

CK 서버 사용자 매뉴얼

2024. 03. 01.

충남대학교 컴퓨터융합학부

목차

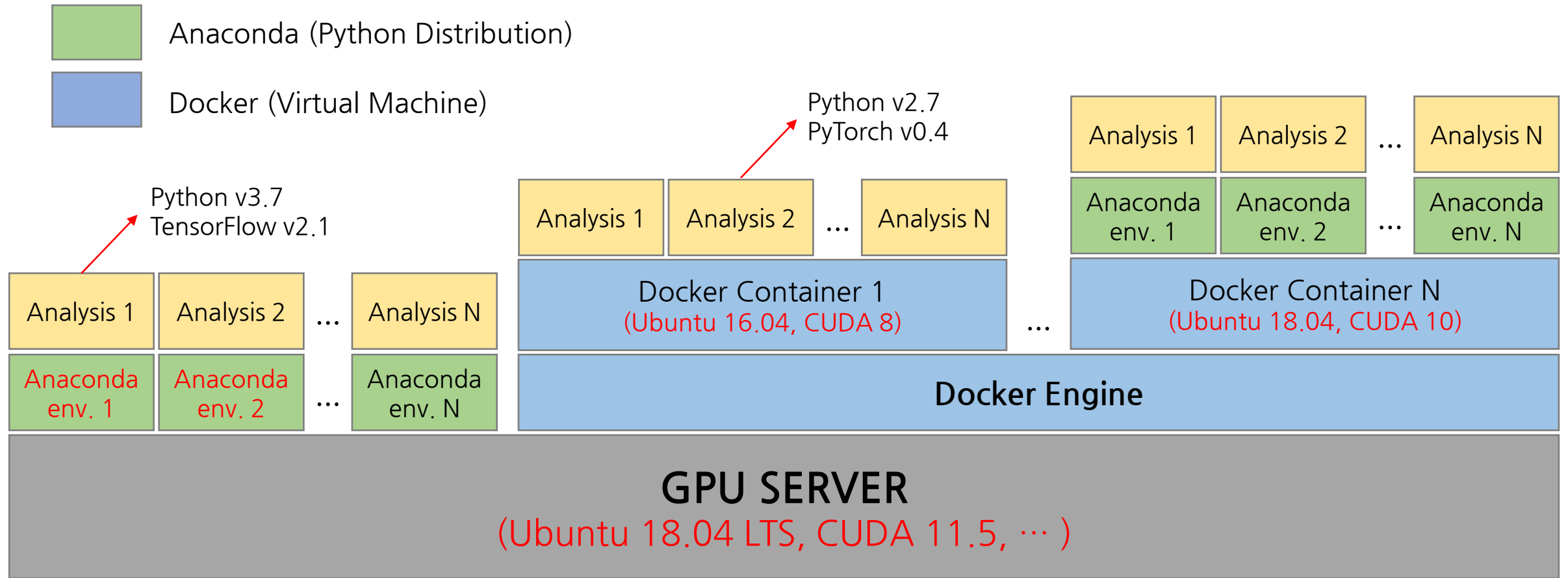
- GPU 서버 스펙 (p. 3)
- GPU 서버에서 제공하는 연구개발 환경 (p. 4-6)
- 사용 가이드 라인 (p. 7-12)
- 핵심 주의사항 (p. 13)
- (권장) Anaconda를 이용한 개발환경 구축 방법 (p. 14-21)
- Docker를 이용한 개발환경 구축 방법 (p. 22-31)

GPU 서버 스펙

CK GPU SERVER

Operating System	Ubuntu 18.04 LTS
Processor	2 x 12-core Intel Xeon E-2650 v4 (48 threads)
Memory	8 x 16GB DDR4 (128GB)
Flash Storage (PCIe)	2TB (/database_grad)
SSD (SATA)	1TB (/), 8TB (/share_hdd)
Graphics Card	8 x NVIDIA Tesla P100 PCIe 16GB
CUDA Toolkit	Driver 495.25, CUDA 11.5
Development Env.	Anaconda3 4.10.1

연구용 GPU 서버가 제공하는 개발환경 개요



- (권장) 기본적으로 Ubuntu 18.04, CUDA 11을 기준으로 Anaconda를 이용해 개발환경 구축
- 버전이 낮아 호환성 문제가 있는 기존 프로젝트는 Docker 컨테이너 이용하여 개발환경 구축
- 필요에 따라 Docker 컨테이너에 직접 개발환경을 구축하거나 Anaconda도 함께 사용하여 구축 가능

GPU 서버 접속 및 프로그램 개발/실행 방법 개요

CK GPU SERVER

IP Address	168.188.128.36
SSH Port	22

id 및 password는
서버 신청 시 개별 메일 안내

- SSH를 통하여 접속 가능하며 대표적인 SSH 접속 도구는 아래와 같음
(22번 포트를 사용하기 때문에 학외에서 SSH로 원격 접속 가능)
 - Windows: Xshell, PuTTY 등
 - MacOS: Terminus, Terminal(built-in app.) 등
- vi, Jupyter Notebook, PyCharm, VS Code 등을 이용하여 코드를 작성하고 원격으로 실행함
 - Vi: SSH 접속 도구를 통해 vi, vim 등의 문서 편집기를 실행하여 코드를 작성하고 직접 프로그램 실행
 - Jupyter Notebook: 웹 브라우저를 통해 파이썬 코드를 작성하고 실행할 수 있는 도구를 사용
 - PyCharm, VS Code: PyCharm, VS Code에서 코드를 작성하고 SSH 원격 인터프리터 기능을 사용하여 개발

저장공간 사용 가이드라인

- 가급적 공간이 많은 /shared_hdd 사용하길 권장
- /share : 1TB
- /shared_hdd : 7.4TB

GPU 할당 가이드라인

아래 가이드라인을 준수하지 않을 경우
하나의 프로그램이 모든 GPU를 사용하여
다른 사용자가 GPU를 사용할 수 없는 경우가 발생할 수 있음

1. 프로그램 실행 전 CPU, 메모리, GPU 사용 현황 확인

- ▶ CPU 및 메모리 현황 확인: htop 명령어
- ▶ GPU 사용 현황 확인: nvidia-smi 명령어

2. 프로그램 실행 환경에 따라 명시적으로 GPU 지정

- ▶ SSH를 통해 직접 프로그램 실행 시 → 슬라이드 8쪽 참조
- ▶ Jupyter Notebook을 통해 프로그램 실행 시 → 슬라이드 9쪽 참조
- ▶ PyCharm의 원격 SSH 인터프리터 실행 시 → 슬라이드 10쪽 참조
- ▶ Docker 컨테이너에서 Host OS의 GPU 접근 → 슬라이드 11쪽 참조

SSH를 통해 직접 프로그램 실행 시 GPU 할당 방법

1. Anaconda를 이용해 개발환경 구축 및 활성화

(슬라이드 13-20쪽 참조)

2. GPU 사용 현황 확인

\$ nvidia-smi

3. 프로그램이 실행될 GPU 지정 (중요)

(예) export CUDA_VISIBLE_DEVICES="0"

(예) export CUDA_VISIBLE_DEVICES="0, 1"

4. 프로그램 실행

\$ python helloworld.py

※ 3. 과정을 무시하고 실행하면

모든 GPU를 점유하여 실행됨

```
new_horizons@GPUSERVER:~$ conda activate py3.7_tf1.14
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ vim helloworld.py
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ nvidia-smi
Mon Nov 18 15:39:21 2019

+-----+
| NVIDIA-SMI 418.67      Driver Version: 418.67      CUDA Version: 10.1      |
+-----+-----+
| GPU   Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
| 0   Tesla P100-PCIE...    On          | 00000000:04:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   34C    P0      27W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 1   Tesla P100-PCIE...    On          | 00000000:06:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   34C    P0      24W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 2   Tesla P100-PCIE...    On          | 00000000:07:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   39C    P0      27W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 3   Tesla P100-PCIE...    On          | 00000000:08:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   34C    P0      25W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 4   Tesla P100-PCIE...    On          | 00000000:0C:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   34C    P0      25W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 5   Tesla P100-PCIE...    On          | 00000000:0D:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   36C    P0      26W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 6   Tesla P100-PCIE...    On          | 00000000:0E:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   33C    P0      25W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+
| 7   Tesla P100-PCIE...    On          | 00000000:0F:00:0 Off |           0MiB / 16280MiB |      0%      Default |
| N/A   32C    P0      25W / 250W |      0MiB / 16280MiB |           |           |
+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
+-----+-----+
| No running processes found                                     |
+-----+

(py3.7_tf1.14) new_horizons@GPUSERVER:~$
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ export CUDA_VISIBLE_DEVICES="0"
(py3.7_tf1.14) new_horizons@GPUSERVER:~$
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ python helloworld.py
1.14.0
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ █
```

[vim을 이용해 간단한 코드를 작성하고 실행하는 예제]

Jupyter Notebook을 통한 프로그램 실행 시 GPU 할당 방법

1. Anaconda를 이용해 개발환경 구축 및 활성화
(슬라이드 13-20쪽 참조)

2. GPU 사용 현황 확인

\$ nvidia-smi

3. Jupyter Notebook이 점유할 GPU 지정 (중요)

(예) export CUDA_VISIBLE_DEVICES="0"

(예) export CUDA_VISIBLE_DEVICES="0, 1"

4. Jupyter Notebook 실행

\$ jupyter notebook ...

```
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ export CUDA_VISIBLE_DEVICES="0"  
(py3.7_tf1.14) new_horizons@GPUSERVER:~$ jupyter notebook ...
```

Jupyter notebook 실행 전에 사용할 GPU를 지정하는 예시

- ※ 3. 과정을 무시하고 Jupyter Notebook을 실행하면
모든 GPU를 점유하여 실행됨

PyCharm - SSH 인터프리터를 사용하는 경우 GPU 할당 방법

※ PyCharm SSH 인터프리터 기능은 professional 버전부터 지원하는데,
대학생은 free educational license를 받아서 사용할 수 있음

1. Anaconda를 이용해 개발환경 구축 및 활성화
(슬라이드 13-20쪽 참조)

2. GPU 사용 현황 확인

\$ nvidia-smi

3. 실행되는 프로그램이 점유할 GPU 지정 (중요)
(실행될 스크립트에 오른쪽과 같은 코드 삽입)

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="0"

...
```

4. 프로그램 실행

※ 3. 과정을 무시하고 실행하면 모든 GPU를 점유하여 실행됨

Docker 컨테이너에서 GPU 할당 방법

```
new_horizons@GPUSERVER:~$ docker run -it --gpus all ubuntu:18.04
```

```
root@152a183ae029:/#  
root@152a183ae029:/#  
root@152a183ae029:/# nvidia-smi  
Mon Mar 1 14:53:41 2021
```

NVIDIA-SMI		460.32.03		Driver Version: 460.32.03		CUDA Version: 11.2	
GPU Fan	Name Temp	Persistence-M Perf	Bus-Id Pwr:Usage/Cap	Disp.A Memory-Usage	Volatile GPU-Util	Uncorr. Compute M.	ECC MIG M.
0	Tesla P100-PCIE...	Off	00000000:04:00:0	Off	0%	0	
N/A	33C P0	33W / 250W	0MiB / 16280MiB			Default	N/A
1	Tesla P100-PCIE...	Off	00000000:06:00:0	Off	0%	0	
N/A	32C P0	31W / 250W	0MiB / 16280MiB			Default	N/A
2	Tesla P100-PCIE...	Off	00000000:07:00:0	Off	0%	0	
N/A	38C P0	33W / 250W	0MiB / 16280MiB			Default	N/A
3	Tesla P100-PCIE...	Off	00000000:08:00:0	Off	0%	0	
N/A	33C P0	31W / 250W	0MiB / 16280MiB			Default	N/A
4	Tesla P100-PCIE...	Off	00000000:0C:00:0	Off	0%	0	
N/A	33C P0	32W / 250W	0MiB / 16280MiB			Default	N/A
5	Tesla P100-PCIE...	Off	00000000:0D:00:0	Off	0%	0	
N/A	35C P0	33W / 250W	0MiB / 16280MiB			Default	N/A
6	Tesla P100-PCIE...	Off	00000000:0E:00:0	Off	0%	0	
N/A	33C P0	31W / 250W	0MiB / 16280MiB			Default	N/A
7	Tesla P100-PCIE...	Off	00000000:0F:00:0	Off	4%	0	
N/A	31C P0	27W / 250W	0MiB / 16280MiB			Default	N/A
Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	
	ID	ID				Usage	
No running processes found							
root@152a183ae029:/#							

`docker run -ti --gpus all ubuntu:18.04`

→ Host OS의 GPU를 컨테이너가 접근할 수 있는 명령어

Docker 사용에 대한 핵심 가이드라인

본래 Docker의 실행 권한은 root에게만 있으나, docker 그룹을 통해서 그룹 내 사용을 허용하고 있음
그래서 잘못된 명령어 사용으로 다른 사용자의 컨테이너와 이미지를 삭제할 가능성이 있음
따라서 아래의 가이드라인에 따라 본인의 이미지 및 컨테이너를 관리해야 함

기본적인 개발 환경은 Anaconda를 권장하며, Docker는 Anaconda로 구축이 불가능한 경우에만 사용할 것을 추천

1. Docker CLI 사용에 대한 공식 매뉴얼 숙지(<https://docs.docker.com/reference/>)

2. 컨테이너는 본인의 **계정명을 이름에 포함되도록 지정**하여 생성 및 실행

(예) `docker run -it --name "계정명" ubuntu:18.04`

→ 계정명이 user01이라면, `docker run it --name "user01-python" ubuntu:18.04`

3. 사용이 끝난 컨테이너는 **이름으로 필터링하여 삭제**

(예) `docker rm $(docker ps -a -q -f "name=계정명" -f "status=exited")`

→ user01-python 이라는 컨테이너 삭제 시, `docker rm $(docker ps -a -q -f "name=user01-python" -f "status=exited")`

연구용 GPU 서버 사용 주의사항

- 다른 사용자 및 본인의 작업을 고려하여 **적절한 수준의 작업을 GPU에 할당**
(슬라이드 7-11쪽 참조)
- Docker 관련 리소스(컨테이너, 이미지 등)에 **본인 계정 이름을 넣어 관리**
(슬라이드 12쪽 참조)
- 사용자의 홈 폴더를 백업하는 서비스를 제공하지 않기 때문에 **중요 파일은 사용자 스스로 외부 디스크에 정기적으로 백업**
- 학과 외부의 다른 사용자에게 **계정을 양도/판매하는 행위 금지**
- 데이터 분석 및 딥러닝 관련 작업 외 **서버에 부담을 주는 다른 작업은 금지**
(예) 비트코인 채굴, 토렌트 운영, 영상 스트리밍, 홈페이지 운영 등

Anaconda를 이용한 개발환경 구축 [1/8]

```
condauser@GPUSERVER:~$ conda env list
# conda environments:
#
base                    * /etc/anaconda3
shared_py3.6_tf1.12     /etc/anaconda3/envs/shared_py3.6_tf1.12
shared_py3.7_torch1.0   /etc/anaconda3/envs/shared_py3.7_torch1.0
condauser@GPUSERVER:~$
```

conda env list → 사용 가능한 개발환경 보기

시스템 관리자가 공유 목적으로 생성한 개발환경

- shared_py3.6_tf1.12 → Python 3.6, TensorFlow 1.12.0
- shared_py3.7_torch1.0 → Python 3.6, Pytorch 1.0

NOTE: 접두어가 shared인 환경은 일반 사용자가 수정 불가

Anaconda를 이용한 개발환경 구축 [2/8]

```
condauser@GPUSERVER:~$ conda create --name myenv python==3.5
Collecting package metadata: done
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /home/condauser/.conda/envs/myenv
```

```
added / updated specs:
- python==3.5
```

```
The following packages will be downloaded:
```

package	build	
ca-certificates-2019.1.23	0	126 KB
certifi-2018.8.24	py35_1	139 KB
pip-10.0.1	py35_0	1.8 MB
python-3.5.0	1	18.1 MB
setuptools-40.2.0	py35_0	571 KB
sqlite-3.26.0	h7b6447c_0	1.9 MB
wheel-0.31.1	py35_0	63 KB
xz-5.0.5	1	572 KB
Total:		23.2 MB

```
The following NEW packages will be INSTALLED:
```

ca-certificates	pkgs/main/linux-64::ca-certificates-2019.1.23-0
certifi	pkgs/main/linux-64::certifi-2018.8.24-py35_1
libedit	pkgs/main/linux-64::libedit-3.1.20181209-hc058e9b_0
libgcc-ng	pkgs/main/linux-64::libgcc-ng-8.2.0-hdf63c60_1
libstdcxx-ng	pkgs/main/linux-64::libstdcxx-ng-8.2.0-hdf63c60_1
ncurses	pkgs/main/linux-64::ncurses-6.1-he6710b0_1
openssl	pkgs/main/linux-64::openssl-1.0.2r-h7b6447c_0
pip	pkgs/main/linux-64::pip-10.0.1-py35_0
python	pkgs/free/linux-64::python-3.5.0-1
readline	pkgs/free/linux-64::readline-6.2-2
setuptools	pkgs/main/linux-64::setuptools-40.2.0-py35_0
sqlite	pkgs/main/linux-64::sqlite-3.26.0-h7b6447c_0
tk	pkgs/free/linux-64::tk-8.5.18-0
wheel	pkgs/main/linux-64::wheel-0.31.1-py35_0
xz	pkgs/free/linux-64::xz-5.0.5-1
zlib	pkgs/main/linux-64::zlib-1.2.11-h7b6447c_3

```
Proceed ([y]/n)? y
```

conda create --name 개발환경명 → 새 개발환경 생성

NOTE: 개발환경 생성과 동시에 Python 버전 및 설치할 패키지 지정 가능

새 개발환경의 각종 파일들이 저장되는 경로 확인

새 개발환경 생성에 따라 다운로드 및 설치되는 패키지 목록을 확인 후 'y'를 입력하여 새 개발환경 생성

Anaconda를 이용한 개발환경 구축 [3/8]

```
condauser@GPUSERVER:~$ conda env list
# conda environments:
#
base                  *  /etc/anaconda3
shared_py3.6_tf1.12    /etc/anaconda3/envs/shared_py3.6_tf1.12
shared_py3.7_torch1.0  /etc/anaconda3/envs/shared_py3.7_torch1.0
myenv                  /home/condauser/.conda/envs/myenv

condauser@GPUSERVER:~$
condauser@GPUSERVER:~$
condauser@GPUSERVER:~$ conda activate myenv
(myenv) condauser@GPUSERVER:~$
(myenv) condauser@GPUSERVER:~$
```

새롭게 생성된 myenv를 확인

conda activate 개발환경명 → 생성된 개발환경으로 진입

NOTE: 정상적으로 진입되면 터미널 입력부분에 (개발환경명)이 출력됨

Anaconda를 이용한 개발환경 구축 [4/8]

```
(myenv) condauser@GPUSERVER:~$  
(myenv) condauser@GPUSERVER:~$ ll  
total 48  
drwxr-xr-x 6 condauser condauser 4096 Mar 1 22:10 ./  
drwxr-xr-x 11 root root 4096 Mar 1 22:09 ../  
-rw-r--r-- 1 condauser condauser 50 Mar 1 22:06 .bash_history  
-rw-r--r-- 1 condauser condauser 220 Apr 5 2018 .bash_logout  
-rw-r--r-- 1 condauser condauser 3771 Apr 5 2018 .bashrc  
drwx----- 3 condauser condauser 4096 Mar 1 18:36 .cache/  
drwxrwxr-x 4 condauser condauser 4096 Mar 1 22:10 .conda/  
drwx----- 3 condauser condauser 4096 Mar 1 18:35 .gnupg/  
drwxrwxr-x 2 condauser condauser 4096 Mar 1 19:17 .keras/  
-rw-r--r-- 1 condauser condauser 807 Apr 5 2018 .profile  
-rw-r--r-- 1 condauser condauser 143 Mar 1 19:17 .python_history  
-rw-r--r-- 1 condauser condauser 55 Mar 1 18:35 .Xauthority  
(myenv) condauser@GPUSERVER:~$  
(myenv) condauser@GPUSERVER:~$ cd .conda  
(myenv) condauser@GPUSERVER:~/.conda$  
(myenv) condauser@GPUSERVER:~/.conda$ ls  
environments.txt  envs  pkgs  
(myenv) condauser@GPUSERVER:~/.conda$  
(myenv) condauser@GPUSERVER:~/.conda$ cd envs  
(myenv) condauser@GPUSERVER:~/.conda/envs$  
(myenv) condauser@GPUSERVER:~/.conda/envs$ ls  
myenv  
(myenv) condauser@GPUSERVER:~/.conda/envs$  
(myenv) condauser@GPUSERVER:~/.conda/envs$ cd myenv  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv$  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv$ ls  
bin  conda-meta  include  lib  share  ssl  x86_64-conda_cos6-linux-gnu  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv$  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv$ cd bin  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv/bin$  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv/bin$ ls  
2to3      idle3      pip        python3.5  pyenv-3.5  toe        xz  
2to3-3.5  idle3.5   pydoc      python3.5-config  reset      tput  
captain   infocmp   pydoc3     python3.5m  sqlite3    tset  
clear     infotocap pydoc3.5   python3.5m-config  tabs       unixz  
c_rehash  ncursesw6-config  python    python3-config  tclsh8.5  wheel  
easy_install  openssl   python3    pyvenv       tic        wish8.5  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv/bin$  
(myenv) condauser@GPUSERVER:~/.conda/envs/myenv/bin$
```

~/.conda/envs 에는
해당 사용자가 생성한 개발환경들이 저장되어 있는 디렉토리임

~/.conda/envs/개발환경명/bin에는
해당 개발환경에 설치된 Python 및 각종 패키지를 확인

NOTE: Python 원격 인터프리터 사용 시 이 경로의 Python을 지정

Anaconda를 이용한 개발환경 구축 [5/8]

```
(myenv) condauser@GPUSERVER:~$ conda search keras-gpu
```

```
Loading channels: done
```

# Name	Version	Build	Channel
keras-gpu	2.0.2	py27_0	pkgs/free
keras-gpu	2.0.2	py35_0	pkgs/free
keras-gpu	2.0.2	py36_0	pkgs/free
keras-gpu	2.0.5	py27_0	pkgs/free
keras-gpu	2.0.5	py35_0	pkgs/free
keras-gpu	2.0.5	py36_0	pkgs/free
keras-gpu	2.0.8	py27hde4dcf2_0	pkgs/main
keras-gpu	2.0.8	py35ha2fb4ba_0	pkgs/main
keras-gpu	2.0.8	py36h0585f72_0	pkgs/main
keras-gpu	2.1.2	py27_0	pkgs/main
keras-gpu	2.1.2	py35_0	pkgs/main
keras-gpu	2.1.2	py36_0	pkgs/main
keras-gpu	2.1.3	py27_0	pkgs/main
keras-gpu	2.1.3	py35_0	pkgs/main
keras-gpu	2.1.3	py36_0	pkgs/main
keras-gpu	2.1.4	py27_0	pkgs/main
keras-gpu	2.1.4	py35_0	pkgs/main
keras-gpu	2.1.4	py36_0	pkgs/main
keras-gpu	2.1.5	py27_0	pkgs/main
keras-gpu	2.1.5	py35_0	pkgs/main
keras-gpu	2.1.5	py36_0	pkgs/main
keras-gpu	2.1.6	py27_0	pkgs/main
keras-gpu	2.1.6	py35_0	pkgs/main
keras-gpu	2.1.6	py36_0	pkgs/main
keras-gpu	2.2.0	0	pkgs/main
keras-gpu	2.2.2	0	pkgs/main
keras-gpu	2.2.4	0	pkgs/main

```
(myenv) condauser@GPUSERVER:~$
```

conda search 패키지명 → 설치 가능한 패키지 검색

Anaconda를 이용한 개발환경 구축 [6/8]

```
(myenv) condauser@GPUSERVER:~$ conda install keras-gpu
```

```
Collecting package metadata: done  
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /home/condauser/.conda/envs/myenv
```

```
added / updated specs:  
- keras-gpu
```

```
The following packages will be downloaded:
```

package	build	
absl-py-0.4.1	py35_0	144 KB
astor-0.7.1	py35_0	43 KB
astor-0.7.1	py35_0	43 KB

```
The following NEW packages will be INSTALLED:
```

_tfselect	pkgs/main/linux-64::_tfselect-2.1.0-gpu
absl-py	pkgs/main/linux-64::absl-py-0.4.1-py35_0
astor	pkgs/main/linux-64::astor-0.7.1-py35_0
blas	pkgs/main/linux-64::blas-1.0-mkl
cudatoolkit	pkgs/main/linux-64::cudatoolkit-9.2.0
cudnn	pkgs/main/linux-64::cudnn-7.3.1-cuda9.2_0
cupti	pkgs/main/linux-64::cupti-9.2.148-0
gast	pkgs/main/linux-64::gast-0.2.0-py35_0
grpcio	pkgs/main/linux-64::grpcio-1.12.1-py35hdbcae40_0
six	pkgs/main/linux-64::six-1.11.0-py35_1
tensorboard	pkgs/main/linux-64::tensorboard-1.10.0-py35hf484d3e_0
tensorflow	pkgs/main/linux-64::tensorflow-1.10.0-gpu_py35hd9c640d_0
tensorflow-base	pkgs/main/linux-64::tensorflow-base-1.10.0-gpu_py35had579c0_0
tensorflow-gpu	pkgs/main/linux-64::tensorflow-gpu-1.10.0-hf154084_0
termcolor	pkgs/main/linux-64::termcolor-1.1.0-py35_1
werkzeug	pkgs/main/linux-64::werkzeug-0.14.1-py35_0
yaml	pkgs/main/linux-64::yaml-0.1.7-had09818_2

```
Proceed ([y]/n)? y
```

conda install 패키지명 → 새로운 패키지 설치
(본 예제는 keras gpu 버전을 설치)

NOTE: 패키지 설치 시 특정 버전 지정 가능 (예: keras-gpu==2.2.2)
만약 지정하지 않는다면 설치 가능한 가장 높은 버전을 설치함

Anaconda를 이용한 개발환경 구축 [7/8]

```
(myenv) condauser@GPUSERVER:~$ python
Python 3.5.0 [Continuum Analytics, Inc.] (default, Oct 19 2015, 21:57:25)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> import keras
Using TensorFlow backend.
>>> keras.__version__
'2.2.2'
>>>
```

설치된 패키지가 잘 import 되는지와 버전 확인

```
(myenv) condauser@GPUSERVER:~$ python keras_mnist.py
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.n
11493376/11490434 [=====] - 16s 1us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
2019-03-01 23:26:53.594955: I tensorflow/core/platform/cpu_feature_
ructions that this TensorFlow binary was not compiled to use: SSE4.
2019-03-01 23:26:54.012320: I tensorflow/core/common_runtime/gpu/gp
k_properties:
0279 - val_acc: 0.9902
Epoch 10/12
60000/60000 [=====] - 4s 68us/step - loss:
0248 - val_acc: 0.9915
Epoch 11/12
60000/60000 [=====] - 4s 68us/step - loss:
0265 - val_acc: 0.9910
Epoch 12/12
60000/60000 [=====] - 4s 69us/step - loss:
0267 - val_acc: 0.9907
Test loss: 0.026707943036882717
Test accuracy: 0.9907
(myenv) condauser@GPUSERVER:~$
```

설치된 패키지의 예제 코드 실행을 통해 동작 확인
(본 예제는 keras mnist를 gpu를 이용해 실행)

Anaconda를 이용한 개발환경 구축 [8/8]

```
(myenv) condauser@GPUSERVER:~$ conda deactivate
condauser@GPUSERVER:~$
condauser@GPUSERVER:~$
condauser@GPUSERVER:~$ conda env list
# conda environments:
#
base                  *  /etc/anaconda3
shared_py3.6_tf1.12    /etc/anaconda3/envs/shared_py3.6_tf1.12
shared_py3.7_torch1.0  /etc/anaconda3/envs/shared_py3.7_torch1.0
myenv                  /home/condauser/.conda/envs/myenv

condauser@GPUSERVER:~$
condauser@GPUSERVER:~$
condauser@GPUSERVER:~$ conda remove --name myenv --all

Remove all packages in environment /home/condauser/.conda/envs/myenv:

## Package Plan ##

  environment location: /home/condauser/.conda/envs/myenv

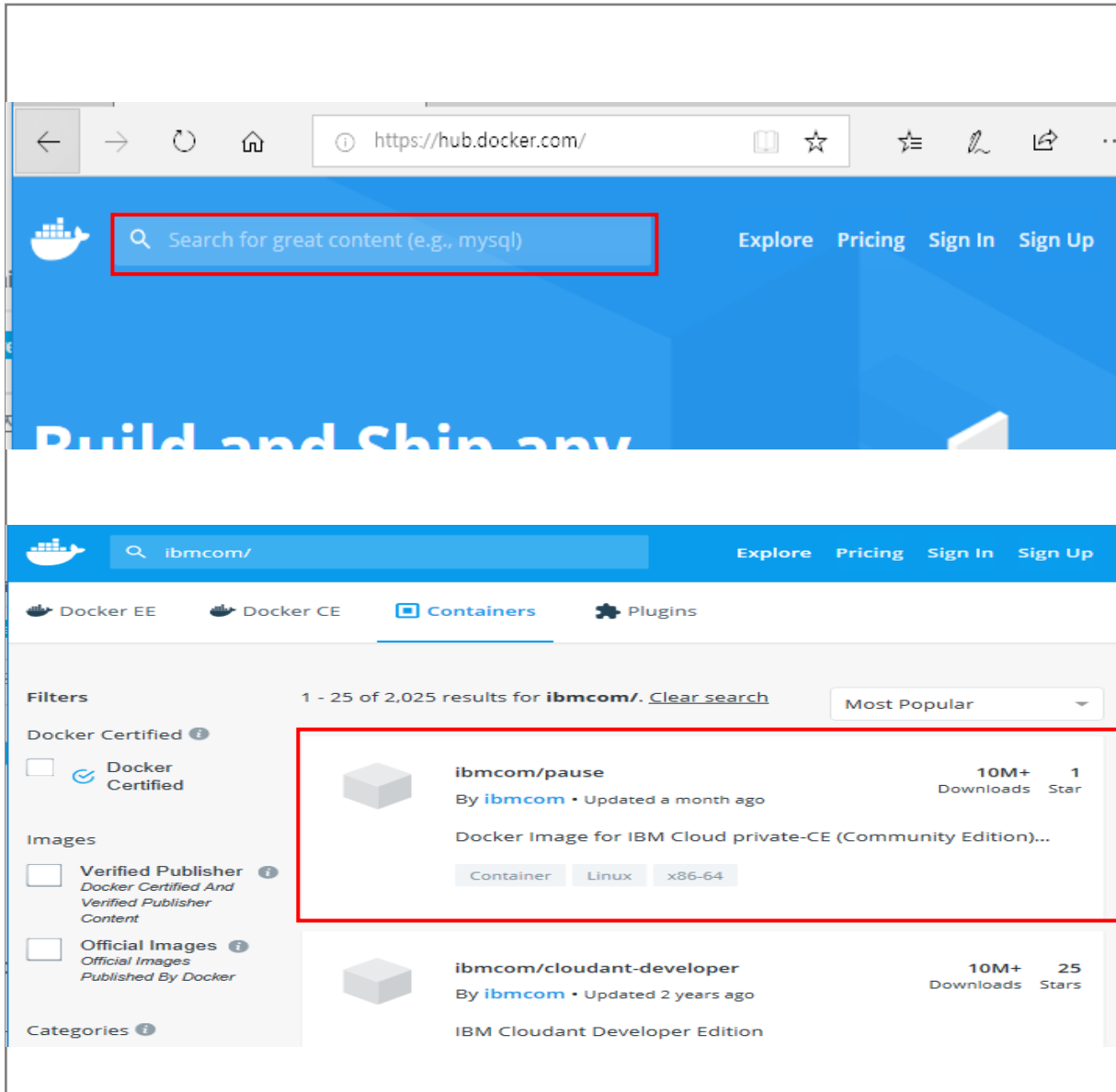
The following packages will be REMOVED:

  _tfflow_select-2.1.0-gpu
  absl-py-0.4.1-py35_0
```

conda deactivate → 현재의 개발환경에서 나오기

conda remove --name 개발환경명 --all → 개발환경 삭제

Docker를 이용한 개발환경 구축 [1/10]



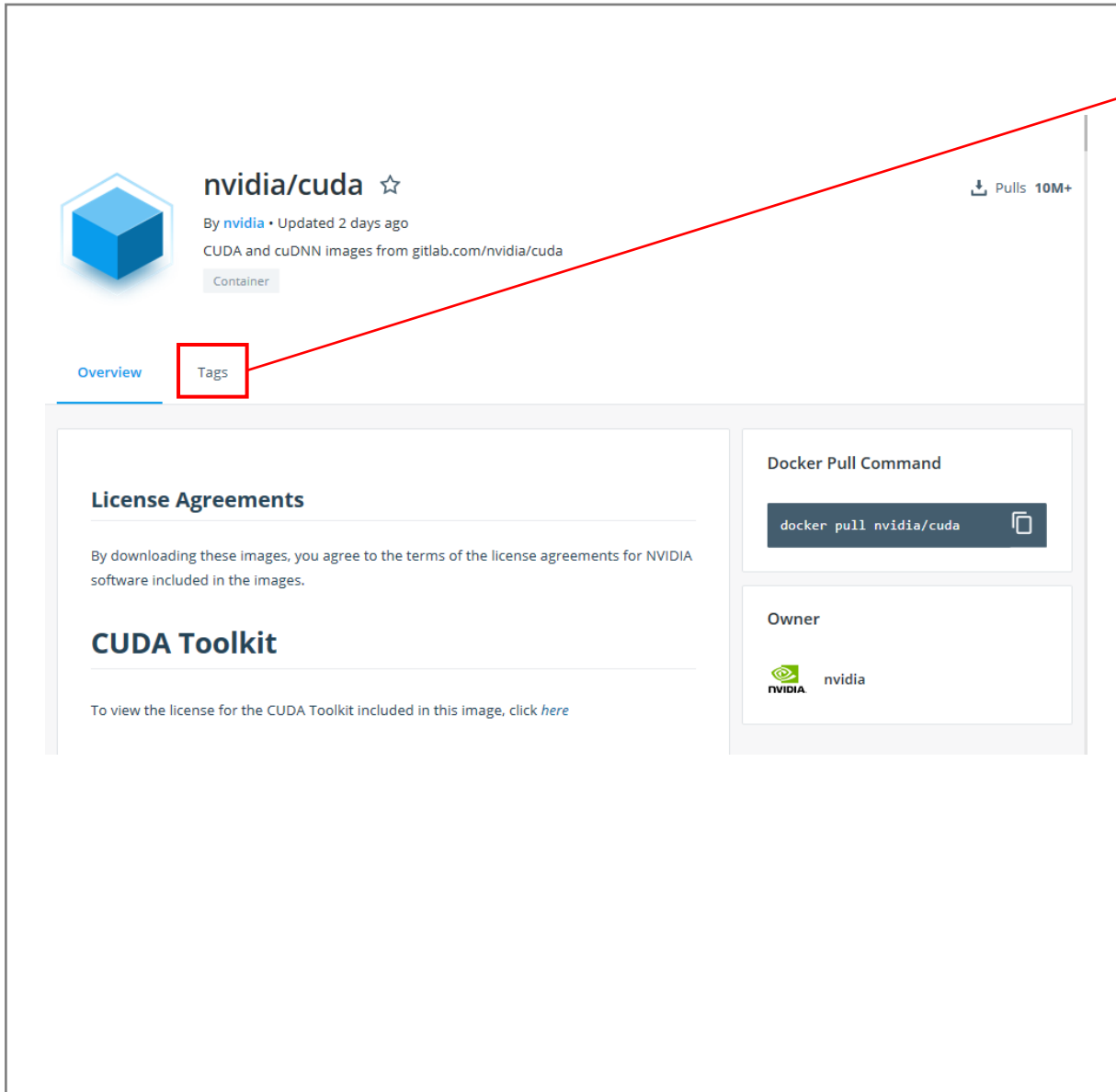
Docker Hub ?

GitHub과 유사하게 docker 이미지를 다른 사용자와 공유하는 플랫폼

원하는 docker 이미지 검색 (예: nvidia , ibmcom, tensorflow 등)

Docker 이미지는 소유자ID/Repository명 으로 구분됨
많이 사용되는 이미지는 다운로드 횟수, 별점이 높음

Docker를 이용한 개발환경 구축 [2/10]



nvidia/cuda ☆

By **nvidia** • Updated 2 days ago

CUDA and cuDNN images from gitlab.com/nvidia/cuda

Container

Overview **Tags**

License Agreements

By downloading these images, you agree to the terms of the license agreements for NVIDIA software included in the images.


CUDA Toolkit

To view the license for the CUDA Toolkit included in this image, click [here](#)

Docker Pull Command

```
docker pull nvidia/cuda
```

Owner

 **nvidia**

Tag를 통해 해당 docker 이미지의 버전, 특징을 확인 가능

Tags (230)

10.1-runtime-centos6 610 MB

Last update: 2 days ago

10.1-devel-centos6 1 GB

Last update: 2 days ago

10.1-cudnn7-runtime-centos6 866 MB

Last update: 2 days ago

Docker를 이용한 개발환경 구축 [3/10]

```
[testuser03@minsky ~]$ docker search nvidia/cuda
```

NAME	DESCRIPTION	STARS
nvidia/cuda	CUDA and cuDNN images from gitlab.com/nvidia...	479
nvidia/digits	DEPRECATED	61
anthonytatowicz/eth-cuda-miner	Image for creating CUDA enabled ethminer con...	14
nvidia/k8s-device-plugin	Images for github.com/NVIDIA/k8s-device-plug...	12
nvidia/opencv	OpenCL images from gitlab.com/nvidia/opencv	9
nvidia/opencv	Beta OpenGL images from gitlab.com/nvidia/op...	9
nvidia/cudagl	CUDA + OpenGL images from gitlab.com/nvidia/...	8
nvidia/cuda-ppc64le	CUDA and cuDNN images from gitlab.com/nvidia...	8
shelmangroup/coreos-nvidia-driver-installer	Container to install Nvidia GPU drivers on C...	4
patssissons/xmrig-nvidia	xmrig-nvidia container for nvidia-docker eng...	2
publicrepository/nvidia-nvvs		1
alexcpn/nvidia-cuda-grpc	# This is a fat image used just for developm...	0
409c2e5be593/nvidia-cuda-ce	nvidia-cuda-ce	0
mindprince/nvidia_gpu_prometheus_exporter	https://github.com/mindprince/nvidia_gpu_pro...	0
sulfurheron/nvidia-cuda		0
dcwangmit01/aws-nvidia-bootstrap	Kubernetes KOPS hook installing Nvidia GPU d...	0
c3sr/go-with-cudatoolkit	Docker images with Go and NVIDIA CUDA Toolki...	0

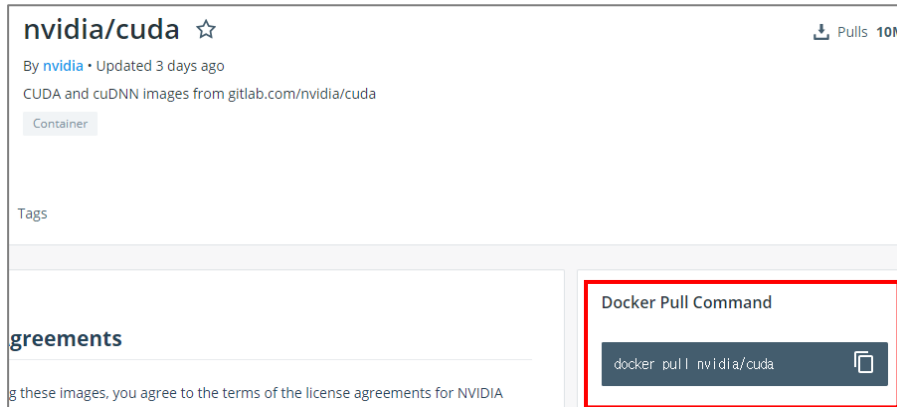
docker search 키워드

Docker Hub가 아닌, Docker 데몬에서도 이미지 검색 가능
(검색된 이미지들은 Docker Hub와 동일한 이미지)

Note: Docker 데몬을 통한 검색보다 Docker Hub(웹)을 통한 검색이 보다 많은 정보를 제공하기 때문에 Docker Hub를 통한 검색을 추천

Docker를 이용한 개발환경 구축 [4/10]

```
[testuser03@minsky ~]$ docker pull centos
Using default tag: latest
latest: Pulling from library/centos
23856db8b6dd: Pull complete
Digest: sha256:184e5f35598e333bfa7de10d8fblcebb5ee4df5bc0f970bf2ble
Status: Downloaded newer image for centos:latest
[testuser03@minsky ~]$ docker images
```



```
[testuser03@minsky ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nvidia/cuda          10.1-runtime-centos6 f1348fell65c       2 days ago         1.14GB
ubuntu               18.04              2457271cld0e       4 weeks ago        128MB
ubuntu               16.04              d67051c921d7       6 weeks ago        155MB
centos                7.6.1810           99936dbde360       3 months ago       256MB
centos                latest              74428bb25902       3 months ago       257MB
bmcom/powerai        1.5.4-all-ubuntu18.04-py3 5fdfbb09ae06       3 months ago       11.6GB
bmcom/powerai        1.5.4-all-ubuntu18.04 4b8f2cfbd12f       3 months ago       9.96GB
```

docker pull 이미지명 또는 docker pull 이미지명:태그명

→ 해당 이미지를 local 저장소(서버의 디스크)로 다운로드 받는 명령어
(태그를 특정하지 않을 경우 latest로 태그된 이미지를 다운로드함)

Note: Docker Hub에 해당 이미지를 다운로드 받는 커맨드를 참고

docker images

현재 local 저장소에 저장된 이미지를 모두 출력하는 명령어

Docker를 이용한 개발환경 구축 [5/10]

```
[testuser03@minsky ~]$ docker image rm centos
```

```
Untagged: centos:latest
```

```
Untagged: centos@sha256:184e5f35598e333bfa7de10d8fblcebb5ee4df5bc0f970bf2ble7c7345136426
```

```
Deleted: sha256:74428bb259024327fa78ba04cbb2dfa533a5098e4faa5db26fe99322e5cd41ef
```

```
Deleted: sha256:2e781a1bbdfb0c13bdd08cca845392eb92f82fla8ffcba7cb1a0e6c3864b88f2
```

```
[testuser03@minsky ~]$ docker image rm nvidia/cuda:10.1-runtime-centos6
```

```
Untagged: nvidia/cuda:10.1-runtime-centos6
```

```
Untagged: nvidia/cuda@sha256:4b421eb59e76e47709ba05d71e80b30b7116fbc81f8c8af8c973e35053c615d3
```

```
Deleted: sha256:f1348fell65cc7aa3b9a143d1247f56dc9f95c07fd1603152057550d555a933a
```

```
Deleted: sha256:c422bb879e6abc60a78d7d3f78421429d95260140a28085edd8c9cf5ea4d18d6
```

```
Deleted: sha256:759201035fc49fc3dd6d8f11222081790847af9dc7dd668e17aad9618b901e43
```

```
Deleted: sha256:b39ff4ec7290e5ad6eda6718da5d4b2cfa884155d2c5e67478375b68fadd5b9
```

```
Deleted: sha256:e8bcc099238e4366bb7bf3c9d13860c11a9f8bbcdf8d7e23ccd0cb8e5f9a659
```

```
Deleted: sha256:2541056108effle77f0e4a599d6c066fb7742786ae72ecda8celc07f48b28b01
```

```
Deleted: sha256:5d574ede99e4892f96319a3899212029e69e0b9d28c9a84f20bc628497e931db
```

```
[testuser03@minsky ~]$
```

docker image rm 이미지명 또는 docker image rm 이미지명:태그명

→ 해당 이미지를 local 저장소에서 삭제하는 명령어

주의!

Docker 이미지는 GPU 서버를 사용하는 모든 유저가 삭제 가능하기 때문에 삭제 명령어는 주의하여 사용

Docker를 이용한 개발환경 구축 [6/10]

```
testuser@GPUSERVER:~$ docker run -i -t braintwister/ubuntu-16.04-cuda-9.0 /bin/bash
root@fa004ee5fa05:/#
```

Host OS	컨테이너
<pre>testuser04@GPUSERVER:~\$ cat /etc/issue Ubuntu 18.04.2 LTS \n \l testuser04@GPUSERVER:~\$ nvcc --version nvcc: NVIDIA (R) Cuda compiler driver Copyright (c) 2005-2018 NVIDIA Corporation Built on Sat Aug 25 21:08:03 CDT 2018 Cuda compilation tools, release 10.0, V10.0.130</pre>	<pre>root@843f063771b2:/# cat /etc/issue Ubuntu 16.04.5 LTS \n \l root@843f063771b2:/# nvcc --version nvcc: NVIDIA (R) Cuda compiler driver Copyright (c) 2005-2017 NVIDIA Corporation Built on Fri Sep 1 21:08:03 CDT 2017 Cuda compilation tools, release 9.0, V9.0.176</pre>

```
testuser@GPUSERVER:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
fa004ee5fa05	braintwister/ubuntu-16.04-cuda-9.0	"/bin/bash"	20 minutes ago	Up 20 minutes
4944f168ad52	braintwister/ubuntu-16.04-cuda-8.0	"/bin/bash"	42 hours ago	Up 42 hours
goofy_gauss				

docker run -i -t 이미지명:태그명 실행파일
→ 해당 이미지를 컨테이너로 생성하는 명령어
(예시) docker run -ti ubuntu18.04 /bin/bash
ubuntu18.04 이미지로 컨테이너 생성하며 bash 셸을 실행

Docker를 통해 다른 버전의 리눅스 OS 사용 가능(단 Host OS 보다 낮거나 같은 버전)
Docker를 통해 다른 버전의 CUDA 사용 가능(단 Host OS 보다 낮거나 같은 버전)

docker ps → 현재 프로그램이 실행되고 있는 컨테이너 확인

Docker를 이용한 개발환경 구축 [7/10]

```
new_horizons@GPUSERVER:~$ docker run -it --gpus all ubuntu:18.04
```

```
root@152a183ae029:/#  
root@152a183ae029:/#  
root@152a183ae029:/# nvidia-smi  
Mon Mar 1 14:53:41 2021
```

+-----+ NVIDIA-SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2 +-----+							
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	Tesla P100-PCIE...	Off	00000000:04:00:0	Off	0%	0	
N/A	33C	P0	33W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
1	Tesla P100-PCIE...	Off	00000000:06:00:0	Off	0%	0	
N/A	32C	P0	31W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
2	Tesla P100-PCIE...	Off	00000000:07:00:0	Off	0%	0	
N/A	38C	P0	33W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
3	Tesla P100-PCIE...	Off	00000000:08:00:0	Off	0%	0	
N/A	33C	P0	31W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
4	Tesla P100-PCIE...	Off	00000000:0C:00:0	Off	0%	0	
N/A	33C	P0	32W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
5	Tesla P100-PCIE...	Off	00000000:0D:00:0	Off	0%	0	
N/A	35C	P0	33W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
6	Tesla P100-PCIE...	Off	00000000:0E:00:0	Off	0%	0	
N/A	33C	P0	31W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
7	Tesla P100-PCIE...	Off	00000000:0F:00:0	Off	4%	0	
N/A	31C	P0	27W / 250W	0MiB / 16280MiB		Default	N/A
+-----+							
+-----+							
Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	
ID	ID	ID				Usage	
+-----+							
No running processes found							
+-----+							
root@152a183ae029:/#							

`docker run -ti --gpus all ubuntu:18.04`

→ Host OS의 GPU를 컨테이너가 접근할 수 있는 명령어

Docker를 이용한 개발환경 구축 [8/10]

```
[testuser02@minsky ~]$ docker attach ed7db7219bee  
pwrai@ed7db7219bee:~$
```

docker attach 컨테이너ID

→ 동작중인 컨테이너로 접속하는 명령어

```
pwrai@ed7db7219bee:~$ read escape sequence  
[testuser02@minsky ~]$
```

컨테이너 안에서 Ctrl + p + q 키 입력

→ 컨테이너에서 Host OS로 이동 (컨테이너는 종료되지 않음)

```
pwrai@ed7db7219bee:~$ exit  
exit  
[testuser02@minsky ~]$
```

exit (셸 종료 명령어)

→ 컨테이너에서 Host OS로 이동 (컨테이너는 종료됨)

```
[testuser02@minsky ~]$ docker kill 60bca8edc4a4  
60bca8edc4a4  
[testuser02@minsky ~]$
```

docker kill 컨테이너ID

→ Host OS에서 해당 컨테이너를 종료시키는 명령어

Docker를 이용한 개발환경 구축 [9/10]

```
testuser@GPUSERVER:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
fa004ee5fa05	braintwister/ubuntu-16.04-cuda-9.0	"/bin/bash"	20 minutes ago	Up 20 minutes
4944f168ad52	braintwister/ubuntu-16.04-cuda-8.0	"/bin/bash"	42 hours ago	Up 42 hours

```
testuser@GPUSERVER:~$ docker commit fa004ee5fa05 test_commit_images
```

sha256:8d06a018dc3dc00e7a4e47d5a1241790531d0236b9555d9104532ac783e95a9b

```
testuser@GPUSERVER:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
commit_test_images	latest	fd0b99a962eb	11 seconds ago	2.48GB
p004967/commit_test_images	latest	a2087147fc03	42 hours ago	2.48GB
braintwister/ubuntu-16.04-cuda-9.0	latest	5f9e8e3bbc38	4 weeks ago	2.16GB
nvidia/cuda	latest	d9a8427c8dd9	4 weeks ago	2.35GB
hello-world	latest	fce289e99eb9	2 months ago	1.84kB
braintwister/ubuntu-16.04-cuda-8.0	latest	0eadlef8395f	5 months ago	2.48GB
nvcr.io/nvidia/tensorflow	17.12	19afd620fc8e	15 months ago	2.88GB

doker commit 컨테이너ID 저장할이미지명

→ 현재까지 작업한 컨테이너를 이미지로 저장 (local 저장소)

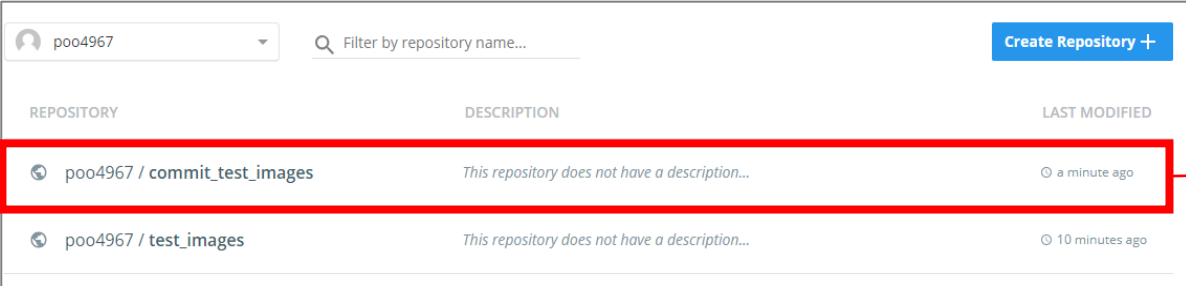
→ 이미지가 현재 서버에 저장됨을 확인

Docker를 이용한 개발환경 구축 [10/10]

```
testuser04@GPUSEVER:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: 

testuser04@GPUSEVER:~$ export DOCKER_ID_USER="poo4967"
testuser04@GPUSEVER:~$ docker tag commit_test_images $DOCKER_ID_USER/commit_test_images

testuser04@GPUSEVER:~$ docker push $DOCKER_ID_USER/commit_test_images
The push refers to repository [docker.io/poo4967/commit_test_images]
340e1727589c: Mounted from poo4967/test_images
de3fe01210b4: Mounted from poo4967/test_images
c134510c9564: Mounted from poo4967/test_images
d807372c383b: Mounted from poo4967/test_images
9f79cd8e84f9: Mounted from poo4967/test_images
43f8c91f5259: Mounted from poo4967/test_images
c90b73c1affe: Mounted from poo4967/test_images
```



REPOSITORY	DESCRIPTION	LAST MODIFIED
poo4967 / commit_test_images	This repository does not have a description...	a minute ago
poo4967 / test_images	This repository does not have a description...	10 minutes ago

docker login

→ commit 한 이미지를 docker hub 에 저장하기 위해 로그인

export DOCKER_ID_USER="Docker Hub ID"

docker tag 이미지명 \$DOCKER_ID_USER/이미지명

→ 해당 계정으로 Docker Hub에 업로드 하려는 이미지 지정

docker push \$DOCKER_ID_USER/이미지명

→ Docker hub에 해당 이미지를 업로드하는 명령어

docker hub 에 로그인하여 업로드된 이미지 확인