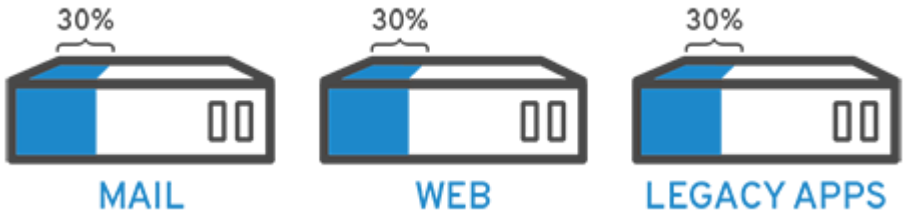
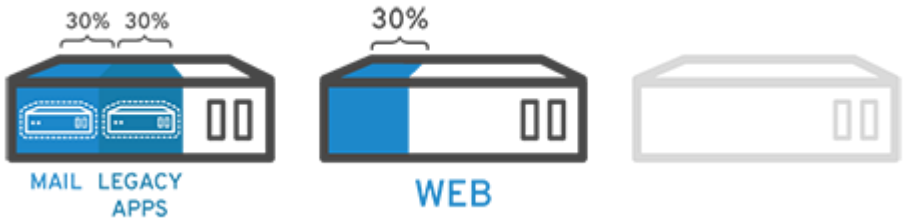
가상화 Virtualization

가상화는 실제 하드웨어에서 추상화된 계층에서 컴퓨터 시스템의 가상 인스턴스를 실행하는 프로세스이다. 물리 장치의 리소스를 분할하거나 통합해서 추상화된 논리적인 리소스로 가상화 장치를 만들고, 사용자는 가상화 장치를 물리장치처럼 사용하는 것이다. 가상화 된 컴퓨터 위에서 실행되는 응용 프로그램은 운영 체제, 라이브러리 및 기타 프로그램이 게스트 가상화 된 시스템에 고유하고 위치하는 호스트 운영 체제에 연결되지 않은 자체 전용 컴퓨터에있는 것처럼 보일 수 있다. 이를 통해 시스템 자원의 효율성을 높일 수 있다.



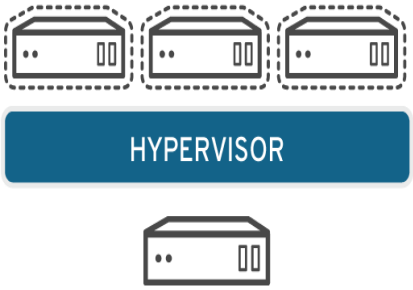
전통적으로 1개의 서버, 1개의 운영체제 같이 개별 서버에서 개별 작업을 실행하는 것이 쉽고 안정적인 경우가 많다. 그러나 가상화를 사용하면 위의 메일 서버를 2개의 고유한 서버로 분할해 독립적인 작업을 처리하고 오른쪽의 legacy apps를 Migration할 수 있다. 마찬가지로 하드웨어도 더 효율적으로 사용할 수 있다. 메일 서버의 사용률을 높이고, 이제 빈 서버를 재활용하거나 서버 사용을 중지해 유지관리 비용을 줄일 수 있다.



가상화에는 흔히 알고 있는 서버뿐만 아니라 물리적으로 존재하는 CPU, 메모리, NIC, 스토리지 등 거의 모든 하드웨어 리소스를 가상화할 수 있다. 이때 물리적으로 존재하는 하드웨어를 물리 장치라 부르고, 그것을 논리적으로 연결하거나 분할, 통합해서 새로운 장치로 만들어낸 것을 가상화 장치라고 부른다. 가상화 장치는 사용자로 하여금 어떠한 물리적인 구성을 통해 생성하였는지 전혀 알 수 없다.

1) 가상화 종류

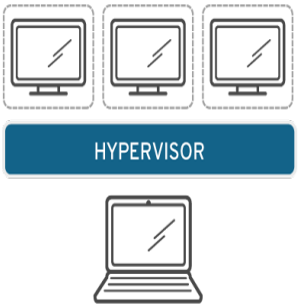
- 서버 가상화 (Server Virtualization)



서버를 가상화하는 기술을 사용해서 한 대의 베어메탈 서버[1]에 여러 대의 가상 시스템을 만드는 것을 말한다. 가상 시스템은 베어메탈 서버의 특성을 대부분 상속받는다. 따라서 가상 시스템을 구축한다면 물리서버로 시스템을 구성할 때와 거의 유사한 순서로 진행하면 된다.

서버는 대량의 특정 태스크를 매우 효과적으로 처리해 노트북 및 데스크탑 등의 다른 컴퓨터가 다양한 태스크를 처리할 수 있도록 하는 컴퓨터이다. 서버를 가상화하면 서버가 이러한 특정 기능을 더 많이 수행할 수 있으며 서버 파티셔닝을 통해 구성 요소로 여러 기능을 지원할 수 있다.

- 운영 체제 가상화, 컨테이너화 (OS-level Virtualization, Containerization)



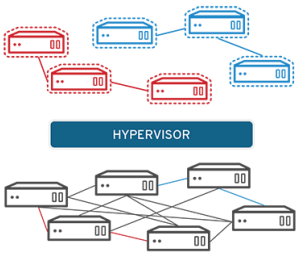
운영 체제 가상화는 운영 체제의 중앙 태스크 관리자인 커널에서 이루어진다. 커널은 OS 의 여러 격리된 사용자 공간(user-space) 인스턴스의 존재를 허용하며, 각각의 인스턴스는 실행중인 프로그램의 관점에선 실제 컴퓨터처럼 보일 수 있다. 이렇게 하면 Linux 환경과 Windows 환경을 함께 실행할 수 있다. 서버 리소스를 효율적으로 사용하기 위해 OS 수준 아키텍처의 낮은 오버 헤드를 활용한다. 이는 단일 물리적 서버에서 많은 VM 을 실행하는 동안 낮은 오버 헤드를 제공한다.

- 컴퓨터에 고도의 OOTB(Out Of The Box) 기능이 필요하지 않으므로 하드웨어에 많은 비용이 소모되지 않습니다.
- 모든 가상 인스턴스를 모니터링하고 격리할 수 있으므로 보안이 강화됩니다.
- 소프트웨어 업데이트와 같은 IT 서비스에 소요되는 시간이 절약됩니다.

---

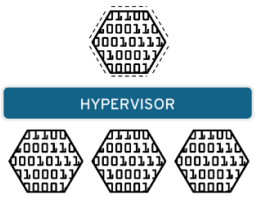
[1] 소프트웨어가 설치되지 않은 하드웨어. 일반적으론 운영체제가 설치되지 않은 상태의 서버를 의미

- 네트워크 기능 가상화



네트워크 기능 가상화(Network Functions Virtualization, NFV)는 디렉터리 서비스, 파일 공유, IP 설정과 같은 네트워크의 주요 기능을 분리하여 이러한 기능을 환경에 배포합니다. 소프트웨어 기능이 속해 있는 물리 머신으로부터 기능을 분리하면 특정 기능을 새 네트워크에 함께 패키징하고 이를 환경에 할당할 수 있습니다. 네트워크를 가상화하면 스위치, 라우터, 서버, 케이블, 허브 등 여러 개의 독립적인 네트워크를 생성하는 데 필요하며 특히 통신 산업에서 일반적으로 사용되는 물리 구성 요소의 수가 줄어듭니다.

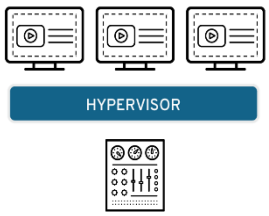
- 데이터 가상화



여러 곳에 분산되어 있는 데이터를 단일 소스로 통합할 수 있다. 데이터를 동적 공급 요소로 다루어 여러 소스에서 데이터를 동시에 가져오고 새로운 데이터 소스를 손쉽게 통합하며 사용자의 요구에 따라 필요한 데이터를 필요한 형식으로 적시에 애플리케이션 또는 사용자에게 제공한다.

기존의 ETL(Extract, Transform, Load / 추출, 변환, 요약) 프로세스처럼 데이터 분석을 위해 데이터를 여기저기 보내거나 복사하여 저장하는 방식과는 달리, 데이터 가상화에서 데이터들은 제자리를 유지한 채 인메모리에 가상의 데이터베이스를 만들어 데이터에 대한 접근을 가능하게 해주는 방식

- 데스크탑 가상화



데스크탑 가상화는 단일 머신에서 여러 운영 체제를 배포할 수 있는 운영 체제 가상화와 혼동하기 쉬우며, 시뮬레이션된 데스크탑 환경이 중앙 관리자 또는 자동화된 관리 툴을 통해 수백 개의 물리 머신에 동시 배포되도록 지원한다. 동일한 컴퓨터의 자체 VM 에서 각각 여러 데스크톱 운영 체제를 실행할 수 있다.

- **VDI (가상 데스크톱 인프라)** 는 중앙 서버의 VM 에서 여러 데스크톱을 실행하고 씬 클라이언트 장치에 로그인하는 사용자에게 스트리밍한다. 이러한 방식으로 VDI 를 사용하면 조직이 장치에 OS 를 설치하지 않고도 모든 장치에서 다양한 OS 에 액세스 할 수 있습니다. 더 자세한 설명은 " VDI (가상 데스크톱 인프라) 란? "를 참조
- **로컬 데스크톱 가상화** 는 로컬 컴퓨터에서 하이퍼 바이저를 실행하여 사용자가 해당 컴퓨터에서 하나 이상의 추가 OS 를 실행하고 필요에 따라 기본 OS 를 변경하지 않고도 한 OS 에서 다른 OS 로 전환 할 수 있다.

각 머신에서 물리적으로 설치, 설정, 업데이트되는 기존의 데스크탑 환경과 달리 데스크탑 가상화는 관리자가 모든 가상 데스크탑에서 설정, 업데이트, 보안 점검을 대규모로 수행할 수 있다.

2) 가상머신으로 가는 과정, Emulation

가상화의 초기 사용은 **에뮬레이션**이었으며, 다른 운영체제의 환경을 복제하는 작업을 뜻한다. 이를 통해 개발자는 호환되지 않는 하드웨어에서 프로젝트를 효과적으로 테스트할 수 있었다. 에뮬레이터는 다른 프로그램이나 장치를 모방하는 컴퓨터 프로그램 또는 전자기기의 능력을 말한다.

컴퓨터와 관련하여 PC 또는 MAC 에 설치하고 실행할 수있는 일부 소프트웨어 에뮬레이터를 보셨을 것이라 생각된다. 이 소프트웨어는 Nintendo 또는 기타 게임 콘솔과 같은 구형 시스템의 특성을 재현한다. 예를 들어 업무용 데스크톱에서 Super Mario Bros.를 실행할 수 있다. 이 경우 소프트웨어 에뮬레이터는 기본 하드웨어가 x86 아키텍처 인 경우에도 게임이 에뮬레이터 내에서 실행될 수 있도록 게임 콘솔을 모방하고 있다.

QEMU 는 대표적인 에뮬레이터 중 하나로써 매우 다양한 종류의 하드웨어를 소프트웨어로 구현해둔 Hypervisor 다.

https://www.linux.com/news/emulation-or-virtualization-which-right-you/

https://ko.wikipedia.org/wiki/%EC%97%90%EB%AE%AC%EB%A0%88%EC%9D%B4%ED%84%B0

https://theithollow.com/2012/03/07/virtualization-vs-emulation/

Differences Between Emulation vs. Simulation

Criteria	Simulation	Emulation
Provided by	Device manufacturers and other companies.	Device manufacturers.
Target area	Internal behavior of the mobile device.	Mobile device hardware, software, and operating system.
Internal structure	Written in high-level language.	Written in machine-level assembly language.
Used/suitable for	Unit testing, automation testing.	Unit testing, automation testing, debugging.
Performance	Faster compared to emulators.	Slower due to latency since it involves binary translation.
Reliability	Low, as it cannot simulate all types of user interactions.	Low, as it cannot simulate all types of user interactions.

참고 Simulation?> 에뮬레이션은 호스트 머신에 존재하지 않는 하드웨어 및 아키텍처를 가상머신에게 제공, 시뮬레이션은 호스트 머신에 존재하는 하드웨어 및 아키텍처를 이용하여 가상머신에 제공하는 것이다.

https://medium.com/naver-cloud-platform/%EB%84%A4%EC%9D%B4%EB%B2%84%ED%81%B4%EB%9D%BC%EC%9A%B0%EB%93%9C-%EA%B8%B0%EC%88%A0-%EA%B2%BD%ED%97%98-%EA%B0%80%EC%83%81%ED%99%94-%EA%B0%9C%EB%85%90-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-1-qemu-vs-kvm-962113641799

3) 가상화(Virtualization)와 하이퍼바이저(Hypervisor)

가상화는 간단히 말하자면 다른 호스트 OS 위에 게스트 OS 를 실행하는 것이다. **가상머신**(Virtual Machine)은 실제 환경을 복제할 수 있는 격리된 환경이다. 호스트는 하드웨어에서 기본적으로 실행되는 운영 체제이며, 게스트는 VM 내부의 호스트 위에서 실행되는 운영 체제이다. 특정 응용 프로그램이나 서비스를 실행하는 데 필요한 기본 환경을 제공하는 것이 바로 게스트 OS 이다.

가상머신은 CPU 자체 가상화를 활용하여 실제 하드웨어에 가상화 된 인터페이스를 제공하고, 에뮬레이터는 코드를 직접 실행할 수 있는 CPU 에 의존하지 않고 하드웨어를 에뮬레이션하고 일부 작업을 가상 컨테이너를 제어하는 하이퍼 바이저로 리디렉션한다.

https://stackoverflow.com/questions/6234711/what-are-the-specific-differences-between-an-emulator-and-a-virtual-machine , https://www.quora.com/What-are-the-differences-between-an-emulator-and-a-virtual-machine

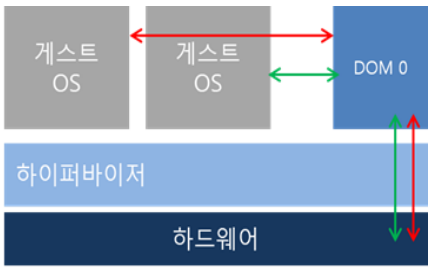
(1)가상화의 분류

https://library.gabia.com/contents/infrahosting/7426/

전가상화와 반가상화의 중요한 차이는 하이퍼 바이저, OS, 하드웨어의 관계이다.

전가상화(Full-virtualization, Native-virtualization)

동일한 아키텍처에서 실행되는 Guest OS 를 수정없이 그대로 띄워주는데 필요한 하드웨어를 모두 에뮬레이션 하는 형식으로 가상화 하는 것을 말한다. 쉽게 말해 CPU 를 뺀 모든 하드웨어를 가상으로 구현한다. 하드웨어 에뮬레이터을 하이퍼바이저(Hypervisor)라 부른다. Oracle VM VirtualBox, VMware Workstation, VMware Server 등이 이에 속한다.



전 가상화는 하드웨어를 모두 가상화하는 방식입니다. 게스트 OS 가 하이퍼바이저에게 하드웨어 시스템 제어를 요구하면, 하이퍼바이저는 하드웨어에게 해당 요구 사항을 전달한다.

각 게스트 OS 는 'DOM 0'라는 관리 머신을 거쳐 하이퍼바이저와 통신하므로, CPU 와 RAM 처럼 I/O 가 잦은 자원을 컨트롤 하기에는 번거롭다. 이런 한계를 개선하기 위해 반가상화 방식이 출현하게 되었다.

반가상화(Para-virtualization)

하드웨어 에뮬레이션 없이 하이퍼바이저(Hypervisor)를 통해 하이퍼바이저가 제공하는 API 를 이용한 OS 를 실행할 수 있는 가상화 방식이다. Guest OS 를 하이퍼바이저의 API 를 이용하도록 수정해야한다. VMware ESX Server, Xen, Solaris xVM Server, Sun Logical Domains, Linux KVM 이 이에 속한다. 성능 저하가 크지 않아 각광받고 있는 기술이다.

반 가상화 방식은 하드웨어를 완전히 가상화하지 않다.
대신 게스트 OS 의 커널을 일부 수정해 하드웨어와 인터랙션할 수 있도록 합니다. OS 를 수정해야 하므로 게스트 OS 가 윈도우일 경우 별도의 Tool 을 이용해야 해서 번거로움이 있다.

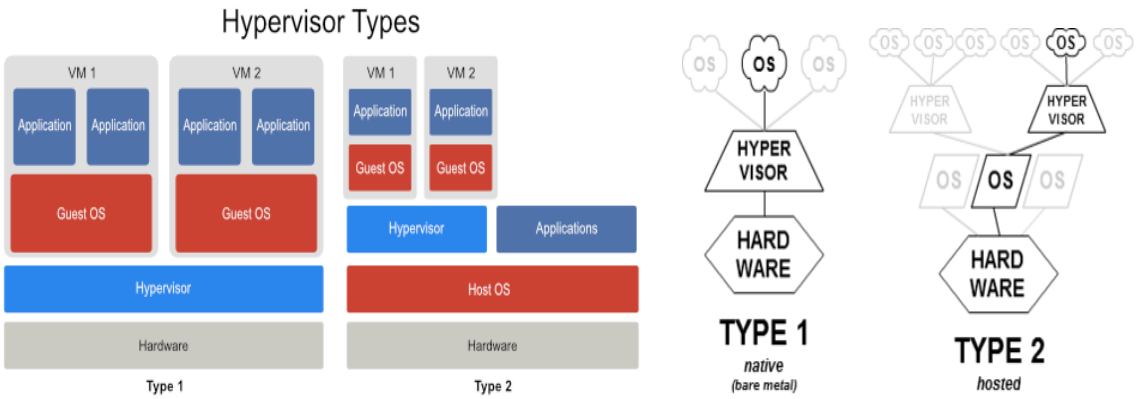
대표적인 반가상화 방식 소프트웨어인 Xen 에서 게스트 OS 는 Hyper Call 이라는 명령어를 통해 하드웨어에 필요한 자원을 바로 하이퍼바이저에 전달하고, 하이퍼바이저는 바로 하드웨어를 제어한다. 즉, 각각의 게스트 OS는 필요한 자원을 직접 요청할 수 있는 능력이 있으므로 모든 요청을 'DOM 0' 가 한꺼번에 처리하여 하드웨어를 제어하는 전가상화 방식에 비해 성능이 좋다.

- 참고로 Xen 은 type 1 과 type2 모두를 지원하므로 아래의 자료를 참고하면 좋습니다.
  http://lass.cs.umass.edu/~shenoy/courses/spring08/lectures/Lec05.pdf
- (*Hardware Assitsted 는 생략, 자세한 내용은 Vmware 공식문서 참고
  https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf)

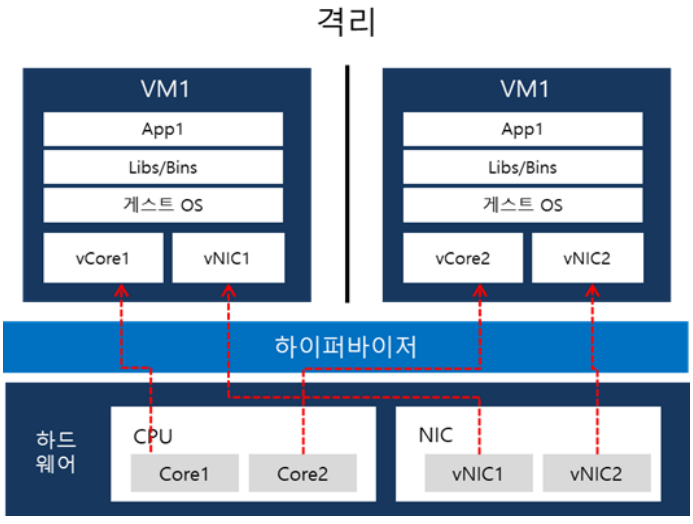
(2) 하이퍼바이저

하이퍼바이저(Hypervisor)는 호스트컴퓨터와 게스트 OS 를 연결해주는 가상화 플랫폼이다. 여러 개의 VM 이 한정된 하드웨어 자원을 나눠쓸 수 있게 해준다. 크게 Type 1 과 Type 2 로 나뉜다.

- Type 1: 하드웨어에서 바로 실행된다. (OS 와 합쳐진 상태) 보통 Type 1 은 Para-virtualization 를 지원할 수 있다. Vmware 의 vSphere ESXi 제품군(OS 설치없이 하이퍼바이저만 베어메탈 서버에 설치)을 예로 들 수 있다.
- Type 2: OS 상에서 실행된다. 운영체제에 많은 기능을 의존하여 하드웨어 자원을 분배한다. Type 2 는 Full-virtualization 를 지원한다. 예를 들어 윈도우나 리눅스를 설치하고 그 위에 VirtualBox 나 KVM 을 설치하는 경우를 말한다.



하이퍼바이저에 의해 구동되는 VM 은 각 VM 마다 독립된 가상 하드웨어 자원을 할당받는다. 논리적으로 분리되어 있어서 한 VM 에 오류가 발생해도 다른 VM 으로 퍼지지 않는다는 장점이 있다.



4) OS 레벨 가상화(OperatingSystem-level virtualization)

커널이 여러 격리된 사용자 공간 인스턴스의 존재를 허용하는 운영 체제 패러다임. 이러한 인스턴스를 **containers** (LXC, Solaris containers, Docker), Zones (Solaris containers), virtual private servers (OpenVZ), partitions, virtual environments (VEs), virtual kernels (DragonFly BSD), or jails (FreeBSD jail or chroot jail) 라고 부른다. -이하 컨테이너로 지칭-

일반 운영체제에서 실행되는 컴퓨터 프로그램은 해당 컴퓨터의 모든 리소스(연결된 장치, 파일 및 폴더, 네트워크 공유 , CPU 성능, 수량화 가능한 하드웨어 기능)를 볼 수 있으나, 컨테이너 내부에서 실행되는 프로그램은 컨테이너에 할당된 컨테이너의 내용과 장치만 볼 수 있다.

https://en.wikipedia.org/wiki/OS-level_virtualization

최근 가상화 기술의 축은 **하이퍼바이저 기반의 가상화에서 컨테이너 기반 가상화**로 이동하고 있다.

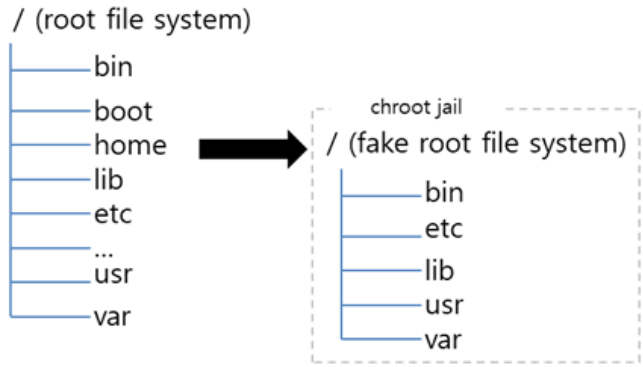
(1)컨테이너화의 원리

컨테이너 기술은 LXC(Linux Container)에서부터 출발한다. LXC는 호스트 OS에서 프로세스 간 벽을 만드는 기술로, 네임 스페이스와 cgroup을 조합한 형태이다.

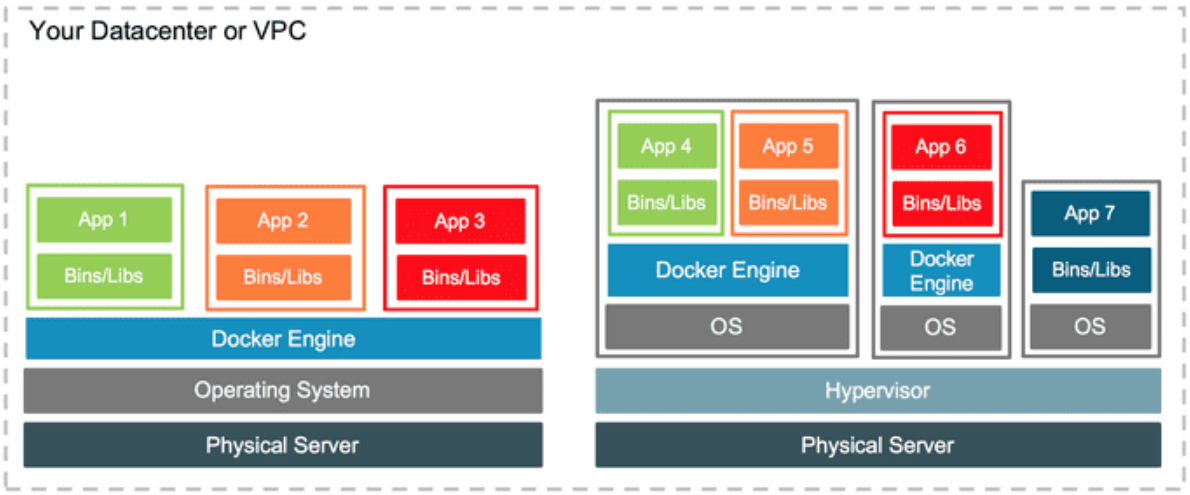
https://library.gabia.com/contents/infrahosting/7426/

- chroot: 특정 디렉토리를 루트(최상위 디렉토리)로 인식하게끔 하는 명령어
- 네임 스페이스: 리눅스 시스템 자원을 묶어 프로세스에 할당하는 방식으로, 하나의 프로세스 자원을 관리하는 기능
- cgroup: CPU, 메모리 등 프로세스 그룹의 시스템 자원 사용량을 관리하여 특정 애플리케이션이 자원을 과다하게 사용하는 것을 제한

먼저 chroot 를 통해 특정 파일 디렉토리가 최상위 계정(Root)으로 인식되도록 한 후, cgroup 과 네임스페이스를 통해 특정 프로세스에 자원을 할당하고 제어하는 방식이다.



(2) VM 과 Container 의 비교



(Docker 공식 블로그에 기재되어있는 예시 그림과 비유 예제를 활용)

https://www.docker.com/blog/containers-and-vms-together/

https://www.docker.com/blog/containers-are-not-vms/

주택 (**VM**)과 아파트 (**Container**)를 비교해보자.

주택 (**VM**)은 완전히 독립적이며 원치 않는 게스트로부터 보호합니다. 그들은 또한 배관, 난방, 전기 등 자체 인프라를 보유하고 있습니다. 또한 대부분의 경우 주택에는 최소한 침실, 거실, 욕실 및 주방이 있어야합니다. 나는 아직 "스튜디오 하우스"를 찾지 못했습니다. 가장 작은 집을 산다고해도 집이 지어지는 방식이기 때문에 필요한 것보다 더 많이 사게 될 수도 있습니다.

아파트(**Container**)도 원치 않는 게스트로부터 보호를 제공하지만 공유 인프라를 기반으로 구축됩니다. 아파트 건물 (Docker Host)은 배관, 난방, 전기 등을 공유합니다. 또한 아파트는 스튜디오에서 멀티 베드룸 펜트 하우스까지 모든 종류의 다양한 크기로 제공됩니다. 필요한 것만 렌트하고 있습니다. 마지막으로, 집과 마찬가지로 아파트에는 원치 않는 손님을 차단하기 위해 잠글 수있는 현관 문이 있습니다.

컨테이너를 사용하면 Docker 호스트의 기본 리소스를 공유하고 애플리케이션을 실행하는데 정확히 필요한 이미지를 빌드합니다. 기본부터 시작하여 필요한 것을 추가합니다. VM 은 반대 방향으로 구축됩니다. 전체 운영 체제로 시작하고 애플리케이션에 따라 원하지 않는 항목을 제거할 수 있습니다.

VM 과 비교했을 때 컨테이너는 하이퍼바이저와 게스트 OS 가 필요하지 않으므로 더 가볍다. 일반적으로 컨테이너에는 OS 가 포함되지 않아 크기가 수십 MB 에 불과하고, 운영체제 부팅이 필요 없으므로 서비스를 시작하는 시간도 짧다. 크기가 작기 때문에 컨테이너 복제와 배포에도 용이하다.

애플리케이션을 실행할 때에도 물리 서버에서 애플리케이션 하나를 실행하는 것과 마찬가지로, 호스트 OS 위에 애플리케이션의 실행 패키지인 '이미지'를 배포하기만 하면 된다.

반면, VM 은 항상 게스트 OS 가 포함되므로 보통 수 GB 에 해당하고, 애플리케이션을 실행할 때에도 먼저 VM 을 띄우고 자원을 할당한 다음, 게스트 OS 를 부팅하여 애플리케이션을 실행시켜야 한다.

이처럼 VM 에 비해 컨테이너는 애플리케이션을 실행, 배포하는 과정이 가볍기 때문에 하나의 물리 서버에서 더 많은 애플리케이션을 구동시킬 수 있어 가장 주목받는 서버 가상화 기술로 여겨지고 있다.

그 외 Reference

https://opensource.com/resources/virtualization

개발자도 궁금한 IT 인프라 – 정송화, 김영선, 전성민

Virtualization and Forensics

https://en.wikipedia.org/wiki/Virtualization

https://en.wikipedia.org/wiki/Hypervisor

https://www.networkworld.com/article/3243262/what-is-a-hypervisor.html

https://www.redhat.com/ko/topics/virtualization/what-is-virtualization

https://www.ibm.com/cloud/learn/virtualization-a-complete-guide

https://www3.technologyevaluation.com/sd/category/virtualization-virtual-machine/articles/virtualization-vs-emulation-vs-simulation

https://www.computerworld.com/article/2551154/emulation-or-virtualization-.html