[IT Essential] Docker 활용법 및 Dockerize Project

> 진행자: 최광호 컨설턴트님
>
> 날짜: 2021-01-27

목차

1. Docker

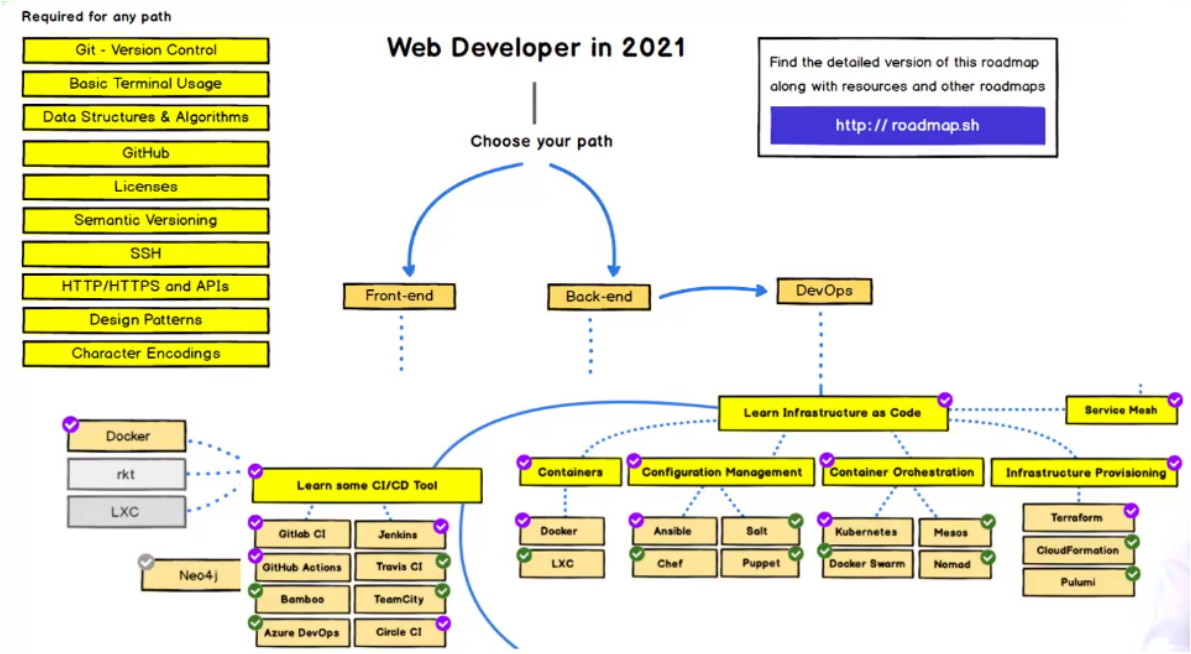
- 도커 컨테이너를 왜 배워야할까?
- Google의 컨테이너

> Google의 업무 방식
>
> Gmail에서 YouTube, 검색에 이르기까지 Google의 모든 제품은 컨테이너에서 실행됩니다. 개발팀은 컨테이너화를 통해 더욱 신속하게 움직이고, 효율적으로 소프트웨어를 배포하며 전례 없는 수준의 확장성을 확보할 수 있게 되었습니다. Google은 매주 수십억 개가 넘는 컨테이너를 생성합니다. 지난 10여 년간 프로덕션 환경에서 컨테이너화된 워크로드를 실행하는 방법에 관해 많은 경험을 쌓으면서 Google은 커뮤니티에 계속 이 지식을 공유해 왔습니다. 초창기에 cgroup 기능을 Linux 커널에 제공한 것부터 내부 도구의 설계 소스를 Kubernetes 프로젝트로 공개한 것까지 공유의 사례는 다양합니다. 그리고 이 전문 지식을 Google Cloud Platform으로 구현하여 개발자와 크고 작은 규모의 회사가 최신의 컨테이너 혁신 기술을 쉽게 활용할 수 있도록 하였습니다.

- 우리가 사용하는 많은 서비스들이 컨테이너를 사용하고 있다.

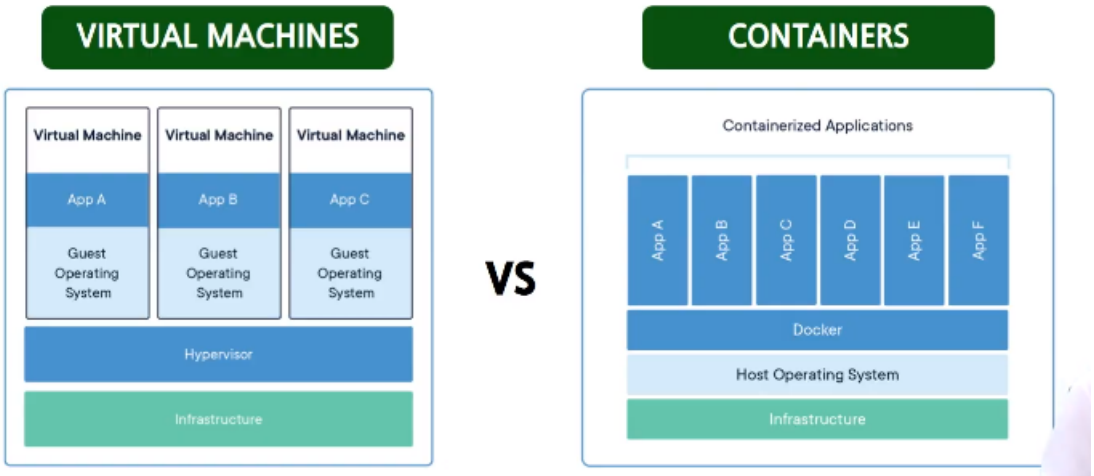
2. 웹 개발자 로드맵

- 웹 개발에 있어서도 Docker는 containerization 기술을 사용하기 위해서 알아야하며, 더 나아가서 백엔드의 영역이 DevOps까지 확장되면서 Infrastructure as Code를 이해하기 위해서는 Docker를 공부해야한다.



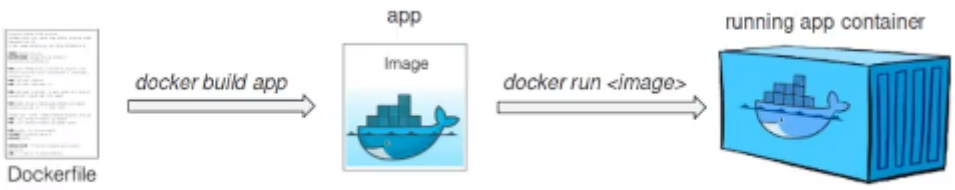
3. Docker란?

- OS 레벨 가상화를 이용하여 컨테이너화 된 소프트웨어 패키징 및 관리가 가능한 **가상화 도구**이다.

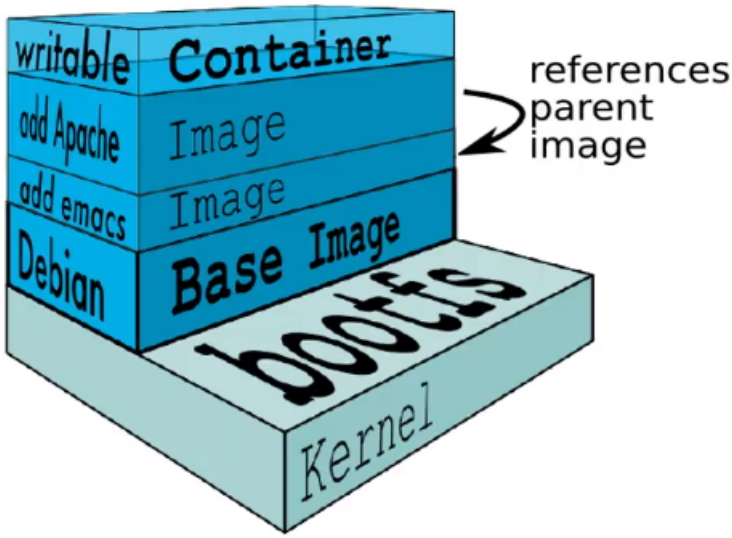


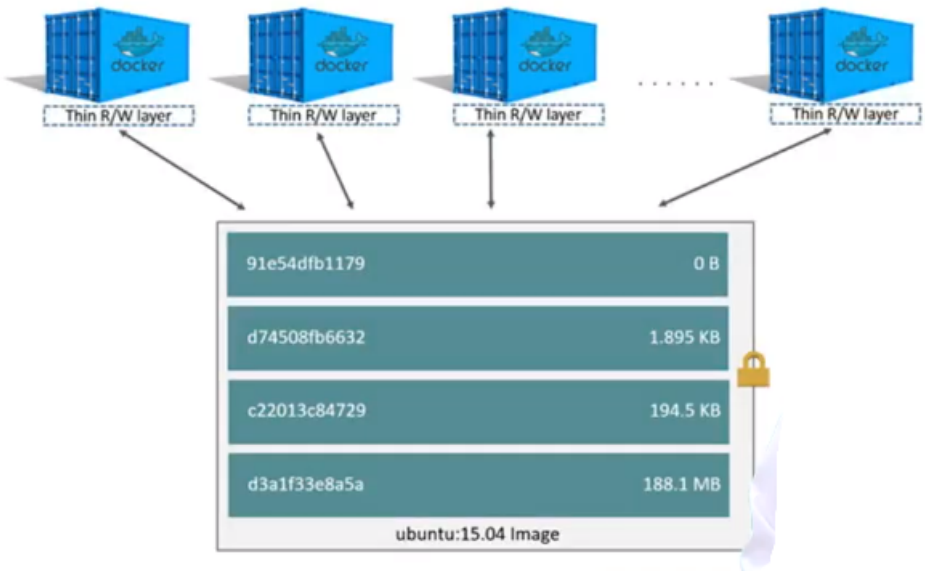
- 컨테이너 이전에는 가상화 기술이 존재하고 있었는데, 하나의 앱을 실행하기 위해 그에 맞는 OS가 통째로 올라가고 Hypervisor를 이용해서 호스트의 하드웨어에 접근하는 방식이었기 때문에 overhead가 크다는 단점이 있었다. 그에 반해 컨테이너 기술은 호스트 OS의 커널과 하드웨어를 바로 사용할 수 있어 overhead가 적고 가볍다는 장점이 있다.
- 컨테이너 기술은 Linux의 커널의 기능을 활용해서 구현되는 것이며, 1991년에 나온 기술이지만 비즈니스와 결합되어 각광 받기 시작한 것은 최근이다.

4. Docker 사용을 위한 2가지 개념



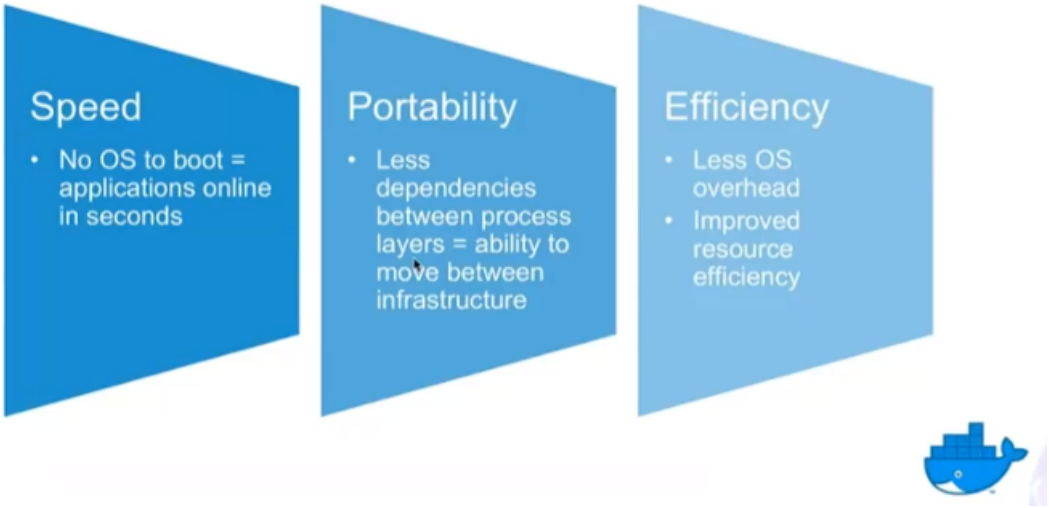
- 컨테이너화를 하기 위해서는 서버를 기술한 설계도인 **Dockerfile**이 필요하다. Dockerfile을 빌드하면 OS와 앱이 포함된 **image**가 생기며, image를 run하면 앱이 실행되고 있는 **container**가 생긴다.
- Docker를 사용하기 위해서는 image와 container의 차이점을 이해해야 한다.





  - image와 container의 관계는 class와 instance의 관계와 비슷하다.
  - image는 스택처럼 쌓인 구조를 가지고 있으며, 자식 이미지는 부모 이미지를 reference한다.
  - image는 read only지만, container는 writable하다.

5. Docker 컨테이너의 장점



1. 빠른 속도와 높은 효율성
    - 커널을 호스트 OS(Linux)와 공유해서 부팅이 필요 없고 격리된 프로세스

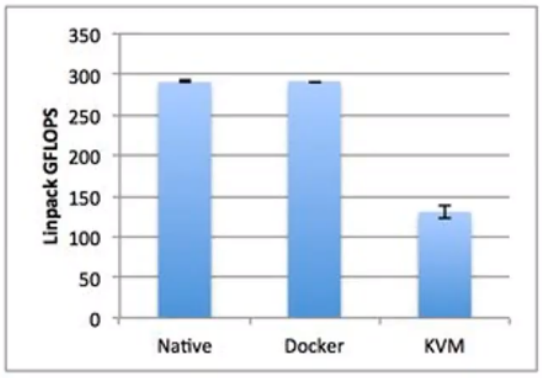
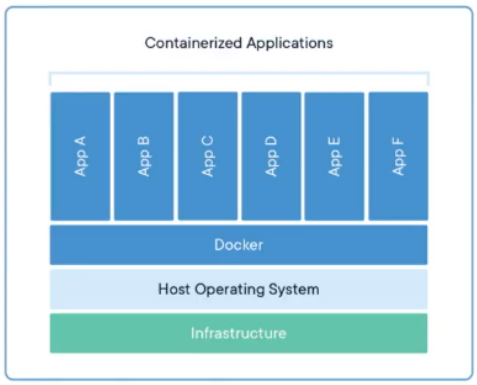
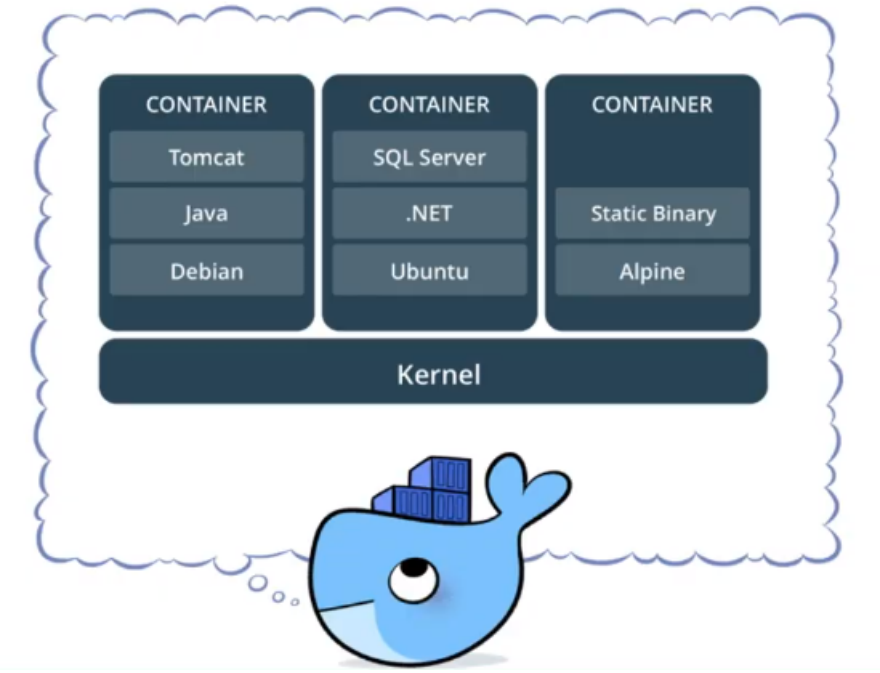


Figure 1. Linpack performance on two sockets (16 cores). Each data point is the arithmetic mean obtained from ten runs. Error bars indicate the standard deviation obtained over all runs.



    - 왼쪽 표를 보면 Native와 Docker로 실행한 속도가 거의 차이가 없을 정도로 컨테이너는 빠른 속도를 보여준다.
    - 호스트와 커널을 공유하기 때문에 따로 부팅 할 필요 없이 앱만 실행하면 된다.

2. 높은 이식성
    - OS(Linux)상에 격리된 독자적인 어플리케이션 실행 환경을 가지기 때문에 어떤 환경에서 구동해도 똑같은 실행을 보장한다. 따라서 높은 이식성을 가진다.



3. 가벼움
    - 커널을 제외한 배포판 유저랜드(프로그램, 라이브러리) 최적화
    - busybox는 아주 경량화 된 리눅스로 임베디드 장비에 사용된다.

      ```
      $ docker run busybox echo hello world
      ```

        - 위의 명령어 하나로 docker는 busybox 이미지를 다운받아 컨테이너를 만들어 실행한 후, hello world를 출력하고 죽는다. 이러한 쉬운 사용과 가벼움이 docker의 가장 큰 장점이다.
    - alpine은 경량화 된 데비안 계열의 리눅스로, 호스트 OS에 상관없이 아래의 명령어를 실행하면 docker 위에서 alpine 리눅스를 사용할 수 있다.

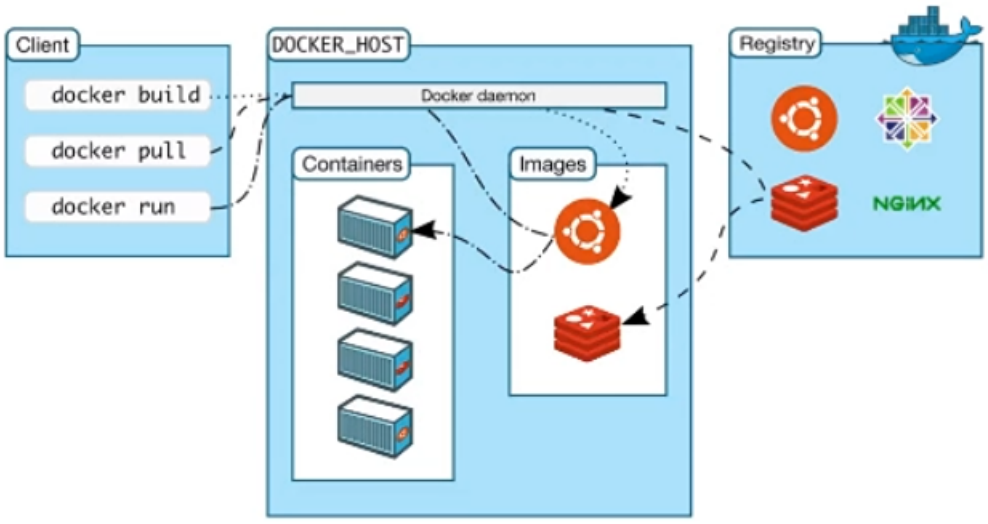
      ```
      $ docker run alpine cat /etc/issue
      ```

    - 내려 받아진 이미지들은 아래의 명령어로 확인할 수 있다.

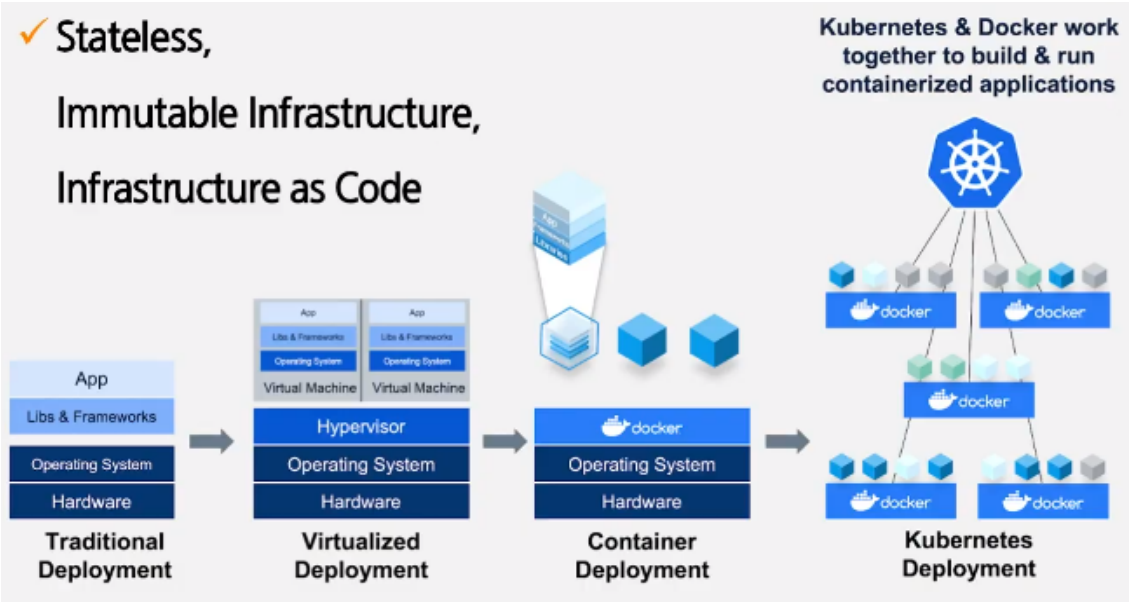
      ```
      $ docker images
      ```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
busybox	latest	b97242f89c8a	12 days ago	
alpine	latest	7731472c3f2a	10 days ago	

4. 일관된 사용성
    - 이미지 패키징 및 공유
    - 컨테이너화된 앱 실행 방식의 추상화 (어떤 앱을 실행해도 똑같은 docker 명령들을 사용한다)



5. Stateless, Immutable Infrastructure, Infrastructure as Code



- 전통적인 배포 방식에서 가상화된 배포, 그리고 container를 이용한 배포로 발전하면서 점점 더 overhead가 줄어들고 주어진 자원을 낭비 없이 효율적으로 사용 할 수 있게 되었다. 여기에서 더 나아가서 Kubernetes는 Docker와 결합해서 더 큰 규모로 효율적인 서버 관리를 할 수 있게 해준다.

6. 참고 링크

- 초보를 위한 도커 안내서
- 왜 굳이 도커(컨테이너)를 써야 하나요?
- Docker 공식 문서