Docker 란 무엇인가

> 생활코딩 Docker 입구 수업 - YouTube 해당 내용 참조.

1. 수업 소개

1. 개발 환경 구축이 좀 어렵더라.
    - 운영체제 위에 여러 의존성을 갖고 있는 Software 를 사용해야 하므로.
2. 다른 Computer 에 Software 가 있고, 이를 빌려오는 형태로 사용한다면 좋지 않을까?
3. 하나의 Computer 에 가상의 컴퓨터를 만들어서, 방금 전 아이디어를 구현하는 것이 가능.
    - 단, 하나의 가상의 컴퓨터를 만든다면, 그 자원이 너무 큼.
4. 그럼 하나의 컴퓨터 안에서 각각의 앱을 격리된 환경 (Container - 그 어플리케이션과, 그 어플리케이션을 실행하는데 필요한 bin, lib 파일들.) 에서 실행하고 이를 우리가 개발하는 어플리케이션이 이를 가져오는 형태로 만든다면 좋지 않을까?
    - Linux 에는 이러한 실행방법이 내장되어있음. = 해당 기술을 Container 라 부름.
    - Container 를 이용하는 걸 도와주는 Solution 중 하나가 **Docker** 임.

2. 설치

- Docker 는 Linux 를 기반으로 하고 있음.
  - 설치 시 알아서 가상머신 + 위에 Docker 도 깔아주므로, 따로 생각 안 해도 됨!

1. docker.com > docs > Guides > Windows
   - 그냥 여기 있는 거 다운 받아서 실행하면 충분함!
2. 이후, Docker 를 GUI 기반으로 조작할 수도 있고, CMD 에서, docker 기반 명령어를 입력함으로써 조절할 수도 있음!
   - 설치 확인 위해 docker images 라 쳐보자!

3. 이미지 pull

기본 개념

- App Store = Docker Hub (Registry)
  - Registry 가 뭘까? 계속 듣기만 듣고. : Microsoft Windows 의 설정 및 정보를 담고 있는 DB.
  - 일종의 Meta 적인 정보를 담고 있는 DB 라고 생각하면 됨!
- Program = Image : 다운로드 받은 실행 파일.
- Process = Container : 실행 파일의 실행.
- Docker Hub -> Image 과정 : Pull
- Image -> Container 과정 : Run

> 기본적으로 Container 는 하나의 컴퓨터 안에서 각각의 앱을 격리된 환경에 구축하는 것. Docker 는 이러한 Container 를 쉽게 사용할 수 있는 Solution 을 제공하는 것임.
>
> - 이에 따라, 아래 비유는 Docker 에서 추가적으로 제공하는 기능인 Hub, Image 등과, Container 와의 관계가 어떻게 되는지를 생각하면 될 듯.
> - 또한, Image 를 실행한다는 것은, 독립된 환경으로서 존재하는 Container 를 구축하는 것이라고 생각하자.
> - 실제로, Image 는 Class 처럼, Container 를 만들 때 사용되는 템플릿이라 보면 된다.
>   - (추가적으로 생각해봤을 때, Program 도 어찌보면 동일한 걸 띄나 싶기도 함. 결국 그냥 있을 때는 가만 있다가, 실행해서 Process 로 배정되는 경우에 cpu 나 ram 등의 자원을 얻어서 실행되는 것이니까.)
> - **결론** :
>   1. MAIN : Container / ADDITIONAL : Hub, Image
>   2. Hub 에서 다운 받은 Image 를 실행하면 Container 가 구축된다. = Image 에서 작성되어 있는 설정 등에 따라 격리된 환경 및 어플리케이션이 생성된다.
>   - Image 를 "실행"한다에서 알 수 있 듯, Container는 Process 와 같이 Run, Stop, Start, Delete 등으로 관리할 수 있다. (구축한 환경 자체를 Process 와 비슷하게 관리하는 것이 가능.)

Docker Hub에서 Image 다운로드 받는 법

1. Docker Hub 들어가기.

2. Containers 클릭.
    - Official Image 라 적혀있는 친구는 Docker Hub 에서 관리하고 있는 공인 Container 임.
3. Containers 안에는 해당 container 를 pull 하는 명령어와, 이 container 를 사용할 수 있는 방법이 작성되어 있음.

- docs.docker.com > references 를 살펴보면, 명령어들의 내용을 알 수 있음.
    - registry 를 이제 모두 다운로드 받을 수 있더라!
- docker pull 하고 난 후, 잘 pull 되었는지 확인하려면 docker images 를 입력하면 됨.
- GUI 에서도 동일하게 하는 것이 가능.

4. Container Run

Docker GUI 에서 처리

1. GUI 에서 그냥 Run 눌러서 만들 수 있음.
    - 이때, Container 이름, Port, Volumes, Container Path 등이 표시.
        - 하나의 Image 가 여러 Container 를 포함할 수 있으므로, 이름 잘 지정해두는 것이 좋음!
2. Containers tab 에서 실행되고 있는 container 를 클릭하면 실행되고 있는 log 를 확인하는 것이 가능함.
3. Inspect (자세한 정보), Stats (현재 사용 자원), 중지, 삭제 등 모두 GUI Tool 에서 관리해줄 수 있음.

Command Line 에서 처리

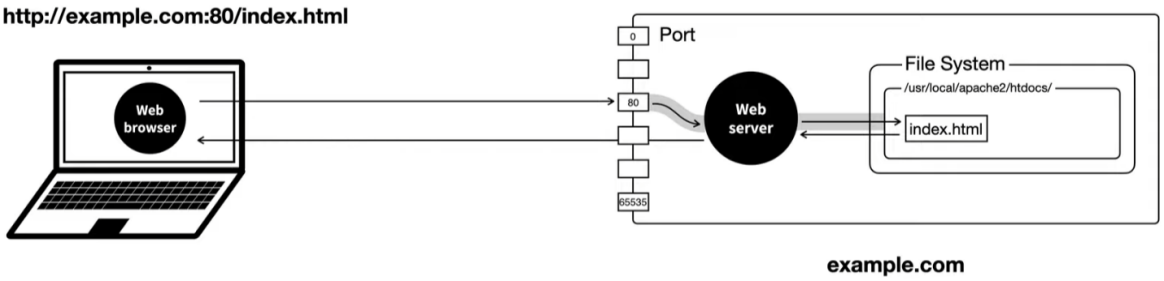
1. docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
    - OPTIONS : IMAGE 실행에 있어 고려하여야 하는 Option 을 설정하는 것이 가능.
    - COMMAND : Container 실행 시, 실행하고 싶은 command 를 해당 부분에다가 입력.
2. ex - docker run httpd
3. docker ps : 현재 실행 중인 container 를 보고 싶을 때.
4. ex - docker run --name ws2 httpd
5. docker stop NAME/CONTAINERID : container 정지.
6. docker ps -a : 정지중인 container 포함 표시.
7. docker start STOPCONTAINERNAME/ID
8. docker logs NAME/CONTAINERID : log 표시
    - docker logs -f NAME/CONTAINERID : 실시간 log 출력.
    - Ctrl + C 누르면 실시간 표시 취소. (이거 exit 인데, 이런 명령어들 확실히 Linux 기반인 듯!)
9. docker rm NAME/CONTAINERID : container 삭제.
    - 정지 중인 container 에 대해서만 동작.
    - docker rm --force 를 통해 실행중인 친구도 삭제 가능.
10. docker rmi IMAGENAME : image 삭제.

5. Network

- Docker 로 동작하는 많은 Image 들이 Network 위에서 동작. 따라서 Network 를 학습해둘 필요가 있음.
    - Web app 을 만드는 기본적인 방법인 WEB1 / HOME Server 를 공부하는 것 추천.

일반적인 경우

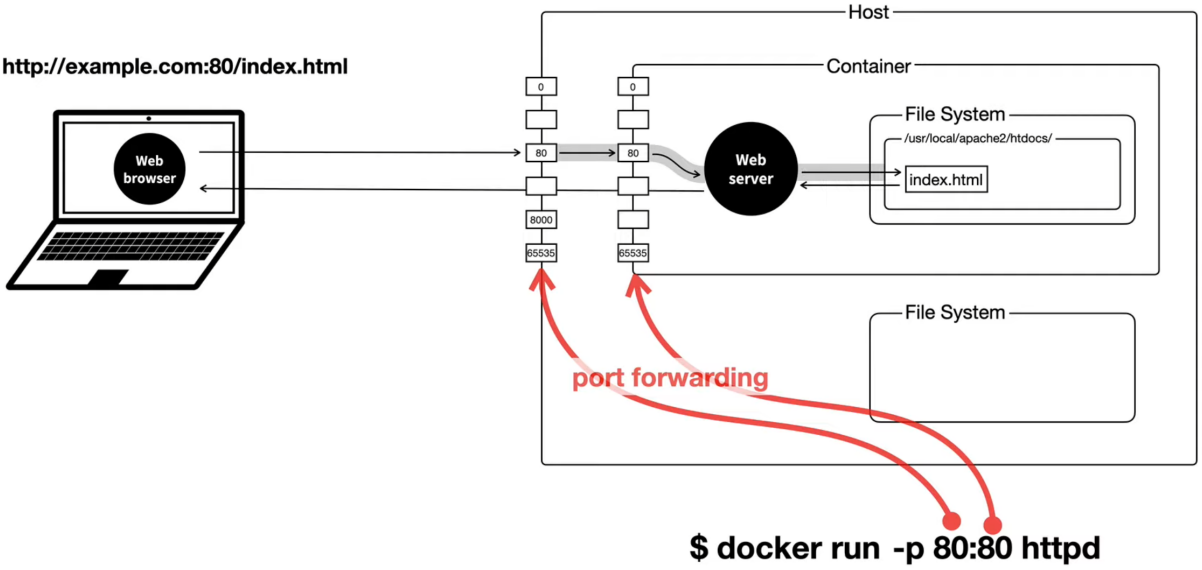
- 일단 여기서 개괄적으로 설명해줌.



1. Web Browser / Web Server 프로그램이 깔린 두 Computer 가 존재한다 하자.
    - Web Server 에는 File System 내에, 웹페이지 표시 등에 관한 "정보"가 존재한다.
    - Web Browser 는 정보가 제시되었을 때, 해당 정보를 표시해주는 역할을 할 수 있다.
    - Computer 는 주소 (IP Address 혹은 Domain) 로서 식별 가능하다.
    - Computer 내의 프로그램은 각 프로그램에 할당된 Port Number (Computer 가 Network Connection 을 위해 갖고 있는 연결점) 을 통해 접근하는 것이 가능하다.
2. Web Browser 는 Computer의 주소와 Port Number 를 통해, Web Server Program 에 접근한다. 접근 후 Web Server 는 정보를 반환하며, Web Browser 는 해당 정보를 표시한다.

Container 사용

1. Image 를 실행하면, Container 구축 시, 해당 Container 에 할당한 Application 인 Web Server 가 설치될 것.
    - 이 Container 가 존재하는 환경을 Host 라 함. (Docker 자체가 하나의 컴퓨터 안에서 각각의 앱을 격리된 환경 (Container) 에서 구현해주는 친구고. 이건 Linux 운영체제 등 기존 존재하는 운영체제에서만 가능하기 때문에.)
    - Docker 가 마련한 Linux 운영체제가 존재하는 가상환경 / Container 에서 독립적으로 마련된 환경 또한 사실 각각 존재하기 때문에, 개별적인 File System 및 Port Number 가 별도로 존재함.
        - *Q. 내가 모르고 있었던 사실이, Container가 "독립된 환경" 이라고 인지하고 있긴 했지만, Port 나 File System 또한 독립적이라는 사실을 인지하지 못하고 있었던 거였네! => A. "1. 수업 소개" 를 살펴보면, Container 가 처음 만들어진 이유가 "밖에 존재하는 Computer 에 Software 를 마련해서 빌려오는 형태로 제작하면 좋겠다 => Computer 안에 가상 환경 구축해서 넣어볼까? => Container 라는 개념을 따로 만들어서, 이걸로 활용하자!" 였기 때문에, Container 만도 Port 및 File System 이 존재하는 것은 자명하다고 볼 수 있겠다.* Container의 목적 자체가, 원래 하나의 컴퓨터에서 가상환경으로 마련할 수 있었던 독립적인 환경을, 더 가볍고 쉽게 만들 수 있도록 해주는 것니까!
    - 이 Docker 가 마련한 Container 가 존재하는 Linux 운영체제를 Host 라 한다.
2. Host 와 Container 가 따로 존재하기 때문에, 따로 Port 가 존재한다. Host 의 Port 를 Container 의 Port 로 연결하여, Container 내부에 존재하는 Application 을 사용해 주어야 하는데, 이와 같이 Port 와 Port 를 연결하는 과정을 Port Fowarding 이라 한다.
- Port Fowarding 예시



3. Port 연결 방법
    1. GUI Tool 에서 Image 를 실행할 때, 해당 Container의 Application Port 가 Host 의 어떤 Port 와 연결될지를 사용해줄 수 있다!
        - 일반적으로 Container 내에서 해당 Application 에 접속할 때 사용하는 Port 의 경우, Image 자체적으로 정해져 있다.
    2. Command Line 에서는 run의 option 부분에서 Container의 Application Port 가 Host 의 어떤 Port 와 연결될지를 설정해줄 수 있다.
        - docker run --name ws3 -p 8081:80 httpd
        - = httpd Image 를 ws3 라는 이름으로 Container 를 생성하고, 8081 Host 와 80 Port 를 연결해줘라.

6. 명령어 실행

- 위 과정을 거치는 경우, web app 의 초기 구성은 끝남. 하지만, Container 안에 들어있는 파일을 바꿔야, 내가 원하는대로 Customizing 할 수 있음.
  - 이를 위해선 Container 안에 들어가서 File System 내의 파일을 바꿀 수 있어야 함.
- Container 내부에 들어가고 싶다!
  1. GUI Tool 에서 CLI 버튼을 찾아 들어감.
     - 이 CLI 는 Container 내부의 내용을 띄워 줌!
  2. CMD 에서는 docker exec NAME/CONTAINERID COMMAND [ARG...]
     - docker exec 치고, 이름 입력 후, 해당 Container 내부에서 실행하고 싶은 명령어를 입력하여 Container 내부에서 명령어를 실행할 수 있음.
     - => 이 방법은 지속적으로 입력을 해줄 때 번거로움.
  3. docker exec -it ws3 /bin/sh (3. Bash Script : Shell 은 Linux 기능을 사용하기 위해 접근하는 겉 껍질, 인터페이스 프로그램.)
     - 이를 사용할 경우, cmd 가 해당 shell 을 기반으로 둘러쌓이게 된다.
     - 이후 사용 부터는 Container 내부에서 실행하고 싶은 명령어를 입력한 것과 동일하게 취급 받게 된다.
     - -it 의 입력 이유 : t는 tty 를 할당. i는 stdin 에 대한 연결을 계속 유지. 이를 안하면, Container shell 과의 연결이 바로 끊어지게 됨.
     - exit 나가면 됨!
  - docker exec -it NAME/CONTAINERID /bin/bash : 이걸로 bash 있는지 살펴보고 (bash 가 좀 더 편리하기 때문에) 만약 없다면, bone shell (그냥 sh) 쓰면 됨!
- httpd mannual 내에 어디에 index.html 파일이 있는지 나와 있기 때문에, 해당 장소에 접근해서 수정하면 됨.
  - vim 이 현재 설치되어 있지 않은 상태이기 때문에, 이를 설치할 필요 있음.
  - apt update -> apt install vim 통해서, vim 설치 후, 해당 파일 수정하면 좋음.
  - vim 사용법은 이전 배웠던 0. VIM 기초 사용 방법 인지해서 하기.

7. Host 와 Container 의 File System 연결

- 위와 같이 Container 내부 파일을 직접 수정하는 경우 문제가 발생할 가능성이 높음.
  - 이유 : Container 등이 삭제되면, 작업한 내용이 사라짐.
  - 해법 : Host 의 파일 수정이, Container 의 파일 시스템에 반영되도록 만들기!
    - 즉, 실행 환경은 Container / 파일 수정은 Host 진행으로 독립.
- docker run -p 8888:80 -v ~/Desktop/htdocs:/usr/local/apache2/htdocs/ httpd
- = Host 의 volumn (저장장소) 를 이용하여, Docker Container 를 구성.

> 처음 Docker 에 대한 이해는 다음과 같았음.
>
> - Windows 등 기타 운영체제 -> Docker 가 알아서 구축한 Linux 가상머신 -> Linux 가상머신 내 Container
>   - 하지만 이 경우, Linux 에서는 알아서 가상 머신 구축하거나 할 필요가 아무래도 없다 보니까, 차이가 많이 발생하게 됨. (Host 에 대한 접근이 Windows 는 두 번 해야 하는데 비해, Linux 는 한번만 해도 되므로.)
> - 이게 의문이 들어, 확인 해보니 Windows 도 동작이 동일함. 아마 Docker 자체에서 알아서 Linux 가상머신 등 경유하여 처리하는 듯함.
> - 즉, Windows 등으로 구성이 되어있다 하더라도, Host 의 Port 나 File System 을 거기서 알아서 하고. Docker 가 알아서 처리되도록 해주기만 하면 됨!

8. 수업을 마치며

- **Docker 로 만들어져 있는 수많은 Software 들을 명령어 한 줄로 만들 수 있겠더라!!**

추가 학습 : 이미지 만드는 법 - Dockerfile & build

> 요약 : **dockerfile 은 docker image 를 만드는 meta file임.** dockerfile 등이 들어있는 directory 를 build 하여, Image 를 만들 수 있음.

- 나도 Image 를 만들고 싶다! => 두 가지 방법이 있음.
  1. 바뀐 Container 를 "**commit**" 하여 Image 를 변경
     - 변경된 이미지를 백업해주는 느낌.
  2. Dockerfile : 기반 Image, RUN, WORKDIR, COPY, EXPOSE and Endpoint 제작에 대한 형식을 만들어, **Build** 명령. 이를 하면, 새로운 Image 가 만들어짐.
     - 우리가 만들고 싶은 Image 를 생성하는 느낌.
- 이미지에는 Web Server 가 존재하여, 사용할 경우, Web Server 를 바로 구동 가능 하도록 만들고 싶다.

Image 만들기

commit 방법

- docker run --name web-server - it ubuntu:20.04
  - ubuntu 20.04 image 가 다운되고, 이게 바로 container 가 만들어짐.
- docker commit web-server web-server-commit
  - web-server container 의 image 인 web-server-commit 이 만들어짐.

Dockerfile & Build

- FROM ubuntu:20.04 #가져올, 기반이 될 image 를 지정해주는 것.
- docker build -t web-server-build(이름 : Tag 라 부름.) "Image만들 때 사용할 DockerFile 의 Path 위치 - 보통 .(current directory) 해서 하면 됨."
- 이렇게 하면 Image 가 만들어짐.

Container 내에 Python3 깔기

- 위 docker image 파일 만들어 둔 것에서, ubuntu:20.04의 web server 를 사용해 주기 위해서 python 을 instance 에 설치할 것임.

Commit : Container 에서 처리하기

1. CLI 혹은 Command line 을 기반으로 해서, 실행되고 있는 Container 내부에 들어간 후.
2. apt update (apt : ubuntu 에서 app store 역할)
3. apt install python3

- 입력해서 python 설치함.
- 이렇게 하고 다시 commit 해야, 해당 container 의 변경 내용 update 됨.

Dockerfile 이용 : Build 될 때 자동으로 처리하기

- Docker 파일 안에서 운영체제의 명령어 실행할 때는 RUN 을 이용함.
- RUN 실행 될 때마다 Layer 가 생성되기 때문에, 웬만하면 하나에 몰아 쓰는 것이 좋음.
- RUN apt update && apt install -y python3
    - -y 는 설치할 것인지 묻는 때에 바로 승인하는 것.

사용할 Directory, File 만들기

Commit : Container 에서 처리하기

- mkdir -p /var/www/html
  - 경로 전체 생성.
- file 만들기 : touch, echo 통해서 file 만들고, redirection 하거나 해서 내용물 넣어주면 됨. 아니면, vim 써서 넣어줘도 되고!

Dockerfile 이용 : Build 될 때 자동으로 처리하기

- WORKDIR /var/www/html
  - Build 될 때 WORKDIR 로서 만들어 짐.
- "RUN echo "Hello, ~" > index.html" 처럼, 동일한 file 만들고 내용물 넣어주는 명령을 RUN 과 함께 작성해주면 됨.
- 혹은, Dockerfile 이 존재하는 directory 에 만들어줬던 file 을 복사하여, Container 에 넣어줄 수도 있음!
  1. Dockerfile 이 존재하는 directory 에 "index.html" 파일 만듬.
  2. COPY ["index.html", "."]
  - 이는 Host Directory 에 존재하는 "index.html"을, "."(working directory = 위에 WORKDIR 해줬으면 만들어짐.) 에 복사하라는 명령임.

Python Web Server 실행

Commit : Container 에서 처리하기

- python3 -m http.server

Dockerfile 이용 : Build 될 때 자동으로 처리하기

- Dockerfile 에서는, 우리가 Container 를 생성할 때, 즉시 위 명령어가 실행되길 원함.
- CMD ["python3", "-u", "-m" , "http.server"]

Build 된 Image 를 Run 해보는 실습

- 직접 실행하는 것 등 보여줌.
- **RUN과 CMD 의 차이**
  - RUN : Build 시점 (Image 가 제작될 때) 에 실행되는 명령어 (Host 반영)
  - CMD : Image 실행 시점 (Container 구축될 때) 실행되는 명령어 (Container 반영)
    - run 해줄 때, CMD 항목에 내용 작성하면, Overriding 되어서 실행 안 됨.