

# ROS2 학습 보고서

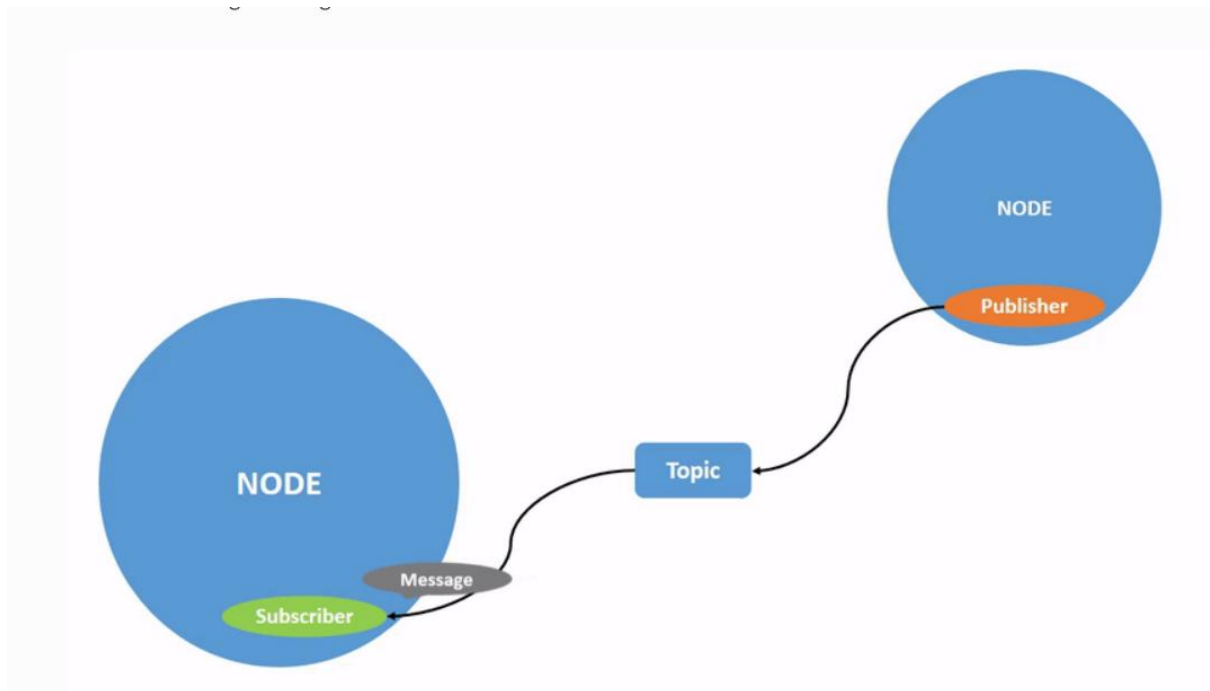
## 목차

1. Topics
2. Services
3. Parameters
4. Actions
5. Beginner: Client Libraries

## 1. Topics

### 개요

- ROS2 는 많은 모듈러 노드들의 복잡한 시스템을 타개
- 토픽은 ROS graph 에서 메시지를 교환하기 위한 노드들의 버스와 같은 역할을 하는 중요한 요소
- 노드는 어떤수의 토픽이든 데이터를 보낸다. 그리고 동시에 어떤 수의 토픽이든 subscriptions 을 가짐
- 토픽은 데이터가 노드 사이를 움직이고 시스템의 다른 부분사이를 움직이는 중요한 방법 중 하나



### Task

#### (1) Setup

- 터미널에 `ros2 run turtlesim turtlesim_node`를 입력함으로써 터틀심 세팅

#### (2) rqt\_graph

- 노드와 토픽의 변화를 시각화, 그들사이의 연결 시각화

- 
- The screenshot shows the `rqt_graph` window with the `RosGraph` tab selected. The interface includes a search bar at the top, a group selector set to `0`, and checkboxes for various node types and visibility options. The main area displays a directed graph with the following components and connections:
- Nodes:**
    - `/teleop_turtle`: Represented by a blue oval.
    - `/turtle1/cmd_vel`: Represented by a red rectangle.
    - `/turtlesim`: Represented by a green oval.
  - Topics (Rectangles):**
    - `/turtle1/rotate_absolute/_action/status`: On the left.
    - `/turtle1/rotate_absolute/_action/feedback`: On the right.
  - Connections:**
    - A curved arrow from `/teleop_turtle` to `/turtle1/cmd_vel`.
    - A straight arrow from `/turtle1/cmd_vel` to `/turtlesim`.
    - A curved arrow from `/turtlesim` to `/turtle1/rotate_absolute/_action/feedback`.
    - A curved arrow from `/turtle1/rotate_absolute/_action/feedback` back to `/teleop_turtle`.
    - A straight arrow from `/turtle1/rotate_absolute/_action/status` to `/teleop_turtle`.

- ros2 topic list: 이 명령어를 치면 토픽 목록을 볼 수 있다.
- ros2 topic list -t: 토픽의 유형을 나타내고 이에 따라 분류

- ros2 topic echo <topic\_name>: 토픽에서 발행된 데이터를 볼 때 사용
- ros2 topic echo /turtle1/cmd\_vel: echo 를 사용하여 해당 토픽 조사 가능

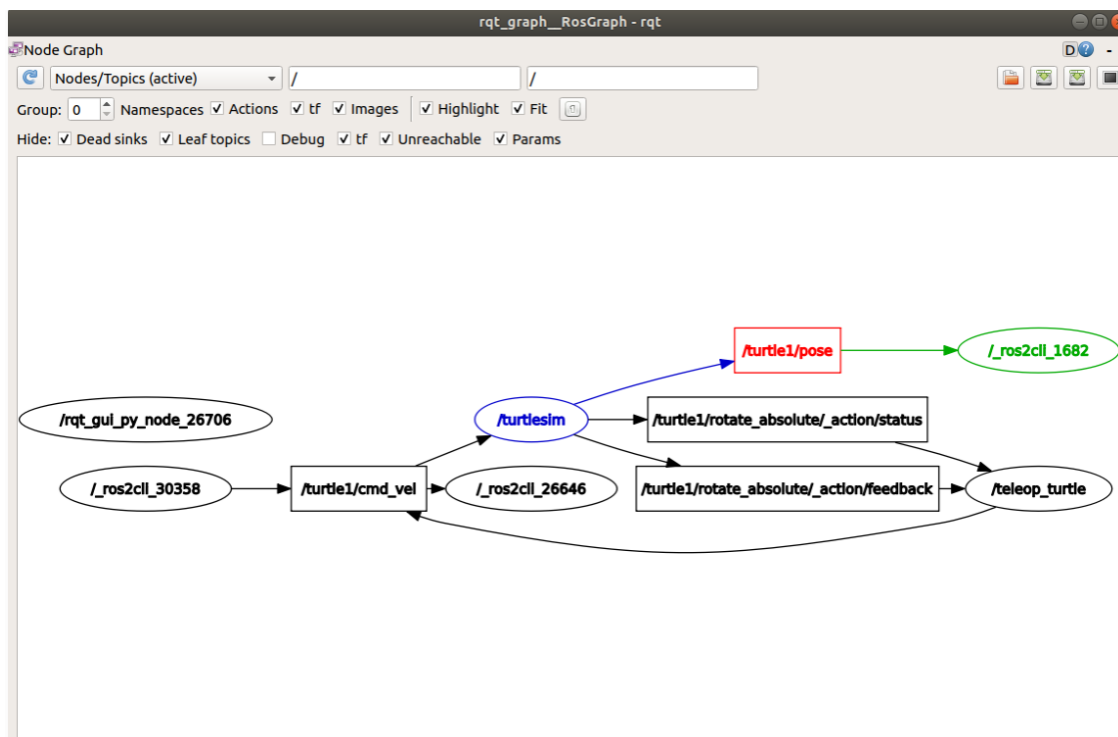
- `ros2 topic info /turtle1/cmd_vel`: 토픽 정보들을 볼 수 있음

## (6) ROS2 interface show

- Publisher 와 subscriber 은 반드시 같은 type 의 메시지를 주고받아야함
- 앞서 얘기한 `ros2 topic list -t` 을 통해 각 토픽에 어떤 메시지 유형이 사용되는지 알 수 있음
- `geometry_msgs/msg/Twist` 해당 문장은 패키지 `geometry_msgs` 안에 `Twist` 라는 메시지가 있음을 뜻함
- `ros2 interface show <msg_type>`: 데이터구조가 어떤 메시지를 받기를 기다리는지 볼 수 있음

## (7) ROS2 topic pub

- 명령어를 사용하여 토픽에 곧바로 데이터를 보낼 수 있다.
- 사용형태는 `ros2 topic pub <topic_name> <msg_type> '<args>'`
- `ros2 topic pub --once -w 2`: 한 번만 실행하고 싶을 때 `--once`, `-w n` 은 `n` 개의 매칭되는 subscription 을 기다림.
- `ros2 topic echo /turtle1/pose`: pose 토픽에 echo 실행 가능



- timestamp 를 포함한 메시지를 보낼 땐 `std_msgs/msg/Header` 이 형식 이용

ex) `ros2 topic pub /pose geometry_msgs/msg/PoseStamped`

- 메시지가 완전한 header 가 필요없다면 `builtin_interfaces/msg/Time` 이용

ex) `ros2 topic pub /reference sensor_msgs/msg/TimeReference`

#### (8) ROS2 topic hz

- `ros2 topic hz /turtle1/pose`: pose 라는 토픽에 데이터를 보내는 속도를 알 수 있음

- `ros2 topic pub --rate 1`: 속도 조절 가능

#### (9) ROS2 topic bw

- `ros2 topic bw /turtle1/pose`: 토픽이 쓰는 대역폭 알 수 있음

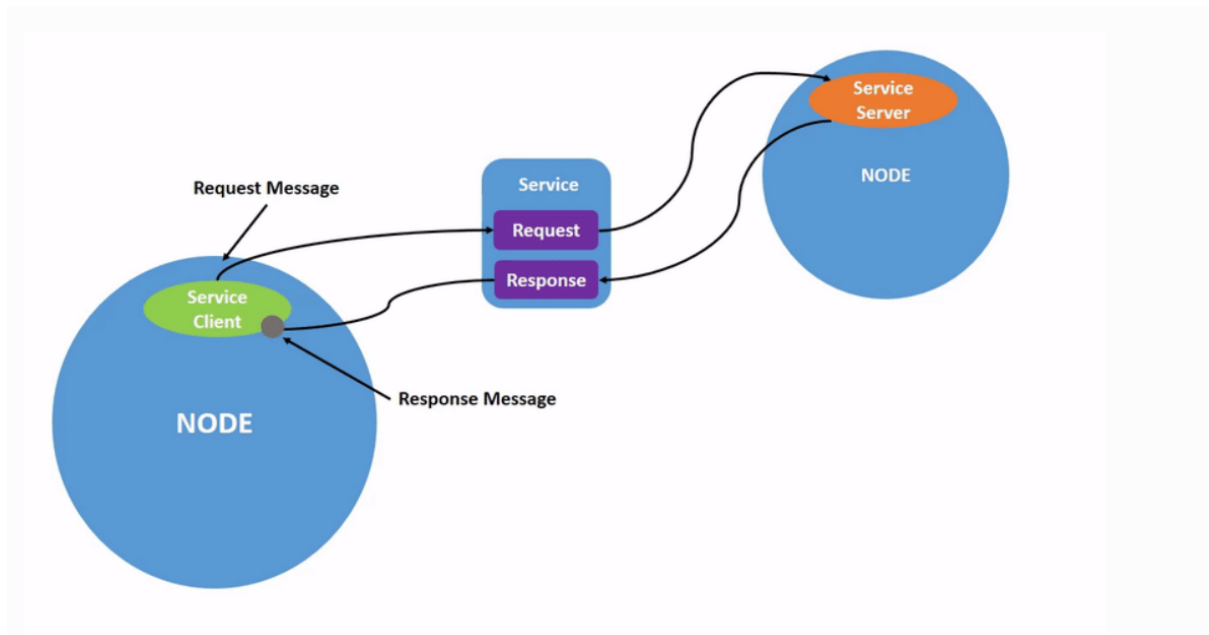
#### (10) ROS2 topic find

- `ros2 topic find <topic_type>`: 해당 type 을 쓰는 이용가능한 토픽들 출력

## 2. Services

### 개요

- service 는 ROS graph 에서 노드들이 소통하는 또 다른 방식
- call-and-response model 을 기반으로 함



### Tasks

#### (1) ros2 service list

- service 목록을 확인 할 수 있음
- `ros2 service list -t`: service 목록과 함께 type를 보여줌

#### (2) ROS2 service type

- `ros2 service type <service_name>`: 서비스의 type을 알 수 있음
- `ros2 service type /clear std_srvs/srv/Empty`: 리셋과 비슷한 역할

#### (3) ROS2 service find

- `ros2 service find <type_name>`: 특정 type의 service들을 찾을 수 있음
- `ros2 service find std_srvs/srv/Empty`: Empty type의 service들을 찾을 수 있음

#### (4) ROS2 interface show

- `ros2 interface show <type_name>`: 서비스를 호출할 수 있음
- `ros2 interface show turtlesim/srv/Spawn`: 요청과 응답이 이루어지는 것을 볼 때 Spawn 이용

#### (5) ROS2 service call

- `ros2 service call <service_name> <service_type> <arguments>`: 해당 인자들을 입력하여 서비스 호출
- `ros2 service call /clear std_srvs/srv/Empty`: 이런식으로 Empty type으로 호출하면 어떠한 인자를 받지 않은 상태로 호출





### 3. Parameters

#### 개요

- Parameter 는 Node 의 설정 값
- Node 는 integers, floats, booleans, strings, and lists 등의 parameter 를 저장 가능

#### (1) ROS2 param list

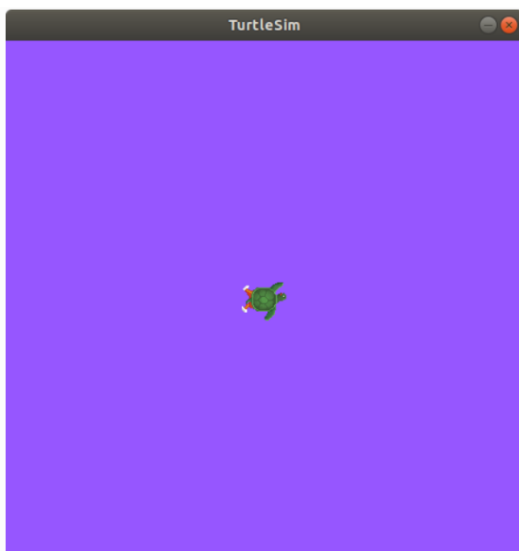
- ros2 param list: 내 노드의 속해있는 parameter들을 확인 가능

#### (2) ROS2 param get

- ros2 param get <node\_name> <parameter\_name>: parameter의 현재 값과 type을 확인 가능

#### (3) ROS2 param set

- ros2 param set <node\_name> <parameter\_name> <value>: 실행할 때 파라미터의 값을 변경할 때 쓰임
- ros2 param set /turtlesim background\_r 150: 터틀심 배경 바꿀 수 있음



#### (4) ROS2 param dump

- ros2 param dump <node\_name>: 모든 Node의 현재 parameter 값을 확인가능
- ros2 param dump /turtlesim > turtlesim.yaml: turtlesim 예시

/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) ROS2 param load

- ros2 param load <node\_name> <parameter\_file>: parameter를 파일로부터 현재 실행중인 Node에 load 가능

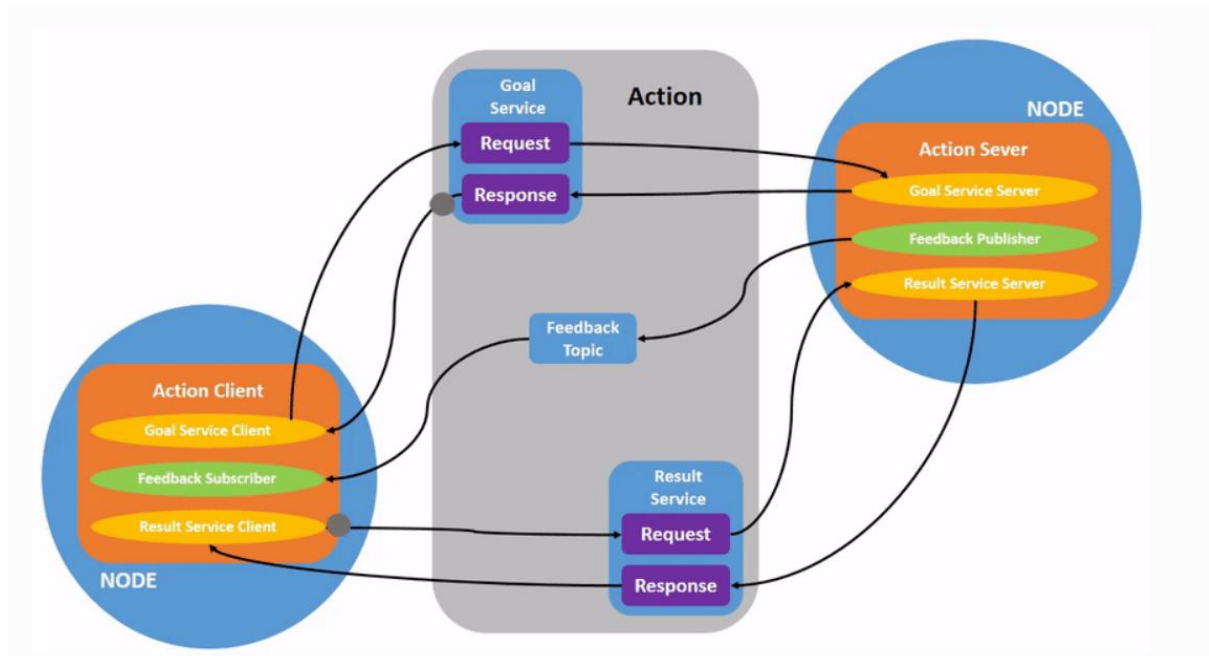
#### (6) Load parameter file on node startup

- ros2 run <package\_name> <executable\_name> --ros-args --params-file <file\_name>: 사용자의 저장된 parameter 값들을 사용한 같은 노드를 시작하기 위해 사용

## 4. Actions

### 개요

- Action 은 ROS2 의 통신 type 중 하나이며, 오래 지속되는 task 를 위해 만들어짐
- Goal, Feedback, Result 3 가지로 구성되어 있음
- Topic 과 Service 를 기반으로 하며, service 와 기능이 유사하지만 action 은 취소할 수 있다.
- 피드백을 꾸준히 제공해줌



### Tasks

#### (1) Use actions

- /teleop\_turtle를 실행하면 다음과 같은 메시지 출력
- Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
- 키를 누르면 [INFO] [turtlesim]: Rotation goal completed successfully 출력
- Action 서버는 새로운 입력이 들어왔을 때 이전 goal을 버림

## (2) ROS2 node info

- `ros2 node info /turtlesim`: turtlesim의 action 목록을 볼 수 있음

## (3) ROS2 action list

- `ros2 action list`: ROS graph의 모든 action을 확인하기 위해 사용
- `ros2 action list -t`: action과 type을 함께 출력

## (4) ROS2 action info

- `ros2 action info /turtle1/rotate_absolute`: 터틀의 action인 rotate\_absolute의 정보를 볼 수 있음

## (5) ROS2 interface show

- `ros2 interface show turtlesim/action/RotateAbsolute`: 다음과 같이 입력하면 Goal, Result, Feedback 순으로 출력됨

## (6) ROS2 action send\_goal

- `ros2 action send_goal <action_name> <action_type> <values>`: action goal을 직접 보내줄 수 있음
- `--feedback`를 사용해 해당 goal의 feedback을 확인할 수 있음

## 5. Beginner: Client Libraries

- pub <-> sub, service <-> client 복습 후 실행 완료

```
kmj@kmj: ~/Desktop/ros2_ws
[INFO] [1758009786.538972440] [minimal_publisher]: Publishing: 'Hello, world! 12'
[INFO] [1758009787.038632419] [minimal_publisher]: Publishing: 'Hello, world! 12'
[INFO] [1758009787.538502926] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009788.038363638] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009788.538298100] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009789.038151929] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009789.538013583] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009790.038132105] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009790.537971079] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009791.038067437] [minimal_publisher]: Publishing: 'Hello, world! 13'
[INFO] [1758009791.537527327] [minimal_publisher]: Publishing: 'Hello, world! 13'

kmj@kmj:~/Desktop/ros2_ws$ . install/setup.bash
kmj@kmj:~/Desktop/ros2_ws$ ros2 run cpp_pubsub listener
[INFO] [1758009783.040151569] [minimal_subscriber]: I heard: 'Hello, world! 121'
[INFO] [1758009783.539876286] [minimal_subscriber]: I heard: 'Hello, world! 122'
[INFO] [1758009784.039774357] [minimal_subscriber]: I heard: 'Hello, world! 123'
[INFO] [1758009784.540473021] [minimal_subscriber]: I heard: 'Hello, world! 124'
[INFO] [1758009785.039802283] [minimal_subscriber]: I heard: 'Hello, world! 125'
[INFO] [1758009785.539851251] [minimal_subscriber]: I heard: 'Hello, world! 126'
[INFO] [1758009786.039638959] [minimal_subscriber]: I heard: 'Hello, world! 127'
[INFO] [1758009786.539340315] [minimal_subscriber]: I heard: 'Hello, world! 128'
[INFO] [1758009787.038984168] [minimal_subscriber]: I heard: 'Hello, world! 129'
[INFO] [1758009787.538848185] [minimal_subscriber]: I heard: 'Hello, world! 130'
[INFO] [1758009788.038691600] [minimal_subscriber]: I heard: 'Hello, world! 131'
[INFO] [1758009788.538801496] [minimal_subscriber]: I heard: 'Hello, world! 132'
[INFO] [1758009789.038546509] [minimal_subscriber]: I heard: 'Hello, world! 133'
[INFO] [1758009789.538407221] [minimal_subscriber]: I heard: 'Hello, world! 134'
[INFO] [1758009790.038729455] [minimal_subscriber]: I heard: 'Hello, world! 135'
[INFO] [1758009790.538393154] [minimal_subscriber]: I heard: 'Hello, world! 136'
[INFO] [1758009791.038619755] [minimal_subscriber]: I heard: 'Hello, world! 137'
[INFO] [1758009791.537907686] [minimal_subscriber]: I heard: 'Hello, world! 138'

kmj@kmj:~/Desktop/ros2_ws/src$ cd cpp_srvcli/src/
kmj@kmj:~/Desktop/ros2_ws/src/cpp_srvcli/src$ cd ..
kmj@kmj:~/Desktop/ros2_ws/src/cpp_srvcli$ cd ..
kmj@kmj:~/Desktop/ros2_ws/src$ cd ..
kmj@kmj:~/Desktop/ros2_ws$ rosdep install -i --from-path src --rosdistro humble -y
ERROR: your rosdep installation has not been initialized yet. Please run:
    sudo rosdep init
    rosdep update
kmj@kmj:~/Desktop/ros2_ws$ colcon build --packages-select cpp_srvcli
Starting >>> cpp_srvcli
Finished <<< cpp_srvcli [3.75s]
Summary: 1 package finished [3.93s]
kmj@kmj:~/Desktop/ros2_ws$ source install/setup.bash
kmj@kmj:~/Desktop/ros2_ws$ ros2 run cpp_srvcli server
[INFO] [1758010277.206351583] [rclcpp]: Ready to add two ints.
[INFO] [1758010333.996805706] [rclcpp]: Incoming request
a: 2 b: 3
[INFO] [1758010333.996852231] [rclcpp]: sending back response: [5]

kmj@kmj:~/Desktop/ros2_ws$ ros2 run cpp_srvcli client 2 3
[INFO] [1758010333.99698157] [rclcpp]: Sum: 5
kmj@kmj:~/Desktop/ros2_ws$
```

### - Parameter

- this->declare\_parameter("my\_parameter", "world");

-> 파라미터를 선언. 이름: my\_parameter, 값: world

Void timer\_callback() 함수에서

- std::string my\_param = this->get\_parameter("my\_parameter").as\_string();

-> 파라미터를 불러와 my\_param에 저장

- RCLCPP\_INFO(this->get\_logger(), "Hello %s!", my\_param.c\_str());

-> 출력

- this->set\_parameters(all\_new\_parameters);

-> vector에 있는 파라미터 값들로 설정해줌

- 실행결과

```

kmj@kmj:~/Desktop/ros2_ws$ colcon build --packages-select cpp_parameters
Starting >>> cpp_parameters
Finished <<< cpp_parameters [3.81s]

Summary: 1 package finished [3.98s]
kmj@kmj:~/Desktop/ros2_ws$ source install/setup.bash
kmj@kmj:~/Desktop/ros2_ws$ ros2 run cpp_parameters minimal_param_node
[INFO] [1758012101.194327498] [minimal_param_node]: Hello world!
[INFO] [1758012102.194379859] [minimal_param_node]: Hello world!
[INFO] [1758012103.195100820] [minimal_param_node]: Hello world!
[INFO] [1758012104.194492732] [minimal_param_node]: Hello world!
[INFO] [1758012105.194542335] [minimal_param_node]: Hello world!
[INFO] [1758012106.194616261] [minimal_param_node]: Hello world!

```

- ros2 param set /minimal\_param\_node my\_parameter earth
- > 파라미터이름 값 이렇게 my\_parameter earth로 커스텀 파라미터를 보낼 수 있음
- 실행결과, Hello earth!가 한 번 출력된 것을 볼 수 있음

```

kmj@kmj:~/Desktop/ros2_ws$ ros2 param list
/minimal_param_node:
  my_parameter
  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
kmj@kmj:~/Desktop/ros2_ws$ ros2 param set /minimal_param_node my_parameter earth
Set parameter successful

```

Launch 파일에서

- package='cpp\_parameters'
- > 패키지 이름
- executable='minimal\_param\_node'
- > 실행 가능한 노드 파일 이름
- name='custom\_minimal\_param\_node'
- > 노드 이름
- output="screen", emulate\_tty=True,
- > 콘솔에 출력되도록 해줌
- parameters=[{'my\_parameter': 'earth'}]
- > 파라미터 값에 earth 넣어줌

- 실행결과

```
kmj@kmj: ~/Desktop/ros2_ws
kmj@kmj: ~/Desktop/ros2_ws 80x24
Starting >>> cpp_parameters
Finished <<< cpp_parameters [0.43s]

Summary: 1 package finished [0.61s]
kmj@kmj:~/Desktop/ros2_ws$ source install/setup.bash
kmj@kmj:~/Desktop/ros2_ws$ ros2 launch cpp_parameters cpp_parameters_launch.py
[INFO] [launch]: All log files can be found below /home/kmj/.ros/log/2025-09-16-17-46-10-465927-kmj-11548
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [minimal_param_node-1]: process started with pid [11549]
[minimal_param_node-1] [INFO] [1758012371.527200407] [custom_minimal_param_node]
: Hello earth!
[minimal_param_node-1] [INFO] [1758012372.527390217] [custom_minimal_param_node]
: Hello world!
[minimal_param_node-1] [INFO] [1758012373.527251538] [custom_minimal_param_node]
: Hello world!
```