

**Ministry of Higher Education and Scientific Research**

**University of Saad Dahleb - BLIDA 1**

**Faculty of Sciences**

Department of Informatics



**Report submitted for the fulfillment of the Master  
degree**

**Computer Science**

Option :

- Information Systems Security
- Computer Systems and Networks

**THEME :**

**Container based virtualization  
Security In Cloud computing  
environment**

Realized by

Benarbia Abdelmalek

Kihli Abdelghani

Supervised by

Mrs.Mancer

Mr. Billal Kalem

Examined by

Mrs.Toubaline

Mr.Sahnoune

25 September 2021



# Acknowledgments

First of all, we want to thank **Allah** for his grace and for giving us the strength and patience to complete this work.

We would like to express our heartfelt thanks and our deep gratitude to **Mrs.Yasmine Mancer** for supervising us in our final thesis,for her encouragement and for leading us in our work.thank you for your guidance, patience, and encouragements throughout this Master's study.

We thank **FORMINI** for the opportunity of this internship and **Mr.Bilal Kalem** for his help, his invaluable advice and continuous support.

We also thank the **members of jury** for agreeing to evaluate our work .

A big thank you to our **families** for their moral and financial support and for their sacrifices.

At the end , we want to thank **everyone** who has participated and helped us in our graduation project.

## Abstract

Containerization, also known as "operating system virtualization" has been gaining popularity in recent years, unlike classic virtualization, Containerization is a lightweight version of virtualization where containers share the same host's kernel which makes them faster and portable.

The adoption of cloud computing and containers have promoted the "MicroServices" architecture, also brings a new concept like function as a service "FaaS" in the cloud, which allows developers to run their code directly without managing any servers. However, containers are less secure compared to VMs because of sharing the host's kernel. Those containers can scale as much as the services on our system and we can have millions of them, which make the attack surface wide, thus securing containers is critical.

In this project, our goal is to offer a security solution to ensure high availability and security that can be implemented in a cloud environment, using containers' security best practices and the newest technologies like docker , CRI-O and the famous orchestration tool Kubernetes to manage our containers. All of this work has been implemented in our local machines, in Formini's testing environment and in cloud platforms such as Microsoft Azure and Google cloud platform.

### **Keywords:**

Containerization , MicroServices , Cloud Computing , Kubernetes, High availability

## Résumé

La conteneurisation, également appelée « virtualisation du système d'exploitation » est devenue populaire ces dernières années, contrairement à la virtualisation classique la conteneurisation est une version légère de la virtualisation, où les conteneurs partagent le noyau du même hôte, ce qui les rend plus rapides et portables.

L'adoption du cloud computing et des conteneurs a favorisé l'architecture « MicroServices », ce qui a apporté également un nouveau concept comme " fonction comme service" « FaaS » dans le cloud, qui permet aux développeurs d'exécuter leur code directement sans gérer aucun serveur. Cependant, les conteneurs sont moins sécurisés que les machines virtuelles en raison du partage de noyau. Ces conteneurs peuvent évoluer autant que les services de notre système et nous pouvons en avoir des millions, pour cela, la surface d'attaque devient si large, donc la sécurité des conteneurs est vraiment importante.

Dans ce projet, notre objectif est de proposer une solution de sécurité pour assurer une haute disponibilité et une sécurité qui pourra être implémentée dans un environnement cloud, en utilisant les meilleures pratiques de sécurité des conteneurs et les dernières technologies comme docker , CRI-O et le célèbre outil d'orchestration Kubernetes pour gérer nos conteneurs. Tout ce travail a été mis en œuvre sur nos machines locales, dans l'environnement de test de Formini et sur des plateformes cloud telles que Microsoft Azure et la plateforme Google cloud .

### **Mots clés:**

Conteneurisation , MicroServices , Cloud Computing , kubernetes , Haute disponibilité

## المخلص

أصبح استعمال الحاويات (Containers)، المعروفة أيضًا باسم "المحاكاة الافتراضية لنظام التشغيل"، شعبية في السنوات الأخيرة. على عكس استخدام الحاويات الافتراضية التقليدية، يمكن أن نقول أن الحاويات نسخة خفيفة الوزن من المحاكاة الافتراضية الكلاسيكية، حيث تشترك الحاويات في نفس نواة المضيف، مما يجعلها أسرع وقابلة للنقل.

تبني الحوسبة السحابية والحوايات تعزيز بنية الخدمات الصغيرة "Micro-Services"، و جلبت أيضًا مفاهيم جديدة ك "Faas" في الحوسبة السحابية، مما يسمح للمطورين بتشغيل الكود الخاص بهم مباشرة دون إدارة أي خادم. ومع ذلك، تعتبر الحاويات أقل أمانًا من الأجهزة الافتراضية بسبب مشاركة النواة (kernel).

يمكن أن تزداد هذه الحاويات بقدر الخدمات الموجودة في نظامنا ويمكن أن يكون لدينا الملايين منها، وبذلك يصبح سطح الهجوم كبيرًا جدًا، لذا فإن أمان الحاويات مهم حقًا.

في هذا المشروع، هدفنا هو توفير حل آمن لضمان التوافر والأمان العاليين اللذين يمكن تنفيذهما في بيئة سحابية، باستخدام أفضل ممارسات أمان الحاويات، وأحدث التقنيات مثل Docker و CRI-O وأداة Kubernetes المشهورة لإدارة حاوياتنا. تم تنفيذ كل هذا العمل على أجهزةنا المحلية، في بيئة اختبار شركة "فورميني" وعلى بعض مزودي أنظمة الحوسبة السحابية مثل "Microsoft Azure" و "Google cloud platform GCP".

الكلمات الدالة:

الحاويات، Containers، Microservices، الحوسبة السحابية، Kubernetes، التوافر العالي.

---

# Contents

---

<b>List of Figures</b>	<b>10</b>
<b>List of Tables</b>	<b>13</b>
<b>Introduction</b>	<b>15</b>
problematic . . . . .	15
Objectives of the work . . . . .	15
Organization : . . . . .	16
<b>1 Virtualization:</b>	<b>17</b>
1.1 Introduction: . . . . .	17
1.2 What is Virtualization ? . . . . .	17
1.3 Hypervisor : . . . . .	19
1.4 Virtualization Types : . . . . .	20
1.5 Containerization : . . . . .	25
1.6 A comparison between "Containerization" and "Virtualization" : . . . . .	26
1.7 Conclusion . . . . .	27
<b>2 Cloud Computing</b>	<b>29</b>
2.1 Introduction : . . . . .	29
2.2 What is Cloud Computing ? : . . . . .	29
2.3 Types of cloud computing : . . . . .	30
2.3.1 Public Cloud : . . . . .	30
2.3.2 Private Cloud : . . . . .	32
2.3.3 Community Cloud : . . . . .	33
2.3.4 Hybrid Cloud : . . . . .	33
2.4 Types of cloud computing services : . . . . .	34
2.4.1 Infrastructure as a Service (IaaS) : . . . . .	34
2.4.2 Platform as a Service ( PaaS ) : . . . . .	34

2.4.3	Software as a Service ( SaaS):	36
2.5	Cloud Essential Characteristics :	36
2.6	Serverless and microservices :	38
2.7	Conclusion :	38
<b>3</b>	<b>Containers Security in the cloud</b>	<b>41</b>
3.1	Introduction :	41
3.2	What is security ? :	41
3.3	The CIA Trade :	42
3.4	Security in Cloud Computing :	43
3.5	Vulnerabilities and attacks on cloud :	44
3.6	Containers Security :	46
3.7	Studied works :	48
3.8	Conclusion :	49
<b>4</b>	<b>Conception :</b>	<b>51</b>
4.1	Introduction :	51
4.2	Problematic :	51
4.3	General architecture :	52
4.4	Securing containers from the host:	52
4.5	Securing containers from running applications:	54
4.6	Securing containers from other containers:	55
4.7	Securing Host from containers:	56
4.8	Conclusion :	58
<b>5</b>	<b>Implementation :</b>	<b>59</b>
5.1	Introduciton :	59
5.2	The working space:	59
5.3	Docker :	59
5.4	VMware workstation :	60
5.5	Kubernetes :	60
5.6	Antrea :	60
5.7	Nessus :	61
5.8	CRI-O :	61
5.9	HAproxy and Keepalived :	61
5.10	Applying our solution :	62
5.10.1	Securing container from host :	62
5.10.2	Securing container from its running application :	62
5.10.3	securing containers from other containers	68
5.10.4	Securing host from containers :	72



5.10.5 High availability : . . . . .	73
5.11 Conclusion : . . . . .	80
<b>Conclusion and perspectives</b>	<b>81</b>
<b>Bibliography</b>	<b>83</b>

---

# List of Figures

---

1.1	Before virtualization [1] . . . . .	18
1.2	After virtualization [1] . . . . .	18
1.3	Hypervisor Type 1 architecture [10] . . . . .	19
1.4	Hypervisor Type 2 architecture [10] . . . . .	20
1.5	Full Virtualization [13] . . . . .	21
1.6	Para Virtualization [13] . . . . .	21
1.7	Processor Virtualization [1] . . . . .	22
1.8	Memory Virtualization [1] . . . . .	22
1.9	Memory Virtualization [1] . . . . .	23
1.10	Storage Virtualization [1] . . . . .	24
1.11	Network Virtualization [14] . . . . .	24
1.12	Network Virtualization [1] . . . . .	25
1.13	Containerization architecture vs Virtualization architecture [19] . . . . .	26
2.1	Cloud Computing [1] . . . . .	30
2.2	Public cloud [28] . . . . .	31
2.3	Public cloud provider top leaders [29] . . . . .	32
2.4	Private cloud [28] . . . . .	33
2.5	Hybrid Cloud [28] . . . . .	34
2.6	Infrastructure as a Service ( IaaS ) [32] . . . . .	35
2.7	Platform as a Service ( PaaS ) [32] . . . . .	35
2.8	Software as a Service ( SaaS ) [32] . . . . .	36
2.9	Cloud Essential Characteristics [1] . . . . .	37
2.10	Microservice vs Monolithic architecture [34] . . . . .	38
3.1	The CIA Triade [36] . . . . .	42
3.2	shared reponsability model [37] . . . . .	43
3.3	Two-factor authentication [40] . . . . .	44
3.4	RBAC model [41] . . . . .	44

3.5	SQL Injection Attack [42]	45
3.6	Cross site scripting Attack " XSS " [43]	46
3.7	SYN Flood attack [44]	47
3.8	some possible attacks in different cloud layers [21]	47
3.9	Comparing studied works	49
4.1	General architecture	52
4.2	How can host affect containers	53
4.3	Securing host	54
4.4	Securing container form running applications	54
4.5	Attacking a container that runs a vulnerable image	55
4.6	Exemple of an orchestrator	56
4.7	Container Network Interface	57
4.8	The proces of hacking a host system through a running container	57
4.9	Our proposed solution	58
5.1	cri-o vs docker vs containerd [54]	61
5.2	Nessus	62
5.3	Scan Result 1	63
5.4	Scan result 2	63
5.5	Dockerfile	64
5.6	Build image from dockerfile	64
5.7	Trivy image scanning 1	65
5.8	Trivy image scanning 2	66
5.9	Scan dockerfile using our script	66
5.10	Our Script code	67
5.11	the table tools of our Database	67
5.12	kubernetes cluster init and the generation of certificates and keys	68
5.13	Kubernetes init applying RBAC	68
5.14	kubernetes cluster init result	69
5.15	Antrea deployment	69
5.16	Antrea deployment result 1	70
5.17	Antrea deployment result 2	71
5.18	Kubernetes nodes after joining the cluster	71
5.19	Antrea status after the remain nodes join	72
5.20	Container operates as root	73
5.23	Limiting memory and cpu for containers	73
5.21	Container rootless user	74
5.22	Runs container in read only mode	74
5.24	High Availability architecture	75

5.25 High Availability architecture . . . . .	76
5.26 Keepalived master . . . . .	76
5.27 Keepalived Backup . . . . .	77
5.28 Keepalived master goes down . . . . .	77
5.29 Keepalived Backup takes the vip . . . . .	77
5.30 kubernetes Cluster respond after losing keepalived master . . . . .	78
5.31 keepalived backup goes down . . . . .	78
5.32 kubernetes Cluster not responding after loosing the backup . . . . .	78
5.33 kubernetes Cluster respond again after turning on our keepalived master . . . . .	79
5.34 Google cloud platform . . . . .	79
5.35 Microsoft cloud platform azure . . . . .	80

---

# List of Tables

---

1.1	Containerization vs Virtualization . . . . .	27
5.1	Hardware ressources used in our project . . . . .	59



---

# Introduction

---

As the industry is converging to the Cloud more and more, the tools and technologies adapt to this change and even the process of software development becomes faster and adapts to the change with new development models , culture and architecture. The classic Virtualization approaches rely on the hypervisor which doesn't fit well with application development modern architectures and needs, unlike virtualization , containerization is light, fast and fits well with the modern development challenges and demands , due that the use of containers have been increasingly implanted in organizations adn on the cloud . However the containers are less secure than virtual machines,they share the host OS's kernel and for that the security, the control and the orchestration of those containers is a challenge.

## Problematic

Cloud brings more challenges to the surface with its agility, new concepts have appeared like function as service (FAAS) and microservices architecture where the software will be divided into small services instead of one monolith application . Containers help to provide such architecture which has a lot of benefits in software development .However having multiple services extend the attack surface and make containers a good target for hackers , that is why securing containers is a big concern for the IT world.

## Objectives of the project

In our project we are aiming into securing containers in cloud environement ensuring these factors :

- Securing containers form the host.
- Securing containers form its running applicaitons.
- Securing containers form the other containers.
- Securing host form the containers.

- ensuring "High availability" for containers .

## **Organization :**

The work described in this memoir is organized into five chapters. The first chapter , we talked about virtualization , hypervisor , virtualization types and containerization. We finished the chapter with a comparison between VMs and containers.

In the second chapter , we have talked about the basics of cloud computing , its services , its types and its essential characteristics and the new concepts that have appeared due to containers and cloud computing Microservices and serverless

The third chapter presents security basics and pillars , security in the cloud , attacks and vulnerabilities that can be found in the cloud, containers security and we have analysed some related works . The fourth chapter represents the modelization of our solution where we have defined how to secure containers and presented some diagrams that explains our solution .

In the fifth chapter , we have implemented our solution , where we have presented the tools and the techniques that we have used to secure containers .

Finally, we will end with a general conclusion and some perspectives for future work.



## Chapter 1

---

# Virtualization:

---

## 1.1 Introduction:

Before virtualization , organizations used one infrastructure to run one operating system and its applications ,which caused underutilization of infrastructures, so to use another operating system they had to buy a new infrastructure which means additional costs, the virtualization came as a solution to this problem [1]. Virtualization is the cornerstone for today's technology, as we all use Internet services, those services are hosted on data centers around the globe in high-performance servers, virtualization is the technique of partitioning those resources to have a multi-environment platform, so without virtualization, we can not have multiple applications on the same server. [2, 3]

## 1.2 What is Virtualization ?

Virtualization is the foundation of cloud computing. It is a technology that allows the maximum usage of the hardware resources and isolating applications or operation systems. It allows using multiple OS simultaneously in the same device,each one is isolated from the others. [4, 5, 6]. Or as NIST [7] defines Virtualization as: "The use of an abstraction layer to simulate computing hardware so that multiple operating systems can run on a single computer". and by simulating an Operating System that means we can use all the stack above it (services/applications). The two figures ( figure 1.1 and figure 1.2) below show the computing architecture before and after virtualization.

**Virtual machine:** Virtual machine (or the guest machine ) is created on top of a physical machine(host),it runs on top of Hypervisor or Virtual Machine Monitor (VMM), it has a virtual environment and its own kernel/ operating system(OS) ,it is separated from the host (isolated) and uses the physical machine resources(CPU, Memory, Storage, Network, I/O) that are provided by the VMM.[8, 9]

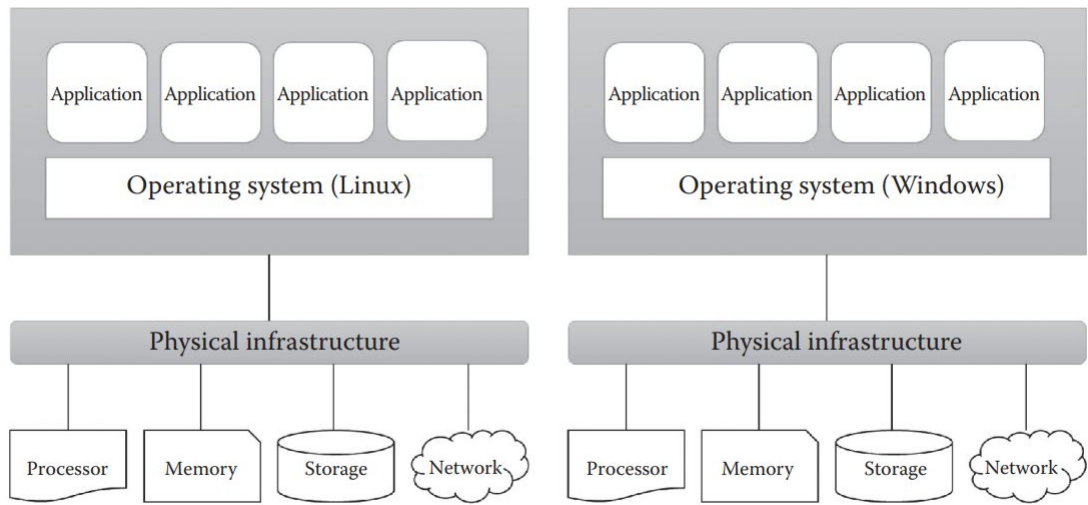


Figure 1.1: Before virtualization [1]

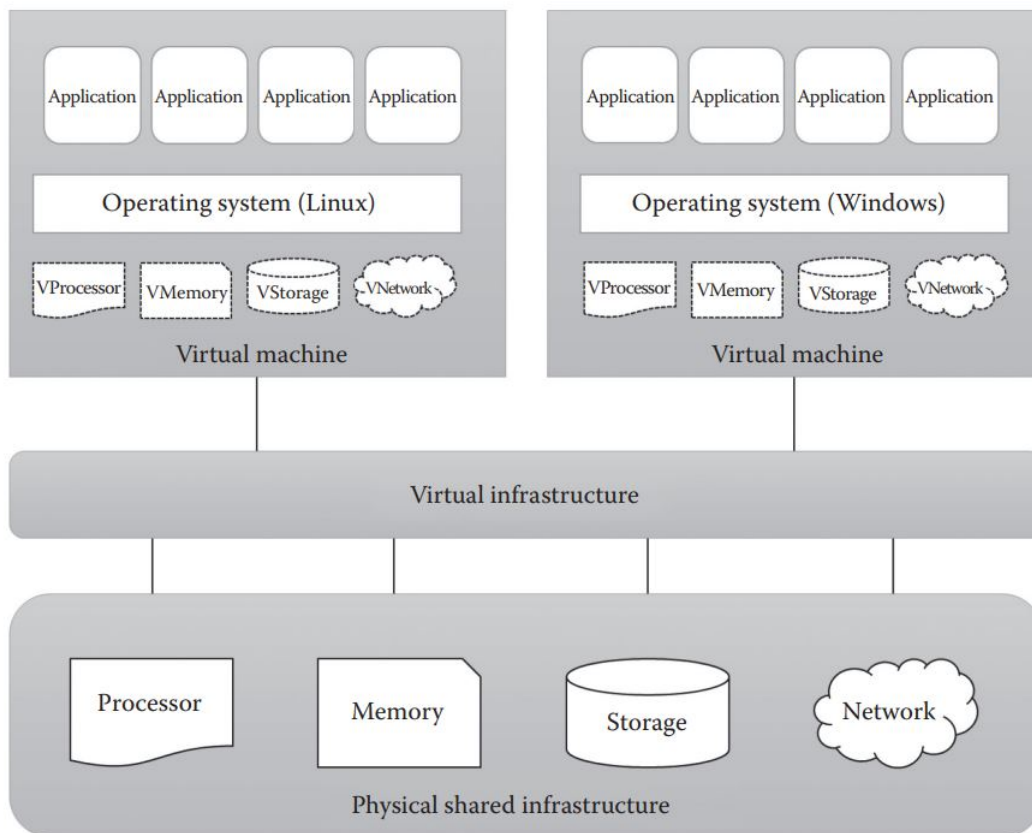


Figure 1.2: After virtualization [1]

## 1.3 Hypervisor :

The hypervisor or Virtual Machine Monitor (VMM) is the tool that allows the guest to run our virtual machines and manage our resources. According to IBM [5]: " A hypervisor is the software layer that coordinates VMs. It serves as an interface between the VM and the underlying physical hardware, ensuring that each has access to the physical resources it needs to execute. It also ensures that the VMs don't interfere with each other by impinging on each other's memory space or compute cycles." We have two types of hypervisors:

- **Hypervisor type 1 :**

Native, bare metal hypervisor installed directly on the host hardware (like an operating system) as shown in the figure 1.1 below. It substitutes the operating system, it interacts with the hardware and is mostly used for the servers [5] . The figure 1.3 below illustrated type 1 hypervisor

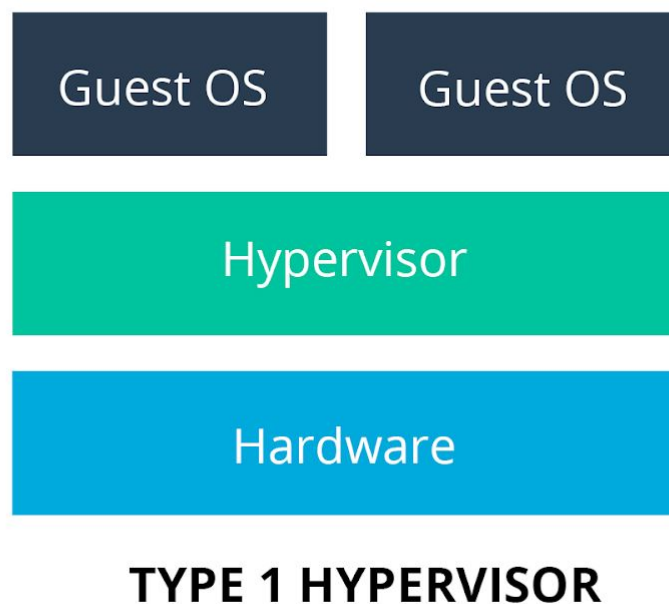
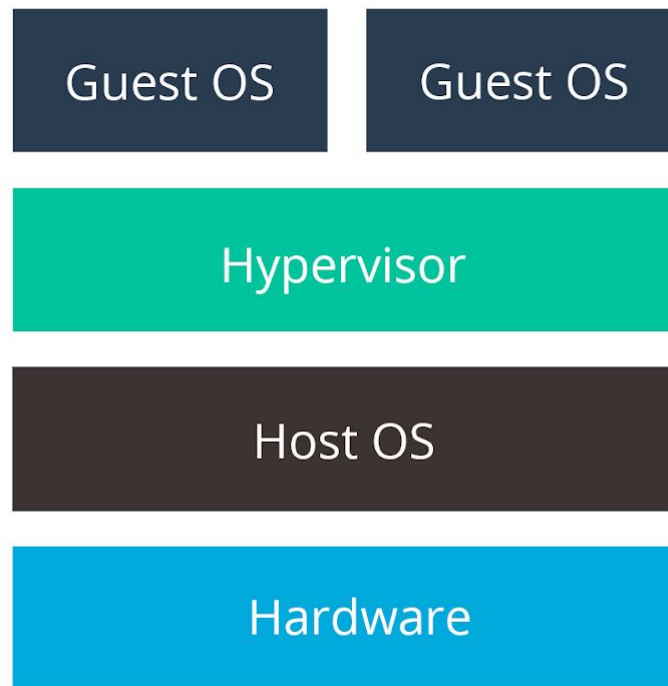


Figure 1.3: Hypervisor Type 1 architecture [10]

- **Hypervisor type 2 :**

It is installed and runs upon the OS like any other application ,it permits the usage of multiple OS simultaneously as Figure below illustrates [5, 11].

The figure 1.4 below illustrated type 2 hypervisor



## TYPE 2 HYPERVISOR

Figure 1.4: Hypervisor Type 2 architecture [10]

### 1.4 Virtualization Types :

Virtualization has a lot of types , we are going to mention :

- **Server Virtualization:** in the " Server Virtualization " wen can find :

- **Full Virtualization:**

Is the major known type , all the resources are going to be simulated ,The guest OS acts like the proprietor of the system , the simulated resources receives requests (for ex: network adaptor) and the hypervisor is just charged to translate the requests received to the original resource , this type provides complete isolation of each virtual machine but the process of translating the requests from the emulated resources to the original consumes too much resources [12].

The figure 1.5 below illustrates Full-virtualization

## Full Virtualization

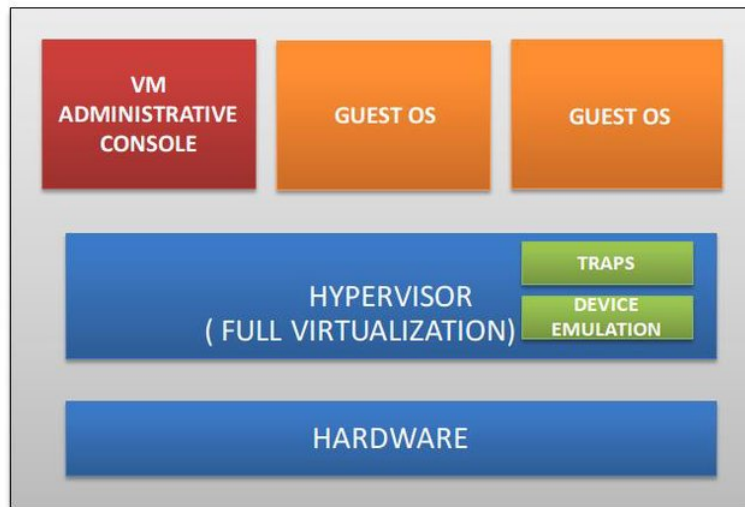


Figure 1.5: Full Virtualization [13]

### – Para-Virtualization:

In this type, the guest OS is edited to be virtualized, only some resources are going to be emulated (basically the CPU and the memory), for this type is performant but not all the OS are editable [12]. The figure 1.6 below illustrates Paravirtualization

## Para Virtualization

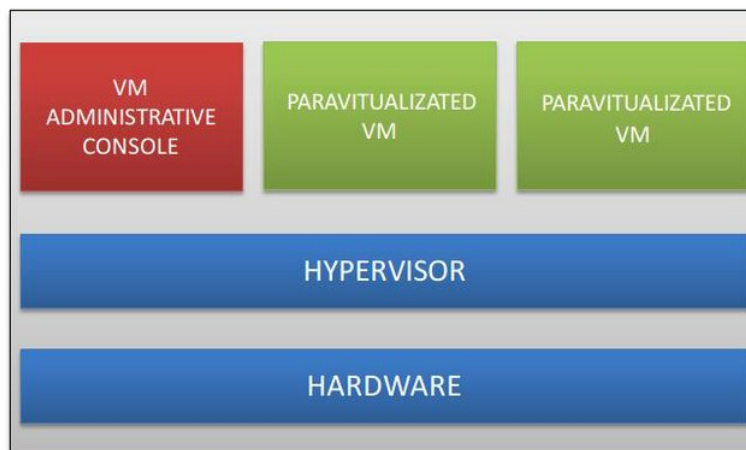


Figure 1.6: Para Virtualization [13]

### • CPU virtualization :

It is the most important technology that makes hypervisors and virtual machines possible, it allows the division of a single CPU into multiple virtual CPUs that can be used by multiple VMs. At the first, it was a little bit difficult, because only the host can execute instructions in supervisor mode, the guests shouldn't have the ability to do it, so some instructions that need a high privileged mode (supervisor mode) are trapped by the hypervisor, until Intel and AMD built new processors with new technologies that support virtualization [1].

The figure 1.7 below shows the processor virtualization.

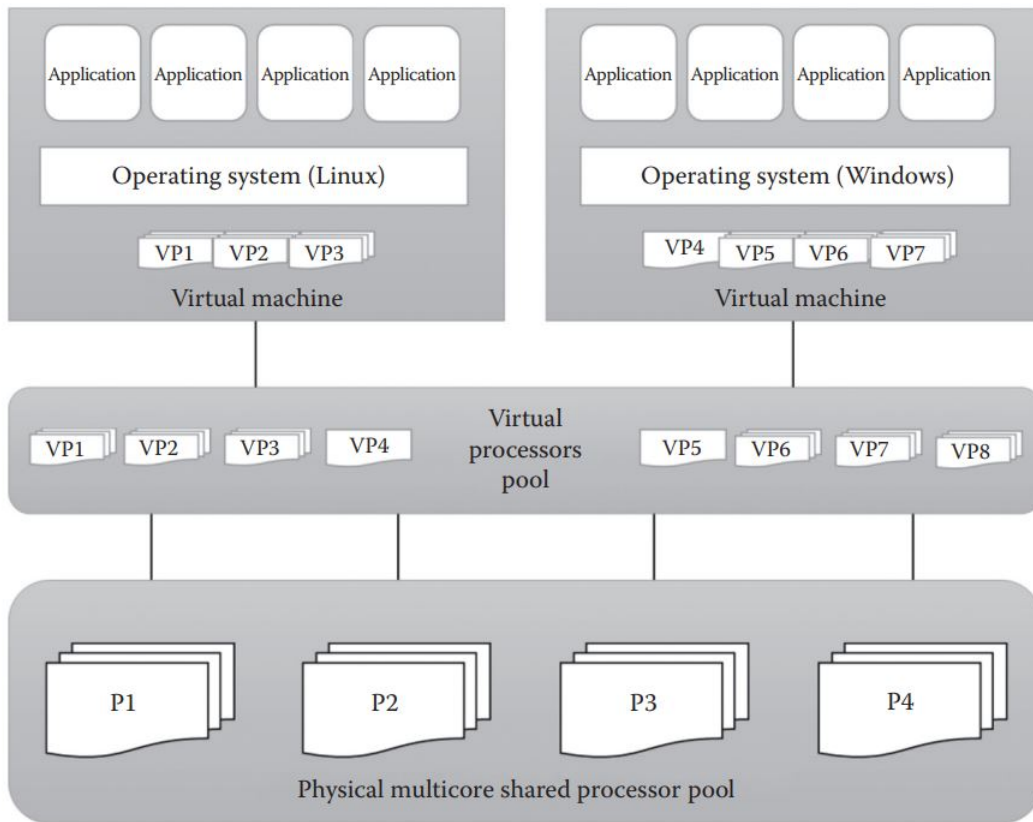


Figure 1.7: Processor Virtualization [1] .

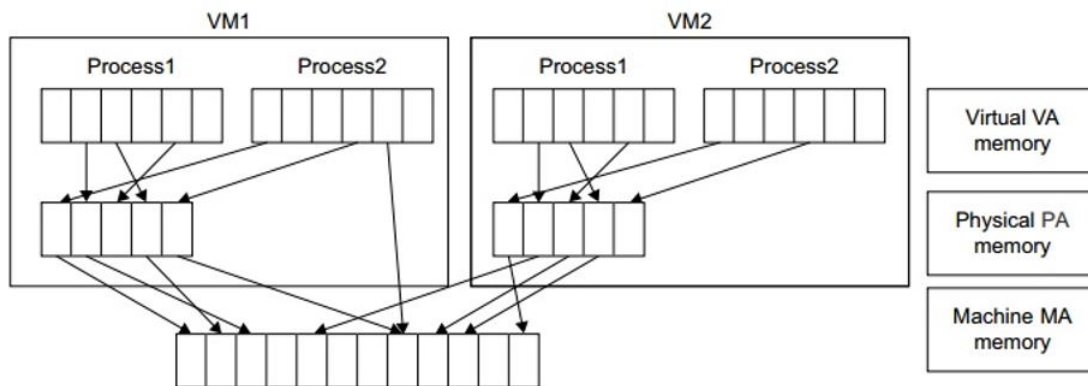


Figure 1.8: Memory Virtualization [1] .

- **Memory virtualization :** It is based on mapping the virtual page numbers into the physical page numbers [1]. The figures (figure 1.8 and figure 1.9 ) illustrate memory virtualization .

- **Storage virtualization :**

according to IBM [5] : "Storage virtualization enables all the storage devices on the network— whether they're installed on individual servers or standalone storage units—to be accessed and managed as a single storage device. Specifically, storage virtualization masses all blocks of storage into a single shared pool from which they can be assigned to any VM on the network as needed. Storage virtualization makes it easier to provision

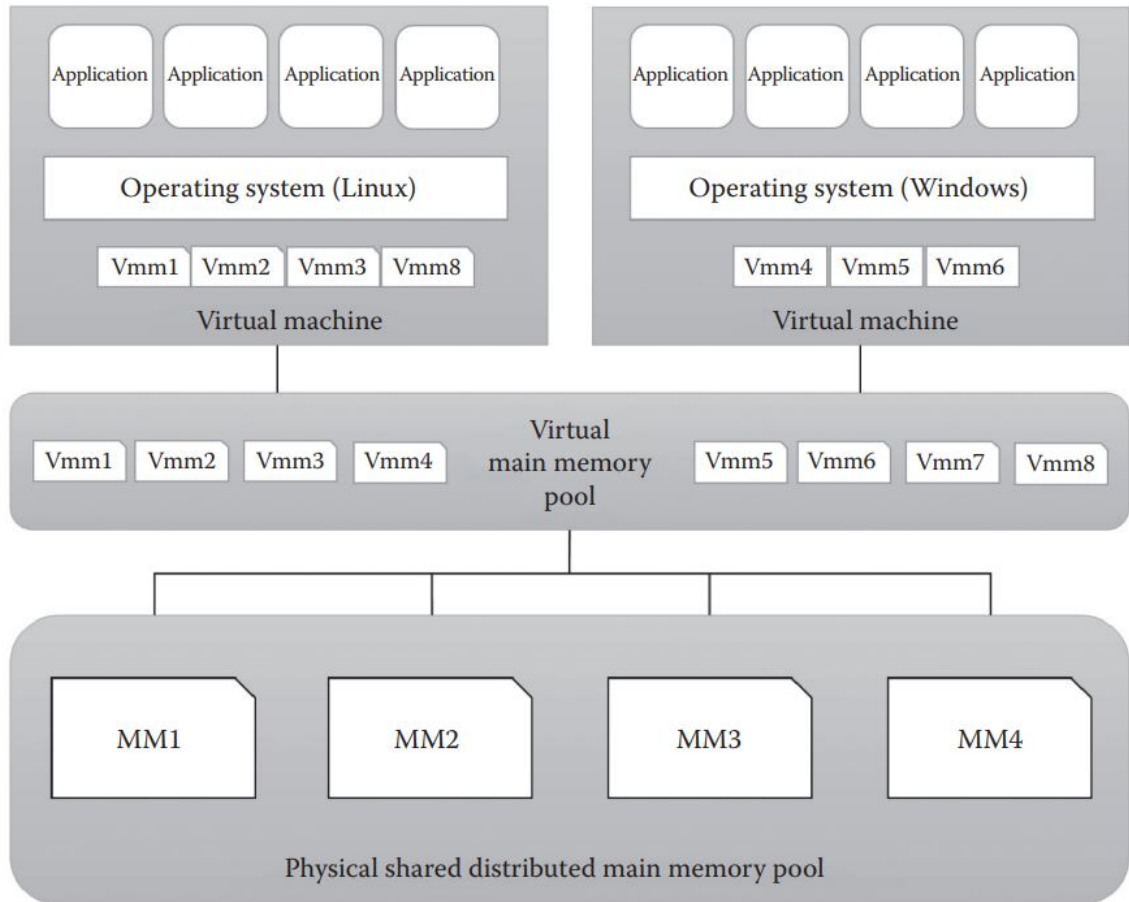


Figure 1.9: Memory Virtualization [1] .

storage for VMs and makes maximum use of all available storage on the network." the figure "1.10" illustrates the process of storage virtualization.

- **Network virtualization :**

create a “view” of the network using software. An administrator can manage the network from a single console. It simulates hardware elements and functions (e.g., connections, switches, routers, etc.) and abstracts them into software running on a hypervisor. The management and control of these elements is done without touching the underlying physical components. There are two types of network virtualization ,the first type is SDN(software-defined networking) which virtualizes hardware that controls network traffic routing. The second type is NFV (network function virtualization), which virtualizes one or more hardware devices that have a specific network function [5]. The figures (figure 1.11 and figure 1.12) below illustrates the process of network virtualization.

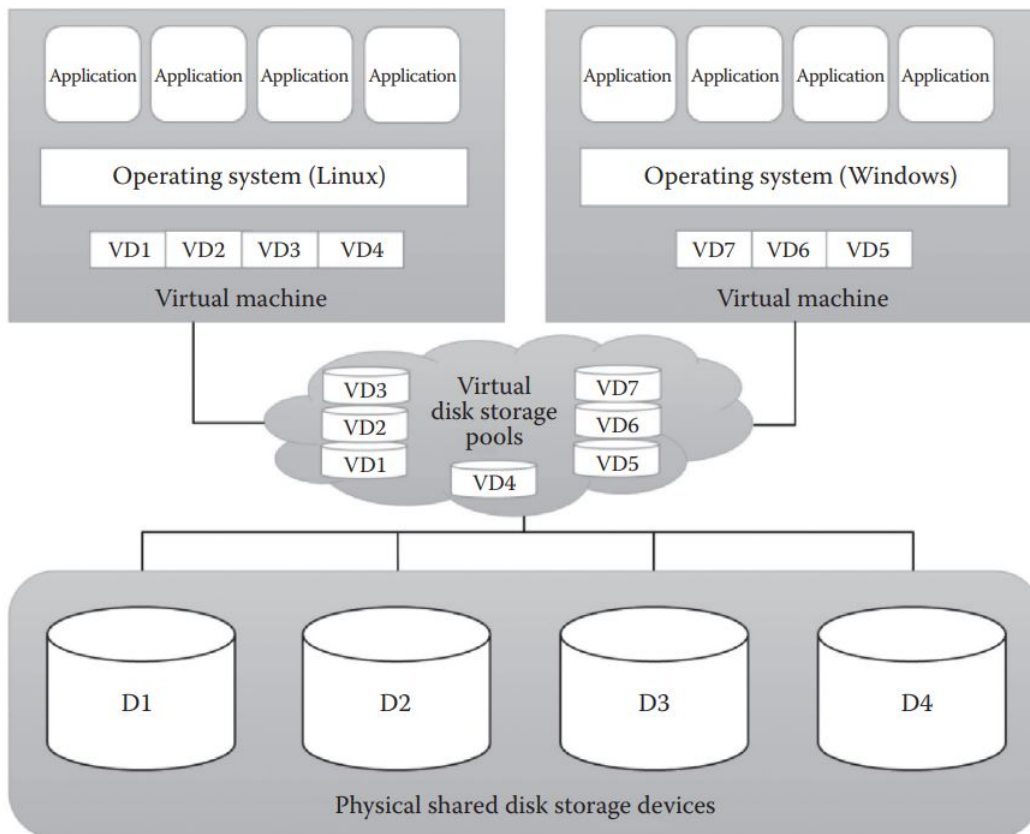


Figure 1.10: Storage Virtualization [1] .

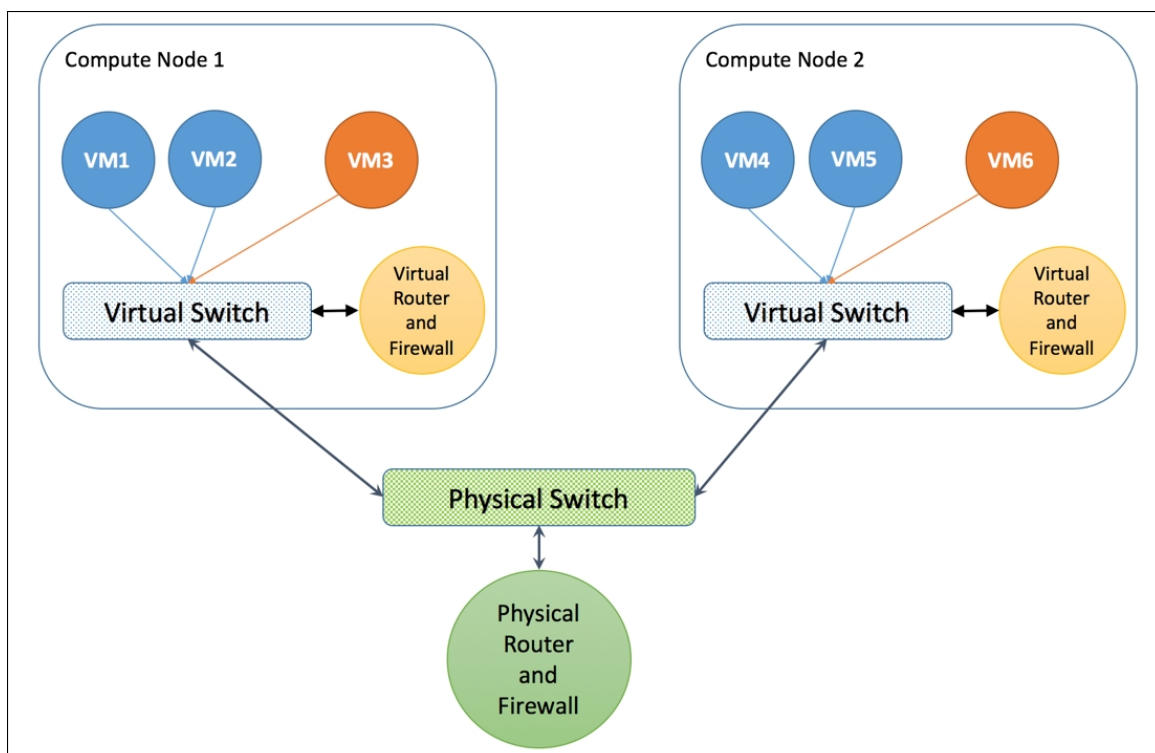


Figure 1.11: Network Virtualization [14]



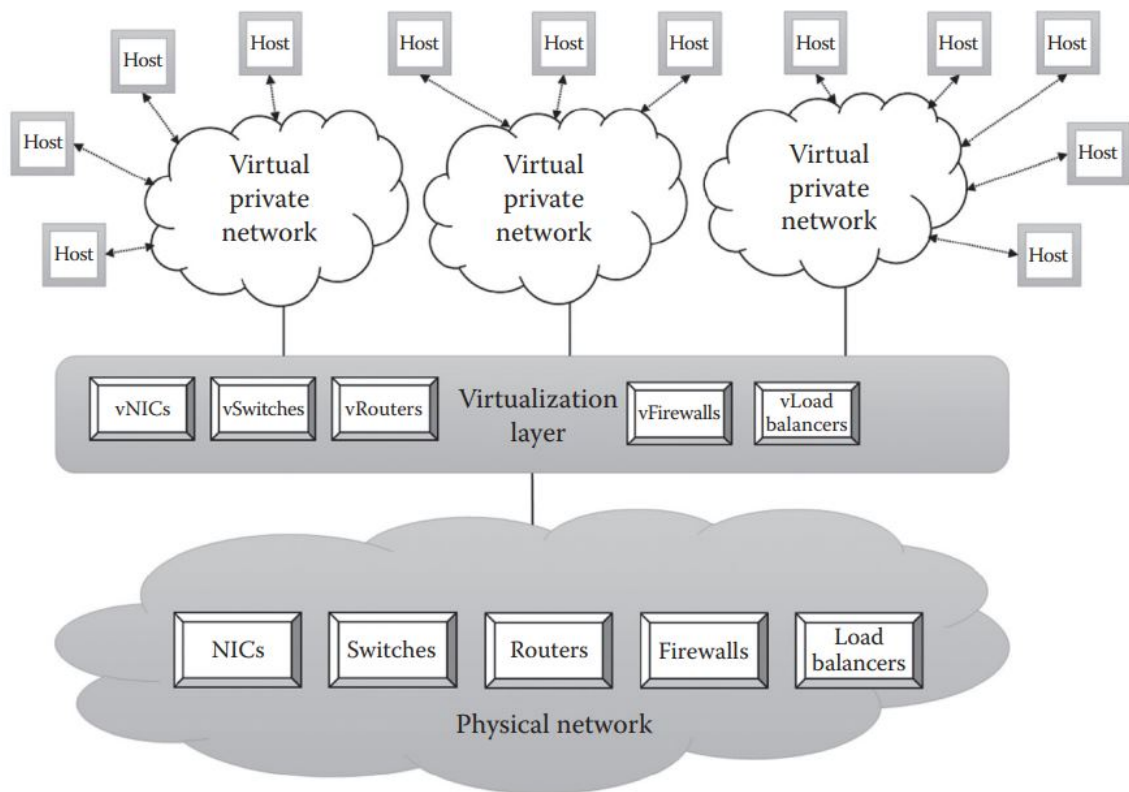


Figure 1.12: Network Virtualization [1] .

## 1.5 Containerization :

Containerization or Container-based virtualization is the last technology of virtualization ,appeared as a lightweight alternative to VMs, more performant and efficient, it packages the code and the dependencies of an application together in one container .It is based on "NameSpaces" and "Control Groups".[15, 16, 9]

A Container is a lightweight process that encapsulates application libraries and dependencies . Containers runs on top of an operating system ,they share the same host os kernel , isolated from each other and other processes by "Namespaces" ,the container ressource(hardware) are limited and managed by "Control Groups" .In other words , containers are a linux processes running on the host machine , they have a limited field of view of that host and because they are just a processes so they run on top of an operating system (host) and they share the host's kernel , their access to the host's physical resources can be limited using "Control Groups" . [15, 16, 17]

Control Groups and NameSpaces are Linux 's kernel technologies .Cgroups or 'Control Groups' is responsible for monitoring and metering resources. It limits the use of hardware resources like the CPU and the Memory for a process. This technologie protects our system from the selfish processes who are greedy in terms of resource consumption. These processes can take all the resources which can affect and interrupt other processes. [18, 17]

NameSpaces permit isolation , allows customization and appears each instance of a container like it has its own " Operating System ".[18]

We will list some containerization benefits below :

- Each instance of containers doesn't need an operating system because they share the machine's OS kernel, so they consume less space than virtual machines .[15, 16]
- Portable because they encapsulate the application dependencies and libraries so they don't have to be edited across different platforms.[16]
- Because of their portability and small size , they are perfect for the new development and application patterns like "Microservices".[16]

The figure 1.13 illustrates a comparison between containerization and virtualization.

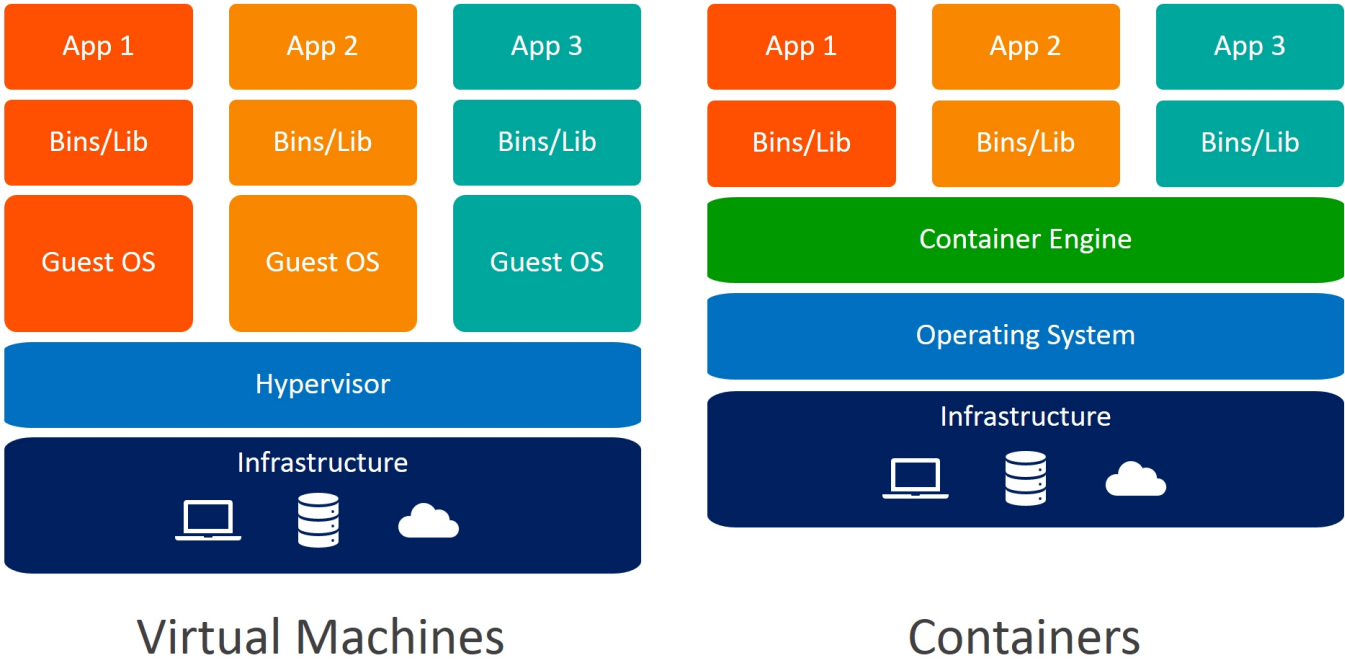


Figure 1.13: Containerization architecture vs Virtualization architecture [19]

## 1.6 A comparison between "Containerization" and "Virtualization" :

Containerization is a virtualization technology,multiple containers can share the host OS kernel where each VM has its own operating system. This point makes containers faster than VMs and Containerization as the best architecture for microservices as the container startup takes less time than VMs. The table below shows the difference between "Containerization" and "Virtualization"[15, 16, 9]. The table 1.1 illustrated the difference between containerization and virtualization

Containerization	Virtualization
Host os level virtualization	Hardward level virtualization
Process isolation	Machine isolation
Share the Host os kernel	each vm has its own operating system
light	heavy
take a less time to startup	take a long time to startup
less secure and they can affect the host	more secure and isolated from the host

Table 1.1: Containerization vs Virtualization

## 1.7 Conclusion

Virtualization is a widely used technology that came to solve the problem of underutilization of resources, it could be native or hosted virtualization, it has two known approaches full-virtualization and paravirtualization, it has been used and developed in different ways which has created the cloud computing paradigm and the "Container-based virtualization" that has been evolved and created a new software development architecture "Micro services".



## Chapter 2

---

# Cloud Computing

---

### 2.1 Introduction :

The cost and the management of the internet technology resources has always been considered as a major pain for companies , and because of the existence of the virtualization technologies and the E-payment methods , " Cloud Computing " came as a solution to solve the companies's major concern and offers a hardware and software services via the internet [20] .

The major concept of cloud computing is to outsource the control and delivery of software and hardware assets to third-party companies " Cloud Service Providers" such as Amazon (AWS) and Microsoft (Azure) , that can provide much better quality of service with a minimized cost [21] .

Nowadays Cloud Computing is one of the best choices for business, even if it could be a good step to take for a small company but also some major industries are on the cloud and they don't have any local infrastructure, like Netflix [22]. The growth of public cloud for 2021 is 23 % to total \$332.3 billion by gartner and they predict that it will continue to scale up. [23]

### 2.2 What is Cloud Computing ? :

Cloud computing is the use of Technologies like Virtualization and automation(self service) to create a platform for end users or business owners where the cloud provider will help industry to focus on it's business instead of wasting time and resources on managing local infrastructure .

According to Amazon : " Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS)." [24]

According to the official NIST definition, "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing

resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." . [25] The figure 2.1 below shows a schema of cloud computing .

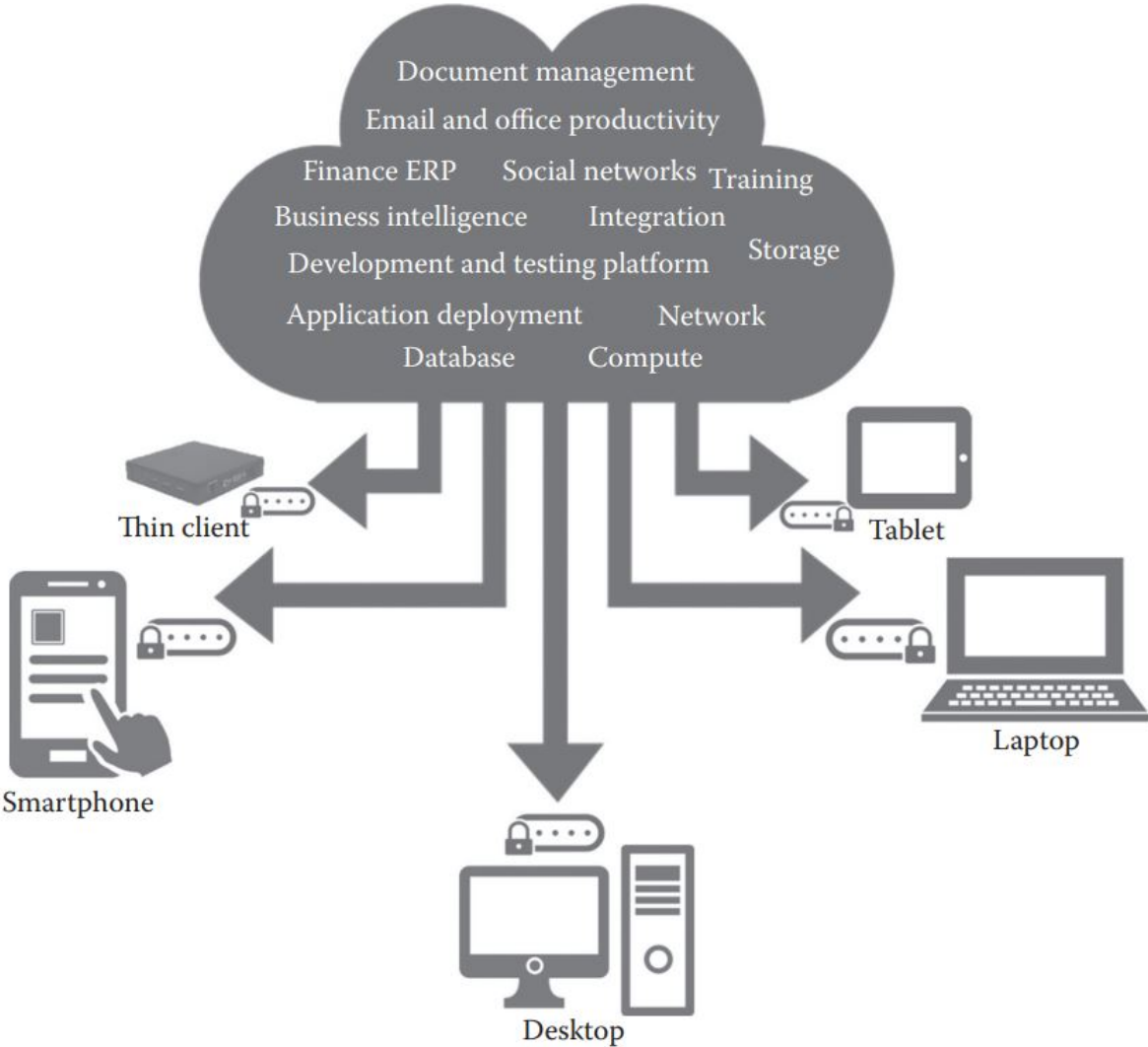


Figure 2.1: Cloud Computing [1]

### 2.3 Types of cloud computing :

There is four cloud computing types :

#### 2.3.1 Public Cloud :

Public clouds are owned and operated by third-party cloud service providers, which deliver their computing resources, like servers and storage, over the Internet. Microsoft Azure is an example of a public cloud. With a public cloud, all hardware, software, and other supporting infrastructure is owned and managed by the cloud provider .We access and manage these services using a web browser [26]. multi users share this cloud . According to NIST [27] : " The cloud

infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider ".In other words , this cloud is available for the general public , which means less security than the private and hybrid cloud , but this type is cheaper than the two other types . the figure 2.2 below shows the model of a public cloud

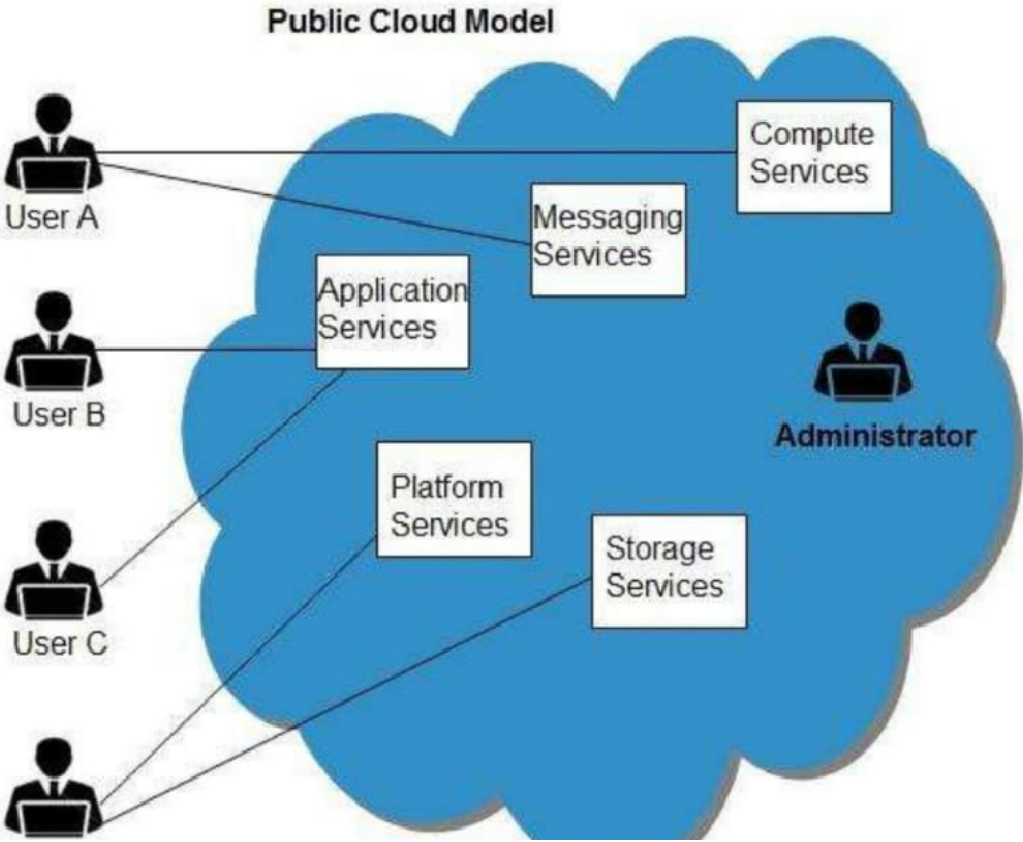


Figure 2.2: Public cloud [28]

In addition to Amazon Web Services(AWS) there is also Google and Microsoft that are providing Cloud services and the figure 2.3 below shows the Top leaders public cloud service providers by Gartner



Figure 2.3: Public cloud provider top leaders [29]

### 2.3.2 Private Cloud :

Most companies that have strict rules about privacy prefer private cloud rather than public cloud, to have full access to manage all its services and their data. According to Microsoft : "A private cloud refers to cloud computing resources used exclusively by a single business or organization. A private cloud can be physically located on the company's on-site datacenter. Some companies also pay third-party service providers to host their private cloud. A private cloud is one in which the services and infrastructure are maintained on a private network"[26].

Multi users share the public cloud ,that is why companies were afraid of there stored data and information , the private cloud refers to non shared cloud , and it is used for the sensitive data , it is more secured than the public cloud but it is expensive compared to the public cloud costs .

Even if the majority of cloud providers have an on premise solution, but in general big organizations use solution like VMware or HPE, or even run their own infrastructure using an open source tools like openstack [30] [31] . the figure 2.4 below shows the model of a private cloud .



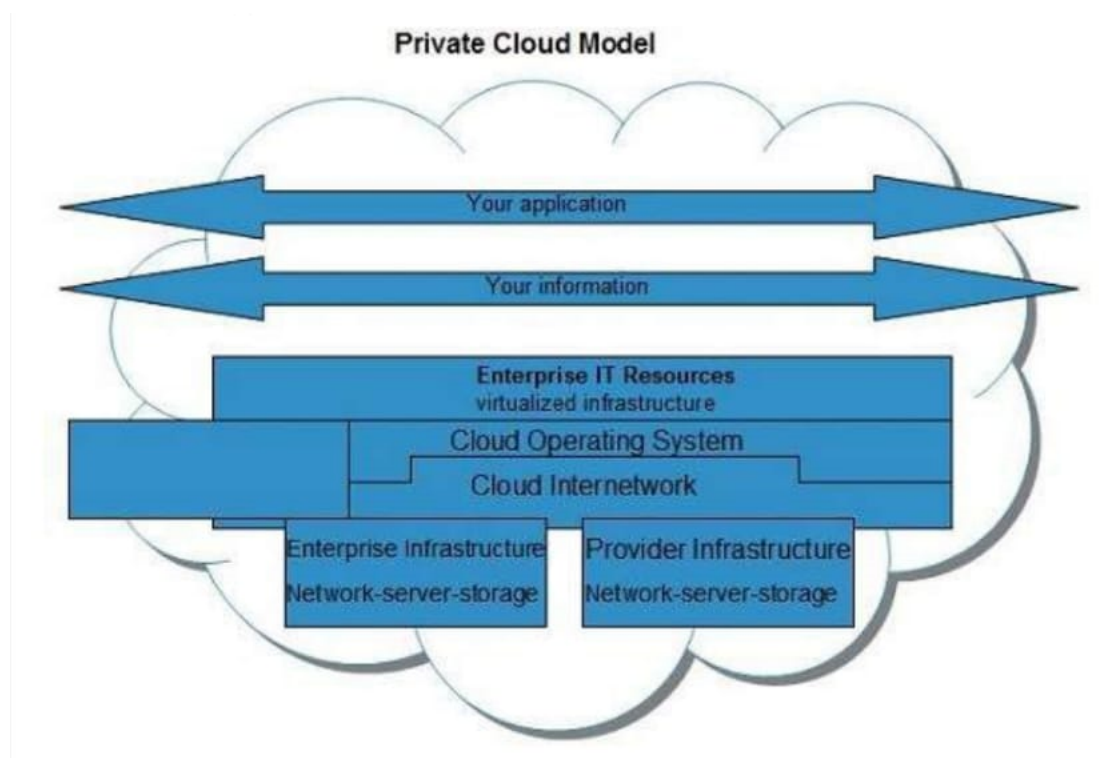


Figure 2.4: Private cloud [28]

### 2.3.3 Community Cloud :

According to NIST [27] : " The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises ".

### 2.3.4 Hybrid Cloud :

Generally speaking the companies that prefer to get the best from private cloud for hosting its sensitive data and public cloud for scalability choose hybrid cloud.

According to Microsoft [26] : "Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. By allowing data and applications to move between private and public clouds, a hybrid cloud gives your business greater flexibility, more deployment options, and helps optimize your existing infrastructure, security, and compliance."

According to NIST [27] : " The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)." the figure 2.5 below shows the model of hybrid cloud

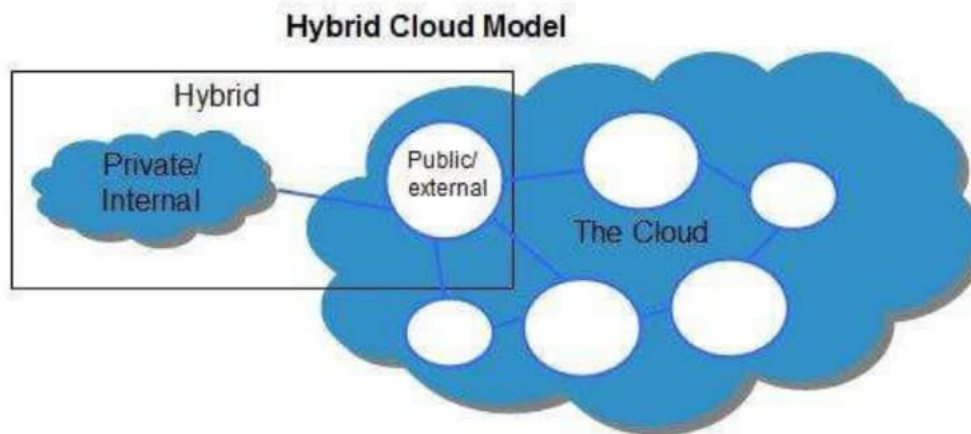


Figure 2.5: Hybrid Cloud [28]

## 2.4 Types of cloud computing services :

### 2.4.1 Infrastructure as a Service (IaaS) :

According to Amazon [24] : "IaaS contains the basic building blocks for cloud IT. It typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS gives you the highest level of flexibility and management control over your IT resources. It is most similar to the existing IT resources with which many IT departments and developers are familiar. " In this type the Cloud service provider offers computing resources such as processing , network and storage , the client is charged to install an operating system and manage the whole system and applications without caring about the hardware control . The figure 2.6 below illustrates Infrastructure as a Service

### 2.4.2 Platform as a Service ( PaaS ) :

According to Amazon [24] : "PaaS removes the need for you to manage underlying infrastructure (usually hardware and operating systems), and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application."

Platform as a service includes IaaS in addition it is in charge of controlling the operating system and its needs like updates and patches, this service is mostly used to deploy applications .

The figure 2.7 below illustrates Platform as a Service

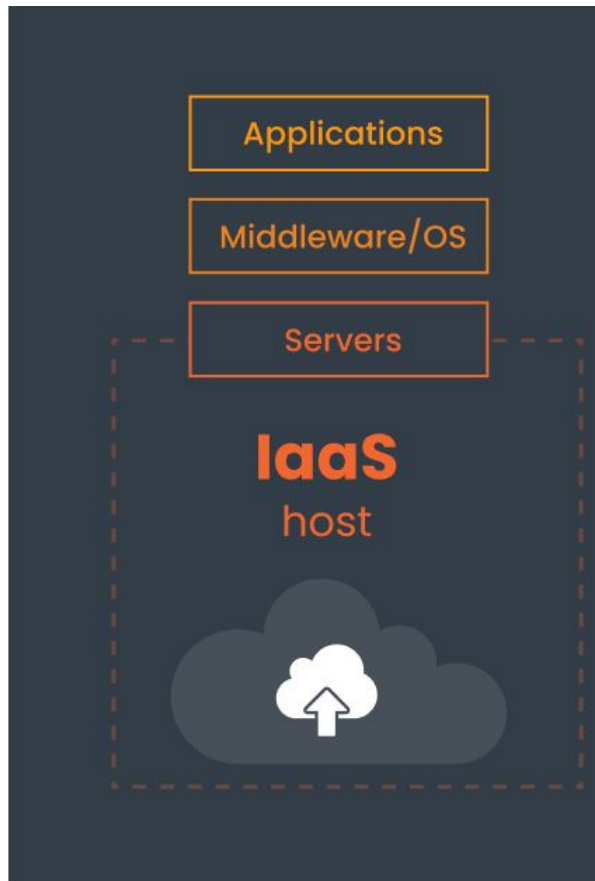


Figure 2.6: Infrastructure as a Service ( IaaS ) [32]

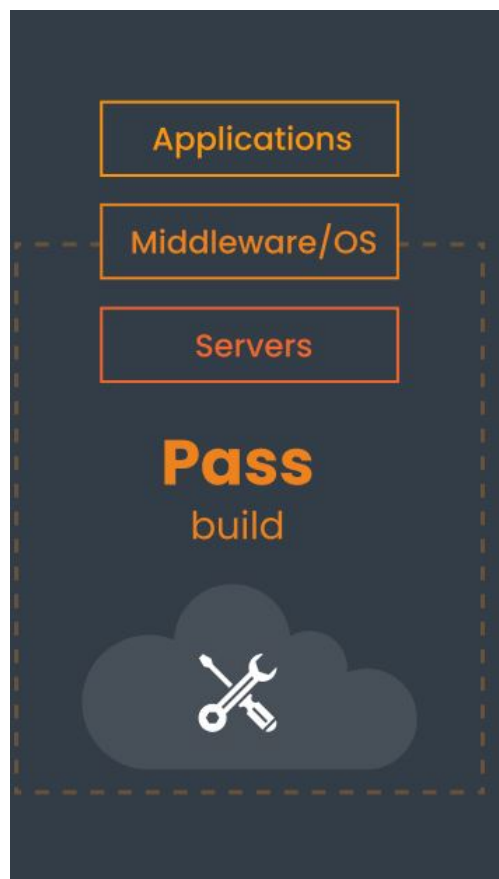


Figure 2.7: Platform as a Service ( PaaS ) [32]

### 2.4.3 Software as a Service ( SaaS):

According to Amazon [24]:"SaaS provides you with a complete product that is run and managed by the service provider. In most cases, people referring to SaaS are referring to end-user applications (such as web-based email). With a SaaS offering, you don't have to think about how the service is maintained or how the underlying infrastructure is managed. You only need to think about how you will use that particular software. "

In other words , it offers an application ready to consume , the client will not manage the network , operating system or the storage , the application is accessible from different devices through a web browser or a program interface .

The figure 2.8 below illustrates Software as a Service

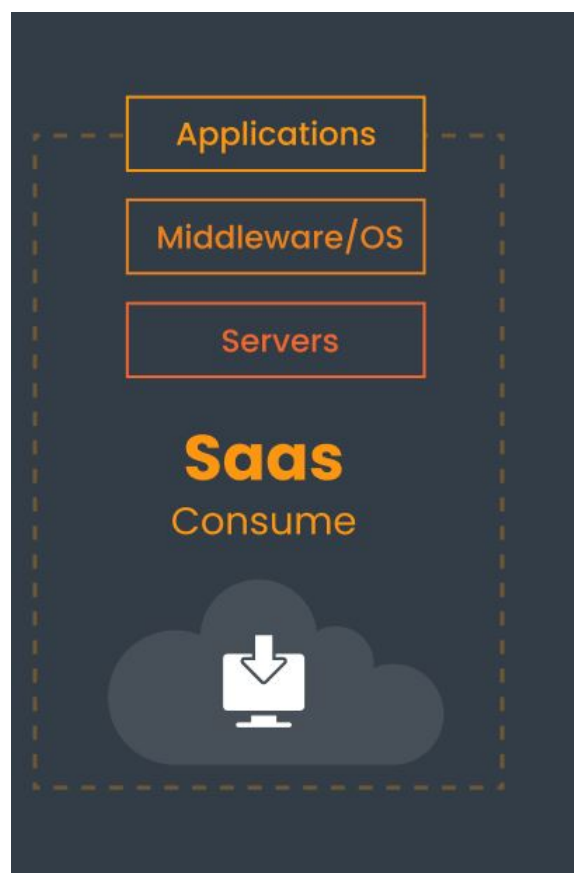


Figure 2.8: Software as a Service ( SaaS ) [32]

## 2.5 Cloud Essential Characteristics :

Cloud computing has five essential characteristics defined by NIST [27]:"

### **On-demand self-service :**

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

### **Broad network access:**

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

**Resource pooling :**

The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

**Rapid elasticity :** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

**Measured service:**

Cloud systems automatically control and optimize resource use by leveraging a metering capability<sup>1</sup> at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service ". the figure 2.9 below illustrates the cloud essentials.

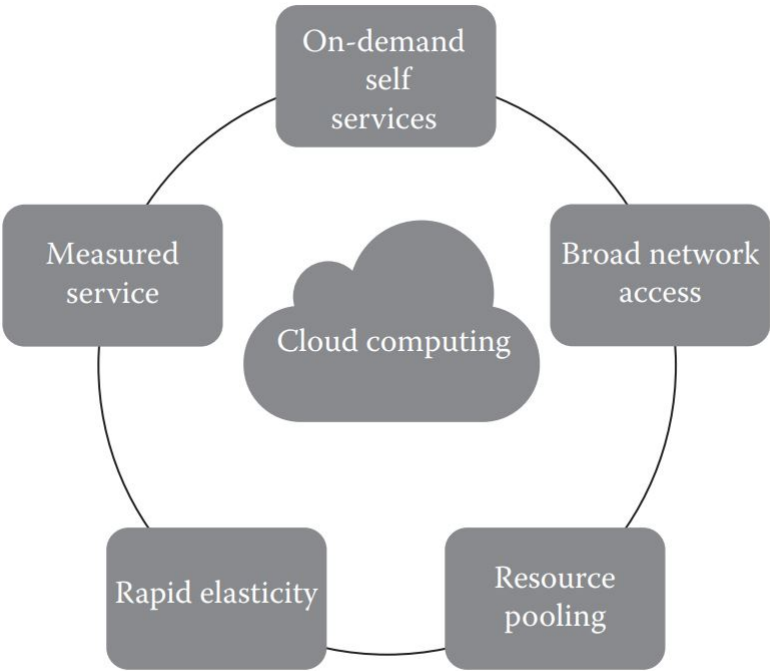


Figure 2.9: Cloud Essential Characteristics [1]

## 2.6 Serverless and microservices :

Serverless is a new cloud computing technologie, where the developers avoid managing hardware ressources (like servers) and focus only on the application code, this concept has created a new cloud computing service which is FAAS (Functicon as a service) , the developers only have to generate the application code , the ressources will be managed by the cloud service provider.[33]

This technologie has created a new development architecture known as " Microservices " , where an application is devided into multiple services , each service is separated from the others so it is completely different compared to the monolithic architecture where the application is one piece , the microservices is based on containers , in other word each service represents a container ,the different services communicate through the network, this communcation is based on the REST API .

The figure 2.10 below shows the diffirence between Microservice architecture and Monolithic architecture .

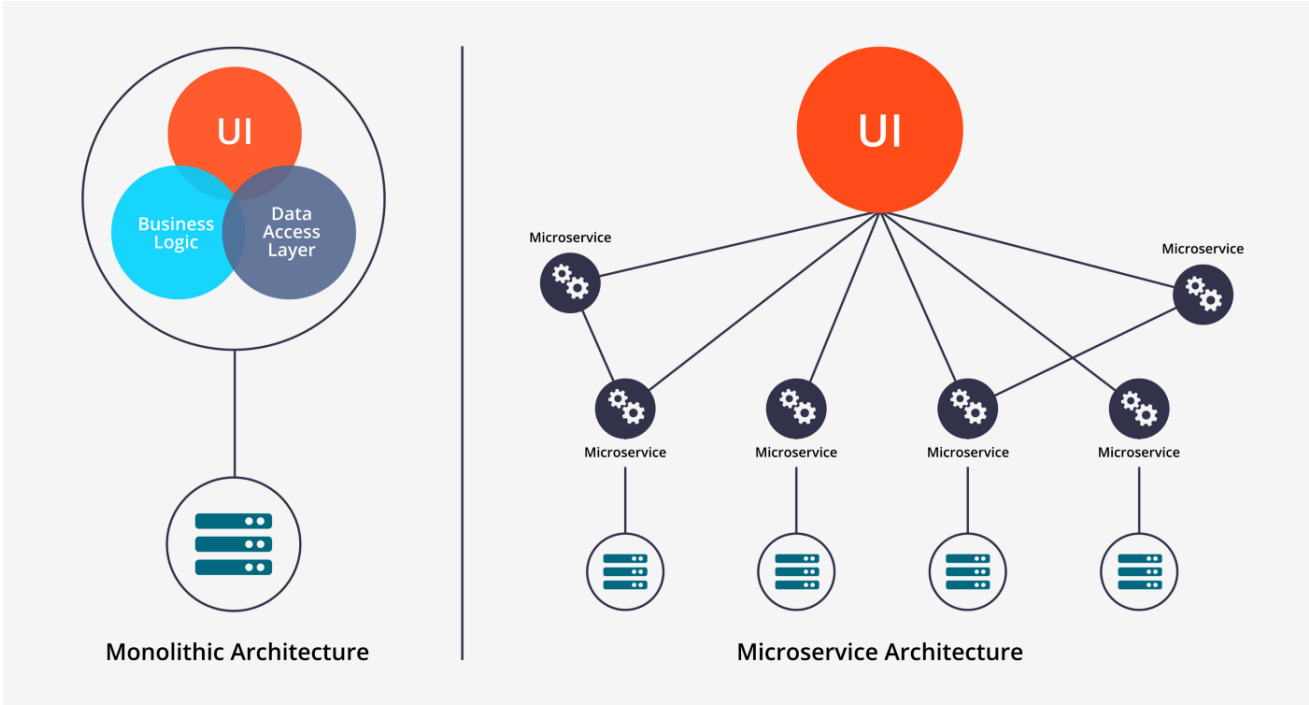


Figure 2.10: Microservice vs Monolithic architecture [34]

## 2.7 Conclusion :

Cloud computing is a computing paradigm based on virtualization , it has solved lot of problems , it has three types public , private and hybrid it offers three services IaaS ,PaaS and Saas and due to containerization new services have been appeared like Faas.Cloud computing has five essential characteristics defined by NIST, due to cloud computing and containerization , developers can focus more on coding and don't worry about the server management.Nowadays organizations have saved a lot of money due to cloud computing , they are not obliged to buy

any hardware infrastructure and they don't have to pay a lot of computer engineers to take care of their data and maintain their hardware resources .





## Chapter 3

---

# Containers Security in the cloud

---

### 3.1 Introduction :

The entrance of cloud computing to our life has solved lot of problems, participated in the creation of new technologies and facilitate IT development. Lot of organizations migrated its data to the cloud, however cloud providers are migrating into using containerization instead of the classic virtualization and knowing that containers are less secure than virtual machines because of sharing the same host operating system's kernel make them good target for lot of hackers, which forces the IT world to focus more on containers and data security on the cloud.

So what is security in general ? What are security pillars ? and how can we guarantee data and container security in the cloud ?

### 3.2 What is security ? :

Information security is protecting our information or host information against unauthorized access, use, disclosure, disruption, modification, or destruction from hackers or from any one who wants to misuse it , in other words protecting our data and our hardware resources from attackers outside and inside our network, natural disaster , power failure ,adverse environmental conditions ,theft or vandalism and protecting the users and the clients who are involved on our operations .till now there is nothing secured in other words there is no system perfectly secure , there is always a vulnerability which can attackers exploit to hack a system , and the human is considered as the weakest target in the information systems because the most of attacks are internal attacks.

So how can we secure our systems ? What are the concepts of information security ?.

The primary concepts of security are " Confidentiality, Integrity, and Availability " commonly known as " the CIA triad",so when we talk about a secure system we have to guarantee confidentiality, integrity, and availability[35].

### 3.3 The CIA Trade :

Confidentiality, integrity, and availability are the base of information security. Confidentiality is nearly the privacy but it is not the same ,in other words confidentiality is a part of privacy and it means protecting our data from unauthorized access i.e protect our data from those who are not allowed to access it [35]. According to Andress Jason "Integrity refers to the ability to prevent our data from being changed in an unauthorized or undesirable manner. This could mean the unauthorized change or deletion of our data or portions of our data, or it could mean an authorized, but undesirable, change or deletion of our data. To maintain integrity, we not only need to have the means to prevent unauthorized changes to our data but also need the ability to reverse authorized changes that need to be undone.We can see a good example of mechanisms that allow us to control integrity in the file systems of many modern operating systems such as Windows and Linux. For purposes of preventing unauthorized changes, such systems often implement permissions that restrict what actions an unauthorized user can perform on a given file. Additionally, some such systems, and many applications, like databases, can allow us to undo or roll back changes that are undesirable.Integrity is particularly important when we are discussing the data that provides the foundation for other decisions. If an attacker were to alter the data that contained the results of medical tests, we might see the wrong treatment prescribed, potentially resulting in the death of the patient"[35]. Availability is the last concept of the triad , it refers to guarantee the access to our data when we need it at any time , such problems can appear from power loss, operating system or application problems, network attacks, compromise of a system, or other problems. When such events are caused by an external party, like an attacker, they are commonly known as a denial of service (DoS) attack. losing this leg of the CIA triad makes our system useless , because we can't access any data and we can't even use our system ,so to secure the system we have to make sure that it will be available at any time we need to access it, and to achieve that we can use the high availability technologies and techniques [35].

The figure 3.1 below shows the CIA trade.



Figure 3.1: The CIA Triade [36]

### 3.4 Security in Cloud Computing :

The security challenges between cloud computing and traditional computing differs in just a little points , cloud computing security has a shared responsibility model , which means that security is shared between the user and the cloud service provider , in the IaaS , the client is supposed to secure the operating system , its applications and services, code and data , where the cloud service provider is charged to secure the hardware part , in the PaaS clients are supposed to secure their applications , for example securing code from the known vulnerabilities , use security best practices,and securing data . The cloud service provider is charged to secure the platform ,in other words ,securing the operating system and its running applications, checking for updates and scanning vulnerabilities in the platform .In the SaaS model , the user has to secure his data and the security of other parts is the responsibility of the cloud service provider .the figure 3.2 below illustrates the shared responsibility model by microsoft.

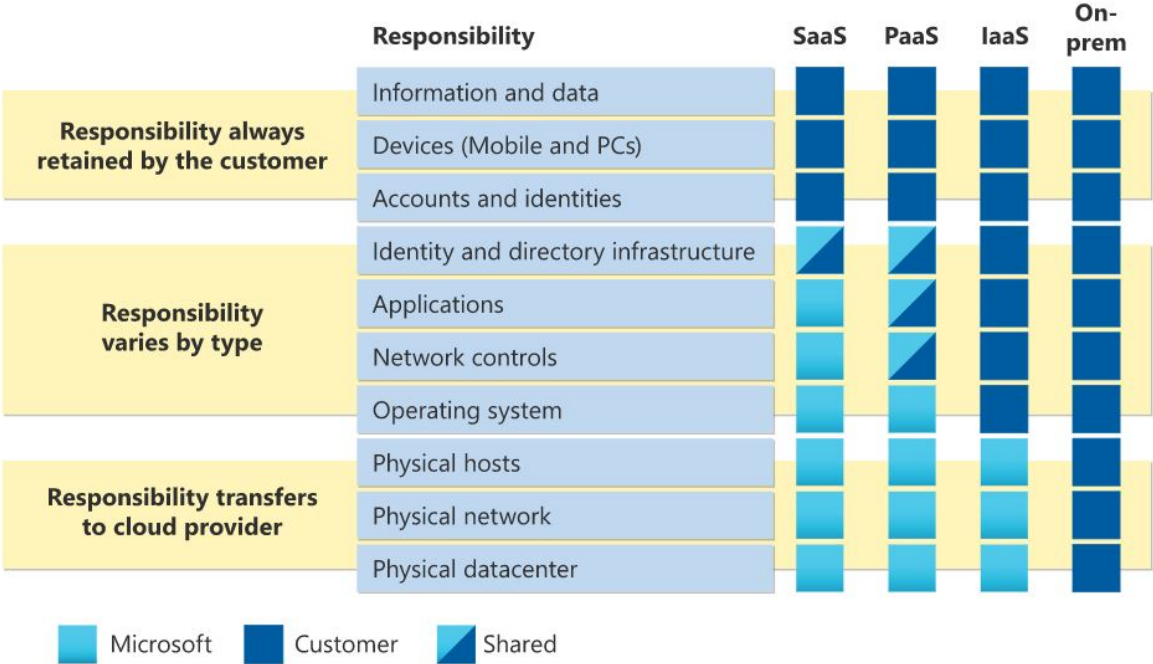


Figure 3.2: shared responsibility model [37]

Lot of users and organizations have doubts of the shared responsibility model , that is why choosing a professional cloud service provider can be a security factor , for exemple " Google" has more than 850 security experts , they have bug bounties programs , penetration testing programs , lot of internships and a high budget for security investments , the same thing for Amazon ( AWS ) and Microsoft ( Azure ) . In addition, they offer tools to facilitate securing data , applying security and network policies , they also offer monitoring tools .However those companies are targets for lot of hackers , knowing that there is no a perfect security in the IT world , we have to take additional steps in securing our applications and data even in the cloud , we have to encrypt our data before storing them in the cloud which means data will not be readable even if they are stolen by an external party , we can implement access control like RBAC

model ( Role-based access control ) and use authentication techniques like the " Two-factor authentication (2FA)" [21] [38] [39], the figure 3.3 below shows 2FA model .



Figure 3.3: Two-factor authentication [40]

The figure 3.4 below shows RBAC model

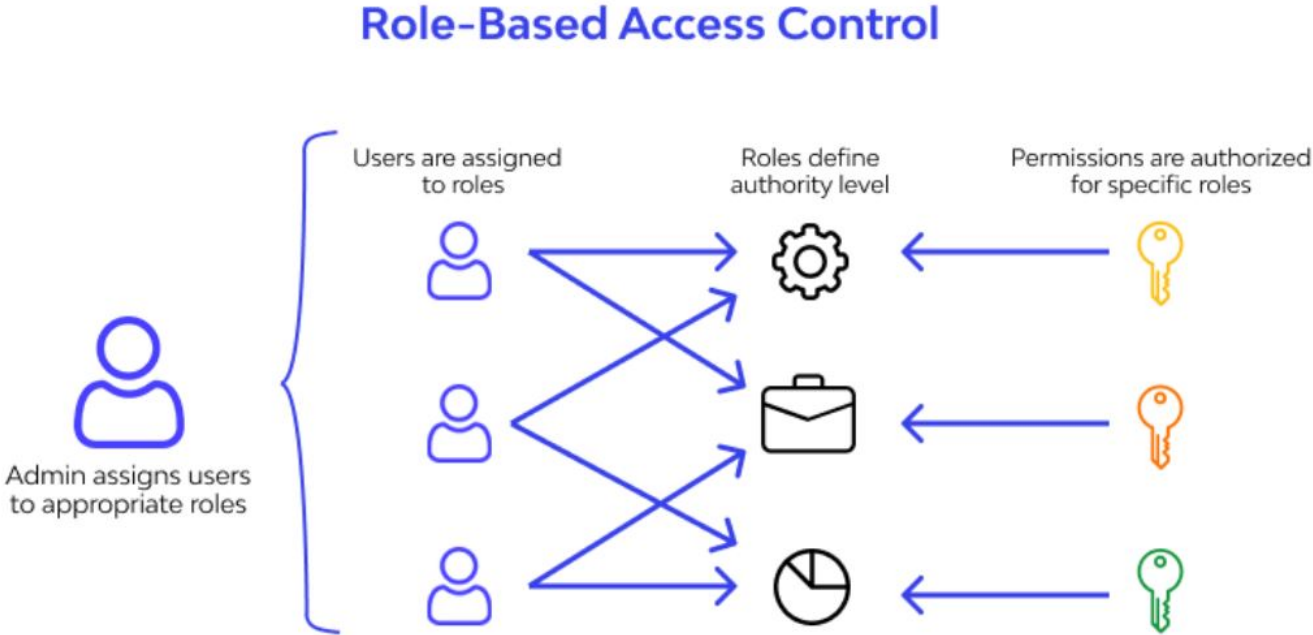


Figure 3.4: RBAC model [41]

### 3.5 Vulnerabilities and attacks on cloud :

Knowing that cloud computing is based on virtualization techniques , the services running on the cloud can present different vulnerabilities in the different abstraction layers, below we will present some attacks and vulnerabilities that can occur on each layer :

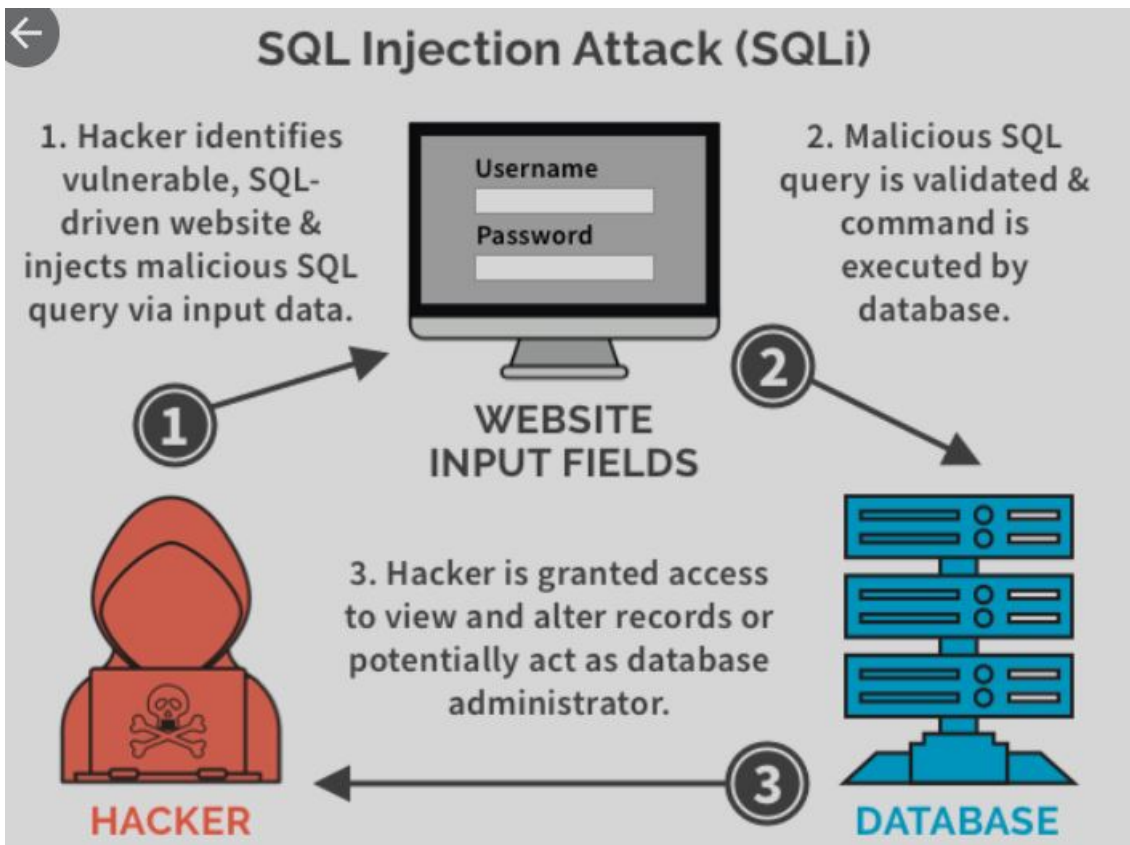


Figure 3.5: SQL Injection Attack [42]

- **Application Layer :** Many applications on the cloud are web based applications which make them vulnerable to a lot of web attacks such as SQL injection which allow an attacker to inject a malicious sql code that will be executed on our SGBD , this attack can destroy our databases which means losing our data [21]. the figure 3.5 illustrates the process of SQL injection attack . Another famous attack can appear in the application layer , it is cross site scripting attack known as " XSS ", this attack allow a hacker to inject scripts ( javascript) this scripts can be executed in victim machines and sent their data into the attacker[21] , the figure 3.6 illustrates XSS attack .
- **Operating System layer** Cloud security can be compromised through operating system like using buffer overflow attacks that are famous against applications and operating systems developed in c or c++ , this attack allows hacker to play in the host memory and oblige the system to run another instruction in more details a jump instruction is used to execute a malicious code , this attack can be also used to allocate a lot of memory .that's why we have to control inputs , another attacks and vulnerabilities can be exploited and compromise the operating system through its services and applications , to avoid this attacks we have to do a vulnerability scan and update the operating system[21] .
- **Hypervisor, Storage, Hardware, and Network :**

As we know , cloud computing is based on virtualization , any virtual machine image or hypervisor vulnerability can be used to compromise the system , or losing the isolation

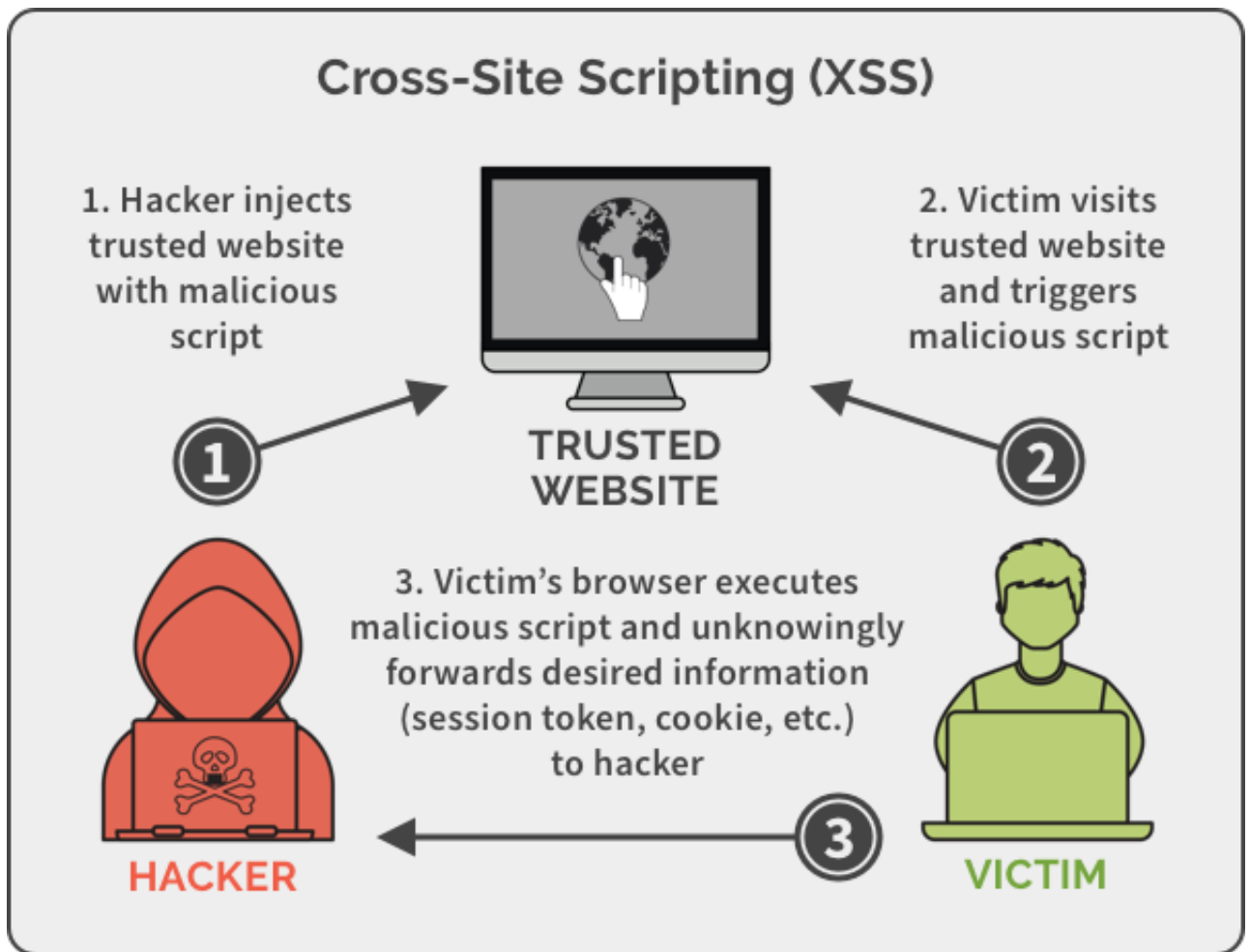


Figure 3.6: Cross site scripting Attack " XSS " [43]

between virtual machines which make our data accessible to another users that share with us the cloud infrastructure , cloud computing can get hacked through hardware trojans , malwares and through network protocol vulnerabilities like using syn flood attack (DOS attack) , this attack is based on the tcp protocol , syn packets will be sent to the server without completing the three way handshake which can cause a "Denial of Service" [21] . The figure 3.7 illustrates the syn flood attack .

The figure 3.8 below illustrates different attacks in different layers .

### 3.6 Containers Security :

Virtual machines are heavy compared to containers ,that is why lot of cloud service providers moved to containerization ,however containers are less secure than VMs because they share the same host kernel and are connected directly to the host hardware without any separation level , which means a compromised container can easily affect the host which will affect another containers , in addition , some containers require high privileges ( deep level of authorization ) to operate [45] .In the cloud , securing our containers is our responsibility that is why we have to pay attention to four points in containers security:



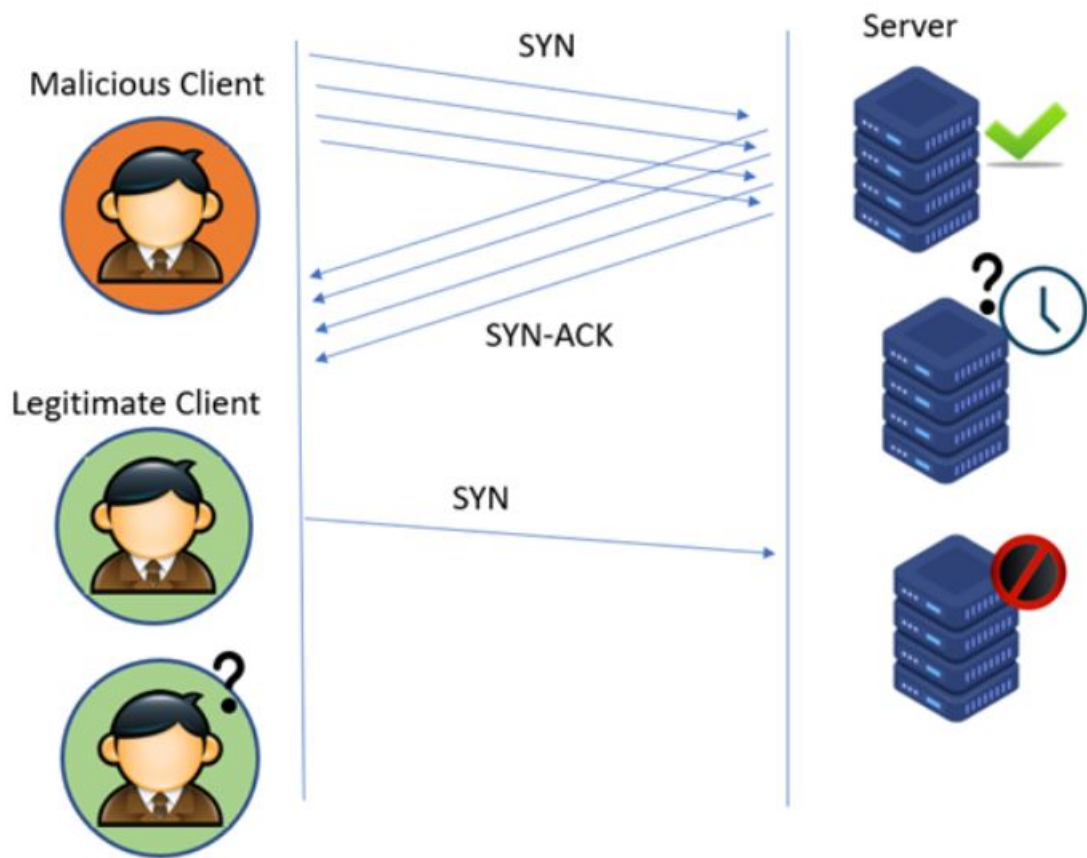


Figure 3.7: SYN Flood attack [44]

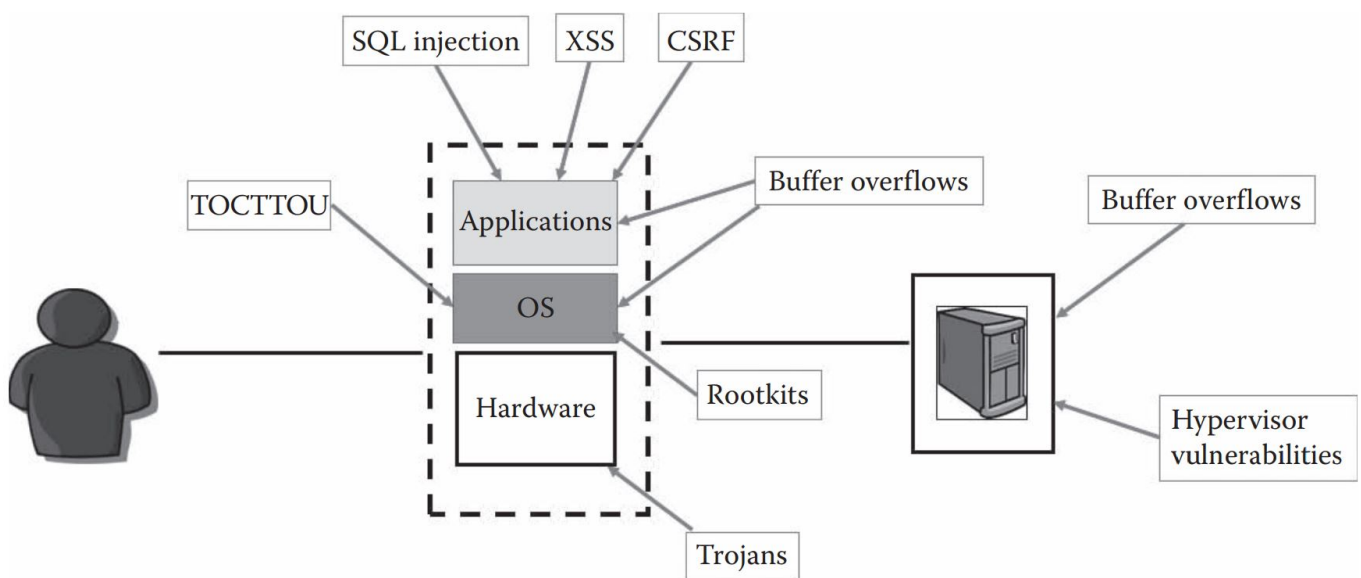


Figure 3.8: some possible attacks in different cloud layers [21]

- **We have to secure containers from running applications**(applications inside the container ) . Attackers can exploit a vulnerable application to hack our containers , in addition they can get access to the host through the hacked container.[9]
- **We have to secure containers from other containers** : containers communicate through the network i.e they are vulnerable to network attacks like DOS. [9]
- **we have to secure containers from host os** : containers can be easily affected by the host due to hardware issues or even host attacks . [9]
- **we have to secure host from containers** containers are connected directly to the host hardware they can use all resources which will affect other host applications and services.[9]

### 3.7 Studied works :

Sultan et al [9] have done containers security analysis and how can we secure them in four use cases : (I) protecting a container from applications inside it, (II) inter-container protection, (III) protecting the host from containers, and (IV) protecting containers from a malicious or semi-honest host , they presented some vulnerabilities and attacks and how can we secure our containers against those attacks , they also analysed some related works.

Tomar et al [46] have implemented a Dos attack on docker , they also talked about seven attack scenarios (Attack Scenario I: Application To Container Attack, Attack Scenario II: Container To Container Attack,Attack Scenario III: Container To Host Attack, Attack Scenario IV: Host To Container Attack, Attack Scenario V: Container To Docker Engine Attack,Attack Scenario VI: Host To Docker Engine Attack,Attack Scenario VII: User (On Internet) To DockerEngine Attack) and fifteen different attack can be used against containers . In the end they have implemented a Dos attack on docker , they installed docker in a kali linux machine and they started from the same host a Dos attack using the tool H3ping with the flag "-flood" , we are not satisfied of the result of this attack, the authors proved that when they start the dos attack the cpu usage get increased to the max 100 % ,but they didn't show the h3ping final result which proofs how many packets have been sent and how many packets the victim has received, so we tried to implement this attack and we noticed that the tool h3 ping can increase the cpu usage to the max without sending packets to the victim and when we stop the attack the received packets are null (0 packet received by the victim ) then we reimplemented the attack using two machines (attacker machine and victim machine) and we have got the same result .

Pankaj Mendki [47] proposed some container security solutions using blockchain technology like using Blockchain based image registry to insure image security.



Kommula and al [45] proposed network solution , in their study they have notices that docker doesn't encrypt the network traffic , and the bridge interface docker0 doesn't have the ability to identify internal or external traffic , so they propose create new bridge interface "ldocker0" this interface can distinguish between internal and external traffic using IA , helping us to identify spoofing attacks and prevent network traffic snooping.

The figure 3.9 illustrates our comparison between the studied works .


Works	Sultan et al [90]	Tomar et al [91]	Pankaj Mendki[75]	Kommula and al [68]
Description	They have analysed containers security, they have compared and tested some related works.	They have analysed containers security in docker and implemented a dos attack.	he has proposed some containers security solutions based on blockchain technology.	They have proposed one solution based on machine learning .
Do the work helps Securing containers from host	yes	yes	No	yes
Do the work helps Securing containers from running applications	yes	yes	yes	yes
Do the work helps Securing containers from other containers	yes	yes	yes	yes
Do the work helps Securing host from containers	yes	yes	No	yes
Our point of view	Sultan et al have done a great job, their analysis were very helpful in our project realisation, we were satisfied from their work.	The analysis and the attack threat model that has been proposed in this work are great and helpful.However we aren't satisfied from the results of the dos attack, we think that the increase of cpu usage is caused by the tool h3ping and not the dos attack .	The analysis and the proposed solutions can be implemented and used to secure containers .	The result and the analysis are perfect , we think that the solutions can be implemented and helps to secure containers in the four study cases, we were very satisfied form their work. 

Figure 3.9: Comparing studied works

### 3.8 Conclusion :

Cloud service providers are in migration to containerization which is a new evolving technology and due to its architecture securing containers is a new challenge to the IT security world especially in a cloud environment which entered our lives and organizations are depending on its services .



## *Chapter 4*

---

# **Conception :**

---

### **4.1 Introduction :**

Containerization is the future of cloud computing and applications development. We discussed in the previous chapters about cloud computing and containerization , in this chapter we will see how we can secure our containers . and what are the steps that we should follow , to get a secure environment , and what should we do after the deployment of the containers and after the set of the security policies.

### **4.2 Problematic :**

As we know the containers are the future of the cloud computing and applications developments, and the security is the major concern for any developer or for any company , they have to care about their stored data and to hide them from other companies, and as a future view , the next war is a war of data , that is why securing the future of application development and cloud computing is a big concern and very important . Containers as a technology offer big advantages like portability , but they share the same host kernel which makes them vulnerable and a target for hackers , so what are the steps to secure containers ? and should we feel secure after making our security policy ? .

## 4.3 General architecture :

Overall this is an overview of the general architecture of our solution ( figure 4.1)

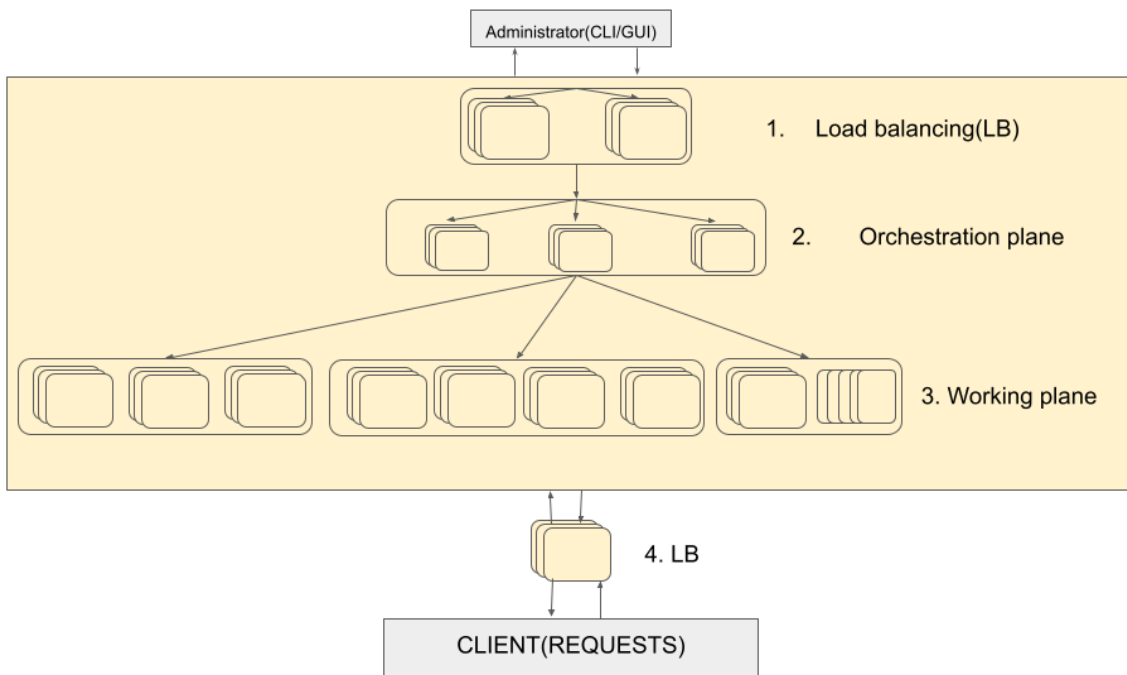


Figure 4.1: General architecture

The administration requests will be forwarded to our loaded balancing cluster with an active-passive configuration, via the virtual IP address assigned for the active one, to prevent a single point of failure

The orchestration plan will be the interface that manages and control our service deployment in different workers, control the communication and replicate to ensure a healthy system.

The work plan is the infrastructure where our services and application runs, due that we must secure:

- Containers from the host.
- Containers from running applications.
- Containers from other containers.
- Host from containers.

after applying those configurations we can expose our service via an external Load Balancer to our end users.

## 4.4 Securing containers from the host:

As we know , containers share the host's kernel so the first thing to secure our container is securing our host system .We have to scan the system and make sure that there are no

vulnerabilities ,no viruses , and the applications should be updated . The figure below 4.2 shows how can host affect containers .

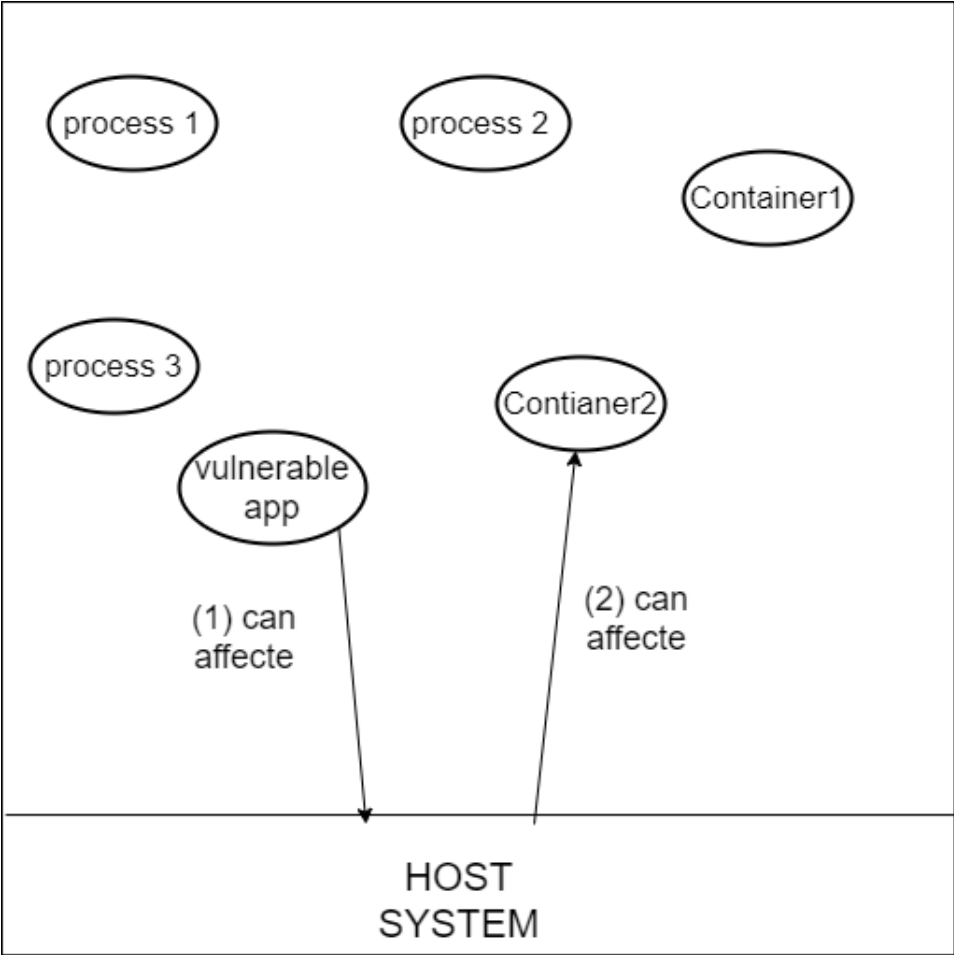


Figure 4.2: How can host affect containers

As we can see in the figure 4.2 , host can be affected by vulnerable applications which can affect our running containers , if the vulnerable applicaiton shuts our system down , our containers will be stopped and affected by this vulnerability .

the figure 4.3 below shows how can we secure our host system

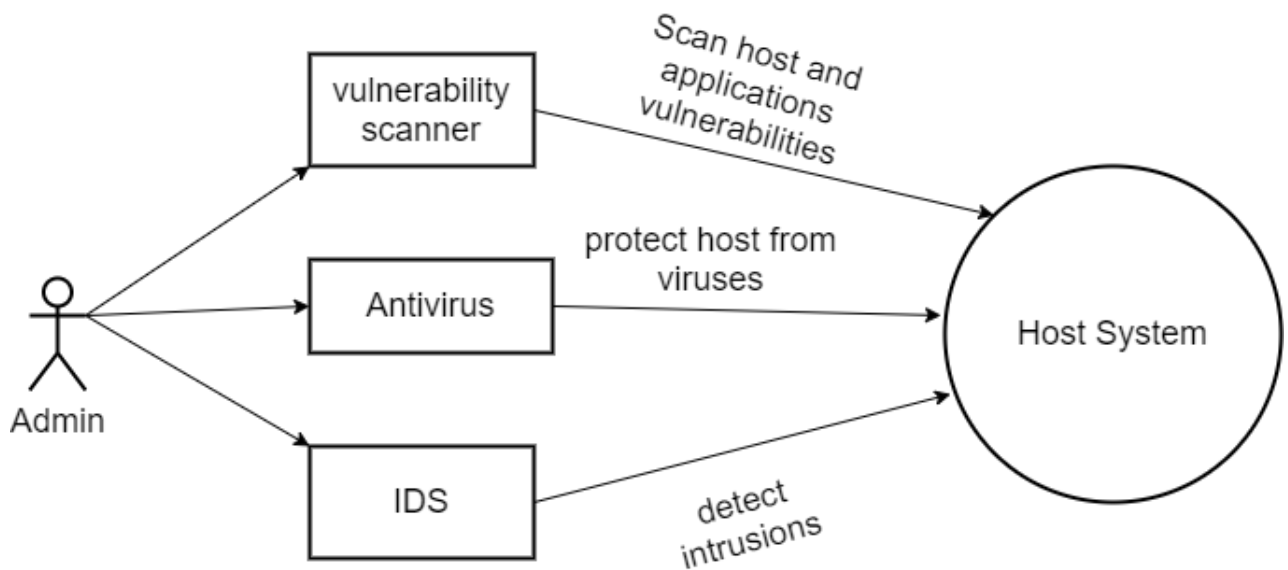


Figure 4.3: Securing host

As we can see in the figure 4.3 , we used vulnerability scanner to protect our host from vulnerabilities and from outdated applications that could have security issues , the scan process should be done periodically . Antivirus and ids can be implemented to detect viruses and intrusions in our system .

## 4.5 Securing containers from running applications:

To run any container in a container we have to run the image of this application , the image refers to the application ( or the source code of the application) including its dependencies and libraries . the images can be built easily and they can be vulnerable if the applications in this image are vulnerable . So to secure our container from the running applications we will use an image scanner that will check the running applications in our image and if they are vulnerable. The figure 4.4 below shows the process of scanning an image .

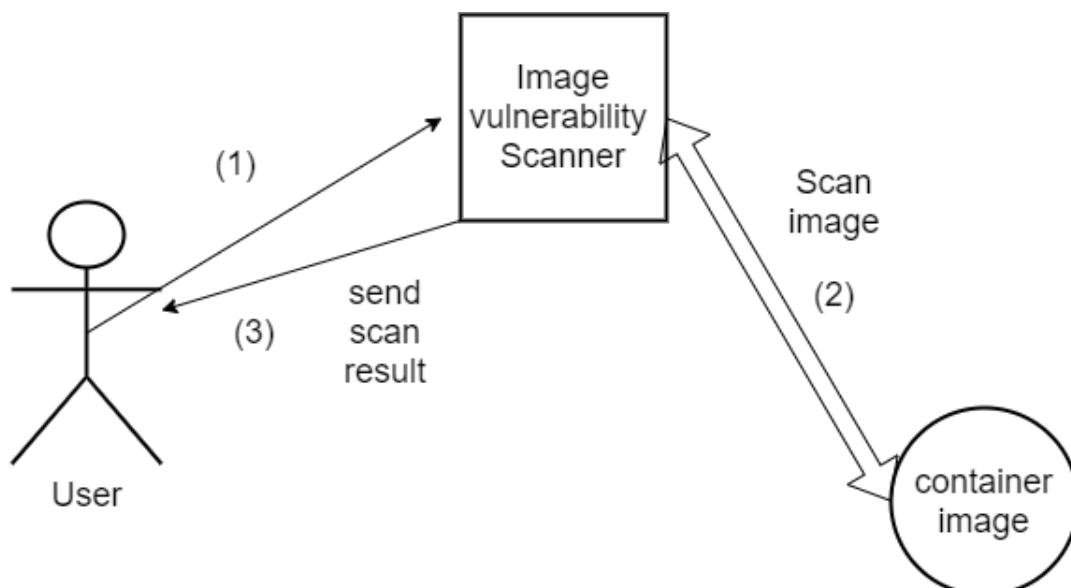


Figure 4.4: Securing container form running applications

The image vulnerability scanner , will scan our image and verify the installed dependancies and libraries if they have any vulnerabilities and will measure the risk of every vulnerability that has been found from low to critical . the figure 4.5 below show the process of attacking a container that runs a vulnerable image.

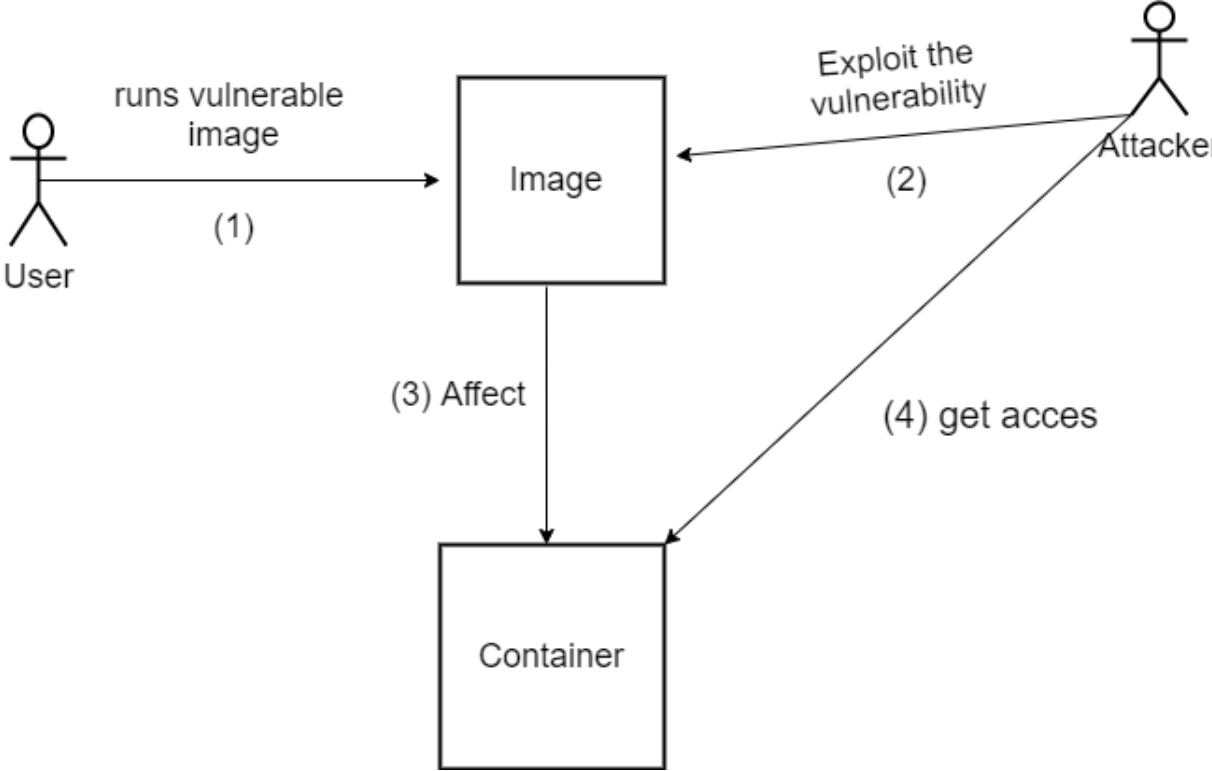


Figure 4.5: Attacking a container that runs a vulnerable image

In addition , we have created a script that check the file used to create an image (known as dockerfile) , we can find container’s libraries and dependencies , some of the installed dependencies could use high privileges , like using sudo in linux , some other applications could be installed which could be a threat to our system , for example ssh or even netcat , if we don’t need those packages , we don’t have to install it , so we listed some important commands on the dockerfile and we tried to detect them and warn the user about them .

### 4.6 Securing containers from other containers:

The containers communicate through the network so they are vulnerable to network attacks in the first place , secondly if one container runs a vulnerable image , he can easily affect other containers by affecting the host system rather than the running containers. So to secure the network we have to apply some network policies , this is possible using a "Container Network Interface ( CNI)" that are orchestrators plugins. the figure 4.6 shows an example of an orchestrator :

The orchestrator will be charged of creating and managing our containers that are running in the orchestrator nodes , that can be in different machines , orchestrator will provide availability

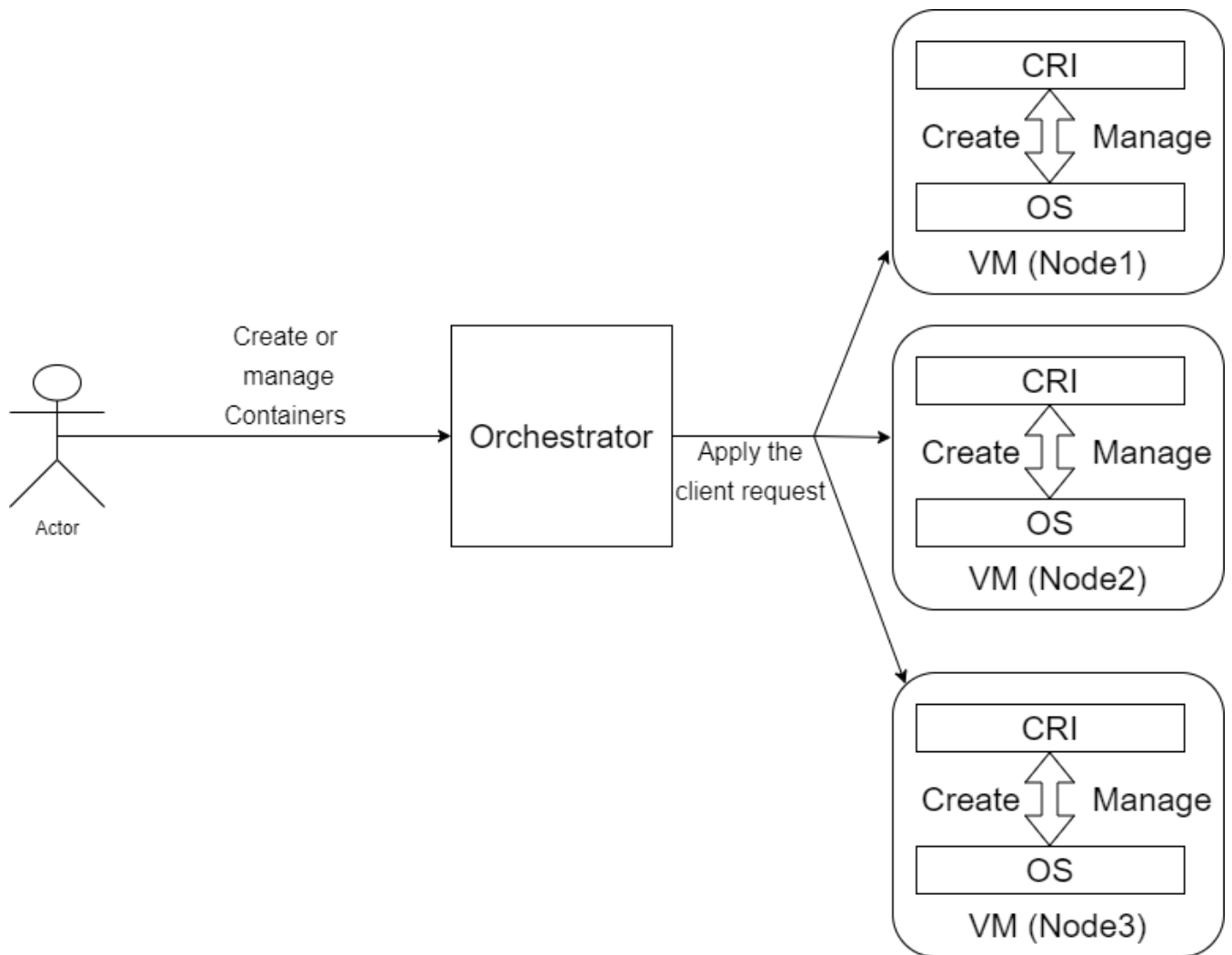


Figure 4.6: Exemple of an orchestrator

for example if one node that runs container A goes down , the orchestrator will run the container A again but in different node .orchestraot can guarantee confidentiality, authentication and authorization by generating certificates and keys.

The figure 4.7 shows the role of a CNI : CNI provides security in the transport layer (L4) which guarantee integrity , we can also apply some network policies , for example to DROP traffic that comes from specified IP , or from another container .

## 4.7 Securing Host from containers:

The host is also threatened with getting hacked through running containers because they share the host's kernel , if an attacker hacked a container he can try to escalate and gain some privileges to get access to the host system . The figure 4.8 below shows the process of hacking a host system through a running container.



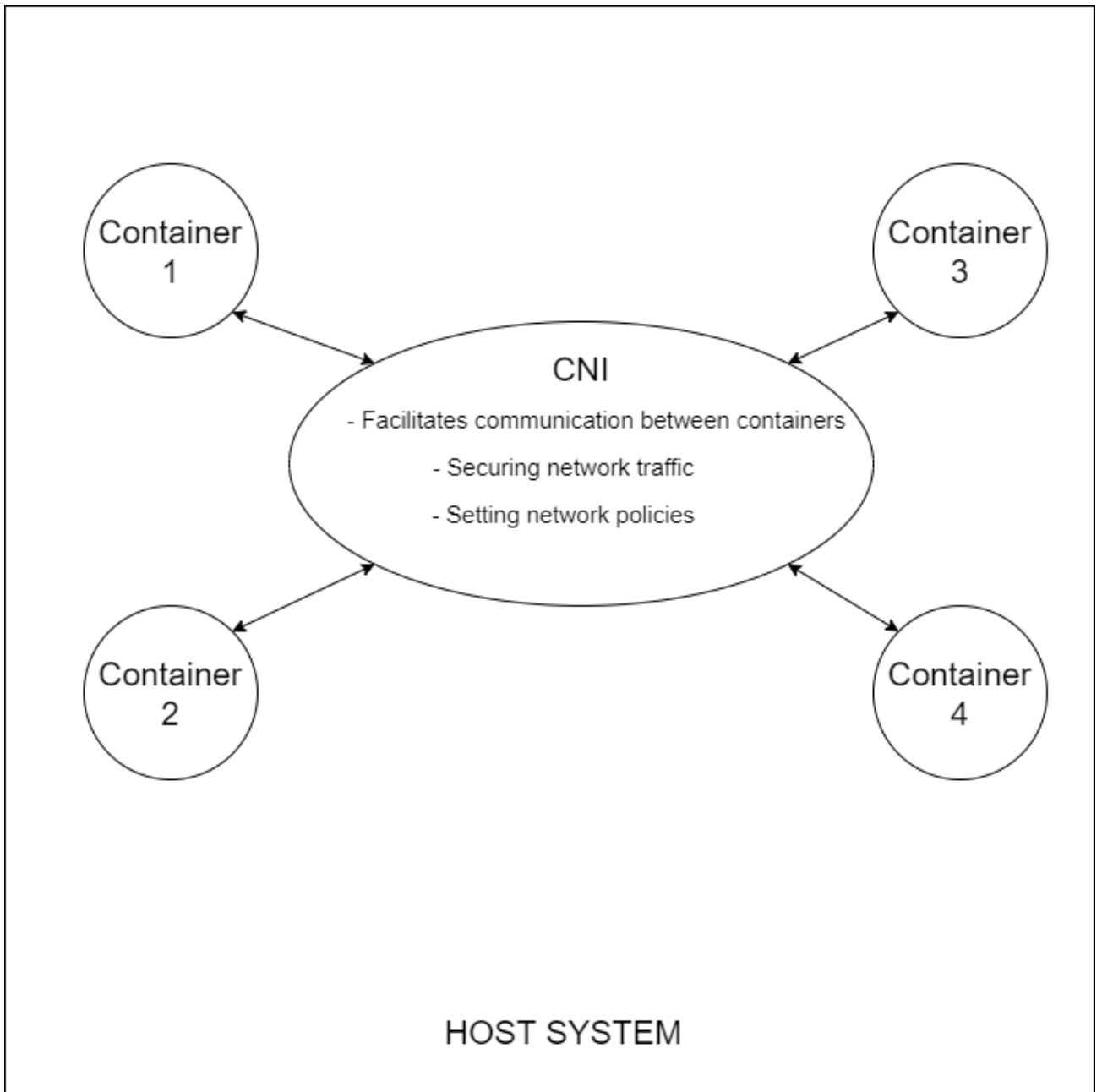
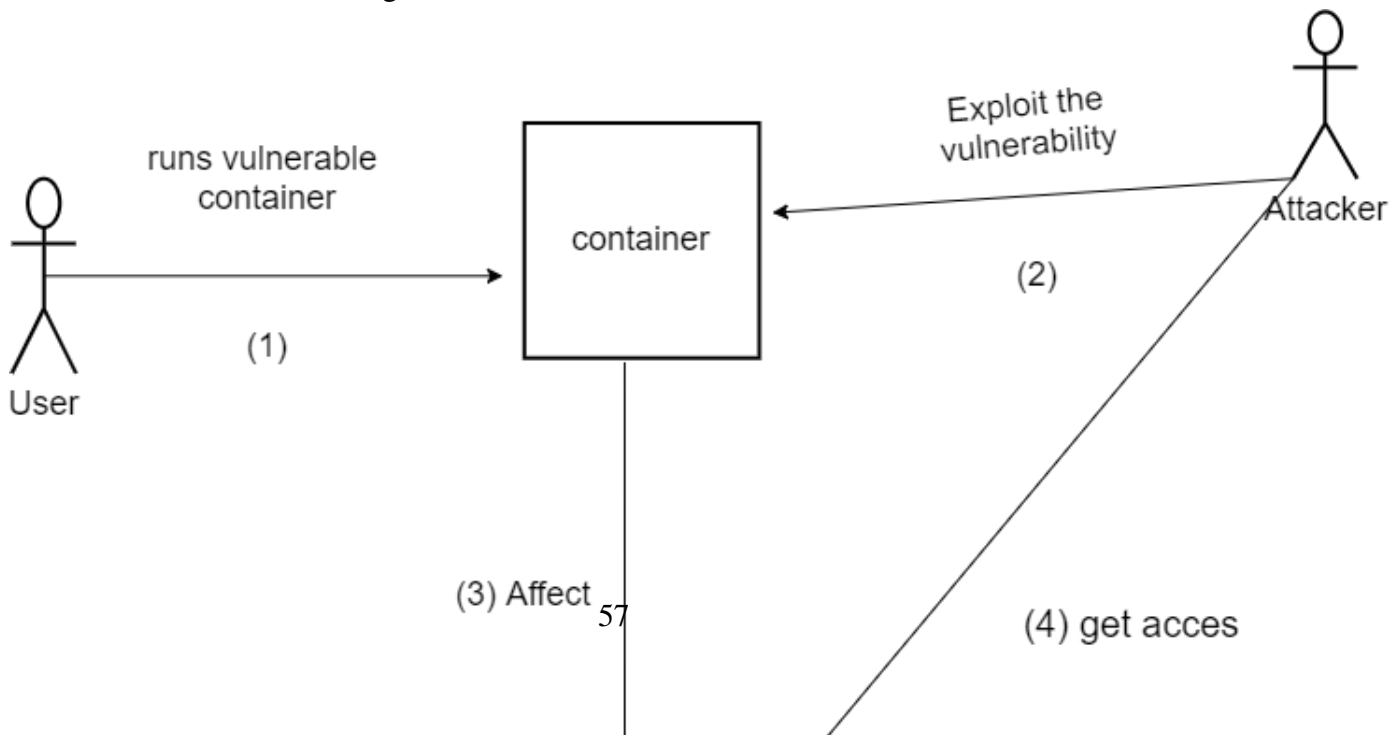


Figure 4.7: Container Network Interface



To secure our host we have to run rootless containers , or even "read only" containers in some cases to avoid giving high privileges to containers which can lead to host breaches. We also have to use antiviruses , periodic scans and monitoring tools .

Finally we have to guarantee high availability for our containers , using load balancers and Virtual ip address that will be attributed to a master machine , if the master goes down a slave machine takes the virtual ip address, using this technique we will not lose the orchestrator server connection and our containers shutdown time will be minimized to a little seconds our it will be null. The figure 4.9 below illustrates our solution.

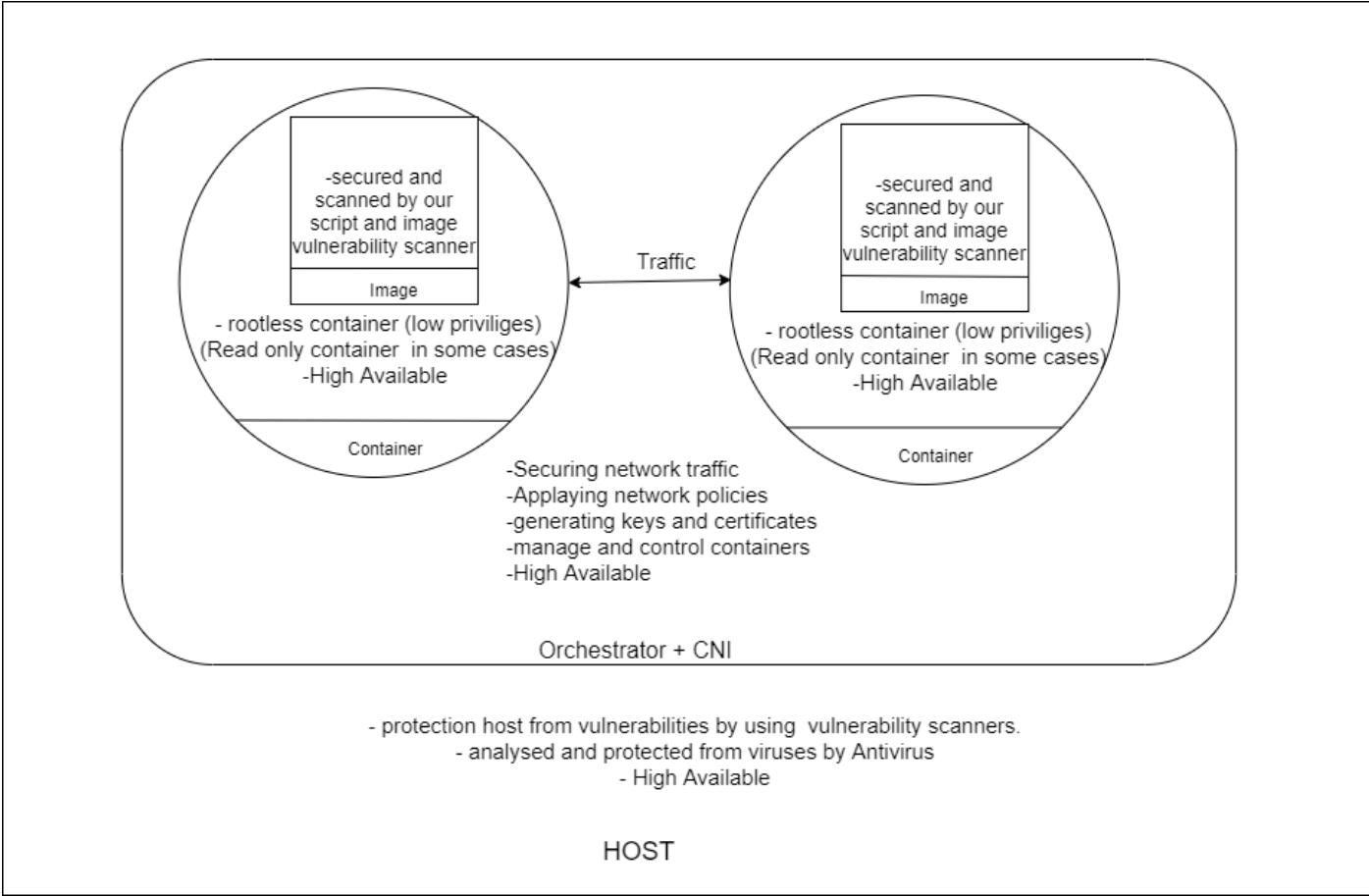


Figure 4.9: Our proposed solution

### 4.8 Conclusion :

In this chapter we have modelized our solution , we have explained some risks and how to secure our containers against them.

We have explained how to secure containers from host , from other running containers , from running applications and how to secure host from containers .

At the end we have presented high availability solution for our containers.

## Chapter 5

---

# Implementation :

---

### 5.1 Introduciton :

In the previous chapter, we presented the design of our solution.

In this chapter we will implement our solution and apply our security policies .

This chapter will include a part to specify the hardware and software resources used as well as screen shots that illustrate the steps of securing containers in cloud environment from host,running applications,other containers and finally securing host form containers .

In addition we will see how to implement a high availability solution .

### 5.2 The working space:

In this section we will see how we can secure our containers, we will start by describing the hardware resources used in our project , the environment and the tools used to accomplish our work and the reason behind choosing these tools . The table 5.1 below describe our hardware resources:

Hardware	Capacity
Motherboard	MSI H110M PRO-VD PLUS (MS-7A15)
CPU	Intel I5-6400 2.70 GHZ 4 Cores
RAM	8 GB DDR4
Storage	240 GB SSD and 1 TB HDD

Table 5.1: Hardware ressources used in our project

### 5.3 Docker :

Docker is the most famous "Container Runtime " , and because of docker's simple architecture, the containerization has been improved and being used by the famous companies .Docker permit to run containers and control their work , allows us to manage this containers and create images

from a source code also specifying the the source's code dependencies and libraries . Docker has been abandoned by kubernetes but it stills one of the best container runtime, we used docker to deploy containers and to create images, because the images are standardized which means the images created by docker can be used by another container runtime.

**Dockerfile** is a file that contains image dependencies and libraries , docker uses this file to create an image .

## 5.4 VMware workstation :

We used VMware Workstation Pro version 16.1.0 to make some virtual machines , as we know containerization works with linux systems so wa had to use VMware workstation to create a virtual machine with a linux system , we could install a linux system directly on our hardware but we preferred to use VMware to avoid reinstalling linux too many times after crushes or bugs , so we have saved a copy of our virtual machine and when we faced some bugs we just had to copy back the virtual machine again without installing the whole system . The second reason for using VMware is to create multiple virtual machines to prove the utility of orchestrators like Kubernetes (K8S) and to guarantee the availability .

## 5.5 Kubernetes :

Kubernetes or K8s , is an open source orchestrator that permit running and control containers in different machines called nodes, in other words "Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications." [48] , kubernetes create a cluster that has one master or more and workers , the master will be charged to run containers and specify other rules like RBAC ( Role-Based Access Control ) , Network policy , Security Policy , Container Network Interface (CNI). A worker machine is where the master can deploy containers , in other words , the master runs the deployment of the container , but the container will be deployed in a worker machine. We can also specify more replicas for our container using kubernetes which means running multiple containers with the same image .

## 5.6 Antrea :

Antrea is a CNI ( Container Network Interface ) and can be defined as kubernetes network plugin open source developed by " VMware " based on " Open vSwitch " technology , he has too many features like securing kubernetes network traffic by encrypting node-to-node communication using IPSec packet encryption. , Antrea allows setting network policy and enforce kubernetes's network policy by assigning network traffic filtering rules to pods.[49, 50, 51, 52]

## 5.7 Nessus :

Nessus is a vulnerability scanner developed by Tenable , he has a lot of databases which are updated everyday , this tool has lot of types of scans , advanced , malware , network and more other advanced options , we used nessus because it is easy to install , free , compatible with windows and can provide some solutions to the vulnerabilities found.

## 5.8 CRI-O :

"CRI-O is an implementation of the Kubernetes CRI (Container Runtime Interface) to enable using OCI (Open Container Initiative) compatible runtimes. It is a lightweight alternative to using Docker as the runtime for kubernetes. It allows Kubernetes to use any OCI-compliant runtime as the container runtime for running pods. Today it supports runc and Kata Containers as the container runtimes but any OCI-conformant runtime can be plugged in principle. CRI-O supports OCI container images and can pull from any container registry. It is a lightweight alternative to using Docker, Moby or rkt as the runtime for Kubernetes." [53] After the kubernetes announcement to abandon Docker , we used cri-o in our kubernetes nodes instead of docker . Cri-o has been approved by the CNCF and he is the container runtime of REDHAT’s platform "OpenShift ". the figure 5.1 below illustrates the difference between cri-o , docker and containerd

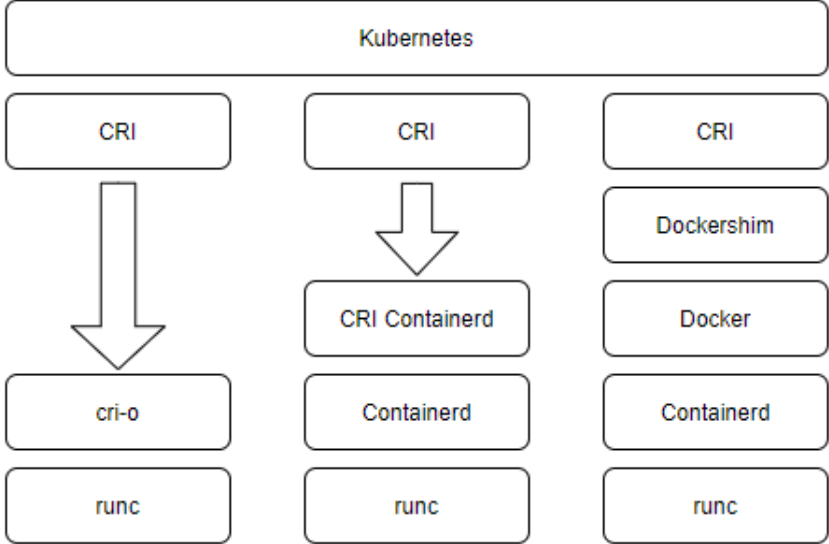


Figure 5.1: cri-o vs docker vs containerd [54]

## 5.9 HAproxy and Keepalived :

HAproxy is a load balancer , we have chosen this tool because it is famous, free ,easy to install and easy to manage , the load balancers are used to distribute the traffic between servers following a known algorithm like roundrobin to avoid server overload . Keepalived is a tool that allows creating virtual ip address using VRRP protocol , this address can be attributed to

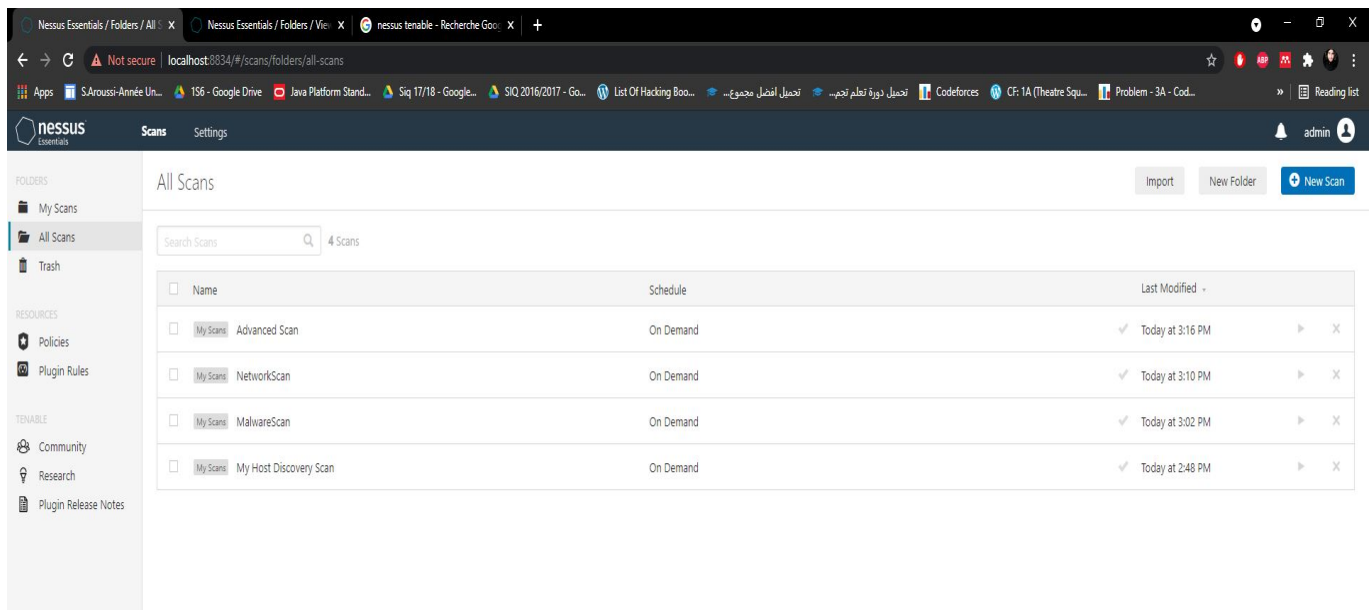


Figure 5.2: Nessus

different hosts , the master host will take the vip address if he goes down a backup host can take the vip address. These tools are used to guarantee high availability , which is very important for our containers' security .

## 5.10 Applying our solution :

### 5.10.1 Securing container from host :

To secure our container from the host , we have to secure our host , in other words we have to make sure that our host has no vulnerabilities and no weaknesses ,that's why we used nessus vulnerability scanner to detect host vulnerabilities , misconfiguration and weaknesses The figures (5.2 , 5.3 and 5.4) show some applied scans on two hosts , the Virtual ip host (Load Balancer), and the master host .

The result didn't have any important vulnerabilities , it just gave us some information about the systems . in the case of detecting a vulnerability , nessus can propose some solutions to apply but we have to do deep research of this vulnerability and fix it , to guarantee our containers' security.

### 5.10.2 Securing container from its running application :

We have used trivy container image scanner , which scans vulnerabilities and weaknesses in container images . we have wrote a dockerfile that is based on the latest ubuntu image as the figure 5.5 illustrates now from this docker file we build an image named docker-security2 as the figure 5.6 illustrates. as we can see the image has been built , now we will scan the image using trivy and check the scan result as the two figures (5.7 and 5.8 ) illustrate The scans shows 26 vulnerability of ubuntu and didn't detect the use of netcat which can be risky to the user , that's

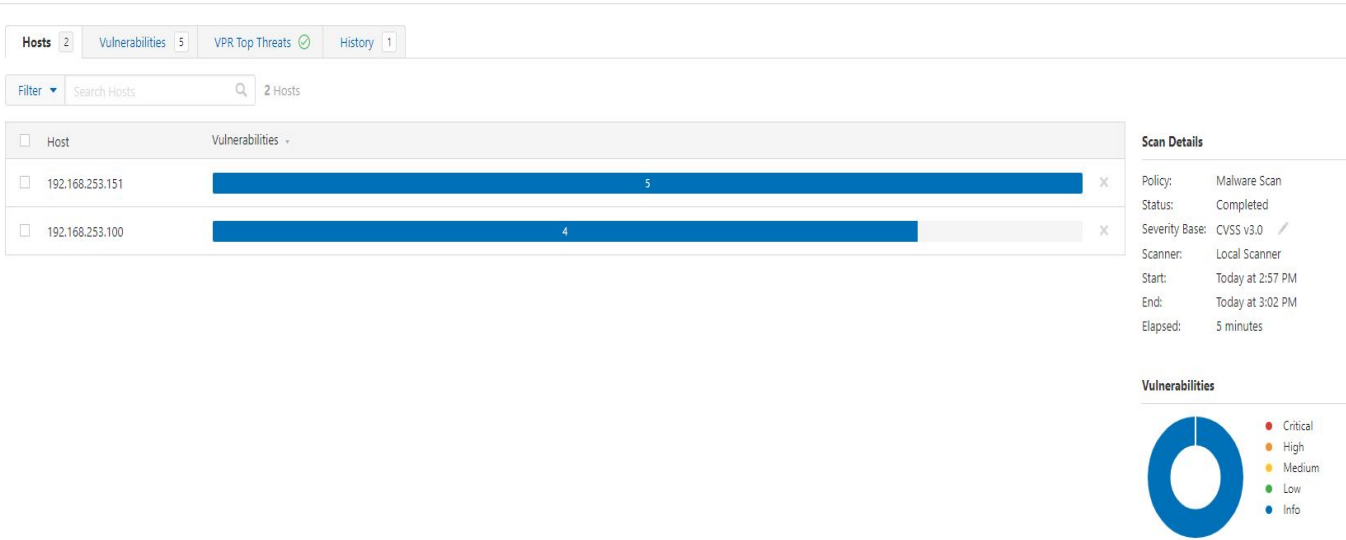


Figure 5.3: Scan Result 1

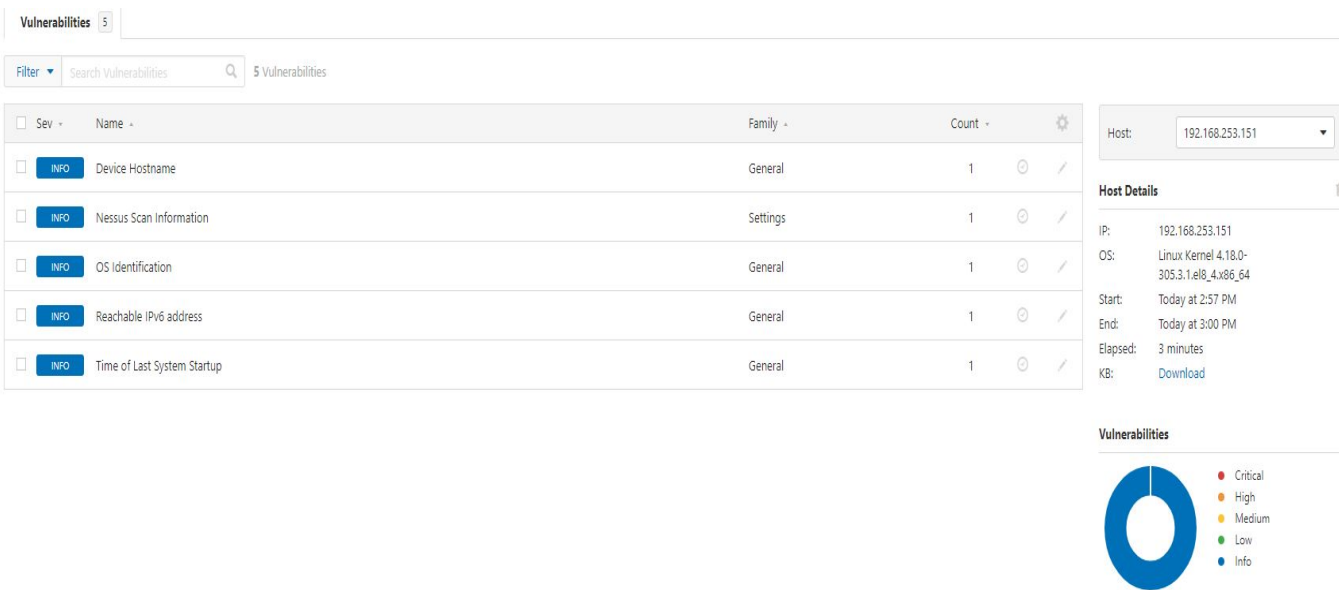


Figure 5.4: Scan result 2

why we have created a script that detects some risky applications to user like ssh and netcat or the use of sudo , our script will warn the user if this packages exists in the dockerfile . The figure 5.10 and 5.11 illustrate our script code and the table tools on our database where the figure 5.9 illustrates the result of scanning the image (5.6 ) using our script

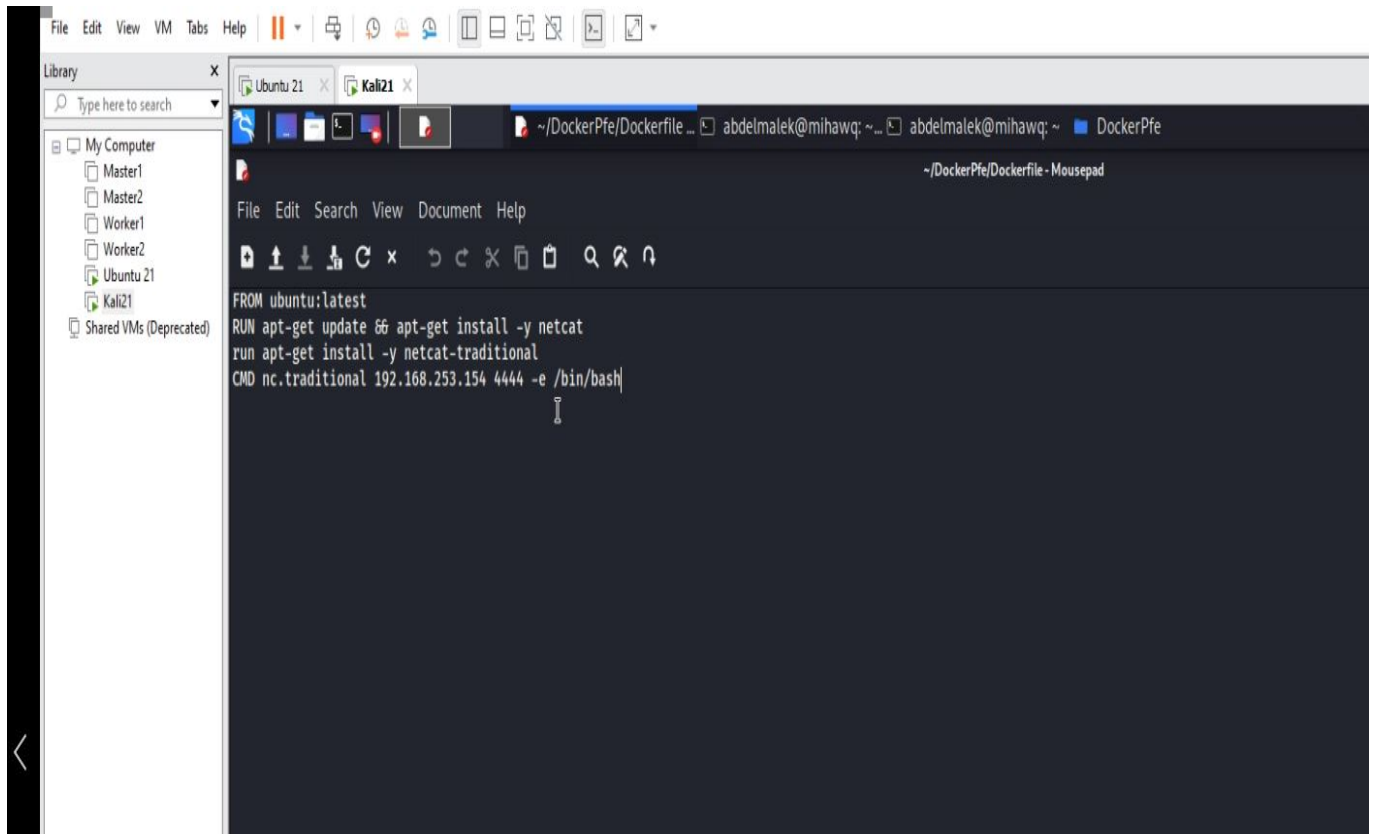


Figure 5.5: Dockerfile

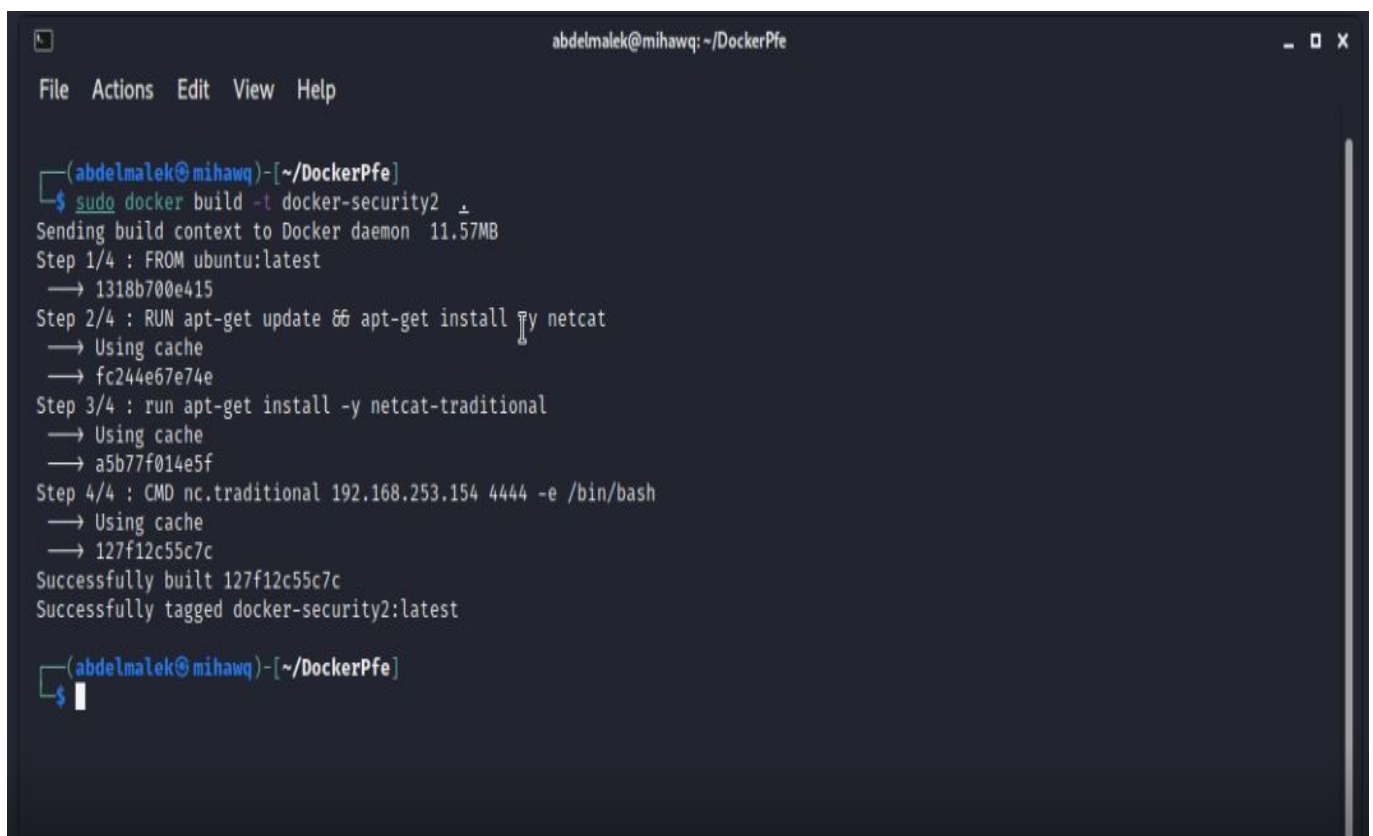


Figure 5.6: Build image from dockerfile



```

abdelmalek@mihawq: ~
File Actions Edit View Help

(abdelmalek@mihawq)-[~]
$ sudo trivy image docker-security2
[sudo] password for abdelmalek:
2021-09-22T23:59:21.271+0100 INFO Need to update DB
2021-09-22T23:59:21.271+0100 INFO Downloading DB ...
24.04 MiB / 24.04 MiB [-----] 100.00% 2.00 MiB p/s 12s
2021-09-22T23:59:34.472+0100 INFO Detected OS: ubuntu
2021-09-22T23:59:34.472+0100 INFO Detecting Ubuntu vulnerabilities ...
2021-09-22T23:59:34.473+0100 INFO Number of PL dependency files: 0

docker-security2 (ubuntu 20.04)
Total: 26 (UNKNOWN: 0, LOW: 25, MEDIUM: 1, HIGH: 0, CRITICAL: 0)

+-----+-----+-----+-----+-----+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION | TITLE |
+-----+-----+-----+-----+-----+
| bash | CVE-2019-18276 | LOW | 5.0-6ubuntu1.1 | | bash: when effective UID is not |
| | | | | | equal to its real UID the ... |
| | | | | | →avd.aquasec.com/nvd/cve-2019-18276 |
+-----+-----+-----+-----+-----+
| coreutils | CVE-2016-2781 | | 8.30-3ubuntu2 | | coreutils: Non-privileged |
| | | | | | session can escape to the |
| | | | | | parent session in chroot |
| | | | | | →avd.aquasec.com/nvd/cve-2016-2781 |
+-----+-----+-----+-----+-----+
| libc-bin | CVE-2016-10228 | | 2.31-0ubuntu9.2 | | glibc: iconv program can hang |
| | | | | | when invoked with the -c option |
| | | | | | →avd.aquasec.com/nvd/cve-2016-10228 |
+-----+-----+-----+-----+-----+
| | CVE-2019-25013 | | | | glibc: buffer over-read in |
| | | | | | iconv when processing invalid |
| | | | | | multi-byte input sequences in... |
| | | | | | →avd.aquasec.com/nvd/cve-2019-25013 |

```

Figure 5.7: Trivy image scanning 1

libcrypt20	CVE-2021-40528	MEDIUM	1.8.5-5ubuntu1	1.8.5-5ubuntu1.1	libcrypt: ElGamal implementation allows plaintext recovery →avd.aquasec.com/nvd/cve-2021-40528
	CVE-2021-33560	LOW			libcrypt: mishandles ElGamal encryption because it lacks exponent blinding to address a ... →avd.aquasec.com/nvd/cve-2021-33560
libgnutls30	CVE-2021-20231		3.6.13-2ubuntu1.3	3.6.13-2ubuntu1.6	gnutls: Use after free in client key_share extension →avd.aquasec.com/nvd/cve-2021-20231
	CVE-2021-20232				gnutls: Use after free in client_send_params in lib/ext/pre_shared_key.c →avd.aquasec.com/nvd/cve-2021-20232
libpcre3	CVE-2017-11164		2:8.39-12build1		pcre: OP_KETRMATCH feature in the match function in pcre_exec.c →avd.aquasec.com/nvd/cve-2017-11164

Figure 5.8: Trivy image scanning 2

```
(abdelmalek@mihawq)-[~/PFE]
└─$ sudo python3 Main.py ../DockerPfe/Dockerfile
WARNING : netcat found in line 2 : RUN apt-get update && apt-get install -y netcat

WARNING : netcat found in line 3 : run apt-get install -y netcat-traditional

WARNING : nc found in line 4 : CMD nc.traditional 192.168.253.154 4444 -e /bin/bash

(abdelmalek@mihawq)-[~/PFE]
└─$
```

Figure 5.9: Scan dockerfile using our script

```

1 import sys
2 import warnings
3 import mysql.connector
4
5 class bcolors:
6     OKGREEN = '\033[92m'
7     WARNING = '\033[93m'
8     BOLD = '\033[1m'
9     UNDERLINE = '\033[4m'
10    ENDC = '\033[0m' # end color
11
12 #liste=["netcat","nc","ssh","sudo","bash"]
13 mydb = mysql.connector.connect(
14     host="localhost",
15     user="root",
16     #password = our database password for exemple: password=dbpass
17     database="containers_sec"
18 )
19 db=mydb.cursor()
20 def check() :
21     file=str(sys.argv[1])
22     file=open(file,"r")
23     i=1
24     for line in file :
25         SQL="select t_name from tools"
26         db.execute(SQL)
27         result=db.fetchall()
28         for tool in result :
29             #print(str(tool[0]))
30             if str(tool[0]) in line :
31                 print(bcolors.WARNING + ' WARNING : '+ bcolors.ENDC, end = '')
32                 print(tool[0] + ' found in line ' +str(i)+ ' : ' +line )
33                 #warnings.warn( bcolors.WARNING + ' WARNING '+tool + ' found in line : '+line ))
34             i=i+1
35
36 check()

```

Figure 5.10: Our Script code

The screenshot shows a database query result for the 'tools' table. The query is 'select \* from tools'. The result grid displays the following data:

id	t_name	t_description
1	netcat	can be used to attack running container
2	nc	can be used to attack running container
3	sudo	can be used to run high privileges commands
4	bash	can be used to run a shell
5	ssh	can be used to get acces to our container
NULL	NULL	NULL

Figure 5.11: the table tools of our Database

### 5.10.3 securing containers from other containers

We decided to use Kubernetes and Antrea orchestrator to protect our containers , kubernetes will create pod that can contain multiple different containers (we can't run same containers in one pod ) kubernetes will give certificates and keys to the nodes and will set rbac and some network policies ,kubernetes will ensure confidentiality authentication and authorization , antrea can enforce kubernetes policies by adding another rules , in addition antrea will crypt our network traffic using IPsec . The figures ( 5.12 and 5.13) below illustrate the creation of the kubernetes cluster and generating the keys and applying rbac default config .

```
[abdelmalek@master1 ~]$ sudo kubeadm init --control-plane-endpoint="192.168.253.100:6443" --upload-certs --pod-network-cidr=10.20.0.0/16 --ignore-preflight-errors=all
[init] Using Kubernetes version: v1.22.2
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master1.localdomain] and IPs [10.96.0.1 192.168.253.151 192.168.253.100]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master1.localdomain] and IPs [192.168.253.151 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master1.localdomain] and IPs [192.168.253.151 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

Figure 5.12: kubernetes cluster init and the generation of certificates and keys

```
[mark-control-plane] Marking the node master1.localdomain as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: xuj7l9.xh7v9s3hw2kmkwa0
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!
```

Figure 5.13: Kubernetes init applying RBAC

at the end we get command and tokens that will proof identity and keys to join our cluster with other nodes as illustrates the figure 5.14 below



```

You can now join any number of the control-plane node running the following command on each as root:

kubeadm join 192.168.253.100:6443 --token xuj719.xh7v9s3hw2kmkwa0 \
--discovery-token-ca-cert-hash sha256:9d6bfed30e3e3a6c6bb131fd1fcc84181dbba202a526ele8a654609c182070e4 \
--control-plane --certificate-key 94435b92b040b820c4875695f28210039634674b87d6dc2d81ca9c9fb96347054

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.253.100:6443 --token xuj719.xh7v9s3hw2kmkwa0 \
--discovery-token-ca-cert-hash sha256:9d6bfed30e3e3a6c6bb131fd1fcc84181dbba202a526ele8a654609c182070e4
[abdelmalek@master1 ~]$

```

Figure 5.14: kubernetes cluster init result

after the initiation of cluster and before joining the nodes we have to deploy our CNI Antrea , antrea will create antrea agents for every node as the figure 5.15 below illustrates

```

[abdelmalek@master1 ~]$ kubectl apply -f https://raw.githubusercontent.com/antrea-io/antrea/main/build/yamls/antrea.yml
customresourcedefinition.apiextensions.k8s.io/antreaagentinfos.clusterinformation.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/antreaagentinfos.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/antreacontrollerinfos.clusterinformation.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/antreacontrollerinfos.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/clustergroups.core.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/clustergroups.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/clusternetnetworkpolicies.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/clusternetnetworkpolicies.security.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/egresses.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/externalentities.core.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/externalentities.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/externalippools.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.security.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/tiers.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/tiers.security.antrea.tanzu.vmware.com created
customresourcedefinition.apiextensions.k8s.io/traceflows.crd.antrea.io created
customresourcedefinition.apiextensions.k8s.io/traceflows.ops.antrea.tanzu.vmware.com created
serviceaccount/antctl created
serviceaccount/antrea-agent created
serviceaccount/antrea-controller created
clusterrole.rbac.authorization.k8s.io/aggregate-antrea-clustergroups-edit created
clusterrole.rbac.authorization.k8s.io/aggregate-antrea-clustergroups-view created
clusterrole.rbac.authorization.k8s.io/aggregate-antrea-policies-edit created
clusterrole.rbac.authorization.k8s.io/aggregate-antrea-policies-view created
clusterrole.rbac.authorization.k8s.io/aggregate-traceflows-edit created
clusterrole.rbac.authorization.k8s.io/aggregate-traceflows-view created
clusterrole.rbac.authorization.k8s.io/antctl created
clusterrole.rbac.authorization.k8s.io/antrea-agent created
clusterrole.rbac.authorization.k8s.io/antrea-cluster-identity-reader created
clusterrole.rbac.authorization.k8s.io/antrea-controller created
clusterrolebinding.rbac.authorization.k8s.io/antctl created
clusterrolebinding.rbac.authorization.k8s.io/antrea-agent created
clusterrolebinding.rbac.authorization.k8s.io/antrea-controller created
configmap/antrea-config-dhb74b822c created
service/antrea created
deployment.apps/antrea-controller created
apiservice.apiregistration.k8s.io/v1alpha1.stats.antrea.io created
apiservice.apiregistration.k8s.io/v1alpha1.stats.antrea.tanzu.vmware.com created
apiservice.apiregistration.k8s.io/v1beta1.system.antrea.io created
apiservice.apiregistration.k8s.io/v1beta1.system.antrea.tanzu.vmware.com created
apiservice.apiregistration.k8s.io/v1beta2.controlplane.antrea.io created
apiservice.apiregistration.k8s.io/v1beta2.controlplane.antrea.tanzu.vmware.com created
daemonset.apps/antrea-agent created
mutatingwebhookconfiguration.admissionregistration.k8s.io/crdmutator.antrea.io created
mutatingwebhookconfiguration.admissionregistration.k8s.io/crdmutator.antrea.tanzu.vmware.com created
validatingwebhookconfiguration.admissionregistration.k8s.io/crdvalidator.antrea.io created
validatingwebhookconfiguration.admissionregistration.k8s.io/crdvalidator.antrea.tanzu.vmware.com created
[abdelmalek@master1 ~]$

```

Figure 5.15: Antrea deployment

after the deployment of antrea we can see that the pods is in init status , we have to wait until they get the status running to get a successful installation , the two figures( 5.16 and 5.17 )below shows the result of antrea deployment

```

X Start page X abdelmalek@loadbalancer1:~ X abdelmalek@master1:~ X
Every 2.0s: kubectl get all --all-namespaces

NAMESPACE   NAME                                     READY   STATUS              RESTARTS   AGE
kube-system  pod/antrea-agent-qmxbm                 0/2     Init:0/1            0           17s
kube-system  pod/antrea-controller-75bf9745bd-j64xv 0/1     ContainerCreating  0           18s
kube-system  pod/coredns-78fcd69978-4q5v5          1/1     Running             0           3m34s
kube-system  pod/coredns-78fcd69978-dc7vg          1/1     Running             0           3m34s
kube-system  pod/etcd-master1.localdomain           1/1     Running             0           3m43s
kube-system  pod/kube-apiserver-master1.localdomain 1/1     Running             0           3m43s
kube-system  pod/kube-controller-manager-master1.localdomain 1/1     Running             0           3m49s
kube-system  pod/kube-proxy-2m8w4                  1/1     Running             0           3m35s
kube-system  pod/kube-scheduler-master1.localdomain 1/1     Running             0           3m43s

NAMESPACE   NAME                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
default     service/kubernetes  ClusterIP     10.96.0.1    <none>        443/TCP          3m50s
kube-system service/antrea      ClusterIP     10.98.60.224 <none>        443/TCP          18s
kube-system service/kube-dns    ClusterIP     10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP 3m48s

NAMESPACE   NAME                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-system  daemonset.apps/antrea-agent         1         1         0       1             0           kubernetes.io/os=linux 17s
kube-system  daemonset.apps/kube-proxy           1         1         1       1             1           kubernetes.io/os=linux 3m48s

NAMESPACE   NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
kube-system  deployment.apps/antrea-controller    0/1     1             0           18s
kube-system  deployment.apps/coredns              2/2     2             2           3m48s

NAMESPACE   NAME                                DESIRED   CURRENT   READY   AGE
kube-system  replicaset.apps/antrea-controller-75bf9745bd 1         1         0       18s
kube-system  replicaset.apps/coredns-78fcd69978          2         2         2       3m34s

```

Figure 5.16: Antrea deployment result 1

```

Every 2.0s: kubectl get all --all-namespaces

NAMESPACE      NAME                                                    READY   STATUS    RESTARTS   AGE
kube-system    pod/antrea-agent-qmxbm                                2/2     Running  0           5m26s
kube-system    pod/antrea-controller-75bf9745bd-j64xv                1/1     Running  0           5m27s
kube-system    pod/coredns-78fcd69978-4g5v5                          1/1     Running  0           8m43s
kube-system    pod/coredns-78fcd69978-dc7vg                          1/1     Running  0           8m43s
kube-system    pod/etcd-master1.localdomain                          1/1     Running  0           8m52s
kube-system    pod/kube-apiserver-master1.localdomain                 1/1     Running  0           8m52s
kube-system    pod/kube-controller-manager-master1.localdomain        1/1     Running  0           8m58s
kube-system    pod/kube-proxy-2m8w4                                   1/1     Running  0           8m44s
kube-system    pod/kube-scheduler-master1.localdomain                 1/1     Running  0           8m52s

NAMESPACE      NAME              TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
default        service/kubernetes ClusterIP      10.96.0.1     <none>         443/TCP          8m59s
kube-system    service/antrea    ClusterIP      10.98.60.224 <none>         443/TCP          5m27s
kube-system    service/kube-dns  ClusterIP      10.96.0.10    <none>         53/UDP,53/TCP,9153/TCP 8m57s

NAMESPACE      NAME                                                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR          AGE
kube-system    daemonset.apps/antrea-agent                            1         1         1         1             1           kubernetes.io/os=linux 5m26s
kube-system    daemonset.apps/kube-proxy                              1         1         1         1             1           kubernetes.io/os=linux 8m57s

NAMESPACE      NAME                                                    READY   UP-TO-DATE   AVAILABLE   AGE
kube-system    deployment.apps/antrea-controller                      1/1     1             1           5m27s
kube-system    deployment.apps/coredns                                2/2     2             2           8m57s

NAMESPACE      NAME                                                    DESIRED   CURRENT   READY   AGE
kube-system    replicaset.apps/antrea-controller-75bf9745bd           1         1         1       5m27s
kube-system    replicaset.apps/coredns-78fcd69978                    2         2         2       8m43s

```

Figure 5.17: Antrea deployment result 2

now we can join with other nodes , in our case we have one more master and two workers .to join the cluster we have to use the kubeadm join commands that we have got earlier from the initiation of our cluster , or we can create new join commands ,this commands has keys which provides authentication and authorization , the figures (5.18 and 5.19) below illustrate the result of joining the cluster , and the antrea final result

```

NAME                STATUS    ROLES    AGE   VERSION
master1.localdomain Ready    control-plane,master 28m   v1.22.1
master2.localdomain Ready    control-plane,master 13m   v1.22.1
worker1.localdomain Ready    <none>    15m   v1.22.1
worker2.localdomain Ready    <none>    14m   v1.22.1
[abdelmalek@master2 ~]$

```

Figure 5.18: Kubernetes nodes after joining the cluster



```

X Start page X abdelmalek@loadbalancer1:~ X abdelmalek@master1:~ X abdelmalek@master2:~ X abdelmalek@worker1:~ X abdelmalek@worker2:~ X
Every 2.0s: kubectl get all --all-namespaces

```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	pod/antrea-agent-bkrvz	2/2	Running	1 (7m27s ago)	13m
kube-system	pod/antrea-agent-gmxbm	2/2	Running	0	24m
kube-system	pod/antrea-agent-wbnbm	2/2	Running	0	12m
kube-system	pod/antrea-agent-xbrbt	2/2	Running	3 (8m1s ago)	14m
kube-system	pod/antrea-controller-75bf9745bd-j64xv	1/1	Running	0	24m
kube-system	pod/coredns-78fcd69978-4q5v5	1/1	Running	0	27m
kube-system	pod/coredns-78fcd69978-dc7vg	1/1	Running	0	27m
kube-system	pod/etcd-master1.localdomain	1/1	Running	1 (11m ago)	27m
kube-system	pod/etcd-master2.localdomain	1/1	Running	0	9m27s
kube-system	pod/kube-apiserver-master1.localdomain	1/1	Running	1 (11m ago)	27m
kube-system	pod/kube-apiserver-master2.localdomain	1/1	Running	0	9m42s
kube-system	pod/kube-controller-manager-master1.localdomain	1/1	Running	3 (5m40s ago)	27m
kube-system	pod/kube-controller-manager-master2.localdomain	1/1	Running	2 (7m44s ago)	9m19s
kube-system	pod/kube-proxy-2m8w4	1/1	Running	0	27m
kube-system	pod/kube-proxy-d5cc7	1/1	Running	0	12m
kube-system	pod/kube-proxy-xphds	1/1	Running	0	13m
kube-system	pod/kube-proxy-zcjh	1/1	Running	0	14m
kube-system	pod/kube-scheduler-master1.localdomain	1/1	Running	2 (8m30s ago)	27m
kube-system	pod/kube-scheduler-master2.localdomain	1/1	Running	2 (5m38s ago)	9m19s

Figure 5.19: Antrea status after the remain nodes join

#### 5.10.4 Securing host from containers :

containers can affect the host , the most important point , is to run rootless container , which means running container without root privileges ( by default the container has root privileges )

The figure 5.20 shows that containers by default operate as root , we created an image based on ubuntu then we deployed its container.

this can be done via creating new user non root , the figure 5.21 illustrates this process

We can also run containers in read only mode , which can help even with root containers as the figure 5.22 illustrates.

These are some steps to secure our host from our containers , some tools require high privileges so we have to pay attention to these kinds of containers and give high privileges only to the desired tools that require root . At the end we can limit the hardware resources for containers , like the memory and CPU usage like the figure 5.23 below illustrates.



```

root@7a2644375bf6:/
File Actions Edit View Help
(abdelmalek@mihawq)-[~/DockerPfe/rootless]
$ sudo docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
docker-sec          latest     127f12c55c7c  3 weeks ago  105MB
docker-security2    latest     127f12c55c7c  3 weeks ago  105MB
docker-security     latest     127f12c55c7c  3 weeks ago  105MB
docker-pfe          latest     cc39405b094d  3 weeks ago  105MB
<none>              <none>     fc00d8c668dc  3 weeks ago  105MB
<none>              <none>     589d31331c6e  3 weeks ago  105MB
ubuntu-testrootless latest     1318b700e415  8 weeks ago  72.8MB
ubuntu              20.04     1318b700e415  8 weeks ago  72.8MB
ubuntu              latest     1318b700e415  8 weeks ago  72.8MB
nginx               latest     08b152afcfae  2 months ago 133MB
hello-world         latest     d1165f221234  6 months ago 13.3kB

(abdelmalek@mihawq)-[~/DockerPfe/rootless]
$ cat Dockerfile
FROM ubuntu:latest

(abdelmalek@mihawq)-[~/DockerPfe/rootless]
$ sudo docker run --rm -it 1318b700e41 /bin/bash
root@7a2644375bf6:/# whoami
root
root@7a2644375bf6:/#

```

Figure 5.20: Container operates as root .

```

root@daa2c5870fd3:/
File Actions Edit View Help
(abdelmalek@mihawq)-[~/DockerPfe/limitmemory]
$ sudo docker run -it --memory="1g" --cpus="1.0" ubuntu
root@daa2c5870fd3:/#

abdelmalek@mihawq: ~
File Actions Edit View Help
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O    PIDS
daa2c5870fd3  fervent_proskuriakova  0.00%    1.453MiB / 1GiB     0.14%    946B / 0B     0B / 0B      1

```

Figure 5.23: Limiting memory and cpu for containers .

### 5.10.5 High availability :

in this part we included HAProxy and Keepalived to implement high available cluster . the two figures (5.24 and 5.25 ) below illustrate our high availability architecture

```

abdelmalek@mihawq: ~/DockerPfe/rootless
File Actions Edit View Help
└─$ cat Dockerfile
FROM ubuntu:latest
RUN groupadd -r nonrootuser && useradd -r -g nonrootuser nonrootuser
RUN chsh -s /usr/sbin/nologin root
ENV HOME /home/nonrootuser

(abdelmalek@mihawq)-[~/DockerPfe/rootless]
└─$ sudo docker build -t ubuntu-nonrootuser .
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM ubuntu:latest
  ──> 1318b700e415
Step 2/4 : RUN groupadd -r nonrootuser && useradd -r -g nonrootuser nonrootuser
  ──> Using cache
  ──> 5e4a0f1a3b4e
Step 3/4 : RUN chsh -s /usr/sbin/nologin root
  ──> Using cache
  ──> 196023eaffb8
Step 4/4 : ENV HOME /home/nonrootuser
  ──> Using cache
  ──> dc1de8ce522e
Successfully built dc1de8ce522e
Successfully tagged ubuntu-nonrootuser:latest

(abdelmalek@mihawq)-[~/DockerPfe/rootless]
└─$ sudo docker run -u nonrootuser --rm -it dc1de8ce522e /bin/bash
nonrootuser@006b2ebc1dba:/$ whoami
nonrootuser
nonrootuser@006b2ebc1dba:/$ su
Password:
su: Authentication failure
nonrootuser@006b2ebc1dba:/$

```

Figure 5.21: Container rootless user .

```

ubuntu          20.04      1318b700e415   8 weeks ago    72.8MB
ubuntu          latest     1318b700e415   8 weeks ago    72.8MB
nginx           latest     08b152afcfae  2 months ago   133MB
hello-world     latest     d1165f221234  6 months ago   13.3kB

```

```

(abdelmalek@mihawq)-[~/DockerPfe/rootless]
└─$ sudo docker run --read-only --rm -it 1318b700e415 /bin/bash
root@823061eedcce:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr
root@823061eedcce:/# cd tmp
root@823061eedcce:/tmp# ls
root@823061eedcce:/tmp# mkdir test2
mkdir: cannot create directory 'test2': Read-only file system
root@823061eedcce:/tmp# touch tst2
touch: cannot touch 'tst2': Read-only file system

```

Figure 5.22: Runs container in read only mode .

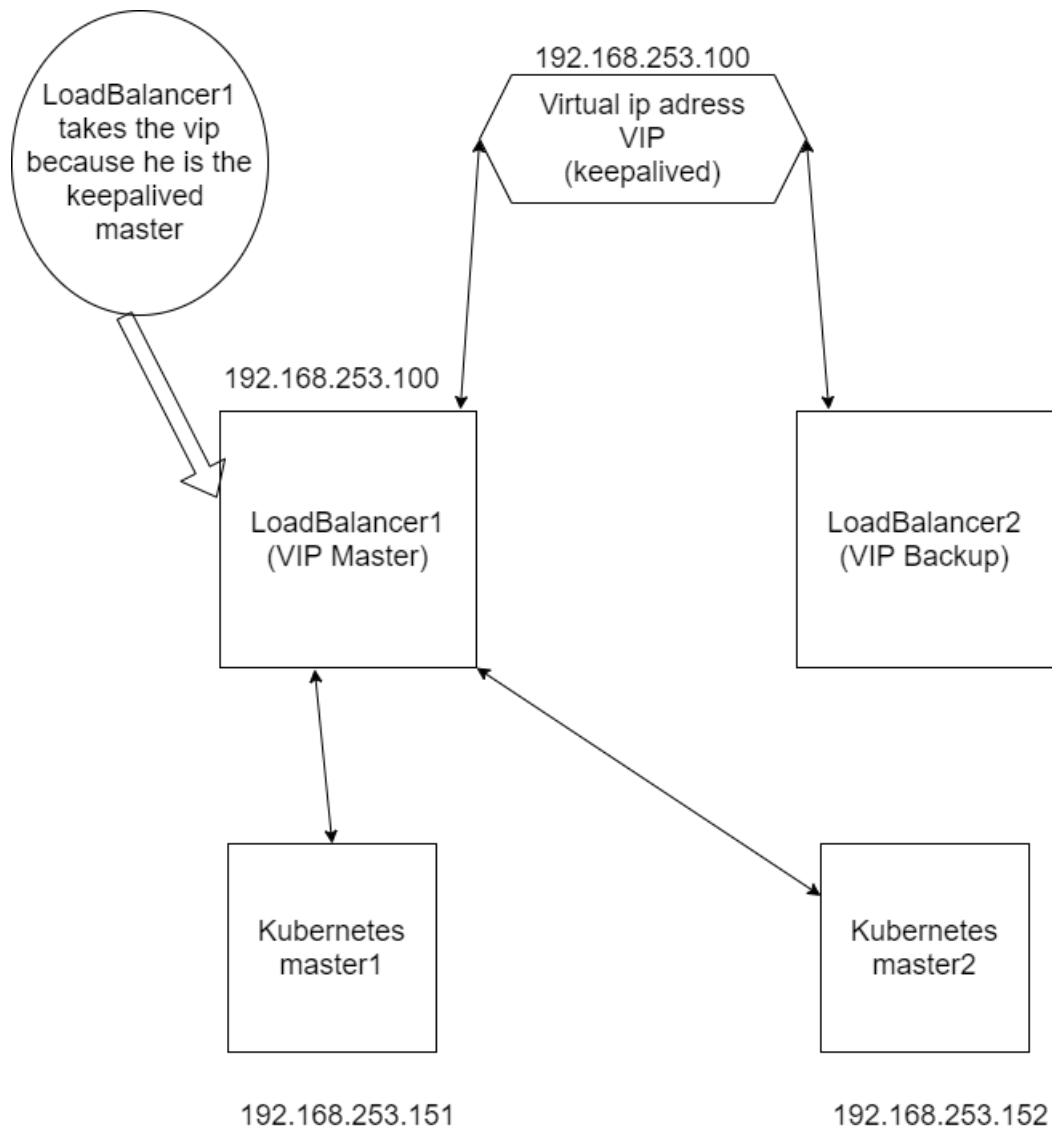


Figure 5.24: High Availability architecture

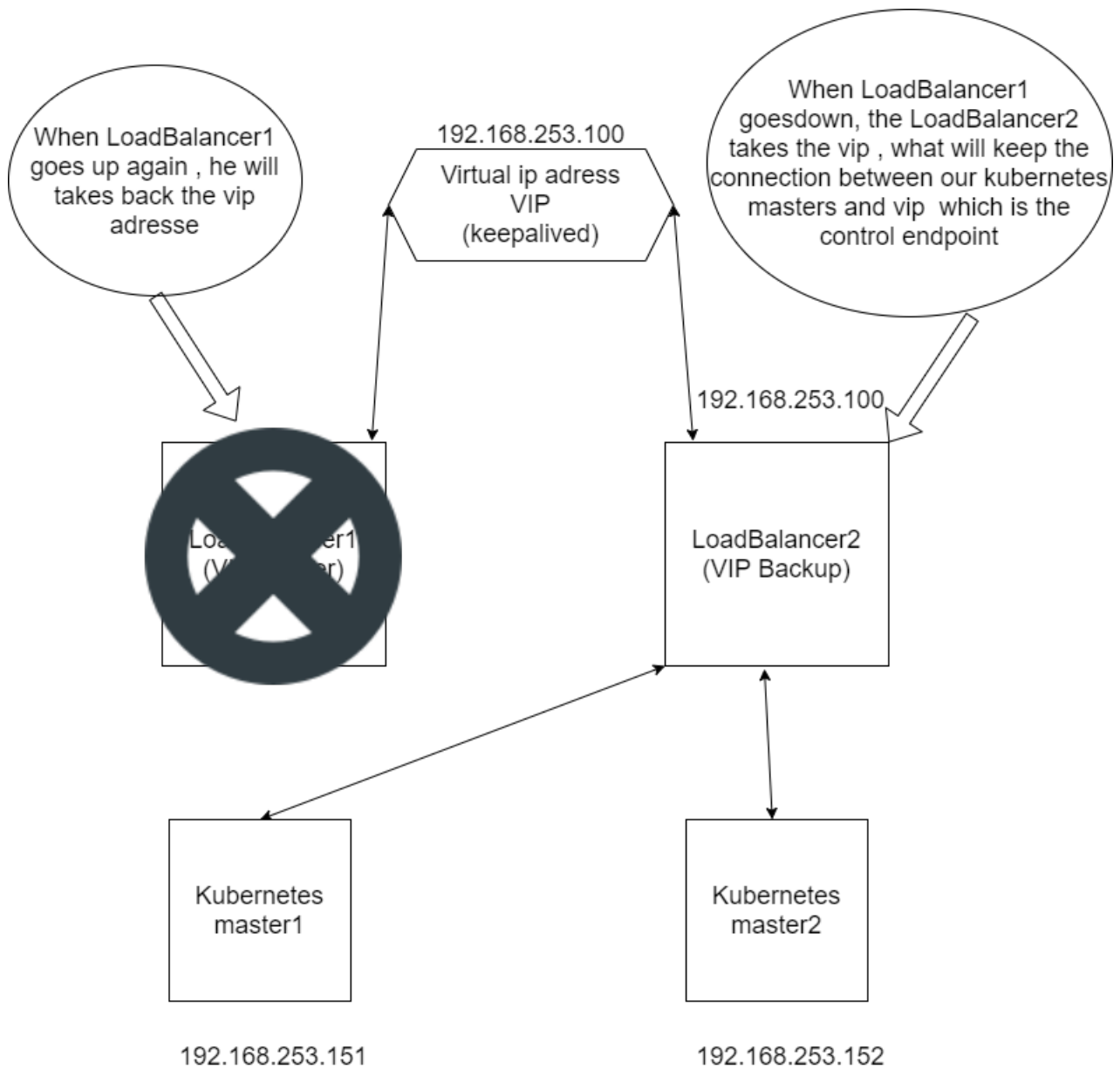


Figure 5.25: High Availability architecture

The figures (5.26 and 5.27 ) below illustrates the implementation of highAvailable cluster

```
[abdelmalek@loadbalancer1 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 00:0c:29:e1:4b:14 brd ff:ff:ff:ff:ff:ff
   inet 192.168.253.158/24 brd 192.168.253.255 scope global dynamic noprefixroute ens33
       valid_lft 1030sec preferred_lft 1030sec
   inet 192.168.253.100/2 scope global ens33
       valid_lft forever preferred_lft forever
   inet6 fe80::20c:29ff:fe1:4b14/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Figure 5.26: Keepalived master



```
[abdelmalek@loadbalancer2 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d6:58:b2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.253.157/24 brd 192.168.253.255 scope global dynamic noprefixroute ens33
        valid_lft 1087sec preferred_lft 1087sec
    inet6 fe80::20c:29ff:fed6:58b2/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Figure 5.27: Keepalived Backup

when the keepalived master goes down , the backup takes the vip until the master goes up again as illustrate the figures (5.28 5.29 ) below

```
[abdelmalek@loadbalancer1 ~]$ sudo systemctl stop keepalived
[sudo] password for abdelmalek:
[abdelmalek@loadbalancer1 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:e1:4b:l4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.253.158/24 brd 192.168.253.255 scope global dynamic noprefixroute ens33
        valid_lft 1642sec preferred_lft 1642sec
    inet6 fe80::20c:29ff:fe1:4b14/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[abdelmalek@loadbalancer1 ~]$
```

Figure 5.28: Keepalived master goes down

```
[abdelmalek@loadbalancer2 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d6:58:b2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.253.157/24 brd 192.168.253.255 scope global dynamic noprefixroute ens33
        valid_lft 1704sec preferred_lft 1704sec
    inet 192.168.253.100/32 scope global inet33
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed6:58b2/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[abdelmalek@loadbalancer2 ~]$
```

Figure 5.29: Keepalived Backup takes the vip

our cluster still respond as shows the figure 5.30 below

```
[abdelmalek@master1 ~]$ kubectl get all --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	pod/antrea-agent-xdknp	2/2	Running	0	2m13s
kube-system	pod/antrea-controller-75bf9745bd-5dh6z	1/1	Running	0	2m14s
kube-system	pod/coredns-78fcd69978-n8r6v	1/1	Running	0	10m
kube-system	pod/coredns-78fcd69978-tggrs	1/1	Running	0	10m
kube-system	pod/etcd-master1.localdomain	1/1	Running	5	10m
kube-system	pod/kube-apiserver-master1.localdomain	1/1	Running	11	10m
kube-system	pod/kube-controller-manager-master1.localdomain	1/1	Running	9	10m
kube-system	pod/kube-proxy-rk227	1/1	Running	0	10m
kube-system	pod/kube-scheduler-master1.localdomain	1/1	Running	9	10m

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
default	service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	10m
kube-system	service/antrea	ClusterIP	10.102.94.110	<none>	443/TCP	2m16s
kube-system	service/kube-dns	ClusterIP	10.96.0.10	<none>	53/UDP,53/TCP,9153/TCP	10m

NAMESPACE	NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR
kube-system	daemonset.apps/antrea-agent	1	1	1	1	1	kubernetes.io/os=linux
kube-system	daemonset.apps/kube-proxy	1	1	1	1	1	kubernetes.io/os=linux

NAMESPACE	NAME	DESIRED	CURRENT	READY	AGE
kube-system	deployment.apps/antrea-controller	1/1	1	1	2m16s
kube-system	deployment.apps/coredns	2/2	2	2	10m

NAMESPACE	NAME	DESIRED	CURRENT	READY	AGE
kube-system	replicaset.apps/antrea-controller-75bf9745bd	1	1	1	2m16s
kube-system	replicaset.apps/coredns-78fcd69978	2	2	2	10m

```
[abdelmalek@master1 ~]$
```

Figure 5.30: kubernetes Cluster respond after losing keepalived master

when the backup also goes down , our cluster will not respond until one of the loadbalancers goes up again as illustrates the figures (5.31 , 5.32 and 5.33) below

```
[abdelmalek@loadbalancer2 ~]$ sudo systemctl stop keepalived
[sudo] password for abdelmalek:
[abdelmalek@loadbalancer2 ~]$ ip a
```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 00:0c:29:d6:58:b2 brd ff:ff:ff:ff:ff:ff
inet 192.168.253.157/24 brd 192.168.253.255 scope global dynamic noprefixroute ens33
valid_lft 1666sec preferred_lft 1666sec
inet6 fe80::20c:29ff:fed6:58b2/64 scope link noprefixroute
valid_lft forever preferred_lft forever

```
[abdelmalek@loadbalancer2 ~]$
```

Figure 5.31: keepalived backup goes down

```
[abdelmalek@master1 ~]$ kubectl get all --all-namespaces
```

```
[abdelmalek@master1 ~]$
```

Figure 5.32: kubernetes Cluster not responding after losing the backup

```
[abdelmalek@master1 ~]$ kubectl get all --all-namespaces
NAMESPACE      NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)
default        service/kubernetes                  ClusterIP      10.96.0.1        <none>           443/TCP
kube-system    service/antrea                     ClusterIP      10.102.94.110   <none>           443/TCP
kube-system    service/kube-dns                   ClusterIP      10.96.0.10      <none>           53/UDP,53/TCP,9153/TCP

NAMESPACE      NAME                                DESIRED       CURRENT        READY          UP-TO-DATE      AVAILABLE      NODE
kube-system    daemonset.apps/antrea-agent        1             1              0              1                0              kubern
kube-system    daemonset.apps/kube-proxy          1             1              0              1                0              kubern

NAMESPACE      NAME                                READY         UP-TO-DATE      AVAILABLE      AGE
kube-system    deployment.apps/antrea-controller  0/1           1                0              3m21s
kube-system    deployment.apps/coredns            0/2           2                0              11m

NAMESPACE      NAME                                DESIRED       CURRENT        READY         AGE
kube-system    replicaset.apps/antrea-controller-75bf9745bd  1             1              0              3m21s
kube-system    replicaset.apps/coredns-78fcd69978           2             2              0              11m
```

Figure 5.33: kubernetes Cluster respond again after turning on our keepalived master

At the end we tested our solution in google and microsoft cloud platforms as the figure 5.34 and 5.35 illustrate

The screenshot shows the Google Cloud Platform console for a project named 'qwiklabs-gcp-04-04e98911ef5b'. The 'Kubernetes Engine' section is active, displaying a table of clusters. One cluster, 'private-cluster2', is shown with a green checkmark, indicating it is in a ready state. The cluster has 3 nodes, 6 vCPUs, and 12 GB of memory. Below the console, a terminal window shows the output of a command, listing internal IP addresses and DNS names for the cluster's nodes.

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	private-cluster2	us-central1-a	3	6	12 GB		-

```

- address: 10.0.4.3
  type: InternalIP
- address: gke-private-cluster2-default-pool-53c9ee73-2scx.us-central1-a.c.qwiklabs-gcp-04-04e98911ef5b.internal
  type: InternalDNS
--
addresses:
- address: 10.0.4.4
  type: InternalIP
- address: gke-private-cluster2-default-pool-53c9ee73-fxjp.us-central1-a.c.qwiklabs-gcp-04-04e98911ef5b.internal
  type: InternalDNS
--
addresses:
- address: 10.0.4.2
  type: InternalIP
- address: gke-private-cluster2-default-pool-53c9ee73-jsxd.us-central1-a.c.qwiklabs-gcp-04-04e98911ef5b.internal
  type: InternalDNS
student-02-e36eeb960a0e@source-instance:~$
```

Figure 5.34: Google cloud platform

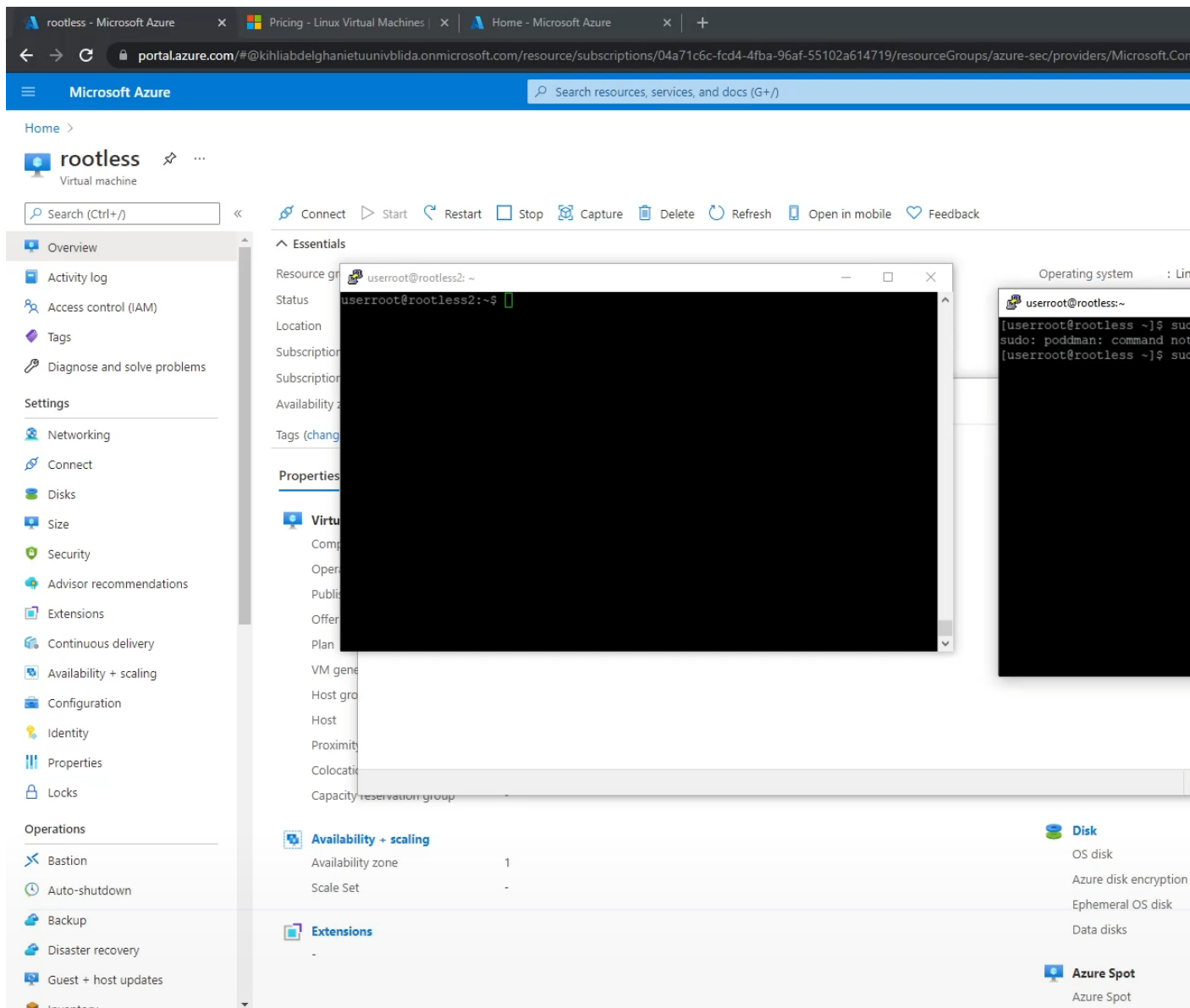


Figure 5.35: Microsoft cloud platform azure

## 5.11 Conclusion :

In this chapter we have presented a container security solution in a cloud environment .

First we have presented our hardware resources that were used in this work , the tools and applications that we have used to implement our solution, then we presented how to secure containers from different actors ,the host , running applications , other containers using kubernetes orchestrator , antrea and cri-o .

At the end we have seen how to secure the host from containers and how to implement high available kubernetes cluster .



---

# Conclusion and perspectives

---

The main goal of our work was securing containers in a cloud computing environment and providing a high available solution . We have started by studying virtualization techniques and made a comparison between VMs and containers than we moved to cloud computing where we defined the cloud , its types , the types of cloud services and its five essential characteristics , we have talked about the new development architecture Microservices and the serverless technologie than we moved to containers security in the cloud, where we have seen the security pillars, cloud security , some attacks and vulnerabilities that can be found in the cloud different layers and containers security basics .Finally we have studied some related works and analysed their solutions .

Our Solution was based on securing containers from the host, from container's running application where we have created a script that can helps securing our image ,securing containers from other containers and finally securing host from containers , we have included the last technologies and tools used in the field like cri-o ,kubernetes and antrea and at the end we have implemented high available solution where we have deployed two load balancers with two masters and two worker nodes .

This work has some perspectives ,like implementing service mesh solutions like istio , monitoring and visualization tools like grafana and prometheus however we were limited by the hardware resources and the internet interruptions.



---

# Bibliography

---

- [1] K. Chandrasekaran. *Essentials of Cloud Computing*. 2018. ISBN: 9781482205442.
- [2] Venkateshwarlu Velde and B. Rama. “Enhancement of Performance and Economy of Data Centers by Virtualization”. In: *International Journal of Simulation: Systems, Science and Technology* 19.6 (2018), pp. 9.1–9.8. ISSN: 1473804X. DOI: 10.5013/IJSSST.a.19.06.09.
- [3] Matthew Portnoy. *Virtualization Essentials Second Edition - SYBEX - 2016*. 2016, p. 336. ISBN: 978-1-119-26772-0.
- [4] VMware. “Security in Kubernetes | VMware | <https://github.com/kubernetes/kubernetes/blob/release-1.3/docs/design/security.md> | Last visit : 13/07/2021 ”. In: (), pp. 1–8. URL: <https://github.com/kubernetes/kubernetes/blob/release-1.3/docs/design/security.md>.
- [5] IBM. *What is virtualization?? | IBM | <https://www.ibm.com/cloud/learn/virtualization-a-complete-guide> | last visit 20/04/2021*. URL: <https://www.ibm.com/cloud/learn/virtualization-a-complete-guide>.
- [6] RedHat. *What is virtualization? | RedHat <https://www.redhat.com/en/topics/virtualization/what-is-virtualization> | last visit 20/04/2021*. URL: <https://www.redhat.com/en/topics/virtualization/what-is-virtualization>.
- [7] Miles Tracy, Wayne Jansen, and Mark McLarnon. “Guidelines on Securing Public Web Servers”. In: *NIST Special Publication 800* (2009), p. 44. URL: <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-44.pdf>.
- [8] Redhat. *What is a virtual machine (VM)? | RedHat | <https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine> | last visit:28/04/2021*. URL: <https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine>.
- [9] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. “Container security: Issues, challenges, and the road ahead”. In: *IEEE Access* 7 (2019), pp. 52976–52996. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2911732.

- [10] *Hypervisor TYPE 1 and Type 2* | *miromedium* | [https://miro.medium.com/max/2400/0\\*uOG3TpWM2BlBYkbg](https://miro.medium.com/max/2400/0*uOG3TpWM2BlBYkbg) (visited on 08/24/2021).  
URL: [https://miro.medium.com/max/2400/0\\*uOG3TpWM2BlBYkbg](https://miro.medium.com/max/2400/0*uOG3TpWM2BlBYkbg) (visited on 08/24/2021).
- [11] VMware. *What is a Hypervisor?* | *VMware* | <https://www.vmware.com/topics/glossary/content/hypervisor> | last visit 20/04/2021. URL: <https://www.vmware.com/topics/glossary/content/hypervisor>.
- [12] Fatma Bazargan, Chan Yeob Yeun, and Mohamed Jamal Zemerly. “State-of-the-Art of Virtualization, its Security Threats and Deployment Models”. In: *International Journal for Information Security Research* 3.3 (2013), pp. 335–343. DOI: 10.20533/ijisr.2042.4639.2013.0039.
- [13] *A Conceptual Study on Load Balancing, Operating System, Storage, and Server Process in Virtualization -A Case Study of Comparative of Cloud Computing* | *Researchgate* | [https://www.researchgate.net/publication/340601749\\_A\\_Conceptual\\_Study\\_on\\_Load\\_Balancing\\_Operating\\_System\\_Storage\\_and\\_Server\\_Process\\_in\\_Virtualization\\_-\\_A\\_Case\\_Study\\_of\\_Comparative\\_of\\_Cloud\\_Computing/figures?lo=1](https://www.researchgate.net/publication/340601749_A_Conceptual_Study_on_Load_Balancing_Operating_System_Storage_and_Server_Process_in_Virtualization_-_A_Case_Study_of_Comparative_of_Cloud_Computing/figures?lo=1) | Last visit : 2021-09-04. URL: [https://www.researchgate.net/publication/340601749\\_A\\_Conceptual\\_Study\\_on\\_Load\\_Balancing\\_Operating\\_System\\_Storage\\_and\\_Server\\_Process\\_in\\_Virtualization\\_-\\_A\\_Case\\_Study\\_of\\_Comparative\\_of\\_Cloud\\_Computing/figures?lo=1](https://www.researchgate.net/publication/340601749_A_Conceptual_Study_on_Load_Balancing_Operating_System_Storage_and_Server_Process_in_Virtualization_-_A_Case_Study_of_Comparative_of_Cloud_Computing/figures?lo=1) (visited on 09/04/2021).
- [14] *Virtual and physical networking - Software-Defined Networking (SDN) with OpenStack* | *OpenStack* | <https://subscription.packtpub.com/book/virtualization-and-cloud/9781786465993/1/ch011v11sec7/virtual-and-physical-networking> | Last visit : 2021-09-04. URL: <https://subscription.packtpub.com/book/virtualization-and-cloud/9781786465993/1/ch011v11sec7/virtual-and-physical-networking> (visited on 09/04/2021).
- [15] Docker. *What is a Container? - App Containerization* | *Docker* | <https://www.docker.com/resources/what-container> | Last visit : 2021-05-21. URL: <https://www.docker.com/resources/what-container> (visited on 05/21/2021).
- [16] IBM. *Containers* | *IBM* | <https://www.ibm.com/cloud/learn/containers> | last visit 20/04/2021. URL: <https://www.ibm.com/cloud/learn/containers>.
- [17] Boston Farnham et al. *Container Security Fundamental Technology Concepts that Protect Containerized Applications*. Tech. rep.
- [18] Docker. *Docker Security* | *Docker* | <https://docs.docker.com/engine/security/> | last visit 22/05/2021. URL: <https://docs.docker.com/engine/security/>.
- [19] *containers-vs-virtual-machines -image* | <https://images.contentstack.io/v3/assets/blt300387d93dabf50e/blt300387d93dabf50e/bltb6200bc085503718/5e1f209a63d1b650containers-vs-virtual-machines.jpg> | Last visit : 2021-08-10. URL: <https://images.contentstack.io/v3/assets/blt300387d93dabf50e/bltb6200bc085503718/5e1f209a63d1b650containers-vs-virtual-machines.jpg> (visited on 08/10/2021).

- [20] MIGUEL SORIANO. *Cloud Computing - Czech Technical University of Prague Faculty of electrical engineering* -. 2017. ISBN: 978-80-01-06212-8.
- [21] John R. Vacca. *Cloud Computing Security - Taylor and francis group*. Ed. by John R. Vacca. 2017. ISBN: 9781482260946. DOI: 10.1201/9781315372112.
- [22] *About Netflix - Completing the Netflix Cloud Migration | NetFlix | <https://about.netflix.com/en/news/completing-the-netflix-cloud-migration> | Last visit : 2021-08-24*. URL: <https://about.netflix.com/en/news/completing-the-netflix-cloud-migration> (visited on 08/24/2021).
- [23] *Gartner Forecasts Worldwide Public Cloud End-User Spending to Grow 23%* <https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021> | Last visit : 2021-08-24. URL: <https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021> (visited on 08/24/2021).
- [24] Amazon. *What is Cloud Computing ? | Amazon | <https://aws.amazon.com/what-is-cloud-computing/> | last visit 20/04/2021*. URL: <https://aws.amazon.com/what-is-cloud-computing/>.
- [25] NIST. *Final Version of NIST Cloud Computing Definition Published | NIST | <https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published> | Last visit : 2021-08-24*. URL: <https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published> (visited on 08/24/2021).
- [26] microsoft. *What Is Cloud Computing? A Beginner's Guide | Microsoft Azure | <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#cloud-deployment-types> | last visit 20/04/2021*. URL: <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#cloud-deployment-types>.
- [27] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology*. Tech. rep.
- [28] *Cloud Deployment models | ideesupp | <https://ideesupp.blogspot.com/2019/02/cloud-deployment-models.html> | Last visit : 2021-09-15*. URL: <https://ideesupp.blogspot.com/2019/02/cloud-deployment-models.html> (visited on 09/15/2021).
- [29] *Magic Quadrant for Cloud Infrastructure and Platform Services | Gartner | <https://www.gartner.com/doc/reprints?id=1-2710E4VR&ct=210802&st=sb> | Last visit : 2021-09-20*. URL: <https://www.gartner.com/doc/reprints?id=1-2710E4VR&ct=210802&st=sb> (visited on 09/20/2021).

- [30] *What is a private cloud?* | VMware | <https://www.vmware.com/topics/glossary/content/private-cloud> | Last visit : 2021-09-20. URL: <https://www.vmware.com/topics/glossary/content/private-cloud> (visited on 09/20/2021).
- [31] *what is openstack ?* | Openstack | <https://www.openstack.org/software> | Last visit : 2021-09-20. URL: <https://www.openstack.org/software> (visited on 09/20/2021).
- [32] *Les solutions SaaS, IaaS et PaaS : pour tout savoir !* | Hector | <https://hectorassetmanager.com/fr/blog/logiciel-saas-iaas-paas-solution/> | Last visit : 2021-09-05. URL: <https://hectorassetmanager.com/fr/blog/logiciel-saas-iaas-paas-solution/> (visited on 09/05/2021).
- [33] *What is Serverless Computing?* | IBM | <https://www.ibm.com/cloud/learn/serverless> | Last Visit: 28/04/2021. URL: <https://www.ibm.com/cloud/learn/serverless>.
- [34] *Monolith vs Microservices. Which one is the best to choose?* | Hengky Sanjaya | Medium | <https://medium.com/hengky-sanjaya-blog/monolith-vs-microservices-b3953650dfd> | Last visit : 2021-09-04. URL: <https://medium.com/hengky-sanjaya-blog/monolith-vs-microservices-b3953650dfd> (visited on 09/04/2021).
- [35] Jason Andress. *The Basics of Information Security Understanding the Fundamentals*. 2014, p. 217. ISBN: 9781597496537.
- [36] *cia triad* | f5 | [https://www.f5.com/content/dam/f5-labs-v2/article/articles/edu/20190709\\_what\\_is\\_the\\_cia\\_triad/cia\\_triad.png](https://www.f5.com/content/dam/f5-labs-v2/article/articles/edu/20190709_what_is_the_cia_triad/cia_triad.png) | Last visit : 04/09/2021. URL: [https://www.f5.com/content/dam/f5-labs-v2/article/articles/edu/20190709\\_what\\_is\\_the\\_cia\\_triad/cia\\_triad.png](https://www.f5.com/content/dam/f5-labs-v2/article/articles/edu/20190709_what_is_the_cia_triad/cia_triad.png) (visited on 08/10/2021).
- [37] *Shared responsibility in the cloud* | Microsoft | <https://docs.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility> | Last visit : 2021-09-20. URL: <https://docs.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility> (visited on 09/20/2021).
- [38] IBM. (1337) *What is Cloud Security?* - YouTube | IBM | <https://www.youtube.com/watch?v=jI8IKpjiCSM&t=2s> | Last visit : 2021-09-20. URL: <https://www.youtube.com/watch?v=jI8IKpjiCSM&t=2s> (visited on 09/20/2021).
- [39] Google. *How to secure your cloud environment* - YouTube | Google | <https://www.youtube.com/watch?v=MHTg2Au78LI&list=PLIivdWyY5sqLO-4ePY-A2yROgONOA6Cz4&index=1> | Last visit : 2021-09-20. URL: <https://www.youtube.com/watch?v=MHTg2Au78LI&list=PLIivdWyY5sqLO-4ePY-A2yROgONOA6Cz4&index=1> (visited on 09/20/2021).
- [40] *Two-Factor Authentication (2FA)* | UNCW | <https://uncw.edu/itsd/2fa/> | Last visit : 2021-09-20. URL: <https://uncw.edu/itsd/2fa/> (visited on 09/20/2021).
- [41] *What is RBAC Role-Based Access Control Types and benefits.* | Wallarm | <https://www.wallarm.com/what/what-exactly-is-role-based-access-control-rbac> | Last visit : 2021-09-20. URL: <https://www.wallarm.com/what/what-exactly-is-role-based-access-control-rbac> (visited on 09/20/2021).

- [42] BARTOSHA. *SQL Injection Attack & Protection From Attack* | BARTOSHA | <https://bartosha.com/sql-injection-attack-protection-from-attack/> | Last visit : 2021-09-20. URL: <https://bartosha.com/sql-injection-attack-protection-from-attack/> (visited on 09/20/2021).
- [43] *Cross-Site Scripting — Web-based Application Security* | Spanning | <https://spanning.com/blog/cross-site-scripting-web-based-application-security-part-3/> | Last visit : 2021-09-20.
- [44] *What is SYN Attack and How to Prevent the Attack?* | Indusface Blog | <https://www.indusface.com/blog/what-is-syn-synchronize-attack-how-the-attack-works-and-how-to-prevent-the-syn-attack> | Last visit : 2021-09-20.
- [45] Abhinav Kommula et al. “Machine Learning Techniques to Enhance Container Network Security”. In: (2021), pp. 622–627. DOI: 10.1109/csci51800.2020.00110.
- [46] Aparna Tomar et al. “Docker security: A threat model, attack taxonomy and real-time attack scenario of DoS”. In: *Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering* (2020), pp. 150–155. DOI: 10.1109/Confluence47617.2020.9058115.
- [47] Pankaj Mendki. “Securing Cloud Native Applications Using Blockchain”. In: (2021), pp. 419–423.
- [48] *Kubernetes officiel web site* | <https://kubernetes.io/> | last visit : 2021-08-21. URL: <https://kubernetes.io/> (visited on 08/21/2021).
- [49] *Securing and accelerating the Kubernetes CNI data plane with Project Antrea and NVIDIA Mellanox ConnectX SmartNICs* | Cloud Native Computing Foundation | <https://www.cncf.io/online-programs/securing-and-accelerating-the-kubernetes-cni-data-plane-with-project-antrea-and-nvidia-mellanox-connectx-smartnics/> | Last visit : 2021-08-23. URL: <https://www.cncf.io/online-programs/securing-and-accelerating-the-kubernetes-cni-data-plane-with-project-antrea-and-nvidia-mellanox-connectx-smartnics/> (visited on 08/23/2021).
- [50] *Container Networking with Antrea* | VMware | <https://www.vmware.com/products/antrea-container-networking.html> | Last visit : 2021-08-23. URL: <https://www.vmware.com/products/antrea-container-networking.html> (visited on 08/23/2021).
- [51] *Antrea - officiel web Site* | Antrea | <https://antrea.io/> | Last Visit : 2021-08-23. URL: <https://antrea.io/> (visited on 08/23/2021).
- [52] Cloud Native Computing Foundation. “Join us for KubeCon + CloudNativeCon Virtual Securing and Accelerating Kubernetes CNI Integrating Project Antrea and NVIDIA | Antrea | Cloud Native Computing Foundation”. In: (2020).
- [53] *What is a cri-o ?* | CRI-o | <https://cri-o.io/> | Last visit : 2021-09-20. URL: <https://cri-o.io/> (visited on 09/20/2021).

- [54] *Using CRI-O as container runtime for Kubernetes* | Medium | <https://medium.com/nerd-for-tech/using-cri-o-as-container-runtime-for-kubernetes-b8ddf8326d38> | | Last visit : 2021-09-20. URL: <https://medium.com/nerd-for-tech/using-cri-o-as-container-runtime-for-kubernetes-b8ddf8326d38> (visited on 09/20/2021).
- [55] Thanh Bui. “Analysis of Docker Security - Aalto University School of Science”. In: (). arXiv: arXiv:1501.02967v1.
- [57] Yunlong Guo et al. “Building trust in container environment”. In: *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019* (2019), pp. 1–9. DOI: 10.1109/TrustCom/BigDataSE.2019.00011.
- [58] Sara Shakeri, Lourens Veen, and Paola Grosso. “Evaluation of Container Overlays for Secure Data Sharing”. In: *Proceedings - 2020 IEEE 45th Local Computer Networks Symposium on Emerging Topics in Networking, LCN Symposium 2020* (2020), pp. 99–108. DOI: 10.1109/LCNSymposium50271.2020.9363266.
- [59] Holger Gantikow, Tom Zohner, and Christoph Reich. “Container anomaly detection using neural networks analyzing system calls”. In: *Proceedings - 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2020* (2020), pp. 408–412. DOI: 10.1109/PDP50117.2020.00069.
- [60] Gerald Budigiri et al. “Network Policies in Kubernetes : Performance Evaluation and Security Analysis”. In: (2021), pp. 407–412.
- [61] Nuno Mateus-Coelho, Manuela Cruz-Cunha, and Luis Gonzaga Ferreira. “Security in microservices architectures”. In: *Procedia Computer Science* 181.2019 (2021), pp. 1225–1236. ISSN: 18770509. DOI: 10.1016/j.procs.2021.01.320. URL: <https://doi.org/10.1016/j.procs.2021.01.320>.
- [62] Olivier Flauzac, Fabien Mauhourat, and Florent Nolot. “A review of native container security for running applications”. In: *Procedia Computer Science* 175.2019 (2020), pp. 157–164. ISSN: 18770509. DOI: 10.1016/j.procs.2020.07.025. URL: <https://doi.org/10.1016/j.procs.2020.07.025>.
- [63] Bunyamin Gunes, Gizem Kayisoglu, and Pelin Bolat. “Cyber security risk assessment for seaports: A case study of a container port”. In: *Computers and Security* 103 (2021), p. 102196. ISSN: 01674048. DOI: 10.1016/j.cose.2021.102196. URL: <https://doi.org/10.1016/j.cose.2021.102196>.
- [64] Igor Vurdelja et al. “A framework for automated dynamic malware analysis for Linux”. In: *2020 28th Telecommunications Forum, TELFOR 2020 - Proceedings* (2020), pp. 28–31. DOI: 10.1109/TELFOR51502.2020.9306520.



- [65] Jose Flora. “Improving the Security of Microservice Systems by Detecting and Tolerating Intrusions”. In: *Proceedings - 2020 IEEE 31st International Symposium on Software Reliability Engineering Workshops, ISSREW 2020* (2020), pp. 131–134. DOI: 10.1109/ISSREW51248.2020.00051.
- [66] Johan Scholliers et al. “Improving the Security of Containers in Port Related Supply Chains”. In: *Transportation Research Procedia* 14 (2016), pp. 1374–1383. ISSN: 23521465. DOI: 10.1016/j.trpro.2016.05.210. URL: <http://dx.doi.org/10.1016/j.trpro.2016.05.210>.
- [67] Red Hat. “A layered approach to container and Kubernetes security”. In: *Red Hat* (2020). URL: <https://www.redhat.com/en/resources/layered-approach-security-detail>.
- [68] Arsh Modak et al. “Techniques to Secure Data on Cloud: Docker Swarm or Kubernetes?”. In: *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018* Icicct (2018), pp. 7–12. DOI: 10.1109/ICICCT.2018.8473104.
- [69] Dibyendu Brinto Bose, Akond Rahman, and Shazibul Islam Shamim. “‘Under-reported’ Security Defects in Kubernetes Manifests”. In: (2021), pp. 10–13. DOI: 10.1109/EnCyCris52570.2021.00009. URL: <https://about.gitlab.com/>.
- [70] William Viktorsson, Cristian Klein, and Johan Tordsson. “Security-Performance Trade-offs of Kubernetes Container Runtimes”. In: *Proceedings - IEEE Computer Society’s Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS 2020-Novem* (2020), pp. 1–4. ISSN: 15267539. DOI: 10.1109/MASCOTS50786.2020.9285946.
- [71] P. P.W. Pathirathna et al. “Security testing as a service with docker containerization”. In: *International Conference on Software, Knowledge Information, Industrial Management and Applications, SKIMA 2017-Decem* (2018). ISSN: 25733214. DOI: 10.1109/SKIMA.2017.8294109.
- [72] Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, and Akond Rahman. “XI Commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices”. In: *Proceedings - 2020 IEEE Secure Development, SecDev 2020* (2020), pp. 58–64. DOI: 10.1109/SecDev45635.2020.00025. arXiv: 2006.15275.
- [73] Major Hayden. “Securing Linux Containers | Major | <https://major.io/2015/08/14/research-paper-securing-linux-containers/> | Last visit : 19-07-2021”. In: (2015), pp. 1–23. URL: <https://major.io/2015/08/14/research-paper-securing-linux-containers/>.

- [74] Brian Kelley et al. “Securing Cloud Containers Using Quantum Networking Channels”. In: *Proceedings - 2016 IEEE International Conference on Smart Cloud, SmartCloud 2016* (2016), pp. 103–111. DOI: 10.1109/SmartCloud.2016.58.
- [75] Massimiliano Mattetti et al. “Securing the infrastructure and the workloads of linux containers”. In: *2015 IEEE Conference on Communications and Network Security, CNS 2015 Spc 2015* (2015), pp. 559–567. DOI: 10.1109/CNS.2015.7346869.
- [76] Adam Miller and Lei Chen. “Securing Your Containers - An Exercise in Secure High Performance Virtual Containers”. In: (2012).
- [77] Murugiah Souppaya, John Morello, and Karen Scarfone. “NIST Special Publication 800-190”. In: ().
- [78] IBM. *What are Containers | IBM* | <https://www.ibm.com/cloud/learn/containers> | Last visit : 2021-05-22. URL: <https://www.ibm.com/cloud/learn/containers> (visited on 05/22/2021).
- [79] Redhat. *What is a virtual machine (VM)? | Redhat* | <https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine> | Last visit : 2021-04-28. URL: <https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine> (visited on 04/28/2021).
- [80] Shashank Mohan Jain. *Linux Containers and Virtualization*. Apress, 2020. DOI: 10.1007/978-1-4842-6283-2.
- [81] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. “Virtualization”. In: *Mastering Cloud Computing*. Elsevier, 2013, pp. 71–109. DOI: 10.1016/B978-0-12-411454-8.00003-6. URL: <https://linkinghub.elsevier.com/retrieve/pii/B9780124114548000036>.
- [82] Margaret Martonosi et al. *Synthesis Lectures on Computer Architecture Editor Hardware and Software Support for Virtualization Datacenter Design and Management: A Computer Architect’s Perspective A Primer on Compression in the Memory Hierarchy Analyzing Analytics Customizable Computing Die-stacking Architecture*. Tech. rep. 2015.
- [83] Xiangjiang Hu et al. “Hardware-virtualization-based software compatibility analysis method and its applications”. In: *PIC 2014 - Proceedings of 2014 IEEE International Conference on Progress in Informatics and Computing*. Institute of Electrical and Electronics Engineers Inc., 2014, pp. 442–445. ISBN: 9781479920334. DOI: 10.1109/PIC.2014.6972374.
- [84] Morty Eisen and Marcum Technology. *Introduction to Virtualization*. Tech. rep. 2011. URL: <http://en.wikipedia.org/wiki/Virtualization>.

- [85] Geeta and Shiva Prakash. “Role of virtualization techniques in cloud computing environment”. In: *Advances in Intelligent Systems and Computing*. Vol. 760. Springer Verlag, 2019, pp. 439–450. ISBN: 9789811303432. DOI: 10.1007/978-981-13-0344-9\_37.
- [86] Mario Gerla et al. *MCC’12 : proceedings of the 1st ACM Mobile Cloud Computing Workshop : August 17, 2012, Helsinki, Finland*, p. 66. ISBN: 9781450315197.
- [87] Moustafa Abdelbaky et al. “Docker Containers across Multiple Clouds and Data Centers”. In: *Proceedings - 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, UCC 2015*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 368–371. ISBN: 9780769556970. DOI: 10.1109/UCC.2015.58.
- [88] S. Anish Babu et al. “System performance evaluation of para virtualization, container virtualization, and full virtualization using Xen, OpenVZ, and XenServer”. In: *Proceedings - 2014 4th International Conference on Advances in Computing and Communications, ICACC 2014*. Institute of Electrical and Electronics Engineers Inc., 2014, pp. 247–250. ISBN: 9781479943647. DOI: 10.1109/ICACC.2014.66.
- [89] Margaret Martonosi et al. *Synthesis Lectures on Computer Architecture Editor Hardware and Software Support for Virtualization Datacenter Design and Management: A Computer Architect’s Perspective A Primer on Compression in the Memory Hierarchy Analyzing Analytics Customizable Computing Die-stacking Architecture*. Tech. rep. 2015.
- [90] Paul Barham et al. *Xen and the Art of Virtualization*. Tech. rep. 2003.
- [91] Margaret Martonosi et al. *Synthesis Lectures on Computer Architecture Editor Hardware and Software Support for Virtualization Datacenter Design and Management: A Computer Architect’s Perspective A Primer on Compression in the Memory Hierarchy Analyzing Analytics Customizable Computing Die-stacking Architecture*. Tech. rep. 2015.
- [92] *What Is AAA Security? | Fortinet | <https://www.fortinet.com/resources/cyberglossary/aaa-security> | Last visit : 2021-08-20*. URL: <https://www.fortinet.com/resources/cyberglossary/aaa-security> (visited on 08/20/2021).
- [93] *Build, Run, and Test a Container Image | KubeAcademy | <https://kube.academy/courses/containers-101/lessons/build-run-and-test-a-container-image> | Last visit : 2021-08-17*. URL: <https://kube.academy/courses/containers-101/lessons/build-run-and-test-a-container-image> (visited on 08/17/2021).
- [94] *Anatomy of a Container | KubeAcademy | <https://kube.academy/courses/containers-101/lessons/anatomy-of-a-container> | Last visit : 2021-08-17*. URL: <https://kube.academy/courses/containers-101/lessons/anatomy-of-a-container> (visited on 08/17/2021).

- [95] *Container Concepts* | KubeAcademy | <https://kube.academy/courses/containers-101/lessons/container-concepts> | Last visit : 2021-08-17. URL: <https://kube.academy/courses/containers-101/lessons/container-concepts> (visited on 08/17/2021).
- [96] *What is virtualization and how it works* | Cloud4Y | <https://www.cloud4y.ru/en/blog/how-virtualization-works/> | Last visit : 2021-08-24. URL: <https://www.cloud4y.ru/en/blog/how-virtualization-works/> (visited on 08/24/2021).
- [97] Hurwitz Judith, Kaufman Marcia, and Halper Fern. *Cloud Services for Dummies*. Ed. by IBM. 2014. ISBN: 9780874216561. arXiv: arXiv:1011.1669v3.
- [98] *Les avantages et les topologies du cloud computing* | Syloe | <https://www.syloe.com/les-avantages-et-les-topologies-du-cloud-computing/> | Last visit : 2021-09-15. URL: <https://www.syloe.com/les-avantages-et-les-topologies-du-cloud-computing/> (visited on 09/15/2021).