NVIDIA IGX Orin Developer Kit User Guide

Version 1.1

Version 1.0 | September 2022

Document History

Document Version 1.0

Version	Date	Authors	Description of Change
1.0	September 21, 2022	Peter G.	
1.1	December 16, 2022	Peter G.	Change in expected checklist; note on dGPU only support in SDK; correct a typo in tech specs for CPU; added expected lsusb results; removed subsection “Reinstalling Optional SDK Packages”; added troubleshooting item for “cannot connect to the device via SSH” error during flashing;

Table of Contents

Introduction

Provides the instructions to flash, setup, and start using the NVIDIA IGX Orin Developer Kit.

Disclaimer: The NVIDIA IGX Orin Developer Kit is not an approved medical device and is not intended for clinical use.

Checklist for Setting up the Developer Kit

Each NVIDIA IGX Orin Developer Kit should be pre-flashed prior to delivery. After you have received the Developer Kit, there should be no need to flash the system, please go through the following actions to power up the system and get started with the Clara Holoscan SDK. Each action is described in its corresponding section of this user guide.

- Read through the **Hardware Setup** requirements and precautions.
- Familiarize yourself with the **System Overview**: the main components and system I/O.
- Power up the system.
- Manually update firmware if needed.
- Install the Clara Holoscan SDK from Github.

If you would like to re-flash the system at any point, please go through each of the following actions.

- Read through the **Hardware Setup** requirements and precautions.
- Familiarize yourself with the **System Overview**: the main components and system I/O.
- Power up the system.
- Flash and update the NVIDIA IGX Orin Developer Kit using SDK Manager.
- Switch from iGPU to dGPU mode.
- Set up the 500GB SSD storage.
- Set up Docker and Docker storage.
- Manually update firmware if needed.
- Install the Clara Holoscan SDK from Github.

Note: The current Clara Holoscan SDK is only tested in dGPU mode, therefore switching from iGPU to dGPU is essential. Running the SDK in iGPU is not supported at the moment.

Hardware Setup

Requirements

- A NVIDIA IGX Orin Developer Kit
- A compatible power cable
    - The power cable(s) included with the NVIDIA IGX Orin Developer Kit may not be compatible with your local electrical requirements.
    - A compatible cable should meet the following requirements:
        - Provides a certified local 3-prong AC power plug.
        - Provides a C13 connector.
        - Supports ratings of 100-120VAC/6A, 200-240VAC/3A, or higher with a minimum wire thickness of 18AWG and insulation rating of 300V or higher.
- An Ubuntu 18.04 or 20.04 host system (for use during flashing)
- A standard USB-A to USB-C or USB-C to USB-C cable with data enabled (for use during flashing)
- A standard Micro USB Type-B cable with data enabled (for accessing the baseboard management controller (BMC) during flashing)
- Connection to the Internet for the host system before and during flashing, and for the NVIDIA IGX Orin Developer Kit at minimum after the OS has been flashed
- A keyboard and mouse, as well as a monitor with DisplayPort connection for the NVIDIA IGX Orin Developer Kit

Precautions

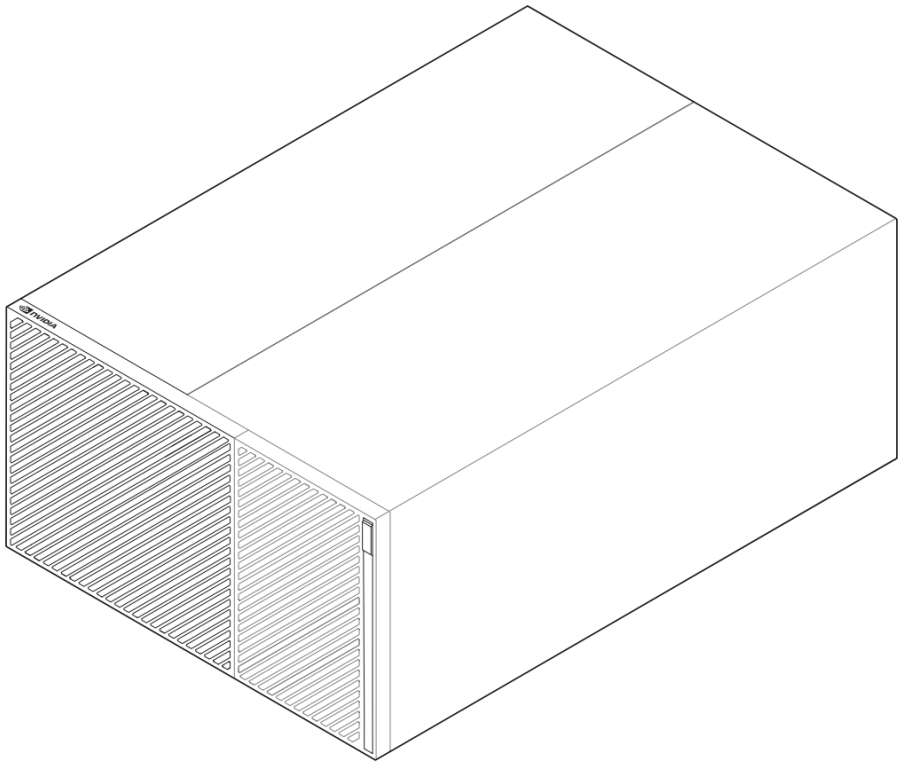
- Only connect and disconnect PCIe cards (e.g. miniSAS or dGPU) when the system is powered down.
- Apply extra care when plugging and removing PCIe cards to avoid stress on the PCIe connectors (e.g. wearing, bending, breaking).

System Overview

Main Components

The NVIDIA IGX Orin Developer Kit contains the following major components:

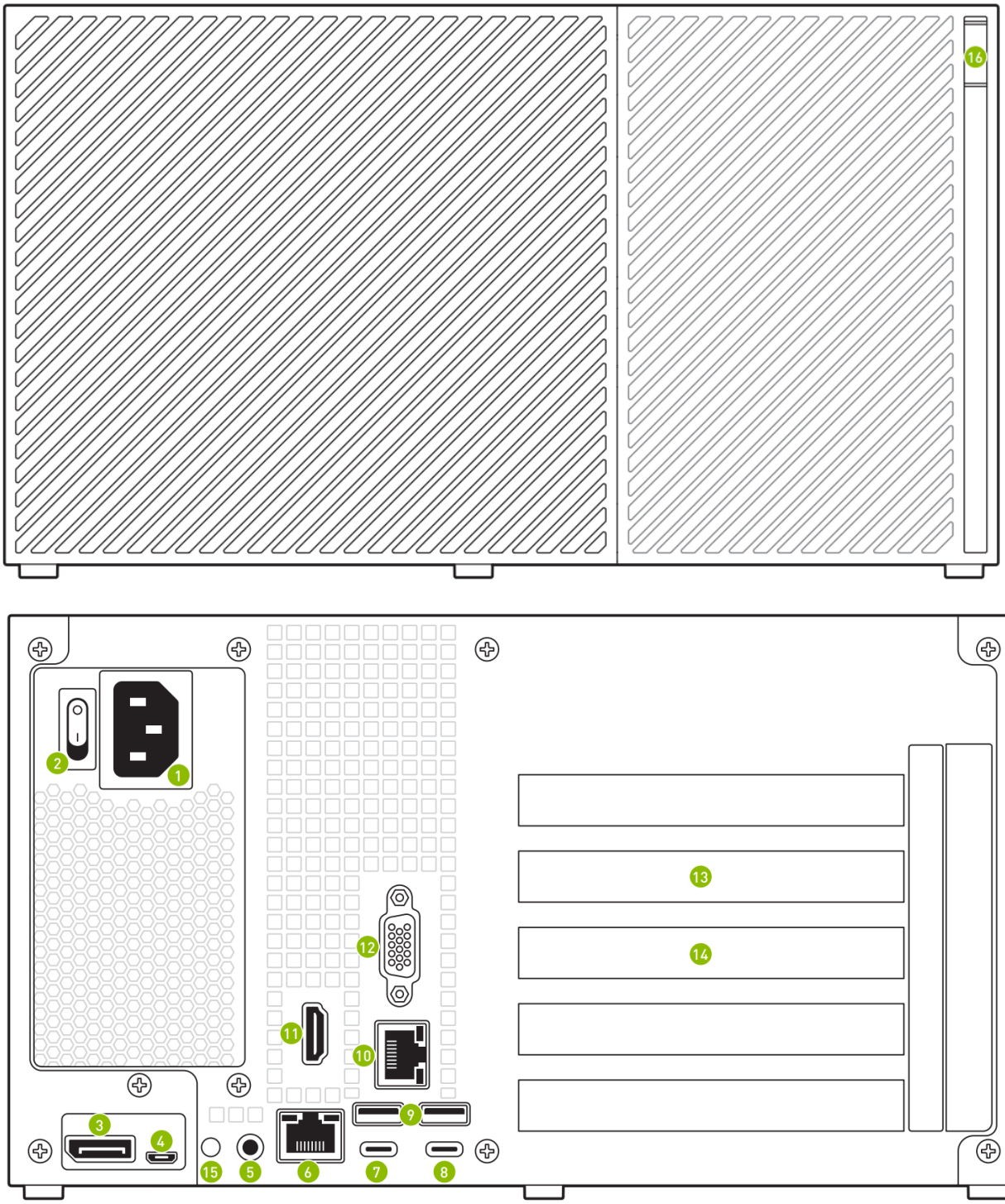
- AGX Orin 32 GB module
- RTX A6000 discrete GPU
- ConnectX-6 DX
- 500GB removeable SSD



Tech Specs

CPU	12-core Arm® Cortex®-A78AE v8.2 64-bit CPU 3MB L2 + 6MB L3
Memory	32GB 256-bit LPDDR5 204.8 GB/s
GPU	RTX A6000 | 48 GB GDDR6 | 768 GB/s | 10,752 CUDA cores | 3rd gen 336 Tensor Cores
Storage	500GB SSD
I/O	Micro USB Type B | (2x) USB3.0 | USB-C | HDMI In | (5x) DisplayPort | 1/100 GbE
Expansion	N/A
Power Supply	850W | 100-240V
Dimensions	262.7mm W X 147.7mm H X 370.0mm L

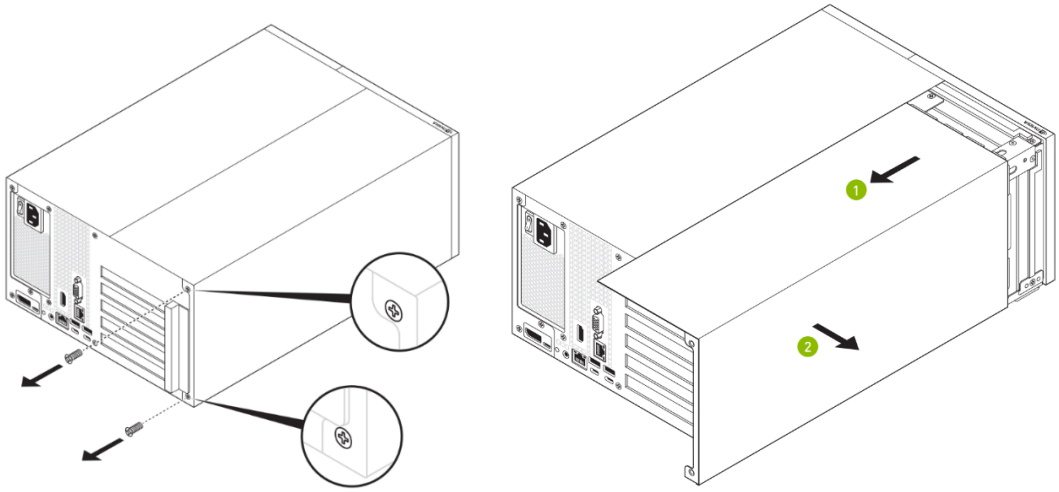
I/O and external interfaces



1) Power cable connection
2) Power switch
3) DisplayPort (DP) output port from Jetson Orin module

4) Micro USB Type B connection to BMC
5) Audio
6) 1 GbE RJ45 Ethernet connection to Orin module
7) Recovery port
8) USB-C port
9) USB-A ports (USB 2.0)
10) 1 GbE RJ45 Ethernet connection to BMC
11) HDMI input
12) VGA port
13) QSFP ConnectX-6 DX board
14) DisplayPort (DP) output ports from RTX A6000
15) Recovery button
16) Power button

To access ports 13-14, remove the left-hand side cover. The process is illustrated below. Unscrew the two Phillips screws that secure the cover at the back of the machine. Next, push and slide the cover towards the back of the machine without lifting (step 1). It should slide about 0.5 inch, or less than 1.5 cm. Finally, you should be able to lift the cover off once it has more than one degree of freedom and can be easily lifted outwards (step 2).



If you'd like to install an AJA video capture card, remove the QSFP ConnectX-6 DX board (13) to free up the PCIe slot and install the AJA card.

Note: Only connect and disconnect a PCIe card (such as the ConnectX-6 DX board) when the system is powered down.

Powering up the System

1) Connect all peripherals to the system before powering up the system.
2) Connect the power cable to the system in the slot labeled (1) in the I/O drawing above and make sure the power switch (2) is on.
3) Once the power is connected, press the power button (16) for approximately 1 second. It should light up.
4) If you have a display connected, you might already see the system booting on it. During flashing or re-flashing, connect your display to the DisplayPort (3), which is connected to the Jetson Orin module. Once the system is set up and running in dGPU mode, connect your display to one of the four the DisplayPorts (14). Reference the GPU section below to determine how to choose between display outputs.

Note: The machine can be powered off by pressing the power button for approximately 10 seconds.

Flashing and Updating the NVIDIA IGX Orin Developer Kit using SDK Manager

1. Register and activate an NVIDIA Developer Account here to access the latest version of Holopack in SDK Manager.
2. If you are running a VPN on your host system, log off before flashing the NVIDIA IGX Orin Developer Kit. Otherwise, you might get an Internet connection error during the flashing process.
3. Using a VM as your host machine isn't officially supported, but it is possible with certain VMs such as VMWare Workstation 16. If using a VM, ensure the ports that connect to the USB-C recovery port (7) and Micro USB Type B port (4) on the NVIDIA IGX Orin Developer Kit are routed to the VM.
4. From the host system, download and install the latest version of NVIDIA SDK Manager. Instructions for downloading and setting up NVIDIA SDK Manager can be found here. **Make sure to log in with the same developer account in SDK Manager.**
5. Connect the NVIDIA IGX Orin Developer Kit to the host system via two cables: USB-C to the recovery port (7) and Micro USB Type B to the BMC access port (4). After connecting the two cables, running lsusb on your host machine should show two results with NVidia Corp:

```
…
Bus 001 Device 002: ID 0955:7020 NVidia Corp.
…
Bus 002 Device 005: ID 0955:7045 NVidia Corp.
…
```

6. Put the unit into recovery mode for flashing via BMC. From a terminal on the host machine, use the following command to find the [number] for BMC:

```
ls /dev/ttyACM*
```

Out of the four results, the smallest number is for Orin and the second smallest number is for BMC. First, enter the command `sudo minicom -D /dev/ttyACM[number]` on your host machine. Next, log in with the credentials `root`/`0penBmc`. Note the number `0`, not the letter `O`, in the password. You may need to press enter after connecting to get the login prompt.

```
$ sudo minicom -D /dev/ttyACM [number]
login: root
Password: 0penBmc
```

Output similar to the following indicates that your host machine successfully connected to the BMC:

```
NVIDIA Mandalore BMC (OpenBMC     Project Reference Distro)
nodistro.0.1643883756.1752021 mandalore ttyS0


mandalore login: root
Password:
root@mandalore:~#
```

From here, enter the command to put the unit into recovery mode:

```
root@mandalore:~# powerctrl recovery
```

If the flashing step fails in SDK Manager, make sure to re-issue this recovery mode command every time before attempting to flash, otherwise you may run into the same error message regardless of the number of attempts.

7. From the NVIDIA SDK Manager, download and flash the NVIDIA IGX Orin Developer Kit. See the step-by-step instructions for more details. After the OS image is flashed in Step 3, SDK Manager may require the NVIDIA IGX Orin Developer Kit to be connected to the Internet before installing the rest of the components.

   Note: We recommend selecting "Manual Setup" mode in the prompt at Step 3 in SDK Manager while the unit is under recovery mode, as selecting "Automatic Setup" mode may cause flashing to stall.

Connect the NVIDIA IGX Orin Developer Kit to the Internet using one of the following methods:

- An Ethernet cable connected to a router or Wi-Fi extender
- A USB Wi-Fi receiver
  - Not all USB Wi-Fi receivers will work out of the box on the NVIDIA IGX Orin Developer Kit.
  - The USB Wi-Fi receiver should support Ubuntu 20.04.
  - If the USB Wi-Fi receiver requires driver installation, use *sudo minicom -D /dev/ttyACM[n]* to access the newly flashed Jetson OS from host, or use an Ethernet cable temporarily for the duration of the USB Wi-Fi receiver setup.

Switching between iGPU and dGPU

The NVIDIA IGX Orin Developer Kit can use either the OrinAGX module GPU (iGPU, integrated GPU) or the RTX A6000 add-in card GPU (dGPU, discrete GPU). You can only use one type of GPU at a time.

By default, the NVIDIA IGX Orin Developer Kit uses the iGPU. To switch between the iGPU and dGPU, use the `nvgpuswitch.py` script located in the `/opt/nvidia/l4t-gputools/bin/` directory. To make the `nvgpuswitch.py` script accessible globally, copy it to a directory included in `$PATH` if it hasn't been already:

```
$ sudo cp /opt/nvidia/l4t-gputools/bin/nvgpuswitch.py /usr/local/bin/
```

To switch from the iGPU to the dGPU, follow these steps:

1. Ensure that the developer kit has an Internet connection.

2. To view the currently installed drivers and their version, use the `query` command:

```
$ nvgpuswitch.py query
iGPU (nvidia-l4t-cuda, 34.1.2-20220613164700)
```

3. To install the dGPU drivers, use the `install` command with the "dGPU" parameter (note that `sudo` must be used to install drivers):

```
$ sudo nvgpuswitch.py install dGPU
```

   The `install` command will print out the list of commands that will be executed as part of the driver install, then continue to execute those commands. This aids with debugging if any of the commands fail to execute for any reason. The following arguments may also be provided with the `install` command:

```
$ nvgpuswitch.py install -h

usage: nvgpuswitch.py install [-h] [-f] [-d] [-i] [-v] [-l LOG] [-r
[L4T_REPO]] {iGPU,dGPU}


positional arguments:
  {iGPU,dGPU}           install iGPU or dGPU driver stack


optional arguments:
  -h, --help            show this help message and exit
```

```
  -f, --force           force reinstallation of the specified driver
stack

  -d, --dry             do a dry run, showing the commands that would
be executed but not actually executing them

  -i, --interactive     run commands interactively (asks before running
each command)

  -v, --verbose         enable verbose output (used with --dry to
describe the commands that would be run)

  -l LOG, --log LOG     writes a log of the install to the specified
file

  -r [L4T_REPO], --l4t-repo [L4T_REPO]

                        specify the L4T apt repo (i.e. when using an
apt mirror; default is repo.download.nvidia.com/jetson)
```

4. Verify the dGPU driver install using the `query` command:

```
$ nvgpuswitch.py query
dGPU (cuda-drivers, 510.73.08-1)

NVIDIA RTX A6000, 49140 MiB
```

5. After the dGPU drivers have been installed, rebooting the system to complete the switch to the dGPU. At this point, the Ubuntu desktop will be output via DisplayPort on the dGPU (port 14), so you must switch the DP cable from the iGPU DP out (port 3) to the dGPU DP out (port 14).

   Note: If the output connection isn't switched before the devkit finishes rebooting, the terminal screen will hang during booting.

6. Modify `PATH` and `LD_LIBRARY_PATH`. CUDA installs its runtime binaries such as nvcc into its own versioned path that is not included in the default `$PATH` environment variable. Because of this, attempts to run commands like nvcc will fail on dGPU unless the CUDA 11.6 path is added to the `$PATH` variable. To add the CUDA 11.6 path for the current user, add the following lines to `$HOME/.bashrc` after the switch to dGPU:

```
export PATH=/usr/local/cuda-11.6/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/cuda-11.6/lib64:$LD_LIBRARY_PATH
```

At this time, the Clara Holoscan SDK is tested and supported only in dGPU mode. Switching back to iGPU mode after switching to dGPU mode is not recommended.

Note: The GPU settings will persist through reboots until it is changed again with *nvgpuswitch.py*.

Setting up SSD Storage

If you don't set up SSDK storage and move docker storage to SSD, you will likely quickly fill up the root directory with Docker image pull operations since a complete installation of Holopack leaves about 40GB of storage remaining in the root 64GB.

The NVIDIA IGX Orin Developer Kit includes a pre-installed 500GB m.2 solid-state drive (SSD), but this drive is not partitioned or mounted by default. This page outlines the steps to partition and format the drive for use after the initial SDK installation.

> **Note:** If the NVIDIA IGX Orin Developer Kit is re-flashed with a new Holopack image, the partition table of the m.2 drive will not be modified and the contents of the partition will be retained. In this case, you can skip the Create Partition steps; however, you should still follow the Mount Partition steps to remount the partition.
>
> Any state, binaries, or docker images that persist on the m.2 drive after flashing the system may be incompatible with new libraries or components that are flashed onto the system. You may need to recompile or rebuild these persistent objects to restore runtime compatibility with the system.
>
> **Note:** The following steps assume that the m.2 drive is identified by the NVIDIA IGX Orin Developer Kit as `/dev/nvme0n1`. This is the case if no additional drives have been attached, but if other drives (such as USB drives) have been attached, then the disk identifier may change. You can verify this by looking at the symlink to the drive that is created for the m.2 hardware address on the system. If the symlink below shows something other than `../../nvme0n1`, replace all instances of "nvme0n1" in the instruction below with the identifier that is being used by your system:

```
$ ls -l /dev/disk/by-path/platform-14160000.pcie-pci-0004\:01\:00.0-nvme-1

lrwxrwxrwx 1 root root 13 Jun  2 14:14 /dev/disk/by-path/platform-
14160000.pcie-pci-0004:01:00.0-nvme-1 -> ../../nvme0n1
```

Create the Partition

1. Launch fdisk utility:

```
$ sudo fdisk /dev/nvme0n1
```

2. Create a new primary partition using the "n" command, then accept the defaults by pressing Enter for the next 4 questions. This will create a single partition that uses the entire drive.

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (1-4, default 1):
First sector (2048-976773167, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-976773167, default
976773167):

Created a new partition 1 of type 'Linux' and of size 465.8 GiB.
```

3. Write the new partition table and exit using the “w” command:

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

4. Initialize the ext4 filesystem on the new partition. Enter “y” when prompted:

```
$ sudo mkfs -t ext4 /dev/nvme0n1
mke2fs 1.45.5 (07-Jan-2020)
Found a dos partition table in /dev/nvme0n1
Proceed anyway? (y,N) y
Discarding device blocks: done
Creating filesystem with 122096646 4k blocks and 30531584 inodes
Filesystem UUID: 004332a8-b255-4156-836c-7ea734cb78c0
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
2654208,
        4096000, 7962624, 11239424, 20480000, 23887872, 71663616,
78675968,
        102400000

Allocating group tables: done
Writing inode tables: done
```

```
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information: done
```

Mount the Partition

1. Create a directory for the mount point. These instructions will use the path `/media/m2`, but any path can be used.

```
$ sudo mkdir /media/m2
```

2. Determine the UUID of the new partition. The UUID will be displayed as a symlink to the `/dev/nvme0n1` partition within the `/dev/disk/by-uuid` directory. For example, the following output shows that the UUID of the `/dev/nvme0n1` partition is 004332a8-b255-4156-836c-7ea734cb78c0:

```
$  ls -l /dev/disk/by-uuid/ | grep nvme
lrwxrwxrwx 1 root root 13 Jun  2 15:13 004332a8-b255-4156-836c-
7ea734cb78c0 -> ../../nvme0n1
```

3. Add the fstab entry. Using the mount path and the UUID from the previous steps, add the following line to the end of `/etc/fstab`:

```
UUID=004332a8-b255-4156-836c-7ea734cb78c0 /media/m2 ext4 defaults 0 2
```

4. Mount the partition. The `/etc/fstab` entry above will mount the partition automatically at boot time. To instead mount the partition immediately without rebooting, use the `mount` command (and `df` to verify the mount):

```
$ sudo mount -a
$ df -h /dev/nvme0n1
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme0n1    458G   73M  435G   1% /media/m2
```

5. **Manage permission on SSD.** Use the “chmod” command to manage file system access permission. For example:

```
$ sudo chmod -R 777 /media/m2
```

Setting up Docker and Docker Storage on SSD

1. Install Docker if it has not been installed on your system:

```
$ sudo apt-get update
$ sudo apt-get install -y docker.io
```

2. Create a Docker data directory on the new m.2 SSD partition. This is where Docker will store all of its data, including build cache and container images. These instructions use the path `/media/m2/docker-data`, but you can use another directory name if preferred.

```
$ sudo mkdir /media/m2/docker-data
```

3. Configure Docker by writing the following to `/etc/docker/daemon.json`:

```
{
    "runtimes": {
        "nvidia": {
            "path": "/usr/bin/nvidia-container-runtime",
            "runtimeArgs": []
        }
    },
    "default-runtime": "nvidia",
   "data-root": "/media/m2/docker-data"
}
```

4. Restart the Docker daemon:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

5. Add the current user to the Docker group so Docker commands can run without `sudo`.

```
# Create the docker group.
$ sudo groupadd docker
# Add your user to the docker group.
$ sudo usermod -aG docker $USER
```

```
# Activate the changes to groups. Alternatively, reboot or re-login.
$ newgrp docker
```

6. Verify that you can run a hello world container:

```
$ docker run hello-world
```

Manually Updating Firmware

Manually updating the firmware on the devkit will automatically disable ACS and no longer require running `sudo setpci -s 0007:02:00.0 ecap_acs+6.w=0` after every reboot.

1. Download the firmware file from here.
2. Connect the Developer Kit to a host machine via port 4 Micro USB Type B with connection to BMC
3. Run command from host where [number] corresponds to the number for BMC:

```
sudo minicom -w -D /dev/ttyACM[number] -b 230400
```

4. PCIe switch console should open. Check your firmware version by running command `version`. If the output shows as below, your firmware version had already been updated and you can skip the rest of this section

```
0x00000000:0000>gasrd 0x2104 1
gas_reg_read <0x2104> [1]
0x40011001
```

   Otherwise if you see an output with a lower version as below, continue.

```
0x00000000:0000>gasrd 0x2104 1
gas_reg_read <0x2104> [1]
0x40001001
```

5. Run command `fw_update` and you should see something as below

```
Welcome to minicom 2.7.1


OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyACM2, 17:35:39


Press CTRL-A Z for help on special keys
```

```
0x00000000:0000>fw_update


Please select file to be transferred in xmodem option of terminal
tool, press [CTRL+C] to cancel...CC
```

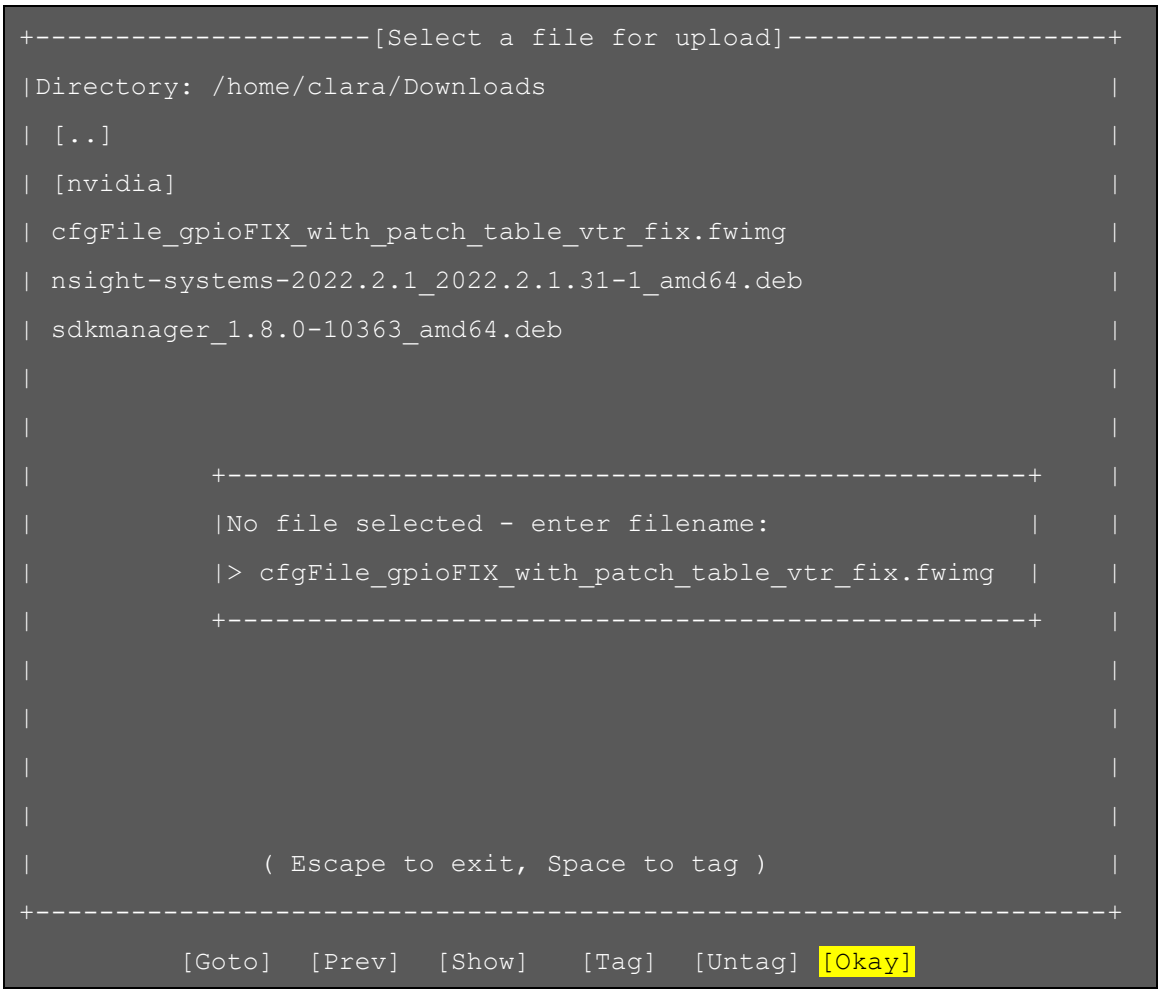
6. Press CTRL+A then S. Select “xmodem” from the window

```
                            +-[Upload]--+
Press CTRL-A Z for help on sp| zmodem    |
                            | ymodem    |
                            | xmodem    |
0x00000000:0000>fw_update   | kermit    |
                            | ascii     |
Please select file to be tran+----------+odem option of terminal
tool, press [CTRL+C] to cancel...CCCCCCC
```

7. Select “Go to” at the bottom and enter the directory where the FW files are saved. Then press enter.

```
+---------------------[Select a file for upload]---------------------+
|Directory: /home/clara                                              |
| [..]                                                               |
| [.cache]                                                           |
| [.config]                                                          |
| [.gnupg]                                                           |
| [.local]                                                           |
| [.mozilla]                                                         |
| [.nsightsystems]                                                   |
| [.nvsdkm]      +---------------------------------------+           |
| [.pki]         |Goto directory:                        |           |
| [.ssh]         |> /home/clara/Downloads                |           |
| [Desktop]      +---------------------------------------+           |
| [Documents]                                                        |
| [Downloads]                                                        |
| [Music]                                                            |
| [Pictures]                                                         |
|              ( Escape to exit, Space to tag )                      |
+--------------------------------------------------------------------+
          [Goto]  [Prev]  [Show]   [Tag]  [Untag] [Okay]
```

8. Within the directory you can see the FW file. Select the FW file



9. Press Enter. The FW should start uploading. You should see a “Transfer complete” message in the end. Press any key to continue.

```
      |                                                                 |
      | READY: press any key to continue...                             |
      +-----------------------------------------------------------------+
```

10. You should see a “File transfer successfully” message and finish.

```
Welcome to minicom 2.7.1

OPTIONS: I18n

Compiled on Aug 13 2017, 15:25:34.

Port /dev/ttyACM2, 17:35:39


Press CTRL-A Z for help on special keys


0x00000000:0000>fw_update

Please select file to be transferred in xmodem option of terminal
tool, press [CTRL+C] to cancel...CCCCCCC

File transfer successfully!

0x00000000:0001>
```

11. Shut down your devkit and restart. A simple reboot may not suffice. Check again the version of your firmware, it should look like below.

```
0x00000000:0000>gasrd 0x2104 1

gas_reg_read <0x2104> [1]

0x40011001
```

Install the Clara Holoscan SDK

The Clara Holoscan SDK is hosted on Github starting from v0.2. See https://github.com/nvidia/clara-holoscan-embedded-sdk for information on installing the Clara Holoscan Embedded SDK.

Known Issues and Troubleshooting

1. RDMA known issue and workaround

There's a known issue that prevents GPU RDMA from being enabled on the NVIDIA IGX Orin Developer Kit. We recommend a manually-applied firmware update that addresses this known. For how to manually update the firmware, please see the section **Manually Updating Firmware**. Without applying the firmware update, the temporary workaround is to run the following command after every reboot to disable ACS:

```
$ sudo setpci -s 0007:02:00.0 ecap_acs+6.w=0
```

2. Automatic Setup hangs during the flashing process

When flashing the devkit using SDK Manager, at the dialog prompt where it says "SDK Manager is about to flash your NVIDIA IGX Orin Developer Kit module" in Step 3, it has been observed that if you choose Automatic Setup, even if your Developer Kit had been flashed before, the SDK Manager UI can hang at this step.

Action: Put your Developer Kit into recovery mode following the steps in the **Flashing and Updating the NVIDIA IGX Orin Developer Kit using SDK Manager** section and choose "Manual Setup" in Step 3 of the SDK Manager flashing process.

3. Attempting to switch to dGPU mode fails, and the system is not in iGPU or dGPU mode

If the `nvgpuswitch.py` script for installing dGPU fails for any reason, the system will not default back to the previous iGPU mode; therefore, the system doesn't have either of the GPU modes enabled.

Action: When you are ready to try again, first check that the `nvgpuswitch.py` script is still in your `$PATH`; if it is missing, find the location of the script and copy it to `$PATH`:

```
$ sudo find / -name nvgpuswitch.py
/opt/nvidia/l4t-gputools/bin/nvgpuswitch.py
```

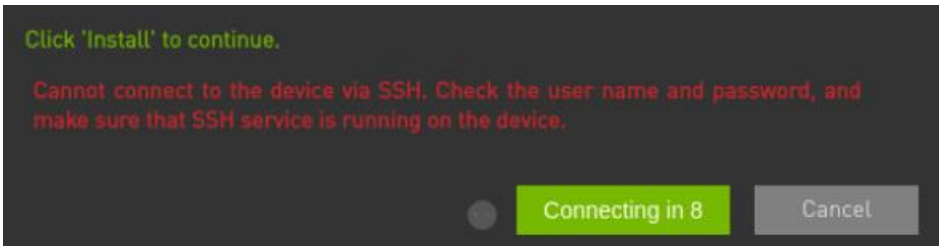
```
$ sudo cp /opt/nvidia/l4t-gputools/bin/nvgpuswitch.py /usr/local/bin/
```

Then, use the `-f` option when running `nvgpuswitch.py` to force the reinstall of the dGPU stack.

```
$ sudo nvgpuswitch.py install dGPU -f
```

4. Error “Cannot connect to the device via SSH” during the flashing stage in “SDK Manager is about to install SDK components on your IGX Orin Developer Kit module”

This is due to the developer kit not having fully booted yet; wait for the OS login screen as prompted before attempting to proceed.



Additional Resources

For other documentation and release notes, see the Clara Holoscan SDK page.

For further Jetson documentation, see the L4T documentation.

For feedback, discussion, and questions, post to the Clara Holoscan SDK Developer Forum.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Arm

Arm, AMBA and Arm Powered are registered trademarks of Arm Limited. Cortex, MPCore and Mali are trademarks of Arm Limited. All other brands or product names are the property of their respective holders. ”Arm” is used to represent Arm Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited.; Arm Taiwan Limited; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS and Arm Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA, the NVIDIA logo, Jetson AGX Xavier, and Quadro are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com