미래산업과 기술동향

<도커>

에스이랩 박연구 부장

2023-07-28

수업소개

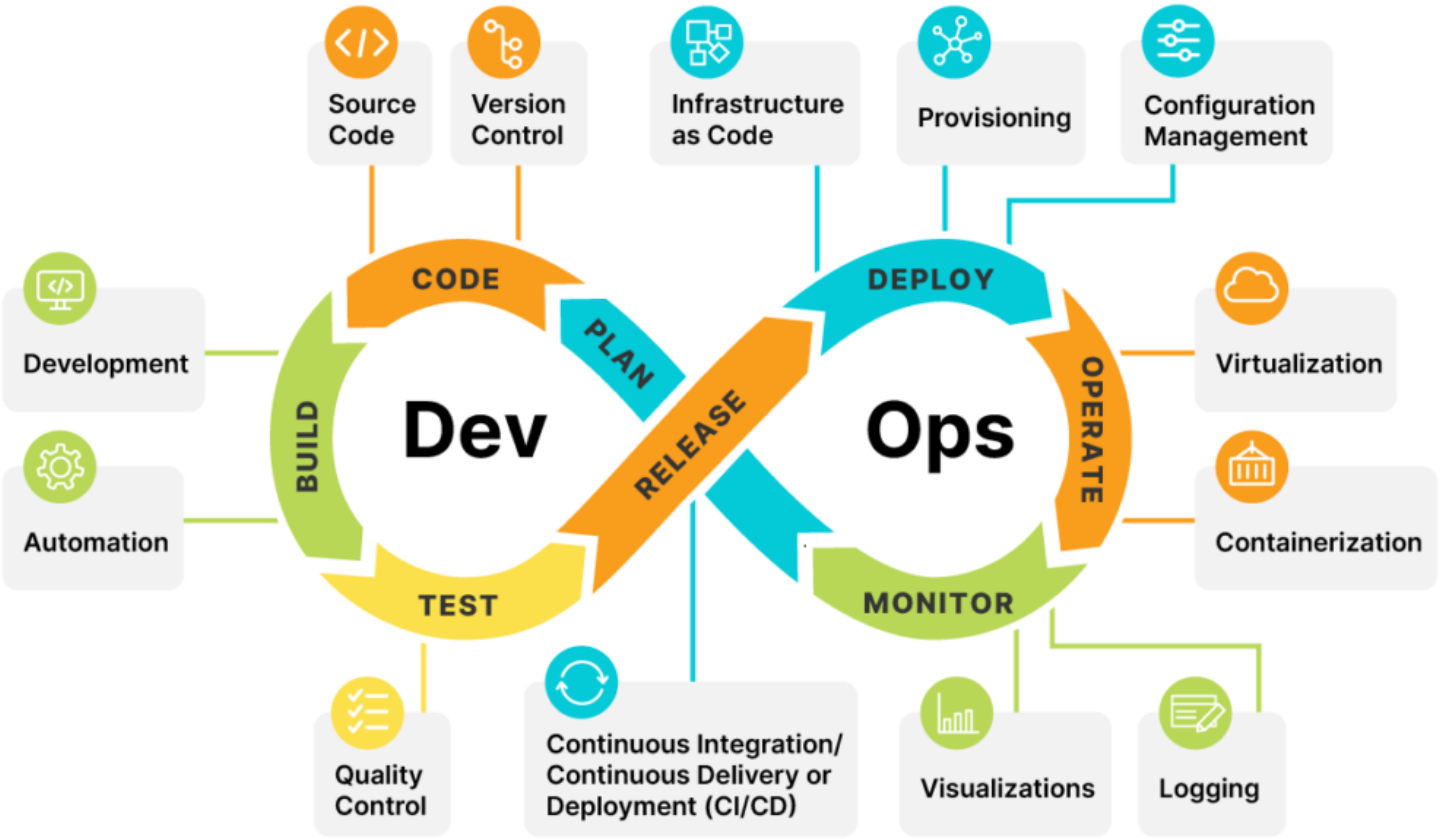
- **클라우드의 SPI 모델**
  - SaaS (Software as a Service)
  - PaaS (Platform as a Service)
  - IaaS (Infrastructure as a Service)
  - On-Premise

- **클라우드 네이티브 개념과 핵심 기술**
  - CI/CD
  - DevOps
  - Microservices
  - Containers

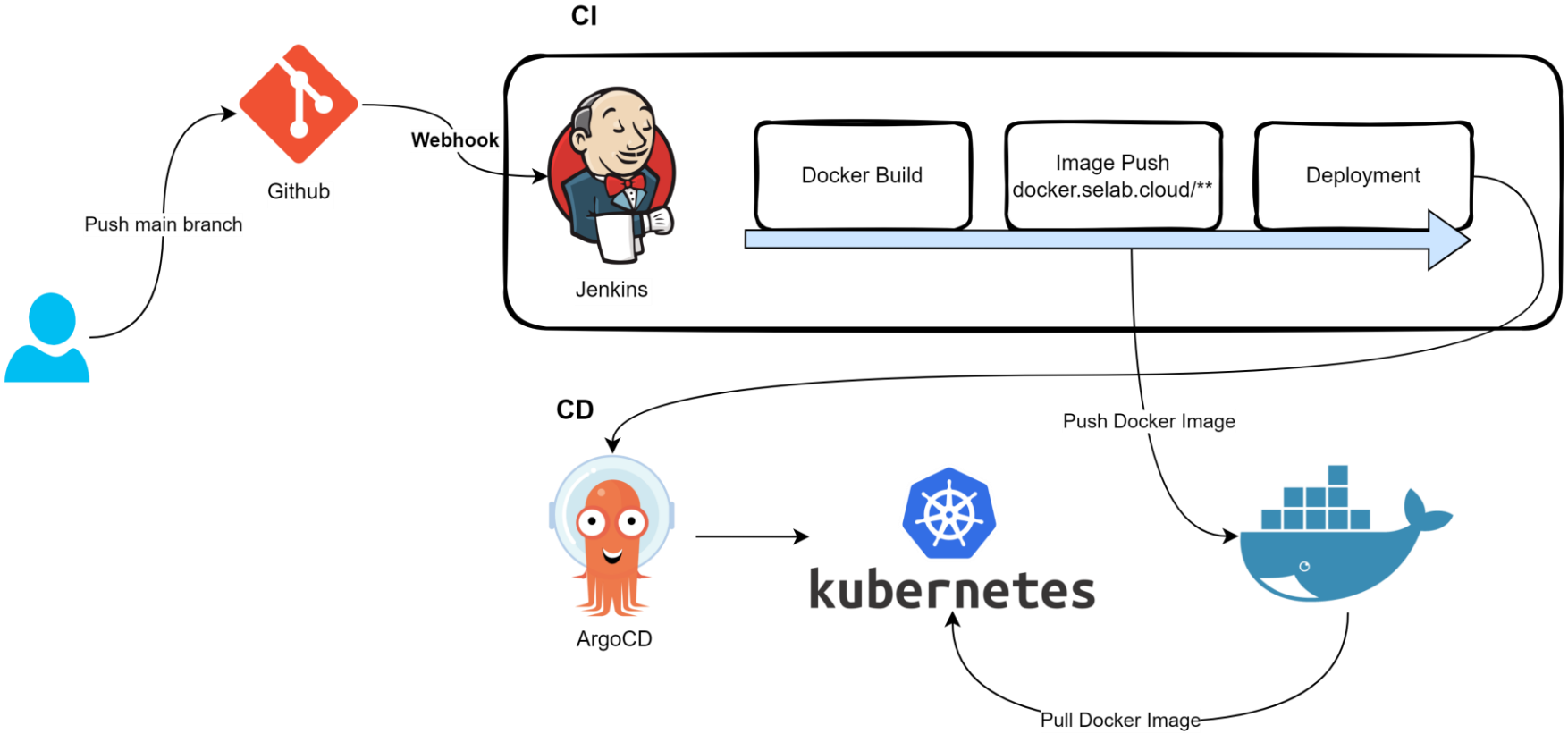
- **VCS (Version Controll System)**
  - Git

- **컨테이너의 이해와 실습**
  - Container vs VM
  - Docker

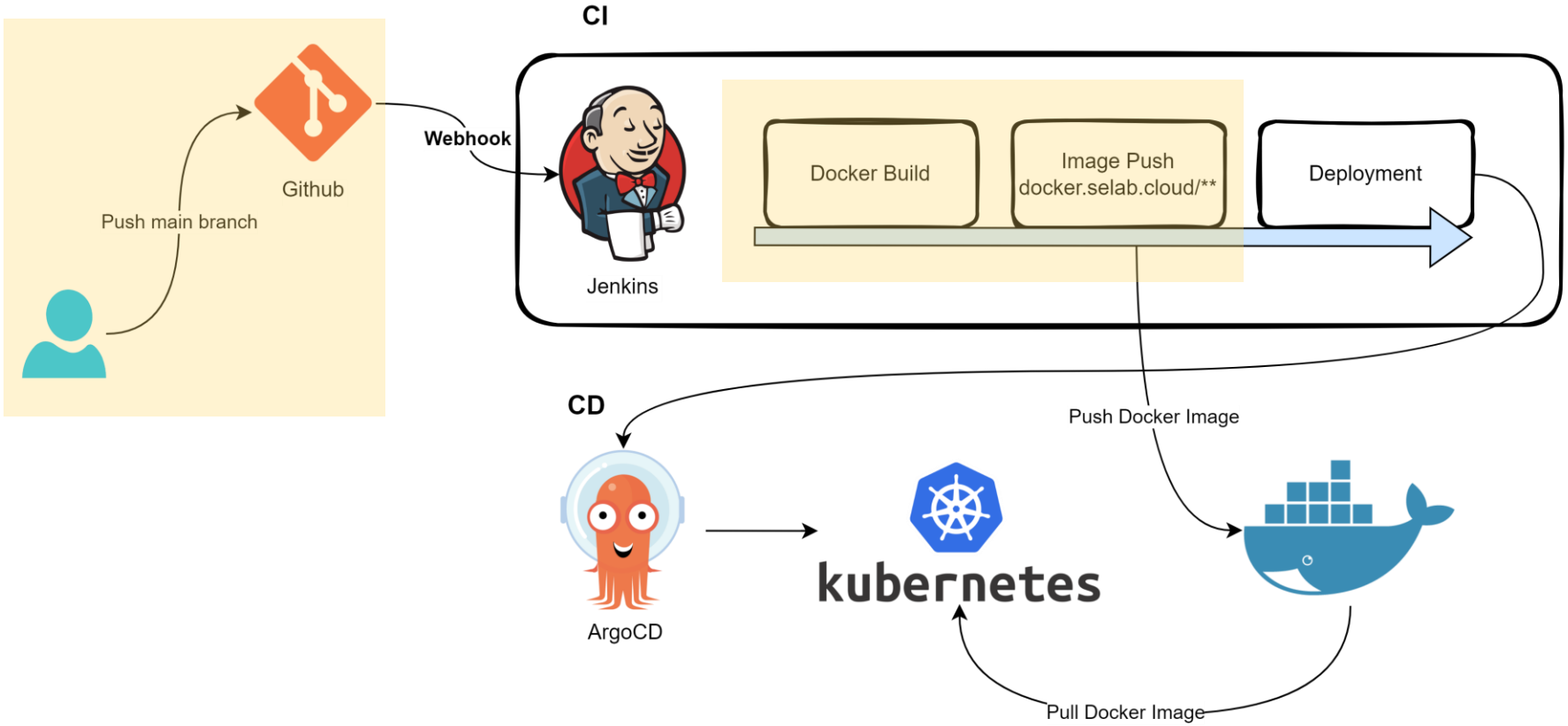
DevOps


Source Code
Version Control
Infrastructure as Code
Provisioning
Configuration Management
Development
Automation
CODE
PLAN
BUILD
Dev
RELEASE
DEPLOY
OPERATE
Ops
TEST
MONITOR
Virtualization
Containerization
Quality Control
Continuous Integration/ Continuous Delivery or Deployment (CI/CD)
Visualizations
Logging

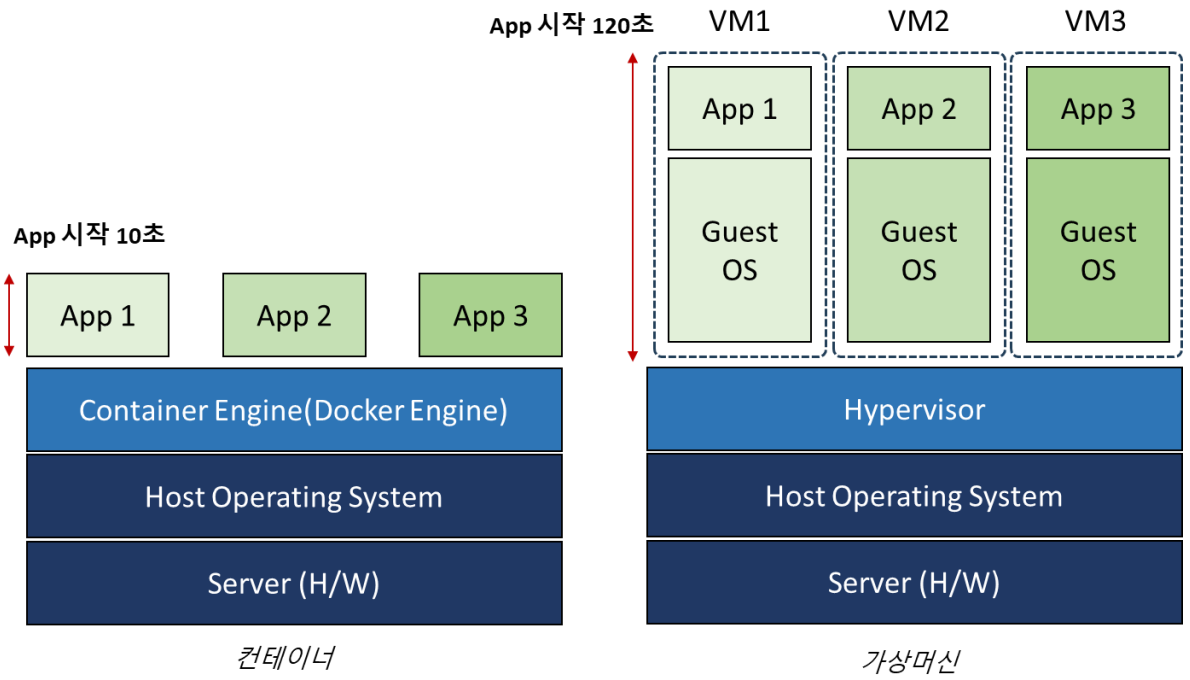
컨테이너 환경에서의 CI/CD



컨테이너 환경에서의 CI/CD



컨테이너 vs 가상머신



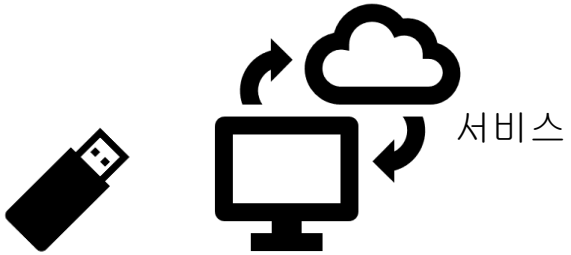
컨테이너

- 빠르게 자주 변경하고 배포하는 모든 애플리케이션
- MSA 애플리케이션
- 클라우드 네이티브 애플리케이션 빌드
- 컨테이너 이미지를 그대로 사용해서 이식성 좋음
- 가상머신에 비해 작은 이미지

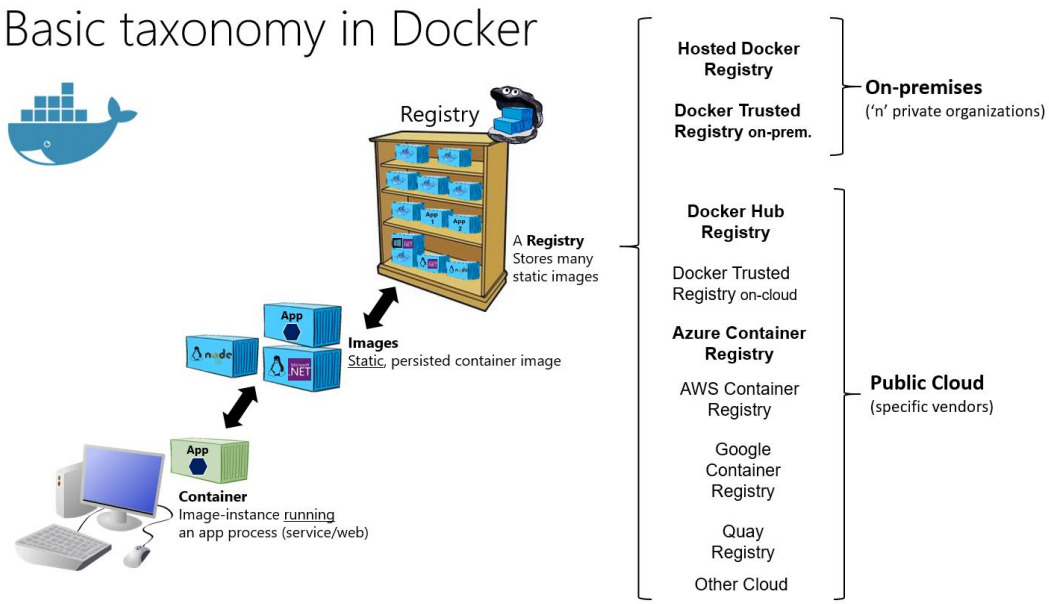
가상머신

- 기존, 레거시 및 모놀로식 워크로드
- 개발 사이클 분리가 어려운 경우
- 인프라 리소스 프로비저닝
- 다른 OS에서 또 다른 OS 실행 (리눅스에서 윈도우 실행 등)

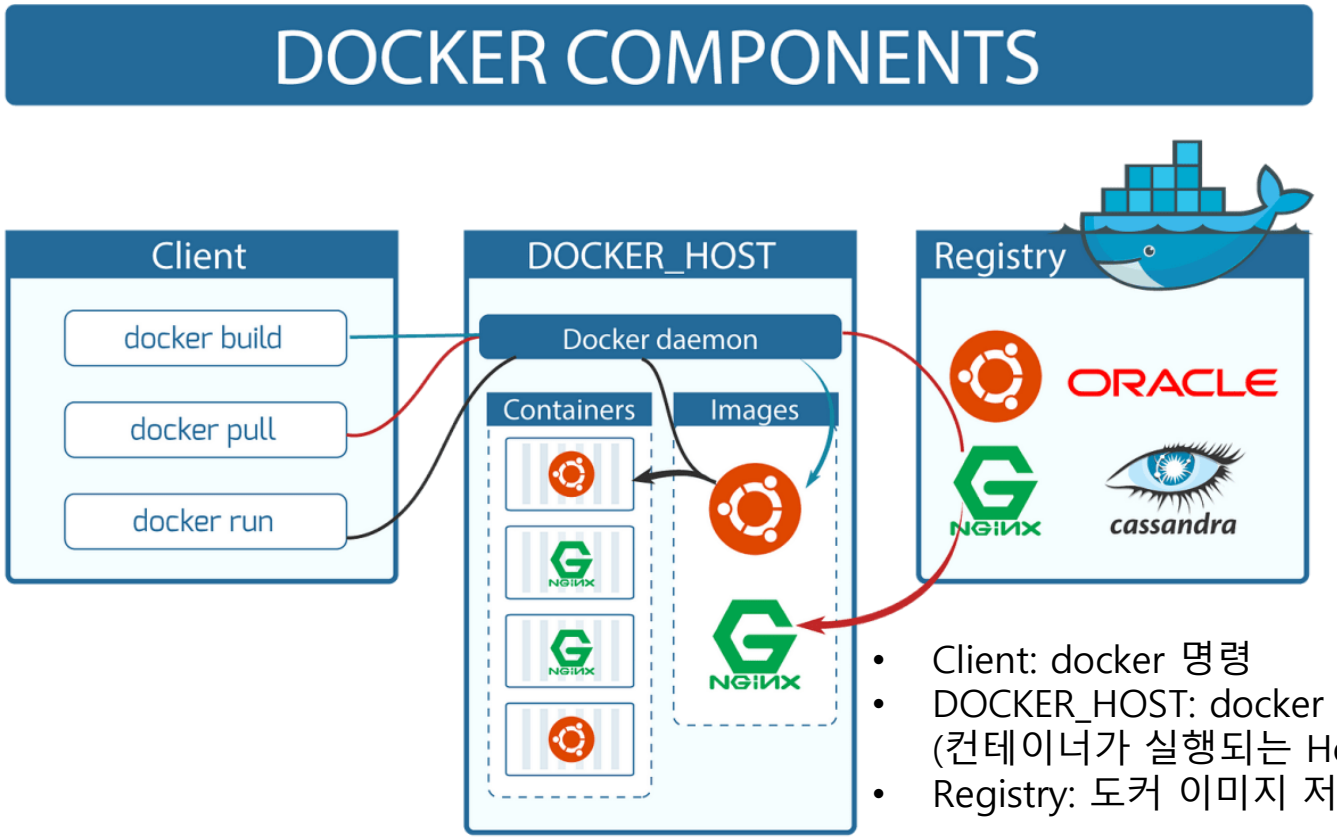
컨테이너 환경으로 변화



- 사람이 한땀 한땀 설치
- 설치 버전, 의존성 등 여러 이슈 발생
- 확장에 유연하지 않음
- 자원을 최대한 효율적으로 활용할 수 없음
- 자원을 한정적으로 사용



Docker



- Client: docker 명령
- DOCKER_HOST: docker engine이 실행되는 공간 (컨테이너가 실행되는 HostOS)
- Registry: 도커 이미지 저장소 (DockerHub)

Docker Desktop ≠ Docker (engine)

Docker Desktop

Docker Desktop is a one-click-install application for your Mac, Linux, or Windows environment that enables you to build and share containerized applications and microservices.

It provides a straightforward GUI (Graphical User Interface) that lets you manage your containers, applications, and images directly from your machine. Docker Desktop can be used either on it's own or as a complementary tool to the CLI.

Docker Desktop reduces the time spent on complex setups so you can focus on writing code. It takes care of port mappings, file system concerns, and other default settings, and is regularly updated with bug fixes and security updates.

What's included in Docker Desktop? | What are the key features of Docker Desktop?

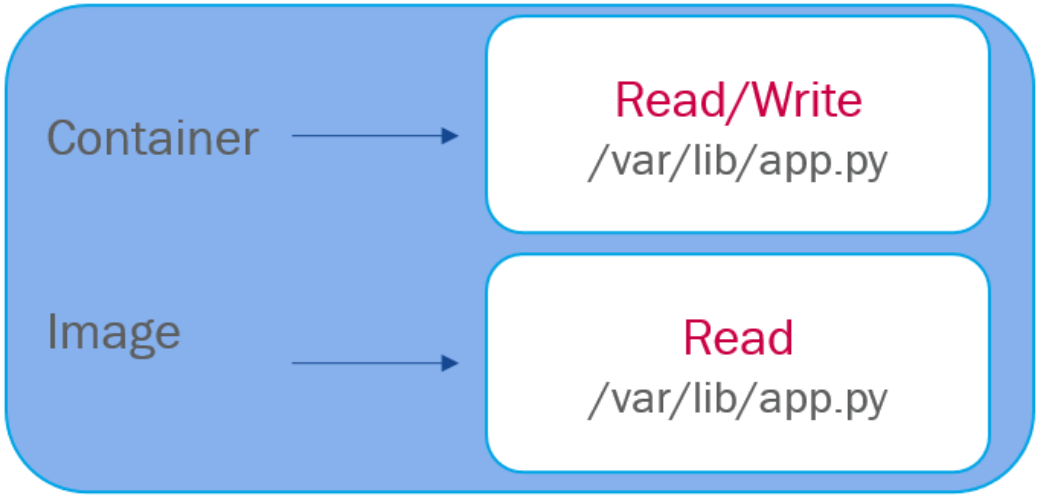
- Docker Engine
- Docker CLI client
- Docker Buildx
- Extensions
- Docker Compose
- Docker Content Trust
- Kubernetes
- Credential Helper

Docker Desktop | Docker Documentation

Docker 명령어

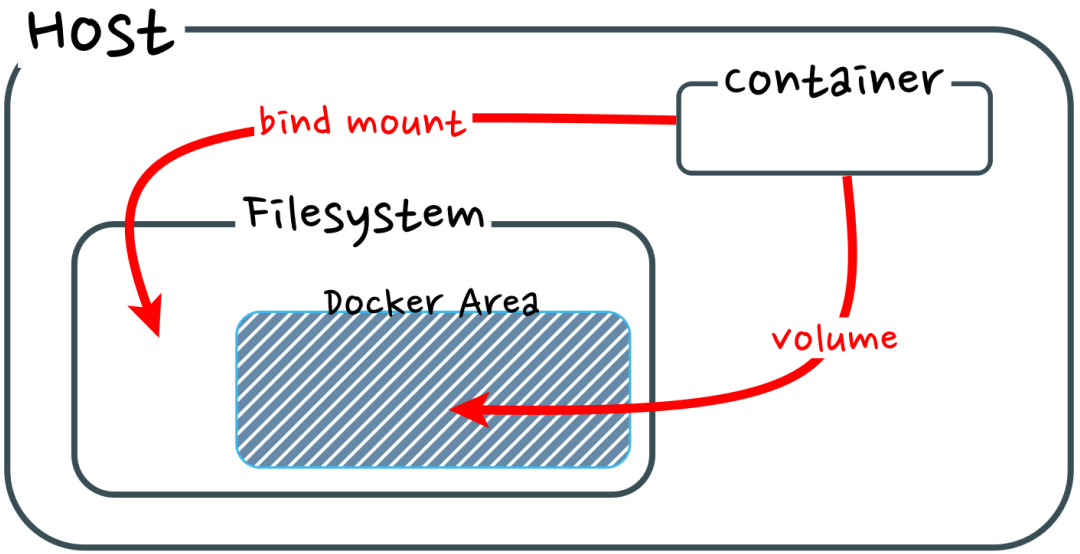
- docker build –t <image_name> <dockerfile_path> → Build docker image
- docker run –d –name=<container_name> <image_name> → Run image as container
- docker logs <container_id> → Check container logs
- docker rm –f <container_id> → Delete a container
- docker images → Check existing images
- docker pull <fully_qualitfied_image_name> → Pull images from hub
- docker push <fully_qualitfied_image_name> → Push images to hub
- docker exec –it <container_id> → Get inside a container

Docker Storage



- 이미지 내의 모든 데이터는 읽기 전용 (stateless)
- 컨테이너가 삭제되면 컨테이너 내의 모든 데이터도 삭제됨
- 볼륨 마운팅을 통해 어떻게든 업데이트/저장할 수 있음
- 명령 : docker run -v data_voume:/var/lib/ <mysql_image>
  이렇게 하면 컨테이너 데이터가 호스트의 data_volume 폴더 저장

Docker mounts



- 컨테이너가 중지된 이후에도 파일을 유지시키기 위한 방법 (stateful)
- Volume 과 bind mount
- Volume: docker engine이 관리, 호스트OS의 핵심 기능과 분리됨
- Bind mount: 호스트OS 어느 곳에서나 저장할 수 있음. 호스트OS의 특정 파일을 마운트 할 수 있음. 도커가 관리하지 않음

Dockerfile, 이미지, 컨테이너

THE MATRIX FROM HELL

UI	?	?	?
API	?	?	?
Database	?	?	?
Queue	?	?	?
Cache	?	?	?
	Dev	Test	Production

여러 종속성(Dependency)이 있는 여러 환경에서 애플리케이션을 관리해야 하는 문제

- Nodejs 버전
- JDK 버전
- Tomcat 버전
- ...

- Docker파일을 빌드하면 Docker 이미지가 생성
- 이미지는 어디든 배포할 수 있음
- 필요한 매개변수 및 환경 변수 등과 함께 이미지를 실행하면 컨테이너(프로세스)가 됨

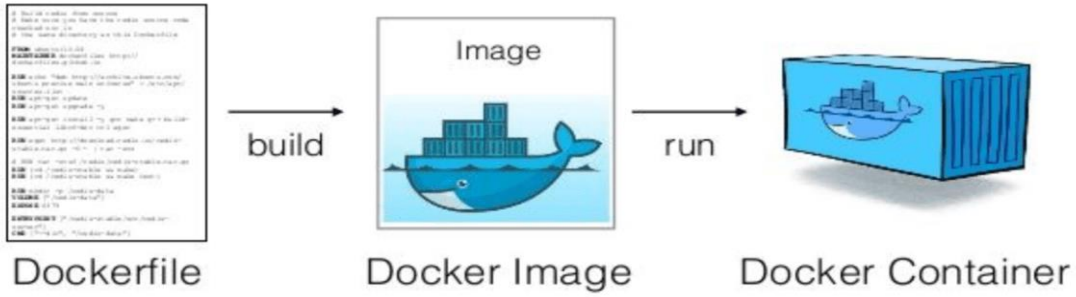
Dockerfile, 이미지, 컨테이너

THE MATRIX FROM HELL

UI	?	?	?
API	?	?	?
Database	?	?	?
Queue	?	?	?
Cache	?	?	?
	Dev	Test	Production

여러 종속성(Dependency)이 있는 여러 환경에서 애플리케이션을 관리해야 하는 문제

- Nodejs 버전
- JDK 버전
- Tomcat 버전
- ...



- Docker파일을 빌드하면 Docker 이미지가 생성
- 이미지는 어디든 배포할 수 있음
- 필요한 매개변수 및 환경 변수 등과 함께 이미지를 실행하면 컨테이너(프로세스)가 됨

Docker Registry

- 도커 이미지 저장소(DockerHub 등)
- 이미지:태그 검색

Dockerfile

예시1: openjdk

```
FROM openjdk:11 as production

COPY target/app-0.0.1-SNAPSHOT.jar /app/app.jar

EXPOSE 11000

ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```

- openjdk:11 이미지를 베이스 이미지로 사용
- app-0.0.1-SNAPSHOT.jar 파일을 컨테이너 /app/app.jar 로 복사
- 컨테이너 포트번호 11000 노출
- Java -jar ... 실행
  (Entrypoint 는 컨테이너가 생성될 때 최초로 실행하는 명령어)

예시2: tomcat

```
FROM tomcat:9-jre11

COPY target/www.war /usr/local/tomcat/webapps

EXPOSE 8080
```

- Tomcat 9 이미지를 베이스 이미지로 사용
- 호스트OS의 target/www.war 파일을 컨테이너 .../webapps 폴더로 복사
- 컨테이너 포트번호 8080 포트 노출

Dockerfile reference | Docker Documentation

실습

- Docker Desktop 설치
- DockerHub 에서 이미지 검색하기
- 이미지를 컨테이너로 실행하기
- Docker compose 맛보기

참고링크

- Docker and Its Ecosystem. Docker is now widely used by leading... | by Kailash Verma | Medium
- [Docker] 데이터 관리(1) Volume 과 Bind mounts (velog.io)
- Docker Hub와 Docker 공식 이미지의 이해 (lainyzine.com)

끝