Docker 101

What is Docker ?

- 서비스 운영 환경을 묶어서 손쉽게 배포하고 실행하는 **경량 컨테이너 기술**
- 하나의 어플리케이션을 **격리된 공간(컨테이너)** 에서 독립적으로 실행할 수 있도록 해줌.

도커 장점

- 복잡한 리눅스 어플리케이션을 컨테이너로 묶어서 실행가능.
- 개발, 테스트, 서비스 환경을 효율적으로 관리 가능.
- 이미지를 여러 사람들과 공유 가능

이미지에 대한 이야기는 뒤에...

도커 특징

- No Guest OS
    - 호스트와 OS 자원 공유
    - 이미지에 서버 운영을 위한 프로그램과 라이브러리만 격리해서 설치하여 이미지 용량이 크게 줄어듬.
- 이미지 생성과 배포에 특화
    - 이미지 버전 관리. 중앙 저장소에 이미지를 push & pull
- No 하드웨어 가상화 계층

도커 이미지와 컨테이너

이미지

- 서비스 운용에 필요한 서브 프로그램(소스 코드, 컴파일된 실행파일을 묶은 형태)
- immutable 한 저장 매체이지만 이 이미지 위에 이것저것 붙여서 새로운 이미지를 만들 수 있음.
- Docker 에서 실제로 실행되는 건 이미지가 아니라 이미지를 기반으로 생성된 컨테이너
- 특정 프로세스를 실행하기 위한 환경

`pull` : 도커 이미지를 받음
`commit` : 파생된 이미지를 만듬
`rmi` : 이미지 삭제

컨테이너

- 이미지를 실행한 형태
- 하나의 이미지로 여러 개의 컨테이너 생성 가능.
- 컨테이너의 기본적인 역할은 미리 규정된 명령어를 실행하는 일.
- **컨테이너 = 프로세스**

`run` : 도커 컨테이너 실행
`restart` : 실행이 종료되었을 때 다시 실행.
`stop` : 실행을 강제로 종료
`rm` : 종료된 컨테이너 삭제

기본 명령어

- 컨테이너 실행 `run`
- 컨테이너 목록 확인 `ps`
- 컨테이너 중지 `stop`
- 컨테이너 제거 `rm`
- 컨테이너 로그보기 `logs`
- 이미지 목록 확인 `images`
- 이미지 다운로드 `pull`
- 이미지 삭제 `rmi`

Docker 설치하고 직접 사용해보자 !!

설치하고 나서 `$docker version` 으로 확인

run command

`docker run ubuntu:16.04`

`run` 명령어를 사용하면 이미지가 저장되어 있는지 확인하여 없으면 `pull` 한 후 컨테이너를 `create` 하고 `start` 한다

컨테이너가 정상적으로 실행됐지만 생성되자마자 바로 종료된다. why? 뭘 하라고 명령어를 전달하지 않았기 때문.

`docker run --rm -it ubuntu:16.04 /bin/sh`

컨테이너 내부에 들어가기 위해 `/bin/sh` 실행.

`-it` : 키보드 입력을 위해 사용.
`--rm` : 프로세스가 종료되면 컨테이너가 자동으로 삭제되도록 하기 위함.

ps command

실행중인 컨테이너 목록을 확인하는 명령어

`docker ps`

중지된 컨테이너도 확인하고 싶으면

`docker ps -a`

stop command

실행중인 하나 또는 여러개의 컨테이너를 중지하는 명령어.

```
docker stop [OPTIONS] CONTAINER [CONTAINER...]
```

rm command

종료된 하나 또는 여러개의 컨테이너를 완전히 제거하는 명령어

```
docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

logs command

컨테이너가 잘 동작하고 있는지 로그를 확인하여 파악할 수 있다.

```
docker logs [OPTIONS] CONTAINER
```

images command

다운로드한 이미지 목록을 확인하는 명령어.

```
docker images [OPTIONS] [REPOSITORY[:TAG]]
```

pull command

이미지를 다운로드하는 명령어

```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
```