ROS Commands Cheat sheet

Core Commands

```
$ source /opt/ros/your_distro/setup.bash #sourcing the environment setup file (if not in the .bashrc)
$ printenv | grep ROS #Checking the environment variables
$ echo $ROS_PACKAGE_PATH #Checking the package ROS package paths
$ roscore #starting the core components for nodes to communicate

$ rospack find [package_name] #finding path to a certain package
$ rospack depends1 [package_name] #viewing the first order dependencies
$ rospack depends [package_name] #All dependencies and their nested dependencies

$ roscd [package_name] #change directory directly to a package
$ roscd [package_name/directory] #move directly to a subdirectory
$ rosls [package_name] #ls directly in a package directly by name
$ roscp [package_name] [file_file_to_copy_path] [copy_path] #copying files from one package to another
$ rosed [package_name] [file_name] #directly editing a file within a package by using
# the file name rather than the having to write the entire package to the file

$ rosnode #active running nodes (IN A NEW TERMINAL AFTER running roscore)
$ rosnode list #list the active nodes
$ rosnode info /node_name #more information about a node
$ rosnode ping [node_name] #check connectivity and status of a specific node or all nodes.

$ rosrun [package_name] [node_name] #running a node directly in a package without having to know the package path
$ rosrun rqt_graph rqt_graph #Viewing the current ROS network structure
% rosrun rqt_plot rqt_plot #sources the rqt_plot GUI to display a scrolling time plot of data published on topics

$ rostopic echo [topic_name] #showing the data published on a topic
$ rostopic list -v #list of all topics currently subscribed to and published
$ rostopic gz [topic_name] #returns the rate at which data is published in a topic
$ rostopic type [topic_name] #returns the message type of any topic
$ rostopic pub [topic_name] [msg_type] [args] #publishing data on to a currently active topic (arguments must be YAML syntax)
$ rostopic pub [topic_name] [msg_type] -r [freq(hz)] [args] #publish on a topic with a specific frequency

$ rosmsg show [message_type] #returns details of the message type

$ rosservice list #print information  about active services
$ rosservice call [service_name] [args] #call the service with the provided args
$ rosservice type [service_name] #print the service type
$ rosservice find #find services by service type
```

```
$ rosservice uri #print service ROSRPC uri

$ rosparam set [param_name] #set parameter value
$ rosparam get [param_name] #get parameter value
$ rosparam load [file_name] [namespace] #load parameters from file
$ rosparam dump [file_name] [namespace] #dump parameters to file
$ rosparam delete #delete parameter
$ rosparam list #list parameter names

$ rosrun rqt_console rqt_console #
$ rosrun rqt_logger_level rqt_logger_level #

$ roslaunch [package] [filename.launch] #starts multiple nodes as defined in a
launch file
$ [ros_tool] -h #ros tools help

$ chmod +x [node_name] #to make a python node executable
```

ROS Workspace

```
#Creating a catkin workspace (running for first time creates a workspace level
CMakeLists.txt)
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make

#Prepending the workspace to ROS_PACKAGE_PATH (after using catkin_make)
$ source devel/setup.bash
```

ROS Package

```
#Creating a catkin Package (Must be done inside the src/ directory).
#This will create a package_name folder which contains a package.xml
#and a CMakeLists.txt which have been partially filled out with the information
you gave to catkin_create_package.
$ catkin_create_pkg [package_name] [dependecy_1] [dependency n]

#building a package in the catkin workspace (always to be done after creating a
new package)
$ ~/catkin_ws
$ catkin_make
$ . ~/catkin_ws/devel/setup.bash #to add the new workspace to the ROS environment
```

Bag Files

```
$ rosbag record -a #to record all the topics in a bag file (must be sourced in
/bagfiles directory)
$ rosbag record -O subset /turtle1/cmd_vel /turtle1/pose #creating a subset.bag
file and recording only the 2 shown topics
$ rosbag info [bag_file] #Checking contents of a bag file without playing it back
$ rosbag play [bag_file] #replaying the bag file. In the default mode rosbag play
will wait for 0.2s after advertising each message before
#it actually begins publishing the contents of the bag file.
#This helps to make sure the subscribers receive all the messages.
$ rosbag play -r 2 [bag_file] #custom rate of command publishing (here twice as
fast)
```