Machine Learning
[Tutorial: Environment Setup]

I-Ching Tseng
d08922025@csie.ntu.edu.tw
mlta-2022-spring@googlegroups.com
National Taiwan University
March 2022

Outline

- **Overview**
- Package Management Tools
- GPU
- Docker
- Conclusion

Overview

- To run a machine learning (ML) model
  - You have to set up an environment first
  - **Using virtualization or package management tools is a good practice**
    - You can migrate the code and reproduce the result easily
    - Different applications will not affect each other
    - If your environment is broken, just create a new environment

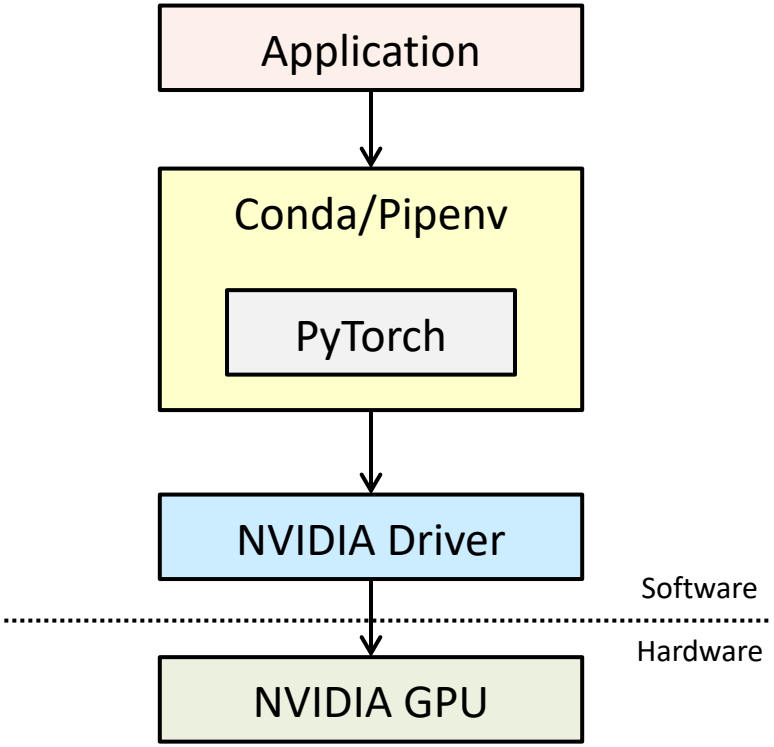
- In this tutorial
  - We will provide some guidelines for setting up environment
  - We will help you understand the environment
    - The software stack
    - NVIDIA GPUs

Outline

- Overview
- **Package Management Tools**
  - Prerequisites
  - Conda
  - Pipenv
  - Summary
- GPU
- Docker
- Conclusion

Prerequisites

- Package management tools
  - Help you to manage to environment
  - Do not manage the GPU driver
- To utilize GPUs, make sure the GPU driver is intalled



Conda

- Conda
  - An open source package and environment management system
  - Supports Windows, MacOS, and Linux

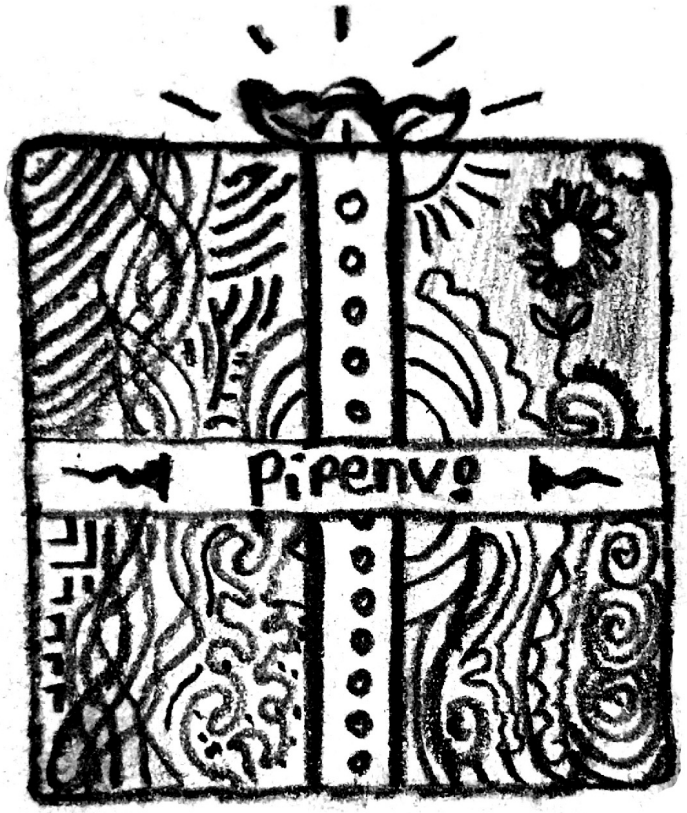
- We take Anaconda as an example

Quick Start - Anaconda

Steps	Linux Command
Install Anaconda with the installer (Check the document for details)	`bash Anaconda3-2021.11-Linux-x86_64.sh`
Create an environment (You can replace `test_env` with your desired environment name)	`conda create -n test_env`
Install packages (You can find the command in the PyTorch official website)	`conda install -n test_env pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch`
Activate the environment	`conda activate test_env`
Run your application	`python ml.py`
Leave the environment	`conda deactivate`

Pipenv

- Pipenv
  - A tool that creates and manages a virtualenv



Quick Start - Pipenv

- To know more about Pipenv, please check the document

Steps	Linux Command
Install Pipenv with pip3	`pip3 install pipenv`
Install packages	`pipenv install numpy torchvision torch --index https://download.pytorch.org/whl/cu113`
Activate the environment	`pipenv shell`
Run your application	`python ml.py`
Leave the environment	`Ctrl + D`

Summary

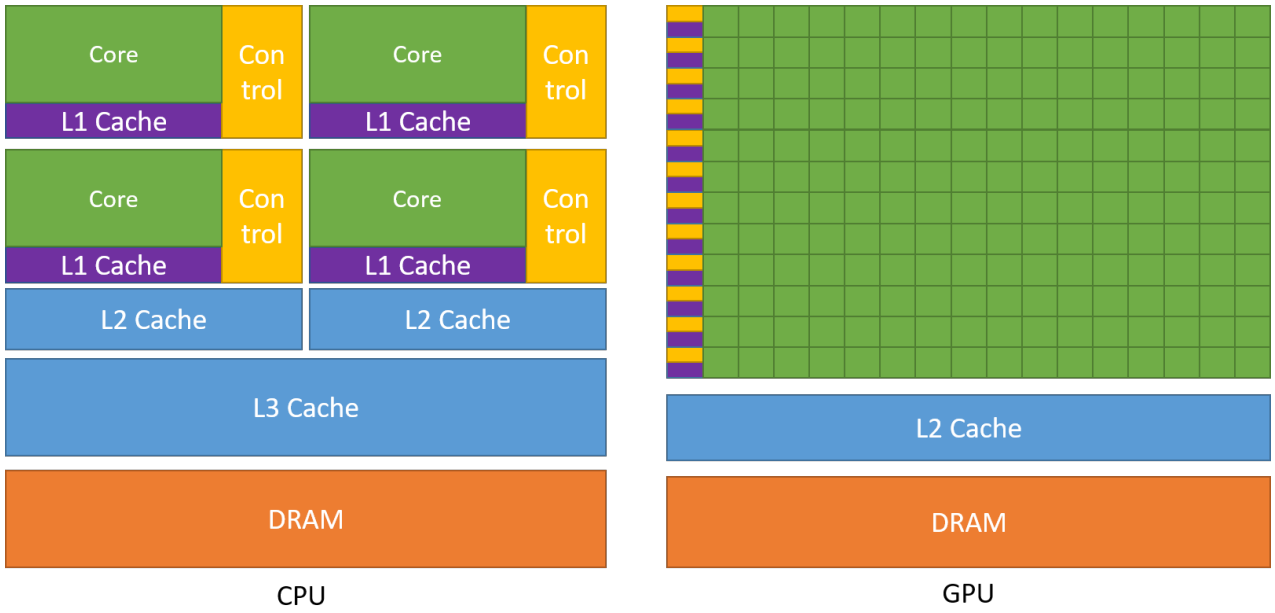
- To utilize GPU, you must install driver on your host machine
- Using Conda or Pipenv to build environments is recommended
  - Portable
  - Reproducible
  - Applications do not affect each other
- You can stop here if you just want to finish the homework
- Why is PyTorch so convenient?
  - "We ship with everything in-built (PyTorch binaries include CUDA, CuDNN, NCCL, MKL, etc.)." [Reference]

Outline

- Overview
- Package Management Tools
- **GPU**
  - NVIDIA GPUs
  - Software Stack
  - NVIDIA Driver
  - CUDA
- Docker
- Conclusion

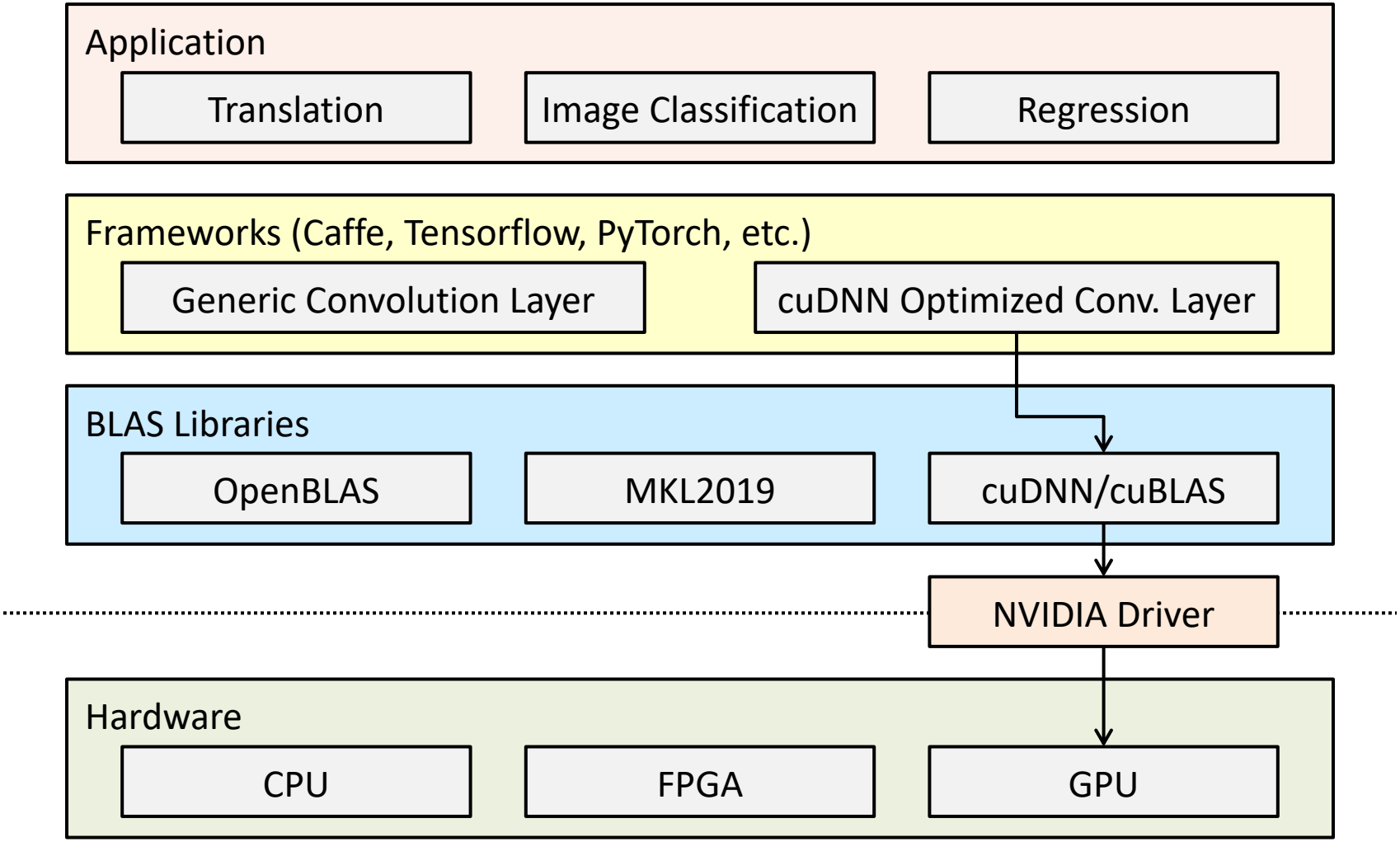
NVIDIA GPUs

- General Purpose Graphics Processing Units (GPGPU)
  - GPUs are originally designed for computer graphic applications
  - GPU is good at parallelizing "simple and repetitive" computations
    - E.g., matrix multiplication
  - There are massive matrix multiplication computations in ML models
    - We use GPU to accelerate ML model training



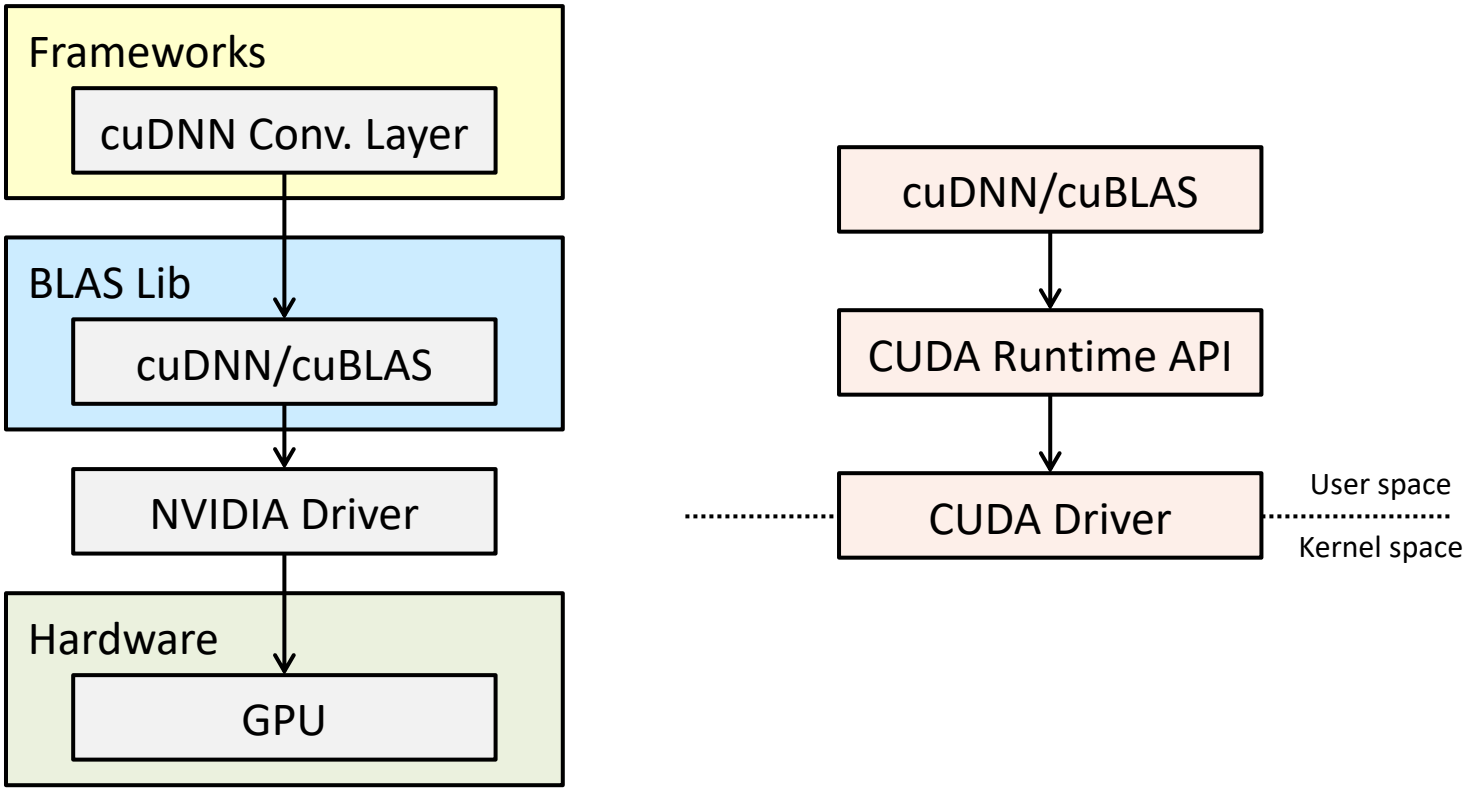
https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Software Stack



NVIDIA Driver

- NVIDIA driver
  - The software that allows operating systems (OS) to communicate with GPUs
  - Includes kernel modules



CUDA

- Compute Unified Device Architecture (CUDA)
  - "A parallel computing platform and application programming interface that allows software to use NVIDIA GPUs" [Wikipedia]
- CUDA Runtime API vs. CUDA Driver API
  - **The driver CUDA version must ≥ the runtime CUDA version**
  - Check the driver CUDA version

```
d08922025@linux-server-3:~/sandbox$ nvidia-smi
Tue Feb 15 09:31:53 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.47.03    Driver Version: 510.47.03    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
```

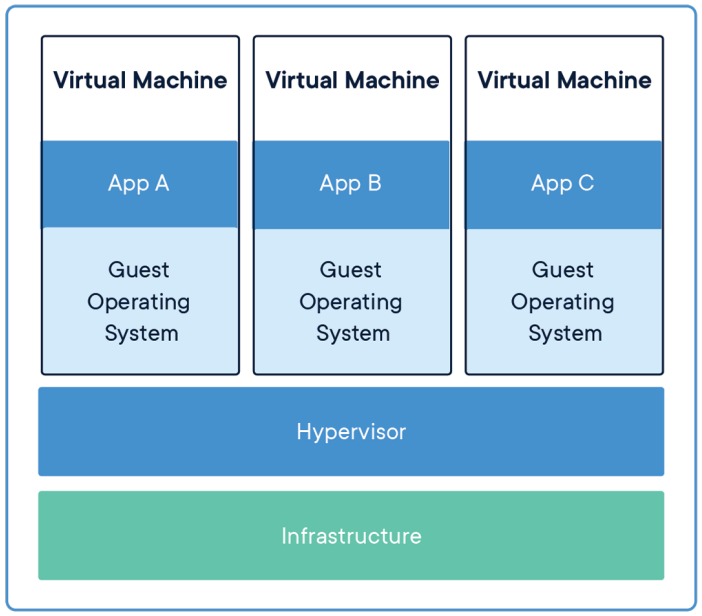
  - When we "install CUDA"
    - We **usually** refer to CUDA runtime
    - You should check the framework compatibility
    - The version should not be greater than the driver CUDA version
    - You should choose the runtime CUDA version carefully

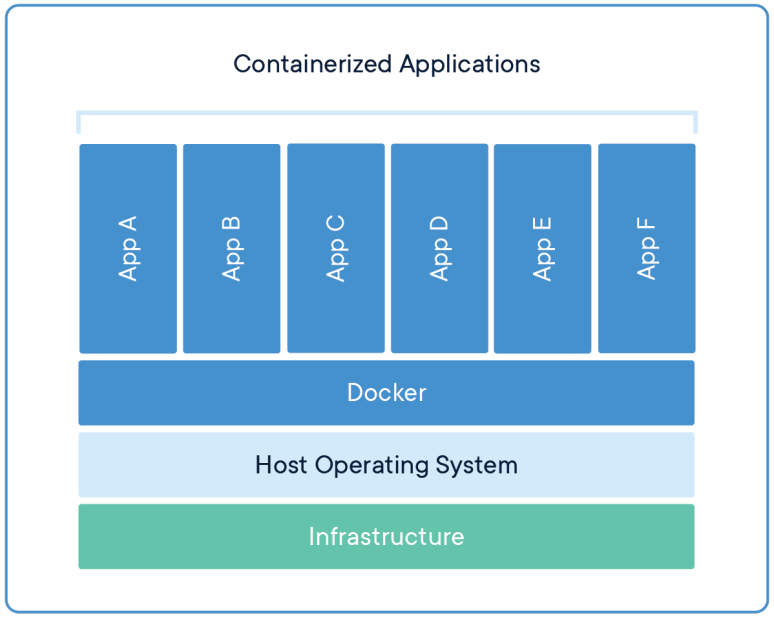
Outline

- Overview
- Package Management Tools
- GPU
- **Docker**
  - Virtualization
  - Why using Container?
  - Contanerization with Docker
  - Pulling Docker Images
  - NVIDIA Docker
- Conclusion

Virtualization

- Virtual machine (VM) and container





- You only have to know that
  - Containers only virtualize software layers above the OS level
    - It is a good choice if we only focus on specific hardware (e.g., NVIDIA GPUs)
  - Containers are relatively lightweight

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

Why using Container?

- Containers can virtualize more complex environments
  - Even if you "only want to train models"
    - You may use other frameworks that do not ship with CUDA and cuDNN
    - You may need NCCL to perform efficient parallel and distributed training
    - You may need to run an old version PyTorch, but the default CUDA version is too old to communicate with the latest powerful GPU

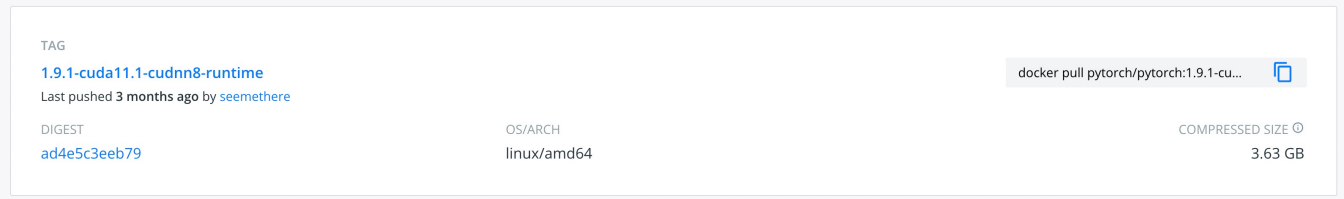
- Slurm and Kubernetes are popular server management tools in both academia and industry
  - Slurm supports singularity container
  - Kubernetes runs application in Docker containers

Containerization with Docker

- Docker
  - A platform for you to build and run with containers
  - Docker installation
    - Docker Desktop (for Mac and Windows) runs a VM
- Docker image
  - A set of instructions for creating a Docker container
- Steps of setting up environment with Docker
  - Install Docker
    - One-time effort
  - Build/pull an image
    - There are lots of built images
  - Run the container
  - Run your application

Pulling Docker Images

- Docker Hub
  - A place for finding and sharing Docker images
    - E.g., Docker Hub repository of PyTorch
- Check the Docker Hub and find the image tag
  - 1.9.1-cuda11.1-cudnn8-devel vs. 1.9.1-cuda11.1-cudnn8-runtime?



  - Run "docker pull <image_tag>"

```
d08922025@linux-server-3:~/sandbox$ docker pull pytorch/pytorch:1.9.1-cuda11.1-cudnn8-runtime
1.9.1-cuda11.1-cudnn8-runtime: Pulling from pytorch/pytorch
284055322776: Already exists
74339e6e5c51: Pull complete
260f45ece716: Pull complete
343d1e51332d: Pull complete
Digest: sha256:ad4e5c3eeb79109fbdf277eb4286684058c6e3f7d7909e318757d727cc96580c
Status: Downloaded newer image for pytorch/pytorch:1.9.1-cuda11.1-cudnn8-runtime
docker.io/pytorch/pytorch:1.9.1-cuda11.1-cudnn8-runtime
```

NVIDIA Docker (1/2)

- Using GPUs in Docker container makes container less portable
  - Containers work in user space
    - Root privilege only means you can use some privileged system calls
  - Using NVIDIA GPUs requires kernel modules and user-level libraries
    - The CUDA version of the driver user-space modules must be exactly the same as the CUDA version of the driver kernel modules
    - The runtime CUDA version can be smaller than the driver CUDA version
  - The host driver must exactly match the version of the driver installed in the container

- We should use NVIDIA Docker
  - Install NVIDIA Docker
  - You do not have to install the NVIDIA driver in the container

https://github.com/NVIDIA/nvidia-docker

NVIDIA Docker (2/2)

- Steps
  - Install the latest NVIDIA driver
    - One-time effort
  - Install NVIDIA Docker
    - One-time effort
  - Build/pull an image
  - Run the container
  - Run your application

```
d08922025@linux-server-3:~/sandbox$ docker run --gpus all -it pytorch/pytorch:1.9.1-cuda11.1-cudnn8-runtime
root@19988f75920c:/workspace# nvidia-smi
Tue Feb 15 10:30:13 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.47.03    Driver Version: 510.47.03    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   38C    P8    28W / 300W |     70MiB / 11264MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

Outline

- ❑ Overview
- ❑ Package Management Tools
- ❑ GPU
- ❑ Docker
- ❑ **Conclusion**

Conclusion

- Whether or not you virtualize your environment
  - You must install the NVIDIA driver on the host to utilize NVIDIA GPUs
  - The runtime CUDA version must be less than or equal to the driver CUDA version
- If you want to use NVIDIA GPUs in containers
  - Using NVIDIA Docker makes your life easier
    - You do not need to install NVIDIA drivers in containers
    - Containers are more portable
  - You only have to pull the built Docker image from Docker Hub
    - You do not have to set up CUDA, cuDNN, and frameworks yourself
    - This is useful especially when the environment is complex

Q&A

Thank You!