

ROS2

국립금오공과대학교

수리빅데이터학과

신 승 혁

Contents

1	ROS2 소개	2
2	ROS2 개발환경 구축	4
2.1	개발환경	4
2.2	기본 운영체제 설치	4
2.3	로봇 운영체제 설치	4
3	개발툴 설치	6
4	ROS2 빌드 테스트	6
5	Run Commands 설정	7
6	IDE 설치	8
7	ROS 기본	13
7.1	기본용어	13
8	ROS2와 DDS(Data Distribution Service)	15
8.1	ROS의 메시지 통신	15
8.2	ROS2와 DDS	16
9	DDS란?	18
10	ROS에서의 사용법	19

1 ROS2 소개

로봇 개발을 위한 툴

1. Robot을 위한 오픈 소스, 메타 운영 시스템
 2. 하드웨어 추상화, 하위 디바이스 제어, 로봇틱스에서 많이 사용 되는 센싱, 인식, 지도 작성, 모션 플래닝 등의 기능 구현, 프로세스 사이의 메시지 패싱, 패키지 관리, 개발환경에 필요한 라이브러리와 다양한 개발, 디버깅 도구를 제공하는 툴
-

메타 운영체제

ROS (Robor Operating System)의 약자로 OS 이다. 하지만 기존 운영체제와는 다른 메타 운영체제의 성격을 가진다 (Meta-Operating System)

Meta-Operating System

애플리케이션과 분산 컴퓨팅 자원 간의 가상화 레이어로 분산 컴퓨팅 자원을 활용하여 스케줄링 및 로드, 감시, 에러처리 등을 실행 하는 시스템

ROS는 독립된 운영체제 (Window, Linux 등)가 아니며, 기존 운영체제를 이용한다. Ubuntu 위에 설치하여 제공하는 프로세스 관리 시스템, 파일시스템, UI, Program Utility 등을 사용하는 미들웨어, 소프트웨어 프레임워크 라고 한다.

ROS의 목적

1. ROS의 목적은 "로보틱스 소프트웨어 개발을 전 세계 레벨에서 공동 작업이 가능하도록 하는 환경 구축"이다.
 2. 로봇 소프트웨어 플랫폼, 미들웨어, 프레임워크를 지향하기 보다는 연구 개발에서의 코드 재사용을 극대화 하는 것에 초점이 잡혀있다.
 3. 이를 위하여, 노드 단위의 분산 프로세스, 공유 및 재배포를 쉽게 하기위한 패키지 단위 관리, 다양한 프로그래밍 언어지원 기능을 갖추고 있다.
-

ROS의 구성

1. ROS 는 약 400여개의 ROS-2 공통 패키지로 구성되어 있다.
 - i. Build System
 - ii. ROS Interface 생성, 관리 Package
 - iii. Interface Package -Topic, Service, Action 에 사용
 - iv. RMW Package -통신 미들웨어 포함
-

- v. RCL Package - 프로그래밍 언어를 지원하기 위한 클라이언트 라이브러리
 - vi. Robotics Application Framework – Robotics Application 작성 도구
 - vii. Robotics Application Package - 응용프로그램
 - viii. Simulation - 가상환경에서 로봇 제어해볼 수 있는 툴
 - ix. Software Development Tool
 - x. Etc.
-

ROS의 역사

2007 May – Stanford Univ. AI LAB 의 STAIR 프로젝트를 위해 Morgan Quigley 박사가 개발한 Switchyard System 에서 시작

2007 July - Willow Garage 가 본격적으로 ROS 개발 시작
* Willow Garage : OpenCV & PCL 개발 및 지원

2010 Jan 22nd – ROS 1.0 출시
2010 Mar 1st – ROS Box Turtle First Release
* BSD License & Apache License 2.0 기반 → 누구나 수정, 재사용 및 배포 가능

공식 Release 6th Version ROS Groovy Galapagos 를 끝으로 2013 년 Willow Garage → OSRF(Open Source Robotics Foundation) 이양

2017 May – OSRF → Open Robotics 변경
* ROS 개발, 운영 및 관

2020 May 23rd – ROS 13th Version / ROS Noetic Ninjemys ROS 1.0 의 마지막 버전 Release

ROS의 버전

ROS2 는 2014년 개발을 시작으로 2015년 Alpha 1 Release를 시작으로 2021년 ROS2의 7 번째 배포판인 ROS2 Galactic Geochelone Release 되었다.

* ROS 는 Release의 첫 글자를 알파벳 순서에 맞춰 공개하고 있으며 Turtle을 Symbol로 사용하고 있다.

ROS의 버전주기

과거 Ubuntu Release 주기와 동일하게 1년에 2번 (4월,10월) 업데이트가 이루어 졌으나, 빈번한 Update로 인하여 2013년 Hydro Medusa Version 부터는 1년에 1번 정식 버전의 Release가 이루어 지고 있다.

(새로운 Ubuntu xx.04 Version이 출시된 후 한 달이 지난 5월에 진행됨)

* 서포트 기간

기본적으로는 Release 후 2년간 지원 받을 수 있으며, LTS 버전에 맞추어 나오는 ROS 버전들은 LTS 서비스 기간과 동일하게 5년간 지원하게 된다.

2 ROS2 개발환경 구축

2.1 개발환경

ROS2 에서 권장하는 개발 환경은 다음과 같지만, 현재 ㈜라스테크에서 사용하는 개발 환경에 맞추어 아래 사양에 맞게 개발 환경을 구축 하였다.

구분	추천	선택 사항	설치 환경
기본 운영 체제	Linux Mint 20.x	Ubuntu 20.04.x LTS (Focal Fossa)	Ubuntu 20.04.x LTS (Focal Fossa)
로봇 운영 체제	ROS 2 Foxy Fitzroy	ROS 2 Rolling Ridley	ROS 2 Foxy Fitzroy
컴퓨터 아키텍처	amd64	amd64, arm64	amd64
통합 개발 환경 (IDE)	Visual Studio Code	QtCreator	Visual Studio Code
프로그래밍 언어	Python 3 (3.8.0), C++ 14	최신의 Python, C++ 버전	Python 3 (3.8.0), C++ 14
시뮬레이터	Gazebo 11.x	Ignition Citadel	Gazebo 11.x
DDS	Fast DDS	Cyclone DDS	Fast DDS
기타	CMake 3.16.3, Qt 5.12.5, OpenCV 4.2.0		CMake 3.16.3, Qt 5.12.5, OpenCV 4.2.0

2.2 기본 운영체제 설치

ubuntu 운영체제 설치 참조

2.3 로봇 운영체제 설치

지역설정	<pre>root@ubuntu-ros:~# apt-get install locale root@ubuntu-ros:~# locale-gen en_US en_US.UTF-8 root@ubuntu-ros:~# update-locale LC_ALL=en_US.UTF-8 root@ubuntu-ros:~# export LANG=en_US.UTF-8</pre>
	<pre>edit ~/.toolchain</pre>
	<pre>root@ubuntu-ros:~# vi .toolchain</pre>
	<pre>export LANG=en_US.UTF-8</pre>

	<pre>edit ~/.bashrc root@ubuntu-ros:~# vi ~/.bashrc if [-f ~/.toolchain]; then . ~/.toolchain fi</pre>
소스 설정	<pre>root@ubuntu-ros:~# apt-get install curl gnupg2 lsb-release root@ubuntu-ros:~# curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg root@ubuntu-ros:~# echo "deb [arch=\$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu focal main" tee /etc/apt/sources.list.d/ros2.list > /dev/null</pre>
ROS2 패키지 설치	<pre>root@ubuntu-ros:~# apt-get install ros-foxy-desktop ros-foxy-rmw- fastrtps* ros-foxy-rmw-cyclonedds* -y</pre>
	<p>설치 확인</p> <p>terminal #1</p> <pre>root@ubuntu-ros:~# . /opt/ros/foxy/setup.bash root@ubuntu-ros:~# ros2 run demo_nodes_cpp talker [INFO] [1657158012.879971684] [talker]: Publishing: 'Hello World: 1' [INFO] [1657158013.883565231] [talker]: Publishing: 'Hello World: 2' [INFO] [1657158014.880557583] [talker]: Publishing: 'Hello World: 3' [INFO] [1657158015.880447070] [talker]: Publishing: 'Hello World: 4' [INFO] [1657158016.880327618] [talker]: Publishing: 'Hello World: 5' [INFO] [1657158017.880235599] [talker]: Publishing: 'Hello World: 6' [INFO] [1657158018.881442342] [talker]: Publishing: 'Hello World: 7'</pre>
	<p>설치 확인</p> <p>terminal #2</p> <pre>root@ubuntu-ros:~# . /opt/ros/foxy/setup.bash root@ubuntu-ros:~# ros2 run demo_nodes_py listener [INFO] [1657158012.890211701] [listener]: I heard: [Hello World: 1] [INFO] [1657158013.885103035] [listener]: I heard: [Hello World: 2] [INFO] [1657158014.885408344] [listener]: I heard: [Hello World: 3] [INFO] [1657158015.881547457] [listener]: I heard: [Hello World: 4] [INFO] [1657158016.881557414] [listener]: I heard: [Hello World:</pre>

	<pre> 5] [INFO] [1657158017.881278489] [listener]: I heard: [Hello World: 6] [INFO] [1657158018.883287993] [listener]: I heard: [Hello World: 7] </pre>
--	---

3 개발툴 설치

기본 소프트웨어 설치	<pre> root@ubuntu-ros:~# apt-get install -y \ > build-essential \ > cmake \ > git \ > libbullet-dev \ > python3-colcon-common-extensions \ > python3-flake8 \ > python3-pip \ > python3-pytest-cov \ > python3-rosdep \ > python3-setuptools \ > python3-vcstool \ > wget </pre>
	<pre> root@ubuntu-ros:~# python3 -m pip install -U \ > argcomplete \ > flake8-blind-except \ > flake8-builtins \ > flake8-class-newline \ > flake8-comprehensions \ > flake8-deprecated \ > flake8-docstrings \ > flake8-import-order \ > flake8-quotes \ > pytest-repeat \ > pytest-rerunfailures \ > pytest </pre>
	<pre> root@ubuntu-ros:~# apt install --no-install-recommends -y \ > libasio-dev \ > libtinyxml2-dev \ > libcunit1-dev </pre>

4 ROS2 빌드 테스트

/opt/ros_ws 폴더를 기본 워크스페이스로 설정

빌드 확인	<pre> root@ubuntu-ros:~# mkdir -p /opt/rosws/src root@ubuntu-ros:~# cd /opt/rosws/src root@ubuntu-ros:/opt/rosws/src# . /opt/ros/foxy/setup.bash root@ubuntu-ros:/opt/rosws/src# colcon build --symlink-install Summary: 0 packages finished [0.21s] root@ubuntu-ros:/opt/rosws/src# ls -al total 20 drwxr-xr-x 5 root root 4096 7월 7 10:59 . drwxr-xr-x 3 root root 4096 7월 7 10:59 .. drwxr-xr-x 2 root root 4096 7월 7 10:59 build drwxr-xr-x 2 root root 4096 7월 7 10:59 install drwxr-xr-x 3 root root 4096 7월 7 10:59 log root@ubuntu-ros:/opt/rosws/src# </pre>
-------	--

5 Run Commands 설정

환경설정	<pre> edit ~/.rosrc root@ubuntu-ros:/opt/rosws/src# vi ~/.rosrc source /opt/ros/foxy/setup.bash source /opt/rosws/src/install/local_setup.bash source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash source /usr/share/vcstool-completion/vcs.bash source /usr/share/colcon_cd/function/colcon_cd.sh export PATH=./opt/ros/foxy:/opt/ros/foxy/bin:\$PATH export _colcon_cd_root=/opt/rosws export ROS_DOMAIN_ID=7 export ROS_NAMESPACE=robot1 export RMW_IMPLEMENTATION=rmw_fastrtps_cpp # export RMW_IMPLEMENTATION=rmw_connext_cpp # export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp # export RMW_IMPLEMENTATION=rmw_gurumdds_cpp # export RCUTILS_CONSOLE_OUTPUT_FORMAT='[{severity} {time}] [{name}]: {message} ({function_name}()) at {file_name}:{line_number}) ' export RCUTILS_CONSOLE_OUTPUT_FORMAT='[{severity}]: {message}' export RCUTILS_COLORIZED_OUTPUT=1 export RCUTILS_LOGGING_USE_STDOUT=0 export RCUTILS_LOGGING_BUFFERED_STREAM=1 </pre>
------	---

	<pre>edit ~/.bashrc</pre>
	<pre>if [-f ~/.rosrc]; then</pre>
	<pre> . ~/.rosrc</pre>
	<pre>fi</pre>

6 IDE 설치

VSCode

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

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Version 1.68 is now available! Read about the new features and fixes from May.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

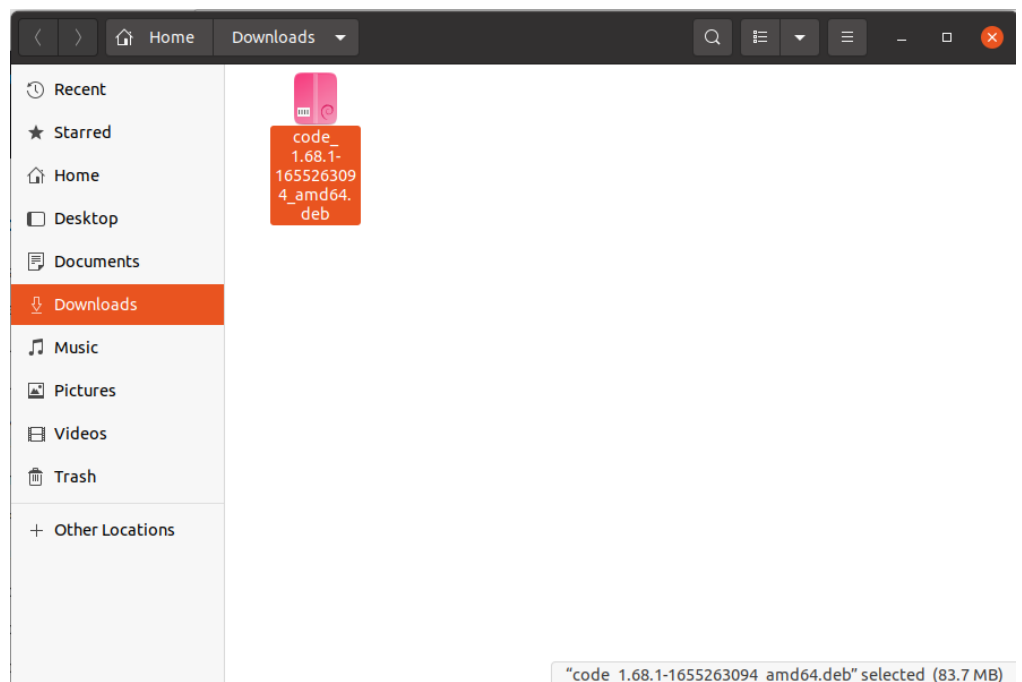
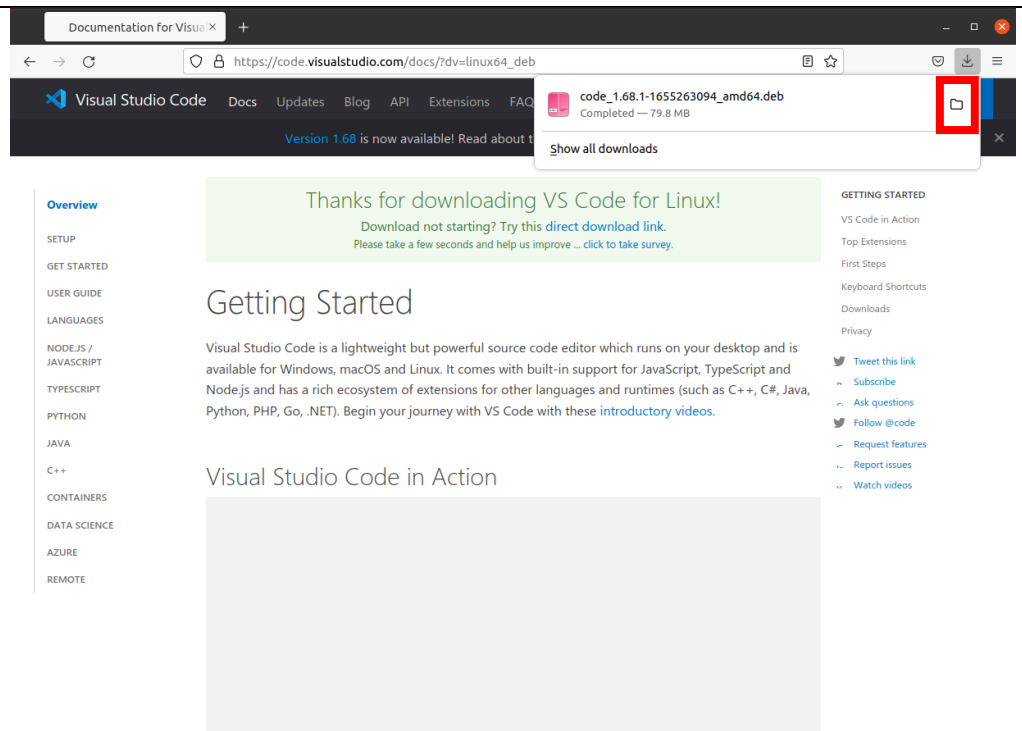
[.deb](#) [.rpm](#)

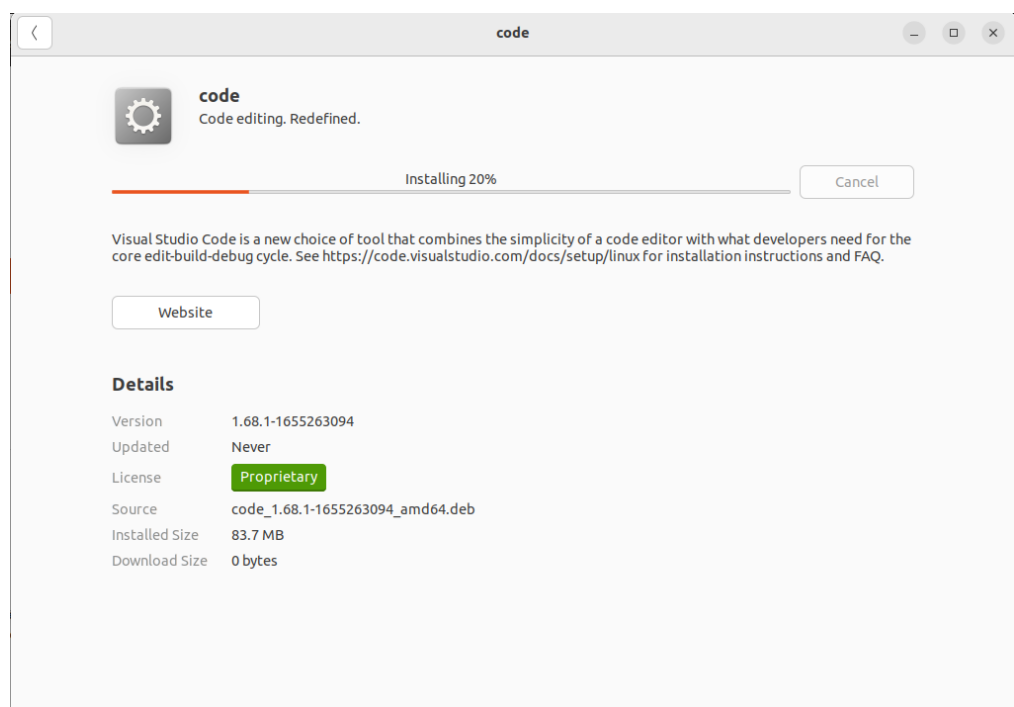
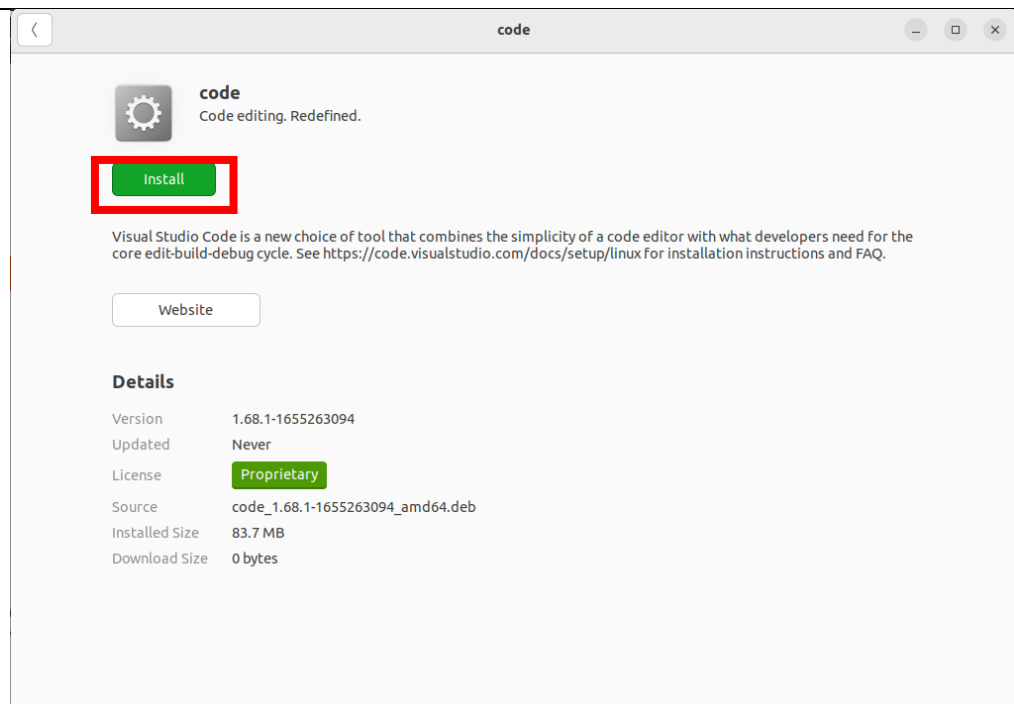
Debian, Ubuntu... Red Hat, Fedora...

Web, Insiders edition, or other platforms

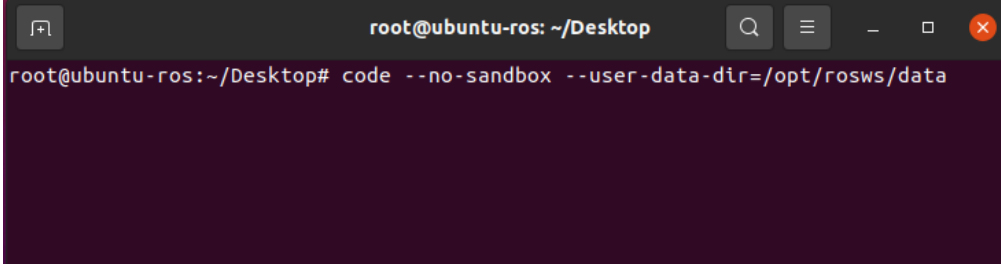
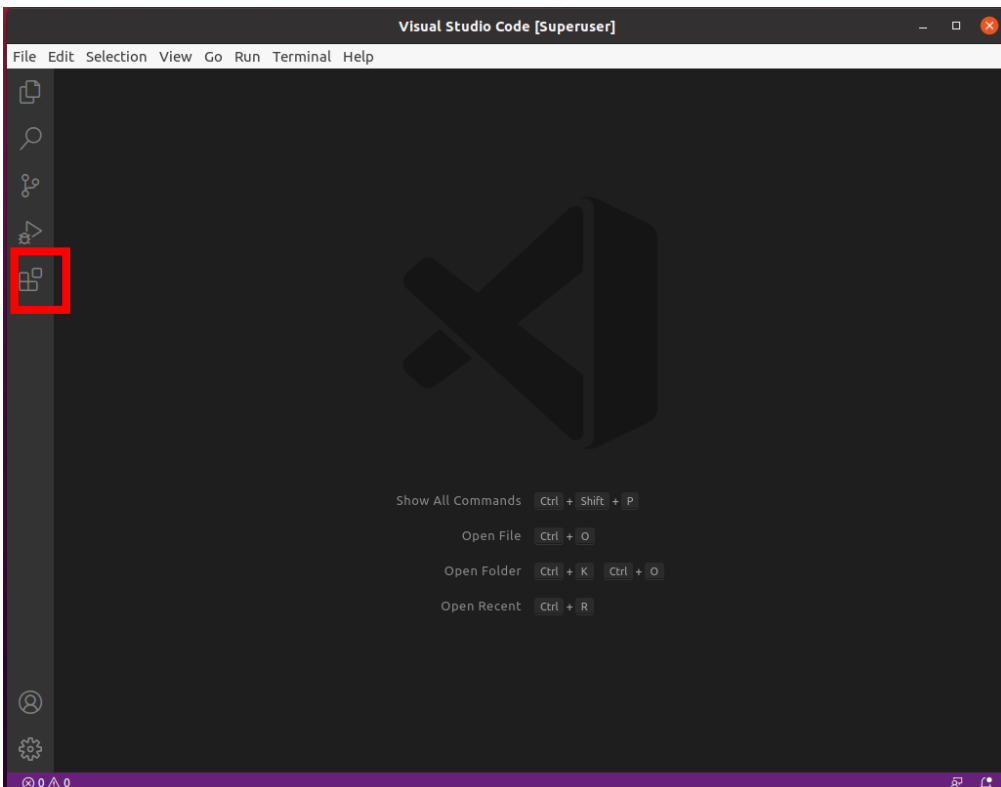
By using VS Code, you agree to its [license and privacy statement](#).

IntelliSense Run and Debug Built-in Git Extensions

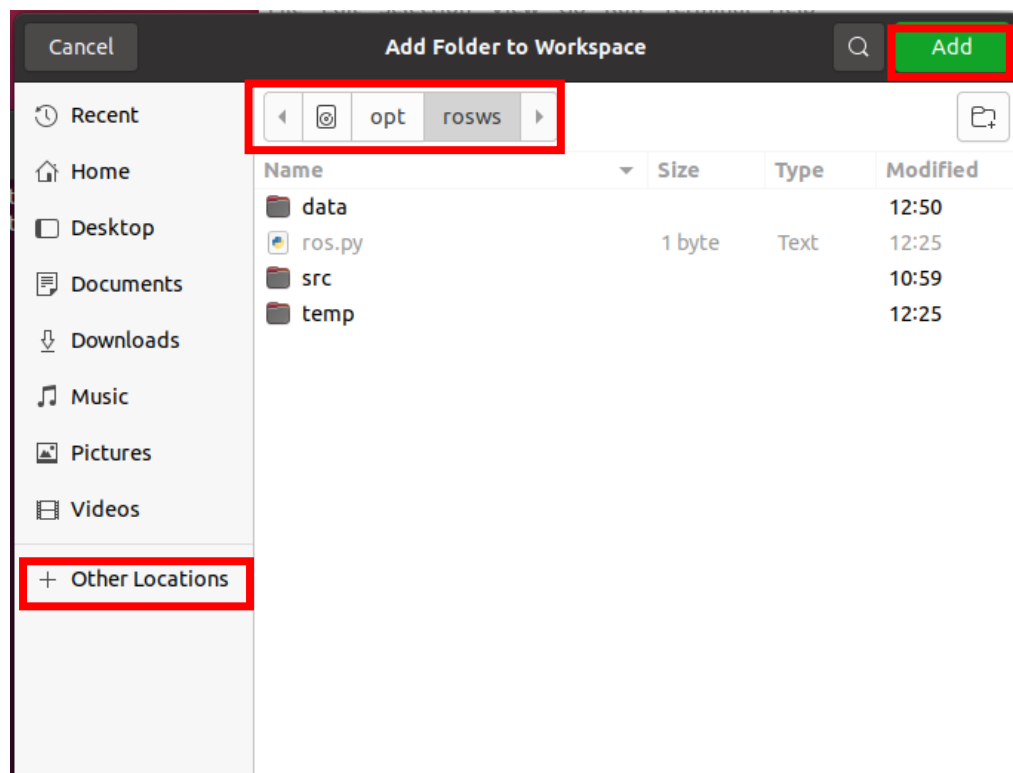
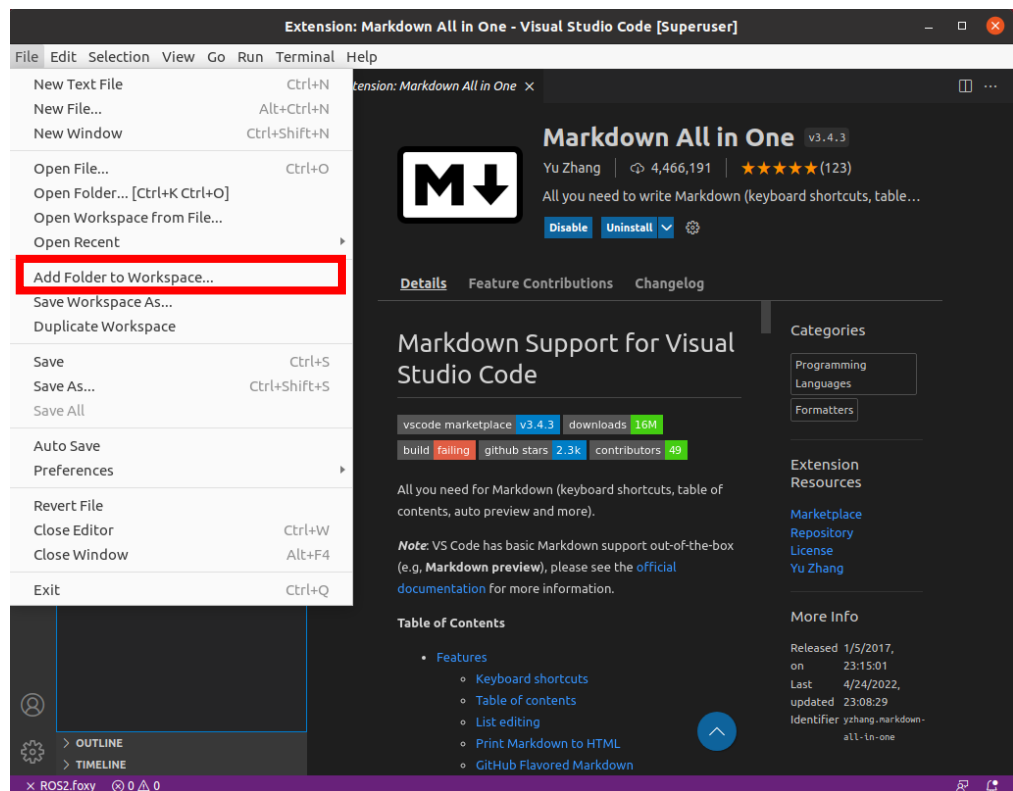


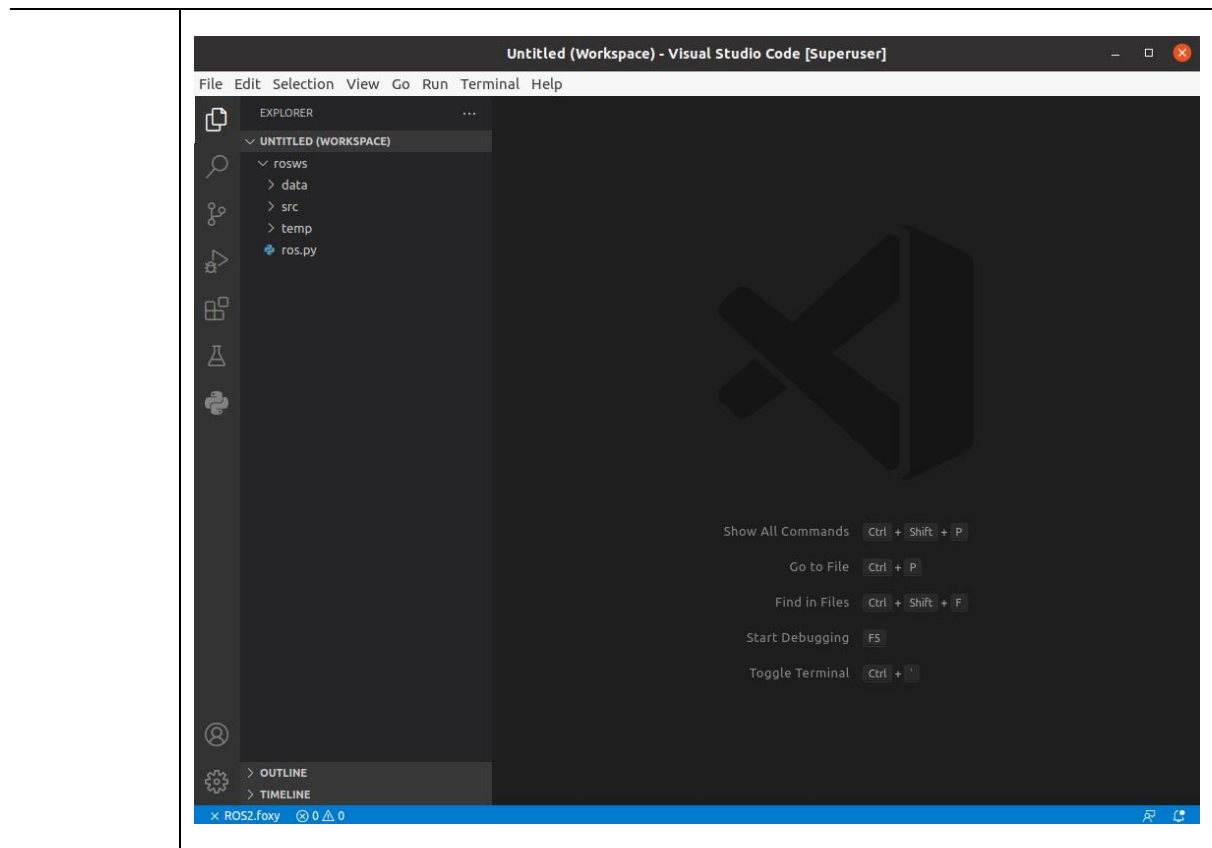


run code

	 <pre>root@ubuntu-ros: ~/Desktop root@ubuntu-ros:~/Desktop# code --no-sandbox --user-data-dir=/opt/ros/ws/data</pre>
vscode 확장 설치	<div><p>Visual Studio Code [Superuser]</p><p>File Edit Selection View Go Run Terminal Help</p><p>Show All Commands <code>Ctrl + Shift + P</code></p><p>Open File <code>Ctrl + O</code></p><p>Open Folder <code>Ctrl + K</code> <code>Ctrl + O</code></p><p>Open Recent <code>Ctrl + R</code></p></div> <div><p>확장 설치</p><p>C/C++, CMake, CMake Tools, Python</p><p>ROS, URDF, Colcon Tasks</p><p>XML Tools, YAML, Markdown All in One</p></div>

WS 설정





7 ROS 기본

7.1 기본용어

master	<ul style="list-style-type: none"> ➤ 노드와 노드 사이의 연결과 메시지 통신을 위한 네임 서버와 같은 역할을 한다 ➤ <code>roscore</code> 가 실행 명령이며 마스터를 실행하면 각 노드의 이름을 등록하고 필요에 따라서 정보만 받을 수 있다 ➤ 마스터가 없는 노드간 접속 및 토픽, 서비스는 서로 통신 할 수 없다.
node	<ul style="list-style-type: none"> ➤ <code>ros</code>에서 실행되는 최소 단위의 프로세서를 지칭한다, 즉 하나의 실행 가능한 프로그램으로 생각하면 된다. ➤ <code>ros</code> 는 하나의 목적에 하나의 노드(실행 파일), 재사용이 쉽게 개발하는 것이 좋다
package	<ul style="list-style-type: none"> ➤ <code>ros</code>를 구성하는 기본적인 단위이다 <code>ros</code>의 응용 프로그램은 패키지 단위로 생성해야 한다, 또한 패키지는 하나 이상의 노드를 포함 하거나 다른 패키지의 노드를 실행하기 위한 설정 파일을 포함하게 된다 <hr/> <p>의존성 라이브러리, 데이터셋 설정 등이 있다.</p> <hr/> <pre>catkin_create_pkg, rospy_tutorial, geometry_msgs, rospy <- ros1</pre>

	catkin_create_pkg 명령으로 geometry_msgs 와 rospy 에 의존성을 가진 rospy_tutorial 패키지 생성
meta package	➤ 메타 패키지는 공통된 목적을 지닌 패키지들의 집단을 말한다.
message	➤ 노드는 메시지를 통하여 노드간의 데이터를 주고 받는다 ➤ 메시지는 integer, floating, point, boolean 과 같은 변수의 형식이다.
topic	➤ 단방향 ➤ 토픽은 publisher 노드가 발행하는 메시지를 말한다 ➤ 노드가 발행하는 토픽의 정보를 subscribe 해서 받을 정보를 바탕으로 노드간의 정보를 교환 한다.
publish & publisher	➤ 퍼블리시는 토픽의 내용에 해당하는 메시지의 형태 데이터를 송신하는 것을 말한다. ➤ 퍼블리셔 노드는 퍼블리시를 수행하기 위하여 토픽을 포함한 자신의 정보들을 마스터에 등록한다
subscribe & subscriber	➤ 토픽의 내용에 해당하는 메시지의 형태의 데이터를 수신하는 것을 말한다 ➤ subscriber 노드는 subscribe를 수행하기 위하여 토픽을 포함한 자신의 정보를 마스터에 등록하고 구독하고자 하는 토픽을 publish 하는 publisher 노드의 정보를 마스터로부터 받는다.
service	➤ 서비스 메시지 통신은 특정 목적의 해당하는 작업에 해당되는 서비스를 요청하는 서비스 클라이언트와 서비스 응답을 담당하는 서비스 서버간의 동기적 양방향 서비스 메시지 통신을 말한다.
action	➤ 액션은 서비스처럼 양방향을 요구 하거나 요청 후 응답까지 오랜시간이 걸리고 중간 결과값이 필요한 경우 사용 되는 메시지 통신 방식이다
parameter	➤ 노드에서 사용 되는 파라미터를 이야기 한다 흔히 윈도우 프로그램에서 *.ini 설정과 같다 ➤ 파라미터는 디폴트로 설정값들이 지정되어 있고 필요에 따라 외부에서 읽거나 쓸 수 있다, 특히 외부에서 쓰기 기능을 사용하여 상황에 따라 설정값들이 실시간으로 변경이 가능하다.

roslaunch	<ul style="list-style-type: none"> ➤ ros의 기본 실행 명령이다 패키지에서 하나의 노드를 실행하는데 사용된다. ➤ ex) roslaunch turtlesim turtlesim_node -> 거북이 소환 <hr/> <p>turtle 소환</p> <hr/> <pre>root@ubuntu-ros:/opt/ros/ws/src# roslaunch turtlesim turtlesim_node</pre> <hr/>
roslaunch	<ul style="list-style-type: none"> ➤ roslaunch 이 하나의 노드를 실행 한다면 런치는 여러개의 노드/실행파일을 한번에 실행하는 개념이다 이 명령어를 통하여 하나 이상의 노드를 실행 시킬 수 있다 <hr/> <p>turtle bringup</p> <hr/> <pre>root@ubuntu-ros:/opt/ros/ws/src# roslaunch turtlebot3_bringup turtlebot3_robot.launch</pre> <hr/>

8 ROS2와 DDS(Data Distribution Service)

8.1 ROS의 메시지 통신

메시지 통신은 ROS 프로그래밍에 있어서 ROS 1과 2의 공통된 중요한 핵심 개념

ROS에서는 프로그램의 재사용성을 극대화하기 위하여 최소 단위의 실행 가능한 프로세서라고 정의하는 노드(node) 단위의 프로그램을 작성 - 하나의 실행 가능한 프로그램

하나 이상의 노드 또는 노드 실행을 위한 정보 등을 묶어 놓은 것을 패키지(package)라고 하며, 패키지의 묶음을 메타패키지(metapackage)라 하여 따로 분리

실제 실행 프로그램인 노드인데 앞서 이야기한 것과 마찬가지로 ROS에서는 최소한의 실행 단위로 프로그램을 나누어 프로그래밍하기 때문에 노드는 각각 별개의 프로그램이라고 이해하면 됨.

이에 수많은 노드들이 연동되는 ROS 시스템을 위해서는 노드와 노드 사이에 입력과 출력 데이터를 서로 주고받게 설계해야만 한다.

여기서 주고받는 데이터를 ROS에서는 message라고 하고 주고받는 방식을 메시지 통신이라고 한다.

데이터에 해당되는 메시지(message)는 integer, floating point, boolean, string 와 같은 변수 형태

메시지 안에 메시지를 품고 있는 간단한 데이터 구조 및 메시지들의 배열과 같은 구조도 사용

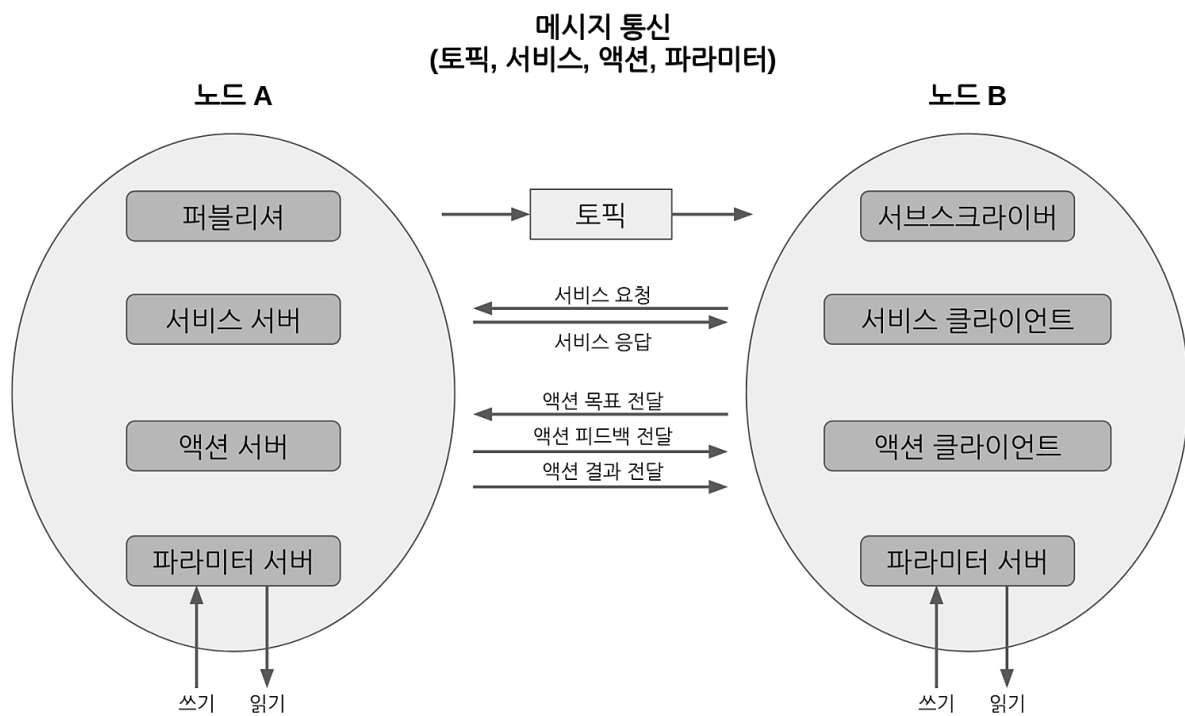
그리고 메시지를 주고받는 통신 방법에 따라

topic

service

action

parameter



8.2 ROS2와 DDS

ROS에서 사용되는 메시지 통신 방법으로는 토픽(topic), 서비스(service), 액션(action), 파라미터(parameter)가 있다.

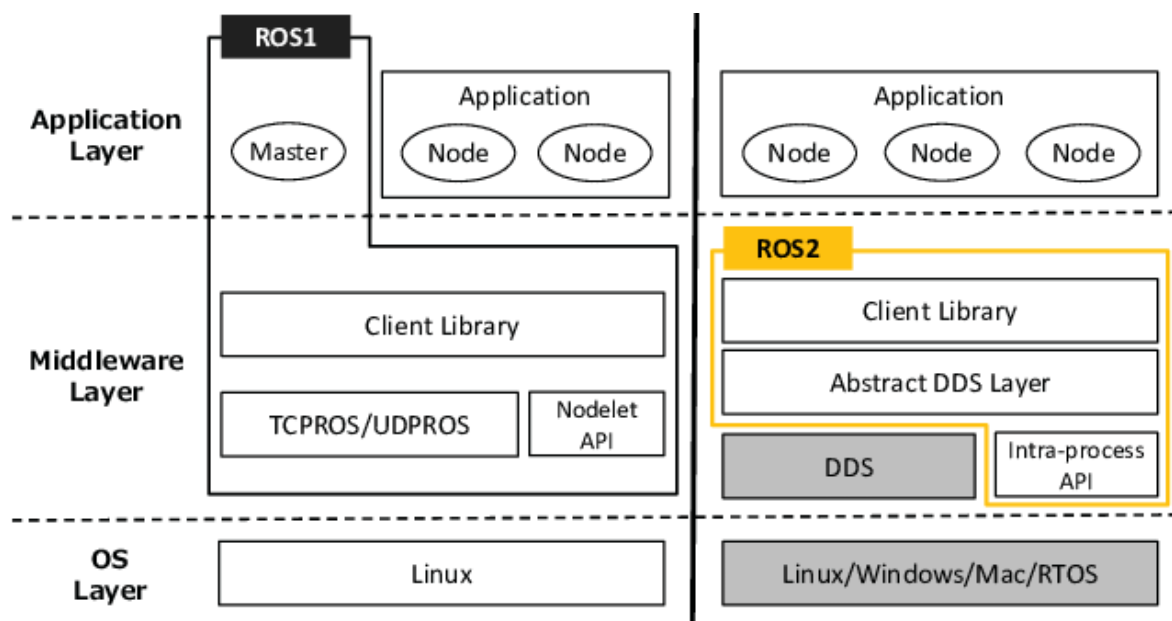
각 메시지 통신 방법의 목적과 사용 방법은 다르기는 하지만 토픽의 발간(publish)과 구독(subscribe)의 개념을 응용하고 있다.

이 데이터를 보내고 받는 발간, 구독 개념은 ROS 1은 물론 ROS 2에서도 매우 중요한 개념으로 변함이 없는데 이 기술에 사용된 통신 라이브러리는 ROS 1, 2에서 조금씩 다르다.

ROS 1에서는 자체 개발한 TCPROS와 같은 통신 라이브러리를 사용하고 있던 반면, ROS 2에서는 OMG(Object Management Group)에 의해 표준화된 DDS(Data Distribution Service)의 리얼타임 퍼블리시와 서브스크라이브 프로토콜인 DDSI-RTPS(Real Time Publish Subscribe)를 사용하고 있다.

ROS 2 개발 초기에는 기존 TCPROS를 개선하거나 ZeroMQ, Protocol Buffers 및 Zeroconf 등을 이용하여 미들웨어처럼 사용하는 방법도 제안되었으나 무엇보다 산업용 시장을 위해 표준 방식 사용을 중요하게 여겼고, ROS 1때와 같이 자체적으로 만들기 보다는 산업용 표준을 만들고 생태계를 꾸려가고 있었던 DDS를 통신 미들웨어로써 사용하기로 하였다.

DDS 도입에 따라 그림과 같이 ROS의 레이아웃은 크게 바뀌게 되었다. 처음에는 DDS 채용에 따른 장점과 단점에 대한 팽팽한 줄다리기 토론으로 걱정의 목소리도 높았지만 지금에 와서는 ROS 2에서의 DDS 도입은 상업적인 용도로 ROS를 사용할 수 있게 발판을 만들었다는 것에 가장 큰 역할을 했다는 평가가 지배적이다.



DDS 도입으로 기존 메시지 형태 이외에도 OMG의 CORBA시절부터 사용되던 IDL(Interface Description Language)를 사용하여 메시지 정의 및 직렬화를 더 쉽게, 더 포괄적으로 다룰 수 있게 되었다. 또한 DDS의 중요 컨셉인 DCPS(data-centric publish-subscribe), DLRL(data local

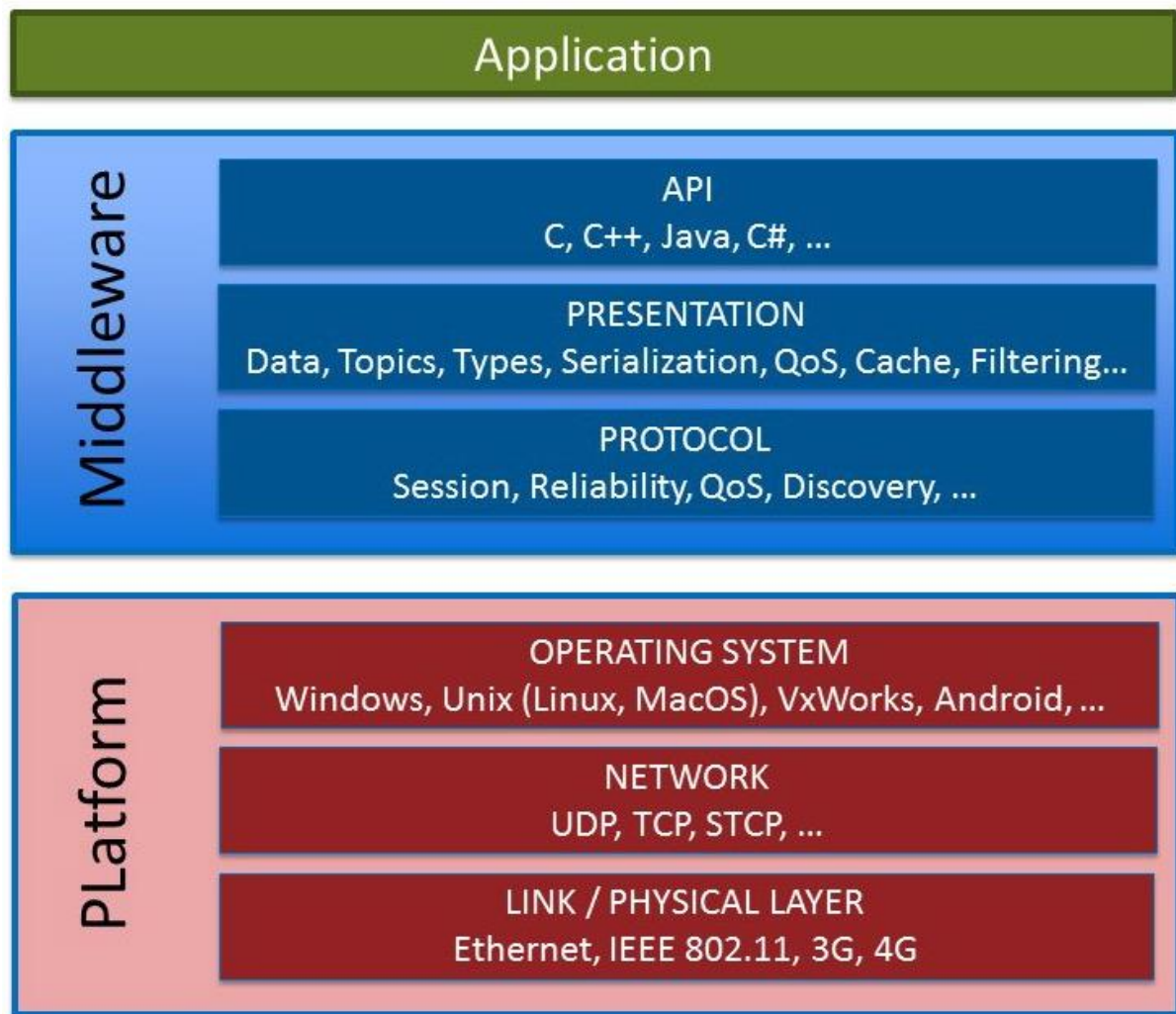
reconstruction layer)의 내용을 담아 재정한 통신 프로토콜로인 DDSI-RTPS를 채용하여 실시간 데이터 전송을 보장하고 임베디드 시스템에도 사용할 수 있게 되었다. DDS의 사용으로 노드 간의 동적 검색 기능을 지원하고 있어서 기존 ROS 1에서 각 노드들의 정보를 관리하였던 ROS Master가 없어도 여러 DDS 프로그램 간에 통신할 수 있다. 또한 노드 간의 데이터 통신을 세부적으로 조정하는 QoS(Quality of Service)를 매개 변수 형태로 설정할 수 있어서 TCP처럼 데이터 손실을 방지함으로써 신뢰도를 높이거나, UDP처럼 통신 속도를 최우선시하여 사용할 수도 있다. 그리고 산업용으로 사용되는 미들웨어인 만큼 DDS-Security도입으로 보안 측면에도 큰 혜택을 얻을 수 있었다. 이러한 다양한 기능을 갖춘 DDS를 이용하여 ROS 1의 퍼블리시, 서브스크라이브형 메시지 전달은 물론, 실시간 데이터 전송, 불안정한 네트워크에 대한 대응, 보안 등이 강화되었다. DDS의 채용은 ROS 1에서 ROS 2로 바뀌면서 가장 큰 변화점이자 그림 3과 같이 개발자 및 사용자로 하여금 통신 미들웨어에 대한 개발 및 이용 부담을 줄여 진짜로 집중해야 할 부분에 더 많은 시간을 쏟을 수 있게 되었다.

9 DDS란?

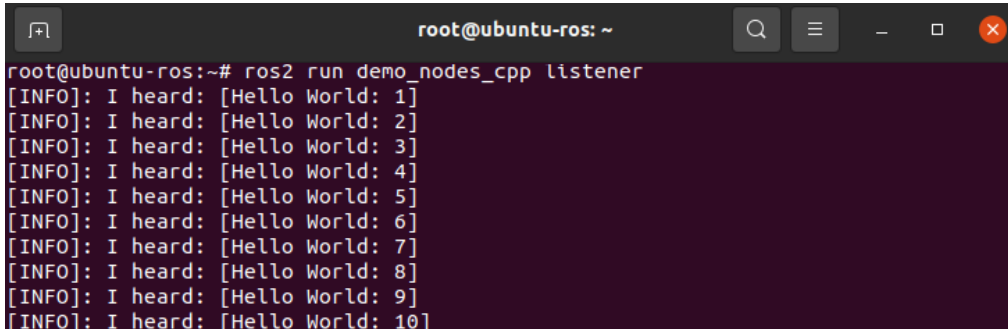
결론부터 말하자면 DDS는 데이터 분산 시스템이라는 용어로 OMG에서 표준을 정하고자 만든 트레이드 마크(TM)였다. 그냥 용어이고 그 실체는 데이터 통신을 위한 미들웨어이다.

DDS가 ROS 2의 미들웨어로 사용하는 만큼 그 자체에 대해 너무 자세히 알 필요는 없을 듯싶고 ROS 프로그래밍에 필요한 개념만 알고 넘어가면 될 듯싶다. 우선 정의부터 알아보자. DDS는 Data Distribution Service, 즉 데이터 분산 서비스의 약자이다.

즉, DDS는 데이터 분산 시스템이라는 개념을 나타내는 단어이고 실제로는 데이터를 중심으로 연결성을 갖는 미들웨어의 프로토콜(DDSI-RTPS, [30])과 같은 DDS 사양[31]을 만족하는 미들웨어 API가 그 실체이다. 이 미들웨어는 그림 4와 같이 ISO 7 계층 레이어[32]에서 호스트 계층(Host layers)에 해당되는 4~7 계층에 해당되고 ROS 2에서는 위에서 언급한 그림 3과 같이 운영 체제와 사용자 애플리케이션 사이에 있는 소프트웨어 계층으로 이를 통해 시스템의 다양한 구성 요소를 보다 쉽게 통신하고 데이터를 공유할 수 있게 된다.



10 ROS에서의 사용법

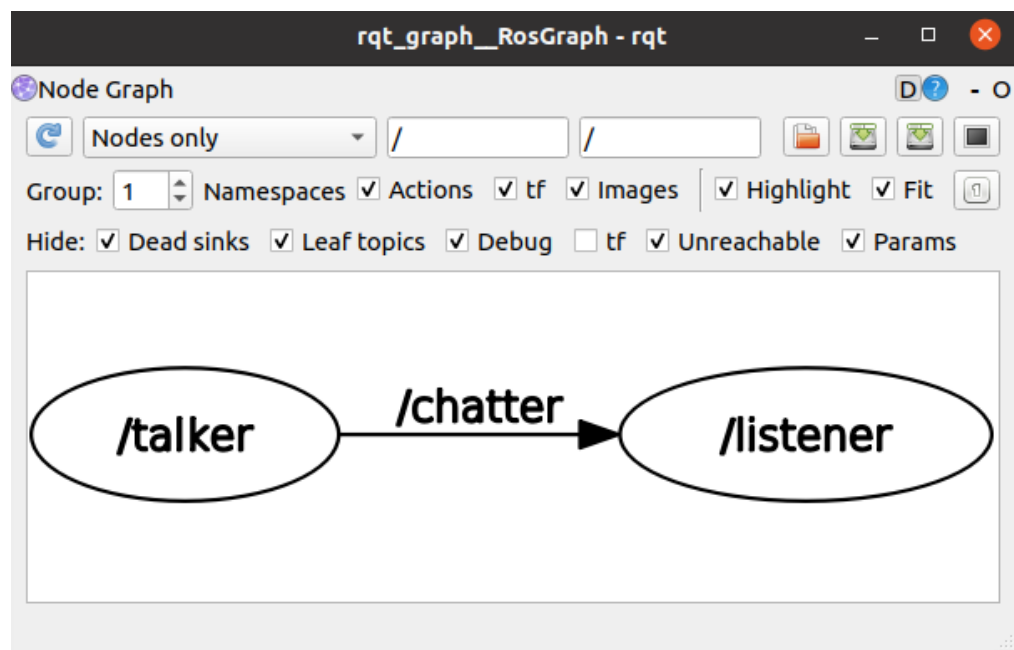
노드 실행	listener 실행  <pre> root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 run demo_nodes_cpp listener [INFO]: I heard: [Hello World: 1] [INFO]: I heard: [Hello World: 2] [INFO]: I heard: [Hello World: 3] [INFO]: I heard: [Hello World: 4] [INFO]: I heard: [Hello World: 5] [INFO]: I heard: [Hello World: 6] [INFO]: I heard: [Hello World: 7] [INFO]: I heard: [Hello World: 8] [INFO]: I heard: [Hello World: 9] [INFO]: I heard: [Hello World: 10] </pre>
-------	--

talker 실행

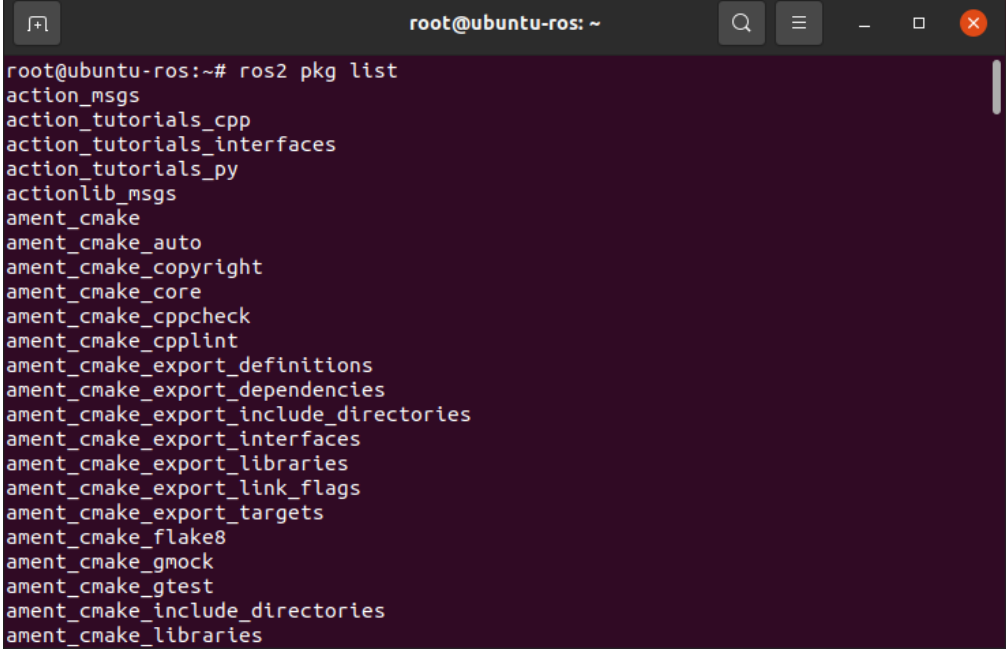
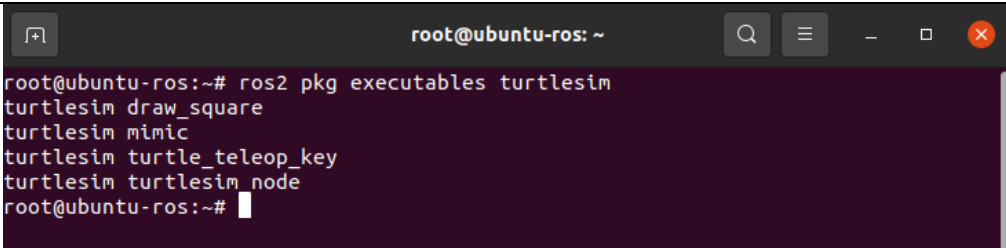
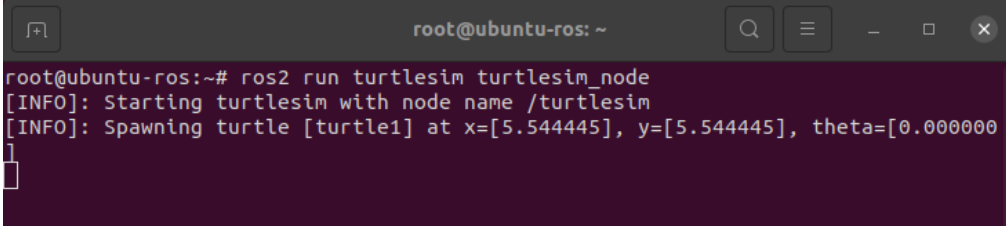
```
root@ubuntu-ros: ~  
root@ubuntu-ros:~# ros2 run demo_nodes_cpp talker  
[INFO]: Publishing: 'Hello World: 1'  
[INFO]: Publishing: 'Hello World: 2'  
[INFO]: Publishing: 'Hello World: 3'  
[INFO]: Publishing: 'Hello World: 4'  
[INFO]: Publishing: 'Hello World: 5'  
[INFO]: Publishing: 'Hello World: 6'  
[INFO]: Publishing: 'Hello World: 7'  
[INFO]: Publishing: 'Hello World: 8'  
[INFO]: Publishing: 'Hello World: 9'  
[INFO]: Publishing: 'Hello World: 10'
```

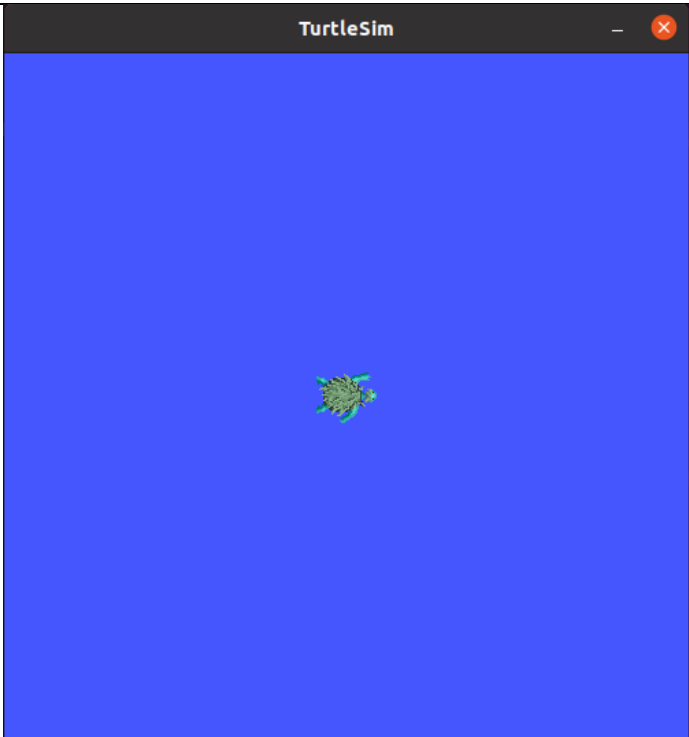
rqt_graph 실행

```
root@ubuntu-ros: ~  
root@ubuntu-ros:~# rqt_graph  
█
```



패키지관리

	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 pkg list action_msgs action_tutorials_cpp action_tutorials_interfaces action_tutorials_py actionlib_msgs ament_cmake ament_cmake_auto ament_cmake_copyright ament_cmake_core ament_cmake_cppcheck ament_cmake_cpplint ament_cmake_export_definitions ament_cmake_export_dependencies ament_cmake_export_include_directories ament_cmake_export_interfaces ament_cmake_export_libraries ament_cmake_export_link_flags ament_cmake_export_targets ament_cmake_flake8 ament_cmake_gmock ament_cmake_gtest ament_cmake_include_directories ament_cmake_libraries</pre>
특정 패키지 관리 (turtlesim)	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 pkg executables turtlesim turtlesim draw_square turtlesim mimic turtlesim turtle_teleop_key turtlesim turtlesim_node root@ubuntu-ros:~#</pre>
turtlesim 실행	<p>turtlesim_node 실행</p>  <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 run turtlesim turtlesim_node [INFO]: Starting turtlesim with node name /turtlesim [INFO]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000] </pre>

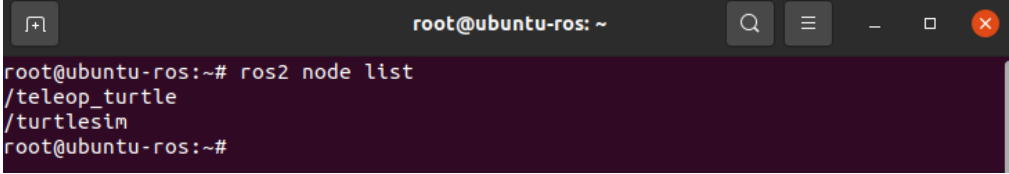
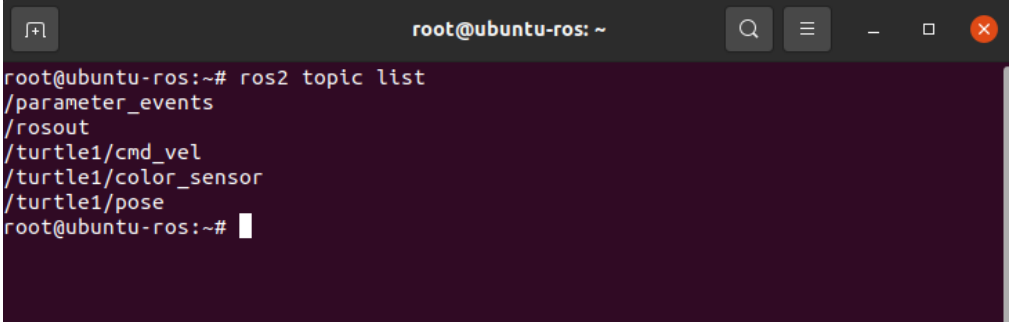
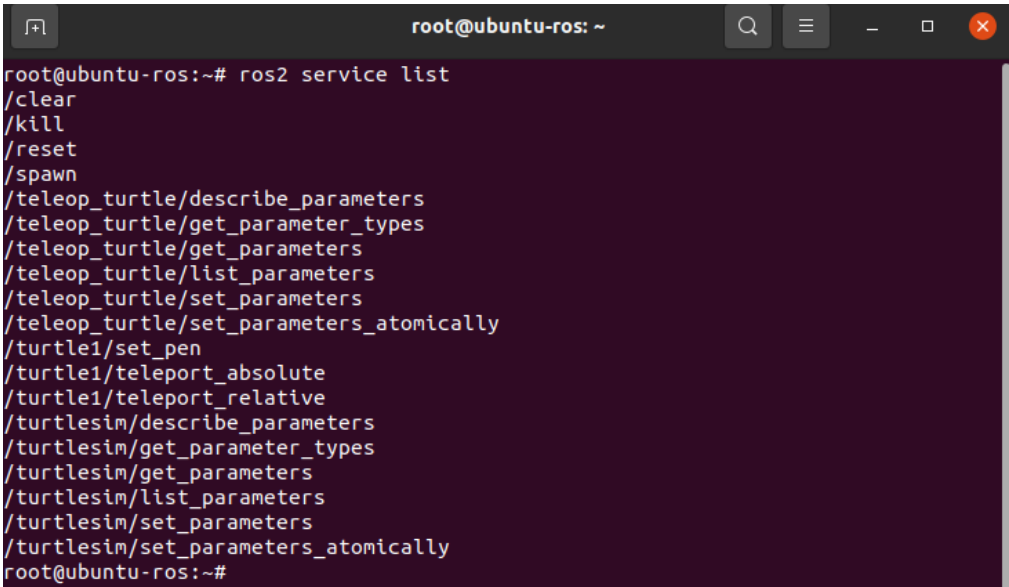
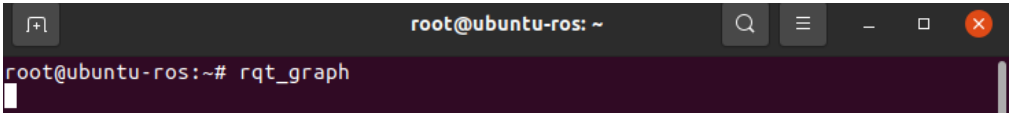


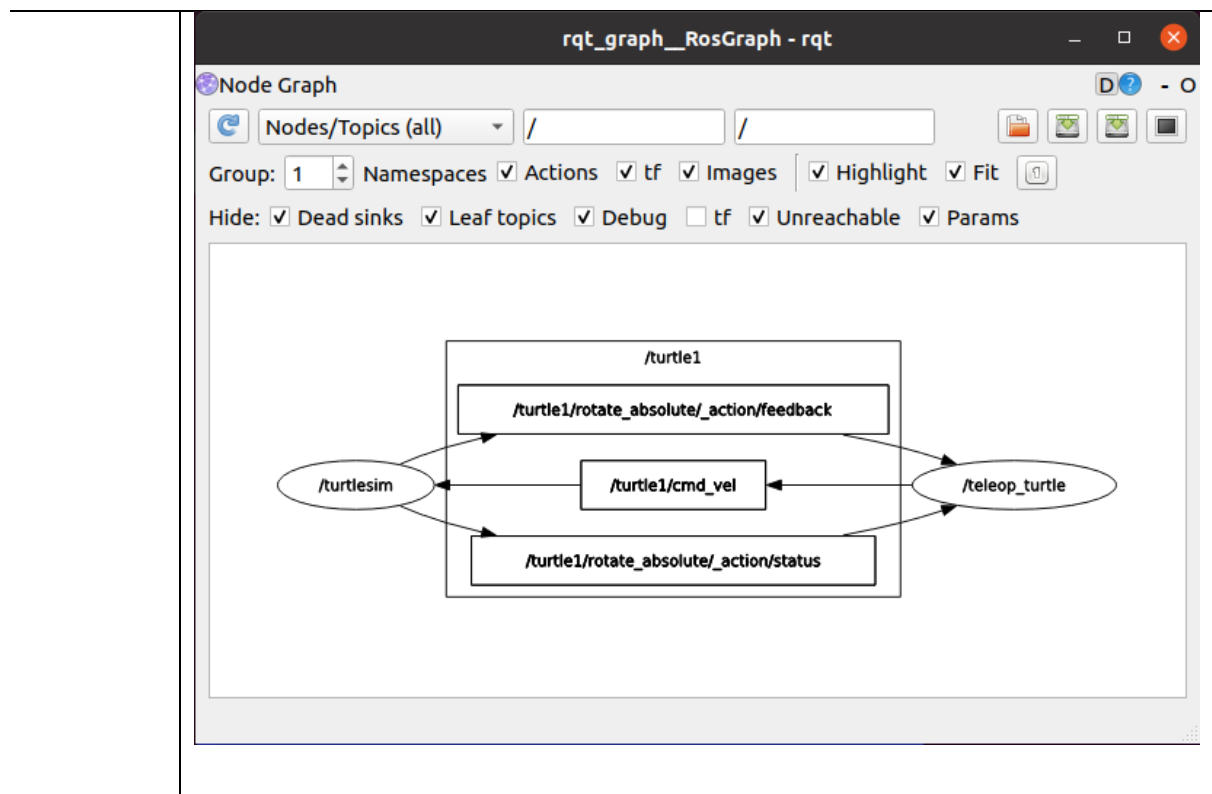
turtle_teleop_key 실행

```
root@ubuntu-ros: ~
root@ubuntu-ros:~# 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.
█
```

E : ↖	R : ↑	T : ↗
D : ←	F	G : →
C : ↙	V : ↓	B : ↘

실행 노드 확인

	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 node list /teleop_turtle /turtlesim root@ubuntu-ros:~#</pre>
실행 action 확인	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 topic list /parameter_events /rosout /turtle1/cmd_vel /turtle1/color_sensor /turtle1/pose root@ubuntu-ros:~#</pre>
service 확인	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 service list /clear /kill /reset /spawn /teleop_turtle/describe_parameters /teleop_turtle/get_parameter_types /teleop_turtle/get_parameters /teleop_turtle/list_parameters /teleop_turtle/set_parameters /teleop_turtle/set_parameters_atomically /turtle1/set_pen /turtle1/teleport_absolute /turtle1/teleport_relative /turtlesim/describe_parameters /turtlesim/get_parameter_types /turtlesim/get_parameters /turtlesim/list_parameters /turtlesim/set_parameters /turtlesim/set_parameters_atomically root@ubuntu-ros:~#</pre>
rqt_graph로 보는 노트와 토픽	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# rqt_graph </pre>

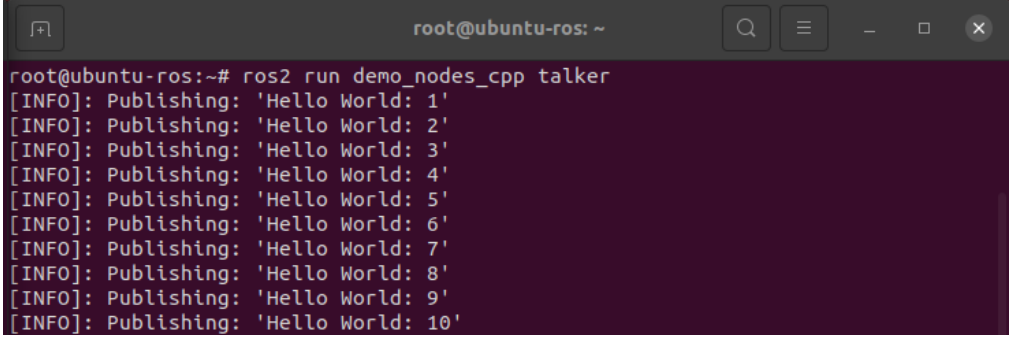
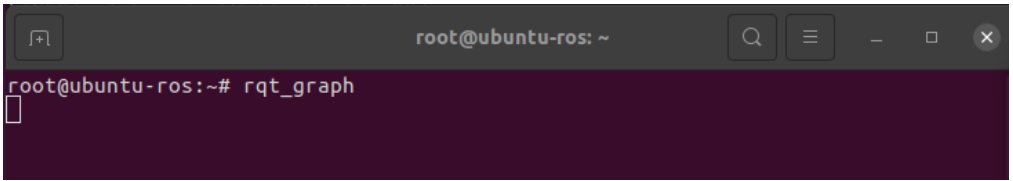
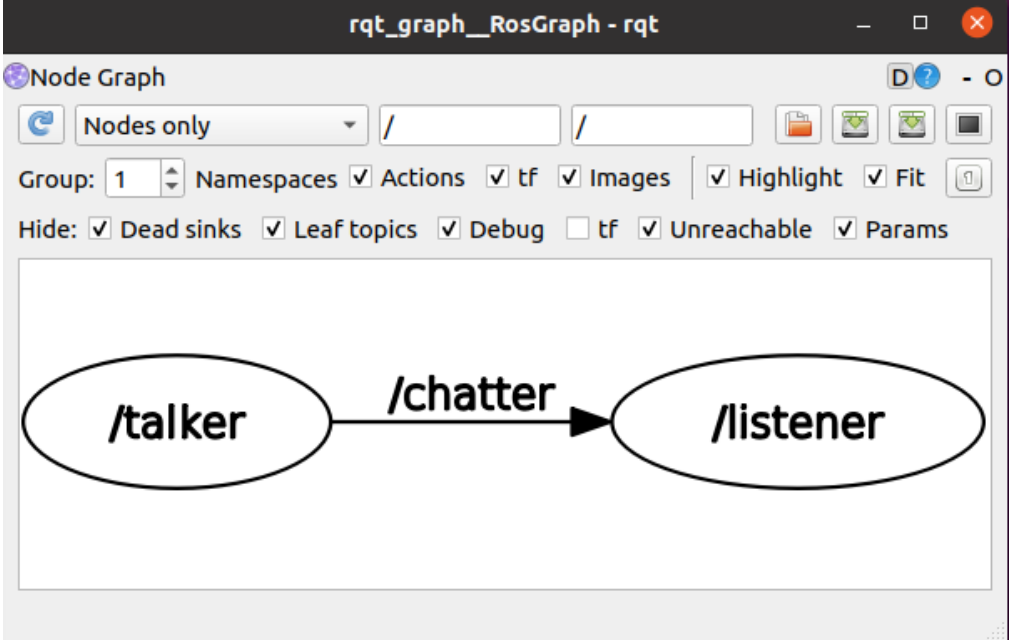


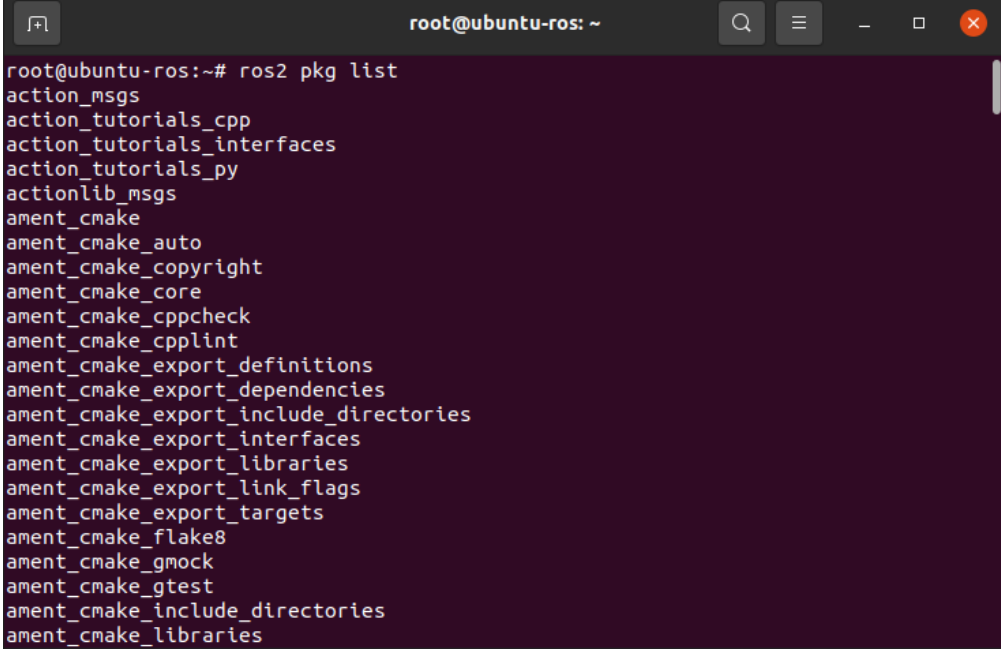
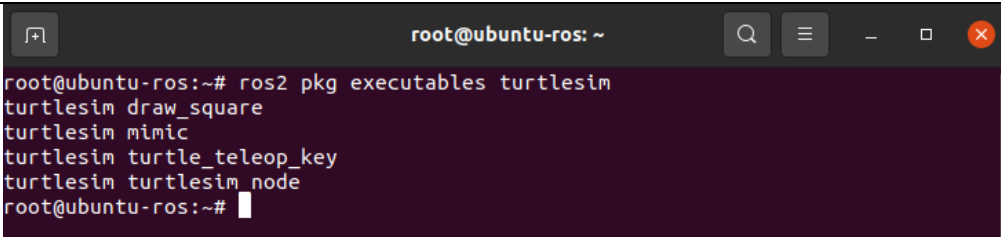
노드 실행

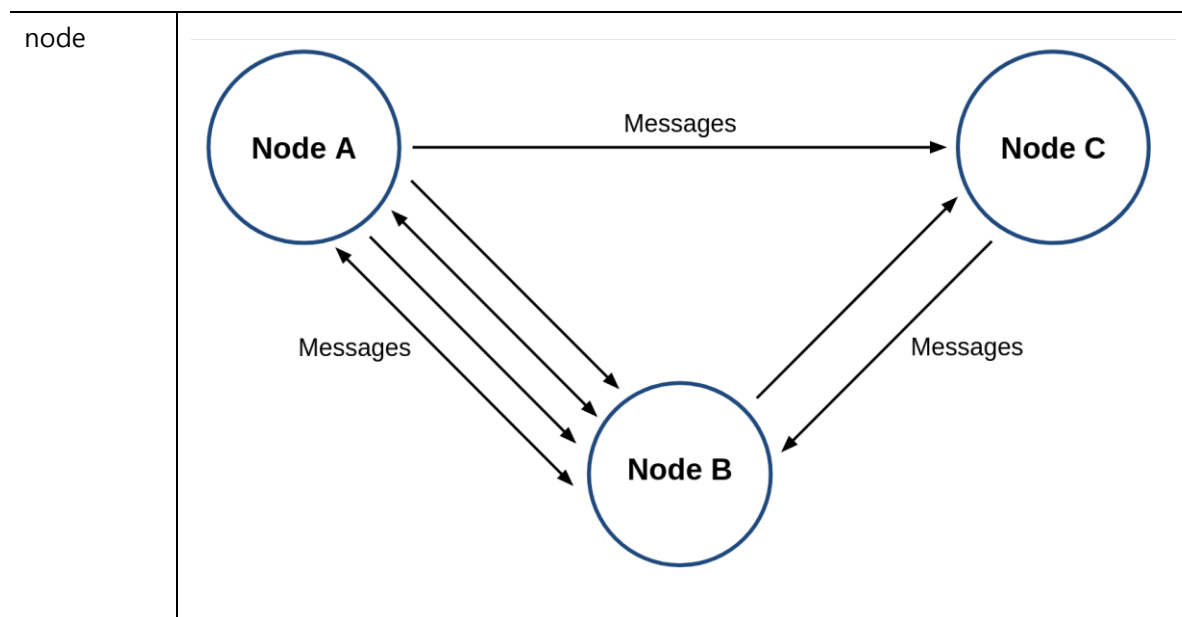
listener 실행

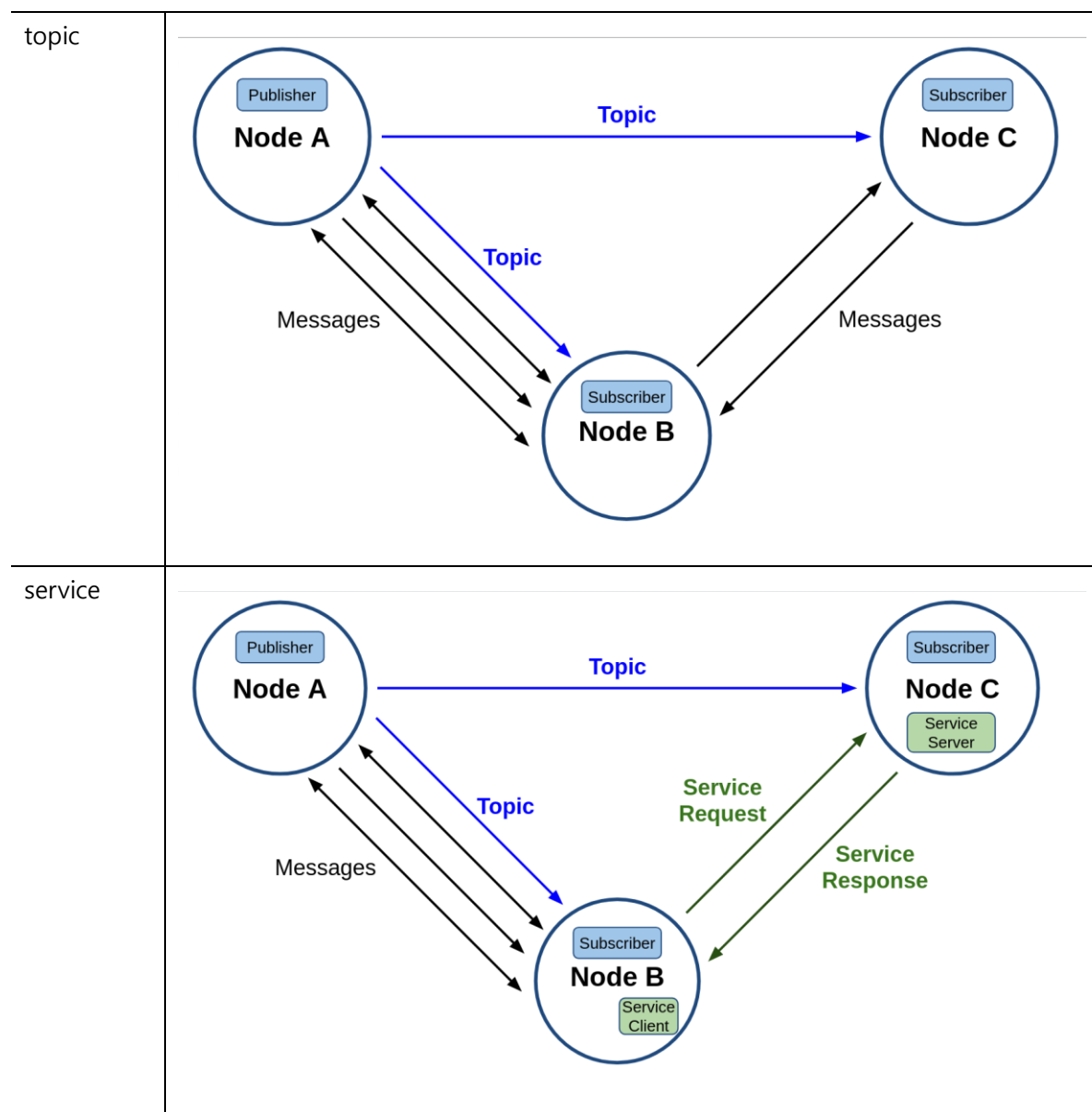
```
root@ubuntu-ros: ~  
root@ubuntu-ros:~# ros2 run demo_nodes_cpp listener  
[INFO]: I heard: [Hello World: 1]  
[INFO]: I heard: [Hello World: 2]  
[INFO]: I heard: [Hello World: 3]  
[INFO]: I heard: [Hello World: 4]  
[INFO]: I heard: [Hello World: 5]  
[INFO]: I heard: [Hello World: 6]  
[INFO]: I heard: [Hello World: 7]  
[INFO]: I heard: [Hello World: 8]  
[INFO]: I heard: [Hello World: 9]  
[INFO]: I heard: [Hello World: 10]
```

talker 실행

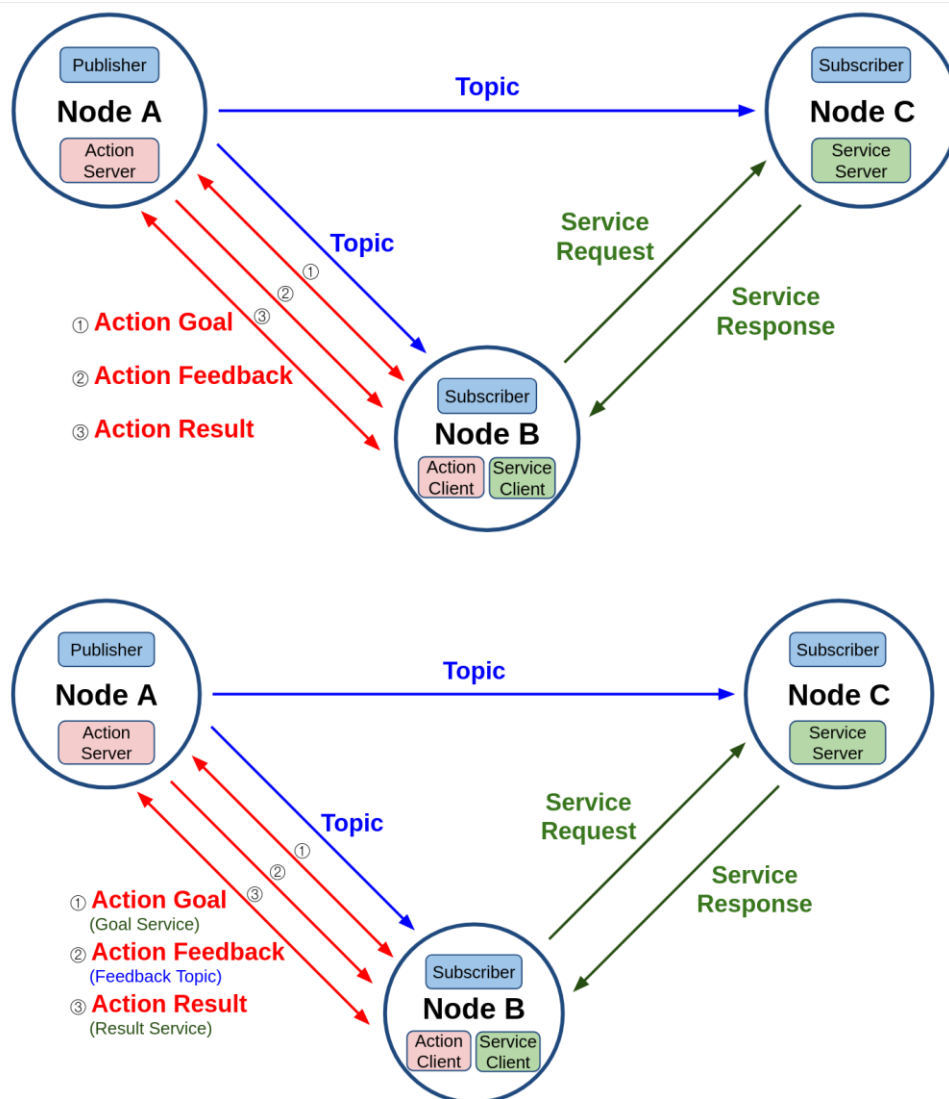
	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 run demo_nodes_cpp talker [INFO]: Publishing: 'Hello World: 1' [INFO]: Publishing: 'Hello World: 2' [INFO]: Publishing: 'Hello World: 3' [INFO]: Publishing: 'Hello World: 4' [INFO]: Publishing: 'Hello World: 5' [INFO]: Publishing: 'Hello World: 6' [INFO]: Publishing: 'Hello World: 7' [INFO]: Publishing: 'Hello World: 8' [INFO]: Publishing: 'Hello World: 9' [INFO]: Publishing: 'Hello World: 10'</pre>
rqt_graph 실행	 <pre>root@ubuntu-ros:~# rqt_graph █</pre>  <p>rqt_graph_RosGraph - rqt</p> <p>Node Graph</p> <p>Nodes only / /</p> <p>Group: 1 Namespaces <input checked="" type="checkbox"/> Actions <input checked="" type="checkbox"/> tf <input checked="" type="checkbox"/> Images <input checked="" type="checkbox"/> Highlight <input checked="" type="checkbox"/> Fit</p> <p>Hide: <input checked="" type="checkbox"/> Dead sinks <input checked="" type="checkbox"/> Leaf topics <input checked="" type="checkbox"/> Debug <input type="checkbox"/> tf <input checked="" type="checkbox"/> Unreachable <input checked="" type="checkbox"/> Params</p> <p>/talker → /chatter → /listener</p>
패키지관리	

	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 pkg list action_msgs action_tutorials_cpp action_tutorials_interfaces action_tutorials_py actionlib_msgs ament_cmake ament_cmake_auto ament_cmake_copyright ament_cmake_core ament_cmake_cppcheck ament_cmake_cpplint ament_cmake_export_definitions ament_cmake_export_dependencies ament_cmake_export_include_directories ament_cmake_export_interfaces ament_cmake_export_libraries ament_cmake_export_link_flags ament_cmake_export_targets ament_cmake_flake8 ament_cmake_gmock ament_cmake_gtest ament_cmake_include_directories ament_cmake_libraries</pre>
특정 패키지 관리 (turtlesim)	 <pre>root@ubuntu-ros: ~ root@ubuntu-ros:~# ros2 pkg executables turtlesim turtlesim draw_square turtlesim mimic turtlesim turtle_teleop_key turtlesim turtlesim_node root@ubuntu-ros:~#</pre>

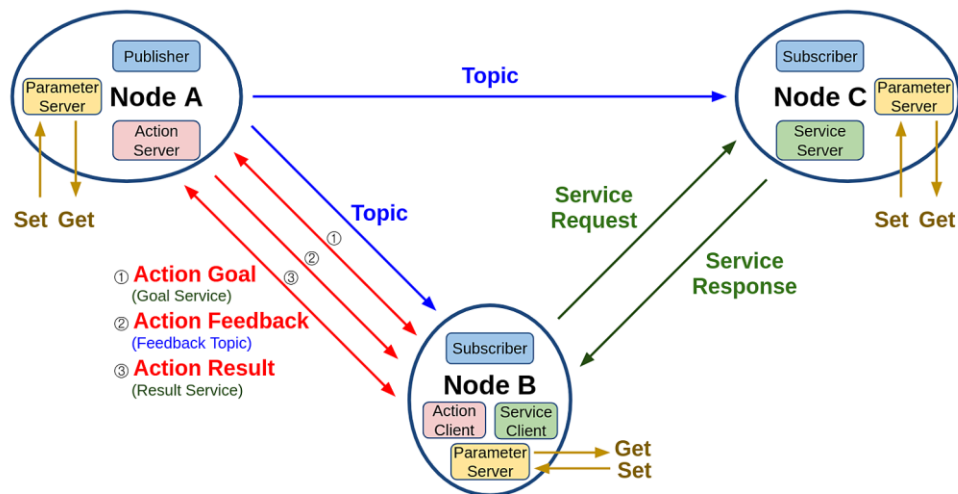




action



parameter

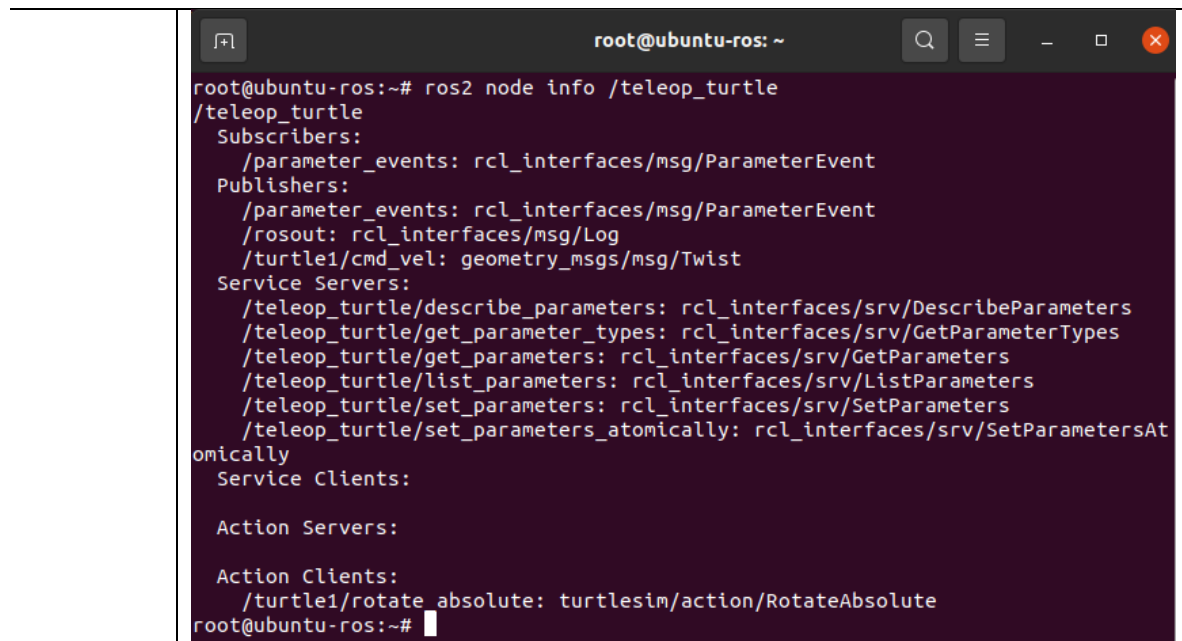


node 정보

```

root@ubuntu-ros: ~
root@ubuntu-ros:~# ros2 node info /turtlesim
/turtlesim
Subscribers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /turtle1/cmd_vel: geometry_msgs/msg/Twist
Publishers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
  /turtle1/color_sensor: turtlesim/msg/Color
  /turtle1/pose: turtlesim/msg/Pose
Service Servers:
  /clear: std_srvs/srv/Empty
  /kill: turtlesim/srv/Kill
  /reset: std_srvs/srv/Empty
  /spawn: turtlesim/srv/Spawn
  /turtle1/set_pen: turtlesim/srv/SetPen
  /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
  /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
  /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
  /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
  /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
  /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomic
Service Clients:
  /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
Action Servers:
  /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
Action Clients:
  /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
root@ubuntu-ros:~#

```

A terminal window titled 'root@ubuntu-ros: ~' with standard Ubuntu window controls. The terminal displays the output of the command 'ros2 node info /teleop_turtle'. The output lists various ROS components associated with the /teleop_turtle node, including subscribers, publishers, service servers, and action clients. The text is as follows:

```
root@ubuntu-ros:~# ros2 node info /teleop_turtle
/teleop_turtle
Subscribers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
  /turtle1/cmd_vel: geometry_msgs/msg/Twist
Service Servers:
  /teleop_turtle/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /teleop_turtle/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /teleop_turtle/get_parameters: rcl_interfaces/srv/GetParameters
  /teleop_turtle/list_parameters: rcl_interfaces/srv/ListParameters
  /teleop_turtle/set_parameters: rcl_interfaces/srv/SetParameters
  /teleop_turtle/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:

Action Servers:

Action Clients:
  /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
root@ubuntu-ros:~#
```