ROS Tutorials

- **Creating a ROS Workspace**
  - ❑ $ mkdir -p ~/catkin_ws/src
    $ cd ~/catkin_ws/
    $ catkin_make
    Source the setup files - $ source devel/setup.bash
  - ❑ Consists of three folders
    - ➔ src-contains our ros package
    - ➔ devel
    - ➔ build
- **Creating a ROS Package**
  - ❑ Catkin package consists of
    - ➔ The package.xml file - provides meta information about the package.
    - ➔ CMakeLists.txt
    - ➔ Each package must have its own folder.

  - ❑ $ cd ~/catkin_ws/src
    $catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
    (ex:$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp)

  - ❑ Building catkin workspace
    $ cd ~/catkin_ws
    $ catkin_make
    source the generated setup file - $. ~/catkin_ws/devel/setup.bash
  - ❑ First order dependencies
    - ➔ The dependencies are stored in package.xml
    - ➔ roscpp
      rospy
      std_msgs

  - ❑ Indirect dependencies : there are many indirect independencies, can be known through - $ rospack depends [Package Name]

- ❏ The package.xml file also needs to be customized. (Link)

- **Building packages**
  - ❏ $ source /opt/ros/kinetic/setup.bash
  - ❏ In a catkin workspace -
    $ catkin_make
  - ❏ This process is run for each CMake project.
- **Ros Nodes**
  - ❏ ROS nodes use a ROS client library to communicate with other nodes. Nodes can publish or subscribe to a Topic. Nodes can also provide or use a Service.
  - ❏ $ roscore - roscore is the first thing you should run when using ROS.
  - ❏ $ rosnode - rosnode displays information about the ROS nodes that are currently running.
    $ rosnode list - The rosnode list command lists these active nodes.
    $ rosnode info - The rosnode info command returns information about a specific node.
  - ❏ $ rosrun [package_name] [node_name] - rosrun allows you to use the package name to directly run a node within a package .
- **ROS Topics**
  - ❏ rqt_graph
    - ➔ rqt_graph creates a dynamic graph of what's going on in the system.
    - ➔ Installation : (replace <distro with the ros version)
      $ sudo apt-get install ros-<distro>-rqt
      $ sudo apt-get install ros-<distro>-rqt-common-plugins
    - ➔ $ rosrun rqt_graph rqt_graph
  - ❏ Rostopic - $ rostopic -h
    - ➔

```
rostopic bw    :  display bandwidth used by topic
rostopic echo :  print messages to screen
rostopic hz     :  display publishing rate of topic
rostopic list    : print information about active topics
rostopic pub   :  publish data to topic
rostopic type  :  print topic type
```

- ❑ rostopic echo
  - ➔ $ rostopic echo [topic]
  - ➔ rostopic echo shows the data published on a topic.
- ❑ rostopic list
  - ➔ rostopic list returns a list of all topics currently subscribed to and published.
  - ➔ Usage: $ rostopic list [/topic]

    ```
    Options:
      -h, --help           show this help message and exit
      -b BAGFILE, --bag=BAGFILE
                           list topics in .bag file
      -v, --verbose        list full details about each topic
      -p                   list only publishers
      -s                   list only subscribers
    ```

- ❑ rostopic pub
  - ➔ rostopic pub publishes data on to a topic currently advertised.
  - ➔ $ rostopic pub [topic] [msg_type] [args]
  - ➔
- ❑ rostopic hz
  - ➔ rostopic hz reports the rate at which data is published.
  - ➔ rostopic hz [topic]
- ❑ ROS Messages
  - ➔ Communication on topics happens by sending ROS messages between nodes.
  - ➔ rostopic type returns the message type of any topic being published.
  - ➔ $ rostopic type [topic]
  - ➔ We can look at the details of the message using rosmsg:

    Eg: $ rosmsg show turtlesim/Velocity

    ```
    Gives :
     float32 linear
     float32 angular
    ```

● ROS Services and Parameters

- ❑ Services are another way that nodes can communicate with each other. Services allow nodes to send a request and receive a response.
- ❑ rosservice can easily attach to ROS's client/service framework with services.
    - ➔

rosservice list	print information about active services
rosservice call	call the service with the provided args
rosservice type	print service type
rosservice find	find services by service type
rosservice uri	print service ROSRPC uri

- ❑ rosservice list
    - ➔ $ rosservice list
    - ➔ Lists all the services about the running node
- ❑ rosservice type
    - ➔ $ rosservice type [service]
    - ➔ Tells about the type of the service , the arguments it takes etc.
- ❑ rosservice call
    - ➔ $ rosservice call [service] [args]
    - ➔ Calls and makes the service function.
- ❑ rosparam
    - ➔ rosparam allows you to store and manipulate data on the ROS Parameter server . The Parameter Server can store integers, floats, boolean, dictionaries, and lists.
- ❑ rosparam set and get
    - ➔ rosparam set [param_name]
    - ➔ rosparam get [param_name]
- ❑ Rosparam dump and load
    - ➔ rosparam dump [file_name] [namespace]
    - ➔ rosparam load [file_name] [namespace]

- **ROS msg and srv**
    - ❏ msg files are simple text files that describe the fields of a ROS message. They are used to generate source code for messages in different languages.
    - ❏ an srv file describes a service. It is composed of two parts: a request and a response.
    - ❏ msg files are stored in the msg directory of a package, and srv files are stored in the srv directory.