Ch1. Docker 기본

생성일	@2022년 1월 19일 오후 8:52
태그	따라하며 배우는 도커와 CI환경

도커를 쓰는 이유

- 일반적으로 프로그램을 다운 받을 때, 갖고 있는 서버, 패키지 버전, 운영체제 등등에 따라 프로그램을 설치하는 과정 중에 많은 에러가 발생할 수 있다.
- 도커를 이용하여 프로그램을 설치하면 예상치 못한 에러도 덜 발생하며, 설치하는 과정도 훨씬 간단하다!

도커란 무엇인가?

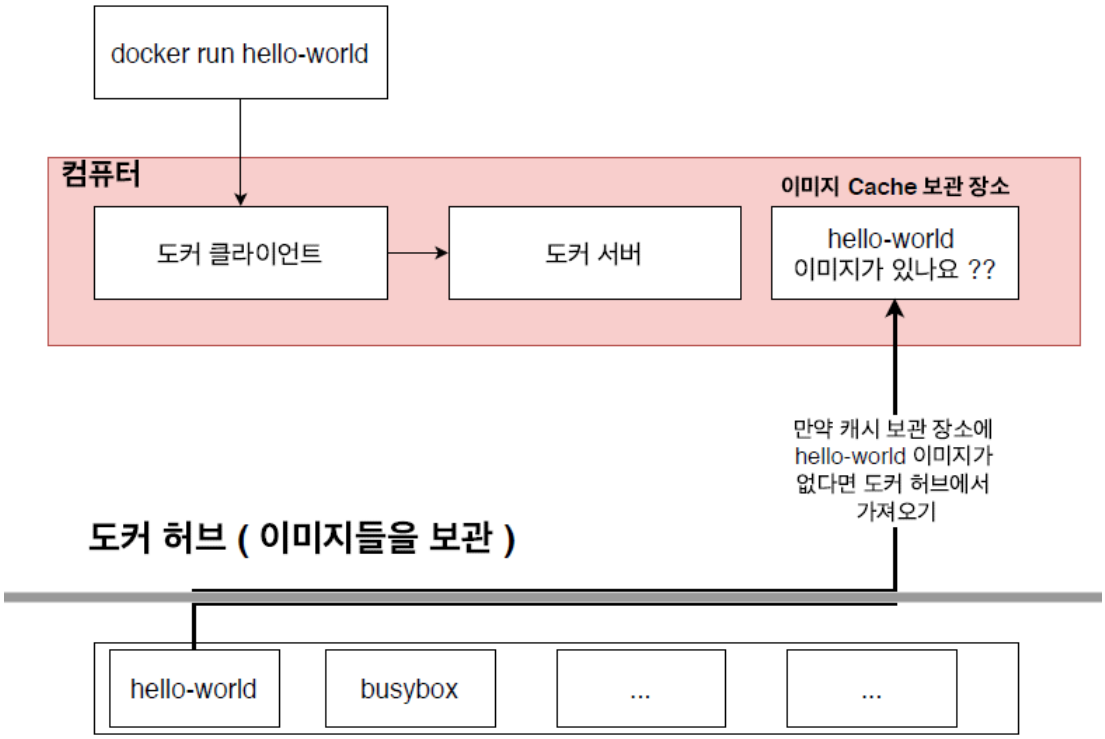
- 도커: 컨테이너를 사용하여 응용프로그램을 더 쉽게 만들고 배포하고 실행할 수 있도록 설계된 도구이며 컨테이너 기반의 오픈 소스 가상화 플랫폼이며 생태계이다.
- 컨테이너의 개념
  - 서버에서 컨테이너 안에 다양한 프로그램, 실행환경을 컨테이너로 추상화하고 동일한 인터페이스를 제공하여 프로그램의 배포와 관리를 쉽게 해준다.
  - 프로그램을 손쉽게 이동 배포 관리를 할 수 있게 해주고, AWS, Azure, GCP 등 어디에서든 실행 가능하게 해준다.

도커 이미지와 도커 컨테이너 정의

- 컨테이너: 코드와 모든 종속성을 패키지화하여 응용 프로그램이 한 컴퓨팅 환경에서 다른 컴퓨팅 환경으로 빠르고 안정적으로 실행되도록 하는 소프트웨의 표준 단위
  - 현재까지 여러 가지 방향으로 컨테이너를 정의할 때 간단하고 편리하게 프로그램을 실행시켜주는 것으로 정의
- 컨테이너 이미지: 코드, 런타임, 시스템 도구, 시스템 라이브러리 및 설정과 같은 응용 프로그램을 실행하는 데 필요한 모든 것을 포함하는 가볍고 독립적이며 실행 가능한 소프트웨어 패키지
  - 런타임에 컨테이너가 되고 도커 컨테이너의 경우 도커 엔진에서 실행될 때 이미지가 컨테이너가 된다.
  - 리눅스와 윈도우 기반 애플리케이션 모두에서 사용할 수 있는 컨테이너화된 소프트웨어는 인프라에 관계없이 항상 동일하게 실행된다.
  - 컨테이너는 소프트웨어를 환경으로부터 격리시키고 개발과 스테이징의 차이에도 불구하고 균일하게 작동하도록 함.

도커이미지는 프로그램을 실행하는 데 필요한 설정이나 종속성을 갖고 있으며 도커 이미지를 이용해서 컨테이너를 생성하고 이 도커 컨테이너를 이용해서 프로그램을 실행한다.

도커를 사용할 때의 흐름



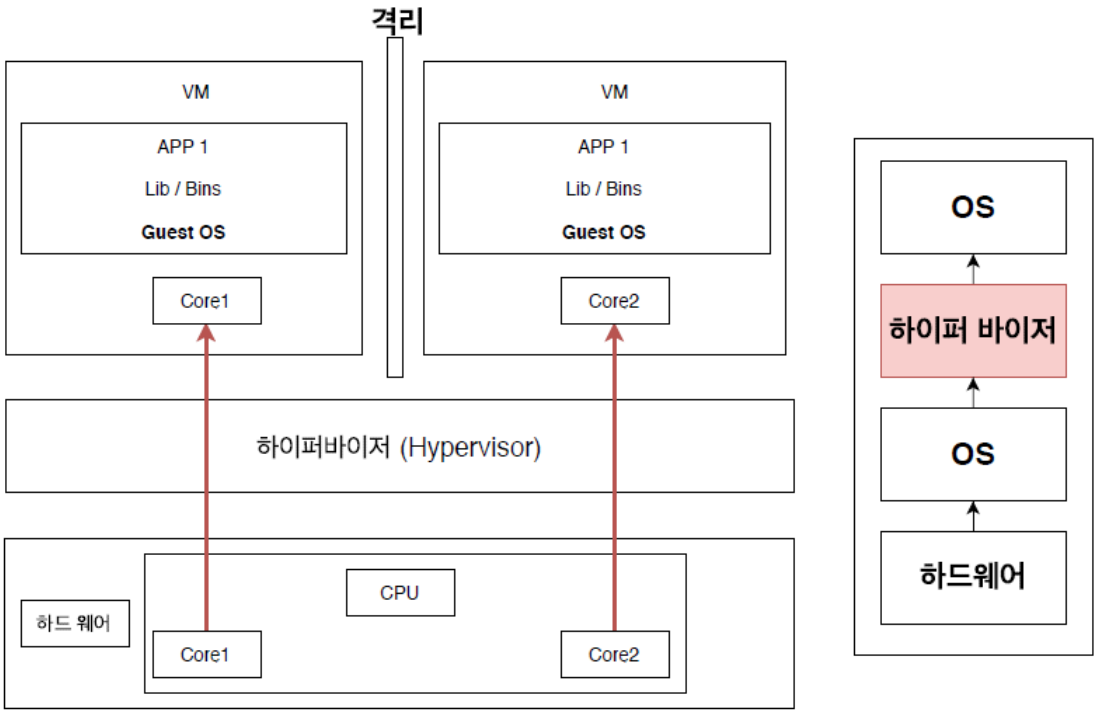
- 도커 클라이언트에 커맨드를 입력하여 클라이언트에서 도커 서버로 요청을 보냄
- 서버에서 hello-world라는 이미지가 이미 로컬에 cache가 되어 있는지 확인
- 현재는 없기에 unable to find image ~ 라는 문구가 2번째 줄에 표시
- 그러면 Docker Hub라는 이미지가 저장되어 있는 곳에 가서 그 이미지를 가져오고 로컬에 Cache로 보관
- 그 후 이제는 이미지가 있으니 그 이미지를 이용해서 컨테이너를 생성
- 그리고 이미지로 생성된 컨테이너는 이미지에서 받은 설정이나 조건에 따라 프로그램을 실행

hello-world 이미지가 캐쉬가 된 이후에 다시 한번 run하면?

- Unable to find image~라는 문구가 없이 프로그램이 실행
- 결국은 캐쉬된 이미지를 이용해서 컨테이너를 만든 후 프로그램을 실행

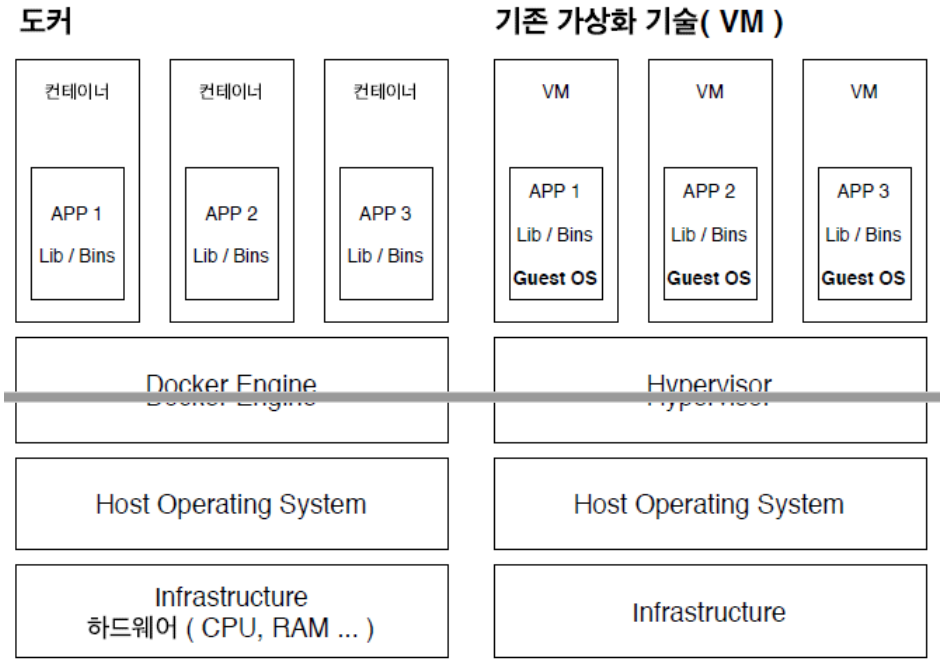
도커와 기존의 가상화 기술과의 차이를 통한 컨테이너 이해

- 기상화 기술이 나오기 전에는
  - 한 대의 서버를 하나의 용도로만 사용
  - 남는 서버 공간을 그대로 방치
  - 하나의 서버에 하나의 운영체제, 하나의 프로그램만을 운영하였음
  - 안정적이나 비효율적
- 하이퍼 바이저 기반의 가상화가 출현한 후에는
  - 논리적으로 공간을 분할하여 VM이라는 독립적인 가상 환경의 서버 이용 가능
  - 하이퍼 바이저는 호스트 시스템에서 다수의 게스트 OS를 구동할 수 있게 하는 소프트웨어
  - 그리고 하드웨어를 가상화하면서 하드웨어와 각각의 VM을 모니터링하는 중간 관리자.



- 하이퍼바이저에 의해 구동되는 VM은 각 VM마다 독립된 가상 하드웨어 자원을 할당받는다. 논리적으로 분리되어 있어서 한 VM에 오류가 발생해도 다른 VM으로 퍼지지 않는다.

도커와 VM 비교



- 공통점
  - 도커 컨테이너와 가상 머신은 기본 하드웨어에서 격리된 환경 내에 어플래케이션을 배치하는 방법
- 차이점
  - VM과 비교했을 때 컨테이너는 하이퍼바이저와 게스트 OS가 필요하지 않을 정도로 더 가볍다.

- 어플리케이션을 실행할 때는 컨테이너 방식에서는 호스트 OS 위에 어플리케이션의 실행 패키지인 이미지를 배포하기만 하면 되는데 VM은 어플리케이션을 실행하기 위해서 VM을 띄우고 자원을 할당한 다음, 게스트 OS를 부팅하여 어플리케이션을 실행 해야 해서 훨씬 복잡하고 무겁게 실행해야 한다.
- 도커 컨테이너에서 돌아가는 애플리케이션은 컨테이너가 제공하는 격리 기능 내부에 샌드박스가 있지만 여전히 같은 호스트의 다른 컨테이너와 동일한 커널을 공유하는 반면 VM은 내부에서 실행되는 모든 것이 호스트 OS나 하이퍼바이저와 독립되어 있다.

어떻게 도커 컨테이너를 격리시킬 수 있는지?

- C group (control groups) : CPU, 메모리, Network Bandwith, HD i/o 등 프로세스 그룹의 시스템 리소스 사용량을 관리 (어떤 어플이 사용량이 너무 많다면 그 어플리케이션 같은 것을 C group에 집어넣어서 CPU와 메모리 사용 제한 가능)
- 네임스페이스 (namespaces) : 하나의 시스템에서 프로세스를 격리시킬 수 있는 가상화 기술, 별개의 독립된 공간을 사용하는 것처럼 격리된 환경을 제공하는 경량 프로세스 가상화 기술
- C group과 네임스페이스는 Linux 기반에서 작동!

이미지로 컨테이너 만들기

- Docker 클라이언트에 docker run <이미지> 입력
- 도커 이미지에 있는 파일 스냅샷을 컨테이너 하드디스크에 옮긴다.
- 이미지에서 가지고 있는 명령어를 이용해서 프로그램 실행