

# Week\_3


## 1. Turtlesim 패키지 설치

- 튜토리얼로 유명한 Turtlesim 패키지를 설치한다.

```
$ sudo apt update
$ sudo apt install ros-humble-turtlesim
```


## 2. Turtlesim이란?

- 터틀심은 2008년의 ROS Tutorial 프로젝트에서 시작한 프로젝트로 2009년에 Turtlesim이라는 이름으로 처음으로 소개되었다.
- turtlesim은 ros\_tutorials 리포지토리의 하위에 속해있다.

 ROS 커뮤니티에서 거북이는 마스코트와 같은 역할을 한다.

## 3. Turtlesim 패키지와 노드

- ROS에서는 프로그램의 재사용성을 극대화하기 위해 최소 단위인 프로세스를 **Node** 라는 이름으로 정의하여 사용한다.
- 하나 이상의 노드 / 노드 실행을 위한 정보들을 모아놓은 것을 **패키지(Package)** 라고 하며
- **패키지**들의 묶음을 **메타패키지(MetaPackage)** 라고 한다.

 자신의 개발환경에 설치된 패키지들을 조회하려면 터미널에  
를 입력하면 된다.

**\$ ros2 pkg list**

- 특정 패키지에 포함된 노드를 확인하는 명령어

**\$ ros2 pkg executables <package>**

ex)

```
kody@desktop:~$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

책에서 나타난 예시와 동일하게 실행되었다.

- 각 노드들에 대한 설명은 다음과 같다
  - draw\_square : 사각형 모양으로 turtle을 움직이게 하는 노드이다
  - mimic : 유저가 지정한 토픽으로 동일 움직임의 turtlesim node를 여러개 실행시킬 수 있는 노드이다.
  - turtle\_teleop\_key : turtlesim\_node를 움직이게 하는 속도 값을 퍼블리시 하는 노드이다.
  - turtlesim\_node : turtle\_teleop\_key로부터 속도를 토픽으로 받아 움직이는 간단한 시뮬레이션 노드이다

## Turtlesim 노드 실행

```
$ ros2 run turtlesim turtlesim_node  
$ ros2 run turtlesim turtle_teleop_key
```

터틀봇 그래픽



터틀봇 teleop

```
kody@desktop:~$ 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.  
█
```

## node, topic, service, action 리스트 조회

- 이 명령들은 현재 어떠한 요소들이 실행되고 있는지 조회하는 명령어이다.

노드 리스트

```
kody@desktop:~$ ros2 node list  
/teleop_turtle  
/turtlesim
```

토픽 리스트

```
kody@desktop:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

service 리스트

```
kody@desktop:~$ 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
```

action 리스트

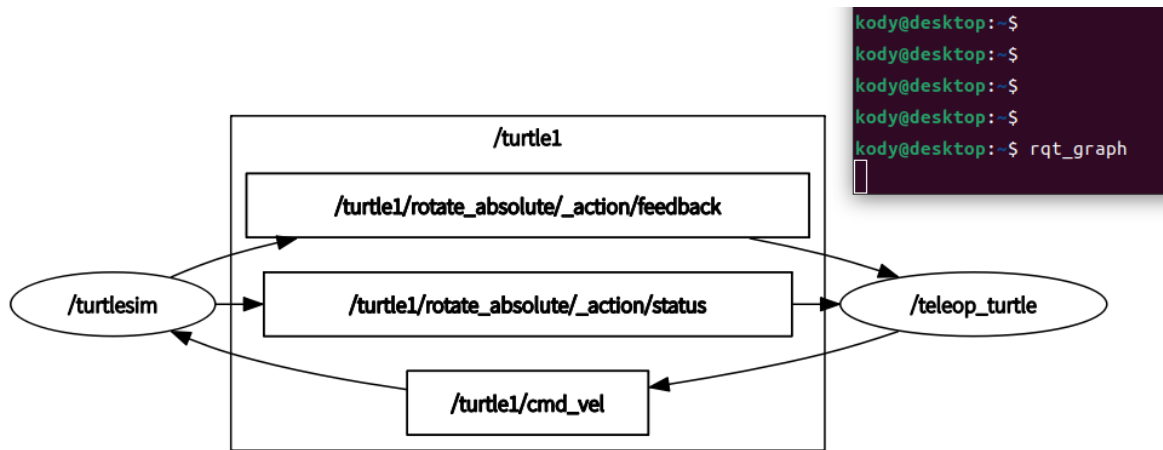
```
kody@desktop:~$ ros2 action list
/turtle1/rotate_absolute
```

## 4.rqt\_graph로 보는 그래픽 뷰

- ROS2 GUI툴을 이용해 rqt\_graph를 실행해 보자

```
$ rqt_graph
```

예제와 같이 잘 실행된다.

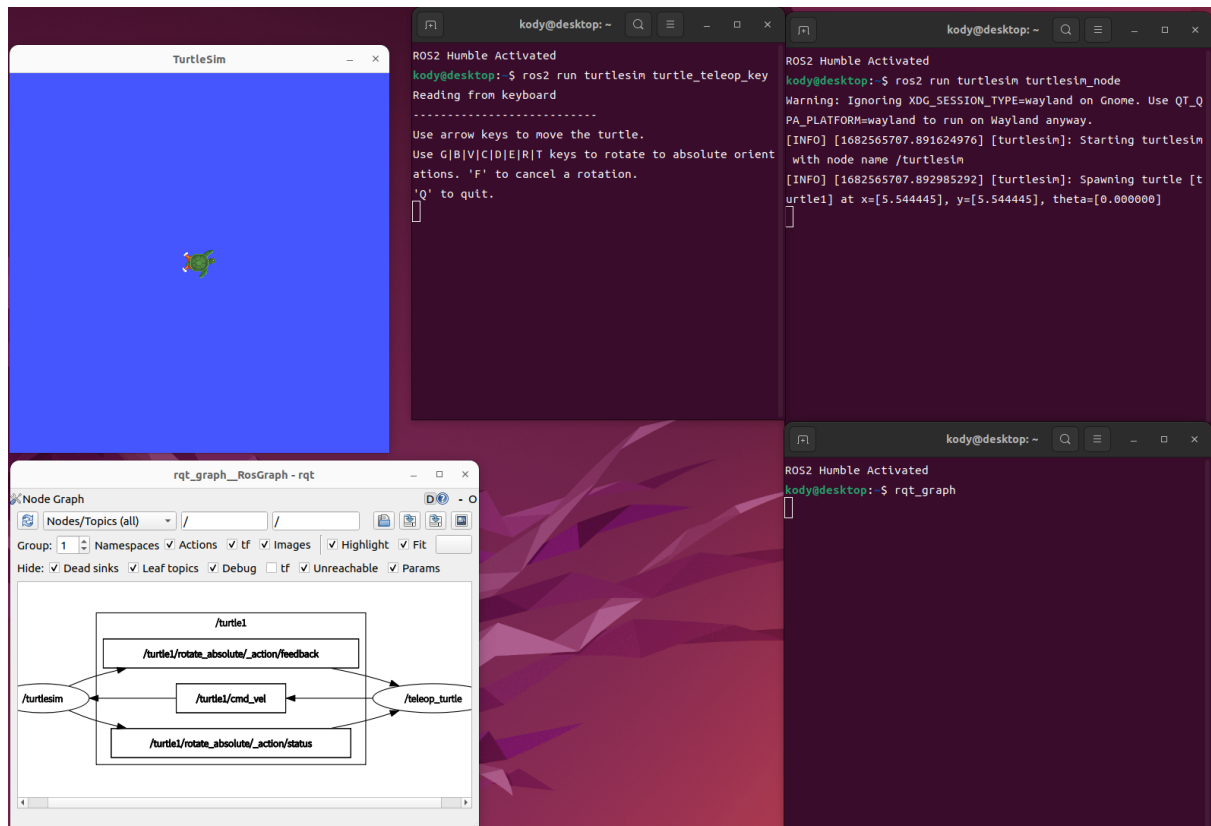


```

kody@desktop:~$ 
kody@desktop:~$ 
kody@desktop:~$ 
kody@desktop:~$ 
kody@desktop:~$ rqt_graph
  
```

## 10장 ROS2 노드와 데이터 통신

- turtlesim\_node
- turtle\_teleop\_key
- rqt\_graph 실행상태에서



\$ ros node list 조회

```
kody@desktop:~$ ros2 node list
/rqt_gui_py_node_12971
/teleop_turtle
/turtlesim
```

\$ ros2 run turtlesim turtlesim\_node \_\_node:=new\_turtle

```
kody@desktop:~$ ros2 run turtlesim turtlesim_node __node:=new_turtle
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway
.
[WARN] [1682566169.614534333] [rcl]: Found remap rule '__node:=new_turtle'. This syntax is deprecated. Use
e '--ros-args --remap __node:=new_turtle' instead.
[INFO] [1682566169.623345084] [new_turtle]: Starting turtlesim with node name /new_turtle
[INFO] [1682566169.625365431] [new_turtle]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], thet
a=[0.000000]
```

새로운 turtlesim\_node의 이름을 new\_turtle으로 지정하여 실행했다.

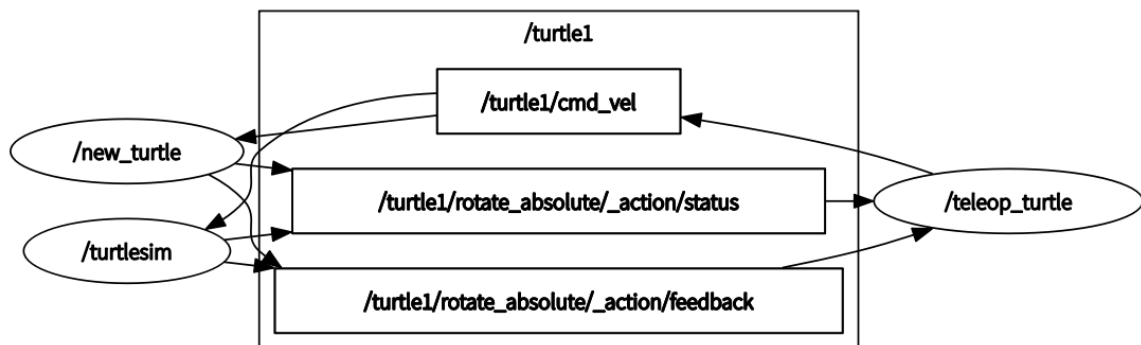


- 두 개의 turtlesim\_node는 teleop\_turtle node를 이용해서 동일하게 움직이도록 할 수 있는데, 둘 다 동일한 토픽을 사용하기 때문이다.

\$ ros2 node list 실행

```
kody@desktop: ~  
ROS2 Humble Activated  
kody@desktop:~$ ros2 node list  
/new_turtle  
/rqt_gui_py_node_12971  
/teleop_turtle  
/turtlesim  
kody@desktop:~$
```

→ new\_turtle 노드가 추가되었다.



- rqt\_graph를 통해서도 확인할 수 있다.

## 노드 정보

```

ROS2 Humble Activated
kody@desktop:~$ 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/SetParametersAtomically
Service Clients:

Action Servers:
  /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
Action Clients:

```

- \$ ros2 node info /turtlesim

ros2 node info <node\_name> 명령어로 노드의 정보를 확인할 수 있다.

/turtlesim의

Subscribers

Publishers

Service Servers

Service Clients

Action Servers

Action Clients를 조회하였다.

\$ ros2 node info /teleop\_turtle

teleop\_turtle에 대한 info를 출력하였다.

```
kody@desktop:~$ 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
```

마찬가지로

Subscribers

Publishers

Service Servers

Service Clients

Action Servers

Action Clients가 출력된다.

## 11장 ROS2 토픽

토픽은 비동기식 단방향 메시지 송수신 방식으로,

- 하나의 토픽을 송수신하는 1:N도 가능하고
- 구성에 따라 N:1, N:N의 방식도 가능하다.
- 원한다면 토픽을 Publish하면서 동시에 Subscribe할수도 있다.
- 토픽은 **비동기성**과 **연속성**을 특징으로 가진다.

```
kody@desktop:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

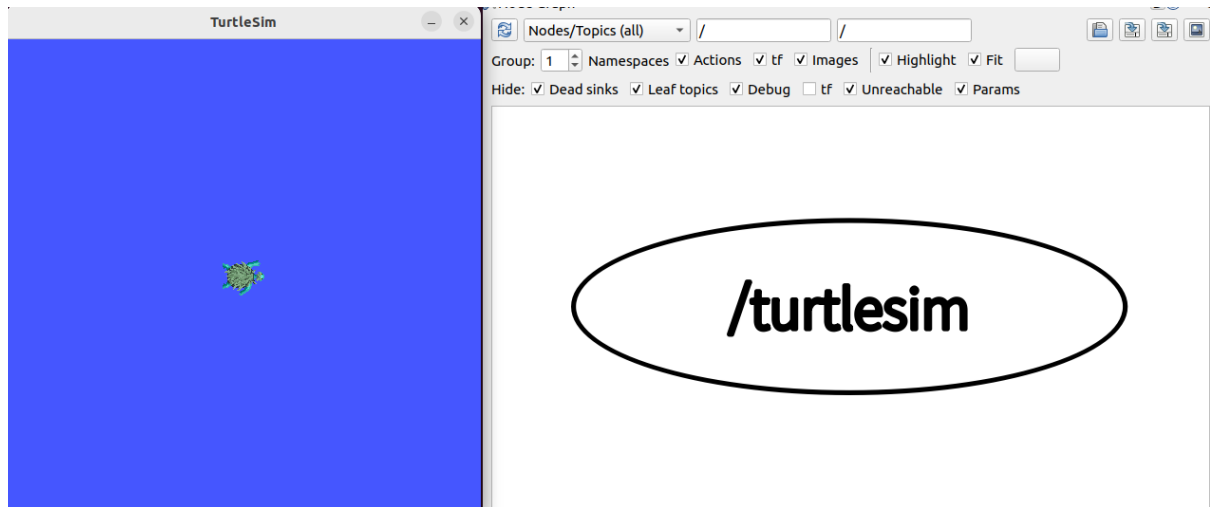
- ros2 topic list명령어를 사용하여 topic 조회



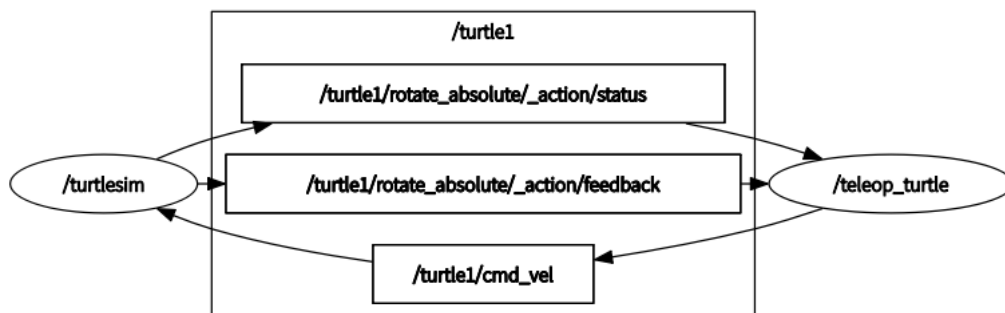
```
kody@desktop:~$ ros2 topic list -t
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
kody@desktop:~$ ros2 topic list
```

- -t 옵션을 사용해서 메시지의 형식(Type)도 표시했다.

turtlesim\_node만 실행한 상태



+teleop\_key도 실행한 상태



- /teleop\_turtle에서 발행하는 cmd\_vel토픽을 /turtlesim 노드에서 Subscribe하고 있음을 알 수 있다.

- 토픽 내용 확인 명령어

```
kody@desktop:~$ ros2 topic echo /turtle1/cmd_vel
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
```

- 토픽 메시지 초당 대역폭 확인 명령어

```
kody@desktop:~$ ros2 topic bw /turtle1/cmd_vel
Subscribed to [/turtle1/cmd_vel]
68 B/s from 2 messages
    Message size mean: 52 B min: 52 B max: 52 B
61 B/s from 3 messages
    Message size mean: 52 B min: 52 B max: 52 B
59 B/s from 4 messages
    Message size mean: 52 B min: 52 B max: 52 B
57 B/s from 5 messages
    Message size mean: 52 B min: 52 B max: 52 B
47 B/s from 5 messages
    Message size mean: 52 B min: 52 B max: 52 B
40 B/s from 5 messages
    Message size mean: 52 B min: 52 B max: 52 B
34 B/s from 5 messages
    Message size mean: 52 B min: 52 B max: 52 B
```

메시지 사이즈가 52바이트임을 알 수 있다.

- 토픽 주기 확인 명령어

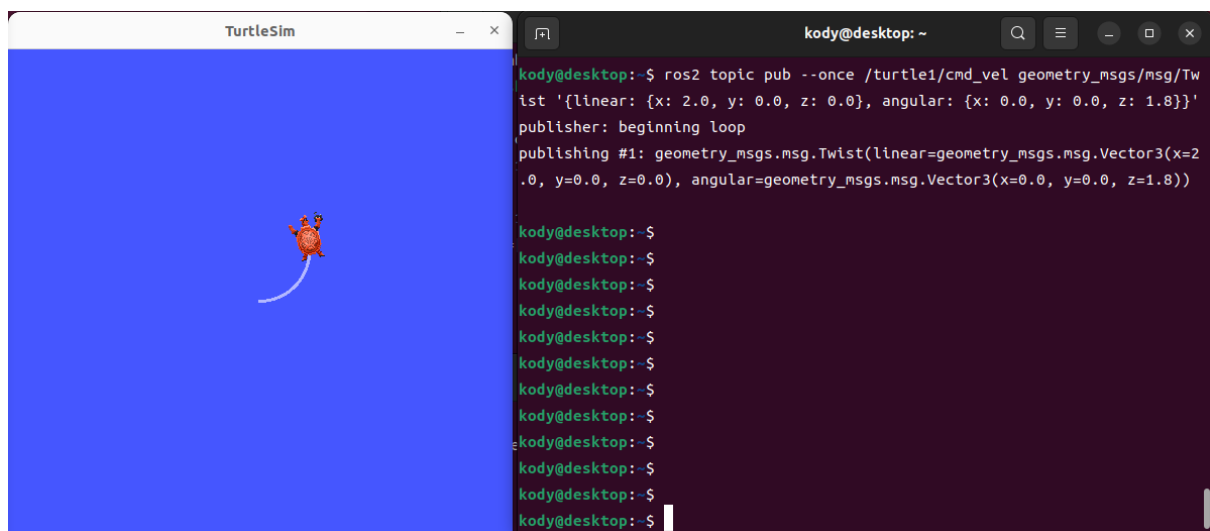
```
kody@desktop:~$ ros2 topic hz /turtle1/cmd_vel
average rate: 33.454
      min: 0.000s max: 0.033s std dev: 0.00526s window: 34
average rate: 33.017
      min: 0.000s max: 0.034s std dev: 0.00382s window: 67
average rate: 32.815
      min: 0.000s max: 0.034s std dev: 0.00317s window: 100
average rate: 32.741
      min: 0.000s max: 0.034s std dev: 0.00277s window: 133
average rate: 32.701
      min: 0.000s max: 0.034s std dev: 0.00251s window: 166
average rate: 32.675
      min: 0.000s max: 0.034s std dev: 0.00230s window: 199
average rate: 32.652
```

최대속도인 33.2hz에 근접하게 테스트해 보았다.

- 토픽의 지연시간을 확인해보려고 했지만 /cmd\_vel 토픽은 헤더를 포함하지 않기 때문에 실행되지 않았다.

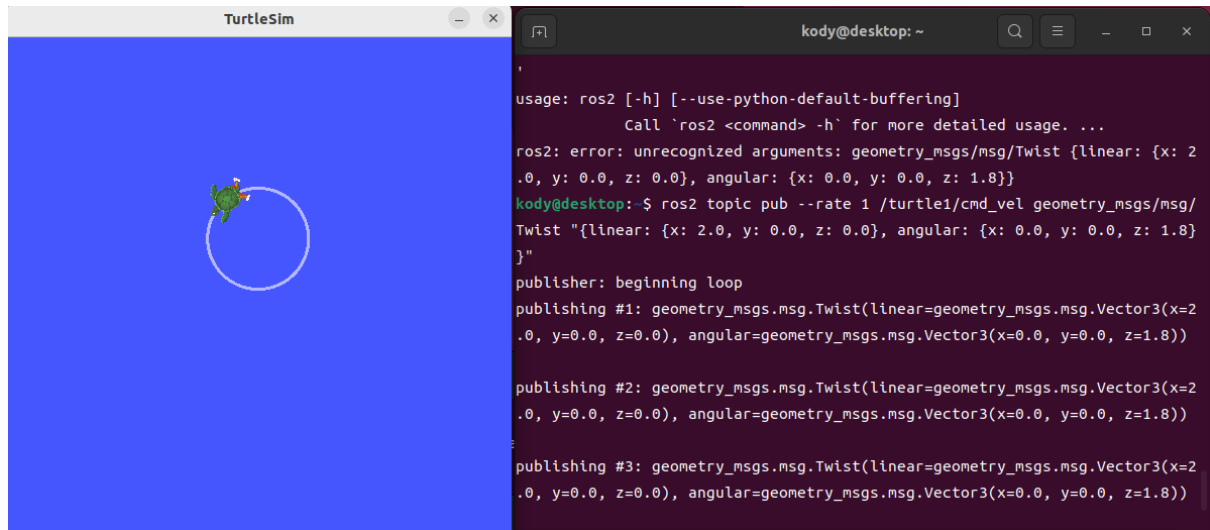
```
kody@desktop:~$ ros2 topic delay /turtle1/cmd_vel
msg does not have header
```

- 토픽 퍼블리시
- 형식: `ros2 topic pub <topic_name> <msg_type> "<args>"`
- ex
  - `$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}'`
- `--once` publish 시현



```
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}'
```

- —rate 1 publish 시현

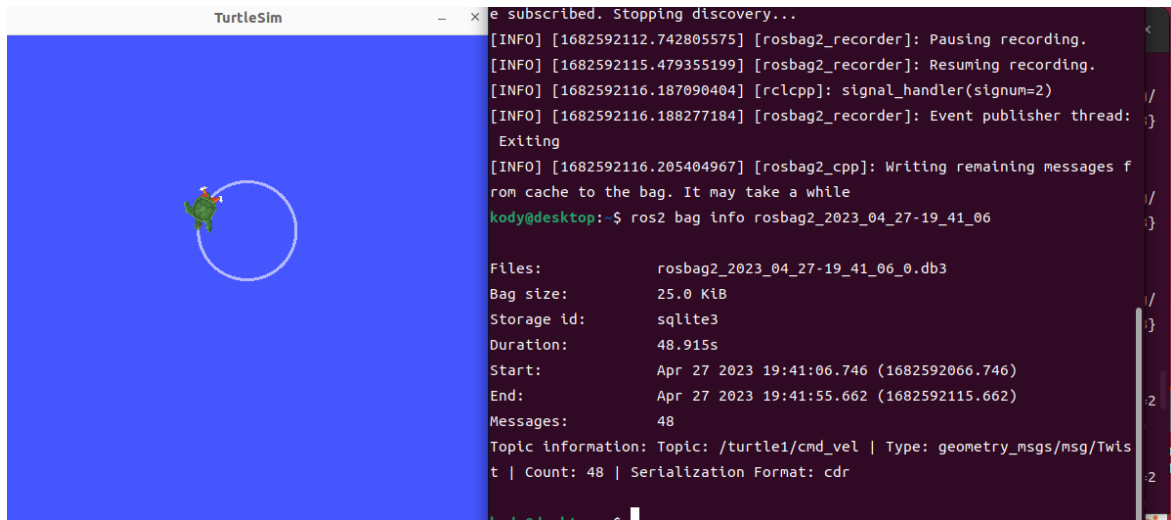


## ros2 bag record (bag기록)

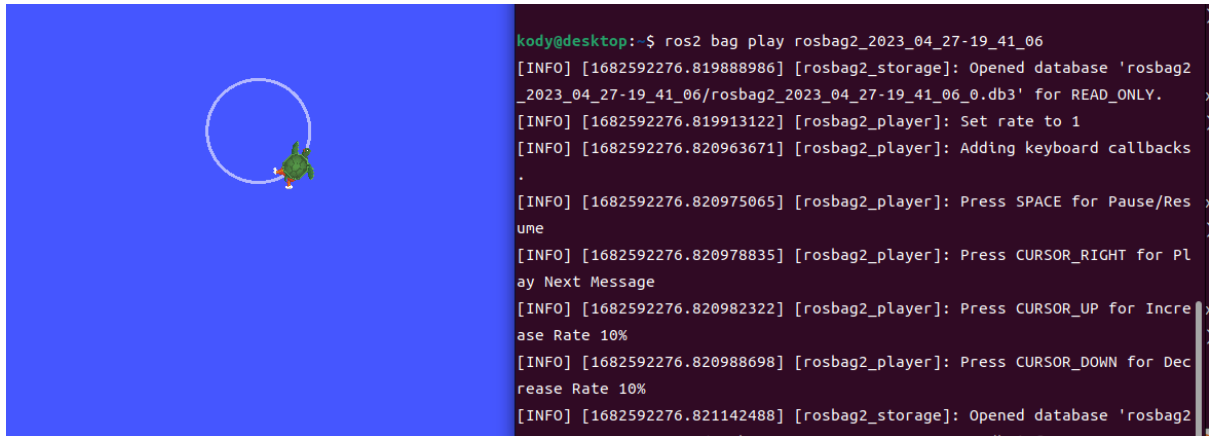
ros2에는 퍼블리시되는 토픽을 파일 형태로 저장하고 필요할 때에 다시 재생할 수 있는 기록이 있으며

이 기능을 **rosbag**이라고 한다. 이는 **디버깅**에 유용하게 활용되는 기능이다.

- ros2 bag info 시현(기록 확인 명령어)



- ros2 bag 재생 시현



## 12장 ROS2 서비스

- 서비스는 동기식 양방향 메시지 송수신 이다.
- 서비스는 요청을 보내고 응답을 기다리며, 서비스 서버는 요청을 받고 처리한 다음 응답을 반환하고, 이 과정은 동기적이다. 즉, 서비스 클라이언트가 서비스 서버로부터 응답을 받을 때까지 기다리기 때문에 동기적이다.
- Service Client는 Service Server에 Service Request를 보내고
- Service Server는 다시 Service Client에게 Service Response를 반환한다.
- Service list 명령어

```
kody@desktop:~$ ros2 service list
/clear
/kill
/reset
/spawn
/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
```

## 서비스 형태 조회

```
kody@desktop:~$ ros2 service type /clear
std_srvs/srv/Empty
kody@desktop:~$ ros2 service type /kill
turtlesim/srv/Kill
kody@desktop:~$ ros2 service type /spawn
```

- 옵션으로 -t를 붙이면 서비스 이름과, 서비스 형태의 목록을 볼 수 있다.
- /~~~(서비스 이름) [메시지 타입] 형태로 출력한다.

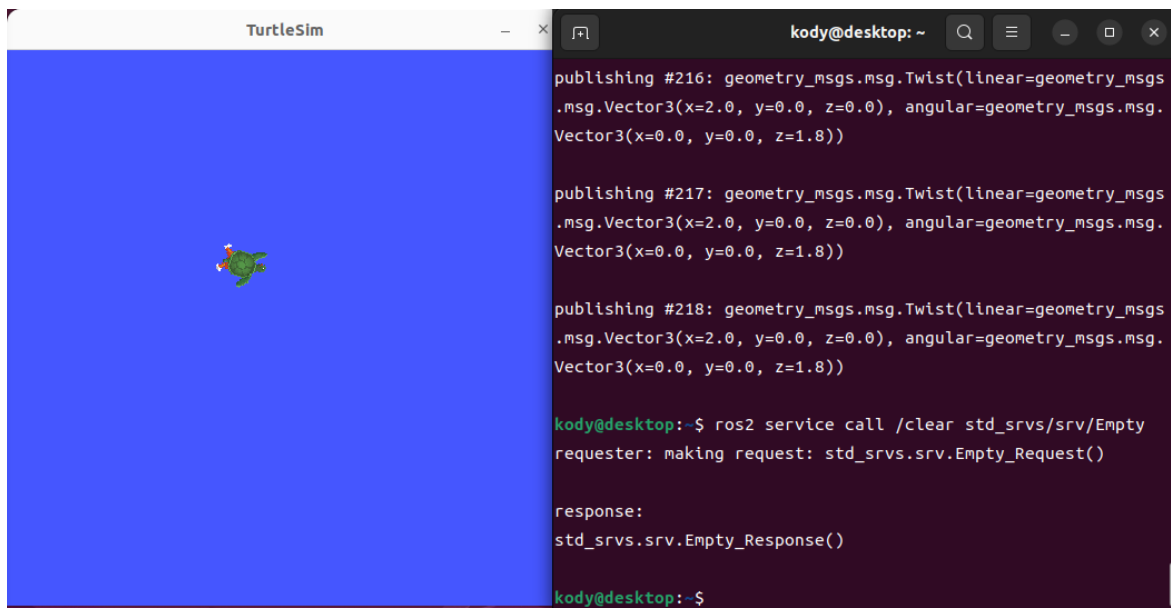
```
kody@desktop:~$ ros2 service list -t
/clear [std_srvs/srv/Empty]
/kill [turtlesim/srv/Kill]
/reset [std_srvs/srv/Empty]
/spawn [turtlesim/srv/Spawn]
/turtle1/set_pen [turtlesim/srv/SetPen]
/turtle1/teleport_absolute [turtlesim/srv/TeleportAbsolute]
/turtle1/teleport_relative [turtlesim/srv/TeleportRelative]
/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/SetParametersAtomically]
```

반대로 [메시지 타입]을 입력하여 서비스 이름을 출력할 수도 있다.

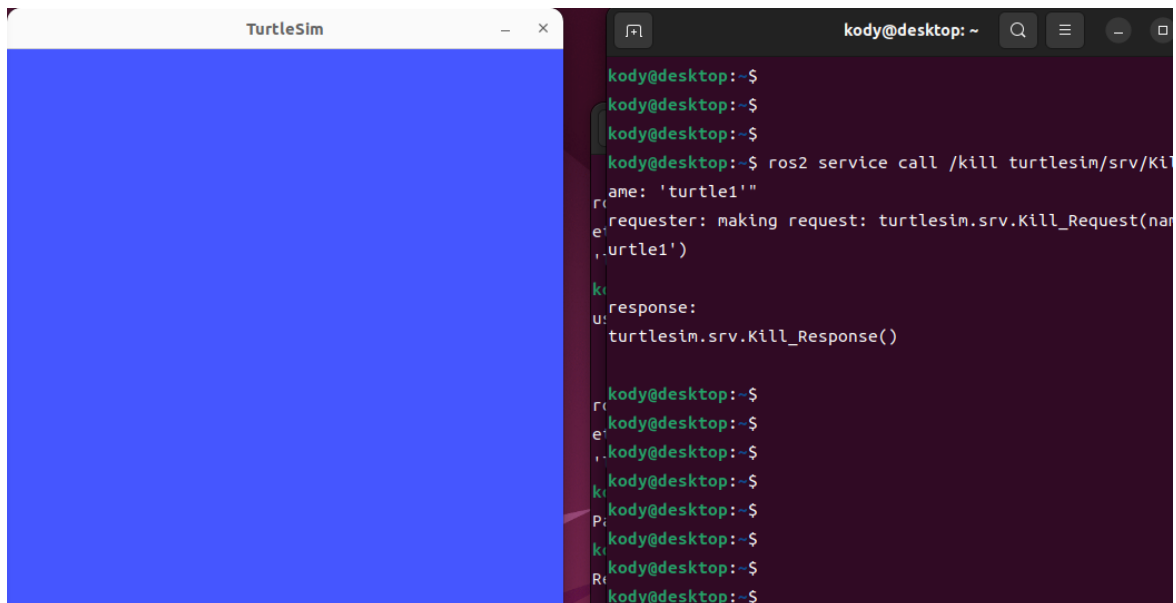
```
kody@desktop:~$ ros2 service find std_srvs/srv/Empty
/clear
/reset
kody@desktop:~$ ros2 service find turtlesim/srv/KILL
kody@desktop:~$ ros2 service find turtlesim/srv/Kill
/kill
```

## 서비스 요청

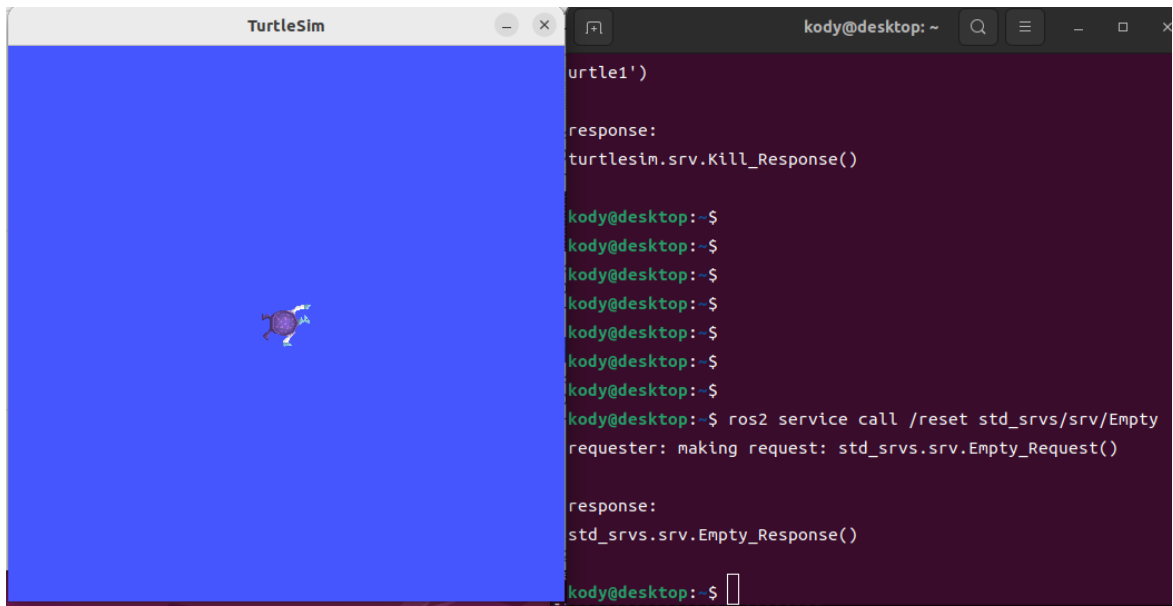
- 형식 : `ros2 service call <service_name> <service_type> "<arguments>"`
- ex) `$ ros2 run turtlesim turtle_teleop_key`
- /clear이라는 이름의, std\_srv/srv/Empty 타입의 서비스를 호출했다.



- /kill서비스 호출



- /reset 서비스 호출 → 새로운 거북이가 표시되었다.



- 펜 그리기 예제
- `ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{"r": 255, "g": 255, "b": 255, "width": 10}'`

