Setting up the BRAIN project on a fresh NVIDIA Jetson AGX/Nano

Alex Baldwin

August 2025

Abstract

This document covers installing the BRAIN vision compute stack on an NVIDIA embedded compute device. The proceedures outlined below have been tested to work on an NVIDIA Jetson AGX Orin and an NVIDIA Jetson Orin Nano.

Contents

1 Prerequisites

The current BRAIN stack is designed to run within docker, with host NVIDIA CUDA drivers of 12.6 or later. The official NVIDIA images come with a full desktop environment and a lot of other bloat. Therefore, BRAIN is currently tested on a host based on the ARM64 Tegra Ubuntu Server 22.04 release. We target a minimum firmware version of 36.4, which is sometimes installed by default. You should check the firmware version on the UEFI BIOS screen upon first boot. If your firmware version is older than 36.4, you will need to follow the reflashing step detailed in subsection 2.1 before continuing. The official NVIDIA images boot the kernel driectly using ExtLinux, but the Ubuntu Server release uses GRUB. To change this setting, you must hit *ESC* to enter the UEFI BIOS settings, and navigate to the submenu .

Side note specific to the Orin AGX

The AGX has an internal 32GB eMMC chip which is usually flashed with an old NVIDIA Ubuntu Desktop image. This can be useful to have around, but to ensure consistency, we will use a SD card for both the AGX and Nano OS images. The AGX will want to boot from the eMMC by default, so you must hit *ESC* to enter the UEFI BIOS settings, and change the default boot device to UEFI SD card.

1.1 Configurig the base OS image

By default, the Ubuntu Server image will attempt to mount a cloud-init config file. After about 1m30s it will abort, and create a default user with the username `ubuntu` and password `ubuntu`. You will be required to change the password immediately.

Start by creating a new user for deployment, adding them to `sudo` group:

```
ubuntu@ubuntu:~$ sudo adduser deploy
[sudo] password for ubuntu:
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for deploy
Enter the new value, or press ENTER for the default
        Full Name []: Deployer McDeployface
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
ubuntu@ubuntu:~$ sudo usermod -aG sudo deploy
```

NOTE: We use the default password `brain!` for non-networked deployments for simplicity. You MUST change this to a secure random value from `pwgen 16` or similar before deploying to a networked environment.

You should then set the hostname for the device. In order for this to fully take affect, you should reboot the device.

```
ubuntu@ubuntu:~$ sudo hostnamectl hostname brain0
ubuntu@ubuntu:~$ sudo reboot
```

At this point you may wish to switch to using the `deploy` user via SSH. Connect the Ethernet port to a LAN with DHCP and issue the command `ip a` to get the IP address assigned.

```
ubuntu@ubuntu:~$ sudo apt update
```

NOTE: For older versions of BRAIN, it was neccesary to set up API keys for NVIDIA's container repository. You can do this by signing up for an NVIDIA developer account, and proceeding to https://org.ngc.nvidia.com/setup/api-keys. Supply the bearer token as follows:

```
deploy@brain:~/git/brain$ docker login nvcr.io
Username: $oauthtoken
Password: <Your Key>
```

1.2 Testing using the NOC L4T base image

The NOC L4T image is a pre-built docker image that contains PyTorch, torchvision and ultralytics (for using YOLO models). It also includes a self-test script as a placeholder to be overwritten with your intended application. This is useful for testing your configuration is correct. Your output should look similar to below:

```
deploy@brain:~/git/brain/vision$ docker run --runtime nvidia -it brain/l4t-base:j62
    -r36.4-2
NOC L4T (Linux 4 Tegra) base image test script

Y Standard dependencies loaded
Y PyTorch CUDA support
    PyTorch Version: 2.8.0
    CUDA Version: 12.6
    CUDA Device: Orin
Y Torchvision loaded
Y OpenCV loaded
Y Ultralytics YOLO loaded

If everything comes back without error, you have configured your Jetson correctly!
This is a base container, you should extend from it with your own application.
```

2 Debugging

2.1 Reflashing firmware

With the software stack on the NVIDIA Jetsons being a little unstable, it's entirely possible to break the firmware from a simple software upgrade. There are a few different ways to reflash firmware on Jetson devices, but the most reliable is to use the `flash.sh` tool supplied by Cannonical. NVIDIA's official instructions are to use the NVIDIA sdk-manager application, but I have found this to be extremely unreliable and time consuming.

Firmware for all Orin devices, including flash.sh can be found at https://developer.nvidia.com/-downloads/embedded/l4t/r36_release_v4.3/release/Jetson_Linux_r36.4.3_aarch64.tbz2. The instructions below are derived from instructions found on Ubuntu's NVIDIA Jetson download page.

2.2 Side note specific to the Orin Nano

One alternative method is to first boot the JetPack 5.1.3 SD card image (https://developer.nvidia.com/-downloads/embedded/l4t/r35_release_v5.0/jp513-orin-nano-sd-card-image.zip) and to use apt to update the firmware. This has had mixed results in the past and is not reccomended.