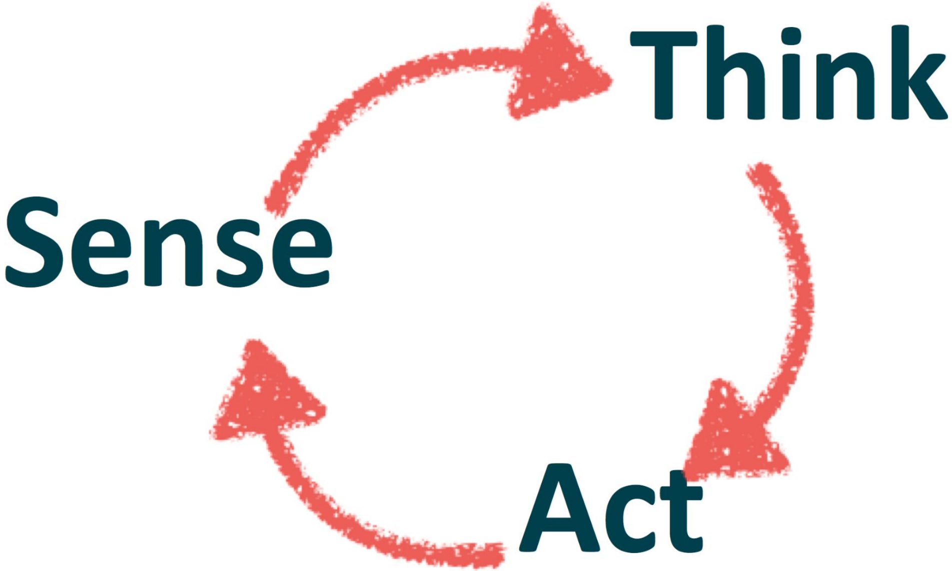
Lab1

Introduction to ROS - Summer 2022 - Innopolis University

By: Hany Hamed
Course instructor: Geesara Prathap

Introduction & Ice break

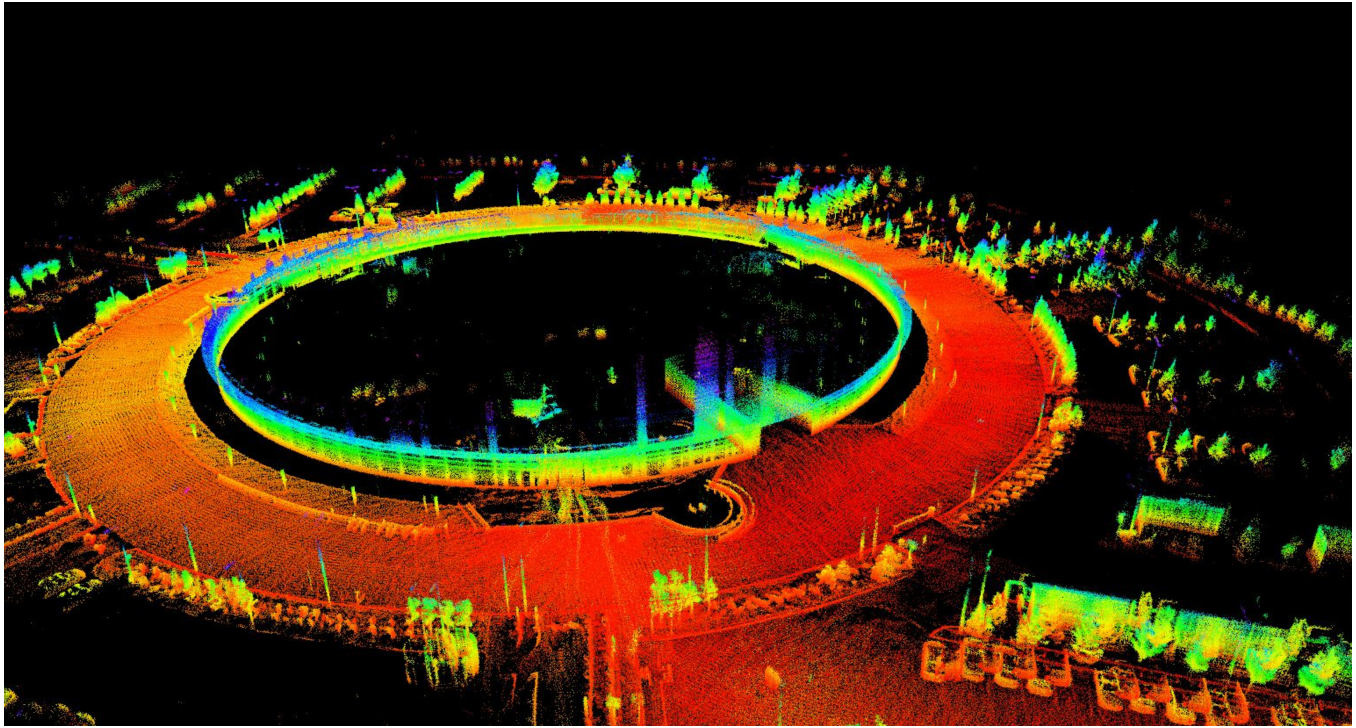
Robotics



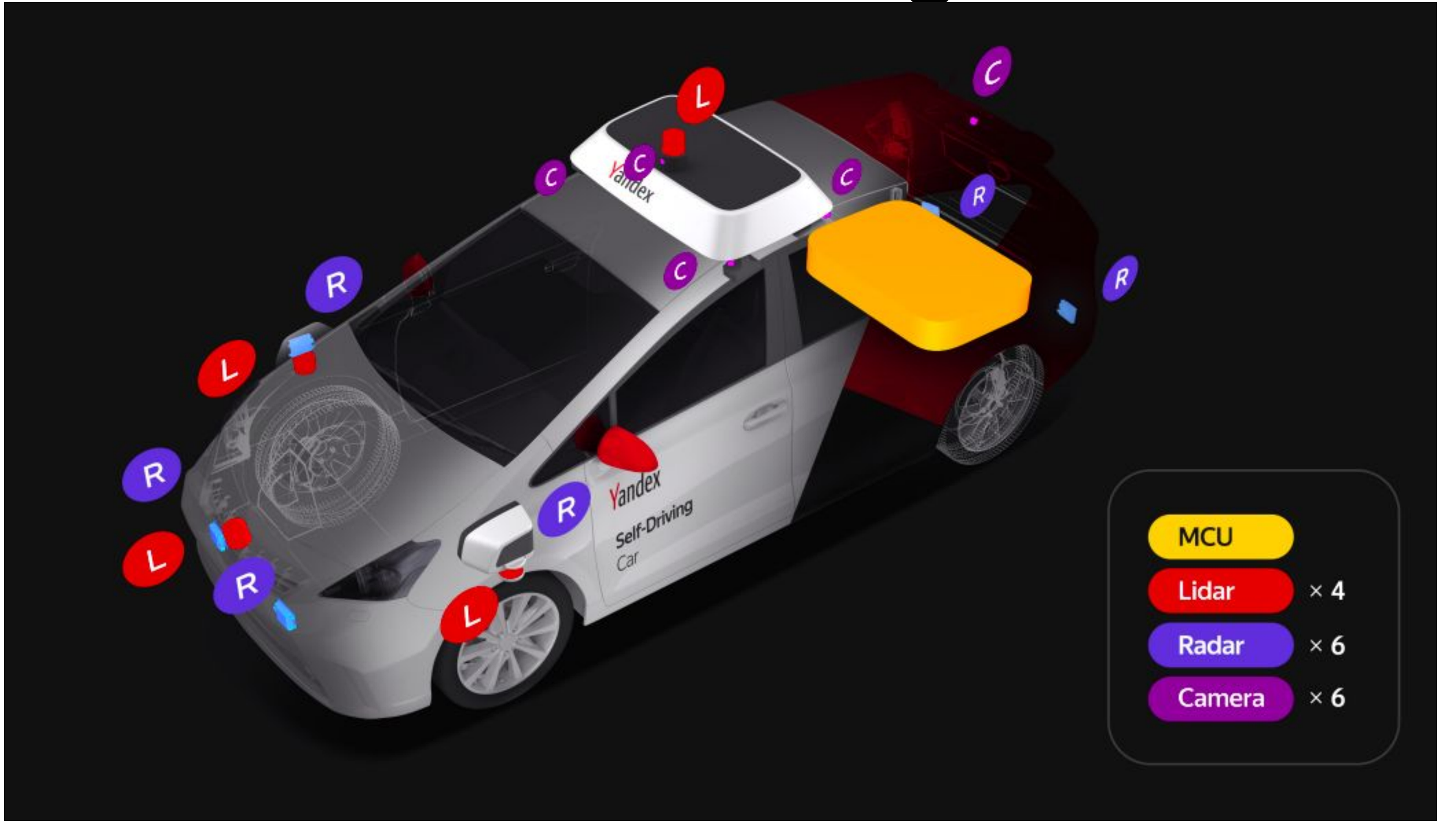
Source: https://research.csiro.au/dss/research/ubiquitous-ai/closed-loop-sensing/

Robotics: Sense, think & act

Robotics: Sense



Robotics: Yandex self-driving cars

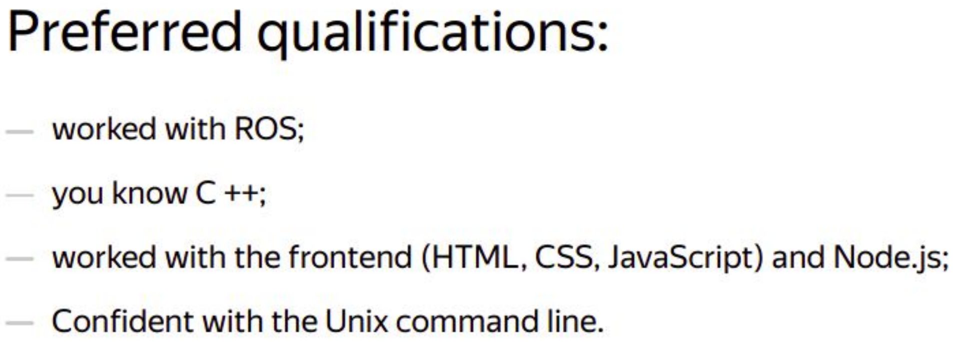


Source: https://unece.org/DAM/trans/doc/2019/wp29grva/GRVA-04-37.pdf

Robotics: Field & Jobs

Example:

- Yandex: Python developer for the team of remote control of unmanned vehicles



Preferred qualifications:

— worked with ROS;

— you know C ++;

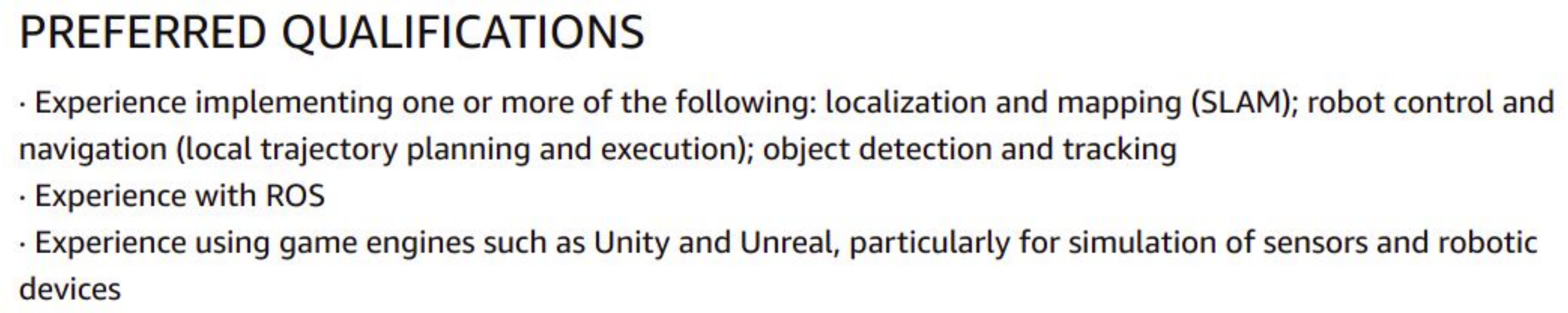
— worked with the frontend (HTML, CSS, JavaScript) and Node.js;

— Confident with the Unix command line.

Robotics: Field & Jobs

Example:

- Amazon Robotics: Software Development Engineer - Amazon Robotics - Autonomous Mobility, Autonomous Mobility (C++/ROS/SLAM) Team



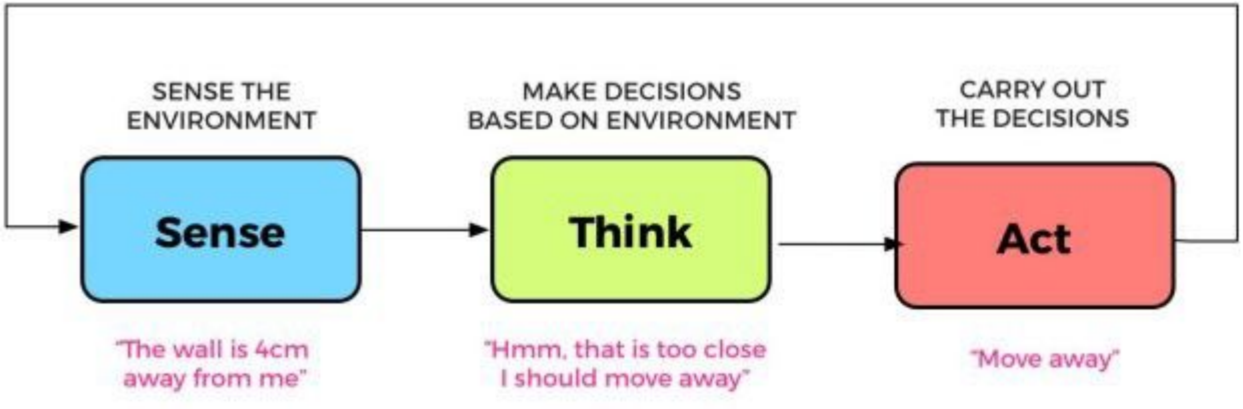
PREFERRED QUALIFICATIONS

· Experience implementing one or more of the following: localization and mapping (SLAM); robot control and navigation (local trajectory planning and execution); object detection and tracking

· Experience with ROS

· Experience using game engines such as Unity and Unreal, particularly for simulation of sensors and robotic devices

Robotics


SENSE THE ENVIRONMENT
MAKE DECISIONS BASED ON ENVIRONMENT
CARRY OUT THE DECISIONS
Sense
Think
Act
"The wall is 4cm away from me"
"Hmm, that is too close I should move away"
"Move away"

Source: https://youngzine.org/article-expert/technology/machines-sense-think-and-act

What is ROS?

- Robot Operating System, is it?
- Open-source robotics middleware
- Set of software frameworks for robot software development

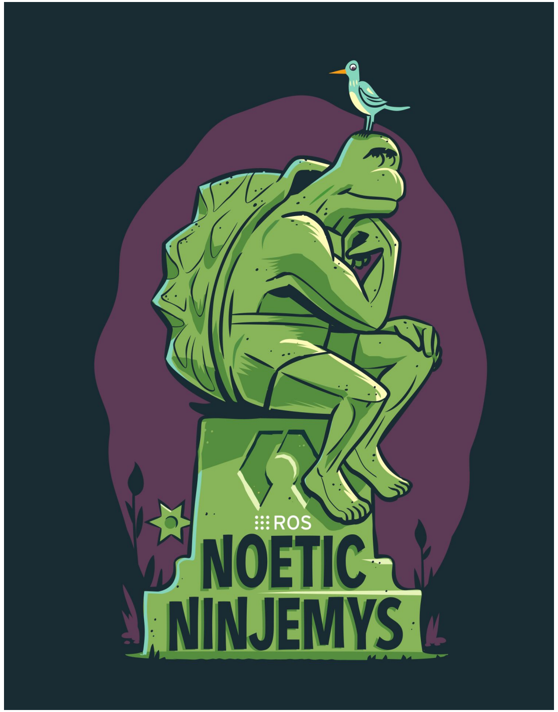
Why ROS?

- Open-source
- Big community
- Powerful, modular & reusable

When to use ROS?

- Simulation
- Hardware: motors, sensors and actuators

ROS Distribution?



ROS Distribution?

ROS Noetic

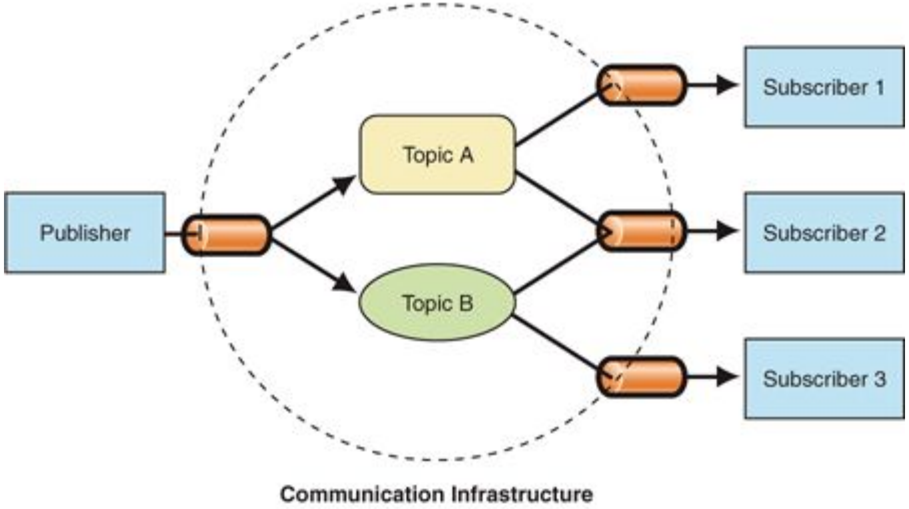
- Ubuntu 20
- Python3
- Gazebo 11
- CMake 3.16

ROS Melodic

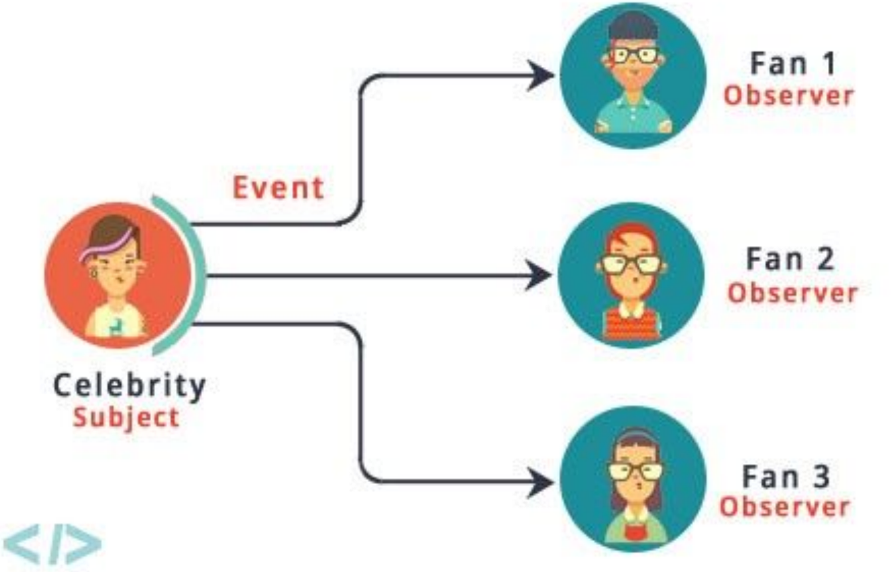
- Ubuntu 18
- Python2
- Gazebo 9
- CMake 3.10.2

Design pattern

Pub-sub



Observer



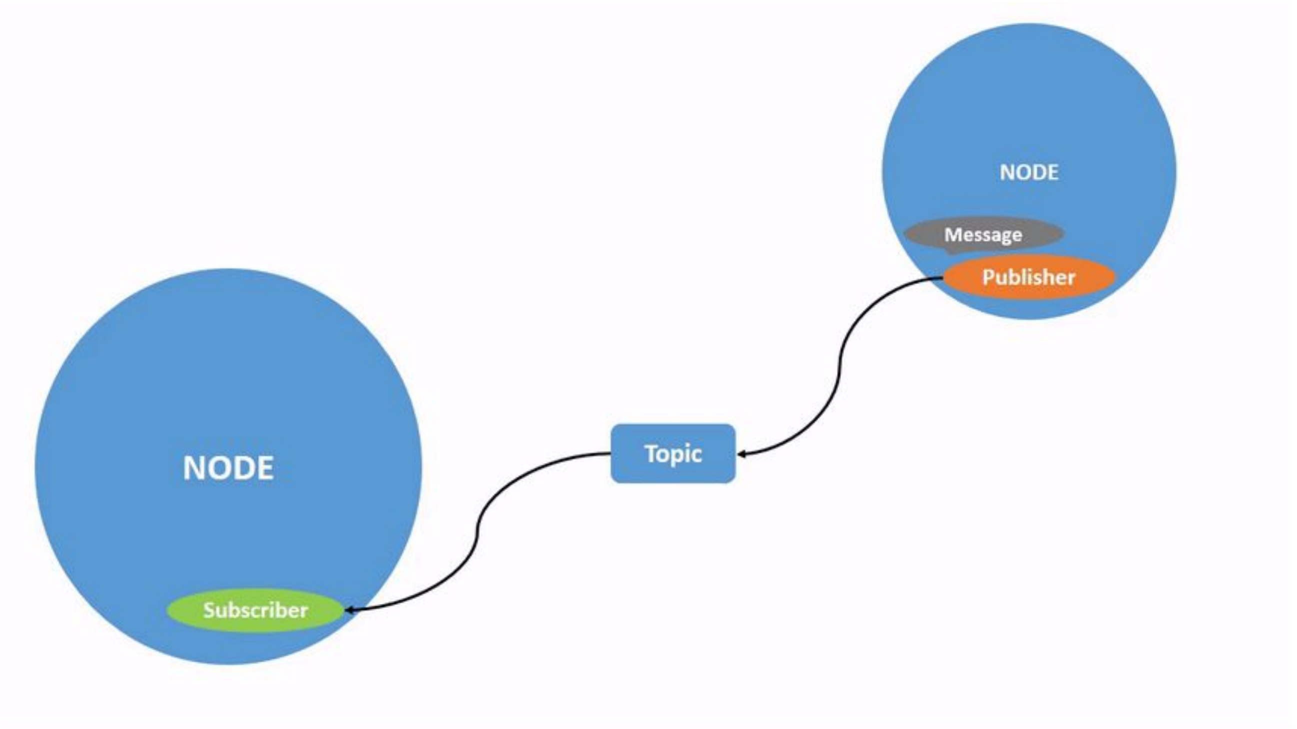
[Source (Pub-sub)]

[Source (Observer)]

[Differences between pub-sub & observer design patterns]

[Differences between pub-sub & observer design patterns2]

ROS architecture: Publisher & Subscriber



Source: https://docs.ros.org/en/foxy/Tutorials/Topics/Understanding-ROS2-Topics.html

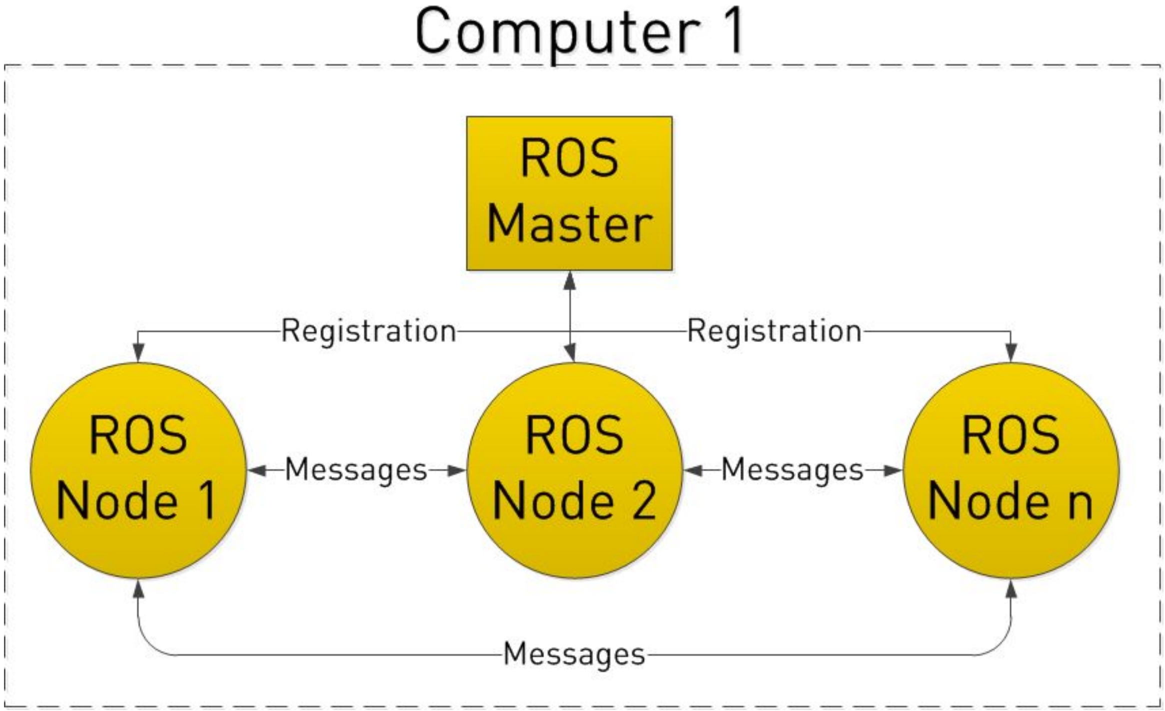
ROS Master

ROS Master provides:

- Naming
- Registration
- “Tracks publishers and subscribers to topics as well as services.”
- “The role of the Master is to enable individual ROS nodes to locate one another.”
- “Once these nodes have located each other, they communicate with each other peer-to-peer.”

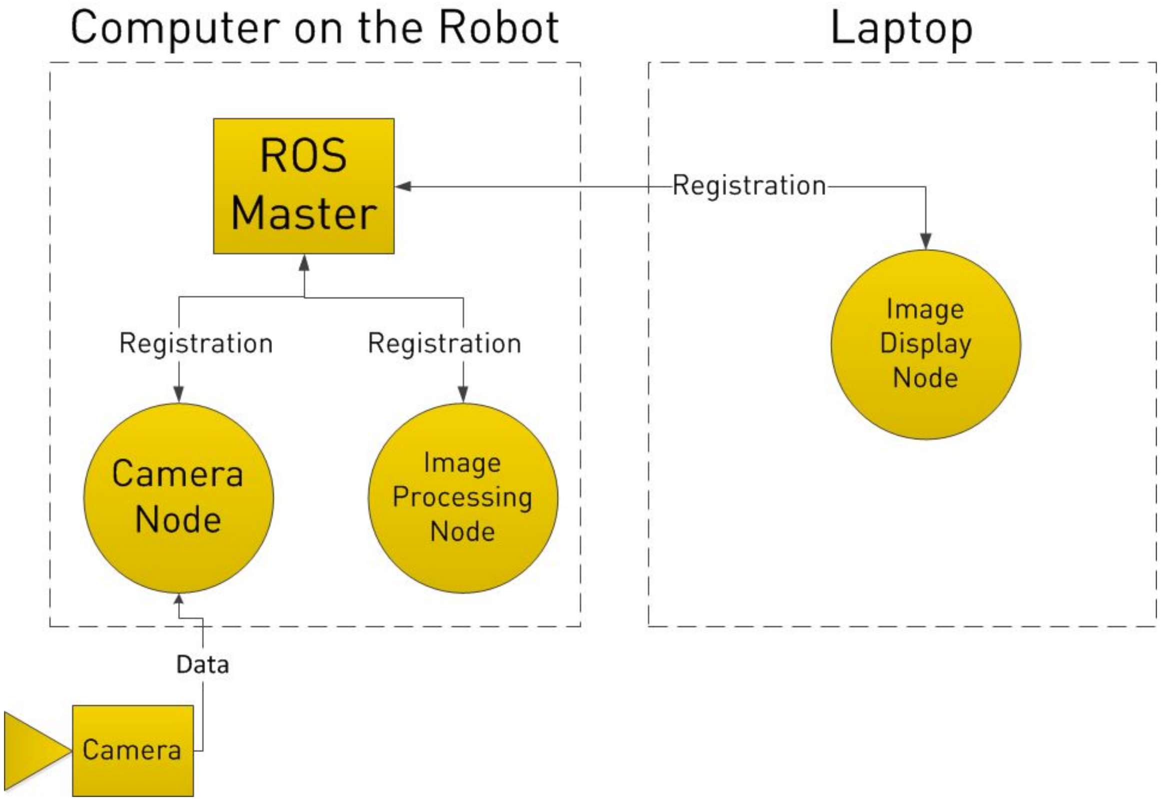
Source:
http://wiki.ros.org/Master#:~:text=The%20ROS%20Master%20provides%20naming,nodes%20to%20locate%20one%20another.

ROS architecture



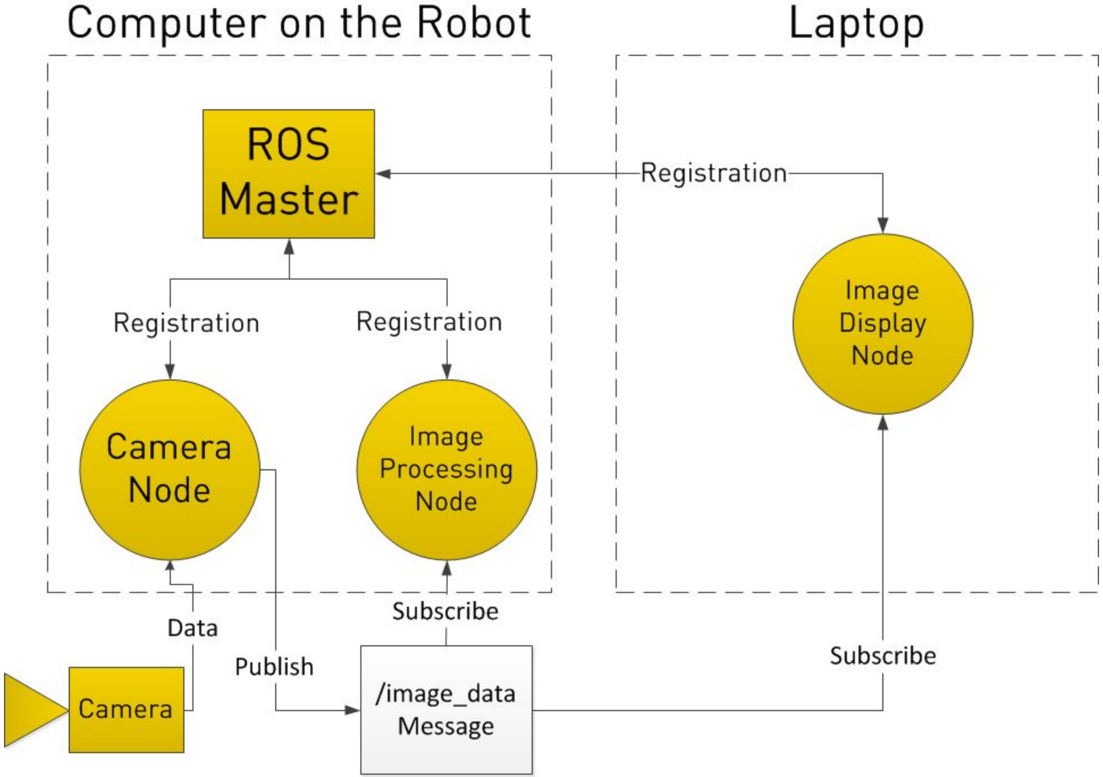
Source: https://www.clearpathrobotics.com/assets/guides/noetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

ROS architecture



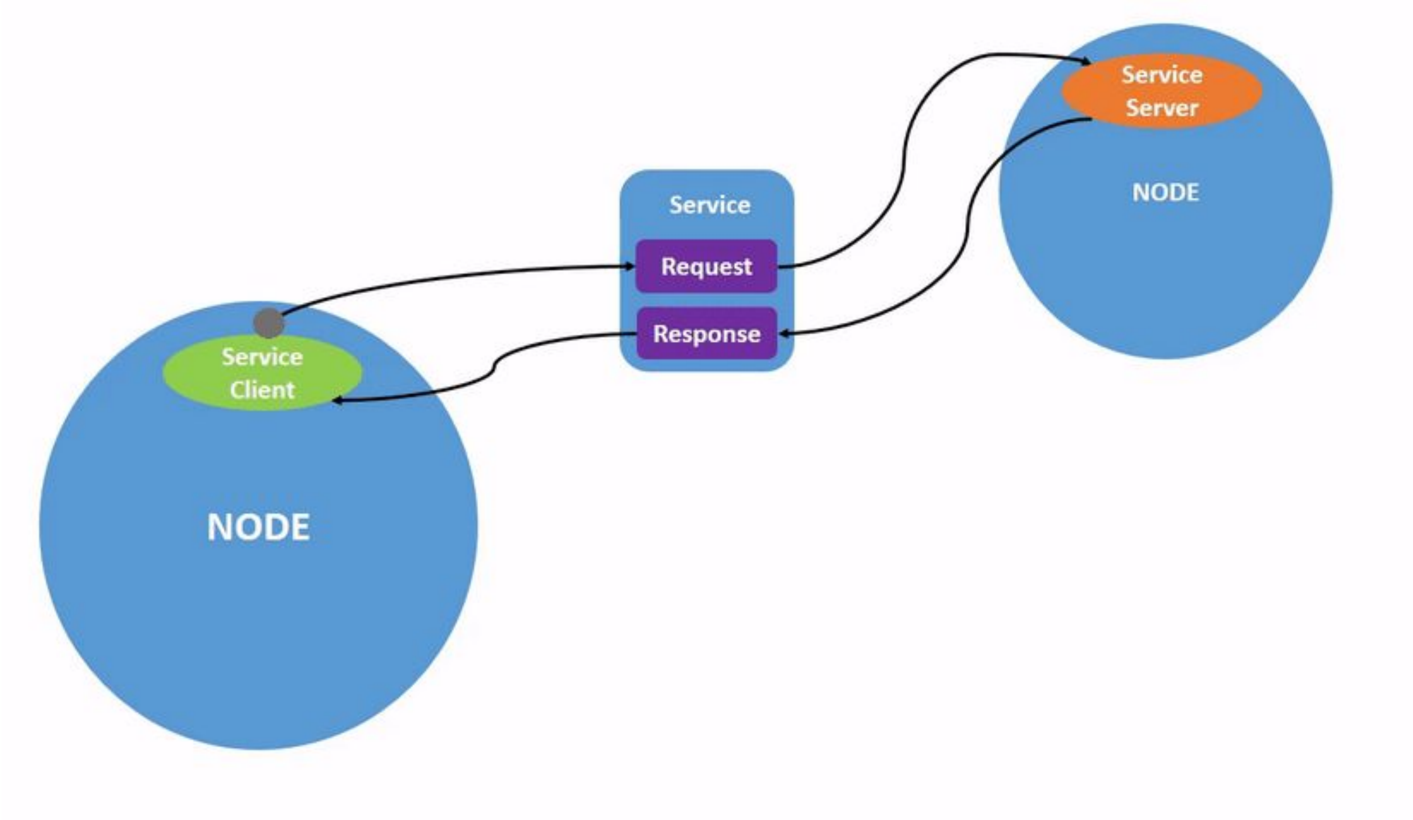
Source:
https://www.clearpathrobotics.com/assets/guides/noetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

ROS architecture



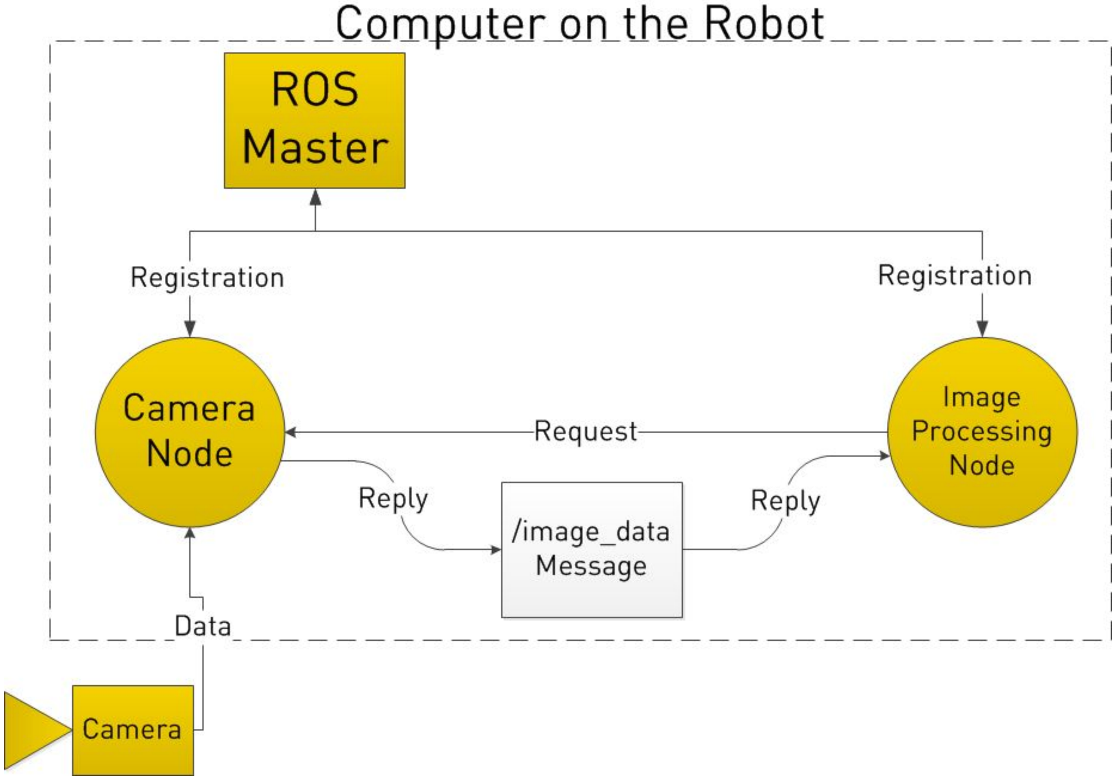
Source:
https://www.clearpathrobotics.com/assets/guides/noetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

ROS architecture: Service-client



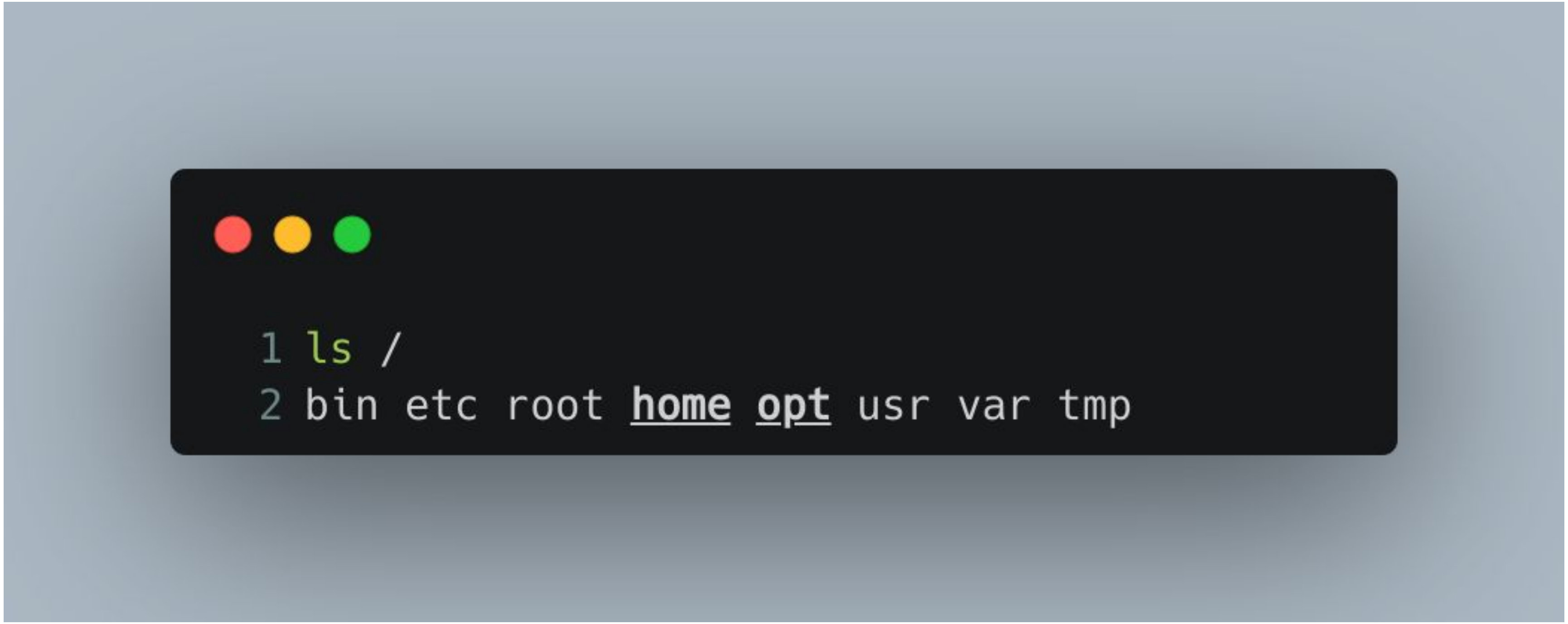
Source: https://docs.ros.org/en/foxy/Tutorials/Services/Understanding-ROS2-Services.html

ROS architecture



Source:
https://www.clearpathrobotics.com/assets/guides/noetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

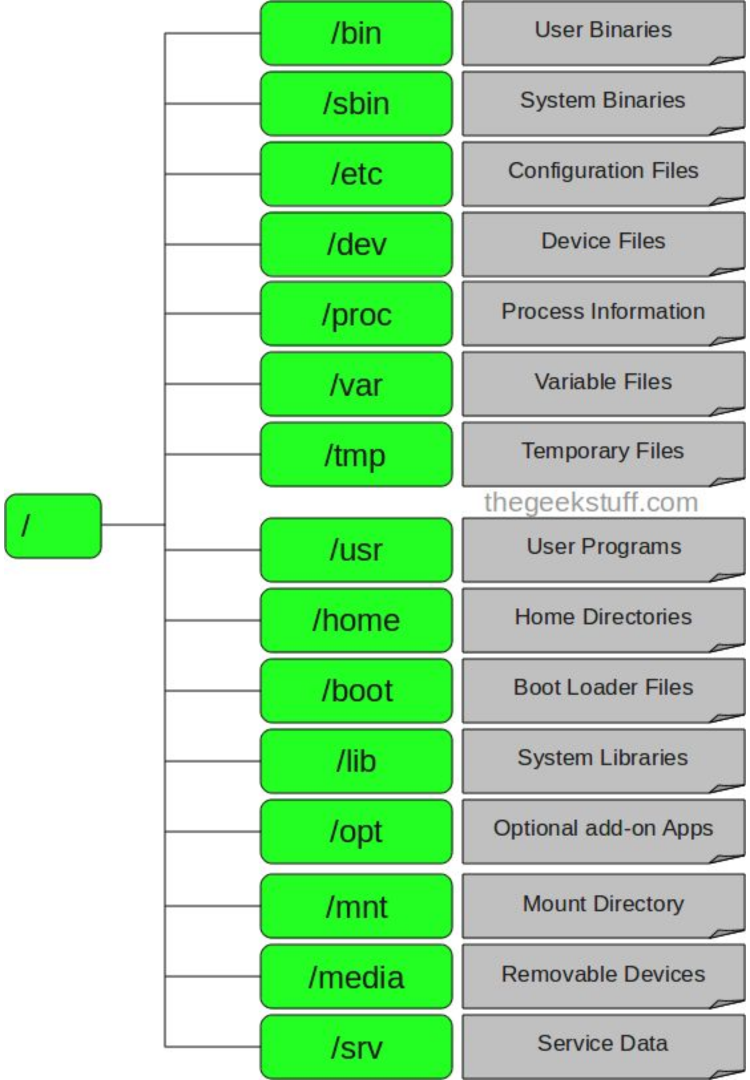
Linux directory structure



```
ls /
bin etc root home opt usr var tmp
```

Source: https://www.geeksforgeeks.org/linux-directory-structure/

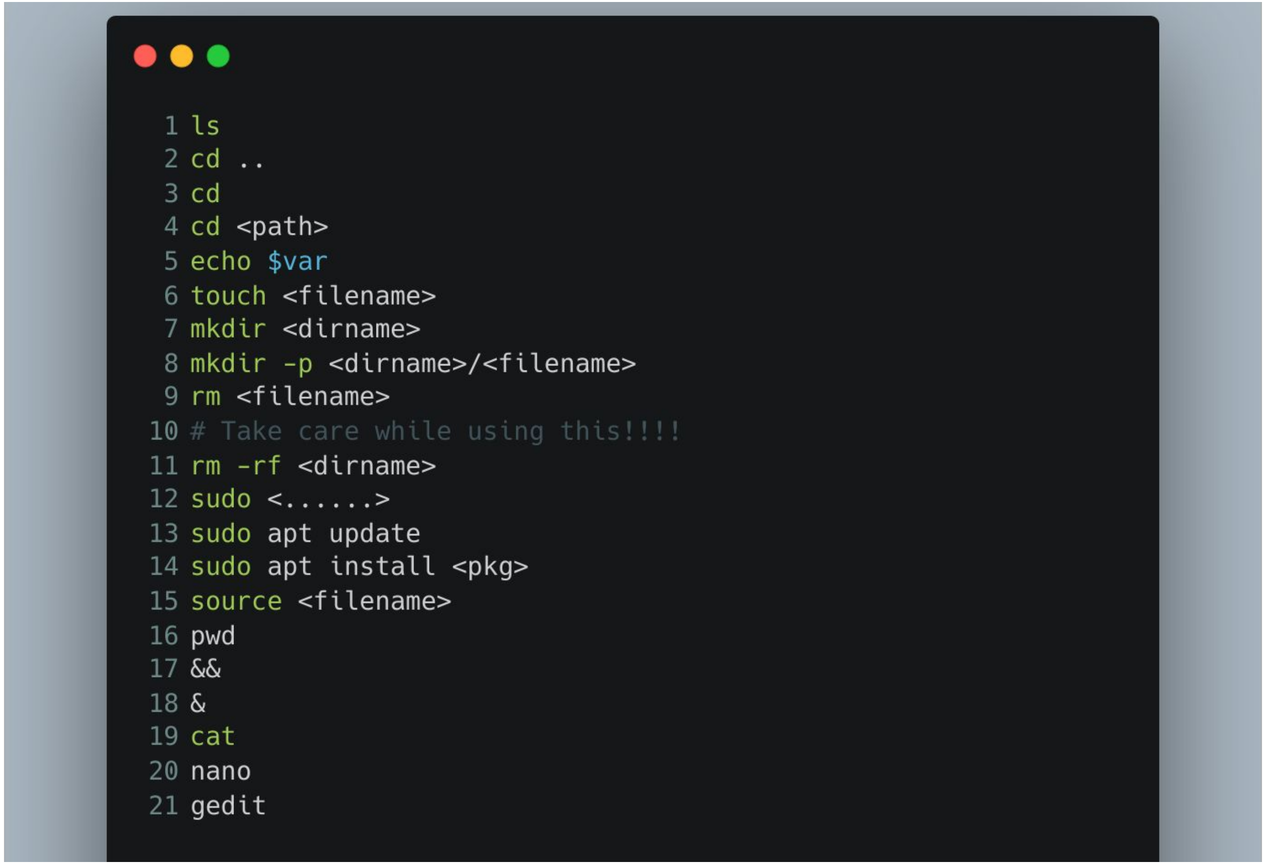
Linux directory structure



/	
/bin	User Binaries
/sbin	System Binaries
/etc	Configuration Files
/dev	Device Files
/proc	Process Information
/var	Variable Files
/tmp	Temporary Files
/usr	User Programs
/home	Home Directories
/boot	Boot Loader Files
/lib	System Libraries
/opt	Optional add-on Apps
/mnt	Mount Directory
/media	Removable Devices
/srv	Service Data

Source: https://www.thegeekstuff.com/2010/09/linux-file-system-structure/

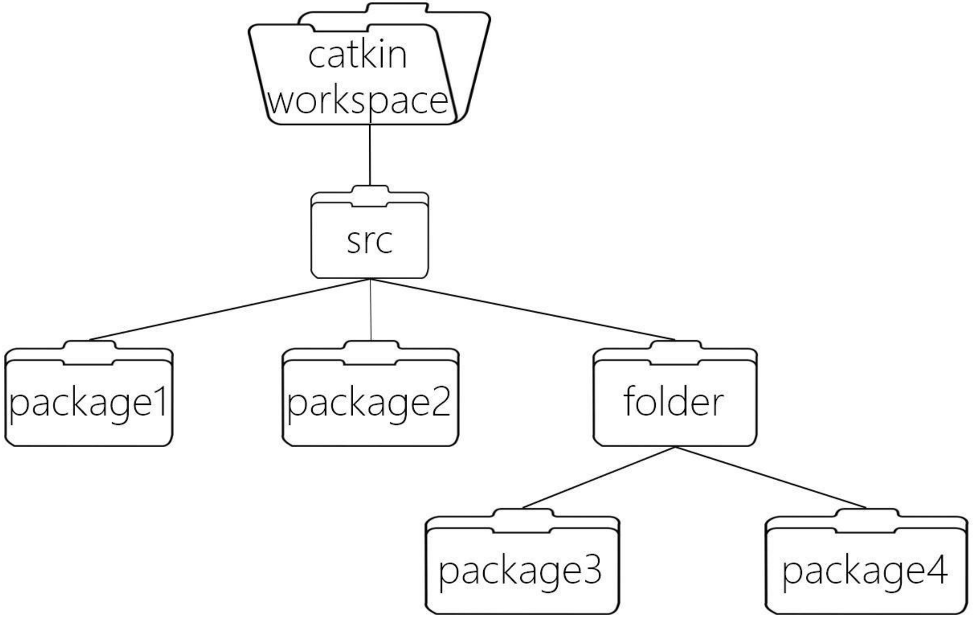
Linux commands



```
ls
cd ..
cd
cd <path>
echo $var
touch <filename>
mkdir <dirname>
mkdir -p <dirname>/<filename>
rm <filename>
# Take care while using this!!!!
rm -rf <dirname>
sudo <......>
sudo apt update
sudo apt install <pkg>
source <filename>
pwd
&&
&
cat
nano
gedit
```

ROS Workspace

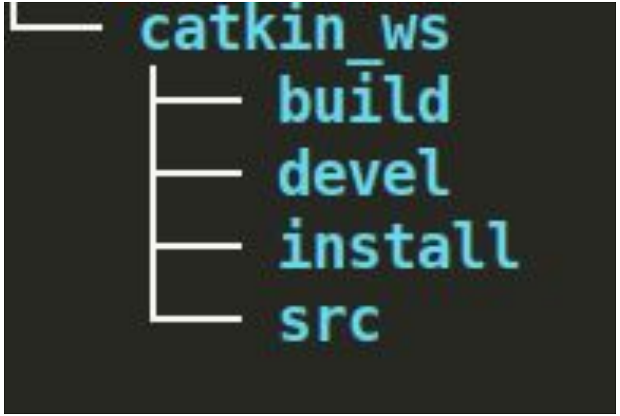
- Workspace
- Package
- Nodes
- Messages
- Services
- Launch



Source: https://russianblogs.com/article/3420187481/

Catkin Workspace

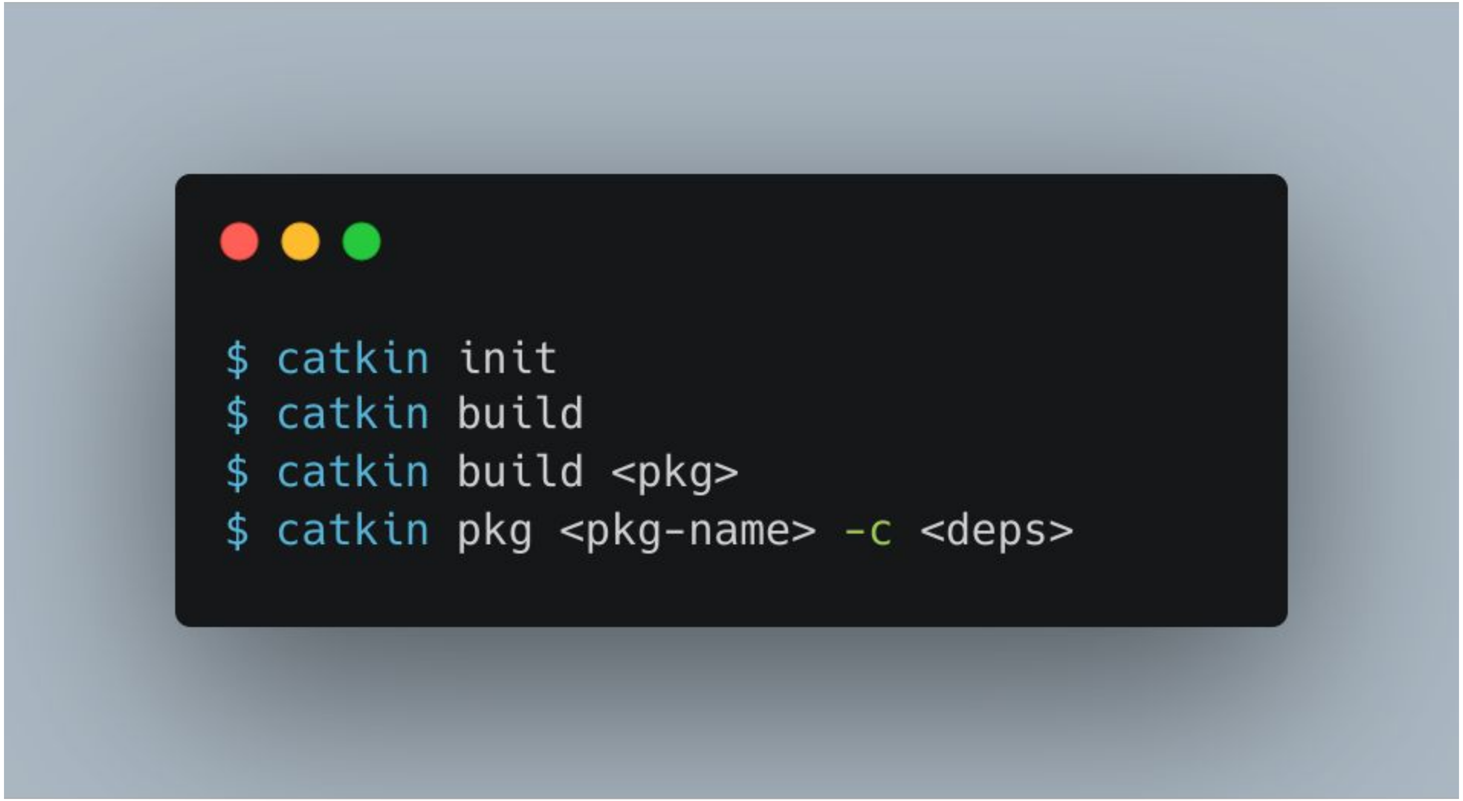
- A catkin workspace is a folder where you modify, build, and install catkin packages.
  - **build**
  - **devel**
  - **install**
  - **src**



```
└── catkin_ws
    ├── build
    ├── devel
    ├── install
    └── src
```

Source: http://wiki.ros.org/catkin/workspaces#Install_Space

Catkin Workspace



```
$ catkin init
$ catkin build
$ catkin build <pkg>
$ catkin pkg <pkg-name> -c <deps>
```

Source: https://catkin-tools.readthedocs.io/en/latest/cheat_sheet.html

Disclaimer

catkin_make vs catkin build

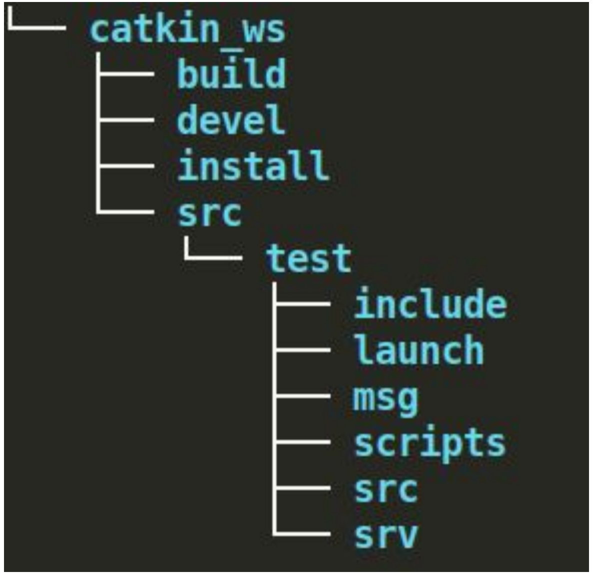
“The main difference is the isolated environment that you get with catkin build.

This makes the whole build configuration much more compartmentalized and robust to changes in the configuration (add/remove package, modify a cmake variable etc.)”

Source:
https://catkin-tools.readthedocs.io/en/latest/migration.html#:~:text=Important%20Distinctions%20between%20catkin_make%20and%20catkin%20build%20%C2%B6,build%2Dtime%20cross%2Dtalk.
https://robotics.stackexchange.com/questions/16604/ros-catkin-make-vs-catkin-build

Package structure

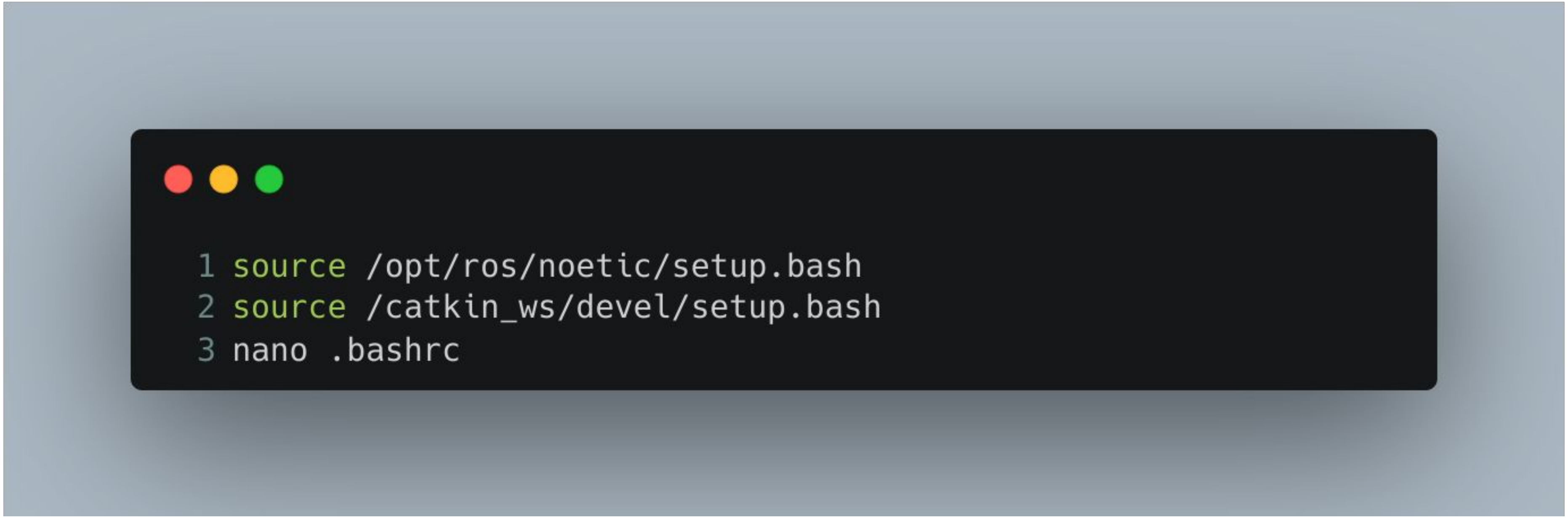
- src
    - Package
        - src
        - include
        - msg
        - scripts
        - srv
        - launch
        - ….



```
└── catkin_ws
    ├── build
    ├── devel
    ├── install
    └── src
        └── test
            ├── include
            ├── launch
            ├── msg
            ├── scripts
            ├── src
            └── srv
```

Source: https://russianblogs.com/article/3420187481/

Source the workspace



```
source /opt/ros/noetic/setup.bash
source /catkin_ws/devel/setup.bash
nano .bashrc
```

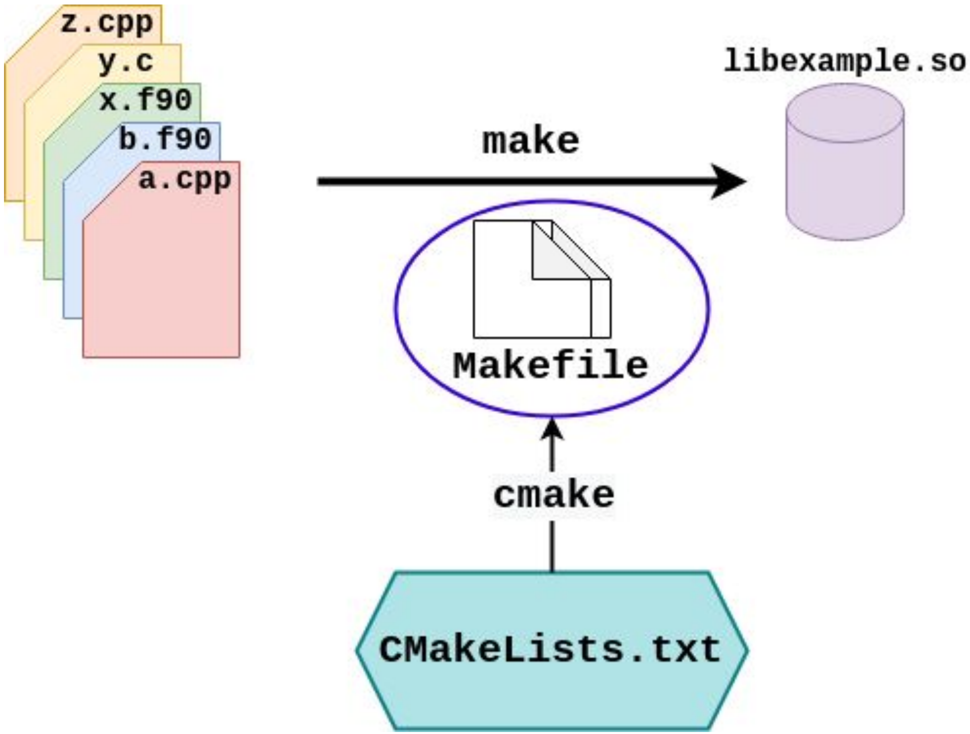
Break (5 minutes)

ROS workspace (Exercise 1)

- Go to the home directory for your user
- Create a directory called labs_ws and inside it a folder called src
- Go inside this directory
- Initiate the catkin workspace
- Go inside labs_ws/src
- Create new package called lab0 without dependencies
- Build the catkin workspace
- What is inside the package folder?
- Do you have any errors?

CMake list

- What is CMake?



C++ & CMake recap (Exercise2)

- Create a header file that contains definition of two function add, sub
- Create a source file that implements these two functions
- Create a Cmake file that build them
- Build them

Source: https://www.andygibson.net/blog/programming/from-java-to-c-project-setup/

CMake list

- Let’s go inside the CMakeLists.txt in ROS

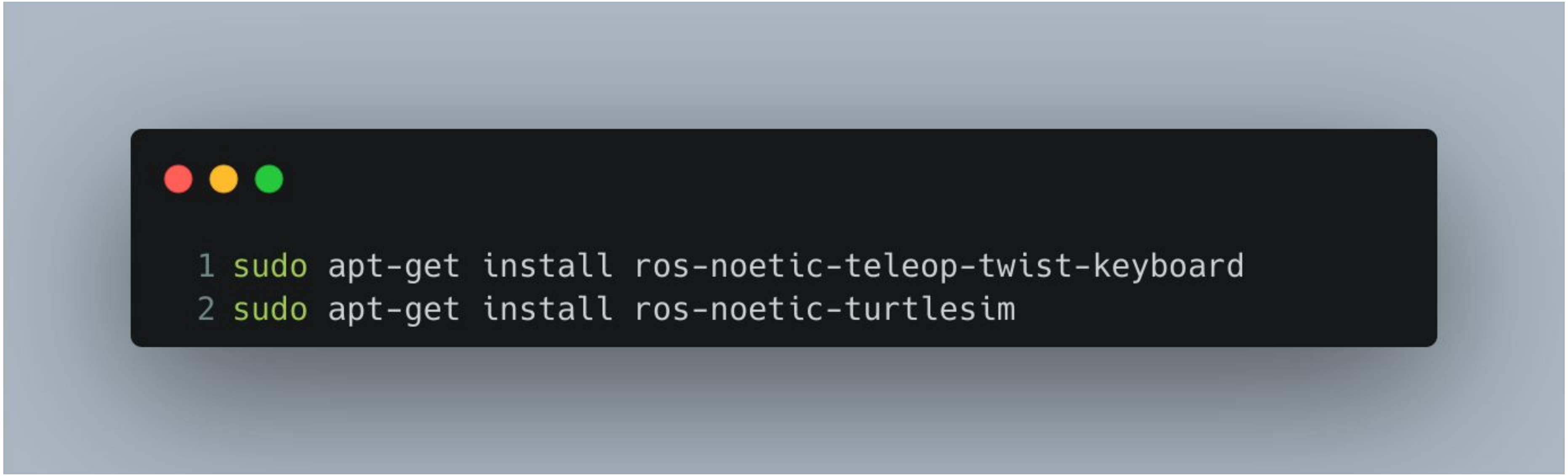
Source: http://wiki.ros.org/catkin/CMakeLists.txt
https://github.com/ros/catkin_tutorials/blob/master/create_package_pubsub/catkin_ws/src/beginner_tutorials/CMakeLists.txt

Package.xml

- What is package.xml file?

  - Just metadata
    - “This file defines properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other catkin packages”
  - Let’s go inside the package.xml

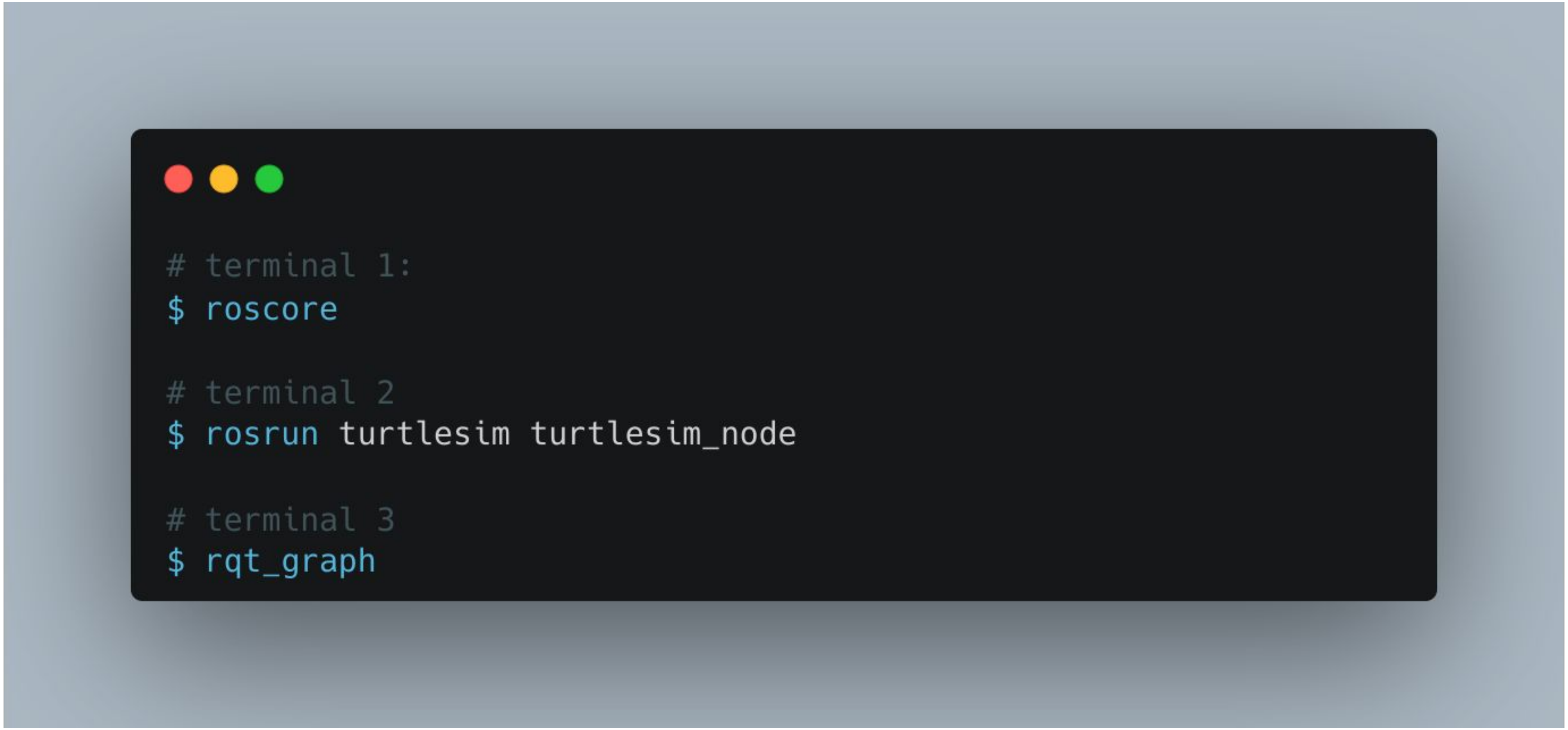
Turtlesim example: Install packages



```
sudo apt-get install ros-noetic-teleop-twist-keyboard
sudo apt-get install ros-noetic-turtlesim
```

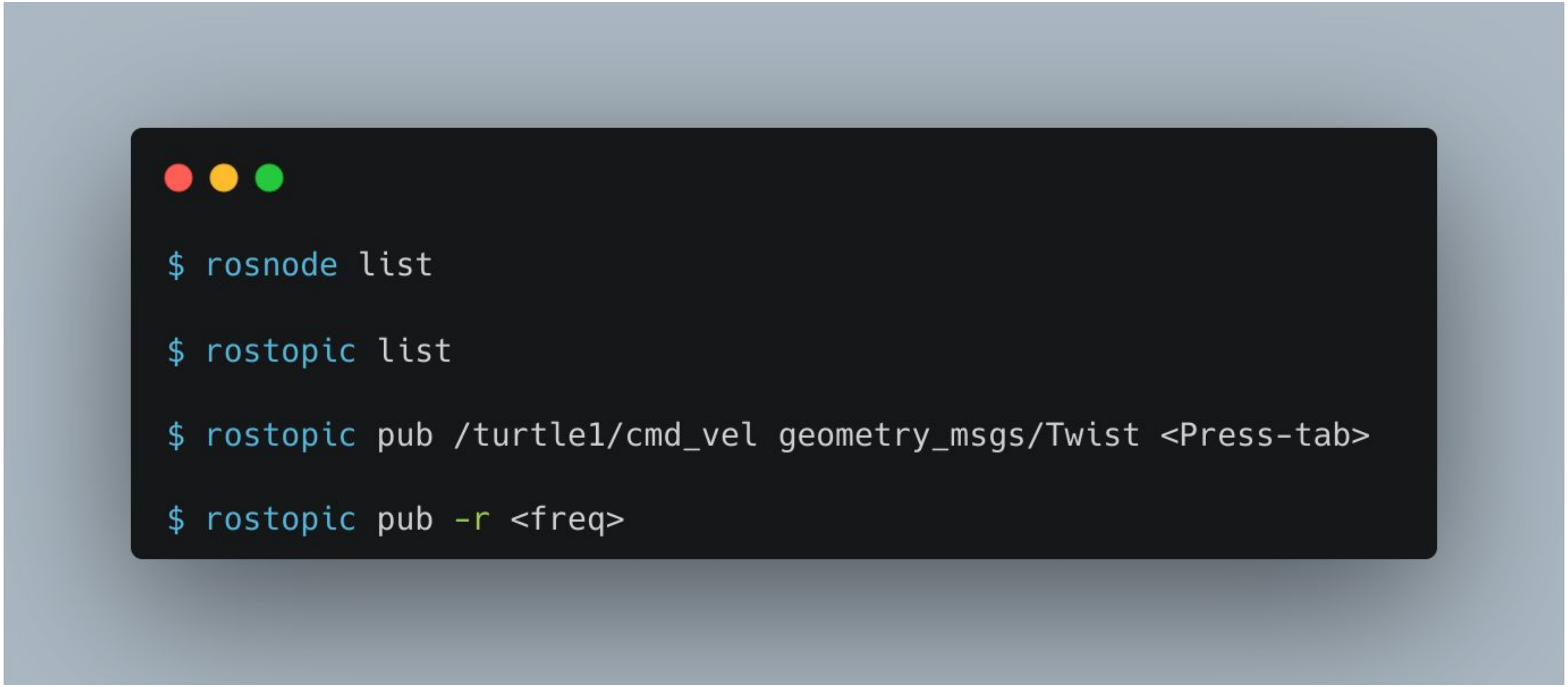
Source: http://wiki.ros.org/teleop_twist_keyboard
http://wiki.ros.org/turtlesim

Turtlesim example: Run!



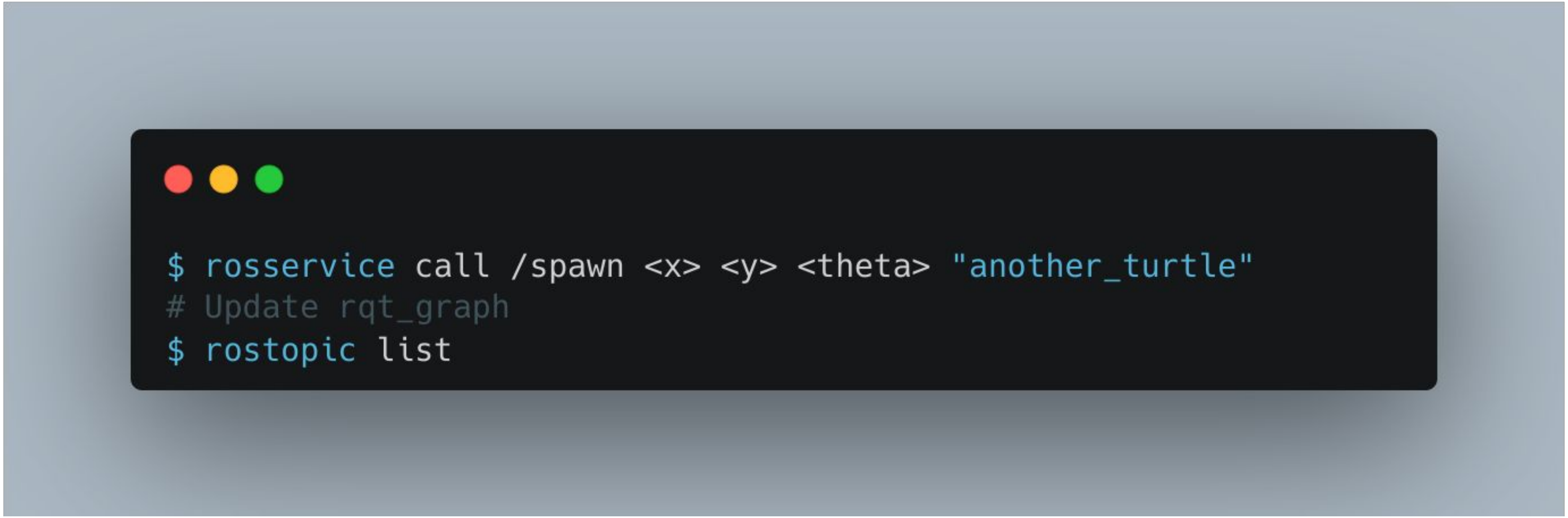
```
# terminal 1:
$ roscore

# terminal 2
$ rosrun turtlesim turtlesim_node

# terminal 3
$ rqt_graph
```

Turtlesim example: Play!



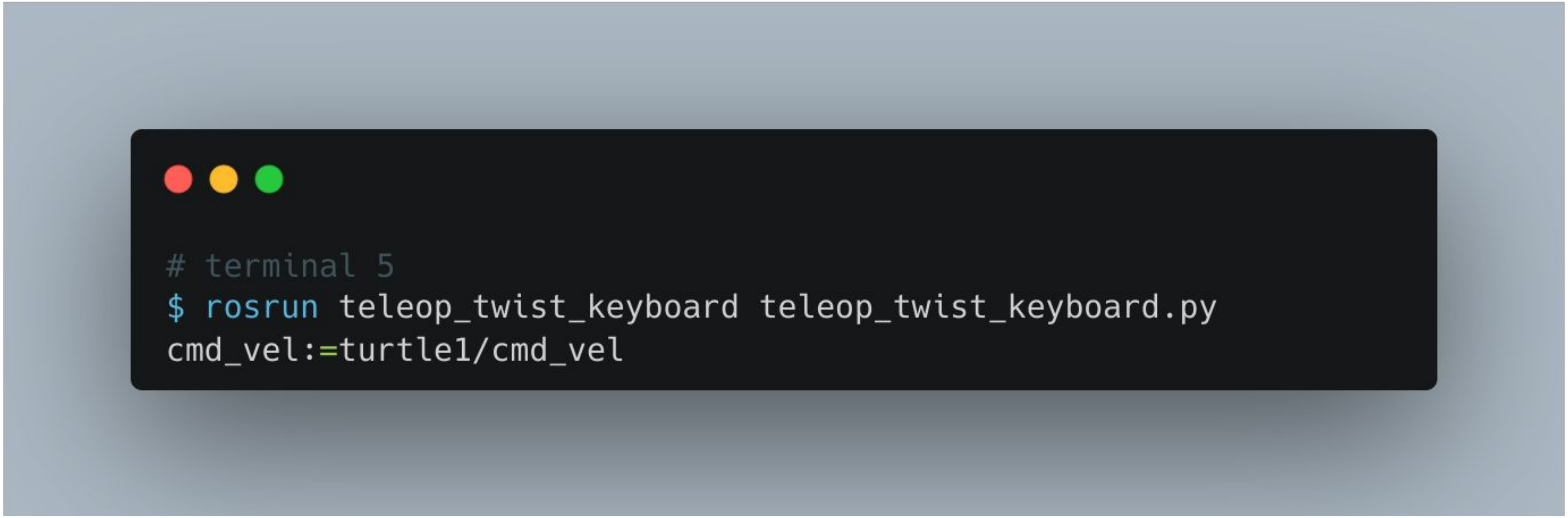
```
$ rosnode list

$ rostopic list

$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist <Press-tab>

$ rostopic pub -r <freq>
```

Turtlesim example: Play more!



```
$ rosservice call /spawn <x> <y> <theta> "another_turtle"
# Update rqt_graph
$ rostopic list
```

Turtlesim example: keyboard control!



```
# terminal 5
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py
cmd_vel:=turtle1/cmd_vel
```

Source
https://answers.ros.org/question/283937/kinetic-turtlesim-and-teleop_key-change-keyboard-key-input/

What is ROS launch?

- “is a tool for easily launching multiple ROS nodes”

- Let us create a launch file to launch everything! **(Exercise 3)**

ROS msg

- std_msgs
- geometry_msgs
- Custom messages **(Exercise 4)**

ROS srv

- Custom services **(Exercise 5)**

Exercise at home (If you like!)

- Understand how to make publisher and subscriber using python or/and C++
- Create a publisher that publishes to control the turtle in turtlesim the way you like
- Spawn multiple turtles
- Run multiple publishers that each one control one of the turtles
- Create a launch file to launch everything

Sources: http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29
http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

Any Questions?