Getting a Docker container that does everything we need is not trivial. The following steps is what I was able to get working for my NVIDIA Jetson Orin Nano. If you already have a docker image saved as a .tar.gz file, please look at the end for instructions.

1. Follow system setup (through at least the “Relocating Docker Data Root” step) from <https://github.com/dusty-nv/jetson-containers/blob/master/docs/setup.md>

If you have NVME storage to add, I found this tutorial helpful: <https://www.digitalocean.com/community/tutorials/how-to-partition-and-format-storage-devices-in-linux>

Also, be sure to add the following line to etc/docker/daemon.json:

"data-root": "/mnt/data"

Note: /data should be whatever your mount folder name is…

If you already have a docker container image, skip to the next section!

1. Choose a base container. I like l4t-ml since it contains cuda enabled pytorch, cuda enabled tensorflow2, and cuda enabled opencv. The full list of containers can be found <https://github.com/dusty-nv/jetson-containers/tree/master>

Now, run the container, e.g.:

jetson-containers run $(autotag l4t-ml)

1. Now we can install ROS. For example, for ROS Noetic, follow the instructions here: <https://wiki.ros.org/noetic/Installation/Ubuntu>

IMPORTANT: Install the **ROS-Base** version. The other version will attempt to modify the opencv, which we cannot do! Thus, install the bare bones version and the selectively install additional packages we may need.

The following is also useful to put in your ~/.bashrc file

source /workspace/catkin\_ws/devel/setup.bash

export ROS\_IP=192.168.xx.xx

export ROS\_MASTER\_URI=http://$ROS\_IP:11311

export ROBOT\_ID=x

1. Install any additional ros packages you need, e.g.

sudo apt install ros-noetic-PACKAGE

I installed the following:

* tf2-msgs
* tf
* gmapping
* diagnostic-updater

For the camera to work, I find it necessary to first run:

apt-get purge -y '\*opencv\*'

Then, follow instructions at <https://github.com/satomm1/ros_astra_camera>, with the following exception: run these lines outside of the docker container

./scripts/create\_udev\_rules

sudo udevadm control --reload && sudo udevadm trigger

1. Install other python packages you need.

IMPORTANT: Any package which has an opencv-python dependency must be sure not to modify the opencv-python package already installed from the original container. Updating will break the package!

Some packages I install include:

* spidev
* confluent-kafka
* rospy-message-converter
* pyignite
* matplotlib
* scipy
* ultralytics\*\*\*\*be careful with opencv-python here!

1. Install Cyclone DDS and it’s python binding.

Follow the directions of <https://github.com/eclipse-cyclonedds/cyclonedds> to first install cycloneDDS. Be careful! Check what the latest version of the cyclonedds python binding is and match this version! The install directory should be /path/to/cyclonedds/install. Next:

export CYCLONEDDS\_HOME="$(pwd)/install"

Even better, put this in the ~/.bashrc file since we need this anytime we use the python binding. Last, we install the python binding via:

pip3 install cyclonedds --no-binary cyclonedds

1. Commit the docker container so you can use it later, e.g.:

docker commit c3f279d17e0a ml\_ros:latest

1. Now, to run the container, we can use the command:

jetson-containers run $(autotag ml\_ros:latest)

You can include any of the usual docker commands with this. For example, my full command is:

jetson-containers run -v ~/workspaces/catkin\_ws:/workspace/catkin\_ws -v ~/gemini\_api:/gemini\_code -v /dev/bus/usb:/dev/bus/usb -v /dev/video0:/dev/video0 -v /dev/video1:/dev/video1 -i --device=/dev/ttyUSB0 --device=/dev/spidev0.0 --rm --privileged --name ros\_noetic $(autotag ml\_ros:latest)

OR

sudo docker run --runtime nvidia --network=host -v ~/workspaces/catkin\_ws:/workspace/catki

n\_ws -v ~/gemini\_api:/gemini\_code -v /dev/bus/usb:/dev/bus/usb -v /dev/video0:/dev/video0 -v /dev/video1:/dev/video1 -it --device=/dev/ttyUSB0 --device=/dev/spidev0.0 --rm --

privileged --name ros\_noetic ml\_ros:latest

1. To save the image (for easy loading on a new machine), use the following command:

docker save myimage:latest | gzip > myimage\_latest.tar.gz

If everything has gone according to plan, you should now have a docker container which has ROS, cuda enabled pytorch/tensorflow, and everything else you might need!

**What if I already have a docker image saved as a \*.tar.gz file?**

In this case, just set docker up and the call:

docker load < your\_image.tar.gz

You should source ROS via:

source /opt/ros/noetic/setup.bash

**Setting Up ROS Workspace**

1) Start the ROS docker container. Within the docker container, navigate to /workspace/catkin\_ws (you may need to create this directory):

cd /workspace/catkin\_ws

2) Now, create the devel and src directories:

mkdir devel src

3) Now, build the workspace:

catkin\_make

(if you receive an error, make sure to source the setup file: source /opt/ros/noetic/setup.bash)

4) Source the build and go to the src directory:

source devel/setup.bash

cd src

5) Now, clone the following repositories and switch to the noetic branch. To clone the noetic branch use:

git clone -b noetic https://github.com/satomm1/<repo name>.git

Git repositories to clone:

mattbot\_record

mattbot\_bringup mattbot\_teleop

mattbot\_dds ros\_astra\_camera

mattbot\_image\_detection rplidar\_ros

mattbot\_mcl twist\_mux (noetic-devel)

mattbot\_navigation slam\_gmapping

6) Navigate back to the catkin\_ws directory and build the packages:

cd /workspace/catkin\_ws

catkin\_make

source devel/setup.bash

7) Add important information to the bashrc file:

nano ~/.bashrc

Append the following:

source /workspace/catkin\_ws/devel/setup.bash

export ROS\_IP=<Your Robot IP Here>

export ROS\_MASTER\_URI=http://$ROS\_IP:11311

export ROBOT\_ID=<Robot ID Here (Integer)>

export CYCLONEDDS\_HOME="/home/cyclonedds/install"

Source the bashrc file:

source ~/.bashrc

8) Test the setup, call roscore. If you have no errors, this is good!

roscore

9) Set up the Astra Camera. You will also need to navigate to the ros\_astra\_camera directory and perform these two commands **outside of the docker container**:

cd ~/workspaces/catkin\_ws/src/ros\_astra\_camera

./scripts/create\_udev\_rules

sudo udevadm control --reload && sudo udevadm trigger

**Setting up the Gemini Container**

**TODO**

Gemini Container:

docker run -v ~/gemini\_api:/gemini\_code -v ~/Desktop/audio:/audio -it --rm --privileged -p 5000:5000 --name gemini gemini:latest