

2022 A Basic CS skill: ABC Winter School

Special Topic 1

Team 8

2022 / 02 / 20



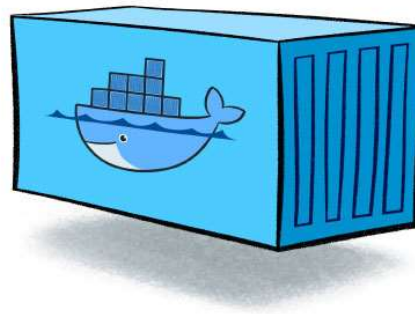
ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY

Docker

- Container를 이용해 각종 응용 프로그램을 빠르게 구축, 테스트, 배포할 수 있도록 도와주는 SW 플랫폼 입니다.
- Linux 환경에서 Container를 생성하고 사용할 수 있도록 도와줍니다.



응용 프로그램



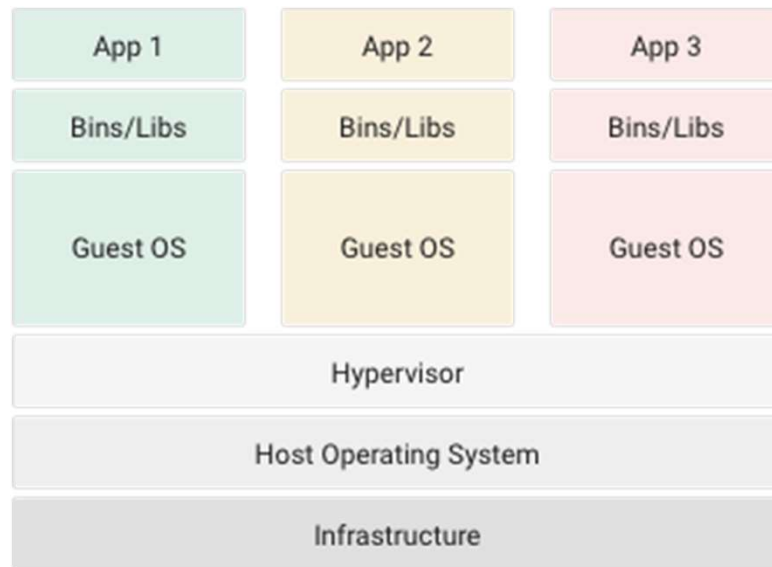
도커 컨테이너



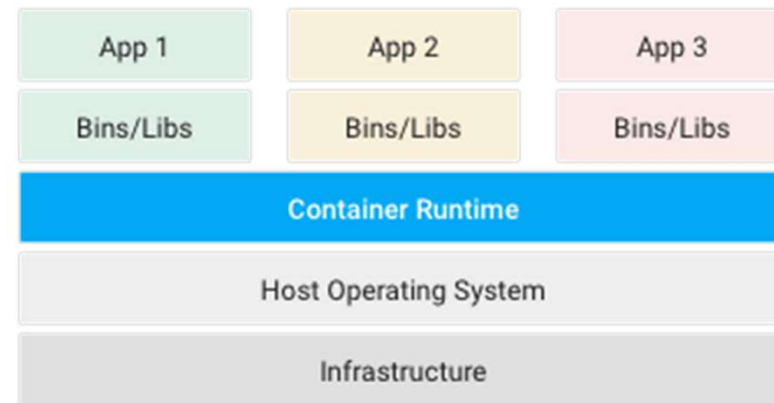
서버

Container

- OS level 가상화라고도 합니다.



Virtual Machines



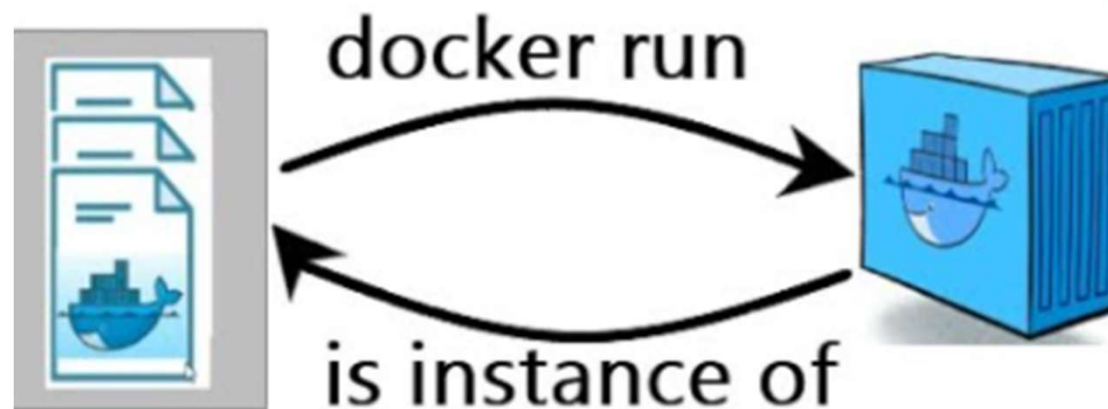
Containers

Container

- VM과는 다르게 개별적인 OS 커널이 없고 Hypervisor 대신 container를 실행해주는 프로그램이 존재합니다.
- 단, container에 따라 **최소화 된 OS 커널**이 존재할 수 있습니다.
- Container 생성 및 관리에 주로 Docker 프로그램을 사용 합니다.
- Container 역시 서로 간섭할 수 없도록 isolation 되어 있습니다.
- 또한 Hardware까지 가상화 하는 것에 비해 매우 가볍습니다.

Docker image

- Docker image는 다른 사람들이 사전에 미리 만들어 놓은 환경 입니다.
- Github처럼 docker hub를 통해 다운 받을 수 있습니다.
- Docker는 이미지를 실행하는 것으로 컨테이너에 해당 이미지에 내포된 환경을 올려서 서비스를 구축/실행 합니다.
- **Docker image가 실행된 상태 => Docker container**



Docker 설치하기

- 우분투 환경에서 다음 명령어를 사용하면 docker가 설치 됩니다.
- `sudo apt-get install docker`
- `sudo apt-get install docker.io`
- 이 후 터미널에 docker를 입력했을 때, 다음과 같은 화면이 나타나면 docker 설치가 성공적으로 완료 된 것입니다.

```
ubuntu@ip-172-31-40-165:~$ docker
Usage: docker [OPTIONS] COMMAND
A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default "/home/ubuntu/snap/docker/796/.docker")
  -c, --context string Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var
                        and default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string   Trust certs signed only by this CA (default "/home/ubuntu/snap/docker/796/.docker/ca.pem")
  --tlscert string     Path to TLS certificate file (default "/home/ubuntu/snap/docker/796/.docker/cert.pem")
  --tlskey string      Path to TLS key file (default "/home/ubuntu/snap/docker/796/.docker/key.pem")
  --tlsverify          Use TLS and verify the remote
  -v, --version        Print version information and quit

Management Commands:
  builder      Manage builds
  config       Manage Docker configs
  container    Manage containers
  context      Manage contexts
  engine       Manage the docker engine
  image        Manage images
  network      Manage networks
  node         Manage Swarm nodes
  plugin       Manage plugins
  secret       Manage Docker secrets
  service      Manage services
  stack        Manage Docker stacks
  swarm        Manage Swarm
  system       Manage Docker
  trust        Manage trust on Docker images
  volume       Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
```

Docker 명령어

- Docker는 os 커널을 사용하기 때문에 반드시 sudo를 붙여서 사용해야 합니다.
 - docker ps : Docker 컨테이너 관리
 - docker search [이미지 이름] : Docker hub에서 이미지 검색
 - docker pull [이미지 이름] : Docker hub에서 이미지 내려 받기
 - docker images : Docker 이미지 관리
 - docker run [이미지 이름] : 이미지를 컨테이너로 실행
 - docker build [Dockerfile 경로] : Dockerfile을 빌드하여 이미지로 생성
 - docker rm [컨테이너 이름] : Docker 컨테이너 제거
 - docker rmi [이미지 이름] : Docker 이미지 제거
 - docker attach [컨테이너 이름] : 실행중인 Docker 컨테이너에 접속

Docker container 관리

- `sudo docker ps -a`를 사용해 실행/종료/대기 중인 컨테이너를 볼 수 있습니다.
- `sudo docker rm [컨테이너 ID 또는 이름]` 으로 종료된 컨테이너를 삭제할 수 있습니다.

```
with1015@ubuntu:~/Docker_study$ sudo docker run abc_mentoring:abc
hello world!
with1015@ubuntu:~/Docker_study$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
c891872f05f0	abc_mentoring:abc	"python hello_world...."	4 minutes ago	Exited (0) 4 minutes ago	
vibrant_greider					
340cd5bdc049	5d6cfd8021e6	"python hello_world...."	52 minutes ago	Exited (0) 52 minutes ago	
focused_shockley					

```
with1015@ubuntu:~/Docker_study$ sudo docker rm 340cd5bdc049
340cd5bdc049
with1015@ubuntu:~/Docker_study$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
c891872f05f0	abc_mentoring:abc	"python hello_world...."	5 minutes ago	Exited (0) 5 minutes ago	
vibrant_greider					

```
with1015@ubuntu:~/Docker_study$
```


Docker image 실행

- Dockerfile이 빌드 되면 docker images 명령어를 통해 이미지가 생성된 것을 확인할 수 있습니다.

```
with1015@ubuntu:~/Docker_study$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
abc_mentoring	abc	5d6cfd8021e6	43 minutes ago	429MB
ubuntu	16.04	8185511cd5ad	4 weeks ago	132MB

```
with1015@ubuntu:~/Docker_study$
```

- 생성된 이미지로 컨테이너를 만들고 실행하기 위해서는 다음 명령어를 사용해야 합니다.
 - `sudo docker run REPOSITORY:TAG`
 - 위 예시의 경우, `sudo docker run abc_mentoring:abc`

```
with1015@ubuntu:~/Docker_study$ sudo docker run abc_mentoring:abc
hello world!
with1015@ubuntu:~/Docker_study$
```

Docker image pull

- (sudo) docker pull [이미지 이름]:[버전]
- Docker hub에 등록 된 다른 사용자가 만든 이미지를 로컬로 내려받습니다.

```
with1015@whale10:~$ sudo docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
59bf1c3509f3: Pull complete
Digest: sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
with1015@whale10:~$
```

Docker file

- Docker를 사용하기 위해서는 직접 docker 명령어를 입력하는 방법도 있지만 docker file을 이용한 방법도 있습니다.
- Make의 Makefile처럼 docker는 Dockerfile이라는 이름을 가진 파일을 통해 사전에 준비한 docker 명령어를 순차적으로 실행하고 필요한 환경을 자동으로 설정할 수 있습니다.
- Dockerfile은 항상 이름이 Dockerfile 이며, 확장자는 없습니다.
- 완성된 Dockerfile은 아래와 같은 명령어로 실행이 가능합니다.
 - `docker build [option] [Dockerfile 경로]`

Docker file 작성 방법

- Dockerfile은 아래와 같은 여러 명령어들을 이용해 작성합니다.
 - FROM : 컨테이너의 기반이 되는 이미지
 - MAINTAINER : 작성자 정보
 - RUN : shell script나 명령어를 실행
 - CMD : 컨테이너가 실행되었을 때 사용할 명령어
 - LABEL : 라벨 작성 (docker inspect 명령어로 확인 가능)
 - ENV : 컨테이너 내부에서 환경변수 설정
 - ADD : 파일 / 디렉토리 추가
 - COPY : 파일 복사
 - ENTRYPOINT : 컨테이너가 시작되었을 때 딱 한 번 실행할 스크립트 실행
 - VOLUME : docker volume 마운트
 - WORKDIR : RUN, CMD, ENTRYPOINT가 실행 될 작업 디렉토리

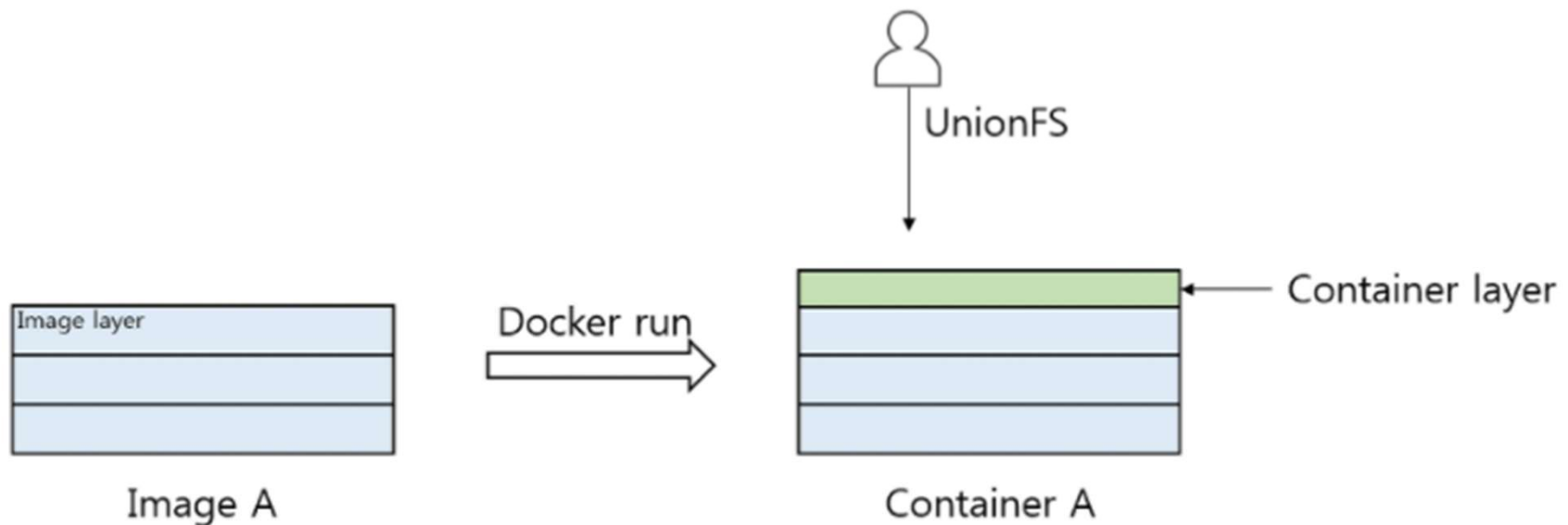
Docker file 예시

```
Dockerfile
1 FROM ubuntu:16.04
2
3 MAINTAINER Hyunjoon_Jeong "with1015@unist.ac.kr"
4
5 RUN apt-get update -y
6 RUN apt-get install -y python-pip python-dev build-essential
7
8 COPY . /app
9
10 WORKDIR /app
11
12 RUN pip install -r requirements.txt
13
14 ENTRYPOINT ["python"]
15 CMD ["hello_world.py"]
```

- 파일 저장 후 **sudo docker build --tag abc_mentoring:abc .** 입력

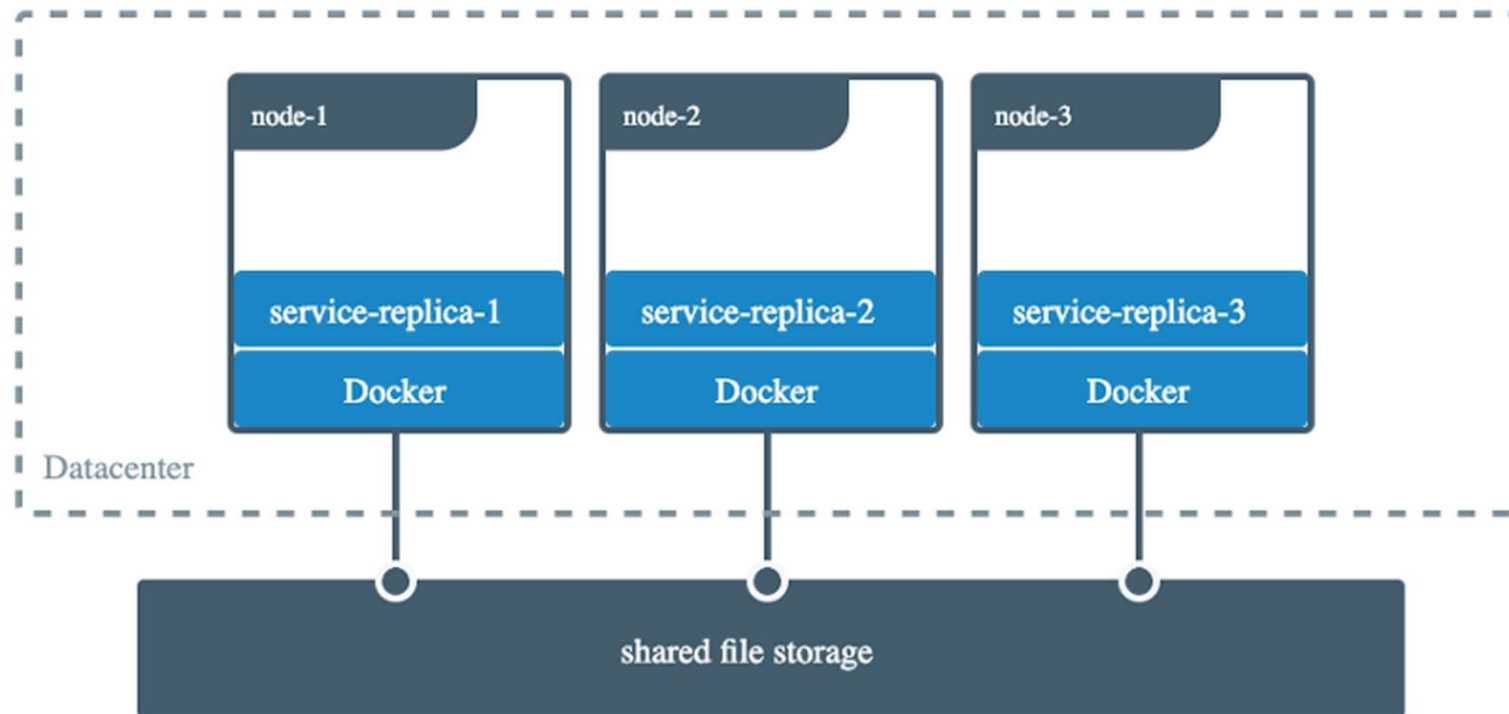
Docker volume

- Docker 컨테이너에서 작업한 내용은 Docker의 container layer 위에 저장됩니다.
- 그러나 이는 **컨테이너가 종료되면 데이터가 모두 사라지게 됩니다.**
- 컨테이너에서 작업한 내용을 저장하고 싶다면 어떻게 해야 할까요?



Docker volume

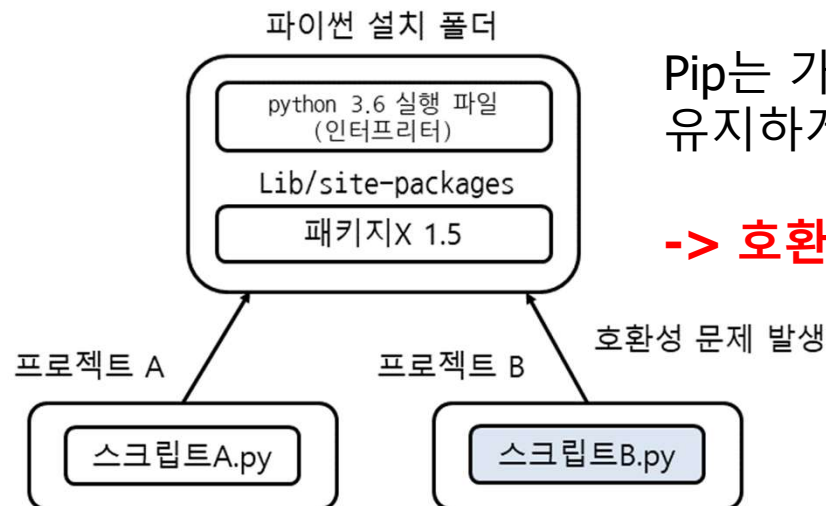
- Docker volume은 host의 폴더가 Docker container 내부에서 서로 공유 되도록 연결 시켜줍니다.
- (sudo) docker run -v [호스트 폴더]:[컨테이너 폴더] [이미지 이름]



Python 가상 환경

- Python 모듈은 Pip를 통해서 /usr/lib/python3/dist-packages (우분투 기준) 경로에 설치 됩니다.
- 만약 두 프로젝트가 서로 다른 module 버전을 요구한다면 어떻게 될까요?
 - Ex) Project A : Pytorch 1.10 버전 요구 vs Project B : Pytorch 1.6 버전 요구

글로벌 파이썬 환경

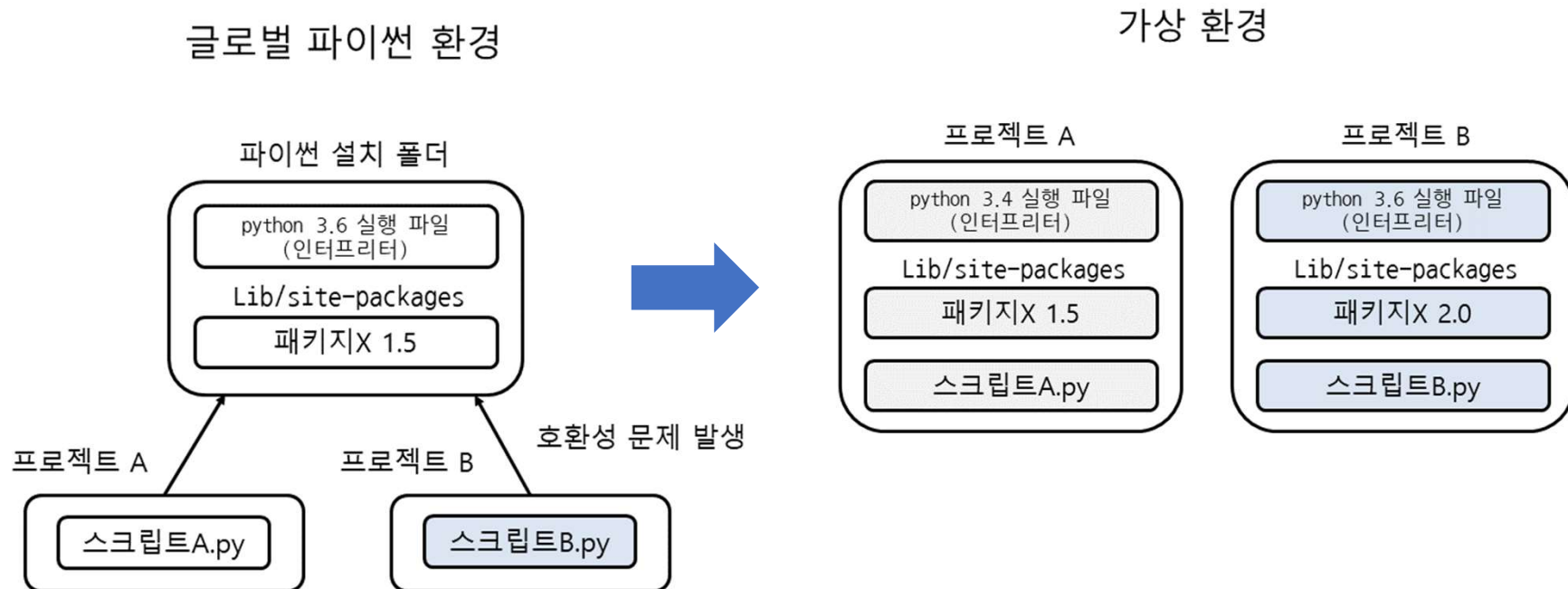


Pip는 가장 마지막에 설치 된 모듈 버전을 유지하게 됩니다.

-> 호환성 문제 발생!

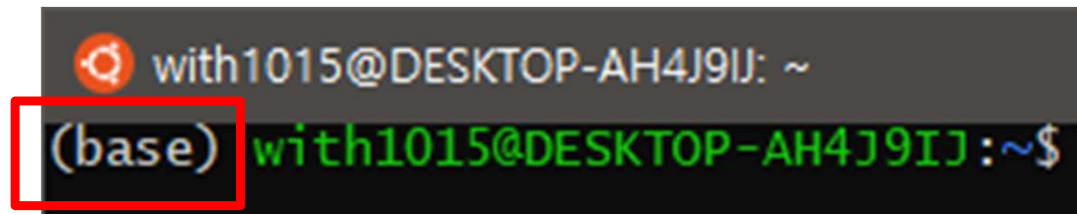
Python 가상 환경

- 따라서 두 프로젝트는 서로 다른 가상 환경을 사용하여 **각각의 프로젝트에서 필요한 모듈 버전을 따로 관리해야 합니다.**



CONDA 가상환경

- Python 가상 환경 관리 프로그램 입니다.
- Python 버전에 따른 패키지, dependency를 관리해줍니다.
- 아래 링크에서 사용 OS에 맞게 설치가 가능합니다.
- <https://www.anaconda.com/products/individual>

A terminal window screenshot showing a user named 'with1015' on a machine named 'DESKTOP-AH4J9IJ'. The prompt is '~'. Below it, the prompt has changed to '(base) with1015@DESKTOP-AH4J9IJ:~\$', indicating that the base Conda environment has been successfully activated. A red rectangle highlights the '(base)' part of the prompt.

```
with1015@DESKTOP-AH4J9IJ: ~  
(base) with1015@DESKTOP-AH4J9IJ:~$
```

성공적으로 설치된 경우, 계정 옆에 가상 환경의 이름이 출력 됩니다.

CONDA 명령어

- `conda create -n [환경 이름] python=3.X`
- `conda env list`
- `conda install [패키지 이름]`
- `conda activate [환경 이름]`
- `conda deactivate`
- `conda env remove -n [환경 이름]`

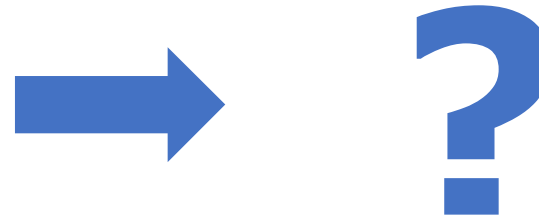
conda install vs pip install

- 간혹 일부 패키지의 경우, conda install로 설치가 되지 않을 때가 있습니다.
- conda install의 경우, anaconda에서 공식적으로 지원하는 패키지만 설치가 가능 합니다.
- pip install의 경우, pypi 서버에 등록된 패키지만 설치 가능합니다.
- 그러나 어느 것을 사용하더라도 가상환경이 activate 된 상태라면 결국 같은 역할을 하게 됩니다.
 - 즉, pip install과 conda install 둘 다 하는 역할은 같으므로, **유동적으로 사용합시다.**

requirement.txt

- 가상환경 안에 있는 패키지를 다른 환경으로 그대로 옮기려면 어떻게 해야 할까요?

```
with1015@octopus:~$ pip list
apt-xapian-index (0.45)
argparse (1.2.1)
chardet (2.0.1)
colorama (0.2.5)
configobj (4.7.2)
html5lib (0.999)
Landscape-Client (14.12)
lxml (3.3.3)
numpy (1.8.2)
PAM (0.4.2)
pip (1.5.4)
pyinotify (0.9.4)
pyOpenSSL (0.13)
pyserial (2.6)
python-apt (0.9.3.5ubuntu2)
python-debian (0.1.21-nmu2ubuntu2)
requests (2.2.1)
setuptools (3.3)
six (1.5.2)
ssh-import-id (3.21)
Twisted-Core (13.2.0)
urllib3 (1.7.1)
virtualenv (16.6.1)
wheel (0.24.0)
wsgiref (0.1.2)
zope.interface (4.0.5)
```



requirements.txt

- `pip freeze > requirements.txt`
 - 가상환경에 설치된 모든 모듈과 버전을 requirements.txt 이름으로 목록을 만들어 내보냅니다.
- `pip install -r requirements.txt`
 - requirements.txt 이름으로 된 모듈 리스트를 모두 버전에 맞게 설치합니다.

```
with1015@octopus:~$ pip freeze > requirements.txt
with1015@octopus:~$ cat re
requirements.txt resnet/          resnet50/          result/
with1015@octopus:~$ cat requirements.txt
Landscape-Client==14.12
PAM==0.4.2
Twisted-Core==13.2.0
apt-xapian-index==0.45
argparse==1.2.1
chardet==2.0.1
colorama==0.2.5
configobj==4.7.2
html5lib==0.999
lxml==3.3.3
numpy==1.8.2
pyOpenSSL==0.13
pyinotify==0.9.4
pyserial==2.6
python-apt==0.9.3.5ubuntu2
python-debian==0.1.21-nmu2ubuntu2
requests==2.2.1
six==1.5.2
ssh-import-id==3.21
urllib3==1.7.1
virtualenv==16.6.1
wheel==0.24.0
wsgiref==0.1.2
zope.interface==4.0.5
with1015@octopus:~$
```

Activity

- Python 가상 환경 파일을 만들고, Docker 컨테이너를 사용해 새로운 가상 환경으로 이식해봅시다.

Thank you

