CUDA on WSL User Guide

User Guide

DG-05603-001_v11.7 | May 2022

Table of Contents

Chapter 1. NVIDIA GPU Accelerated Computing on WSL 2

WSL or Windows Subsystem for Linux is a Windows feature that enables users to run native Linux applications, containers and command-line tools directly on Windows 11 and later OS builds. CUDA support in this user guide is specifically for WSL 2, which is the second generation of WSL that offers the following benefits

- Linux applications can run as is in WSL 2. WSL 2 is characteristically a VM with a Linux WSL Kernel in it that provides full compatibility with mainstream Linux kernel allowing support for native Linux applications including popular Linux distros.
- Faster file system support and that's more performant.
- WSL 2 is tightly integrated with the Microsoft Windows operating system, which allows it to run Linux applications alongside and even interop with other Windows desktop and modern store apps.

For the rest of this user guide, WSL and WSL 2 may be used interchangeably.

Typically, developers working across both Linux and Windows environments have a very disruptive workflow. They either have to:

- Use different systems for Linux and Windows, or
- Dual Boot i.e. install Linux and Windows in separate partitions on the same or different hard disks on the system and boot to the OS of choice.

In both cases, developers have to stop all the work and then switch the system or reboot. Also this has historically restricted the development of seamless, well integrated tools and software systems across two dominant ecosystems.

WSL enables users to have a seamless transition across the two environments without the need for a resource intensive traditional virtual machine and to improve productivity and develop using tools and integrate their workflow. More importantly WSL 2 enables applications that were hitherto only available on Linux to be available on Windows. WSL 2 support for GPU allows for these applications to benefit from GPU accelerated computing and expands the domain of applications that can be developed on WSL 2.

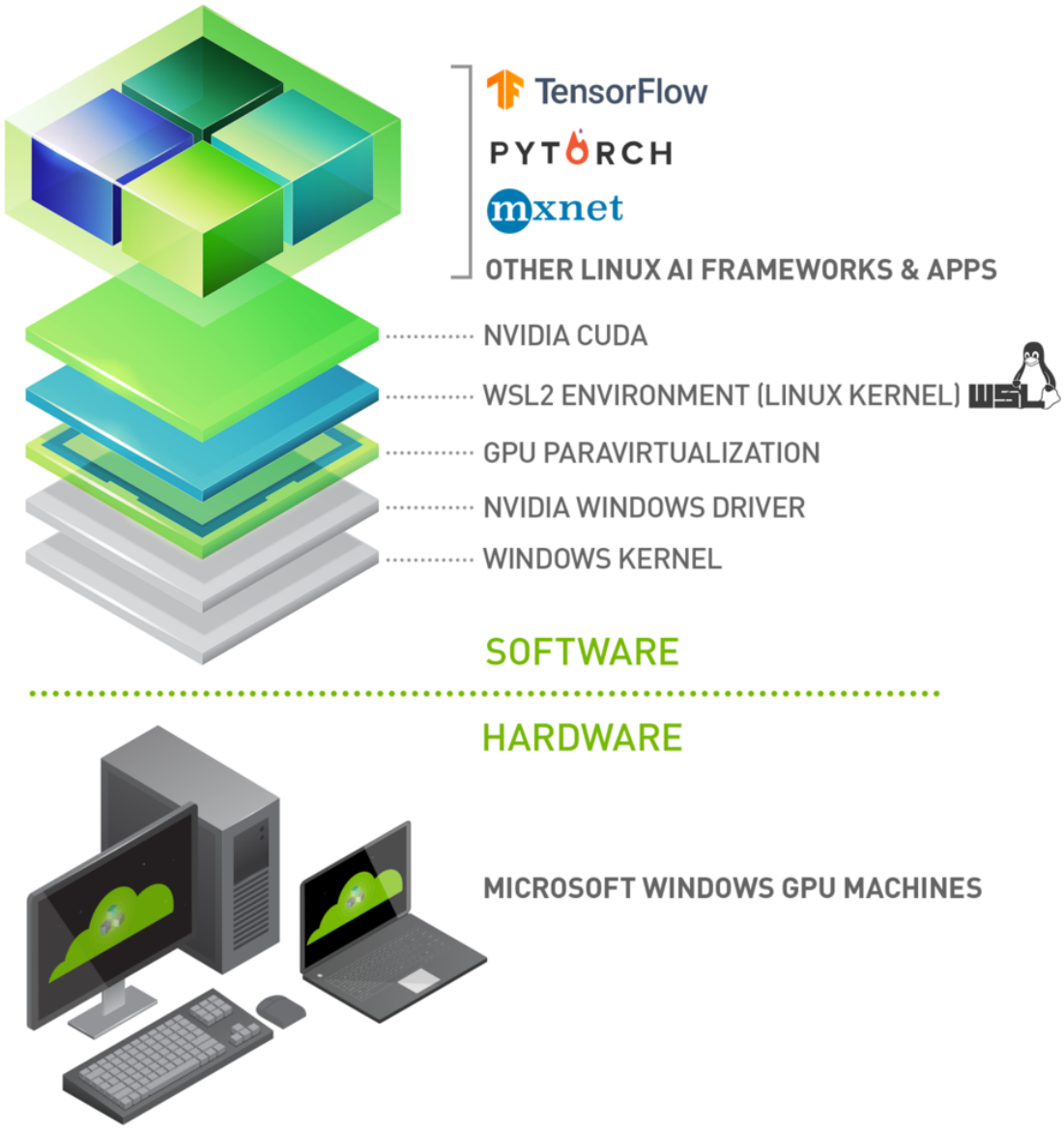
With NVIDIA CUDA support for WSL 2, developers can leverage NVIDIA GPU accelerated computing technology for data science, machine learning and inference on Windows through WSL. GPU acceleration also serves to bring down the performance overhead of running an

application inside a WSL like environment close to near-native by being able to pipeline more parallel work on the GPU with less CPU intervention.

NVIDIA driver support for WSL 2 includes not only CUDA but also DirectX and Direct ML support. For some helpful examples, see https://docs.microsoft.com/en-us/windows/win32/direct3d12/gpu-tensorflow-wsl.

WSL 2 is a key enabler in making GPU acceleration to be seamlessly shared between Windows and Linux applications on the same system a reality. This offers flexibility and versatility while also serving to open up GPU accelerated computing by making it more accessible.

Figure 1. Illustration of the possibilities with NVIDIA CUDA software stack on WSL 2



This document describes a workflow for getting started with running CUDA applications or containers in a WSL 2 environment.

1.1. NVIDIA Compute Software Support on WSL 2

This table captures the readiness and suggested software versions for NVIDIA software stack for WSL 2.

Package	Suggested Versions	Installation
NVIDIA Windows Driver x86	Use the latest Windows x86 production driver. R495 and later windows will have CUDA support for WSL 2. NVIDIA-SMI will have a Limited Feature Set on WSL 2. Legacy CUDA IPC APIs are support from R510.	Windows x86 drivers can be directly downloaded from here for WSL 2 support on Pascal or later GPUs.
Docker support	Supported. NVIDIA Container Toolkit - Minimum versions - v2.6.0 with libnvidia-container - 1.5.1+ CLI and Docker Desktop Supported.	Refer to the NVIDIA Docker Deployment Guide for Linux x86.
CUDA Toolkit and CUDA Developer Tools	Preview Support Compute Sanitizer - Pascal and later Nsight Systems CLI, and CUPTI (Trace) - Volta and later Developer tools - Debuggers and Profilers are not supported yet	Latest Linux CUDA toolkit package - WSL-Ubuntu from 11.x releases can be downloaded from here.
RAPIDS	22.04 or later 1.10 - Experimental Support for single GPU.	https://docs.rapids.ai/notices/rgn0024/
NCCL	2.12 or later 1.4+	Refer to the NCCL Installation guide for Linux x86.

Chapter 2. Getting Started with CUDA on WSL 2

To get started with running CUDA on WSL, complete these steps in order:

2.1. Step 1: Install NVIDIA Driver for GPU Support

- Install NVIDIA GeForce Game Ready or NVIDIA RTX Quadro Windows 11 display driver on your system with a compatible GeForce or NVIDIA RTX/Quadro card from https://www.nvidia.com/Download/index.aspx?lang=en-us. rRefer to the system requirements in the Appendix.)

Note: This is the only driver you need to install. Do not install any Linux display driver in WSL.

2.2. Step 2: Install WSL 2

1. Launch your preferred Windows Terminal / Command Prompt / Powershell and install WSL:

```
wsl.exe --install
```

2. Ensure you have the latest WSL kernel:

```
wsl.exe --update
```

2.3. Step 3: Set Up a Linux Development Environment

From a Windows terminal, enter WSL:

```
C:\> wsl.exe
```

The default distro is Ubuntu. To update the distro to your favorite distro from the command line and to review other WSL commands, refer to the following resources:

- https://docs.microsoft.com/en-us/windows/wsl/install
- https://docs.microsoft.com/en-us/windows/wsl/basic-commands

From this point you should be able to run any existing Linux application which requires CUDA. Do not install any driver within the WSL environment. For building a CUDA application, you will need CUDA Toolkit. Read the next section for further information.

Chapter 3. CUDA Support for WSL 2

The latest NVIDIA Windows GPU Driver will fully support WSL 2. With CUDA support in the driver, existing applications (compiled elsewhere on a Linux system for the same target GPU) can run unmodified within the WSL environment.

To compile new CUDA applications, a CUDA Toolkit for Linux x86 is needed. CUDA Toolkit support for WSL is still in preview stage as developer tools such as debugger and profilers are not available yet. However, CUDA application development is fully supported in the WSL2 environment, as a result, users should be able to compile new CUDA Linux applications with the latest CUDA Toolkit for x86 Linux.

Once a Windows NVIDIA GPU driver is installed on the system, CUDA becomes available within WSL 2. The CUDA driver installed on Windows host will be stubbed inside the WSL 2 as `libcuda.so`, therefore **users must not install any NVIDIA GPU Linux driver within WSL 2**. One has to be very careful here as the default CUDA Toolkit comes packaged with a driver, and it is easy to overwrite the WSL 2 NVIDIA driver with the default installation. We recommend developers to use a separate CUDA Toolkit for WSL 2 (Ubuntu) available here to avoid this overwriting. This WSL-Ubuntu CUDA toolkit installer will not overwrite the NVIDIA driver that was already mapped into the WSL 2 environment. To learn how to compile CUDA applications, please read the CUDA documentation for Linux.

First, remove the old GPG key:

```
sudo apt-key del 7fa2af80
```

Option 1: Installation of Linux x86 CUDA Toolkit using WSL-Ubuntu Package – Recommended

The CUDA WSL-Ubuntu local installer does not contain the NVIDIA Linux GPU driver, so by following the steps below, you will be able to get just the CUDA toolkit installed on WSL.

```
wget https://developer.download.nvidia.com/compute/cuda/repos/wsl-ubuntu/x86_64/
cuda-wsl-ubuntu.pin
sudo mv cuda-wsl-ubuntu.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget https://developer.download.nvidia.com/compute/cuda/11.7.0/local_installers/
cuda-repo-wsl-ubuntu-11-7-local_11.7.0-1_amd64.deb
sudo dpkg -i cuda-repo-wsl-ubuntu-11-7-local_11.7.0-1_amd64.deb
sudo apt-get update
sudo apt-get -y install cuda
```

Option 2: Installation of Linux x86 CUDA Toolkit using Meta Package

If you installed the toolkit using the WSL-Ubuntu package, please skip this section. Meta packages do not contain the driver, so by following the steps below, you will be able to get just the CUDA toolkit installed on WSL.

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64/
cuda-ubuntu2004.pin
sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget https://developer.download.nvidia.com/compute/cuda/11.7.0/local_installers/
cuda-repo-ubuntu2004-11-7-local_11.7.0-515.43.04-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2004-11-7-local_11.7.0-515.43.04-1_amd64.deb
wget https://developer.download.nvidia.com/compute/cuda/repos/wsl-ubuntu/x86_64/
cuda-keyring_1.0-1_all.deb
sudo dpkg -i cuda-keyring_1.0-1_all.deb
sudo apt-get update
sudo apt-get -y install cuda-toolkit-11-7
```

CUDA 11.7 is used for illustration purposes. For other CUDA versions, refer to https://developer.nvidia.com/cuda-downloads.

The installation instructions for the CUDA Toolkit can be found in the CUDA Toolkit download page for each installer. But DO NOT choose the "`cuda`", "`cuda-11-7`", or "`cuda-drivers`" meta-packages under WSL 2 as these packages will result in an attempt to install the Linux NVIDIA driver under WSL 2. Install the `cuda-toolkit-11-x` metapackage only.

You can also install other components of the toolkit by choosing the right meta-package.

Chapter 4. WSL 2 Support Constraints

- WSL 2 GPU acceleration will be available on Pascal and later GPU architecture on both GeForce and Quadro product SKUs in WDDM mode. It will not be available on Quadro GPUs in TCC mode or Tesla GPUs yet.
- Ensure you are on the latest WSL Kernel or at least 4.19.121+. We recommend 5.10.16.3 or later for better performance and functional fixes.
- If you are on Windows 11, you no longer need to be on Windows Insider Program to use WSL. Refer to Windows11 system requirements in the Microsoft Blog.
- If you are continuing to use Windows 10, see Windows Insider Preview and Windows 10 Support.

4.1. Known Limitations for Linux CUDA Applications

The following table lists the known limitations on WSL 2 that may affect CUDA applications that use some of these features that are fully supported on Linux.

Limitations	Impact
Maxwell GPU is not supported.	Maxwell GPUs are not officially supported in WSL 2, but it may still work. Pascal and later GPU is recommended.
Unified Memory - Full Managed Memory Support is not available on Windows native and therefore WSL 2 will not support it for the foreseeable future.	UVM full features will not be available and therefore applications relying on UVM full features may not work. If your application is using Managed Memory, your application could see reduced performance and high system memory usage. Concurrent CPU/GPU access is not supported. CUDA queries will say whether it is supported or not and applications are expected to check this.
Pinned system memory (example: System memory that an application makes resident for GPU accesses) availability for applications is limited.	For example, some deep learning training workloads, depending on the framework, model and dataset size used, can exceed this limit and may not work.

Limitations	Impact
Root user on bare metal (not containers) will not find nvidia-smi at the expected location.	Use `/usr/lib/wsl/lib/nvidia-smi` or manually add `/usr/lib/wsl/lib/` to the PATH).
With the NVIDIA Container Toolkit for Docker 19.03, only `--gpus all` is supported.	On multi-GPU systems it is not possible to filter for specific GPU devices by using specific index numbers to enumerate GPUs.

4.2. Features Not Yet Supported

The following table lists the set of features that are currently not supported.

Limitations	Impact
CUDA Developer tools - Debuggers and Profilers	Developers who require debugging support are encouraged to find alternatives in the meanwhile.
NVML (nvidia-smi) does not support all the queries yet.	GPU utilization, active compute process are some queries that are not yet supported. Modifiable state features (ECC, Compute mode, Persistence mode) will not be supported.
OpenGL-CUDA Interop is not yet supported.	Applications relying on OpenGL will not work.

Appendix A. Appendix

A.1. Windows Insider Preview and Windows 10 Support

- If you are on Windows 11 please skip this section. Windows 11 is generally available to the public and therefore does not require special registration. All the instructions at the beginning of this user guide were mainly focused toward Windows 11 users.
- If you are looking to use WSL 2 on Windows 10 or to be on the bleeding edge of WSL 2 development, you may want to register for the Windows Insider Program and choose the appropriate flighting channel (previously fast rings) and get the latest build for your needs.
- Learn more on Releasing Windows 10 Build 19043.1263 (21H1) to Release Preview Channel.
- You can check your build version number by running `winver` via the Run command.

A.2. Troubleshooting

A.2.1. Container Runtime Initialization Errors

In some cases, when running a Docker container, you may encounter `nvidia-container-cli : initialization error`:

```
$ sudo docker run --gpus all nvcr.io/nvidia/k8s/cuda-sample:nbody nbody -gpu -
benchmark
docker: Error response from daemon: OCI runtime create failed:
 container_linux.go:349: starting container process caused "process_linux.go:449:
 container init caused \"process_linux.go:432: running prestart hook 0 caused
 \\\"error running hook: exit status 1, stdout: , stderr: nvidia-container-cli:
 initialization error: driver error: failed to process request\\\\n\\\"\"": unknown.
ERRO[0000] error waiting for container: context canceled
```

This usually indicates that the right Windows OS build or Microsoft Windows Insider Preview Builds (Windows 10 only), WSL 2, NVIDIA drivers and NVIDIA Container Toolkit may not be installed correctly. Review the known issues and changelog sections to ensure the right versions of the driver and container toolkit are installed.

Ensure you have followed through the steps listed under Setup under Running CUDA containers; especially ensure that the `docker` daemon is still running.

```
$ sudo service docker stop
$ sudo service docker start
```

Or start the daemon directly and see if that resolves the issue:

```
$ sudo dockerd
```

If you are still running into this issue, use the `dxdiag` tools from the Run dialog and provide the diagnostic logs to NVIDIA by posting in the Developer Forums or by filing a report.

You can also use the CUDA on WSL 2 Developer Forums to get in touch with NVIDIA product and engineering teams for help.

A.2.2. Checking WSL Kernel Version

1. Ensure you have the latest kernel by running the following command in PowerShell:

```
$ wsl cat /proc/version

Linux version 5.10.16.3-microsoft-standard-WSL2
(x86_64-msft-linux-gcc (GCC) 9.3.0, GNU ld (GNU Binutils) 2.34.0.20200220) #1 SMP
 Fri Apr 2 22:23:49 UTC 2021
```

2. If you don't have the latest WSL kernel, you will see the following blocking warning upon trying to launch a Linux distribution within the WSL 2 container:

A.3. Traditional Virtual Machines vs WSL 2

Whether to efficiently use hardware resources or to improve productivity, virtualization is a more widely used solution in both consumer and enterprise space. There are different types of virtualizations, and it is beyond the scope of this document to delve into the specifics. But traditional virtualization solutions require installation and setup of a virtualization management software to manage the guest virtual machines.

Although WSL 2 is itself a Virtual Machine, unlike traditional VMs it is easy to setup as it is provided by the host operating system provider and is quite lightweight. Applications running within WSL see less overhead compared to traditional VMs especially if they require access to the hardware or perform privileged operations compared to when run directly on the system. This is especially important for GPU accelerated workload. While VMs allow applications to be run unmodified, due to constraints from setup and performance overhead, they are not the best option in many situations.

A.4. Containers vs WSL 2

While a VM provides a secure self-contained, execution environment with a complete user space for the application, containers enable application composability without the overhead of VMs. Containers compose all the dependencies of the applications such as libraries, files etc.,

to be bundled together for development and easy and predictable deployment. Containers run on the operating system that is installed on the system directly and therefore do not provide full isolation from other containers like a VM does, but keeps overhead negligible as a result.

To learn more about differences between VMs and containers, refer to https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm.

Notice

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

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

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

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

OpenCL

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

Trademarks

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

Copyright

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