DRIVE Platform Installation with NVIDIA SDK Manager

Installation Guide
NVIDIA DRIVE OS 5.2.0

Table of Contents

1 Introduction

What is DRIVE AGX? What is DRIVE OS? Common terminology explained here.

1.1 NVIDIA DRIVE

Scalable AI Platform for Autonomous Driving

Autonomous vehicles will transform the way we live, work, and play, creating safer and more efficient roads. To realize these revolutionary benefits, the car of the future will require a massive amount of computational horsepower. Tapping into decades-long experience in AI, NVIDIA DRIVE™ hardware and software solutions deliver industry-leading performance to help automakers, truck makers, tier 1 suppliers, and startups make autonomous driving a reality.

NVIDIA DRIVE™ platforms include an in-vehicle computer (DRIVE AGX) and complete reference architecture (DRIVE Hyperion™), as well as data center-hosted simulation (DRIVE Constellation™) and deep neural network (DNN) training platforms (DGX™). These platforms also include rich software developer kits (SDKs) to accelerate autonomous vehicle (AV) development.

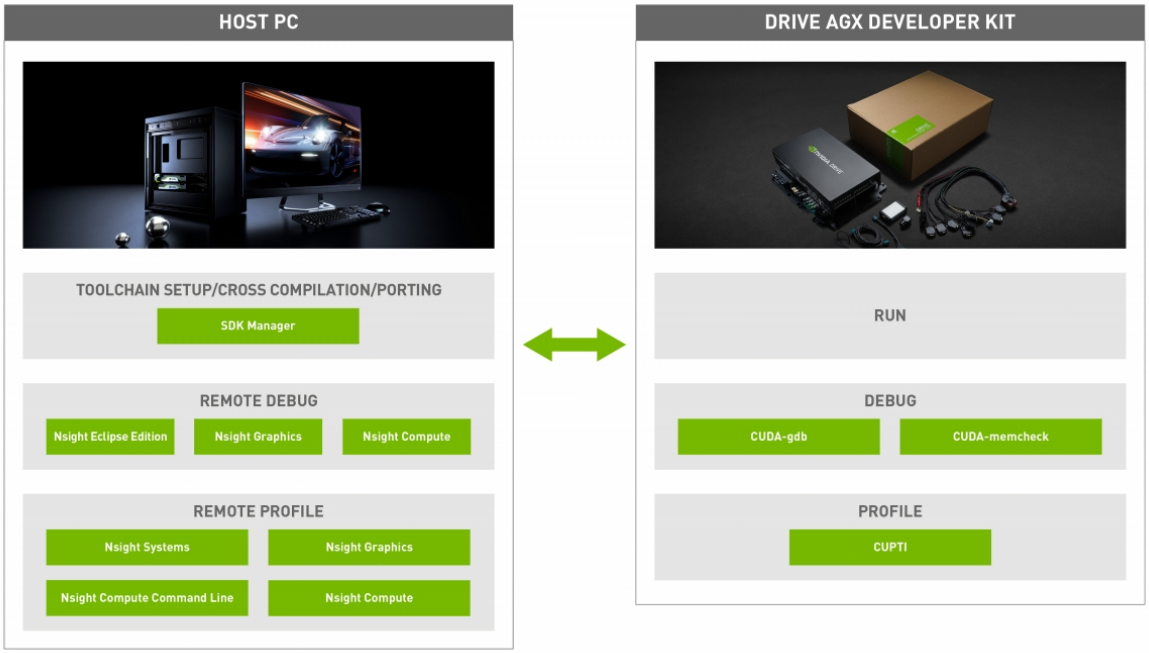
To learn more about the NVIDIA DRIVE platform, see:

https://developer.nvidia.com/drive

1.2 NVIDIA DRIVE AGX Developer Kit

The NVIDIA DRIVE™ AGX Developer Kit provides the hardware, software, and sample applications needed for development of production level autonomous vehicles (AV). The NVIDIA DRIVE AGX System is built on production auto-grade silicon, features an open software framework, and has a large ecosystem of automotive partners (include auto-grade sensor vendors, automotive Tier 1 suppliers) to choose from.

The development environment for DRIVE AGX requires a host development PC (not included with the DRIVE AGX Developer Kit). The image below illustrates the development workflow. All the tools illustrated below are installed on the host PC through NVIDIA SDK Manager.

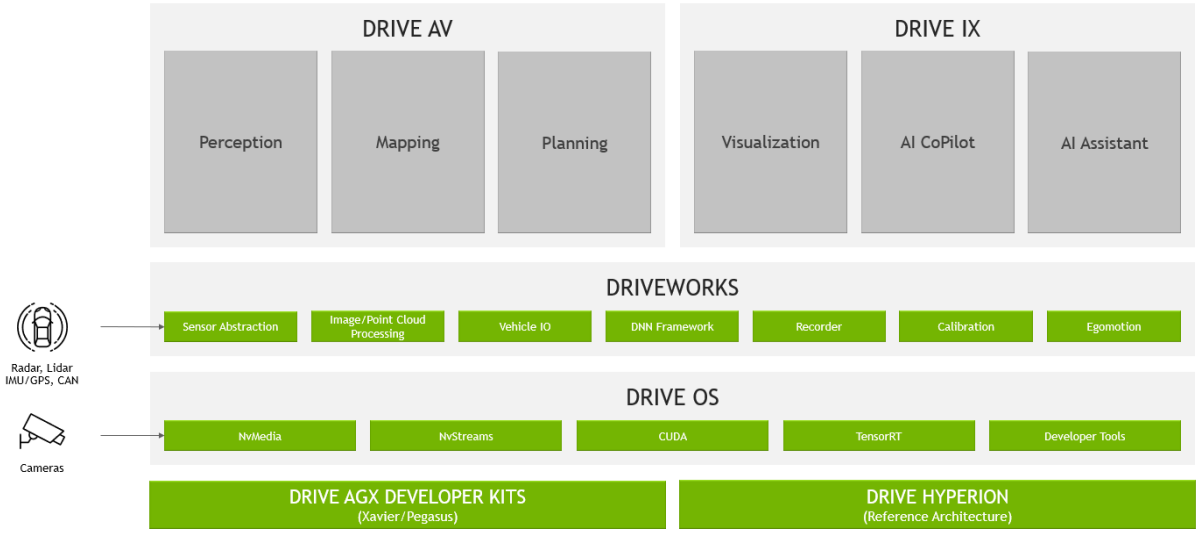


To learn more about the DRIVE AGX Developer Kit, see:

https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/

1.3 DRIVE OS

NVIDIA DRIVE™ OS is the reference operating system and associated software stack designed specifically for developing and deploying autonomous applications on DRIVE AGX-based hardware. NVIDIA DRIVE OS delivers a safe and secure execution environment for safety-critical applications, providing services such as secure boot, security services, firewall, and over-the-air updates. The included foundational software stack consists of a Type-1 Hypervisor, NVIDIA® CUDA® libraries, NVIDIA TensorRT™, NvMedia, and other components optimized to provide direct access to DRIVE AGX hardware acceleration engines.



1.4 DriveWorks

NVIDIA® DriveWorks SDK is middleware for autonomous vehicle software development. Included in each release are a collection of modules for common AV software use cases, such as interfacing with the vehicle's sensors, processing sensor data to be used as input to perception algorithms, and performing DNN inference on pre-trained networks. Also included are a set of tools that support AV software development, such as sensor data recording, sensor calibration, and DNN optimization. DriveWorks abstracts away the details of the underlying DRIVE AGX platform while still exposing its power, thereby reducing the time to develop complex and accelerated AV applications.

To learn more about DriveWorks, see:

https://developer.nvidia.com/drive/driveworks

2 Requirements for Your Development Environment

2.1 Getting Started

Follow these high-level steps in order to set up your development environment – host machine and target DRIVE AGX Developer Kit:

1. Verify you meet the system requirements for installing software on your DRIVE AGX Developer Kit.
2. Review the Important Installation Information section.
3. Review other important documentation, such as:
    - DRIVE OS 5.2.0 Release Notes
4. Download SDK Manager.
5. Finally, review the SDK Manager installation section of this guide to install DRIVE Software or DRIVE OS on your host machine, and flash the target device.

2.1.1 System Requirements

Category	Requirement
Host Machine	• Ubuntu Desktop 18.04 LTS • Working Internet connection
Architecture	X86_64
Memory	8GB
Free Disk Space	A minimum of 40GB and up to 120GB (during flash) free disk space on the system volume is needed for each full (host and target) deployed SDK version.
GUI	X11 must be enabled on the host.

<table>
<tr><th>Category</th><th>Requirement</th></tr>
<tr><td>Graphics Driver</td><td>The most recent graphics driver for your GPU needs to be installed on the host system. To update your driver, use one of the following methods:
Download the .run file for your particular GPU and OS.
OR
Use the apt-get method.
WARNING: Do not mix both the download and apt-get methods, as this is not supported.
The CUDA version and minimum Linux x86_64 graphics driver for the DRIVE OS 5.2.0 release is:<table><tr><th>CUDA Toolkit</th><th>Linux x86_64 Driver Version</th></tr><tr><td>CUDA 10.2.187</td><td>>=450.51.06</td></tr></table></td></tr>
<tr><td>Target Device</td><td>The following supported development platform:
NVIDIA DRIVE AGX Developer Kit</td></tr>
<tr><td>Additional Hardware</td><td>A2A USB cable to connect the host machine to the target</td></tr>
</table>

2.2 Important Installation Information

We care about the safety and security of your data; therefore, you will be prompted to enter a new Linux username and password during the installation process.

For DRIVE OS users (both with and without DriveWorks), the DRIVE AGX Developer Kit will prompt you via the target Linux console to enter a new Linux username and password on the first boot after flashing.

MCU Power Good LED Notification

The Aurix MCU controls the board power-up sequence on the DRIVE AGX Developer Kit. An error was found in some power devices for 5V rails on the DRIVE AGX board such that they may not provide proper voltage output if enabled when voltage is in a certain range. To work around this, the Aurix MCU FW in the release implements a feature to monitor these voltage rails and not power up these rails until the existing voltage is in the proper range. However, this will increase the time required for the board to boot or for the board or Xavier devices to reset (i.e., `aurixreset` or `tegrareset` commands) to ~14 sec in the average case. In addition, if voltage rails never fall to a valid level after 30 seconds, then the rails will be powered up and the board will boot, but board functions may be affected.

Because of this, the Aurix MCU indicates the board status using one LED located next to the debug USB port. For healthy boot with no error, the LED will blink once every four seconds (i.e., Power Good indication); for error timeout, the LED will blink three

times rapidly every four seconds (i.e., Power Bad indication). More information on this is given in the Development Guide section "MCU LED Notification."

In order to verify that networking is properly configured for the DRIVE AGX Developer Kit, please review the DRIVE AGX board setup information found here.

Please review the DRIVE AGX Developer Kit Flashing Technical Bulletin [login required] for known flashing issues, resolutions, and/or workarounds.

Also note the Additional Resources section of this document, which provides links to useful information for configuring your NVIDIA DRIVE AGX Developer Kit.

2.2.1 Valid Upgrade and Downgrade Paths

If you are installing NVIDIA DRIVE OS on DRIVE AGX Xavier or DRIVE AGX Pegasus System for the first time, please skip to the Download and Run SDK Manager section.

If you previously installed a different version of DRIVE OS or DRIVE Software, this section provides important information about supported migration paths between DRIVE OS (with and without DriveWorks) and DRIVE Software releases.

Each release of NVIDIA DRIVE OS provides increasing levels of capability and reliability. Additionally, each release provides slightly different features that may be more suitable for your application. In most cases, you will install newer DRIVE OS versions as they are released. However, in some cases, you may decide to install an earlier release.

The tables below describe the supported DRIVE OS and DRIVE Software migration paths. For example, before you install another product version over another product version, please ensure that migration path is supported. If a desired migration path is unsupported, you may need to install another product version first before installing the desired product version.

The specific order or product versions allows the installer to address product version differences, such as VBIOS, PCIe, and Aurix.

For information on the features added in each release, ask your NVIDIA representative for the release notes for DRIVE OS or DRIVE Software for the releases under consideration.

2.2.1.1 Update Matrix: DRIVE OS and DRIVE Software

To update to DRIVE OS 5.2.0, you must have DRIVE OS 5.1.15.0 or DRIVE OS 5.1.15 with DriveWorks installed on your host development environment and flashed on your target DRIVE AGX System. Once your host development environment and DRIVE AGX System are updated to DRIVE OS 5.2.0, you may only downgrade to DRIVE OS 5.1.15 or DRIVE Software 10.0 with specified caveats.

DRIVE Software 10.0	**DRIVE OS 5.1.12**	**DRIVE OS 5.1.15**	**DRIVE OS 5.1.15 with DriveWorks 3.0**	**DRIVE OS 5.2.0**	**To:**
N/A	See Note #1	NO	YES	NO	DRIVE Software 10.0
See Note #1	N/A	YES	NO	NO	DRIVE OS 5.1.12
NO	YES	N/A	YES	YES	DRIVE OS 5.1.15
YES	NO	YES	N/A	YES	DRIVE OS 5.1.15 with DriveWorks 3.0
NO	NO	YES	YES	N/A	DRIVE OS 5.2.0

YES	Supported
NO	Not supported
Note #1	Not directly supported. Contact your NVIDIA representative for details.

3 Setting Up DRIVE OS QNX

To use NVIDIA SDK Manager to set up a DRIVE OS QNX development environment, you need to ensure that your system meets certain requirements, and configure QNX as outlined below.

3.1 QNX SDP Installation Instructions for DRIVE QNX

As part of the NVIDIA DRIVE OS QNX release, an installation of the QNX Software Development Platform (SDP) is required. The QNX SDP release is published by QNX for download in the QNX Software Center (QSC). Use the following section to configure QNX SDP 7.0 to be compatible with DRIVE OS QNX.

While installing the QNX SDP, QSC may display a message indicating that it cannot complete the installation because one or more required items could not be found. This may indicate that your company needs permission to additional QNX SDP packages specific to NVIDIA. Please notify your NVIDIA Customer Program Manager and Technical Support Engineer, providing your corporate email domain, to request permission for these items.

The QNX SDP contains symbolic links to some files. It is necessary to preserve all links for use on a DRIVE QNX target.

To begin, you need to use the QNX Software Center application, which at the time of this writing, can be downloaded from:

http://www.qnx.com/download/group.html?programid=29178

From there, you need to download the Linux Host version of the QNX Software Center application. Note that you must be a registered QNX SDP 7.0 user with a myQNX account to download the QNX Software Center.

At the time of this writing, the QNX Software Center 1.6.1: Installation Note can be found at:

http://www.qnx.com/developers/articles/inst_6800_1.html

For the most up-to-date version of the QNX Software Center Installation Note:

1. Go to the QNX website, www.qnx.com, and log in to your myQNX account.
2. Select the **Developers** tab at the top of the page, and click the **QNX Software Center** link.
3. Scroll down the page, and click on the Linux Host link for **See Installation/Release notes**.

Refer to the user guide at:

http://www.qnx.com/download/download/43710/qnx_qsc_user_guide_2019-06-18.pdf

3.2 Prerequisites

- myQNX user account issued from QNX
- Activated QNX SDP license keys issued from QNX

DRIVE OS 5.2.0 Requires QNX OS for Safety for Development

In addition to the QNX SDP license required for development, this release requires a QNX OS for Safety (QOS) project license. Please contact your NVIDIA Representative for QNX QOS Licensing information.

3.3 Import Offline Package

Follow these instructions to install the QNX SDP that is compatible with DRIVE QNX.

The DRIVE OS QNX SDK includes a QNX patch set file generated by using the Export feature of the QNX Software Center. The following instructions describe how to Import this file to install the QNX SDP.

The QNX Software Center lets you export and import patch sets. A patch set is like a recipe for installing packages. It tells the QNX Software Center not only which packages to install, but also which version of each package to install. QNX patch sets are explained in the Advanced Topics section of the QNX Software Center User Guide.

It is recommended that you remove `${HOME}/.qnx/swupdate/dropins/` prior to importing packages.

1. In the QNX Software Center application, navigate to the Welcome to the QNX Software Center panel by clicking on the "home" icon that looks like a house.
2. On the Welcome to the QNX Software Center panel, click on Import Offline Package.
3. This opens the File Import Wizard.
    a. In Select File: enter the full path to the QNX patch set file included in the DRIVE QNX SDK. The filename is:

```
drive-t186ref-qnx-5.2.0.0-sdp-patchset.qpkg
```

b. Click the **Add new installation** radio button.

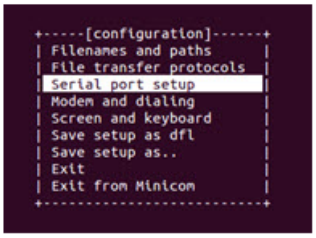
c. Click the **Finish** button.

4. This will open the **New Installation Wizard** where you set installation properties.
   a. Set the **Installation Folder:** and **Name:** fields to your desired location and name.
   b. Set **Update Policy:** to the Conservative option.
   c. To install debug symbols, enable the **Install debug symbols** checkbox.
   d. Select the **Install experimental packages** checkbox.
   e. Ensure only the aarch64le checkbox is selected for **Target Architectures**.
   f. Click **Next>**.
5. This will open the **Install** window where you check the items that you wish to install.
   a. Ensure the checkboxes are selected for all packages.
   b. Click **Next>**.
6. Review Packages
   a. Review package names and versions.
   b. Click **Next>** again.
7. License Key Selection
   a. Select the appropriate license key.
   b. Click **Finish**.

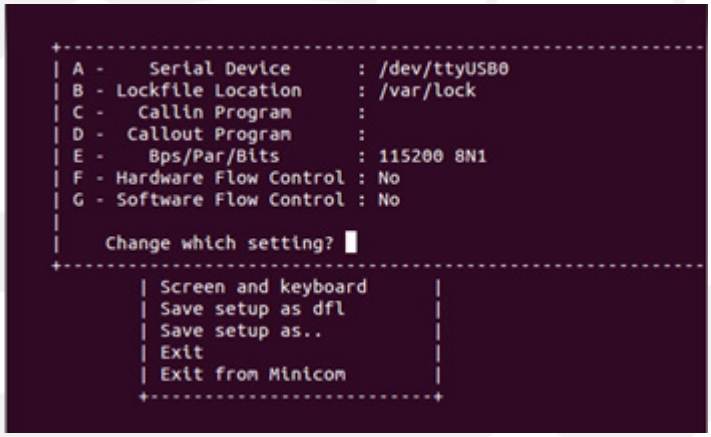
3.4 Set Up and Configure Minicom

If you do not configure minicom automatically through the SDK Manager installer, you can set it up manually by using the following steps.

1. From a terminal window on the Linux host, check if minicom is installed by executing the command:
   ```
   minicom -s
   ```
2. If Minicom is NOT installed, install by executing the command:
   ```
   sudo apt-get install minicom
   ```
3. Configure minicom on the Linux host by once again executing the command:
   ```
   sudo minicom -s
   ```
   A configuration dialog displays.
4. From the configuration dialog, select **Serial Port Setup**.



5. Define the configuration as follows:
    - Serial Device: /dev/ttyUSB2
    - Lockfile location: /var/lock
    - Bps/Par/Bits: 115200 8N1
    - Hardware Flow Control: No
    - Software Flow Control: No



    - /dev/ttyUSB2 maps to Xavier A
    - /dev/ttyUSB3 maps to AURIX
    - /dev/ttyUSB6 maps to Xavier B
6. Press **Enter** and click on **Save setup as dfl**, then select **Exit**.

Upon successful completion, a prompt window displays for the target system.

4 Download and Run SDK Manager

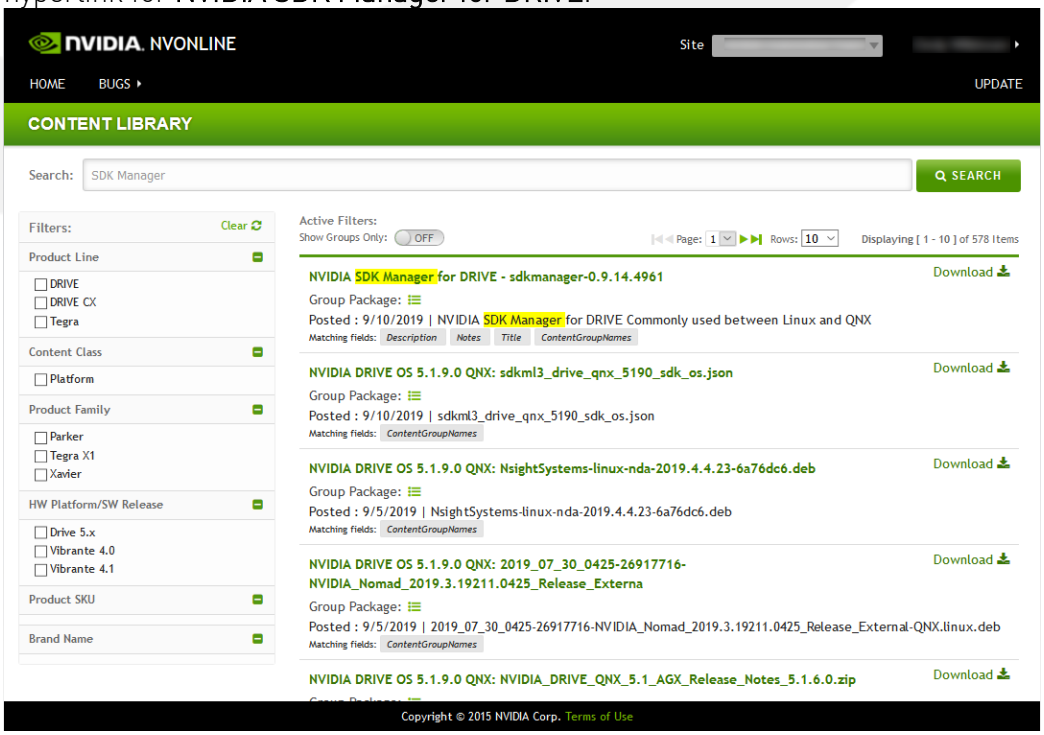
NVIDIA SDK Manager provides an end-to-end development environment setup solution for both the host machine and target Development Kits (DRIVE AGX and Jetson).

Instructions for downloading and running SDK Manager from NVONLINE are detailed below.

4.1 Download via NVONLINE

The following instructions are for NVONLINE users.

1. Log in to partners.nvidia.com. If it is your first time logging in, you must accept the NVONLINE CONDITIONS OF USE agreement before proceeding.
2. To locate the SDK Manager package, use one of the following methods:
    a. In the search field, type "SDK Manager" and click **Search**. Locate and click the hyperlink for **NVIDIA SDK Manager for DRIVE**.



b. Scroll through the list of available packages. Locate and click the hyperlink for NVIDIA SDK Manager for DRIVE.

3. A new tab opens with the SDK Manager installer to download, `sdkmanager-[version].[build#].deb`, where `[version]` and `[build]` represent the current version number and build number of SDK Manager.
4. Download the SDK Manager `.deb` file to your host machine.

4.2 Install the SDK Manager Package

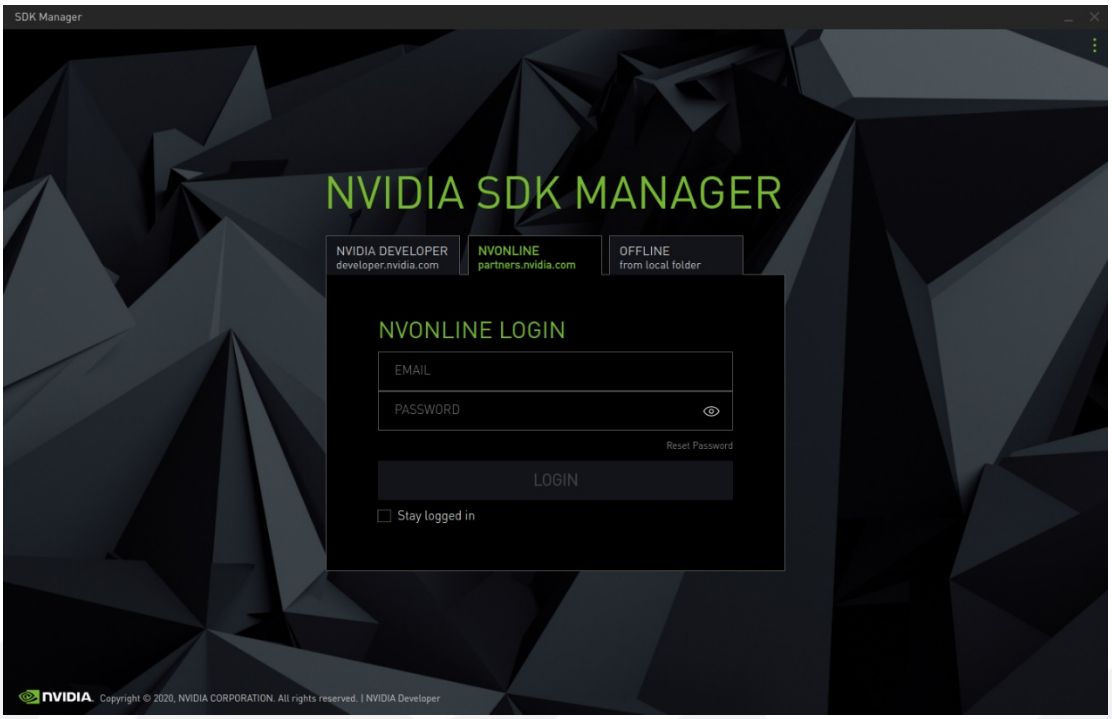
Once you have downloaded the SDK Manager .deb file to your host machine, do the following.

1. From a terminal, install the Debian package:
   ```
   sudo apt install ./sdkmanager-[version].[build#].deb
   ```
2. Next, you can start SDK Manager using one of the following two methods:
   a. Launch SDK Manager from the Ubuntu launcher.
   b. Open a terminal and launch SDK Manager with the following command:
      ```
      sdkmanager
      ```
   c. SDK Manager also supports a command line interface. To see the options, run:
      ```
      sdkmanager --help
      ```
      To learn more, see Command Line Install.

4.3 Login to SDK Manager

1. From the SDK Manager launch screen, select the appropriate login tab for your account type and installation.
   - NVIDIA DRIVE Developer Program — developer.nvidia.com
   - NVONLINE — partners.nvidia.com
   - Offline — to install SDKs that were previously downloaded, and are available from a local folder or mounted drive. For more information, see Offline Install.

   The default login tab is for NVIDIA Developer.
2. Enter the credentials for your account type, and click Login.



3. Before proceeding, choose whether or not to enable data collection.

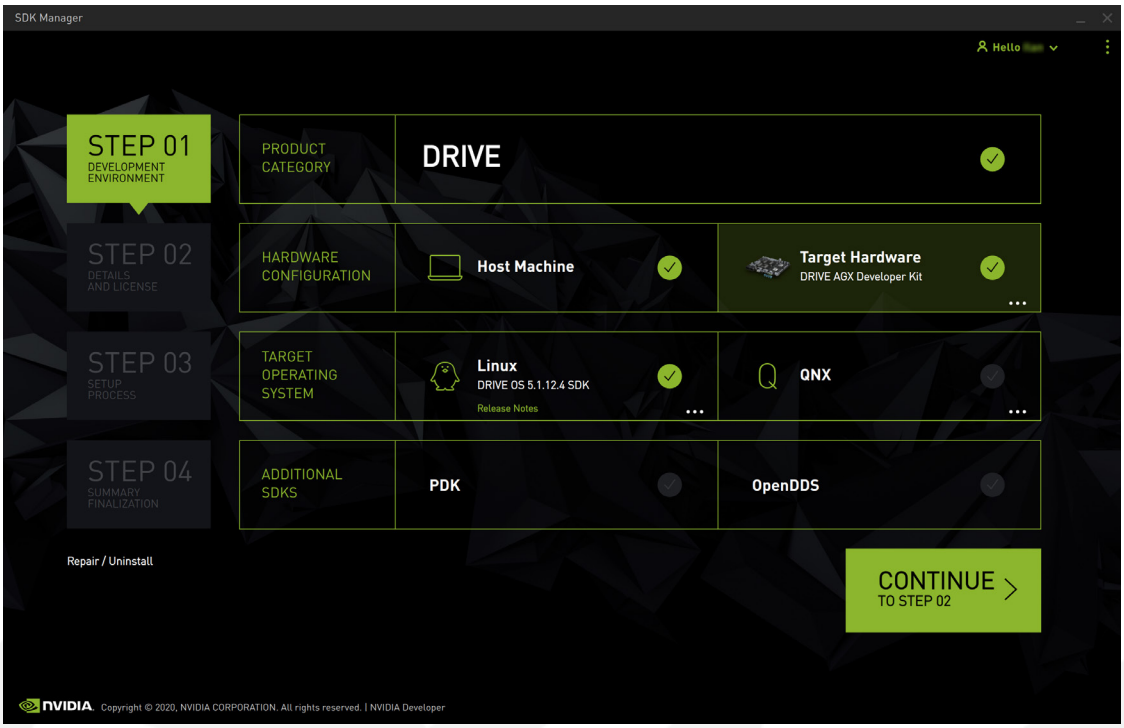
5 Install DRIVE Platform with SDK Manager

This section is intended to help you use the NVIDIA SDK Manager GUI to successfully configure your development environment.

5.1 Step 1: Setup the Development Environment

1. From the **Step 01 Development Environment** window, select the following:
    - From the **Product Category** panel, select the DRIVE development environment.
    - From the **Hardware Configuration** panel, select the host machine and target hardware.
    - From the **Target Operating System** panel, select the desired operating system, such as Linux or QNX. Notice that the target operating systems available may change, depending on the options that were selected in the other panels.
    - If relevant, select any **Additional SDKs** that you wish to install.

    An ellipsis (...) in the bottom right corner of a category box indicates that more than one option is available. Click on the ellipsis to show a drop-down menu of available options.

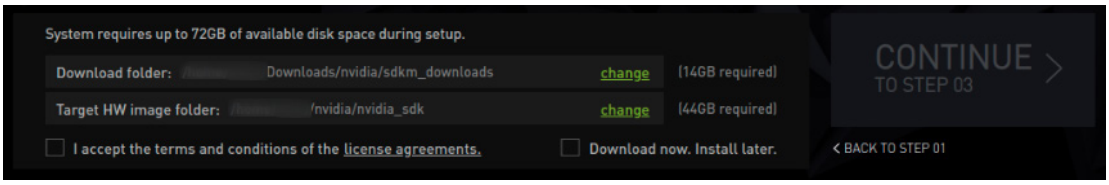


Your display may differ from the one shown here. The information in this screen is populated from your NVIDIA user account access and permissions. If you don't see your product category in the available selections, you may need to verify that your NVIDIA account is registered to the required programs.

2. Click **Continue** to proceed to the next step.

5.2 Step 2: Review Components and Accept Licenses

1. From **Step 02 Details and License**, you can expand the host components and target components panels to review the components that will be installed on your system.
2. To review the licenses, click on the **license agreements** hyperlink at the bottom of the page.
3. Enable the checkbox to accept the terms and conditions of the license agreements.
4. If you want SDK Manager to download all setup files to a location other than the default path, locate the **Download & Install Options** at the bottom of the screen, then select the path you wish to use.

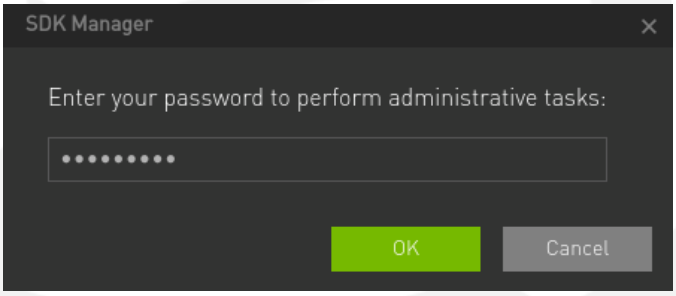


For more information about the Download & Install Options, see Offline Install.

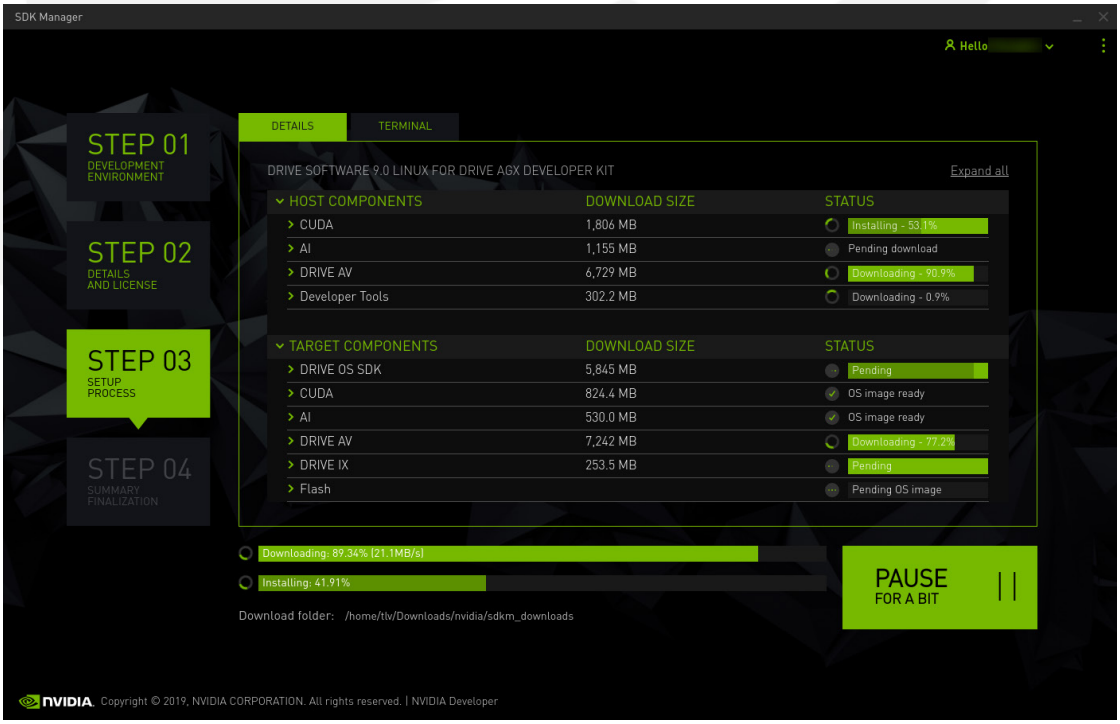
5. Select **Continue** to proceed to the next step.

5.3 Step 3: Installation

1. Before the installation begins, SDK Manager prompts you to enter your sudo password.



2. The display shows the progress of the download and installation of the software.



Select **Pause** / **Resume** to toggle the download and installation process.

3. At the top, you can toggle between the **Details** and **Terminal** tabs. The Terminal tab shows detailed information about the download and installation, with any errors highlighted.
4. On the Terminal tab, you can use the **Filter text** field to filter and search for specific information.
5. SDK Manager opens a dialog when it is ready to flash your target device. A prompt provides instructions for preparing your device to get it ready for flashing.

The instructions in the flashing dialog vary based on your host and target environment settings.

Pop-up Windows on the Linux Host During Target Flashing

When flashing the DRIVE AGX platform, different windows may pop up on the host. This can be seen on all DRIVE Software and DRIVE OS with DriveWorks releases. These pop-up windows are harmless and do not affect flashing of the unit. However, they can be managed as follows:

- For the **"Unable to Mount Functions Gadget ADB"** pop-up window:

  This message is a known bug on the Ubuntu host [200133277] - https://bugs.launchpad.net/ubuntu/+source/gvfs/+bug/1314556

  Workaround

  The recommended workaround is to kill the gvfs-mtp-volume-monitor before attempting to flash:

  ```
  $ killall gvfs-mtp-volume-monitor
  ```

- For the **"'_TEMP_DUMP' Folder Pops Up During Flashing"** pop-up window:

  While flashing, a pop-up window may appear titled "..<install location>flashtools/bootburn/_temp_dump folder" [200447039].

  Workaround

  The recommended workaround is to disable the "automount-open" option for Desktop media-handling settings in the Ubuntu host dconf database. Perform the following steps before flashing:

  1. Download the dconf-editor tool – from a terminal window.

     ```
     sudo apt-get install dconf-editor
     ```

  2. Enter the following command:

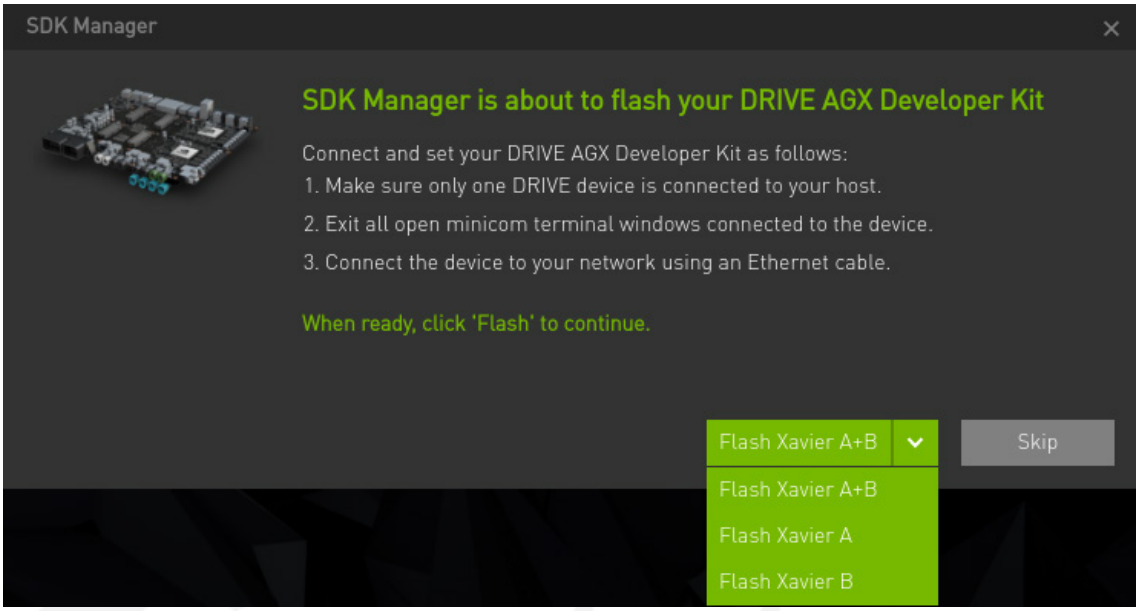
     ```
     gsettings set org.gnome.desktop.media-handling
     automount-open false
     ```

  If you wish to manually disable the "automount-open" option, you can also perform the following:

  1. Download the dconf-editor tool – from a terminal window.

     ```
     sudo apt-get install dconf-editor
     ```

  2. Select "org -> gnome -> desktop -> media-handling" in the tree.
  3. Set "automount-open" to "false" or uncheck it (depending on the Ubuntu and tool version).



For **DRIVE OS users (both with and without DriveWorks)**, the DRIVE AGX Developer Kit will prompt you on the platform console to enter a new Linux username and password on the first boot after flashing.

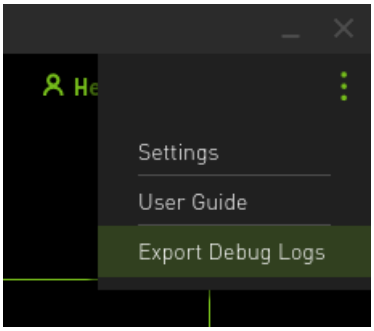
SDK Manager will now complete the installation of the software libraries. Skipping this step will not install any new components on your target hardware and will keep the current operating system on your device.

Please note when installing DRIVE OS with DriveWorks, the HDMI display will not be activated when flashing is completed. (See Enable Display below.) Connect to the DRIVE AGX platform console via a terminal emulator to determine when flashing has been completed. Instructions for connecting a terminal emulator to the platform are in the DRIVE OS 5.1 Linux Developer Guide sections Using tcu_muxer and Terminal Emulation.

5.4 Step 4: Finalize Development Environment Setup

1. From **Step 04 Summary Finalization**, there is a summary of the components that were installed, along with any warnings or errors that were encountered.
2. The **Export Logs** link creates a ZIP file of all log files created during installation. This ZIP file is located in the same folder path where the SDK Manager installer downloaded all components.

   Alternatively, click the menu icon in the top right corner of the window (":"), and choose Export Debug Logs from the drop-down menu in the top-right corner.



3. Consult the **Error Messages** for information about any errors you may encounter.
4. Click **Finish and Exit** to complete the installation.

5.5 Finalize DRIVE AGX System Setup

1. **Update the default Linux username and password.**

   To ensure your data safety and security, connect the DRIVE AGX Developer Kit via a terminal emulator to execute the prompts to update the default username and password. See DRIVE OS Linux User Setup for more information.

2. **Enable SSH.**

   In order to protect the security of your system when placed in a vehicle, the DRIVE AGX SSH server is disabled by default. To re-enable, use the following steps at the DRIVE AGX platform console via a terminal emulator. See Setting Up SSH Server Service for more information.

   a. Remove the stamp file to unblock SSH server:
   ```
   $ sudo rm -f /etc/ssh/sshd_not_to_be_run
   ```
   b. Start SSH server service on the current boot:
   ```
   $ sudo systemctl start ssh
   ```
   c. Start service to add SSH host-keys to the target:
   ```
   $ sudo systemctl start nv_ssh_host_keys
   ```
   d. After completing the above steps, the SSH server service is started, and is run on every boot. SSH clients may now connect to this SSH server.

3. **Enable Display.**

   To maximize the compute capacity of the DRIVE AGX Platform, DRIVE OS release does **not** include a Linux Desktop by default. To enable use of the display or to install a desktop, do one of the following:

   a. To enable display without a desktop, start the X server as below. See Manually Starting X Server for more information.
   ```
   $ sudo -b X -ac -noreset -nolisten tcp
   ```

b. To install the Ubuntu desktlop, use the following instructions. See Installing GUI on the Target for more information.

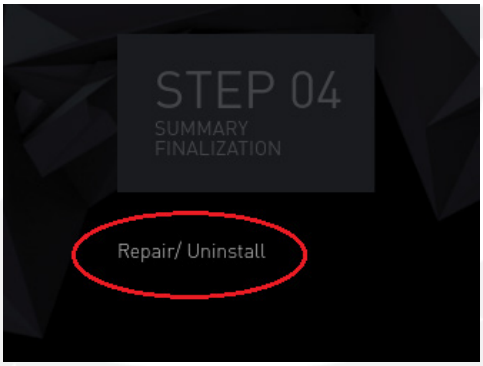
```
$ sudo apt-get update
```

```
$ sudo apt-get install gdm3 ubuntu-unity-desktop
```

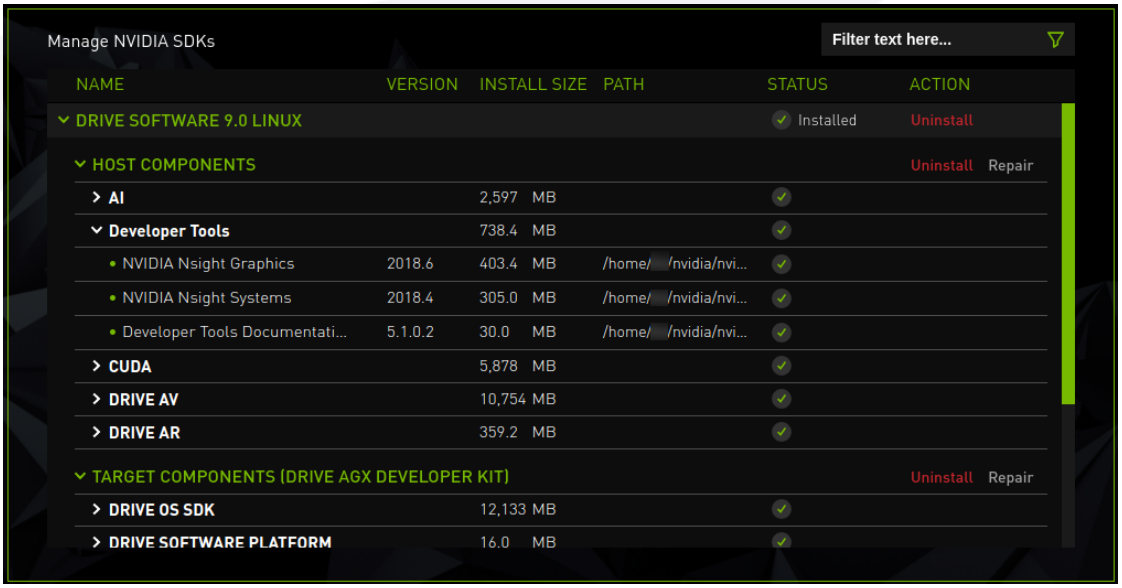
5.6 Repair and Uninstall

To update or uninstall an SDK on your system, launch SDK Manager again.

1. On Step 1, under the installation step numbers, click the **Repair / Uninstall** hyperlink.



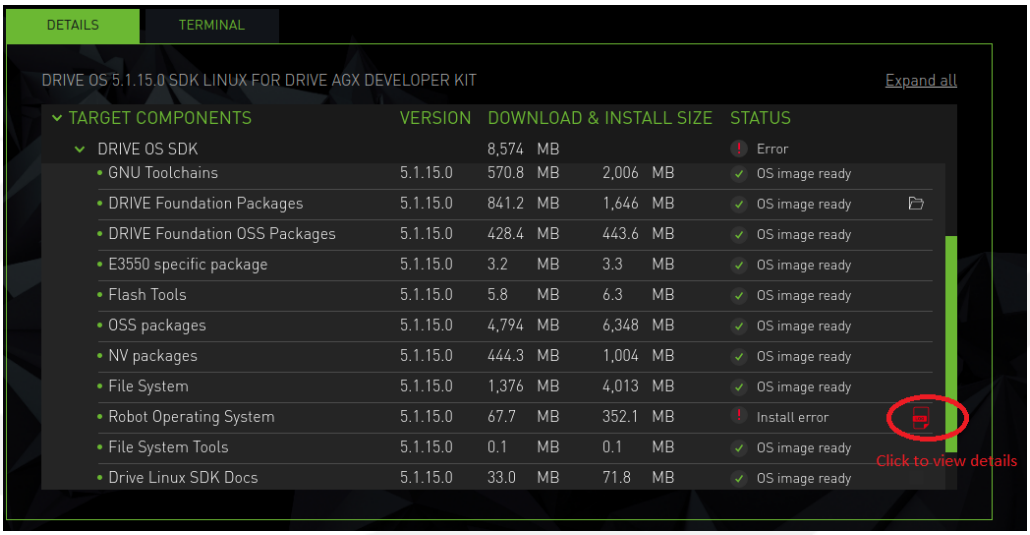
2. The **Manage NVIDIA SDKs** screen shows what has been installed on your system. You can select whether to repair a broken installation, update an existing SDK, or uninstall an SDK.



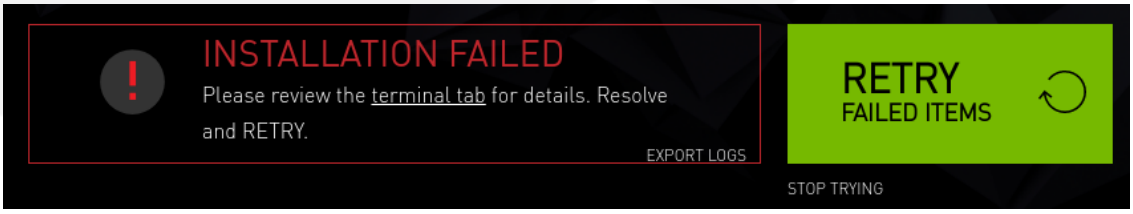
5.6.1 Recommended Recovery Steps

There are many causes of various installation errors. Below is a checklist of common installation issues, which may help you recover from a broken installation.

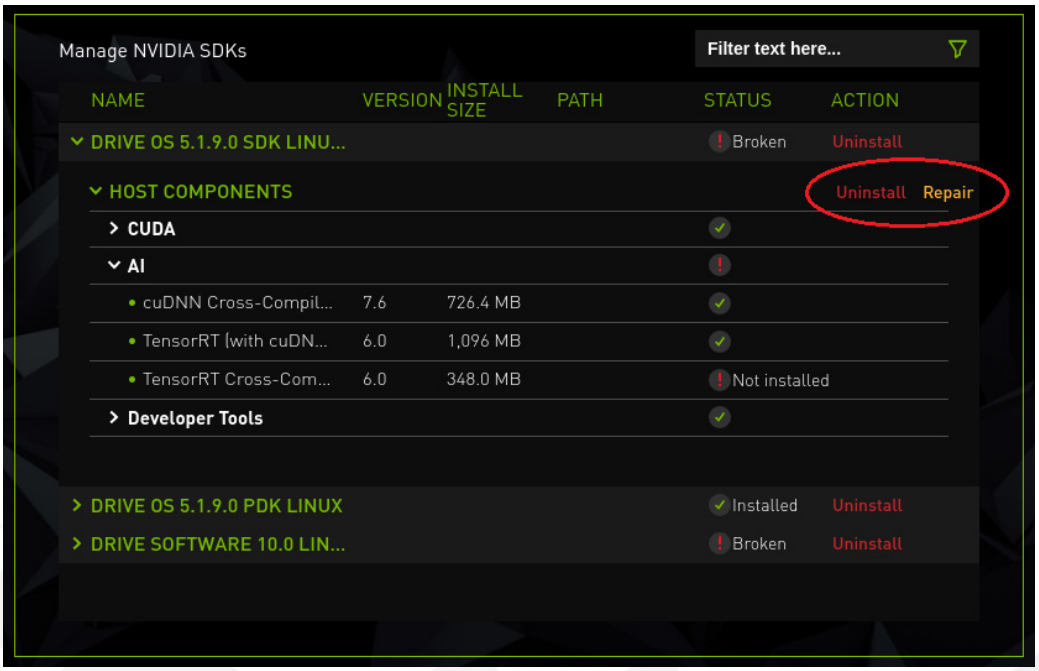
1. Review the summary table to identify which component failed.
    a. Expand the group with the "Error" status.
    b. When you find the failed component, click the details icon to the right of **Install Error** to be redirected to the **Terminal** tab, which will display the exact error.



2. If the error is related to an environment issue, such as a broken apt repository or missing prerequisite, try to fix it manually, then click the **Retry Failed Items** button.



3. Retrying the installation is also available in two other ways:
    a. From STEP 01, use the Repair/Uninstall button to get to the **Manage NVIDIA SDKs** page. If needed, expand the SDK that has the "Broken" status, then click **Repair** for the relevant part (Host or Target).



b. At STEP 01, select the required SDK and run through the installation again.

4. Finally, try to uninstall and reinstall the relevant SDK.

6 CUDA Toolkit Installation Instructions

The Safety toolkit is used primarily by customers going to Production with the Safety Certified DriveOS binaries and libraries and complying with the DriveOS Safety Manual.

The safety build, aarch64-QNX-safety, is only supported on QNX OS. The software stack is designed according to standard safety software development practices. This document assumes you have a safety-hardened stack running on QNX OS and safety-certified hardware.

The CUDA safety toolkit has these safety-certified elements:

- CUDA runtime library
- CUDA math library (no other CUDA libraries)
- NVCC compiler

The safety build does not support CUDA developer tools. The users of the CUDA software stack are expected to debug the GPU software by building the software for aarch64-QNX.

The safety toolkit allows you to do the following:

- Install more than one parallel safety toolkits on the same machine.
- Use the safety certified NVCC compiler to build the application using standard QNX DriveOS build for debugging using CUDA development tools.

For details on CUDA host installation and cross development installation, see https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html and https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#cross-platform.

NOTE: The CUDA safety driver is a subset of the CUDA standard library. If you need to debug using standard build, use the standard CUDA driver and flash the standard build of DriveOS.

Terms and abbreviations:

AV	Autonomous vehicle; refers to autonomous vehicles as a class or an application area, as in "AV pipelines" or "AV use cases".
Safety	Used by developers for production with the Safety Certified and hardened APIs and DRIVE OS libraries that are compliant with the DRIVE OS Safety Manual.
Standard	Used by developers to create algorithms and AV applications for AV use cases on NVIDIA Automotive platforms using NVIDIA DRIVE OS APIs. Variants of Standard build include X86, AArch64 Linux, and AArch64 QNX.

6.1 Pre-installation Actions

Some actions must be taken before the CUDA Toolkit and Driver can be installed on Linux:

- Verify the system has a CUDA-capable GPU.
- Verify the system is running a supported version of Linux.
- Verify the system has the correct kernel headers and development packages installed.
- Verify the system has a host compiler such as qcc for QNX or gcc for Linux.
- Verify the host system has at least 10GB free on the root partition.
- Download the NVIDIA CUDA Toolkit.

You can override the install-time prerequisite checks by running the installer with the --override flag. Remember that the prerequisites will still be required to use the NVIDIA CUDA Toolkit.

6.1.1 Verify You Have a CUDA-Capable GPU

To verify that your GPU is CUDA-capable, go to your distribution's equivalent of System Properties, or, from the command line, enter:

```
$ lspci | grep -i nvidia
```

If you do not see any settings, update the PCI hardware database that Linux maintains by entering `update-pciids` (generally found in `/sbin`) at the command line and rerun the previous `lspci` command.

If your graphics card is from NVIDIA and it is listed in http://developer.nvidia.com/cuda-gpus, your GPU is CUDA-capable.

The Release Notes for the CUDA Toolkit also contain a list of supported products.

6.1.2 Verify You Have a Supported Version of Linux

The CUDA Development Tools are only supported on specific distributions of Linux. The CUDA Safety toolkit is only supported on Ubuntu 18.04.

To determine which distribution and release number you're running, type the following at the command line:

```
$ uname -m && cat /etc/*release
```

6.1.3 Verify the System Has the Correct Kernel Headers and Development Packages Installed

If you perform a system update which changes the version of the linux kernel being used, make sure to rerun the command below to ensure you have the correct kernel headers and kernel development packages installed. Otherwise, the CUDA Driver will fail to work with the new kernel.

Ubuntu

The kernel headers and development packages for the currently running kernel can be installed with:

```
$ sudo apt-get install linux-headers-$(uname -r)
```

6.2 Host Installation

Basic instructions can be found in the Quick Start Guide. Read on for more detailed instructions.

6.2.1 Overview

The Package Manager installation interfaces with your system's package management system. When using Deb, the downloaded package is a repository package. Such a package only informs the package manager where to find the actual installation packages, but will not install them.

If those packages are available in an online repository, they will be automatically downloaded in a later step. Otherwise, the repository package also installs a local repository containing the installation packages on the system. Whether the repository is available online or installed locally, the installation procedure is identical and made of several steps.

See Ubuntu for installation instructions, below.

Finally, some helpful additional package manager capabilities are detailed.

These instructions are for native development only. For cross-platform development, see the cross development installation section.

The package "cuda-core" has been deprecated in CUDA 9.1. Please use "cuda-compiler" instead.

6.2.2 Ubuntu

1. Perform the pre-installation actions.

2. Install repository meta-data

```
$ sudo dpkg -i cuda-repo-<distro>_<version>_<architecture>.deb
```

3. Installing the CUDA public GPG key

   When installing using the local repo:

```
$ sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub
```

   When installing using network repo on Ubuntu 18.04:

```
$ sudo apt-key adv --fetch-
keys https://developer.download.nvidia.com/compute/cuda/repos/<distro>
/<architecture>/7fa2af80.pub
```

4. Update the Apt repository cache

```
$ sudo apt-get update
```

5. Install CUDA

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

6. Perform the post-installation actions.

6.2.3 Additional Package Manager Capabilities

Below are some additional capabilities of the package manager that users can take advantage of.

6.2.3.1 Available Packages

The recommended installation package is the cuda package. This package will install the full set of other CUDA packages required for native development and should cover most scenarios.

The cuda package installs all the available packages for native development. That includes the compiler, the debugger, the profiler, the math libraries, and so on. For x86_64 patforms, this also include the visual profiler. It also includes the NVIDIA driver package.

On supported platforms, the cuda-cross-armhf, cuda-cross-aarch64, and cuda-cross-ppc64el packages install all the packages required for cross-platform development to ARMv7, ARMv8, and POWER8, respectively. The libraries and header files of the target architecture's display driver package are also installed to enable the cross compilation of driver applications. The cuda-cross-<arch> packages do not install the native display driver.

The packages installed by the packages above can also be installed individually by specifying their names explicitly. The list of available packages be can obtained with:

```
$ cat /var/lib/apt/lists/*cuda*Packages | grep "Package:"
```

6.2.3.2 Package Upgrades

The cuda package points to the latest stable release of the CUDA Toolkit. When a new version is available, use the following command to upgrade the toolkit and driver:

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

The cuda-cross-<arch> packages can also be upgraded in the same manner.

The cuda-drivers package points to the latest driver release available in the CUDA repository. When a new version is available, use the following command to upgrade the driver:

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

Some desktop environments, such as GNOME or KDE, will display a notification alert when new packages are available.

To avoid any automatic upgrade and lock down the toolkit installation to the X.Y release, install the cuda-X-Y or cuda-cross-<arch>-X-Y package.

Side-by-side installations are supported. For instance, to install both the X.Y CUDA Toolkit and the X.Y+1 CUDA Toolkit, install the cuda-X.Y and cuda-X.Y+1 packages.

6.2.3.3 Meta Packages

Meta packages are RPM/Deb packages which contain no (or few) files but have multiple dependencies. They are used to install many CUDA packages when you may not know the details of the packages you want. Below is the list of meta packages.

Table 1: Meta Packages Available for CUDA

Meta Package	Purpose
cuda	Installs all CUDA Toolkit and Driver packages. Handles upgrading to the next version of the cuda package when it's released.
cuda-10-2	Installs all CUDA Toolkit and Driver packages. Remains at version until an additional version of CUDA is installed.
cuda-toolkit-10-2	Installs all CUDA Toolkit packages required to develop CUDA applications. Does not include the driver.
cuda-tools-10-2	Installs all CUDA command line and visual tools.
cuda-runtime-10-2	Installs all CUDA Toolkit packages required to run CUDA applications, as well as the Driver packages.
cuda-compiler-10-2	Installs all CUDA compiler packages.
cuda-libraries-10-2	Installs all runtime CUDA Library packages.
cuda-libraries-dev-10-2	Installs all development CUDA Library packages.
cuda-drivers	Installs all Driver packages. Handles upgrading to the next version of the Driver packages when they're released.

6.3 Cross Development Installation

Perform the following steps to install CUDA for Safety.

1. Obtain the three local repos:
    - cross-qnx-standard
    - cross-qnx-safe
    - minimal-safe-toolkit

    > cross-qnx-standard is a standard build and optional for safety-only development.

2. Run the following commands, substituting the build number for *n*:

```
$ sudo dpkg -i cuda-repo-minimal-safe-toolkit-10-2-local-10.2.n_1.0-1_amd64.deb
$ sudo apt install cuda-qnx-safe-toolkit-10-2-n
$ sudo dpkg -i cuda-repo-cross-qnx-standard-10-2-local-10.2.n_all.deb
$ sudo apt install cuda-qnx-standard-cross-qnx-10-2-n
$ sudo dpkg -i cuda-repo-cross-qnx-safe-10-2-local-10.2.n_all.deb
$ sudo apt update
$ sudo apt install cuda-cross-qnx-safe-10-2-n
```

6.4 Target File Installation

Perform the following steps to install the target files:

1. Obtain the target-repo, for example, `cuda-repo-qnx-10-2-local-target-10.2.109_1.0-1_amd64.deb`.
2. Run the following comands:

```
$ sudo dkg -i <package-name>.deb
$ sudo apt-get update
$ sudo apt-get install cuda-qnx-10-2
```

The target files will be installed on the host system at `/usr/local/cuda-targets`. You will need to copy these files onto the target file system.

> The target installation is only needed for standard builds. There are no dynamic libraries for safety build and there are no target files to be copied over.

6.5 Post-installation Actions

The post-installation actions must be manually performed. These actions are split into mandatory, recommended, and optional sections.

6.5.1 Mandatory Actions

Some actions must be taken after the installation before the CUDA Toolkit and Driver can be used.

Verify the installation was performed correctly by running the following command included in the cross safety packages:

```
$ verify_cuda_installation.sh
```

6.5.1.1 Environment Setup

The PATH variable needs to include:

```
/usr/local/cuda-safe-10.2/bin
```

To add this path to the PATH variable:

```
$ export PATH=/usr/local/cuda-safe- /bin:/usr/local/cuda-safe-
10.2/NsightCompute-2019.1${PATH:+:${PATH}}
```

In addition, when using the runfile installation method, the LD_LIBRARY_PATH variable needs to contain `/usr/local/cuda-safe-10.2/lib64` on a 64-bit system, or `/usr/local/cuda-safe-10.2/lib` on a 32-bit system

- To change the environment variables for 64-bit operating systems:

```
$ export LD_LIBRARY_PATH=/usr/local/cuda-safe-10.2/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

- To change the environment variables for 32-bit operating systems:

```
$ export LD_LIBRARY_PATH=/usr/local/cuda-safe-10.2/lib\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

Note that the above paths change when using a custom install path with the runfile installation method.

6.5.2 Recommended Actions

Other actions are recommended to verify the integrity of the installation.

6.5.2.1 Install Writable Samples

In order to modify, compile, and run the samples, the samples must be installed with write permissions. A convenience installation script is provided:

```
$ cuda-install-samples- .sh <dir>
```

This script is installed with the `cuda-samples-10-2` package. The `cuda-samples-10-2` package installs only a read-only copy in `/usr/local/cuda-10.2/samples`.

6.5.2.1.1 Verify the Installation

Before continuing, it is important to verify that the CUDA toolkit can find and communicate correctly with the CUDA-capable hardware. To do this, you need to compile and run some of the included sample programs.

> Ensure the PATH and, if using the runfile installation method, LD_LIBRARY_PATH variables are set correctly.

6.5.2.1.1.1 Compiling the Examples

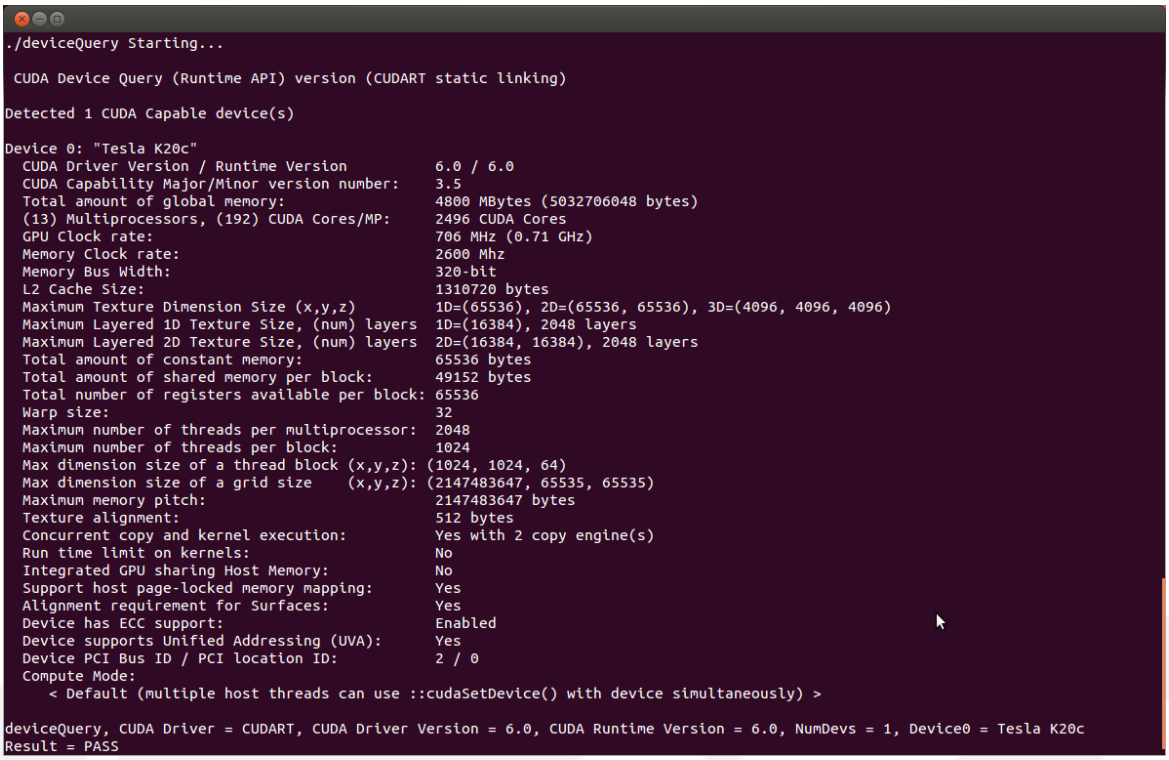
The version of the CUDA Toolkit can be checked by running nvcc -V in a terminal window. The nvcc command runs the compiler driver that compiles CUDA programs. It calls the host compiler such as qcc or gcc for C code and the NVIDIA PTX compiler for the CUDA code.

The NVIDIA CUDA Toolkit includes sample programs in source form. You should compile them by changing to `~/NVIDIA_CUDA-10.2_Samples` and typing `make`. The resulting binaries will be placed under `~/NVIDIA_CUDA-10.2_Samples/bin`.

6.5.2.1.1.2 Appendix A: Running in the Binaries

After compilation, find and run deviceQuery under `~/NVIDIA_CUDA-10.2_Samples`. If the CUDA software is installed and configured correctly, the output for deviceQuery should look similar to that shown in Figure 1.

Figure 1: Valid Results from deviceQuery CUDA Sample



The exact appearance and the output lines might be different on your system. The important outcomes are that a device was found (the first highlighted line), that the device matches the one on your system (the second highlighted line), and that the test passed (the final highlighted line).

If a CUDA-capable device and the CUDA Driver are installed but deviceQuery reports that no CUDA-capable devices are present, this likely means that the `/dev/nvidia*` files are missing or have the wrong permissions.

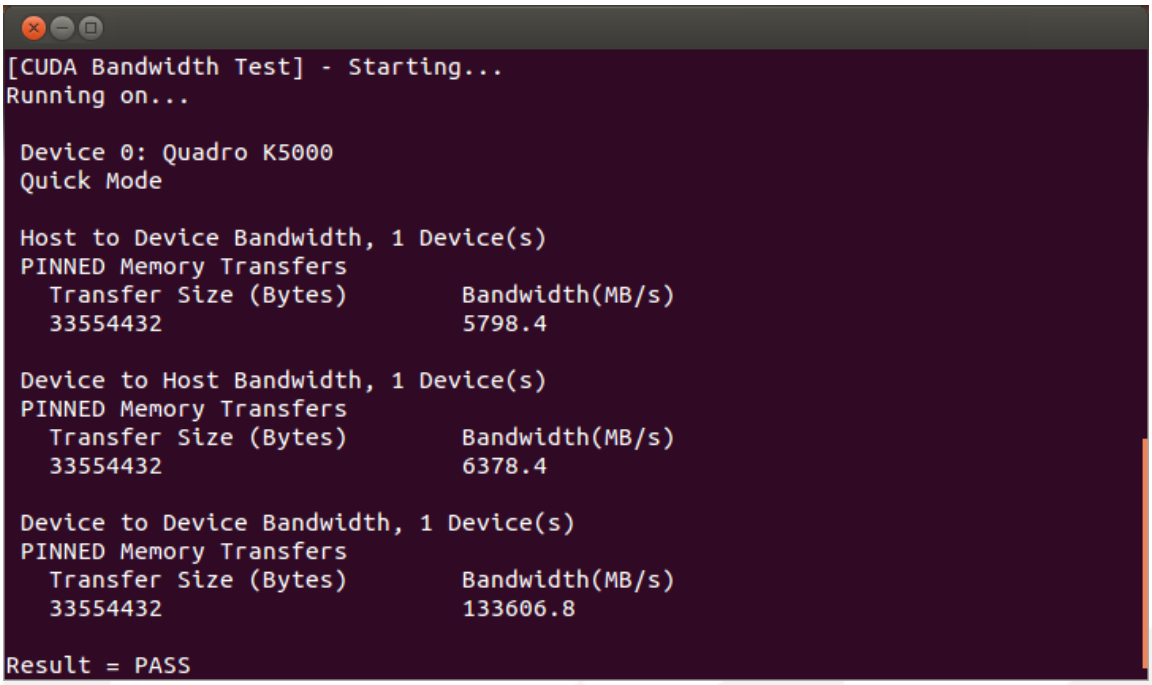
On systems where SELinux is enabled, you might need to temporarily disable this security feature to run deviceQuery. To do this, type:

```
$ setenforce 0
```

from the command line as the superuser.

Running the bandwidthTest program ensures that the system and the CUDA-capable device are able to communicate correctly. Its output is shown in Figure 2.

Figure 2: Valid Results from bandwidthTest CUDA Sample



Note that the measurements for your CUDA-capable device description will vary from system to system. The important point is that you obtain measurements, and that the second-to-last line (in Figure 2) confirms that all necessary tests passed.

If the tests do not pass, make sure you have a CUDA-capable NVIDIA GPU on your system and make sure it is properly installed.

If you run into difficulties with the link step (such as libraries not being found), consult the Linux Release Notes found in the doc folder in the CUDA Samples directory.

6.6 Removing CUDA Toolkit and Driver

Follow the below steps to properly uninstall the CUDA Toolkit and NVIDIA Drivers from your system. These steps will ensure that the uninstallation will be clean.

Ubuntu

To remove CUDA Toolkit:

```
$ sudo apt-get --purge remove "*cublas*" "cuda*"
```

To remove NVIDIA Drivers:

```
$ sudo apt-get --purge remove "*nvidia*"
```

6.7 Troubleshooting

This section lists several scenarios in which anomalous conditions cause unintended results, and possible solutions.

Situation: The user provides the wrong input.

Response: `dpkg/apt` will produce an error indicating the instructions were incorrect and not take any action. Consult the documentation for more information.

Situation: The system reboots in the middle of an install.

Response: The install will fail. Any installed packages should be purged using `sudo dpkg --purge` and the installation should be triggered again.

Situation: The installation/verification fails due to file or package corruption. Package corruption may produce an error such as:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  cuda-nvdisasm-10-2
0 upgraded, 1 newly installed, 0 to remove and 105 not upgraded.
Need to get 0 B/28.2 MB of archives.
After this operation, 29.1 MB of additional disk space will be used.
Get:1 file:/var/cuda-repo-ubuntu1804-10-2-local  cuda-nvdisasm-10-
2 10.2.167-1 [28.2 MB]
Err:1 file:/var/cuda-repo-ubuntu1804-10-2-local  cuda-nvdisasm-10-
2 10.2.167-1
  Hash Sum mismatch
  Hashes of expected file:
   - SHA512:646bebb453fd6161932fb6c7c5a19a209cdde80771808cd80aaad549e03dd2f8
f626ccadd7d9d9d67e4cbf988e8bd7cc59f202f4f5e7a20960f9267b84e2ce3f
   - SHA256:85c2209901934d547c7f9aab0f3026d5e1b70b4c19a666e6595e6803507f5a64
   - SHA1:68a1166e5b5dfc584626e4875ce1e5f551b22506 [weak]
   - MD5Sum:f6a7047a162bf45c9123865ffb4441d0 [weak]
   - Filesize:28164774 [weak]
  Hashes of received file:
   - SHA512:8eafd2a331977ce24fb19a36484b629cbb029cd788834d7a393be020805cbf44
0475f8af221eddcfd67f25496235be13e232387f06f1c7d339287558493e665a
   - SHA256:8a45f9b9d71a9666b7a2d27136e87c5f26c3c9dc70e578ed4d1b126b8e422a59
   - SHA1:be23bbc663ffb056084a19c161eacf22e0d373ac [weak]
   - MD5Sum:4c5e577308dcd696c97a0a29d4501be0 [weak]
   - Filesize:28158580 [weak]
  Last modification reported: Thu, 14 May 2020 18:54:27 +0000
E: Failed to fetch file:/var/cuda-repo-ubuntu1804-10-2-local/./cuda-
nvdisasm-10-2_10.2.167-1_amd64.deb  Hash Sum mismatch
   Hashes of expected file:
    - SHA512:646bebb453fd6161932fb6c7c5a19a209cdde80771808cd80aaad539e03dd2f
8f626ccadd7d9d9d67e4cbf988e8bd7cc59f202f4f5e7a20960f9267b84e2ce3f
    - SHA256:85c2209901934d547c7f9aab0f3026d5e1b70b4c19a666e6595e6803507f5a6
```

```
4
    - SHA1:68a1166e5b5dfc584626e4875ce1e5f551b22506 [weak]
    - MD5Sum:f6a7047a162bf45c9123865ffb4441d0 [weak]
    - Filesize:28164774 [weak]
   Hashes of received file:
    - SHA512:8eafd2a331977ce24fb19a36484b629cbb029cd788834d7a992be020805cbf4
40475f8af221eddcfd67f25496235be13e232387f06f1c7d339287558493e665a
    - SHA256:8a45f9b9d71a9666b7a2d27136e87c5f26c3c9dc70e578ed4d1b126b8e422a5
9
    - SHA1:be23bbc663ffb056084a19c161eacf22e0d373ac [weak]
    - MD5Sum:4c5e577308dcd696c97a0a29d4501be0 [weak]
    - Filesize:28158580 [weak]
   Last modification reported: Thu, 14 May 2020 18:54:27 +0000
E: Unable to fetch some archives, maybe run apt-get update or try with --
fix-missing?
```

Response: Purge (using `sudo dpkg --purge`) any installed packages and attempt a reinstall. If the problem persists, try on a different host system. If it still persists, check for legitimate corruption by comparing the md5sum hash of the downloaded installer to the expected md5sum hash of the download.

Situation: Package dependencies are missing. If dependencies are missing, the error can manifest in two different forms:

```
Err:1 file:/var/cuda-repo-ubuntu1804-10-2-local  cuda-nvdisasm-10-
2 10.2.167-1
  File not found - /var/cuda-repo-ubuntu1804-10-2-local/./cuda-nvdisasm-10-
2_10.2.167-1_amd64.deb (2: No such file or directory)
The following packages have unmet dependencies:
cuda : Depends: cuda-10-2 (>= 10.2.167) but it is not going to be installed
```

Response: Ensure the required files are in the correct directories.

Situation: Package signing failed, or the signing key was not imported. The error will appear during the apt-get update step, and may look like:

```
Reading package lists... Done
W: An error occurred during the signature verification. The repository is no
t updated and the previous index files will be used. GPG error: file:/var/cu
da-repo-ubuntu1804-10-2-
local  Release: The following signatures couldn't be verified because the pu
blic key is not available: NO_PUBKEY F60F4B3D7FA2AF80
W: Failed to fetch file:/var/cuda-repo-ubuntu1804-10-2-
local/Release.gpg  The following signatures couldn't be verified because the
 public key is not available: NO_PUBKEY F60F4B3D7FA2AF80
W: Some index files failed to download. They have been ignored, or old ones
used instead.
```

Response: Import the required key.

Situation: System has run out of disk space. The error may look like:

```
Errors were encountered while processing:
 /tmp/apt-dpkg-install-PySCgh/088-nsight-systems-2020.2.5_2020.2.5.8-
1_amd64.deb
 /tmp/apt-dpkg-install-PySCgh/106-cuda-documentation-10-2_10.2.167-
```

```
1_amd64.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

Response: Add disk space and retry the installation.

7 TensorRT 6.3.1 Installation Instructions

7.1 Getting Started

Ensure you are familiar with the following installation requirements and notes.

- If you are using the TensorRT Python API and PyCUDA isn't already installed on your system, see *Installing PyCUDA*. If you encounter any issues with PyCUDA usage, you may need to recompile it yourself. For more information, see Installing PyCUDA on Linux.
- Ensure you are familiar with the TensorRT 6.3.1 Release Notes in the *DRIVE OS 5.2.0.0 SDK/PDK Release Notes*.
- Verify that you have the CUDA Toolkit installed; version 10.2 is supported.
- The TensorFlow to TensorRT model export requires TensorFlow 1.14.0.
- The PyTorch examples have been tested with PyTorch 1.3.0, but may work with older versions.
- If the target system has both TensorRT and one or more training frameworks installed on it, the simplest strategy is to use the same version of cuDNN for the training frameworks as the one that TensorRT ships with. If this is not possible, or for some reason strongly undesirable, be careful to properly manage the side-by-side installation of cuDNN on the single system. In some cases, depending on the training framework being used, this may not be possible without patching the training framework sources.
- The `libnvcaffe_parser.so` library functionality from previous versions is included in `libnvparsers.so` since TensorRT 5.0. The installed symbolic link for `libnvcaffe_parser.so` is updated to point to the new libnvparsers.so library. The static library `libnvcaffe_parser.a` is also symbolically linked to `libnvparsers_static.a`.

7.2 Installing TensorRT

The Debian installation automatically installs any dependencies; however, it:

- requires sudo or root privileges to install
- provides no flexibility as to which location TensorRT is installed into
- requires that the CUDA Toolkit and cuDNN have also been installed using Debian
- does not allow more than one minor version of TensorRT to be installed at the same time

TensorRT versions: TensorRT is a product made up of separately versioned components. The version on the product conveys important information about the significance of new features while the library version conveys information about the compatibility or incompatibility of the API.

Table 2: Versioning of TensorRT components

Product or Component		Previously Released Version	Current Version	Version Description
TensorRT product		6.2.0	6.3.1	+1.0 when significant new capabilities are added. +0.1 when capabilities have been improved.
nvinfer libraries, headers, samples, and documentation.		6.2.0	6.3.1	+1.0 when the API or ABI changes in a non-compatible way. +0.1 when the API or ABI changes are backward compatible
UFF	uff-converter-tf Debian and RPM packages	6.2.0	6.3.1	+0.1 while we are developing the core functionality. Set to 1.0 when we have all base functionality in place.
	uff-*.whl file	0.6.6	0.6.6	
graphsurgeon	graphsurgeon-tf Debian and RPM packages	6.2.0	6.3.1	+0.1 while we are developing the core functionality.

Product or Component		Previously Released Version	Current Version	Version Description
	graphsurgeon-*.whl file	0.4.1	0.4.1	Set to 1.0 when we have all base functionality in place.
libnvinfer python packages	python-libnvinfer python-libnvinfer-dev python3-libnvinfer python3-libnvinfer-dev Debian and RPM packages	6.2.0	6.3.1	+1.0 when the API or ABI changes in a non-compatible way. +0.1 when the API or ABI changes are backward compatible.
	tensorrt.whl file	6.2.0	6.3.1	

7.2.1 Debian Installation

This section contains instructions for a developer installation and an app server installation.

Developer Installation: The following instructions set up a full TensorRT development environment with samples, documentation and both the C++ and Python API.

Ensure you are a member of the NVIDIA Developer Program. If not, follow the prompts to gain access.

Refer to Download and Run SDK Manager for instructions on setting up your development environment with NVIDIA SDK Manager.

Go to https://developer.nvidia.com/nvidia-sdk-manager for more information about using SDK Manager.

7.2.1.1 Using The NVIDIA Machine Learning Network Repo For RPM Installation

It's suggested that you set up the NVIDIA CUDA network repository first before setting up the NVIDIA Machine Learning network repository to satisfy package dependencies. We provide some example commands below to accomplish this task. For more information, see the CUDA installation chapter in the *DRIVE OS 5.2.0.0 SDK Development Guide*.

1. Install the NVIDIA CUDA network repository installation package.

```
os="ubuntu1x04"
cuda="x.y.z"
wget https://developer.download.nvidia.com/compute/cuda/repos/${os}/x8
6_64/cuda-repo-${os}_${cuda}-1_amd64.deb
sudo dpkg -i cuda-repo-*.deb
```

Where:

- OS version: ubuntu1x04 is 1804
- CUDA version: x.y.z is 10.2.89

2. Install the NVIDIA Machine Learning network repository installation package.

```
os="ubuntu1x04"
wget https://developer.download.nvidia.com/compute/machine-
learning/repos/${os}/x86_64/nvidia-machine-learning-repo-${os}_1.0.0-
1_amd64.deb

sudo dpkg -i nvidia-machine-learning-repo-*.deb
sudo apt-get update
```

3. Install the TensorRT package that fits your particular needs.

   a. For only running TensorRT C++ applications:

   ```
   sudo apt-
   get install libnvinfer6 libnvonnxparsers6 libnvparsers6 libnvinf
   er-plugin6
   ```

   b. For also building TensorRT C++ applications:

   ```
   sudo apt-get install libnvinfer-dev libnvonnxparsers-dev
   libnvparsers-dev libnvinfer-plugin-dev
   ```

   c. For running TensorRT Python applications:

   ```
   sudo apt-get install python-libnvinfer python3-libnvinfer
   ```

4. When using the NVIDIA Machine Learning network repository, Ubuntu will by default install TensorRT for the latest CUDA version. The following commands will install libnvinfer6 for an older CUDA version and hold the libnvinfer6 package at this version. Replace 6.x.x with your version of TensorRT and cudax.x with your CUDA version for your install.

```
version="6.x.x-1+cudax.x"
sudo apt-
get install libnvinfer6=${version} libnvonnxparsers6=${version} libnvp
arsers6=${version} libnvinfer-plugin6=${version} libnvinfer-
dev=${version} libnvonnxparsers-dev=${version} libnvparsers-
dev=${version} libnvinfer-plugin-dev=${version} python-
libnvinfer=${version} python3-libnvinfer=${version}

sudo apt-
mark hold libnvinfer6 libnvonnxparsers6 libnvparsers6 libnvinfer-
plugin6 libnvinfer-dev libnvonnxparsers-dev libnvparsers-
dev libnvinfer-plugin-dev python-libnvinfer python3-libnvinfer
```

If you want to upgrade to the latest version of TensorRT or the latest version of CUDA, then you can unhold the libnvinfer6 package using the following command.

```
sudo apt-
mark unhold libnvinfer6 libnvonnxparsers6 libnvparsers6 libnvinfer-
plugin6 libnvinfer-dev libnvonnxparsers-dev libnvparsers-
dev libnvinfer-plugin-dev python-libnvinfer python3-libnvinfer
```

You may need to repeat these steps for libcudnn7 to prevent cuDNN from being updated to the latest CUDA version. Refer to the TensorRT 6.3.1 Release Notes in the *DRIVE OS 5.2.0.0 SDK/PDK Release Notes* for the specific version of cuDNN that was tested with your version of TensorRT. Example commands for downgrading and holding the cuDNN version can be found in the *Safety Supported Samples And Tools* section. Refer to the CUDA installation section of the *DRIVE OS 5.2.0.0 SDK Development Guide*.

If both the NVIDIA Machine Learning network repository and a TensorRT local repository are enabled at the same time you may observe package conflicts with either TensorRT or cuDNN. You will need to configure APT so that it prefers local packages over network packages. You can do this by creating a new file at `/etc/apt/preferences.d/local-repo` with the following lines:

```
Package: *
Pin: origin ""
Pin-Priority: 1001
```

This preference change will affect more than just TensorRT in the unlikely event that you have other repositories which are also not downloaded over HTTP(S). To revert APT to its original behavior simply remove the newly created file.

7.2.2 Additional Installation Methods

Aside from installing TensorRT from the product package, you can also install TensorRT from the following locations.

TensorRT container	The TensorRT container provides an easy method for deploying TensorRT with all necessary dependencies already packaged in the container. For information about installing TensorRT via a container, see the TensorRT Container Release Notes.
JetPack	JetPack bundles all Jetson platform software, including TensorRT. Use it to flash your Jetson Developer Kit with the latest OS image, install NVIDIA SDKs, and jump-start your development environment. For information about installing TensorRT through JetPack, see the JetPack documentation. For JetPack downloads, see Develop: JetPack.
NVIDIA DriveWorks	With every release, TensorRT delivers features to make the DRIVE Development Platform an excellent computing platform for Autonomous Driving. For more information about installing TensorRTs through DriveWorks, see the DriveWorks documentation. For DriveWorks downloads, see NVIDIA Developer: DRIVE Downloads.

7.3 Safety Supported Samples And Tools

Title	TensorRT Sample Name	Description
dlaSafetyBuilder	dlaSafetyBuilder	A tool to generate the NvMedia DLA loadable from models without having to develop your own application.
dlaSafetyRuntime	dlaSafetyRuntime	A tool to load a DLA loadable and run inference using safety certified NvMedia DLA APIs.
"Hello World" For TensorRT Safety	sampleSafeMNIST	Consists of two parts; build and infer. The build part of this sample demonstrates how to use the builder flag for safety. The inference part of this sample demonstrates how to use the safe runtime, engine and execution context.

7.3.1 Safety C++ Samples

You can find the Safety samples in the `/usr/src/tensorrt/samples` package directory. The following Safety samples are shipped with TensorRT:

Running Safety Samples

To run one of the Safety samples, the process typically involves the following steps:

1. Download the dataset.
2. Download the prototxt file.
3. Put all the images and files into the data directory
4. Compile the sample.

For more information on running samples, see the README.md file included with the sample.

7.3.2 "Hello World" For TensorRT Safety

What does this sample do?

This sample, `sampleSafeMNIST`, consists of two parts; build and infer. The build part of this sample demonstrates how to use the builder `IBuilderConfig::setEngineCapability()` flag for safety. The inference part of this sample demonstrates how to use the safe runtime, engine and execution context.

The build part builds a safe version of a TensorRT engine and saves it into a binary file, then the infer part loads the prebuilt safe engine and performs inference on an input image. The infer part uses the safety header proxy, with the CMakeLists.txt file demonstrating how to build it against the safety subset. This sample can be run in FP16 and INT8 modes.

Specifically, this sample demonstrates how to:

- Perform the basic setup and initialization of TensorRT using the Caffe parser

- Import A Caffe Model Using The C++ Parser API
- Preprocess the input and store the result in a managed buffer
- Build An Engine In C++
- Serialize A Model In C++
- Perform Inference In C++

For step-by-step instructions, refer to the *DRIVE OS 5.2.0.0 TensorRT 6.3.1 API Reference* PDF in the DRIVE OS 5.2.0.0 SDK product package.

Where is this sample located?

This sample is maintained under the `/usr/src/tensorrt/samples/sampleSafeMNIST` directory. If using the Debian or RPM package, the sample is located at `/usr/src/tensorrt/samples/sampleSafeMNIST`. If using the tar or zip package, the sample is at `<extracted_path>/samples/sampleSafeMNIST`.

How do I get started?

Refer to the `/usr/src/tensorrt/samples/sampleSafeMNIST/README.md` file for detailed information about how this sample works, sample code, and step-by-step instructions on how to run and verify its output.

7.4 Cross Compiling Samples for AArch64 Users

The following sections show how to cross compile TensorRT samples for AArch64 users.

7.4.1 Prerequisites

1. Install the CUDA cross-platform toolkit for the corresponding target and set the environment variable `CUDA_INSTALL_DIR`.

   ```
   $ export CUDA_INSTALL_DIR="your cuda install dir"
   ```

   Where `CUDA_INSTALL_DIR` is set to `/usr/local/cuda` by default.

2. Install the cuDNN cross-platform libraries for the corresponding target and set the environment variable `CUDNN_INSTALL_DIR`.

   ```
   $ export CUDNN_INSTALL_DIR="your cudnn install dir"
   ```

   Where `CUDNN_INSTALL_DIR` is set to `CUDA_INSTALL_DIR` by default.

3. Install the TensorRT cross compilation Debian packages for the corresponding target.

NOTE: If you are using the tar file release for the target platform, then you can safely skip this step. The tar file release already includes the cross compile libraries so no additional packages are required.

AArch64 QNX	libnvinfer-dev-cross-qnx
AArch64 Linux	libnvinfer-dev-cross-aarch64

7.4.2 Building Samples for AArch64 QNX

Download the QNX tool-chain and export the following environment variables.

```
$ export QNX_HOST=/path/to/your/qnx/toolchains/host/linux/x86_64
$ export QNX_TARGET=/path/to/your/qnx/toolchain/target/qnx7
```

Build the samples by issuing:

```
$ cd /path/to/TensorRT/samples
$ make TARGET=qnx
```

7.4.3 Building Samples for AArch64 Linux

For Linux AArch64 you need to first install the corresponding GCC compiler, aarch64-linux-gnu-g++. In Ubuntu, this can be installed via:

```
$ sudo apt-get install g++-aarch64-linux-gnu
```

Build the samples by issuing:

```
$ cd /path/to/TensorRT/samples
$ make TARGET=aarch64
```

7.5 Upgrading TensorRT

Upgrading TensorRT to the latest version is only supported when the currently installed TensorRT version is equal to or newer than the last two public releases. For example, TensorRT 6.x.x supports upgrading from TensorRT 5.1.x and TensorRT 6.0.x. If you want to upgrade from an unsupported version, then you should upgrade incrementally until you reach the latest version of TensorRT.

7.5.1 Ubuntu Users

The following section provides step-by-step instructions for upgrading TensorRT for Ubuntu users.

7.5.1.1 Upgrading from TensorRT 5.x.x to TensorRT 6.x.x

These upgrade instructions are for Red Hat Enterprise Linux (RHEL) and CentOS users only. When upgrading from TensorRT 5.x.x to TensorRT 6.x.x, ensure you are familiar with the following notes:

Using an RPM file	• The RPM packages are designed to upgrade your development environment without removing any runtime components that other packages and programs might rely on. If you installed TensorRT 5.x.x via an RPM package and you want to upgrade to TensorRT 6.x.x, your documentation, samples, and headers will all be updated to the TensorRT 6.x.x content. After you have downloaded the new local repo, issue:
	`sudo rpm -Uvh nv-tensorrt-repo-rhel7-cudax.x-trt6.x.x.x-ga-yyyymmdd-1-1.x86_64.rpm`
	`sudo yum clean expire-cache`
	`sudo yum install tensorrt libcudnn7`
	• If using Python 2.7:
	`sudo yum install python-libnvinfer-devel`
	• If using Python 3:
	`sudo yum install python3-libnvinfer-devel`
	• If using uff-converter and/or graphsurgeon:
	`sudo yum install uff-converter-tf graphsurgeon-tf`
	• After you upgrade, ensure you see the `/usr/src/tensorrt` directory and the corresponding version shown by the `rpm -qa tensorrt` command is 6.x.x.x.
	• If you are currently or were previously using the NVIDIA Machine Learning network repository, then it may conflict with the version of libcudnn7 that is expected to be installed from the local repository for TensorRT. The following commands will change libcudnn7 to version 7.6.x.x, which is supported and tested with TensorRT 6.x.x, and hold the libcudnn7 package at this version. Replace cuda10.x with the appropriate CUDA version for your install.
	`sudo yum downgrade libcudnn7-7.6.x.x-1.cuda10.x \`
	`  libcudnn7-devel-7.6.x.x-1.cuda10.x`
	`sudo yum install yum-plugin-versionlock`
	`sudo yum versionlock libcudnn7 libcudnn7-devel`

7.6 Uninstalling TensorRT

To uninstall TensorRT using the Debian package, follow these steps:

1. Uninstall libnvinfer6 which was installed using the Debian package.
   `sudo apt-get purge "libnvinfer*"`
2. Uninstall `uff-converter-tf` and `graphsurgeon-tf`, which were also installed using the Debian package.
   `sudo apt-get purge graphsurgeon-tf`

The `uff-converter-tf` will also be removed with the above command.

You can use the following command to uninstall `uff-converter-tf` and not remove `graphsurgeon-tf`, however, it is no longer required.

```
sudo apt-get purge uff-converter-tf
```

You can later use `autoremove` to uninstall `graphsurgeon-tf` as well.

```
sudo apt-get autoremove
```

3. Uninstall the Python TensorRT wheel file.

   If using Python 2.7:

   ```
   sudo pip2 uninstall tensorrt
   ```

   If using Python 3.x:

   ```
   sudo pip3 uninstall tensorrt
   ```

4. Uninstall the Python UFF wheel file.

   If using Python 2.7:

   ```
   sudo pip2 uninstall uff
   ```

   If using Python 3.x:

   ```
   sudo pip3 uninstall uff
   ```

5. Uninstall the Python GraphSurgeon wheel file.

   If using Python 2.7:

   ```
   sudo pip2 uninstall graphsurgeon
   ```

   If using Python 3.x:

   ```
   sudo pip3 uninstall graphsurgeon
   ```

7.7 Installing PyCUDA

This section provides useful information regarding PyCUDA including how to install.

ATTENTION: If you have to update your CUDA version on your system, do not install PyCUDA at this time. Perform the steps in *Updating CUDA* first, then install PyCUDA.

PyCUDA is used within Python wrappers to access NVIDIA's CUDA APIs. Some of the key features of PyCUDA include:

- Maps all of CUDA into Python.
- Enables run-time code generation (RTCG) for flexible, fast, automatically tuned codes.
- Added robustness: automatic management of object lifetimes, automatic error checking
- Added convenience: comes with ready-made on-GPU linear algebra, reduction, scan.
- Add-on packages for FFT and LAPACK available.

- Fast. Near-zero wrapping overhead.

To install PyCUDA first make sure nvcc is in your PATH, then issue the following command:

```
pip install 'pycuda>=2019.1.1'
```

If you encounter any issues with PyCUDA usage after installing PyCUDA with the above command, you may need to recompile it yourself. For more information, see Installing PyCUDA on Linux.

7.7.1 Updating CUDA

Existing installations of PyCUDA will not automatically work with a newly installed CUDA Toolkit. That is because PyCUDA will only work with a CUDA Toolkit that is already on the target system when PyCUDA was installed. This requires that PyCUDA be updated after the newer version of the CUDA Toolkit is installed. The steps below are the most reliable method to ensure that everything works in a compatible fashion after the CUDA Toolkit on your system has been upgraded.

1. Uninstall the existing PyCUDA installation.
2. Update CUDA. For more information, see the CUDA installation chapter in the *DRIVE OS SDK Development Guide*.
3. Install PyCUDA. To install PyCUDA, issue the following command:
   ```
   pip install 'pycuda>=2019.1.1'
   ```

7.8 Troubleshooting

For troubleshooting support refer to your support engineer or post your questions onto the NVIDIA Developer Forum.

NVIDIA Developer Forum

8 Additional Resources

For additional information, common problems, and error messages, please refer to the following:

8.1 NVONLINE Users

For users that access DRIVE products through NVIDIA NVONLINE (partners.nvidia.com), see the following:

- DRIVE AGX Developer Kit Hardware Errata [login required] [DocID 1029836]
- NVIDIA DevTalk Forums: DRIVE AGX Developer Kit FAQ