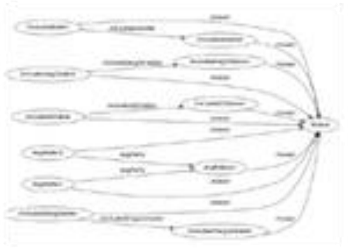
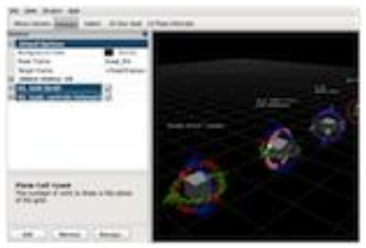
What is ROS?

ROS = Robot Operating System



+



+

+

ros.org

Plumbing	Tools	Capabilities	Ecosystem
▪ 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

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

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 Master

- Manages the communication between nodes (processes)
- Every node registers at startup with the master

ROS Master

Start a master with

```
> roscore
```

More info
http://wiki.ros.org/Master

ROS Nodes

- Single-purpose, executable program
- Individually compiled, executed, and managed
- Organized in *packages*

Run a node with

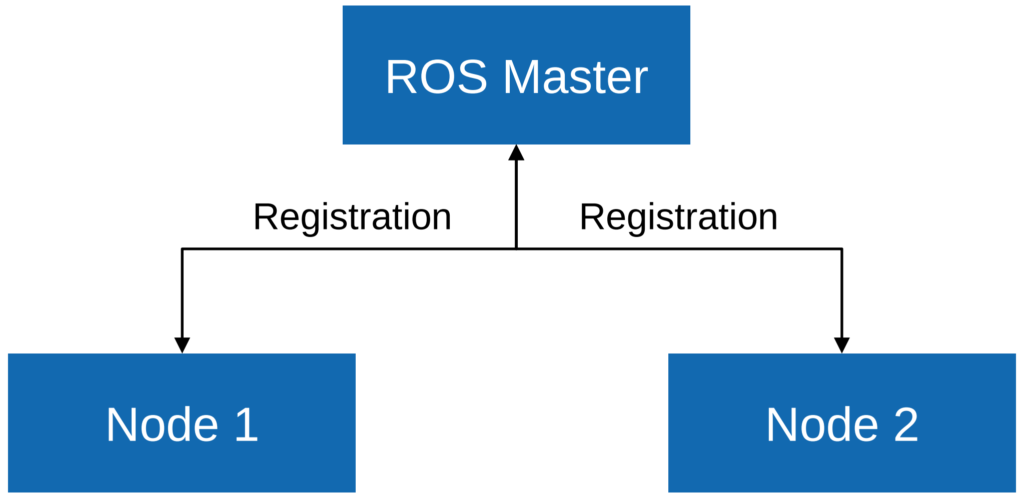
```
> rosrun package_name node_name
```

See active nodes with

```
> rosnode list
```

Retrieve information about a node with

```
> rosnode info node_name
```



More info
http://wiki.ros.org/rosnode

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

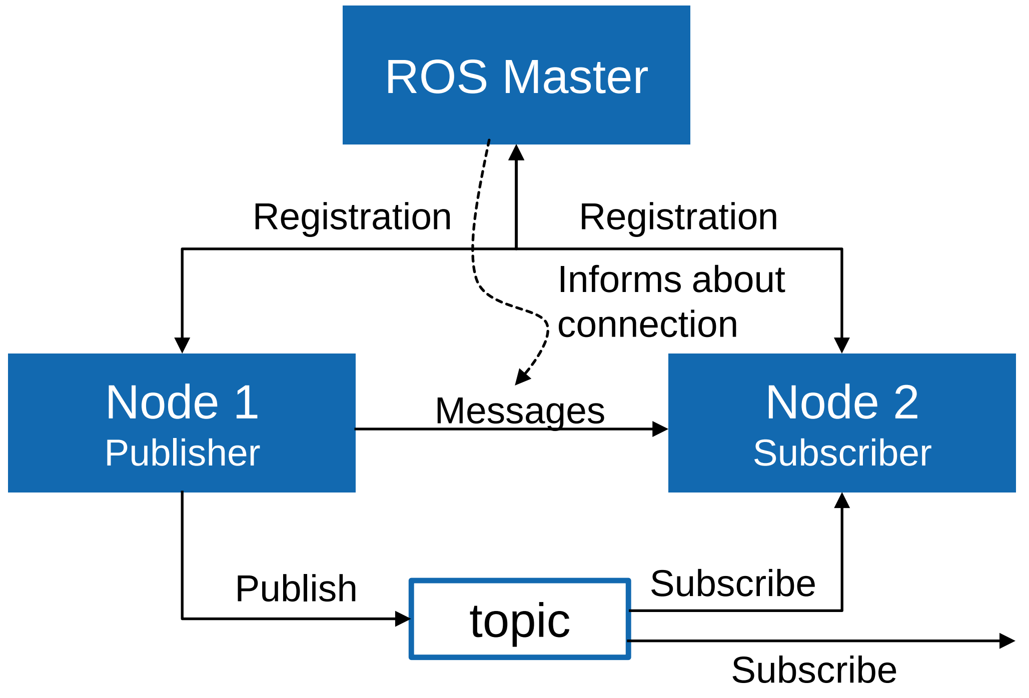
```
> rostopic list
```

Subscribe and print the contents of a topic with

```
> rostopic echo /topic
```

Show information about a topic with

```
> rostopic info /topic
```



More info
http://wiki.ros.org/rostopic

ROS Messages

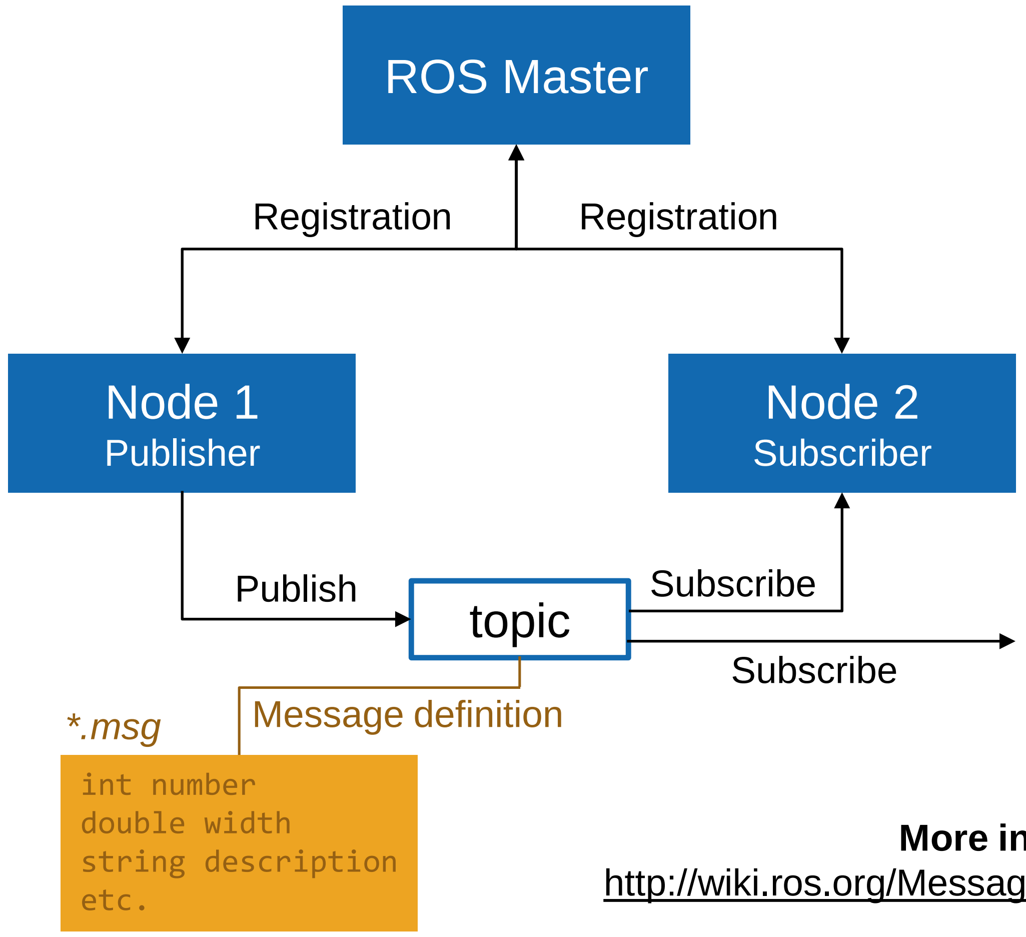
- Data structure defining the *type* of a topic
- Comprised of a nested structure of integers, floats, booleans, strings etc. and arrays of objects
- Defined in **.msg* files

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

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
```

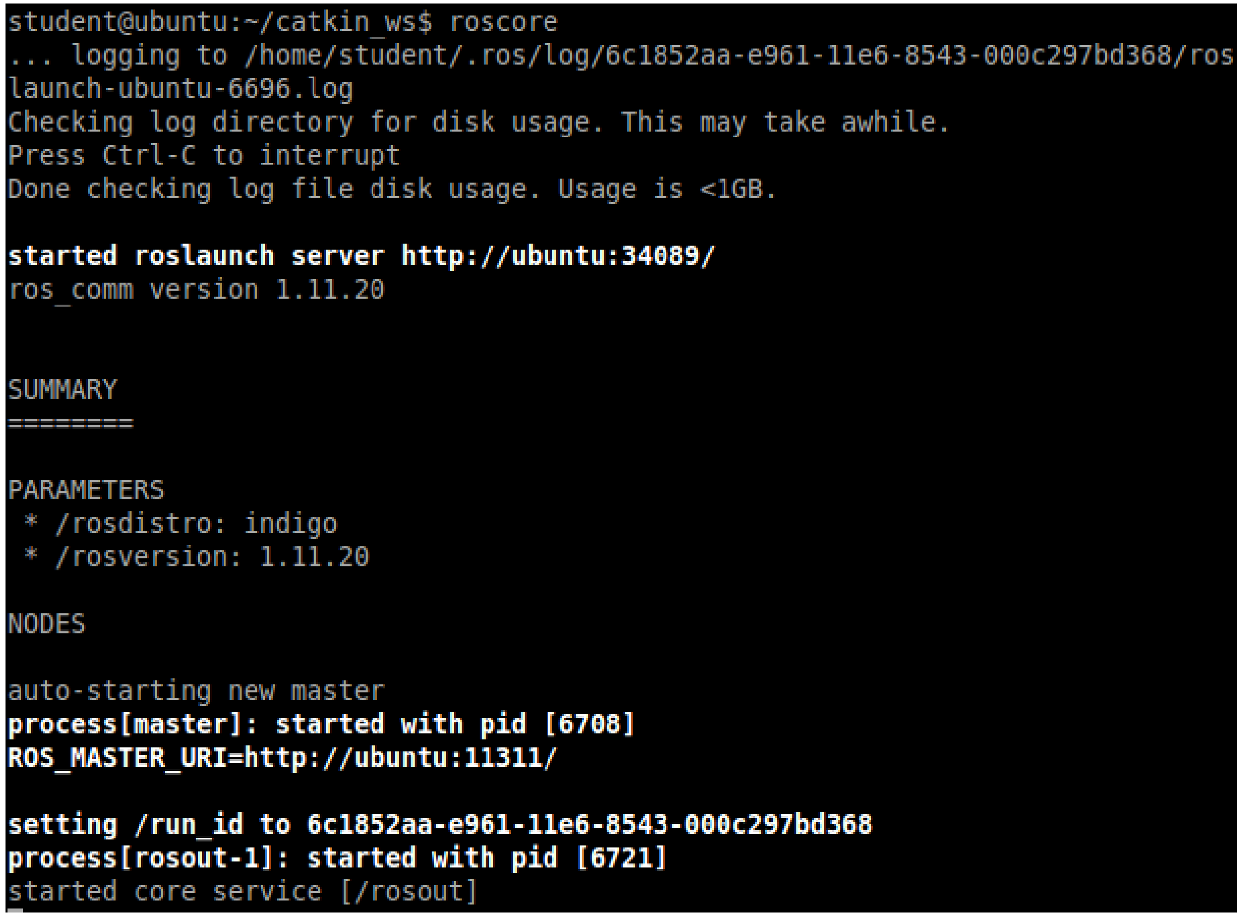
geometry_msgs/PoseStamped.msg

```
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
```

Example
Console Tab Nr. 1 – Starting a *roscore*

Start a roscore with

```
> roscore
```

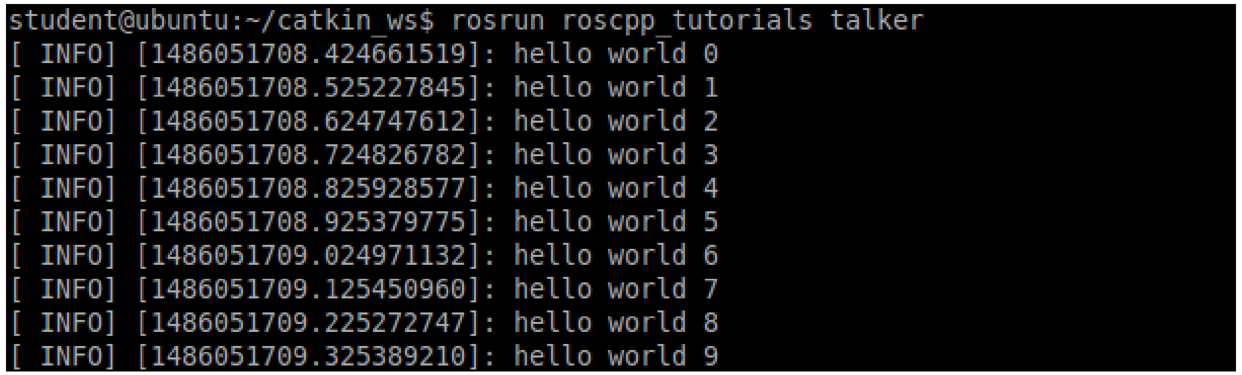


```
student@ubuntu:~/catkin_ws$ roscore
... logging to /home/student/.ros/log/6c1852aa-e961-11e6-8543-000c297bd368/ros
launch-ubuntu-6696.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:34089/
ros_comm version 1.11.20


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.20

NODES

auto-starting new master
process[master]: started with pid [6708]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 6c1852aa-e961-11e6-8543-000c297bd368
process[rosout-1]: started with pid [6721]
started core service [/rosout]
```

Example
Console Tab Nr. 2 – Starting a *talker* node

Run a talker demo node with

```
> rosrun roscpp_tutorials talker
```

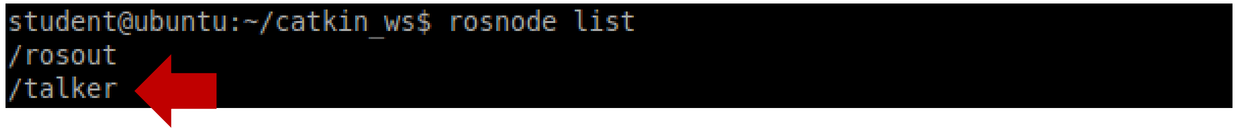


```
student@ubuntu:~/catkin_ws$ rosrun roscpp_tutorials talker
[ INFO] [1486051708.424661519]: hello world 0
[ INFO] [1486051708.525227845]: hello world 1
[ INFO] [1486051708.624747612]: hello world 2
[ INFO] [1486051708.724826782]: hello world 3
[ INFO] [1486051708.825928577]: hello world 4
[ INFO] [1486051708.925379775]: hello world 5
[ INFO] [1486051709.024971132]: hello world 6
[ INFO] [1486051709.125450960]: hello world 7
[ INFO] [1486051709.225272747]: hello world 8
[ INFO] [1486051709.325389210]: hello world 9
```

Example
Console Tab Nr. 3 – Analyze *talker* node

See the list of active nodes

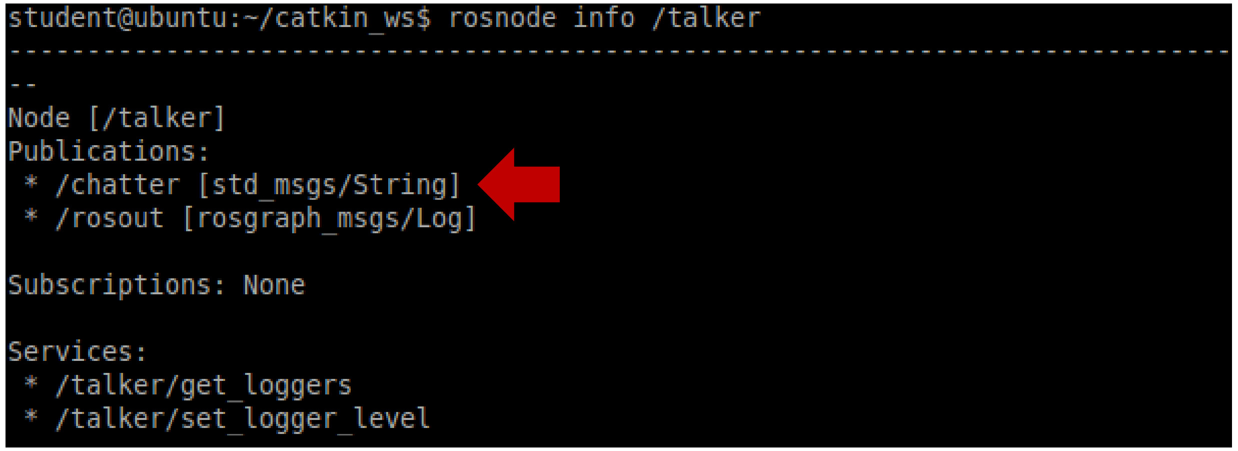
```
> rosnode list
```



```
student@ubuntu:~/catkin_ws$ rosnode list
/rosout
/talker
```

Show information about the *talker* node

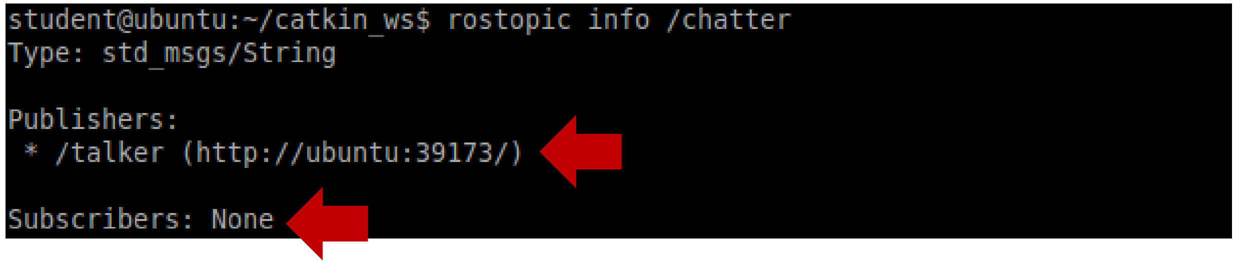
```
> rosnode info /talker
```



```
student@ubuntu:~/catkin_ws$ rosnode info /talker
--------------------------------------------------------------------------------
--
Node [/talker]
Publications:
 * /chatter [std_msgs/String]
 * /rosout [rosgraph_msgs/Log]

Subscriptions: None

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

See information about the *chatter* topic

```
> rostopic info /chatter
```

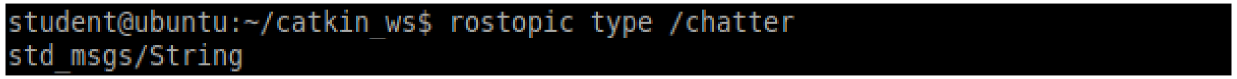


```
student@ubuntu:~/catkin_ws$ rostopic info /chatter
Type: std_msgs/String

Publishers:
 * /talker (http://ubuntu:39173/)

Subscribers: None
```

Example
Console Tab Nr. 3 – Analyze *chatter* topic

Check the type of the *chatter* topic

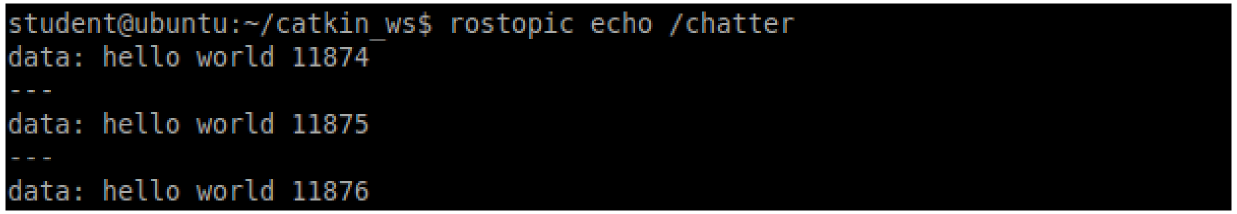
```
> rostopic type /chatter
```



```
student@ubuntu:~/catkin_ws$ rostopic type /chatter
std_msgs/String
```

Show the message contents of the topic

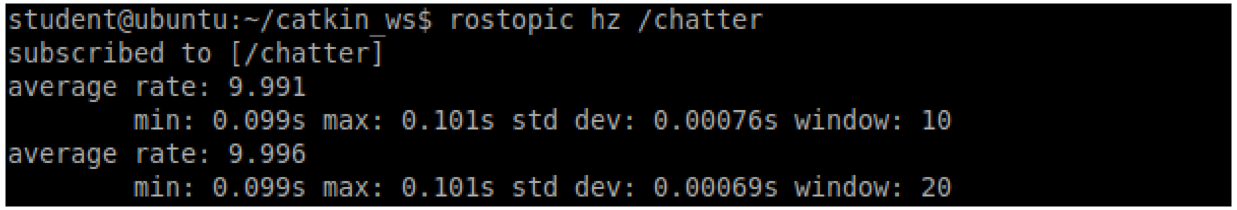
```
> rostopic echo /chatter
```



```
student@ubuntu:~/catkin_ws$ rostopic echo /chatter
data: hello world 11874
---
data: hello world 11875
---
data: hello world 11876
```

Analyze the frequency

```
> rostopic hz /chatter
```



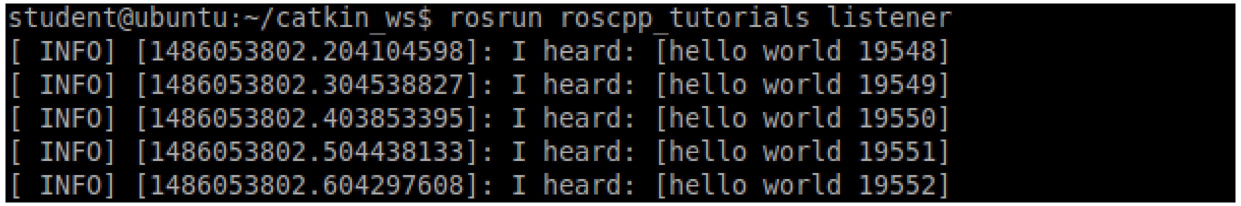
```
student@ubuntu:~/catkin_ws$ rostopic hz /chatter
subscribed to [/chatter]
average rate: 9.991
        min: 0.099s max: 0.101s std dev: 0.00076s window: 10
average rate: 9.996
        min: 0.099s max: 0.101s std dev: 0.00069s window: 20
```

Example

Console Tab Nr. 4 – Starting a *listener* node

Run a listener demo node with

```
> rosrun roscpp_tutorials listener
```



```
student@ubuntu:~/catkin_ws$ rosrun roscpp_tutorials listener
[ INFO] [1486053802.204104598]: I heard: [hello world 19548]
[ INFO] [1486053802.304538827]: I heard: [hello world 19549]
[ INFO] [1486053802.403853395]: I heard: [hello world 19550]
[ INFO] [1486053802.504438133]: I heard: [hello world 19551]
[ INFO] [1486053802.604297608]: I heard: [hello world 19552]
```

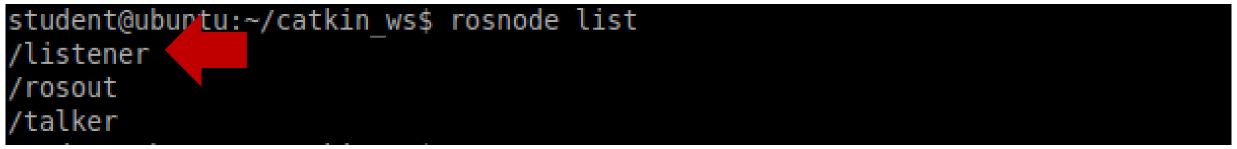
Example
Console Tab Nr. 3 – Analyze

See the new *listener* node with

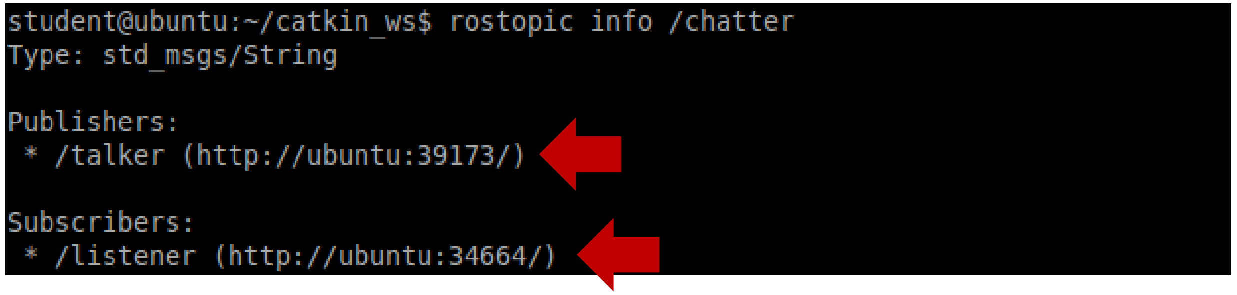
```
> rosnode list
```

Show the connection of the nodes over the chatter topic with

```
> rostopic info /chatter
```



```
student@ubuntu:~/catkin_ws$ rosnode list
/listener
/rosout
/talker
```



```
student@ubuntu:~/catkin_ws$ rostopic info /chatter
Type: std_msgs/String

Publishers:
 * /talker (http://ubuntu:39173/)

Subscribers:
 * /listener (http://ubuntu:34664/)
```

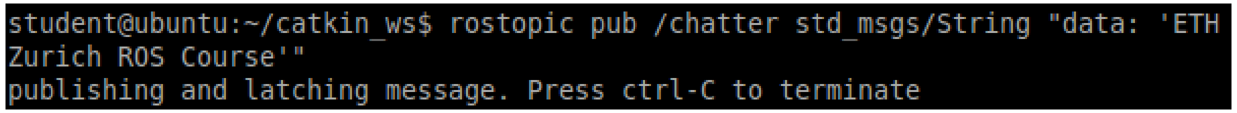
ETH zürich

Example
Console Tab Nr. 3 – Publish Message from Console

Close the *talker* node in console nr. 2 with `Ctrl` + `C`

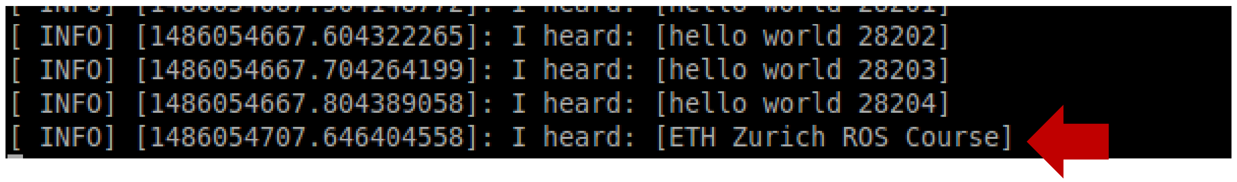
Publish your own message with

```
> rostopic pub /chatter std_msgs/String
  "data: 'ETH Zurich ROS Course'"
```



```
student@ubuntu:~/catkin_ws$ rostopic pub /chatter std_msgs/String "data: 'ETH
Zurich ROS Course'"
publishing and latching message. Press ctrl-C to terminate
```

Check the output of the *listener* in console nr. 4



```
[ INFO] [1486054667.604322265]: I heard: [hello world 28202]
[ INFO] [1486054667.704264199]: I heard: [hello world 28203]
[ INFO] [1486054667.804389058]: I heard: [hello world 28204]
[ INFO] [1486054707.646404558]: I heard: [ETH Zurich ROS Course]
```

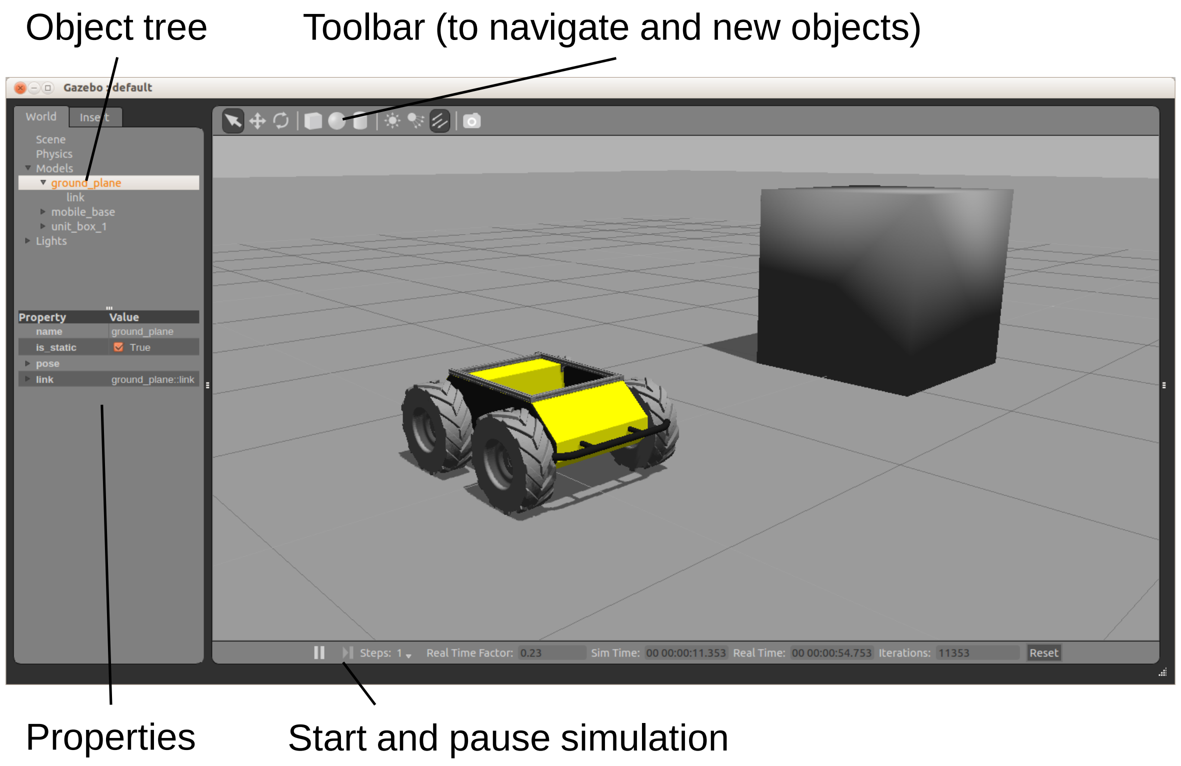
Robotic Systems Lab

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

Run Gazebo with

```
> rosrun gazebo_ros gazebo
```



Property	Value
name	ground_plane
is_static	True
pose	
link	ground_plane::link

Steps: 1 Real Time Factor: 0.23 Sim Time: 00 00:00:11.353 Real Time: 00 00:00:54.753 Iterations: 11353 Reset

More info
http://gazebosim.org/
http://gazebosim.org/tutorials

Further References

- **ROS Wiki**
  - http://wiki.ros.org/
- **Installation**
  - http://wiki.ros.org/ROS/Installation
- **Tutorials**
  - http://wiki.ros.org/ROS/Tutorials
- **Available packages**
  - http://www.ros.org/browse/
- **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