CUDA Compatibility

Table of Contents

List of Figures

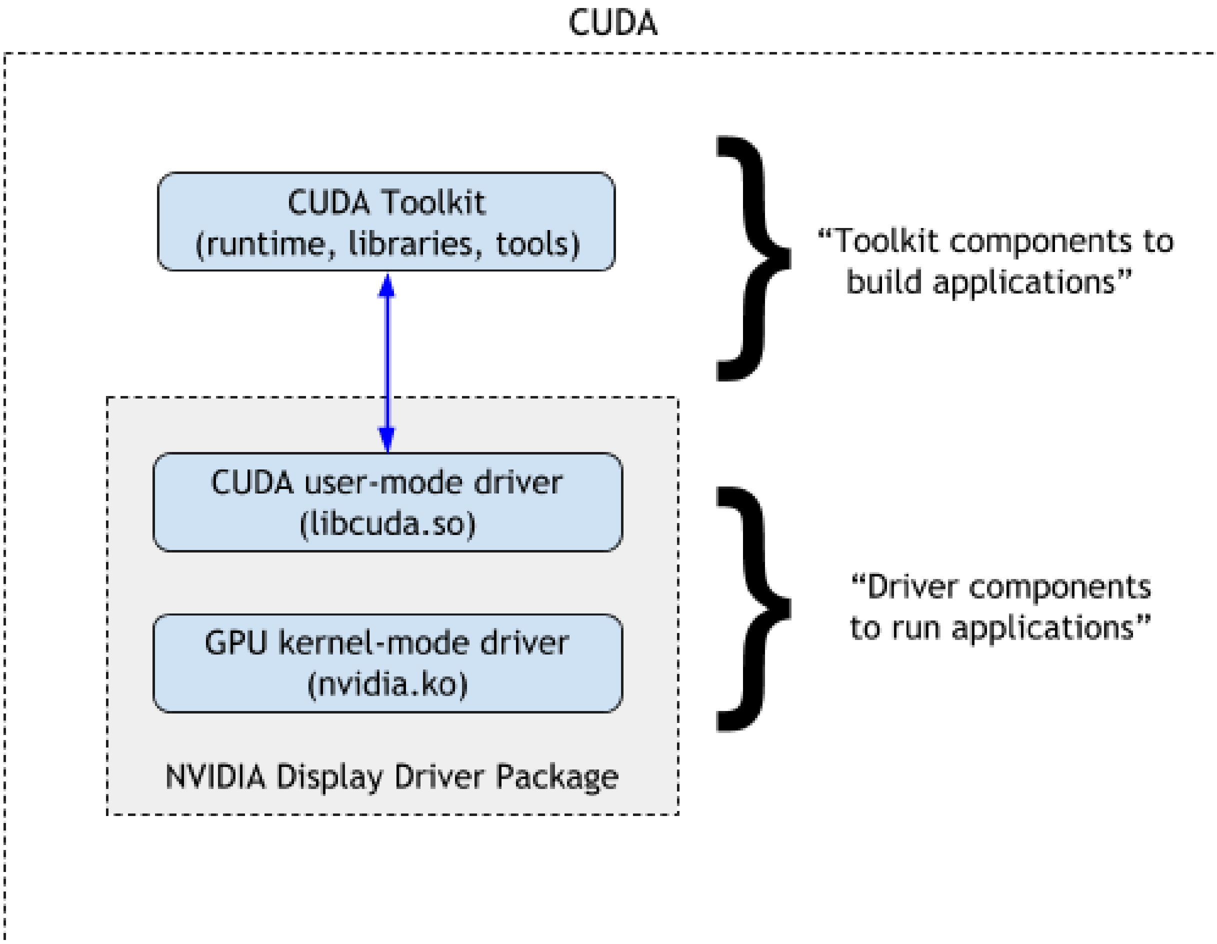
List of Tables

Chapter 1. Why CUDA Compatibility

The **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 display 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.

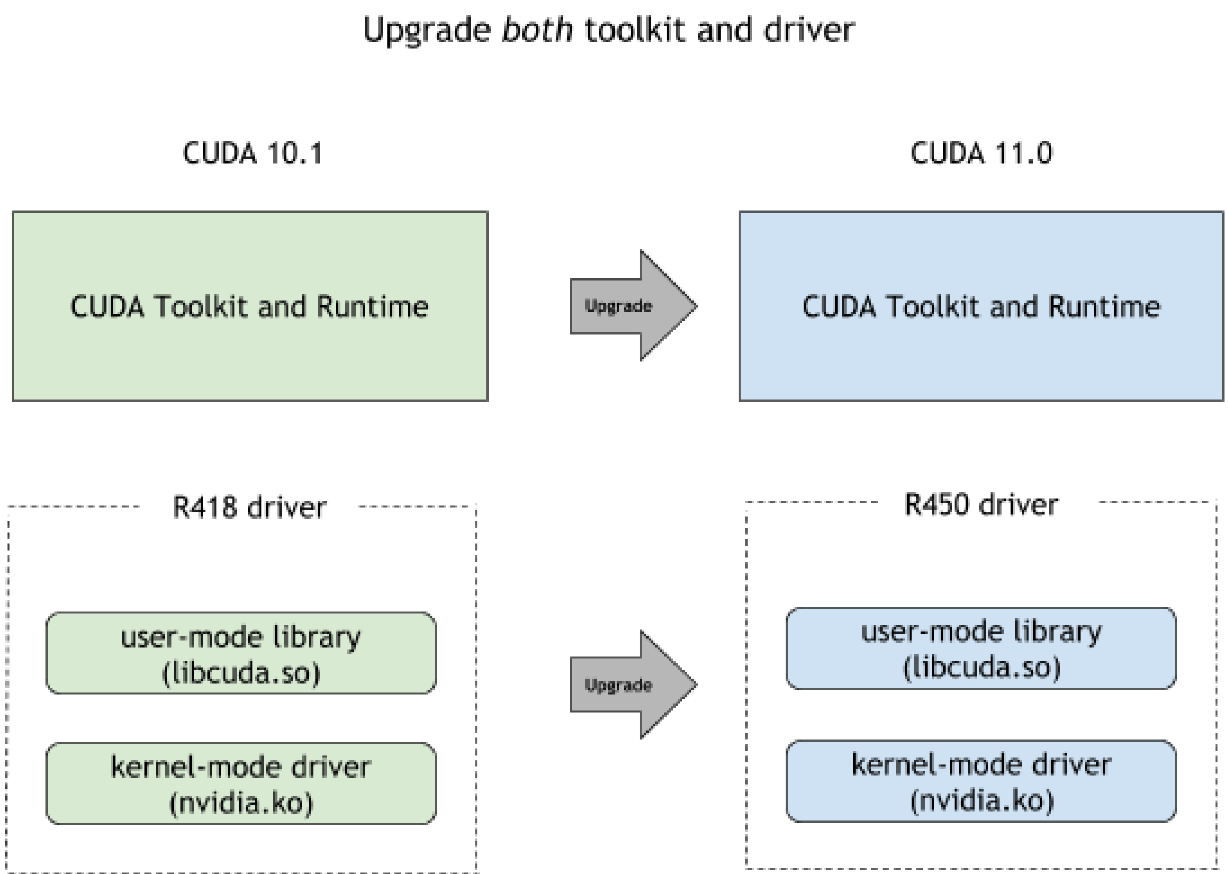
Figure 1. Components of CUDA



Every CUDA toolkit also ships with an NVIDIA display 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 the bleeding edge toolkit and driver capabilities.

Figure 2. CUDA Upgrade Path



But this is not always required. CUDA Compatibility guarantees allow for upgrading only certain components and that will be the focus of the rest of this document. We will see how the upgrade to a new CUDA Toolkit can be simplified to not always require a full system upgrade.

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. Example CUDA Toolkit 11.x Minimum Required Driver Versions (Refer to CUDA Release Notes)

CUDA Toolkit	Linux x86_64 Minimum Required Driver Version	Windows Minimum Required Driver Version
CUDA 11.x	>= 450.80.02*	>=452.39*

* *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.*

While applications built against any of the older CUDA Toolkits always continued to function on newer drivers due to binary backward compatibility, prior to CUDA 11, applications built against newer CUDA Toolkit releases were not supported on older drivers without forward compatibility package (see Forward Compatibility Support Across Major Toolkit Versions).

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.

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

CUDA Toolkit	Linux x86_64 Minimum Required Driver Version	Windows MinimumRequired 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     |
                |-------------------------------+----------------------
+----------------------+

                ...<snip>...

                $ 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)

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

                ...<snip>...

                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 release notes. Note that if the minimum required driver version is not installed in the system, applications will return an error as shown below.

```
$ samples/bin/x86_64/linux/release/deviceQuery
            samples/bin/x86_64/linux/release/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 admins should note two 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, admins should then upgrade the installed driver also.

  Application developers can avoid running into this problem by having the application explicitly check for the availability of features. Refer to the CUDA Compatibility Developers Guide for more details.

- **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 admins have to upgrade the installed driver.

  PTX Developers should refer to the CUDA Compatibility Developers Guide and PTX programming guide in the CUDA C++ Programming Guide for details on this limitation.

2.3. Deployment Considerations for Minor Version Compatibility

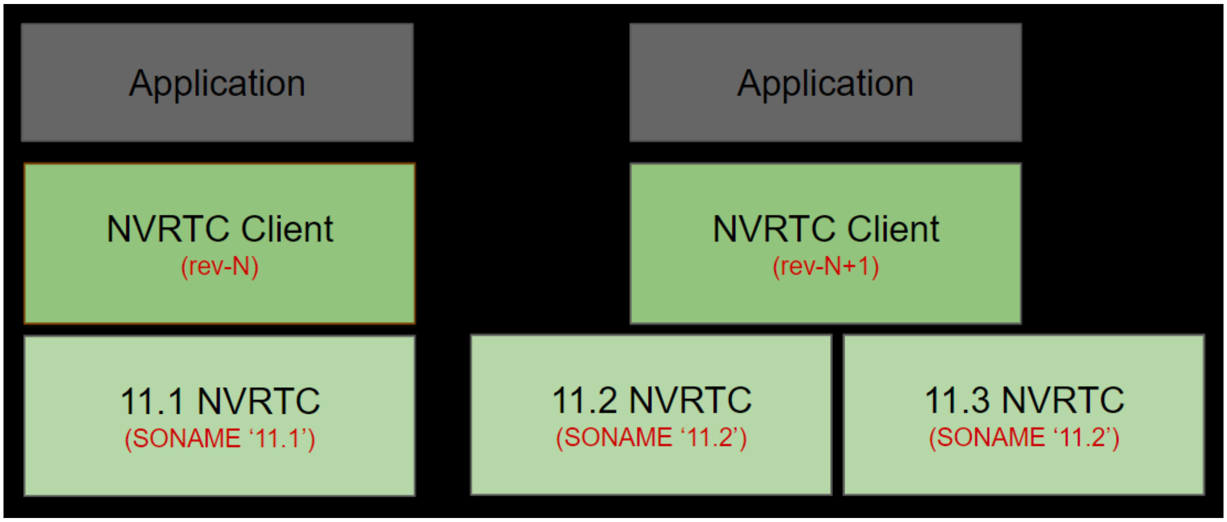
As described, applications that directly rely only on the CUDA runtime can be deployed in the following two scenarios:

1. CUDA driver that's installed on the system is newer than the runtime.
2. CUDA runtime is newer than the CUDA driver on the system but they are from the same major release of CUDA Toolkit.

In scenario 2, system admins should be aware of the aforementioned limitations and should be able to tell why an application may be failing if they run into any issues.

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



However, if an application is unable to leverage the minor version compatibility due to any of the aforementioned reasons, then the Forward Compatibility model can be used as an alternative even though Forward Compatibility is mainly intended for compatibility across major toolkit versions.

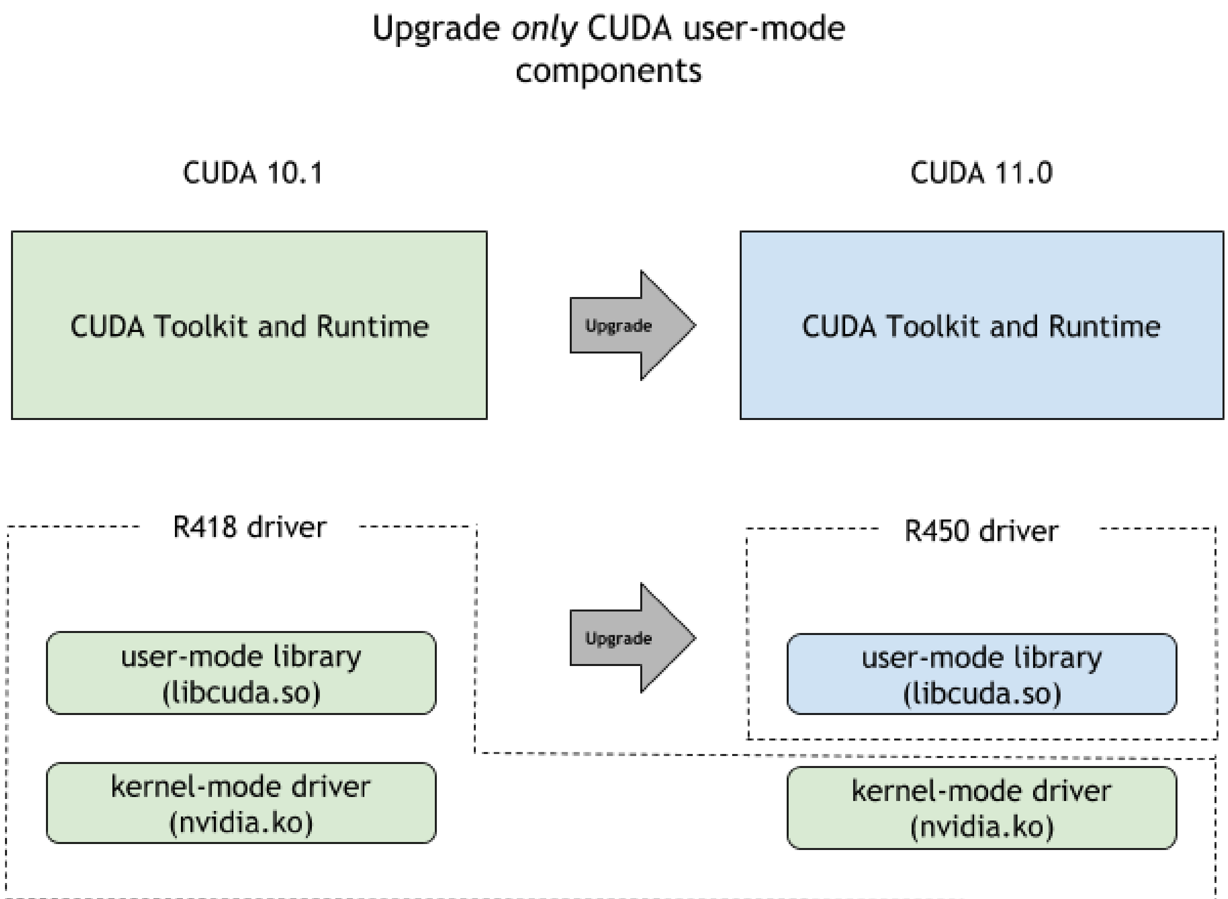
Chapter 3. Forward Compatibility

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



Forward Compatibility is applicable only for systems with NVIDIA Data Center GPUs or select NGC Server Ready SKUs of RTX cards. 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 compat package".

3.2. Installing the Forward Compatibility Package

3.2.1. From Network Repositories or Local Installers

The CUDA compat package is available in the local installers or the CUDA network repositories provided by NVIDIA as `cuda-compat-11.5`.

Install the package on the system using the package installer.

On Ubuntu, for example:

```
$ sudo apt-get install -y cuda-compat-11-5
```

The compat package will then be installed to the versioned toolkit location typically found in the toolkit directory. For example, for 11.5 it will be found in `/usr/local/cuda-11.5/`.

The cuda-compat package consists of the following files:

- `libcuda.so.*` - the CUDA Driver
- `libnvidia-nvvm.so.*` - JIT LTO (CUDA 11.5 and later only)
- `libnvidia-ptxjitcompiler.so.*` - the JIT (just-in-time) compiler for PTX files

These files should be kept together as the CUDA driver is dependent on the `libnvidia-ptxjitcompiler.so.*` of the same version.

Note: This package only provides the files, and does not configure the system.

Example

CUDA Compatibility is installed and the application can now run successfully as shown below. In this example, the user sets `LD_LIBRARY_PATH` to include the files installed by the `cuda-compat-11-5` package.

```
$ sudo apt-get install -y cuda-compat-11-5
Selecting previously unselected package cuda-compat-11-5.
(Reading database ... 220718 files and directories currently installed.)
Preparing to unpack .../cuda-compat-11-5_495.29.05-1_amd64.deb ...
Unpacking cuda-compat-11-5 (495.29.05-1) ...
Setting up cuda-compat-11-5 (495.29.05-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
```

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

```
$ ls -l /usr/local/cuda/compat
total 59456
lrwxrwxrwx 1 root root       12 Oct  1 11:07 libcuda.so -> libcuda.so.1
lrwxrwxrwx 1 root root       20 Oct  1 11:07 libcuda.so.1 -> libcuda.so.495.29.05
-rw-r--r-- 1 root root 22708136 Oct  1 00:16 libcuda.so.495.29.05
lrwxrwxrwx 1 root root       19 Oct  8 10:59 libnvidia-nvvm.so -> libnvidia-
nvvm.so.4
lrwxrwxrwx 1 root root       23 Oct  8 10:15 libnvidia-nvvm.so.4 -> libnvidia-
nvvm.so.4.0.0
-rw-r--r-- 1 root root 26807480 Oct  1 00:29 libnvidia-nvvm.so.4.0.0
lrwxrwxrwx 1 root root       37 Oct  1 11:07 libnvidia-ptxjitcompiler.so.1 ->
 libnvidia-ptxjitcompiler.so.495.29.05
-rw-r--r-- 1 root root 11365720 Sep 30 23:55 libnvidia-ptxjitcompiler.so.495.29.05
```

The user can set `LD_LIBRARY_PATH` to include the files installed before running the CUDA 11.5 application:

```
$ LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH 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)

Device 0: "Tesla V100-PCIE-32GB"
  CUDA Driver Version / Runtime Version          11.5 / 11.5
  CUDA Capability Major/Minor version number:    7.5
  ...<snip>...
```

```
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.5, CUDA Runtime Version
 = 11.5, NumDevs = 1
Result = PASS
```

3.2.2. Manually Installing from Runfile

The cuda-compat 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 three CUDA compatibility upgrade files, 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.

> **Note:** Symlinks under `/usr/local/cuda/compat` need to be created manually when using the runfile installer.

3.3. Deployment Considerations for Forward Compatibility

3.3.1. Use the Right Compat Package

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

The CUDA compat package is named after the highest toolkit that it can support. If you are on the R418 driver but require 11.2 application support, please install the cuda-compat package for 11.2. But 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 450.80.02 which is the minimum required driver version for the 11.2 toolkit, then the cuda-compat package is not required in most cases. 10.x and 11.0 applications will be supported due to backward compatibility and future 11.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

	NVIDIA Kernel Mode Driver - Production Branch					
CUDA Forward Compatible Upgrade	418.40.04+ (CUDA 10.1)	450.36.06+ (CUDA 11.0)	470.57.02+ (CUDA 11.4)	495.29.05+ (CUDA 11.5)	510.39.01+ (CUDA 11.6)	515.43.04+ (CUDA 11.7)
11-7	X	C	C	X*	C	Not Required
11-6	C	C	C		Not required	X
11-5	C	C	C		X	X
11-4	C	C	Not required		X	X
11-3	C	C	X		X	X
11-2	C	C	X		X	X
11-1	C	C	X		X	X
11-0	C	Not required	X		X	X
10-2	C	X	X		X	X
10-1	Not required	X	X		X	X
10-0	X	X	X		X	X

C - Compatible

X – Not compatible

* – New Feature Branches (such as 495.xx) are not supported targets for CUDA Forward Compatibility.

Branches R465, R460, R455, R440, R410, R396, R390 are end of life and are not supported targets for compatibility.

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 Compatible Upgrade	**CUDA – OpenGL/Vulkan Interop**	**POWER9 ATS**	**cuMemMap* set of functionalities**
System Base Installation: 450 (>=.80.02) Driver			
11-x	No	Yes [2]	Yes [1]
System Base Installation: 418 (>=.XX) Driver			
11-x	No	Yes [2]	Yes [1]
10-2	No	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 CUDA Forward Compatible Upgrade is supported. System administrators should be aware of these error codes to determine if there are errors in the deployment.

1. `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.
2. `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 compat package. We recommend the use of the 'shared' deployment mode.

- Shared deployment: Allows sharing the same compat package across installed toolkits in the system. Download and extract the latest forward compatibility package with the highest toolkit version in its name. Using RPATH, or through LD_LIBRARY_PATH 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 11.1 application:

  ```
  $ LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH
  ```
- 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.

- **What is the difference between CUDA forward compatible upgrade and CUDA minor version compatibility? When should users use these features?**

Area	CUDA Forward Compatible Upgrade	CUDA Minor Version Compatibility
Compatibility	Across older drivers from different major release versions of CUDA.	Across minor release versions of CUDA only.
	Between kernel driver and user mode CUDA driver.	Between libraries or runtimes that link to the CUDA driver.
When to use	If you cannot upgrade the kernel driver but need to use the latest CUDA Toolkit.	If you want to support newer applications on older drivers within the same major release family.
GPUs supported	NVIDIA Data Center GPUs only	All GPU products supported
OS distributions supported	Linux only	Windows, Linux
Mobile platforms supported	No	Yes
Features Supported	Some features such as (CUDA-GL interop, Power 9 ATS, cuMemMap APIs) are not supported. These features depend on a new kernel mode driver and thus are not supported. These are explicitly called out in the documentation.	All existing CUDA features (from older minor releases) work. Users may have to incorporate checks in their application when using new features in the minor release (that require a new driver) to ensure graceful errors. More information can be found in the "Handling New CUDA Features" section.
CUDA Releases Supported	All CUDA releases supported through the lifetime of the datacenter driver branch. For example, R418 (CUDA 10.1) EOLs in March 2022 - so all CUDA versions released (including major releases)	Only works within a 'major' release family (e.g. 11.0 through 11.x). Compatibility is not supported across major CUDA releases.

Area	CUDA Forward Compatible Upgrade	CUDA Minor Version Compatibility
	during this timeframe are supported.	
Application includes PTX or uses NVRTC	No additional workflow required.	Users should use the new PTX static library to rebuild binaries. Refer to the workflow section for more details.
Requires administrator involvement	Depends on the deployment. Users can also set up `LD_LIBRARY_PATH` with the new libraries from the `cuda-compat-*` package.	Not required.

- **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.

- **Which GPUs are supported by the driver ?**

  The CUDA compatible upgrade is meant to ease the management of large production systems for enterprise customers. 11.4 UMD (User Mode Driver) and later will extend forward compatibility support to select NGC Ready NVIDIA RTX boards. Prior to that forward compatibility will be supported only on NVIDIA Data Center cards.

  It's important to note that HW support is defined by the kernel mode driver and as such, newer CUDA drivers on their own will not enable new HW support. Refer to the following table to determine which hardware is supported by your system.

Hardware Generation	Compute Capability	Driver					
		418.40.04+	450.36.06+	470.57.02+	495.29.05+	510.39.01+	515.43.04+
NVIDIA Ampere GPU Arch.	8.x	No	Yes	Yes	Yes	Yes	Yes
Turing	7.5	Yes	Yes	Yes	Yes	Yes	Yes
Volta	7.x	Yes	Yes	Yes	Yes	Yes	Yes
Pascal	6.x	Yes	Yes	Yes	Yes	Yes	Yes
Maxwell	5.x	Yes	Yes	Yes	Yes	Yes	Yes
Kepler	3.x	Yes	Yes	Yes	No	No	No

- **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 the CUDA Compatibility Developer's Guide CUDA C++ Programming Guide.

- **If we build our CUDA application using CUDA 11.0, can it continue to be used with newer NVIDIA drivers (e.g. 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 is the minimum CUDA 11.x driver that supports the CUDA minor version compatibility?**

  The minimum driver version required is 450.80.02.

- **What about new features introduced in minor releases of CUDA? How does a developer build an application using newer CUDA Toolkits (e.g. 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 admins 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.

Trademarks

Copyright

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