컨테이너란 무엇인가?

운영체계를 기반으로 만들어진 대부분의 Software는 그 실행을 위하여 OS와 Software가 사용하는 동적 Library에 대하여 의존성을 갖습니다. 즉, Software의 실행을 위해선 OS와 Library를 포함, Software가 필요로 하는 파일 등으로 구성된 실행환경이 필요한데, 하나의 시스템 위에서 둘 이상의 Software를 동시에 실행하려고 한다면 문제가 발생할 수 있습니다. 예를 들어, Software A와 B가 동일한 Library를 사용하지만 서로 다른 버전을 필요로 하는 경우라던지 두 software의 운영 체제가 다를 경우 등 다양한 경우에서 문제가 발생할 수 있습니다. 이런 상황에서가장 간단한 해결책은 두 Software를 위한 시스템을 각각 준비하는 것인데, 시스템을 각각 준비할 경우 비용의 문제가 발생하게 된다(10개의 software일 경우 10개의 시스템이 필요). 이러한 문제점을 효율적으로 해결한 것이 바로 **컨테이너**입니다. 컨테이너(Container)는 개별 Software의 실행에 필요한 실행환경을 독립적으로 운용할 수 있도록 기반환경 또는 다른 실행환경과의 간섭을 막고 **실행의 독립성을 확보해주는 운영체계 수준의 격리 기술**을 말합니다. 컨테이너는 애플리케이션을 실제 구동 환경으로부터 **추상화**할 수 있는 논리 패키징 메커니즘을 제공합니다.

'컨테이너'에 들어가기 위한 요소

이미지(Image)

이미지는 **컨테이너 실행에 필요한 파일과 설정값등을 포함하고 있는 것**으로 상태값을 가지지 않고 변하지 않습니다(Immutable). 컨테이너는 이미지를 실행한 상태라고 볼 수 있고 추가되거나 변하는 값은 컨테이너에 저장됩니다. 같은 이미지에서 여러개의 컨테이너를 생성할 수 있고 컨테이너의 상태가 바뀌거나 컨테이너가 삭제되더라도 이미지는 변하지 않고 그대로 남아있습니다.

레이어 저장방식

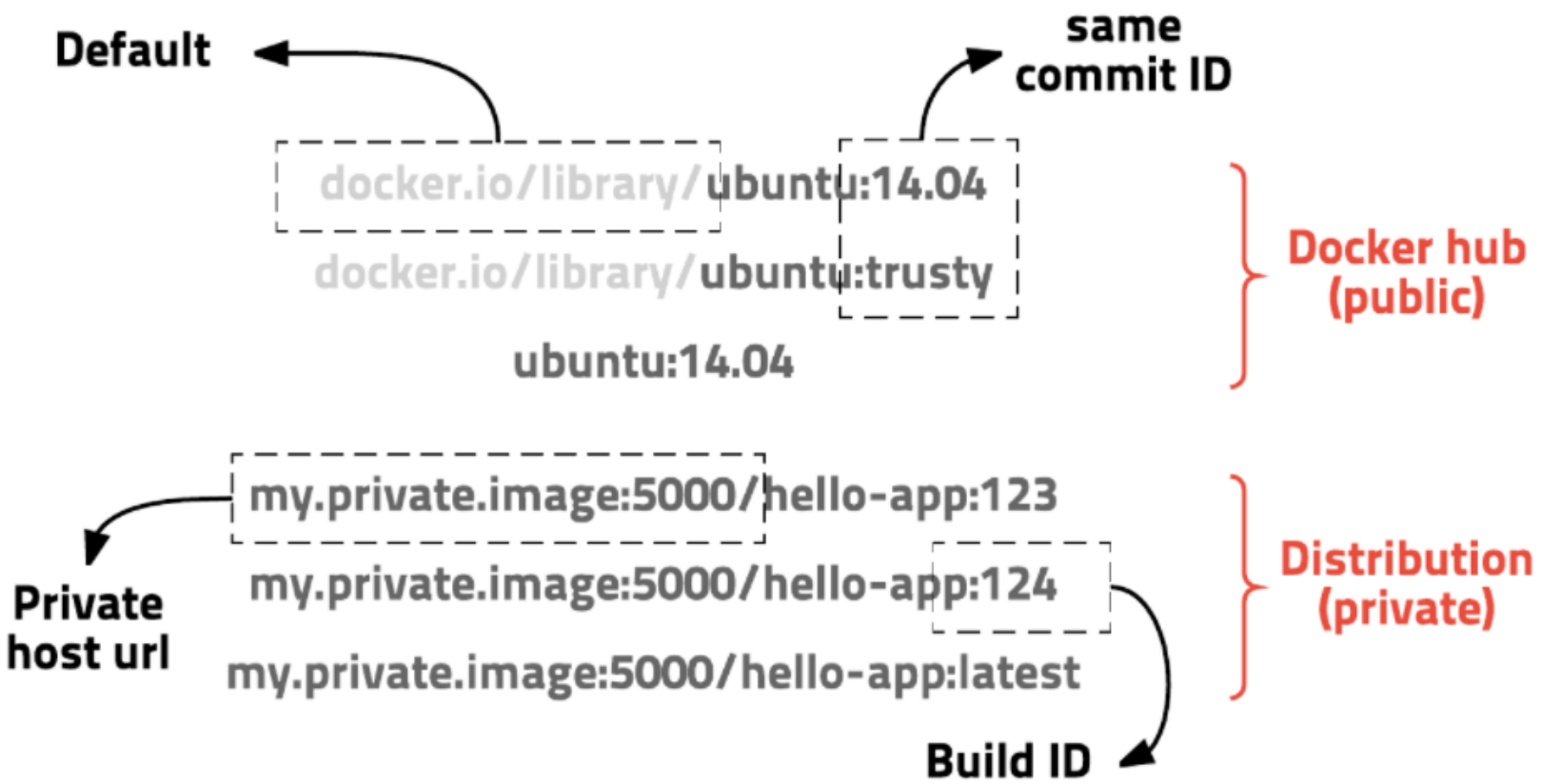
Docker Layer

도커 이미지는 컨테이너를 실행하기 위한 모든 정보를 가지고 있기 때문에 보통 용량이 수백메가MB에 이릅니다. 처음 이미지를 다운받을 땐 크게 부담이 안되지만 기존 이미지에 파일 하나 추가했다고 수백메가를 다시 다운받는다면 매우 비효율적일 수 밖에 없습니다.

도커는 이런 문제를 해결하기 위해 **레이어layer**라는 개념을 사용하고 유니온 파일 시스템을 이용하여 여러개의 레이어를 하나의 파일시스템으로 사용할 수 있게 해줍니다. 이미지는 여러개의 읽기 전용 read only 레이어로 구성되고 파일이 추가되거나 수정되면 새로운 레이어가 생성됩니다. ubuntu 이미지가 `A` + `B` + `C`의 집합이라면, ubuntu 이미지를 베이스로 만든 nginx 이미지는 `A` + `B` + `C` + `nginx`가 됩니다. webapp 이미지를 nginx 이미지 기반으로 만들었다면 예상대로 `A` + `B` + `C` + `nginx` + `source` 레이어로 구성됩니다. webapp 소스를 수정하면 `A`, `B`, `C`, `nginx` 레이어를 제외한 새로운 `source(v2)` 레이어만 다운받으면 되기 때문에 굉장히 효율적으로 이미지를 관리할 수 있습니다.

컨테이너를 생성할 때도 레이어 방식을 사용하는데 기존의 이미지 레이어 위에 읽기/쓰기read-write 레이어를 추가합니다. 이미지 레이어를 그대로 사용하면서 컨테이너가 실행중에 생성하는 파일이나 변경된 내용은 읽기/쓰기 레이어에 저장되므로 여러개의 컨테이너를 생성해도 최소한의 용량만 사용합니다.

이미지 경로



이미지는 url 방식으로 관리하며 태그를 붙일 수 있습니다. ubuntu 14.04 이미지는 `docker.io/library/ubuntu:14.04` 또는 `docker.io/library/ubuntu:trusty` 이고 `docker.io/library`는 생략 가능하여 `ubuntu:14.04` 로 사용할 수 있습니다. 이러한 방식은 이해하기 쉽고 편리하게 사용할 수 있으며 태그 기능을 잘 이용하면 테스트나 롤백도 쉽게 할 수 있습니다.