Sommaire

Liste des Figures

Liste des Tables

Introduction générale

Il était une fois un jeune développeur a passé toute une nuit à coder tranquillement sur son ordinateur. Après des heures de travail, il a testé son application, l'application était là, et elle fonctionnait à merveille ! Le lendemain le jeune développeur a transféré son projet sur l'ordinateur de son collègue. Mais malheureusement, l'application ne fonctionne pas ! Une explication a ce problème ce que l'application de notre codeur ne fonctionne pas sur l'ordinateur de son ami à cause d'un problème d'environnement. Deux systèmes peuvent avoir des différences de version sur les dépendances ou encore des bibliothèques manquantes.

Dans l'exemple ci-dessus, le problème se limite à 2 systèmes, mais imaginez une équipe de 10 personnes avec des ordinateurs sous OS X, Linux ou même Windows, un serveur de test sous Ubuntu 12.04, et un serveur de production sous CentOS 7. Assurer le fonctionnement de leur application sur tous ces environnements peut s'avérer être un vrai cauchemar !

les docker viennent proposer une solution à ce type de problème (et au problème d'hétérogénéité) en général. Il s'agit d'une plateforme qui permet d'exécuter le code à l'intérieur d'un conteneur indépendamment de la machine sur laquelle on est ! Un conteneur ressemble à une machine virtuelle sauf qu'il n'inclut pas un système d'exploitation ce qui lui permet de s'exécuter en quelque secondes et d'être beaucoup plus léger. Mais, cette plateforme a des insuffisances. En effet, la création des conteneurs nécessite la saisie des plusieurs lignes de commandes. Par la suite l'utilisateur va mettre beaucoup de temps pour finir une tâche (création, modification, suppression) et beaucoup de concentration. Le temps sera facturé et l'entreprise va subir les coûts. Afin, d'optimiser les coûts et éviter que les employés se lassent, on a besoin de trouver une solution Web pour la gestion des conteneurs du docker, qui permet de créer , modifier ,publier et supprimer ces conteneurs tout en diminuant le temps alloué pour effectuer une telle tâche en plus d'un tableau de bord. D'ou, notre projet consiste à créer une interface web qui permet de gérer les conteneurs du docker avec un minimum des lignes de commandes et par la suite moins du temps consommé et par conséquent on peut créer plusieurs conteneurs sur un même serveur.

Pour réaliser notre projet on va organiser notre travail comme suit : un premier chapitre portera sur la prise de connaissance et l'analyse de l'existant pour mieux définir les besoins et les fonctions de notre application. Dans un deuxième chapitre, nous nous intéresserons à l'état de l'art. Analyse et spécification des besoins seront présentées dans un troisième chapitre. La conception de notre application sera abordée dans un quatrième chapitre. Nous examinerons la phase de modélisation théorique de l'application. Enfin, dans le chapitre V consacré à la réalisation, nous proposons une description détaillée des outils utilisés pour développer l'application web, l'architecture du système et le matériel de déploiement de l'application. Nous couronnons notre travail par une conclusion générale.

Chapitre 1 : Etude de l'existant

Introduction :

Ce chapitre sera consacré à présenter notre projet de fin d'études, ses objectifs et ses spécifications. En premier lieu, on va commencer par une présentation de notre société de stage. Par la suite, nous on citera les deux parties de notre application.

I. Présentation de la société Devercipe

1) Présentation générale

Devercipe est une entreprise d'informatique créative qui aide les développeurs et les startups à créer de très bons produits. Elle offre des services de formation, de conseil, d'externalisation, de conception d'expérience utilisateur, de réflexion sur les produits, de recherche d'utilisateurs, ainsi que la création d'applications Web et mobiles et d'informatique en nuage. Son siège social est à Tozeur, en Tunisie, fondée en Février 2017 et son objectif est de faire connaître les dernières technologies et cultures de coupe aux entreprises technologiques.

La figure suivante présente l'organigramme de Devercipe.

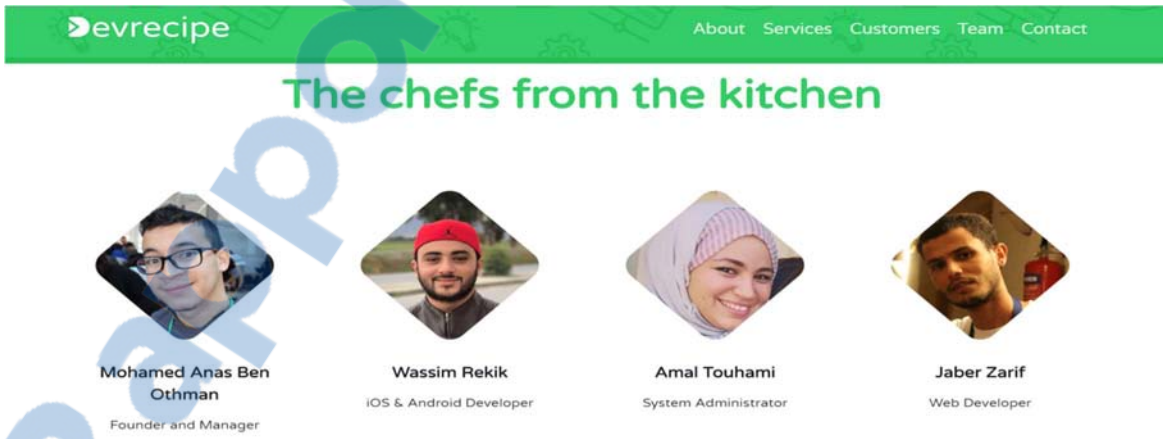

Figure 1 : Chronogramme de devercipe

2) Domaine d'activité de Devercipe

La société Devercipe développe des systèmes d'information orientés vers le commerce électronique sur internet, efficaces et conçus sur la base des meilleures pratiques du commerce en ligne.

En plus Devercipe organise :

- ✓ La vente des programmes informatiques.
- ✓ Les services après-vente.
- ✓ La maintenance et la réparation.
- ✓ Le développement des logiciels.
- ✓ La création des animations 3D.
- ✓ La conception et la réalisation des projets internet/intranet.

3) Les outils informatiques de Devercipe

La société dispose de différents logiciels et des gestionnaires qui peuvent être classés comme suit :

- ✓ Les systèmes d'exploitation :
  - UNIX pour serveur DELL
  - Windows XP et Windows 7 pour les micros ordinateurs.
- ✓ Les gestionnaires de réseaux :
  - TCP/IP pour tous les postes
  - NETWARE

Les systèmes de gestion de base de données :

- Oracles
- MySQL

- ✓ Les langages de programmation :
  - PHP, HTML, CSS.
  - JQUERY, C, C++.
  - JAVA, JEE
  - DELPHI
  - 3DMAX
  - UML
  - ACTIONSCRIPT 2 ,3 .
- ✓ Programmes de gestion de base de données :
  - EASY PHP 1.7, 1.8, 5.2 ……
  - ECLIPSE

II. Critique de l'existant :

A fin de réduire le cout d'utilisation de deux serveurs, la société entend créer deux sites d'environnement différent venant d'un même serveur en utilisant comme logiciel le Docker.[1] Mais le problème dans ce cas, ce que à chaque fois pour créer les conteneurs, on a besoin de saisir plusieurs lignes de commandes et par la suite on met beaucoup de temps pour finir une tâche (création, modification, suppression) et beaucoup de concentration.

III. Solution proposée :

Il s'agit de trouver une solution Web pour la gestion des conteneurs du docker, qui permet de créer , modifier ,publier et supprimer ces conteneurs tout en diminuant le temps alloué pour effectuer une telle tâche en plus d'un tableau de bord. D'ou, on propose la création d'une interface web qui permet de gérer les conteneurs du docker avec un minimum des lignes de commandes et par la suite moins du temps consommé et par conséquent on peut créer plusieurs conteneurs sur un même serveur.

Conclusion :

Ce chapitre a donné l'occasion de présenter la société accueillante, Dynamique Service Plus, et d'introduire le sujet.

Après avoir mis le travail dans son contexte, nous présenterons dans le chapitre suivant une étude théorique des différentes notions en rapport avec notre sujet.

I. [1] Une technologie qui a pour but de faciliter les déploiements d'application, et la gestion du dimensionnement de l'infrastructure sous-jacente. Docker : tout ce qu'il faut savoir Antoine Crochet-Damais

Chapitre 2: Etat de l'art

Introduction :

Après avoir faire une étude de l'existant et faire le choix de application web, ce chapitre portera sur une étude théorique sur les technologies de base adoptées pour réaliser ce projet.

I. La technologie de Docker :

1) Définition du Docker :

Il s'agit d'un logiciel libre pour automatiser le déploiement d'applications dans des conteneurs logiciels. La firme de recherche sur l'industrie 451 Research a définit le Docker comme un outil qui permet d'empaqueter une application et ses dépendances dans un conteneur isolé, Ce logiciel pourra aussi être exécuté sur n'importe quel serveur Linux. Et par la suite, la portabilité d'exécution et la flexibilité d'une application s'étendent, que ce soit sur la machine locale, , une machine nue, un Cloud public ou privé etc.

Le Docker est utilisé pour créer et gérer des conteneurs, ce qui permet la simplification de la mise en œuvre de systèmes distribués en permettant l'éxécution de multiples applications, tâches de fond et autres processus. l'exécution sera de façon autonome sur une seule machine physique ou à travers un éventail de machines isolées. Ceci permet le déploiement des nœuds en tant que ressources sur besoin.

La figure suivante montre l'architecture du Docker :

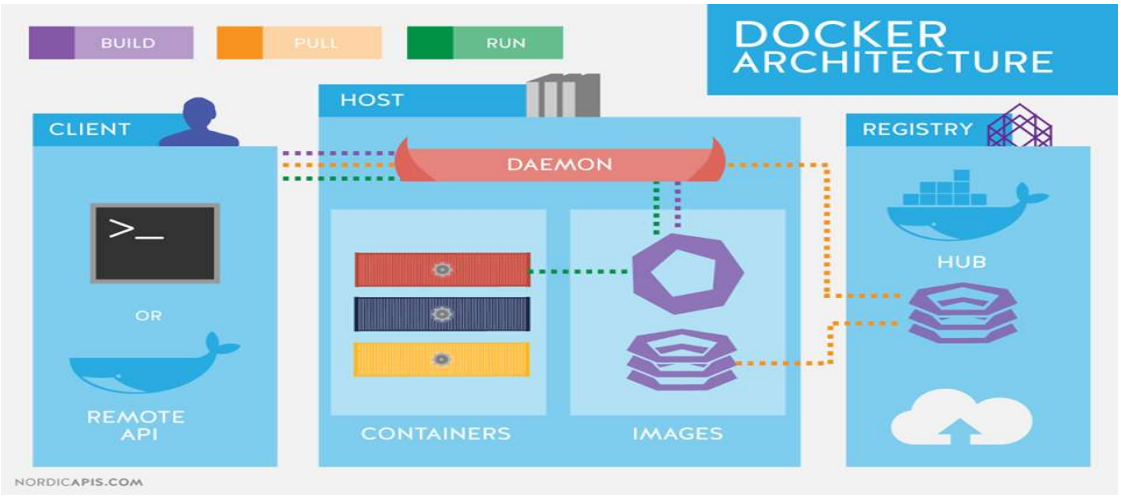

Figure 2 : Architecture du docker

Docker contient trois composants[2] :

- **Docker Daemon :** Il a la fonction d'un root et il orchestre essentiellement tous les conteneurs
- **Docker images :** Il s'agit des images pour système d'exploitation virtuels, techniquement Docker images ont de manière moins d'empreinte que les images d'OS réels. Les systèmes de fichiers sont sauvegardés et empilés, et par la suite ils peuvent être unifiés à la dernière révision à tout moment. Toutes ces opérations d'écriture sont également mises en cache vu la nature du système des fichiers. À savoir , si Docker découvre qu'on ne dispose 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 permet un énorme avantage de performance sur le temps de provisionnement. En plus de tous ces avantages, on peut échanger les images avec d'autres personne.
- **Docker repositories :**

C'est le composant écosystème du docker. Autrement dit c'est le référentielle. Les images que vous détenez doit être sauvegardées quelque part, que ce soit privé ou public.

2) Histoire du docker :

Dans le cadre d'un projet interne de dotCloud, l'entreprise française Solomon Hykes a développé Docke. cette entreprise propose une plate-forme en tant que service, avec les contributions d'Andrea Luzzardi et Francois-Xavier Bourlet. Docker est une évolution basée sur les technologies propriétaires de dot Cloud.

Depuis Mars 2013? Docker a été distribué en tant que projet open source.

- ✓ Au 18 novembre 2013, le projet a été favorisé plus de 7 300 fois sur GitHub (Il présente le 14e projet le plus populaire), avec plus de 900 forks et 200 contributeurs.
- ✓ Au 9 mai 2014, le projet a été favorisé plus de 11 000 fois sur GitHub, avec plus de 1 900 forks et 420 contributeurs.
- ✓ En octobre 2015, le projet a été favorisé plus de 25 000 fois sur GitHub, avec plus de 6 500 forks et 1 100 contributeurs.

[2] http://www.supinfo.com/articles/single/104-qu-est-que-docker-pourquoi-est-il-different-machines-virtuelles

- ✓ En septembre 2016, le projet a été favorisé plus de 34 000 fois sur GitHub, avec plus de 10 000 forks et 1 400 contributeurs[3]

3) Pourquoi utiliser le docker :

Le choix du Docker est justifié par:

- La distribution des applications est facilitée
- Les applications en Dev/Qualif/Prod ont le même Comportement.
- Rapide déploiement, lancement et arrêt.
- Linux et Windows (en preview dans Windows Server 2016)
- Reconstruire d'un container à partir d'un simple fichier"Docker file"
- Gérer des containers identiques sur toutes les plateformes en utilisant peu d'outils.
- Des API sont disponibles pour piloter l'ensemble depuis d'autres applications.

4) Les outils Docker libres[4] :

- DOCKER MOTEUR/CLIENT : permet l'utilisation de docker en cli.
- DOCKER MACHINE : Permet le lancement de Docker, Il de crée automatiquement un environnement virtuel.
- DOCKER COMPOSE : permet le lancement des applications multi-containers.
- DOCKER SWARM : permet la gestion des containers Docker dans un cluster.
- DOCKER REGISTRY : c'est une application pour gérer des images locales.
- NOTARY : Il s'agit des outils pour la signature des images par le fournisseur et la vérification de l'intégrité par le client.
- LIBNETWORK : c'est un outil d'abstraction réseau qui permet la communication et l'isolation entre les containers, même sur des nœuds différents.

[3] https://fr.wikipedia.org/wiki/Docker_(logiciel)
[4]http://cache.media.eduscol.education.fr/file/Numerique/06/5/Architecture_Microservices_avec_Docker_545065.pdf

La figure ci dessous montre la configuration le docker client :

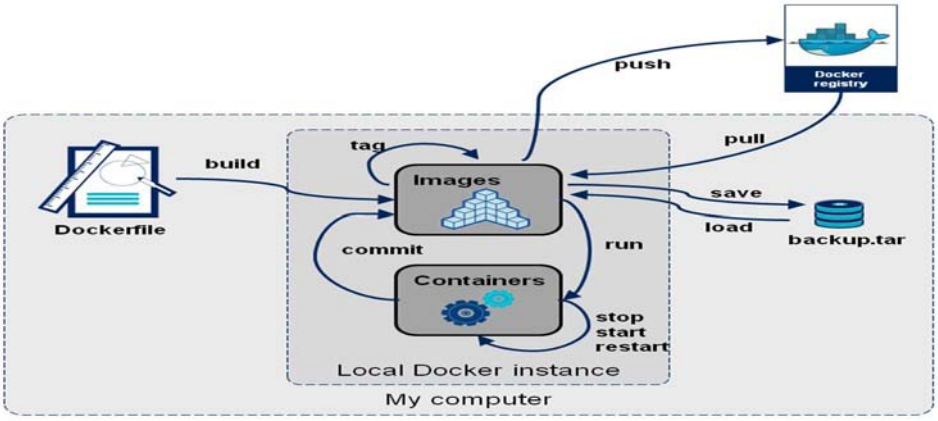

Figure 3 : Docker Moteur / client

5) Caractéristiques de Docker [5](Florent Jaillet, Novembre 2015)

Les conteneurs ont plusieurs caractéristiques :

- Combiner et adapter des différents concepts et outils éprouvés (conteneurs, scripts, gestion de version, gestion de paquets) avec une approche DevOps
- Principal intérêt : utilisation simple et auto-suffisance
- Logiciel libre écrit en Go
- Uniquement des conteneurs pour le noyau Linux, pour l'instant
- Disponibilité sous Linux, sous Windows et Mac via la Docker Toolbox fournissant une machine virtuelle Linux minimaliste, et sous de nombreuses infrastructures nébuleuses.

6) Les conteneurs :

Fondamentalement, un conteneur est le nouveau terme de fantaisie autour du niveau du système d'exploitation de virtualisation. Il permet de créer des instances dans un espace utilisateur isolés au lieu d'un seul. En d'autres termes, le conteneur partage le matériel et rend disponible de nombreux opérateurs sur le noyau lui-même plutôt que d'une autre couche comme ESX, Hyper-V font. Ces instances d'espace utilisateur sont souvent appelés Virtualisation Engine(VE) ou Serveurs Privés Virtuels(VPS). Alos Docker n'est pas le seul qui peut le faire, mais de nombreux projets ont le même concept.

[5] Article docker pour le développement logiciel et recherche reproductible par l'auteur "florant jaillet"

L'utilisation de la technologie de conteneur de Docker permet d'étendre des systèmes distribués de façon à ce qu'ils s'exécutent de manière autonome a partir d'une seule machine physique ou d'une seule instance par nœud.

II. Comparaison entre virtualisation classique et Docker:

A fin de comparer docker et la virtualisation classique, les hyperviseurs doivent créer une copie complète du système d'exploitation. Cette copie fonctionne sur un espace matériel virtuel. L'hyperviseur sera donc responsable de tous les échanges de données. Exécuter plusieurs machines virtuelles sur un même serveur requiert de hautes performances et de ressources suffisantes pour assurer le fonctionnement de plusieurs machines virtuelles. (chose difficile avec la virtualisation classique)

Tandis que les conteneurs permet :

- Exécution avec une vitesse plus rapide probablement 2 à 3 secondes au lieu de minutes
- Utilisation de la mémoire du conteneur au lieu de disques
- Encombrement minimal - à la place de milliers de Mo, il est des dizaines de Mo
- De nombreuse exécution - Possibilité d'éxécuter 100 conteneurs en parallèle sur une seule boîte.
- Résilience - En cas d' un accident , on peut relancer immédiatement
- Sécurité - Dans le cas ou un conteneur à une attaque DDOS, les autres conteneurs ne seront pas affectés
- Sauvegarde - en utilisant de fichiers file-system

la différence de vitesse d'exécution entre la virtualisation et le Docker :

virtualisation		**Docker**
		
déploiement dans des minutes	⟹	déploiement dans quelques secondes
pendant un moins	⟹	pendant quelques heures/minutes
Agile		Devops

Docker étend le format de conteneur Linux standard, LXC, avec une API de haut niveau proposant une solution de virtualisation qui effectue les processus de façon isolée. Docker fait usage de LXC, Cgroups, et le noyau Linux lui-même. Contrairement aux machines virtuelles traditionnelles, un conteneur Docker ne contient pas de système d'exploitation, se basant sur les fonctionnalités du système d'exploitation fournies par l'infrastructure sous-jacente.

le schéma ci-dessous compare le mode Docker au celui de virtualisation classique :

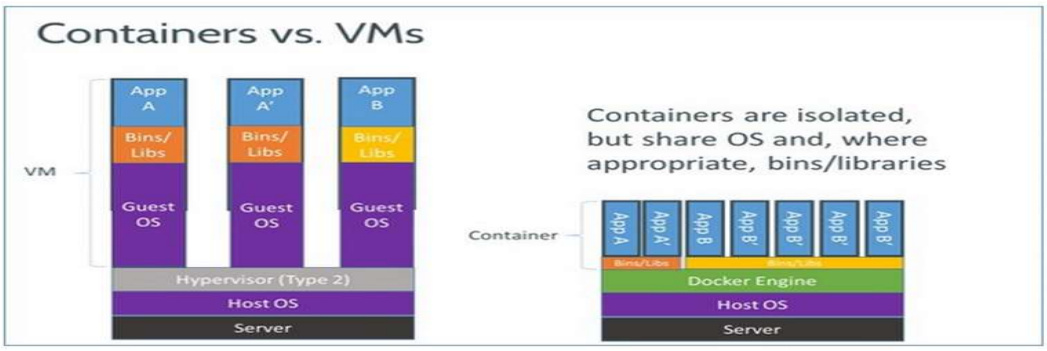

Figure 4 : comparaison au mode de virtualisation classique

III. Les avantages de la solution de conteneurs/docker[6] :

Dans le domaine du Web ainsi que dans le domaine de l'IT, l'utilisation de Docker possède plusieurs avantages. Les services IT fournissent de plus en plus de services orientés web, les conteneurs pourraient être un élément clef permettant la transitions des services IT vers le cloud.

1) Légèreté des conteneurs

Par opposition à un serveur virtuel sous Linux, le conteneur n'a besoin que de quelques centaines Mo de disques. De même l'empreinte mémoire est réduite, car nous n'avons que de la mémoire employée pour l'application (pas de couche OS). Il est ainsi avec démarrage plus rapide aussi le conteneur peut être facilement déplacé d'une machine à une autre.

2) Rapidité et facilité de déploiement des applications

Après avoir récupérer un template en une commande, et afin d'exécuter le conteneur il nous suffit une autre commande, celui-ci démarre en quelques secondes. Lors du démarrage, des paramètres peuvent être ajoutés en les transmettant au conteneur. Par exemple, la spécification de l'accès à la base de données ou d'autres paramètres d'automatisation.

L'idéal consiste à mettre en place une gestion de configuration, grâce à elle, les conteneurs deviennent autonomes. Ces conteneurs seront gérés comme un package logiciel. Cette facilité aide le développeur à provisionner les environnements qui sont déployés en production par exemple sans connaissance particulière d'infrastructures ou d'administration système.

3) Portabilité et multicloud

Il y a peu de formats d'enveloppes virtuelles qui soient nativement multi-Cloud. Il est toujours compliqué de récupérer sur sa ferme VMware son POC fait sur AWS, ou encore migrer sa machine virtuelle Open stack sur une Ressource Group Azure.

[6] https://actu.alfa-safety.fr/devops/conteneur-docker-fonctionnement-et-avantages-pour-heberger-ses-applications/

IV. Les Facteurs et les concepts :

1) Les facteurs : [7]

a. Code base

- Un logiciel de suivi de version (git, mercurial, ...) gère les code.
- Une application = code source

b. Dependencies

- Préciser clairement toutes les dépendances.
- Le système cible n'est pas censé inclure de programme pré-installé.
- absence de dépendances implicites.

c. Config

- On considère une configuration, tout ce qui diffère d'un environnement à l'autre (dev, qualif, prod, autre site).
- les élément de configuration doivent être passé par des variables d'environnement.
- Aucune référence ne figure à la configuration dans le code.

d. Backing Services

- Il s'agit d' une ressource externe au conteneur (base mysql, smtp, activemq, memcache, ...).
- L'accès à ces ressources se fait en paramètre. Pas de différence entre les services locaux et distants.

e. Build, release, run

- L'application et l'environnement sont recréés avant tout déploiement d'une nouvelle version.
- On apporte aucune modification sur l'application déployée.
- A chaque version déployée on accorde un numéro de version unique (timestamp, numero de commit, ...).

7 http://cache.media.eduscol.education.fr/file/Numerique/06/5/Architecture_Microservices_avec_Docker_545065.pdf

f. Création de conteneurs (Build)

- Notion d'image qui permet une construction multicouche des conteneurs
- Il est possible de créer une nouvelle image tout en modifiant une image existante
-

Build	Génération automatisée d'images avec langage de script spécialisé dans un Docker file, exemple : FROM docker/whalesay:latest RUN apt-get -y update && apt-get install -y fortunes CMD /usr/games/fortune -a | cowsay
Commit	Création d'une image à partir des modifications lors de l'exécution d'un conteneur
History	Affichage de l'historique d'une image (couches avec possibilité de dépiler)
save, load, export, import	Sauvegarde et rechargement d'images et de conteneurs

Tableau 1 : Creation de conteneurs

g. Dépôts de conteneurs (Ship)

Search	Rechercher d'images sur le **Docker Hub**
pull, **push**	Récupérer / envoyer d'image depuis / sur un dépôt (par défaut le Docker Hub)

Tableau 2 : Dépots de conteneurs

- Plusieurs images officielles (gcc, python, ruby, java, golang, , haskell, centos, perl, , debian, fedora, ubuntu, …)
- Plusieurs images publiques d'utilisateurs ou d'organisations
- Versionnage des images

•

• Possibilité d'employer son propre dépôt avec Docker Registry, notamment disponible sous forme d'image Docker officielle.

h. Exécution et supervision de conteneurs (Run)

Daemon	Lancer la partie serveur (démon) de l'architecture client/serveur de Docker
Run	Lancer une commande dans un nouveau conteneur construit à partir d'une image
start, stop, restart, pause, unpause, kill	Contrôler l'exécution des conteneurs
ps, logs, top, stats	Surveiller des conteneurs

Tableau 3 : Exécution et supervision de conteneurs

• Plusieurs conteneurs se lient via des liens réseau

• Le partage et la persistance de données se font en utilisant de volumes pour

• Coordonner l'exécution de multiples conteneurs (Docker Compose) est possible

i. Processus

- L'application est effectuée dans l'environnement d'exécution en tant qu'un ou plusieursprocessus.
- Stockage de touteslesdonnéesdansune ressource externe (base de données).
- Le non stockage local des variables de sessions utilisateurs.

j. Port binding

L'application donne un service qui écoute sur un port.

k. Concurency

- Possibilité de mettre à l'echelle chaque application. Pour répartir la charge, les conteneurs peuvent être lancés x fois.
- Le programme dans le conteneur ne peut pas être lancé en tâche de fond.

- L'arrêt du conteneur suite à l'arrêt du programme.

l. Disposability

- Le conteneur doit être à usage unique.
- Il peut être lancé très rapidement.
- Un arrêt inopportun ne doit pas détériorer les données.

m. Dev/prod parity

- dés qu'il finit la saisie du code le développeur doit pouvoir le déployer rapidement.
- Le responsable du dévéloppement doit être plus proche du déploiement (DevOps).
- Maintenir la production et le développement aussi semblables que possible en utilisant les mêmes outils. Pour limiter les surprises en production, éviter de prendre des backends différents en prod et en dev (ex: base de données, ...)

n. Logs

- Pour la visualisation et l'archivage à long terme, les applications doivent externaliser leurs journaux
- Posibilité d'afficher les journaux dans la sortie standard de l'application, mais pas dans un fichier du conteneur.

o. Admin process

- Les commandes d'administration doivent être saisies dans un environnement semblables aux autres processus d'exploitation.
- Mêmes variables d'environnement, même conteneur, mais en mode conversationnel.

2) Concepts[8] :

a. Layers

- Les conteneurs et leurs images sont dissociés en couches (layers)
- Les layers peuvent être remployés entre différents conteneurs
- Optimisation de la gestion de l'espace disque.

8 https://osones.com/formations/pdf/docker.pdf

a.1 Layers : Une Image

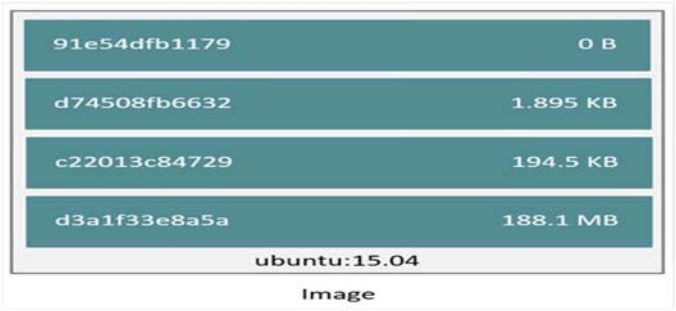

Figure 5 : décomposition d'une image en layers

a.2 Layers : Un Conteneur

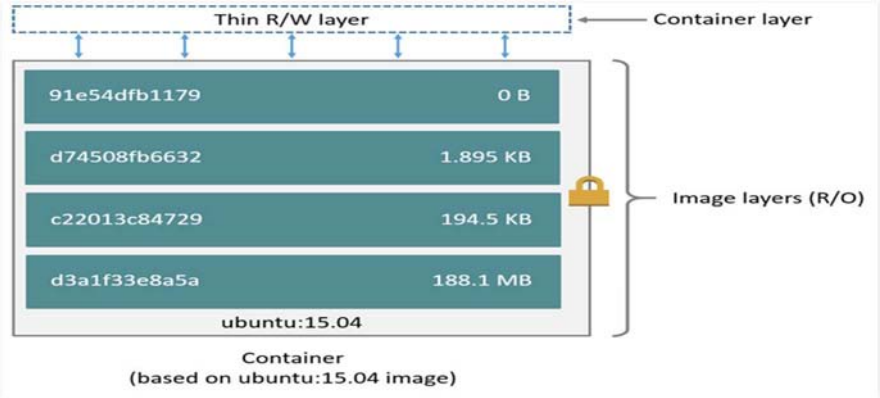

Figure 6 : Un conteneur est la somme d'une image et un layer r/w

a.3 Layers : Plusieurs Conteneurs

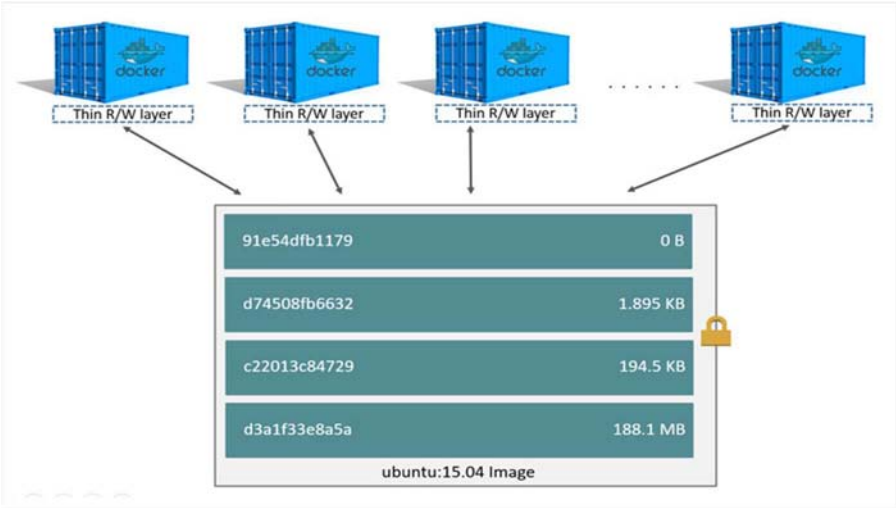

Figure 7 : Une image pour créer plusieurs conteneurs

a.4 Layers : Répétition Des Layers:

Figure 8 : Les layers sont indépendants de l'image

b. Stockage :

Le stockage se fait pour les Images Docker, données des conteneurs, volumes ainsi pour des multilples backends (extensibles via plugins) tels que:

- ✓ AUFS
- ✓ Device Mapper
- ✓ Over lay FS
- ✓ NFS(Via plugin Convoy)
- ✓ Gluster FS(Via plugin Convoy)

b.1 Stockage AUFS

- A unification file system
- Stable : moyenne performance écriture
- Non intégrédans le KernelLinux(mainline)

b.2 Stockage Device Mapper :

- Basé sur Logiciel Volume Manager
- Thin Provisionning et snapshot
- Intégré dans le Kernel Linux (mainline)
- Stable : moyenne performance écriture

b.3 Stockage Over layFS :

- Successeur de AUFS
- Performances accrues
- Relativement stable mais moins éprouvé que AUFS ou Device Mapper

b.4 Stockage PLUGINS :

- Étendre les drivers de stockages disponibles
- Utilisation des systèmes de fichier distribués (GlusterFS)
- Partage des volumes entre plusieurs hôtes docker

c. Volume :

- Assurer une persistance des données
- Le volume est indépendent des conteneur et des layers
- On distingue deux types de volumes :
  - ✓ Conteneur : Les données sont stockées dans un data conteneur
  - ✓ Hôte : Montage d'un dossier de l'hôte docker dans le conteneur
- Le volumes peut être partagé entre plusieurs conteneurs
- Volume : PLUGINS

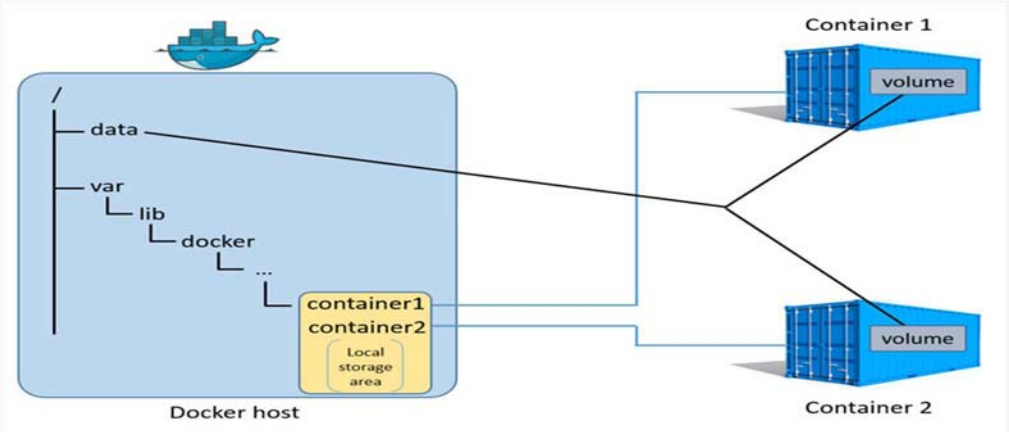

Figure 9 : Un volume monté sur deux conteneurs

- ✓ Permet le partage de volumes entre différents hôtes
- ✓ Fonctionnalité avancées : migration, snapshot, restauration

d. Réseaux :

NETWORK ID	NAME	DRIVER
42f7f9558b7a	Bridge	Bridge
6ebf7b150515	None	Null
0509391a1fbd	Host	Host

Tableau 4 : Réseaux

d.1 Réseau Bridge :

Un réseau Bridgé est le plus employé dans Docker. Les réseaux Bridgés créés par les utilisateurs sont identiques au réseau bridge par défaut créé lors de l'installation de Docker (Docker0). Certes, de nouvelles fonctionnalités sont ajoutées, telle que la gestion du DNS...

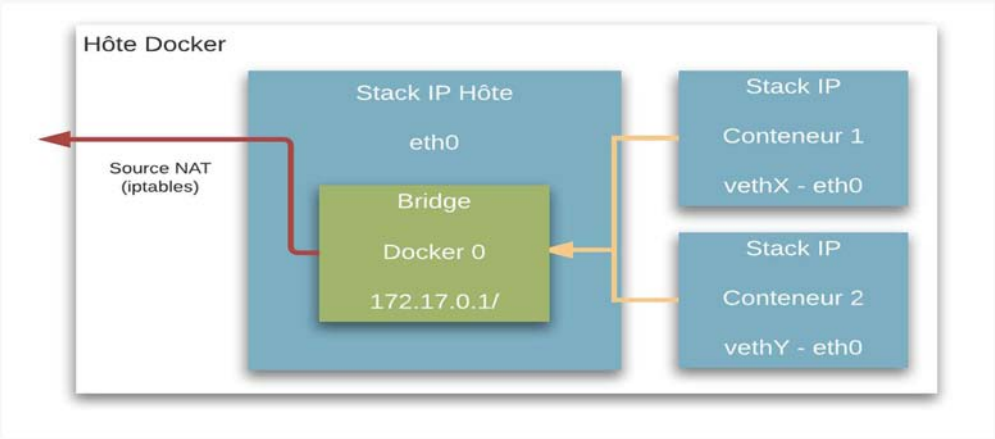

Figure 10 : Réseaux Bridge

d.2. Réseau Les Évolutions :

- Les composants réseau sont refactorisés (lib network)
- Système de plugins : multi host et multi backend (overlay network)

d.3. Réseau Multihost Overlay :

C'est un réseau multi hôtes pour Docker, pour faire communiquer les conteneurs sur un réseau « interne » commun. Bien sûr plusieurs réseaux overlay peuvent être créés.

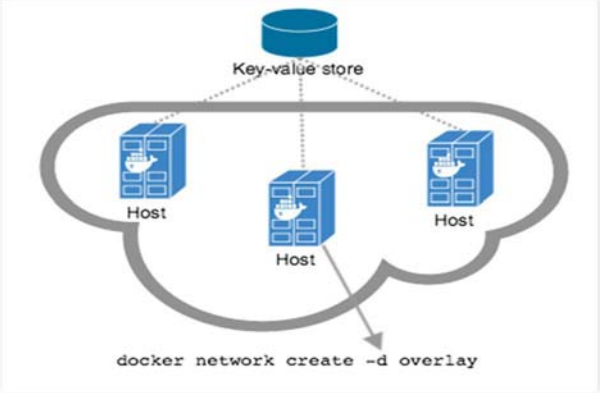

Figure 11 : Réseaux Multi Host

e. Publication De Ports :

- se fait dans le cas d'un réseau diffèrent de l'hôte
- De même, dans le cas où les conteneurs ne sont pas accessibles depuis l'extérieur
- Les ports peuvent être publiés depuis l'hôte vers le conteneur
- L'hôte est utilisée comme proxy de service

Ce schéma montre la publication de ports :

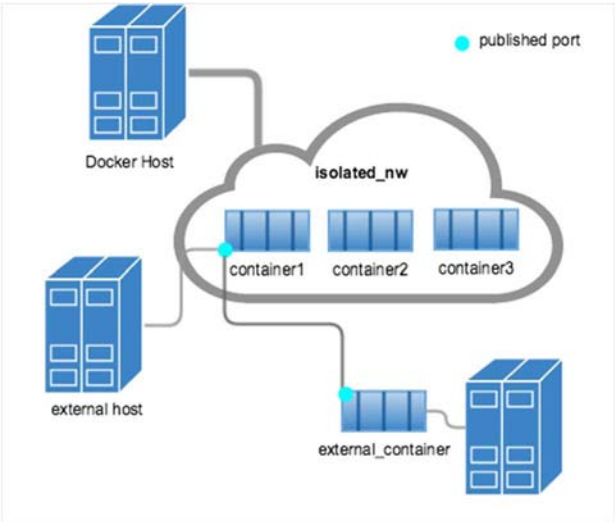

Figure 12 : schéma de publication des ports

f. Links :

- communication des conteneurs d'un même réseau via IP
- Les liens pour lier deux conteneurs par nom
- les liens peuvent être remplacés par les discovery services

Conclusion

Pour facilité la création et l'administration docker nous nous proposons de concevoir et de réaliser une application web qui permet d'éteindre cet objectif on automatisons le processus d'administrations .

Chapitre 3: Analyse et spécification de besoin

Introduction :

La phase d'analyse de spécification représente la première étape du processus de développement que nous avons adopté. Elle permet d'obtenir une idée sur ce que va réaliser le système en termes de métier (comportement du système)

I. Analyse de besoin :

cette partie sera consacrée à poser les bases du recueil des besoins du système à réaliser pour clarifier les besoins des utilisateurs de notre application. Nous allons présenter besoins fonctionnels ainsi que les besoins non fonctionnels

1) Les besoins fonctionnels

il s'agit des fonctionnalités du système. En d'autres terme, ce sont les besoins spécifiant un comportement d'entrée / sorties du système :

a. Coté d'administrateur :

On a besoin de:

- ✓ chercher des conteneurs
- ✓ demander liste des conteneurs
- ✓ passer liste des conteneurs au chef d'équipe
- ✓ Inspecter la situation des conteneurs
- ✓ Gestion des utilisateurs
- ✓ créer un tableau de bard pour afficher :
  - Nombre de conteneur exécuter et nombre de conteneur arrêter
  - Nombre de volume
  - Nombre de Network
  - Affiche le node info

b. coté chef d'équipe :

On a besoin de:

- ✓ Recevoir la liste des conteneurs de son directeur
- ✓ Gérer les conteneurs : cette partie consiste
  - ajouter les conteneurs
  - démarrer les conteneurs
  - résumer les conteneurs
  - arrêter les conteneurs
  - supprimer les conteneurs
  - redémarrer les conteneurs
- ✓ Gérer les images :
  - télécharger l'image à partir docker hub
  - afficher liste des images
- ✓ Gérer le volume :
  - ajouter les volumes
  - afficher liste des volumes
  - supprimer les volumes
- ✓ Gérer Network :
  - ajouter Network
  - afficher liste network
  - supprimer Network

2) Les besoins non fonctionnels :

Il s'agit des contraintes à prendre en considération pour mettre en place une solution adéquate aux attentes des concepteurs et des architectures dynamique. Notre application doit nécessairement assurer:

- ❖ **Contrainte ergonomique :** Le site web doit avoir des interfaces ergonomiques et conviviales faciles à utiliser par n'importe quel utilisateur qu'il soit informaticien ou non.
- ❖ **Contrainte d'évolution :** Les fonctions existantes dans notre site web doivent être évolutives afin de répondre aux changements des besoins des utilisateurs.
- ❖ **Contrainte de maintien :** Les interfaces de site doivent être faciles à maintenir.

❖ **Contrainte temporelle :** Les fonctions existantes dans notre site ne doivent pas prendre beaucoup de temps lors d'exécution.

II. diagramme de cas d'utilisation :

Ce diagramme décrit les utilisations requises d'un système, ou la fonction d'un système. Les acteurs, cas d'utilisation et sujets représentent les principaux concepts de ce diagramme. Un sujet est un système avec lequel les acteurs et autres sujets interagissent.

1) Diagramme de cas d'utilisation Globale :

La figure 13 montre le diagramme de cas d'utilisation global de notre application et qui englobe toutes les fonctionnalités à traiter.

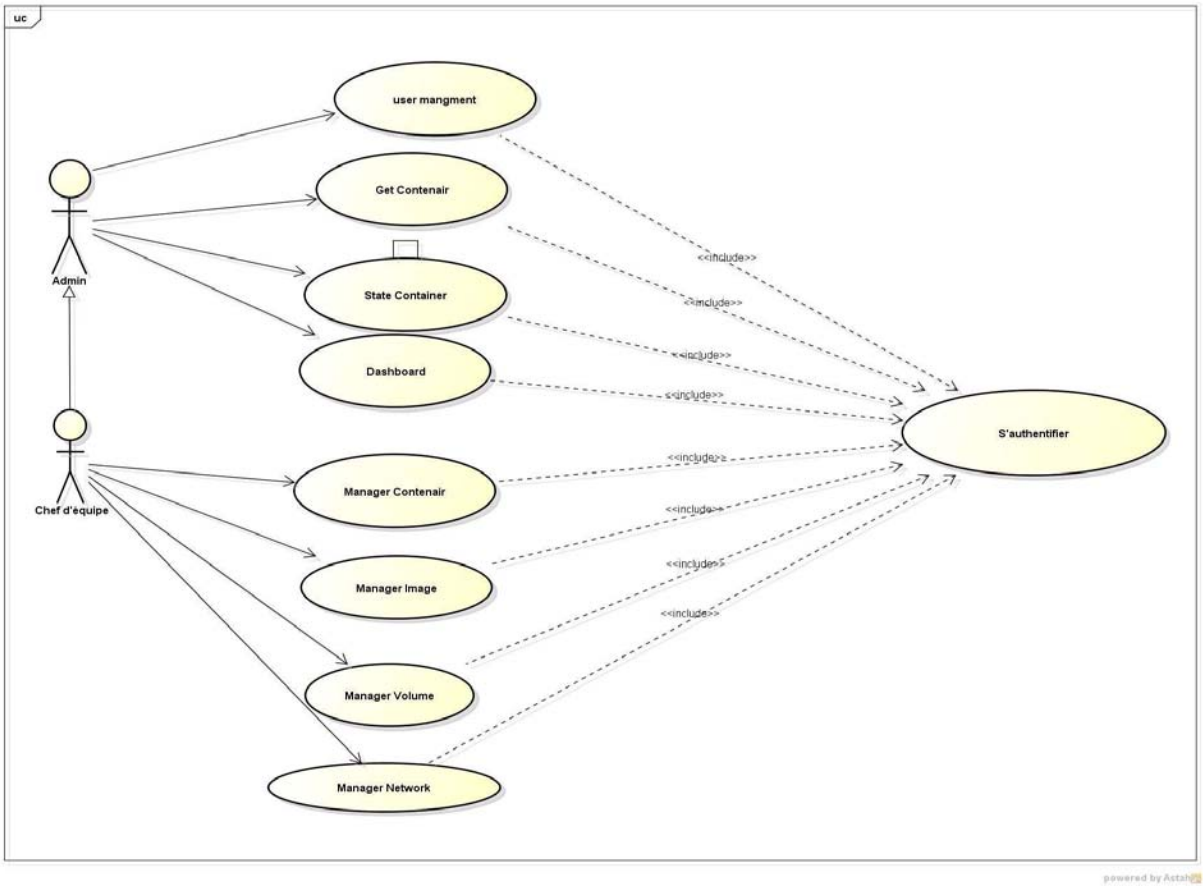

Figure 13 : Diagramme de cas d'utilisation général de système

2) Diagramme de cas d'utilisation détaillé :

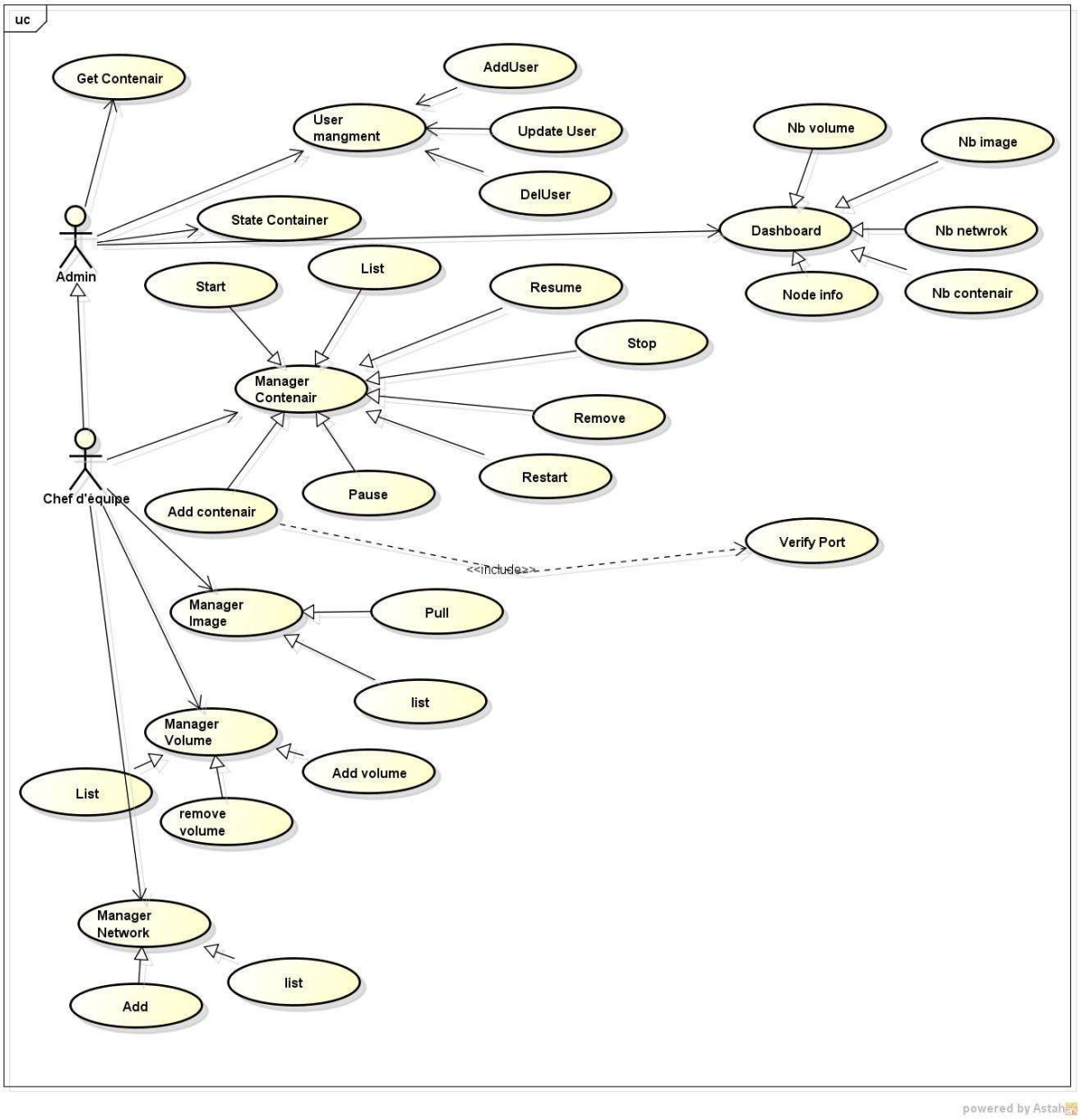

Figure 14 : Diagramme de cas d'utilisation détaillé

a. Cas d'utilisation : "Get Container"

Acteur : Admin

Description : envoyer une demande de création d'un conteneur

Date : 03/09/2017

Pré-conditions : L'utilisateur doit être authentifié en tant que Administrateur

Le scénario nominal :

1. le système affiche le formulaire de la demande
2. L'administrateur rempli les champs du formulaire
3. L'administrateur valide l'envoi de la demande
4. L'administrateur vérifie la liste des conteneurs
5. L'administrateur envoi la liste des conteneurs au Chef d'équipe

Les scénarios alternatifs

- champs obligatoire non saisi
- la demande sera annulée

Post-conditions :

Demande envoyée à l'acteur Chef d'équipe

b. cas d'utilisation "Add Container" :

Acteur : chef d'équipe

Description : créer un nouveau conteneur

Date : 03/09/2017

Pré-conditions : L'utilisateur doit être authentifié en tant que chef d'équipe

Le scénario nominal :

1. Chef d'équipe cherche une image dans docker hub
2. Chef d'équipe choisi l'image
3. Chef d'équipe télécharge l'image
4. Chef d'équipe clique sur "valider"
5. Chef d'équipe affecte le port de conteneur
6. Chef d'équipe crée volume

Les scénarios alternatifs

- Réseaux en panne
- données non valides

Post-conditions :

Conteneur est déjà créé

c. cas d'utilisation " Dashboard " :

Acteur : Admin

Description : Le tableau de bord **permet d'**afficher la liste des conteneurs à exécuter ainsi que la liste de conteneur à arrêter et il permet d'afficher aussi la liste des images, la liste de volumes et la liste de networks.

Date : 03/09/2017

Pré-conditions : L'authentification de l'utilisateur en tant que Administrateur

Le scénario nominal :

1. Administrateur affiche nombre de conteneurs à exécuter sur tableau de bord
2. Administrateur affiche nombre de conteneurs à arrêter sur tableau de bord
3. Administrateur affiche liste d'mages téléchargées sur tableau de bord
4. Administrateur affiche liste de volume sur tableau de bord
5. Administrateur affiche liste de network sur tableau de board

Les scénarios alternatifs

- données non valide
- Administrateur absent

Post-conditions :

Tableau de bord est déjà affiché

d. cas d'utilisation " Add Network " :

Acteur : chef d'équipe

Description : Cette section permet de donner un aperçu sur le comportement réseau par défaut de Docker, y compris le type de réseaux créé par défaut et la manière de créer nos propres réseaux définis par l'utilisateur.

Date : 03/09/2017

Pré-conditions : L'authentification de l'utilisateur en tant que Chef d'équipe

Le scénario nominal :

1. Chef d'équipe connecte un conteneur à un Network
2. Chef d'équipe crée Network
3. Chef d'équipe déconnecte un conteneur d'un network
4. Chef d'équipe affiche liste de conteneur
5. Chef d'équipe supprime un ou plusieurs networks

Le scénario alternatif :

- Réseaux en panne

Post-conditions :

Network est créé

3) Identification des acteurs :

Au niveau de cette section, nous présentons les différents acteurs susceptibles d'interagir avec le système, mais tout d'abord, nous donnons une définition du concept acteur.

Un acteur représente l'abstraction d'un rôle joué par des entités externes qui interagissent directement avec le système étudié.

Notre application exige l'existence de deux types d'acteurs :

- **Administrateur du système :** qui permet de chercher les conteneurs sur docker hub puis lister et passer au chef d'équipe

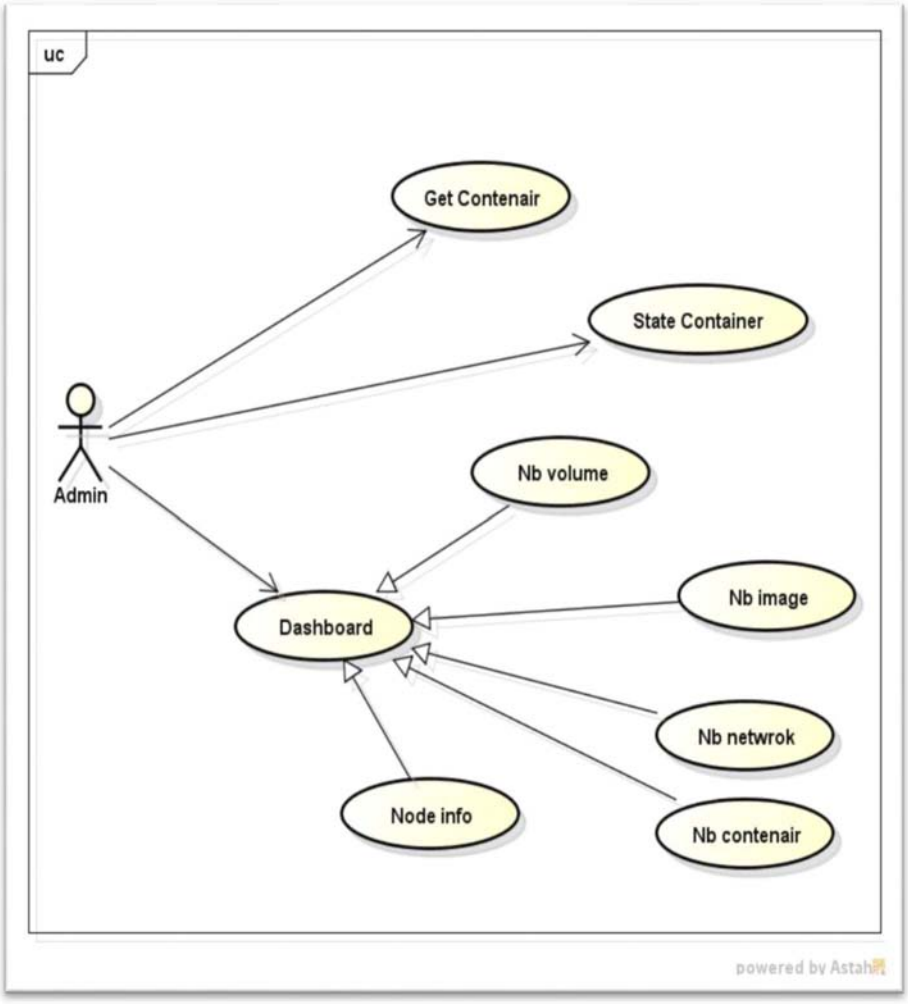

Figure 15 : Diagramme de cas d'utilisation "administrateur de systéme"

- **Chef d'équipe :** son rôle de gérer les conteneurs (création, modification et suppression) suite à une autorisation de l'administrateur

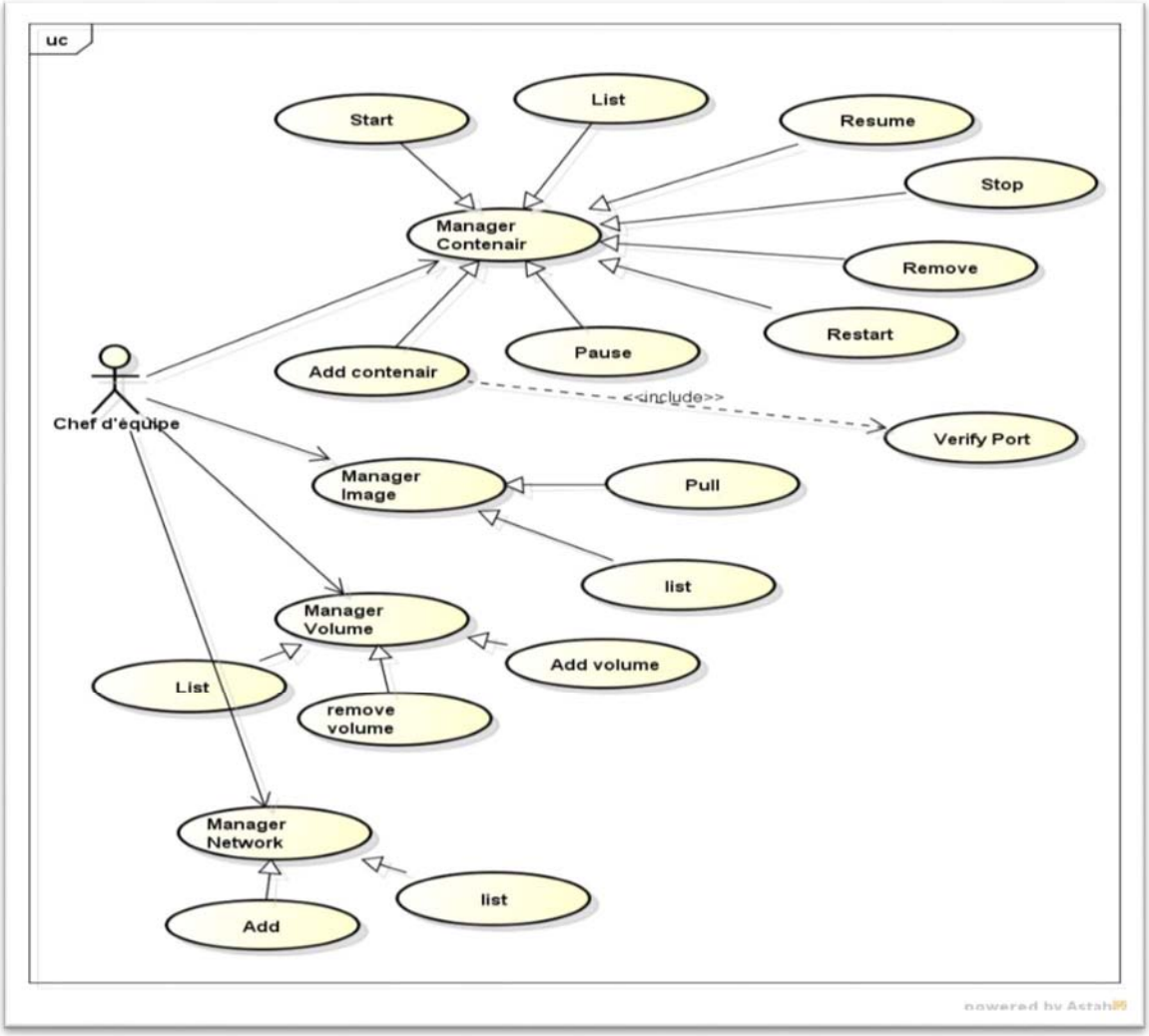

Figure 16 : Diagramme de cas d'utilisation "Chef d'equipe"

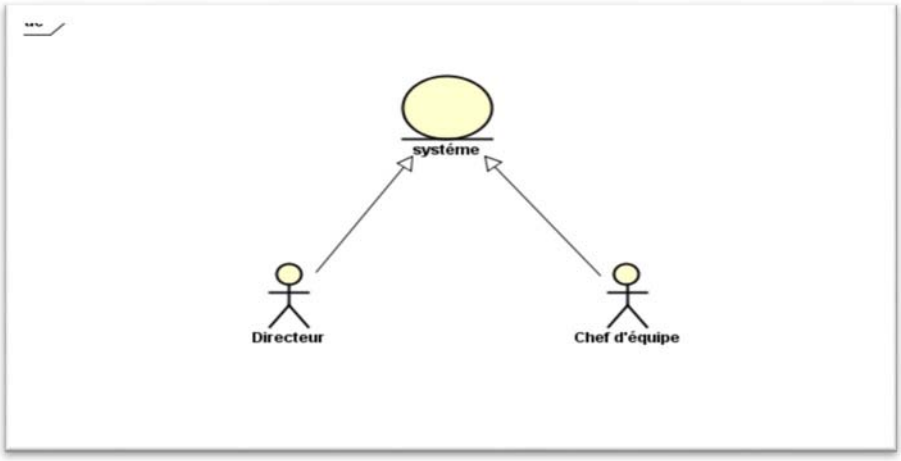

Figure 17 : Diagramme des acteurs

Conclusion :

Dans ce chapitre, nous nous sommes intéressés à l'analyse et à la spécification des besoins. Cette étude incontournable fournit une compréhension plus profonde du système à concevoir. Dans le chapitre suivant nous aborderons la conception de l'application.

Chapitre 4 : Conception

Introduction

La phase de la conception est la phase initiale qui consiste à créer et à mettre en œuvre notre projet. En fait, cette phase est considérée comme importante car il s'agit d'une étape de réflexion dans le processus de développement logiciel après la phase de l'analyse et de spécification.

Dans ce chapitre, nous présenterons en détails la conception de notre projet à travers les diagrammes UML suivants : les diagrammes de séquence, le diagramme de classes et les diagrammes de déploiement.

I. Description des diagrammes :

Dans le précédent chapitre, nous avons présenté le digramme de cas d'utilisation général et digramme de cas d'utilisation détaillé qui décrivent le comportement fonctionnel du système tel qu'il est vu par l'utilisateur, et ceci est insuffisant, pour cette raison, nous allons traiter le digramme de séquence pour décrire le comportement dynamique entre les acteurs et les objets de système.

1) Diagramme de déploiement :

Comme la montre la figure 19 ci-dessous, notre application est une application de type client/serveur, toutes les données sont centralisées sur la machine serveur. Par conséquent l'accès, l'administration, ainsi que la mise à jour des données seront simple.

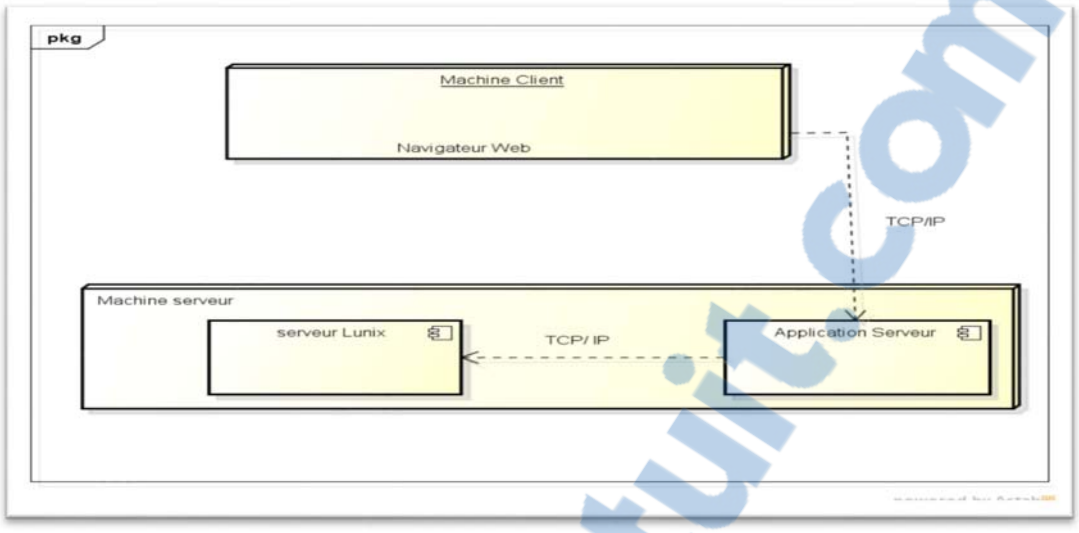

Figure 18 : Diagramme de déploiement

2) Diagramme de classe :

Le diagramme de classes est le diagramme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Il montre la structure interne du système. Il donne une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation. Il s'agit d'une vue statique car le facteur temporel n'est pas pris en considération dans le comportement du système. les classes et leurs relations représentent les principaux éléments de cette vue statique: association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation. Une classe-association a les caractéristiques des associations et des classes : elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations. Une classe-association est caractérisée par un trait discontinu entre la classe et l'association.

Une classe est la description d'un groupe d'objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations ou méthodes) et de relations avec d'autres objets (les associations et les agrégations).

Une classe de conception est composée de :

- **Attribut :** chaque instance d'une classe a le même attribut. cette classe.
- **Méthode :** Il s'agit de définir le comportement d'une classe elle-même, et non pas les comportements de ses instances qui peuvent être différents.

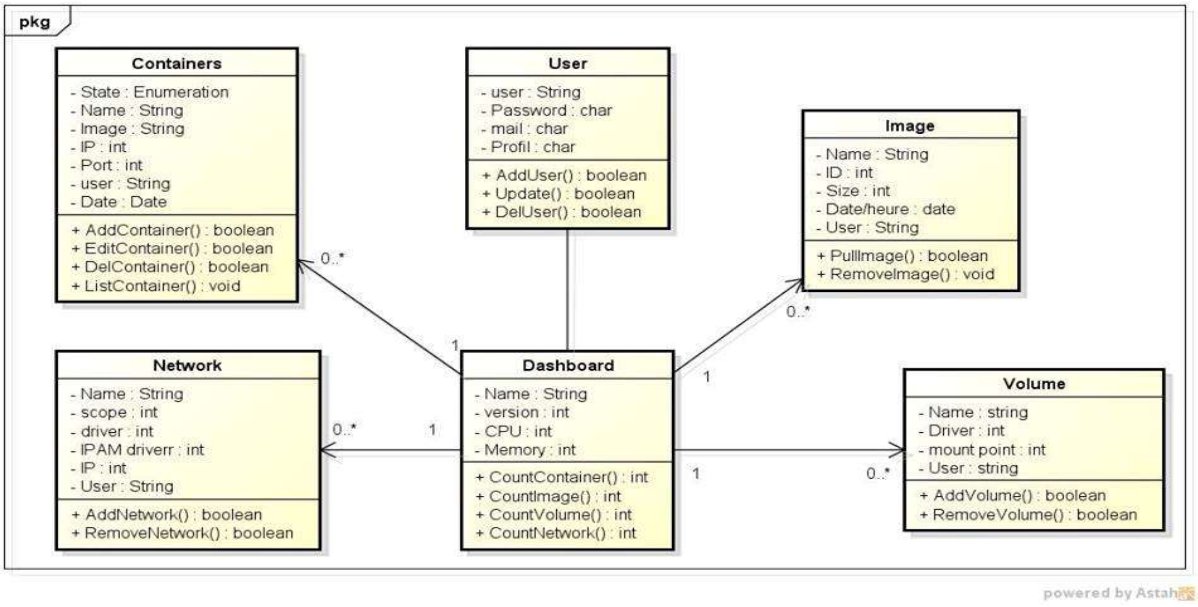

Figure 19 : diagramme de classe

- **Classe Contenaires :** Elle contient comme attributs les informations relatives (....) et les méthodes qu'elle implémente permettant d'ajouter, modifier, supprimer et lister les conteneurs.

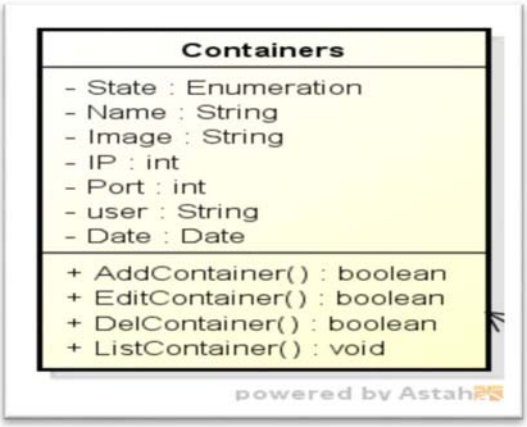

Figure 20 : classe contenaires

❖ **Classe Image :** elle contient comme attributs les informations concernant une image et les méthodes qu'elle implémente permettant de télécharger et supprimer image.

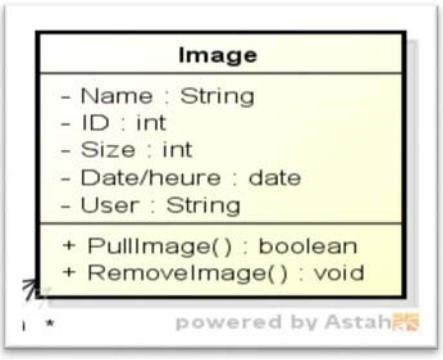

Figure 21 : Classe Image

❖ **Classe User :** elle présente l a liste des utilisateurs à ajouter et contient comme attributs (user, password , mail, profil) et des methodes AddUser ,Update , DelUser**.**

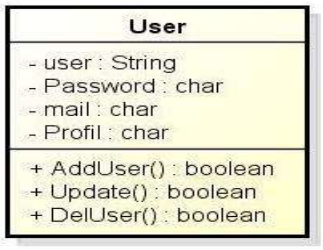

Figure 22 : Classe User

❖ **classe volume :** elle présente la liste de volume et contient comme attributs : Name, driver, Mount point, user et comme méthodes AddVolume( ),RemoveVolume(), pour l'ajout et la suppression

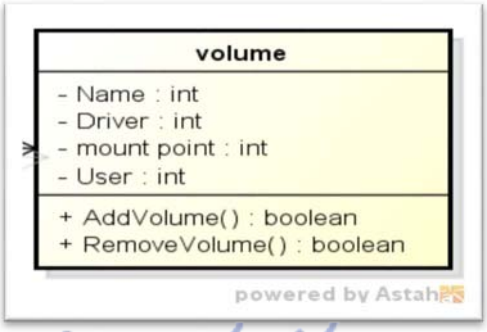

Figure 23 : Classe Volume

❖ **Classe Network :** Elle contient comme attributs les informations nécessaires associées à chaque utilisateur de l'application (name, scope, driver,ip ,user), elle contient aussi des méthodes permettent ajouter Network et supprimer Network.

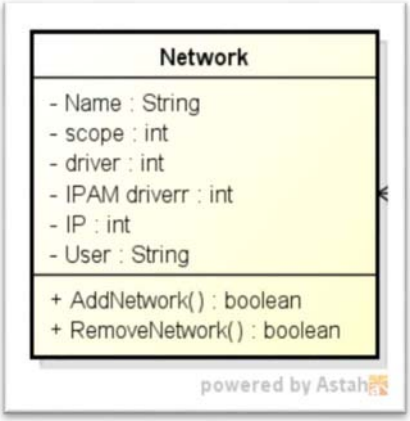

Figure 24 : Classe Network

❖ **Classe Dashboard :**

elle présente les statistiques de notre application et contient comme des attributs (Name , version , CPU, Memory) elle contient aussi des fonctions :

- count contenair pour afficher liste de conteneurs a exécuter
- count image : pour afficher liste d'image
- count voloume : pour afficher liste de volume
- count network : pour afficher liste de network

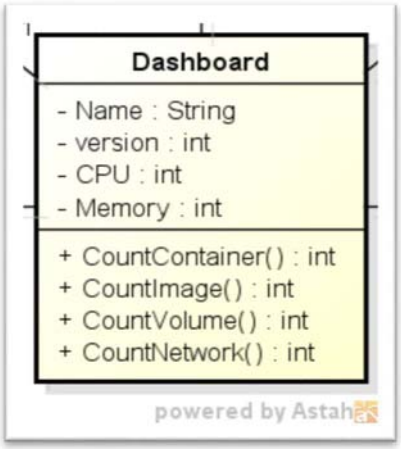

Figure 25 : Classe Dashboard

3) Diagramme de séquence [9]:

Un diagramme de séquence est un document graphique pour montrer des scénarios de cas d'utilisation précis, les événements générés et les interactions entre objets en se basant sur des messages ordonnés. Chaque message transitant sur un lien est symbolisé par une flèche porteuse d'une expression. La lecture se fait de haut en bas, et l'ordre chronologique doit respecter ce sens.

La réalisation de diagramme de séquence permet de lister les méthodes dont on aura besoin lors de la phase de développement. Pour ce faire, la description doit être suffisamment générale et exhaustive pour identifier tous les algorithmes.

a. Diagramme de séquence " Get Contenair " :

En premier lieu l'administrateur doit chercher des conteneurs. Puis, il doit demander liste des conteneurs. Finalement, il doit passer liste des conteneurs au chef d'équipe.

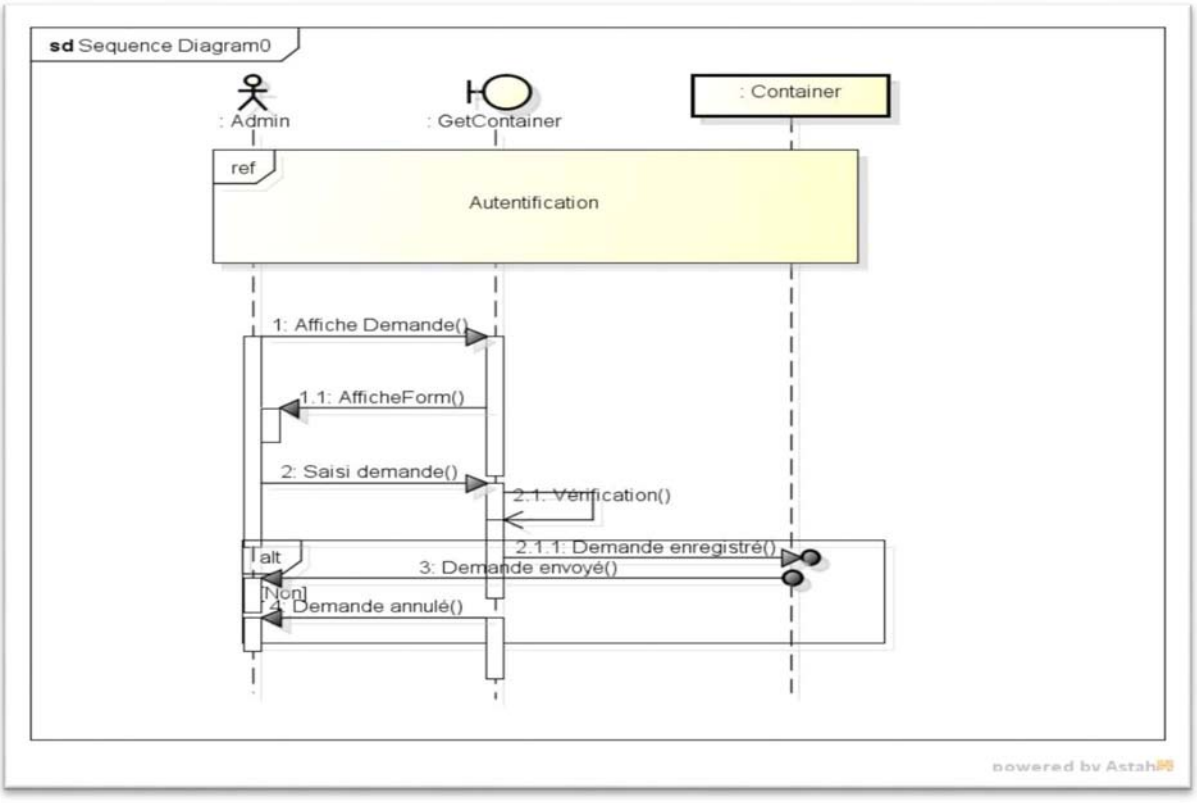

Figure 26 : Diagramme de séquence demande contenair

9 http://laurent-audibert.developpez.com/Cours-UML/html/Cours- UML.html#htoc179(28/03/2013)

b. Diagramme de séquence "s'authentifier" :

Avant d'entrer au contenu du projet et faire l'ensemble des autres scénarios l'utilisateur doit se connecter en utilisant son login et mot de passe.

Le diagramme qui suit montre l'enchainement de la phase d'authentification.

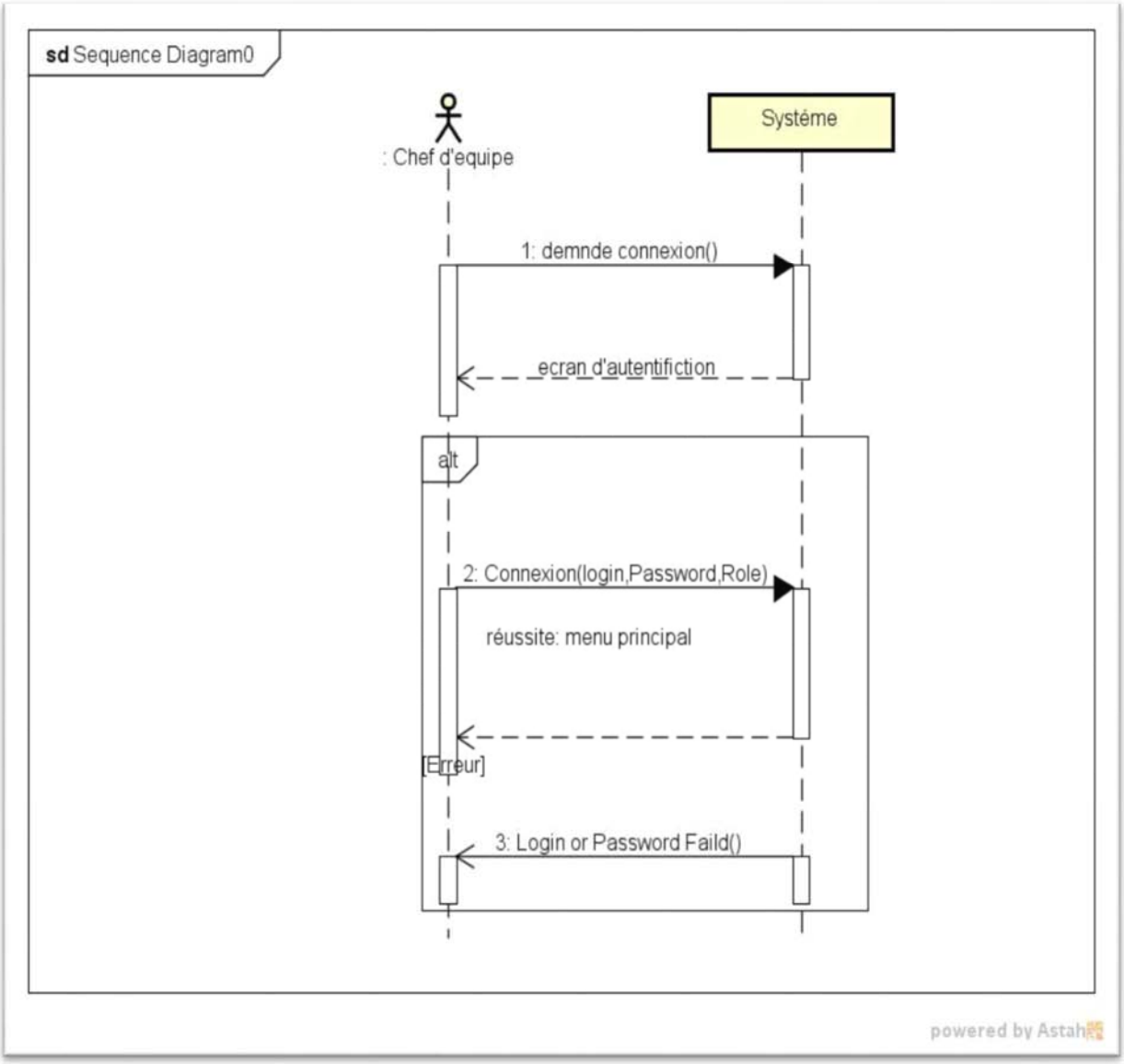

Figure 27 : Diagramme de séquence s'authentifier utilisateur

c. Diagramme de séquence " ajouter contenair"

Le chef d'équipe Crée des conteneurs, en téléchargent des images docker nécessaires de docker hub et en affectant le port et le volume.

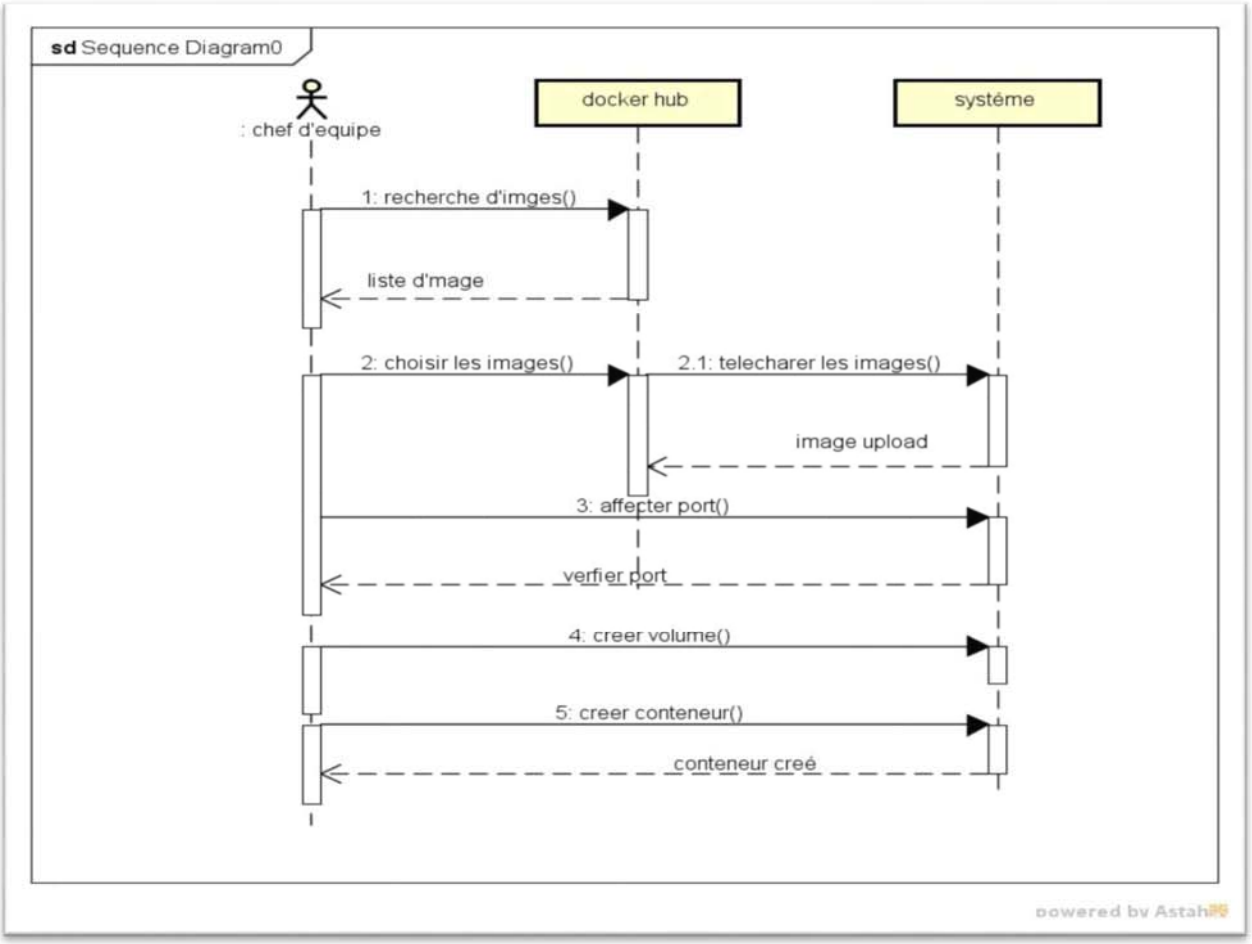

Figure 28 : Diagramme de séquence crée conteneurs

d. Diagramme de séquence "supprimer un contenair "

Le chef d'équipe peut supprimer un conteneur. Ce dernier ne peut pas être supprimer sauf qu'il est en état d'arrêt.

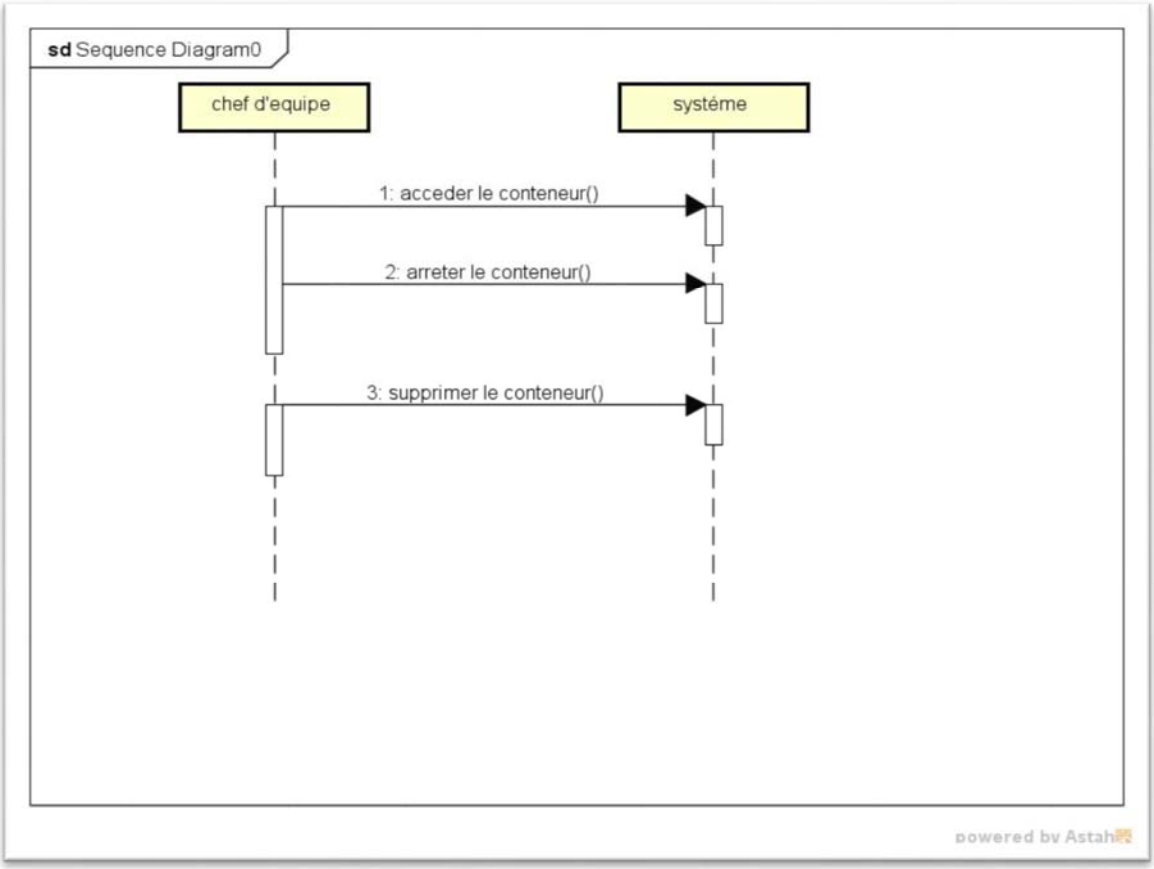

Figure 29 : Diagramme de séquence supprimer conteneurs

II. Architecture MVC[10]

Le patron Modèle-Vue-Contrôleur, tout comme les patrons Modèle-vue-présentation ou Présentation, abstraction, contrôle, est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

Ce paradigme regroupe les fonctions nécessaires en trois catégories :

- un modèle (modèle de données),
- une vue (présentation, interface utilisateur)
- un contrôleur (logique de contrôle, gestion des événements, synchronisation

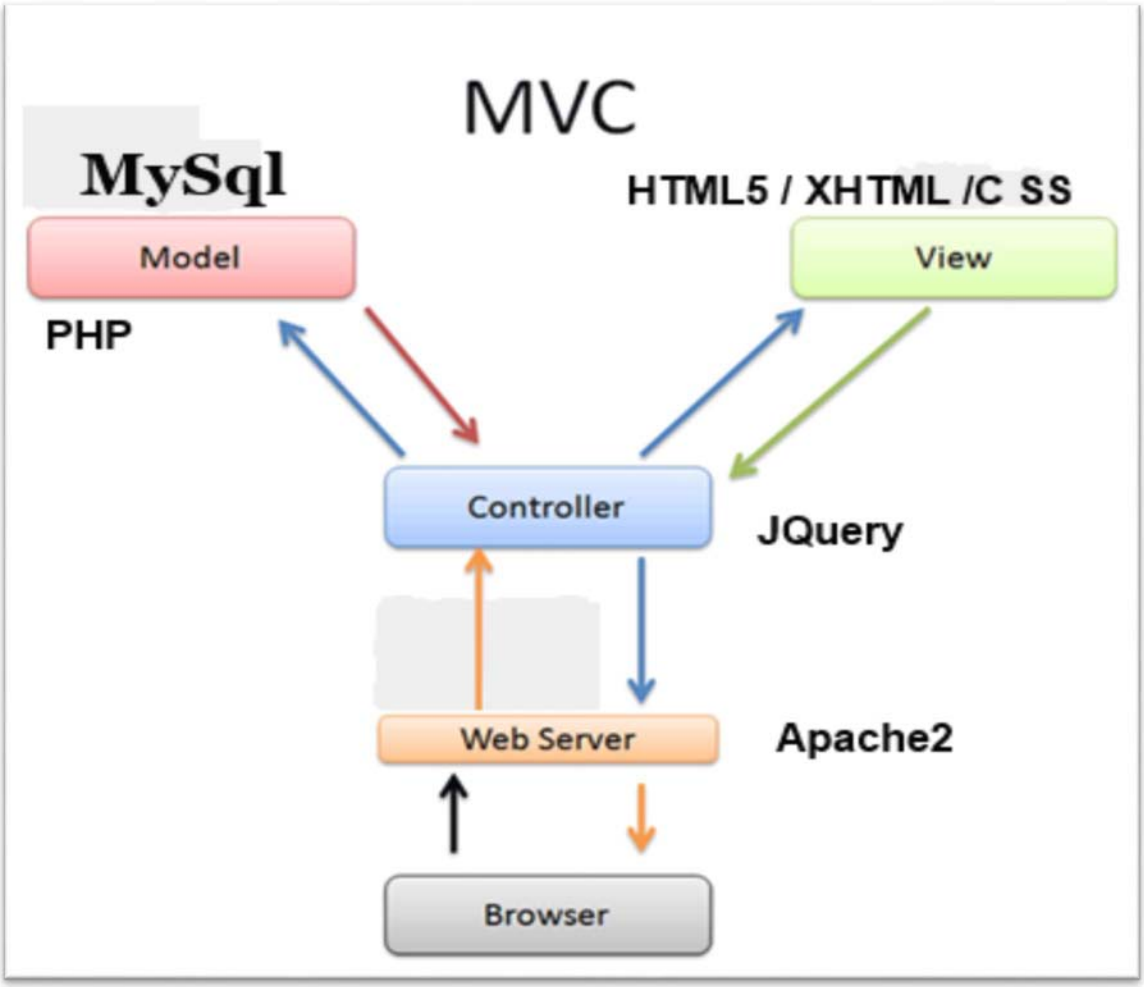

Figure 30 : Modèle MVC

[10] Model-View-Controller

Conclusion

Dans ce chapitre, nous avons présenté le langage UML. Aussi nous avons fait la description du diagramme de classe, diagramme de séquence et diagramme de déploiement afin de déterminer le cadre de notre travail et préparer un terrain favorable pour la prochaine étape.

Maintenant, notre application est prête à être codée. Dans le chapitre suivant, nous allons nous intéresser à l'implémentation de notre système en se basant sur la conception détaillée abordée dans ce chapitre

Chapitre 5 : Réalisation

Introduction :

Dans les chapitres précédents nous avons d'abord présenté les étapes de la conception de l'application, ainsi que les différents diagrammes des cas d'utilisation, puis nous avons abordé en détail cette phase en établissant les diagrammes des classes ainsi que les diagrammes de séquences et .

Dans ce chapitre, nous allons traiter les différentes étapes d'implémentation de l'application, et au fur et à mesure nous allons établir un ensemble de tests : Il s'agit des étapes de la phase de réalisation.

Premièrement, nous allons décrire l'environnement matériel et logiciel. Puis nous allons donner un aperçu sur le travail accompli au cours de la période de développement.

I. Environnement de travail :

Dans cette partie, nous étudions le choix des outils matériels et surtout les outils logiciels du développement web.

1) Environnement matériel :

Ci-dessous les caractéristiques de l'ordinateur sur lequel nous développons l'application parce qu'elles peuvent donner une idée sur les conditions du travail:

- ✓ **Processus :** Intel® Core™ i5 CPU M520 @ 2.40 GHz 2.40 GHz.
- ✓ **Mémoire installé(RAM) :** 6.00 Go (3.86 Go utilisable).
- ✓ **Type de système :** système d'exploitation 64 bits.
- ✓ **Disque dur :** 500 Go.
- ✓ **Ecran :** 15,6 pouces.

2) Environnement logiciels :

Au cours de cette partie, nous énumérons les différents outils utilisés tout au long de ce projet pour l'étude et la mise en place de notre application.

a. Système d'exploitation :

Nous avons utilisé comme système d'exploitation :

- Microsoft Windows 7 Edition Integral.
- Service pack1.

b. Outil de modélisation UML :

Pour la modéliser UML de l'application, nous avons exploité l'outil Astah Community qui est un outil flexible, complet et puissant, conçu pour les plateformes Windows, Linux et MacOS.

Figure 31 : Astah Community

c. Adobe Dreamweaver CS 6:

Adobe Dreamweaver CS6 est un éditeur HTML créé par Adobe pour les web designers pour la partie client.

Figure 32 : Dreamweaver CS 6

d. PhpStorm :

PhpStorm est un environnement de développement (IDE pour integrated development environment) conçu pour PHP, développé par la société JetBrains s.r.o. depuis 2009.

Figure 33 : PhpStorm 2017

e. Technologies utilisées

Nous présentons dans cette partie les différentes technologies exploitées lors de l'implémentation et de l'intégration de l'application

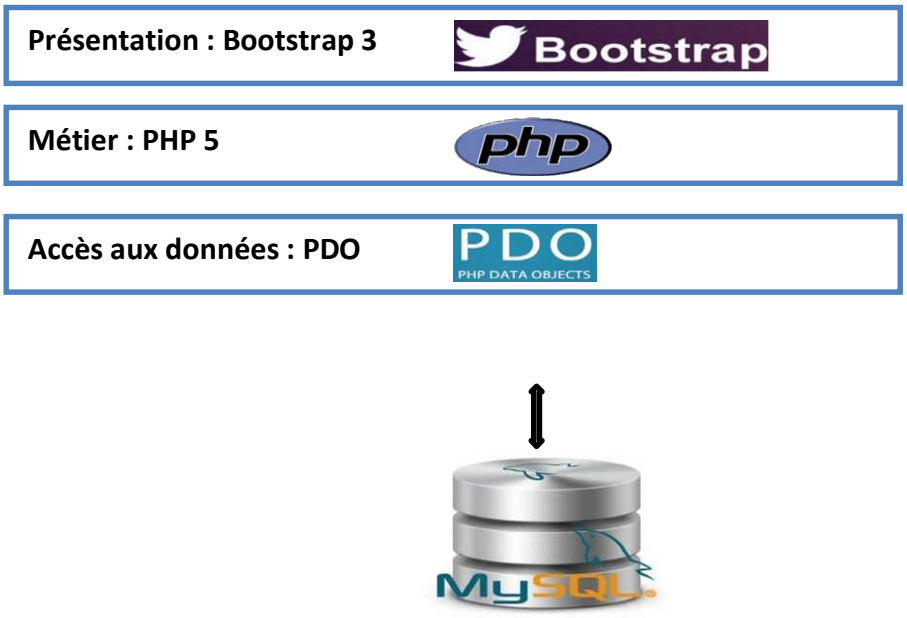

Figure 34 : Technologies utilisées

- **Bootstrap 3 :**Framework open source qui contient des codes HTML5, CSS3 et Javascript, des formulaires, boutons, outils de navigation et autres éléments interactifs.
    - **PHP 5 :** Langage de script les plus actifs sur le Web.
    - **PDO :** Extension PHP qui définit une interface d'accès à une base de données.

II. Captures d'écran :

Cette section sera consacrée aux diverses captures d'écran afin de présenter l'ensemble des fonctionnalités offertes par notre application.

1) Authentification

L'interface présentée dans la figure 35 est l'interface de connexion au système. Elle permet aux utilisateurs de s'authentifier afin de parvenir aux diverses fonctionnalités du système.

Figure 35 : Authentification

2) Interface de Menu:

Cette interface présente le menu de notre application, elle contient plusieurs liens :

- Dashboard : qui présente une vue globle de l'applicaton
- Containers : gérer les conteneurs
- Image : gérer les images
- Networks : gérer Ntework
- volumes : gérer les volumes
- utilisateurs : gérer les utilisateurs

Figure 36 : Interface de Menu de notre application

3) Interface Profil d'utilisateur :

Cette interface affiche le profil d'utilisateur a connecté

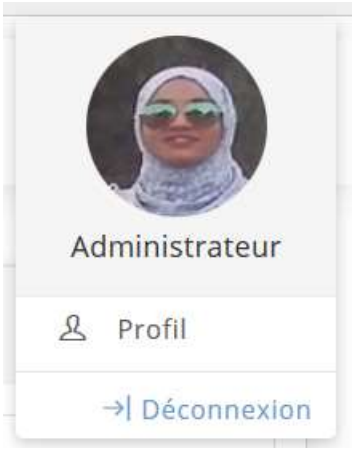

Figure 37 : Interface Profil d'utilisateur

4) Interface Ajouter un utilisateur :

Cette interface présente les champs requis pour ajouter un utilisateur

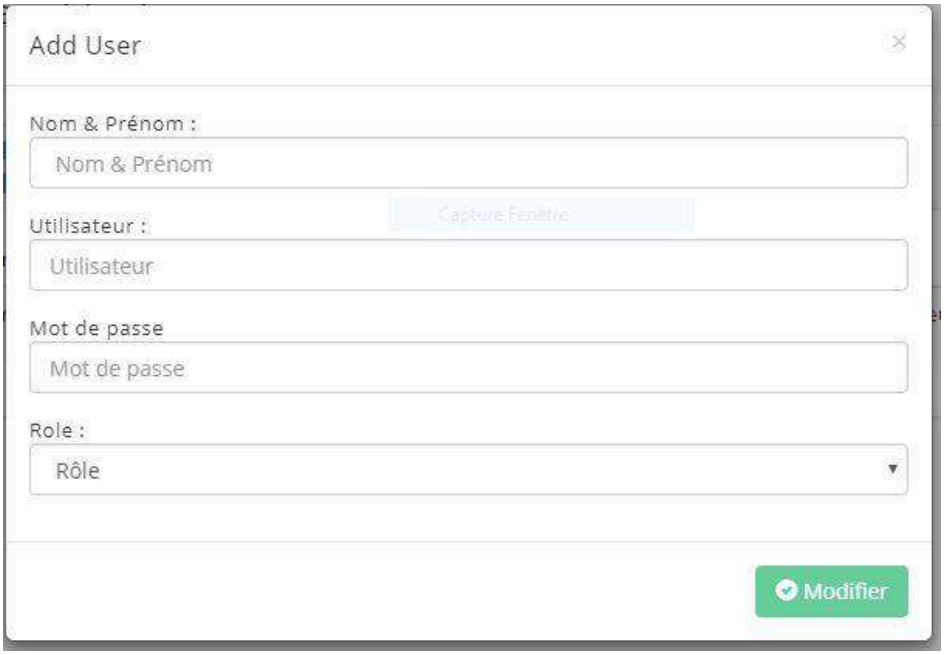

Figure 38 : Interface Ajouter un utilisateur

5) Interface de gestion un utilisateur :

Cette interface affiche liste des utilisateurs ajoutés

Figure 39 : Interface de gestion un utilisateur

6) Bootstrap Modal Plugin :

Le plugin Modal est une boîte de dialogue qui s'affiche en haut de la page en cours

Figure 40 : Le plugin modal

Le plugin Modal Bootstrap lance automatiquement la fenêtre modale lorsque le DOM est entièrement chargé via JavaScript.

7) Bootstrap Notify

Notify.js est un plugin jQuery pour fournir des notifications simples mais entièrement personnalisables.

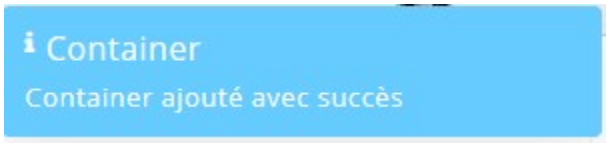

Figure 41 : Bootstrap Notify

8) Validation des données

On ne peut pas laisser l'utilisateur remplir la formullaire n'importe comment. On va devoir valider les informations entrées à l'aide d'un plugin Bootstrap Validator.

Figure 42 : Bootstrap Validator

9) Interface de tableau de bord :

Ce tableau affiche la liste des conteneurs à exécuter, liste des networks, la liste de volume et la liste des images.

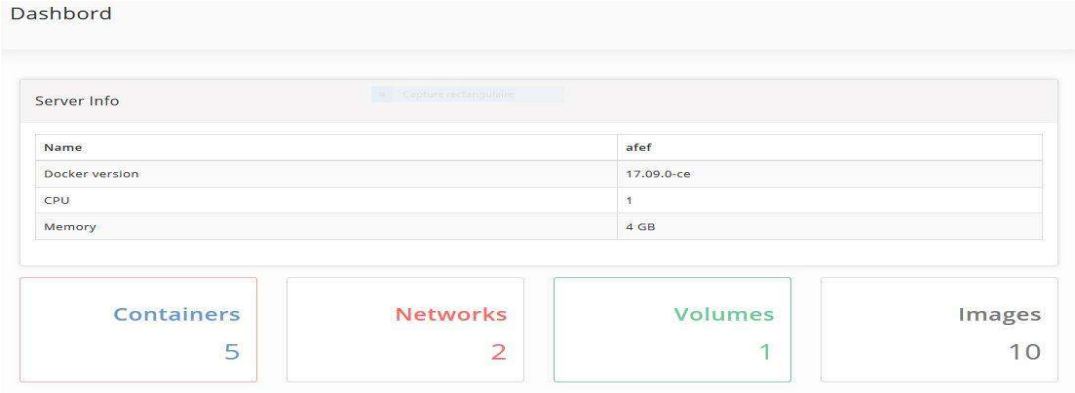

Figure 43 : Interface de tableau de bord

10) Interface pour crée un conteneur :

Cette interface verfie les étapes pour creér un conteneur, télécharger l'image a traves docker hub et affecter le port et le volume

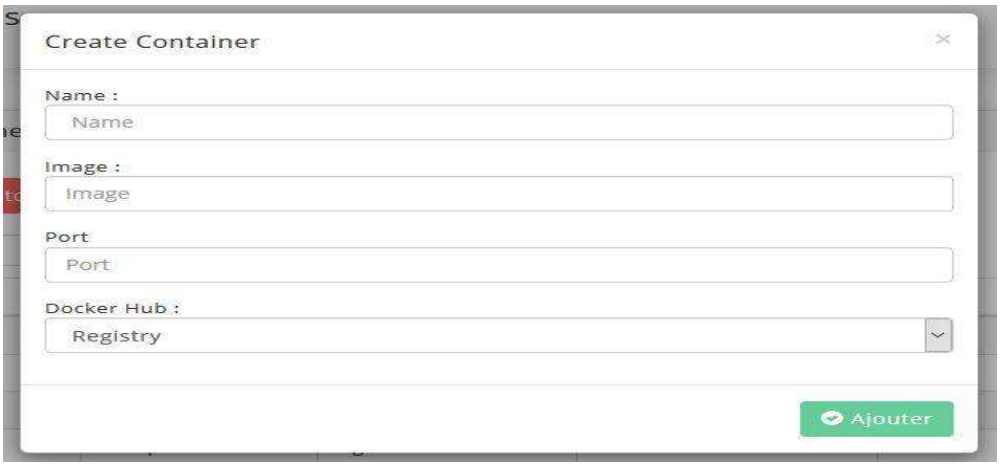

Figure 44 : Interface pour crée un conteneur

11) Interface de liste des conteneurs :

Elle affiche la liste de conteneurs crées.

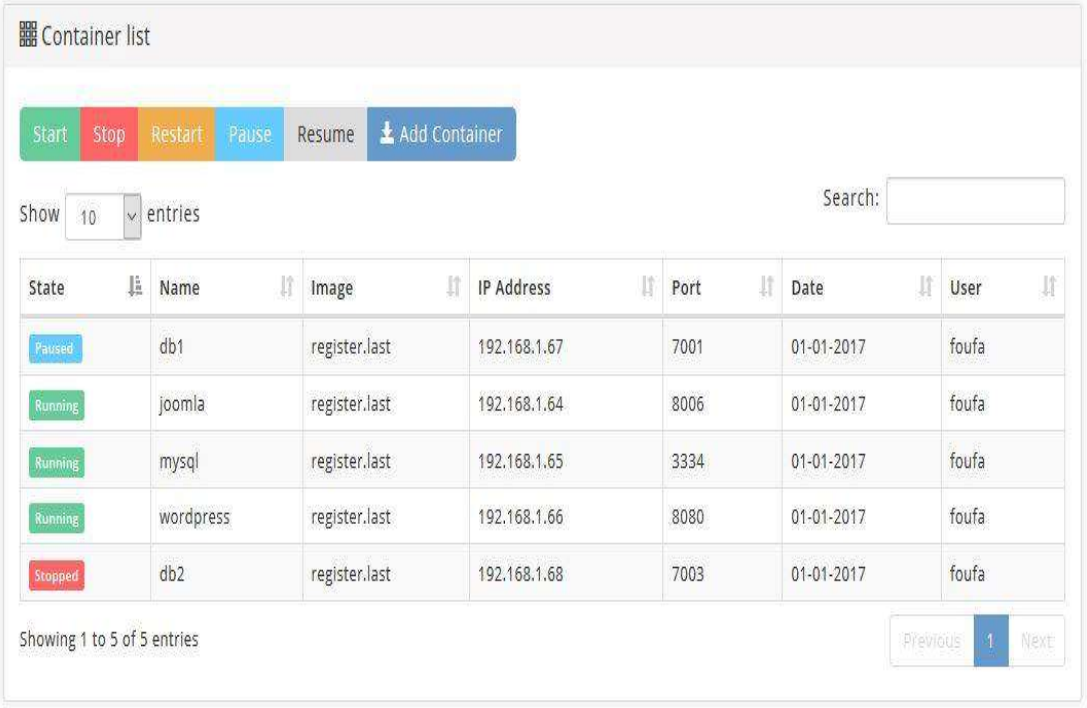

Figure 45 : Interface de liste des conteneurs

Pagination, recherche instantanée et commande multi-colonnes (tri)

12)

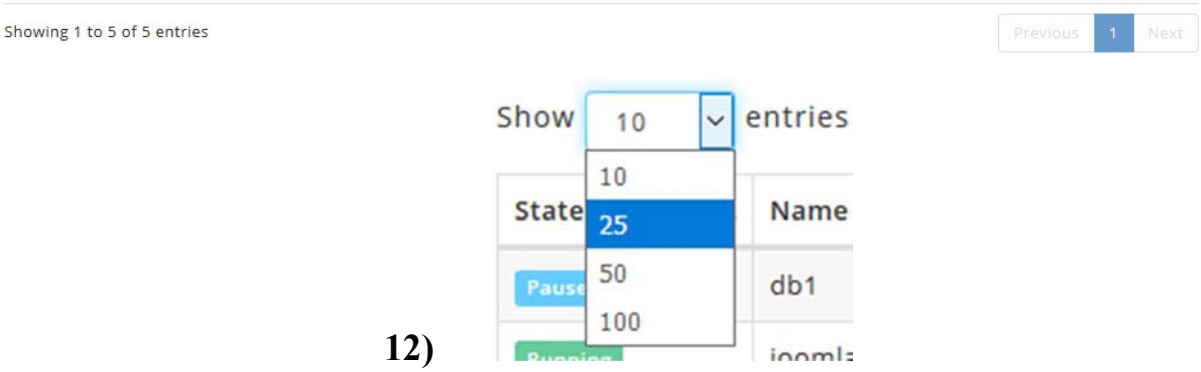

Figure 46 : Data Tables

Data Tables est un plugin pour la bibliothèque jQuery, ajoutant des capacités de tri, de pagination et de filtrage aux tables HTML avec un simple code source.

Conclusion

La dernière partie de ce projet a été dédiée à la navigation dans notre application. Elle constitue le dernier volet de notre rapport. Dans cette parte nous avons présenté l'environnement logiciel et matériel de réalisation. Et on a clôturé par l'exposition des captures écrans décrivant quelques interfaces de notre outil.

Conclusion génerale

Le but de ce projet est de développer une application web pour le management des conteneurs docker. Une telle application permettra l'intervention d'un utilisateur dans un temps optimum et avec un coût minimal.

Tout au long de ce rapport, nous avons présenté les différentes étapes de réalisation de l'application au sein de la société Devercipe . Pour développer cette application nous avons utilisé le langage UML, ce qui nous a permis de mener correctement la tâche d'analyse des besoins à l'aide du diagramme de cas d'utilisation et la tâche de conception, ainsi on a présenté les scénarios en détail afin d'expliquer toutes les tâches faites Puisque nous avons utilisé la technologie PHP qu'on a jugée nécessaire pour accomplir ce projet.

Ce projet représente une opportunité pour s'initier à la vie professionnelle dans un milieu réel et avoir un début d'expérience significative, et il nous a appris comment compter sur soi pour résoudre les problèmes au cas où ils se présentent, comment être méticuleuses dans notre travail, comment être bien organisées pour accomplir dans les meilleurs délais, et meilleures conditions le travail qui nous a été confiés.

Lors de la réalisation de notre projet, nous avons été astreints par quelques limites notammsent, la contrainte du temps qui était relativement un obstacle devant l'ajout de certaines autres fonctionnalités. Certes, il était une occasion pour mettre en évidence et d'explorer sur le plan pratique nos connaissances en informatique.

Cette application peut être améliorée on ajoutons par exemple , un module qui permet l'installation et contrôle du docker sur de serveur distant .