Credit to Markus Grotz, Joshua Smith and others on the EE 545 staff

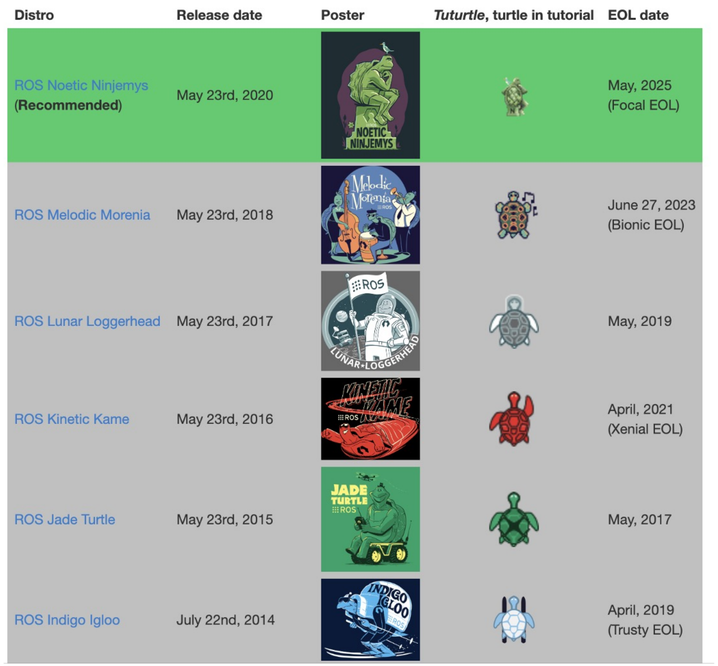
# Introduction to ROS

Slides adapted from: [http://www.rsl.ethz.ch/education-students/lectures/ros.html](http://www.rsl.ethz.ch/education-students/lectures/ros.html)

UNIVERSITY *of* WASHINGTON

## ROS Terminology

> ROS versions are identified by name
  - first letter of name increments with each new version…
> In this class, we are using ROS Melodic (latest version is Noetic)
> Previous versions were Lunar and Kinetic



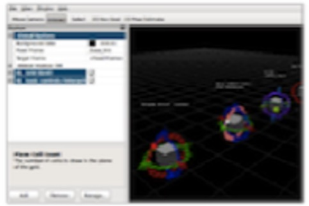
| Distro | Release date | Poster | *Tuturtle*, turtle in tutorial | EOL date |
|---|---|---|---|---|
| ROS Noetic Ninjemys **(Recommended)** | May 23rd, 2020 | | | May, 2025 (Focal EOL) |
| ROS Melodic Morenia | May 23rd, 2018 | | | June 27, 2023 (Bionic EOL) |
| ROS Lunar Loggerhead | May 23rd, 2017 | | | May, 2019 |
| ROS Kinetic Kame | May 23rd, 2016 | | | April, 2021 (Xenial EOL) |
| ROS Jade Turtle | May 23rd, 2015 | | | May, 2017 |
| ROS Indigo Igloo | July 22nd, 2014 | | | April, 2019 (Trusty EOL) |

## ROS 1 vs. ROS 2

- Currently ROS vs ROS2
- Major difference
  - Single base library for C++ and Python
  - No roscore
  - Services are now asynchronous
  - Quality of Service (QoS)

## What is ROS?

**ROS = Robot Operating System**

+

+

+

ros.org

| Plumbing | Tools | Capabilities | Ecosystem |
|---|---|---|---|

**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

## History of ROS

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory
- Since 2013 managed by OSRF
- Today used by many robots,
- universities and companies
- De facto standard for robot programming

ros.org

UNIVERSITY *of* WASHINGTON

## ROS Philosophy

- Peer to peer
  - Individual programs communicate over defined API (ROS messages, services, etc.).
- Distributed
  - Programs can be run on multiple computers and communicate over the network.
- Multi-lingual
  - ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
- Light-weight
  - Stand-alone libraries are wrapped around with a thin ROS layer.
- Free and open-source
  - Most ROS software is open-source and free to use.

## ROS Nodes

> Single-purpose, executable program
> Individually compiled, executed, andmanaged
> Organized in packages

Run a node with

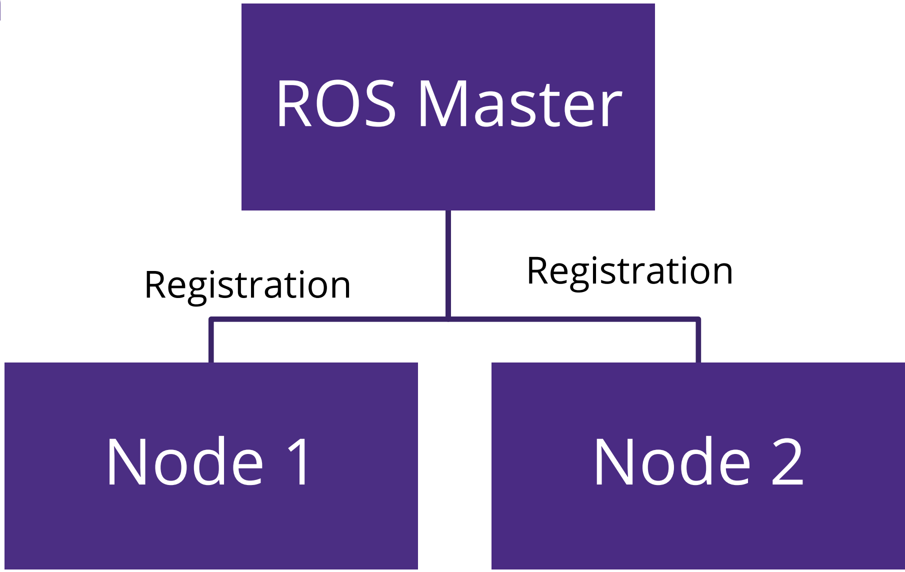
➔ `rosrun` *package_name node_name*

See active node list

➔ `rosnode` *list*

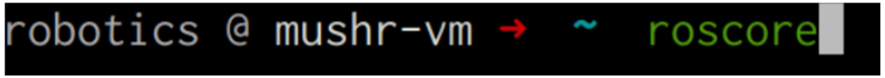
Retrieve information about a node with

➔ `rosnode` *info node_name*

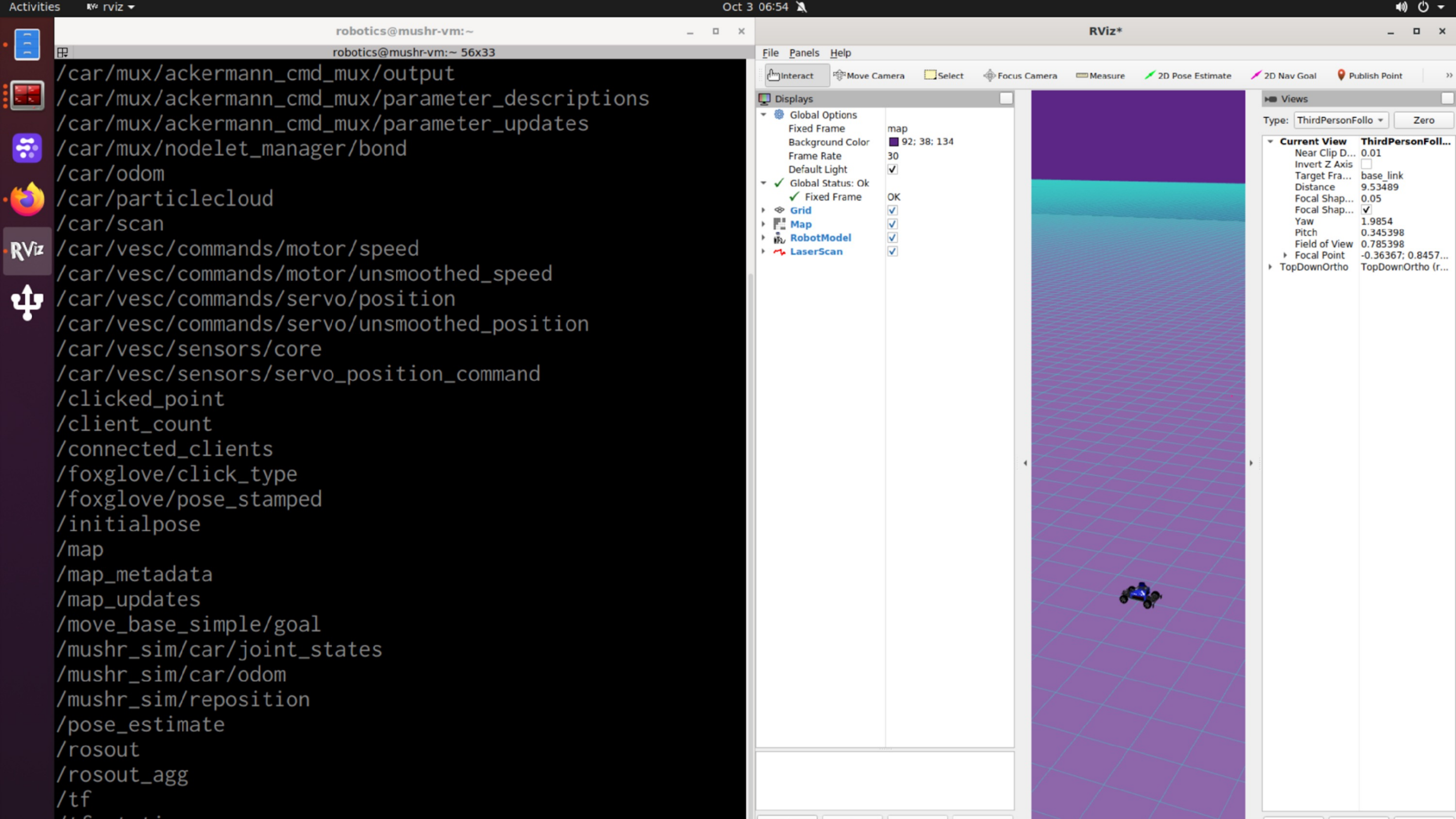


## ROS Master

- Manages the communication between nodes (processes)
- Every node registers at startup with the master
- No longer required in ROS2



```
robotics @ mushr-vm ➜  ~  roscore
```



```
/car/mux/ackermann_cmd_mux/output
/car/mux/ackermann_cmd_mux/parameter_descriptions
/car/mux/ackermann_cmd_mux/parameter_updates
/car/mux/nodelet_manager/bond
/car/odom
/car/particlecloud
/car/scan
/car/vesc/commands/motor/speed
/car/vesc/commands/motor/unsmoothed_speed
/car/vesc/commands/servo/position
/car/vesc/commands/servo/unsmoothed_position
/car/vesc/sensors/core
/car/vesc/sensors/servo_position_command
/clicked_point
/client_count
/connected_clients
/foxglove/click_type
/foxglove/pose_stamped
/initialpose
/map
/map_metadata
/map_updates
/move_base_simple/goal
/mushr_sim/car/joint_states
/mushr_sim/car/odom
/mushr_sim/reposition
/pose_estimate
/rosout
/rosout_agg
/tf
```

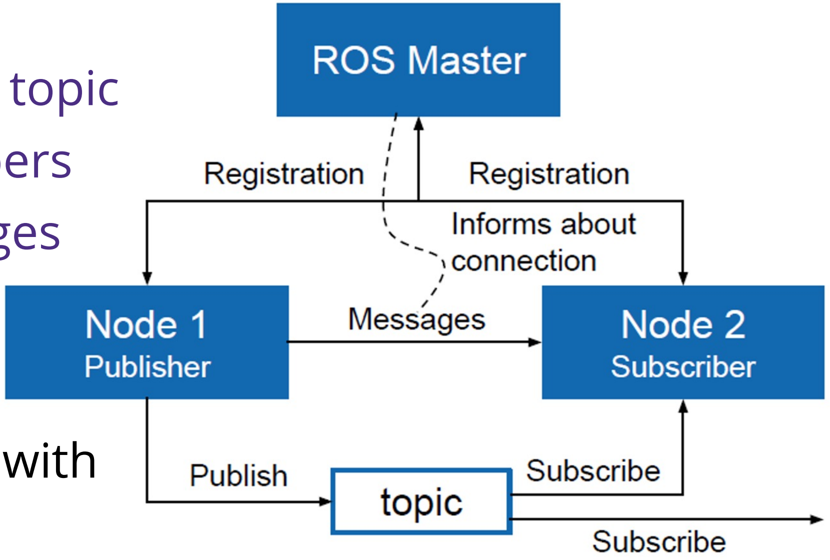
| | |
|---|---|
| Global Options | |
| Fixed Frame | map |
| Background Color | 92; 38; 134 |
| Frame Rate | 30 |
| Default Light | ✓ |
| Global Status: Ok | |
| Fixed Frame | OK |
| Grid | ✓ |
| Map | ✓ |
| RobotModel | ✓ |
| LaserScan | ✓ |

Type: ThirdPersonFollo

| | |
|---|---|
| Current View | ThirdPersonFoll... |
| Near Clip D... | 0.01 |
| Invert Z Axis | |
| Target Fra... | base_link |
| Distance | 9.53489 |
| Focal Shap... | 0.05 |
| Focal Shap... | ✓ |
| Yaw | 1.9854 |
| Pitch | 0.345398 |
| Field of View | 0.785398 |
| Focal Point | -0.36367; 0.8457... |
| TopDownOrtho | TopDownOrtho (r... |

## ROS Topics

- Nodes communicate over topics
  - Nodes can publish or subscribe to a topic
  - Typically, 1 publisher and n subscribers
- Topic is a name for a stream of messages



List active topics with

➔ `rostopic *list*`

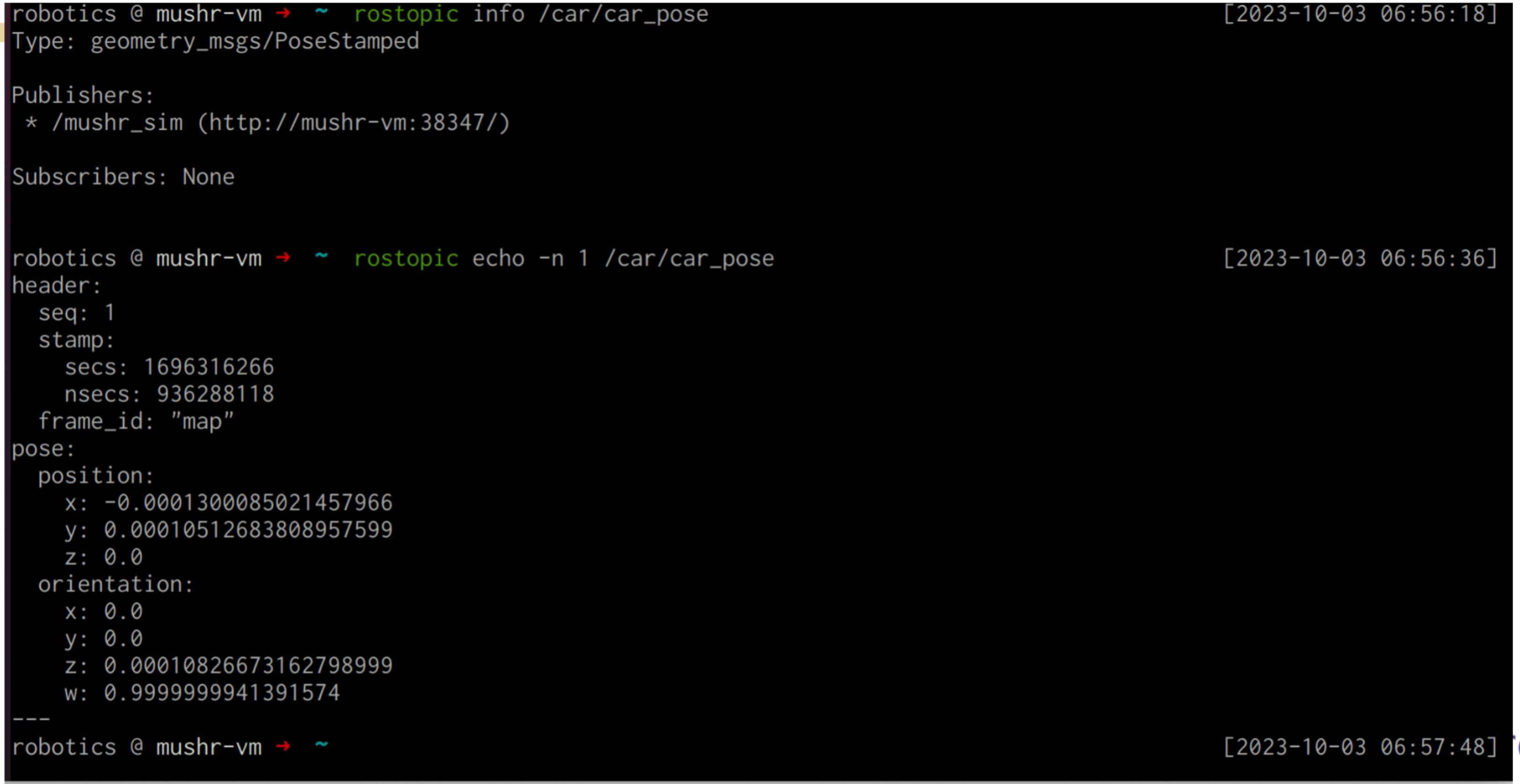
Subscribe and print the contents of a topic with

➔ `rostopic *echo /topic*`

Show information about a topic with

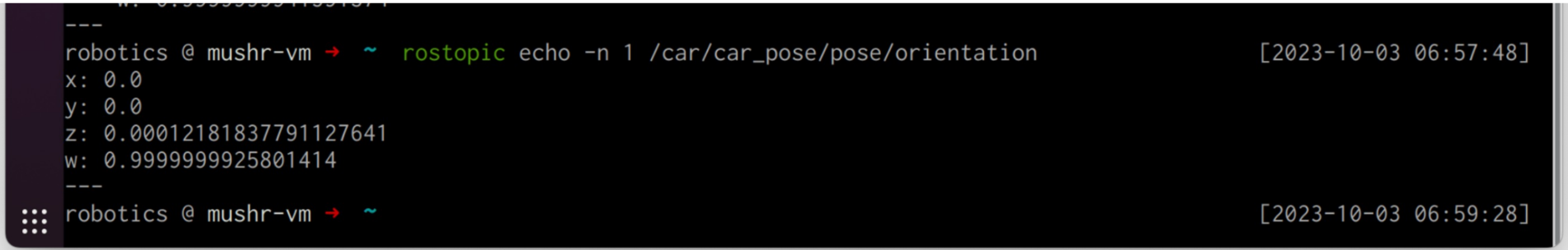
➔ `rostopic info /topic`

# rostopic: info + echo



```
robotics @ mushr-vm ➜ ~  rostopic info /car/car_pose                          [2023-10-03 06:56:18]
Type: geometry_msgs/PoseStamped

Publishers:
 * /mushr_sim (http://mushr-vm:38347/)

Subscribers: None


robotics @ mushr-vm ➜ ~  rostopic echo -n 1 /car/car_pose                     [2023-10-03 06:56:36]
header:
  seq: 1
  stamp:
    secs: 1696316266
    nsecs: 936288118
  frame_id: "map"
pose:
  position:
    x: -0.0001300085021457966
    y: 0.00010512683808957599
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.00010826673162798999
    w: 0.9999999941391574
---
robotics @ mushr-vm ➜ ~                                                       [2023-10-03 06:57:48]
```

# Note hierarchical naming of topics



```
---
robotics @ mushr-vm ➜ ~  rostopic echo -n 1 /car/car_pose/pose/orientation        [2023-10-03 06:57:48]
x: 0.0
y: 0.0
z: 0.00012181837791127641
w: 0.9999999925801414
---
robotics @ mushr-vm ➜ ~                                                            [2023-10-03 06:59:28]
```

Here we went further down the tree to display just orientation, not all the other parts of the car_pose topic

Many things in ROS use hierarchical naming
(Naming is similar to paths in a filesystem)

# ROS Messages

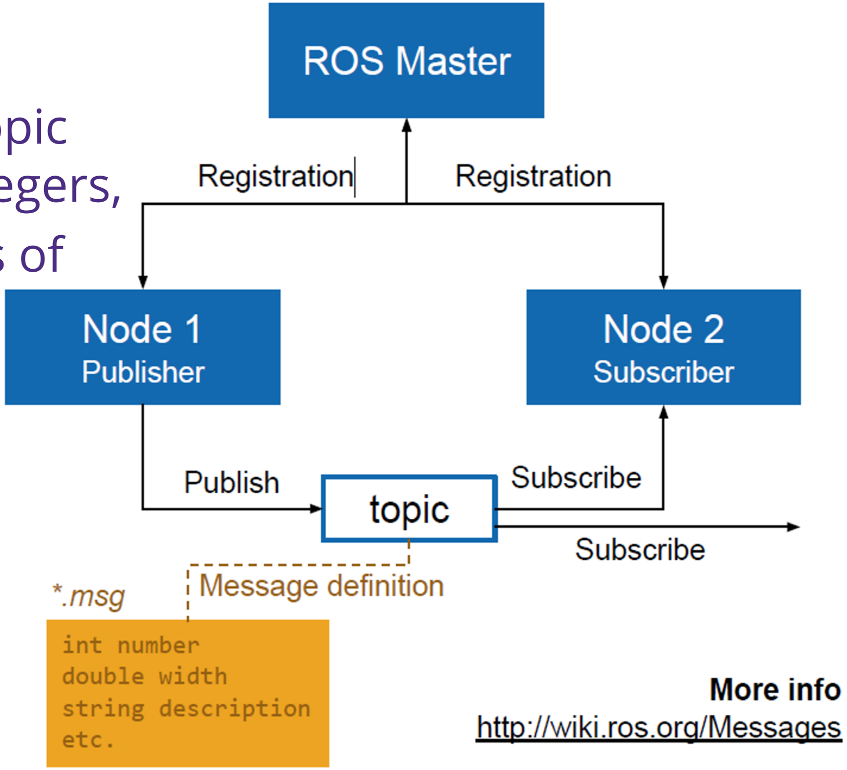
- Data structure defining the type of a topic
- Comprised of a nested structure of integers, floats, booleans, strings etc. and arrays of objects
- Defined in *.msg files

See the type of a topic

➔ rostopic *type /topic*

Publish a message to a topic

➔ rostopic *pub /topic type data*

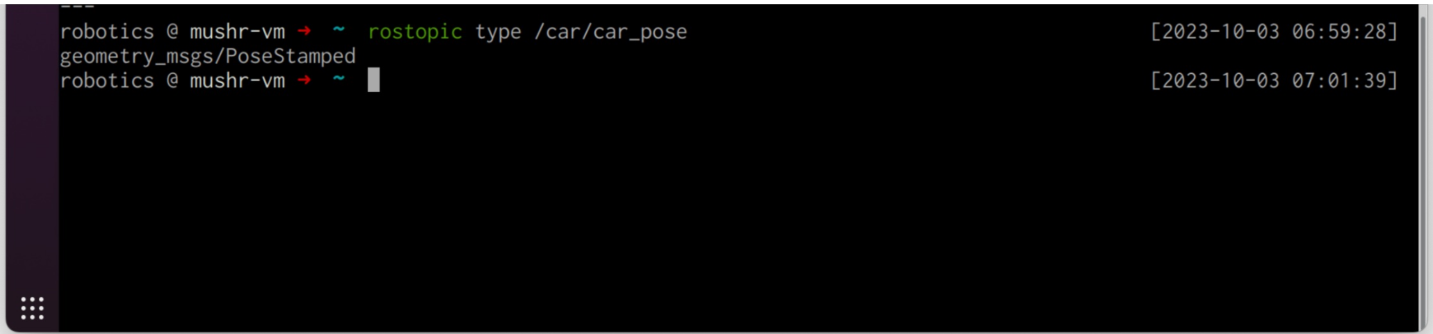


```
int number
double width
string description
etc.
```

**More info**
http://wiki.ros.org/Messages

## car_pose

> Get the message type of a topic



```
robotics @ mushr-vm ➜  ~  rostopic type /car/car_pose                    [2023-10-03 06:59:28]
geometry_msgs/PoseStamped
robotics @ mushr-vm ➜  ~                                                 [2023-10-03 07:01:39]
```

UNIVERSITY of WASHINGTON

# ROS Messages

## Pose Stamped Example

*geometry_msgs/Point.msg*

```
float64 x
float64 y
float64 z
```

*sensor_msgs/Image.msg*

```
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
```

## Some ROS Messages we will use

> geometry_msgs/PoseStamped
> sensor_msgs/LaserScan
> ackermann_msgs/AckermannDriveStamped
> geometry_msgs/Quaternion

# ROS Workspace Environment

> Defines context for the current workspace
> Default workspace loaded with

➔ `source /opt/ros/noetic/setup.bash`

Overlay your catkin workspace with

➔ `cd ~/catkin_ws`
➔ `source devel/setup.bash`

Check your workspace with

➔ `echo $ROS_PACKAGE_PATH`

This is already setup in the provided installation.

See setup with

➔ `cat ~/.zshrc`

## The catkin build  system

- catkin is the ROS build system to generate executables, libraries, and interfaces
- We suggest to use the Catkin Command Line Tools

Navigate to your catkin workspace with

➔ `cd ~/catkin_ws`

Build a package with

➔ `catkin_make`

Whenever you build a new package, update your environment

➔ `source devel/setup.bash`

The catkin command line tools are pre-installed in the provided installation.

## The catkin build system

### The catkin workspace contains the following spaces

Work here

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.

Don’t touch

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.

Don’t touch

devel

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

If necessary, clean the entire build and devel space with

➔ `catkin clean`

UNIVERSITY *of* WASHINGTON

# ROS Launch

> launch is a tool for launching multiple nodes (as well as setting parameters)
> Are written in XML as *.launch files
> If not yet running, launch automatically starts a roscore

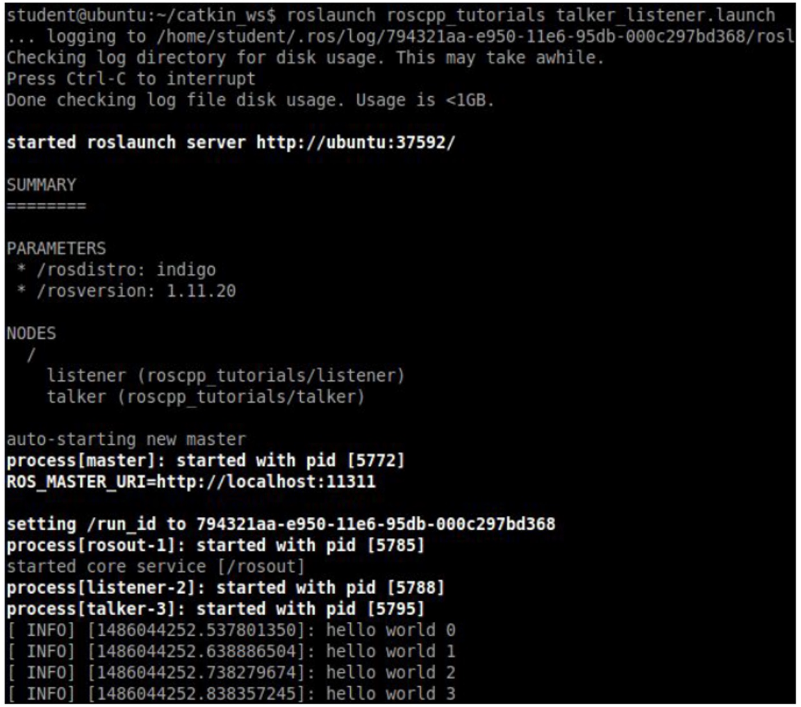
Browse to the folder and start a launch file with

➔ `roslaunch file_name.launch`

Start a launch file from a package with

➔ `roslaunch package_name file_name.launch`

Example console output for
`roslaunch roscpp_tutorials talker_listener.launch`



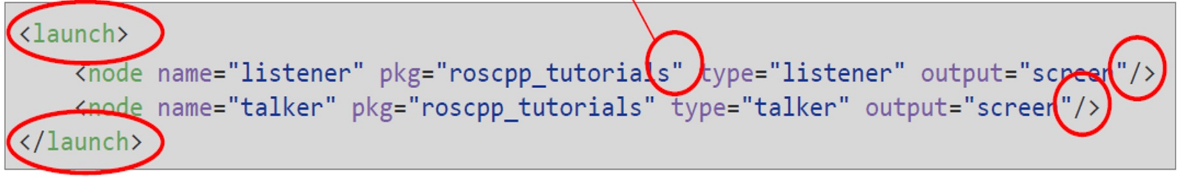
```
student@ubuntu:~/catkin_ws$ roslaunch roscpp_tutorials talker_listener.launch
... logging to /home/student/.ros/log/794321aa-e950-11e6-95db-000c297bd368/rosl
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:37592/

SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.20

NODES
  /
    listener (roscpp_tutorials/listener)
    talker (roscpp_tutorials/talker)

auto-starting new master
process[master]: started with pid [5772]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 794321aa-e950-11e6-95db-000c297bd368
process[rosout-1]: started with pid [5785]
started core service [/rosout]
process[listener-2]: started with pid [5788]
process[talker-3]: started with pid [5795]
[ INFO] [1486044252.537801350]: hello world 0
[ INFO] [1486044252.638886504]: hello world 1
[ INFO] [1486044252.738279674]: hello world 2
[ INFO] [1486044252.838357245]: hello world 3
```

# ROS Launch

## File Structure

! Attention when copy & pasting code from the internet

*talker_listener.launch*



```
<launch>
    <node name="listener" pkg="roscpp_tutorials" type="listener" output="screen"/>
    <node name="talker" pkg="roscpp_tutorials" type="talker" output="screen"/>
</launch>
```

! Notice the syntax difference for self-closing tags: `<tag></tag>` and `<tag/>`

- **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)

# ROS Launch

## Arguments

- Create re-usable launch files with <arg> tag, which works like a parameter (default optional)

```
<arg name="arg_name" default="default_value"/>
```

- Use arguments in launch file with

```
$(arg arg_name)
```

- When launching, arguments can be set with

```
> roslaunch launch_file.launch arg_name:=value
```

*range_world.launch (simplified)*

```
<?xml version="1.0"?>
<launch>
  <arg name="use_sim_time" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>

  <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

  <include file="$(find gazebo_ros)
                              /launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/
                        test/test_worlds/$(arg world).world"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
  </include>
</launch>
```

# ROS Launch

## Including Other Launch Files

- Include other launch files with <include> tag to organize large projects

```
<include file="package_name"/>
```

- Find the system path to other packages with

```
$(find package_name)
```

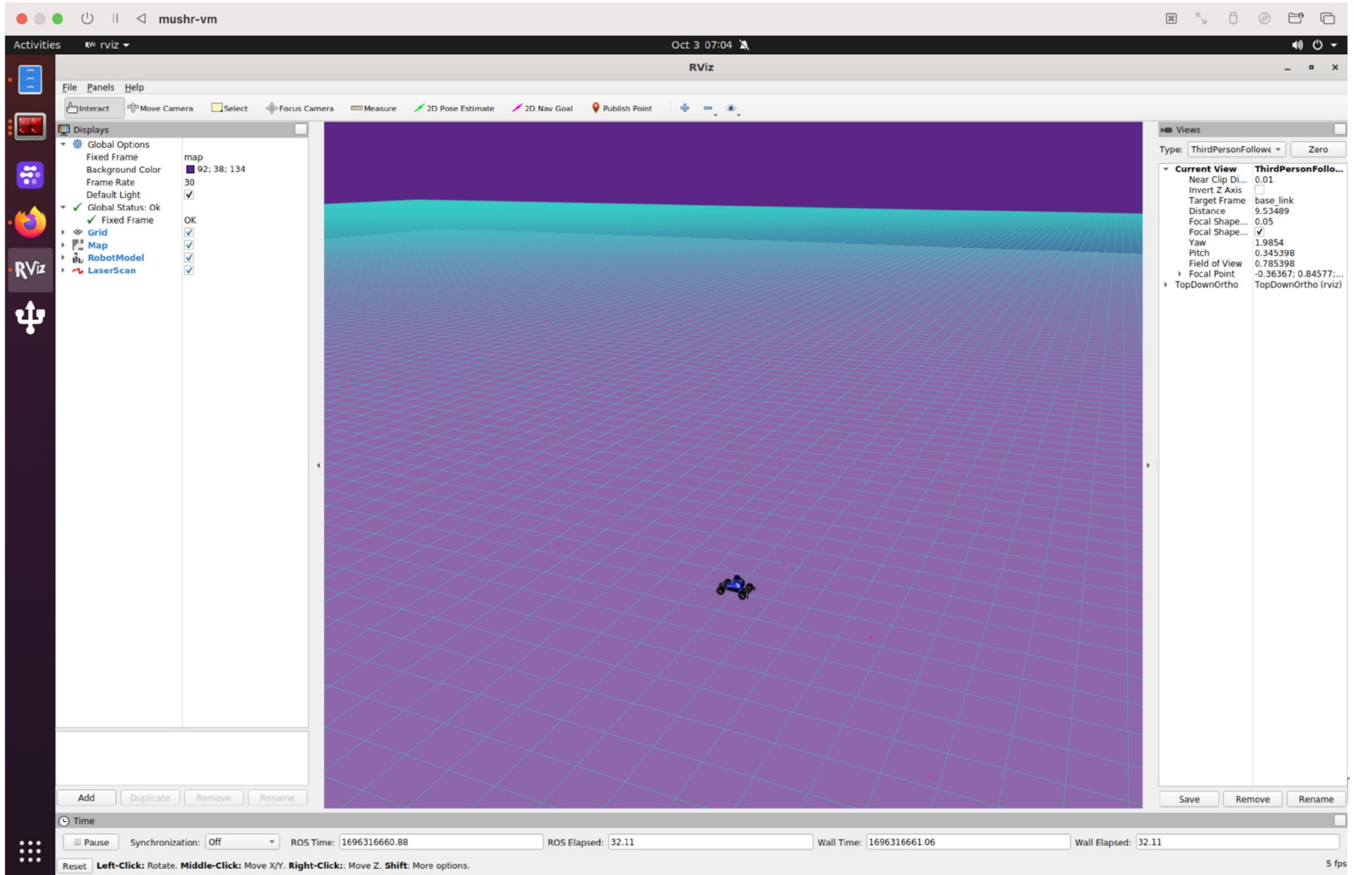
- Pass arguments to the included file

```
<arg name="arg_name" value="value"/>
```

*range_world.launch (simplified)*

```
<?xml version="1.0"?>
<launch>
  <arg name="use_sim_time" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>

  <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

  <include file="$(find gazebo_ros)
                              /launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/
                          test/test_worlds/$(arg world).world"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
  </include>
</launch>
```

# RVIZ Simulator



| | |
|---|---|
| Global Options | |
| Fixed Frame | map |
| Background Color | 92; 38; 134 |
| Frame Rate | 30 |
| Default Light | ✓ |
| Global Status: Ok | |
| Fixed Frame | OK |
| Grid | ✓ |
| Map | ✓ |
| RobotModel | ✓ |
| LaserScan | ✓ |

Type: ThirdPersonFollowe Zero

| | |
|---|---|
| Current View | ThirdPersonFollo... |
| Near Clip Di... | 0.01 |
| Invert Z Axis | ☐ |
| Target Frame | base_link |
| Distance | 9.53489 |
| Focal Shape... | 0.05 |
| Focal Shape... | ✓ |
| Yaw | 1.9854 |
| Pitch | 0.345398 |
| Field of View | 0.785398 |
| Focal Point | -0.36367; 0.84577;... |
| TopDownOrtho | TopDownOrtho (rviz) |

Synchronization: Off ROS Time: 1696316660.88 ROS Elapsed: 32.11 Wall Time: 1696316661.06 Wall Elapsed: 32.11

**Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options.

5 fps

## Further References

**ROS** Wiki

> http://wiki.ros.org/

**Installation**

> http://wiki.ros.org/ROS/Installation

**Tutorials**

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

**Available packages**

> https://index.ros.org/packages/

## Further References II

**ROS Cheat Sheet**

> https://www.clearpathrobotics.com/ros-robot-operating-system-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