

JetPack-6.x Installation for ADL-AI2500

Updated source date: April 08, 2026

<p>WHAT YOU WILL LEARN</p> <p>Include the kernel files in the Jetson OS image; install Jetson OS; install Jetson SDK components.</p>	<p>ENVIRONMENT</p> <p>Hardware: ADL-AI2500 with 2TB NVME SSD Host OS: Ubuntu 22.04 LTS</p>
<p>COMPATIBLE MODULES</p> <p>Jetson Orin NX and Jetson Orin Nano modules. Only the BSP archive changes by module.</p>	<p>RECOMMENDED WORKFLOW</p> <p>Build the OS image first, apply the BSP files, flash the device, then install the runtime / SDK components.</p>

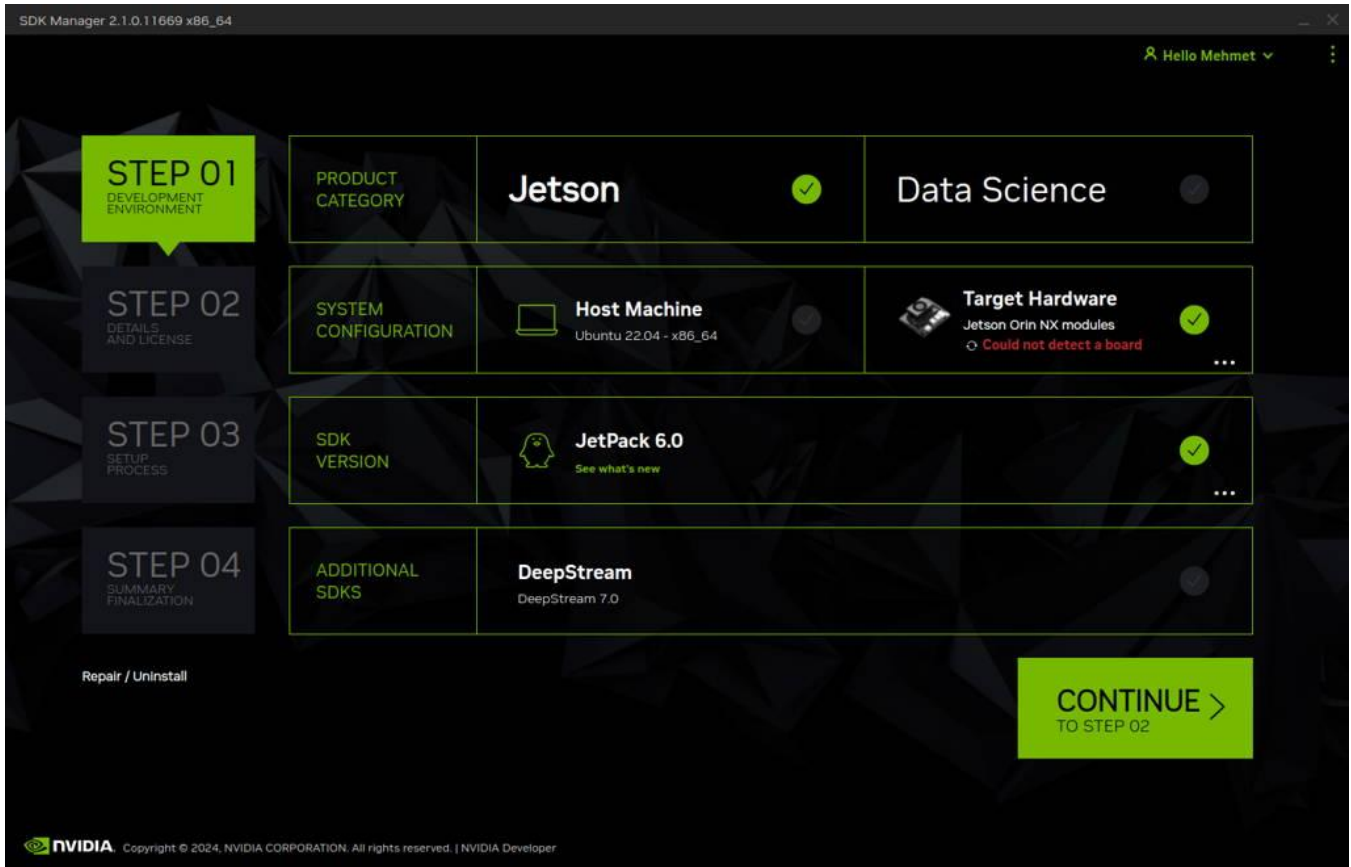
1. Overview

This guide walks through the ADL-AI2500 JetPack 6.x setup in three phases: prepare the Jetson OS image with the correct BSP files, flash the Jetson OS to the target device, and install the runtime / SDK components over Ethernet.

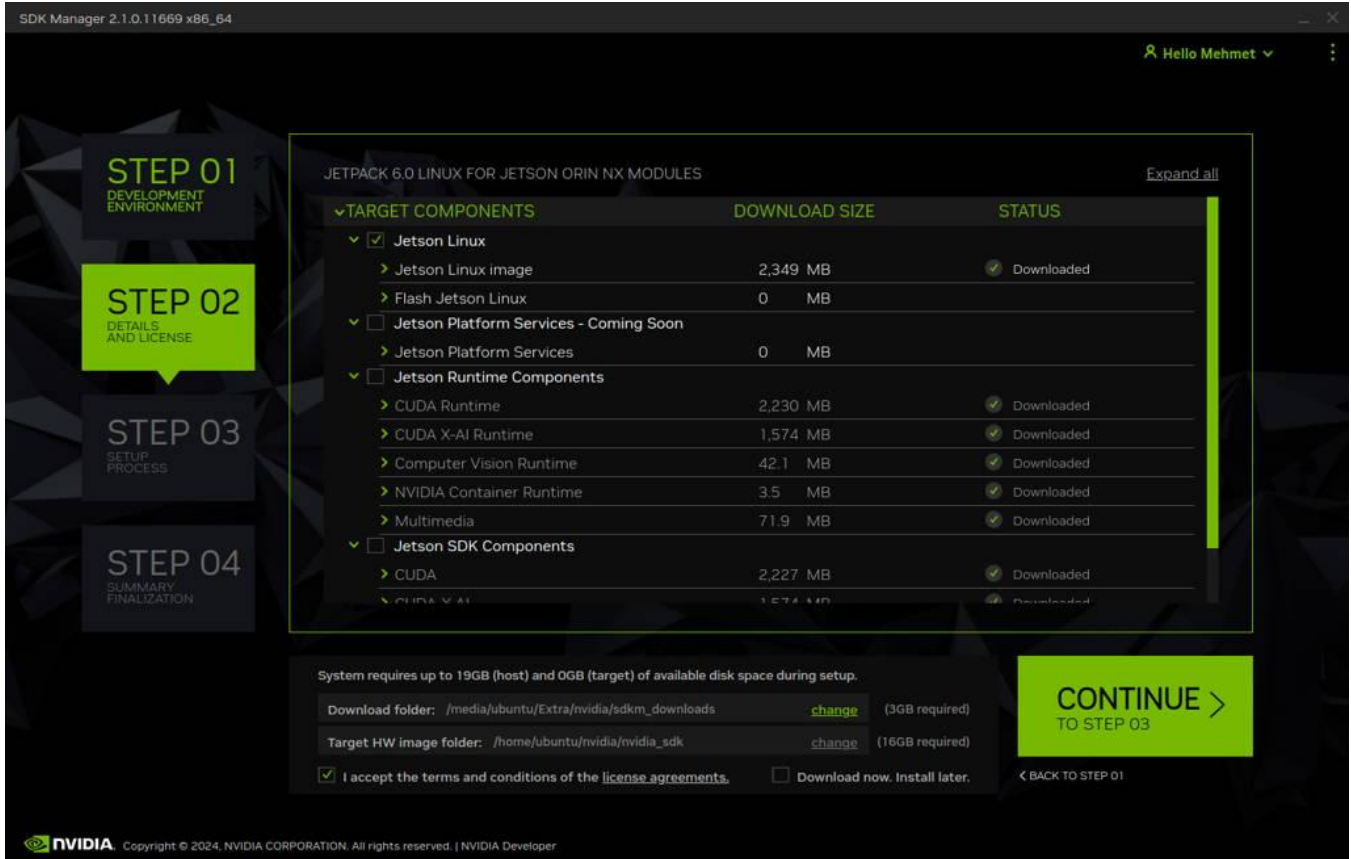
2. Prepare the Jetson OS image

Open NVIDIA SDK Manager and select the correct JetPack version and target module (Jetson Orin Nano modules or Jetson Orin NX modules). Host Machine components are not required.

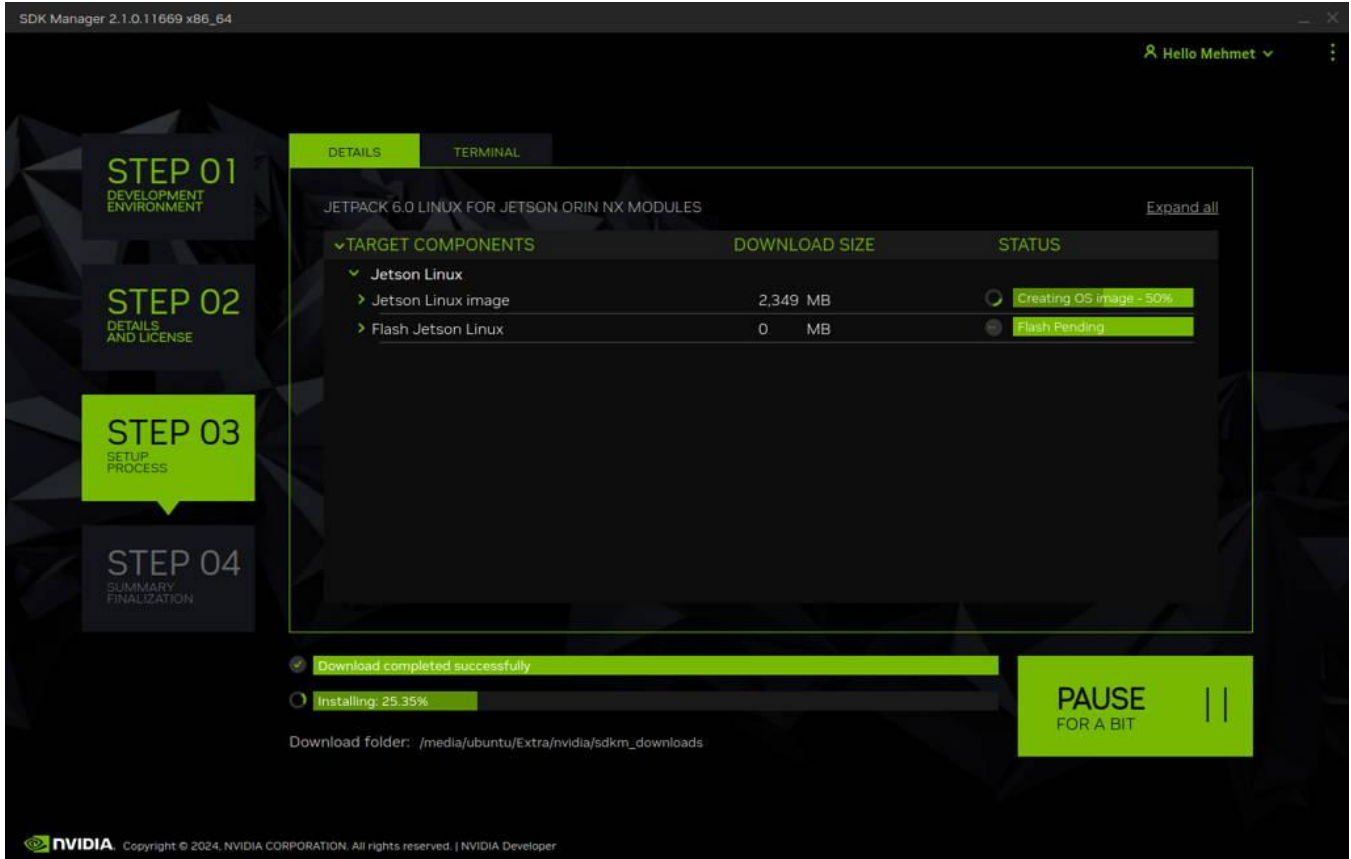
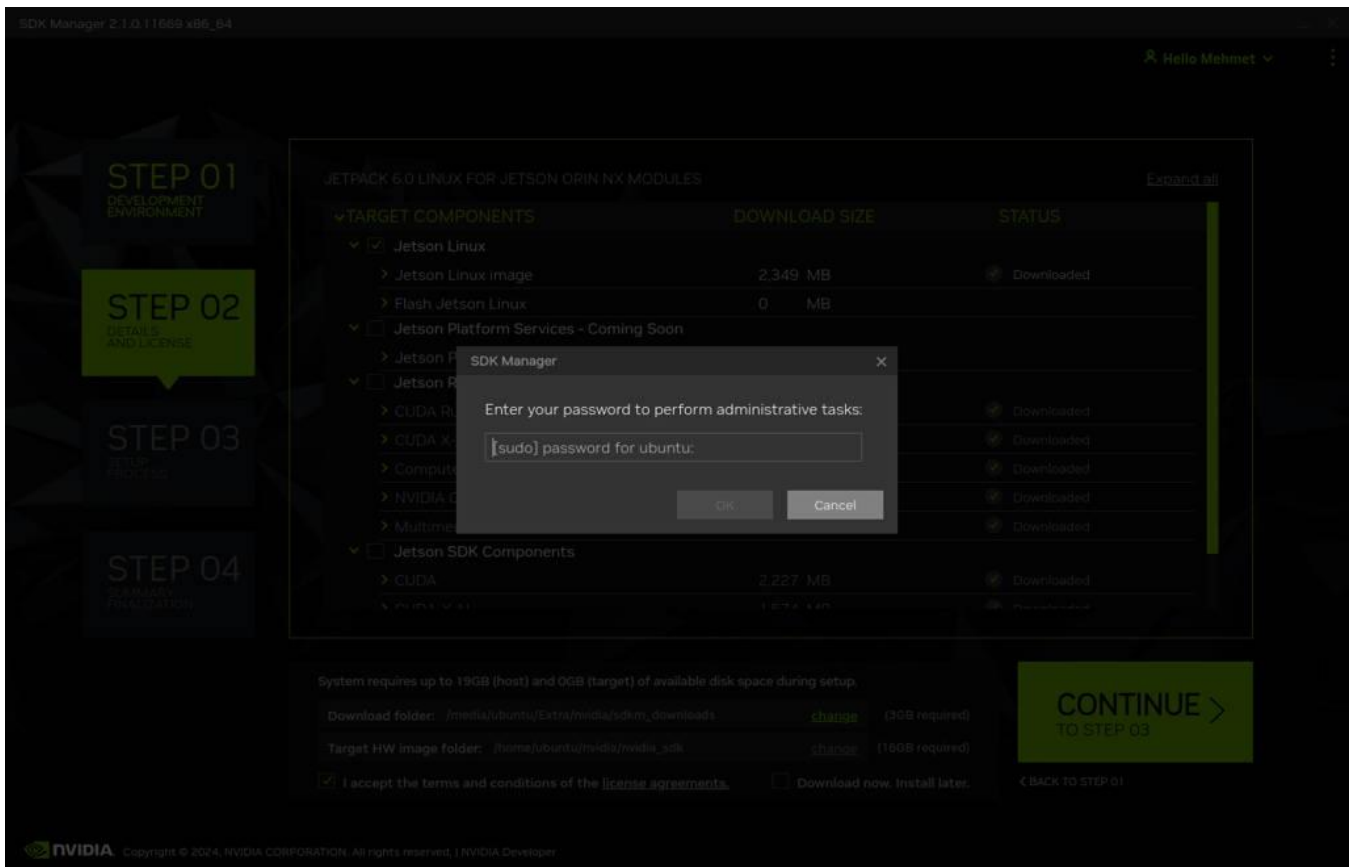
Continue to Step 2..



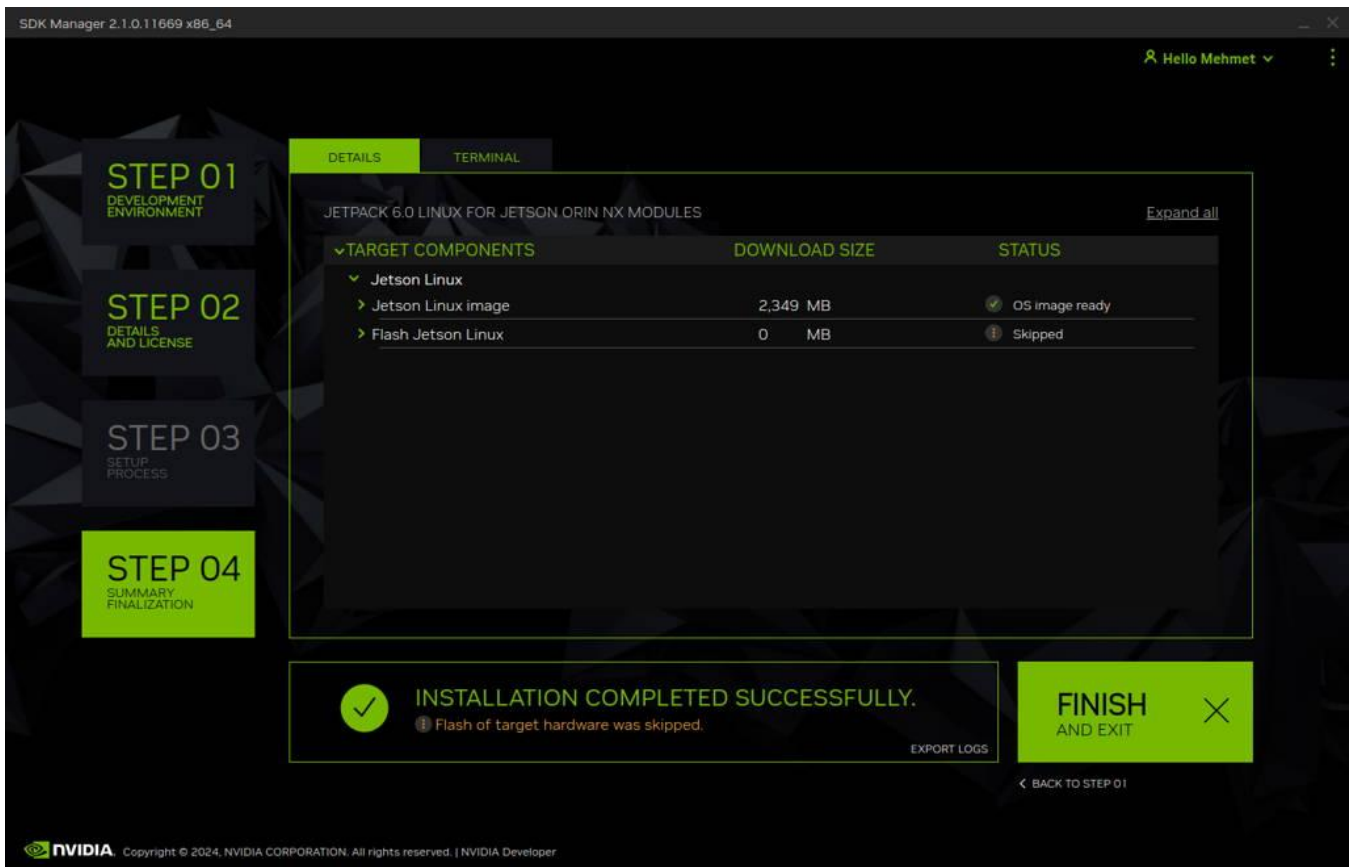
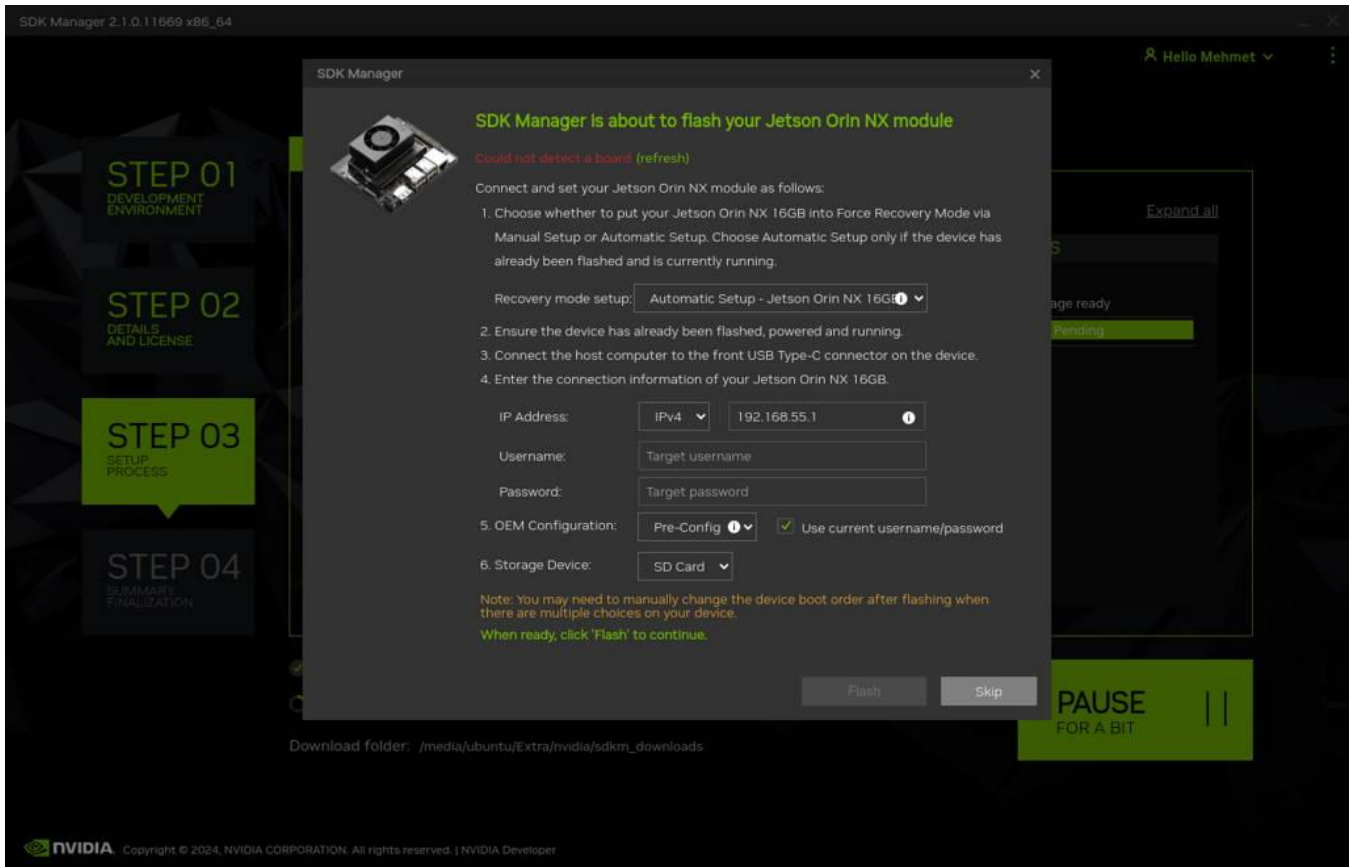
Select only Jetson Linux, accept the terms and conditions, and continue to Step 3.



When SDK Manager asks for the Linux username password, enter it and continue.



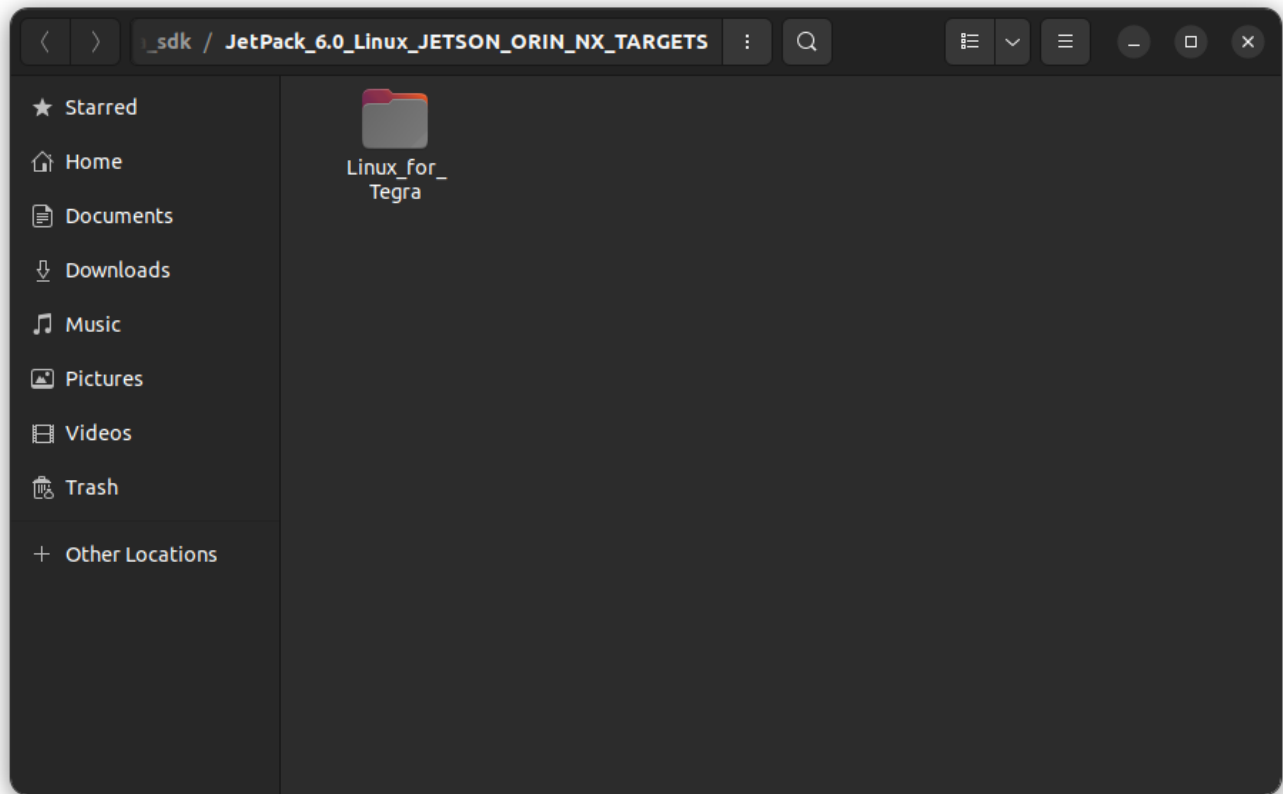
Let SDK Manager build the Jetson OS image. When it later asks about flashing style, skip that step and exit SDK Manager.



Target image folder locations

Open the target HW image folder.

JetPack	Orin NX	Orin Nano
6.0	~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/	~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NANO_TARGETS/
6.1	~/nvidia/nvidia_sdk/JetPack_6.1_Linux_JETSON_ORIN_NX_TARGETS/	~/nvidia/nvidia_sdk/JetPack_6.1_Linux_JETSON_ORIN_NANO_TARGETS/
6.2	~/nvidia/nvidia_sdk/JetPack_6.2_Linux_JETSON_ORIN_NX_TARGETS/	~/nvidia/nvidia_sdk/JetPack_6.2_Linux_JETSON_ORIN_NANO_TARGETS/
6.2.1	~/nvidia/nvidia_sdk/JetPack_6.2.1_Linux_JETSON_ORIN_NX_TARGETS/	~/nvidia/nvidia_sdk/JetPack_6.2.1_Linux_JETSON_ORIN_NANO_TARGETS/



Target HW image folder

Download and apply the BSP files

Download the matching BSP archive for your JetPack version and module type, extract it, then copy the extracted files into the target HW image folder. The remaining steps are the same across module types; only the BSP archive differs.

For NX:

For JetPack – [6.0 Orin NX BSP Files](#)

For JetPack – [6.1 Orin NX BSP Files](#)

For JetPack – [6.2 Orin NX BSP Files](#)

For JetPack – [6.2.1 Orin NX BSP Files](#)

For Nano:

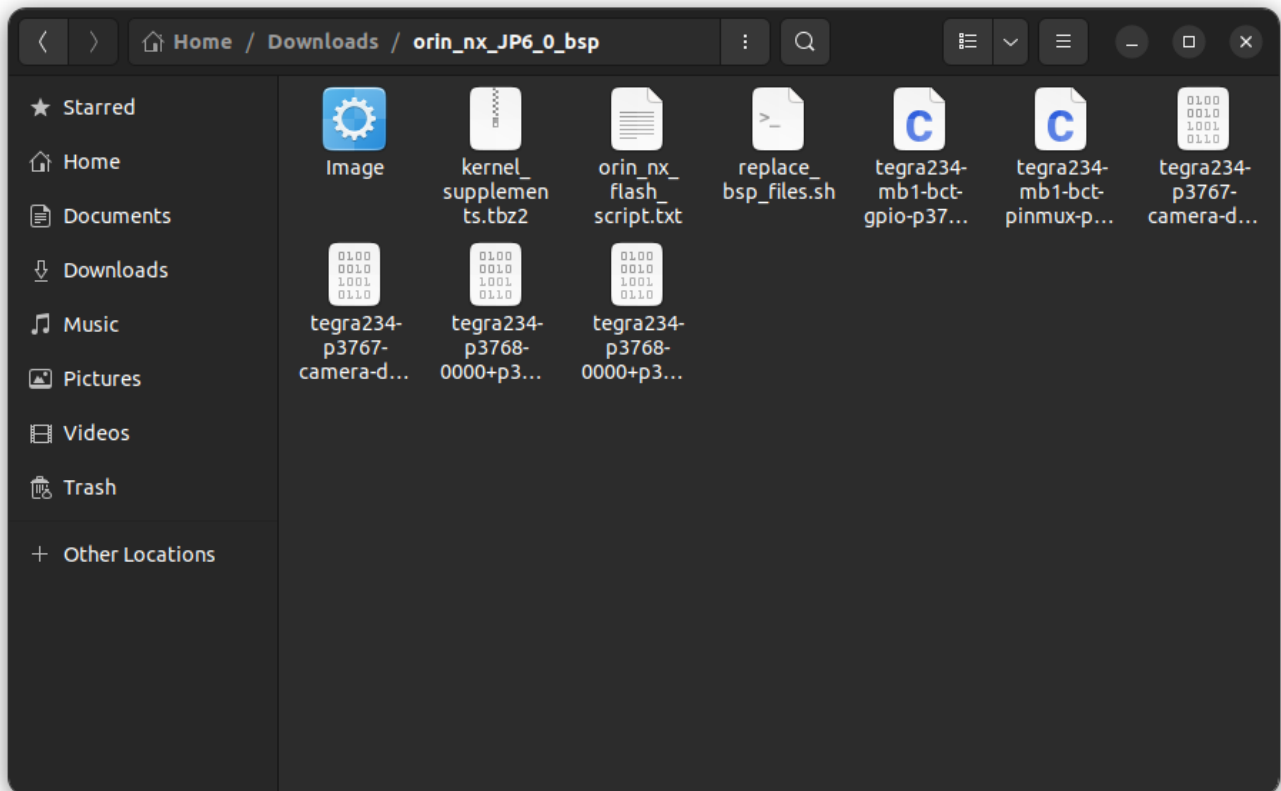
For JetPack – [6.0 Orin Nano BSP Files](#)

For JetPack – [6.1 Orin Nano BSP Files](#)

For JetPack – [6.2 Orin Nano BSP Files](#)

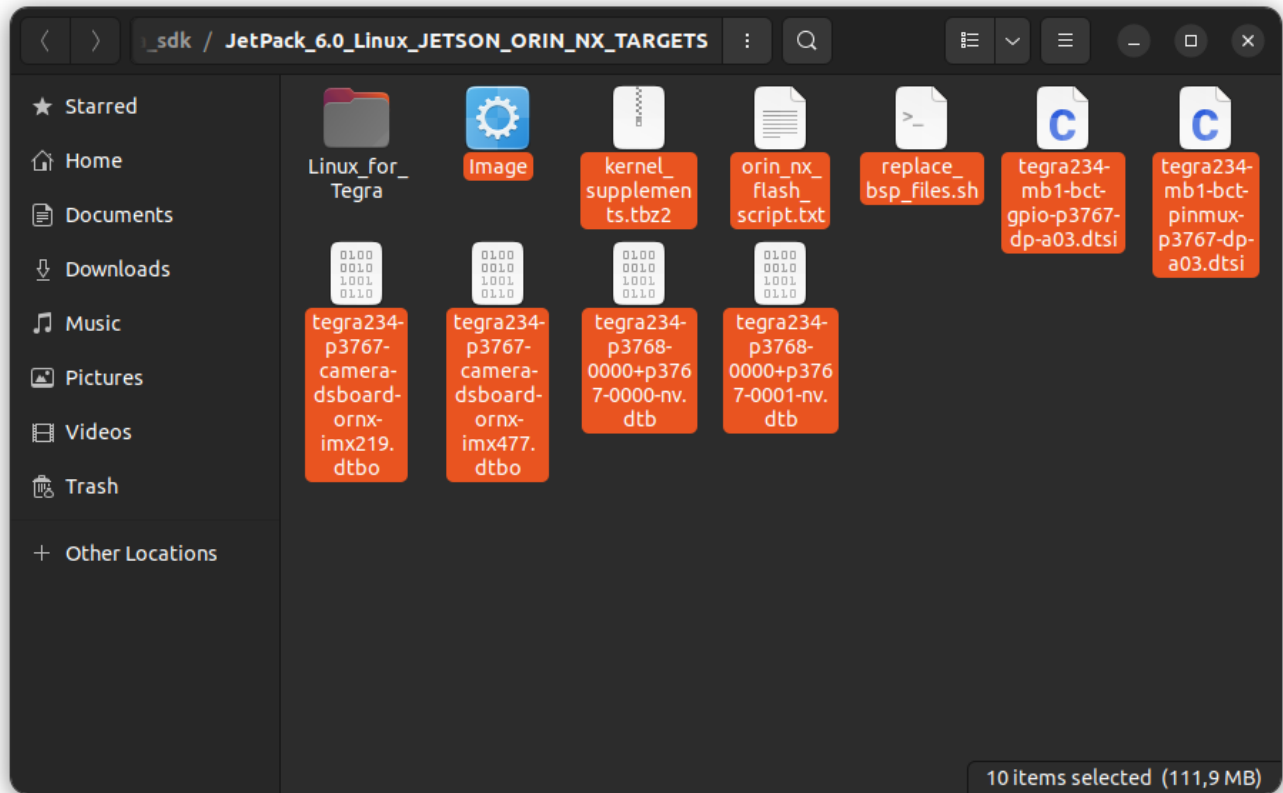
For JetPack – [6.2.1 Orin Nano BSP Files](#)

*Note: **Hint:** The following steps are for the Orin NX, but they are the same for the other Jetson module types (only the BSP files are different).*



Example BSP package extracted files

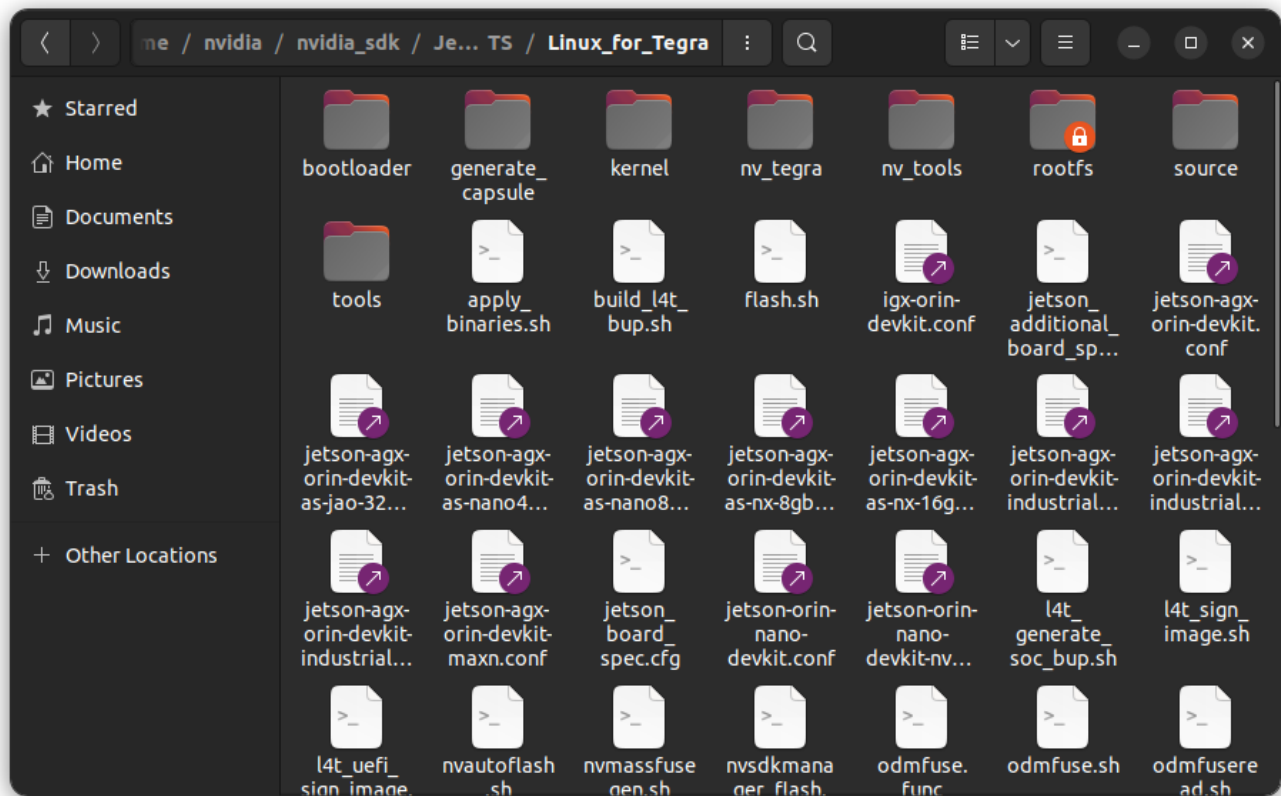
Copy all files to the target HW image folder.



Example BSP package copied to the target image folder

Create the system binaries

Open a terminal in the "Linux_for_Tegra" folder.



Linux_for_Tegra folder

Create the system binaries and apply the NVIDIA binaries.

```
sudo ./tools/l4t_flash_prerequisites.sh  
sudo ./apply_binaries.sh
```

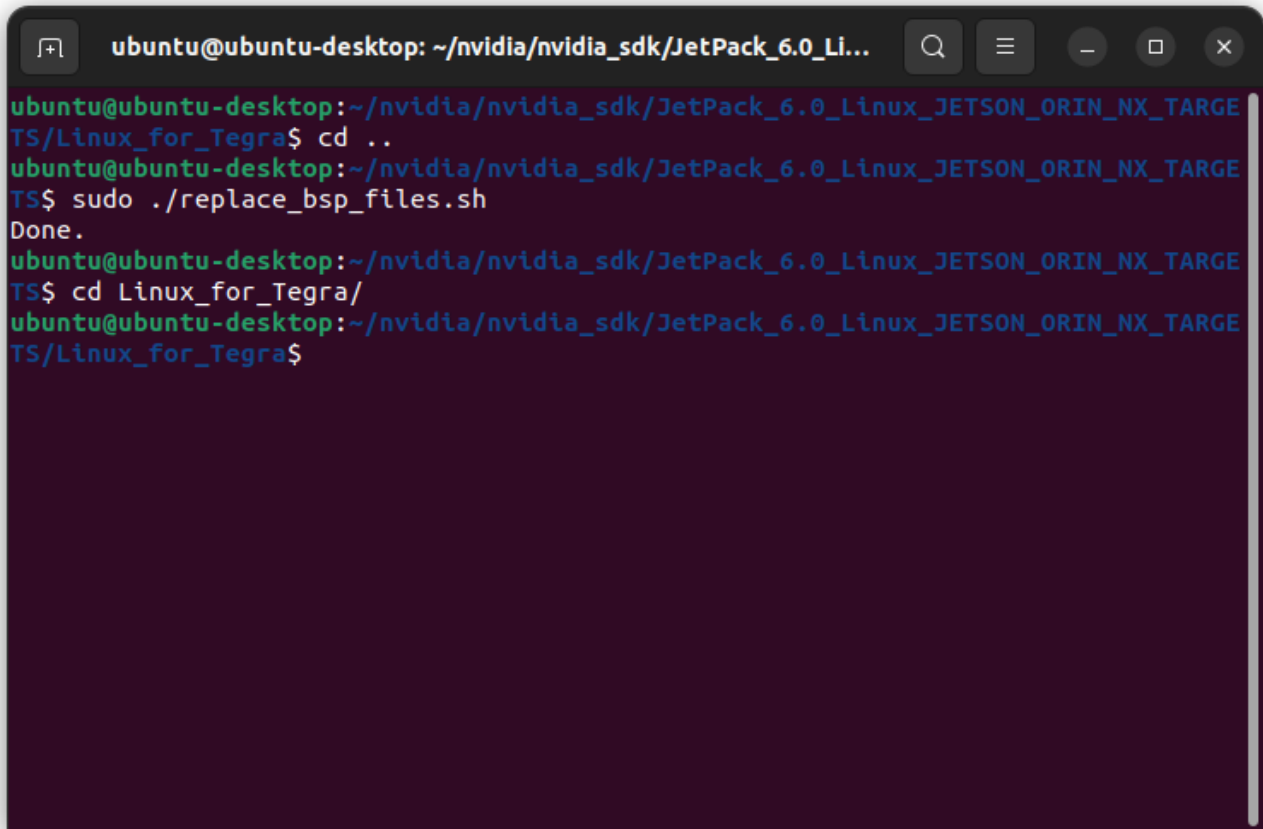
```
ubuntu@ubuntu-desktop: ~/nvidia/nvidia_sdk/JetPack_6.0_Li...
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/Linux_for_Tegra$ sudo ./tools/l4t_flash_prerequisites.sh
```

```
ubuntu@ubuntu-desktop: ~/nvidia/nvidia_sdk/JetPack_6.0_Li...
device-tree-compiler is already the newest version (1.6.1-1).
dosfstools is already the newest version (4.2-1build3).
lz4 is already the newest version (1.9.3-2build2).
python3-yaml is already the newest version (5.4.1-1ubuntu1).
whois is already the newest version (5.5.13).
zstd is already the newest version (1.4.8+dfsg-3build1).
abootimg is already the newest version (0.6-1build1).
lbzip2 is already the newest version (2.5-2.3).
sshd is already the newest version (1.09-1).
binutils is already the newest version (2.38-4ubuntu2.6).
cpio is already the newest version (2.13+dfsg-7ubuntu0.1).
libxml2-utils is already the newest version (2.9.13+dfsg-1ubuntu0.4).
nfs-kernel-server is already the newest version (1:2.6.1-1ubuntu1.2).
openssl is already the newest version (3.0.2-0ubuntu1.15).
rsync is already the newest version (3.2.7-0ubuntu0.22.04.2).
udev is already the newest version (249.11-0ubuntu3.12).
uuid-runtime is already the newest version (2.37.2-4ubuntu3.4).
qemu-user-static is already the newest version (1:6.2+dfsg-2ubuntu6.19).
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/Linux_for_Tegra$ sudo ./apply_binaries.sh
```

Apply the New BSP files and interface configuration

Move up one level, run the BSP replacement script, then return to Linux_for_Tegra.

```
cd ..  
sudo ./replace_bsp_files.sh  
cd Linux_for_Tegra/
```

A terminal window screenshot showing the execution of the BSP replacement script. The terminal title is 'ubuntu@ubuntu-desktop: ~/nvidia/nvidia_sdk/JetPack_6.0_Li...'. The commands and output are: 'cd ..', 'sudo ./replace_bsp_files.sh', 'Done.', and 'cd Linux_for_Tegra/'.

```
ubuntu@ubuntu-desktop: ~/nvidia/nvidia_sdk/JetPack_6.0_Li...  
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/Linux_for_Tegra$ cd ..  
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS$ sudo ./replace_bsp_files.sh  
Done.  
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS$ cd Linux_for_Tegra/  
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/Linux_for_Tegra$
```

Optional: create a default user before first boot

If you want to bypass the Ubuntu first-boot wizard, create the username, password, and hostname in advance with the command below:

```
sudo tools/l4t_create_default_user.sh -u {USERNAME} -p {PASSWORD} -a -n {HOSTNAME} --accept-license
```

Example:

```
sudo tools/l4t_create_default_user.sh -u ADL -p ADL -a -n ADL-desktop --accept-license
```

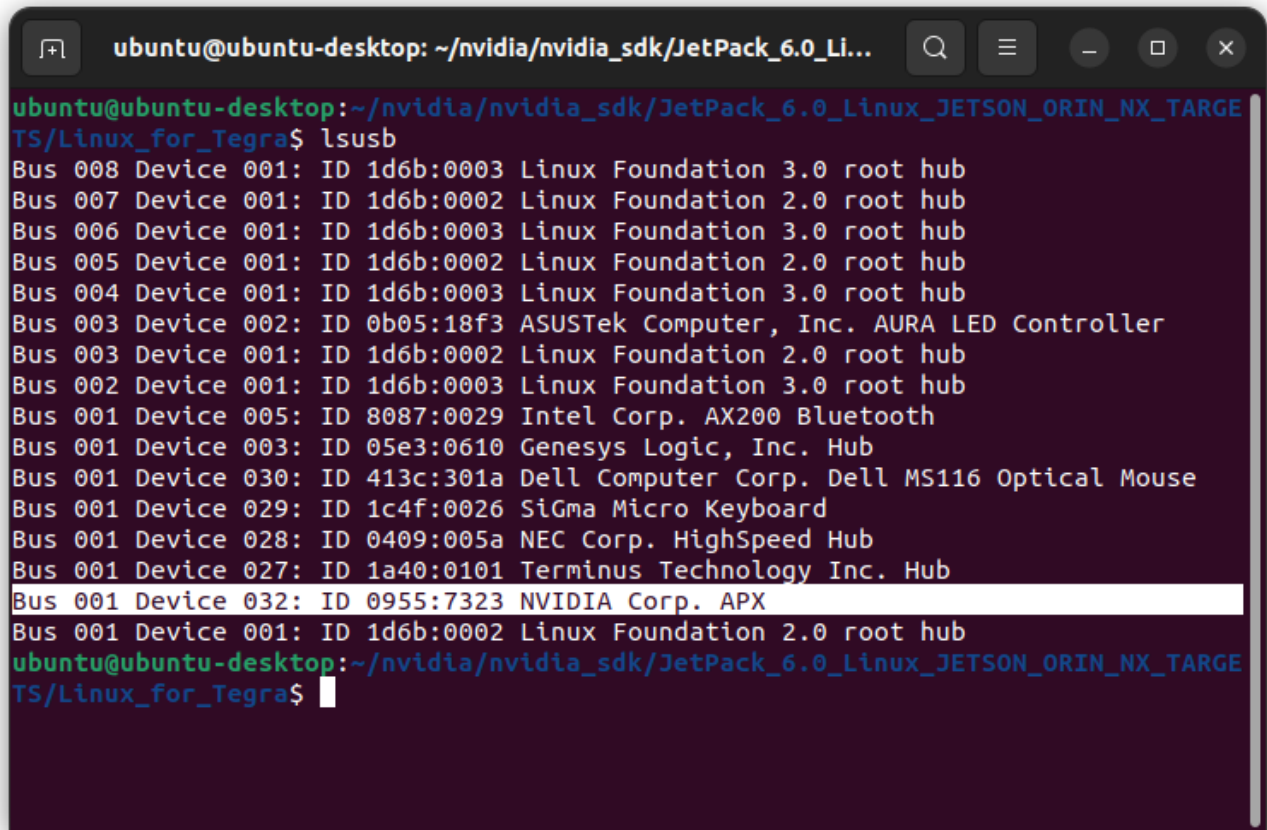
3. Flash Jetson OS to the ADL-AI2500

1. Connect the recovery USB between the installer PC and the ADL-AI2500 recovery USB, then connect power.
2. With power connected, press Reset and Recovery together, release Reset, then release Recovery about 3 seconds later to enter Recovery mode.

Attention: for the most stable USB connection, do not flash through a USB hub or docking station. Connect the Jetson directly to the host PC.

3. Run lsusb and confirm the board is detected in Recovery mode.

Device ID	Module
0955:7323	Orin NX 16GB
0955:7423	Orin NX 8GB
0955:7523	Orin Nano 8GB
0955:7623	Orin Nano 4GB



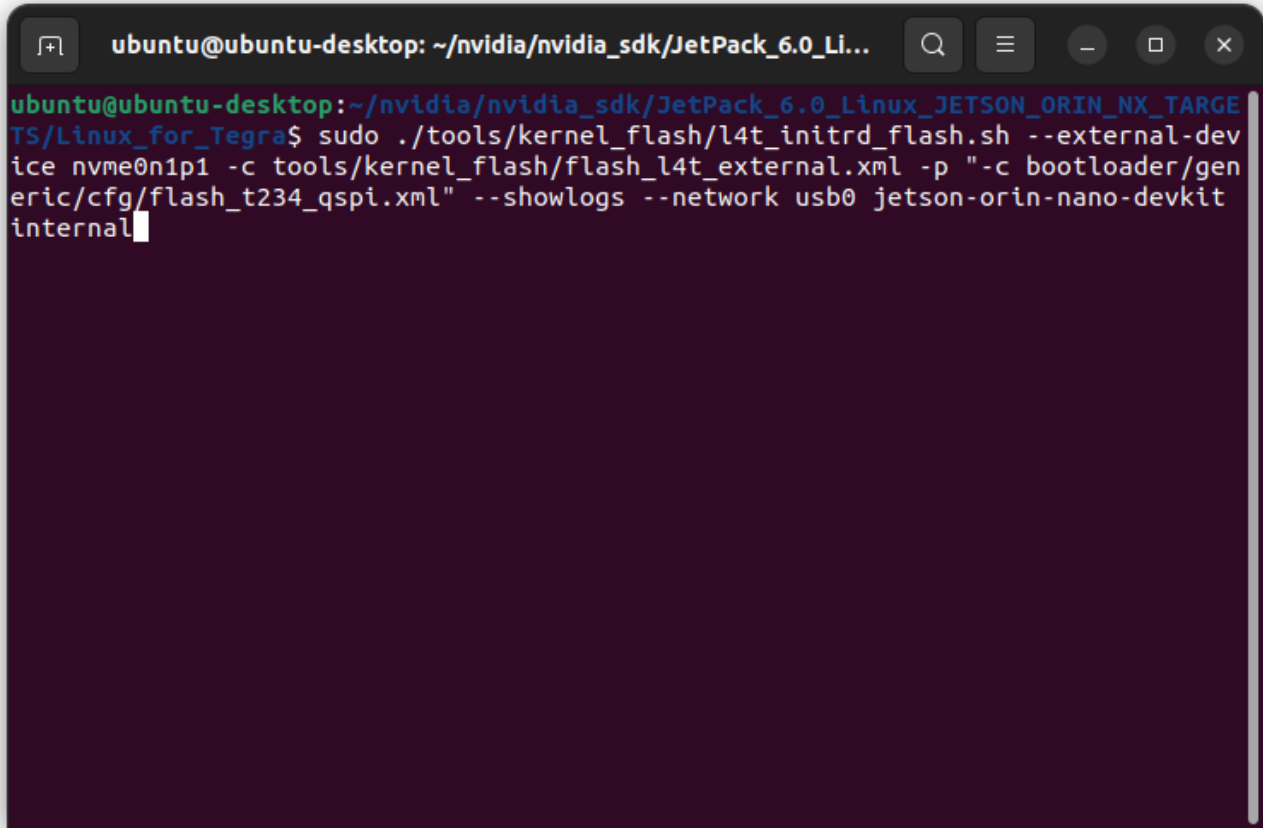
Verify Recovery mode with lsusb

4. Flash the OS to the NVMe drive with the standard command below (To flash it with Super Configuration – Skip to next Step).

```
sudo ./tools/kernel_flash/l4t_initrd_flash.sh --external-device nvme0n1p1 -c
tools/kernel_flash/flash_l4t_external.xml -p "-c bootloader/generic/cfg/flash_t234_qsapi.xml" --
showlogs --network usb0 jetson-orin-nano-devkit internal
```

5. If you want to flash with Super Configuration (supported since JetPack 6.2), use the Super command instead.

```
sudo ./tools/kernel_flash/l4t_initrd_flash.sh --external-device nvme0n1p1 -c
tools/kernel_flash/flash_l4t_external.xml -p "-c bootloader/generic/cfg/flash_t234_qspi.xml" --
showlogs --network usb0 jetson-orin-nano-devkit-super internal
```

A terminal window screenshot showing the execution of the flashing command. The window title is 'ubuntu@ubuntu-desktop: ~/nvidia/nvidia_sdk/JetPack_6.0_Li...'. The prompt is 'ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/Linux_for_Tegra\$'. The command entered is 'sudo ./tools/kernel_flash/l4t_initrd_flash.sh --external-device nvme0n1p1 -c tools/kernel_flash/flash_l4t_external.xml -p "-c bootloader/generic/cfg/flash_t234_qspi.xml" --showlogs --network usb0 jetson-orin-nano-devkit-super internal'. The cursor is at the end of the command.

```
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Li...
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGETS/Linux_for_Tegra$ sudo ./tools/kernel_flash/l4t_initrd_flash.sh --external-device nvme0n1p1 -c tools/kernel_flash/flash_l4t_external.xml -p "-c bootloader/generic/cfg/flash_t234_qspi.xml" --showlogs --network usb0 jetson-orin-nano-devkit-super internal
```

Example flashing command

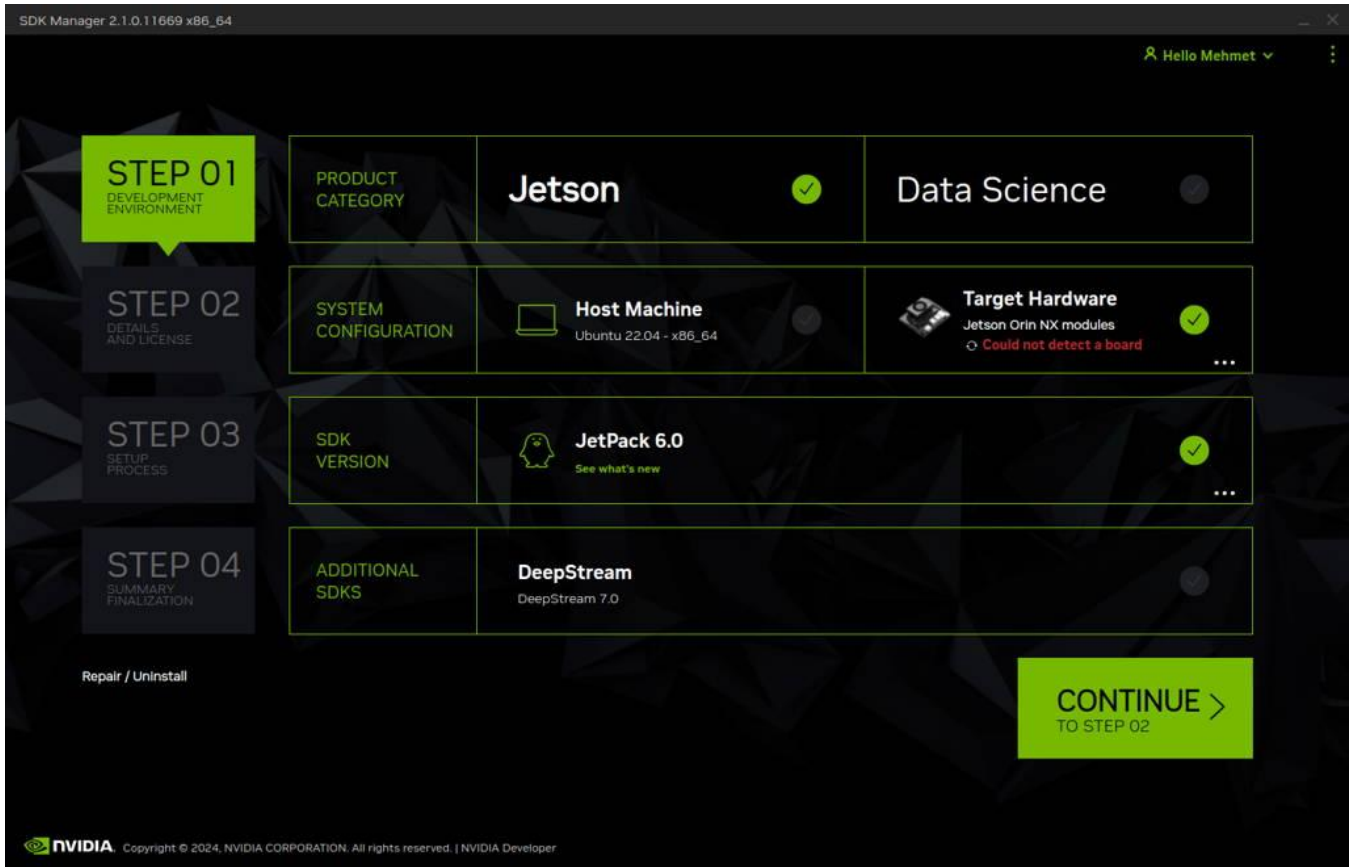
When the script finishes, the device reboots. If you did not create a default user earlier, complete the Ubuntu installation wizard on the ADL-AI2500 (language, keyboard layout, location, username, and password).

```
ubuntu@ubuntu-desktop: ~/nvidia/nvidia_sdk/JetPack_6.0_Li...
Writing /mnt/internal/qspi_bootblob_ver.txt (109 bytes) into /dev/mtd0:66912256
Copied 109 bytes from /mnt/internal/qspi_bootblob_ver.txt to address 0x03fd0000
in flash
Writing qspi_bootblob_ver.txt (parittion: A_VER) into /dev/mtd0
Sha1 checksum matched for /mnt/internal/qspi_bootblob_ver.txt
Writing /mnt/internal/qspi_bootblob_ver.txt (109 bytes) into /dev/mtd0:66977792
Copied 109 bytes from /mnt/internal/qspi_bootblob_ver.txt to address 0x03fe0000
in flash
Writing gpt_secondary_3_0.bin (parittion: secondary_gpt) into /dev/mtd0
Sha1 checksum matched for /mnt/internal/gpt_secondary_3_0.bin
Writing /mnt/internal/gpt_secondary_3_0.bin (16896 bytes) into /dev/mtd0:670919
68
Copied 16896 bytes from /mnt/internal/gpt_secondary_3_0.bin to address 0x03ffbe0
0 in flash
[ 220]: l4t_flash_from_kernel: Successfully flash the qspi
[ 220]: l4t_flash_from_kernel: Flashing success
[ 220]: l4t_flash_from_kernel: The device size indicated in the partition layout
xml is smaller than the actual size. This utility will try to fix the GPT.
Flash is successful
Reboot device
Cleaning up...
Log is saved to Linux_for_Tegra/initrdlog/flash_1-1_0_20240507-095413.log
ubuntu@ubuntu-desktop:~/nvidia/nvidia_sdk/JetPack_6.0_Linux_JETSON_ORIN_NX_TARGE
TS/Linux_for_Tegra$
```

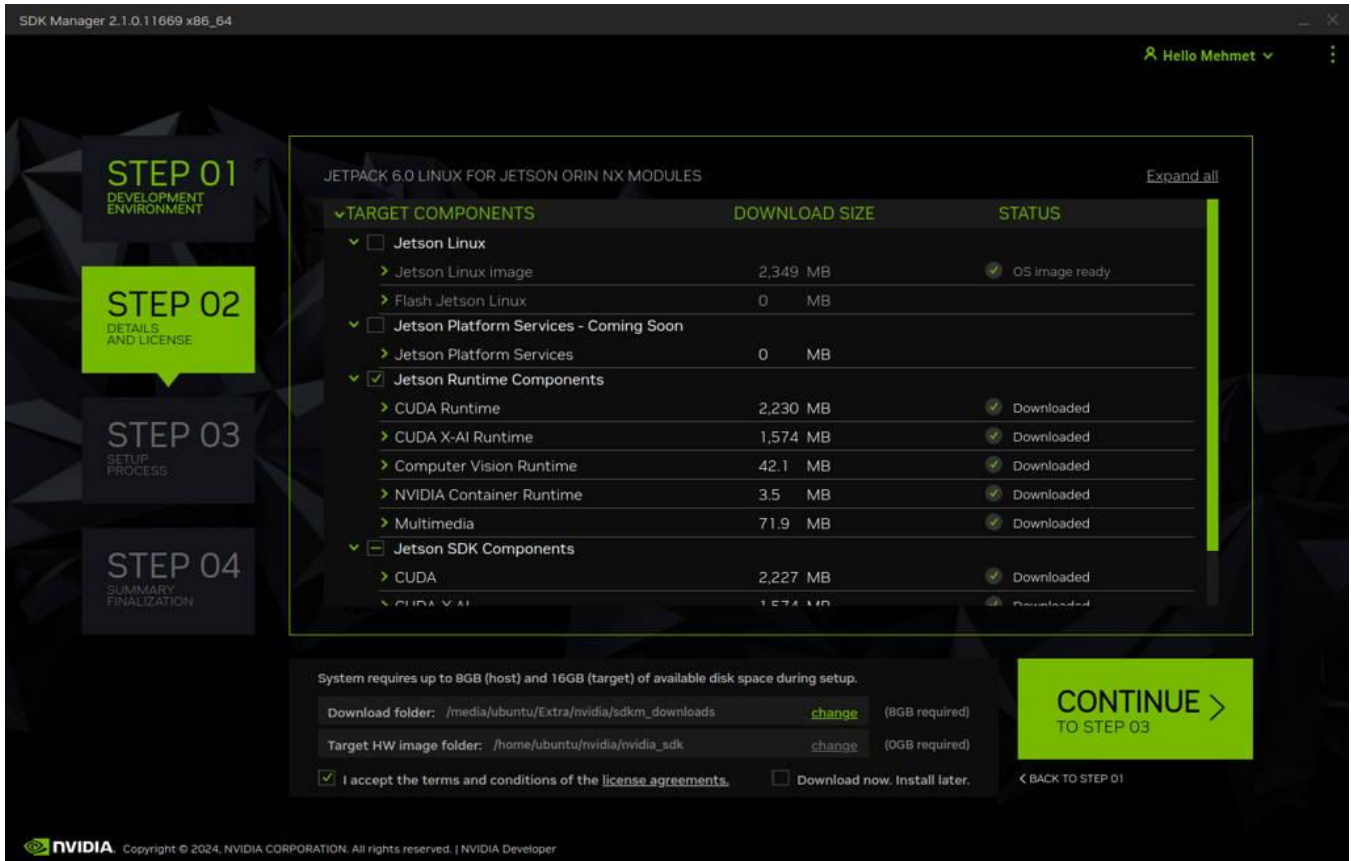
End of the flashing process

4. Install Jetson SDK components

1. Connect the ADL-AI2500 to Ethernet (or use the Recovery USB).
2. Open NVIDIA SDK Manager again and select the correct JetPack version and target module. Host Machine components are still not required. Additional SDKs such as DeepStream are optional.
3. In Step 2, select at least Jetson Runtime Components. Jetson SDK Components are optional depending on the application.
4. Enter the Linux host password when prompted.
5. Provide the ADL-AI2500 IP address (if using Ethernet), username, and password, then install the selected components.



Select the Jetson Runtime Components



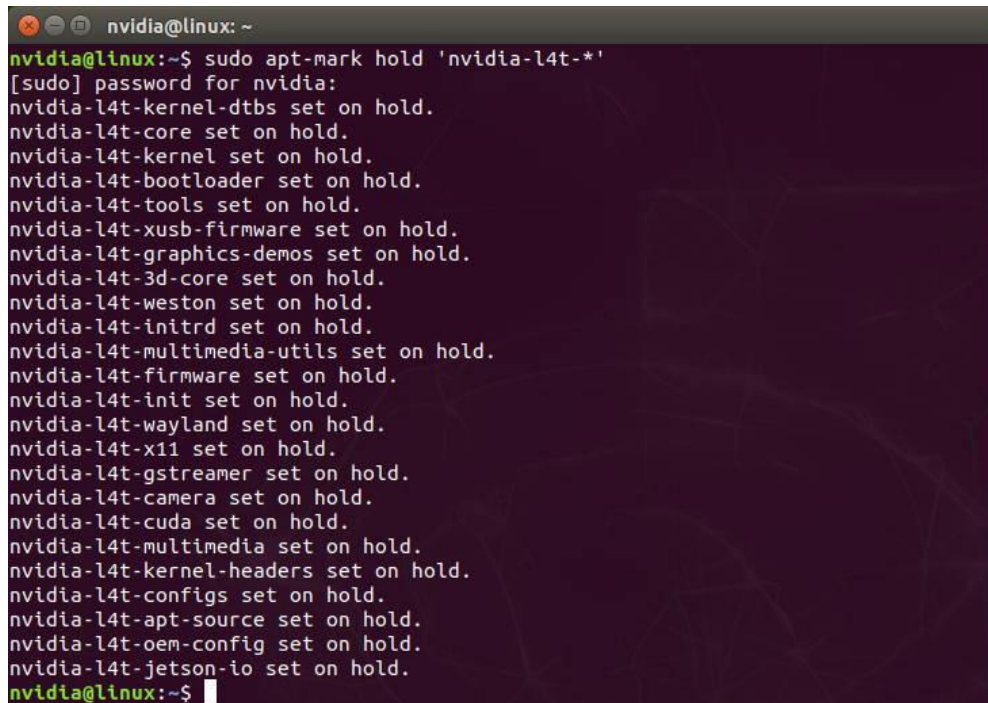
5. Final note

To avoid unwanted kernel updates from "apt upgrade" or "apt-get upgrade", follow this Jetson module guide:

Holding the L4T Packages

Open a terminal and type this command below:

```
sudo apt-mark hold 'nvidia-l4t-*
```



```
nvidia@linux: ~  
nvidia@linux:~$ sudo apt-mark hold 'nvidia-l4t-*' [sudo] password for nvidia:  
nvidia-l4t-kernel-dtbs set on hold.  
nvidia-l4t-core set on hold.  
nvidia-l4t-kernel set on hold.  
nvidia-l4t-bootloader set on hold.  
nvidia-l4t-tools set on hold.  
nvidia-l4t-xusb-firmware set on hold.  
nvidia-l4t-graphics-demos set on hold.  
nvidia-l4t-3d-core set on hold.  
nvidia-l4t-weston set on hold.  
nvidia-l4t-initrd set on hold.  
nvidia-l4t-multimedia-utils set on hold.  
nvidia-l4t-firmware set on hold.  
nvidia-l4t-init set on hold.  
nvidia-l4t-wayland set on hold.  
nvidia-l4t-x11 set on hold.  
nvidia-l4t-gstreamer set on hold.  
nvidia-l4t-camera set on hold.  
nvidia-l4t-cuda set on hold.  
nvidia-l4t-multimedia set on hold.  
nvidia-l4t-kernel-headers set on hold.  
nvidia-l4t-configs set on hold.  
nvidia-l4t-apt-source set on hold.  
nvidia-l4t-oem-config set on hold.  
nvidia-l4t-jetson-io set on hold.  
nvidia@linux:~$
```

All “nvidia-l4t-*” packages set on hold.

Upgrading All Packages Without L4T

To check the updates on all packages, type this command on terminal:

```
sudo apt-get update
```

```
nvidia@linux: ~
nvidia@linux:~$ sudo apt-get update
Hit:1 http://ports.ubuntu.com/ubuntu-ports bionic InRelease
Get:2 http://ports.ubuntu.com/ubuntu-ports bionic-updates InRelease [88.7 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports bionic-backports InRelease [74.6 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports bionic-security InRelease [88.7 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports bionic-updates/main arm64 DEP-11 Meta
data [291 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports bionic-updates/universe arm64 DEP-11
Metadata [295 kB]
Hit:7 https://repo.download.nvidia.com/jetson/common r32.5 InRelease
Get:8 https://repo.download.nvidia.com/jetson/t194 r32.5 InRelease [2,565 B]
Get:9 http://ports.ubuntu.com/ubuntu-ports bionic-backports/universe arm64 DEP-1
1 Metadata [9,256 B]
Get:10 http://ports.ubuntu.com/ubuntu-ports bionic-security/main arm64 DEP-11 Me
tadata [49.0 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports bionic-security/universe arm64 DEP-1
1 Metadata [54.5 kB]
Get:12 https://repo.download.nvidia.com/jetson/t194 r32.5/main arm64 Packages [2
1.5 kB]
Fetched 975 kB in 12s (78.6 kB/s)
Reading package lists... Done
nvidia@linux:~$
```

Then, you can try to upgrade them:

```
sudo apt-get upgrade
```

```
nvidia@linux: ~
nvidia@linux:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
 nvidia-l4t-3d-core nvidia-l4t-apt-source nvidia-l4t-bootloader
 nvidia-l4t-camera nvidia-l4t-configs nvidia-l4t-core nvidia-l4t-cuda
 nvidia-l4t-firmware nvidia-l4t-graphics-demos nvidia-l4t-gstreamer
 nvidia-l4t-init nvidia-l4t-initrd nvidia-l4t-jetson-io nvidia-l4t-kernel
 nvidia-l4t-kernel-dtbs nvidia-l4t-kernel-headers nvidia-l4t-multimedia
 nvidia-l4t-multimedia-utils nvidia-l4t-oem-config nvidia-l4t-tools
 nvidia-l4t-wayland nvidia-l4t-weston nvidia-l4t-x11 nvidia-l4t-xusb-firmware
 ubuntu-drivers-common
The following packages will be upgraded:
 appport appport-gtk apt aptdaemon aptdaemon-data aspell avahi-autoipd
 avahi-daemon avahi-utils base-files bind9-host binutils
 binutils-aarch64-linux-gnu binutils-common bluez bluez-obexd
 busybox-initramfs busybox-static ca-certificates chromium-browser
 chromium-browser-l10n chromium-codecs-ffmpeg-extra containerd cpio deja-dup
 dirmngr distro-info-data dnsmasq-base docker.io file-roller fwupd
 fwupd-signed ghostscript ghostscript-x gir1.2-appindicator3-0.1
 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0
 gir1.2-javascriptcoregtk-4.0 gir1.2-mutter-2 gir1.2-polkit-1.0
```

At the beginning of this command, it can be seen that all hold packages will keep back.

Press “y” to accept installing packages on the Jetson module.

```
nvidia@linux: ~  
libwavpack1 libwbclient0 libwebkit2gtk-4.0-37 libwebp6 libwebpdemux2  
libwebpmux3 libwhoopsie0 libwinpr2-2 libx11-6 libx11-data libx11-dev  
libx11-doc libx11-xcb-dev libx11-xcb1 libxml2 libxml2-dev libzstd1  
linux-base linux-firmware linux-libc-dev locales login lshw  
multiarch-support mutter mutter-common nfs-common ntfs-3g oem-config  
oem-config-gtk openssh-client openssh-server openssh-sftp-server openssl  
p11-kit p11-kit-modules passwd policykit-1 python-apt-common  
python-jetson-gpio python2.7 python2.7-minimal python3-apport python3-apt  
python3-aptdaemon python3-aptdaemon.gtk3widgets python3-distupgrade  
python3-httplib2 python3-jetson-gpio python3-lxml python3-problem-report  
python3-software-properties python3-xdg python3.6 python3.6-minimal  
qt5-gtk-platformtheme rpcbind rsync runc samba-lsmbd snapd  
software-properties-common software-properties-gtk squashfs-tools sudo  
systemd systemd-sysv tar thunderbird thunderbird-gnome-support tzdata  
ubiquity ubiquity-frontend-debconf ubiquity-frontend-gtk  
ubiquity-frontend-kde ubiquity-ubuntu-artwork ubuntu-keyring  
ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk udev unzip  
update-notifier update-notifier-common vim vim-common vim-runtime whoopsie  
wireless-regdb wpasupplicant xdg-utils xserver-common xserver-xephyr  
xserver-xorg-core xserver-xorg-legacy xterm xwayland xxd  
302 upgraded, 0 newly installed, 0 to remove and 25 not upgraded.  
Need to get 451 MB of archives.  
After this operation, 191 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

A few minutes later... (it depends on network speed)

```
nvidia@linux: ~  
Processing triggers for menu (2.1.47ubuntu2.1) ...  
Setting up gir1.2-mutter-2:arm64 (3.28.4+git20200505-0ubuntu18.04.2) ...  
Setting up mutter (3.28.4+git20200505-0ubuntu18.04.2) ...  
Setting up gnome-shell (3.28.4-0ubuntu18.04.7) ...  
Processing triggers for dictionaries-common (1.27.2) ...  
Processing triggers for ca-certificates (20210119-18.04.2) ...  
Updating certificates in /etc/ssl/certs...  
0 added, 0 removed; done.  
Running hooks in /etc/ca-certificates/update.d...  
done.  
Processing triggers for initramfs-tools (0.130ubuntu3.13) ...  
update-initramfs: Generating /boot/initrd.img-4.9.201-tegra  
cryptsetup: WARNING: failed to detect canonical device of /dev/root  
cryptsetup: WARNING: could not determine root device from /etc/fstab  
Warning: couldn't identify filesystem type for fsck hook, ignoring.  
I: The initramfs will attempt to resume from /dev/zram5  
I: (UUID=a3cee9e7-7990-493b-879e-23e9436f8399)  
I: Set the RESUME variable to override this.  
/sbin/ldconfig.real: Warning: ignoring configuration file that cannot be opened:  
/etc/ld.so.conf.d/aarch64-linux-gnu_EGL.conf: No such file or directory  
/sbin/ldconfig.real: Warning: ignoring configuration file that cannot be opened:  
/etc/ld.so.conf.d/aarch64-linux-gnu_GL.conf: No such file or directory  
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...  
nvidia@linux:~$
```

At the end of the upgrading step, you can check it again.

```
nvidia@linux: ~
nvidia@linux:~$ sudo apt-get update
[sudo] password for nvidia:
Hit:1 http://ports.ubuntu.com/ubuntu-ports bionic InRelease
Hit:2 http://ports.ubuntu.com/ubuntu-ports bionic-updates InRelease
Hit:3 http://ports.ubuntu.com/ubuntu-ports bionic-backports InRelease
Hit:4 http://ports.ubuntu.com/ubuntu-ports bionic-security InRelease
Hit:5 https://repo.download.nvidia.com/jetson/common r32.5 InRelease
Hit:6 https://repo.download.nvidia.com/jetson/t194 r32.5 InRelease
Reading package lists... Done
nvidia@linux:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
 nvidia-l4t-3d-core nvidia-l4t-apt-source nvidia-l4t-bootloader
 nvidia-l4t-camera nvidia-l4t-configs nvidia-l4t-core nvidia-l4t-cuda
 nvidia-l4t-firmware nvidia-l4t-graphics-demos nvidia-l4t-gstreamer
 nvidia-l4t-init nvidia-l4t-initrd nvidia-l4t-jetson-io nvidia-l4t-kernel
 nvidia-l4t-kernel-dtbs nvidia-l4t-kernel-headers nvidia-l4t-multimedia
 nvidia-l4t-multimedia-utils nvidia-l4t-oem-config nvidia-l4t-tools
 nvidia-l4t-wayland nvidia-l4t-weston nvidia-l4t-x11 nvidia-l4t-xusb-firmware
 ubuntu-drivers-common
0 upgraded, 0 newly installed, 0 to remove and 25 not upgraded.
nvidia@linux:~$
```

As it can be seen, the holding command worked.