물리적 서버

가상 서버

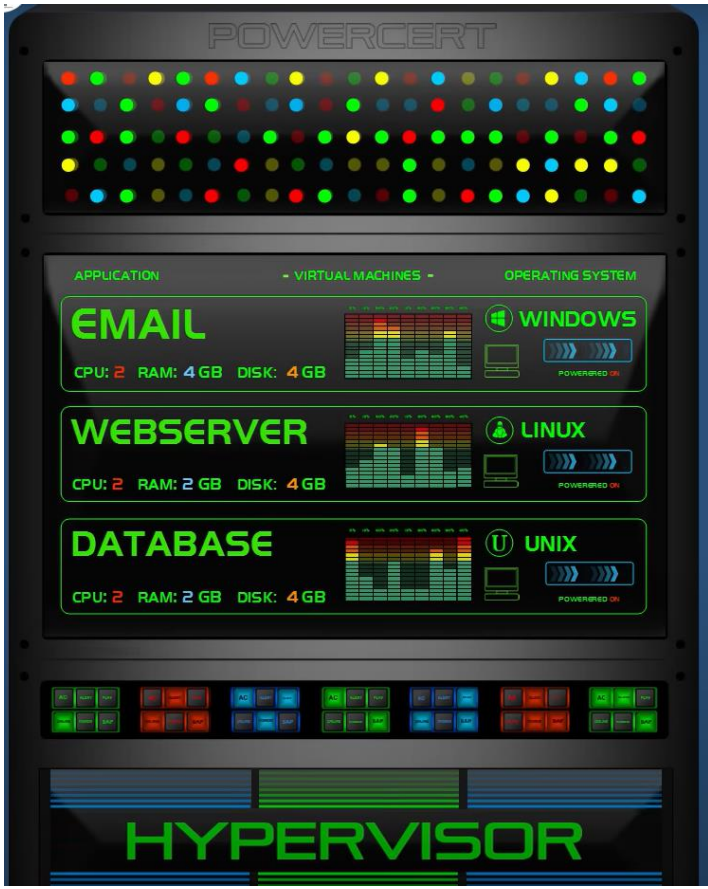
Container

물리 서버

안정성을 위해 한 서버에 한 개의 Application을 운영

물리 서버의 한계
- IDC의 공간 차지
- 전기, Cooling 시스템
- 활용성의 한계
- 배포(설치 및 구성) 및 운영하는데 시간이 많이 걸린다

가상 서버



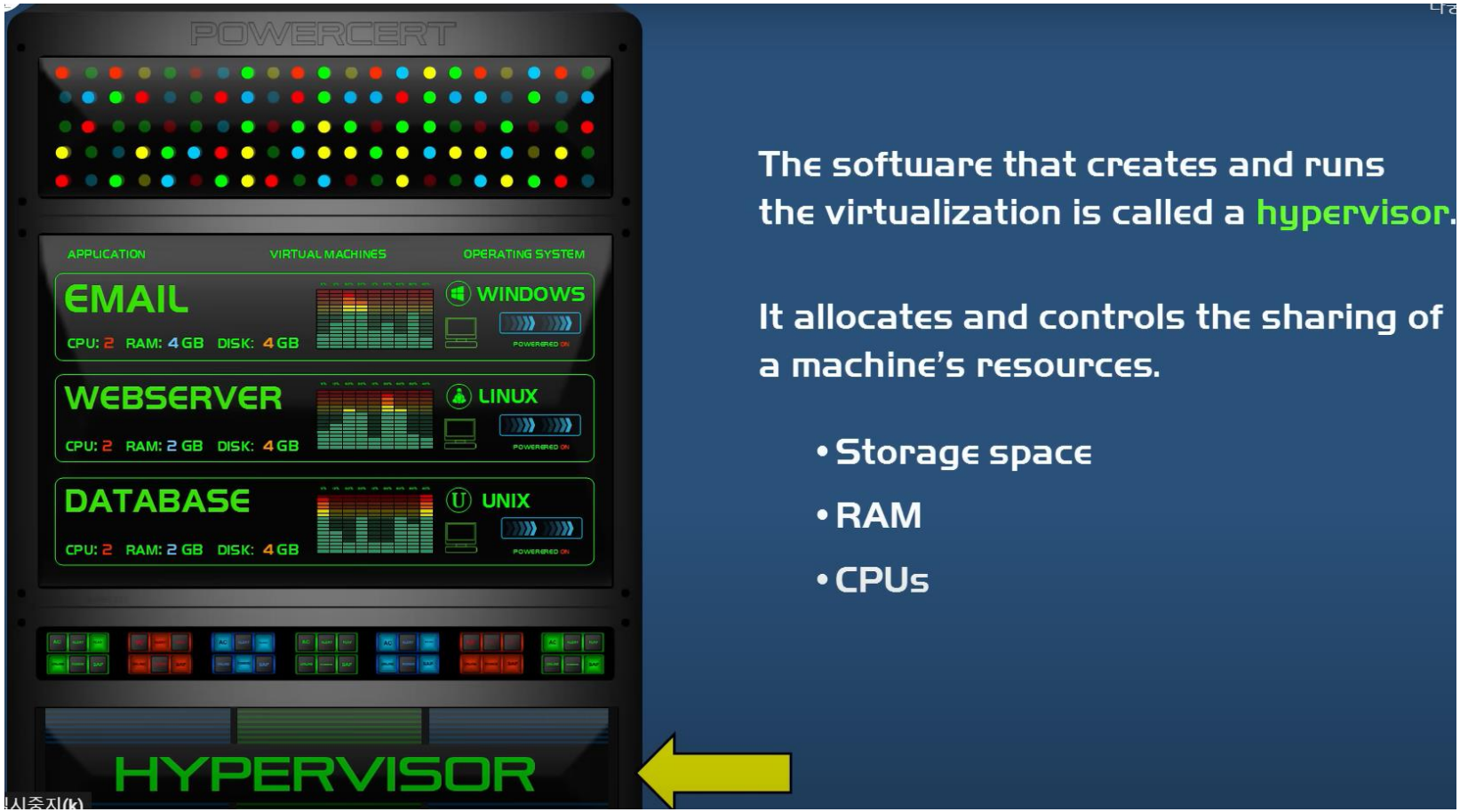
This one server is running
3 VMs (virtual machines).

Running 3 different applications.

Running 3 different operating systems.

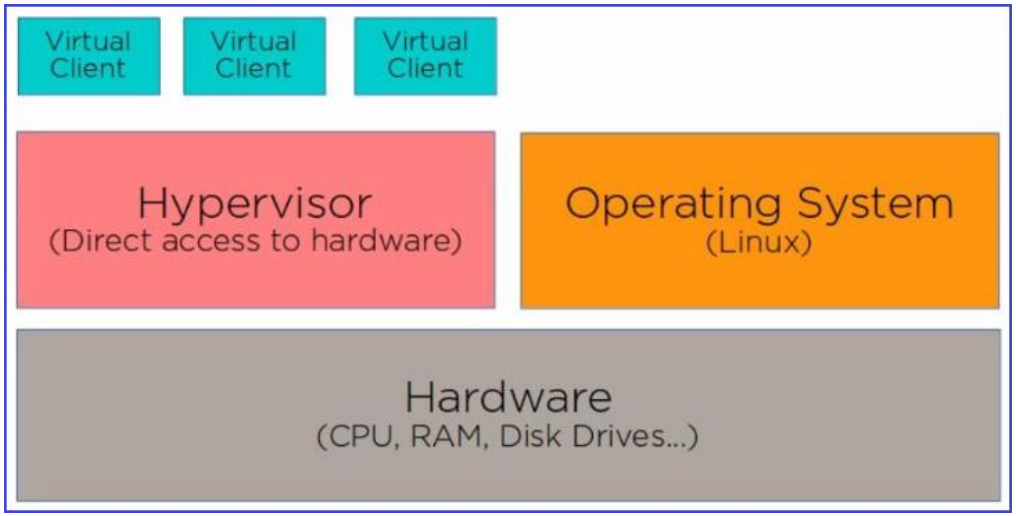
물리 서버에 Hypervisor를 설치하면
여러 개의 VM을 운영할 수 있고,
각 VM에 하나의 Application을 운영

가상 서버


POWERCERT
APPLICATION
VIRTUAL MACHINES
OPERATING SYSTEM
EMAIL
CPU: 2 RAM: 4 GB DISK: 4 GB
WINDOWS
WEBSERVER
CPU: 2 RAM: 2 GB DISK: 4 GB
LINUX
DATABASE
CPU: 2 RAM: 2 GB DISK: 4 GB
UNIX
HYPERVISOR
The software that creates and runs the virtualization is called a hypervisor.
It allocates and controls the sharing of a machine's resources.
• Storage space
• RAM
• CPUs

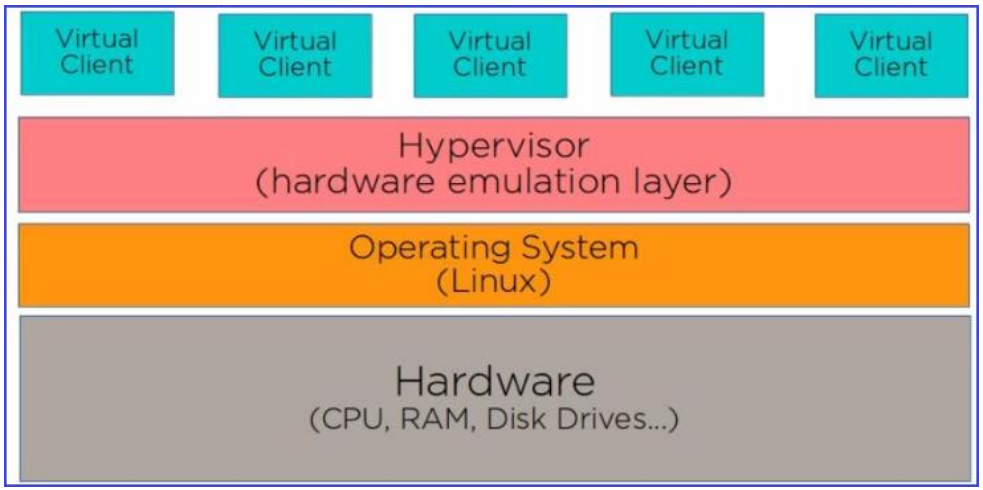
1-서버 가상화란?

- Full Virtualization
  - **Type 1 Hypervisor**: Native hypervisor(Bare-metal hypervisor)
    - Type 1 can provide **excellent stability and performance**.
    - H/W에 대한 Driver를 VM이 가지고 있다
      - VMware vSphere with ESX/ESXi
      - Kernel-based Virtual Machine (KVM)
      - Microsoft Hyper-V
      - Oracle VM
      - Citrix Hypervisor



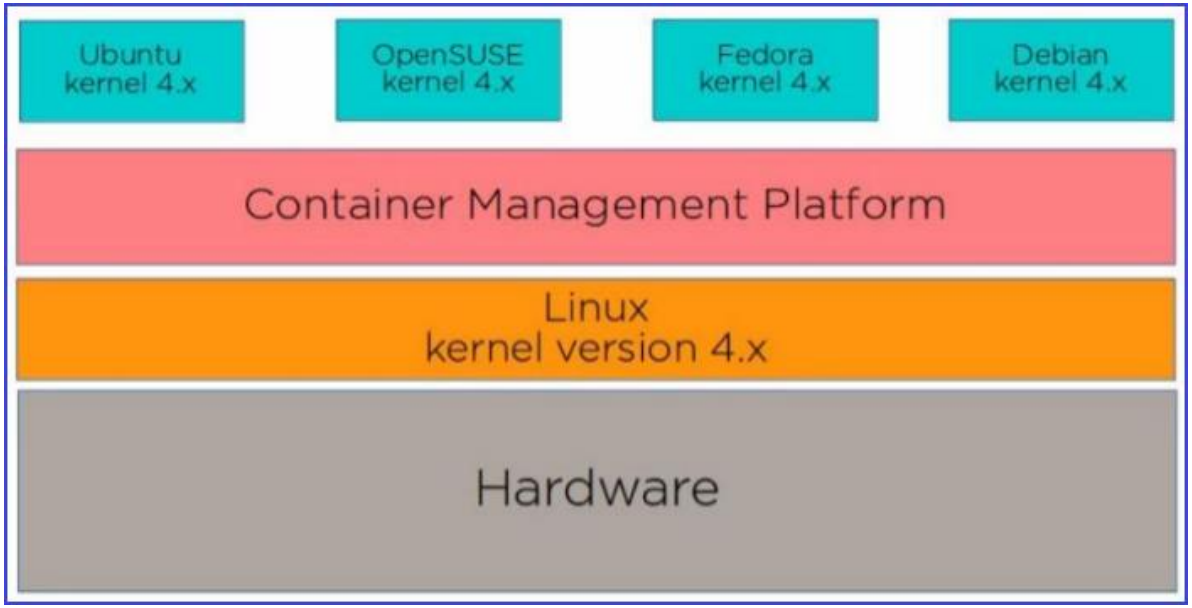
1-서버 가상화란?

- Full Virtualization
  - **Type 2 Hypervisor:** Hosted hypervisor
    - Type 2 is installed **inside the operating system** of the host machine.
    - Unlike the Type 1 hypervisor, this one has one software layer underneath.
    - H/W에 대한 Driver를 VM이 가지고 있지 않다
    - VM이 H/W에 액세스할 때 Host OS가 H/W를 가상화시켜준다**(Emulation)**
      - Oracle VM VirtualBox
      - VMWare Workstation Pro/VMWare Fusion
      - Parallels Desktop



1-서버 가상화란?

- 가상화 방식
  - **OS-level Virtualization** (Kernel-dependent Container)
    - Container들이 Host OS의 kernel을 공유한다
    - VM의 Application 비해 **성능이 좋고, 이식성이 좋다**
    - OS-level virtualization doesn't use a hypervisor and doesn't apply a host-guest paradigm. Instead, it utilizes a process called "containerization" which creates multiple user-space instances (containers or virtual environments) through a kernel in the OS.

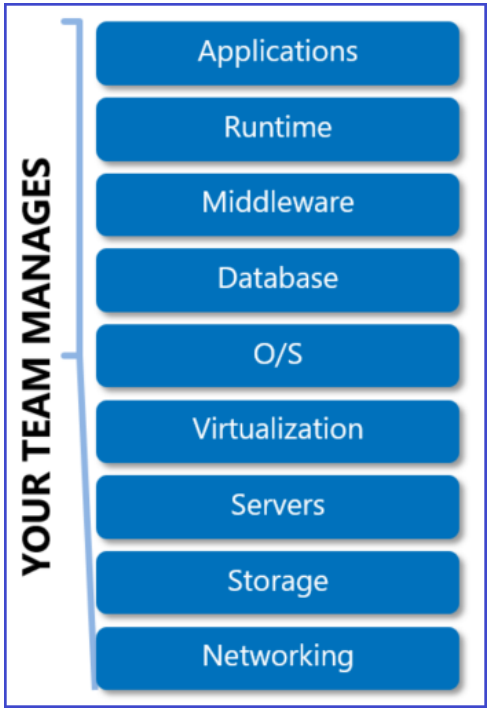


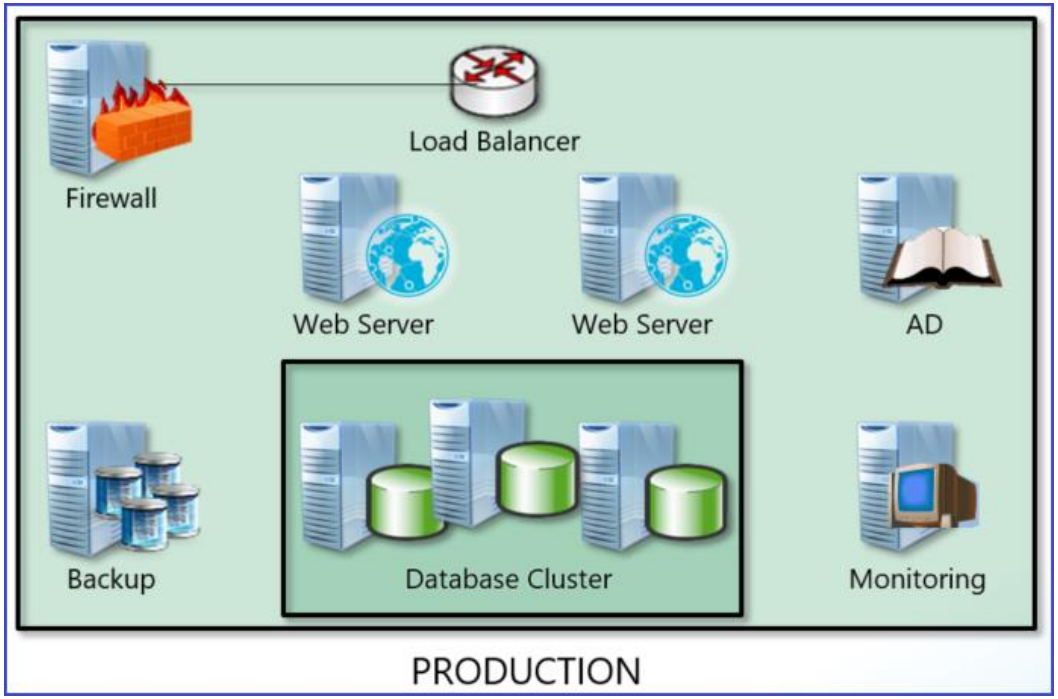
1-서버 가상화란?

- 서버 가상화의 장점
  - 에너지 절약, 녹색 성장
  - 데이터센터의 규모 축소
  - 개발 및 테스트 작업에 효율적
  - 신속한 Server Provisioning
  - 장비 업체에 대한 종속 방지
  - 서버 가동시간의 증가
  - 재난 복구 향상
  - 애플리케이션 독립
  - 오래된 애플리케이션의 수명 연장
  - Cloud로 이동 가능

가상 서버

- Application으로 서비스하는 것이 우리의 목표
  - 회사 업무를 지원하기 위해서 이메일, 포탈, 협업, 제조 시스템, 고객관리시스템을 준비하기 위해 Database server, Web server가 필요하다
  - 이를 위해서 Networking, Storage, Server H/W, Server S/W(OS), Virtualization 등등의 Infrastructure가 추가적으로 필요하다





가상 서버

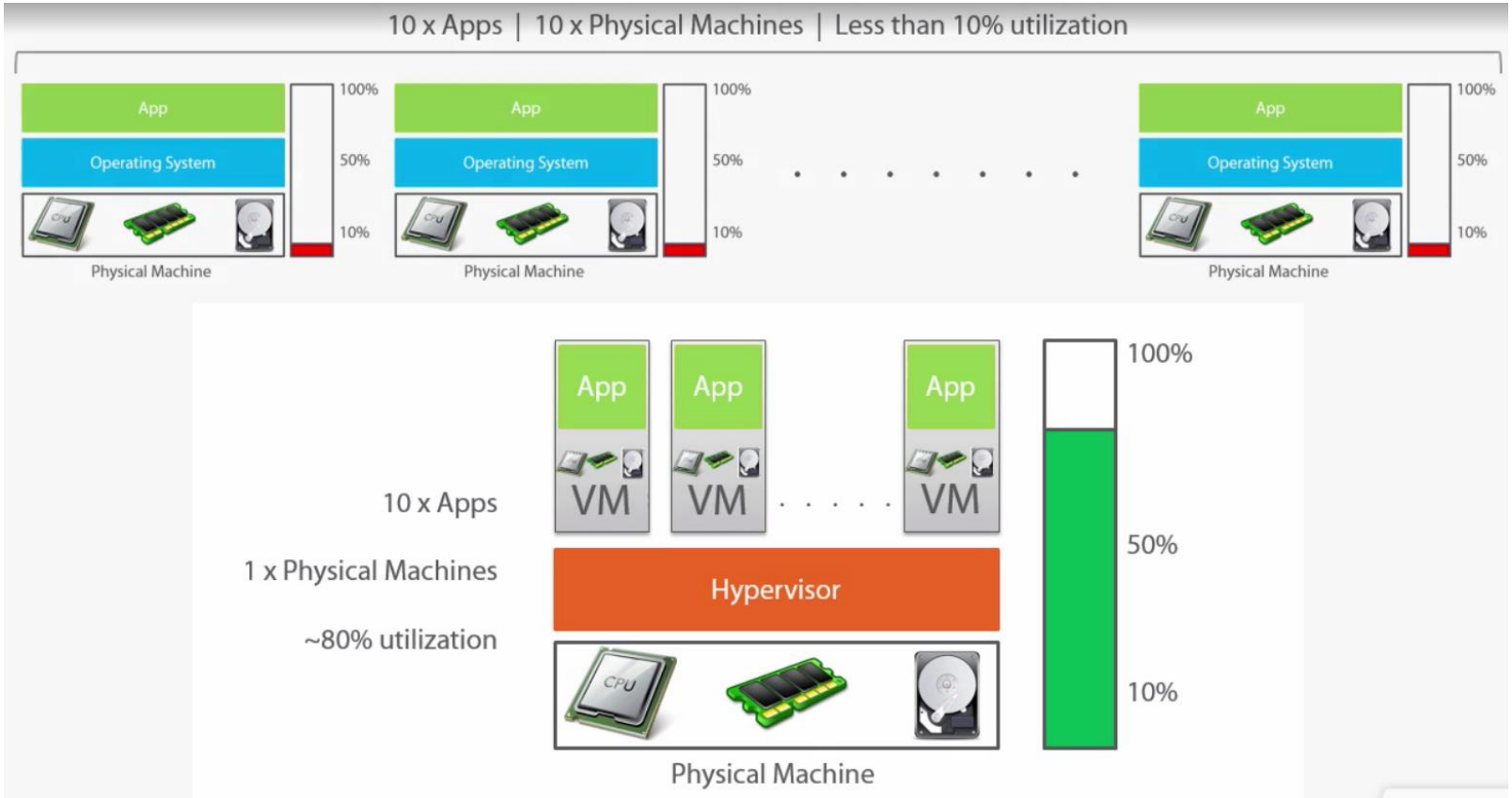
- Application으로 서비스하는 것이 우리의 목표
  - 이렇게 구성하기 위해서는 다음과 같은 문제점들이 있다
    - 서버들을 확장하기 위해 전용 H/W 및 S/W **필요**
    - 추가적으로 여러 가지 환경(Prod, Staging, Test, Dev) **필요**
    - 서버 이용률이 매우 낮음(10~20%)
      - 한 서버에 하나의 서비스만 운영하는 것이 일반적이다
    - 서버 확장을 위해서는 모든 것이 준비되어 있어야 하며, 또한 서비스의 일시 중지가 요구되므로 유연하지 못함

가상 서버

- Application으로 서비스하는 것이 우리의 목표
  - 이것을 해결한 것이 바로 **하나의 물리적인 Server에 여러 개의 Virtual Machine을 운영하는 것**이다
  - 다양한 종류의 Hypervisor 기술을 사용하여 다양한 OS에 다양한 Application(=Service)를 제공하여 **자원 활용률을 증가시킨다**

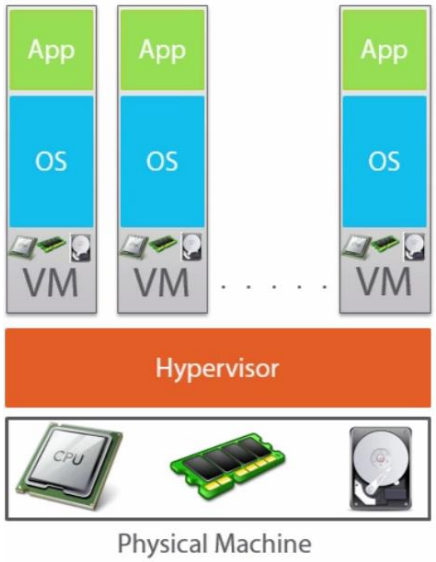
가상 서버

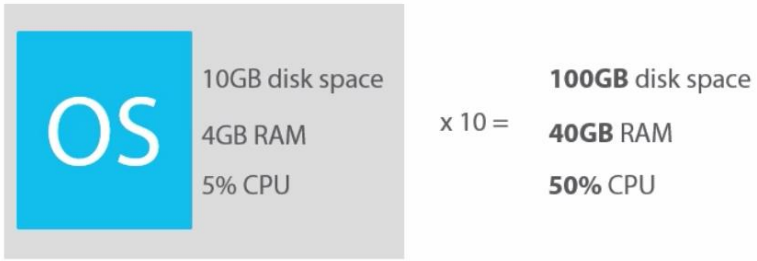
- Application으로 서비스하는 것이 우리의 목표
  - 10% 정도의 자원 활용률을 높이기 위해 Virtualization 기술을 사용하면 효율적으로 자원을 이용하여 서비스할 수 있다



가상 서버

- Application으로 서비스하는 것이 우리의 목표
  - 하지만 VM이 많아진다는 것은 **관리해야 할 Server가 많아지는 것과 동일하다**
  - VM으로 서비스를 하면
    - 물리적인 서버(Host Machine)에 비하여 **성능이 떨어진다**
    - Host Machine의 고장으로 인한 피해를 줄이기 위해 **Clustering 기술이 필요**하며, 당연히 **SAN Storage가 필요하게 되어 비용이 증대된다**
    - 이러한 VM을 효과적으로 관리하기 위해서는 **추가적인 소프트웨어 구매가 필수적이어서** 계속해서 관리 비용이 많이 드는 결과를 초래한다

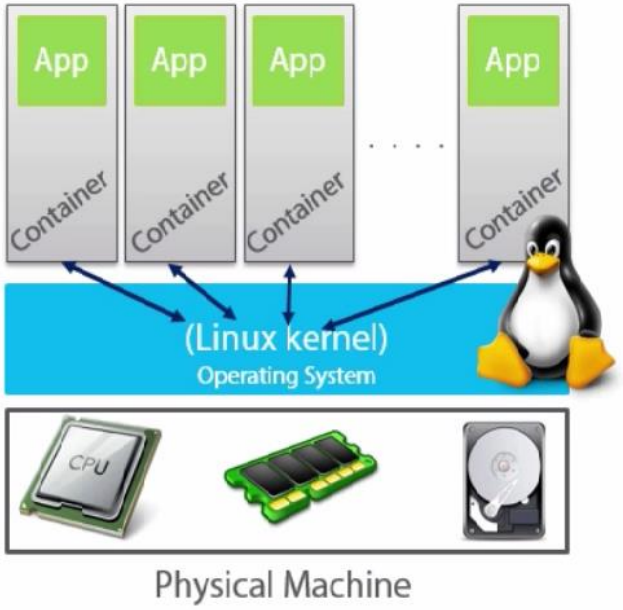




Numbers and values shown here are for illustration purposes only. Real world values will differ.

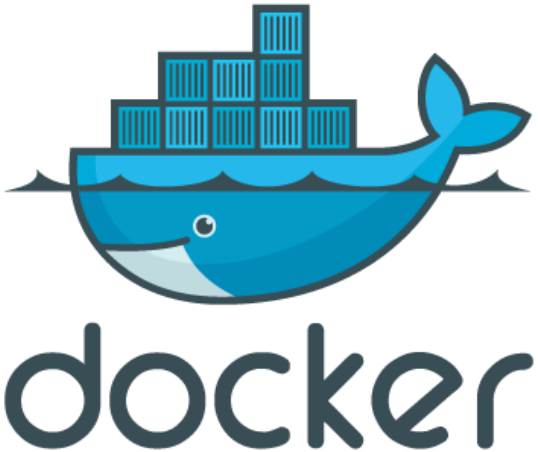
Container

- Application으로 서비스하는 것이 우리의 목표
  - 초기 구입 및 관리 비용을 줄이면서 효과적으로 Application을 서비스하기 위해서 VM보다 가벼운 Container 기술을 이용하는 것이 Docker다
  - Linux Kernel을 공유하는 Container 기술을 사용하면 VM에 비하여 CPU, RAM, Disk를 덜 사용하게 된다
    - Web Service를 서비스할 때 VM에서 운영하는 것과 Container에서 제공할 때 CPU 사용율과 Memory 사용량을 확인해 보면 알 수 있다



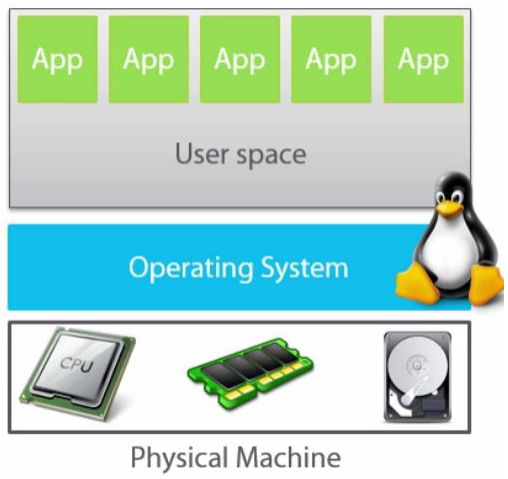
Container

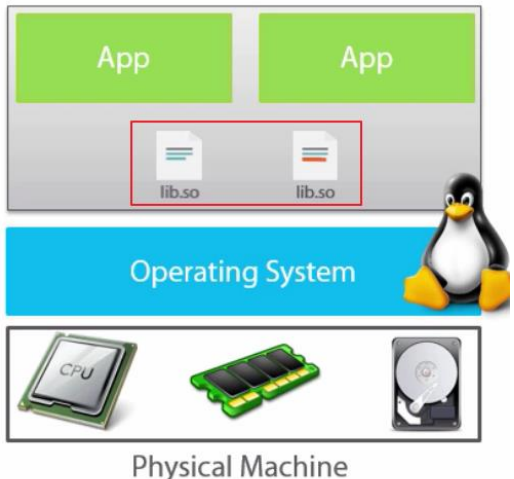
- Container란?
  - 배(ship)에 물건을 선적하여 운행할 때, 폭풍우를 만나면 물건끼리 부딪혀서 파손되기도 한다
  - 개인의 물건들을 안전하게 배송하려면 각 물건을 Container에 넣어서 Container를 Dock에 선적하여 운행하면 된다
  - Container는 서로의 영역을 침범하지 않는 독립적인 공간을 제공하는 것이다
  - Linux도 이러한 Container 기술을 사용하여 Application간의 충돌을 방지하고, 각 Container들은 공통의 자원인 CPU, RAM, Disk, Network를 사용하여 쉽고 빠르고 자원 활용률을 증대하여 서비스를 한다

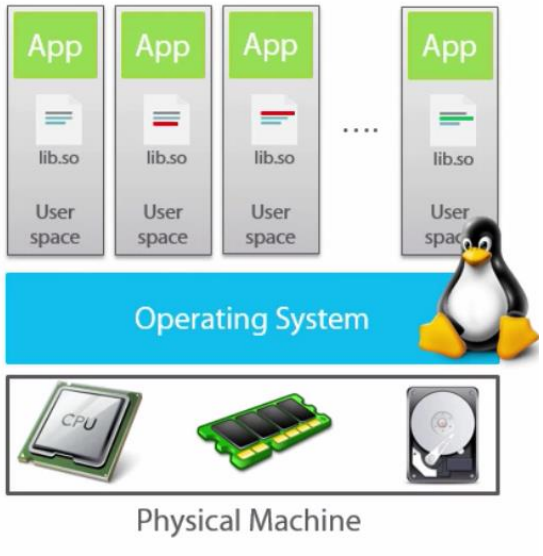


Container

- Container란?
  - 한 운영 체제의 하나의 User space 상황에서 .dll 파일을 사용하는 여러 개의 Application이 실행되는 경우가 있다
  - 이 때 각각 Application의 이름은 동일하지만 Version이 다른 경우라면 난감하게 된다
  - 이러한 상황을 해결한 것이 바로 Application마다 서로 방해가 되지 않도록 하는 독립적인 실행 환경을 제공하면 된다
  - 즉, 각각의 Application마다 각각의 고유한 User space를 제공하여 각각에 필요한 Version의 .dll 파일을 사용하도록 하면 된다.
  - 이것이 바로 Container 기술이다

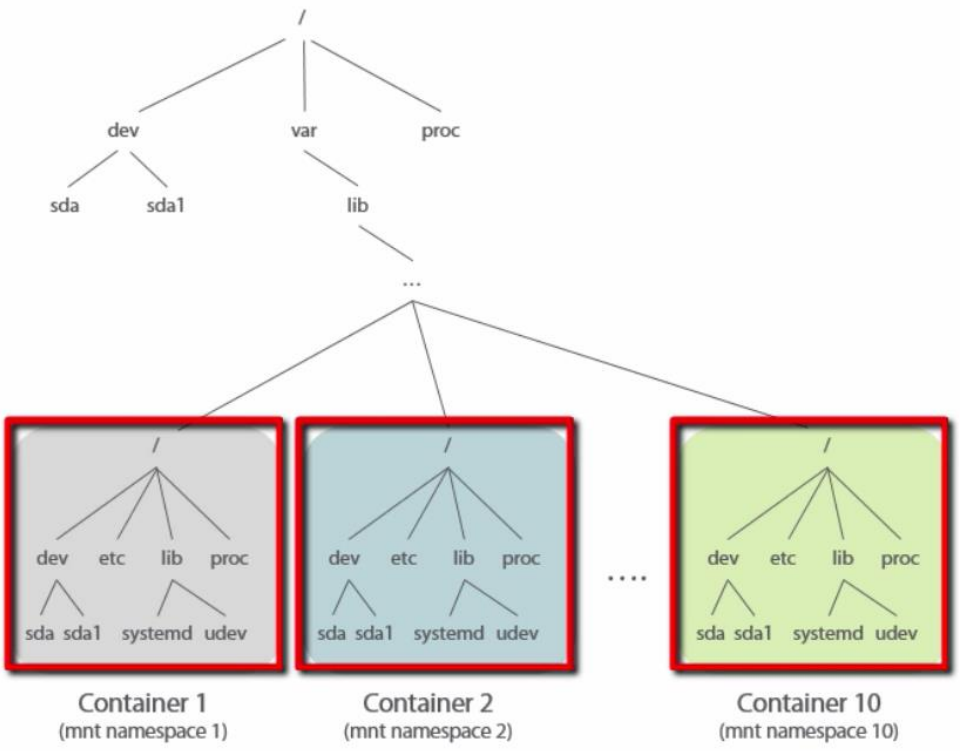


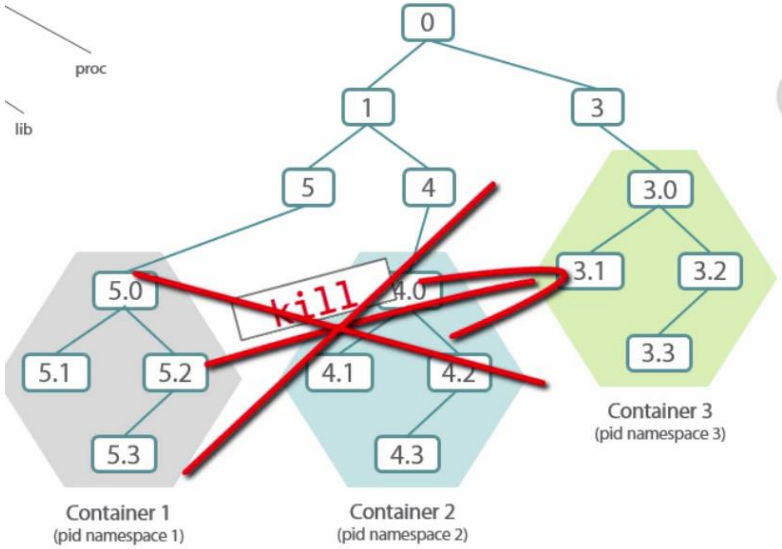




Container

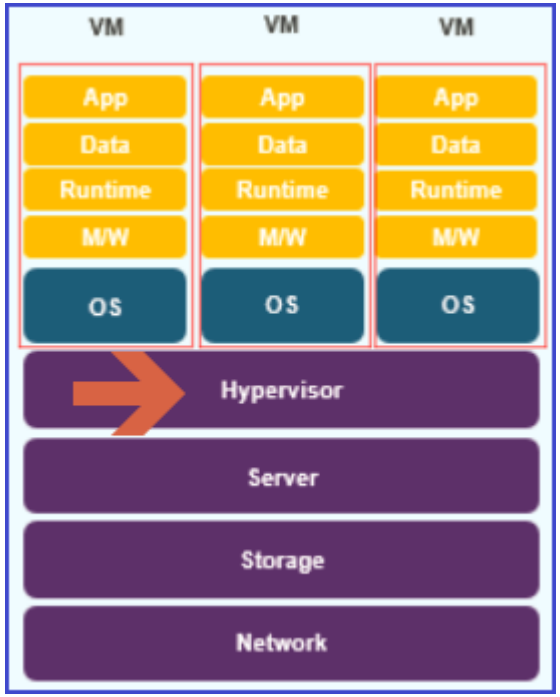
- Container란?
  - 각 Container들은 OS의 Kernel을 공유한다
  - 각 User Space(=Container) 단위로 root filesystem을 갖도록 하는 것이 container 기술이다
  - 각 Container 단위로 systemd를 갖도록 하면 process간에 kill하지 못한다

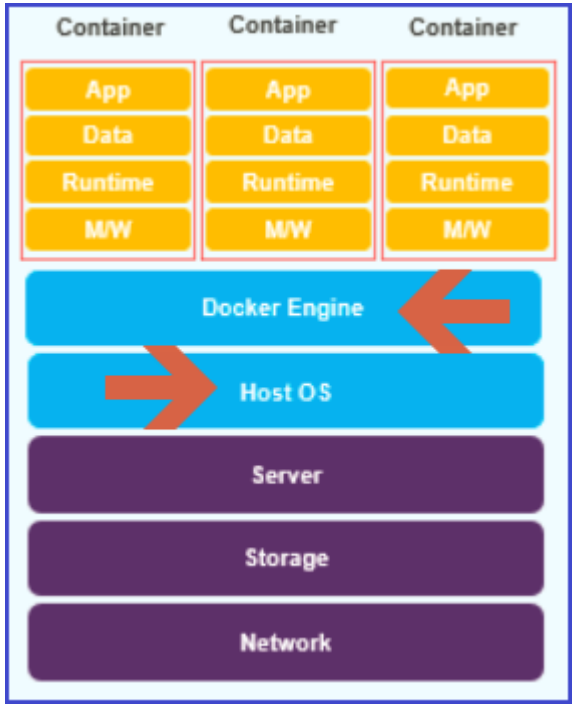




가상 컴퓨터와 Container 비교

- VM과 Container 비교
  - VM
    - Hypervisor로 하드웨어를 가상화하고, Hypervisor 위에서 Guest OS가 설치된 VM들을 구동한다
  - Container
    - OS 레벨에서 CPU, RAM, Disk, Network 등의 자원을 격리하여 Container에 할당하기 때문에 Guest OS(VM)가 따로 필요 없다





가상 컴퓨터와 Container 비교

- VM과 Container 비교
  - 효율성
    - 기업환경에서는 안정적인 운영을 위해, 1개의 Virtual Machine에 1개의 서비스를 구동하는 것을 권장한다. 이의 경우 VM의 모든 자원을 사용하는 것이 아니기 때문에 성능적 오버헤드가 발생한다
    - 반면 Container의 경우, OS 커널을 공유하기 때문에 자원을 필요한 만큼 효율적으로 사용할 수 있다
  - 신속성
    - 사용자의 서비스 요청량이 증가함에 따라, 기업에서는 VM이나 Container를 추가적으로 배포한다
    - VM의 크기는 최소 몇 GB이지만, Container인 경우 Guest OS가 없어 MB단위의 크기를 가진다
    - 결과적으로 VM은 배포하는데 수 분에서 수십 분의 시간이 소요되지만, Container는 배포에 소요되는 시간이 수 초에 불과하다

가상 컴퓨터와 Container 비교

- VM과 Container 비교
  - 라이센스 비용 절감
    - 가상화 서버의 경우 VM의 개수만큼 Guest OS의 라이센스 비용이 발생한다
    - 반면에 Container인 경우 Host OS 1대의 라이센스 비용만 발생한다. 만약 서버의 수가 많아진다면 비용의 차이도 기하급수적으로 증가할 것이다
  - 안정성
    - VM의 경우 정확히 할당된 자원 내에서 VM이 운영되기 때문에, Container에 비해 안정적으로 운영할 수 있다
    - 반면에 Container들은 OS 커널을 공유하기 때문에, 하나의 Container가 무리하게 자원을 사용하게 될 수 있다
    - 이를 해결하기 위해서 실행할 Container에 CPU, Memory와 같은 자원 할당량을 사전에 지정시킬 수 있다
    - 또는 Orchestration을 하는 Docker Swarm, Kubernetes으로 해결할 수 있다

VM과 Docker 비교하기

- VM vs. Docker

조건	VM	Docker
메모리 필요량	많은 메모리 양 필요	적은 메모리 양 요구
부팅 시간	오랜 부팅 시간	부팅 시간이 엄청 빠름
성능	많은 VM 실행으로 불안정한 성능	성능 좋음
확장 여부	Scale out 어려움	Scale out하기 쉬움
효율성	효율성이 낮음	효율성이 높음
이동성	다른 Platform으로 이동하면 호환성 문제 발생	다른 Platform으로 이동 가능
공간 할당	데이터 볼륨의 공유가 어려움	데이터 볼륨이 공유되고, 다른 컨테이너도 같이 사용할 수 있다