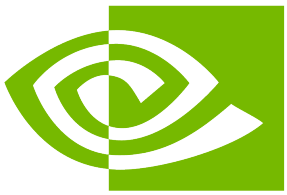

CUDA Compatibility

Release r580

NVIDIA Corporation

Aug 11, 2025

InnerLinkColorHTML000000

Contents

CUDA Compatibility describes the use of new CUDA toolkit components on systems with older base installations.

Chapter 1. Why CUDA Compatibility

The NVIDIA® CUDA® Toolkit enables developers to build NVIDIA GPU accelerated compute applications for desktop computers, enterprise, and data centers to hyperscalers. It consists of the CUDA compiler toolchain including the CUDA runtime (cudart) and various CUDA libraries and tools. To build an application, a developer has to install only the CUDA Toolkit and necessary libraries required for linking.

In order to run a CUDA application, the system should have a CUDA enabled GPU and an NVIDIA driver that is compatible with the CUDA Toolkit that was used to build the application itself. If the application relies on dynamic linking for libraries, then the system should have the right version of such libraries as well.

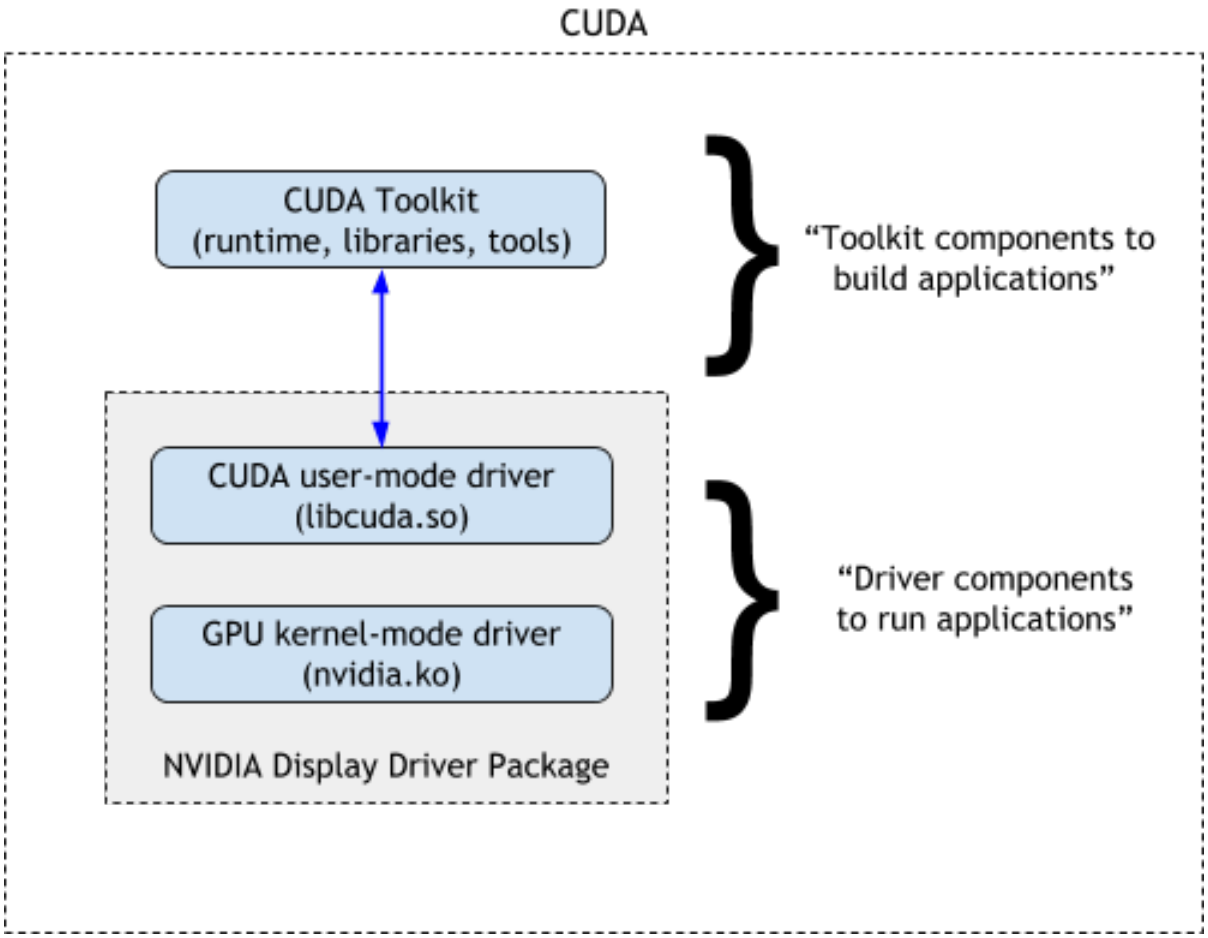

Figure 1: Components of CUDA

Every CUDA toolkit also ships with an NVIDIA driver package for convenience. This driver supports all the features introduced in that version of the CUDA Toolkit. Please check the toolkit and driver version mapping in the release notes. The driver package includes both the user mode CUDA driver (`libcuda.so`) and kernel mode components necessary to run the application.

Typically, upgrading a CUDA Toolkit involves upgrading both the toolkit and the driver to get up to date toolkit and driver capabilities.

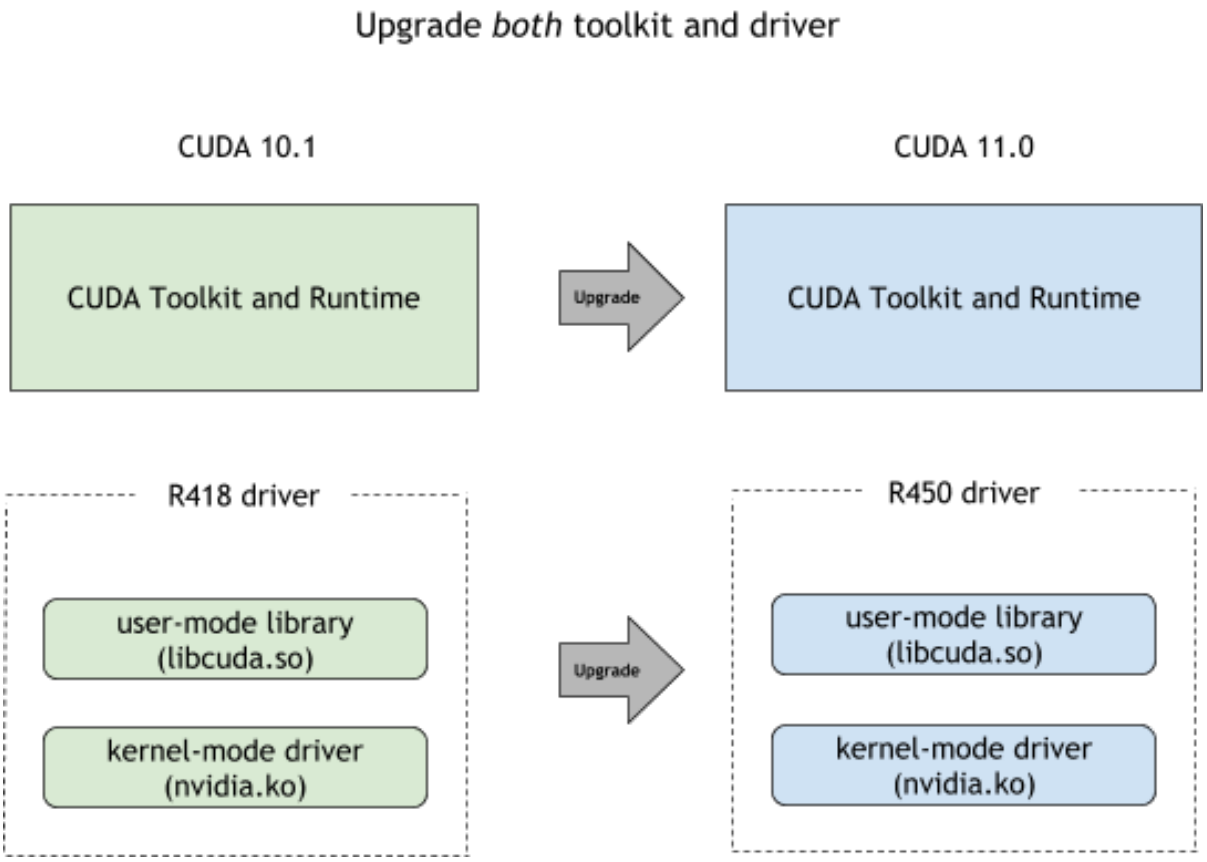

Figure 2: CUDA Upgrade Path

As you can see, when the NVIDIA driver is upgraded along with the CUDA Toolkit, the `CUDA Driver` and `CUDA Runtime` versions in the sample output of `deviceQuery` match perfectly:

```
$ ./deviceQuery
./deviceQuery Starting...

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

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce RTX 4070 SUPER"
  CUDA Driver Version / Runtime Version          12.8 / 12.8
  CUDA Capability Major/Minor version number:    8.9
[...]
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.8, CUDA Runtime Version =
↪12.8, NumDevs = 1
Result = PASS
```

This is not always required. CUDA Compatibility guarantees allow for upgrading only certain components:

- Backwards compatibility ensures that a newer NVIDIA driver can be used with an older CUDA Toolkit. This is implicit and most simple way of doing upgrades.
- Minor version and forward compatibility ensure that an older NVIDIA driver can be used with a newer CUDA Toolkit.

Applications that directly rely only on the CUDA runtime can be deployed in the following scenarios:

CUDA driver (libcuda.so)	**Compatibility type**	**Requirements**
Newer than the CUDA runtime	Backwards compatibility	None
Older than the CUDA runtime, but **same major version** of the CUDA runtime	*Minor Version Compatibility*	No PTX (requires SASS), NVCC target architecture required
Older than the major version of the CUDA runtime	*Forward Compatibility*	Extra CUDA compatibility package

Chapter 2. Minor Version Compatibility

2.1. CUDA 11 and Later Defaults to Minor Version Compatibility

From CUDA 11 onwards, applications compiled with a CUDA Toolkit release from within a CUDA major release family can run, with limited feature-set, on systems having at least the minimum required driver version as indicated below. This minimum required driver can be different from the driver packaged with the CUDA Toolkit but should belong to the same major release.

Refer to the CUDA Toolkit Release Notes for the complete table.

Table 1: CUDA Toolkit 11.x, 12.x, and 13.x Minimum and Maximum Required Driver Versions (Refer to CUDA Release Notes)

CUDA Toolkit	Minimum Required Driver Version	Maximum Required Driver Version
CUDA 13.x	>= 580	N/A
CUDA 12.x	>= 525	< 580
CUDA 11.x	>= 450[1]	< 525

While applications built against any of the older CUDA Toolkits always continued to function on newer drivers due to binary backward compatibility, before CUDA 11, applications built against newer CUDA Toolkit releases were not supported on older drivers without forward compatibility package.

If you are using a new CUDA 10.x minor release, then the minimum required driver version is the same as the driver that's packaged as part of that toolkit release. Consequently, the minimum required driver version changed for every new CUDA Toolkit minor release until CUDA 11.1. Therefore, system administrators always have to upgrade drivers in order to support applications built against CUDA Toolkits from 10.x releases.

[1] CUDA 11.0 was released with an earlier driver version, but by upgrading to Tesla Recommended Drivers 450.80.02 (Linux) / 452.39 (Windows) as indicated, minor version compatibility is possible across the CUDA 11.x family of toolkits.

Table 2: CUDA Toolkit 10.x Minimum Required Driver Versions

CUDA Toolkit	Linux x86_64 Minimum Required Driver Version	Windows Minimum Required Driver Version
CUDA 10.2	>= 440.33	>=441.22
CUDA 10.1	>= 418.39	>=418.96
CUDA 10.0	>= 410.48	>=411.31

With minor version compatibility, upgrading to CUDA 11.1 is now possible on older drivers from within the same major release family such as 450.80.02 that was shipped with CUDA 11.0, as shown below:

```
$ nvidia-smi
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.80.02    Driver Version: 450.80.02    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
[...]
```

```
$ ./deviceQuery
./deviceQuery Starting...

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

Detected 1 CUDA Capable device(s)

Device 0: "Tesla T4"
  CUDA Driver Version / Runtime Version          11.0 / 11.1
  CUDA Capability Major/Minor version number:    7.5
[...]

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.0, CUDA Runtime Version =
↪11.1, NumDevs = 1
Result = PASS
```

Minimum required driver version guidance can be found in the CUDA Toolkit Release Notes. Note that if the minimum required driver version is not installed in the system, applications will return an error as shown below.

```
$ ./deviceQuery
./deviceQuery Starting...

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

cudaGetDeviceCount returned 3
-> initialization error
Result = FAIL
```

2.2. Application Considerations for Minor Version Compatibility

Developers and system administrators should note three important caveats when relying on minor version compatibility. If either of these caveats are limiting, then a new CUDA driver from the same minor version of the toolkit that the application was built with or later is required.

- **Limited feature set**

  Sometimes features introduced in a CUDA Toolkit version may actually span both the toolkit and the driver. In such cases an application that relies on features introduced in a newer version of the toolkit and driver may return the following error on older drivers: `cudaErrorCallRequiresNewerDriver`.

  As mentioned earlier, administrators should then also upgrade the installed driver. Alternatively, application developers can avoid running into this problem by having the application explicitly check for the availability of features.

- **Applications using PTX will see runtime issues**

  Applications that compile device code to PTX will not work on older drivers.

  If the application requires PTX then administrators have to upgrade the installed driver. PTX Developers should refer to the PTX programming guide in the CUDA C++ Programming Guide for details on this limitation.

- **Requires target architecture argument to NVCC**

  Minor Compatibility requires passing the target architecture argument to NVCC, for example `nvcc -arch=sm_xx`.

2.3. Deployment Considerations for Minor Version Compatibility

Minor version compatibility has another benefit that offers flexibility in the use and deployment of libraries. Applications that use libraries that support minor version compatibility can be deployed on systems with a different version of the toolkit and libraries without recompiling the application for the difference in the library version. This holds true for both older and newer versions of the libraries provided they are all from the same major release family. Note that libraries themselves have interdependencies that should be considered. For example, each cuDNN version requires a certain version of cuBLAS.

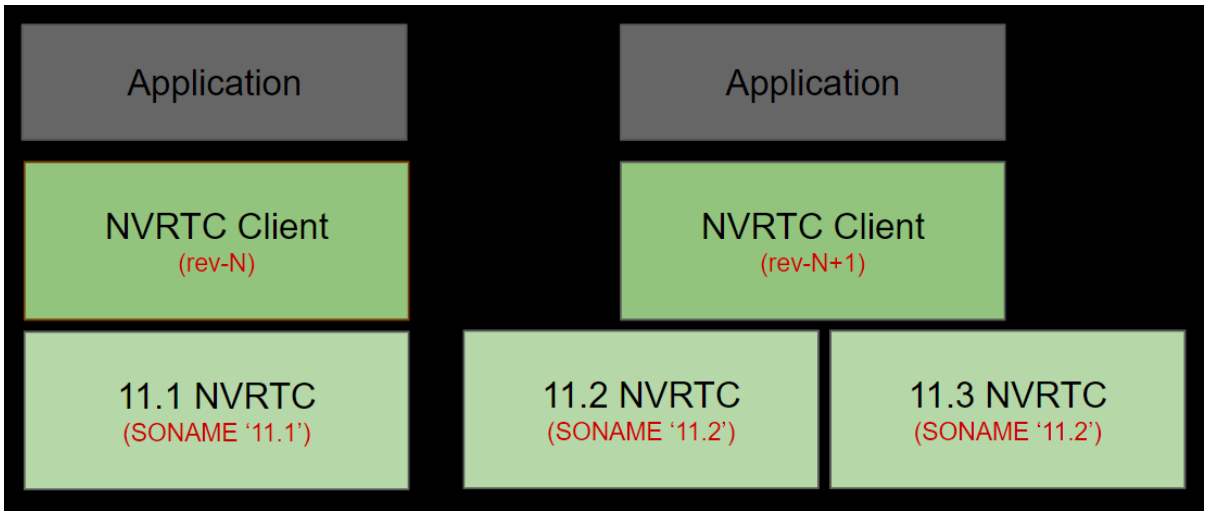

Figure 3: NVRTC supports minor version compatibility from CUDA 11.3 onwards

Chapter 3. Forward Compatibility

3.1. Forward Compatibility Support Across Major Toolkit Versions

Increasingly, data centers and enterprises may not want to update the NVIDIA GPU Driver across major release versions due to the rigorous testing and validation that happens before any system level driver installations are done.

To support such scenarios, CUDA introduced a Forward Compatibility Upgrade path in CUDA 10.0.

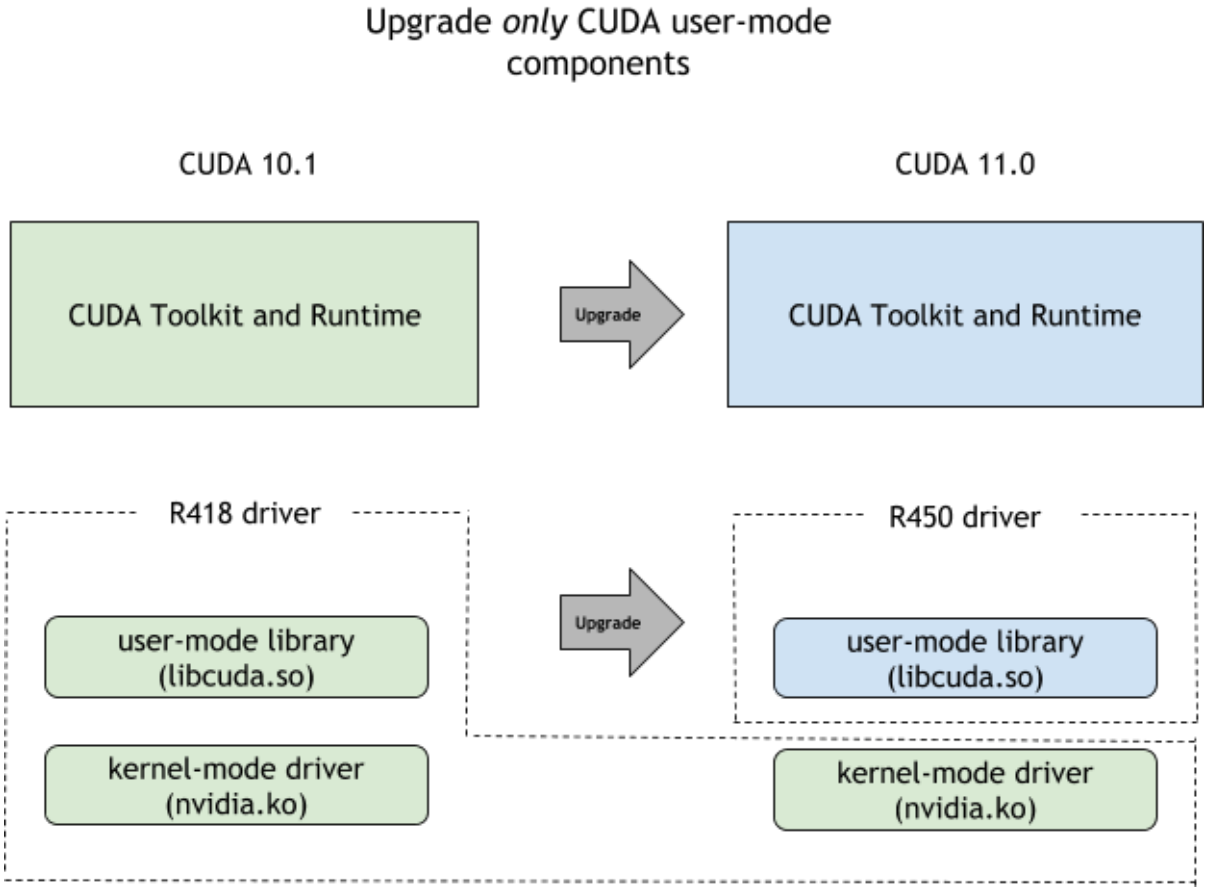

Figure 4: Forward Compatibility Upgrade Path

Warning

Forward Compatibility is applicable only for systems with:

- NVIDIA Data Center GPUs.
- Select NGC Server Ready SKUs of RTX cards.
- Jetson boards.

It's mainly intended to support applications built on newer CUDA Toolkits to run on systems installed with an older NVIDIA Linux GPU driver from different major release families. This new forward-compatible upgrade path requires the use of a special package called "CUDA Comaptibility Package".

3.2. Installing the CUDA Forward Compatibility Package

3.2.1. From Network Repositories or Local Installers

The CUDA forward compatibility package is available in the local installers or the CUDA network repositories provided by NVIDIA as `cuda-compat-<cuda_major_version>-<cuda_minor_version>`.

Each NVIDIA driver is built with an accompanying CUDA supported release; and this package is generated from the NVIDIA driver that matches with the CUDA release for which the compatibility is requested. For example, the `cuda-compat-12-8` package is built using the NVIDIA driver libraries version 570.

Install the package on the system using the package installer.

Ubuntu / Debian:

```
# apt install cuda-compat-12-8
```

Red Hat Enterprise Linux / Rocky Linux / Oracle Linux / Amazon Linux / Fedora / KylinOS:

```
# dnf install cuda-compat-12-8
```

SLES / OpenSUSE:

```
# zypper install cuda-compat-12-8
```

Azure Linux:

```
# tdnf install cuda-compat-12-8
```

The CUDA forward compatibility package will then be installed to the versioned toolkit directory. For example, for the CUDA forward compatibility package of 12.8, the GPU driver libraries of 570 will be installed in `/usr/local/cuda-12.8/compat/`.

The `cuda-compat-<cuda_major_version>-<cuda_minor_version>` package consists of the following files:

- `libcuda.so.*` - CUDA driver

- `libcudadebugger.so.*` - GPU debugging support for CUDA Driver (CUDA 11.8 and later only)
- `libnvidia-nvvm.so.*` - JIT LTO (CUDA 11.5 and later only)
- `libnvidia-ptxjitcompiler.so.*` - JIT compiler for PTX files
- `libnvidia-pkcs11*.so.*` - OpenSSL support for CUDA driver

Note

This package only provides the libraries, and does not configure the system to find such libraries.

3.2.1.1 Example installation and configuration

In this example, the user sets `LD_LIBRARY_PATH` to include the files installed by the `cuda-compat-12-1` package on an Ubuntu system.

Install the package:

```
# apt install -y cuda-compat-12-1
Selecting previously unselected package cuda-compat-12-1.
(Reading database ... 339974 files and directories currently installed.)
Preparing to unpack .../cuda-compat-12-0_530.30-1_amd64.deb ...
Unpacking cuda-compat-12-1 (530.30-1) ...
Setting up cuda-compat-12-1 (530.30-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
```

Check the files installed under `/usr/local/cuda-12.1/compat`:

```
$ ls -l /usr/local/cuda-12.1/compat
total 145676
lrwxrwxrwx 1 root root       12 Jun  3 00:45 libcuda.so -> libcuda.so.1
lrwxrwxrwx 1 root root       17 Jun  3 00:45 libcuda.so.1 -> libcuda.so.530.30
-rw-r--r-- 1 root root 26255520 Jun  3 00:45 libcuda.so.530.30
lrwxrwxrwx 1 root root       25 Jun  3 00:45 libcudadebugger.so.1 -> libcudadebugger.
↪so.530.30
-rw-r--r-- 1 root root 10938424 Jun  3 00:45 libcudadebugger.so.530.30
lrwxrwxrwx 1 root root       19 Jun  3 00:45 libnvidia-nvvm.so -> libnvidia-nvvm.so.4
lrwxrwxrwx 1 root root       24 Jun  3 00:45 libnvidia-nvvm.so.4 -> libnvidia-nvvm.so.
↪530.30
-rw-r--r-- 1 root root 92017376 Jun  3 00:45 libnvidia-nvvm.so.530.30
lrwxrwxrwx 1 root root       34 Jun  3 00:45 libnvidia-ptxjitcompiler.so.1 ->
↪libnvidia-ptxjitcompiler.so.530.30
-rw-r--r-- 1 root root 19951576 Jun  3 00:45 libnvidia-ptxjitcompiler.so.530.30
```

The user can set `LD_LIBRARY_PATH` to include the files installed before running the CUDA 12.1 application:

```
$ export LD_LIBRARY_PATH=/usr/local/cuda-12.1/compat
$ samples/bin/x86_64/linux/release/deviceQuery
samples/bin/x86_64/linux/release/deviceQuery Starting...

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

 Detected 1 CUDA Capable device(s)
```

(continues on next page)

(continued from previous page)

```
Device 0: "Tesla T4"
CUDA Driver Version / Runtime Version 12.1 / 12.1
CUDA Capability Major/Minor version number: 9.0
[...]

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.1, CUDA Runtime Version
↪= 12.1, NumDevs = 1
Result = PASS
```

3.2.2. Manually Installing from Runfile

The CUDA forward compatibility package files can also be extracted from the appropriate datacenter driver 'runfile' installers (`.run`) available in NVIDIA driver downloads. To do this:

1. Download the latest NVIDIA Data Center GPU driver, and extract the `.run` file using option `-x`.
2. Copy the CUDA libraries, listed at the start of this section, into a user or `root` created directory.
3. Follow your system's guidelines for making sure that the system linker picks up the new libraries.

3.3. Deployment Considerations for Forward Compatibility

3.3.1. Use the Right CUDA Forward Compatibility Package

The CUDA forward compatibility package should be used only in the following situations when forward compatibility is required across major releases.

The CUDA forward compatibility package is named after the highest toolkit that it can support. If you are on the 535 driver but require 12.5 application support, please install the CUDA compatibility package for 12.5.

Note

When performing a full system upgrade, when choosing to install both the toolkit and the driver, remove any forward compatible packages present in the system.

For example, if you are upgrading the driver to 525.60.13 which is the minimum required driver version for the 12.x toolkits, then the CUDA compatibility package is not required in most cases. 11.x and 12.x applications will be supported due to backward compatibility and future 12.x applications will be supported due to minor-version compatibility.

But there are feature restrictions that may make this option less desirable for your scenario - for example: Applications requiring PTX JIT compilation support. Unlike the minor-version compatibility that is defined between CUDA runtime and CUDA driver, forward compatibility is defined between the kernel driver and the CUDA driver, and hence such restrictions do not apply. In order to circumvent the limitation, a forward compatibility package may be used in such scenarios as well.

Table 3: CUDA Application Compatibility Support Matrix

CUDA forward compatibility package	NVIDIA driver									
	535+ (CUDA 12.2)		550+ (CUDA 12.4)		570+ (CUDA 12.8)		575+ (CUDA 12.9)		580+ (CUDA 13.0)	
cuda-compat-13-0	C		C		C		C		N/A	
cuda-compat-12-9	C		C		C		N/A		X	
cuda-compat-12-8	C		C		N/A		X		X	
cuda-compat-12-6	C		C		X		X		X	
cuda-compat-12-5	C		C		X		X		X	
cuda-compat-12-4	C		N/A		X		X		X	
cuda-compat-12-3	C		X		X		X		X	
cuda-compat-12-2	N/A		X		X		X		X	

- C - Compatible
- X - Not compatible
- Branches not listed in the table above are end of life and are not supported targets for compatibility.
- New Feature Branches are not supported targets for CUDA Forward Compatibility.

Examples of how to read this table:

- The CUDA 12.4 forward compatibility package is compatible (C) with driver versions 535. It is not applicatble (N/A) for 550, as 12.4 was paired with 550 and therefore no extra packages are needed.
- The CUDA 12.3 forward compatibility package is not-compatible (X) with driver version 550 as it was released prior to the driver. Binaries created in 12.3 are still subject to the backwards compatibility guarantees described in this document.
- Please note that this table is in reference to use for the compatibility libraries only. A line indicating “X” does not mean that backward or minor version compatibility do not apply to this release. It means that NVIDIA does not provide a compatibility package for this combination.

3.3.2. Feature Exceptions

There are specific features in the CUDA driver that require kernel-mode support and will only work with a newer kernel mode driver. A few features depend on other user-mode components and are therefore also unsupported.

Table 4: Forward-Compatible Feature-Driver Support Matrix

CUDA forward compatibility package	CUDA - OpenGL/Vulkan Interop	cuMemMap* set of functionalities
System Base Installation: >= r580 Driver		
cuda-compat-13-x	No	Yes [1]
System Base Installation: >= r525 Driver		
cuda-compat-12-x	No	Yes [1]
System Base Installation: >= r450 Driver		
cuda-compat-11-x	No	Yes [1]

[1] This relies on `CU_DEVICE_ATTRIBUTE_HANDLE_TYPE_POSIX_FILE_DESCRIPTOR_SUPPORTED` and `CU_DEVICE_ATTRIBUTE_VIRTUAL_ADDRESS_MANAGEMENT_SUPPORTED`, which should be queried if you intend to use the full range of this functionality.

[2] Supported on Red Hat Enterprise Linux operating system version 8.1 or higher.

3.3.3. Check for Compatibility Support

In addition to the CUDA driver and certain compiler components, there are other drivers in the system installation stack (for example, OpenCL) that remain on the old version. The forward-compatible upgrade path is for CUDA only.

A well-written application should use following error codes to determine if a CUDA Forward Compatible Upgrade is supported. System administrators should be aware of these error codes to determine if there are errors in the deployment.

- `CUDA_ERROR_SYSTEM_DRIVER_MISMATCH = 803`. This error indicates that there is a mismatch between the versions of the display driver and the CUDA driver.
- `CUDA_ERROR_COMPAT_NOT_SUPPORTED_ON_DEVICE = 804`. This error indicates that the system was upgraded to run with forward compatibility but the visible hardware detected by CUDA does not support this configuration.

3.4. Deployment Model for Forward Compatibility

There are two models of deployment for the CUDA compatibility package. We recommend the use of the 'shared' deployment mode.

- Shared deployment: Allows sharing the same CUDA compatibility package across installed toolkits in the system. Download and extract the latest forward compatibility package with the highest toolkit version in its name. Set the `LD_LIBRARY_PATH` variable or through an automatic loader (for example, `ld.so.conf`), point to that package. This is the most recommended choice.

  The user can set `LD_LIBRARY_PATH` to include the files installed before running the CUDA application:

  ```
  $ LD_LIBRARY_PATH=/usr/local/cuda-12.1/compat
  ```

- Per-application deployment: Individual applications can choose a package of their choice and place it as part of the Modules system tied to the toolkit and the libraries. Using the Modules system, the admin, or the user, can set up 'module' scripts for each version of each toolkit package, and then load the module script for the toolkit as needed.

  ```
  $ module load cuda/11.0
  ```

  There is an important consideration to the per-application deployment approach: older forward compatibility packages are not supported on new driver versions. Therefore the module load scripts should proactively query the system for whether the compatibility package can be used before loading the files. This is especially critical if there was a full system upgrade. In the cases where the module script cannot use CUDA compatible upgrade, a fallback path to the default system's installed CUDA driver can provide a more consistent experience and this can be achieved using RPATH.

Chapter 4. Conclusion

The CUDA driver maintains backward compatibility to continue support of applications built on older toolkits. Using a compatible minor driver version, applications build on CUDA Toolkit 11 and newer are supported on any driver from within the corresponding major release. Using the CUDA Forward Compatibility package, system administrators can run applications built using a newer toolkit even when an older driver that does not satisfy the minimum required driver version is installed on the system. This forward compatibility allows the CUDA deployments in data centers and enterprises to benefit from the faster release cadence and the latest features and performance of CUDA Toolkit.

CUDA compatibility helps users by:

- Faster upgrades to the latest CUDA releases: Enterprises or data centers with NVIDIA GPUs have complex workflows and require advance planning for NVIDIA driver rollouts. Not having to update the driver for newer CUDA releases can mean that new versions of the software can be made available faster to users without any delays.
- Faster upgrades of the CUDA libraries: Users can upgrade to the latest software libraries and applications built on top of CUDA (for example, math libraries or deep learning frameworks) without an upgrade to the entire CUDA Toolkit or driver. This is possible as these libraries and frameworks do not have a direct dependency on the CUDA runtime, compiler or driver.

Chapter 5. Frequently Asked Questions

This section includes some FAQs related to CUDA compatibility.

- **Does CUDA forward compatible upgrades work intra-branch?**

  Users can upgrade the kernel mode driver within the same branch. Sometimes this may require updating the `cuda-compat-*` package. This use-case is supported only for drivers on LLB and LTS branches of driver for select GPUs.

- **What's the minimum required driver version of a toolkit?**

  Refer to the Release notes.

- **The developer is using PTX code in the application and seeing some errors or issues. What should we do?**

  PTX and application compatibility information can be found in Binary Compatibility.

- **If we build our CUDA application using CUDA 11.0, can it continue to be used with newer NVIDIA drivers (such as CUDA 11.1/R455, 11.x etc.)? Or is it only the other way around?**

  Drivers have always been backwards compatible with CUDA. This means that a CUDA 11.0 application will be compatible with R450 (11.0), R455 (11.1) and beyond. CUDA applications typically statically include all the libraries (for example cudart, CUDA math libraries such as cuBLAS, cuFFT) they need, so they should work on new drivers or CUDA Toolkit installations.

  In other words, since CUDA is backward compatible, existing CUDA applications can continue to be used with newer CUDA versions.

- **What about new features introduced in minor releases of CUDA? How does a developer build an application using newer CUDA Toolkits (such as 11.x) work on a system with a CUDA 11.0 driver (R450)?**

  By using new CUDA versions, users can benefit from new CUDA programming model APIs, compiler optimizations and math library features.

  - A subset of CUDA APIs don't need a new driver and they can all be used without any driver dependencies. For example, async copy APIs introduced in 11.1 do not need a new driver.
  - To use other CUDA APIs introduced in a minor release (that require a new driver), one would have to implement fallbacks or fail gracefully. This situation is not different from what is available today where developers use macros to compile out features based on CUDA versions. Users should refer to the CUDA headers and documentation for new CUDA APIs introduced in a release.

  There are some issues that administrators can advise the application developers to accommodate in their code. Please refer to the Best Practices Guide for further information.

- **Does CUDA compatibility work with containers?**

Yes. CUDA minor version compatibility and CUDA forward compatible upgrade both work when using either NGC Deep Learning Framework containers or using containers that are based on the official CUDA base images. The images include the CUDA compatible upgrade libraries and the NVIDIA Container Toolkit (nvidia-docker2) has logic to correctly load the required libraries.

- **I'm running an NGC container and see this error: "This container was built for NVIDIA driver Release 450.51 or later, but version 418.126.02 was detected and compatibility mode is UNAVAILABLE.". What could be wrong?**

  It is possible you are either running a wrong version of the NVIDIA driver on the system or your system does not have an NVIDIA Data Center GPU.

Chapter 6. Notices

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.

6.1. Trademarks

Copyright