

Remerciements

Si nous sommes arrivés à ce stade de nos vies c'est seulement grâce à **Dieu**, c'est lui qui nous a mis sur la bonne voie, nous a donné la bonne santé et nous a entouré de personnes formidables, merci **Dieu**.

En premier temps nos meilleurs remerciements vont également à notre encadreur au Faculté des sciences de Bizerte, **Mr. MOHAMED OUELD EL HASSEN** pour sa disponibilité, son bon sens pédagogique et son soutien permanent qui nous ont énormément aidés à l'élaboration de ce travail.

En second lieu, on tient à manifester l'expression de nos respects les plus distingués à **MR.SOUFIENNE CHRAIGUI** pour avoir accepté d'être notre encadreur ainsi que pour les multiples conseils et orientations qu'il nous a prodigué au sein **d'HORIZAN-DATA**.

Nous remercions le personnel pour nous avoir accueillis les bras ouverts, et fait vivre pendant quatre mois une expérience humaine et professionnelle inoubliable.

Enfin nos remerciements s'adressent à **tous nos enseignants au Faculté des sciences de Bizerte** pour les connaissances qu'ils nous ont données tout au long de ces trois années d'étude.

Sommaire

Remerciements	1
Introduction générale :	1
Chapitre 1 : Présentation Générale Du projet	4
I. Introduction :	4
II. Organisme de l'accueil : HORIZON-DATA.....	4
III. Etude de l'Existant :	4
Etude Comparative des solutions de virtualisation:.....	4
IV. Solutions proposées : Les conteneurs avec Docker	5
Densifier le serveur	6
V. Problématique:	6
VI. Conclusion :	6
Chapitre 2 : Installation et manipulation d'un outil de gestion des conteneurs.....	7
I. Introduction :	7
1. Quest Ce que docker:	7
2. Vue d'ensemble sur Docker :	8
3. Historique:.....	8
II. Composants :	10
1. Sur quels outils se base docker :	10
• LXC.....	10
• Cgroups	11
• Namespace :	12
• Language utilisée par Docker :	13
III. Architecture client-serveur Docker :	15
1. Description de l'architecture:	15
IV. Installation & configuration :	19
1. Environnement de développement:.....	19
2. Installation de Docker :	20
a. Mettre en place le référentiel	20
b. Obtenir Docker CE.....	20
c. Tester votre installation Docker CE	21
• Note:.....	21
V. conclusion:	40
Chapitre 3: Proposition et Simulation d'une application de paiement en ligne	41
I. Introduction:.....	41
• Diagramme :	45
Chapitre 4: Mise en place des règles de sécurité pour la solution	46
I. Sécurisation des paiements en lignes et méthodes alternatives de paiement	46
1. Les signes de sécurité visibles:	46
2. Les signes de sécurité invisibles:	47
II. La norme PCI-DSS	49

1. Les 12 règles :	50
III. Utilisation d'un porte-monnaie électronique :	52
1. Principe	52
IV. Conclusion :	55
Chapitre 5 : tests et recommandation	56
I. Introduction :	56
II. Recommandations :	56
• Docker Images :	56
• Network Namespaces :	56
• Logging & Auditing :	57
• SELinux ou AppArmor :	57
• Privilèges du démon docker :	58
• cgroups :	59
• Les fichiers binaires SUID / GUID :	59
• Groupe de contrôle des périphériques (/ dev / *) :	60
• Services et applications :	60
• Points de montage :	61
• Linux Kernel (Le Noyeau linux) :	61
• User Namespaces :	61
• Virtualization complète :	63
Les points d'interet :	64
⇒ Docker Bench for Security	65
III. Conclusion :	67
Conclusion générale	68
ANNEXE :	70
Quelques tests effectués sur la machine ubuntu :	71
Bibliographie	79

LISTE DES FIGURES

Figure 1: Explication.....	2
Figure 2: VM VS Conteneur.....	5
Figure 3: Architecture client-server Docker	15
Figure 4: Docker version.....	16
Figure 5: architecture d'api.....	17
Figure 6: Containers vs. VMS	18
Figure 7: les versions stables d'Ubuntu	19
Figure 8: Evolution de la developpement avec les conteneurs	40
Figure 9: Architecture plus détaillée de l'application.....	43
Figure 10: carte de paiement.....	46
Figure 11: Explication des requêtes faites Durant la phase du paiement.....	53
Figure 12: Les points principales	55

Introduction générale :

Aujourd'hui, notre monde est confronté à une évolutivité permanente dans le domaine de l'informatique, environnemental et sociétal. En effet, les entreprises qui utilisent des serveurs afin d'y traiter leurs données doivent organiser leurs infrastructures informatiques pour être le plus efficient possible. Afin d'optimiser leurs fondations et réduire les couts, les entreprises font appel à la virtualisation. Ce projet concernera uniquement les entreprises, le but de ce mécanisme informatique est de mutualiser les capacités de chaque serveur. Il permet à l'utilisateur de réaliser des économies sur l'infrastructure physique. Les contraintes sont les techniques, En effet pour utiliser la virtualisation, l'entreprise doit faire appel à une personne compétente dans ce domaine .de plus, les dépenses pour l'entreprise sont importantes sur une utilisation à long terme.

L'informatique dans les nuages, connue sous le nom de « Cloud computing » est un concept informatique récent qui vise à décrire un ensemble de techniques utilisées pour délivrer des capacités informatiques en tant que services. Ce procédé interconnecte et met en coopération des ressources informatiques au sein d'une même entité, ou bien au sein de structures externes comme une solution de serveur mail que l'on aurait externalisée et qui serait gérée par un tiers. Pour fonctionner, le Cloud s'appuie sur les technologies de la virtualisation et d'automatisation. Les protocoles et les standards Internet sont utilisés comme base pour les modes d'accès.

Depuis quelques années, les entreprises s'intéressent de plus en plus aux technologies de virtualisation. Bien que ce concept ne soit pas nouveau, de nombreuses solutions sont actuellement mises en œuvre autour de la virtualisation.

Ainsi la virtualisation est un ensemble de techniques matérielles et/ou logicielles qui autorisent l'exécution de plusieurs applications indépendantes sur une même machine hôte. Grâce à la virtualisation, il est possible d'exécuter plusieurs systèmes d'exploitation (OS invité) sur un même serveur.

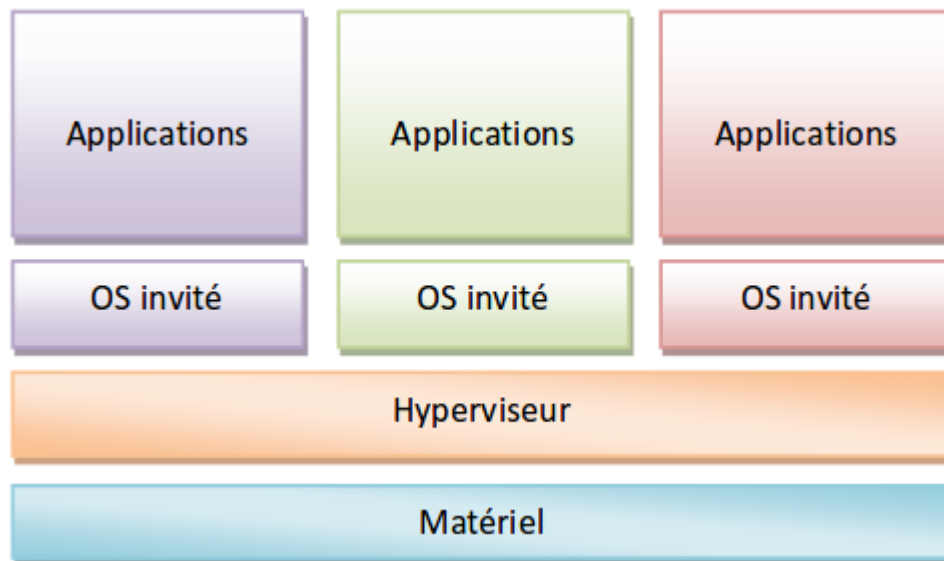


Figure 1: Explication

Ainsi, il n'est plus nécessaire d'utiliser un serveur par application. On parle souvent d'environnement virtuel (Virtual Environment — VE) ou de serveur privé virtuel (Virtual Private Server— VPS) lorsqu'une machine exploite la virtualisation. Pour bénéficier de cette technologie, il suffit d'équiper une machine d'un logiciel de virtualisation permettant d'ajouter une couche de virtualisation, appelée hyperviseur.

Ce hyperviseur masque les véritables ressources physiques de la machine afin de proposer des ressources différentes et spécifiques en fonction des applications qui tournent. IL y a donc une totale indépendance entre le matériel et les applications.

Le logiciel de virtualisation simule autant de machines virtuelles que de systèmes d'exploitation souhaité. Chaque OS croit alors qu'il est installé seul sur une machine alors qu'en réalité, plusieurs OS peuvent fonctionner en parallèle en partageant les mêmes ressources.

En consolidant la charge de travail de différents serveurs sur une même machine, on augmente la criticité de la machine. Le piratage d'un serveur hôte entraîne l'arrêt de l'ensemble des services consolidés sur la machine.

La virtualisation entraîne une augmentation des risques de sécurité informatique. La principale raison réside dans le fait qu'une machine supportant plusieurs serveurs

virtuels est forcément plus vulnérable qu'un seul serveur physique. Au sein des entreprises, les directions des systèmes d'information et des risques commencent à en prendre conscience.

Dans ce cadre s'inscrit notre projet de fin d'étude qui consiste à réaliser une étude sur les solutions libres de la virtualisation et leur sécurité.

Ce travail a été réalisé en vue d'obtenir le diplôme en technologies de l'informatique et de Télécommunication à la faculté des sciences de Bizerte. Le présent rapport sera composé de Cinq chapitres :

Dans le premier chapitre, l'organisme d'accueil, qui nous a accueilli durant notre période de stage, sera présenté en rappelant les produits et les services qu'il offre. Par la suite ,nous rappellerons les problématiques qui se posent dans le réseau de l'entreprise.

Dans le deuxième on va parler sur un outil de gestion des conteneurs , son role , ses composants et les étapes de son installation ,

Par la suite la troisième chapitre décrit l'implémentation d'une application dans les conteneurs et la configuration nécessaire pour son fonctionnement

Et la quatrième est consacrée pour la mise en place des règles de sécurité pour notre application

Enfin une étude de sécurité sera réalisée sur la nouvelle technique de virtualisation en appliquant les tests de pénétrations sur les conteneurs docker

Chapitre 1 : Présentation Générale Du projet

I. Introduction :

Ce chapitre permet de mettre notre projet dans son cadre général. Il comporte des parties: la première porte sur l'entreprise d'accueil et ses activités et l'étude de l'existant et définit la problématique de ce projet.

II. Organisme de l'accueil : HORIZON-DATA

HORIZON-DATA est une entreprise multi-nationalité Opérant en Tunisie et à l'International qui offre à ces clients plusieurs services tel que le Web services, e-commerce et développement des Applications mobiles ainsi plusieurs solutions tel que le Cloud computing, E-Mailing et Collaboration et Système de pointage.

Parmi les missions principales de cette entreprise après intégration des solutions et de faire suivre la progression et la performance de ces solutions par la construction des Audit techniques, sémantiques, concurrentiels Grâce à son expérience et à son équipe créative, dynamique et expérimentée dans divers secteurs, ainsi que l'intégration technique des recommandations par les Rapports trimestriels faite par l'équipe .

III. Etude de L'Existant :

Etude Comparative des solutions de virtualisation:

Une machine virtuelle (VM - Virtual Machine) « imite » intégralement un serveur.

Dans un serveur virtualisé type, chaque VM « invitée » contient un système d'exploitation complet, avec ses pilotes, fichiers binaires ou bibliothèques, ainsi que l'application elle-même. Chaque VM s'exécute alors sur un hyperviseur, qui s'exécute à son tour sur un système d'exploitation hôte, qui lui-même fait fonctionner le matériel du serveur physique.

Bien que la méthode ait fait ses preuves, on s'aperçoit aisément que chaque itération du système d'exploitation hôte et des fichiers binaires associés risque d'entraîner des

doublons entre les VM , cela gaspille la mémoire du serveur et limite forcément le nombre de VM prises en charge par chaque serveur. Ce qui —pour certains — remet en cause le futur de cette technologie.

A la base, le concept de conteneurisation permet aux instances virtuelles de partager un système d'exploitation hôte unique, avec ses fichiers binaires, bibliothèques ou pilotes.

Cette approche réduit le gaspillage des ressources car chaque conteneur ne renferme que l'application et les fichiers binaires ou bibliothèques associés. On utilise donc le même système d'exploitation (OS) hôte pour plusieurs conteneurs, au lieu d'installer un OS (et d'en acheter la licence) pour chaque VM invitée. Ce procédé est souvent appelé virtualisation au niveau du système d'exploitation.

IV. Solutions proposées : Les conteneurs avec Docker

À l'aide de conteneurs, tout ce qui est nécessaire pour faire exécuter un logiciel est emballé dans des conteneurs isolés. Contrairement aux machines virtuelles, les conteneurs ne regroupent pas un système d'exploitation complet: seules les bibliothèques et les paramètres requis pour que le logiciel fonctionne soient nécessaires. Cela permet des systèmes autonomes, légers et garantit que les logiciels fonctionneront toujours de la

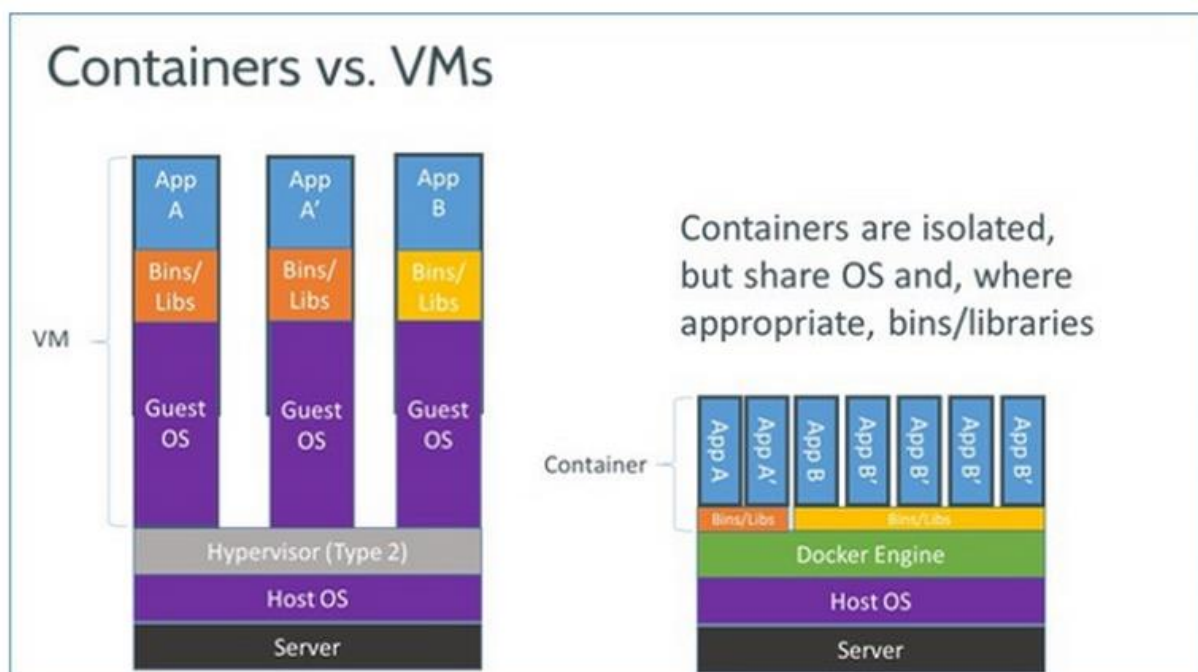


Figure 2: VM VS Conteneur

Densifier le serveur

Nous utilisons moins de ressources, et nous améliorons leur densité sur le serveur.

V. Problématique:

La Performance VS La sécurité

La sécurité en question :

« DOCKER Parce que ça reste assez jeune, des questions de sécurité se posent encore !! »

Et pour vérifier le niveau de sécurité, nous allons faire une Etude de la sécurité de conteneurisation d'une application de paiement en ligne en se référant aux régies PCI—DSS.

Pour permettre une approche unique de la mise en place des exigences de sécurité, les organisations de cartes de crédit VISA et Mastercard ont développé une norme de sécurité commune (PCI DSS = Payment Card Industry Data Security Standard).

VI. Conclusion :

Pour vérifier l'intégrité et la sécurité de nos applications sur cette nouvelle infrastructure on va essayer de déployer une application de paiement en ligne et pour faire cela on est besoin de passer par une étape exhaustif mais nécessaire pour bien comprendre le fonctionnement des conteneurs .donc comme première étape on va installer un outil de gestion des conteneurs<<Docker>>

Chapitre 2 : Installation et manipulation d'un outil de gestion des conteneurs

I. Introduction :

1. Quest Ce que docker:

Docker est un logiciel libre qui automatise le déploiement d'applications dans des conteneurs logiciels. « Docker est un outil qui peut emballer une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur Linux ». Ceci permet d'étendre la flexibilité de construction des applications et leur portabilité, que ce soit sur la machine locale, un Cloud privé ou public, etc.

Docker étend le format de conteneur Linux standard, LXC, avec une API de haut niveau fournissant une solution de virtualisation qui exécute les processus de façon isolée. Docker utilise LXC, Cgroups, et le noyau Linux lui-même. Contrairement aux machines virtuelles traditionnelles, un conteneur Docker n'inclut pas de système d'exploitation, s'appuyant sur les fonctionnalités du système d'exploitation fournies par l'infrastructure sous-jacente.

La technologie de conteneur de Docker peut être utilisée pour étendre des systèmes distribués de façon à ce qu'ils s'exécutent de manière autonome depuis une seule machine physique ou une seule instance par nœud. Cela permet aux nœuds d'être déployés au fur et à mesure que les ressources sont disponibles, offrant un déploiement transparent et similaire aux PaaS pour des systèmes comme Apache Cassandra, Riak ou d'autres systèmes distribués.

2. Vue d'ensemble sur Docker :

Docker permet la mise en œuvre de conteneurs s'exécutant en isolation, via une API de haut-niveau. Construit sur des capacités du noyau Linux (surtout les Cgroups et espaces de nommage) un conteneur Docker, à l'opposé de machines virtuelles traditionnelles, ne requiert aucun système d'exploitation séparé et n'en fournit aucun. Il s'appuie plutôt sur les fonctionnalités du noyau et utilise l'isolation de ressources (comme le processeur, la mémoire, les entrées et sorties et les connexions réseau) ainsi que des espaces de noms séparés pour isoler le système d'exploitation tel que vu par l'application.

Utiliser Docker pour créer et gérer des conteneurs peut simplifier la mise en œuvre de systèmes distribués en permettant à des multiples applications, tâches de fond et autres processus de s'exécuter de façon autonome sur une seule machine physique ou à travers un éventail de machines isolées.

3. Historique:

Docker a été développé par Solomon Hykes pour un projet interne de dotCloud, une société proposant une plate—forme en tant que service, avec les contributions d'Andrea Luzzardi et François—Xavier Bourlet, également employés de dotCloud.

Docker est une évolution basée sur les technologies propriétaires de dotCloud, elles—mêmes construites sur des projets open source.

Docker a été distribué en tant que projet open source à partir de mars 2013.

Au 18 novembre 2013, le projet a été mis en favoris plus de 7300 fois sur GitHub (projet le plus populaire), avec plus de 900 forks et 200 contributeurs.

Au 9 mai 2014, le projet a été mis en favoris plus de 11000 fois sur GitHub, avec plus de 1900 forks et 420 contributeurs.

En octobre 2015, le projet a été mis en favoris plus de 25000 fois sur GitHub, avec plus de 6500 forks et 1100 contributeurs.

En septembre 2016, le projet a été mis en favoris plus de 34000 fois sur GitHub, avec plus de 10000 forks et 1400 contributeurs

Vous pouvez penser Docker comme étant un gestionnaire de conteneurs pour les systèmes d'exploitation basés sur UNIX. Donc, fondamentalement, il vous permet de créer de nombreux environnements isolés sur un seul noyau.

- **Docker pour les développeurs**

Docker automatise les tâches répétitives de configuration des environnements de développement afin que les développeurs puissent se concentrer sur ce qui compte: la construction d'un excellent logiciel.

Les développeurs utilisant Docker ne doivent pas installer et configurer des bases de données complexes, ni se soucier de basculer entre des versions de chaîne d'outils de langue incompatibles. Lorsqu'une application est embarquée, cette complexité est poussée dans des conteneurs facilement construits, partagés et exécutés. La mise en place d'un collègue à une nouvelle base de code ne signifie plus les heures passées à installer le logiciel et à expliquer les procédures d'installation. Le code fourni avec Dockerfiles est plus simple à utiliser: les dépendances sont tirées sous forme d'images Docker normalement emballées et toute personne avec Docker et un éditeur installé peuvent créer et déboguer l'application en quelques minutes.

- **Docker Pour l'entreprise :**

Docker est au cœur de la plate-forme d'application moderne, développeur de pont et informatique, Linux et Windows. Docker fonctionne dans le Cloud aussi bien que sur place; Et prend en charge les architectures traditionnelles et les micros services. Utilisez Docker pour créer, mettre en réseau, sécuriser et planifier des conteneurs et les gérer du développement à la production. Docker définit les entreprises sur la voie de la transformation numérique en permettant à toutes les applications d'être agiles, compatibles avec les nuages et sécurisés à des coûts optimaux.

II. Composants :

les composants de base de Docker :

Docker a 3 composants: Docker Daemon, Docker Images et Docker repositories.

Docker Daemon fonctionne en tant que root et orchestre essentiellement tous les conteneurs.

Docker images sont des images de système d'exploitation virtuels et techniquement ils ont de manière moins d'empreinte que les images d'OS réels. Les systèmes de fichiers sont sauvegardés et empilés, ils peuvent donc être unifiés à la dernière révision à tout moment. Un peu comme des « snapshot » d'une VM, mais en mieux. En raison de la nature du système de fichiers toutes ces opérations d'écriture sont également mises en cache. À savoir que, si Docker découvre que vous ne disposez pas des changements jusqu'à une certaine révision, il joue le tout jusqu'à ce que la révision, notamment du cache en appliquant les écritures. Ceci donne un énorme avantage de performance sur le temps de provisionnement. En plus de tous ces avantages, les images peuvent aussi être échangées avec d'autres personnes.

Dernière composante de Docker l'écosystème est le référentielle (repositories). Les images que vous détenez doit être stockées quelque part, que ce soit privé ou public. Docker fournit alors un référentiel commun.

1. Sur quels outils se base docker :

- **LXC**

, contraction de l'anglais Linux Containers est un système de virtualisation, utilisant l'isolation comme méthode de cloisonnement au niveau du système d'exploitation. Il est utilisé pour faire fonctionner des environnements Linux isolés les uns des autres dans des conteneurs partageant le même noyau et une plus ou moins grande partie du système hôte. Le conteneur apporte une virtualisation de l'environnement d'exécution (processeur, mémoire vive, réseau, système de fichier...) et non pas de la machine. Pour cette raison, on parle de «conteneur» et non de machine virtuelle.

LXC repose sur les fonctionnalités des cgroups du noyau Linux. Il repose également sur d'autres fonctionnalités de cloisonnement comme le cloisonnement des espaces de nommage du noyau, permettant d'éviter à un système de connaître les ressources utilisées par le système hôte ou un autre conteneur

Docker est un gestionnaire de conteneurs initialement basé sur LXC.

- **Cgroups**

(Control groups) est une fonctionnalité du noyau Linux pour limiter, compter et isoler l'utilisation des ressources (processeur, mémoire, utilisation, disque etc.).

L'un des buts de la conception de Cgroups a été de fournir une interface unifiée à différents cas d'utilisation, en allant du contrôle de simples processus (comme Nice) à la virtualisation au niveau du système d'exploitation (comme OpenVZ, Linux-Vserver, LXC).

Cgroups fournit :

- ° Limitation des ressources : des groupes peuvent être mis en place afin de ne pas dépasser une limite de mémoire — cela inclut aussi le cache du système de fichier.
- ° Priorisation : certains groupes peuvent obtenir une plus grande part de ressources processeur ou de bande passante d'entre /sortie.
- ° Comptabilité : permet de mesurer la quantité de ressources consommées par certains systèmes en vue de leur facturation par exemple.
- ° Isolation : séparation par espace de nommage pour les groupes, afin qu'ils ne puissent pas voir les processus des autres, leurs connexions réseaux ou leurs fichiers
- ° Contrôle : figer les groupes ou créer un point de sauvegarde et redémarrer.

°explication de fonctionnement du Cgroup :

Un groupe de contrôle est une suite de processus qui sont liés par le même critère. Ces groupes peuvent être organisés hiérarchiquement, de façon que chaque groupe hérite des limites de son groupe parent. Le noyau fournit l'accès à plusieurs contrôleurs (sous-systèmes) à travers l'interface Cgroup. Par exemple, le contrôleur «Memory» limite l'utilisation de la mémoire, «cpuacct» comptabilise l'utilisation du processeur, etc.

- **Namespace :**

Le terme espace de noms (Name Space) désigne en informatique un lieu abstrait conçu pour accueillir des ensembles de termes appartenant à un même répertoire

Isolation par espace de nommage

Bien que ne faisant pas techniquement partie du travail des groupes de contrôle, une fonctionnalité liée est l'isolation par espace de nommage, dans laquelle des ensembles de processus sont séparés de telle façon qu'ils ne puissent pas «voir» les ressources des autres groupes. Par exemple, un espace de nommage par identifiant de processus (PID) fournit un ensemble distinct d'identifiants de processus dans chaque espace de nommage. Sont aussi disponibles des espaces de nommage par mount, réseau, UTS . Très tôt dans le développement des groupes de contrôle, le sous—système «ns» a été ajouté, pour intégrer les espaces de nommage et les groupes de contrôle. Si le groupe de contrôle «ns» était monté, chaque espace de nommage aurait dû aussi créer un nouveau groupe dans la hiérarchie des groupes de nommage.

° L'espace de nommage par identifiant de processus (PID namespace) fournit l'isolation pour l'allocation des identifiants de processus (PIDs), la liste des processus et de leurs détails.

Tandis que le nouvel espace de nom est isolé de ses adjacents, les processus dans son espace de nommage «parent» voient toujours tous les processus dans les espaces de nommage enfants — quoique avec des numéros de PIB différent.

° L'espace de nommage réseau (Network namespace) isole le contrôleur de l'interface

réseau (physique ou virtuel), les règles de pare-feu iptables, les tables de routage, etc.

° Les espaces de nommage réseau peuvent être connectés les uns avec chacun des autres en utilisant le périphérique virtuel Ethernet

° L'espace de nommage « UTS» ("UTS" namespace) permet le changement de nom d'hôte.

° L'espace de nommage de montage (Mount namespace) permet de créer différents modèles de systèmes de fichiers, ou de créer certains points de montage en lecture-seule.

Les espaces de nom sont créés avec la commande «unshare» ou un appel système.

- **La platform Docker**

Docker offre la possibilité d'emballer et d'exécuter une application dans un environnement peu isolé appelé conteneur, L'isolement et la sécurité vous permettent d'exécuter plusieurs conteneurs simultanément sur un hôte donné. Les conteneurs sont légers car ils n'ont pas besoin de la charge supplémentaire d'un hyperviseur, mais fonctionnent directement dans le noyau de la machine hôte.

Cela signifie que vous pouvez exécuter plus de conteneurs sur une combinaison matérielle donnée que si vous utilisiez des machines virtuelles. Vous pouvez même exécuter des conteneurs Docker dans des machines hôtes qui sont en fait des machines virtuelles!

- **Language utilisée par Docker :**

Go est un langage de programmation compilé concurrent inspiré de C et Pascal . Ce langage a été développé par Google.

Go veut faciliter et accélérer la programmation à grande échelle : en raison de sa simplicité, sa compilation serait de 80 % à 90 % plus rapide que la compilation classique du C, et il est donc concevable de l'utiliser aussi bien pour écrire des applications, des scripts ou de grands systèmes.

Cette simplicité est nécessaire aussi pour assurer la maintenance et l'évolution des programmes sur plusieurs générations de développeurs.

Projets utilisant go :

- **Docker** - Création, déploiement et exécution d'application dans des conteneurs
- **Kubernetes** - Système de gestion d'applications conteneurisées
- **Grafana** - Outil de monitoring, d'analyse de métriques et de création de Dashboard
- **Caddy** - Serveur web
- **Influxdb** - Base de donnée destinées à l'enregistrement d'évènements, de métriques
- **Cayley** - Base de donnée orientée graphe
- **Cockroach** - Base de donnée SQL distribuée

«Hello, world»

Voici un exemple d'un programme Hello World typique écrit en Go.

Package main

Import "fmt"

Func main () {

fmt.Printf ("Hello, world\n")

Pour quoi Go?

Les cinq raisons pour lesquelles nous avons utilisé Go?

- 1) compilation statique.
- 2) neutre
- 3) il a ce dont nous avons besoin
- 4) environnement de développement complet
- 5) construction multi-architecture

III. Architecture client-serveur Docker :

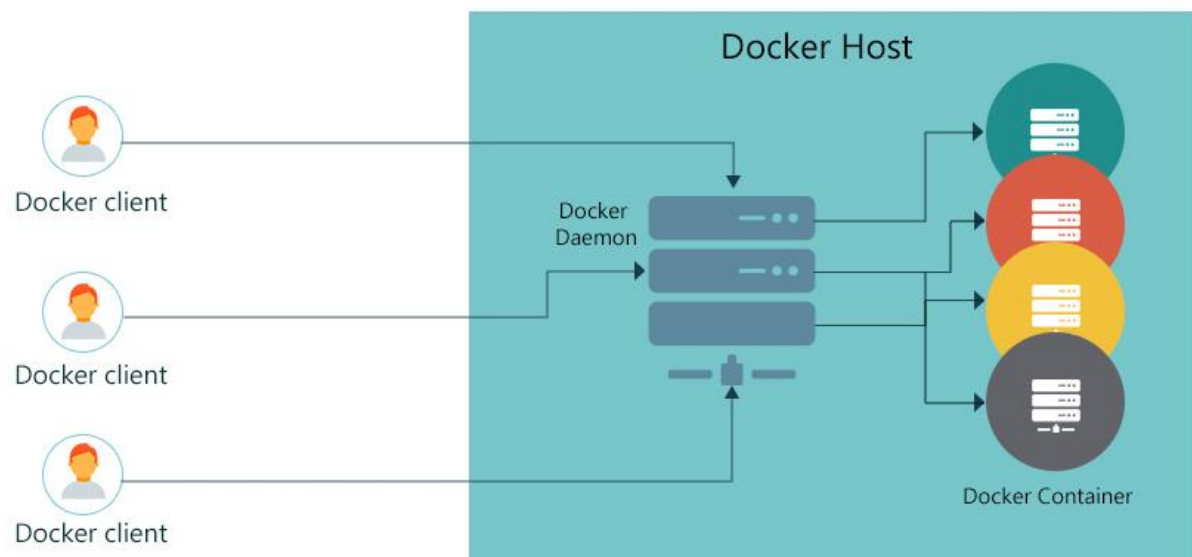


Figure 3: Architecture client-serveur Docker

1. Description de l'architecture:

Docker utilise une architecture client—serveur. Le client Docker parle avec le démon Docker, qui fait le travail lourd de la construction, de l'exécution et de la distribution de vos conteneurs Docker. Le client Docker et Daemon peuvent s'exécuter sur le même système ou vous pouvez connecter un client Docker à un démon Docker distant. Le client Docker et Daemon communiquent à l'aide d'une API REST, sur des sockets UNIX ou une interface réseau.

- **API REST:**

Parmi toutes les fonctionnalités disponibles, il en est une qui explique pour beaucoup le dynamisme phénoménal de l'écosystème Docker. Il s'agit de Docker Remote API, une API REST extrêmement bien conçue exposant toutes les fonctionnalités du moteur Docker et permettant ainsi de piloter un hôte Docker depuis une machine ou une application distante.

Toutes les solutions d'orchestration de conteneurs Docker utilisent directement ou indirectement l'API

Pour voir la version de l'API, votre démon Docker et son support client, utilisez la version docker:

```
iheb@iheb-Compaq-Presario-A900-Notebook-PC ~ $ su --
Password:
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker version
Client:
Version:      17.03.1-ce
API version:  1.27
Go version:   go1.7.5
Git commit:   c6d412e
Built:        Mon Mar 27 17:10:36 2017
OS/Arch:      linux/amd64

Server:
Version:      17.03.1-ce
API version:  1.27 (minimum version 1.12)
Go version:   go1.7.5
Git commit:   c6d412e
Built:        Mon Mar 27 17:10:36 2017
OS/Arch:      linux/amd64
Experimental: false
iheb-Compaq-Presario-A900-Notebook-PC iheb # █
```

Figure 4: Docker version

Voila l'architecture de l'API de DOCKER :

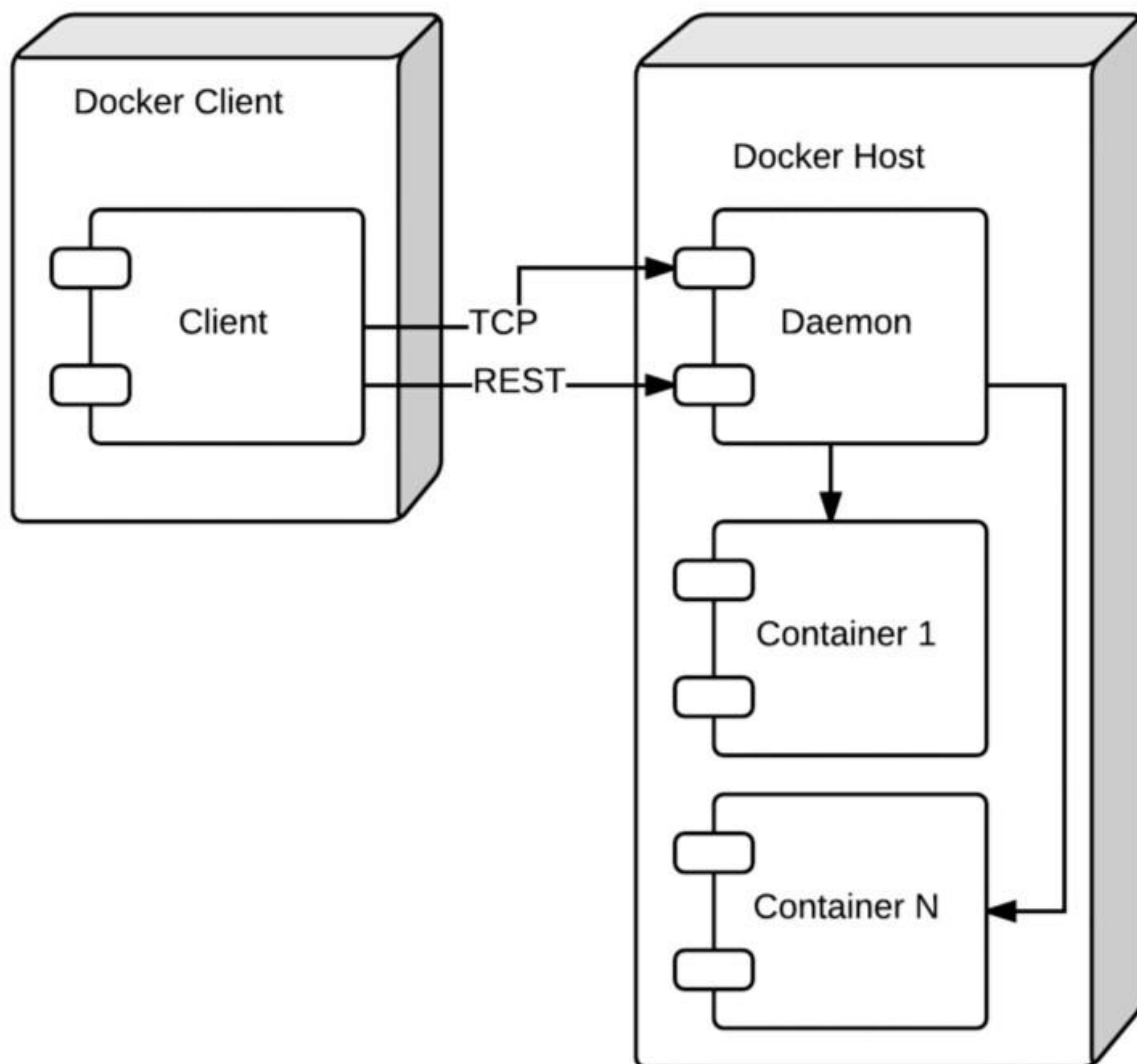


Figure 5: architecture d'api

- **Les appels systèmes:**

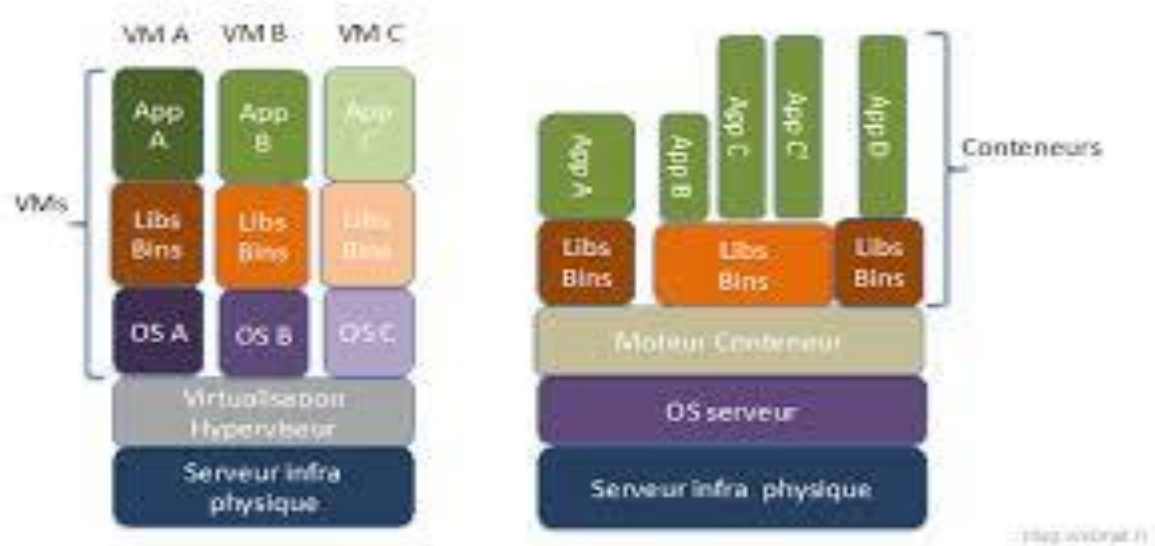


Figure 6: Containers vs. VMS

Pour la virtualisation traditionnelle on utilise les machine virtuelle qui nécessite un système d'exploitation et ses dépendances et un couche qui sépare les machines virtuelles de système hôte, les appels système et l'échange des informations avec le kernel se fait en passant par l'OS invité et puis par l'hyperviseur et enfin par l'os de l'hôte sur laquelle seront installer ces Vms. Par contre, cette couche est éliminé dans la méthode de conteneurisation ce qui implique un allégement de point de vue des appelles systèmes ou ils sont faite directement entre les conteneurs et le kernel.

On parle aussi de la portabilité pour les VM's on ne peut pas déplacer une VM d'un serveur a un autre. Ce n'est pas le cas du container on peut les exécuter sur n'importe quelle environnement sur le quelle il est installer docker. On peut construire un conteneur dans un environnement UNIX et on l'exécute sur un MacOS.

Donc comme on a expliqué les conteneurs sont plus efficace et se Bénéficie de plusieurs avantages que la virtualisation traditionnelle. On va continuer a la phase d'installation.

IV. Installation & configuration :

1. Environnement de développement:

Environnement linux: le choix de la distribution affecte le bien déroulement du projet, c'est pour cela on a pris la stabilité de la distribution pour préparer notre environnement de travaille : pour cela on a choisit deux version :

Ubuntu 16.04 LTS: (LTS: Long Term Support)

La liste suivante recense les versions stables d'Ubuntu profitant toujours de mises à jour de sécurité, de la plus ancienne à la plus récente.

Numéro de version	Nom de code	Date de sortie	Date de fin de soutien
Ubuntu 14.04 LTS	The Trusty Tahr (le Bélier Confiant)	17 avril 2014	Avril 2019
Ubuntu 16.04 LTS	The Xenial Xerus (le Xerus Hospitalier)	21 avril 2016	Avril 2021
Ubuntu 16.10	The Yakkety Yak (le Yak Bavard)	13 octobre 2016	Juillet 2017
Ubuntu 17.04	The Zesty Zapus (le Zapus vif)	13 avril 2017	Janvier 2018

Figure 7: les versions stables d'Ubuntu

Reference : <https://doc.ubuntu-fr.org/versions>

Linux Mint 17.3 LTS "Rosa" :

Sortie en version stable le 4 décembre 2015.

Reference : https://fr.wikipedia.org/wiki/Versions_de_Linux_Mint

2. Installation de Docker :

Pour installer Docker, vous avez besoin de la version 64 bits de l'une de ces versions Ubuntu:

Yakkety 16.10

Xenial 16.04 (LTS)

Trust 14.04 (LTS)

Dans notre cas pour Ubuntu :

a. Mettre en place le référentiel

Mettre en place le dépôt CE Docker sur Ubuntu. Le `lsb_release` de sous-empreintes commande le nom de votre version d'Ubuntu, comme xenial ou de confiance.

```
sudo apt-get -y install \
apt-transport-https \
ca-certificates \
boucle
pelotonner -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
« Deb [arch = amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -CS) \
stable »
sudo apt-get update
```

b. Obtenir Docker CE

Installer la dernière version de Docker CE sur Ubuntu:

```
sudo apt-get -y install-ce docker
```


c. Tester votre installation Docker CE

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
\$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
<https://cloud.docker.com/>

For more examples and ideas, visit:
<https://docs.docker.com/engine/userguide/>

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # █
```

Docker est installé et en cours d'exécution. Vous devez utiliser sudo pour exécuter les commandes Docker.

Pour mettre à niveau Docker, lancez la commande

```
$ Sudo apt-get upgrade docker-ce
```

- **Note:**

«Désinstaller Docker»

si vous voulez désinstaller docker il faut suivre c'est commande :

```
$Sudo apt get remove --purge docker-ce
```

Les images, les conteneurs, les volumes ou les fichiers de configuration personnalisés sur votre hôte ne sont pas automatiquement supprimés. Pour supprimer toutes les images, conteneurs et volumes:

```
$ Sudo rm -rf /var / lib / docker
```

- **utilisation de docker :**

Quelque commande de base pour la manipulation de docker :

* pour commencer il faut démarrer le service docker :

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # service docker start
iheb-Compaq-Presario-A900-Notebook-PC iheb # █
```

*pour voir la version de votre docker :

```
iheb@iheb-Compaq-Presario-A900-Notebook-PC ~ $ su --
Password:
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker version
Client:
 Version:      17.03.1-ce
 API version:  1.27
 Go version:   gol.7.5
 Git commit:   c6d412e
 Built:        Mon Mar 27 17:10:36 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.03.1-ce
 API version:  1.27 (minimum version 1.12)
 Go version:   gol.7.5
 Git commit:   c6d412e
 Built:        Mon Mar 27 17:10:36 2017
 OS/Arch:      linux/amd64
 Experimental: false
iheb-Compaq-Presario-A900-Notebook-PC iheb # █
```

*pour avoir plus d'information sur votre docker :

Error response from daemon: The Dockerfile (Dockerfile) cannot be empty

iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker info

```
Containers: 19
  Running: 7
  Paused: 0
  Stopped: 12
Images: 10
Server Version: 17.03.1-ce
Storage Driver: aufs
  Root Dir: /var/lib/docker/aufs
  Backing Filesystem: extfs
  Dirs: 168
  Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
  Volume: local
  Network: bridge host macvlan null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 4ab9917febca54791c5f071a9d1f404867857fcc
runc version: 54296cf40ad8143b62dbcaald90e520a2136ddfe
init version: 949e6fa
Security Options:
  apparmor
Kernel Version: 4.4.0-78-generic
Operating System: Linux Mint 18.1
OSType: linux
Architecture: x86_64
CPUs: 2
Total Memory: 1.944 GiB
Name: iheb-Compaq-Presario-A900-Notebook-PC
ID: WQH6:AF02:GXY2:30EW:U752:WNRG:KC5C:XBUG:NKGY:BIFT:UT3X:TOFN
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Username: ihebbf
Registry: https://index.docker.io/v1/
WARNING: No swap limit support
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false
```

*comment avoir un conteneur exécutable dans quelques secondes :

Description : on a commencé par chercher l'image d'Ubuntu avec la commande « sudo docker search ubuntu » puis on télécharge une image de résultat obtenu suite a la recherche avec la commande « sudo docker pull 'nom_image_choisie' ».

A la fin on la lance soit par la commande « sudo docker run 'nom_image_choisie' »

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker search ubuntu
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
ubuntu	Ubuntu is a Debian-based Linux operating s...	6036	[OK]	
rastasheep/ubuntu-sshd	Dockerized SSH service, built on top of of...	86		[OK]
ubuntu-upstart	Upstart is an event-based replacement for ...	73	[OK]	
ubuntu-debootstrap	debootstrap --variant=minbase --components...	30	[OK]	
torusware/speedus-ubuntu	Always updated official Ubuntu docker imag...	27		[OK]
nuagebec/ubuntu	Simple always updated Ubuntu docker images...	20		[OK]
nickistre/ubuntu-lamp	LAMP server on Ubuntu	17		[OK]
solita/ubuntu-systemd	Ubuntu + systemd	8		[OK]
nimmis/ubuntu	This is a docker images different LTS vers...	7		[OK]
darksheer/ubuntu	Base Ubuntu Image -- Updated hourly	2		[OK]
vcatechnology/ubuntu	A Ubuntu image that is updated daily	1		[OK]
webhippie/ubuntu	Docker images for ubuntu	1		[OK]
jordi/ubuntu	Ubuntu Base Image	1		[OK]
admiringworm/ubuntu	Base ubuntu images based on the official u...	1		[OK]
forumi0721ubuntuarhf/ubuntu-armhf-dev	ubuntu-armhf-dev	0		[OK]
forumi0721ubuntuaarch64/ubuntu-aarch64-dev	ubuntu-aarch64-dev	0		[OK]
forumi0721ubuntux64/ubuntu-x64-dev	ubuntu-x64-dev	0		[OK]
labengine/ubuntu	Images base ubuntu	0		[OK]
datenbetrieb/ubuntu	custom flavor of the official ubuntu base ...	0		[OK]
forumi0721ubuntux64/ubuntu-x64-dev-armbian	ubuntu-x64-dev-armbian	0		[OK]
teamrock/ubuntu	TeamRock's Ubuntu image configured with AW...	0		[OK]
forumi0721ubuntux64/ubuntu-x64-dev-android	ubuntu-x64-dev-android	0		[OK]
lynxtp/ubuntu	https://github.com/lynxtp/docker-ubuntu	0		[OK]
konstruktoid/ubuntu	Ubuntu base image	0		[OK]
esycat/ubuntu	Ubuntu LTS	0		[OK]

```
iheb-Compaq-Presario-A900-Notebook-PC iheb #
```

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker pull ubuntu
```

```
Using default tag: latest
```

```
latest: Pulling from library/ubuntu
```

```
b6f892c0043b: Pull complete
```

```
55010f332b04: Pull complete
```

```
2955fb827c94: Pull complete
```

```
3deef3fcbd30: Pull complete
```

```
cf9722e506aa: Pull complete
```

```
Digest: sha256:382452f82a8bbd34443b2c727650af46aced0f94a44463c62a9848133ecb1aa8
```

```
Status: Downloaded newer image for ubuntu:latest
```

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker run -it --name ihebubuntu --rm ubuntu bash
```

```
root@19409afc9d90:/#
```

*pour tester la connectivité d'un conteneur on peut utiliser la commande Ping comme le montre cette figure :

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker exec -it 812210f761c bash
```

```
bash-4.3# ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:07
          inet addr:172.17.0.7  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:7%32741/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8852 (8.6 KiB)  TX bytes:648 (648.0 B)
```

```
lo
```

```
Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1%32741/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
bash-4.3# route -n
```

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	172.17.0.1	0.0.0.0	UG	0	0		eth0
172.17.0.0	0.0.0.0	255.255.0.0	U	0	0		eth0

```
bash-4.3# ping 172.17.0.1
```

```
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.142 ms
64 bytes from 172.17.0.1: seq=1 ttl=64 time=0.224 ms
64 bytes from 172.17.0.1: seq=2 ttl=64 time=0.175 ms
64 bytes from 172.17.0.1: seq=3 ttl=64 time=0.145 ms
64 bytes from 172.17.0.1: seq=4 ttl=64 time=0.175 ms
64 bytes from 172.17.0.1: seq=5 ttl=64 time=0.164 ms
64 bytes from 172.17.0.1: seq=6 ttl=64 time=0.155 ms
64 bytes from 172.17.0.1: seq=7 ttl=64 time=0.148 ms
64 bytes from 172.17.0.1: seq=8 ttl=64 time=0.192 ms
64 bytes from 172.17.0.1: seq=9 ttl=64 time=0.154 ms
```

```
^C
```

```
--- 172.17.0.1 ping statistics ---
```

```
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.142/0.167/0.224 ms
```

```
bash-4.3#
```

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide

iheb@iheb-Compaq-Presario-A900-Notebook-PC ~ $ ping 172.17.0.7
PING 172.17.0.7 (172.17.0.7) 56(84) bytes of data:
64 bytes from 172.17.0.7: icmp_seq=1 ttl=64 time=0.199 ms
64 bytes from 172.17.0.7: icmp_seq=2 ttl=64 time=0.109 ms
64 bytes from 172.17.0.7: icmp_seq=3 ttl=64 time=0.103 ms
64 bytes from 172.17.0.7: icmp_seq=4 ttl=64 time=0.119 ms
64 bytes from 172.17.0.7: icmp_seq=5 ttl=64 time=0.089 ms
64 bytes from 172.17.0.7: icmp_seq=6 ttl=64 time=0.100 ms
64 bytes from 172.17.0.7: icmp_seq=7 ttl=64 time=0.111 ms
64 bytes from 172.17.0.7: icmp_seq=8 ttl=64 time=0.095 ms
^C
--- 172.17.0.7 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 6997ms
rtt min/avg/max/mdev = 0.089/0.115/0.199/0.034 ms
iheb@iheb-Compaq-Presario-A900-Notebook-PC ~ $ ifconfig
br-4a33bd010852 Link encap:Ethernet  HWaddr 02:42:f9:a7:29:7f
          inet addr:172.19.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:f9ff:fea7:297f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1608 (1.6 KB)  TX bytes:15652 (15.6 KB)

br-aff06928aa92 Link encap:Ethernet  HWaddr 02:42:1f:47:f4:67
          inet addr:172.18.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

docker0  Link encap:Ethernet  HWaddr 02:42:6b:e8:0f:22
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:6bff:fe08:f22/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:85 errors:0 dropped:0 overruns:0 frame:0
          TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

*pour lister les conteneurs :

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3150a0429b4d	wordpress	"docker-entrypoint..."	2 weeks ago	Exited (0) 2 weeks ago		wordpresscompose_wordpr
ess_1						
b3b0b88f9597	mariadb	"docker-entrypoint..."	2 weeks ago	Exited (0) 2 weeks ago		wordpresscompose_mariad
b_1						
93bf21ca2eb5	wordpress	"docker-entrypoint..."	2 weeks ago	Exited (1) 2 weeks ago		projet_wordpress_1
c569924307c0	corbinu/docker-phpmyadmin	"/bin/sh -c phpmya..."	2 weeks ago	Exited (137) 2 weeks ago		projet_phpmyadmin_1
6f885f9ebc75	mariadb	"docker-entrypoint..."	2 weeks ago	Exited (137) 2 weeks ago		projet_wordpress_db_1
4070dbc28c24	mysql:5.7	"docker-entrypoint..."	2 weeks ago	Up 6 hours	3306/tcp	projet_db_1
54a8b01e2df5	portainer/portainer	"/portainer -H tcp..."	4 weeks ago	Exited (255) 4 weeks ago	0.0.0.0:8000->8000/tcp, 9000/tcp	jovial_visvesvaraya
5e3f8e17764d	portainer/portainer	"/portainer -H tcp..."	4 weeks ago	Created		elegant_nobel
a4566d833e24	portainer/portainer	"/portainer -H tcp..."	4 weeks ago	Created		nostalgic_morse
efbceable074	portainer/portainer	"/portainer"	4 weeks ago	Exited (255) 4 weeks ago	0.0.0.0:9000->9000/tcp	condescending_franklin

```
iheb-Compaq-Presario-A900-Notebook-PC iheb #
```

*pour lister les images:

```
wahbi@wahbi-u:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	ebcd9d4fca80	8 days ago	118 MB
prestashop/prestashop	latest	08ffae20032a	11 days ago	915 MB
mysql	latest	9e64176cd8a2	4 weeks ago	407 MB
hello-world	latest	48b5124b2768	4 months ago	1.84 kB

*pour lister les réseaux :

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker network ls
```

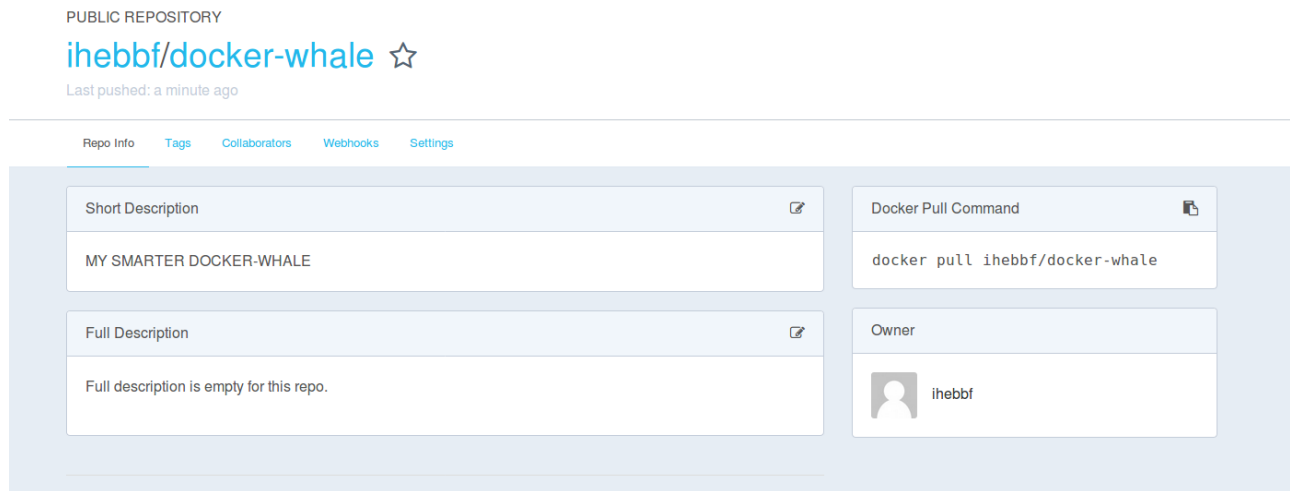
NETWORK ID	NAME	DRIVER	SCOPE
44ca356ea10e	bridge	bridge	local
aff06928aa92	dockerwordpressworkflowdemo_back	bridge	local
532f6714c282	host	host	local
f5dfcf4a46a8	none	null	local
4a33bd010852	projet_default	bridge	local

*on peut accéder au docker hub à partir d'un terminal pour publier une nouvelle image en utilisant docker push :

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: ihebbf
Password:
Login Succeeded
iheb-Compaq-Presario-A900-Notebook-PC iheb #
```

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker push ihebbf/docker-whale
The push refers to a repository [docker.io/ihebbf/docker-whale]
3b043ea18618: Layer already exists
5f70bf18a086: Layer already exists
d061ee1340ec: Layer already exists
d511ed9e12e1: Layer already exists
091abc5148e4: Layer already exists
b26122d57afa: Layer already exists
37ee47034d9b: Layer already exists
528c8710fd95: Layer already exists
1154ba695078: Layer already exists
latest: digest: sha256:55082a41283e9ef509e9ca326d408d962c1b98571f673b330690bb9c9df2cfc4c size: 2614
iheb-Compaq-Presario-A900-Notebook-PC iheb #
```


Après la publication de la nouvelle l'image on consulte le compte docker Hub pour vérifier si l'image est vraiment publiée :



*on peut aussi inspecter le contenu d'un conteneur en utilisant la commande « sudo docker inspect ID_conteneur ».

Quand on inspecte un conteneur il nous affiche tout les informations relative a cette conteneur, image de base, sa date de création, son état, son chemin, etc....

```
wahbi@wahbi-u: /var/www/html$ sudo docker inspect 96348cf51bf8
[
  {
    "Id": "96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb",
    "Created": "2017-05-30T23:47:51.648525821Z",
    "Path": "bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 8616,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2017-05-30T23:47:55.940341378Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:ebcd9d4fca80e9e8afc525d8a38e7c56825dfb4a220ed77156f9fb13b14d4ab7",
    "ResolvConfPath": "/var/lib/docker/containers/96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb/hostname",
    "HostsPath": "/var/lib/docker/containers/96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb/hosts",
    "LogPath": "/var/lib/docker/containers/96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb/96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb-json.log",
    "Name": "/uuuuu",
    "RestartCount": 0,
    "Driver": "aufs",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      }
    }
  },
  ...
]
```

```

],
"Isolation": "",
"CpuShares": 0,
"Memory": 0,
"NanoCpus": 0,
"CgroupParent": "",
"BlkioWeight": 0,
"BlkioWeightDevice": null,
"BlkioDeviceReadBps": null,
"BlkioDeviceWriteBps": null,
"BlkioDeviceReadIOps": null,
"BlkioDeviceWriteIOps": null,
"CpuPeriod": 0,
"CpuQuota": 0,
"CpuRealtimePeriod": 0,
"CpuRealtimeRuntime": 0,
"CpusetCpus": "",
"CpusetMems": "",
"Devices": [],
"DiskQuota": 0,
"KernelMemory": 0,
"MemoryReservation": 0,
"MemorySwap": 0,
"MemorySwappiness": -1,
"OomKillDisable": false,
"PidsLimit": 0,
"Ulimits": null,
"CpuCount": 0,
"CpuPercent": 0,
"IOMaximumIOps": 0,
"IOMaximumBandwidth": 0,
"OOMKillDisable": false,
"ShmSize": 67108864,
"Runtime": "runc",
"ConsoleSize": [
  0,
  0
],

```

```

},
"GraphDriver": {
  "Name": "aufs",
  "Data": null
},
"Mounts": [],
"Config": {
  "Hostname": "96348cf51bf8",
  "Domainname": "",
  "User": "",
  "AttachStdin": true,
  "AttachStdout": true,
  "AttachStderr": true,
  "Tty": true,
  "OpenStdin": true,
  "StdinOnce": true,
  "Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  ],
  "Cmd": [
    "bash"
  ],
  "Image": "ubuntu",
  "Volumes": null,
  "WorkingDir": "",
  "Entrypoint": null,
  "OnBuild": null,
  "Labels": {}
}

```

```

},
"NetworkSettings": {
  "Bridge": "",
  "SandboxID": "17730a36128d55d7fb7c1ed2678dd6c958aa311c175b5d1da7e24e5bd37231fa",
  "HairpinMode": false,
  "LinkLocalIPv6Address": "",
  "LinkLocalIPv6PrefixLen": 0,
  "Ports": {},
  "SandboxKey": "/var/run/docker/netns/17730a36128d",
  "SecondaryIPAddresses": null,
  "SecondaryIPv6Addresses": null,
  "EndpointID": "bb65809dc619b6a7ea43c84d0e95799b01342e04b8f52bb4538066f0bdd40524",
  "Gateway": "172.17.0.1",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "IPAddress": "172.17.0.2",
  "IPPrefixLen": 16,
  "IPv6Gateway": "",
  "MacAddress": "02:42:ac:11:00:02",
  "Networks": {
    "bridge": {
      "IPAMConfig": null,
      "Links": null,
      "Aliases": null,
      "NetworkID": "f0c60e27db3f630265f51aa87e7b975e064725e3d5a9a0ca00a713b4a96a1fcb",
      "EndpointID": "bb65809dc619b6a7ea43c84d0e95799b01342e04b8f52bb4538066f0bdd40524",
      "Gateway": "172.17.0.1",
      "IPAddress": "172.17.0.2",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "MacAddress": "02:42:ac:11:00:02"
    }
  }
}
}
}

```


On peut aussi inspecter le réseau avec la même commande mais en ajoutant <network > syntaxe : <docker network inspect nom-de-réseau> cette commande nous offre des informations sur ce réseau ainsi que les conteneurs attaché a lui.

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "c48ee6c4371172ea8df01e10c41f02a6579a7a55b301b64d93fa155a476a8500",
    "Created": "2017-05-24T10:39:07.109412026+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Containers": {
      "deea866e58eb1625d434eed8d02c5d6b2e2e4b08ee18dd0732119018b420a544": {
        "Name": "boring_shockley",
        "EndpointID": "30cbce6104425501633c1e7d5c81c72997a9b7ca0adfc08da7d114e533ae1e3e",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
```

On peut supprimer un conteneur :

```
root@wahbi-u:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAM
ES						
0e2d54a468fe	hello-world	"/hello"	8 seconds ago	Exited (0) 5 seconds ago		goo
fy_pare						
eddf2b72b0a0	hello-world	"/hello"	37 minutes ago	Exited (0) 37 minutes ago		jol
ly_meitner						
76ce948ead4d	ubuntu	"bash -expose 80 -..."	5 days ago	Exited (2) 5 days ago		bli
ssful_hodgkin						
7be5783c0313	hello-world	"/hello"	6 days ago	Exited (0) 6 days ago		aff C
ectionate_kare						
53474e70d3d2	prestashop/prestashop	"docker-php-entryp..."	10 days ago	Exited (255) 10 days ago	0.0.0.0:8080->80/tcp	son
e-prestashop						
87287610328d	mysql	"docker-entrypoint..."	10 days ago	Exited (255) 10 days ago	3306/tcp	pre
stDB						
26d99af5de49	hello-world	"/hello"	11 days ago	Exited (0) 11 days ago		ped
antic_bardeen						

```
root@wahbi-u:~# docker rm 0e2d54a468fe
0e2d54a468fe
root@wahbi-u:~# |
```

Remarque: un conteneur est en cours d'exécution est basé sur une image dedans, quand on veut supprimer cette image il nous montre une erreur, donc il faut arrêter la conteneur pour qu'on puisse effacer cette image.

On utilise le docker images pour connaitre l'ID de l'image à supprimer, puis on la supprime, a la fin on vérifie si elle existe encore ou pas.

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu               latest              ebcd9d4fca80       8 days ago         118 MB
wordpress            latest              2d81af9ee904       4 weeks ago        401 MB
mariadb              latest              21ba6715f846       4 weeks ago        395 MB
electron2335/ubuntu latest              f7b3f317ec73       4 weeks ago        117 MB
mysql                5.7                d5127813070b       6 weeks ago        407 MB
phpmyadmin/phpmyadmin latest              976dd870f876       6 weeks ago        118 MB
portainer/portainer latest              89883cee365b       6 weeks ago        9.96 MB
ihebbf/docker-whale  latest              9aa505c30236       2 months ago       275 MB
wordpress            4.7.1              dfbefe759772       3 months ago       400 MB
venkatjee93/alpine_hadoop latest              bf45c0f92c95       6 months ago       453 MB
corbinu/docker-phpmyadmin latest              5c663962b799       14 months ago      473 MB
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker rmi ebcd9d4fca80
Untagged: ubuntu:latest
Deleted: sha256:ebcd9d4fca80e9e8afc525d8a38e7c56825dfb4a220ed77156f9fb13b14d4ab7
Deleted: sha256:ef5b99eed7c2ed19ef39f72ac19bb66e16ed6c0868053daae60306a73858fbd4
Deleted: sha256:257e51479af1e9d2e0c9b958e68f6b992329904df24d81efa191cef515a9bf8b
Deleted: sha256:6e1d2d371500e2fe6df75f5755d0b9f2a3b69a42fe88100d514212bba7ad23f
Deleted: sha256:afa9e7a5e3f3b006942d128c562a3273947c7ab50cdac33fea7213890072a5b6
Deleted: sha256:2df9b8def18a090592bf1cbd1079e1ac2274435c53f027ee5ce0a8faaa5d6d4b
iheb-Compaq-Presario-A900-Notebook-PC iheb # docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
wordpress            latest              2d81af9ee904       4 weeks ago        401 MB
mariadb              latest              21ba6715f846       4 weeks ago        395 MB
electron2335/ubuntu latest              f7b3f317ec73       4 weeks ago        117 MB
mysql                5.7                d5127813070b       6 weeks ago        407 MB
phpmyadmin/phpmyadmin latest              976dd870f876       6 weeks ago        118 MB
portainer/portainer latest              89883cee365b       6 weeks ago        9.96 MB
ihebbf/docker-whale  latest              9aa505c30236       2 months ago       275 MB
wordpress            4.7.1              dfbefe759772       3 months ago       400 MB
venkatjee93/alpine_hadoop latest              bf45c0f92c95       6 months ago       453 MB
corbinu/docker-phpmyadmin latest              5c663962b799       14 months ago      473 MB
iheb-Compaq-Presario-A900-Notebook-PC iheb #
```

on va pas tout lister ! mais il existe plein de commandes pour manipuler docker.

En fin il faut arrêter le service docker :

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # service docker stop
```

*Depuis un terminal, si vous exécutez la commande docker, vous obtiendrez une liste de commandes exécutables :

\$ Docker

Commande:

Attach : joindre à un conteneur courant

Build : Créez une image à partir d'un fichier Docker

Commit : Créer une nouvelle image à partir des modifications d'un conteneur

CP : Copie les fichiers / dossiers du système de fichiers d'un conteneur sur le chemin hôte

Create : Créer un nouveau conteneur

Diff : Inspectez les modifications dans le système de fichiers d'un conteneur

Events : obtenez des événements en temps réel à partir du serveur

Exec : Exécutez une commande dans un conteneur existant

Export : Transférer le contenu d'un conteneur comme archive tar

History : affiche l'historique d'une image

Images: liste d'images

Import : Créez une nouvelle image de système de fichiers à partir des contenus d'un tarball

Info : afficher l'information sur l'ensemble du système

Inspect : renvoyer des informations de bas niveau sur un conteneur

Kill: tuer un conteneur

Load : Charge une image à partir d'une archive tar

Login : Enregistrez-vous ou connectez-vous à un serveur de registre Docker

Logout : déconnectez-vous d'un serveur de registre Docker

Logs : récupérer les journaux d'un conteneur

Port : recherchez le port public qui est NAT-ed à PRIVATE_PORT

Pause: arrête tous les processus dans un conteneur

Ps : liste des conteneurs

Pull : tirez une image ou un dépôt d'un serveur de registre Docker

Push : appuyez sur une image ou un dépôt vers un serveur de registre Docker

Restart : redémarrez un conteneur

RM : enlevez un ou plusieurs récipients

RMI : Supprime une ou plusieurs images

Run : exécuter une commande dans un nouveau conteneur

Save : Enregistrez une image dans une archive tar

Search : recherchez une image sur le Hub Docker

START : Démarrer un conteneur arrêté

Stop : Arrêtez un conteneur en cours d'exécution

Tag : marquer une image dans un référentiel

Top : recherche les processus de fonctionnement d'un conteneur

Unpause : ne pas répartir un récipient en pause

Version: affiche les informations de la version Docker

Wait : Bloquez jusqu'à ce qu'un conteneur s'arrête, puis imprimez son code de sortie

Et voilà, avec cette liste, vous avez tout en main pour gérer vos conteneurs et vos images.

- **Le fichier docker file**

Docker peut créer des images automatiquement en lisant les instructions d'un Dockerfile.

Un Dockerfile est un document texte qui contient toutes les commandes qu'un utilisateur peut faire appel à la ligne de commande pour assembler une image.

En utilisant docker les utilisateurs peut créer une génération automatisée qui exécute plusieurs instructions de ligne de commande successivement.

Voila un exemple d'un docker file :

```

iheb@iheb-Compaq-Presario-A900-Notebook-PC ~ $ su --
Password:
iheb-Compaq-Presario-A900-Notebook-PC iheb # cd mydockerbuild/
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # touch dockerfile
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # pluma dockerfile

(pluma:8809): EggSMClient-WARNING **: Failed to connect to the session manager: None of the auth

(pluma:8809): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)

(pluma:8809): dconf-WARNING **: failed to commit changes to dconf: The connection is closed

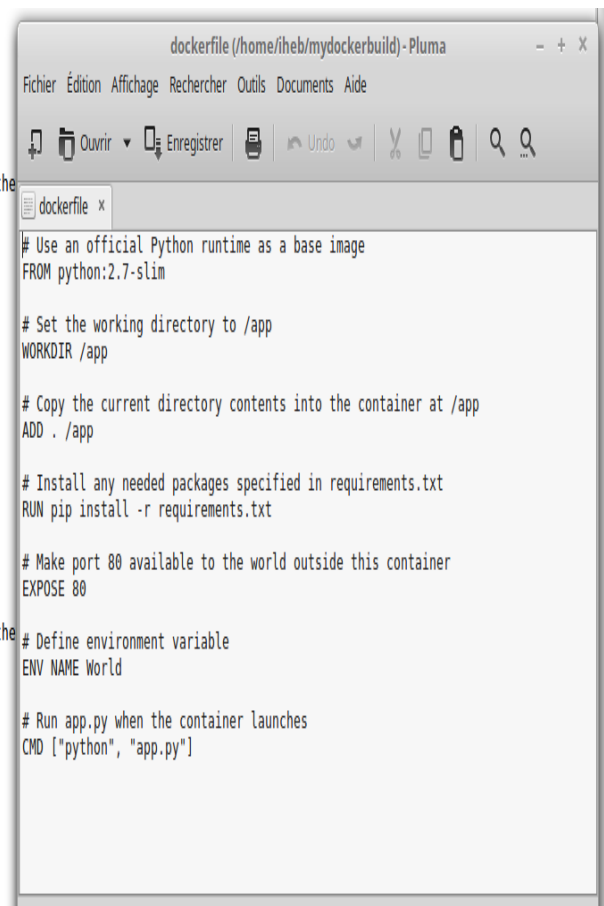
(pluma:8809): dconf-WARNING **: failed to commit changes to dconf: The connection is closed

(pluma:8809): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # pluma

(pluma:8942): EggSMClient-WARNING **: Failed to connect to the session manager: None of the auth

(pluma:8942): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # pluma dockerfile

```



- **Docker build :**

permet de construire une image à partir d'un dockerfile

Les figures Ci-dessous expliquent l'utilisation des commandes docker pour construire une image en utilisant le docker file ainsi l'attribution des ports. Voila un exemple qui montre l'utilisation de Docker build.

Image : « friendlyhello »

```

iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # ls
app.py dockerfile requirements.txt
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker build -t friendlyhello .
Sending build context to Docker daemon 4.608 kB
Step 1/7 : FROM python:2.7-slim
2.7-slim: Pulling from library/python
10a267c67f42: Pull complete
f68a39a6a5e4: Pull complete
9beaffc0cf19: Pull complete
3c1fe835fb6b: Pull complete
Digest: sha256:fdee3139831a8e66543effffdc5addf73c7b18b66c575bd030f9f6f3a7052a6b
Status: Downloaded newer image for python:2.7-slim
--> 1c7128a655f6
Step 2/7 : WORKDIR /app
--> cea0b3c5e2b3
Removing intermediate container a6944f34f49a
Step 3/7 : ADD . /app
--> bb8f98d0272d
Removing intermediate container alf79778a67e
Step 4/7 : RUN pip install -r requirements.txt
--> Running in 2b98d4d5dd9c
Collecting Flask (from -r requirements.txt (line 1))
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting Redis (from -r requirements.txt (line 2))
  Downloading redis-2.10.5-py2.py3-none-any.whl (60kB)
Collecting itsdangerous>=0.21 (from Flask->-r requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2>=2.4 (from Flask->-r requirements.txt (line 1))
  Downloading Jinja2-2.9.6-py2.py3-none-any.whl (340kB)
Collecting Werkzeug>=0.7 (from Flask->-r requirements.txt (line 1))
  Downloading Werkzeug-0.12.2-py2.py3-none-any.whl (312kB)
Collecting click>=2.0 (from Flask->-r requirements.txt (line 1))
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->Flask->-r requirements.txt (line 1))
  Downloading MarkupSafe-1.0.tar.gz
Building wheels for collected packages: itsdangerous, MarkupSafe
Running setup.py bdist_wheel for itsdangerous: started
Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
Stored in directory: /root/.cache/pip/wheels/fc/a8/66/24d655233c757e178d45dea2de22a04c6d92766abfb741129a
Running setup.py bdist_wheel for MarkupSafe: started
Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
Stored in directory: /root/.cache/pip/wheels/88/a7/30/e39a54a87bcbe25308fa3ca64e8ddc75d9b3e5afa21ee32d57

Successfully built itsdangerous MarkupSafe
Installing collected packages: itsdangerous, MarkupSafe, Jinja2, Werkzeug, click, Flask, Redis
Successfully installed Flask-0.12.2 Jinja2-2.9.6 MarkupSafe-1.0 Redis-2.10.5 Werkzeug-0.12.2 click-6.7 itsdangerous-0.24
--> af30205843e4
Removing intermediate container 2b98d4d5dd9c
Step 5/7 : EXPOSE 80
--> Running in d48d960c5a7a
--> e2ea8f956393
Removing intermediate container d48d960c5a7a
Step 6/7 : ENV NAME World
--> Running in 3da1261f9827
--> 349740bcebfb
Removing intermediate container 3da1261f9827
Step 7/7 : CMD python app.py
--> Running in 28616f480554
--> cf83616b3de6
Removing intermediate container 28616f480554
Successfully built cf83616b3de6

```

```
iheb-Compaq-Presario-A900-Notebook-PC iheb # cd mydockerbuild/
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # touch Dockerfile
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # pluma Dockerfile
```

```
(pluma:8927): EggSMClient-WARNING **: Failed to connect to the session manager: None of the authentication protocols specified are supported
```

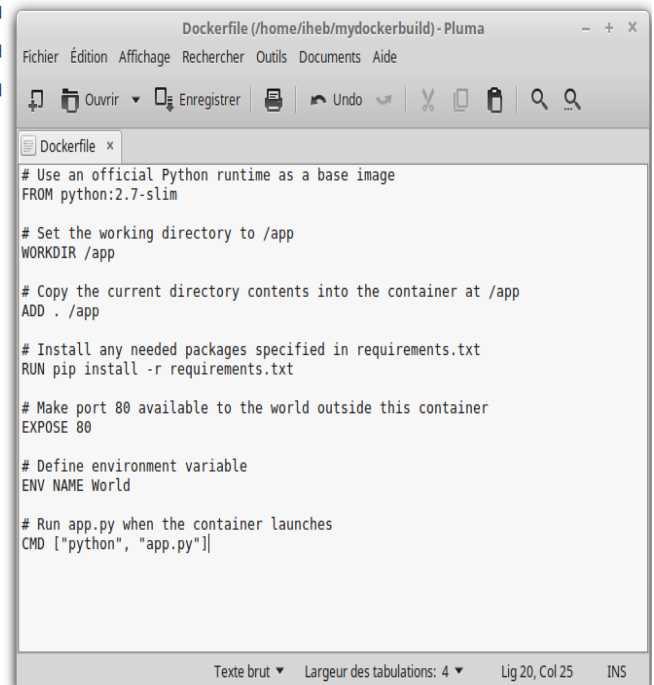
```
(pluma:8927): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
Error creating proxy: The connection is closed (g-io-error-quark, 18)
```

```
(pluma:8927): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
```

```
(pluma:8927): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
```

```
(pluma:8927): dconf-WARNING **: failed to commit changes to dconf: The connection is closed
```

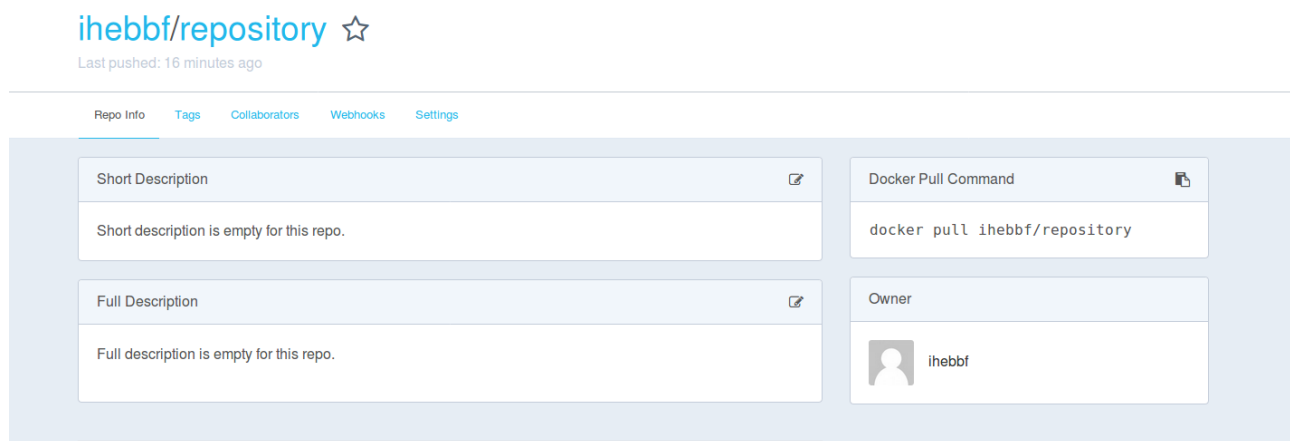
```
□
```



on utilise `<docker push>` pour publier cette image sur le Docker Hub après avoir accès avec `<docker login>` (authentification) .

```
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker stop 1fa4ab2cf395
Error response from daemon: No such container: 1fa4ab2cf395
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker stop cc5f843cf598
cc5f843cf598
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username (ihebbf):
Password:
Login Succeeded
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker tag friendlyhello username/repository:tag
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker tag friendlyhello ihebbf/repository:tag
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker push ihebbf/repository:tag
The push refers to a repository [docker.io/ihebbf/repository]
a9b57a5d4201: Pushed
b215c7a8dbe8: Pushed
da7ca770c2c4: Pushed
7b7f69d6236f: Mounted from library/python
667e68ed0db3: Mounted from library/python
5eac2de68a97: Mounted from library/python
8d4d1ab5ff74: Mounted from library/python
tag: digest: sha256:b0f2e3491656777da14be472d4ada869e4084e96b571fc1b7b3e28b3a18e727 size: 1787
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # █
```


On consulte le compte Docker Hub sur le browser :



On lance un conteneur avec l'image friendlyhello sur le port 4000 :

```
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker run -p 4000:80 friendlyhello
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```

On peut voir avec la commande <docker ps> si le conteneur exécute avec l'image demandée :

```
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker ps -a
```

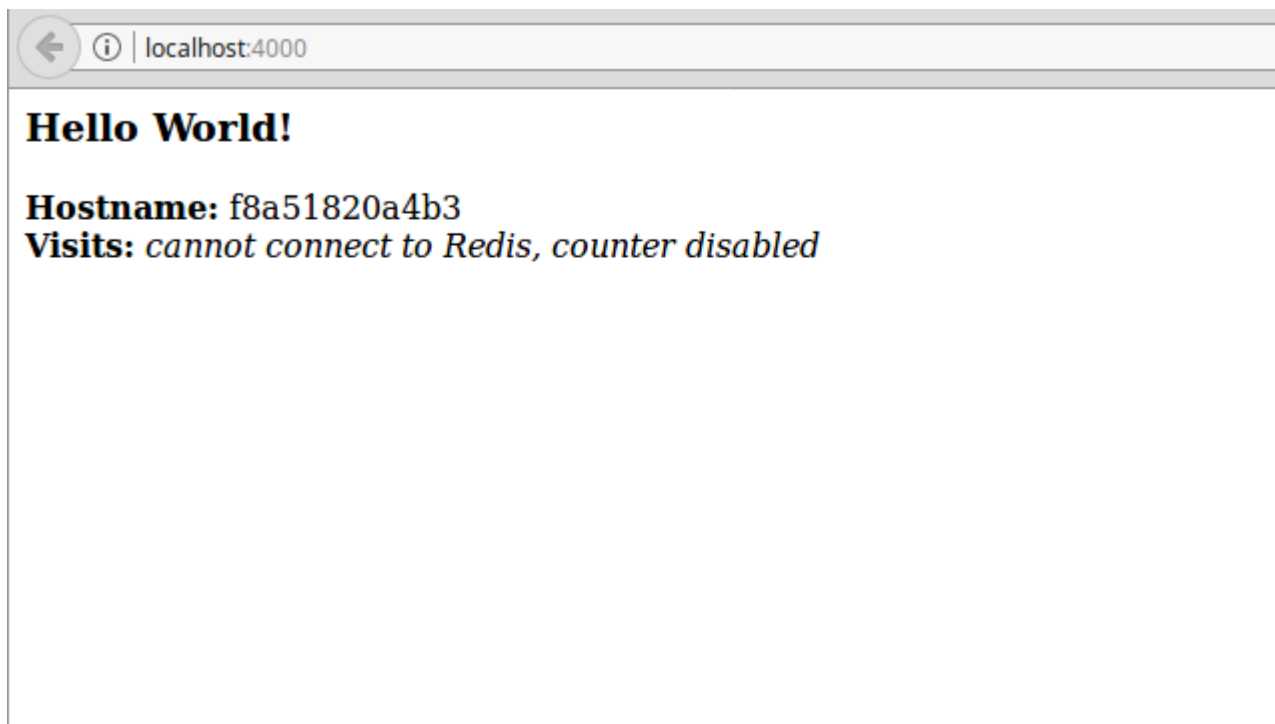
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
cc5f843cf598	friendlyhello	"python app.py"	About a minute ago	Exited (0) 31 seconds ago	
012210f76e1c	unanktiao03/alpine:latest	"bash"	2 days ago	Exited (137) 41 hours ago	

On trouve l'image 'friendlyhello' on listant les images :

```
iheb-Compaq-Presario-A900-Notebook-PC mydockerbuild # docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
friendlyhello	latest	cf83616b3de6	5 minutes ago	195 MB

A la fin on consulte le local host sur le port déclarer(4000) et voila la résultat :



Cette figure représente les principales commandes de manipulation des conteneurs

```
docker build -t friendlyname . # Create image using this directory's Dockerfile
docker run -p 4000:80 friendlyname # Run "friendlyname" mapping port 4000 to 80
docker run -d -p 4000:80 friendlyname # Same thing, but in detached mode
docker ps # See a list of all running containers
docker stop <hash> # Gracefully stop the specified container
docker ps -a # See a list of all containers, even the ones not running
docker kill <hash> # Force shutdown of the specified container
docker rm <hash> # Remove the specified container from this machine
docker rm $(docker ps -a -q) # Remove all containers from this machine
docker images -a # Show all images on this machine
docker rmi <imagename> # Remove the specified image from this machine
docker rmi $(docker images -q) # Remove all images from this machine
docker login # Log in this CLI session using your Docker credentials
docker tag <image> username/repository:tag # Tag <image> for upload to registry
docker push username/repository:tag # Upload tagged image to registry
docker run username/repository:tag # Run image from a registry
```

- **Docker compose :**

Docker Compose est un outil pour définir et exécuter des applications Docker multi-conteneurs. Avec Compose, vous utilisez un fichier Compose.yml pour configurer les services de votre application. Ensuite, en utilisant une seule commande, vous créez et

démarrez tous les services à partir de votre fichier .

L'utilisation de docker Compose est essentiellement un processus en trois étapes.

Définissez l'environnement de votre application avec un Dockerfile afin qu'il puisse être reproduit n'importe où.

Définissez les services qui composent votre application dans docker—compose.yml afin qu'ils puissent être exécutés ensemble dans un environnement isolé.

Enfin, exécutez docker-compose up et Compose va démarrer et exécuter votre application entière.

Voila un exemple pour un fichier docker-compose.yml :

```
version: '2'
services:
  apache:
    depends_on:
      - db
    image: eboraas/apache
    restart: always
    volumes:
      - /var/html:/var/www/html/
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: p4ssw0rd!
    ports:
      - 80:80 # Expose http and https
      - 443:443
    networks:
      - back
  db:
    image: mysql:5.7
    restart: always
    volumes:
      - /var/lib/mysql:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: p4ssw0rd!
    networks:
      - back
  phpmyadmin:|
    depends_on:
      - db
    image: phpmyadmin/phpmyadmin
    restart: always
    ports:
      - 8080:80
    environment:
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: p4ssw0rd!
    networks:
      - back
networks:
  back:
```

V. conclusion:

Pour conclure, ce deuxième chapitre permet de décrire le passage d'une ancienne méthode de virtualisation avec les VM et décrit la nouvelle méthode de conteneurisation en utilisant le logiciel docker .ce dernier a plusieurs avantages mais la majeur c'est qu'il facilite le développement et le rend plus rapide comme le montre la figure




Datacenter		Virtualisation		Docker
				
Déploiement dans le mois	⇒	Déploiement dans la minute	⇒	Déploiement dans la seconde
Pendant des années		Pendant des mois		Pendant quelques heures/minutes
Développement en cascade		Agile		DevOps

Figure 8: Evolution du développement avec les conteneurs

Chapitre 3: Proposition et Simulation d'une application de paiement en ligne

I. Introduction:

Dans cette partie, nous allons parler de la façon dont nous allons prouver que la technique des conteneurs peut garantir la sécurité de l'information tant du client que de l'entreprise. Par un autre mot, notre objectif dans ce chapitre est de développer une application utilisée tous les jours Dans le monde réel, et l'implémenter sur cette Nouvelle infrastructure de Cloud computing.

Notre exemple pour ce type d'applications sera une application qui traitera l'une des Choses les plus précieuses pour nos clients, bien sûr, c'est leur argent. Et pour cela, le processus de paiement de bout en bout doit être totalement sécurisé. Bien sûr, développer une telle application n'est pas une chose facile à faire, ce travail nécessite un groupe de personnes qualifiées, de professionnels du développement et de professionnels de la sécurité, la question est de savoir pourquoi notre application doit être sécurisée et ne contient pas d'erreurs et de bugs si nous allons Pour le développer localement et ne pas vraiment le mettre dans le monde réel? En fait, la réponse est très simple, notre principal objectif est de tester le niveau de sécurité de nos conteneurs, alors comment le faire si l'application qui est implémentée n'est pas Sécurisée?

Rappel : Une brève explication des conteneurs :

Une image est un paquet léger, autonome et exécutable qui comprend tout ce qui est nécessaire pour exécuter un logiciel, y compris le code, un temps d'exécution, des bibliothèques, des variables d'environnement et des fichiers de configuration.

Un conteneur est une instance d'exécution d'une image - ce que l'image devient en mémoire lorsqu'il est effectivement exécuté. Il est complètement isolé de l'environnement hôte par défaut, en accédant uniquement aux fichiers hôtes et aux ports s'il est configuré pour le faire.

Les conteneurs exécutent des applications nativement sur le noyau de la machine hôte. Ils ont de meilleures caractéristiques de performance que les machines virtuelles qui n'ont qu'un accès virtuel aux ressources de l'hôte via un hyperviseur.

Les conteneurs peuvent obtenir un accès natif, chacun exécuté dans un processus discret, ne prenant plus de mémoire que tout autre exécutable.

Nous devons également supposer que vous connaissez quelque concept avant de continuer:

- *Adresses IP et Ports

- *Machines virtuelles

- *Modification des fichiers de configuration

- *Connaissance de base des idées de dépendances de code et de construction

- *Les termes d'utilisation des ressources machines, comme les pourcentages de CPU, l'utilisation de RAM en octets, etc.

II. Architecture globale de l'application :

Site :E- Merchant

Site :payement



Page: checkout.php



Page :paiment.php

Comme le montre la figure, notre application est constituée de deux sites web. Le premier est le site E-Merchant qui présente le « front-end », c'est la page qui s'affiche sur le navigateur de client pour consulter les offres possibles.

Quand ce client décide l'achat et confirme son choix, il sera redirigé vers le deuxième site (le service web : paiement.php) pour payer le frais de produit.

A ce niveau, le client choisit la carte bancaire avec laquelle il va payer {Master Card, Visa...}.

Il existe un troisième tiers dans cette application pour faire réellement cette achat en-ligne, c'est la banque. Donc l'argent ici est fictif !

L'étape suivante sert à accéder à la banque de client et faire la transaction vers le compte de site E-Merchant.

On va se limiter au niveau de site paiement. Ce dernier, à chaque requête reçue, envoie une réponse aléatoire à la page de client (succès, refus, échec, solde insuffisant).

- **Architecture plus détaillée de l'application :**

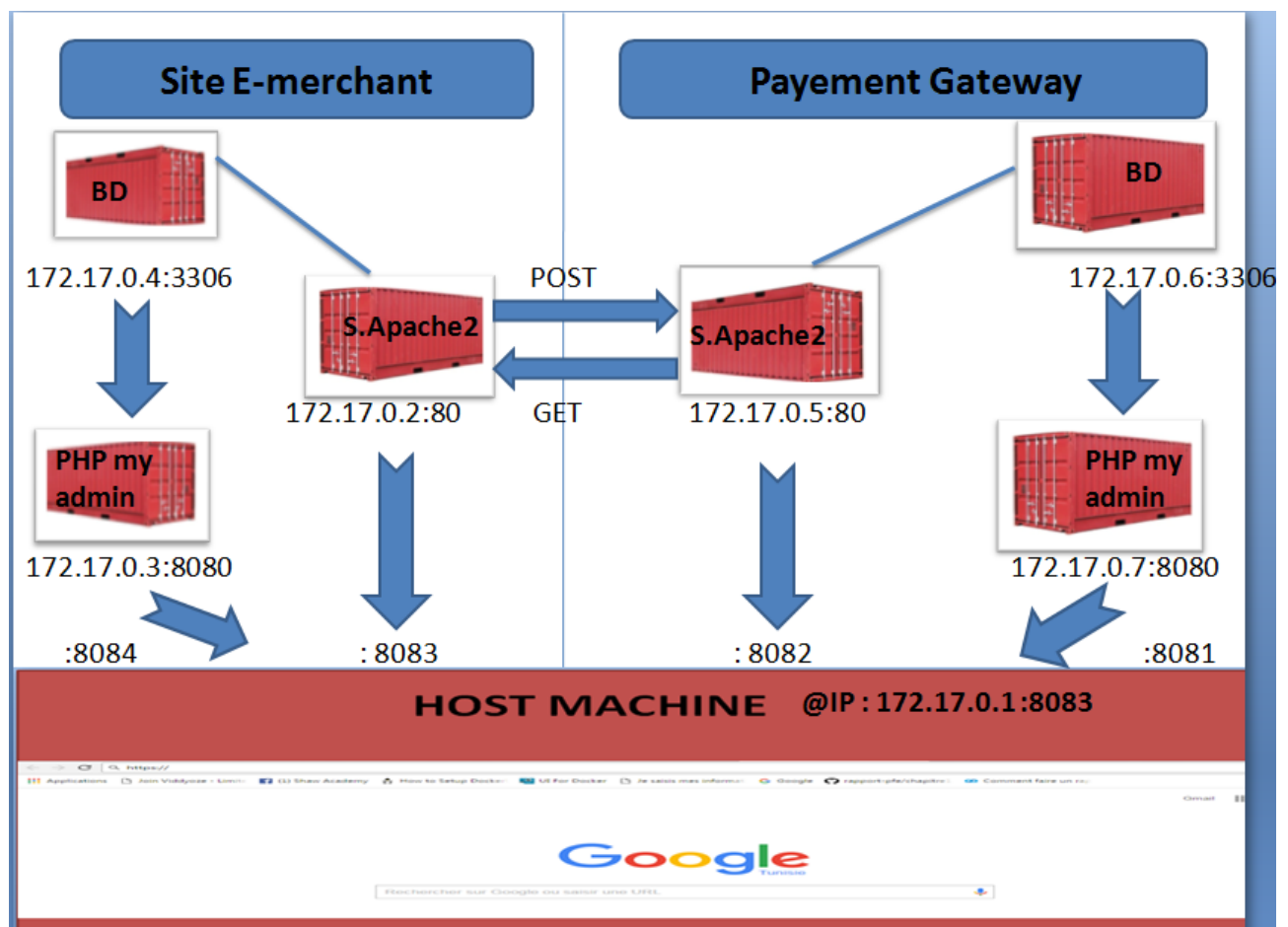


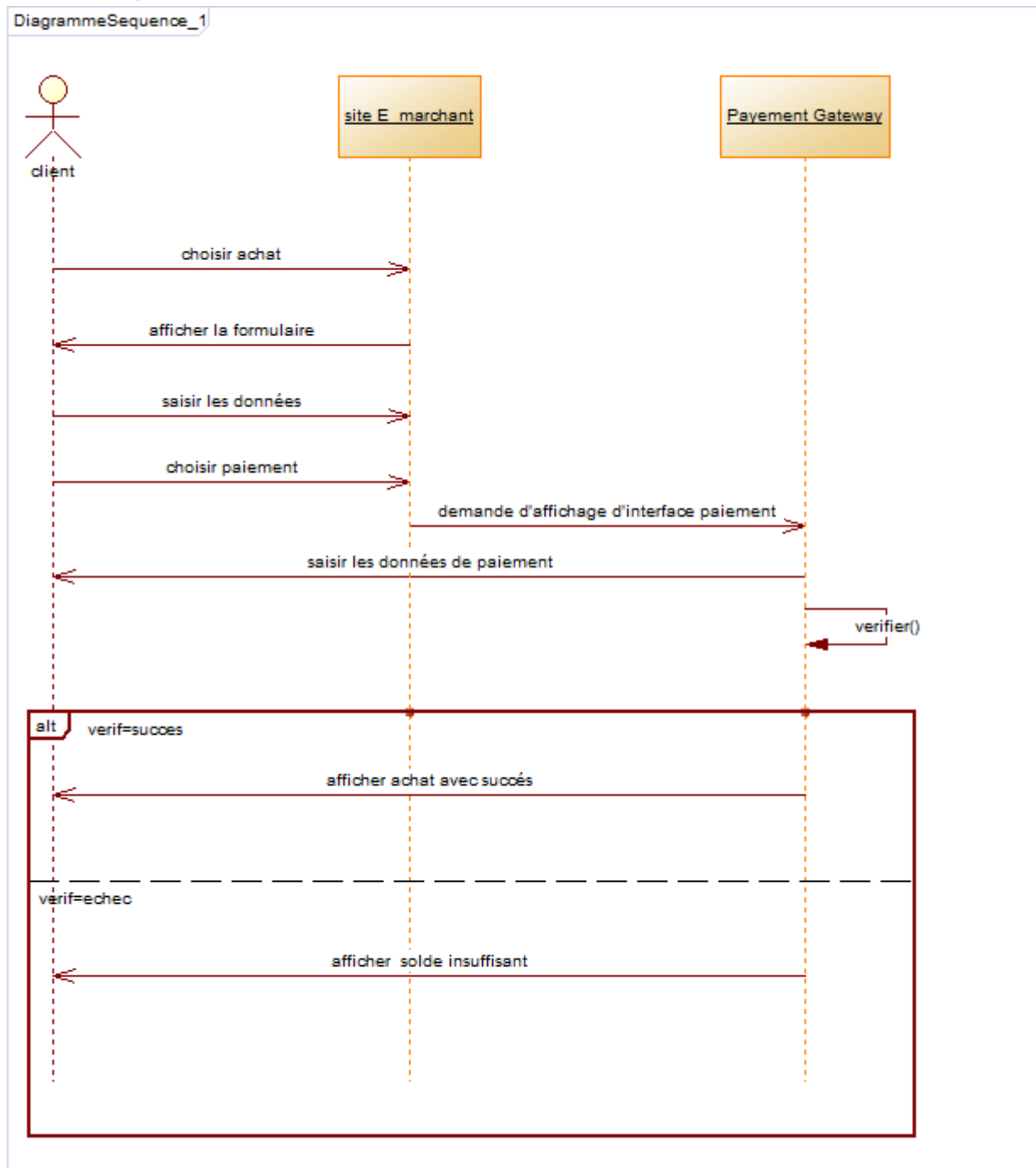
Figure 9: Architecture plus détaillée de l'application

- **Brève description :**

La figure précédente représente la nouvelle architecture de l'application après son implémentation dans les conteneurs, ou les services de base qui construisent notre application sont implémentés chacun dans son propre conteneur qui facilite le développement du code de chaque micro-service.

Et cette conception de l'architecture nous ramène à définir les configurations nécessaires pour la liaison entre les services comme le type de réseaux, les adresses IP, les volumes partagés, le mappage des ports, et la dépendance entre les conteneurs.

- Diagramme :



Chapitre 4: Mise en place des règles de sécurité pour la solution

I. Sécurisation des paiements en lignes et méthodes alternatives de paiement

Comment sécuriser vos paiements en ligne ?

Entre 2010 et 2013, les chiffres démontrent que c'est sur internet que la fraude à la carte bancaire a montré sa plus forte progression.

Même si le taux de fraude relatif aux paiements par carte bancaire est resté stable en 2013, par rapport à 2012 le montant total des sommes concernées a augmenté selon le rapport de l'Observatoire national de la délinquance et des réponses pénales (ONDRP).

En 2013 en France, le taux de fraude s'est maintenu à 0,080% du montant des transactions effectuées, soit 469,9 millions d'euros, contre 450,7 millions en 2012.

Avec le développement du commerce en ligne, les hackers ont très vite compris les faiblesses de la carte de paiement dans son utilisation via internet.

Revenons sur la carte de paiement et les signes sécurité qu'elle contient :

Signes de sécurité d'une carte de paiement >> sous forme de box :



Figure 10: carte de paiement

1. Les signes de sécurité visibles:

i. Le PAN (Primary Account Number) : C'est un numéro (en général 16 chiffres), qui définit le réseau émetteur de la carte (Visa, Mastercard, ...), le type de carte, ainsi que la banque émettrice de cette carte, le tout suivi d'un numéro de séquence qui permet de définir de manière unique chaque numéro de carte.

ii. La date d'expiration de la carte

iii. Le nom du détenteur de la carte

iv. La signature du détenteur de la carte

v. Le cryptogramme visuel (CVV : Card Verification Value) : C'est une valeur de contrôle, calculée sur base de tout ou parti de ces éléments précédents et qui permet de valider leur cohérence.

2. Les signes de sécurité invisibles:

vi. La piste magnétique : Elle est divisée en plusieurs tranches, dont l'une, la piste ISO2 principalement utilisée en Europe, contient, le PAN, la date d'expiration et des données additionnelles de sécurité ; la piste ISO1 utilisée principalement aux USA contient en plus le nom du porteur.

vii. Le code secret (PIN : Personal Identification Number) : mot de passe numérique sur 4 à 6 positions, défini et connu seulement par le porteur de carte

viii. La puce : Nouveau moyen technologique, qui contient les signes visibles et des éléments de sécurité additionnels. Avant l'arrivée de la puce, il suffisait souvent de réaliser un simple clonage (copie) de la piste magnétique sur un plastique vierge et d'utiliser cette copie sur des terminaux où il n'y avait pas d'authentification du porteur (via le PIN ou la signature).

Après l'arrivée de la puce, il était malheureusement toujours possible de réaliser ce clonage de la piste magnétique et d'utiliser la carte sur des terminaux ou dans des pays qui n'avaient pas encore adopté la puce.

C'est le cas actuellement avec la généralisation de EMV (EuroPay, MasterCard, VISA) en Europe mais pas encore aux Etats-Unis ni sur d'autres continents.

Pour utiliser frauduleusement une carte de paiement sur internet, le hacker doit simplement obtenir les signes de sécurité visibles de la carte.

Pour pallier cela, l'industrie des cartes a inventé de nouvelles barrières, qui ne sont plus directement visibles sur la carte, mais connues seulement du porteur :

- **Le 3D Secure:**

Afin d'éviter les fraudes liées aux tentatives d'usurpation d'identité, le 3D Secure consiste à s'assurer, lors de chaque paiement en ligne, que la carte est utilisée par son véritable titulaire : pour cela, le porteur de carte doit saisir en plus des informations habituelles, un mot de passe (statique ou dynamique) qu'il aura préalablement défini via le site internet de sa banque.

A chaque nouvelle tentative réussie de détournement des moyens de sécurité (c.-à-d. : lorsque les pertes deviennent suffisamment conséquentes pour que l'investissement technologique nouveau soit rentabilisé), une nouvelle réponse technologique est inventée et empilée sur les précédentes.

Dans la configuration actuelle des moyens de paiement utilisés sur internet, le hacker cherche par tous les moyens à obtenir les signes de sécurité visibles et invisibles de votre moyen de paiement ; et pour cela, il a deux méthodes :

- **Vous lui fournissez:**

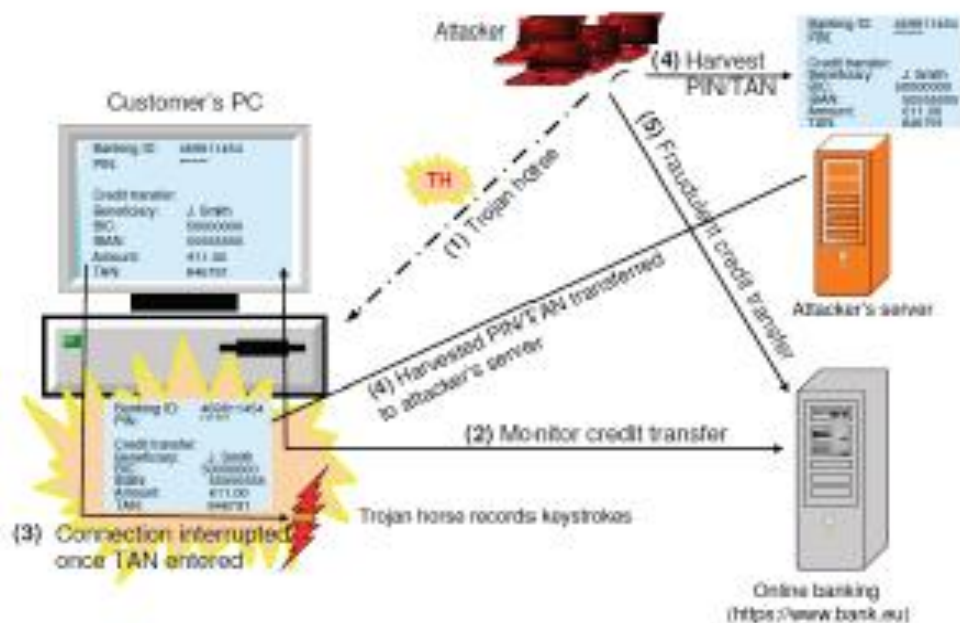
C'est le cas du phishing, où le hacker utilise la faiblesse humaine pour détourner votre attention et votre esprit critique et obtenir de vous des informations, que vous savez pourtant personnelles et strictement confidentielles

Dans ce cas, aucun moyen technologique, ni personne ne peut vous empêcher de donner ce que le hacker attend de vous !



- **Il vous les subtilise:**

Dans ce cas, on a plutôt à faire à une attaque « technologique », où le hacker « écoute aux portes » et s'installe impunément sur votre matériel informatique afin de récupérer toutes les informations que vous y stockez ou que vous encodez via votre clavier.



On parle alors d'attaque de type :

- Man-in-the-middle (Le hacker introduit un programme informatique entre votre PC et le serveur de la banque et détourne tout ce qui passe : montant de transactions, numéros de compte, code de sécurité, ...)
- Cheval de Troie (Le hacker installe sur votre PC un programme informatique, détourne tout ce qui passe : montant de transactions, numéros de compte, code de sécurité, ...)

Pour éviter les détournements d'informations confidentielles, l'industrie des cartes de paiement s'est regroupée afin d'édicter un niveau supplémentaire de règles de sécurité lorsque les données confidentielles de cartes de paiement sont stockées, traitées ou transmises par quelque intervenant que ce soit dans la chaîne de traitement des opérations par carte.

II. La norme PCI-DSS

Le Conseil pour la standardisation et la sécurité de l'Industrie des Cartes de Paiement (PCIDSS) a été créé en 2004. Une première version du PCI-DSS (Payment Card Industry – Data Security Standards) est alors publiée.

Il s'agit de 12 règles que les commerçants, les processeurs de données de cartes de paiement, les banques émettrices de cartes s'engagent à respecter afin d'assurer la confidentialité des informations des cartes de paiements. Les émetteurs de cartes exigent que les commerçant et processeurs de cartes soient conformes avec PCI-DSS.

1. Les 12 règles :

O La sécurité du réseau (1-2) :

Installer un Firewall ; Ne pas conserver les paramètres par défaut des matériels installés sur le réseau informatique

O La protection des données du porteur (3-4)

Chiffrer les données de cartes qui sont stockées (ne jamais les conserver en clair) ; Chiffrer la transmission de données sur les réseaux ouverts ou publics.

O L'environnement applicatif sécurisé (5-6)

Utiliser des logiciels anti-virus ; Tenir à jour les environnements hardware et software en fonction des mises à jour constructeur

O Méthodes fortes de contrôle d'accès (7-9)

Ne laisser l'accès aux données de cartes de paiement uniquement aux personnes ou applications qui en ont besoin ; Donner un identifiant unique à tout utilisateur qui accède à l'environnement informatique ; Mettre en place une restriction d'accès à tout immeuble contenant des cartes ou des données de cartes de paiement.

O Monitoring et test du réseau (10-11)

Faire un suivi de tout accès au réseau ou aux données de cartes de paiement ; Réaliser régulièrement des tests de pénétration du réseau informatique.

O Une politique de sécurité (12)

Mettre en place une politique de sécurité au sein de l'entreprise pour tous les aspects de sécurité.

Au vu de ces règles, deux sentiments contradictoires peuvent être exprimés :

- PCI DSS propose exclusivement des exigences de sécurité minimales.

Vous pouvez être conforme aux règles PCI DSS, et être non sécurisé au regard des vulnérabilités mises au jour quotidiennement pour les applications online et toutes autres. Par ailleurs, certains affirment que les règles PCI DSS sont très chères à implémenter, qu'il est très complexe de s'y conformer et qu'elles sont subjectives dans leur interprétation et leur exécution.

- Au contraire, d'autres insistent que PCI DSS était le pas en avant qui a permis à toute

l'industrie de prêter plus attention à la sécurité IT, même si un standard minimum n'était certainement pas suffisant pour éradiquer complètement les problèmes de sécurité. Ainsi, on peut considérer que tous les règlements, les législations, qu'ils soient émis par des institutions politiques, des régulateurs, des groupements industriels permettent de prendre en compte la sécurité de manière plus sérieuse.

Le concept de carte de paiement, tel qu'il a été conçu originellement, est-il alors le bon outil pour initier des paiements dans le monde de l'internet ?

Peut-être y a-t-il d'autres moyens de paiement, mieux adaptés à cet environnement particulier ?

On a vu que le paiement par carte bancaire sur internet, engendre de nombreuses fraudes, qui sont réalisées tant du côté du porteur de carte (infection de son matériel informatique) que du côté du commerçant (infection de son matériel informatique: serveur, base de donnée, terminal de paiement ...)

D'autres alternatives que la carte de paiement existent déjà. Celles-ci pourraient apporter une éventuelle solution à ce problème. De fraude sur internet.

- **Redirection vers le site e-banking**

Il s'agit d'un processus technique, par lequel ce n'est plus le commerçant qui collecte et stocke les données de cartes ou informations personnelles du client, mais une connexion triangulaire qui est établie entre le client, le commerçant et la banque du client. Cette redirection peut être établie de 2 façons:

- **Via le navigateur du client :**

Lorsque ce dernier est connecté à son PC :

Le client effectue son parcours d'achat sur le site du commerçant. Lors du paiement, le site du commerçant ouvre une connexion vers le site e-banking du client en fournissant les données utiles à l'initiation d'un paiement.

Le client se connecte sur son e-banking avec ses identifiants habituels et initie le paiement; lors de la clôture de la phase de paiement, le système e-banking est fermé et le client redirigé vers le site du marchand.

Le marchand n'a donc à aucun moment stocké quoi que ce soit, des informations personnelles et confidentielles du client.

- **Via Smartphone et QR-Code :**

Dans cette configuration, l'expérience d'achat est la même, sauf que la redirection est faite lorsque le client choisit de scanner un QR-Code généré par le commerçant, avec toutes les informations du panier.

L'application sur le Smartphone établit le lien de confiance entre le commerçant et la banque du client (à cet effet, le client et le commerçant se sont préalablement enregistrés et authentifiés auprès de cette application).

Le QR-Code contient toutes les données nécessaires pour initier le paiement via le système e-banking du client, pour lequel le client donne son autorisation. De telles solutions existent déjà au Luxembourg.

III. Utilisation d'un porte-monnaie électronique :

1. Principe

Le processeur qui propose ce service ouvre un compte bancaire dans une banque "P"; ce Compte recevant tous les mouvements de tous les pseudo-comptes de ses clients. Chaque client étant considéré comme une sous-rubrique de ce compte global.

Schéma explicatif de la porte- monnaie électronique

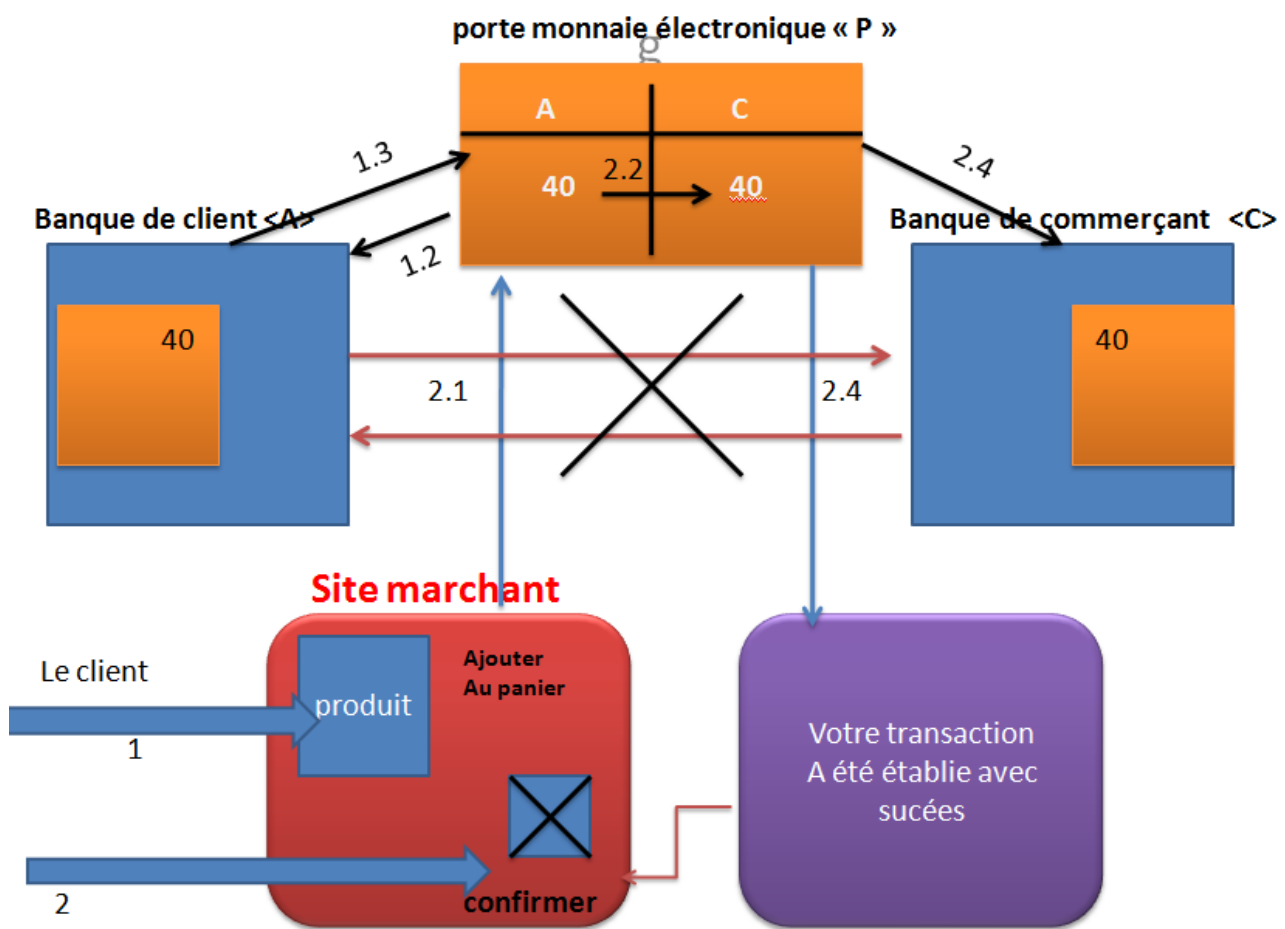


Figure 11: Explication des requêtes faites Durant la phase du paiement

- **Description :**

1. le client A entre sur le site marchand, il consulte les offres possibles, il peut ajouter au panier pour qu'il peut valider son choix d'achat avec le bouton confirmer .
 2. en cliquant sur confirmer le site marchand redirigé cette requête vers la porte- monnaie électronique .ce dernier accède a la banque de client A (1.2) et effectue la transaction de montant vers le compte de A dans la porte- monnaie électronique.
- Il existe un autre compte pour le marchand C dans la porte- monnaie électronique alors le montant pris de A sera transférer vers le compte de C et en fin il sera transférer vers la banque de marchand (3) .

A la fin un message sera envoyer vers le client quand le transfert est bien effectuer pour informer le client de ce que se passe.

La porte- monnaie électronique joue le rôle du pont entre deux banque car l'opération de transaction entre deux banque directement n'est pas vraiment sécurisé.

- **Mettre de l'argent dans son porte-monnaie électronique**

Le client "A" fournis ses données bancaires (numéros de compte ou informations de Carte) au processeur "P", dans un environnement sécurisé.

Le processeur initie auprès de la banque "A" du client "A", le transfert entre le compte "A" vers le compte "P" du processeur dans la banque "P".

Le livre de compte du processeur "P" fait alors apparaître un solde en faveur du client "A".

- **Processus d'achat**

Le client et le commerçant doivent tous les 2 posséder un porte-monnaie électronique. Lors de la phase de paiement, le site du commerçant effectue une redirection vers le site du processeur "P", avec toutes les informations utiles au paiement; le client s'y connecte et valide la transaction.

A ce moment là, seul un jeu d'écriture est réalisé entre le sous-compte du client "A" et le sous-compte du commerçant "C"; le solde du compte du processeur "P" restant inchangé. Il n'y a pas de transfert entre une banque à une autre.

Le client "A" et le commerçant "C" voient alors une information de ces transactions s'afficher sur leur tableau de bord.

- **Point de vue**

Avec un porte-monnaie électronique, la phase d'achat est totalement séparée de l'accès aux informations du compte bancaire ou de la carte de paiement du client.

Toute la sécurité repose sur la connexion à l'application porte-monnaie et au stockage des informations confidentielles que peut faire le processeur "P".

Ces processeurs de porte-monnaie électronique, sont bien évidemment conformes aux règles PCI-DSS, mais vont au-delà dans leurs exigences de sécurité. Ils effectuent en temps réel des analyses de risque qui permettent de réduire au maximum les fraudes et contournement des règles sécurité mises en place

- **Variante**

Une variante de ce fonctionnement en mode porte-monnaie électronique concerne les Smartphones avec l'utilisation de QR-Code pour déclencher le paiement. Cette possibilité permet d'utiliser le porte-monnaie électronique aussi bien sur un site internet qui génère et affiche un QR-Code lors de la finalisation du panier, mais aussi et surtout chez les commerçants physiques qui génèrent ce QR-Code en plus du ticket de caisse.

De telles solutions existent déjà au Luxembourg.

La carte bancaire de par son déploiement massif chez les clients bancaires et son très important niveau d'acceptation chez les commerçants, est effectivement le moyen naturel de paiement; Mais le monde de l'internet n'est pas adapté à la carte de paiement, ou plutôt, la carte de paiement n'a pas été créée et définie pour une utilisation sur internet. Beaucoup de choses ont été faites et sont faites pour pallier les défauts de ce moyen de paiement dans cet environnement particulier qu'est l'internet. Mais il est important de considérer les moyens alternatifs, voire imaginer des ponts entre ces moyens alternatifs et la carte de paiement, afin de limiter et réduire les risques lors des différentes phases du paiement sur internet.

Enfin, l'inadaptation de la carte au paiement en ligne a conduit au développement d'outils de prévention de la fraude; ces outils sont basés sur la collecte et la centralisation d'informations de plus en plus nombreuses sur les porteurs, ce qui pourrait être évité si un moyen de paiement plus adapté au monde de l'internet était développé et mis à disposition des clients.

IV. Conclusion :

Après avoir compris le niveau de difficulté et la complexité de la mission, nous avons fait une petite image qui explique ce qu'il faut faire pour nous mettre au bon chemin :

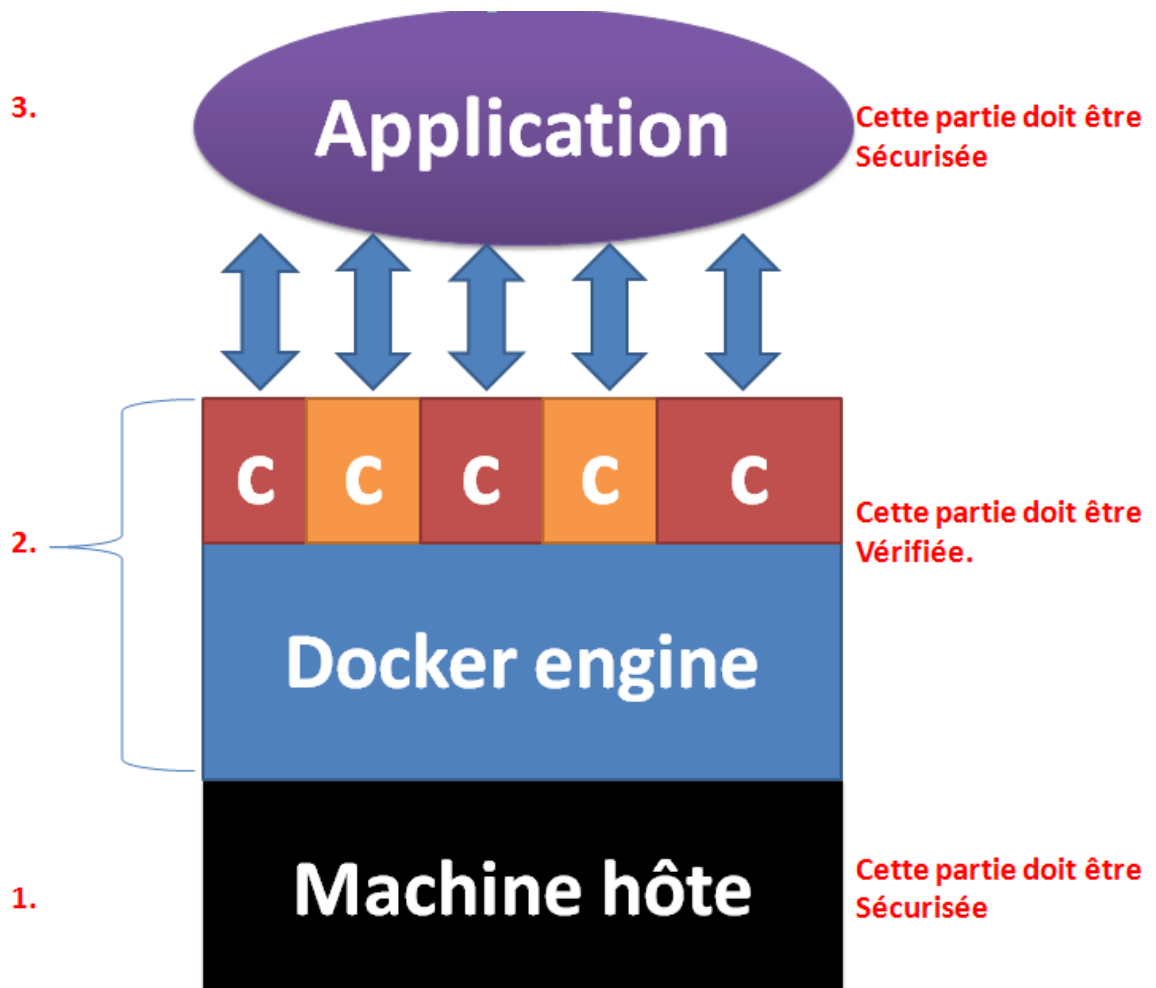


Figure 12: Les points principales

Chapitre 5 : tests et recommandation

I. Introduction :

Comme Docker exploite le même noyau que le système hôte pour réduire le besoin de ressources, les conteneurs peuvent être exposés à des risques de sécurité importants s'ils ne sont pas correctement configurés.

La liste détaillée ci-dessous suggère des mesures de durcissement qui peuvent être prises pour améliorer la posture de sécurité des conteneurs dans leur environnement respectif. Il convient de noter que les solutions proposées s'appliquent uniquement au déploiement de conteneurs Linux Docker sur des hôtes basés sur Linux, en utilisant la version la plus récente de Docker au moment de la rédaction de ce document.

Remarque: La plupart des options de ligne de commande suggérées peuvent être stockées et utilisées de manière similaire à l'intérieur d'un fichier Docker pour la création automatisée d'images.

II. Recommandations :

• Docker Images :

Docker prend désormais en charge les signatures cryptographiques [3] pour déterminer l'origine et l'intégrité des images de dépôt officielles. Cette fonctionnalité est cependant encore un travail en cours, car Docker émettra un avertissement mais n'empêchera pas l'image de s'exécuter réellement. En outre, il ne s'applique pas aux images non officielles. En général, assurez-vous que les images ne sont récupérées que dans des référentiels fiables et que l'option de ligne de commande `--insecure-registry = []` n'est jamais utilisée.

• Network Namespaces :

Par défaut, l'API REST du docker utilisée pour contrôler les conteneurs exposés via le démon Docker système n'est accessible que localement via un socket de domaine Unix.

L'exécution de Docker sur un port TCP (c.-à-d. Le forçage de l'adresse de liaison à l'aide de l'option `-H` lors du lancement du démon Docker) permettra à toute personne ayant accès à ce port d'accéder au conteneur, ce qui pourrait générer un accès root sur l'hôte également dans certains scénarios Où l'utilisateur local appartient au groupe docker .

Lorsque vous autorisez l'accès au démon sur TCP, assurez-vous que les communications sont correctement chiffrées à l'aide de SSL et que les contrôles d'accès empêchent efficacement les parties non autorisées d'interagir avec elle.

Les règles iptables de pare-feu Kernel peuvent être appliquées à docker0, l'interface de pont réseau standard pour Docker, pour appliquer ces contrôles.

Par exemple, la plage IP source d'un conteneur Docker peut être restreinte de parler avec le monde extérieur à l'aide du filtre iptables suivant :

```
Iptables -t filter -A FORWARD -s <source_ip_range> -j REJECT --reject-with  
icmp-admin-prohibited
```

- **Logging & Auditing :**

Collectez et archivez les journaux de sécurité relatifs à Docker pour des raisons de vérification et de surveillance.

L'accès aux fichiers journaux en dehors du conteneur, depuis l'hôte , peut être effectué en utilisant la commande suivante:

```
Docker run -v / dev / log: / dev / log <container_name> / bin / sh
```

Utilisation de la commande Docker intégrée:

```
docker logs ... (-f pour suivre la sortie journal)
```

Les fichiers journaux peuvent également être exportés pour un stockage persistant dans un fichier tarball en utilisant: `docker export ...`

- **SELinux ou AppArmor :**

Les modules de sécurité du noyau Linux telles que Security-Enhanced Linux (SELinux) et AppArmor peuvent être configurés, par le biais des politiques de sécurité de contrôle d'accès, à mettre en oeuvre des contrôles d'accès obligatoires (MAC) confinant les processus à un ensemble limité de ressources système ou privilèges.

SELinux peut être activé dans le conteneur à l'aide de `setenforce` , s'il était précédemment installé et configuré. Le support SELinux pour le démon Docker est désactivé par défaut et doit être activé en utilisant :

`--selinux-enabled`

Introduit dans Docker , le confinement d'étiquettes pour le conteneur peut être configuré à l'aide de l'argument nouvellement ajouté `--security-opt` pour charger les règles SELinux ou AppArmor, comme indiqué dans l'extrait de référence d'exécution de Docker ci-dessous.

`--security-opt = "label: user: USER"`: définit l'étiquette utilisateur pour le conteneur

`--security-opt = "label: role: ROLE"`: définit le rôle d'étiquette pour le conteneur

`--security-opt = "label: type: TYPE"`: définit le type d'étiquette pour le conteneur

`--security-opt = "label: level: LEVEL"`: définit le niveau de l'étiquette pour le conteneur

`--security-opt = « AppArmor: PROFIL »`: définir le profil AppArmor à appliquer au conteneur

Exemple:

```
docker run --security-opt=label:level:s0:c100,c200 -i -t centos bash
```

`libseccomp` (et l'extension `seccomp-bpf`) :

La bibliothèque `libseccomp` permet de restreindre l'utilisation des procédures `syscall` du kernel Linux en fonction d'une approche de liste blanche. Les procédures `Syscall` qui ne sont pas essentielles au fonctionnement du système devraient idéalement être désactivées pour éviter tout abus ou mauvaise utilisation dans un conteneur compromis.

Cette fonctionnalité est actuellement un travail en cours (disponible dans le pilote LXC, pas dans `libcontainer` qui est maintenant par défaut).

Pour redémarrer le démon Docker pour utiliser le pilote LXC taper cet

Commande `docker -d -e lxc`

• Privilèges du démon docker :

N'utilisez pas l'option `--privileged` de ligne de commande . Cela permettrait également au conteneur d'accéder à tous les périphériques de l'hôte et, en plus, fournirait au conteneur une configuration spécifique LSM (c'est-à-dire SELinux ou AppArmor) qui lui donnerait le même niveau d'accès que les processus exécutés sur l'hôte.

Éviter l'utilisation de `--privileged` aide à réduire la surface d'attaque et le potentiel de compromis de l'hôte. Cependant, cela ne signifie pas que le démon sera exécuté sans les privilèges de l'administrateur qui sont actuellement requis dans la dernière version.

La capacité de lancer le démon et les conteneurs ne doit être confiée qu'à un utilisateur de confiance.

Réduisez les privilèges appliqués à l'intérieur du conteneur en utilisant l'option `-u`.

Exemple:

```
docker run -u <username> -it <container_name> /bin/bash
```

Toute partie utilisateur du groupe docker pourrait éventuellement prendre l'accès root sur la machine hôte à partir de la conteneur

- **cgroups :**

Afin d'éviter les attaques de déni de service (DoS) via l'épuisement des ressources du système, un certain nombre de restrictions de ressources peuvent être appliquées en utilisant des arguments de ligne de commande spécifiques.

L'utilisation du processeur:

```
docker run -it --rm --cpuset=0,1 -c 2 ...
```

Utilisation de la mémoire:

```
docker run -it --rm -m 128m ...
```

Utilisation du stockage:

```
docker -d --storage-opt dm.basesize=5G
```

- **Les fichiers binaires SUID / GUID :**

Les fichiers binaires SUID et GUID peuvent s'avérer dangereux lorsqu'ils sont vulnérables aux attaques conduisant à une exécution de code arbitraire (par exemple, des débordements de tampon), car ils seront exécutés dans le contexte du propriétaire ou du groupe de fichiers du processus.

Dans la mesure du possible, interdire à SUID et SGID de prendre effet en réduisant les capacités offertes aux conteneurs utilisant des arguments de ligne de commande spécifiques.

```
docker run -it --rm --cap-drop SETUID --cap-drop SETGID ...
```

Sinon, envisagez de supprimer les capacités SUID du système en associant le système de fichiers avec l'attribut nosuid.

Une dernière option pourrait être d'éliminer complètement les binaires SUID et GUID indésirables du système. Ces types de binaires peuvent être trouvés sur un système Linux en exécutant les commandes suivantes:

```
find / -perm -4000 -exec ls -l {} \; 2>/dev/null
```

```
find / -perm -2000 -exec ls -l {} \; 2>/dev/null
```

Les autorisations de fichier SUID et GUID peuvent ensuite être supprimées à l'aide de commandes similaires aux suivantes :

```
sudo chmod u-s nom_du_fichier.
```

```
sudo chmod -R g-s nom_du_repertoire.
```

- **Groupe de contrôle des périphériques (/ dev / *) :**

Si nécessaire, montez les périphériques à l'aide de l'option intégrée (--device) ,Ne pas utiliser -v avec l'argument (--privileged) .

Des autorisations granulaires peuvent être attribuées à chaque périphérique en utilisant un troisième ensemble d'options : rwm pour remplacer respectivement les autorisations de read, write et mknod (la valeur par défaut inclut toutes les autorisations).

Exemple (pour l'utilisation de la carte son avec l'autorisation de read-only):

```
docker run --device=/dev/snd:/dev/snd:r ...
```

- **Services et applications :**

Pour réduire le risque de déplacement latéral si un conteneur Docker devait être compromis, envisager d'isoler des services sensibles

(Par exemple, exécuter le service SSH sur un hôte bastion ou dans une machine virtuelle)

En outre, ne lancez pas d'applications non approuvées avec les privilèges root dans les conteneurs.

- **Points de montage :**

Ceci est géré automatiquement par Docker lors de l'utilisation de la bibliothèque de conteneurs native (c'est-à-dire libcontainer).

Cependant, lorsque vous utilisez la bibliothèque de conteneurs LXC, les points de montage sensibles devraient idéalement être montés manuellement avec des autorisations en lecture seule, y compris:

```
/sys  
/proc/sys  
/proc/sysrq-trigger  
/proc/irq  
/proc/bus
```

Les autorisations de montage devraient ensuite être supprimées pour éviter le remontage.

- **Linux Kernel (Le Noyeau linux) :**

Assurez-vous que le noyau est à jour en utilisant l'utilitaire de mise à jour fourni par le système (par exemple apt-get, yum, etc.). Les noyaux périmés sont plus susceptibles d'être vulnérables aux vulnérabilités divulguées publiquement.

Utilisez un noyau renforcé avec GRSEC ou PAX, qui fournit par exemple une sécurité accrue contre les bugs de corruption de mémoire.

- **User Namespaces :**

Docker ne prend pas en charge les espaces de noms d'utilisateur mais est une fonctionnalité en cours de développement [13]. Le mappage UID est actuellement pris en charge par le pilote LXC, mais pas dans la bibliothèque native de libcontainer.

Cette fonctionnalité permettrait au Docker daemon de s'exécuter en tant qu'utilisateur non privilégié sur l'hôte, mais apparaîtrait en tant que root dans les conteneurs. Tout en utilisant le pilote LXC, cela peut être utilisé en utilisant l'option -lxc-conf, comme indiqué dans l'exemple de commande d'exécution de Docker ci-dessous.

```
docker run -lxc-conf="lxc.id_map = u 0 100000 65536" -lxcconf="
```

```
lxc.id_map = g 0 100000 65536" ...
```

Les arguments spécifiés dans la commande ci-dessus demanderont à Docker de mapper les UID et les GID 100000 à 65536 sur l'hôte aux UID et GID 0 à 65536 à un compte d'utilisateur spécifique, défini au niveau du système sur l'hôte dans une configuration similaire à

```
/etc/subgid:<username>:100000:65536
```

```
/etc/subuid:<username>:100000:65536
```

Capacités :

Laissez les capacités linux au minimum chaque fois que cela est possible. Les fonctionnalités par défaut de Docker incluent:

chown, dac_override, fowner, kill, setgid, setuid, setpcap, net_bind_service, net_raw, sys_chroot, mknod, setfcap, et audit_write.

Peut être contrôlé lors du lancement d'un conteneur depuis la ligne de commande avec --cap-add=[] ou --cap-drop=[].

Exemple :

```
docker run --cap-drop setuid --cap-drop setgid -ti <container_name> /bin/sh
```

Environnements multi-location :

En raison de la nature partagée du noyau des conteneurs Docker, la séparation des tâches dans les environnements à logements multiples ne peut pas être assurée en toute sécurité.

Il est recommandé que les conteneurs soient exécutés sur des hôtes qui n'ont pas d'autre but et ne soient pas utilisés pour des opérations sensibles. Envisagez de transférer tous les services dans des conteneurs contrôlés par Docker.

Dans la mesure du possible, maintenez les communications entre conteneurs au minimum en configurant le démon Docker pour utiliser --icc = false et spécifiez -link avec docker run lorsque nécessaire ,Ou --export = port pour exposer un port du conteneur sans le publier sur l'hôte.

Organiser des groupes de conteneurs mutuellement fiables pour séparer les machines.

- **Virtualization complète :**

Utilisez une solution de virtualisation complète pour contenir Docker, comme KVM. Cela empêchera l'escalade du conteneur vers l'hôte si une vulnérabilité du noyau est exploitée à l'intérieur de l'image Docker.

Audits de sécurité :

Effectuez des audits de sécurité régulières de votre système hôte et de vos conteneurs pour identifier les mauvaises configurations ou les vulnérabilités qui pourraient exposer votre système à un compromis.

Pour faire une récapitulation on peut dire que dans la partie précédente on a citer les recommandations de base dans l'opération de la sécurisation des conteneurs, Et nous ajoutons dans la prochaine partie un rappel sur les points principales pour garantir un environnement plus sécurisé pour nos applications.

NOTE: Avant de faire installer docker il faut sécuriser notre machine:
Notre environnement de travail est notre machine ubuntu 16.04

- Renforcez la sécurité sur notre machine Ubuntu :

16.04 LTS en installant et en configurant ce qui suit:

1-Installer et configurer le pare-feu – ufw.

2-Sécuriser la mémoire partagée - fstab.

3-SSH - Connexion basée sur clé, désactiver la connexion root et changer le port

4-Apache SSL - Désactiver le support SSL v3

5-Protégez le su en limitant l'accès uniquement au groupe d'administration

6-Renforcer le réseau avec les paramètres sysctl

7-Désactiver la recrescence DNS ouverte et supprimer les informations de version - Bind9
DNS

8-Empêcher l'IP Spoofing

9-blinder le PHP pour la sécurité.

10-Limiter la fuite d'informations Apache

11-Installer et configurer le pare-feu Apache Application– ModSecurity

12-Protéger des attaques DDOS (déni de service) avec Mod_Evasive

- 13-Enregistrez les journaux et interdisez les hôtes suspects - DenyHosts et Fail2Ban
- 14-Détection d'intrusion – PSAD
- 15-Vérifiez les RootKits - RKHunter et CHKRootKit
- 16-Scan open Ports – Nmap
- 17-Analyser les fichiers LOG du système – LogWatch
- 18-SELinux - Apparmor
- 19-Auditez la sécurité de votre système - Tiger

Les points d'intérêt :

- **Sécurité des images Docker:**

Les conteneurs Docker ont besoin d'images pour fonctionner. C'est le premier lien dans la chaîne que nous devons valider.

Cela introduit une petite révolution pour l'équipe IT, certains tests assignés aux opérateurs ont été transférés dans la chaîne d'intégration continue des développeurs. Ils sont maintenant réalisés dans le processus de construction ou de post-construction.

Ces tests peuvent être exécutés à travers des solutions telles que 'Docker scanning services' de Docker.com 'Twistlock Trust' développé par twistlock.com, ou en utilisant la solution opensource 'Clair' de Coreos.com.

Ces solutions (Deep Container Inspection, DCI) visent à garantir que les outils et logiciels intégrés dans les images ne sont pas vulnérables.

L'exploitation de vulnérabilités pourrait conduire une personne malveillante à prendre le contrôle du conteneur. Il pourrait alors tenter d'obtenir des privilèges root (élévation de privilège) et / ou essayer de prendre le contrôle des conteneurs voisins.

- **Sécurité des images dans le Docker Hub :**

Par des besoins faciles à utiliser ou de prototypage rapide, vous pouvez décider d'utiliser l'image de construction à partir du registre.

Il faut être vigilant car le 'Hub Docker Registry' offre une multitude d'images téléchargeables. Toujours préférer les images officielles proposées et validées par Docker. Ils sont régulièrement mis à jour, bien documentés et vérifiés par le service « Docker scanning service ».

Si vous n'avez d'autre choix que d'utiliser une image non officielle, assurez-vous qu'elle a été suffisamment téléchargée. Vous pouvez lire les commentaires sur l'image et vérifier le 'Dockerfile' sur github pour vous assurer que l'image correspond à ce qui est indiqué.

Si vous êtes prêt à utiliser un registre tiers parce que l'image n'est pas disponible sur Docker Hub, vérifiez sa réputation et vérifiez que la communication établie avec ce nouveau registre est sécurisée et chiffrée.

- **La sécurité des données partagées :**

N'oubliez pas de consulter les données partagées entre les conteneurs

(Volumes dans la langue du docker).

Ne pas exposer plus d'informations que nécessaire. L'utilisation de conteneurs est également un moyen de compartimenter l'information;

Évitez de perdre de l'espace sur le disque du serveur hôte, par exemple, l'image Elasticsearch utilise le répertoire hôte: /usr/share/

elasticsearch / data *

* Ces informations sont disponibles à l'aide de la commande 'docker volume ls' et 'docker volume inspect'.

Sécurité du moteur Docker :

Pour exécuter les conteneurs, vous devez configurer correctement le moteur Docker.

Le projet open source «docker-bench-security» met en œuvre les recommandations de la CIS (the Center for Internet Security) à l'aide d'un script à exécuter. Il vérifie que l'hôte qui exécute les conteneurs définit les meilleures pratiques de sécurité. Il s'agit d'appliquer des conseils de durcissement à Docker, sa configuration et les bons paramètres du profil de sécurité (apparmor, seccomp, selinux ..).

Cependant, veillez à ce que les outils ne permettent pas d'automatiser toutes les recommandations. Certains tests ne peuvent être effectués que par une vérification, selon l'environnement déployé.

⇒ **Docker Bench for Security**

Description :

Le Docker Bench for Security est un script qui vérifie des douzaines de pratiques courantes communes concernant le déploiement de conteneurs Docker dans la

production. Ces contrôles sont basés sur toutes les recommandations tirées du document CEM Docker 1.6 Benchmark

Centre d'intérêt : principalement le serveur Docker et quelques conseils pour les images et les conteneurs.

- Langage: script Shell
- Méthodologie: exécutez le script dans le même serveur où Docker est en cours d'exécution ou à partir d'un conteneur. Il créera un rapport de shell avec les alertes INFO, WARN ou PASS.
- License: Apache 2.0
- Niveau d'installation / utilisabilité: facile

⇒ **OpenSCAP Container Compliance :**

Description: Basé sur la même philosophie que son projet parent OpenSCAP qui prend en charge l'analyse CVE, plusieurs formats de rapports et des politiques personnalisées. Des instructions et des paquets spécifiques pour RedHat 7 sont ici. Note: SCAP est la norme américaine maintenue par l'Institut national des normes et de la technologie (NIST). Le projet OpenSCAP est une collection open source d'outils pour la mise en oeuvre et l'application de cette norme.

Focus: Images et conteneurs.

Langage: script Shell .

Méthodologie: exécutez la commande oscap-docker contre une image ou un conteneur et obtenez les résultats sur un rapport html très utile et descriptif.

Licence: GPL v3.

Niveau d'installation / utilisabilité: facile

⇒ **Batten :**

Description : Outil de durcissement et d'audit pour les serveurs et conteneurs docker. C'est à peu près le même que Drydock ou Docker Bench for Security.

Centre d'intérêt : serveur Docker et conteneurs

Langage: Go

Méthodologie : Exécutez-le en tant que conteneur et vérifiez le serveur et les conteneurs à l'aide de CIS Docker 1.6 Benchmark.

License: MIT

Niveau d'installation / utilisabilité: facile

⇒ **Drydock :**

Description : Drydock est un outil d'audit de sécurité Docker écrit en Python. Initialement, il a été inspiré par Docker Bench for Security mais vise à fournir une manière plus souple d'évaluer les installations et les déploiements de Docker. Drydock permet une création et une utilisation faciles de profils d'audit personnalisés afin d'éliminer le bruit et les fausses alarmes. Les rapports sont enregistrés au format JSON pour faciliter l'analyse. Drydock utilise fortement l'API du client docker-py pour communiquer avec Docker. Il est basé sur CIS Docker 1.6 Benchmark.

Centre d'intérêt : Serveur Docker et conteneurs.

Langage: Python

Méthodologie : Il utilise certaines des meilleures pratiques de CIS Docker 1.6 Benchmark existantes pour vérifier les options de configuration du serveur.

License: GPL v2

Niveau d'installation / utilisabilité: facile

Exemple de test avec l'outil Drydock sur notre machine :

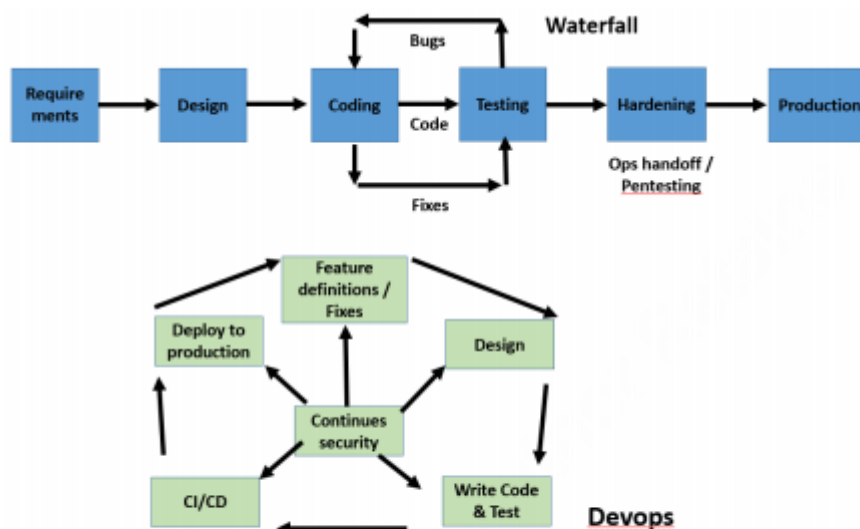
III. Conclusion :

De nombreux outils liés aux tests de sécurité des conteneurs Docker sont neufs avec seulement quelques mois ou semaines depuis la sortie. Comme Docker augmente rapidement et libère des versions avec beaucoup de nouvelles fonctionnalités (y compris les améliorations de sécurité) presque chaque semaine, c'est une course difficile à garder l'un de ces outils à jour. Un premier objectif devrait être d'aborder l'audit et l'évaluation de la vulnérabilité de l'écosystème des conteneurs, qu'il s'agisse de développement, mise en scène, essai ou environnement de production. Mais ça ne devrait pas s'arrêter là.

Conclusion générale

Pour terminer ce recherche nous allons noter que ce type d'étude est généralement faite par un tout un équipe informatique composé des membres professionnelles et expert dans tous les parties de ce recherche , un Développeur en WEB , Un ingénieur en sécurité et un expert en linux et Administration des systèmes et réseaux , ce type d'équipe est connu par le nom DevOps : DevOps est un nouveau terme issu de la collision de deux grandes tendances connexes. Le premier était également appelé «administration du système agile» ou «opérations agiles»; Il est né de l'application des nouvelles approches Agile et s'appuyer sur le fonctionnement des opérations. La seconde est une compréhension beaucoup plus étendue de la valeur de la collaboration entre le personnel de développement et d'opérations à toutes les étapes du cycle de développement lors de la création et l'exploitation d'un service et de l'importance des opérations dans notre monde de plus en plus axé sur les services ,C'est bien et intéressant, mais peut-être un peu trop ésotérique et spécifique aux types de « Internet startup ». Je crois que vous pouvez définir DevOps plus pratiquement comme :

DevOps est la pratique des ingénieurs d'opérations et de développement qui participent ensemble à l'ensemble du cycle de vie des services, de la conception au processus de développement au support de production.



Waterfall versus DevOps development cycle.

Après avoir terminé de mentionner tout ce qui est nécessaire pour travailler sur des opérations aussi complexes, nous voulons dire que ce document n'est que la première étape et l'étape de base pour les deux étudiants suivants qui accompliront la mission de rendre l'infrastructures du « Cloud Computing » plus sûre En effectuant des recherches de ce type .



ANNEXE :

Quelques tests effectués sur la machine ubuntu :

- En utilisant l'outil **docker bench security** :

```
root@wahbi-u:/home/wahbi/Desktop/docker-bench-security# sh docker-bench-security.sh
# -----
# Docker Bench for Security v1.3.2
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker 1.13 Benchmark.
# -----

Initializing 23 21:41:51 CET 2017

[INFO] 1 - Host Configuration
[WARN] 1.1 - Create a separate partition for containers
[NOTE] 1.2 - Harden the container host
[INFO] 1.3 - Keep Docker up to date
[INFO] * Using 17.03.1, when 17.05.0 is current as of 2017-05-01
[INFO] * Your operating system vendor may provide support and security maintenance for Docker
[INFO] 1.4 - Only allow trusted users to control Docker daemon
[INFO] * docker:x:999:
[WARN] 1.5 - Audit docker daemon - /usr/bin/docker
[WARN] 1.6 - Audit Docker files and directories - /var/lib/docker
[WARN] 1.7 - Audit Docker files and directories - /etc/docker
[WARN] 1.8 - Audit Docker files and directories - docker.service
[WARN] 1.9 - Audit Docker files and directories - docker.socket
[WARN] 1.10 - Audit Docker files and directories - /etc/default/docker
[INFO] 1.11 - Audit Docker files and directories - /etc/docker/daemon.json
[INFO] * File not found
[WARN] 1.12 - Audit Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Audit Docker files and directories - /usr/bin/docker-runc

[INFO] 2 - Docker Daemon Configuration
[WARN] 2.1 - Restrict network traffic between containers
[PASS] 2.2 - Set the logging level
[PASS] 2.3 - Allow Docker to make changes to iptables
[PASS] 2.4 - Do not use insecure registries
[WARN] 2.5 - Do not use the aufs storage driver
[INFO] 2.6 - Configure TLS authentication for Docker daemon
[INFO] * Docker daemon not listening on TCP
[INFO] 2.7 - Set default ulimit as appropriate
[INFO] * Default ulimit doesn't appear to be set
```

[WARN] 2.8 - Enable user namespace support
 [PASS] 2.9 - Confirm default cgroup usage
 [PASS] 2.10 - Do not change base device size until needed
 [WARN] 2.11 - Use authorization plugin
 [WARN] 2.12 - Configure centralized and remote logging
 [WARN] 2.13 - Disable operations on legacy registry (v1)
 [WARN] 2.14 - Enable live restore
 [PASS] 2.15 - Do not enable swarm mode, if not needed
 [PASS] 2.16 - Control the number of manager nodes in a swarm (Swarm mode not enabled)
 [PASS] 2.17 - Bind swarm services to a specific host interface (Swarm mode not enabled)
 [WARN] 2.18 - Disable Userland Proxy
 [PASS] 2.19 - Encrypt data exchanged between containers on different nodes on the overlay network
 [PASS] 2.20 - Apply a daemon-wide custom seccomp profile, if needed
 [PASS] 2.21 - Avoid experimental features in production
 [PASS] 2.22 - Use Docker's secret management commands for managing secrets in a Swarm cluster (Swarm mode not enabled)
 [PASS] 2.23 - Run swarm manager in auto-lock mode (Swarm mode not enabled)
 [NOTE] 2.24 - Rotate swarm manager auto-lock key periodically

[INFO] 3 - Docker Daemon Configuration Files
 [PASS] 3.1 - Verify that docker.service file ownership is set to root:root
 [PASS] 3.2 - Verify that docker.service file permissions are set to 644 or more restrictive
 [PASS] 3.3 - Verify that docker.socket file ownership is set to root:root
 [PASS] 3.4 - Verify that docker.socket file permissions are set to 644 or more restrictive
 [PASS] 3.5 - Verify that /etc/docker directory ownership is set to root:root
 [PASS] 3.6 - Verify that /etc/docker directory permissions are set to 755 or more restrictive
 [INFO] 3.7 - Verify that registry certificate file ownership is set to root:root
 [INFO] * Directory not found
 [INFO] 3.8 - Verify that registry certificate file permissions are set to 444 or more restrictive
 [INFO] * Directory not found
 [INFO] 3.9 - Verify that TLS CA certificate file ownership is set to root:root
 [INFO] * No TLS CA certificate found
 [INFO] 3.10 - Verify that TLS CA certificate file permissions are set to 444 or more restrictive
 [INFO] * No TLS CA certificate found
 [INFO] 3.11 - Verify that Docker server certificate file ownership is set to root:root
 [INFO] * No TLS Server certificate found
 [INFO] 3.12 - Verify that Docker server certificate file permissions are set to 444 or more restrictive
 [INFO] * No TLS Server certificate found
 [INFO] 3.13 - Verify that Docker server key file ownership is set to root:root
 [INFO] * No TLS Key found
 [INFO] 3.14 - Verify that Docker server key file permissions are set to 400 or more restrictive
 [INFO] * No TLS Key found
 [PASS] 3.15 - Verify that Docker socket file ownership is set to root:docker

```
[PASS] 3.16 - Verify that Docker socket file permissions are set to 660 or more restrictive
[INFO] 3.17 - Verify that daemon.json file ownership is set to root:root
[INFO] * File not found
[INFO] 3.18 - Verify that daemon.json file permissions are set to 644 or more restrictive
[INFO] * File not found
[PASS] 3.19 - Verify that /etc/default/docker file ownership is set to root:root
[PASS] 3.20 - Verify that /etc/default/docker file permissions are set to 644 or more restrictive
```

```
[INFO] 4 - Container Images and Build Files
[INFO] 4.1 - Create a user for the container
[INFO] * No containers running
[NOTE] 4.2 - Use trusted base images for containers
[NOTE] 4.3 - Do not install unnecessary packages in the container
[NOTE] 4.4 - Scan and rebuild the images to include security patches
[WARN] 4.5 - Enable Content trust for Docker
[WARN] 4.6 - Add HEALTHCHECK instruction to the container image
[WARN] * No Healthcheck found: [ubuntu:latest]
[WARN] * No Healthcheck found: [prestashop/prestashop:latest]
[WARN] * No Healthcheck found: [mysql:latest]
[WARN] * No Healthcheck found: [hello-world:latest]
[INFO] 4.7 - Do not use update instructions alone in the Dockerfile
[INFO] * Update instruction found: [prestashop/prestashop:latest]
[INFO] * Update instruction found: [mysql:latest]
[NOTE] 4.8 - Remove setuid and setgid permissions in the images
[INFO] 4.9 - Use COPY instead of ADD in Dockerfile
[INFO] * ADD in image history: [ubuntu:latest]
[INFO] * ADD in image history: [prestashop/prestashop:latest]
[INFO] * ADD in image history: [mysql:latest]
[NOTE] 4.10 - Do not store secrets in Dockerfiles
[NOTE] 4.11 - Install verified packages only
```

```
[INFO] 5 - Container Runtime
[INFO] * No containers running, skipping Section 5
```

```
[INFO] 6 - Docker Security Operations
[INFO] 6.1 - Perform regular security audits of your host system and containers
[INFO] 6.2 - Monitor Docker containers usage, performance and metering
[INFO] 6.3 - Backup container data
[INFO] 6.4 - Avoid image sprawl
[INFO] * There are currently: 4 images
[INFO] 6.5 - Avoid container sprawl
[INFO] * There are currently a total of 6 containers, with 0 of them currently running
```

- En utilisant l'outil **Drydock** :

```
wahbi@wahbi-u:~/Desktop/drydock$ sudo python drydock.py
[sudo] password for wahbi:
/bin/sh: 1: auditctl: not found
2017-05-31 08:31:47,532 - ERROR - auditd is not installed. REMINDER:      safedock should be run as root
drydock v0.3 Audit Results
=====
```

1.Host Configuration

1.1 Create a separate partition for containers

Status: **Fail**

Description: No separate partition for containers

1.2 Use the updated kernel version

Status: **Pass**

Description: Host uses an updated kernel

Output:

4.8.0-52-generic

1.5 Remove all non-essential services from the host

Description: Host has 8 open ports

Output:

[[':', 443], [':', 21], ['10.0.3.1', 53], ['127.0.0.1', 5940], ['127.0.1.1', 53], [':', 80], [':', 8080], [':', 3306]]

1.6 Keep Docker up to date

Status: **Pass**

Description: Host uses an updated Docker version

Output:

17.03.1-ce

1.7 Only allow trusted users to control Docker daemon

Description: 0 users in docker group

1.8 - 1.19 Audit docker daemon, files and directories

Status: **Pass**

Description: All auditd rules are in place

Output:

{'found': [], 'missing': []}

2.Docker Daemon Configuration

Generic method to detect insecure arguments for running docker

Status: **Pass**

Description: No insecure arguments found

Generic method to detect missing security-hardening arguments

Status: **Fail**

Description: Docker is running with 3 arguments missing

Output:

['--icc=false', '--registry-mirror', '--default-ulimit']

3.Docker daemon configuration files

Check permissions, according to perms for a given file or folder

Status: **Fail**

Description: 1 file(s) with wrong permissions

Output:

['/var/run/docker.sock']

Check file user and group owner.

Status: **Pass**

Description: File user and group owner are correct

4.Container Images and Build File

4.1 Create a user for the container

Status: **Fail**

Description: 3 container(s) running as root

Output:

['u'96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb', u'169a41e215fffd7087d830394ce29b7b413b0b0a52eb3606d517540bb4e25351', u'930e1127a4ef2538c5ec7777aadafe0482aadb35fb08d95093da60432b8e296c']

5.Container Runtime

5.1 Verify AppArmor profile

Status: **Fail**

Description: 3 container(s) with no AppArmor profile.

Output:

[u'96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb', u'169a41e215fffd7087d830394ce29b7b413b0b0a52eb3606d517540bb4e25351', u'930e1127a4ef2538c5ec7777aadaf0482aadbc35fb08d95093da60432b8e296c']

5.2 Verify SELinux security options

Status: **Fail**

Description: 3 containers with no SELinux policies.

Output:

[u'96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb', u'169a41e215fffd7087d830394ce29b7b413b0b0a52eb3606d517540bb4e25351', u'930e1127a4ef2538c5ec7777aadaf0482aadbc35fb08d95093da60432b8e296c']

5.3 Verify that containers are running only a single main process

Status: **Fail**

Description: 1 containers have more than one main process

Output:

[u'169a41e215fffd7087d830394ce29b7b413b0b0a52eb3606d517540bb4e25351']

5.4 Restrict Linux Kernel Capabilities within containers

Status: **Pass**

Description: No kernel capabilities within containers

5.5 Do not use privileged containers

Status: **Pass**

Description: No privileged containers detected.

5.6 Do not mount sensitive host system directories on containers

Status: **Pass**

Description: No sensitive dirs mounted

5.7 Do not run ssh within containers

Status: **Pass**

Description: No container is running ssh

5.8 Do not map privileged ports within containers

Status: **Pass**

Description: No unauthorized privileged ports found

5.9 Open only needed ports on container

Status: **Fail**

Description: Unneeded exposed ports found

Output:

[[[u'80/tcp'], u'phpmyadmin/phpmyadmin'], ([u'3306/tcp'], u'mysql:5.7')]

5.10 Do not use host network mode on container

Status: **Pass**

Description: All containers are inside a seperate network stack

5.11 Limit memory usage for container

Status: **Fail**

Description: 3 container(s) have no memory limits

Output:

[u'96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb', u'169a0e1127a4ef2538c5ec7777aadaf0482aadb35fb08d95093da60432b8e296c']

5.12 Set container CPU priority appropriately

Status: **Fail**

Description: 3 container(s) have no CPU shares set

Output:

[u'96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb', u'160e1127a4ef2538c5ec7777aadaaf0482aadb35fb08d95093da60432b8e296c']

5.13 Mount container's root filesystem as read-only

Status: **Fail**

Description: 3 container(s) have writable root FS

Output:

[u'96348cf51bf8df306d6fd84273f4d278453af78add29cbeba6a30ceafbb342cb', u'160e1127a4ef2538c5ec7777aadaaf0482aadb35fb08d95093da60432b8e296c']

5.14 Bind incoming container traffic to a specific host interface

Status: **Fail**

Description: Containers listen to any host interface

Output:

[[u'8080'], u'sha256:976dd870f876b8bf38d3e3a65768f81957729aadaaabe644be4b

5.15 Set the 'on-failure' container restart policy to 5

Status: **Fail**

Description: 2 container(s) have bad restart policy

Output:

[u'169a41e215fffd7087d830394ce29b7b413b0b0a52eb3606d517540bb4e25351', u'93

5.16 Do not share the host's process namespace

Status: **Pass**

Description: All containers' process namespace is isolated

5.17 Do not share the host's IPC namespace

Status: **Pass**

Description: All containers' process namespace is isolated

5.18 Do not directly expose host devices to containers

Description: No host devices exposed to containers

Overview

Profile: conf/default.yml

Date: 2017-05-31 08:31:48.799171

Score: **13/26**

wahbi@wahbi-u:~/Desktop/drydock\$ |

Bibliographie

- [1] Docker, Linux Containers (LXC), and security (August, 2014). Jérôme Petazzoni. [presentation slides] <http://www.slideshare.net/jpetazzo/docker-linux-containers-lxc-and-security>
- [2] Docker and SELinux (July, 2014). Daniel Walsh [video] <https://www.youtube.com/watch?v=zWGFqMuEHdw>
- [3] Docker 1.3: Signed Images, Process Injection, Security Options, Mac shared directories (October, 2014). Scott Johnston <http://blog.docker.com/2014/10/docker-1-3-signed-images-process-injection-security-options-mac-shared-directories/>
- [4] Exploring LXC Networking (November, 2013). Milos Gajdos. <http://containerops.org/2013/11/19/lxc-networking/>
- PaaS under the hood, episode 1: kernel namespaces (November, 2012). Jérôme Petazzoni. <http://blog.dotcloud.com/under-the-hood-linux-kernels-on-dotcloud-part>
- Exploring networking in Linux containers (January, 2014). Milos Gajdos. [presentation slides] <https://speakerdeck.com/gyre007/exploring-networking-in-linux-containers>
- [5] How to grant rights to users to use Docker in Fedora (October 2014). Daniel Walsh <http://opensource.com/business/14/10/docker-user-rights-fedora>
- [6] Running Docker with https. [Docker documentation] <https://docs.docker.com/articles/https/>
- [7] security suggestions when running malicious code, Google Groups (August, 2013). Jérôme Petazzoni <https://groups.google.com/forum/#!msg/docker-user/uuiQ3Nk3uSY/SuFpdO6BPmYJ>
- [8] Monitoring Images and Containers. [Red Hat documentation] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Resource_Management_and_Linux_Containers_Guide/sec-Monitoring_Images.html
- [9] Docker 1.3: Signed Images, Process Injection, Security Options, Mac shared directories (October, 2014). Scott Johnston <http://blog.docker.com/2014/10/docker-1-3-signed-images-process-injection-security-options-mac-shared-directories/>
- [10] Resource management in Docker (September, 2014). Marek Goldmann. <https://goldmann.pl/blog/2014/09/11/resource-management-in-docker/>
- Gathering LXC and Docker Containers Metrics (October, 2013). Jérôme Petazzoni. <http://blog.docker.com/2013/10/gathering-lxc-docker-containers-metrics/>
- [11] Removing SUID and SGID flags off binaries (August, 2008). Eric Thern. <http://www.thern.org/projects/linux-lecture/intro-to-linux/node10.html>
- [12] Announcing Docker 1.2.0 (August, 2014). Victor Vieux. <http://blog.docker.com/2014/08/announcing-docker-1-2-0/>
- [13] Having non-root privileges on the host and root inside the container #2918 (November, 2013). [GitHub issue] <https://github.com/docker/docker/issues/2918>
- Support for user namespaces #4572 (March 2014). [GitHub issue] <https://github.com/docker/docker/pull/4572>
- Proposal: Support for user namespaces #7906 (September, 2014). [GitHub issue] <https://github.com/docker/docker/issues/7906>
- Issue 8447: syscall, os/exec: Support for User Namespaces (July, 2014) [Google Code issue] <https://code.google.com/p/go/issues/detail?id=8447>
- [14] Introduction to unprivileged containers (January, 2014). Stéphane Graber <https://www.stgraber.org/2014/01/17/lxc-1-0-unprivileged-containers/>
- [15] Docker 0.9: Introducing Execution Drivers and libcontainer (March, 2014). Solomon Hykes <http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/>
- [16] A simple helper script to help people build seccomp profiles for Docker/LXC (November 2013). Martijn van Oosterhout. <https://github.com/docker/docker/blob/487a417d9fd074d0e78876072c7d1ebfd398ea7a/contrib/mkseccomp.pl>
<https://github.com/docker/docker/blob/487a417d9fd074d0e78876072c7d1ebfd398ea7a/contrib/mkseccomp.samp>
- [17] Announcing Docker 1.2.0 (August, 2014). Victor Vieux. <http://blog.docker.com/2014/08/announcing-docker-1-2-0/>
- [18] Docker Container Breakout Proof-of-Concept Exploit (June, 2014). James Turnbull <http://blog.docker.com/2014/06/docker-container-breakout-proof-of-concept-exploit/>
- [19] docker2docker GitHub repository. Jérôme Petazzoni. <https://github.com/jpetazzo/docker2docker>