Docker

: 컨테이너 기반의 오픈소스 가상화 플랫폼

서버에서의 컨테이너:

- 다양한 프로그램, 실행환경을 컨테이너로 추상화하고 동일한 인터페이스를 제공하여

프로그램의 배포 및 관리를 단순하게 해준다.

- 백엔드 프로그램, 데이터베이스 서버, 메시지 큐 등 어떤 프로그램도 컨테이너로 추상화 가능하다.

- 조립PC, AWS, Azure, Google Cloud 등 어디에서든 실행할 수 있다.

컨테이너

: 격리된 공간에서 프로세스가 동작하는 기술

익숙한 VMware나 VirtualBox같은 가상머신은 '**호스트 OS위에 게스트 OS 전체를 가상화'하여 사용**하는 방식

- 여러 가지 OS를 가상화(리눅스에서 윈도우를 돌린다던가)할 수 있고, 비교적 사용법이 간단하지만

무겁고 느려서 운영환경에서는 사용할 수 없다.

위와 같은 점을 개선하기 위해 CPU의 가상화기술(HVM : Hardware Virtual Machine)을 이용한

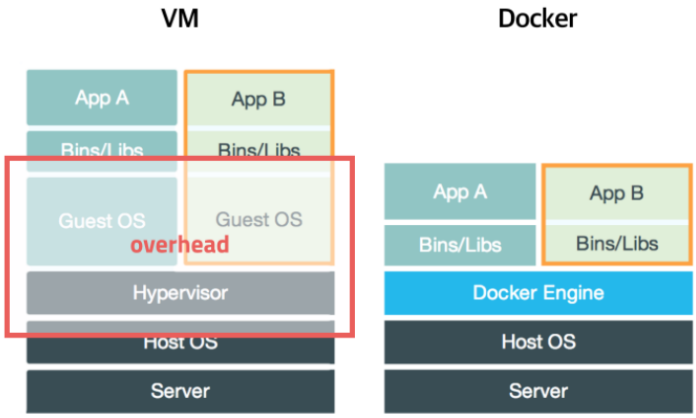
KVM(Kernel-based Virtual Machine)과

반가상화(Para Virtualization)방식의 Xen이 등장한다.

이러한 방식은 게스트 OS가 필요하기는 하지만,

전체 OS를 가상화하는 방식이 아니었기 때문에 호스트형 가상화 방식에 비해 성능이 향상되었다.

이런 기술들은 OpenStack이나 AWS, Rackspace와 같은 클라우드 서비스에서 가상 컴퓨팅 환경의 기반이 되었다.



가상머신과 도커

***KVM**(Kernel-base Virtual Machine)

: KVM을 통해 Linux를 hypervisor(or VMM : Virtual Machine Monitor)로 전환하여

호스트 머신이 게스트 또는 VM(가상 머신) 등 독립된 가상 환경 여러 개를 실행할 수 있다.

***반가상화**(PV : Para Virtualization)

: GuestOS(VM)이 control domain을 거치지 않고 바로 hypervisor를 거쳐 CPU와 RAM을 사용하여

성능 향상을 꾀함.

전가상화든 반가상화든 추가적인 OS를 설치하여 가상화하는 방법은 성능문제가 있었으며,

이를 개선하기 위해 '**프로세스를 격리**'하는 방식이 등장.

리눅스에서는 이 방식을 리눅스 컨테이너라고 하고, 단순히 프로세스를 격리시키기 때문에 가볍고 빠르게 동작한다.

CPU나 메모리는 딱 프로세스가 필요한 만큼만 추가로 사용하고, 성능적으로도 거어어—이의—의 손실이 없다.

*Docker의 기본 네트워크 모드는 Bridge 모드로 약간의 성능 손실이 있다. 네트워크 성능이 중요한 프로그램의 경우

--net=host 옵션을 고려해야 한다.

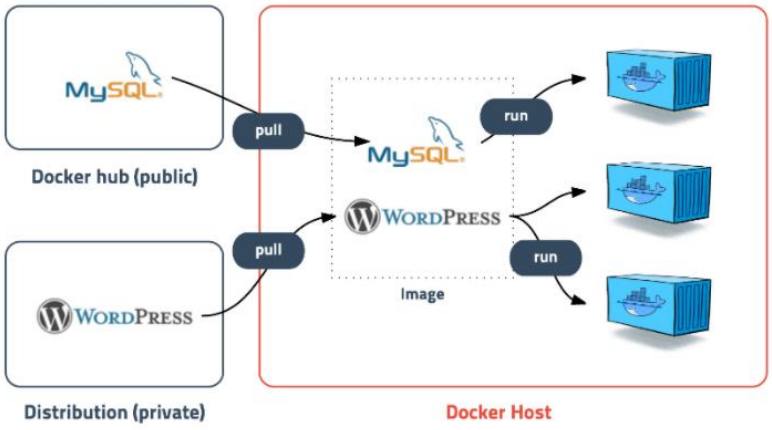
하나의 서버에 여러 개의 컨테이너를 실행하면 서로 영향을 미치지 않고 독립적으로 실행되어 마치 가벼운 VM을 사용하는

느낌을 준다. 실행중인 컨테이너에 접속하여 명령어를 입력할 수 있고 apt-get이나 yum으로 패키지를 설치할 수 있으며,

사용자도 추가하고, 여러 개의 프로세스를 백그라운드로 실행할 수도 있다. CPU나 메모리 사용량을 제한할 수 있고, 호스트의

특정 포트와 연결하거나 호스트의 특정 디렉토리를 내부 디렉토리인 것처럼 사용할 수도 있다.

이미지



Docker image

도커에서 가장 중요한 개념은 '이미지'라는 개념이다.

이미지 : 컨테이너 실행에 필요한 파일과 설정값 등을 포함하고 있는 것으로 상태값을 가지지 않고 변하지 않는다.(Immutable)

같은 이미지에서 여러 개의 컨테이너를 생성할 수 있고, 컨테이너의 상태가 바뀌거나 컨테이너가 삭제되더라도 이미지는 변하지 않고

그대로 남아 있다.

이미지는 컨테이너를 실행하기 위한 모든 정보를 갖고 있기 때문에 더 이상 의존성 파일을 컴파일하고 설치를 추가로 할 필요가 없다.

- 새로운 서버 추가 -> 이미지 다운 -> 컨테이너 생성

도커 이미지는 Docker hub에 등록하거나 Docker Registry 저장소를 직접 만들어 관리할 수 있다.

레이어

이미지는 컨테이너를 실행하기 위한 모든 정보를 가지고 있기 때문에 보통 용량이 수백 메가에 이른다.

처음 이미지를 다운받을 때는 부담이 되지 않으나 기존 이미지에 파일 하나 추가했다고 다시 다운받는다면 매우 비효율적일 것

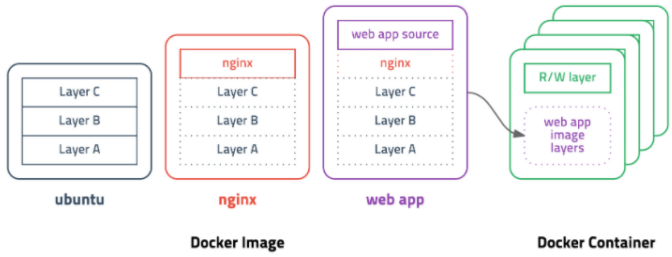
레이어!

A + B + C 가 기존의 이미지라면

nginx 이미지는

A + B + C + nginx가 된다.

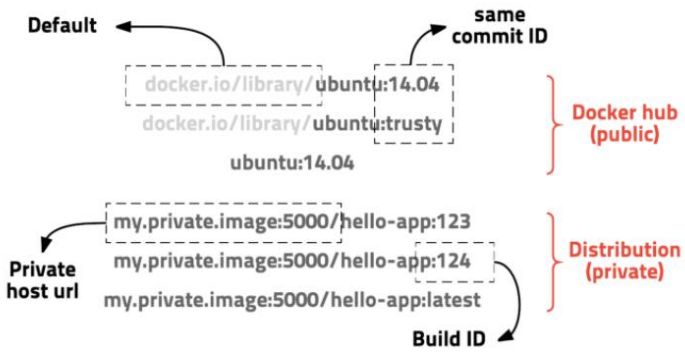
레이어 저장방식



Docker Layer

이미지는 url 방식으로 관리하며 태그를 붙일 수 있다.

태그 기능을 잘 이용하면 테스트나 롤백도 쉽게 가능하다.



Docker image url

도커는 이미지를 만들기 위해 **Dockerfile** 이라는 파일에 자체 DSL(Domain-Specific Language)언어를 이용하여 이미지 생성 과정을 적는다.

서버에 어떤 프로그램을 설치하려고 이것 저것 의존성 패키지를 설치하고 설정 파일을 만들었던 경험이 있다면 더 이상 그 과정을 블로깅하거나 메모장에 적지 말고 **Dockerfile**로 관리하면 된다. 소스와 함께 버전 관리되며 원한다면 누구나 이미지 생성과정을 보고 수정할 수 있다.

[출처] https://subicura.com/2017/01/19/docker-guide-for-beginners-1.html