컨테이너 기술의 이점은 커널을 포함하지 않고 프로그램을 관리한다는 점이다. 커널을 포함 하지 않기에 VM에 비하여 용량이 매우 적으며 안전하다. 하지만 이러한 점 때문에 도커가 그리 각광받았을까? 과연 도커를 사용하는데에 있어 이점은 어떠한 것들이 있을까?

리눅스 기반 명령어

Namespace, Cgroup, chRoot를 활용하여 컨테이너 구현이 가능하다 각각의 역할들을 살펴보면 다음과 같다.. 먼저 chroot라고 불리는 change root directory는 말그대로 루트 디렉토리를 바꾼다는 뜻으로 트리구조에서 새로운 루트노드를 만들어 사용한다 생각하시면 편합니다. 그림을 보면 알 수 있듯이 이런 동작과정으로 process k의 루트는 더 이상 /가 아니게 되며 k의 루트 디렉토리는 더 이상 /와 B,C에 접근 할 수 없습니다. Namespace는 프로세사나 프로세스 그룹별로 자원을 불리하는 방법으로 하드웨어 뿐만 아니라 소프트웨어 요소에도 적용됩니다. 네트워크 조회 등에서도 사용되죠 Cgroup은 프로세스가 사용할 수 있는 자원의 양을 제한합니다.

이처럼 소프트웨어를 격리시키고 하드웨어 자원을 제한함으로써 컨테이너가 만들어진다. 굳이 도커를 사용하지 않아도 리눅스의 기능만으로 컨테이너는 구현 가능하다. 이런 컨터이너 기술이 빛을 발휘하는 상황을 하나 생각해보다. A 서버와 B서버에 하나의 소프트웨어를 설치한다. A서버는 기존에 존재 했던 서버이며 B서버는 새로 확장된 서버이다. 만약 A서버와 B서버의 프로그램 버전이 다르다면 어떤 일이 일어날까? 최선의 경우에는 아무 일도 일어나지 않을 수 있다. 하지만 최악의 경우 서버에 장애가 생길 수 있는데 그 이유는 다음과 같다. 설치한 소프트웨어의 최신기능을 사용하였던 경우 소프트웨어와 연관있는 라이브러리의 버전이 달라진 경우 단순히 소프트웨어의 버전이 달라진 경우 등등이다. 이러한 상황에서 서버의 에러를 고치기 위해 서버 상태를 잘 이해하고 있는 인력이 필요하지만 현업상황에서 원활하게 이루어지지 않을 수도 있다.

이런 일들을 막고자 서버로그를 사용하는데 서버에서 작업을 진행할 때 마다 이를 사내 문서에 기록하는 것이다. 도커는 이러한 서버의 기록을 코드화하여 즉 운영기록을 자유롭게 사용가능하게 해준다. 이미지와 파일로서 도커를 사용할 때 서버의 활동은 안정화되며 개발 또한 용이해진다. 이런 서버 코드화의 장점을 정리해보면 다음과 같다 서버제작 과정에 견고함과 유연성이 생긴다. 다른이가 만든 서버를 가져다 사용할 수 있다. 여러 곳에 쉽게 배포 가능하다.

결론 편하다.