Sistemas Informáticos (Computer Systems)

Unit 04. Virtual machines and containers

Authors: Sergi García, Alfredo Oltra

Updated October 2022

Licencia

Nomenclatura

A lo largo de este tema se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

📖 **Important**

❕ **Attention**

💬 **Interesting**

INDEX

UNIT 04. VIRTUAL MACHINES AND CONTAINERS

1. VIRTUALIZATION

Virtualization is a set of hardware/software techniques that let to abstract the real hardware and software and simulate physical resources, operating systems, etc.

Virtualization is often implemented in institutions, companies, schools, etc. because it is easy to implement, and it has a lot of advantages.

An example of virtualization are Virtual machines. Also, there are another examples of virtualization, like video game emulation or containers. Remember, virtualization is a set of techniques. That techniques are used to create Virtual machines and containers.

2. DEFINITION OF VIRTUAL MACHINE

Sometimes, we need to test a new operating system, a configuration or a program in a clean environment. How can we do it easily? Using virtual machines.

A virtual machine let to simulate a computer (with his own OS) and execute programs, configurations, etc. Virtual machines are created using a virtualization program that runs on the top of the operating system of a real machine.

There are a lot of examples of virtual machines, each one with its own features: VirtualBox, VMware, VirtualPC, Parallels, JavaVM, .NET, …

3. TYPES OF VIRTUAL MACHINES

According to their functionality, we can classify virtual machines in two types

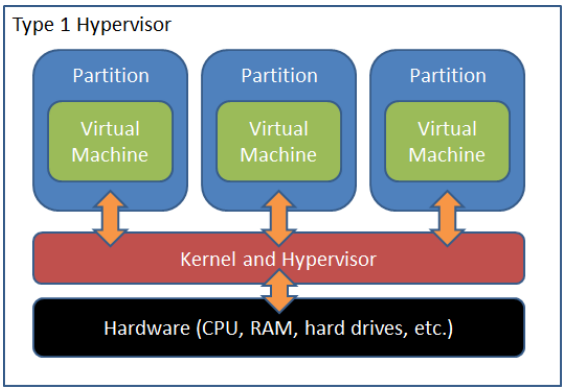
- System Virtual Machines (SVM).
- Process Virtual Machines (PVM).

3.1 System Virtual Machines (SVM)

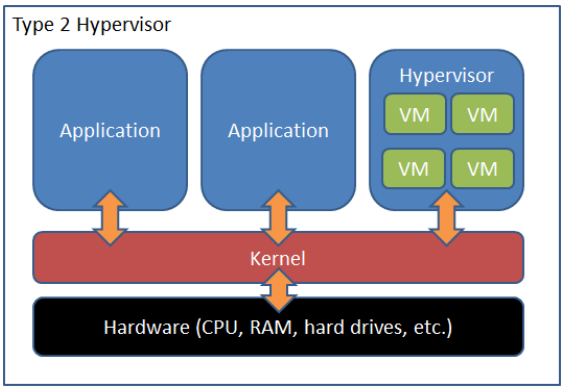
They let to replicate the real machine in several virtual machines, each one with its own OS. The software that does the virtualization is called “hypervisor”.

There are two types of hypervisor:

- **Hypervisor type 1**: hypervisor runs directly on hardware.
  - Examples of Type 1 hypervisor: VMware ESXi (free), VMware ESX, Xen(libre), Citrix XenServer (free), Microsoft Hyper-V Server (free).



- **Hypervisor type 2**: hypervisor runs as an application on a host OS.
  - Hypervisor runs in a "host" OS, and it creates virtual machines with "guest" OS.
  - Usually, OS of the real machine is called "host" and OS of the virtual machine is called "guest".
  - Disadvantage: host OS shares resources with guest OS.
  - Examples of Type 2 hypervisor: VirtualBox (free), VMware Workstation, VMware Player (free), QEMU (free).



Attention: in our subject, we are going to use Type 2 hypervisor. In our class notes, Type 2 hypervisor will be called simply "Virtual machine".

In our subject, we are going to use Type 2 hypervisor. In our class notes, Type 2 hypervisor will be called simply "Virtual machine".

3.2 Process Virtual Machines (PVM)

This type of virtual machines runs as a normal application inside a host OS and supports a single process. It is created when that process is started and destroyed when it exits. Its purpose is to provide a platform-independent programming.

Some examples:

- **Java virtual machine:** "Java compiler" generates "Java Bytecodes" and a "Java Virtual Machine" runs that "Java Bytecodes" in each OS where a "Java Virtual Machine" exists.
- **.NET**: run .NET applications where .NET is implemented (Mono for Linux, different Windows versions, etc.).

4. ADVANTAGES/DISADVANTAGES OF VIRTUAL MACHINES

Advantages of Virtual machines:

- Use several operating systems at the same time.
- Try an OS before installing it in a real machine.
- Use applications not available in your host OS.
- Emulate a different kind of computer (with other set of instructions).
- Create test environments.
- Save the current state and restore it later.
- Save energy, resources, space, etc. emulating old computers.
- Easy to clone or backup.
- Environmentally friendly (avoid to build and destroy computers, components, ...)

Disadvantages of Virtual machines:

- Share resources with other virtual machines and with host OS.
- Performance is lower than real machines.

5. HOW TO CONFIGURE A VIRTUAL MACHINE: VIRTUALBOX

VirtualBox is a free "Hypervisor type 2" available for the most popular OS. It is available in https://www.virtualbox.org/

There are several tutorials available on Internet that explain how to do it :

- How to install VirtualBox on Ubuntu: https://www.youtube.com/watch?v=QkJmahizwO4
- How to install VirtualBox on Windows 10: https://www.youtube.com/watch?v=OjBQC81oXqc
- Set up a Virtual machine on VirtualBox: https://www.youtube.com/watch?v=H_ustCy4Ks8

To improve its performance, VirtualBox let you install in your guest OS an application called "Guest additions". It is very useful, and I recommend installing it.

- Install "Guest additions" on Ubuntu: https://www.youtube.com/watch?v=Q84boOmiPW8
- Install "Guest additions" on Windows 10: https://www.youtube.com/watch?v=Bb_kJd3lSxQ

6. VIRTUAL MACHINES VS CONTAINERS

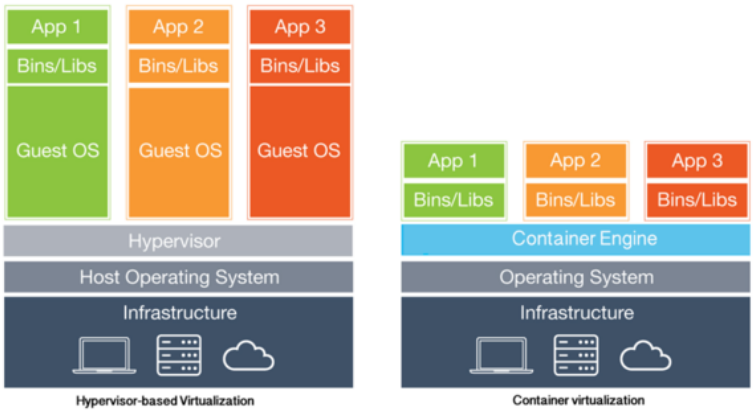
Virtual machines consist in a hypervisor that works on a physical hardware, simulating one or several "fake hardware". Each of those "fake hardware" allows emulating virtual machines (each one with its own operating systems).

Virtualization has been very successful in recent years in the deployment of applications and provisioning of infrastructure, but now the paradigm that triumphs is "Lightweight containers" or simply "Containers".

"Containers" are a similar technology to virtual machines but with the great difference that instead of using and hypervisor, there is a single-shared host operating system for containers.

Each container is really a "private environment" of the host operating system. Each container shares resources with the host operating system, without virtualizing hardware.

Thanks to this, the system is faster (it does not go through the virtualization layer) and the size of the containers is much smaller, making them efficient, easier to migrate, start, recover, move to the cloud, etc.



6.1 Containers or Virtual machines?

To summarize, is useful to run a container if we want to:

- Obtain better performance than a classical Virtual machine.
- Develop an application that can be distributed without configuration or dependence problems in the same operating system that our host.
- Develop an application easy to port to the cloud with almost any changes.
- Execute multiple copies of the same application (set of applications that work together).

If we want flexibility (for example, use a different operating system than host operating system) or execute multiple different applications in different operating systems, we must use virtual machines.

7. CONTAINERS: DOCKER

There are technologies that let us use Linux containers, such as LXC or LXD. They are popular, but the most popular container system is Docker.

You can find more information about differences between container technologies in: https://unix.stackexchange.com/questions/254956/what-is-the-difference-between-docker-lxd-and-lxc

Docker is an open source project that automates the deployment of applications within software containers, providing an additional layer of abstraction and automation of virtualization at the operating system level in Linux.

> 💬 **Interesting:** you have a lot of information about how to use Docker in https://sergarb1.github.io/CursoIntroduccionADocker/.

Docker uses Linux kernel resource isolation features:

- **cgroups**: control groups, is a feature that limits accounts and isolates the use of resources such as CPU, memory, disk I / O, network, etc.) from a collection of processes.
- **namespaces**: limits which resources can be viewed by a set of processes.

This allows independent "containers" to run within a single instance of Linux, avoiding the overhead of starting and maintaining virtual machines.

We can create containers (with commands like "*docker run*"), we can communicate with Docker containers linking input/output from a console to the container (commands "docker attach") and with commands such as *"docker cp*" to copy files.

Therefore, Docker doesn't support graphical interface. However, it is possible to handle them with graphical interface with some of these solutions:

- Installing an X Server and connect with an XWindows client to the Docker container.
- Using remote administrator software such as VNC (We recommend install NoVNC on our Docker container). With this option we don't require a special client, and we can operate directly from the browser.

More information: https://es.wikipedia.org/wiki/Docker_(software)

7.1 Docker advantages

Some of the advantages of Docker are:

- Independence of the platform: allows the use of containers in any compatible system: it could be Windows, Mac, Linux, etc.
- On Windows systems, if they have Linux capabilities, it works in native mode. In those that do not have them, install a virtual machine (on Virtual Box) and inside that virtual machine launch Docker.
- It is very easy to create and start a container.
- Each container has its own network environment, configurable, shared with other containers if it is needed.
- There are Docker images. Those are like "templates". We can create as many containers as we want under the same image.
    - Also, we can download images from third party and use them to create containers.
- We control versions of all software inside the container: operating system, version for our database, application server, etc. This eliminates configuration problems when porting the system from one machine to another.
- There are clustering and high availability solutions such as Kubernetes or Docker swarm used in production environments.
- It has a powerful image search engine already pre-generated in "Docker Hub" where we can find both official images and personal images shared by the community.
- It has support in the main systems of the cloud: Azure, AWS, Google Cloud, OVH. In fact, when you hire a VPS (Virtual Private Server), it's usually a container.

8. Installing Docker

> ! **Attention:** although it is possible, we DO NOT recommend installing Docker in a system different to Linux. For educational purposes, it is better to use a Linux Virtual machine and install Docker, than install Docker on Windows or macOS. **Do it at your risk.**

Installing Docker on Ubuntu Linux:

- https://docs.docker.com/engine/install/ubuntu/
- https://www.youtube.com/watch?v=wCSMDtHPBso

Installing Docker on Windows:

- https://docs.docker.com/desktop/install/windows-install/

Installing Docker on macOS

- https://docs.docker.com/desktop/install/mac-install/

8.1 Useful Docker commands

Official Docker reference is https://docs.docker.com/reference/

Some useful Docker commands are:

- "*`docker run -it image`*": creates a container with given image. If that image it is not in the machine, automatically downloads that image from Docker Hub and creates the container. The "-it" parameter links the container's input and output to the current console.
    - BE CAREFUL: if you run this command two or three times… you will create two or three containers!! To start again a container, use "*`docker start container`*".
- "*`docker start -i container`*": start a created container. If you have created a container and want to run it again, you should use this command. -i parameter will link the container input to the current console.
- "*`docker ps`*": It allows you to see Docker machines currently running.
- "*`docker ps -a`*": It allows you to see all Docker machines currently running or not.

- "*docker image ls*": It allows you to see the Docker images (not confuse with containers) that you have download in your machine.
- "*docker cp origen destino*": it lets to copy file between real machine and a Docker container.
- "*docker login*": command to log in "Docker hub" using console.
- "*docker pull urlmaquina:etiqueta*": command to download a Docker image.
- "*docker commit etiqueta*": command to commit a label with changes done in a Docker container.
- "*docker push urlmaquina:etiqueta*": command to upload changes in a Docker image to "Docker hub"

In our Moodle Site there is a Docker Cheat Sheet in Spanish with several commands and their associated examples.

https://raw.githubusercontent.com/sergarb1/CursoIntroduccionADocker/main/FuentesCurso/Docker%20CheatSheet%20COMPLETA.pdf

8.2 Docker Compose

"Docker Compose" is a tool that helps to configure one or several "Docker containers" using a file in YAML format. It is very easy to use and very practical to create/remove quickly "Docker containers". More information in: https://docs.docker.com/compose/

In order to find useful practical examples of how "Docker Compose" you have to read "Docker Compose" section of https://sergarb1.github.io/CursoIntroduccionADocker/

9. BIBLIOGRAPHY

[1] Virtualization

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

[2] Hypervisors

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

[3] Virtual machine

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

[4] Docker

https://en.wikipedia.org/wiki/Docker_(software)