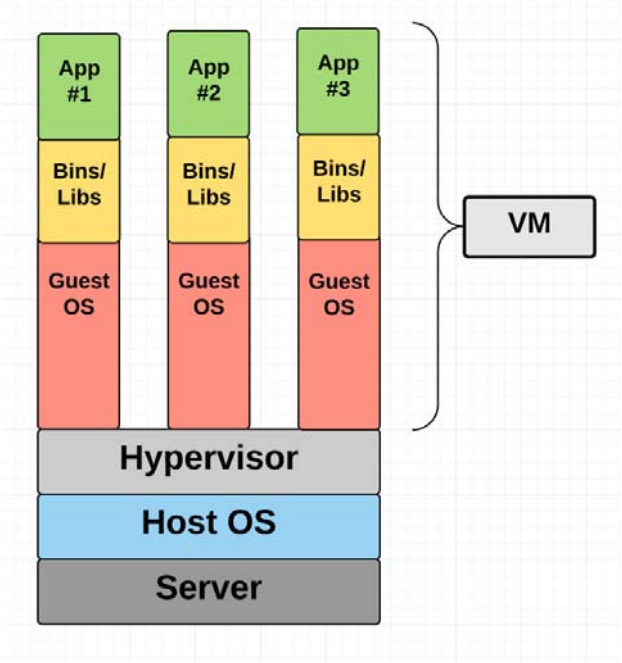
새로운 가상화 기술 : Container

가상화의 종류 복습

Hypervisor 기반의 가상화 : Guest OS를 설치하고 동작시키므로 무거움

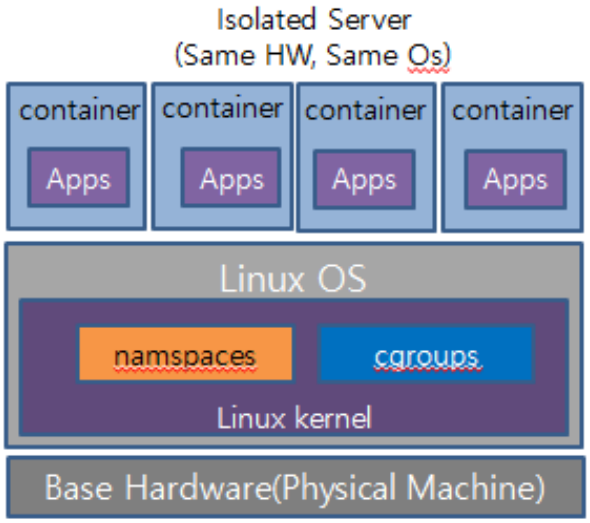


새로운 가벼운 방법: 프로세스 **격리** 구조(컨테이너) – 어떻게 격리 시키지?

- HOST OS 에서 Guest OS 없이 바로 동작 가능한 구조
- 컨테이너라는 독립된 공간에서 프로세스들이 동작(**운영체제는 공유**)

컨테이너란?

- 단일 Host OS 위에서 여러 개의 프로세스가 고립된 공간에서 동작하는 구조
- 대표적인 초기기술로 **리눅스 컨테이너가 있음(LXC)**
  - Linux의 Namespace, Cgroup(control group) 기술을 이용



LXC(Linux Container)

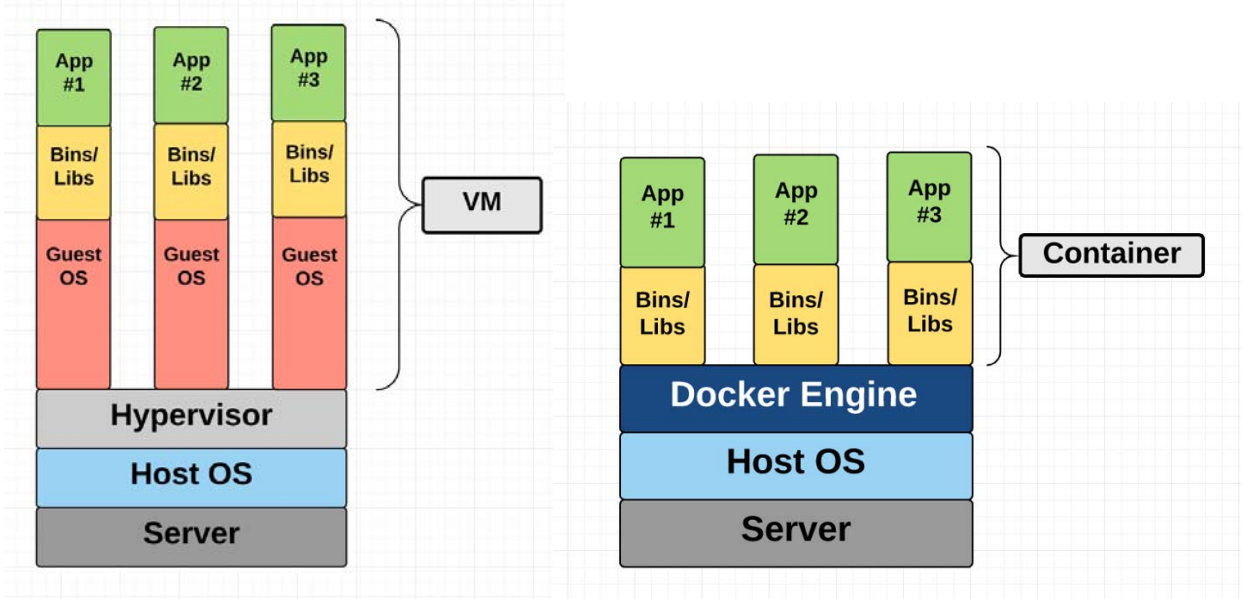
https://containerd.io/

리눅스 컨테이너(LXC)

- Linux 운영체제에서 애플리케이션(프로세스)을 격리하여 독립성 제공
- 컨테이너 공간의 격리를 리눅스 커널의 cgroup과 namespace 개념을 이용함
  - Namespace : 사용되는 다양한 변수 등등의 Name들을 분리기술을 적용해서 각 그룹이 독자적으로 사용하게 함
  - Cgroup : 프로세스 그룹의 cpu, 메모리와 같은 시스템 자원을 격리
- 장점
  - 프로세스만 동작 시키는 구조이기 때문에 생성, 시작, 종료가 빠름
  - 하나의 OS에서 동작 하기 때문에 자원을 효율적으로 사용가능
  - 이식성이 높음 (Portability)
- 단점
  - 컨테이너는 커널을 HOST OS와 공유해야 하기 때문에 HOST OS에 종속적임 (ex, 리눅스 컨테이너 에서는 리눅스 외 OS동작 불가)
  - 컨테이너별 커널 구성 불가 (커널 공유)

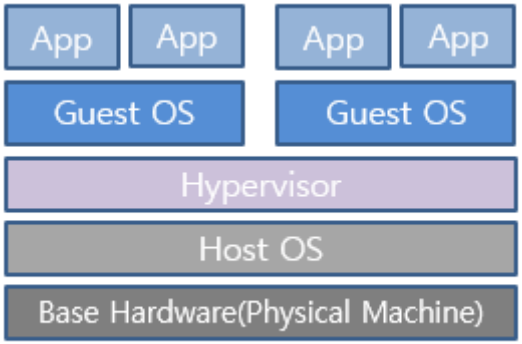
VM vs 컨테이너

- VM
  - HOST OS 위에서 Hypervisor를 통해 자원을 가상화 하여 VM을 동작
  - HOST OS 위에 Guest OS가 동작하는 구조
- Container
  - HOST OS에서 프로세스를 위한 공간을 별도로 분리
  - 기본적인 Binary, Library 만을 guest os 대신 사용

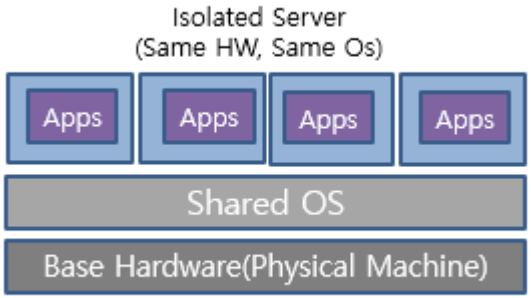


VM vs 컨테이너

Container vs VM



Virtual Machine



Container

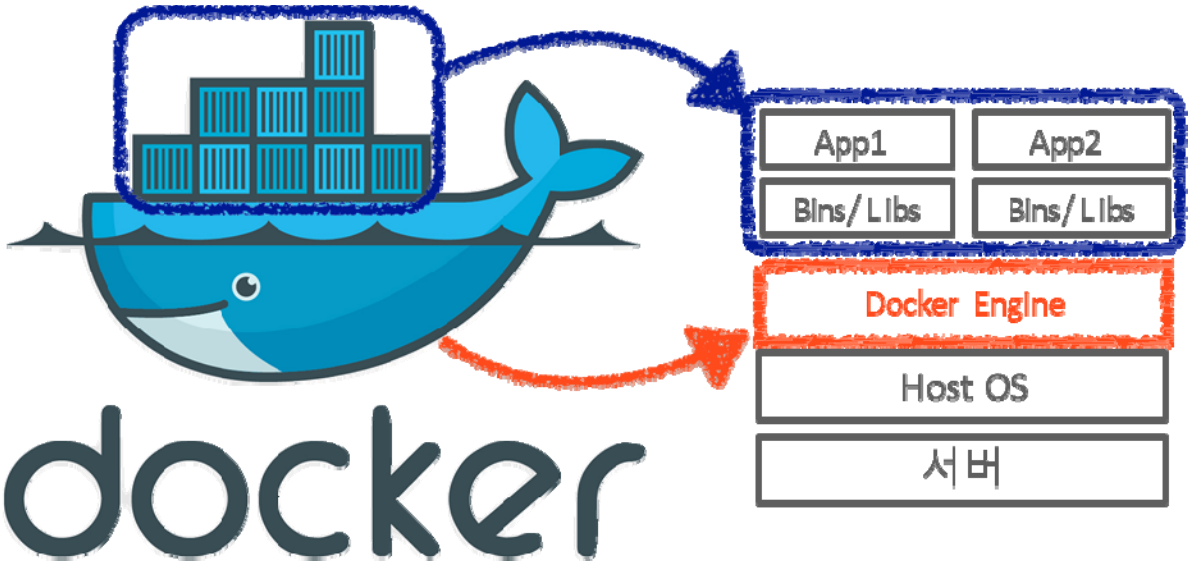
항목	컨테이너	가상 머신
하이퍼바이저	**X**	**O**
Guest OS	**X**	**O**
커널 자원 분리	**X**	**O**
시작 및 종료 시간	빠름	느림
자원 효율성	높음	낮음

첫 확산된 컨테이너 화 기술 : 도커(Docker), 컨테이너 관리기술: Docker (Swarm)

- 2013년 3월에 시작된 오픈 소스 프로젝트

그러나 지금의 컨테이너 관리기술은 **Kubernetes**가 대세

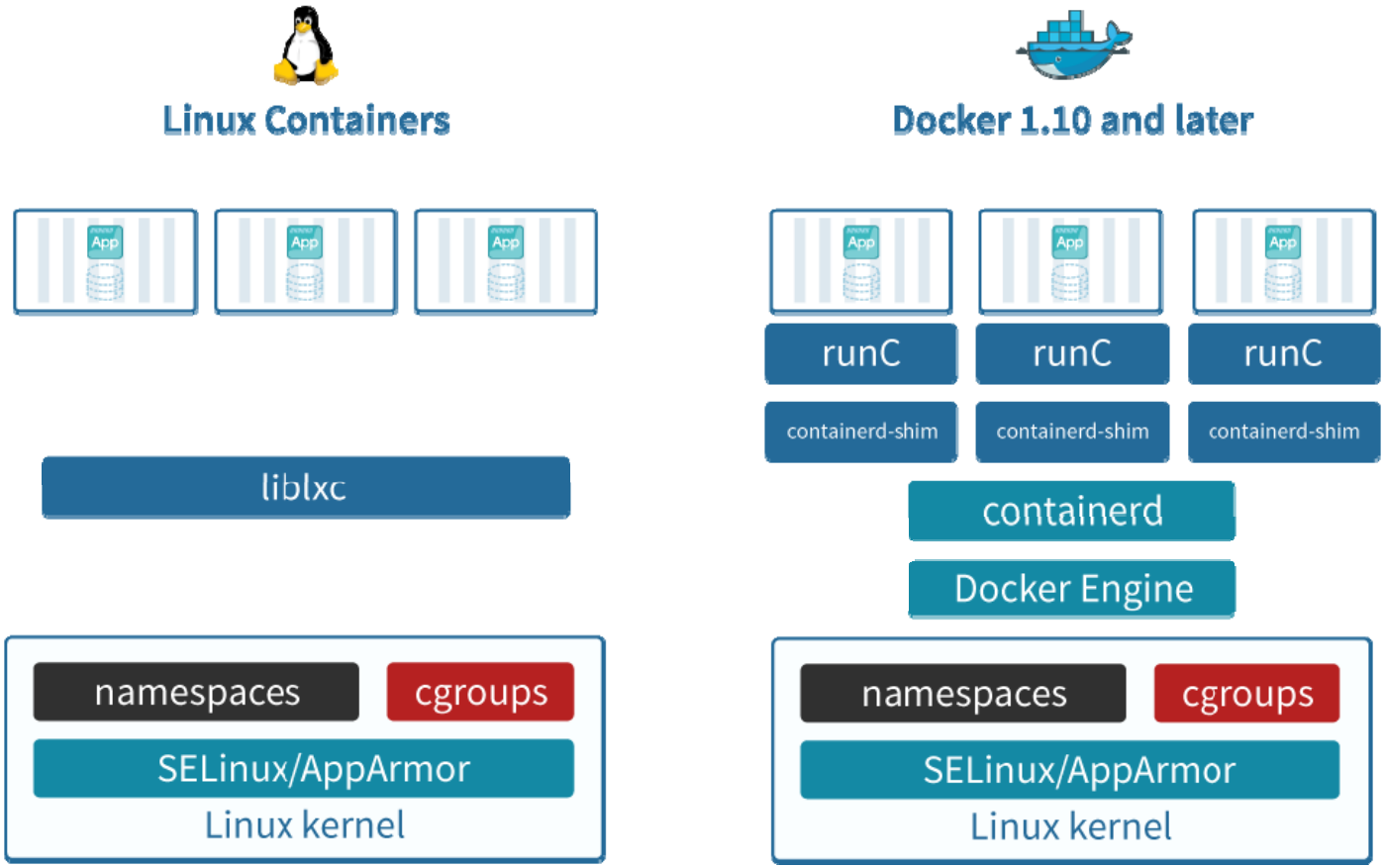
- Linux Container, Control Group, Namespace, AUFS 등의 기술을 사용



https://docs.docker.com/get-started/#test-docker-version

리눅스 컨테이너→ Docker

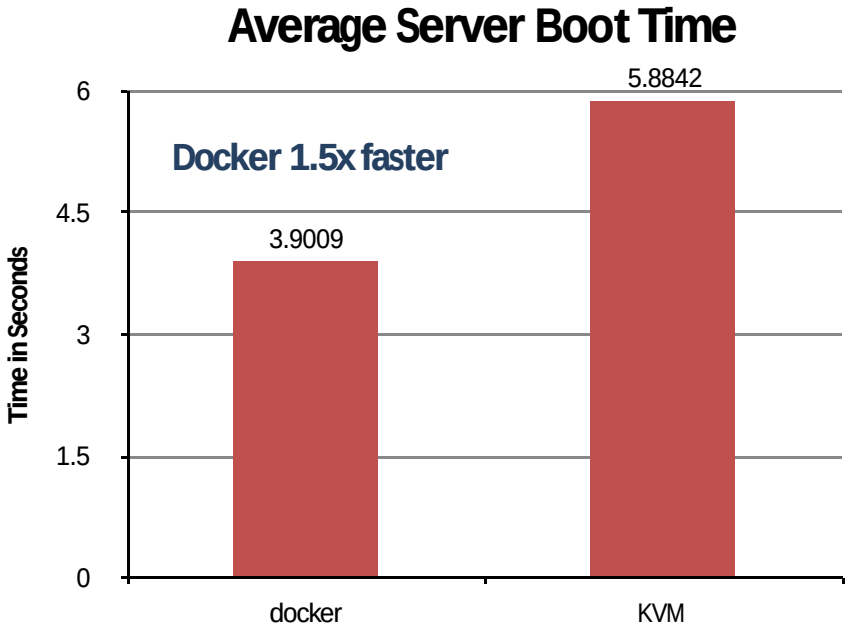
- Linux Container 시작은 Libcontainer 라는 자체 라이브러리를 사용
- Docker는 관리데몬(Containerd)과 런타임 (runc)를 사용

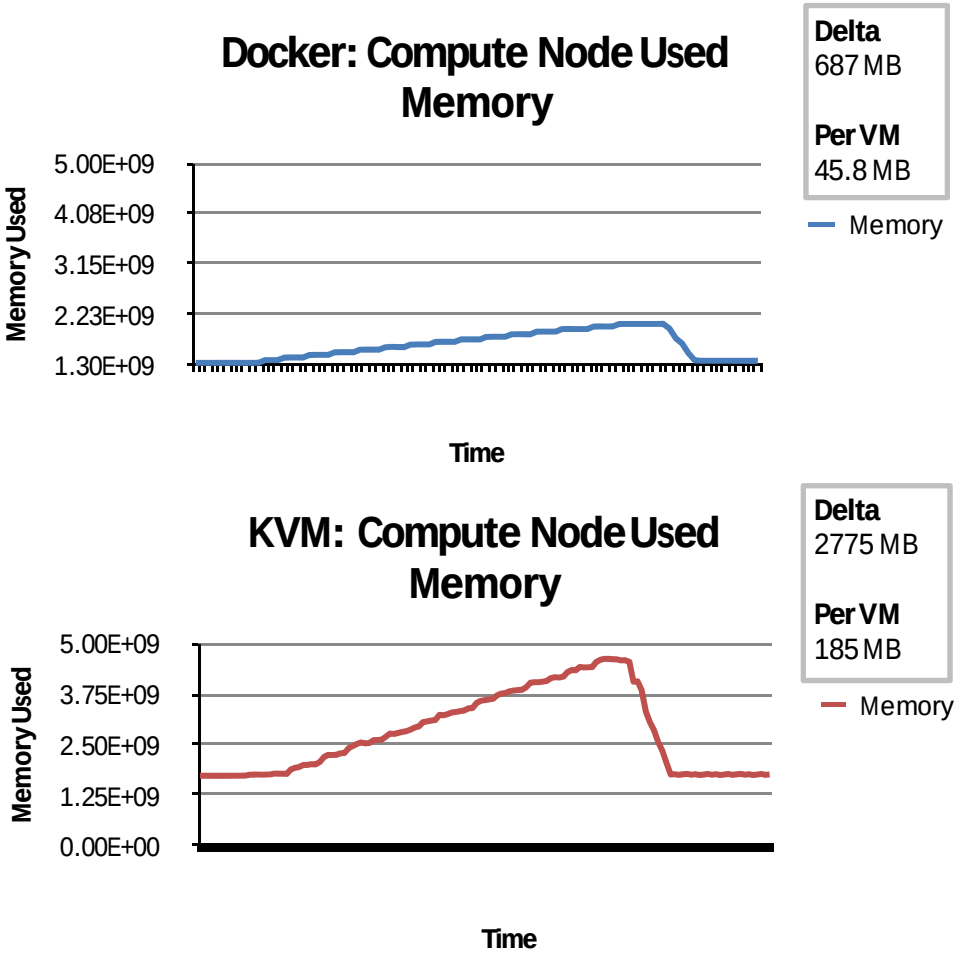


http://wiki.rockplace.co.kr/pages/viewpage.action?pageId=3868344

Docker vs VM(성능 비교)

- **Docker vs KVM:** 부팅 시간 및 메모리 사용량 **(15**개의 **VM** 연속**)**

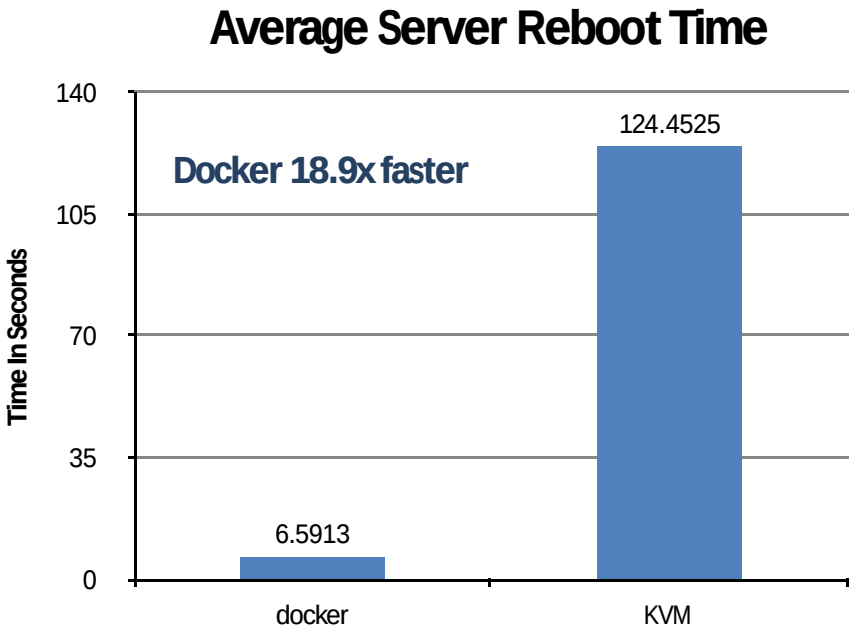


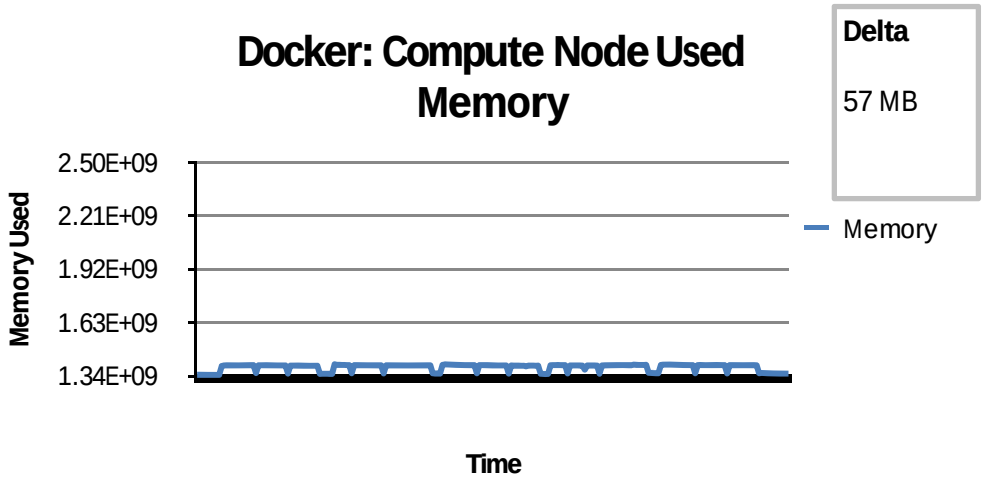


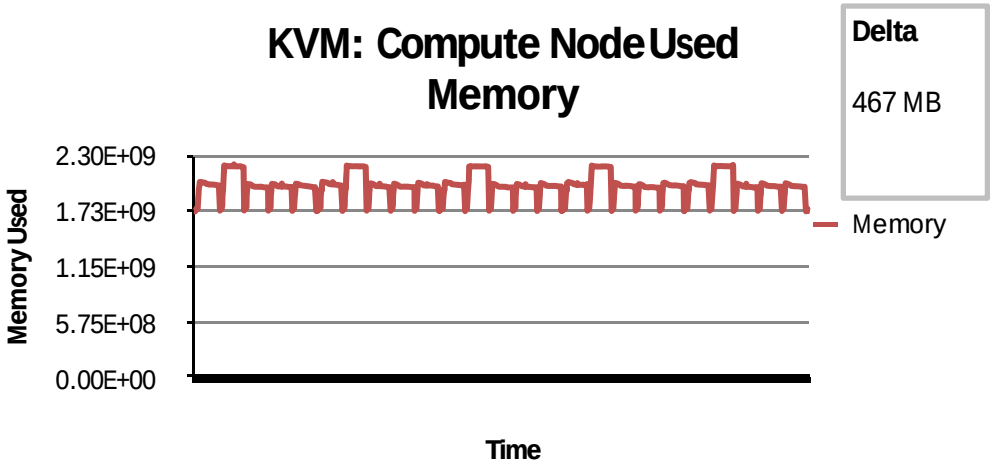
자료 출처: http://www.slideshare.net/BodenRussell/kvm-and-docker-lxc-benchmarking-with-openstack

Docker vs VM(성능 비교)

- **Docker vs KVM: 재부팅 시간 및 메모리 사용량 (15개의 VM 연속)**







자료 출처: http://www.slideshare.net/BodenRussell/kvm-and-docker-lxc-benchmarking-with-openstack

Container 의 장점

- 애플리케이션의 빠른 배포
  - 컨테이너 형식이 표준화됨
  - 개발자와 관리자의 업무를 분리
    (개발자는 컨테이너 내부만, 관리자는 컨테이너 관리만)
  - 개발 테스트 및 수정이 빠름

- 설치 및 확장이 쉽고, **이식성이 좋음**
  - **다양한 인프라에 설치가능(베어메탈, VM, 타 IaaS)**
  - 실험환경**에서 쉽게** 실제 동작하는 환경으로 이동 가능
  - 컨테이너 규모 확장 및 축소가 쉽고 빠름
  - 컨테이너 실행/종료가 빠름

- 오버헤드가 낮음
  - 하이퍼바이저가 필요 없음
  - 서버 자원을 더 효과적으로 사용 가능
  - 장비 및 라이센스에 소비되는 비용 감소

- 관리가 쉬움
  - 수정된 부분만 소규모로 적용