Installing Autoware ROS2 Stack

Approximate Time Investment: 3 hours

Autoware is one of the foundations of getting our car to drive autonomously. It is a ROS2 stack that acts as a middleware software between ROS2 and the vehicle's ADAS sensors. It is a large stack that can be containerized in Docker or installed from source. We will install it from source and run some basic simulations to check that it has installed correctly. As previously mentioned, open source software is updated frequently so it is best to read through the references below.

References:

- Official Autoware Source Installation Guide

Installing the Autoware ROS2 Stack

1. Follow the instructions for setting up a development environment on the Autoware source installation guide site.

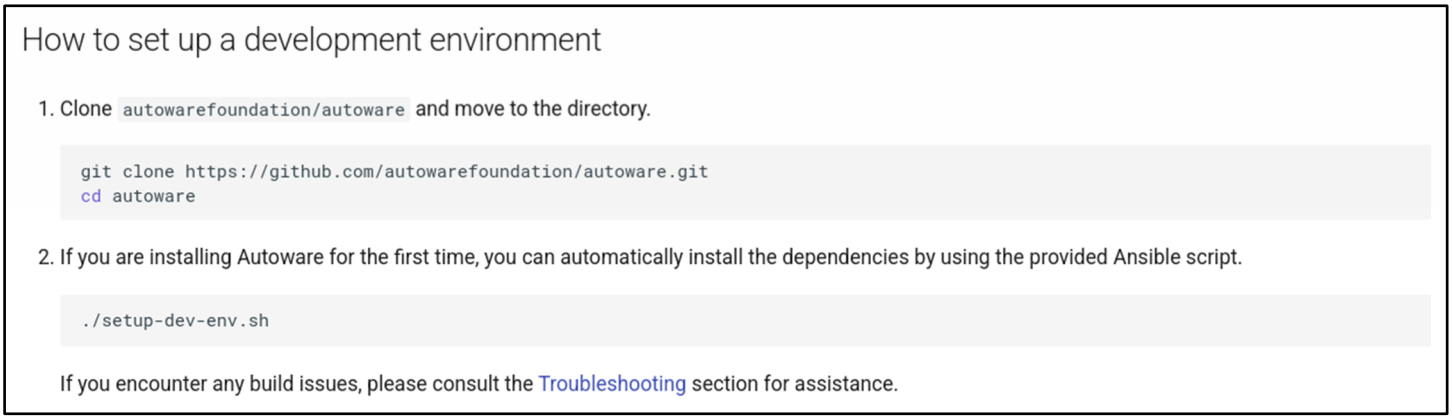

Figure 1: Setting up development environment per Autoware's guide

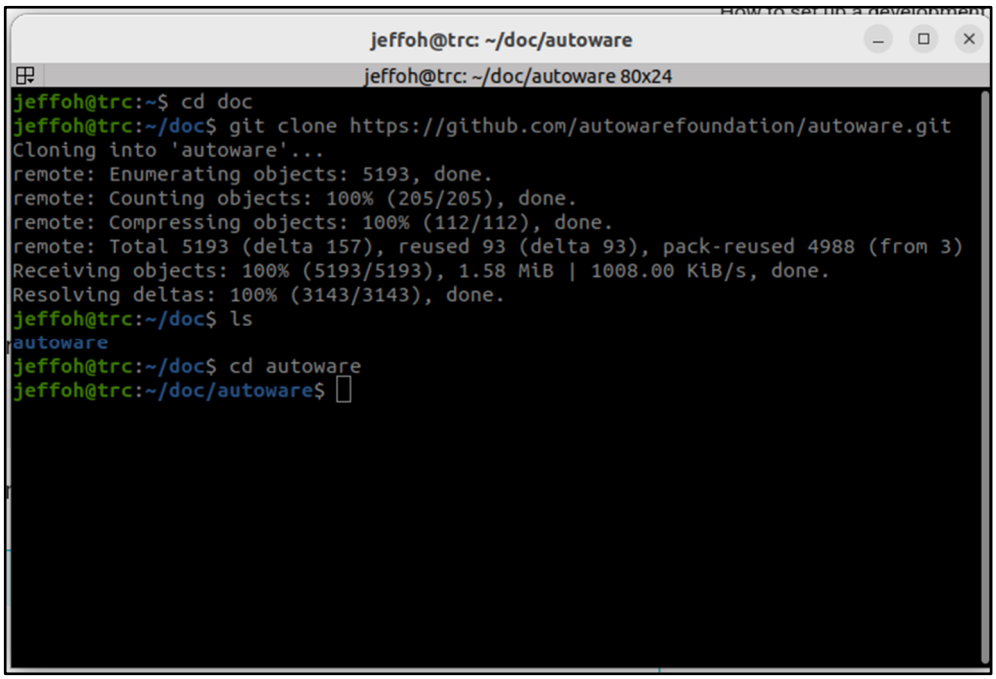

Figure 2: Output of following step 1 in Figure 1

Because I already have an autoware environment in my HOME directory, I have decided to clone the autoware git into a folder called git, see figure 2. You can clone the Autoware repository to any directory best convenient for you.

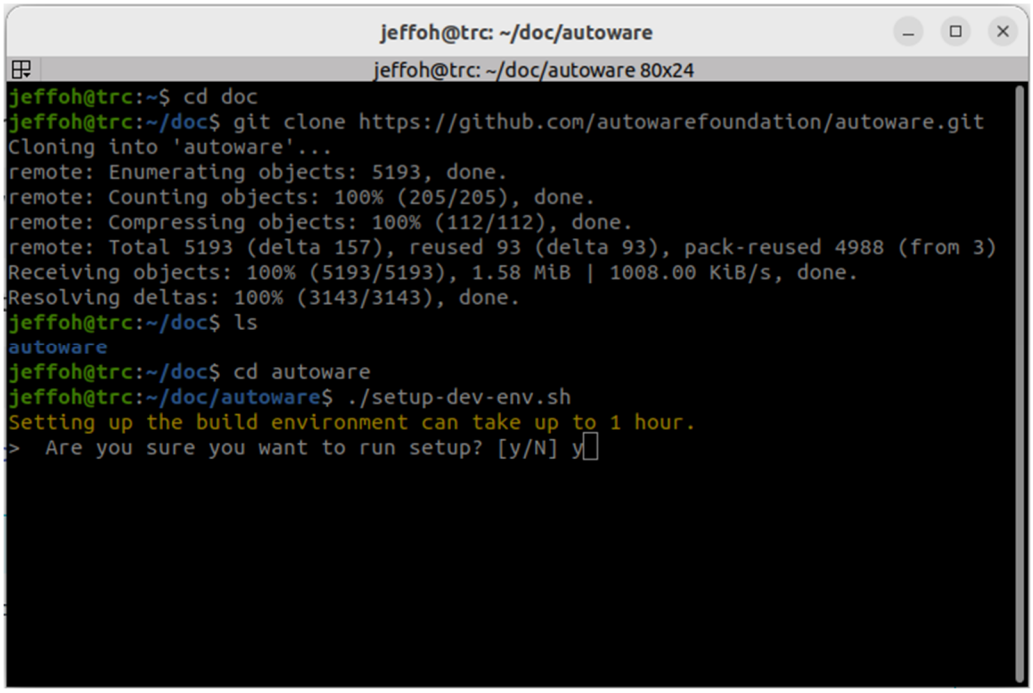

Figure 3: Running step two of setting up development environment (installing dependencies)

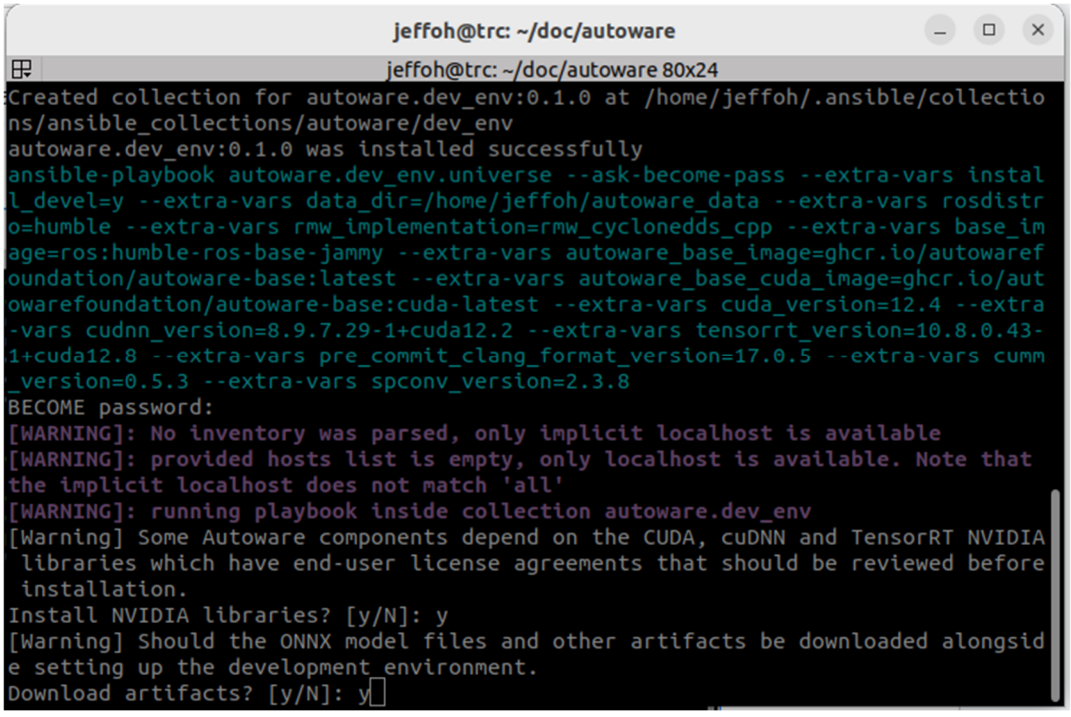

Figure 4: Ansible script for installing dependencies

After running the ansible script in step two, it will ask if you would like to install Nvidia libraries and artifacts. Respond yes to those by typing “y’ and pressing enter. The Nvidia libraries will install Cuda and TensorRT drivers which will enable GPU acceleration when using Autoware. The artifacts addon will install the perception drivers for Autoware.

This dependency install procedure will take anywhere from 30 minutes to 1.5 hours, so be patient and let it run. Once finished, move onto setting up the workspace.

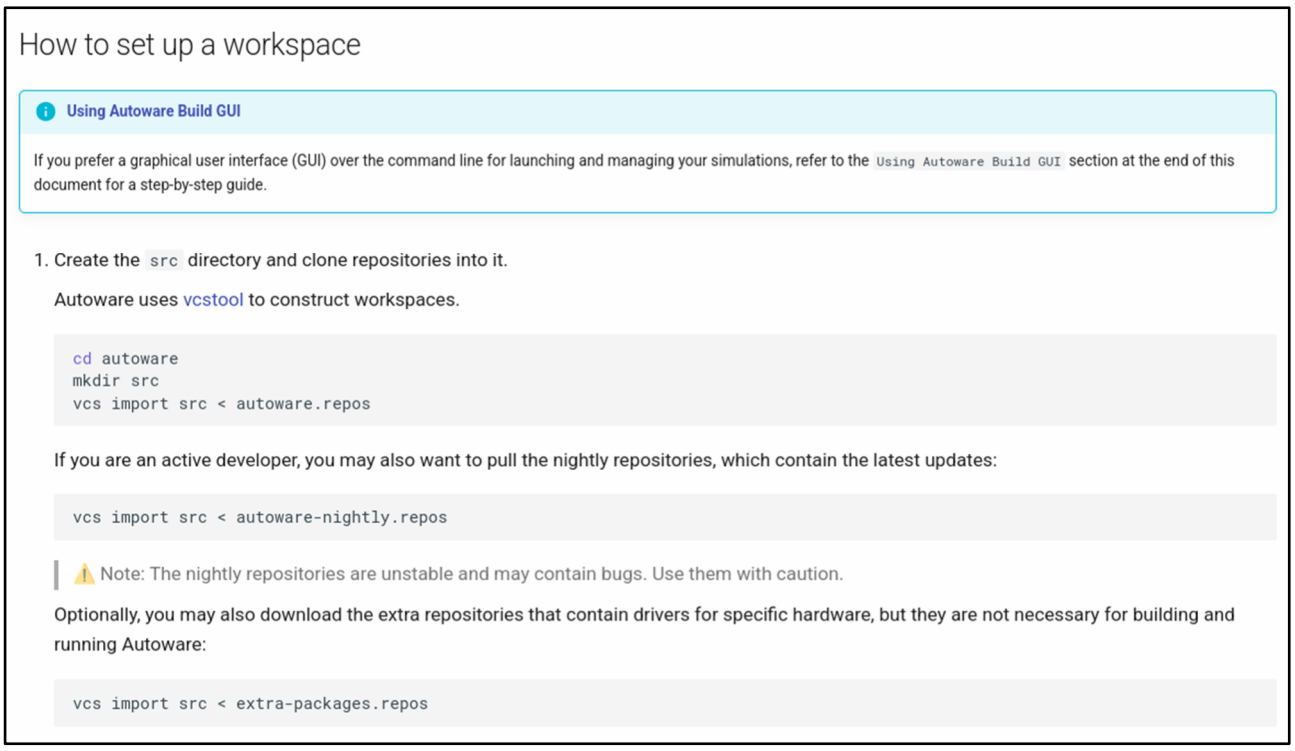

Figure 5: Official Autoware instructions for creating src folder and cloning critical repositories

2. We will follow Autoware's instructions on setting up the workspace. We should now have the workspace folder "autoware" after completing step 1.
    a. Run the following the commands to create the "src" folder and to import Autoware packages.

```
cd autoware
mkdir src
vcs import src < autoware.repos
vcs import src < extra-packages.repos
```

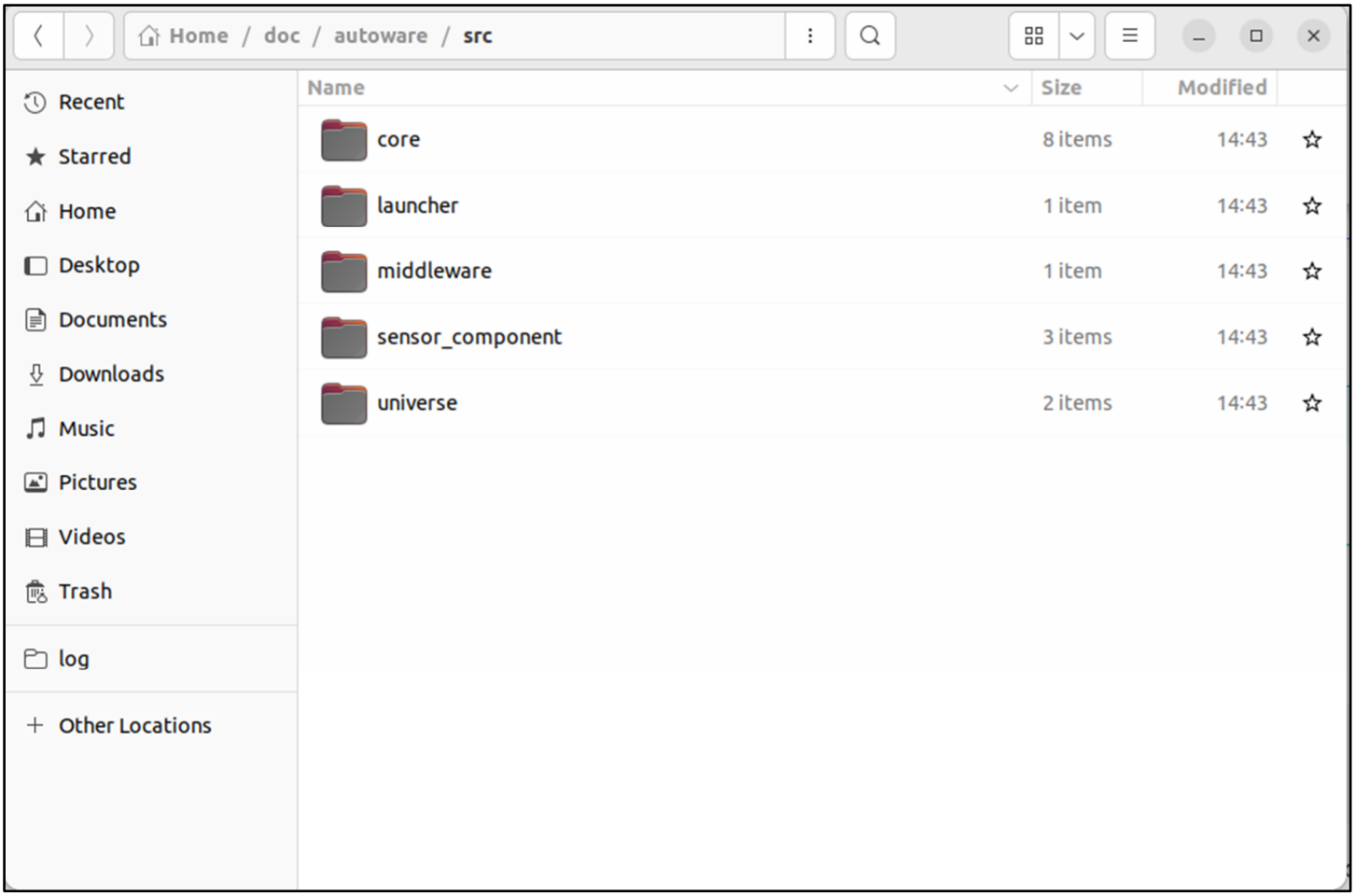

Figure 6: After importing the repositories, our src folder should be populated with Autoware elements

2. Install dependent ROS packages.

   Autoware requires some ROS 2 packages in addition to the core components. The tool `rosdep` allows an automatic search and installation of such dependencies. You might need to run `rosdep update` before `rosdep install`.

```
source /opt/ros/humble/setup.bash
# Make sure all previously installed ros-$ROS_DISTRO-* packages are upgraded to their latest version
sudo apt update && sudo apt upgrade
rosdep update
rosdep install -y --from-paths src --ignore-src --rosdistro $ROS_DISTRO
```

3. Install and set up ccache to speed up consecutive builds. *(optional but highly recommended)*

4. Build the workspace.

   Autoware uses colcon to build workspaces. For more advanced options, refer to the documentation.

```
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
```

   If there is any build issue, refer to Troubleshooting.

5. Follow the steps in Network Configuration before running Autoware.

6. Apply the settings recommended in Console settings for ROS 2 for a better development experience. *(optional)*

Figure 7: Autoware's official instructions continued

3. Now we will install dependent ROS packages and then build the workspace. Run the following commands to do so.

```
sudo apt update && sudo apt upgrade
rosdep update
rosdep install -y –from-paths-src –ignore-src –rosdistro
```

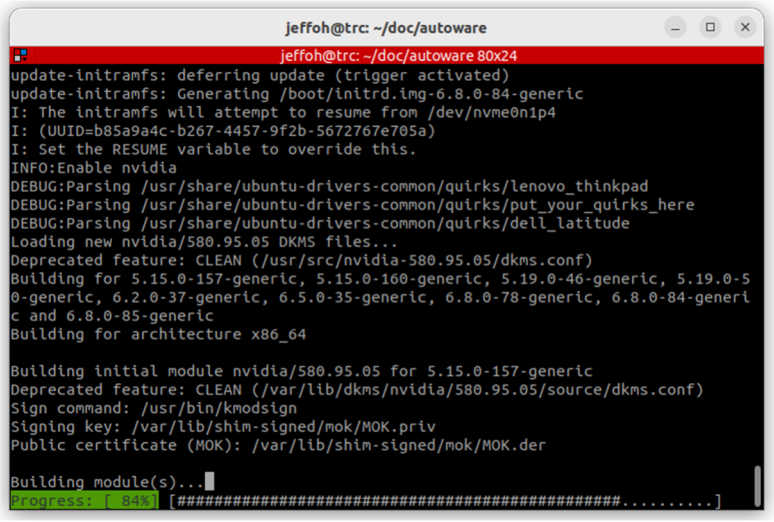

Figure 8: Running sudo apt udate && sudo apt upgrade

a. Now that we have installed ROS dependency packages, we will build our workspace. While in the autoware directory, run the following command. Make sure that you are in your autoware directory in terminal before running this command. There are over 400 packages to build so this process has taken 1 hour and 20 minutes on average. There may be standard errors (stderr) when building some of the packages due to missing dependencies or dependency discrepancy/redundancy in the respective package's code but these can be ignored.

```
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
```

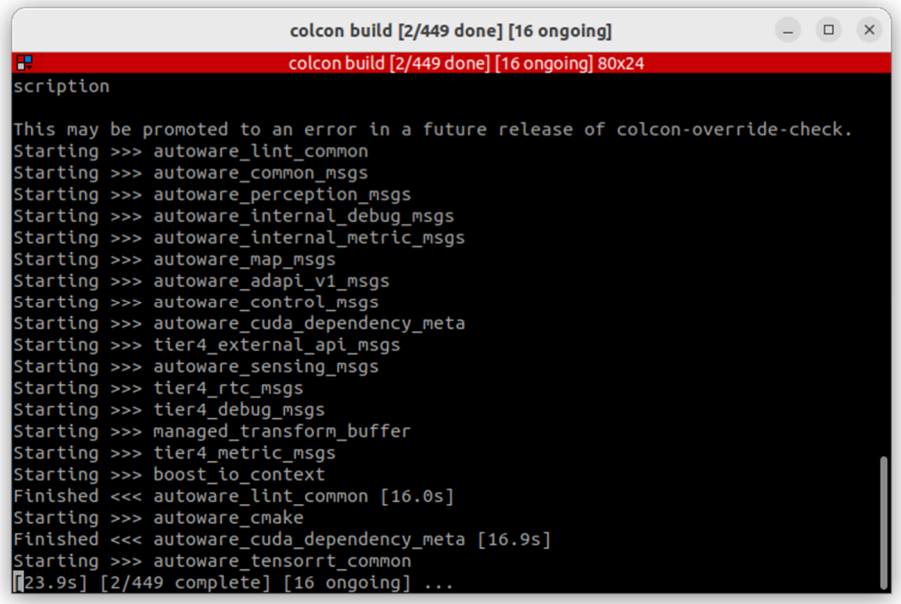

Figure 7: Building the Autoware workspace

4. Now that the Autoware stack is built, we will set up automatic sourcing of the Autoware's setup.bash so ROS2 detects the Autoware packages.

a. Open the bashrc file using the following command.

```
gedit ~/.bashrc
```

b. Add a line near the bottom for sourcing our Autoware workspace's setup.bash file. The directory will vary depending on where you installed Autoware. See the highlighted source line on figure 8.

Figure 8: bashrc file with Autoware's setup.bash file sourced

5. Now that we have installed Autoware and are automatically sourcing the Autoware workspace's setup, we will check that the Autoware packages are available to us with the following command:

Close your current terminal and open a new one so the Autoware setup is sourced prior to running ros2 pkg list.

```
ros2 pkg list
```

You should see packages related to Autoware and the middleware software that came with installation.

Recall that we installed over 400 packages, many of the autoware packages names start with Autoware and the ros2 pkg list command may not list packages starting with the letter A.

Another way to sanity check the list of Autoware packages is the following:

```
ros2 launch autoware   # DO NOT RUN BUT TYPE THIS OUT
# hit the Tap key twice and it will list all the possible package beginning with "autoware"
```

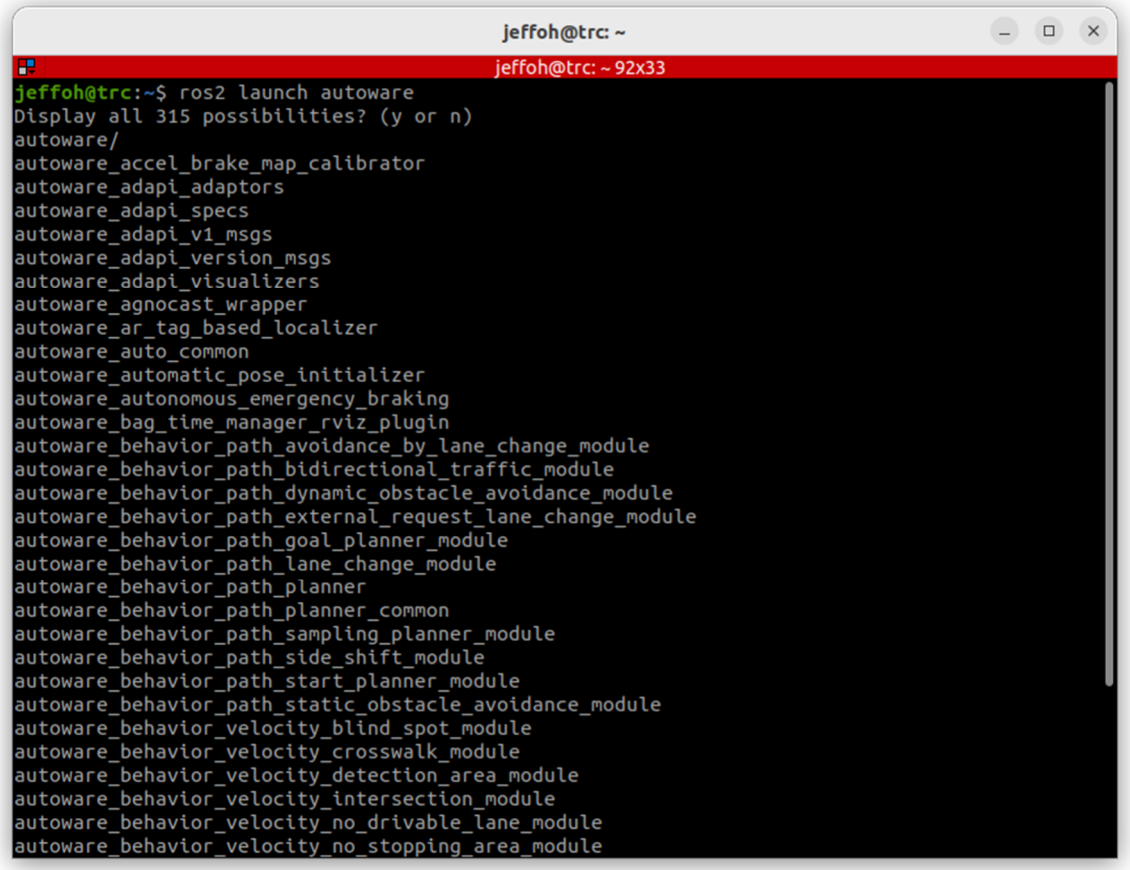

Figure 9: List of Autoware packages populated

We can confirm that Autoware has been installed. We will experiment with Autoware's simulation capabilities in the next module.