1- Basics Of ROS 2

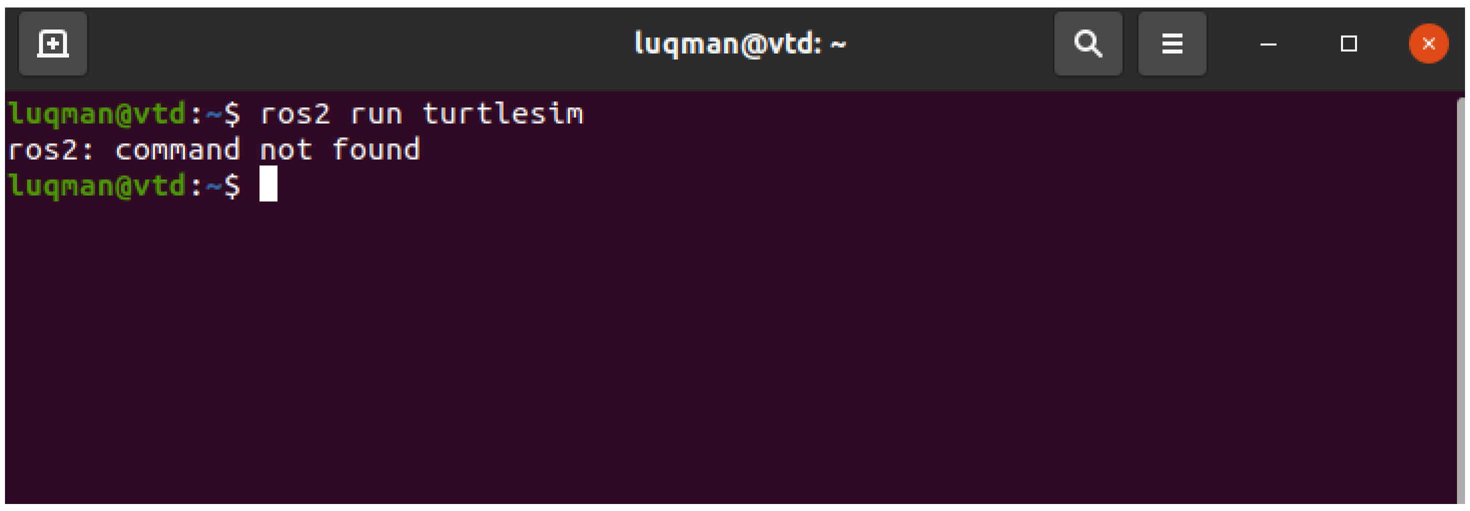
- [ ] Installation and Colcon Install
- [ ] Sourcing and Running TurtleBot Simulation
- [ ] Exploring TurtleBot Simulation
- [ ] Lets Create our Own package , Executable setup
- [ ] Writing our own Nodes
- [ ] A node that is Publisher and a Subscriber Simulatenously

1- Documentation link for easy understanding and referencing
- https://docs.ros.org/en/foxy/Tutorials/Configuring-ROS2-Environment.html

Sourcing and running Turtle-Bot Simulation

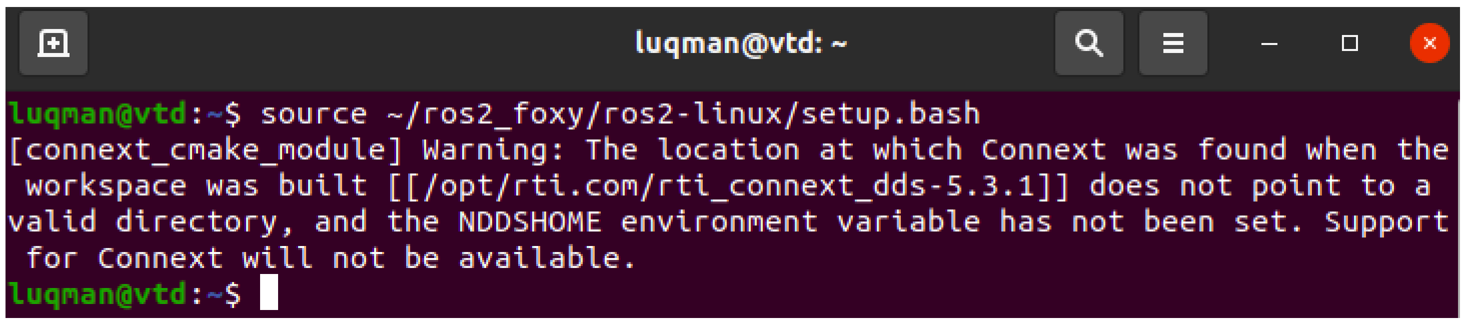
1. After Installing ROS2 , lets run Turtlesim

- Lets run the the basic ROS package which we will be exploring through OUT this video **TURTLESIMULATION**



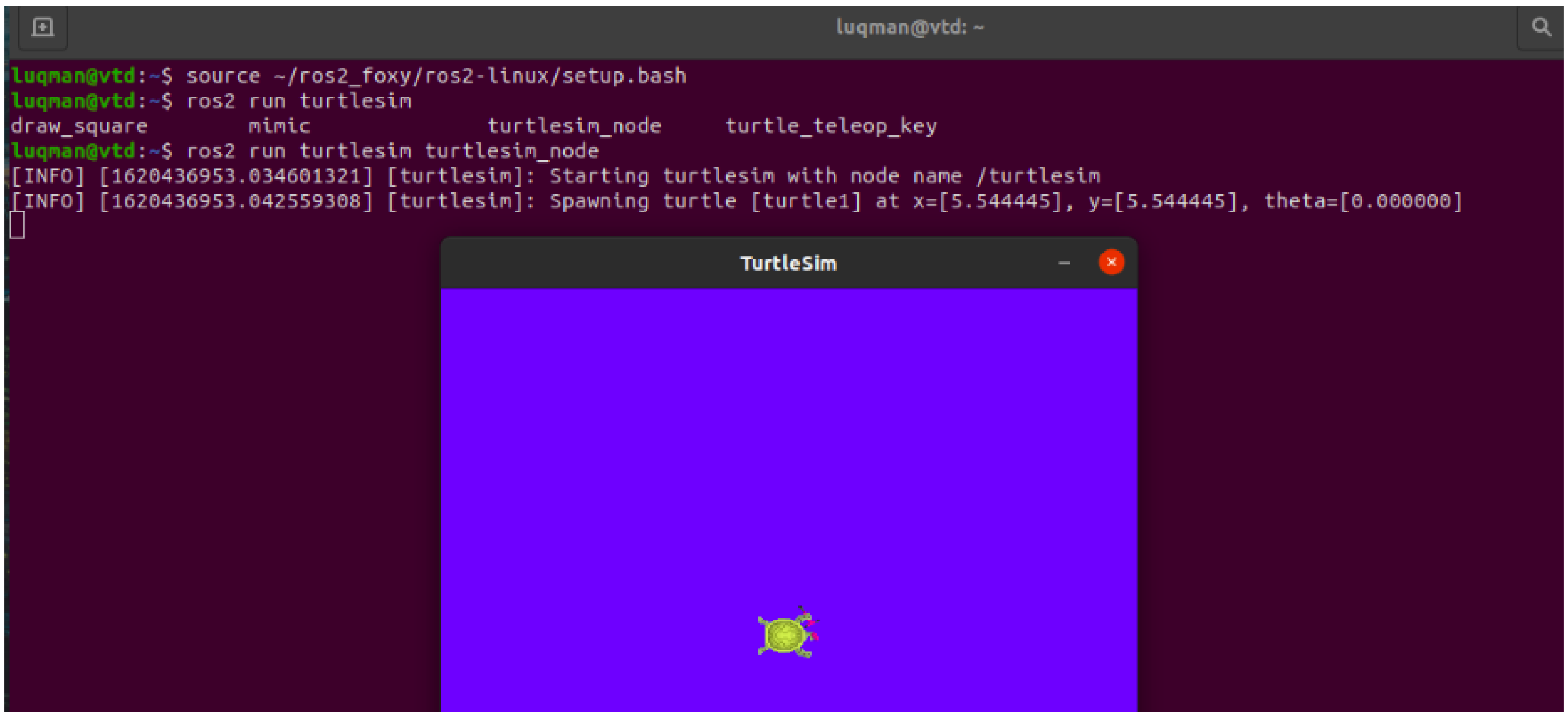
```
luqman@vtd:~$ ros2 run turtlesim
ros2: command not found
luqman@vtd:~$ 
```

- WHAT HAPPENED ????
- Sourcing ROS2 **source ~/ros2_foxy/ros2-linux/setup.bash**
  -Now we are get this LONG warning !



```
luqman@vtd:~$ source ~/ros2_foxy/ros2-linux/setup.bash
[connext_cmake_module] Warning: The location at which Connext was found when the
 workspace was built [[/opt/rti.com/rti_connext_dds-5.3.1]] does not point to a
valid directory, and the NDDSHOME environment variable has not been set. Support
 for Connext will not be available.
luqman@vtd:~$ 
```

- Install the solution with this command - **sudo apt-get install ros-foxy-rmw-connext-cpp** (for non-commercial Use Only)
- Finally the turtleBot and running a Node (explain syntax package,node)

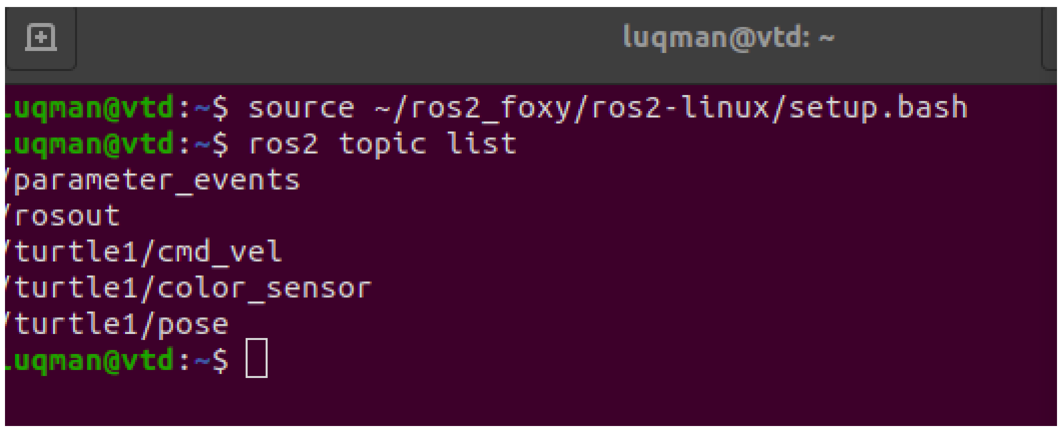


```
luqman@vtd:~$ source ~/ros2_foxy/ros2-linux/setup.bash
luqman@vtd:~$ ros2 run turtlesim
draw_square        mimic              turtlesim_node     turtle_teleop_key
luqman@vtd:~$ ros2 run turtlesim turtlesim_node
[INFO] [1620436953.034601321] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1620436953.042559308] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

- First lets Drive It (teleop Node)
- Enough Playing like a remote control car lets understand what is happening behind it and litterally what happens behind **ROBOTICS**

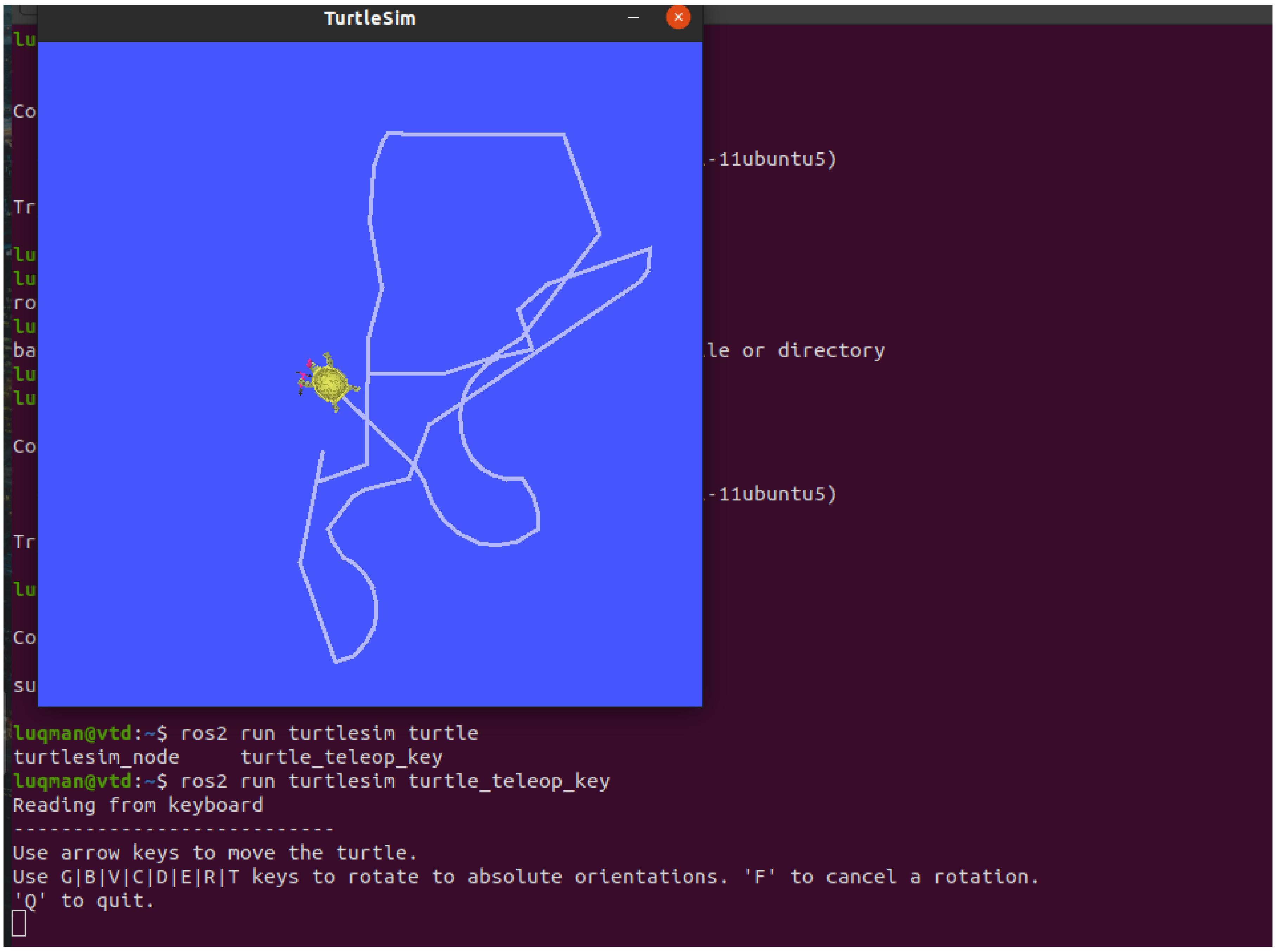
Explain ROS Communication (Nodes[sub/pub],Topic,Message)

- Command → **ros2 topic list** and **ros2 topic list -t**



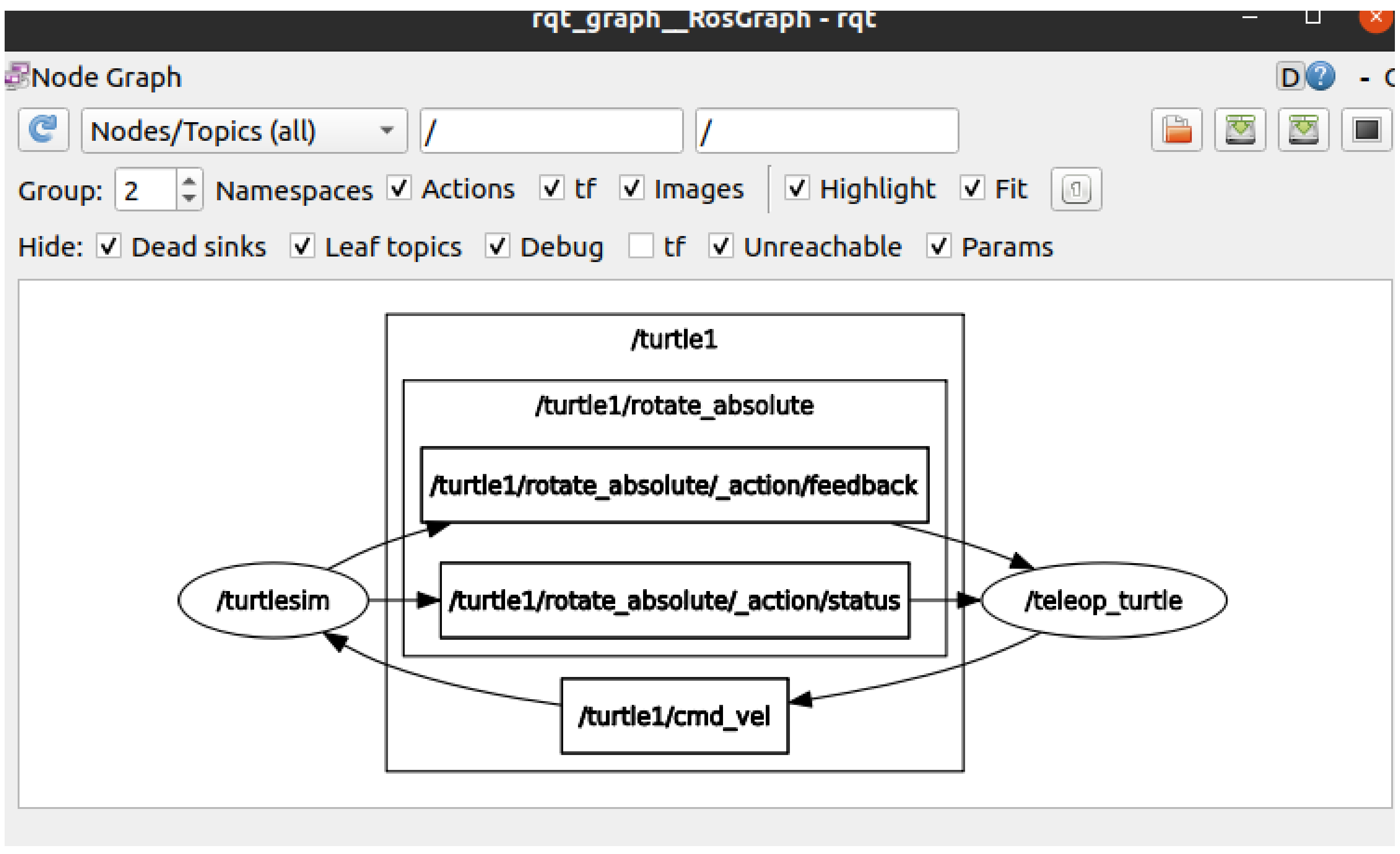
```
.uqman@vtd:~$ source ~/ros2_foxy/ros2-linux/setup.bash
.uqman@vtd:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
.uqman@vtd:~$
```

- Starting a node that becomes publisher → **ros2 run tuetlesim-teleop**



```
luqman@vtd:~$ ros2 run turtlesim turtle
turtlesim_node      turtle_teleop_key
luqman@vtd:~$ 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 rotation.
'Q' to quit.
```

- ros2 topic info
  -- explain publisher and subscriber
- ros2 topic echo
  -- show them live log of things
- Now let me introduce its gui verion (remove all to active) or **ros2 node info /turtlesim_teleop** or **/turtlesim**


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
Group: 2
✓ Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit
Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug tf ✓ Unreachable ✓ Params
/turtle1
/turtle1/rotate_absolute
/turtle1/rotate_absolute/_action/feedback
/turtlesim
/turtle1/rotate_absolute/_action/status
/teleop_turtle
/turtle1/cmd_vel

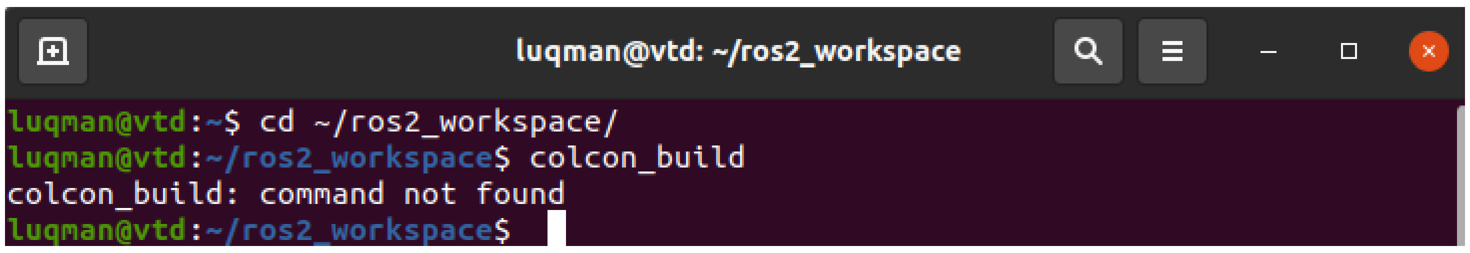
Lets Command the robot Statically

- **ros2 interface show geometry_msgs/msg/Twist**
- **ros2 topic pub --rate 1 /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}}"**

Lets Create our Own package , Starting Real Stuff

- First we need to create a directory **mkdir -p ~/ros2_workspace/src**
- in /src folder **git clone https://github.com/ros/ros_tutorials.git -b foxy-devel**
- If you colcon build it without sourcing ros foxy installation it will generate errors

What ??



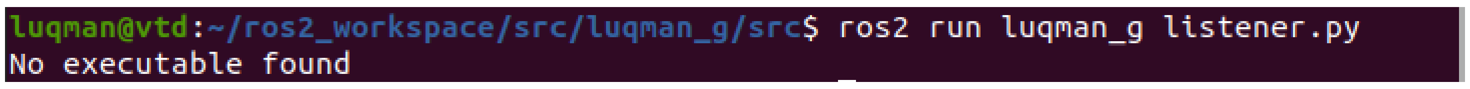
```
luqman@vtd:~$ cd ~/ros2_workspace/
luqman@vtd:~/ros2_workspace$ colcon_build
colcon_build: command not found
luqman@vtd:~/ros2_workspace$ 
```

- Now its time i should not hide the proper way of doing things , I think you have grown enough :D

#source ~/ros2_foxy/ros2-linux/setup.bash
#source /usr/share/colcon_cd/function/colcon_cd.sh
#export _colcon_cd_root=~/ros2_foxy

Add these upper lines to your bashrc file

- Create a new package **ros2 pkg create luqman_g --build-type ament_python** *(for python it is neccesory as executable file making and compiling is different than cpp)*
- Using codes from wiki which are object oriented to understand custom Publisher/Subscriber
- Pre written code from *https://docs.ros.org/en/foxy/Tutorials/Writing-A-Simple-Py-Publisher-And-Subscriber.html#build-and-run*



```
luqman@vtd:~/ros2_workspace/src/luqman_g/src$ ros2 run luqman_g listener.py
No executable found
```

What happened ?? We have to check things in order (first check linking,sourcing,building)

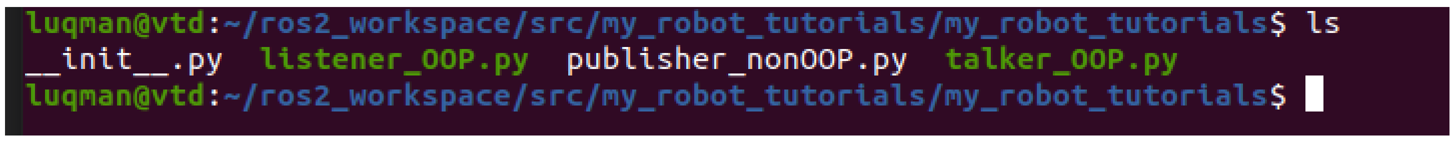
- python executable file process
  - Are my files Executeables
  - Does the package knows about which files to execute
  - Show process of executable file adding into setup file
  - Libraries required to run files .. are they linked ?

- Look at your scripts . Does all libraries included in package ? rcl and std in package.xml
- So we have to add our python files to execution list → entry points in setup.py file

- Then just build my package **colcon build --packages-select luqman_g** (this one is only for my package)

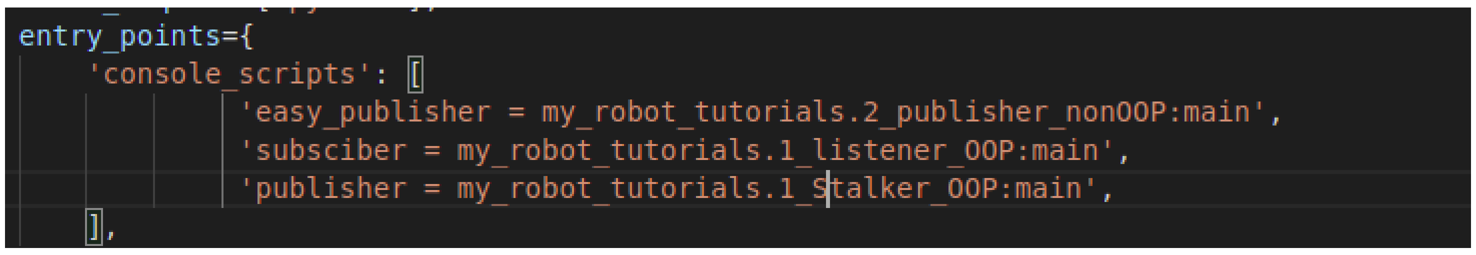
Writing our own Scripts (Real Deal)

- Writing our own Publisher Subscriber which are NON OOP :)
- It is not showing in executables and white in color… There you have it → need to make EXECUTABLE



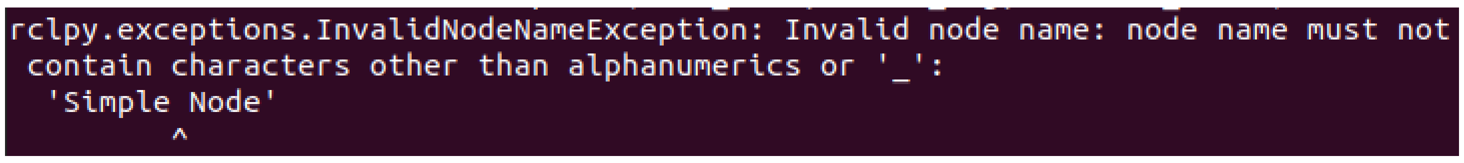
```
luqman@vtd:~/ros2_workspace/src/my_robot_tutorials/my_robot_tutorials$ ls
__init__.py  listener_OOP.py  publisher_nonOOP.py  talker_OOP.py
luqman@vtd:~/ros2_workspace/src/my_robot_tutorials/my_robot_tutorials$
```

There you go



```
entry_points={
    'console_scripts': [
            'easy_publisher = my_robot_tutorials.2_publisher_nonOOP:main',
            'subsciber = my_robot_tutorials.1_listener_OOP:main',
            'publisher = my_robot_tutorials.1_Stalker_OOP:main',
    ],
```

Got an error because of Naming Convention of Node Name.



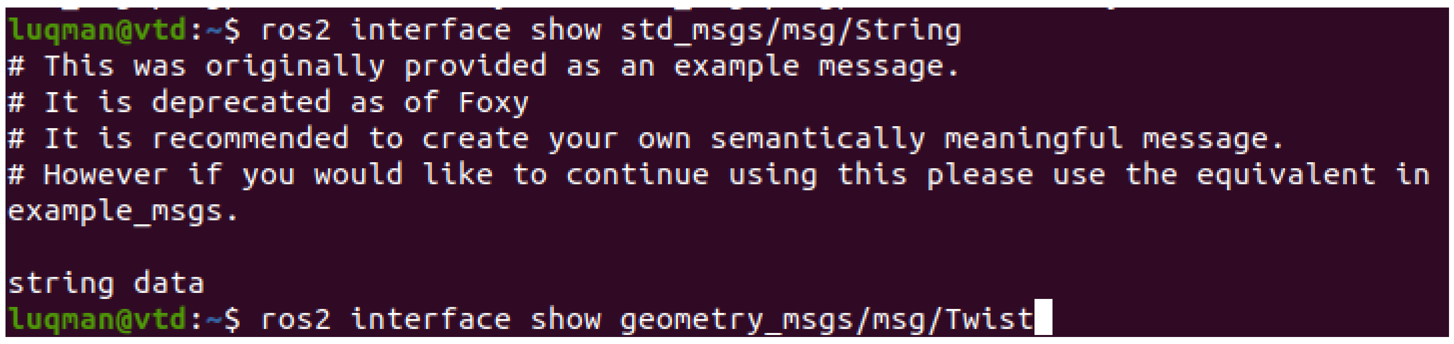
```
rclpy.exceptions.InvalidNodeNameException: Invalid node name: node name must not
 contain characters other than alphanumerics or '_':
  'Simple Node'
         ^
```

Then wheile looking at OOP pub/sub i will be writing non OOP pub

- Giving proper names to topic so when running all 4 we can distinguish

---

Now lets find about the message types and write our own Command velocity publisher for turtle simulation



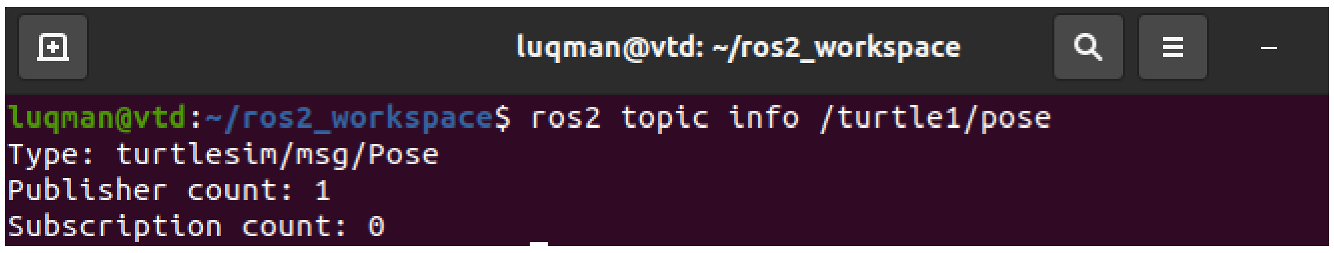
```
luqman@vtd:~$ ros2 interface show std_msgs/msg/String
# This was originally provided as an example message.
# It is deprecated as of Foxy
# It is recommended to create your own semantically meaningful message.
# However if you would like to continue using this please use the equivalent in
example_msgs.

string data
luqman@vtd:~$ ros2 interface show geometry_msgs/msg/Twist
```

- Write a script for publishing just linear velocity for 2 seconds (3_turtlesim_pub)
- Then add Angular values as well

A node that is Publisher and a Subscriber Simulatenously

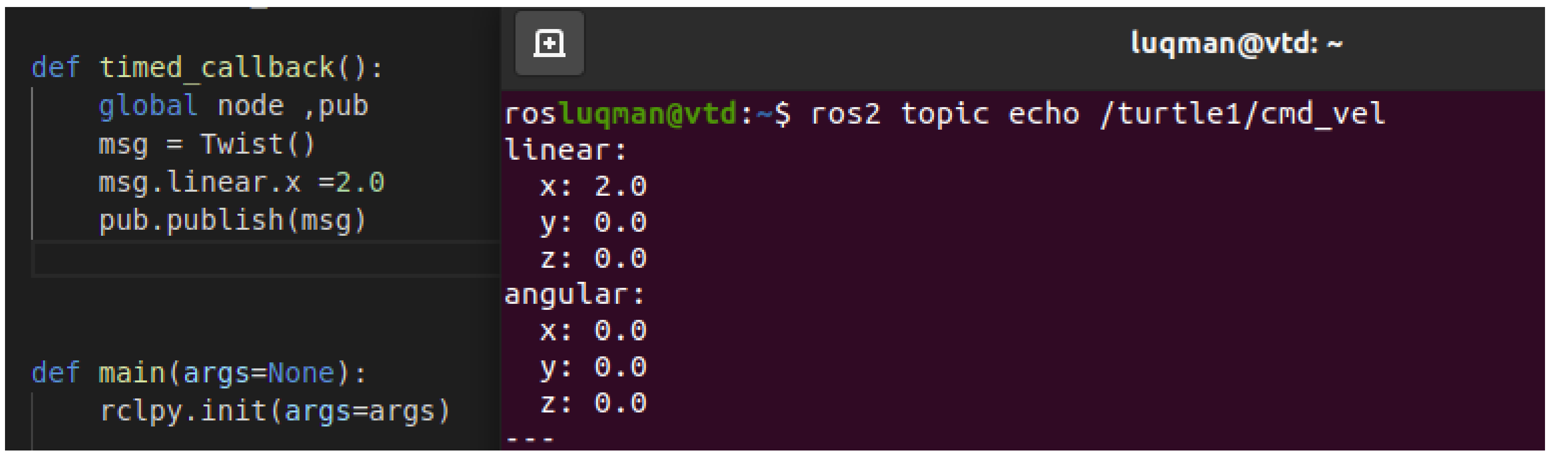
Creating a node containing pub(cmd_vel)and sub(pose) , to make the turtle bot move to a specific position

- Explore the pose message used (ros2 topic info /turtle1/pose) → read its message type



```
luqman@vtd:~/ros2_workspace$ ros2 topic info /turtle1/pose
Type: turtlesim/msg/Pose
Publisher count: 1
Subscription count: 0
```

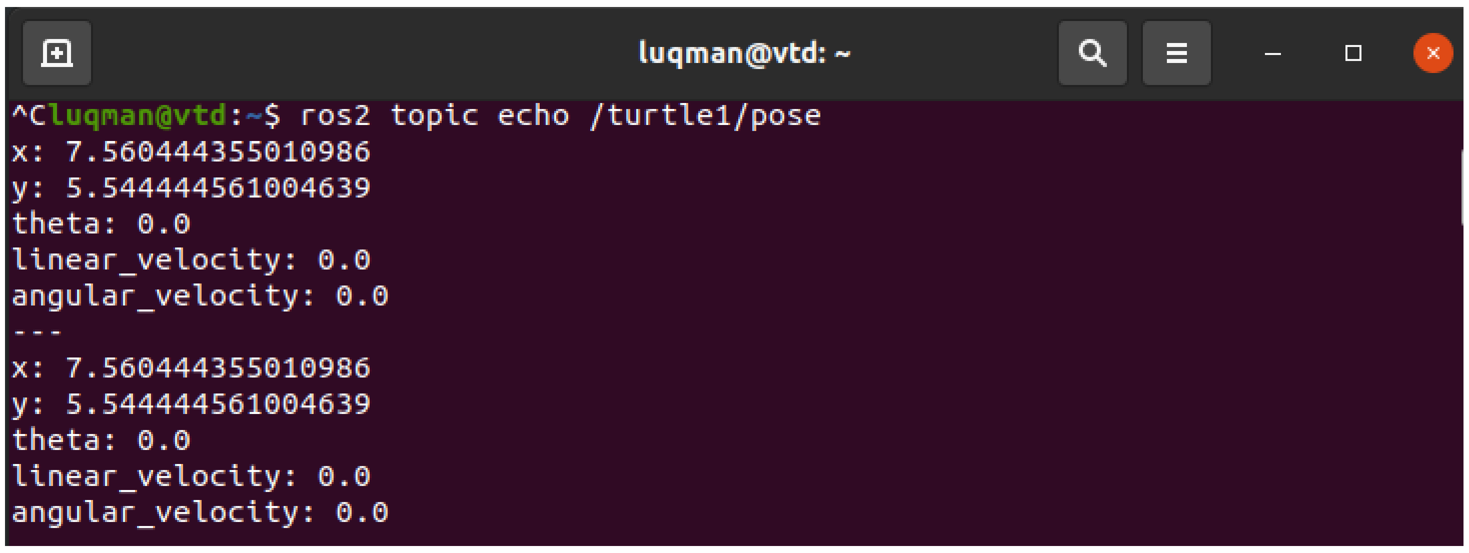
- ros2 interface show **ros2 interface show turtlesim/msg/Pose**
- (important) I spent time —→ geometeric/msgs/Pose because for Twist it is using geometric message as i remeber in ROS1 it used geometric_msg/pose
- ros2 topic echo /turtle1/pose and decode it through cmd_vel message



```
def timed_callback():
    global node ,pub
    msg = Twist()
    msg.linear.x =2.0
    pub.publish(msg)


def main(args=None):
    rclpy.init(args=args)
```

```
rosluqman@vtd:~$ ros2 topic echo /turtle1/cmd_vel
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

For the pose message we have



```
^Cluqman@vtd:~$ ros2 topic echo /turtle1/pose
x: 7.560444355010986
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 7.560444355010986
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
```

Just Adding Launch Files