Intelligent Robots Lab 01

PROF. QI HAO

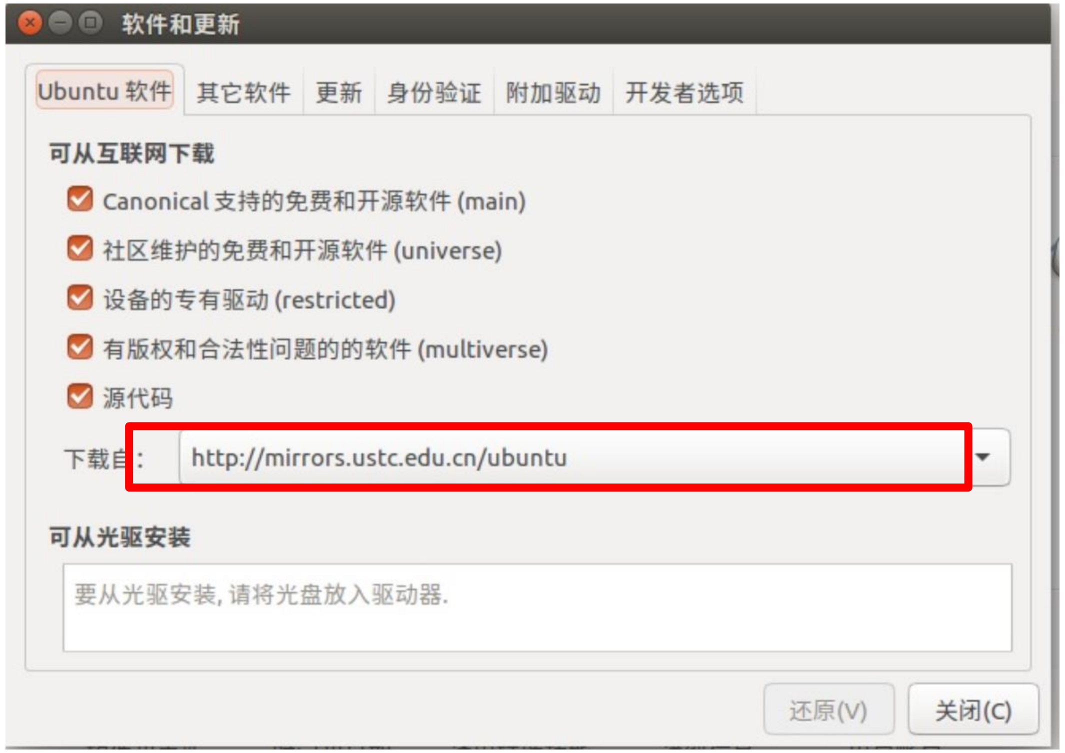
02/26/2019

- Be familiar with ROS

Install ROS

You may have already called you turtle out and moved it around!

- Attention 1:



软件和更新

Ubuntu 软件 | 其它软件 | 更新 | 身份验证 | 附加驱动 | 开发者选项

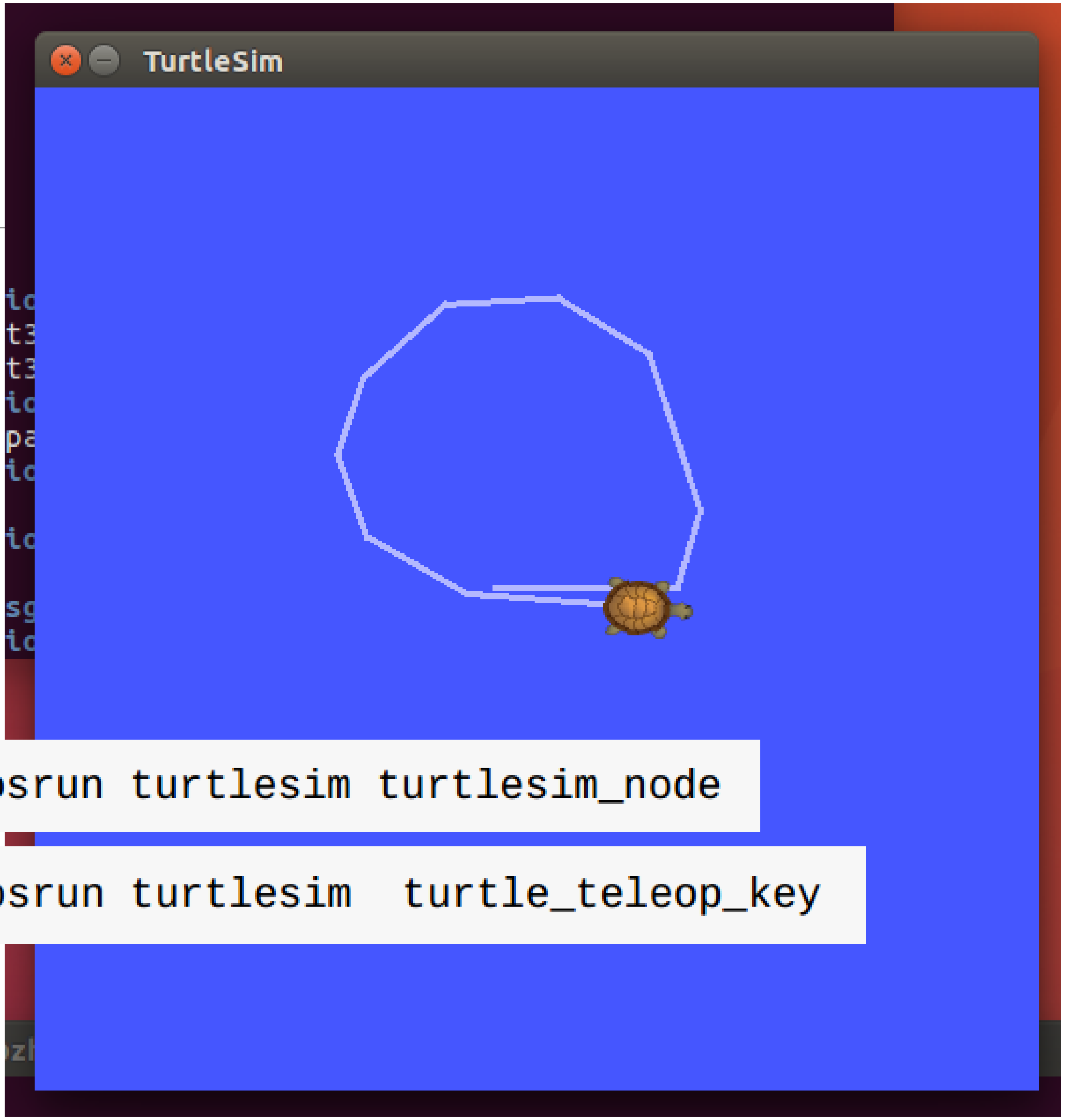
可从互联网下载

- [✓] Canonical 支持的免费和开源软件 (main)
- [✓] 社区维护的免费和开源软件 (universe)
- [✓] 设备的专有驱动 (restricted)
- [✓] 有版权和合法性问题的的软件 (multiverse)
- [✓] 源代码

下载自: http://mirrors.ustc.edu.cn/ubuntu

可从光驱安装

要从光驱安装, 请将光盘放入驱动器.



```
$ rosrun turtlesim turtlesim_node
```

```
$ rosrun turtlesim  turtle_teleop_key
```

Install ROS

- Attention 2:

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F7
17815A3895523BAEEB01FA116
```

Try these two by substituting for the above
hkp://pgp.mit.edu:80 or *hkp://keyserver.ubuntu.com:80*

- Attention 3:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Make sure the system can find your installation directory every time you open a new shell.

ROS Workspace Environment

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

```
source /opt/ros/kinetic/setup.bash
```

  , which we have already set up.
- Add your own workspace to environment, like */catkin_ws*

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

- See the difference using the following command before and after run the command above

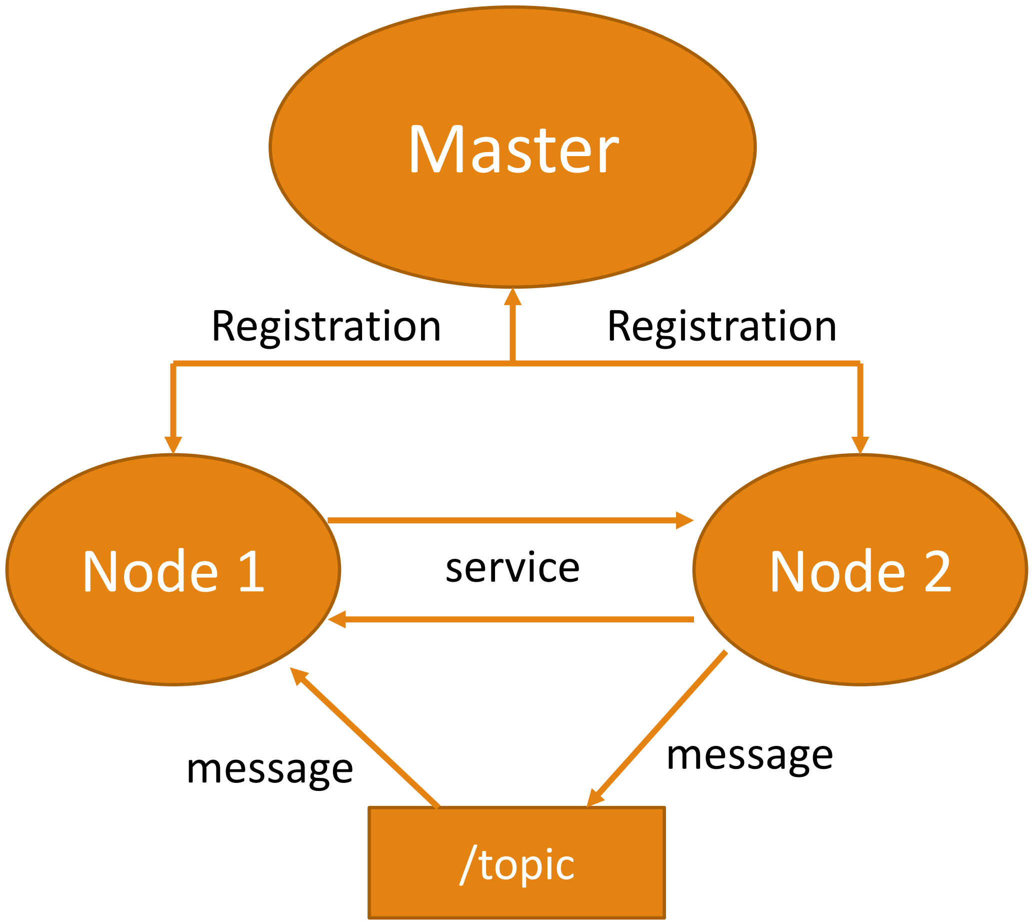
```
$ echo $ROS_PACKAGE_PATH
```

ROS Master and Nodes

- Every node has to register with Master, than communicate with other nodes peer to peer.
- Each node is a single-purpose, executable program (C++/Python) in a package. Run a node as:

```
$ rosrun [package_name] [node_name]
```

- Nodes are individually compiled, executed, and managed. A whole task is divided into many nodes.



Useful Tools of *rosnode*

- ROS provide powerful tools to check information of nodes

***rosnode* command**	**Functions**
rosnode list	List active nodes
rosnode info [node_name]	Print information about node
rosnode kill [node_name]	Kill a running node
rosnode cleanup	Purge registration information of unreachable nodes
rosnode ping	Test connectivity to node
rosnode machine	List nodes running on a particular machine or list machines

- How to get these command?

```
$ rosnode -h
```

or

```
$ rosnode <tab><tab>
```

ROS Communication among Nodes

- There are four main communication methods in ROS:

  1. Topic
  2. Service
  3. Parameter service
  4. actionlib

- First two are the most common used.

ROS Topics

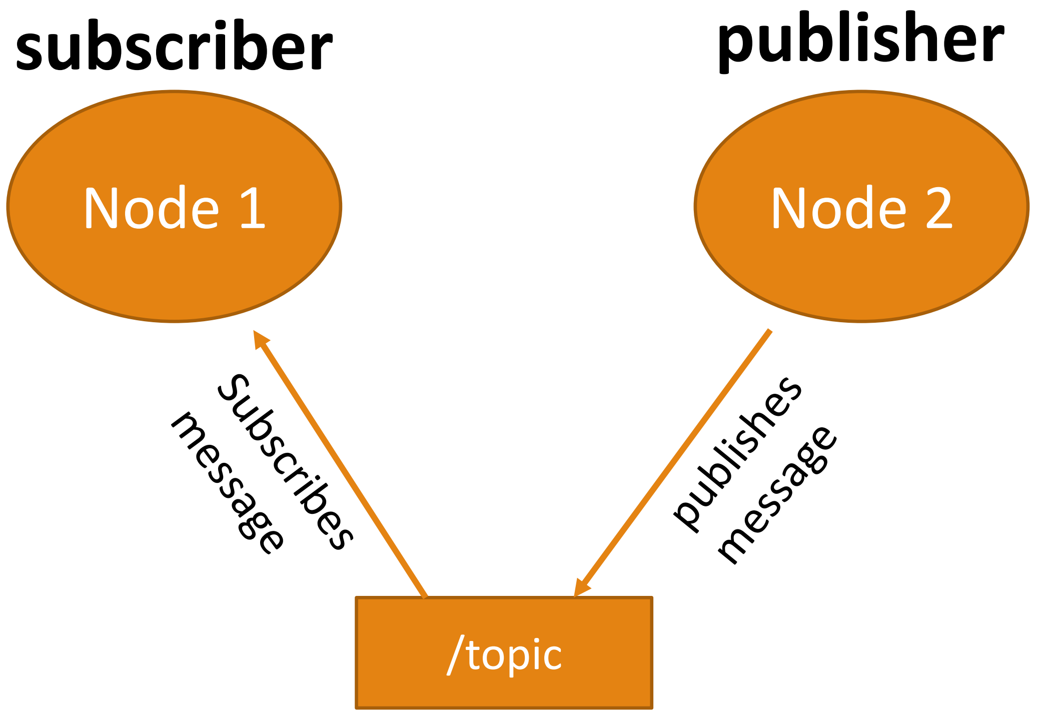
- Topic is a name for a stream of *messages.*
- Node who publishes messages to a certain /topic called publisher.
- Node who subscribes messages from a certain /topic called publisher.
- A node can both be a publisher and subscriber simultaneously, or publish to/subscribe from many topics.
- **Useful tools:**

try

```
$ rostopic -h
```

or

```
$ rostopic <tab><tab>
```



△ ROS network remove other things

ROS Messages

- Each /topic has a specific format to regular the messages transported on it.
- The format is written in a .msg file in directory msg/.
- Data types in ROS: http://wiki.ros.org/msg
- **Useful tools:**

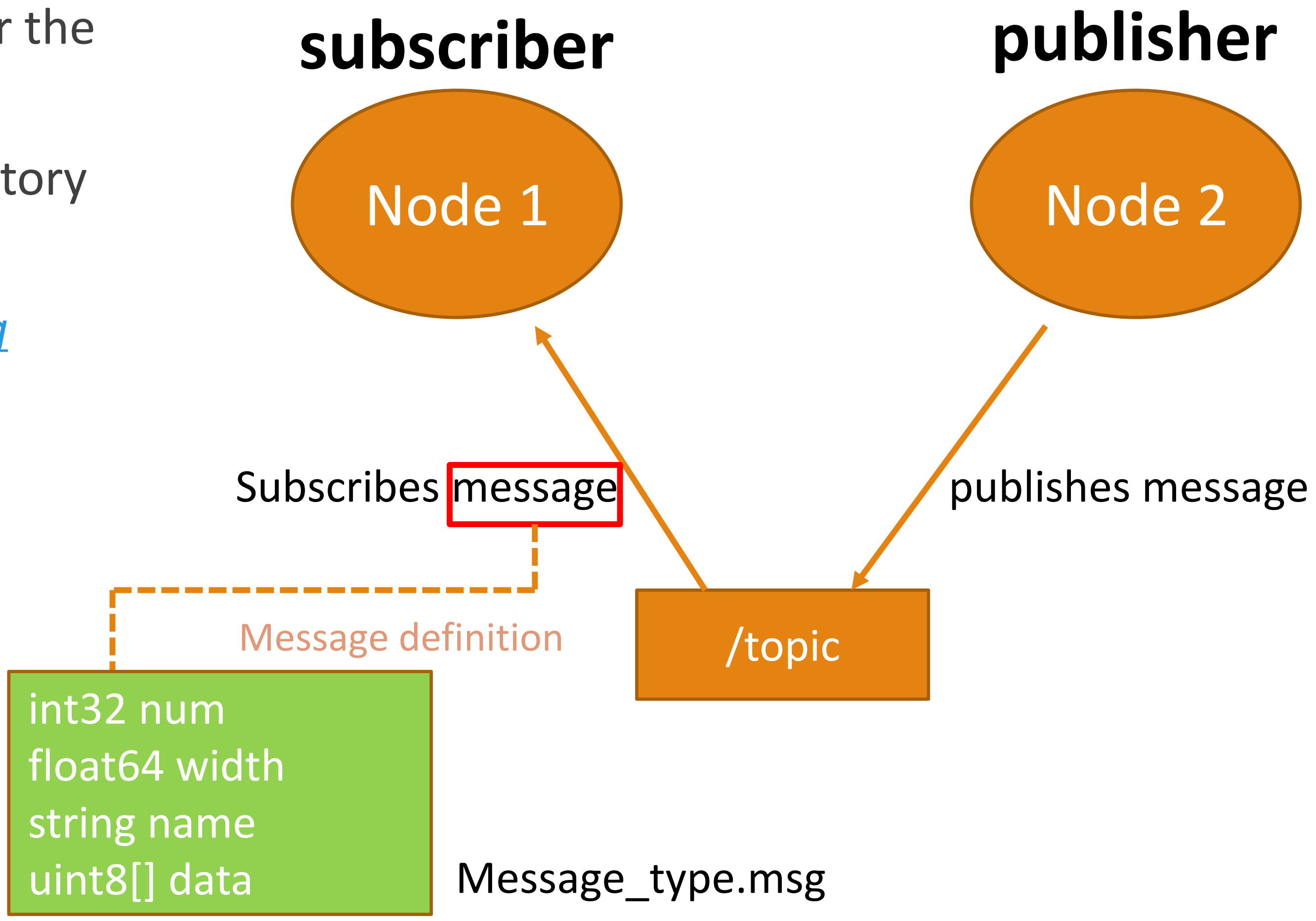
```
$ rosmsg -h
```

or

```
$ rosmsg <tab><tab>
```

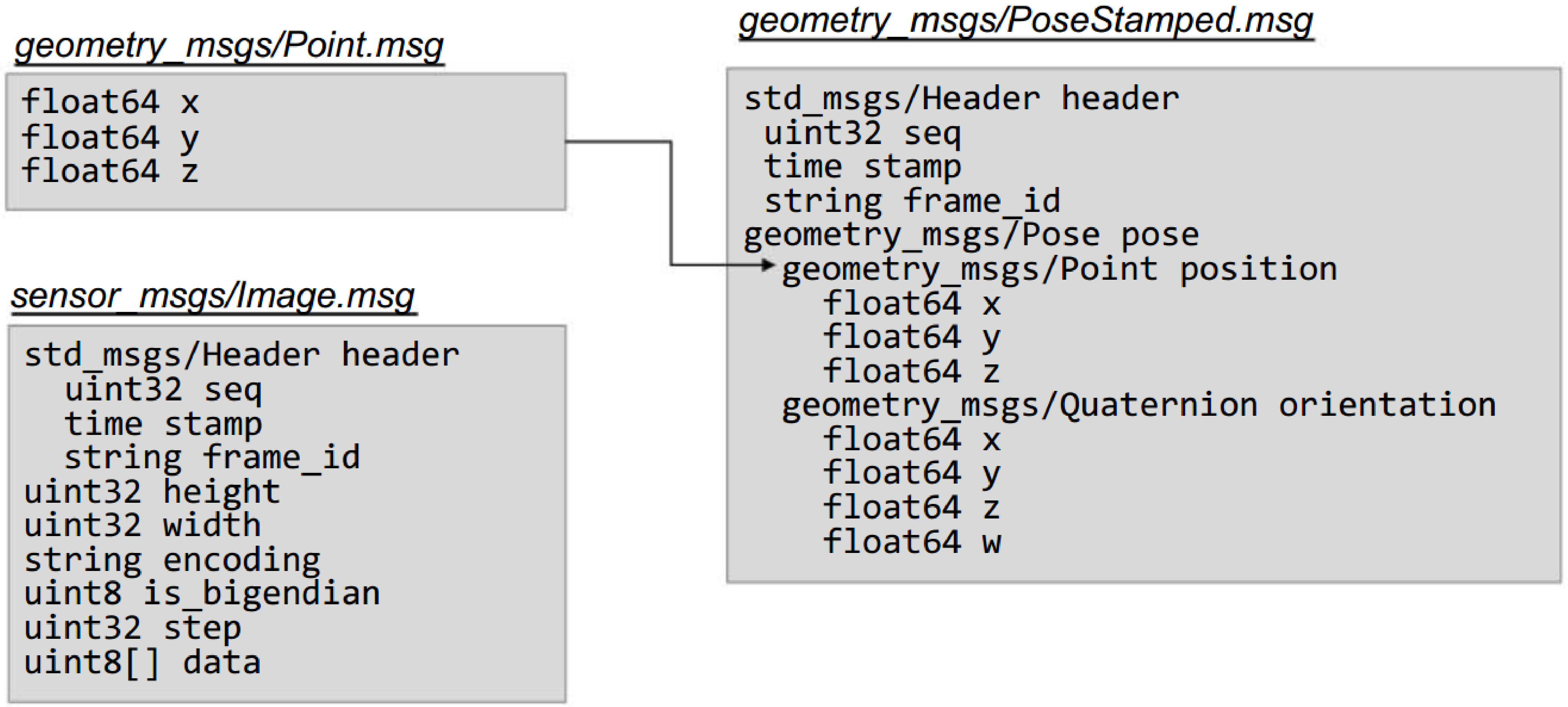
See the type of a topic

```
$ rostopic type /topic_name
```



```
int32 num
float64 width
string name
uint8[] data
```

ROS Messages Examples



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
```

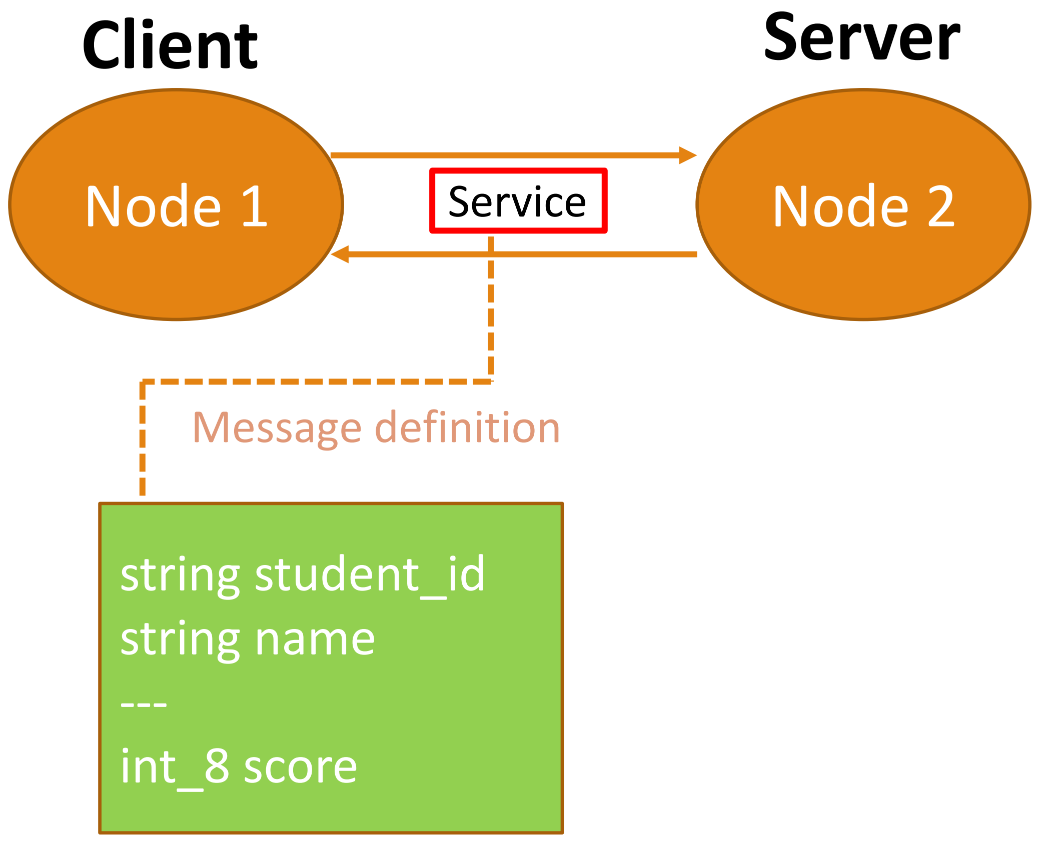
ROS Service

- One node called client will call another node called server and get the feedback.
- Service format is defined by .srv files in directory srv/. The request and response are separated by "---".
- **Useful tools:**

```
$ rossrv -h
$ rossrv <tab><tab>
$ rosservice -h
$ rosservice <tab><tab>
```

- Similar tools:

```
$ rospack
$ rosparam
```



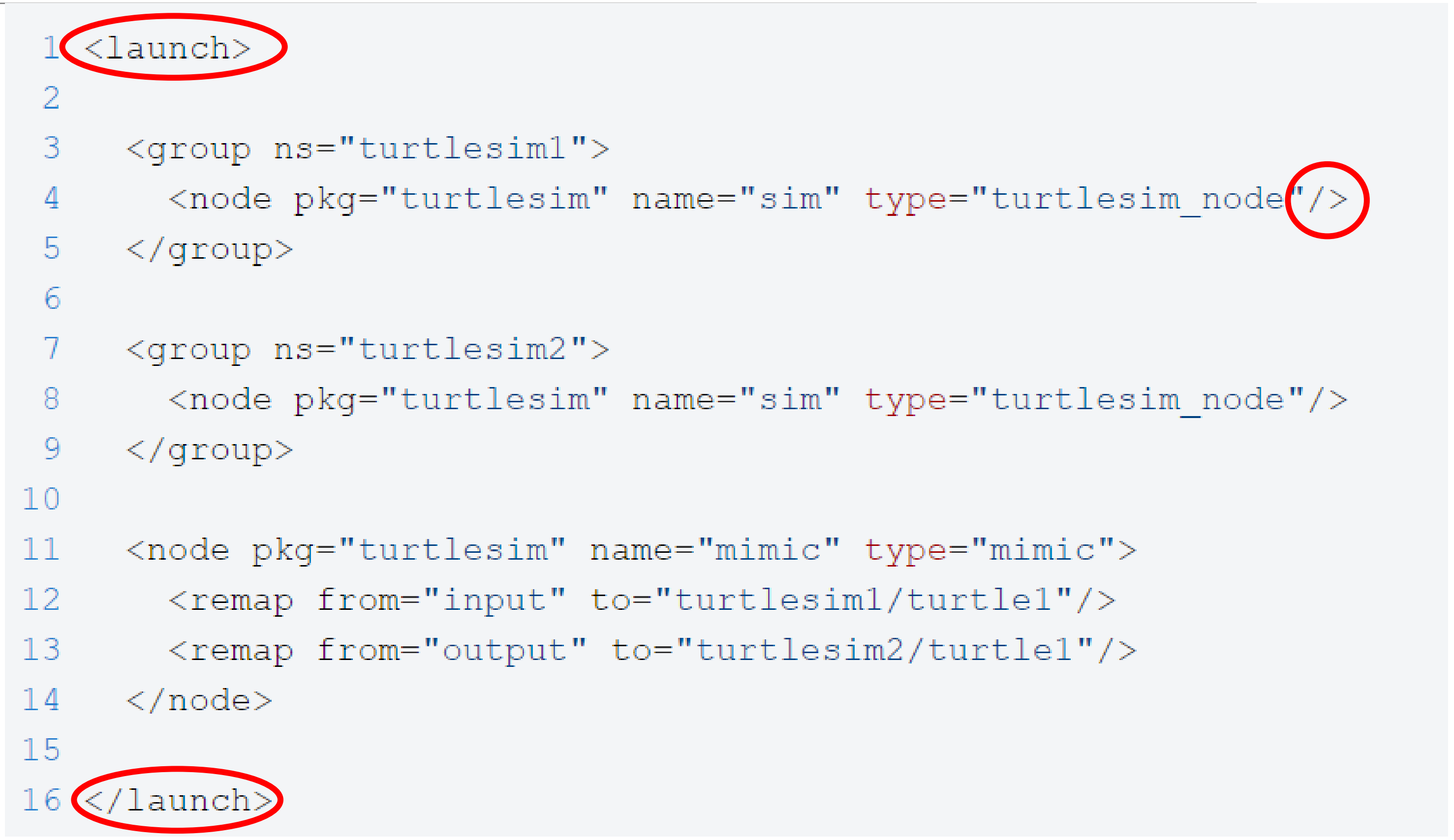
```
string student_id
string name
---
int_8 score
```

ROS Launch

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

- **File Structure** ⇨
- *launch* is a tool for launching multiple nodes (as well as setting parameters) , including Master if not running.
- Are written in XML as **.launch* files
- Start a launch file from a package with

```
$ roslaunch [package_name] [file_name.launch]
```



```
<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>
```

More info
http://wiki.ros.org/roslaunch/XML
http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20tips%20for%20larger%20projects

ROS Launch : Aguments

- Arguments make launch file reusable, which works like a parameter (default optional)

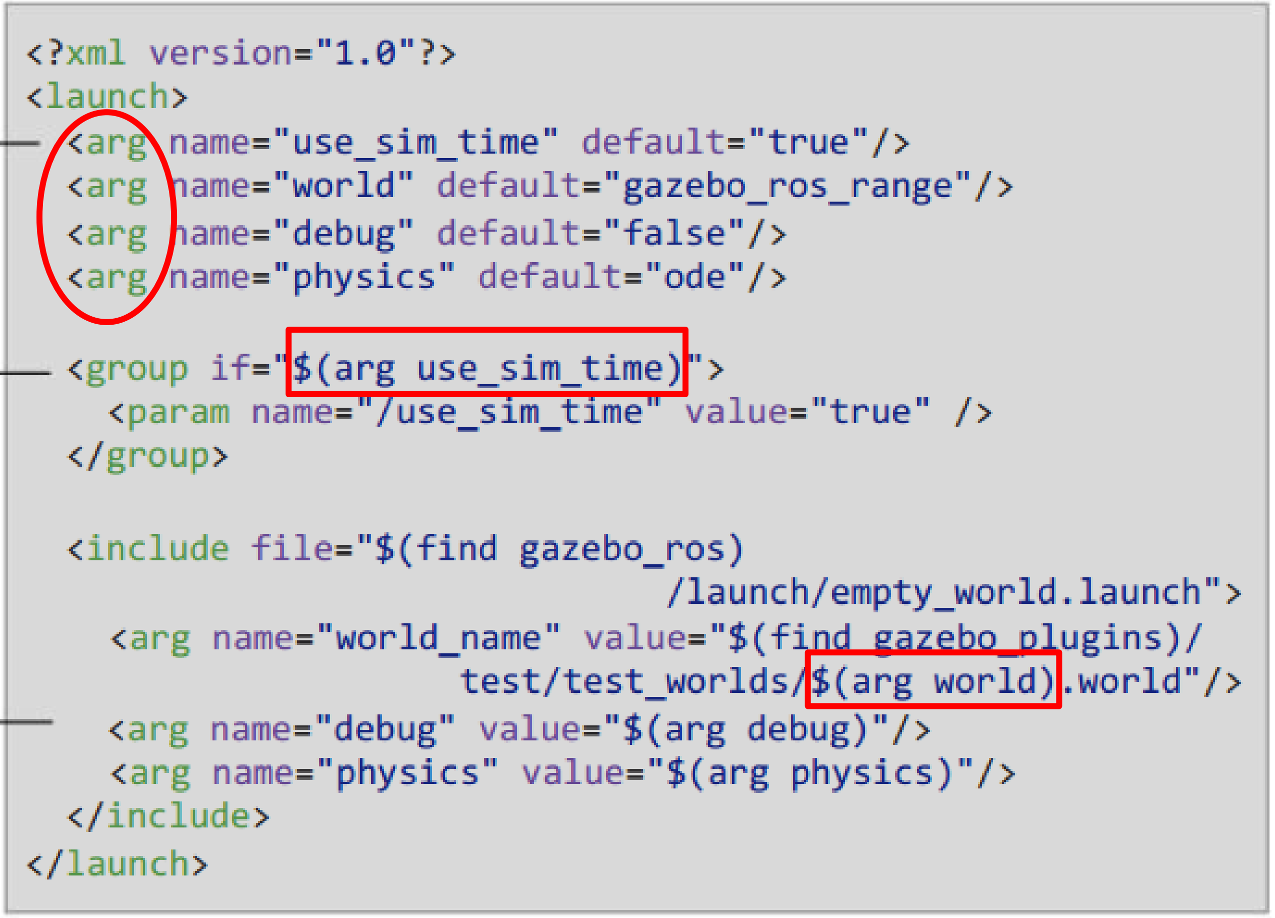
```
<arg name="arg_name" default="default_value"/>
```

- Use arguments in launch file with

```
$(arg arg_name)
```

- When launching, arguments can be set with

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



```
<?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

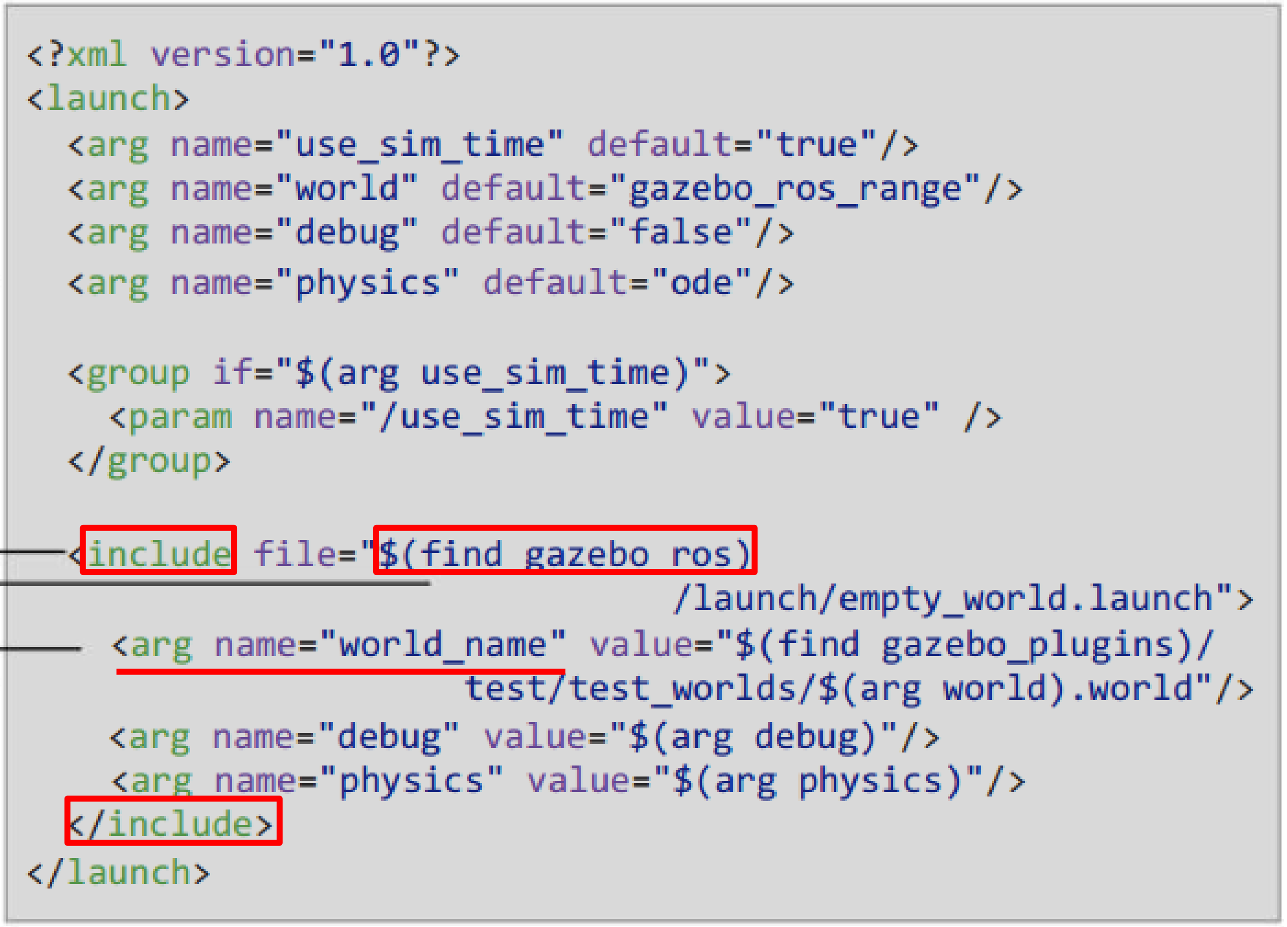
```
<include file="package_name"/>
```

- Find the system path to other packages with

```
$(find package_name)
```

- Pass arguments to the included file

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



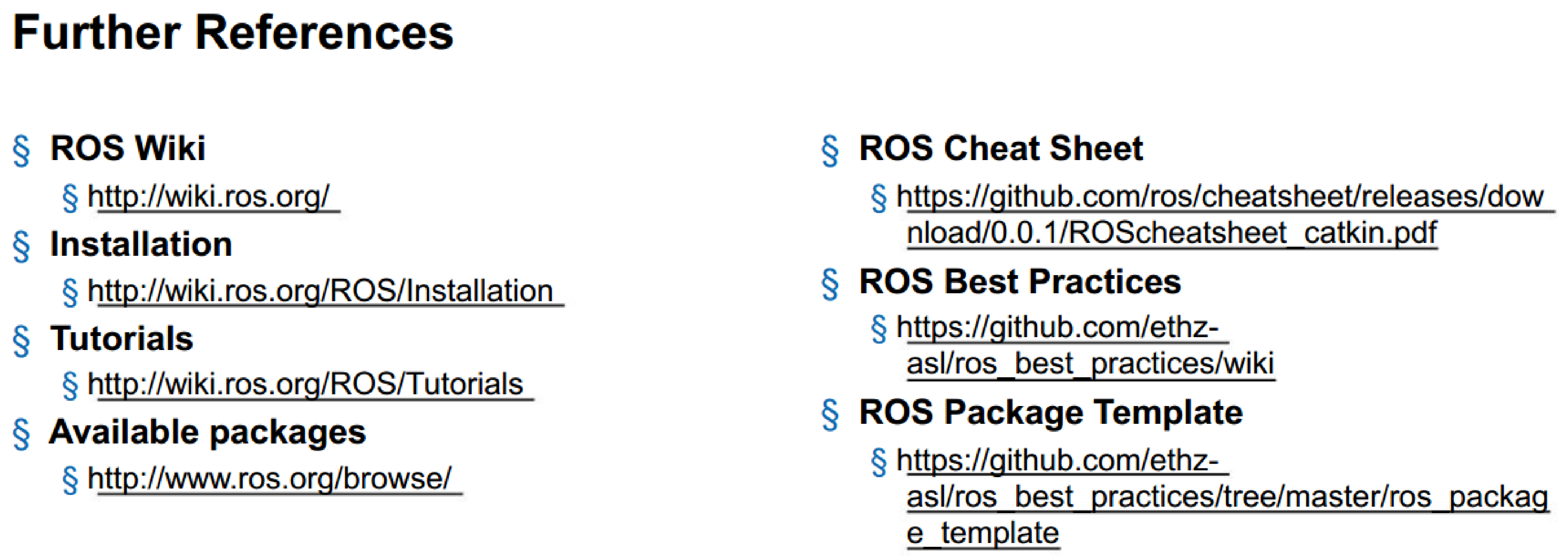
```
<?xml version="1.0"?>
<launch>
  <arg name="use_sim_time" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>

  <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>
```

Task

- Complete the Beginner Level tutorials in *http://wiki.ros.org/ROS/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://github.com/ros/cheatsheet/releases/dow
    nload/0.0.1/ROScheatsheet_catkin.pdf
- **ROS Best Practices**
  - https://github.com/ethz-
    asl/ros_best_practices/wiki
- **ROS Package Template**
  - https://github.com/ethz-
    asl/ros_best_practices/tree/master/ros_packag
    e_template