

MISE EN PLACE D'UNE SOLUTION SIEM DANS UN CONTEXTE DE CLOUD PRIVE BASEE SUR LA PILE ELK, SURICATA ET ZABBIX

Table des matières

Introduction générale.....	9
Chapitre 1 : Contexte général du projet.....	11
Introduction	11
1- Problématique.....	14
2- Déroulement	14
4- Présentation des environnements de travail.....	15
4.1- 1er environnement : Réseau WAN	15
4.2- 2ème environnement : Réseau LAN.....	15
5- Objectifs.....	16
Chapitre 2 : Notions fondamentales sur le Cloud Computing	17
Introduction.....	17
1- Définition.....	17
2- Historique.....	18
3- Éléments constitutifs du Cloud Computing.....	19
3.1- La virtualisation.....	19
3.2- Datacenter.....	20
3.3- Plateforme collaborative.....	21
4- Différence fondamentale entre le service Internet et le Cloud Computing.....	22
5- Architecture du Cloud Computing.....	22
5.1- IaaS : Infrastructure as a service.....	22
5.2- PaaS : Platform as a service.....	22
5.3- SaaS : Software as a service.....	23
6- Types de Cloud Computing.....	25
6.1- Les Cloud privés.....	25
6.2- Les Cloud publics.....	25
6.3- Les Cloud hybrides.....	25
7- Avantages et inconvénients du Cloud Computing.....	26
7.1- Cloud Public.....	26
7.1.1- Les avantages.....	26
7.1.2- Les inconvénients.....	26
7.2- Cloud Privé.....	27
7.2.1- Les avantages.....	27
7.2.2- Les inconvénients.....	28
8- La sécurité dans le Cloud Computing.....	29
8.1- Les risques de sécurité majeurs du Cloud Computing.....	29
8.2- Solutions proposées.....	29
9- Acteurs du secteur.....	32
9.1- Cloud public.....	32
9.2- Cloud privé.....	33
10- Les solutions Cloud existantes.....	33

10.1- Solutions propriétaires.....	33
10.1.1- Windows Azure.....	33
10.1.2- Google AppEngine.....	34
10.1.3- La plateforme EC2 d'Amazon.....	34
10.1.4- VMWare vCloud.....	35
10.2- Solutions libres.....	35
10.2.1- Eucalyptus.....	35
10.2.2- OpenNebula.....	37
10.2.3- Xen Cloud Plateform.....	39
10.2.4- Openstack.....	39
10.2.5- OwnCloud.....	40
10.2.6- Cloudstack.....	41
Chapitre 3 : Analyse, présentation et installation de notre solution SIEM.....	42
Introduction.....	42
1- Notions de base.....	42
2- Principe de fonctionnement.....	44
2.1- Elastic stack (Pile ELK).....	44
2.2- IDS Suricata.....	46
2.3- Outil de supervision Zabbix.....	47
3- Procédure d'installation de notre solution SIEM dans un réseau WAN.....	49
3.1- Architecture.....	50
3.2- Désactivation de SELinux.....	50
3.3- Installation de Java.....	51
3.4- Installation et configuration d'Elasticsearch.....	51
3.5- Installation et configuration de Kibana avec Nginx.....	52
3.5.1- Installation et configuration de Kibana.....	52
3.5.2- Installation et configuration de Nginx.....	53
3.6- Installation et configuration de Logstash.....	54
3.7- Installation et configuration de Filebeat sur les deux serveurs Client : miro-node1 et miro-node2.....	57
3.7.1- Serveur miro-node1.....	57
3.7.2- Serveur miro-node2.....	60
3.8- Installation et configuration de Suricata sur les deux serveurs Client : miro-node1 et miro-node2.....	60
3.9- Configuration du PAT (Port Address Translation) sur le routeur lié au serveur alan-elk1 et redirection des ports 5044 et 5045.....	64
3.9.1- Configuration du PAT.....	64
3.9.2- Redirection des ports 5044 et 5045.....	65
3.10- Configuration du PAT (Port Address Translation) sur le routeur lié aux serveurs miro-node1 et miro-node2.....	65
3.10.1- Configuration du PAT.....	65
3.10.2- Redirection du port 49155.....	66
3.10.3- Redirection du port 49156.....	67

4- Procédure d'installation de notre solution SIEM dans un réseau LAN.....	67
4.1- Architecture.....	68
4.2- Désactivation de SELinux.....	68
4.3- Installation de Java.....	69
4.4- Installation et configuration d'Elasticsearch.....	69
4.5- Installation et configuration de Kibana avec Nginx.....	71
4.5.1- Installation et configuration de Kibana.....	71
4.5.2- Installation et configuration de Nginx.....	72
4.6- Installation et configuration de Logstash.....	73
4.7- Installation et configuration de Filebeat sur les deux serveurs Client : client1-srv et client2-srv.....	78
4.7.1- Serveur client1-srv.....	78
4.7.2- Serveur client2-srv.....	80
4.8- Installation et configuration de Suricata sur les deux serveurs Client : client1-srv et client2-srv.....	81
4.9- Installation et configuration de Zabbix sur le serveur : zabbix-srv.....	85
4.9.1- Modification du fichier SELinux.....	85
4.9.2- Installation d'Apache2/httpd.....	85
4.9.3- Installation et configuration de PHP 7.2.....	86
4.9.4- Installation et configuration de MariaDB.....	87
4.9.5- Installation et configuration de Zabbix 4.0.....	88
4.9.6- Configuration de Firewalld.....	91
4.10- Installation et configuration de Zabbix-Agent sur les deux serveurs Client : client1-srv et client2-srv.....	92
4.10.1- Serveur client1-srv.....	92
4.10.2- Serveur client2-srv.....	93
4.11- Installation et configuration de Grafana sur le serveur zabbix-srv.....	93
4.12- Configuration initiale et test de la pile ELK.....	96
4.13- Configuration initiale et test du serveur Zabbix.....	102
4.14- Configuration initiale et test du serveur Grafana.....	107
BIBLIOGRAPHIE ET WEBOGRAPHIE.....	112
Annexe A.....	115

Table des figures

Figure 1 : Image du Cloud Computing.....	17
Figure 2 : Principe de la virtualisation.....	20
Figure 3 : Exemple d'un Datacenter.....	20
Figure 4 : Représentation d'une plateforme collaborative.....	21
Figure 5 : Les trois grands modèles de service du Cloud.....	23
Figure 6 : Répartition des charges en fonction du modèle du Cloud.....	24
Figure 7 : Logo de Microsoft Azure.....	33
Figure 8 : Logo de Google App Engine.....	34
Figure 9 : Logo d'EC2 d'Amazon.....	35
Figure 10 : Logo du vCloud de VMWare.....	35
Figure 11 : Logo d'Eucalyptus.....	36
Figure 12 : Architecture d'Eucalyptus.....	37
Figure 13 : Logo d'OpenNebula.....	37
Figure 14 : Architecture interne d'OpenNebula.....	38
Figure 15 : Logo de Xen Cloud Platform.....	39
Figure 16 : Logo d'Openstack.....	40
Figure 17 : Logo d'OwnCloud.....	40
Figure 18 : Logo de Cloudstack.....	41
Figure 19 : Principe de fonctionnement d'Elastic stack.....	45
Figure 20 : Principe de fonctionnement de Suricata.....	46
Figure 21 : Modes de checks actifs et passifs.....	47
Figure 22 : Architecture du système de supervision Zabbix.....	48
Figure 23 : Architecture du réseau du déploiement.....	50
Figure 24 : Architecture du réseau du déploiement.....	68
Figure 25 : Droits d'accès.....	74

Introduction générale

Au fur et à mesure que les systèmes informatiques évoluent, la demande en quantité d'espace de stockage, de convivialité et de simplicité dans le travail va grandissant. Il y a quelques années, les espaces de stockage réduits, les lignes de commandes et les systèmes complexes étaient le quotidien des employés d'entreprise.

Les entreprises modernes traitent de grandes quantités d'informations aussi nombreuses que variées. Ainsi, elles ont besoin de grande capacité de stockage ainsi que d'une puissance de calcul élevée. Les ressources matérielles et logicielles nécessaires n'étant pas à la portée de toutes les entreprises, le Cloud Computing est une solution pour résoudre ce problème.

Dans le souci d'accroître leur performance et leur compétitivité, plusieurs entreprises modernes ont recours à l'utilisation du Cloud Computing.

Le terme Cloud Computing, ou « informatique dans les nuages », est un nouveau modèle informatique qui consiste à proposer les services informatiques sous forme de services à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui. Cette nouvelle technologie permet à des entreprises d'externaliser le stockage de leurs données et de leur fournir une puissance de calcul supplémentaire pour le traitement de grosse quantité d'information.

Avec l'adoption croissante de cette technologie, les problèmes de sécurité sont devenus une réelle préoccupation pour les entreprises. En effet, le manque de contrôle physique ou de points d'entrée et de sortie bien définis pose toute une série de questions autour de la sécurité des données dans le Cloud. Voici quelques exemples de thématiques : l'exposition et la fuite de données sensibles, les contrôles d'accès faibles, les interruptions de la disponibilité, mélange de données, la vulnérabilité aux cyberattaques, etc.

Afin de combler ce besoin imminent de sécurité, plusieurs solutions ont été mise en place pour détecter les intrusions ainsi que pour sécuriser les charges de travail dans le cloud. C'est dans ce cadre que s'inscrit ce projet intitulé « Mise en place d'une solution SIEM dans un contexte de 'Cloud Privé' basée sur la pile ELK, Suricata et Zabbix ». Au cours de ce projet, on s'intéressera à l'étude et à la mise en œuvre des logiciels libres open source pour assurer la sécurité d'un cloud privé.

Le présent rapport est organisé de la façon suivante : nous présenterons dans le premier chapitre le contexte, les environnements du travail et les objectifs du projet. Le deuxième chapitre introduit les notions fondamentales du Cloud Computing, la technologie qui le constitue, les différents acteurs du secteur ainsi que les différentes solutions Cloud existante. Dans le troisième chapitre, nous allons commencer par

l'étude des solutions libres (Open source) que nous allons installer ainsi que leur principe de fonctionnement. Ensuite, nous présenterons la procédure d'installation de notre solution SIEM dans les deux environnements : « réseau WAN et LAN » et procéderons à la configuration de la pile ELK ainsi que les serveurs Zabbix et Grafana.

Chapitre 1 : Contexte général du projet

Introduction

Dans ce chapitre nous présenterons en premier lieu la problématique rencontrée et la solution proposée. En second lieu, nous présenterons le déroulement des solutions, les différents environnements de travail et les objectifs visés.

1- Problématique

Le monde interconnecté dans lequel nous vivons, repose sur la parfaite accessibilité des données, n'importe où, à tout moment, et sur tout type d'appareil. La rapidité et l'agilité qu'offrent les services et applications hébergés dans le cloud, sont par ailleurs essentielles à sa réussite. À ce titre, leurs avantages intrinsèques ont poussé de nombreuses structures à migrer certaines ou l'ensemble de leurs applications ou infrastructures vers le cloud.

Bien que le cloud offre des avantages considérables, les entreprises ont conscience des problématiques de sécurité qu'il présente. Ces dernières concernent la protection, la conformité et le fonctionnement général, notamment la capacité à déployer des solutions pour sécuriser des charges de travail dans le cloud.

Dans ce cadre, notre projet vise le côté « sécurité », par la mise en place d'une solution SIEM à l'aide des outils open source.

2- Déroulement

La solution à finaliser est composée de plusieurs modules.

Mon travail consiste à développer une plateforme d'analyse de fichiers logs (Elastic stack) dans un réseau WAN, l'installation de l'IDS "Suricata" ainsi que la configuration des composants réseaux liés au serveur de contrôle (Elastic stack) et aux serveurs hébergeant des machines virtuelles constituant un cloud privé.

Aussi, la mise en place de cette plateforme dans un réseau LAN, l'installation de l'IDS "Suricata", l'installation de l'outil de supervision "Zabbix" et du logiciel de la visualisation "Grafana".

Enfin, la configuration initiale et test d'Elastic stack (pile ELK), du serveur Zabbix et du serveur Grafana.

4- Présentation des environnements de travail

Nous allons travailler dans deux environnements :

4.1- 1^{er} environnement : Réseau WAN

Le parc informatique dans lequel nous allons travailler dans cet environnement est composé de trois serveurs et deux routeurs :

- Serveur ELK : alan-elk1.
- Serveurs Client : miro-node1 et miro-node2.
- Routeur lié au serveur alan-elk1.
- Routeur lié aux serveurs miro-node1 et miro-node2.

Les serveurs fonctionnent avec des systèmes d'exploitation différents :

- Système d'exploitation installé sur le serveur alan-elk1 : CentOS 7.
- Système d'exploitation installé sur les serveurs miro-node1 et miro-node2 : Ubuntu.

Chacun des deux serveurs « miro-node1 et miro-node2 » héberge des machines virtuelles.

4.2- 2^{ème} environnement : Réseau LAN

Le parc informatique dans lequel nous allons travailler dans cet environnement est composé de cinq serveurs :

- Serveur ELK : elk-srv.
- Serveurs Client : client1-srv et client2-srv.
- Serveur Zabbix : zabbix-srv.
- Serveur Grafana : zabbix-srv.

Les serveurs mentionnés ci-dessus fonctionnent avec le système d'exploitation : Centos 7.

5- Objectifs

L'objectif principal de notre projet est la mise en place d'une solution SIEM pour assurer les besoins fonctionnels suivants :

- _ Surveiller les fichiers logs.
- _ Surveiller le fichier d'événements collectés par Suricata.

- Surveiller le fichier d'événements collectés par Zabbix.
- Surveiller l'état physique des machines.
- Surveiller la charge des machines : nombre d'utilisateurs connectés, l'usage de la CPU, le débit réseau, etc.
- Surveiller les performances du réseau : débit, latence, taux d'erreur...
- Surveiller les modifications et l'intégrité des fichiers critiques.

Chapitre 2 : Notions fondamentales sur le Cloud Computing

Introduction

Dans ce chapitre, nous allons présenter les notions fondamentales du Cloud Computing, la technologie qui le constitue, ses types, ainsi que les risques de sécurité majeurs de ce dernier, les solutions proposées, les différents acteurs du secteur et les solutions Cloud existantes.

1- Définition

Le Cloud Computing, littéralement l'informatique dans les nuages est un concept qui consiste à déporter sur des serveurs distants des stockages et des traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur. Il consiste à proposer des services informatiques sous forme de service à la demande, accessible de n'importe où, n'importe quand et par n'importe qui. Cette définition est loin d'être simple à comprendre, toutefois, l'idée principale à retenir est que le Cloud n'est pas un ensemble de technologies, mais un modèle de fourniture, de gestion et de consommation de services et de ressources informatiques.

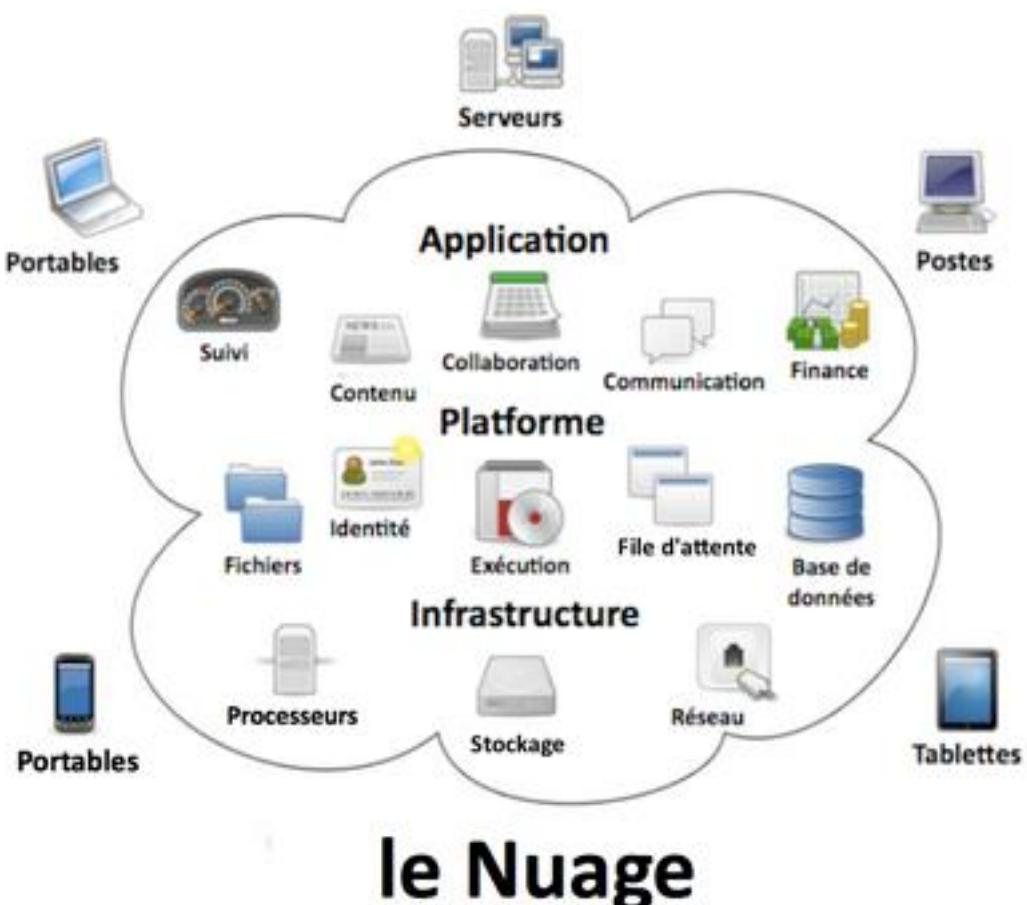


Figure 1 : Image du Cloud Computing

Les applications et les données ne se trouvent plus sur l'ordinateur local, mais « Dans le Cloud » composé d'un certain nombre de serveurs distants, interconnectés au moyen d'une excellente bande passante, indispensable à la fluidité du système. L'accès au service se fait par une application standard facilement disponible, la plupart du temps un navigateur web.

Actuellement, l'utilisation du Cloud Computing s'est démocratisée avec l'accès à des connexions internet dites « haut débit » et à la vulgarisation des ordinateurs de grande capacité, et ayant une grande puissance de traitement. Il peut être considéré comme la cinquième évolution de l'informatique après le mainframe [1], le PC, le client serveur, le web et pour finir le Cloud Computing d'après Microsoft.

Le Cloud Computing peut être comparé à la distribution de l'énergie électrique. Bien que chaque particulier pourrait produire sa propre énergie électrique (solaire, éolienne), un opérateur le fait à grande échelle et propose ce service au client à travers un réseau de distribution (câble électrique). Chaque client est facturé uniquement en fonction de sa consommation. On retrouve dans cet exemple des analogies dans la terminologie du Cloud Computing et de la distribution de l'énergie électrique :

- Énergie électrique services informatiques.
- Réseau de distribution Réseau informatique.

[1] mainframe : un ordinateur central ou un macroordinateur (*mainframe computer*), est un ordinateur de grande puissance de traitement qui sert d'unité centrale à un réseau de terminaux.

2- Historique

La première utilisation de l'expression « Cloud Computing » remonte à 1997, lorsque Ramnath Chellappa, professeur en systèmes d'information et en management, l'a utilisée pour décrire un nouveau modèle de gestion de l'informatique, dans lequel les limites ne seraient plus définies par des problématiques techniques mais par des choix économiques. L'intérêt de ce modèle réside notamment dans la transition d'une partie des coûts informatiques du CapEx vers l'OpEx [2].

En 2002, Amazon lance le premier service clairement estampillé « Cloud Computing ». Le leader de la vente de livres en ligne avait en effet investi dans un parc de serveurs largement surdimensionné, capable d'absorber les pics de charge des commandes des fêtes de fin d'année. Ce parc de serveurs étant sous-utilisé le reste de l'année, le cybercommerçant a alors eu l'idée de louer de la capacité de calcul inutilisée à d'autres entreprises afin de rentabiliser son investissement.

Le terme Cloud Computing sera rendu populaire en 2006 par le directeur exécutif de Google, Éric Schmidt, qui qualifiait ainsi le déport vers « le nuage Internet » de données et d'applications qui auparavant étaient situées sur les serveurs et ordinateurs des sociétés, des organisations ou des particuliers.

[2] Le CapEx, ou Capital Expenditure et l'OpEx, ou Operational Expenditure désignent respectivement les coûts d'investissement et les coûts opérationnels. Traditionnellement, la mise en place de systèmes informatiques se traduit par des coûts d'investissement élevés (achats de serveur, achats de logiciels, développements informatiques spécifiques...).

3- Éléments constitutifs du Cloud Computing

3.1- La virtualisation

La virtualisation consiste à faire fonctionner un ou plusieurs systèmes d'exploitation sur un ou plusieurs ordinateurs. Cela peut sembler étrange d'installer deux systèmes d'exploitation sur une machine conçue pour en accueillir qu'un, mais comme nous le verrons par la suite, cette technique a de nombreux avantages.

Il est courant pour des entreprises de posséder de nombreux serveurs, tels que les serveurs de mail, de nom de domaine, de stockage pour ne citer que ceux-ci. Dans un contexte économique où il est important de rentabiliser tous les investissements, acheter plusieurs machines physiques pour héberger plusieurs serveurs n'est pas judicieux. De plus, une machine fonctionnant à 15 pour cent ne consomme pas plus d'énergie qu'une machine fonctionnant à 90 pour cent. Ainsi, regrouper ces serveurs sur une même machine peut donc s'avérer rentable si leurs pointes de charge ne coïncident pas systématiquement.

Enfin, la virtualisation des serveurs permet une bien plus grande modularité dans la répartition des charges et la reconfiguration des serveurs en cas d'évolution ou de défaillance momentanée.

Les intérêts de la virtualisation sont multiples. On peut citer :

- L'utilisation optimale des ressources d'un parc de machines (répartition des machines virtuelles sur les machines physiques en fonction des charges respectives).
- L'économie sur le matériel (consommation électrique, entretien physique, surveillance)
- L'installation, tests, développements sans endommager le système hôte

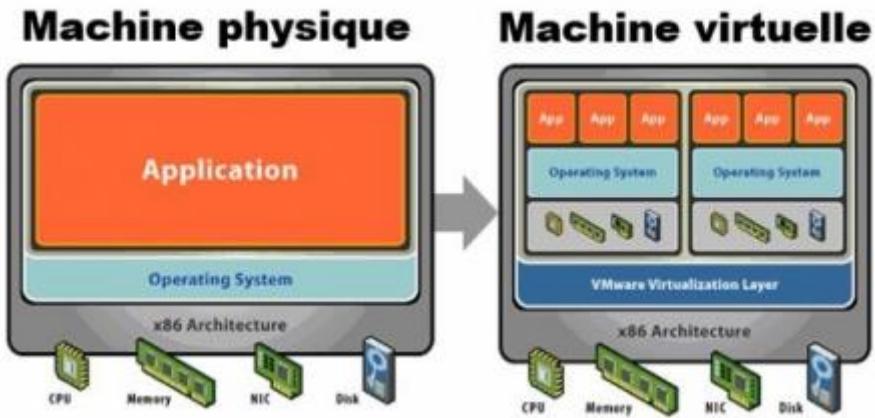


Figure 2 : Principe de la virtualisation

3.2- Datacenter

Un centre de traitement de données (datacenter en anglais) est un site physique sur lequel se trouvent regroupés des équipements constituants du système d'information de l'entreprise (mainframes, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.). Il peut être interne et/ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc.), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée.

Cette infrastructure peut être propre à une entreprise et utilisé par elle seule ou à des fins commerciales. Ainsi, des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies.

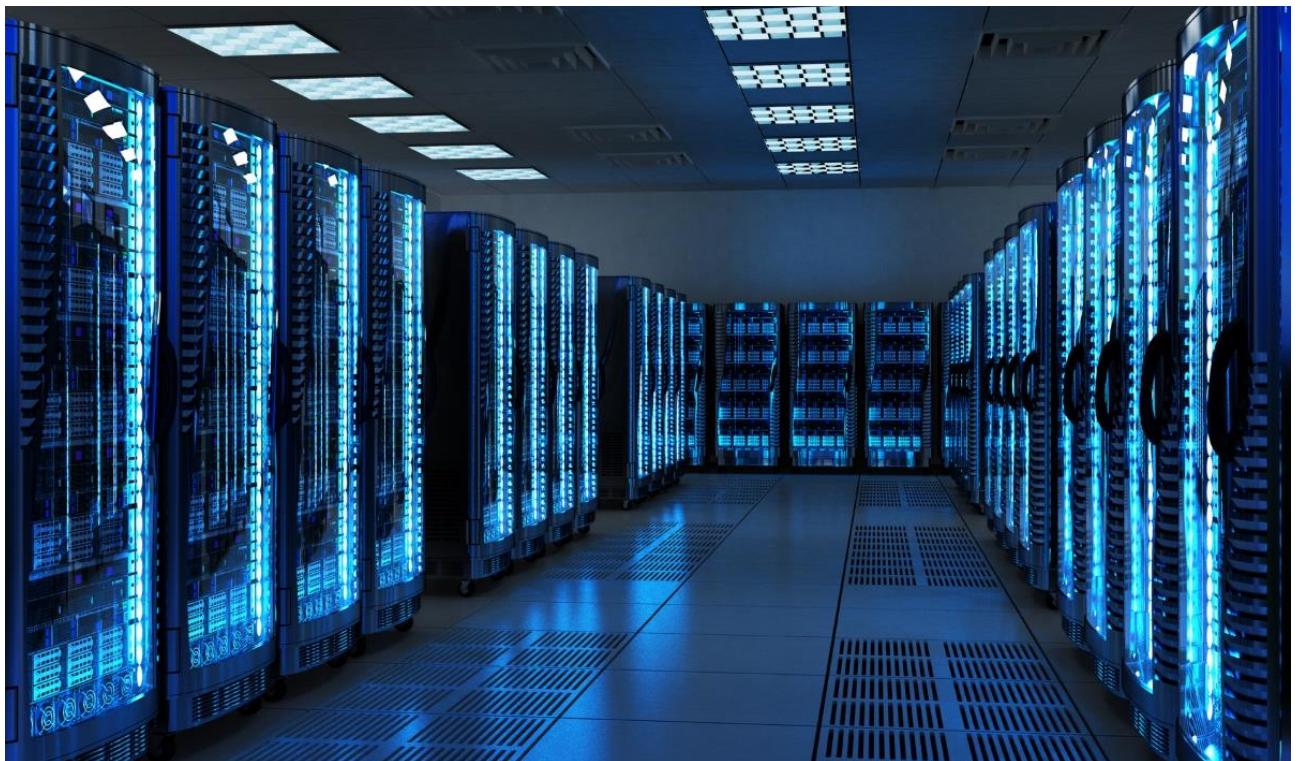


Figure 3 : Exemple d'un Datacenter

3.3- Plateforme collaborative

Une plate-forme de travail collaboratif est un espace de travail virtuel. C'est un site qui centralise tous les outils liés à la conduite d'un projet et les met à disposition des acteurs.

L'objectif du travail collaboratif est de faciliter et d'optimiser la communication entre les individus dans le cadre du travail ou d'une tâche. Les plates-formes collaboratives intègrent généralement les éléments suivants :

- Des outils informatiques.
- Des guides ou méthodes de travail en groupe, pour améliorer la communication, la production, la coordination.
- Un service de messagerie.
- Un système de partage de ressources et de fichiers.
- Des outils de type forum, pages de discussions.
- Un trombinoscope, ou annuaire des profils des utilisateurs.
- Des groupes, par projet ou par thématique.
- Un calendrier.



Figure 4 : Représentation d'une plateforme collaborative

4- Différence fondamentale entre le service Internet et le Cloud Computing

Bien qu'assez similaire dans leur représentation et leur façon de fonctionner, les services Internet et les services Cloud Computing sont assez différents.

Internet est bien plus ancien que le Cloud Computing. C'est grâce à Internet que le développement du Cloud Computing a été aussi rapide. En effet, on peut parler d'internet sans Cloud Computing, mais on ne peut pas parler de Cloud Computing sans Internet. Internet est le moyen le plus utilisé par les fournisseurs de solution de Cloud Computing pour proposer leur service. Mais, internet peut être dans certains cas substitué à tout autre réseau (intranet, réseau téléphonique) comme nous le verrons dans la suite.

5- Architecture du Cloud Computing

Une des notions principales quand on parle de Cloud Computing est le principe « virtualisation ». La virtualisation a été la première pierre vers l'ère du Cloud Computing. En effet, cette notion permet une gestion optimisée des ressources matérielles dans le but de pouvoir y exécuter plusieurs systèmes « virtuels » sur une seule ressource physique et fournir une couche supplémentaire d'abstraction du matériel.

Basiquement, le Cloud Computing propose trois modèles principaux :

- L'infrastructure (Iaas : Infrastructure as a service).
- La plate-forme (Paas : Platform as a service).
- L'application (Saas : Software as a service).

5.1- Iaas : Infrastructure as a service

Infrastructure as a service ou l'infrastructure en tant que service en français est une des couches du Cloud Computing. C'est un modèle où l'entreprise dispose d'une infrastructure informatique (serveurs, stockage, réseau) qui se trouve en fait chez le fournisseur. Cependant, elle y a accès sans restriction, comme si le matériel se trouvait dans ses locaux. Ceci permet à l'entreprise de s'affranchir complètement de l'achat et de la gestion du matériel. L'entreprise exploite le matériel comme un service à distance. Cette couche permet à l'entreprise de se concentrer en premier sur ses processus métiers sans se préoccuper du matériel.

5.2- Paas : Platform as a service

Platform as a service ou plate-forme en tant que service est un modèle composé de tous les éléments nécessaires pour soutenir la construction, la livraison, le

déploiement et le cycle de vie complet des applications et des services disponibles sur Internet. Cette plateforme offre des facilités à gérer, des canevas de travail lors du design, du développement, du test, du déploiement et de l'hébergement d'applications web à travers des outils et services tels que :

- La collaboration d'équipe
- La gestion de la sécurité, de la capacité
- La gestion des bases de données

Ces services sont fournis au travers d'une solution complète destinée aux développeurs et disponible via Internet. Exemple :

<https://www.salesforce.com/fr/?ir=1>

5.3- SaaS : Software as a service

Software as a service ou encore application en tant que service en français est le modèle le plus utilisé dans le monde après le service d'email. C'est un modèle de déploiement d'application dans lequel un fournisseur loue une application clé en main à ses clients en tant que service à la demande au lieu de leur facturer la licence du logiciel. De cette façon, l'utilisateur final n'a plus besoin d'installer le logiciel, le maintenir, ou le mettre à jour. Toutes ces opérations de maintenance sont effectuées par le fournisseur de service. Exemple : Google docs <https://docs.google.com>

Les trois modèles du Cloud Computing peuvent être résumés dans cette illustration :

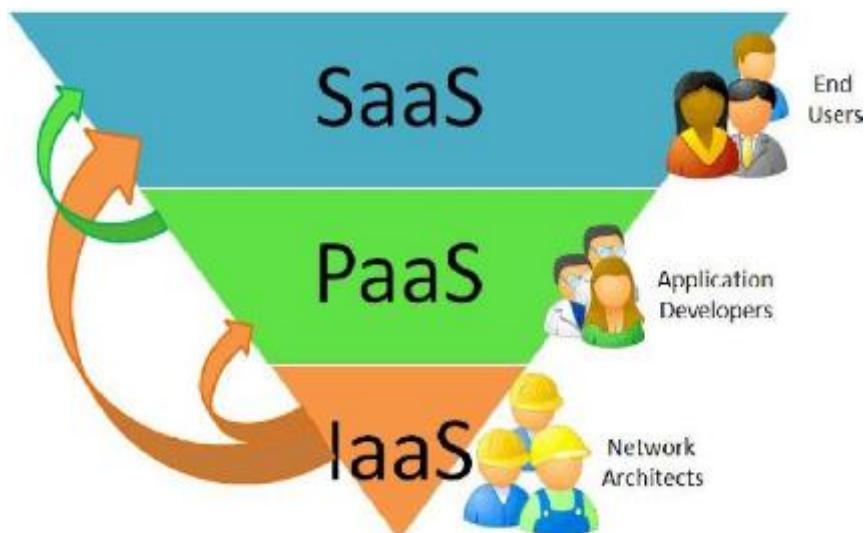


Figure 5 : Les trois grands modèles de service du Cloud

L'Infrastructure en tant que service offre une base matérielle (hardware) aux plateformes en tant que service. Ces infrastructures sont mises en place et gérées par des administrateurs réseau avec un bon niveau d'expertise. Elles sont le plus souvent

constituées d'équipements réseaux et de serveurs la plupart du temps entièrement virtualisés.

La plateforme en tant que service est un ensemble de composants reposant sur l'infrastructure offerte par la couche Iaas. Elle permet aux développeurs d'applications d'avoir une plateforme de travail adaptable, distribuée et virtualisée dans laquelle ils n'ont plus besoin d'installer l'architecture sous-jacente (réseaux, matériels, serveur, système d'exploitation).

L'application en tant que service est une application souple et déployée dans une plateforme en tant que service. C'est une application souple qui est accessible uniquement à travers un réseau et qui est le plus souvent facturée à l'utilisateur final. Cette organisation permet de séparer les domaines de compétences, ainsi l'utilisation d'une couche est complètement non assujettie aux couches inférieures ou supérieures.

Nous pouvons résumer avec cette figure qui explique qui contrôle quoi en fonction du modèle utilisé :

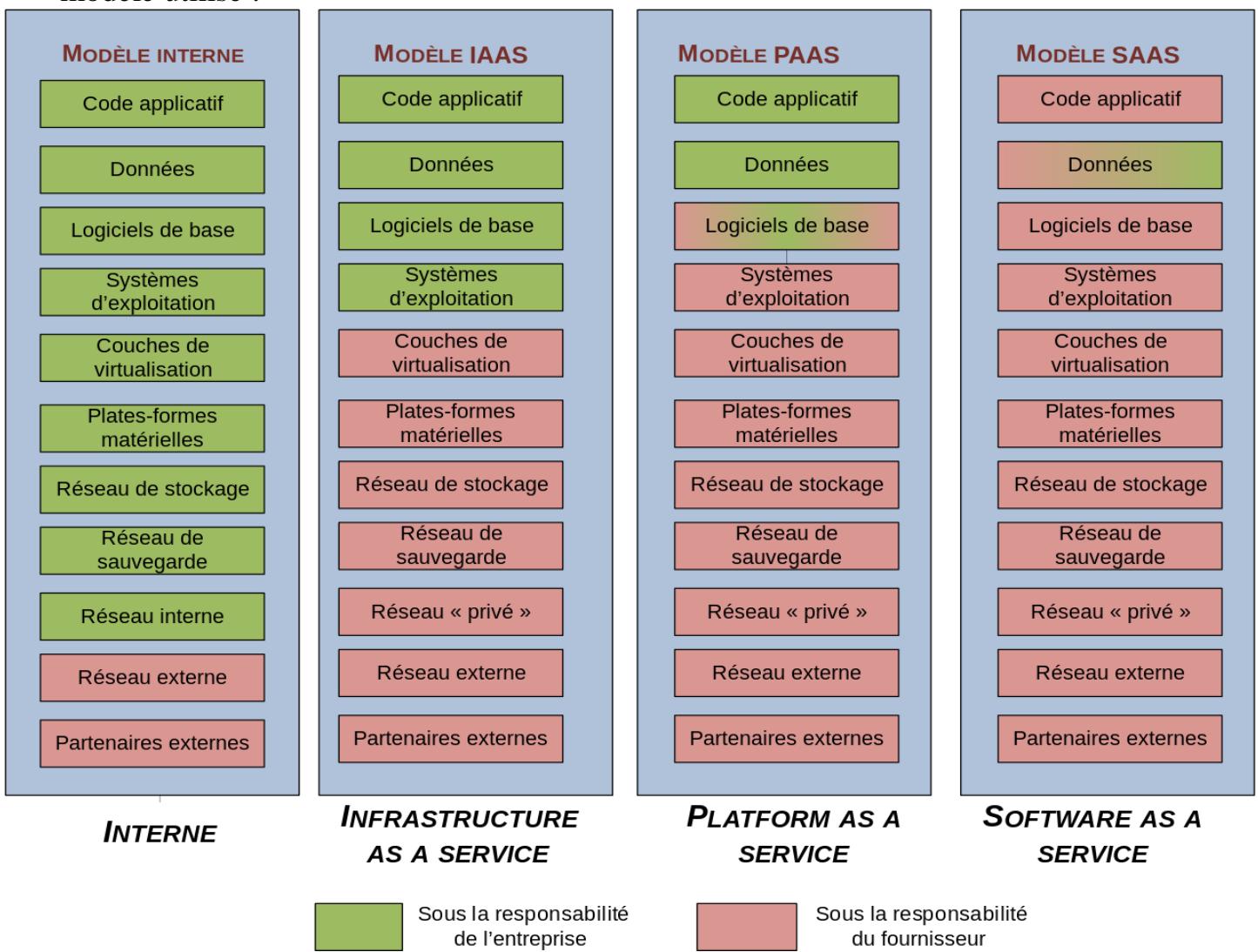


Figure 6 : Répartition des charges en fonction du modèle de Cloud

6- Types de Cloud Computing

Il existe trois types de Cloud Computing :

6.1- Les Cloud privés

Ce type de Cloud est créé et géré en interne pour les besoins d'une entreprise. Les serveurs hébergeant les services sont localisés dans les bâtiments de l'entreprise accessible à travers un réseau sécurisé, interne et fermé. Il est sous l'administration de la direction du service informatique. Dans ce cas, l'entreprise met sur pied sa propre politique de gestion de son Cloud.

Le Cloud privé est une nouvelle approche qui améliore l'organisation et la gestion des services et de ressources informatiques, mais aussi la façon dont l'entreprise les consomme et les met à l'œuvre.

Les ressources informatiques ne remplissent plus des rôles prédéfinis et limités, elles forment un pool flexible dans lequel les processus et le personnel de l'entreprise viennent puiser en fonction des besoins et au moment précis où ils en ont besoin.

6.2- Les Cloud publics

Ce type de Cloud est créé par un organisme spécialisé qui met à disposition ses infrastructures, ses ressources pour des entreprises aussi multiples que variées. Ainsi, ces entreprises consommatrices de services utilisent et payent à la demande des services dont elles ont besoin. Les fournisseurs de services Cloud Computing garantissent une disponibilité et une qualité de service à travers un contrat signé avec le consommateur du service.

Le Cloud public se compose d'une palette de services tiers accessibles via Internet qui ne cesse de s'élargir. Les services étant proposés par des géants comme Google, l'ampleur de l'offre est énorme et le coût unitaire, potentiellement très faible. Une petite entreprise peut externaliser son infrastructure technologique tout entière et la majorité de ses applications métiers dans le Cloud public, et toute jeune entreprise peut faire largement appel au Cloud.

6.3- Les Cloud hybrides

Ce type de Cloud combine les Cloud public et privé. On peut ainsi déporter nos applications vers un Cloud public qui consommera des données stockées et exposées dans un Cloud privé, ou bien faire communiquer deux applications hébergées dans deux Cloud privés distincts, ou encore consommer plusieurs services hébergés dans des Cloud publics différents.

7- Avantages et inconvénients du Cloud Computing

7.1- Cloud Public

7.1.1- Les avantages

- La simplicité : Une entreprise peut créer facilement ses « instances » en quelques minutes, et par exemple, disposer d'une « machine virtuelle » préchargée (OS et logiciels de base de données). Ce type de service réduit la complexité et les délais de mise en œuvre, imputables aux tests et au déploiement de nouvelles applications.

Les clients ne se préoccupent pas non plus de la maintenance du système contrairement au Cloud Privé.

- La réduction des coûts : Selon Sig Nag, Directeur de recherche chez Gartner, les entreprises réalisent 14 % d'économies sur leur budget en adoptant ce type d'hébergement.

A la différence du Cloud Privé, les entreprises n'ont en effet pas besoin d'acheter de matériel physique puisque les serveurs sont virtuels et hébergés par un tiers. Les ressources étant mutualisées, elles peuvent les consommer « As A Service ».

Le paiement du service est généralement mensuel ou annuel et selon l'usage (« pay-as-you-go »). Les entreprises peuvent personnaliser leur service en choisissant le nombre d'utilisateurs et les applications auxquelles ils peuvent accéder.

- Un SI plus souple et agile : Un hébergement dit public permet de s'adapter facilement à des pics de charges. Selon ses besoins, le client peut ajouter ou supprimer des ressources (Processeur, Mémoire, Disque, etc....) ou ajuster le nombre d'instances au grès des besoins. Ce qui permet d'apporter plus de liberté aux équipes de développement dans la gestion des ressources nécessaires aux tests.

7.1.2- Les inconvénients

- La sécurité informatique : La protection et la confidentialité des données hébergées par des fournisseurs cloud restent les deux principales craintes des entreprises.

Pourtant, Gartner affirme que « les services de Clouds publics offerts par les principaux fournisseurs sont sécurisés. La vraie difficulté est de les utiliser de manière sécurisée ».

Cette nuance est essentielle : les entreprises doivent en effet adopter des bonnes pratiques (chiffrement des informations, gestion des accès...) en matière de cybersécurité.

- Un usage qui peut devenir onéreux : Le mode « pay-as-you-go » est pratique, mais il peut s'avérer onéreux si celui-ci n'est pas maîtrisé en interne. Laissé en libre-service sans contrôle des coûts, il peut s'avérer exorbitant.

- La reprise de l'existant : Dans le cas de l'utilisation d'instances ou de machines virtuelles, ce type d'hébergement n'est pas adapté à tous les environnements, notamment les systèmes existants, pour plusieurs raisons :

+ Certaines applications ne supportent pas la virtualisation et nécessitent d'être sur des serveurs physiques dédiés.

+ Les problématiques de Licences logicielles : Certaines licences ne sont pas réutilisables. L'entreprise doit alors racheter des licences ou migrer vers une nouvelle version.

+ La nécessité de maîtriser la performance : Certaines applications peuvent être très gourmandes en ressources et ont besoin d'avoir des performances qui ne peuvent être garanties que par des serveurs dédiés.

7.2- Cloud Privé

7.2.1- Les avantages

- Du sur-mesure pour répondre à des contraintes spécifiques : Même si les offres de Cloud public proposent une large palette de services, le Cloud privé est mieux adapté à certaines contraintes métiers. Cet environnement sur-mesure permet de prendre en compte toutes sortes de contraintes, comme par exemple :

+ Le système de licencing imposé par certains éditeurs : Même si c'est de moins en moins le cas, certains éditeurs ne disposent pas de modèle de licencing adapté à des environnements de Cloud Public ou ne permettent pas la réutilisation de licences existantes sur le Cloud public. Pour éviter un nouvel investissement, l'entreprise fait donc le choix du Cloud Privé.

+ La reprise des environnements existants : Certaines applications ne sont pas éligibles pour tourner sur du Cloud Public et restent souvent indispensables. L'entreprise a donc la possibilité de maintenir ces environnements grâce au Cloud Privé.

- **Un contrôle total sur la sécurité des données** : La configuration du matériel, des unités de stockage et du réseau est définie de façon à répondre précisément aux cahiers des charges des responsables de la sécurité informatique et aux contraintes de conformité de son secteur d'activité (santé, énergie...). Parmi les préconisations, il y a notamment l'isolement des informations et de l'infrastructure sous-jacente. Enfin, l'entreprise garde un contrôle total de ses données.
- **Des performances plus élevées** : Le côté sur-mesure du Cloud Privé permet à l'entreprise de choisir spécifiquement le matériel sur lequel va reposer son infrastructure et d'aller chercher un niveau élevé de performance pour répondre aux contraintes métiers.

7.2.2- Les inconvénients

- **Des investissements conséquents** : Bien que la virtualisation soit la clé de la mise en œuvre d'un Cloud privé car elle permet d'économiser sur les coûts en utilisant l'infrastructure matérielle existante, l'entreprise sera amenée à investir dans des ressources (matérielles et logicielles). Pour gagner en agilité et répondre aux demandes des utilisateurs internes, elle devra mettre en œuvre des solutions de Self-Service et de Provisionning automatique qui représentent souvent de lourds investissements.

L'entreprise a aussi la possibilité de rationaliser ses coûts en transformant ses investissements CAPEX (voir [2] 2- Historique) en dépenses d'exploitations OPEX.

- **Un service informatique très sollicité** : L'entreprise doit relever un défi majeur : être capable de prévoir comment tous les composants vont pouvoir s'assembler. Un plan d'évolution bien pensé s'avère donc indispensable. Par ailleurs, le service informatique devra superviser cette infrastructure : maintenance énergétique, climatisation, mise à jour des logiciels et des systèmes d'exploitation, déploiement et configuration de solutions de cybersécurité...

Ces tâches peuvent être déléguées à un fournisseur de service afin de permettre au service informatique de se concentrer sur son cœur de métier et ses utilisateurs.

- **Des ressources (potentiellement) non extensibles** : Des limitations du matériel physique et un manque de place pourraient entraîner un plafonnement des capacités. Dans ce cas, l'entreprise pourrait s'appuyer sur un Cloud hybride afin de gagner en scalabilité et de consommer des ressources à l'usage sur des environnements de Cloud Public.

8- La sécurité dans le Cloud Computing

8.1- Les risques de sécurité majeurs du Cloud Computing

La sécurité et la conformité émergent systématiquement comme les principales préoccupations des responsables informatiques lorsqu'il est question de Cloud Computing, des préoccupations encore plus accentuées lorsqu'il s'agit de Cloud public. La sécurité permet de garantir la confidentialité, l'intégrité, l'authenticité et la disponibilité des informations.

Certaines questions légitimes reviennent sans cesse :

- Mes données sont-elles sûres dans le cloud ?
- Qui aura accès ?
- Aurais-je accès à mes données à n'importe quel moment ?
- Qu'arrive-t-il en cas de rupture de contrat ?
- Où sont stockées mes données ?

La mise sur pied d'une solution de Cloud Computing comporte des problèmes de sécurité inhérents à la solution elle-même. Le fait de centraliser toutes les informations sur un site pose un grand nombre de problèmes. On peut citer comme problème potentiel :

- Vulnérabilités des systèmes
- Attaques de l'intérieur
- Une possible interruption massive du service.
- Une cible de choix pour les hackers.
- Interface et API non sécurisé.

Ces points de vulnérabilité du Cloud Computing font depuis quelques années l'objet de recherches avancées. Il a été créé un organisme chargé de mettre sur pied des normes en matière de sécurité dans le Cloud Computing. Cet organisme s'appelle CSA (Cloud Security Alliance) [3].

[3] Cloud Security Alliance (CSA) est une organisation à but non lucratif dont la mission est de « promouvoir l'utilisation des meilleures pratiques pour fournir une assurance de sécurité dans le Cloud Computing et de fournir une formation sur les utilisations du Cloud Computing.

8.2- Solutions proposées

1- Authentification et Autorisation :

OAuth2 : OAuth est un protocole libre (Actuellement de version 2.0) qui permet d'autoriser une application client à utiliser l'API sécurisée d'une autre application pour le compte d'un utilisateur. L'intérêt majeur d'OAuth vient du fait que l'utilisateur n'a plus besoin de fournir ses informations d'identification à une application tierce car la connexion se passe sur l'application de l'API. Cela suppose que l'utilisateur lui a à priori fait confiance.

Role-Based Access Control (RBAC) : Fait référence à la méthode permettant de mettre en place la gestion des droits d'accès dans une entreprise. Avec cette méthode, les droits d'accès ne sont pas attribués au cas par cas en fonction d'un individu, mais sur la base de rôles RBAC. Ces rôles sont ensuite à leur tour définis en fonction du service, du poste, de l'emplacement et du centre de coûts de l'employé dans l'entreprise. Les rôles RBAC sont en général consignés et enregistrés dans une matrice RBAC.

MiLAMob : Est un framework de middleware centré qui simplifie le processus d'authentification en temps réel. Le middleware utilise la technique OAuth 2.0 (par exemple Facebook, Google+ et personnels Login) pour identifier l'utilisateur final et utilise des jetons de sécurité pour gérer l'authentification fastidieuse avec Amazon S3 pour le compte de l'utilisateur/demandeur.

FermiCloud : Utilise une autre approche pour l'authentification et l'autorisation, basée sur le PKI (public key infrastructure) X.509. La PKI, ou Public Key Infrastructure ou encore Infrastructure à clé publique permet de sécuriser de façon globale l'accès à un réseau, à des informations et données. La PKI est utilisée dans les domaines suivants : E-mail, e-commerce, VPNs (réseau privé virtuels), Extranets. Elle permet d'assurer de bout en bout la sécurisation des accès et de transferts de données. Plusieurs éléments entrent en jeu dans ce système, notamment les serveurs de certificats et d'authentification, les signatures électroniques, le cryptage du trafic internet.

2- Identité et gestion d'accès :

Shibboleth : Est un logiciel Open Source, qui établit une relation de confiance entre une entité bien définie et autre identifiée. Il assure un accès filtré et sécurisé aux ressources de ces entités limité uniquement aux membres de ces mêmes entités. Shibboleth, dans sa version 2, se place ainsi comme le logiciel Open Source de référence de fédération d'identité qui supporte la norme SAML2.0.

SAML (Security Assertion Markup Language) est un standard définissant un protocole pour échanger des informations liées à la sécurité, à la fédération et à la délégation d'identités. Basé sur le langage XML, SAML a été développé par OASIS

(The Organization for the Advancement of Structured Information Standards, un consortium mondial qui travaille pour la standardisation de formats de fichiers ouverts basés notamment sur XML). Le problème le plus important que SAML tente de résoudre est celui de l'authentification unique SSO (Single Sign-On) sur le web. Il s'agit de permettre à un utilisateur de naviguer sur plusieurs sites différents en s'authentifiant une seule fois, sans pour autant que ces sites aient accès à des informations trop confidentielles.

IBHMCC (identity-based hierarchical model for cloud computing) : Dans le Cloud Computing, il est fréquent que les entités communiquent mutuellement entre eux. Pour parvenir à la sécurité dans la communication, il est important de proposer un cryptage et régime de signature. Par conséquent, en proposant un chiffrement basé sur l'identité (IBE) et signature identitaire (IBS) pour les régimes IBHMCC. Dans le cadre identitaire, tel que proposé par Shamir en 1984, la clé publique d'un utilisateur est tout simplement son identité, ce qui simplifie les exigences de PKI.

IBE (Identity-Based Encryption) : permet à un expéditeur de chiffrer un message sans accès à un certificat de clé publique. Cette méthode a de nombreuses applications pratiques. Par exemple, un utilisateur peut envoyer un courrier chiffré à un destinataire, sans exiger une existence d'une infrastructure à clé publique (PKI).

3- Confidentialité, Intégrité et disponibilité :

Swap and Play : Les hyperviseurs fournissent les moyens d'exécuter plusieurs machines virtuelles isolées sur le même hôte physique. En règle générale, la mise à jour des hyperviseurs nécessite un redémarrage de l'hôte conduisant à l'interruption des services qui est hautement indésirable, en particulier dans les environnements de Cloud Computing. Néanmoins, les mises à jour de sécurité doivent être appliquées rapidement pour réduire le risque d'attaques, exigeant une solution qui élimine le compromis entre la disponibilité et les risques de sécurité. La mise à jour en direct, en général, est très difficile et a été étudié pendant des décennies. Cependant, toutes les solutions proposées à ce jour nécessitent une modification du flux de contrôle du logiciel et/ou la cause de la dégradation des performances. De plus, actuellement, il n'y a pas de solutions pour la mise à jour en direct des hyperviseurs et tous les principaux produits (par exemple, Hyper-V, Xen, ESXi) nécessitent un redémarrage pour la mise à jour. Dans, les auteurs proposent Swap and Play, le premier mécanisme de la mise à jour en direct pour les hyperviseurs, cette solution est facile à utiliser, évolutive et, en particulier, déployable dans des environnements de Cloud Computing.

3- Surveillance de la sécurité et de réponse aux incidents.

4- Gestion politique de sécurité :

A4cloud : a pour but de développer les solutions pour assurer la responsabilité et la transparence dans les environnements du Cloud Computing. Les utilisateurs doivent avoir la possibilité de suivre leur utilisation de données pour savoir comment le fournisseur de Cloud répond à leurs attentes en matière de protection des données. A cet effet, les fournisseurs de Cloud doivent employer des solutions qui fournissent aux utilisateurs le contrôle et la transparence appropriées sur leurs données.

PMaaS (Project Management as-a-Service) est défini comme les capacités fournies aux clients de gérer les politiques d'accès aux services et produits en cours d'exécution sur une infrastructure cloud qui est accessible via des interfaces.

5- Chiffrement des données dans le Cloud.

6- RéPLICATION DES DONNÉES À TRAVERS PLUSIEURS DATACENTERS : Elle permet une récupération facile en cas de désastre.

7- L'isolation des machines virtuelles.

Conclusion

- La sécurité absolue n'existe pas, donc le problème de sécurité reste le plus souvent un problème de confiance entre le fournisseur de service et le consommateur de service. Cette confiance se traduit par la signature d'un contrat nommé SLA (Service Level Agreement) [4]. Ce contrat précise les taux de disponibilité du service. En règle générale, et pour la plupart des fournisseurs, ce taux est supérieur à 99%.

[4] Service-level agreement (SLA) ou « entente de niveau de service » est un document qui définit la qualité de service, prestation prescrite entre un fournisseur de service et un client. Autrement dit, il s'agit de clauses basées sur un contrat définissant les objectifs précis attendus et le niveau de service que souhaite obtenir un client de la part du prestataire et fixe les responsabilités.

9- Acteurs du secteur

9.1- Cloud public

Les entreprises qui fournissent une solution Cloud public sont celles qui disposent d'assez de ressources financière et techniques pour mettre sur pied d'énormes fermes de serveurs, des logiciels hyper puissants et une couverture mondiale. Les entreprises mondiales qui offrent des solutions de Cloud Computing public sont :

- Amazon Web Services.
- Microsoft.

- Google Cloud Platform.
- Alibaba.
- IBM.
- Vmware.
- OVH.
- Dell Technologies et HPE.
- Cisco.
- Oracle.
- Salesforce.
- SAP.
- Workday.

9.2- Cloud privé

Toute entreprise publique ou privée peut mettre sur pied son propre Cloud. Ceci à des fins purement internes. Pour cela, il faudrait que l'activité de l'entreprise s'y prête et qu'elle dispose du matériel adéquat, d'une expertise dans ce domaine. Plusieurs entreprises se tournent vers cette solution car, elles restent propriétaires de leurs données, et contrôlent le processus du début à la fin.

10- Les solutions Cloud existantes

Dans le domaine du Cloud Computing, plusieurs acteurs sont impliqués : les fournisseurs d'offres publiques et ceux proposant le système sous forme de logiciels pouvant être employés en privé. Les solutions du cloud sont ainsi classées en deux grandes catégories : les solutions propriétaires et les solutions open source.

10.1- Solutions propriétaires

Il existe à ce jour une multitude d'offre de logiciel pour installer son propre Cloud privé.



Figure 7 : Logo de Microsoft Azure

10.1.1- Windows Azure

Azure est une plateforme de Microsoft pour les services PaaS du cloud computing. Il s'agit d'une plateforme de développement d'applications fournissant les services d'exécution et d'administration d'applications en offrant les outils nécessaires. Elle permet aux développeurs de programmer et de stocker directement leurs applications sur Internet en leur allouant dynamiquement des machines virtuelles de son centre de données (datacenter). Windows Azure est une plateforme flexible qui supporte plusieurs langages de programmations tels que .Net, C#, Java, PHP, Python, etc. De plus, elle supporte les standards et protocoles tels que SOAP, XML, REST.

L'infrastructure soutenant la plateforme Azure est basée sur la solution de virtualisation Xen.

10.1.2- Google AppEngine

AppEngine est une offre de Google pour les services de type PaaS. Le développement et le déploiement d'applications sur la plateforme de Google sont rendus possibles grâce à un SDK conçu par Google et mis à la disposition des utilisateurs afin de leur permettre de développer en local pour ensuite déployer l'application vers l'Internet. L'idée est de permettre aux utilisateurs d'employer l'infrastructure de Google pour héberger leurs applications avec la possibilité de définir le groupe d'utilisateurs de cette dernière. Ces applications bénéficient de la haute disponibilité des infrastructures de Google.



Figure 8 : Logo de Google App Engine

10.1.3- La plateforme EC2 d'Amazon

Les services d'Amazon EC2 (Elastic Compute Cloud) concernent l'exposition de machines virtuelles pour les activités telles que l'hébergement, les grilles de calcul ou les tests en réseaux informatiques.

L'utilisation des services d'Amazon est facturée selon le temps d'utilisation des machines louées.



Figure 9 : Logo d'EC2 d'Amazon

10.1.4- VMWare vCloud

vCloud est un projet de Cloud Computing mené par VMWare. Il a pour but de permettre à ses clients de migrer leur travail, à leur demande, à partir de leur stockage interne des hyperviseurs VMware vers un stockage à distance (des hyperviseurs VMware également). Le but du projet est de fournir la puissance du cloud computing avec la flexibilité permise par la virtualisation. Le projet a été annoncé à la conférence 2008 de VMworld à Las Vegas et a retenu l'attention médiatique importante.



Figure 10 : Logo du vCloud de VMWare

10.2- Solutions libres

Face à des solutions payantes et propriétaires, il existe des solutions libres et gratuites pour créer son Cloud privé.

10.2.1- Eucalyptus

Eucalyptus est un outil open source issu d'un projet de recherche de l'université de Californie. Il est développé en C, Java, Python et est disponible sous deux licences. Une licence GPL gratuite supportant les hyperviseurs Xen et KVM et une licence commerciale offrant des fonctionnalités avancées telles que le support de VMware. Il permet de construire aussi bien les solutions privées du cloud computing que les solutions publiques. Son grand avantage est qu'il est intégré dans les distributions Ubuntu et Debian. Eucalyptus offre des interfaces compatibles avec les services EC2 d'Amazon. Ce qui lui confère la possibilité d'être employé pour les solutions hybrides de cloud computing.

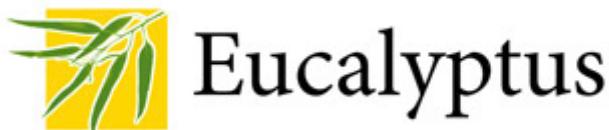


Figure 11 : Logo d'Eucalyptus

L'architecture d'Eucalyptus est constituée de quatre composants principaux :

- Le contrôleur de nœud (Node controller NC) : contrôle l'exécution, et l'arrêt des machines virtuelles présentes sur le nœud où il est exécuté.
- Le contrôleur de cluster (cluster controller CC) : collecte les informations sur les différents nœuds d'un cluster et planifie l'exécution des machines virtuelles sur chaque nœud.
- Le contrôleur de stockage (Warlus) : c'est le composant qui gère l'accès au service de stockage. Il est souvent intégré au contrôleur du cloud (CLC).
- Le contrôleur de cloud (CLC) : C'est le point d'entrée (Front end) des utilisateurs et administrateurs du système. Il collecte des informations sur les nœuds et planifie leur exécution au travers des contrôleurs de clusters (CCs). Il expose les services du cloud à travers une application Web mais également à travers des interfaces compatibles EC2.

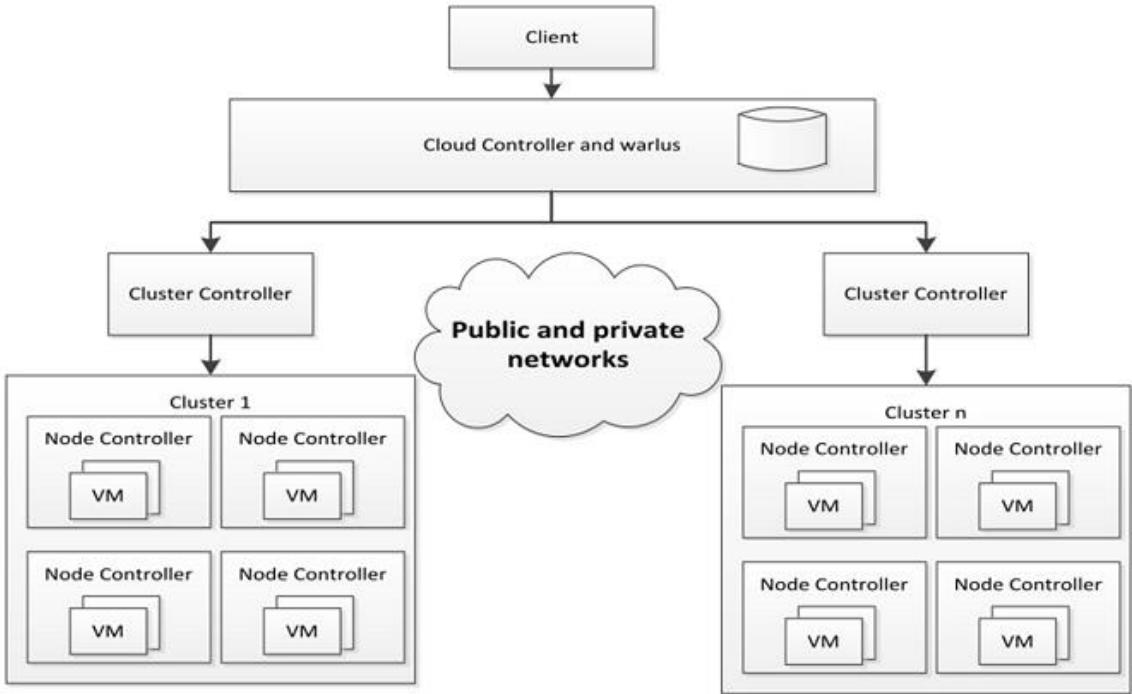


Figure 12 : Architecture d'Eucalyptus

10.2.2- OpenNebula



Figure 13 : Logo d'OpenNebula

Il s'agit d'une plateforme purement open source permettant de déployer des cloud privés, publics et hybrides. Mais l'idée fondamentale de la solution d'OpenNebula est orientée vers une implémentation en privé afin de permettre aux utilisateurs de se connecter aux ressources internes via une interface Web. La solution OpenNebula est rigidement centralisée autour d'un nœud appelé front-end, chargé de la supervision de toute l'architecture. Elle est basée sur un haut niveau de personnalisation en

fournissant plusieurs modules configurables et adaptables aux besoins. Cette haute modularité est à la fois un avantage, en ce sens qu'elle permet de construire l'architecture à sa manière, mais aussi un inconvénient parce qu'elle conduit à des difficultés de configuration entraînant des erreurs de mise en œuvre et des échecs de déploiement des machines virtuelles dans le réseau. OpenNebula supporte les hyperviseurs XEN, KVM et optionnellement VMware.

L'architecture interne d'OpenNebula est constituée de trois couches d'éléments appelées respectivement : tools, core et drivers.

- Tools : c'est l'ensemble des outils de gestion de l'architecture. Il est constitué des interfaces de lignes de commandes CLI (Command Line Interface) pour l'interaction avec le système, d'un portail Web d'administration et d'utilisation du cloud, appelé sunstone localisé au niveau du nœud central de l'architecture.
- Core : ce niveau consiste en un ensemble de composants impliqués dans la gestion et le contrôle des nœuds, des utilisateurs et des machines virtuelles de l'architecture. Ces différents composants communiquent en utilisant le protocole XML RPC.
- Driver : c'est à ce niveau que se déroulent les processus liés aux transferts de machines virtuelles d'un nœud à un autre.

Ces différentes couches s'observent sur le nœud frontal de l'architecture.

Ce dernier est ensuite relié aux nœuds d'exécution (hosts) par l'intermédiaire d'un réseau local. Les nœuds d'exécution (hosts) ne requièrent l'installation d'aucun service d'OpenNebula. Seul le front-end héberge tous les services de supervision et de gestion du cloud ; ce qui rend ce nœud l'unique point de défaillance du système.

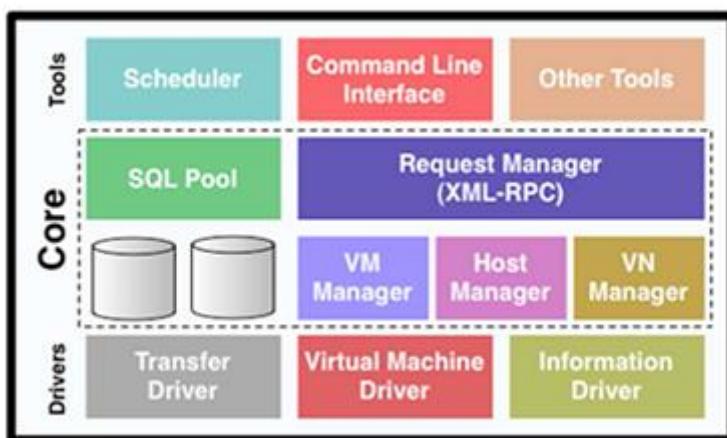


Figure 14 : Architecture interne d'OpenNebula

10.2.3- Xen Cloud Plateform

Xen Cloud Platform (ou XCP) est une solution de virtualisation open source clé en main qui fournit une virtualisation et un cloud computing prêts à l'emploi. XCP inclut Xen Hypervisor, la pile d'outils API Xen prête à l'emploi pour l'entreprise et des intégrations pour les solutions de cloud, de stockage et de mise en réseau. Les fonctionnalités supplémentaires disponibles dans XCP incluent :

- Cycle de vie d'une machine virtuelle : instantanés en direct, point de contrôle, migration.
- Pools de ressources : stockage flexible et mise en réseau.
- Suivi des événements : progrès, notification.
- Fonctions de mise à niveau et de correction.
- Surveillance des performances et alertes en temps réel.
- Prise en charge intégrée et modèles pour les invités Windows et Linux.
- Prise en charge Open vSwitch intégrée.
- Storage XenMotion® Live Migration (migration entre pools, migration VDI)..



Figure 15 : Logo de Xen Cloud Plateform

10.2.4- Openstack

Openstack est un ensemble de logiciels open source permettant de déployer des infrastructures de cloud computing (infrastructure en tant que service). La technologie possède une architecture modulaire composée de plusieurs projets corrélos (Nova, Swift, Glance...) qui permettent de contrôler les différentes

ressources des machines virtuelles telles que la puissance de calcul, le stockage ou encore le réseau inhérent au centre de données sollicité.

OpenStack est un logiciel libre distribué selon les termes de la licence Apache.



Figure 16 : Logo d'Openstack

10.2.5- OwnCloud

OwnCloud est un logiciel libre offrant une plateforme de services de stockage et partage de fichiers et d'applications diverses en ligne. Il est présenté comme une alternative à Dropbox, lequel est basé sur un Cloud public. Dans OwnCloud, le stockage des données se fait au sein de l'infrastructure de l'entreprise et les accès sont soumis à la politique de sécurité informatique de celle-ci.

OwnCloud peut être installé sur n'importe quel serveur supportant une version récente de PHP (au moins 5.4 pour OwnCloud 8) et supportant SQLite (base de données par défaut), MariaDB, MySQL ou PostgreSQL, à l'exception notable des serveurs sous Mac OS X.



Figure 17 : Logo d'OwnCloud

10.2.6- Cloudstack

CloudStack est un logiciel de cloud computing à code source ouvert permettant de créer, de gérer et de déployer des services de cloud d'infrastructure. Il utilise des hyperviseurs existants tels que KVM, VMware vSphere, VMware ESXi, VMware vCenter et XenServer/XCP pour la virtualisation. En plus de sa propre API, CloudStack prend également en charge l'API Amazon Web Services (AWS) et l'interface Open Cloud Computing du forum Open Grid.



Figure 18 : Logo de Cloudstack

Chapitre 3 : Analyse, présentation et installation de notre solution SIEM

Introduction

Dans ce chapitre, nous allons commencer par l'étude des solutions libres (Open source) que nous allons installer ainsi que leur principe de fonctionnement. Ensuite, nous présenterons les procédures d'installation de notre solution SIEM dans les deux environnements suivants : « réseau WAN et LAN ».

1- Notions de base

Elasticsearch est un serveur utilisant Lucene pour l'indexation et la recherche des données. Il fournit un moteur de recherche distribué et multi-entité à travers une interface REST. C'est un logiciel libre écrit en Java et publié en open source sous licence Apache.



Elasticsearch

Logstash est un outil open source de collecte, analyse et stockage de logs. Il est généralement associé avec Elasticsearch, moteur de recherche distribué, et Kibana, interface d'Elasticsearch. Il fournit un pipeline en temps réel pour les collectes de données. Il collecte les syslogs, les convertit en documents JSON et les stocke dans Elasticsearch. Logstash est capable d'intégrer une multitude de sources simultanément.



Logstash

Kibana est un outil de visualisation de données, open source, pour Elasticsearch publié sous la licence libre Apache version 2. Il fournit des fonctions de visualisation sur du contenu indexé dans une grappe Elasticsearch. Les utilisateurs peuvent créer des diagrammes en barre, en ligne, des nuages de points, des camemberts et des cartes de grands volumes de données.



Kibana

Filebeat est un expéditeur de journaux appartenant à la famille "Beats", un groupe d'expéditeurs légers installés sur des hôtes pour l'envoi de différents types de données dans la pile ELK (Elasticsearch-Logstash-Kibana) à des fins d'analyse.



Filebeat

Voici les expéditeurs "Beats" qui sont actuellement disponibles chez Elastic:



- Filebeat : collecte et expédie les fichiers journaux "Syslogs".

Beats

- Metricbeat : collecte les métriques des systèmes et services.
- Packetbeat : collecte et analyse les données du réseau.
- Winlogbeat : collecte les journaux d'événements Windows.
- Auditbeat : collecte les données de la structure d'audit Linux et surveille l'intégrité des fichiers.
- Heartbeat : surveille la disponibilité des services avec une vérification active.
- Functionbeat : collecte et expédie les événements générés par les services de cloud.

Nginx est un logiciel libre de serveur Web (ou HTTP) ainsi qu'un proxy inverse écrit par Igor Sysoev, dont le développement a débuté en 2002 pour les besoins d'un site russe à très fort trafic (Rambler). C'est depuis avril 2019, le serveur web le plus utilisé au monde.



Suricata est un logiciel open source de détection d'intrusion (IDS), de prévention d'intrusion (IPS), et de supervision de sécurité réseau (NSM). Il est développé par la fondation OISF (Open Information Security Foundation). Suricata permet l'inspection des Paquets en Profondeur (DPI) [5]. De nombreux cas d'utilisations déontologiques peuvent être mis en place permettant notamment la remontée d'informations qualitatives et quantitatives.



MariaDB est un système de gestion de base de données édité sous licence GPL. Il s'agit d'un fork communautaire de MySQL : la gouvernance du projet est assurée par la fondation MariaDB, et sa maintenance par la société Monty Program AB, créateur du projet. Cette gouvernance confère au logiciel l'assurance de rester libre.



Zabbix est un logiciel libre permettant de surveiller l'état de divers services réseau, serveurs et autres matériels réseau et produisant des graphiques dynamiques de consommation des ressources. C'est un logiciel open source créé par Alexei Vladishev.



AGENT AVAILABILITY

Zabbix Agent : Il s'agit d'un logiciel qui se charge de l'envoi des données du serveur surveillé au serveur Zabbix. C'est un logiciel natif, développé en langage C, peut s'exécuter sur diverses plates, notamment Linux, UNIX et



Windows, et collecter des données telles que l'utilisation du processeur, de la mémoire, des disques et des interfaces réseau à partir d'un périphérique. L'installation d'un Zabbix Agent sur un hôte offre essentiellement une surveillance active des ressources locales, des applications, ... etc

Zabbix Frontend est tout simplement l'interface de visualisation des évènements, mais aussi, et surtout l'interface d'administration et de configuration de Zabbix. Etant une interface Web (php), il a l'avantage d'être accessible depuis n'importe quelle plateforme possédant un navigateur internet.

Zabbix Proxy permet de collecter des informations sur la performance et la disponibilité des données sur un hôte, avant de les transmettre au Zabbix Server. Il offre la possibilité de réduire la charge d'un serveur Zabbix. En effet, toutes les informations collectées peuvent être traitées en local, avant leur transmission au serveur.

Il est idéal pour une surveillance centralisée de sites distants, fonctionnant comme un serveur intermédiaire, il remplit parfaitement son rôle de collecteur de données d'équipements variés. Distant d'un serveur Zabbix, il agit comme une sonde de collecte et de traitement des données.

Grafana est un logiciel libre sous licence Apache 2.0 qui permet la visualisation et la mise en forme de données métriques. Il permet de réaliser des tableaux de bord et des graphiques depuis plusieurs sources dont des bases de données de série temporelle (Time Series Database) comme Graphite, InfluxDB et OpenTSDB.



[5] L'inspection profonde de paquets ou en anglais Deep Packet Inspection, abrégée IPP ou DPI est une technique d'analyse des flux passant dans des équipements réseau au-delà de l'entête. L'équipement recherche des informations dans la charge utile des paquets plutôt que dans les entêtes (à l'inverse des approches classiques). Des signatures sont le plus souvent recherchées pour détecter des types de flux et agir en conséquence.

2- Principe de fonctionnement

2.1- Elastic stack (Pile ELK)

Comme mentionné précédemment, il y a plusieurs types d'expéditeurs chez Elastic. Le choix du type d'expéditeur adéquat dépend du type de données que nous souhaitons analyser.

Dans notre cas, nous allons opter pour l'expéditeur « Filebeat » afin d'analyser les fichiers journaux. Nous allons adopter l'architecture suivante :



Figure 19 : Principe de fonctionnement d'Elastic stack

Lorsque l'événement est enregistré dans le fichier journal, Filebeat le collecte et se charge de l'expédier à l'instance Logstash.

Logstash garantit que les entrées (événements) sont transformées en un des formats cible pris en charge qui seront ensuite stockées dans la base de données NoSQL "Elasticsearch".

Le traitement de Logstash se déroule en trois étapes :

- + Entrée (Input) : L'entrée, en plus d'un fichier (file), peut également être un syslog, redis, etc ... [6]
- + Transformation et filtrage (Filter) : Cette étape se déroule conformément à l'ensemble de règles définies précédemment [7]. Par exemple, une transformation de format de date, les informations d'adresse IP (ville/pays de l'adresse IP) peuvent également être enregistrées.
- + Sortie (Output) : La sortie peut être transformée via des plugins dans presque n'importe quel format. Dans notre cas, ce sera "elasticsearch" [8].

Dès que Logstash termine son travail, Elasticsearch peut déjà traiter les données. Ces dernières sont indexées et stockées sous format JSON dans la base Elasticsearch.

Kibana est fondamentalement juste une interface utilisateur permettant l'affichage des données.

Le serveur Web Nginx agit en tant que serveur mandataire inverse (Proxy Inverse) pour l'application Kibana.

[6] Voir liste des plugins d'entrée de Logstash :

<https://www.elastic.co/guide/en/logstash/current/input-plugins.html>

[7] Voir liste des plugins de filtrage de Logstash :

<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>

[8] Voir liste des plugins de sortie de Logstash :

<https://www.elastic.co/guide/en/logstash/current/output-plugins.html>

2.2- IDS Suricata

Suricata fonctionne en obtenant un paquet à la fois du système. Ceux-ci sont ensuite pré-traités, après, ils sont transmis au moteur de détection. Suricata peut utiliser "pcap" [9] à cet effet en mode IDS (Intrusion Detection System), mais peut également se connecter à une fonction spéciale de Linux, nommée "nfnetlink_queue". Cette file d'attente permet au noyau de copier un paquet dans un processus utilisateur (dans ce cas, Suricata). Il attend ensuite que ce processus de l'espace utilisateur émette un verdict sur ce paquet, avec 'accept' et 'drop' étant les deux verdicts possibles.

Suricata fonctionne avec des règles. Ces règles peuvent comporter des actions telles que 'alert', 'log', etc. Suricata IPS (Intrusion Prevention System) introduit trois nouvelles actions spécifiques au mode IPS, 'drop', 'sdrop' et 'reject' (sdrop est l'action drop en mode silencieux). Comme indiqué ci-dessus, les deux seuls verdicts possibles pour un paquet sont 'accept' et 'drop', alors comment Suricata peut-il rejeter un paquet ? Cela se fait en deux étapes :

- le paquet est abandonné en utilisant le verdict 'drop'.
- Suricata envoie une erreur icmp ou tcp-reset à l'adresse IP incriminée.

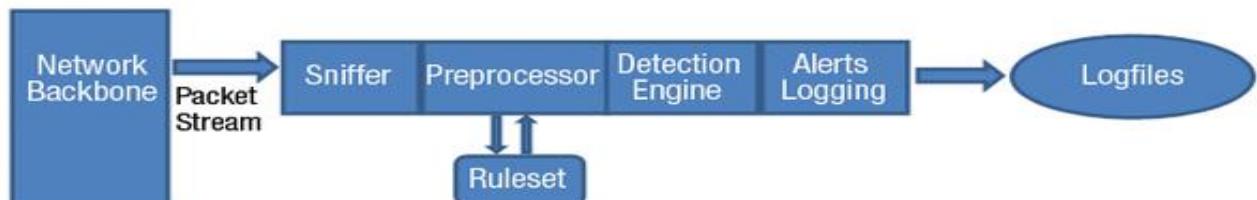


Figure 20 : Principe de fonctionnement de Suricata

[9] pcap (« packet capture ») est une interface de programmation permettant de capturer un trafic réseau. Elle est implémentée sous les systèmes GNU/Linux, FreeBSD, NetBSD, OpenBSD et Mac OS X par la bibliothèque libpcap. WinPcap est le portage sous Windows de libpcap.

2.3- Outil de supervision Zabbix

Zabbix comporte trois composants principaux : Serveur Zabbix, agent Zabbix et interface Web. Les agents sont installés sur les composants informatiques pour vérifier les performances et collecter les données. L'agent fait ensuite un rapport à un serveur de gestion Zabbix centralisé. Ces informations sont incluses dans les rapports ou présentées de manière visuelle dans l'interface utilisateur graphique de Zabbix. S'il y a des problèmes concernant ce qui est surveillé, Zabbix enverra une notification ou une alerte à l'utilisateur.

Zabbix utilise le protocole JSON pour communiquer avec les agents Zabbix.

Ces derniers prennent en charge les checks passifs "polling" (interrogation) et actifs "trapping" (interception). Zabbix peut effectuer des checks selon un intervalle, mais il est également possible de programmer des heures spécifiques pour l'interrogation des éléments (items [10]).

Checks passifs (polling) :

- Le serveur Zabbix (ou proxy) demande une valeur à l'agent Zabbix en lui envoyant une requête contenant un item (l'élément que nous souhaitons avoir sa valeur).
- L'agent traite la demande et renvoie la valeur au serveur Zabbix (ou au proxy).

Checks actifs (trapping) :

- L'agent Zabbix demande au serveur Zabbix (ou au proxy) une liste des checks actifs (liste d'items définie par l'intermédiaire des templates [11]).
- L'agent envoie les résultats périodiquement.

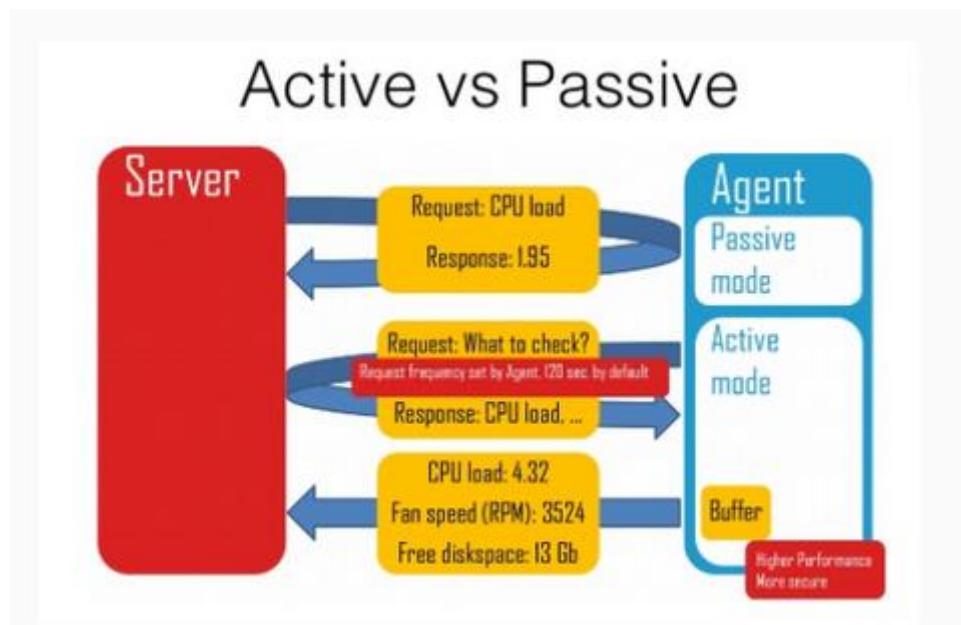


Figure 21 : Modes de checks actifs et passifs

Le serveur Zabbix peut fonctionner sans avoir recours aux agents, mais dans ce cas, il ne remontera qu'une quantité limitée d'informations. Il peut également utiliser le protocole SNMP [12] pour superviser des hôtes.

L'interface graphique Web de Zabbix permet aux utilisateurs de visualiser leur environnement informatique via des tableaux de bord personnalisables basés sur des widgets, des graphiques, des cartes réseau, des diaporamas et des rapports. L'URL pour se connecter au serveur Zabbix est :

http://ZABBIX_SERVER_IP_ADDRESS/zabbix/

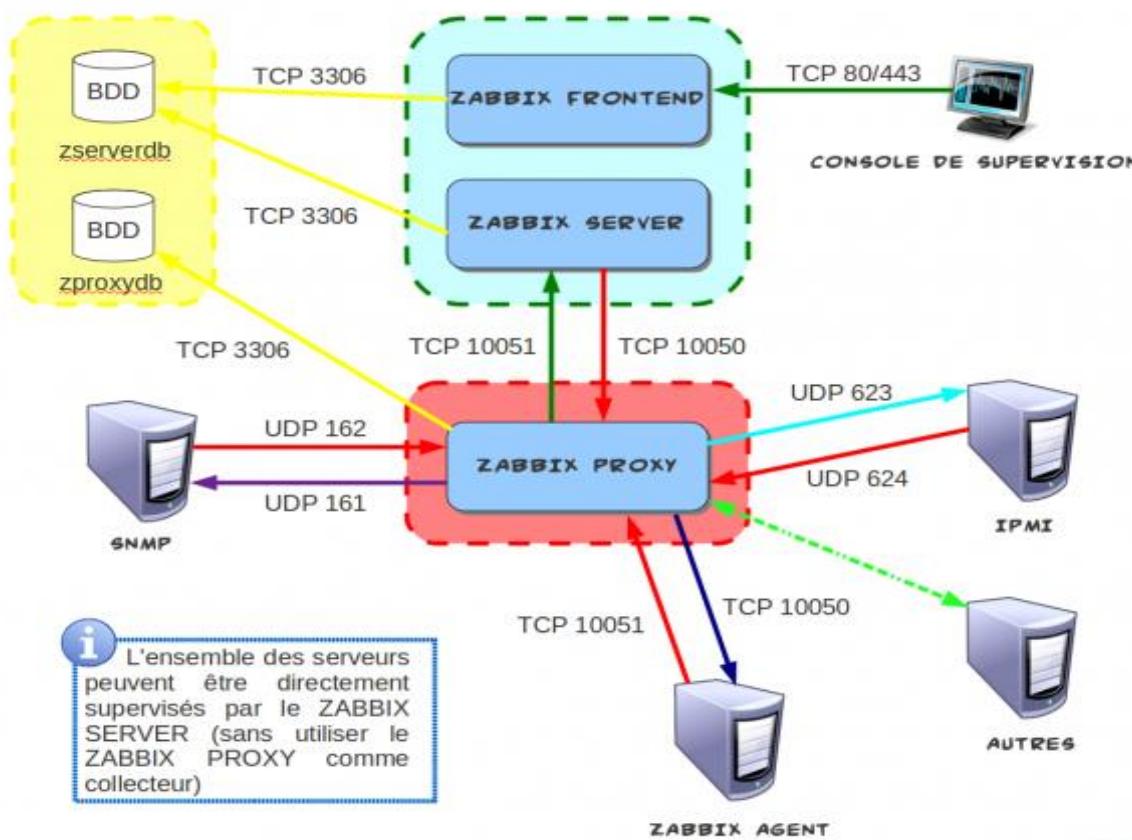
Afin de réduire la charge d'un serveur Zabbix, nous pouvons installer un "Zabbix Proxy" qui va agir comme un serveur intermédiaire, c'est-à-dire un collecteur. Il utilise comme le serveur Zabbix une base de données pour le stockage des informations collectées avant les transmettre à ce dernier.

L'outil de supervision Zabbix dispose également d'un système d'alerte pour avertir les administrateurs de toute nouvelle alerte.

Il génère une alerte (ou notification) de type EMAIL, SMS, ou JABBER.

Conclusion

Le schéma ci-dessous montre les protocoles et flux utilisés par les différents éléments qui composent une supervision Zabbix :



[13]

Figure 22 : Architecture du système de supervision Zabbix

[10] Item : une donnée particulière que vous voulez recevoir d'un hôte, une métrique de données.

[11] Templates : un ensemble d'entités (éléments, déclencheurs, graphiques, écrans, applications, règles de découverte de bas niveau, scénarios Web) prêtes à être appliquées à un ou plusieurs hôtes.

L'intérêt des modèles est d'accélérer le déploiement des tâches de surveillance sur un hôte ; également pour faciliter l'application de modifications collectives aux tâches de surveillance. Les modèles sont directement liés à des hôtes individuels.

[12] Simple Network Management Protocol (abrégé SNMP), en français « protocole simple de gestion de réseau », est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer des problèmes réseaux et matériels à distance.

[13] L'Interface de gestion intelligente de matériel, (ou IPMI, Intelligent Platform Management Interface) est un ensemble de spécifications d'interfaces communes avec du matériel informatique (principalement des serveurs) permettant de surveiller certains composants (ventilateur, sonde de température, ...), mais également de contrôler l'ordinateur à distance, reboot, interrupteur, console à distance.

Pour vous familiariser avec les termes couramment utilisés dans Zabbix, vous pouvez consulter la page suivante :

<https://www.zabbix.com/documentation/4.0/fr/manual/definitions>

3- Procédure d'installation de notre solution SIEM dans un réseau WAN

Conditions préalables :

Privilège root.

Nous allons procéder à :

- Désactivation de SELinux sur le serveur ELK : alan-elk1
- Installation de Java
- Installation et configuration d'Elasticsearch
- Installation et configuration de Kibana avec Nginx
- Installation et configuration de Logstash
- Installation et configuration de Filebeat sur les deux serveurs Client : miro-node1 et miro-node2
- Installation et configuration de Suricata sur les deux serveurs Client : miro-node1 et miro-node2

- Configuration du PAT (Port Address Translation) sur le routeur lié au serveur alan-elk1 et redirection des ports 5044 et 5045
- Configuration du PAT (Port Address Translation) sur le routeur lié aux serveurs miro-node1 et miro-node2 et redirection des ports 49155 et 49156

3.1- Architecture

L'architecture réseau dans laquelle nous déployons notre solution est la suivante :

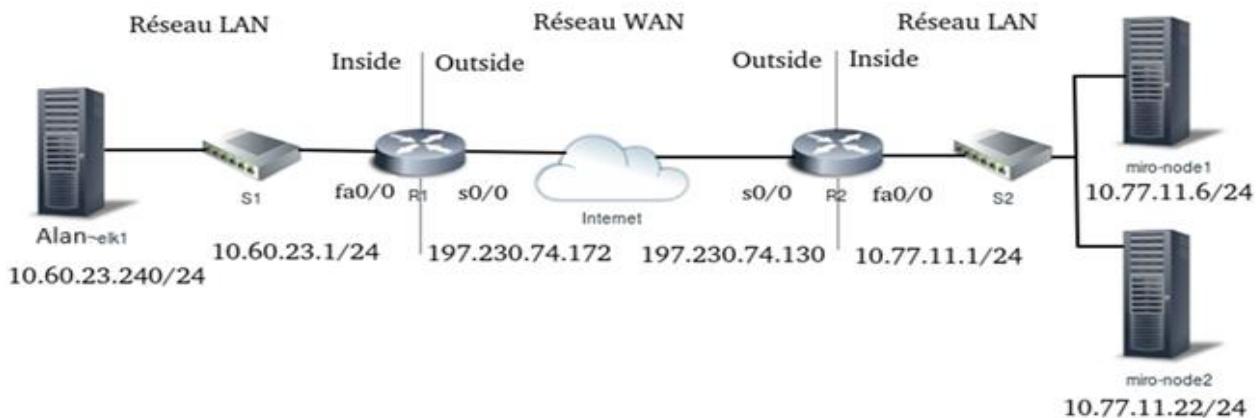


Figure 23 : Architecture du réseau du déploiement

- Serveur ELK : alan-elk1, adresse publique : 197.230.74.172 – CentOS 7
- Serveur Client : miro-node1, adresse publique : 197.230.74.130 – Ubuntu
- Serveur Client : miro-node2, adresse publique : 197.230.74.130 – Ubuntu
- Routeur lié au serveur alan-elk1 : R1
- Routeur lié aux serveurs miro-node1 et miro-node2 : R2

3.2- Désactivation de SELinux

Nous allons désactiver SELinux sur le serveur ELK :

```
vi /etc/sysconfig/selinux
```

Modification de la valeur de SELinux en 'disabled' :

```
SELINUX=disabled
```

Nous redémarrons le serveur:

```
reboot
```

Vérification de l'état SELinux après redémarrage :

```
getenforce
```

Le résultat doit être désactivé.

3.3- Installation de Java

Java est requis pour le déploiement de la pile Elastic. Elasticsearch nécessite Java 8.

Téléchargement et installation de Java 8 :

```
yum -y install java-1.8.0 wget
```

Vérification de la version de Java :

```
java -version
```

3.4- Installation et configuration d'Elasticsearch

Avant d'installer Elasticsearch, nous ajoutons de la clé elastic.co au serveur :

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Téléchargement et installation d'Elasticsearch 6.7.1 :

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.7.1.rpm
```

```
rpm -ivh elasticsearch-6.7.1.rpm
```

Configuration du fichier elasticsearch.yml :

```
vi /etc/elasticsearch/elasticsearch.yml
```

Activation du verrouillage de la mémoire pour Elasticsearch (cela désactive l'échange de mémoire pour Elasticsearch) :

```
bootstrap.memory_lock: true
```

Dans le bloc "Network", nous enlevons les commentaires des lignes network.host et http.port :

```
network.host: localhost
http.port: 9200
```

Nous éditons le fichier de configuration sysconfig pour Elasticsearch :

```
vi /etc/sysconfig/elasticsearch
```

Nous enlevons le commentaire de la ligne MAX_LOCKED_MEMORY :

```
MAX_LOCKED_MEMORY=unlimited
```

Activation et démarrage du service de Elasticsearch :

```
systemctl daemon-reload  
systemctl enable elasticsearch  
systemctl start elasticsearch
```

Vérification ("l'état du port 9200 doit être "LISTEN") :

```
netstat -plntu
```

ou bien en utilisant la commande curl :

```
curl -XGET 'localhost:9200/?pretty'
```

ou bien :

```
systemctl status elasticsearch
```

3.5- Installation et configuration de Kibana avec Nginx

3.5.1- Installation et configuration de Kibana

Kibana écoute l'adresse IP de l'hôte local et Nginx agit en tant que proxy inverse pour l'application Kibana.

Téléchargement et installation de Kibana 6.7.1 :

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-6.7.1-x86_64.rpm
```

```
rpm -ivh kibana-6.7.1-x86_64.rpm
```

Nous éditons le fichier de configuration Kibana :

```
vi /etc/kibana/kibana.yml
```

Nous enlevons les commentaires des lignes de configuration pour server.port, server.host et elasticsearch.hosts :

```
server.port: 5601
```

```
server.host: "localhost"  
elasticsearch.hosts: ["http://localhost:9200"]
```

Activation et démarrage du service de Kibana :

```
systemctl enable kibana  
systemctl start kibana
```

Vérification (Kibana sera exécuté sur le port 5601) :
netstat -plntu

ou bien :

```
systemctl status kibana
```

3.5.2- Installation et configuration de Nginx

Nginx est disponible dans le package "epel" :

```
yum -y install epel-release
```

Installation des packages Nginx et httpd-tools :

```
yum -y install nginx httpd-tools
```

Le paquet httpd-tools contient des outils pour le serveur Web.

Nous éditons le fichier de configuration Nginx et supprimons le bloc 'server {}' pour pouvoir ajouter une nouvelle configuration d'hôte virtuel :

```
vi /etc/nginx/nginx.conf
```

Nous créons un nouveau fichier de configuration d'hôte virtuel dans le répertoire conf.d :

```
vi /etc/nginx/conf.d/ alan-elk1.conf
```

Configuration à y ajouter :

```
=====  
server {  
  
    listen 80;  
    server_name alan-elk1;
```

```
auth_basic "Restricted Access";
auth_basic_user_file /etc/nginx/.kibana-user;

location / {
    proxy_pass http://localhost:5601;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}
=====
```

Création d'un nouveau fichier d'authentification :

```
htpasswd -c /etc/nginx/.kibana-user admin
TYPE_PASSWORD
```

Test de la configuration de Nginx et démarrage du service :

```
nginx -t
systemctl enable nginx
systemctl start nginx
```

Vérification :

```
systemctl status nginx
```

3.6- Installation et configuration de Logstash

Téléchargement et installation de Logstash :

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.7.1.rpm
rpm -ivh logstash-6.7.1.rpm
```

Nous générerons un nouveau fichier de certificat SSL afin que le client puisse identifier le serveur ELK :

Nous générerons le fichier de certificat avec la commande openssl :

```
openssl req -config /etc/pki/tls/openssl.cnf -x509 -days 3650 -batch -nodes -newkey
rsa:2048 -keyout /etc/pki/tls/private/logstash-forwarder.key -out
/etc/pki/tls/certs/logstash-forwarder.crt -subj /CN=alan-elk1
```

Nous créons un nouveau fichier de configuration 'configlogstash.conf' dans le sous-répertoire 'conf.d' pour configurer les syslogs provenant des machines clients, le traitement des syslog et pour définir la sortie Elasticsearch :

```
vi /etc/logstash/conf.d/configlogstash.conf
```

```
=====
input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
  }
  beats {
    port => 5045
    ssl => true
    ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
  }
}

filter {

  if ("Syslogs" or "Authentication-logs" or "Suricata-logs" in [tags]) {
    mutate{
      add_field => [ "event_received_date", "%{@timestamp}" ]
    }
  }

  if "Syslogs" or "Authentication-logs" in [tags] {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}%{SYSLOGHOST:syslog_hostname}%{DATA:syslog_program}(?:\[ %{POSINT:syslog_pid}\]\)?%{GREEDYDATA:syslog_message}" }
    }

    date {
      match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
```

```

if "Suricata-logs" in [tags] {
    date {
        match => [ "timestamp", "ISO8601" ]
    }
}

ruby {
    code => "
        if event.get('event_type') == 'fileinfo'
            event.set('[fileinfo][type]', event.get('[fileinfo][magic]').to_s.split(',')[])
        end
    "
}
}

if [http] {
    useragent {
        source => "[http][http_user_agent]"
        target => "[http][user_agent]"
    }
}

if [src_ip] {
    geoip {
        source => "src_ip"
        target => "geoip"
    }
}

if ![geoip.ip]{
    if [dest_ip] {
        geoip {
            source => "dest_ip"
            target => "geoip"
        }
    }
}
}

output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "logstash-%{+YYYY.MM.dd}"
    }
}
=====
```

Nous configurons le pare-feu pour que Logstash reçoive les syslogs des machines clients :

```
firewall-cmd --permanent --add-port=5044/tcp  
firewall-cmd --permanent --add-port=5045/tcp  
firewall-cmd --reload
```

Activation et démarrage du service de logstash :

```
systemctl enable logstash  
systemctl start logstash
```

Vérification :

```
systemctl status logstash
```

ou bien :

```
netstat -plntu
```

Dernière étape : Transfert du certificat aux serveurs « miro-node1 et miro-node2»

Nous copions le fichier de certificat sur le serveur miro-node1 via le port 49155 :

```
scp -P 49155 /etc/pki/tls/certs/logstash-forwarder.crt  
root@197.230.74.130:/etc/pki/tls/certs/
```

Nous copions le fichier de certificat sur le serveur miro-node2 via le port 49156 :

```
scp -P 49156 /etc/pki/tls/certs/logstash-forwarder.crt  
root@197.230.74.130:/etc/pki/tls/certs/
```

Remarque : nous devons configurer le service sshd sur le serveur miro-node1 pour que SSH écoute sur le port 49155. Idem pour le port 49156 sur le serveur miro-node2.

3.7- Installation et configuration de Filebeat sur les deux serveurs Client : miro-node1 et miro-node2

3.7.1- Serveur miro-node1

Conditions préalables :

Nous éditions le fichier de configuration sshd afin que SSH écoute sur deux ports :

```
vi /etc/ssh/sshd_config
```

Port 22
Port 49155

Redémarrage du service ssh :

/etc/init.d/ssh restart

Nous configurons le pare-feu pour que le serveur miro-node1 reçoive le fichier de certificat sur le port 49155 :

ufw allow 49155/tcp

=====

Nous importons la clé Elasticsearch sur le serveur client:

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

Téléchargement et installation de Filebeat avec dpkg:

wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.7.1-amd64.deb

dpkg -i filebeat-6.7.1-amd64.deb

Nous éditons ensuite le fichier de configuration "filebeat.yml" :

vi /etc/filebeat/filebeat.yml

Dans la section "filebeat.inputs", nous ajoutons les nouveaux fichiers journaux. Nous allons ajouter trois fichiers '/var/log/syslog', '/var/log/auth.log' et '/var/log/suricata/eve.json' :

=====

filebeat.inputs:

```
- type: log
  enabled: true
  paths:
    - /var/log/syslog
  tags: ["Syslogs"]
- type: log
  paths:
    - /var/log/auth.log
```

```
tags: ["Authentication-logs"]
- type: log
  paths:
    - /var/log/suricata/eve.json
tags: ["Suricata-logs"]
json.keys_under_root: true
json.add_error_key: true
```

=====

Filebeat utilise Elasticsearch comme cible de sortie par défaut, nous allons le changer en Logstash. Nous allons donc désactiver la sortie Elasticsearch (en commentant la sortie) et activons la sortie Logstash (en enlevant les commentaires) :

```
=====
output.logstash:
  # The Logstash hosts
  hosts: ["alan-elk1:5044"]
  ssl.certificateAuthorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]
```

=====

Remarque : étant donné que nous avons créé un certificat avec le nom de domaine "alan-elk1", nous allons l'utiliser pour la configuration de la sortie Logstash.

Ensuite, nous ajoutons une entrée d'hôte (serveur alan-elk1) sur la machine miro-node1 :

```
vi /etc/hosts
```

```
197.230.74.172 alan-elk1 alan-elk1
```

Activation et démarrage du service de Filebeat :

```
systemctl enable filebeat
```

```
systemctl start filebeat
```

Vérification :

```
systemctl status filebeat
```

3.7.2- Serveur miro-node2

Conditions préalables :

Nous éditions le fichier de configuration sshd afin que SSH écoute sur deux ports :

```
vi /etc/ssh/sshd_config
```

Port 22

Port 49156

Redémarrage du service ssh :

```
/etc/init.d/ssh restart
```

Nous configurons le pare-feu pour que le serveur client2 reçoive le fichier de certificat sur le port 49156 :

```
ufw allow 49156/tcp
```

```
=====
Nous suivons la même procédure d'installation indiquée ci-dessus et nous activons la sortie logstash comme suit :
```

```
=====
output.logstash:
  # The Logstash hosts
  hosts: ["alan-elk1:5045"]
  ssl.certificateAuthorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]
```

3.8- Installation et configuration de Suricata sur les deux serveurs Client : miro-node1 et miro-node2

Installation des dépendances sur Ubuntu :

```
apt -y install libpcre3 libpcre3-dbg libpcre3-dev build-essential autoconf \
automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev \
libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libjansson4 pkg-config
```

Téléchargement et installation de Suricata :

```
wget http://www.openinfosecfoundation.org/download/suricata-4.1.4.tar.gz
tar -xvf suricata-4.1.4.tar.gz
cd suricata-4.1.4
```

Pour avoir la configuration dans '/etc/suricata' au lieu de '/usr/local/etc/' et avoir '/var/log/suricata' au lieu de '/usr/local/var/log/suricata' comme répertoire de Log, nous utilisons :

```
./configure --sysconfdir=/etc --localstatedir=/var
```

Installation :

```
make  
make install
```

Installation des fichiers de configuration par défaut de Suricata :

```
make install-conf
```

Téléchargement et installation des règles IDS :

```
wget http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz  
tar zxvf emerging.rules.tar.gz  
cp -r rules /etc/suricata/
```

Vérification :

```
ls /etc/suricata/rules
```

Afin de créer le fichier /var/lib/suricata/rules/suricata.rules, nous exécutons la commande suivante depuis le sous-répertoire suricata-4.1.4 :

```
make install-rules
```

Configuration de Suricata :

Nous éditons le fichier de configuration suricata.yaml comme suit :

- Le mot clé "default-log-dir" doit pointer vers l'emplacement des fichiers journaux Suricata.
- Dans la section "vars" :
"HOME_NET" doit pointer sur le réseau local à inspecter par Suricata.
"! \$ HOME_NET" (attribué à EXTERNAL_NET) fait référence à tout autre réseau que le réseau local.
- Nous activons la sortie "dns".
- Nous activons l'option "force-magic" de la sortie "files".

- Nous activons la sortie "http.log" pour avoir le fichier de sortie "http.log" qui contiendra tous les logs HTTP (optionnel).
- L'IDS moderne a mis au point une inspection dite "ciblée", dans laquelle le moteur d'inspection adapte son algorithme de détection en fonction du système d'exploitation cible du trafic. Dans notre cas, nous allons activer "Linux" comme stratégie IDS par défaut. A cet effet, Suricata appliquera l'inspection basée sur Linux si aucune information de système d'exploitation n'est connue pour une adresse IP particulière.

```
vi /etc/suricata/suricata.yaml
```

```
=====
default-log-dir: /var/log/suricata/
```

vars:

```
# more specific is better for alert accuracy and performance address-groups:
HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
#HOME_NET: "[192.168.1.0/24]"
#HOME_NET: "[10.0.0.0/8]"
#HOME_NET: "[172.16.0.0/12]"
#HOME_NET: "any"

EXTERNAL_NET: "!$HOME_NET"
#EXTERNAL_NET: "any"
```

port-groups:

```
HTTP_PORTS: "80"
SHELLCODE_PORTS: "!80"
ORACLE_PORTS: 1521
SSH_PORTS: 22
DNP3_PORTS: 20000
MODBUS_PORTS: 502
FILE_DATA_PORTS: "[${HTTP_PORTS},110,143]"
FTP_PORTS: 21
```

- dns:

```
# Enable/disable this logger. Default: enabled.
enabled: yes
```

- files:

```
force-magic: yes
```

```
- http-log:  
  enabled: true  
  fileneame: http.log
```

```
host-os-policy:  
# Make the default policy windows.  
windows: []  
bsd: []  
bsd-right: []  
old-linux: []  
linux: [0.0.0.0/0]  
old-solaris: []  
solaris: []  
hpx10: []  
hpx11: []  
irix: []  
macos: []  
vista: []  
windows2k3: []
```

Test et exécution de Suricata :

Test des erreurs :

```
suricata -c /etc/suricata/suricata.yaml -i enp0s3 --init-errors-fatal  
(avec enp0s3 comme nom d'interface).
```

Exécution de Suricata :

```
suricata -c /etc/suricata/suricata.yaml -i enp0s3
```

Exécution de Suricata en tant que daemon :

```
suricata -D -c /etc/suricata/suricata.yaml -i enp0s3
```

Vérification:

```
tail -f /var/log/suricata/fast.log  
tail -f /var/log/suricata/http.log  
tail -f /var/log/suricata/eve.json
```

3.9- Configuration du PAT (Port Address Translation) sur le routeur lié au serveur alan-elk1 et redirection des ports 5044 et 5045

L'adresse privée du serveur alan-elk1 est : 10.60.23.240/24.

L'adresse publique du serveur alan-elk1 est : 197.230.74.172.

L'adresse publique des serveurs miro-node1 et miro-node2 est : 197.230.74.130.

Supposant que l'interface fa0/0 est du côté privé et l'interface s0/0 est du côté public.

3.9.1- Configuration du PAT

Nous allons suivre les étapes ci-dessous afin de configurer le NAT overload (PAT) sur notre routeur :

1- Identification de l'interface LAN (celle qui est connectée à notre réseau local).

2- Identification l'interface WAN (celle qui est connectée à Internet).

3- Création d'une règle d'accès ACL permettant le réseau LAN de se connecter au réseau WAN.

4- Association de la règle d'accès ACL à l'interface WAN.

Sur le routeur :

1-Nous identifions l'interface s0/0 comme l'interface WAN (outside) :

```
Router>enable  
Router#configure terminal  
Router(config)#interface serial 0/0  
Router(config-if)#ip nat outside  
Router(config-if)#exit
```

2- Nous identifions l'interface fa0/0 comme l'interface LAN (inside) :

```
Router(config)#interface fastethernet 0/0  
Router(config-if)#ip nat inside  
Router(config-if)#exit
```

3- Nous allons créer la règle d'accès ACL :

```
Router(config)#access-list 1 permit 10.60.23.0 0.0.0.255
```

4- Association de la règle d'accès ACL à l'interface WAN :

```
Router(config)# ip nat inside source list 1 interface serial 0/0 overload
```

3.9.2- Redirection des ports 5044 et 5045

1- Nous allons créer la règle d'accès ACL :

```
Router(config)#access-list 101 permit tcp host 197.230.74.130 host 197.230.74.172  
range 5044 5045
```

Remarque: si la commande ci-dessus n'a pas abouti, nous pouvons la remplacer par celle-ci :

```
Router(config)#access-list 101 permit tcp any any range 5044 5045
```

2- Nous allons créer un pool d'adresses (cette plage contiendra une seule adresse) :

```
ip nat pool TEST 10.60.23.240 10.60.23.240 netmask 255.255.255.0
```

3- Association de la règle d'accès ACL à la plage d'adresses :

```
ip nat inside destination list 101 pool TEST overload
```

Désormais, tout trafic TCP provenant de l'hôte 197.230.74.130 et entrant dans l'interface publique de notre routeur (Serial 0/0) avec une adresse de destination 197.230.74.172 et un port de destination compris entre 5044 et 5045 sera transféré à l'hôte à l'adresse 10.60.23.240 au même port puisque nous avons ajouté le mot clé "overload" à la fin de la commande.

3.10- Configuration du PAT (Port Address Translation) sur le routeur lié aux serveurs miro-node1 et miro-node2

L'adresse publique des serveurs miro-node1 et miro-node2 est : 197.230.74.130.
L'adresse publique du serveur alan-elk1 est : 197.230.74.172.

Supposant que les adresses privées respectives des serveurs miro-node1 et miro-node2 sont : 10.77.11.6/24 et 10.77.11.22/24.

Supposant que l'interface fa0/0 est du côté privé et l'interface s0/0 est du côté public.

3.10.1- Configuration du PAT

Sur le routeur :

1- Nous identifions l'interface s0/0 comme l'interface WAN (outside) :

```
Router>enable  
Router#configure terminal  
Router(config)#interface serial 0/0  
Router(config-if)#ip nat outside  
Router(config-if)#exit
```

2- Nous identifions l'interface fa0/0 comme l'interface LAN (inside) :

```
Router(config)#interface fastethernet 0/0  
Router(config-if)#ip nat inside  
Router(config-if)#exit
```

3- Nous allons créer la règle d'accès ACL :

```
Router(config)#access-list 1 permit 10.77.11.0 0.0.0.255
```

4- Association de la règle d'accès ACL à l'interface WAN :

```
Router(config)# ip nat inside source list 1 interface serial 0/0 overload
```

3.10.2- Redirection du port 49155

1- Nous allons créer la règle d'accès ACL :

```
Router(config)#access-list 101 permit tcp host 197.230.74.172 host 197.230.74.130  
eq 49155
```

Remarque : si la commande ci-dessus n'a pas abouti, nous pouvons la remplacer par celle-ci :

```
Router(config)#access-list 101 permit tcp any any eq 49155
```

2- Nous allons créer un pool d'adresses (cette plage contiendra une seule adresse) :

```
ip nat pool TEST_P49155 10.77.11.6 10.77.11.6 netmask 255.255.255.0
```

3- Association de la règle d'accès ACL à la plage d'adresses :

```
ip nat inside destination list 101 pool TEST_P49155 overload  
Tout trafic TCP provenant de l'hôte 197.230.74.172 et entrant dans l'interface  
publique de notre routeur (Serial 0/0) avec une adresse de destination 197.230.74.130
```

et un port de destination 49155 sera transféré à l'hôte à l'adresse 10.77.11.6 au même port.

3.10.3- Redirection du port 49156

1- Nous allons créer la règle d'accès ACL :

```
Router(config)#access-list 102 permit tcp host 197.230.74.172 host 197.230.74.130  
eq 49156
```

Remarque : si la commande ci-dessus n'a pas abouti, nous pouvons la remplacer par celle-ci :

```
Router(config)#access-list 102 permit tcp any any eq 49156
```

2- Nous allons créer un pool d'adresses (cette plage contiendra une seule adresse) :

```
ip nat pool TEST_P49156 10.77.11.22 10.77.11.22 netmask 255.255.255.0
```

3- Association de la règle d'accès ACL à la plage d'adresses :

```
ip nat inside destination list 102 pool TEST_P49156 overload
```

Tout trafic TCP provenant de l'hôte 197.230.74.172 et entrant dans l'interface publique de notre routeur (Serial 0/0) avec une adresse de destination 197.230.74.130 et un port de destination 49156 sera transféré à l'hôte à l'adresse 10.77.11.22 au même port.

4- Procédure d'installation de notre solution SIEM dans un réseau LAN

Conditions préalables :

Privilèges root.

Nous allons procéder à :

- Désactivation de SELinux sur le serveur ELK : elk-srv
- Installation de Java
- Installation et configuration d'Elasticsearch
- Installation et configuration de Kibana avec Nginx
- Installation et configuration de Logstash
- Installation et configuration de Filebeat sur les deux serveurs Client : client1-srv et client2-srv

- Installation et configuration de Suricata sur les deux serveurs Client : client1-srv et client2-srv
- Installation et configuration de Zabbix sur le serveur zabbix-srv
- Installation et configuration de Zabbix-Agent sur les deux serveurs Client : client1-srv et client2-srv
- Installation et configuration de Grafana sur le serveur zabbix-srv
- Configuration initiale et test de la pile ELK
- Configuration initiale et test du serveur Zabbix
- Configuration initiale et test du serveur Grafana

4.1- Architecture

L'architecture réseau dans laquelle nous déployons notre solution est la suivante :

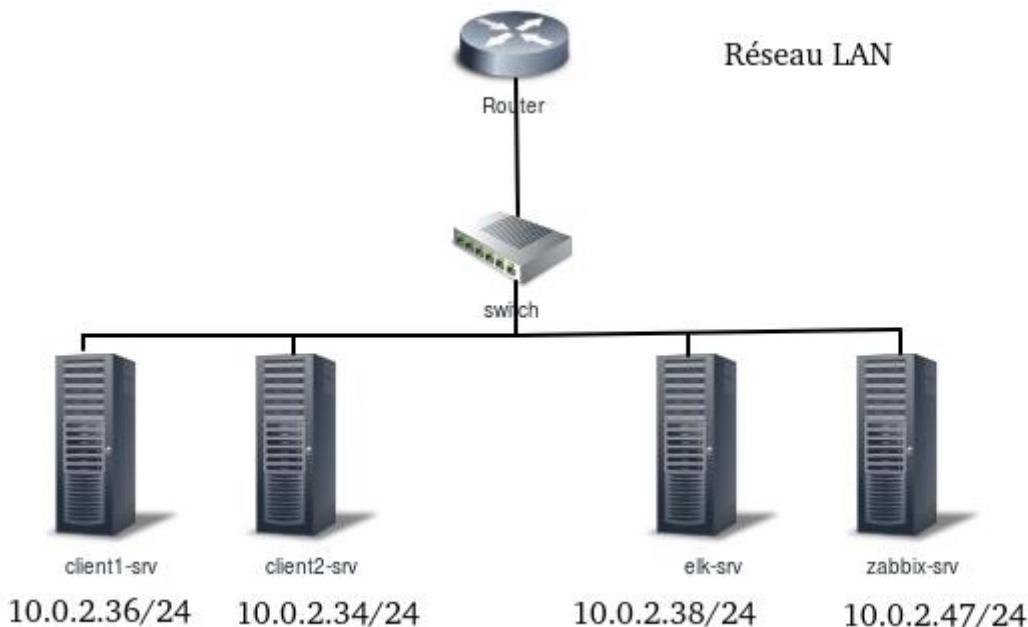


Figure 24 : Architecture du réseau du déploiement

- Serveur ELK : elk-srv, adresse privée : 10.0.2.38/24 – CentOS 7
- Serveur Client : client1-srv, adresse privée : 10.0.2.36/24 – CentOS 7
- Serveur Client : client2-srv, adresse privée : 10.0.2.34/24 – CentOS 7
- Serveur Zabbix : zabbix-srv, adresse privée : 10.0.2.47/24 – CentOS 7

4.2- Désactivation de SELinux

Nous allons désactiver SELinux sur le serveur ELK :

```
vi /etc/sysconfig/selinux
```

Modification de la valeur de SELinux en 'disabled' :

```
SELINUX=disabled
```

Nous redémarrons le serveur :

```
reboot
```

Vérification de l'état SELinux après redémarrage :

```
getenforce
```

Le résultat doit être désactivé.

4.3- Installation de Java

Java est requis pour le déploiement de la pile Elastic. Elasticsearch nécessite Java 8.

Téléchargement et installation de Java 8 :

```
yum -y install java-1.8.0 wget
```

Vérification de la version de Java :

```
java -version
```

4.4- Installation et configuration d'Elasticsearch

Avant d'installer Elasticsearch, nous ajoutons de la clé elastic.co au serveur :

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Téléchargement et installation d'Elasticsearch 6.7.1 :

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.7.1.rpm
```

```
rpm -ivh elasticsearch-6.7.1.rpm
```

Configuration du fichier elasticsearch.yml :

```
vi /etc/elasticsearch/elasticsearch.yml
```

Activation du verrouillage de la mémoire pour Elasticsearch (cela désactive l'échange de mémoire pour Elasticsearch) :
bootstrap.memory_lock: true

Dans le bloc "Network", nous enlevons les commentaires des lignes network.host et http.port :

```
network.host: localhost  
http.port: 9200
```

Nous éditons le fichier de configuration sysconfig pour Elasticsearch :

```
vi /etc/sysconfig/elasticsearch
```

Nous enlevons le commentaire de la ligne MAX_LOCKED_MEMORY :

```
MAX_LOCKED_MEMORY=unlimited
```

Activation et démarrage du service de Elasticsearch:

```
systemctl daemon-reload  
systemctl enable elasticsearch  
systemctl start elasticsearch
```

Vérification :
netstat -plntu

L'état du port 9200 doit être "LISTEN".

ou bien en utilisant la commande curl :

```
curl -XGET 'localhost:9200/?pretty'
```

```
[root@elk-srv elks]# curl -XGET 'localhost:9200/?pretty'  
{  
  "name" : "4N7yjeV",  
  "cluster_name" : "elasticsearch",  
  "cluster_uuid" : "6lhF0QWYRpCJxhAKTj3fig",  
  "version" : {  
    "number" : "6.7.1",  
    "build_flavor" : "default",  
    "build_type" : "rpm",  
    "build_hash" : "2f32220",  
    "build_date" : "2019-04-02T15:59:27.961366Z",  
    "build_snapshot" : false,  
    "lucene_version" : "7.7.0",  
    "minimum_wire_compatibility_version" : "5.6.0",  
    "minimum_index_compatibility_version" : "5.0.0"  
  },  
  "tagline" : "You Know, for Search"  
}
```

ou bien :

systemctl status elasticsearch

```
[root@elk-srv elksrv]# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; vendor preset: disabled)
   Active: active (running) since sam. 2019-06-15 01:09:50 WEST; 21min ago
     Docs: http://www.elastic.co
 Main PID: 15526 (java)
    Tasks: 60
   CGroup: /system.slice/elasticsearch.service
           ├─15526 /bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=...
           └─15602 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

juin 15 01:09:50 elk-srv systemd[1]: Started Elasticsearch.
```

4.5- Installation et configuration de Kibana avec Nginx

4.5.1- Installation et configuration de Kibana

Kibana écoute l'adresse IP de l'hôte local et Nginx agit en tant que proxy inverse pour l'application Kibana.

Téléchargement et installation de Kibana 6.7.1 :

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-6.7.1-x86_64.rpm
```

```
rpm -ivh kibana-6.7.1-x86_64.rpm
```

Nous éditons le fichier de configuration Kibana :

```
vi /etc/kibana/kibana.yml
```

Nous enlevons les commentaires des lignes de configuration pour server.port, server.host et elasticsearch.hosts :

```
server.port: 5601
server.host: "localhost"
elasticsearch.hosts: ["http://localhost:9200"]
```

Activation et démarrage du service de Kibana :

```
systemctl enable kibana
systemctl start kibana
```

Vérification (Kibana sera exécuté sur le port 5601) :

netstat -plntu

ou bien :

systemctl status kibana

```
[root@elk-srv elksrv]# systemctl status kibana
● kibana.service - Kibana
  Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: disabled)
  Active: active (running) since sam. 2019-06-15 01:32:07 WEST; 49s ago
    Main PID: 17256 (node)
      Tasks: 11
     CGroup: /system.slice/kibana.service
             └─17256 /usr/share/kibana/bin/../node/bin/node --no-warnings --max-http-header-size=65536 /...
[...]
juin 15 01:32:31 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:31Z","tags": [...ch"]}
juin 15 01:32:31 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:31Z","tags": [...on]}
juin 15 01:32:31 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:31Z","tags": [...ch]}
juin 15 01:32:31 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:31Z","tags": [...ch]}
juin 15 01:32:31 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:31Z","tags": [...ve]}
juin 15 01:32:33 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:33Z","tags": [...n."]}
juin 15 01:32:33 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:33Z","tags": [...ml"]}
juin 15 01:32:34 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:34Z","tags": [...ed"]}
juin 15 01:32:36 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:36Z","tags": [...01"]}
juin 15 01:32:36 elk-srv kibana[17256]: {"type":"log","@timestamp":"2019-06-15T00:32:36Z","tags": [...ch"]}
Hint: Some lines were ellipsized, use -l to show in full.
```

4.5.2- Installation et configuration de Nginx

Nginx est disponible dans le package "epel" :

yum -y install epel-release

Installation des packages Nginx et httpd-tools :

yum -y install nginx httpd-tools

Le paquet httpd-tools contient des outils pour le serveur Web.

Nous éditons le fichier de configuration Nginx et supprimons le bloc 'server {}' pour pouvoir ajouter une nouvelle configuration d'hôte virtuel :

vi /etc/nginx/nginx.conf

Nous créons un nouveau fichier de configuration d'hôte virtuel dans le répertoire conf.d :

vi /etc/nginx/conf.d/elk-srv.conf

Configuration à y ajouter :

=====

```

server {

    listen 80;
    server_name elk-srv;

    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/.kibana-user;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
=====
```

Création d'un nouveau fichier d'authentification :

```
htpasswd -c /etc/nginx/.kibana-user admin
TYPE_PASSWORD
```

Test de la configuration de Nginx et démarrage du service :

```
nginx -t
systemctl enable nginx
systemctl start nginx
Vérification :
systemctl status nginx
```

```
[root@elk-srv elksrv]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since sam. 2019-06-15 01:26:20 WEST; 4min 23s ago
     Main PID: 16690 (nginx)
        Tasks: 3
       CGroup: /system.slice/nginx.service
               └─16690 nginx: master process /usr/sbin/nginx
                  ├─16691 nginx: worker process
                  ├─16692 nginx: worker process
                  └─16693 nginx: worker process

juin 15 01:26:20 elk-srv systemd[1]: Starting The nginx HTTP and reverse proxy server...
juin 15 01:26:20 elk-srv nginx[16685]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
juin 15 01:26:20 elk-srv nginx[16685]: nginx: configuration file /etc/nginx/nginx.conf test is successful
juin 15 01:26:20 elk-srv systemd[1]: Started The nginx HTTP and reverse proxy server.
Hint: Some lines were ellipsized, use -l to show in full.
```

4.6- Installation et configuration de Logstash

Téléchargement et installation de Logstash :

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.7.1.rpm  
rpm -ivh logstash-6.7.1.rpm
```

Nous générerons un nouveau fichier de certificat SSL afin que le client puisse identifier le serveur ELK :

Nous générerons le fichier de certificat avec la commande openssl :

```
openssl req -config /etc/pki/tls/openssl.cnf -x509 -days 3650 -batch -nodes -newkey  
rsa:2048 -keyout /etc/pki/tls/private/logstash-forwarder.key -out  
/etc/pki/tls/certs/logstash-forwarder.crt -subj /CN=elk-srv
```

Nous allons créer un nouveau fichier de configuration 'cfglog.conf' dans le sous-répertoire 'conf.d' pour configurer les syslogs provenant des machines clients, le traitement des syslog et pour définir la sortie Elasticsearch.

Important : pour que l'utilisateur "logstash" puisse procéder à l'analyse du fichier CSV : "zbx_problems_export.csv", il doit avoir au moins le droit d'exécution sur tous les sous-répertoires où ce dernier se situe, le droit de lecture sur le dernier sous-répertoire et être le propriétaire du fichier.

Le chemin de notre fichier est : "/home/elksrv/zbx_problems_export.csv". Les droits d'accès doivent être comme ci-dessous :

```
[root@elk-srv elksrv]# ls -l / | grep 'home'  
drwxr-xr-x. 3 root root 20 11 juin 22:11 home  
[root@elk-srv elksrv]# ls -l /home  
total 4  
drwx---r-x. 18 elksrv elksrv 4096 17 juin 15:01 elksrv  
[root@elk-srv elksrv]# ls -l zbx_problems_export.csv  
-rwxr-xrwx 1 logstash logstash 554 15 juin 03:05 zbx_problems_export.csv
```

Figure 25 : Droits d'accès

Remarque : nous supprimons l'entête du fichier CSV.

```
vi /etc/logstash/conf.d/cfglog.conf
```

```
=====  
input {  
  beats {  
    port => 5044
```

```

ssl => true
ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
}

beats {
  port => 5045
  ssl => true
  ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
  ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
}

file {
  path => ["/home/elksrv/zbx_problems_export.csv"]
  start_position => "beginning"
  sincedb_path => "/dev/null"
  tags => "Zabbix-logs"
}
}

filter {

  if ("Messages-logs" or "Secure-logs" or "Suricata-logs" or "Zabbix-logs" in [tags])
  {
    mutate{
      add_field => [ "event_received_date", "%{ @timestamp }" ]
    }
  }

  if "Messages-logs" or "Secure-logs" in [tags] {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}%{SYSLOGHOST:syslog_hostname}%{DATA:syslog_program}(?:\[ %{POSINT:syslog_pid}\])?:%{GREEDYDATA:syslog_message}" }
    }
  }

  date {
    match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
  }
}

if "Suricata-logs" in [tags] {
  date {

```

```

match => [ "timestamp", "ISO8601" ]
}

ruby {
code => "
if event.get('event_type') == 'fileinfo'
  event.set('[fileinfo][type]', event.get('[fileinfo][magic]').to_s.split(',')[0])
end
"
}
}

if [http] {
useragent {
  source => "[http][http_user_agent]"
  target => "[http][user_agent]"
}
}

if [src_ip] {
geoip {
  source => "src_ip"
  target => "geoip"
}
}

if !#[geoip.ip]{
  if [dest_ip] {
    geoip {
      source => "dest_ip"
      target => "geoip"
    }
  }
}
}

if "Zabbix-logs" in [tags] {
csv {
  skip_empty_columns => true
  separator => ","
  columns => ["Severity", "Time", "Recovery time", "Status", "Host", "Problem",
"Duration", "Ack", "Actions"]
}
date {
  match => [ "Time" , "yyyy-MM-dd HH:mm:ss" ]
}
}

```

```

        }
        fingerprint {
            key => "965210MIAE"
            source => ["message"]
            target => "Fingerprint"
            method => "SHA256"
            concatenate_sources => true
        }
    }
}

output {
    if "Zabbix-logs" not in [tags] {
        elasticsearch {
            hosts => ["localhost:9200"]
            index => "logstash-%{+YYYY.MM.dd}"
        }
    } else {
        elasticsearch {
            hosts => ["localhost:9200"]
            index => "zabbix-%{+YYYY.MM.dd}"
            document_id => "%{Fingerprint}"
        }
    }
}
=====
```

Nous configurons le pare-feu pour que Logstash reçoive les syslogs des machines clients :

```
firewall-cmd --permanent --add-port=5044/tcp
firewall-cmd --permanent --add-port=5045/tcp
firewall-cmd --reload
```

Activation et démarrage du service de logstash :

```
systemctl enable logstash
systemctl start logstash
```

Vérification :

```
systemctl status logstash
```

```
[root@elk-srv elksrv]# systemctl status logstash
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: disabled)
   Active: active (running) since sam. 2019-06-15 01:30:13 WEST; 5s ago
     Main PID: 17126 (java)
        Tasks: 14
       CGroup: /system.slice/logstash.service
                  └─17126 /bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=...
```

```
juin 15 01:30:13 elk-srv systemd[1]: Started logstash.
```

ou bien :

netstat -plntu

Proto	Recv-Q	Send-Q	Adresse locale	Adresse distante	Etat	PID/Program name
Connexions Internet actives (seulement serveurs)						
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	1/systemd
tcp	0	0	0.0.0.0:80	0.0.0.0:*	LISTEN	5155/nginx: master
tcp	0	0	0.0.0.0:6000	0.0.0.0:*	LISTEN	3345/X
tcp	0	0	192.168.122.1:53	0.0.0.0:*	LISTEN	5533/dnsmasq
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	3493/master
tcp	0	0	127.0.0.1:5601	0.0.0.0:*	LISTEN	2753/node
tcp6	0	0	:::111	:::*	LISTEN	1/systemd
tcp6	0	0	:::1:9200	:::*	LISTEN	3231/java
tcp6	0	0	127.0.0.1:9200	:::*	LISTEN	3231/java
tcp6	0	0	:::6000	:::*	LISTEN	3345/X
tcp6	0	0	:::5044	:::*	LISTEN	2750/java
tcp6	0	0	:::1:9300	:::*	LISTEN	3231/java
tcp6	0	0	127.0.0.1:9300	:::*	LISTEN	3231/java
tcp6	0	0	:::5045	:::*	LISTEN	2750/java
tcp6	0	0	:::1:25	:::*	LISTEN	3493/master
tcp6	0	0	127.0.0.1:9600	:::*	LISTEN	2750/java
udp	0	0	192.168.122.1:53	0.0.0.0:*		5533/dnsmasq
udp	0	0	0.0.0.0:67	0.0.0.0:*		5533/dnsmasq
udp	0	0	0.0.0.0:111	0.0.0.0:*		1/systemd
udp	0	0	0.0.0.0:799	0.0.0.0:*		2746/rpcbind
udp6	0	0	:::111	:::*		1/systemd
udp6	0	0	:::799	:::*		2746/rpcbind

4.7- Installation et configuration de Filebeat sur les deux serveurs Client : client1-srv et client2-srv

4.7.1- Serveur client1-srv

Conditions préalables :

Nous nous connectons au serveur ELK depuis le serveur Client afin d'importer le fichier de certificat :

ssh root@10.0.2.38

Nous copions le fichier de certificat sur le serveur Client :

```
scp /etc/pki/tls/certs/logstash-forwarder.crt root@10.0.2.36:/etc/pki/tls/certs/
```

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Téléchargement et installation de Filebeat avec rpm :

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.7.1-x86_64.rpm
```

```
rpm -ivh filebeat-6.7.1-x86_64.rpm
```

Nous éditons ensuite le fichier de configuration "filebeat.yml" :

```
vi /etc/filebeat/filebeat.yml
```

Dans la section "filebeat.inputs", nous ajoutons les nouveaux fichiers journaux. Nous allons ajouter trois fichiers '/var/log/messages', '/var/log/secure' et '/var/log/suricata/eve.json' :

```
=====
```

```
filebeat.inputs:
```

```
- type: log
  enabled : true
  paths:
    - /var/log/messages
  tags: ["Messages-logs"]
- type: log
  paths:
    - /var/log/secure
  tags: ["Secure-logs"]
- type: log
  paths:
    - /var/log/suricata/eve.json
  tags: ["Suricata-logs"]
  json.keys_under_root: true
  json.add_error_key: true
```

```
=====
```

Filebeat utilise Elasticsearch comme cible de sortie par défaut, nous allons le changer en Logstash. Nous allons donc désactiver la sortie Elasticsearch et activons la sortie Logstash :

```
=====
```

```
output.logstash:
```

```
  # The Logstash hosts
  hosts: ["elk-srv:5044"]
  ssl.certificateAuthorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]
```

```
=====
```

Nous ajoutons une entrée d'hôte (serveur elk-srv) sur la serveur client1-srv :

```
vi /etc/hosts
```

```
10.0.2.38 elk-srv elk-srv
```

Activation et démarrage du service de Filebeat :

systemctl enable filebeat

systemctl start filebeat

Vérification :

systemctl status filebeat

```
[root@client1-srv clt1srv]# systemctl status filebeat -l
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
   Loaded: loaded (/usr/lib/systemd/system/filebeat.service; enabled; vendor preset: disabled)
   Active: active (running) since lun. 2019-06-17 16:23:08 WEST; 2min 20s ago
     Docs: https://www.elastic.co/products/beats/filebeat
Main PID: 3151 (filebeat)
   Tasks: 12
      CGroup: /system.slice/filebeat.service
              └─3151 /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml -path.home /usr/share/filebeat -path.config /etc/filebeat -path.data /var/lib/filebeat -path.logs /var/log/filebeat

juin 17 16:23:08 client1-srv systemd[1]: Started Filebeat sends log files to Logstash or directly to Elasticsearch..
```

4.7.2- Serveur client2-srv

Conditions préalables :

Nous nous connectons au serveur ELK depuis le serveur Client afin d'importer le fichier de certificat :

ssh root@10.0.2.38

Nous copions le fichier de certificat sur le serveur Client :

scp /etc/pki/tls/certs/logstash-forwarder.crt root@10.0.2.34:/etc/pki/tls/certs/

Nous suivons la même procédure d'installation indiquée ci-dessus et nous activons la sortie logstash comme suit :

```
=====
output.logstash:
  # The Logstash hosts
  hosts: ["elk-srv:5045"]
  ssl.certificateAuthorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]
=====
```

Vérification :

systemctl status filebeat

```
[root@client2-srv clt2srv]# systemctl status filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
   Loaded: loaded (/usr/lib/systemd/system/filebeat.service; enabled; vendor preset: disabled)
   Active: active (running) since sam. 2019-06-15 00:56:45 WEST; 4min 41s ago
     Docs: https://www.elastic.co/products/beats/filebeat
Main PID: 15483 (filebeat)
   CGroup: /system.slice/filebeat.service
           └─15483 /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml -path.home /usr/share/fil...
juin 15 00:56:45 client2-srv systemd[1]: Started Filebeat sends log files to Logstash or directly to El...ch..
Hint: Some lines were ellipsized, use -l to show in full.
```

4.8- Installation et configuration de Suricata sur les deux serveurs Client : client1-srv et client2-srv

Pré-requis :

Installation du package "epel" :

```
yum -y install epel-release
```

Installation des dépendances sur CentOS 7 :

```
yum -y install gcc libpcap-devel pcre-devel libyaml-devel file-devel \
 zlib-devel jansson-devel nss-devel libcap-ng-devel libnet-devel tar make \
 libnetfilter_queue-devel lua-devel
```

Téléchargement et installation de Suricata :

```
wget http://www.openinfosecfoundation.org/download/suricata-4.1.4.tar.gz
tar -xvf suricata-4.1.4.tar.gz
cd suricata-4.1.4
```

Pour avoir la configuration dans '/etc/suricata' au lieu de '/usr/local/etc/' et avoir '/var/log/suricata' au lieu de '/usr/local/var/log/suricata' comme répertoire de Log, nous utilisons :

```
./configure --sysconfdir=/etc --localstatedir=/var
```

Installation :

```
make
```

```
make install
```

Installation des fichiers de configuration par défaut de Suricata :

```
make install-conf
```

Téléchargement et installation des règles IDS :

```
wget http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz  
tar zxvf emerging.rules.tar.gz  
cp -r rules /etc/suricata/
```

Vérification :

```
ls /etc/suricata/rules
```

Afin de créer le fichier /var/lib/suricata/rules/suricata.rules, nous exécutons la commande suivante depuis le sous-répertoire suricata-4.1.4 :

```
make install-rules
```

Configuration de Suricata :

Nous éditons le fichier de configuration suricata.yaml comme suit :

- Le mot clé "default-log-dir" doit pointer vers l'emplacement des fichiers journaux Suricata.
- Dans la section "vars" :
"HOME_NET" doit pointer sur le réseau local à inspecter par Suricata.
"! \$ HOME_NET" (attribué à EXTERNAL_NET) fait référence à tout autre réseau que le réseau local.
- Nous activons la sortie "dns".
- Nous activons l'option "force-magic" de la sortie "files".
- Nous activons la sortie "http.log" pour avoir le fichier de sortie "http.log" qui contiendra tous les logs HTTP (optionnel).
- L'IDS moderne a mis au point une inspection dite "ciblée", dans laquelle le moteur d'inspection adapte son algorithme de détection en fonction du système d'exploitation cible du trafic. Dans notre cas, nous allons activer "Linux" comme stratégie IDS par défaut. A cet effet, Suricata appliquera l'inspection basée sur Linux si aucune information de système d'exploitation n'est connue pour une adresse IP particulière.

```
vi /etc/suricata/suricata.yaml
```

```
default-log-dir: /var/log/suricata/
```

vars:

```
# more specific is better for alert accuracy and performance address-groups:
```

```
HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
```

```
#HOME_NET: "[192.168.1.0/24]"
```

```
#HOME_NET: "[10.0.0.0/8]"
```

```
#HOME_NET: "[172.16.0.0/12]"
```

```
#HOME_NET: "any"
```

```
EXTERNAL_NET: "!$HOME_NET"
```

```
#EXTERNAL_NET: "any"
```

port-groups:

```
HTTP_PORTS: "80"
```

```
SHELLCODE_PORTS: "!80"
```

```
ORACLE_PORTS: 1521
```

```
SSH_PORTS: 22
```

```
DNP3_PORTS: 20000
```

```
MODBUS_PORTS: 502
```

```
FILE_DATA_PORTS: "[${HTTP_PORTS},110,143]"
```

```
FTP_PORTS: 21
```

- dns:

```
# Enable/disable this logger. Default: enabled.
```

```
enabled: yes
```

- files:

```
force-magic: yes
```

- http-log:

```
enabled: true
```

```
fileneame: http.log
```

host-os-policy:

```
# Make the default policy windows.
```

```
windows: []
```

```
bsd: []
```

```
bsd-right: []
```

```
old-linux: []
```

```
linux: [0.0.0.0/0]
```

```
old-solaris: []
```

```
solaris: []
```

```
hpux10: []
hpux11: []
irix: []
macos: []
vista: []
windows2k3: []
=====
```

Test et exécution de Suricata :

Test des erreurs :

```
suricata -c /etc/suricata/suricata.yaml -i enp0s3 --init-errors-fatal
```

(avec enp0s3 comme nom d'interface).

Exécution de Suricata :

```
suricata -c /etc/suricata/suricata.yaml -i enp0s3
```

Sur le serveur client1-srv :

```
[root@client1-srv cltlsvr]# suricata -c /etc/suricata/suricata.yaml -i enp0s3
14/6/2019 -- 23:16:43 - <Notice> - This is Suricata version 4.1.4 RELEASE
14/6/2019 -- 23:16:47 - <Notice> - all 2 packet processing threads, 4 management threads initialized, engine started.
^C14/6/2019 -- 23:33:32 - <Notice> - Signal Received. Stopping engine.
14/6/2019 -- 23:33:33 - <Notice> - Stats for 'enp0s3': pkts: 113764, drop: 0 (0.00%), invalid checksum: 0
```

Sur le serveur client2-srv :

```
[root@client2-srv clt2srv]# suricata -c /etc/suricata/suricata.yaml -i enp0s3
15/6/2019 -- 00:07:11 - <Notice> - This is Suricata version 4.1.4 RELEASE
15/6/2019 -- 00:07:14 - <Notice> - all 2 packet processing threads, 4 management threads initialized, engine started.
^C15/6/2019 -- 00:22:57 - <Notice> - Signal Received. Stopping engine.
15/6/2019 -- 00:22:58 - <Notice> - Stats for 'enp0s3': pkts: 135903, drop: 0 (0.00%), invalid checksum: 0
```

Exécution de Suricata en tant que daemon :

```
suricata -D -c /etc/suricata/suricata.yaml -i enp0s3
```

Vérification:

```
tail -f /var/log/suricata/fast.log
tail -f /var/log/suricata/http.log
tail -f /var/log/suricata/eve.json
```

4.9- Installation et configuration de Zabbix sur le serveur : zabbix-srv

Le serveur Zabbix dépend des applications logicielles suivantes :

- Serveur web Apache
- PHP avec les extensions requises
- Serveur de base de données MySQL/MariaDB

Nous allons procéder à :

- Passation de SELinux en mode permissif sur le serveur Zabbix
- Installation d'Apache2/httpd
- Installation et configuration de PHP 7.2
- Installation et configuration de MariaDB
- Installation et configuration de Zabbix 4.0
- Configuration de Firewalld

4.9.1- Modification du fichier SELinux

```
vi /etc/sysconfig/selinux
```

Modification de la valeur de SELinux en 'permissive' :

```
SELINUX=permissive
```

Nous redémarrons le serveur :

```
reboot
```

Vérification de l'état SELinux après redémarrage :

```
getenforce
```

Le résultat doit être permissif.

4.9.2- Installation d'Apache2/httpd

Nous allons exécuter Zabbix sous le serveur Web Apache.

Installation d'Apache/httpd :

```
yum -y install httpd
```

Démarrage et activation du service "httpd" au démarrage du système :

```
systemctl start httpd  
systemctl enable httpd
```

Vérification :
netstat -plntu

L'état du port 80 doit être "LISTEN".

Ou bien :

```
systemctl status httpd
```

```
[root@zabbix-srv tstsrv]# systemctl status httpd  
● httpd.service - The Apache HTTP Server  
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)  
  Active: active (running) since lun. 2019-06-17 17:10:49 WEST; 11min ago  
    Docs: man:httpd(8)  
          man:apachectl(8)  
 Main PID: 3162 (httpd)  
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"  
     Tasks: 6  
    CGroup: /system.slice/httpd.service  
           ├─3162 /usr/sbin/httpd -DFOREGROUND  
           ├─5448 /usr/sbin/httpd -DFOREGROUND  
           ├─5451 /usr/sbin/httpd -DFOREGROUND  
           ├─5452 /usr/sbin/httpd -DFOREGROUND  
           ├─5453 /usr/sbin/httpd -DFOREGROUND  
           └─5454 /usr/sbin/httpd -DFOREGROUND  
  
juin 17 17:10:38 zabbix-srv systemd[1]: Starting The Apache HTTP Server...  
juin 17 17:10:45 zabbix-srv httpd[3162]: AH00558: httpd: Could not reliably determine the server's ful...sage  
juin 17 17:10:49 zabbix-srv systemd[1]: Started The Apache HTTP Server.  
Hint: Some lines were ellipsized, use -l to show in full.
```

4.9.3- Installation et configuration de PHP 7.2

Nous installerons PHP 7.2 à partir du package 'webtatic'.

Avant d'installer PHP 7 avec toutes les extensions nécessaires, nous ajoutons les packages 'Webtatic' et 'EPEL' au système :

```
yum -y install epel-release  
rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm
```

Installation des paquets PHP 7.2 à partir du package webtatic :

```
yum -y install mod_php72w php72w-cli php72w-common php72w-devel php72w-pear  
php72w-gd php72w-mbstring php72w-mysql php72w-xml php72w-bcmath
```

Nous éditons le fichier de configuration de PHP pour mettre à jour les paramètres de fuseau horaire et de PHP :

```
vi /etc/php.ini
```

```
max_execution_time = 600  
max_input_time = 600  
memory_limit = 256M  
post_max_size = 32M  
upload_max_filesize = 16M  
date.timezone = Africa/Casablanca
```

Nous redémarrons le service httpd :
systemctl restart httpd

4.9.4- Installation et configuration de MariaDB

Zabbix prend en charge de nombreuses bases de données pour l'installation, notamment les bases de données MySQL, PostgreSQL, SQLite et Oracle. Dans notre cas, nous utiliserons MariaDB comme base de données :

```
yum -y install mariadb-server
```

Démarrage et activation de "mariadb" au démarrage du système :

```
systemctl start mariadb  
systemctl enable mariadb
```

Vérification :

```
systemctl status mariadb
```

```
[root@zabbix-srv tstsrv]# systemctl status mariadb
● mariadb.service - MariaDB database server
  Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Active: active (running) since lun. 2019-06-17 17:10:55 WEST; 12min ago
    Process: 3407 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID (code=exited, status=0/SUCCESS)
   Process: 3148 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir %n (code=exited, status=0/SUCCESS)
 Main PID: 3404 (mysqld_safe)
   Tasks: 43
  CGroup: /system.slice/mariadb.service
          └─3404 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
                  ├─3657 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql...

juin 17 17:10:38 zabbix-srv systemd[1]: Starting MariaDB database server...
juin 17 17:10:41 zabbix-srv mariadb-prepare-db-dir[3148]: Database MariaDB is probably initialized in /...ne.
juin 17 17:10:41 zabbix-srv mariadb-prepare-db-dir[3148]: If this is not the case, make sure the /var/l...ir.
juin 17 17:10:41 zabbix-srv mysqld_safe[3404]: 190617 17:10:41 mysqld_safe Logging to '/var/log/mariad...og'.
juin 17 17:10:41 zabbix-srv mysqld_safe[3404]: 190617 17:10:41 mysqld_safe Starting mysqld daemon with...ysql
juin 17 17:10:55 zabbix-srv systemd[1]: Started MariaDB database server.
Hint: Some lines were ellipsized, use -l to show in full.
```

Par défaut, l'installation de MariaDB n'est pas sécurisée. Nous allons le faire en exécutant le script "mysql_secure_installation" :

mysql_secure_installation

Nous répondons aux questions suivantes :

```
=====
Enter current password for root (enter for none):
```

```
Set root password? [Y/n]: Y
```

```
New password:
```

```
Re-enter new password:
```

```
Remove anonymous users? [Y/n]: Y
```

```
Disallow root login remotely? [Y/n]: Y
```

```
Remove test database and access to it? [Y/n]: Y
```

```
Reload privilege tables now? [Y/n]: Y
```

```
=====
Le script ci-dessus définit le mot de passe root, supprime la base de données de test,  
supprime également les utilisateurs anonymes et interdit la connexion root depuis une  
machine distante. Ensuite, nous allons créer une nouvelle base de données pour notre  
installation de Zabbix et un utilisateur nommé 'zabbix' défini par le mot de passe  
'ZABBIX_USER_DB_PASSWORD'.
```

Connexion au shell MySQL :

```
mysql -u root -p
```

```
=====
MariaDB [(none)]> create database zabbix character set utf8 collate utf8_bin;  
MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@'localhost' identified  
by 'ZABBIX_USER_DB_PASSWORD';  
MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@'%' identified by  
'ZABBIX_USER_DB_PASSWORD';  
MariaDB [(none)]> flush privileges;  
MariaDB [(none)]> quit
```

4.9.5- Installation et configuration de Zabbix 4.0

1- Installation de Zabbix :

Nous allons installer Zabbix à partir du package officiel.

Ajout du package Zabbix au système CentOS 7 :

```
yum install -y https://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-release-4.0-  
1.el7.noarch.rpm
```

Installation du serveur et l'agent Zabbix :

```
yum -y install zabbix-get zabbix-server-mysql zabbix-web-mysql zabbix-agent
```

2- Import de l'exemple de la base de données 'Zabbix' :

Maintenant, nous allons importer l'exemple de base de données 'Zabbix' dans notre base de données 'zabbix'.

Dans le répertoire de documentation de Zabbix, nous extrayons le fichier 'Zabbix sql' :

```
cd /usr/share/doc/zabbix-server-mysql-4.0.9/  
gunzip create.sql.gz
```

Import de l'exemple de base de données dans notre base de données 'zabbix' :

```
mysql -u root -p zabbix < create.sql
```

3- configuration du serveur Zabbix :

Le serveur Zabbix est le processus central du système logiciel Zabbix. Nous allons configurer le serveur en éditant le fichier de configuration.

```
vi /etc/zabbix/zabbix_server.conf
```

```
=====  
DBHost=localhost  
DBPassword=ZABBIX_USER_DB_PASSWORD  
=====
```

Démarrage et activation du service au démarrage du système :

```
systemctl start zabbix-server  
systemctl enable zabbix-server
```

Vérification :

```
systemctl status zabbix-server
```

```
tstsrv@zabbix-srv:/home/tstsrv
Fichier Édition Affichage Rechercher Terminal Aide
● zabbix-server.service - Zabbix Server
  Loaded: loaded (/usr/lib/systemd/system/zabbix-server.service; enabled; vendor preset: disabled)
  Active: active (running) since lun. 2019-06-17 17:10:42 WEST; 14min ago
    Process: 3149 ExecStart=/usr/sbin/zabbix_server -c $CONFFILE (code=exited, status=0/SUCCESS)
   Main PID: 3637 (zabbix_server)
     Tasks: 34
    CGroup: /system.slice/zabbix-server.service
            ├─3637 /usr/sbin/zabbix_server -c /etc/zabbix/zabbix_server.conf
            ├─7022 /usr/sbin/zabbix_server: configuration syncer [synced configuration in 0.026503 sec, idle 6
0 sec
            ├─7023 /usr/sbin/zabbix_server: housekeeper [startup idle for 30 minutes]
            ├─7024 /usr/sbin/zabbix_server: timer #1 [updated 0 hosts, suppressed 0 events in 0.000485 sec, id
le 59 sec
            ├─7025 /usr/sbin/zabbix_server: http poller #1 [got 0 values in 0.000878 sec, idle 5 sec]
            ├─7026 /usr/sbin/zabbix_server: discoverer #1 [processed 0 rules in 0.000333 sec, idle 60 sec]
            ├─7027 /usr/sbin/zabbix_server: history syncer #1 [processed 0 values, 0 triggers in 0.000010 sec,
idle 1 sec
            ├─7028 /usr/sbin/zabbix_server: history syncer #2 [processed 1 values, 1 triggers in 0.254424 sec,
idle 1 sec
            ├─7029 /usr/sbin/zabbix_server: history syncer #3 [processed 0 values, 0 triggers in 0.000027 sec,
idle 1 sec
            ├─7030 /usr/sbin/zabbix_server: history syncer #4 [processed 0 values, 0 triggers in 0.000019 sec,
idle 1 sec
--Plus--
```

4- Configuration de l'agent Zabbix :

Nous allons installer l'agent Zabbix pour collecter des données sur le statut du serveur Zabbix lui-même :

```
vi /etc/zabbix/zabbix_agentd.conf
```

```
=====
Server=127.0.0.1
ServerActive=127.0.0.1
Hostname=Zabbix server
```

Remarque : le nom d'hôte enregistré dans le fichier de configuration de l'agent Zabbix sera utilisé comme nom d'hôte lors de la configuration de l'interface Web du serveur Zabbix.

Démarrage et activation du service au démarrage du système :

```
systemctl start zabbix-agent
systemctl enable zabbix-agent
```

Vérification :

```
systemctl status zabbix-agent
```

```
[root@zabbix-srv tstsrv]# systemctl status zabbix-agent
● zabbix-agent.service - Zabbix Agent
   Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; enabled; vendor preset: disabled)
   Active: active (running) since lun. 2019-06-17 17:10:39 WEST; 15min ago
     Process: 3151 ExecStart=/usr/sbin/zabbix_agentd -c $CONFFILE (code=exited, status=0/SUCCESS)
    Main PID: 3167 (zabbix_agentd)
      Tasks: 6
     CGroup: /system.slice/zabbix-agent.service
             └─3167 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
               ├─3179 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
               ├─3180 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
               ├─3181 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
               ├─3182 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
               └─3183 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]

juin 17 17:10:38 zabbix-srv systemd[1]: Starting Zabbix Agent...
juin 17 17:10:39 zabbix-srv systemd[1]: PID file /run/zabbix/zabbix_agentd.pid not readable (yet?) after start
juin 17 17:10:39 zabbix-srv systemd[1]: Started Zabbix Agent.
Hint: Some lines were ellipsized, use -l to show in full.
```

4.9.6- Configuration de Firewalld

Dans cette étape, nous allons ouvrir les ports HTTP et HTTPS pour l'interface utilisateur Web de l'administrateur Zabbix et ajouter un port supplémentaire pour le serveur et l'agent Zabbix :

```
firewall-cmd --add-service={http,https} --permanent
firewall-cmd --add-port={10051/tcp,10050/tcp} --permanent
firewall-cmd --reload
```

Vérification :

```
firewall-cmd --list-all
```

```
[root@zabbix-srv tstsrv]# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpcv6-client http https
  ports: 10051/tcp 10050/tcp 3000/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Redémarrage de tous les services est requis :

```
systemctl restart zabbix-server
systemctl restart zabbix-agent
systemctl restart httpd
```

4.10- Installation et configuration de Zabbix-Agent sur les deux serveurs Client : client1-srv et client2-srv

4.10.1- Serveur client1-srv

Ajout du package Zabbix au système CentOS 7 :

```
yum install -y https://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-release-4.0-1.el7.noarch.rpm
```

Installation du serveur et l'agent Zabbix :

```
yum -y install zabbix-agent
```

Nous allons configurer l'agent Zabbix pour qu'il communique avec le serveur Zabbix en modifiant son fichier de configuration :

```
vi /etc/zabbix/zabbix_agentd.conf
```

```
=====
Server=10.0.2.47
ServerActive=10.0.2.47
Hostname=client1-srv
=====
```

Remarque : le nom d'hôte enregistré dans le fichier de configuration de l'agent Zabbix sera utilisé comme nom d'hôte lors de la création de ce dernier sur l'interface Web du serveur Zabbix.

Configuration du Firewalld :

```
firewall-cmd --permanent --add-port=10050/tcp
firewall-cmd --reload
```

Démarrage et activation du service au démarrage du système :

```
systemctl start zabbix-agent
systemctl enable zabbix-agent
```

Vérification :

```
systemctl status zabbix-agent
```

```
[root@client1-srv clt1srv]# systemctl status zabbix-agent
● zabbix-agent.service - Zabbix Agent
  Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; enabled; vendor preset: disabled)
  Active: active (running) since lun. 2019-06-17 17:31:45 WEST; 2min 18s ago
    Process: 3126 ExecStart=/usr/sbin/zabbix_agentd -c $CONFFILE (code=exited, status=0/SUCCESS)
   Main PID: 3148 (zabbix_agentd)
     Tasks: 6
    CGroup: /system.slice/zabbix-agent.service
            └─3148 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
                ├─3149 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
                ├─3150 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
                ├─3151 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
                ├─3152 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
                └─3153 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]

juin 17 17:31:44 client1-srv systemd[1]: Starting Zabbix Agent...
juin 17 17:31:45 client1-srv systemd[1]: Started Zabbix Agent.
```

4.10.2- Serveur client2-srv

Nous suivons la même procédure d'installation indiquée ci-dessus et nous modifions le fichier de l'agent Zabbix comme suit :

```
=====
Server=10.0.2.47
ServerActive=10.0.2.47
Hostname=client2-srv
=====
```

Vérification :

`systemctl status zabbix-agent`

```
[root@client2-srv clt2srv]# systemctl status zabbix-agent
● zabbix-agent.service - Zabbix Agent
  Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; enabled; vendor preset: disabled)
  Active: active (running) since lun. 2019-06-17 17:34:57 WEST; 2min 21s ago
    Process: 3152 ExecStart=/usr/sbin/zabbix_agentd -c $CONFFILE (code=exited, status=0/SUCCESS)
   Main PID: 3170 (zabbix_agentd)
     Tasks: 6
    CGroup: /system.slice/zabbix-agent.service
            └─3170 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
                ├─3171 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
                ├─3172 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
                ├─3173 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
                ├─3174 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
                └─3175 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]

juin 17 17:34:56 client2-srv systemd[1]: Starting Zabbix Agent...
juin 17 17:34:57 client2-srv systemd[1]: Started Zabbix Agent.
```

4.11- Installation et configuration de Grafana sur le serveur zabbix-srv

Installation de Grafana via le package YUM :

Nous créons un fichier 'repo' :

```
vi /etc/yum.repos.d/grafana.repo
```

Nous ajoutons le contenu suivant au fichier :

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

```
yum -y install grafana
```

Installation des paquets de polices supplémentaires :

```
yum -y install fontconfig
```

```
yum -y install freetype*
```

```
yum -y install urw-fonts
```

Configuration du Firewalld :

Grafana écoute sur le port 3000/tcp :

```
firewall-cmd --add-port=3000/tcp --permanent
firewall-cmd --reload
```

Activation et démarrage du service de Grafana :

```
systemctl start grafana-server
systemctl enable grafana-server
```

Vérification :

```
systemctl status grafana-server
```

```
[root@zabbix-srv tstsrv]# systemctl status grafana-server
● grafana-server.service - Grafana instance
  Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; vendor preset: disabled)
  Active: active (running) since lun. 2019-06-17 17:10:59 WEST; 17min ago
    Docs: http://docs.grafana.org
 Main PID: 5995 (grafana-server)
   CGroup: /system.slice/grafana-server.service
           └─5995 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/var/run/grafana/gr...

juin 17 17:10:59 zabbix-srv grafana-server[5995]: t=2019-06-17T17:10:59+0100 lvl=info msg="Initializing...ver
juin 17 17:10:59 zabbix-srv systemd[1]: Started Grafana instance.
juin 17 17:10:59 zabbix-srv grafana-server[5995]: t=2019-06-17T17:10:59+0100 lvl=info msg="Initializing...er"
juin 17 17:10:59 zabbix-srv grafana-server[5995]: t=2019-06-17T17:10:59+0100 lvl=info msg="HTTP Server ...et=
Hint: Some lines were ellipsized, use -l to show in full.
```

Installation du plug-in Zabbix :

```
grafana-cli plugins list-remote | egrep "zabbix"
```

```
grafana-cli plugins install alexanderzobnin-zabbix-app
```

```
[root@zabbix-srv tstsrv]# grafana-cli plugins install alexanderzobnin-zabbix-app
installing alexanderzobnin-zabbix-app @ 3.10.2
from url: https://grafana.com/api/plugins/alexanderzobnin-zabbix-app/versions/3.10.2/download
into: /var/lib/grafana/plugins

✓ Installed alexanderzobnin-zabbix-app successfully

Restart grafana after installing plugins . <service grafana-server restart>
```

Le répertoire d'installation du plugin par défaut est /var/lib/grafana/plugins. Nous redémarrons le service Grafana :

```
systemctl restart grafana-server
```

Vérification :

```
netstat -plntu
```

Proto	Recv-Q	Send-Q	Adresse locale	Adresse distante	Etat	PID/Program name
Connexions Internet actives (seulement serveurs)						
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	3632/mysqld
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	1/systemd
tcp	0	0	0.0.0.0:6000	0.0.0.0:*	LISTEN	3271/X
tcp	0	0	192.168.122.1:53	0.0.0.0:*	LISTEN	5736/dnsmasq
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	3677/master
tcp	0	0	0.0.0.0:10050	0.0.0.0:*	LISTEN	3182/zabbix agentd
tcp	0	0	0.0.0.0:10051	0.0.0.0:*	LISTEN	3575/zabbix server
tcp6	0	0	:::111	:::*	LISTEN	1/systemd
tcp6	0	0	:::80	:::*	LISTEN	3169/httpd
tcp6	0	0	:::6000	:::*	LISTEN	3271/X
tcp6	0	0	:::3000	:::*	LISTEN	6445/grafana-server
tcp6	0	0	:::1:25	:::*	LISTEN	3677/master
tcp6	0	0	:::10050	:::*	LISTEN	3182/zabbix agentd
tcp6	0	0	:::10051	:::*	LISTEN	3575/zabbix server
udp	0	0	192.168.122.1:53	0.0.0.0:*		5736/dnsmasq
udp	0	0	0.0.0.0:67	0.0.0.0:*		5736/dnsmasq
udp	0	0	0.0.0.0:111	0.0.0.0:*		1/systemd
udp	0	0	0.0.0.0:748	0.0.0.0:*		2711/rpcbind
udp6	0	0	:::111	:::*		1/systemd
udp6	0	0	:::748	:::*		2711/rpcbind

4.12- Configuration initiale et test de la pile ELK

Dans un navigateur Web et nous saisissons l'URL suivant : <http://localhost:5601>

Nous créons deux index "zabbix-" et "logstash-", définissons le "Time Filter" :

Kibana

localhost:5601/app/kibana#/management/kibana/index?_g=(refreshInterval)

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross Cluster Replication
- Remote Clusters
- License Management
- 7.0 Upgrade Assistant

Create index...

No default index pattern. You must select or create one to continue.

Kibana

Index Patterns

- Saved Objects
- Spaces
- Reporting
- Advanced Settings

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch Indices for things like visualizations.

Include system indices

Step 1 of 2: Define index pattern

Index pattern

zabbix*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

> Next step

✓ Success! Your index pattern matches 6 indices.

zabbix-2019.06.15

zabbix-2019.06.16

zabbix-2019.06.17

zabbix-2019.06.18

zabbix-2019.06.19

zabbix-2019.06.23

Kibana

localhost:5601/app/kibana#/management/kibana/index?_g=(refreshInterval)

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross Cluster Replication
- Remote Clusters
- License Management
- 7.0 Upgrade Assistant

Create index...

No default index pattern. You must select or create one to continue.

Kibana

Index Patterns

- Saved Objects
- Spaces
- Reporting
- Advanced Settings

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch Indices for things like visualizations.

Include system indices

Step 2 of 2: Configure settings

You've defined zabbix-* as your index pattern. Now you can specify some settings before we create it.

Time Filter field name

@timestamp

Refresh

The Time Filter will use this field to filter your data by time.
You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

> Show advanced options

< Back

Create Index pattern

The screenshot shows the Kibana interface for managing Elasticsearch indices. On the left, there's a sidebar with various icons and sections like 'Elasticsearch' and 'Kibana'. In the main area, a blue button labeled 'Create Inde...' is visible. Below it, a section titled '★ zabbix-*' is shown, with a note about a 'Time Filter field name: @timestamp'. A table lists fields with their types and search/aggregation capabilities. The table has columns for Name, Type, For..., Sear..., Aggr..., and Excl... . Several fields are listed, including '@timestamp', '@version', '@version.keyword', 'Ack', 'Ack.keyword', and 'Duration'. At the bottom right of the table, there are edit icons for each row.

Name	Type	For...	Sear...	Aggr...	Excl...
@timestamp	date	•	•		
@version	string	•			
@version.keyword	string	•	•		
Ack	string	•			
Ack.keyword	string	•	•		
Duration	string	•			

L'index a été créé.

Nous créons le deuxième index "logstash-" :

The screenshot shows the Kibana management interface at [localhost:5601/app/kibana#/management/kibana/index?_g=\(refreshInterval](http://localhost:5601/app/kibana#/management/kibana/index?_g=(refreshInterval). On the left, a sidebar lists 'Elasticsearch' and 'Kibana' sections. In the main area, a 'Create index...' button is visible above a search bar containing 'zabbix-*'. A success message indicates 'Your index pattern matches 2 indices.' with results for 'logstash-2019.06.24' and 'logstash-2019.06.25'. A 'Next step' button is on the right.

The screenshot shows the Kibana management interface at localhost:5601/app/kibana#/management/kibana/indices/c04fb10-96e1-11. It displays the 'logstash-*' index details. The 'Fields (57)' tab is selected, showing a table of fields with columns: Name, Type, For..., Sear..., Aggr..., and Excl... . Key fields listed include @timestamp, @version, _id, _index, _score, and _source.

Name	Type	For...	Sear...	Aggr...	Excl...
@timestamp	date	●	●		
@version	string	●	●		
_id	string	●	●		
_index	string	●	●		
_score	number				
_source					

Nous choisissons l'index "logstash-" comme index par défaut.

Dans le menu "Discover", nous verrons tous les syslogs des serveurs client1-srv et client2-srv ainsi que les événements collectés par Suricata et Zabbix.

The screenshot shows the Kibana Discover interface with the following details:

- Header:** Discover - Kibana, localhost:5601/app/kibana#discover?_g=(refreshInterval:(pause:1t,value:0))
- Left Sidebar:** kibana, Canvas, Maps, Machine Learning, Infrastructure, Logs, APM, Uptime, Dev Tools, Monitoring, Management, Default, Collapse.
- Top Bar:** 54,265 hits, New, Save, Open, Share, Inspect, Auto-refresh, Last 30 days.
- Search Bar:** Search... (e.g. status:200 AND extension:PHP)
- Filter Bar:** Add a filter +, logstash-* selected.
- Time Range:** June 2nd 2019, 03:12:01.786 - July 2nd 2019, 03:12:01.786, Auto.
- Chart:** Count vs @timestamp per 12 hours, showing a peak around June 23rd at approximately 35,000 events.
- Table:** Time, _source

▶ July 2nd 2019, 02:59:54.000	_source
	syslog_message: Started Virtualization daemon. log.file. path: /var/log/messages syslog_timestamp: Jul 2 02:59:54 source: /var/log/messages offset: 2,783,577 tags: Messages-logs, beats_input_codec_plain_applied prospector.type: log message: Jul 2 02:59:54 client2-srv systemd: Starte
▶ July 2nd 2019, 02:59:54.000	
	syslog_message: Started Virtualization daemon. log.file. path: /var/log/messages syslog_timestamp: Jul 2 02:59:54 source: /var/log/messages offset: 2,783,577 tags: Messages-logs, beats_input_codec_plain_applied prospector.type: log message: Jul 2 02:59:54 client2-srv systemd: Starte

Exemple de sortie JSON d'un événement collecté par Zabbix.

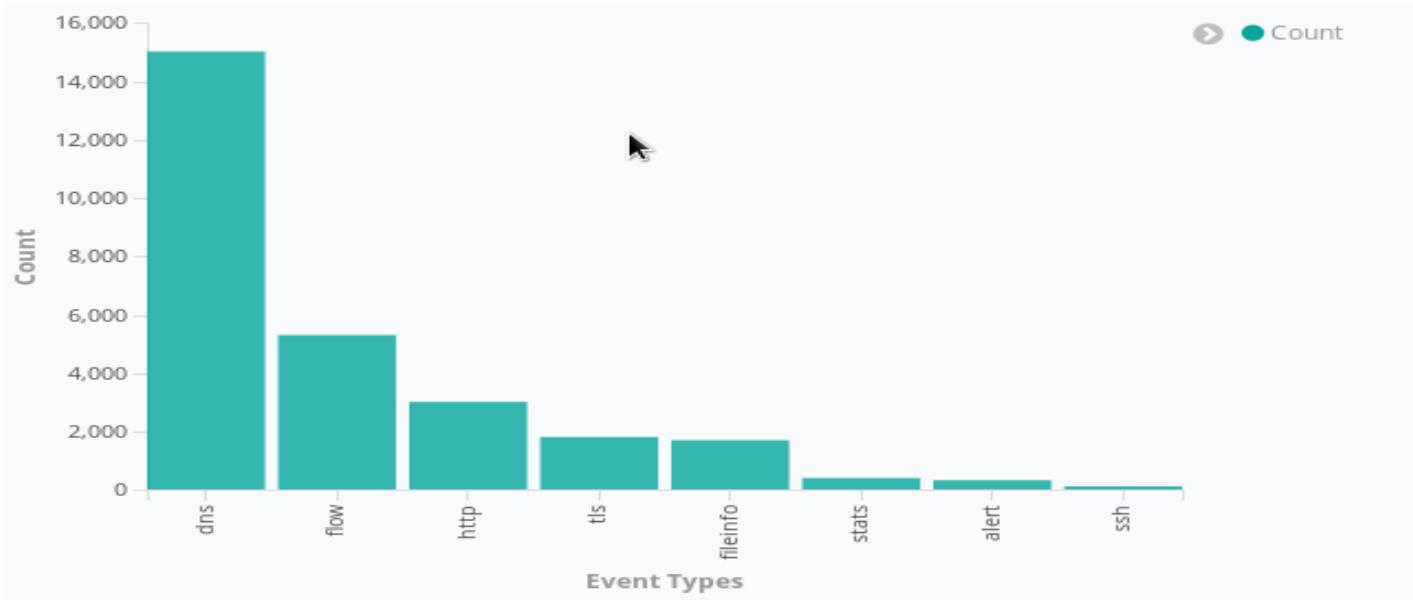
The screenshot shows the Kibana Visualize interface with the following details:

- Left Sidebar:** kibana, Discover, Visualize, Dashboard, Timelion, Canvas, Maps, Machine Learning, Infrastructure, Logs, APM, Uptime, Dev Tools, Monitoring, Management, Default, Collapse.
- Table View:** Table, JSON

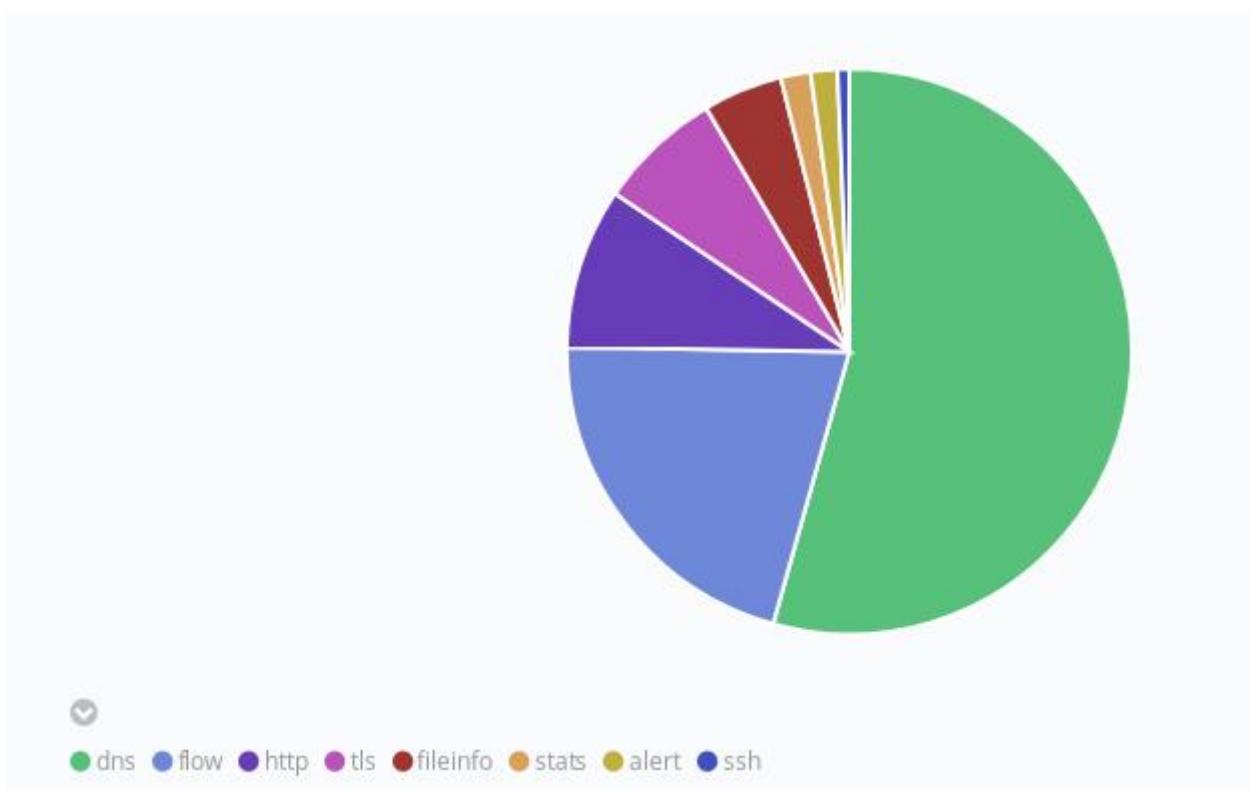
	Table	JSON	View surrounding documents	View single document
t Status	@timestamp	June 23rd 2019, 16:00:50.000		
t Time	@version	1		
t _id	Ack	No		
t _index	Duration	1m		
# _score	Fingerprint	c10a21035f2a75044d82ee140bf6165da8dbdb260c200c60bd7426c905abf057		
t _type	Host	Zabbix server		
event_rec...	Problem	Zabbix unreachable poller processes more than 75% busy		
t host	Recovery time	2019-06-23 16:01:50		
t message	Severity	Average		
t path	Status	RESOLVED		
t tags	Time	2019-06-23 16:00:50		
	_id	c10a21035f2a75044d82ee140bf6165da8dbdb260c200c60bd7426c905abf057		
	_index	zabbix-2019.06.23		
	# _score	-		
	_type	doc		
	event_received_date	June 24th 2019, 15:15:25.902		
	host	elk-srv		
	message	"Average", "2019-06-23 16:00:50", "2019-06-23 16:01:50", "RESOLVED", "Zabbix server", "Zabbix unreachable poller processes more than 75% busy", "1m", "No", "", "		
	path	/home/elksrv/zbx_problems_export.csv		

Sur Kibana, nous pouvons créer des visualisations, des tableaux de bord, etc...

Exemple d'une visualisation de types d'événements collectés par Suricata (Vertical Bar) :



Exemple d'une visualisation de types d'événements collectés par Suricata (Pie) :



Exemple d'une visualisation (Timelion), événements HTTP en rouge et tous les autres événements en bleu :

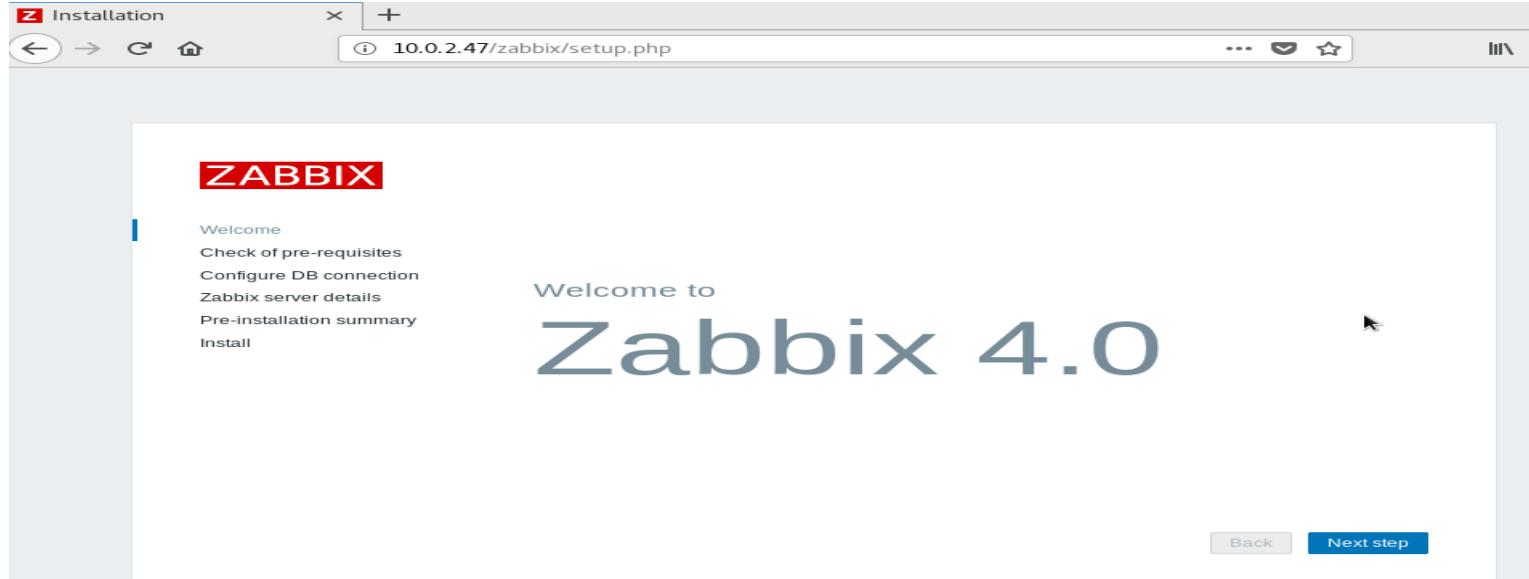
Exemple d'un tableau de bord d'événements collectés par Suricata :



4.13- Configuration initiale et test du serveur Zabbix

Dans un navigateur Web et nous saisissons l'URL suivant : <http://10.0.2.47/zabbix/>

Nous serons redirigés vers la page d'accueil de Zabbix :



Check of pre-requisites

	Current value	Required	
PHP version	7.2.17	5.4.0	OK
PHP option "memory_limit"	256M	128M	OK
PHP option "post_max_size"	32M	16M	OK
PHP option "upload_max_filesize"	16M	2M	OK
PHP option "max_execution_time"	600	300	OK
PHP option "max_input_time"	600	300	OK
PHP option "date.timezone"	Africa/Casablanca		OK
PHP databases support	MySQL		OK
PHP bcmath	on		OK
PHP mbstring	on		OK

Back

Next step

Maintenant, Zabbix vérifiera toutes les exigences système pour son installation :

Pour la configuration de la base de données, nous saisissons le nom de la base et les informations de l'utilisateur que nous avons créé précédemment :

ZABBIX

Welcome
Check of pre-requisites
Configure DB connection
Zabbix server details
Pre-installation summary
Install

Configure DB connection

Please create database manually, and set the configuration parameters for connection to this database. Press "Next step" button when done.

Database type	MySQL
Database host	localhost
Database port	0
Database name	zabbix
User	zabbix
Password	*****

[Back](#) [Next step](#)

Pour la configuration des détails du serveur Zabbix, nous saisissons le nom d'hôte que nous avons enregistré dans le fichier de configuration de l'agent Zabbix installé sur le serveur Zabbix lui-même :

ZABBIX

Welcome
Check of pre-requisites
Configure DB connection
Zabbix server details
Pre-installation summary
Install

Zabbix server details

Please enter the host name or host IP address and port number of the Zabbix server, as well as the name of the installation (optional).

Host	localhost
Port	10051
Name	Zabbix server



[Back](#) [Next step](#)

Nous nous assurons que toutes les configurations sont correctes avant de commencer l'installation de Zabbix frontend :



Pre-installation summary

Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

Please check configuration parameters. If all is correct, press "Next step" button, or "Back" button to change configuration parameters.

Database type	MySQL
Database server	localhost
Database port	default
Database name	zabbix
Database user	zabbix
Database password	*****

Zabbix server	localhost
Zabbix server port	10051
Zabbix server name	Zabbix server

[Back](#)

[Next step](#)



Install

Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

Congratulations! You have successfully installed Zabbix frontend.

Configuration file "/etc/zabbix/web/zabbix.conf.php" created.

[Back](#)

[Finish](#)

Lorsque l'installation est terminée, nous aurons le message suivant :

Ensuite, nous serons redirigés vers la page de connexion de Zabbix. Nous nous connectons avec l'utilisateur par défaut "Admin" et le mot de passe "zabbix" :

The screenshot shows the Zabbix login interface. At the top is the Zabbix logo. Below it is a form with fields for 'Username' containing 'Admin' and 'Password' containing 'zabbix'. There is a checked checkbox for 'Remember me for 30 days'. A large blue 'Sign in' button is at the bottom. Below the button, a link says 'or sign in as guest'.

Le tableau de bord d'administration Zabbix :

The screenshot shows the Zabbix dashboard titled 'Global view'. The top navigation bar includes links for Monitoring, Inventory, Reports, Configuration, Administration, and a search bar. The main content area has several sections: 'System information' (listing various metrics like server status, number of hosts, items, and triggers); 'Problems by severity' (showing no data found); 'Local' (displaying a large analog clock); and 'Problems' (also showing no data found). On the right side, there are sections for 'Favourite...' and 'No maps added.'

L'installation est terminée.

NB : Il ne faut pas oublier de changer le mot de passe par défaut de l'administrateur Zabbix.

Création d'un hôte sur l'interface Web de Zabbix :

The screenshot shows the 'Hosts' creation page in the Zabbix web interface. The 'Host' tab is selected. The 'Host name' field contains 'client1-srv'. The 'Groups' dropdown shows 'Linux servers' selected. The 'Agent interfaces' section lists one interface: IP address 10.0.2.36, DNS name client1-srv, Connect to IP, Port 10050, and Default selected. Buttons for 'Add' and 'Remove' are visible.

Host name: client1-srv

Visible name:

Groups: Linux servers

* At least one interface must exist.

Agent interfaces	IP address	DNS name	Connect to	Port	Default
	10.0.2.36	client1-srv	IP	10050	<input checked="" type="radio"/> Remove

SNMP interfaces: Add

JMX interfaces: Add

IPMI interfaces: Add

Nous ajoutons un template. Le système d'exploitation installé sur notre hôte est CentOS 7, nous allons donc choisir "Template OS Linux" :

The screenshot shows the 'Hosts' template linking interface. The 'Templates' tab is selected. A table shows a linked template: Name 'Template OS Linux' and Action 'Unlink'. Below it, a search bar and 'Add' button are shown. Buttons for 'Add' and 'Cancel' are at the bottom.

Linked templates	Name	Action
	Template OS Linux	Unlink

Link new templates: type here to search

Add

Add Cancel

L'hôte a été créé.

Liste des hôtes supervisés par le serveur Zabbix :

	Name	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates	Status	Availability	Agent encryption	Info
	client1-srv	Applications 10	Items 50	Triggers 19	Graphs 10	Discovery 2	Web	10.0.2.36: 10050	Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX SNMP JMX IPMI	NONE	
	client2-srv	Applications 10	Items 34	Triggers 15	Graphs 5	Discovery 2	Web	10.0.2.34: 10050	Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX SNMP JMX IPMI	NONE	
	Zabbix server	Applications 11	Items 92	Triggers 50	Graphs 16	Discovery 2	Web	127.0.0.1: 10050	Template App Zabbix Server, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX SNMP JMX IPMI	NONE	

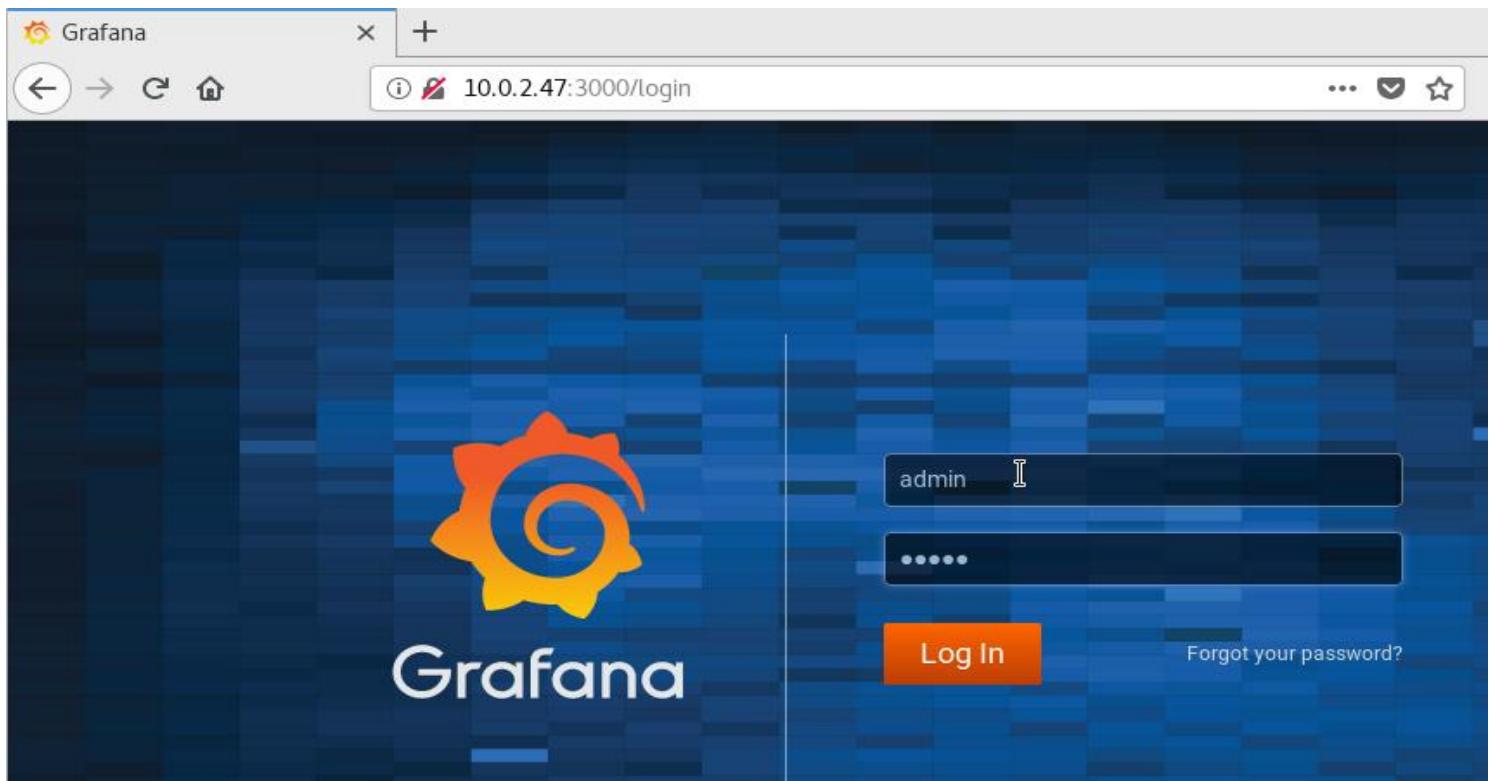
Displaying 3 of 3 found

4.14- Configuration initiale et test du serveur Grafana

Dans un navigateur Web et nous saisissons l'URL suivant : <http://10.0.2.47:3000>

Nous serons redirigés vers la page de connexion de Grafana. Nous nous connectons avec l'utilisateur par défaut "admin" et le mot de passe "admin" :

Nous serons automatiquement redirigés vers la page de la modification du mot de passe :





Sur le tableau de bord de Grafana, nous activons le plugin "Zabbix" que nous avons installé précédemment :

A screenshot of the Grafana Home Dashboard. The dashboard has a dark theme. On the left, there is a sidebar with various icons. The main area features several buttons: "Install Grafana" (with a checkmark icon), "Add data source" (with a database icon), "New dashboard" (with a grid icon), "Add Users" (with a user icon), and "Explore plugin repository" (with a bird icon). Below these buttons, there are sections for "Starred dashboards" and "Recently viewed dashboards". To the right, there are sections for "Installed Apps", "Installed Panels", and "Installed Datasources". The "Add data source" button is highlighted with a green border.

Afin d'importer les données depuis le serveur Zabbix, nous cliquons sur "Add data source", ensuite sur l'icône "Zabbix". Nous saisissons les informations requises pour la configuration :

URL : http://10.0.2.47/zabbix/api_jsonrpc.php

Zabbix API Details : nom d'utilisateur et mot de passe de l'administrateur de l'interface Web de Zabbix.

Nous activons l'option "Trends". Cette dernière nous permet d'avoir les dernières valeurs affichées sur notre tableau de bord (affichage en temps réel).

Zabbix version : 4.x

Le niveau d'affichage de Grafana est limité, un seul type d'alerte par source de données "Zabbix". Dans notre cas, nous allons choisir le type "Average" dans la rubrique "Alerting" après l'avoir activée.

The screenshot shows the Grafana interface for managing a Zabbix data source and its API configuration.

Zabbix: Settings - Grafana tab is active.

HTTP section:

- URL:** http://10.0.2.47/zabbix/api_jsonrpc...
- Access:** Server (Default)
- Whitelisted Cookies:** Add Name

Auth section:

- Basic Auth:**
- With Credentials:**
- TLS Client Auth:**
- With CA Cert:**
- Skip TLS Verify:**
- Forward OAuth Identity:**

Zabbix API details section:

- Username:** Admin
- Password:** [REDACTED]
- Trends:**
- After:** 7d
- Range:** 4d

Direct DB Connection section:

- Enable:**

Alerting section:

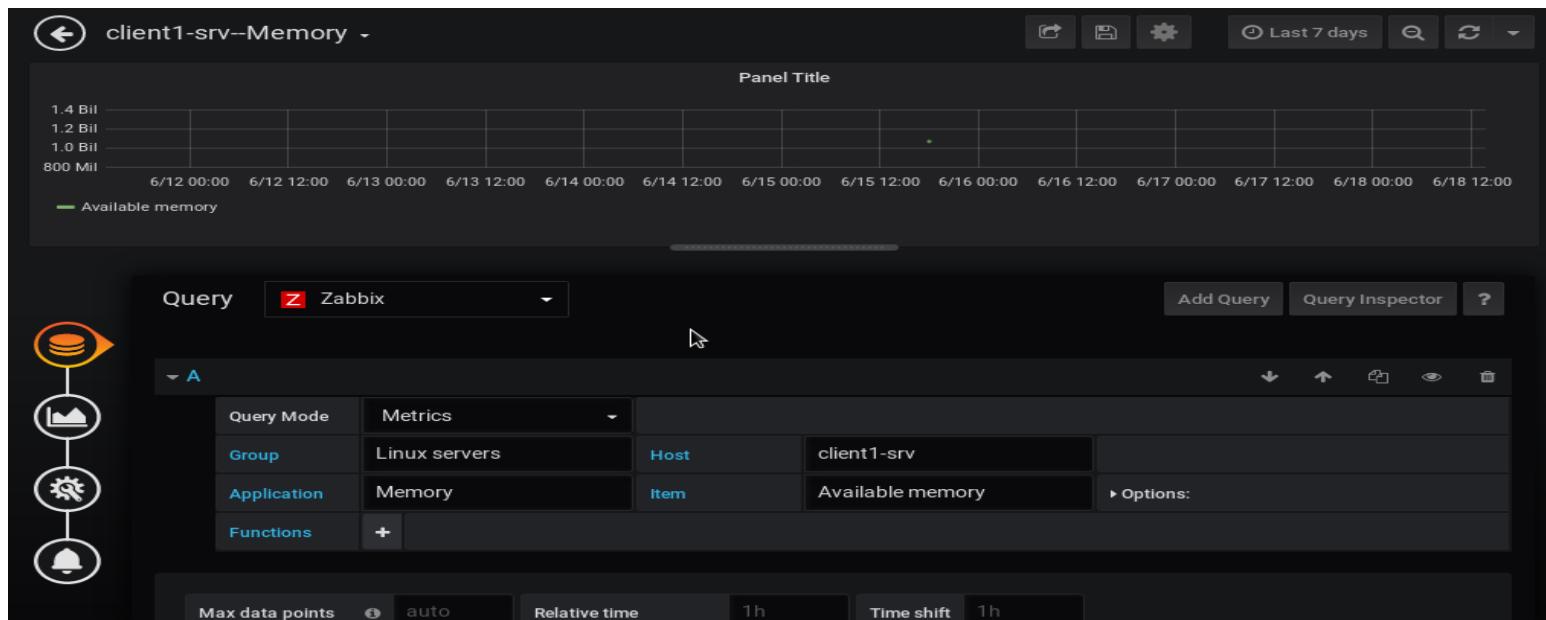
- Enable alerting:**
- Add thresholds:**
- Min severity:** Average

Other section:

Création d'un tableau de bord sur Grafana :

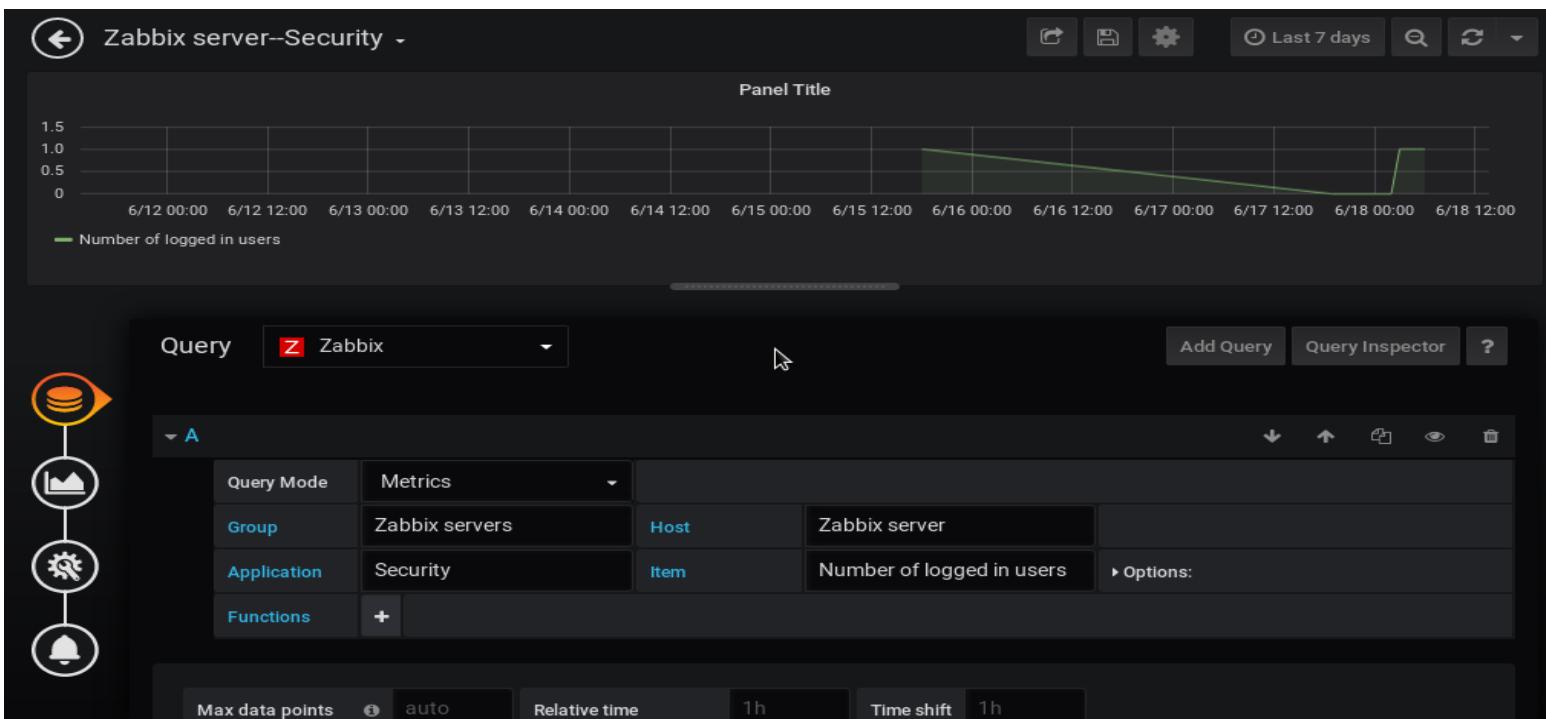
Exemple 1 : nous allons créer un tableau de bord pour afficher les valeurs de l'application "Memory", item "Available memory" du serveur "client1-srv".

Nous cliquons sur "New Dashboard", ensuite sur l'icône "Add Query". Nous soumettons notre requête comme illustré ci-dessous :



Remarque : plusieurs requêtes peuvent être soumises simultanément.

Exemple 2 : tableau de bord pour afficher les valeurs de l'application "Security", item "Number of logged in users" du serveur "Zabbix server" :



BIBLIOGRAPHIE ET WEBOGRAPHIE

1. Nicolas GREVET, Le Cloud Computing : Evolution ou Révolution ?, M2IRT, 2009

2. Alain TCHANA (Université Toulouse) : Sécurité dans le cloud computing

https://fr.wikipedia.org/wiki/Cloud_computing

<https://www.futura-sciences.com/tech/definitions/informatique-cloud-computing-11573/>

https://fr.wikipedia.org/wiki/Ordinateur_central

<https://connect.ed-diamond.com/MISC/MISC-060/Introduction-au-Cloud-Computing-risques-et-enjeux-pour-la-vie-privee>

<https://www.lebigdata.fr/virtualisation-definition>

https://fr.wikipedia.org/wiki/Centre_de_donn%C3%A9es

<https://www.lebigdata.fr/definition-data-center-centre-donnees>

https://fr.wikipedia.org/wiki/Plate-forme_collaborative

<https://www.1min30.com/dictionnaire-du-web/plateforme-collaborative/>

<https://www.differencebetween.com/difference-between-internet-and-vs-cloud-computing/>

<https://www.lemagit.fr/definition/IaaS>

https://fr.wikipedia.org/wiki/Infrastructure_as_a_service

https://fr.wikipedia.org/wiki/Plate-forme_en_tant_que_service

https://fr.wikipedia.org/wiki/Logiciel_en_tant_que_service

<https://www.lebigdata.fr/cloud-computing/cloud-prive>

<https://www.computerland.fr/difference-cloud-prive-public/>

<https://www.appvizer.fr/magazine/services-informatiques/cloud-prive/quest-ce-quun-cloud-prive-comment-choisir-son-modele>

<https://www.lemagit.fr/definition/Cloud-public>

<https://www.lemagit.fr/definition/Cloud-hybride>

<https://www.scalair.fr/blog/cloud-public>

<https://www.scalair.fr/blog/cloud-prive>

<https://www.lebigdata.fr/securite-cloud-1101>

<http://www.globbsecurity.fr/securite-cloud-voici-les-12-menaces-ne-pas-oublier-44635/>

https://en.wikipedia.org/wiki/Cloud_Security_Alliance

https://fr.wikipedia.org/wiki/Service-level_agreement

<http://www.7avoir.net/article-quel-sla-pour-le-cloud-75951325.html>

<https://www.zdnet.fr/actualites/top-2019-des-fournisseurs-de-cloud-aws-azure-gcp-ibm-sur-l-hybride-et-salesforce-domine-le-saas-39880577.htm>

https://www.academia.edu/26201885/S%C3%A9curité%C3%A9_et_confidentialité%C3%A9_des_donn%C3%A9es_sensibles_dans_le_Cloud_Computing_une_enqu%C3%A9e_sur_les_d%C3%A9veloppements_r%C3%A9cents

<https://www.netexplorer.fr/les-donnees-sont-elles-cryptées-dans-le-cloud>

<https://www.institut-numerique.org/chapitre-3-les-solutions-du-cloud-computing-51c0279cafce2>

<https://fr.wikipedia.org/wiki/VCloud>

<http://www-archive.xenproject.org/products/cloudxen.html>

<https://fr.wikipedia.org/wiki/OpenStack>

<https://fr.wikipedia.org/wiki/OwnCloud>

https://en.wikipedia.org/wiki/Apache_CloudStack

<https://fr.wikipedia.org/wiki/Elasticsearch>

<https://fr.wikipedia.org/wiki/Logstash>

<https://fr.wikipedia.org/wiki/Kibana>

<https://logz.io/blog/filebeat-tutorial/>

<https://www.elastic.co/guide/en/beats/functionbeat/current/functionbeat-overview.html>

<https://fr.wikipedia.org/wiki/NGINX>

[https://fr.wikipedia.org/wiki/Suricata_\(logiciel\)](https://fr.wikipedia.org/wiki/Suricata_(logiciel))

https://fr.wikipedia.org/wiki/Deep_packet_inspection

<https://fr.wikipedia.org/wiki/MariaDB>

<https://fr.wikipedia.org/wiki/Zabbix>

https://www.zabbix.com/zabbix_agent

<https://wiki.monitoring-fr.org/zabbix/zabbix-work>

<https://fr.wikipedia.org/wiki/Grafana>

<https://www.vuurmuur.org/trac/wiki/Suricata>

<https://fr.wikipedia.org/wiki/Pcap>

<https://www.zabbix.com/documentation/4.0/fr/manual/definitions>

https://fr.wikipedia.org/wiki/Simple_Network_Management_Protocol

https://fr.wikipedia.org/wiki/Intelligent_Platform_Management_Interface

Annexe A

Procédure d'installation de la pile ELK sur Ubuntu 18.04 LTS

I- Installation d'Oracle Java JDK 8 sur Ubuntu 18.04 à l'aide de PPA :

Ajout du PPA JAVA JDK à l'aide de la commande add-apt-repository :

```
add-apt-repository -y ppa:webupd8team/java
```

Installation :

```
apt install -y oracle-java8-installer
```

Définition du Java par défaut :

```
apt install -y oracle-java8-set-default
```

Vérification :

```
java -version
```

Configuration des variables d'environnement JAVA :

Certaines installations d'applications Java nécessitent la configuration préalable de variables d'environnement.

Pour définir les variables d'environnement JAVA, nous créons un nouveau fichier dans le répertoire /etc/profile.d :

```
vi /etc/profile.d/javajdk.sh
```

Nous plaçons les variables en fonction de l'emplacement et de la version du JDK :

```
export PATH=$PATH:/usr/jdk1.8.0_191/bin
```

```
export JAVA_HOME=/usr/jdk1.8.0_191
```

```
export JRE_HOME=/usr/jdk1.8.0_191/jre/
```

```
export J2SDKDIR=/usr/jdk1.8.0_191
```

```
export J2REDIR=/usr/jdk1.8.0_191/jre
```

Chargement des environnements dans la session en cours :

```
source /etc/profile.d/javajdk.sh
```

Vérification :

```
echo $JAVA_HOME
```

II- Installation d'Elasticsearch :

Avant d'installer Elasticsearch, nous ajoutons de la clé elastic.co au serveur :

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

Téléchargement et installation d'Elasticsearch :

```
echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | tee -a  
/etc/apt/sources.list.d/elastic-6.x.list
```

```
apt-get update
```

```
apt-get install -y elasticsearch
```

Configuration de elasticsearch.yml :

```
vi /etc/elasticsearch/elasticsearch.yml
```

Activation du verrouillage de la mémoire pour Elasticsearch (cela désactive l'échange de mémoire pour Elasticsearch) :

```
bootstrap.memory_lock: true
```

Dans le bloc "Network", nous enlevons les commentaires des lignes network.host et http.port :

```
network.host: localhost
```

```
http.port: 9200
```

Nous éditons le fichier de configuration sysconfig pour Elasticsearch:

```
vi /etc/sysconfig/elasticsearch
```

Nous enlevons le commentaire de la ligne MAX_LOCKED_MEMORY :

```
MAX_LOCKED_MEMORY=unlimited
```

Activation et démarrage du service Elasticsearch:

```
systemctl daemon-reload
```

```
systemctl enable elasticsearch
```

```
systemctl start elasticsearch
```

Vérification :

```
netstat -plntu
```

```
curl -XGET 'localhost:9200/?pretty'
```

III- Installation de Kibana avec Nginx :

1- Installation et configuration de Kibana :

apt-get install kibana

Nous éditons dans le fichier de configuration de Kibana :

vi /etc/kibana/kibana.yml

Nous enlevons les commentaires des lignes de configuration pour server.port, server.host et elasticsearch.hosts :

server.port: 5601

server.host: "localhost"

elasticsearch.hosts: ["http://localhost:9200"]

Activation et démarrage du service Kibana:

systemctl enable kibana

systemctl start kibana

Verification :

netstat -plntu

systemctl status kibana

Kibana sera exécuté sur le port 5601 en tant qu'application de noeud.

2- Installation et configuration de Nginx :

Installation des packages Nginx et httpd-tools:

apt-get install nginx apache2-utils

Nous créons un nouveau fichier de configuration d'hôte virtuel dans le répertoire conf.d :

vi /etc/nginx/conf.d/elk-srv2.conf

Configuration à y ajouter:

```
=====
```

server {

listen 80;

```

server_name elk-srv2;

auth_basic "Restricted Access";

auth_basic_user_file /etc/nginx/.kibana-user;

location / {

    proxy_pass http://localhost:5601;

    proxy_http_version 1.1;

    proxy_set_header Upgrade $http_upgrade;

    proxy_set_header Connection 'upgrade';

    proxy_set_header Host $host;

    proxy_cache_bypass $http_upgrade;

}

}
=====
```

Avec "elk-srv2" comme nom d'hôte de notre serveur ELK.

Création d'un nouveau fichier d'authentification :

```
htpasswd -c /etc/nginx/.kibana-user admin
```

```
TYPE_PASSWORD
```

Test de la configuration de Nginx et démarrage du service:

```
nginx -t
```

```
systemctl enable nginx
```

```
systemctl start nginx
```

IV- Installation de Logstash :

```
apt-get install logstash
```

Création d'un certificat SSL :

Nous générerons le fichier de certificat avec la commande openssl :

```
openssl req -config /etc/pki/tls/openssl.cnf -x509 -days 3650 -batch -nodes -newkey rsa:2048 -keyout /etc/pki/tls/private/logstash-forwarder.key -out /etc/pki/tls/certs/logstash-forwarder.crt -subj /CN=elk-srv2
```

Le fichier logstash-forwarder.crt doit être copié sur les serveurs Client.

Nous allons créer un nouveau fichier de configuration 'conflog.conf' dans le sous-répertoire 'conf.d' pour configurer les syslogs provenant des machines clients, le traitement des syslog et pour définir la sortie Elasticsearch.

```
vi /etc/logstash/conf.d/conflog.conf
```

Voir 4.6- Installation et configuration de Logstash : page 73

Nous configurons le pare-feu pour que Logstash reçoive les journaux des machines clients:

```
ufw allow 5044/tcp
```

```
ufw allow 5045/tcp
```

```
ufw reload
```

Activation et démarrage du service logstash :

```
systemctl enable logstash
```

```
systemctl start logstash
```

Vérification :

```
systemctl status logstash
```