Pré-mémoire

de M. David GOMES

effectué dans la société Cirrusware à Mérignac

Tuteur : M. Vincent Jaulin

Optimisation de l'infrastructure cloud

Master 1 Expert développement mobile et IoT - Ynov

Année 2018/2019

**Table des Matières**

[**Lexique**](#_covvxvrunl8l) **2**

[**Introduction**](#_33yrqmuyxsp1) **3**

[**Virtualisation**](#_5t9cyxla9wua) **5**

[Principe](#_nspn90g3h9gm) 5

[Autres Types de virtualisation](#_genk5h5cpn9) 7

[Types d’hyperviseurs](#_fel7z7l66xqc) 8

[Conséquences](#_s5s1bvko619q) 8

[**Conteneurisation**](#_x39vxs9z1gon) **11**

[Conteneurisation et Docker](#_eujew262tde9) 11

[Machine virtuelle vs Conteneur](#_pklu5pgqe803) 12

[Docker](#_hxzzv5r84atw) 14

[Test comparatif de performance entre les machines virtuelles et les conteneurs](#_okucbqnd5fp8) 16

[Orchestration et Kubernetes](#_qtcr8etozwdc) 18

[Cluster](#_d97zeehvctw7) 19

[Pods](#_95f83k96ucbt) 20

[Service](#_olnxui5jmsz3) 21

[Volumes](#_48pnqxva2kbk) 21

[Pipeline CI/CD](#_qh2epsckw323) 21

[Intégration continue](#_ad900sw22sll) 22

[Distribution continue](#_sjkumh7r40jw) 22

[Déploiement continu](#_vkl63h2rkp0i) 23

[Avantages et inconvénient](#_6oerwgdi1af0) 23

[**Conclusion**](#_2yicfebg2ra9) **25**

[**Bibliographie**](#_1npm3af6hxt9) **26**

# 

# Lexique

API : une interface de programmation applicative (souvent désignée par le terme API pour application programming interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de point d’accès par lequel un logiciel offre des services à des applicatifs externe.

Cluster : Au sein d’un système informatique, un cluster de serveurs est un groupe de serveurs et d’autres ressources indépendantes fonctionnant comme un seul système. Les serveurs sont généralement situés à proximité les uns des autres, et sont interconnectés par un réseau dédié. Ainsi, les clusters permettent de profiter d’une ressource de traitement de données centralisée. Un client dialogue avec le groupe de serveurs comme s’il s’agissait d’une seule machine.

# Introduction

L’informatique est pour moi un monde de “tous les possibles” avec un grand esprit de partage de connaissances, de logiciels (open source). En effet, tout est possible, avec un simple ordinateur et beaucoup d’envie des entreprises qui ont commencé dans des garages sont à présent de grandes puissances aujourd’hui.

Je suis actuellement en contrat de professionnalisation dans l’entreprise Cirrusware à Mérignac. J’ai souhaité travailler dans une TPE (sept personnes) pour avoir des projets avec des responsabilités. Alors que dans une grande structure cela aurait été difficile. La bonne surprise c’est que chez Cirrusware l’ambiance y est conviviale et productive.

Cette société est spécialisée dans le développement web, particulièrement au travers du framework Symfony. Ils l’utilisent surtout comme support de développement pour la plateforme Send-Up, leur solution de communication digitale mais aussi pour d’autres projets clients. Dès l’entretien d’embauche, le contact avec les deux associés a été professionnel et convivial. Ils m’ont fait comprendre que pour eux l’aspect humain était aussi important que la dimension technique.

Cela fait à présent un peu plus d’un an que j’y travaille en tant que développeur web, j’ai notamment été intégré au développement de leur produit phare : Send-Up. J’y aborde plusieurs dimensions :

* l’évolutions de l’outil (nouvelles fonctionnalités, ergonomie, …)
* optimisation (factorisation de code)
* maintenance (débogage, prise en compte des problèmes clients)

De plus, j’ai eu l’occasion de réaliser des applications métiers pour le compte de JouéClub et Océalia.

Enfin, la société étant en pleine expansion, j’ai participé à des entretiens d’embauches avec mon tuteur en tant que consultant technique, ainsi qu’à l'élaboration de tests de compétences.

La société Cirrusware gère de plus en plus d’applications et sollicite d’autant plus les infrastructures cloud. Cette évolution demande un besoin d’organisation croissant et de réaliser des arbitrages permanents sur les infrastructures (choix des serveurs, mise en place de services, outils de monitoring,...).

La virtualisation et la “conteneurisation” sont des concepts qui sont au coeur de la stratégie de l’administration d’un environnement cloud applicatif. Je vous propose de voir en quoi ces concepts améliorent la livraison continue de contenus applicatifs.

Nous allons aborder ces approches technologiques, dans un premier temps la virtualisation qui permet de simuler un environnement serveur. Puis nous verrons la conteneurisation des applications permettant de simplifier le déploiement applicatif.

# Virtualisation

Le concept de virtualisation date des années 1960, initié par International Business Machines Corporation (IBM) qui a déployé des efforts considérables pour essayer de développer une technologie efficace de partage du temps CPU pour leurs serveurs. Avant la fin des années 90, la société de technologie VMware a développé les premiers produits de virtualisation capables de virtualiser l’architecture x86[[1]](#footnote-0). Avec la virtualisation x86, le matériel et le système d'exploitation ont été divisés en deux par une couche d'abstraction. La couche d'abstraction permet aux systèmes d'exploitation et aux applications virtualisées s'exécutant sur la machine physique de devenir indépendante du matériel, ce qui augmente l'agilité et la continuité des activités. Il n'est plus nécessaire d'arrêter les services pour la maintenance du matériel ou les sauvegardes, car les applications peuvent facilement être migrées ou copiées vers d'autres machines physiques.

Autrefois, les serveurs en entreprise étaient sous utilisés. Ils étaient mono-tâches avec des systèmes d’exploitation spécifiques pour des questions de sécurité et de fiabilité du service hébergé. Les processeurs étaient mono-coeur et peu puissant, en conséquence ils ne pouvaient pas traiter beaucoup de tâches simultanément. Mais avec la montée en puissance des processeurs, l’arrivée du multi-coeur et du multi threading ont nettement augmenté leur capacité de calcul.

Je vais dans un premier temps vous exposer son principe de fonctionnement, puis les progrès et les conséquences que cela a engendré.

## Principe

La virtualisation serveur permet de créer et d’exécuter une ou plusieurs machines en simultané sur une même machine physique. Chaque ordinateur simulé (que l’on appelle communément machine virtuelle ou VM) possède son propre OS et est étanche des autres instances virtuelles.

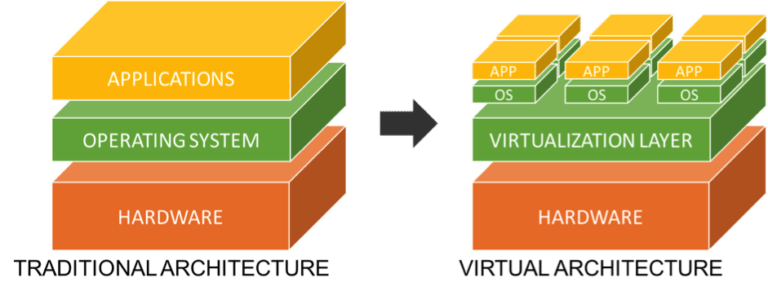


FIGURE 1. Représentation d’une virtualisation serveur

Un serveur virtualisé et mutualisé se détaille comme ci-après:

* Matériel: c’est la couche physique (CPU, mémoire RAM, disque dur, réseaux), c’est cette partie que l’on souhaite exploiter au maximum dans le but de réduire les coûts.
* Couche de virtualisation (Virtualization layer ou Hyperviseur) : Son rôle est de s’occuper de la communication et de la répartition des ressources matérielles entre les VM ainsi que la couche matérielle. (CF détail de fonctionnement page 7, paragraphe [Autre types de virtualisation](#_genk5h5cpn9))
* Machines virtuelles: dispose d’un OS dédié, des drivers nécessaires pour exploiter le matériel simulé et enfin les applications et/ou services.

La virtualisation doit couvrir quatre points clés :

* Partitionnement : pouvoir gérer plusieurs environnements virtualisés sur une machine physique tout en pilotant les ressources (CPU, RAM, stockage)
* Isolation : isolez la gestion des pannes et de la sécurité au niveau matériel, tout en garantissant des performances élevées par le biais de contrôles avancées des ressources
* Encapsulation : permettre d’enregistrer un état de la machine virtuelle dans des fichiers afin d’en faciliter son déplacement et sa copie
* Indépendance vis-à-vis du matériel : permettre de dupliquer les VM et/ou les migrer sur n’importe quel serveur

### Autres Types de virtualisation

#### Réseaux :

Consiste à rassembler des ressources réseaux physiques et logicielles en une seule entité. Cette entité est plus facilement administrable et fait abstraction du matériel et de sa complexité éventuelle. Cela permet de monitorer plus facilement le réseau, de partager les ressources entre les différents serveurs ou postes de travail. À titre d’exemple, ceci permet de faire communiquer des machines entre elles sur des réseaux distincts ou encore d’en isoler par mesure de sécurité.

#### Stockage :

Permet de regrouper plusieurs disques interconnectés afin d’en former un seul. Ainsi, occulter la complexité de la gestion d’un parc de stockage (réplication des disques durs, sécurité…). Ce qui permet de faciliter la maintenance, le remplacement de disques durs défectueux de façon transparente pour l’utilisateur.

#### Poste de travail :

Les postes de travail des employés d’une entreprise peuvent tous être hébergés sur des serveurs. L’ordinateur de ces employés (souvent très minimaliste dans ce cas) ne fait office que de relais entre l’utilisateur et l’environnement de travail hébergé sur le serveur. Ainsi, l’employé peut s’y connecter depuis n’importe où pour accéder à son environnement de travail. L’investissement de la puissance machine est investi sur les serveurs qui virtualisent l’environnement de travail et la maintenance des ordinateurs de bureau est fortement réduite. D’autre part, ce contexte permet une meilleure maîtrise de la sécurité informatique.

### Types d’hyperviseurs

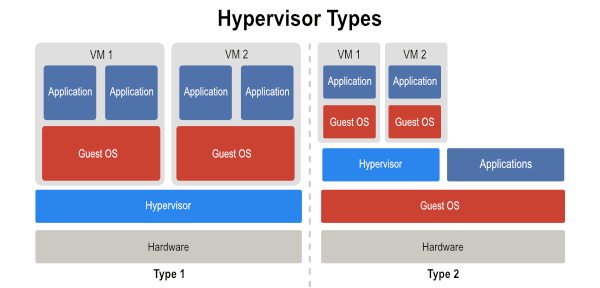


FIGURE 2. Types d’hyperviseurs

#### Type 1 ou bare-metal

Ce type d'hyperviseur s'exécute directement sur le matériel sans système d'exploitation. Comme nous le voyons à la figure 2, chaque système d'exploitation invité s'exécute de manière indépendante dans la machine virtuelle, et l'hyperviseur joue le rôle de maintenir l'interaction entre le matériel physique et les machines virtuelles. C’est l’option la plus choisi en entreprise car la moins gourmande en ressource. Xen, VMware ESX, KVM sont des exemples de ce type d’hyperviseur.

#### Type 2 ou hosted

Ce type d’hyperviseur s’exécute sur un système d’exploitation hôte et va se servir des fonctions de celui-ci pour interagir avec le matériel. Dans la figure 2, chaque VM s'exécute via l’hyperviseur et elles sont vues par le système hôte comme de simples applications. C’est très utilisé pour le développement d’application ou la sécurité informatique. VirtualBox et VMware Workstation sont les logiciels les plus populaires utilisant ce type d’hyperviseur.

## Conséquences

Cette technologie à complètement changer le paysage des système d’information en entreprise.

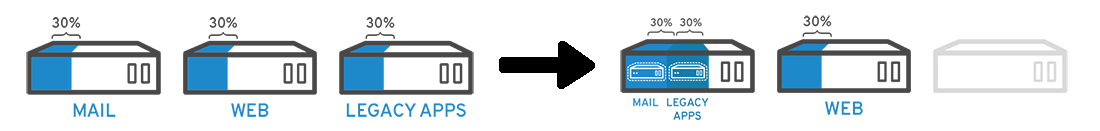


FIGURE 3. Avant et après la virtualisation

Dans la figure 3, une entreprise avec 3 serveurs, chacun dédié à un usage. Dans ce cas, on constate que les serveurs sont sous-exploités (30% de leurs capacité) car il y a beaucoup de temps de non sollicitation entre les demandes de traitements. Dans un soucis d’optimisation, le défis était de regrouper ces différents services pour réduire le nombre de machines physiques peu sollicitées sous un seul serveur tout en gardant un aspect mono-tâches pour chaque environnement virtualisé. Grâce à cette stratégie, la virtualisation a permis d’optimiser l’usage des serveurs physiques.

Sans cette technologie, lorsqu’une entreprise avait besoin d’un serveur plus puissant pour répondre à ses besoins d’évolution et sa demande en ressource système, elle devait acheter un serveur spécifique avec potentiellement des erreurs d’estimation des besoins en ressources (sous-dimensionné ou sur-dimensionné). Avec la virtualisation serveur, les entreprises vont plutôt prendre de très gros serveurs et les découper en machines virtuelles qui elles sont flexibles sur les ressources que l’on souhaite allouer.

Sur les aspects administration, chaque machine virtuelle étant séparée des autres il n’y a aucun risque que des applications installées sur des VM distinctes ne puissent s'interférer.

A titre d’exemple, voici quelques avantages :

* Sécurité : si une application ou un serveur virtualisé comporte une faille ou se retrouve infecté par programme malveillant, seule cette VM sera affectée
* Librairies et dépendances : une application ancienne nécessitant de vieilles librairies pour fonctionner pourrait empêcher une autre application de bénéficier des dernières mises à jour

Nous constatons les bénéfices suivant :

* Optimisation de la ressource machine physique
  + Créer des environnements où l’on maîtrise CPU, RAM et disques pour chaque VM
  + Pilotage des ressources en fonction du besoin immédiat
* Amélioration de l’administration et de la sécurité
  + Environnement étanche
* Maintenance des machines plus rationnel en réduisant le nombre de machines

La virtualisation des serveurs ne cesse de s’améliorer, à tel point qu’il est aujourd’hui possible d’opérer des migrations de VM à chaud, c’est à dire d’en transférer d’un serveur physique trop sollicité ou en cas de panne vers un autre serveur, sans stopper l’exploitation. Cela permet de garantir des performances constantes ainsi qu’une très haute disponibilité.

Cette technologie a aussi apporté des changement dans le développement des applications. Elle a faciliter les tests sur différents environnements, systèmes d’exploitation, version de navigateurs sans même devoir changer de poste de travail. Elle a aussi permis la naissance de l’architecture micro-service, une multitude de plus petites applications inter-connectées plus facilement maintenable au lieu d’une application “monolithique”.

Un point souvent oublié mais pourtant très utilisé par les professionnels de la sécurité informatique. Ils utilisent les machines virtuelles afin d’analyser des virus, malware, …, ainsi :

* l’ordinateur du professionnel n’est pas mis en péril
* on évite la propagation de ces programmes malveillant sur le réseau
* facilite l’étude de ces programmes grâce au système de sauvegarde et de remise en état des VM (appelé snapshot)

Sans la virtualisation le Cloud computing n’aurait pu exister. Il consiste à la livraison de ressources informatiques sous la forme d’un service. Il apporte une flexibilité et un paiement à l’utilisation, comme si on achetait de l’électricité ou de l’eau par exemple.

La virtualisation facilite grandement la maintenance des serveurs et l’exploitation des applications malheureusement certaines sont très difficiles à gérer et fastidieuse. La virtualisation peut donc engendrer diverses complications sur le serveur. Notamment dans le cas d’applications ayant des exigences en temps réel ou quasi réel car dans certains systèmes de virtualisation, l'horloge système peut-être temporairement en retard de 5 à 10 secondes lorsque les machines virtuelles sont soumises à une charge importante. Généralement ce n’est pas un gros problème mais dans le cas d’une application en temps réel elle ne pourrait pas fonctionner dû à ces retards. C'est pourquoi la virtualisation n'est pas recommandée dans certains cas. Aussi étant donné que de nombreuses machines virtuelles s'exécutent sur un seul matériel physique, et que différents systèmes d’exploitation ou plusieurs instances du même système d’exploitation s’exécutent, cela va se traduire par une utilisation élevée du processeur, des besoins en mémoire importants et de nombreuses opérations de lecture et d’écriture sur le disque dur. De ce fait, cela peut éventuellement conduire à un goulot d'étranglement (appelé communément bottleneck). Une autre raison du goulot d'étranglement est également due au fait que de nombreuses applications des VM peuvent solliciter la même ressource au même moment. De nombreux logiciels ne sont pas pris en charge sur la virtualisation ou inversement. Bien qu’aujourd’hui la gestion de la virtualisation s’est simplifiée, l’administration des système virtualisés est réalisée par des personnes ayant une bonne connaissances de la virtualisation. Un manque de compétence ou de connaissance peut mener à une surconsommation des ressources voir de rendre inopérante des applications ou services.

Bien que la virtualisation soit maintenant bien ancré au sein des systèmes d’information d’entreprise et dans le Cloud, depuis 5 ans une nouvelle technologie de déploiement d’application se démocratise : la conteneurisation. Vient-elle concurrencer la virtualisation ou au contraire la compléter ?

# Conteneurisation

La conteneurisation est une virtualisation au niveau du système d'exploitation utilisée pour assurer l'isolation et la gestion des ressources d’une application, principalement dans les environnements Linux. Le nom de conteneurisation provient des conteneurs d’expédition standardisés et universels, dans le contexte des applications cela désigne une manière agile de conditionner des applications dans un environnement d'exécution isolé et indépendant assurant son interopérabilité, peu importe son environnement.

“ Un conteneur est une unité standard de logiciel qui regroupe le code et toutes ses dépendances afin que l'application s'exécute rapidement et de manière fiable depuis un environnement informatique à un autre. Une image de conteneur Docker est un package logiciel léger, autonome et exécutable, qui inclut tout ce qui est nécessaire à l'exécution d'une application: code, exécution, outils système, bibliothèques système et paramètres.” (www.docker.com)

Nous allons voir dans un premier temps les principes de la conteneurisation et Docker, le logiciel qui l’a démocratisé, puis nous verrons l’orchestration des conteneurs qui vient compléter cette technologie pour la rend encore plus compétitive. Enfin, nous verrons les pipelines CI/CD (Intégration Continue et Déploiement Continue) et comment ils améliorent les cycles de développement et d’exploitation d’une application.

## Conteneurisation et Docker

La conteneurisation, est une méthode permettant d’exécuter une application dans un environnement virtuel au sein d’une zone appelée Conteneur. Dans un conteneur, les ressources matérielles (systèmes de fichiers, réseau, processeur, mémoire vive, …) sont virtualisé. Dans cet environnement sont encapsuler l’application ainsi que toutes ces dépendances : fichiers, bibliothèques, etc. Les conteneur se connecte au noyau (kernel) d’un système d’exploitation par le biais d’une couche de virtualisation semblable à l’hyperviseur pour les machines virtuelles, dans le cas de Docker c’est le Docker Engine qui va occuper ce rôle avec les conteneurs. Grâce à l’utilisation du noyau du système d’exploitation hôte il n’est donc pas nécessaire comme pour les machines virtuelle, d’installer un nouveau système d’exploitation. Ici le noyau gèrent les ressources de l’ordinateur et permet au différents composants matériels et logiciels de communiquer entre eux. La conteneurisation rend donc le déplacement d’application virtuelle plus simple entre des systèmes d’exploitation pouvant exécuter des conteneurs et demande moins de ressources de mémoire, de RAM, de CPU, etc.

### Machine virtuelle vs Conteneur

Lorsque l'on compare la technologies de conteneurisation à des machines virtuelles, il existe plusieurs avantages. Les conteneurs, libérés de la charge d'un OS sont nettement plus légers, plus rapides à configurer et à déployer. Tandis que les machines virtuelles sont plus volumineuses et lentes à configurer donc à déployer, elles demandent aussi un certains temps de démarrage notamment à cause de l’OS embarqué. Les conteneurs étant très flexibles, cette technologie est un excellent outil pour les processus de développement et de mise en production que l'on peut combiner avec des pipelines CI / CD (Intégration continue et Déploiement continue). De cette façon, on obtient une application évolutive, rapidement installable sur un environnement souhaité, de la tester et de la transférer sur les environnement de production.

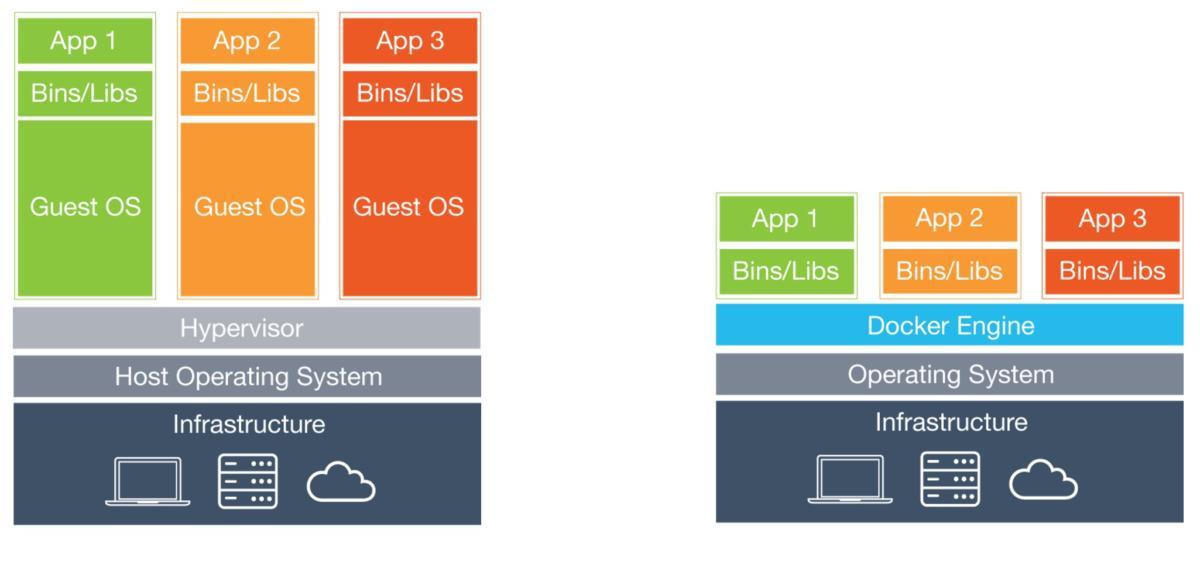


FIGURE 4. Comparaison des structures entre une architecture par machine virtuelle et par conteneur.

N’embarquant que le strict minimum, un conteneur exige aussi moins de mémoire permettant aux serveurs d’en héberger potentiellement beaucoup plus, de l’ordre de 10 à 100 fois par rapport à une VM (cf figure 5). Ceci encourage encore plus le développement d’application utilisant une architecture micro-service. Chaque conteneur est lancé dans un processus isolé, ce qui le rend indépendant mais avec une possibilité de communiquer avec d’autres conteneurs. Lancer un conteneur peut être assimilé à lancer une simple application de bureau classique.

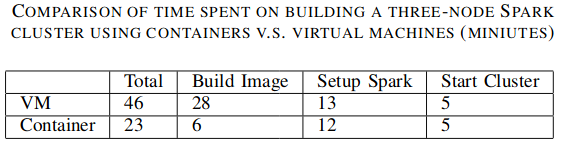


FIGURE 5. Comparaison du temps passé entre la construction et le déploiement de trois clusters chacun composé de trois VM ou conteneur Docker (en minutes).

On peut remarquer que les conteneurs Docker mettent deux fois moins de temps pour être entièrement démarré, en effet chaque VM doit copier et installer les fichiers de l’image tandis que les containers d’un cluster peuvent partager une même image une fois créé comme en atteste les résultats de “Build image”.

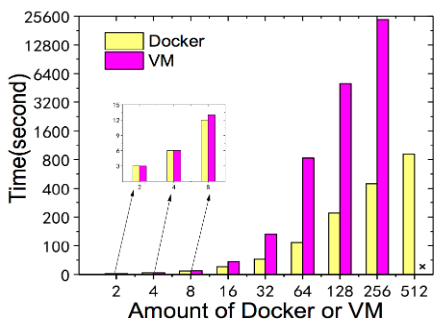


FIGURE 6. Temps nécessaire au démarrage en fonction du nombre de VM ou conteneur (la croix signifie l'absence de données à cause d’un échec de démarrage)

Pour un faible nombre d’instances les temps de démarrage sont très similaires mais nous pouvons remarquer que très vite que Docker devance largement les VM, pour 256 instances Docker démarre 50 fois plus vite. Pour 512 instances, les conteneurs mettent chacun moins de deux secondes pour démarrer.

Lors du lancement d’un conteneur il est aussi possible d’y connecter des “volumes”, c’est à dire partager un dossier de la machine hôte avec le conteneur. Dans un environnement de production par exemple, c’est utilisé afin de préserver les fichiers de logs que l’application va produire au fur et à mesure de son exploitation. En effet, tous les fichiers créés et modifiés au sein du conteneur pendant son exécution sont perdus lors de l’arrêt de celui-ci car ils ne font pas partie de l’image utilisé. Ce procédé et aussi très utilisé dans un environnement de développement, notamment dans le cas d’un serveur web qui sera lancé dans un conteneur et pour éviter de recréer une image à chaque test du serveur. Le dossier du projet web va être connecté au conteneur dans un répertoire spécifique afin que le développeur et le serveur ait accès aux fichiers en simultané. Le montage de volume permet aussi le partage de fichier entre différents conteneurs (figure 7), là encore plusieurs cas d’usage sont possibles, pour le replica de serveur ou encore un serveur de production qui va produire des fichiers de logs dans ce volume partagé et un autre conteneur qui va les traiter pour des outils monitoring.

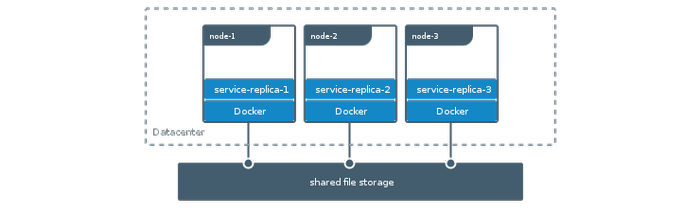


FIGURE 7. Représentation d’un partage de volume

Bien que ces deux technologies sont concurrentes, elles sont souvent combiner pour accroître la sécurité. Les conteneurs pourrait parfaitement être exécuté en bare-metal comme les machines virtuelles, mais l’isolation qu’ils procurent n’est pas aussi forte que celles des machines virtuelles notamment à cause du partage du noyau. Pour ce faire, des VM sont installé sur les serveurs physique, et c’est dans ces VM que seront exécutés les conteneurs. Ainsi, si un conteneur comporte un faille permettant à une personne mal intentionné d’accéder à la machine hôte, cela affectera seulement la VM et ses conteneurs au lieu de l'entièreté du serveur physique donc de réduire le champ d’action de l’assaillant.

### Docker

Docker est un logiciel open source développé initialement par deux français en 2015, il a réussi à démocratiser la conteneurisation d’application. Un conteneur se base sur une image afin de se créer. Une image comprend tous les fichiers sans les processus au moment de sa création, avec les logiciels qui ont été décidés d’y être installés (serveur apache, nodejs, mysql, Java, …), un conteneur est donc l’exécution d’une image.

Docker était initialement un logiciel exclusif à Linux car il utilisait des fonctionnalités seulement présente sur ce dernier (container Linux LXC). Mais depuis quelques années, Docker a fait abstraction de Linux et possède à présent des versions pour Mac OS, Windows et même Windows Server. L’idée première des conteneurs, de déployer un environnement de développement uniforme entre les développeurs ou une application en production peut importe le système d’exploitation de la machine hôte et sa configuration, est pleinement satisfaite.

L’un des grands atouts de Docker, est Docker Registry. Une application permettant et facilitant la distribution d’image Docker. C’est une bibliothèque d’image privé à installer sur ses propres serveurs ou en utilisant celle fournit par son fournisseur cloud (Amazon Elastic Container Registry, Google Cloud Container Registry, Azure Container Registry par exemple), on peut aussi utiliser Docker Hub, le Docker Registry officiel et public de Docker. C’est dans ce dernier que l’on va retrouver la plupart des images, tel que des images Ubuntu, de serveur Apache ou encore NodeJS, sur lesquels construire les images des applications conteneurisé à déployer.

Docker Registry est un composant majeur dans l’écosystème Docker, car il va permettre :

* Aux développeurs de distribuer des images prêtes à l’emploi des applications et de les versionner avec un système de tags.
* Aux nouveaux arrivants dans une société de mettre en place rapidement un environnement de développement à partir de ces images.
* À des outils d’intégration en continu de lancer des tests unitaire et d’intégration grâce à Docker
* À des systèmes automatisés de déployer les applications sur les serveurs de développement, de tests, de recette et de production.

Malgré l’encapsulation et l’isolation que Docker procure à ses conteneurs, il souffre d’une certaine réputation d’un manque de sécurité, voir déconseillé dans un environnement de production par les développeurs de Docker, bien heureusement ce dernier point était d’actualité il y a quelques années mais plus aujourd’hui.

Les failles de sécurité possibles sont :

* toutes les images se basant sur une autre pour se construire, si une image utilisé directement ou indirectement pour construire notre conteneur possède une faille de sécurité ou est infecté par un logiciel malveillant, notre conteneur se retrouve exposé. C’est pour cette raison qu’il est primordial d’utiliser des images provenant de source de confiance.
* le Docker daemon (le service qui exécute et contrôle les conteneurs Docker) doit posséder les droits administrateur de la machine pour fonctionner. Si une personne mal intentionné en prend le contrôle, il pourrait tout bonnement mettre hors-service le serveur voir la totalité du cluster auquel il appartient.

Le principal concurrent de Docker : Rocket (rkt) développé par CoreOs joue de cette réputation de Docker pour en faire un argument et vante la sécurité de ses conteneurs avec aussi une système de certification et de chiffrement des images. Le coeur de Docker a aussi été confié à la Cloud Native Computing Foundation (CNCF) en mars 2017 qui regroupe tous les grands groupes du milieu informatique, ce qui a permis à Docker de bénéficier de l’aides de ces derniers dans le but de combler ses principales lacunes ainsi que des axes d’amélioration ou nouveautés nécessaire pour ces acteurs qui pour la plupart l’utilisent.

### Test comparatif de performance entre les machines virtuelles et les conteneurs

Je vais vous présenter quelques résultats de deux études qui ont eu pour but de comparer les performances de solution de virtualisation et native.

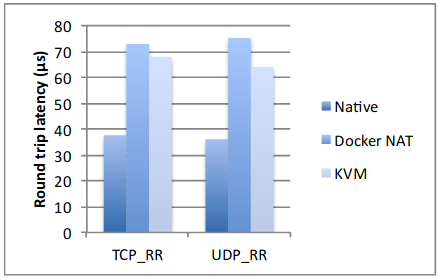


FIGURE 8. Test de latence réseaux

Lors des tests, Docker et KVM[[2]](#footnote-1) ont montré des temps de latences deux fois supérieur comparé au test sur le serveur physique directement, la gestion réseaux est tout de même légèrement meilleur sur KVM que Docker.

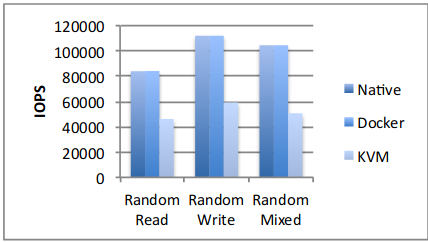
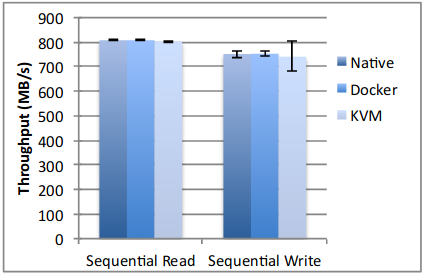


FIGURE 9 et 10. Test de performance sur disque dur

Pour des opérations de lecture et d’écriture sur disque dur, les tests directement sur serveurs physique et via Docker montre des performances en tout point similaire voir négligeable ce qui n’est pas le cas avec une machine virtuelle qui dans le cas d’une utilisation normal (figure 10) va effectuer deux fois moins d’opération.

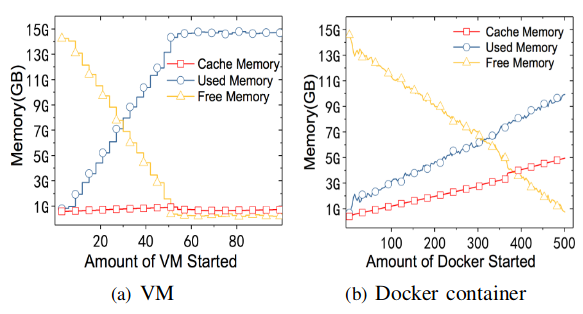


FIGURE 11. Utilisation de la mémoire RAM en fonction du nombre d’instances

Les VM nécessitant une quantité de mémoire RAM fixe au démarrage, elles vont très vite occupées toute la mémoire disponible de la machine physique même si à l’intérieur elles ne l’utilisent pas entièrement cette quantité leur est déjà alloué. Tandis que les conteneurs étant géré comme de simples applications leurs besoin en ressource équivaut seulement à ce qu’ils ont réellement besoin.

Docker est une technologie très efficace pour la gestion de conteneurs uniques. Cependant, à mesure qu'augmente le nombre de conteneurs et d'applications conteneurisées, la gestion et leur orchestration se complexifient. Regrouper plusieurs conteneurs pour assurer la distribution des services (réseau, sécurité, télémétrie, etc.) vers tous vos conteneurs. C'est précisément à ce niveau qu'intervient la technologie Kubernetes.

## Orchestration et Kubernetes

Kubernetes est une plateforme open source qui a pour but d’automatiser le déploiement, la mise à l'échelle et l’exploitation de conteneur. Il est le principal moteur d’orchestration de conteneurs dans le domaine de la conteneurisation. Développé par Google, il utilise notamment les images Docker comme base pour déployer des applications dans les conteneurs, mais prend aussi en charge depuis peu des images autre que Docker tel que rkt (Rocket développé par CoreOs). Avec Kubernetes, les conteneurs peuvent facilement être agrandis, détruits et refaits. Par rapport aux machines virtuelles normales, elles sont déployées plus rapidement, plus efficacement et de manière fiable. Dans un monde continuellement en croissance, les applications et les services doivent être déployés rapidement et efficacement. C’est là que Kubernetes entre en scène.

Les fournisseurs Cloud fournissent une multitude de services clé en main tel que des environnements de développement, le déploiement et l'exécution des applications. Le fournisseur assure une grande responsabilité sur l’installation, la configuration des environnements mais bien souvent ces solutions sont propriétaires ou fortement liées à la plateforme Cloud utilisée ce qui à pour conséquence de créer une dépendance envers ce fournisseur. C'est ce que fait Kubernetes, à la différence que le cluster[[3]](#footnote-2) est géré par le développeur ou l'administrateur du cluster et que cela peut être fait localement. Ces fonctionnalités incluent, par exemple, le déploiement, la mise à l'échelle, l'équilibrage de charge, la journalisation et la surveillance. Kubernetes comprend un ensemble de processus de contrôle indépendants qui contrôlent l'état actuel du déploiement en fonction du résultat souhaité. Kubernetes est devenu très vite populaire et a été intégré dans la plupart des fournisseur Cloud.

Docker a aussi son propre orchestrateur de conteneur, Docker Swarm, mais possèdent des attributs tel que l’évolutivité, la portabilité et le redéploiement automatique en cas de crash ou de panne serveur. Kubernetes existe depuis plus longtemps que Docker Swarm et a donc beaucoup plus de documentation. Il propose également une prise en charge d'applications tierces plus étendue.

### Cluster

Un cluster désigne l’ensemble des ressources que Kubernetes utilise pour exécuter les différentes tâches assignées

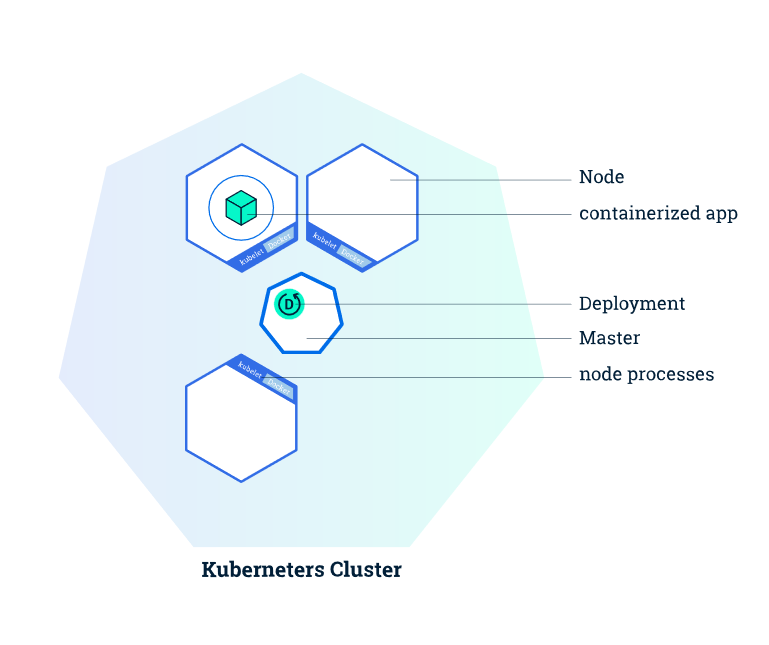


FIGURE 12. Représentation d’un cluster Kubernetes

Le “master” est le chef du cluster Kubernetes. Il se compose de plusieurs composants, un serveur API qui permet de communiquer et d’envoyer des instructions pour gérer le cluster au master, un planificateur et un gestionnaire de contrôleur. Il est chargé de la planification et de la gestion de tous les pods au sein de son cluster. Toutes les instructions effectuées via kubectl[[4]](#footnote-3) passent par le serveur API, qui sont ensuite redirigées vers les noeuds de travail désignés.

Les unités de travail à l'intérieur des clusters s'appellent des noeuds (ou node en anglais). Un noeud est un serveur unique dans le cluster, il peut s'agir d'une machine physique ou virtuelle. Leur but est d’exécuter les pods. Chaque noeud exécute deux composants gérés par Kubernetes, tels que le kubelet, un processus qui veille au bon fonctionnement du noeud et de ces pods, et le proxy kube qui lui est chargé de la gestion des communications réseau du noeud que ça soit au sein du cluster ou vers l’extérieur. Tous les noeuds sont gérés par le master Kubernetes et leur travail consiste à effectuer tout le travail fourni par celui-ci.

### Pods

Un pod est la plus petite unité administrer par Kubernetes qu’il crée et déploie, il regroupe un ou plusieurs conteneurs, qui vont partager une espace de stockage et le même réseau. Le pod possède aussi la façon dont il doit gérer l’exécution des conteneurs et ils sont gérés par les noeuds. Les pods sont le plus souvent utilisés pour regrouper les conteneurs inter-connectés telle qu’une application et un gestionnaire de logs ou de sauvegarde par exemple, mais ils peuvent aussi être utilisés pour regrouper quelques petites applications conteneurisées afin de simplifier l’architecture globale.

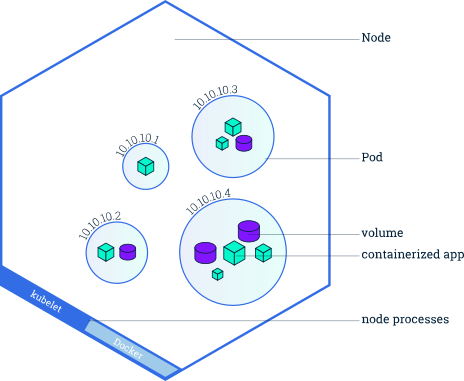


FIGURE 13. Représentation d’un noeud Kubernetes avec ses pods

Les pods sont des unités éphémères, ils ne vivent que le temps de l'exécution de la tâche qui leur a été assigné puis sont tout bonnement détruits, ce qui est aussi le cas si le serveur ou le noeud viendrait à crasher ou tomber en panne, un pod n’est jamais recréé à partir d’un autre pod. Cela permet aux processus s'exécutant à l'intérieur du pod d'avoir une bonne disponibilité, et d’avoir des reliquats de la précédente exécution.

Les conteneurs de différents pods ont des adresses IP uniques ce qui leurs permettent de communiquer avec d’autres pods. Les conteneurs peuvent aussi communiquer via des communications interprocessus (IPC) au sein d’un même pod, bien qu’il soit possible d’utiliser ce type de communication entre différents pods mais ceci nécessite une configuration spécifique.

### Service

Un service est le composant qui définit les règles et gère comment accéder aux pods afin d’exposer certaines fonctionnalités à des utilisateurs ou d’autres services. Les services s’occupe notamment de l’équilibrage des charges (load balancing), des redirections de port, DNS etc…

### Volumes

Un volume Kubernetes est similaire au volumes Docker à quelques différence près. Là où les volumes Docker sont simplement des dossiers partagés entre la machine hôte et le conteneur. Dans Kubernetes, plusieurs types de volumes existent, certains qui ont une durée de vie qui est lié à celle du pod auquel ils appartiennent afin de partager des données entre conteneur ou d’en garder entre chaque démarrage d’un conteneur. Il existe aussi des volumes persistant qui agissent comme une ressource au sein du cluster tout comme un noeud, ils sont notamment utilisés dans le stockage de base de données ou tout autres données qui doit perdurer dans le temps.

## Pipeline CI/CD

L’acronyme “CI/CD” a plusieurs signification. “CI” pour Continuous Integration (intégration continue en français) désigne les processus d’automatisation de test du code d’une application auquel un développeur y a apporté des modifications puis de le fusionner avec le reste de l’application dans un référentiel partagé. Cette solution permet d'éviter de travailler en même temps sur un trop grand nombre d'éléments d'une application, qui pourraient entrer en conflit les uns avec les autres.

Le “CD” de “CI/CD” désigne la distribution continue (Continuous Delivery en anglais) ou/et le déploiement continue (Continuous Deployment). Ce sont deux concepts très proche, le plus souvent ils sont englobés sous le seul terme distribution continue. La distribution continue désigne les processus automatisé de test du code d’une application dans sa globalité et de sa préparation pour un déploiement en production. Le déploiement continue désigne le transfert automatique des nouvelles versions d’une application vers l'environnement de production, où elle peut être utilisées par les clients. Ce processus permet de soulager les équipes d'exploitation surchargées par les tâches manuelles qui ralentissent la distribution des applications.

Tous ces processus automatisé mis bout à bout constitue un pipeline par lequel une application va passer, du développement jusqu’à sa mise en production (cf figure 14)

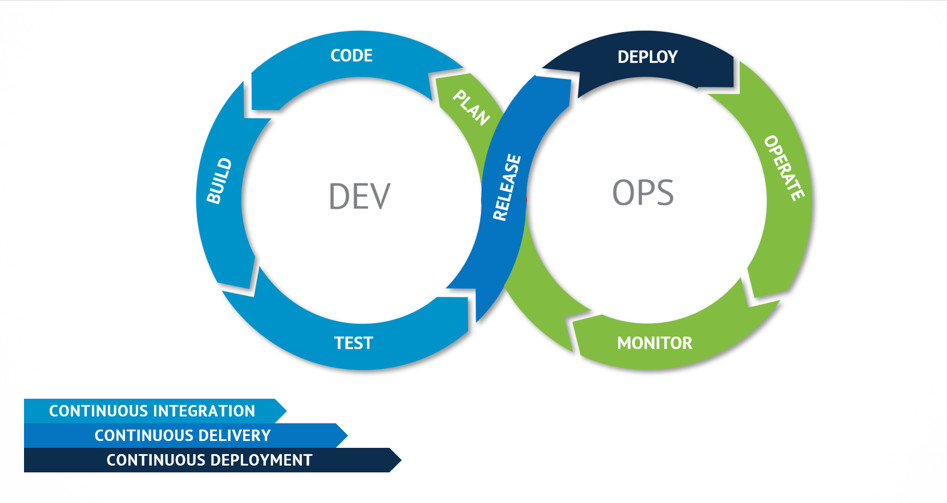


FIGURE 14. Représentation d’un pipeline CI/CD (en bleu) dans un cycle DevOps

### Intégration continue

Le concept de développement d'applications modernes consiste à faire travailler plusieurs développeurs simultanément sur différentes fonctions d'une même application. Toutefois, lors de la fusion des différents morceaux de code produit par les développeurs, la tâche peut s’avérer fastidieuse et complexe, nécessitant beaucoup de procédures manuelles et de temps. En effet, les modifications apportées par un développeur à une application peuvent entrer en conflit avec les différentes modifications apportées simultanément par d'autres développeurs.

L'intégration continue (CI) permet aux développeurs de fusionner plus fréquemment leurs modifications de code dans une « branche » partagée, ou un « tronc » au sein du référentiel. Une fois que les modifications apportées par un développeur sont fusionnées, elles sont validées par la création automatique de l'application et l'exécution de différents niveaux de test automatisés (généralement des tests unitaires et d'intégration) qui permettent de vérifier que les modifications n'entraînent pas de dysfonctionnement au sein de l'application. Le processus d'intégration continue permet de résoudre les dysfonctionnements plus facilement, plus rapidement et plus fréquemment.

### Distribution continue

Après l'automatisation de la création et des tests unitaires et d'intégration dans le cadre de l'intégration continue, la distribution continue automatise la publication du code validé dans un référentiel. Afin de garantir l'efficacité du processus de distribution continue, il est primordial d’avoir introduit un processus d'intégration continue dans le pipeline de développement. La distribution continue permet de disposer d'une base de code toujours prête à être déployée dans un environnement de production.

Dans le cadre de la distribution continue, chaque étape (de la fusion des modifications de code jusqu'à la distribution des versions prêtes pour la production) implique l'automatisation des processus de test et la préparation de l’application pour la publication du code. C’est pendant cette étape que les différents fichiers de débogage seront supprimé, et dans le cas de l’utilisation de Docker, l’application sera conteneurisé et publier sur un Docker Registry. À la fin de ce processus, l'équipe d'exploitation est en mesure de déployer facilement et rapidement une application dans un environnement de production.

### Déploiement continu

L'étape finale d'un pipeline CI/CD mature est le déploiement continu. En complément du processus de distribution continue, le déploiement continu automatise le lancement d'une application dans un environnement de production. Ce processus peut comprendre le déploiement via Kubernetes.

Dans la pratique, dans le cadre du déploiement continu, une modification apportée par un développeur à une application pourrait être publiée quelques minutes seulement après la rédaction du code en question (en supposant qu'elle passe les tests automatisés). Il est ainsi beaucoup plus facile de recevoir et d'intégrer en continu les commentaires des utilisateurs. Ensemble, ces trois pratiques CI/CD réduisent les risques liés au déploiement des applications, puisqu'il est plus simple de publier des modifications par petites touches qu'en un seul bloc. Cette approche nécessite néanmoins un investissement de départ considérable, car les tests automatisés devront être rédigés de manière à s'adapter à un large éventail d'étapes de test et de lancement dans le pipeline CI/CD.

### Avantages et inconvénient

L’inconvénient majeur est la tentation de créer des outils interne pour les différents processus du pipeline CI / CD. Ce serait une grande perte de temps et d’énergie, il existe sur le marché une multitude d'outils pour subvenir à ces besoin, surtout dans le cas de l’utilisation d’un framework comme base d’une application pour lequel des outils on été créé par la communauté ou l’éditeur lui-même. Beaucoup d’entreprises essaient de construire un pipeline en utilisant des scripts bash ou python. Dans la plupart des cas, elles se retrouvent avec une structure de fichiers et de méthodes que seul le développeur connaît donc difficilement maintenable surtout si le développeur vient à quitter l’entreprise.

Les pipelines mal conçus peuvent également être dangereux car ils peuvent limiter le nombre de choix d’outils pouvant être utilisé. Une perte de temps peut survenir à trouver l’outil ou la combinaison d’outils idéal afin de construire le pipeline souhaité. Un pipeline CI/CD se doit d’être évolutif pour parfaitement s’adapter aux évolutions et besoin de ou des applications pour lequel il est construit.

Les avantages des pipelines sont indéniable lorsqu’ils sont bien conçus, tel que :

* les développeurs n’ont qu'à se soucier du code et de ses dépendances sans les craintes lors de la mise en production
* les testeurs peuvent voir quels tests ont été automatiquement exécutés, puis en automatiser davantage, en enquêtant sur les zones à problèmes ou tout simplement en consacrant du temps à la nouvelle fonctionnalité pour la valider par rapport aux exigences du client.
* l'introduction de nouvelles versions ou de corrections de bogues en sont facilité et passes par les mêmes processus de validation
* un déploiement facilité et ce rapidement, ainsi les nouvelles fonctionnalités ou correction sont sur les environnement de production beaucoup plus rapidement

# Conclusion

La virtualisation joue un grand rôle dans le monde informatique. Une technologie très complexe qui apporte une grande facilité dans la gestions des serveurs tout en réduisant drastiquement les coûts pour les entreprises.

Le grand défis qui m’attend est la mise en place de tous ces outils et processus dans les cycles de développements de mon entreprise, avec les changements de fonctionnements qui vont devoir être fait.

# Bibliographie

Bastien L. Virtualisation : qu’est-ce que c’est et à quoi ça sert ? 13 mai 2019 [Consulté en juin 2019]. Disponible à l’adresse : <https://www.lebigdata.fr/virtualisation-definition>

Frédérique Ostré & Thierry Dostes. La virtualisation. 10 juin 2011 [Consulté en mars 2019]. Disponible à l’adresse : <https://resinfo.org/IMG/pdf/Josy_virtualisation_introduction.pdf>

Emilie Ravet. Six types de virtualisation. 2018 [Consulté en mars 2019]. Disponible à l’adresse : <https://www.scalair.fr/blog/virtualisation>

Neal Weinberg. Quel avenir pour la virtualisation de serveurs ? 12 Juillet 2018 [Consulté en mars 2019]. Disponible à l’adresse : <https://www.lemondeinformatique.fr/actualites/lire-quel-avenir-pour-la-virtualisation-de-serveurs%C2%A0-72313.html>

Qu'est-ce que la virtualisation ? [Consulté en avril 2019]. Disponible à l’adresse : <https://www.redhat.com/fr/topics/virtualization/what-is-virtualization>

C’est quoi la virtualisation ? 15 décembre 2015 [Consulté en avril 2019]. Disponible à l’adresse : <https://www.culture-informatique.net/cest-quoi-la-virtualisation/>

<https://www.vmware.com/fr/solutions/virtualization.html>

Présentation de la virtualisation du réseau 2012 [Consulté en avril 2019]. Disponible à l’adresse : <https://docs.oracle.com/cd/E37927_01/html/E36562/gfkbw.html>

Les 10 avantages de la virtualisation des serveurs [Consulté en avril 2019]. Disponible à l’adresse : <https://www.silicon.fr/hub/hpe-intel-hub/les-10-avantages-de-la-virtualisation-des-serveurs>

Aperçu des concepts de virtualisation

21 juin 2019 [Consulté en juin 2019]. Disponible à l’adresse : <https://www.ionos.fr/digitalguide/serveur/configuration/la-virtualisation/>

Laurent Charvériat. Les risques majeurs de la virtualisation. 1 septembre 2010 [Consulté en mars 2019]. Disponible à l’adresse : <https://www.journaldunet.com/solutions/expert/48262/les-risques-majeurs-de-la-virtualisation.shtml>

Bastien L. Cloud Computing – Définition, avantages et exemples d’utilisation. 10 février 2017 [Consulté en avril 2019]. Disponible à l’adresse : <https://www.lebigdata.fr/definition-cloud-computing>

Lucas Bonnet. État de l'art des solutions libres de virtualisation pour une petite entreprise. 12 février 2008 [Consulté en avril 2019]. Disponible à l’adresse : <https://systeme.developpez.com/tutoriels/virtualisation/livre-blanc-lucas-bonnet/>

Nicolas Maquestiaux. Conteneurisation et Virtualisation : où se situe le point de bascule ? 27 avril 2018 [Consulté en mars 2019]. Disponible à l’adresse : <https://www.cyres.fr/blog/conteneurisation-virtualisation-point-de-bascule/>

Différences et compatibilité entre virtualisation et conteneurisation. [Consulté en mai 2019]. Disponible à l’adresse : <http://www.qualit-blog.com/differences-et-compatibilite-entre-virtualisation-et-conteneurisation/>

Alexandre Fillate. Docker pour les nu... pour les débutants. 20 octobre 2014 [Consulté en mai 2019]. Disponible à l’adresse : <https://blog.ippon.fr/2014/10/20/docker-pour-les-nu-pour-les-debutants/>

Sébastien Bernard. Introduction aux risques et à la sécurisation de Docker. 20 avril 2017 [Consulté en mai 2019]. Disponible à l’adresse : <https://www.lemagit.fr/tribune/Introduction-aux-risques-et-a-la-securisation-de-Docker-1e-partie>

DOCKER, qu'est-ce que c'est ? [Consulté en mai 2019]. Disponible à l’adresse : <https://www.redhat.com/fr/topics/containers/what-is-docker>

Noureddine GRASSA. Cours Virtualisation et Cloud. publié le 7 août 2014 [Consulté en mai 2019]. Disponible à l’adresse : <http://n.grassa.free.fr/cours/cours_virtualisation_et_Cloud.pdf>

[Consulté en juin 2019]. Disponible à l’adresse : <https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf>

Kumar Yadav, Anuj & L. Garg, M & Mehra, Ritika. (2019). Docker Containers Versus Virtual Machine-Based Virtualization: Proceedings of IEMIS 2018, Volume 3. 10.1007/978-981-13-1501-5\_12.

[Consulté en juin 2019]. Disponible à l’adresse : <https://www.researchgate.net/publication/327389502_Docker_Containers_Versus_Virtual_Machine-Based_Virtualization_Proceedings_of_IEMIS_2018_Volume_3>

Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu, and Wei Zhou. A Comparative Study of Containers and Virtual Machines in Big Data Environment. 5 juillet 2018 [Consulté en juin 2019]. Disponible à l’adresse : <https://arxiv.org/pdf/1807.01842.pdf>

[Consulté en juin 2019]. Disponible à l’adresse : <https://hal.archives-ouvertes.fr/hal-01578325/file/Paper.pdf>

[Consulté en juin 2019]. Disponible à l’adresse : <https://kubernetes.io/fr/docs/tutorials/kubernetes-basics/>

Kubernetes, qu'est-ce que c'est ?. [Consulté en juillet 2019]. Disponible à l’adresse : <https://www.redhat.com/fr/topics/containers/what-is-kubernetes>

Bastin L. Kubernetes – Tout savoir sur la plateforme d’orchestration de containers. publié le 25 octobre 2018 [Consulté en juillet 2019]. Disponible à l’adresse : <https://www.lebigdata.fr/kubernetes-definition>

Guillaume Serries. Pourquoi Kubernetes est devenu très tendance (et comment en profiter). publié le 18 avril 2019 [Consulté en juillet 2019]. Disponible à l’adresse : <https://www.zdnet.fr/pratique/pourquoi-kubernetes-est-devenu-tres-tendance-et-comment-en-profiter-39864740.htm>

[Consulté en juillet 2019]. Disponible à l’adresse : <https://aws.amazon.com/fr/kubernetes/>

Marko Anastasov. CI/CD Pipeline: A Gentle Introduction. publié le 14 mars 2019 [Consulté en août 2019]. Disponible à l’adresse : <https://semaphoreci.com/blog/cicd-pipeline>

Understanding the CI/CD Pipeline: What It Is, Why It Matters. mis à jour le 29 juillet 2019 [Consulté en aout 2019]. Disponible à l’adresse : <https://www.plutora.com/blog/understanding-ci-cd-pipeline>

[Consulté en août 2019]. Disponible à l’adresse : <https://aws.amazon.com/fr/getting-started/projects/set-up-ci-cd-pipeline/>

Continuous Delivery Pipeline. mis à jour le 10 juin 2019 [Consulté en aout 2019]. Disponible à l’adresse : <https://www.scaledagileframework.com/continuous-delivery-pipeline/>

Izzy Azeri. What is CI/CD? publié le 23 janvier 2019 [Consulté en août 2019]. Disponible à l’adresse : <https://www.mabl.com/blog/what-is-cicd>

1. Architecture utilisé par les processeurs Intel 32 bits [↑](#footnote-ref-0)
2. Hyperviseur open source de machine virtuelle de type 1 utilisant le noyau Linux [↑](#footnote-ref-1)
3. Désigne un groupe de serveur, de machine virtuelle ou de conteneur par exemple. [↑](#footnote-ref-2)
4. Outil en ligne de commande de Kubernetes permettant la communication avec le master et son cluster [↑](#footnote-ref-3)