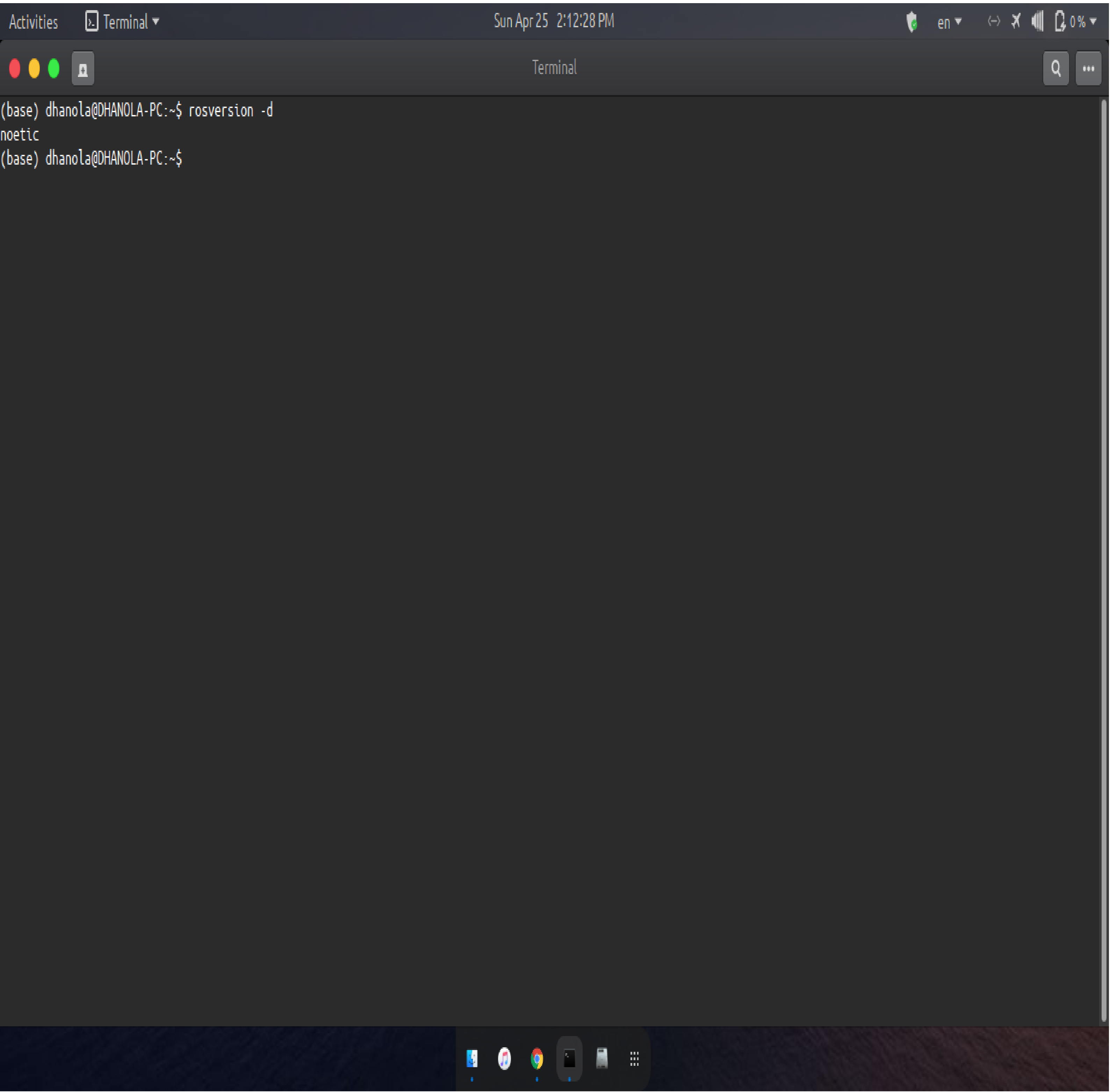
EXPERIMENT-9

Installing ROS And Other Packages, Basic Programs.

Step1 : Installation Of ROS And Dependencies:

Version Of ROS Installed:-

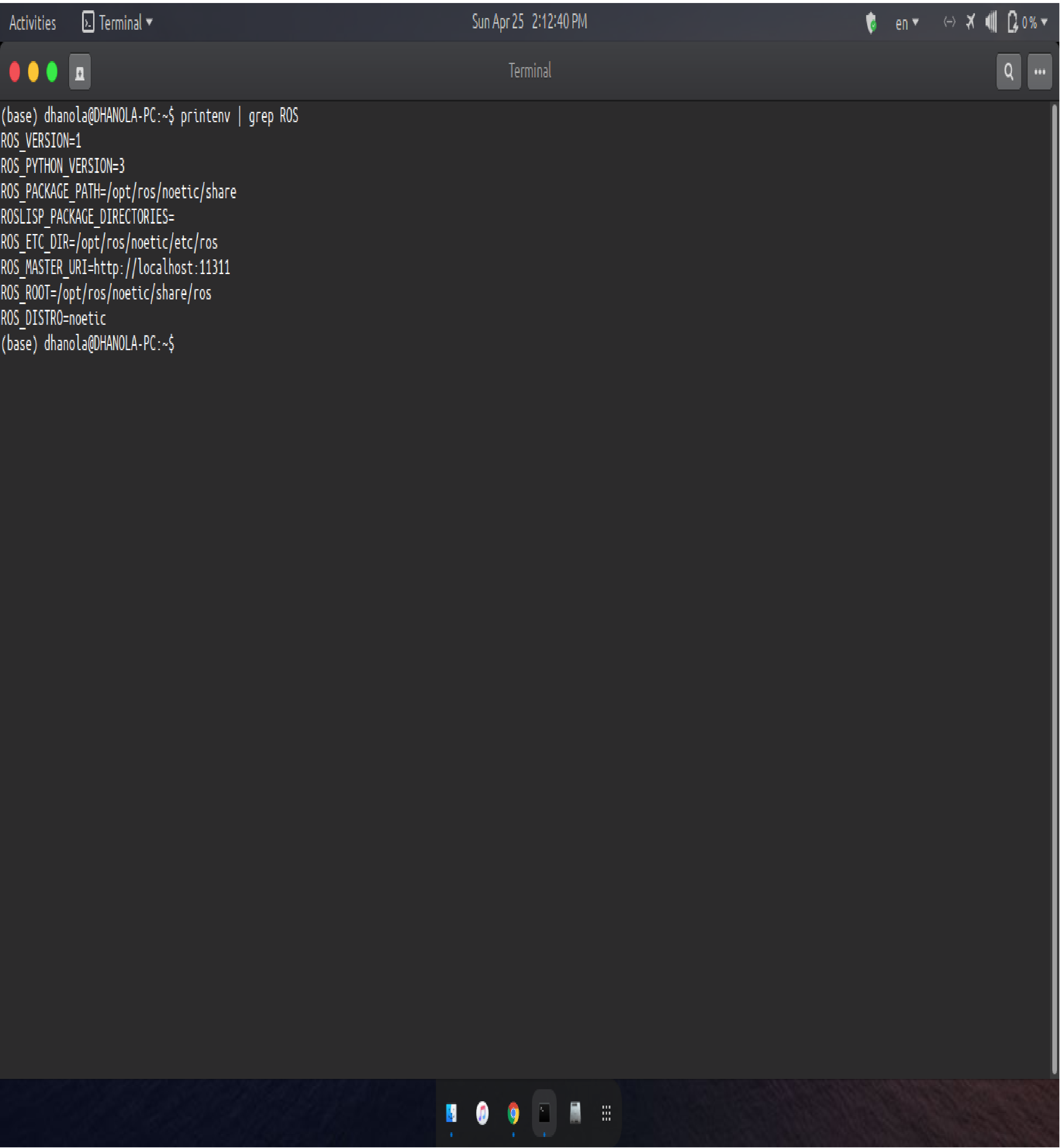


```
(base) dhanola@DHANOLA-PC:~$ rosversion -d
noetic
(base) dhanola@DHANOLA-PC:~$
```

Step 2: Managing The Environment:

Check The Environment Settings With The Command:-

```
printenv | grep ROS
```

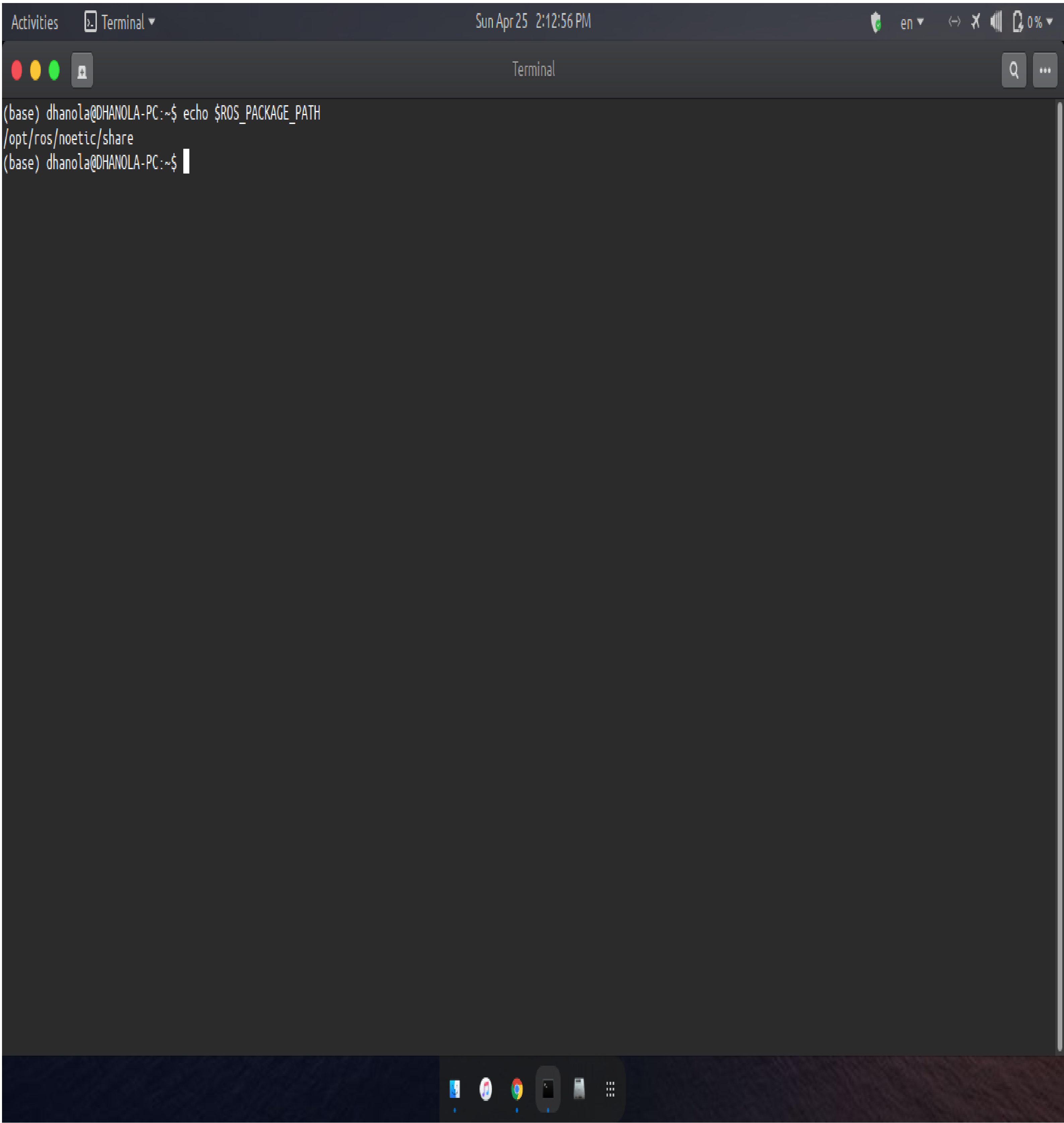


```
(base) dhanola@DHANOLA-PC:~$ printenv | grep ROS
ROS_VERSION=1
ROS_PYTHON_VERSION=3
ROS_PACKAGE_PATH=/opt/ros/noetic/share
ROSLISP_PACKAGE_DIRECTORIES=
ROS_ETC_DIR=/opt/ros/noetic/etc/ros
ROS_MASTER_URI=http://localhost:11311
ROS_ROOT=/opt/ros/noetic/share/ros
ROS_DISTRO=noetic
(base) dhanola@DHANOLA-PC:~$
```

Step 3: Creating A ROS Workspace:

Use The Command:-

```
echo $ROS_PACKAGE_PATH
```

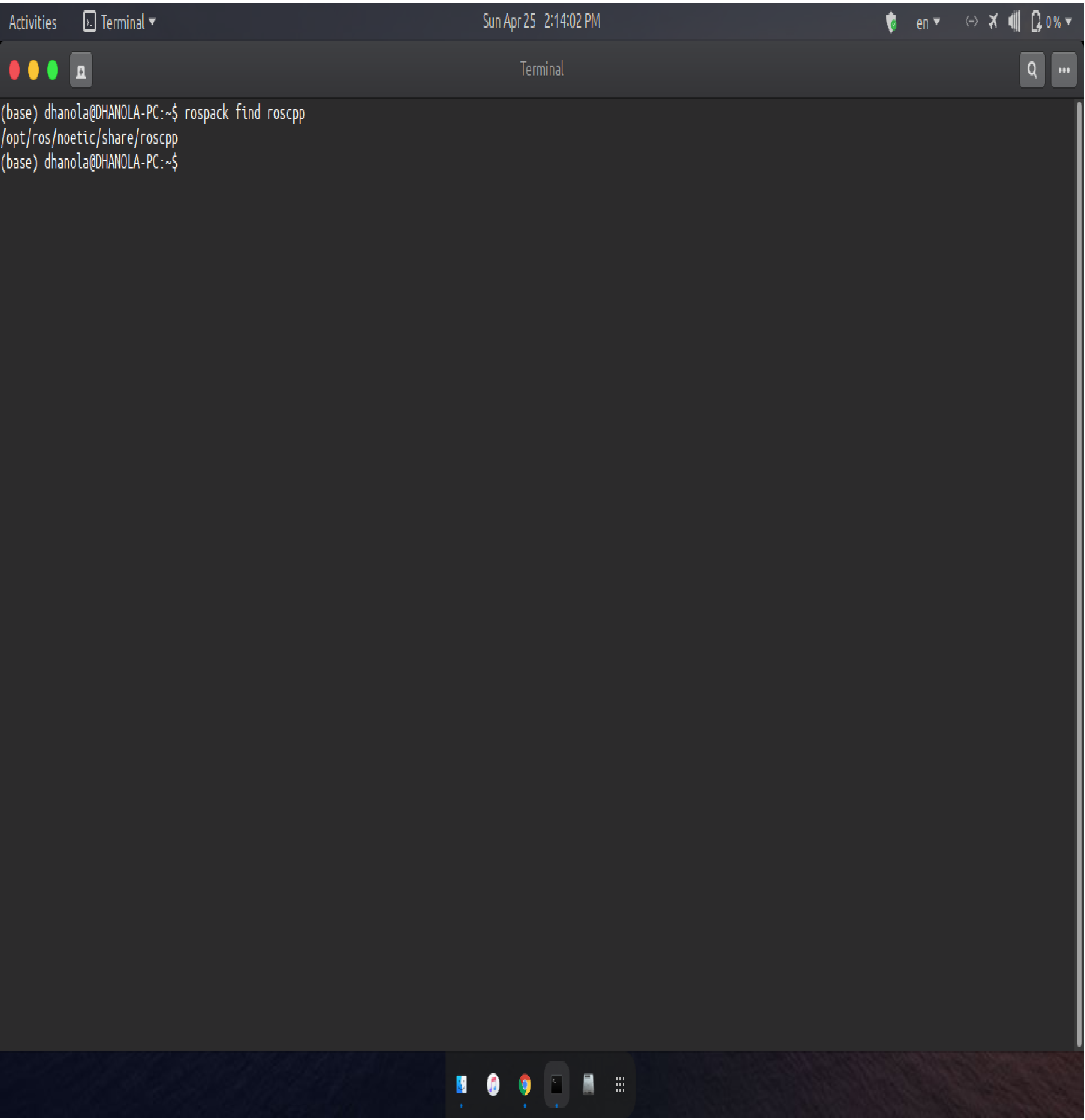


```
(base) dhanola@DHANOLA-PC:~$ echo $ROS_PACKAGE_PATH
/opt/ros/noetic/share
(base) dhanola@DHANOLA-PC:~$
```

Step 4: Filesystem Tools:

A) Using Rospack :- Allows To Get Information About Packages.

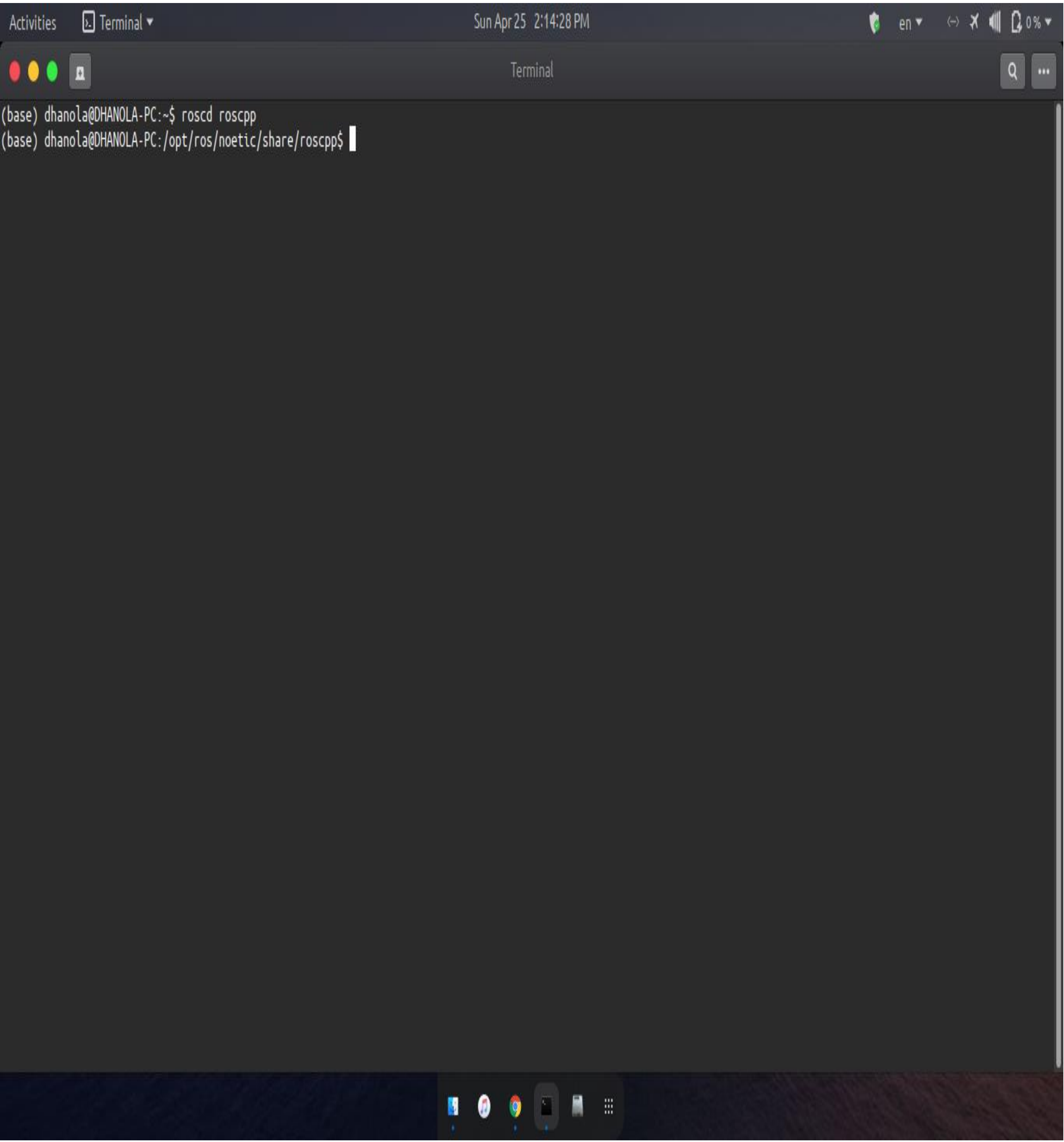
```
rospack find roscpp
```



```
(base) dhanola@DHANOLA-PC:~$ rospack find roscpp
/opt/ros/noetic/share/roscpp
(base) dhanola@DHANOLA-PC:~$
```

B) Using Roscd:- Allows To Change Directory To A Package Or Stack.

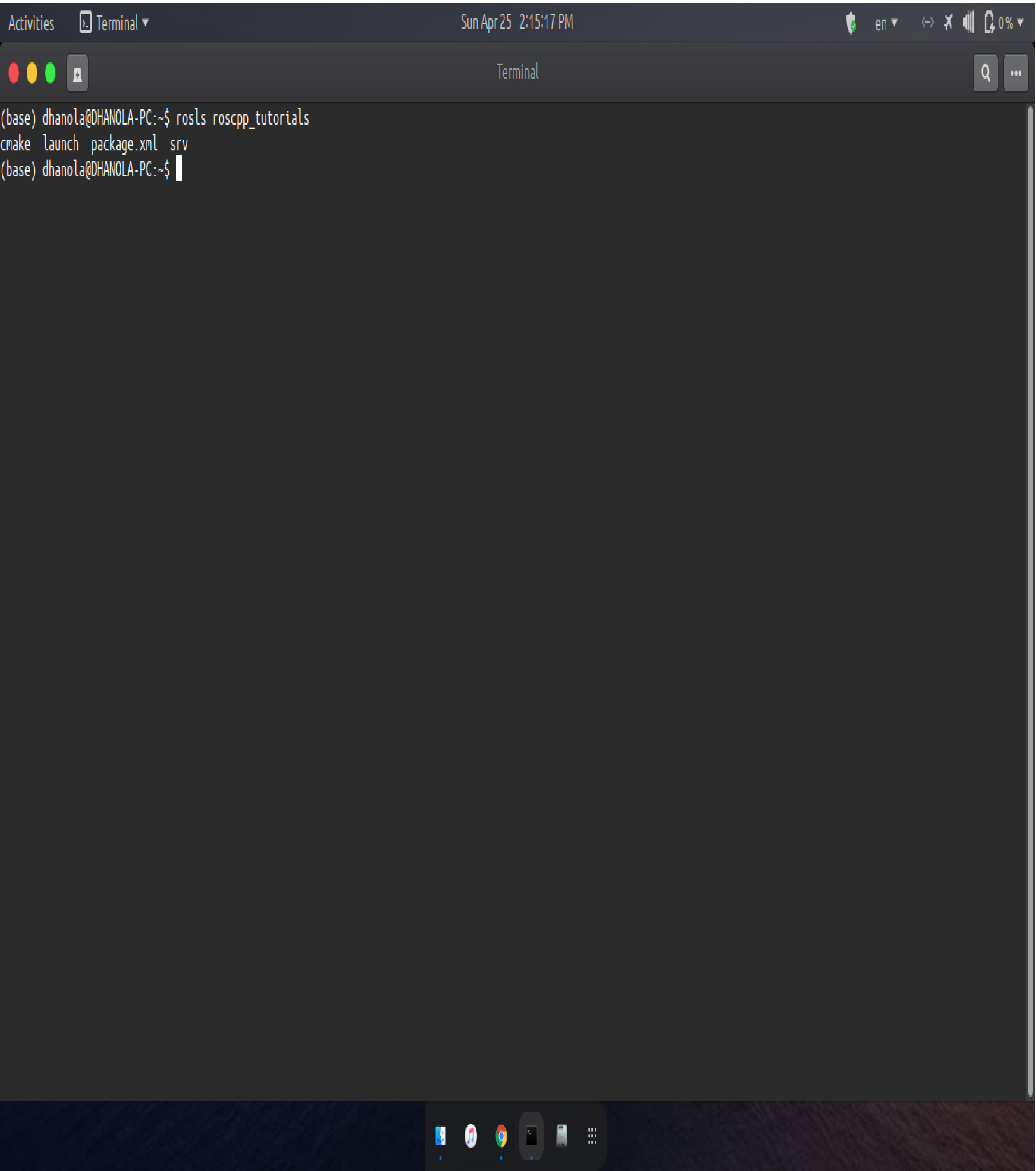
roscd roscpp



```
(base) dhanola@DHANOLA-PC:~$ roscd roscpp
(base) dhanola@DHANOLA-PC:/opt/ros/noetic/share/roscpp$
```

C)Using Rosls:- Allows To Ls Directly In A Package Rather Than Absolute Path.

rosls roscpp_tutorials

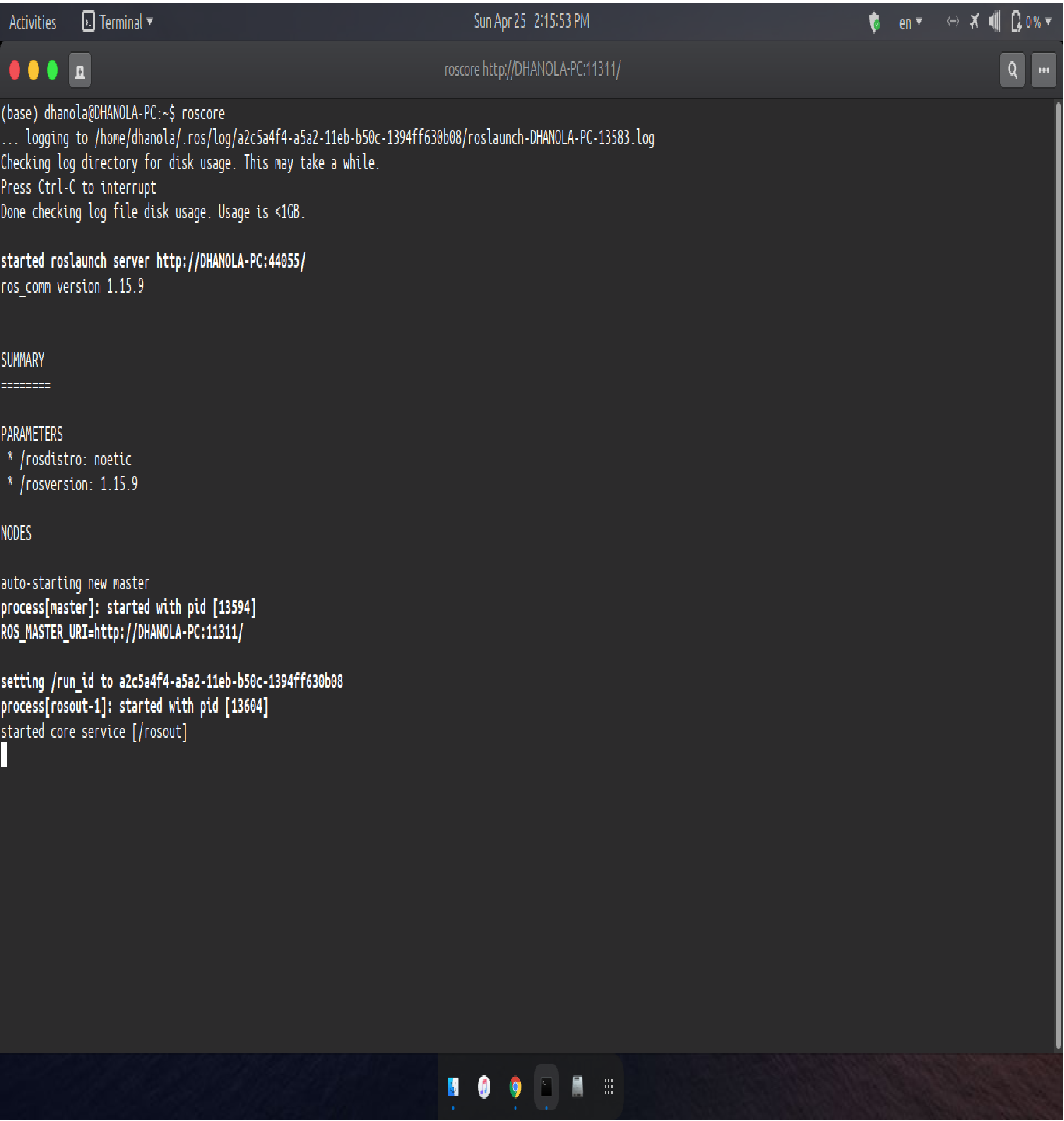


```
(base) dhanola@DHANOLA-PC:~$ rosls roscpp_tutorials
cmake  launch  package.xml  srv
(base) dhanola@DHANOLA-PC:~$
```

Step 5: ROS Nodes :

A) Run The Command Roscore:-

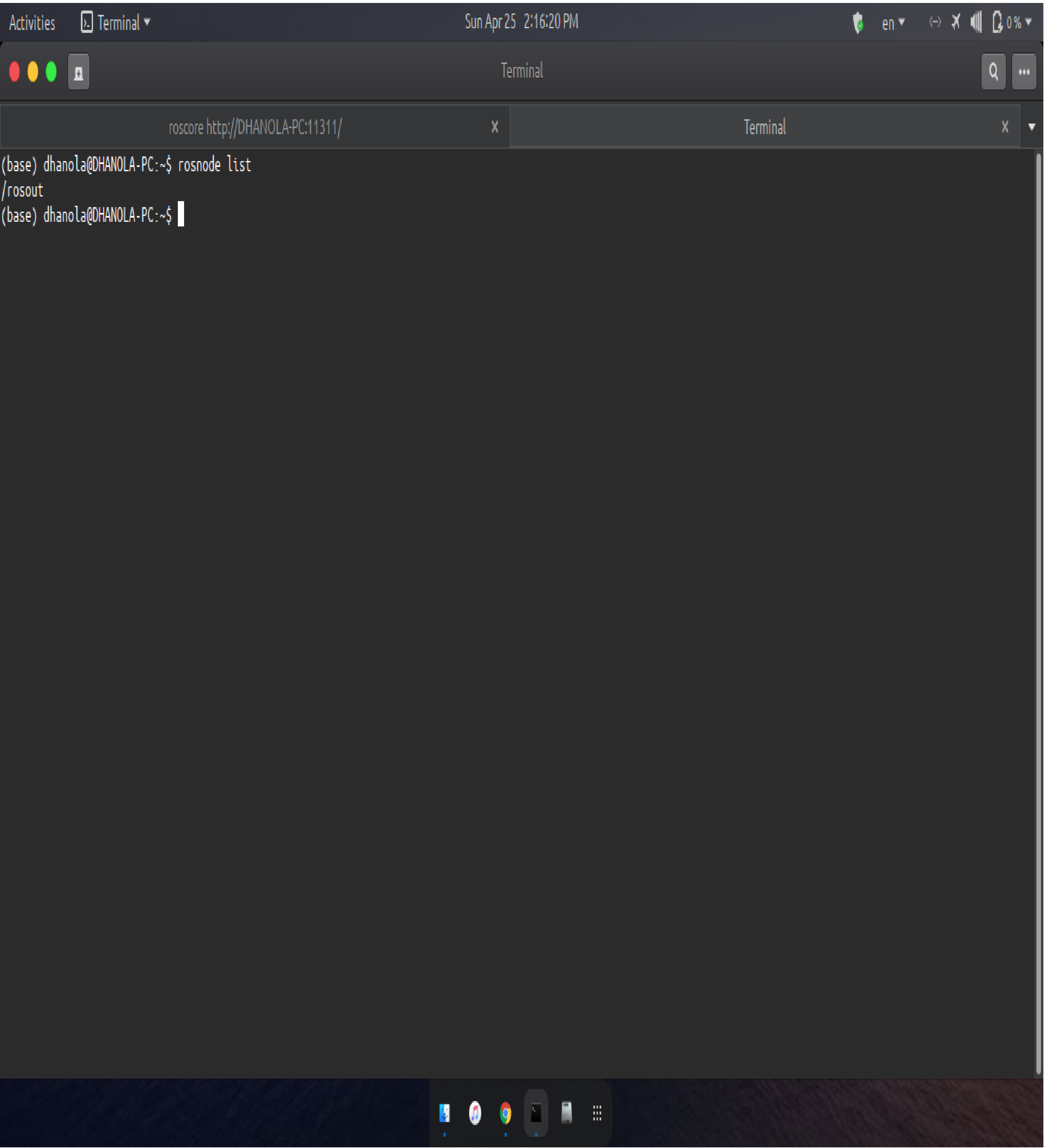
```
roscore
```



```
(base) dhanola@DHANOLA-PC:~$ roscore
... logging to /home/dhanola/.ros/log/a2c5a4f4-a5a2-11eb-b50c-1394ff630b08/roslaunch-DHANOLA-PC-13583.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://DHANOLA-PC:44055/
ros_comm version 1.15.9


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.9

NODES

auto-starting new master
process[master]: started with pid [13594]
ROS_MASTER_URI=http://DHANOLA-PC:11311/

setting /run_id to a2c5a4f4-a5a2-11eb-b50c-1394ff630b08
process[rosout-1]: started with pid [13604]
started core service [/rosout]
```

B) Using Rosnode:-

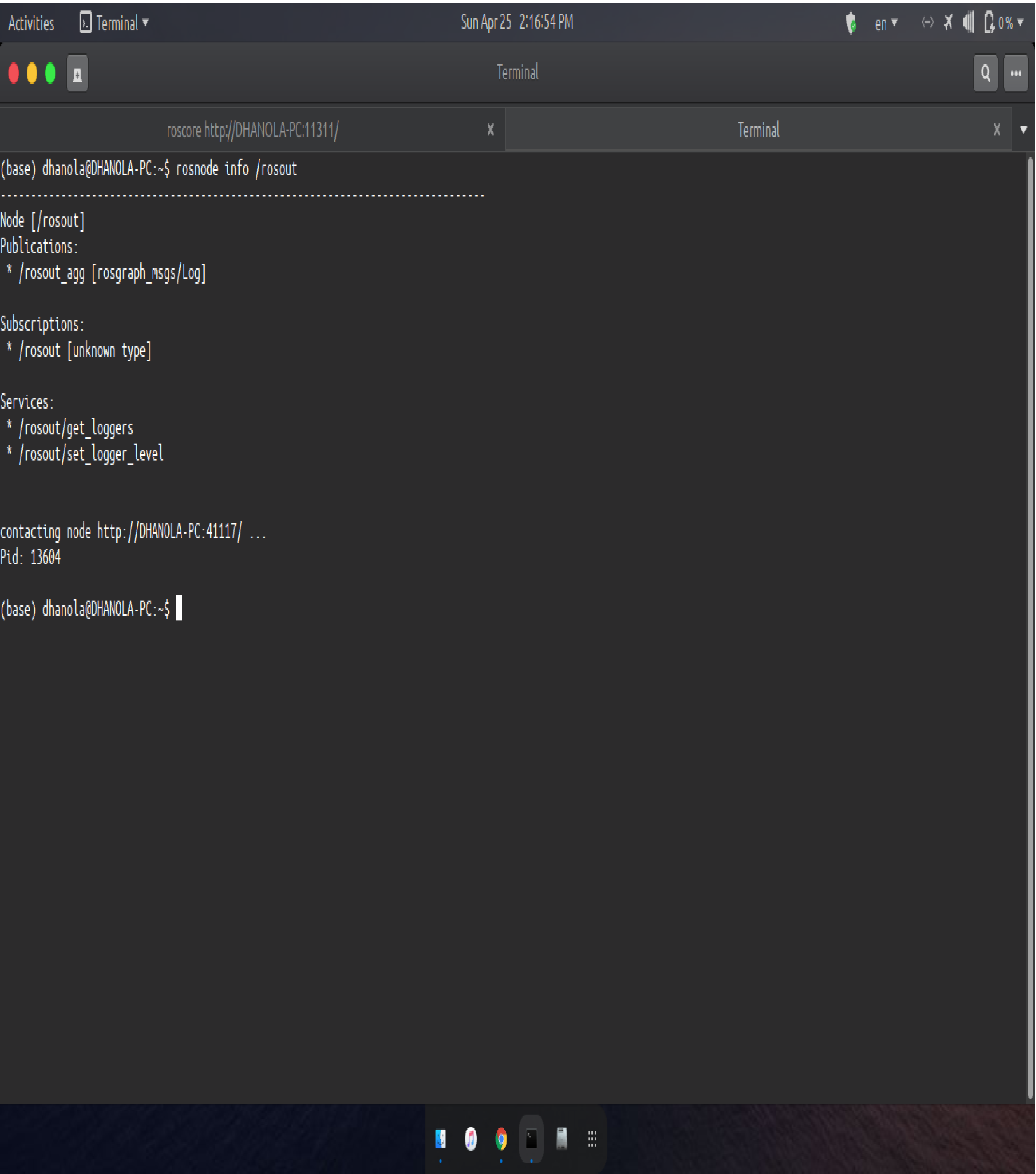
rosnode list



```
(base) dhanola@DHANOLA-PC:~$ rosnode list
/rosout
(base) dhanola@DHANOLA-PC:~$ 
```

Lets Try To Get Some Info By The Command :-

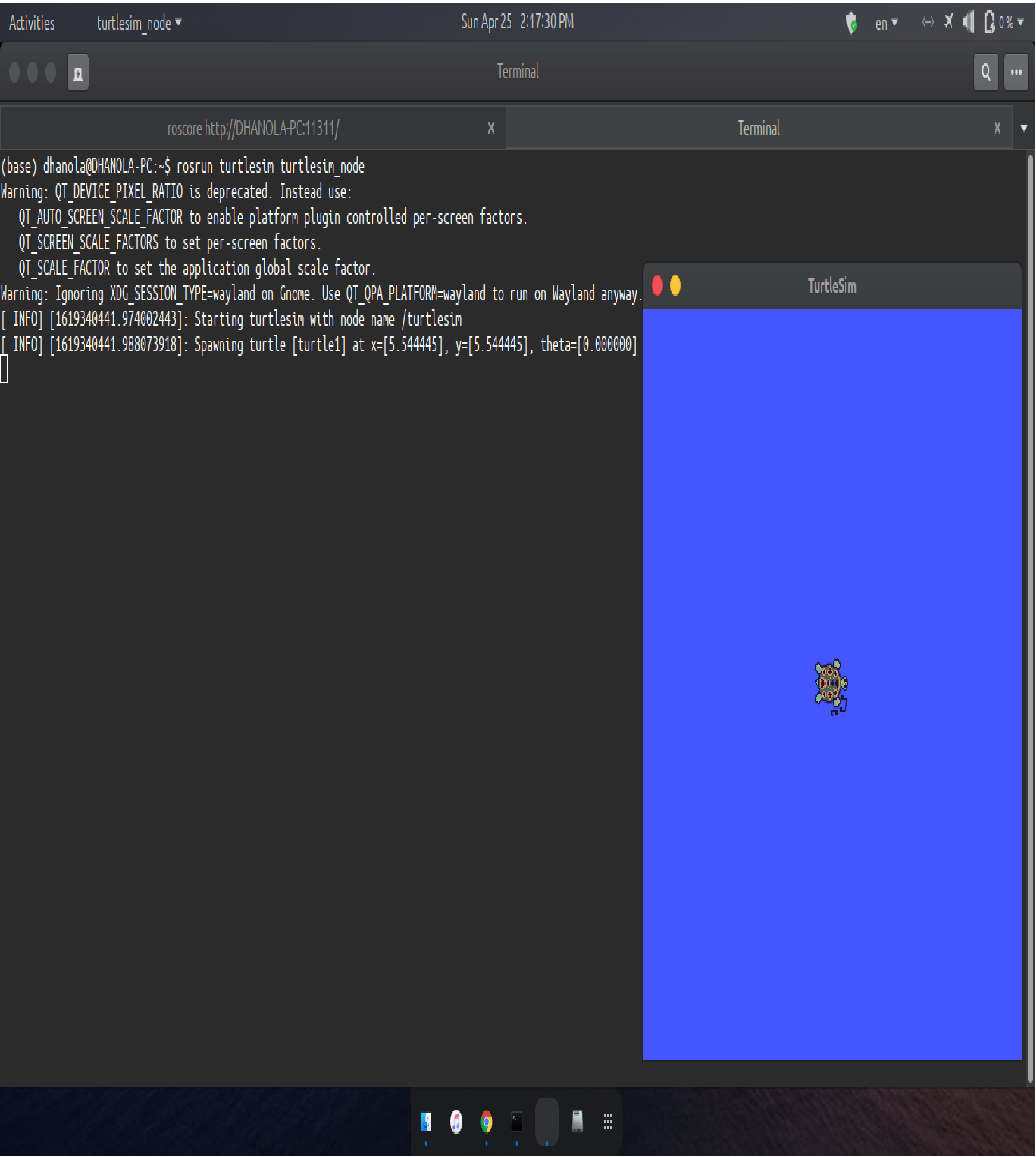
rosnode info /rosout



```
(base) dhanola@DHANOLA-PC:~$ rosnode info /rosout
--------------------------------------------------------------------------------
Node [/rosout]
Publications:
 * /rosout_agg [rosgraph_msgs/Log]

Subscriptions:
 * /rosout [unknown type]

Services:
 * /rosout/get_loggers
 * /rosout/set_logger_level


contacting node http://DHANOLA-PC:41117/ ...
Pid: 13604

(base) dhanola@DHANOLA-PC:~$
```

C) Using Rosrun:-

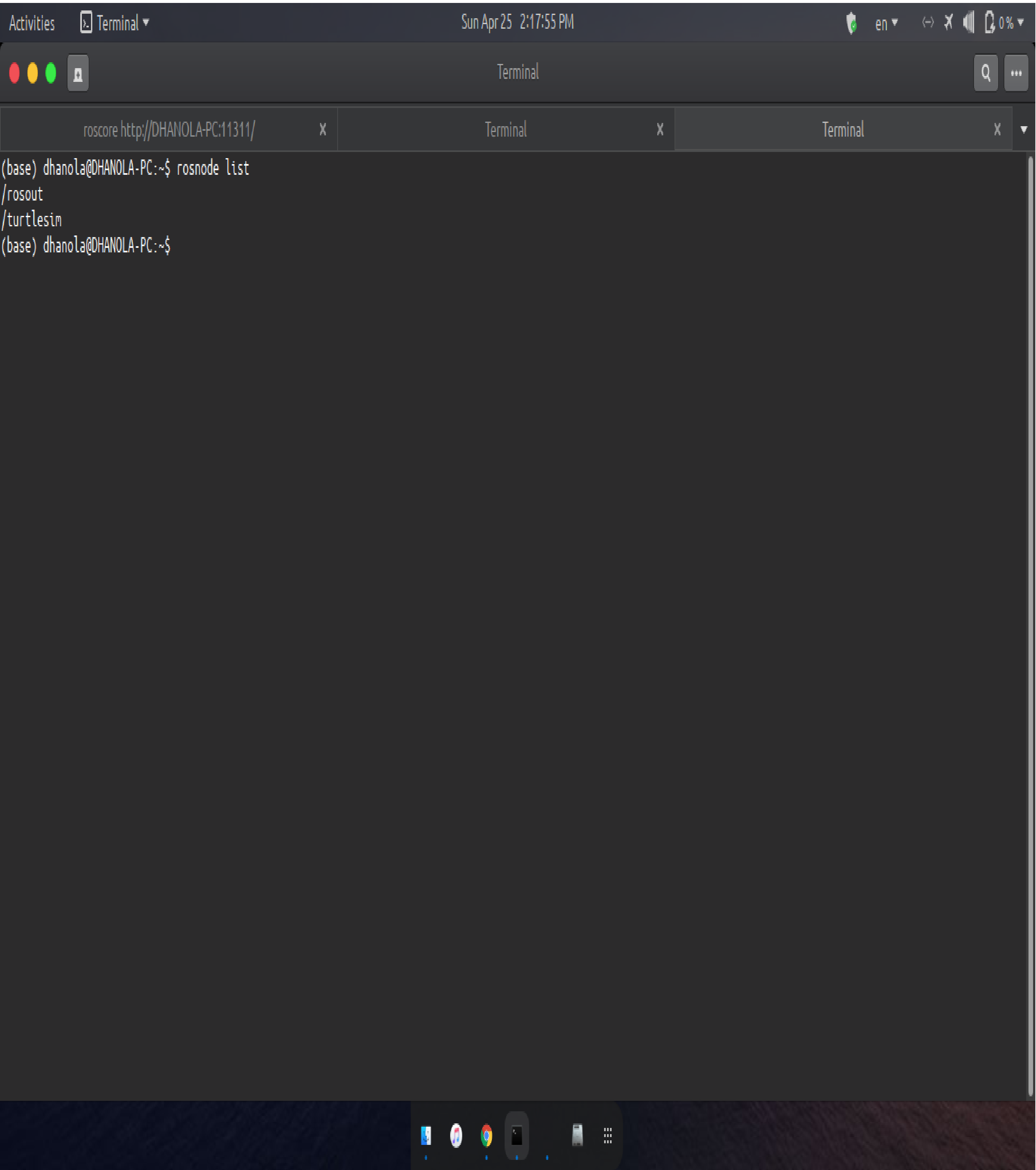
```
rosrun turtlesim turtlesim_node
```



```
(base) dhanola@DHANOLA-PC:~$ rosrun turtlesim turtlesim_node
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[ INFO] [1619340441.974002443]: Starting turtlesim with node name /turtlesim
[ INFO] [1619340441.988073918]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

Now If We Open Another Terminal And See The Output For The Command:-

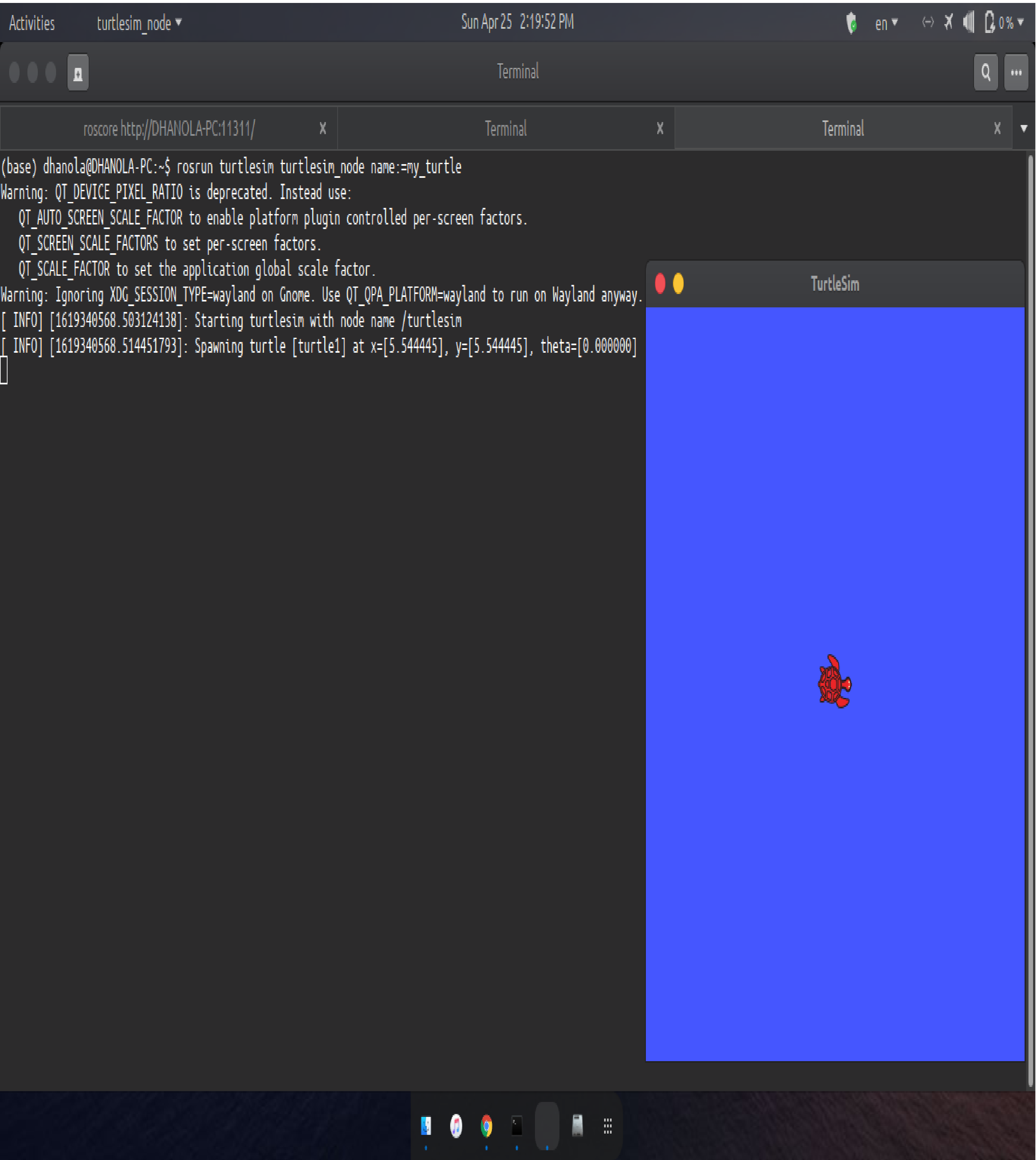
rosnode list



```
(base) dhanola@DHANOLA-PC:~$ rosnode list
/rosout
/turtlesim
(base) dhanola@DHANOLA-PC:~$
```

Now Lets Use A Remapping Argument To Change The Nodes Name As:-

```
rosrun turtlesim turtlesim_node name:=my_turtle
```

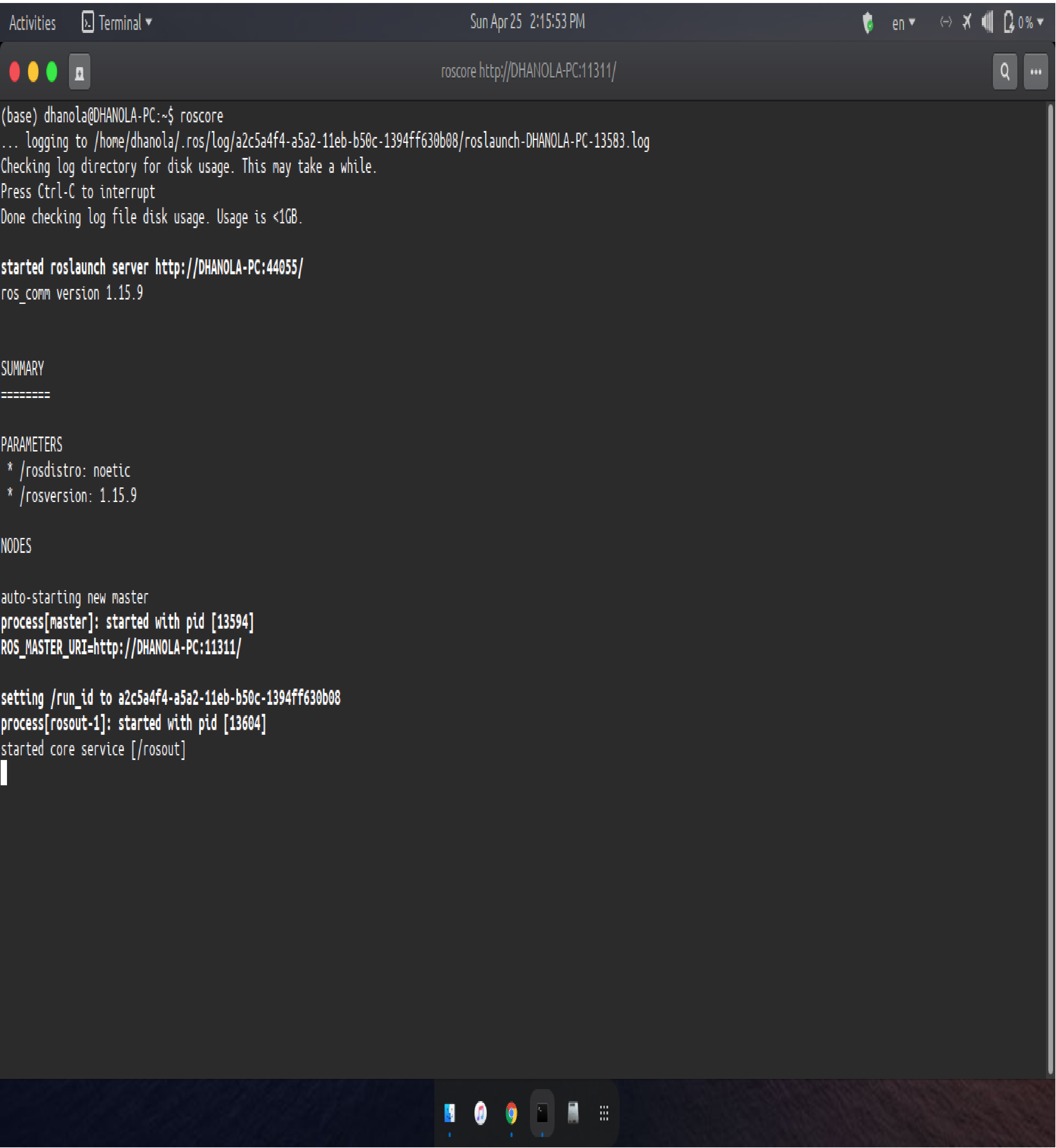


```
(base) dhanola@DHANOLA-PC:~$ rosrun turtlesim turtlesim_node name:=my_turtle
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[ INFO] [1619340568.503124138]: Starting turtlesim with node name /turtlesim
[ INFO] [1619340568.514451793]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

Understanding ROS Topics:

A) Let's Start By Making Sure That We Have Roscore Running, In A New Terminal:-

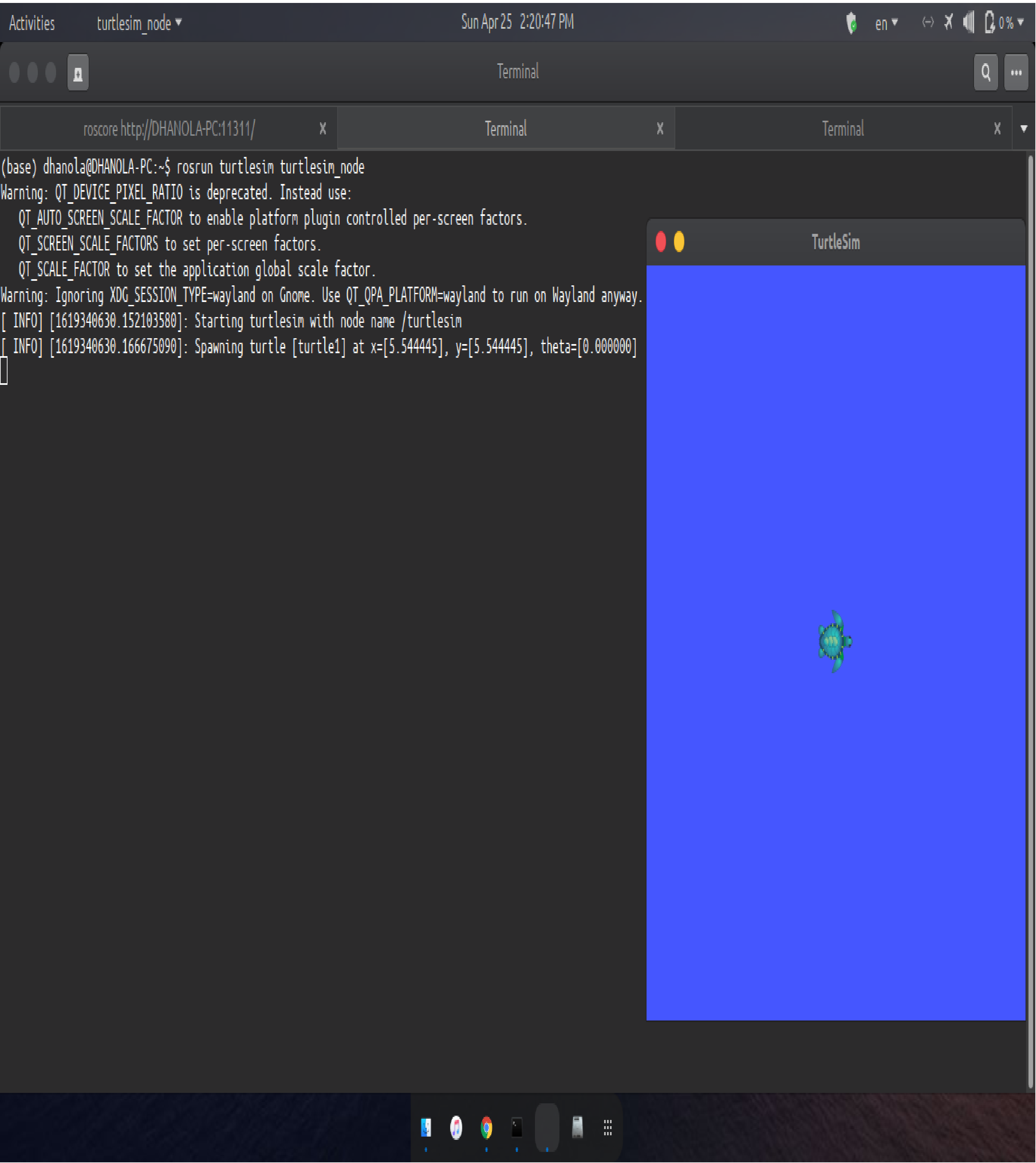
```
roscore
```



```
(base) dhanola@DHANOLA-PC:~$ roscore
... logging to /home/dhanola/.ros/log/a2c5a4f4-a5a2-11eb-b50c-1394ff630b08/roslaunch-DHANOLA-PC-13583.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://DHANOLA-PC:44055/
ros_comm version 1.15.9


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.9

NODES

auto-starting new master
process[master]: started with pid [13594]
ROS_MASTER_URI=http://DHANOLA-PC:11311/

setting /run_id to a2c5a4f4-a5a2-11eb-b50c-1394ff630b08
process[rosout-1]: started with pid [13604]
started core service [/rosout]
```

B) Run In A New Terminal:-

rosrun turtlesim turtlesim_node

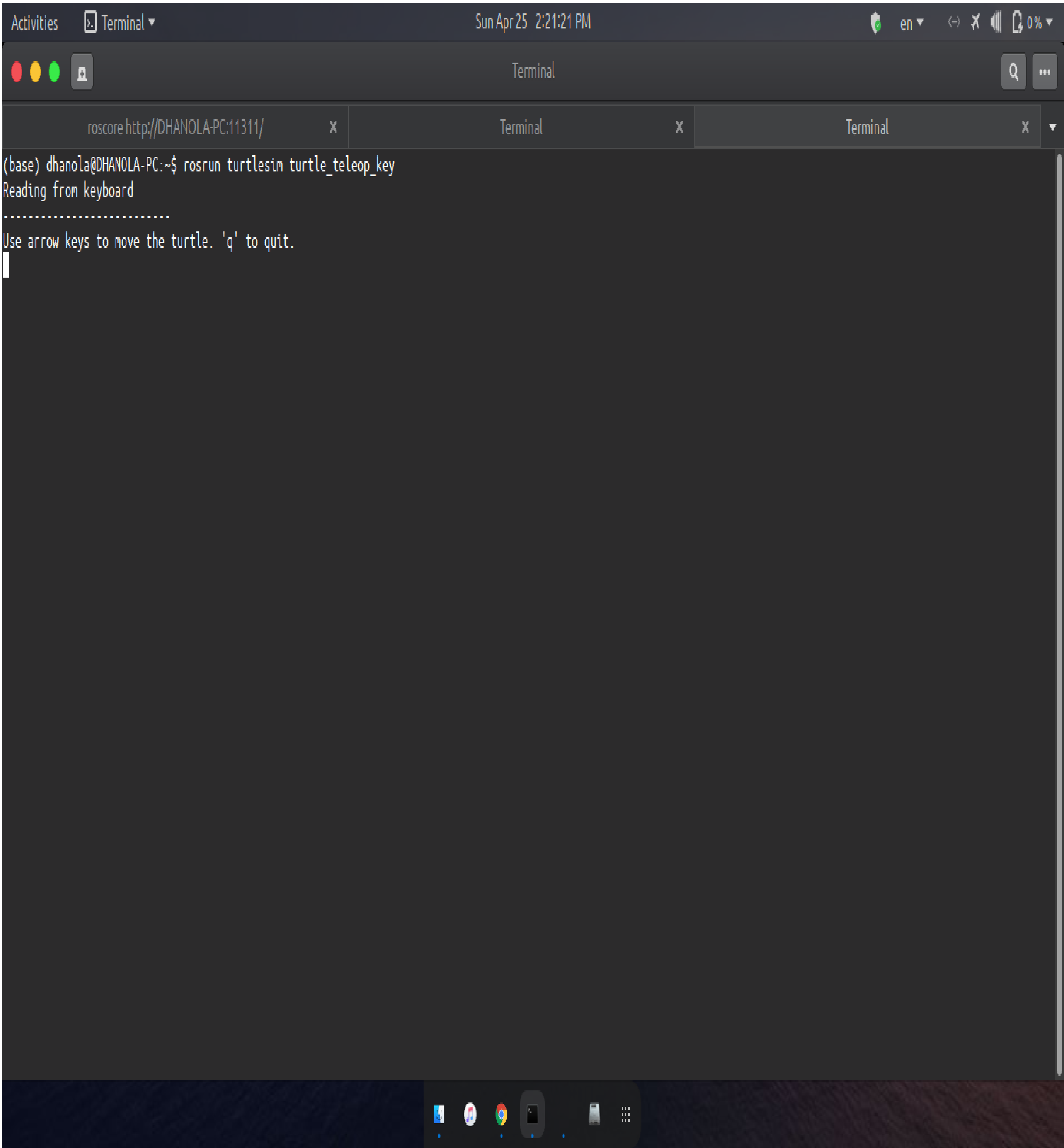


```
(base) dhanola@DHANOLA-PC:~$ rosrun turtlesim turtlesim_node
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[ INFO] [1619340630.152103580]: Starting turtlesim with node name /turtlesim
[ INFO] [1619340630.166675090]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

C) Turtle Keyboard Teleoperation:-

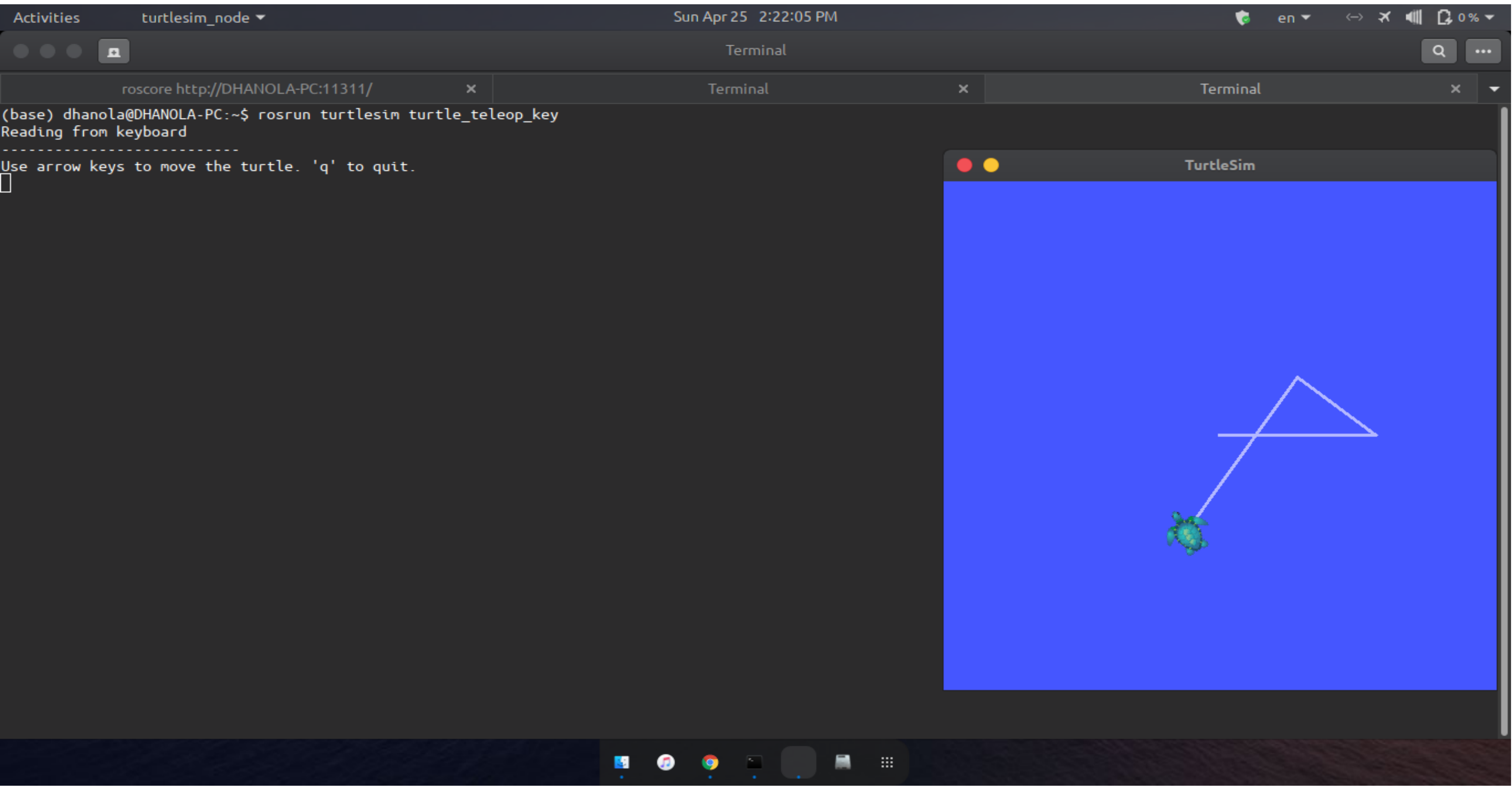
We Should Get The Following Outputs.

rosrun turtlesim turtle_teleop_key



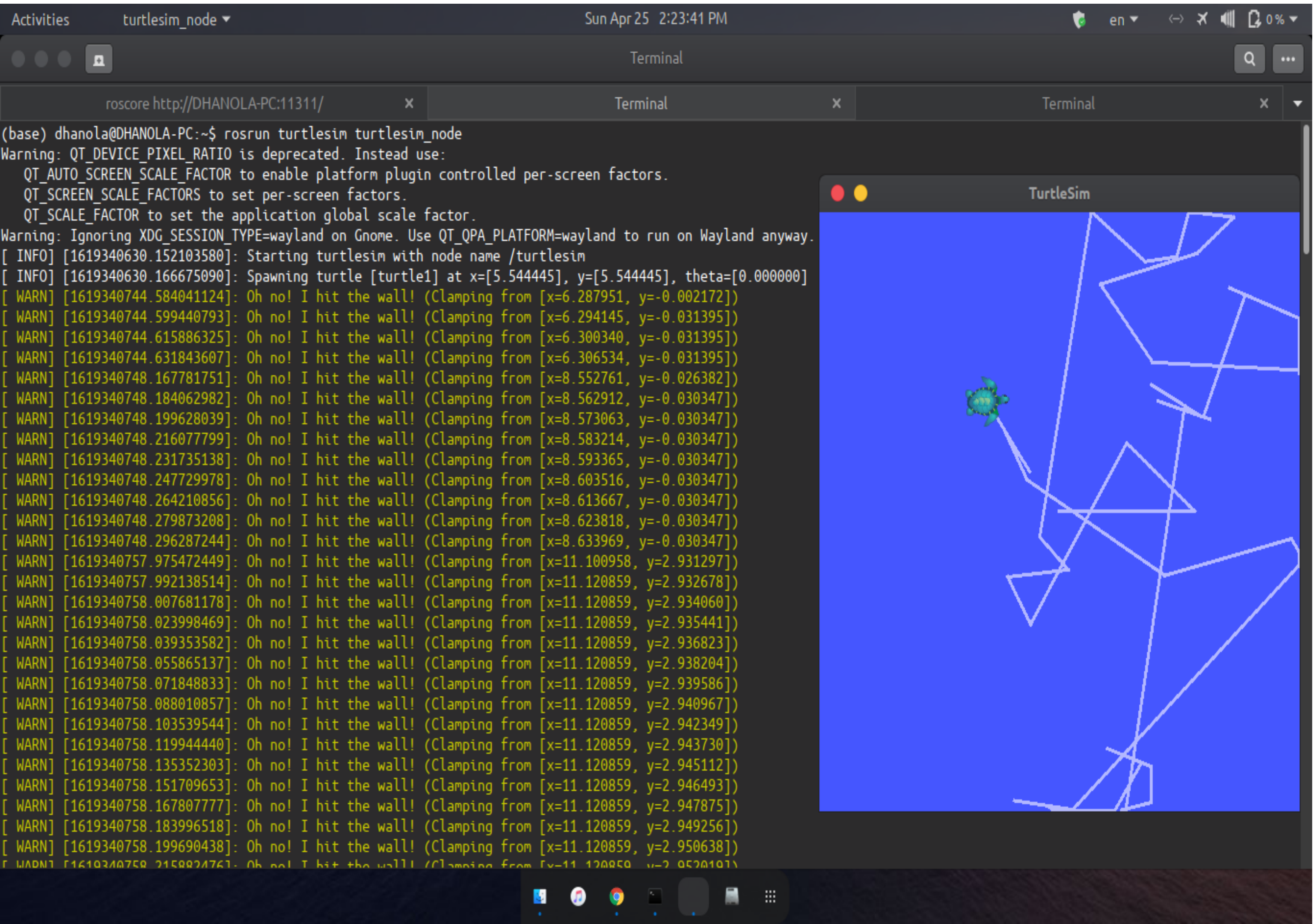
```
(base) dhanola@DHANOLA-PC:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle. 'q' to quit.
```

And,



```
(base) dhanola@DHANOLA-PC:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle. 'q' to quit.
```

When The Turtle Hits The Walls We Also See This Output:-

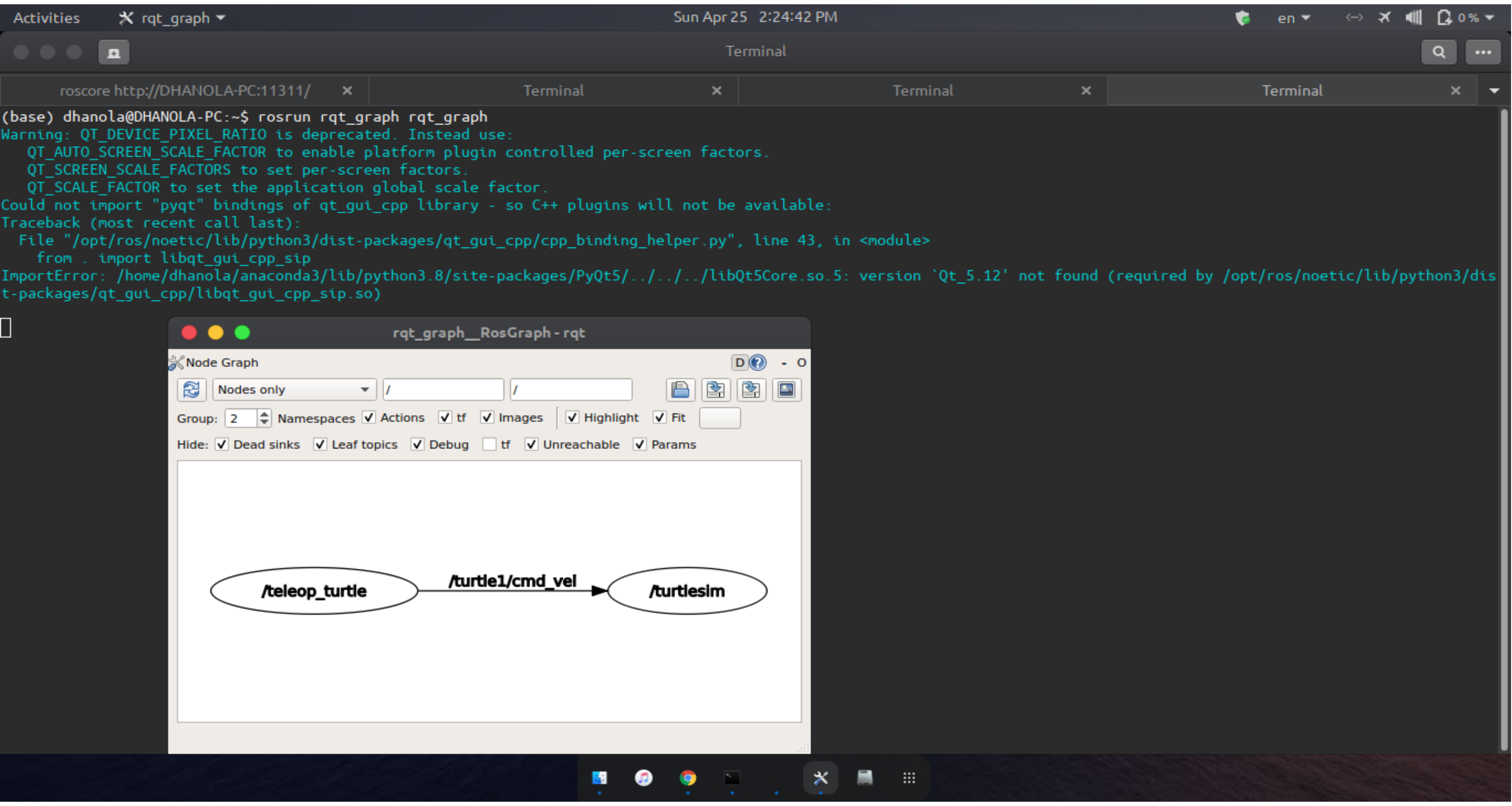


```
(base) dhanola@DHANOLA-PC:~$ rosrun turtlesim turtlesim_node
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[ INFO] [1619340630.152103580]: Starting turtlesim with node name /turtlesim
[ INFO] [1619340630.166675090]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[ WARN] [1619340744.584041124]: Oh no! I hit the wall! (Clamping from [x=6.287951, y=-0.002172])
[ WARN] [1619340744.599440793]: Oh no! I hit the wall! (Clamping from [x=6.294145, y=-0.031395])
[ WARN] [1619340744.615886325]: Oh no! I hit the wall! (Clamping from [x=6.300340, y=-0.031395])
[ WARN] [1619340744.631843607]: Oh no! I hit the wall! (Clamping from [x=6.306534, y=-0.031395])
[ WARN] [1619340748.167781751]: Oh no! I hit the wall! (Clamping from [x=8.552761, y=-0.026382])
[ WARN] [1619340748.184062982]: Oh no! I hit the wall! (Clamping from [x=8.562912, y=-0.030347])
[ WARN] [1619340748.199628039]: Oh no! I hit the wall! (Clamping from [x=8.573063, y=-0.030347])
[ WARN] [1619340748.216077799]: Oh no! I hit the wall! (Clamping from [x=8.583214, y=-0.030347])
[ WARN] [1619340748.231735138]: Oh no! I hit the wall! (Clamping from [x=8.593365, y=-0.030347])
[ WARN] [1619340748.247729978]: Oh no! I hit the wall! (Clamping from [x=8.603516, y=-0.030347])
[ WARN] [1619340748.264210856]: Oh no! I hit the wall! (Clamping from [x=8.613667, y=-0.030347])
[ WARN] [1619340748.279873208]: Oh no! I hit the wall! (Clamping from [x=8.623818, y=-0.030347])
[ WARN] [1619340748.296287244]: Oh no! I hit the wall! (Clamping from [x=8.633969, y=-0.030347])
[ WARN] [1619340757.975472449]: Oh no! I hit the wall! (Clamping from [x=11.100958, y=2.931297])
[ WARN] [1619340757.992138514]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.932678])
[ WARN] [1619340758.007681178]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.934060])
[ WARN] [1619340758.023998469]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.935441])
[ WARN] [1619340758.039353582]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.936823])
[ WARN] [1619340758.055865137]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.938204])
[ WARN] [1619340758.071848833]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.939586])
[ WARN] [1619340758.088010857]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.940967])
[ WARN] [1619340758.103539544]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.942349])
[ WARN] [1619340758.119944440]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.943730])
[ WARN] [1619340758.135352303]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.945112])
[ WARN] [1619340758.151709653]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.946493])
[ WARN] [1619340758.167807777]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.947875])
[ WARN] [1619340758.183996518]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.949256])
[ WARN] [1619340758.199690438]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.950638])
[ WARN] [1619340758.215882476]: Oh no! I hit the wall! (Clamping from [x=11.120859, y=2.952019])
```

D) ROS Topics:

In The New Terminal Run:-

```
rosrun rqt_graph rqt_graph
```



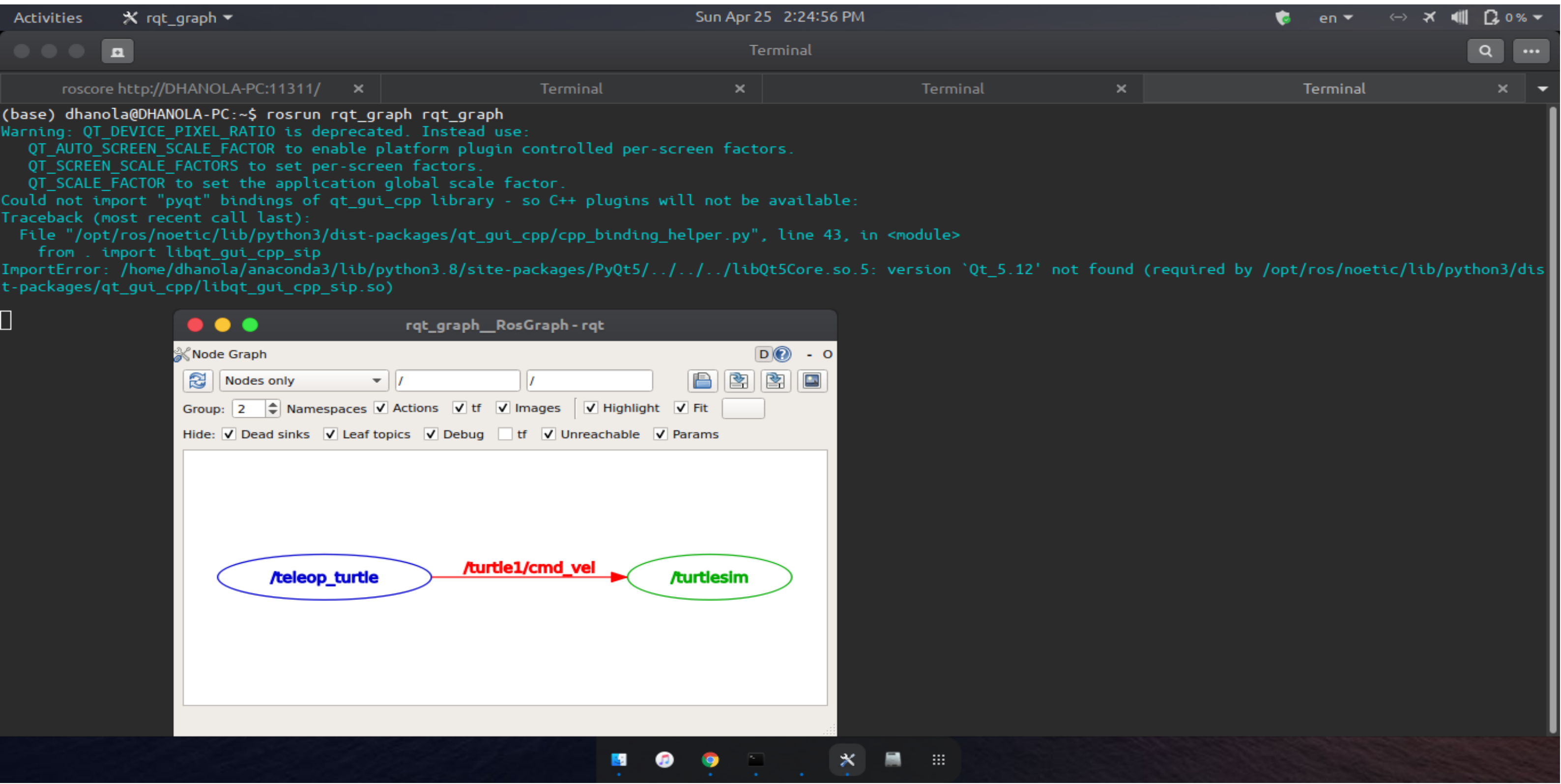
```
(base) dhanola@DHANOLA-PC:~$ rosrun rqt_graph rqt_graph
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
Could not import "pyqt" bindings of qt_gui_cpp library - so C++ plugins will not be available:
Traceback (most recent call last):
  File "/opt/ros/noetic/lib/python3/dist-packages/qt_gui_cpp/cpp_binding_helper.py", line 43, in <module>
    from . import libqt_gui_cpp_sip
ImportError: /home/dhanola/anaconda3/lib/python3.8/site-packages/PyQt5/../../../libQt5Core.so.5: version `Qt_5.12' not found (required by /opt/ros/noetic/lib/python3/dist-packages/qt_gui_cpp/libqt_gui_cpp_sip.so)
```

Nodes only | / | /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

If We Hover Our Mouse Over /Turtle1/Cmd_Vel It Will Highlight The ROS Nodes (Here Blue And Green) And Topics (Here Red):-



```
(base) dhanola@DHANOLA-PC:~$ rosrun rqt_graph rqt_graph
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
Could not import "pyqt" bindings of qt_gui_cpp library - so C++ plugins will not be available:
Traceback (most recent call last):
  File "/opt/ros/noetic/lib/python3/dist-packages/qt_gui_cpp/cpp_binding_helper.py", line 43, in <module>
    from . import libqt_gui_cpp_sip
ImportError: /home/dhanola/anaconda3/lib/python3.8/site-packages/PyQt5/../../../libQt5Core.so.5: version `Qt_5.12' not found (required by /opt/ros/noetic/lib/python3/dist-packages/qt_gui_cpp/libqt_gui_cpp_sip.so)
```

Nodes only | / | /

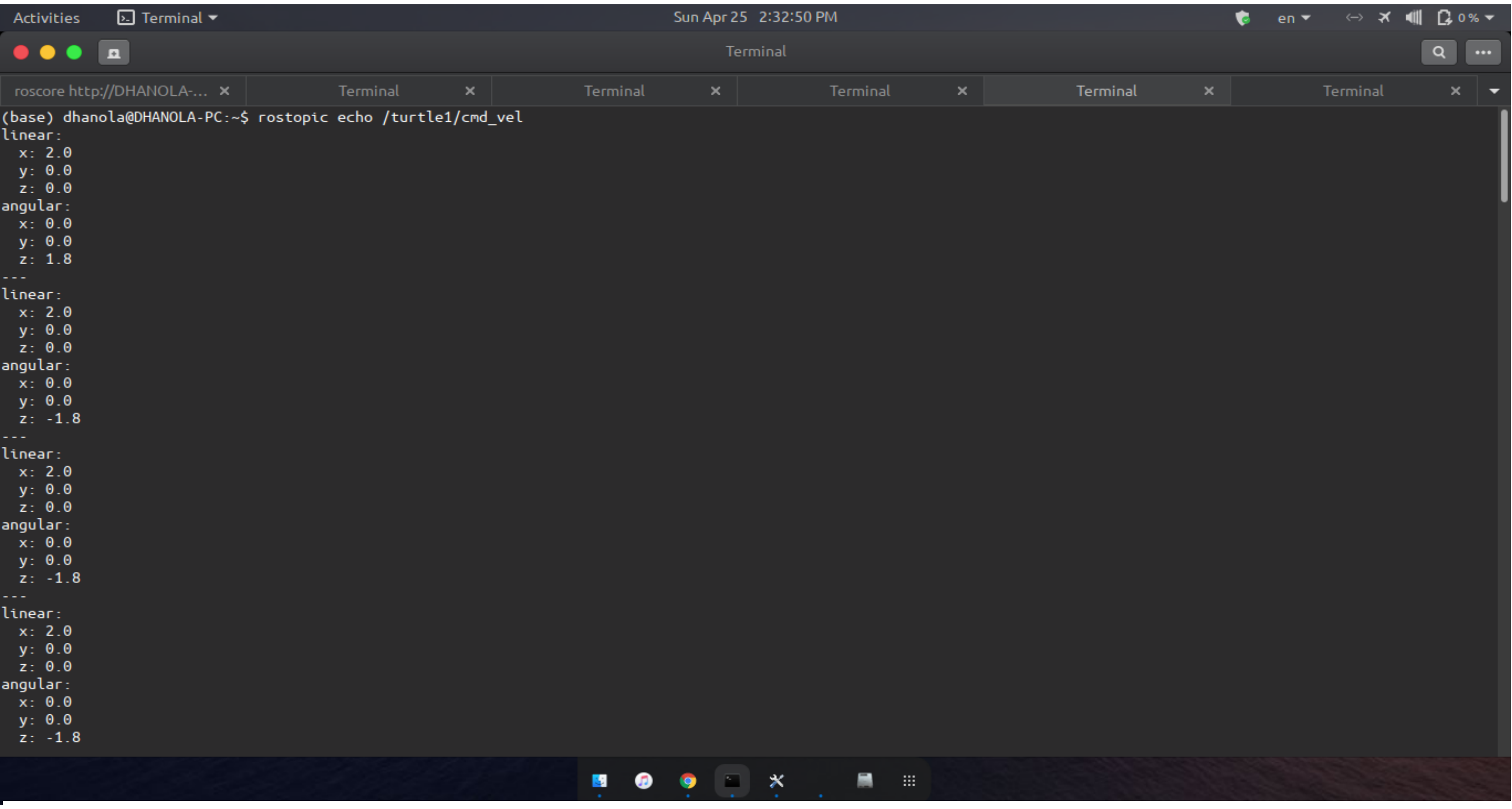
Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

We Use The Command:-

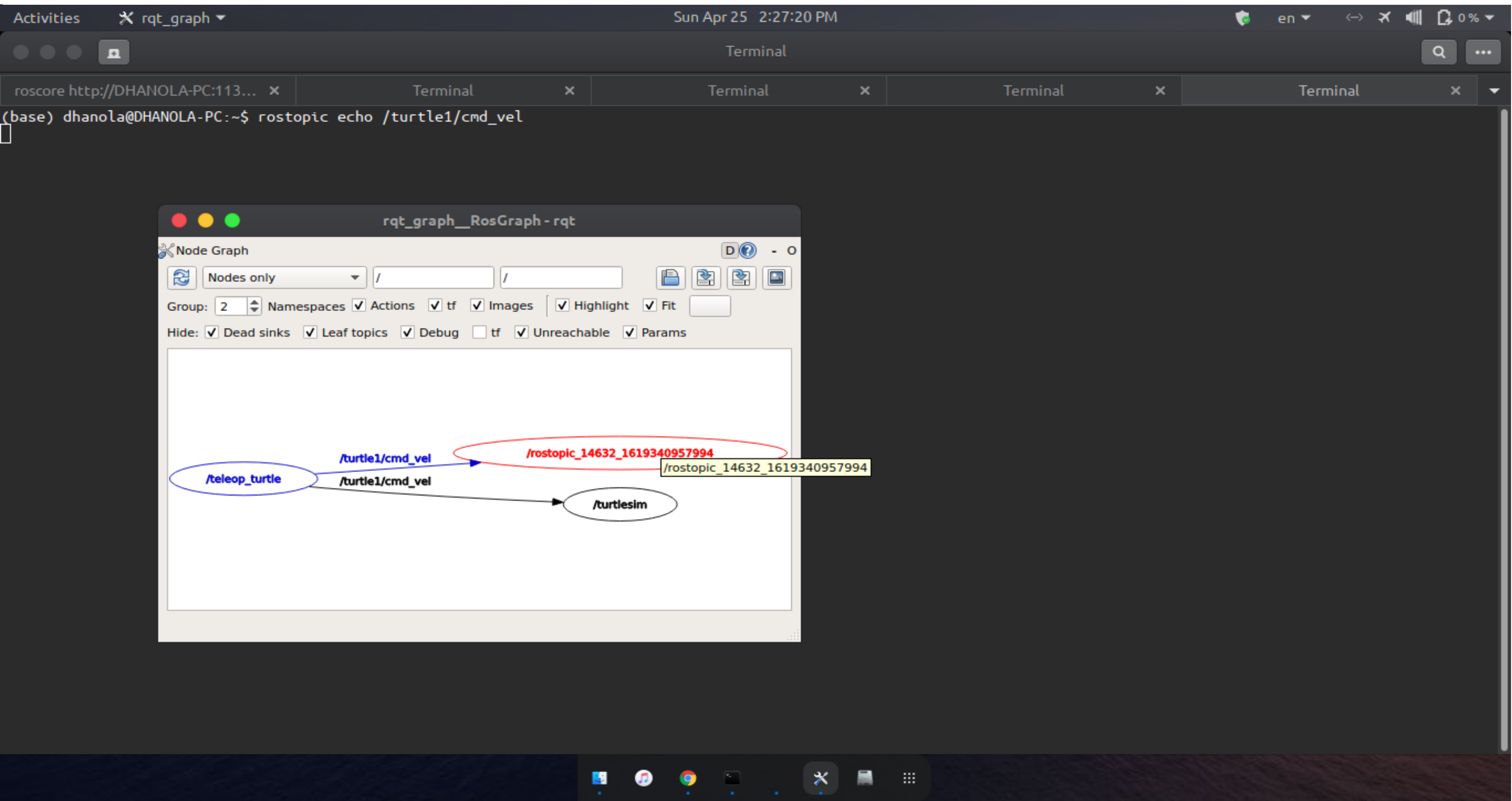
rostopic echo /turtle1/cmd_vel

And We Get The Data Being Published As Ros Topic:-



```
(base) dhanola@DHANOLA-PC:~$ rostopic echo /turtle1/cmd_vel
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.8
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -1.8
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -1.8
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -1.8
```

Now Let's Look At Rqt_Graph Again. Press The Refresh Button In The Upper-Left To Show The New Node. As You Can See Rostopic Echo, Shown Here In Red, Is Now Also Subscribed To The Turtle1/Cmd_Vel Topic :-



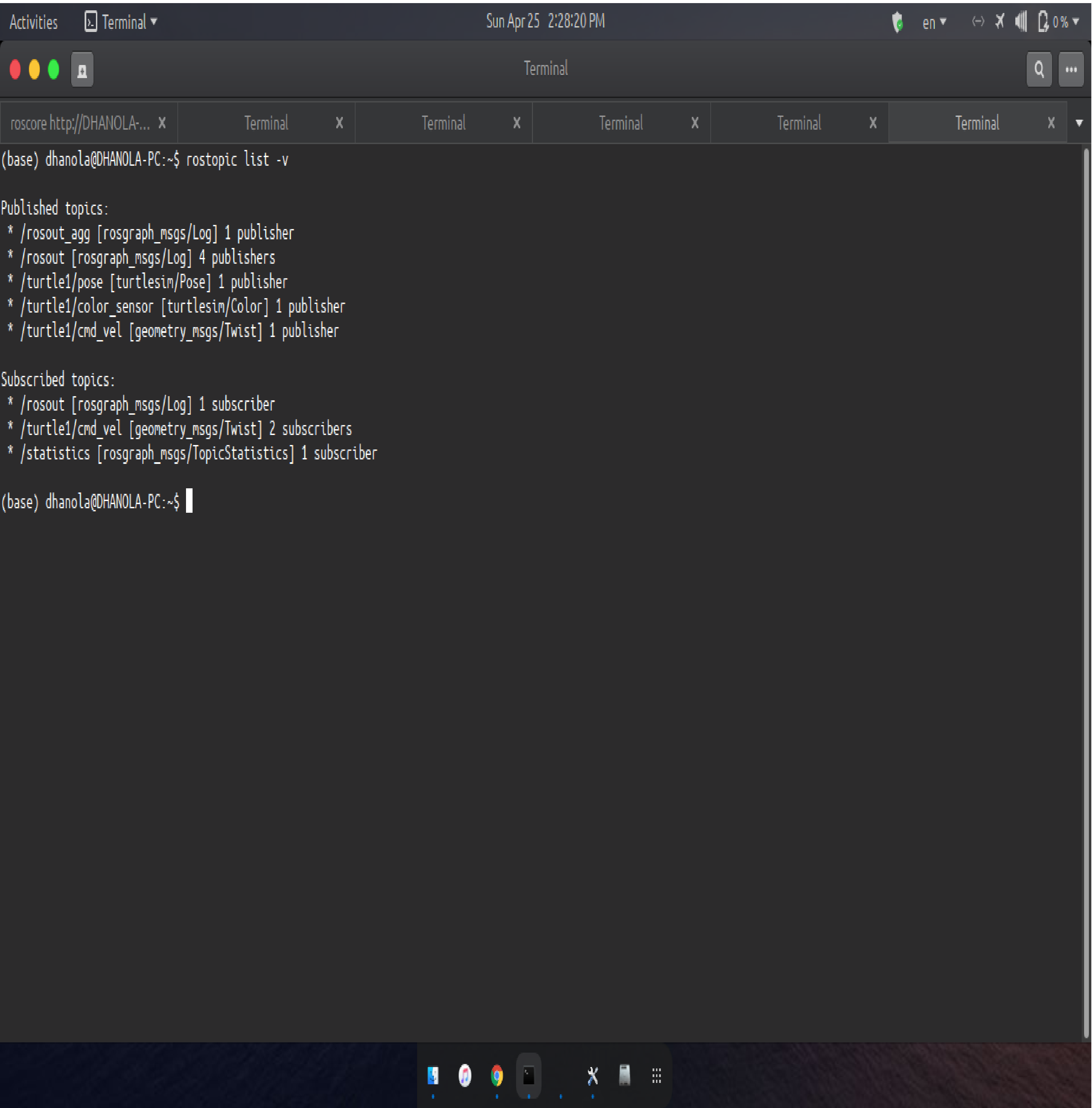
```
(base) dhanola@DHANOLA-PC:~$ rostopic echo /turtle1/cmd_vel
```

E) Rostopic List:

This Tell Us The Whole List Of Topics : The Published And The Subscribed.

Use The Command:-

rostopic list -v



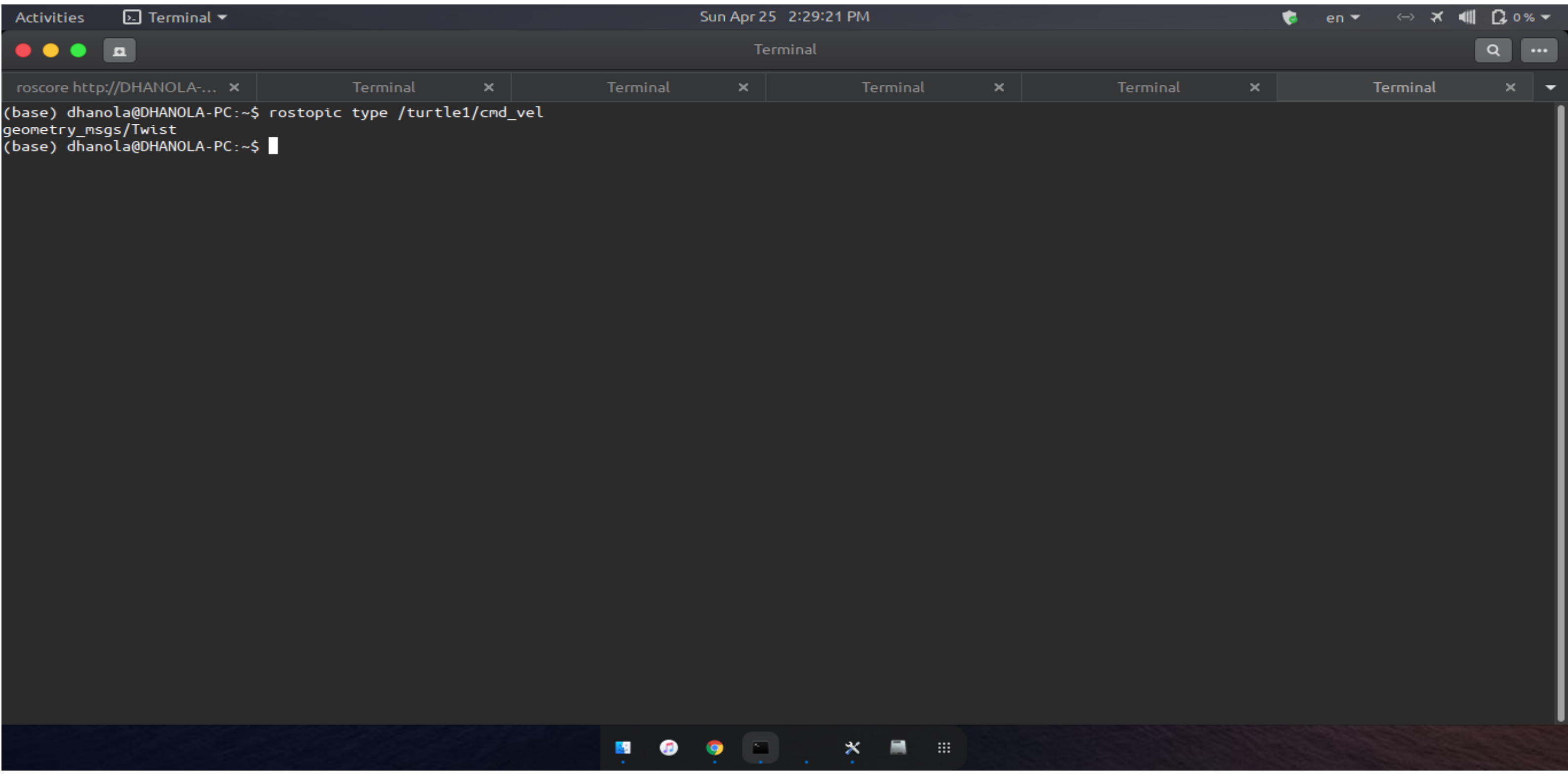
```
(base) dhanola@DHANOLA-PC:~$ rostopic list -v

Published topics:
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher
 * /rosout [rosgraph_msgs/Log] 4 publishers
 * /turtle1/pose [turtlesim/Pose] 1 publisher
 * /turtle1/color_sensor [turtlesim/Color] 1 publisher
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 publisher

Subscribed topics:
 * /rosout [rosgraph_msgs/Log] 1 subscriber
 * /turtle1/cmd_vel [geometry_msgs/Twist] 2 subscribers
 * /statistics [rosgraph_msgs/TopicStatistics] 1 subscriber

(base) dhanola@DHANOLA-PC:~$
```

F) ROS Messages:

Communication On Topics Happens By Sending Ros Messages Between Nodes. For The Publisher (Turtle_Teleop_Key) And Subscriber (Turtlesim_Node) To Communicate, The Publisher And Subscriber Must Send And Receive The Same Type Of Message.

Enter The Command:-

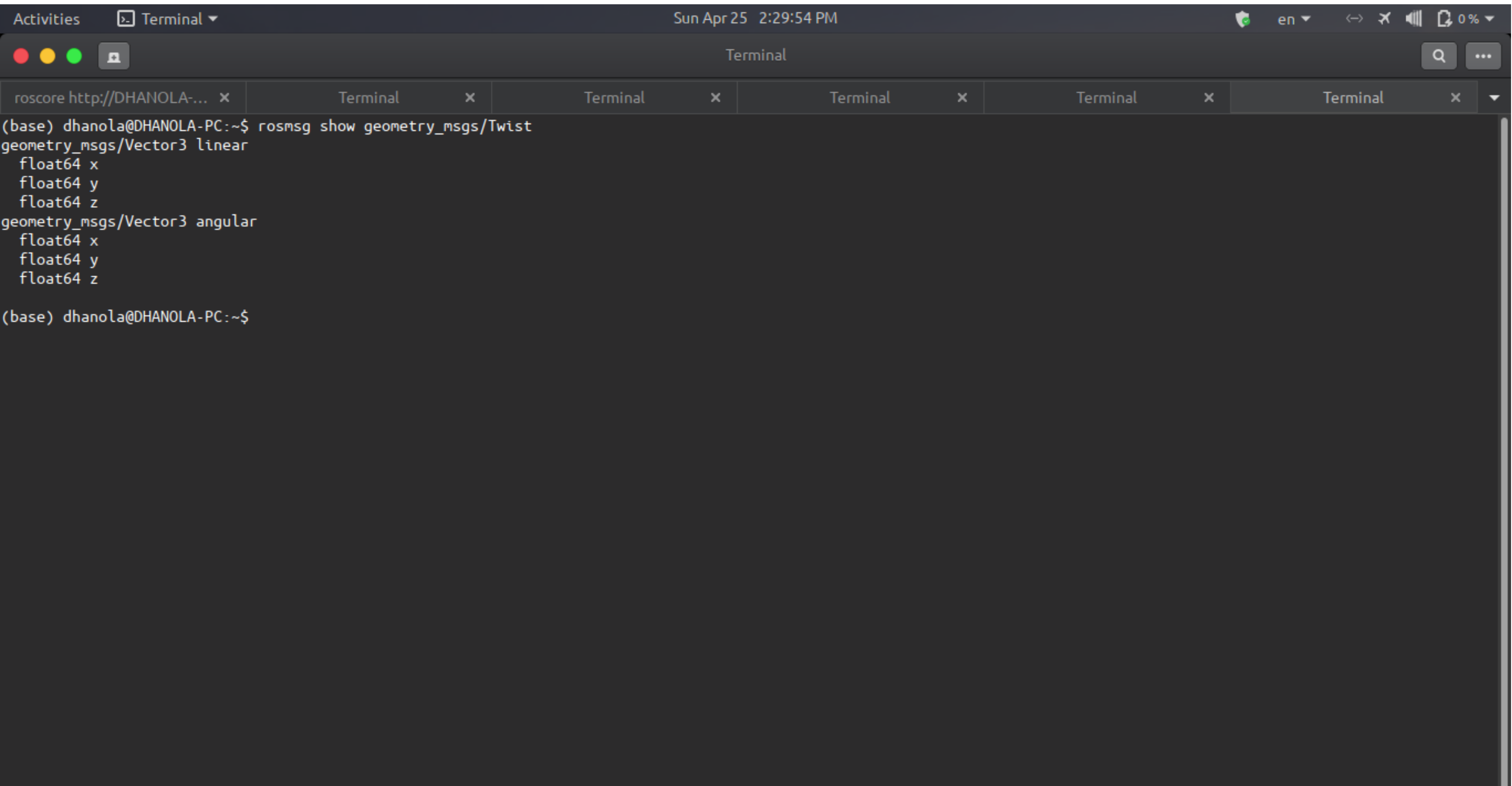
```
rostopic type /turtle1/cmd_vel
```



```
(base) dhanola@DHANOLA-PC:~$ rostopic type /turtle1/cmd_vel
geometry_msgs/Twist
(base) dhanola@DHANOLA-PC:~$ 
```

We Can Look At The Details Of The Message Using Rosmsg:

```
rostopic type /turtle1/cmd_vel
```

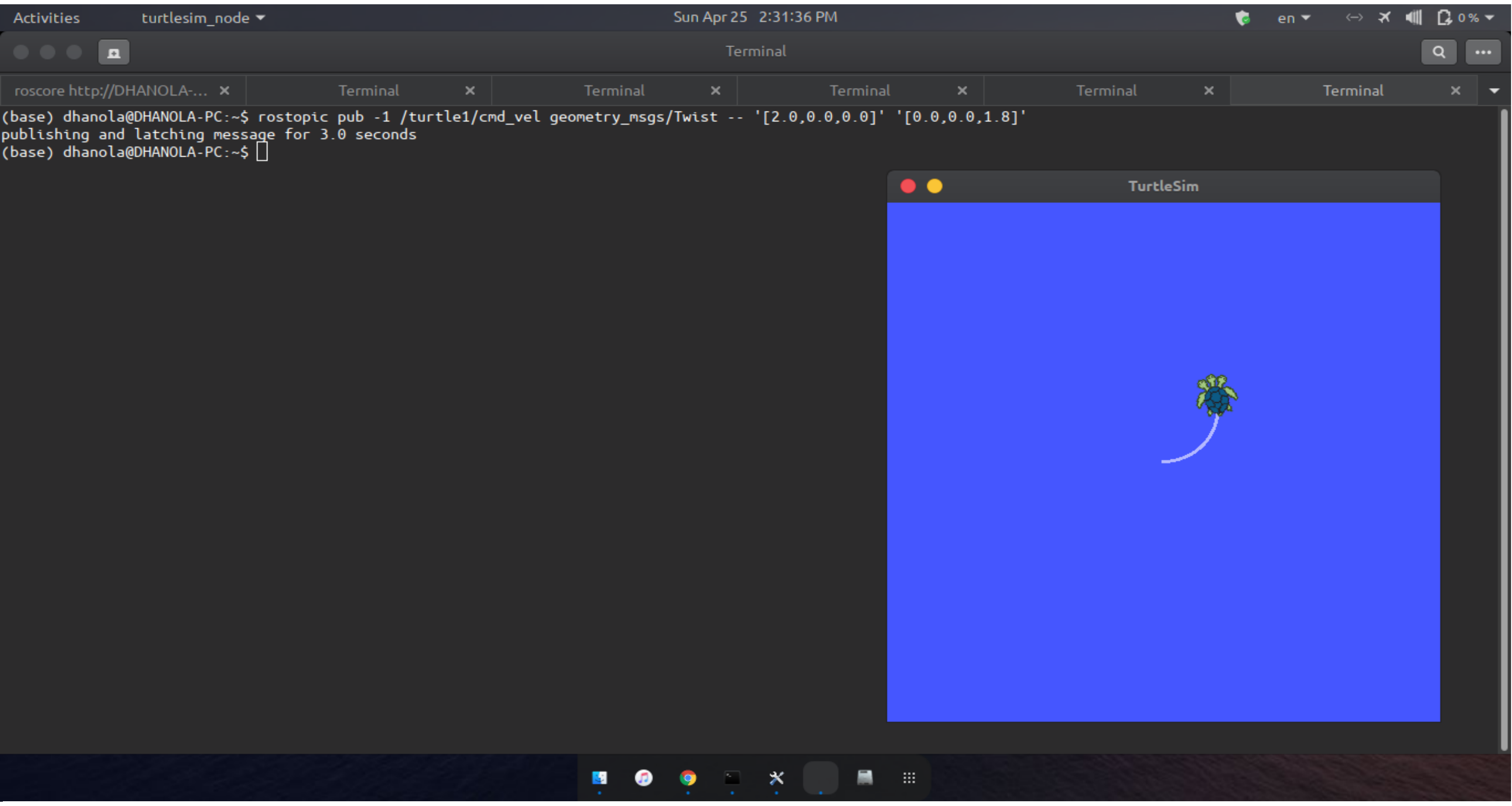


```
(base) dhanola@DHANOLA-PC:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z

(base) dhanola@DHANOLA-PC:~$
```

Also, Rostopic Pub Publishes Data On To A Topic Currently Advertised.

For Example Entering, This Command Will Send A Single Message To Turtlesim Telling It To Move With A Linear Velocity Of 2.0, And An Angular Velocity Of 1.8:-

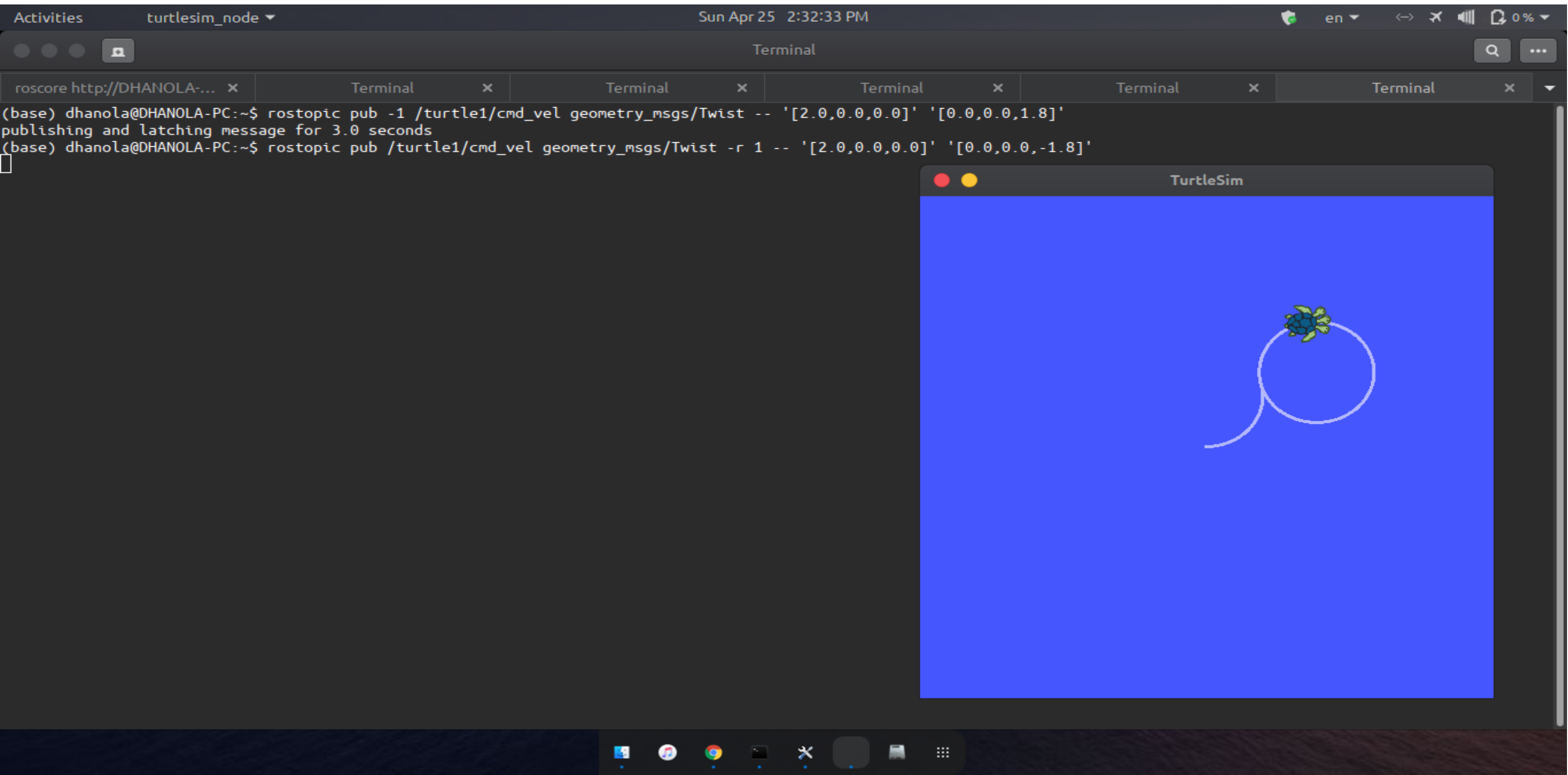
rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'



```
(base) dhanola@DHANOLA-PC:~$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0,0.0,0.0]' '[0.0,0.0,1.8]'
publishing and latching message for 3.0 seconds
(base) dhanola@DHANOLA-PC:~$ 
```

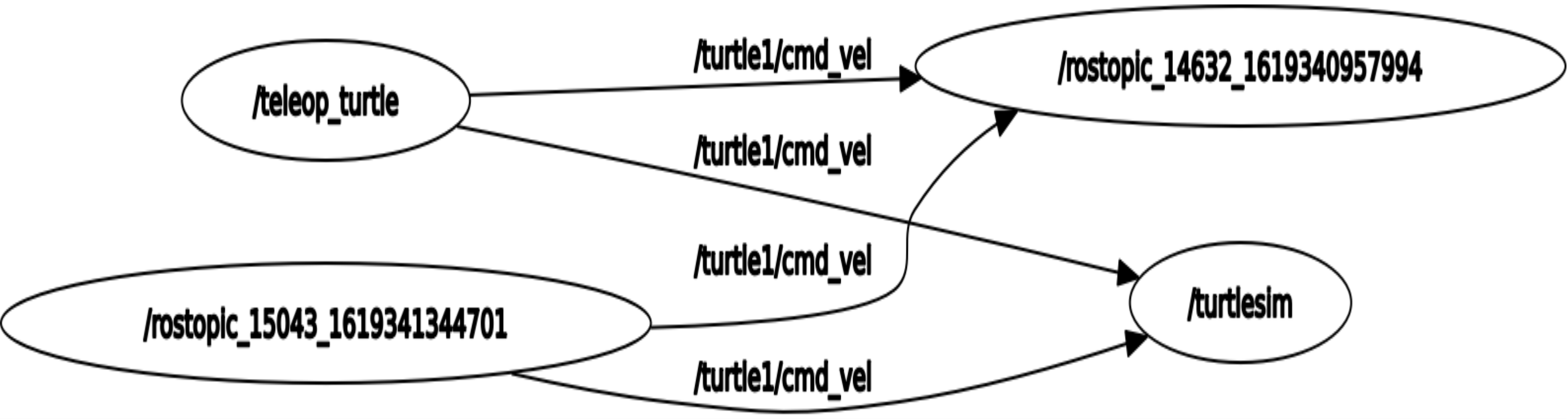
We Can Publish A Steady Stream Of Commands Using Rostopic Pub-R :-

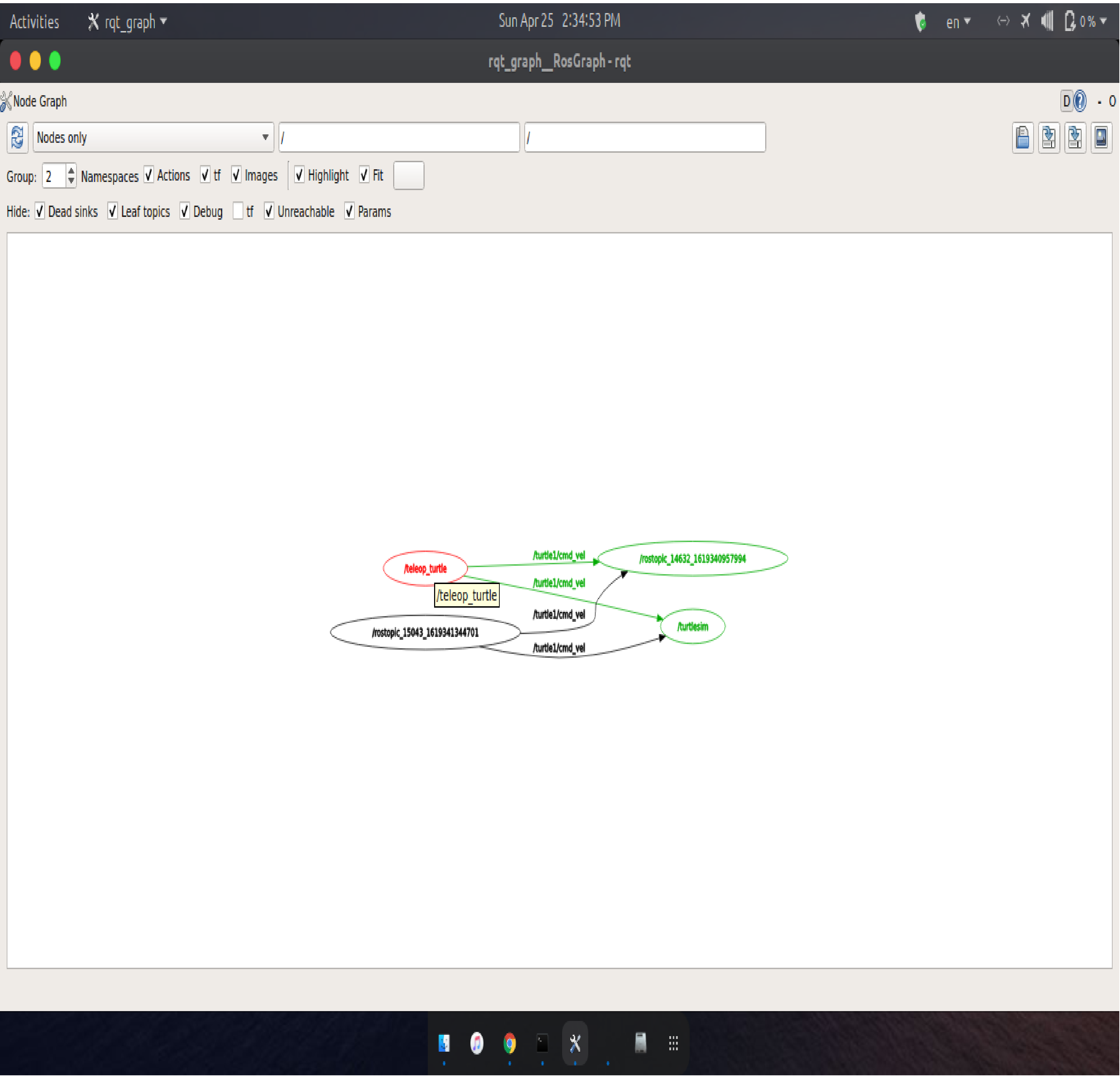
rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, -1.8]



```
(base) dhanola@DHANOLA-PC:~$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0,0.0,0.0]' '[0.0,0.0,1.8]'
publishing and latching message for 3.0 seconds
(base) dhanola@DHANOLA-PC:~$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0,0.0,0.0]' '[0.0,0.0,-1.8]'
```

We Can Also Look At What Is Happening In Rqt_Graph. Press The Refresh Button In The Upper-Left. The Rostopic Pub Node (Here In Red) Is Communicating With The Rostopic Echo Node (Here In Green):-





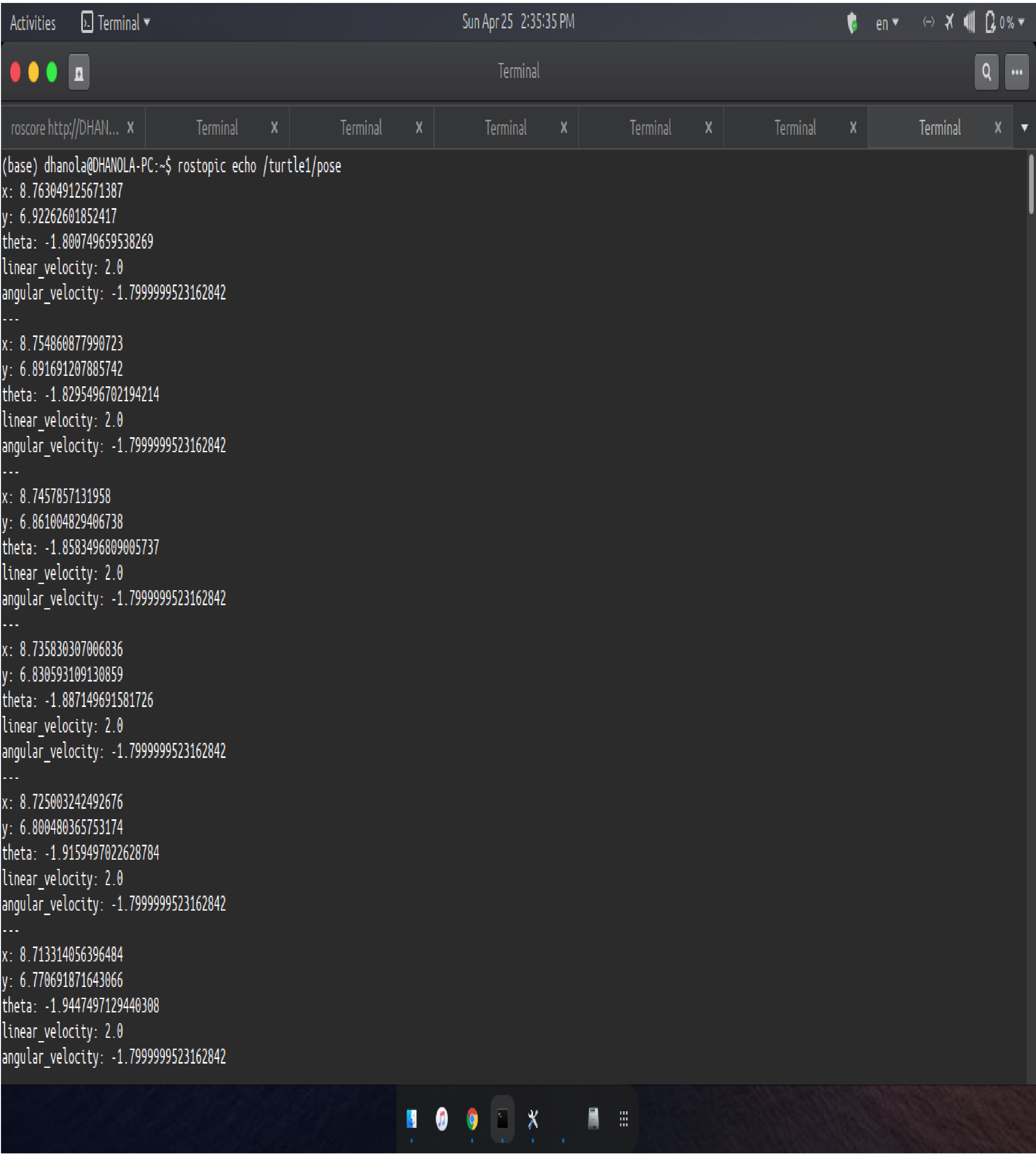
Nodes only ▾ | / | /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug ☐ tf ☑ Unreachable ☑ Params

Now We Can Use The Command To Echo The Details As:-

rostopic echo /turtle1/pose

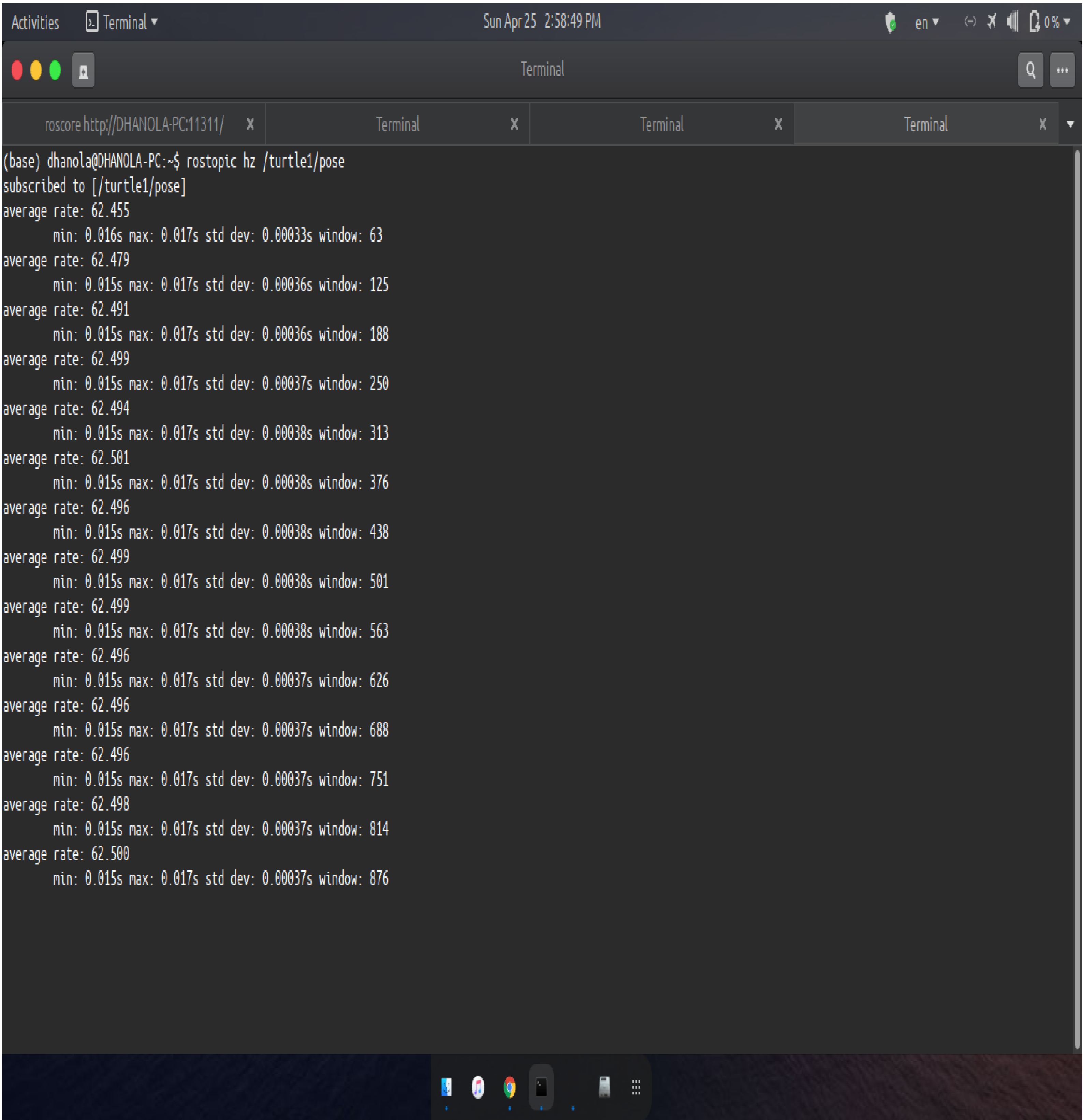


```
(base) dhanola@DHANOLA-PC:~$ rostopic echo /turtle1/pose
x: 8.763049125671387
y: 6.92262601852417
theta: -1.800749659538269
linear_velocity: 2.0
angular_velocity: -1.7999999523162842
---
x: 8.754860877990723
y: 6.891691207885742
theta: -1.8295496702194214
linear_velocity: 2.0
angular_velocity: -1.7999999523162842
---
x: 8.7457857131958
y: 6.861004829406738
theta: -1.8583496809005737
linear_velocity: 2.0
angular_velocity: -1.7999999523162842
---
x: 8.735830307006836
y: 6.830593109130859
theta: -1.887149691581726
linear_velocity: 2.0
angular_velocity: -1.7999999523162842
---
x: 8.725003242492676
y: 6.800480365753174
theta: -1.9159497022628784
linear_velocity: 2.0
angular_velocity: -1.7999999523162842
---
x: 8.713314056396484
y: 6.770691871643066
theta: -1.9447497129440308
linear_velocity: 2.0
angular_velocity: -1.7999999523162842
```

We Can Also See How Fast The Turtlesim_Node Is Publishing /Turtle1/Pose:

Use The Command:-

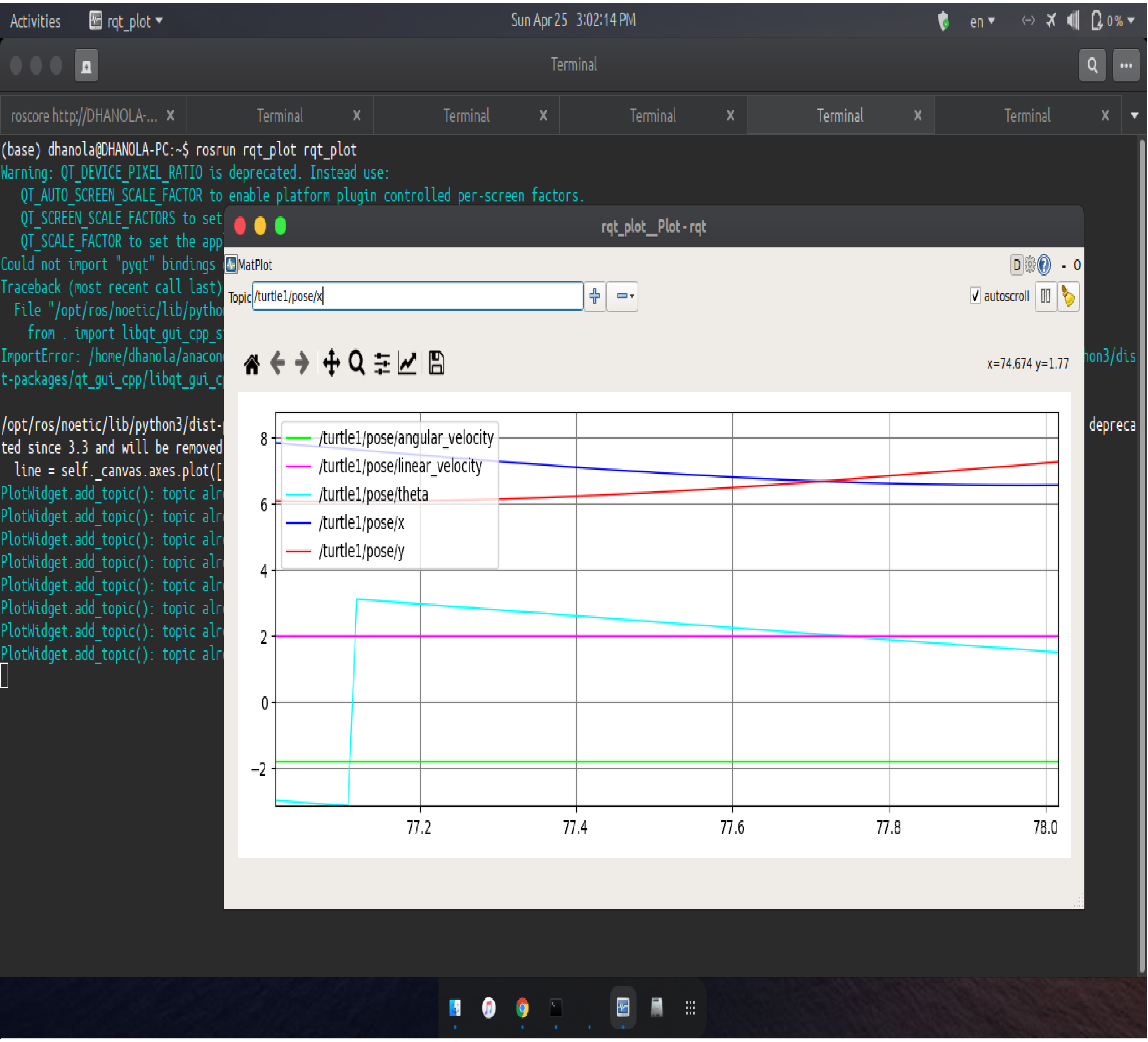
```
rostopic hz /turtle1/pose
```



```
(base) dhanola@DHANOLA-PC:~$ rostopic hz /turtle1/pose
subscribed to [/turtle1/pose]
average rate: 62.455
        min: 0.016s max: 0.017s std dev: 0.00033s window: 63
average rate: 62.479
        min: 0.015s max: 0.017s std dev: 0.00036s window: 125
average rate: 62.491
        min: 0.015s max: 0.017s std dev: 0.00036s window: 188
average rate: 62.499
        min: 0.015s max: 0.017s std dev: 0.00037s window: 250
average rate: 62.494
        min: 0.015s max: 0.017s std dev: 0.00038s window: 313
average rate: 62.501
        min: 0.015s max: 0.017s std dev: 0.00038s window: 376
average rate: 62.496
        min: 0.015s max: 0.017s std dev: 0.00038s window: 438
average rate: 62.499
        min: 0.015s max: 0.017s std dev: 0.00038s window: 501
average rate: 62.499
        min: 0.015s max: 0.017s std dev: 0.00038s window: 563
average rate: 62.496
        min: 0.015s max: 0.017s std dev: 0.00037s window: 626
average rate: 62.496
        min: 0.015s max: 0.017s std dev: 0.00037s window: 688
average rate: 62.496
        min: 0.015s max: 0.017s std dev: 0.00037s window: 751
average rate: 62.498
        min: 0.015s max: 0.017s std dev: 0.00037s window: 814
average rate: 62.500
        min: 0.015s max: 0.017s std dev: 0.00037s window: 876
```

We Can Use The Rqt_Plot To Display A Scrolling Time Plot Of The Data Published On Topics As:

rosrun rqt_plot rqt_plot



```
(base) dhanola@DHANOLA-PC:~$ rosrun rqt_plot rqt_plot
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set
   QT_SCALE_FACTOR to set the app
Could not import "pyqt" bindings
Traceback (most recent call last)
  File "/opt/ros/noetic/lib/pytho
    from . import libqt_gui_cpp_s
ImportError: /home/dhanola/anacon                                hon3/dis
t-packages/qt_gui_cpp/libqt_gui_c

/opt/ros/noetic/lib/python3/dist-                                depreca
ted since 3.3 and will be removed
  line = self._canvas.axes.plot([
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
PlotWidget.add_topic(): topic alr
```

Topic: /turtle1/pose/x

☑ autoscroll

THANK YOU

UPES
UNIVERSITY WITH A PURPOSE