Virtualization of IT resources

●●●

Virtual machines and containers

Hélène Coullon, Associate prof., IMT Atlantique, Inria, LS2N - helene.coullon@imt-atlantique.fr
Jonathan Pastor, postdoc, IMT Atlantique, Inria, LS2N - jonathan.pastor@imt-atlantique.fr

Virtual machines

Principles of virtualization with virtual machines

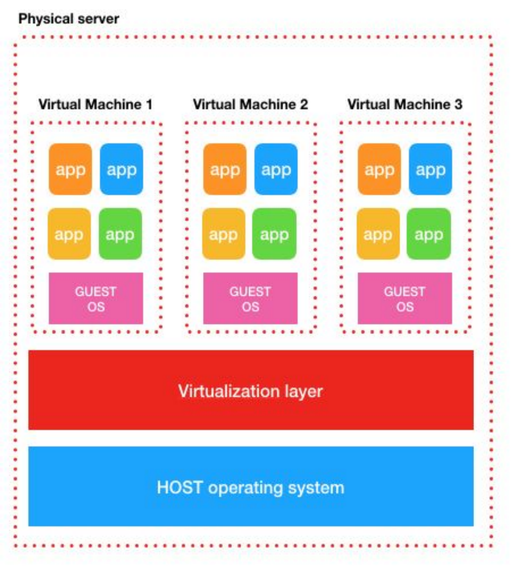
Partition of computer (*host*) in several virtual computers (*guests*) that run concurrently

Each *guest* runs its own operating system

Each *guest* is isolated from the other guests

Allocations of *host*'s resources to *guests* is managed by a Virtual Machine Manager (VMM)

There are different kind of virtualization (hypervisor based virtualisation, kernel-level virtualisation, *shared-kernel virtualisation*)



History of virtualization with virtual machines

Early 1960s: IBM M44/44X is an IBM 7044 divided in several sub computers (first use of ‘Virtual machines’ word)

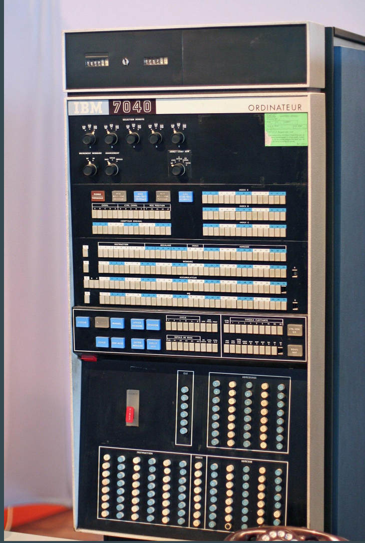
1964: IBM CP-40 research system used to prove the capability of time sharing using virtual machines. It paved the ground for the CP-67 system (1967)

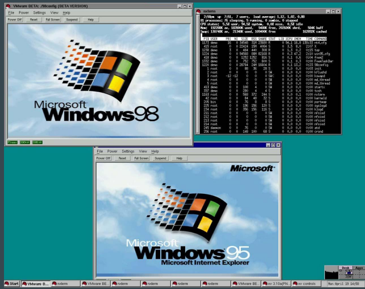
1972: IBM VM/370 (first standard release with VMs)

Late 1990s: Computers are cheaper and faster. Need for solutions that enable to run other OS (VMWare 1.0 in 1999)

2000s: Development of hypervisors (XEN, VMware).

⟶ Virtualisation of servers, Cloud Computing (IaaS layer)





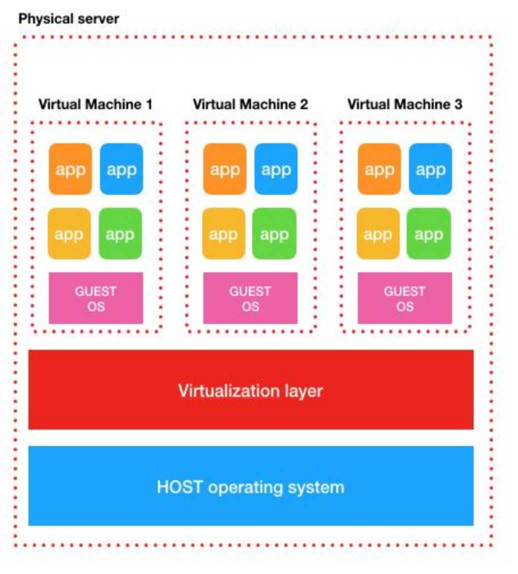
What is a virtual machine?

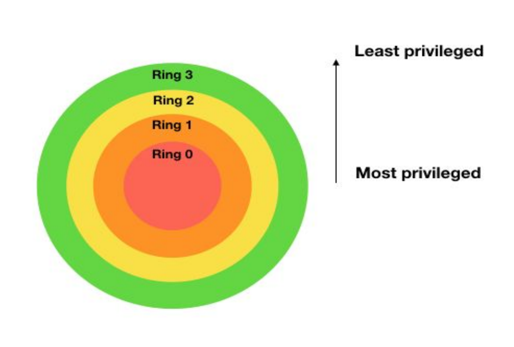
Like a typical computer, a virtual machine will have:

- One or several virtual CPUs
- An allocation of RAM
- One or several virtual disks
- One or several network interfaces

A virtual machine is created from a virtual image, which is like a snapshot of a file system.

Virtual machines support operations such as snapshotting, live migration.



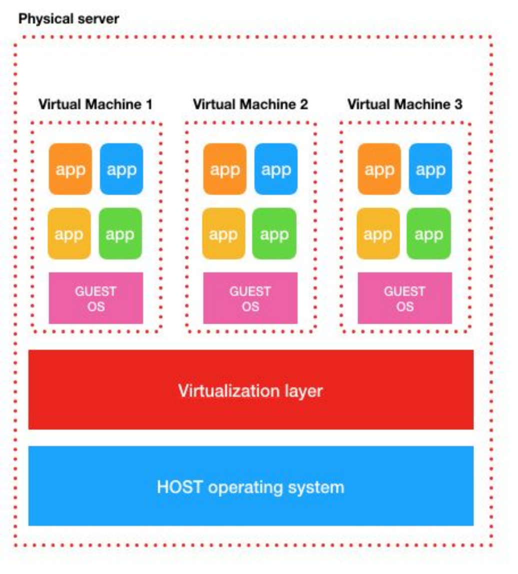


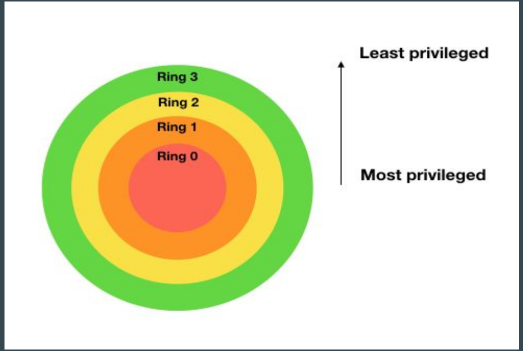
How does it work?

Protection rings define privilege levels for instructions executed by a processor

An operating system needs to run privileged instructions (memory management, IOs, …) : it runs in ring 0, while applications runs in ring 3

Different techniques enabled a guest operating system to run privileged instructions despite not running directly on ring 0.

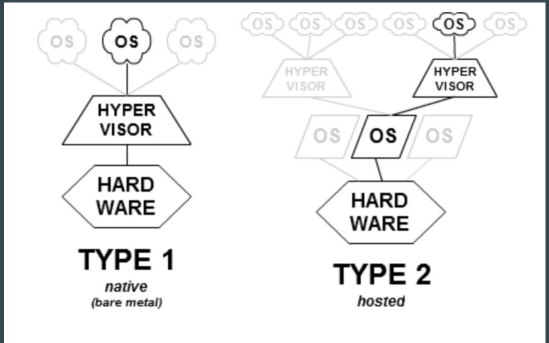




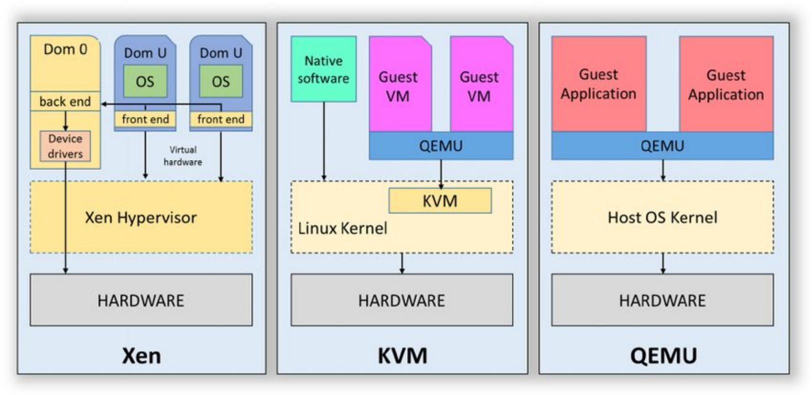
Hypervisors (list them)

- Native hypervisors:
  - Xen
  - KVM
  - VMWare ESXi
  - Microsoft HyperV (Azure)

- Hosted hypervisors:
  - QEMU
  - VMWare workstation/fusion
  - VirtualBox



credits : wikipédia



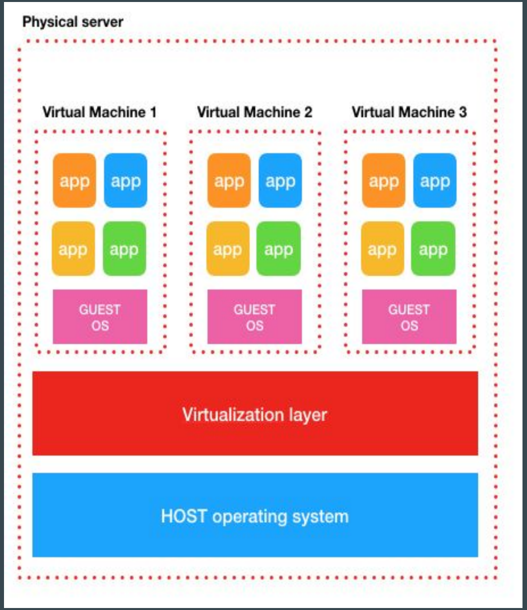
Credits: [2015 Song]

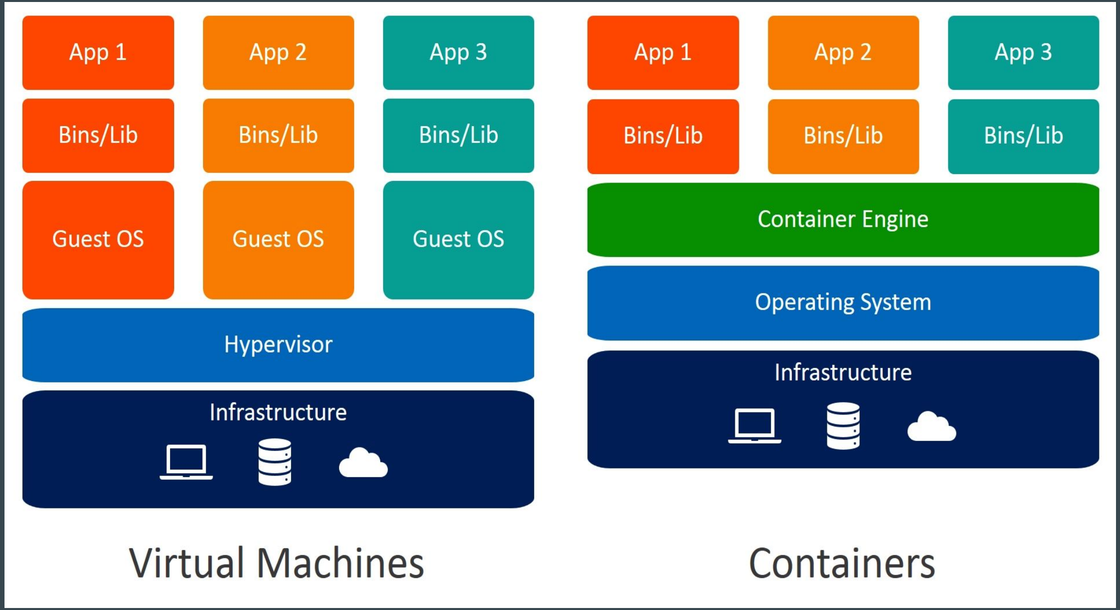
Demo

Virtualization

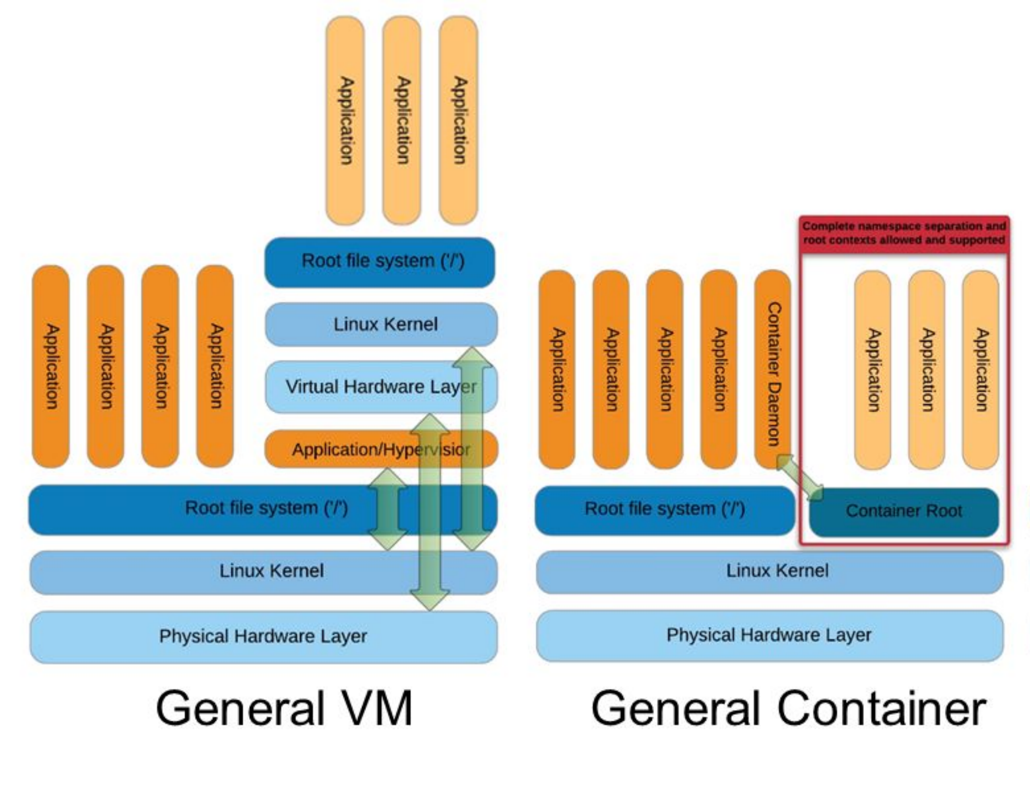
Containers

VMs vs containers - the big picture





VMs vs containers - a system vision



History

- 2002 namespace isolation in the Linux kernel
  - Isolation of processes
  - E.g. partitioning memory allocation into namespaces
- 2007 control groups (cgroups) developed by Google in the Linux kernel
  - Partitioning of the resource usage of processes
  - Resource limitation, priorities, accounting, control
- 2008 Linux Containers (LXC)
  - Combination of namespaces isolation and cgroups
- 2013 Docker containers
  - Initially an evolution of LXC
  - Oriented for the packaging of apps or services

- *Other solutions: CoreOS (Rocket), runC, Singularity (HPC) etc. (see this link)*
- *Containers open specification: OCI*

Pros & cons containers

Pros

- Light virtualization mechanism
- Almost no overheads compared to VMs
- Faster startup time
- Easy packaging of apps (Docker)
- Portability and simplified infrastructure management

Cons

- Live migration more difficult than VMs *[2017 Al-Dhuraibi]*
- Basically one operating system for all applications
- Security and isolation more difficult than VMs [2019 Shen]

LXC vs Docker - Pet vs Cattle

- Scale up, evolution
- Long lifecycle

- Scale out
- Consume and discard

Operating system level containers - LXC/LXD (pet)

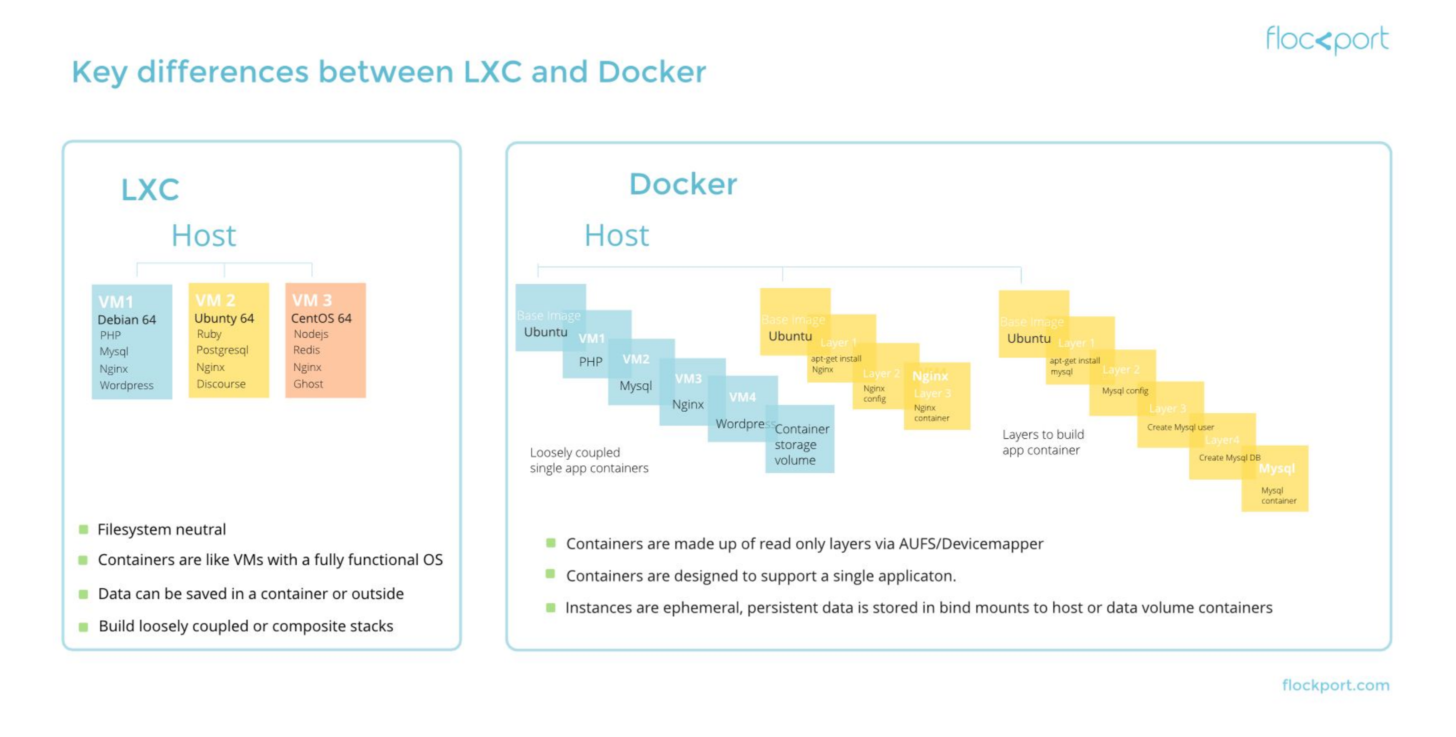
- **Usage close to a virtual machine**
  - One OS init for each container
  - Access and modification of the operating system
  - Persistent data embedded within the container
- Very light ecosystem easy to handle

- LXD
  - Extension of LXC Written in Go
  - REST API that connects to libxlc
  - A host can run many LXC containers using only a single system daemon
  - Improves security within LXC containers
  - Simplifies networking and data storage sharing between containers
  - Container migration and snapshot of a running container

Application level containers - Docker (cattle)

- Docker started as a project to build single-application LXC containers
- Introduction of several changes to LXC that make containers more portable and flexible to use
- Nowadays have their own container technology
- Written with the Go language
- Complex Docker ecosystem
  - Docker Hub
  - Docker Desktop
  - Docker swarm, Kubernetes, Marathon, Mesos etc.

LXC vs Docker images



Key differences between LXC and Docker

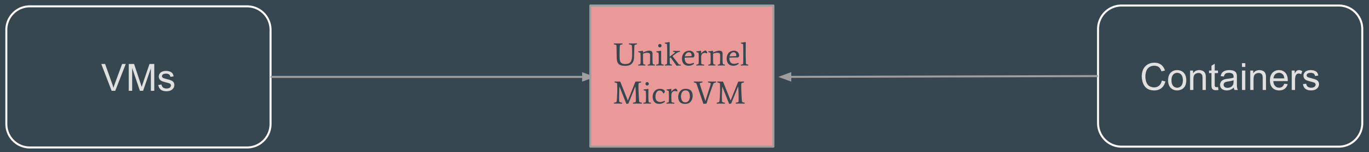
- Filesystem neutral
- Containers are like VMs with a fully functional OS
- Data can be saved in a container or outside
- Build loosely coupled or composite stacks

- Containers are made up of read only layers via AUFS/Devicemapper
- Containers are designed to support a single applicaton.
- Instances are ephemeral, persistent data is stored in bind mounts to host or data volume containers

VMs and containers

- Isolation & security
- Multiple OSs
- Live migration

- Light
- Efficiency
- Apps packaging



https://unit42.paloaltonetworks.com/making-containers-more-isolated-an-overview-of-sandboxed-container-technologies/

Demo

Containers

1. LXC (pet)
   - https://linuxcontainers.org/lxd/try-it
   - Create a container and use it
   - Snapshot and restore (LXD)
2. Docker (cattle)
   - Pull an image from Docker Hub
   - Run a container
   - Create a new layered image
   - Docker containers creation
   - Data volumes creation

What you have learned so far

- The basic concepts about virtual machines
- The basic concepts about containers
- The main differences between VMs and containers
- The basic usages of VMs and containers

What if you are an infrastructure provider?

- You have to handle many VMs and containers (resources) simultaneously
- Some of them being interdependent
- You have to handle a multi-node and even multi-cluster infrastructure
- You have to handle many users simultaneously

Operating system for virtualized infrastructures?

- VMs and containers orchestration
  - Subpart of OpenStack (opensource), AWS EC2 etc. (for VMs)
  - Kubernetes, Mesos, Docker Swarm, AWS ECS/EKS etc. (for containers)
- OpenStack is the de-facto open source operating system to handle the IaaS level of the Cloud Computing paradigm

Next step
THE CLOUD COMPUTING PARADIGM

Questions?

Some references

- [2017 Al-Dhuraibi] *Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER*. Y. Al-Dhuraibi and F. Paraiso and N. Djarallah and P. Merle. CLOUD 2017.
- [2019 Shen] *X-Containers: Breaking Down Barriers to Improve Performance and Isolation of Cloud-Native Containers*. Zhiming Shen, Zhen Sun, Gur-Eyal Sela, Eugene Bagdasaryan, Christina Delimitrou, Robbert Van Renesse and Hakim Weatherspoon. ASPLOS 2019.
- [2015 Song] *Hardware and Software Aspects of VM-Based Mobile-Cloud Offloading*. Song, Yang & Wang, Haoliang & Soyata, Tolga. (2015).