

# LIMO ROS2 Manual

---

WeGo & LIMO

# 목 차

1. Use Docker for ROS2 Development
2. How to Use

**01**

---

# **Use Docker for ROS2 Development**

# 01 Use Docker for ROS2 Development

- 기존의 Jetson Nano는 공식적으로 Ubuntu 20.04를 지원하지 않고 있다. (2022 / 05)
- ROS2를 잘 지원하는 OS는 Ubuntu 20.04이며, ROS2 Version은 Foxy 이상을 권장한다.
- 따라서, 이에 대한 개발을 위해 Ubuntu 18.04에서 Docker를 통해 Ubuntu 20.04를 가상화한 후, 이를 통해 ROS2의 개발을 진행해야 한다.
- 또한 Docker Cloud를 통해 개발 환경을 쉽게 공유할 수 있다.
- 이 챕터에서, Docker를 활용한 LIMO의 ROS2 개발 방법에 대해 소개하고, 장점에 대해 설명합니다.

# 01 Use Docker for ROS2 Development

- 추가적으로, 여러 팀 간의 개발 중에는 다음과 같은 문제가 필수적으로 발생합니다.
  - 빠르게 공통적인 개발 환경을 배포해야 합니다. (다른 아키텍처 타입의 기계에서 구동 되는)
  - 다른 사람의 PC에서는 정상적으로 컴파일 되지만, 내 PC에서는 오류가 납니다.
  - 지원하는 라이브러리의 차이 또는 지원하는 라이브러리 사이의 충돌
  - 특히 컴퓨터에서 여러 개의 작업을 수행해야 하는 경우 (예를 들어, ROS1, ROS2, QT, Deep Learning 등)
- 이와 같은 문제 상황에서 우리는 Docker를 사용해야 합니다.

# 01 Use Docker for ROS2 Development

- 이 후의 챕터에서 다음의 내용을 설명합니다.
  - VS Code + Docker
    - 어떤 조합을 사용하셔도 상관없지만, 위의 내용을 권장드립니다.
  - 실제 LIMO에서의 사용 방법
- 이와 같은 내용을 배우는데 오랜 시간이 걸리겠지만, 이를 배우고 나면, 다양한 곳에 활용할 수 있습니다.
- VS Code와 Docker 사용에 익숙하지 않으시면, 다음 내용을 참고하시면 좋습니다.
- ([https://anthonymsun256.github.io/docker-with-vsc\\_best-practice/](https://anthonymsun256.github.io/docker-with-vsc_best-practice/))

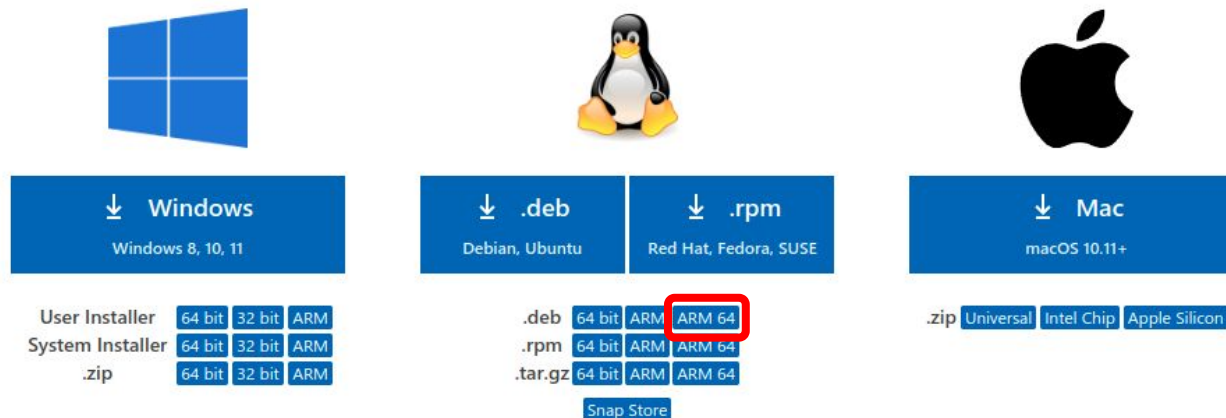
**02**

---

## **How To Use**

## 02 How to Use

- Docker 및 VS Code 설치
  - 우선 LIM0의 Jetson Nano에서 Nvidia-Docker2를 설치합니다.
  - `$ sudo apt install nvidia-docker2`
  - 그리고 아래 링크에서 VS Code ARM64 버전을 다운로드 받습니다.
  - <https://code.visualstudio.com/#alt-downloads>

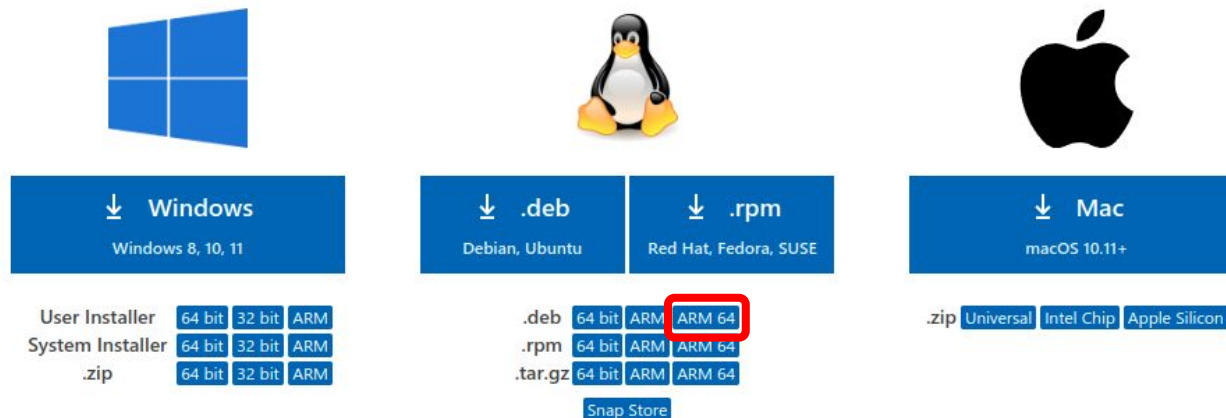




## 02 How to Use

- Docker 및 VS Code 설치

- 아래 명령어를 입력하여, VS Code가 다운로드된 폴더로 이동하여, 설치합니다.
- `$ cd ~/Downloads/ && sudo dpkg -i code_1..._arm64.deb`
- code\_1... 부분은 다운로드 받아진 버전을 적어주시면 됩니다. (code까지 적고 <Tab> 입력)



## 02 How to Use

- Docker 사용을 위한 그룹 및 사용자 등록
  - 최초로 Docker를 사용하면, 관리자 권한이 없으면 사용이 불가능합니다.
  - 이를 그룹 및 사용자 등록을 통해 일반 사용자도 사용가능하게 수정합니다.
  - `$ sudo groupadd docker`
  - `$ sudo usermod -aG docker $USER`
  - `$ newgrp docker`
  - 위 명령어를 순차적으로 입력하고, 재부팅합니다.
  - `$ sudo reboot`

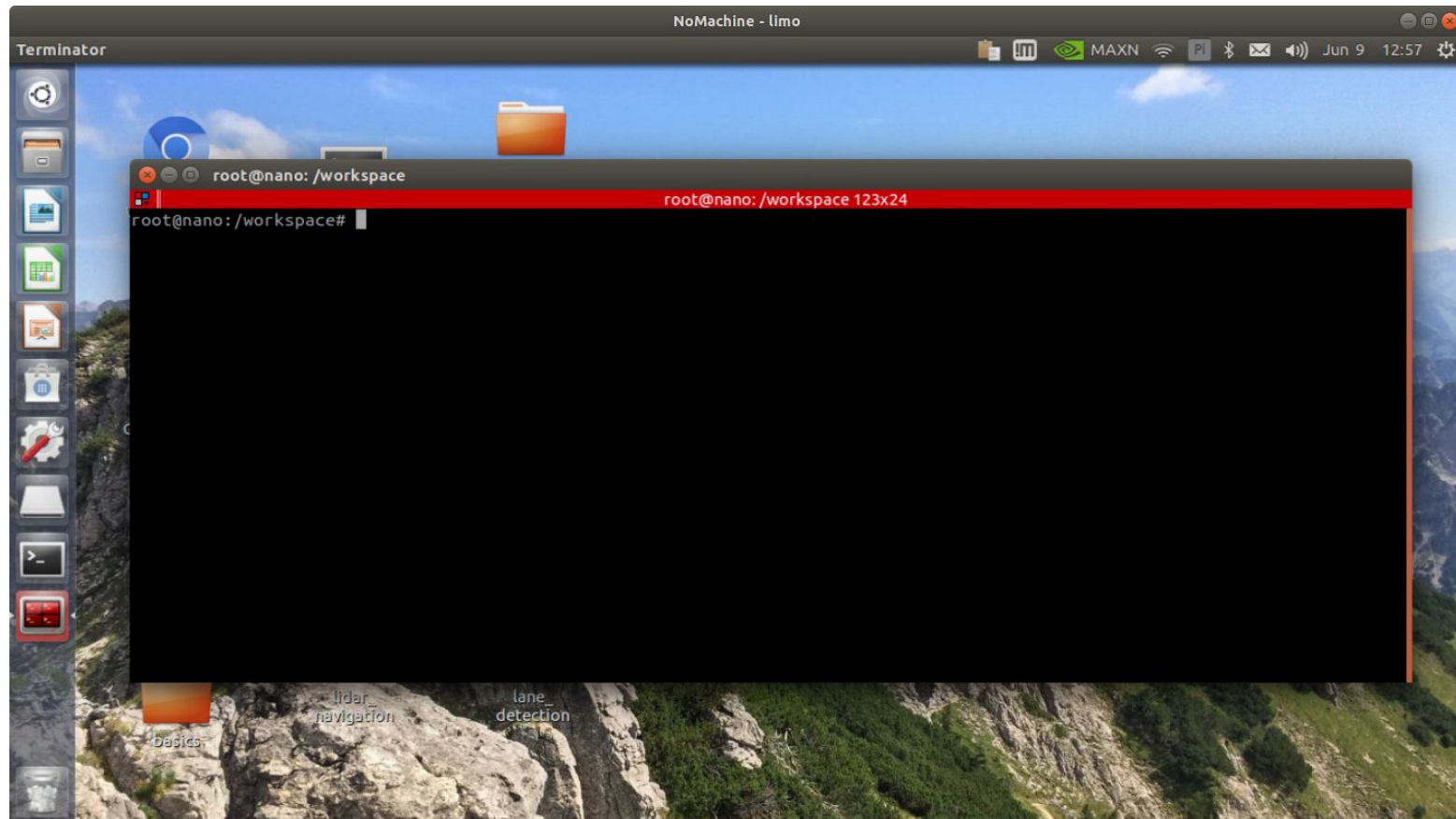
## 02 How to Use

- Docker 작업을 위한 환경 구축

- Docker 작업을 할 폴더를 만듭니다.
- `$ mkdir limo_foxy_dev && cd limo_foxy_dev`
- ROS2 Foxy를 실행할 수 있는 Image를 불러와서 실행합니다.
- `$ docker run --network=host --cap-add=SYS_PTRACE`  
`--cap-add=SYS_RAWIO --security-opt=seccomp:unconfined`  
`--security-opt=apparmor:unconfined --volume=/tmp/.X11-unix:/tmp/.X11-unix`  
`--runtime=nvidia --device /dev/ttyTHS1:/dev/ttyTHS1 --device`  
`/dev/ttyUSB0:/dev/ydlidar --device /dev/bus/usb/:/dev/bus/usb -v`  
`$(pwd)/:/workspace --name limo-foxy-dev -w /workspace -id ros:foxy`
- 최종적으로 사용할 수 있게 terminal 환경을 켭니다
- `$ docker exec -it limo-foxy-dev /bin/bash`

## 02 How to Use

- Docker 작업을 위한 환경 구축
  - 다음과 같이 root 권한의 상태에서 /workspace 폴더에 있으면 정상 실행된 것이다.



## 02 How to Use

- Docker 작업을 위한 환경 구축
  - 다음과 같이 root 권한의 상태에서 /workspace 폴더에 있으면 정상 실행된 것이다.
  - 이제 해당 터미널에서 아래 내용을 입력하여, 패키지 설치 관련 서버를 변경해주자.
  - `$ sed -i "s/ports.ubuntu.com/mirrors.ustc.edu.cn/g" /etc/apt/sources.list`
  - `$ sed -i "s/packages.ros.org/repo.huaweicloud.com/g"`  
`/etc/apt/sources.list.d/ros2-latest.list`
  - `$ apt update`
  - 그리고 다음의 문구를 입력하여, bashrc에 ros 관련 내용을 등록하자.
  - `$ echo "source /opt/ros/foxy/setup.bash" >> /etc/bash.bashrc`

## 02 How to Use

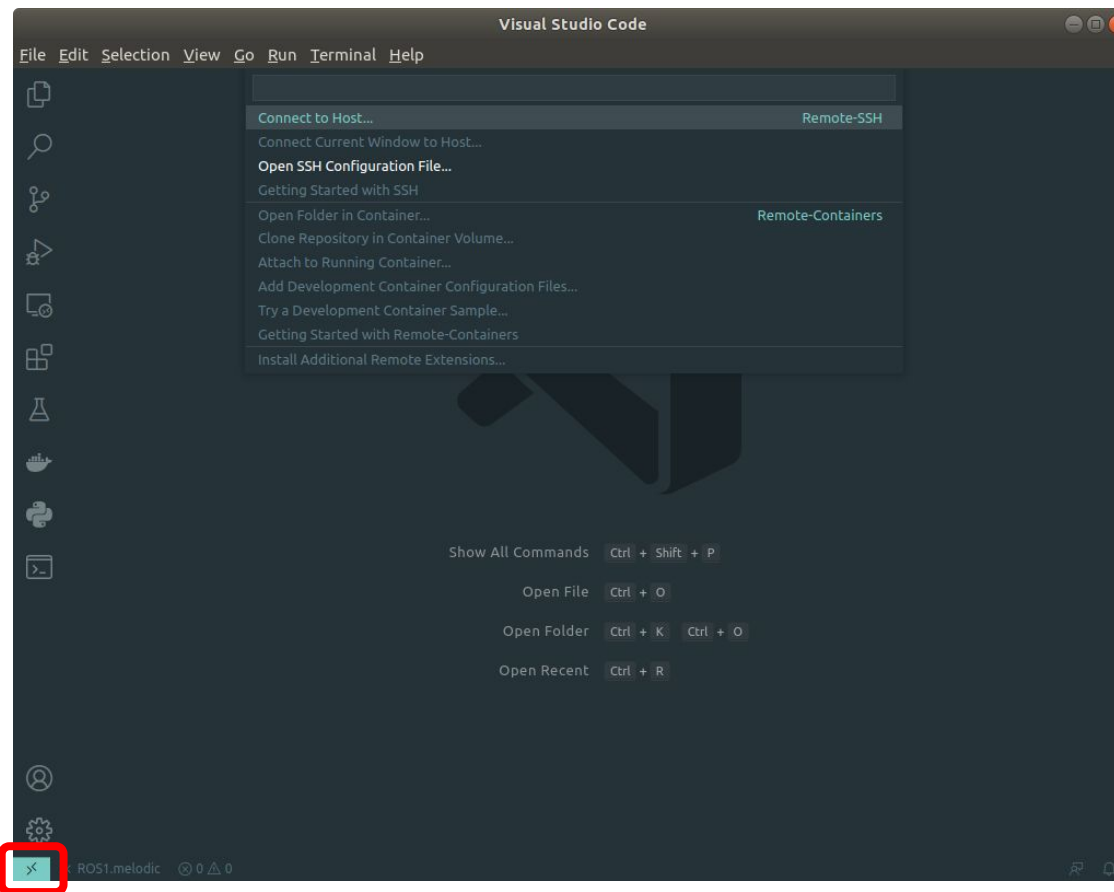
- Docker 작업을 위한 환경 구축
  - 다음으로 필요한 패키지를 설치하겠습니다.
  - `$ apt install -y python3-pip`
  - `$ pip install rosdep`
  - `$ rosdep init && rosdep update`
  - 그리고 원격 설정을 위한 패키지를 설치하겠습니다.
  - `$ apt install openssh-server systemctl udev swig`
  - `$ service ssh start`
  - `$ echo -e "Port 10022\nPermitRootLogin yes\nPermitEmptyPasswords yes" >> /etc/ssh/sshd_config.d/dev.conf`
  - `$ service ssh restart`

## 02 How to Use

- Docker 작업을 위한 환경 구축
  - 다음 명령어를 입력하여, 비밀번호를 지정합니다. (입력해도, 비밀번호는 표시되지 않습니다.)
  - `$ passwd`

## 02 How to Use

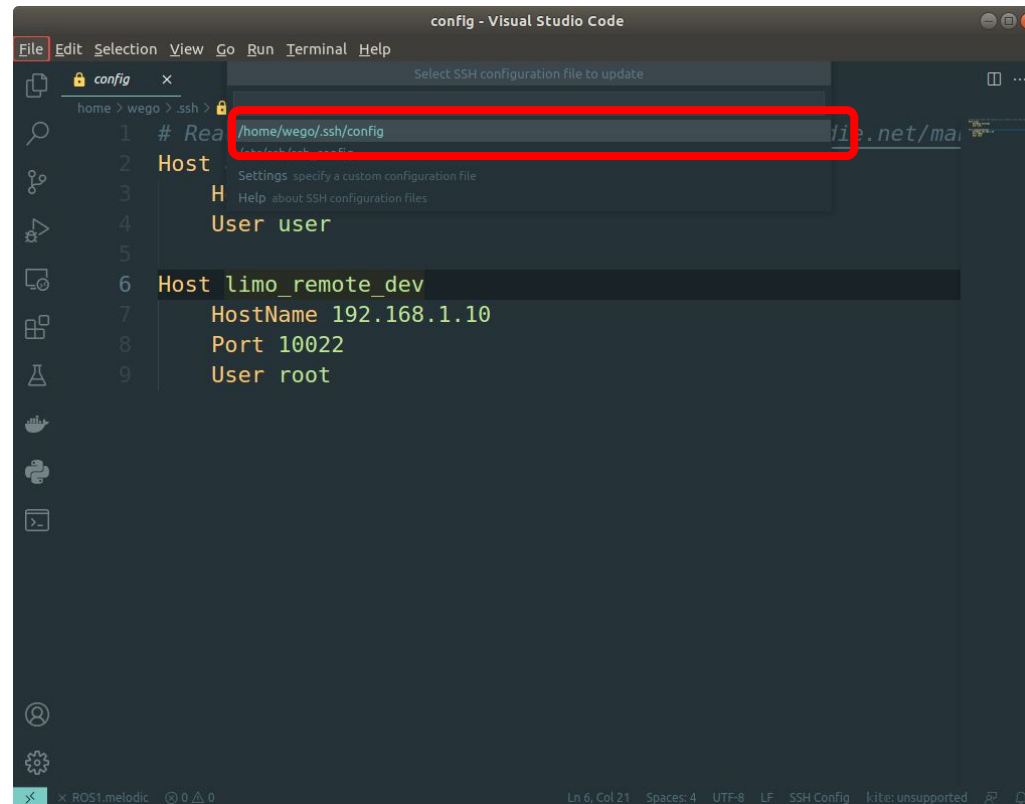
- Docker 접속을 통한 개발
  - 원격 접속을 해서 테스트할 PC에서 VS Code를 켭니다.
  - 좌측 하단을 누르고 위에 나오는 창에서 "Open SSH Configuration File ..."를 선택합니다.





## 02 How to Use

- Docker 접속을 통한 개발
  - 원격 접속을 해서 테스트할 PC에서 VS Code를 켭니다.
  - 좌측 하단을 누르고 위에 나오는 창에서 "Open SSH Configuration File ..."을 선택합니다.
  - 제일 위에 있는 것을 선택합니다.



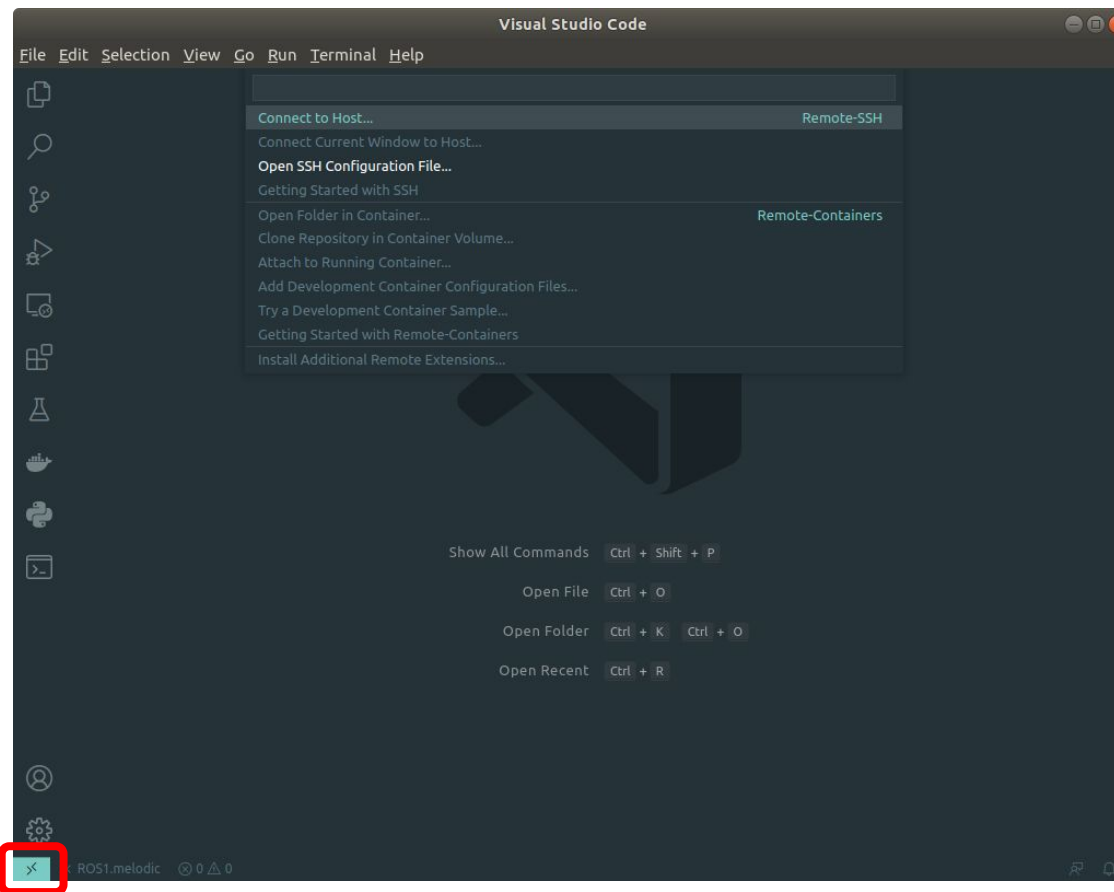
## 02 How to Use

- Docker 접속을 통한 개발
  - 열린 창에 다음과 같은 내용을 입력합니다.
  - HostName 부분은 LIMO의 IP주소를 입력하면 됩니다.

```
Host limo_remote_dev
  HostName 192.168.1.10
  Port 10022
  User root
```

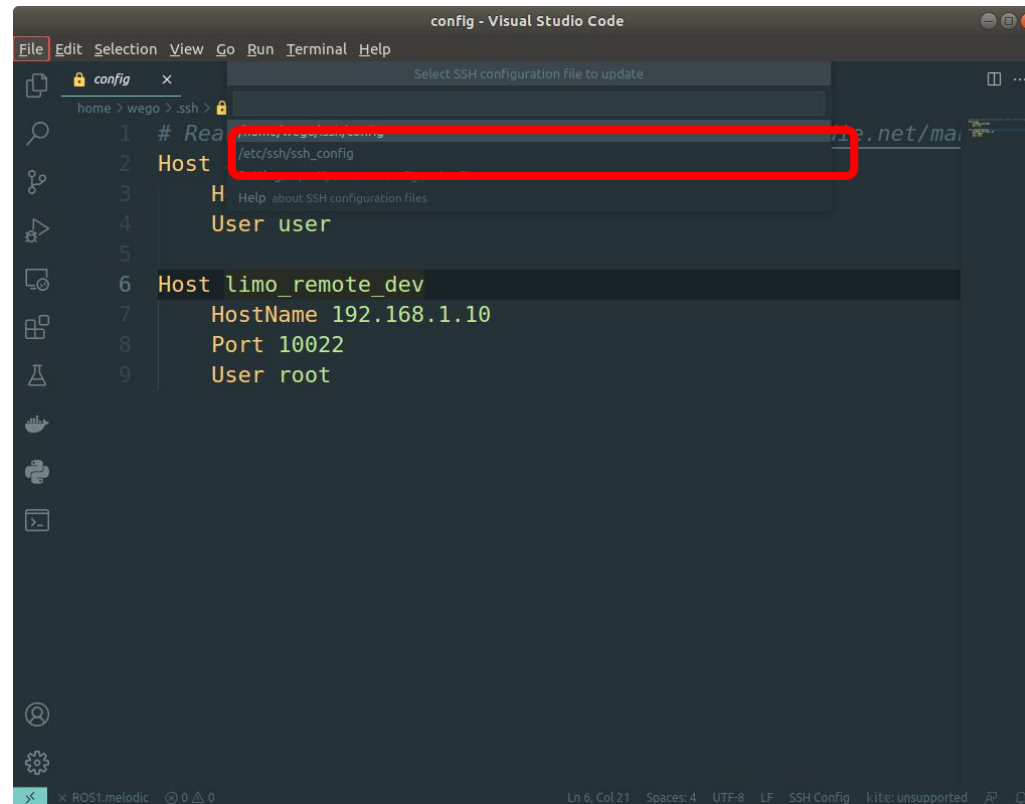
## 02 How to Use

- Docker 접속을 통한 개발
  - 원격 접속을 해서 테스트할 PC에서 VS Code를 켭니다.
  - 좌측 하단을 누르고 위에 나오는 창에서 "Open SSH Configuration File ..."를 선택합니다.



## 02 How to Use

- Docker 접속을 통한 개발
  - 원격 접속을 해서 테스트할 PC에서 VS Code를 켭니다.
  - 좌측 하단을 누르고 위에 나오는 창에서 "Open SSH Configuration File ..."을 선택합니다.
  - 위에서 두번 째에 있는 것을 선택합니다.



## 02 How to Use

- Docker 접속을 통한 개발
  - ForwardX11 no 라고 되어있는 부분을 ForwardX11 yes로 바꾸고, 주석을 풀어줍니다.
  - 이 후 재부팅합니다. (원격 접속하려는 PC를 재부팅)

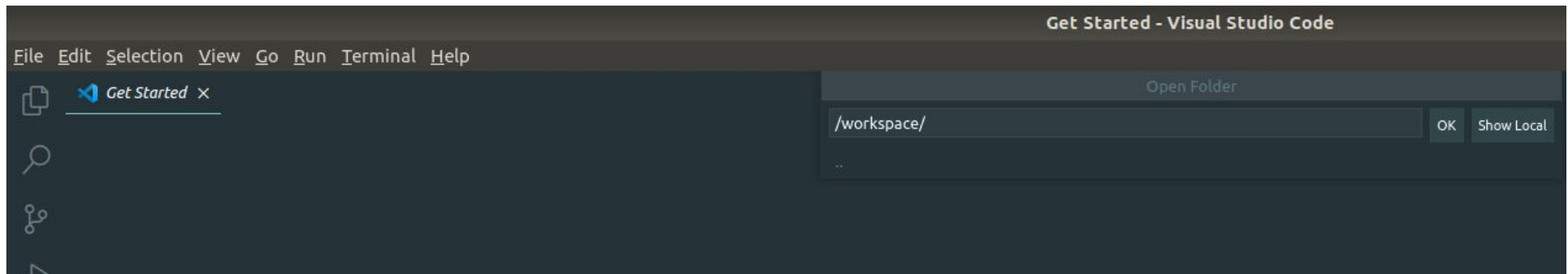
```
# Site-wide defaults for some commonly used options.  For a comprehensive
# list of available options, their meanings and defaults, please see the
# ssh_config(5) man page.
```

Host \*

```
# ForwardAgent no
ForwardX11 yes
# ForwardX11Trusted yes
# PasswordAuthentication yes
```

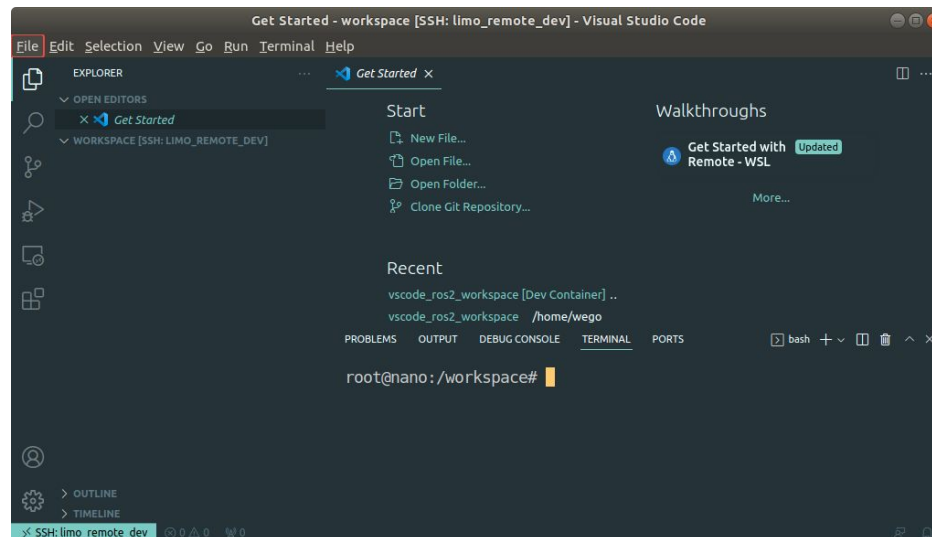
## 02 How to Use

- Docker 접속을 통한 개발
  - 다시 좌측 하단의 버튼을 클릭한 후, 이번에는 "Connect to Host ..."을 선택합니다.
  - 출력되는 목록 중 limo\_remote\_dev를 선택합니다.
  - 연결이 되면, Yes를 눌러 연결하고, 이전에 설정한 비밀번호를 입력합니다.
  - 좌측 상단의 File - Open Folder를 선택하시고, /workspace/ 폴더로 선택해줍니다.
  - 마찬가지로 비밀번호를 입력해줍니다.



## 02 How to Use

- Docker 접속을 통한 개발
  - 다시 좌측 하단의 버튼을 클릭한 후, 이번에는 "Connect to Host ..."을 선택합니다.
  - 출력되는 목록 중 limo\_remote\_dev를 선택합니다.
  - 연결이 되면, Yes를 눌러 연결하고, 이전에 설정한 비밀번호를 입력합니다.
  - 좌측 상단의 File - Open Folder를 선택하시고, /workspace/ 폴더로 선택해줍니다.
  - 마찬가지로 비밀번호를 입력해줍니다.
  - Ctrl + ` 를 입력하였을 때, docker에 연결된 터미널이 표시되면 정상적으로 접속된 것입니다.



## 02 How to Use

- 필요 패키지 다운로드 받기

- Ctrl + ` 를 입력해서 나오는 터미널에서 다음 내용을 입력합니다.
- \$ git clone <https://ghproxy.com/https://github.com/YDLIDAR/YDLidar-SDK.git>
- \$ mkdir -p YDLidar-SDK/build
- \$ cd YDLidar-SDK/build
- \$ cmake ..
- \$ make
- \$ make install
- \$ cd ..
- \$ pip install .
- \$ cd .. && rm -rf YDLidar-SDK
- \$ git clone --recursive [https://ghproxy.com/https://github.com/agilexrobotics/limo\\_ros2.git](https://ghproxy.com/https://github.com/agilexrobotics/limo_ros2.git) src



## 02 How to Use

- 필요 패키지 다운로드 받기
  - `/workspace/src/ydlidar_ros2/src/ydlidar_ros2_driver_node.cpp` 파일을 열어서 수정합니다.
  - 61번줄 `int optval = 115200;`
  - 76번줄 `optval = 3;`
  - 108번줄 `b_optvalue = true;`
  - 134번줄 `f_optvalue = 12.f;`
  - 143번줄 `f_optvalue = 8.f;`

## 02 How to Use

- 사용을 위해 컴파일 하기
  - `$ rosdep install --from-paths src --ignore-src -r -y`
  - `$ colcon build --symlink-install`
  - 아래와 같은 경고는 정상입니다.

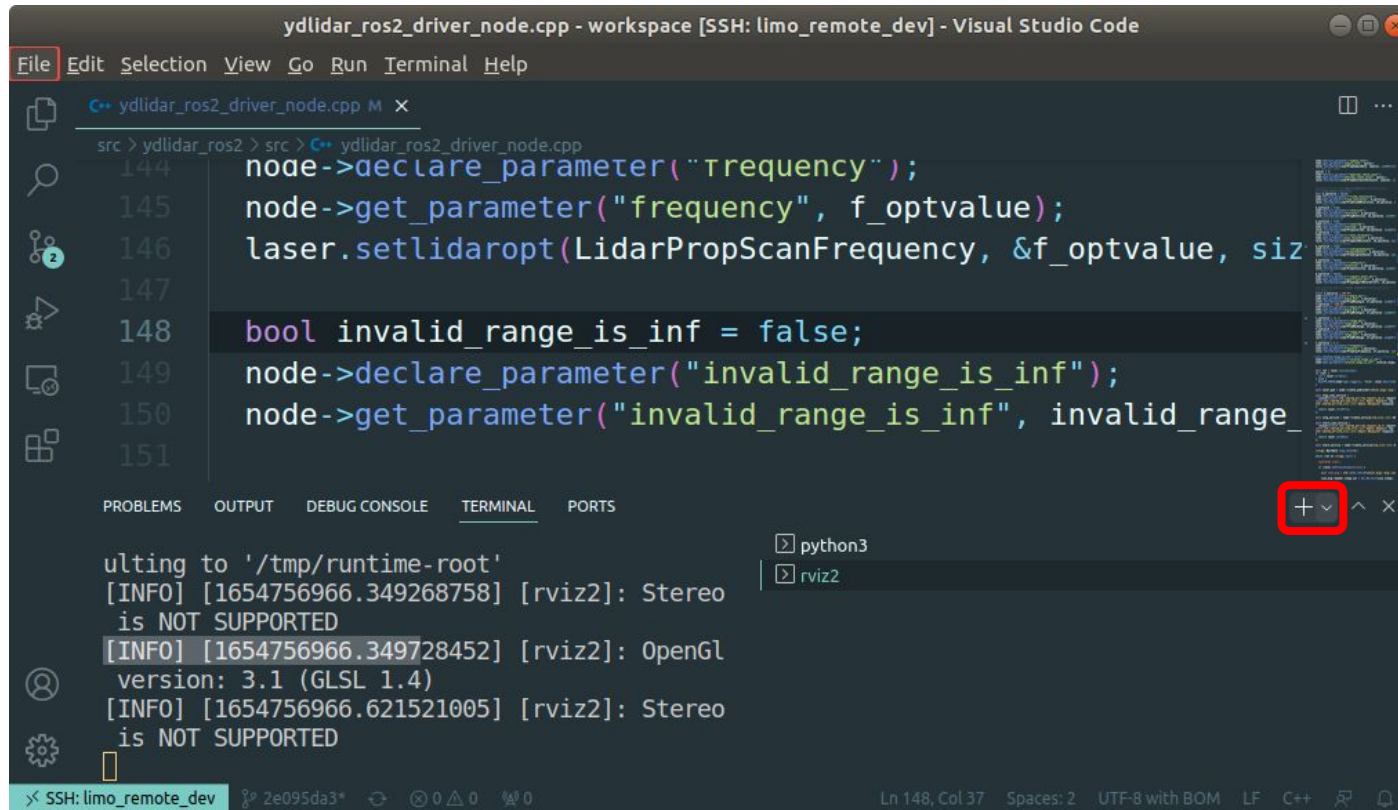
```
/workspace/src/limo_base/src/limo_driver.cpp: In member function 'void AgileX::LimoDriver::publishIMUData(double)':  
/workspace/src/limo_base/src/limo_driver.cpp:492:16: warning: unused variable 'present_theta_' [-Wunused-variable]  
492 |         double present_theta_ = imu_data_.yaw;  
    |                  ^  
/workspace/src/limo_base/src/limo_driver.cpp:493:16: warning: unused variable 'last_theta_' [-Wunused-variable]  
493 |         double last_theta_ = imu_data_.yaw;  
    |                  ^  
---  
Finished <<< limo_base [43.4s]  
  
Summary: 5 packages finished [1min 12s]  
2 packages had stderr output: limo_base ydlidar_ros2_driver  
root@agilex-desktop:/workspace#
```

## 02 How to Use

- 사용을 위해 컴파일 하기
  - 컴파일이 완료되었으면, 사용을 위해 등록해줍니다.
  - `$ source install/setup.bash`
  - 아래 명령어를 입력하여, 실행을 합니다.
  - `$ ros2 launch limo_bringup limo_start.launch.py`

## 02 How to Use

- 사용을 위해 컴파일 하기
  - 아래 버튼을 클릭하여, 새로운 터미널을 열고 다음 명령어를 입력합니다.
  - `$ rviz2`



The screenshot shows the Visual Studio Code interface. The editor window displays a C++ file named `ydlidar_ros2_driver_node.cpp` with the following code:

```
node->declare_parameter("Tfrequency");
node->get_parameter("frequency", f_optvalue);
laser.setlidaropt(LidarPropScanFrequency, &f_optvalue, siz

bool invalid_range_is_inf = false;
node->declare_parameter("invalid_range_is_inf");
node->get_parameter("invalid_range_is_inf", invalid_range_
```

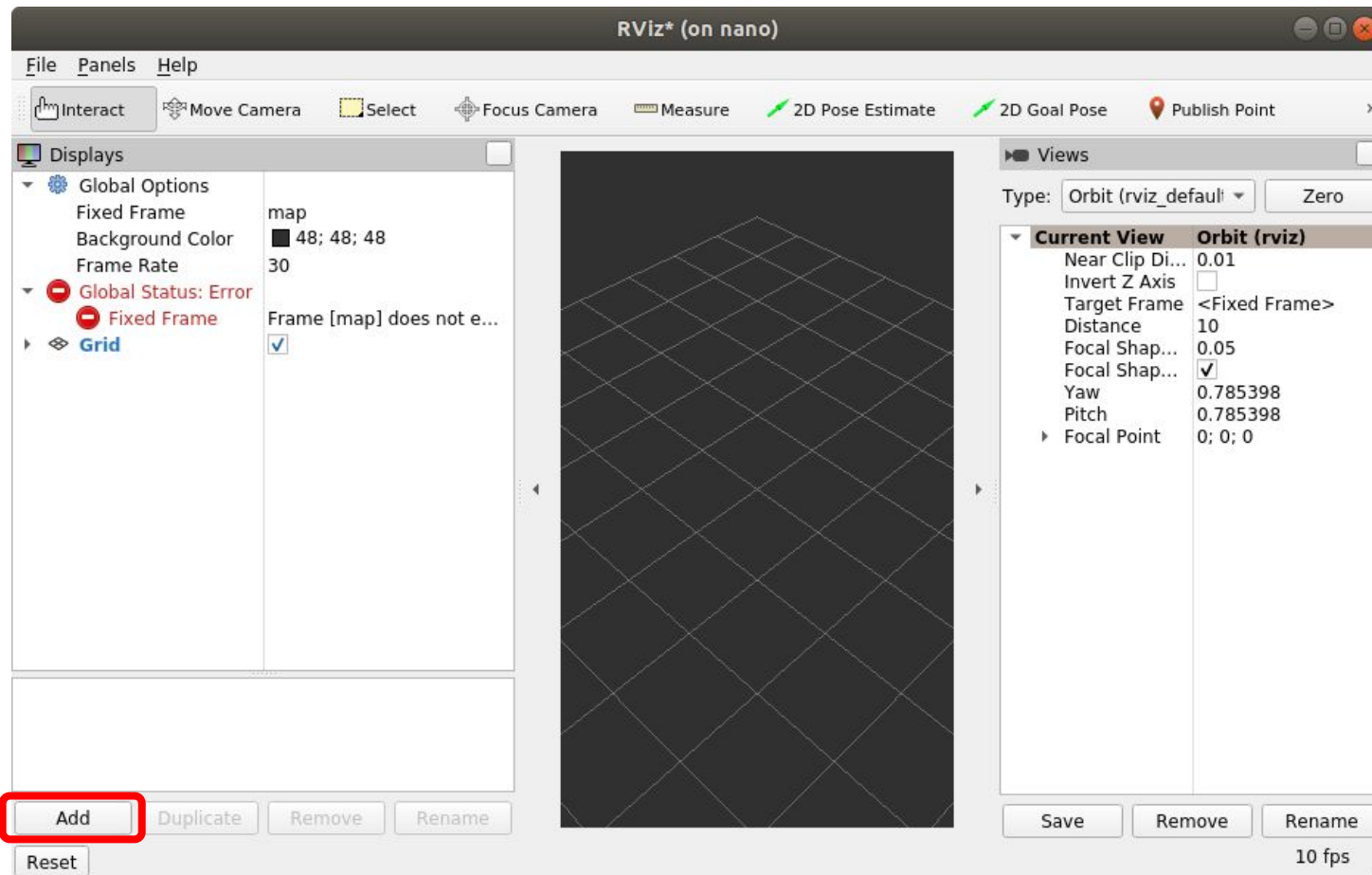
The terminal window at the bottom shows the output of the `rviz2` command:

```
ulting to '/tmp/runtime-root'
[INFO] [1654756966.349268758] [rviz2]: Stereo
is NOT SUPPORTED
[INFO] [1654756966.349728452] [rviz2]: OpenGL
version: 3.1 (GLSL 1.4)
[INFO] [1654756966.621521005] [rviz2]: Stereo
is NOT SUPPORTED
```

A red box highlights the terminal window's title bar, which includes a plus sign icon for opening a new terminal.

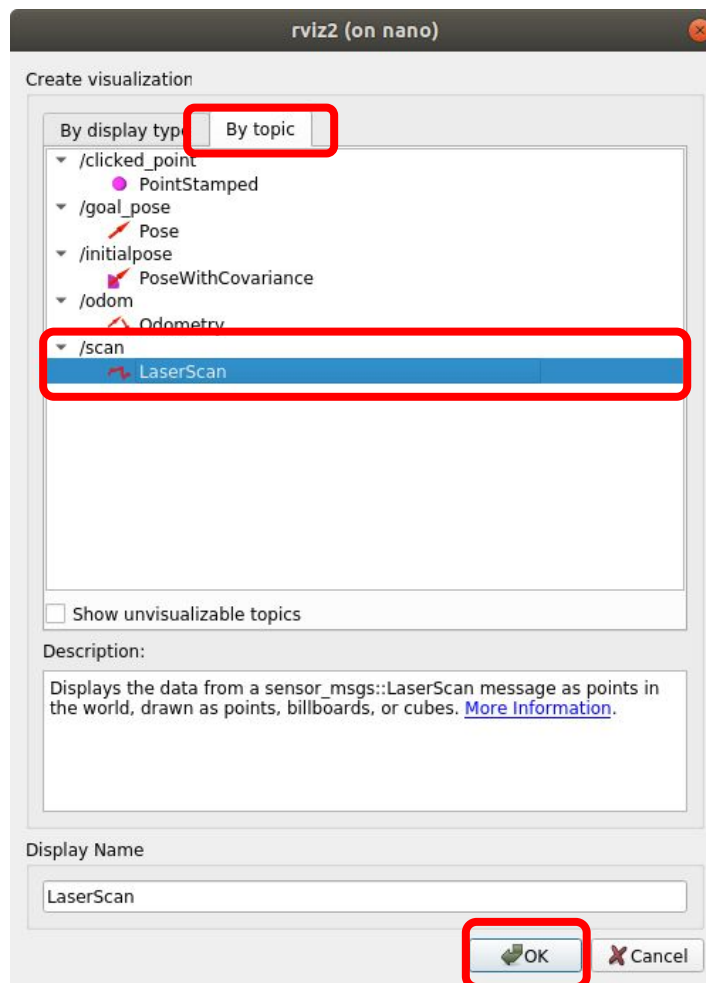
## 02 How to Use

- 사용을 위해 컴파일 하기
  - 열리는 창에서 다음과 같이 설정합니다. (Add 버튼 선택)



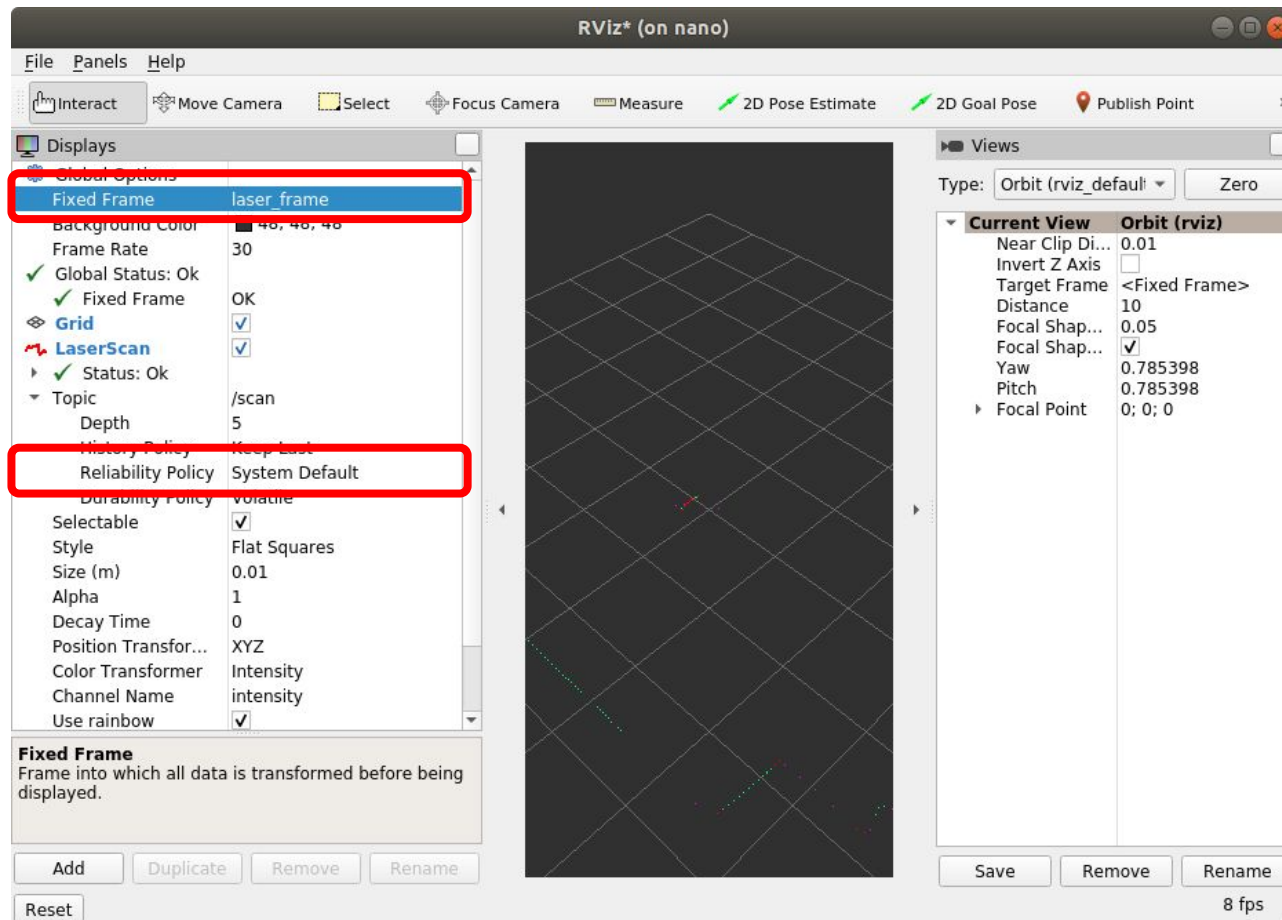
## 02 How to Use

- 사용을 위해 컴파일 하기
  - 열리는 창에서 다음과 같이 설정합니다. (By topic 선택 후, /scan - LaserScan 선택 후 Ok)



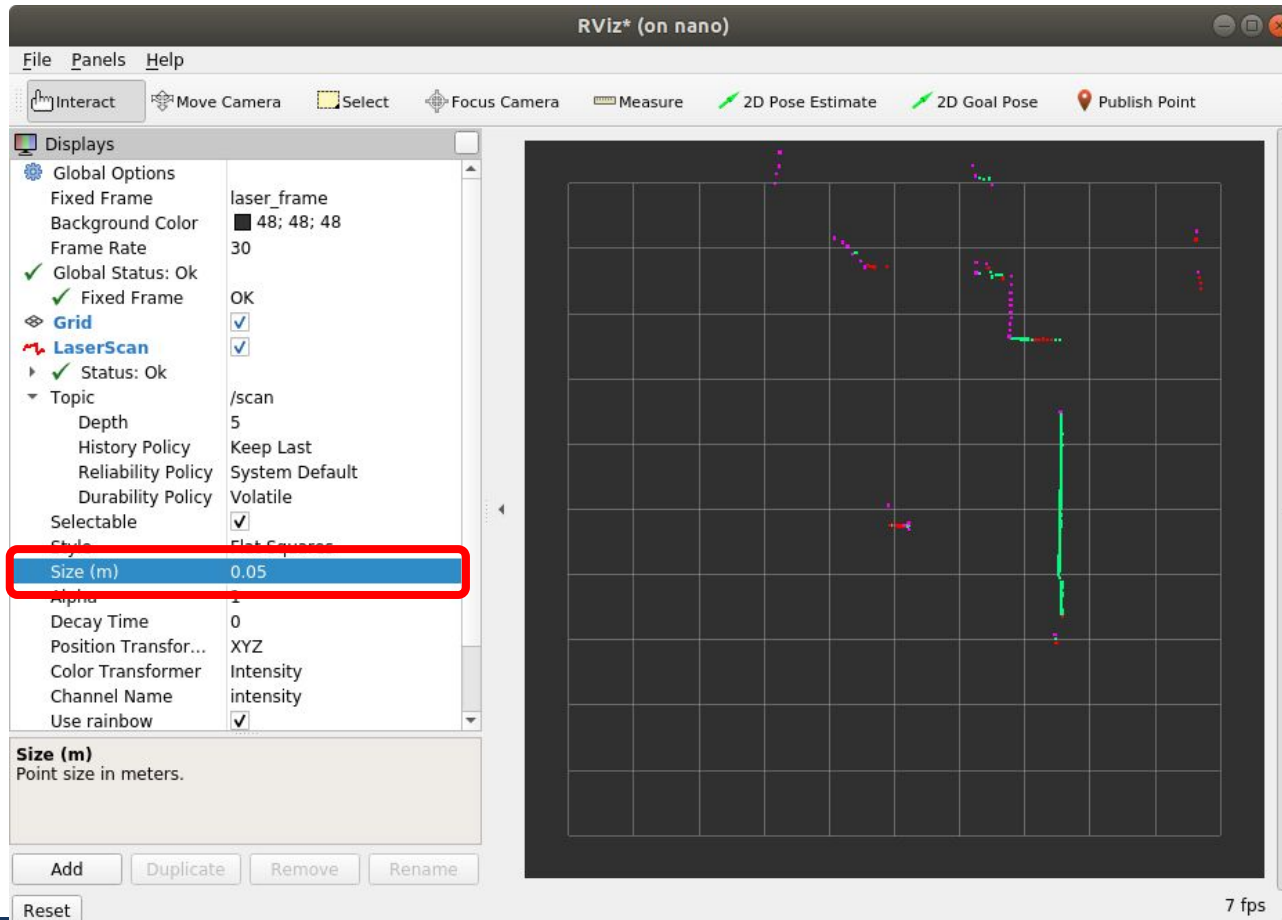
## 02 How to Use

- 사용을 위해 컴파일 하기
  - 열리는 창에서 다음과 같이 설정합니다. (Fixed Frame을 laser\_frame으로, LaserScan - Topic - Reliability Policy - System Default로 설정)



## 02 How to Use

- 사용을 위해 컴파일 하기
  - 열리는 창에서 다음과 같이 설정합니다. (LaserScan - Size를 0.05로 설정)
  - 아래와 같이 LiDAR 데이터를 확인할 수 있으면 정상입니다.





## 02 How to Use

- **사용을 위해 컴파일 하기**
  - 이와 같은 과정을 통해, 기본 패키지를 설치할 수 있으며, 나머지 필요한 패키지를 설치하여 Jetson Nano에서 구동되는 Ubuntu 20.04 - ROS2 Foxy 버전을 사용할 수 있습니다.
  - 현재까지 공식적으로 지원하는 부분은 LIMO 및 LiDAR입니다.



# WeGo Robotics

**Tel.** 031 – 229 – 3553

**Fax.** 031 – 229 – 3554



**제품 문의:** [go.sales@wego-robotics.com](mailto:go.sales@wego-robotics.com)

**기술 문의:** [go.support@wego-robotics.com](mailto:go.support@wego-robotics.com)