TESTING=OFF
```

Docker execution

- Docker provides the ability to package and run an application in a loosely isolated environment called a container
- To run the docker prepare for the course with ROS 2 and simulation

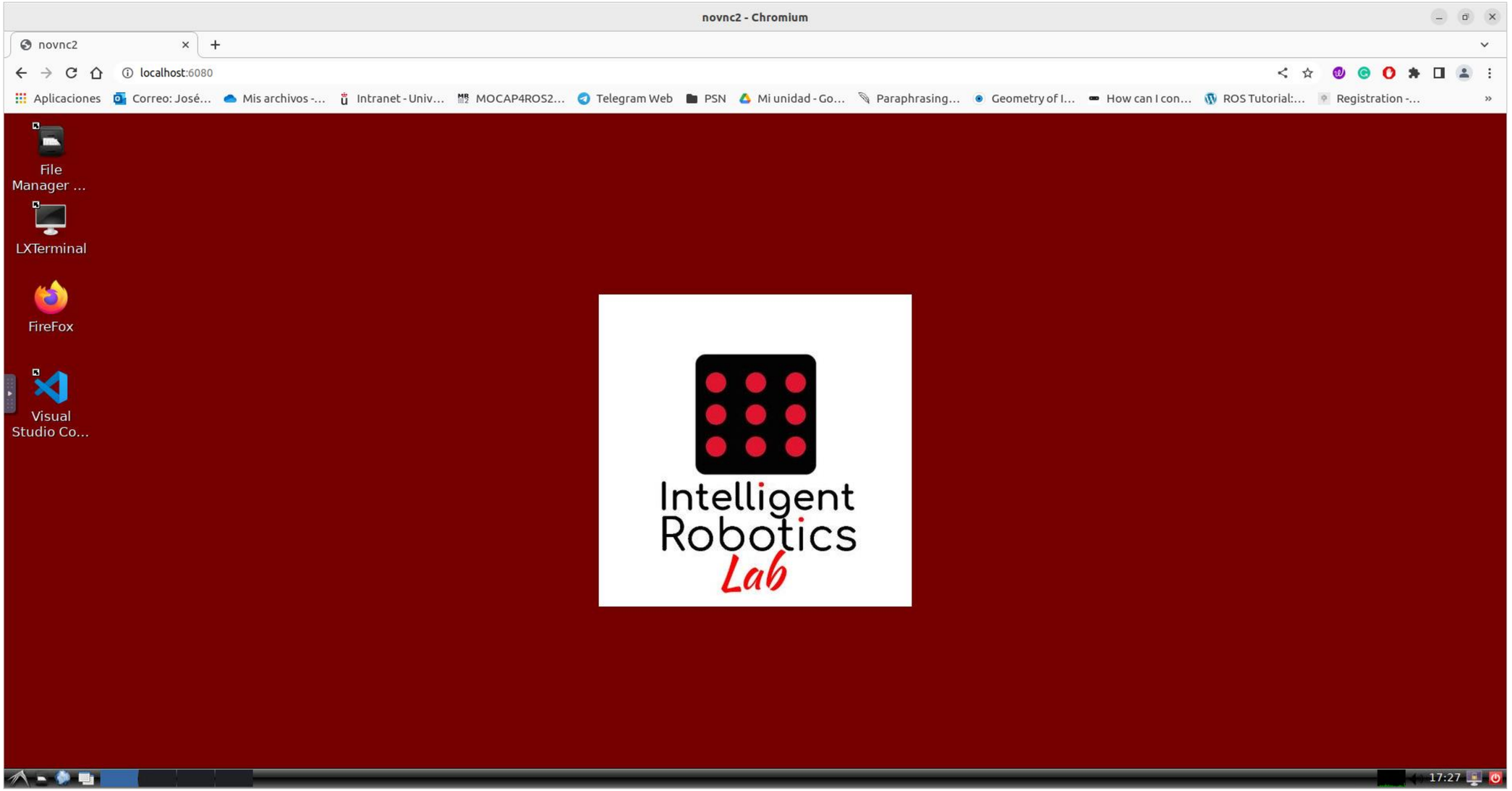
```
$ docker run -p 6080:80 --privileged --name ikerlan -d jmguerreroh/irlab_ros2:humble
```

- Check the status, when it appears as healthy, it means that the docker is running successfully



- Now open your browser and connect to: localhost:6080

Docker execution



Docker execution

- Inside the docker, we can find a workspace: ros2_ws
- It contains all packages necessary to launch the simulation of the Tiago robot and its navigation

```
$ ros2 launch tiago_simulator simulation.launch.py
```

- This simulation is provided by the tiago_simulator package
- Inside this package, we can find a config file: config/params.yaml
- These parameters allow to change the scenario, robot position, and arm

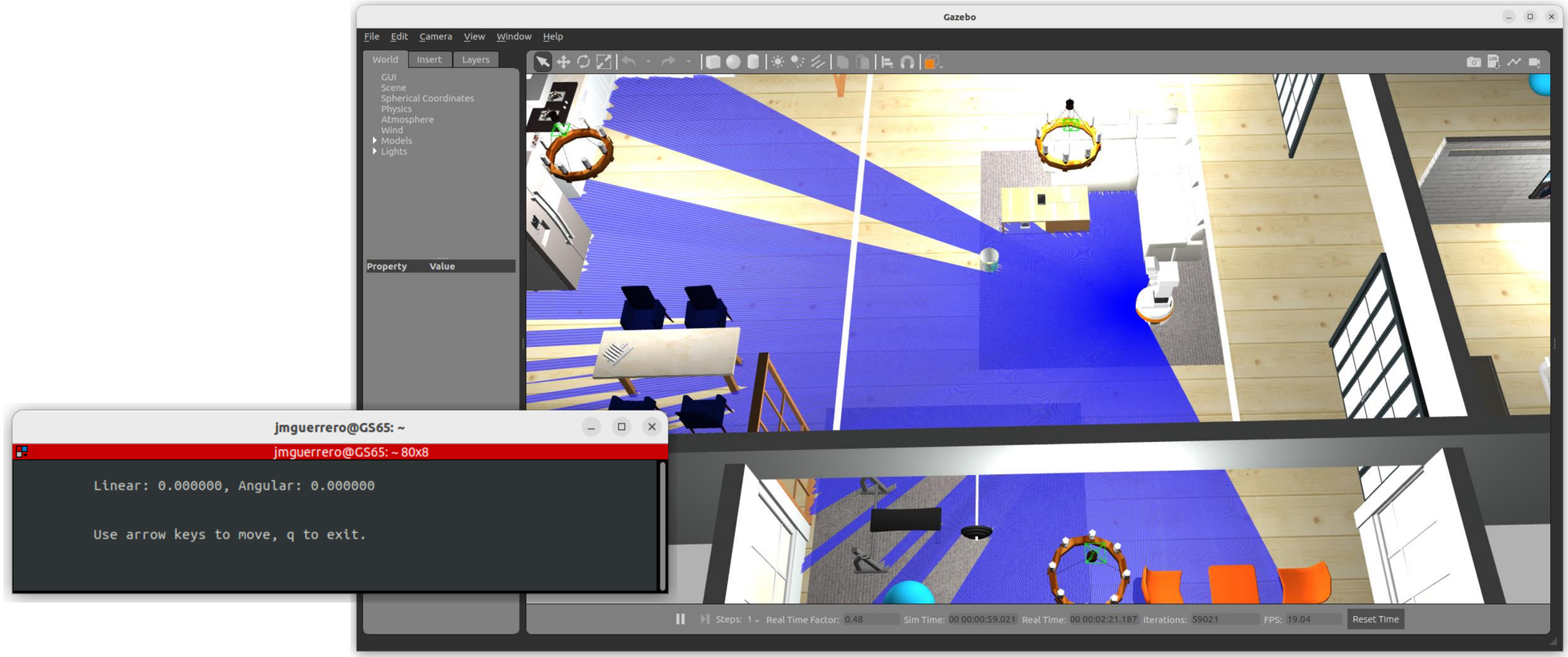
```yaml
tiago_simulator:
  world: aws_house
  robot_position:
    x: 0.0
    y: 0.0
    z: 0.0
    roll: 0.0
    pitch: 0.0
    yaw: 0.0
  tiago_arm: no-arm
```

Intelligent Robotics Lab

Docker execution

- To teleoperate the robot, we can use the key_teleop package

```
$ ros2 run key_teleop key_teleop
```



Docker execution

- If you want to stop the docker:

```
$ docker stop ikerlan
```

- If you want to run the docker again:

```
$ docker start ikerlan
```

RViz2

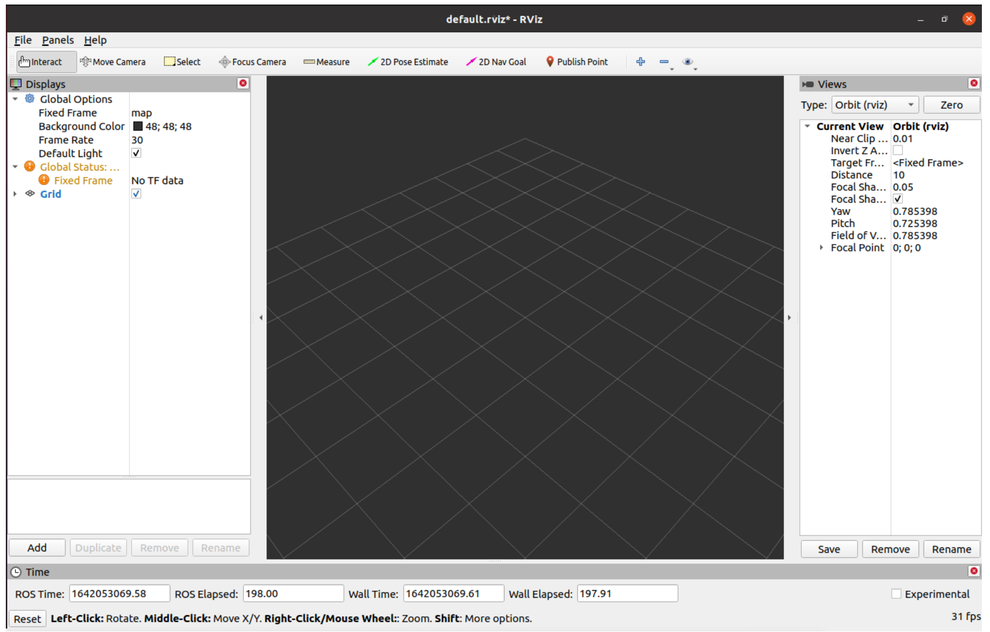
- Open a new terminal and enter the command to open rviz2

```
$ rviz2
```

- * Sometimes is necessary to source the workspace in order to access different messages and paths:

```
$ source <path/to/ws>install/setup.bash
```

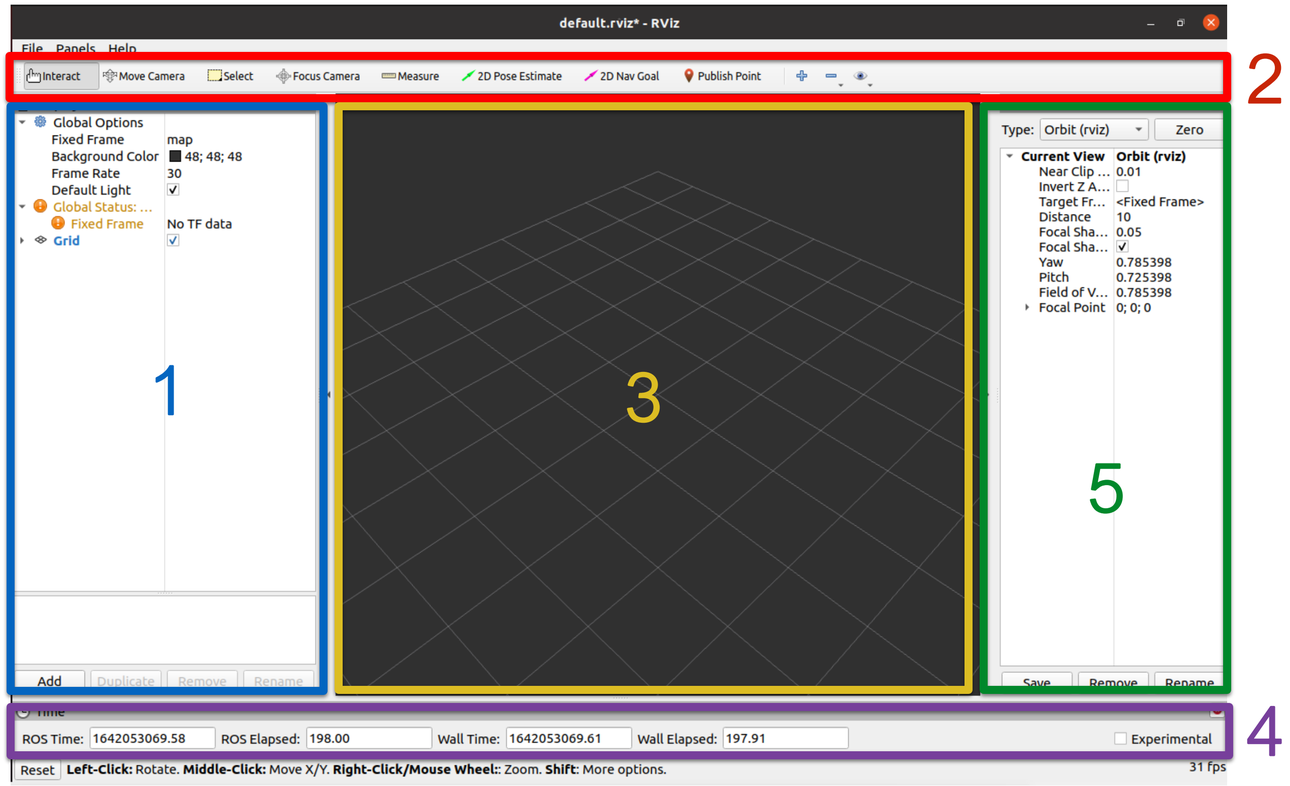
- The next window will be open:



Intelligent Robotics Lab

RViz2

1. List of monitors: a monitor is something that draws something in the 3D world and may have some options available in the display list
2. Toolbar: allows users to use various function buttons to select tools with multiple functions
3. The middle part is the 3D view: the main screen where various data can be viewed in 3D.
4. The time display area, including system time and ROS time
5. Observation angle setting area: different observation angles can be set



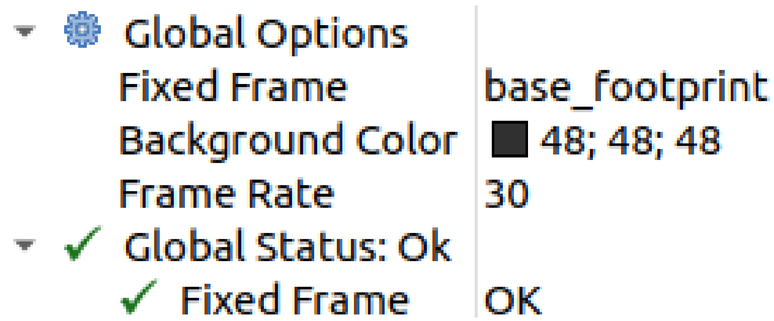
RViz2

- **Global options**

- Key parameters:

  Fixed frame: Indicates the name of the frame used as a reference for all the other frames. You can select every frame available in the combo box. map or odom are the best choices.

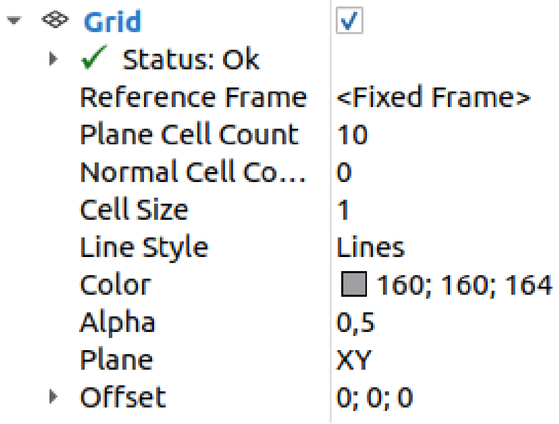
  Frame rate: The maximum frequency used to update the 3D view. 30 or 60 FPS are good values.



RViz2

- **Grid**

- This plugin allows you to visualize a grid normally associated with the floor plane
- Key parameters:
  - Reference frame: The frame used as a reference for the grid coordinates (normally: <fixed_frame>)
  - Plane cell count: The size of the grid in cells
  - Normal cell count: The number of cells in the direction normal to the grid plane (normally: 0)
  - Cell size: Dimensions in meters of each grid cell
  - Plane: The two axes that identify the grid plane



RViz2

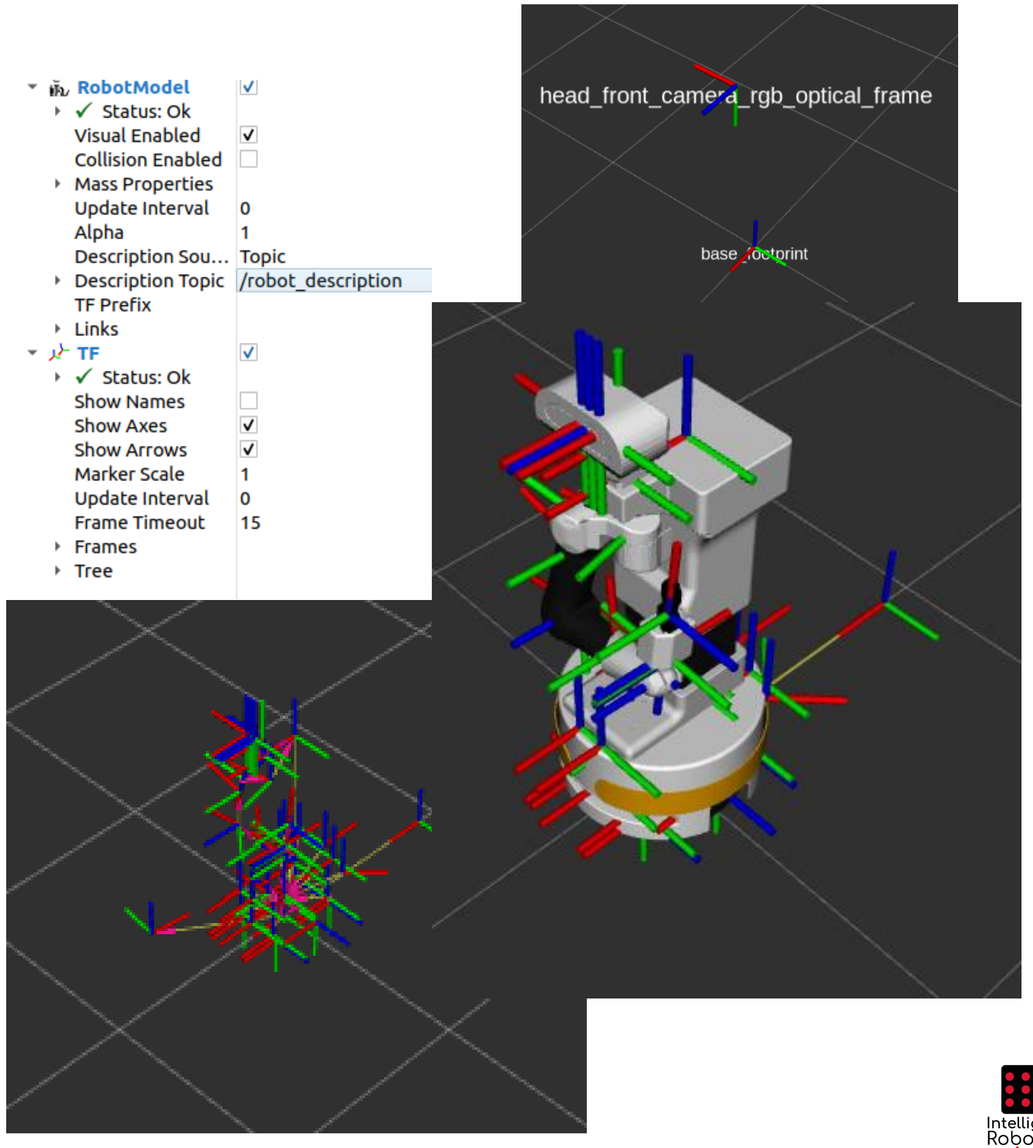
- **Robot model**

- This plugin allows you to visualize the Robot Model according to its description from the URDF model
- Key parameters:
  - Visual enabled: Enable/disable the 3D visualization of the model
  - Description Source: You can choose between File and Topic
- By expanding the Links voice, you can see the whole model tree, with all the joints and the links available and the relative position and orientation in the space relative to the fixed frame

RViz2

- **TF**

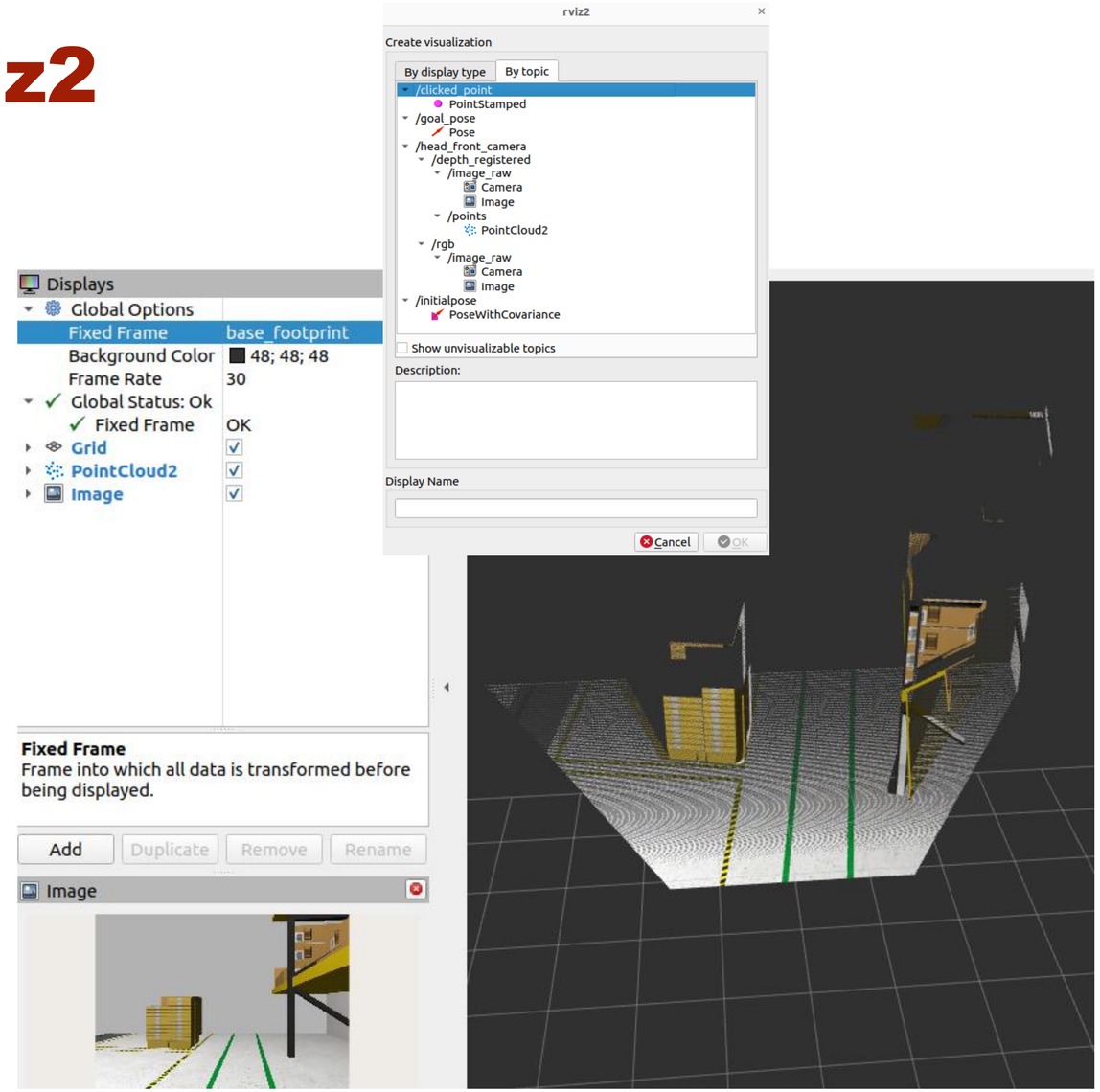
- This plugin allows you to visualize the position and orientation of all frames that compose the TF Hierarchy.
- Key parameters:
  - Show names: Enable/disable the 3D visualization of link names
  - Show axes: Enable/disable the 3D visualization of the axes of the frames
  - Show arrows: Enable/disable the 3D visualization of the arrows that connect the various frames
  - Marker Scale: Used to rescale all the TF objects to let them be more visible and less chaotic
  - Update interval: The update time in seconds. Leave at 0 to see each update
- Critical to using this plugin is the ability to enable/disable visualizing individual frames. This allows you to concentrate only on the parts that are most important for your current task.



RViz2

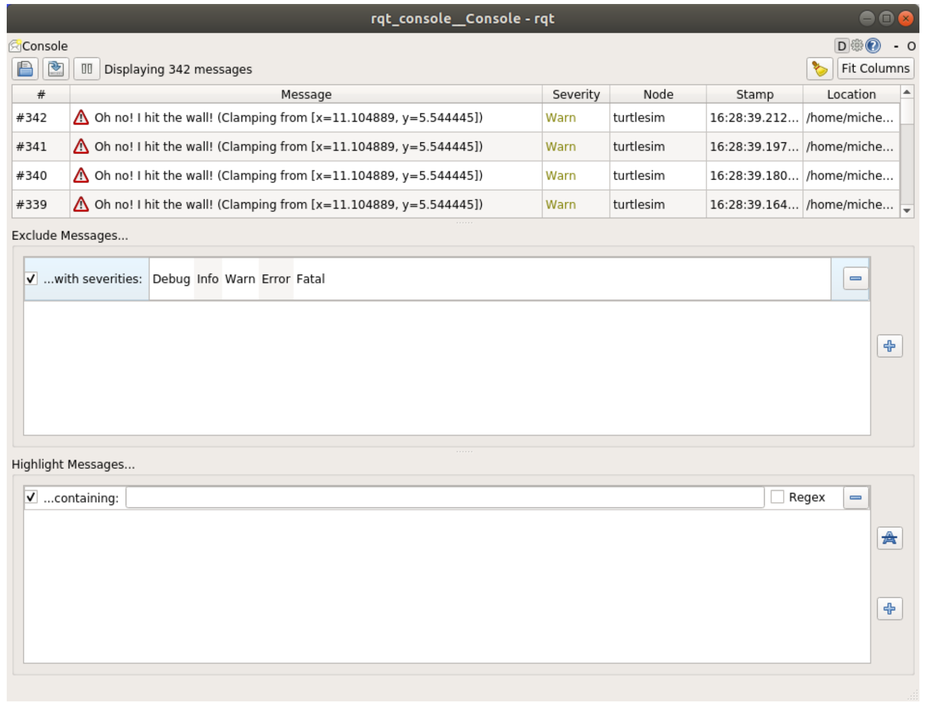
- **Adding displays**

- By default, RViz2 loads the Global options and Grid
- To add new displays, press Add button below this section
- You can also add by actual topics selecting the tab
- Camera and PointCloud, including other topics that publish for example VisualMarkers, can be added

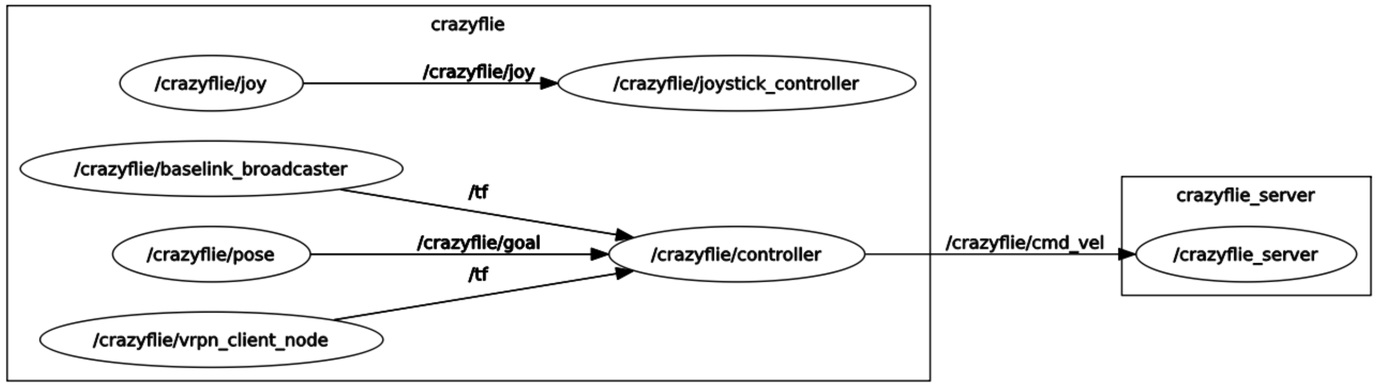


RQT Tools

rqt_console



rqt_graph



rqt_gui !!!