2️⃣

Week 2

1. ROS1과 ROS2의 차이점

Features	ROS1	ROS2
Platforms	Linux, MacOS	Linux, MacOS, Windows
Node manager	ROS Master	use DDS's dynamic discovery
Life cycle	-	non-isolated build, no devel space
Command	rosrun, roslaunch, ros topic …	ros2 run, ros2 launch, ros2 topic
roslaunch	XML	Python, XML, YAML
Build system	rosbuild → catkin (CMAKE)	Ament (CMake), Python setuptools
Build tools	catkin_make, catkin_tools	Colcon

1.1. roslaunch

- ROS1 에서는 xml 형식을 이용해 launch 파일을 구성했다.
- ROS2 에서는 python코드를 사용해서 launch 파일을 구성할 수 있다.

Python, XML, or YAML: Which should I use?

Note

Launch files in ROS 1 were written in XML, so XML may be the most familiar to people coming from ROS 1. To see what's changed, you can visit Migrating launch files from ROS 1 to ROS 2.

For most applications the choice of which ROS 2 launch format comes down to developer preference. However, if your launch file requires flexibility that you cannot achieve with XML or YAML, you can use Python to write your launch file. Using Python for ROS 2 launch is more flexible because of following two reasons:

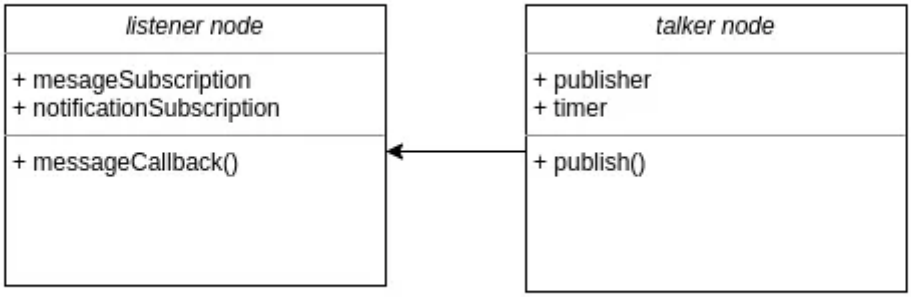
- Python is a scripting language, and thus you can leverage the language and its libraries in your launch files.
- ros2/launch (general launch features) and ros2/launch_ros (ROS 2 specific launch features) are written in Python and thus you have lower level access to launch features that may not be exposed by XML and YAML.

That being said, a launch file written in Python may be more complex and verbose than one in XML or YAML.

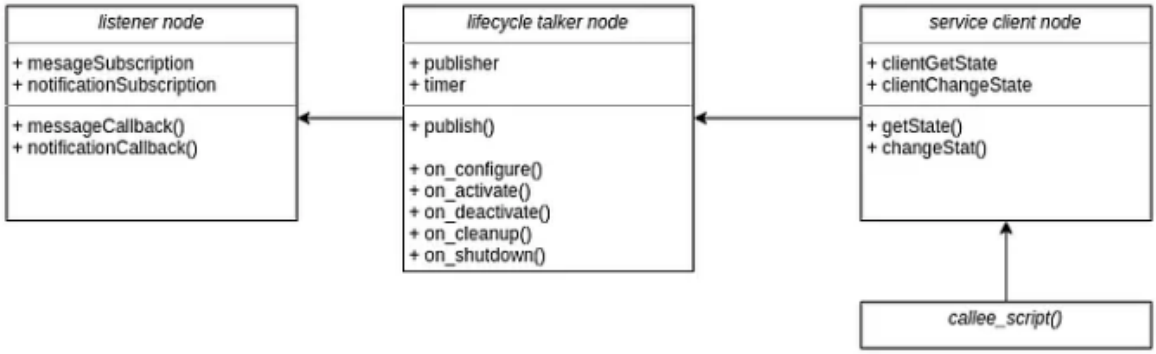
- ROS1에 익숙한 사람들은 xml형식을 이용해서 launch 파일을 작성하는것을 선호한다.
- Python, xml, yaml 중에서 어떤것을 써도 무관하지만, Python을 이용하면 다양한 라이브러리를 활용할 수 있다는 장점이 있다.

1.2. Life cycle

- ROS2 에서는 패키지의 각 노드들의 현재 상태를 모니터링하고 상태럴 제어할 수 있는 Lifecycle 기능을 라이브러리에 포함시켰다.
- 기존 ROS1에서는 할 수 없었던 노드의 상태를 모니터링할 수 있고, 노드의 상태에 따라 재시작하거나 교체할 수도 있음.
- 3부 7장에 자세한 내용이 나오지만 대략적으로 설명하면 아래와 같다.



- 기본적인 talker-listener 예제를 보면, 노드를 실행하는 즉시 talker 노드는 message를 publish하고 listener 노드는 message를 subscribe 한다.
- But, 노드가 한번 시작하면 그 이후에는 제어할 수 없음.



- ROS2에서는 런타임 도중에 노드를 제어할 수 있도록 하는 관리 노드가 도입됨
- 예를 들어, talker 노드가 실행 중일 때 on_configure() 메서드를 불러들일 수 있음.

1.3. DDS

- ROS1 에서는 ROS Master를 통해 각 노드들의 정보를 확인하였다.

- 따라서 ROS1 에서는 노드 사이의 연결을 위해 네임 서비스를 마스터에서 실행했었어야 했고, 이 ROS Master가 연결이 끊기거나 죽는 경우 모든 시스템이 마비되는 단점이 있었다.
- 하지만 DDS의 동적 검색 기능을 사용함에 따라 노드를 DDS의 Participant로 취급하게 되었으며, 동적 검색 기능을 이용하여 이를 연결할 수 있게 되었다.

- 예를 들어, 노드 A가 노드B에게 메시지를 보내려고 하는데, 노드 B의 연결상태가 고르지 못한다고 가정해보자.
- 이 경우, DDS는 자동으로 메시지를 저장하고, 노드 B가 다시 연결되면 메시지를 전송한다.
- 이처럼 DDS는 네트워크 상태에 관계없이 노드 간에 안정적으로 메시지를 전달할 수 있게 해준다.

- DDS는 QOS(Quality of Service) 라는 기능을 제공한다. 이는 ‘메시지가 얼마나 빨리 전달되어야 하는가?’ “메시지가 얼마나 오래 유지되어야 하는가?’ 와 같은 설정을 할 수 있게 해준다.

1.4. Build system

- ROS2에서는 빌드 시스템으로 ‘colcon’을 사용한다. ROS1에서는 ‘catkin_make’를 사용했다.
- ‘빌드’라는것은 내 소스코드를 실행 하능한 프로그램으로 변환하는 과정을 의미한다.
- ‘CMake’와 ‘Colcon’은 모두 이런 빌드 과정을 쉽게 하기 위한 도구이다. 이들은 ROS 패키지 내부의 소스코드를 컴파일하고, 필요한 라이브러리를 찾아주며, 패키지 간의 의존성을 파악하고 관리해준다.

- ‘Colcon’은 ‘CMake’보다 다양한 종류의 패키지와 언어를 지원한다. 예를 들어, CMake 기반의 패키지뿐만 아니라 Python 기반의 패키지도 지원한다.
- ‘Colcon’은 여러 패키지를 병렬로 빌드할 수 있어, 빌드 시간을 단축시킬 수 있다.