

CSC752 Autonomous Robotic Systems

- Introduction into ROS (1) -

Ubbo Visser

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

September 2022



UNIVERSITY OF MIAMI
ROBOCANES



- ▶ ROS architecture & philosophy
- ▶ ROS master, nodes, and topics
- ▶ Console commands
- ▶ Catkin workspace and build system
- ▶ Launch-files
- ▶ Gazebo-Simulator



WHAT IS ROS?

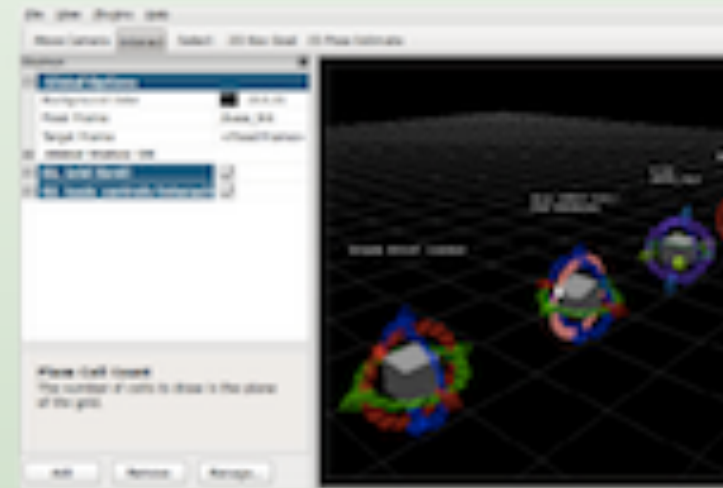
▶ ROS: Robot Operating System



=



+



+



+



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 Stanford (AI Lab)
- ▶ Managed by Open Source Robotics Foundation (OSRF), est. 2012
- ▶ Widespread use for many robots by Universities and companies
- ▶ Emerging de-facto standard for robot programming



Source: ros.org

- ▶ 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 (open source)

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

- ▶ Manages the communication between processes (called “nodes” in ROS language)
- ▶ All nodes are registered with the master (or core) at startup

ROS Master/
Core

Start a Master/Core with

```
~$roscore
```

```
roscore http://192.168.1.206:11311/
roscore http://192.168.1.206:11311/ 89x30
visser@ubuntu:~$ roscore
... logging to /home/visser/.ros/log/561bd072-e2fe-11ea-8e9f-001c42bffd40/roslau
nch-ubuntu-4785.log
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://192.168.1.206:35863/
ros_comm version 1.12.14

SUMMARY
=====
PARAMETERS
* /roscore: kinetic
* /rosversion: 1.12.14

NODES

auto-starting new master
process[roscore]: started with pid [4799]
ROS_MASTER_URI=http://192.168.1.206:11311/

setting /run_id to 561bd072-e2fe-11ea-8e9f-001c42bffd40
process[roscore-1]: started with pid [4820]
started core service [/roscore]
```

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

- ▶ 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

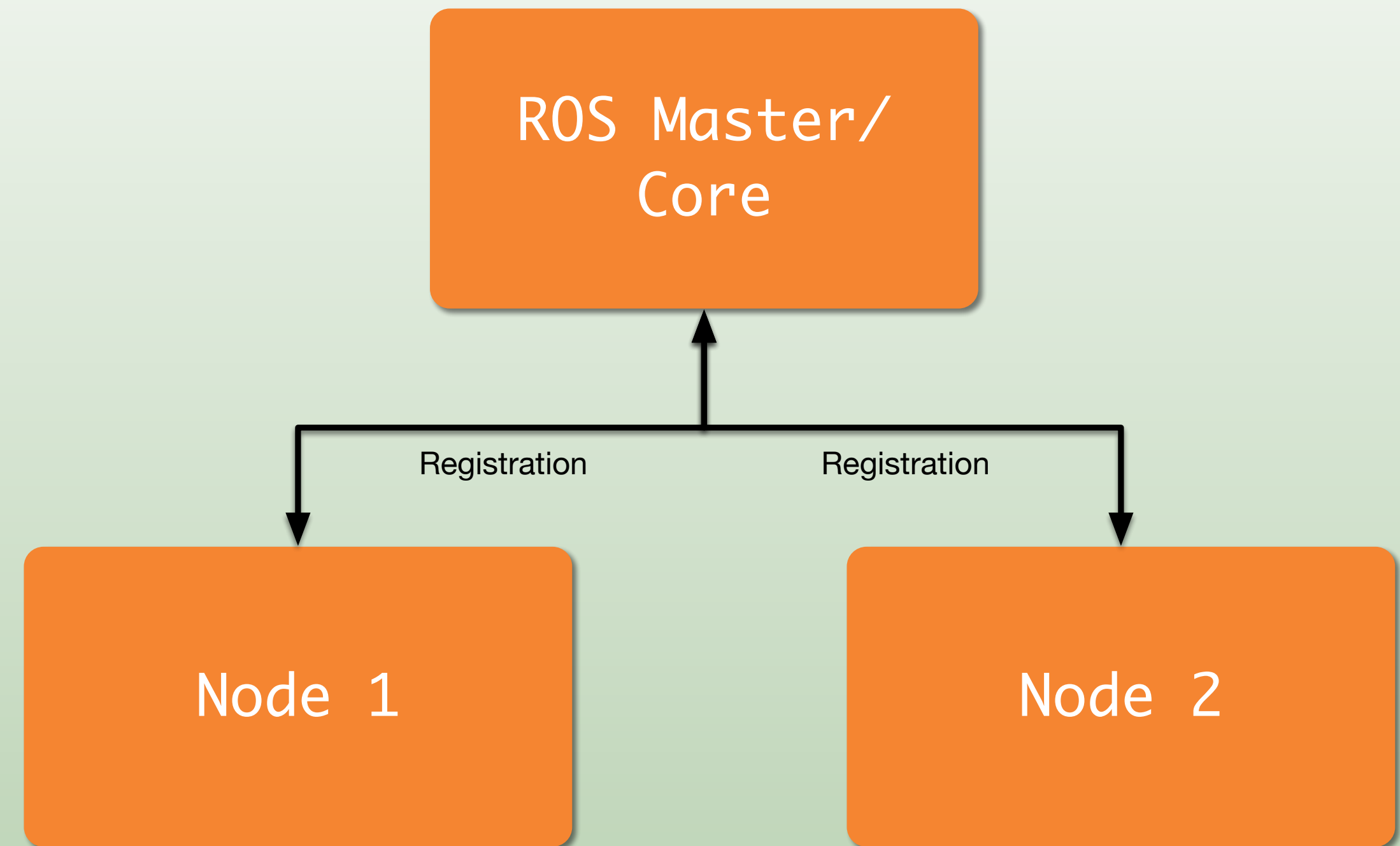
```
~$rosrun package_name node_name
```

List active nodes

```
~$roscall node_name list
```

Node information

```
~$roscall node_name info
```



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

- ▶ 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

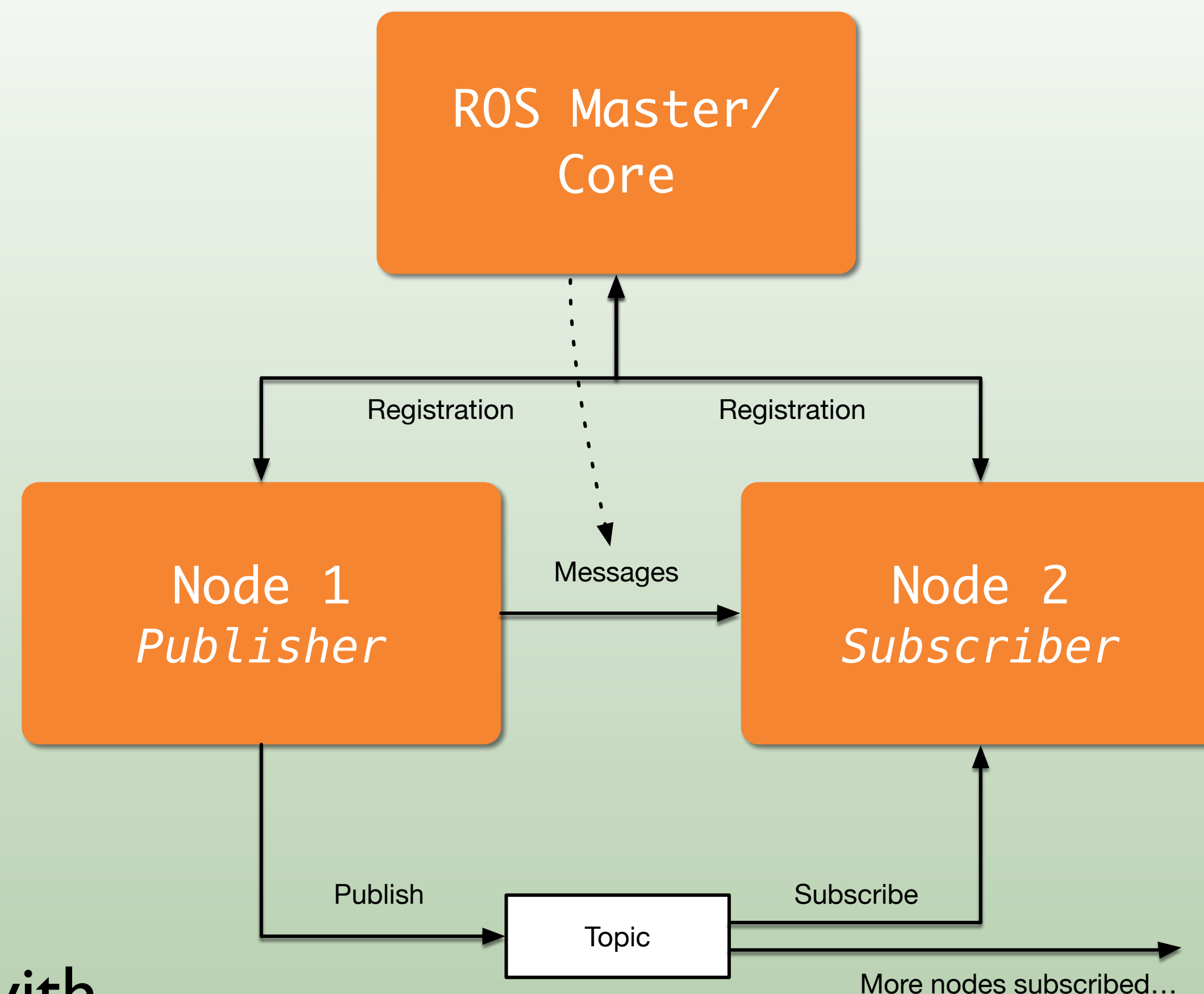
```
~$rostopic list
```

Subscribe and print the contents of a topic with

```
~$rostopic echo /topic
```

Show information about a topic with

```
~$rostopic info /topic
```



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

ROS MESSAGES

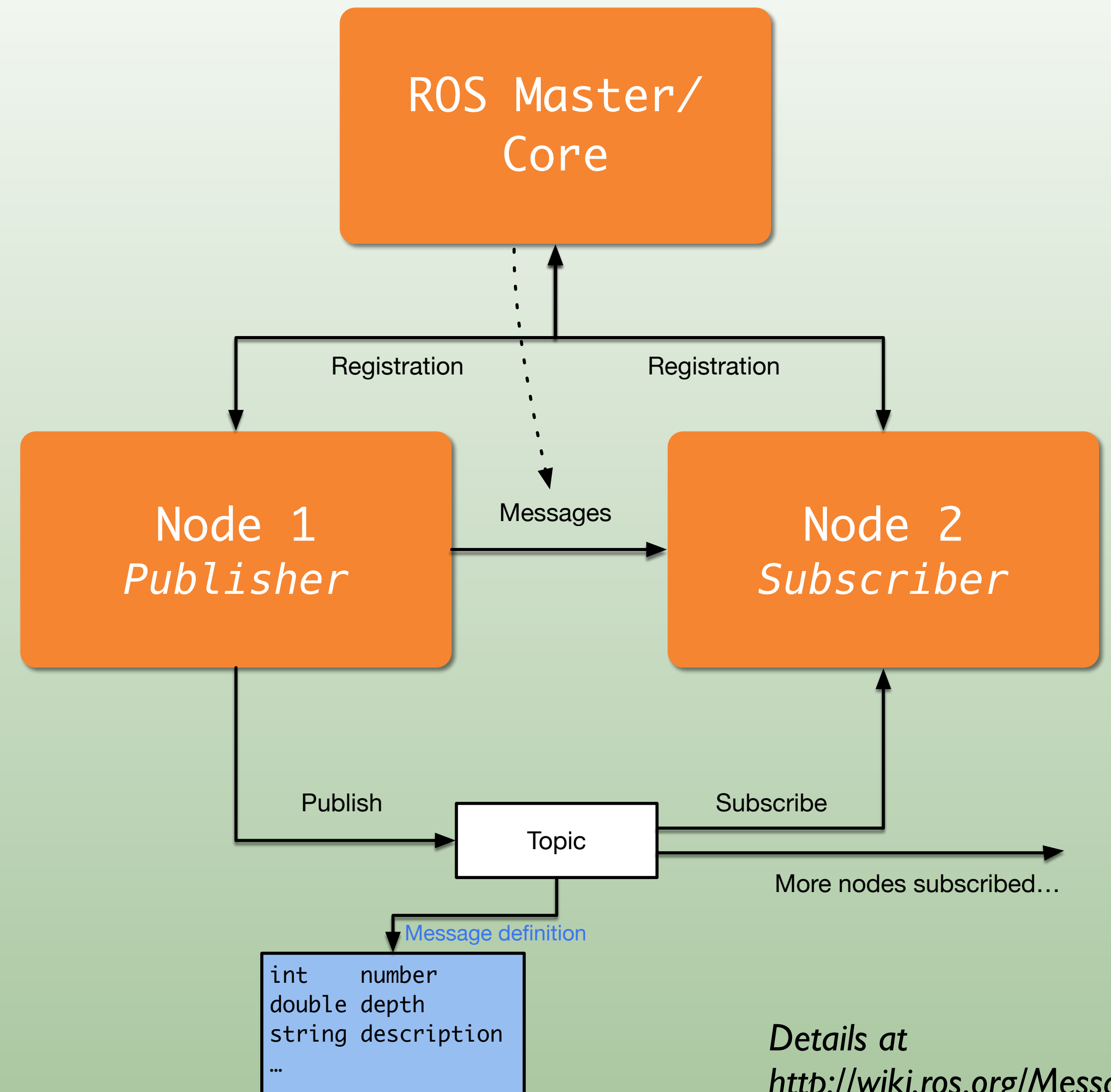
- ▶ Data structure defines *type* of topic
- ▶ Nested structure of primitive data types and arrays
- ▶ Defined in .msg files

List type of topic

```
~$rostopic type /topic
```

Publish a message to a topic

```
~$rostopic pub /topic type data
```



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

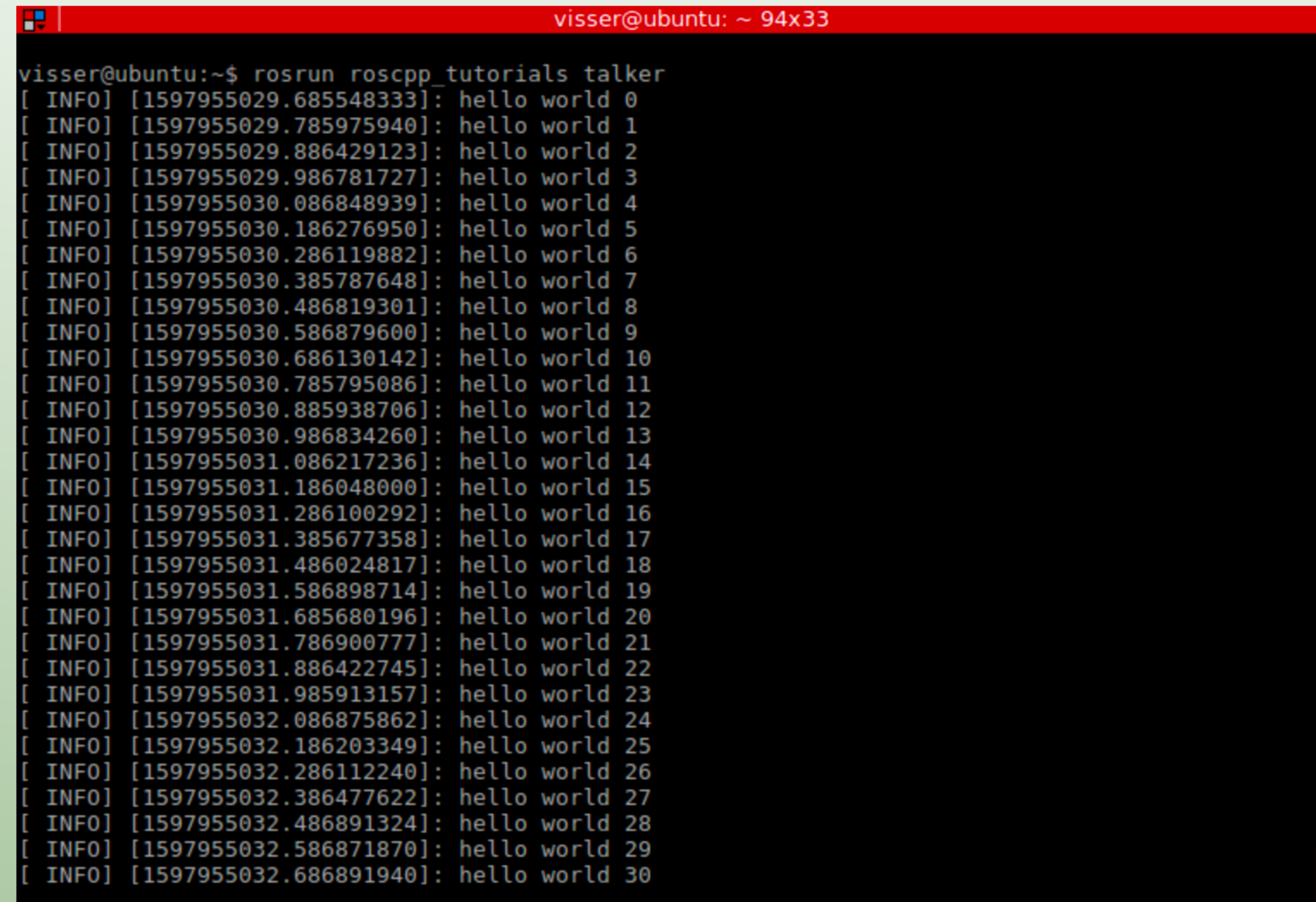
```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
Geometry msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion
orientation
  float64 x
  float64 y
  float64 z
  float64 w
```

EXAMPLE

- ▶ Terminal 1: roscore (already running)
- ▶ Terminal 2: *talker* node

Run talker node

```
~$rosrun roscpp_tutorials talker
```

A terminal window with a red title bar containing the text "visser@ubuntu: ~ 94x33". The terminal content shows the command "rosrun roscpp_tutorials talker" being executed, followed by 31 lines of output. Each line starts with "[INFO]" followed by a timestamp and the message "hello world" followed by a number from 0 to 30.

```
visser@ubuntu:~$ rosrun roscpp_tutorials talker
[ INFO] [1597955029.685548333]: hello world 0
[ INFO] [1597955029.785975940]: hello world 1
[ INFO] [1597955029.886429123]: hello world 2
[ INFO] [1597955029.986781727]: hello world 3
[ INFO] [1597955030.086848939]: hello world 4
[ INFO] [1597955030.186276950]: hello world 5
[ INFO] [1597955030.286119882]: hello world 6
[ INFO] [1597955030.385787648]: hello world 7
[ INFO] [1597955030.486819301]: hello world 8
[ INFO] [1597955030.586879600]: hello world 9
[ INFO] [1597955030.686130142]: hello world 10
[ INFO] [1597955030.785795086]: hello world 11
[ INFO] [1597955030.885938706]: hello world 12
[ INFO] [1597955030.986834260]: hello world 13
[ INFO] [1597955031.086217236]: hello world 14
[ INFO] [1597955031.186048000]: hello world 15
[ INFO] [1597955031.286100292]: hello world 16
[ INFO] [1597955031.385677358]: hello world 17
[ INFO] [1597955031.486024817]: hello world 18
[ INFO] [1597955031.586898714]: hello world 19
[ INFO] [1597955031.685680196]: hello world 20
[ INFO] [1597955031.786900777]: hello world 21
[ INFO] [1597955031.886422745]: hello world 22
[ INFO] [1597955031.985913157]: hello world 23
[ INFO] [1597955032.086875862]: hello world 24
[ INFO] [1597955032.186203349]: hello world 25
[ INFO] [1597955032.286112240]: hello world 26
[ INFO] [1597955032.386477622]: hello world 27
[ INFO] [1597955032.486891324]: hello world 28
[ INFO] [1597955032.586871870]: hello world 29
[ INFO] [1597955032.686891940]: hello world 30
```

EXAMPLE

- ▶ Terminal 3: analyse *talker* node

Active nodes

```
~$roscnode list
```

Info about node

```
~$roscnode info /talker
```

Info about topic

```
~$rostopic info /chatter
```

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

```
visser@ubuntu: ~ 62x24
visser@ubuntu:~$ roscnode info /talker
-----
Node [/talker]
Publications:
 * /chatter [std_msgs/String]
 * /rosout [rosgaph_msgs/Log]

Subscriptions: None

Services:
 * /talker/get_loggers
 * /talker/set_logger_level

contacting node http://192.168.1.206:45553/ ...
Pid: 23654
Connections:
 * topic: /rosout
   * to: /rosout
   * direction: outbound
   * transport: TCPROS
```

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

Publishers:
 * /talker (http://192.168.1.206:45553/)

Subscribers: None
```

EXAMPLE

▶ Terminal 3: analyse *chatter* topic

Type of topic

```
~$rostopic type /chatter
```

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

Message contents

```
~$rostopic 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

```
~$rostopic 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

- ▶ Terminal 4: starting new *listener* node

Type of topic

```
~$roslaunch roscpp_tutorials 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

- ▶ Back to Terminal 3: analyse

New listener node visible

```
~$roscd list
```

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

Show the connection of the nodes over the chatter topic with

```
~$rostopic 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

```
~$rostopic pub /chatter std_msgs/  
String "data: 'RoboCanes greets  
CSC752 students'"
```

Message shows up *once* in T4 (listener)

```
visser@ubuntu: ~ 62x24  
[ INFO] [1597956676.265466514]: I heard: [hello world 54]  
[ INFO] [1597956676.365644389]: I heard: [hello world 55]  
[ INFO] [1597956676.465646935]: I heard: [hello world 56]  
[ INFO] [1597956676.564679535]: I heard: [hello world 57]  
[ INFO] [1597956693.494551197]: I heard: [hello world 3]  
[ INFO] [1597956693.595212991]: I heard: [hello world 4]  
[ INFO] [1597956693.693550733]: I heard: [hello world 5]  
[ INFO] [1597956693.795065225]: I heard: [hello world 6]  
[ INFO] [1597956693.895147783]: I heard: [hello world 7]  
[ INFO] [1597956693.994281497]: I heard: [hello world 8]  
[ INFO] [1597956694.095181808]: I heard: [hello world 9]  
[ INFO] [1597956694.193673195]: I heard: [hello world 10]  
[ INFO] [1597956694.293703798]: I heard: [hello world 11]  
[ INFO] [1597956694.393892663]: I heard: [hello world 12]  
[ INFO] [1597956694.494038553]: I heard: [hello world 13]  
[ INFO] [1597956694.593948299]: I heard: [hello world 14]  
[ INFO] [1597956694.694770899]: I heard: [hello world 15]  
[ INFO] [1597956694.795028634]: I heard: [hello world 16]  
[ INFO] [1597956694.894331292]: I heard: [hello world 17]  
[ INFO] [1597956694.994895419]: I heard: [hello world 18]  
[ INFO] [1597956695.093597604]: I heard: [hello world 19]  
[ INFO] [1597956747.580596249]: I heard: [RoboCanes greets CSC  
752 students]
```


- ▶ 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
```

Details at
<http://wiki.ros.org/kinetic/Installation/Ubuntu>
<http://wiki.ros.org/catkin/workspaces>

- ▶ *catkin* is ROS' build system (similar to make etc.). *catkin* generates binaries, libraries, and interfaces
- ▶ We recommend the command line tools (CLI tools). Use *catkin_make* instead of *catkin_build*.
- ▶ Navigate to your *catkin* workspace, e.g.:

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

- ▶ Build a package with

```
~$catkin_make [package_name]
```

- ▶ Update your environment after a new package is build

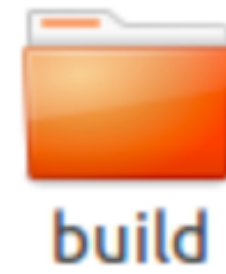
```
~$source devel/setup.bash
```

Details at
<http://wiki.ros.org/catkin/Tutorials>
<https://catkin-tools.readthedocs.io/en/latest/>

- ▶ The catkin workspace contains the following spaces



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.



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.



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

Clean entire workspace with

```
~$catkin clean
```

CATKIN BUILD SYSTEM

- ▶ The catkin workspace setup can be checked

```
~$catkin config
```

- ▶ Example: set build type to Debug/Release use

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

```
visser@ubuntu: ~/catkin_ws  
visser@ubuntu: ~/catkin_ws 80x27  
visser@ubuntu:~/catkin_ws$ catkin config  
-----  
Profile:                                default  
Extending:                               [cached] /opt/ros/kinetic  
Workspace:                               /home/visser/catkin_ws  
-----  
Build Space:                             [exists] /home/visser/catkin_ws/build  
Devel Space:                             [exists] /home/visser/catkin_ws/devel  
Install Space:                           [unused] /home/visser/catkin_ws/install  
Log Space:                               [missing] /home/visser/catkin_ws/logs  
Source Space:                            [exists] /home/visser/catkin_ws/src  
DESTDIR:                                 [unused] None  
-----  
Devel Space Layout:                      linked  
Install Space Layout:                   None  
-----  
Additional CMake Args:                   None  
Additional Make Args:                   None  
Additional catkin Make Args:             None  
Internal Make Job Server:                True  
Cache Job Environments:                  False  
-----  
Whitelisted Packages:                    None  
Blacklisted Packages:                    None  
-----  
Workspace configuration appears valid.  
-----
```

Details at

https://catkin-tools.readthedocs.io/en/latest/verbs/catkin_config.html

https://catkin-tools.readthedocs.io/en/latest/cheat_sheet.html

EXAMPLE

- ▶ In your catkin workspace

```
~$cd catkin_ws
```

- ▶ Build packages

```
~$catkin_make
```

- ▶ Re-source your workspace

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

- ▶ Launch node

```
~$roslaunch hsr_b_gazebo_launch hsr_b_empty_world.launch
```

```
visser@ubuntu: ~/catkin_ws 80x27
visser@ubuntu:~/catkin_ws$ catkin_make
Base path: /home/visser/catkin_ws
Source space: /home/visser/catkin_ws/src
Build space: /home/visser/catkin_ws/build
Devel space: /home/visser/catkin_ws/devel
Install space: /home/visser/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/visser/catkin_ws/build"
####
####
#### Running command: "make -j8 -l8" in "/home/visser/catkin_ws/build"
####
[ 0%] Built target turtlebot_description_xacro_generated_to_devel_space_
[100%] Built target turtlebot_teleop_joy
visser@ubuntu:~/catkin_ws$
```

```
/opt/ros/kinetic/share/husky_gazebo/launch/husky_empty_world.launch http://localhost:11311 80x27
visser@ubuntu:~/catkin_ws$ roslaunch husky_gazebo husky_empty_world.launch
... logging to /home/visser/.ros/log/39b7917c-e3fe-11ea-94fe-001c42bffd40/roslau
nch-ubuntu-13204.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

inconsistent namespace redefinitions for xmlns:xacro:
old: http://ros.org/wiki/xacro
new: http://www.ros.org/wiki/xacro (/opt/ros/kinetic/share/lms1xx/urdf/sick_lms
1xx.urdf.xacro)
started roslaunch server http://192.168.1.206:38515/

SUMMARY
=====

PARAMETERS
* /ekf_localization/base_link_frame: base_link
* /ekf_localization/frequency: 50
* /ekf_localization/imu0: imu/data
* /ekf_localization/imu0_config: [False, False, Fa...
* /ekf_localization/imu0_differential: True
* /ekf_localization/imu0_queue_size: 10
* /ekf_localization/imu0_remove_gravitational_acceleration: True
* /ekf_localization/odom0: husky_velocity_co...
```

- ▶ *launch* is used for launching multiple nodes. A launch file also acts like a config file
- ▶ Written in XML as .launch files
- ▶ Also starts a roscore if not running

Browse to folder and start launch file

```
~$roslaunch file_name.launch
```

Start launch file from package

```
~$roslaunch package_name file_name.launch
```

```
visser@ubuntu: ~/catkin_ws 80x27
3;J
visser@ubuntu:~/catkin_ws$ roslaunch roscpp_tutorials talker_listener.launch
... logging to /home/visser/.ros/log/270bfa12-e3ff-11ea-94fe-001c42bffd40/roslau
nch-ubuntu-15679.log
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://192.168.1.206:34647/

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

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

auto-starting new master
process[master]: started with pid [15689]
ROS_MASTER_URI=http://localhost:11311
```

```
roslaunch roscpp_tutorials talker_listener.launch
```

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

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>
```

- ▶ **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)

Details at

<http://wiki.ros.org/roslaunch/XML>

<http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20tips%20for%20larger%20projects>

- ▶ Create re-usable launch files with <arg> tag

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

- ▶ Use arguments in launch file

```
$(arg arg_name)
```

- ▶ When launching, arguments can be set

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

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

  <!-- these are the arguments you can pass this launch file, for example -->
  <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_world.launch" />
    <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 LAUNCH NESTED LAUNCH FILES

- ▶ Use `<include>` tag for large projects

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

- ▶ Find the system path to other packages

```
$(find package_name)
```

- ▶ Pass arguments to included file

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

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

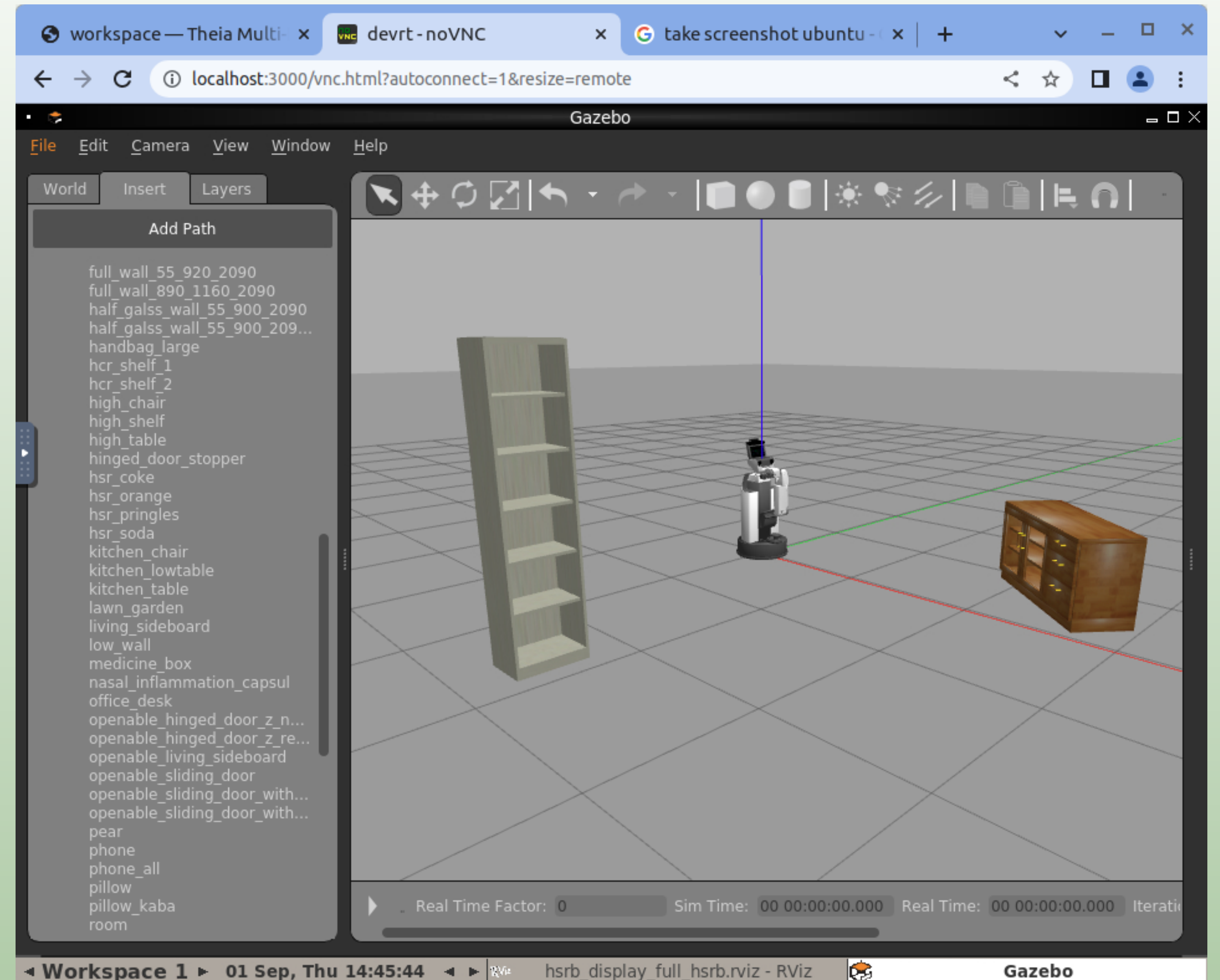
  <!-- these are the arguments you can pass this launch file, for example -->
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="extra_gazebo_args" default=""/>
  <arg name="gui" default="true"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>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

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



```
~$rosrun gazebo_ros gazebo
```

FURTHER REFERENCES

▶ ROS Wiki

▶ <http://wiki.ros.org/>

▶ Installation

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

▶ Tutorials

▶ <http://wiki.ros.org/ROS/Tutorials>

▶ Packages

▶ <https://www.ros.org/browse/list.php>

▶ 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 Templates

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

Material is based on ROS Wiki and ETH Zürich ROS Introduction (<https://rsl.ethz.ch/>)