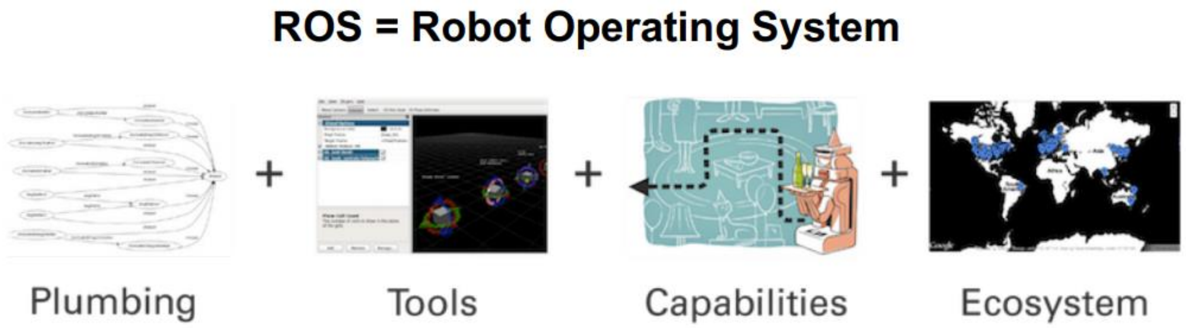
ROS Introduction

Hao-Yun Chen

2020.08.11

What is ROS?



- Process management
- Inter-process communication
- Device drivers

- Simulation
- Visualization
- Graphical user interface
- Data logging

- Control
- Planning
- Perception
- Mapping
- Manipulation

- Package organization
- Software distribution
- Documentation
- Tutorials

ROS Features

- Peer to Peer

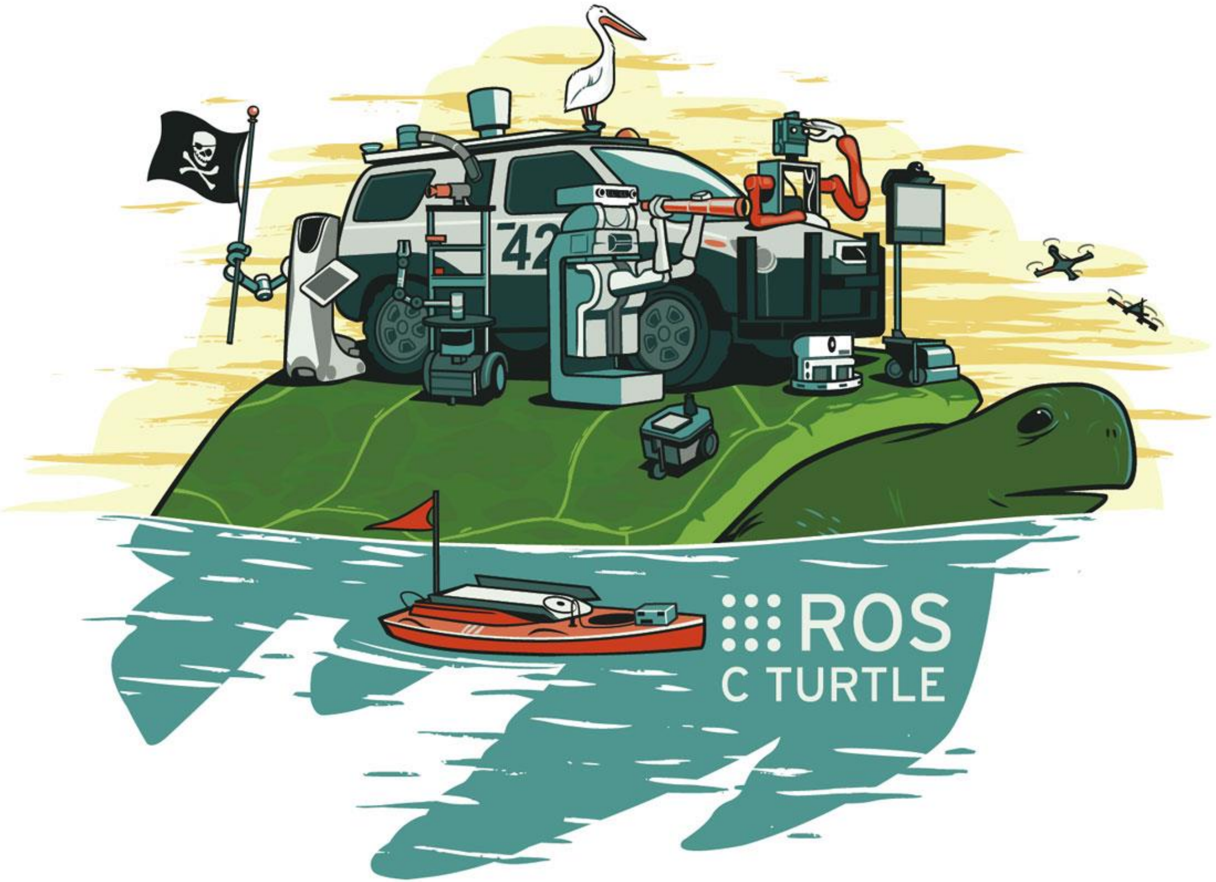
  Processes are loosely coupled and can communicate using the ROS protocol.
- Distributed

  Programs can be executed on different computer and communicate with network.
- Multi-lingual

  You can write ROS program in different languages (C++/Python/Java)
- Free and open-source

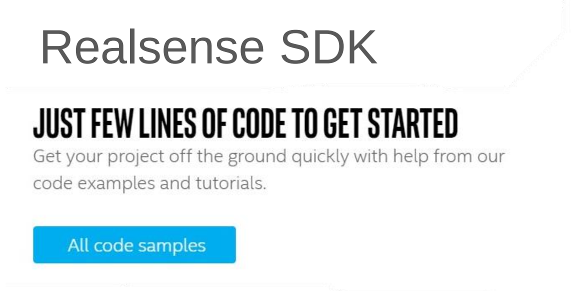
  Most of the projects using ROS are free and open-source.

Why using ROS?


42
ROS
C TURTLE

Different platforms have different SDK

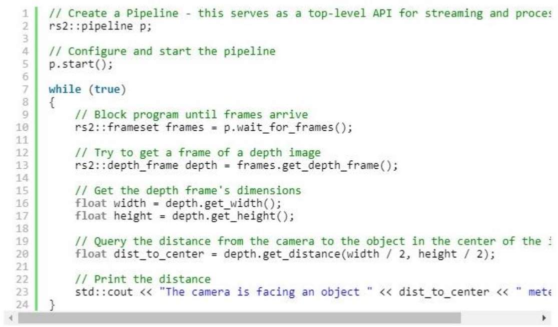
- Realsense SDK



JUST FEW LINES OF CODE TO GET STARTED

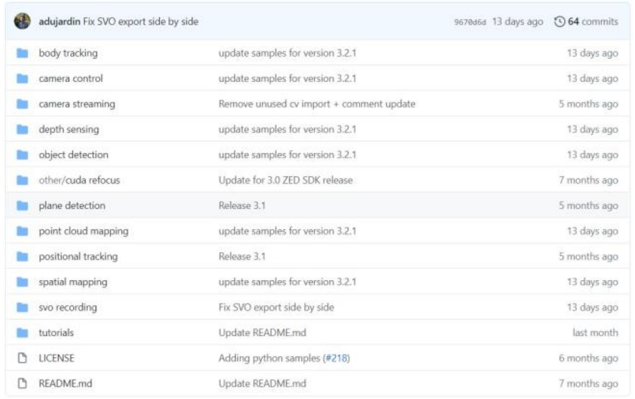
Get your project off the ground quickly with help from our code examples and tutorials.

All code samples



```
// Create a Pipeline - this serves as a top-level API for streaming and proce
rs2::pipeline p;

// Configure and start the pipeline
p.start();

while (true)
{
    // Block program until frames arrive
    rs2::frameset frames = p.wait_for_frames();

    // Try to get a frame of a depth image
    rs2::depth_frame depth = frames.get_depth_frame();

    // Get the depth frame's dimensions
    float width = depth.get_width();
    float height = depth.get_height();

    // Query the distance from the camera to the object in the center of the i
    float dist_to_center = depth.get_distance(width / 2, height / 2);

    // Print the distance
    std::cout << "The camera is facing an object " << dist_to_center << " mete
}
```

- Zed camera SDK

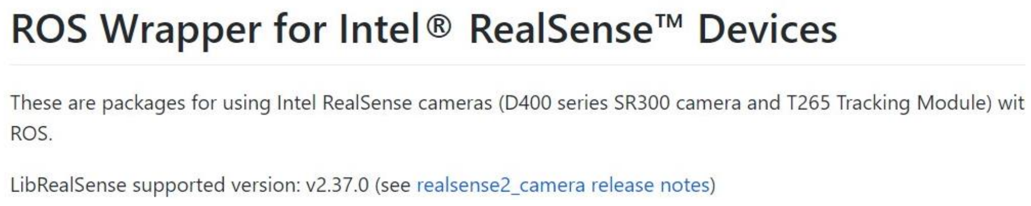


adujardin Fix SVO export side by side 9670d6d 13 days ago 64 commits

body tracking	update samples for version 3.2.1	13 days ago
camera control	update samples for version 3.2.1	13 days ago
camera streaming	Remove unused cv import + comment update	5 months ago
depth sensing	update samples for version 3.2.1	13 days ago
object detection	update samples for version 3.2.1	13 days ago
other/cuda refocus	Update for 3.0 ZED SDK release	7 months ago
plane detection	Release 3.1	5 months ago
point cloud mapping	update samples for version 3.2.1	13 days ago
positional tracking	Release 3.1	5 months ago
spatial mapping	update samples for version 3.2.1	13 days ago
svo recording	Fix SVO export side by side	13 days ago
tutorials	Update README.md	last month
LICENSE	Adding python samples (#218)	6 months ago
README.md	Update README.md	7 months ago

Lots of them have ROS wrapper

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

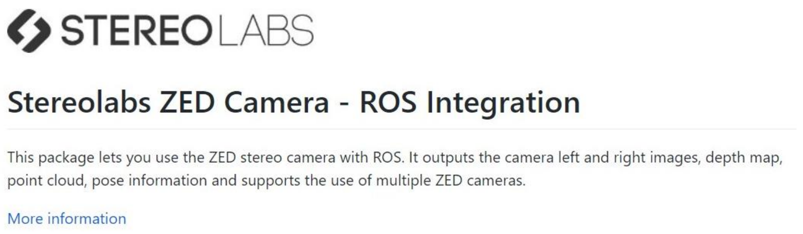


ROS Wrapper for Intel® RealSense™ Devices

These are packages for using Intel RealSense cameras (D400 series SR300 camera and T265 Tracking Module) wit ROS.

LibRealSense supported version: v2.37.0 (see realsense2_camera release notes)

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



STEREOLABS

Stereolabs ZED Camera - ROS Integration

This package lets you use the ZED stereo camera with ROS. It outputs the camera left and right images, depth map, point cloud, pose information and supports the use of multiple ZED cameras.

More information

Write different code for different platforms

Use ROS to deploy same code on all platforms

ROS File System

ROS package

- 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.*

- Install ROS tutorial packages:

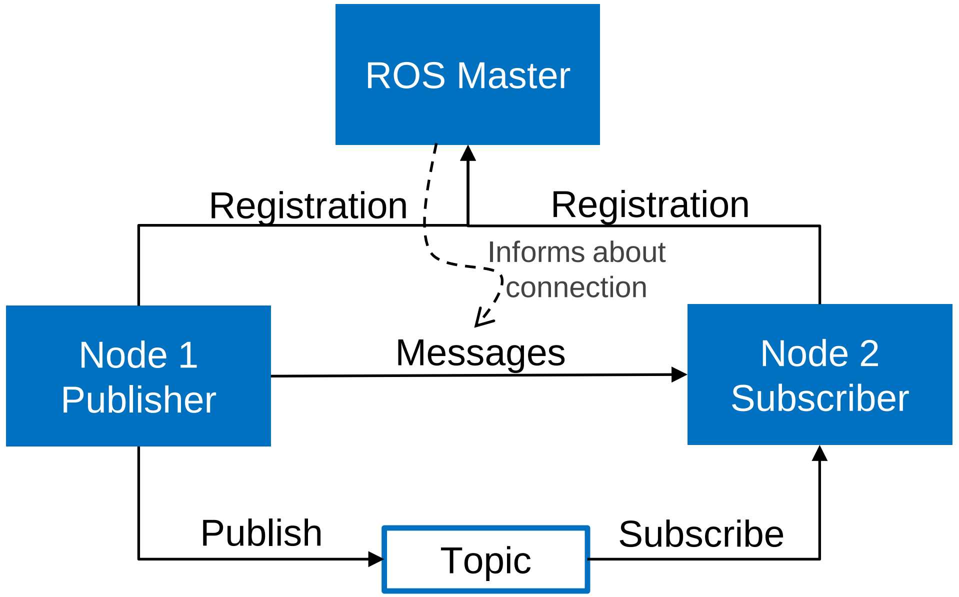
```
$ sudo apt install ros-kinetic-ros-tutorial
```

- Find the package location:

```
$ roscd rospy_tutorial
$ pwd
/opt/ros/kinetic/share/rospy_tutorials
```

ROS node concept

ROS node



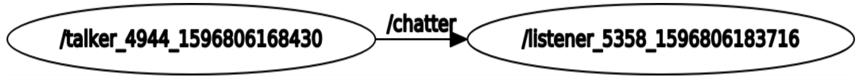
- Node: an executable that uses ROS to communicate with other nodes.
- Topic: Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.
- Master: Name service for ROS (i.e. helps nodes find each other)

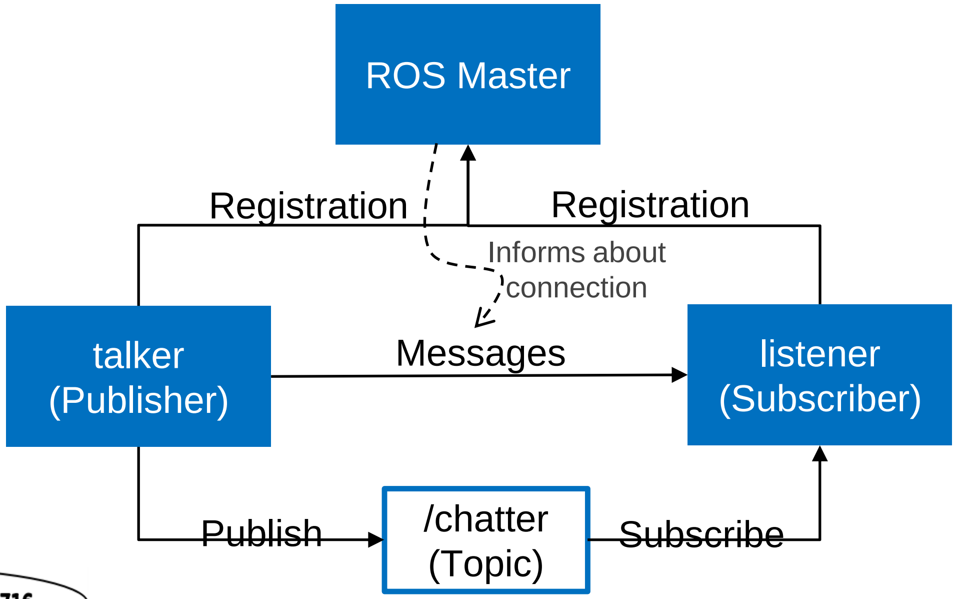
ROS node

- Run a node with
  `$ rosrun <package_name> <node_name>`
- See active node with
  `$ rosnode list`
- See node information
  `$ rosnode info <node_name>`

Run a simple ROS publisher/subscriber

1. Start the ROS master
   `$ roscore`
2. Run publisher node
   `$ rosrun rospy_tutorials talker`
3. Run subscriber node
   `$ rosrun rospy_tutorials listener`
4. Visualization
   `$ rqt_graph`





Hand-on writing publisher/subscriber

Create ROS workspace

- Create your ROS workspace
  ```
  $ mkdir -p ~/catkin_ws/src
  $ cd ~/catkin_ws
  ```
- You can still compile with empty workspace
  ```
  $ catkin_make
  ```
  You should have a 'build' and 'devel' folder under 'catkin_ws'
- 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

- Enter the src folder in your workspace

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

- Create a package

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

- Compile

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

- Source your environment

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

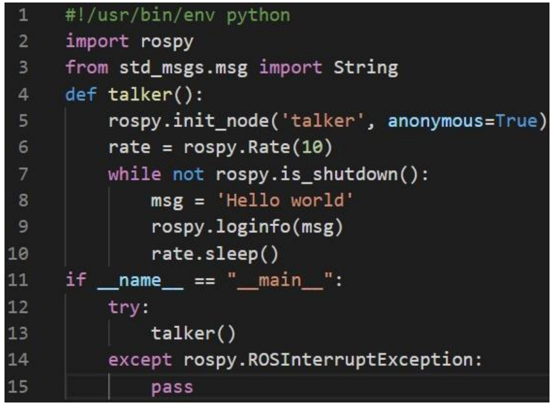
- Enter your package folder

```
$ roscd beginner_tutorials
```

Create your own publisher/subscriber

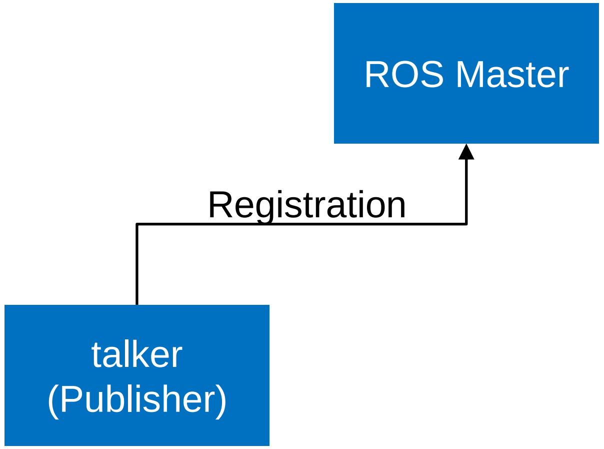
- Create a talker node

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



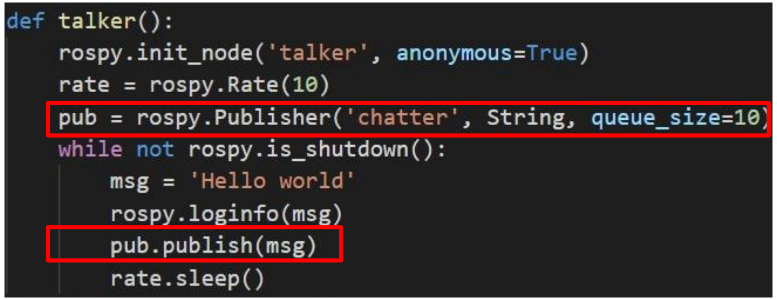
```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
def talker():
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10)
    while not rospy.is_shutdown():
        msg = 'Hello world'
        rospy.loginfo(msg)
        rate.sleep()
if __name__ == "__main__":
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

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



Create your own publisher/subscriber

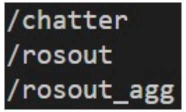
- Write a publisher



```python
def talker():
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10)
    pub = rospy.Publisher('chatter', String, queue_size=10)
    while not rospy.is_shutdown():
        msg = 'Hello world'
        rospy.loginfo(msg)
        pub.publish(msg)
        rate.sleep()
```

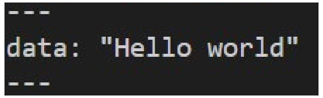
```
$ rosrun beginner_tutorials talker.py
```

```
$ rostopic list
```

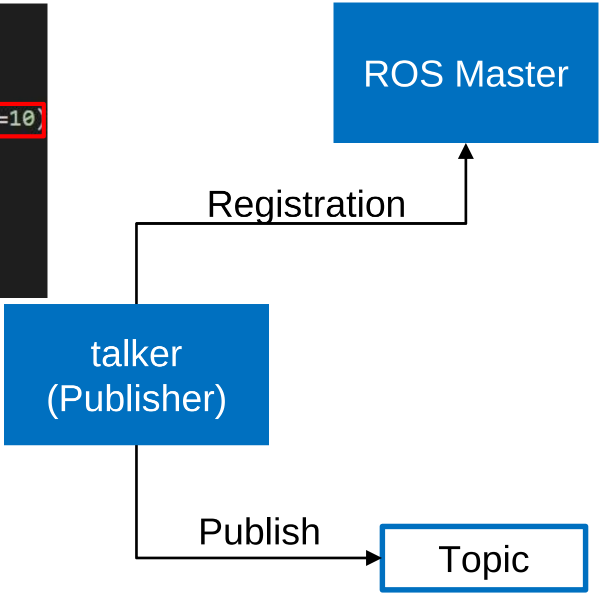


```
/chatter
/rosout
/rosout_agg
```

```
$ rostopic echo /chatter
```



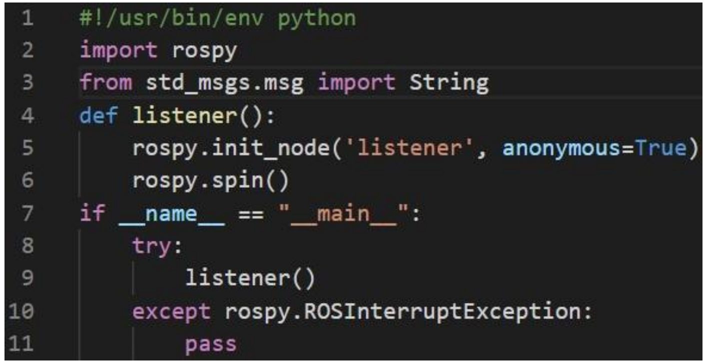
```
---
data: "Hello world"
---
```



Create your own publisher/subscriber

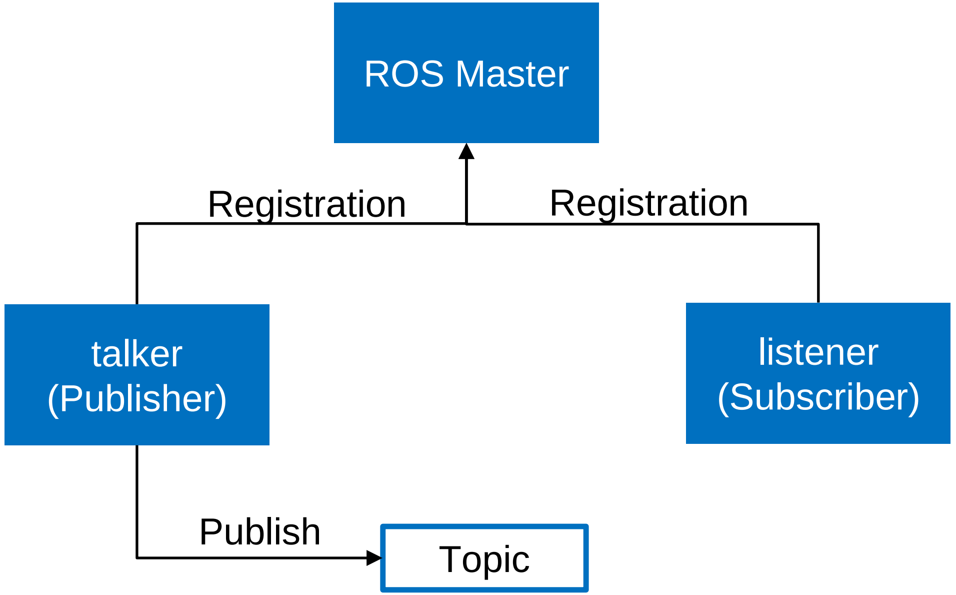
- Create a listener node

```
$ roscd beginner_tutorials/scripts
$ code listener.py
$ sudo chmod +x listener.py
```



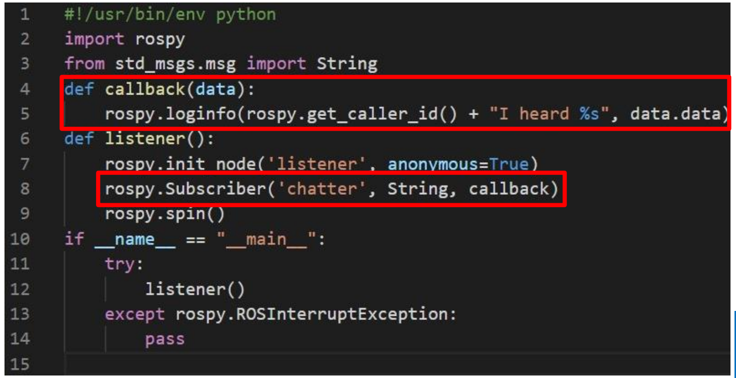
```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.spin()
if __name__ == "__main__":
    try:
        listener()
    except rospy.ROSInterruptException:
        pass
```

```
$ rosrun beginner_tutorials listener.py
```



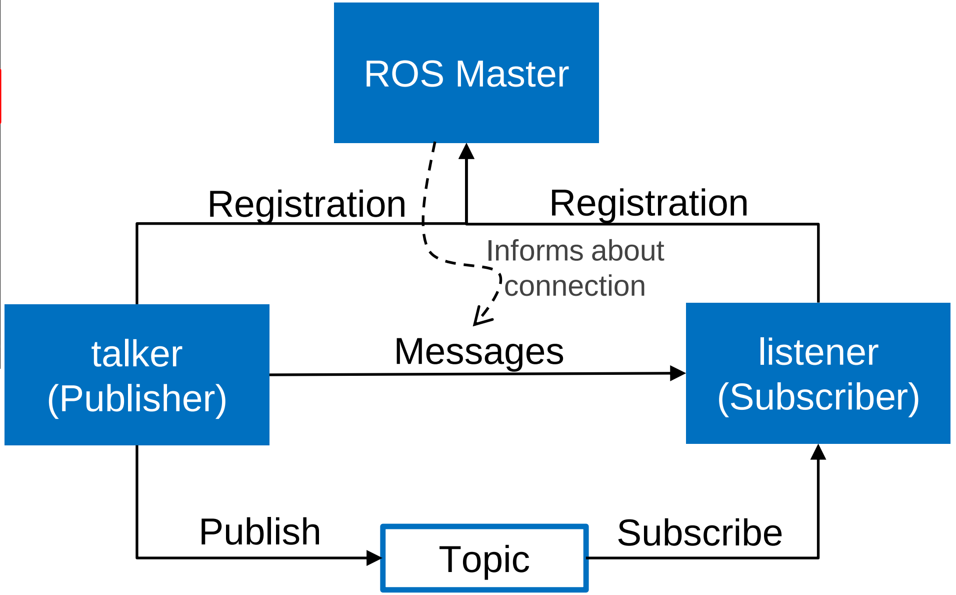
Create your own publisher/subscriber

- Write a subscriber



```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)
def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber('chatter', String, callback)
    rospy.spin()
if __name__ == "__main__":
    try:
        listener()
    except rospy.ROSInterruptException:
        pass
```

```
$ rosrun beginner_tutorials listener.py
$ rqt_graph
```

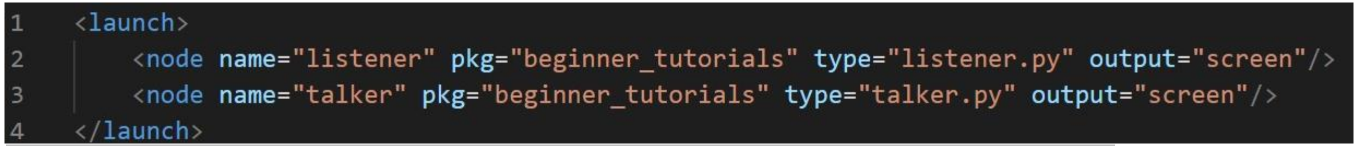


Roslaunch

- Roslaunch is a tool for easily launching multiple ROS nodes

- Write a simple launch file

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



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

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

How to install other’s packages

1. Install from apt

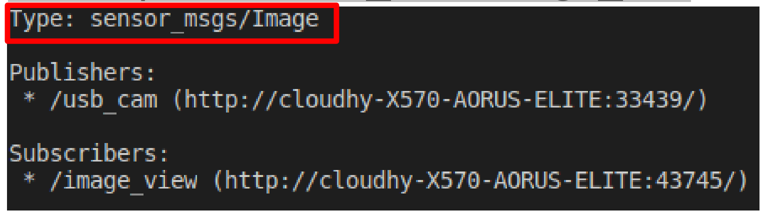
   e.g., `sudo apt install ros-kinetic-<package_name>`

   The package will be installed in `/opt/ros/kinetic/share`
2. Install from source (http://wiki.ros.org/usb_cam)

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

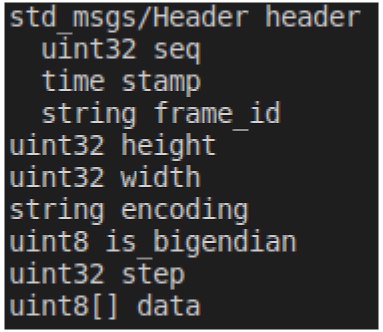
Find message type of packages

- Find the message type:
  $ rostopic info /usb_cam/image_raw



```
Type: sensor_msgs/Image

Publishers:
 * /usb_cam (http://cloudhy-X570-AORUS-ELITE:33439/)

Subscribers:
 * /image_view (http://cloudhy-X570-AORUS-ELITE:43745/)
```

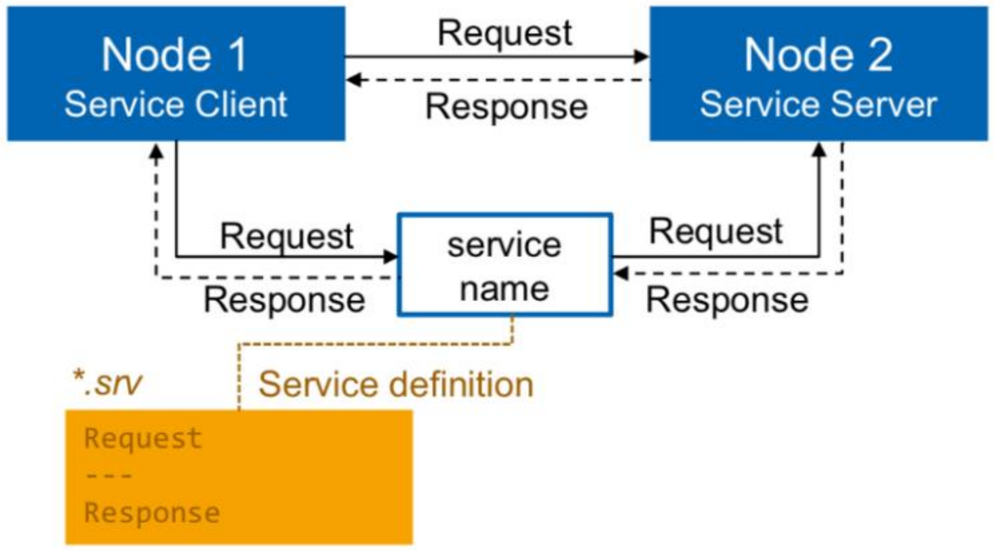
- Find the data structure of message:
  1. Google it (http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html)
  2. $ rosmsg info sensor_msgs/Image



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

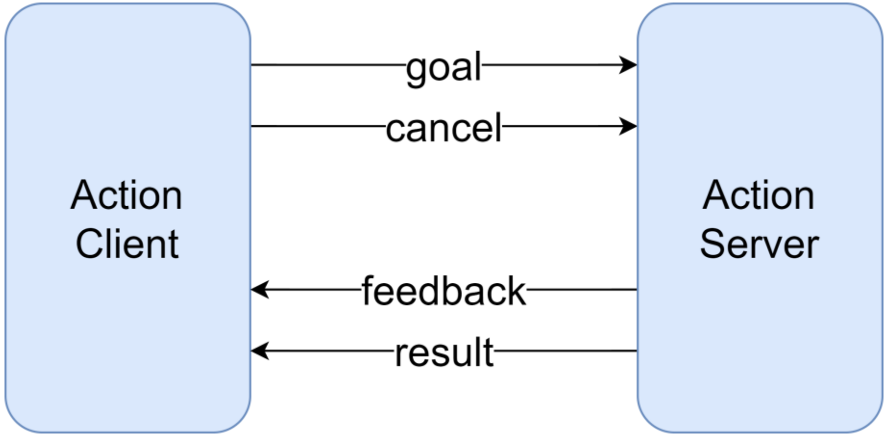
Service

- Synchronized message communication
- Unlike the topic, the service is one time message communication, which is useful to reduce the load.



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.



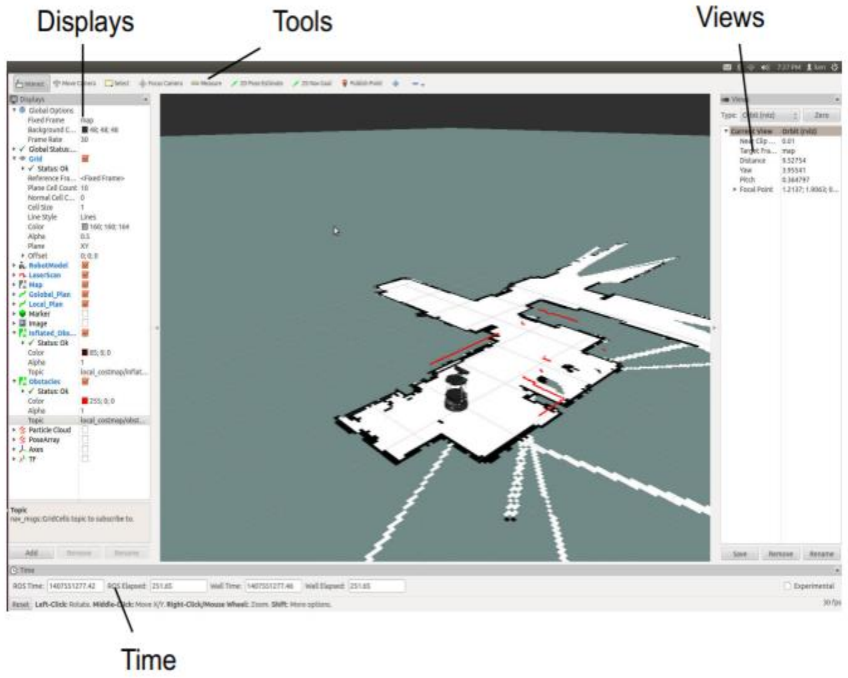
http://wiki.ros.org/actionlib_tutorials/Tutorials

RViz

- 3D visualization tool for ROS
- Subscribes to topic and visualizes the message contents
- Interactive tools to publish user information

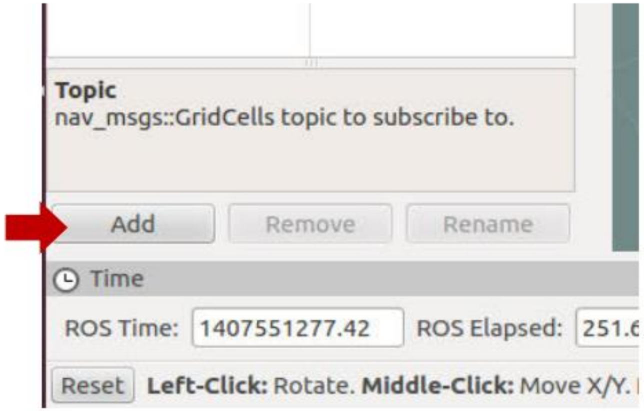
- Use Rviz to see the camera view

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



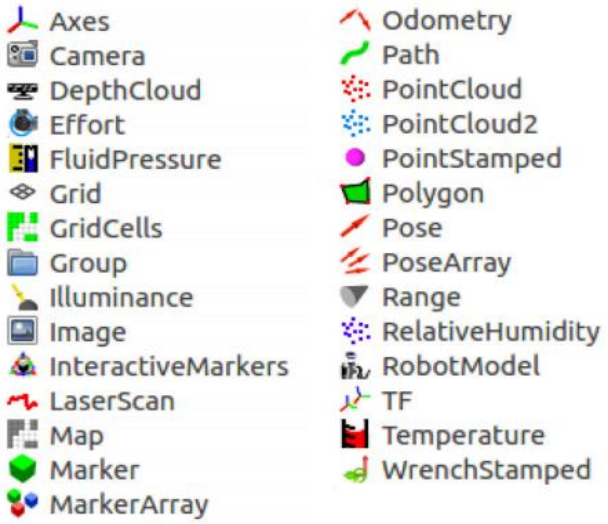
RViz

Display Plugins


Topic
nav_msgs::GridCells topic to subscribe to.

ROS Time: 1407551277.42 ROS Elapsed: 251.6

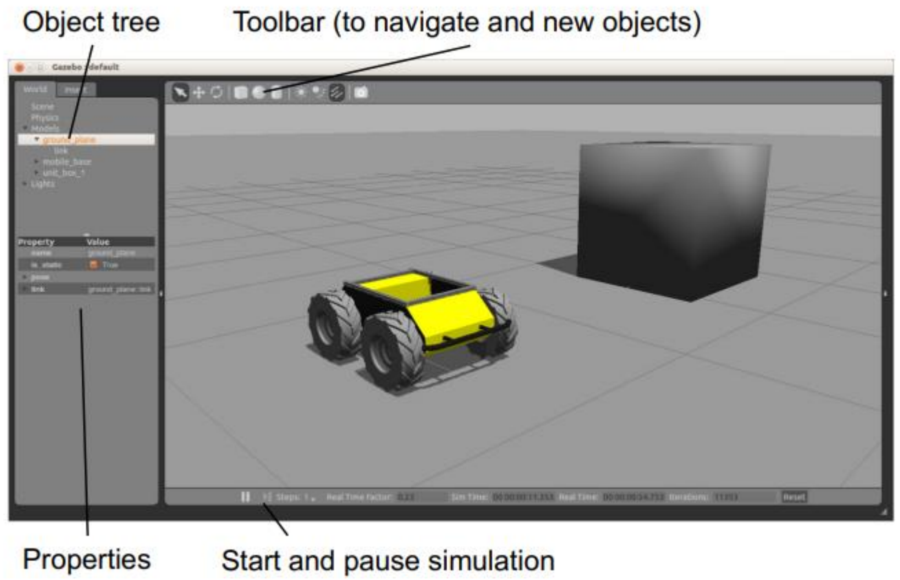
Left-Click: Rotate. **Middle-Click:** Move X/Y.



Gazebo

- Simulat 3d rigid-body dynamics
- Simulate a varity of sensors
- 3d visualization and user interaction
- Includes data of many robots and en
- Provides ROS interface

- Run with
  $ rosrun gazebo_ros gazebo



Some Useful packages

1. Navigation(move_base, amcl, gmapping)

2. rosAria(mobility revelevent, for your robotic class)

3. ros_pepper(lots of useful pepper development function)

4. moveit(for inverse kinematic)

ROS Commands Cheatsheet

Command	Command Explanation	Description
roscore	ros + core	master(ROS name service)
roscd	ros + cd(change directory)	Move to the directory of the desgnated ROS package
rosls	ros + ls(lists files)	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
rosnode	ros + node	Check ROS node information
rosparam	ros + parameter	Check and edit ROS parameter information

ROS Commands Cheatsheet (Cont.)

Command	Command Explanation	Description
rosbag	ros + bag	Record and play ROS message
rosmsg	ros + msg	Check ROS message information
rospack	ros + package	View information regarding a specific ROS package
catkin_create_pkg	create ros package	Automatic creation of package
catkin_make	make the file in the catkin_ws	Build based on catkin build system

Some Useful links

- ROS Wiki
  http://wiki.ros.org/
- Tutorials
  http://wiki.ros.org/ROS/Tutorials
- Available packages
  http://www.ros.org/browse/
- ROS Course
  http://www.rsl.ethz.ch/education-students/lectures/ros.html