ROS Cheat Sheet

Filesystem Command-line Tools

rospack/rosstack	A tool inspecting packages/stacks.
roscd	Changes directories to a package or stack.
rospd	pushd equivalent of roscd.
rosd	Lists directories in the directory-stack.
rosls	Lists package or stack information.
rosdep	Installs ROS package system dependencies.
roswtf	Displays a errors and warnings about a running ROS system or launch file.
catkin_create_pkg	Creates a new ROS stack.
wstool	Manage several SCM repositories.
catkin_make	Builds a ROS package.
rqt_dep	Displays package structure and dependencies.

Usage:

```
$ rospack find [package]
$ roscd [package[/subdir]]
$ rospd [package[/subdir] | +N | -N]
$ rosd
$ rosls [package[/subdir]]
$ rosed [package] [file]
$ roscp [package] [file] [destination]
$ rosdep install [package]
$ roswtf or roswtf [file]
$ catkin_create_pkg [package_name] [depend1] [depend2]
$ wstool [init | set | update]
$ catkin_make
$ rqt_dep [options]
```

Common Command-line Tools

roscore

A collection of nodes and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate.

Usage:

```
$ roscore
```

rosnode

Displays debugging information about ROS nodes, including publications, subscriptions and connections.

Commands:

rosnode ping	Test connectivity to node.
rosnode list	List active nodes.
rosnode info	Print information about a node.
rosnode machine	List nodes running on a particular machine.
rosnode kill	Kills a running node.

Examples:

Kill all nodes:
```
$ rosnode kill -a
```
List nodes on a machine:
```
$ rosnode machine aqy.local
```
Ping all nodes:
```
$ rosnode ping --all
```

rosmsg/rossrv

rosmsg/rossrv displays Message/Service (msg/srv) data structure definitions.

Commands:

rosmsg show	Display the fields in the msg.
rosmsg users	Search for code using the msg.
rosmsg md5	Display the msg md5 sum.
rosmsg package	List all the messages in a package.
rosnode packages	List all the packages with messages.

Examples:

Display the Pose msg:
```
$ rosmsg show Pose
```
List the messages in nav_msgs:
```
$ rosmsg package nav_msgs
```
List the files using sensor_msgs/CameraInfo:
```
$ rosmsg users sensor_msgs/CameraInfo
```

rosrun

rosrun allows you to run an executable in an arbitrary package without having to cd (or roscd) there first.

Usage:
```
$ rosrun package executable
```
Example:

Run turtlesim:
```
$ rosrun turtlesim turtlesim_node
```

roslaunch

Starts ROS nodes locally and remotely via SSH, as well as setting parameters on the parameter server.

Examples:

Launch on a different port:
```
$ roslaunch -p 1234 package filename.launch
```
Launch a file in a package:
```
$ roslaunch package filename.launch
```
Launch on the local nodes:
```
$ roslaunch --local package filename.launch
```

rostopic

A tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Commands:

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 find	Find topics by type.

Examples:

Publish hello at 10 Hz:
```
$ rostopic pub -r 10 /topic_name std_msgs/String hello
```
Clear the screen after each message is published:
```
$ rostopic echo -c /topic_name
```
Display messages that match a given Python expression:
```
$ rostopic echo --filter "m.data=='foo'" /topic_name
```
Pipe the output of rostopic to rosmsg to view the msg type:
```
$ rostopic type /topic_name | rosmsg show
```

rosservice

A tool for listing and querying ROS services.

Commands:

rosservice list	Print information about active services.
rosservice node	Print the name of the node providing a service.
rosservice call	Call the service with the given args.
rosservice args	List the arguments of a service.
rosservice type	Print the service type.
rosservice uri	Print the service ROSRPC uri.
rosservice find	Find services by service type.

Examples:

Call a service from the command-line:
```
$ rosservice call /add_two_ints 1 2
```
Pipe the output of rosservice to rossrv to view the srv type:
```
$ rosservice type add_two_ints | rossrv show
```
Display all services of a particular type:
```
$ rosservice find rospy_tutorials/AddTwoInts
```

rosparam

A tool for getting and setting ROS parameters on the parameter server using YAML-encoded files.

Commands:

rosparam set	Set a parameter.
rosparam get	Get a parameter.
rosparam load	Load parameters from a file.
rosparam dump	Dump parameters to a file.
rosparam delete	Delete a parameter.
rosparam list	List parameter names.

Examples:

List all the parameters in a namespace:
```
$ rosparam list /namespace
```
Setting a list with one as a string, integer, and float:
```
$ rosparam set /foo "['1', 1, 1.0]"
```
Dump only the parameters in a specific namespace to file:
```
$ rosparam dump dump.yaml /namespace
```

tf Command-line Tools

tf_echo

A tool that prints the information about a particular transformation between a source_frame and a target_frame.

Usage:
```
$ rosrun tf tf_echo <source_frame> <target_frame>
```
Examples:

To echo the transform between /map and /odom:
```
$ rosrun tf tf_echo /map /odom
```

view_frames

A tool for visualizing the full tree of coordinate transforms.

Usage:
```
$ rosrun tf view_frames
$ evince frames.pdf
```

Logging Command-line Tools

rosbag

This is a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids deserialization and reserializationof the messages.

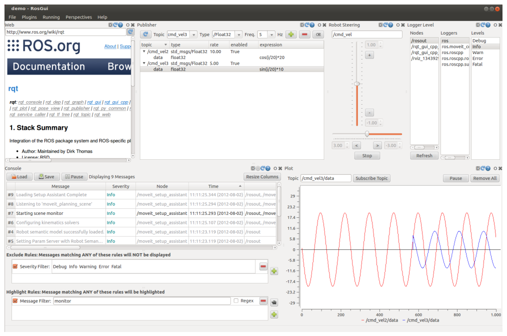
rosbag record will generate a ".bag" file with the contents of all topics that you pass to it.
Examples:
Record all topics:
```
$ rosbag record -a
```
Record select topics:
```
$ rosbag record topic1 topic2
```

rosbag play will take the contents of one or more bag file, and play them back in a time-synchronized fashion.
Examples:
Replay all messages without waiting:
```
$ rosbag play -a demo_log.bag
```
Replay several bag files at once:
```
$ rosbag play demo1.bag demo2.bag
```

Graphical Tools

rqt

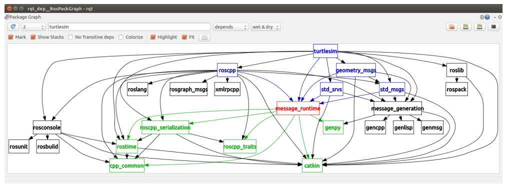
Qt-based framework for ROS that can run all the existing GUI tools as dockable windows within rqt.



Usage:
```
$ rqt (and choose Plugins from a menu)
```

rqt_dep

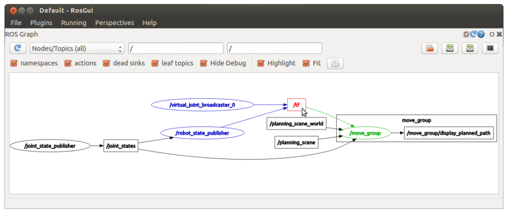
Visualize the ROS dependency graph.



Usage:
```
$ rqt_dep
```

rqt_graph

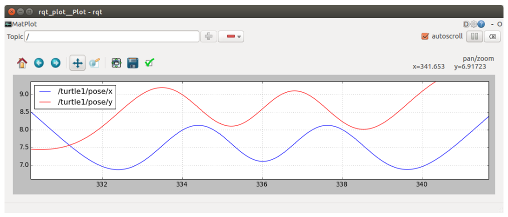
Displays a graph of the ROS nodes that are currently running, as well as the ROS topics that connect them.



Usage:
```
$ rqt_graph
```

rqt_plot

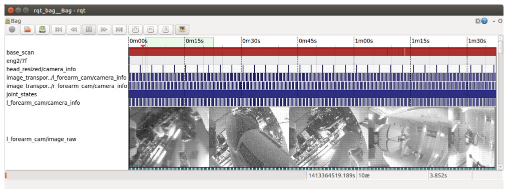
A tool for plotting data from one or more ROS topic fields using different plotting backends.



Examples:
To graph multiple data:
```
$ rqt_plot /turtle1/pose/x /turtle1/pose/y
```
To graph multiple fields of a message:
```
$ rqt_plot /turtle1/pose/x:y
```

rqt_bag

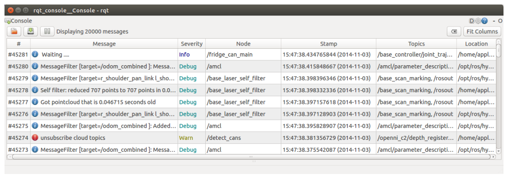
A tool for visualizing, inspecting, and replaying histories (bag files) of ROS messages.



Usage:
```
$ rqt_bag bag_file.bag
```

rqt_console

A tool for displaying and filtering messages published on rosout.



rqt_console__Console - rqt
Console
Displaying 20000 messages

#	Message	Severity	Node	Stamp	Topics	Location
#45281	Waiting	Info	/fridge_can_main	15:47:38.434765844 (2014-11-03)	/base_controller/joint_traj...	/home/appl...
#45280	MessageFilter [target=/odom_combined]: Messa...	Debug	/amcl	15:47:38.415848667 (2014-11-03)	/amcl/parameter_descripti...	/opt/ros/hy...
#45279	MessageFilter [target=r_shoulder_pan_link l_sho...	Debug	/base_laser_self_filter	15:47:38.398396346 (2014-11-03)	/base_scan_marking, /rosout	/opt/ros/hy...
#45278	Self filter: reduced 707 points to 707 points in 0.0...	Debug	/base_laser_self_filter	15:47:38.398332336 (2014-11-03)	/base_scan_marking, /rosout	/home/appl...
#45277	Got pointcloud that is 0.046715 seconds old	Debug	/base_laser_self_filter	15:47:38.397157618 (2014-11-03)	/base_scan_marking, /rosout	/home/appl...
#45276	MessageFilter [target=r_shoulder_pan_link l_sho...	Debug	/base_laser_self_filter	15:47:38.397128903 (2014-11-03)	/base_scan_marking, /rosout	/opt/ros/hy...
#45275	MessageFilter [target=/odom_combined]: Added...	Debug	/amcl	15:47:38.395828907 (2014-11-03)	/amcl/parameter_descripti...	/opt/ros/hy...
#45274	unsubscribe cloud topics	Warn	/detect_cans	15:47:38.381356729 (2014-11-03)	/openni_c2/depth_register...	/home/appl...
#45273	MessageFilter [target=/odom_combined]: Messa...	Debug	/amcl	15:47:38.375542087 (2014-11-03)	/amcl/parameter_descripti...	/opt/ros/hy...

Usage:
```
$ rqt_console
```

catkin Build Tools

catkin_create_pkg

A tool to create ROS package.
Usage:
```
$ catkin_create_pkg <package_name> [depend1] [depend2]
```
Examples:
```
$ source /opt/ros/hydro/setup.bash
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

wstool

A tool to manage several SCM repositories based on a single workspace. definition file (.rosinstall).
Examples:
```
$ cd ~/catkin_ws/src
$ wstool init
$ wstool set ros_tutorials --git
         git://github.com/ros/ros_tutorials.git
$ wstool update
```

catkin_make

A tool to build code in a catkin workspace.
Examples:
```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
```

CMakeLists.txt

Your CMakeLists.txt file MUST follow this format otherwise your packages will not build correctly.
`cmake_minimum_required()` Specify the name of the package
`project()` Project name which can refer as ${PROJECT_NAME}
`find_package()` Find other packages needed for build
`catkin_package()` Specify package build info export

Build Executables and Libraries:
Use CMake function to build executable and library targets. These macro should call after `catkin_package()` to use `catkin_*` variables.
```
include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(hoge src/hoge.cpp)
add_library(fuga src/fuga.cpp)
target_link_libraries(hoge fuga ${catkin_LIBRARIES}))
```

Message generation:
There are `add_{message,service,action}_files()` macros to handle messages,services and actions respectively. They must call before `catkin_package()`.
```
find_package(catkin COMPONENTS message_generation std_msgs)
add_message_files(FILES Message1.msg)
generate_messages(DEPENDENCIES std_msgs)
catkin_package(CATKIN_DEPENDS message_runtime)
```
If your package builds messages as well as executables that use them, you need to create an explicit dependency.
```
add_dependencies(hoge ${PROJECT_NAME}_generate_messages_cpp)
```

東京オープンソース ロボティクス協会
Tokyo Opensource Robotics Kyokai Association

ROS