Robot Group’s Workshop

ROS Introduction

曾思銓、林子涵、黃珮涵

2021.08.04

Outline

- Introduction
- ROS File System
- Example1: Publisher and Subscriber
- Example2: Turtlesim
- Something Useful

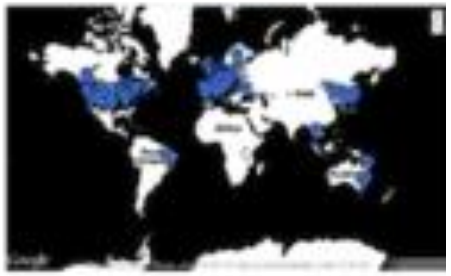
What is ROS?

ROS = **R**obot **O**perating **S**ystem

+

+

+



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

ROS Philosophy

- **Peer to peer** (ROS messages, services, etc.)

  Individual programs communicate over defined API
- **Distributed**

  Programs can be run on multiple computers and communicate over the network.
- **Multi-lingual** (C++, Python, MATLAB, Java, etc.)

  ROS modules can be written in any language for which a client library exists
- **Free and open-source**

  Most ROS software is open-source and free to use.

Why using ROS?

- Different platforms have different SDK

  Realsense SDK

  Zed camera SDK

  Pepper SDK

  LiDAR SDK

- Lots of them have ROS wrapper

  https://github.com/IntelRealSense/realsense-ros

  https://github.com/stereolabs/zed-ros-wrapper

  https://github.com/ros-drivers/velodyne

  https://github.com/robopeak/rplidar_ros

Outline

- Introduction
- ROS File System
- Example1: Publisher and Subscriber
- Example2: Turtlesim
- Something Useful

ROS File System

- Package: Packages are the software organization unit of ROS code. Each package can contain libraries, executables, scripts, or other artifacts.
- Manifests (package.xml): A manifest is a description of a package. It serves to define dependencies between packages and to capture meta information about the package

Some useful tools:

$ rospack find **<package_name>** ⇨ returns the path to package

$ roscd **<package_name>** ⇨ change directory directly to a package

$ rosls **<package_name>** ⇨ ls directly in a package by name

Create ROS Workspace

- Create a workspace

```
$ mkdir -p ~/catkin_ws/src        ⇨ create a folder named catkin_ws
$ cd ~/catkin_ws/                 ⇨ Enter the folder catkin_ws
```

名稱	大小	類型
src	0 個項目	資料夾

- Compile with empty workspace

```
$ catkin_make
```

名稱	大小	類型
build	1 個項目	資料夾
devel	0 個項目	資料夾
src	0 個項目	資料夾

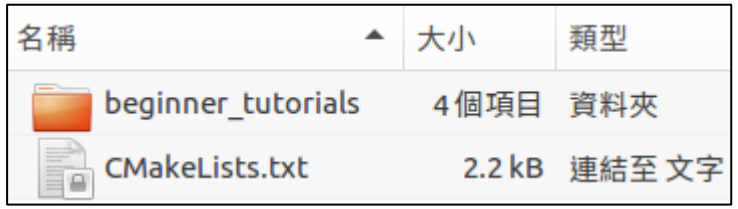
- Overlay this workspace on top of your environment

```
$ source devel/setup.bash
```

- Make sure your workspace is properly overlayed

```
$ echo $ROS_PACKAGE_PATH
⇨ /home/<user_name>/catkin_ws/src:/opt/ros/kinetic/share
```

Create Your Own Package



名稱	大小	類型
beginner_tutorials	4 個項目	資料夾
CMakeLists.txt	2.2 kB	連結至 文字

- Enter the src folder in your workspace you just created

```
$ cd ~/catkin_ws/src
```

⇨ Enter the folder catkin_ws/src

- create a new package

catkin_create_pkg **<package_name> [depend1] [depend2] [depend3]**

```
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

- Compile again

```
$ cd ~/catkin_ws
$ catkin_make
```

⇨ Go back to the previous folder catkin_ws

名稱	大小	類型
include	1 個項目	資料夾
src	0 個項目	資料夾
CMakeLists.txt	7.1 kB	文字
package.xml	2.9 kB	標記

- Source your environment

```
$ source devel/setup.bash
```

Outline

- Introduction
- ROS File System
- Example1: Publisher and Subscriber
- Example2: Turtlesim
- Something Useful

Create Your Own Publisher/Subscriber

- Start a master

```
$ roscore
```

- Write the nodes:

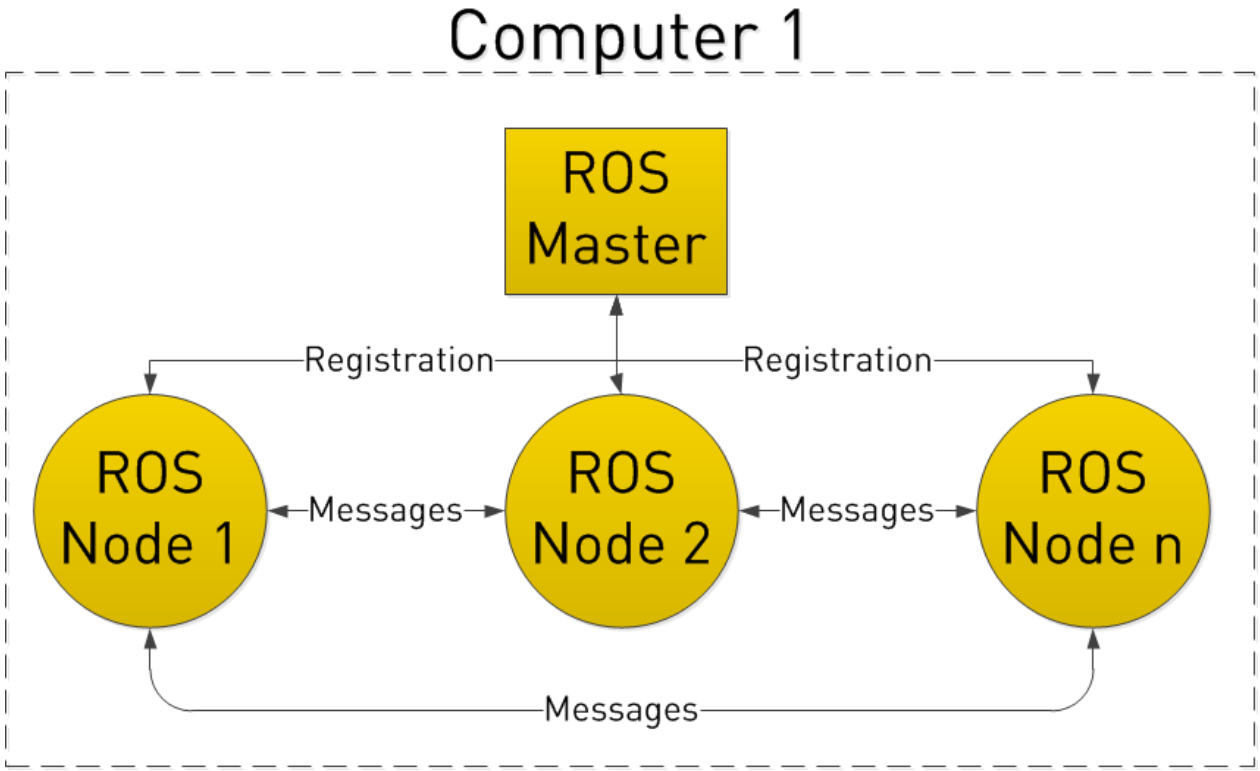
  talker, listener

- Communication (Topic):

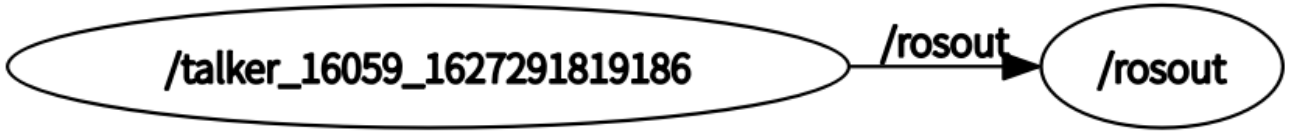
  /chatter

- Install rqt package

```
$ sudo apt-get install ros-kinetic-rqt
$ sudo apt-get install ros-kinetic-rqt-graph
$ sudo apt-get install ros-kinetic-rqt-common-plugins
```



Writing Publisher Node



- Create a talker node

```
$ roscd beginner_tutorials
$ mkdir scripts
$ cd scripts
$ code talker.py
```

- Run the node

```
$ sudo chmod +x talker.py
$ rosrun beginner_tutorials talker.py
```

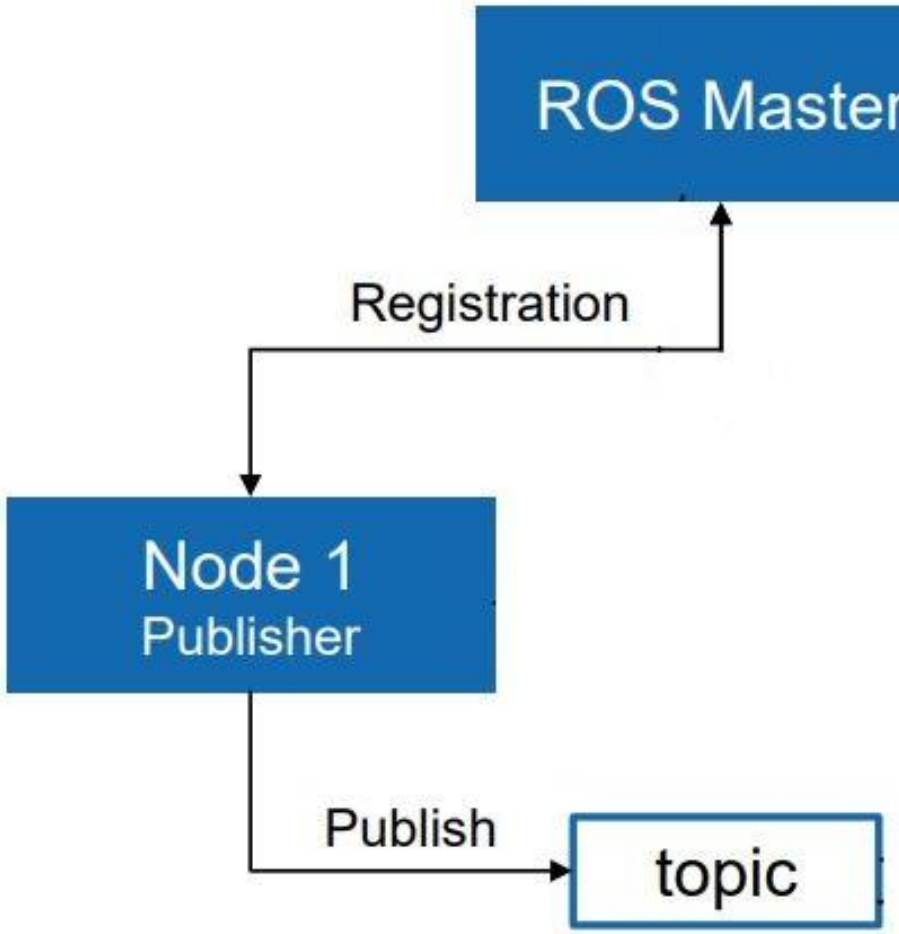
- Check the node

```
$ rosnode list
$ rosnode info  /talker{tab}
```

```
user@user-All-Series:~$ rosnode list
/rosout
/talker_16059_1627291819186
```

```
user@user-All-Series:~$ rosnode info /talker_16059_1627291819186
--------------------------------------------------------------------------------
Node [/talker_16059_1627291819186]
Publications:
 * /chatter [std_msgs/String]
 * /rosout [rosgraph_msgs/Log]

Subscriptions: None

Services:
 * /talker_16059_1627291819186/get_loggers
 * /talker_16059_1627291819186/set_logger_level


contacting node http://user-All-Series:45473/ ...
Pid: 16059
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
```

Run Publisher Node

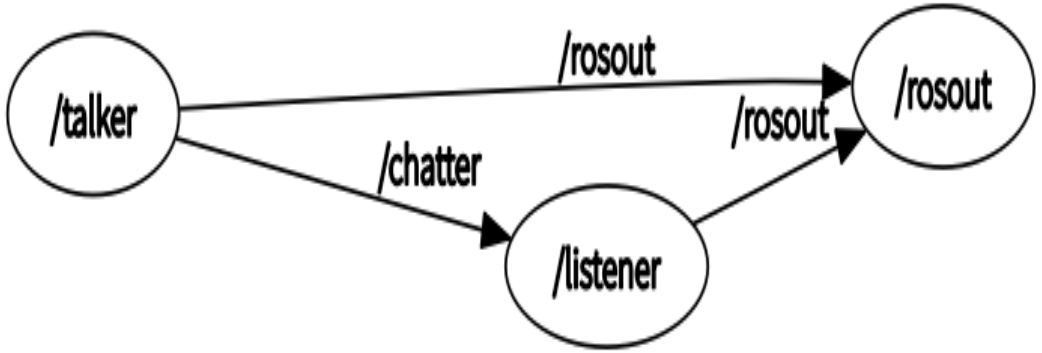



```
#!/usr/bin/env python
# talker.py
import rospy
from std_msgs.msg import String

def talker():
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10)  # frequency 10Hz
    pub = rospy.Publisher('chatter', String, queue_size=10)
    while not rospy.is_shutdown():
        msg = "hello world“
        rospy.loginfo(msg)
        pub.publish(msg)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

node

Publisher

Writing Subscriber Node



- Create a listener node

```
$ roscd beginner_tutorials
$ mkdir scripts && cd scripts
$ code listener.py
```

- Run the node

```
$ sudo chmod +x talker.py
$ rosrun beginner_tutorials listener.py
```

- Check the topic

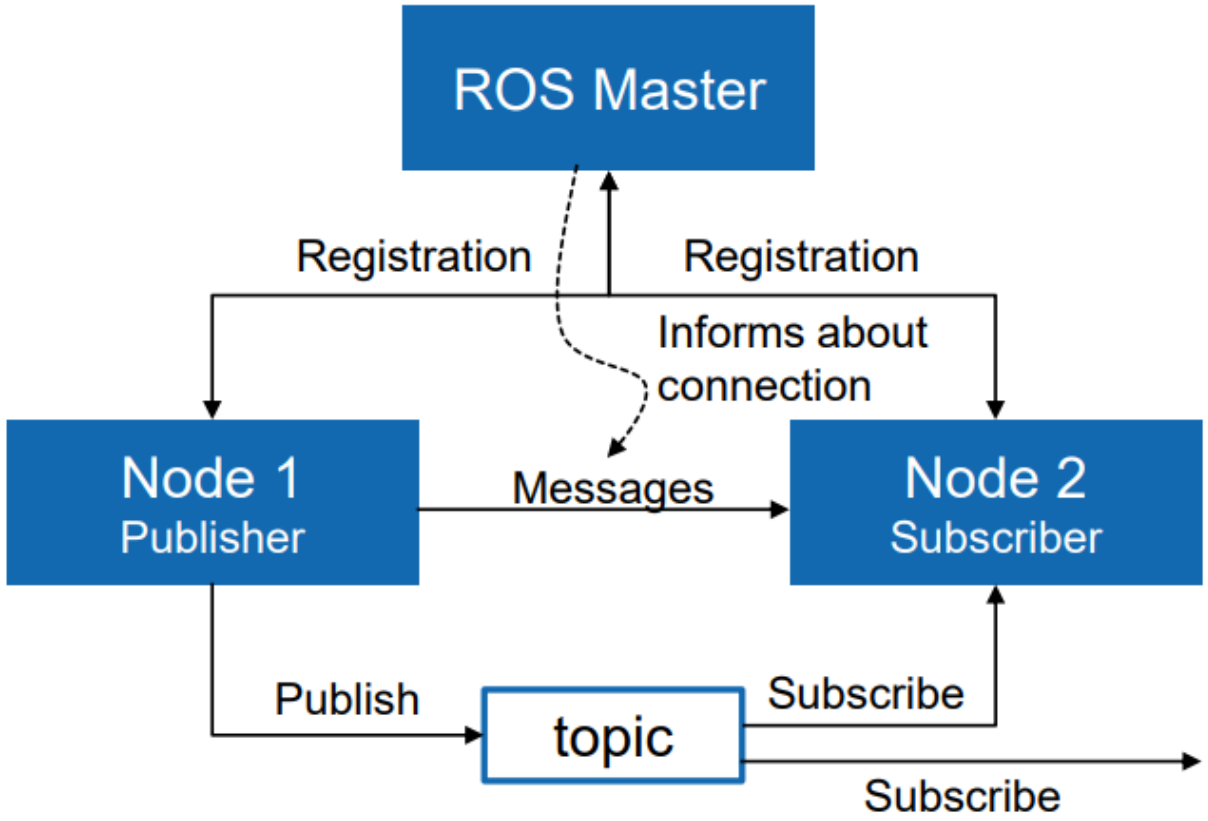
```
$ rqt_graph
$ rostopic list
$ rostopic echo /chatter
$ rostopic info /chatter
```

```
user@user-All-Series:~$ rostopic list
/chatter
/rosout
/rosout_agg
```

```
user@user-All-Series:~$ rostopic echo chatter
data: "Hello world "
---
data: "Hello world "
---
data: "Hello world "
```

```
user@user-All-Series:~$ rostopic info /chatter
Type: std_msgs/String

Publishers:
 * /talker_6912_1627258734572 (http://user-All-Series:43289/)

Subscribers:
 * /listener_7058_1627258769197 (http://user-All-Series:40923/)
```

Run Subscriber Node




```
#!/usr/bin/env python
# listener.py
import rospy
from std_msgs.msg import String

def callback(msg):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", msg.data)
def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber('chatter', String, callback)
    # simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

node

Subscriber

Another way to run the nodes - ROS Launch

- Launch is a tool for launching multiple nodes
  (as well as setting parameters)
- If not yet running, launch automatically starts a roscore
- Write a simple launch file

```
$ roscd beginner_tutorials
$ mkdir  launch && cd launch
$ code talker_listener.launch
```

- Run the launch roslaunch **<package> <filename>**

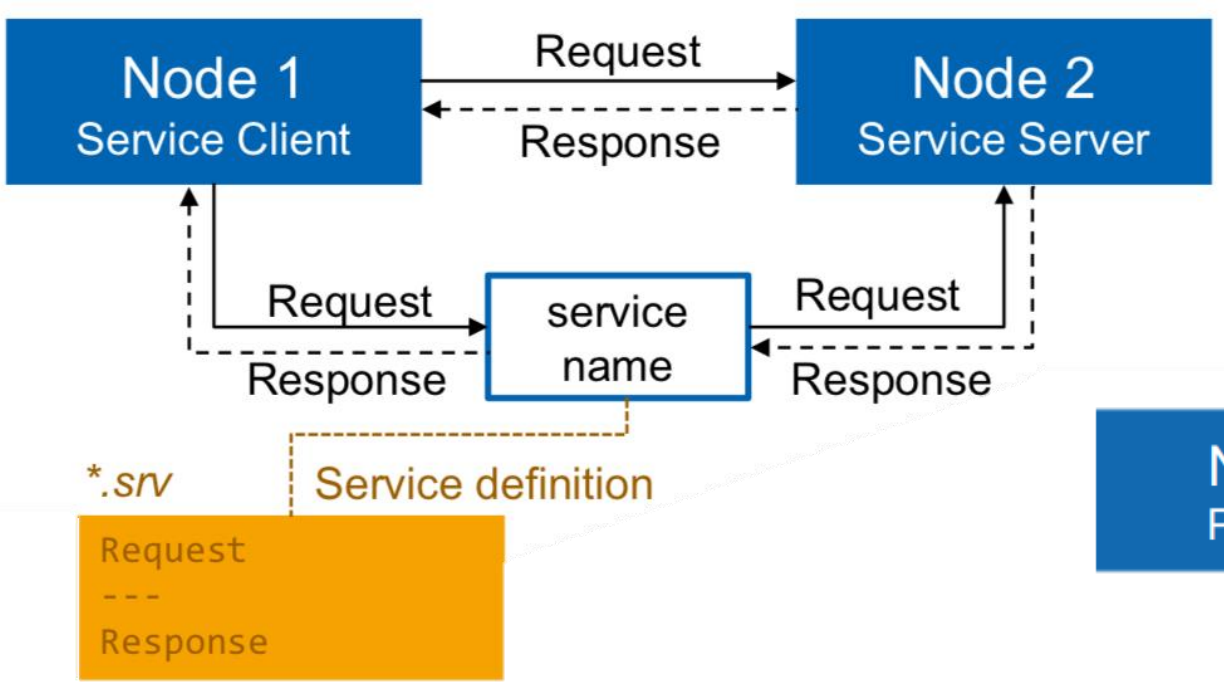
```
$ roslaunch beginner_tutorials talker_listener.launch
```

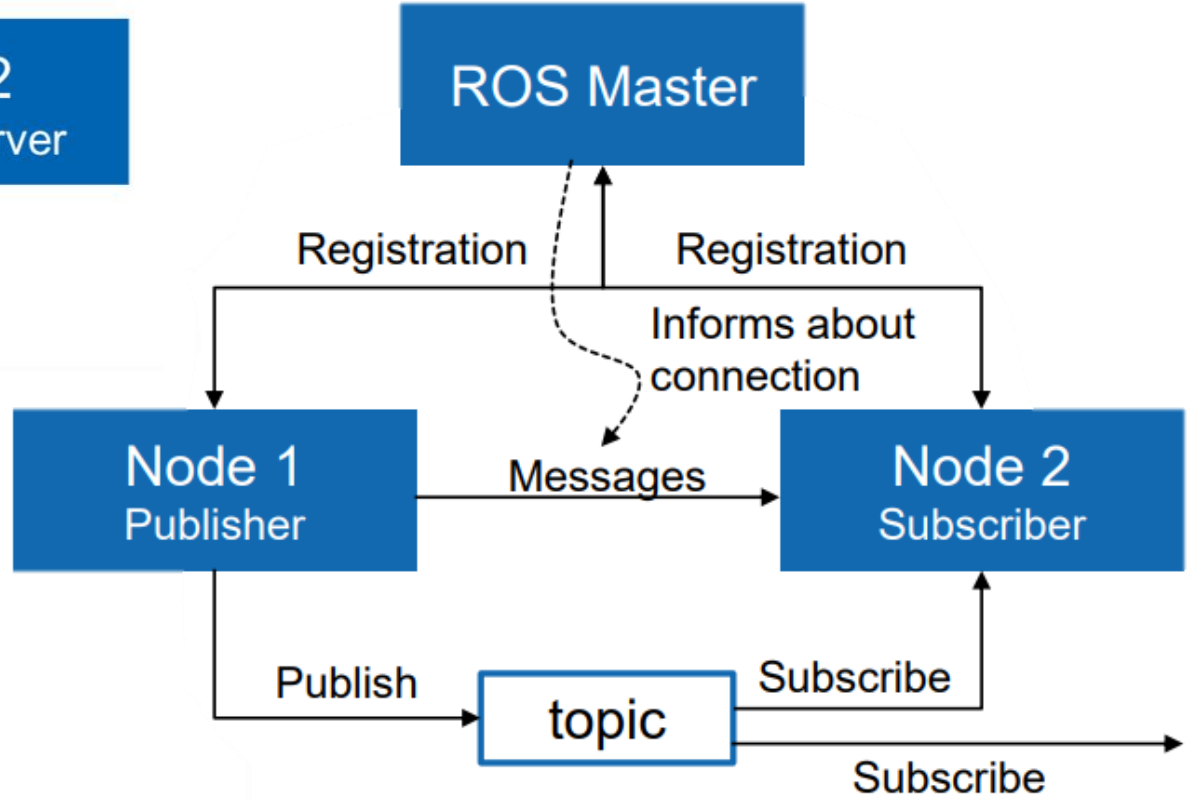
ROS Launch

```
<launch>
    <node name="talker" pkg="beginner_tutorials" type="talker.py" output="screen"/>
    <node name="listener" pkg="beginner_tutorials" type="listener.py" output="screen"/>
</launch>
```

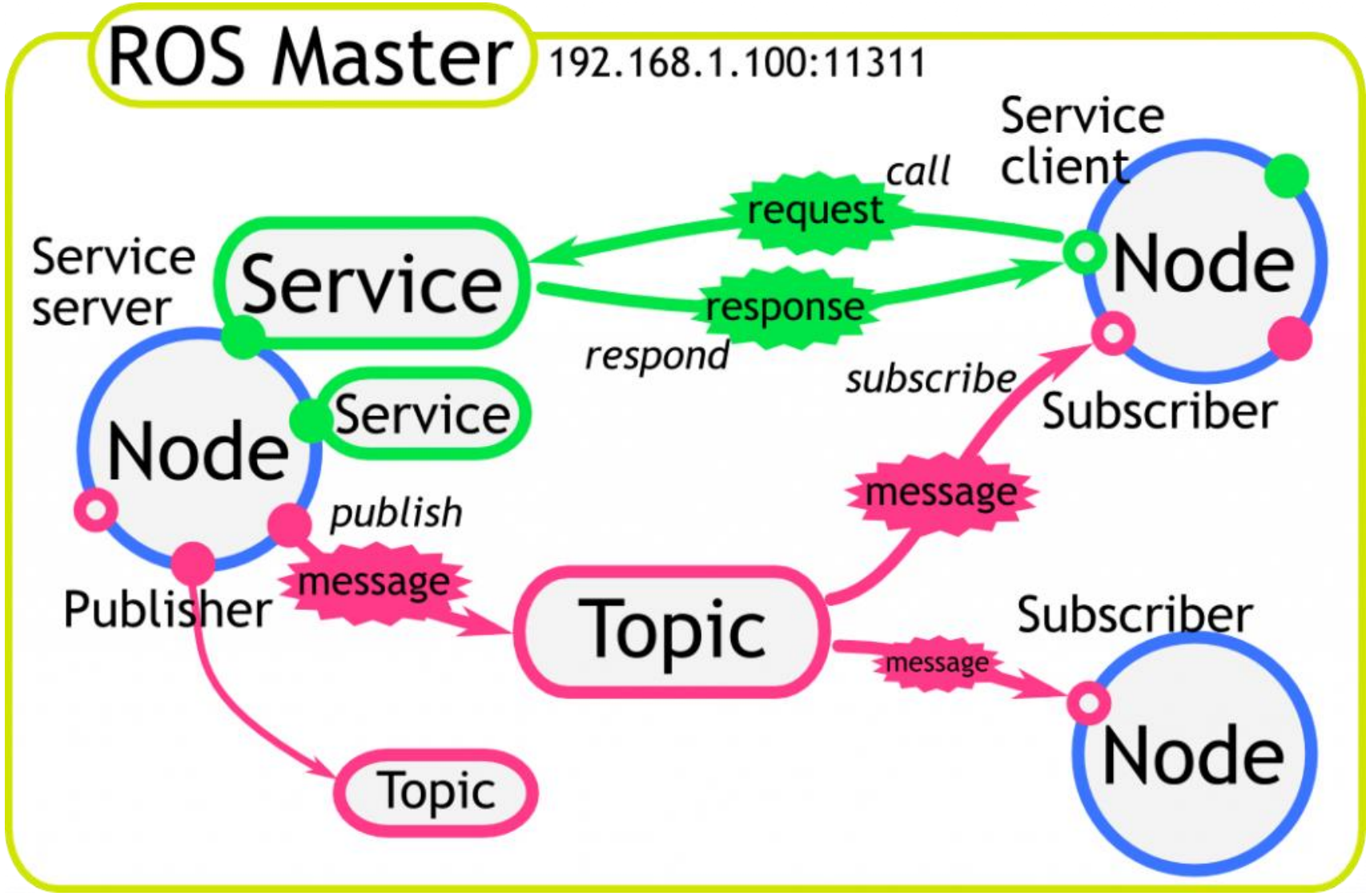
- **launch**: Root element of the launch file
- **node**: Each <node> tag specifies a node to be launched
- **name**: Name of the node (free to choose)
- **pkg**: Package containing the node
- **type**: Type of the node, there must be a corresponding executable with the same name
- **output**: Specifies where to output log messages
  (screen: console, log: log file)

Compare ROS Services to Topics



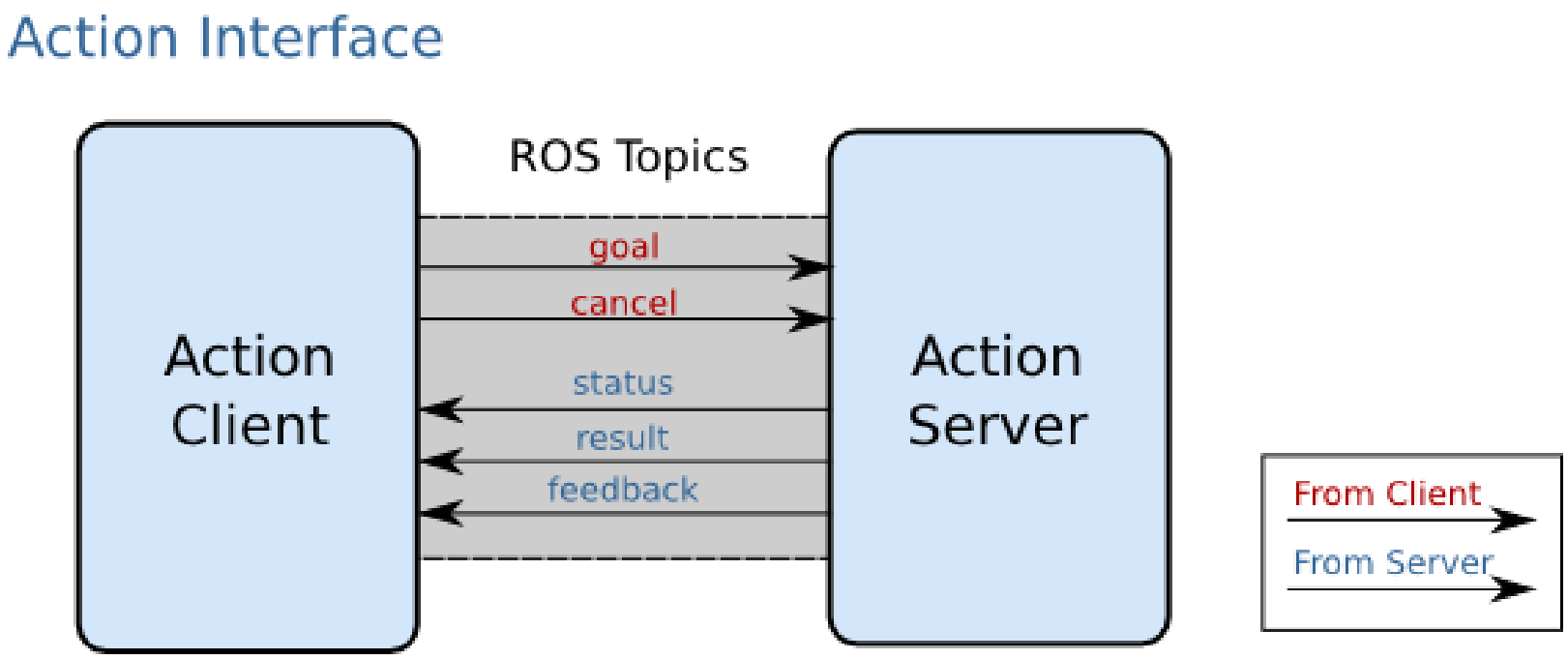


ROS Topics & Services



ROS Action

- A service takes a long time to execute, e.g., navigation
- Goals and results can be analogue to request and response
- Client can cancel the goal
- Server periodically send feedback to client



Outline

- Introduction
- ROS File System
- Example1: Publisher and Subscriber
- Example2: Turtlesim
- Something Useful

Turtlesim Package

- Install turtlesim package

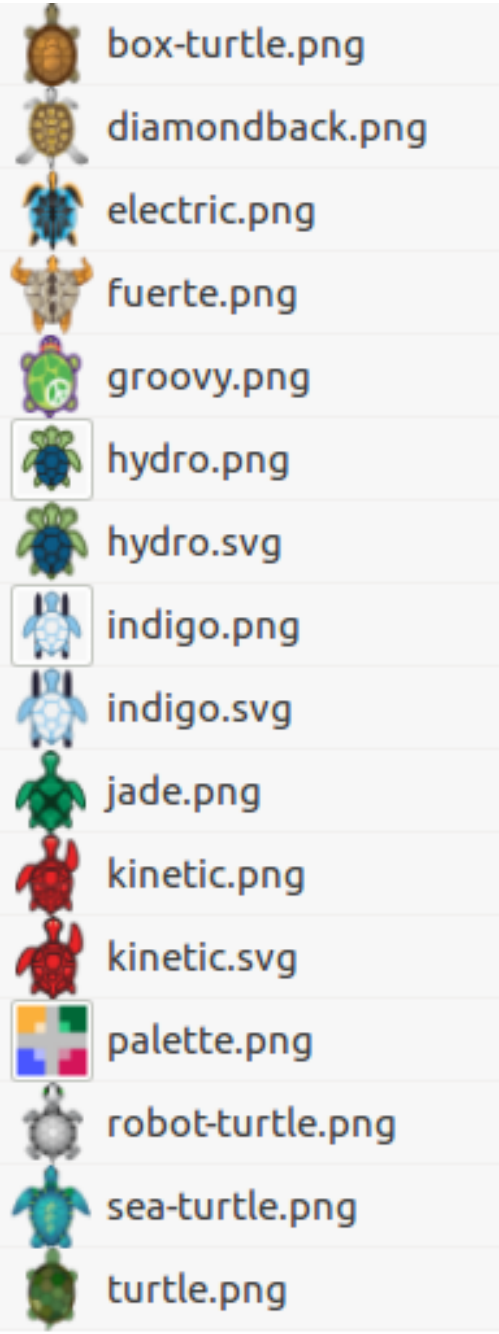
```
$ sudo apt-get install ros-$(rosversion -d)-turtlesim
```

- Check the packages

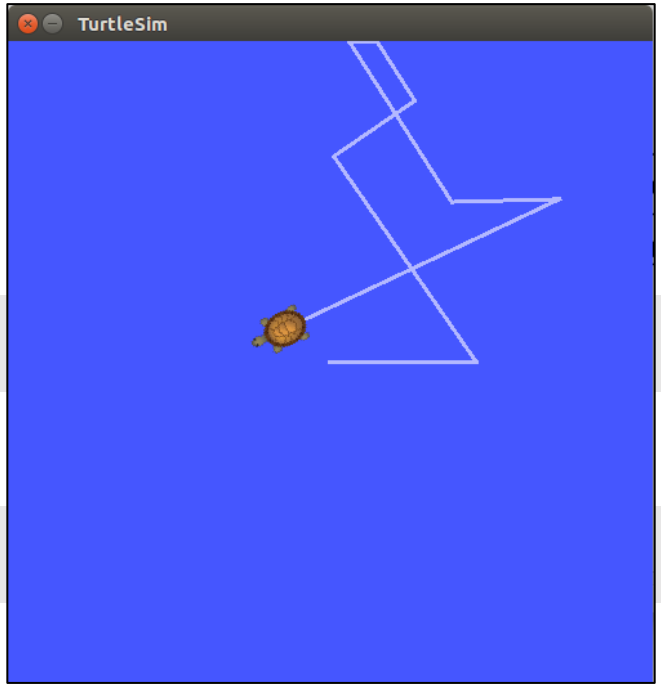
```
$ cd /opt/ros/kinetic/share/turtlesim/
$ cd /opt/ros/kinetic/ lib/turtlesim/
```

名稱	大小
srv	5 個項目
msg	2 個項目
images	16 個項目
cmake	4 個項目
package.xml	1.5 kB

名稱	大小
draw_square	76.4 kB
mimic	64.0 kB
turtlesim_node	253.2 kB
turtle_teleop_key	27.2 kB



Example2-1: Turtlesim



- Run turtlesim node

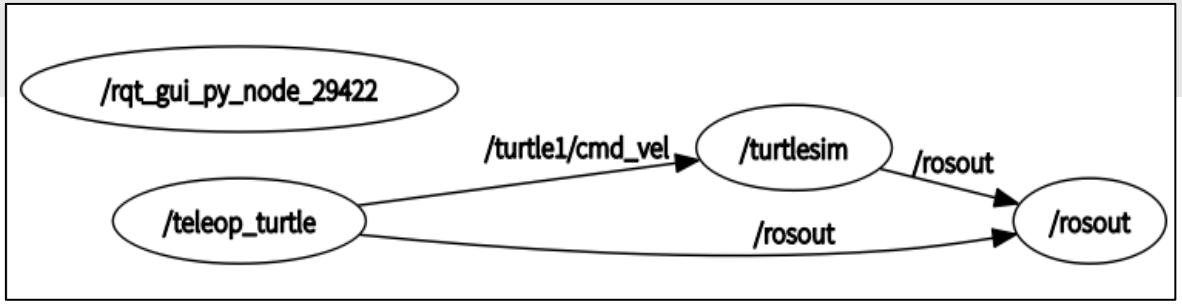
```
$ rosrun turtlesim turtlesim_turtlesim_node
```

- Keyboard teleoperation

```
$ rosrun turtlesim turtle_teleop_key
```

- Show rqt_graph

```
$ rqt_graph
```



- Shows the data published on a topic

```
$ rostopic list
$ rostopic echo turtle1/pose
```

```
---
x: 7.56044435501
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 7.56044435501
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
```

Example2-1: Turtlesim - ROS Service

- Check the service turtlesim node provides

```
$ rosservice list
```

- Clear the background

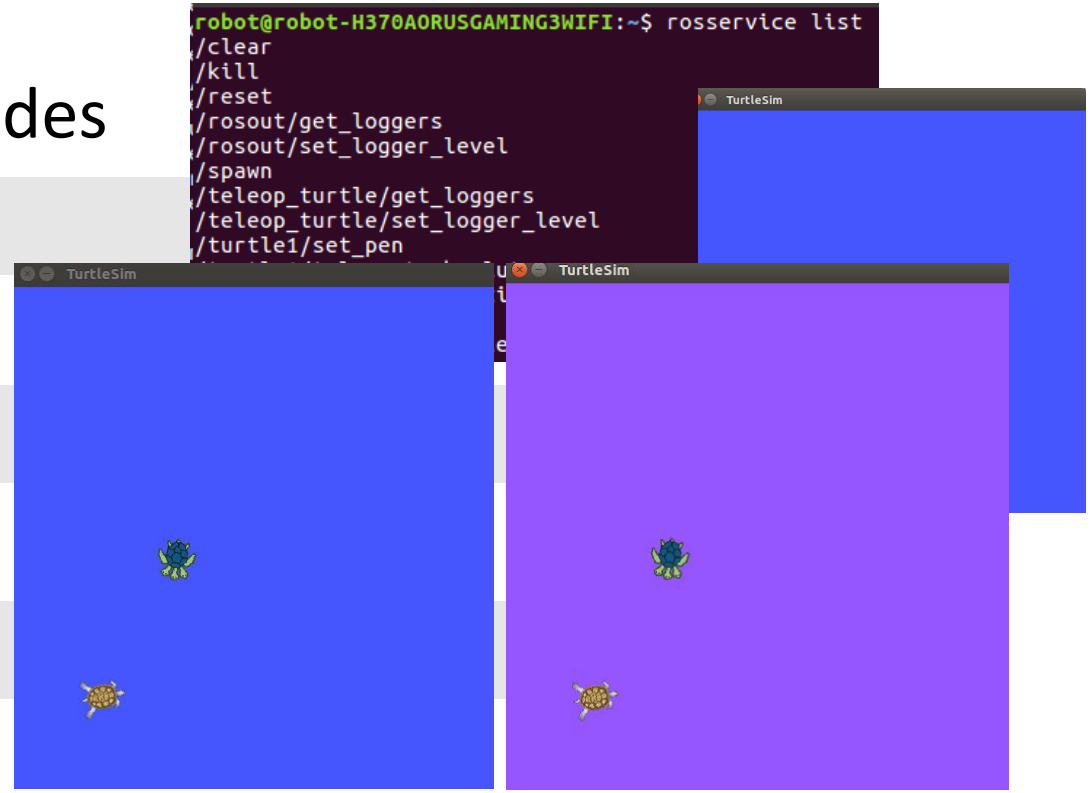
```
$ rosservice call /clear
```

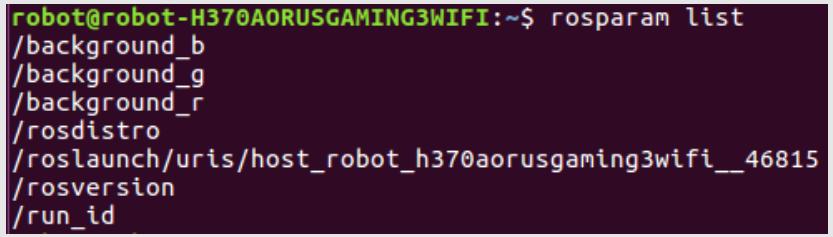
- Produce another turtle at (2, 2, 0.2)

```
$ rosservice call /spawn 2 2 0.2 ""
```

- Change the background color

```
$ rosparam list
$ rosparam set /background_r 150
$ rosservice call /clear
```





Example2-2: Turtlesim

- **Keyboard teleoperation**

```
$ rosrun turtlesim turtle_teleop_key
```

- **Run a launch file**

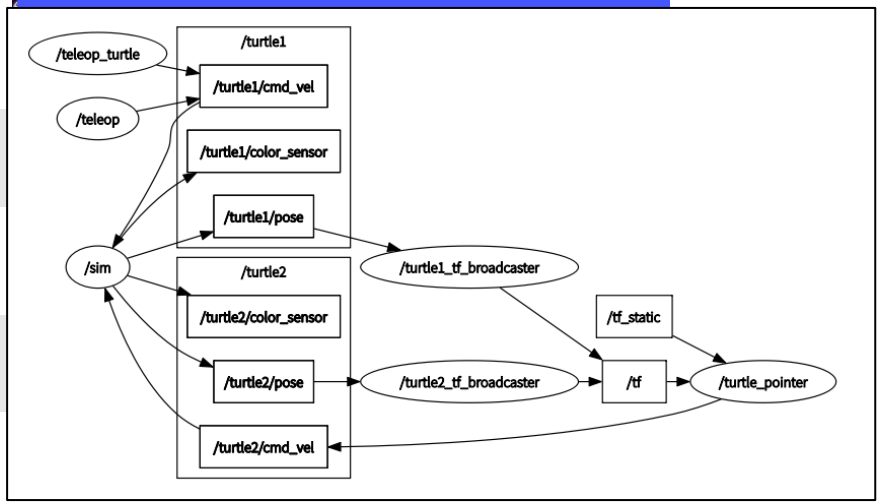
```
$ roslaunch turtle_tf turtle_tf_demo.launch
```

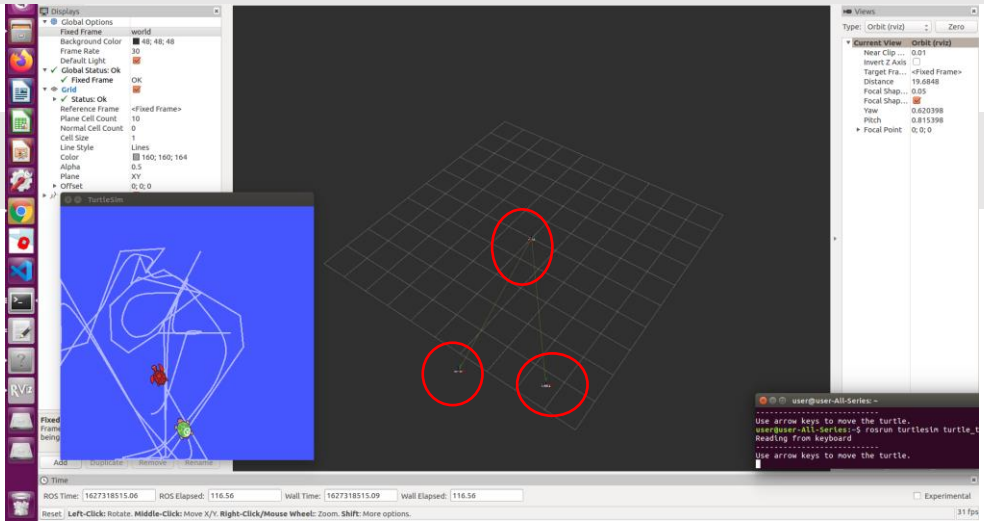
- **Show rqt_graph**

```
$ rqt_graph
```

- **Open Rviz**

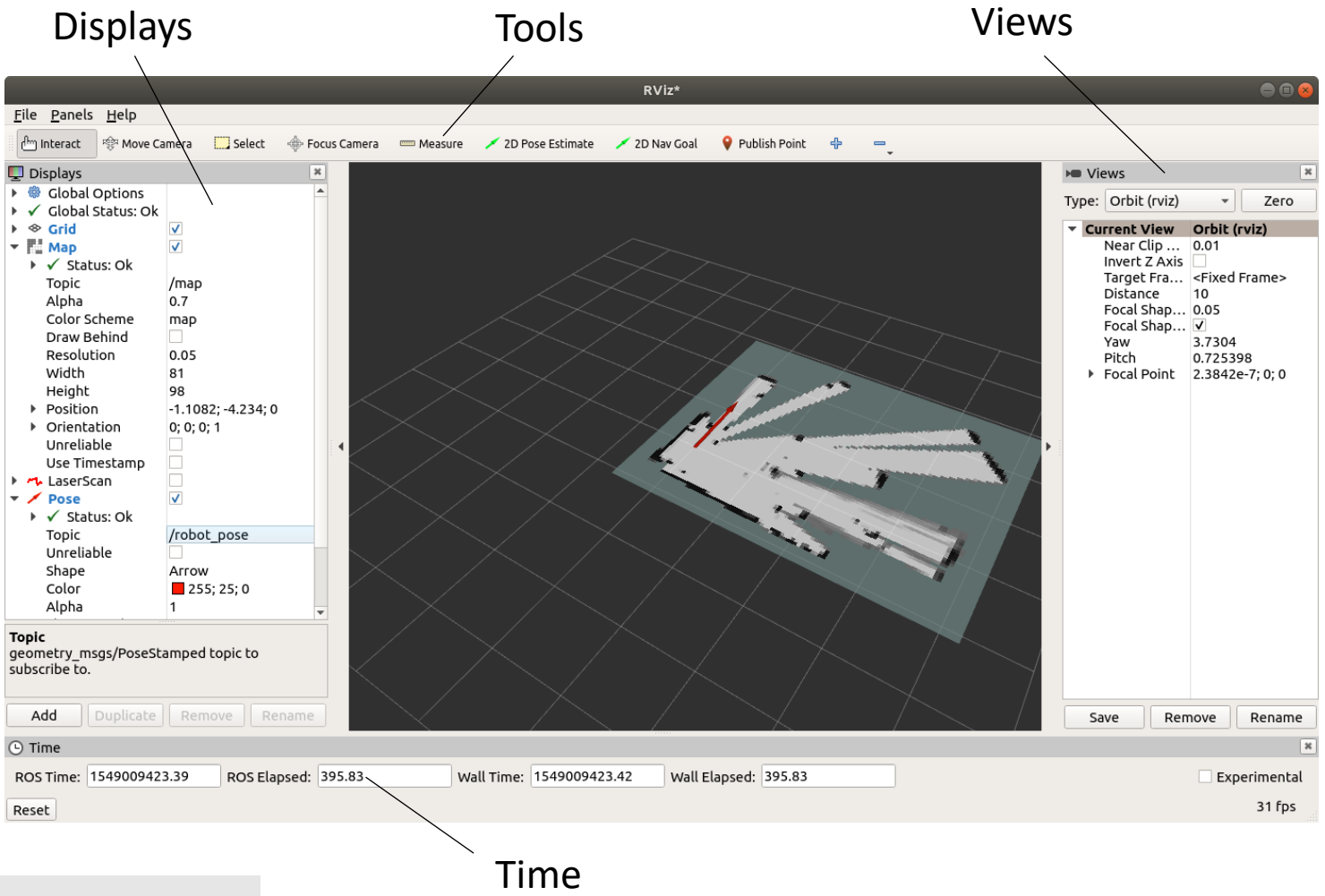
```
$ rviz
```





RViz

- 3D localization tool for ROS
- Subscriber to topic and visualizes the message contents
- Interactive tools to publish user information
- Example: use rviz to see the camera view

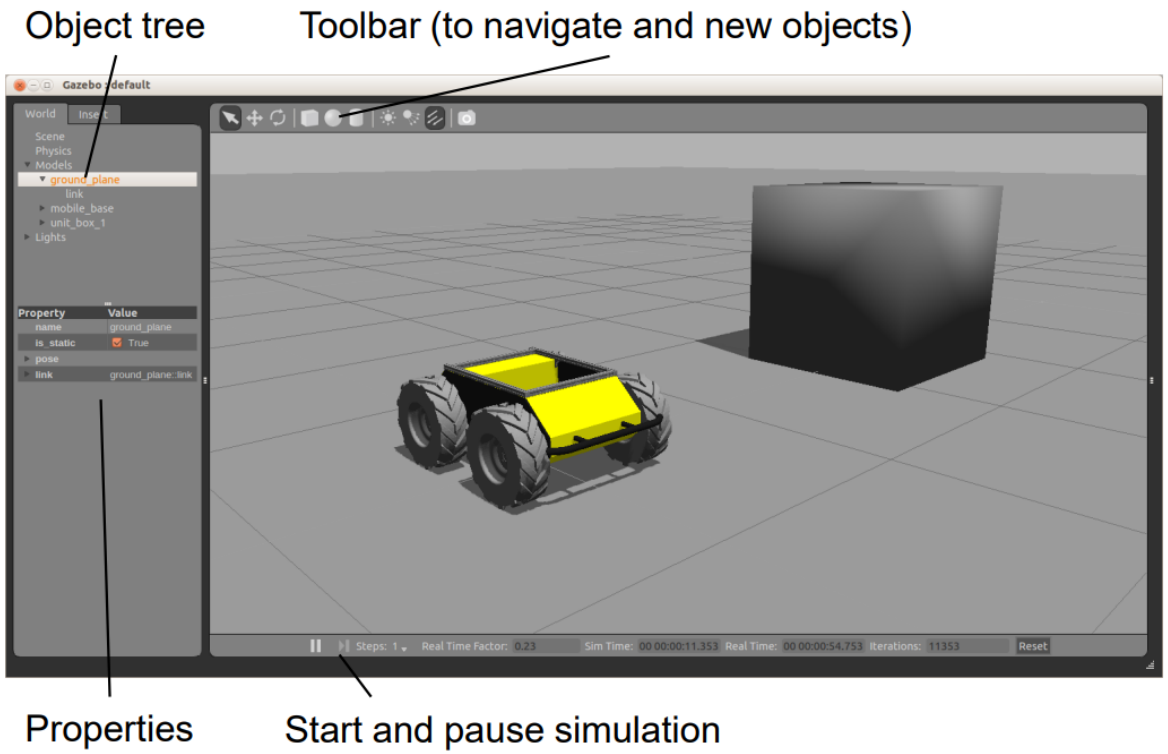



```
$ roslaunch usb_cam usb_cam-test.launch
$ rosrun rviz rviz
```

Gazebo Simulator

- Simulate 3d rigid-body dynamics
- Simulate a variety of sensors
- 3d visualization and user interaction
- Includes a database of many robots and environments (Gazebo worlds)
- Extensible with plugins
- Run Gazebo with

```
$ rosrun gazebo_ros gazebo
```



Outline

- Introduction
- ROS File System
- Example1: Publisher and Subscriber
- Example2: Turtlesim
- Something Useful

How To Install Other's Packages

- Install from apt

  The package will be installed in /opt/ros/kinetic/share

```
$ sudo apt install ros-kinetic-<package_name>
```

- Install from source (http://wiki.ros.org/usb_cam)

```
$ cd ~/catkin_ws/src
$ git clone http://github.com/ros-drivers/usb_cam.git
$ cd ~/catkin_ws
$ catkin_make
$ roslaunch usb_cam usb_cam-test.launch
```

Some useful packages

- Navigation(move_base, amcl, gmapping)
- rosAria(mobility revelevent)
- ros_pepper(lots of useful pepper development function)
- moveit(for inverse kinematic)

ROS Commands Cheatsheet 1/2

Command	Explanation	Description
roscore	ros+core	master
roscd	ros+cd	Move to the directory of the designated ROS package
rosls	ros+ls	Check file list of ROS package
rosrun	ros +run	Run node
roslaunch	ros + launch	Launch multiple nodes and configure options
rostopic	ros +topic	Check ROS topic information
rosservice	ros +service	Check ROS service information

ROS Commands Cheatsheet 2/2

Command	Explanation	Description
rosnode	ros + node	Check ROS node information
rosparam	ros + parameter	Check and edit ROS parameter information
rosbag	ros + bag	Record and play ROS message
rosmsg	ros + msg	Check ROS message information
rospack	ros + pack	View information regarding a specific ROS package
catkin_create_pkg	create package	Automatic creation of package
Catkin_make	make the file in ws	Build based on catkin build system

Some Useful links

ROS wiki

- http://wiki.ros.org/

ROS tutorials

- http://wiki.ros.org/ROS/Tutorials

Available packages for kinetic

- http://repositories.ros.org/status_page/ros_kinetic_default.html

ROS Course

- https://rsl.ethz.ch/education-students/lectures/ros.html

Some Useful links

ROS Cheat Sheet

- https://kapeli.com/cheat_sheets/ROS.docset/Contents/Resources/Documents/index

ROS Best Practices

- https://github.com/leggedrobotics/ros_best_practices/wiki

ROS Package Template

- https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template