République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieure et de la Recherche Scientifique
Université Abderrahmane Mira, Bejaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme de master professionnel en informatique

Spécialité : Administration et sécurité des réseaux

Thème

Mise en place d'une architecture sécurisé d'un environnement conteneurisé en utilisant Docker et Kubernetes

Réalisé par :

Youbi Massinissa

Yousnadj Mohamed Karim

Soutenu devant le jury composé de :

Dr. Yaici Malika	MCB	Université de Bejaia	Président
Dr. Chekrid Mohamed	MCB	Université de Bejaia	Examinateur
Dr. Battat Nadia	MCB	Université de Bejaia	Encadrant

Année universitaire: 2023-2024

Remerciement

Nous tenons à exprimer notre gratitude envers **Dieu le Tout-Puissant** de nous avoir donné la santé et la volonté de commencer et de terminer ce travail.

Tout d'abord, nous tenons à remercier **nos chers parents** pour leurs encouragements et leur soutien inconditionnel.

Nous souhaitons exprimer notre plus profonde gratitude envers notre superviseure**, Mme BATTAT Nadia**, d'avoir accepté de superviser notre travail et de nous avoir encouragés tout au long de cette recherche. Nous la remercions également pour ses recommandations lors de la rédaction de ce mémoire.

Nous tenons également à exprimer notre reconnaissance envers tous **les enseignants** qui ont assuré notre formation durant ces cinq dernières années.

Enfin, nous adressons nos remerciements à l'ensemble des **membres du jury** qui ont fait l'honneur de juger notre travail.

Résumé

L'émergences des technologies de conteneurisation à complètement transformer la manière dont les applications sont développées et déployer, cette popularité a aussi attirait l'intention des acteurs malveillants, un autre problème pour les experts de la sécurité. Selon des études récentes, les menaces pesant sur les conteneurs et leurs orchestrateurs sont en constante augmentation, ce qui fait de la sécurisation de ces environnements un enjeu crucial.

L'objectif de ce travail est d'examiner les diverses menaces associées aux conteneurs et aux orchestrateurs, tout en explorant les stratégies pour sécuriser ces environnements et établir des pratiques de déploiement sûres. En plus de cela, nous présentons une solution spécifique, qui se focalise sur l'audit et la surveillance continue du moteur de conteneurisation Docker et de ses composants. Cette solution permet d'effectuer une analyse régulière des journaux afin de détecter les activités suspectes et malveillantes.

Les résultats obtenus confirment que la mise en œuvre de cette approche offre un niveau de sécurité optimal, permettant aux organisations d'exploiter pleinement les avantages offerts par les conteneurs dans leurs environnements de production.

Mots clé : Conteneur, Docker, Orchestrateur, Sécurité.

Abstract

The emergence of containerization technologies has completely transformed the way applications are developed and deployed. However, this popularity has also attracted the attention of malicious actors, posing a significant security challenge. Recent studies indicate a steady rise in threats targeting containers and their orchestrators, underscoring the critical need to secure these environments.

This study aims to examine various threats associated with containers and orchestrators, while exploring strategies to secure these environments and establish safe deployment practices. Additionally, we propose a specific solution focused on auditing and continuous monitoring of the Docker container engine and its components. This solution involves regular log analysis to detect suspicious and malicious activities.

The results obtained confirm that the implementation of this approach provides an optimal level of security, allowing organizations to fully leverage the benefits offered by containers in their production environments.

Keywords : Container, Docker, Orchestrator, Security.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS

API Application Programming Interface

BSD Berkeley Software Distribution

CA Certificate Authority

CD Continuous-Delivery

CI Continuous-Integration

CIS Center for Internet Security

CLI Command Lineinterface

CNCF Cloud Native Computing Foundation

CNI Container Network Interface

CRD Custom Resource Definition

CVE Common Vulnerabilities and Exposures

DevOps Development-Operations

DoS Denial-of-Service

eBPF extended Berkeley Packet Filter

ESXi Elastic Sky X Integrated

FPM FastCGI Process Manager

HTTP Hypertext Transfer Protocol

IBM International Business Machines

IP Internet Protocol

IPC Inter-Process Communication

IT Information technology

JSON JavaScript Object Notation

JVM Java Virtual Machine

LXC LinuX Containers

NAS Network Attached Storage

NAT Network address translation

OPA Open Policy Agent

OS Operating System

PHP FastCGI Process Manager

PV Persistant Volume

PVC Persistant Volume Claim

RAM Random Access Memory

REST Representational state transfer

RGPD Règlement Général sur la Protection des Données

SAN Storage Area Network

SQL Structured Query Language

SSD Solid State Drive

SSH Secure Shell

TLS Transport Layer Security

veth Virtual Ethernet devices

VLAN Virtual Local Area Network

VM Machine virtuelle

VMM Virtual Machines Monitor

YAML YAML Ain't Markup Language

INTRODUCTION GÉNÉRALE

Nous assistons ces dernières années à une augmentation considérable de l'utilisation des technologies de virtualisation. Selon une étude récente effectuée par WebinarCare en 2024, 77% des charges de travail des serveurs sont maintenant virtualisées [1]. Cela rend plus évidente l'exigence d'une solution de virtualisation performante. Pour répondre à ce besoin, deux approches populaires se sont imposées : la virtualisation par hyperviseur et la conteneurisation.

La virtualisation par hyperviseur, également connue sous le nom de virtualisation traditionnelle, fait appel à un hyperviseur pour générer de nouvelles machines virtuelles et les exécuter sur le serveur hôte. Chaque machine virtuelle fonctionne comme un environnement isolé avec son propre système d'exploitation.

Tandis que la virtualisation par conteneur repose sur une virtualisation au niveau du système d'exploitation, cette technologie est largement utilisée dans les environnements informatiques en raison de ses divers avantages, notamment en termes d'efficacité et de facilité de déploiement. Les conteneurs partagent le même noyau du système d'exploitation, ce qui leur permet de nécessiter moins de ressources que les machines virtuelles. Le "CNCF Annual Survey Report 2023" [2] montre que 84% des entreprises utilisant des conteneurs ont constaté une amélioration significative de l'efficacité de leurs opérations, et 78% ont noté une accélération du cycle de développement logiciel.

1. Problématique

Bien que la conteneurisation permette de créer des environnements virtuels légers et efficaces, elle présente néanmoins plusieurs menaces en matière de sécurité, comme toute autre technologie informatique. Notamment, les menaces liées aux images des conteneurs, les configurations défectueuses et diverses vulnérabilités ouvrent la porte à des acteurs malveillants. Ces derniers peuvent mener des attaques par déni de service (DoS) et exploiter des failles pour échapper aux conteneurs.

D'après le rapport « Cloud-Native Security and Usage » publié par Sysdig en 2023 [3] , sur plus de 3 millions de conteneurs analysés, 75% présentaient des vulnérabilités considérées comme «

élevées » ou « critiques ». De même, 85% des images en production présentaient au moins une vulnérabilité nécessitant une correction.

2. Contribution

Cette étude se focalise sur l'analyse des diverses menaces associées aux conteneurs et aux orchestrateurs, tout en soulignant les meilleures pratiques visant à renforcer la sécurité de ces environnements. En outre, nous présentons notre solution axée sur l'établissement de règles d'audit à l'aide d'un script écrit en Bash. Ce script permet de journaliser les activités du moteur de conteneurisation Docker [4], offrant une visibilité sur les activités suspectes ou malveillantes. Cela facilite la détection des vulnérabilités de sécurité, telles que les tentatives d'accès non autorisé aux images et aux volumes.

3. Organisation du mémoire

Ce mémoire se compose de quatre chapitres : Le premier chapitre établit les bases nécessaires pour les chapitres suivants. Nous commençons par explorer la virtualisation, puis nous étudions les conteneurs, leur histoire et leur structure, en les comparant aux machines virtuelles. Nous abordons également les architectures monolithiques et les microservices. Enfin, nous introduisons brièvement les principes de DevOps et de CI/CD (Intégration Continue/ Déploiement Continue) .

Dans le deuxième chapitre, nous explorons les deux techniques leaders dans le domaine de la conteneurisation et de l'orchestration : Docker et Kubernetes. Nous examinons leur architecture ainsi que leurs fonctionnalités et composants.

Dans le troisième chapitre, nous traiterons de la sécurité des conteneurs avec Docker et de l'orchestration avec Kubernetes, en explorant en détail les enjeux de sécurité associés. Ensuite, nous présenterons les bonnes pratiques à mettre en place pour renforcer la sécurité et garantir un environnement fiable.

Le quatrième chapitre se concentre sur la mise en place des différentes pratiques précédemment citées dans le chapitre trois et la mise en pratique de notre solution. Pour ce faire, nous avons créé un site web personnel (portfolio), et utilisé Docker v20.10.25 [5] pour la conteneurisation et Kubernetes 1.29 [6] pour le déploiement.

CHAPITRE I
CONCEPTS FONDAMENTAUX

I.1 Introduction

Afin d'avoir une meilleure compréhension de notre thème, Ce premier chapitre se focalise sur les bases nécessaires à la compréhension des concepts fondamentaux abordés dans ce mémoire. Nous commençons par explorer la virtualisation, une technologie révolutionnaire qui permet une gestion optimale des ressources informatiques. Ensuite, nous passons à la technologie de conteneurisation, une approche plus légère et plus efficace pour l'isolation des applications. Nous clarifions également les distinctions entre les architectures monolithiques et les architectures basées sur les microservices.

I.2 La virtualisation

I.2.1 Historique

Le concept de virtualisation a pris racine dans le domaine de l'informatique dès 1970 avec le développement de la machine VM/370 par IBM. Cette machine novatrice a introduit la notion de virtualisation à grande échelle, offrant la possibilité à plusieurs utilisateurs d'accéder simultanément à un même ordinateur en mode temps partagé.

Cependant, ce n'est que dans les années 1990 que la virtualisation a réellement pris son envol sur le marché, grâce aux efforts de la société VMware. Diane Greene et Mendel Rosenblum ont développé un hyperviseur logiciel révolutionnaire, permettant l'exécution de plusieurs systèmes d'exploitation sur une seule machine physique. Cette nouvelle technologie a profondément transformé le paysage des centres de données, offrant aux entreprises la possibilité de regrouper leurs serveurs, de réduire leurs coûts d'exploitation et d'optimiser leurs performance. [7]

I.2.2 Définition

La virtualisation consiste à créer une version virtuelle d'un dispositif ou d'une ressource, comme un serveur, un dispositif de stockage, un système d'exploitation ou une ressource réseau.

Cela permet aux organisations de partitionner un ordinateur ou serveur physiques en plusieurs machines virtuelles. Chaque machine virtuelle peut alors interagir de manière indépendante, et exécuter différents systèmes d'exploitation ou applications tout en partageant les ressources d'un seul ordinateur hôte. En créant plusieurs ressources à partir d'un seul ordinateur ou serveur, la virtualisation améliore l'extensibilité des charges de travail, tout en réduisant le nombre de serveurs utilisés, la consommation d'énergie, les coûts d'infrastructure et la maintenance. [8]

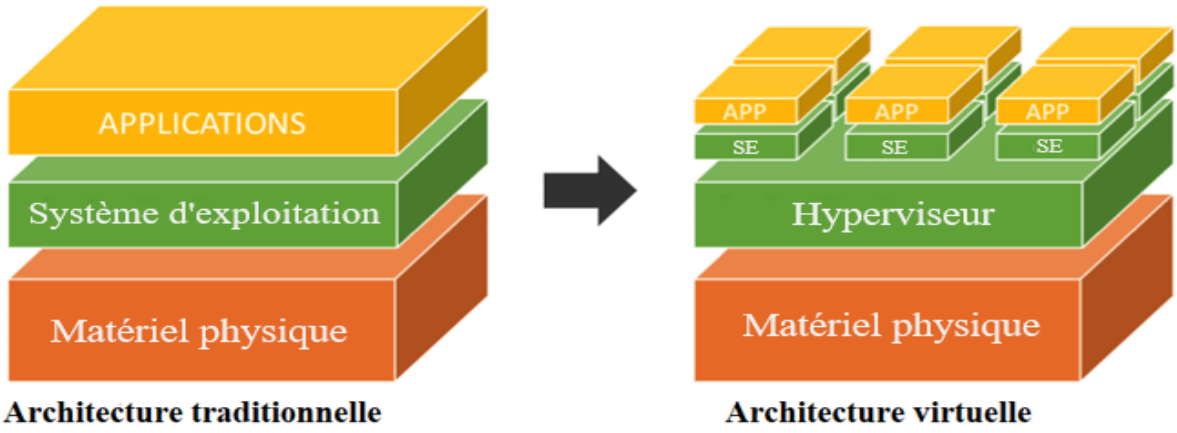

Figure I.1 : Architecture traditionnelle vs architecture virtuelle. [9]

I.2.3 Types de virtualisation

Dans le domaine de la virtualisation, il existe des catégories bien définies, chacune jouant un rôle essentiel dans l'optimisation des infrastructures informatiques :

1. **Virtualisation du serveur** : Cette catégorie permet à un serveur de fournir et de gérer plusieurs postes de travail individuels. En consolidant les ressources matérielles, la virtualisation du serveur améliore l'efficacité opérationnelle tout en réduisant les coûts liés à l'infrastructure.
2. **Virtualisation du réseau** : Conçue pour optimiser la bande passante et assurer une meilleure gestion des flux de données, la virtualisation réseau divise la bande passante en canaux indépendants. Cette approche permet une allocation plus efficace des ressources et une isolation sécurisée des différents flux de trafic.
3. **Virtualisation du stockage** : La virtualisation du stockage est la mise en commun de stockage physique de multiples périphériques de stockage réseau dans ce qui semble être un dispositif de stockage unique, qui est géré depuis une Console centrale.

La virtualisation du stockage est couramment utilisée dans un réseau de stockage NAS (Network Attached Storage) et SAN (Storage Area Network).

Dans ce qui suit, nous focalisons notre attention sur la virtualisation des serveurs, en examinant les deux techniques utilisées, à savoir la virtualisation applicative et la virtualisation système.

I.2.3.1 La virtualisation système

La technique de virtualisation du système fait référence à l'utilisation d'un hyperviseur pour permettre à la machine hôte d'exécuter plusieurs instances virtuelles en même temps. Ces dernières sont communément appelées des VMs (Virtual Machines), tandis que l'hyperviseur est connu sous le nom de VMM (Virtual Machines Monitor). [10]

a) **Machine virtuelle (VM) :** Une machine virtuelle s'exécute sur une machine physique. Elle opère avec son propre système d'exploitation (OS) et partage les mêmes ressources avec la machine physique qui l'héberge, tels que le processeur, la RAM, le disque dur et la carte réseau.

b) **Hyperviseur (VMM) :** Un hyperviseur**,** ou gestionnaire de machines virtuelles, est un logiciel de virtualisation qui crée et gère plusieurs machines virtuelles (VM) depuis une seule machine physique hôte.

La virtualisation peut être effectuée au moyen d'hyperviseurs de type 1 ou de type 2.

- Hyperviseur de type 1 (natif) : Il s'installe directement sur la couche physique du matériel du serveur. Ainsi, au démarrage de ce dernier, ce VMM prend le contrôle exclusif du matériel. Hyper-V, VMware,ESXi sont des hyperviseurs de type 1. [11]

- Hyperviseur de type 2 (hébergé) : Une approche similaire à celle d'un émulateur, où le logiciel d'hyperviseur est installé sur le système d'exploitation du serveur. Certains exemples d'hyperviseurs de type 2 incluent Oracle VM, Virtual Box et VMware Workstation. Ces hyperviseurs permettent l'exécution de machines virtuelles en tant qu'applications à l'intérieur du système d'exploitation hôte, offrant ainsi une solution de virtualisation plus flexible pour les environnements de développement, de test ou de bureau.

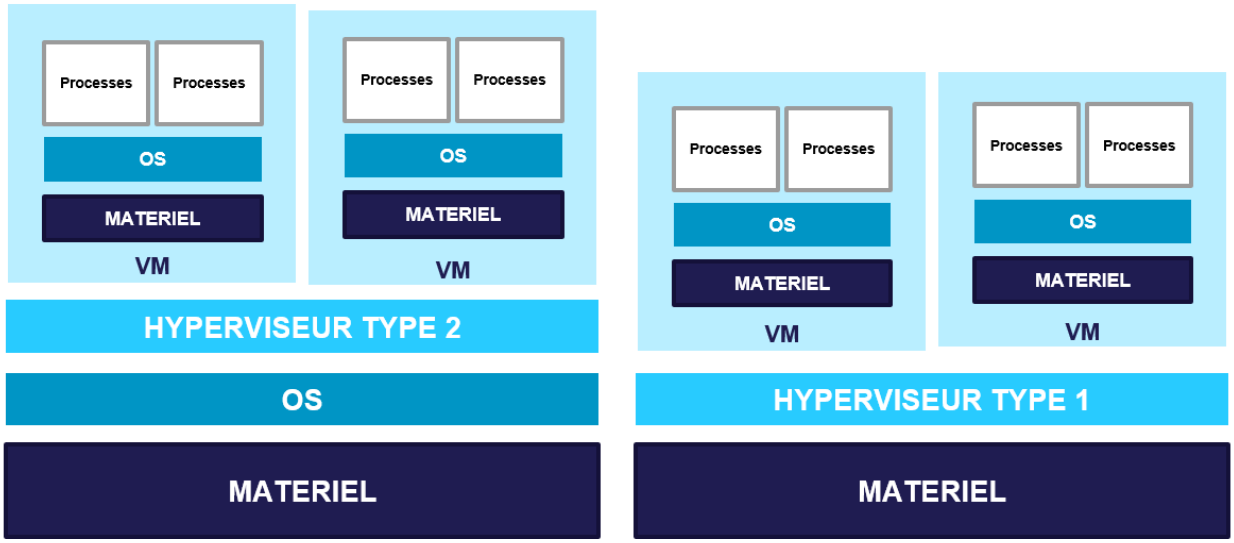

Figure I.2 : Types d'hyperviseur de la virtualisation serveur.

I.2.3.2 La virtualisation applicative

Contrairement à la virtualisation système, le principe de cette technique est de mettre en œuvre une couche de virtualisation sous forme d 'une application qui elle-même crée des instances virtuelles et les isole des spécificités de la machine hôte, c'est-à-dire de celles de son architecture ou de son système d'exploitation. Ceci permet au concepteur de l'application de ne pas avoir à rédiger plusieurs versions de son logiciel pour qu'il soit exécutable dans tous les environnements. Les machines virtuelles applicatives les plus connues sont la JVM (Java Virtual Machine) et les conteneurs.

a) **Java Virtual Machine (JVM) :** La JVM est une machine virtuelle qui permet l'exécution du code de java. La machine virtuelle joue le rôle d'interprète entre le langage de programmation Java et le matériel sous-jacent. Il fournit un environnement d'exécution pour que les applications Java fonctionnent sur différentes plateformes et systèmes d'exploitation. [12]

b) **Conteneurs :** Les conteneurs sont des environnements légers et isolés qui permettent l'exécution de processus et d'applications de manière indépendante.

I.3 La conteneurisation

La conteneurisation consiste à rassembler le code du logiciel et tous ses composants (bibliothèques, frameworks et autres dépendances) de manière à les isoler dans leur propre environnement d'exécution appelé conteneur. [13]

I.3.1 Les origines des conteneurs

Dans le contexte actuel de la transformation informatique, les conteneurs sont souvent considérés comme une technologie nouvelle révolutionnaire. Cependant, leur histoire s'étend bien au-delà de ce que l'on pourrait penser. Dès 1979, lorsque l'appel système chroot a été introduit dans Unix V7, les débuts de l'isolation des processus ont été réalisés. Les technologies de conteneurisation ont connu une évolution constante depuis cette innovation.

Au cours des années, des avancées significatives ont été réalisées. Les "FreeBSD Jails" ont émergé deux décennies plus tard pour partitionner les systèmes FreeBSD. En parallèle, Linux VServer a introduit des mécanismes similaires pour les systèmes Linux.

Par la suite, les "Solaris Containers" ont intégré des contrôles de ressources et une séparation des limites grâce aux zones, offrant ainsi des fonctionnalités avancées comme les instantanés et les clones en 2004.

En 2006, Google a introduit les Containers de Processus, qui ont été renommés plus tard Groupes de Contrôle (cgroups), pour limiter et isoler l'utilisation des ressources d'un groupe de processus. Cette innovation a été intégrée au noyau Linux.

LXC (LinuX Containers) est devenu en 2008 le premier gestionnaire de conteneurs Linux complet, utilisant les espaces de noms Linux et les cgroups pour la création et la gestion de conteneurs.

En 2013, Docker a introduit une nouvelle approche dans le domaine des conteneurs, rendant la création, la gestion et le déploiement des conteneurs beaucoup plus faciles.

Aujourd'hui, Kubernetes est largement reconnu comme la solution idéale pour l'orchestration et la gestion des conteneurs, avec une multitude d'entreprises proposant des solutions de gestion de conteneurs, de CI/CD et de DevSecOps. Cette évolution s'accompagne d'une croissance importante, mais également d'un intérêt croissant pour des enjeux tels que la sécurité. [14]

I.3.2 La structure des conteneurs

Les conteneurs sont des paquets logiciels légers et portables qui renferment le code d'une application ainsi que les fichiers de configuration nécessaires, les bibliothèques et les dépendances indispensables à son fonctionnement rapide et fiable, quel que soit l'environnement informatique. Ils s'exécutent en tant que processus isolés des autres processus, grâce à l'utilisation des espaces de noms du noyau et des cgroups. Les conteneurs peuvent être déployés sur des machines locales, des machines virtuelles ou même dans le cloud, et ils sont compatibles avec tous les systèmes d'exploitation. [15]

La figure I.3 montre les différentes couches nécessaires à la virtualisation des conteneurs.

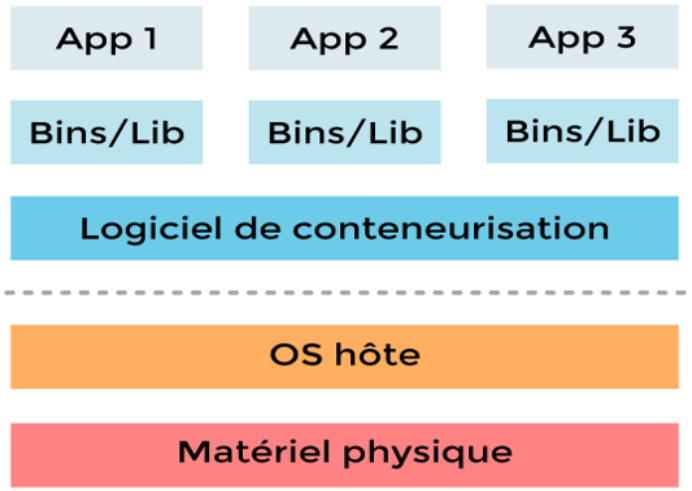

Figure I.3 : Structure d'un système de conteneurisation

Les avantages des conteneurs par rapport aux machines virtuelles expliquent leur adoption massive ces dernières années

- **Portabilité** : Les conteneurs permettent d'encapsuler entièrement tout ce dont une application a besoin pour fonctionner y compris les fichiers de configuration, les bibliothèques et les dépendances, Ce qui permet de les exécuter de manière cohérente sur différentes plateformes et environnements.
- **Scalabilité** : Des orchestrateurs comme Kubernetes permettent d'automatiser facilement le déploiement, la mise à l'échelle et la gestion des conteneurs.

- **Maintenance et gestion :** grâce aux orchestrateurs, la mise à jour des conteneurs est rapide et peut souvent se faire sans temps d'arrêt.
- **Coût** : En raison de leur utilisation plus efficace des ressources, les conteneurs peuvent réduire les coûts d'infrastructure.

Bien que les conteneurs présentent de nombreux avantages, il existe des situations où les machines virtuelles (VM) peuvent être plus adaptées. Le choix entre les conteneurs et les VM dépend des exigences de votre projet et de la configuration de votre infrastructure. Dans un scenario ou votre projet nécessite une isolation complète et une sécurité renforcée, les VM peuvent être plus appropriées, car elles offrent une séparation plus stricte entre les environnements.

I.3.3 Moteurs de conteneurisation

Le moteur de conteneur, ou environnement d'exécution de conteneur, est un programme logiciel qui crée des conteneurs en fonction des images de conteneurs. Il agit en tant qu'agent intermédiaire entre les conteneurs et le système d'exploitation, en fournissant et en gérant les ressources dont l'application a besoin. [16]

Parmi les moteurs de conteneur les plus populaires, on trouve Docker, Podman, et containerd, qui offrent des solutions de conteneurisation flexibles et adaptées à différents besoins et environnements d'exécution.

I.3.4 L'orchestration de conteneurs

L'orchestration de conteneurs est le processus automatique de gestion ou de planification du travail de conteneurs individuels pour des applications basées sur des microservices au sein de plusieurs clusters. Les plateformes d'orchestration de conteneurs largement déployées s'appuient sur des versions open source comme Kubernetes, Docker Swarm ou la version commerciale de Red Hat OpenShift. [17]

Les orchestrateurs de conteneurs possèdent un ensemble riche de fonctionnalités qui facilitent la gestion des applications conteneurisées. Parmi les principales fonctionnalités que on peut mentionnées :

- **Équilibrage de charge (Load Balancing) :** La répartition automatique du trafic réseau entrant entre plusieurs instances d'un service pour garantir la disponibilité de service.

- **Mise à l'échelle (Scaling) :** l'ajustement des instances et de la capacité des ressources allouées à une application ou à un service en réponse à la demande.
- **Déploiements automatisés et Rollbacks :** Le processus de déploiement de nouvelles versions d'applications et la capacité à revenir automatiquement à une version précédente de manière automatique et progressive, sans interruption de service.
- **Allocation automatique des ressources :** L'optimisation de l'utilisation des ressources disponibles et assurer un service de qualité tout en respectant les contraintes spécifiées.

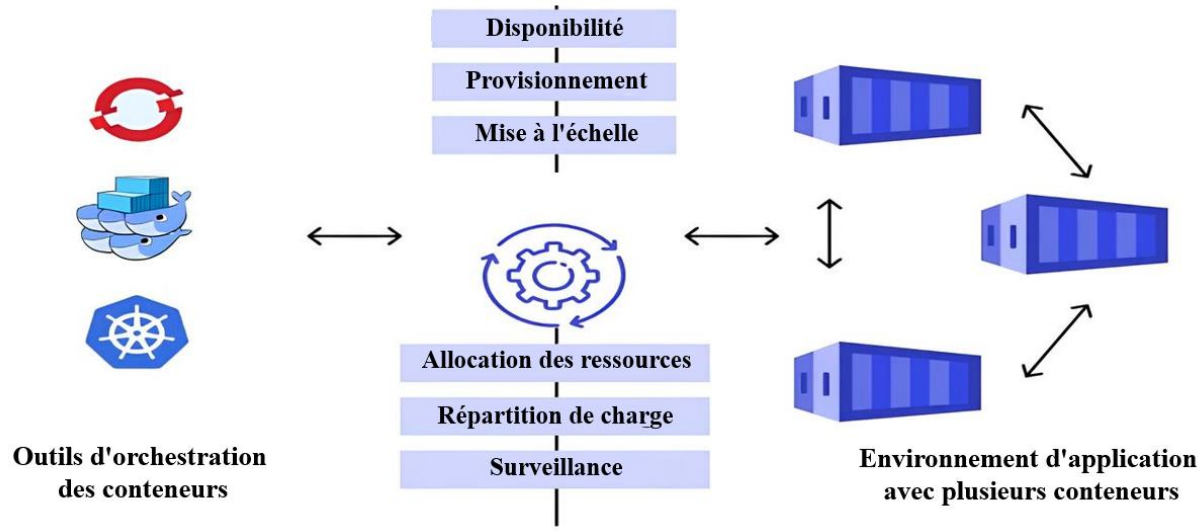

Figure I.4 : Fonctionnalités et Outils d'Orchestration de Conteneurs. *[18]*

I.4 Les machines virtuelles vs les conteneurs

Les machines virtuelles et les conteneurs présentent une grande similitude technologique. Au niveau inférieur, la virtualisation consiste à virtualiser les ressources physiques (RAM, processeur, Disque) Cela se fait à l'aide d'un hyperviseur qui gère ces ressources et permet aux machines virtuelles de fonctionner de manière autonome avec leur propre système d'exploitation, offrant ainsi une isolation complète.

D'un autre côté, la containerisation se trouve à un niveau plus élevé (niveau de l'application), Elle permet de virtualiser uniquement le système d'exploitation, en partageant le même noyau entre tous les conteneurs d'une machine. Chaque conteneur dispose de son propre espace utilisateur isolé, ce qui permet une isolation légèrement moins rigide par rapport aux machines virtuelles. Cependant, les conteneurs sont plus légers et plus rapides.

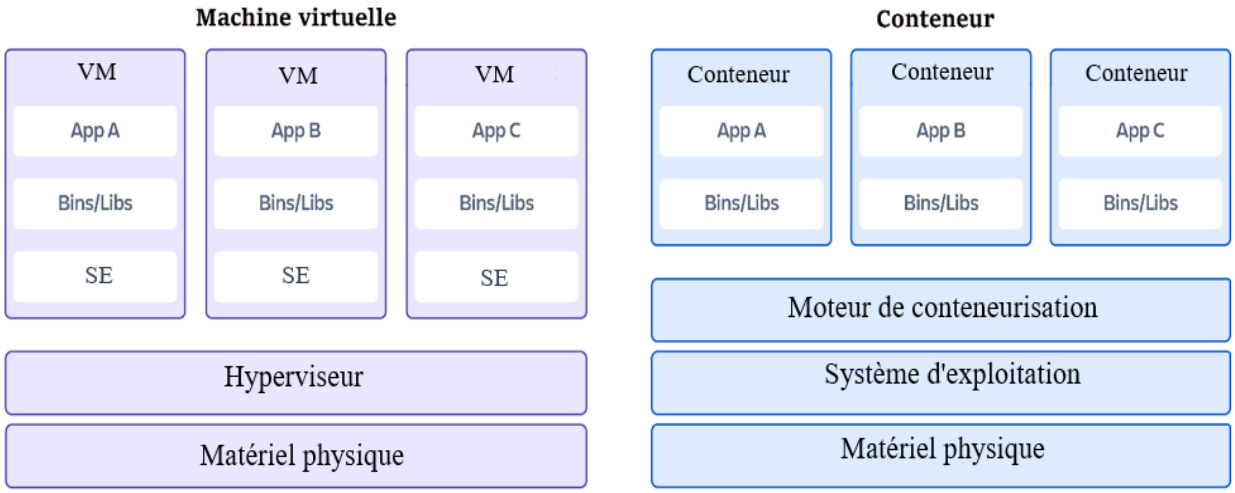

Figure I.5 : Les machines virtuelles vs les conteneurs. *[19]*

Le tableau ci-dessous résume les principales différences entre les machines virtuelles et les conteneurs :

Paramètres	Machines virtuelles	Conteneur
Temps de démarrage	Les VM nécessitent le démarrage d'un système d'exploitation complet, ce qui prend généralement plusieurs minutes.	Les conteneurs peuvent être créés et démarrés en quelques secondes
Consommation des ressources	Chaque VM dispose de ses propres ressources virtuelles, cela signifie que les ressources allouées à une VM sont exclusivement réservées même si elles ne sont pas pleinement utilisées (Forte utilisation des ressources et gaspillage).	Les conteneurs sont plus légers en termes de consommation des ressources. Ils peuvent être configurés pour utiliser exactement la quantité de ressources nécessaire à l'application qu'ils exécutent, sans gaspillage.
Sécurité	Les machines virtuelles offrent une isolation complète entre les différentes instances. Ce qui peut renforcer la sécurité en limitant le potentiel de mouvement latéral des attaquants.	Dépend de la configuration et de la mise en œuvre appropriées. Le partage de même noyau avec OS de l'hôte peut introduire un certain degré de risque en cas de vulnérabilités du noyau.
Communication	Via des commutateurs virtuels	Les mécanismes IPC (Inter-Process Communication) et les commutateurs virtuels

Tableau I-1: Comparaison entre les machines virtuelles et les conteneurs

I.5 L'architecture monolithique et l'architecture des microservices.

Une architecture monolithique est un modèle de développement logiciel traditionnel qui utilise une base de code unique pour exécuter plusieurs fonctions métier. Tous les composants logiciels d'un système monolithique sont interdépendants en raison des mécanismes d'échange de données au sein du système. La modification d'une architecture monolithique est contraignante et prend du temps, car de petites modifications ont un impact sur des pans entiers de la base de code. À l'inverse, les microservices sont une approche architecturale qui consiste à décomposer le logiciel en petits composants ou services indépendants. Chaque service joue un rôle unique et communique avec les autres services au moyen d'une interface bien définie. Comme ils s'exécutent indépendamment, vous pouvez mettre à jour, modifier, déployer ou mettre à l'échelle chaque service selon vos besoins. [20]

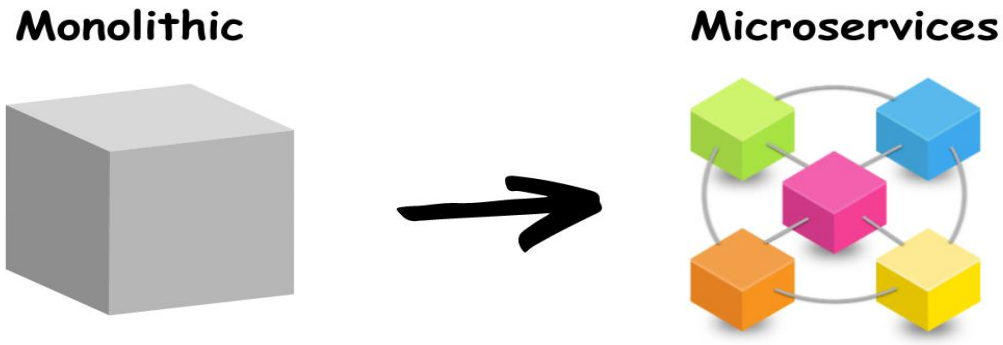

Figure I.6 : L'architecture monolithique et les microservices. *[21]*

I.6 DevOps

Le concept de DevOps repose sur la collaboration entre les équipes de développement (Dev) et les opérateurs (Ops). Cette collaboration et communication continue visent à automatiser le processus de développement et de déploiement de logiciel, ce qui augmente l'efficacité, la vitesse et la qualité du produit.

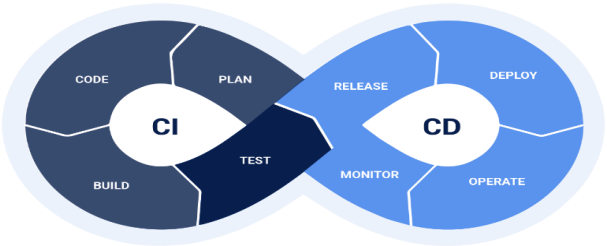

Figure I.7 : DevOps.

I.7 Conculsion

En conclusion, ce chapitre introductif a posé les bases essentielles pour aborder le sujet de la sécurité des environnements conteneurisés dans ce mémoire. Nous avons débuté par une exploration de la virtualisation dans son ensemble, ouvrant ainsi la voie vers la conteneurisation. Ensuite, nous avons étudié les conteneurs, leur histoire. La comparaison entre conteneurs et machines virtuelles a souligné les avantages des conteneurs en termes de taille, de mobilité et d'efficacité. Nous avons également examiné les architectures logicielles monolithiques et les microservices pour illustrer l'évolution vers des méthodes de développement plus flexibles. Enfin, une brève introduction aux principes de DevOps a mis en avant l'importance croissante de l'automatisation et de la collaboration dans les environnements de déploiement modernes.

Ce chapitre établit ainsi les bases pour approfondir notre compréhension des technologies de conteneurisation, en particulier Docker et Kubernetes, qui seront détaillées dans le prochain chapitre.

CHAPITRE II
INFRASTRUCTURE DE DÉPLOIEMENT MODERNE

II.1 Introduction

Dans le monde actuel du développement de logiciels, il est essentiel d'être efficace et flexible. Pour cela, l'adoption des pratiques et des outils modernes est recommandée. Une approche clé dans cette transformation est le DevOps, une culture d'entreprise qui encourage la collaboration entre les équipes de développement et d'exploitation afin de déployer rapidement des logiciels fiables et de qualité. Dans ce contexte, Docker et Kubernetes émergent comme des piliers essentiels de DevOps. Ces technologies révolutionnaires offrent une approche innovante pour la gestion des conteneurs et l'orchestration des applications.

II.2 Docker et Kubernetes dans la méthodologie DevOps

Le DevOps vise à automatiser les processus de développement, de test, de déploiement et de gestion des applications, afin d'améliorer l'efficacité, la qualité des logiciels, Docker et Kubernetes jouent des rôles fondamentaux. Docker permet aux développeurs d'encapsuler une application avec ses dépendances dans un conteneur, créant ainsi un environnement d'exécution autonome. En utilisant Kubernetes, ces applications peuvent être déployées de manière cohérente sur différentes plateformes, grâce à sa capacité à gérer efficacement les conteneurs. Cette automatisation du déploiement, permet aux équipes de livrer du code de manière répétitive et fiable, réduisant ainsi considérablement le délai de mise sur le marché et garantissant une haute disponibilité des applications.

II.3 Docker

Docker est une plateforme de conteneurisation open source qui fournit un système complet pour créer et exécuter des conteneurs, développé en 2013 par Solomon Hykes. À ses débuts, Docker était exclusivement compatible avec Linux, mais il s'est depuis étendu pour inclure d'autres systèmes d'exploitation populaires tels qu'Apple macOS et Microsoft Windows.

Aujourd’hui, Docker demeure le moteur de conteneurisation dominant, bénéficiant d’un vaste écosystème de projets, d’outils et de services qui le soutiennent.

Figure II.1 : Logo Docker. *[22]*

II.3.1 L'architecture de docker

Docker utilise une architecture client-serveur. Le client Docker communique avec le démon Docker, qui effectue le gros du travail de création, d'exécution et de distribution des conteneurs Docker. Le client Docker et le démon peuvent s'exécuter sur le même système. Ils communiquent à l'aide d'une API REST, via des sockets UNIX ou une interface réseau. [23] Ci-dessous le schéma simple d'une architecture Docker :

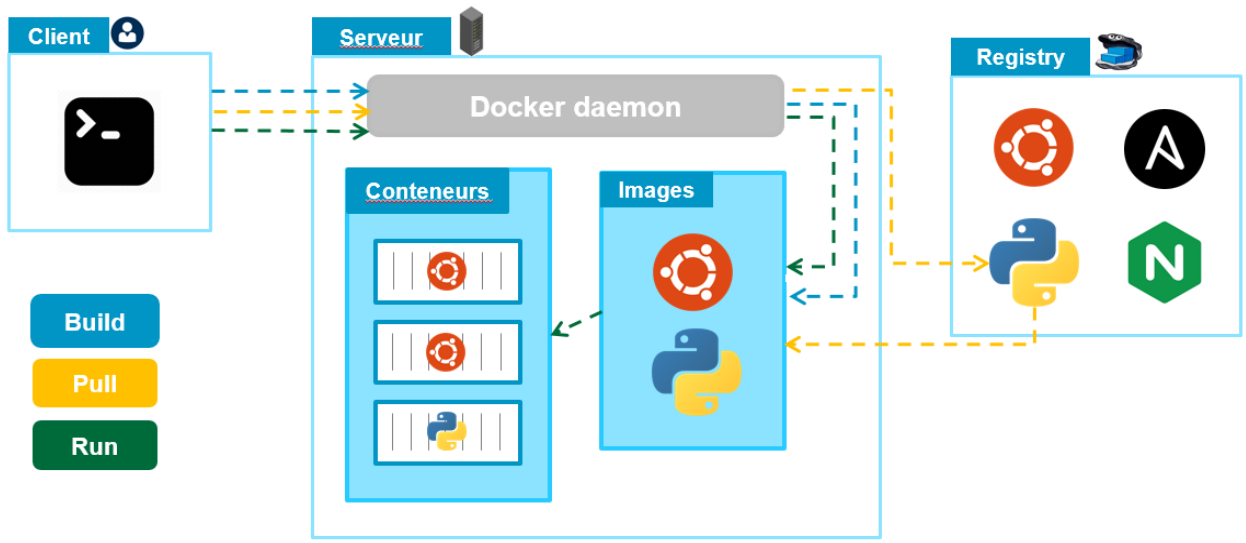

Figure II.2 : L'architecture simple de Docker.

- **Docker Host :** Un hôte Docker est un serveur physique ou virtuel (machine virtuelle) sur lequel vous exécutez Docker.
- **Docker démon (Dockerd) :** Le démon Docker est chargé de l'exécution des requêtes provenant du client. Son rôle principal est la création, l'exécution et la gestion d'images, de conteneurs et de réseaux.

- **Client Docker :** Le client Docker (CLI) est l'outil principal par lequel la plupart des utilisateurs Docker interagissent avec le système Docker.
- **API Docker :** L'API Docker est un service RESTful basé sur HTTP exposé par le démon Docker. Faire des requêtes à l'API permet d'appeler n'importe quelle action disponible pour gérer vos conteneurs, images et autres ressources Docker. Elle est accessible via un socket Unix ou une connexion TLS.
- **Docker Desktop :** Docker Desktop représente l'interface graphique offerte par Docker, permettant aux utilisateurs de visualiser aisément leurs images, leurs volumes ainsi que les conteneurs en cours d'exécution. Cette solution est compatible avec les systèmes d'exploitation Mac, Windows et Linux. En outre, elle propose des extensions tierces facultatives ainsi que des outils de sécurité et d'analyse supplémentaires, qui ne sont pas inclus dans une installation autonome du moteur.
- **Image :** Une image représente un logiciel complet, léger et autonome, capable d'exécuter des conteneurs de manière efficace. Elle encapsule tous les éléments nécessaires à l'exécution, tels que le code, les bibliothèques, les variables d'environnement et les outils système. Les images sont construites sur un modèle à couches, où chaque couche est empilée pour former un objet unique. Ce modèle en lecture seule est utilisé comme base pour créer des conteneurs Docker. Chaque conteneur possède sa propre couche inscriptible, où toutes les modifications sont enregistrées de manière isolée. [24]

  La figure II.3 présente le concept multicouche dans une images Docker.

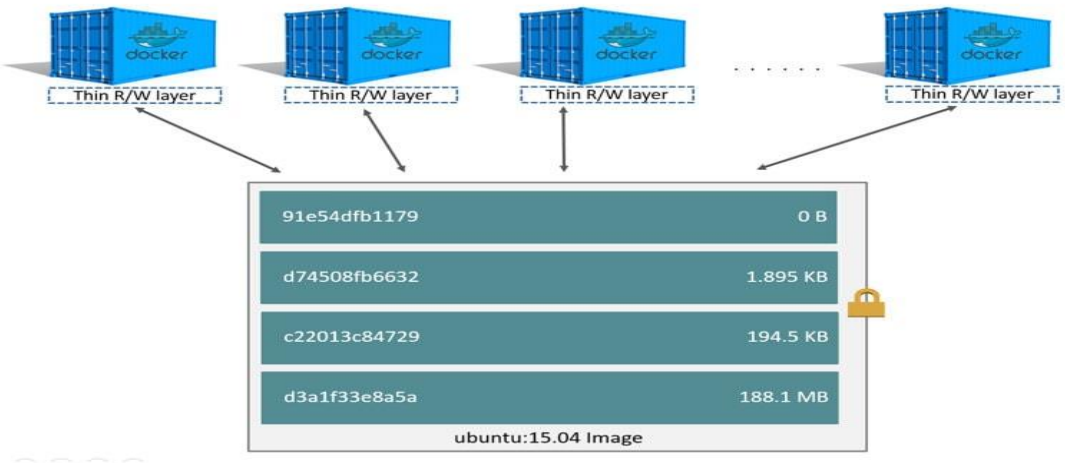

Figure II.3 : Notion de multicouches dans une image docker. *[25]*

- **Docker Hub :** Docker Hub est un service de registre public qui facilite la distribution le stockage, et la gestion des images. Cela permet aux utilisateurs de télécharger des images prêtes à l'emploi pour leurs propres projets, ainsi que de partager leurs propres images avec la communauté.
- **Docker Registry :** Un registre Docker est un système de stockage et de distribution d'images. La même image peut avoir plusieurs versions différentes, identifiées par leurs tags. Un registre Docker est organisé en Docker repositories, où un repositorie contient toutes les versions d'une image spécifique. Le registre permet aux utilisateurs de Docker d'extraire des images localement, ainsi que de transférer de nouvelles images vers le registre. **[26]**

II.3.2 Principaux Outils de Docker

Malgré la diversité des solutions de conteneurisation, Docker s'est impose comme le leader dans ce domaine. Cette domination n'est pas seulement à cause de la puissance de sa plateforme de conteneurisation, mais également à son ensemble d'outils puissants qui ont facilité la façon dont les conteneurs sont développés et exécutés.

II.3.2.1 Dockerfile

Dockerfile est un fichier texte qui permet d'automatiser le processus de configuration et de création d'image, c'est un ensemble d'instruction qui décrivent les étapes nécessaires pour configurer l'environnement à l'intérieur du conteneur, copier des fichiers, exposer des ports, exécuter des commandes.

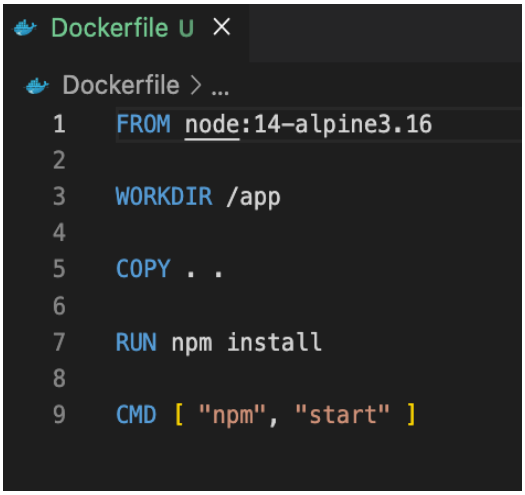

Figure II.4 : Exemple d'un Dockerfile

II.3.2.2 Docker compose

Docker Compose est un outil qui permet de définir et d'exécuter des applications Docker multi-conteneurs, grâce à des fichiers de configuration simples utilisant le langage YAML. Ensuit avec une commande simple, "docker-compose up", vous pouvez générer vos images et lancer vos conteneurs spécifier dans votre configuration.

II.3.2.3 Docker Swarm

Swarm est le premier gestionnaire de conteneurs Docker, lancé par la société Docker en 2014. Il est devenu une fonctionnalité native et intégré au démon Docker depuis la version1.12 en 2016. L'activation du mode Swarm peut pallier les lacunes de Docker en termes de déploiement, d'exploitation et de gestion sur plusieurs hôtes [27].

L'orchestration avec Swarm se concentre sur une architecture de cluster simple. Un cluster Swarm se compose d'un nœud gestionnaire (manager Swarm) qui est responsable de la gestion de cluster, et un ou plusieurs nœuds de travail (worker Swarm) qui exécutent les services.

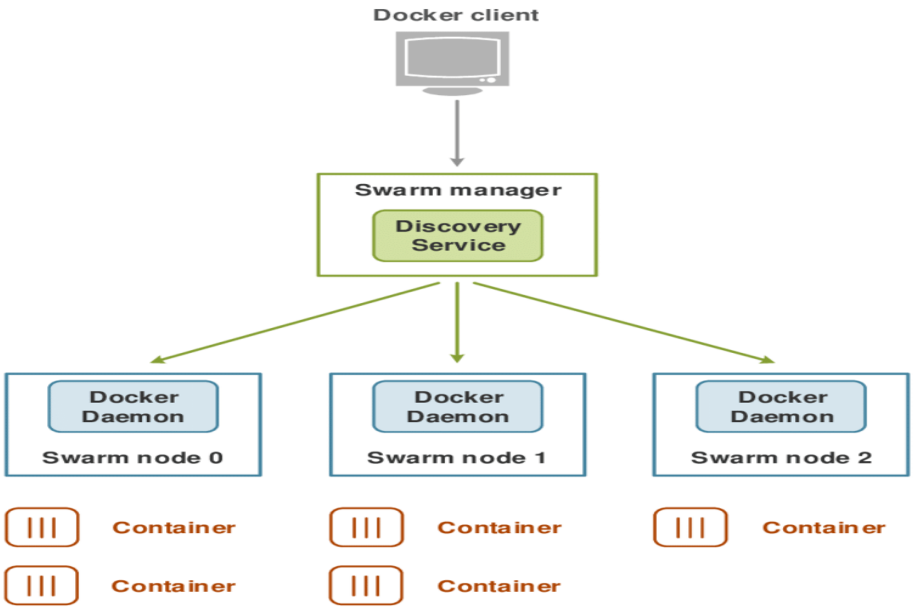

Figure II.5 : Architecture de Docker Swarm. *[28]*

II.3.3 Principaux concepts réseaux dans Docker

Un réseau est un groupe de deux appareils ou plus qui peuvent communiquer entre eux, que ce soit physiquement ou virtuellement. Le réseau Docker est un réseau virtuel créé par Docker pour

permettre la communication entre les conteneurs et le monde extérieur via la machine hôte sur laquelle le démon Docker s'exécute.

Docker est livré avec des pilotes réseau intégrés qui implémentent les fonctionnalités réseau de base :

- **Bridge** : Le réseau de type Bridge établit un pont virtuel au niveau de l'hôte. Cela permet aux conteneurs de communiquer entre eux et aussi avec l'hôte lui-même, en utilisant des adresses IP tout en restant isolés des réseaux externes. Par défaut, lorsque vous démarrez un conteneur sans spécifier de réseau, Docker le connecte automatiquement à ce réseau de pont. [29]
- **Host** : Dans le mode host les conteneurs partagent l'espace de noms réseau de l'hôte. En conséquence, l'adresse IP et les ports de l'hôte seront utilisés par le conteneur pour exécuter le processus directement sur l'hôte.
- **None :** None est un type de réseau dans lequel le conteneur n'est attaché à aucun réseau. Par conséquent, le conteneur est incapable de communiquer avec un réseau externe ou d'autres conteneurs. Il est isolé de tout autre réseau.
- **Overlay :** Overlay est un type de réseau distribué qui s'étend sur plusieurs hôtes Docker. Le réseau permet à tous les conteneurs exécutés sur n'importe quel hôte de communiquer entre eux.

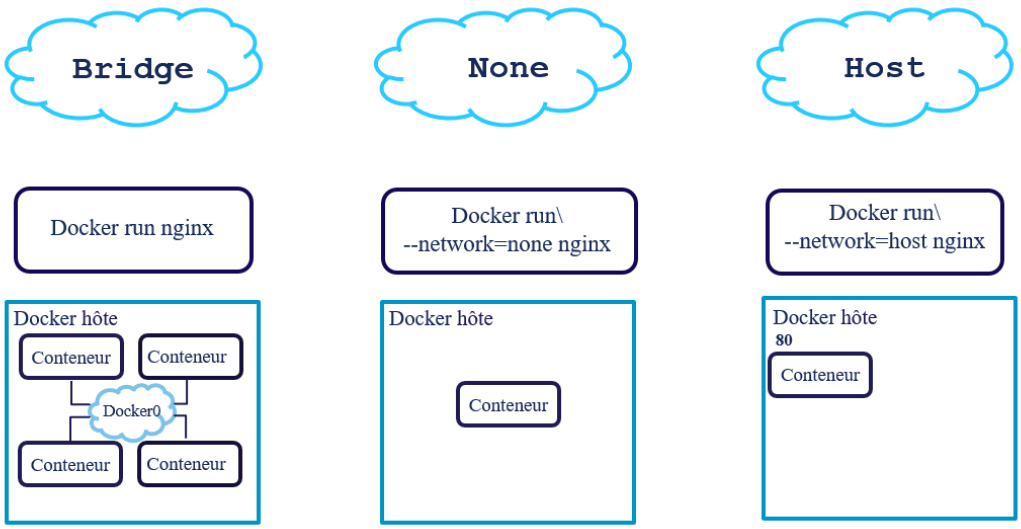

Figure II.6 : Les réseaux dans Docker. *[30]*

II.3.4 Méthodes de gestion des données avec Docker

La gestion des données avec Docker est importante, pour assurer le bon fonctionnement des applications exécutées dans des conteneurs. Les données peuvent être classées en deux catégories : persistantes et non persistantes. Les données non persistantes font référence aux données stockées à l'intérieur du conteneur Docker qui sont éphémères et sont perdues lorsque le conteneur est arrêté ou supprimé. En revanche, les données persistantes sont celles qui doivent survivre aux cycles de vie des conteneurs, et même s'ils sont supprimés telles que les bases de données, les fichiers de configuration ou les journaux d'application.

Docker propose deux méthodes pour la gestion des données persistantes : les volumes et le montage de répertoire hôtes (bind mounting). Ces mécanismes offrent la possibilité de partager des données de manière efficace entre les conteneurs Docker et l'hôte.

II.4 Kubernetes

Kubernetes, également connu sous le nom de K8s, est un système open source permettant de gérer des applications conteneurisées sur plusieurs hôtes. Il fournit des mécanismes de base pour le déploiement, la maintenance et la mise à l'échelle des applications.

Initialement développé et conçu par des ingénieurs de Google dans le cadre du projet Borg, Kubernetes a été offert à la Cloud Native Computing Foundation (CNCF) en 2015. [31]

Figure II.7 : Kubernetese Logo. *[32]*

II.4.1 L'architecture de kubernetes

Kubernetes adopte une architecture client-serveur. Un cluster Kubernetes représente une forme d'architecture de déploiement de cette technologie. L'infrastructure de base de Kubernetes est constituée de deux éléments principaux : le plan de contrôle, qui joue le rôle de serveur, et les nœuds ou machines, qui agissent comme des clients. Chaque nœud peut être soit une machine physique, soit virtuelle, et il exécute des pods, qui sont composés de conteneurs. [33]

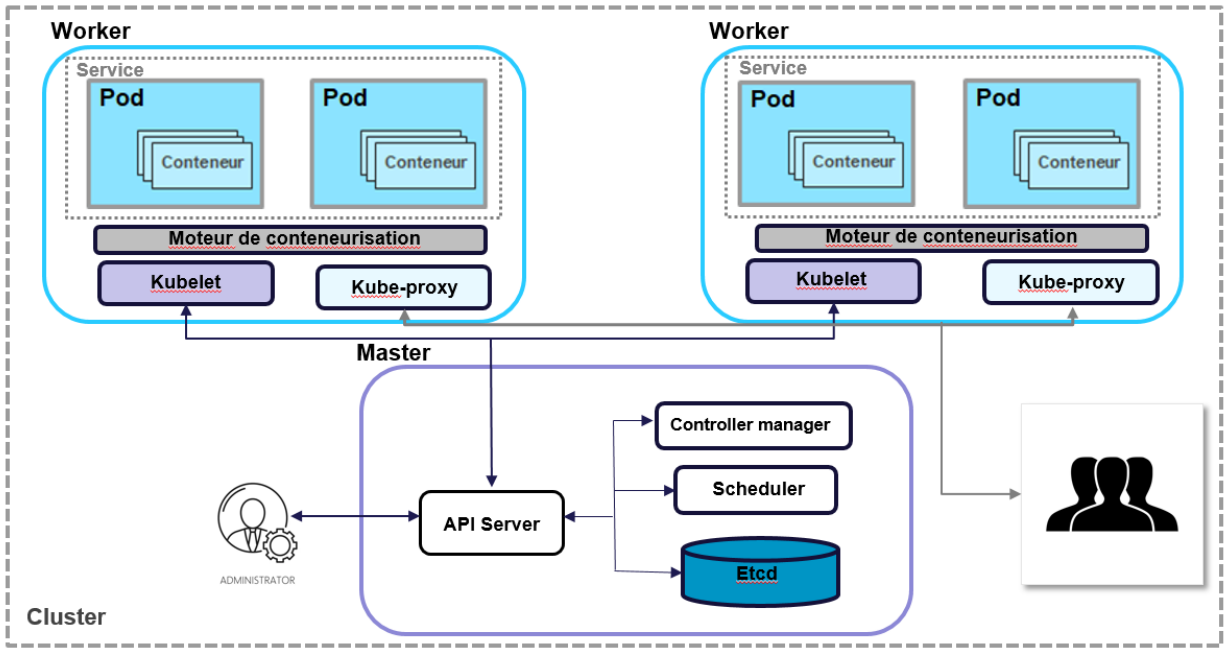

Figure II.8 : Architecture d'un cluster Kubernetes.

II.4.1.1 Le plan de contrôle (nœud maître)

Le plan de contrôle ou le nœud maître dans Kubernetes est un composant central du cluster responsable de la gestion et de la coordination des opérations dans l'environnement Kubernetes. Il héberge les principaux composants, tels que l'API server, le scheduler, le controller-manager, et Etcd. Le nœud maître contrôle également la distribution et la gestion des tâches aux nœuds de travail (workers) et assure la communication entre les différentes parties du cluster.

- **L'API server :** Une API serveur Kubernetes agit comme le point d'entrée principal du plan de contrôle, Elle gère les requêtes entrantes et sortantes. L'accès à cette API peut se faire via des appels REST (Representational state transfer), par l'intermédiaire de l'interface en ligne de commande kubectl ou encore par d'autres outils en ligne de commande tels que kubeadm.
- **Le Scheduler** : Le Scheduler est responsable de planifier l'affectation des pods aux différents nœuds de cluster, tout en garantissant que les besoins en ressources des pods sont satisfaits et en optimisant l'utilisation des ressources disponibles.

- **Le controller-manager :** Le controller-manager est responsable de l'exécution des contrôleurs, qui sont des processus qui s'exécutent en arrière-plan chargés de surveiller l'état du cluster afin d'assurer la cohérence et le bon fonctionnement du cluster. Ces contrôleurs incluent des fonctionnalités telles que la gestion des nœuds et pods. **[34]**
- **Etcd :** Etcd est une base de données clé-valeur haute disponibilité utilisée pour stocker les informations relatives à l'état du cluster.

II.4.1.2 Le Nœud Worker

Les nœuds worker, également appelés simplement nœuds, sont les machines sur lesquelles Kubernetes déploie et exécute les pods. Ces nœuds peuvent être des machines virtuelles ou des serveurs physiques. Ils sont constitués d'un ensemble de composants : Kubelet, Le container runtime et le Kube-proxy. [35]

- **Kubelet :** Le kubelet est le service principal sur un nœud worker. Sa mission est de gérer et de garantir que les pods et leurs conteneurs sont sains et fonctionnent dans l'état souhaité, et de rapporter au nœud maître l'état de santé de l'hôte sur lequel il s'exécute.
- **Kube-proxy :** Kube-proxy s'agit d'un proxy de réseau. Sa principale fonction est de gérer la connectivité réseau pour les services. Il offre plusieurs fonctionnalités clés, notamment l'équilibrage de charge, la découverte des services et la mise en œuvre de politiques réseau pour contrôler le trafic.
- **Le container runtime :** Le container runtime est le logiciel responsable de l'exécution des conteneurs. Kubernetes prend en charge plusieurs runtimes de conteneurs, notamment Docker, containerd, CRI-O.

II.4.2 Les objets Kubernetes

Les objets Kubernetes sont des entités persistantes utilisée pour représenter et gérer différents aspects d'une application ou d'un système dans un environnement Kubernetes. Ces objets décrivent l'état souhaité d'une application. Ils sont déployés et gérés par Kubernetes, qui travaille en continu pour garantir que l'état actuel du cluster correspond à celui défini par ces objets. [36]

Presque tous les objets Kubernetes comprennent deux champs d'objets imbriqués : spec et status, la spécification de l'objet (spec) est utilisée lors de la création de l'objet pour décrire l'état souhaité

de la ressource, tandis que le champ d'état de l'objet (status) est mis à jour par Kubernetes pour refléter l'état réel de la ressource dans le cluster et prendre des mesures pour corriger tout écart avec l'état souhaité.

Dans cette section, nous allons définir et expliquer certains des objets Kubernetes les plus courants :

II.4.2.1 Pod

Un Pod représente une unité de déploiement : une instance unique d'une application dans Kubernetes, qui peut consister soit en un unique conteneur soit en un petit nombre de conteneurs qui sont étroitement liés et qui partagent des ressources. [37]

II.4.2.2 Déploiement

Un déploiement Kubernetes est un objet de type ressource dans Kubernetes, qui fournit des mises à jour déclaratives pour des applications. Il vous permet de décrire le cycle de vie d'une application, en spécifiant par exemple les images à utiliser, le nombre de pods à exécuter et la façon dont ils doivent être mis à jour. [38]

II.4.2.3 ReplicaSet

Un ReplicaSet est un contrôleur Kubernetes qui garantit le maintien d'un ensemble stable de Pods à tout moment. Si un pod échoue ou est supprimé, le ReplicaSet en crée automatiquement un nouveau pour le remplacer, assurant ainsi la disponibilité et la tolérance aux pannes.

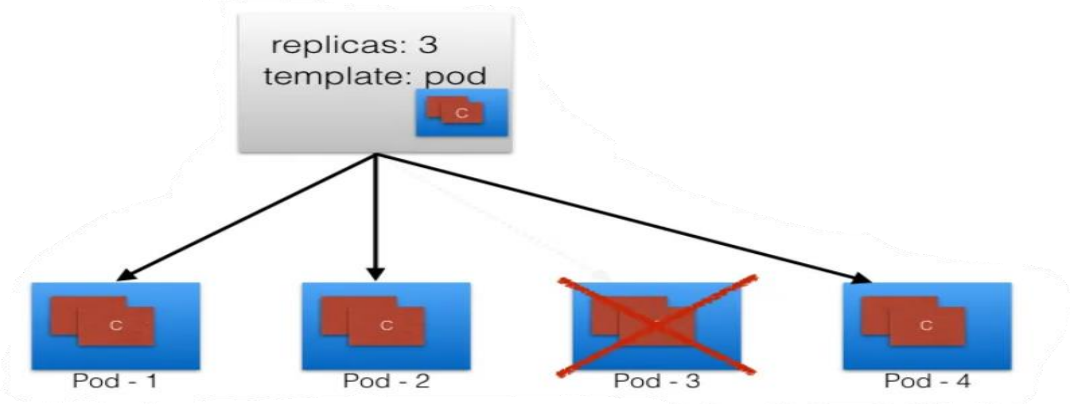

Figure II.9 : ReplicaSet Kubernetes.

II.4.2.4 Namespace

Dans Kubernetes, les espaces de noms fournissent un mécanisme pour isoler des groupes de ressources au sein d'un seul cluster en sous-clusters virtuels.

II.4.2.5 Labels et Selectors

Les labels sont des paires clé/valeur qui sont attachées aux objets Kubernetes permettent d'ajouter des informations d'identification supplémentaires aux objets, ce qui facilite leurs organisation, classification et la sélection.

Les sélecteurs, d'autre part, sont un mécanisme utilisé pour filtrer et sélectionner des ensembles spécifiques d'objets en fonction de leurs labels.

II.4.2.6 ConfigMaps et les secrets

Les ConfigMaps sont des objets Kubernetes qui permettent de stocker des données de configuration (non sensible) sous forme de paires clé-valeur. Les secrets sont similaires aux ConfigMaps mais sont destinés à stocker des données sensibles, telles que des mots de passe, des tokens d'authentification, des clés SSH, etc.

II.4.2.7 DaemonSet

Un DaemonSet est un contrôleur qui garantit que chaque nœud (ou un sous-ensemble de nœuds) exécute une copie d'un pod spécifié. Il est utilisé pour déployer des pods de manière uniforme sur tous les nœuds du cluster, souvent pour des tâches de gestion ou de surveillance. [39]

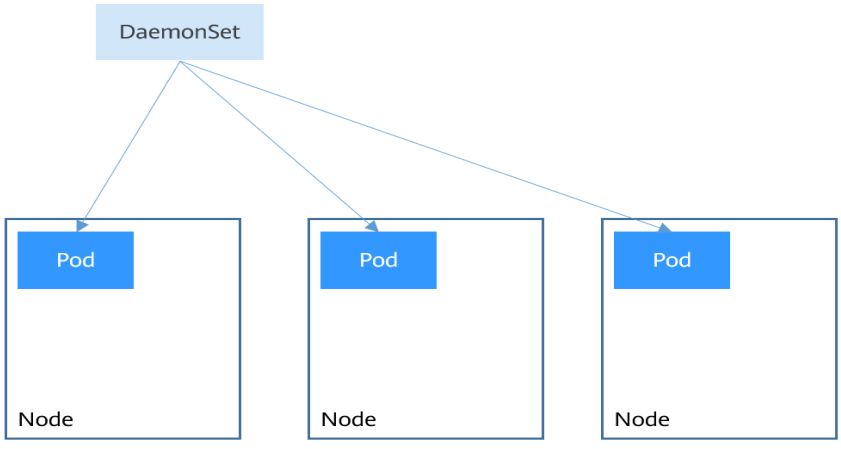

Figure II.10 : DaemonSet Kubernetes.

II.4.3 Principaux concepts réseaux dans kubernetes

Kubernetes lui-même ne gère pas directement la communication réseau entre les pods et les nœuds. Mais il fournit l'Interface de Réseau de Conteneur (CNI) pour la mise en réseau via des plug-ins CNI tels que Flannel et Calico. [40]

Selon Kubernetes, pour que la communication réseau dans un cluster Kubernetes fonctionne correctement, Les pods et les nœuds doivent pouvoir communiquer entre eux via un réseau sans NAT, ce qui permet une transparence totale et une gestion simplifiée de la connectivité entre les pods.

II.4.3.1 Service

Un service est une abstraction qui définit une logique d'accès à un ensemble de pods. Il permet de regrouper plusieurs pods sous une même adresse IP et un même port, facilitant ainsi la communication entre les composants d'une application.

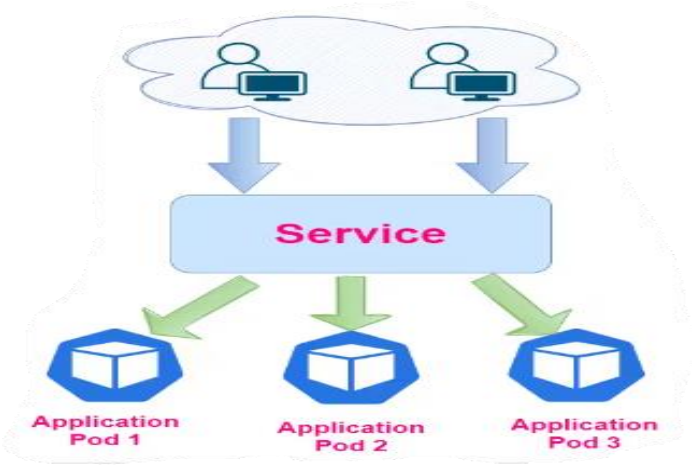

Figure II.8 : Kubernetes Services. *[41]*

Il existe quatre principaux types de services Kubernetes :

- **ClusterIP** : Est le service par défaut, il expose le service sur une adresse IP interne du cluster, accessible uniquement depuis l'intérieur du cluster. Il permet la communication entre différents composants d'une même application.
- **Nodeport** : Ce type expose le service à l'extérieur du cluster en utilisant une adresse IP de chaque nœud du cluster, à travers un port spécifique compris entre 30000 et 32767. Le service est accessible via <NodeIP>:<NodePort>.

- **LoadBalancer :** Ce type expose le service à l'extérieur du cluster à l'aide d'un load balancer externe, fourni par un fournisseur de cloud. Ce load balancer obtient une adresse IP publique (ou un nom de domaine) et distribue le trafic entrant de manière équilibrée vers les pods correspondant au service.
- **ExternalName :** Ce type est utilisé pour rediriger les requêtes vers un nom DNS externe au cluster Kubernetes. Cela permet aux applications à l'intérieur du cluster d'accéder facilement à des services hébergés en dehors du cluster, comme une base de données externe.

II.4.3.2 Ingresse

Un Ingress est un objet Kubernetes qui gère l'accès externe aux services dans un cluster, généralement du trafic HTTP .il peut fournir un équilibrage de charge, une terminaison TLS et un hébergement virtuel basé sur un nom.

II.4.3.3 Politique réseau

La politique réseau dans Kubernetes, souvent appelée Network Policy, est un mécanisme utilisé pour contrôler le trafic réseau entre les pods. Elle permet de définir des règles de communication entre les pods et de restreindre ou de permettre le trafic en fonction de divers critères tels que les labels de pods, les namespaces, et les ports. [42]

II.4.4 Gestion du Stockage dans Kubernetes

L'un des défis de Kubernetes est la nature éphémère des pods, ce qui pose un problème pour la gestion du stockage persistant, indispensable pour les applications stateful. Pour répondre à ce besoin, Kubernetes offre un ensemble de fonctionnalités permettant de séparer le stockage des cycles de vie des applications.

- **Persistant Volume (PV) :** Un persistant volume est un élément de stockage dans le cluster qui a été provisionné par un administrateur ou provisionné dynamiquement à l'aide de Storage Classes. [43]
- **Persistant Volume Claim (PVC) :** C'est une demande de stockage faite aux Persistent Volumes (PVs) par les développeurs d'applications. Créée à l'aide d'un manifeste YAML qui précise les besoins en termes de capacité et de mode d'accès, sans nécessiter de connaissances sur les détails de l'infrastructure de stockage sous-jacente.

- **StorageClass** : Une StorageClass est un mécanisme permettant de mettre à disposition dynamiquement des volumes persistants (PVs) dans un cluster. Les administrateurs définissent des classes de stockage avec des propriétés spécifiques, et les pods peuvent ensuite demander le type de stockage dont ils ont besoin. [44]

II.5 Conclusion

En conclusion, ce chapitre a été consacré à une analyse de deux technologies fondamentales dans le domaine de la conteneurisation et de l'orchestration : Docker et Kubernetes. Nous avons commencé par situer ces outils dans le contexte plus large de la méthodologie DevOps, soulignant leur rôle important dans ce domaine. Ensuite, nous avons entamé une étude de Docker, en examinant son architecture, ses outils, ses fonctionnalités de réseau et de stockage, afin de comprendre pleinement son fonctionnement. Par la suite, nous nous sommes tournés vers Kubernetes, en explorant son architecture, ses principaux objets et ses fonctionnalités réseau.

Ce chapitre fournit ainsi un aperçu général de Docker et Kubernetes, ouvrant la porte au chapitre suivant qui s'intéressera à la sécurité.

CHAPITRE III
SÉCURITÉ DANS LES ENVIRONNEMENTS CONTENEURISÉS

III.1 Introduction

Avec la croissance de l'utilisation et de la popularité de la conteneurisation, la question de la sécurité devient un grand défi. Les conteneurs offrent une approche efficace pour le déploiement d'applications, mais ils apportent également de nouveaux défis en termes de sécurité.

Dans ce troisième chapitre, nous explorerons en profondeur les enjeux de sécurité affectant l'intégrité, la confidentialité et la disponibilité des conteneurs ainsi que de leurs orchestrateurs. Nous aborderons également les bonnes pratiques à mettre en place afin de renforcer la sécurité, et garantir un environnement fiable.

III.2 La sécurité des conteneurs

La sécurité des conteneurs consiste à définir et appliquer des pratiques de création, de déploiement et d'exécution qui protègent un conteneur complet, c'est-à-dire des applications qu'il prend en charge jusqu'à l'infrastructure sur laquelle il repose. [45]

L'étude menée par S. Sultan, T. Dimitriou et I. Ahmad [46] met en lumière quatre cas essentiels sur lesquels repose la sécurité des conteneurs :

- **Cas 1 :** Garantir la sécurité des conteneurs face aux potentielles vulnérabilités des applications qu'ils hébergent.
- **Cas 2 :** Mettre en place des mesures de protection entre les conteneurs eux-mêmes pour prévenir les attaques latérales.
- **Cas 3 :** Assurer la sécurité de l'hôte lui-même, ainsi que des applications qu'il héberge, contre les menaces émanant des conteneurs.
- **Cas 4 :** Protéger les conteneurs contre les menaces provenant de l'hôte sur lequel ils sont déployés.0

III.2.1 Menaces contre les conteneurs

L'adoption croissante des conteneurs soulève des préoccupations majeures en matière de sécurité. Selon une enquête menée par Tripwire [47], qui a interrogé 311 professionnels de la sécurité informatique qui gèrent des environnements avec des conteneurs dans des entreprises de plus de 100 employés, environ 60% des organisations utilisant des conteneurs ont connu des incidents de sécurité liés à ces derniers. De plus, 47% des répondants ont déployé des conteneurs connus pour présenter des vulnérabilités, et 46% ont admis avoir déployé des conteneurs sans connaître leur état de vulnérabilité.

Cette partie de notre mémoire, se concentre sur l'analyse des menaces contre les conteneurs, en se basant sur deux références de premier plan dans le domaine de la sécurité informatique : OWASP Top 10 [48] et CVE [49].

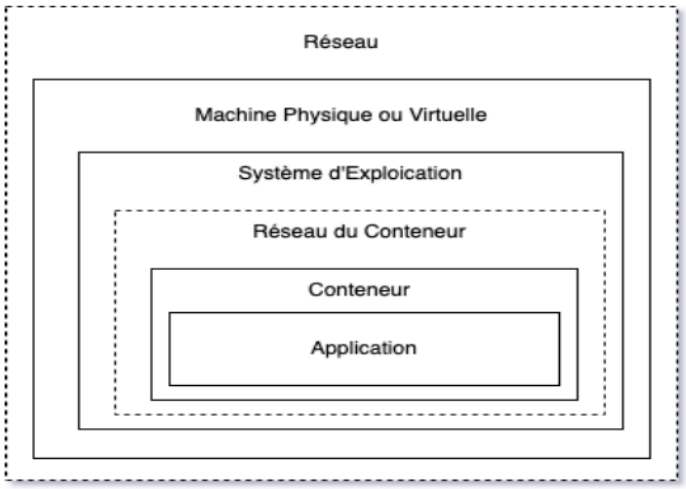

Figure III.1 : Vecteurs d'attaques des conteneurs.

La figure III.2 illustre les diverses couches qui peuvent être utilisé comme vecteurs d'attaques contre un conteneur tout au long de son cycle de vie.

Un vecteur d'attaque désigne une méthode ou un chemin par lequel un attaquant peut accéder à un système ou à un réseau informatique afin de lancer une attaque malveillante. [50]

III.2.1.1 Menace d'image

a) Vulnérabilités des Images

Certaines images pourraient comporter des vulnérabilités non résolues, ce qui expose le système à des attaques. Ce danger est accru lors du téléchargement d'images de conteneurs à partir de

registre public tels que DockerHub. [51] ,Selon l'étude menée par S. Jung en 2021 [52] , plus de 68% des images DockerHub présentaient au moins une vulnérabilité de gravité "haute" ou "critique".

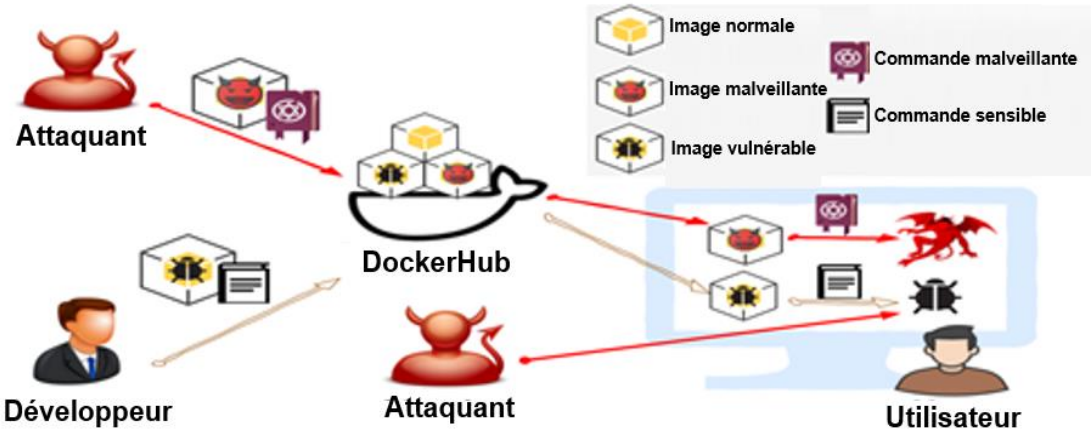

Figure III.2 : Propagation des Images Docker Malveillantes et Vulnérables via DockerHub.

b) Image non mise à jour

L'utilisation d'images non mises à jour représente une menace pour la sécurité des conteneurs. Un exemple concret est la vulnérabilité CVE-2019-14287 dans la version 1.3.1 de sudo, qui permet à des utilisateurs non autorisés de bénéficier de privilèges inappropriés. Par ailleurs, l'utilisation d'images basées sur une telle version peut compromettre la sécurité globale des conteneurs. Selon les résultats d'analyse effectués par K. Wist et D. Gligoroski sur plus de 2 500 images sur DockerHub, environ 15 % des images officielles disponibles n'ont pas été mises à jour depuis plus de 200 jours. [53]

c) L'utilisation des images non fiable

L'exécution de logiciels non fiables est un scénario à haut risque répandu dans tous les environnements. Les conteneurs, avec leur portabilité et leur facilité de réutilisation, rendent plus tentant pour les équipes d'exécuter des images provenant de sources externes qui peuvent ne pas être bien validées ou dignes de confiance. Par exemple, certaines entreprises, dans le but de maintenir leurs services et de résoudre des problèmes spécifiques, peuvent utiliser des images provenant de tiers. Cependant, ces images peuvent contenir des vulnérabilités ou des logiciels malveillants intégrés à leur insu, uniquement dans le but de résoudre le problème en question.

III.2.1.2 Mauvaise configuration

Bien que les conteneurs offrent des mesures de protection initiales, l'enjeu principal réside dans leur configuration. Une mauvaise configuration pourrait représenter des menaces de sécurité, exposant ainsi les applications et leurs données à des risques.

a) Conteneur privilégié

La principale menace réside dans l'utilisation d'un microservice exécuté en tant que superutilisateur (root) à l'intérieur du conteneur. En cas de faille dans ce service, un attaquant pourrait exploiter celle-ci et acquérir tous les privilèges à l'intérieur du conteneur, lui permettant ainsi de s'échapper du conteneur et d'accéder à d'autres parties du système hôte ou à d'autres conteneurs. [114]

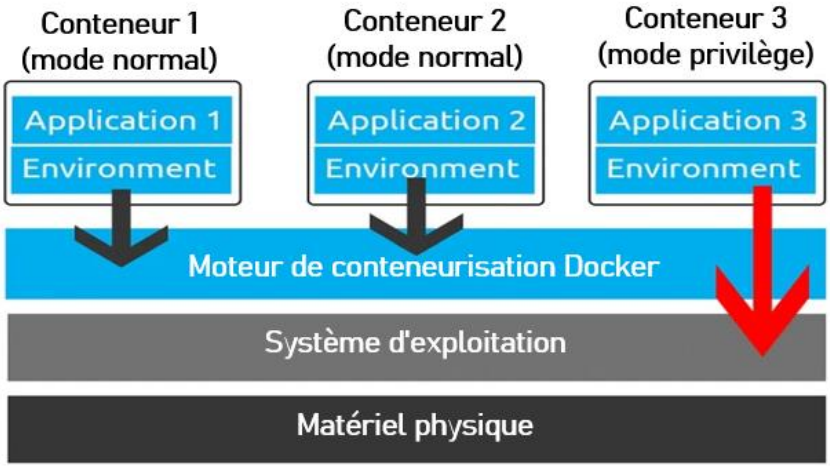

Figure III.3 : Évasion de conteneur.

b) Non limitations de Ressources

L'exécution de conteneurs sans contraintes de ressources présente un risque potentiel. Dans cette menace le conteneur devient vulnérable aux attaques de déni de service (DoS) [54]. Les attaquants peuvent exploiter cette faille en surchargeant le conteneur avec un flux de trafic malveillant, épuisant ainsi toutes les ressources disponibles sur la machine hôte. Cette situation compromet non seulement la disponibilité du conteneur lui-même, mais également celle de la machine hôte et des autres conteneurs opérant sur le même système.

c) Secrets en claire

Intégrer des secrets en clair dans n'importe quel composant de conteneur, peut poser une vulnérabilité majeure. Si une personne malveillante ou non autorisée obtient accès, elle peut facilement extraire et exploiter ces informations sensibles.

Voici un exemple d'un Dockerfile avec secrets en claire :

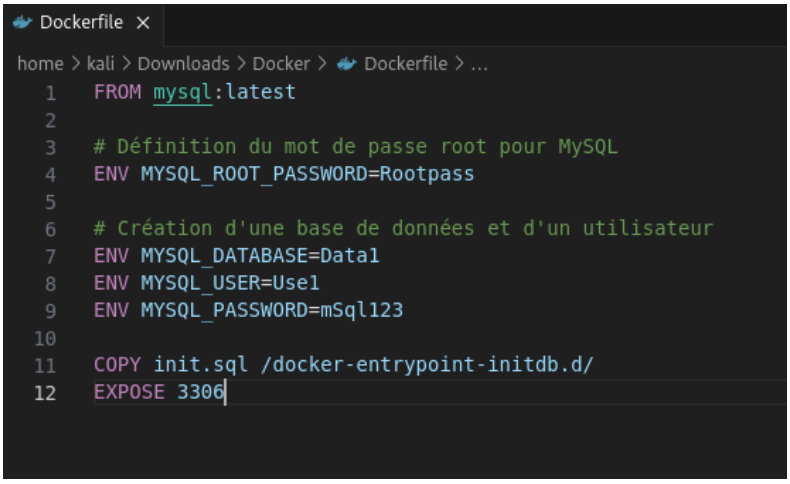

Figure III.4 : Dockerfile avec secrets en clair.

d) Absence de journalisation

Docker, bien qu'il puisse générer des logs pour les conteneurs, ne propose pas nativement de solution robuste pour la journalisation et la surveillance des événements internes tels que l'accès aux répertoires et aux fichiers, ainsi que la communication entre les différents composants (containerd, runc, daemon). Pour combler cette lacune, les utilisateurs doivent souvent recourir à des outils de journalisation externes. [55]

III.2.1.3 Menaces liées à l'infrastructure

Il est essentiel de comprendre que la sécurité des conteneurs ne se résume pas uniquement à protéger le conteneur lui-même. Souvent, nous nous focalisons uniquement sur ces aspects apparents, en négligeant les menaces qui peuvent provenir de l'infrastructure.

a) Menaces liées au moteur de conteneurisation

L'utilisation des moteurs de conteneurisation facilite la gestion, la création et l'exécution des applications conteneurisées. Cependant, ces technologies introduisent également des vulnérabilités qui peuvent compromettre la sécurité des conteneurs.

- **Mauvaise exposition de l'API :** Les moteurs de conteneurisation exposent souvent des API pour la gestion des conteneurs. Si ces API ne sont pas correctement sécurisées, elles peuvent être exploitées pour manipuler les conteneurs, extraire des données sensibles ou déployer des conteneurs malveillants. [56]
- **Utilisation d'une version obsolète :** L'utilisation d'une version obsolète d'un moteur de conteneurisation expose les conteneurs à des menaces telles que l'exploitation de vulnérabilités connues, comme la CVE-2024-21626 dans le runc v1.1.11. [57]
- **L'accès non restreint aux fichiers de configuration du daemon :** Les attaquants peuvent altérer les fichiers de configuration du daemon pour changer son comportement.

b) Menaces liées au réseau

Il existe diverses méthodes pour connecter les conteneurs. Une approche consiste à accorder aux conteneurs un accès direct à l'interface réseau de la machine physique ou virtuelle. Cependant, cette méthode présente plusieurs vulnérabilités. En cas de faille dans le réseau, les attaquants peuvent exploiter ces vulnérabilités pour compromettre la sécurité des conteneurs et intercepter le trafic. [58]

c) Menaces liées au système hôte

Les conteneurs sont déployés au sein de machines hôtes. Cette architecture partagée entre le conteneur et son hôte implique une interconnexion profonde, où ils partagent non seulement des ressources mais également un noyau commun. Cette union étroite signifie que la sécurité de l'environnement conteneurisé est intimement liée à celle de son hôte. En cas de compromission de ce dernier, tous les conteneurs exécutés sur cette machine deviennent vulnérables, exposant ainsi l'ensemble du système à des risques d'attaques et de failles de sécurité. [59]

Le tableau III-1 présente quelques menaces qui peuvent affecter à la fois les conteneurs et les systèmes hôtes.

Menace	Impact sur les conteneurs	Impact sur l'hôte
Faille dans le noyau	Une faille dans le noyau peut être exploitée pour accéder à tous les conteneurs.	Risque élevé d'escalade des privilèges et de compromission à l'échelle de l'hôte.
Falsification du système de fichiers et répertoire de stockage.	Si le répertoire de stockage et les fichiers de configuration des conteneurs ne sont pas sécurisés, un attaquant pourrait altérer ou détruire les données stockées, et modifier les configurations.	Un attaquant accédant à un conteneur avec des partitions/répertoires sensibles de l'hôte pourrait modifier des données sensibles.
Droits inappropriés	Les utilisateurs avec des droits trop élevés sur l'hôte représentent une menace pour les conteneurs.	Un attaquant compromettant un conteneur avec des droits root pourrait potentiellement obtenir des privilèges élevés sur l'hôte.
Utilisation d'un OS non spécialisé pour les conteneurs	Isolation insuffisante et performance dégradée, avec possibilité d'échapper entre conteneurs.	Surface d'attaque élargie, plus de points d'entrée potentiels pour les attaquants.
Absence de journalisation	Difficulté à diagnostiquer les problèmes, augmentant le risque de persistance des menaces.	Difficulté à retracer les activités malveillantes, et incapacité à surveiller les performances et la sécurité du système.

Tableau III-2 : Menaces de sécurité affectant les conteneurs et le système hôte.

III.2.2 Les contre-mesures de sécurité

Cette section abordera les recommandations essentielles pour protéger les environnements conteneurisés et minimiser les risques, afin de garantir leurs intégrité, confidentialité et disponibilité.

III.2.2.1 La mise à jour régulière de l'hôte et du moteur de conteneurisation

La sécurité des conteneurs dépend de celle de l'hôte. Si l'hôte présente des vulnérabilités, celles-ci peuvent potentiellement être exploitées pour affecter tous les conteneurs, vue qu'ils partagent le même noyau. De plus, si le moteur de conteneurisation est vulnérable, les conteneurs le seront également. [60]

III.2.2.2 Utilisation des images de base minimales sécurisées et vérifiées

Il est important de suivre des pratiques stricts concernant les images avant de les déployés. L'utilisation d'images de base minimales, telles qu'Alpine, permet de réduire la surface d'attaque. De plus, il est essentiel d'utiliser des images provenant de sources fiables et marquées comme "officielles", ce qui garantit un niveau de sécurité et de maintenance plus élevé. En autre, il est recommandé d'utiliser des outils d'analyse pour identifier les vulnérabilités dans les images avant de les déployer. [61]

III.2.2.3 Restriction des privilèges

La restriction des privilèges constitue une étape importante dans le processus de la sécurité des conteneurs. Tout d'abord, il est essentiel de configurer les conteneurs pour qu'ils fonctionnent avec des utilisateurs non-root, réduisant ainsi les privilèges. Ensuite, il est recommandé d'appliquer le principe du moindre privilège en limitant les privilèges de chaque conteneur afin qu'il ne dispose que des permissions minimales nécessaires à son fonctionnement. [62]

III.2.2.4 Gestion des secrets et des informations sensibles

Les conteneurs nécessitent des mécanismes spécifiques pour gérer les données sensibles telles que les clés d'API (Application Programming Interface), les jetons d'authentification, les mots de passe. Les secrets doivent être stockés de manière sécurisée et ne doivent pas être exposés directement dans les images de conteneurs ou dans les fichiers de configuration, des outils tels que docker secret et Buildkit peuvent être utilisés pour stocker de manière sécurisée et distribuer les secrets aux conteneurs au moment de l'exécution. [63]

III.2.2.5 Limitation des ressources

Pour assurer une utilisation équitable des ressources système et prévenir les attaques par déni de service (DoS), il est essentiel de limiter de manière appropriée les ressources attribuées à chaque conteneur

III.2.2.6 Surveillance et Journalisation

La journalisation est le processus de collecte des modifications d'événements à partir d'un environnement et de les enregistrer dans un journal de logs. Cet environnement peut être, une application web, et l'évènement est une situation particulière qui se produit dans cet

environnement, impliquant généralement une tentative de changement d'état dans cet environnement. [64]

La journalisation, souvent considérée comme un élément fondamental de la gestion et de la surveillance des systèmes informatiques, offre plusieurs avantages :

- **Diagnostic des problèmes :** Les journaux permettent de retracer les événements passés, facilitant ainsi l'identification et la résolution des problèmes.
- **Sécurité :** La journalisation permet de détecter les activités suspectes, ce qui facilite le renforcement la sécurité globale du système.
- **Analyse des performances :** En examinant les journaux, les administrateurs peuvent identifier les lacunes dans un système et appliquer des améliorations pour optimiser leurs processus.
- **Audit et conformité** : La journalisation fournit une trace de vérification des activités système, essentielle pour répondre aux exigences de conformité et aux normes de sécurité, telles que celles du RGPD (Règlement Général sur la Protection des Données), qui impose aux entreprises de conserver et de surveiller les journaux d'événements pour garantir l'intégrité et la confidentialité des données. [65]

a) Quelques outils et mécanisme de journalisation

Des outils tels que Portainer et Checkmk [66] peuvent être utilisés pour surveiller les conteneurs et avoir une journalisation complète.

Le driver par défaut, json-file, enregistre les logs de conteneurs au format JSON sur le système de fichiers local. Chaque ligne de sortie (stdout et stderr) d'un conteneur est capturée et stockée dans un fichier JSON. Cependant il ne configure pas de mécanisme de rotation des logs. En conséquence, les fichiers de log générés par ce driver peuvent consommer une quantité significative d'espace disque, surtout pour les conteneurs produisant beaucoup de sorties, ce qui peut mener à une saturation de l'espace disque. Pour cela Docker propose l'utilisation d'autres drivers de journalisation, tels que syslog, fluentd et journald, pour s'adapter aux différents environnements et besoins de gestion des logs. [67]

III.3 La sécurité de Kubernetes

Selon le rapport de sécurité Kubernetes 2023 de Red Hat [68] révèle que 37 % des répondants ont subi une perte de revenus ou de clients en raison d'un incident de sécurité lié aux conteneurs ou à Kubernetes. Trente-huit pour cent des répondants ont cité la sécurité comme l'une de leurs principales préoccupations dans leurs stratégies de conteneurs et Kubernetes.

Ces incidents et préoccupations soulignent l'importance de sécuriser les orchestrateurs de conteneurs tels que Kubernetes, ainsi que de suivre et de surveiller en continu la sécurité des charges de travail.

III.3.1 Enjeux de sécurité

En plus des menaces inhérentes aux conteneurs, les orchestrateurs peuvent faire face à des risques supplémentaires. La diversité et la richesse des fonctionnalités offertes par les systèmes orchestrés exigent une attention particulière aux configurations de sécurité, aux contrôles d'accès et à la gestion des ressources. Pour illustrer quelques-uns de ces risques, nous avons utilisé la matrice élaborée par Microsoft en 2020 (figure III.5).

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking	Denial of service
Application vulnerability	Application exploit (RCE)		Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files	
Exposed Dashboard	SSH server running inside container					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

Figure III.5 : Matrice des techniques d'attaque Kubernetes. *[69]*

Comme on peut le voir, la matrice présente 9 tactiques différentes. Chacune de ces tactiques comprend plusieurs techniques que les attaquants peuvent utiliser pour atteindre divers objectifs.

III.3.1.1 Communication pod à pod

La mise en réseau pod à pod dans Kubernetes offre une grande flexibilité, permettant aux pods de communiquer au sein d'un cluster. Cependant, cette liberté comporte des risques de sécurité. Par défaut, Kubernetes ne restreint pas la communication entre les pods, ce qui signifie qu'un pod compromis peut devenir un point d'attaque pour les autres. [70]

III.3.1.2 Les limites des secrets Kubernete

Bien que Kubernetes offre une solution pour gérer les secrets, ces secrets peuvent être une cible d'attaque en raison de leur encodage en base64 et non de leur chiffrement. Cette méthode d'encodage est simplement une forme de représentation des données et n'offre pas de protection contre les attaques. [71]

III.3.1.3 Compromission via les jetons

Kubernetes par défaut, attribue automatiquement un compte de service à chaque pod, avec un jeton secret pour accéder à l'API. Cependant, de nombreuses applications, comme les serveurs web, les bases de données, n'ont pas besoin de ce jeton. Cela signifie que, si une application est compromise, un agent malveillant peut obtenir les jetons de compte depuis le pod et les utiliser pour compromettre davantage le cluster. [72]

III.3.1.4 Exposition du tableau de Bord

Le tableau de bord Kubernetes est une interface web qui permet aux utilisateurs de gérer et de surveiller les clusters Kubernetes. Cependant, il peut présenter des menaces importantes s'il est mal configuré ou exposé. Si le compte de service utilisé par ce tableau de bord dispose de privilèges élevés, les risques augmentent considérablement. En effet, un attaquant qui parvient à accéder à ce tableau de bord peut utiliser ces privilèges pour effectuer des actions critiques, telles que la modification des déploiements et l'extraction des secrets au sein du cluster Kubernetes.

Un exemple notable de cette vulnérabilité est l'attaque subie par Tesla en 2018. Dans cette attaque, des attaquants ont trouvé une faille dans le tableau de bord, leur permettant d'accéder au

cluster et de déployer des conteneurs malveillants dans le but de miner de la cryptomonnaie, une pratique connue sous le nom de cryptojacking. [73]

III.3.1.5 Les risques d'etcd

Etcd est un composant critique de Kubernetes qui stocke des informations sensibles, telles que les secrets du cluster. Garder la configuration par défaut d'etcd dans Kubernetes peut poser plusieurs menaces graves pour la sécurité. L'accès en écriture ou en lecture à etcd, qui peut être effectué avec un outil tel qu'etcdctl, équivaut à obtenir les privilèges root sur l'ensemble du cluster.

III.3.2 Les 4C's de la sécurité Kubernetes Cloud Native

Les 4C's de la sécurité Kubernetes Cloud Native représentent les quatre couches de sécurité qu'il faut prendre en compte pour protéger une infrastructure basée sur Kubernetes :

1. **Code :** Cela inclut la sécurité du code source et des applications déployées dans le conteneur.
2. **Conteneur :** Cette couche concerne la sécurité des conteneurs eux-mêmes, en appliquant par exemple les bonnes pratiques citées précédemment.
3. **Cluster :** C'est la troisième couche, elle implique la sécurité de l'infrastructure Kubernetes.
4. **Cloud :** Implique la sécurité de l'infrastructure qui héberge le cluster Kubernetes, que ce soit dans le cloud ou dans des Datacenter.

III.3.2.1 Quelques mesures de sécurité pour Kubernetes

a) Chiffrerement des secrets

La gestion sécurisée des secrets dans Kubernetes est une étape essentielle pour assurer la sécurité de l'infrastructure. Comme déjà vu, Kubernetes stocke ces secrets encodés et non chiffrés.

Pour remédier à cette vulnérabilité, il est recommandé d'utiliser des outils de gestion des secrets, tels que HashiCorp Vault et Sealed Secrets. Ces outils offrent plusieurs avantages, notamment le chiffrement des secrets, la journalisation et l'audit, ainsi que le renouvellement automatique des secrets. Le renouvellement automatique des secrets ainsi que la journalisation et l'audit.

Par définition, un audit informatique est une évaluation et analyse approfondie des systèmes, des infrastructures et des politiques informatiques d'une entreprise. Son objectif est de vérifier que les contrôles de sécurité en place protègent adéquatement les actifs de l'entreprise, assurent

l'intégrité des données et s'alignent avec les objectifs stratégiques de l'organisation. L'auditeur s'assure que ces contrôles sont correctement mis en œuvre afin de limiter les risques de fuites de données et autres menaces. [85]

b) Utilisation des namespaces

La séparation des charges de travail ou des équipes dans des espaces de noms distincts au sein d'un cluster Kubernetes peut apporter des avantages importants en matière de sécurité, Si une charge de travail est compromise dans un espace de noms, cela limite les dégâts à cet espace de noms spécifique et empêche la propagation de l'attaque à d'autres parties du cluster. [74]

c) Autorisation et authentification

L'authentification et l'autorisation sont les mécanismes clés utilisés pour limiter l'accès aux ressources du cluster, pour garantir cela, toute entité cherchant à accéder au cluster, qu'il s'agisse d'un compte de service ou d'un utilisateur ordinaire, doit présenter un certificat émis par l'autorité de certification du cluster. Une fois cette entité autorisée à accéder au cluster, elle se voit attribuer un ensemble de rôles qui restreignent ses privilèges. De plus, les contrôleurs d'admission peuvent être utilisés pour appliquer des politiques supplémentaires avant d'autoriser l'accès.

La figure ci-dessus illustre le processus complet de manière concise

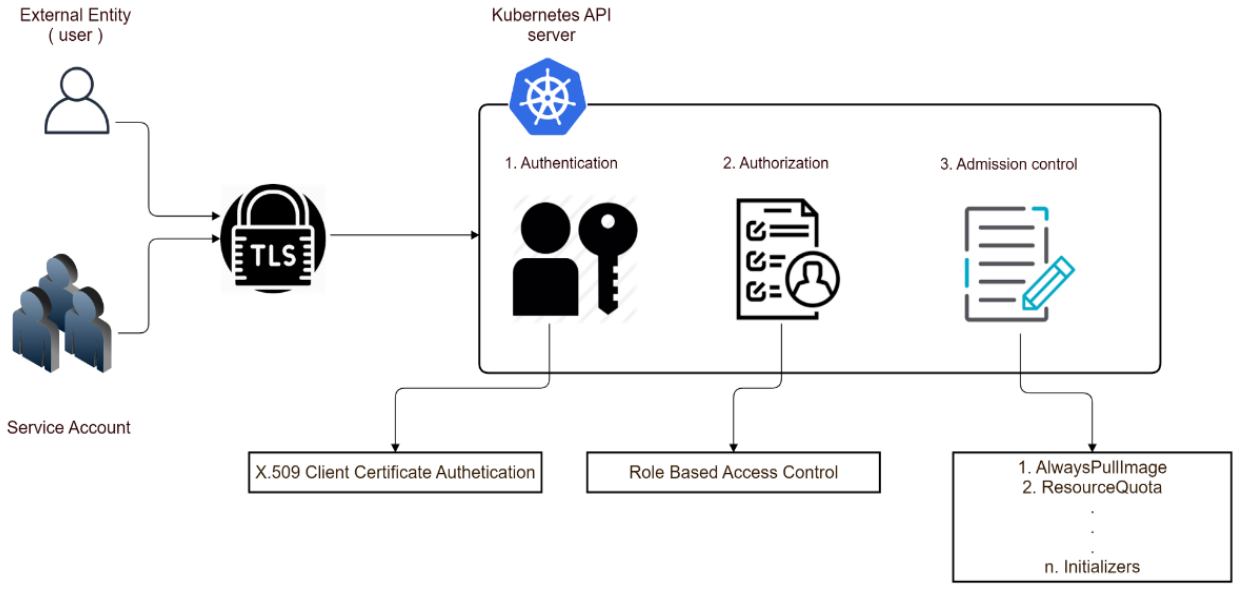

Figure III.6 : Mécanisme d'authentification et d'autorisation dans Kubernetes. *[75]*

d) Mise en œuvre des politiques réseau

La mise en place de politiques réseau dans Kubernetes est essentielle pour la sécurité. Ces politiques permettent de contrôler les communications entre les pods, limitant ainsi la surface d'attaque et empêchant les mouvements latéraux.

e) Chiffrer et restreindre l'accès à etcd.

Etcd contient des informations sensibles, ce qui en fait une cible de choix pour les attaquants. Pour cette raison, il est recommandé d'utiliser :

- Le protocole TLS pour sécuriser les communications entre les clients et les etcd.
- Un gestionnaire de secrets pour chiffrer les données.
- Des politiques RBAC (Role-Based Access Control) afin de restreindre l'accès aux seules entités autorisées.

Tout comme pour les conteneurs, il est recommandé de limiter les ressources des déploiements et de désactiver l'élévation des privilèges. En outre, il est essentiel de maintenir l'orchestrateur à jour et de mettre en place un système d'audit et de journalisation pour surveiller les activités et détecter les anomalies.

III.4 Quelques travaux relatifs

L'étude de Hugo B. Lapointe [76] explore les risques de sécurité associés aux environnements conteneurisés. L'auteur propose un système de détection d'intrusion basé sur l'apprentissage automatique, qui surveille les appels système entre les conteneurs et le système hôte afin de détecter des comportements anormaux sans modifier les conteneurs.

Maxime Bélair, examine les défis de sécurité dans les environnements conteneurisés [77]. Il propose une solution nommée SNAPPY, un framework conçu pour renforcer la sécurité des conteneurs en utilisant les espaces de noms Linux et eBPF (extended Berkeley Packet Filter), qui permet d'appliquer des politiques de sécurité programmables au niveau du noyau, facilitant ainsi la surveillance des appels système et la détection des comportements anormaux des conteneurs.

L'étude menée par M.Manuel [78] se concentre sur la mise en œuvre de contre-mesures pour traiter les différents aspects de la sécurité des conteneurs en utilisant Docker et Kubernetes. Elle offre aux entreprises soucieuses de la sécurité de leur infrastructure conteneurisée des pistes de

solutions. Plus précisément, l'étude vise à fournir des recommandations pratiques afin de sécuriser au mieux cette infrastructure.

Álvaro Martínez, se concentre sur l'analyse des problèmes de sécurité dans les architectures Kubernetes et propose des solutions pratiques pour atténuer ces vulnérabilités [76]. En explorant des aspects tels que les permissions des pods, la vulnérabilité des images et le chiffrement des secrets. Cette recherche offre aux entreprises des recommandations concrètes pour sécuriser leurs infrastructures conteneurisées.

III.5 Conclusion

En conclusion, ce chapitre a souligné l'importance de la sécurité dans les environnements conteneurisés. Nous avons commencé par définir les principales menaces de sécurité propres à ces environnements, comme la mauvaise configuration et l'utilisation d'images non fiables. Cela nous a permis d'introduire les bonnes pratiques à adopter pour réduire ces risques et sécuriser ces environnements.

Enfin, nous nous sommes concentrés sur la sécurité dans Kubernetes, en soulignant les fonctionnalités et mécanismes mis en place pour renforcer la sécurité des clusters.

.

CHAPITRE IV
MISE EN PLACE DE LA SOLUTION

IV.1 Introduction

Dans les chapitres précédents, nous avons abordé la conteneurisation et les orchestrateurs de conteneurs, en mettant l'accent sur Docker et Kubernetes, ainsi que sur les menaces associées à ces technologies. Dans ce chapitre, nous allons non seulement examiner comment sécuriser nos conteneurs et quelles étapes à suivre pour créer un environnement sécurisé, mais aussi proposer notre propre solution pour renforcer la sécurité du moteur de conteneurisation Dokcer.

Afin d'illustrer les pratiques à mettre en place, nous avons créé un site web personnel de type portfolio, qui sera conteneurisé avec Docker et déployé à l'aide de Kubernetes. Bien que le site web constitue un élément significatif du projet, l'accent est principalement mis sur la sécurisation des conteneurs et de leur orchestration. À cette fin, divers outils et pratiques de sécurité seront appliqués tout au long du projet afin de réaliser les objectifs suivants :

1. **Déploiement avec Docker et Kubernetes :** Utiliser Docker v20.10.25 pour créer des conteneurs d'applications, et de les orchestrer avec Kubernetes v1.29 pour faciliter la gestion des déploiements, la mise à l'échelle, et la sécurité.
2. **Adoption des Meilleures Pratiques de Sécurité pour les Conteneurs :** Mettre en œuvre et tester les mesures de sécurité pour protéger les conteneurs et leur orchestrateur.
3. **Adoption de l'Architecture Basée sur les Microservices** : Apprendre et appliquer l'architecture basée sur les microservices, qui offre une meilleure modularité, évolutivité, par rapport à l'architecture monolithique traditionnelle.
4. **Mise en place d'une solution de journalisation pour Docker :** Mettre en œuvre une solution de journalisation pour Docker et ses composant, afin de surveiller et analyser toutes les interactions et activités interne.

IV.2 Nouvelle approche pour l'audit et journalisation de Docker

Ce projet de fin d'études vise à proposer des solutions pratiques issues de la sécurité informatique pour renforcer la sécurité d'une architecture de conteneurs Docker gérée par Kubernetes. L'accent est mis sur des solutions gratuites et faciles à mettre en œuvre, adaptées aux petites et moyennes entreprises.

Dans ce cadre, notre contribution se manifeste par un script Bash dédié à l'application de certaines règles d'audit et la journalisation de toutes interactions et activités internes de Docker. Cela permet la détection des anomalies et la réponse aux incidents, renforçant ainsi la robustesse et la fiabilité de l'infrastructure conteneurisé.

La figure IV.1 représente le mécanisme de fonctionnement de notre solution

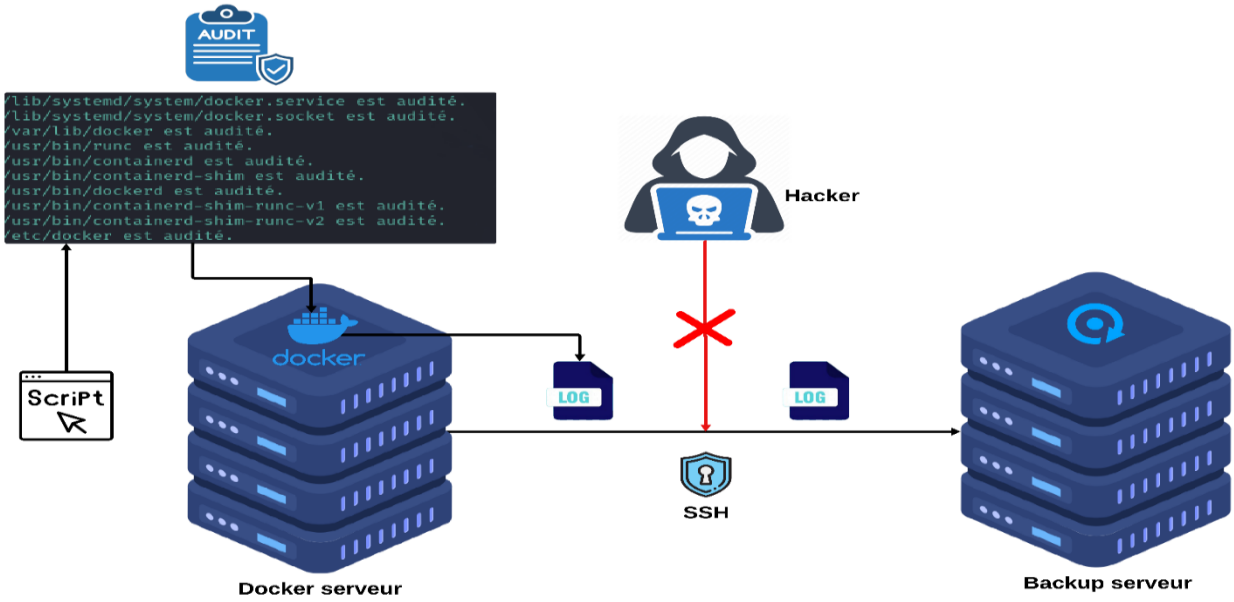

Figure IV.1 : Mécanisme de fonctionnement de la Solution.

IV.2.1 Fonctionnalités du Script

Notre script comprend les fonctionnalités suivantes :

1. **Surveillance Continue :** Le script surveille en temps réel les fichiers de configuration et les répertoires des binaires et de stockage.
2. **Journalisation Centralisée :** Les logs sont centralisés dans un emplacement unique et sécurisé pour faciliter l'analyse et le dépannage.
3. **Gestion des Logs Éphémères :** Le script assure la conservation des logs même après la suppression des conteneurs ou redémarrage.
4. **Audit de Sécurité** : Le script effectue des règles d'audits de sécurité sur les différents composants de la plateforme, détectant les anomalies et les accès non autorisés.

IV.3 Environnement de Développement

Dans cette partie, nous présentons notre environnement de travail, et l'ensemble d'outils que nous avons utilisés pour la conception et l'implémentation.

IV.3.1 Environnement de travail

Nous avons utilisé un PC Dell équipé d'un processeur Intel Core i5 de 7e génération, de 8 Go de RAM et d'un disque SSD de 256 Go, nous avons installé trois machines virtuelles (VM) (un Master et deux Worker) exécutant Kali Linux. Ce choix de Kali a été fait dans le but de mener des tests de sécurité avant le déploiement, en réponse à nos exigences spécifiques.

IV.3.2 Outils et technologies

IV.3.2.1 Vmware workstation (version 14.1.8)

VMware Workstation est un hypervisor de type 2 qui permet aux utilisateurs d'exécuter des machines virtuelles, des conteneurs et des clusters Kubernetes. [79]

IV.3.2.2 Kubeadm (version 1.29)

Kubeadm est un outil de ligne de commande qui simplifie le processus de création d'un cluster Kubernetes. Il automatise les tâches courantes nécessaires pour configurer un cluster Kubernetes de manière sécurisée et efficace, tel que l'initialisation du cluster principal et l'ajout de nœuds de travail au cluster. [80]

IV.3.2.3 Kubectl (version 1.29)

Kubectl est l'outil en ligne de commande utilisé pour contrôler un cluster Kubernetes. Il permet de déployer des applications et de gérer les objets dans cluster.

IV.3.2.4 Calico (version 3.28.0)

Calico est une solution qui permet de connecter et de sécuriser les applications dans Kubernetes ainsi que les anciennes applications non-Kubernetes, assurant une communication fluide et sécurisée entre elles. [81]

IV.3.2.5 Lynis (version 3.1.1)

Lynis est un outil de sécurité éprouvé pour les systèmes Linux, macOS ou basés sur Unix. Il effectue une analyse approfondie de l'état de santé de vos systèmes afin de soutenir le renforcement

de la sécurité et les tests de conformité. Cet outil est un logiciel open source sous licence GPL (General Public License) et est disponible depuis 2007. [82]

IV.3.2.6 CIS Benchmark

Les CIS Benchmarks sont des recommandations de configuration sécurisée établies par le Center for Internet Security (CIS), pour renforcer la résistance des technologies des organisations contre les cyberattaques. [83]

IV.3.2.7 Framework d'audit linux (version 3.1.2)

Le Framework d'Audit Linux, également connu sous le nom d'auditd, est un outil pour configurer et établir des politiques d'audit destinées aux processus de l'espace utilisateur, tels que Docker. Il permet aux administrateurs système de consigner divers événements système, offrant ainsi la possibilité d'une analyse ultérieure. [84]

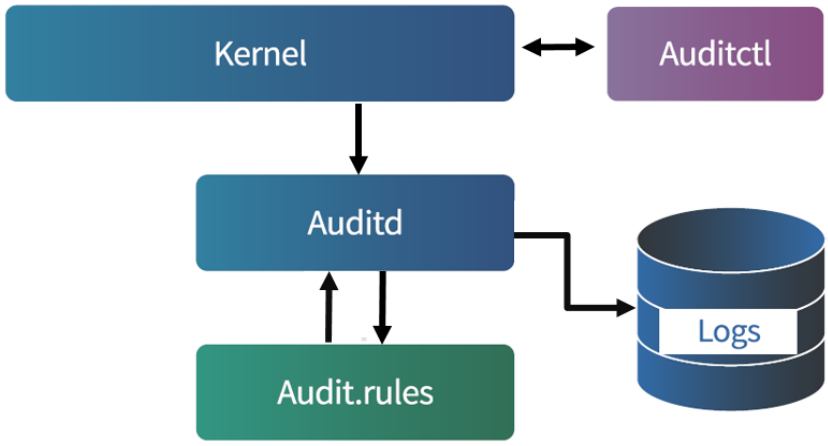

Figure IV.2 : Composant d'auditd.

Le diagramme suivant présente les différents composants qui composent auditd:

- **Auditd** : Le démon d'audit, il lit les règles d'audit et utilise auditctl pour configurer ces règles dans le noyau Linux. Il enregistre les événements d'audit dans le journal d'audit.
- **Audit logs**: Contient les journaux d'événements de toutes les règles d'audit configurées.
- **Auditctl** : Logiciel client utilisé pour gérer et contrôler le framework, ainsi que pour créer ou supprimer des règles d'audit.
- **Audit.rules** : Un fichier de configuration contenant des règles d'audit.

Tout l'audit est géré par le noyau Linux. Chaque fois qu'un appel système est effectué par un service de l'espace utilisateur, le noyau vérifie les règles d'audit pour déterminer si le service en

question possède des règles d'audit. Si c'est le cas, il envoie l'événement d'audit à Auditd, qui à son tour envoie le journal d'événements à audit.logs pour le stockage et l'analyse. [85]

IV.3.2.8 Trivy (version 0.52.2)

Trivy est un outil open-source de scanner de sécurité, développé par Aqua Security conçu pour détecter les vulnérabilités dans les conteneurs Docker, les images, et plus encore. [86]

Mise en Œuvre de la Solution

IV.4 Sécurisation de l'hôte et du moteur de conteneurisation

Avant d'entamer la création des conteneurs et de passer à leur sécurisation, il est important de s'assurer que l'hôte et moteur docker sont sécurisés. La sécurisation de ces deux est un processus complexe qui fait appel à plusieurs outils d'audit de sécurité afin d'établir un niveau de sécurité élevé. Ce processus permettra d'obtenir un environnement conforme aux recommandations du CIS Docker Benchmark. La sécurisation de ces deux éléments en deux étapes :

Dans un premier temps, nous utiliserons un outil de sécurité du système d'exploitation appelé Lynis. Cet outil nous aidera à sécuriser et à renforcer le système d'exploitation de l'hôte grâce à ses recommandations.

Une fois que le système d'exploitation de l'hôte aura été renforcé, on peut reprendre l'exécution du script généré. Ce script permet d'appliquer des règles d'audit sur les différents composants de Docker et d'assurer la journalisation de toute activité.

IV.4.1 Sécuriser l'hôte avec Lynis

Une fois que Lynis est installé, on peut facilement effectuer une analyse de système en exécutant la commande suivante : $ sudo lynis audit system

Lynis produira des informations importantes sur la posture de sécurité de notre système ainsi que diverses erreurs de configuration de sécurité et vulnérabilités, et générera également des informations sur la manière de corriger ou d'ajuster ces vulnérabilités et erreurs de configuration.

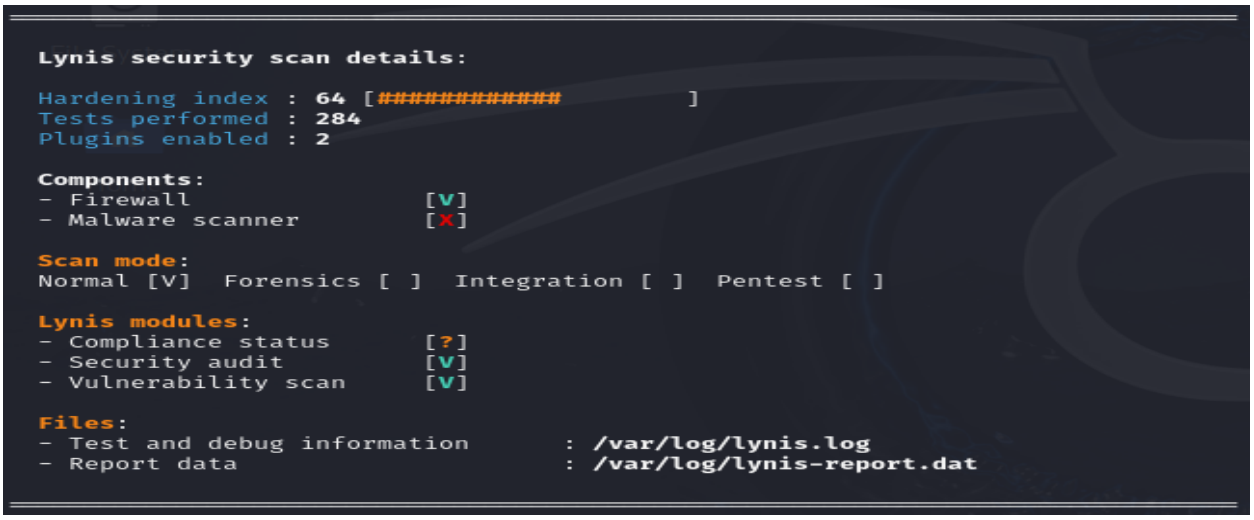

Figure IV.3 : Résultat de scan avec Lynis.

La sortie de d'analyse comprend également un score noté sur 100, ce qui nous permet d'évaluer le niveau de sécurité de notre système. Dans notre cas, ce score est de 64, comme indiqué à la figure IV.3. Nous sommes déterminés à l'améliorer en suivant les suggestions proposées dans la figure IV.4 qui détaillent les différentes configurations et mécanisme de sécurité que nous devons mettre en place.

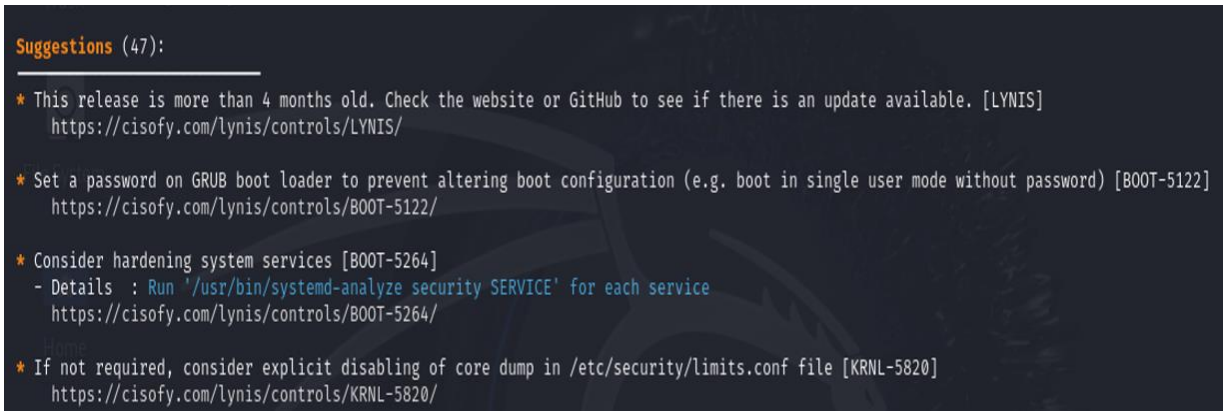

Figure IV.4 : Suggestion de Lynis.

Dans la suite, nous allons mettre en pratique certains mécanismes de sécurité en suivant attentivement les suggestions proposées. Nous cherchons ainsi à démontrer l'efficacité de ces mesures pour renforcer la sécurité de notre système.

IV.4.1.1 Désactivation des connexions root

Les bonnes pratiques de sécurité Linux recommandent de désactiver les connexions root et de créer un compte administratif ‘user1’ distinct, auquel on a attribué des privilèges en l’ajoutant au groupe sudo pour l’exécution de certaines commandes avec des privilèges root. Suivre cette étape contribuera à atténuer les menaces pesant sur le compte root et réduira la surface d'attaque globale de l'hôte.

Lorsqu’on essaie de nous switcher au compte root, le message "This account is currently not available" s’affiche comme le montre la figure IV.5, indiquant ainsi le succès de l'opération.

Figure IV.5 : Désactivation des connexions root.

IV.4.1.2 Configuration de l'authentification clés avec SSH

L'authentification basée sur des clés repose sur un principe de cryptage asymétrique, où deux clés distinctes sont générées pour le chiffrement et le déchiffrement des données. Pour créer ces paires de clés SSH, nous utilisons la commande "ssh-keygen -t rsa" sur le client. Cette commande génère une paire de clés RSA, l'une étant publique "id_rsa.pub" et l'autre privée "id-rsa".

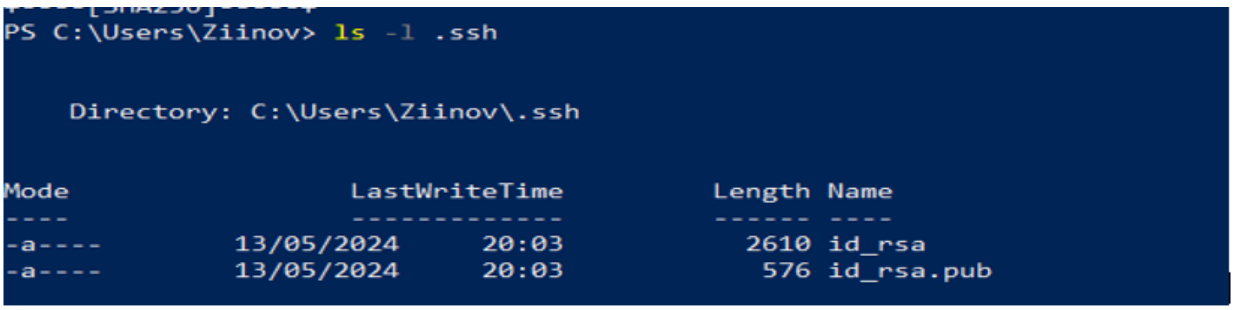

Figure IV.6 : générations d’une paire de clés RSA.

Une fois que la paire de clés est générée par le client, la clé publique est enregistrée sur le serveur pour permettre l'authentification du client lors des connexions ultérieures.

La figure IV.7 illustre le processus de transfert de la clé publique du client vers le serveur via SSH.

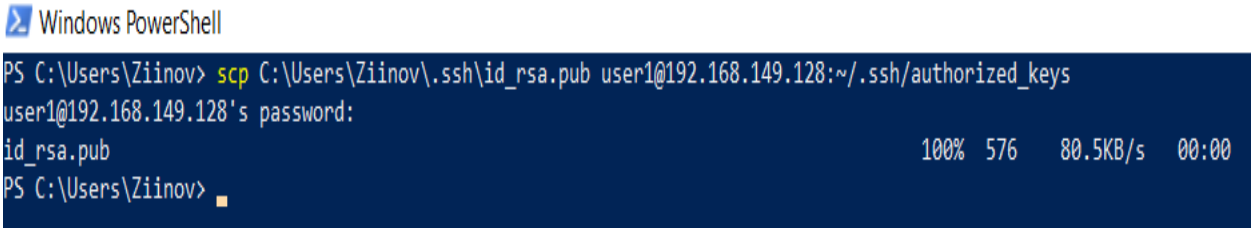

Figure IV.7 : Copie de la clé publique du client vers le serveur via SSH.

L'étape suivante consiste à désactiver complètement l'authentification par mot de passe et activer l'authentification par clés, ce qui garantira qu'aucun utilisateur ne pourra s'authentifier à distance via SSH sans sa paire de clés respective, éliminant ainsi les attaque de brute force. Cela peut être fait en modifiant le fichier de configuration /etc/ssh/sshd_config et en définissant l'option PasswordAuthentication sur "no" et PublickeyAuthentication sur "yes".

Une fois ces modifications sont effectuées, nous pourrons nous connecter en utilisant notre clé privée comme le montre la figure IV.8.

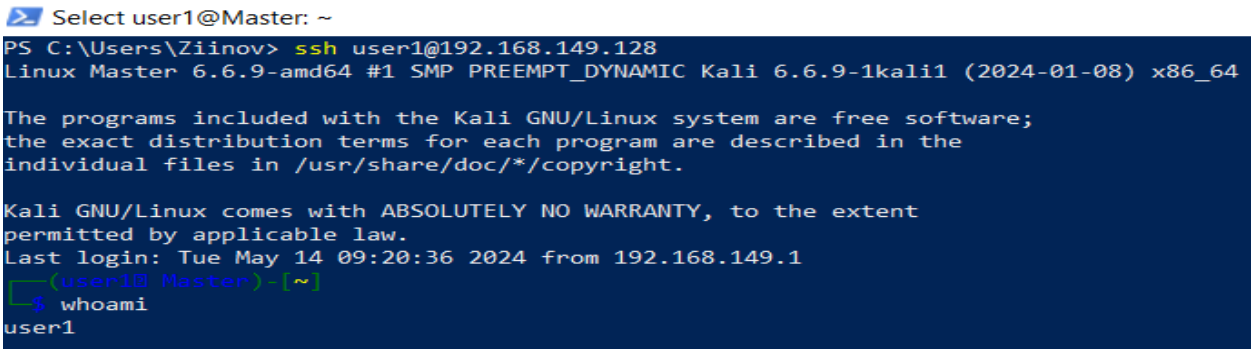

Figure IV.8 : Connexion ssh client-serveur avec clés.

IV.4.1.3 Vérification de score

Après avoir mis en œuvre d'autres recommandations fournies par Lynis, comme l'installation de PAM Manager Password pour imposer des politiques de mot de passe strictes, l'intégration d'un antivirus Sophos et la désactivation des ports USB pour prévenir toute exfiltration de données et autre mécanisme de sécurité, notre surface d'attaque a été significativement réduite.

Nous sommes maintenant prêts à exécuter à nouveau un audit du système avec Lynis pour vérifier notre score qui devrait avoir augmenté en conséquence du suivi des recommandations de sécurité. Nous remarquons une augmentation de notre score, comme illustré dans la figure IV.9.

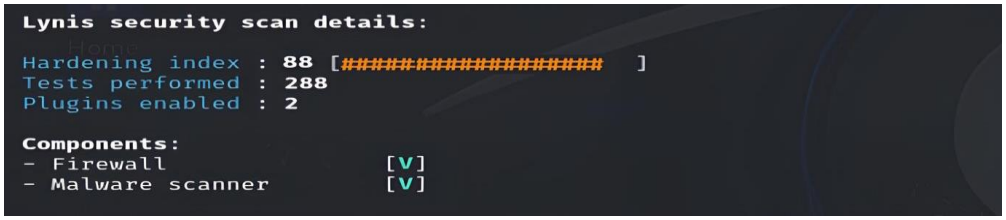

Figure IV.9 : Évolution du score de sécurité suite à l'audit avec Lynis.

IV.4.2 Configuration des règles d'audit pour docker

Pour surveiller toutes les actions et communications effectuées sur Docker, nous avons développé un script permettant de définir un certain nombre de règles d'audit pour superviser le moteur de conteneurisation

Une fois que notre script M_Audit_Docker est exécuté, il procède à toutes les vérifications nécessaires. En premier lieu, le script vérifie si l'utilitaire d'audit auditd, le cœur de notre script, est installé. Si tel est le cas, il affiche sa version ; sinon, il demande à l'utilisateur de le télécharger avant de poursuivre.

Figure IV.10 : Vérification de l'installation de l'utilitaire auditd.

Une fois l'utilitaire auditd téléchargé, le script affiche la version de auditctl ainsi qu'une liste des chemins des composants Docker, incluant des fichiers et répertoires critiques. Parmi ces chemins, on trouve les répertoires de configuration et de stockage, et plusieurs binaires essentiels. Le script

propose ensuite à l'utilisateur d'ajouter des règles d'audit pour surveiller ces chemins, garantissant ainsi que toutes les activités liées à ces composants soient enregistrées à des fins de sécurité. L'utilisateur peut choisir d'auditer tous les chemins proposés en une seule fois, ou de sélectionner individuellement les chemins à auditer.

Figure IV.11 : Proposition d'audit des chemins Docker par le script.

Si l'utilisateur choisit d'auditer tous les chemins, ce qui signifie que tous les chemins seront audités, le script va auditer et sauvegarder les règles d'audit dans /etc/audit/rules.d/audit.rules.
Afin d'assurer les services ' tolérance aux panne ' et disponibilité, notre script recommande d'enregistrer les logs générés par auditd sur un autre serveur de logs. Ce qui permet de surveiller et d'analyser les événements de sécurité de plusieurs serveurs en un seul endroit, et réduire le risque que ces fichiers soient altérés ou supprimés en cas d'attaque sur le serveur principal.

Si l'utilisateur choisit de suivre cette recommandation, le script lui demandera de spécifier l'adresse IP du serveur de logs, l'utilisateur, le chemin où les logs devront être enregistrés, ainsi qu'un mot de passe qui sera utilisé pour chiffrer les logs et un intervalle de temps pour la synchronisation des logs.

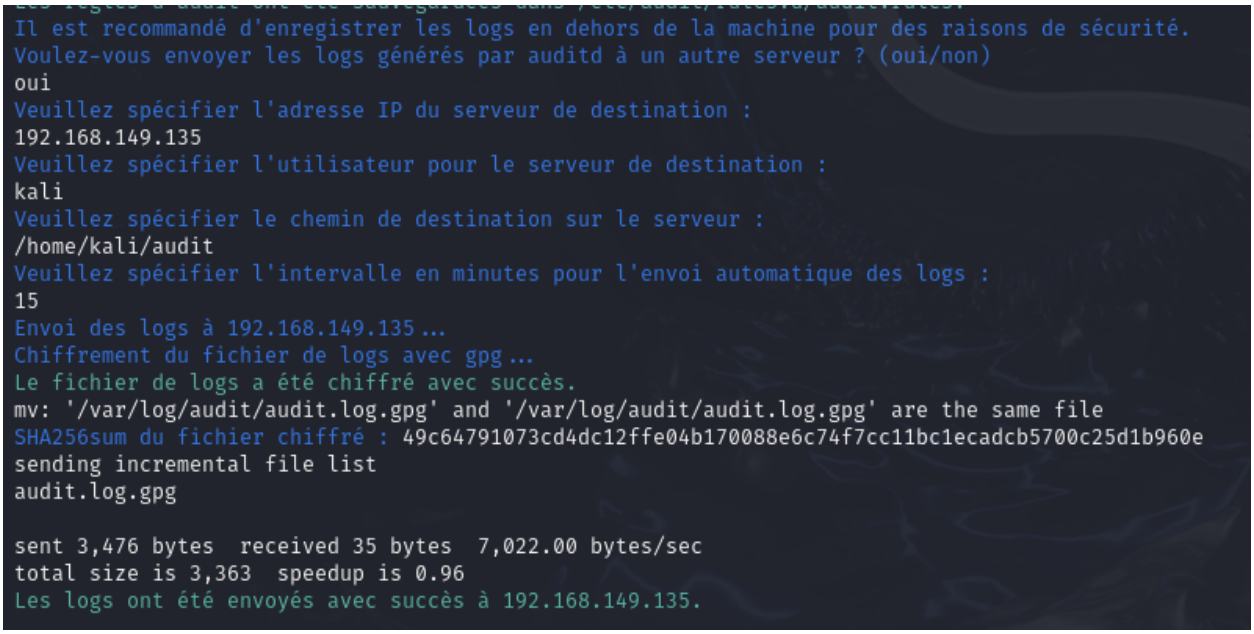

Figure IV.12 : Enregistrement des logs dans un serveur de logs.

Maintenant que le script est terminé, on peut exécuter la commande aureport -k pour voir les logs et vérifier les événements enregistrés avec les clés d'audit définies. D'après la figure IV.13, chaque log indique que l'utilisateur avec l'ID 1000 a exécuté la commande auditctl pour configurer des règles d'audit sur des fichiers et répertoires, et que ces règles ont été appliquées avec succès.

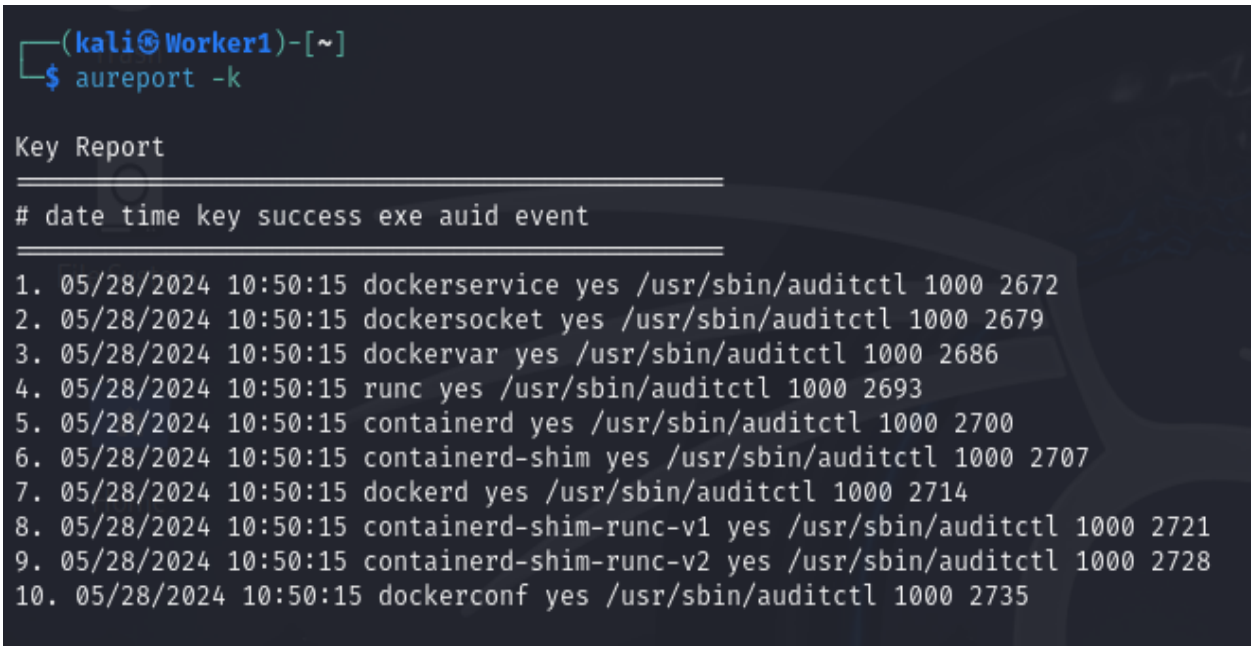

Figure IV.13 : Rapport de logs 01.

Ces logs montrent l'application des règles d'audit elles-mêmes, mais pas encore d'activités spécifiques sur les fichiers ou répertoires surveillés. Pour générer de telles activités, nous avons essayé d'ajouter une ligne à la fin du fichier docker.service , mais cette tentative a échoué en raison de permissions insuffisantes. Ensuite, nous avons créé un fichier test01 dans le répertoire de stockage.

```
┌──(kali㉿Worker1)-[~]
└─$ sudo echo "# Test modification" >> /lib/systemd/system/docker.service
zsh: permission denied: /lib/systemd/system/docker.service

┌──(kali㉿Worker1)-[~]
└─$ sudo touch /var/lib/docker/test01
```

Figure IV.14 : Génération d'activités surveillées.

Ces actions visent à déclencher des événements d'audit, permettant ainsi de vérifier que les activités spécifiques sur les fichiers et répertoires surveillés sont correctement enregistrées. Les lignes 11 et 14 dans le rapport de logs montrent ces activités.

```
Key Report
═══════════════════════════════════════════════════
# date time key success exe auid event
═══════════════════════════════════════════════════
1. 05/28/2024 10:50:15 dockerservice yes /usr/sbin/auditctl 1000 2672
2. 05/28/2024 10:50:15 dockersocket yes /usr/sbin/auditctl 1000 2679
3. 05/28/2024 10:50:15 dockervar yes /usr/sbin/auditctl 1000 2686
4. 05/28/2024 10:50:15 runc yes /usr/sbin/auditctl 1000 2693
5. 05/28/2024 10:50:15 containerd yes /usr/sbin/auditctl 1000 2700
6. 05/28/2024 10:50:15 containerd-shim yes /usr/sbin/auditctl 1000 2707
7. 05/28/2024 10:50:15 dockerd yes /usr/sbin/auditctl 1000 2714
8. 05/28/2024 10:50:15 containerd-shim-runc-v1 yes /usr/sbin/auditctl 1000 2721
9. 05/28/2024 10:50:15 containerd-shim-runc-v2 yes /usr/sbin/auditctl 1000 2728
10. 05/28/2024 10:50:15 dockerconf yes /usr/sbin/auditctl 1000 2735
11. 05/28/2024 10:51:46 dockerservice no /usr/bin/zsh 1000 2752
12. 05/28/2024 10:52:10 dockervar no /usr/bin/zsh 1000 2753
13. 05/28/2024 10:52:10 dockervar no /usr/bin/zsh 1000 2754
14. 05/28/2024 10:52:13 dockervar yes /usr/bin/touch 1000 2755
```

Figure IV.15 : Rapport de logs 02.

Afin d'analyser l'événement avec l'ID 2752 ligne 11, nous avons utilisé l'outil ausearch, qui nous permet d'extraire des informations détaillées à partir des journaux d'audit générés par Auditd. Le rapport généré indique que l'utilisateur "kali" a tenté d'ouvrir le fichier /lib/systemd/system/docker.service en mode écriture à l'aide d'un appel système. Cependant, cette

opération a échoué en raison d'une erreur de permission"EACCES (Permission denied)". L'événement est enregistré sous la clé "dockerservice", ce qui suggère que l'accès à ce fichier est surveillé conformément à nos règles d'audit.

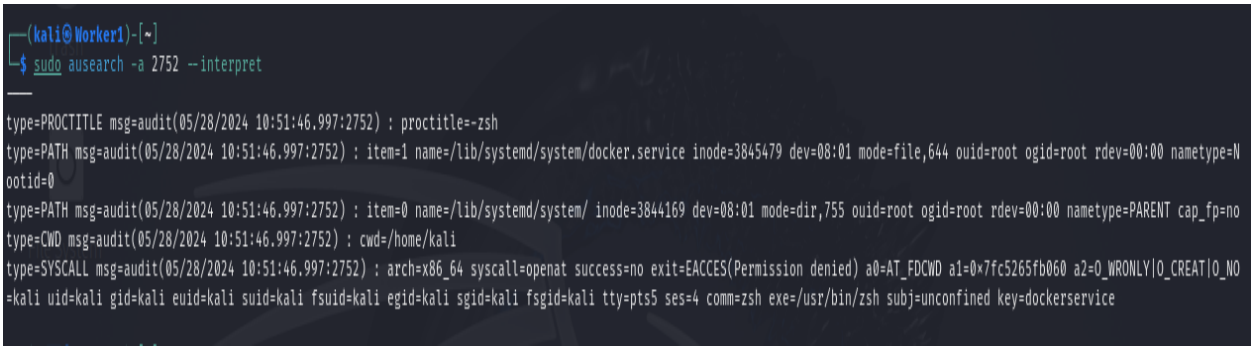

Figure IV.16 : Analyse de l'événement ID 2752.

Maintenant que nous avons mis en place des mesures de protection pour l'hôte, et bien que la sécurité absolue ne puisse être garantie, il est temps de passer à la création et à la sécurisation des conteneurs, ainsi qu'à leur déploiement dans le cluster Kubernetes.

IV.5 Développement et sécurisation des conteneurs

Dans cette section, nous allons détailler le processus de création de conteneurs Docker sécurisés pour notre site. Nous commencerons par élaborer les conteneurs en suivant les bonnes pratiques de sécurité pour minimiser les vulnérabilités. Ensuite, nous effectuerons des tests pour évaluer la sécurité de notre solution et son bon fonctionnement. Enfin, une fois que nous aurons validé l'efficacité et la sécurité de nos conteneurs, nous les déploierons dans notre cluster de production.

IV.5.1 Création de DockerFile

Afin de créer un Dockerfile sécurisé et optimisé, voici les étapes que nous avons suivies :

1. **Base de l'image** : Nous avons utilisé une image officielle de base php:8.3.8RC1-fpm-alpine3.20 basée sur Alpine Linux, ce qui permet de réduire significativement la taille de l'image et la surface d'attaque.
2. **Création d'un utilisateur non-root** : Nous avons créé un utilisateur non-root nommé massyl, ce qui est une mesure de sécurité essentielle, qui permet de réduire les risques liés à l'exécution de processus avec des privilèges élevés.
3. **Désactivation de l'accès au shell pour l'utilisateur root** : Nous avons désactivé l'accès au shell pour l'utilisateur root. Cela empêche les connexions interactives de l'utilisateur root.
4. **Utilisation du multi-stage build** : Nous avons utilisé le multi-stage build pour réduire la taille de l'image finale.

```
docker > Dockerfile > ...
  1  # Étape 1 : Image de build
  2  FROM php:8.3.8RC1-fpm-alpine3.20 AS build
  3
  4  LABEL maintainer="MassylYb"
  5
  6  # Installation des dépendances nécessaires pour la compilation
  7  RUN apk add --no-cache $PHPIZE_DEPS
  8
  9  # Installation et activation des extensions PHP nécessaires
 10  RUN docker-php-ext-install mysqli pdo pdo_mysql
 11
 12  # Étape 2 : Image finale
 13  FROM php:8.3.8RC1-fpm-alpine3.20
 14  LABEL maintainer="MassylYb"
 15
 16  # Ajout du groupe et de l'utilisateur
 17  RUN addgroup -S ms-group && adduser -S -G ms-group massyl
 18
 19  # Désactivation de l'accès au shell pour l'utilisateur root
 20  RUN sed -i 's/^root:.*/root:!::0:::::/' /etc/shadow
 21
 22  # Copie des extensions compilées depuis l'image de build
 23  COPY --from=build /usr/local/lib/php/extensions/* /usr/local/lib/php/extensions/
 24  COPY --from=build /usr/local/etc/php/conf.d/ /usr/local/etc/php/conf.d/
 25
 26  # Définition du répertoire de travail
 27  WORKDIR /var/www
 28
 29  # Passage à l'utilisateur non-root
 30  USER massyl
```

Figure IV.17 : Dockerfile php-fpm alpine.

IV.5.2 Scanne de l'image avec Trivy

Après la build de Dockerfile, il est essentiel de tester et scanner l'image avec Trivy, pour nous assurer que notre image Docker est non seulement fonctionnelle mais aussi sécurisée, minimisant ainsi les risques avant son déploiement en production.

Le résultat de scanne n'a détecté aucune vulnérabilité dans l'image Docker, ce qui est un bon indicateur de sécurité.

```
┌──(kali㉿Master)-[~]
└─$ trivy image my-app
2024-06-02T07:46:09.176-0400    INFO    Detected OS: alpine
2024-06-02T07:46:09.176-0400    INFO    Detecting Alpine vulnerabilities...
2024-06-02T07:46:09.176-0400    INFO    Number of PL dependency files: 0

my-app (alpine 3.20.0)
══════════════════════
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

Figure IV.18 : Résultat de scanne avec Trivy.

IV.5.3 Test de l'image

Avant d'utiliser notre Dockerfile dans Docker Compose en toute confiance, il est essentiel de tester l'image dans un conteneur afin de garantir que la configuration actuelle limite efficacement les privilèges de l'utilisateur "massyl", assurant ainsi une isolation et une sécurité.

D'âpres la figure IV.19, nous observons que la commande whoami confirme que le conteneur utilise l'utilisateur non-root "massyl", respectant ainsi les bonnes pratiques de sécurité. De plus, en tentant d'installer nmap, nous avons rencontré des erreurs "Permission denied". Ces erreurs indiquent que l'utilisateur "massyl" n'a pas les permissions nécessaires pour effectuer des installations. Cela est dû aux restrictions de privilèges imposées à cet utilisateur

La restriction des privilèges est particulièrement importante car, en cas de compromission du conteneur, elle limite la surface d'attaque et la capacité d'un attaquant à causer des dommages. Plus précisément, si un attaquant parvient à accéder au conteneur, les privilèges restreints l'empêcheront de modifier des fichiers sensibles, d'installer des logiciels malveillants, ou d'accéder à des ressources critiques. Cela peut réduire le risque que l'attaquant puisse également compromettre l'hôte ou d'autres conteneurs sur le même système.

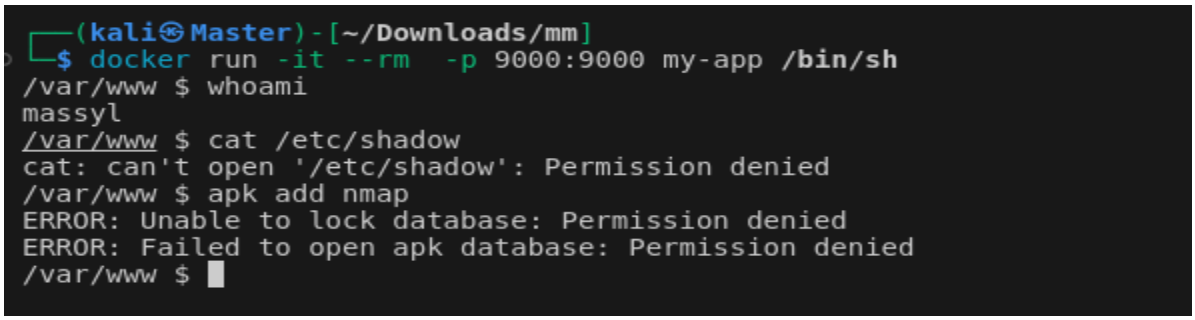

Figure IV.19 : Test de l'image.

IV.5.4 Création de Docker-compose

Dans le cadre de notre stratégie de déploiement actuelle, nous avons mis en place une série de mesures pour sécuriser et optimiser nos conteneurs, qui servent actuellement d'environnement d'essai avant notre transition vers Kubernetes pour le déploiement en production. Alors que nous nous apprêtons à migrer vers Kubernetes, notre intention est de consolider et d'améliorer ces pratiques afin de garantir une infrastructure fiable.

1. **Sélection et sécurisation des images Docker :** nous avons choisi Docker Compose et utilisé des images officielles de MySQL,phpmyadmin et nginx, soigneusement vérifiées avec Trivy pour garantir leur sécurité.

2. **Gestion optimale des ressources :** Notre configuration inclut des limitations de ressources pour chaque composant, notamment en termes de CPU et de RAM, afin d'optimiser l'utilisation des ressources système et de prévenir les attaques DDoS.
3. **Sécurisation des données sensibles :** Nous avons mis en place une approche sécurisée en utilisant un fichier d'environnement pour stocker les données sensibles telles que les mots de passe.
4. **Isolation des services :** Nous avons configuré des réseaux spécifiques pour chaque service, garantissant une isolation optimale entre eux et renforçant ainsi la sécurité de notre infrastructure.
5. **Limitation des privilèges des processus :** Pour renforcer encore la sécurité des conteneurs, nous avons activé l'option no-new-privileges, limitant les privilèges des processus à l'intérieur des conteneurs et empêchant les attaques par élévation de privilèges.

Voici un aperçu d'une partie de notre configuration Docker Compose

```
nginx:
  image: nginx:1-alpine
  container_name: nginx_container
  restart: always
  ports:
    - 8000:80
  volumes:
    - ../src:/var/www
    - ./nginx:/etc/nginx/conf.d
  networks:
    - frontend
  deploy:
    resources:
      limits:
        cpus: '0.25' # Limite le conteneur à 25% d'un CPU
        memory: 256M # Limite la RAM à 256MB
  security_opt:
    - no-new-privileges:true
  user: 1000:1000
```

Figure IV.20 : Configuration de Docker-compose.

IV.5.5 Préparation pour la migration vers Kubernetes

Maintenant que nous avons sécurisé nos images et conteneurs, il est temps de lancer notre docker-compose et d'évaluer les résultats obtenus.

La page d'accueil du portfolio se charge correctement, affichant les informations du créateur, ses compétences et ses certifications avec un défilement (Figure IV.21).

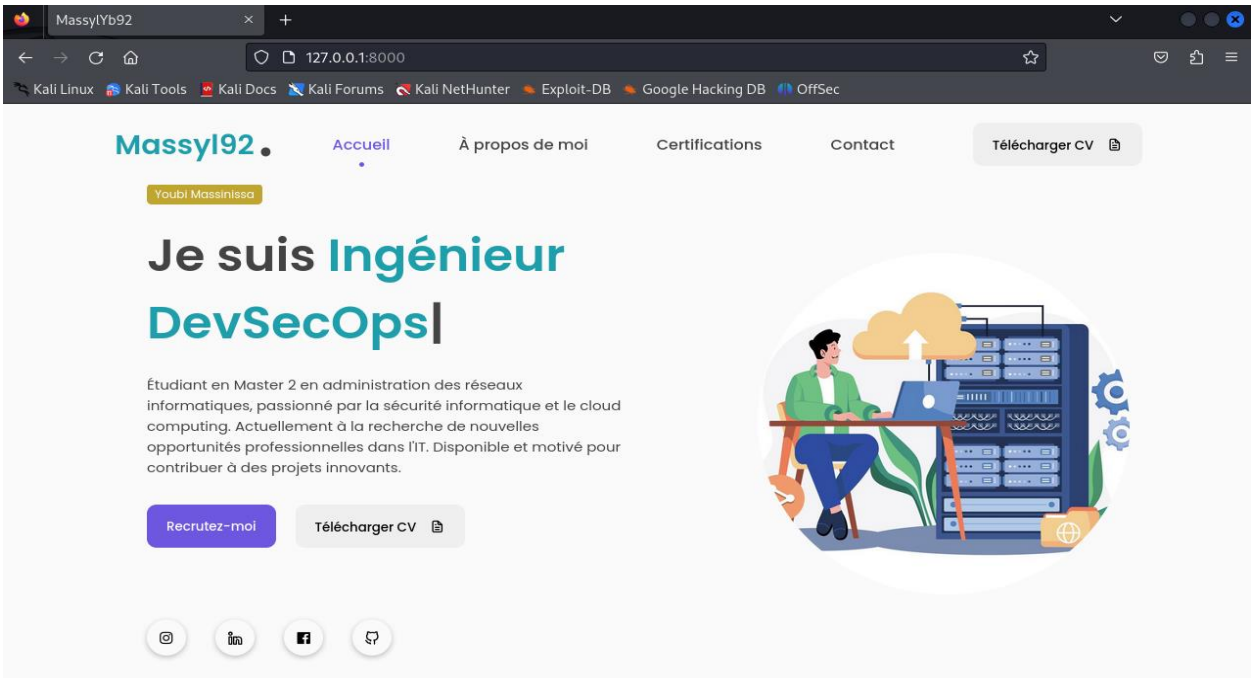

Figure IV.21 : Page d'accueil Portfolio.

La page contient également un formulaire de contact permettant aux visiteurs, clients ou employeurs de contacter le créateur (Figure IV.22).

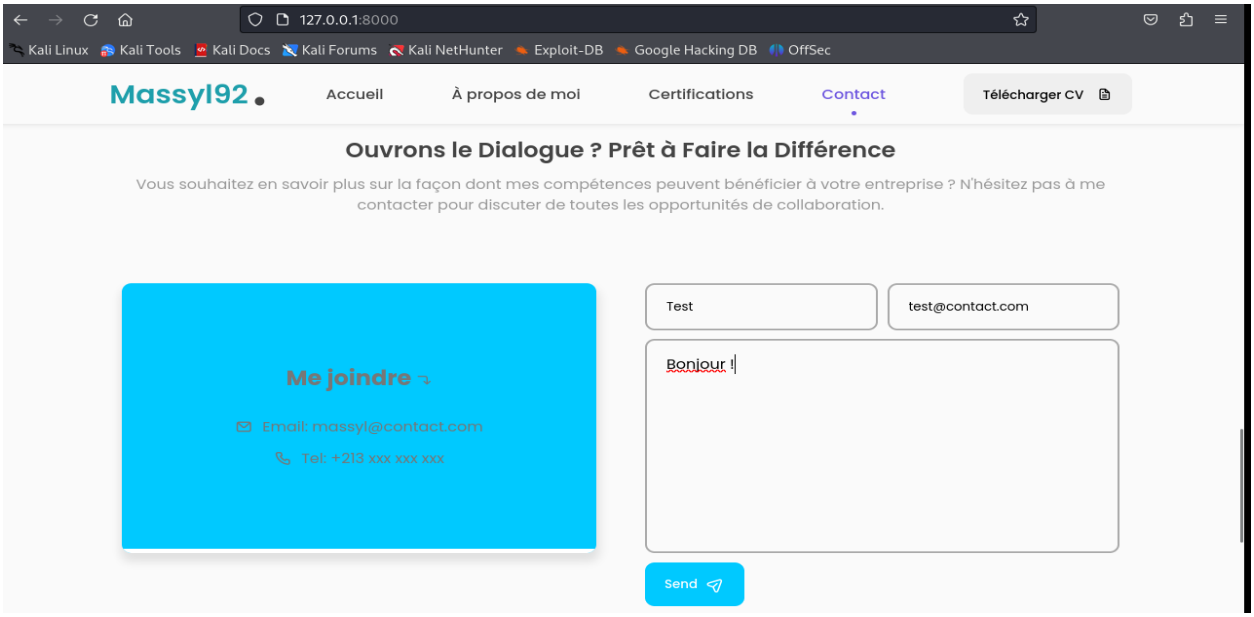

Figure IV.22 : Formulaire de contact.

Le serveur Nginx redirige efficacement les requêtes vers le conteneur PHP, qui les traite et les insère dans la base de données MySQL (Figure IV.23).

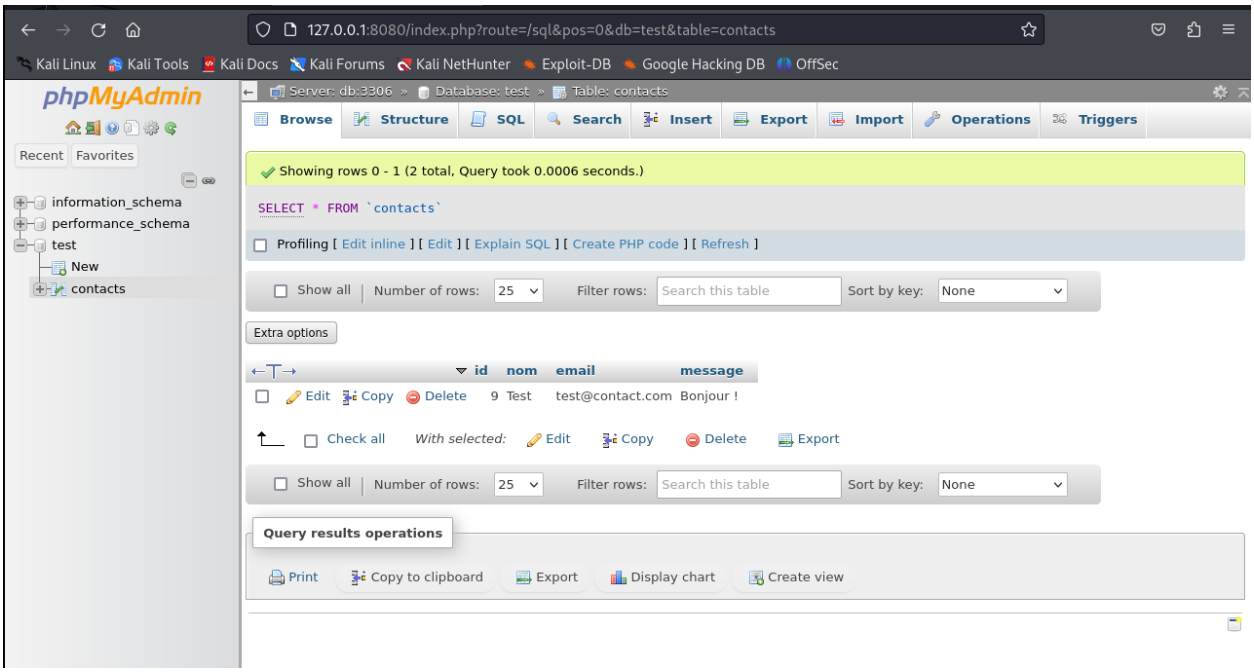

Figure IV.23 : Base de données MYSQL.

Grâce à ces résultats, nous avons atteint notre objectif initial : valider le bon fonctionnement de nos conteneurs. Cette validation nous permet désormais d'envisager la migration vers Kubernetes avec confiance. Kubernetes nous offrira une meilleure orchestration des conteneurs.

IV.6 Mise en place de cluster Kubernetes et déploiement sécurisé

Après avoir étudié la sécurité des conteneurs et validé le bon fonctionnement de nos conteneurs, nous entamons maintenant la deuxième étape de notre projet : la mise en place d'un cluster Kubernetes sécurisé, où nous explorerons la configuration et la sécurisation de notre cluster Kubernetes, ainsi que le déploiement des conteneurs.

IV.6.1 Sécurisation du cluster Kubernetes

Maintenant que notre cluster est initialisé, nous allons passer à la phase dédiée à la sécurité où nous suivrons quelques bonnes pratiques pour garantir la sécurisation de notre cluster Kubernetes.

IV.6.1.1 Création d'un Namespace

Pour assurer la sécurité de notre cluster Kubernetes, nous débutons par la création d'un namespace que nous nommerons "mywebapp". Cela nous permet de fournir un espace de travail isolé et organisé. En définissant ensuite une politique réseau spécifique, nous pouvons restreindre l'accès à nos pods, renforçant ainsi la sécurité.

Figure IV.24 : Création de namespace mywebapp.

IV.6.1.2 L'ajout d'un utilisateur : authentification et autorisation

Dans le cadre de l'ajout et de l'administration d'utilisateurs dans notre cluster, la gestion des accès joue un grand rôle dans la sécurité de l'environnement. Dans cette section, nous décrirons le processus que nous avons suivi pour ajouter un nouvel utilisateur afin d'administrer le cluster.

a) Génération de la clé privée et de demande de signature de certificat (CSR) :

La première étape pour ajouter un nouvel utilisateur à notre cluster Kubernetes consiste à générer une paire de clés (clé privée et clé publique) et une demande de signature de certificat (CSR).

Pour la génération de clé privée, nous avons utilisé l'outil OpenSSL. Cette opération a permis de générer une clé privée de 2048 bits, stockée dans le fichier massyl.key. Ensuite, nous avons utilisé cette clé privée pour créer une demande de signature de certificat (CSR), stockée dans le fichier massyl.csr.

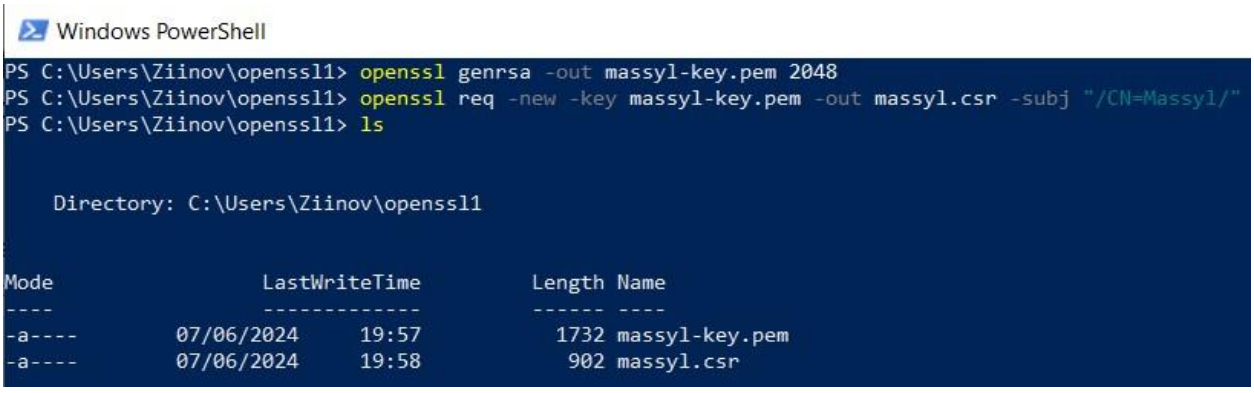

Figure IV.25 : Génération de Clé Privée et de CSR.

b) Soumission et approbation de CSR

Après avoir généré la clé privée et le CSR, nous devons soumettre cette demande via l'objet CertificateSigningRequest à API Kubernetes ou plus précisément pour le Kubernetes Certificate Controller qui signe le certificat en utilisant l'autorité de certification (CA) configurée pour le cluster Kubernetes.

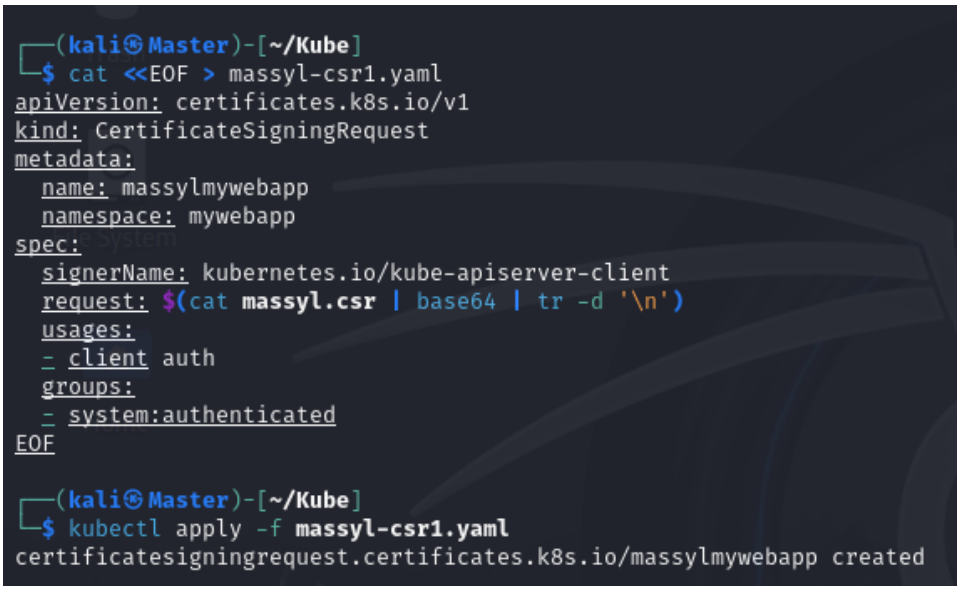

Figure IV.26 : YAML configuration de CertificateSigningRequest.

Une fois le fichier YAML appliqué , Kubernetes crée un objet CertificateSigningRequest. Cet objet doit être approuvé par un administrateur pour que le certificat soit émis. L'état de la demande passe de "Pending" à "Approved,Issued". La figure IV.27 montre le processus d'approbation de la demande de certificat.

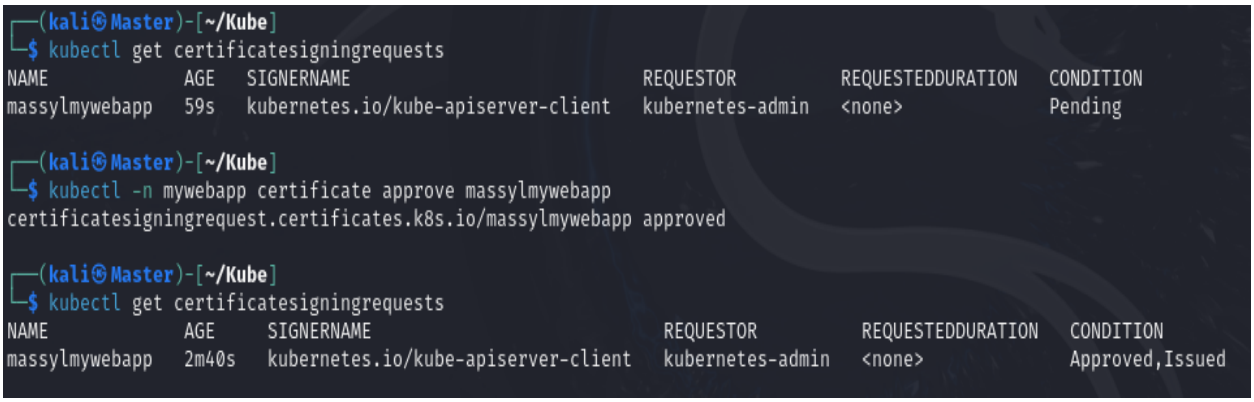

Figure IV.27 : Approbation de CSR.

Maintenant que le CSR est signé nous devons récupérer le certificat en utilisant la commande " kubectl get csr massylmywebapp -o jsonpath='{.status.certificate}' | base64 --decode > massyl.crt , qui sera utilisé avec la clé prive généré afin de configurer kubectl pour l'authentification et l'accès au cluster .

IV.6.1.3 Gestion des privilèges de nouvel utilisateur

Par défaut le nouvel utilisateur créé, il n'a aucun privilège pour interagir avec les ressources du cluster. Par conséquent, il est de la responsabilité de l'administrateur du cluster d'attribuer de manière explicite des permissions appropriées aux utilisateurs en fonction de leurs besoins opérationnels.

Pour cela, nous avons été chargés de configurer les permissions pour l'utilisateur Massyl, à travers un objet Kubernetes appelé Role. Afin qu'il puisse effectuer des tâches spécifiques dans le namespace mywebapp. Pour ce faire, nous avons défini un ensemble de permissions bien sélectionnées qui vont lui permettre de gérer divers types de ressources telles que les pods, les configmaps, les secrets, les deployments, et statefulsets.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: mywebapp
  name: mywebapp-role
rules:
- apiGroups: [""]
  resources: ["pods", "configmaps"]
  verbs: ["get", "list", "watch", "create", "update", "delete"]
- apiGroups: ["apps"]
  resources: ["deployments", "replicasets"]
  verbs: ["get", "list", "watch", "create", "update", "delete"]
- apiGroups: [""]
  resources: ["persistentvolumeclaims"]
  verbs: ["get", "list", "watch", "create", "update", "delete"]
- apiGroups: ["networking.k8s.io"]
  resources: ["networkpolicies"]
  verbs: ["get", "list", "watch", "create", "update", "delete"]

~
```

Figure IV.28 : Configuration du rôle pour le namespace mywebapp.

Une fois le Role créé, nous avons utilisé un objet RoleBinding pour lier le Role défini à l'utilisateur Massyl, lui permettant ainsi de bénéficier de ces nouvelles permissions.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: mywebapp-rolebinding
  namespace: mywebapp
subjects:
- kind: User
  name: Massyl
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: mywebapp-role
  apiGroup: rbac.authorization.k8s.io
```

Figure IV.29 : Configuration du RoleBinding.

Pour vérifier si les autorisations ont été correctement attribuées, nous avons utilisé la commande "kubectl auth can-i list deployments --namespace mywebapp" Cette commande interroge le serveur Kubernetes pour déterminer si l'utilisateur courant a les droits nécessaires pour effectuer l'action souhaitée "list" sur les ressources de type "deployments" dans le namespace mywebapp. La sortie de commande est "yes" ce qui indique que les permissions ont bien été accordées à Massyl.

Ensuite, nous avons testé la même commande sans spécifier de namespace, ce qui par défaut interroge le namespace "default". La sortie "no" confirme que l'utilisateur n'a pas les permissions nécessaires sur d'autres namespaces. Cela prouve que les règles sont spécifiques au namespace "mywebapp", garantissant ainsi la sécurité de notre cluster.

```
PS C:\Users\Ziinov> kubectl auth can-i list deployments --namespace mywebapp
yes
PS C:\Users\Ziinov> kubectl auth can-i list deployments
no
```

Figure IV.30 : Vérification des permissions pour l'utilisateur Massyl.

IV.6.1.4 Configuration du service account pour le dashboard Kubernetes

Le tableau de bord (Dashboard) de Kubernetes est une interface utilisateur Web qui permet de gérer et de visualiser les ressources d'un cluster Kubernetes de manière simple et conviviale. Pour

accéder au Dashboard, il est nécessaire de configurer un service account avec les permissions adéquates.

Pour cela, nous devons d'abord déployer le Dashboard Kubernetes. Ensuite, nous créons un service account dédié à ce Dashboard, nommé massyl-admin dans le namespace mywebapp, auquel nous attribuons des rôles spécifiques tels que list et get sur certains objets. Cela permet de consulter et de voir l'état de ces objets. Cependant, ce service account ne dispose pas de toutes les permissions nécessaires pour effectuer certaines actions, ce qui permet de réduire la surface d'attaque en cas d'exploitation du Dashboard.

Dans la figure IV.31, une tentative de suppression d'un service nommé nginx-service dans le namespace mywebapp, mais cette action n'a pas abouti. Le message d'erreur indique que l'action est interdite, soulignant les limitations des permissions attribuées.

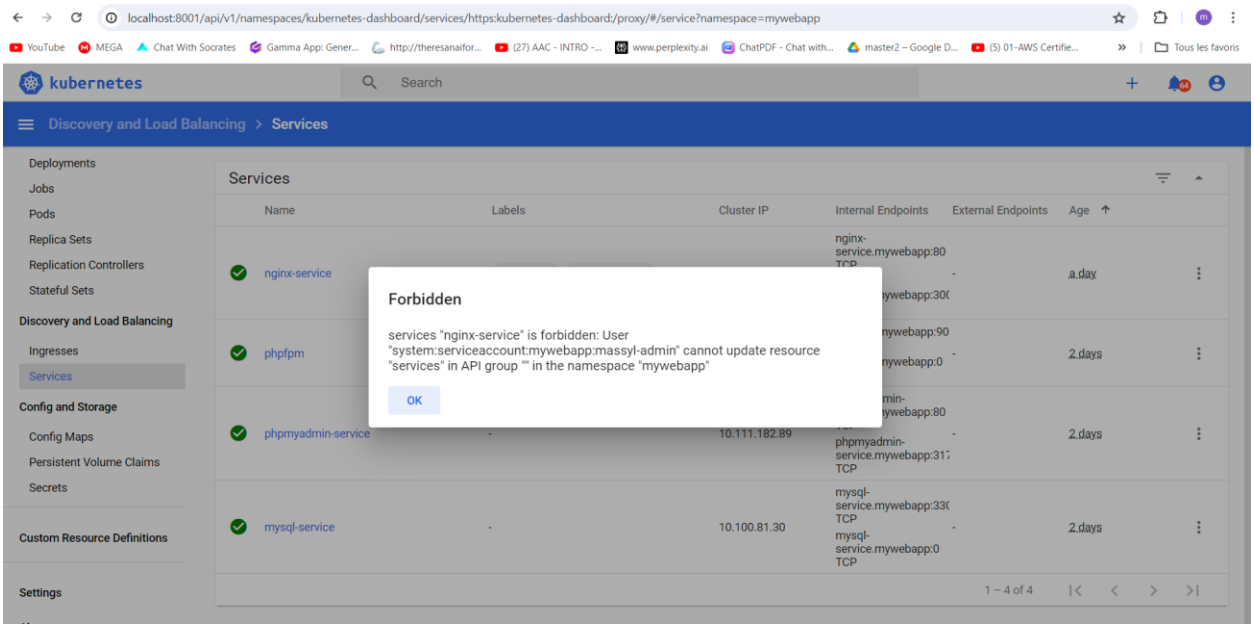

Figure IV.31 : Dashboard Kuberntes.

IV.6.1.5 Mise en place d'un contrôleur d'admission

Pour garantir la sécurité et la conformité de nos déploiements, il est impératif de s'assurer que toutes les images de conteneurs proviennent de notre registre de confiance où nous avons

précédemment poussé nos images Docker déjà construite. Cette mesure prévient l'utilisation d'images potentiellement vulnérables ou malveillantes.

Pour atteindre cet objectif, nous avons utilisé les contrôleurs d'admission Gatekeeper, basés sur le projet Open Policy Agent (OPA), qui permet de définir et appliquer des contraintes de sécurité. Où nous avons défini une règle écrite en Rego qui impose l'utilisation d'images provenant de notre registre.

```
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredImagePrefix
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8srequiredimageprefix

        violation[{"msg": msg, "details": {"image": image}}] {
          input.review.object.spec.containers[_].image
          image := input.review.object.spec.containers[_].image
          not startswith(image, prefix)
          msg := sprintf("L'image '%v' ne provient pas de notre registre de confiance /youbimassinissa .", [image])
        }

        prefix := input.parameters.prefixes[_]
```

Figure IV.32 : Règle de sécurité basé sur l'image.

Notre règle est ensuite encapsulée dans un modèle de contrainte, appliqué à l'aide d'une ressource CRD (Custom Resource Definition) de type K8sRequiredImagePrefix, qui spécifie que pour les Pods, Deployments et StatefulSets dans le namespace "mywebapp", seules les images commençant par le préfixe "youbimassinissa/" sont autorisées.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredImagePrefix
metadata:
  name: require-youbimassinissa-image
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
      - apiGroups: ["apps"]
        kinds: ["Deployment", "StatefulSet"]
    namespaces: ["mywebapp"]
  parameters:
    prefixes:
      - "youbimassinissa/"
```

Figure IV.33 : K8sRequiredImagePrefix.

Pour assurer l'application de notre règle, nous avons créé un simple Deployment utilisant l'image nginx. Lorsque nous tentons de le déployer, une erreur est générée (figure IV.33), indique que le

webhook d'admission a refusé la requête conformément à notre règle (require-youbimassinissa-image), car l'image 'nginx' ne provient pas du registre de confiance /youbimassinissa que nous avons spécifié. Cela confirme que notre règle est en place et fonctionne comme prévu.

```
PS C:\Users\Ziinov> kubectl apply -f nginx.yaml
Error from server (Forbidden): error when creating "nginx.yaml": admission webhook "validation.gatekeeper.sh" denied the
request: [require-youbimassinissa-image] L'image 'nginx' ne provient pas de notre registre de confiance /youbimassinissa
.
PS C:\Users\Ziinov>
```

Figure IV.34 : Résultat de contrainte.

IV.6.1.6 Mise en place de la politique réseau

Afin de garantir la séparation des pods dans notre espace de noms de travail, nous avons mis en place une politique réseau qui restreint la connexion entre les pods d'autres espaces de noms et ceux de l'espace de noms mywebapp.

Notre politique autorise uniquement la connexion avec le pod Nginx de notre espace de noms depuis l'espace de noms nginx-ingress-namespace. Cette décision est prise en vue de l'exposition ultérieure du service Nginx à l'extérieur du cluster à l'aide d'un Ingress Nginx.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: mywebapp
  name: allow-nginx-ingress
spec:
  podSelector:
    matchLabels:
      app: nginx
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          name: nginx-ingress-namespace
~
~
```

Figure IV.35 : Politique réseau de Namespace mywebapp.

En plus des mesures de sécurité déjà adoptées, nous avons mis en place des mécanismes supplémentaires pour garantir la sécurité de notre cluster et de nos déploiements. Cela inclut la

limitation de l'escalation des privilèges dans les déploiements, et la limitation des ressources pour contrôler l'utilisation des CPU et de la mémoire. Ces mesures renforcent la protection et assurent un environnement sécurisé pour nos applications. Maintenant que notre cluster est sécurisé, il est temps de passer aux déploiements.

IV.6.2 Déploiement de site web personnel sur Kubernetes

Pour le processus de déploiement de notre site web personnel sur notre cluster Kubernetes, nous avons conçu une architecture composée de quatre composants principaux : PHP-FPM, phpMyAdmin, Nginx, et MySQL. Chacun de ces composants utilise des images Docker personnalisées et sécurisées provenant de notre registre docker privé, garantissant ainsi des performances optimales et une sécurité renforcée.

La figure IV.36 illustre l'architecture complète de notre environnent de déploiement :

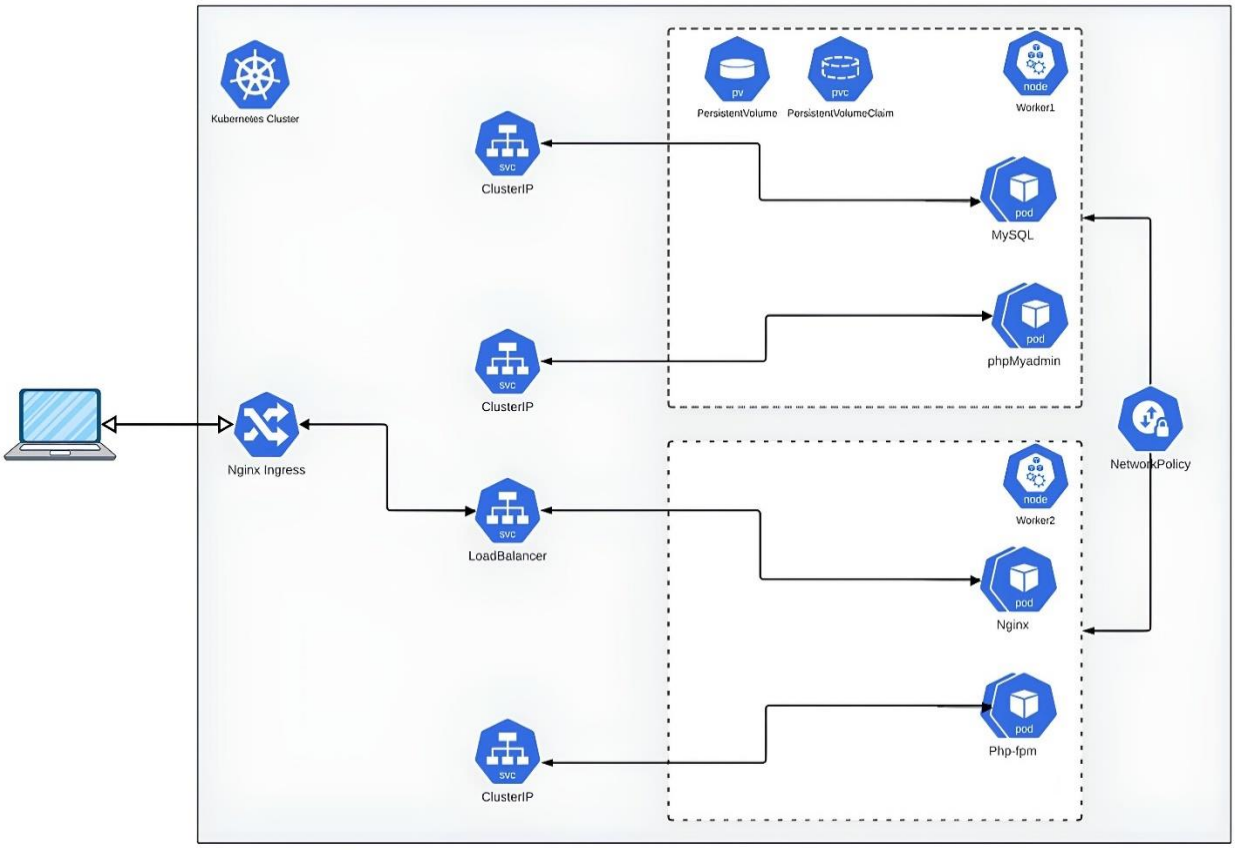

Figure IV.36 : l'architecture de note cluster kubernetes.

IV.6.2.1 Architecture Détaillée

a) Composants déployés et placement des pods

Les composants PHP-FPM, phpMyAdmin, et Nginx sont déployés sous forme de Deployments avec 2 réplicas chacun. MySQL est déployé en StatefulSet avec 2 réplicas. Des taints sont configurés pour que MySQL et phpMyAdmin soient déployés sur Worker1, tandis que Nginx et PHP-FPM sont déployés sur Worker2.

b) Exposition des services

Afin de garantir une gestion efficace des accès, les services MySQL et phpMyAdmin, PHP-FPM sont exposés en tant que ClusterIP, ce qui les rend accessibles uniquement au sein du cluster. En revanche, Nginx est exposé en tant que LoadBalancer et via Nginx Ingress pour gérer les requêtes HTTP/HTTPS externes.

c) Volumes persistants

Pour assurer la persistance des données de MySQL, nous utilisons des PersistentVolumes (PV) et des PersistentVolumeClaims (PVC).

IV.6.2.2 Lancement du site

Après avoir mis en place les mesures de protection pour sécuriser notre cluster, nous allons lancer notre site afin de vérifier la bonne exécution des conteneurs. L'URL du site est : " massyl92.io". La Figure IV.37 illustre le lancement réussi du site.

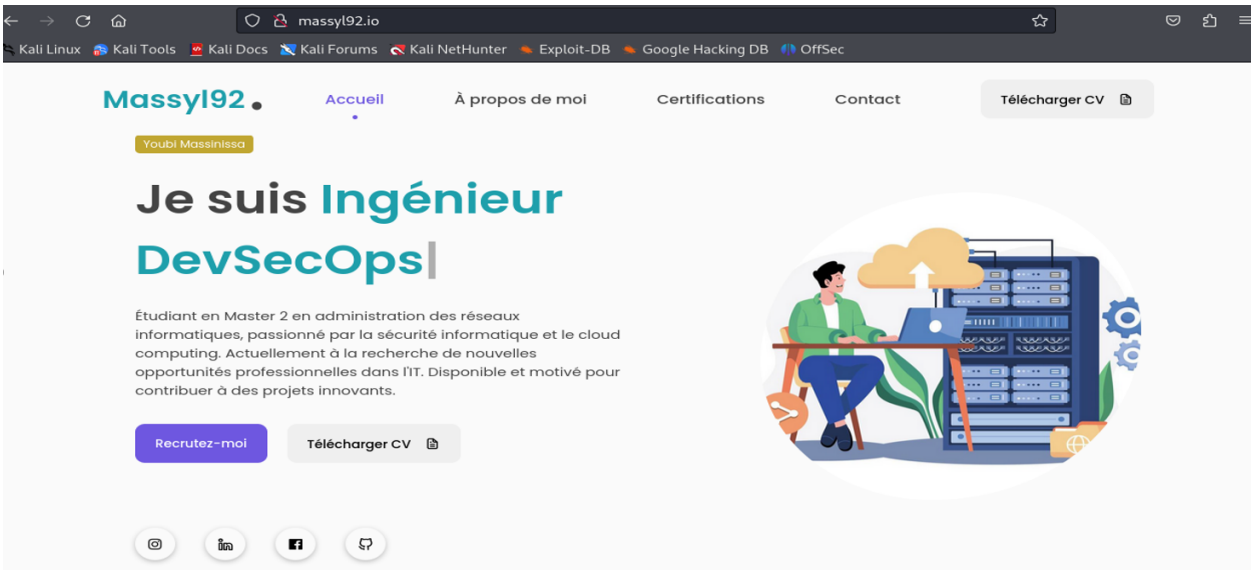

Figure IV.38 : Lancement réussi du site.

IV.7 Conclusion

Dans ce chapitre, nous avons abordé les solutions de sécurité pour les conteneurs avec Docker et l'orchestrateur Kubernetes. Pour illustrer ces pratiques, nous avons créé un site web, conteneurisé avec Docker et déployé à l'aide de Kubernetes.

Nous avons commencé par présenter notre propre contribution : un script conçu pour appliquer des règles d'audit spécifiques et enregistrer toutes interactions et activités internes de Docker. Ensuite, nous avons détaillé les différentes étapes nécessaires pour sécuriser notre environnement de travail et les conteneurs.

Une fois cette base de sécurité établie, nous avons migré vers Kubernetes, où nous avons intégré les principes de sécurité tout au long de ce processus, garantissant ainsi que l'orchestration des conteneurs se fasse dans un cluster sécurisé.

Même si nous avons couvert de nombreux aspects, garantir une sécurité globale pour une architecture Docker gérée par Kubernetes reste impossible. En effet, le risque zéro n'existe pas et la sûreté de notre architecture est également influencée par tous les éléments qui y sont connectés. Néanmoins, nous avons déployé tous nos efforts pour réduire ces risques dans le cadre de ce projet de fin d'études.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

La conteneurisation a révolutionné la virtualisation traditionnelle. Elle permet de réduire considérablement les ressources nécessaires pour exécuter des applications en partageant directement le noyau de la machine hôte tout en isolant les environnements avec des conteneurs. Cette approche répond à la demande de productivité accrue et de réduction des coûts d'infrastructure.

Cependant, l'adoption de la conteneurisation ne se fait pas sans défis, l'un des défis majeurs est celui de la sécurité. Ceci est devenue l'une des principales préoccupations des utilisateurs et frein l'adoption de cette nouvelle technologie.

L'objectif de ce travail est de proposer une solution permettant de renforcer la sécurité de l'architecteur en conteneur (d'une entreprise, d'un établissement ou autre) exécutée par Docker et orchestrée par kuberentes. Pour cela, nous avons analysé les divers enjeux de sécurité associés aux conteneurs et aux orchestrateurs, tout en explorant les stratégies visant à protéger la sécurité de ces environnements et à établir des pratiques de déploiement sûres. De plus, nous avons présenté notre solution, axée sur l'audit et la journalisation du moteur de conteneurisation Docker et de ses composants, ce qui permet d'analyser toutes les interactions et activités internes.

Les résultats obtenus ont clairement démontré que cette approche basée sur l'audit et la surveillance continue du moteur de conteneurisation offre des avantages significatifs en terme de détection précoce des menaces et de réponse rapide au incidents de sécurité. En minimisant les risques potentiels liées aux failles de sécurité des conteneurs, notre solution contribue à garantir l'intégrité des applications et la confidentialité des données sensibles.

Ce travail, nous a offert l'opportunité d'enrichir notre compréhension et d'approfondir nos connaissances des défis et des solutions associées aux conteneurs et à leur sécurité. Il nous a permis également d'explorer de nouvelles technologies et outils tel que Docker, kuberentes,Lynis, HashiCorp Vault,Trivy,Helm ce qui contribue à renforcer notre expertise professionnelle dans ce domaine en évolution.

Bien que nous ayons été désireux de sécuriser l'infrastructure conteneurisée d'une entreprise (ou n'importe quel autre établissement) en Algérie, malheureusement nous n'avons pas eu

l'opportunité de le faire. En effet, nous avons rencontré des difficultés à trouver des entreprises utilisant la technologie de conteneurisation.

En outre, notre engagement en matière de solutions ne s'arrête pas là. Nous avons l'intention d'ajouter une autre dimension à notre solution en capturant tous les appels système et les permissions nécessaires pour chaque conteneur. Ces données seront ensuite utilisées pour créer des profils AppArmor et Seccomp, renforçant ainsi la sécurité. Cette démarche est le résultat d'une recherche approfondie qui a révélé les difficultés rencontrées par les ingénieurs DevOps et DevSecOps pour mémoriser toutes les permissions et les appels système requis par chaque conteneur.

Bibliographie

[1] S. Bennett, «Server Virtualization Statistics 2024 – Everything You Need to Know,» webinarcare, 2024 juin 2024. [En ligne]. Disponible: https://webinarcare.com/best-server-virtualization-software/server-virtualization-statistics/. [Accès le 14 juin 2024].

[2] Cloud Native Computing Foundation, «CNCF Annual Survey Report 2023,» 2023.

[3] Sysdig, «Cloud-Native Security and Usage Report» 2023. [En ligne]. Disponible: https://sysdig.com/blog/2023-cloud-native-security-usage-report/. [Accès le 05 juin 2024].

[4] A. Krajnc, «Qu'est-ce que Docker,» 26 janvier 2024. [En ligne]. Disponible: https://www.jedha.co/formation-data/formation-docker. [Accès le 29 mars 2024].

[5] Docker , «Docker Engine 20.10 release notes,» 04 avril 2023. [En ligne]. Disponible: https://docs.docker.com/engine/release-notes/20.10/. [Accès le 03 juillet 2024].

[6] Kubernetes, «Kubernetes v1.29: Mandala,» 2023 Décembre 13. [En ligne]. Disponible: https://kubernetes.io/blog/2023/12/13/kubernetes-v1-29-release/. [Accès le 02 juillet 2024].

[7] Marie, «Virtualisation : Un bref historique et une vue d'ensemble,» 4 Avril 2023. [En ligne]. Disponible: https://commentouvrir.com/info/virtualisation-un-bref-historique-et-une-vue-densemble/. [Accès le 26 janvier 2024].

[8] Microsoft, «What is virtualization?,» 2019. [En ligne].Disponible: https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-virtualization/. [Accès le 27 janvier 2024].

[9] Syleo, «Qu'est-ce que la virtualisation open source ?,» [En ligne].Disponible: https://www.syloe.com/glossaire/virtualisation-open-source/. [Accès le 14 mars 2024].

[10] G. M. DIOUF, «UNE PLATEFORME D'ORCHESTRATION DE CONTENEURS DOCKER,» Université du Québec, Montréal, 2019.

[11] Fayyad, Hasan & LucPerneel, & Timmerman, Martin. (2013). Benchmarking the Performance of Microsoft Hyper-V server, VMware ESXi and Xen Hypervisors. Journal of Emerging Trends in Computing and Information Sciences. Vol. 4, No. 12 , December 2013, pp: 922-933, ISSN 2079-8407.

[12] Lenovo, «Qu'est-ce qu'une machine virtuelle Java (JVM) ?,» [En ligne]. Disponible: https://canada.lenovo.com/fr/ca/en/glossary/jvm/. [Accès le 30 Janvier 2024].

[13] Red hat, «La conteneurisation, qu'est-ce que c'est ?,» 3 août 2023. [En ligne]. Disponible: https://www.redhat.com/fr/topics/cloud-native-apps/what-is-containerization. [Accès le 12 février 2024].

[14] *A Brief History of Containers.* [Enregistrement audio]. Disponible: https://www.youtube.com/watch?v=NtCNAxq_jVw&t=4. 2021.

[15] Ahmed Lounici et Anis Messaoudi, «Deploying and Securing a High Availability Kubernetes Cluster,» Universite de Bejaia, 2022.

[16] Amzone, «Qu'est-ce que la conteneurisation ?,» [En ligne]. Disponible: https://aws.amazon.com/fr/what-is/containerization/. [Accès le 05 mai 2024].

[17] avinetworks, «container-orchestration,» [En ligne]. Disponible: https://avinetworks.com/glossary/container-orchestration/. [Accès le 19 mars 2024].

[18] M. Beschokov, «Container Orchestration?,» 26 fevrier 2024. [En ligne]. Disponible: https://www.wallarm.com/what/what-is-container-orchestration-7-benefits-and-4-best-tools. [Accès le 15 mars 2024].

[19] I. BUCHANAN, «Containers vs. virtual machines,» juin 2023. [En ligne]. Disponible: https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms. [Accès le 27 mars 2024].

[20] Amazon-AWS, «Quelle est la différence entre une architecture monolithique et une architecture de microservices ?,» [En ligne].Disponible : https://aws.amazon.com/fr/compare/the-difference-between-monolithic-and-microservices-architecture/. [Accès le 22 mars 2024].

[21] cloudacademy, «Advantages and Disadvantages of Microservices Architecture,» 17 juillet 2023. [En ligne].Disponible: https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/ [Accès le 29 mars 2024].

[22] «docker logo,» [En ligne]. Disponible: https://logowik.com/docker-vector-logo-2767.html. [Accès le 20 mai 2024].

[23] Docker, «Docker overview,» [En ligne]. Disponible: https://docs.docker.com/get-started/overview/#:~:text=Docker%20architecture,to%20a%20remote%20Docker%20daemon.. [Accès le 29 mars 2024].

[24] A. Kumar, «Docker Images: A Complete Guide For Beginners,» 18 janvier 2024. [En ligne]. Disponible: https://k21academy.com/docker-kubernetes/docker-image-and-layer-overview-for-beginners/. [Accès le 15 avril 2024].

[25] O. Birade, «Docker images and beyond,» 21 janvier 2020. [En ligne]. Disponible: https://medium.com/interleap/docker-images-and-beyond-b1a849900ec1. [Accès le 20 mai 2024].

[26] aquasec, «Docker Registry,» 15 Decembe 2020. [En ligne]. Disponible: https://www.aquasec.com/cloud-native-academy/docker-container/docker-registry/. [Accès le 15 avril 2024].

[27] A. Luzzardi et V. Victor, «Swarm : a Docker-native clustering system.,» [En ligne]. Disponible: https: //gi thub. com/docker/swarm/. [Accès le 15 avril 2024].

[28] A. Javed, «Container-based IoT Sensor Node on Raspberry Pi and the Kubernetes Cluster Framework,» août 2016. [En ligne]. Disponible: https://www.researchgate.net/figure/Architectural-diagram-of-Swarm_fig5_346921304. [Accès le 20 mai 2024].

[29] J. Walker, «Docker Networking – Basics, Network Types & Examples,» 12 mai 2023. [En ligne]. Disponible: https://spacelift.io/blog/docker-networking. [Accès le 24 avril 2024].

[30] «Docker Netwking,» 27 avril 2023. [En ligne]. Disponible: https://www.geeksforgeeks.org/basics-of-docker-networking/. [Accès le 20 mai 2024].

[31] Redhat, «Kubernetes, qu'est-ce que c'est ?,» 27 mars 2020. [En ligne]. Disponible: https://www.redhat.com/fr/topics/containers/what-is-kubernetes. [Accès le 01 mai 2024].

[32] B. Matador, «What is Kubernetes?,» juin 2022. [En ligne]. Disponible: https://www.bluematador.com/learn/kubernetes. [Accès le 27 mars 2024].

[33] «Kubernetes Architecture,» [En ligne]. Disponible: https://avinetworks.com/glossary/kubernetes-architecture/. [Accès le 01 mai 2024].

[34] «Architecture Kubernetes,» 17 Novembre 2020. [En ligne]. Disponible: https://www.aquasec.com/cloud-native-academy/kubernetes-101/kubernetes-architecture/. [Accès le 02 mai 2024].

[35] Vmware, «Qu’est-ce que l’architecture Kubernetes ?,» [En ligne]. Disponible: https://www.vmware.com/topics/glossary/content/kubernetes-architecture.html. [Accès le 02 mai 2024].

[36] «Objects In Kubernetes,» [En ligne]. Disponible: https://kubernetes.io/docs/concepts/overview/working-with-objects/. [Accès le 05 mai 2024].

[37] Kubernetes, «Aperçu du Pod,» [En ligne]. Disponible: https://kubernetes.io/fr/docs/concepts/. [Accès le 04 mai 2024].

[38] Redhat, «Un déploiement Kubernetes, qu'est-ce que c'est ?,» 23 Avril 2020. [En ligne]. Disponible: Un déploiement Kubernetes, qu'est-ce que c'est ?. [Accès le 04 Mai 2024].

[39] S. Tech, Réalisateur, *Kubernetes DaemonSet Explained in 2 Minutes | Beginner Friendly |.* [Film]. https://www.youtube.com/watch?v=49jgQADF638, 2023.

[40] S. Pattnaik, «Kubernetes Networking: The Complete Guide,» 06 avril 2024. [En ligne]. Disponible: https://www.linkedin.com/pulse/kubernetes-networking-beginners-guide-swadhin-pattnaik-rg3tc/. [Accès le 20 mai 2024].

[41] R. Jain, «Kubernetes Core Concepts - Services,» 25 août 2021. [En ligne]. Disponible: https://blog.learncodeonline.in/kubernetes-core-concepts-services. [Accès le 23 mai 2024].

[42] J. Walker, «Kubernetes Network Policy – Guide with Examples,» 18 Décembre 2023. [En ligne]. Disponible : https://spacelift.io/blog/kubernetes-network-policy. [Accès le 15 mai 2024].

[43] Kubernetes, «Persistent Volumes,» [En ligne]. Disponible: https://kubernetes.io/docs/concepts/storage/persistent-volumes/. [Accès le 19 mai 2024].

[44] Y. Perry, «Kubernetes StorageClass: Concepts and Common Operations,» 30 mars 2022. [En ligne]. [Accès le 19 mai 2024].

[45] RedHate, «La sécurité des conteneurs, qu'est-ce que c'est ?,» 13 avril 2023. [En ligne]. Disponible: https://www.redhat.com/fr/topics/security/container-security. [Accès le 05 mai 2024].

[46] Sari Sultan, Tassos Dimitriou et Imtiaz Ahmad, «Containers' Security: Issues,Challenges, and Road Ahead,» in *IEEE Access*, vol. 7, pp. 52976-52996, 2019, doi: 10.1109/ACCESS.2019.2911732.

[47] Tripwire, «Guide to Container Security – Everything You Need to Know,» 22 october 2019. [En ligne]. Disponible: https://www.tripwire.com/state-of-security/guide-container-security. [Accès le 07 mai 2024].

[48] owasp, «OWASP Docker Top 10,» [En ligne]. Disponible: owasp.org/www-project-docker-top-10/. [Accès le 14 mai 2024].

[49] CVE, «Docker : Product details, threats and statistics,» [En ligne]. Disponible: https://www.cvedetails.com/product/28125/Docker-Docker.html?vendor_id=13534. [Accès le 24 mai 2024].

[50] Fortinet, «Qu'est-ce qu'un vecteur d'attaque ?,» 2022. [En ligne]. Disponible: https://www.fortinet.com/fr/resources/cyberglossary/attack-vector. [Accès le 2021 juin 2024].

[51] Bélair, Maxime. (2021). Défense contre les attaques à l'aune des nouvelles formes de virtualisation des infrastructures. IMT Atlantique campus de Nantes.

[52] Doan, Phuc & Jung, Souhwan. (2022). DAVS: Dockerfile Analysis for Container Image Vulnerability Scanning. Computers, Materials & Continua. 72. 1699-1711. 10.32604/cmc.2022.025096.

[53] Wist, Katrine & Helsem, Malene & Gligoroski, Danilo. (2021). Vulnerability Analysis of 2500 Docker Hub Images. 10.1007/978-3-030-71017-0_22.

[54] D. Guides, «Container Attacks,» [En ligne]. Disponible: https://devsecopsguides.com/docs/attacks/container/. [Accès le 21 juin 2024].

[55] M. Paliwal et A. Anand, «Docker Logging - Types, Configuring Drivers, Logging Strategies,» 16 mai 2024. [En ligne]. Disponible: https://signoz.io/blog/docker-logging/. [Accès le 17 juin 2024].

[56] M. Thevarmannil, «10 Container Security Risks to look out for in 2024,» 25 janvier 2024. [En ligne]. Disponible: https://www.practical-devsecops.com/container-security-risks/. [Accès le 21 juin 2024].

[57] G. Georgieva, «Docker Security Advisory: Multiple Vulnerabilities in runc, BuildKit, and Moby,» 31 janvier 2024. [En ligne].Disponible : https://www.docker.com/blog/docker-security-advisory-multiple-vulnerabilities-in-runc-buildkit-and-moby [Accès le 22 juin 2024].

[58] Florian Wendland et Christian Banse, THREAT ANALYSIS OF CONTAINER-AS-A-SERVICE. Fraunhofer Institute for Applied and Integrated Security.

[59] Hua, Zhichao & Yu, Yang & Gu, Jinyu & Xia, Yubin & Chen, Haibo & Zang, Binyu. (2021). TZ-Container: protecting container from untrusted OS with ARM TrustZone. Science China Information Sciences. 64. 10.1007/s11432-019-2707-6.

[60] J. Jim Manico, «Docker Security Cheat Sheet,» [En ligne]. Disponible: https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html. [Accès le 20 mai 2024].

[61] A. Sheps, «Top 10 Container Security Best Practices,» 30 november 2023. [En ligne]. Disponible: https://www.aquasec.com/cloud-native-academy/container-security/container-security-best-practices/. [Accès le 22 mai 2024].

[62] J. Hale, «Top 20 Docker Security Tips,» 07 November 2019. [En ligne]. Disponible: https://resources.experfy.com/bigdata-cloud/top-20-docker-security-tips/. [Accès le 25 mai 2024].

[63] HackTricks, «Docker Security,» [En ligne]. Disponible: https://book.hacktricks.xyz/v/fr/linux-hardening/privilege-escalation/docker-security. [Accès le 25 mai 2024].

[64] C. P. K. J. S. Anton Chuvakin, Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management, Syngress 978-1597496353 , 2013.

[65] Rejoindre-plus-que-Pro, «Les avantages cruciaux de la surveillance des journaux d'événements pour une sécurité informatique inébranlable,» 12 avril 2024. [En ligne]. [Accès le 22 juin 2024].

[66] VirtualizationHowto, Compositeur, *Best Docker Container Monitoring Tools.* [Enregistrement audio]. https://www.youtube.com/watch?v=zxAmqY63eJE&t=13s. 2023.

[67] Docker, «View container logs,» [En ligne]. Disponible: https://docs.docker.com/config/containers/logging/. [Accès le 15 juin 2024].

[68] Redhat, «State of Kubernetes security report,» https://www.redhat.com/en/resources/state-kubernetes-security-report-2023, 2023.

[69] Threat matrix for Kubernetes, «Threat matrix for Kubernetes,» 02 avril 2020. [En ligne]. Disponible: https://www.microsoft.com/en-us/security/blog/2020/04/02/attack-matrix-kubernetes/. [Accès le 04 juin 2024].

[70] Redhat, «La sécurité de Kubernetes, qu'est-ce que c'est ?,» 18 septembre 2024. [En ligne]. Disponible : https://www.redhat.com/fr/topics/containers/kubernetes-security. [Accès le 02 juin 2024].

[71] Pai, Santosh & Kunte, Srinivasa. (2023). Secret Management in Managed Kubernetes Services. International Journal of Case Studies in Business, IT, and Education. 130-140. 10.47992/IJCSBE.2581.6942.0263.

[72] Á. R. Martínez, «Study of Security Issues in Kubernetes (K8s) Architectures; Tradeoffs and Opportunities,» uppsala university, 2023.

[73] D. GOODIN, «Tesla cloud resources are hacked to run cryptocurrency-mining malware,» 02 Février 2018. [En ligne].Disponible : https://arstechnica.com/information-technology/2018/02/tesla-cloud-resources-are-hacked-to-run-cryptocurrency-mining-malware/ [Accès le 15 juin 2024].

[74] Darwesh, Ghadeer & Hammoud, Jaafar & Vorobeva, A.A.. (2022). SECURITY IN KUBERNETES: BEST PRACTICES AND SECURITY ANALYSIS. 2. 63-69. 10.14529/secur220209.

[75] A. VEDPATHAK, «User Authentication And Authorization In Kubernetes,» 19 juillet 2020. [En ligne]. Disponible: https://www.linkedin.com/pulse/user-authentication-authorization-kubernetes-ajit-vedpathak. [Accès le 07 juin 2024].

[76] H. B. Lapointe, «Vers la Sécurité des Conteneurs - Les Comprendre et les Sécuriser,» Département d'Informatique et de Recherche Opérationnelle,, Université de Montréal, 2022.

[77] M. BÉLAIR, «Défense contre les attaques à l'aune des nouvelles formes de virtualisation des infrastructures,» L'ÉCOLE NATIONALE SUPÉRIEURE MINES-TÉLÉCOM ATLANTIQUE BRETAGNE, 2021.

[78] M. Manuel, «Sécurité des architectures en conteneurs Docker orchestrés par Kubernetes,» IESN henallux Namur, 2020.

[79] VMware, «VMware Workstation,» [En ligne]. Disponible: https://www.vmware.com/products/workstation-pro/faq.html. [Accès le 10 mai 2024].

[80] H. Inc, «Configurer et gérer un cluster Kubernetes avec Kubeadm,» 09 Aout 2023. [En ligne]. Disponible: https://hackernoon.com/fr/configurer-et-gerer-un-cluster-kubernetes-avec-kubeadm. [Accès le 22 juin 2024].

[81] Tigera, «About Calico,» [En ligne]. Disponible: https://docs.tigera.io/calico/latest/about/. [Accès le 25 mai 2024].

[82] «Lynis,» [En ligne]. Disponible: https://cisofy.com/lynis/. [Accès le 12 mai 2024].

[83] «CIS Benchmarks,» [En ligne]. Disponible: https://www.cisecurity.org/cis-benchmarks-overview. [Accès le 14 mai 2024].

[84] J. NYCKOWSKI, «What You Need to Know About Linux Auditing,» 30 Août 2022. [En ligne]. Disponible : https://goteleport.com/blog/linux-audit/ [Accès le 20 mai 2024].

[85] «Understanding Linux Audit,» [En ligne]. Disponible: https://documentation.suse.com/sles/12-SP5/html/SLES-all/cha-audit-comp.html. [Accès le 20 mai 2024].

[86] J. Benabra, «Trivy: A tool for identifying and mitigating vulnerabilities in docker images,» 15 Décembre 2020. [En ligne]. [Accès le 22 juin 2024].

[87] A. Reddy, «Evolution of Containers: Past, Present, and Future,» 25 novembre 2022. [En ligne]. Disponible: https://www.linkedin.com/pulse/evolution-containers-past-present-future-abhay-reddy/. [Accès le 15 mars 2024].

[88] redhat, «Qu'est-ce qu'un espace de noms ?,» [En ligne]. Disponible: https://access.redhat.com/documentation/fr-fr/red_hat_enterprise_linux/9/html/managing_monitoring_and_updating_the_kernel/what-namespaces-are_setting-limits-for-applications. [Accès le 16 mars 2024].

[89] S. v. Kalken, «What Are Namespaces and cgroups, and How Do They Work?,» 21 juillet 2021. [En ligne]. Disponible: https://www.nginx.com/blog/what-are-namespaces-cgroups-how-do-they-work. [Accès le 18 mars 2024].

[90] P. Inyang et T. Fernandez, «Docker Volumes: Efficient Data Management in Containerized Environments,» 05 décembre 2023. [En ligne]. Disponible: https://semaphoreci.com/blog/docker-volumes. [Accès le 01 mai 2024].

[91] Docker, «Bind mounts,» [En ligne]. Disponible: https://docs.docker.com/storage/bind-mounts/. [Accès le 22 mai 2024].

[92] Extio Technology, «Mastering Kubernetes Pod-to-Pod Communication: A Comprehensive Guide,» 16 juin 2023. [En ligne]. Disponible: https://medium.com/@extio/mastering-kubernetes-pod-to-pod-communication-a-comprehensive-guide-46832b30556b. [Accès le 12 mai 2024].

[93] «Container Networking,» 02 Août 2023. [En ligne]. Disponible: https://support.huaweicloud.com/intl/en-us/basics-cce/kubernetes_0023.html. [Accès le 12 mai 2024].

[94] IPI, «LA SÉCURITÉ INFORMATIQUE, QU'EST-CE QUE C'EST ?,» [En ligne]. Disponible : https://www.ipi-ecoles.com/securite-informatique [Accès le 05 mai 2024].

[95] SÉCURITÉ INFORMATIQUE, Mila: Université AbdelHafid Boussouf-Mila, 2019.

[96] Websitesecuritystore, «What is The CIA TRIAD & its Importance for Cybersecurity,» 2022 Aout 2022. [En ligne]. Disponible: https://websitesecuritystore.com/blog/what-is-the-cia-triad/. [Accès le 07 mai 2024].

[97] J. Marsal, «What is DevSecOps? And what you need to do it well,» 19 janvier 2023. [En ligne]. Disponible: https://www.dynatrace.com/news/blog/what-is-devsecops. [Accès le 08 mai 2024].

ANNEXE

Tables des matières

Annexe 1 : Historique et Concepts Clés des Conteneurs

1. Le résumé de l'évolution et les origines des conteneurs.

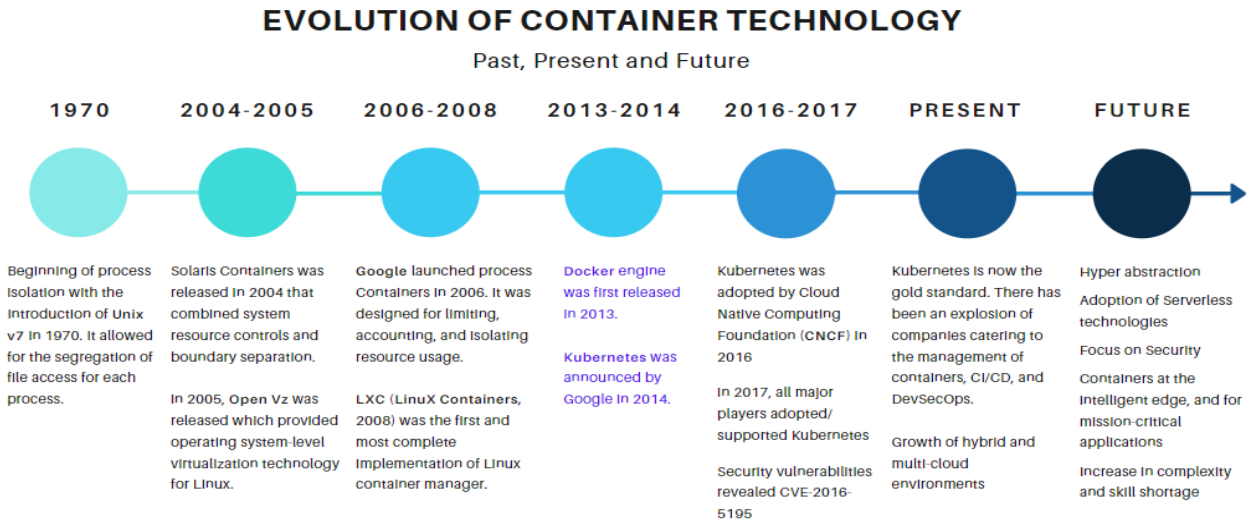

Figure 01 : Histoire des conteneurs. *[87]*

2.Les espaces de nom linux

Les espaces de noms sont une fonctionnalité du noyau linux les plus importantes pour organiser et identifier les objets logiciels. Elle permet d'encapsuler une ressource (par exemple un point de montage, un périphérique réseau ou un nom d'hôte) dans une abstraction qui donne aux processus de cet espace de noms l'impression qu'ils disposent de leur propre instance isolée de la ressource globale. Les conteneurs sont l'une des technologies les plus couramment associées à l'utilisation des espaces de noms. [88]

Le tableau suivant présente les espaces de noms pris en charge et les ressources qu'ils isolent :

Nom	**Ressource isolée**
Mount	Points de montage
UTS	Nom d'hôte et nom de domaine NIS
IPC	Queues de messages
PID	Identifiants de processus
NETWORK	Dispositifs de réseau, piles, ports, etc
USER	Identifiant utilisateur et groupe
Control groups	Répertoire racine du groupe de contrôle

Tableau 01 : Liste des namespaces.

Certaines namespaces (pid_namespace et user_namesapce) fonctionnent sous forme d'arbre. Ceci signifie que lorsqu'une nouvelle namespace est créée, elle est considérée comme fille de la namespace du processus à l'origine de sa création. L'objectif de tels arbres est de permettre aux namespaces parent de voir les namespaces filles mais pas l'inverse, permettant ainsi d'avoir une namespace fille plus isolée que la namespace parent.

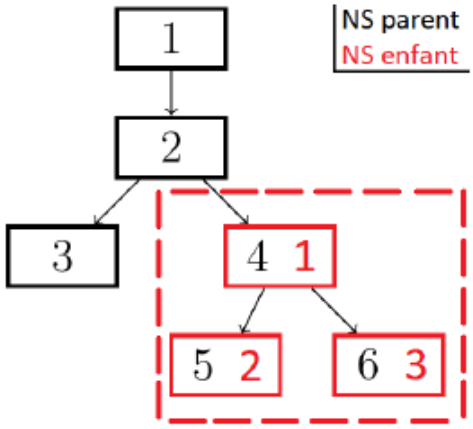

Figure 02 : Exemple d'arbre de processus et de namespace PID.

3.Groupe de Contrôle

Un groupe de contrôle (cgroup) est une fonctionnalité du noyau Linux qui limite, compte et isole l'utilisation des ressources (CPU, mémoire, E/S disque, réseau, etc.) d'un ensemble de processus. [89]

Les cgroups offrent les fonctionnalités suivantes :

- **Limites de ressources** : Vous pouvez configurer un cgroup pour limiter la quantité d'une ressource particulière (mémoire ou CPU, par exemple) qu'un processus peut utiliser.
- **Priorisation** : Vous pouvez contrôler la quantité d'une ressource (CPU, disque ou réseau) qu'un processus peut utiliser par rapport aux processus d'un autre cgroup en cas de conflit de ressources.
- **Comptabilité** : Les limites de ressources sont surveillées et rapportées au niveau du cgroup.
- **Contrôle** : Vous pouvez changer l'état (arrêté ou redémarré) de tous les processus dans un cgroup avec une seule commande.

Le schéma ci-dessous montre comment lorsque vous attribuez un pourcentage spécifique des ressources système disponibles à un cgroupe (ici cgroup-1), le pourcentage restant est alloué à d'autres cgroups (et processus individuels) sur le système. [89]

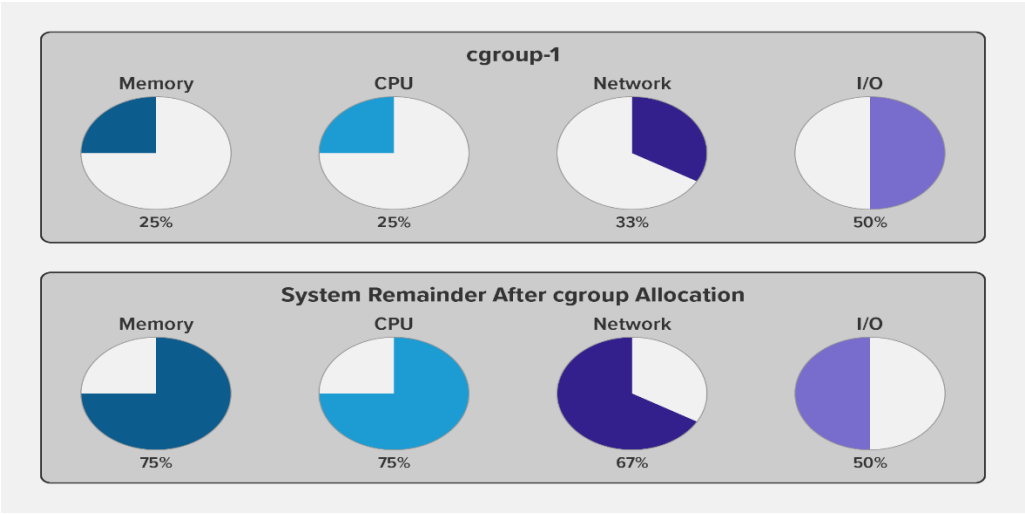

Figure 03 : Allocation des ressources système en utilisant Cgroups.

4.Les Capabilités Linux

Selon le manuel page linux, pour effectuer des vérifications de permission, les implémentations UNIX traditionnelles distinguent deux catégories de processus : les processus privilégiés (dont l'identifiant d'utilisateur effectif est 0, désigné comme superutilisateur ou root) et les processus non privilégiés (dont l'UID effectif est différent de zéro). Les processus privilégiés contournent toutes les vérifications de permission du noyau, tandis que les processus non privilégiés sont soumis à une vérification complète des permissions basée sur les informations d'identification du processus, Ce qui diminue les risques en évitant d'accorder des privilèges excessifs.

Dans le domaine de la conteneurisation, il est fréquent d'observer des conteneurs nécessitant des autorisations spécifiques pour assurer leur bon fonctionnement., les conteneurs sont faits pour héberger des microservices, cette liste de droits supplémentaires est donc généralement très limitée. Les capabilités sont donc une solution permettant de réduire les privilèges au minimum en n’autorisant que le strict nécessaire au fonctionnement correct du conteneur.

Annexe 2 : Gestion du stockage dans Docker et Dockerfile

1.Docker volumes

Les volumes Docker sont une fonctionnalité de Docker qui permet de stocker et de gérer de manière persistante des données dans des conteneurs. Un volume est un répertoire ou un emplacement de stockage nommé en dehors du système de fichiers du conteneur et accessible à un ou plusieurs conteneurs. Il permet de partager et de conserver les données même lorsque les conteneurs sont arrêtés, démarrés ou supprimés. [90]

2.Montage de répertoire hôtes (Bind Mounting)

Le bind mounting est une fonctionnalité qui permet de monter un répertoire ou un fichier depuis le système hôte directement dans un conteneur Docker. Cela permet au conteneur d'accéder et de manipuler les fichiers ou répertoires du système hôte, comme s'ils étaient directement présents à l'intérieur du conteneur lui-même. Contrairement aux volumes Docker, les processus non-Docker peuvent accéder et modifier ces fichiers à tout moment.

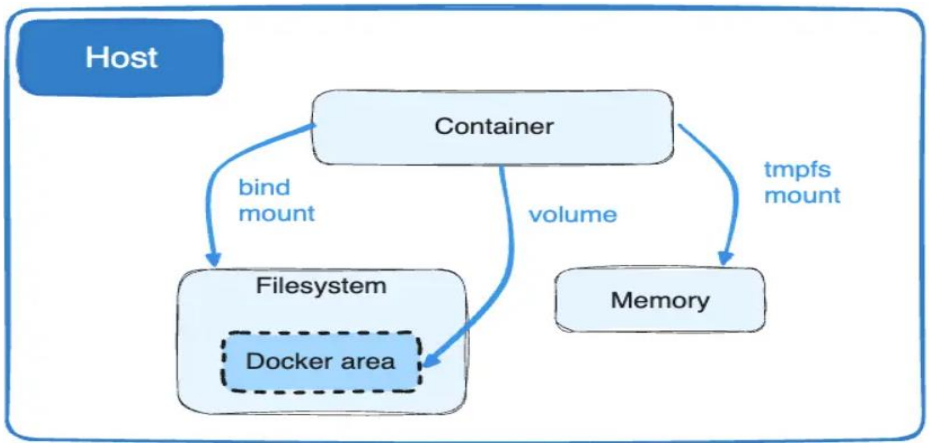

Figure 04 : Le stockage avec Docker *[91]*

3.Dockerfile

Dockerfile est un fichier texte écrit dans une syntaxe facile à comprendre qui décrit la création d'un conteneur. Il contient les instructions pour créer une image Docker. Chacune de ces instructions crée un calque qui rend ces images légères, petites et rapides à modifier. Ces instructions sont placées dans un fichier YAML que l'on renomme Dockerfile. Parmes les instructions principales qu'il contient on peut citer :

- **FROM:** défini l'image de base sur laquelle les instructions suivantes seront exécutées.

- **LABEL**: ajoute des métadonnées telles que le nom de l'auteur du Dockerfile.
- **ARG:** variables temporaires utilisables dans le Dockerfile.
- **ENV:** variables d'environnements utilisables dans le Dockerfile et le conteneur.
- **RUN:** permet d'exécuter une ou plusieurs commandes dans le conteneur pour par exemple installer une application.
- **CMD:** spécifie les commandes à exécuter lors du démarrage de l'image et non de sa construction, donc différent de RUN.
- **EXPOSE:** ouvre un port réseau durant l'exécution de l'image dans un conteneur.
- **COPY:** copie des dossiers et fichiers d'une source vers une destination dans le système de fichiers du conteneur.
- **ADD:** ressemble à COPY, mais prend en charge en plus la présence d'une adresse URL en tant que source d'un élément à copier.
- **ENTRYPOINT:** exécute une commande principale au démarrage du conteneur.
- **VOLUME:** une fois un conteneur supprimé, les données qu'il contient le seront Aussi. Cette instruction permet de définir le répertoire qui sera partagé entre le Conteneur et l'hôte afin de persister les données.
- **USER:** Indique le nom d'utilisateur à utiliser pour lancer les instructions RUN, CMD et ENTRYPOINT, par défaut c'est l'utilisateur root.
- **WORKDIR:** permet la modification du répertoire courant qui sera utilisé pour lancer les commandes CMD et/ou ENTRYPOINT, ainsi que le répertoire du démarrage du conteneur.
- **ONBUILD:** ajoute les instructions à exécuter lorsque l'image sert de modèle pour la construction d'image enfant. Cette commande fonctionne comme si RUN était inséré directement après FROM dans le Dockerfile de la nouvelle image. Cela permet de construire régulièrement une nouvelle image contenant des valeurs qui changent de façon dynamique.

Annexe 3 : Mécanisme de communication inter-pod dans Kubernetes

1.La communication entre pods sur le même nœud

Lorsqu'un pod communique avec des systèmes externes, il utilise des dispositifs veth (virtual Ethernet devices). Ces dispositifs veth agissent comme des tunnels entre différents espaces de noms réseau. Les pods sur le même nœud se connectent via ces veth au bridge qui joue le rôle d'un routeur à l'intérieure de nœud, et qui permet aussi l'attribution des adresses IP CIDR dynamiquement aux pods. [92]

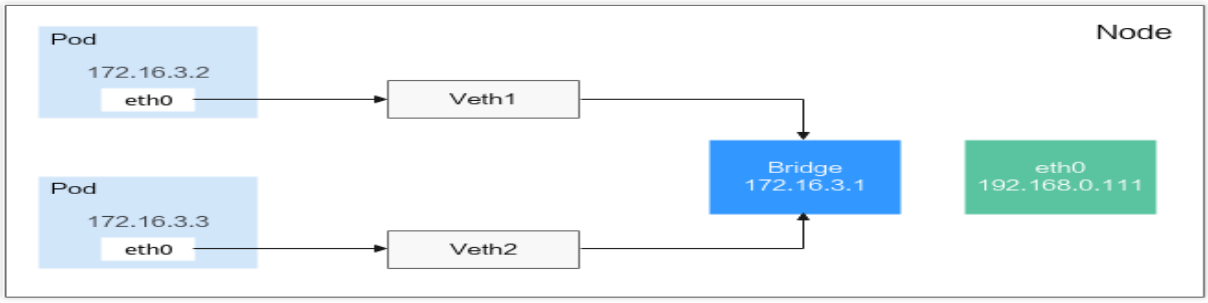

Figure 05 : Exemple d'une communication entre pods sur le même nœud *[93]*

2.La communication entre pods sur des différents nœuds

Dans un cluster Kubernetes, chaque pod doit avoir une adresse IP unique qui est leur moyenne de connexion. Pour ce faire, chaque nœud du cluster se voit attribuer un sous-réseau distinct, permettant ainsi d'assurer l'unicité des adresses IP des pods. Selon la configuration réseau sous-jacente, cette communication peut se faire en mode réseau superposé (overlay network), de routage ou en utilisant le réseau sous-jacent (underlay network). Ce processus de gestion de la connectivité réseau est implémenté à l'aide des plugins CNI qui sont responsables de l'attribution des adresses IP, de la création de réseaux virtuels et de la gestion du trafic entre les pods. [93]

- **Réseau superposé (overlay network) :** Un réseau superposé est construit au-dessus du réseau physique des nœuds. Il utilise l'encapsulation de tunnel pour créer un réseau virtuel avec son propre espace d'adressage IP.
- **Réseau de routage :** Dans un réseau de routage, une table de routage est utilisée en conjonction avec le réseau physique sous-jacent pour gérer la communication entre les pods et les nœuds.

- **Réseau sous-jacent :** Dans un réseau sous-jacent, les interfaces réseau physiques des nœuds sont directement exposées aux pods , les VLAN sont couramment utilisés pour segmenter le réseau sous-jacent et garantir la séparation du trafic.

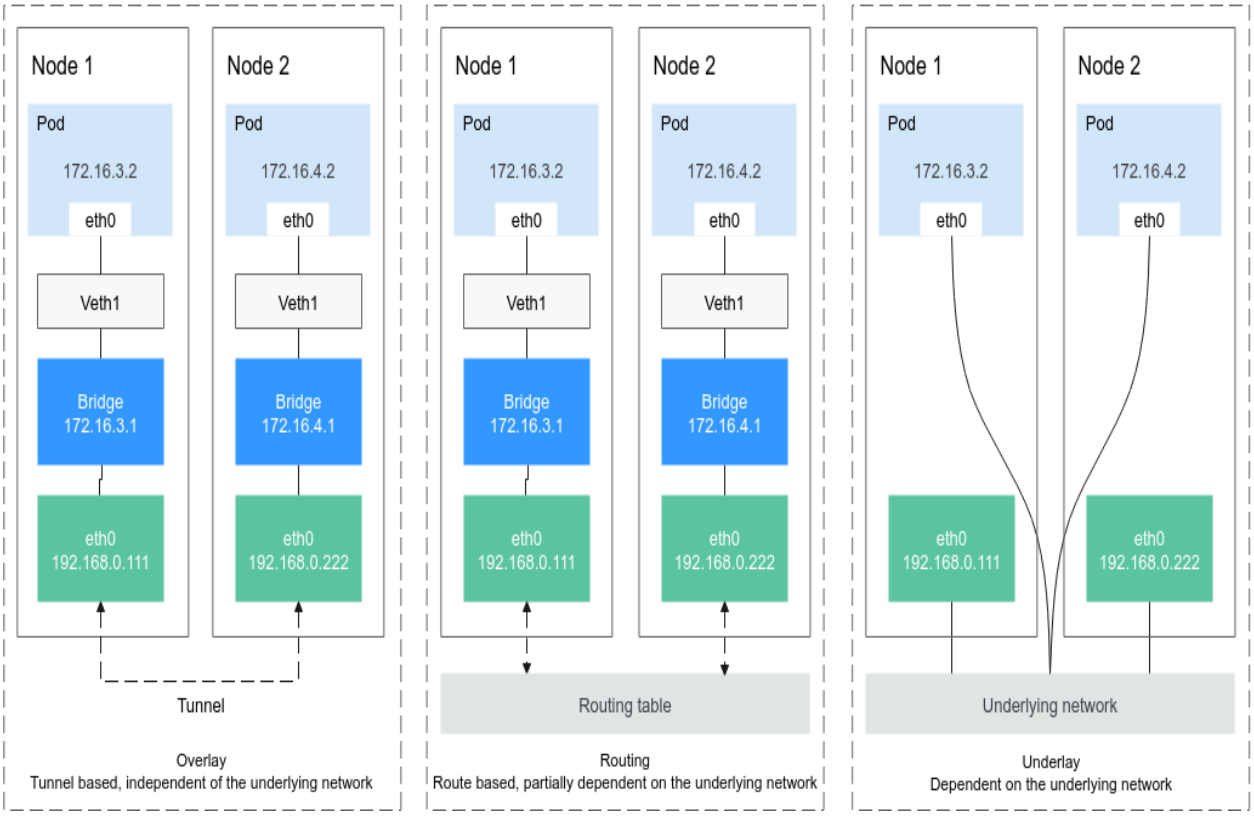

Figure 06 : Exemple d'une communication entre pods sur des nœuds différents. *[93]*

Annexe 4 : Sécurité informatique et DevSecOps

1.Sécurité informatique

La sécurité informatique est l'ensemble des moyens techniques, organisationnels, juridiques et humains pour protéger l'intégrité et la confidentialité des informations stockées dans un système informatique. [94]

La sécurité informatique concerne deux domaines [95] :

- « Sûreté = Safety (en anglais) » : protection de systèmes informatiques contre les accidents dus à l'environnement et les défauts du système.
- « Sécurité = Security (en anglais) » : protection des systèmes informatiques contre des actions malveillantes intentionnelles.

2.La triade CIA

Les trois lettres de la « triade CIA » représentant les trois piliers essentiels de la sécurité informatique : Confidentialité, Intégrité et Disponibilité. La méthode CIA correspond à un modèle commun qui constitue la base du développement des systèmes de sécurité fiable et efficaces. Elle est utilisée pour identifier les vulnérabilités et les méthodes de création de solutions. [96]

Dans son ouvrage "The Basics of Information Security", Jason Andress décrit les trois principes fondamentaux de la sécurité informatique comme suit :

- **Confidentialité** : se réfère à la capacité de protéger les données contre ceux qui ne sont pas autorisés à les consulter.
- **Intégrité :** désigne la capacité de prévenir toute modification non autorisée ou indésirable de nos données.
- **Disponibilité :** La disponibilité fait référence à notre capacité à accéder à nos données lorsque nous en avons besoin.

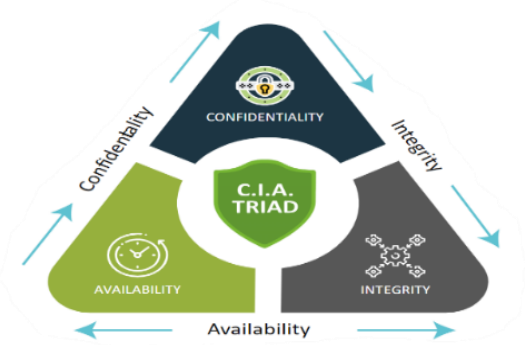

Figure 07 : la triade CIA. [96]

3. Le DevSecOps

DevSecOps est l'abréviation de "Développement, Sécurité et Opérations". DevSecOps est perçu comme une évolution du la méthodologie de DevOps, en allant au-delà en mettant particulièrement l'accent sur l'importance de la sécurité à chaque étape du cycle de vie du développement logiciel [97]. Cela permet de garantir que les considérations de sécurité sont prises en compte dès la conception et tout au long du déploiement des logiciels, réduisant ainsi les vulnérabilités potentielles et renforçant la sécurité des système

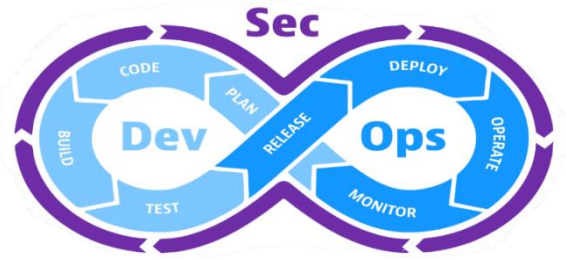

Figure 08 : DevSecOps. [97]

Annexe 5 : Initialisation de cluster Kubernetes

Pour créer notre cluster, nous choisissons d'utiliser kubeadm, un outil qui simplifie considérablement le processus de déploiement d'un cluster Kubernetes.

Le processus de configuration de notre cluster se divise en plusieurs étapes :

1. Initialisation du Plan de Contrôle

La première étape consiste à initialiser le plan de contrôle sur le nœud maître. Cette initialisation configure les composants principaux de Kubernetes, tels que l'API server, le scheduler et le controller manager.

Figure 09 : Initialisation de master.

Une fois cette initialisation terminée, un token est généré qui sera utilisé ultérieurement par les workers pour se connecter au cluster (figure 10).

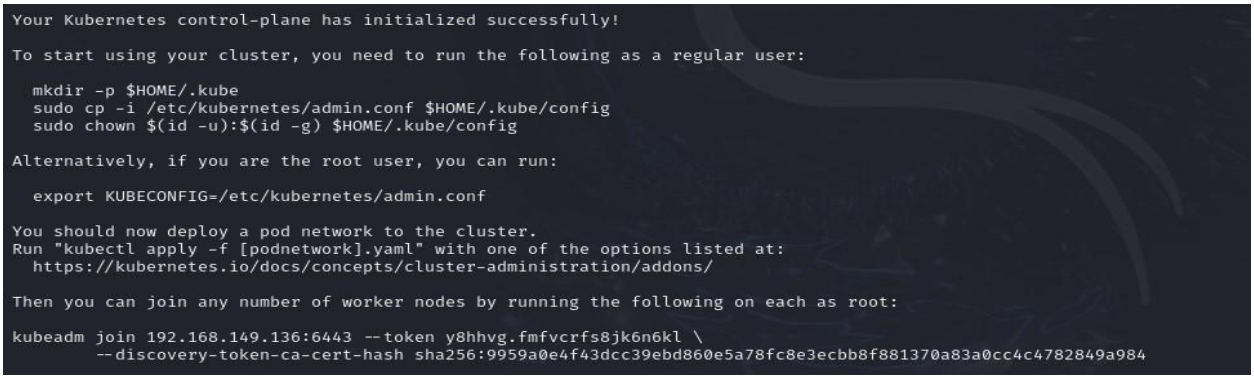

Figure 10 : Cluster Token.

2. Déploiement d'un plugin réseau

Ensuite, il est important de déployer un réseau de pods pour assurer la communication entre les différents composants et nœuds du cluster. C'est pour cela que nous avons choisi Calico, qui est une solution de mise en réseau et de sécurité dans un cluster Kubernetes.

3. Ajout des nœuds worker au cluster

Après l'installation du plugin Calico, nous avons ajouté nos deux nœuds workers au cluster en utilisant la commande kubeadm join, générée lors de l'initialisation du plan de contrôle. Cette commande inclut un token d'authentification et un hash de certificat pour sécuriser la connexion des nouveaux nœuds au cluster.

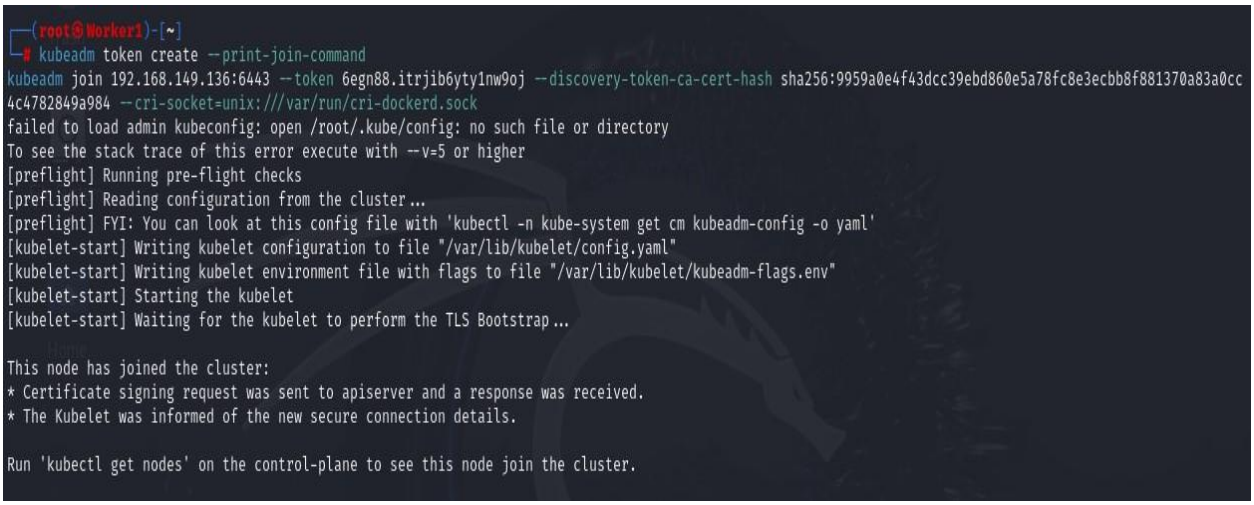

Figure 11 : L'ajout des nœuds.

4. Vérification de l'État des Nœuds

Une fois les nœuds worker ajoutés, il est recommandé de vérifier leur état pour s'assurer qu'ils ont bien rejoint le cluster et qu'ils sont prêts à exécuter des charges de travail. Tous les nœuds ont le statut "Ready" ce que nous a permis de nous assurer que le cluster est correctement configuré et que tous les nœuds peuvent communiquer entre eux.

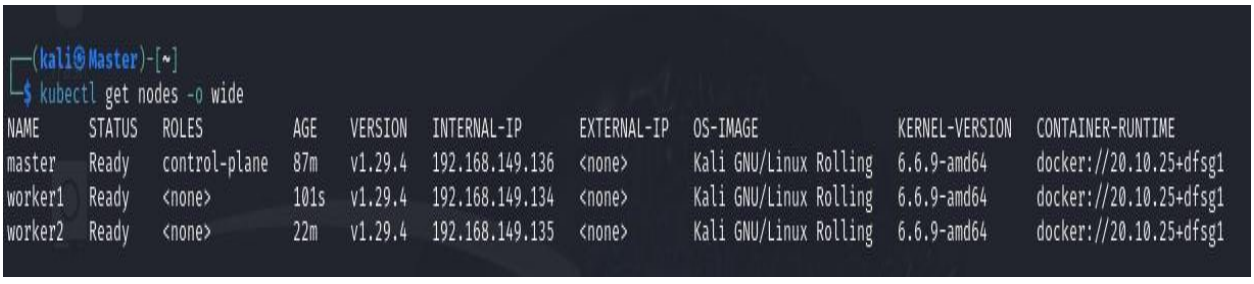

Figure 12 : L'état de cluster.