First step of building this system is to have an environment that everyone in the team can easily access to code, which is Docker. I familiarized myself with using Docker container of a **ros:foxy** image in an easily accessed and monitored Linux environment with ROS2 Foxy distro setup.

Providing Docker is successfully installed.

Bash

$docker run -it -v <path\_to\_repo>/Docker\_ws/src/:/Docker\_ws/src/ --name f1tenth\_lab1 ros:foxy #to set up ros:foxy environment in the repo path

$ docker start f1tenth\_lab1 #if previously turned off

$ docker exec -it f1tenth /bin/bash

$ source /opt/ros/foxy/setup.bash

$ cd <workspace>/src

$ ros2 pkg create –build-type ament\_python lab1\_pkg3 #form a exec package in workspace

This folder contains 2 Exec nodes: Talker and Relay. Both subscribe to the “ROS2\_Drive” topic. Talker node published incrementals of speed “v” and steering angle “d” sim on timing callback.

Relay node publishes “v” and “d” data x3 on subscriber callback.

Dependencies and required configuration were adjusted inside the .xml and .cfg file

Bash

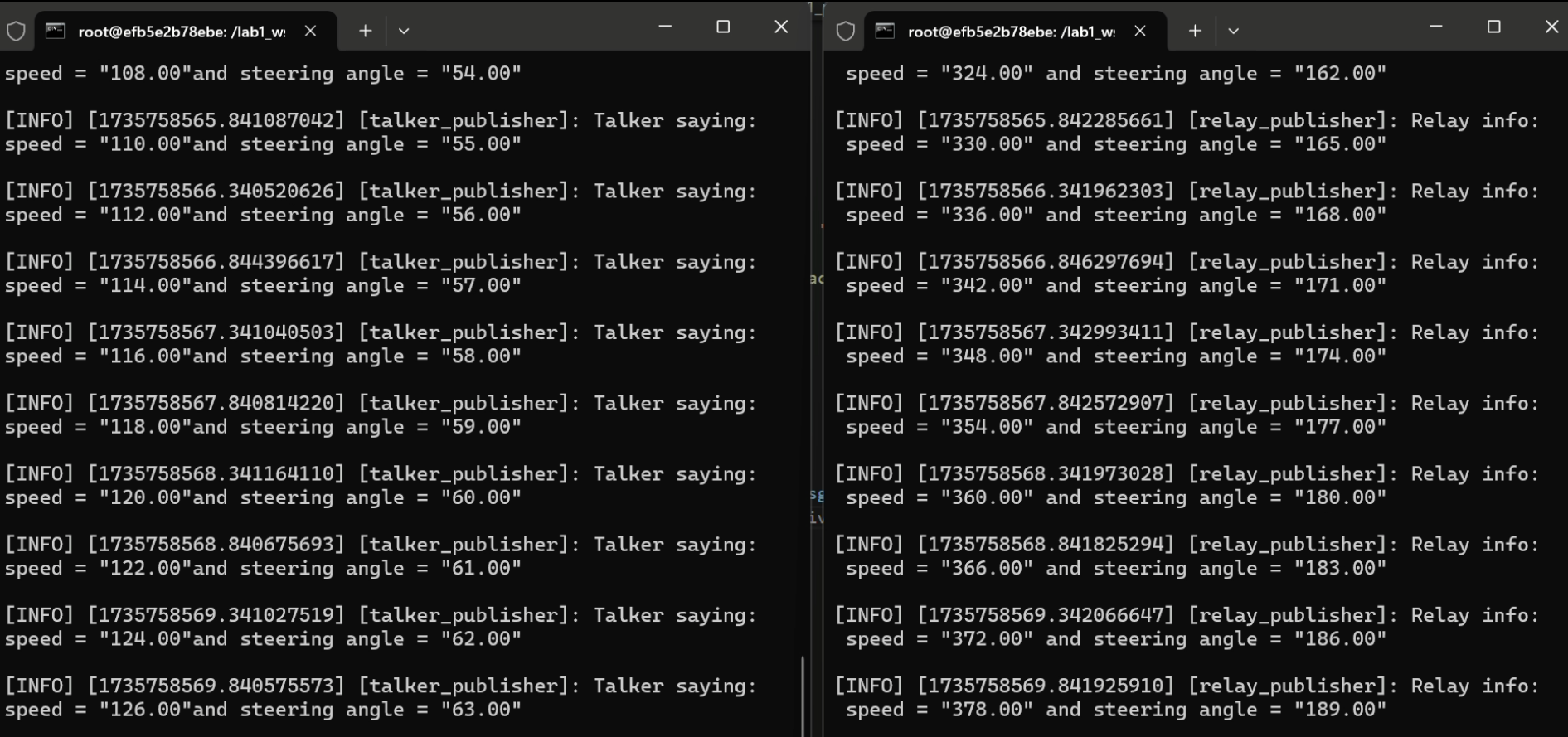
$ cd Docker\_ws

$ colcon build

$ source install/setup.bash

$ rosdep install -i –from-path src –rosdistro foxy -y # install dependencies

$ ros2 run lab1\_pkg3 talker # or $ ros2 run lab1\_pkg3 relay



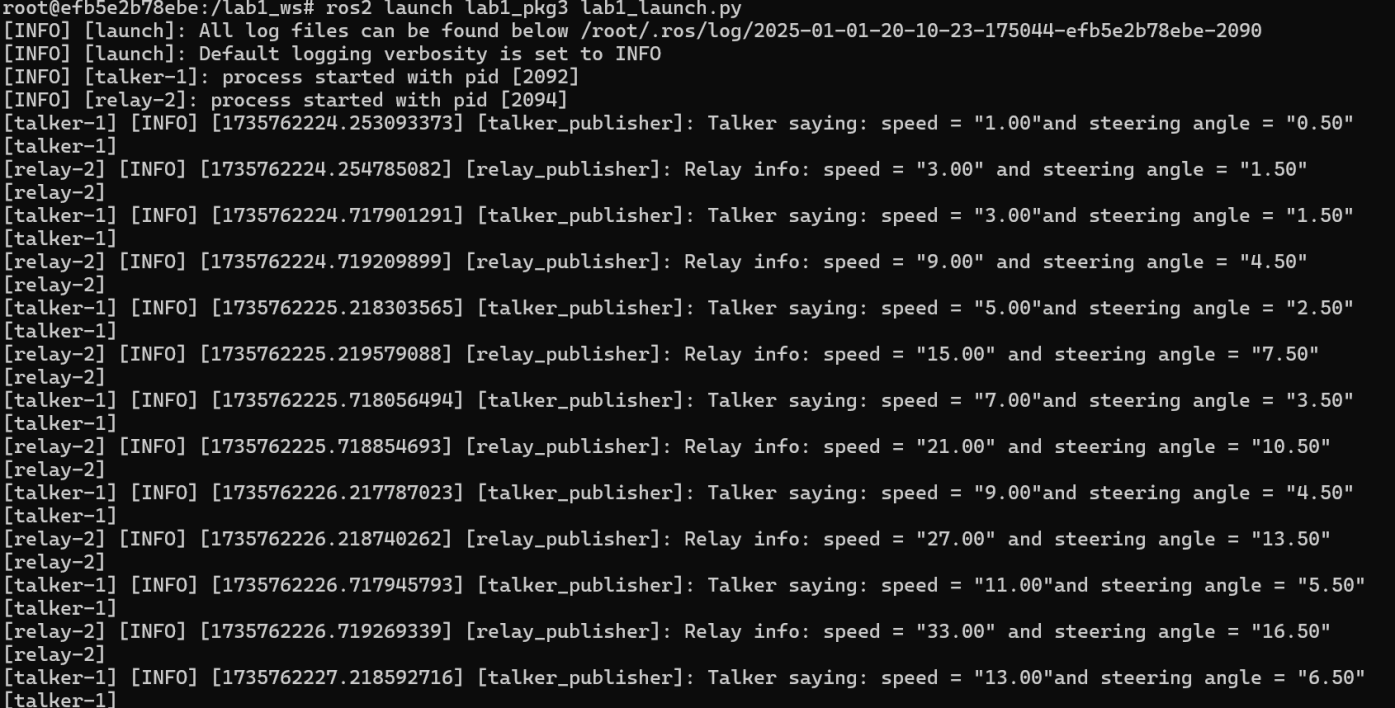
As seen here, 2 different cmd's are running inside the same container.

Both Exec nodes can be run simultaneously using lab1\_launch.py

Bash

$ cd Docker\_ws

$ ros2 launch lab1\_pkg3 lab1\_launch.py



Here, the launch file sets the initial parameters “speed” and “steering angle” to 1 and 0.5 and executes.

By using the launch file, I can open another cmd and observe the nodes change inside a topic.

Bash

$ ros2 topic echo ROS2\_Drive