# Week 1 Lab – Getting Started with ROS

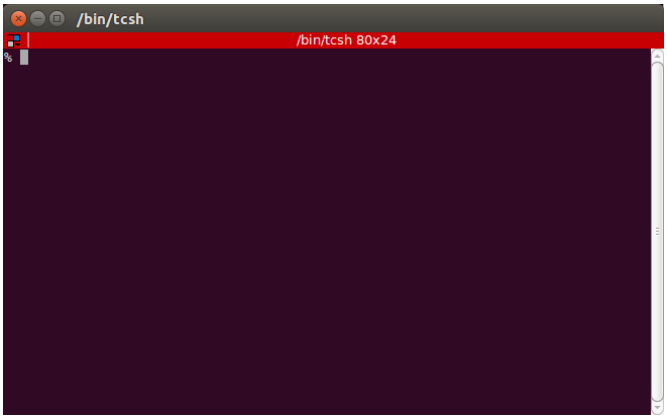
This first lab is designed to help you get started with ROS. This lab will help you set-up your ROS environment, teach you about ROS command-line tools, GUIs and other basic interfaces. There will be virtually no code, but instead you will have a survey of the tools available to you.

---

## Exercise 1 – Setting Up ROS on Docker

There are several things that are required in a ROS environment. These include a supported shell (we will use bash), sourcing ROS and creating a workspace. We will also set up a Docker environment to run ROS in:

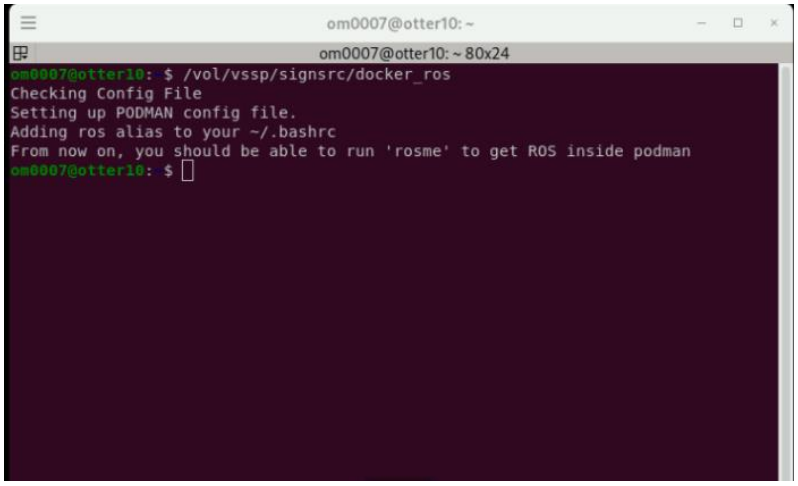
1. **Open a terminal:** Open the Dash by pressing the windows key on the keyboard, type "*terminator*", and select the Terminator application from the results that appear. You should see a window like this one:




2. **Set Up Docker**: Type the command (**note: ignore the $**)

```
$ /vol/vssp/signsrc/docker_ros
```

and press enter. The terminal should now look like this:

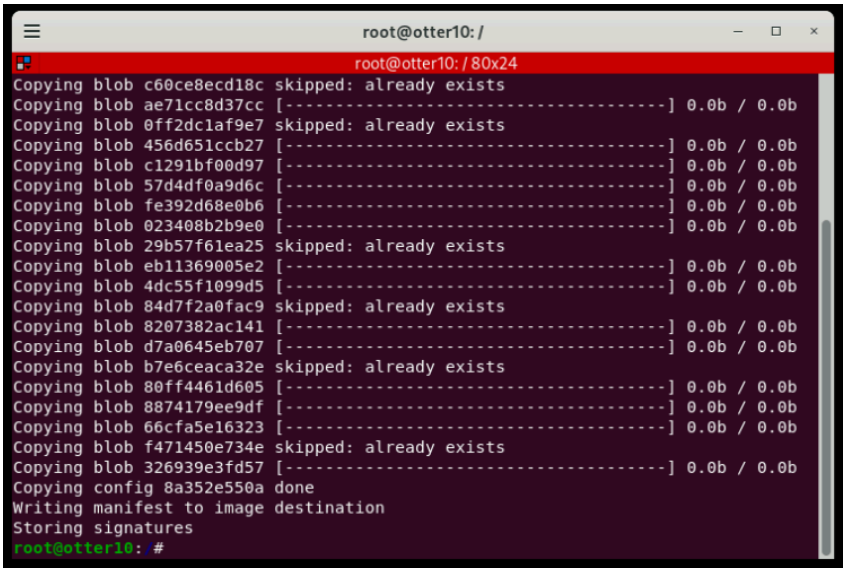



**NOTE: CLOSE AND RESTART YOUR TERMINATOR OR RE-SOURCE YOUR ~/.BASHRC!**

3. **Start Docker:** In a new Terminal, you can use the command

```
$ rosme
```

to start a docker image with ROS running inside. You should do this at the beginning of every lab and ensure that your terminal looks like this:



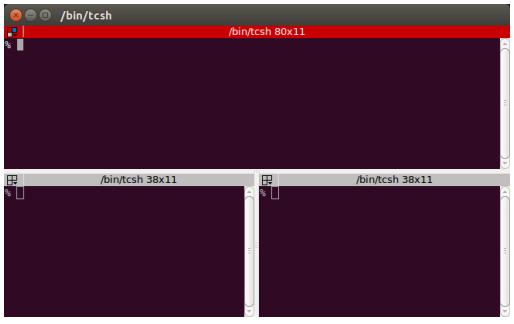

**NOTE: the user has changed to "root" and you are now operating inside the docker container.**

4. **New Terminator:** To simplify things, you may want to run terminator *inside* docker:

```
$ terminator
```

which will allow you to open new tabs and split the windows while remaining inside your docker container. From now on, we will assume ALL commands are run inside docker. Terminator is a type of terminal that allows you to split your window into spaces, and have multiple tabs. ROS requires multiple terminals to use effectively, so we will use this tool to avoid having dozens of terminal windows open.

**Experiment with right clicking to open new tabs and split the terminal horizontally and vertically and See if you can recreate the layout below**




once you are comfortable with Terminator, move on to the next step.

5. **Source ROS**: Ensure you are inside your Docker environment (root@otterXX). Run the command

```
$ source /opt/ros/melodic/setup.bash
```

This command will give you access to all the ROS command-line tools, pre-installed packages and GUIs. However, before we continue, we should make this happen automatically.

6. **Permanently source ROS from .bashrc:** Run the command

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

This will append the source command to your .**bashrc** file. This file gets run every time bash is started, which means ROS is now sourced automatically. **Note:** If you have anaconda or any similar system set up, it may clash with ROS. You should comment it out for these labs.

7. **Explore the .bashrc file**: Run the command

```
$ gedit ~/.bashrc &
```

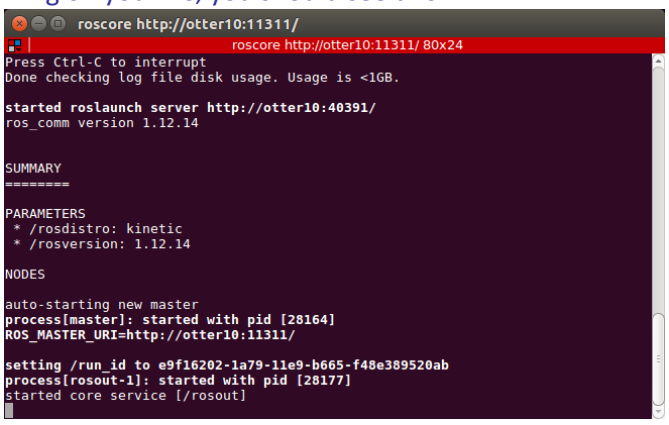
and familiarise yourself with this file. The command you added should be at the bottom. ***You should never edit anything above this line.***

Eventually, you will need to source your own workspaces in the same way we sourced ROS.

8. **Run the ROS Master:** Start a new terminal, run bash and type the command

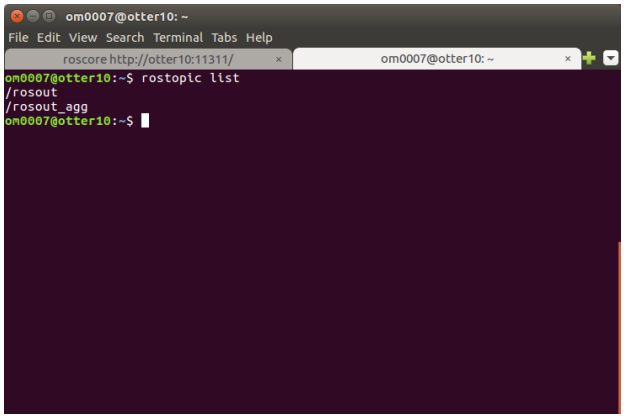
```
$ roscore
```

ROS is now running on your PC, you should see this:




and you should be able to continue with the exercises on this lab!

9. **Test ROS:** Open new tab by pressing *Ctrl+Shift+T* while in Terminator, alternatively *Ctrl+Shift+O* can be used to split the screen horizontally, and *Ctrl+Shift+E* to split vertically. Enter the command




```
$ rostopic list
```

which will show you two low-level ROS topics,
If this command ran successfully, then ROS is ready to run on your user.
When you are done, close the new tab/space, but keep your roscore running.

**Try some of the commands you saw in lectures:**
**rospack, rosls, roscd**

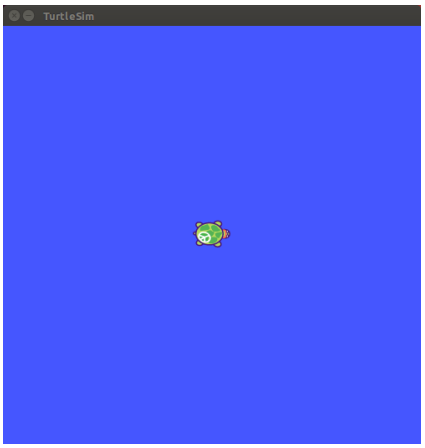
## Exercise 2 – Trying out ROS

We will now work our way through the Turtlesim example from lectures. This will allow us to explore some of the basic concepts that ROS is built upon, as well as give you an opportunity to get used to the command-line tools.

1. **Start Turtlesim Node:** Open a new terminator window, make sure ROS is sourced, and run the command

   ```
   $ rosrun turtlesim turtlesim_node
   ```

   you should now see a screen like the one in the lectures

   in a new tab, run the command

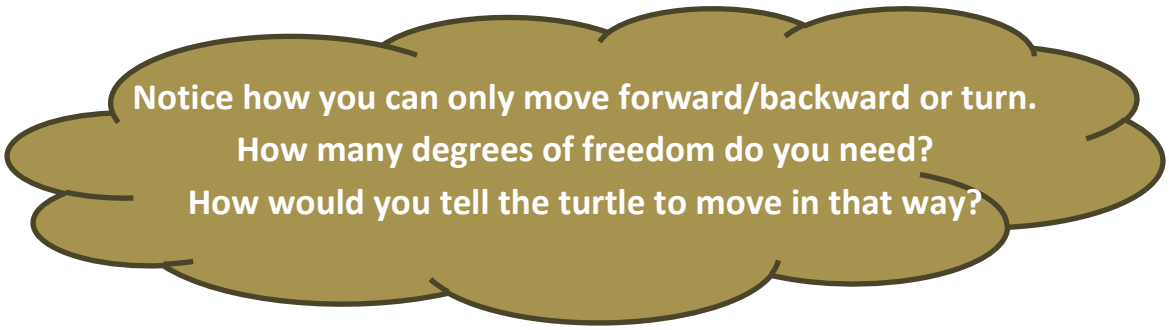
   ```
   $ rosnode list
   ```

   to ensure that the node ***/turtlesim*** is running.

2. **Start Keyboard Teleop Node:** Open a new Tab, or split terminator into a new Space, and run the command

   ```
   $ rosrun turtlesim turtle_teleop_key
   ```

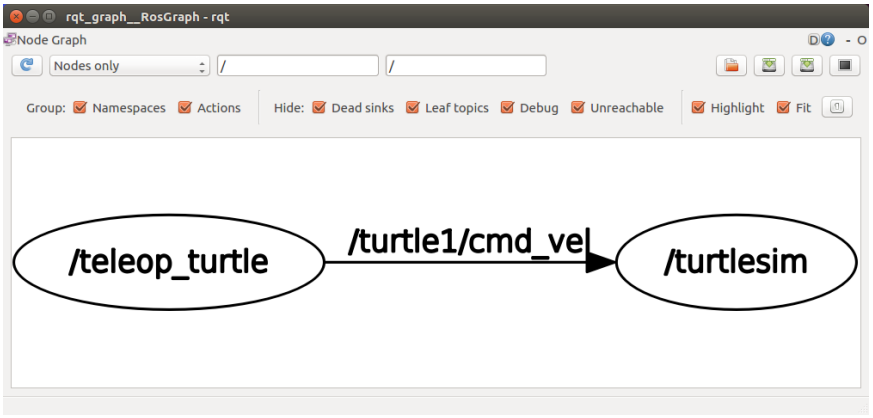
   which will let you control the turtle in the simulator using your keyboard keys. Note that the terminal widow must be selected (not the simulator!). Try moving the turtle around to get a feel for how it works.



3. **Inspect the Node Graph:** Open a new Terminal/Tab, and run the command

```
$ rosrun rqt_graph rqt_graph
```

this should show you the following window,




with nodes currently running on your system as ovals and the topics they use to communicate as arrows. Inspect this graph, make sure you understand what everything means.

4. **Inspect Published Topics:** Run the command

```
$ rostopic list
```

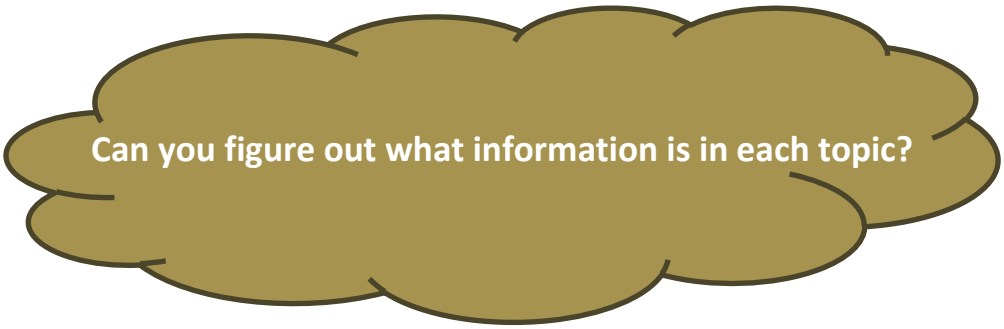
you should be able to see three new topics, under the namespace */turtle1*. Can you guess what each does? Try using

```
$ rostopic list -v
```

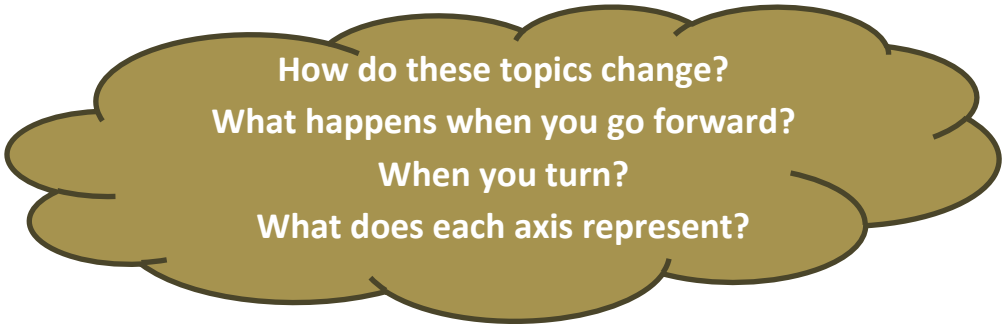
to figure out what message type each topic has! Now that you know the type of message being published, use the command

```
$ rostopic echo <topic_name>
```

to figure show what each of the topics is outputting. **NOTE: <topic_name> is a placeholder for any of the 3 topics.**




Use the keyboard to control the turtlebot while you listen in on the “cmd_vel” and “pose” topics.




Make sure you understand what the **difference** between these topics is, which one is used to control the turtle and what the messages are encoding. Once you have these answers move on to the next task. **HINT**: Use the commands

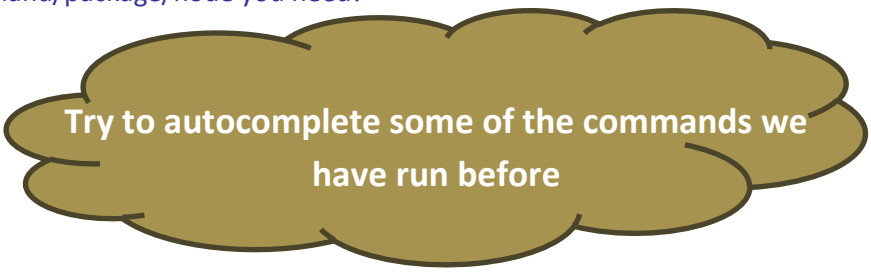
```
$ rostopic type <topic_name>
```

and

```
$ rosmsg show <message_type>
```

to get the message types and their format.

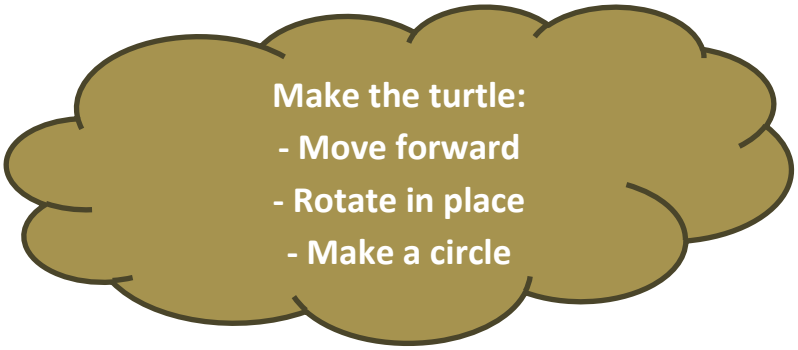
5. **Autocomplete:** While it is possible to remember the syntax required to use a command, the ROS autocomplete function is very well developed and extremely useful. To use it, press the **<TAB>** key while typing a command. Note that if there are multiple choices, double press <TAB> and ROS will give you options, this is especially useful when you don't know the command/package/node you need!




6. **Moving the Turtle:** Using autocomplete, try the command

```
$ rostopic pub /turtle1/cmd_vel <TAB> <TAB>
```

notice how ROS autocompletes the message type and the message contents/syntax. Did the turtle move? Why not? Modify the command to make the turtle do different motions.
**HINT: use the "-r 1" flag!**



Can you also trace a number “8” shape? Why/why not? What would you need to do?

Keep the circle command running, we will use it in the next steps.

7. **Visualising Data:** Try using

```
$ rostopic echo /turtle1/pose
```

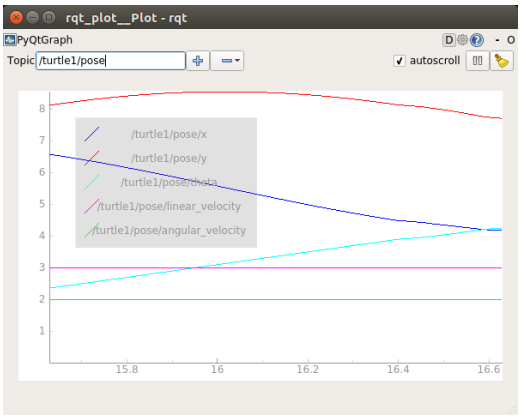
to show the pose of the robot. Does it make sense? It might be easier to visualise this in a graphical interface, so lets run

```
$ rosrun rqt_plot rqt_plot
```

you should see a GUI like this (you made need to drag the corner to expand it)

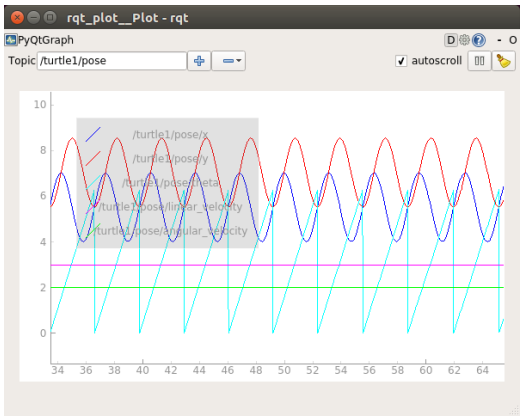

add the pose topic (/turtle1/pose) to the text box on the top left and visualise the data, it should look like this

this view presents you with a plot that represents the scalar value of each element in the pose message (position in each axis and linear/angular velocities). However, it is difficult to




see at the current zoom level. Try clicking the 4-way arrow tool then **Right Click + move the mouse Up/Down or left/right to change vertical and horizontal Scale**, or **Left Click + Drag**

**to Pan**. Get the plot to look like this:




**Try moving the turtle in different ways.**
**Can you figure out what the plots will do?**

**NOTE: rqt_plot can be used to visualise other topics as well (as long as they are made up of standard ros messages). You can see more detail here: [http://wiki.ros.org/rqt_plot](http://wiki.ros.org/rqt_plot)**

8. **Continue Exploring Command-line Tools:** We've only explored a small subsection of command-line tools. There are a lot more, listed here: [http://wiki.ros.org/ROS/CommandLineTools](http://wiki.ros.org/ROS/CommandLineTools)

   See of you can:

   - Use **rosed** to open a file
     i. Add "export EDITOR=gedit" to your .bashrc (no quotes)
   - cd into to the **turtlesim** package using **roscd**
   - **List** all nodes using **rosnode**
     i. Try **ping**ing and getting **info** from them
   - **List** all services using **rosservice**
     i. Try **clear**ing and **reset**ting the turtle (remember to autocomplete!) by **call**ing these services
   - **List** all parameters using **rosparam**
   - Locate the turtlesim git repo using **roslocate**

- **Record** and **Play** Poses/Commands using **rosbag**

**We will explore these functions in more detail later, but try getting them to work now!**

**NOTE: The first two labs are a condensed version of the official ROS tutorials (http://wiki.ros.org/ROS/Tutorials).**

**They have been modified to fit the course material and time constraints.**

**Going through the official ROS tutorials is highly recommended!**