

---

# Intelligent Robots Practice

[LAB] ROS 소개 및 설치

---

Chungbuk National University, Korea  
Intelligent Robots Lab. (IRL)

Prof. Gon-Woo Kim

# 준비사항

---

- 노트북
  - 인터넷 사용가능여부
  - HDD용량 30GB 이상 확보 권장
- Ubuntu 16.04 LTS 설치  
(단독, 듀얼 부팅 사용환경 권장 – 가상 머신 x)



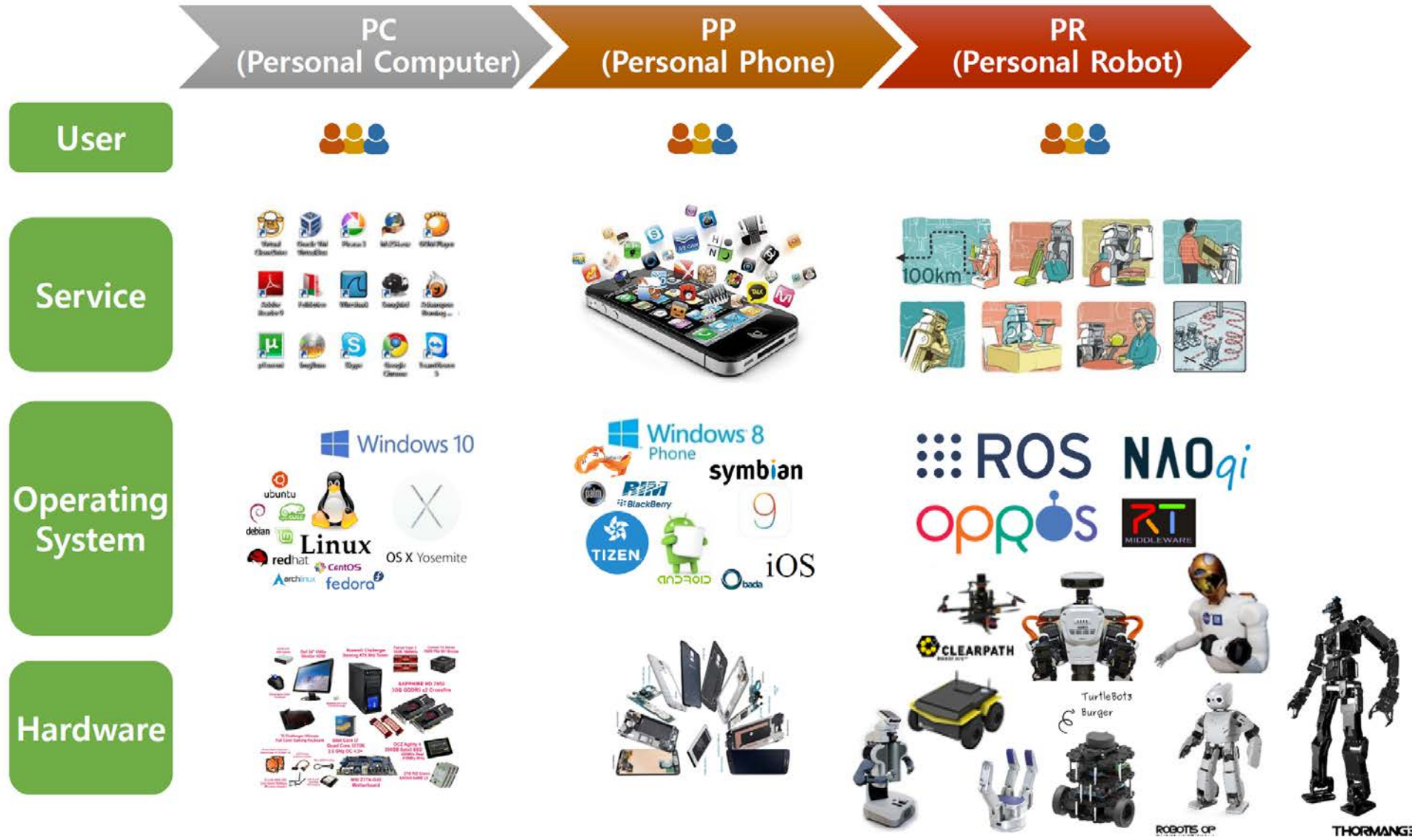
# Overview

---

- ROS 란?
- ROS 설치
- 개발 환경 구축
- ROS 파일 시스템 구조
- ROS 도구 및 명령어



# ROS란?



# ROS란?

---

- ROS 10 Year Montage



# ROS란?

---

- ROS Wiki 정의

**ROS** is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

<http://www.ros.org/wiki/>



Intelligent Robots Lab.  
Chungbuk National University

# ROS란?

- Robot Operation System(ROS)
  - Robot software를 개발하기 위한 framework
  - ROS wiki에서는 Meta-operation system이라고 명명.
  - Node 단위의 분산 프레임워크로 구성되어 코드의 재사용 및 협업 용이(모듈화)
  - Node간 message 형식만 맞으면 다양한 언어 사용 가능
  - 현재 Melodic Release 및 ROS 2개발이 진행되고 있지만, 실습에서는 ROS kinetic사용



# ROS란?

- ROS의 구성

Editable file: [https://docs.google.com/drawings/d/1N3skvSyzZi\\_dWzCqWfEsNd3gtwTNGFDwpm\\_TM\\_Vk2s/edit](https://docs.google.com/drawings/d/1N3skvSyzZi_dWzCqWfEsNd3gtwTNGFDwpm_TM_Vk2s/edit)  
Please contribute by leaving comments or by asking for the writable access right and edit.  
Copyrights should follow that of [ros.org](http://ros.org), where this file is supposed to be shared on primarily.  
Originally made by Isaac I. Y. Saito on 3/20/2016

client (Language) Layer	roscpp	rospy	roslisp	simulation			gazebo_ros_pkgs	stage_ros
	rosjava	roslibjs						
ROBOTICS APPLICATION	MoveIt!	navigation	executive_smach (teleop pkgs)	descartes	rospieex	ar_track_alvar		
				rocon	mapviz	people		
ROBOTICS APPLICATION Framework	dynamic_reconfi gure	robot_localiza tion	robot_pose_ekf	industrial_core	robotwebtools	ros_realtime	mavros	
	tf	robot_state_pub lisher	robot_model	ros_control	calibration	octomap_mapping		
	vision_opencv	image_pipeline	laser_pipeline	perception_pcl	laser_filters	ecto		
communication Layer	common_msgs	rosbag	actionlib	pluginlib	SOFTWARE DEVELOPMENT TOOLS		RViz	rqt
	rostopic	rosservice	roslaunch	roscnode			wstool	rospack
	ros_console	rosparam	rosmaster	rosout			catkin	roscdep
Hardware Interface Layer	camera_drivers	(GPS/IMU drivers)	(range finder drivers)	(3D sensor drivers)	diagnostics			
	joystick_driver s	audio_common	(Force/Torque sensor drivers)	(Power supply drivers)	roscserial	(Ethercat drivers)	ros_canopen	





# ROS란?

---

- ROS의 철학

- P2P(Peer-to-peer)
  - 서로 연결되어 메시지를 주고받는 프로그램들로 구성되어 있음
  - 작은 시스템에서는 더 복잡하지만 큰 시스템에서 확장성이 좋음
- Tool의 집합
  - 표준화된 통합환경이 없음, 별도의 프로그램들로 구성.
  - 사용자가 작업환경에 맞는 프로그램을 선택가능
- 언어 독립성
  - C++, Python, Lisp 구현되어 있고
  - Java, Lua, MATLAB등 여러환경에서 개발가능
- 오픈소스
  - 핵심부분이 상업적, 비상업적인 이용이 모두 가능
  - 많은 사용자들이 개발한 Package 존재



# ROS란?

---

- ROS의 특징: 통신 기능

- 노드 간 데이터 통신을 제공
- 통상적 미들웨어로 지칭되는 메시지 전달 인터페이스 지원
- 메시지 파싱 기능
  - 로봇 개발 시에 빈번히 사용되는 통신 시스템 제공
  - 캡슐화 및 코드 재사용을 촉진하는 노드들간의 메시지 전달 인터페이스
- 메시지의 기록 및 재생
  - 노드간 송/수신되는 데이터인 메시지를 저장하고 필요시에 재사용 가능
  - 저장된 메시지를 기반으로 반복적인 실험 가능, 알고리즘 개발에 용이함
- 메시지 사용으로 인한 다양한 프로그래밍 언어 사용 가능
  - 노드간의 데이터 교환이 메시지를 사용하기 때문에 각 노드는 서로 다른 언어로 작성 가능
  - 클라이언트 라이브러리: roscpp, rospy, roslisp, rosjava, roslua, roscs, roseus, PhaROS, rosR
- 분산매개변수시스템
  - 시스템에서 사용되는 변수를 글로벌 키값으로 작성하여 공유 및 수정하여 실시간으로 반영



# ROS란?

- ROS의 특징: 로봇 관련 다양한 기능

- 로봇에 대한 표준 메시지 정의

- 카메라, IMU, 레이저 등의 센서 / 오도메트리, 경로 및 지도 등의 내비게이션 데이터 등의 표준 메시지를 정의하여 모듈화, 협업 작업을 유도, 효율성 향상

- 로봇 기하학 라이브러리

- 로봇, 센서 등의 상대적 좌표를 트리화 시키는 TF 제공

- 로봇 기술 언어

- 로봇의 물리적 특성을 설명하는 XML 문서 기술

- 진단 시스템

- 로봇의 상태를 한눈에 파악할 수 있는 진단 시스템 제공

- 센싱/인식

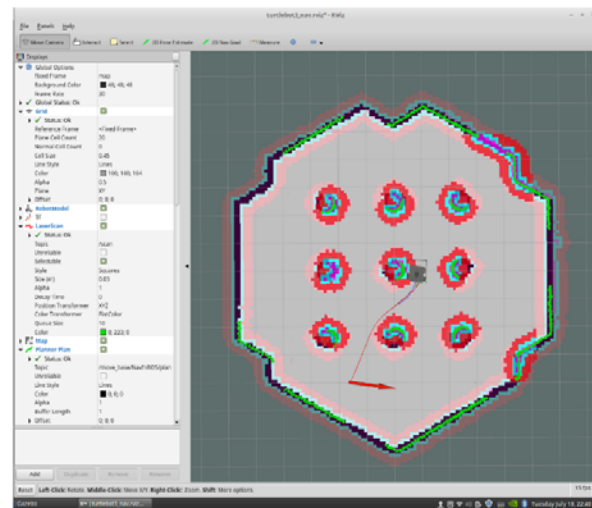
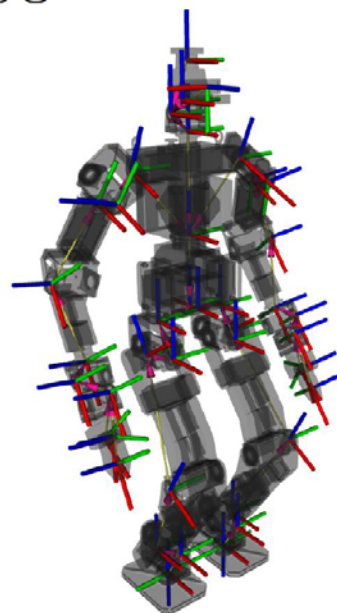
- 센서 드라이버, 센싱/인식 레벨의 라이브러리 제공

- 내비게이션

- 로봇에서 많이 사용되는 로봇의 포즈(위치/자세) 추정, 지도내의 자기 위치 추정 제공
- 지도 작성에 필요한 SLAM, 작성된 지도 내에서 목적지를 찾아가는 Navigation 라이브러리를 제공

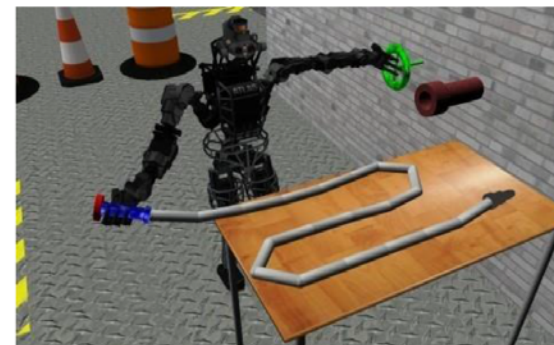
- 매니퓰레이션

- 로봇 암에 사용되는 IK, FK 는 물론 응용단의 Pick and Place 를 지원하는 다양한 Manipulation 라이브러리 제공
- GUI 형태의 매니퓰레이션 Tools 제공(MoveIt!)



# ROS란?

- ROS의 특징: 로봇 관련 다양한 기능
  - 로봇 개발에 필요한 다양한 개발 도구를 제공
  - 로봇 개발의 효율성 향상
  - Command-Line Tools
    - GUI 없이 ROS에서 제공되는 명령어로만 로봇 액세스 및 거의 모든 ROS 기능 소화
  - RViz
    - 강력한 3D 시각화툴 제공
    - 레이저, 카메라 등의 센서 데이터를 시각화
    - 로봇 외형과 계획된 동작을 표현
  - RQT
    - 그래픽 인터페이스 개발을 위한 Qt 기반 프레임 워크 제공
    - 노드와 그들 사이의 연결 정보 표시(rqt\_graph)
    - 인코더, 전압, 또는 시간이 지남에 따라 변화하는 숫자를 플로팅(rqt\_plot)
    - 데이터를 메시지 형태로 기록하고 재생(rqt\_bag)
  - Gazebo
    - 물리 엔진을 탑재, 로봇, 센서, 환경 모델 등을 지원, 3차원 시뮬레이터
    - ROS와의 높은 호환성



# ROS란?

## • ROS Distribution

- 2010.03.02 ROS Box Turtle을 시작으로 여러 배포판이 배포됨
- LTS는 배포를 기점으로 5년간 지원되며, 현재 지원되는 버전은 Melodic, Kinetic
- 2020년 Noetic버전이 배포될 예정.

Distro	Release date	Poster	Turtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys	May, 2020 (planned, see <a href="#">Upcoming Releases</a> )	TBA	TBA	May, 2025 (planned)
ROS Melodic Morenia (Recommended)	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015
ROS Groovy Galapagos	December 31, 2012			July, 2014
ROS Fuerte Turtle	April 23, 2012			--

# ROS란?

- ROS vs ROS2?

Features	ROS	ROS2
<b>Platforms</b>	Tested on Ubuntu Maintained on other Linux flavors as well as OS X	ROS2 is currently being CI tested and supported on Ubuntu Xenial, OS X El Capitan as well as Windows 10
<b>C++</b>	C++03 // don't use C++11 features in its API	Mainly uses C++11 Start and plan to use C++14 & C++17
<b>Python</b>	Target Python 2	>= Python 3.5
<b>Middleware</b>	Custom serialization format (transport protocol + central discovery mechanism)	Currently all implementations of this interface are based on the DDS standard.
<b>Unify duration and time types</b>	The duration and time types are defined in the client libraries, they are in C++ and Python	In ROS2 these types are defined as messages and therefore are consistent across languages.
<b>Components with life cycle</b>	In ROS every node usually has its own main function.	The life cycle can be used by tools like roslaunch to start a system composed of many components in a deterministic way.
<b>Threading model</b>	In ROS the developer can only choose between single-threaded execution or multi-threaded execution.	In ROS2 more granular execution models are available and custom executors can be implemented easily.
<b>Multiple nodes</b>	In ROS it is not possible to create more than one node in a process.	In ROS2 it is possible to create multiple nodes in a process.
<b>roslaunch</b>	In ROS roslaunch files are defined in XML with very limited capabilities.	In ROS2 launch files are written in Python which enables to use more complex logic like conditionals etc.



# ROS란?

- ROS Concept
  - 소프트웨어의 모듈화



=



<PC의 하드웨어 모듈>



인지

판단

제어

위치인식

차선인식

신호등인식

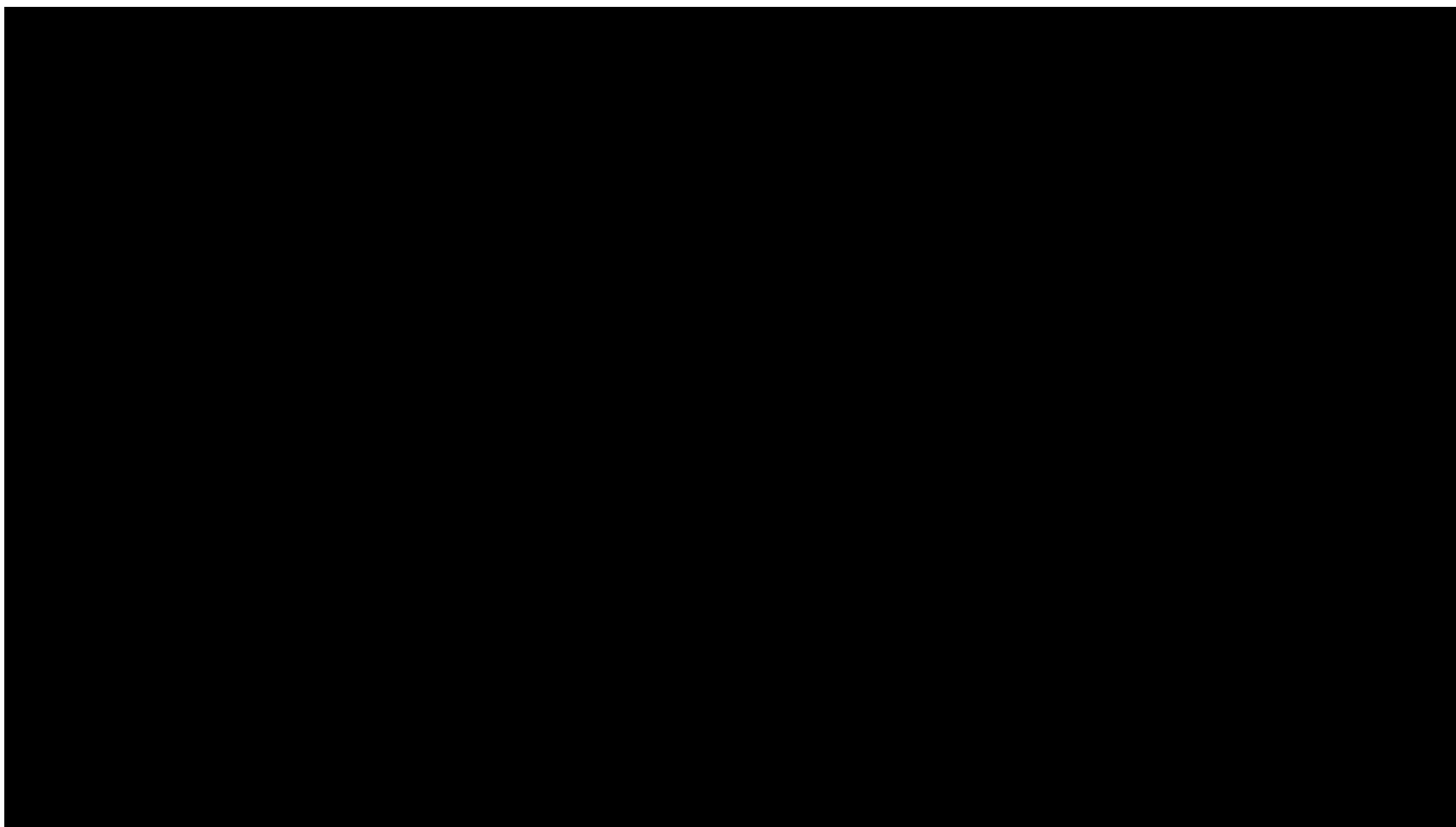
...

<자율주행 차량의 sw모듈>

# ROS란?

---

- ROS Navigation Stack





# ROS 설치 방법

---

- ROS Kinetic 설치

- source.list에 package.ros.org를 추가.

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

- apt-key 설정, apt update

```
$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

```
$ sudo apt-get update
```

- install

```
$ sudo apt-get install ros-kinetic-desktop-full
```



# ROS 설치 방법

---

- 환경설정

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

- Dependency 설치

```
$ sudo apt install python-rosdep python-rosinstall python-rosinstall-generator  
python-wstool build-essential
```



# ROS 설치 방법

- 설치 확인 방법

\$ `roscore`

- 사진과 같은 결과 화면 확인

```
a@a:~$ roscore
... logging to /home/a/.ros/log/ec0568d0-82a7-11ea-b75f-f406693830a1/roslaunch-a-15676.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:34439/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [15686]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to ec0568d0-82a7-11ea-b75f-f406693830a1
process[rosout-1]: started with pid [15699]
started core service [/rosout]
```



# 개발 환경 구축

---

- ROS IDE
- 다양한 IDE 사용가능 (<http://wiki.ros.org/IDEs>)
  - Eclipse
  - CLion
  - CodeBlocks
  - Emacs
  - Vim
  - NetBeans
  - QtCreator – (recommended)
  - PyCharm (community edition)
  - KDevelop
  - RoboWare Studio
  - Visual Studio Code (VSCode) - (recommended)



# 개발 환경 구축

---

- Qt creator vs VSCode ??
    - C++ 개발에 유리한 Qt creator (gdb 지원)
    - But, python intellisense plug-in 개별 설치 및 설정 필요
    - 사용자가 많고 다양한 Extension 지원하는 VSCode
    - But, GDB Debugger 및 ros catkin\_make 설정 필요
- C++ 개발 - Qt creator, Python - vscode



Code less.  
Create more.  
Deploy everywhere.



Visual Studio Code



Intelligent Robots Lab.  
Chungbuk National University

# 개발 환경 구축

---

- Qt creator 설치

- Qt creator xenial online installer download.

<https://qtcreator-ros.datasys.swri.edu/downloads/installers/xenial/qtcreator-ros-xenial-latest-online-installer.run>

- 이전 버전 qtcreator 제거

```
$ sudo apt install ppa-purge  
$ sudo ppa-purge -o beineri  
$ sudo ppa-purge levi-armstrong/qt-libraries-xenial  
$ sudo ppa-purge levi-armstrong/ppa
```



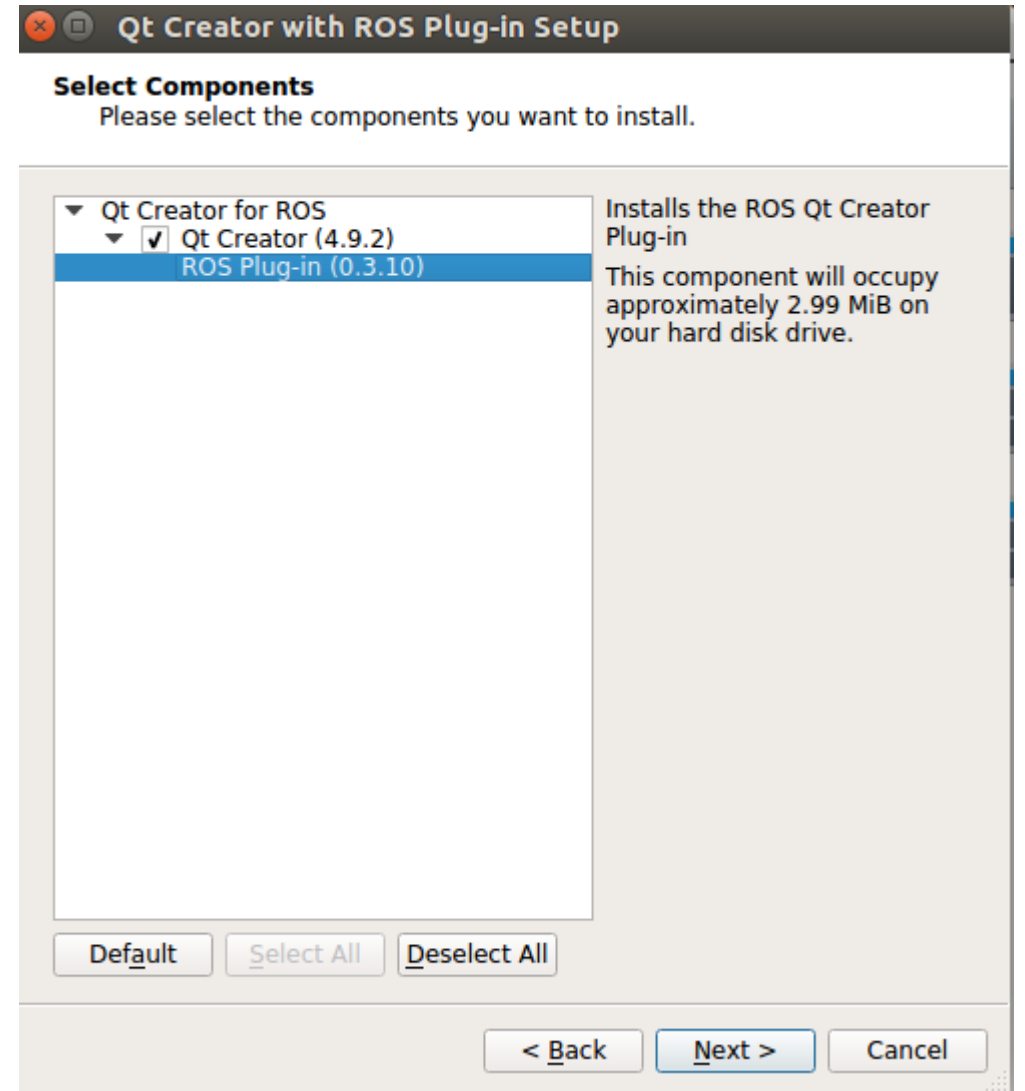
# 개발 환경 구축

- Qt creator 설치

```
$ chmod u+x qtcreator-ros-xenial-latest-online-installer.run
```

```
$ ./qtcreator-ros-xenial-latest-online-installer.run
```

- 설치시 오른쪽 화면과 같이 체크(ROS Plug-in포함)

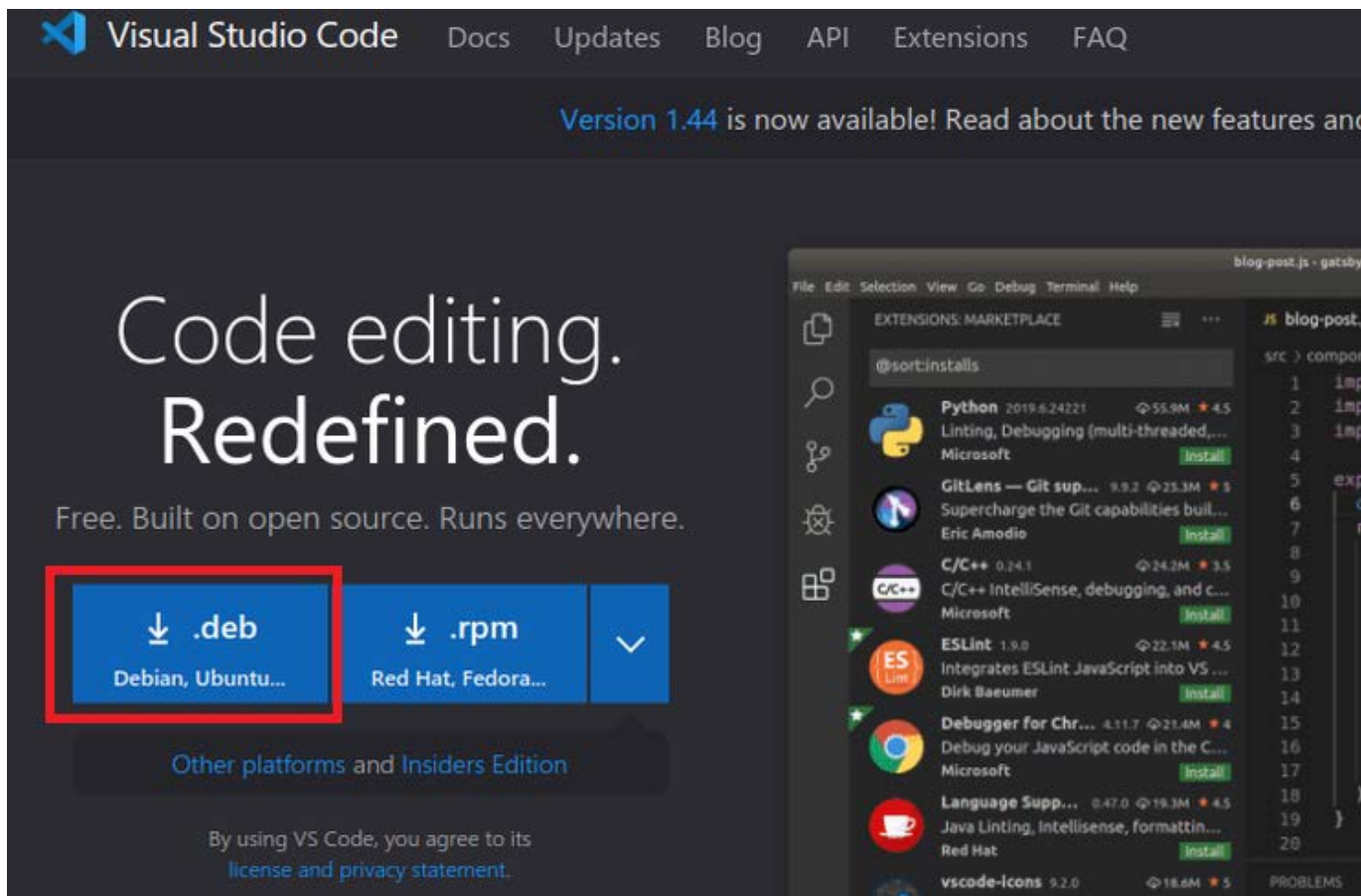


# 개발 환경 구축

- VS code 설치
  - .deb 파일 다운로드 후 설치

<https://code.visualstudio.com/>

```
$ sudo dpkg -i <다운받은 .deb file>
```





# 개발 환경 구축

---

- VS code – ROS extension
  - Ctrl + shift + x 로 extension으로 진입
  - “ROS” 검색 후 설치

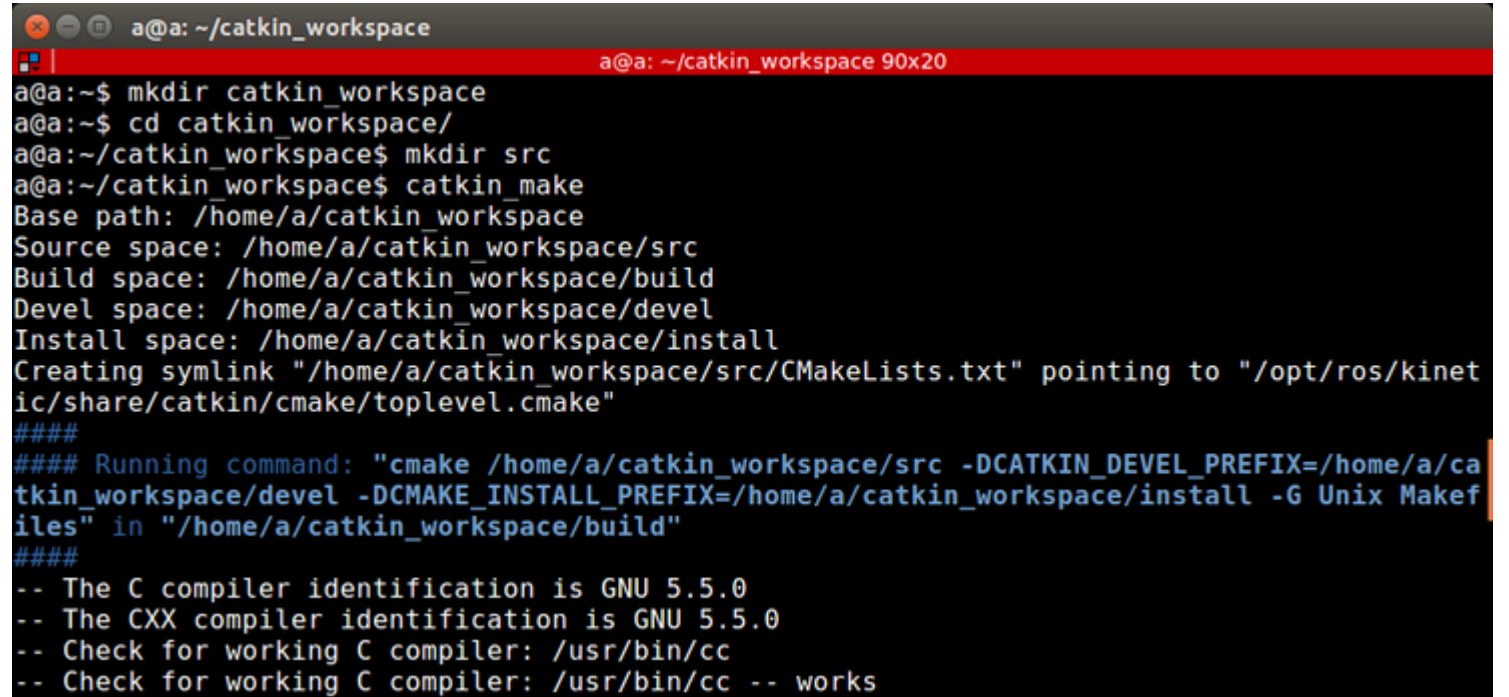


# Create ROS workspace

- Create ROS Workspace

```
$ mkdir <name of workspace>
$ cd <name of workspace>
$ mkdir src
$ catkin_make
```

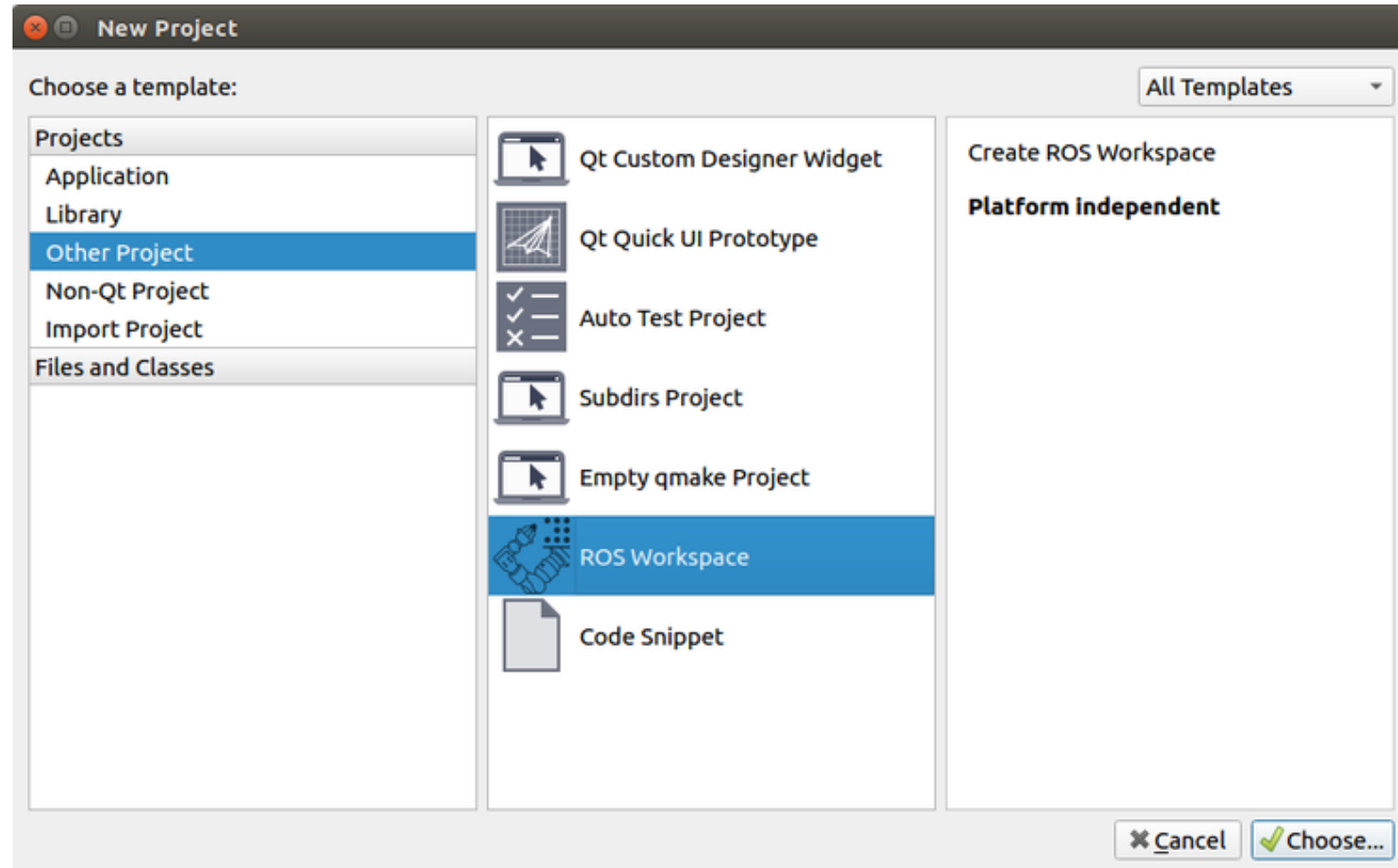
- Workspace의 이름은 변경가능



```
a@a: ~/catkin_workspace
a@a: ~/catkin_workspace 90x20
a@a:~$ mkdir catkin_workspace
a@a:~$ cd catkin_workspace/
a@a:~/catkin_workspace$ mkdir src
a@a:~/catkin_workspace$ catkin_make
Base path: /home/a/catkin_workspace
Source space: /home/a/catkin_workspace/src
Build space: /home/a/catkin_workspace/build
Devel space: /home/a/catkin_workspace/devel
Install space: /home/a/catkin_workspace/install
Creating symlink "/home/a/catkin_workspace/src/CMakeLists.txt" pointing to "/opt/ros/kinetic/share/catkin/cmake/toplevel.cmake"
####
#### Running command: "cmake /home/a/catkin_workspace/src -DCATKIN_DEVEL_PREFIX=/home/a/catkin_workspace/devel -DCMAKE_INSTALL_PREFIX=/home/a/catkin_workspace/install -G Unix Makefiles" in "/home/a/catkin_workspace/build"
####
-- The C compiler identification is GNU 5.5.0
-- The CXX compiler identification is GNU 5.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
```

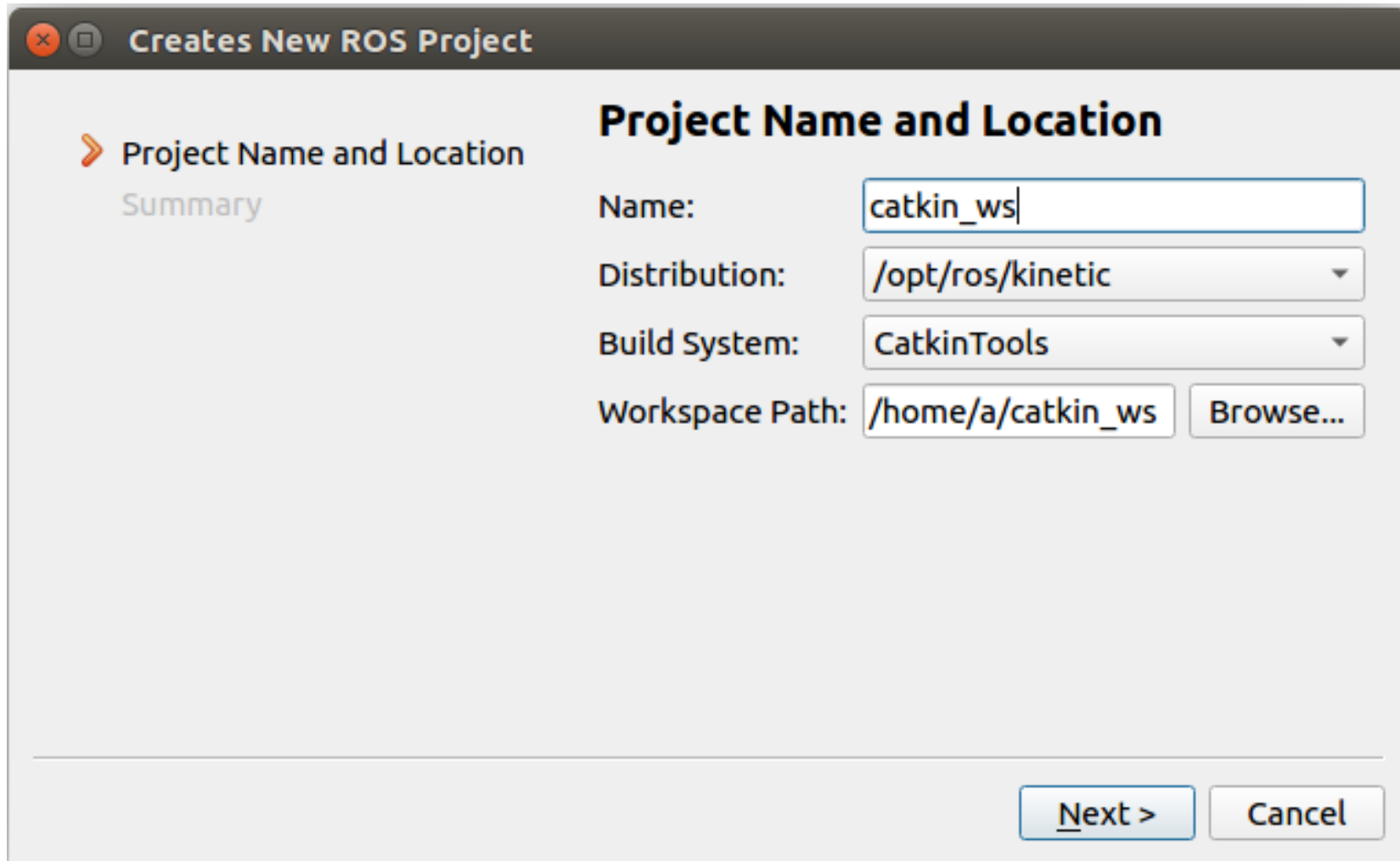
# Create ROS workspace

- Qtcreator – open workspace
  - <File> - <New Project>
  - Other Project – ROS Workspace



# Create ROS workspace

- 열고자 하는 workspace를 <Browse..>에서 선택



**Creates New ROS Project**

**Project Name and Location**

Project Name and Location  
Summary

Name:

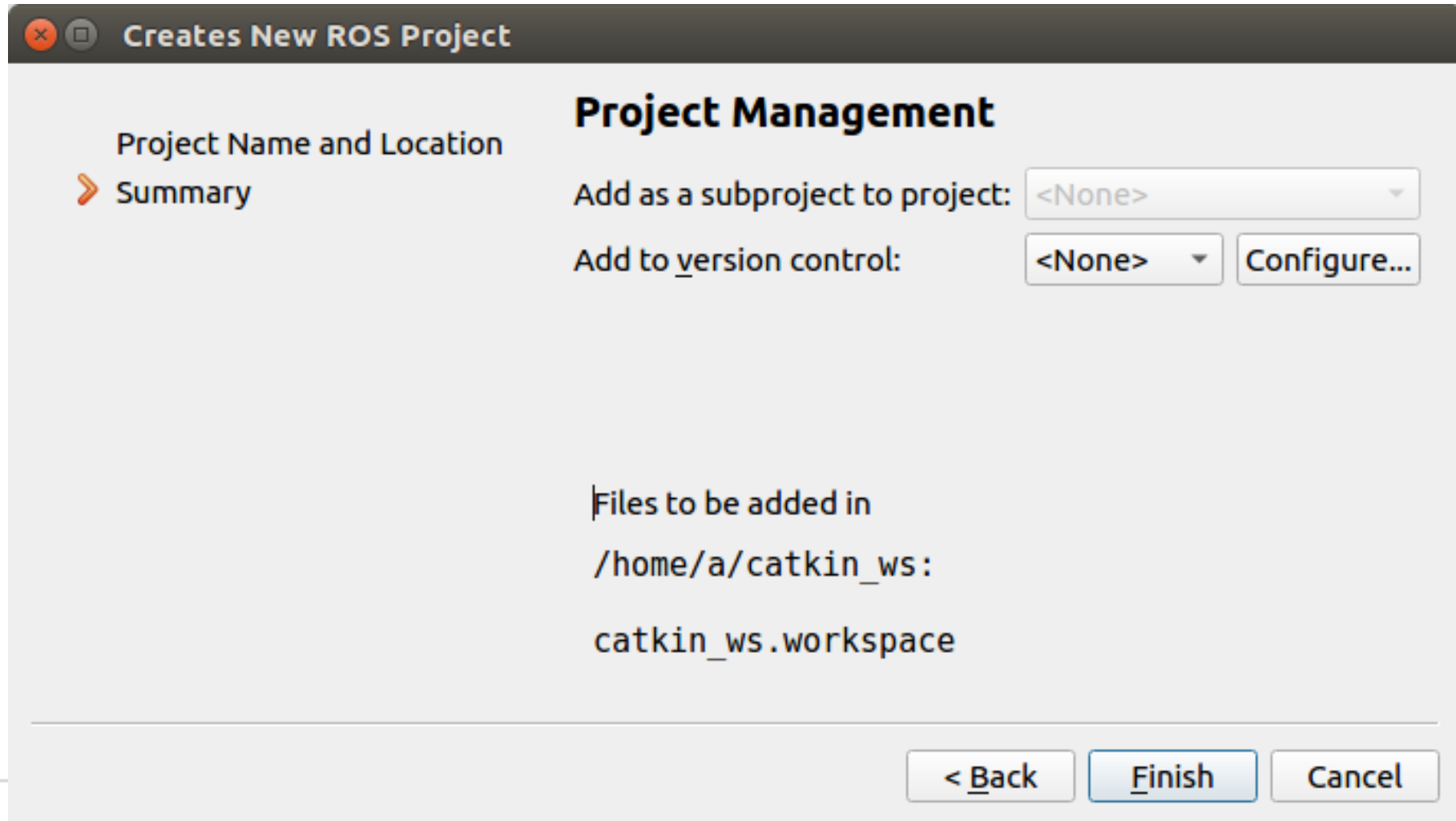
Distribution:

Build System:

Workspace Path:

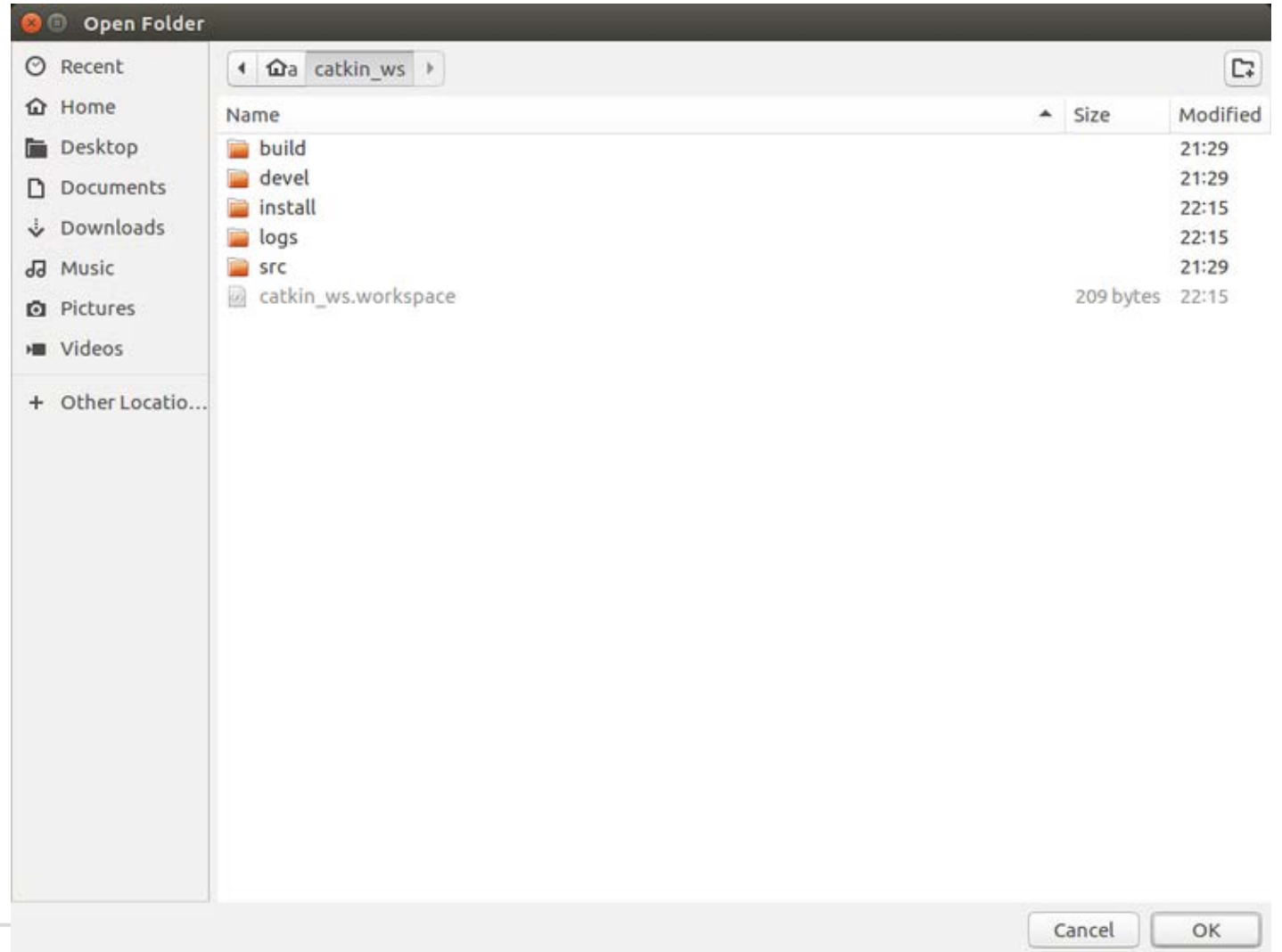
# Create ROS workspace

- <Version control> git... 사용시 선택 가능, 실습에서는 None으로 선택



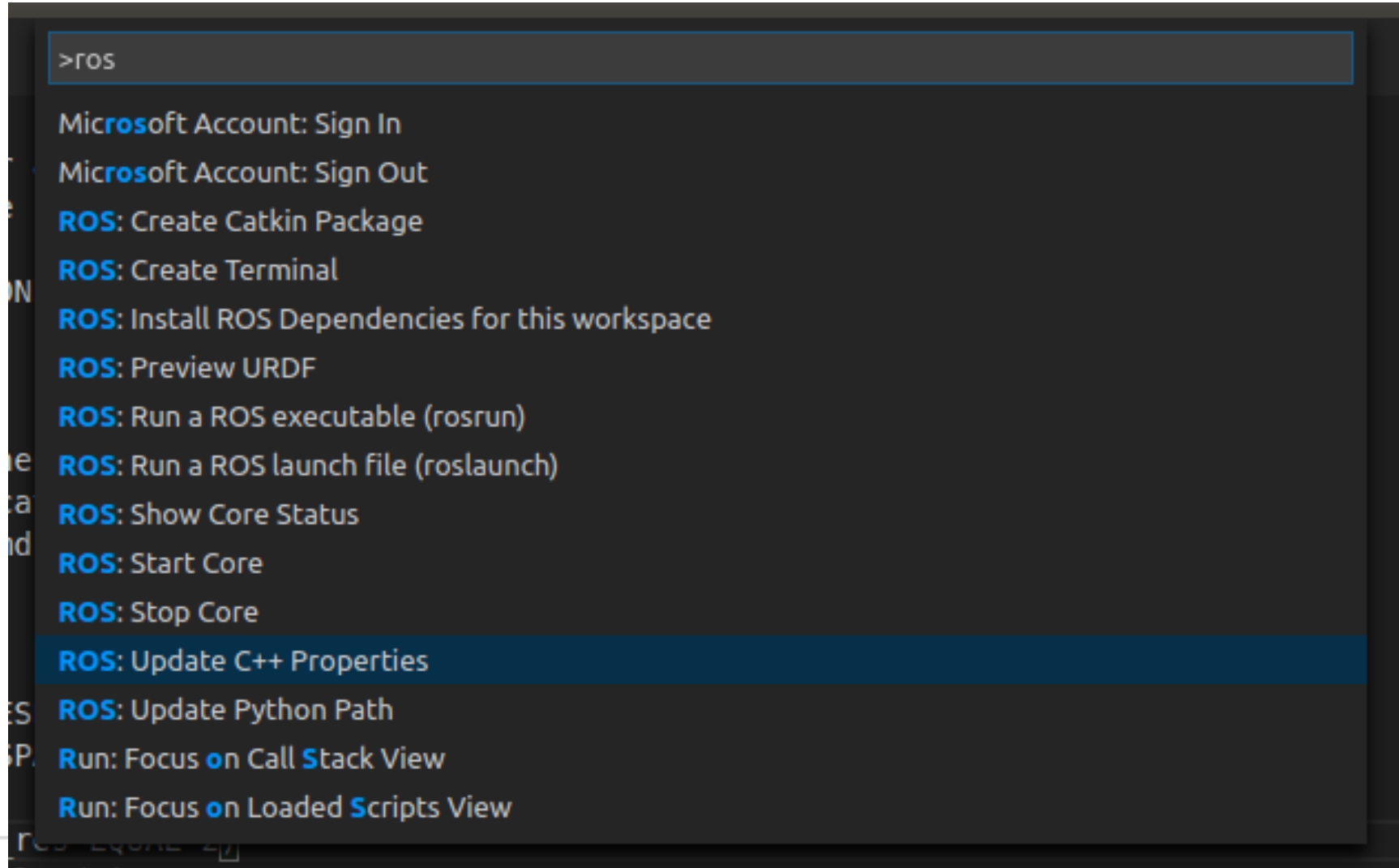
# Create ROS workspace

- VSCode – open work space
  - <ctrl> + <K> , <O>
  - Workspace 선택



# Create ROS workspace

- VSCode – open work space
  - <ctrl> + <Shift> + <p>
  - “Ros” 입력 후 <update c++ properties>
  - <Update python path>



# ROS 파일 시스템 구조

---

- ROS File-system Level

- Packages : ROS software를 구성하는 기본단위. ROS runtime process(Node), 종속 라이브러리, Dataset, 구성 파일 등 이 포함됨. Build, Release 할 수 있는 기본 단위
  - Meta packages : 관련된 Package의 그룹
  - Package Manifests : package.xml, 패키지의 이름, 버전, 설명, 라이선스 정보 및 종속성 및 기타 메타정보를 포함.
  - Message types : ROS message Data 구조 정의
  - Service types : ROS service의 요청 및 응답, 데이터 구조 정의
-



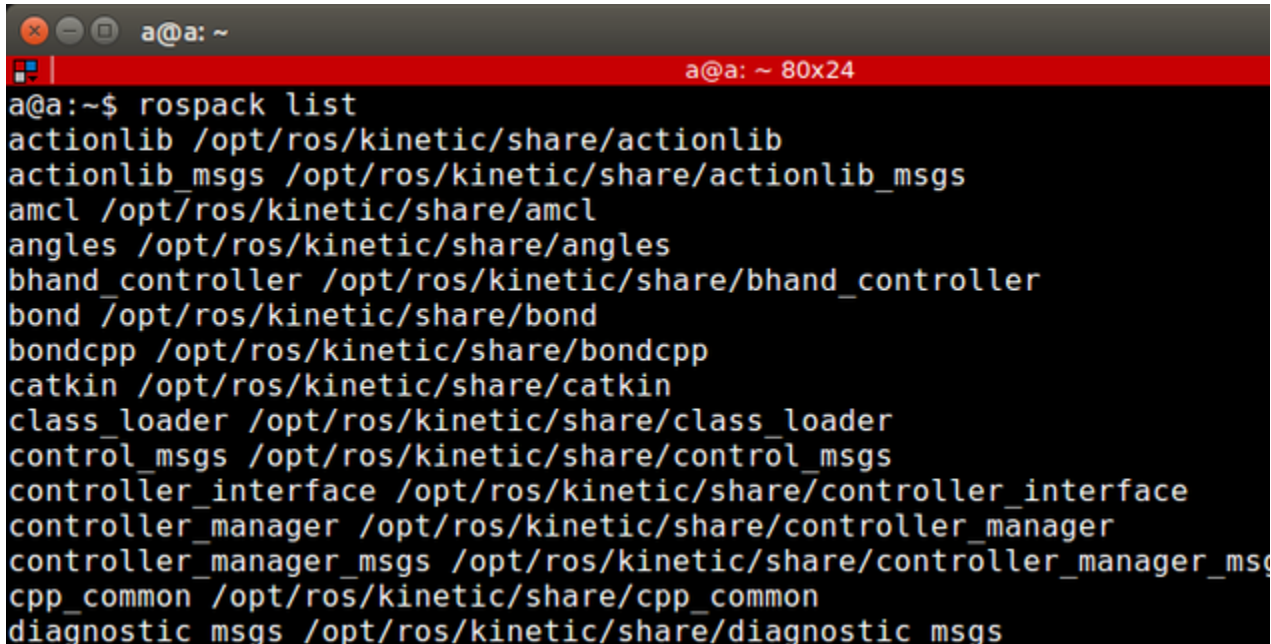
# ROS 파일 시스템 구조

---

- ROS File-system Tools

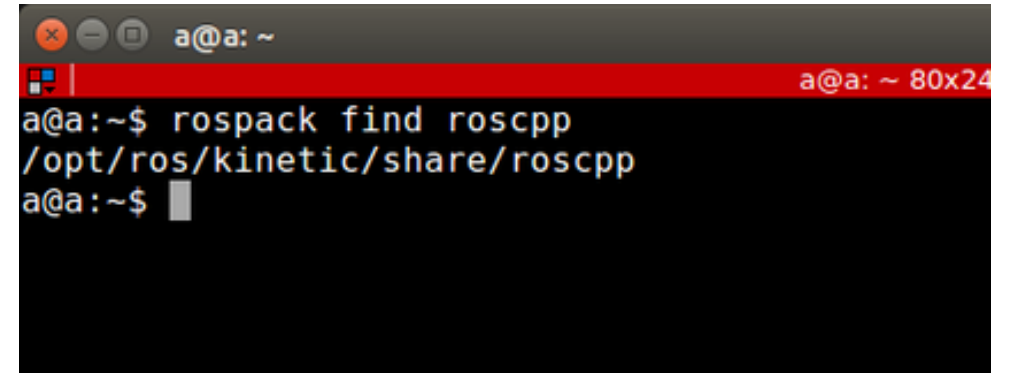
- Rospack – Terminal에서 ls, cd대신에 ros package를 쉽게 찾을 수 있게 함.
  - rospack list : 현재 실행 가능한 package 전체를 보여 줌.

\$ rospack list



```
a@a: ~  
a@a: ~ 80x24  
a@a:~$ rospack list  
actionlib /opt/ros/kinetic/share/actionlib  
actionlib_msgs /opt/ros/kinetic/share/actionlib_msgs  
amcl /opt/ros/kinetic/share/amcl  
angles /opt/ros/kinetic/share/angles  
bhand_controller /opt/ros/kinetic/share/bhand_controller  
bond /opt/ros/kinetic/share/bond  
bondcpp /opt/ros/kinetic/share/bondcpp  
catkin /opt/ros/kinetic/share/catkin  
class_loader /opt/ros/kinetic/share/class_loader  
control_msgs /opt/ros/kinetic/share/control_msgs  
controller_interface /opt/ros/kinetic/share/controller_interface  
controller_manager /opt/ros/kinetic/share/controller_manager  
controller_manager_msgs /opt/ros/kinetic/share/controller_manager_msgs  
cpp_common /opt/ros/kinetic/share/cpp_common  
diagnostic_msgs /opt/ros/kinetic/share/diagnostic_msgs
```

\$ rospack find [package name]

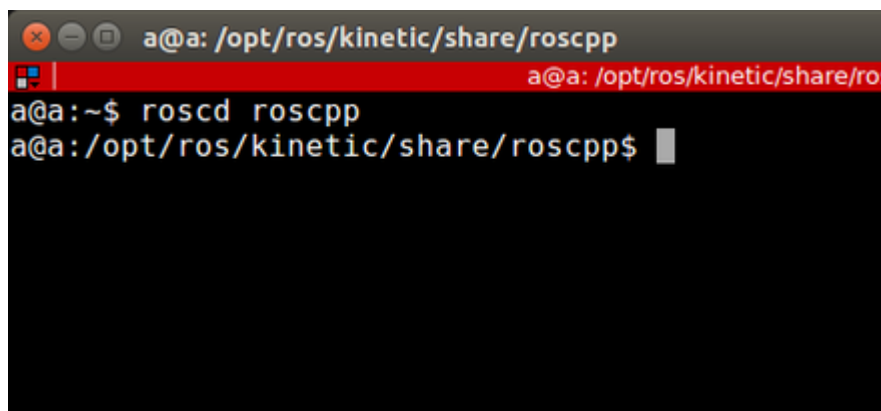


```
a@a: ~  
a@a: ~ 80x24  
a@a:~$ rospack find roscpp  
/opt/ros/kinetic/share/roscpp  
a@a:~$
```

# ROS 파일 시스템 구조

- ROS File-system Tools

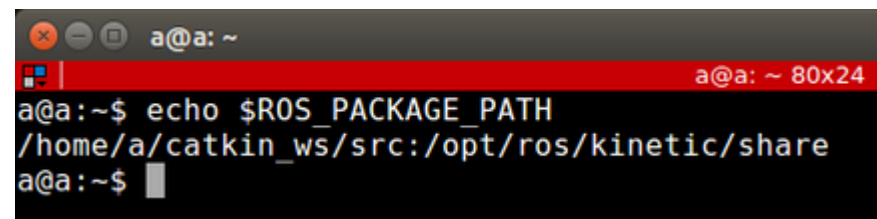
\$ roscd [package name]

A terminal window with a dark background and a red title bar. The title bar contains the text 'a@a: /opt/ros/kinetic/share/roscpp'. The terminal shows the command 'roscd roscpp' being executed, and the prompt changes to 'a@a:/opt/ros/kinetic/share/roscpp\$'.

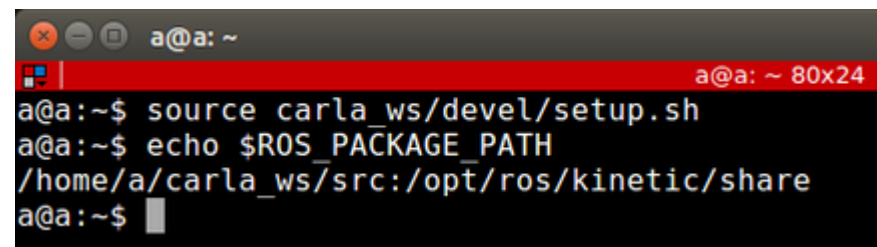
- ROS file system은 ~/.bashrc에 source된 path를 따름.

- Path확인, 추가

\$ echo \$ROS\_PACKAGE\_PATH

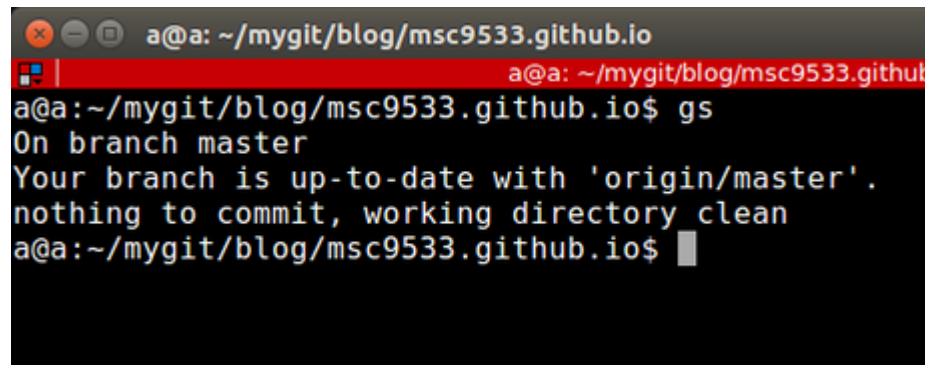
A terminal window with a dark background and a red title bar. The title bar contains the text 'a@a: ~ 80x24'. The terminal shows the command 'echo \$ROS\_PACKAGE\_PATH' being executed, and the output is '/home/a/catkin\_ws/src:/opt/ros/kinetic/share'.

\$ source <workspace>/devel/setup.sh

A terminal window with a dark background and a red title bar. The title bar contains the text 'a@a: ~ 80x24'. The terminal shows the command 'source carla\_ws/devel/setup.sh' being executed, followed by 'echo \$ROS\_PACKAGE\_PATH', and the output is '/home/a/carla\_ws/src:/opt/ros/kinetic/share'.

# ROS 파일 시스템 구조

- .bashrc 를 이용한 ROS\_PACKAGE\_PATH
  - 새로운 터미널이 생성될 때, ~/.bashrc에 정의된 스크립트를 실행함
    - \$ [editor] ~/.bashrc
    - \$ gedit ~/.bashrc
  - Source [path]를 추가,
  - Alias로 단축명령어를 정의 가능



```
a@a: ~/mygit/blog/msc9533.github.io
a@a: ~/mygit/blog/msc9533.github.io$ gs
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
a@a:~/mygit/blog/msc9533.github.io$
```

```
# enable programmable completion features (you
# this, if it's already enabled in /etc/bash.ba
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_compl
    . /usr/share/bash-completion/bash_completic
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
alias eb='gedit ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='catkin_make'
alias seb='source devel/setup.sh'

source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://localhost:11311
export ROS_HOSTNAME=localhost
export TURTLEBOT3_MODEL=burger
# export UE4_ROOT=~/.UnrealEngine_4.18
export UE4_ROOT=~/.UnrealEngine_4.24

# >>> conda initialize >>>
# !! Contents within this block are managed by
__conda_setup="$('/home/a/anaconda3/bin/conda'
```