ROS2_Lec2

(*Robot Operating System*)

Ros2 Index

https://index.ros.org/ (For Self Study)

ROS Resources: Documentation | Support | Discussion Forum | Service Status | Q&A answers.ros.org

ROS Index BETA

ABOUT INDEX DOC CONTRIBUTE STATS Search ROS

PACKAGE LIST REPOSITORY LIST

Search ROS

Welcome to ROS Index

ROS Index is the entry point for searching ROS and ROS 2 resources, including packages, repositories, system dependencies and documentation.

You can enter keywords and phrases in the search bar and then filter results by resource type, or you can browse the complete package, repository and system dependency lists under the **Index** tab.

Under the **Doc** tab, you'll find the official ROS 2 documentation, including installation instructions, tutorials, distribution features and summaries, contributing guides, and more.

To go directly to the installation pages, click here.
To go directly to the tutorials, click here.

Active Distributions

ROS Melodic

ROS Noetic

ROS 2 Dashing

ROS 2 Foxy

Development Distribution

ROS 2 Rolling

More resources

ROS Discourse

ROS Answers

Source the setup files

- You will need to run this command on every new shell you open to have access to the ROS 2 commands
- Command to source the setup files : `source /opt/ros/<distro>/setup.bash`
- If you don't want to have to source the setup file every time you open a new shell, you can add the command to your shell startup script: `echo "source /opt/ros/<distro>/setup.bash" >> ~/.bashrc`

ROS2 packages

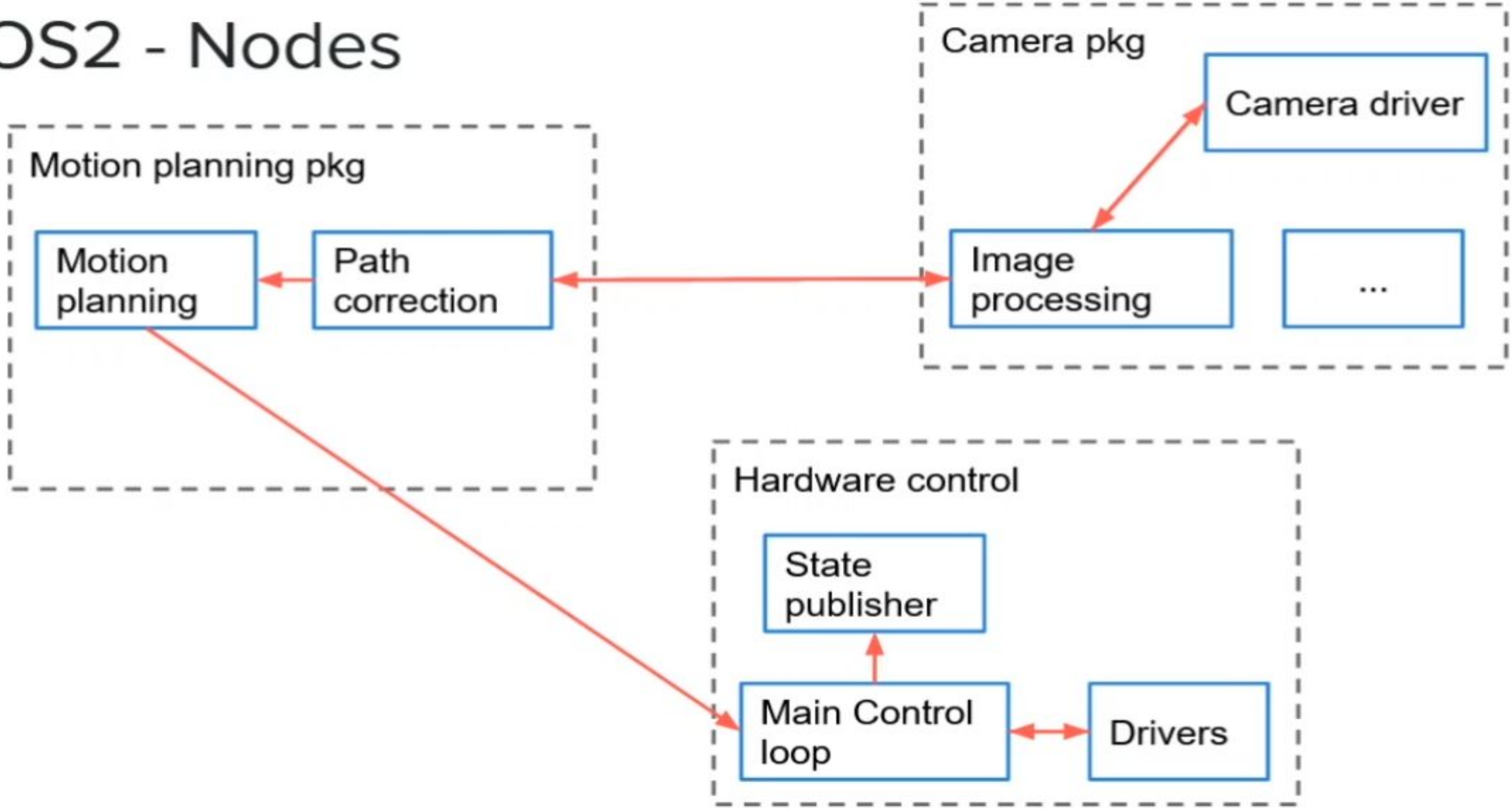
ROS2 - Packages

Motion planning pkg

Camera pkg

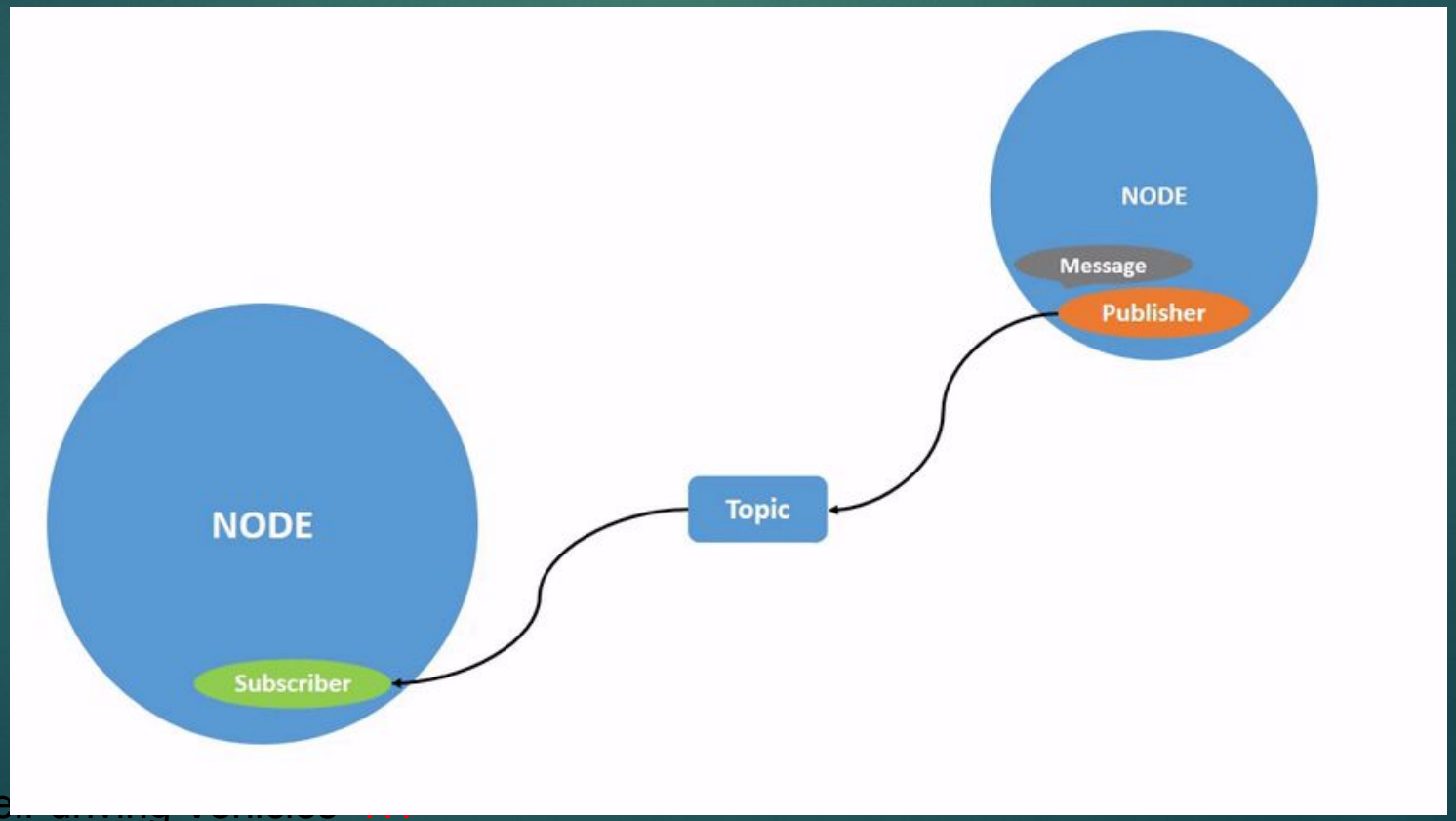
Hardware control

ROS2 - Nodes

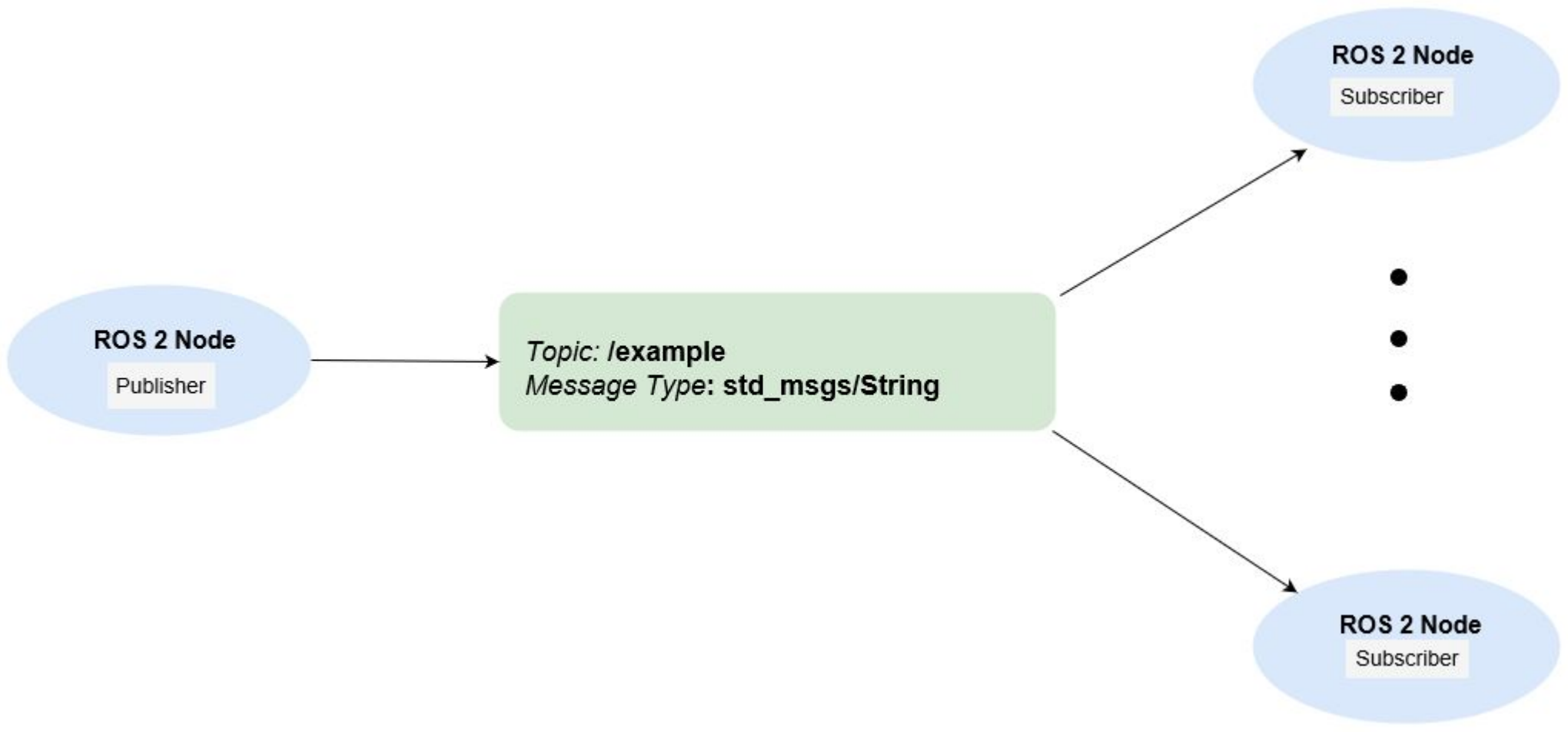

Camera pkg
Camera driver
Image processing
...
Motion planning pkg
Motion planning
Path correction
Hardware control
State publisher
Main Control loop
Drivers

Understanding ROS 2 topics

- Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages.



Publisher&Subscriber



build Tool

- colcon is an iteration on the ROS build tools catkin_make, catkin_make_isolated, catkin_tools and ament_tools.
- to install colcon :-

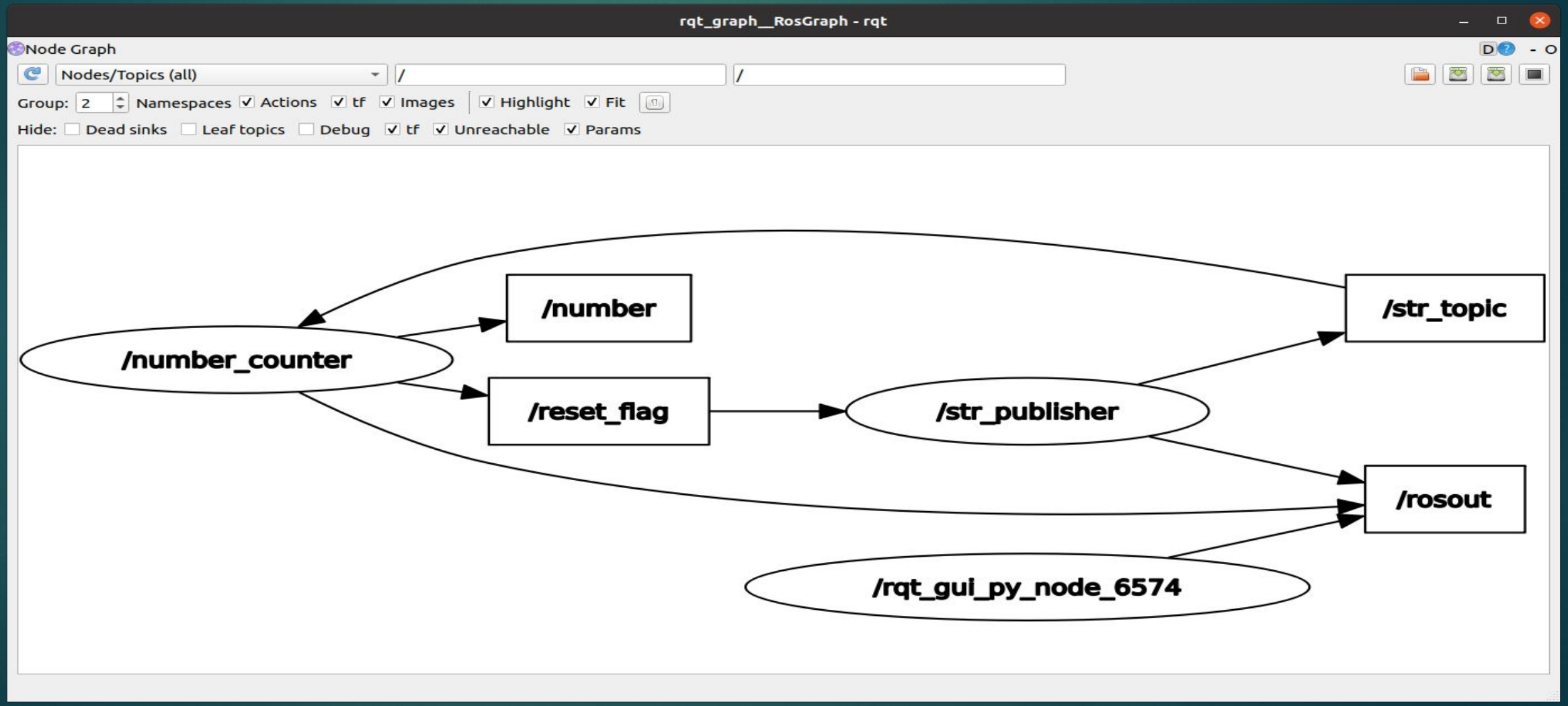
```
sudo apt install python3-colcon-common-extensions
```

- colcon autocomplete

  add the following command to bashrc

```
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash
```

Make ros2 python package <ITI_LAB1> include 2 node (node1,node2) as shown ,node1 publish str msg (“ <your name> is publish <,><counter>”) counter is start with 0 ,node2 receive str msg , publish the counter and reset flag to reset counter in publisher node (max value for counter is 5) show your rqt_graph as shown save it as png image.



How to use Colcon

to build Workspace use the Following Command

```
colcon build
```

you can ma symlink to your pkg use the following command

```
colcon build --packages-select <pkg_name> --symlink-install
```

Ros2 Node Tools (debug)

ros2 run -h

ros2 node (list,info)

ros2 node -h

Don't make 2 node with same name

source .bashrc

Remap : ros2 run pkg exec --ros-args --remap __node:=new_name

Colcon (pkg-select,symlink)

Topic Tools

```
ros2 topic list
ros2 topic echo   (subscriber in terminal)
ros2 topic info
ros2 topic hz
ros2 topic pub
ros2 topic pub /topic example_interfaces/msg/String "{data: "msg"}"
ros2 topic pub -r 5 /my_topic example_interfaces/msg/String "{data: "msg"}"
```

Remap Topic

```
ros2 run my_1 pub --ros-args --remap __node:=new_one --remap my_topic:=new_topic
```

How to Run Ros2 Node

- Ros2 Run

```
ros2 run <package_name> <executable_name>
```

- To run turtlesim, open a new terminal, and enter the following command:

```
ros2 run turtlesim turtlesim_node
```

- The turtlesim window will open

- Here, the package name is turtlesim and the executable name is turtlesim_node

How to know running Node names

- ros2 node list

```
ros2 node list
```

- The terminal will return the node name: /turtlesim
- Open another new terminal and start the teleop node with the command

```
ros2 run turtlesim turtle_teleop_key
```

- Here, we are searching the turtlesim package again, this time for the executable named turtle_teleop_key.
- Return to the terminal where you ran ros2 node list and run it again. You will now see the names of two active nodes:/turtlesim, /teleop_turtle

After running previous command

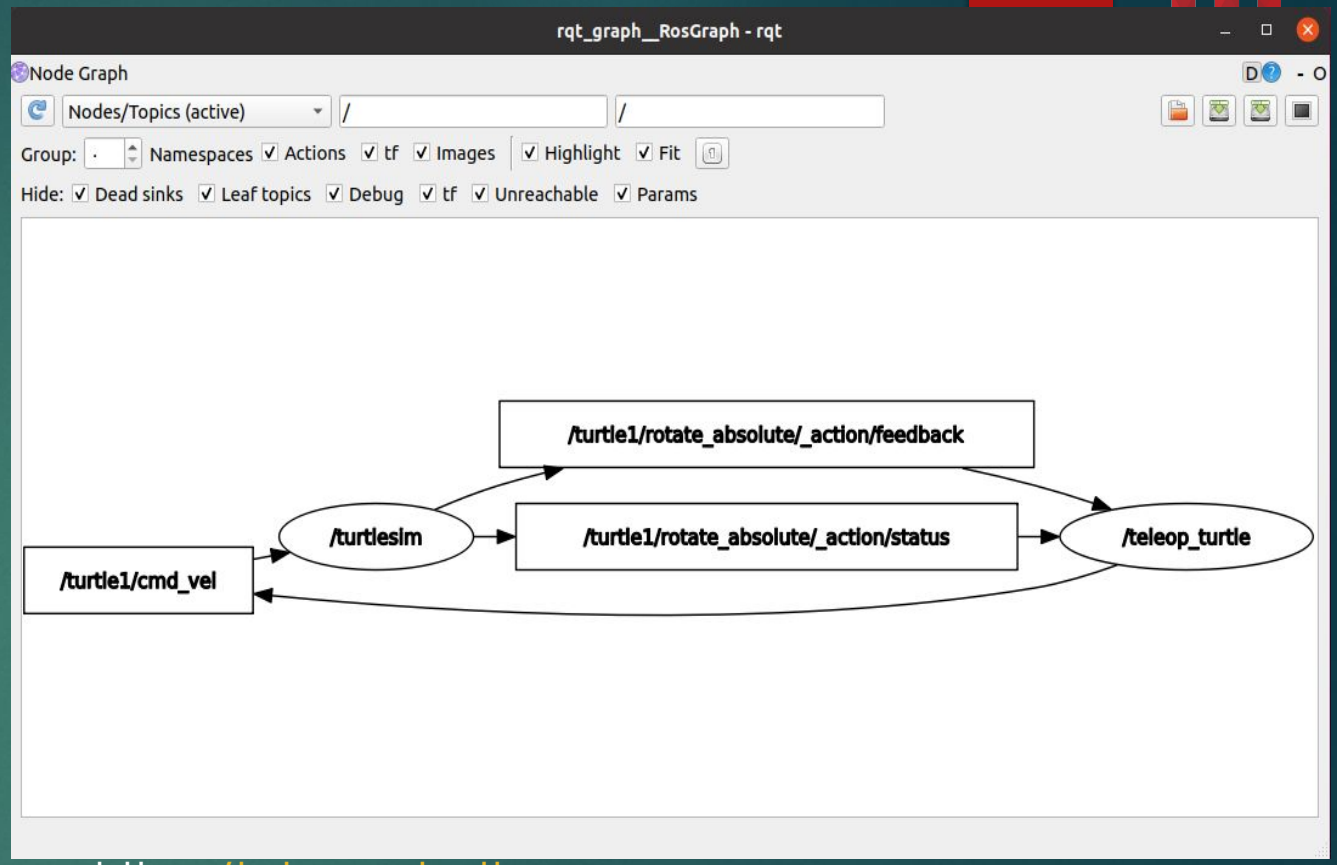
```
ahmed@ahmed-VirtualBox:~$ ros2 run  turtlesim turtlesim_node
[INFO] [1606933202.324425739] [turtlesim]: Starting turtlesim with node name /tu
rtlesim
[INFO] [1606933202.348482946] [turtlesim]: Spawning turtle [turtle1] at x=[5.544
445], y=[5.544445], theta=[0.000000]
```

```
ahmed@ahmed-VirtualBox:~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rot
ation.
'Q' to quit.
```

- Type ros2 node list ,terminal will return the following:

```
ahmed@ahmed-VirtualBox:~$ ros2 node list
/teleop_turtle
/turtlesim
ahmed@ahmed-VirtualBox:~$
```

Rqt_graph



- Open a new terminal and run:

```
ros2 run turtlesim turtlesim_node
```

- Open another terminal and run:

```
ros2 run turtlesim turtle_teleop_key
```

- Now run rqt_grapgh in another terminal

```
rqt_graph
```

- You should see the above nodes and topic.
- The graph is depicting how the /turtlesim node and the /teleop_turtle node are communicating with each other over a topic. The /teleop_turtle node is publishing data (the keystrokes you enter to move the turtle around) to the /turtle1/cmd_vel topic, and the /turtlesim node is subscribed to that topic to receive the data.

How to show running topics

- Running the ros2 topic list command in a new terminal will return a list of all the topics currently active in the system:

```
ros2 topic list
```

```
ahmed@ahmed-VirtualBox: ~
ahmed@ahmed-VirtualBox:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
ahmed@ahmed-VirtualBox:~$
```

- ros2 topic list -t will return the same list of topics, this time with the topic type appended in brackets after each:

```
ahmed@ahmed-VirtualBox: ~
ahmed@ahmed-VirtualBox:~$ ros2 topic list -t
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
ahmed@ahmed-VirtualBox:~$
```

How To see the data being published on a topic

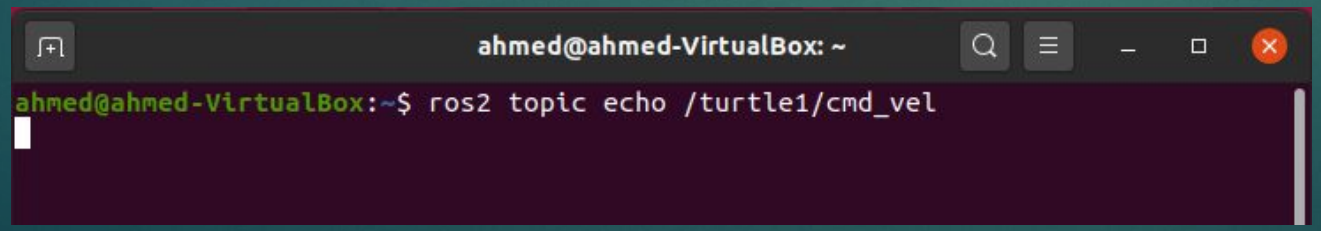
- To see the data being published on a topic, use:

```
ros2 topic echo <topic_name>
```

- Since we know that /teleop_turtle publishes data to /turtlesim over the /turtle1/cmd_vel topic, let's use echo to introspect on that topic:

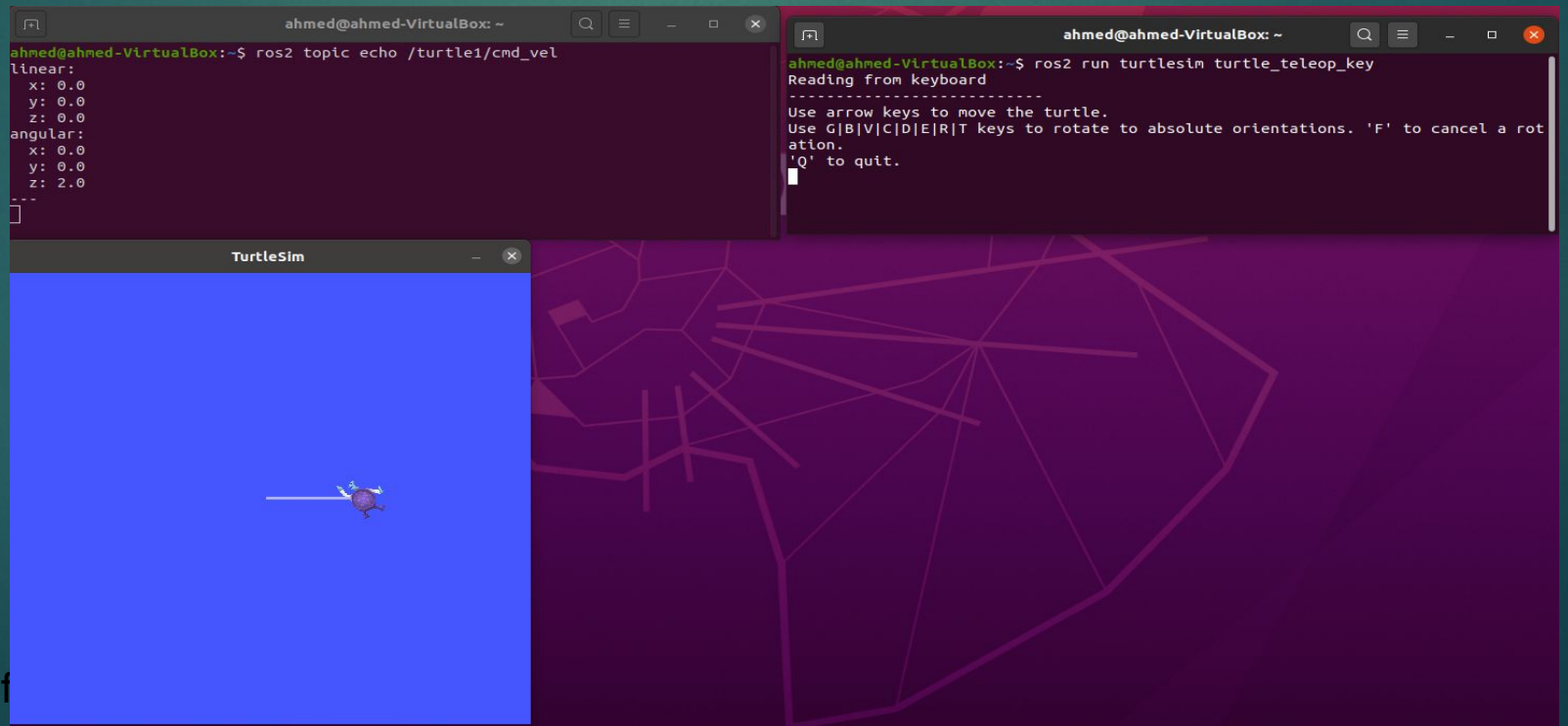
```
ros2 topic echo /turtle1/cmd_vel
```

- At first, this command won't return any data. That's because it's waiting for /teleop_turtle to publish something



How To see the data being published on a topic

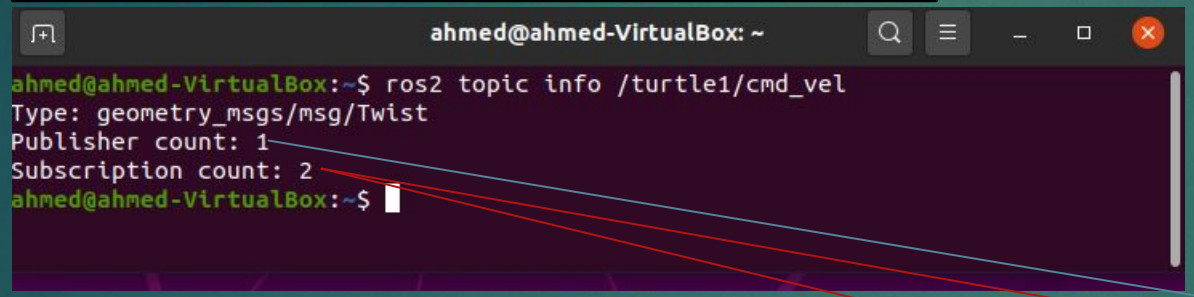
- Return to the terminal where turtle_teleop_key is running and use the arrows to move the turtle around. Watch the terminal where your echo is running at the same time, and you'll see position data being published for every movement you make:

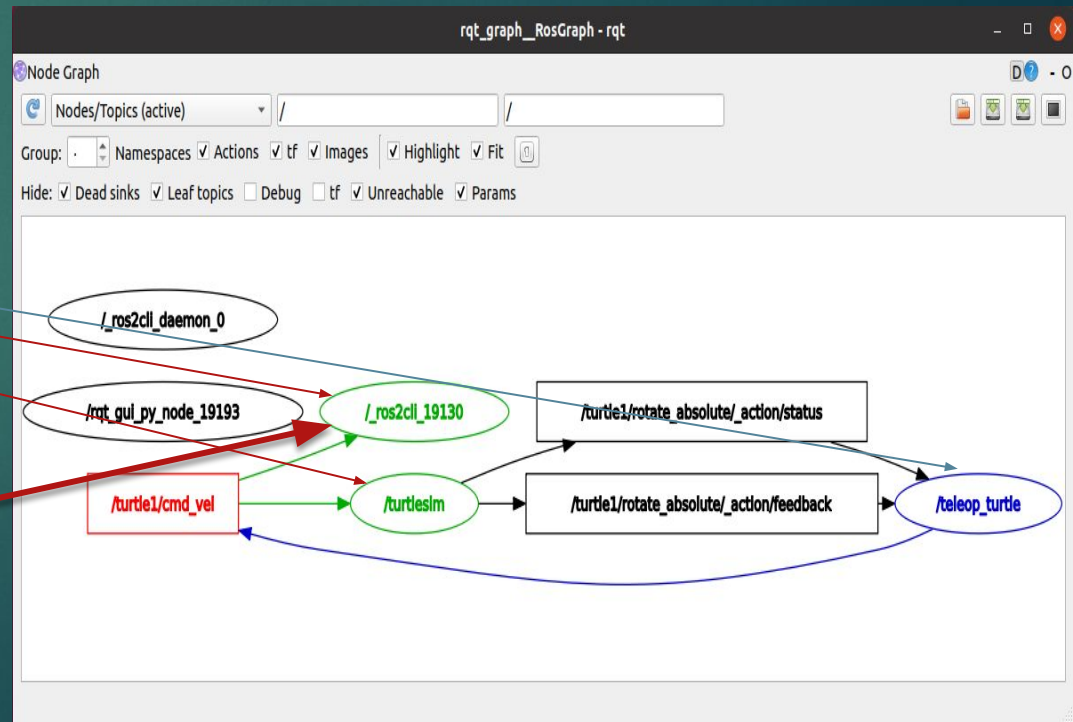


Ros2 topic info

- Topics don't have to only be point-to-point communication; it can be one-to-many, many-to-one, or many-to-many.
- To know further type:

```
ros2 topic info /turtle1/cmd_vel
```





Echo node that subscribe from topic /turtle1/cmd_vel

How to publish data onto a topic directly from the command line

- Now that you have the message structure, you can publish data onto a topic directly from the command line using:

```
ros2 topic pub <topic_name> <msg_type> '<args>'
```

- The '<args>' argument is the actual data you'll pass to the topic
- It's important to note that this argument needs to be input in YAML syntax
- Example

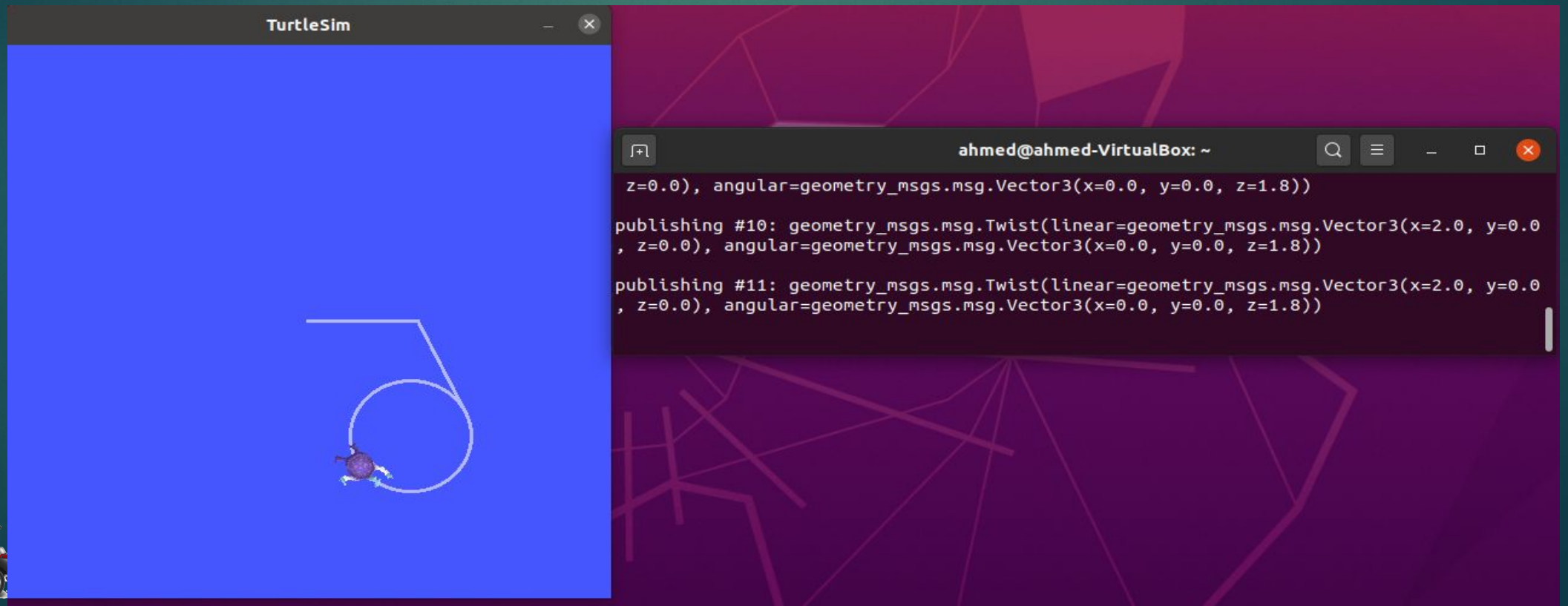
```
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

- --once is an optional argument meaning "publish one message then exit".

How to publish data onto a topic directly from the command line

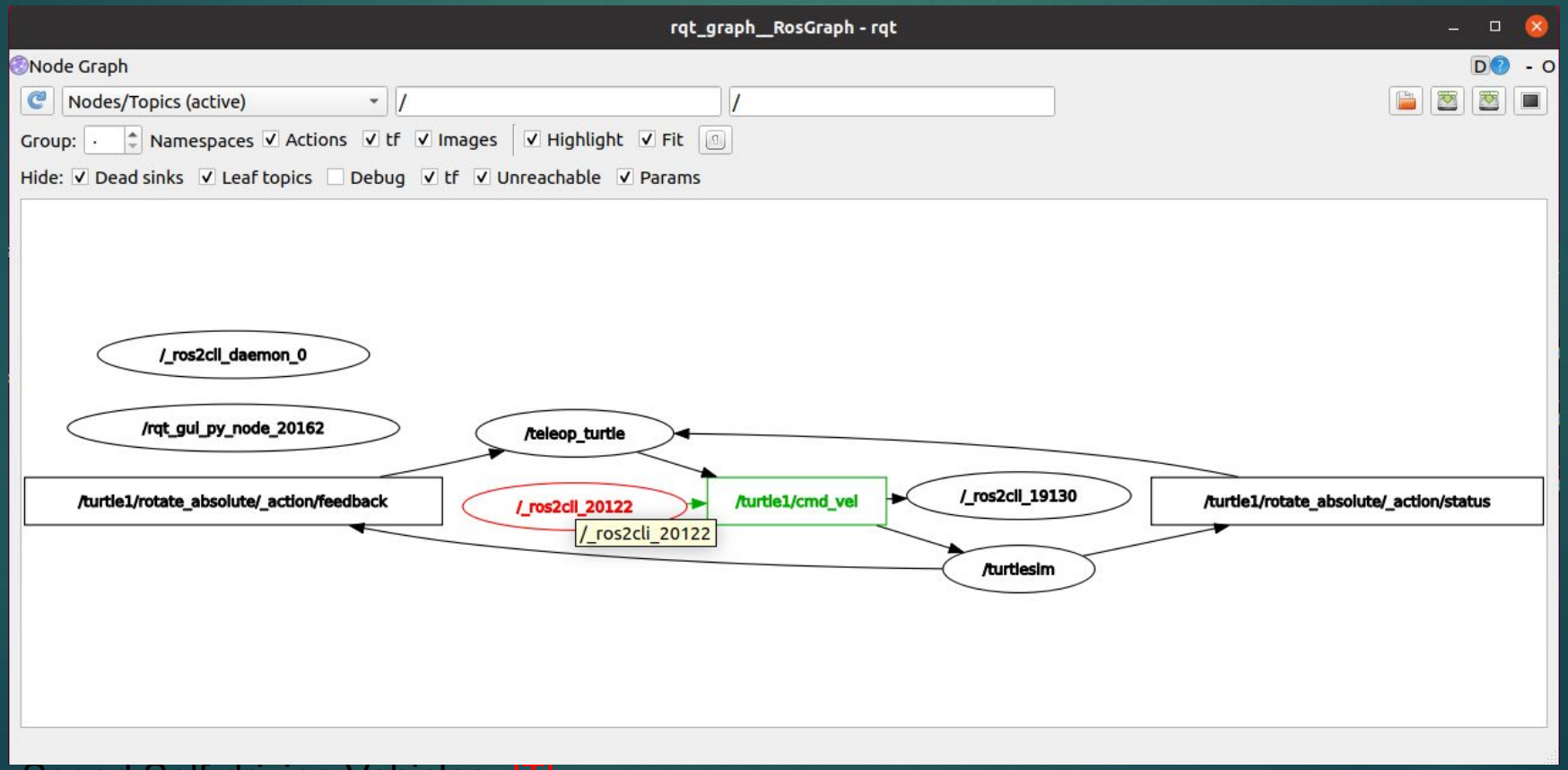
- We can specify the frequency of puplishing data using option:

```
ros2 topic pub –r<frequency number> /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```



publish data onto a topic directly from the command line

- You can refresh rqt_graph to see what's happening graphically.



ros2 topic hz

- you can report the rate at which data is published using:

```
ros2 topic hz /turtle1/cmd_vel
```

- It will return data on the rate at which the node is publishing data to the cmd_vel topic.

```
ahmed@ahmed-VirtualBox: ~
ahmed@ahmed-Virtual...   ahmed@ahmed-Virtual...   ahmed@ahmed-Virtual...
ahmed@ahmed-VirtualBox:~$ ros2 topic hz /turtle1/cmd_vel
average rate: 1.999
        min: 0.500s max: 0.500s std dev: 0.00043s window: 3
average rate: 2.001
        min: 0.498s max: 0.500s std dev: 0.00078s window: 6
average rate: 2.001
        min: 0.498s max: 0.500s std dev: 0.00069s window: 8
average rate: 2.001
        min: 0.498s max: 0.500s std dev: 0.00063s window: 10
```

C++ Publisher/ subscriber

- Create a ROS package

```
ros2 pkg create --build-type ament_cmake cpp_pubsub --dependencies rclcpp std_msgs
```

- Create nodes

```
touch publisher.cpp
```

- Add executable to CMakeLists.txt

```
add_executable(talker src/publisher.cpp)
ament_target_dependencies(talker rclcpp std_msgs)
install(TARGETS
  talker
  DESTINATION lib/${PROJECT_NAME})
```

- Make sure dependencies are correct in package.xml
- Build and run

```
colcon build --packages-select cpp_pubsub
ros2 run cpp_pubsub talker
```