Docker 19

19.Docker 활용 시나리오

지금까지 Docker 기본 사용 방법과 웹 애플리케이션 구축 예제를 알아보았습니다. 이 장에서는 대표적인 Docker 활용 시나리오 4가지를 소개하겠습니다.

클라우드 서비스가 등장하면서 이제 클릭 몇 번 만으로 수십, 수백대의 서버를 사용할 수 있게 되었습니다.

그래서 예전처럼 서버를 1년 365일 켜두고 쓰는 것이 아니라 필요할 때만 서버 인스턴스를 생성하여 사용하는 형태로 바뀌고 있습니다.

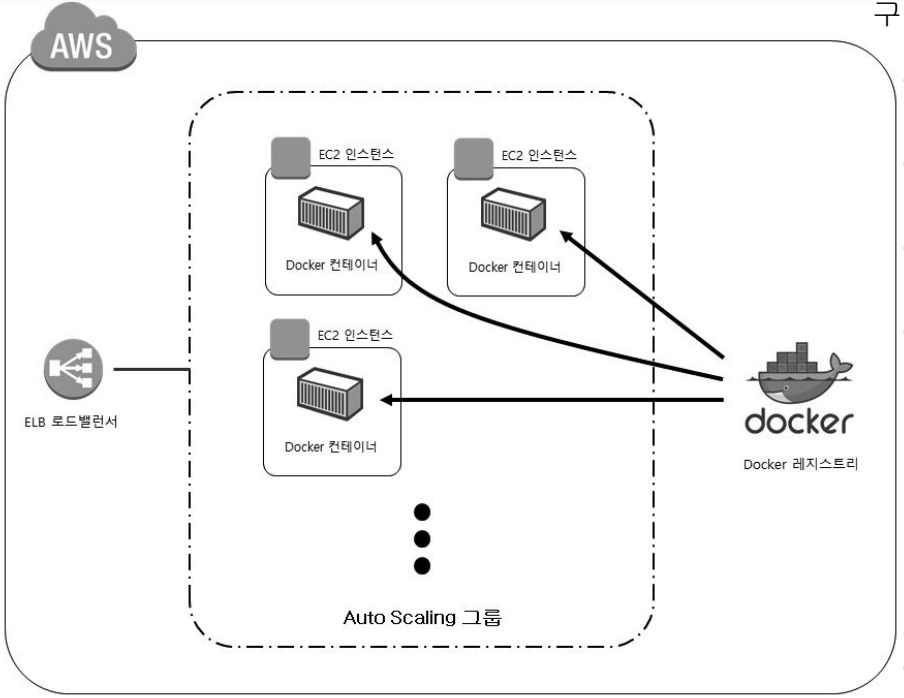
특히 부하를 분산하는 로드 밸런서와 자동 확장(Auto Scaling) 기능을 활용하여 급격히 늘어나는 트래픽에도 유연하게 대처할 수 있게 되었습니다.

Docker를 활용하면 클라우드 서비스에서 제공하는 기능과 조합하여 좀더 편리한 자동 확장 환경을 구축할 수 있습니다.

로드 밸런서

하나의 인터넷 서비스가 발생하는 트래픽이 많을 때 여러 대의 서버가 분산처리하여 서버의 로드율 증가, 부하량, 속도저하 등을 고려하여 적절히 분산처리하여 해결해주는 서비스입니다.

19-1.로드 밸런서와 연계한 확장 전개



그림은 AWS의 ELB 로드밸런서와 Auto Scaling에 Docker를 활용한 구성도입니다.

- ELB 로드밸런서는 외부에서 들어오는 트래픽을 Auto Scaling 그룹에 속한 EC2 인스턴스에 분배합니다.
- Auto Scaling은 트래픽에 따라 EC2 인스턴스를 생성하거나 삭제합니다.
- 새로 생성되는 EC2 인스턴스는 Docker 레지스트리에서 이미지를 받아와서 컨테이너로 실행합니다.
- EC2 인스턴스를 생성할 때 모든 서비스 환경이 설정된 AMI(Amazon Machine Images)를 이용하는 것보다 Docker를 사용하는 쪽이 좀 더 간편합니다. 특히 Docker를 사용하면 EC2 인스턴스와 내부 테스트환경 그리고 개발자의 PC를 동일한 환경으로 구성할 수 있습니다.
    - AMI에는 Docker만 설치한 상태로 사용합니다.
    - Docker 이미지를 빨리 받아올 수 있도록 Docker 레지스트리는 AWS 내부에 구축합니다.
- CoreOS를 사용하면 좀더 편리하게 Docker 이미지를 배포하고 고가용성(High Availablity) 서비스를 구축할 수 있습니다.

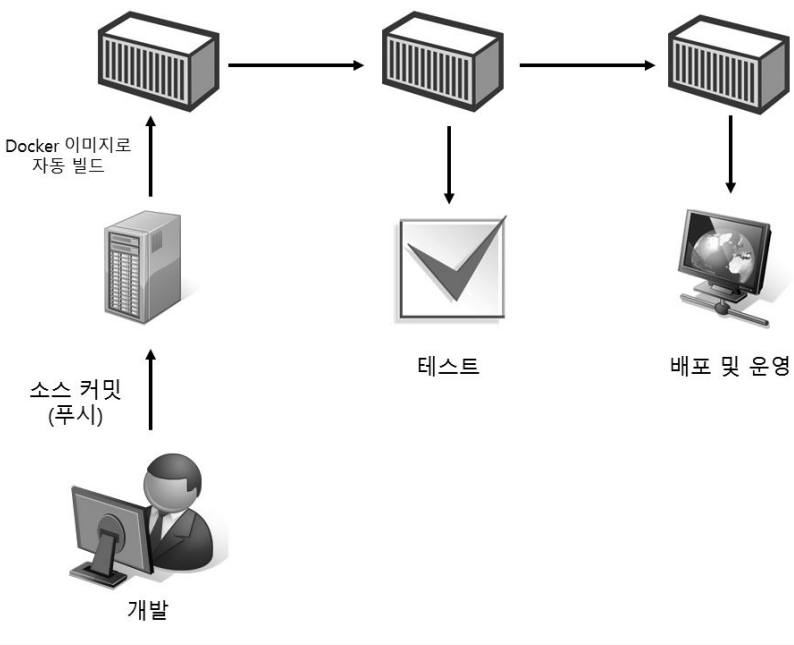
이런 방식으로 AWS뿐만 아니라 다른 클라우드 서비스에서도 동일하게 구축할 수 있습니다.

19-2.개발, 테스트, 운영을 통합

얼마전부터 DevOps라는 말이 등장하기 시작했습니다.
DevOps(Dev + Ops)는 Chef의 개발사인 Opscode에서 만든 용어입니다.
보통 개발(Development)과 운영(Operation)은 조직이 분리되어 있고, 업무도 따로 진행됩니다.
이렇게 되면 업무 효율이 떨어지고, 커뮤니케이션 비용이 발생합니다.

DevOps는 개발, 테스트, 운영에 이르는 전 구간을 자동화하여 배포 주기를 짧게하고,
표준화된 도구를 사용하여 커뮤니케이션 비용을 줄이는 환경을 뜻합니다.

19-2. 개발, 테스트, 운영을 통합



그림은 Docker로 개발, 테스트, 운영을 통합한 환경의 개념도입니다.

- 개발자가 소스 서버/빌드 서버에 소스를 올리면 Docker 이미지를 자동으로 빌드합니다.
  - 소스 서버는 Git, Subversion, Mercurial, Perforce 등과 같은 도구로 구성합니다.
  - 빌드 서버는 Jenkins, CruiseControl 등과 같은 도구로 구성합니다.
- 자동 빌드된 Docker 이미지는 미리 정의된 테스트 케이스로 자동 테스트하거나 사람이 직접 테스트합니다.
  - 웹 사이트는 phantomjs와 같은 Headless 웹 브라우저로 손쉽게 테스트할 수 있습니다.
- 테스트가 완료되면 Docker 이미지를 서비스용 서버에 배포하고, 컨테이너로 실행합니다.

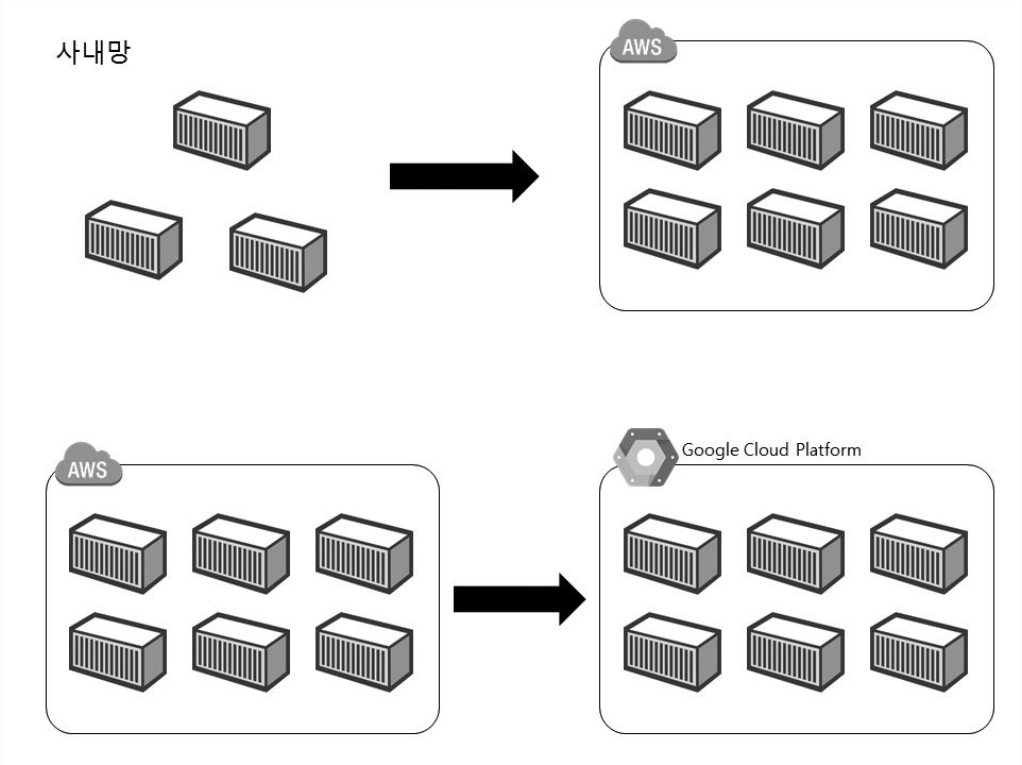
제품이나 서비스를 개발하다 보면 테스트 담당자는 계속 안된다고 얘기하고, 서비스는 사소한 문제로 늘 장애가 발생합니다. 이런 상황에서 개발자가 가장 자주하는 말이 “내 PC에서는 잘 되는데요”입니다.

Docker를 사용하면 개발자는 자신의 PC에서 이미지와 컨테이너를 생성하여 서비스 환경과 동일한 상태에서 개발을 할 수 있습니다. 그리고 테스트 담당자는 서비스 환경, 개발자의 PC와 동일한 상태에서 테스트를 할 수 있습니다. 즉 Docker라는 공통된 도구를 사용하므로 개발자 PC의 환경, 테스트 환경, 서비스 환경이 달라서 발생하는 각종 문제를 방지할 수 있습니다.

DevOps에서 한 발 더 나가아서 운영 부서가 필요없는 NoOps 환경도 생각해볼 수 있습니다.

19-3.손쉬운 서비스 이전

Docker만 설치되어 있으면 어디든지 Docker 컨테이너를 생성할 수 있기 때문에 사내망에서 클라우드 서비스로 또는, 클라우드 서비스에서 다른 클라우드 서비스로 손쉽게 이전할 수 있습니다.



19-4.테스트 용도

리눅스 기반으로 개발은 VMware나 VirtualBox와 같은 가상 머신을 활용하여 개발 환경을 구성해보고 테스트를 합니다. 하지만 가상 머신은 완전한 운영체제를 한 번 더 실행하기 때문에 상당히 무겁고 용량도 많이 차지합니다.

Docker를 활용하면 실험적인 환경을 빠르게 구성해볼 수 있습니다. 다양한 라이브러리와 패키지를 설치해서 써보고 필요 없을때는 컨테이너를 삭제하면 그만입니다. 한 번 만들어놓은 개발 환경은 Dockerfile로 작성하여 공유하면 편리합니다.