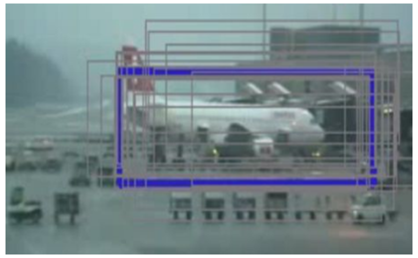
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

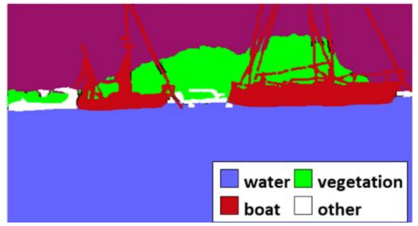
Corso di Visione e Percezione

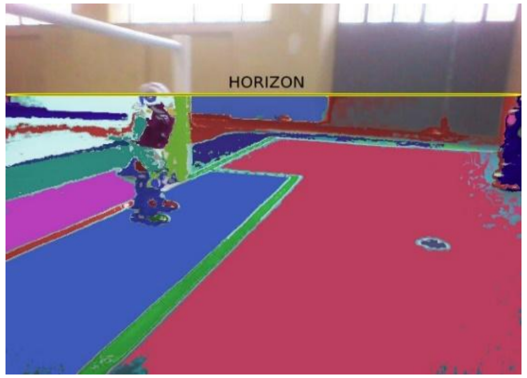
Docente

Domenico D. Bloisi

ROS intro

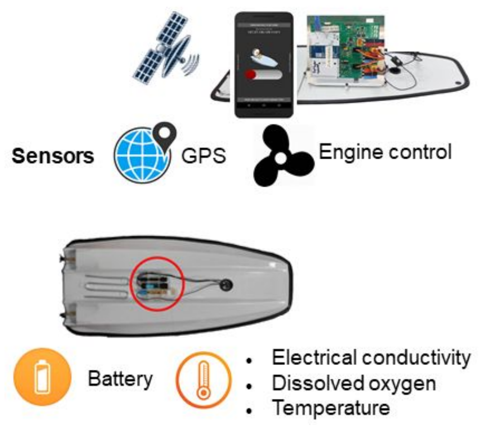


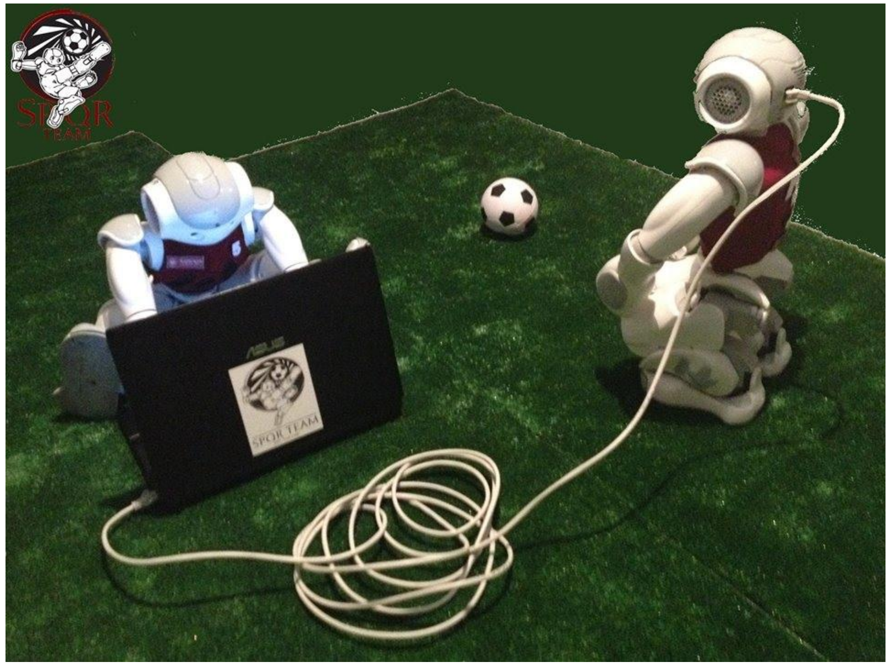




Domenico Daniele Bloisi

- Ricercatore RTD B
  Dipartimento di Matematica, Informatica ed Economia
  Università degli studi della Basilicata
  http://web.unibas.it/bloisi
- SPQR Robot Soccer Team
  Dipartimento di Informatica, Automatica e Gestionale Università degli studi di Roma "La Sapienza"
  http://spqr.diag.uniroma1.it





Informazioni sul corso

- Home page del corso
  http://web.unibas.it/bloisi/corsi/visione-e-percezione.html
- Docente: Domenico Daniele Bloisi
- Periodo: II semestre marzo 2021 – giugno 2021

  Martedì 17:00-19:00 (Aula COPERNICO)
  Mercoledì 8:30-10:30 (Aula COPERNICO)

Codice corso Google Classroom:
https://classroom.google.com/c/NjI2MjA4MzgzNDFa?cjc=xgolays

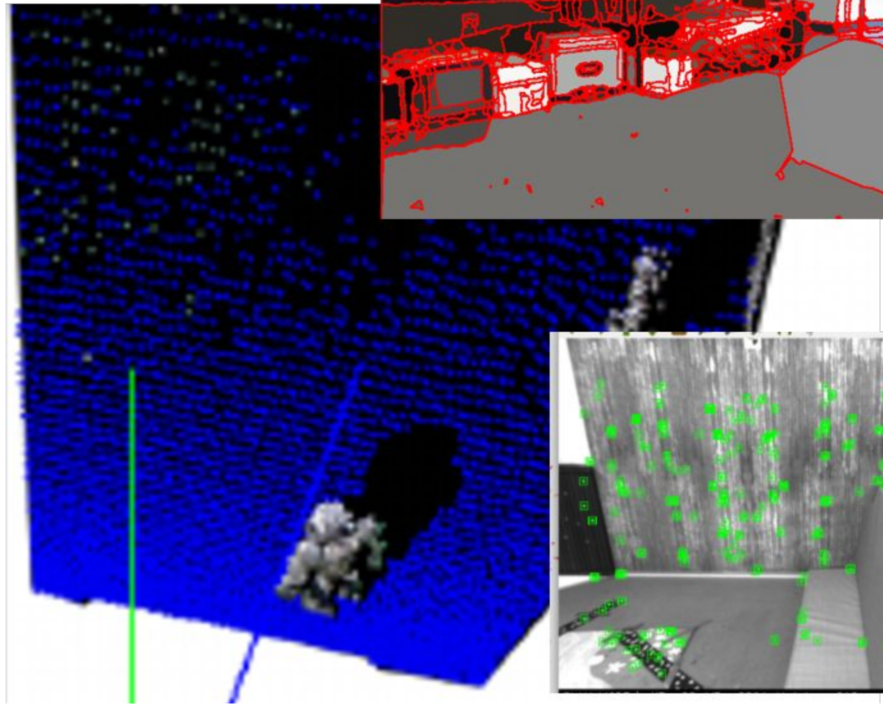
Ricevimento

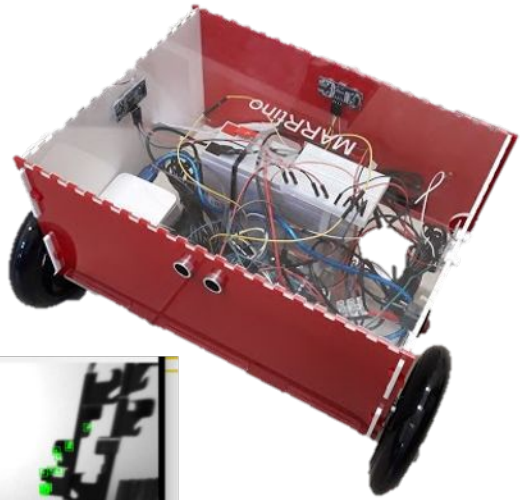
- Su appuntamento tramite Google Meet

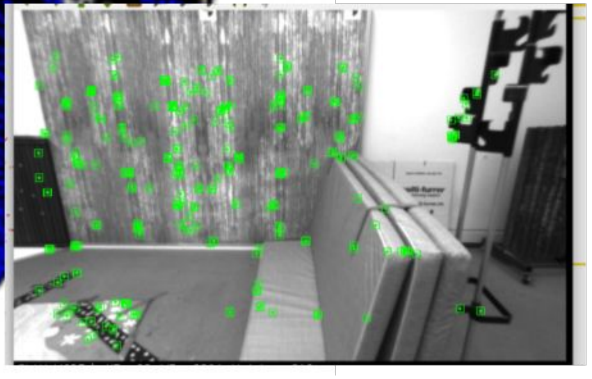
Per prenotare un appuntamento inviare una email a domenico.bloisi@unibas.it

Programma – Visione e Percezione

- Introduzione al linguaggio Python
- Elaborazione delle immagini con Python
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL







References and Credits

- Introduction to ROS
  Roberto Capobianco, Daniele Nardi

- Robot Programming - Robotic Middlewares
  Giorgio Grisetti, Cristiano Gennari

ROS

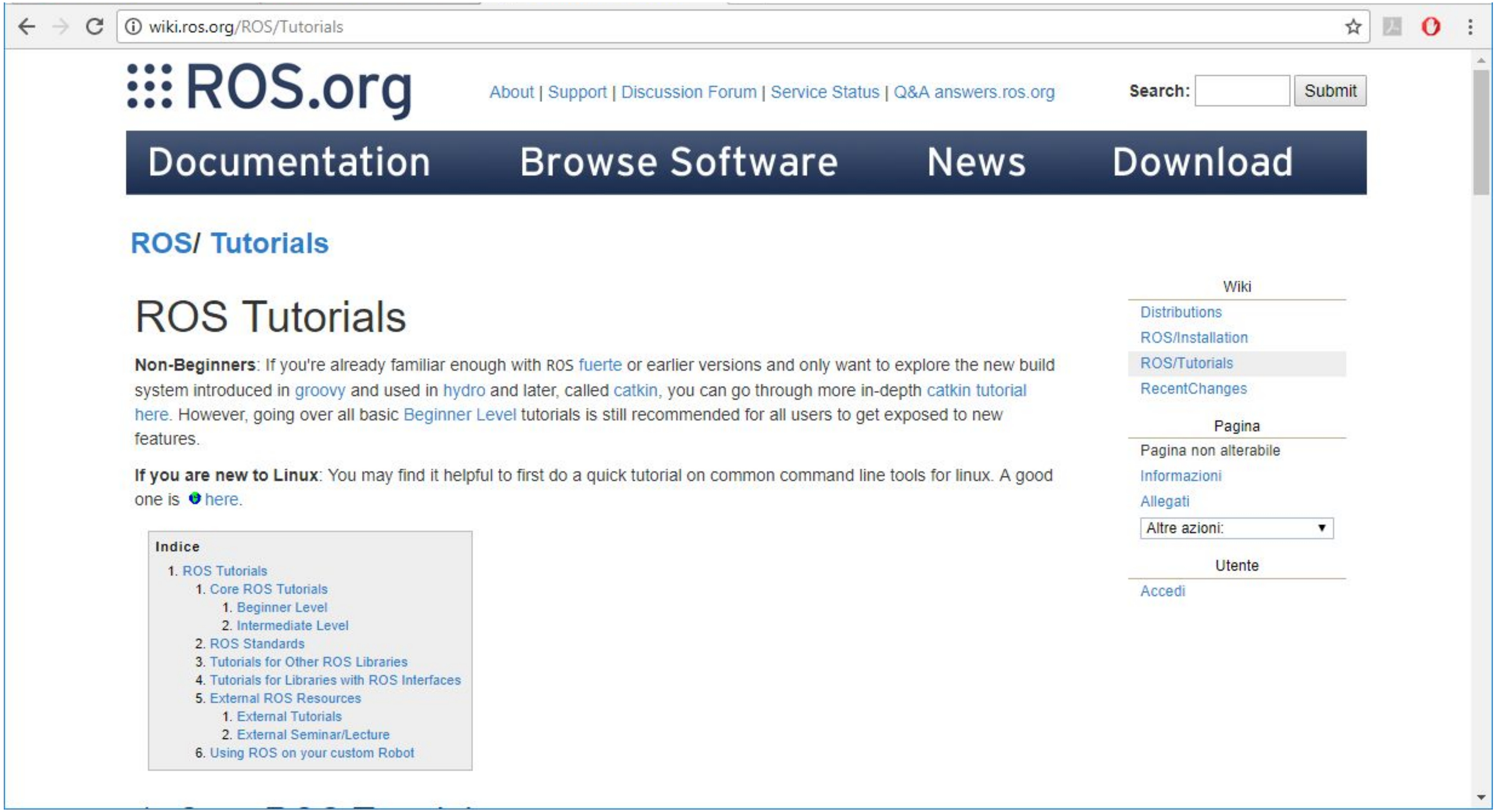
ROS (Robot Operating System) is an open-source, flexible framework for writing robot software

Site: http://www.ros.org/

Blog: http://www.ros.org/news/

Documentation: http://wiki.ros.org/

ROS Tutorials


wiki.ros.org/ROS/Tutorials

ROS.org

About | Support | Discussion Forum | Service Status | Q&A answers.ros.org

Search: Submit

Documentation Browse Software News Download

ROS/ Tutorials

ROS Tutorials

Non-Beginners: If you're already familiar enough with ROS fuerte or earlier versions and only want to explore the new build system introduced in groovy and used in hydro and later, called catkin, you can go through more in-depth catkin tutorial here. However, going over all basic Beginner Level tutorials is still recommended for all users to get exposed to new features.

If you are new to Linux: You may find it helpful to first do a quick tutorial on common command line tools for linux. A good one is here.

Indice

Idea

- Use processes to isolate functionalities of the system
- Processes communicate through messages (less efficient than using shared memory, but safer)
- Benefits
  - If a process crashes, it can be restarted
  - A functionality can be exchanged by replacing a process that provides it
  - Decoupling of modules through inter-process communication

ROS features

- Code reuse (exec. nodes, grouped in packages)
- Distributed, modular design (scalable)
- Language independent (C++, Python, Java, …)
- ROS-agnostic libraries (code is ROS indep.)
- Easy testing (ready-to-use)
- Vibrant community & collaborative environment

ROS = plumbing + tools + capabilities + ecosystem

publish-subscribe messaging infrastructure designed to support the quick and easy construction of distributed computing systems.

tools for configuring, starting, introspecting, debugging, visualizing, logging, testing, and stopping distributed computing systems.

a broad collection of libraries that implement useful robot functionality, with a focus on mobility, manipulation, and perception.

ROS is supported and improved by a large community, with a strong focus on integration and documentation.

https://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating-system/

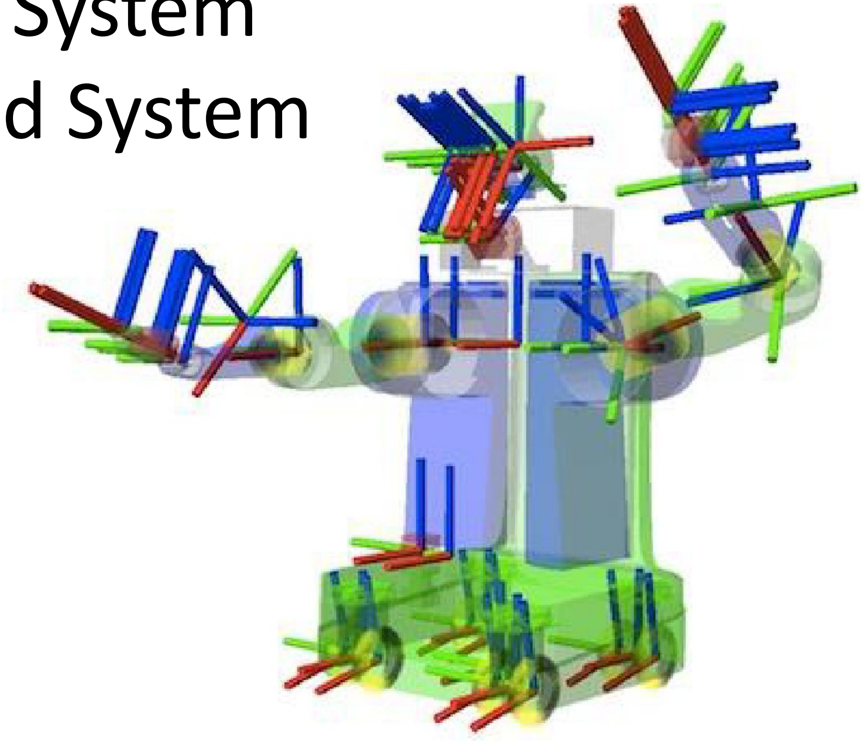
Robot specific features

Provides tools for

- Message Definition
- Process Control
- File System
- Build System

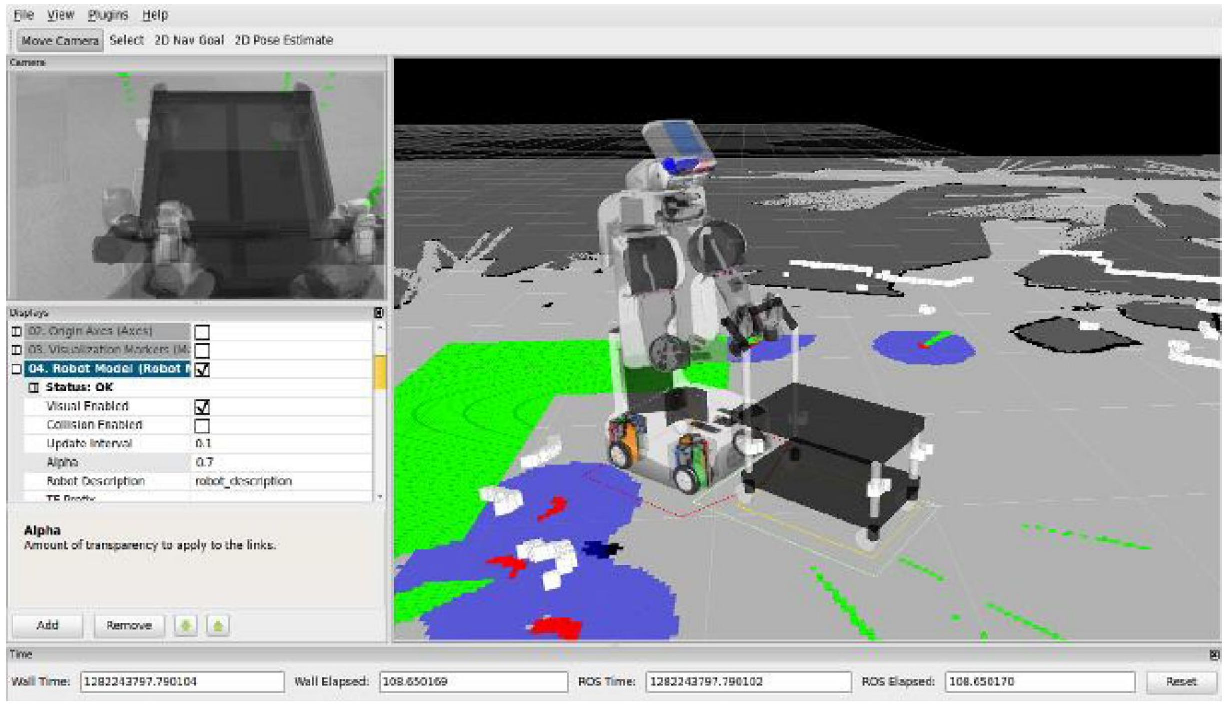
Provides basic functionalities like:

- Device Support
- Navigation
- Control of Manipulator
- Object Recognition



ROS tools

- Command-line tools
- Rviz
- rqt (e.g., rqt_plot, rqt_graph)



02. Origin Axes (Axes)	☐
03. Visualization Markers (M	☐
04. Robot Model (Robot M	✓
Status: OK	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0.1
Alpha	0.7
Robot Description	robot_description
TF Prefix	

Alpha
Amount of transparency to apply to the links.

Wall Time: 1282243797.790104 Wall Elapsed: 108.650169 ROS Time: 1282243797.790102 ROS Elapsed: 108.650170

Integration with external libraries

ROS provides seamless integration of external libraries and popular open-source projects

OpenCV GAZEBO pcl

and many others

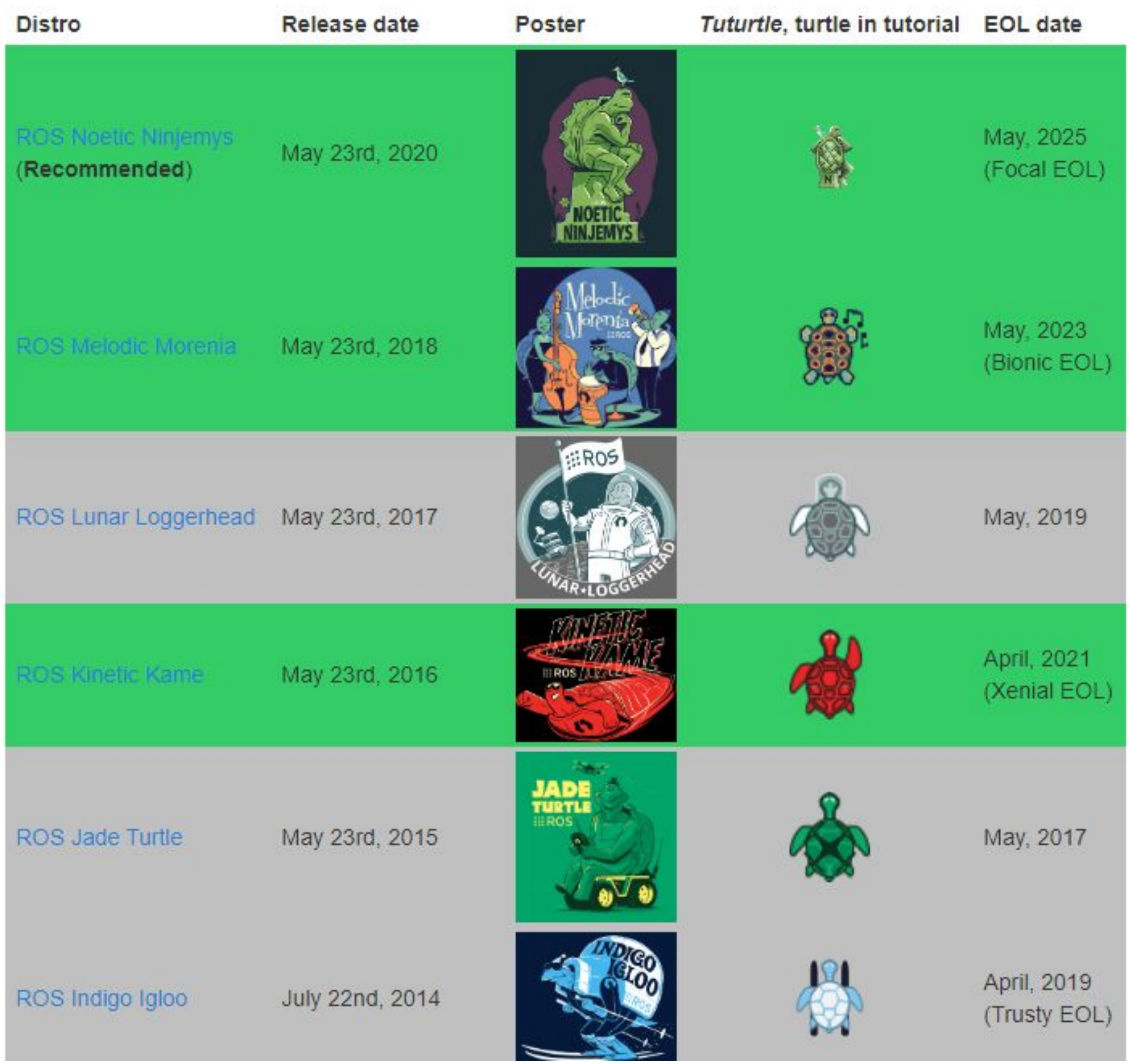
ROS distribution

- A ROS distribution is a versioned set of ROS packages. These are akin to Linux distributions (e.g., Ubuntu).

- The purpose of the ROS distributions is to let developers work against a relatively stable codebase until they are ready to roll everything forward.

http://wiki.ros.org/Distributions

ROS list of distributions



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			May, 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)

http://wiki.ros.org/Distributions

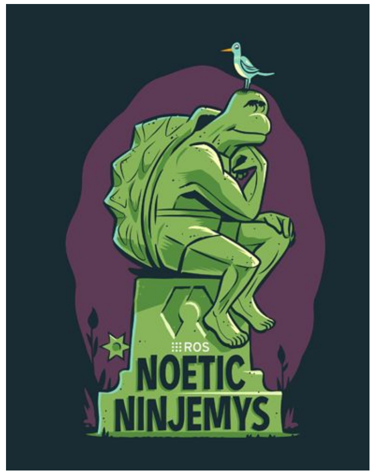
ROS installation

Suggested OS: Ubuntu 20.04 LTS (Focal Fossa)

Suggested ROS distro: Noetic Ninjemys

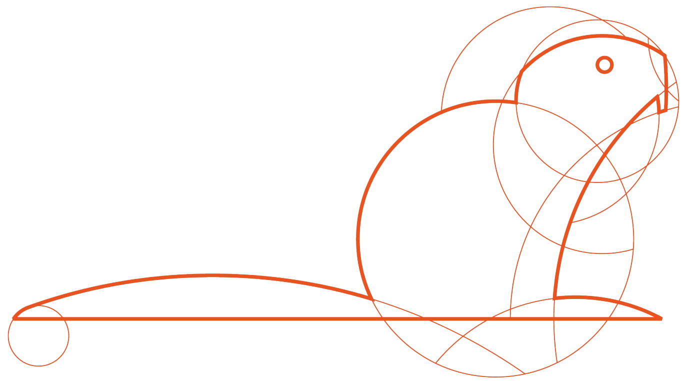
- Install ROS from Debian packages: http://wiki.ros.org/noetic/Installation/Ubuntu

- Install ROS from source (not recommended): http://wiki.ros.org/noetic/Installation/Source



In alternativa

OS: Ubuntu 18.04 LTS (Bionic Beaver)



ROS distro: Melodic Morenia

- Install ROS from Debian packages:
  http://wiki.ros.org/melodic/Installation/Ubuntu

- Install ROS from source (not recommended):
  http://wiki.ros.org/melodic/Installation/Source

Post installation

Initialize rosdep in your system:

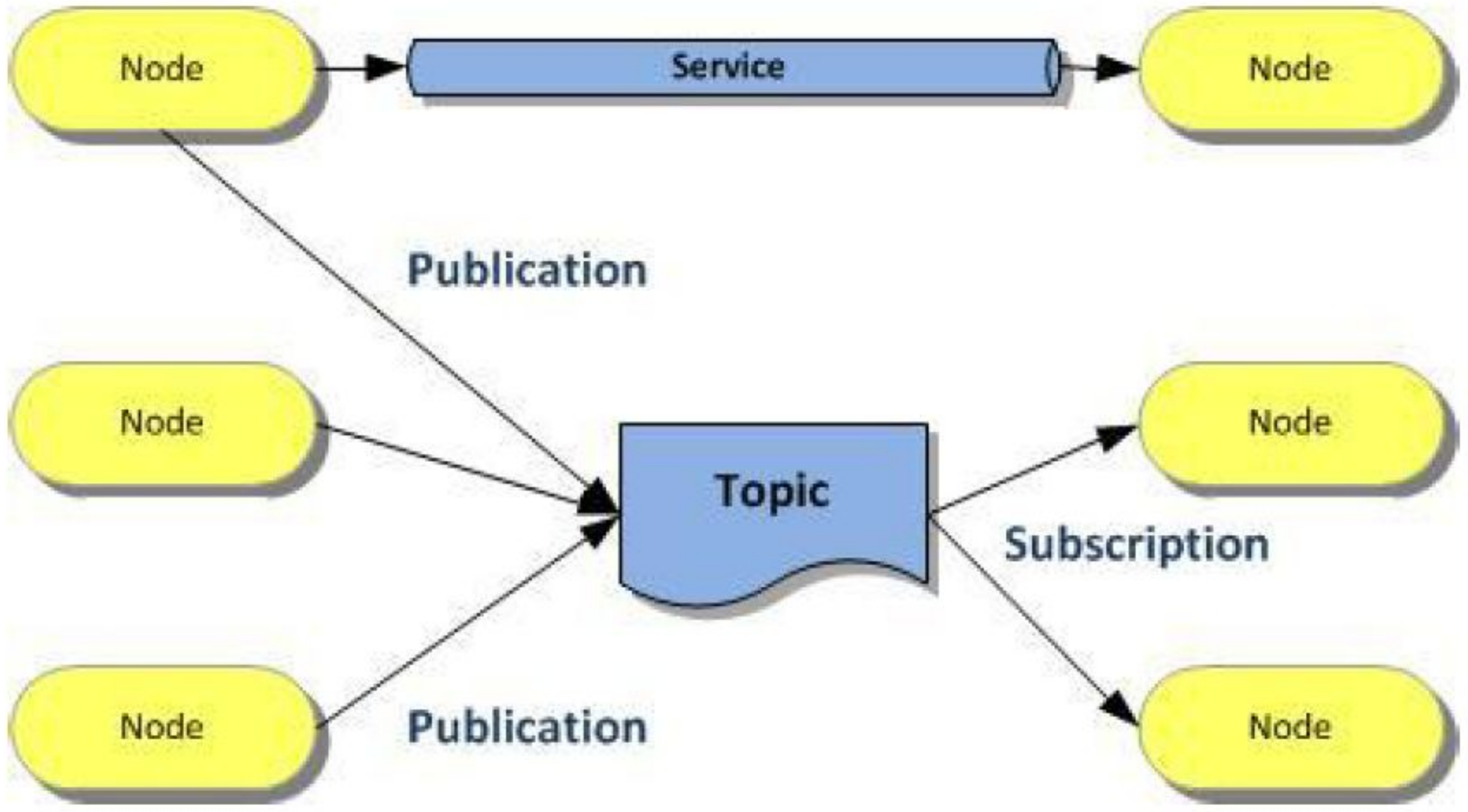
http://wiki.ros.org/rosdep

```
sudo rosdep init
rosdep update
```

rosdep is a tool for checking and installing package dependencies in an OS-independent way

Note: do not use sudo for rosdep update. It is not required and will result in permission errors later on.

ROS definitions



http://wiki.ros.org/ROS/Concepts

ROS definitions

- **Node**: process
- **Message**: Type of a data structure used to communicate between processes
- **Topic**: stream of message instance of the same type used to communicate the evolution of a quantity
  e.g., a CameraNode will publish a stream of images. Each image is of type ImageMessage (a matrix of pixels)
- **Publishing**: the action taken by a node when it wants to broadcast a message
- **Subscribing**: requesting messages of a certain topic

ROS Computation Graph level

ROS creates a network where all the processes are connected.

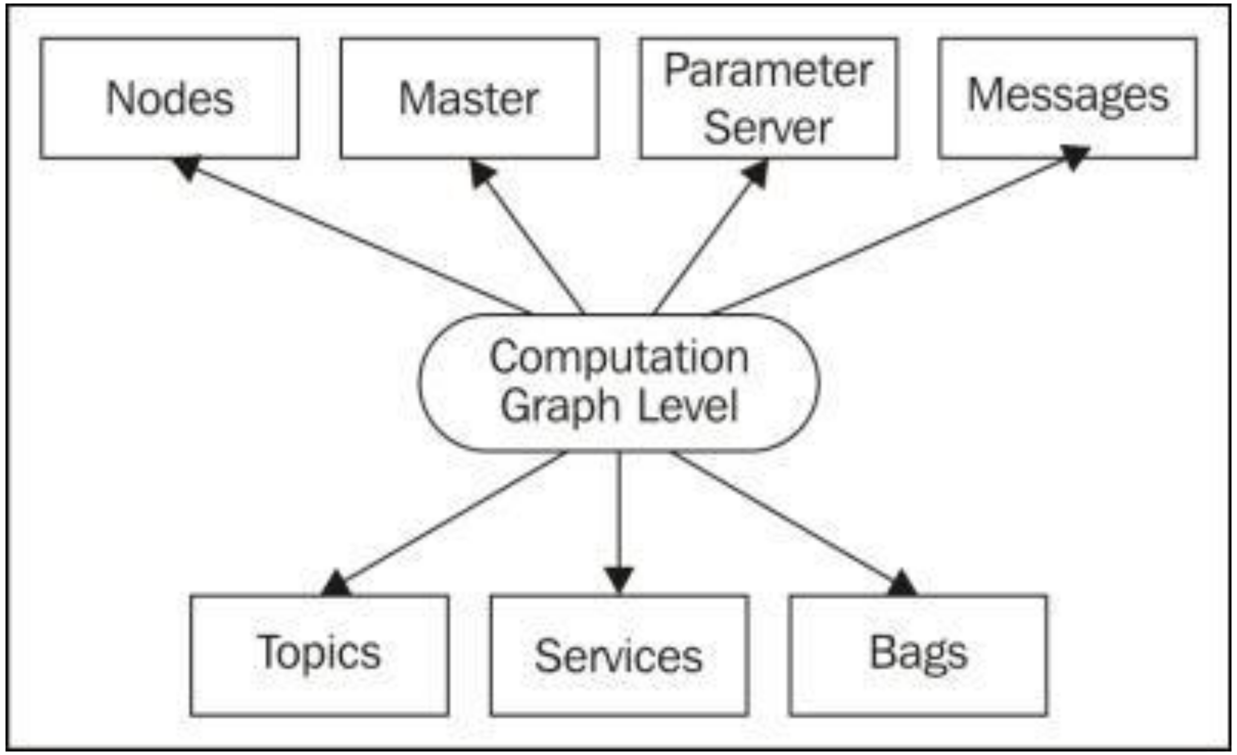


Image from "Learning ROS for Robotics Programming - Second Edition"

ROS Graph example

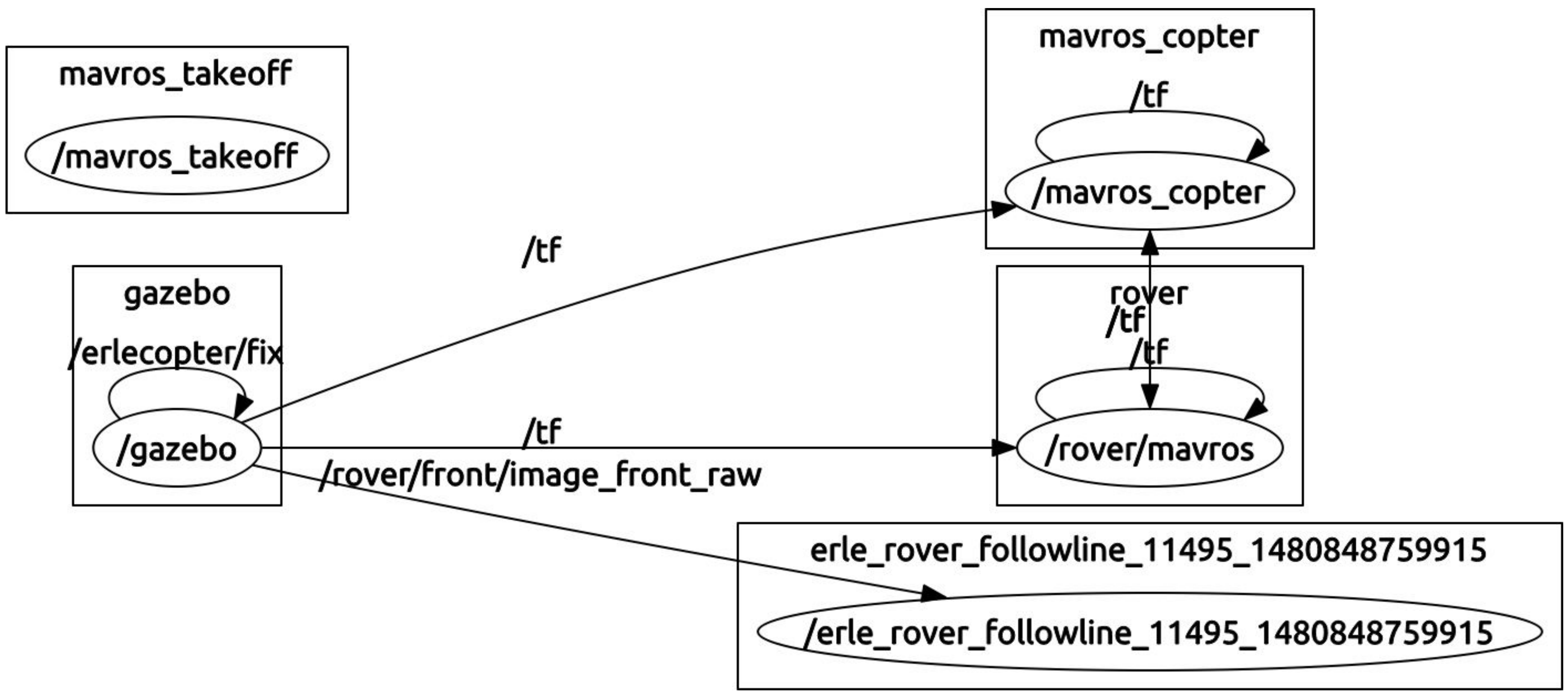


Image from https://answers.ros.org/question/249450/roswtf-unexpectedly-connected-message-and-creation-of-new-nodes/

ROS master

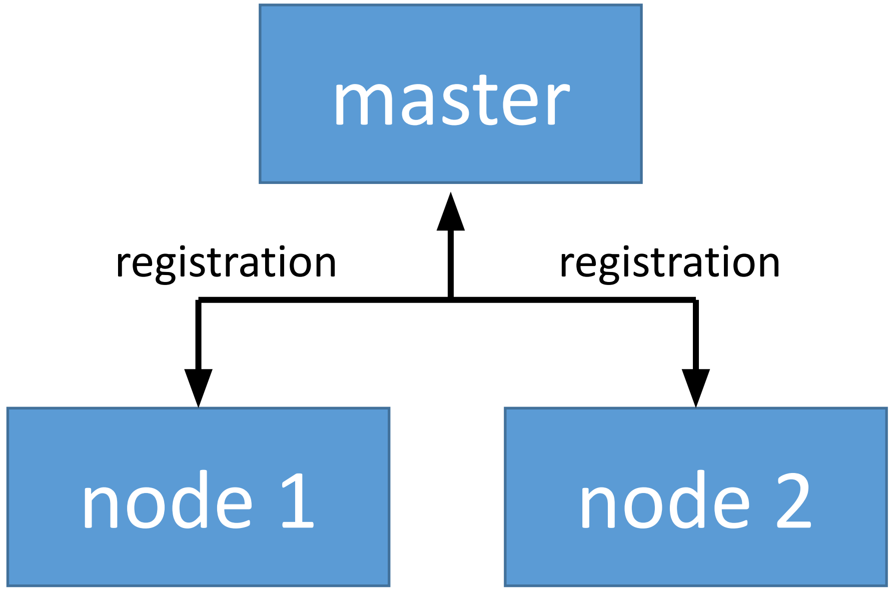
- One of the goals of ROS is to enable the use of small and mostly independent programs (nodes), all running at the same time
- The ROS master provides naming and registration services to enable the nodes to locate each other and, therefore, to communicate
- Every node registers at startup with the master

`roscore`

- Start the ROS master on a terminal with
  ```
  roscore
  ```
- It provides bookkeeping of which nodes are active, which topics are requested by whom, and other facilities
- Nodes need to communicate with the master only at the beginning to know their peers, and which topics are offered
- After that the communication among nodes is peer-to-peer

Nodes

- Running instance of a ROS program
- Starting a node:
  `rosrun <package-name> <node-name>`
- Listing running nodes:
  `rosnode list`
  - `/rosout` is a node started by roscore (similar to `stdout`)
  - `/` indicates the global namespace



rosnode

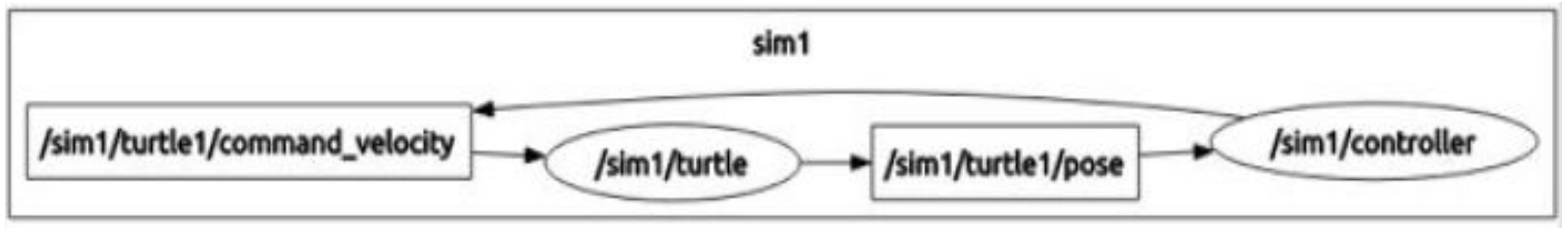
- Inspecting a node (list of topics published and subscribed, services, PID and summary of connections with other nodes):
  `rosnode info` *`node-name`*

- Kill a node (also CTRL+C, but unregistration may not happen)
  `rosnode kill` *`node-name`*

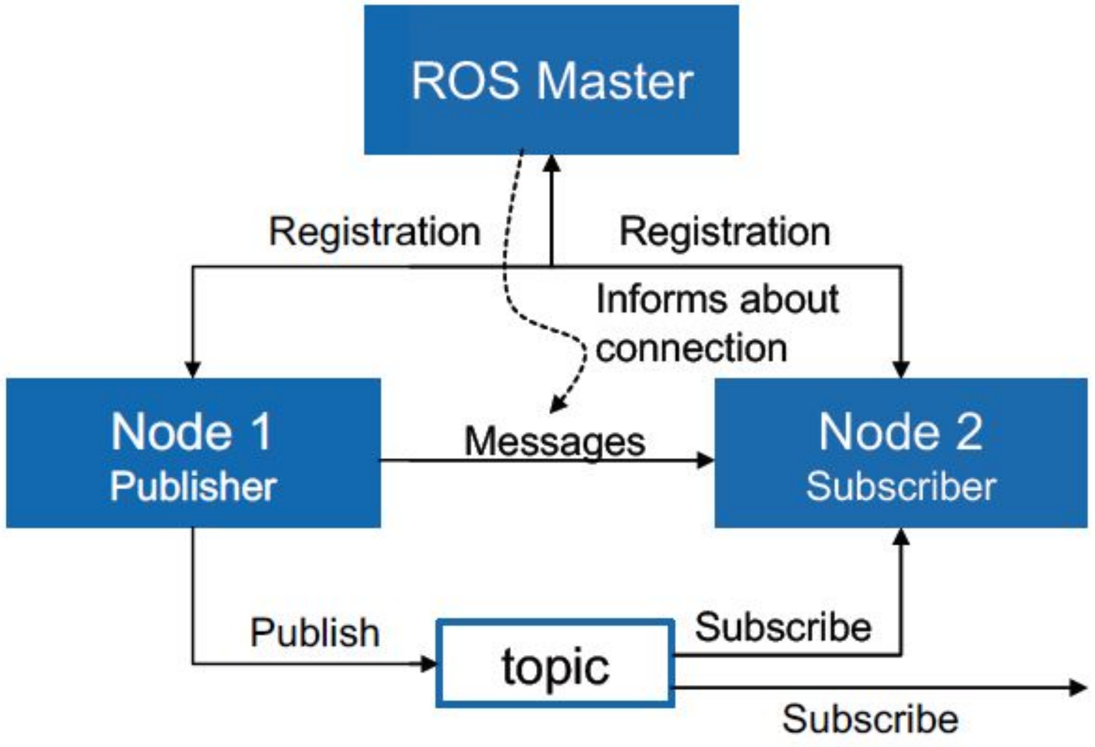
- Remove dead nodes:
  `rosnode cleanup`

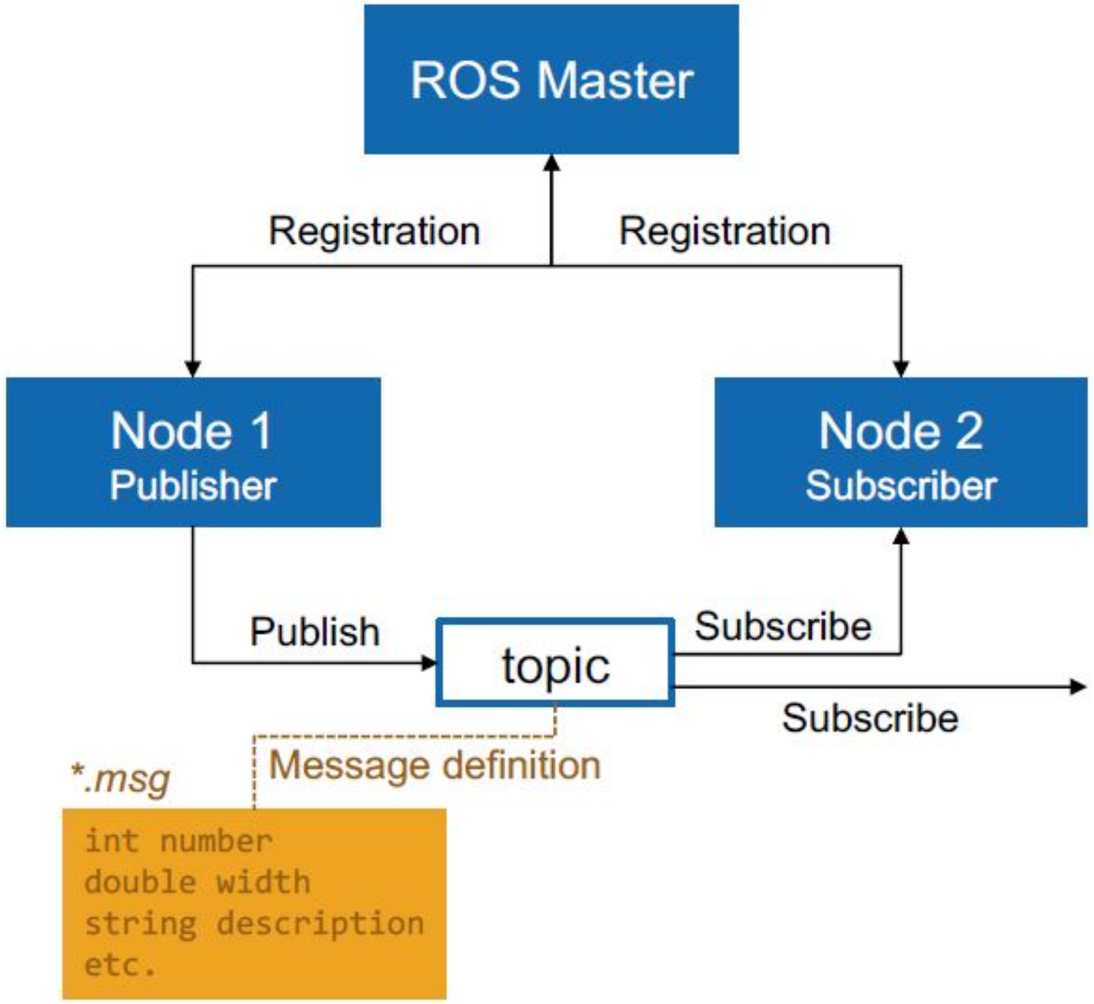
Topics and Messages

- Communication in ROS exploits *messages*
- Messages are organized in *topics*
- A node that wants to share information will *publish* messages on a topic(s)
- A node that wants to receive information will *subscribe* to the topic(s)
- ROS master takes care of ensuring that publishers and subscribers can find each other
- Use of namespaces



Topics and Messages

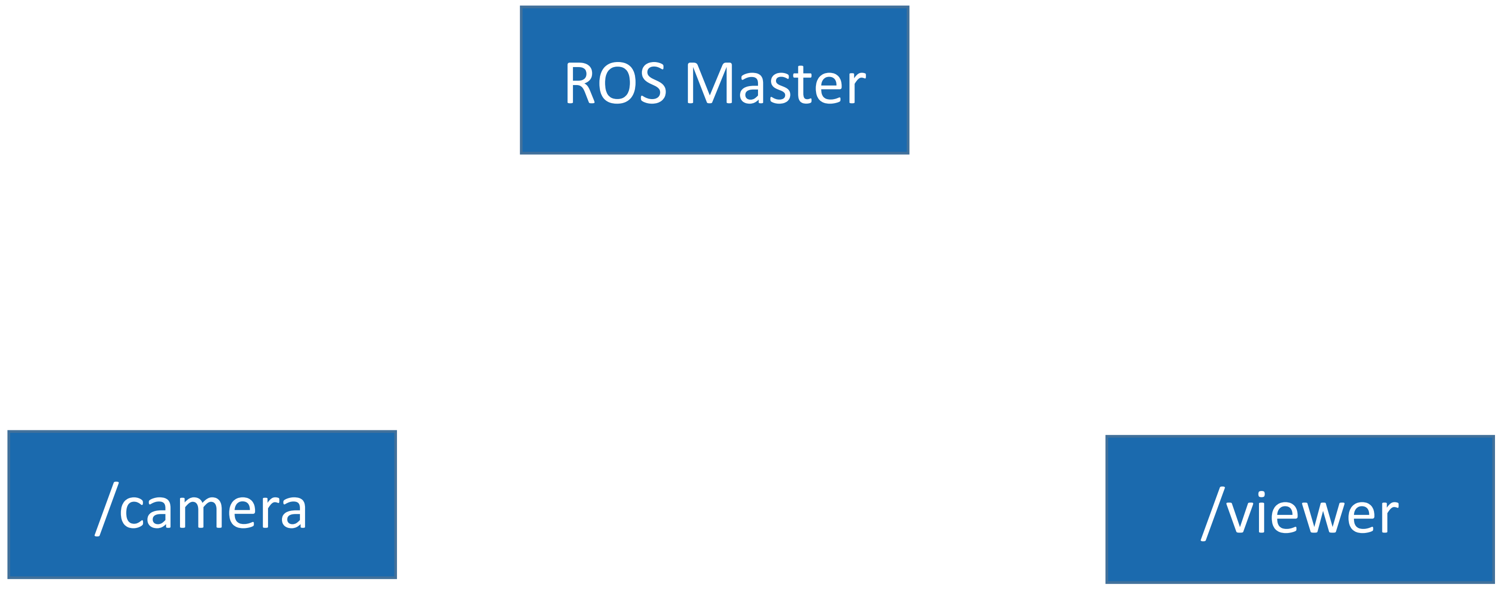




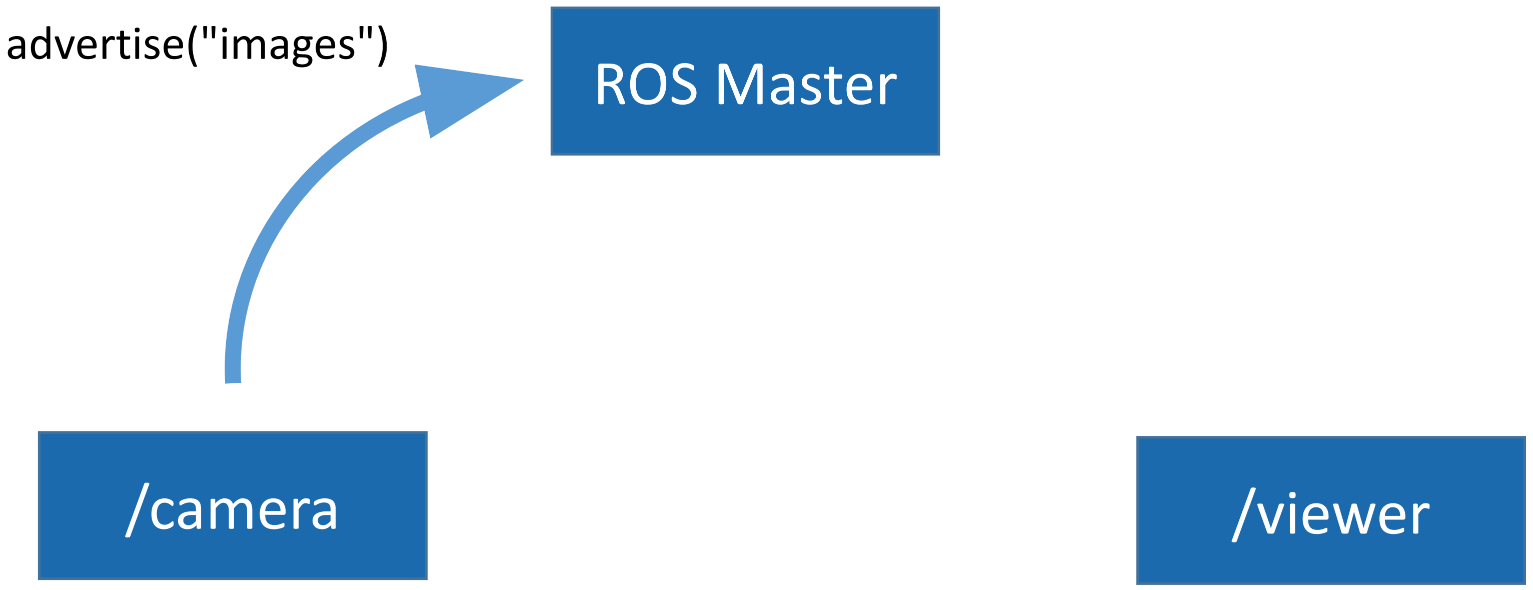
```
int number
double width
string description
etc.
```

Images taken from
Programming for Robotics
Péter Fankhauser, Dominic Jud, Martin Wermelinger, Prof. Dr. Marco Hutter

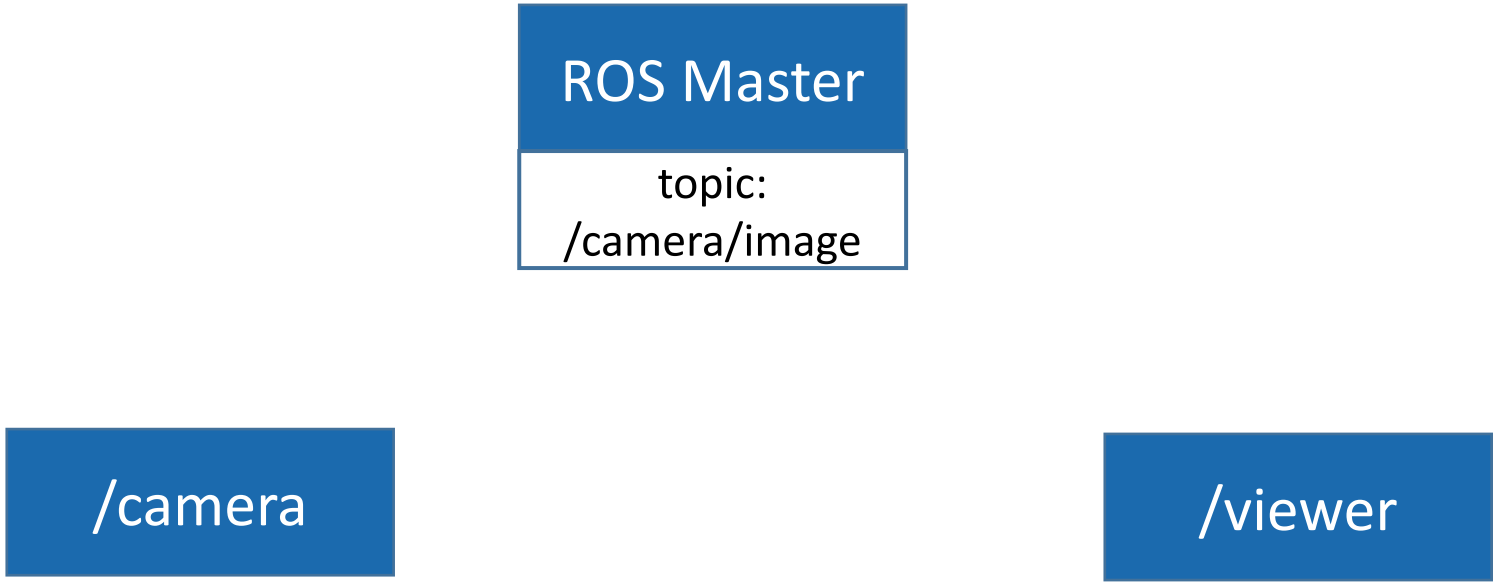
Example



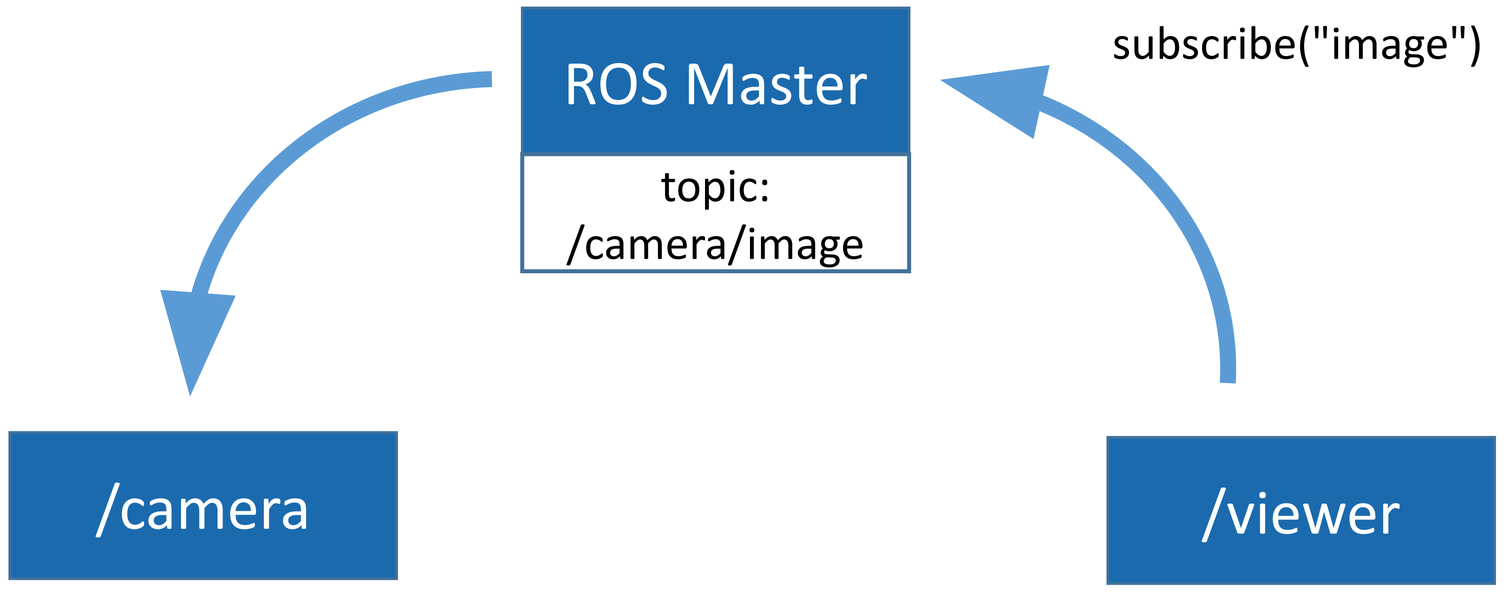
Example



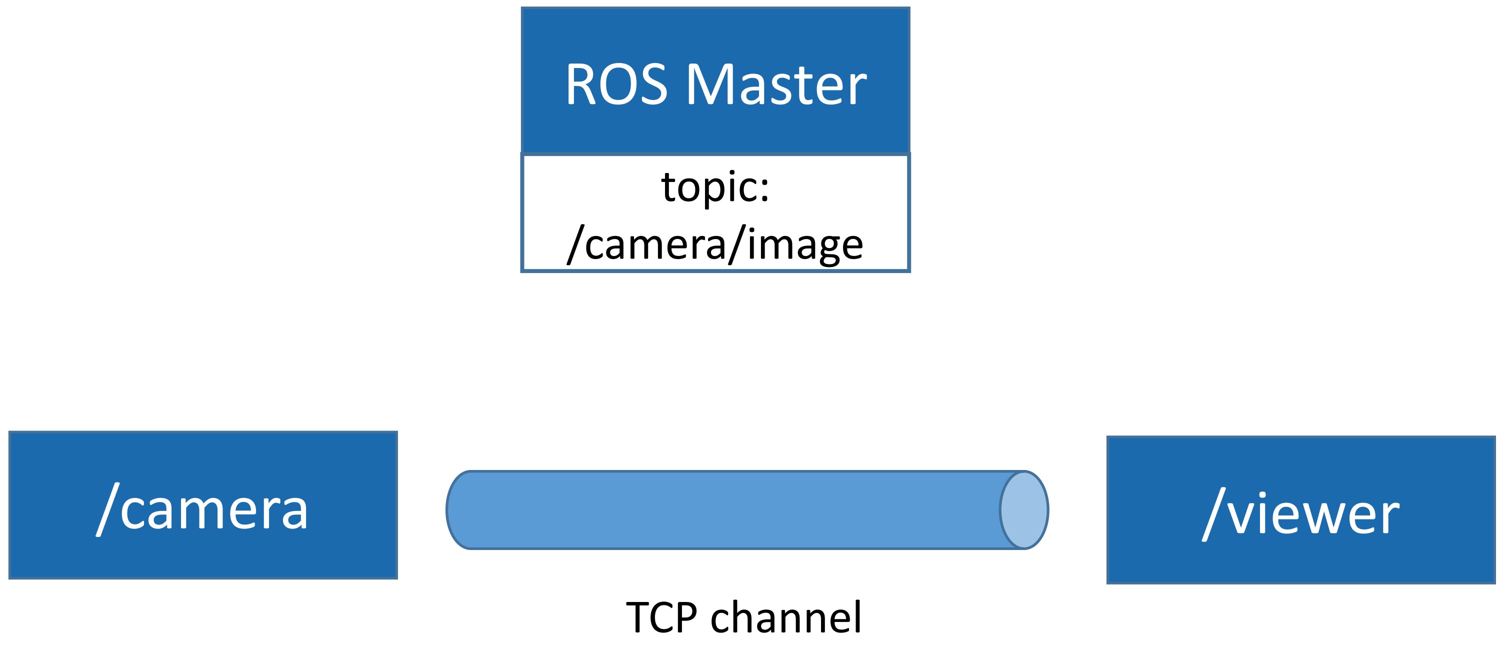
Example



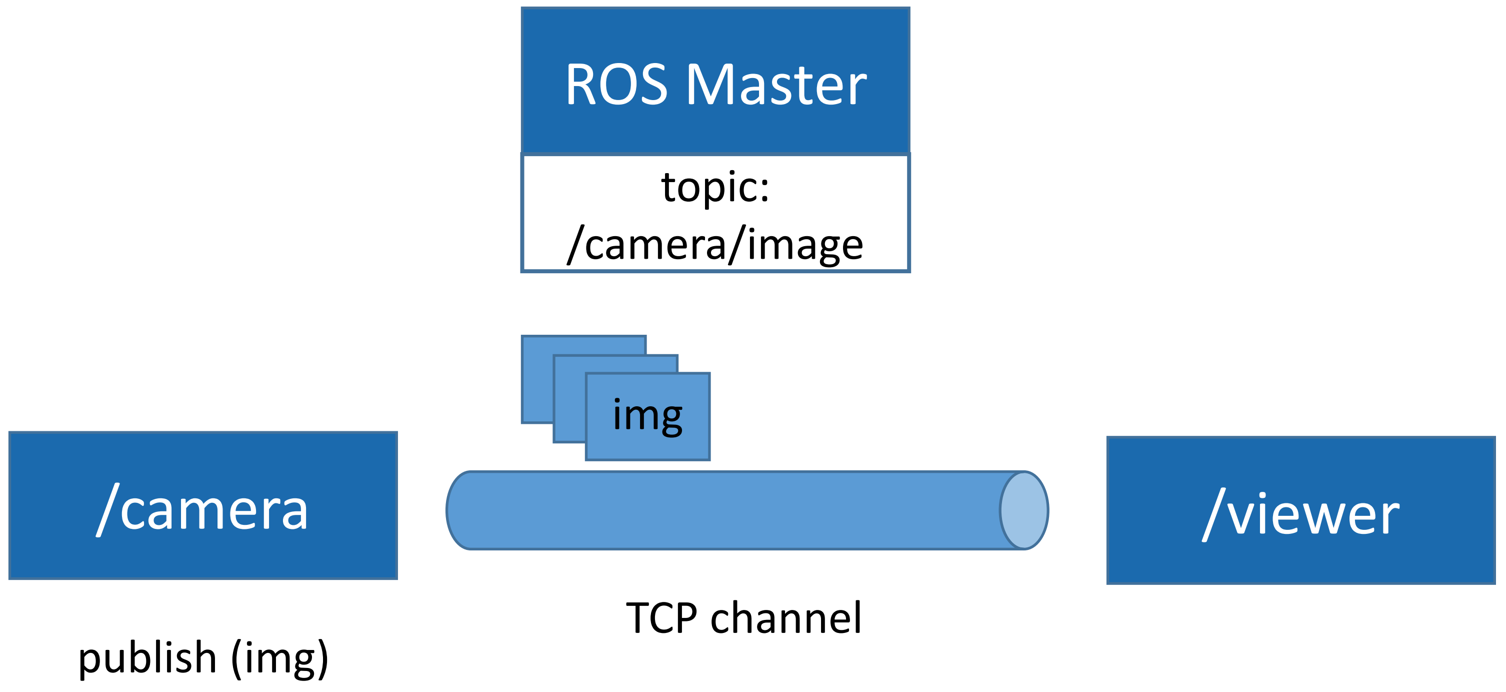
Example


ROS Master
topic:
/camera/image
subscribe("image")
/camera
/viewer

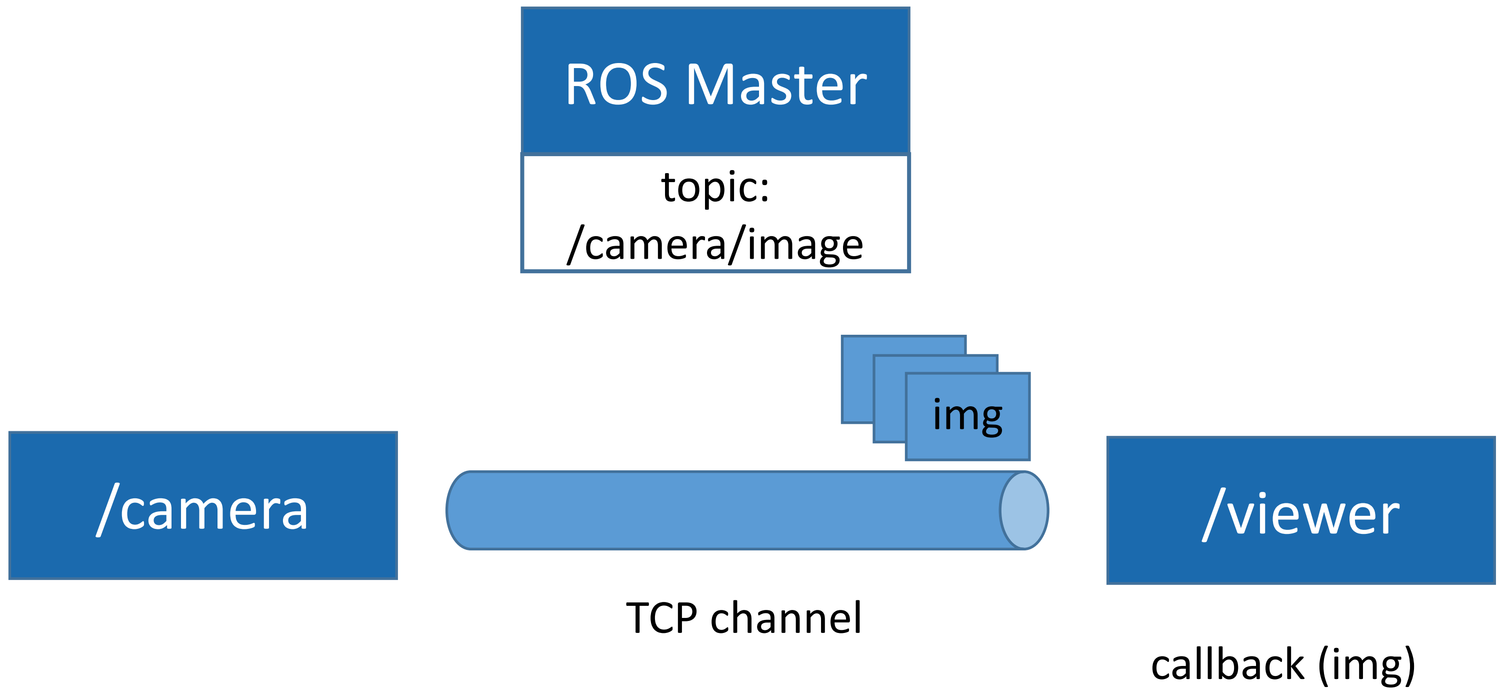
Example



Example


ROS Master
topic:
/camera/image
img
/camera
/viewer
TCP channel
publish (img)

Example

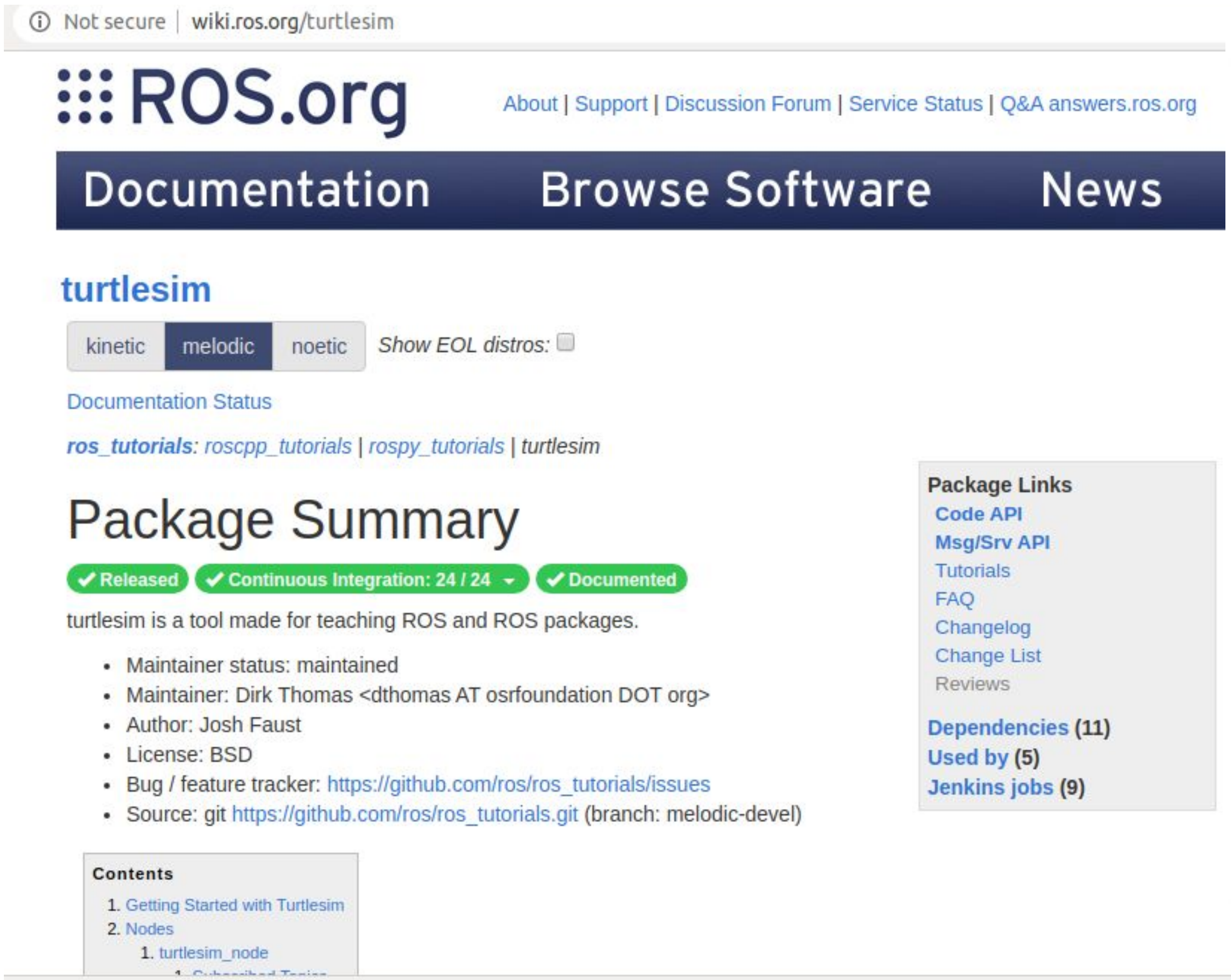

ROS Master
topic:
/camera/image
img
/camera
/viewer
TCP channel
callback (img)

Inspecting topics

- Listing active topics:
```
rostopic list
```
- Seeing all messages published on topic:
```
rostopic echo topic-name
```
- Checking publishing rate:
```
rostopic hz topic-name
```
- Inspecting a topic (message type, subscribers, etc…):
```
rostopic info topic-name
```
- Publishing messages trough terminal line:
```
rostopic pub -r rate-in-hz topic-name
message-type message-content
```

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

TurtleSim



Not secure | wiki.ros.org/turtlesim

ROS.org

About | Support | Discussion Forum | Service Status | Q&A answers.ros.org

Documentation Browse Software News

turtlesim

kinetic [✓ melodic] noetic *Show EOL distros:*

Documentation Status

***ros_tutorials**: roscpp_tutorials | rospy_tutorials | turtlesim*

Package Summary

✓ Released ✓ Continuous Integration: 24 / 24 ✓ Documented

turtlesim is a tool made for teaching ROS and ROS packages.

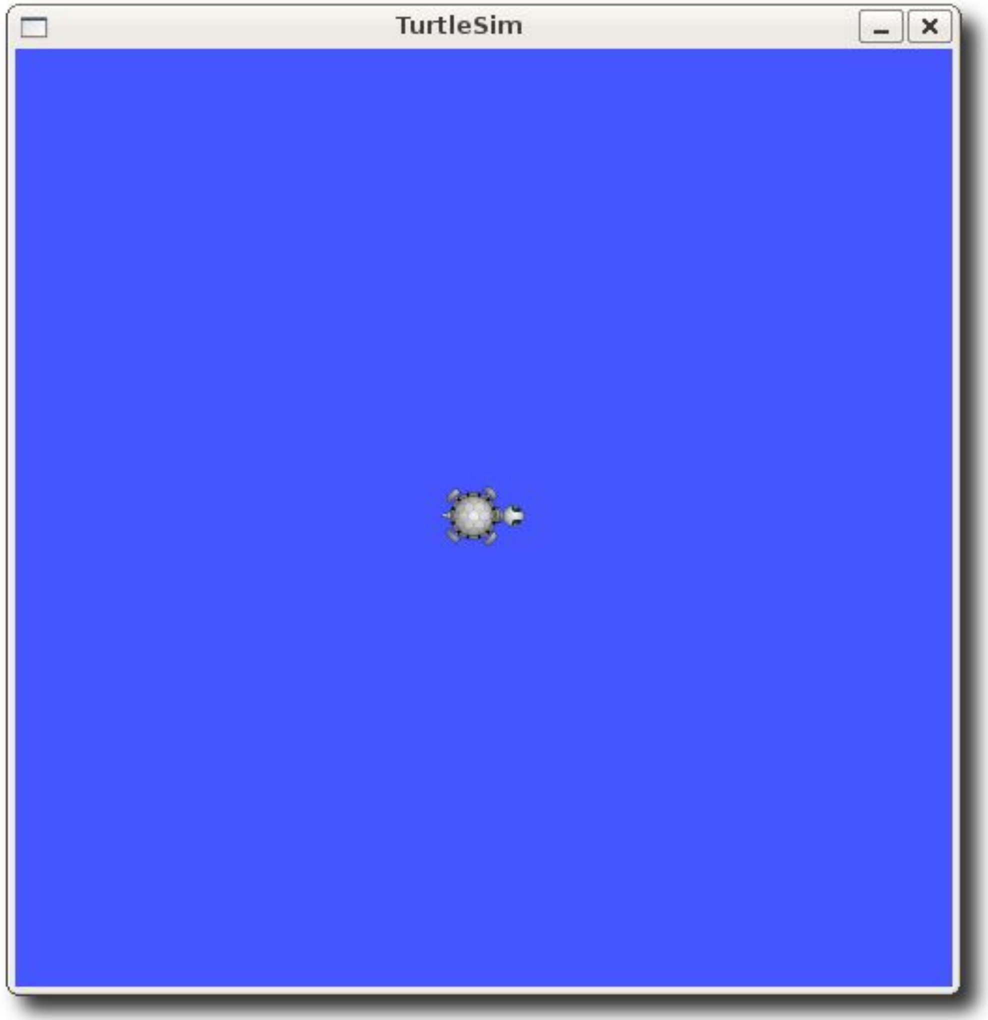
- Maintainer status: maintained
- Maintainer: Dirk Thomas <dthomas AT osrfoundation DOT org>
- Author: Josh Faust
- License: BSD
- Bug / feature tracker: https://github.com/ros/ros_tutorials/issues
- Source: git https://github.com/ros/ros_tutorials.git (branch: melodic-devel)

Contents

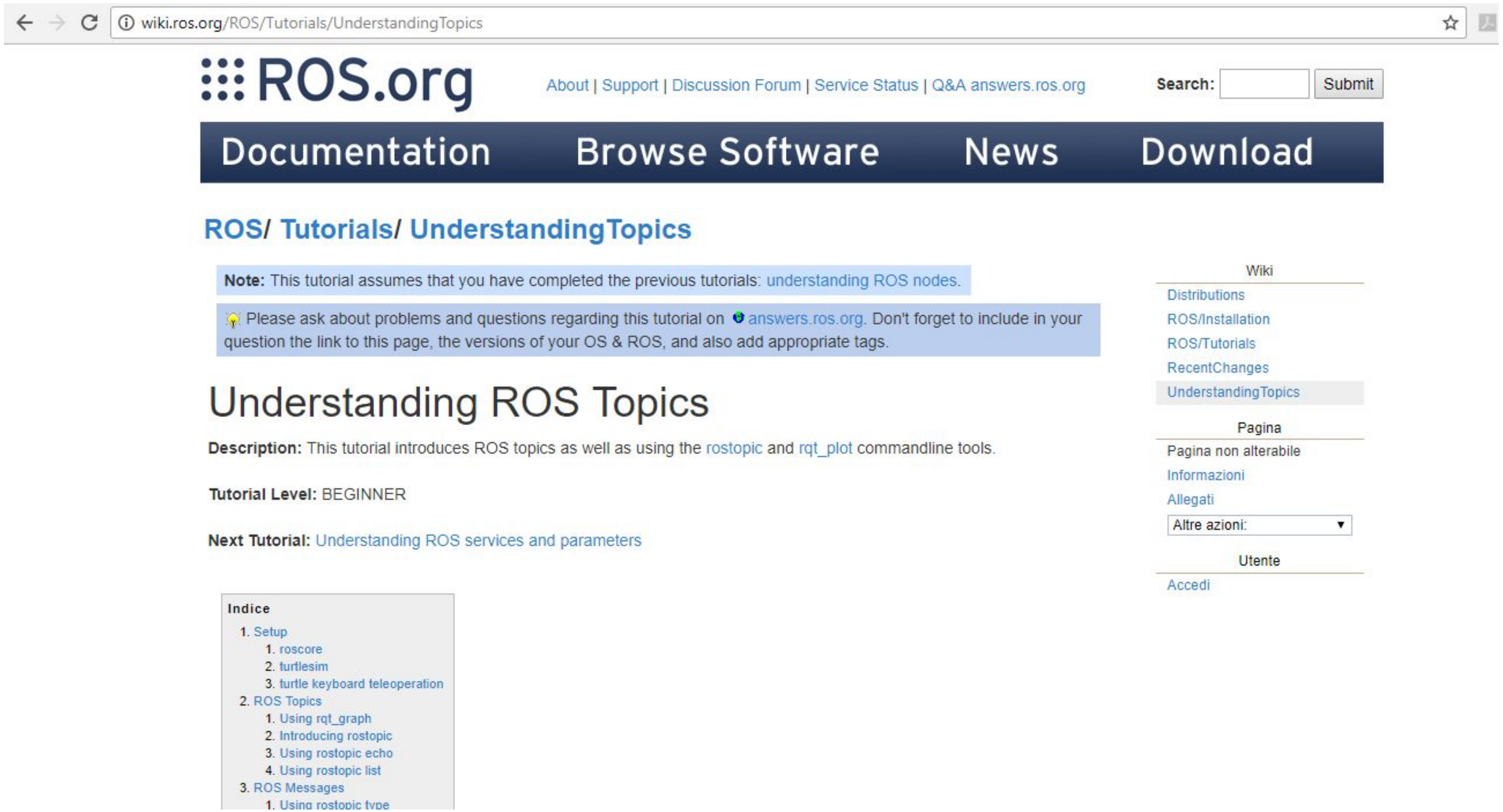
1. Getting Started with Turtlesim
2. Nodes
   1. turtlesim_node

Package Links
Code API
Msg/Srv API
Tutorials
FAQ
Changelog
Change List
Reviews

Dependencies (11)
Used by (5)
Jenkins jobs (9)



Demo TurtleSim



wiki.ros.org/ROS/Tutorials/UnderstandingTopics

ROS.org

About | Support | Discussion Forum | Service Status | Q&A answers.ros.org

Search: Submit

Documentation Browse Software News Download

ROS/ Tutorials/ UnderstandingTopics

Note: This tutorial assumes that you have completed the previous tutorials: understanding ROS nodes.

Please ask about problems and questions regarding this tutorial on answers.ros.org. Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Understanding ROS Topics

Description: This tutorial introduces ROS topics as well as using the rostopic and rqt_plot commandline tools.

Tutorial Level: BEGINNER

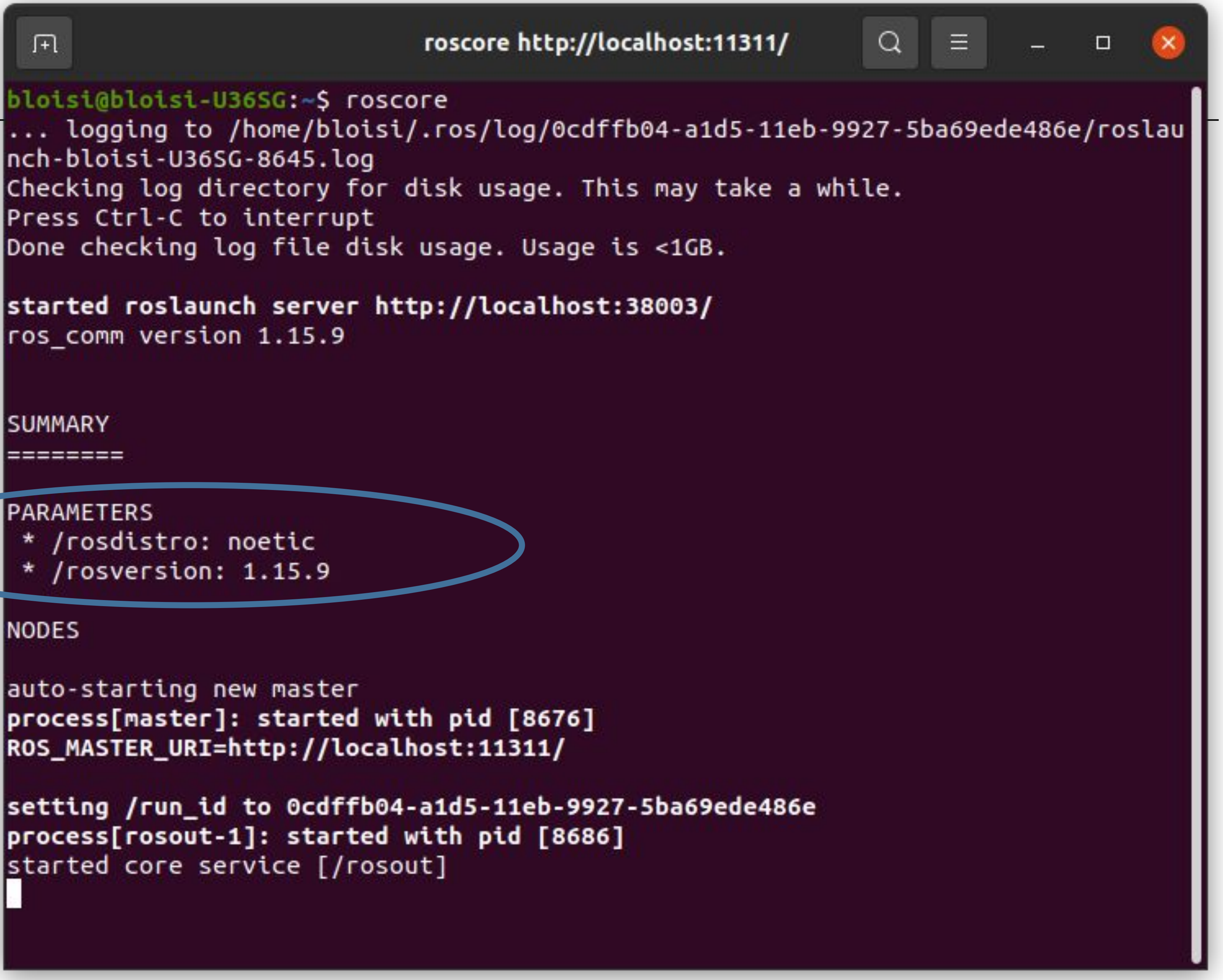
Next Tutorial: Understanding ROS services and parameters

Indice

roscore

- Open a terminal

- digit
  `roscore`



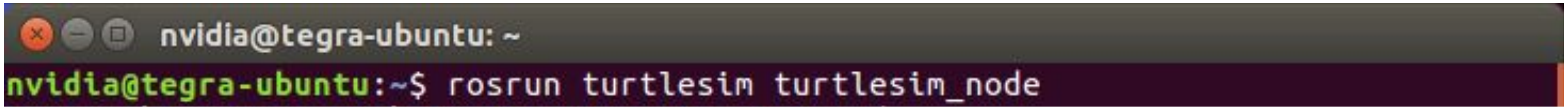
roscore http://localhost:11311/

```
bloisi@bloisi-U36SG:~$ roscore
... logging to /home/bloisi/.ros/log/0cdffb04-a1d5-11eb-9927-5ba69ede486e/roslau
nch-bloisi-U36SG-8645.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:38003/
ros_comm version 1.15.9


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.9

NODES

auto-starting new master
process[master]: started with pid [8676]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to 0cdffb04-a1d5-11eb-9927-5ba69ede486e
process[rosout-1]: started with pid [8686]
started core service [/rosout]
```

Run turtlesim_node

1. Open a **new terminal**

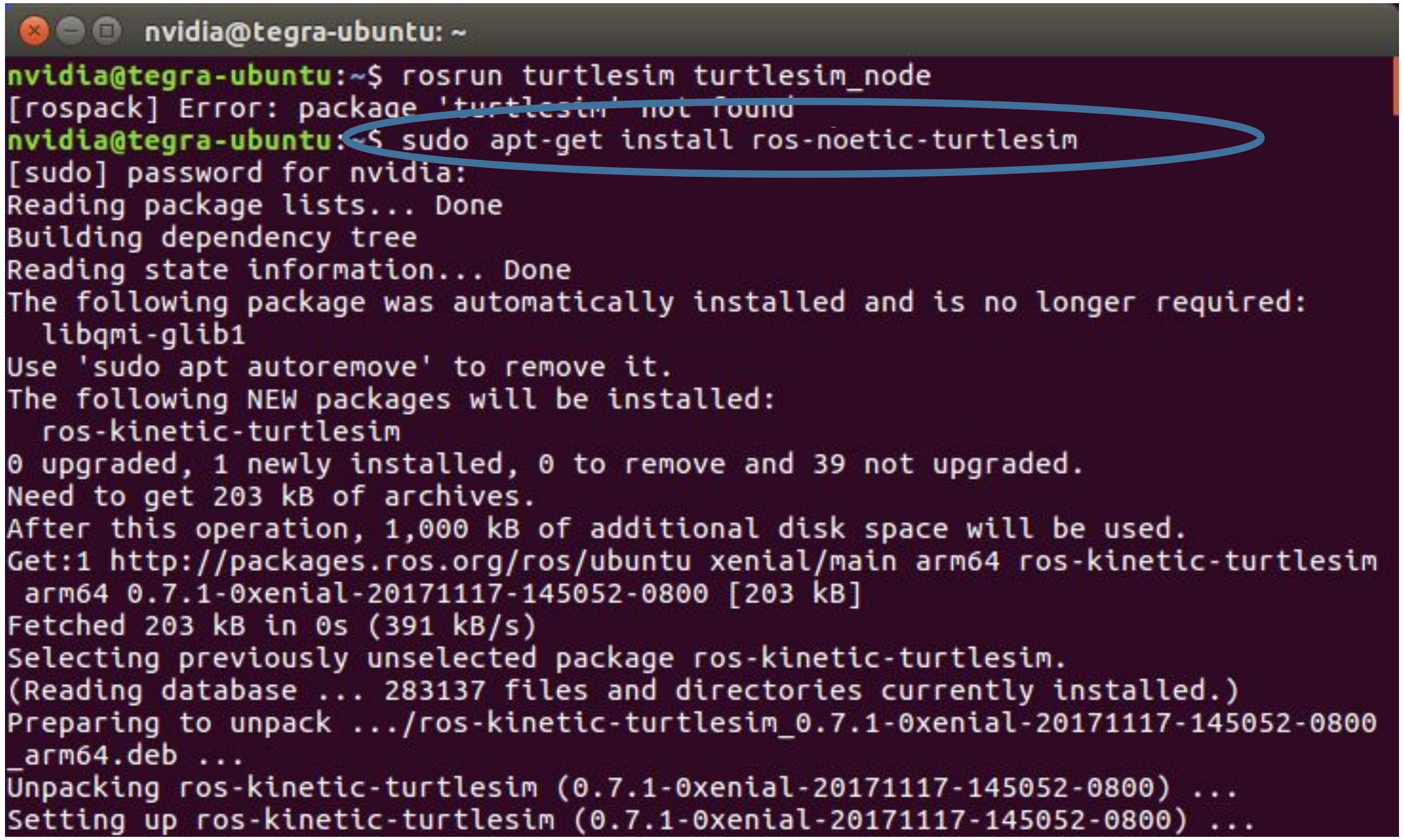
2. run:
```
$ rosrun turtlesim turtlesim_node
```



```
nvidia@tegra-ubuntu:~$ rosrun turtlesim turtlesim_node
```

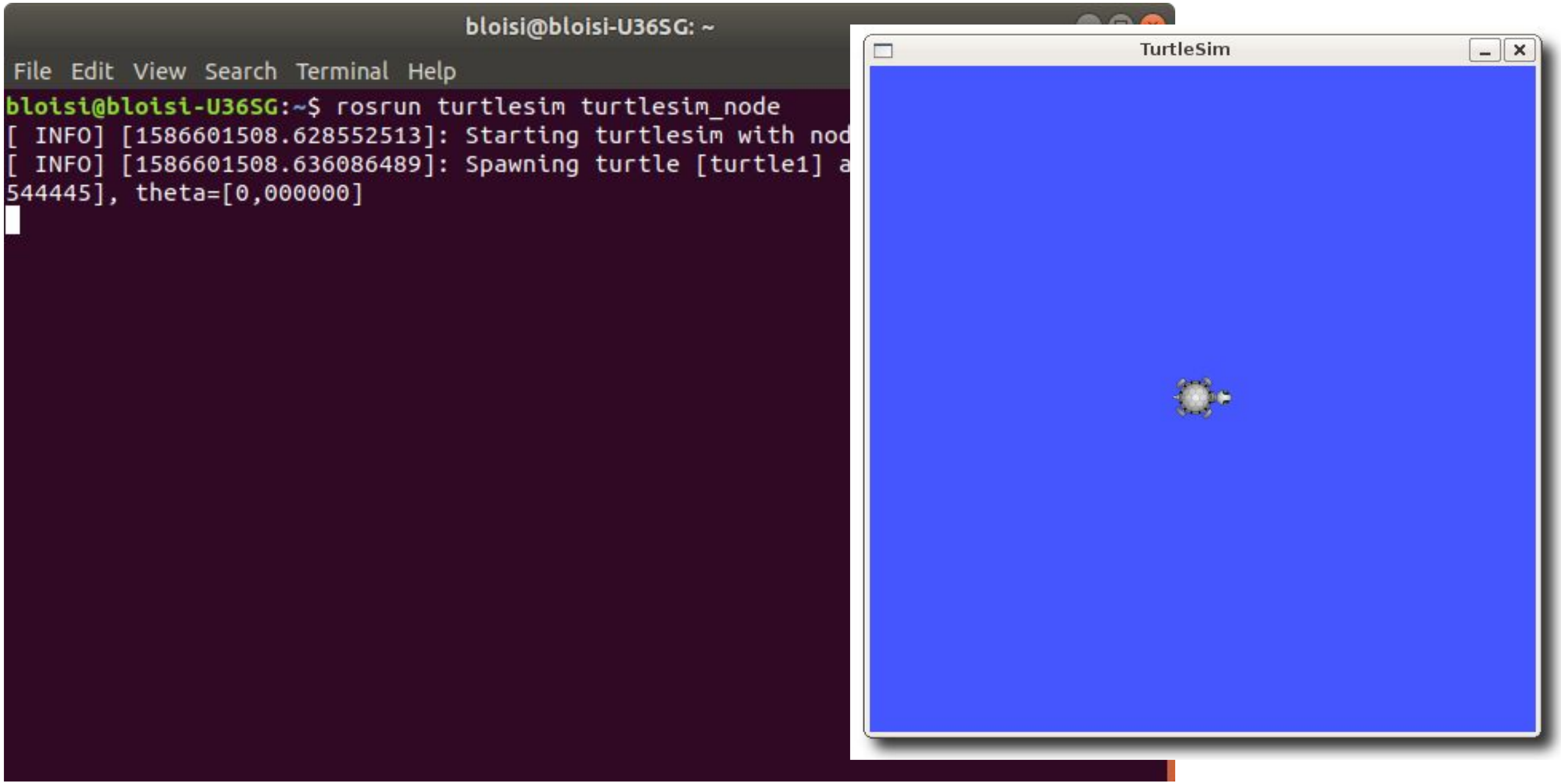
Installing a new package

If package turtlesim is not found, we can install it



```
nvidia@tegra-ubuntu:~$ rosrun turtlesim turtlesim_node
[rospack] Error: package 'turtlesim' not found
nvidia@tegra-ubuntu:~$ sudo apt-get install ros-noetic-turtlesim
[sudo] password for nvidia:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libqmi-glib1
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  ros-kinetic-turtlesim
0 upgraded, 1 newly installed, 0 to remove and 39 not upgraded.
Need to get 203 kB of archives.
After this operation, 1,000 kB of additional disk space will be used.
Get:1 http://packages.ros.org/ros/ubuntu xenial/main arm64 ros-kinetic-turtlesim
 arm64 0.7.1-0xenial-20171117-145052-0800 [203 kB]
Fetched 203 kB in 0s (391 kB/s)
Selecting previously unselected package ros-kinetic-turtlesim.
(Reading database ... 283137 files and directories currently installed.)
Preparing to unpack .../ros-kinetic-turtlesim_0.7.1-0xenial-20171117-145052-0800
_arm64.deb ...
Unpacking ros-kinetic-turtlesim (0.7.1-0xenial-20171117-145052-0800) ...
Setting up ros-kinetic-turtlesim (0.7.1-0xenial-20171117-145052-0800) ...
```

turtlesim_node running



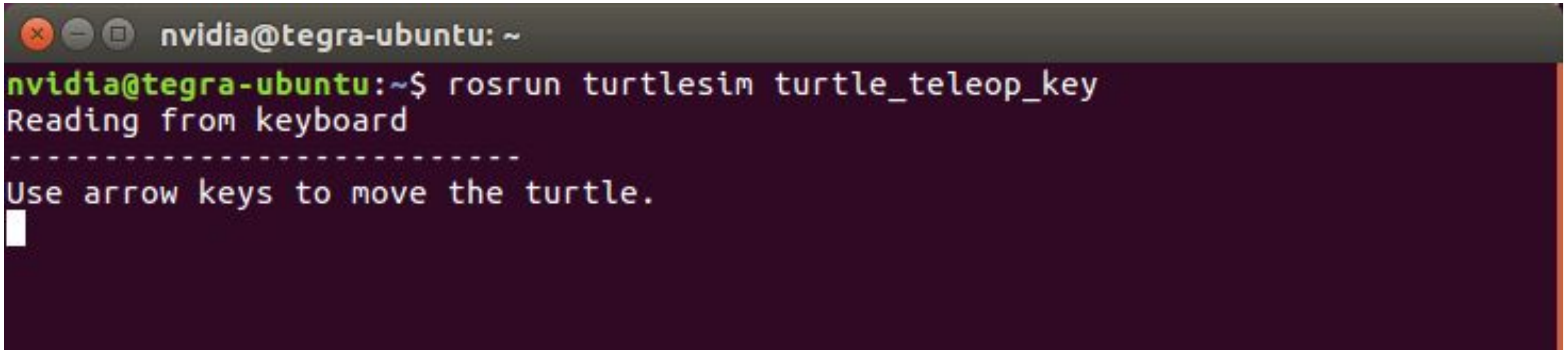
```
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1586601508.628552513]: Starting turtlesim with nod
[ INFO] [1586601508.636086489]: Spawning turtle [turtle1] a
544445], theta=[0,000000]
```

turtle_teleop_key node

1. Open a **new terminal**

2. run:

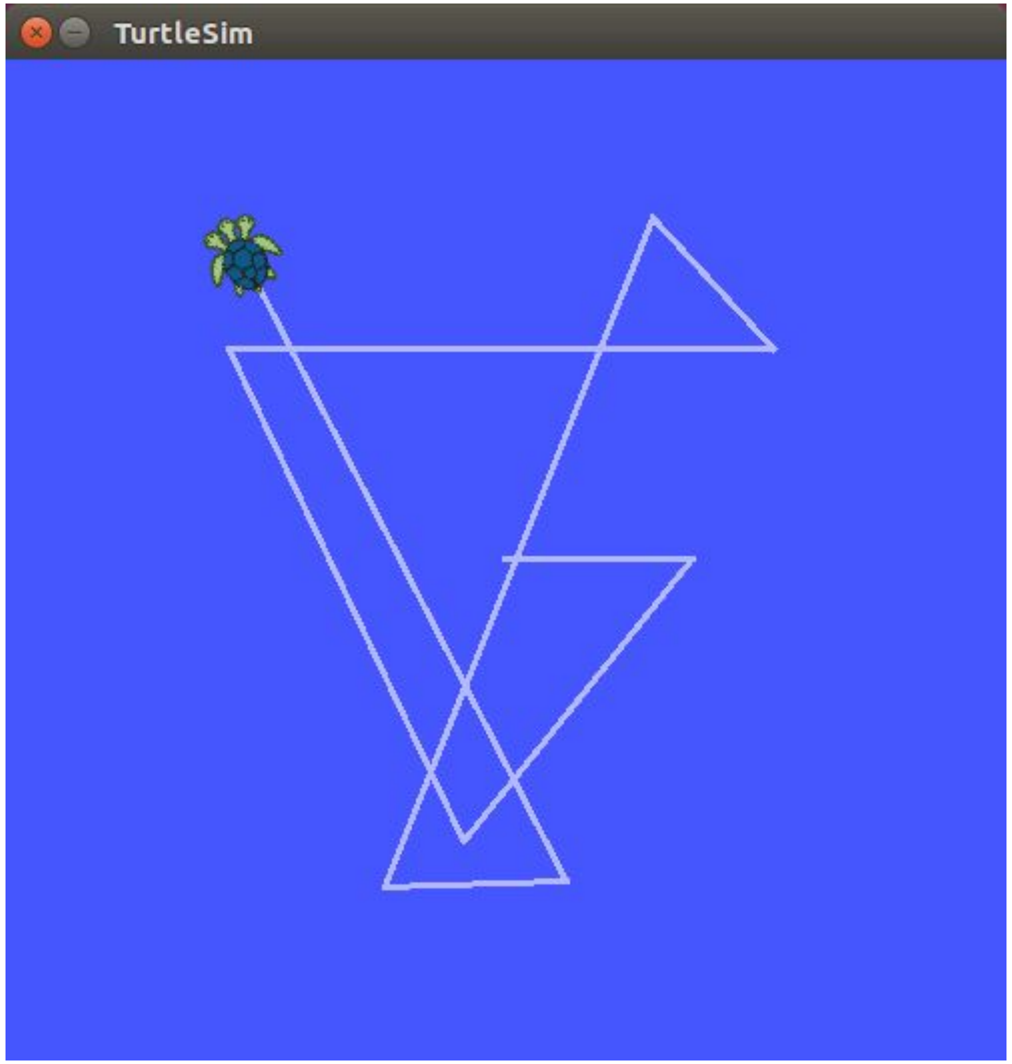
```
$ rosrun turtlesim turtle_teleop_key
```



nvidia@tegra-ubuntu: ~

```
nvidia@tegra-ubuntu:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

Playing with the turtle



ROS filesystem

- **Package**
  unit for organizing software in ROS. Each package can contain libraries, executables, scripts, or other artifacts

- **Manifest** (package.xml)
  meta-information about a package (e.g., version, maintainer, license, etc.) and description of its dependencies (other ROS packages, messages, services, etc.)

http://wiki.ros.org/catkin/package.xml

package.xml

```xml
<?xml version="1.0"?>
<package>
<name>my_package</name>
<version>1.0</version>
<description>My package description</description>
<!-- One maintainer tag required, multiple allowed, one
person per tag -->
<maintainer email="my@mail.com">Your Name</maintainer>
<!-- One license tag required, multiple allowed, one
license per tag. Commonly used license strings: BSD,
MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1,
LGPLv3 -->
<license>LGPLv3</license>
```

Url tags and Author tags

```
<!-- Url tags are optional, but mutiple are allowed, one per tag.
Optional attribute type can be: website, bugtracker, or repository
-->
<url type="website">http://wiki.ros.org/my_package</url>

<!-- Author tags are optional, mutiple are allowed, one per tag.
Authors do not have to be maintianers, but could be -->
<author email="my@mail.com">Your Name</author>

<!-- The *_depend tags are used to specify dependencies.
Dependencies can be catkin packages or system dependencies. Use
build_depend for packages you need at compile time. Use
buildtool_depend for build tool packages. Use run_depend for
packages you need at runtime. Use test_depend for packages you need
only for testing. -->
```

Dependencies

```
<buildtool_depend>catkin</buildtool_depend>

<build_depend>message_generation</build_depend>
<build_depend>roscpp</build_depend>
<build_depend>roslib</build_depend>

<run_depend>message_runtime</run_depend>
<run_depend>roscpp</run_depend>
<run_depend>roslib</run_depend>

<!-- The export tag contains other, unspecified, tags --> <export>
<!-- You can specify that this package is a metapackage here: -->
<!-- <metapackage/> -->
<!-- Other tools can request additional information be placed here -->

</export>
</package>
```

Catkin workspace configuration

```
$ source /opt/ros/noetic/setup.bash
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
$ cd ~/catkin_ws/
$ catkin_make
```

load default workspace

Open ~/.bashrc and add the following lines:

```
#ROS
source ~/catkin_ws/devel/setup.bash
```

overlay your catkin workspace

Catkin workspace

```
catkin_ws/                -- WORKSPACE
  src/               -- SOURCE SPACE
    CMakeLists.txt       -- The 'toplevel' cmake file
    package_1/
      CMakeLists.txt
      package.xml
      ...
    package_n/
      CMakeLists.txt
      package.xml
      ...
  devel/               -- DEVELOPMENT SPACE
  build/               -- BUILD SPACE
```

catkin_make

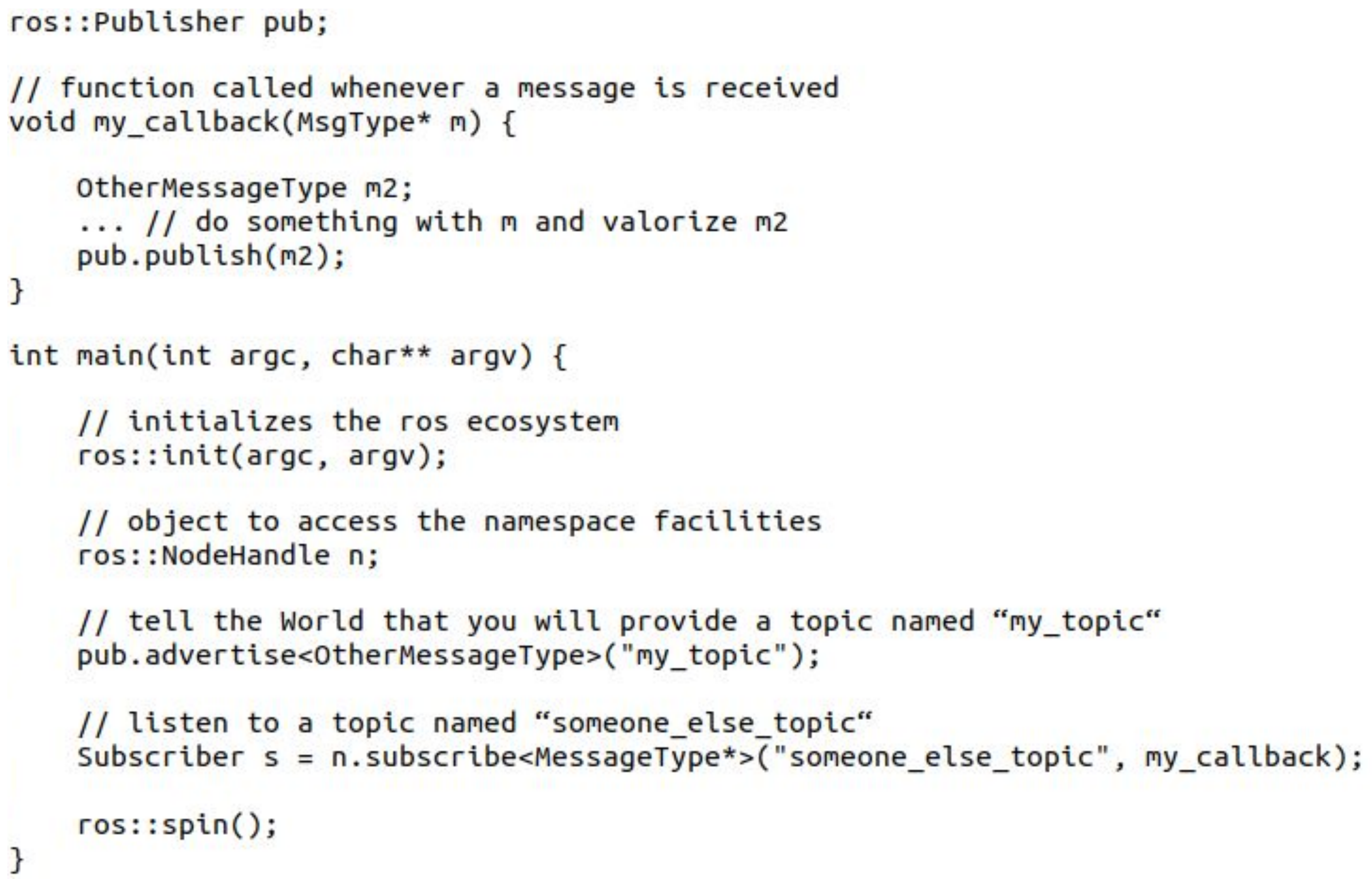
- `catkin_make` is a convenience tool for building code in a catkin workspace

- Execute `catkin_make` in the root of your catkin workspace

- Running the command is equivalent to:

```
$ mkdir build
$ cd build
$ cmake ../src -DCMAKE_INSTALL_PREFIX=../install
-DCATKIN_DEVEL_PREFIX=../devel
$ make
```

Anatomy of a ROS Node



```
ros::Publisher pub;

// function called whenever a message is received
void my_callback(MsgType* m) {

    OtherMessageType m2;
    ... // do something with m and valorize m2
    pub.publish(m2);
}

int main(int argc, char** argv) {

    // initializes the ros ecosystem
    ros::init(argc, argv);

    // object to access the namespace facilities
    ros::NodeHandle n;

    // tell the World that you will provide a topic named “my_topic“
    pub.advertise<OtherMessageType>("my_topic");

    // listen to a topic named “someone_else_topic“
    Subscriber s = n.subscribe<MessageType*>("someone_else_topic", my_callback);

    ros::spin();
}
```

Creating messages

- Messages in ROS are .msg files stored in the corresponding package folder, within the msg dir
- Supported field types are:
  - int8, int16, int32, int64 (plus uint*)
  - float32, float64
  - string
  - time, duration
  - other msg files
  - variable length array [] and fixed length array [C]
  - Header: timestamp and coordinate frame information

Example: creating messages

```
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Exercise 1

Create a message Num.msg with a field `num` of type `int64`

Exercise 1 - Solution

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg new_package std_msgs rospy roscpp
$ source ../devel/setup.bash
$ roscd new_package
$ mkdir msg
$ echo "int64 num" > msg/Num.msg
```

rosbag

- A bag is a serialized message data in a file

- rosbag for recording or playing data

  `rosbag record –a` Record all the topics

  `rosbag info` *`bag-name`* Info on the recorded bag

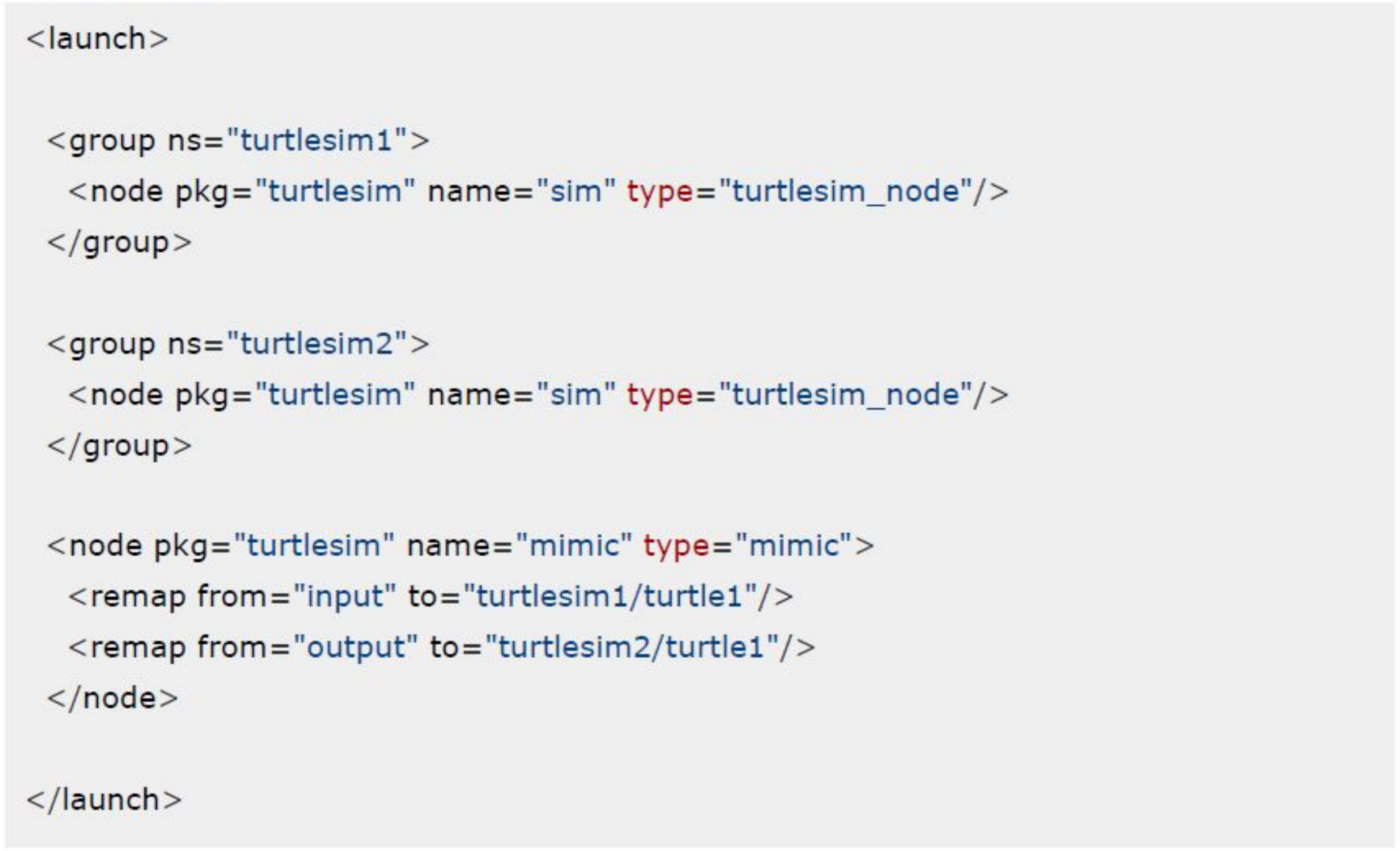
  `rospag play --pause` *`bag-name`* Play the recorded bag, starting paused

  `rospag play -r` *`#number bag-name`* Play the recorded bag at rate #number

roslaunch

The ROS master and the nodes can be activated all at once, using a launch file

See details at:
http://wiki.ros.org/roslaunch/XML



```
<launch>

  <group ns="turtlesim1">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
  </group>

  <group ns="turtlesim2">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
  </group>

  <node pkg="turtlesim" name="mimic" type="mimic">
    <remap from="input" to="turtlesim1/turtle1"/>
    <remap from="output" to="turtlesim2/turtle1"/>
  </node>

</launch>
```

```
roslaunch package-name launch-file-name
```

Exercise 2

- Follow the ROS beginner tutorials:
  – Build and run the “Simple Publisher and Subscriber”

- Modify the talker node and the listener node
  1. Publish the message Num (created earlier) on the topic oddNums:
     - the message Num should be sent if the variable count is odd
     - Num should contain the value of count
  2. Additionally subscribe to topic oddNums
  3. Create a callback function oddNumsCallback to print the content of the received message

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Visione e Percezione

Docente

Domenico D. Bloisi

ROS intro