ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE
LA MAÎTRISE EN TECHNOLOGIE DE L'INFORMATION

PAR
JEAN ROOSWELT AMAZAN

SURVEILLANCE DE CONTENEURS DOCKER - Identifier, sélectionner et expérimenter des logiciels libres potentiels qui, combinés, pourraient permettre de faire une surveillance efficace de DOCKER.

MONTRÉAL, LE 08 JANVIER 2016

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Prof. Alain APRIL, directeur de projet, projet industriel de 15 crédits
Département de génie logiciel et des TI à l'École de technologie supérieure

Prof. Roger Champagne, jury
Département de génie logiciel et TI à l'École de technologie supérieure

REMERCIEMENTS

J’adresse premièrement mes remerciements au professeur Alain APRIL, mon directeur de projet, qui a su faire usage de patience envers moi, me permettant ainsi de mener à bien mon projet.

Ensuite, je veux remercier spécialement M. David Lauzon qui m’a apporté un soutien technique ainsi que tous mes collègues ayant partagé avec moi leurs connaissances en vue de me permettre d’atteindre les objectifs de ce projet.

En dernier lieu, je remercie mes amis ainsi que tous les membres de ma famille m’ayant supporté de près ou de loin dans l’accomplissement de mon projet de maîtrise.

SURVEILLANCE DES CONTENEURS DOCKER

Jean Rooswelt AMAZAN

RÉSUMÉ

Ce rapport présente le travail de recherche appliquée effectué en vue d'identifier, de sélectionner et d'expérimenter des logiciels libres potentiels qui, combinés, pourraient permettre de faire une surveillance efficace de DOCKER,

L'ensemble des logiciels pour concevoir le système de surveillance proposé est composé de 4 produits logiciels: 1) « *collectd* » qui sert de collecteur de données; 2) « *elasticsearch* » qui est un moteur de recherche permettant de stocker et d'indexer de grandes quantités de données de surveillance; 3) de « *logstash* » qui est un outil permettant de récupérer les données collectées par « *collectd* » et de les envoyer à l'entrée de « *elasticsearch* », et finalement 4) de « *kibana* » qui est une plateforme de visualisation incluant une interface utilisateur permettant d'afficher les données de surveillance recueillies.

Cette solution offre un prototype de tableau de bord permettant de visualiser les métriques et évènements collectés sur les conteneurs de Docker en opération ainsi que ceux du système hôte en quasi temps réel. Un système d'alerte y est intégré afin de notifier les gestionnaires du système par courriel en cas de violation de certains seuils définis au préalable.

Pour parvenir à ce résultat, il a fallu effectuer les étapes suivantes : faire un état de l'art pour comprendre Docker et la problématique posée par la surveillance des conteneurs, étudier et tester différents outils afin de trouver les meilleurs pouvant aider à solutionner cette problématique.

Mots-Clés : Docker, Virtualisation, Surveillance, Métrique, Tableau de bord

X

SURVEILLANCE DES CONTENEURS DOCKER

Jean Rooswelt AMAZAN

ABSTRACT

This report presents the applied research work that has been done to find, select and experiment a number of existing open source software that, together, could allow monitoring Docker containers. Docker is an open platform based on container virtualization technology. Such containers are deployed operationally on an IT infrastructure.

The proposed list of open source software to provide system monitoring of DOCKER based solutions is composed of the following set of four products:

- Collectd as a collector,
- Elasticsearch which is a search and analytic engine that is used to index log data,
- Logstash to process unstructured data collected with Collectd and stored in Elasticsearch, and
- Kibana which is an open source visualization platform that allows graphical interaction with the collected data.

This solution presents a surveillance dashboard prototype that allows fast and efficient analysis on the collected data from operational Docker Containers as well as the system host. Also an alert system has been integrated to notify the system admin in real time via email based on violation of some predefined thresholds.

To reach the goal, the following steps were necessary:

- State of the art to understand Docker and the issues caused by containers monitoring,
- Study and test different tools in the way to find the best ones that could be used to resolve the identified issues.

Keywords: Docker, Virtualization, Monitoring, Metric, Dashboard

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

API	: Application Programing Interface
CGROUP	: Control Group File System
CFS	: Completely Fair Scheduler
CLI	: Command LIne
CPU	: Control Process Unit
CSV	: Comma Separated Value
DRS	: Distributed Resource Scheduler
E/S	: Entrées / Sorties
FT	: Fault Tolerance
GUI	: Graphic User Interface
HA	: High Availability
IaaS	: Infrastructure as a Service
I/O	: Input/Output
MV	: Machine Virtuelle
OS	: Operating System
PaaS	: Platform as a Service
RAM	: Random Access Memory
RRD	: Round Robin Database
SaaS	: Software as a Service
SE	: Système d'exploitation
SI	: Système d'information
TI	: Technologie de l'information
VM	: Virtual Machine (Machine Virtuelle)

INTRODUCTION

La gestion des infrastructures est l'un des domaines les plus importants des Technologies de l'Information (TI). Les avancements, dans ce domaine, apparaissent à une vitesse de plus en plus rapide. Avec les progrès effectués par la recherche dans le domaine des TI's, les outils de surveillance des infrastructures sont rapidement devenus indispensables pour les entreprises. Avec l'avènement des nouveaux logiciels et technologies opérant en temps réel, sur Internet, il est incontournable que les fournisseurs de services TI fassent une utilisation optimale des ressources déployées sur leurs serveurs (c.-à-d. ordinateurs puissants) et pour tout autre matériel informatique sur lequel opèrent les logiciels utilisés par la clientèle. Pour parvenir à cette fin, autant sur des serveurs physiques que sur des serveurs virtualisés, il importe de surveiller les ressources et l'usage fait par la clientèle afin de s'assurer d'offrir un niveau de service optimal et de faire une gestion efficace des infrastructures TI.

La surveillance des machines virtuelles (VM) est facilitée grâce au fait qu'elles sont une émulation d'équipements physiques sur lesquels sont installées toutes les applications de surveillance nécessaires, d'où les journaux (c.-à-d. les fichiers «logs» qui contiennent les évènements survenus) sont générés et placés, parfois à l'intérieur de la VM, généralement dans le répertoire «/var/log/». Les administrateurs de système utilisent des commandes telles que: «*dmesg*», «*cat», «grep», «tail» et «awk»* afin d'analyser ces données ou encore: «*tload» et «top»* qui permettent de consulter certaines mesures de performance. Par contre, pour la technologie émergente DOCKER, trouver une solution de surveillance n'est pas facile actuellement, et ce pour les raisons suivantes [45]:

- Les journaux sont localisés dans des endroits isolés,
- Les fichiers peuvent se situer n'importe où,
- Généralement «syslog» qui contient les journaux (de l'anglais «logs») du système ne s'exécute pas à l'intérieur des conteneurs,
- La nécessité de rotation des journaux,
- Les journaux envoyés par les conteneurs aux «STDERR» et «STDOUT» sont accessibles à partir des commandes «logs» de DOCKER.

Quelques solutions existent visant à surveiller les conteneurs DOCKER, mais elles sont soit coûteuses soit adaptables à un système particulier. Le défi actuel est de trouver des logiciels libres qui, ensemble, pourront collecter toutes les informations concernant les évènements produits par les conteneurs de DOCKER ainsi que de collecter les métriques en vue de les centraliser à des fins d'analyses ultérieures sachant que les conteneurs peuvent être localisés sur un seul serveur ou distribués à travers une grappe de serveurs.

Ce projet appliqué, de 15 crédits de recherche, a pour objectif d'identifier, sélectionner et expérimenter des logiciels libres potentiels qui, combinés, pourraient permettre de faire une surveillance efficace de DOCKER. Cette solution devra pouvoir colliger des métriques matérielles et logicielles afin d'en déduire des indicateurs de performance aux fins de certaines analyses. Ces analyses auront pour objectifs, par exemple, d'apporter des solutions réactives ou proactives sur les serveurs utilisant cette technologie. Ce rapport est composé de trois chapitres. Le premier chapitre présente la revue littéraire ainsi que la problématique du projet appliqué. Le deuxième chapitre présente une proposition de solution ainsi que l'expérimentation effectuée. Finalement, le troisième et dernier chapitre discute des résultats obtenus, des limites de cette courte recherche appliquée ainsi que les perspectives d'avenir dans ce domaine.

CHAPITRE 1

REVUE LITTÉRAIRE ET PROBLÉMATIQUE

Ce chapitre présente la revue littéraire concernant les technologies actuelles de surveillance des infrastructures TI et plus précisément de virtualisation et de conteneurs dans le contexte de l'infonuagique. On y présente un bref historique ainsi qu'une comparaison sommaire entre la technologie de virtualisation et celle des conteneurs. Par la suite, une synthèse de ce qu'est la surveillance d'infrastructure TI, c'est-à-dire ce à quoi l'on s'attend typiquement d'un bon logiciel de surveillance d'infrastructure TI ainsi que quelques solutions actuellement disponibles sont présentées. En dernier lieu, cet état de l'art discute de la problématique qui a lancé ce projet de recherche, le contexte de cette recherche suivi d'un résumé de synthèse.

1.1 La technologie Docker

1.1.1 Historique

DOCKER est une plateforme de virtualisation par conteneurs qui a vu le jour en mars 2013, lancée par le Français Solomon Hykes. Le slogan du site officiel, traduit de l'anglais, donne une idée bien concise de ce qu'est DOCKER et de son utilisation potentielle : « *Construire, déployer et exécuter n'importe quelle application, n'importe où* ! » [1]

Initialement dotCloud, ancien nom de la compagnie Docker Inc., visait à fournir une «Platform as a Service» (PaaS) moderne qui devait affranchir les concepteurs de logiciels de la contrainte de déploiement des applications. Les ingénieurs de dotCloud visaient ainsi à offrir des applications isolées, pouvant être déployées rapidement, tout en s'assurant que leur plateforme pouvait non seulement faire tourner plusieurs types de langages, mais aussi s'exécuter sur différents types d'ordinateurs et de systèmes d'exploitation. Afin d'atteindre leurs objectifs, ils ont pensé à emballer chaque application ainsi que toutes leurs dépendances dans un seul et même conteneur logiciel, et c'est ce dernier qu'il a fallu rendre compatible

avec différents types d'ordinateurs. C'est cette solution qui a été reproduite et publiée en logiciel libre et qui est maintenant connue sous le nom de DOCKER. (Beshawred, 2014)

Quoique la technologie DOCKER ait été pensée à l'origine pour les distributions Linux, elle n'est pas limitée qu'à ces dernières. Elle est également utilisée dans les plateformes Cloud et sur d'autres systèmes d'exploitation. Ainsi, en 2014, DOCKER a pu rallier les acteurs du nuage et du logiciel; et même les géants des plateformes propriétaires tels que VMWare, Microsoft l'ont adopté (Clapaud, 2015).

1.1.2 Qu'est-ce que Docker ?

Certains auteurs qualifient DOCKER de technologie de virtualisation par conteneurs (Charles Anderson, 2015), par contre le site officiel de DOCKER le définit comme une plateforme libre permettant de développer, transporter et d'exécuter des applications qui fournit, à sa base, un moyen d'exécuter presque toutes les applications de manière sécurisée et isolée dans un conteneur. [33]

1.1.3 Qu'est-ce qu'un Conteneur ?

Un conteneur est une unité logique, dans laquelle on empaquète une application avec toutes les dépendances dont elle a besoin. Elle est donc capable de s'exécuter de manière isolée par le noyau du système hôte. Le mot «*conteneur*» n'est pas choisi au hasard, il vise à recréer l'image, en tout point semblable aux conteneurs en métal (Guillaume, 2014) [43], qui sont fabriqués suivant une norme précise leur permettant d'être pris en charge, quel que soit leur contenu, par n'importe quel transporteur. À l'image de ces conteneurs physiques, une «application logicielle *dockérisée*»[1] doit pouvoir fonctionner et être transportée sur n'importe quel hôte, quel que soit son contenu (Guillaume, 2014).

---

[1] Une application *dockérisée* est une application qui a été empaquetée avec ses composantes dans un conteneur DOCKER.

1.1.4 Les principales composantes de Docker

DOCKER est composé de deux composantes principales qui sont les suivantes [33]:

- DOCKER qui est la plateforme libre de virtualisation par conteneurs,
- DOCKER «hub» qui est la plateforme de logiciel comme service (c.-à-d. Software as a Service) permettant de partager et de gérer les conteneurs de DOCKER.

1.1.5 Architecture de Docker

DOCKER utilise une architecture client-serveur dans laquelle le client de DOCKER communique avec le «démon» de DOCKER responsable de la construction, du déploiement et de l'exécution des conteneurs à travers des interfaces de connexion (c.-à-d. «sockets» en anglais). (Docker, 2015)

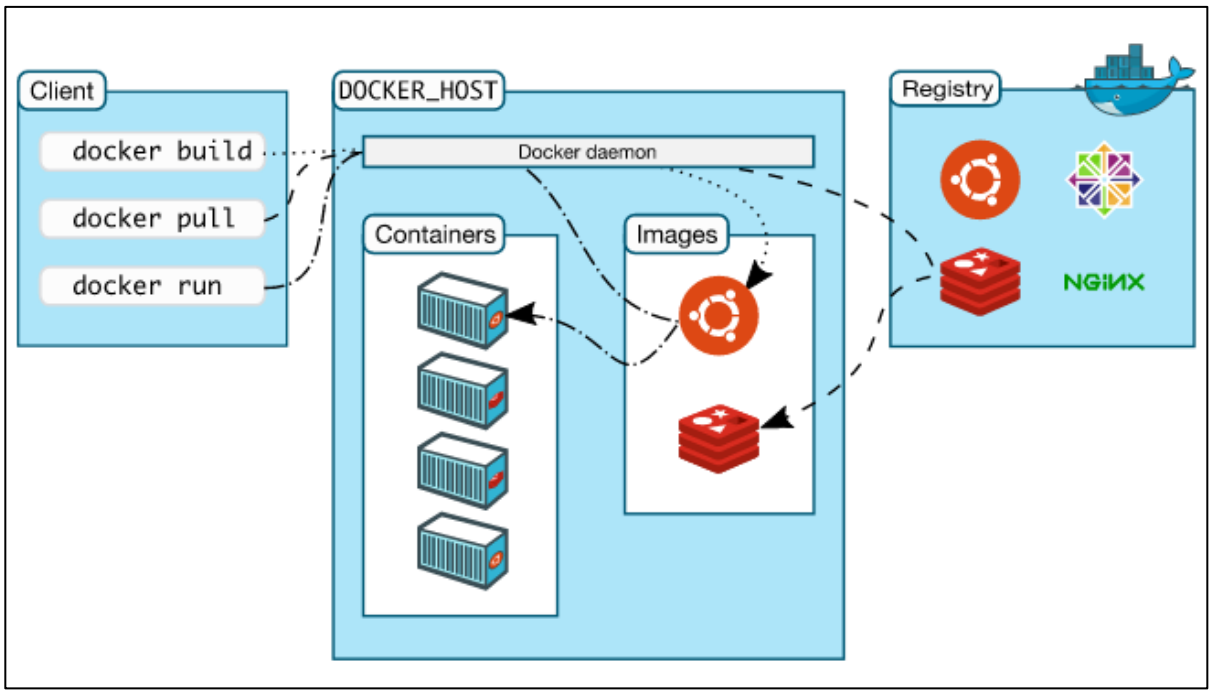

Figure 1.1 Communication entre le client et le démon de Docker [33]

1.1.6 Composantes internes de Docker [33]

À l'interne, DOCKER est composé des trois éléments suivants :

Les images : Elles sont les composantes de construction de DOCKER et représentent des modèles (c.-à-d. templates en anglais) accessibles en lecture seulement,

Les registres : Ils sont les composantes de distribution de DOCKER qui hébergent les images,

Les conteneurs : Ils sont les composantes d'exécution de DOCKER et sont similaires à des répertoires.

La figure 1.2 présente les composantes internes de DOCKER et les interactions entre elles.

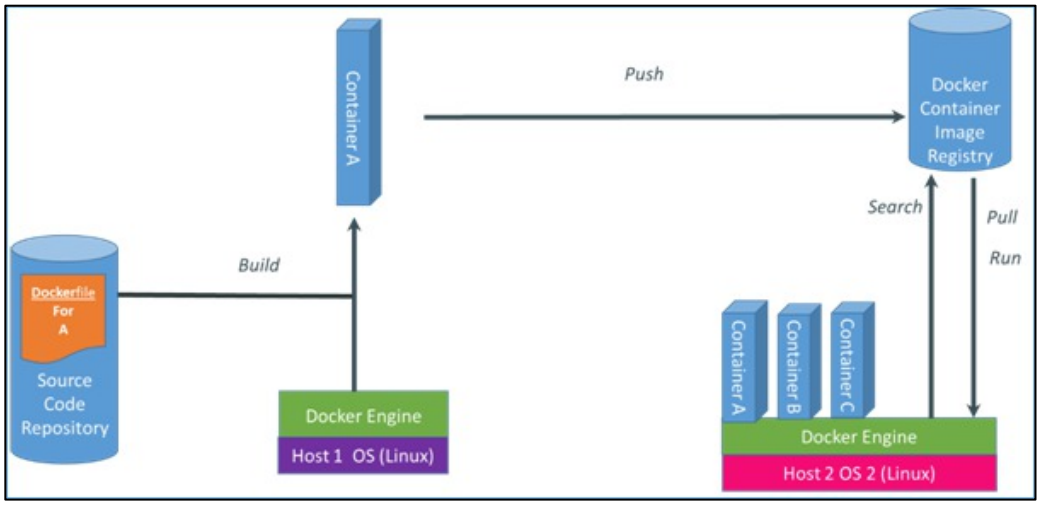

Figure 1.2 Composantes internes de Docker [28]

1.1.7 Comparaison des conteneurs de Docker aux machines virtuelles

La virtualisation par hyperviseurs (c.-à-d. par machine virtuelle) diffère de la visualisation par conteneurs de DOCKER par différents points. Premièrement, au niveau de l'architecture : La couche Hyperviseur est éliminée dans la virtualisation par conteneur. Les applications utilisent le système d'exploitation de l'hôte et ses pilotes, ce qui rend les conteneurs DOCKER plus rapide que les machines virtuelles [7]. La figure 1.3 présente la différence existant entre les différentes couches logiques de ces deux technologies.

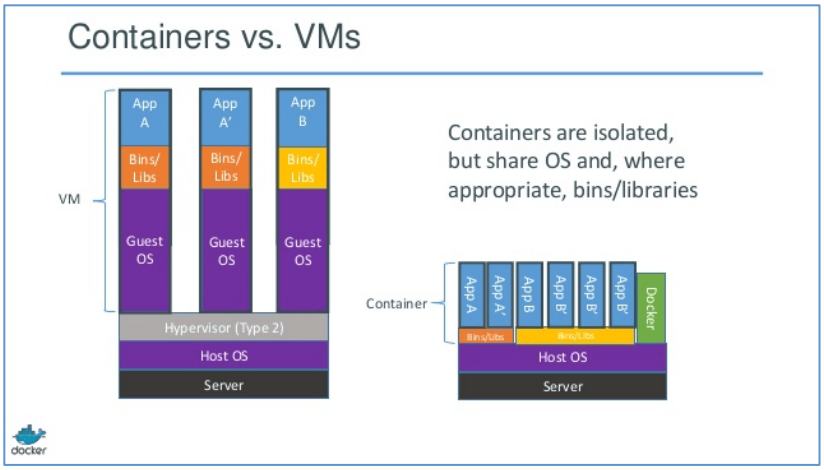

Figure 1.3 Comparaison Conteneurs et VM [1]

Deuxièmement, au niveau de performance et de la vitesse: la virtualisation par hyperviseur permet que les machines virtuelles, étant une abstraction des machines physiques, nécessitent un système d'exploitation, des démons (de l'anglais «daemons»), de la gestion de mémoire et des pilotes de matériels, ce qui les rend plus lourds et occupent plus d'espace mémoire par rapport aux conteneurs qui utilisent le système d'exploitation (SE) de l'hôte.

Tableau 1-1 Comparaison entre VM et Conteneurs Docker suivant des paramètres [5]

Paramètres	**Machines Virtuelles**	**Conteneurs**
Système d'exploitation	Chaque machine virtuelle fonctionne sur un matériel virtuel et le noyau est chargé dans sa propre région de mémoire	Tous les conteneurs partagent le même SE et le même noyau. L'image du noyau est chargée dans la mémoire physique.
Communication	Se fait à travers les périphériques Ethernet	Mécanisme d'IPC standards tels que: les signaux, les tuyaux et les soquets (c.-à-d. sockets en anglais)
Sécurité	Dépend de l'implémentation de l'hyperviseur	Le contrôle d'accès obligatoire peut être exploité
Performance	Les instructions-machine étant transmises à partir du SE de la MV, ce qui occasionne une certaine surcharge	Les conteneurs fournissent une performance quasi native par rapport au SE d'accueil sous-jacent

Tableau 1-2 Comparaison entre VM et Conteneurs Docker suivant des paramètres [5] suite

Isolation	Impossible de partager les librairies et les fichiers entre l'hôte et la VM	Les sous-répertoires peuvent être montés de manière transparente et être partagés
Temps de démarrage	Les VM démarrent en quelques minutes	Les conteneurs démarrent en quelques secondes
Stockage	Les VM consomment plus d'espace de stockage, car on doit installer sur elles le SE et les programmes à exécuter	Les conteneurs consomment moins d'espace de stockage, car ils partagent avec l'hôte le SE.

1.1.8 Caractéristiques d'un bon outil de surveillance de la performance

Recommander l'utilisation de bons outils de surveillance est bien, mais faut-il encore savoir ce qu'est un bon outil de surveillance. Un bon outil de surveillance devrait pouvoir effectuer les tâches suivantes :

1. Collecter les informations essentielles : métriques de CPU, RAM, Disk I/O, flux de données et état des services pouvant être utilisées afin de diagnostiquer des incidents en vue d'identifier les éventuelles causes et permettre de les résoudre ou de les prévenir,
2. Enregistrer ces informations dans une base de données ou dans des journaux afin d'en avoir un historique de ces données pour des analyses et traitements ultérieurs,
3. Afficher en quasi temps réel les informations critiques dans un tableau de bord dynamique via une interface conviviale pouvant être personnalisée au besoin de l'utilisateur final,
4. Alerter les personnes concernées en cas d'erreur ou si une métrique a dépassé ou est en dessous d'un certain seuil fixé,
5. Exécuter certains tâches et scripts automatiquement sous certaines conditions.

1.1.9 Les outils de surveillance existants

De nombreux outils de surveillance existent. Parmi les plus connus et utilisés pour surveiller les technologies de virtualisation, nous pouvons lister les suivants:

- Nagios [18] : Outil de surveillance de système réseau, des hôtes et services désirés avec possibilité d'alerter les gestionnaires en cas de violation de police,
- Hyperic-HQ [8] : Logiciel libre permettant de surveiller des applications web et gérer les performances en environnement virtuel, nuagique et physique. Il a une interface graphique assez conviviale,
- New Relic [15] : Système de surveillance très puissant permettant de surveiller plusieurs systèmes nuagiques à différents niveaux de la pile logicielle,
- Container Advisor [10] : Outil de Google qui fournit des caractéristiques de performance et d'utilisation des conteneurs en cours d'exécution.

1.2 Problématique de recherche

DOCKER est utilisé généralement dans un contexte infonuagique avec l'objectif d'offrir des services (SaaS et PaaS) en empaquetant : base de données, serveur mail, application web et bien d'autres applications. Une application web offerte en service peut être structurée de manière à ce que ses composantes soient installées sur différents conteneurs communiquant entre eux. Dans l'éventualité qu'il y ait un arrêt du service (ou des problèmes de performance) avec les outils de surveillance système existants (c.-à-d. «*cat», «tail», «grep» ou même DOCKER «logs»*) il peut être difficile de détecter le conteneur qui en est la cause. Surveiller l'état des conteneurs en temps réel, être alerté rapidement lors d'une panne, réduire le temps passé à identifier les causes des incidents, automatiser la résolution des problèmes mineurs et réduire le temps d'arrêt des services représentent les principaux défis pour un gestionnaire qui utilise des conteneurs DOCKER.

1.3 Mise en contexte

La technologie émergente DOCKER intéresse de plus en plus de gestionnaires d'infrastructure (Filippone, 2015). Que ce soit dans de grandes organisations offrant des services d'infonuagique ou dans de moyennes ou petites entreprises offrant des «PaaS» ou désirant déployer rapidement des solutions d'un environnement test vers la production, DOCKER peut être une solution intéressante. Tout système, y compris ceux basés sur la technologie DOCKER, peut entraîner des problèmes de performance. Avoir une solution de surveillance est un incontournable pour tout gestionnaire de système en vue de pouvoir mener à bien ses diagnostics. Au moment de la rédaction de ce rapport, il existe très peu de solutions de surveillance pour la nouvelle technologie DOCKER.

1.4 Les approches existantes pour la surveillance de Docker

Afin de surveiller les conteneurs de Docker plusieurs approches sont proposées parmi lesquelles deux seront brièvement présentées :

1.4.1 L'approche surveillance de l'ensemble des services formant l'application

Dans cette approche, l'accent est mis sur l'application ou le service (aussi nommé le cargo) plutôt que sur les conteneurs eux-mêmes. Ceci garantit d'avoir une vue globale sur la façon dont les services sont exécutés ainsi que leur état. En conséquence, les outils utilisant cette approche font une surveillance du logiciel plutôt que de son infrastructure. Ainsi des logiciels de surveillance tels que ruxit [13], sysdig [14] et New Relic [15] utilisent cette approche popularisée sous le nom de «l'approche de la boîte noire».

1.4.2 L'approche de surveillance des conteneurs de Docker

Une autre approche consiste à surveiller les conteneurs DOCKER en allant chercher les différentes mesures des ressources physiques telles que : le «CPU», la mémoire, les entrées/sorties (É/S) de disques et du réseau localisées dans les « cgroups ». C'est à l'aide de

ces données que des analyses sont effectuées en vue d'une prise de décision. Parmi les outils utilisant cette approche, nous pouvons citer Datadog [16] et Axibase [17].

1.5 Résumé

Ce chapitre présente DOCKER comme une solution de virtualisation disponible en logiciel libre qui utilise des conteneurs dans lesquels sont empaquetées des applications ainsi que leurs dépendances en vue de les rendre portables et aptes à être exécutées sur n'importe quelle plateforme. La technologie DOCKER diffère de la technologie de virtualisation par hyperviseur du fait que ses conteneurs ne contiennent que les applications/librairies et utilisent le noyau et le système d'exploitation de l'hôte. Cette conception rend DOCKER plus rapide et plus léger par rapport à la technologie par machines virtuelles qui, contrairement, émulent des machines physiques en hébergeant chacune leur propre système d'exploitation. Les journaux générés par les conteneurs DOCKER peuvent se trouver n'importe où sur le disque et même à l'intérieur des conteneurs. En cas de problèmes de performance ou d'arrêt de service, il peut être très fastidieux et compliqué de diagnostiquer des problèmes de performance ou d'arrêt de services. Conséquemment, une solution de surveillance des conteneurs DOCKER est nécessaire afin de collecter les mesures et les journaux essentiels au diagnostic d'éventuels problèmes. Le chapitre qui suit décrit en détail une proposition de solution de surveillance des conteneurs DOCKER en identifiant et sélectionnant des logiciels libres potentiels qui, combinés, pourraient permettre de faire une surveillance efficace de DOCKER.

CHAPITRE 2

PRÉSENTATION D'UNE SOLUTION DE SURVEILLANCE DOCKER

Ce chapitre décrit les recherches effectuées menant aux choix de logiciels libres potentiels qui, combinés, pourraient permettre de faire une surveillance efficace de DOCKER. Il y est présenté le détail du fonctionnement de la proposition, des processus d'implantation, des mesures à collecter, des différents logiciels choisis et de l'environnement dans lequel les expérimentations d'un cas d'étude seront effectuées. En dernier lieu, une synthèse des raisons pouvant justifier que cette solution sera efficace afin d'adresser cette problématique sera abordée.

2.1 Contexte

La préoccupation de tout gestionnaire de plateforme et d'infrastructure logicielle est d'avoir une visibilité sur l'état et la santé de ces dernières, préférablement en temps réel, afin d'agir rapidement en cas de besoin pour assurer une disponibilité du service en tout temps avec les meilleures performances possible. Sans de bons outils de surveillance permettant de rendre disponibles les informations nécessaires aux diagnostics de certains évènements, la gestion d'infrastructure peut devenir un cauchemar pour celui qui en a la charge. DOCKER qui est une plateforme de virtualisation qui connaît une popularité croissante, pour les diverses raisons citées au chapitre précédent (cf. section 1.3) n'est pas actuellement facile à surveiller. En vue d'apporter une solution à la surveillance des conteneurs de DOCKER, la méthodologie décrite dans les lignes qui suivent a été adoptée.

2.2 Méthodologie utilisée

2.2.1 Méthodologie de recherche

Pour mener à bien ce projet, la méthodologie suivante a été appliquée :

- Élaborer une revue de littérature de la technologie DOCKER. Cette étape visait à comprendre ce qu'est DOCKER, son historique, son évolution et la différence existant entre ses conteneurs et les machines virtuelles,
- Comprendre la problématique posée au niveau de la surveillance des conteneurs DOCKER et ce qui rend cette tâche plus difficile à effectuer sur les conteurs DOCKER par rapport aux machines virtuelles,
- Préciser les objectifs de l'expérimentation du projet,
- Faire un choix et se familiariser avec une distribution de Linux en vue de comprendre le fonctionnement de ce système d'exploitation (dans le cadre de ce projet, le choix s'est porté sur Ubuntu Trusty, précisément la version 14.04),
- Installer DOCKER sur Ubuntu afin de maîtriser cette nouvelle technologie, maîtriser le fonctionnent des conteneurs. Expérimenter DOCKER à l'aide de quelques images en vue de mieux comprendre comment aborder la problématique de surveillance,
- Chercher dans les blogues, les webinaires et les forums les besoins, les problématiques ainsi que les solutions de surveillance proposées, si elles existent, par des utilisateurs de DOCKER en ce qui a trait à sa surveillance,
- Comprendre et définir, pour ce projet, ce qu'est la surveillance et ce que devrait faire un bon outil de surveillance,
- Identifier, sélectionner et expérimenter des logiciels libres potentiels qui, combinés, pourraient permettre de faire une surveillance efficace de DOCKER,
- Appliquer l'ensemble des logiciels libres proposé à une étude de cas.

2.2.2 Cycle de vie d'un processus de surveillance d'infrastructure TI

Avant de commencer à réfléchir sur une solution permettant de surveiller les conteneurs de DOCKER, il est primordial de comprendre les différentes étapes du cycle de vie d'un processus de surveillance. En se basant sur les caractéristiques d'un bon outil de surveillance

définies au chapitre premier de ce document (cf. section 1.2.1), il a été identifié qu'un processus de surveillance a un cycle de vie composé de cinq étapes présentées comme suivent :

Étape 1 « Collecte des données » : À cette étape, toutes les données du (des) système(s) surveillé(s) sont collectées. Cette étape est primordiale, car elle est la base de la surveillance, parce que sans données il est impossible de faire des analyses. Un point important à ce niveau, c'est de bien spécifier les sources où les données seront collectées sinon les données collectées ne seront pas utilisables pour atteindre les objectifs.

Étape 2 « Modélisation et Tri des données » : Les données recueillies à l' «étape 1» sont disponibles, mais peuvent être de toute sorte et de formats différents suivant les sources d'où elles proviennent, ce qui les rend difficiles à analyser et à manipuler. Pour corriger le problème, il est important de faire le tri des données et de les modéliser en les mettant dans un format unique pouvant permettre une automatisation de l'analyse des informations y résultant.

Étape 3 « Analyse des données » : À cette étape, les informations recueillies sont analysées en vue de déterminer s'il y a certains mesures et évènements produits qui méritent une certaine attention.

Étape 4 « Notification » : Cette étape n'est pas obligatoire, mais est fortement recommandée, car elle peut aider à sauver du temps dans l'identification d'incidents mineurs et de leurs éventuelles causes. L'idéal c'est de définir certains seuils et d'avertir le gestionnaire par SMS, courriel ou message infobulle en cas de violation de ces seuils ou d'erreur ou d'avertissement.

Étape 5 « Prise de décision » : Le gestionnaire étant imbu de l'état de santé de son système, il est en mesure de prendre des décisions proactives ou réactives en vue de garder son système opérationnel pour la satisfaction de ses clients.

2.2.3 Représentation du cycle de vie d'un système de surveillance

Les étapes du cycle de vie d'un système de surveillance d'infrastructure TI décrites à la section précédente peuvent être synthétisées au schéma de la figure 2.1

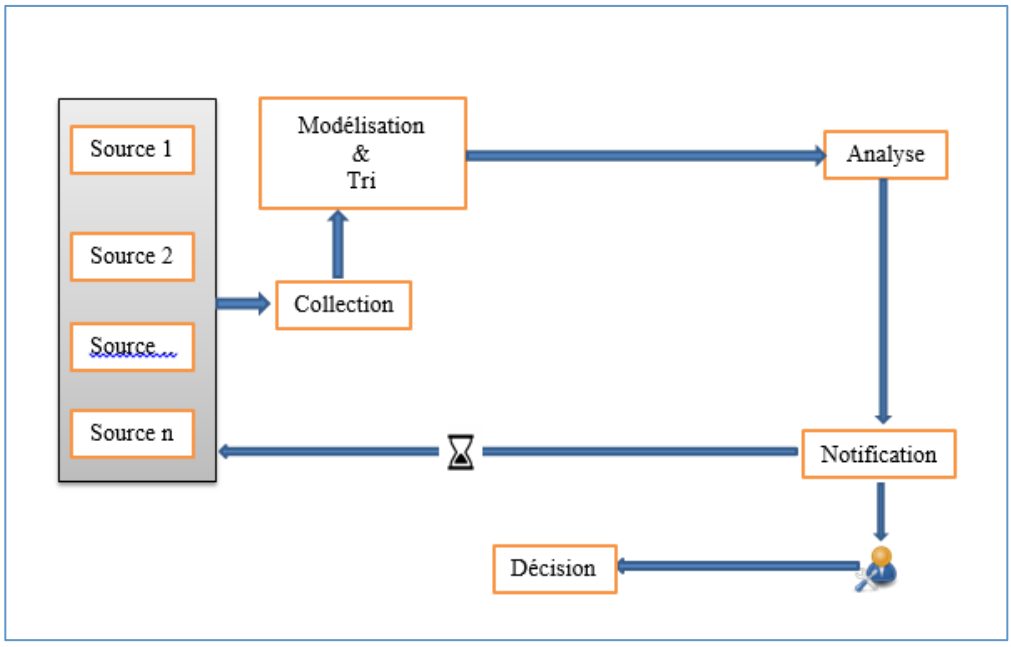

Figure 2.1 Étapes du cycle de vie de surveillance d'infrastructure TI

2.3 Objectifs de la recherche

Les objectifs de ce projet de recherche sont aussi de pouvoir collecter : 1) des informations des fichiers journaux; et 2) des mesures de conteneurs DOCKER, et 3) des mesures du système sur lequel ils sont hébergés. Pour ce faire, la définition des termes suivants s'avère nécessaire : fichier journal, évènement et métrique.

2.3.1 Fichier Journal

Un fichier journal est un fichier dans lequel tous les évènements d'un système ou d'une application sont inscrits et qui peut être consulté à volonté en vue de détecter les causes de

certaines pannes ou de certains problèmes de performance. Presque tous les fichiers journaux de système hôte se trouvent dans le répertoire /var/log et peuvent être consultés grâce aux commandes «*less», «cat», «tail», «head»* pour ne citer que celles-là (cf. ANNEXE V). Mais pour les conteneurs Docker, les fichiers «logs» peuvent se trouver à l'intérieur des conteneurs ou dans les dossiers «*cgroups»* montés. On peut visualiser les journaux d'un conteneur spécifique en utilisant la commande «DOCKER logs ID_Conteneur» [33].

2.3.2 Évènement

Un évènement est toute action produite par le système lui-même ou par une application. Slideshare [28] le définit comme « le message réel plus un tas de métadonnées ». Chaque évènement produit est inscrit dans des fichiers journaux ce qui permet de faire des analyses ultérieures en vue de prendre certaines décisions. En général, les évènements du système sont stockés dans le fichier «syslog» et peuvent être visualisés par une lecture de ce fichier à l'aide des commandes *«dmesg»* ou *«cat syslog».* Dans le cas de DOCKER, les évènements sont redirigés vers «STDOUT» ou «STDERR», car tous les conteneurs n'utilisent pas «syslog». À partir de la version 1.6 de DOCKER, le concept de pilote de journaux (c.-à-d. logging drivers en anglais) a été introduit et ceci a facilité la gestion des évènements des conteneurs de DOCKER. [34]

2.3.3 Métrique

En logiciel, une métrique (synonyme de mesure) est la mesure d'une caractéristique particulière, par exemple la performance, d'un logiciel [29]. Voici des exemples de métriques : les différentes valeurs de mesure du «CPU», de la mémoire consommée, de la mémoire cache, des flux de données, des trafics sur le réseau. Les commandes *«top»* et *«tload»* peuvent être utilisées pour visualiser la performance d'un système Linux, mais dans le cas des conteneurs DOCKER, les commandes suivantes sont de préférence utilisées : «*docker top ID_Conteneur*» et «*docker stats ID_Conteneur*». Les figures A IV-1 et A IV-2 présentent respectivement les résultats des commandes « *docker stats `docker ps –qa`»* et *«top»*.

2.4 Objectifs techniques du projet

Les étapes du cycle de vie de surveillance d'un système étant identifiées, il est utile de les appliquer à la surveillance des conteneurs DOCKER et ainsi les utiliser pour préciser les objectifs du projet de recherche appliquée synthétisés au tableau suivant.

Tableau 2-1 Objectifs techniques du projet

Objectifs :	Description
Centraliser les évènements du système hôte et des conteneurs DOCKER.	Capturer et rediriger les évènements produits dans les conteneurs vers le serveur des journaux du système (c.-à-d. «logging server») communément appelé «syslog».
Choix d'un logiciel libre collecteur	Des différents logiciels de collection expérimentés, choisir celui pouvant permettre de collecter les fichiers journaux et les métriques des conteneurs ainsi que du système hôte avec la meilleure performance.
Modélisation des données collectées	Trouver un logiciel libre permettant de traiter les données en vue de les transformer sous un format unique et les filtrer.
Indexation des données	Sélectionner un logiciel libre pouvant stocker et indexer les données en vue de pouvoir les retrouver facilement.
Combinaison des différents logiciels libres	Configurer et combiner les logiciels en vue de pouvoir obtenir des données prêtes à être présentées dans un tableau de bord.
Création d'un prototype de tableau de bord	Concevoir un prototype de tableau de bord présentant les métriques.

2.5 Choix des logiciels libres de surveillance des conteneurs DOCKER

En se basant sur les objectifs du projet et sur les différentes étapes du cycle de vie de surveillance d'un système, les logiciels libres suivants ont été identifiés pour construire un

système de surveillance : Collectd, Logstash, Elasticsearch et Kibana. Les raisons de ces choix sont décrites dans ce chapitre. Ces quatre logiciels libres sont disponibles actuellement. Le concepteur du logiciel Elasticsearch, a embauché les concepteurs de Logstash et de Kibana récemment. Maintenant, cette liste de logiciels, ensemble, se nomme la pile ELK : Elasticsearch, Logstash et Kibana. [32]

2.6 Présentation des composantes logicielles de la solution proposée

2.6.1 Outil de collection des évènements et métriques

Parmi les nombreux logiciels de collection de fichiers journaux et de mesures existants, tels que: Fluentd [26], EventLog Analyzer [27], SumoLogic [47], le choix a été porté sur «collectd» qui permet de collecter les métriques et journaux de l'ensemble des conteneurs opérationnels de Docker et du système hôte. «Collectd» est un logiciel puissant qui permet de collecter périodiquement des statistiques de performance de système avec opportunité de les sauvegarder dans différents formats tels que CSV et RRD [30], il a été priorisé par rapport aux autres logiciels pour les raisons suivantes :

- L'ensemble de ses «plugins», qui sont des scripts (et il y en a plus de 90), [30] permettent d'aller chercher les différentes données dans les différentes sources du système,
- Il permet de définir certains seuils à travers le «plugin *threshold»* et de notifier par courriel (c.-à-d. «*notify_email*») ou par message sur le bureau (c.-à-d. «*notify_desktop»*) le gestionnaire du système si ces seuils ne sont pas respectés,
- Il permet d'injecter des scripts dans les «*plugins Exec»* ou «*Python»* pouvant permettre d'aller chercher des données beaucoup plus spécifiques, dont les «plugin», disponibles par défaut dans «collectd», n'en font pas collection.

Parmi les «plugins» disponibles, les suivants sont ceux qui sont indispensables dans l'atteinte des objectifs fixés et leurs utilisations sont décrites à la section wiki du site officiel de «collectd» [44] :

- *CPU* : Ce «plugin» permet de collecter le temps que passe chaque CPU dans différents états notamment dans l'exécution des codes d'usager ou dans l'attente d'opérations IO.
- *CPUFreq* : Comme son nom l'indique, il permet de collecter la fréquence actuelle du CPU.
- *Exec* : Exécute des applications ou des scripts puis écrit les informations retournées par ces programmes au «STDOUT».
- *Interface* : Collecte les informations à propos du trafic (c.-à-d. exprimé en Octet par seconde) et le nombre d'erreurs par seconde sur une interface.
- *Load* : Ce «plugin» permet de faire la collecte des charges du système.
- *Memory* : Ce «plugin» permet de faire la collecte des informations sur l'utilisation de la mémoire physique.
- *Network* : Ce «plugin» permet d'envoyer vers d'autres instances ou de recevoir des valeurs d'autres instances.
- *Notify_Email* : Disponible à partir de la version 4.3 de «collectd», ce «plugin» permet d'envoyer des notifications par courriel en utilisant «libESMTP[2]».
- *Python* : Ce «plugin» intègre un interpréteur en langage Pyton dans «collectd» et expose les «API» aux scripts en langage «Pyton». Il permet d'écrire des scripts personnalisés en utilisant ce langage de plus en plus populaire.
- *Threshold* : Disponible à partir de la version 4.3, ce «plugin» analyse les données collectées et génère une alerte puis envoie une notification au cas où un problème est détecté avec une des trois sévérités suivantes : Okay, Warning, Failed.

2.6.2 Outil de filtrage et de modélisation des données

Après la collection des données, afin de ne pas se retrouver avec un ensemble de données dont un pourcentage très significatif peut ne pas être d'un grand intérêt pour le gestionnaire, le logiciel «logstash» peut être utilisé complémentairement à «collectd» pour transformer les

[2] libESMTP est une librairie utilisée pour gérer les courriels en utilisant un MTA préconfiguré [50]

données collectées dans un format clair avec des couples clé-valeur exploitables par la suite [31], au besoin, les filtrer et enfin les rediriger vers la sortie désirée : «stdout», fichier ou «Elasticsearch». Il peut être aussi utilisé pour analyser et stocker des évènements.

Logstash fonctionne sur le principe de blocs de configuration ou le traitement des données collectées peut être fait suivant la chaîne suivante [31] :

- ***Input*** : «plugin» permettant de générer des évènements ou de les lire à partir d'autres sources. Les «Inputs» les plus populaires sont : «*file, stdin, syslog et udp»,*
- ***Filter*** : «plugin» permettant de filtrer et mettre les évènements dans un format plus facile à manipuler. Les filtres les plus populaires sont : «*date, grep, grok, multiline et mutate»,*
- ***Output*** : «plugin» permettant d'envoyer les données traitées vers d'autres applications ou les afficher directement sur l'écran. Les «outputs» les plus populaires sont : «*email, file, mongodb, nagios, redis, stdout et elasticsearch».*

2.6.3 Outil d'indexation et de stockage de données

Les informations collectées étant filtrées, et traitées, le moteur de recherche distribué basé sur Apache Lucene, «Elasticsearch», sera utilisé pour indexer les informations et les stocker dans une base de données NoSQL ou RESTful qui lui est intégrée. Il ne requiert pas de configuration spéciale et son installation en tant que service n'est pas requise dans le cadre des expérimentations faites pour ce projet.

2.6.4 Outil de visualisation des données

Kibana a été choisi comme plateforme de visualisation des données analysées et stockées dans Elasticsearch. D'autres outils comme Graphite et Nagios pourraient être utilisés, mais vu que la pile ELK est déjà intégrée, il est plus judicieux d'utiliser Kibana en combinaison avec Logstash et Elasticsearch plutôt que d'autres logiciels libres.

2.7 Représentation graphique des logiciels choisis

Les logiciels choisis peuvent être représentés graphiquement par la figure suivante :

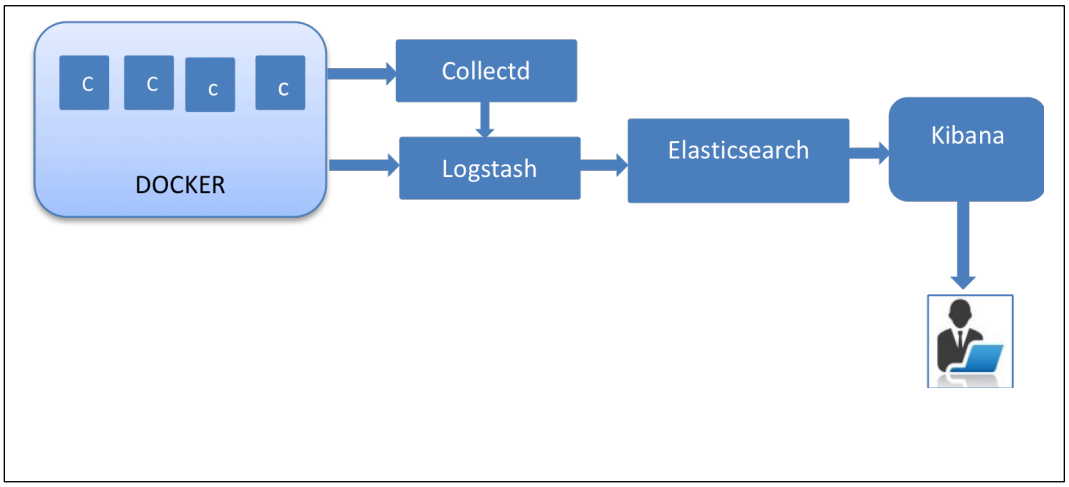

Figure 2.2 Système de surveillance proposé

2.8 Mise en place du système de surveillance proposé

Pour pouvoir interconnecter les différents logiciels, des ajustements ont été apportés au niveau des fichiers configuration de chaque application en spécifiant l'adresse IP et le port du serveur sur lequel les informations seront transmises. Ainsi l'interconnexion a été faite grâce aux configurations suivantes :

2.8.1 Configuration de COLLECTD

Vu que «collectd» peut agir à la fois comme client et serveur et que lors de l'expérimentation aucune autre instance de «collectd» n'a été utilisée alors il a été configuré de sorte que la machine considérée joue les deux rôles, ainsi les lignes suivantes ont été ajoutées dans le fichier «collectd.conf» localisé par défaut dans le répertoire «/etc/collectd/» :

```
Hostname    "192.168.2.29"
LoadPlugin network
<Plugin network>
#       # client setup:
        <Server "192.168.2.29" "25826">

        </Server>
#       # server setup:
        <Listen "192.168.2.29" "25826">
                Interface "wlan0"
        </Listen>
```

Figure 2.3 Configuration de «collectd» comme serveur et client

2.8.2 Configuration de LOGSTASH

Un fichier configuration a été créé pour saisir à son entrée les informations envoyées par «collectd» et «syslog» et les rediriger vers Elasticsearch grâce aux instructions suivantes :

```
# Les sources de logstash
input {
# Requis : Informations provenant de collectd pour collecter les métriques
        udp {
                host => "192.168.2.29"
                port => 25826
                codec => collectd { }
                type => collectd
        }
# Requis : Informations provenant de syslog pour collecter les évènements
        syslog {
                port => 5000
                type => syslog
        }
# Non requis : Informations provenant de stdin c.-à-d. entrées au clavier, pour
tester
        stdin {
        }
}
```

Figure 2.4 Configuration des entrées de logstash

```
# Les destinations de logstash
output {
# Informations affichées à l'écran
      stdout {
            codec => rubydebug              # Utilise le format JSON
      }
# Informations envoyées vers elasticsearch sur le port 9200
      elasticsearch {
            hosts => "192.168.2.29:9200"
            index => "logstash-%{+YYYY.MM.dd}"
            codec => rubydebug              # Utilise le format JSON
      }
}
```

Figure 2.5 Configuration des sorties de logstash

2.8.3 Configuration de ELASTICSEARCH

En spécifiant l'adresse IP et le port par défaut 9200 au «plugin» Elasticsearch à la sortie de Logstash, l'interconnexion s'est tout de suite faite. Il ne reste qu'à exécuter Elasticsearch qui ne requiert pas de configuration pour pouvoir fonctionner correctement. Pour exécuter Elasticsearch sans l'installer comme service, il suffit de taper cette ligne de commande dans le répertoire «bin» d'Elasticsearch qui aura été téléchargé et extrait, où *ajr_cl* et *ajr_cl_node_01* représentent respectivement le nom de la grappe et le nom du nœud créés :

```
./elasticsearch --network.host _non_loopback_ --cluster.name ajr_cl --node.name ajr_cl_node_01
```

Figure 2.6 Lancement d'elasticsearch

2.8.4 Configuration de KIBANA

Tout comme Elasticsearch, Kibana ne requiert aucune configuration particulière pour pouvoir s'interconnecter avec Elasticsearch. Grâce à son interface web, en quelques clics on

peut s'y connecter, il suffit de connaître le nom d'un indice créé dans le «plugin elasticsearch[3]» de la sortie de Logstash et le spécifier dans le champ requis à cet effet dans Kibana.

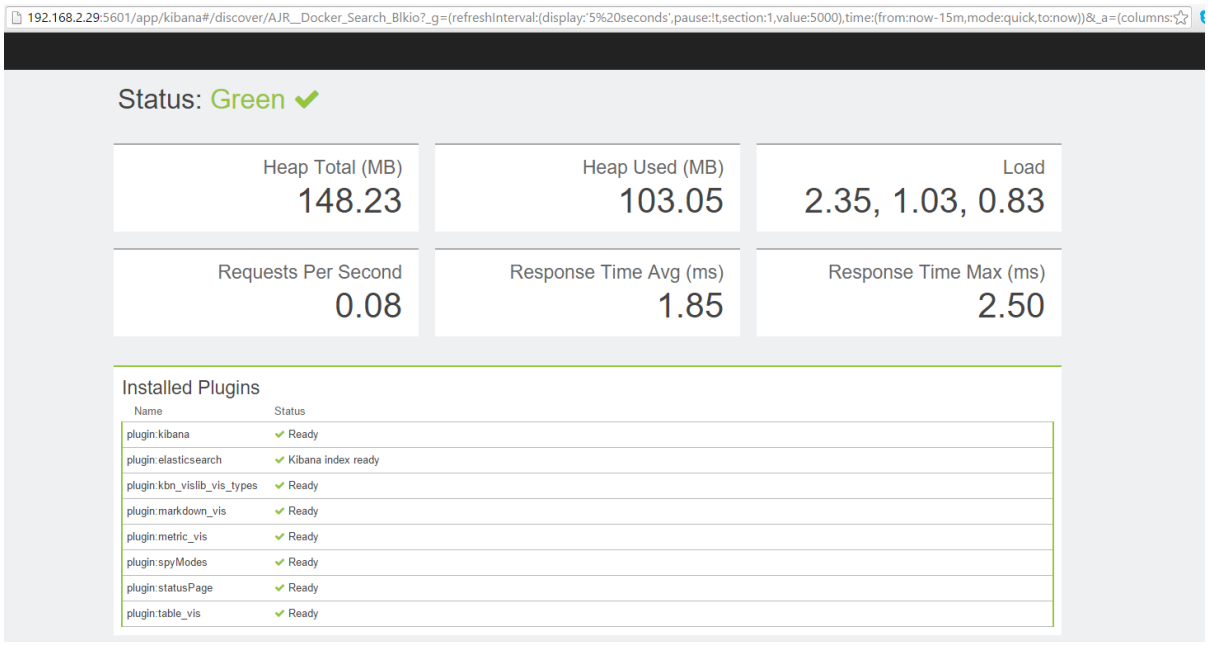

Figure 2.7 Aperçu du statut de Kibana

2.9 Les données qui seront collectées

L'outil «*collectd*» du système de surveillance proposé, et présenté à la section 2.6.1, permettra la collection des données suivantes :

2.9.1 Les métriques

L'objectif étant de collecter les métriques des conteneurs opérationnels et du système hôte, deux catégories de métriques seront collectées :

[3] Ici «elasticsearch» désigne le nom du «plugin » spécifié à la sortie de «logstash» et non le logiciel libre.

2.9.2 Au niveau des conteneurs DOCKER

À l'aide du «plugin» *docker-collectd-plugin,* disponible sur github[4] [35], utilisant les nouvelles statistiques d'API [36] de DOCKER, disponibles depuis la version 1.5, les informations suivantes pourront être collectées :

- Mémoire : pour chaque conteneur, les métriques de la figure suivante seront obtenues

```
cache 11492564992
rss 1930993664
mapped_file 306728960
pgpgin 406632648
pgpgout 403355412
swap 0
pgfault 728281223
pgmajfault 1724
inactive_anon 46608384
active_anon 1884520448
inactive_file 7003344896
active_file 4489052160
unevictable 32768
hierarchical_memory_limit 9223372036854775807
hierarchical_memsw_limit 9223372036854775807
total_cache 11492564992
total_rss 1930993664
total_mapped_file 306728960
total_pgpgin 406632648
total_pgpgout 403355412
total_swap 0
total_pgfault 728281223
total_pgmajfault 1724
total_inactive_anon 46608384
total_active_anon 1884520448
total_inactive_file 7003344896
total_active_file 4489052160
total_unevictable 32768
```

Figure 2.8 Métriques sur la mémoire contenue dans les groupes de contrôle [36]

[4] Github est, selon openclassroom, un site web collaboratif basé sur Git. « Il est considéré comme une sorte de réseau social pour les développeurs ». (Openclassroom, 2015)

- CPU : pour chaque conteneur, on obtiendra l'utilisation du «CPU» accumulé par les processus du conteneur subdivisés en temps d'utilisateur, c'est-à-dire le temps durant lequel les processus sont en contrôle direct du «CPU», et de système, c'est-à-dire le temps durant lequel le «CPU» exécute des appels système en lieu et place des processus. Ces temps sont exprimés en centième de seconde.

- Bloc I/O : Les métriques du bloc I/O collectées par conteneur sont divisées en blkio.sectors, blkio.io_service_bytes, blkio.io_serviced et blkio.io_queued

- Réseau : Pour chaque conteneur, les flux de paquets envoyés et reçus sur le réseau seront collectés. [46]

2.9.3 Au niveau du système hôte

Pour le système hôte, les métriques sur les ressources suivantes seront collectées :

- CPU,
- Mémoire,
- Network I/O,
- SWAP, et
- Fichier système du Disque (c.-à-d. DF ou «Disk FileSystem»).

2.9.4 Les évènements

Contrairement aux métriques, tous les évènements des conteneurs DOCKER seront redirigés, dès leur exécution, vers un serveur de journaux soit le «*syslog*» qui contient aussi les évènements du système hôte et par conséquent tous seront collectés. Les évènements des conteneurs DOCKER devraient idéalement contenir les informations suivantes [28] :

- La date à laquelle l'évènement s'est produit, très importante, car elle permet de retracer les évènements. Savoir à quel moment un évènement a été produit pourra

permettre de déterminer la cause d'un incident produit à un moment précis. Elle doit être complète en suivant le format conforme à ISO8601[5],

- Le nom de la machine hôte ou son adresse IP, information importante pouvant nous aider à déterminer sur quel hôte le conteneur est hébergé dans le cas où l'on étend la solution sur des grappes de serveurs (c.-à-d. clusters en anglais),
- L'ID ou le nom du conteneur pour identifier l'instance qui a généré l'évènement,
- L'ID de l'image pour faire la relation avec les évènements des autres conteneurs, et
- L'ID du processus pour corréler avec les autres évènements du processus s'il n'y a pas d'autres moyens de l'identifier.

2.10 Expérimentations de la proposition

2.10.1 Objectifs des expérimentations

La proposition du système de surveillance des conteneurs DOCKER est basée sur les connaissances théoriques et le potentiel décrit des différentes composantes logicielles. Pour les raisons suivantes, une phase expérimentale est nécessaire :

- Des tests des logiciels proposés, ainsi que la collecte de mesures réelles, vont permettre d'évaluer la proposition en vue de confirmer si elle fonctionne telle que désirée;
- La phase expérimentale permettra également de détecter les faiblesses de la proposition, ce qui permettra d'évaluer les correctifs nécessaires, et
- La phase expérimentale permettra d'identifier les limites de la solution proposée et ainsi faire certaines recommandations en vue de continuer ce projet de recherche.

[5] ISO 8601 décrit un mode de représentation numérique de la date et de l'heure accepté à l'échelon international. [49]

2.10.2 Mise en situation

2.10.3 Mise en place de l'environnement

Pour effectuer les expérimentations, un ordinateur portatif ayant les spécifications suivantes a été utilisé. La figure A I.1 présente les informations sur le système hôte sur lequel les tests ont été effectués :

- Nom de l'ordinateur : FDNTBK,
- Adresse IP de la machine : 192.168.2.29,
- Système d'Exploitation : Ubuntu 14.04 (Trusty),
- Mémoire : 2 Go,
- Disque Dur : 150 Go, et
- Version du noyau : 3.16.0-50-Generic.

La version 1.8.3 de DOCKER a été installée sur cet ordinateur en suivant les étapes présentées à l'ANNEXE I. Pour tester la technologie DOCKER, l'image officielle de MYSQL a été téléchargée, à partir de laquelle deux conteneurs ont été créés. Un pour une version serveur de MYSQL et l'autre pour une version cliente qui sera lié au conteneur roulant la version serveur [cf ANNEXE II]. Chaque composante de l'architecture peut être installée sur un serveur différent, mais dans le cadre de cette expérimentation, toutes les composantes, c'est-à-dire Collectd, Logstash, Elasticsearch et Kibana ont été installées et exécutées sur une seule et même machine. Pour collecter les évènements des conteneurs et du système, un fichier *syslog.conf* a été créé dans lequel les évènements sont redirigés vers le port 5000 de la machine test en suivant le protocole UDP.

2.10.4 Les tests effectués

Afin de tester la proposition de solution, des conteneurs seront créés en suivant la nomenclature suivante : «*ajrcont0X*» où *X* varie de 1 à 9 afin de faciliter l'identification de chaque conteneur. En vue de générer certains trafics dans les conteneurs, des mises à jour

seront lancées et des scripts comme l'affichage en boucle d'une chaîne de caractères y seront exécutés.

Pour voir comment les ressources attribuées aux conteneurs affectent le comportement des conteneurs, la quantité de ressources allouées à certains conteneurs seront limitées à l'aide des paramètres ci-dessous qui seront passés en ligne de commande lors de l'exécution des conteneurs :

- Pour limiter la mémoire allouée au conteneur : «-m ou --memory».
- Pour limiter le quota CFS de CPU : «--CPU-quota».

La figure suivante présente les activités en cours dans trois conteneurs en exécution identifiés par les terminaux 1, 2, et 3; et le terminal 4 présente une partie des résultats retournés par Logstash sur les informations collectées pour le conteneur «*ajrcont01*».

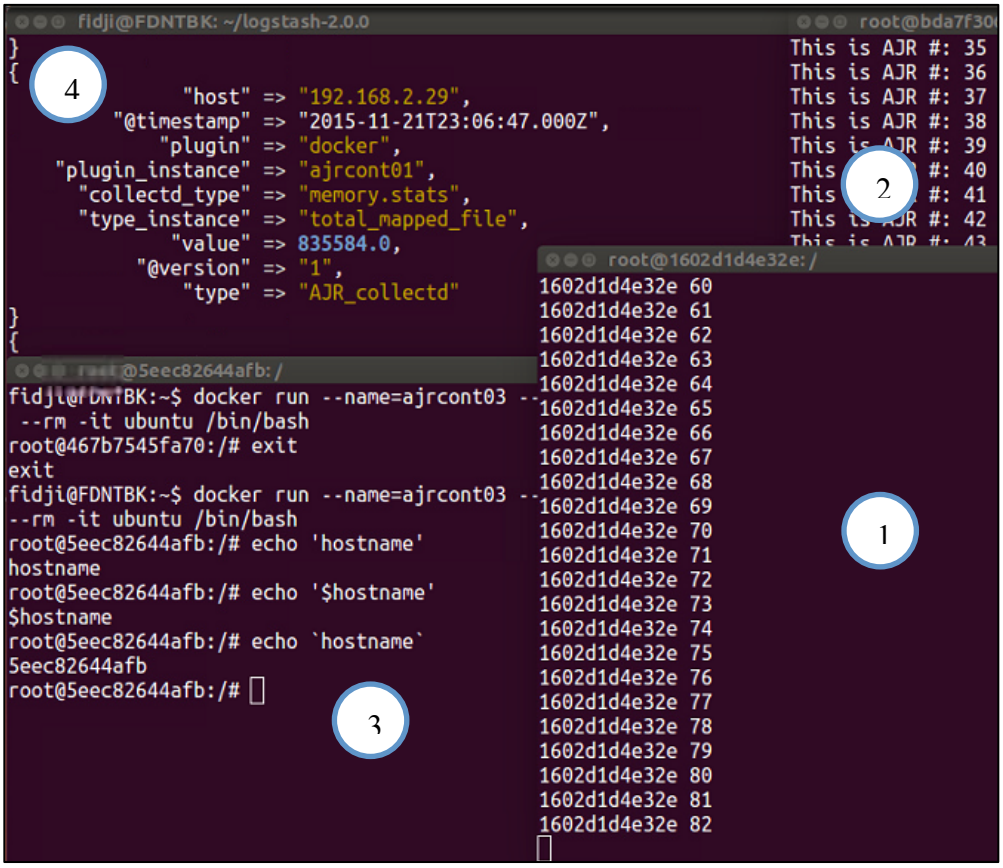

Figure 2.9 Conteneurs opérationnels et aperçu de Collectd

2.11 Conclusion

Suite à l'exécution de toutes les étapes de la méthodologie adoptée, une proposition de solution afin de résoudre la problématique de cette recherche a été présentée. Les logiciels libres identifiés et choisis sont tous compatibles les uns avec les autres et sont faciles à installer et à configurer. Logstash et Collectd sont deux outils puissants de collection de données et semblent très flexibles et permettent de collecter les informations désirées et les mettre sous un format tout à fait compréhensible par un utilisateur après les avoir filtrées. Elasticsearch est un moteur de recherche visant à faire l'analyse, en temps réel, et qui est également utilisé pour indexer et stocker des données, ce qui faciliterait la recherche d'information pour des analyses ultérieures. Finalement, Kibana propose de visualiser les informations collectées grâce à un tableau de bord personnalisable. La combinaison de ces logiciels libres pourrait être une solution pouvant donner, à tout gestionnaire d'infrastructure DOCKER, une visibilité sur tous les conteneurs de son système. Grâce à cette solution, il pourrait savoir, en quasi temps réel, les ressources consommées par les conteneurs et leurs états de santé. Le prochain chapitre fait état des résultats de l'utilisation de la solution proposée.

CHAPITRE 3

PRÉSENTATION ET ANALYSE DES RÉSULTATS

Ce chapitre présente la synthèse de l'expérimentation de la proposition de solution sous forme de graphiques de résultats de la surveillance de quelques conteneurs DOCKER configurés en utilisant la solution proposée. Une présentation des performances de la proposition ainsi que des éléments bloquants y est faite également. Ce chapitre se termine avec la description des tâches recommandées pour une itération future en vue d'améliorer cette première proposition de système de surveillance pour DOCKER.

3.1 Présentation des métriques collectées

Tel que présenté au chapitre précédent (cf. section 2.9.2), toutes les métriques, exprimées en bits, relatives à la mémoire ont été collectées pour chaque conteneur et la figure suivante les présente à l'aide du logiciel Kibana ainsi que leurs valeurs pour le conteneur *ajrcont01*.

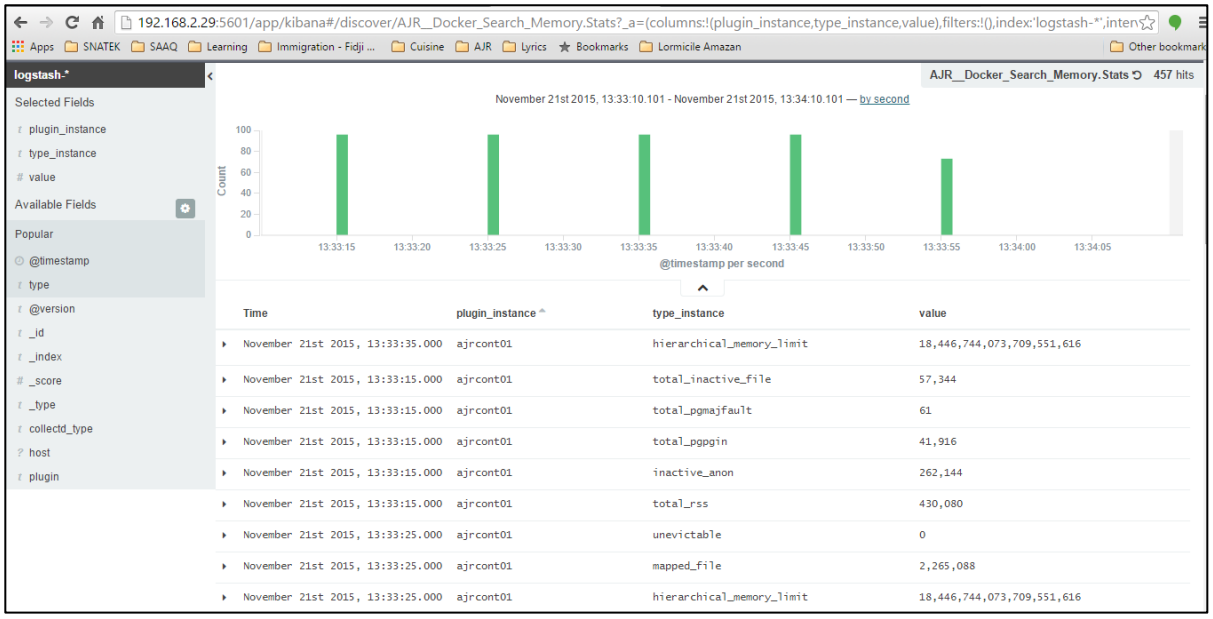

Figure 3.1 Métriques collectées sur la mémoire utilisée par un conteneur : *ajrcont01*

Pour aboutir à ce résultat, une recherche a été lancée dans le champ *«search»* de Kibana avec l'expression suivante : *collectd_type="memory.stats"*.

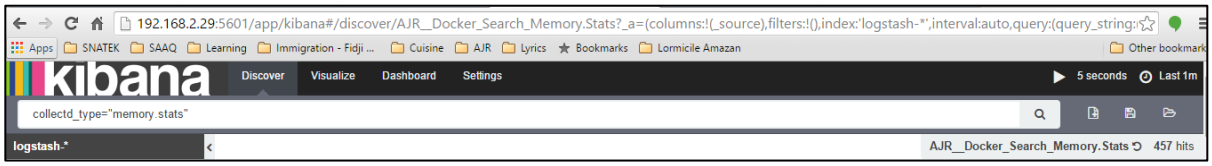

Figure 3.2 Recherche du mot-clé *memory.stats*

De même en lançant une recherche sur «*blkio*» dans Kibana, il retourne toutes les métriques collectées relatives au Bloc I/O.

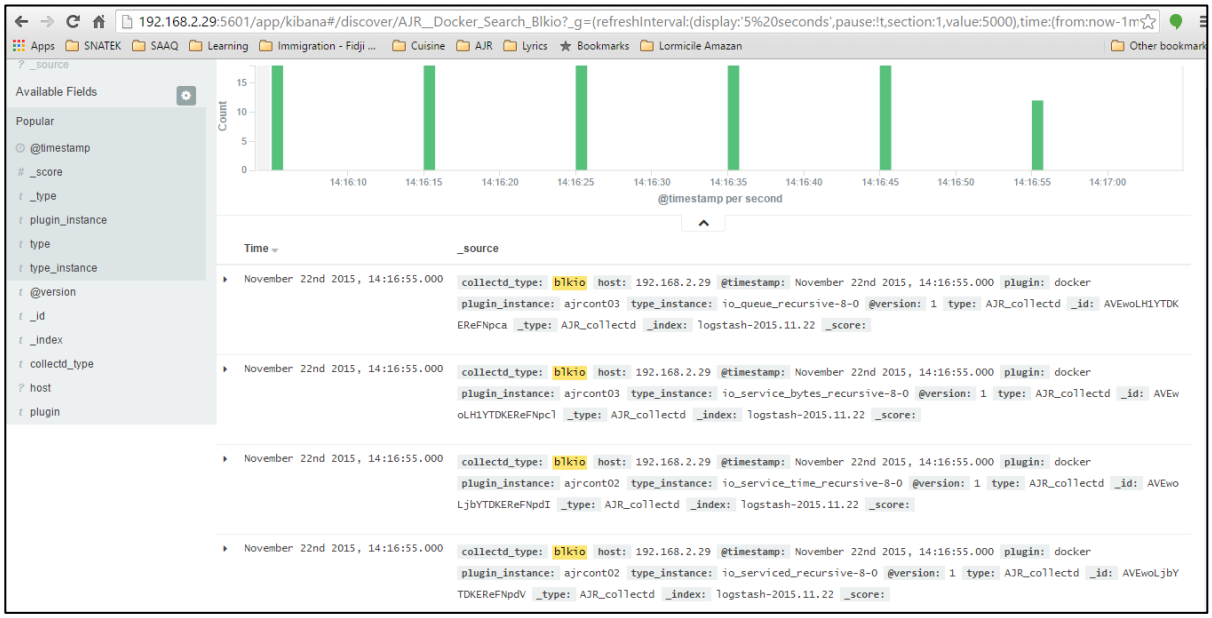

Figure 3.3 Liste des métriques relatives au Bloc I/O collectées pour chaque conteneur

Pour voir si effectivement les évènements et les métriques relatives au réseau et au CPU et à n'importe quelle autre source ont été collectés, il suffit de lancer la recherche avec les bons mots-clés.

À partir des résultats de ces recherches, il est facile de construire, par la suite, des visualisations en utilisant l'outil de visualisation jugé le plus adapté pour représenter les métriques ou informations pertinentes.

3.2 Les Tableaux de bord

La combinaison des outils présentés dans le chapitre II de ce rapport a permis de créer ces prototypes de tableau de bord pouvant permettre d'avoir une vue globale sur les métriques et évènements du système surveillé.

Trois prototypes de tableau de bord sont présentés afin d'éviter de surcharger un seul tableau de bord qui contiendrait toutes les informations collectées sur le système et les conteneurs DOCKER.

3.2.1 Tableau de Bord : ERROR

Ce tableau de bord présente uniquement les occurrences du mot «Error» dans le fichier «syslog» peu importe que l'évènement soit produit par la machine hôte, par une application, par un programme ou un conteneur DOCKER opérationnel comme indiqué dans la colonne *Program* de la figure ci-dessous.

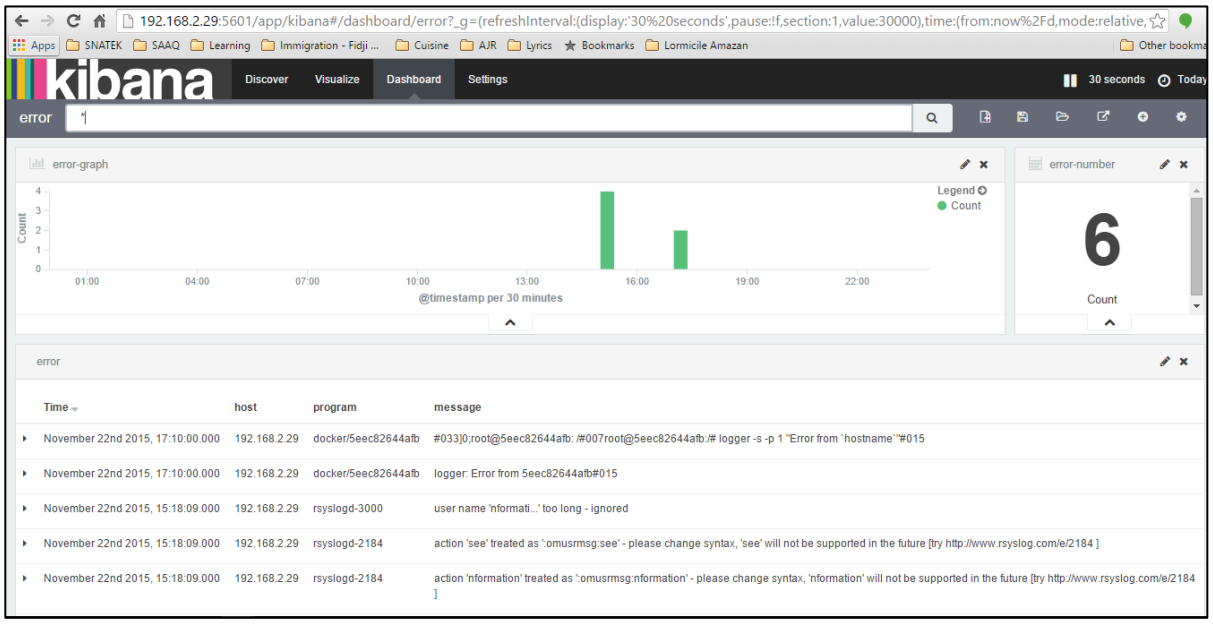

Figure 3.4 Tableau de bord Error

3.2.2 Tableau de Bord : HARDWARE

Dans ce tableau de bord sont présentées quelques mesures des ressources du système hôte telles que l'utilisation du CPU, de la mémoire, de l'espace disque ainsi que le SWAP.

Figure 3.5 Tableau de bord Hardware

3.2.3 Tableau de Bord : DOCKER

Afin de pouvoir voir quelques activités dans le prototype de tableau de bord nommé DOCKER qui affiche des vues de certaines métriques sur les conteneurs opérationnels tels que le nombre de conteneurs en exécution, le Bloc IO par conteneur, le pourcentage de CPU utilisé, il a fallu créer quelques conteneurs, trois au total, et y générer certains trafics. La figure 3.6 permet d'identifier quel «containerID» est associé à quel nom, la figure 3.7 donne les statistiques (c.-à-d. en utilisant les commandes de Docker) sur les ressources utilisées par

les conteneurs et les figures 3.8 et 3.9 montrent les différentes métriques relatives aux conteneurs en exécution.

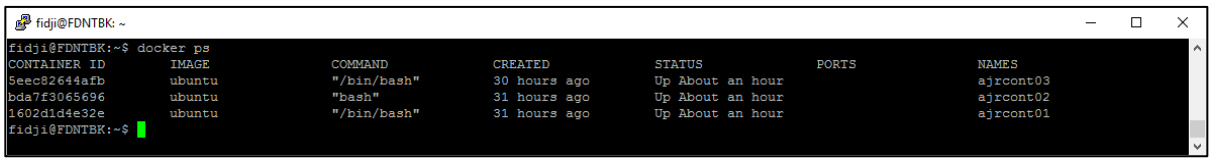

Figure 3.6 Liste des conteneurs en exécution faisant le lien entre leur nom et leur ID

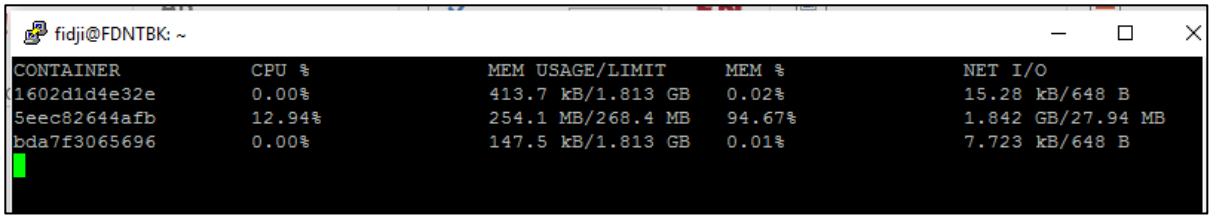

Figure 3.7 Statistiques sur les DOCKER en opération obtenues par «*docker stats*»

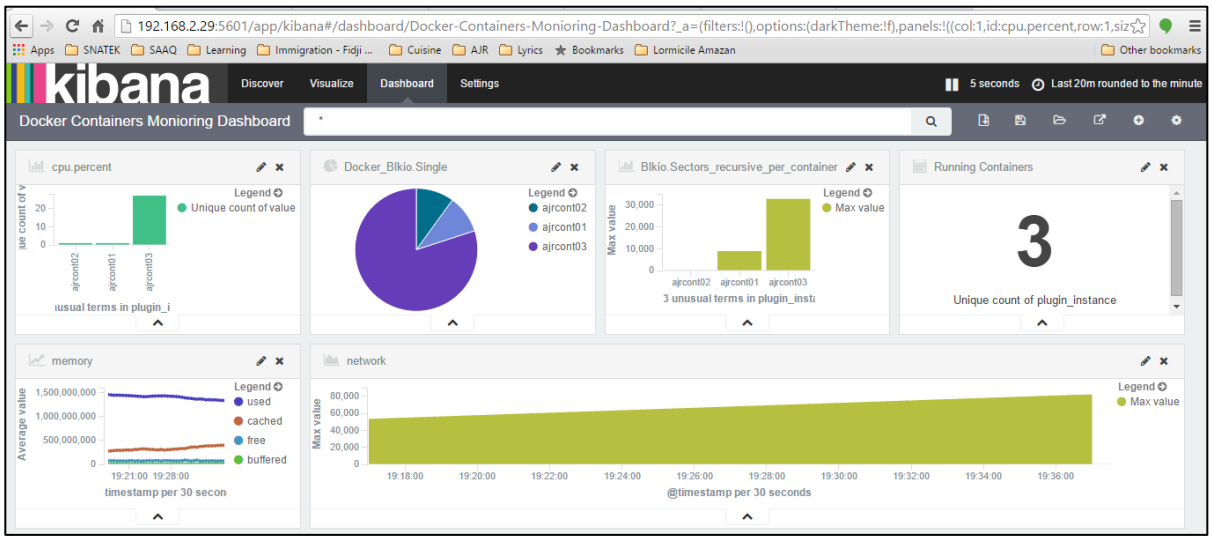

Figure 3.8 Tableau de bord affichant les métriques sur les DOCKER en opération

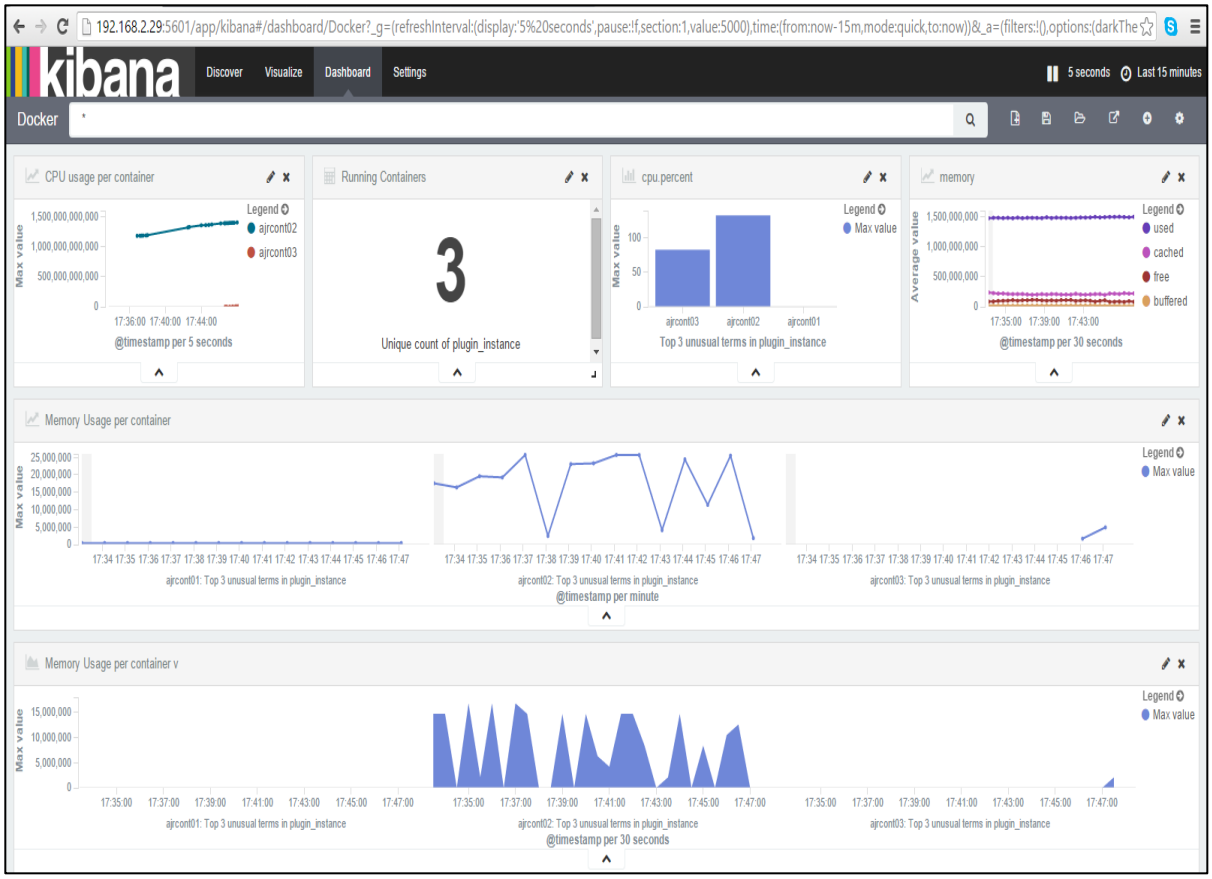

Figure 3.9 Tableau de bord DOCKER suivant une autre disposition

3.3 Les alertes et notifications

Tel que mentionné lors de la présentation de l'outil «*collectd*», ce dernier offre la possibilité à travers ses différents «plugins» de définir certains seuils et envoyer certaines notifications dans le cas où ce seuil serait franchi. En vue de tester ces fonctionnalités, des seuils ont été définis de telle sorte qu'ils soient violés assurément même sur un court délai de cinq à dix minutes. En configurant un seuil minimal de valeur «1» et un seuil maximal de valeur «10» dans le «plugin threshold» pour les conteneurs DOCKER faisant usage du cpu1 comme présenté à la figure 3.10, il a été observé que les notifications sont envoyées au récipient défini dans le «plugin notify_email» comme espéré.

Ainsi, en exécutant la commande «apt-get update» à l'intérieur du conteneur «ajrcont03», les notifications envoyées se présentent sous cette forme :

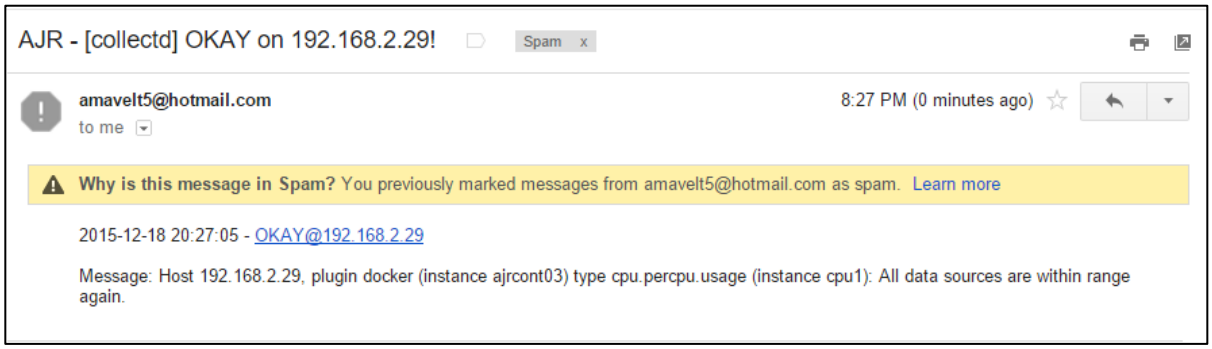

Figure 3.10 Notification reçue quand les seuils sont respectés

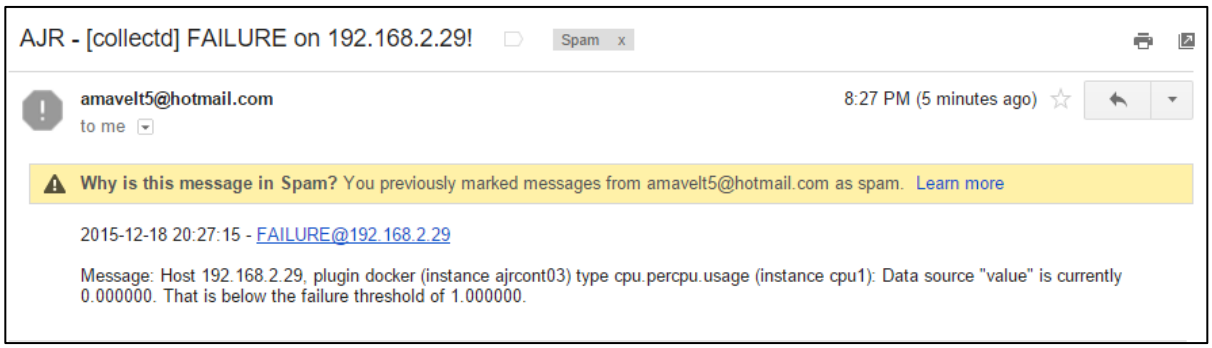

Figure 3.11 Notification reçue quand la valeur est en dessous du seuil minimal

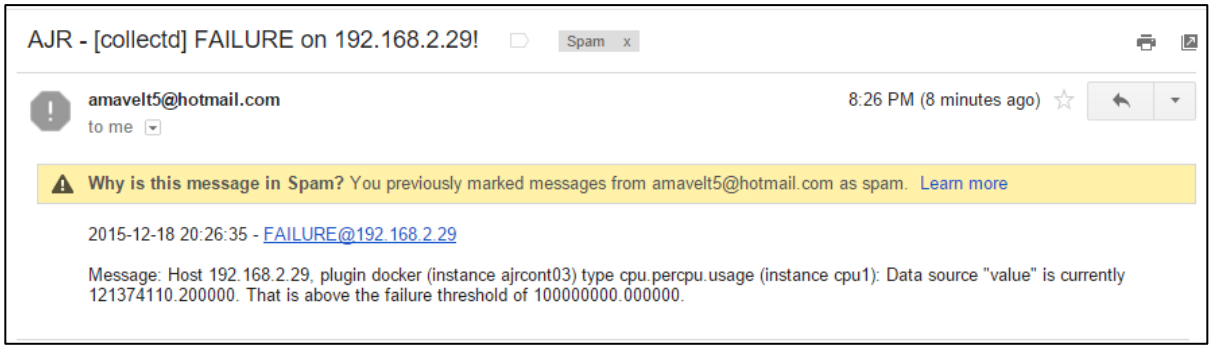

Figure 3.12 Valeur du CPU est au-dessus du seuil maximal fixé

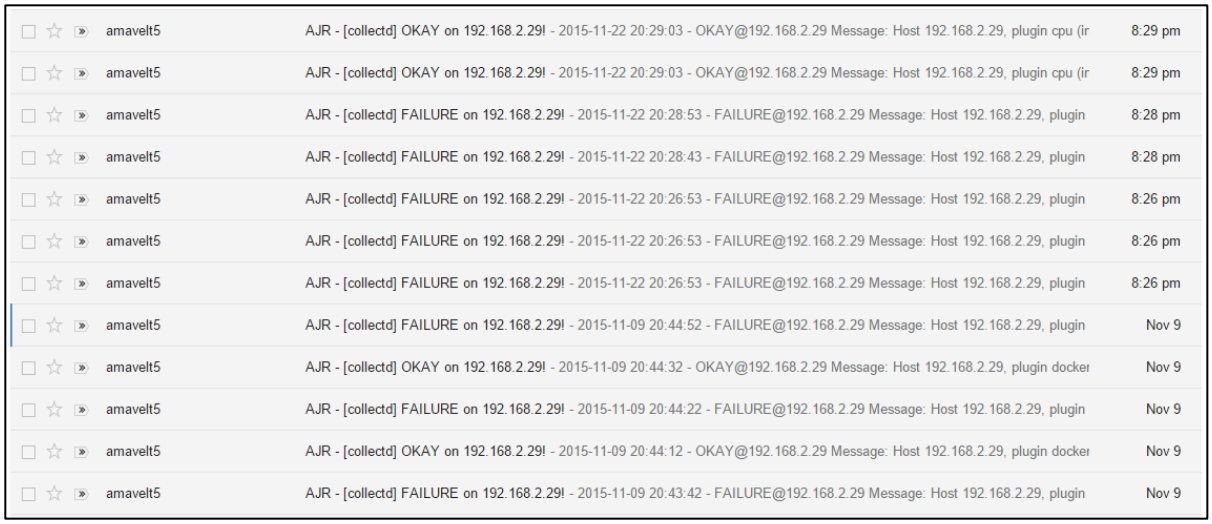

Figure 3.13 Présentation du flux de notifications reçues

À noter que tel que décrit dans la documentation du «plugin» traitant de la notification dans le logiciel «collectd», les notifications n'ont que trois types de niveau de sévérité : «OKAY», «WARNING» et «FAILURE». Aussi, il est observé que le flux de courriels de notification reçu est conforme à ce que dit la documentation du site officiel de «collectd».

3.4 Performance de la proposition

Les expérimentations ont été effectuées sur un ordinateur à faible capacité, soit deux Gigaoctets de mémoire vive et un CPU AMD bi-cœur. L'expérimentation s'est montrée très performante quoique toutes les composantes du système de surveillance proposé aient été installées sur cette machine, lui assignant les rôles suivants :

- Serveur hôte DOCKER,
- Serveur Rsyslog,
- Serveur Collectd,
- Serveur Logstash,
- Serveur Elasticsearch,
- Serveur Kibana.

En effet, le démarrage des composantes se fait en quelques secondes. Lors des tests effectués, les messages saisis au «STDIN» sont affichés instantanément au «STDOUT» et sont disponibles en quasi-temps réel dans Kibana. En configurant Kibana pour se rafraîchir toutes les cinq secondes, les variations des valeurs des métriques peuvent être vues dans le tableau de bord.

3.5 Les éléments bloquants observés

Au cours de ce projet, les difficultés rencontrées ont été nombreuses, et ce pour différentes causes:

- La technologie DOCKER étant nouvelle, il était nécessaire de se familiariser avec elle avant de s'attaquer au cœur du projet (c.-à-d. sélectionner les logiciels pouvant permettre de faire une surveillance efficace des conteneurs DOCKER). Par conséquent, consacrer du temps à effectuer des recherches sur DOCKER et l'expérimenter afin d'en connaître les bases et avoir un certain confort dans sa manipulation était incontournable,
- Étant un utilisateur de Windows version Interface Graphique d'Utilisateur (GUI) et non la version CŒUR avec des connaissances très limitées en LINUX et plus limitées encore en ligne de commande (CLI), le temps d'apprentissage et d'adaptation a été long, car Linux est sensible à la casse, ce qui est différent en Windows,
- De nombreux logiciels libres existent et sont disponibles sur Internet. Ils présentent l'avantage d'être libres et gratuits. Vu que différents logiciels peuvent accomplir une même tâche, sélectionner ceux qui doivent être utilisés pour le projet a été difficile. Ensuite, les apprendre et les maîtriser tous a été tout un défi,
- Les logiciels de la pile ELK évoluent à une vitesse importante, en l'espace de quatre mois, trois versions différentes de chaque logiciel sont sorties et qu'il fallait comprendre en parcourant la documentation. Par exemple, en juillet 2015, lors des premiers tests avec Elasticsearch, la version 0.90 était utilisée. De juillet à octobre 2015, les versions suivantes ont vu le jour : 1.3 – 1.4 – 1.5 – 1.6 – 1 .7 – 2.0. C'est

cette dernière qui a été utilisée dans la solution proposée. Un mois plus tard, la version 2.1, qui est en phase bêta, est apparue. Ce qui donne une idée de la vitesse à laquelle les nouvelles versions apparaissent. Et finalement,

- Beaucoup d'effort a été nécessaire afin d'essayer de comprendre et de faire fonctionner les nouvelles versions des composantes de la solution proposée afin de rester à jour avec les nouvelles versions.

3.6 Axes d'amélioration pour une nouvelle itération

Cette première proposition est loin d'être une solution complète, car elle comporte certaines faiblesses et vu que la plateforme DOCKER elle-même est en évolution et que les outils utilisés dans la proposition s'améliorent de jour en jour, il serait souhaitable que, dans une nouvelle itération les six points suivants soient étudiés plus en détail :

1- Tableaux de bord

- Ajouter une vue présentant le nombre de conteneurs au total versus le nombre de conteneurs en exécution,
- Afficher le statut et le nom de chaque conteneur,
- Ajouter des fonctionnalités permettant de catégoriser les conteneurs par service ou par hôte,
- Ajouter une vue de géolocalisation permettant de voir sur une carte la position des serveurs ou centres de données exécutant DOCKER localisés à partir de leur adresse IP publique, et
- Améliorer l'affichage du nombre de conteneurs en exécution afin qu'il affiche uniquement les conteneurs qui sont en train de s'exécuter au moment où l'on visualise le tableau de bord.

2- Système d'alerte

- Intégrer un autre système d'alerte à la solution, qui est beaucoup plus complet que celui de Collectd, et

- Créer une interface permettant de définir les seuils et les paramètres de notification afin d'éviter que des utilisateurs non avisés modifient les fichiers de configuration.

3- Date et Heure auxquelles les données sont collectées

- Créer un script qui vérifie et s'assure que le serveur Collectd est réglé à la bonne heure et la bonne date avant d'enclencher le processus de collection de données.

4- Les composantes

- Dockériser la proposition en vue de la rendre indépendante de toute plateforme,
- Sécuriser l'accès dans Kibana pour visualiser les données, et
- Adapter la solution aux nouvelles versions des composantes qui apparaissent de manière fulgurante.

5- Les métriques

- Le nombre de métriques pouvant être collectées par conteneurs étant très élevé, offrir un choix aux utilisateurs de sélectionner les métriques qu'ils veulent visualiser serait un atout,
- Afficher les adresses IP des conteneurs, et
- Agréger les métriques en vue d'avoir la valeur réelle utilisée.

6- Les services

- Faire la surveillance des services tels que les bases de données (MySQL, Oracle...) et
- Ajouter des moniteurs montrant le statut de chaque service.

CONCLUSION

Ce projet de recherche visait à trouver une solution afin de surveiller les conteneurs DOCKER ainsi que le système hôte sur lequel ils s'exécutent, collecter les métriques et les journaux afin de les analyser, les traiter et les afficher dans un tableau de bord de manière à faciliter la tâche du gestionnaire de système.

Pour y parvenir, une revue littéraire de DOCKER a été effectuée afin de comprendre la plateforme et la problématique posée en ce qui a trait à la surveillance de ses conteneurs. Partant de la problématique, des objectifs ont été définis et une méthodologie a été appliquée, suite à laquelle une solution potentielle a été proposée. Plusieurs logiciels libres ont été évalués et les plus performants ont été retenus. Ainsi pour collecter les métriques et les évènements provenant des diverses sources telles que les conteneurs et le système hôte, *Collectd* a été sélectionné et expérimenté pour faire la collection des données. Le logiciel libre *Elasticsearch* a été sélectionné et expérimenté pour stocker et indexer les données collectées. Pour recueillir les données non structurées de *Collectd* et les rediriger vers *Elasticsearch*, le logiciel libre *logstash* a été sélectionné et expérimenté. En dernier lieu, la plateforme de visualisation libre *Kibana* a été sélectionnée et expérimentée pour représenter graphiquement les données collectées dans un nombre de prototypes de tableaux de bord.

En vue de tester la fonctionnalité et la performance de la solution proposée, son application a été faite à une étude de cas. Elle s'est avérée fonctionnelle et jugée performante se basant sur les tests effectués et sur l'environnement dans lequel ils ont été effectués.

La proposition est loin d'être parfaite et requiert une amélioration sur différents axes et vu que DOCKER ainsi que les composantes de la solution proposée sont en pleine évolution, il va sans dire qu'une veille technologique sera nécessaire afin d'améliorer cette solution.

L'expérience a été plus que fructueuse, car elle fut une occasion d'acquérir plus de connaissances en Linux, d'apprendre à utiliser la plateforme Docker qui est nouvelle et de

faire connaissance avec de nombreux logiciels tels que *vi, vim, collectd, fluentd, logstash, nginx, graphite, apache2, syslog, elasticsearch, kibana*, pour ne citer que ceux-là. Ce fut également une bonne occasion pour mettre en pratique toutes les connaissances théoriques cumulées tout au long du cursus en maîtrise et ainsi en découvrir leur intérêt.

ANNEXE I

INSTALLATION DE DOCKER

Tous les tests sont déroulés dans un environnement LAB, sur un ordinateur portatif ayant les caractéristiques suivantes:

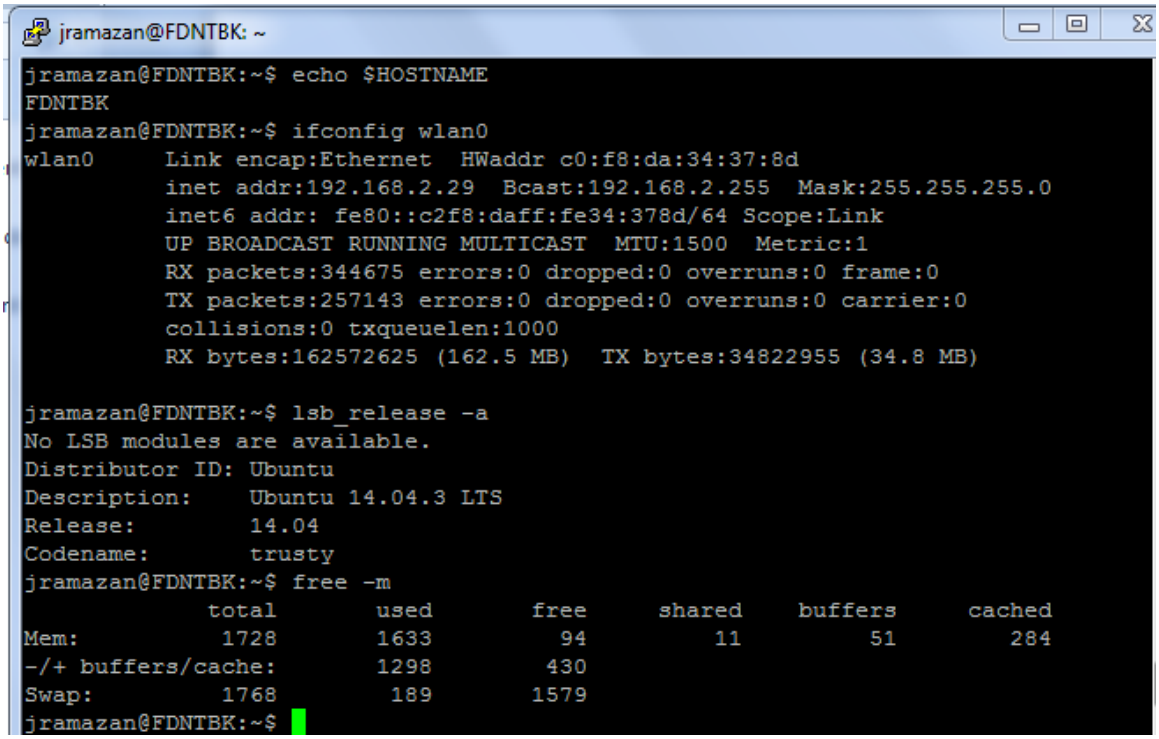

Figure-A I-1 Information sur le système Hôte

 # Installation de Docker: [1]

sudo apt-get update

curl –sSL https://get.docker.com/ | sh

Vérification si Docker est bien installé:

sudo docker run hello-world

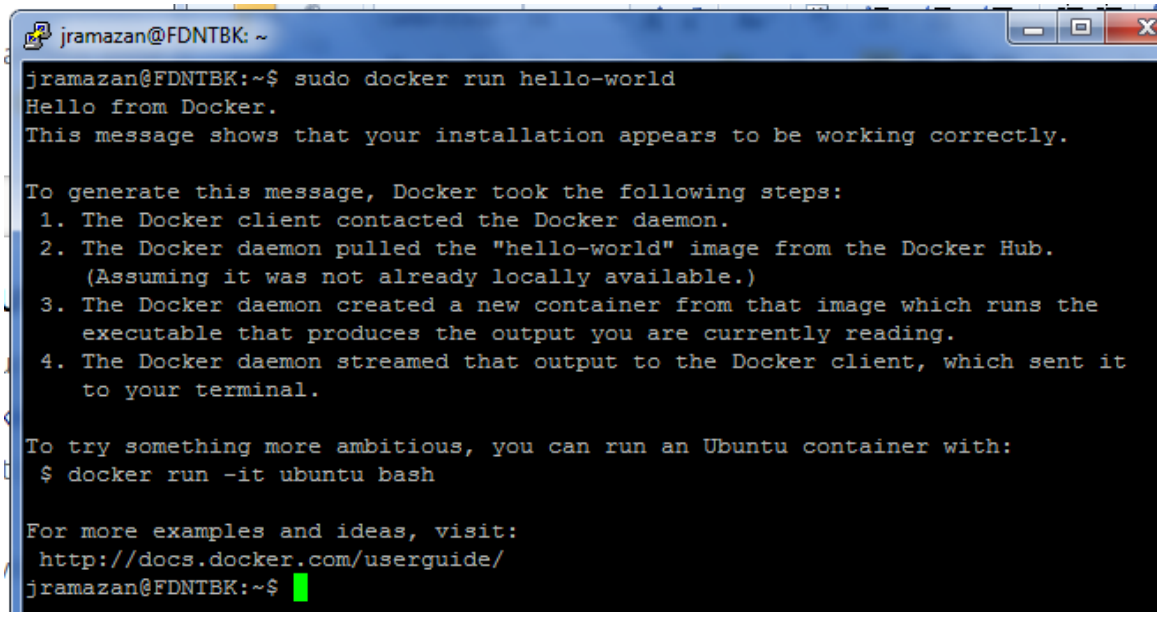

Figure-A I-2 Affichage de Hello Word pour tester si Docker est bien installé

#Vérification de l'emplacement où sont stockés les conteneurs de Docker

sudo docker info

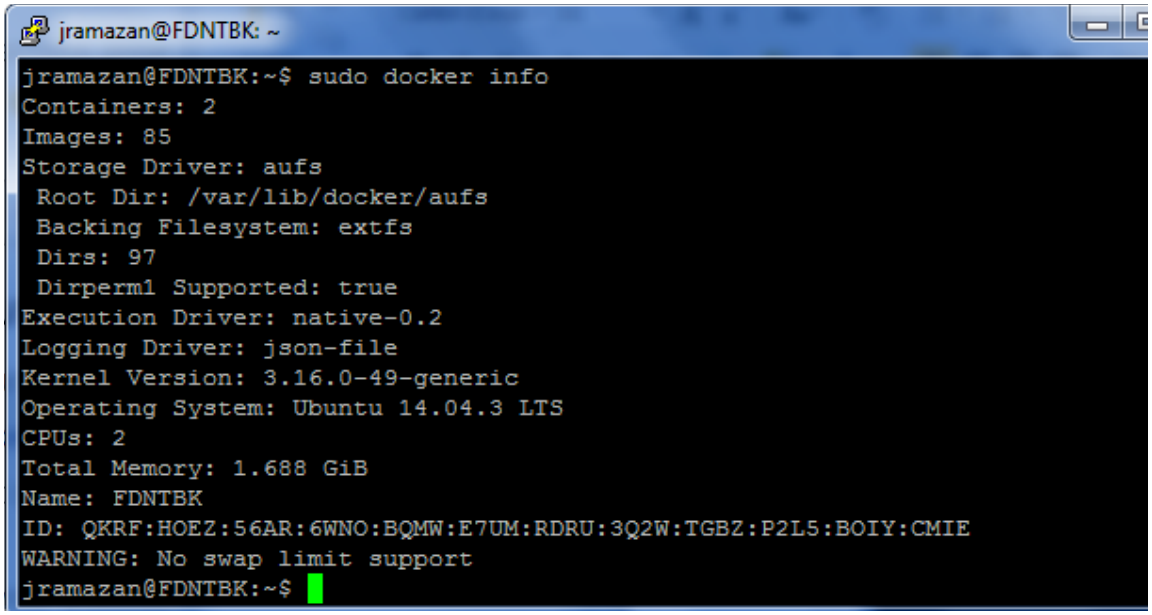

Figure-A I-3 Résultat de docker info

Vérification de la version de docker

sudo docker version # On peut utiliser `docker –v` également

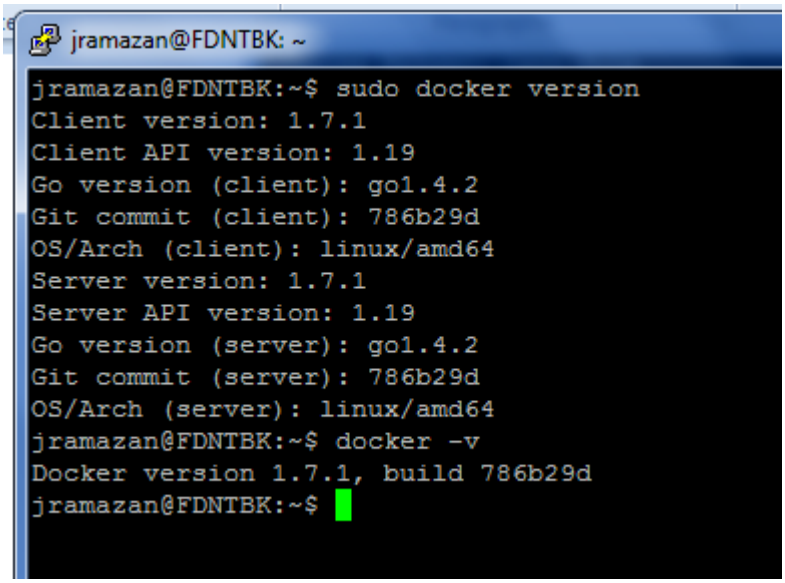

Figure-A I-4 Vérification de la version de DOCKER installé

Note : docker requiert des privilèges de « root »

Pour pouvoir utiliser docker en tant que non-root usager, il faut ajouter l'usager courant dans le groupe de Docker. Pour ce faire :

sudo groupadd docker # ajout du groupe docker

sudo gpasswd –a ${user} docker # Ajout de l'usager courant dans le groupe docker

sudo service docker restart # redémarrage du service docker

ANNEXE II

MANIPULATION DES CONTENEURS DE DOCKER [1]

#Recherche d'une image

sudo docker search _nom_image

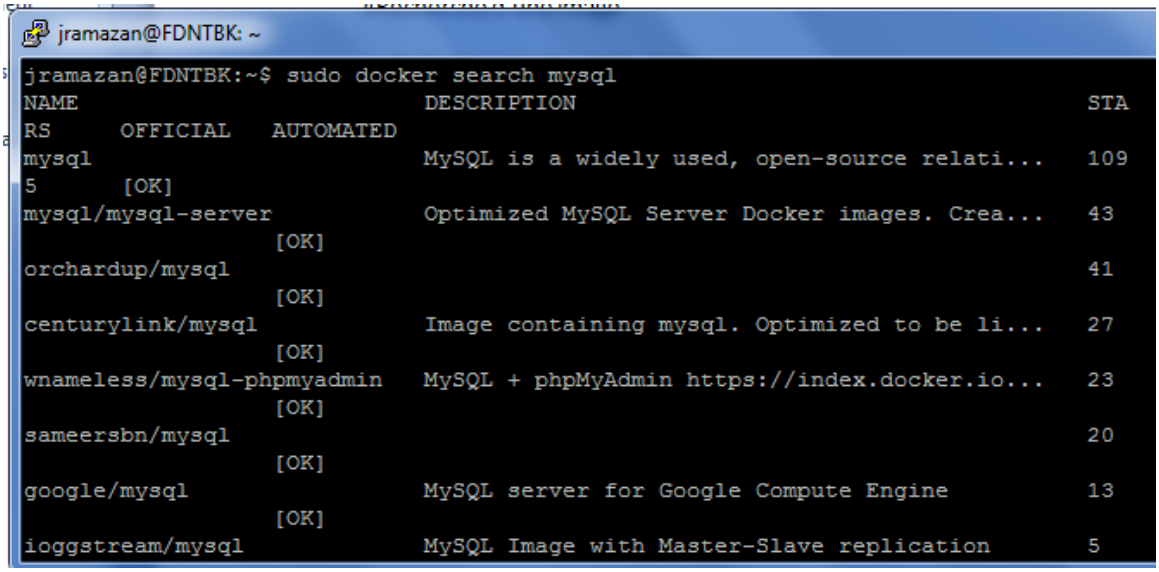

Figure-A II-1 Résultat de la recherche sur l'image mysql

#Téléchargement d'une image

Sudo docker pull nom_image #Les images qui n'ont pas de préfixe sont validées par DOCKER

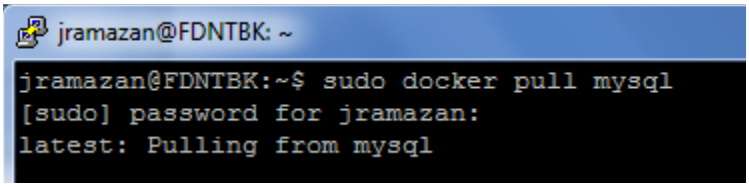

Figure-A II-2 Téléchargement d'une image

Vérification de l'image

sudo docker inspect nom_image

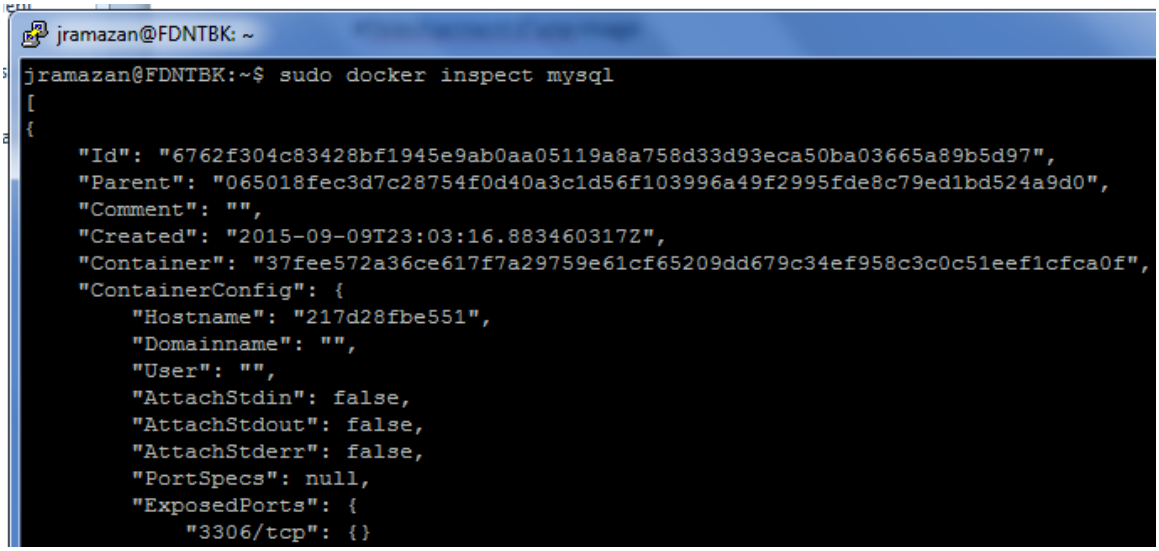

Figure-A II-3 Résultat de la commande docker inspect sur une image.

Pour démarrer un container mysql

#`docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag`

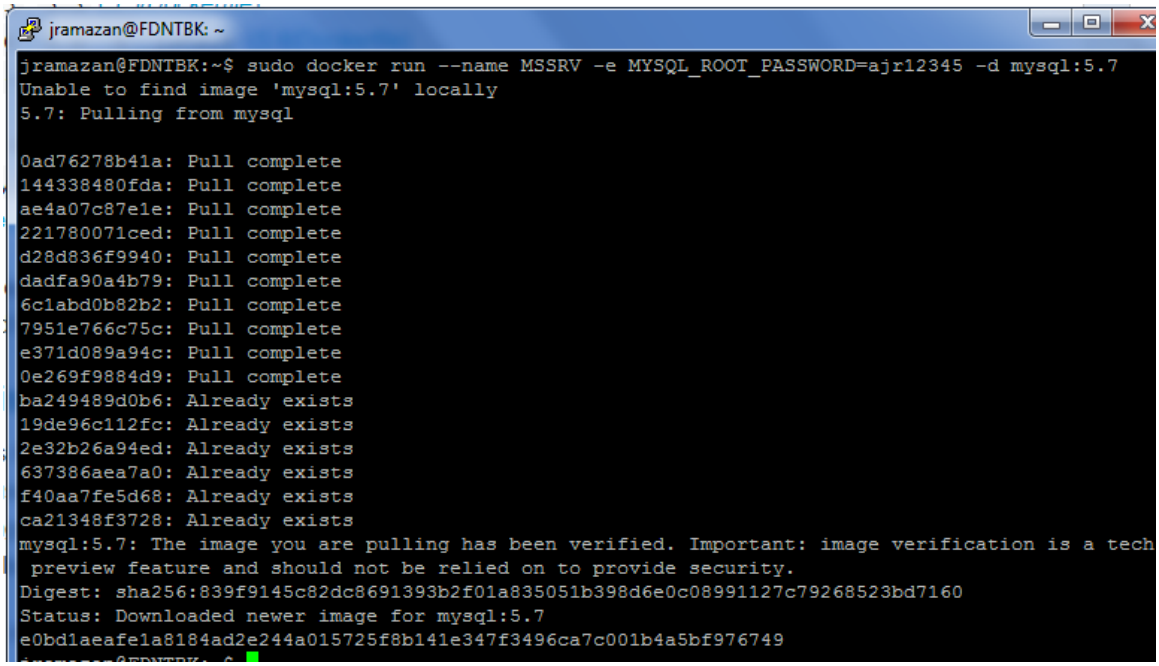

Figure-A II-4 Démarrage d'un conteneur mysql en mode serveur

#To connect to mysql

```
docker run -it --link some-mysql:mysql --rm mysql sh -c 'exec mysql -h"$MYSQL_PORT_3306_TCP_ADDR" -P"$MYSQL_PORT_3306_TCP_PORT" -uroot -p"$MYSQL_ENV_MYSQL_ROOT_PASSWORD"'
```

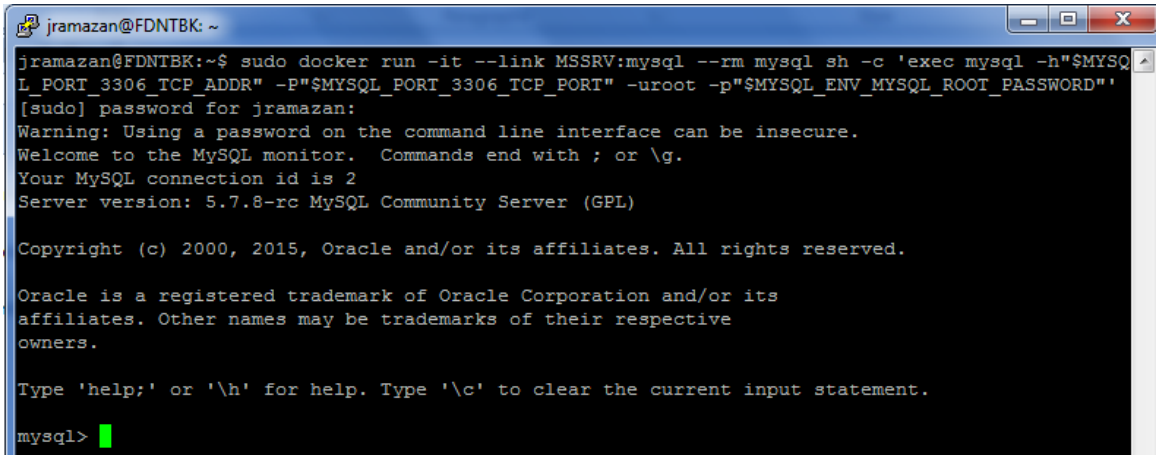

Figure-A II-5 Démarrage d'un conteneur mysql en mode client

Une fois à l'intérieur du conteneur mysql vous pouvez effectuer toutes les opérations d'un SGBD

#Affichage des bases de données

show databases ; #Notez bien les commandes se terminent par un ";"

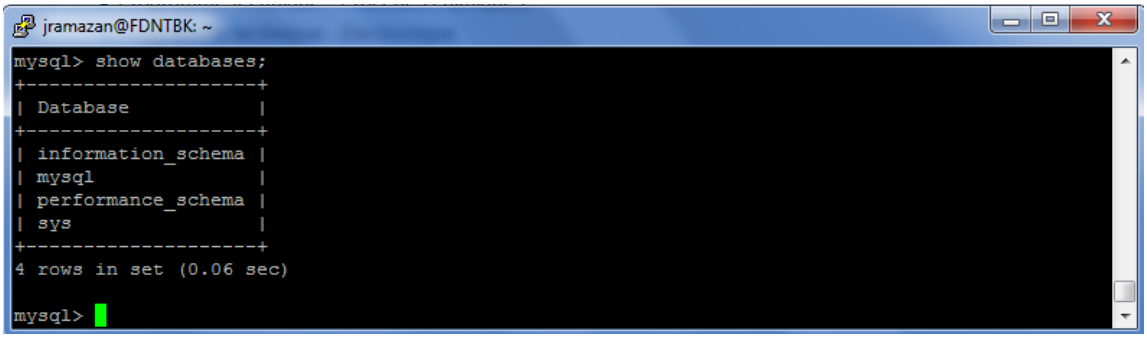

Figure-A II-6 Affichage de la liste des bases de données existantes

Pour lister les containers en exécution

sudo docker ps –l

Pour lister tous les containers même ceux qui sont arrêtés

```
sudo dcoker ps -a
# Pour stopper un conteneur
sudo docker stop container_id
# Pour redémarrer un conteneur
sudo docker start container_id
# Pour supprimer un conteneur
sudo docker rm container_id
```

ANNEXE III

INSTALLATION ET CONFIGURATION DE « COLLECTD »

1.- Téléchargement du fichier compressé de la dernière version de collectd à date, soit collectd

sudo wget https://collectd.org/files/collectd-5.5.0.tar.bz2

2.- Extraction du fichier:

tar xjf ./collectd-5.5.0.tar.bz2

3.- Changement de répertoire

cd collectd-5.5.0

4.- Compilation

./compile

5.- Installation

make install all

6.- Start collectd service

sudo service collectd start

ANNEXE IV

VISUALISATION DES MÉTRIQUES

Pour visualiser les métriques des conteneurs de DOCKER, généralement on utilise la commande *docker stats* suivie du nom du conteneur, pour un conteneur spécifique, ou *docker stats `docker ps –qa`* pour tous les conteneurs. La figure suivante est le résultat de la commande *docker stats `docker ps –qa`*

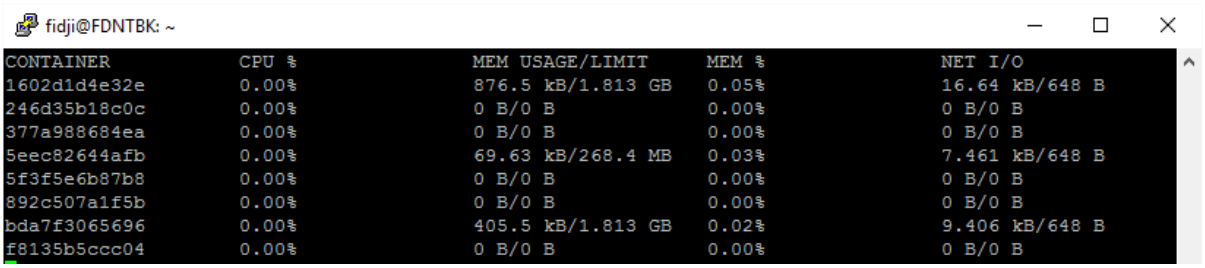

Figure-A IV-1 Résultat de la commande docker stats `docker ps –qa`

La commande top est utilisée pour visualiser la performance du système hôte, soit Ubuntu dans ce cas.

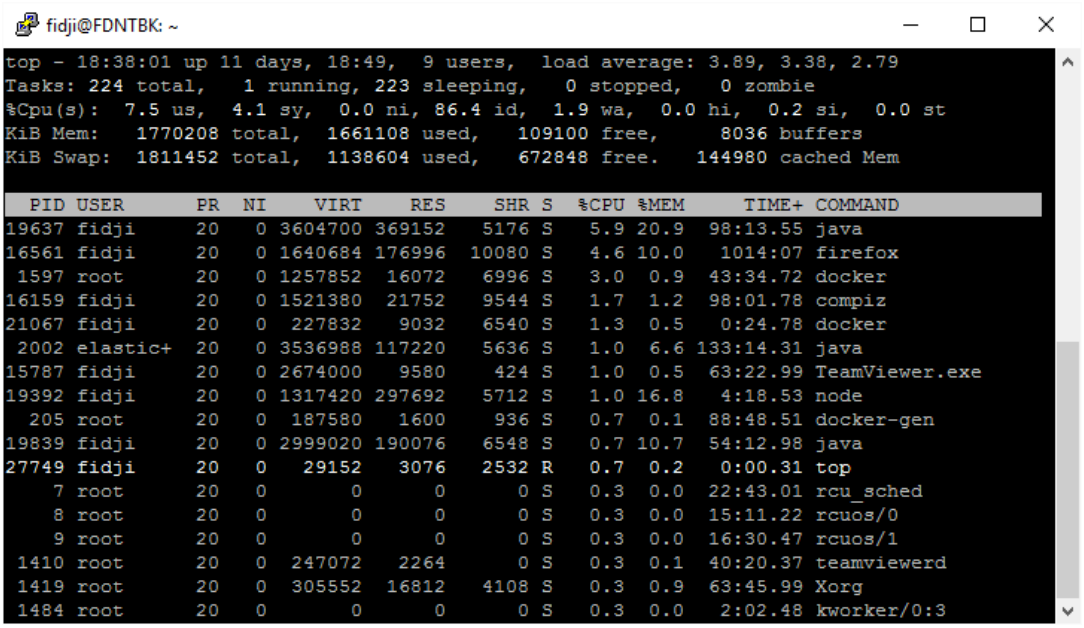

Figure-A IV-2 Utilisation de top pour visualiser les métriques du système

ANNEXE V

VISUALISATION DES JOURNAUX D'UN SYSTÈME

Afin de visualiser les journaux d'un système (c.-à-d. les logs en anglais), les commandes less, tail, cat, head sont utilisées pour afficher tout ou une partie du contenu du fichier syslog qui les contient généralement.

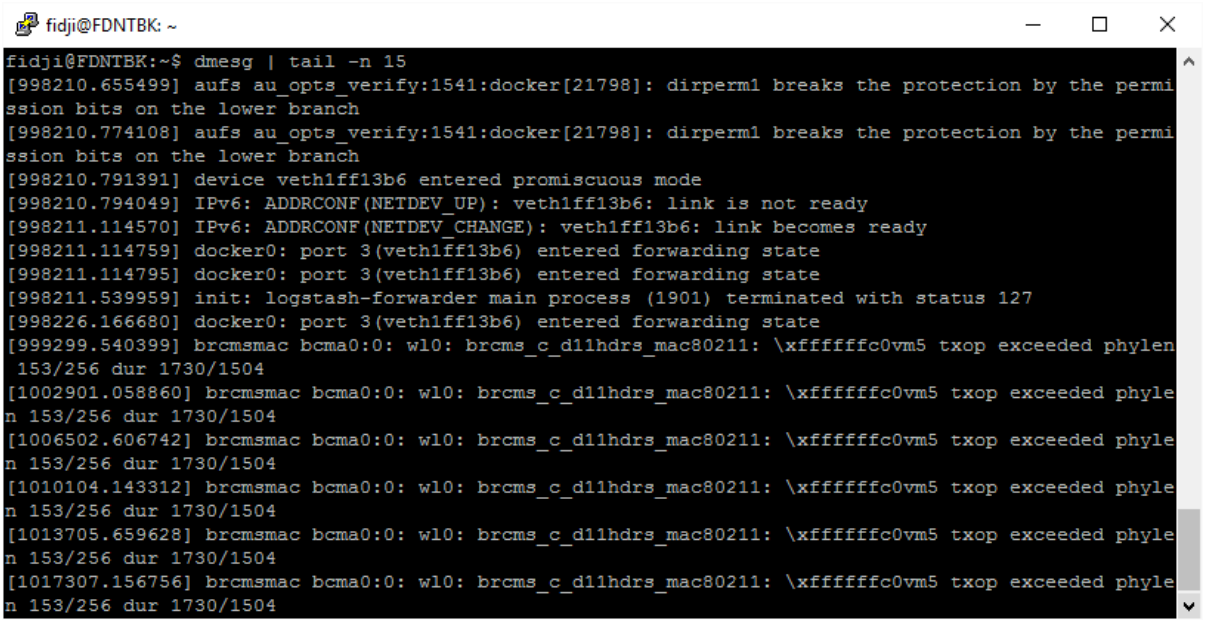

Figure-A V-1 Affichage des 15 dernières lignes du résultat de la commande *dmesg*

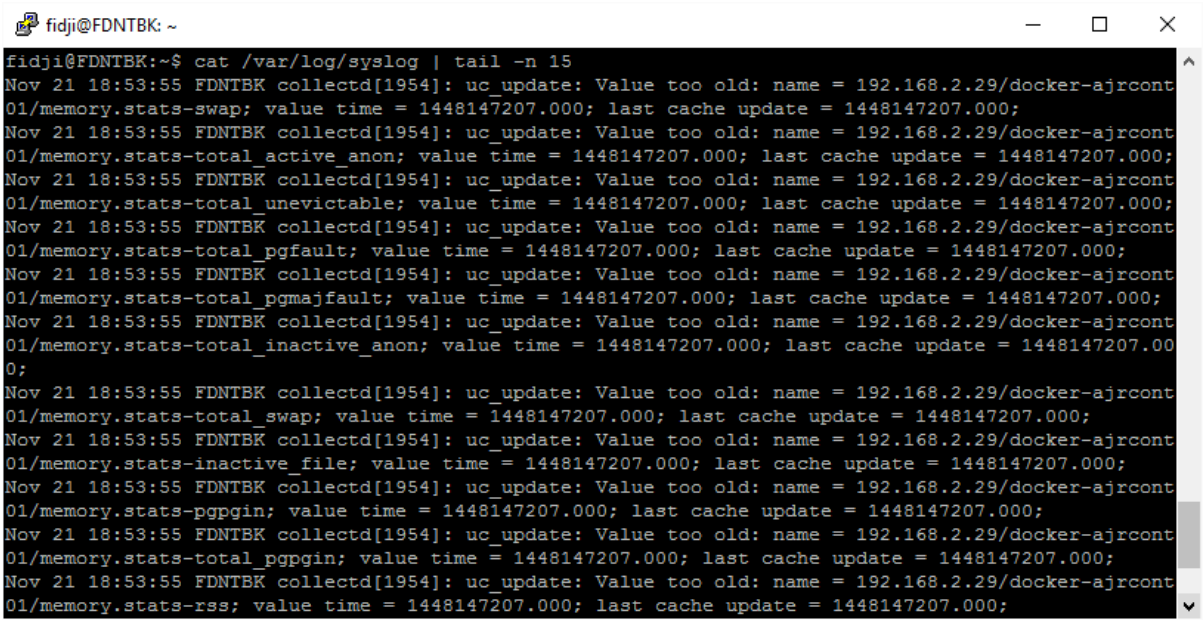

Figure-A V-2 Affichage des 15 dernières lignes du fichier syslog

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Docker.com, « What is docker », [En ligne] Disponible : https://www.docker.com/ [Accès le 20 mai 2015]

[2] Beshawred, Yonas. 2014. «How Docker Was Born». In Posts. En ligne. < http://stackshare.io/posts/how-docker-was-born>. Consulté le 28 septembre 2015

[3] Alain Clapaud. 2015. Docker : Pourquoi la technologie container open source remporte un tel succès. En ligne <http://pro.clubic.com/it-business/actualite-750387-docker-pourquoi-technologie-container-remporte-tel-succes.html >. Consulté le 15 juin 2015.

[4] Charles, Anderson. 2015. «Docker». En ligne <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7093032>. Consulté le 15 juin 2015.

[5] Dua, Rajdeep; Raja, A Reddy; Kakadia, Dharmesh. 2014. «Virtualization vs Containerization to Support PaaS». En ligne <http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6903537>. Consulté le 20 juin 2015.

[6] Antonio, Coradi; Diana J., Arrojo. 2014. «Monitoring applications and services to improve the Cloud Foundry PaaS». En ligne <http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6912627>. Consulté le 20 juin 2015.

[7] Von Eicken, Thorsten. 2014. «Docker vs. VMs? Combining Both for Cloud Portability Nirvana ». En ligne. <http://www.rightscale.com/blog/cloud-management-best-practices/docker-vs-vms-combining-both-cloud-portability-nirvana>. Consulté le 30 septembre 2015.

[8] «HyperForge, Cloud Fondery ». En ligne <https://support.hyperic.com/display/hyperforge/Cloud+Foundry>. Consulté le 30 septembre 2015

[9] «A propos de Nagios». En ligne <http://doc.monitoring-fr.org/3_0/html/about.html>. Consulté le 10 Octobre 2015

[10] «Google/CAdvisor». En ligne <https://hub.docker.com/r/google/cadvisor/>. Consulté le 10 octobre 2015.

[11] « Because Software is your business ». En ligne <http://newrelic.com/platform>. Consulté le 10 octobre 2015.

[12] «Monitor Docker Performance with Datadog». En ligne https://www.datadoghq.com/blog/monitor-docker-datadog/. Consulté le 10 octobre 2015.

[13] Mayr, Alois. 2015. «The black-box Docker monitoring approach». En ligne <https://blog.ruxit.com/how-to-monitor-docker-containers/>. Consulté le 30 juin 2015.

[14] https://sysdig.com/

[15] Paul, Frédéric. 2015. «New relic's docker monitoring now generally available». En ligne <https://dzone.com/articles/new-relic%E2%80%99s-docker-monitoring>. Consulté le 17 novembre 2015.

[16] Lê-Quôc, Alexis. 2014. «Monitor docker performance with datadog». En ligne <https://www.datadoghq.com/blog/monitor-docker-datadog/>. Consulté le 17 novembre 2015.

[17] https://axibase.com/docker-monitoring/

[18] https://www.nagios.org/

[19] Rouse, Margaret. «Container-based virtualization (operating system-level virtualization) definition». En ligne http://searchservervirtualization.techtarget.com/definition/container-based-virtualization-operating-system-level-virtualization. Consulté le 17 novembre 2015.

[20] Davis, David. 2012. «Choosing the right virtualisation, monitoring and management tools». En ligne < http://searchdatacenter.techtarget.com/feature/Choosing-the-right-virtualization-management-tools>. Consulté le 30 juin 2015.

[21] Bryant, Daniel. 2015. «Monitoring Microservices and Containers: A Challenge by Adrian Cockcroft ». En ligne <http://www.infoq.com/news/2015/06/monitoring-microservices>. Consulté le 03 juillet 2015.

[22] Sanitas, Nicolas. 2015. «Docker : la technologie de conteneurisation d'applications à découvrir ». En ligne <http://ictexpertsluxembourg.lu/ict-cloud/docker-technologie-conteneurisation-dapplications-decouvrir/>. Consulté le 06 juillet 2015.

[23] Alain Clapaud. 2015. Docker : Pourquoi la technologie container open source remporte un tel succès. En ligne <http://pro.clubic.com/it-business/actualite-750387-docker-pourquoi-technologie-container-remporte-tel-succes.html >. Consulté le 07 juillet 2015.

[24] Filippone, Dominique. 2015. « Docker lève 95 million de dollars ». En ligne <http://www.lemondeinformatique.fr/actualites/lire-docker-leve-95-millions-de-dollars-60857.html>. Consulté le 25 juillet 2015.

[25] Intercloud. 2015. « Cloud 2.0: IBM et Microsoft rejoignent le projet Docker de google». En ligne <http://www.scoop.it/t/cloudnews/p/4025497037/2014/07/30/cloud-2-0-ibm-et-microsoft-rejoignent-le-projet-docker-de-google>. Consulté le 08 décembre 2015.

[26] « What is Fluentd ». En ligne <http://www.fluentd.org/architecture>. Consulté le 10 octobre 2015.

[27] «Syslog and Windows Event Logs Collector ». En ligne <https://www.manageengine.com/products/eventlog/collect-eventlogs.html>. Consulté le 10 octobre 2015.

[28] Hoover, Dwayne; Beedgen, Christian. 2014. « Six millions ways to log in Docker ». En ligne http://www.slideshare.net/raychaser/6-million-ways-to-log-in-docker-nyc-docker-meetup-12172014>. Consulté le 11 octobre 2015.

[29] « Metric ». En ligne <http://whatis.techtarget.com/definition/metric>. Consulté le 11 octobre 2015.

[30] «About RRD tool ». En ligne < https://collectd.org/>. Consulté le 12 octobre 2015.

[31] «Gestion des logs avec Logstash, ElasticSearch & Kibana ». En ligne <http://linuxfr.org/news/gestion-des-logs-avec-logstash-elasticsearch-kibana>. Consulté le 12 octobre 2015.

[32] «Setting up Elasticsearch, Logstash and Kibana». En ligne <https://www.ddreier.com/setting-up-elasticsearch-kibana-and-logstash/>. Consulté le 13 octobre 2015.

[33] «Linux Log Files Location And How Do I View Logs Files on Linux?». En ligne <http://www.cyberciti.biz/faq/linux-log-files-location-and-how-do-i-view-logs-files/>. Consulté le 21 novembre 2015.

[34] «Docker logging». En ligne <http://www.fluentd.org/guides/recipes/docker-logging>. Consulté le 21 novembre 2015.

[35] «Docker-collectd-plugin». En ligne <https://github.com/lebauce/docker-collectd-plugin>. Consulté le 21 novembre 2015.

[36] «Runtime metrics». En ligne <https://docs.docker.com/engine/articles/runmetrics/>. Consulté le 21 novembre 2015.

[37] «Understand the architecture» [En ligne], Disponible, [Accès le 25 Novembre 2015] https://docs.docker.com/engine/introduction/understanding-docker/

[38] «Docker definition». En ligne] <http://searchenterpriselinux.techtarget.com/definition/Docker>. Consulté le 25 novembre 2015.

[39] «What is docker…». En ligne <http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/>. Consulté le 25 novembre 2015.

[40] Bernstein, David. 2014. «Containers and Cloud: From LXC to Docker to Kubernetes» En ligne <http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7036275>. Consulté le 25 novembre 2015.

[41] «Qui est ce français à la tête d'un des plus grands projets open source du monde». En ligne <http://www.journaldunet.com/solutions/cloud-computing/1167814-sam-alba-ce-francais-a-la-tete-de-l-ingenieurie-d-un-des-plus-grands-projets-open-source/. Consulté le 08 décembre 2015.

[42] Crochet-Damais, Antoine. 2015. «Cloud : Pourquoi Docker peut tout changer?». En ligne <http://www.journaldunet.com/solutions/cloud-computing/docker-definition-avantages-inconvenients.shtml>. Consulté le 25 novembre 2015.

[43] «Docker comme solution de virtualisation : théorie». En ligne <https://www.guillaume-leduc.fr/docker-comme-solution-de-virtualisation-theorie.html>. Consulté le 26 novembre 2015.

[44] https://collectd.org/wiki

[45] Young, K. 2015. «The docker monitoring problem». En ligne <https://www.datadoghq.com/blog/the-docker-monitoring-problem/>. Consulté le 26 novembre 2015.

[46] Young, k. 2015. «How to monitor docker resource metrics. In BLOG». En ligne <https://www.datadoghq.com/blog/how-to-monitor-docker-resource-metrics/>. Consulté le 26 novembre 2015.

[47] https://www.sumologic.com/

[48] https://blog.docker.com/tag/docker-monitoring/

[49] ISO. «Représentation de la date et de l'heure – ISO8601». In ISO. En ligne <http://www.iso.org/iso/fr/home/standards/iso8601.htm>. Consulté le 07 décembre 2015

[50] SOC. «What is libESMTP? ». In MANUAL. En ligne <http://soc.if.usp.br/manual/libesmtp5/README>. Consulté le 08 décembre 2015.