



DGL

Release Notes

Table of Contents

Chapter 1. DGL Overview.....	1
Chapter 2. Pulling A Container.....	2
Chapter 3. Running DGL.....	3
Chapter 4. DGL Release 25.06.....	5
Chapter 5. DGL Release 25.05.....	6
Chapter 6. DGL Release 25.04.....	9
Chapter 7. DGL Release 25.03.....	10
Chapter 8. DGL Release 25.02.....	13
Chapter 9. DGL Release 25.01.....	14
Chapter 10. DGL Release 24.12.....	17
Chapter 11. DGL Release 24.11.....	18
Chapter 12. DGL Release 24.10.....	21
Chapter 13. DGL Release 24.09.....	22
Chapter 14. DGL Release 24.08.....	25
Chapter 15. DGL Release 24.07.....	27
Chapter 16. DGL Release 24.06.....	30
Chapter 17. DGL Release 24.05.....	31
Chapter 18. DGL Release 24.04.....	34
Chapter 19. DGL Release 24.03.....	37
Chapter 20. DGL Release 24.02.....	39
Chapter 21. DGL Release 24.01.....	40
Chapter 22. DGL Release 23.12.....	43
Chapter 23. DGL Release 23.11.....	44
Chapter 24. DGL Release 23.09.....	47
Chapter 25. DGL Release 23.07.....	49

Chapter 1. DGL Overview

The NVIDIA® Deep Learning SDK accelerates widely-used deep learning frameworks such as DGL. DGL is an easy-to-use, high performance and scalable Python package for deep learning on graphs. DGL is framework agnostic, meaning if a deep graph model is a component of an end-to-end application, the rest of the logics can be implemented in any major frameworks, such as PyTorch, Apache MXNet, or TensorFlow.

The DGL container consists of the latest versions of DGL and PyTorch, their dependencies, and the latest performance optimizations to run your code with GPU-accelerated performance immediately.

The GPU-accelerated NVIDIA DGL containers help developers and data scientists who work with Graph Neural Networks (GNN) on large, heterogeneous graphs with billions of edges. These containers allow developers to work more efficiently in an integrated, GPU-accelerated environment that combines DGL and PyTorch. Instead of using home-grown software that is expensive to maintain, developers can use end-to-end GNN solutions through tested, validated, and supported containers.

This document describes the key features, software enhancements and improvements, known issues, and how to run this container.

Chapter 2. Pulling A Container

About this task

Release 24.01 is based on [CUDA 12.3.2](#), which requires [NVIDIA Driver](#) release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the [CUDA Application Compatibility](#) topic. For more information, see [CUDA Compatibility and Upgrades](#).

Before you can pull a container from the NGC container registry:

- ▶ Install Docker.
 - ▶ For NVIDIA DGX™ users, see [Preparing to use NVIDIA Containers Getting Started Guide](#).
 - ▶ For non-DGX users, see NVIDIA® GPU Cloud™ (NGC) container registry [installation documentation](#) based on your platform.
- ▶ Ensure that you have an NGC API Key to log in to the NGC container registry.

Refer to [NGC Getting Started Guide](#) for more information.

Chapter 3. Running DGL

Before you can run an NGC deep learning framework container, your Docker environment must support NVIDIA GPUs. To run a container, issue the appropriate command as explained in [Running A Container](#) and specify the registry, repository, and tags.

On a system with GPU support for NGC containers, when you run a container, the following occurs when running a container:

- ▶ The Docker engine loads the image into a container that runs the software.
- ▶ You define the container's runtime resources by including the additional flags and settings that are used with the command.

These flags and settings are described in [Running A Container](#).

- ▶ The GPUs are explicitly defined for the Docker® container, which defaults to all GPUs, but can be specified by using the `NVIDIA_VISIBLE_DEVICES` environment variable.

For more information, refer to the [nvidia-docker documentation](#).



Note: Starting in Docker 19.03, complete the steps below.

The method implemented in your system depends on the DGX OS version installed (for DGX systems), the specific NGC Cloud Image provided by a Cloud Service Provider, or the software that you have installed in preparation for running NGC containers on TITAN PCs, Quadro PCs, or vGPUs.

1. Issue the command for the applicable release of the container that you want.

The following command assumes you want to pull the latest container:

```
docker pull nvcr.io/nvidia/dgl:<xx.xx>-py3
```

2. Open a command prompt and paste the `pull` command.

Ensure that the pull process completes successfully before proceeding to step 3.

3. Run the container image.

To run the container, select one of the following modes:

- ▶ **Interactive mode:**

If you have **Docker 19.03 or later**, a typical command to launch the container is:

```
docker run --gpus all -it --rm -v local_dir:container_dir nvcr.io/nvidia/dgl:<xx.xx>-py3
```

If you have **Docker 19.02 or earlier**, a typical command to launch the container is:

```
nvidia-docker run -it --rm -v local_dir:container_dir nvcr.io/nvidia/dgl:<xx.xx>-py3
```

► **Non-interactive mode:**

If you have **Docker 19.03 or later**, a typical command to launch the container is:

```
docker run --gpus all --rm -v local_dir:container_dir nvcr.io/nvidia/dgl:<xx.xx>-py3  
<command>
```

If you have **Docker 19.02 or earlier**, a typical command to launch the container is:

```
nvidia-docker run --rm -v local_dir:container_dir nvcr.io/nvidia/dgl:<xx.xx>-py3  
<command>
```



Note: If you use multiprocessing for multi-threaded data loaders, the default shared memory segment size that the container runs with might not be enough. To increase the shared memory size, run one of the following commands:

```
--ipc=host
```

or

```
--shm-size=<requested memory size>
```

in the command line to

```
docker run --gpus all
```

You might want to pull data and model descriptions from locations outside the container for use by DGL or save results to locations outside the container. The easiest method is to mount one or more host directories as Docker [data volumes](#).

Chapter 4. DGL Release 25.06

There is no DGL container in DLFW release 25.06.

Chapter 5. DGL Release 25.05

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.5.
- ▶ RAPIDS 24.10
- ▶ This container also contains WholeGraph 24.08 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.9.0.43](#)
- ▶ [NVIDIA cuBLAS 12.9.0.13](#)
- ▶ [NVIDIA cuDNN 9.10.1.3](#)
- ▶ [NVIDIA NCCL 2.26.5](#)
- ▶ [Apex](#)
- ▶ [rdma-core 50.0](#)
- ▶ NVIDIA HPC-X 2.21
- ▶ [OpenMPI 4.1.7](#)
- ▶ GDRCopy 2.4.1
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2025.2.0.11](#)
- ▶ [Nsight Systems 2025.3.1.90](#)
- ▶ [NVIDIA TensorRT™ 10.10.0.31](#)
- ▶ [Torch-TensorRT 2.8.0a0](#)
- ▶ [NVIDIA DALI® 1.49](#)

- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v2.3
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 25.05 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ Starting with the 24.07 release, the DGL container supports distributed in-memory sampling and feature gathering with WholeGraph. It can be easily integrated with DGL GraphBolt dataloader, enabling out-of-the-box distributed GNN training.
- ▶ DGL GraphBolt does not depend on the deprecated torchdata package anymore.
- ▶ DGL GraphBolt changes the CUDA memory allocation configuration to reduce memory footprint.

Announcements

- ▶ Starting with the 25.01 release, the NVIDIA Optimized Deep Learning Framework containers are optimized for Blackwell GPU architectures.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
25.05	24.04	NVIDIA CUDA 12.9.0	2.5	25.05
25.01		NVIDIA CUDA 12.8.0	2.5	25.01
24.11		NVIDIA CUDA 12.6.3	2.5	24.11

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.09	22.04	NVIDIA CUDA 12.6.1	2.4	24.09
24.07		NVIDIA CUDA 12.5.1	2.3	24.07
24.05		NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for edata or ndata, a `DGLError` will occur.

Chapter 6. DGL Release 25.04

There is no DGL container in DLFW release 25.04.

Chapter 7. DGL Release 25.03

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.4.
- ▶ RAPIDS 25.02
- ▶ This container also contains WholeGraph 24.08 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ Built on [PyTorch 25.03](#).

GPU Requirements

Release 25.03 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ No new features.

Announcements

- ▶ None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
25.03	24.04	NVIDIA CUDA 12.8.1	2.5	25.03
25.01		NVIDIA CUDA 12.8.0	2.5	25.01
24.11		NVIDIA CUDA 12.6.3	2.5	24.11
24.09	22.04	NVIDIA CUDA 12.6.1	2.4	24.09
24.07		NVIDIA CUDA 12.5.1	2.3	24.07
24.05		NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLError` will occur.

Chapter 8. DGL Release 25.02

There is no DGL container in DLFW release 25.02.

Chapter 9. DGL Release 25.01

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.5.
- ▶ RAPIDS 24.10
- ▶ This container also contains WholeGraph 24.08 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.8.0.038](#)
- ▶ [NVIDIA cuBLAS 12.8.3.14](#)
- ▶ [NVIDIA cuDNN 9.7.0.66](#)
- ▶ [NVIDIA NCCL 2.25.1](#)
- ▶ [Apex](#)
- ▶ [rdma-core 50.0](#)
- ▶ NVIDIA HPC-X 2.21
- ▶ [OpenMPI 4.1.7](#)
- ▶ GDRCopy 2.4.1
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2025.1.0.14](#)
- ▶ [Nsight Systems 2024.6.2.225](#)
- ▶ [NVIDIA TensorRT™ 10.8.0.40](#)
- ▶ [Torch-TensorRT 2.6.0.a0](#)
- ▶ [NVIDIA DALI® 1.45](#)

- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.14
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 25.01 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ Starting with the 24.07 release, the DGL container supports distributed in-memory sampling and feature gathering with WholeGraph. It can be easily integrated with DGL GraphBolt dataloader, enabling out-of-the-box distributed GNN training.
- ▶ DGL GraphBolt does not depend on the deprecated torchdata package anymore.
- ▶ DGL GraphBolt changes the CUDA memory allocation configuration to reduce memory footprint.

Announcements

- ▶ Starting with the 25.01 release, the NVIDIA Optimized Deep Learning Framework containers are optimized for Blackwell GPU architectures.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
25.01	24.04	NVIDIA CUDA 12.8.0	2.5	25.01
24.11		NVIDIA CUDA 12.6.3	2.5	24.11
24.09	22.04	NVIDIA CUDA 12.6.1	2.4	24.09

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.07		NVIDIA CUDA 12.5.1	2.3	24.07
24.05		NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.data.loading.DataLoader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLEError` will occur.

Chapter 10. DGL Release 24.12

There is no DGL container in DLFW release 24.12.

Chapter 11. DGL Release 24.11

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.5.
- ▶ RAPIDS 24.10
- ▶ This container also contains WholeGraph 24.08 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.6.3](#)
- ▶ [NVIDIA cuBLAS 12.6.4.1](#)
- ▶ [NVIDIA cuDNN 9.5.1.17](#)
- ▶ [NVIDIA NCCL 2.23.4](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.21
- ▶ [OpenMPI 4.1.7](#)
- ▶ GDRCopy 2.4.1
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2024.3.2.3](#)
- ▶ [Nsight Systems 2024.6.1.90](#)
- ▶ [NVIDIA TensorRT™ 10.6.0.26](#)
- ▶ [Torch-TensorRT 2.6.0.a0](#)
- ▶ [NVIDIA DALI® 1.43](#)

- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.12
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 24.11 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ Starting with the 24.07 release, the DGL container supports distributed in-memory sampling and feature gathering with WholeGraph. It can be easily integrated with DGL GraphBolt dataloader, enabling out-of-the-box distributed GNN training.
- ▶ DGL GraphBolt does not depend on the deprecated torchdata package anymore.
- ▶ DGL GraphBolt changes the CUDA memory allocation configuration to reduce memory footprint.

Announcements

Volta GPU compute architecture support will be discontinued by the 25.01 release..

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.11	24.04	NVIDIA CUDA 12.6.3	2.5	24.11
24.09	22.04	NVIDIA CUDA 12.6.1	2.4	24.09
24.07		NVIDIA CUDA 12.5.1	2.3	24.07

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.05		NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLEError` will occur.

Chapter 12. DGL Release 24.10

There is no DGL container in DLFW release 24.10.

Chapter 13. DGL Release 24.09

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.4.
- ▶ RAPIDS 24.08
- ▶ This container also contains WholeGraph 24.08 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.6.1](#)
- ▶ [NVIDIA cuBLAS 12.6.3.1](#)
- ▶ [NVIDIA cuDNN 9.4.0.58](#)
- ▶ [NVIDIA NCCL 2.22.3](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.20
- ▶ [OpenMPI 4.1.7](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2024.2.1.2](#)
- ▶ [Nsight Systems 2024.2.1.133](#)
- ▶ [NVIDIA TensorRT™ 10.2.0.19](#)
- ▶ [Torch-TensorRT 2.5.0.a0](#)
- ▶ [NVIDIA DALI® 1.41](#)

- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.10
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 24.09 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ Starting with the 24.07 release, the DGL container supports distributed in-memory sampling and feature gathering with WholeGraph. It can be easily integrated with DGL GraphBolt dataloader, enabling out-of-the-box distributed GNN training.
- ▶ DGL GraphBolt does not depend on the deprecated torchdata package anymore.
- ▶ DGL GraphBolt changes the CUDA memory allocation configuration to reduce memory footprint.

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.09	22.04	NVIDIA CUDA 12.6.1	2.4	24.09
24.07		NVIDIA CUDA 12.5.1	2.3	24.07
24.05		NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLError` will occur.

Chapter 14. DGL Release 24.08

This container is only available to NVAIE customers.

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Driver Requirements

Release 24.08 is based on [CUDA 12.6](#) which requires [NVIDIA Driver](#) release 560 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 drivers, which are not forward-compatible with CUDA 12.6. For a complete list of supported drivers, see the [CUDA Application Compatibility](#) topic. For more information, see [CUDA Compatibility and Upgrades](#).

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.3.
- ▶ RAPIDS 24.06
- ▶ This container also contains WholeGraph 24.06 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.6](#)
- ▶ [NVIDIA cuBLAS 12.6.0.22](#)
- ▶ [NVIDIA cuDNN 9.3.0.75](#)
- ▶ [NVIDIA NCCL 2.22.3](#)

- ▶ NVIDIA RAPIDS™ 24.06
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.19
- ▶ [OpenMPI 4.1.7](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.16.2](#)
- ▶ [Nsight Compute 2024.3.0.15](#)
- ▶ [Nsight Systems 2024.4.2.133](#)
- ▶ [NVIDIA TensorRT™ 10.3.0.26](#)
- ▶ [Torch-TensorRT 2.5.0.a0](#)
- ▶ [NVIDIA DALI® 1.40](#)
- ▶ [nvlImageCodec 0.2.0.7](#)
- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.4.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.9
- ▶ [TensorRT Model Optimizer 0.15.0](#)

GPU Requirements

Release 24.08 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ The DGL container supports distributed in-memory sampling and feature gathering with WholeGraph. It can be easily integrated with DGL GraphBolt dataloader, enabling out-of-the-box distributed GNN training.

Known Issues

- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.data.Dataloader` class. When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLError` will occur.

Chapter 15. DGL Release 24.07

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.3.
- ▶ RAPIDS 24.04
- ▶ This container also contains WholeGraph 24.06 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.5.1](#)
- ▶ [NVIDIA cuBLAS 12.5.3.2](#)
- ▶ [NVIDIA cuDNN 9.2.1.18](#)
- ▶ [NVIDIA NCCL 2.22.3](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.19
- ▶ [OpenMPI 4.1.7](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2024.2.1.2](#)
- ▶ [Nsight Systems 2024.2.1.133](#)
- ▶ [NVIDIA TensorRT™ 10.2.0.19](#)
- ▶ [Torch-TensorRT 2.4.0.a0](#)
- ▶ [NVIDIA DALI® 1.39](#)

- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.8
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 24.07 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ Starting with the 24.07 release, the DGL container supports distributed in-memory sampling and feature gathering with WholeGraph. It can be easily integrated with DGL GraphBolt dataloader, enabling out-of-the-box distributed GNN training.

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.07	22.04	NVIDIA CUDA 12.5.1	2.3	24.07
24.05		NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLError` will occur.

Chapter 16. DGL Release 24.06

There is no DGL container in DLFW release 24.06.

Chapter 17. DGL Release 24.05

This DGL container release is intended for use on the NVIDIA® Hopper Architecture GPU, NVIDIA H100, the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries.

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.2 (including DGL-Graphbolt, a recently released GNN dataloader library which has achieved state-of-the-art performance on NVIDIA GPUs).
- ▶ RAPIDS 24.04
- ▶ This container also contains WholeGraph 24.04 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.4.1](#)
- ▶ [NVIDIA cuBLAS 12.4.5.8](#)
- ▶ [NVIDIA cuDNN 9.1.0.70](#)
- ▶ [NVIDIA NCCL 2.21.5](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.19
- ▶ [OpenMPI 4.1.4+](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2024.1.1.4](#)
- ▶ [Nsight Systems 2024.2.1.106](#)
- ▶ [NVIDIA TensorRT™ 10.0.1.6](#)
- ▶ [Torch-TensorRT 2.4.0.a0](#)

- ▶ [NVIDIA DALI[®] 1.37.1](#)
- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.6
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 24.05 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ In this release of the NVIDIA DGL container, NVIDIA enhances support for distributed feature gathering by integrating NVSHMEM, further improving on the feature fetching performance for distributed GNN tasks. Check out the examples located at: `/workspace/examples/wholegraph-examples`
- ▶ Add NVIDIA [Synthetic Graph Generation tool](#) for generating graphs with an arbitrary size, including node and edge tabular features.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.05	22.04	NVIDIA CUDA 12.4.1	2.2	24.05
24.04			2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for edata or ndata, a `DGLError` will occur.

Chapter 18. DGL Release 24.04

This DGL container release is intended for use on the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 8 libraries..

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.1+7c51cd16 (including DGL-Graphbolt, a recently released GNN dataloader library which has achieved state-of-the-art performance on NVIDIA GPUs).
- ▶ RAPIDS 24.02
- ▶ This container also contains WholeGraph 24.02 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ [NVIDIA CUDA® 12.4.1](#)
- ▶ [NVIDIA cuBLAS 12.4.5.8](#)
- ▶ [NVIDIA cuDNN 9.1.0.70](#)
- ▶ [NVIDIA NCCL 2.21.5](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.18
- ▶ [OpenMPI 4.1.4+](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2024.1.0.13](#)
- ▶ [Nsight Systems 2024.2.1.38](#)
- ▶ [NVIDIA TensorRT™ 8.6.3](#)
- ▶ [Torch-TensorRT 2.30.a0](#)
- ▶ [NVIDIA DALI® 1.36](#)

- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine v1.5
- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 24.04 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ In this release of the NVIDIA DGL container, NVIDIA enhances support for distributed feature gathering by integrating NVSHMEM, further improving on the feature fetching performance for distributed GNN tasks. Check out the examples located at: `/workspace/examples/wholegraph-examples`
- ▶ Add NVIDIA [Synthetic Graph Generation tool](#) for generating graphs with an arbitrary size, including node and edge tabular features.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.04	22.04	NVIDIA CUDA 12.4.1	2.1+e1f7738	24.04
24.03		NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
23.11		NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for edata or ndata, a `DGLError` will occur.

Chapter 19. DGL Release 24.03

The NVIDIA container image for DGL, release 24.03, is available on [NGC](#).

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ DGL 2.1+7c51cd16 (including DGL-Graphbolt, a recently released GNN dataloader library which has achieved state-of-the-art performance on NVIDIA GPUs)
- ▶ RAPIDS 24.02
- ▶ This container also contains WholeGraph 24.02 with NVSHMEM support. WholeGraph is a part of the NVIDIA RAPIDS library which provides an underlying graph storage structure to enhance GNN training, especially optimized for NVIDIA hardware.
- ▶ Built on PyTorch 24.03 (see contents of PyTorch container).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ [DGL](#) container image version 24.03 is based on DGL 2.1+7c51cd16
- ▶ In this release of the NVIDIA DGL container, we extend to NVSHMEM for distributed feature scatter. See examples located at: `/workspace/examples/wholegraph-examples`.
- ▶ Add NVIDIA [Synthetic Graph Generation tool](#) for generating graphs with an arbitrary size, including node and edge tabular features.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.03	22.04	NVIDIA CUDA 12.4.0.41	2.1+7c51cd16	24.03
24.01		NVIDIA CUDA 12.3.2	1.2 +c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for edata or ndata, a `DGLError` will occur.

Chapter 20. DGL Release 24.02

There is no DGL container in DLFW release 24.02.

Chapter 21. DGL Release 24.01

The NVIDIA container image for DGL, release 24.01, is available on [NGC](#).

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ [DGL 1.2+c660f5c](#)
- ▶ PyTorch 24.01
- ▶ RAPIDS 23.12
- ▶ [NVIDIA CUDA® 12.3.2](#)
- ▶ [NVIDIA cuBLAS 12.3.4.1](#)
- ▶ [NVIDIA cuDNN 8.9.7.29](#)
- ▶ [NVIDIA NCCL 2.19.4](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.16rc4
- ▶ [OpenMPI 4.1.4+](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2023.3.1.1](#)
- ▶ [Nsight Systems 2023.4.1.97](#)
- ▶ [NVIDIA TensorRT™ 8.6.1.6](#)
- ▶ [Torch-TensorRT 1.4.0](#)
- ▶ [NVIDIA DALI® 1.33.0](#)
- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ [TransformerEngine v1.2.1](#)

- ▶ [NVSHMEM 2.10.1](#)

GPU Requirements

Release 24.01 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- ▶ [DGL](#) container image version 24.01 is based on DGL 1.1.1.
- ▶ In this release of the NVIDIA DGL container, we enhance support for distributed feature gathering by integrating NVSHMEM, further improving on the feature fetching performance for distributed GNN tasks. Check out the examples located at: `/workspace/examples/wholegraph-examples`
- ▶ Add NVIDIA [Synthetic Graph Generation tool](#) for generating graphs with an arbitrary size, including node and edge tabular features.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
24.01	22.04	NVIDIA CUDA 12.3.2	1.2+c660f5c	24.01
23.11		NVIDIA CUDA 12.3.0	1.1.2	
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.data.loading.DataLoader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLEError` will occur.

Chapter 22. DGL Release 23.12

There is no DGL container in DLFW release 23.12.

Chapter 23. DGL Release 23.11

The NVIDIA container image for DGL, release 23.11, is available on [NGC](#).

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ [DGL 1.2+4b6faec](#)
- ▶ PyTorch 23.11
- ▶ RAPIDS 23.10
- ▶ [NVIDIA CUDA® 12.3.0](#)
- ▶ [NVIDIA cuBLAS 12.2.5.6](#)
- ▶ [NVIDIA cuDNN 8.9.6](#)
- ▶ [NVIDIA NCCL 2.19.3](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ NVIDIA HPC-X 2.16
- ▶ [OpenMPI 4.1.4+](#)
- ▶ GDRCopy 2.3
- ▶ [TensorBoard 2.12.0](#)
- ▶ [Nsight Compute 2023.3.0.12](#)
- ▶ [Nsight Systems 2023.3.1.92](#)
- ▶ [NVIDIA TensorRT™ 8.6.1.6](#)
- ▶ [Torch-TensorRT 1.4.0](#)
- ▶ [NVIDIA DALI® 1.31.0](#)
- ▶ [MAGMA 2.6.2](#)
- ▶ JupyterLab 2.3.2 including [Jupyter-TensorBoard](#)
- ▶ PyTorch quantization wheel v2.1.2
- ▶ TransformerEngine 0.3.0

- [NVSHMEM 2.10.1](#)

GPU Requirements

Release 23.11 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- [DGL](#) container image version 23.11 is based on DGL 1.1.1.
- In this release of the NVIDIA DGL container, we enhance support for distributed feature gathering by integrating NVSHMEM, further improving on the feature fetching performance for distributed GNN tasks. Check out the examples located at: `/workspace/examples/wholegraph-examples`
- Add NVIDIA [Synthetic Graph Generation tool](#) for generating graphs with an arbitrary size, including node and edge tabular features.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
23.11	22.04	NVIDIA CUDA 12.3.0	1.1.2	23.11
23.09		NVIDIA CUDA 12.2.1	1.1.2	23.09
23.07		NVIDIA CUDA 12.1.1	1.1.1	23.07

Known Issues

- ▶ When cpu sampling is enabled (`use_uva=False` and `num_workers>0`), DGL sampling process would initialize cuda instance ([issue-6561](#)), which could result in a segmentation fault with the current cuda driver in the container.
- ▶ The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.data.loading.DataLoader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLEError` will occur.

Chapter 24. DGL Release 23.09

The NVIDIA container image for DGL, release 23.09, is available on [NGC](#).

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ [DGL 1.1.1](#)
- ▶ PyTorch 23.09
- ▶ RAPIDS 23.08
- ▶ [Ubuntu 20.04](#) including [Python 3.8](#)
- ▶ [NVIDIA CUDA® 12.2.1](#)
- ▶ [NVIDIA cuBLAS 12.2.5.6](#)
- ▶ [NVIDIA cuDNN 8.9.5](#)
- ▶ [NVIDIA NCCL 2.18.5](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ [OpenMPI 4.1.4+](#)
- ▶ GDRCopy 2.3
- ▶ [Nsight Compute 2023.2.1.3](#)
- ▶ [Nsight Systems 2023.3.1.92](#)
- ▶ NVIDIA HPC-X 2.16
- ▶ [TensorRT 8.6.1.6](#)
- ▶ SHARP 2.5
- ▶ TensorBoard 2.7.0
- ▶ [DALI 1.29.0](#)

GPU Requirements

Release 23.09 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- [DGL](#) container image version 23.09 is based on DGL 1.1.1.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
23.09	22.04	NVIDIA CUDA 12.2.1	1.1.1	23.09
23.07		NVIDIA CUDA 12.1.1	0.9.x	23.07

Known Issues

- The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.Dataloader` class.

When you attempt to use a graph with a non-contiguous or view tensors for `edata` or `ndata`, a `DGLError` will occur.

Chapter 25. DGL Release 23.07

The NVIDIA container image for DGL, release 23.07, is available on [NGC](#).

Contents of the DGL container

This container image contains the complete source of the version of DGL in `/opt/dgl/dgl-source`. It is pre-built and installed as a system Python module.

The container includes the following:

- ▶ [DGL 1.1+f635e2a](#)
- ▶ PyTorch 23.07
- ▶ RAPIDS 23.06
- ▶ [Ubuntu 20.04](#) including [Python 3.8](#)
- ▶ [NVIDIA CUDA® 12.1.1](#)
- ▶ [NVIDIA cuBLAS 12.1.3.1](#)
- ▶ [NVIDIA cuDNN 8.9.3](#)
- ▶ [NVIDIA NCCL 2.18.3](#)
- ▶ [Apex](#)
- ▶ [rdma-core 39.0](#)
- ▶ [OpenMPI 4.1.4+](#)
- ▶ GDRCopy 2.3
- ▶ [Nsight Compute 2023.1.1.4](#)
- ▶ [Nsight Systems 2023.2.3.1001](#)
- ▶ NVIDIA HPC-X 2.15
- ▶ [TensorRT 8.6.1.6](#)
- ▶ SHARP 2.5
- ▶ TensorBoard 2.7.0
- ▶ [DALI 1.27.0](#)

GPU Requirements

Release 23.07 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see [CUDA GPUs](#). For additional support details, see [Deep Learning Frameworks Support Matrix](#).

Key Features and Enhancements

This DGL release includes the following key features and enhancements.

- [DGL](#) container image version 23.07 is based on DGL 1.1.1.

The major features of the release can be found in the DGL [release notes](#).

Announcements

None.

NVIDIA DGL Container Versions

The following table shows what versions of Ubuntu, CUDA, DGL, and TensorRT are supported in each NVIDIA containers for DGL. For older container versions, refer to the [Frameworks Support Matrix](#).

Container Version	Ubuntu	CUDA Toolkit	DGL	PyTorch
23.07	22.04	NVIDIA CUDA 12.1.1	0.9.x	23.07

Known Issues

- The tensors that are used as node features must be contiguous and cannot be views of other tensors when the `use_uva` flag is set to `True` in the `dgl.dataloading.DataLoader` class.

When you attempt to use a graph with a non-contiguous or view tensors for edata or ndata, a `DGLError` will occur.

Notice

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

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

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

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

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

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

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

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

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

HDMI

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

OpenCL

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

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, DALI, DGX, DGX-1, DGX-2, DGX Station, DLProf, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NvCaffe, PerfWorks, Pascal, SDK Manager, Tegra, TensorRT, Triton Inference Server, Tesla, TF-TRT, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2021-2025 NVIDIA Corporation & Affiliates. All rights reserved.

