

## 스터디 13주차 – ROS CLI

### 홍송은

#### [Service]

- ROS 그래프에서 노드 간의 또 다른 통신 방법
- 동기식 양방향 통신 방식
  - 클라이언트 노드가 서버 노드에 명시적인 요청(Request)을 보내면, 서버는 이에 대한 응답(Response)을 반환
- 토픽의 게시자-구독자 모델이 아닌 **호출(Request)-응답(Response) 모델 기반**
- 노드들은 서비스 서버(Service Server)와 서비스 클라이언트(Service Client)로 나뉘며, 클라이언트가 서버에 요청을 보내고, 서버는 처리 결과를 응답으로 반환

#### 특징

- 호출-응답 기반 통신 방식
  - 토픽(Topic)의 게시-구독(pub-sub) 모델과는 달리, 서비스는 명확한 요청이 있을 때에만 동작
- 1:1 통신
  - 요청에 대한 응답은 오직 요청을 보낸 클라이언트에게만 돌아가며, 동시에 여러 요청 처리 불가
  - 서비스 서버는 하나의 요청이 끝난 후에야 다음 요청 처리 가능
- 단발성 명령 수행
  - 지속적인 데이터 스트리밍보다는 단발성 요청 처리에 적합
  - 빠르게 수행되고 종료되는 동작에 알맞음
- 다수의 클라이언트가 하나의 서버에 요청 가능
  - 여러 노드가 하나의 서비스 서버를 공유해 요청할 수 있으나, 서버는 순차적으로만 응답 처리

#### Topic과의 차이점

구분	Topic	Service
통신 방식	Publish/Subscribe (게시/구독)	Request/Response (요청/응답)
통신 방향	1:N (여러 구독자에게 전송)	1:1 (단일 요청-응답 관계)
동작 방식	<b>지속적인 데이터 흐름, 주기적 전송 가능</b>	<b>단발성 명령 수행, 요청이 있을 때만 동작</b>
사용 목적	센서 스트리밍, 상태 전파 등	명령 실행, 설정 변경 등
응답 여부	응답 없음 (일방향 전송)	명시적 응답 존재

- 명확한 요청의 주체가 존재하고, 작업이 빠르게 완료되어야 하는 경우 → Service
- 불특정 다수의 수신자가 존재하고, 데이터를 주기적으로 계속 전송해야 하는 경우 → Topic

상황	적절한 ROS 통신 방식
로봇 팔에 "지금 물건을 잡아라"라고 지시 → 명령 후 즉시 응답 필요	Service
카메라 이미지 프레임을 계속 송신	Topic
거북이에게 현재 위치로 순간이동 명령	Service
IMU 센서에서 가속도 데이터를 실시간 전송	Topic

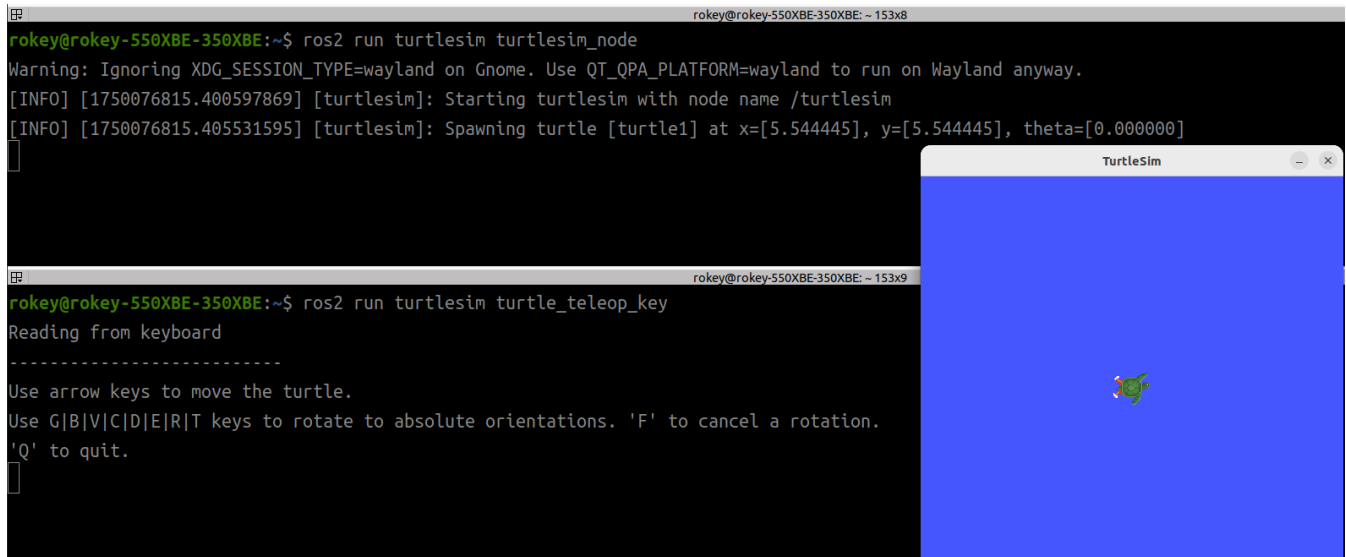
## 환경 준비

### 1. turtlesim\_node 실행

```
ros2 run turtlesim turtlesim_node
```

### 2. turtle\_teleop\_key 실행

```
ros2 run turtlesim turtle_teleop_key
```



```
rokey@rokey-550XBE-350XBE:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[INFO] [1750076815.400597869] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1750076815.405531595] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

rokey@rokey-550XBE-350XBE:~$ 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.
```

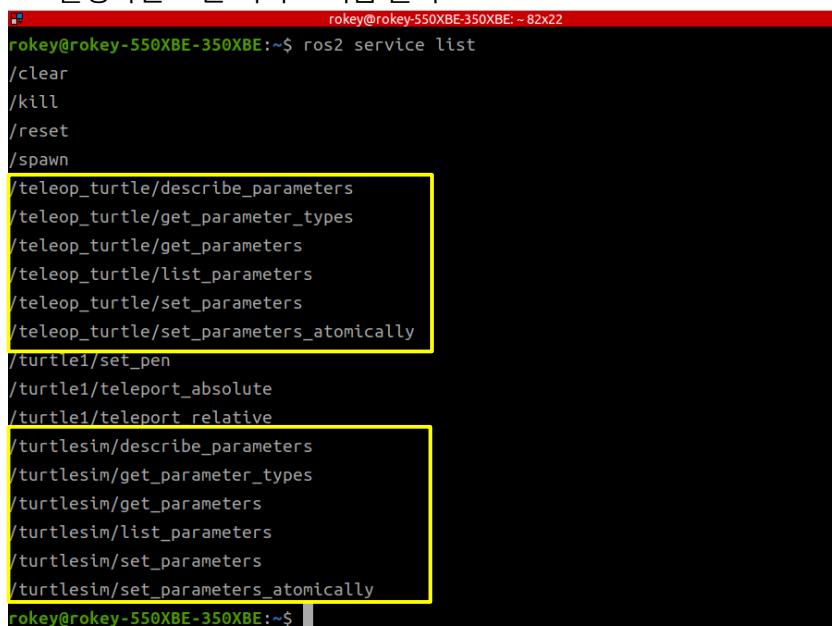
## Tutorial - Understanding services

<https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Services/Understanding-ROS2-Services.html>

### 1. 서비스 목록 조회

```
ros2 service list
```

- 활성화된 모든 서비스 이름 출력



```
rokey@rokey-550XBE-350XBE:~$ 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
rokey@rokey-550XBE-350XBE:~$
```

## 2. 서비스 타입 확인

```
ros2 service type <service_name>
```

```
rokey@rokey-550XBE-350XBE: ~ 82x22
rokey@rokey-550XBE-350XBE:~$ ros2 service type /clear
std_srvs/srv/Empty
```

- Empty: 요청, 응답 없는 서비스 타입
- /clear: 거북이가 그린 선을 지우는 기능. 인자 없이 호출.

서비스 이름	서비스 타입	입력 인자	특징
/spawn	turtlesim/srv/Spawn	O	거북이 생성 시 위치(x, y, $\theta$ ), 이름 입력 필요
/turtle1/set_pen	turtlesim/srv/SetPen	O	색상과 선 두께 지정, 펜 on/off 제어 가능
/turtle1/teleport_absolute, /teleport_relative	TeleportAbsolute, TeleportRelative	O	이동 위치/각도를 인자로 받음.
/kill	turtlesim/srv/Kill	O	선택적으로 주목. 이름을 지정해 거북이 삭제, 이름 전달 필요
/clear, /reset	std_srvs/srv/Empty	X	단순 호출 (학습 초기에만 주목)
/teleop_turtle/*, /turtlesim/* (parameters 계열)	rcl_interfaces/srv/*	O	ROS 2 파라미터 시스템 관련: 노드 설정의 동적 변경을 위한 인터페이스

## 3. 서비스 목록 + 타입 함께 보기

```
ros2 service list -t
```

```
rokey@rokey-550XBE-350XBE: ~ 87x22
rokey@rokey-550XBE-350XBE:~$ ros2 service list -t
/clear [std_srvs/srv/Empty]
/kill [turtlesim/srv/Kill]
/reset [std_srvs/srv/Empty]
/spawn [turtlesim/srv/Spawn]
/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]
/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]
```

## 4. 특정 타입을 사용하는 서비스 찾기

```
ros2 service find <type_name>
```

```
rokey@rokey-550XBE-350XBE:~$ ros2 service find std_srvs/srv/Empty
/clear
/reset
```

## 5. 서비스의 인터페이스 구조 보기

```
ros2 interface show <type_name>
```

```
rokey@rokey-550XBE-350XBE:~$ ros2 interface show turtlesim/srv/Spawn
float32 x
float32 y
float32 theta
string name # Optional. A unique name will be created and returned if this is empty
---
string name
```

- 서비스 요청과 응답 필드가 ---로 구분되어 출력됨
- Spawn은 요청 시 x, y, theta, name을 주고, 응답으로 name을 받음

## 6. 서비스 호출하기

```
ros2 service call <service_name> <service_type> <arguments>
```

### a. 인자 없는 서비스 호출

```
ros2 service call /clear std_srvs/srv/Empty
```

```
rokey@rokey-550XBE-350XBE:~$ ros2 service call /clear std_srvs/srv/Empty
requester: making request: std_srvs.srv.Empty_Request()

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

### b. 인자 있는 서비스 호출

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2.0, y: 2.0, theta: 0.2, name: ''}"
```

```
rokey@rokey-550XBE-350XBE:~$ ros2 service call /spawn turtlesim/srv/Spawn "{x: 2.0, y: 2.0, theta: 0.2, name: ''}"
requester: making request: turtlesim.srv.Spawn_Request(x=2.0, y=2.0, theta=0.2, name='')

response:
turtlesim.srv.Spawn_Response(name='turtle2')
```

## 추가 실습

### 1. turtle1/set\_pen 서비스로 거북이 펜 색 바꾸기

```
ros2 interface show turtlesim/srv/SetPen
```

```
ros2 service call /turtle1/set_pen turtlesim/srv/SetPen -- "{r': 255, 'g': 0, 'b': 0, 'width': 5, 'off': 0}"
```

```
rokey@rokey-550XBE-350XBE:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen -- "{r': 255, 'g': 0, 'b': 0, 'width': 5, 'off': 0}"
waiting for service to become available...
requester: making request: turtlesim.srv.SetPen_Request(r=255, g=0, b=0, width=5, off=0)

response:
turtlesim.srv.SetPen_Response()
```



## 2. turtle1/teleport\_absolute로 위치 순간이동

```
ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.0, y: 5.0, theta: 1.57}"
```

```
rokey@rokey-550XBE-350XBE:~$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 5.0, y: 5.0, theta: 1.57}"
requester: making request: turtlesim.srv.TeleportAbsolute_Request(x=5.0, y=5.0, theta=1.57)

response:
turtlesim.srv.TeleportAbsolute_Response()
```



## 3. kill 서비스로 특정 거북이 제거

```
ros2 service call /kill turtlesim/srv/Kill "{name: 'turtle2'}"
```



```
rokey@rokey-550XBE-350XBE:~$ ros2 service call /kill turtlesim/srv/Kill "{name: 'turtle2'}"
waiting for service to become available...
requester: making request: turtlesim.srv.Kill_Request(name='turtle2')

response:
turtlesim.srv.Kill_Response()
```

## [Parameter]

- ROS 2에서 노드(Node)의 설정값 또는 구성 정보를 의미
- 노드의 동작을 동적으로 조정하거나 초기 설정값을 지정하는 데 사용
- 파라미터는 노드 실행 시 자동으로 로드되거나, 실행 중에 실시간으로 조회 및 수정 가능

### 특징

- 파라미터는 **노드 단위로 독립적으로 관리**  
: 노드 A의 파라미터는 노드 B와 공유되지 않으며, 각 노드는 자신의 파라미터 공간을 유지
- 파라미터는 다양한 데이터 타입을 지원:
  - integer (정수)
  - float (실수)
  - boolean (참/거짓)
  - string (문자열)
  - list (리스트, 배열 형태)

### 활용 목적

- 노드의 기본 동작을 설정하거나, 실행 중 조정할 수 있도록 지원
- 시스템 전체를 다시 빌드하지 않고도, 특정 노드의 동작 방식(예: 속도, 색상, 이름 등)을 유연하게 변경 가능
- 실제 로봇 적용 시, 센서 감도, 제어기 게인, 통신 주기 등의 값을 파라미터로 관리하여 유지보수와 확장성 UP

### 파라미터의 영속성(persistence)

- `ros2 param set`으로 설정한 파라미터는 **해당 세션에만 유효하며, 노드를 종료하면 초기화**
- `ros2 param dump`를 통해 현재 설정을 `.yaml` 파일로 저장할 수 있으며, `ros2 param load`를 통해 언제든지 복원 가능
- 노드 실행 시 `--ros-args --params-file <파일경로>` 옵션을 사용하면 시작부터 설정값을 적용 가능

## Tutorial - Understanding parameters

<https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Parameters/Understanding-ROS2-Parameters.html>

### 1. 현재 파라미터 목록 보기

`ros2 param list`

```
rokey@rokey-550XBE-350XBE: ~ 82x22
rokey@rokey-550XBE-350XBE:~$ ros2 param list
/teleop_turtle:
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  background_r
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  use_sim_time
```

### 2. 파라미터 값 확인

`ros2 param get <node_name> <parameter_name>`

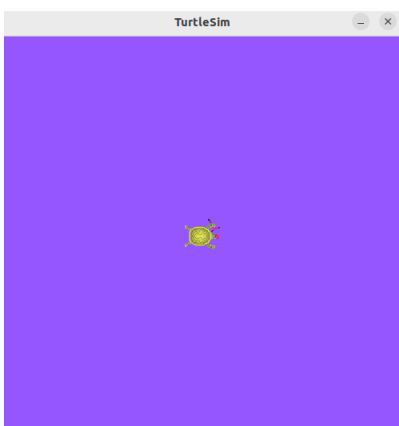
```
rokey@rokey-550XBE-350XBE: ~ 82x22
rokey@rokey-550XBE-350XBE:~$ ros2 param get /turtlesim background_g
Integer value is: 86
```

### 3. 파라미터 값 변경

`ros2 param set <node_name> <parameter_name> <value>`

- Set parameter successful 출력되면 성공

```
rokey@rokey-550XBE-350XBE:~$ ros2 param set /turtlesim background_r 150
Set parameter successful
```



#### 4. 현재 파라미터를 파일로 저장

```
ros2 param dump <node_name>
```

```
ros2 param dump /turtlesim > turtlesim.yaml
```

→ 현재 실행 중인 노드에만 적용, 노드 종료 시 초기화

```
rokey@rokey-550XBE-350XBE:~$ ros2 param dump /turtlesim > turtlesim.yaml
rokey@rokey-550XBE-350XBE:~$ cat turtlesim.yaml
/turtlesim:
  ros__parameters:
    background_b: 255
    background_g: 86
    background_r: 150
    qos_overrides:
      /parameter_events:
        publisher:
          depth: 1000
          durability: volatile
          history: keep_last
          reliability: reliable
    use_sim_time: false
```

#### 5. 저장한 설정값 다시 불러오기

```
ros2 param load <node_name> <parameter_file>
```

```
ros2 param load /turtlesim turtlesim.yaml
```

- 파라미터가 Set parameter successful 메시지와 함께 재적용됨
- 단, qos\_overrides 같은 read-only 파라미터는 런타임에 변경 불가하여 실패 메시지가 뜰 수 있음

```
rokey@rokey-550XBE-350XBE:~$ ros2 param load /turtlesim turtlesim.yaml
Set parameter background_b successful
Set parameter background_g successful
Set parameter background_r successful
Set parameter qos_overrides./parameter_events.publisher.depth failed: parameter
'qos_overrides./parameter_events.publisher.depth' cannot be set because it is
read-only
Set parameter qos_overrides./parameter_events.publisher.durability failed: para
meter 'qos_overrides./parameter_events.publisher.durability' cannot be set beca
use it is read-only
Set parameter qos_overrides./parameter_events.publisher.history failed: paramet
er 'qos_overrides./parameter_events.publisher.history' cannot be set because it
is read-only
Set parameter qos_overrides./parameter_events.publisher.reliability failed: par
ameter 'qos_overrides./parameter_events.publisher.reliability' cannot be set be
cause it is read-only
Set parameter use_sim_time successful
```

#### 6. 파라미터 파일을 노드 실행 시 적용

```
ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>
```

```
ros2 run turtlesim turtlesim_node --ros-args --params-file turtlesim.yaml
```

- 노드 시작과 동시에 설정 파일이 반영됨
- 이 방식은 read-only 파라미터까지 포함한 전체 설정을 적용할 수 있음