RV500-DY

Military Jetson AGX THOR Rugged Computer

User's Manual

Revision Date: Jun.12. 2026

Safety Information

Electrical safety

- To prevent electrical shock hazard, disconnect the power cable from the electrical outlet before relocating the system.
- When adding or removing devices to or from the system, ensure that the power cables for the devices are unplugged before the signal cables are connected. If possible, disconnect all power cables from the existing system before you add a device.
- Before connecting or removing signal cables from the motherboard, ensure that all power cables are unplugged.
- Seek professional assistance before using an adapter or extension cord. These devices could interrupt the grounding circuit.
- Make sure that your power supply is set to the correct voltage in your area.
- If you are not sure about the voltage of the electrical outlet you are using, contact your local power company.
- If the power supply is broken, do not try to fix it by yourself. Contact a qualified service technician or your local distributor.

Operation safety

- Before installing the motherboard and adding devices on it, carefully read all the manuals that came with the package.
- Before using the product, make sure all cables are correctly connected and the power cables are not damaged. If you detect any damage, contact your dealer immediately.
- To avoid short circuits, keep paper clips, screws, and staples away from connectors, slots, sockets and circuitry.
- Avoid dust, humidity, and temperature extremes. Do not place the product in any area where it may become wet.
- Place the product on a stable surface.
- If you encounter any technical problems with the product, contact your local distributor

Statement

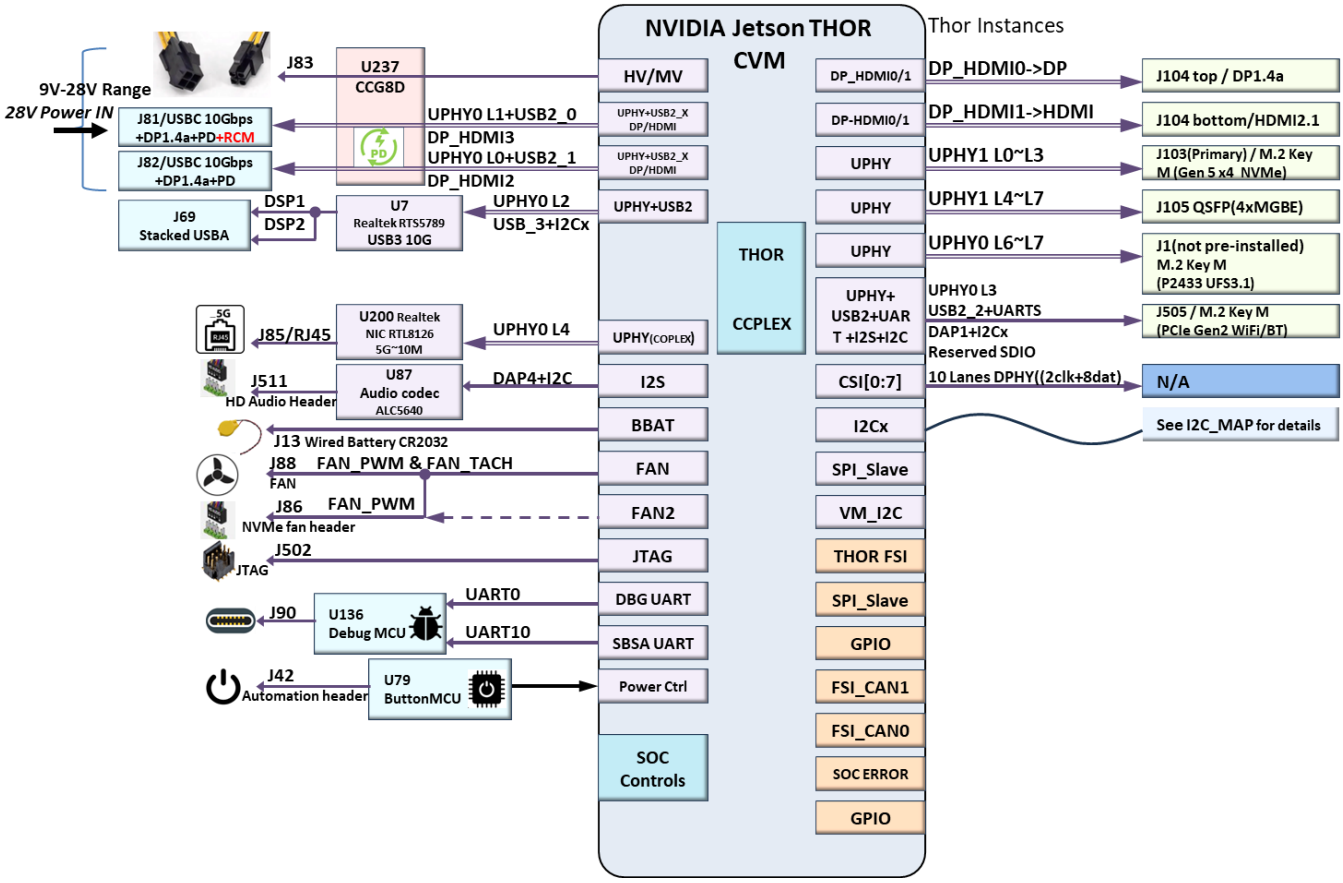
Table of Contents

Specifications

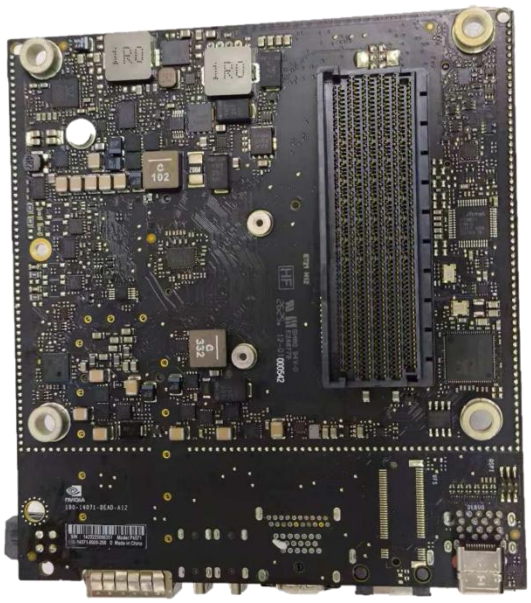
System	
AI Performance	2,070 TFLOPS (FP4—Sparse)
High performance Processor	14-core Arm® Neoverse®-V3AE 64-bit CPU 64 KB I-Cache, 64 KB D-Cache 1 MB L2 cache per core 16 MB shared system L3 cache
GPU	2560-core NVIDIA Blackwell architecture GPU with 96 fifth-gen Tensor Cores Multi-Instance GPU (MIG) with 10 TPCs
Memory	128 GB 256-bit LPDDR5X, 273 GB/s
Expansion Slot	M.2 Key E slot with x1 PCIe Gen5 (populated with WiFi and Bluetooth module)1x M.2 2230 E Key
Display	
Graphics Interfaces	1 x HDMI 2.0(max resolution 3840x2160) 1 x Display Port 1.4a
Storage	
M.2	1 x M.2 Key M slot with x4 PCIe Gen5 (populated with 1 TB NVMe)
Ethernet	
Controller	1 x 5GbE RJ-45 (RTL8126) 1 x QSFP for 4x 10GbE(Default) or 25GbE
Front I/O	
Power Button	1 x IP65 Power Button
Power In	9V~28V DC-IN with Phoenix Jack
Networking I	1 x QSFP28, 4x 10GbE(Default) or 25GbE
USB	2 x USB3.1 Type-C (one for Recovery)
Display	1 x HDMI 2.0b 1 x Display port 1.4a
Networking II	1 x 5GbE RJ45 connector
USB	2 x USB3.2 Type-A
Access I/O	
Debug	1 x USB Type-C
Recovery	1 x Recover Button
Reset	1 x Reset Button

Inner I/O	
	2 x 13-pin CAN header 2 x 6-pin automation header 2 x 5-pin JTAG connector 1 x 4-pin fan connector—12 V, PWM, and Tach 2 x 5-pin audio panel header 2-pin RTC backup battery connector
Power Requirement	
Power Input	9-28V DC-in; 40-130W
Applications, Operating System	
Applications	Energy/Smart Grid/Power Plant Management, Intelligent Automation and manufacturing applications/ AI
Operating System	Ubuntu 24.04 with JetPack7.X
Physical	
Dimensions	250 x 250 x 118 mm (W x D x H)
Weight	TBC
Chassis	Aluminum Alloy AL6061
Heatsink	Aluminum Alloy, Corrosion Resistant
Finish	Anodic aluminum oxide ; Color: PANTONE 7743C , Black (optional)
Environmental	
Designed to meet	MIL-STD-810G, IEC-61850-3, IEEE-1613, CE and FCC, RoHS
Operating Temp.	-25°C to +55°C
Storage Temp.	-40°C to +85°C
Relative Humidity	5% to 95%, non-condensing

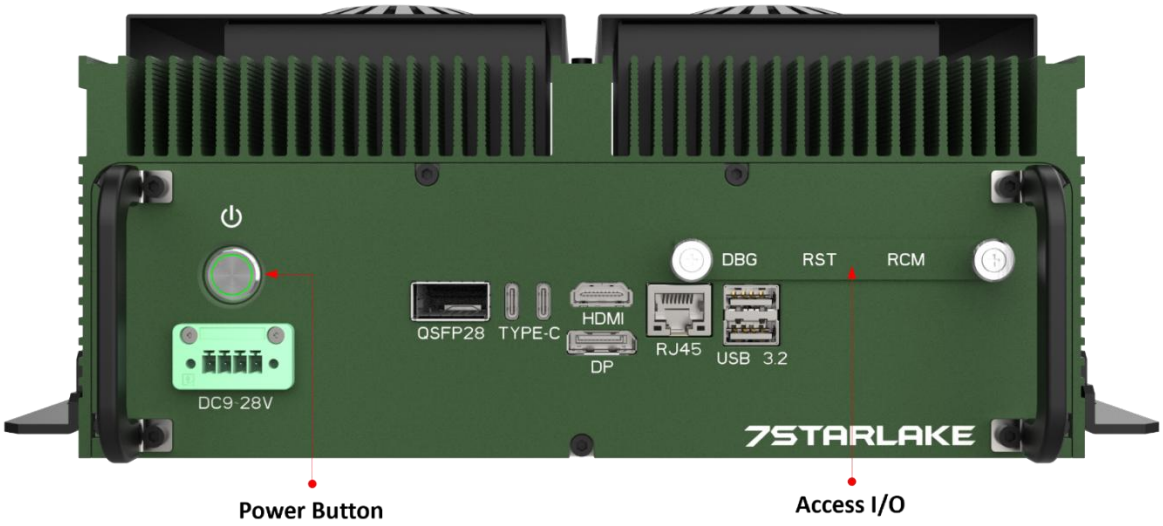
Block Diagram

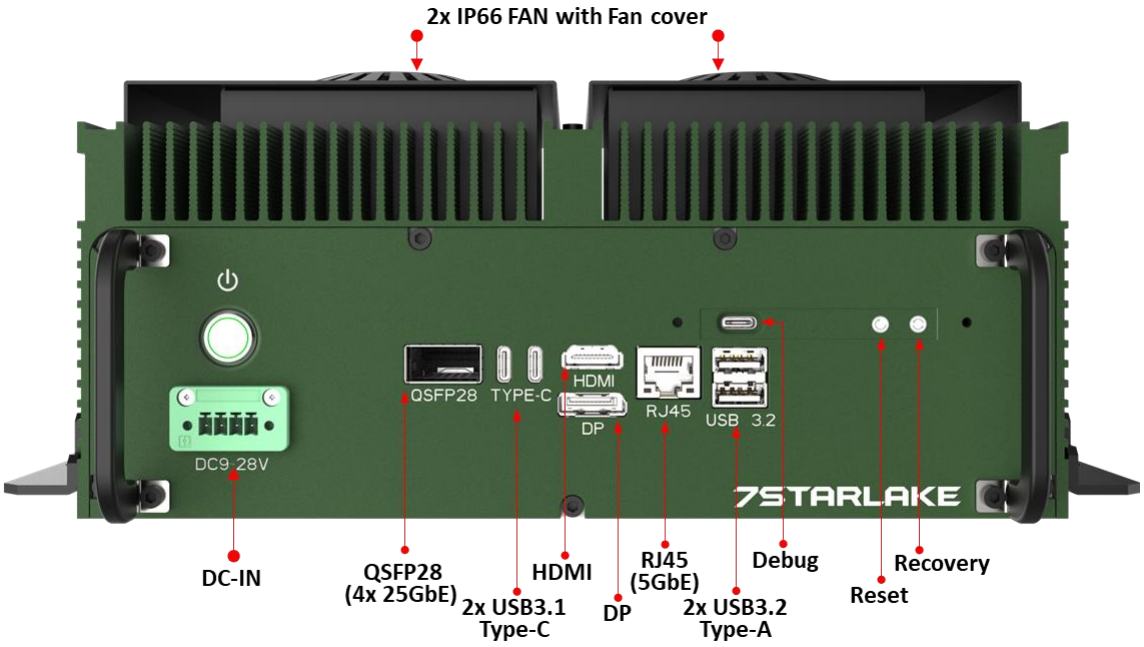


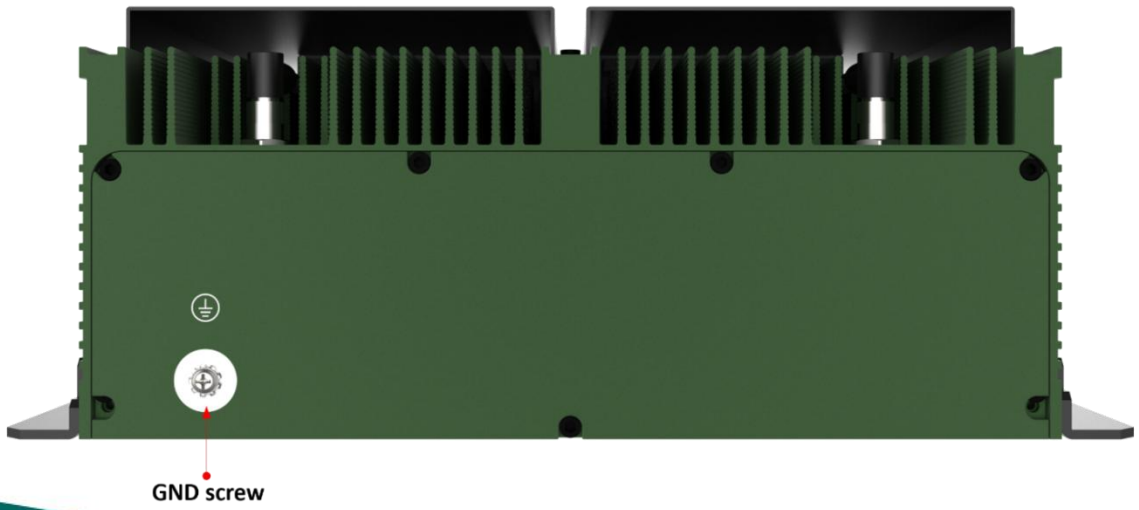
Board Visuals



Front/Rear IO Visuals







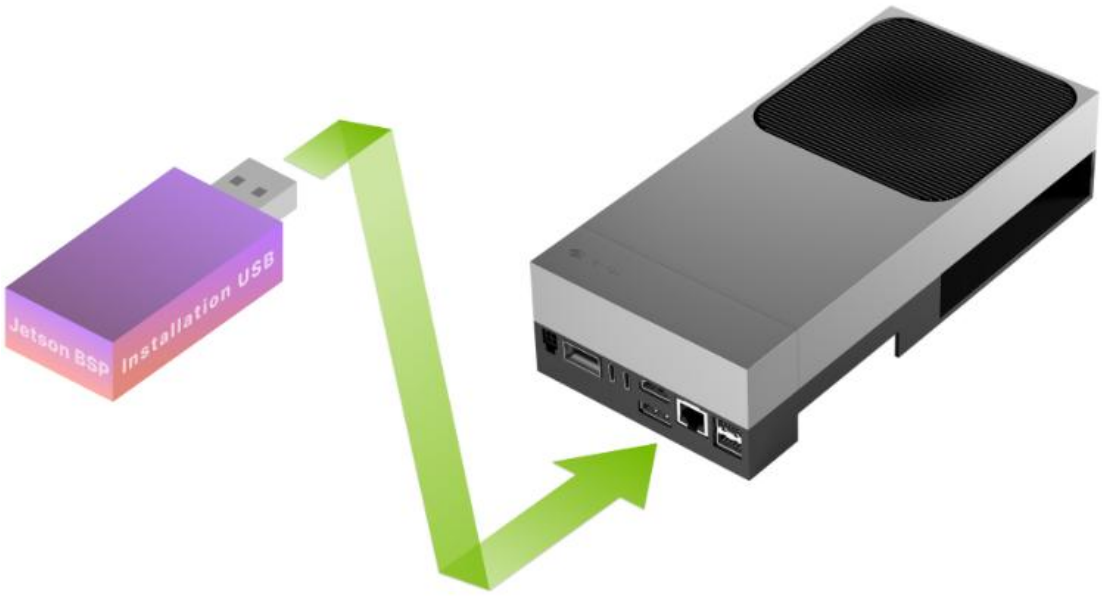
Ordering Information

Model No	NV500	NV500-IR	NV500-DY
GPU	2560-core NVIDIA Blackwell architecture GPU with 96 fifth-gen Tensor Cores Multi-Instance GPU (MIG) with 10 TPCs, 1.57 GHz Max.		
Memory	128 GB 256-bit LPDDR5X, 273 GB/s		
AI Performance	2070 TFLOPS		
CPU	14-core Arm® Neoverse®-V3AE 64-bit CPU 64 KB I-Cache, 64 KB D-Cache, 1 MB L2 cache per core 16 MB shared system L3 cache, 2.6GHz Max.		
Module total module power	120W is default power mode for Jetson T5000. Maximum Total Module Power (TMP): 130W GPU/CPU clocks are throttled when TMP exceeds 130W for Jetson T5000		
Storage	1x M.2 NVMe 1TB		
Front I/O			
Power Button	1x IP65 Power button	1x IP65 Power button	1x IP65 Power button
X1	18V~32V DC-IN with D38999 connector	9V~28V DC-IN with M12 connector	9V~28V DC-IN with Phoenix Jack
X2	1x QSFP with MPO D38999	1x QSFP with MPO D38999	1x QSFP28 jack
X3	1x HDMI with D38999	1x HDMI with M20 connector	2x USB3.1 type-C port
X4	1x USB3.0 Type A with D38999	1x USB3.0 with M20 connector	1x HDMI 2.0b port
X5	1x USB2.0+ 1x RS232+ 2x CAN+4x GPIO with D38999	1x USB2.0+1x RS232 with M12connector	1x Display port 1.4a port
X6	NA	2x CAN+4x GPIO with M12 connector	1x 5GbE RJ45 connector
X7	NA	NA	2x USB3.2 Gen.2 type A
Rear I/O			**Front Access cover**
Access panel	1x Debug USB Type-C	1x Debug USB Type-C	Reset Button
	1x Recovery USB Type-C	1x Recovery USB Type-C	Recovery mode Button
	1x Recovery Button	1x Recovery Button	1x USB type-C Debug port
	1x Reset Button	1x Reset Button	
Ground	1x GND Screw		
Color	Pantone 7743C		
Dimensions	250 x 250 x 118mm(WxDxH)		

Software Information

Quick Start Guide

- https://docs.nvidia.com/jetson/agx-thor-devkit/user-guide/latest/quick_start.html



Overview

In this guide, we will walk you through the steps to quickly install the BSP (Jetson Linux) on your Jetson AGX Thor Developer Kit using a bootable installation USB stick.

It's a new process introduced with JetPack 7.0 (and continued in later versions) that enables you to install the Jetson BSP (Jetson Linux) without using a host Ubuntu PC, making the initial setup process simpler and similar to how you would install Ubuntu on your laptop or a PC. .

Attention

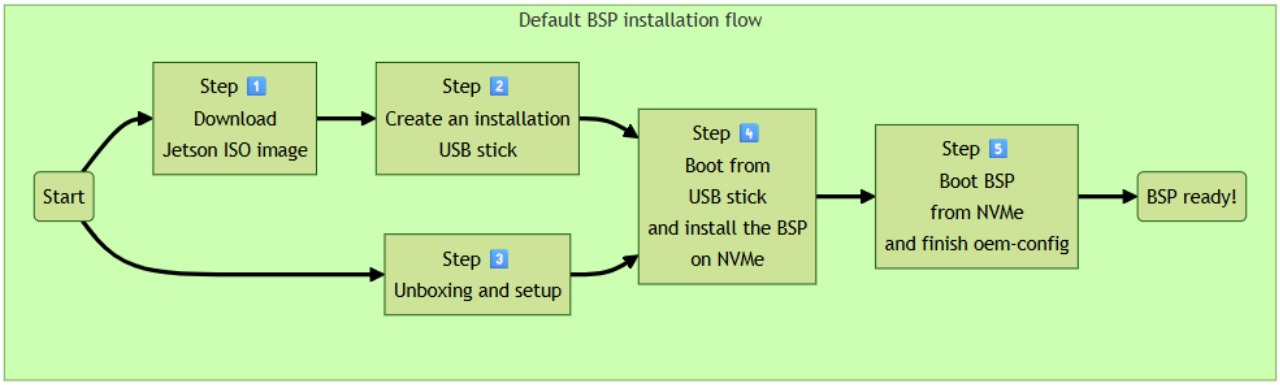
Jetson's bootable installation USB stick employs a very minimal version of Linux and made purely for the purpose of installing the Jetson BSP (*Jetson Linux*) on Jetson AGX Thor Developer Kit.

Unlike Canonical's official Ubuntu installation USB, it does NOT offer the capability of booting into full Jetson BSP for the purpose of test drive, thus **it is not "Live USB" stick** in that sense.

What you'll need

- A laptop or PC (Windows, Mac, or Linux) with at least 25GB of storage space
- A USB flash drive (16GB or larger)

Overall Flow



> We will show the alternative way to install the BSP (*Jetson Linux*) in the BSP Setup page.

Steps

1) Download ISO image

First, you need to download the Jetson BSP installation media ("Jetson ISO") image file from NVIDIA's website.

Direct Download Link:

https://developer.nvidia.com/downloads/embedded/L4T/r39_Release_v2.0/iso/jetsoninstaller-r39.2.0-2026-06-01-23-53-13-arm64.iso

Alternatively, you can go to JetPack Download Page, find the latest JetPack version for Jetson AGX Thor Developer Kit, and download the Jetson ISO image file on to your laptop or PC.

2) Create installation USB

To install Jetson BSP (Jetson Linux) on your Jetson AGX Thor Developer Kit, we first need to create an installation media by writing the downloaded ISO image to a USB stick.

> **Caution**
>
> You cannot just simply copy the ISO image file to the USB stick on your OS (e.g. Windows Explorer, or Mac Finder). You need to create a bootable USB stick using some special software.

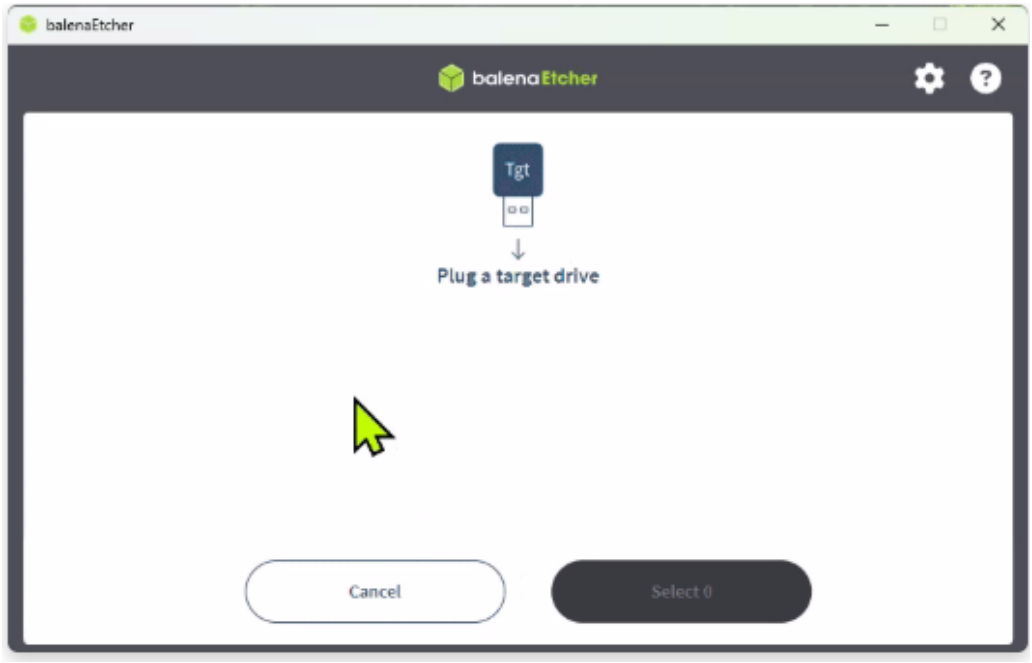
t

his guide, we will use Balena Etcher to create a bootable USB stick.

> **If you don't have Etcher installed / downloaded yet** ›

Once you start Etcher, select the downloaded ISO image file, and select the USB stick you want to write the ISO image to.

https://docs.nvidia.com/jetson/agx-thor-devkit/user-guide/latest/_images/jetson-iso_etcher-flash-start.gif



3) Two setup styles

You can use Jetson AGX Thor NV500-DY in two styles:

1. **Monitor-attached:** You use Jetson as standard computer, with a monitor, keyboard and mouse.
2. **Headless**, so you use your laptop or a PC to remotely access Jetson, and Jetson will be used as a server.

You can set up Jetson using Jetson BSP installation USB stick in both styles.

4) Power-on Jetson

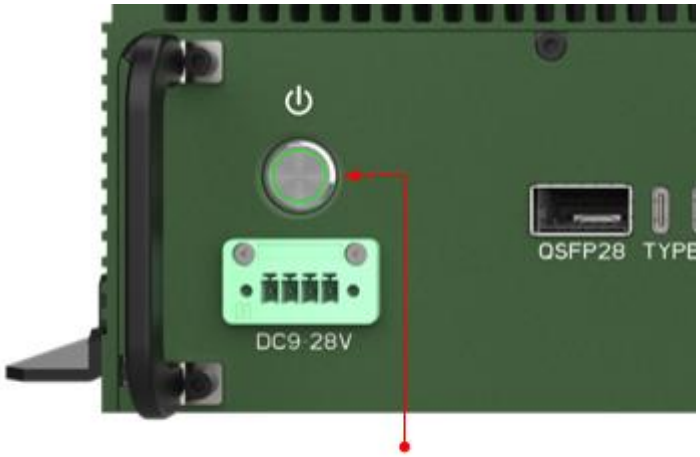
> **Caution**
>
> It's strongly recommended to use the bundled USB-C power supply for stable operation.

n

itor-attached:

1. Attach a display via HDMI or DisplayPort.
2. Connect a USB keyboard and mouse.

3. Attach the bootable installation USB stick (that you created using Etcher on the PC) either on the USB Type-A port or on the USB-C port of Jetson AGX Thor NV500-DY.
4. Connect the power supply plug into one of the two USB-C ports.
5. Boot Jetson by pressing the power button (11 ⏻ the left button in the picture below).



Hint

Check the output on the display and carry out the following steps on the monitor using the keyboard (and the mouse) attached to the Jetson.

5) Boot from the USB stick and install the BSP on NVMe

Caution

If you are repeating this process with a Jetson ISO earlier than JetPack 7.2 (perhaps because you found something went wrong during the installation and want to start over):

➔ Please check the ⚠ Reinstall with USB (Previously Used Device) section.

Tip

Not sure if your UEFI firmware is compatible? Check the UEFI Firmware & Jetson ISO Compatibility page for the compatibility matrix between UEFI firmware versions and Jetson ISO versions.

UEFI Boot Order Check:

Note

Jetson AGX Thor Developer Kit should be **pre-configured at the factory to boot from any USB stick if attached** (over the default NVMe SSD drive). So you don't need to check or manually change the boot order in most cases.

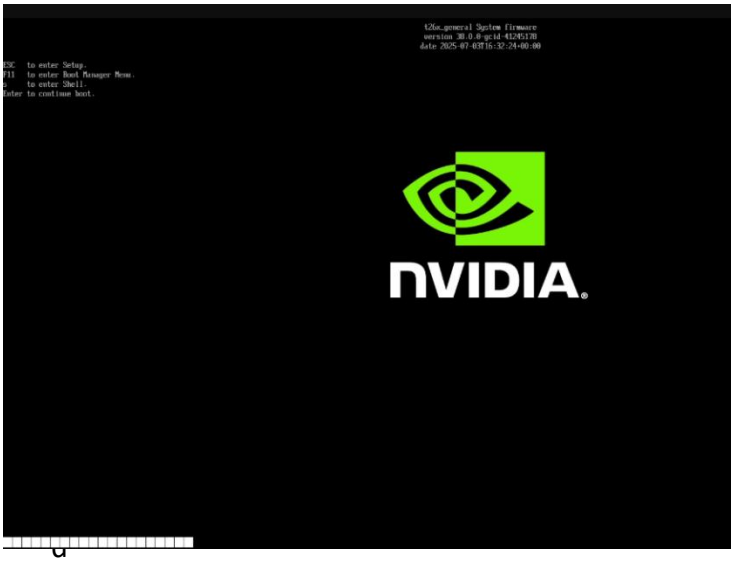
Only if you are unsure (and need to change the boot order), click the section below to learn how to check the boot order and change the boot order if needed.

If need to check the boot order ›

from the installation USB stick:

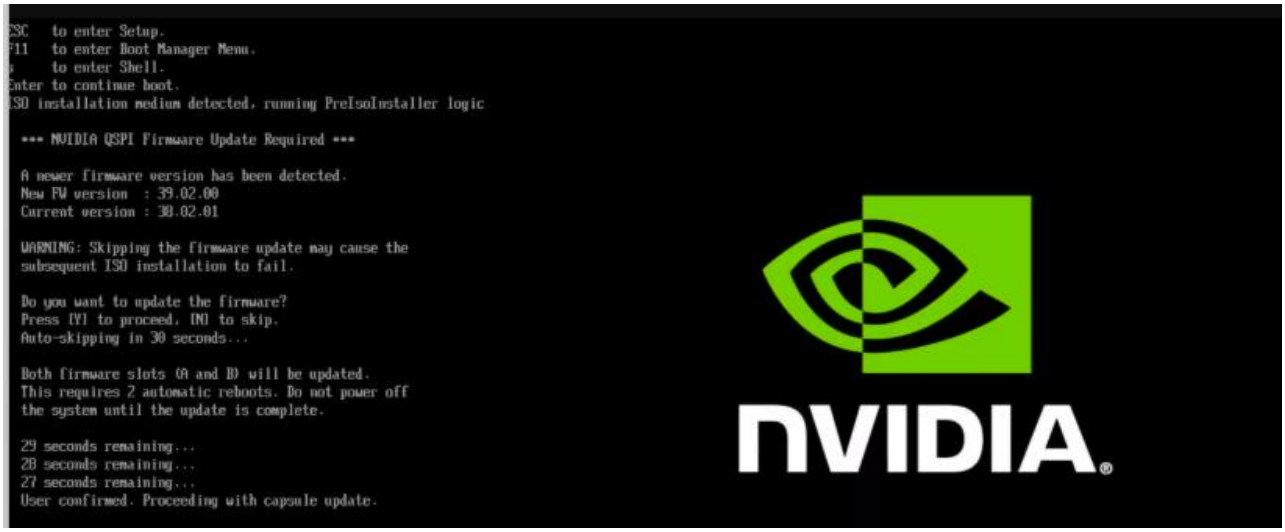
1. When your Jetson powers on, it will first show the pre-boot options. After a short timeout or if you press the Enter key, the system will begin booting from the USB stick.

> Jetson AGX Thor Developer Kit is shipped with factory-flashed UEFI firmware, version `38.0.0-gcid-41245178`

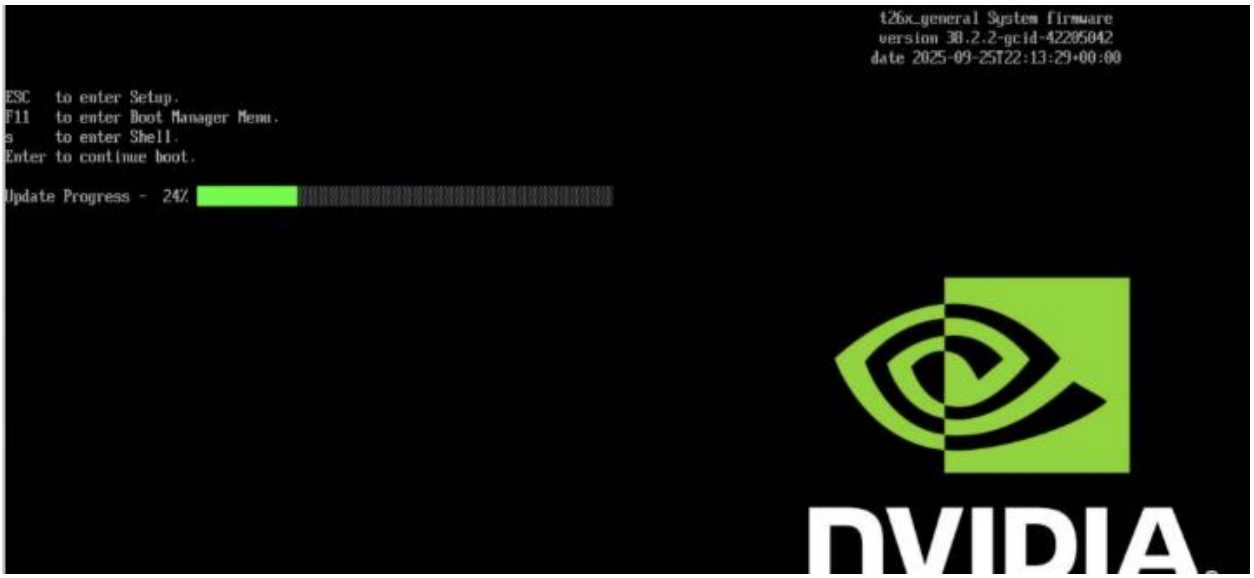


to confirm a QSPI capsule update, press Y before continuing to the ISO installation.

> **Important**
>
> This prompt can appear when the device firmware is older than the firmware required by the Jetson ISO. The capsule update is required for compatibility with the Jetson ISO release.
>
> First confirm the prompt by pressing `Y`, then wait for the capsule update to complete. After the capsule update finishes, the Jetson ISO installation continues.
>
> Do not skip this prompt. If you miss the prompt or are unsure whether the capsule update completed, restart the installation and confirm the capsule update when prompted.



the QSPI capsule update to complete. A progress bar is displayed while the update is in progress. The update runs twice.



see the Jetson BSP installation GRUB menu as shown below.

c

t a suitable storage option by hitting " ↑ " (up) and " ↓ " (down) keys, and press Enter key.

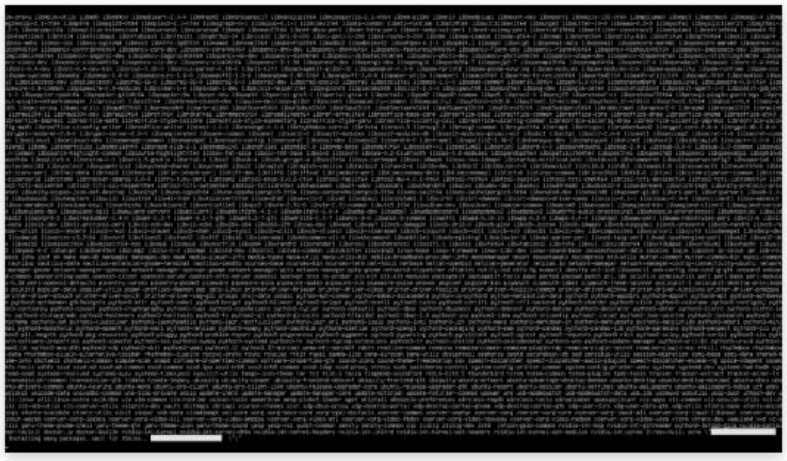
`Install on NVMe` is recommended.

s

tallation process starts.

Note

During the installation process, you will white texts scrolling on the screen. You may see screen stuck with following screen for a while.



It will take little less than 10 minutes to complete the installation.

7. Reboot after BSP installation with Jetson ISO

After about 10 minutes, the installation process will finish and your Jetson will reboot.

6) Boot BSP from NVMe for the first time:

Remove the installation USB stick

Important

Remove the bootable installation USB stick now , otherwise it may boot from the USB stick again instead of the installed BSP on NVMe.

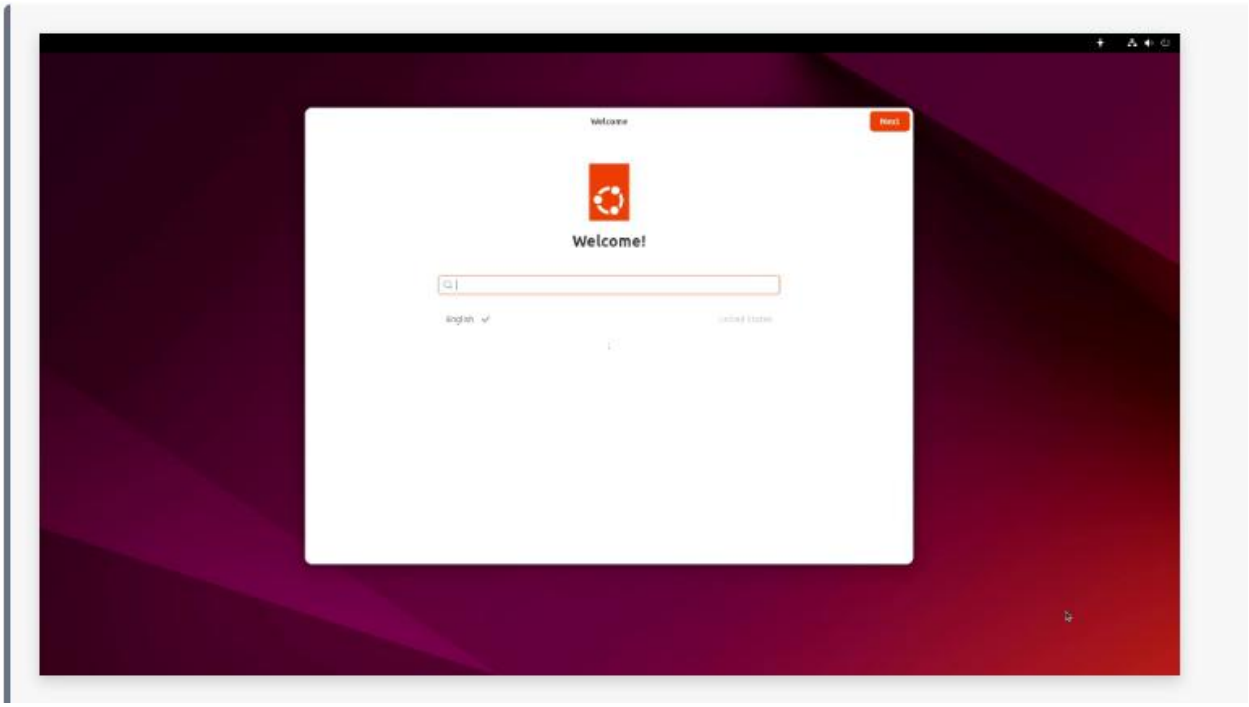
You can remove the USB stick and reset or power off and on the dev kit. For more details, see UEFI Boot Order Check.

Initial software setup

After removing the USB sitck now you can start the initial Ubuntu setup process (oem-config) to create the default user account and set up other things. Once done, you can start using Jetson with Jetson BSP fully set up.

Monitor-attached | Headless

1) You will see the GUI oem-config screen on your attached monitor.

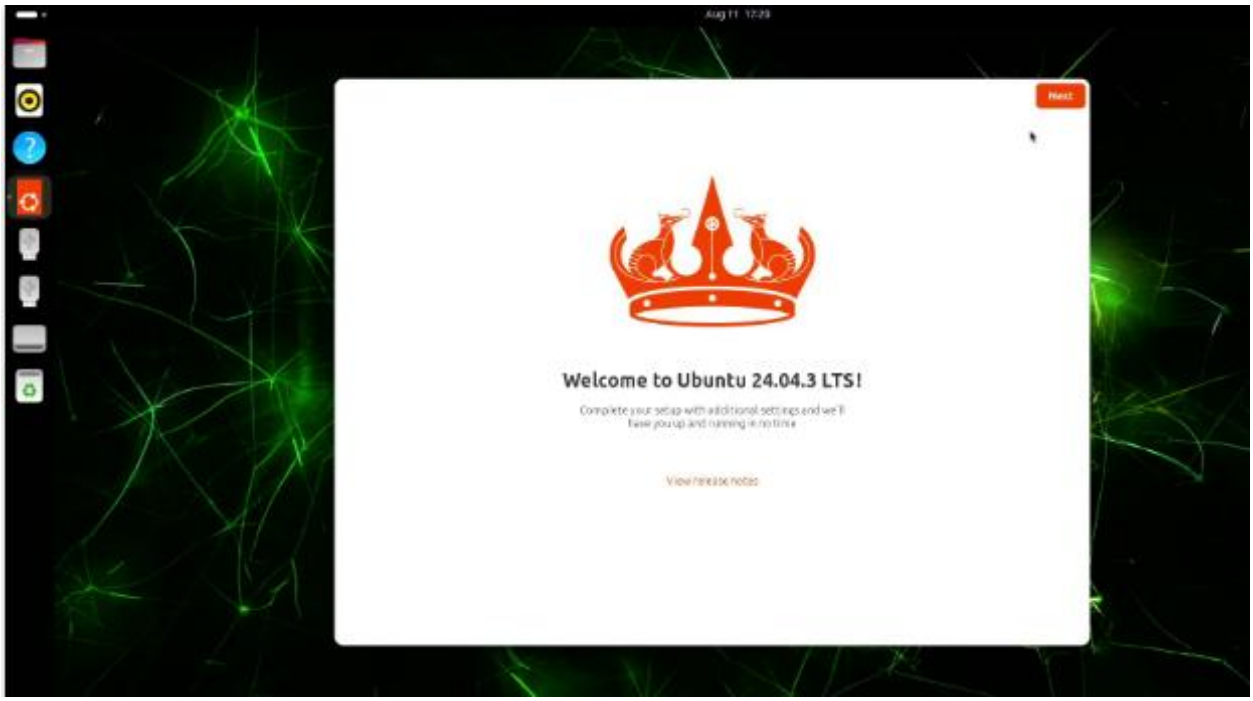


2) On the "Welcome" screen, click Next button to proceed to the next step.
3) On the "Typing" screen, select English (US) or appropriate keyboard layout for your keyboard attached to Jetson AGX Thor Developer Kit, and click Next button to proceed to the next step.
4) On the next screen, read through the "NVIDIA Driver License Agreement" and click Accept button to proceed to the next step.
5) On the "Network" screen, select your Wi-Fi access point to connect to the internet, or click Skip button to skip the network setup and proceed to the next step.
6) On the "Time Zone" screen, select your time zone and click Next button to proceed to the next step.
7) On the "About You" screen, enter your username and other information, and click Next button to proceed to the next step.
8) On the "Password" screen, enter your password, and click Next button to proceed to the next step.
9) On the "Setup Complete" screen, click Start Using Ubuntu button to finish the oem-config process.

Your Next Step

Congratulation!

You can now start the development on your Jetson AGX Thor Developer Kit.



Check JetPack SDK Setup page for installing CUDA and other JetPack component software.

> **Tip**
>
> Already have BSP installed and want to upgrade to a newer L4T revision? See the BSP Upgrade section for in-place upgrade instructions.

⚠ Reinstall with USB (Previously Used Device)

Starting with JetPack 7.2, you can reinstall using the Jetson ISO without manually changing UEFI settings.

If you are using a Jetson ISO from a release earlier than JetPack 7.2 on a Jetson AGX Thor Developer Kit that has been used or set up before, you might need to perform a special operation in the UEFI setup menu.

Please refer to the Re-enable USB stick installation page for the workaround.

BSP Install Troubleshoot

I don't have video out on the monitor (connected through KVM)

It is known that some KVM switch and KVM device cannot handle the video output from the Jetson AGX Thor Developer Kit well. Please try to connect the monitor directly to the Jetson AGX Thor Developer Kit.

QSPI capsule update prompt appears

If the installer asks you to confirm a QSPI capsule update, press Y.

This prompt appears before ISO installation when the device firmware is older than the firmware bundled with the Jetson ISO. The update is required for compatibility with the Jetson ISO release.

First confirm the prompt by pressing Y, then wait for the capsule update to complete. After the capsule update finishes, the ISO installation continues.

If you miss the prompt or are unsure whether the update completed, restart the ISO installation and confirm the capsule update when prompted.

Screen goes black once installation starts

If you are using a Jetson installation USB stick earlier than JetPack 7.2 on a Jetson AGX Thor Developer Kit that has been used or set up before, you might need to enable the SOC Display Hand-Off mode in the UEFI setup menu.

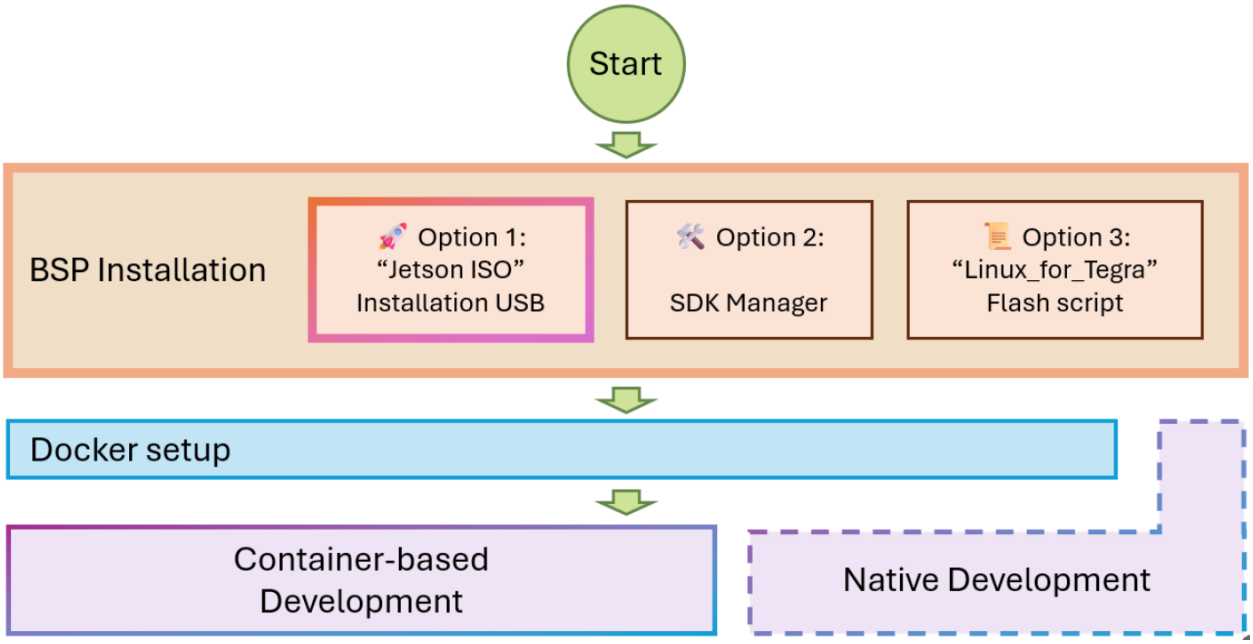
For JetPack 7.2 OOBE recovery, if Ubuntu hangs while starting the OOBE service after using the efifb display hand-off method, use the tested simplefb path instead.

See Re-enable USB stick installation for the detailed UEFI steps.

BSP Setup

BSP Installation

Installation Options



Comparison

	Jetson ISO	SDK Manager	Flash script
Short Summary	USB installation disk to flash BSP (plus alpha)	GUI software to flash BSP and install JetPack packages	Ubuntu script to flash BSP
Ubuntu Host PC	❌ Not required	✅ Required	✅ Required
Flash time	Less than 15 min	⏳ About 30 min	⏳ About 30 min
Install BSP?	✅ Yes	✅ Yes	✅ Yes
Install Docker?	✅ Yes	💡 Yes, when chosen in 2nd stage	❌ No
Install JetPack components	❌ No	💡 Yes, when chosen in 2nd stage	❌ No
Who is this for?	Everybody including beginner	Who has an Ubuntu PC	Product Developers

Option 1. 🚀 "Jetson ISO" Installation USB

Recommended for everybody.

This method is what's illustrated in the Quick Start Guide.

Option 2. 🛠 NVIDIA SDK Manager

See Jetson section of SDK Manager documentation.

https://docs.nvidia.com/sdk-manager/install-with-sdkm-jetson/index.html

Option 3. 📜 "Linux_for_Tegra" flash script

See Jetson Linux Developer Guide.

https://docs.nvidia.com/jetson/archives/r39.2/DeveloperGuide/IN/QuickStart.html

https://docs.nvidia.com/jetson/archives/r39.2/DeveloperGuide/SD/FlashingSupportJetsonThor.html

Open the target HW image folder.

BSP Upgrade

This section covers how to upgrade between supported L4T minor revisions using APT.

> **Important**
>
> Upgrading to **r39.2** is not supported through the APT steps below. To move to r39.2, use one of the installation methods described in the Quick Start Guide or the BSP Installation section: Jetson ISO, NVIDIA SDK Manager, or the `Linux_for_Tegra` flash script.

> **See also**
>
> For more details on the update mechanism, see the official Jetson Linux Developer Guide:
>
> Updating to a New Minor Release

Upgrading from r38.2.x to r38.4.x

Follow these steps to upgrade your Jetson Thor from L4T r38.2.x to r38.4.x.

1. Check the Current L4T Revision

First, verify your current L4T version:

```
cat /etc/nv_tegra_release
```

You should see output indicating R38 (release), REVISION: 2.x.

2. Backup and Update the APT Source List

Before modifying the source list, create a backup:

```
sudo sed -i 's/r38\.2/r38.4/g' /etc/apt/sources.list.d/nvidia-l4t-apt-source.list
```

sudo cp /etc/apt/sources.list.d/nvidia-l4t-apt-source.list /etc/apt/sources.list.d/nvidia-l4t-apt-source.list.bak

Update the source list to point to r38.4:

```
sudo cp /etc/apt/sources.list.d/nvidia-l4t-apt-source.list /etc/apt/sources.list.d/nvidia-l
```

Verify the change was applied correctly:

```
tail -n5 /etc/apt/sources.list.d/nvidia-l4t-apt-source.list
```

3. Perform the Upgrade

Update the package list and upgrade:

```
sudo apt update && sudo apt upgrade -y
```

4. Reboot

Reboot to apply the changes:

```
sudo reboot
```

5. Verify the Upgrade

After reboot, confirm the upgrade was successful:

```
cat /etc/nv_tegra_release
```

You should now see R38 (release), REVISION: 4.x.

> **Tip**
>
> If something goes wrong during the upgrade, you can restore the original source list from the backup:
>
> ```
> sudo cp /etc/apt/sources.list.d/nvidia-l4t-apt-source.list.bak /etc/apt/sources.lis
> ```

Docket Setup

Docker brings reproducible, containerized workflows to Jetson, giving you seamless access to the GPU without sacrificing speed or flexibility.

Do you need Docker?

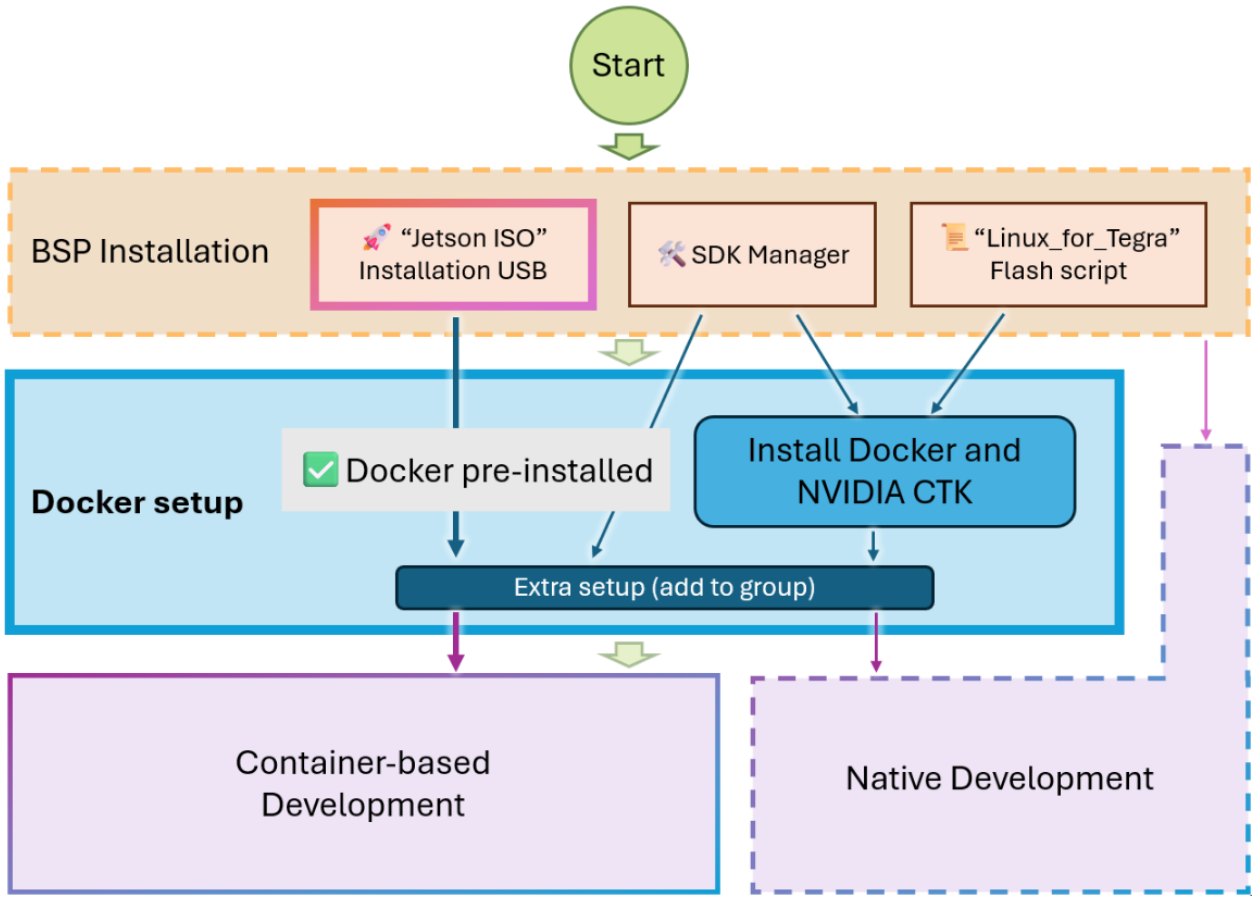
If you fall into the following categories, you want/need to install Docker:

You want to run NGC and other containers that's pre-built for Jetson.
e.g. You want to use jetson-containers.

You want to manage multiple software stack configurations for your development (Container-based Development).

Docker Setup Flow

To summarize the options and steps, the following diagram shows the flow of the Docker setup process.



Step 1: Install Docker and NVIDIA Container Toolkit

This step is only needed if you flashed BSP using "Linux_for_Tegra" flash script or flashed BSP using SDK Manager

Step 2: Rest of the Docker Setup

Add the default runtime to the Docker daemon configuration file, so that we don't need to specify --runtime nvidia every time we run a container.

```
sudo apt install -y jq
sudo jq '. + {"default-runtime": "nvidia"}' /etc/docker/daemon.json | \
    sudo tee /etc/docker/daemon.json.tmp && \
    sudo mv /etc/docker/daemon.json.tmp /etc/docker/daemon.json
```

Let's make sure you have the following in your /etc/docker/daemon.json file.

```
$ cat /etc/docker/daemon.json
{
  "runtimes": {
    "nvidia": {
      "args": [],
      "path": "nvidia-container-runtime"
    }
  },
  "default-runtime": "nvidia"
}
```

It's also recommended to add your username ($USER) to the docker group to avoid using sudo to run Docker commands.

```
sudo usermod -aG docker $USER
newgrp docker
```

Finally, restart the Docker service so it picks up the new default runtime. Without this, Docker will run but containers will not have GPU access.

```
sudo systemctl restart docker
```

After that, you may need to restart your terminal/session to apply the changes.

Docker Setup Test

Example 1: Run PyTorch container

```
docker run --rm -it \
    -v "$PWD":/workspace \
    -w /workspace \
    nvcr.io/nvidia/pytorch:25.08-py3
```

Once in the container, you can test PyTorch with GPU.

```
python3 <<'EOF'
import torch
print("PyTorch version:", torch.__version__)
print("CUDA available:", torch.cuda.is_available())
if torch.cuda.is_available():
    print("GPU name:", torch.cuda.get_device_name(0))
    x = torch.rand(10000, 10000, device="cuda")
    print("Tensor sum:", x.sum().item())
EOF
```

You should see something like this:

```
PyTorch version: 2.8.0a0+34c6371d24.nv25.08
CUDA available: True
GPU name: NVIDIA Thor
Tensor sum: 50001728.0
```

Docker Setup Troubleshooting

Error: permission denied

If you see the following error:

```
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/dc
```

permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.51/containers/json": dial unix /var/run/docker.sock: connect: permission denied

It's most likely that you did not or failed to add your username ($USER) to the docker group.

Solution 1: Add your username to the docker group

Execute the following commands to add your username to the docker group, and then restart your terminal/session to apply the changes,

```
sudo usermod -aG docker $USER
newgrp docker
```

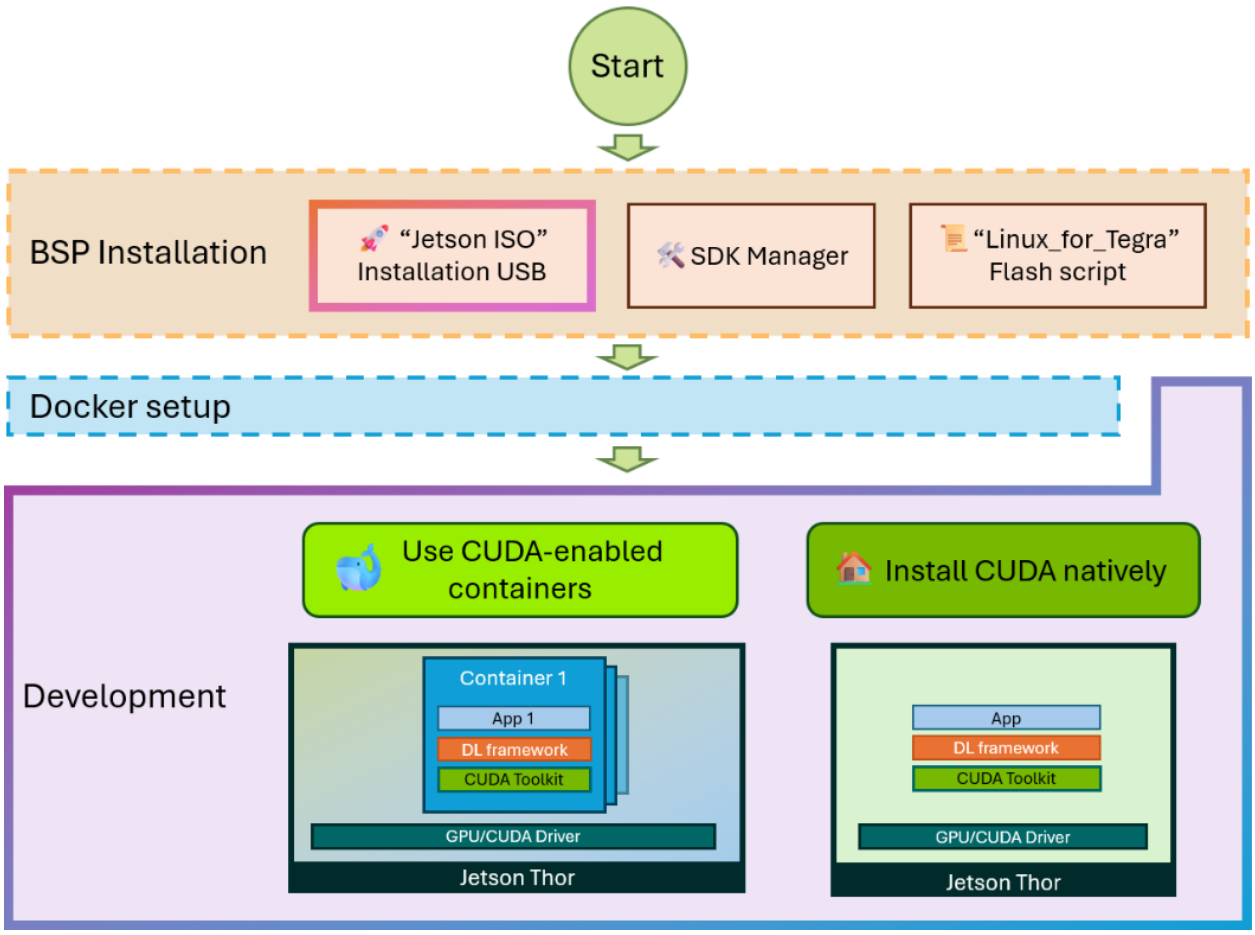
Solution 2: Use sudo

Or, for now, you execute docker command with sudo.

```
sudo docker info
```

CUDA Setup

CUDA Setup Options



Comparison

While experiences vary and some points are open to debate, the table below provides a practical side-by-side comparison of container-based and native development. It's not meant as an absolute judgment, but as guidance to help weigh trade-offs when choosing an environment for your development and deployment.

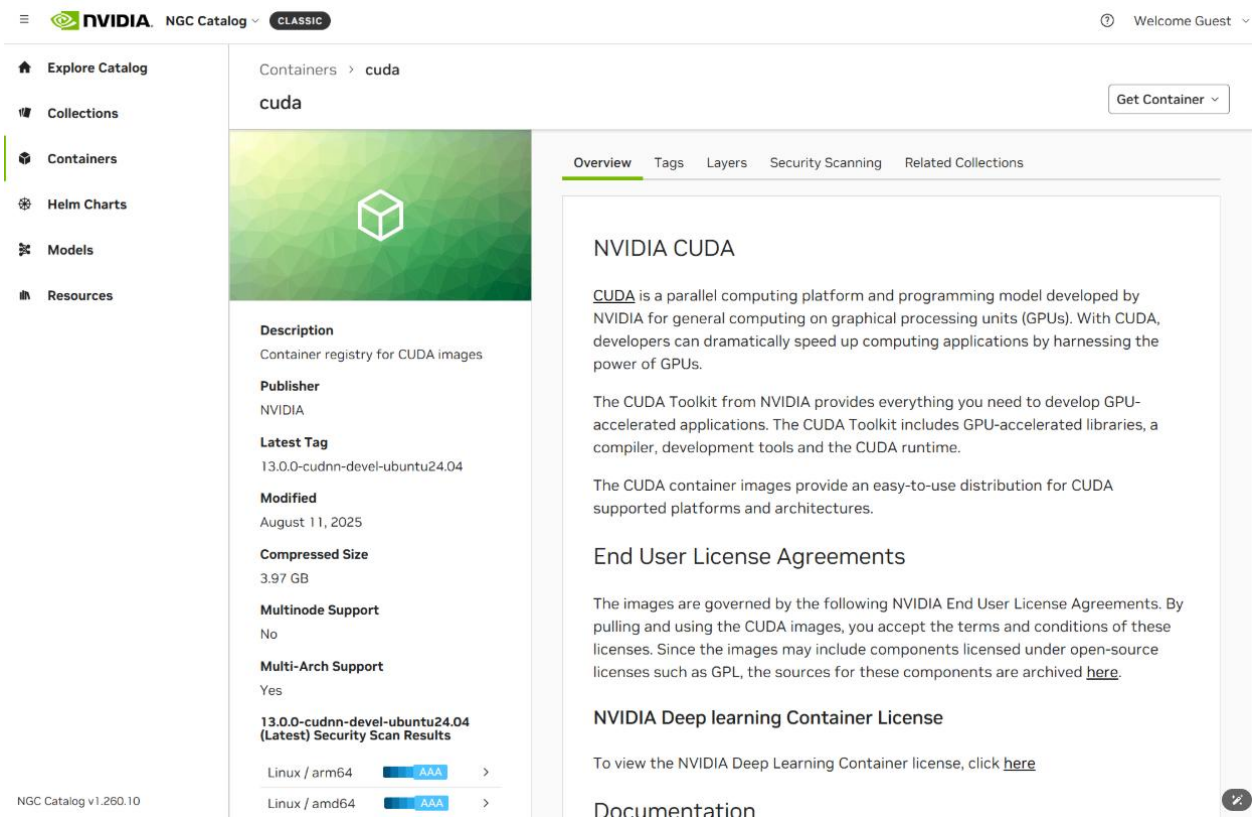
Comparison Table for Container-based vs. Native

Ways to use CUDA-enabled containers

If you have Docker already setup, you can immediately pull (download) and run CUDA-enabled containers, without installing anything on your host environment.

Run NGC container

NGC (NVIDIA GPU Cloud) is the hub for GPU-optimized software for deep learning and more, and a registry for NVIDIA-provided containers.



Under the "Tags" section, you can see a list of tags for the hosted container.

Notice that it shows "**2 Architectures**". This means that the container image is available for both x86_64 and **arm64** architectures.

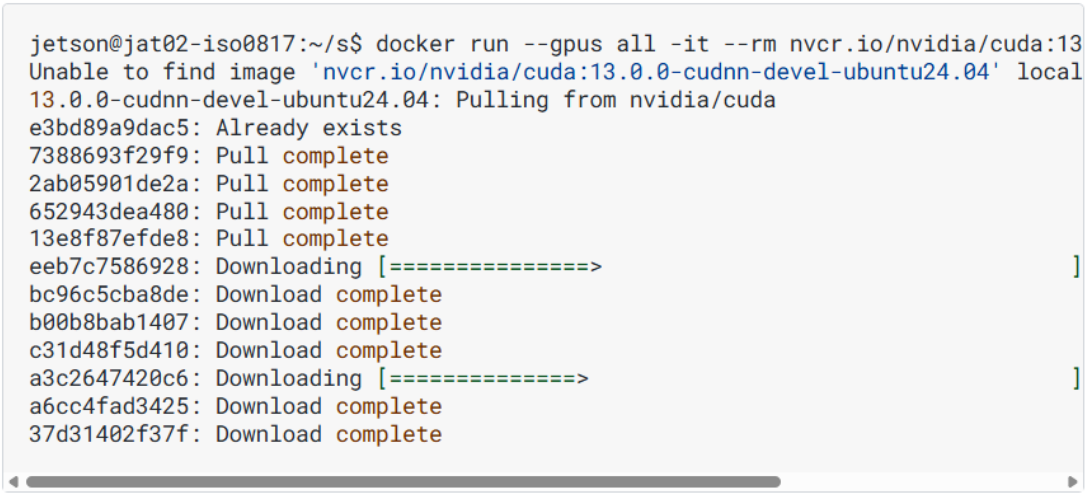
You can click on icon to copy the container image path with that tag into your clipboard.

Example 1: Run CUDA container

You can run the docker run command with the copied container image path.

```
docker run -it --rm nvcr.io/nvidia/cuda:13.0.0-devel-ubuntu24.04
```

It will start pulling (downloading) the container image from NGC, and



jetson@jat02-iso0817:~/s$ docker run --gpus all -it --rm
nvcr.io/nvidia/cuda:13.0.0-cudnn-devel-ubuntu24.04
Unable to find image 'nvcr.io/nvidia/cuda:13.0.0-cudnn-devel-ubuntu24.04' locally
13.0.0-cudnn-devel-ubuntu24.04: Pulling from nvidia/cuda
e3bd89a9dac5: Already exists
7388693f29f9: Pull complete
2ab05901de2a: Pull complete
652943dea480: Pull complete
13e8f87efde8: Pull complete
eeb7c7586928: Downloading [==========>]
489.2MB/1.594GB
bc96c5cba8de: Download complete
b00b8bab1407: Download complete
c31d48f5d410: Download complete
a3c2647420c6: Downloading [==========>]
630.2MB/2.113GB
a6cc4fad3425: Download complete
37d31402f37f: Download complete

Once download and extraction is complete, or if you have already pulled the container image, you will see something like this:

jetson@jat02-iso0817:~/$ docker run -it --rm nvcr.io/nvidia/cuda:13.0.0-devel-ubuntu24.04

==========
== CUDA ==
==========

CUDA Version 13.0.0

Container image Copyright (c) 2016-2023, NVIDIA CORPORATION & AFFILIATES. All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

A copy of this license is made available in this container at /NGC-DL-CONTAINER-LICENSE for your convenience.

WARNING: The NVIDIA Driver was not detected. GPU functionality will not be available.
Use the NVIDIA Container Toolkit to start this container with GPU support; see
https://docs.nvidia.com/datacenter/cloud-native/ .

root@99042c33f4b2:/#

Example 2: Build *cuda-samples* using NGC CUDA container

On your Jetson (Docker host), crate a diretory to be mounted on the container, so that you will not loose your built binaries.

```
cd ~
mkdir -p $HOME/cuda-work && cd $HOME/cuda-work
docker run --rm -it \
    -v "$PWD":/workspace \
    -w /workspace \
    nvcr.io/nvidia/cuda:13.0.0-devel-ubuntu24.04
```

Once in the container, you can build cuda-samples using the following command.

```
apt update && apt install -y --no-install-recommends git make cmake
git clone --depth=1 --branch v13.0 https://github.com/NVIDIA/cuda-samples.git
cd cuda-samples/Samples/1_Utilities/deviceQuery
cmake . -DGPU_TARGETS=all -DCMAKE_BUILD_TYPE=Release
make -j$(nproc)
./deviceQuery
```

You should see something like this:

```
root@84419057c31d:/workspace/cuda-samples/Samples/1_Utilities/deviceQuery# ls
CMakeCache.txt  CMakeFiles  CMakeLists.txt  Makefile  README.md  cmake_install.cmake  devic
root@84419057c31d:/workspace/cuda-samples/Samples/1_Utilities/deviceQuery# ./deviceQuery
./deviceQuery Starting...

  CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA Thor"
  CUDA Driver Version / Runtime Version          13.0 / 13.0
  CUDA Capability Major/Minor version number:    11.0
  Total amount of global memory:                 125772 MBytes (131881811968 bytes)
  (020) Multiprocessors, (128) CUDA Cores/MP:    2560 CUDA Cores
  GPU Max Clock rate:                            1049 MHz (1.05 GHz)
  Memory Clock rate:                             0 Mhz
  Memory Bus Width:                              0-bit
  L2 Cache Size:                                 33554432 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        233472 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously)

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 13.0, CUDA Runtime Version = 13.0,
Result = PASS
```

root@84419057c31d:/workspace/cuda-samples/Samples/1_Utilities/deviceQuery# ls

CMakeCache.txt CMakeFiles CMakeLists.txt Makefile README.md cmake_install.cmake deviceQuery deviceQuery.cpp
root@84419057c31d:/workspace/cuda-samples/Samples/1_Utilities/deviceQuery# ./deviceQuery
./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA Thor"
CUDA Driver Version / Runtime Version 13.0 / 13.0
CUDA Capability Major/Minor version number: 11.0
Total amount of global memory: 125772 MBytes (131881811968 bytes)
(020) Multiprocessors, (128) CUDA Cores/MP: 2560 CUDA Cores
GPU Max Clock rate: 1049 MHz (1.05 GHz)
Memory Clock rate: 0 Mhz
Memory Bus Width: 0-bit
L2 Cache Size: 33554432 bytes
Maximum Texture Dimension Size (x,y,z) 1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 49152 bytes
Total shared memory per multiprocessor: 233472 bytes
Total number of registers available per block: 65536
Warp size: 32
Maximum number of threads per multiprocessor: 1536
Maximum number of threads per block: 1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
Maximum memory pitch: 2147483647 bytes
Texture alignment: 512 bytes
Concurrent copy and kernel execution: Yes with 1 copy engine(s)
Run time limit on kernels: Yes
Integrated GPU sharing Host Memory: Yes
Support host page-locked memory mapping: Yes
Alignment requirement for Surfaces: Yes

Device has ECC support: Disabled
Device supports Unified Addressing (UVA): Yes
Device supports Managed Memory: Yes
Device supports Compute Preemption: Yes
Supports Cooperative Kernel Launch: Yes
Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
Compute Mode:
< Default (multiple host threads can use ::cudaSetDevice (with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 13.0, CUDA Runtime Version = 13.0, NumDevs = 1
Result = PASS

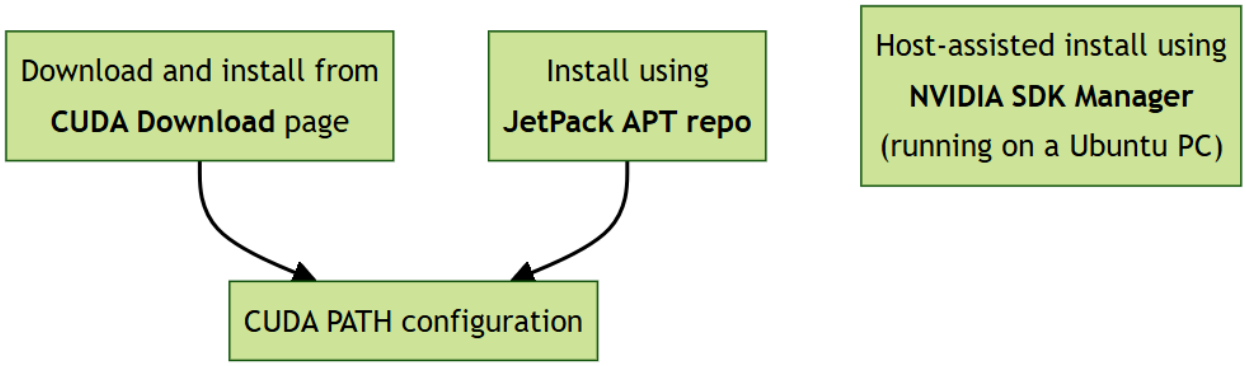
Use jetson-containers

Follow Getting Started section of jetson-containers repository to git-clone and install dependencies.

Then, you can easily run various pre-built containers using jetson-containers CLI.

```
jetson-containers run $(autotag stable-diffusion-webui)
```

Ways to natively install CUDA Toolkit



Option 1: CUDA Download Page

You can go to NVIDIA's CUDA download page and download the Debian package for CUDA Toolkit, and install it on your Jetson.

Jump to CUDA download page and select:

> **Linux → arm64-sbsa → Native → Ubuntu → 24.04 → deb (local)**

or, click this direct link.

Then, execute the commands given in the installation instructions.

CUDA Toolkit 13.0 Downloads

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the CUDA EULA.

Operating System	Linux, Windows
Architecture	x86_64, arm64-sbsa
Compilation	Native, Cross
Distribution	Amazon-Linux, Azure-Linux, KylinOS, RHEL, SLES, Ubuntu
Version	22.04, 24.04
Installer Type	deb (local), deb (network), runfile (local)

Download Installer for Linux Ubuntu 24.04 arm64-sbsa

The base installer is available for download below.

> CUDA Toolkit Installer

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/sbsa/cuda-ubuntu2404.pin
$ sudo mv cuda-ubuntu2404.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ wget https://developer.download.nvidia.com/compute/cuda/13.0.0/local_installers/cuda-repo-ubuntu2404-13-0-local_13.0.0-580.65.06-1_arm64.deb
$ sudo dpkg -i cuda-repo-ubuntu2404-13-0-local_13.0.0-580.65.06-1_arm64.deb
$ sudo cp /var/cuda-repo-ubuntu2404-13-0-local/cuda-*-keyring.gpg /usr/share/keyrings/
$ sudo apt-get update
$ sudo apt-get -y install cuda-toolkit-13-0
```

Additional installation options are detailed here.

Download Installer for Linux Ubuntu 24.04 arm64-sbsa

The base installer is available for download below.

> CUDA Toolkit Installer

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/sbsa/cuda-ubuntu2404.pin
$ sudo mv cuda-ubuntu2404.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ wget https://developer.download.nvidia.com/compute/cuda/13.0.0/local_installers/cuda-repo-ubuntu2404-13-0-local_13.0.0-580.65.06-1_arm64.deb
$ sudo dpkg -i cuda-repo-ubuntu2404-13-0-local_13.0.0-580.65.06-1_arm64.deb
$ sudo cp /var/cuda-repo-ubuntu2404-13-0-local/cuda-*-keyring.gpg /usr/share/keyrings/
$ sudo apt-get update
$ sudo apt-get -y install cuda-toolkit-13-0
```

Additional installation options are detailed here.

> Driver Installer

NVIDIA Driver Instructions (choose one option)

To install the open kernel module flavor:

```
$ sudo apt-get install -y nvidia-open
```

DO NOT DO THIS

To install the proprietary kernel module flavor:

```
$ sudo apt-get install -y cuda-drivers
```

To switch between NVIDIA Driver kernel module flavors see here.

Option 2: JetPack APT repo

Install the whole JetPack SDK Components | Install only the CUDA Toolkit

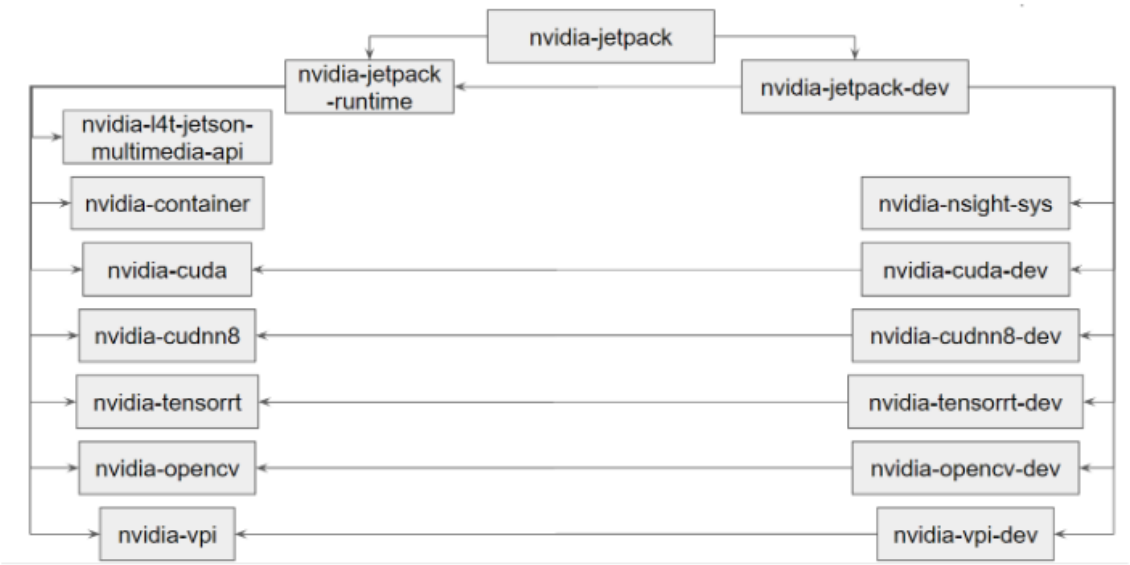
To install the whole JetPack component software/SDK on your Jetson, including CUDA Toolkit for development, you can use the following command:

```
sudo apt update
sudo apt install nvidia-jetpack
```

Note that this will consume **15GB+** of storage space on your Jetson.

Note

nvidia-jetpack is a meta package that will install the following components:



Therefore, you can install specific JetPack components by specifying the sub meta-package, like:

```
sudo apt update
sudo apt install nvidia-cuda-dev
```

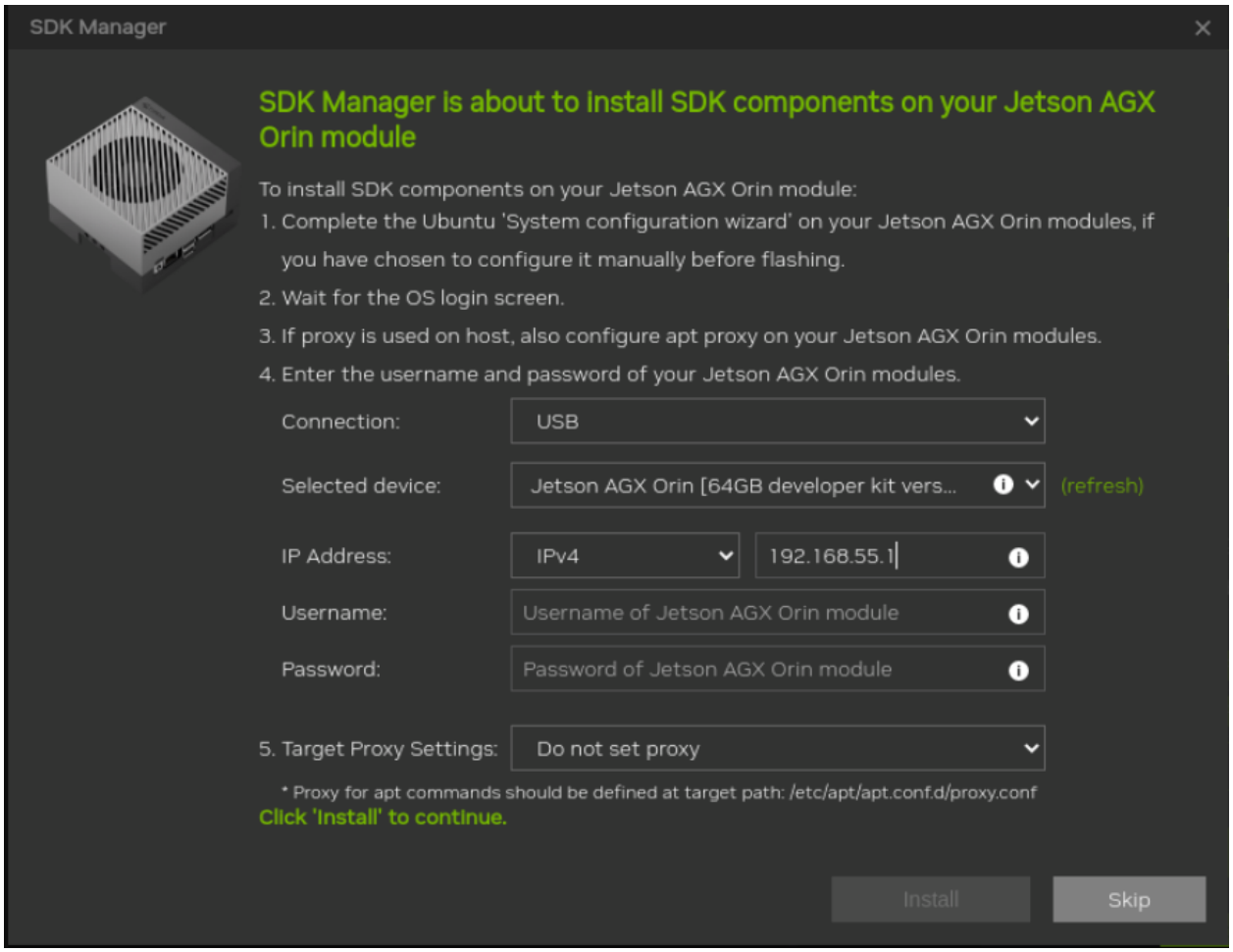
Hint

nvidia-cuda contains only the CUDA runtime libraries, while *nvidia-cuda-dev* contains the CUDA development tools.

Option 3: SDK Manager

If you have a host Ubuntu PC, you can run SDK Manager to install CUDA Toolkit on your Jetson.

Connect your PC and Jetson with a USB cable and follow the instructions after Step 03 - 6 in SDK Manager Documentation's "Install Jetson Software with SDK Manager" page.



Post Install Setup : CUDA PATH configuration

After installing CUDA Toolkit, you need to configure the PATH and LD_LIBRARY_PATH environment variables to use the CUDA Toolkit.

You can do this by adding the following lines to your ~/.bashrc file.

```
echo "export PATH=/usr/local/cuda/bin:$PATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH" >> ~/.bashrc
source ~/.bashrc
```

CUDA Installation Troubleshooting

To Be Added

JetPack SDK Setup

JetPack installation

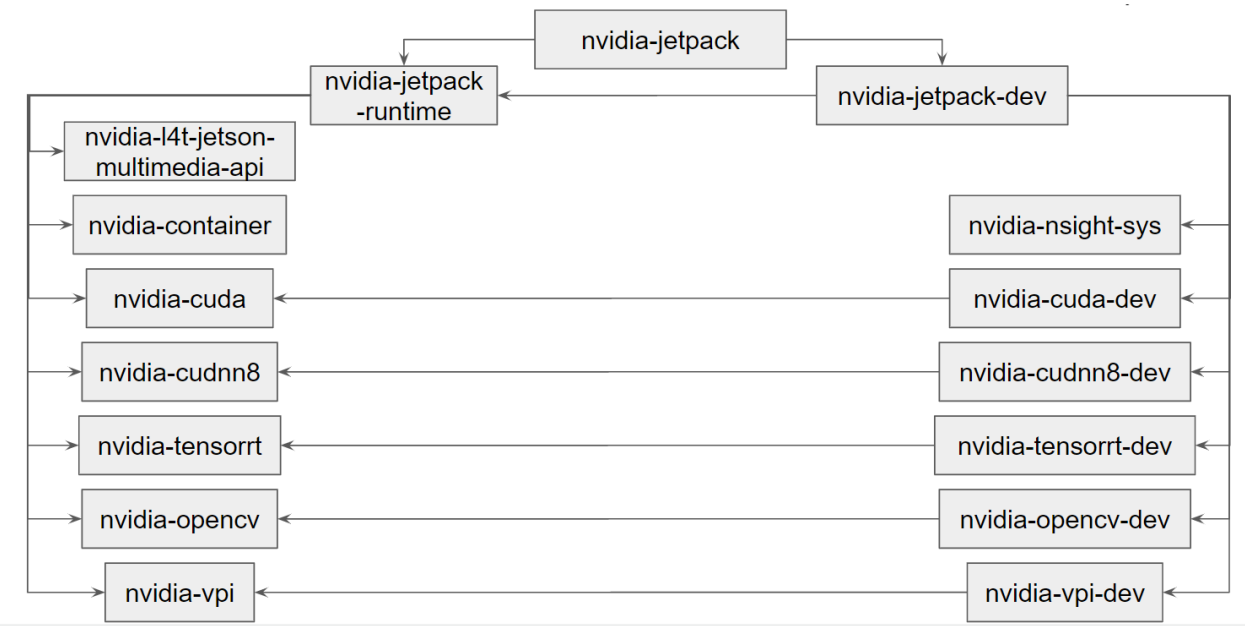
Install the whole JetPack components

To install the whole JetPack component software/SDK on your Jetson, you can use the following command:

```
sudo apt update
sudo apt install nvidia-jetpack
```

Install specific JetPack components

nvidia-jetpack is a meta package that will install the following components:



Therefore, you can install specific JetPack components by specifying the sub meta-package, like:

```
sudo apt update
sudo apt install nvidia-cuda-dev
```

Note

nvidia-cuda contains only the CUDA runtime libraries, while *nvidia-cuda-dev* contains the CUDA development tools.

Install CUDA using JetPack's sub meta-package

As explained above, to natively install just the CUDA Toolkit for your development on your Jetson using JetPack's sub meta-package, you can use the following command:

```
sudo apt update
sudo apt install nvidia-cuda-dev
```

> ⚠ **Caution**
>
> **Please do not install `nvidia-cuda-toolkit` package.**
>
> While `nvidia-cuda-dev` and `nvidia-cuda-toolkit` appear similar, the one ends with `-toolkit` is a package managed under Ubuntu repository, and may not offer the CUDA designed for Jetson.

Install CUDA using Debian Package from CUDA Downloads page

As to installing CUDA Toolkit on your Jetson, there is an alternative path.

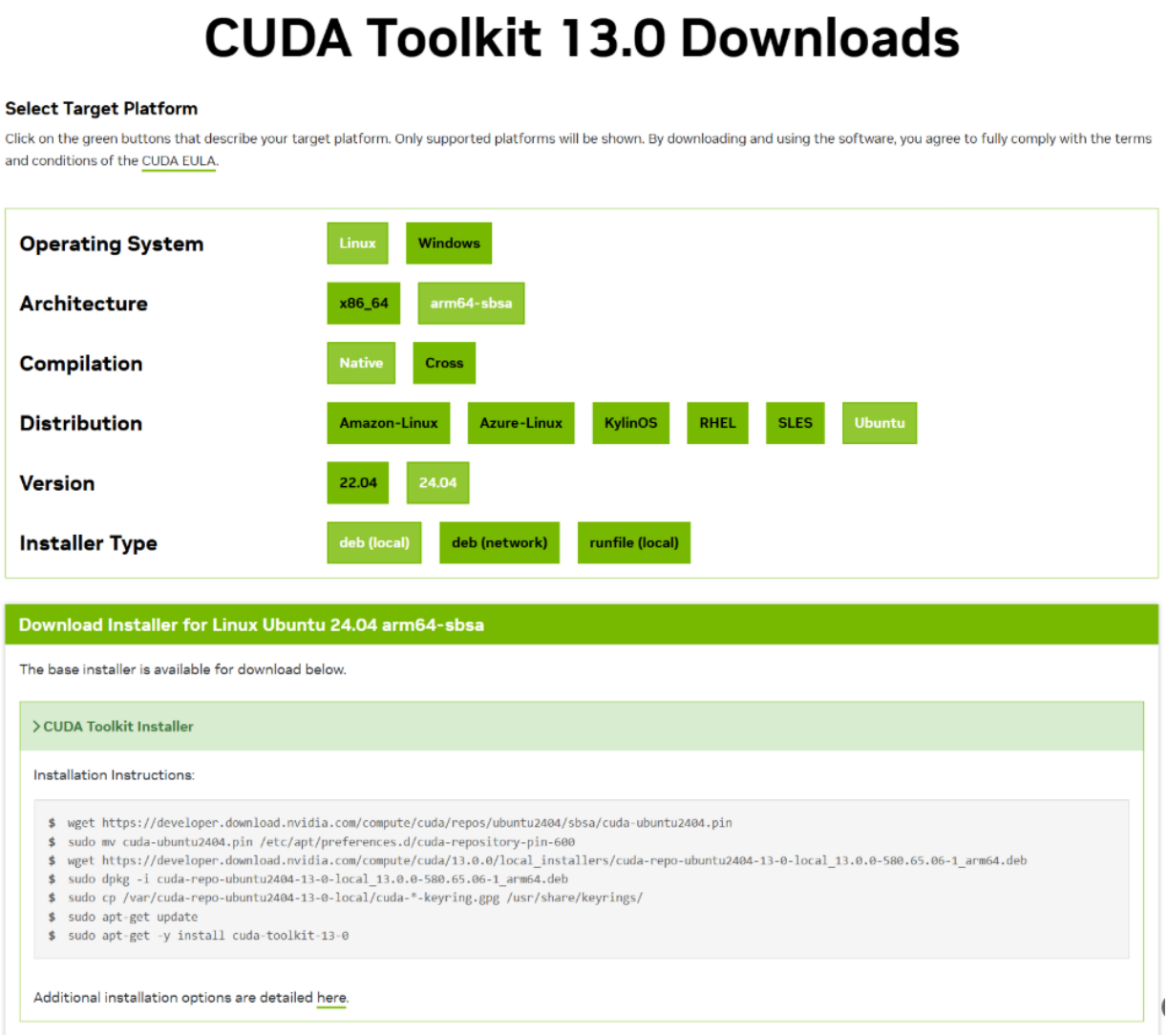
You can go to NVIDIA's CUDA download page the Debian package for CUDA Toolkit, and install it on your Jetson.

- Jump to CUDA download page and select:

  > **Linux → arm64-sbsa → Native → Ubuntu → 24.04 → deb (local)**

- or, click this direct link.

Then, execute the commands given in the installation instructions.



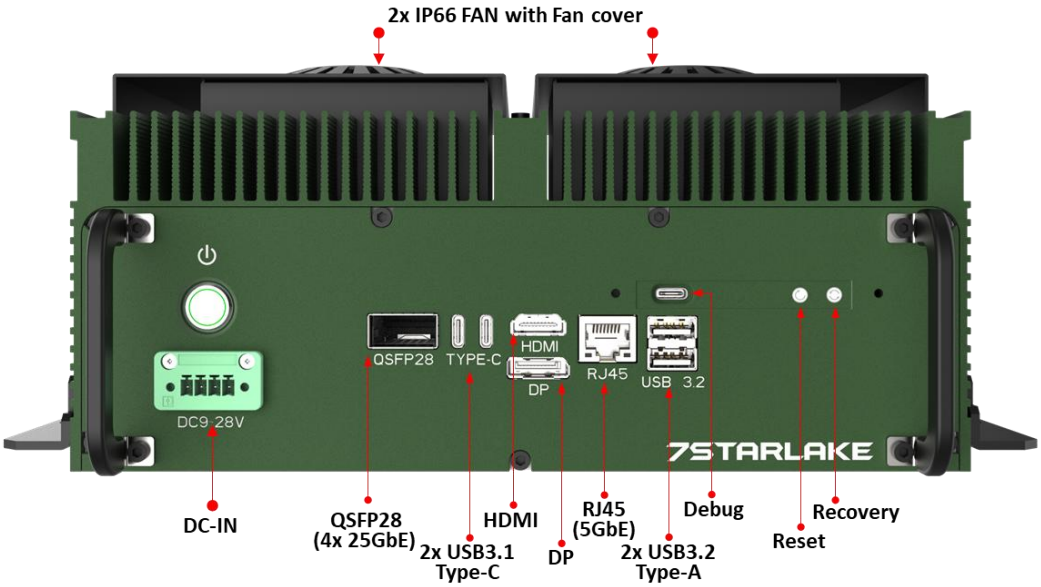
PATH setting after installing CUDA

```
echo "export PATH=/usr/local/cuda/bin:$PATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH" >> ~/.bashrc
source ~/.bashrc
```

Hardware Layout

> **Note**
>
> For the detailed specification of each connector, header and carrier board, please refer to the NVIDIA Jetson AGX Thor Developer Kit Carrier Board Specification.

IO Side Layout



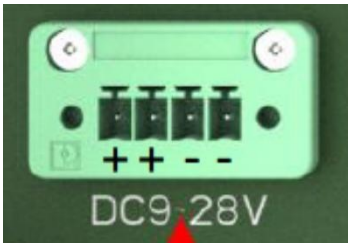
USB Type-A ports (2x)	USB 3.2 Gen 2 (10Gbps)
RJ45 Ethernet port	5Gbps
DisplayPort output	DP
HDMI output	HDMI
USB-Type-C port with Force-Recovery Mode functionality (Left)	• **Force-Recovery Mode functionality** • UFP (USB Device Mode) • DFP (USB 3.2 Gen 1 - 5Gbps) • PD Sink (140W)
USB Type-C port (Right)	• UFP (USB Device Mode) • DFP (USB 3.2 Gen 1 - 5Gbps) • PD Sink (140W)
QSFP28 cage	4x 10GbE(Default) or 25GbE
Phoenix Jack power input	9-28V DC input (Up to 8A)
Debug USB-C port	(Behind the lid cover)

Attention

If you are using an Ubuntu host PC to externally flash Jetson AGX Thor Developer Kit (using either the SDK Manager or the *Linux_for_Tegra* Flash Script), **you need to connect a USB from your PC into USB-C port next to the HDMI connector** (`5a`) and thus connect the USB-C power supply into the other USB-C port (`5b`).

If you don't see `APX` device show up on `lsusb` commands ran on your PC even when you put the Jetson AGX Thor Developer Kit into the Force Recovery Mode (RCM), check whether the USB-C cable connection for flashing is going into the right USB-C port.

Phoenix Power Connector Specification



Button Side Layout

Power Button with LED Back Light
Recovery Button
Reset Button

How to enter Force Recovery Mode

To enter Force Recovery Mode, follow the instructions below:

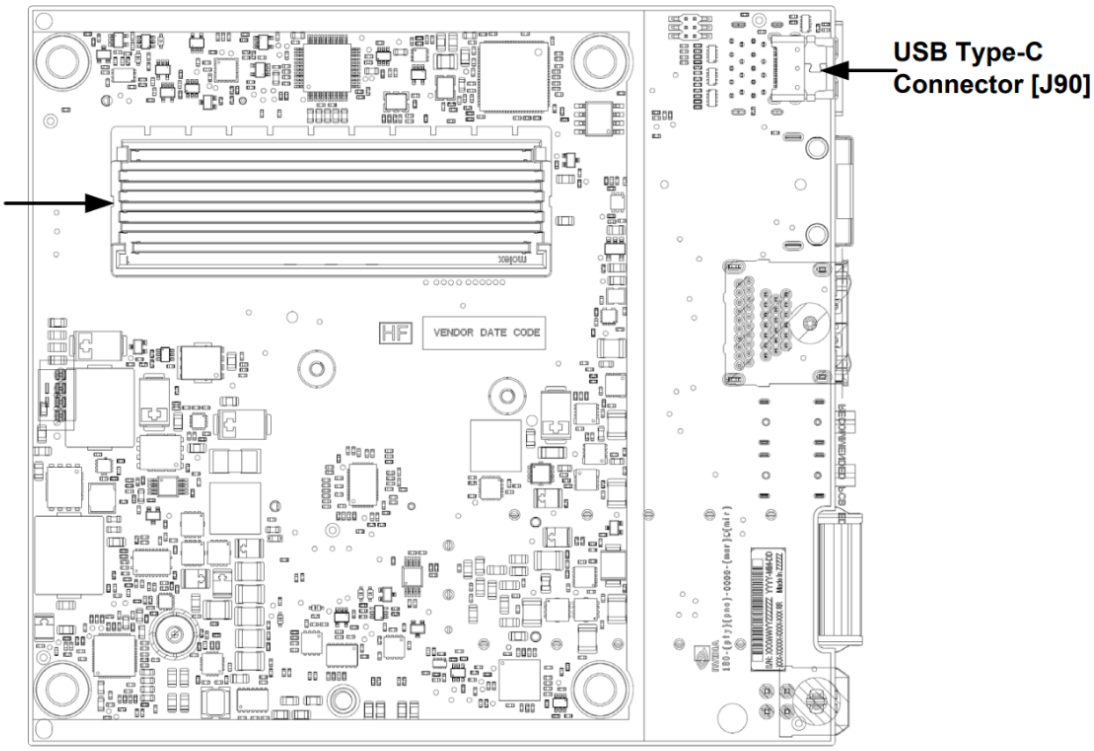
1. Press and hold the Force Recovery button (the left button)
2. While the Force Recovery button is pressed, briefly press the Reset button (the right button)
3. Let go the Force Recovery button
4. The board will reboot into Force Recovery Mode.

Carrier Board Layout

> **Note**
>
> The diagrams below are from NVIDIA Jetson AGX Thor Developer Kit Carrier Board Specification v1.0.

Module Connector Side Layout

Figure 1-2. Carrier Board Placement – Module Connector Side

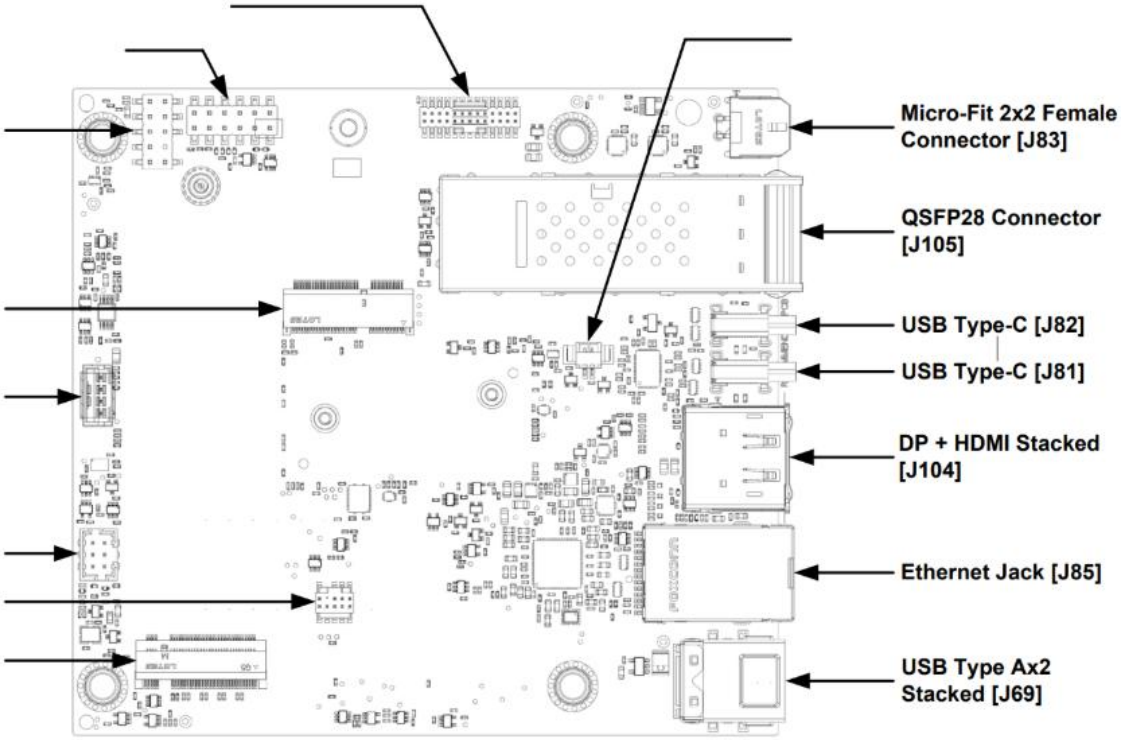


J3 Jetson Module Connector

J90 USB Type-C Connector (Debug)

Non-Module Connector Side Layout

Figure 1-3. Carrier Board Placement – Non-Module Connector Side



J13	RTC Battery Backup Connector	**J87**	4-Pin FAN Header
J42	Automation Header	**J91**	Button & LED Header
J47	CAN Header	**J103**	M.2 Key M Connectivity Slot
J69	USB Type-A Stacked	**J104**	DP + HDMI Stacked
J81	USB Type-C Connector #1	**J105**	QSFP28 Connector
J82	USB Type-C Connector #2	**J502**	JTAG Header
J83	Micro-Fit 2x2 Female Connector	**J505**	M.2 Key E Connectivity Slot
J85	Ethernet Jack	**J511**	Audio Panel Header

	Label	Name	Details
	J103	**M.2 Key M Connectivity Slot**	1TB NVMe SSD pre-populated

Enable 25 Gigabit Ethernet on QSFP Port

In the Jetson AGX Thor Developer Kit, the QSFP port can support both 10 Gigabit Ethernet (10GbE) and 25 Gigabit Ethernet (25GbE). The default is 10GbE.

The QSFP port can support only one configuration at a time. To enable 25GbE, apply the following patches, rebuild the BSP, and flash the device.

1. Update the kernel device tree by patching

`Linux_for_Tegra/kernel/dtb/tegra264-p4071-0000+p3834-0008-nv.dtb` as follows:

```
iff --git a/nv-platform/tegra264-p4071-0000.dtsi b/nv-platform/tegra264-p4071-0000.dtsi
index 936b556..317d59d 100644
--- a/nv-platform/tegra264-p4071-0000.dtsi
+++ b/nv-platform/tegra264-p4071-0000.dtsi
@@ -219,8 +219,7 @@
        mgbe0: ethernet@a808a10000 {
            status = "okay";
            nvidia,mac-addr-idx = <1>;
-           nvidia,uphy-gbe-mode = <1>;
-           nvidia,phy-iface-mode = <0x0>;
+           nvidia,uphy-gbe-mode = <2>;
            nvidia,max-platform-mtu = <9000>;
            nvidia,pcs-rx-eq-sw-ovrd = <1>;
            nvidia,pps_op_ctrl = <8>;
@@ -228,7 +227,7 @@

            fixed-link {
                full-duplex;
-               speed = <10000>;
+               speed = <25000>;
            };
        };

@@ -236,8 +235,7 @@
            mgbe1: ethernet@a808b10000 {
            status = "okay";
            nvidia,mac-addr-idx = <2>;
-           nvidia,uphy-gbe-mode = <1>;
-           nvidia,phy-iface-mode = <0x0>;
+           nvidia,uphy-gbe-mode = <2>;
            nvidia,max-platform-mtu = <9000>;
            nvidia,pcs-rx-eq-sw-ovrd = <1>;
            nvidia,pps_op_ctrl = <8>;
```

```
@@ -245,7 +243,7 @@

            fixed-link {
                full-duplex;
-               speed = <10000>;
+               speed = <25000>;
            };
        };

@@ -253,8 +251,7 @@
            mgbe2: ethernet@a808d10000 {
            status = "okay";
            nvidia,mac-addr-idx = <3>;
-           nvidia,uphy-gbe-mode = <1>;
-           nvidia,phy-iface-mode = <0x0>;
+           nvidia,uphy-gbe-mode = <2>;
            nvidia,max-platform-mtu = <9000>;
            nvidia,pcs-rx-eq-sw-ovrd = <1>;
            nvidia,pps_op_ctrl = <8>;
```

```
@@ -262,7 +259,7 @@

             fixed-link {
                 full-duplex;
-                speed = <10000>;
+                speed = <25000>;
             };
         };

@@ -270,8 +267,7 @@
             mgbe3: ethernet@a808e10000 {
             status = "okay";
             nvidia,mac-addr-idx = <4>;
-            nvidia,uphy-gbe-mode = <1>;
-            nvidia,phy-iface-mode = <0x0>;
+            nvidia,uphy-gbe-mode = <2>;
             nvidia,max-platform-mtu = <9000>;
             nvidia,pcs-rx-eq-sw-ovrd = <1>;
             nvidia,pps_op_ctrl = <8>;
@@ -279,7 +275,7 @@

             fixed-link {
                 full-duplex;
-                speed = <10000>;
+                speed = <25000>;
             };
         };
```

2. Build the DTB and copy it to the following locations:
   - `Linux_for_Tegra/kernel/dtb/tegra264-p4071-0000+p3834-0008-nv.dtb`
   - `Linux_for_Tegra/rootfs/boot/tegra264-p4071-0000+p3834-0008-nv.dtb`
   - `Linux_for_Tegra/bootloader/tegra264-p4071-0000+p3834-0008-nv.dtb`
3. Update the BPMP device tree by patching `Linux_for_Tegra/bootloader/generic/tegra264-bpmp-3834-0008-4071-xxxx.dtb` as follows:

```
--- a/tegra264-bpmp-3834-0008-4071-xxxx.dts
+++ b/tegra264-bpmp-3834-0008-4071-xxxx.dts
@@ -68,15 +68,15 @@
  *
  *   UPHY1
  *   Option | Lane 0 | Lane 1 | Lane 2 | Lane 3 |        Lane 4    |      Lane 5      |
- *      7   |          PCIE C5 x4 (DM)          |     MGBE0 10G    |     MGBE1 10G    |
+ *      8   |          PCIE C5 x4 (DM)          |     MGBE0 25G    |     MGBE1 25G    |
  */
         status = "okay";
         uphy0-config = <7>;       // Configure UPHY0-Lane6-7 for UFS
-        uphy1-config = <7>;       // Configure UPHY1-Lane0-3 for C5 to enable mgbe
-        mgbe0-speed = <2>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
-        mgbe1-speed = <2>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
-        mgbe2-speed = <2>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
-        mgbe3-speed = <2>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
+        uphy1-config = <8>;       // Configure UPHY1-Lane0-3 for C5 to enable mgbe
+        mgbe0-speed = <3>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
+        mgbe1-speed = <3>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
+        mgbe2-speed = <3>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
+        mgbe3-speed = <3>;        /* 0 for 2.5G, 1 for 5G, 2 for 10G, 3 for 25G */
     };

     nvtherm {
```

4. Build the BPMP DTB and copy it to the following locations:
   - `Linux_for_Tegra/bootloader/generic/tegra264-bpmp-3834-0008-4071-xxxx.dtb`
   - `Linux_for_Tegra/bootloader/tegra264-bpmp-3834-0008-4071-xxxx.dtb`
5. Re-flash the board.

Supported Hardware

Please check the NVIDIA Jetson Thor Series Supported Components List hosted on Jetson Download Center for the list of supported hardware.
https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Thor%20Series%20Supported%20Component%20List

Interim Solutions

This section contains temporary workarounds and interim solutions for known issues.

- UEFI Firmware & Jetson ISO Compatibility
  https://docs.nvidia.com/jetson/agx-thor-devkit/user-guide/latest/twa_uefi_iso_compatibility.html
- Re-enable USB stick installation

  https://docs.nvidia.com/jetson/agx-thor-devkit/user-guide/latest/twa_display_handoff.html
- Headless installation on UEFI 38.0.0

  https://docs.nvidia.com/jetson/agx-thor-devkit/user-guide/latest/twa_headless_on_uefi-38-0-0.html