Week3

9장 패키지 설치와 노드 실행

Turtlesim

turtlesim은 ROS를 사용하는 사람들이 튜토리얼로써 많이 애용하는 패키지이다.

이번 장에서는 turtlesim 패키지를 가지고 패키지 설치와 노드 실행을 진행해 보겠다.

- Turtlesim이란?

turtlesim은 ros_tutorials 리포지터리 하위에 속해 있으며 ROS 학습용 패키지로 제작되었다.

turtlesim 패키지의 이름에는 거북이(Turtle)이 사용되었는데 이 패키지안의 turtlesim_node 노드를 실행하면 나오는 아이콘도 거북이(Turtle)이 사용되고 ROS 커뮤니티의 공식 모바일 로봇 플랫폼으로 널리 사용 중인 TurtleBot에도 turtle이 포함될 정도로 ROS 커뮤니티에서 거북이는 마스코드 역할을 한다.

Turtle의 기원은 1940년대에 컴퓨터 화면에 움직이는 대로 라인을 그려 그림을 그릴 수 있는Turtlegraphics 프로그램에서 유래되었다. 이는 아이들의 컴퓨터 프로그래밍 언어 교육에 사용되었는데, 쉽게 프로그래밍 언어를 가르치자는 취지의 개념이 함축된 Turtle이 이후 다른 여러 프로그램에서도 쓰이게 되었다.

Turtlesim 패키지와 노드

ROS에서는 프로그램의 재사용의 극대화를 위해 최소 단위의 실행 가능한 프로세스인 노드(Node)단위로 프로그램을 작성하게 된다. 하나 이상의 노드 실행을 위한 정보 등을 묶어 놓은 것을 패키지(Package)라고 하며, 패키지의 묶음의 메타패키지(Metapackage)라 한다.

자신의 개발환경에 어떠한 패키지가 있는지 확인하려면 아래의 ROS2 CLI를 사용하면 된다.

```
$ ros2 pkg list
```

turtlesim 패키지에 포함된 노드가 어떤 것이 있는지 확인하려면 ros2 pkg executables <패키지명> 을 사용하면 된다.

```
$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

- draw_square : 사각형 모양으로 turtle을 움직이게 하는 노드
- mimic : 유저가 지정한 토픽으로 동일 움직임의 turtlesim_node를 복수 개 실행시킬 수 있는 노드
- turtle_teleop_key : turtlesim_node 를 움직이게 하는 속도 값을 퍼블리시 하는 노드
- turtlesim_node : turtle_teleop_key로부터 속도 값을 토픽으로 받아 움직이게 하는 간단한 2D 시뮬레이션 노드

Turtlesim 패키지의 노드 실행

노드를 실행할 때는 ros2 run으로 실행하고 <패키지명> <노드명> 순서로 작성하면 된다.

```
$ ros2 run turtlesim turtlesim_node
```

```
$ 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.
```

노드, 토픽, 서비스, 액션의 조회

ROS2에서는 현재 실행되는 노드와 발행되는 토픽, 서비스, 액션을 각각의 ROS2 CLI를 통해 조회할 수 있다. 현재 turttlesim_node와 turtle_teleop_key 두 노드가 실행되고 있을 때 다음과 같이 조회된다.

```
$ ros2 node list
/teleop_turtle
/turtlesim
```

```
$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

```
$ 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
```

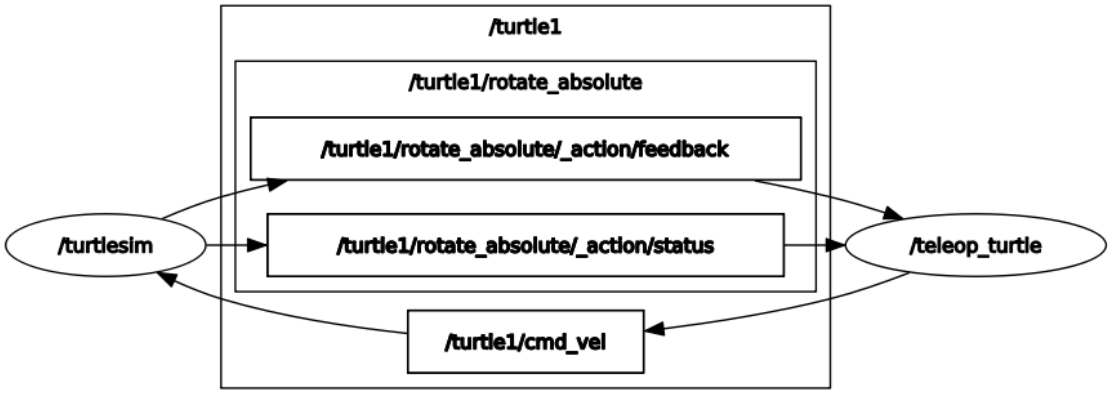
```
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
```

```
$ ros2 action list
/turtle1/rotate_absolute
```

rqt_graph로 보는 노드와 토픽의 그래프 뷰

ROS2 에서는 rqt_graph로 실행되고 있는 노드와 토픽을 시각적으로 볼 수 있다.

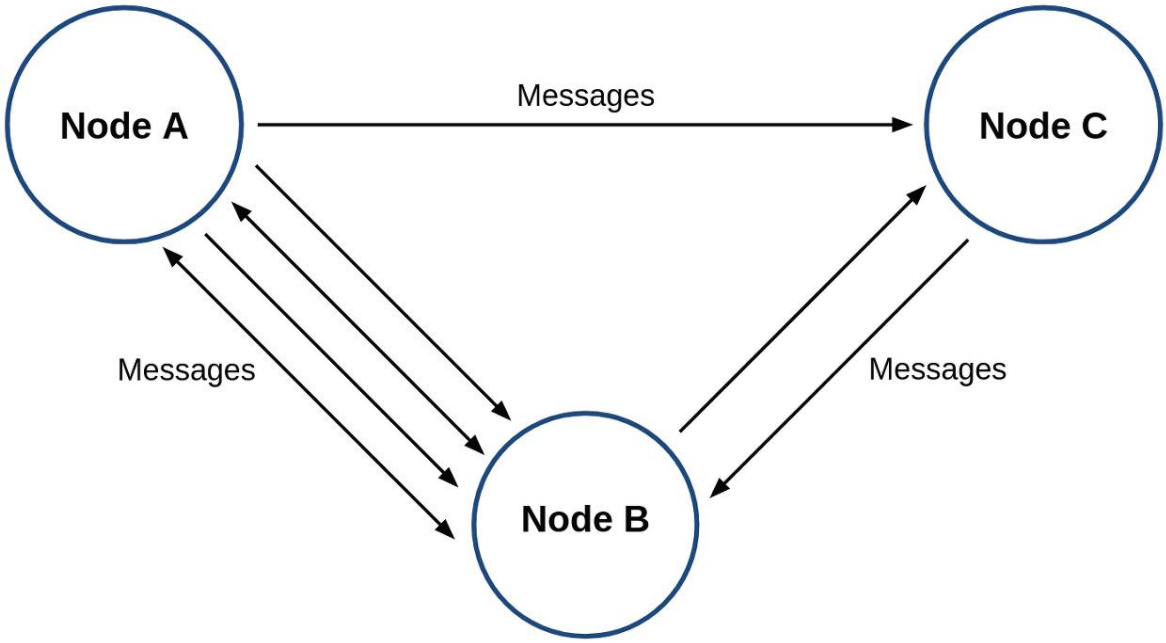
```
$ rqt_graph
```



그림에서 보이는 동그라미가 노드이고 네모는 토픽, 또는 액션이며, 화살표는 메시지의 방향이다. 서비스는 순간적으로 사용되는 형식이라 표시되지 않는다.

10장 ROS2 노드의 데이터 통신

노드(Node)는 최소단위의 실행가능한 프로세스를 가리키며 ROS에서는 노드 단위로 프로그램을 나누어 작업하게 된다. 하나의 ROS시스템은 수많은 노드들이 존재하는데 노드와 노드 사이에 입출력데이터를 서로 주고받게 설계되있다. 여기서 주고 받는 데이터를 ROS에서는 메시지(Message)라고 하고 이 메시지를 주고 받는 통신 방법에 따라 토픽(Topic), 서비스(Service), 액션(Action), 파라미터(Parameter)노 나눈다.



노드 실행

ros2 run <패키지명> <노드명> 명령어를 사용하면 된다.

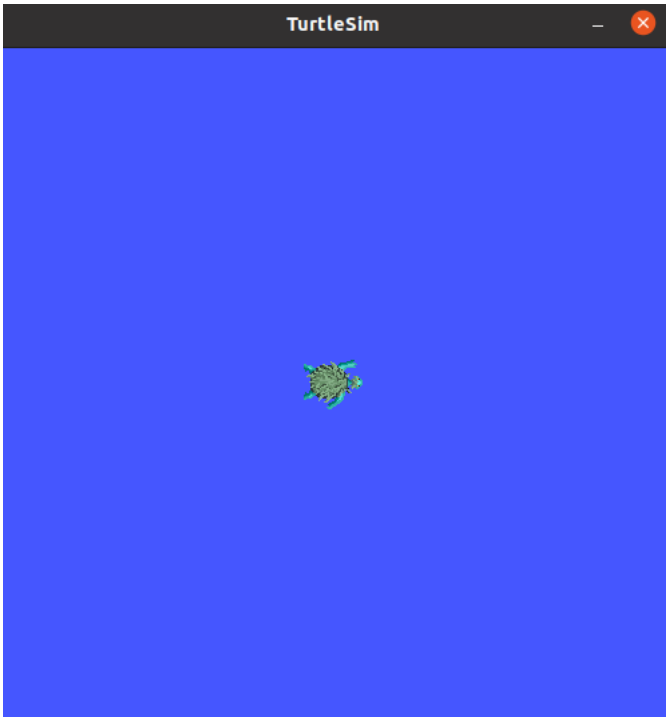
```
$ ros2 run turtlesim turtlesim_node
```

```
$ 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.
```

만약 동일 노드를 복수 개 실행시키려면 같은 실행 코드를 입력해도 되지만 동일한 노드 이름으로 실행되기 때문에 시스템 구성에서 좋지 못하다. 만약 노드명을 달리하고 싶다면 실행 코드 뒤에 __node:=<새로운 노드명>을 붙여서 사용하면 된다.

```
$ ros2 run turtlesim turtlesim_node __node:=new_turtle
[WARN] [1689778739.218468584] [rcl]: Found remap rule '__node:=new_turtle'. This syntax is deprecated. Use '--ros-args --remap __node:
[INFO] [1689778739.237841429] [new_turtle]: Starting turtlesim with node name /new_turtle
[INFO] [1689778739.241885590] [new_turtle]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

위의 [WARN] 항목을 통해 현재 foxy 이후부터는 --ros-args --remap __node:=new_turtle 의 형식으로 하는 것이 권장된다는 사실을 알게 되었다.



```
$ ros2 node list
/new_turtle
/teleop_turtle
/turtlesim
```

실행시키면 위와 같이 new_turtle이라는 이름으로 노드가 실행됨을 알 수 있다.

노드 정보(ros2 node info)

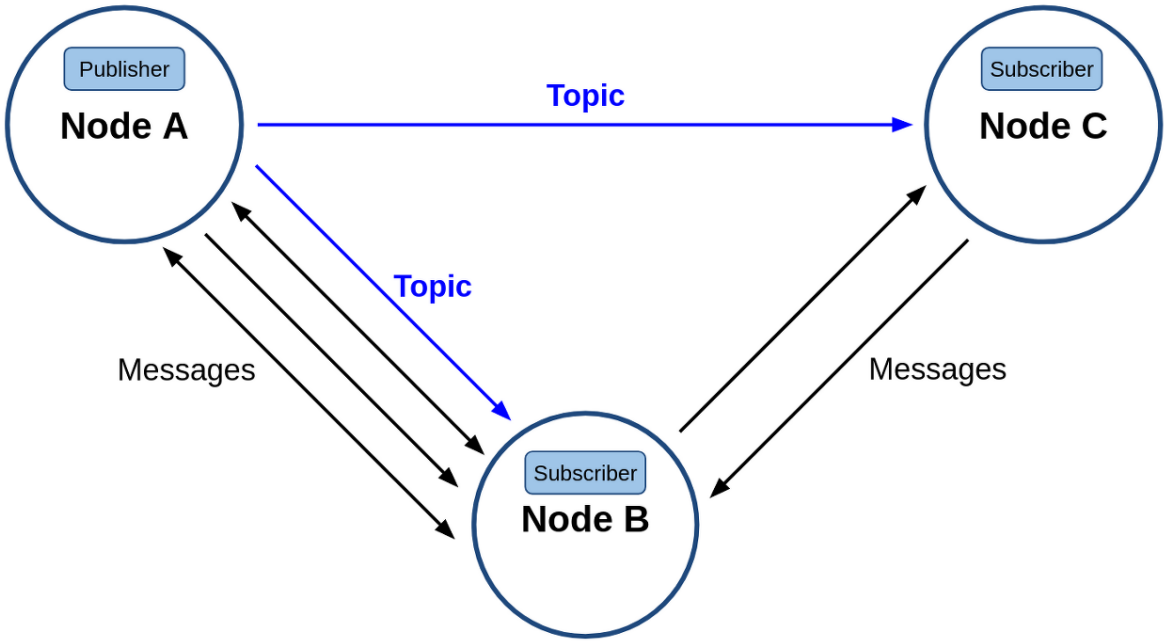
노드의 정보를 확인하려면 ros2 node info <노드명> 을 실행시키면 된다. 이 정보에는 노드의 Publisher, Subscriber, Service, Action, Parameter 정보를 확인 할 수 있다.

```
$ 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:
```

11장 ROS2 토픽

토픽(Topic)은 노드 간의 비동기식 단방향 메시지 송수신 방식으로 msg 인터페이스 형태의 메시지를 퍼블리시하는 Publisher와 메시지를 서브스크라이브하는 Subscriber 간의 통신이다. 1:1통신을 기본으로 하지만 1:N, N:1, N:N 통신도 가능하다. 토픽은 ROS 메시지 통신에서 가장 널리 사용되는 방식이다.



하나의 노드는 여러개의 Publisher와 여러개의 Subscriber를 생성해서 통신할 수 있다.

아까의 turtlesim과 turtle_teleop_key 노드를 실행했을 때 토픽 목록 확인은 ros2 topic list로 확인할 수 있다.

```
$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

토픽안의 메시지 형태를 알고 싶다면 -t 명령어를 추가하면 된다.

```
$ 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]
```

토픽 정보 확인(ros2 topic info)

토픽의 정보 확인은 ros2 topic info <토픽명> 명령어를 통해 가능하다. 명령어를 실행시 토픽 메시지 형태, 토픽의 퍼블리셔, 서브스크라이버 정보를 확인 가능하다.

```
$ ros2 topic info /turtle1/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 1
Subscription count: 2
```

토픽 내용 확인(ros2 topic echo)

토픽 내용의 실시간확인은 ros2 topic echo <토픽명> 명령어를 통해 가능하다.

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

위의 정보를 통해 /turtle1/cmd_vel 토픽은 linear에 x,y,z, angular에 x,y,z 값이 총 6개의 데이터로 구성되어있음을 알 수 있다. 참고로 모든 메시지는 SI단위(meter, second, degree, kg등)를 기본으로 사용한다.

ros2 topic bw 명령어를 통해 토픽의 대역폭, 즉 송수신 받는 메시지의 크기를 확인할 수 있다.

```
$ ros2 topic bw /turtle1/cmd_vel
Subscribed to [/turtle1/cmd_vel]
184 B/s from 2 messages
        Message size mean: 52 B min: 52 B max: 52 B
```

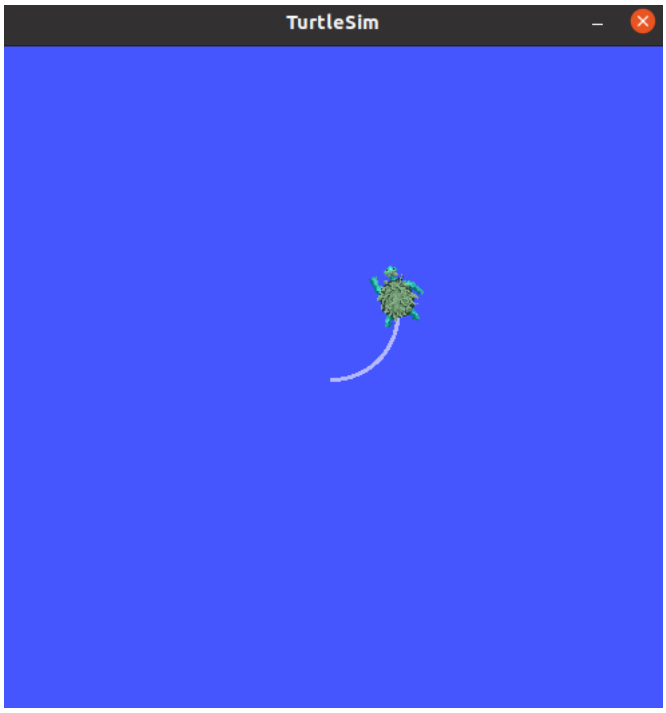
그밖에 토픽 주기 확인(ros2 topic hz), 토픽 지연 시간 확인(ros2 topic dalay)가 있다.

토픽 퍼블리시(ros2 topic pub)

노드안의 퍼블리셔에서 토픽을 발행하지 않고 명령창에서 직접 토픽을 퍼블리시 할 수 있다.

```
$ 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}
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0,
```

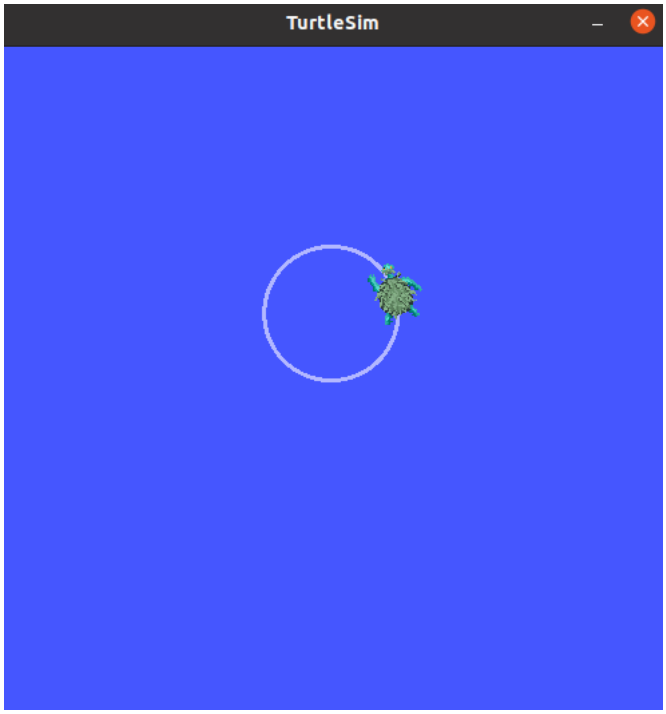
💡 위의 “메시지” 항목 작성시에 : 뒤에 스페이스로 분리를 해줘야 한다.



토픽명 앞에 '—once' parameter항목을 '—rate 1' 와 같이 수정하면 1hz의 주기로 토픽을 계속 퍼블리시 한다.

```
$ 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.
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0,

publishing #2: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0,
```



bag 기록(ros2 bag record)

ROS2에는 퍼블리시 되는 토픽을 파일 형태로 저장하고 필요할 때 저장된 토픽을 다시 불러와 동일한 주기로 재할 수 있는 rosbag기능이 있다. 로봇의 기능 검증을 위해 매번 구동시켜서 데이터를 취득하는 것은 매우 힘든 과정이고 테스트마다 센서 정보나 로봇의 상태값에 따라 결과 값이 달라지기에 같은 결과가 도출되기 어렵다. 따라서 rosbag 기능을 이용하여 입력값을 고정하고 반복 테스트하여 알고리즘 개선 작업이나 성능 검증 테스트를 진행하는 것이 매우 효율적이다.

rosbag 실행 명령어는 기본적으로 ros2 bag record <토픽명1> <토픽명2> … 로 하면 된다.

기록 종료는 프로그램을 종료하면 된다.

```
$ ros2 bag record /turtle1/cmd_vel
[INFO] [1689781817.701522451] [rosbag2_storage]: Opened database 'rosbag2_2023_07_20-00_50_17/rosbag2_2023_07_20-00_50_17_0.db3' for R
[INFO] [1689781817.712184208] [rosbag2_transport]: Listening for topics...
[INFO] [1689781817.713712800] [rosbag2_transport]: Subscribed to topic '/turtle1/cmd_vel'
```

rosbag의 정보 확인은 ros2 bag info <rosbag 파일명>을 사용하면 기록 타임스탬프와 취득한 토픽의 이름, 메시지 형태, 메시지 별 개수 등이 기록되어 있다.

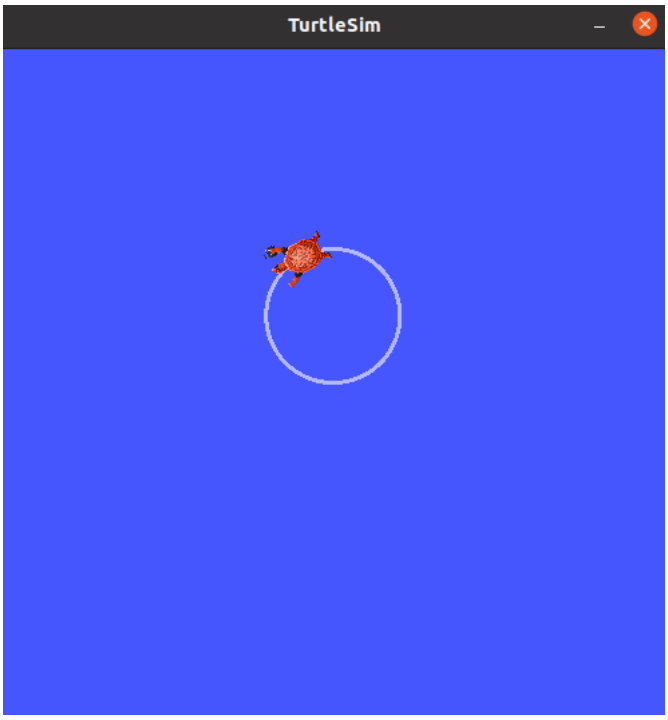
```
$ ros2 bag info rosbag2_2023_07_20-00_50_17/

Files:             rosbag2_2023_07_20-00_50_17_0.db3
Bag size:          16.8 KiB
Storage id:        sqlite3
Duration:          8.999s
Start:             Jul 20 2023 00:50:18.197 (1689781818.197)
```

```
End:                Jul 20 2023 00:50:27.197 (1689781827.197)
Messages:           10
Topic information: Topic: /turtle1/cmd_vel | Type: geometry_msgs/msg/Twist | Count: 10 | Serialization Format: cdr
```

rosbag의 재생은 ros2 bag play <rosbag 파일명>을 사용하면 된다. 실행하게 되면 해당 토픽이 재생됨을 볼 수 있다.

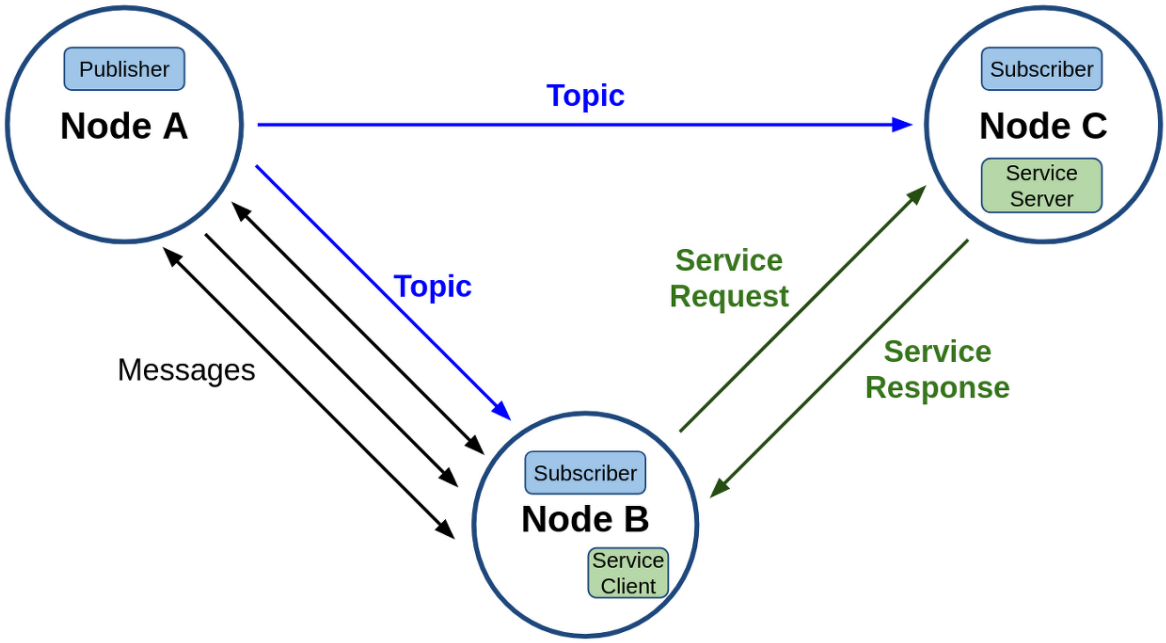
```
$ ros2 bag play rosbag2_2023_07_20-00_50_17/
[INFO] [1689782177.623294543] [rosbag2_storage]: Opened database 'rosbag2_2023_07_20-00_50_17//rosbag2_2023_07_20-00_50_17_0.db3' for
```



rosbag를 실행시키면 새로 실행한 turtlesim이 해당 토픽을 받아서 같은 움직임을 보이는 것을 볼 수 있다.

12장 ROS2 서비스

서비스(Service)는 노드 간의 동기식 양방향 메시지 송수신 방식으로 서비스 요청(Request)을 하는 쪽을 Service Client, 요청받은 서비스를 수행한 후 서비스 응답(Response)하는 쪽을 Service Server라고 한다. 서비스 요청과 응답은 msg 인터페이스의 변형인 srv 인터페이스를 사용한다.



서비스는 동일한 서비스 서버에 대해 복수의 클라이언트를 가질 수 있다. 단, 서비스 응답은 서비스 요청이 있었던 클라이언트에게만 응답하는 형식이다.

서비스 목록확인은 ros2 service list 으로 할 수 있다.

```
$ 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
```

토픽과 마찬가지로 -t 로 서비스 형태로 같이 출력할 수 있다.

서비스 형태 확인은 ros2 service type <서비스명> 으로 할 수 있다.

```
$ ros2 service type /spawn
turtlesim/srv/Spawn
```

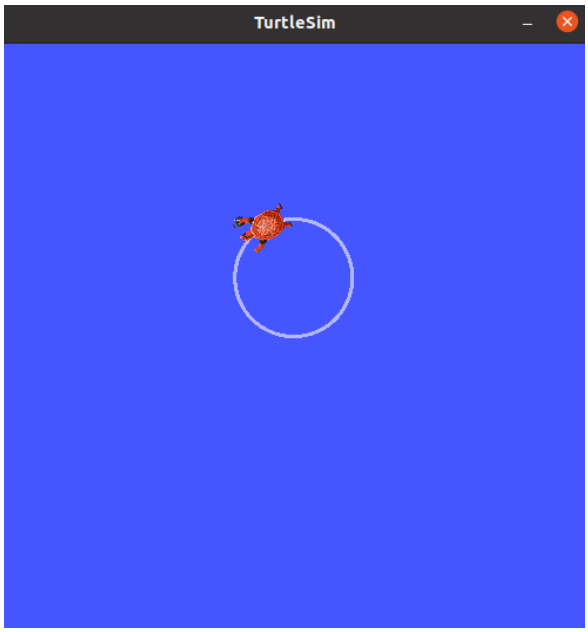
서비스 찾기는 서비스 형태 확인과 달리 형태를 입력해서 해당 서비스 형태를 사용하는 서비스 명을 확인 할 수 있다. 명령어는 ros2 service find <서비스 형태>의 형식을 사용한다.

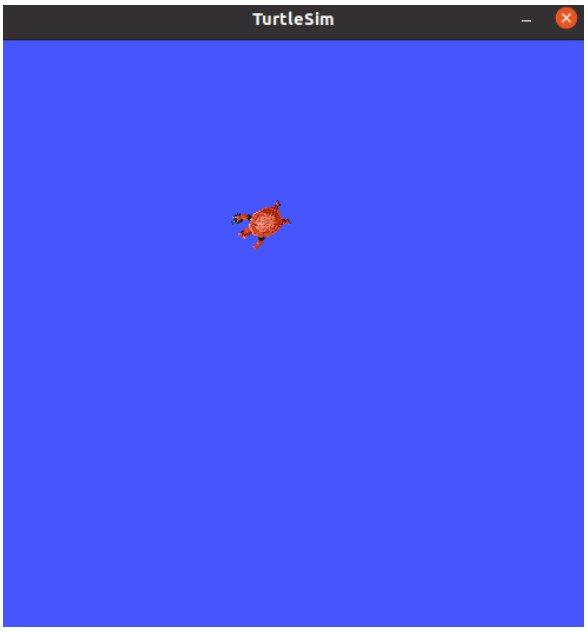
```
$ ros2 service find turtlesim/srv/Spawn
/spawn
```

명령창에서 직접 서비스 서버에 서비스 요청을 할 수 있다. 명령어는 ros2 service call <서비스명> <서비스형태> “<서비스 요청 내용>” 의 형식을 사용한다.

```
$ ros2 service call /clear std_srvs/srv/Empty
requester: making request: std_srvs.srv.Empty_Request()

response:
std_srvs.srv.Empty_Response()
```





/clear 라는 서비스를 요청하면 turtlesim에 그려져있던 경로가 지워짐을 볼 수 있다.