Robot Operating System (ROS)

organized by

BMSCE IEEE PES and Sensors Council

May 6th and 7th 2024

Mukil Saravanan

Natish Murugesh

Kalaivanan K

Indian Institute of Science, Bangalore

Objectives of Workshop

- To understand ROS & its significance.
- To provide a hands-on experience with how ROS concepts work together.
- To explore advanced ROS tools and concepts such as SLAM and Navigation in simulation.
- To glimpse state-of-the-art research activities and future trends in robotics with ROS.

Outcomes of Workshop

- Effectively comprehend and create ROS packages
- Establish various communication methods between ROS nodes with Python3
- Deploy and navigate a simulated wheeled mobile robot through complex environments
- Leverage powerful ROS tools like Gazebo and Rviz.
- Explore cutting-edge research areas and discover career possibilities with ROS.

Agenda – Day 1

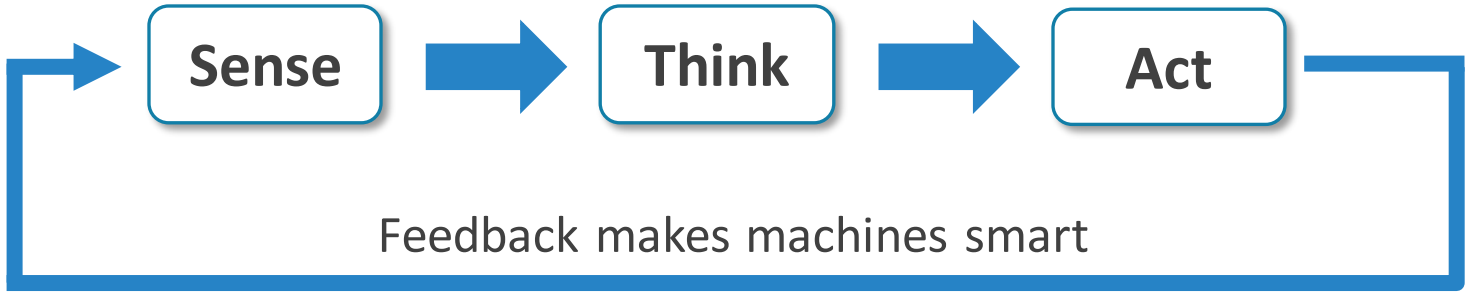
- ROS & its importance in Robotics
- Installation of ROS Noetic and a brief overview of Linux commands
- Creating and building the custom ROS workspace
- Mastering building blocks of ROS - nodes, topics, messages, services, parameters
- Delving into ROS concepts through hands-on practice with Turtlesim

What is a Robot?

What is a Robot?

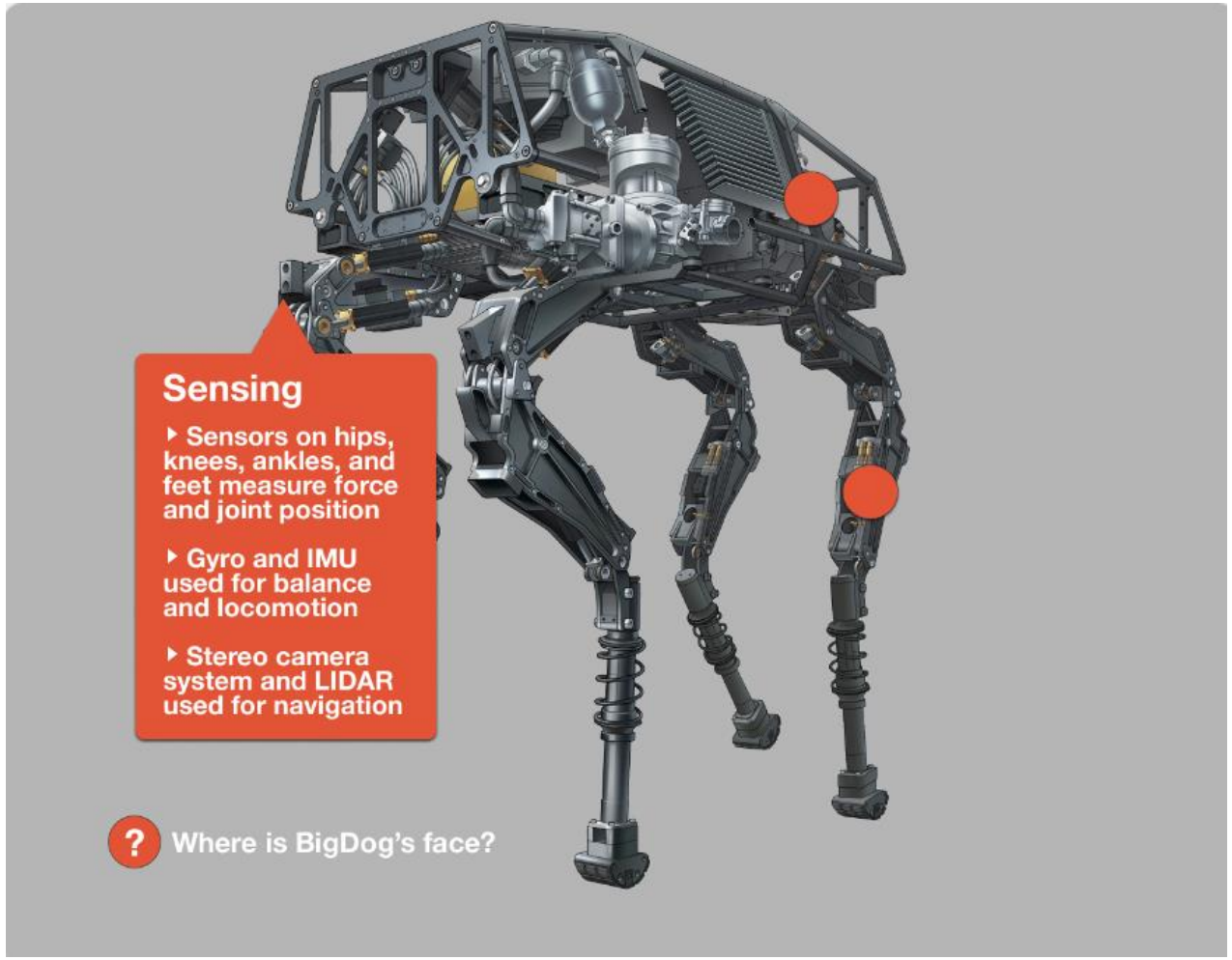
"A robot is an autonomous machine capable of sensing its environments, carrying out computations to make decisions and performing actions in the real world"[1]

- Rodney Brooks, Roomba creator



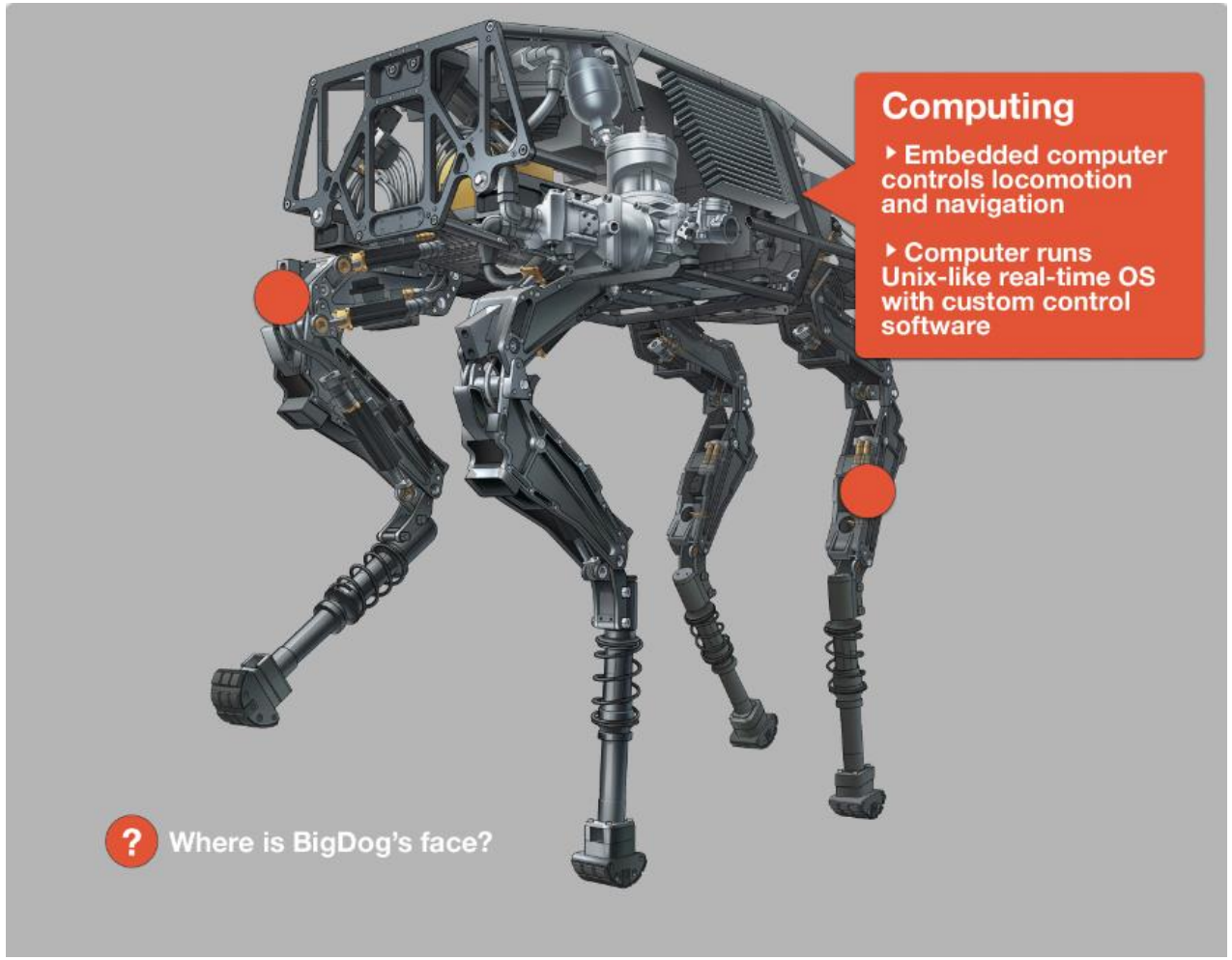
[1] What Is a Robot? - ROBOTS: Your Guide to the World of Robotics (ieee.org)

Sense – Think – Act



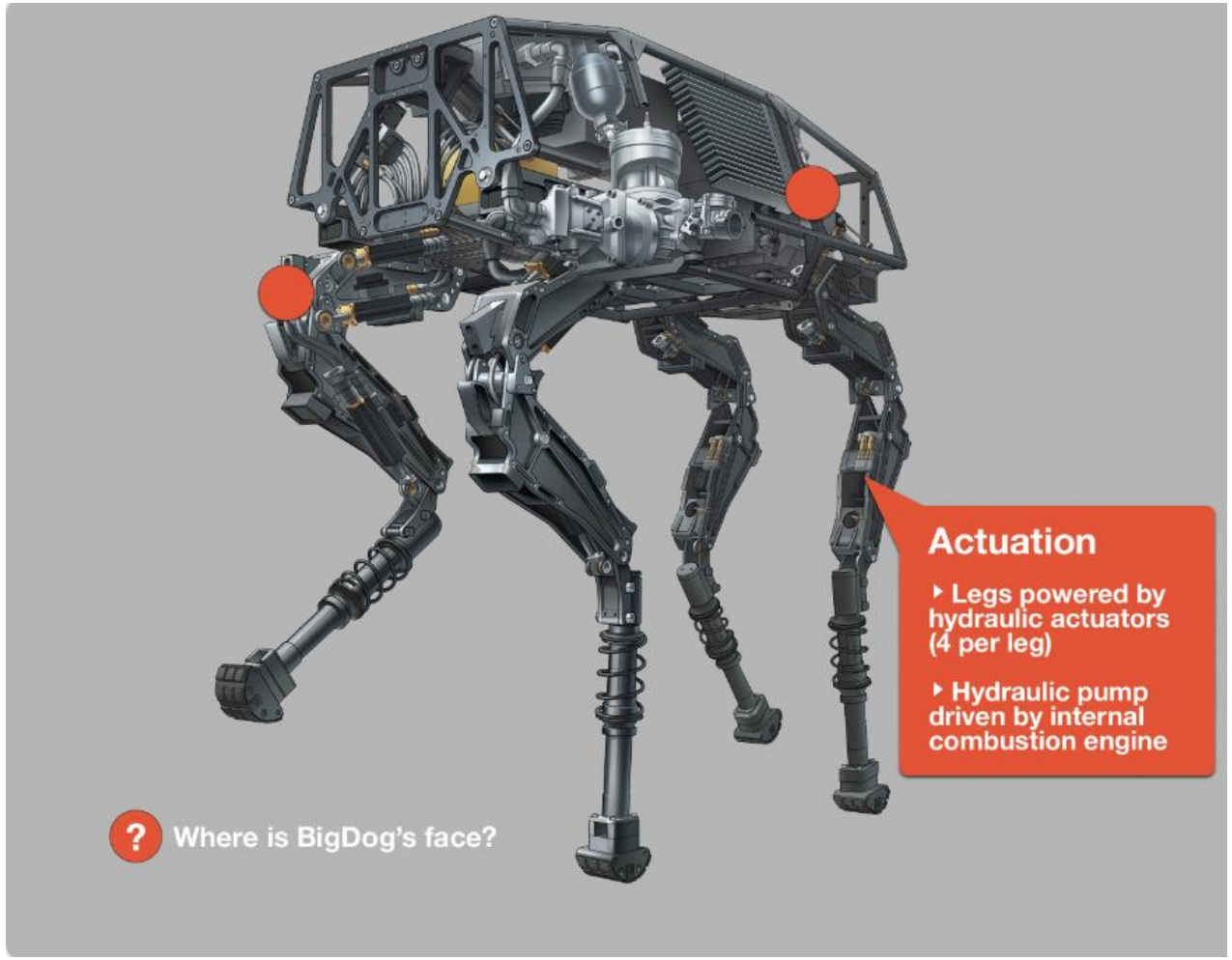
Credits: Big Dog - Boston Dynamics

Sense – Think – Act



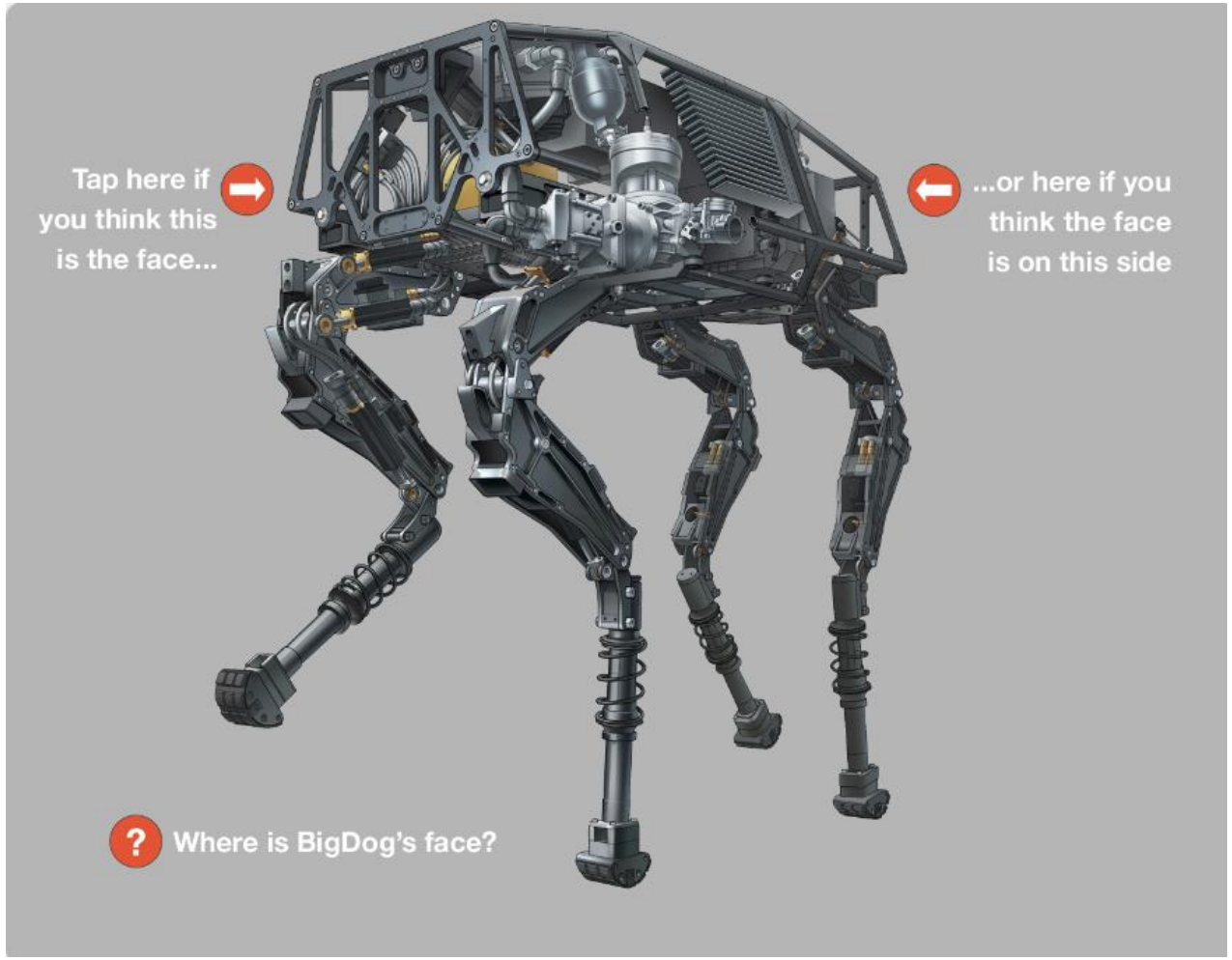
Credits: Big Dog - Boston Dynamics

Sense – Think – Act



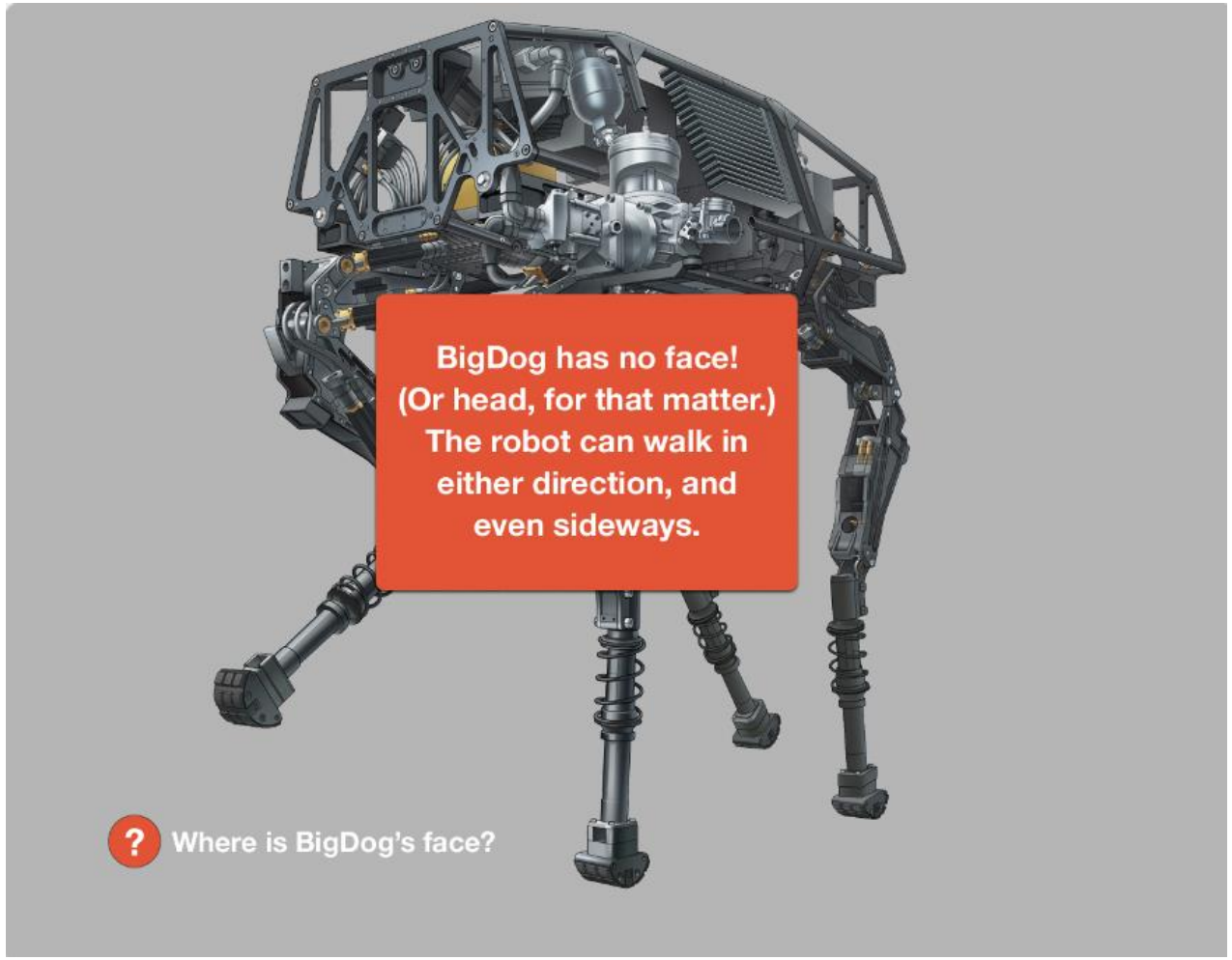
Credits: Big Dog - Boston Dynamics

Sense – Think – Act



Credits: Big Dog - Boston Dynamics

Sense – Think – Act



Credits: Big Dog - Boston Dynamics

Why do we need ROS in robotics?

Credits: Comic commissioned at Willow Garage, from Jorge Cham, to illustrate the wasted time in robotics R&D

What is ROS?

What is ROS?

Robot Operating System (ROS) is a set of software libraries and tools that helps to build robot applications [2]

[2] https://www.ros.org/

Is it an operating system?

Similarity – low –level device control, message passing between processes, package management

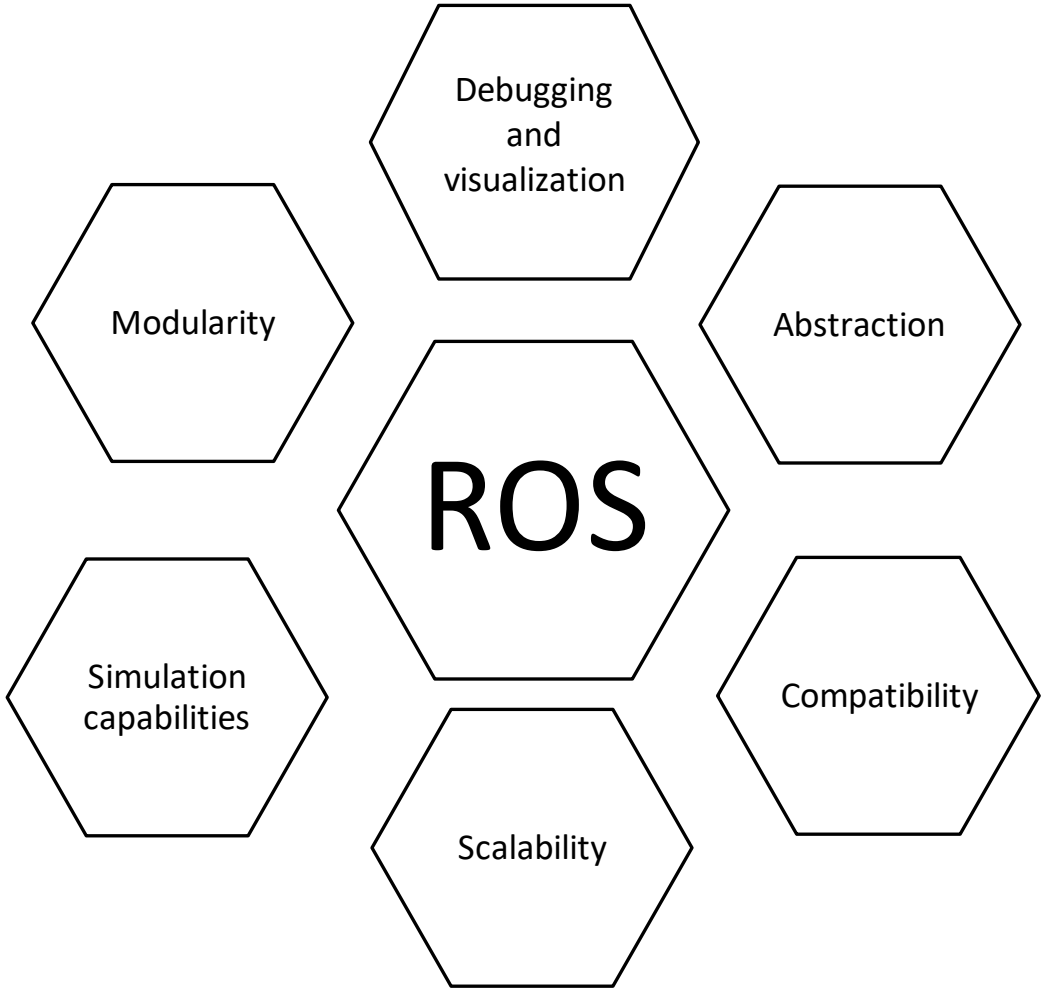
Middleware – Acts as a toolbox to build robots

OS dependency – Memory management, process scheduling, hardware interaction

Framework not OS

ROS
Robot Operating System

Key Features



BASIC COMMANDS IN UBUNTU

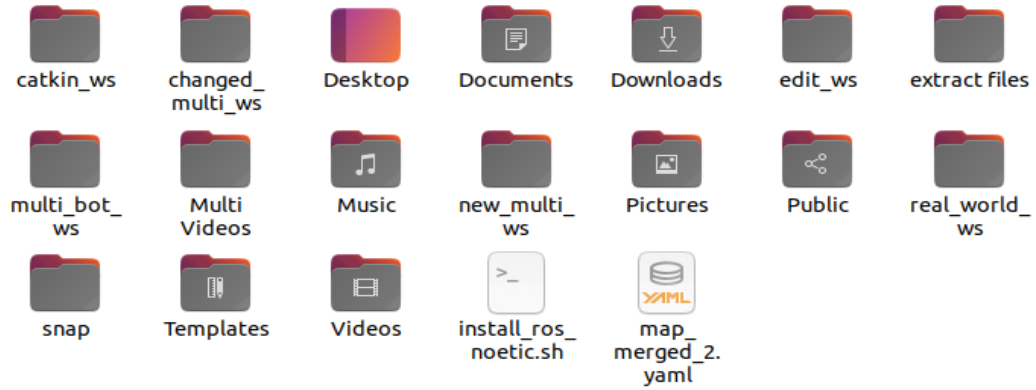
BASIC COMMANDS IN UBUNTU

1. ls

The ls command is used to list files and directories in the current working directory.

```
ls
```

```
natish@natish-ril: ~ 80x24
natish@natish-ril:~$ ls
 catkin_ws           edit_ws                'Multi Videos'   real_world_ws
 changed_multi_ws    'extract files'         Music           snap
 Desktop             install_ros_noetic.sh   new_multi_ws    Templates
 Documents           map_merged_2.yaml       Pictures        Videos
 Downloads           multi_bot_ws            Public
natish@natish-ril:~$
```

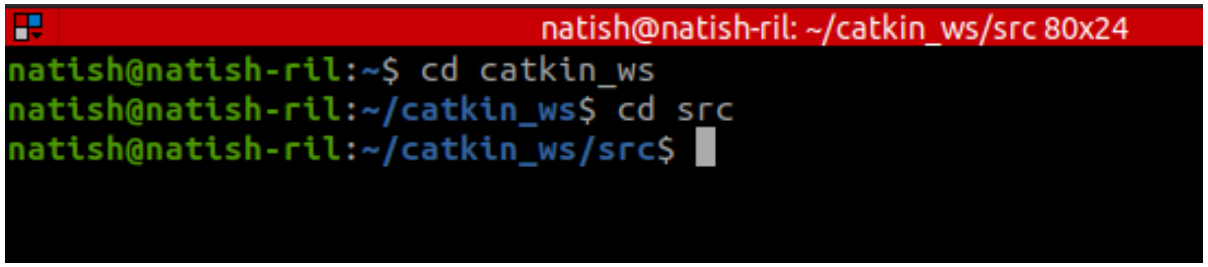


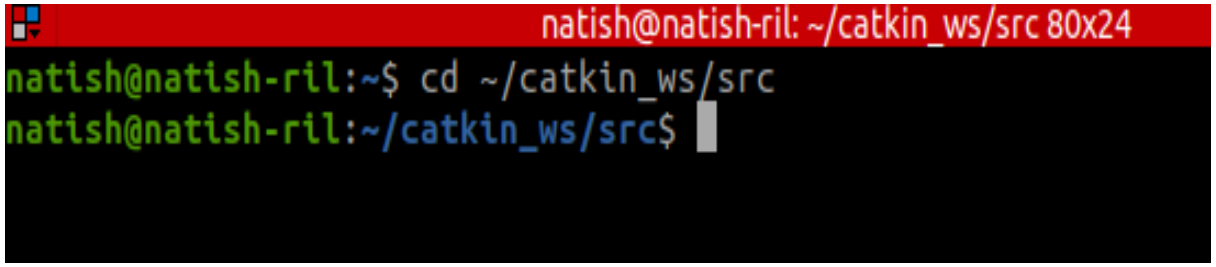
2. cd

The cd command is used to change directories. Either one by one or altogether.

Commands:

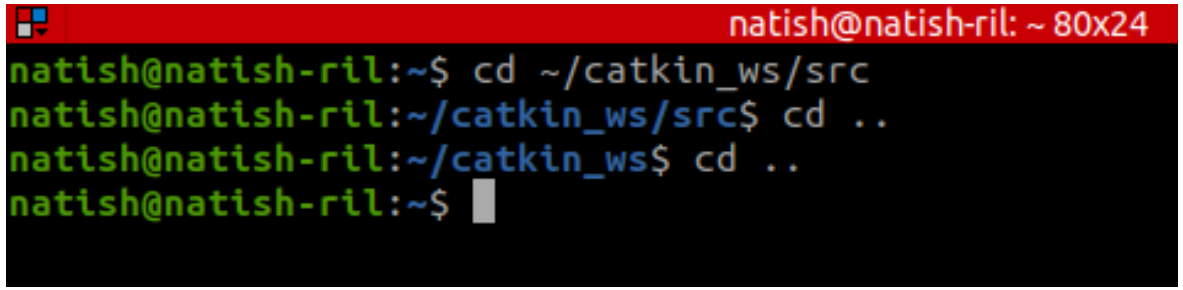
```
cd catkin_ws → cd src → cd robots
cd ~/catkin_ws/src/robots
```



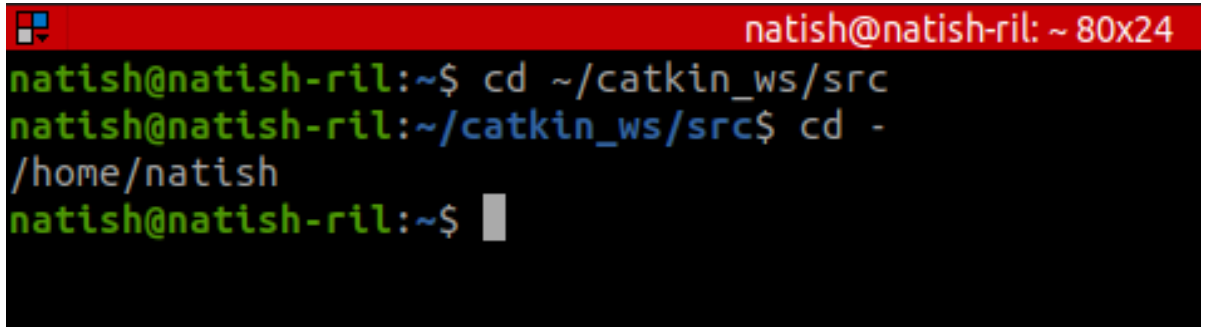


BASIC COMMANDS IN UBUNTU

To go back to the previous directory: `cd ..`

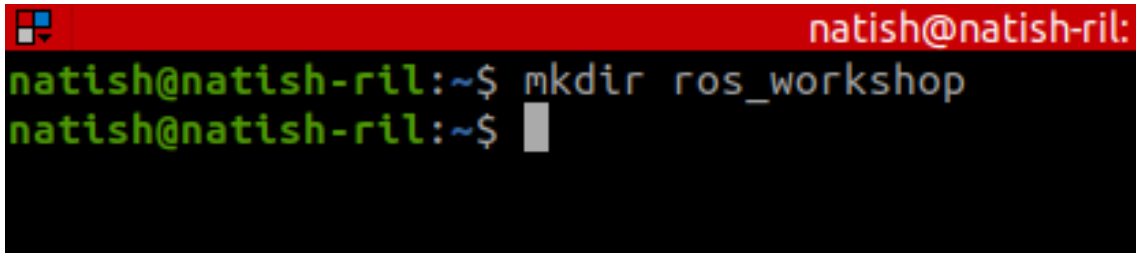


To go back to the home directory: `cd -`



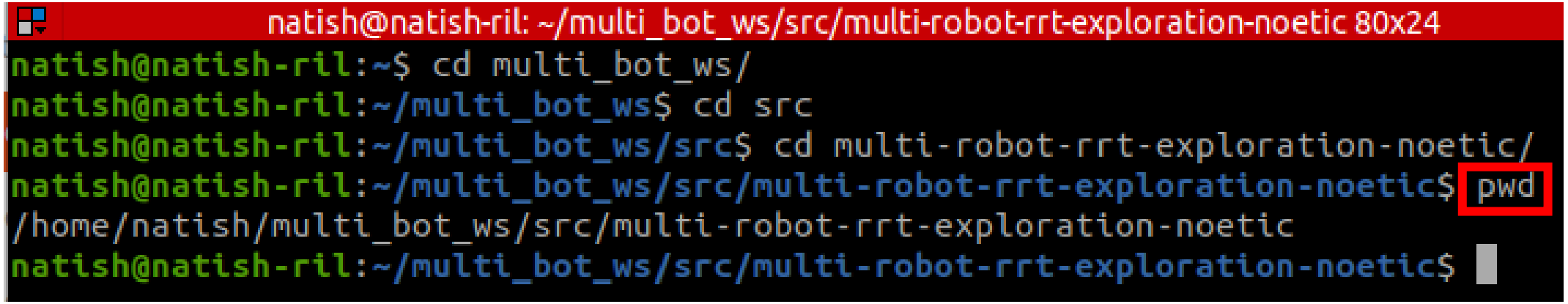
3. mkdir

This command is used to create a new directory. Very handy command to create directories in the command line.



4. pwd

The pwd command allows you to print the current working directory on your terminal.



5. rm

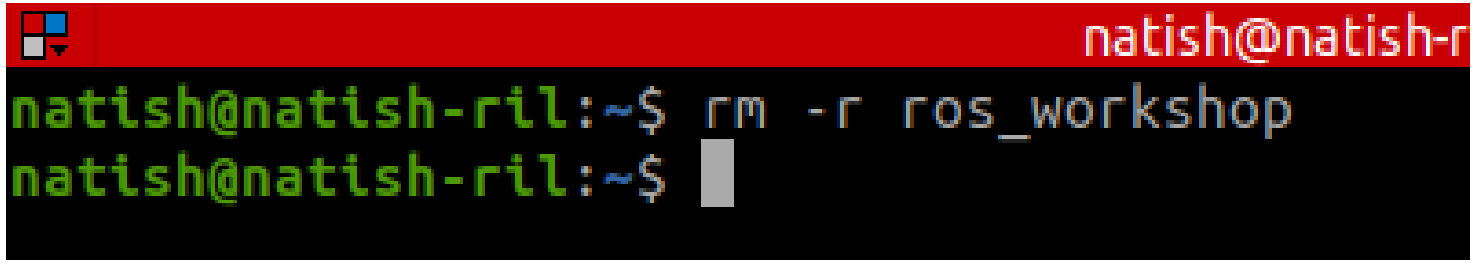
The rm command is used to delete files and folders. To delete directories the –r is added to the command.

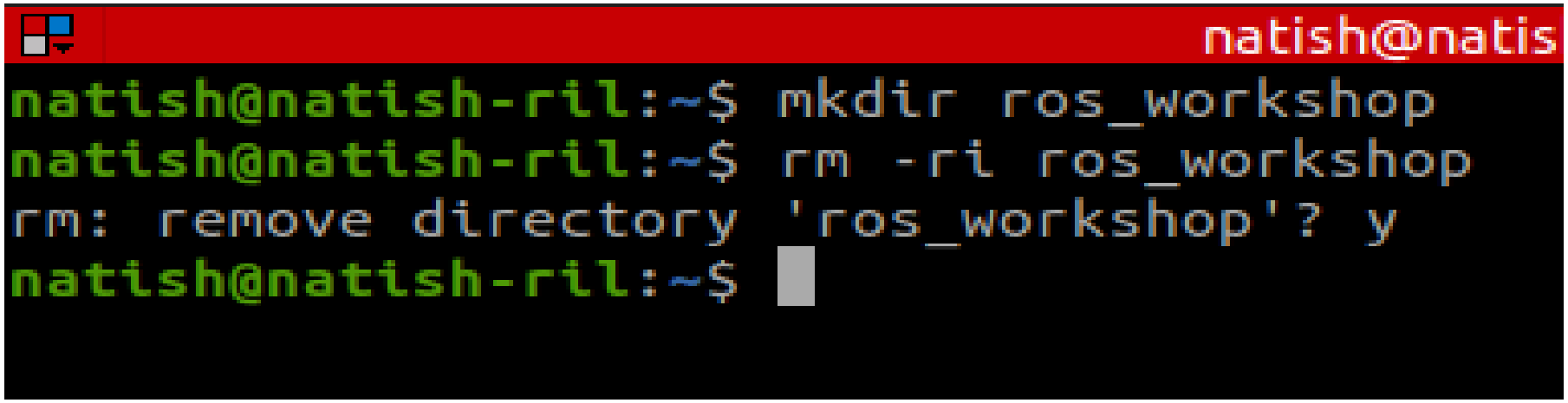
```
rm -r <folder or directory name>
```

Note: This will delete the files without comfirmation. To ask for confirmation while deleting use,

```
rm –ri <folder or directory name>
```

BASIC COMMANDS IN UBUNTU

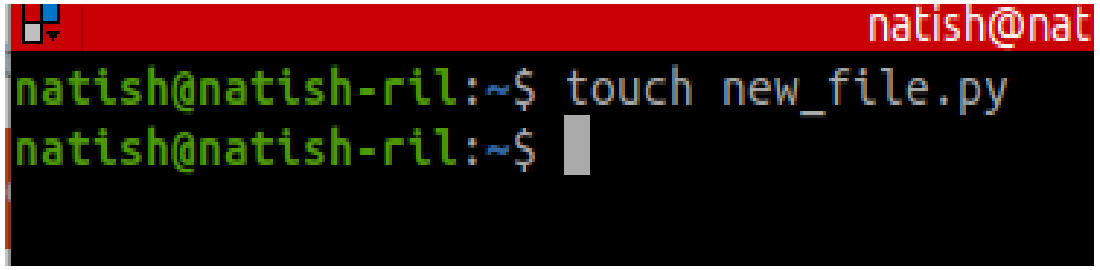


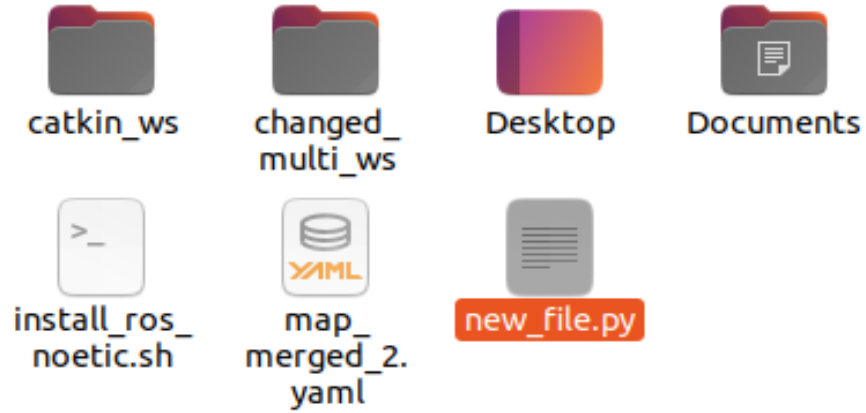


6. touch

This command is used to create a new and empty file.

touch <file name>

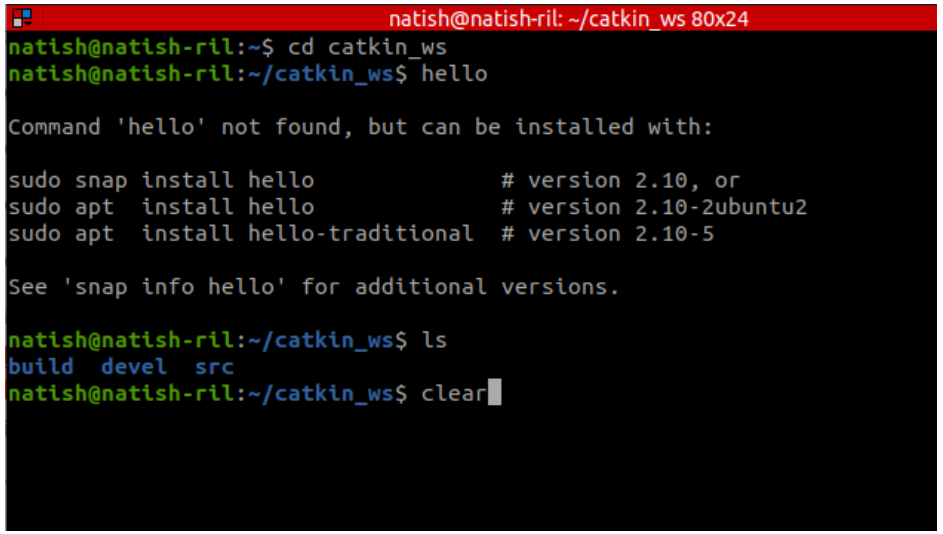


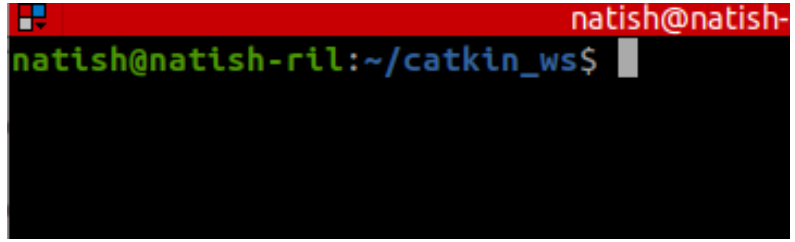


7. clear

The clear command allows you to clear the terminal and make it clean !

`clear`





8. cat

The cat command is used to open the contents inside a file.

cat <file name>

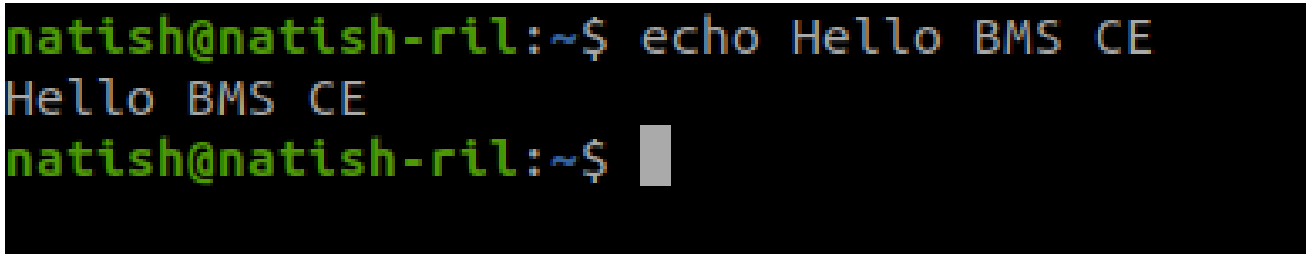
```
natish@natish-ril:~/real_world_ws/src/multi-robot-rrt-exploration-noetic/rrt_exp
loration$ cd scripts
natish@natish-ril:~/real_world_ws/src/multi-robot-rrt-exploration-noetic/rrt_exp
loration/scripts$ cat filter.py
#!/usr/bin/env python3

# --------Include modules---------------
from copy import copy
import rospy
from visualization_msgs.msg import Marker
from geometry_msgs.msg import Point
from nav_msgs.msg import OccupancyGrid
from geometry_msgs.msg import PointStamped
import tf
```

9. echo

This command prints whatever is written in the terminal. [In more advanced terms, it will print the messages published in a Rostopic]

echo <some words or name>

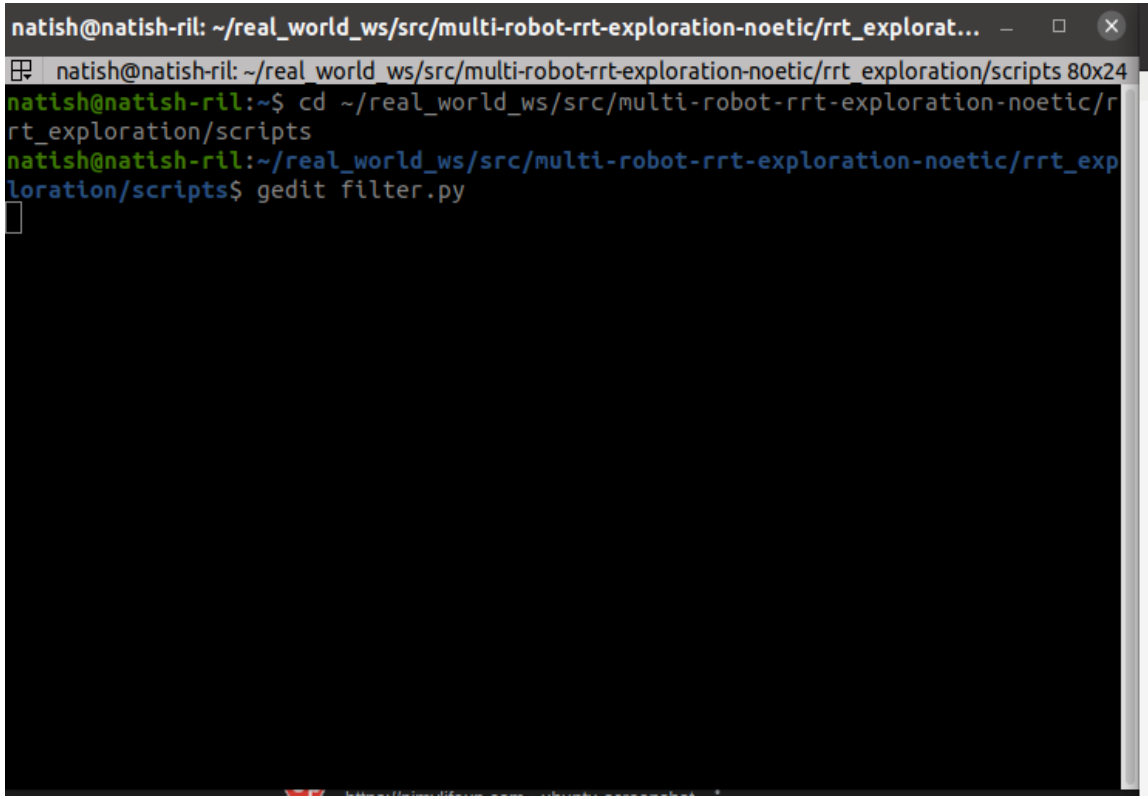


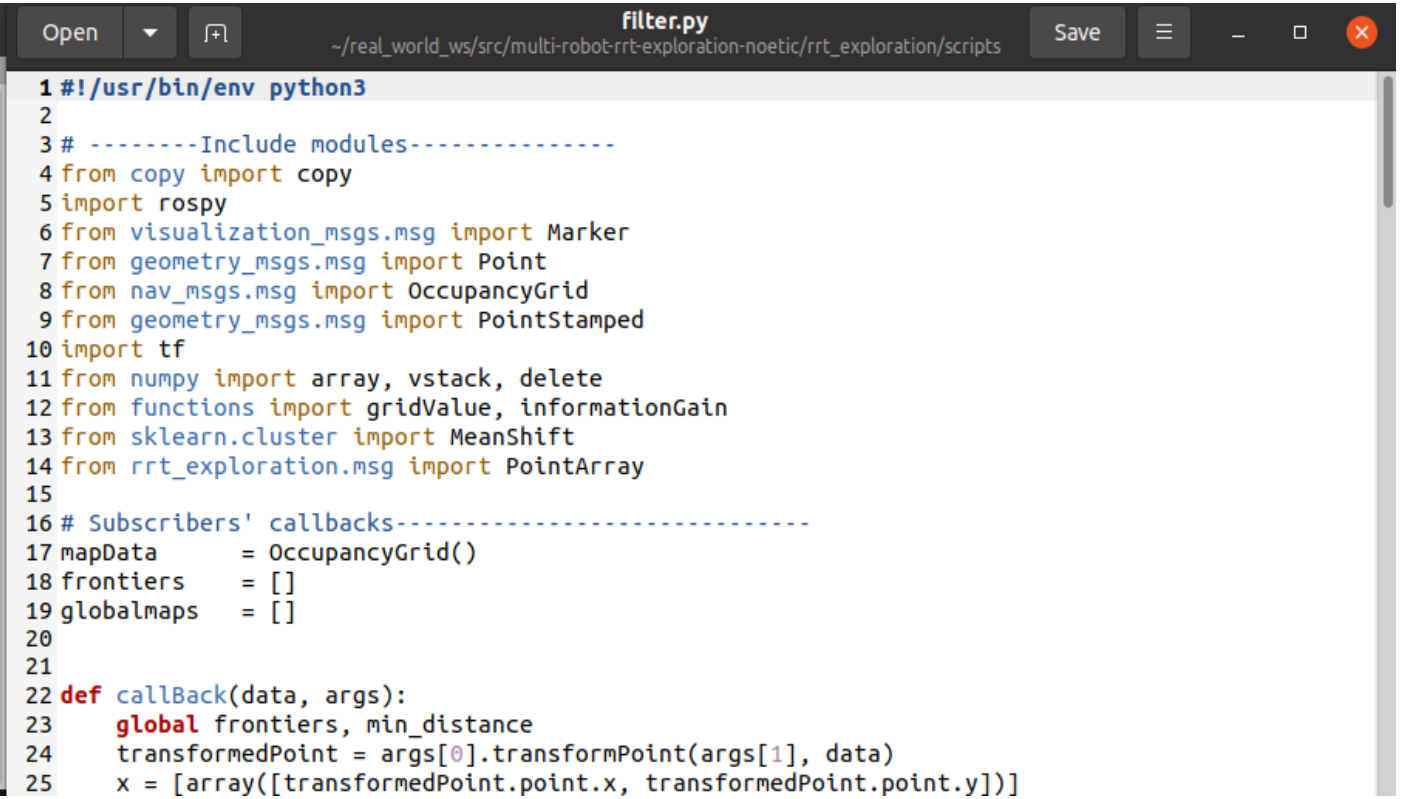
10. gedit

gedit is the official text editor of the GNOME desktop environment. You can open any file in gedit using the command

```
gedit new_file.py
```

BASIC COMMANDS IN UBUNTU





11. nano

nano is another famous text editor used in linux. Any file can be used in linux using the command:

```
nano new_file.py
```

BASIC COMMANDS IN UBUNTU

```
natish@natish-ril:~/real_world_ws/src/multi-robot-rrt-exploration-noetic/rrt_exp
loration/scripts$ nano filter.py
```

```
  GNU nano 4.8                          filter.py                          Modified
#!/usr/bin/env python3

# --------Include modules---------------
from copy import copy
import rospy
from visualization_msgs.msg import Marker
from geometry_msgs.msg import Point
from nav_msgs.msg import OccupancyGrid
from geometry_msgs.msg import PointStamped
import tf
from numpy import array, vstack, delete
from functions import gridValue, informationGain
from sklearn.cluster import MeanShift
from rrt_exploration.msg import PointArray

# Subscribers' callbacks------------------------------
mapData       = OccupancyGrid()
frontiers     = []
globalmaps    = []
```

12. source

It lets you bring in commands from a file and use them directly in your terminal. i.e Even if the file is from a different directory, it lets you open in the current terminal environment.

```
source FILENAME
```

These are the most important linux commands !!

BASIC COMMANDS IN UBUNTU

```
natish@natish-ril:~$ cd multi_bot_ws/
natish@natish-ril:~/multi_bot_ws$ roslaunch ros_multi_tb3 single_tb3_house.launc
h
RLException: [single_tb3_house.launch] is neither a launch file in package [ros_
multi_tb3] nor is [ros_multi_tb3] a launch file name
The traceback for the exception was written to the log file
natish@natish-ril:~/multi_bot_ws$ source devel/setup.bash
natish@natish-ril:~/multi_bot_ws$ roslaunch ros_multi_tb3 single_tb3_house.launc
h
... logging to /home/natish/.ros/log/531174c2-0a00-11ef-8cc8-f942026cf68c/roslau
nch-natish-ril-8523.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://192.168.0.125:38689/
```

TERMINATOR !

Terminator is a Linux terminal on drugs which lets you:

- ❖ Split horizontally
- ❖ Split vertically

Without opening separate windows !!

Installing Terminator

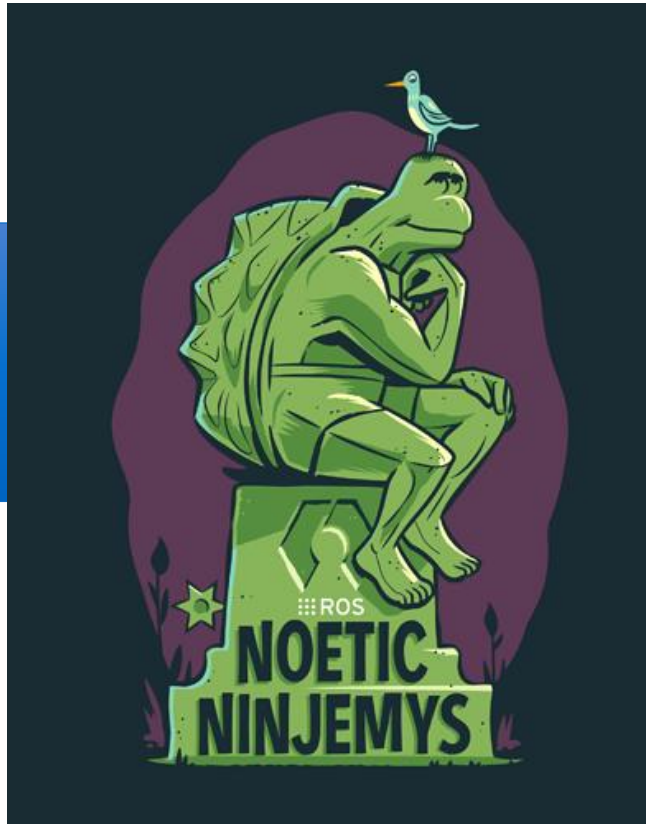
Use the command:

```
sudo apt install terminator
```

Enter your password and you have **terminator** in you PC !

Installing ROS Noetic

Refer to the readme files provided



GitHub Repository

https://github.com/MukilSaravanan/ROS1_Workshop

Installing ROS Noetic

1. Setup your computer to accept software from packages.ros.org

```
sudo sh -c 'echo "deb
http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list’
```

2. Install cURL

```
sudo apt install curl
curl -s
https://raw.githubusercontent.
com/ros/rosdistro/master/ros.a
sc | sudo apt-key add -
```

Installing ROS Noetic

BONUS 

Why cURL?

cURL, which stands for client URL, is a command line tool that developers use to transfer data to and from a server.

3. Make sure your package is up-to date:

```
sudo apt update
```

4. Install ROS ☺

```
sudo apt install ros-noetic-desktop-full
```

5. Source ROS Noetic

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

6. Install the dependencies

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

7. Initialize Rosdep

```
sudo rosdep init
rosdep update
```

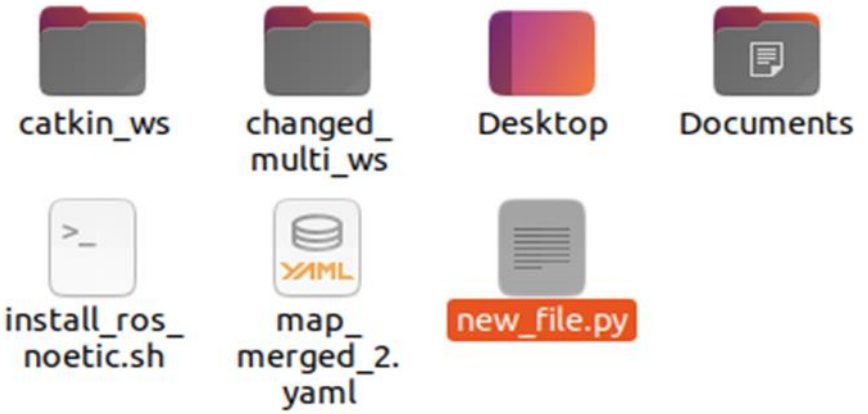
Check if ros is installed correctly !

Using the command:

```
rosversion -d
```

ROS Workspace

- A workspace is a directory containing ROS packages (You will get to know about it soon)
- Before using ROS , it's necessary to source your ROS installation workspace in the terminal you plan to work in.
- This makes ROS packages available for you to use in that terminal.



From the fig above, The catkin_ws and changed_multi_ws are ROS workspaces

Credits: https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html

Creating a workspace

1. Create a new directory with the name of the workspace.
2. Create a folder called ‘src’ inside the workspace directory.
3. Use the command catkin_make while remaining in the created directory.
4. Two additional folders named devel and build will be created inside the catkin_ws workspace

Note: While using catkin_make, it's essential not to be within the src folder but rather in the root directory of your Catkin workspace, which contains the src folder

Let us create our first workspace !

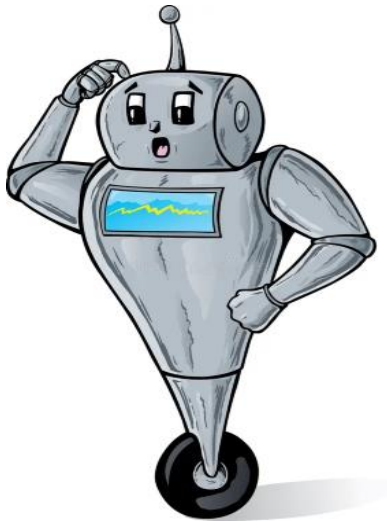
1.	`mkdir catkin_ws`	Create a workspace with a any name
2.	`cd catkin_ws`	Go inside the workspace
3.	`mkdir src`	Create a folder named ‘src’
4.	`cd ..`	Go back to the catkin_ws directory
5.	`catkin_make`	Use the command catkin_make to build the ws

Sourcing the workspace

1. You can be at only one workspace at a moment.

2. When you "source" a workspace, you essentially tell the shell environment where to find the ROS packages and scripts contained within that workspace.

Eg: Even you move from **catkin_ws** to **robot_ws**, you will still be at **catkin_ws** if you don’t source the **robot_ws**. Hence sourcing is essential.



How to source a workspace?

1. Navigate to the workspace

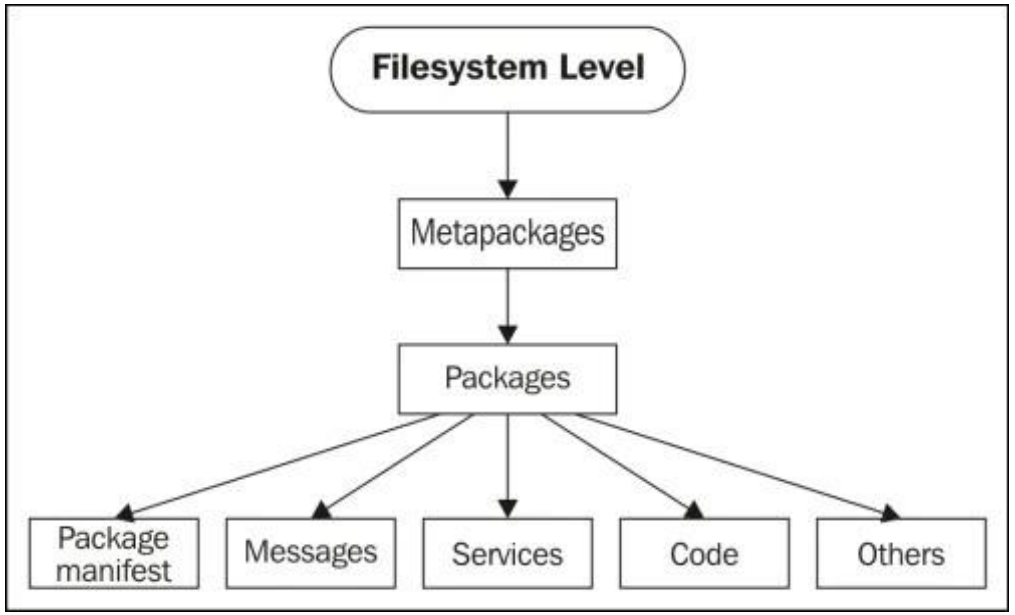
```
cd <workspace_name>
```

2. Use the command

```
source devel/setup.bash
```

ROS Package

- The ROS **packages** are the most basic unit of the ROS software.
- They contain the ROS runtime process (**nodes**), libraries, configuration files, and so on, which are organized together as a single unit.
- Packages are the atomic build items of ROS



Credits: https://vnav.mit.edu/labs/lab2/ros101.html

Creating a package

1. Navigate inside the src folder of the created workspace.
2. Use the command **catkin_create_pkg <package_name> <dependencies>** to create a package
3. Navigate again to catkin_ws folder and use **catkin_make** command to build the workspace

Note: Every time you create a new package, don't forget to build the workspace

Creating our first package !

1. `cd ~/catkin_ws/src` — Navigate to src folder

2. `Catkin_create_pkg first_package std_msgs roscpp rospy` — Command to create package

Name of the package Dependencies

4. `cd ..` — Navigating to catkin_ws directory

5. `catkin_make` — Building the ws

ROS Concepts [3]

- ROS Node
- ROS Topic
- ROS Message
- ROS Service
- ROS Parameter Server
- ROS Master
- ROS Core

[3] ROS/Concepts - ROS Wiki

ROS Node

- Process that performs computation [4]
- E.g. Processing image stream from camera, computing wheel velocity to motor

ROS Node 1

Image Stream

ROS Node 2

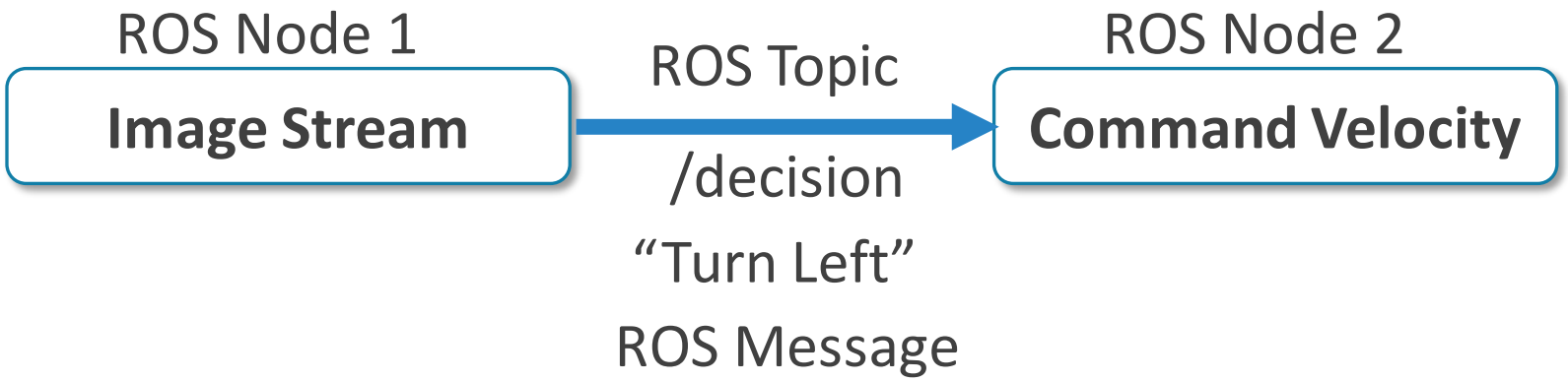
Command Velocity

- Specialized for a single purpose & modular
- Written in Python or C++
- Can talk to other nodes

[4] Nodes - ROS Wiki

ROS Topic

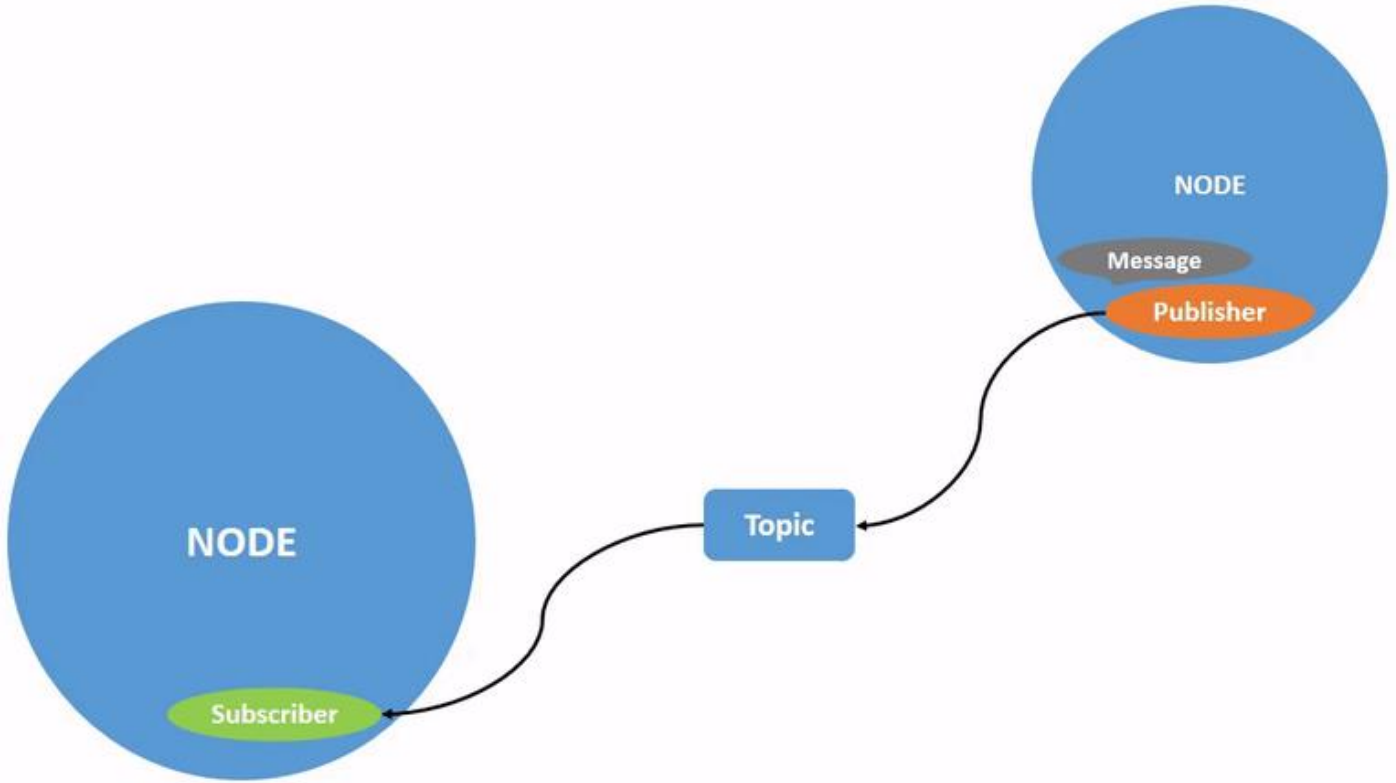
- Named buses over which nodes exchange messages [5]
- Unidirectional
- Pub – Sub model



[5] Topics - ROS Wiki
Credits: Understanding topics — ROS 2 Documentation: Humble documentation

ROS Topic

- Named buses over which nodes exchange messages [5]
- Unidirectional
- Pub – Sub model



[5] Topics - ROS Wiki
Credits: Understanding topics — ROS 2 Documentation: Humble documentation

ROS Message

- Simply a data structure, comprising typed fields [6,7]
- Simple or Nested
- Will not block until receipt, messages get queued (buffer length)

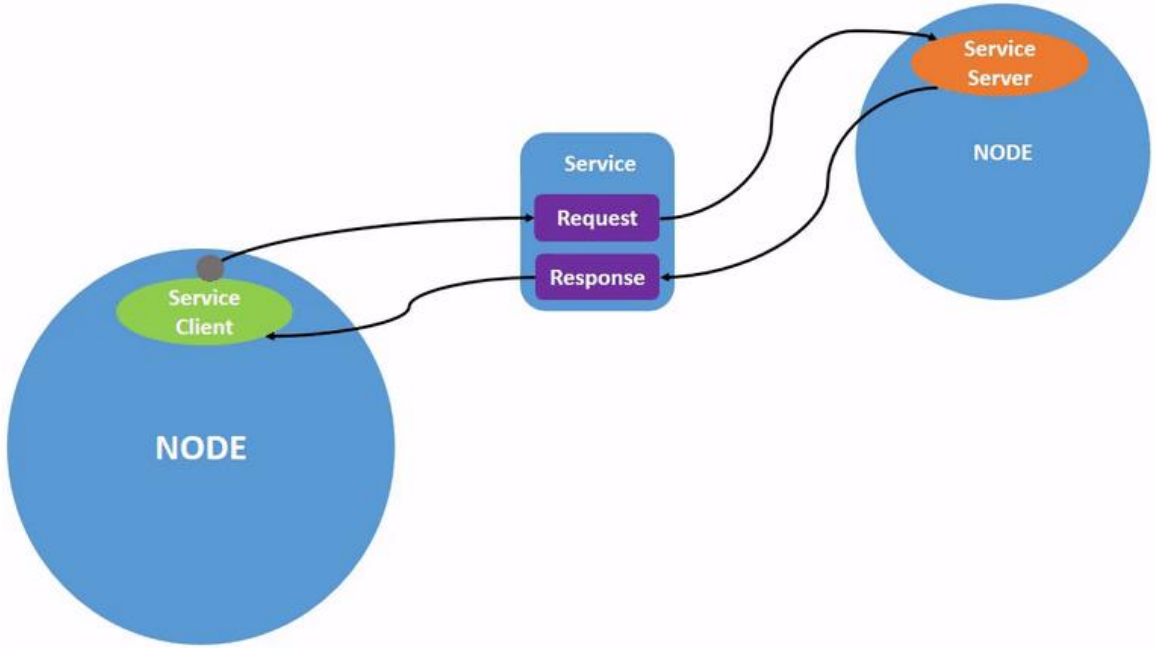
fieldtype1 fieldname1
E.g.,
float32 left_wheel_velocity

Simple	Nested	Common
Bool Int8/16 Uint8/6 Float32/16 String	Int8MultiArray Uint8MultiArray ColorRGBA	std_msgs geometry_msgs sensor_msgs

[6] Messages - ROS Wiki
[7] msg - ROS Wiki

ROS Service

- Mechanism for a node to send a request to another node, and receive a response [8]
- Bidirectional
- Client – Server model



[8] Services - ROS Wiki
Credits: Understanding services — ROS 2 Documentation: Humble documentation

ROS Parameter Server

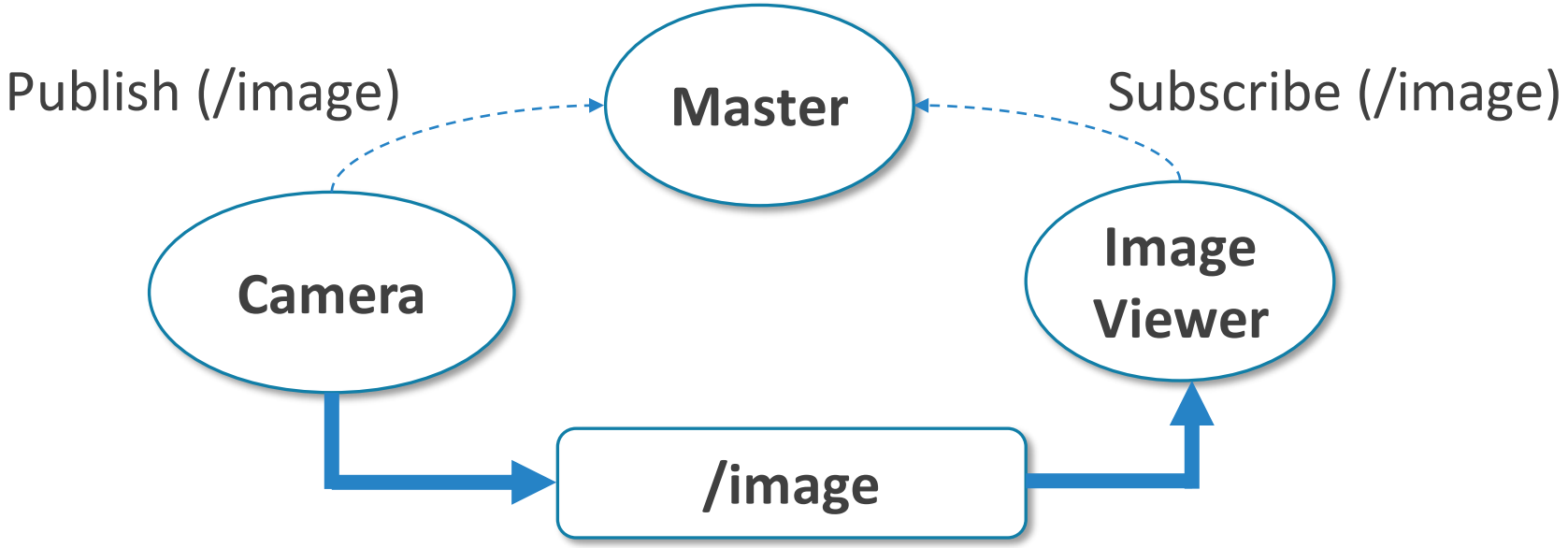
- Simply a shared dictionary to store static data [9]
- E.g. Storing wheel radius, camera fps
- Nodes use to store and retrieve parameters at runtime

```
/camera/left/name: leftcamera
/camera/left/fps: 30
/camera/right/name: rightcamera
/camera/right/fps: 60
/wheel/left/radius: 0.05
/wheel/right/radius: 0.05
```

[9] Parameter Server - ROS Wiki

ROS Master

- Matchmaker between nodes
- Provides naming & registration services to nodes [10]
- ROS Master should be running on a computer/robot.



[10] Master - ROS Wiki

ROS Core

- ROS Core will start up [11]
  - ROS Master
  - ROS Parameter Server
  - rosout logging node
- ROS Core is mandatory for nodes to communicate

[11] Master - ROS Wiki

GitHub Repository

https://github.com/MukilSaravanan/ROS1_Workshop

Understanding with Turtlesim

ROS Node

- $ roscore
- $ rosnode list
- $ rosnode info /rosout
- $ rosrun turtlesim turtlesim_node
- $ rosnode list
- $ rosrun turtlesim turtlesim_node __name:=my_turtle
- $ rosnode list
- $ rosnode ping my_turtle

Understanding with Turtlesim

ROS Topic

- $ roscore
- $ rosrun turtlesim turtlesim_node
- $ rosrun turtlesim turtle_teleop_key
- $ rosrun rqt_graph rqt_graph
- $ rostopic -h
- $ rostopic echo /turtle1/cmd_vel
- $ rostopic list -h
- $ rostopic type /turtle1/cmd_vel

Recap

- Understood ROS & its importance
- Installed ROS Noetic
- Learnt Basic Linux Commands
- ROS Concepts – Nodes, Topics, Messages, Service, Parameter Server, Master
- Had hands-on experience delving deeper into ROS Concepts with Turtlesim

Thank You

Email: kalaivanank@iisc.ac.in
LinkedIn: Kalaivanan K

Email: mukil.saravanan.edu@gmail.com
Portfolio: mukilsaravanan.github.io
LinkedIn: Mukil Saravanan

Email: natishmeng@gmail.com
LinkedIn: Natish Murugesh

Let's get connected!