

NOSS | Coursework 1 :

Build and deployment of a virtualized network infrastructure

Table of Contents :

1. Research on hypervisors

- **Introduction**
 - Brief overview of virtualization
 - Definition and role of hypervisors
- **Types of Hypervisors**
 - Type 1 Hypervisors (Bare-Metal)
 - Definition and characteristics
 - Common examples
 - Type 2 Hypervisors (Hosted)
 - Definition and characteristics
 - Common examples
- **Architecture of Hypervisors**
 - General architecture overview
 - Architecture of Type 1 Hypervisors
 - Architecture of Type 2 Hypervisors
 - Conclusion about architecture overview
- **Features and Hardware Requirements**
 - Core features of hypervisors
 - Hardware requirements for Type 1 and Type 2
 - Performance considerations
- **Advantages and Scalability**
 - Advantages of using type 1 or type 2 hypervisors
 - Scalability aspects of Type 1 and Type 2 hypervisors
 - Case studies or examples
- **Security Considerations**
- **Conclusion : Global view on type 1 and type 2**

2. Research on IP addresses

- **Introduction**
 - Overview of IP Addressing in Digital Communication
 - Importance and Scope of the Study
- **IP Address Structure and Format**
 - **Understanding IP Addresses**
 - Definition and Role in Network Communication
 - Basic Function and Importance
 - **IPv4 vs. IPv6 Structures**
 - Overview of IPv4 Structure and Limitations
 - Introduction to IPv6 and Its Advantages
 - **Binary and Decimal Representation**
 - Fundamental Concepts of Binary Computing
 - Representation Differences in IPv4 and IPv6
- **Representation of IPv4 and IPv6 Addressing**
 - **IPv4 Structure and Representation**
 - Composition and Dotted Decimal Format
 - Address Classes and Allocation
 - **IPv6 Structure and Representation**
 - Composition and Hexadecimal Format
 - New Features and Enhancements
 - **Efficiency and Capacity Comparison**
 - Address Space Expansion and Routing Efficiency
 - Impact on Security and Address Management
- **Subnetting and Network Organization**
 - **Concept of Subnetting**
 - Definition and Purpose in Network Efficiency
 - Impact on IP Address Allocation
 - **Techniques in IPv4 and IPv6**
 - Subnetting in IPv4: Subnet Masks and their Application
 - IPv6 Subnetting: Hierarchical Network Design

- **Efficient Utilization of IP Addresses**
 - **Challenges in Address Allocation**
 - IPv4 Address Exhaustion
 - Transition Strategies to IPv6
 - **Techniques for Optimized Allocation**
 - CIDR in IPv4: Flexible Address Division
 - Advantages of IPv6 in Address Allocation
 - **Role of Subnetting and Address Management**
 - Network Performance Optimization
 - Stability and Scalability in Network Architecture

1. Research on hypervisors :

Introduction :

Virtualization is a technology that allows the creation of virtual, rather than physical, versions of digital resources such as operating systems, servers, storage devices, and network resources. It involves using software to simulate hardware functionality, creating multiple simulated environments or dedicated resources from a single physical hardware system. This technology enhances flexibility and efficiency, as it enables better resource utilization, easier maintenance, and the ability to run different operating systems and applications on the same hardware, leading to cost savings and simplified management in IT environments.

A key component of virtualization is the hypervisor, a software layer that enables multiple operating systems to run concurrently on a single physical machine by efficiently distributing hardware resources such as CPU, memory, and storage among different virtual machines.

Two types of hypervisors are available on the market :

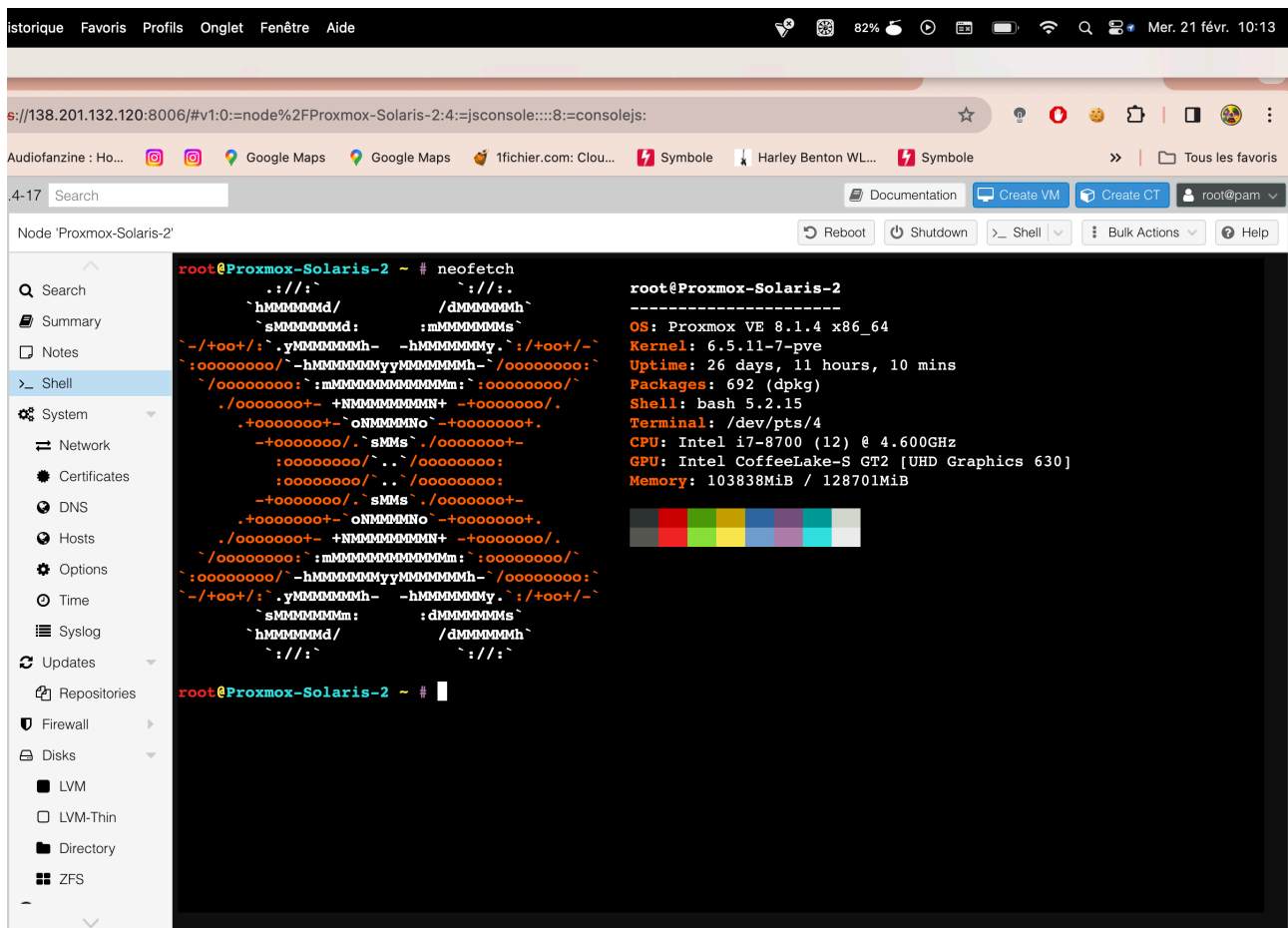
- Type 1, known as Bare-Metal supervisors, used for high availability and high scalability
- Type 2, running on the top of an existing operating system, more user friendly.

In this report, these two types of hypervisors will be explained and compared.

Types of hypervisors :

- Type 1 Hypervisors :

A type 1 hypervisor is well known as a layer of software installed directly on the top of a hardware (server, supercomputer, computer for this usage only...). In some way, a type 1 hypervisor sits as the main operating system himself or integrates one. For example, ProxMox (a type 1 hypervisor used everywhere because of its rich community and its availability, it is free) runs with a Debian operating system, allowing to control the hypervisor as using the host machine itself. The most known type 1 hypervisors are Proxmox, VMWare ESXi, Microsoft HyperV, and Xen.



Proxmox VE 8.1 with Debian 12 Bookworm operating system

- Type 2 Hypervisors :

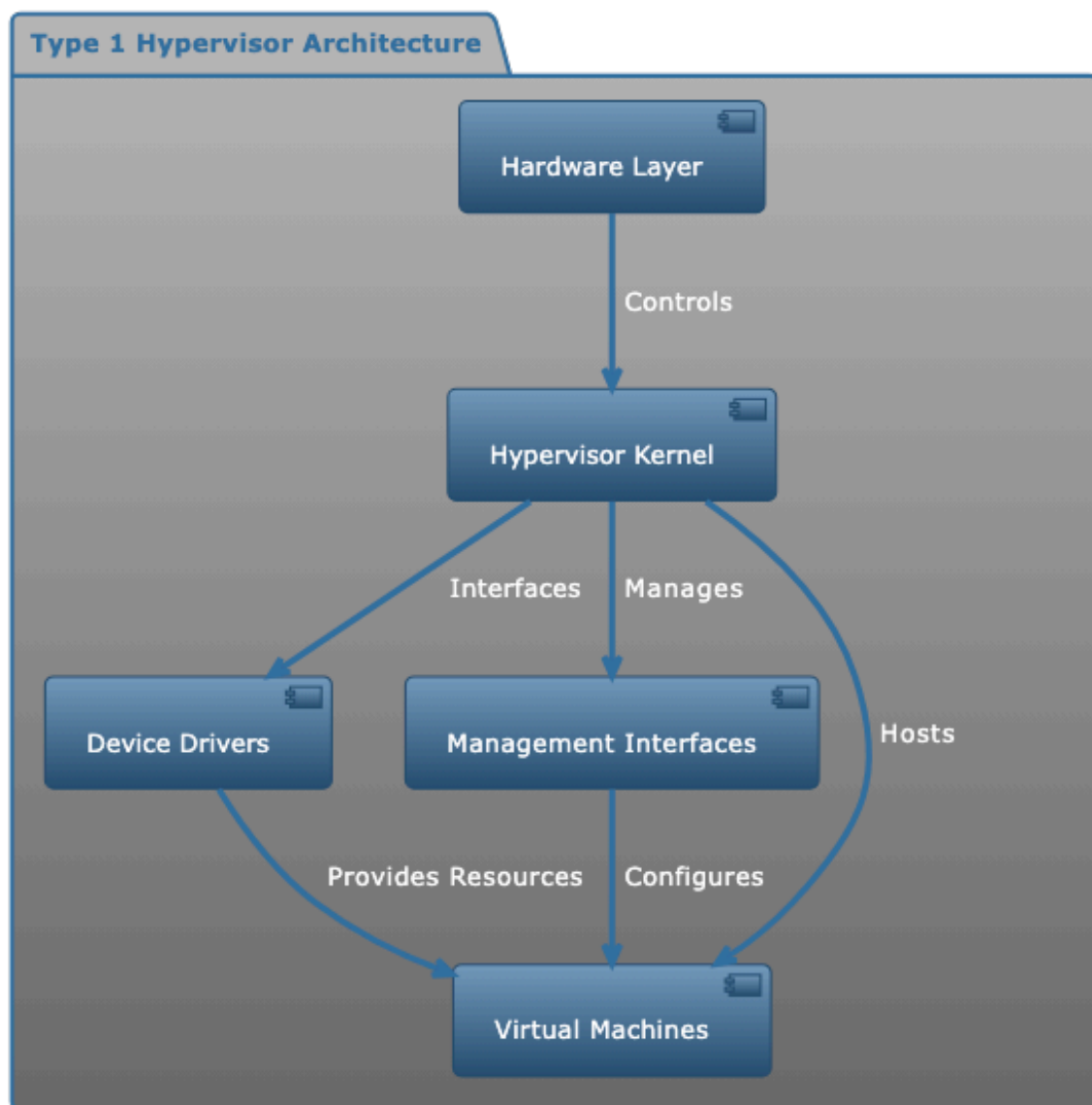
A type 2 hypervisors runs quite differently from a type 1 hypervisor. A type 2 hypervisor runs directly on the host's operating system, and is not acting as a deep system layer as type 2, the OS being running behind. Type 2 hypervisors have not a total control of the hardware like type 1 and are using the ressources manager of the host's operating system. To install a type 2 hypervisor, the user just has to download a setup wizard on the internet.

The most known type 2 hypervisors are Oracle Virtualbox (open-source), VMWare Workstation, Parallels, and UTM.

Architecture of hypervisors :

- Architecture of type 1 hypervisors :

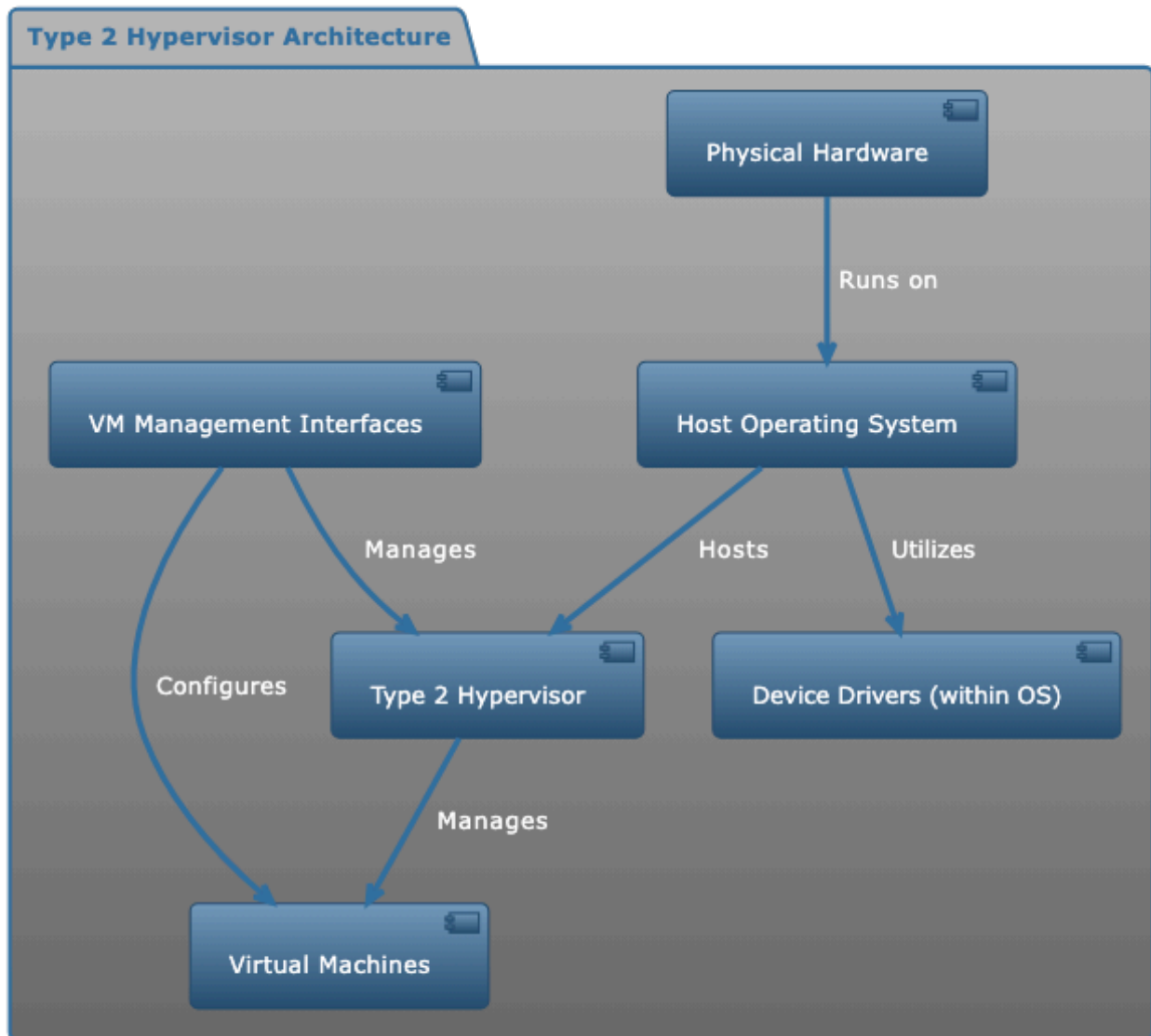
Type 1 hypervisors run directly above the host's hardware. The kernel (a term to name the system core of an OS, or here an hypervisor) controls host's hardware. Kernel actions can be controlled by the user with interfaces. These interfaces are composed by device drivers, and different management interfaces (VM creation process, performance monitoring, network configurations...). All of these user and non-user (drivers) interfaces make up the hypervisor.



Type 1 Hypervisor Diagram, generated by GPT with Architecture description

- **Architecture of type 2 hypervisors :**

Type 2 hypervisors can be installed on host's Operating System already configured, such as Microsoft Windows, any Linux distribution, or MacOS. Then, ressources and hardware management are not directly managed by the hypervisor, but by the local OS installed on the host. For Type 2 Hypervisors, management interfaces are mixed between operating system and the hypervisor itself.



Type 2 Hypervisor Diagram, generated by GPT with Architecture description

Conclusion on architecture overview :

In conclusion, the architectures of Type 1 and Type 2 hypervisors represent two distinct approaches to virtualization. Type 1 hypervisors, being installed directly on the host's hardware, offer a more efficient and direct control over the hardware resources. The kernel in these hypervisors directly interacts with the hardware, with user and system-level interfaces managing various aspects like VM creation and network configurations. This direct approach generally results in higher performance and better resource utilization, making Type 1 hypervisors suitable for enterprise and high-performance environments.

On the other hand, Type 2 hypervisors are installed on an existing operating system, relying on the host OS for hardware resource management. This layer adds some overhead, potentially reducing efficiency compared to Type 1 hypervisors. However, Type 2 hypervisors offer greater flexibility and ease of use, particularly for users who need to run virtual environments on their existing OS without modifying the underlying hardware setup. This makes them ideal for development, testing, and educational purposes where simplicity and convenience are prioritized over raw performance.

Hardware requirements comparison :

- For Type 1, taking Proxmox as reference :

According to proxmox and Debian official website, the resources needed to make the hypervisor running are :

- Any intel or AMD 1 Ghz processor with Intel Virtualisation technology (Intel-VT) or AMD-V, x86 or ARM64 (still in beta)
- Minimum 2 Gb of RAM
- SSD disk, minimum 25 GB of storage (Debian requires minimum 10 Gb, so we can consider that the Proxmox system will be 5 to 10 gb, plus one or two virtual machines hosting Debian or any equivalent Linux OS).
- Network Card (any / not specific)

- For Type 2, taking VirtualBox as reference :

According to VirtualBox official website, the resources needed to make the hypervisor running are :

- Any processor with x86 architecture only (No Intel-VT or AMD-V needed)
- Minimum of 2 Gb of RAM to run at least a windows 7 VM or equivalent
- SSD disk with minimum 10 Gb available on host for a windows 8 VM or equivalent
- A supported host OS : Windows, many Linux distributions, Mac OS X (x86 ONLY, not the new ARM 64 chipsets at the moment), Solaris and OpenSolaris.

Performances comparison :

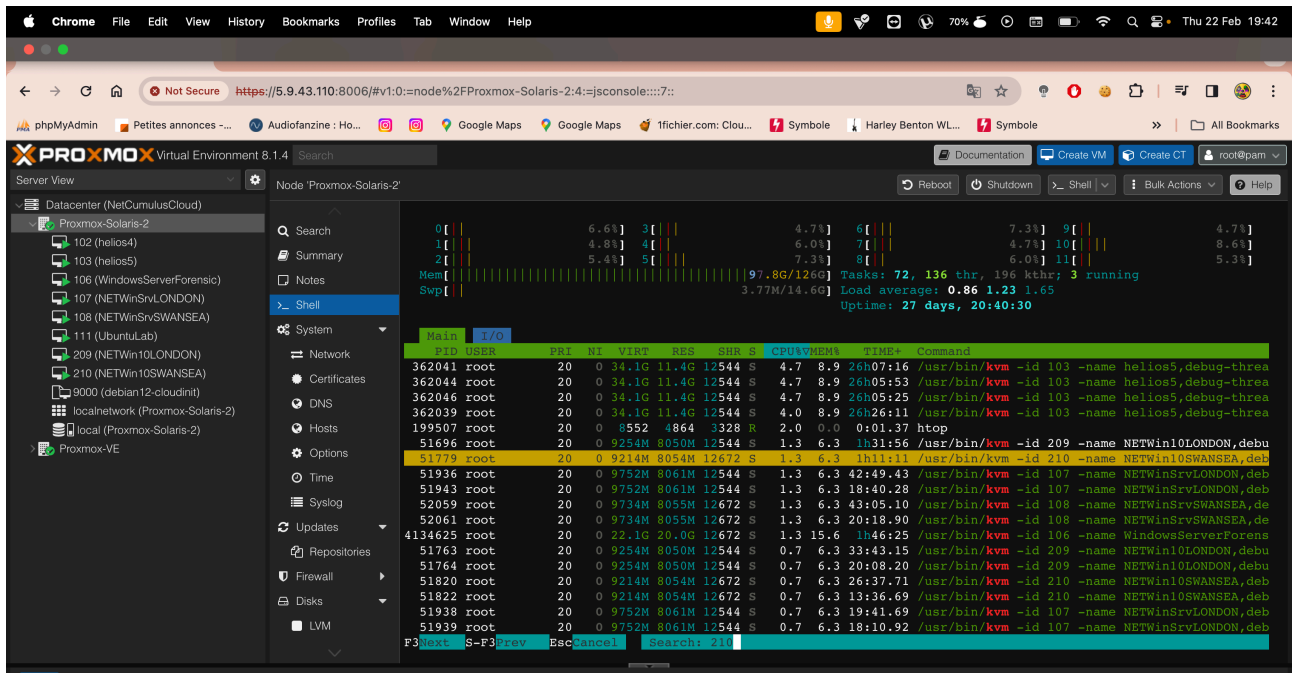
To have a better understanding about the hardware requirements and how running a type 1 or a type 2 affects the host, it has been decided to run the exact same Windows 10 Virtual machine on two hypervisors :

- Type 1 Proxmox
- Type 2 Virtualbox

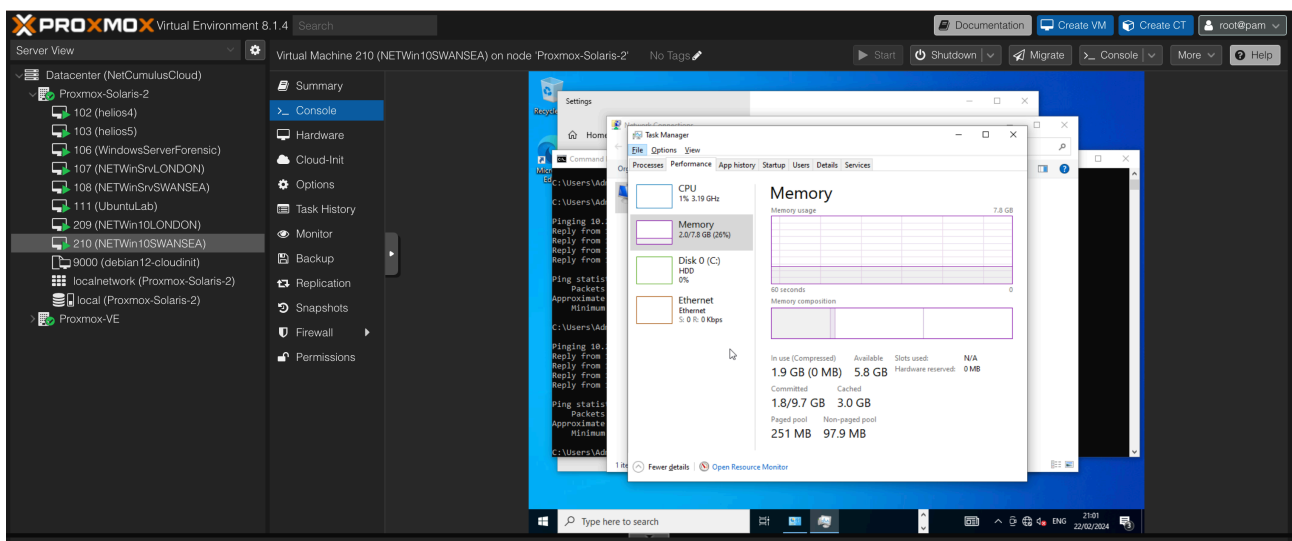
Then, we used a couple of minutes both virtual machines with same software, at the same time, with simple tasks (copying files, browsing, text edit, programs starting).

- On the type 1 hypervisor, 1.3% CPU is used, and on HTOP command, as 100% corresponds to one core used on host CPU, the server having 12 cores (1200% available), it corresponds to 0.10% of the entire processor used to run this virtual machine by doing these simple tasks. However, we see that the proxmox hypervisor allocates the 8 Gb of ram immediately, as the VM process uses 8054 Mb of ram. The VMs uses however only 2.0 Gb of ram, as it can be seen in the task manager. Proxmox guests (VMs) can be configured by using memory ballooning

technology (the allocated memory grows up til the limit set by the user only if it needs to, and goes down if all it is not used), but it is not used by default on Proxmox virtual machines.

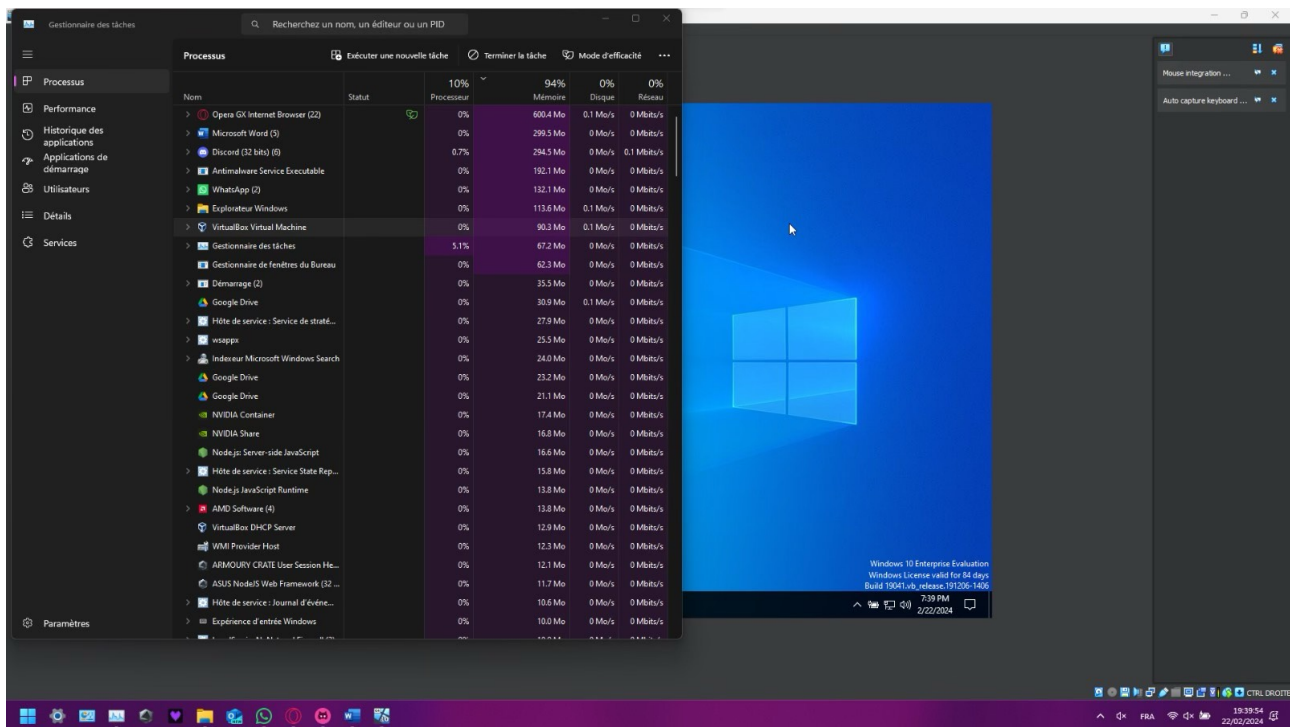


VM 210 (Windows 10 Client) hardware usage, yellow line, Proxmox host HTOP

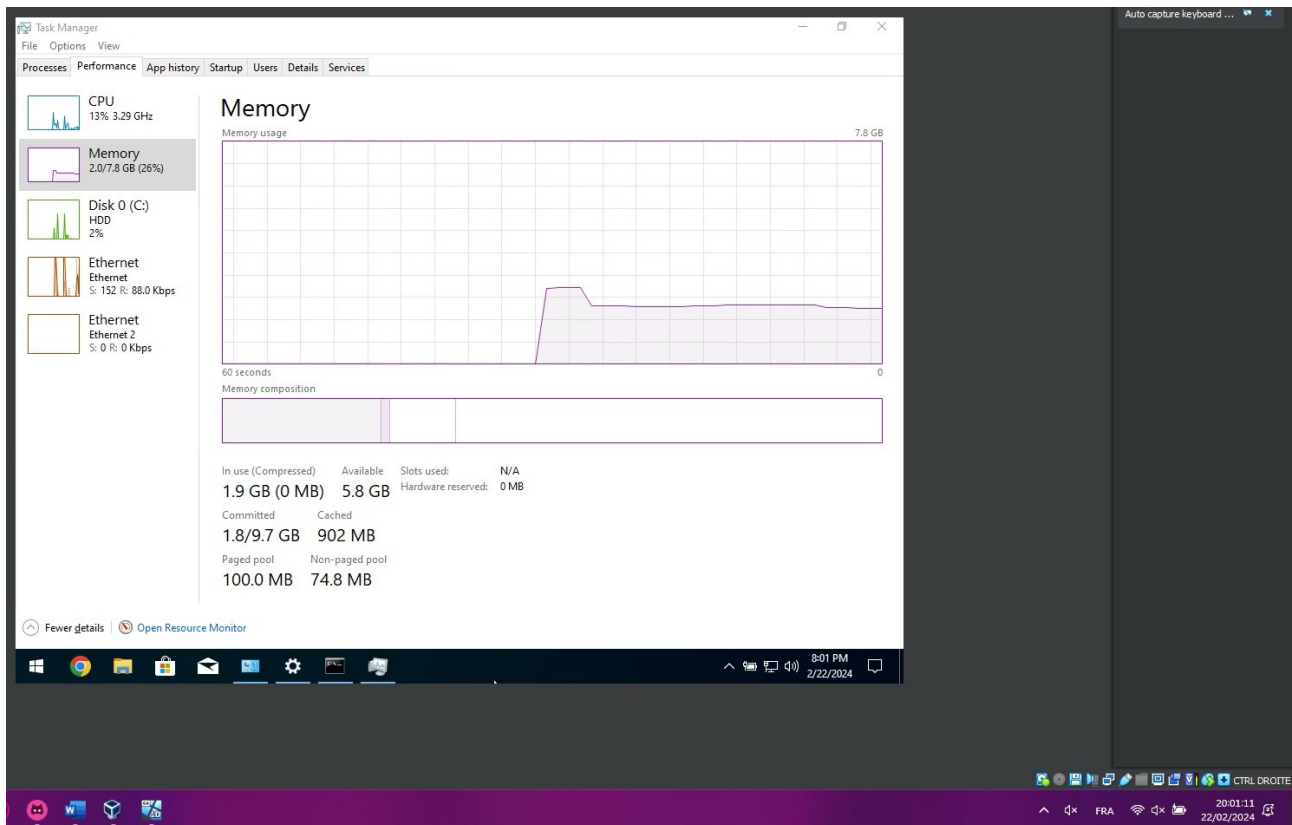


Proxmox VM ID 210 real RAM usage on the guest system, task manager

- On the type 2 hypervisor, on windows task manager, virtual box seems to use less than 1% of the global CPU. It means that there is no noticeable difference in the way that virtualbox uses CPU for VMs compared to Proxmox VMs. However, it uses less memory, with only 90.3 Mb of RAM used. It means that it probably uses a memory ballooning technology by default, got from the OS kernel. However, the VMs uses itself exactly the same amount of ram than on Proxmox, as it can be seen on the task manager (2.0 Gb of RAM).



Hardware ressources used by Virtualbox on guest, host OS task manager.



Virtualbox Windows 10 VM, real RAM usage on the guest system, task manager, same values results than on the Type 1 hosted VM.

We can conclude both hypervisors deliver similar performances on virtual machines. However, the CPU activity is lower on the Proxmox host, due to the lack of OS user operations comparing to the Windows laptop host (Opera GX, Discord, and the OS itself take between 10 to 30% constantly on laptop's hardware capacity).

Advantages and Scalability

Advantages of using a Type 1 Hypervisor :

- Without any OS supervision like it is the case on type 2 hypervisors, type 1 hypervisors can provide a total control of the hardware, allowing a better ratio of VMs per hosted per machine (no resources used by an OS let more resources available to create and manage VMs)
- Type 1 hypervisors are useful for high availability infrastructures, being extremely stable without any risks of OS errors or instabilities, because the hypervisor is the only operating system (OS).
- A great part of Type 1 hypervisors are capable of working in node groups —> Clusters. For example, multiple servers with the same hypervisors (for example, proxmox) can be connected

each other to obtain a fully redundant system, with backup or virtual machine replication, to prevent any blackout, or simply to share out / split the global system load average of an infrastructure.

- Type 1 hypervisors contain a lot of debug resources, scripts, programs, patches, updates, tutorials, established by communities, or the companies themselves (for example, proxmox has its own community forum, <https://forum.proxmox.com/>).
- Templates available to deploy VMs in just a couple of minutes

Advantages of using a Type 2 Hypervisor :

- Interfaces are intuitive and visually designed to be easy to handle by beginners —> User Friendly
- No need to uninstall or reconfigure actual host's OS
- Fast installation, just a setup wizard,

Scalability :

Type 1 hypervisors are engineered to offer robust scalability and flexibility, inherently free from the risks of operating system errors or instabilities since the hypervisor itself functions as the sole operating system. This architecture enhances system reliability and efficiency. Many Type 1 hypervisors, such as Proxmox, are adept at operating in node groups or clusters. When configured in such a manner, multiple servers running the same hypervisor can be interconnected to create a fully redundant system. This setup enables functionalities like backup and virtual machine replication, crucial for preventing downtime or system blackouts. Additionally, it facilitates the distribution or balancing of the overall system load across the infrastructure, optimizing performance and resource utilization. Scalability is the also greatest advantage of a type 1 hypervisor, as seen precedently.

Type 2 hypervisors, operating atop a host operating system, exhibit a different approach to scalability compared to Type 1 hypervisors. Their scalability is inherently linked to the capabilities and limitations of the underlying host OS. While they may not offer the same level of resource efficiency and performance scalability as Type 1 hypervisors, Type 2 hypervisors are still quite adaptable for scaling up virtual environments within the constraints of the host system.

This makes them suitable for use cases where the scalability demands are moderate and do not require extensive direct hardware manipulation. In scenarios like software development, testing, and education, where the flexibility to run multiple and varied virtual environments on a single physical machine is more critical than raw performance, Type 2 hypervisors provide a practical and scalable

solution. However, for large-scale, resource-intensive applications, their reliance on the host OS can be a limiting factor in scalability.

Security Considerations for hypervisors

Type 1 hypervisors, like Proxmox, offer enhanced security by operating directly on hardware, minimizing exposure to vulnerabilities common in operating systems. They benefit from isolation between VMs and direct hardware interaction. However, they must be vigilantly managed and updated to address hardware-level vulnerabilities and maintain their secure posture. Type 1 hypervisors like proxmox offers powerful pre-installed security functions, as an integrated fully-modifiable smart firewall, auto update protocols, user-access management (with role permissions, and groups), datacenter-scale protections (if a cluster scheme is used with multiple proxmox nodes connected each other).

Conversely, Type 2 hypervisors, such as Oracle VirtualBox, rely on the security of the host operating system. While this adds a layer of dependency and automatisms, it also allows them to leverage the host's full suite of security tools and updates, including firewalls and anti-malware programs. The security of Type 2 hypervisors is as strong as the host system's security measures, and not fully customisable.

Conclusion : all Type 1 and Type 2 functionalities on a table

Aspect	Type 1 Hypervisor	Type 2 Hypervisor
Installation	Installed directly on hardware.	Installed on an existing operating system.
Architecture	Runs above the host's hardware. Kernel controls hardware directly.	Runs on host's OS. Relies on the host OS for hardware resource management.
Hardware Interaction	Direct interaction with hardware.	Indirect interaction through the host OS.
Performance	Generally higher performance due to direct hardware control.	Potentially lower performance due to additional OS layer.
Resource Utilization	Efficient resource utilization, suitable for high-performance environments.	Less efficient due to overhead of host OS.
Scalability	Highly scalable, suitable for large-scale operations. Clusters for redundancy and load balancing.	Scalability depends on host OS capabilities. Suitable for moderate scalability demands.
Security	Enhanced security due to direct hardware operation. Needs vigilant management.	Security depends on the host OS. Leverages host's security tools.
Use Cases	Ideal for high availability infrastructures, enterprise environments, and where high scalability is needed.	Suitable for development, testing, educational purposes, and where ease of use is prioritized.
Examples	Proxmox, VMWare ESXi, Microsoft HyperV, Xen.	Oracle VirtualBox, VMWare Workstation, Parallels, UTM.
Resource Requirements	Higher resource needs (e.g., for Proxmox, 1 GHz CPU with VT support, 2 GB RAM, 25 GB SSD).	Lower resource needs (e.g., for VirtualBox, any x86 processor, 2 GB RAM, 10 GB on host SSD).
Management Interfaces	Advanced interfaces for VM management, performance monitoring, network configurations.	More user-friendly, intuitive interfaces.
Advantages	Total control of hardware, high VM to host ratio, stable, supports clusters, robust debug resources.	User-friendly interfaces, no need to alter host OS, quick installation, suitable for personal/small-scale use.

2. Research on IP addressing :

Introduction

In the dynamic world of digital communication, the role of IP (Internet Protocol) addressing is foundational and multifaceted. IP addresses serve as unique identifiers for each device (known as host) on a network, enabling them to communicate effectively over the global network. This report delves into the intricacies of IP addressing, discussing its structure, the transition from IPv4 to the more advanced IPv6, the concept of subnetting, and strategies for efficient utilization.

IP Address Structure and Format

Understanding IP Addresses

An IP address is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. Its primary function is to identify the host or network interface and provide the location of the host in the network, thus facilitating the routing of internet traffic, letting it be more efficient and fast.

IPv4 vs. IPv6 Structures

IPv4 (known as Internet Protocol Version 4), is the first Internet Protocol to be majorly deployed. It utilizes a 32-bit address scheme, traditionally represented in decimal format as four numbers separated by periods (e.g., 192.168.0.1). It has been thought to be simple to memorize and type. However, due to the exponential growth of the internet and the scarcity of IPv4 addresses (limit of 32-bit Internet Protocol is 4 billion unique addresses), IPv6 was introduced, and solve the problem of IPv4 rarity. IPv6 addresses are 128-bit, significantly increasing the number of available addresses (trillions of unique identifiers...). They are typically represented in hexadecimal format, divided by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0 370:7334).

Binary and Decimal Representation

At their core, both IPv4 and IPv6 addresses are binary numbers. IPv4 addresses are often represented in decimal form for ease of use, whereas IPv6 addresses use hexadecimal representation to make them more human-readable due to their length. The binary representation underscores the fundamental nature of computing, where data transmission occurs in this binary form.

Representation of IPv4 and IPv6 Addressing

IPv4 Structure and Representation

IPv4 addresses, composed of 32 bits, are divided into four octets, each representing a byte (8 bits). The format, known as 'dotted decimal', separates these octets with periods. For example, in the address 192.168.1.1, each number between the periods represents an eight-bit binary number. IPv4 addresses are further categorized into different classes (A, B, C, D, and E) based on the network size and number of hosts they can accommodate, impacting how the address space is allocated.

bit -->	0	31	Address Range			
	0		CLASS A ADDRESS 0.0.0.0 – 127.255.255.255			
	1	0	CLASS B ADDRESS 128.0.0.0 – 191.255.255.255			
	1	1	0	CLASS C ADDRESS 192.0.0.0 – 223.255.255.255		
	1	1	1	0	CLASS D ADDRESS 224.0.0.0 – 239.255.255.255	
	1	1	1	1	0	RESERVED ADDRESS 240.0.0.0 – 247.255.255.255

IPv6 Structure and Representation

IPv6, the successor to IPv4, is using 128 bits, allowing for a vastly greater number of unique addresses. Its hexadecimal representation, separated by colons, is more efficient for expressing large addresses. For instance, 2001:0db8:85a3:0000:0000:8a2e:0370:7334 is an IPv6 address. IPv6 also introduces features like address autoconfiguration and simplified header format, enhancing the protocol's efficiency and scalability.

Efficiency and Capacity Comparison

The transition from IPv4 to IPv6 is primarily driven by the need for more address space, but it also offers improvements in routing efficiency, security, and address management. IPv6's larger address pool significantly reduces the need for techniques like Network Address Translation (NAT : NAT stands for network address translation. It's a way to map multiple private addresses inside a local network to a public IP address before transferring the information onto the internet.), enabling more direct and efficient routing of internet traffic.

Subnetting and Network Organization

Concept of Subnetting

Subnetting is a method used to divide a single IP network into multiple smaller networks, enhancing routing efficiency and network management. It involves partitioning the host part of the IP address into a network and a sub-network, allowing for finer control over IP address allocation and reducing broadcast domains (A broadcast domain is a logical area of a computer network where any computer connected to the network can transmit directly to all other computers in the same domain, without having to go through a router.).

Techniques in IPv4 and IPv6

In IPv4, subnetting is achieved by manipulating the subnet mask, a 32-bit number that separates the IP address into network and host portions. IPv6 subnetting, while similar in concept, deals with a larger address space, allowing for more hierarchical and flexible network design.

Efficient Utilization of IP Addresses

Challenges in Address Allocation

The growth of the internet has led to challenges in IP address allocation, particularly in IPv4, where address exhaustion is a significant concern. The only solution is to limit the using of the v4 protocol, or migrate progressively networks to v6 protocol.

Techniques like CIDR (Classless Inter-Domain Routing) in IPv4 help optimize address allocation by allowing more flexible division of address spaced (/24 /28 ...). In IPv6, the vast address space itself mitigates many of these challenges (because of autconfiguration, for example)

Role of Subnetting and Address Management

Subnetting lets optimizing network performance by reducing routing table complexity and improving address allocation efficiency. Proper IP address management, including the adoption of IPv6, helps to maintain a stable network architecture.

