DEEP LEARNING

Tutorial: Environment Setup

Mingbo Li

Department of Computer Science and Technology

limingbo@stu.xmu.edu.cn

NVIDIA GPU SOFTWARE STACK

Architecture: Virtual Environment

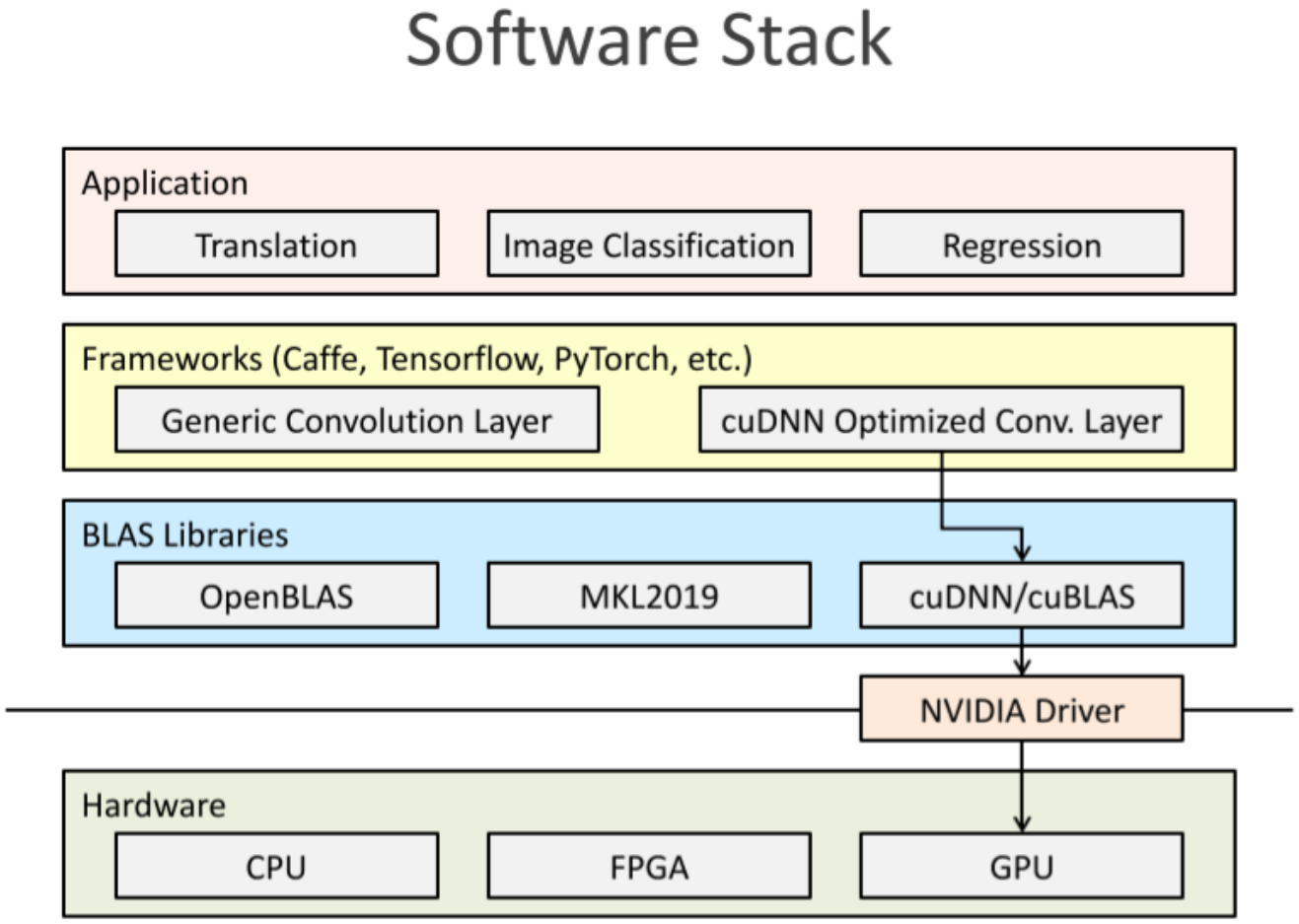
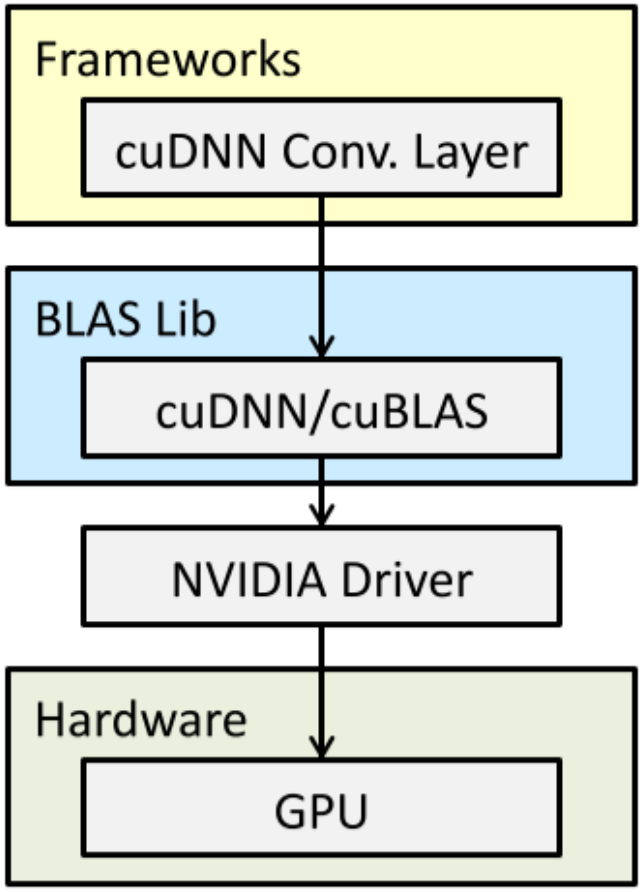


Image source: Hung-yi Lee, Environment Setup

Architecture: Virtual Environment



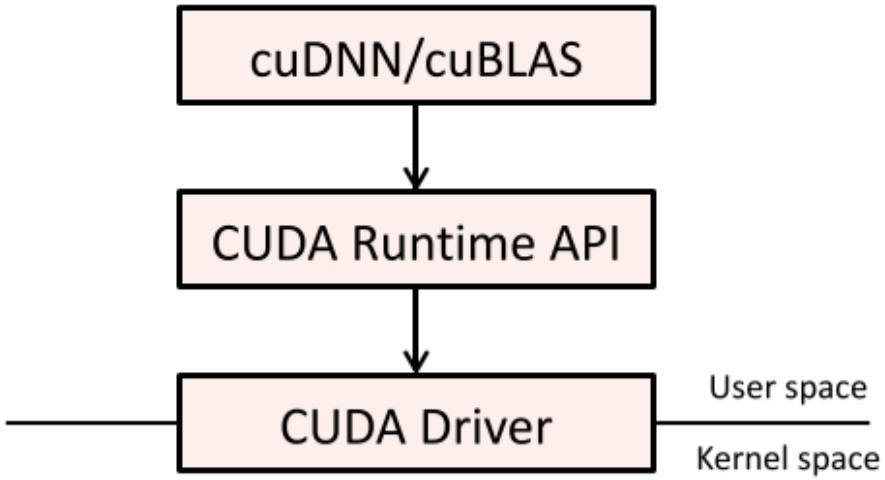


Image source: Hung-yi Lee, Environment Setup

Architecture: Docker Environment

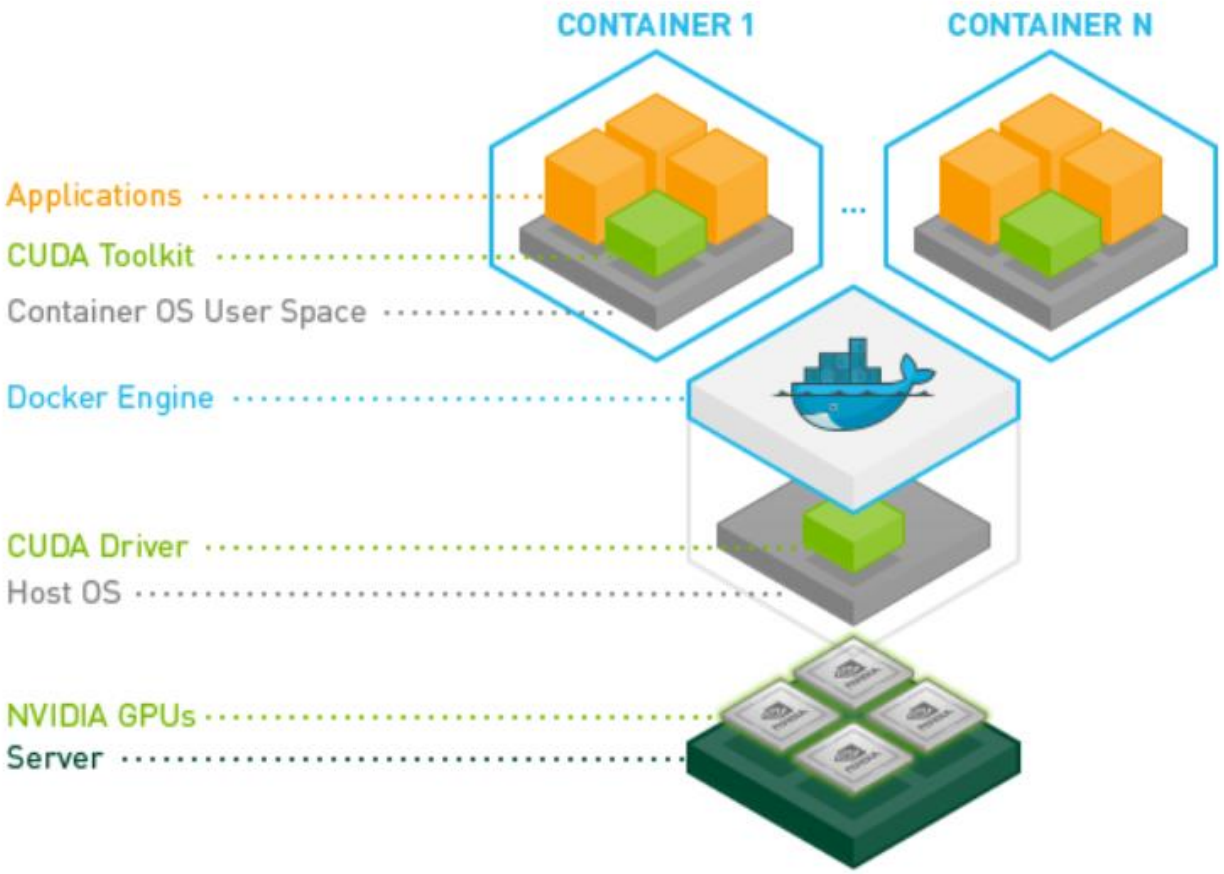


Image source: NVIDIA Docker: GPU Server Application Deployment Made Easy | NVIDIA Technical Blog

Architecture: Docker Environment

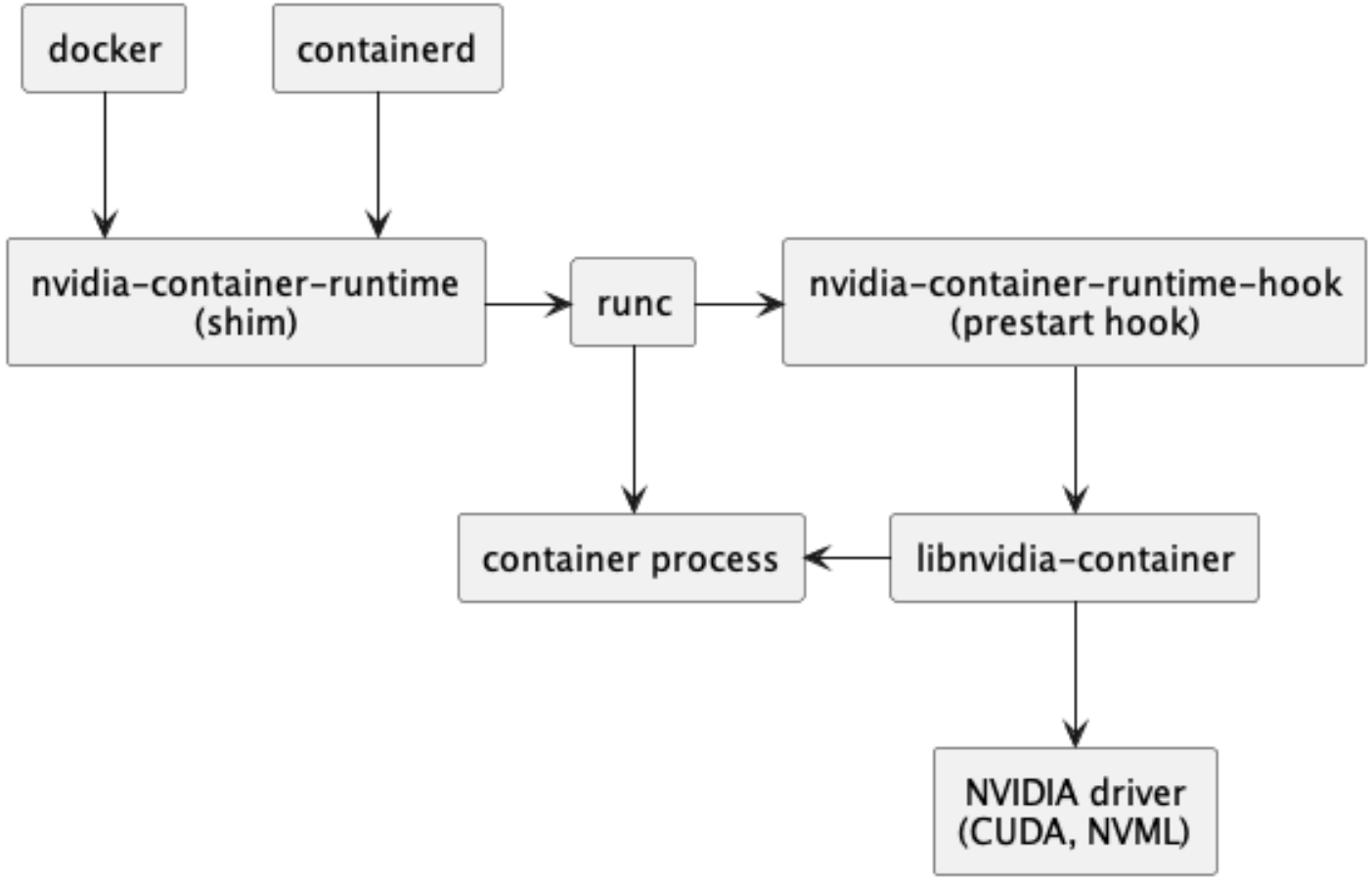


Image source: https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/_images/runtime-architecture.png

NVIDIA GPU Driver and its CUDA version

- Review OS:
  - What is a device driver?
  - A device driver is a special kind of software program that controls a specific hardware device attached to a computer.
  - The NVIDIA driver that allows operating systems (OS) to communicate with GPUs
- Install NVIDIA Driver in OS
  - Official Website of variable version CUDA and Driver: cuda-toolkit-archive
  - Here, the CUDA version is the driver CUDA version but not the runtime CUDA version.
  - The TensorFlow and Pytorch rely on CUDA runtime
  - The driver CUDA version must ≥ the runtime CUDA version

Reference: Hung-yi Lee, EnvironmentSetup; https://www.techtarget.com/searchenterprisedesktop/definition/device-driver

driver installation suggestion from NVIDIA

- The recommended way is to use your package manager and install the cuda-drivers package (or equivalent).
- When no packages are available, you should use an official "runfile".
- It’s important to Disabling Nouveau when you install driver on linux

Reference: how-do-i-install-the-nvidia-driver, cuda-installation-guide-linux

Installation of GPU Driver(Package Managers)

- For example: Ubuntu LTS

3.1. Ubuntu LTS

This section includes instructions for installing the NVIDIA driver on Ubuntu 16.04 LTS and Ubuntu 18.04 LTS distributions using the package manager.

1. The NVIDIA driver requires that the kernel headers and development packages for the running version of the kernel be installed at the time of the driver installation, as well whenever the driver is rebuilt. For example, if your system is running kernel version 4.4.0, the 4.4.0 kernel headers and development packages must also be installed.

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

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

2. Install the CUDA repository public GPG key. This can be done via the `cuda-keyring` package or a manual installation of the key. The usage of `apt-key` is deprecated.

   ```
   $ distribution=$(. /etc/os-release;echo $ID$VERSION_ID | sed -e 's/\.//g')
   ```

   ```
   $ wget https://developer.download.nvidia.com/compute/cuda/repos/$distribution/x86_64/cuda-keyring_1.0-1_all.deb
   ```

   ```
   $ sudo dpkg -i cuda-keyring_1.0-1_all.deb
   ```

3. Update the APT repository cache and install the driver using the cuda-drivers meta-package. Use the `--no-install-recommends` option for a lean driver install without any dependencies on X packages. This is particularly useful for headless installations on cloud instances.

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

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

4. Follow the post-installation steps in the CUDA Installation Guide for Linux to setup environment variables, NVIDIA persistence daemon (recommended) and to verify the successful installation of the driver.

Reference: NVIDIA Driver Installation Quickstart Guide

Installation of GPU Driver

- The driver CUDA version: 12.0
- Visit: https://developer.nvidia.com/cuda-toolkit-archive

Previous releases of the CUDA Toolkit, GPU Computing SDK, documentation and developer drivers can be found using the links below. Please select the release you want from the list below, and be sure to check www.nvidia.com/drivers for more recent production drivers appropriate for your hardware configuration.

Download Latest CUDA Toolkit Learn More about CUDA Toolkit 11

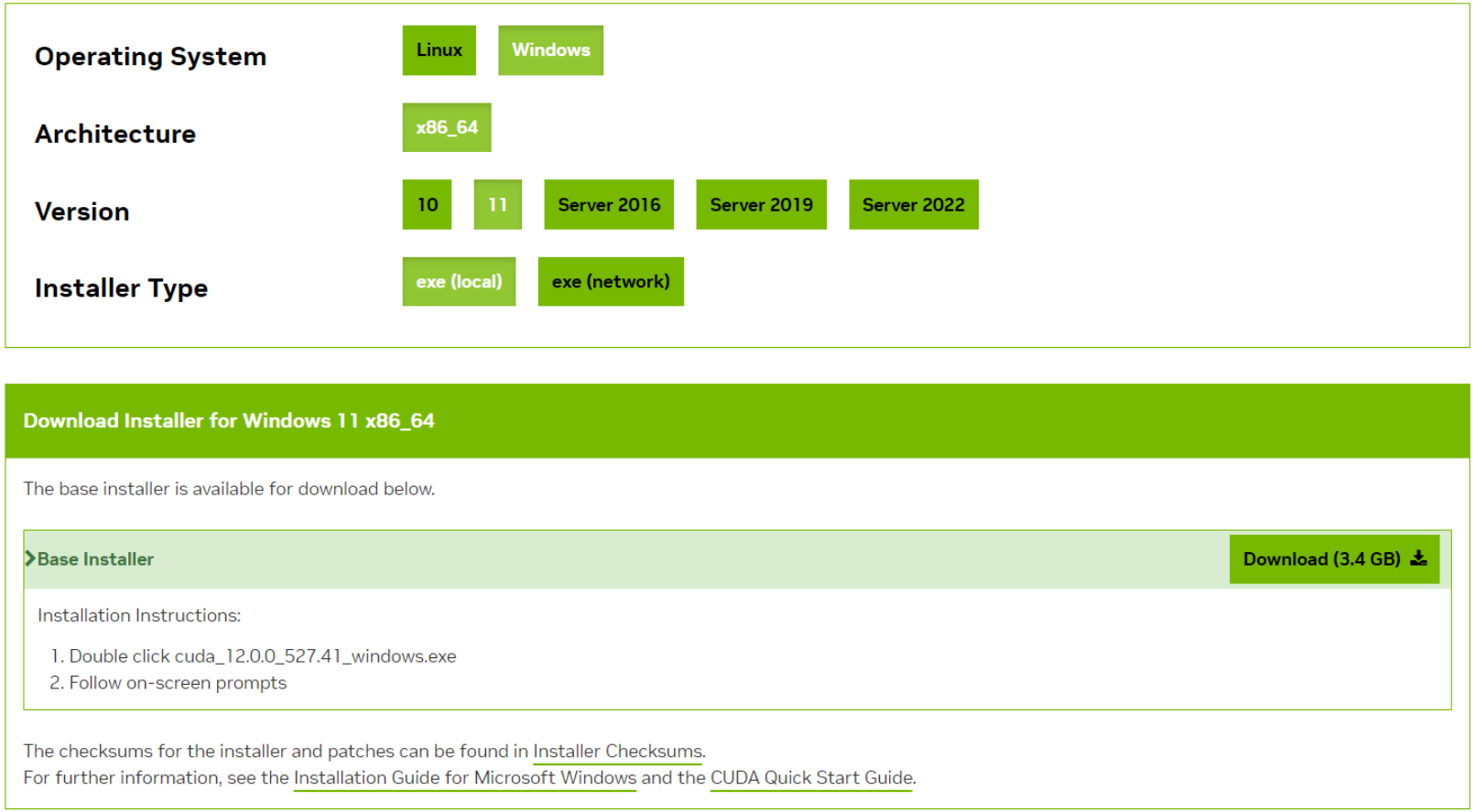
Latest Release

CUDA Toolkit 12.1.1 (April 2023), Versioned Online Documentation

Archived Releases

CUDA Toolkit 12.1.0 (February 2023), Versioned Online Documentation
CUDA Toolkit 12.0.1 (January 2023), Versioned Online Documentation
CUDA Toolkit 12.0.0 (December 2022), Versioned Online Documentation
CUDA Toolkit 11.8.0 (October 2022), Versioned Online Documentation
CUDA Toolkit 11.7.1 (August 2022), Versioned Online Documentation
CUDA Toolkit 11.7.0 (May 2022), Versioned Online Documentation
CUDA Toolkit 11.6.2 (March 2022), Versioned Online Documentation
CUDA Toolkit 11.6.1 (February 2022), Versioned Online Documentation
CUDA Toolkit 11.6.0 (January 2022), Versioned Online Documentation
CUDA Toolkit 11.5.2 (February 2022), Versioned Online Documentation
CUDA Toolkit 11.5.1 (November 2021), Versioned Online Documentation
CUDA Toolkit 11.5.0 (October 2021), Versioned Online Documentation
CUDA Toolkit 11.4.4 (February 2022), Versioned Online Documentation
CUDA Toolkit 11.4.3 (November 2021), Versioned Online Documentation
CUDA Toolkit 11.4.2 (September 2021), Versioned Online Documentation
CUDA Toolkit 11.4.1 (August 2021), Versioned Online Documentation
CUDA Toolkit 11.4.0 (June 2021), Versioned Online Documentation
CUDA Toolkit 11.3.1 (May 2021), Versioned Online Documentation

Installation of GPU Driver on Windows



Installation of GPU Driver on Ubuntu(runfile)

- Install build tool:

  sudo apt-get update && sudo apt-get install -y build-essential git libgfortran3

- or

  sudo apt-get update && sudo apt-get install g++ gcc make

```
(base) limingbo@luoyegroup-ubuntu:~$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

(base) limingbo@luoyegroup-ubuntu:~$ g++ --version
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

(base) limingbo@luoyegroup-ubuntu:~$ make --version
GNU Make 4.2.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
(base) limingbo@luoyegroup-ubuntu:~$
```

Installation of GPU Driver on Ubuntu(runfile)

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the CUDA EULA.

Operating System	Linux, Windows
Architecture	x86_64, ppc64le, arm64-sbsa, aarch64-jetson
Distribution	CentOS, Debian, Fedora, KylinOS, OpenSUSE, RHEL, Rocky, SLES, Ubuntu, WSL-Ubuntu
Version	18.04, 20.04, 22.04
Installer Type	deb (local), deb (network), runfile (local)

Installation of GPU Driver on Ubuntu(run file)

Download Installer for Linux Ubuntu 22.04 x86_64

The base installer is available for download below.

Base Installer

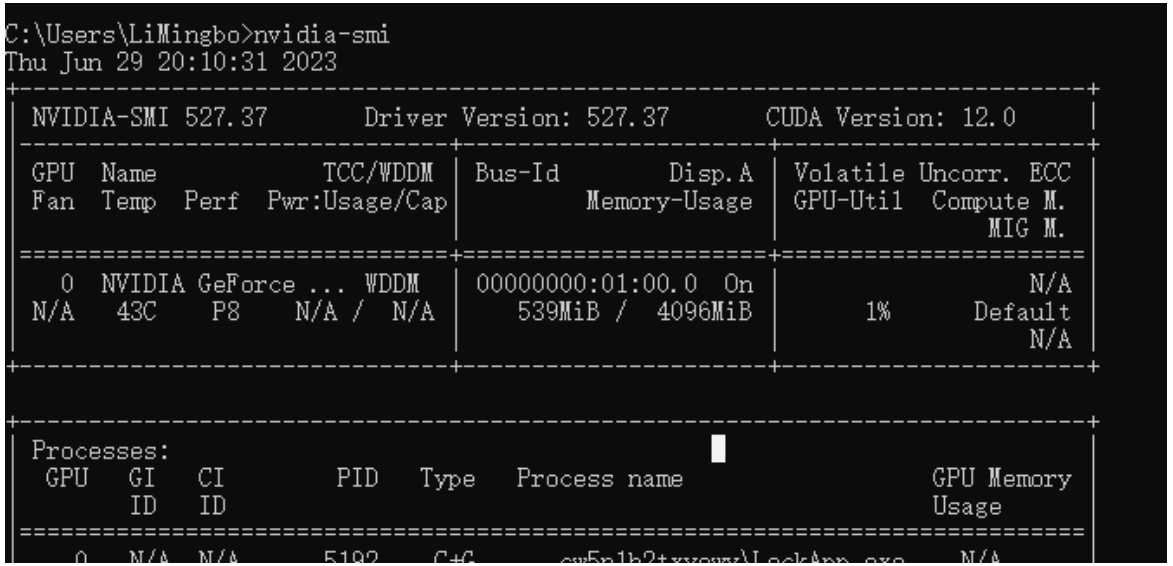
Installation Instructions:

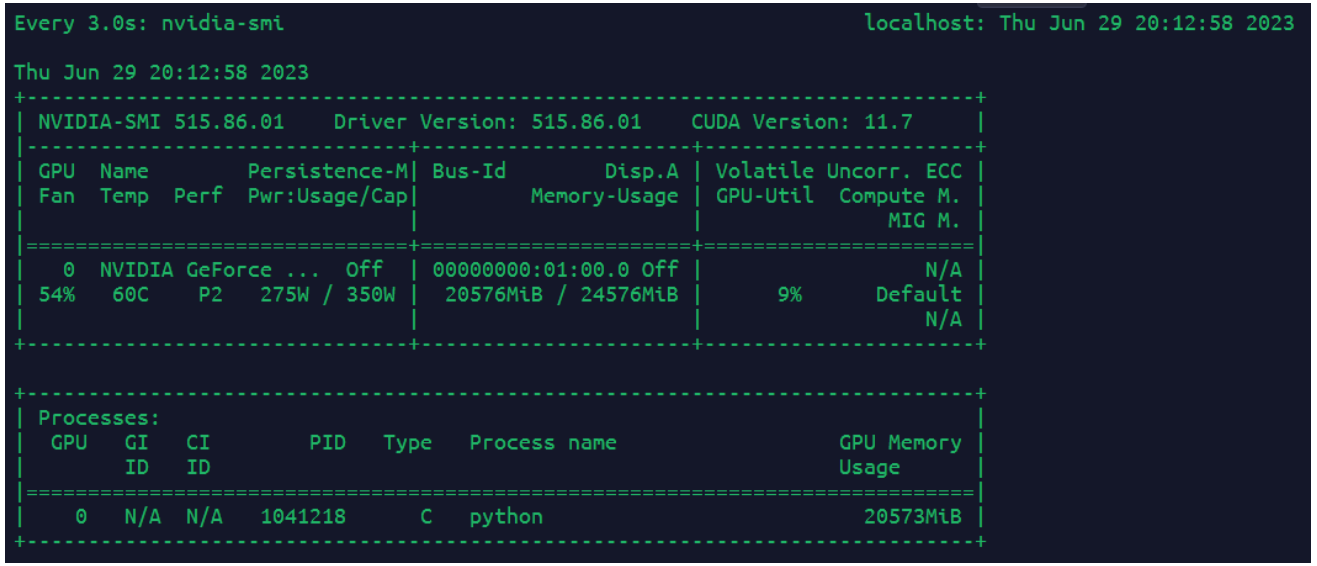
```
$ wget https://developer.download.nvidia.com/compute/cuda/12.0.0/local_installers/cuda_12.0.0_525.60.13_linux.run
$ sudo sh cuda_12.0.0_525.60.13_linux.run
```

The CUDA Toolkit contains Open-Source Software. The source code can be found here.
The checksums for the installer and patches can be found in Installer Checksums.
For further information, see the Installation Guide for Linux and the CUDA Quick Start Guide.

Check NVIDIA Driver

- nvidia-smi
- watch –n 3 nvidia-smi (only in linux)





VERSION REQUIREMENT

Introduction of version dependence

- Usually, There are strict dependence among Python, TensorFlow, PyTorch and CUDA runtime.
- In this Course, the requirement of version as following:

Software	Version
Python	3.9
PyTorch	1.12.0
CUDA runtime in PyTorch	11.3
TensorFlow(Windows)	2.7.0
CUDA runtime in TensorFlow	11.2
cuDNN in TensorFlow	8.1

Help of version issue

- In the future, you may need variable version of these software, where I could find the dependence of version?
  - TensorFlow: https://www.tensorflow.org/install/source
  - PyTorch:
    - https://pytorch.org/get-started/previous-versions/
    - https://pytorch.org/get-started/locally/

PACKAGE MANAGEMENT SOFTWARE

Conda

- Conda is an open source package management system and environment management system that runs on Windows, macOS, and Linux. Conda quickly installs, runs and updates packages and their dependencies.

Reference: https://docs.conda.io/projects/conda/en/stable/

Miniconda and Anaconda

- Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on, and a small number of other useful packages, including pip, zlib and a few others
- Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.
- RTFM: conda-user-guide

Reference: Miniconda, Anaconda (Python distribution) - Wikipedia

Download Mirror

- Due to Network, it's slow to download miniconda and anaconda from official website, Here are some mirror site:
  - Anaconda:
    - https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/
  - Miniconda:
    - https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/
- Also, it also common to use mirror to accelerate the installation of packages.
  - https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/
  - https://mirror.sjtu.edu.cn/docs/anaconda
  - https://developer.aliyun.com/mirror/anaconda/
- More way need to explore yourself.

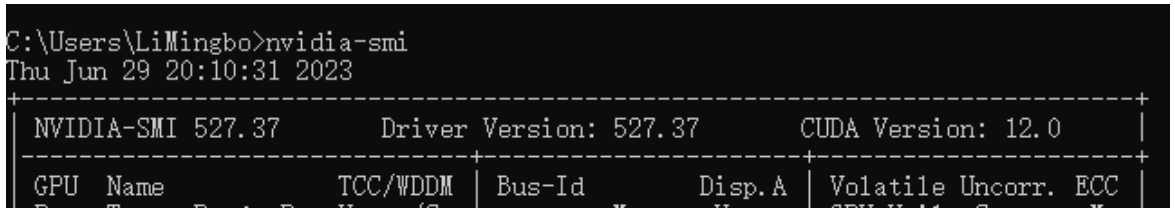
Export and load Anaconda Environments

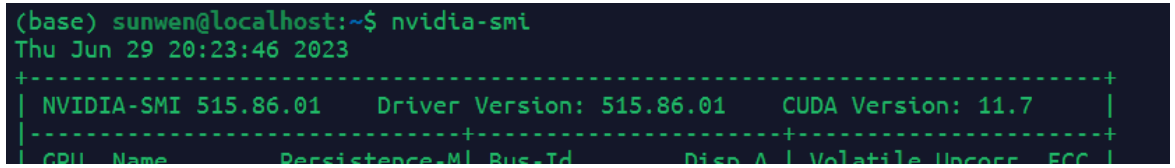
- Just two command to export and load anaconda environment
  - conda env export > environment.yml
  - conda env create -f environment.yml
- The environment config file of our course:
  - TensorFlow：
  - PyTorch：

INSTALLATION OF THE FRAMEWORKS

Attention

- Check your CUDA version of the driver
- The driver CUDA version must ≥ the runtime CUDA version
- Here, for demonstration
  - we use CUDA ==12 at Windows
  - We use CUDA == 11.7 at Ubuntu





PyTorch

- Create an environment: conda create -n torch_gpu python=3.9
- Activate environment: conda activate torch_gpu
- Install PyTorch:
  - conda install pytorch==1.12.0 torchvision==0.13.0 torchaudio==0.12.0 cudatoolkit=11.3 -c pytorch
- Install Jupyter notebook: pip install jupyter
- Start Jupyter notebook: jupyter notebook
- Check your installation:
  - import torch
  - torch.cuda.is_available()

PyTorch

- Windows

```
(torch_gpu) C:\Users\LiMingbo>python
Python 3.9.16 (main, May 17 2023, 17:49:16) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.__version__
'1.12.0'
>>> torch.cuda.is_available()
True
>>>
```

- Linux

```
(torch_gpu) sunwen@localhost:~$ python
Python 3.9.16 (main, May 15 2023, 23:46:34)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.__version__
'1.12.0'
>>> torch.cuda.is_available()
True
>>>
```

TensorFlow in Windows

- Create an environment: conda create –n tf_gpu python=3.9
- Activate environment: conda activate tf_gpu
- Install runtime CUDA: conda install -c conda-forge cudatoolkit=11.2
- Install cudnn: conda install -c conda-forge cudnn=8.1
- Install tensorflow:
  - Windows: pip install tensorflow_gpu==2.7.0
  - mirror acceleration:

pip install tensorflow_gpu==2.7.0 –i https://pypi.tuna.tsinghua.edu.cn/simple/

TensorFlow in Windows

- Install Jupyter notebook: pip install jupyter
- Start Jupyter notebook: jupyter notebook
- Check your installation:
  - import tensorflow as tf
  - tf.test.is_gpu_available()
  - tf.config.list_physical_devices('GPU’)

Some issue about TensorFlow in Windows

- Caution: TensorFlow 2.10 was the last TensorFlow release that supported GPU on native-Windows. Starting with TensorFlow 2.11, you will need to install TensorFlow in WSL2, or install tensorflow or tensorflow-cpu and, optionally, try the TensorFlow-DirectML-Plugin

Reference: https://www.tensorflow.org/install/pip#windows-native

TensorFlow in Windows

- Windows

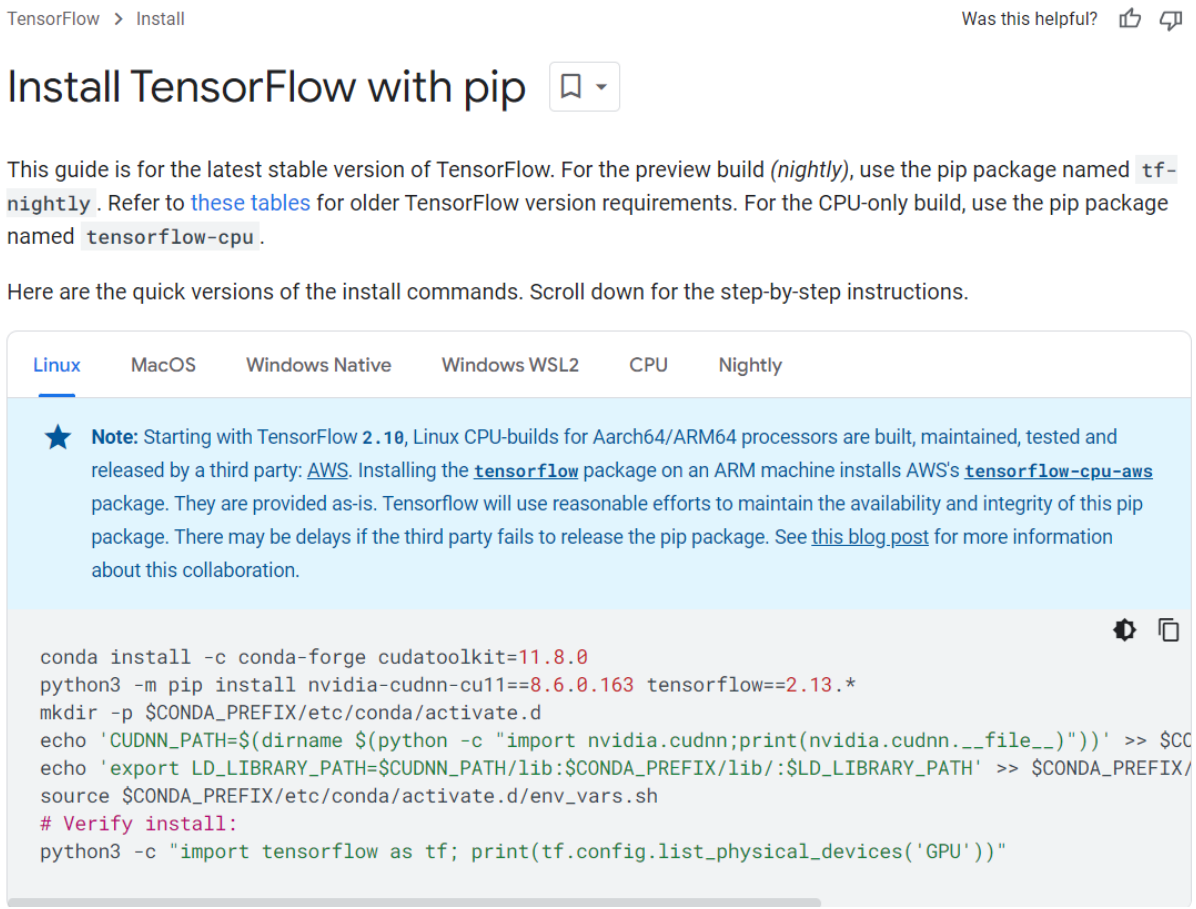
```
(tf_gpu) C:\Users\LiMingbo>python
Python 3.9.16 (main, May 17 2023, 17:49:16) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.7.0'
>>> tf.test.is_gpu_available()
WARNING:tensorflow:From <stdin>:1: is_gpu_available (from tensorflow.python.framework.test_u
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
2023-06-29 21:12:09.312755: I tensorflow/core/platform/cpu_feature_guard.cc:151] This Tensor
ormance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-06-29 21:12:09.854877: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525] Created
ute capability: 6.1
True
>>> tf.config.list_physical_devices('GPU')
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
>>>
```

TensorFlow in Linux

- conda install -c conda-forge cudatoolkit=11.8.0
- python3 -m pip install nvidia-cudnn-cu11==8.6.0.163 tensorflow==2.12.*
- mkdir -p $CONDA_PREFIX/etc/conda/activate.d
- echo 'CUDNN_PATH=$(dirname $(python -c "import nvidia.cudnn;print(nvidia.cudnn.__file__)"))' >> $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
- echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/:$CUDNN_PATH/lib' >> $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
- source $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
- # Verify install:
- python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"

Reference: https://www.tensorflow.org/install/pip#linux

TensorFlow in Linux



TensorFlow > Install

Was this helpful?

Install TensorFlow with pip

This guide is for the latest stable version of TensorFlow. For the preview build *(nightly)*, use the pip package named `tf-nightly`. Refer to these tables for older TensorFlow version requirements. For the CPU-only build, use the pip package named `tensorflow-cpu`.

Here are the quick versions of the install commands. Scroll down for the step-by-step instructions.

Linux | MacOS | Windows Native | Windows WSL2 | CPU | Nightly

Note: Starting with TensorFlow **2.10**, Linux CPU-builds for Aarch64/ARM64 processors are built, maintained, tested and released by a third party: AWS. Installing the **tensorflow** package on an ARM machine installs AWS's **tensorflow-cpu-aws** package. They are provided as-is. Tensorflow will use reasonable efforts to maintain the availability and integrity of this pip package. There may be delays if the third party fails to release the pip package. See this blog post for more information about this collaboration.

```
conda install -c conda-forge cudatoolkit=11.8.0
python3 -m pip install nvidia-cudnn-cu11==8.6.0.163 tensorflow==2.13.*
mkdir -p $CONDA_PREFIX/etc/conda/activate.d
echo 'CUDNN_PATH=$(dirname $(python -c "import nvidia.cudnn;print(nvidia.cudnn.__file__)"))' >> $CO
echo 'export LD_LIBRARY_PATH=$CUDNN_PATH/lib:$CONDA_PREFIX/lib/:$LD_LIBRARY_PATH' >> $CONDA_PREFIX/
source $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
# Verify install:
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```


Reference: https://www.tensorflow.org/install/pip#linux

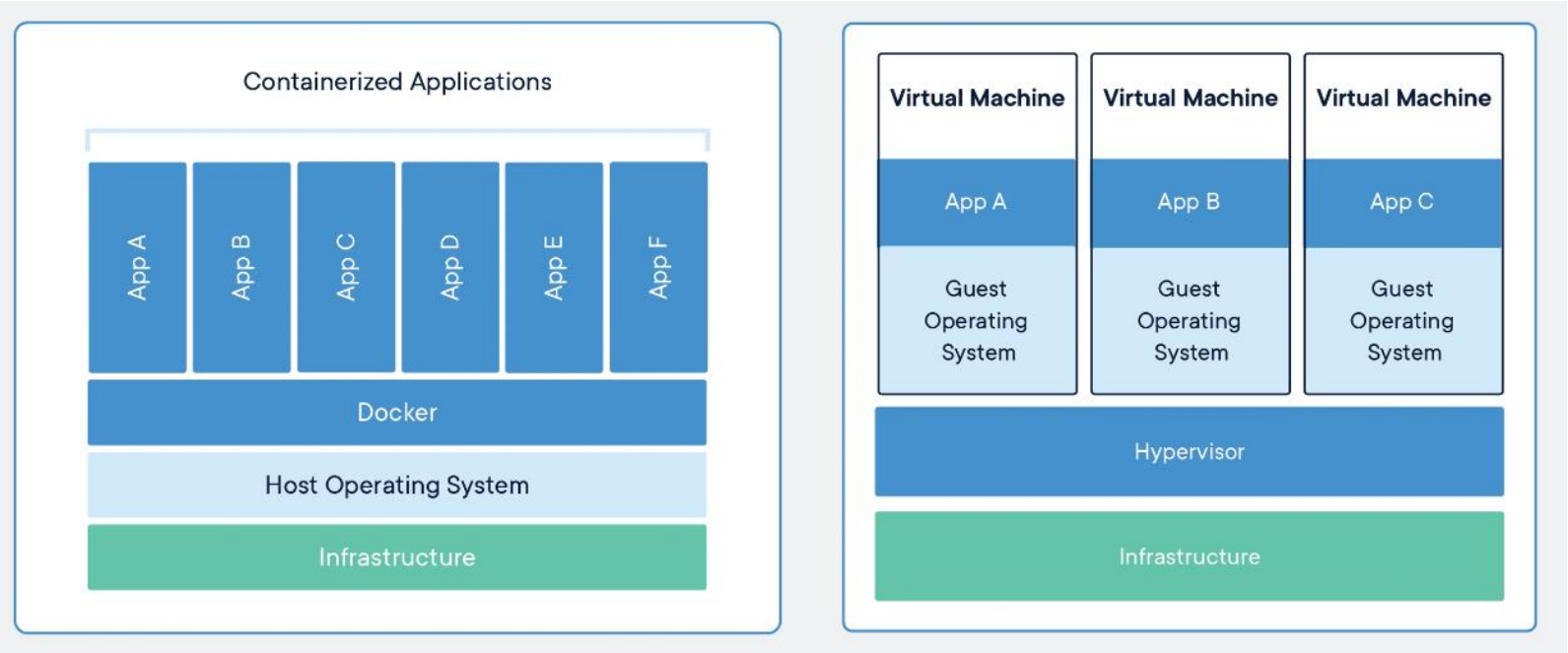
TensorFlow in Linux

- Maybe it is difficult to install TensorFlow in Linux

NVIDIA DOCKER

Docker : Standard, Lightweight and Secure

- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.



Reference: https://www.docker.com/resources/what-container/

Why Docker?

- In 2013, Docker introduced what would become the industry standard for containers. Containers are a standardized unit of software that allows developers to isolate their app from its environment, **solving the "it works on my machine" headache.**
- Developing apps today requires so much more than writing code. Multiple languages, frameworks, architectures, and discontinuous interfaces between tools for each lifecycle stage creates enormous complexity. Docker simplifies and accelerates your workflow, while giving developers the freedom to innovate with their choice of tools, application stacks, and deployment environments for each project.

Reference: https://www.docker.com/why-docker/

Review: NVIDIA Docker

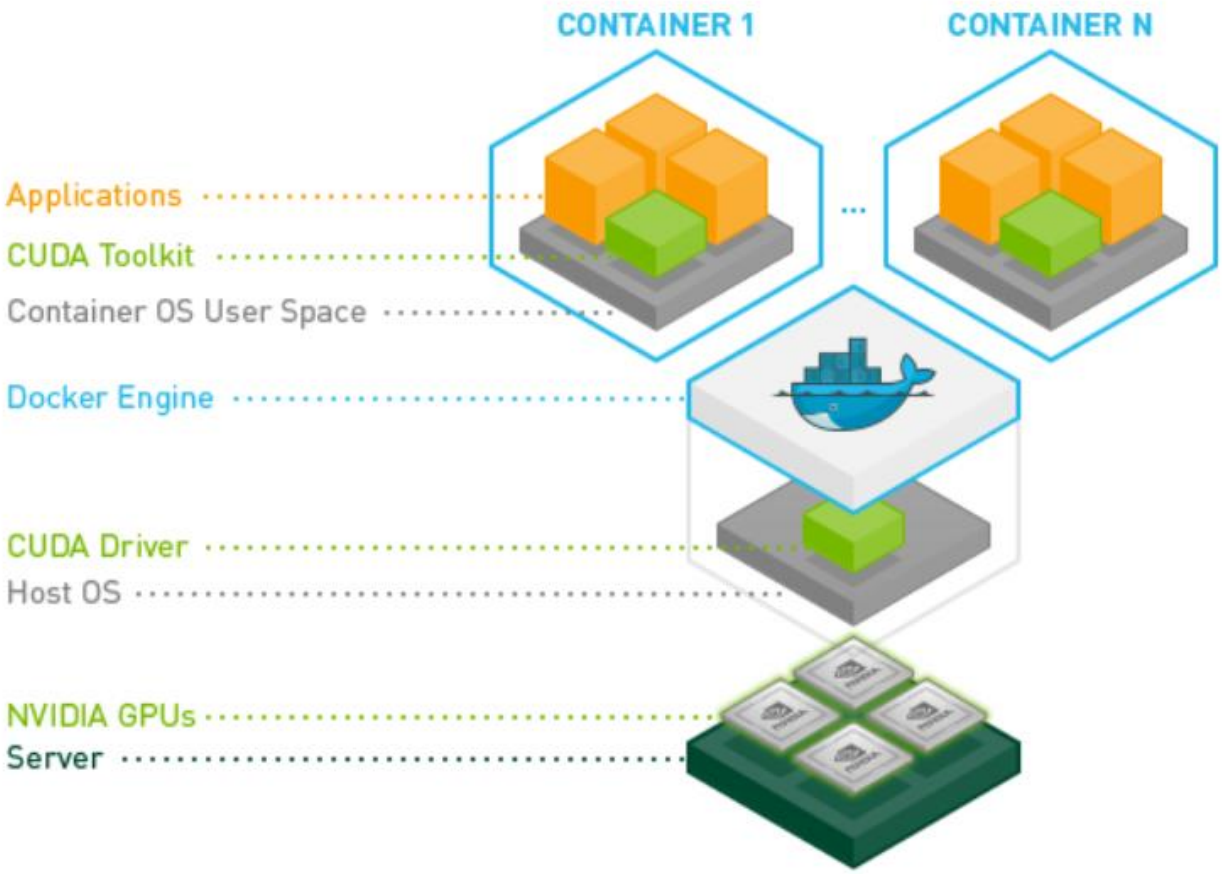


Image source: NVIDIA Docker: GPU Server Application Deployment Made Easy | NVIDIA Technical Blog

Review: NVIDIA Docker

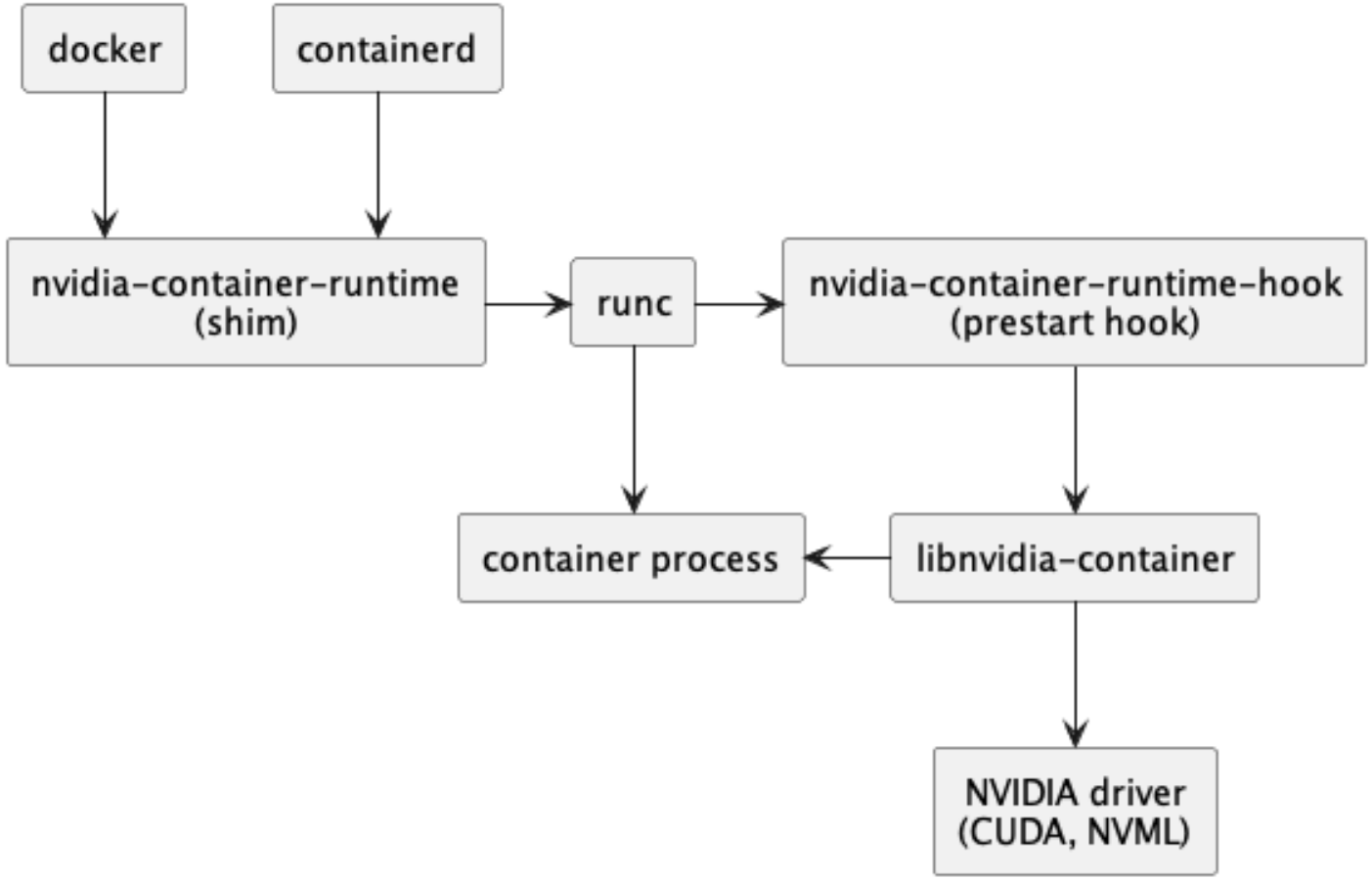


Image source: https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/_images/runtime-architecture.png

How to install NVIDIA Docker

- Install Docker: https://github.com/docker/docker-install
- Check your GPU driver and it's CUDA version
- The driver CUDA version must ≥ the runtime CUDA version
- Install NVIDIA Container Toolkit
  - https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html
  - If your OS is Windows, please use it with WSL2:
    - https://github.com/NVIDIA/nvidia-docker/wiki#is-microsoft-windows-supported
- RTFM: Docker overview | Docker Documentation

Image: CUDA

- nvidia/cuda: https://hub.docker.com/r/nvidia/cuda/

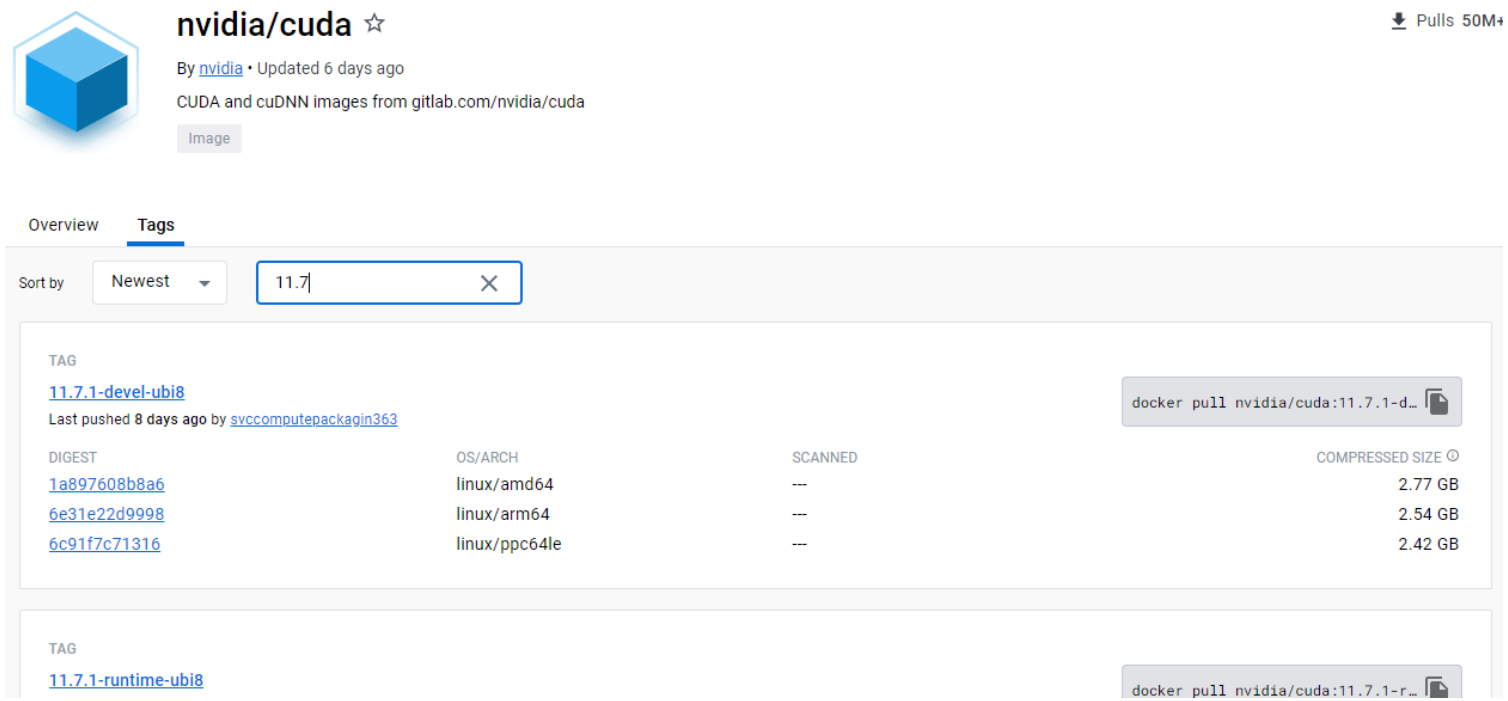


Image: PyTorch

- pytorch/pytorch: https://hub.docker.com/r/pytorch/pytorch/

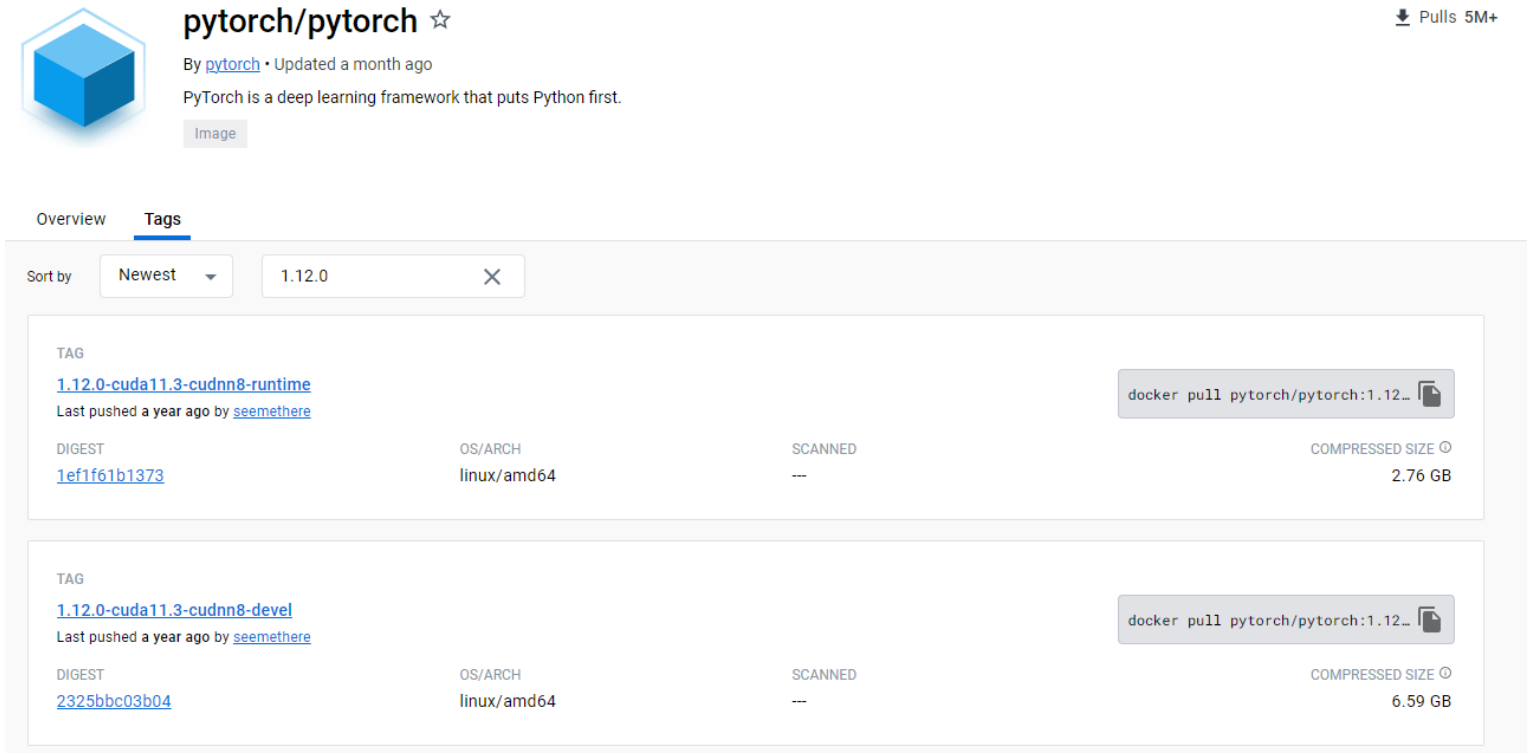
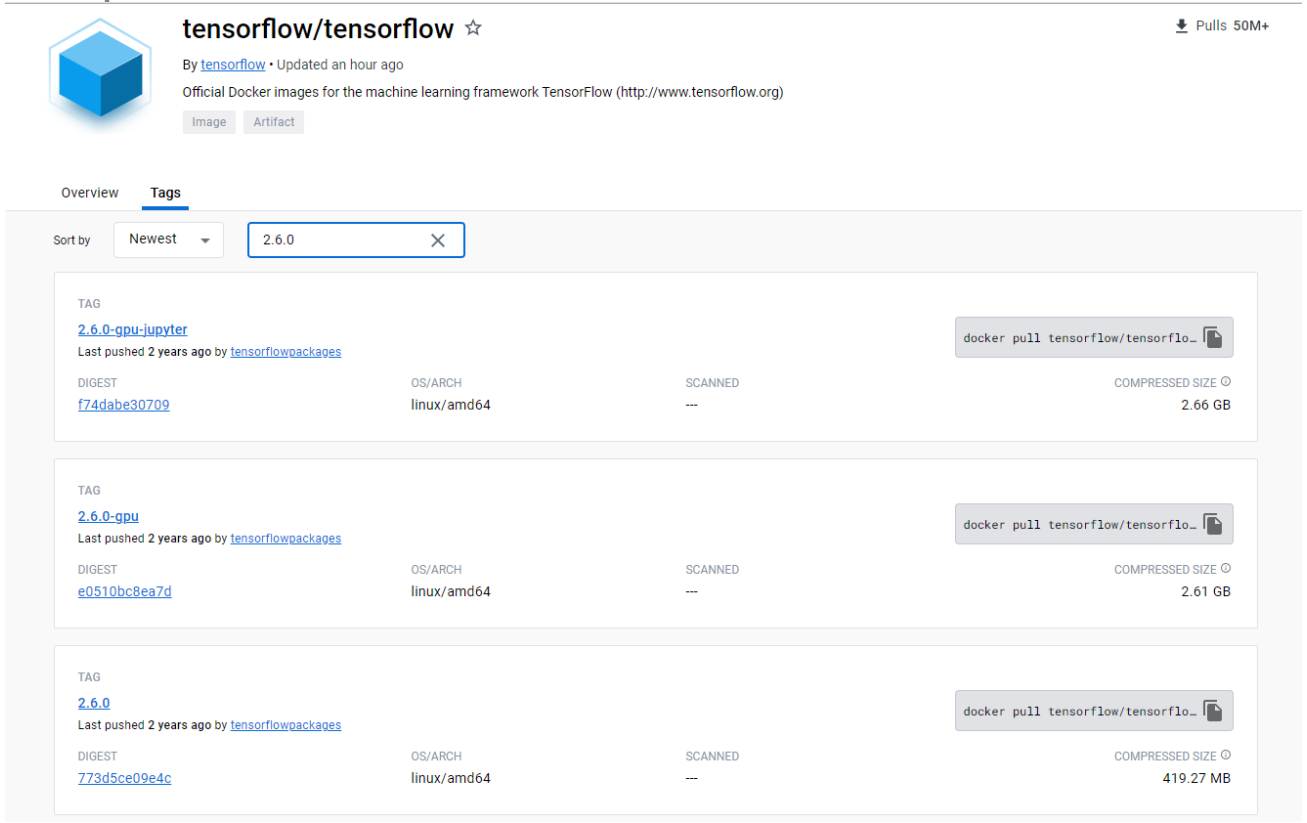


Image: TensorFlow

- tensorflow/tensorflow:
- https://hub.docker.com/r/tensorflow/tensorflow/



Conclusion

After this tutorial, you should know:

- How does the code use GPU to accelerate.
- How to install GPU driver in your computer.
- How to build a PyTorch and TensorFlow development environment based with GPU acceleration.
- How to build a complex PyTorch and TensorFlow development environment based with NVIDIA Docker.

Thank you!

- Any question?
- Don’t hesitate to send email to me for asking questions and discussion. ☺