Architecture Cloud Computing chez PSA

Michaël Labouebe

CONSERVATOIRE NATIONAL
DES
ARTS ET MÉTIERS

le cnam

MÉMOIRE D'INGÉNIEUR

présenté par : **Michaël LABOUEBE**

24 septembre 2015

pour obtenir le titre de : **Ingénieur du Conservatoire National des Arts et Métiers**

Discipline / Spécialité : **Informatique / Réseaux, Systèmes et Multimédia**

Architecture Cloud Computing chez PSA

MÉMOIRE DIRIGÉ PAR

M. DESCAMPS Philippe — *Enseignant Chercheur, Laboratoire IRTES-SET*

TUTEURS ENTREPRISE

M. FRICKER Alexandre — *Architecte technique système d'information, PSA*
M. DESSABLES François — *Expert architecte technique système d'information, PSA*

Résumé

Les systèmes d'informations deviennent de plus en plus complexes et exigeants. Les infrastructures matérielles et logicielles évoluent à un rythme effréné et les services informatiques doivent donc constamment s'adapter afin de répondre aux exigences de délais de mise à disposition et de continuité de service. Pour accompagner ces évolutions, des nouveaux modes de déploiement d'infrastructures ne cessent d'émerger et l'un d'entre eux est le logiciel de service d'infrastructure OpenStack.

Qu'est-ce qu'un logiciel de service d'infrastructure ? Quels en sont les caractéristiques ? OpenStack est-il capable de répondre aux besoins naissant des services informatiques ? La première partie de ce mémoire tentera donc de répondre à ces questions et traitera de l'administration du stockage dans OpenStack.

Le niveau très poussé d'automatisation des tâches élémentaires par OpenStack exige la mise en œuvre de puissants mécanismes de traçabilité. Ceci afin de faciliter et améliorer l'analyse d'éventuels dysfonctionnements. Les moyens de surveillance ont donc évolué afin d'être plus flexibles et mieux intégrés à ces nouveaux outils. La deuxième partie de ce mémoire abordera un outil de surveillance, par traitement des logs, utilisable dans le contexte d'OpenStack : Elasticsearch.

Abstract

Information systems are becoming increasingly complex and demanding. Hardware and software infrastructures are evolving at a rapid pace and therefore IT departments must constantly adapt to meet resources provisioning deadlines and business continuity requirements. To deal with these evolutions, new infrastructure deployment architectures emerge every day and one of them is the infrastructure as a service software : OpenStack.

What is “infrastructure as a service” software ? What are its characteristics ? Is OpenStack able to meet the emerging needs of IT services ? The first part of this paper will therefore try to answer these questions and address storage administration within OpenStack.

With the degree of automation in OpenStack, a high level of traceability is required to facilitate problem analysis and solving. As a result, these monitoring tools have evolved to become more flexible and be better integrated within the OpenStack framework. The second part of this paper will therefore explore a log processing tool used to monitor OpenStack : Elasticsearch.

Remerciements

Je tiens tout d'abord à remercier M. Philippe DESCAMPS qui, en tant que tuteur CNAM, m'a apporté une aide compétente, accordé sa confiance et une grande liberté d'exécution.

Mes remerciements s'adressent également à mes tuteurs au sein de PSA, M. Alexandre FRICKER et M. François DESSABLES, qui par leurs conseils avisés ont guidé mes travaux.

Je remercie également M. Stéphane DURUPT, membre de l'équipe des architectes techniques, ainsi que M. Christophe DUCHENOY, responsable du pôle d'expertise Linux, pour leur collaboration et contributions aux différents projets.

Enfin, je souhaitais adresser mes remerciements les plus sincères à mes collègues qui m'ont apporté leur aide, contribuant ainsi à la réussite de ce mémoire.

Liste des Abréviations

AOE ATA over Ethernet.
API Application Programming Interface.

CIFS Common Internet File System.
CNAM Conservatoire National des Arts et Métiers.

DAS Direct Attached Storage.
DRBD Distributed Replicated Block Device.
DSIN Direction des Systèmes d'INformation.
DSOA Data and Services Oriented Architecture.

HBA Host Bus Adapater.

IaaS Infrastructure as a Service.
ISCSI Internet Small Computer System Interface.
ISER ISCSI Extensions for RDMA.
ISTA Infrastructure, Security, Telecom & Architecture.

JSON JavaScript Object Notation.

LDAP Lightweight Directory Access Protocol.
LUN Logical Unit Number.

MIB Management Information Base.

NAS Network Attached Storage.
NFS Network File System.

PaaS Plateform as a Service.
PRA Plan de Reprise d'Activité.

RAM Random Access Memory.

SaaS Software as a Service.
SAML Security Assertion Markup Language.
SAN Storage Area Network.
SIXS Storage Infrastructure eXpertise and Support.
SNMP Simple Network Management Protocol.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

VRRP Virtual Router Redundancy Protocol.

WWN World Wide Name.

XML Extensible Markup Language.

Glossaire

Cloud Computing

Le cloud computing est l'accès via un réseau de télécommunications, à la demande ou en libre-service, à des ressources informatiques partagées configurables. Il s'agit donc d'une délocalisation de l'infrastructure informatique..

hyperviseur

Un hyperviseur est un système de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler en même temps sur une même machine physique..

Network Attached Storage

Un système de stockage NAS est un serveur de fichiers relié à un réseau (principalement Ethernet). Sa principale fonction est le stockage de données centralisées. A la différence du SAN le transfert des données s'effectue généralement en mode fichier et non en mode bloc. Il sert des clients réseau hétérogènes tels que Windows à travers le protocole CIFS et Unix à travers le protocole NFS..

Storage Area Network

Un réseau de stockage SAN relie des baies de stockage et des serveurs via un ensemble d'équipement dédiés tel que des switchs SAN et des routeurs SAN. Ce réseau dédie est appelé une "fabric" et permet de partager des informations à travers le protocole Fibre Channel. Ce protocole de transport permet un accès bas niveau aux disques à distance. Les échanges de données se font en mode block via des commandes SCSI qui permet de rendre les disques distants similaires à des disques locaux aux serveurs..

Virtual Router Redundancy Protocol

VRRP est un protocole standard dont le but est d'augmenter la disponibilité de la passerelle par défaut des hôtes d'un même réseau. VRRP utilise la notion de routeur virtuel, auquel est associée une adresse IP virtuelle ainsi qu'une adresse MAC virtuelle. Parmi un groupe de routeurs participant à VRRP dans un réseau, le protocole va élire un maître, qui va répondre aux requêtes ARP pour l'adresse IP virtuelle, ainsi qu'un ou plusieurs routeurs de secours, qui reprendront l'adresse IP virtuelle en cas de défaillance du routeur maître..

Sommaire

Introduction

> *In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers.*
>
> Grace Murray Hopper

Lors de la dernière décennie un changement fondamental en informatique a eu lieu. Pour subvenir aux besoins grandissant en ressources informatiques, la réponse habituelle était de rajouter des ressources à une unité existante. Cependant la limitation de la capacité produite par unité a amené les constructeurs à revoir leurs stratégies. En effet, nous sommes passés dans un modèle où il est préférable de rajouter d'autres unités pour rendre le système plus capacitif plutôt que d'agrandir les unités existantes.

Une des premières transformations allant dans ce sens fut la parallélisation des processeurs. En effet, depuis une dizaine d'années la fréquence des processeurs n'a plus évoluée suite à une contrainte sur le rapport entre la puissance utilisée et la chaleur dissipée dans ceux-ci. L'évolution des processeurs consiste aujourd'hui à rajouter des cœurs de calcul par unité de surface plutôt que d'essayer de créer des processeurs plus rapide.

Ce changement fondamental de méthode de scalabilité a poussé les éditeurs de logiciels à revoir les architectures de leurs produits. Ils ont donc dû adapter leurs logiciels pour, dans un premier temps fonctionner sur plus d'un processeur, puis finalement fonctionner sur plusieurs machines physiques en parallèle.

Cet apport fonctionnel a permis l'émergence du monde du "Cloud Computing[2]" où de nombreuses machines travaillent en parallèle afin de réaliser des tâches. Ainsi de nouveaux logiciels prévus pour déployer, surveiller et maintenir ce type d'infrastructure ont dû être créés car ceux existant n'avaient pas pris en compte ce nouveau paradigme. Les anciens logiciels de gestion étant conçus pour gérer indépendamment la configuration fine de chaque équipement, alors que les nouveaux doivent gérer des déploiements massifs, rapides, standardisés et reproductibles.

Mis devant le fait accompli, les administrateurs des systèmes actuels de la plupart des grandes entreprises ont deux choix :

- Subir l'exode progressif des utilisateurs et développeurs attirés par la réactivité et la flexibilité des infrastructures mises à disposition par les acteurs majeurs des Clouds public tels que Amazon ou Microsoft Azure ... Acteurs étant eux-mêmes promoteurs de ces systèmes de déploiement massif.
- Repenser leurs méthodes actuelles de gestion des ressources qui atteignent leurs limites et créer des Cloud privés internes ayant une ouverture sur les Cloud public. Ceci afin d'éviter la fuite des utilisateurs et rassurer les directions des systèmes d'information concernant les inquiétudes sur la protection des données sur les Cloud public.

Au sein de PSA nous avons donc fait le choix d'implémenter un Cloud privé interne. Par intérêt personnel pour le sujet je me suis donc rapproché de l'équipe des architectes au sein de PSA afin de participer à ce projet. Il m'a donc été donné l'opportunité de faire partie de l'équipe projet en charge de l'étude et la mise en place initiale du système de Cloud privé interne au groupe.

En guise d'introduction, dans un premier temps, je présenterai le groupe PSA ainsi que l'organisation de l'informatique dans le groupe. Cela consistera à décrire les différentes strates hiérarchiques mises en place pour gérer l'informatique. Je décrirai ainsi quel est mon rôle au sein de cette organisation.

Dans un second temps je présenterai les tenants et aboutissants du projet de Cloud privé interne. Le sujet du mémoire d'ingénieur CNAM sera resitué dans le contexte de ce projet et une explication des enjeux plus détaillée y figurera. Je reviendrai aussi sur les concepts et savoir-faire nécessaires à l'élaboration de ce projet. Finalement un planning macroscopique des différentes étapes du projet ainsi qu'une feuille de route de celui-ci y seront présentés.

Dans un troisième temps je décrirai les concepts et composants nécessaires à une infrastructure de type Cloud. Je reviendrai sur l'historique de l'infrastructure informatique de PSA afin de résumer et comparer les changements nécessaires qu'ils soient organisationnels ou techniques.

Les deux chapitres suivants constitueront le cœur technique de mon mémoire. Dans ces deux chapitres seront abordés à la fois le système de déploiement et de maintenance d'un système de Cloud privé (OpenStack) ainsi que le système de surveillance de celui-ci (Elasticsearch). La majorité de mes contributions au projet y figureront et je réaliserai un bilan de chacun d'entre eux.

Finalement je conclurai ce mémoire par un résumé des points clés de celui-ci, ce qu'il m'aura apporté ainsi que les opportunités et ouvertures restantes à étudier.

Chapitre

1

L'informatique chez PSA

1.1 PSA : Le groupe

1.1.1 Présentation générale du groupe PSA

Peugeot est une entreprise qui a été fondée en 1810 par Émile PEUGEOT à Montbéliard en Franche-Comté. Elle produit à l'origine des moulins à café puis s'oriente vers le domaine de l'automobile à partir de 1891.

Suite à des divergences d'opinions et des tensions familiales, Armand PEUGEOT, de son côté, a fondé en 1896 la "Société des automobiles Peugeot". Et en 1965, la Société des Automobiles Peugeot a été renommé en PSA - Peugeot Société Anonyme. Après avoir racheté Citroën (en faillite et au bord du dépôt de bilan) en 1976, PSA devient PSA Peugeot Citroën. En 1978, PSA rachète Chrysler Europe qui prend le nom de Talbot. PSA devient alors numéro un en Europe et numéro quatre dans le monde derrière Général Motors, Ford et Toyota.

Dans une optique d'expansion et pour ne plus dépendre en majorité du marché Européen, PSA s'est développé dans le reste du monde, tout particulièrement en Chine. Elle y a installé non seulement plusieurs usines, mais aussi un département de Recherche et de Développement, pour mieux répondre aux spécificités du marché local.

1.1.2 Quelques chiffres

En 2014 PSA Peugeot Citroën est détenue par 3 actionnaires principaux : l'entreprise chinoise Dongfeng, l'État français et la famille Peugeot détenant chacun 14%. PSA a réalisé en 2014 un chiffre d'affaires de 53,6 milliards d'euros avec 2,9 millions de véhicules vendus. Ceci fait de PSA le deuxième constructeur européen de voitures particulières et le neuvième mondial. Depuis 2014 Carlos Tavares est le PDG du groupe qui compte 184.804 salariés dans le monde dont environ 84.000 en France.

La production automobile est au cœur de l'activité de PSA Peugeot Citroën. Près de 84.000 personnes sont réparties dans 21 centres de production automobile (dont un site d'éléments détachés) et 15 usines de mécanique bruts. Le groupe possède des activités complémentaires à l'automobile avec Banque PSA Finance et Faurecia (équipementier).

1.1.3 Implantations en Europe

Le groupe PSA est bien implanté en Europe (Fig. 1.1) :

- France : Mulhouse, Poissy, Rennes, Sevelnord, Sochaux, Vesoul.
- Europe : Graz (Autriche), Kolin (République tchèque), Madrid (Espagne), Mangualde (Portugal), Trnava (Slovaquie), Vigo (Espagne), Val di Sangro (Italie).

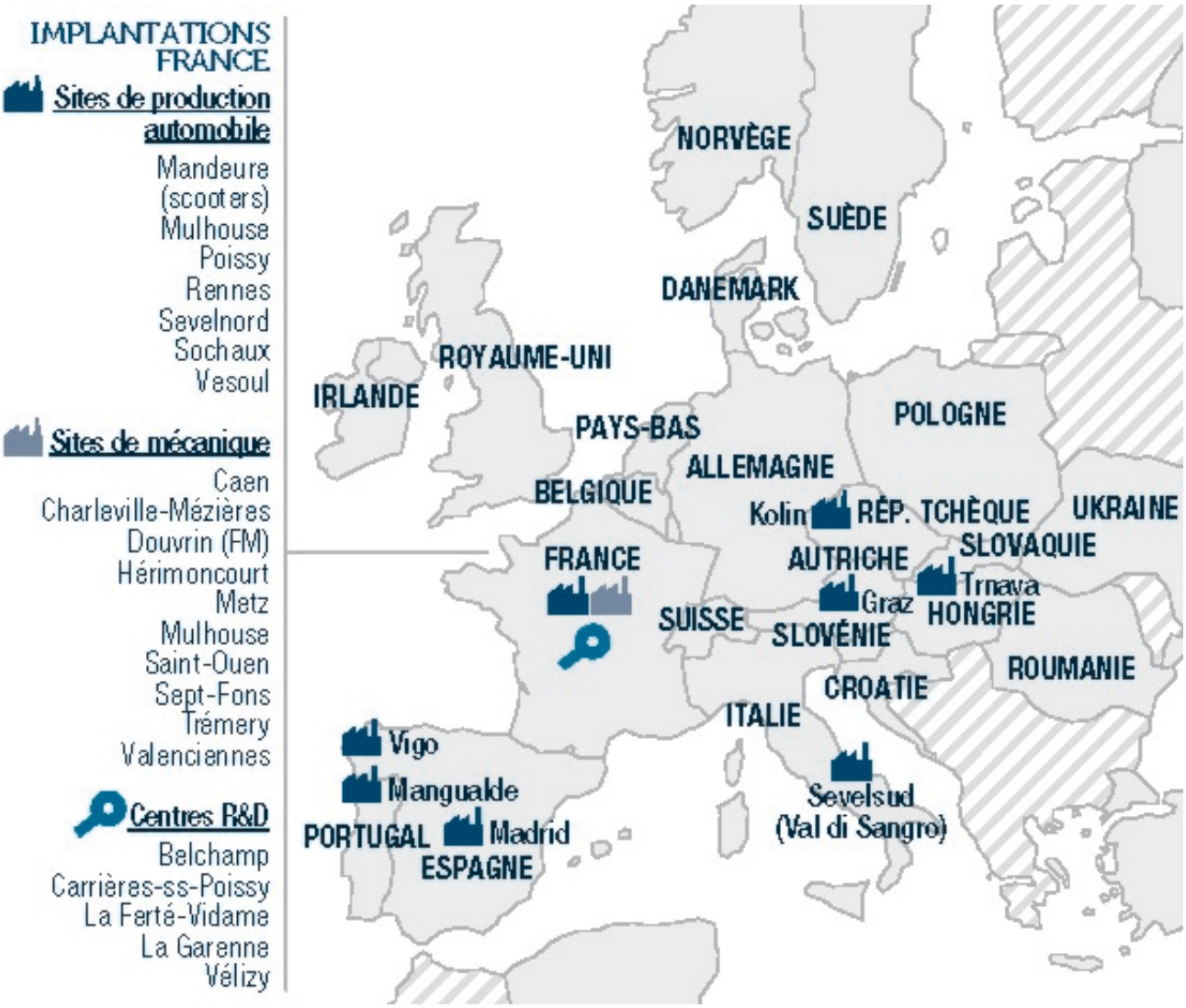

Fig 1.1 – Implantations de PSA en Europe

1.1.4 Implantations dans le monde

Le groupe PSA s'est aussi implanté dans divers pays (Fig. 1.2) :

- Buenos Aires (Argentine)
- Bursa (Turquie)
- Kaluga (Russie)
- Porto Real (Brésil)
- Wuhan (Chine)
- ainsi que d'autres sites en partenariats avec divers constructeurs.

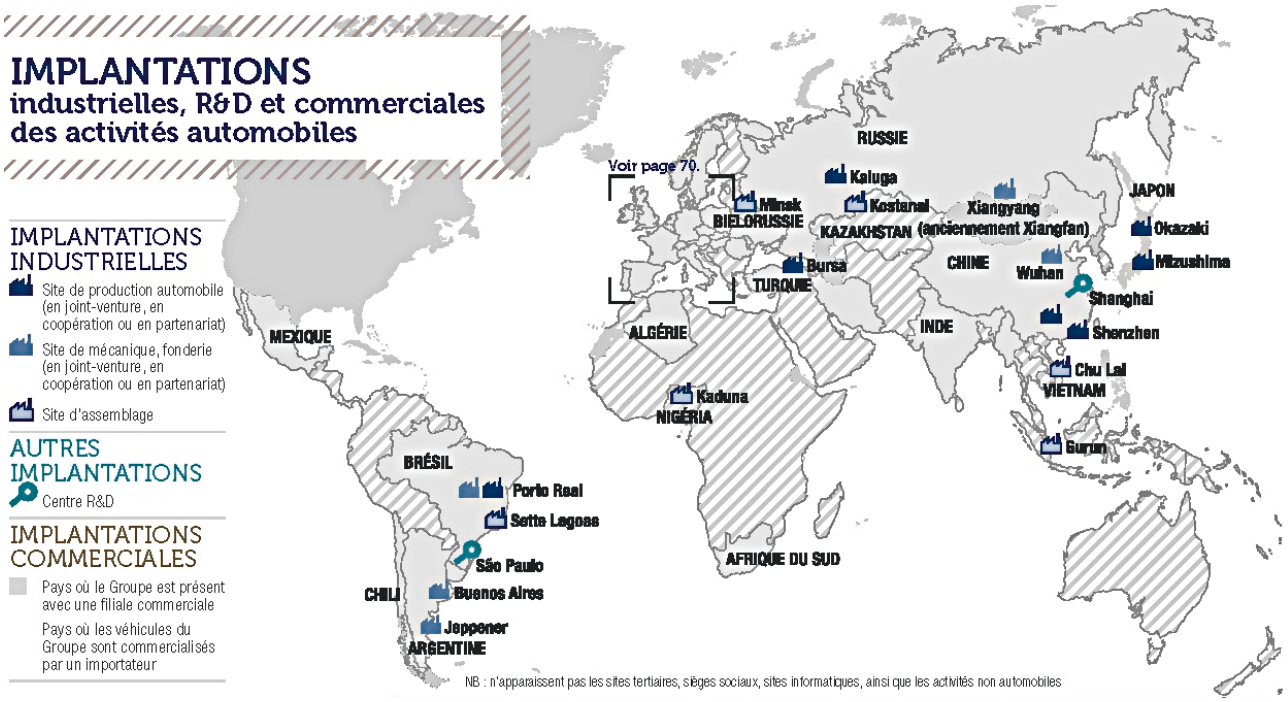

Fig 1.2 – Implantations de PSA dans le monde

1.1.5 Les Centres informatiques de PSA

L'informatique de PSA est centralisée sur deux sites (Fig. 1.3) : Bessoncourt (près de Belfort) et Achères (près de Poissy). Ces deux sites permettent la mise en place d'un Plan de Reprise d'Activité (PRA) pour les applications les plus critiques. C'est-à-dire que si l'un des sites n'est plus opérationnel pour une raison quelconque, l'autre site prendra le relais.

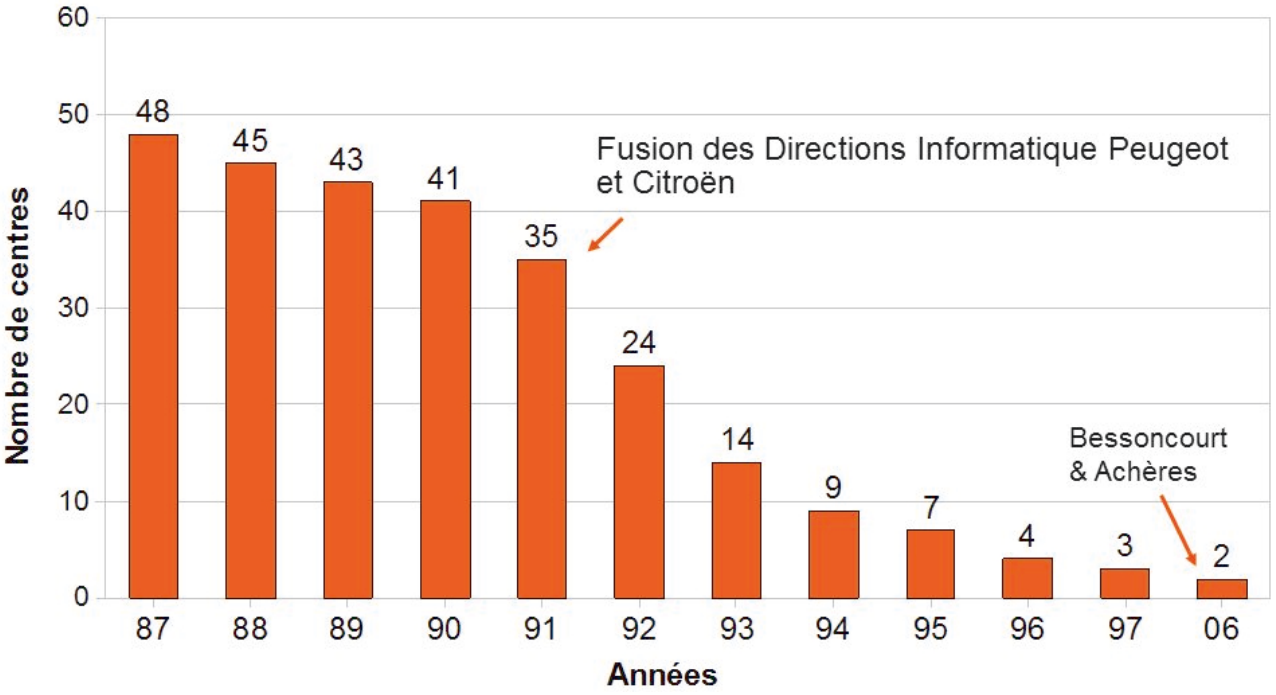

Fig 1.3 – Evolution du nombre de Datacenters du Groupe

Le site de Bessoncourt a été ouvert en mai 1980, puis son importance grandissante au sein de groupe a fait qu'il a connu trois extensions suivantes : la première en 1992, la seconde en 1997 et la dernière en 2007.

Au fil de ses extensions, le site a connu une augmentation de ses effectifs (plus de 330 personnes actuellement), la surface des salles informatiques est de 3000 m^2 et la consommation électrique annuelle est de 17400 MWh.

1.2 La Direction des Systèmes d'INformation (DSIN)

La Direction des Systèmes d'INformation (DSIN) élabore et met en œuvre les schémas directeurs des systèmes d'information pour l'ensemble des directions du groupe. Elle assure le maintien et le bon fonctionnement des infrastructures informatiques du groupe. La DSIN a quatre missions principales indiquées ci-dessous :

- Assurer l'adéquation des SI aux ambitions du Groupe (priorités business).
- Garantir la cohérence des architectures fonctionnelles et techniques des SI.
- Assurer au quotidien la performance et la disponibilité des applications et leur distribution aux postes de travail.
- Faire bénéficier le Groupe des nouvelles technologies de l'information et de la communication afin d'optimiser le ratio performance / coût.

Constituée de plus de 2100 employés la DSIN est structurée de la manière de la manière suivante (Fig. 1.4) :

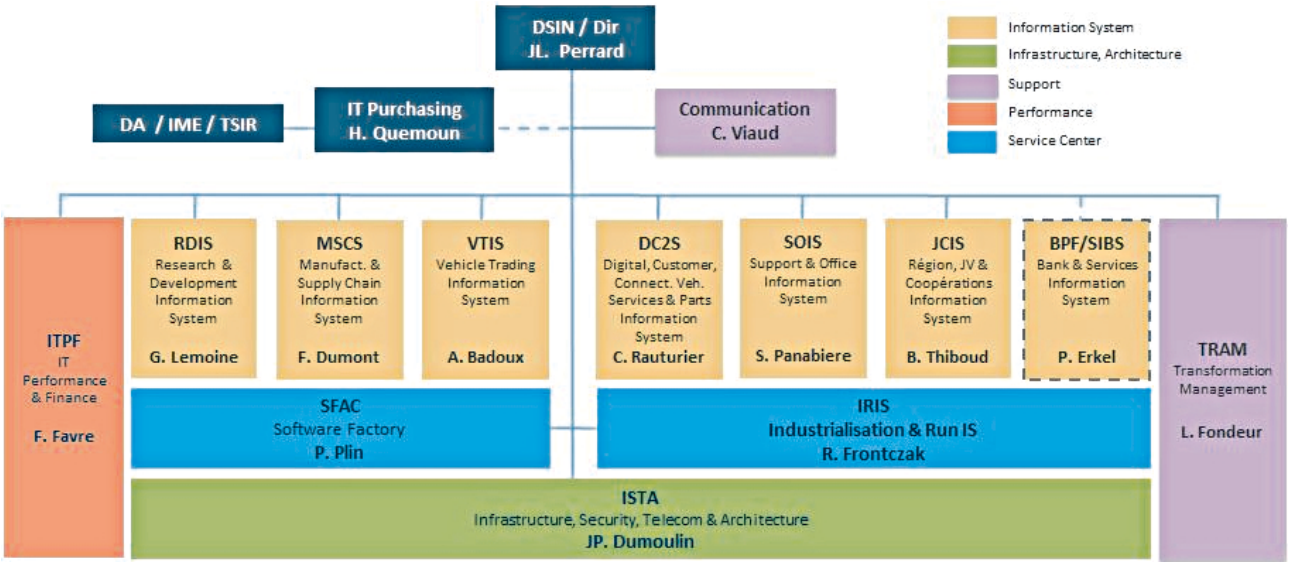

Fig 1.4 – Hiérarchie DSIN

Les deux équipes avec lesquelles j'ai collaboré lors de mon projet se situent dans le service Infrastructure, Security, Telecom & Architecture (ISTA).

Le service ISTA (Fig. 1.5) est composé de l'ensemble des équipes ayant en charge l'infrastructure informatique de PSA.

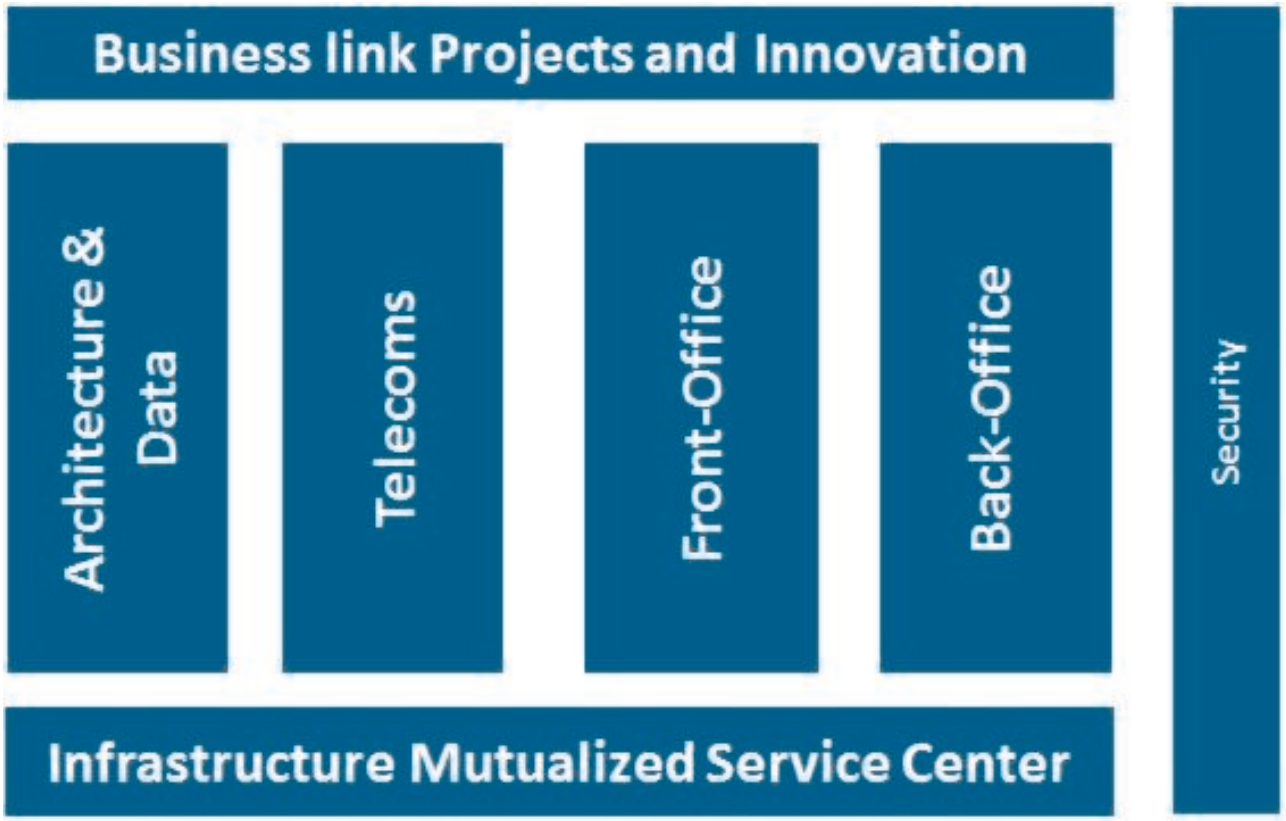

Fig 1.5 – Hiérarchie ISTA

Le service qui est mon assignation actuelle se nomme Storage Infrastructure eXpertise and Support (SIXS) et se situe dans le groupe "Back Office".
Le service avec qui j'ai collaboré sur mes projets se nomme Data and Services Oriented Architecture (DSOA) et se situe dans le groupe "Architecture & Data".

1.3 Le service Storage Infrastructure eXpertise and Support (SIXS)

Le service Storage Infrastructure eXpertise and Support (SIXS) est une branche de l'entité "Back Office" constitué de neuf personnes (Fig. 1.6).

Fig 1.6 – Hiérarchie SIXS

Le service SIXS a pour objectifs de :

- Définir et mettre en œuvre les architectures de stockage fiables, évolutives et répondant aux besoins du groupe.
- Administrer les espaces de stockage mutualisés en veillant à optimiser leur usage.

Les missions principales du service SIXS sont les suivantes :

- Définir une architecture globale et cohérente de stockage, répondant aux besoins d'accès aux données, incluant les moyens nécessaires matériels et logiciels.
- Réaliser les tests et validations techniques des matériels et logiciels concourant à la gestion du stockage.
- Réaliser le dimensionnement des infrastructures de stockage et des réseaux de stockage en recherchant le meilleur compromis besoins/coûts.
- Maintenir des relations privilégiées avec les fournisseurs du domaine (disques, SAN), pour proposer des évolutions et innovations porteuses d'avantages pour PSA.
- Administrer les espaces de stockage mutualisés en mettant à disposition des applications et des projets les espaces disques les plus adaptés en termes de performance, de coût et de répartition.
- Administrer les éléments des réseaux de stockage (switchs, routeurs) en prenant en compte les aspects de sécurité et de performance.
- Assurer la gestion technique (installation, traitements des incidents, prise en compte des évolutions) des matériels et logiciels de stockage.

Actuellement j'occupe un poste d'administrateur système de stockage au sein de ce service. Mon rôle au sein de cette équipe consiste à fournir, en tant que développeur, un ensemble d'outils permettant l'automatisation des tâches redondantes afin d'améliorer l'efficacité opérationnelle du service. Je suis aussi un des principaux représentants de notre service dans les projets d'avance de phase tels que le stockage objet, la centralisation des logs et le service d'infrastructure OpenStack.

1.4 Le service Data and Services Oriented Architecture (DSOA)

Le service Data and Services Oriented Architecture (DSOA) est une l'entité "Architecture & Data" constitué de 31 personnes (Fig. 1.7). Le service est principalement constitué d'architectes techniques et contient un sous service dédié à la gestion des bases de données.

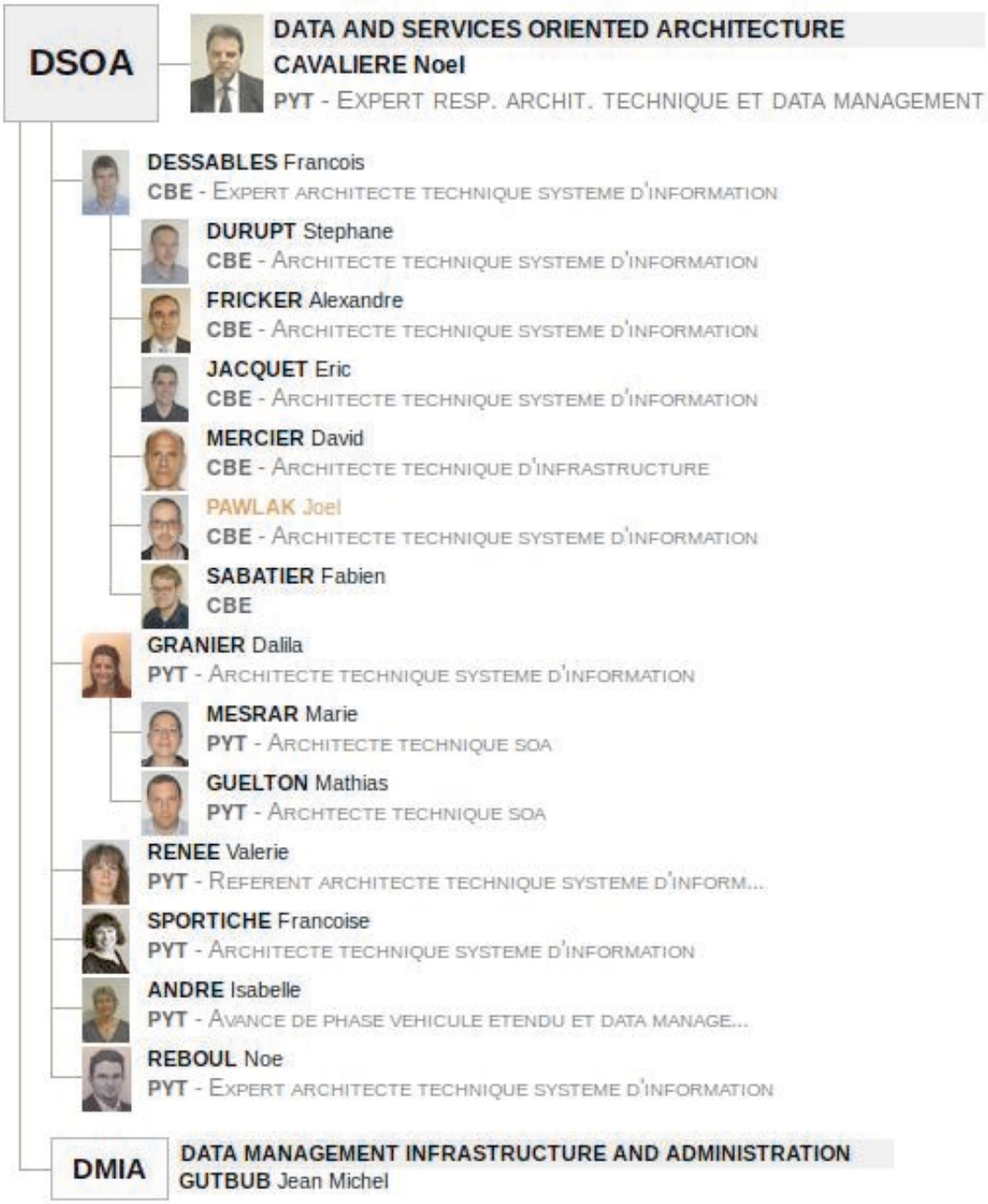

Fig 1.7 – Hiérarchie DSOA

DSOA a en charge l'architecture technique d'entreprise et en particulier les solutions d'infrastructure pour le traitement de la donnée. Le service construit des solutions qui répondent aux enjeux du Digital, du Véhicule Connecté et des nouvelles technologies de gestion de la donnée.

Les missions principales du service DSOA sont les suivantes :

- Dans son rôle d'architecture il construit la vision et la stratégie de l'architecture technique du groupe. En particulier dans le domaine du traitement de la donnée, le service définit la cible, la feuille de route et les projets de mise en œuvre de l'architecture et de l'infrastructure.
- Dans son rôle de réalisateur il met en œuvre et assure les évolutions de l'architecture et de l'infrastructure pour le traitement de la donnée. Le service pilote les projets d'infrastructure et accompagne les projets des autres entités qui utilisent cette infrastructure.
- Dans son rôle de mainteneur il garantit la qualité de service de l'infrastructure en favorisant les actions préventives et assure la vie courante du domaine des bases de données.

C'est auprès de ce service que la plupart de mes contributions constituant ce mémoire ont été effectuées. En effet, j'ai collaboré avec M. DURUPT sur l'outil de service d'infrastructure OpenStack et avec M. FRICKER sur la centralisation des logs via Elasticsearch. M. DESSABLES étant l'expert architecte m'ayant fait confiance pour travailler sur ses deux sujets.

Chapitre

2 Un projet pluridisciplinaire

2.1 Contexte du projet

Le groupe PSA, via ces deux Datacenters principaux, compte plus de 3600 serveurs physiques, 7000 machines virtuelles, 10 Po de stockage, 280 switchs SAN et 640 switchs LAN.

Une majorité des outils servant à administrer les différents équipements et installer les machines ont étés développés en interne par les différents experts des domaines concernés. Le matériel et les logiciels évoluant de plus en plus rapidement et dans un contexte de très forte concurrence, les restrictions budgétaires et de personnels n'aidant pas, il devient difficile pour le groupe de développer et maintenir ses propres outils internes de gestion. Il en est de même pour la partie surveillance des équipements et machines. Plusieurs outils internes s'occupent de différents niveaux de détails dans la surveillance avec des fonctions parfois redondantes et ne collaborant pas forcément entre eux pour avoir une vision globale de l'état de l'infrastructure.

Ce projet s'inscrit donc dans un contexte de modernisation des outils de gestion de l'infrastructure chez PSA. Il implique la collaboration entre différents équipes au sein de PSA.

Le service DSOA étant responsable de la conception, la planification et la mise en place initiale des nouveaux outils, il doit s'appuyer sur l'expertise des personnes des différents domaines afin d'avoir un projet cohérent répondant aux attentes des différents services. Parmi ces différents services se trouve l'équipe dont je fais partie (SIXS) qui a en charge la gestion du stockage.

En tant que développeur et concepteur des automatismes de mon service, il m'a été attribué le rôle de représentant technique pour la partie stockage. Le sujet de ce mémoire se décompose donc en deux grandes parties principalement centrées sur la partie stockage :

- La partie service d'infrastructure OpenStack
- La partie centralisation des logs dans Elasticsearch

2.2 Enjeux du projet

2.2.1 Enjeux du projet OpenStack

Le projet de service d'infrastructure OpenStack a été initié pour plusieurs raisons. Une des principales raisons est que les outils internes de PSA sont de plus en plus difficiles à maintenir pour des raisons budgétaires, de charge de développement et parfois de manque de compétences internes. Un exemple simple demandant de nouveaux développements et/ou validations est l'introduction d'un nouveau type de matériel au sein des Datacenters.

Une personne du service gérant ce type de matériel doit effectuer un nouveau développement afin de l'intégrer aux outils internes existants d'automatisation. Ceci prend un temps variable en fonction de la charge du service concerné et de la disponibilité des compétences en interne de ce service. Pendant la période de transition menant à l'automatisation de ces équipements, les opérations nécessaires sur ces équipements sont effectuées manuellement par le personnel du service. Cela entraîne une charge temporaire supplémentaire pour le reste de l'équipe déjà privée de la personne en charge de l'automatisation.

De plus, les automatismes de déploiement des machines virtuelles deviennent de plus en plus difficiles à faire évoluer. En effet, l'architecture technique des outils sous-jacents n'a pas été pensée de manière très évolutive et certains besoins récents d'automatisme mettent en exergue la dette technique associée à ceux-ci.

A cela il faut ajouter le fait que les délais de mise à disposition des environnements pour les utilisateurs sont très variables selon la complexité des architectures demandées. Ils peuvent aller de deux jours dans les meilleurs cas, à neuf semaines entre l'expression du besoin utilisateur et la livraison de l'environnement fonctionnel.

Il a donc été décidé de réaliser une étude de faisabilité et un comparatif de la solution généraliste de service d'infrastructure OpenStack en remplacement de certains des outils internes PSA.

Les principaux objectifs de ce projet sont donc :

- Comparer les possibilités de la solution OpenStack avec les solutions internes.
- Estimer les gains de temps de déploiement d'une nouvelle technologie et les coûts de maintenance.
- Tenter de standardiser les méthodes de déploiement des infrastructures.
- Donner à l'utilisateur final une plus grande autonomie dans la définition de ses besoins.
- Fournir de meilleurs délais de mise à disposition de l'infrastructure.
- Explorer des cas d'usages et l'éligibilité des infrastructures à ce nouveau mode de gestion.

Parmi ces objectifs mes contributions dans ce mémoire graviteront autour des points suivants :

- Se familiariser avec le logiciel de service d'infrastructure OpenStack.
- Comparer les possibilités de la partie stockage de la solution OpenStack avec les solutions internes.
- Estimer les gains de temps de déploiement d'une nouvelle technologie de stockage.
- Fournir au service DSOA mon expertise dans le domaine du stockage.
- Effectuer les tests fonctionnels et valider le bon fonctionnement de la partie stockage d'OpenStack.
- Explorer la possibilité de remplacer une partie de nos outils internes par OpenStack.

2.2.2 Enjeux du projet Elasticsearch

Au fil des années et des différentes demandes des utilisateurs de nombreuses méthodes de surveillance des équipements et applications ont été développées en interne par PSA. Elles sont plus ou moins toutes basées sur des traces d'exécution (logs) diverses et variées.

Parmi elles on peut citer par exemple :

- Les traces des différents logiciels communs à l'infrastructure de fournisseurs externes.
- Les traces des différents matériels de fournisseurs externes.
- Les traces des systèmes d'exploitation.
- Les traces applicatives qui fournissent des informations concernant le déroulement des logiciels développés en interne.

En fonction de la provenance de ces logs, la responsabilité de suivre, stocker et analyser ces logs change. On va donc retrouver les responsabilités suivantes :

- Les traces des différents logiciels communs à l'infrastructure de fournisseurs externes : responsabilité du service spécialisé qui fournit ces logiciels.
- Les traces des différents matériels de fournisseurs externes : responsabilité du service en charge de ce matériel.
- Les traces des systèmes d'exploitation : responsabilités des experts système.
- Les traces applicatives qui fournissent des informations concernant le déroulement des logiciels développés en interne : responsabilité commune entre les développeurs et personnes effectuant le déploiement des applications.

Ces logs sont exploités dans différents contextes comme la surveillance et la détection d'incidents, ou encore l'analyse de causes se rapportant à un incident.

La difficulté principale dans cette gestion est la complexité des architectures : Chacune des couches matérielles ou logicielles possède ses propres logs. Un serveur physique héberge un ou plusieurs serveurs logiques par la méthode du partitionnement. Chaque serveur logique peut contenir plusieurs serveurs virtuels, chacun avec son propre système d'exploitation. Chaque serveur virtuel héberge de nombreux logiciels qui peuvent s'exécuter dans une JVM (machine virtuelle Java) dédiée. Chacune de ces couches matérielles ou logicielles possède ses propres logs.

De plus, une seule application peut faire appel à de multiples couches applicatives et en même temps être répartie sur des dizaines de serveurs. Dans ce cas le responsable d'une application a la mission quasi impossible de maintenir cet ensemble en état de marche. Pour rajouter à la complexité de la tâche chaque couche applicative et chaque type de matériel aura son propre format de log et son propre mode d'horodatage.

Pour répondre à cette problématique, il a été décidé de mettre en place une solution basée sur Elasticsearch.

Les objectifs généraux de ce projet sont donc :

- Avoir une première approche de la couche applicative basée sur Elasticsearch.
- Comparer les possibilités de la solution Elasticsearch avec les solutions internes.
- Tenter de standardiser les méthodes de collecte et analyse des logs.
- Donner à l'utilisateur final une interface commune pour gérer les logs des plusieurs provenances.
- Explorer des cas d'usages de la solution Elasticsearch, tels que le croisement des informations de différentes sources.

Parmi ces objectifs mes contributions dans ce mémoire graviteront autour des points suivants :

- Avoir une première approche de la couche applicative basée sur Elasticsearch.
- Comparer les possibilités de la solution Elasticsearch avec nos solutions internes de surveillance du stockage.
- Fournir au service DSOA mon expertise dans le domaine du stockage.
- Créer une interface personnalisée basée sur la solution Elasticsearch pour la prise en compte d'alertes du domaine du stockage.

2.3 Concepts, savoir-faire et outils

2.3.1 Maîtrise des concepts

Ces deux projets de modernisation de l'infrastructure nécessitent la maîtrise d'un certain nombre de concepts qui peuvent paraître abstraits de prime abord.

Parmi les concepts qui sont communs aux deux solutions, de par leurs implémentations, on retrouve le concept d'application distribuée. En effet, ces deux solutions sont constituées de plusieurs logiciels sous-jacents ne fonctionnant pas forcément sur une unique machine. Afin de répondre aux besoins d'élasticité à la charge la plupart des logiciels sous-jacents sont capables d'être exécutés sur plusieurs serveurs en parallèle.

Le deuxième concept à maîtriser est celui de séparation des responsabilités. L'architecture technique des deux solutions est complexe et il est irréaliste de construire ces solutions de manière monolithique. Ces solutions reposent donc sur un ensemble de logiciels indépendants discutant entre eux de manière standardisée afin de rendre un service final complexe qui peut impliquer un grand nombre d'opérations atomiques.

Le dernier concept utilisé dans ce type de solution, et peut-être le plus important, est celui des couches d'abstractions. Pour pouvoir maîtriser la diversité des matériels et logiciels rendant le même service il est nécessaire de trouver une base commune entre eux. Cette base commune constitue alors la couche d'abstraction, laissant les détails des implémentations aux soins des modules spécialisés gérants les logiciels et matériels.

2.3.2 Les savoir-faire

Les savoir-faire nécessaires à la bonne conduite de ces projets sont les suivants :

- La gestion de projet afin de planifier les différentes étapes de ceux-ci.
- La collaboration inter-équipes est très importante pour le partage de compétences.
- Le développement applicatif est nécessaire pour l'adaptation des outils aux besoins internes.
- L'analyse de problèmes afin de détecter et résoudre les dysfonctionnements des systèmes.

2.3.3 Les outils

Ces projets m'ont permis de perfectionner et d'élargir mon utilisation d'un ensemble d'outils tel que :

- Git [13] qui est un logiciel de gestion de versions collaboratif utilisé par les développeurs des deux projets de logiciels libres OpenStack et Elasticsearch.
- Chef [1] est un logiciel libre permettant la gestion de la configuration des serveurs utilisés pour déployer une des distributions OpenStack que nous avons testée.
- Wireshark [33] est un analyseur de paquets réseau qui m'a beaucoup aidé dans les deux projets.

2.4 Planning et livrables

2.4.1 Planning et livrables du projet OpenStack

Les livrables dans le contexte d'OpenStack, pour mon domaine de spécialité qui est le stockage, sont les suivants :

- Effectuer les tests fonctionnels de la gestion du stockage dans OpenStack pour le matériel actuellement présent au sein de PSA.
- Documenter les configurations de références, les points bloquants et les meilleures pratiques du stockage dans OpenStack.

Voici dans les grandes lignes le déroulement du projet OpenStack (Table 2.1) ainsi que son diagramme de Gantt associé (Fig. 2.1) :

Tâche 1 : Janvier 2014	Lancement du projet
Tâche 2 : Février à décembre 2014	Premiers tests de la version communautaire OpenStack et de la version SUSE Cloud
Événement 1 : Septembre 2014	Formation extérieure de 5 jours à SUSE Cloud
Événement 2 : 3 au 7 novembre 2014	Participation à l'OpenStack Summit Kilo à Paris [28]
Tâche 3 : Janvier à décembre 2015	Tests complémentaires et tests de la version HP Helion
Tâche 4 : Mai 2015	Appel d'offre fournisseurs de distribution OpenStack
Tâche 5 : Décembre 2015	Choix final de fournisseur de distribution OpenStack

Table 2.1 – Planning du projet OpenStack

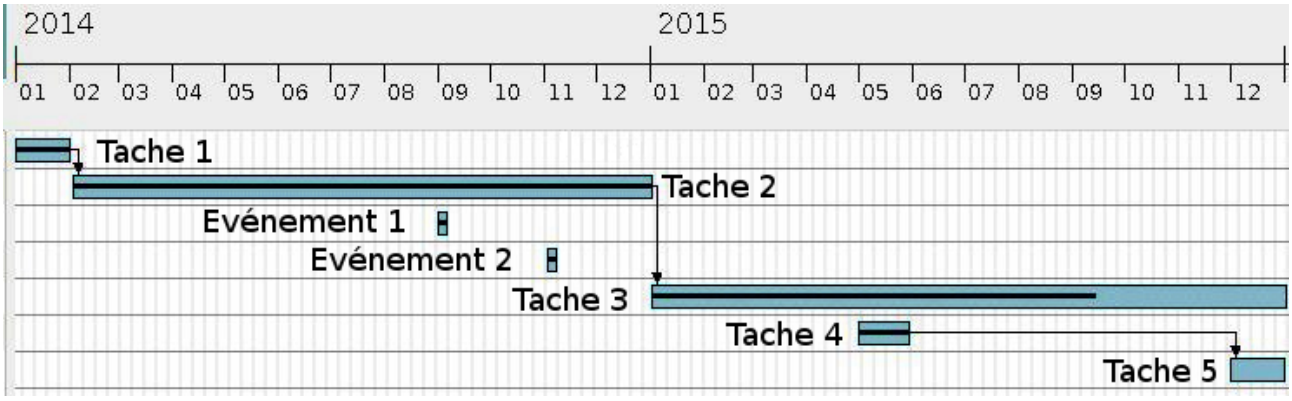

Fig 2.1 – Diagramme de Gantt du projet OpenStack

2.4.2 Planning et livrables du projet Elasticsearch

Dans le projet Elasticsearch les livrables qui sont à ma charge sont les suivants :

- Définir la liste des différents matériels nécessitant la centralisation des logs dans le domaine du stockage.
- Développer un moyen de déployer automatiquement la configuration nécessaire à la centralisation sur les équipements.
- Effectuer la standardisation des logs de ceux-ci et référencer les méthodes pour y arriver.
- Développer un outil interne pour le service en charge du stockage (SIXS) permettant de simplifier la prise en compte et l'analyse des événements via le logiciel Elasticsearch.

Voici un macro planning du déroulement du projet Elasticsearch (Table 2.2) ainsi que son diagramme de Gantt associé (Fig. 2.2) :

Tâche 1 : Janvier 2014	Lancement du projet
Tâche 2 : Février à décembre 2014	Mise en place de la plateforme de test et début de l'intégration de SVC, Switch et NETAPP
Tâche 3 : Février à mars 2015	Développement de l'outil interne de traitement des alertes
Tâche 4 : Juillet 2015	Passage en production de l'outil interne de traitement des alertes

Table 2.2 – Planning du projet Elasticsearch

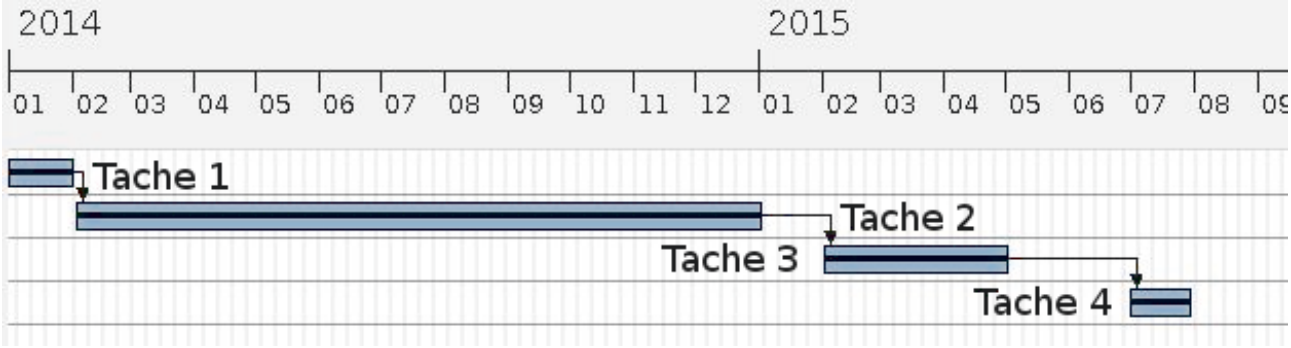

Fig 2.2 – Diagramme de Gantt du projet Elasticsearch

Chapitre

3 Les services d'infrastructures

3.1 L'infrastructure c'est quoi ?

L'infrastructure au sens large du terme est constituée de tout ce qui est nécessaire au fonctionnement d'une machine, qu'elle soit physique ou virtuelle. Basiquement il faut cinq types de ressources primaires :

- De la puissance de calcul : des processeurs physiques ou virtuels.
- De la mémoire pour effectuer les traitements : que ce soit de la RAM physique ou réallouée à une machine virtuelle.
- De l'espace de stockage pour stocker les données générées.
- Un accès réseau au monde extérieur pour communiquer les données générées.
- Un système d'exploitation afin d'orchestrer les échanges de données.

A noter que dans nos ressources nous excluons l'électricité, la climatisation et les ressources humaines qui sont des ressources physiques difficilement déployables à la volée. Bien sûr il existe une multitude de ressources annexes qui peuvent être vues comme des extensions ou variantes des catégories précédentes. Parmi elles on peut trouver par exemple :

- Catégorie "Espaces de stockage" : stockage NAS, stockage SAN, système de fichier distribué, stockage objet ...
- Catégorie "Réseaux" : routeur, répartiteur de charge, pare-feu, VPN, DNS ...

3.2 Les méthodes de gestion

La majorité de ces ressources peuvent être implémentées de deux manières différentes. Soit l'implémentation de la fonction est reliée à un matériel physique et dans ce cas le déploiement requiert une intervention humaine dans le Datacenter afin d'effectuer le câblage nécessaire, soit de manière logicielle et dans ce cas il est envisageable d'automatiser l'opération.

Une illustration simplifiée des deux manières de déployer une ressource est l'accès au réseau LAN d'un serveur. Dans le cas de l'implémentation physique cela implique de connaître à l'avance toutes les informations nécessaires afin d'intervenir physiquement pour relier le serveur au bon réseau. Alors que dans le cas d'une implémentation logicielle le serveur est déjà pré-câblé à un réseau générique. Lorsque l'utilisation de ce serveur devient nécessaire, il suffit d'activer et configurer le port réseau auquel il est relié depuis le switch ou le routeur pour que le serveur rejoigne le bon réseau.

Aujourd'hui une majorité des hébergeurs tels que Amazon, Microsoft Azure ou autre, gèrent une infrastructure dont les fonctions sont implémentées de manière logicielle. Le principe est donc de déployer une importante quantité de matériel physique à l'avance est de les mettre en "pause". Lorsque de nouveaux besoins de ressources se font ressentir, du fait d'une demande des administrateurs ou des utilisateurs, soit le matériel physique actuellement en fonctionnement peut fournir les ressources logiques demandées, soit le juste nécessaire de matériels physique sera activé à la volée pour répondre aux attentes.

Cette souplesse de mise à disposition de ressources par les grands acteurs d'internet a vite été assimilée au terme de "Cloud Computing". Ceci est dû au fait que les ressources mises à disposition sont attribuées de manière totalement autonome et automatisée par un système informatique. Les utilisateurs ne se soucient donc plus forcément de l'emplacement physique des ressources, elles sont quelque part... "dans les nuages".

3.3 Les services d'infrastructures

Le terme de service d'infrastructure ou Infrastructure as a Service (IaaS) en anglais décrit donc un système automatisé permettant de déployer des ressources pour les mettre à disposition des utilisateurs avec le minimum d'interventions physiques possibles afin de raccourcir au maximum le temps d'attente. Un service d'infrastructure est donc composé d'une multitude de logiciels devant collaborer entre eux dans ce but.

Imaginons à présent qu'une entreprise veuille créer son propre logiciel de service d'infrastructure. Elle devra alors créer un logiciel assemblant à la manière d'un puzzle l'ensemble des logiciels gérant chaque type de ressources. Les personnes en charge du réseau LAN devront fournir soit le code, soit les spécifications permettant d'effectuer des opérations sur leurs matériels réseaux. Il en est de même pour les personnes en charge du stockage, des systèmes d'exploitations, des pare-feu, ... Sans compter que chaque type de ressource peut avoir des sous-ensembles de matériels / logiciels provenant de différents fournisseurs et n'ayant pas obligatoirement les mêmes versions au sein d'un sous-ensemble. Cela peut très vite devenir un projet très complexe, dû à l'explosion combinatoire des possibilités, impliquant un grand nombre de personnes n'ayant pas obligatoirement les mêmes attentes et objectifs.

Afin d'éviter que chaque entreprise s'aventure à recréer son propre logiciel de service d'infrastructure, des développeurs ont décidé de prendre les choses en mains. Ils se sont mis à développer des logiciels de service d'infrastructure qui soient des logiciels libres afin que chacun puisse y contribuer quel que soit sa provenance. Celui qui rencontre le plus de succès auprès de la communauté est OpenStack.

3.4 Toujours plus d'automatisation mais...

Dans un souci d'automatiser toujours plus les actions répétitives, il est maintenant possible de s'attaquer à l'automatisation de l'installation et configuration de logiciels de bases, voir des applications en elles-mêmes, dans les serveurs déployés via un service d'infrastructure.

Dans le modèle Plateform as a Service (PaaS) (Fig. 3.1) le système d'exploitation ainsi que les logiciels de bases (serveurs web, bases de données, middleware de messagerie, ...) peuvent être assemblés à la volée ou choisis parmi un catalogue et déployés sur le serveur installé au préalable. Ce modèle tente de fournir une plateforme standardisée pour le déploiement de logiciels, laissant aux développeurs la seule charge de fournir leur propre application.

Le modèle Software as a Service (SaaS) quant à lui embarque aussi directement l'application dans le système de déploiement. Un exemple de cas d'usage serait de déployer sur plusieurs serveurs le même site web afin de répartir la charge des utilisateurs se connectant à celui-ci.

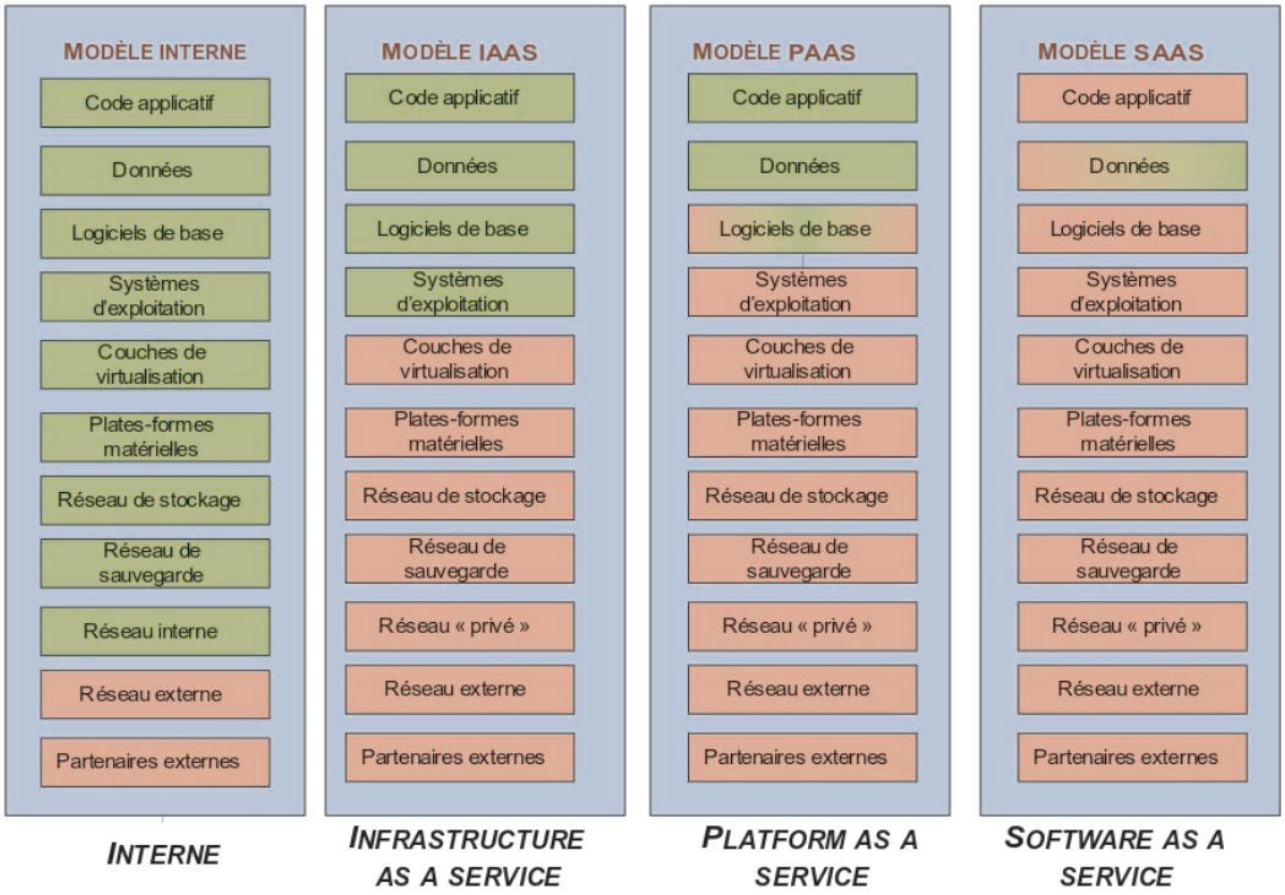

Fig 3.1 – Modèles "as a Service"

Cependant, afin d'automatiser toujours plus les déploiements, les méthodes de gestion des infrastructures doivent évoluer vers une plus grande standardisation et reproductibilité. Cela implique de réduire la complexité des infrastructures à gérer et donc de reporter la responsabilité de la résilience aux applications (Fig. 3.2). Il y a donc un fort impact sur la manière historique de développer une application.

	Orchestrated	Policy Based	Design For Fail
Proposition de valeur	Prévu pour gérer la complexité	Prévu pour gérer la complexité	Prévu pour réduire la complexité
	Résilience au niveau HW	Résilience au niveau HW et SW	Résilience Applicative
	Faible impact sur la DSI	Impact moyen sur la DSI	Important impact sur la DSI
	Faible ROI à long terme	ROI moyen à long terme	Haut ROI à très long terme
	Quelques éléments scalable	Scalable	Largement Scalable
Comment sont ils construit	Principalement Cloud privé	Publique ou Privée	Principalement publique
	Architecture hétérogène	Architecture hétérogène	Architecture uniforme (x86)
	Construit avec des SW chers	Construit avec des SW peu chers	Construit avec des SW peu chers
	Mise en œuvre fastidieuse	Mise en œuvre simple	Mise en œuvre très simple
	Dur à maintenir	Facile à maintenir	Le plus facile à maintenir
	Orchestre les services HW (security/net)	Intègre les service HW ou Virtuels (security/net)	Intègre les services virtuels (security/net)
Comment sont ils consommés	Orienté ITIL	Orienté ITIL	Orienté DevOps
	WebUI	WebUI and API	API and sometime WebUI
	Applications d'entreprise	Applications d'entreprise (x86)	Cloud applications

Fig 3.2 – Les différents types de gestion d'un Cloud

Par le passé un grand nombre d'éditeurs de logiciels, qu'ils soient interne ou externe à l'entreprise, fonctionnaient de la manière suivante :

- Ils développaient en interne leur logiciel selon une étude de marché et les spécifications fonctionnelles requisent pour ce marché.
- Le logiciel résultant était présenté et vendu aux utilisateurs finaux des entreprises.
- Les utilisateurs faisaient appel aux personnes de l'infrastructure afin de fournir les ressources nécessaires.
- Une fois l'application déployée, les utilisateurs se souciant de la résilience de leur application, contactaient le fournisseur afin de trouver une solution pour sécuriser l'application.
- Les éditeurs du logiciel, n'ayant pas prévu cet aspect en compte, demandaient aux personnes en charge de l'infrastructure de gérer la résilience.
- Les personnes gérant l'infrastructure se retrouvaient garants de la résilience d'applications mal conçues et devaient complexifier leurs infrastructures afin de pallier aux lacunes applicatives.

Le manque de communication entre les trois parties impliquées dans la mise en place de nouvelles applications a amené les entreprises à la situation actuelle : héberger un grand nombre d'applications difficilement automatisables en termes d'infrastructure. Cela a été un élément majeur dans les choix des méthodes d'automatisation de déploiements.

Un grand nombre d'applications n'est donc pas forcément compatible avec les méthodes de déploiement et d'hébergement employées dans les services d'infrastructure modernes. En effet elles ont étés conçus en partant du principe que la résilience doit être gérée par l'application et non pas par l'infrastructure. Si l'on a besoin de résilience ou de plus de performance, au lieu d'augmenter la capacité du serveur et de créer des infrastructures complexes, on déploie plusieurs instances du même logiciel sur plusieurs machines. L'application ayant en charge l'orchestration et la redondance de ses instances. Un changement fondamental de méthodes de conception des applications est donc nécessaire (Fig. 3.3).

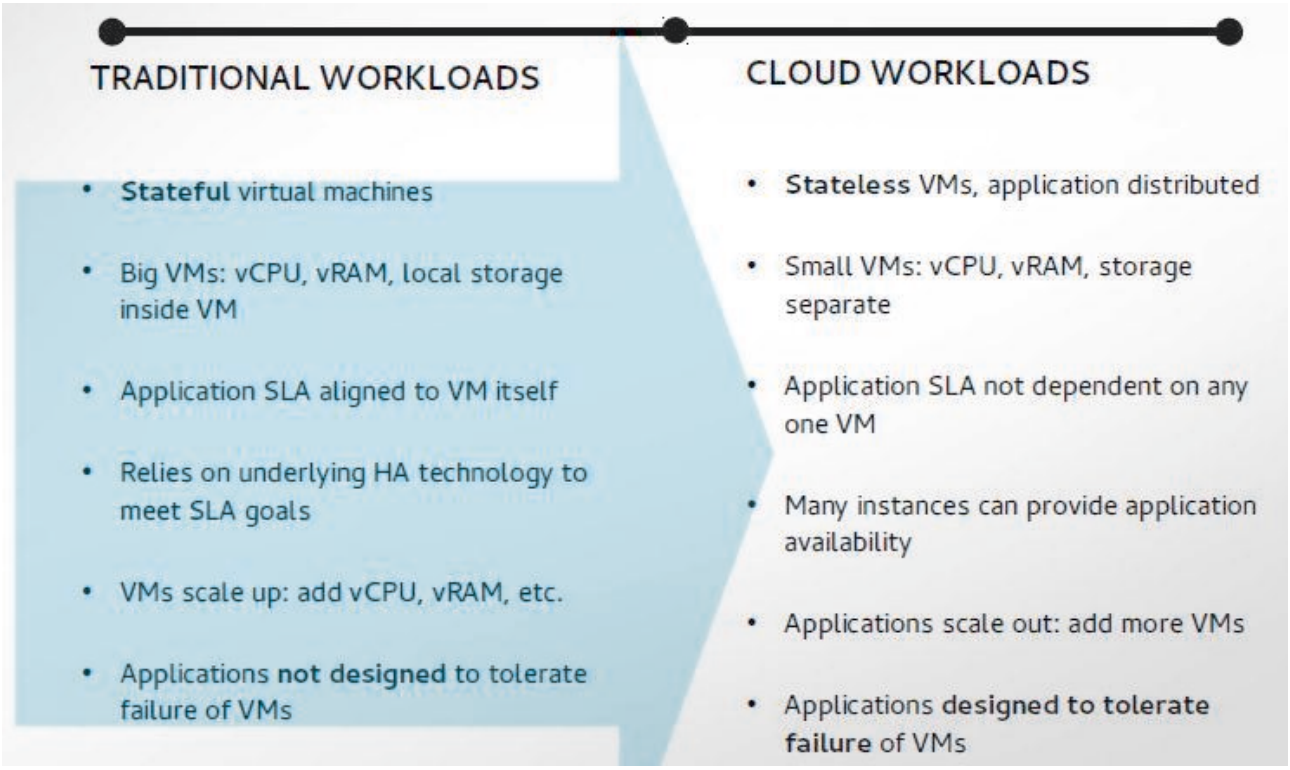

Fig 3.3 – Les prérequis des services d'infrastructures modernes

Chapitre

4 L'infrastructure actuelle chez PSA

4.1 Contexte et historique

Historiquement chez PSA, les équipes en charge de l'infrastructure sont séparées par domaine de compétences. On retrouve les équipes suivantes :

- Quatre équipes en charge des systèmes d'exploitation (une par OS) : Aix, Linux, Solaris et Windows.
- Une équipe en charge des systèmes de stockage NAS et SAN.
- Une équipe en charge du réseau LAN.
- Une équipe en charge des logiciels communs.
- Une équipe en charge des sauvegardes.

Afin de réaliser l'installation complète d'un serveur chacune de ces équipes devaient intervenir à tour de rôle. Au sein de chaque service l'automatisation des tâches était variable : certaines équipes faisait tout manuellement, d'autres avaient un haut niveau d'automatisation.

Pour réduire le temps de mise à disposition des serveurs et par là même occasion réduire les coûts de déploiement, un premier projet a été lancé en 2010 : le projet "Infra 2.0". Le but de ce projet était de créer un système précurseur de service d'infrastructure. Une équipe a été désignée comme responsable de la coordination et du développement et assemblage des différents éléments. Pendant ce temps une équipe tierce était chargée de développer un portail d'accès au système pour les utilisateurs. Ce portail ayant pour but de centraliser et orchestrer les demandes utilisateurs, ainsi que les informer sur les coûts liés à leurs demandes.

Les premières équipes impliquées dans ce projet (outre celle de coordination) furent celles gérant les systèmes d'exploitation et celle gérant le stockage NAS et SAN. Le projet faisant ses preuves en termes de gains de temps, d'autres équipes ont étés sollicitées telle que l'équipe du réseau LAN, celle de la sauvegarde et celle des logiciels de base.

Ce système de service d'infrastructure, actuellement toujours utilisé au sein de PSA, est un service d'infrastructure "orchestré" (Fig. 3.2) au sens ou il est prévu pour gérer une infrastructure complexe et hétérogène. Nous allons voir à présent comment ce système est architecturé et quels sont ses points forts et faibles.

4.2 Le portail utilisateur COMUT

Le premier élément important du projet "Infra 2.0" est le portail COMUT. Il s'agit d'un site web interne à PSA permettant à l'utilisateur d'enregistrer des demandes d'architectures nécessaires pour le déploiement d'applications. Les technologies mises en œuvre pour sa réalisation sont le langage PHP et Flash.

L'utilisateur, via différents assistants, se voit offrir un catalogue d'architectures prédéfinies ou, le cas échéant construit une architecture manuellement (Fig. 4.1).

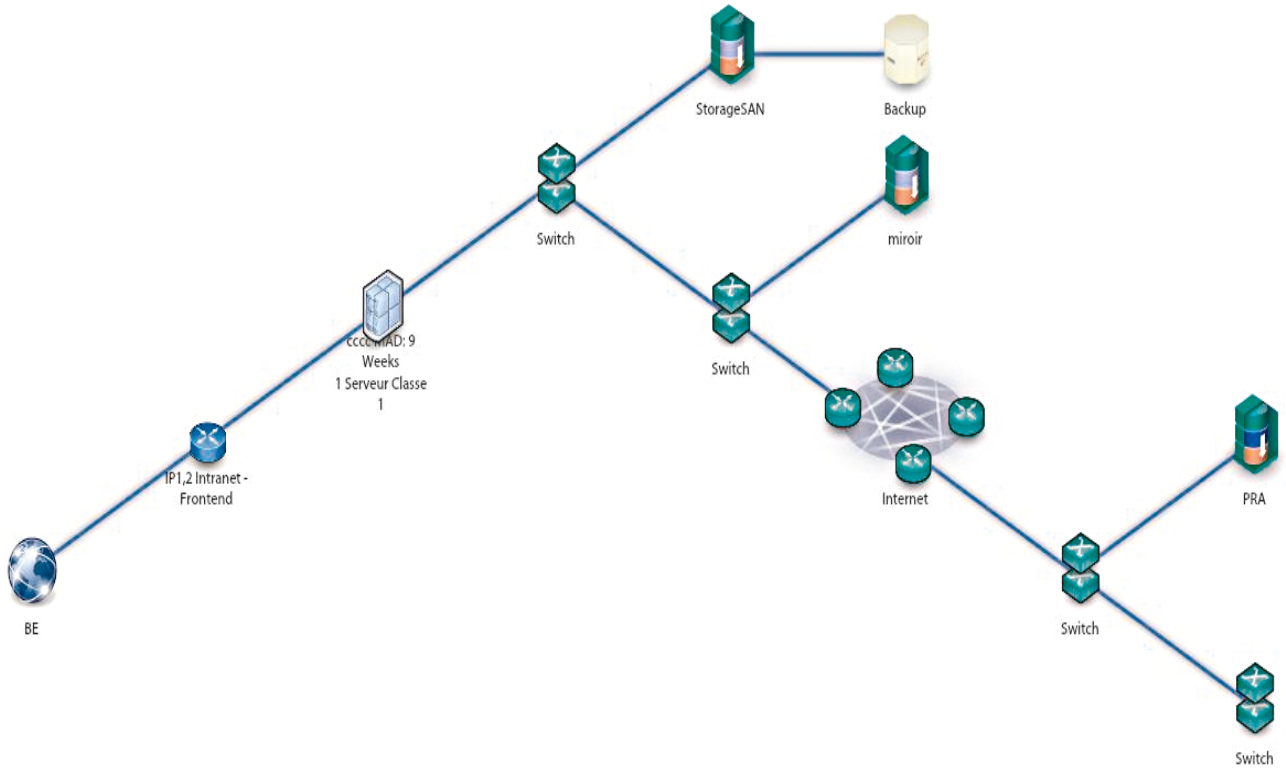

Fig 4.1 – Exemple d'architecture générée par le portail COMUT

Une fois la demande de l'utilisateur enregistrée, celle-ci entre dans un cycle de validation. En fonction des spécificités de la demande, plusieurs équipes gérant l'infrastructure vont valider ou refuser la demande pour l'affiner. Ces équipes ont défini un certain nombre de critères permettant de réduire la durée du cycle en validant certaines étapes automatiquement.

Pour l'attribution du stockage au serveur, voici un exemple de critère permettant l'éligibilité de la demande à l'automatisation. Si la demande de l'utilisateur en termes de stockage est inférieure à 500Go, celle-ci sera validée sans intervention des administrateurs du stockage. Parmi les différents critères définis par chaque équipe, il suffit qu'un seul d'entre eux ne soit pas respecté pour que la demande ne soit plus éligible à l'automatisation.

Le portail COMUT vérifie les critères d'éligibilité et de validation automatique à l'aide de données provenant de différents référentiels métiers (Fig. 4.2). Chaque équipe de l'infrastructure a ses propres référentiels dont les données sont stockées de manière différente et accessibles via différents mécanismes.

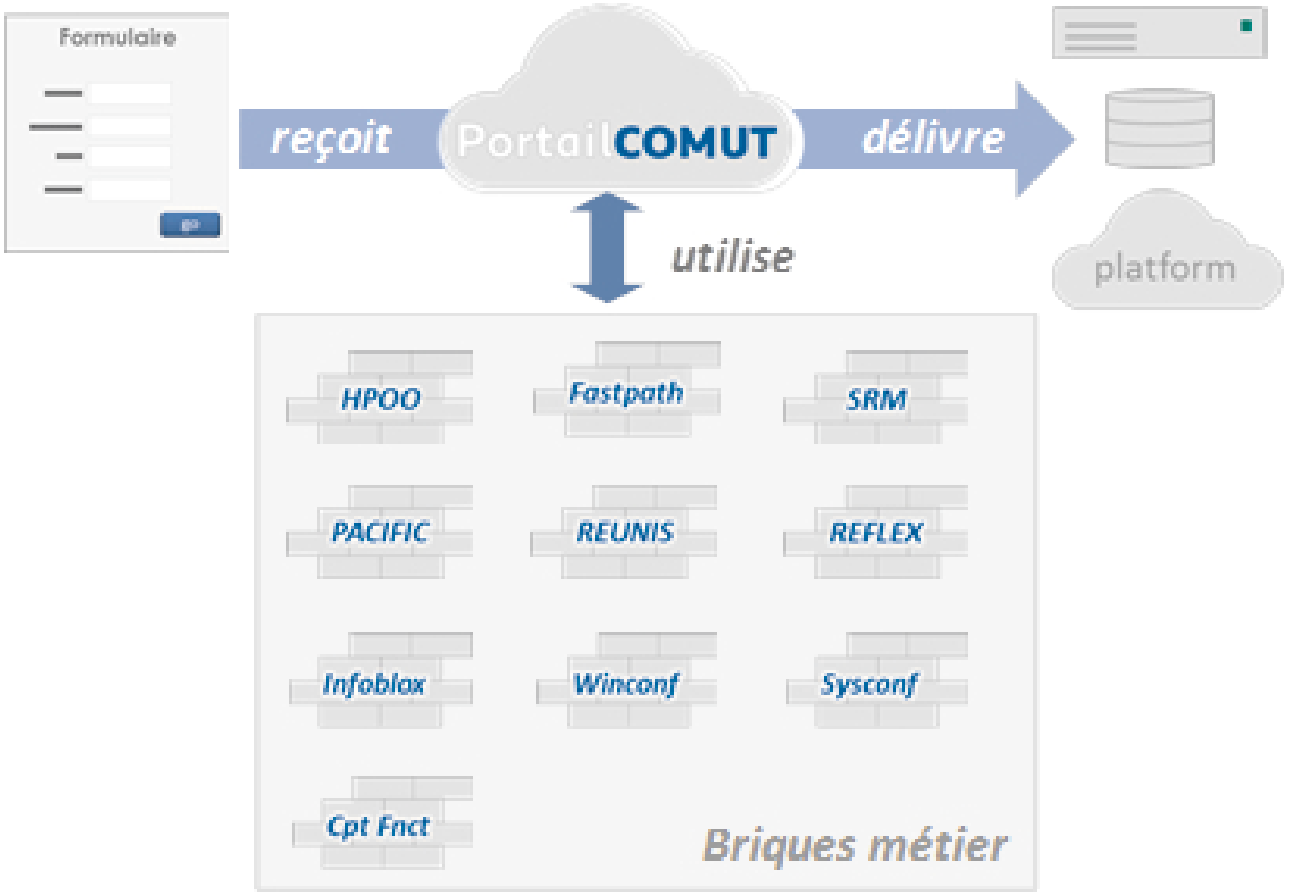

Fig 4.2 – Architecture du Portail COMUT

Lorsque la demande d'architecture est entièrement validée il peut se produire deux événements :

- Si l'architecture demandée n'est pas éligible à l'automatisation le portail COMUT va initier l'ouverture d'un ticket auprès de l'outil PSA de gestion des interventions. Les opérations nécessaires à la mise en place de l'architecture seront donc réalisées manuellement par les différentes équipes. Les délais de réalisation de la demande peuvent aller jusqu'à neuf semaines dans ce cas.
- Si l'architecture demandée est éligible à l'automatisation le portail COMUT va transmettre la demande à une autre application en charge de la création automatique de l'architecture : FastPath. Les délais de réalisation de la demande seront généralement de quelques heures à 48 heures selon le cas.

4.3 Le service d'infrastructure FastPath

FastPath est un service d'infrastructure développé en interne par PSA. Il est constitué d'un ensemble de scripts Shell permettant de mettre à disposition les différents éléments nécessaires à la création d'une machine virtuelle. Au sein de l'automatisation il a le rôle d'orchestrer les demandes venant du portail COMUT.

FastPath reçoit donc régulièrement des demandes de création ou de mise à jour d'architecture en provenance du portail COMUT. Ces demandes sont décrites sous la forme d'un fichier "ADN" qui n'est autre qu'un fichier texte décrivant les besoins enregistrés dans le portail COMUT.

Lorsqu'une demande d'architecture arrive dans FastPath, celui-ci découpe la demande en morceaux élémentaires (Fig. 4.3) et, après analyse, il fait appel aux scripts nécessaires afin de traiter la demande. Les scripts appelés remplissent un ensemble de fonctions telles que la création d'une adresse IP, l'ajout de RAM, de CPU ou encore la création du stockage pour une machine virtuelle.

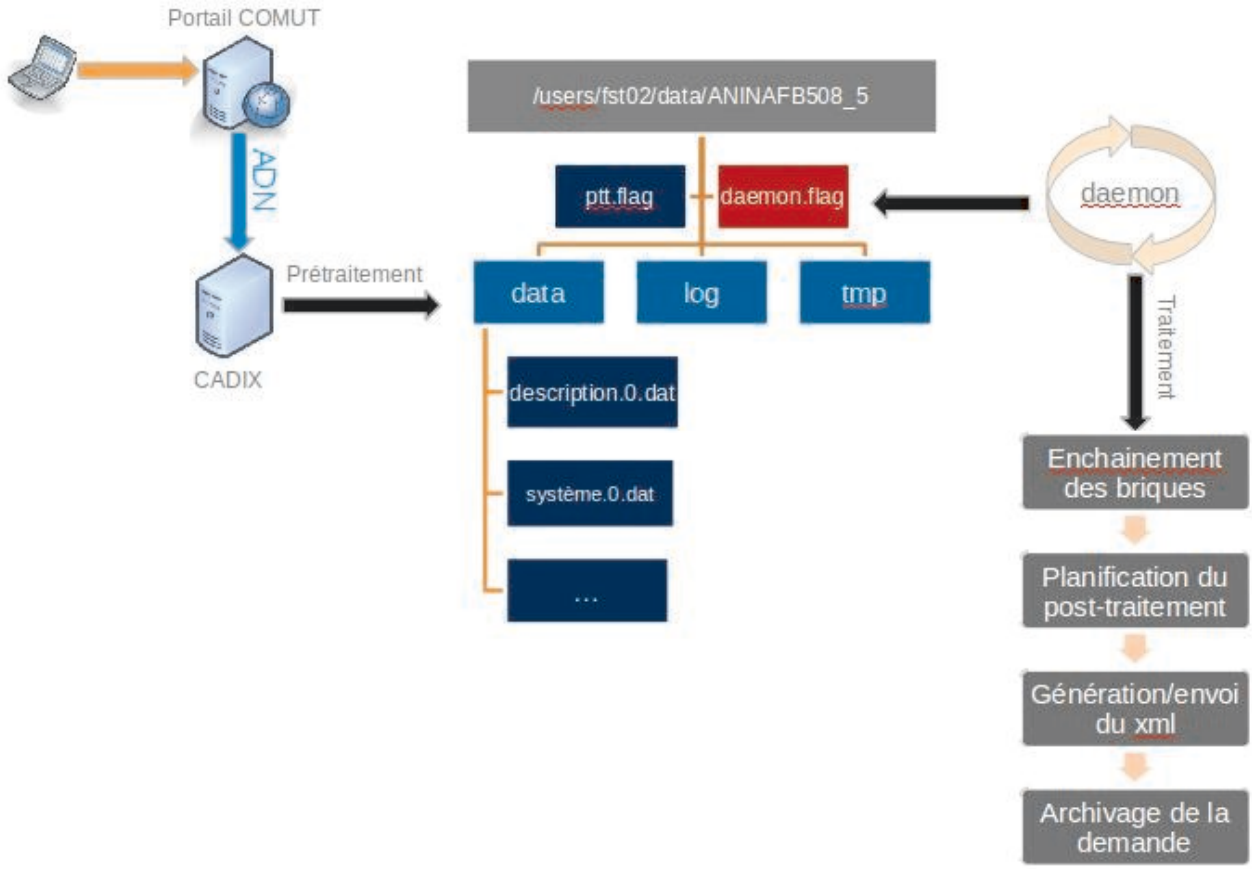

Fig 4.3 – Architecture de FastPath

Chaque équipe au sein de l'infrastructure fournit ses propres scripts à FastPath et définit une interface d'appel de ceux-ci selon sa convenance. La maintenance et l'adaptation de ces scripts incombe donc aux équipes de l'infrastructure. Ceux-ci nécessitent donc de constantes modifications en fonction de l'évolution des logiciels et matériels.

En ce qui concerne les systèmes d'exploitation, FastPath a été conçu de manière à pouvoir gérer Windows Server, Linux et Solaris. Le niveau d'automatisation des tâches chez les équipes systèmes varie beaucoup. Les équipes en charge de Linux et Solaris ont fortement automatisé le déploiement de leur système d'exploitation, tout en permettant facilement l'analyse en cas de dysfonctionnements. Les équipes en charge de Windows ont quant à elles relativement peu automatisé les tâches. De plus leur système d'automatisation reste obscur d'un point de vue de l'infrastructure. En cas de dysfonctionnement il faut systématiquement les contacter afin de savoir de quoi résulte le problème.

Pour la partie stockage je suis, depuis le début du projet en 2010, la personne qui fournit l'ensemble des scripts permettant l'ajout, l'agrandissement et le retrait d'espaces de stockage. J'ai donc en charge depuis près de cinq années les spécifications, le développement et les recettes de l'ensemble de l'automatisation concernant le stockage SAN. Le stockage NAS étant en cours d'intégration dans FastPath. Afin d'assurer une cohérence entre le portail COMUT et FastPath, je suis aussi chargé de faire les maquettes des formulaires concernant le stockage qui sont présentées à l'utilisateur final dans le portail COMUT.

4.4 Bilan

Mes nombreuses contributions aux projets du portail COMUT et FastPath m'ont permis d'acquérir une expertise des outils d'automatisation de l'infrastructure chez PSA. Ces connaissances me permettent de tirer des enseignements de ces projets après cinq années d'exploitation.

Le portail COMUT a l'avantage indéniable d'être conçu pour répondre aux besoins spécifiques de notre environnement. En effet, l'ensemble des règles et restrictions de chaque équipe y est implémenté afin de respecter au mieux les politiques internes. Les termes utilisés dans l'interface de celui-ci sont très spécifiques à l'environnement PSA, peut-être trop même... Certains utilisateurs ont parfois du mal à comprendre les informations qui leur sont demandées. L'interface en elle-même souffre de ralentissements dus, en partie aux technologies web vieillissantes qu'elle utilise, et en partie aux nombreuses requêtes externes d'informations.

Afin de garantir la cohérence des informations dans le portail COMUT, celui-ci interroge donc un ensemble de référentiels externes. Cependant aucune normalisation des méthodes d'accès aux données n'a été définie. Il en résulte que certains référentiels sont accédés via des requêtes de bases de données, d'autres via des services web ou encore des copies de fichiers à travers différents protocoles. L'architecture de l'application est donc très complexe. De plus la plupart des référentiels ne sont pas capable de fournir des informations de la situation de l'infrastructure en temps réel. Une majorité de ceux-ci fournissent des informations de la veille. Finalement il faut aussi retenir que ses différents référentiels échangent des données entre eux de manière journalière et non concertée (Fig. 4.4). On peut facilement se retrouver avec un système où les différents référentiels n'ont pas de vision cohérente de l'infrastructure à un instant donné.

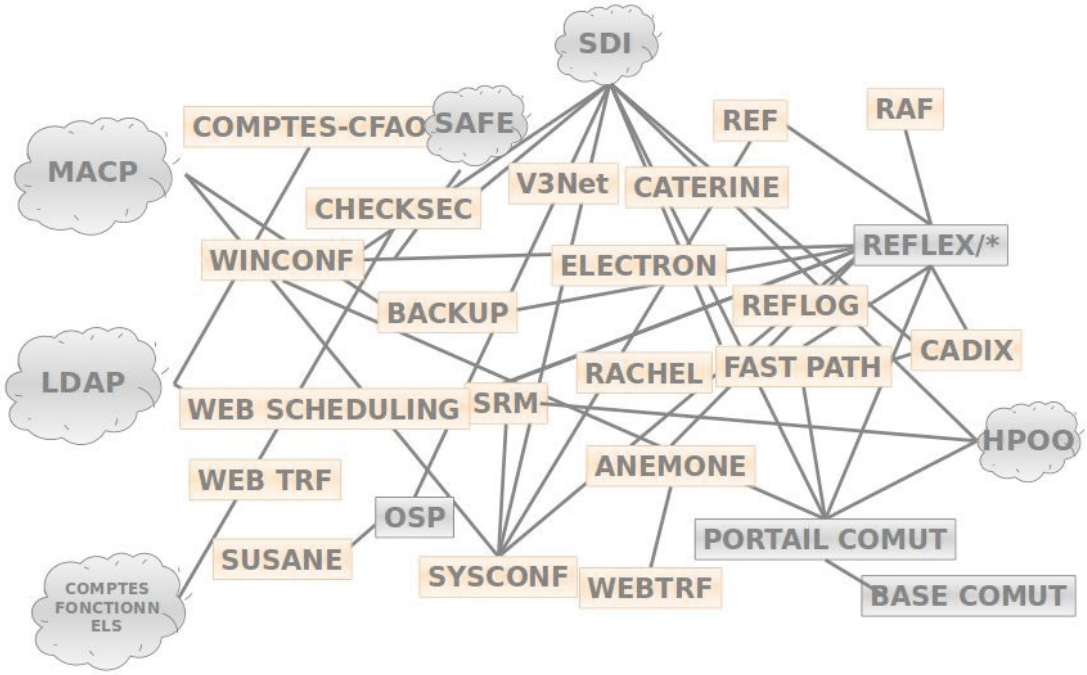

Fig 4.4 – Relations inter-référentiels

Finalement le format d'échange des données (le fichier "'ADN') entre le portail COMUT et FastPath n'est pas standardisé. Il s'agit d'un fichier texte pseudo-structuré sans réelle notion de typage. Il n'y figure pas de notion de types normalisés telles que les tableaux, les tables de hachage ou encore les booléens. Ce format de représentation de données est l'exemple typique d'une invention interne par manque de connaissance des standards du marché.

En ce qui concerne le service d'infrastructure FastPath, son principal point fort est qu'il est capable de gérer plusieurs systèmes d'exploitation différents. Par contre il possède plusieurs faiblesses dont la principale est le fait que les scripts le constituant sont en majorité maintenue par une seule personne par domaine d'activité. Si une personne ayant en charge la maintenance de scripts est amenée à quitter son service, les dits scripts se retrouveraient orphelins à moins qu'une autre personne du service reprenne le flambeau. Il s'agît plus d'une faiblesse d'organisation du projet que d'une faiblesse technique.

La principale dette technique de FastPath provient de son utilisation du langage de script Shell. Ce langage permet difficilement d'exploiter des formats d'échanges standardisés tels que Extensible Markup Language (XML) ou JavaScript Object Notation (JSON). Cela a été un facteur déterminant dans la décision de réinventer un format d'échange interne à PSA dépourvu de typage. Il a certainement aussi conditionné le fait que les appels aux scripts élémentaires d'actions sur l'infrastructure soient non normalisés.

Le dernier point impliquant des difficultés n'est pas à proprement parler un problème de FastPath en lui-même, mais plus un problème d'architecture. Lorsqu'une nouvelle technologie fait son apparition dans l'infrastructure PSA, il devient rapidement nécessaire d'automatiser son administration pour ne pas ralentir les délais de mise à disposition. Le service concerné va devoir développer cette automatisation pour FastPath, mais devra auparavant intégrer cette technologie à l'ensemble des référentiels internes. Ces multiples dépendances à des référentiels externes impliquent un long cycle de développement. Un exemple typique est l'introduction d'un nouveau système de stockage. Via mon expérience passée sur les développements du domaine du stockage, nous avons estimé à plus de 100 hommes/jours le travail nécessaire afin de réaliser une intégration complète d'un nouveau matériel. Cela peut représenter, en fonction des charges annexes au développement, jusqu'à plus d'un an de décalage entre la mise en production de la nouvelle technologie et son automatisation. Ces difficultés de maintenance, d'évolution et les méthodes de fonctionnement illustrent bien les lacunes d'un service d'infrastructure de type "orchestré".

Malgré ces lacunes le projet "Infra 2.0" est un succès. Il a permis d'automatiser une majorité des tâches d'infrastructure qui nécessitaient auparavant des interventions des administrateurs à faible valeur ajoutée. Il illustre bien le principe de Pareto [30] (ou loi des 80-20) qui stipule que 80% des objectifs d'un projet sont réalisés en 20% du temps alloué à celui-ci. Il fait clairement aussi ressortir que, les développements se complexifiant, les 20% d'automatisation manquant prennent à eux seuls 80% du temps. Le temps et les budgets de développements étant de plus en plus restreints, il a été décidé d'évaluer d'autres possibilités afin de faire évoluer le service rendu par l'infrastructure chez PSA. Une des solutions proposée et retenue est de remplacer petit à petit notre service d'infrastructure interne par un projet externe majeur : OpenStack.

Chapitre

5

OpenStack : Le service d'infrastructure de demain

5.1 Historique du projet

Fin 2009 la NASA cherchait un moyen de fournir une plateforme unique d'hébergement pour que ses développeurs évitent de recréer chacun dans leur côté une platefome différente [17]. Après avoir travaillé quelques mois sur un prototype d'architecture Plateform as a Service (PaaS), ils se sont vite rendu compte qu'il fallait déjà commencer par automatiser les couches inférieures d'infrastructure avant tout.

Au même moment est apparu un besoin de stockage massif d'images satellites au sein de la NASA. La NASA a donc profité de l'occasion pour répondre aux deux besoins par le même projet. Ce projet de service d'infrastructure a été nommé "Nebula". Les budgets étant restreints, l'équipe en charge de ce projet était constituée de huit personnes dont la plupart n'y était pas consacré à temps plein et l'infrastructure matérielle était hébergée dans un conteneur situé à Ames Research Center en Californie (Fig. 5.1). L'équipe pris dès le début du projet l'initiative de réaliser ce projet à l'aide de logiciels libres et de distribuer leurs travaux sous la même forme.

Fig 5.1 – Infrastructure physique de Nebula

L'équipe décida de rendre le logiciel le plus modulaire possible afin de bien séparer les différents éléments à gérer dans une infrastructure. En l'espace d'un weekend l'équipe écrira le premier module de Nebula : Nova. Nova était le module en charge de gérer les hyperviseurs afin d'y créer des machines virtuelles. Nova fut mis en production chez la NASA en mai 2010.

Rackspace, un hébergeur de site web américain, ayant eu vent du projet décida de contacter la NASA afin de déterminer si une collaboration était possible. En juillet 2010 Rackspace décida donc de distribuer, sous forme de logiciel libre, le code source de son système d'hébergement de fichiers qui sera par la suite nommé Swift [24]. La NASA et Rackspace annoncèrent conjointement le début du projet de service d'infrastructure "OpenStack".

5.2 Un projet très dynamique

Le projet OpenStack va très vite attirer un grand nombre de contributeurs, qu'ils soient amateurs ou au sein d'entreprises fournissant des services autour d'OpenStack. Le projet a eu un tel succès qu'il a dû mettre en place, en septembre 2012 [22], une fondation pour en gérer la promotion, le financement ainsi que l'organisation de conférences. Après cinq années, la communauté OpenStack est la deuxième plus grosse communauté après le noyau Linux. Elle ne compte aujourd'hui pas moins de 500 entreprises et plus de 27,000 membres (Fig. 5.2) [23].

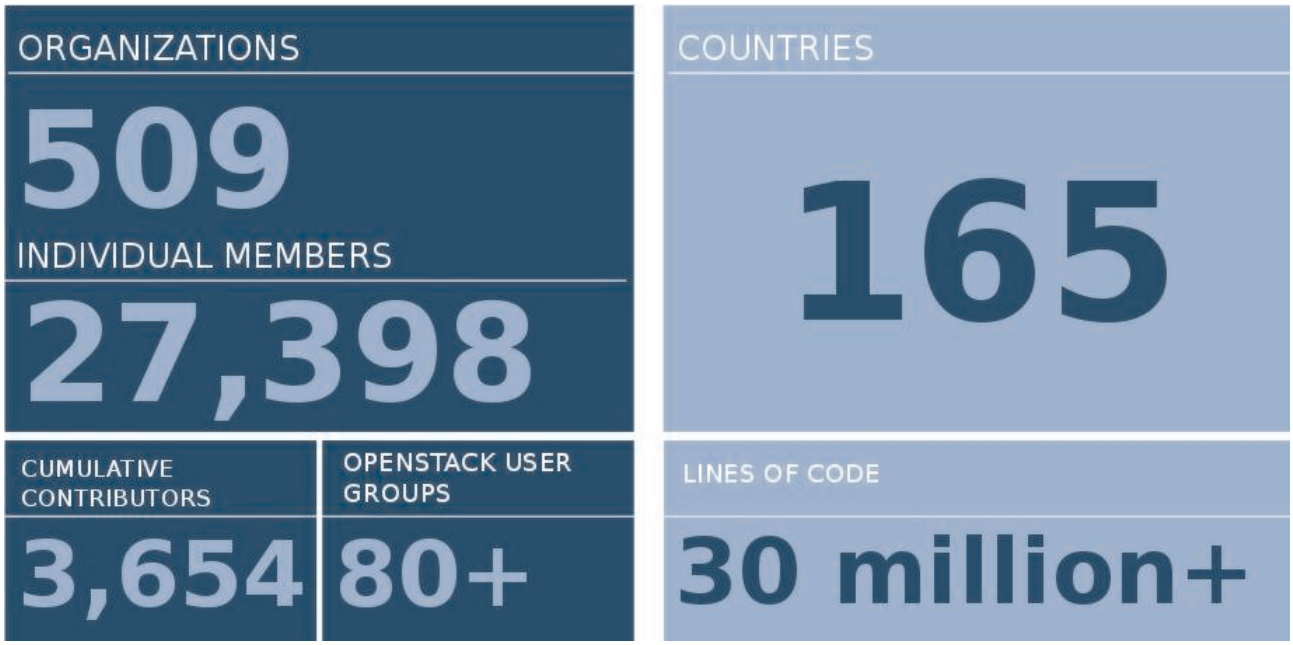

Fig 5.2 – La communauté OpenStack

La progression fulgurante du nombre de développeurs (Fig. 5.2 [18], plus de 3600 aujourd'hui) ainsi que la taille gigantesque du projet (plus de 30 millions de lignes de code), n'auraient pas été possible sans un cycle de développement et de livraison accéléré.

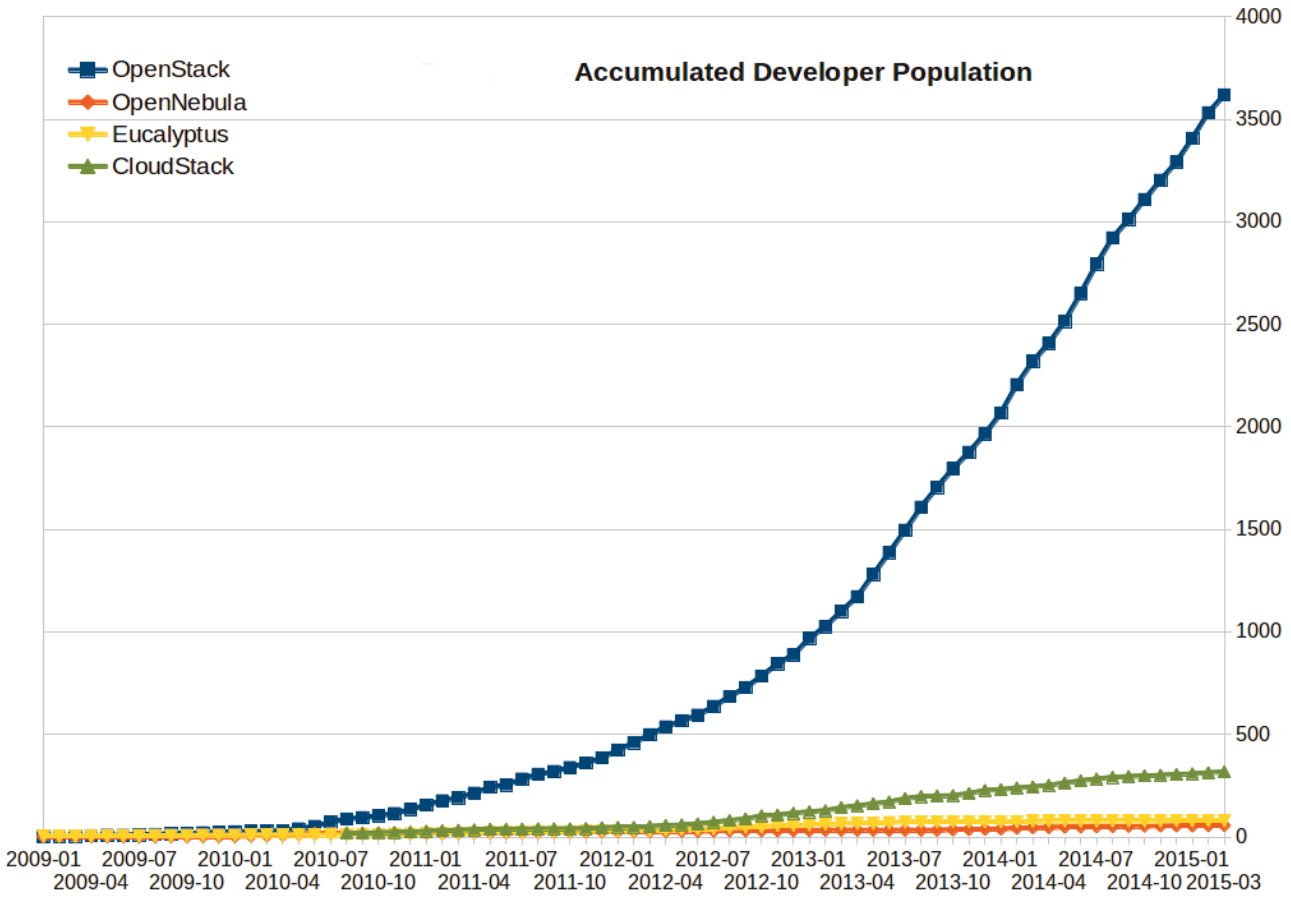

Fig 5.3 – Evolution du nombre de contributeurs au projet OpenStack

Depuis les débuts du projet il a été décidé de distribuer une version stable d'OpenStack régulièrement. Aujourd'hui le rythme de sortie des versions stables est établi à une version tous les 6 mois, rendant les plannings de mises à jour prévisibles. Quasiment chaque version a apporté de nouveaux composants à l'écosystème (Table 5.1).

Date de sortie	**Nom de code**	**Nouveaux composants**
21 octobre 2010	Austin	Nova, Swift
3 février 2011	Bexar	Glance
15 avril 2011	Cactus	
22 septembre 2011	Diablo	
5 avril 2012	Essex	Horizon, Keystone
27 septembre 2012	Folsom	Quantum, Cinder
4 avril 2013	Grizzly	
17 octobre 2013	Havana	Neutron, Heat, Ceilometer
17 avril 2014	Icehouse	Trove
16 octobre 2014	Juno	Sahara
30 avril 2015	Kilo	Ironic
16 octobre 2015	Liberty	Zaqar, Manila, Designate, Barbican
Avril 2016	Mitaka	?

Table 5.1 – Historique des versions d'OpenStack

A chaque sortie d'une nouvelle version, les utilisateurs et développeurs se rencontrent lors d'une conférence afin de présenter leurs travaux passés et discuter du planning de la prochaine version. M. DURUPT, M. DUCHENOY et moi-même avons participé à la conférence d'octobre 2014 à Paris au Palais des Congrès. L'ampleur de la conférence est proportionnelle aux ambitions du projet : plus de 5000 personnes étaient rassemblées sur les cinq jours de la conférence. Cette conférence m'a permis de peaufiner mes connaissances des différents composants d'OpenStack, ainsi que de prendre des contacts auprès des développeurs d'OpenStack gérant la partie du stockage.

5.3 Les composants

OpenStack est un service d'infrastructure modulaire développé en grande partie dans le langage de programmation python. Le langage python est bien adapté à l'administration des systèmes et de l'infrastructure. Il comprend un très grand nombre de modules utiles aux administrateurs et développeurs et est assez facile d'approche. Bien qu'étant très modulaire OpenStack réutilise, quand cela est possible, du code qui a été mis commun par les différents "modules" ou "projets" qui le composent.

Chaque projet dans OpenStack a un rôle bien précis (Fig. 5.4). Parmi les projets principaux on retrouvera :

- Nova : le gestionnaire d'hyperviseurs.
- Glance : le dépôt d'images des systèmes d'exploitation.
- Keystone : le gestionnaire d'authentification.
- Cinder et Swift : le fournisseur d'espace de stockage en mode block et celui en mode objet.
- Neutron et Designate : la gestion du réseau et des réservations des noms de domaine.
- Ironic : la gestion des machines physiques.
- Ceilometer : la centralisation des informations de performances.
- Horizon : le portail web.
- Barbican : le gestionnaire de clés et certificats.
- Heat : l'orchestrateur.
- Trove et Sahara : le gestionnaire de bases de données et celui d'architecture BigData Hadoop.
- Zaqar : le fournisseur de file de messages.

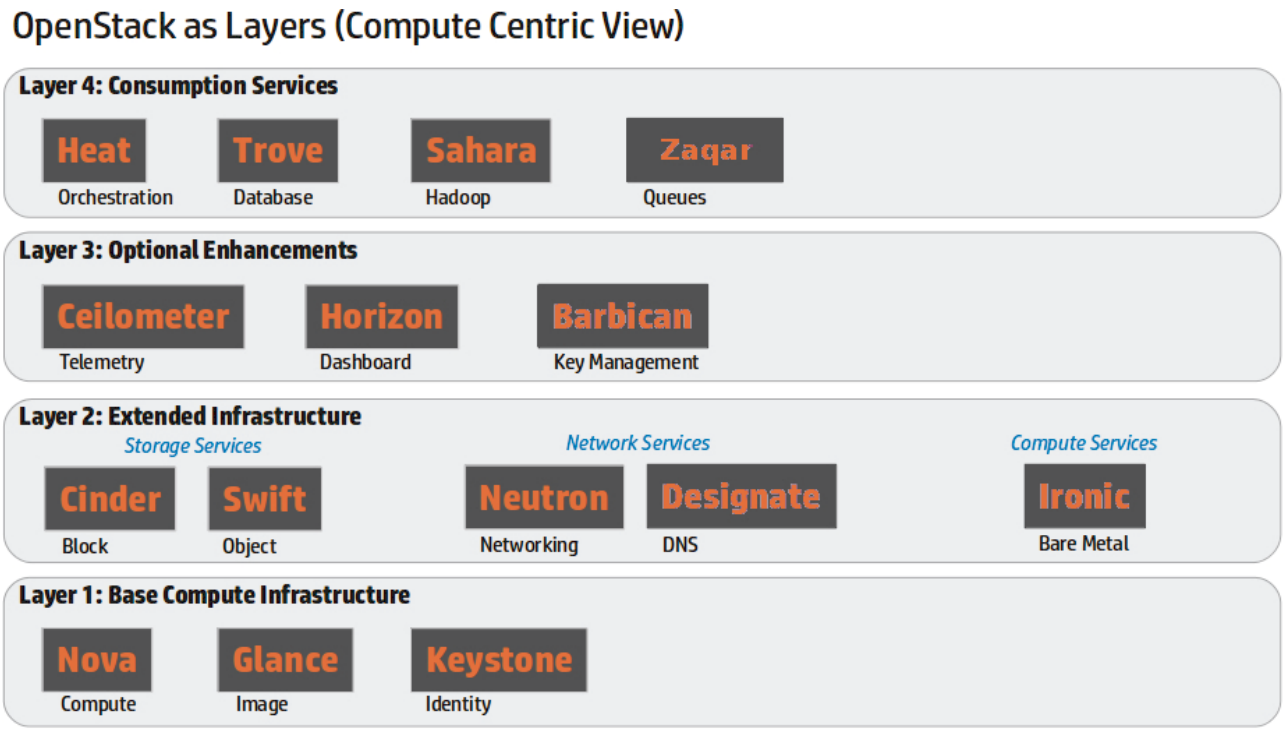

Fig 5.4 – Les couches logiciels d'OpenStack

Un projet traite principalement des données de trois manières différentes (Fig. 5.5) :

- Le projet doit stocker ses données.
- Des données sont échangées entre les différents composants d'un projet.
- Le projet échange des données avec d'autres projets extérieurs.

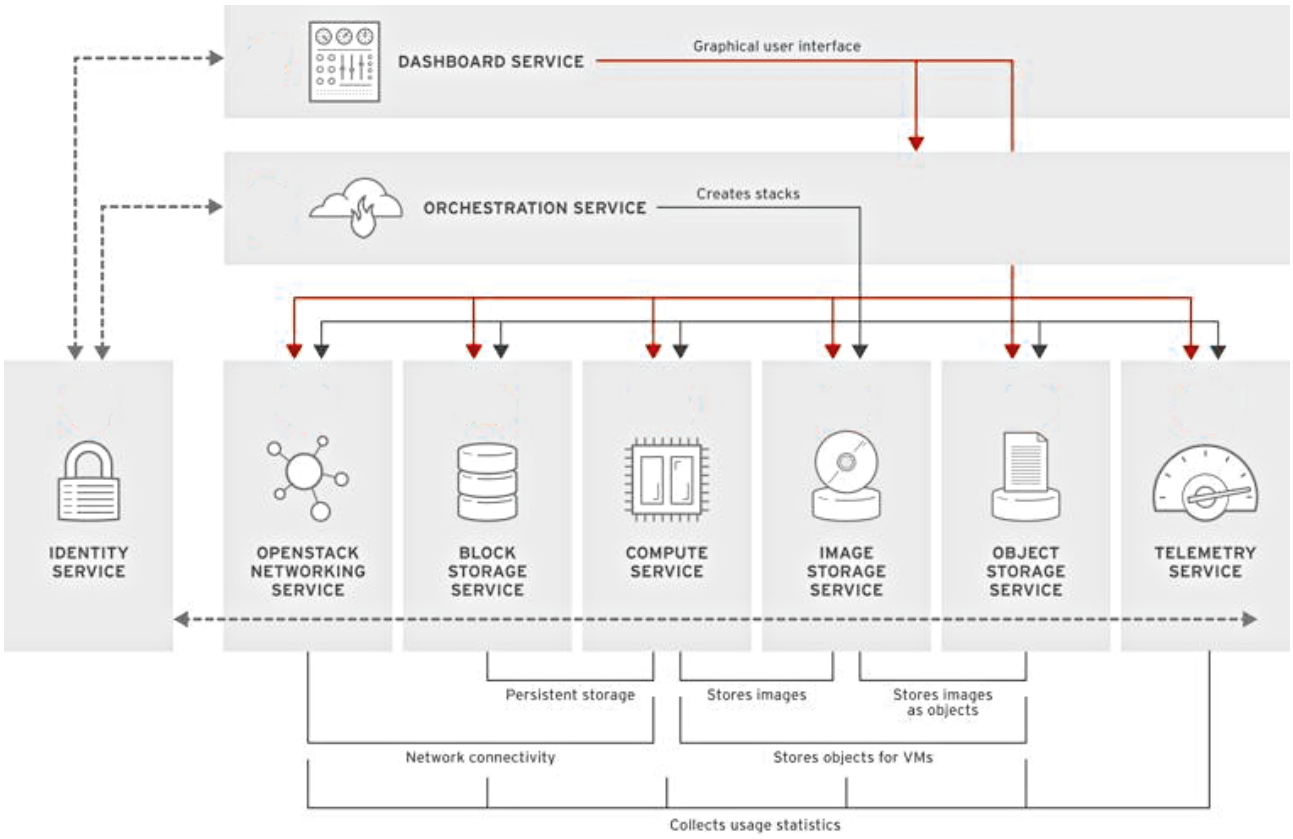

Fig 5.5 – Les interactions dans OpenStack

Afin de gérer les données nécessaires à un projet, il lui est généralement dédié une base de données. La base de donnée la plus couramment utilisée dans les projets d'OpenStack est MySQL [16].

Pour s'échanger des données entre eux, les composants d'un projet utilisent des files de messages. Elles sont utilisées car elles ont l'avantage de préserver l'ordre d'émissions des messages et de gérer les verrous d'accès aux données. Il ne serait pas très judicieux d'utiliser une base de données pour ce rôle car il faudra créer un mécanisme de verrous d'accès aux données qui serait fastidieux à implémenter. Le logiciel de files de messages le plus commun dans OpenStack est RabbitMQ [31].

L'échange des données vers l'extérieur se fait via des Application Programming Interface (API) standardisées. Les API sont accessibles via le protocole HTTP et échangent des données sous le format JSON.

Ces trois méthodes d'interrogation des données permettent aux projets d'OpenStack de fournir en temps réel les informations concernant chaque domaine de l'infrastructure.

Chaque projet d'OpenStack a donc la mission de collecter, référencer, modifier et échanger les données relatives à son domaine de spécialisation. Cependant, afin d'y parvenir, il faut pouvoir communiquer avec les logiciels ou matériels sous-jacents. Dans tous les domaines (hyperviseurs, réseau, stockage, ...) les logiciels et matériels fournissant le même service sont foison. Pour éviter que le code d'un projet devienne un entremêlement inextricable de cas particuliers, le projet implémente une couche d'abstraction. Cette couche d'abstraction définie un certain nombre de fonctionnalités que les modules gérant les matériels ou logiciels doivent implémenter pour prétendre à être intégré au projet. Les fournisseurs de logiciels et matériels ont donc en charge la conformité et l'écriture des modules leur appartenant.

L'avantage principal de ces couches d'abstraction est qu'il est facile de changer de module sous-jacent. Cela permet à chaque domaine de l'infrastructure de ne plus être dépendant d'un fournisseur en particulier. En effet, il n'est pas rare que des entreprises, ayant développé leur propre automatisation, ne peuvent que très difficilement changer de fournisseur. Ceci car elles ont utilisé des fonctionnalités propres à un fournisseur.

De plus, pour éviter toute dépendance à des fournisseurs spécifiques, un module de référence est maintenu pour chaque projet. Ce module est une pure implémentation logicielle et ne requiert pas de matériels particulier. Cette implémentation de référence est un logiciel libre qui permet d'avoir dans tous les cas le choix entre utiliser un logiciel libre ou utiliser un autre module non libre. Le choix de l'implémentation à utiliser se fait simplement en éditant les fichiers de configuration du projet.

5.4 Les distributions

OpenStack étant constitué d'une multitude de composants, il n'est parfois pas facile de l'installer sans en comprendre les tenants et aboutissants. Afin de faciliter l'installation, les différentes distributions construisent des architectures de références et fournissent des outils d'installation simplifiés.

Il existe deux types de distributions :

- Les distributions libres communautaires et gratuites.
- Les distributions commerciales payantes.

Les avantages principaux des distributions commerciales par rapport aux distributions communautaires sont :

- La facilité d'installation des architectures de référence.
- L'apport d'expertise dans la conception d'architectures.
- L'accès au support fournisseur en cas de problèmes.

Elles ont cependant aussi quelques inconvénients :

- Les outils d'installation sont parfois très rigides et certaines options ne sont pas prévues.
- Le support fournisseur est souvent limité aux configurations de référence : on ne peut pas tout faire et espérer que le fournisseur supporte notre configuration.
- Certains fournisseurs essayent de détourner la philosophie d'OpenStack en remplaçant des composants parfaitement fonctionnels par leurs produits non libres pour tenter de vendre des licences logicielles qui ne sont pas nécessaires et rendre le client dépendant des logiciels du fournisseur.
- Il est parfois plus fastidieux d'essayer de faire comprendre au support du fournisseur un problème que de contacter directement les développeurs dans la communauté et leur exposer le problème.
- Il n'y a pas, pour le moment, de consensus chez les fournisseurs et la communauté sur les outils à utiliser pour l'installation d'OpenStack.

Dans un premier temps, de janvier à décembre 2014, nous avons testé la version communautaire d'OpenStack via la distribution OpenSUSE [26], puis la version commerciale SUSE Cloud [29]. Nous avons aussi suivi une formation de cinq jours sur SUSE Cloud en septembre 2014 afin de mieux comprendre les outils d'installation. Par la suite, à partir de janvier 2015, nous avons testé une deuxième distribution commerciale d'OpenStack : HP Helion [25].

Nous avons choisi de faire un appel d'offre qui a débuté en mai 2015 auquel sept fournisseurs ont répondu. J'ai participé aux différentes présentations des fournisseurs et j'ai aidé au dépouillement et analyse des réponses. PSA annoncera en décembre 2015 quel fournisseur sera retenu pour le marché.

5.5 Architecture technique

5.5.1 La partie "contrôleur"

Avant tout OpenStack est un service d'infrastructure de type "design for fail". C'est-à-dire qu'il considère que tout élément de l'infrastructure peut faillir et que les applications qui seront hébergées sur cette infrastructure devront être redondantes. Ce choix d'architecture vient du fait que les premières entreprises utilisatrices l'utilisaient principalement pour des hébergements de sites web.

Il est relativement simple de rendre un site web redondant aux pannes de matérielles : il suffit de mettre en place le site web sur plusieurs serveurs virtuels et mettre un ou plusieurs répartiteurs de charge en frontal. Cela permet de répartir la charge et de pallier aux défaillances des serveurs. Il en est de même pour les données des sites web : il faut installer un base de données en mode cluster avec un ou plusieurs répartiteurs de charge entre le site web et la base données (Fig. 5.6).

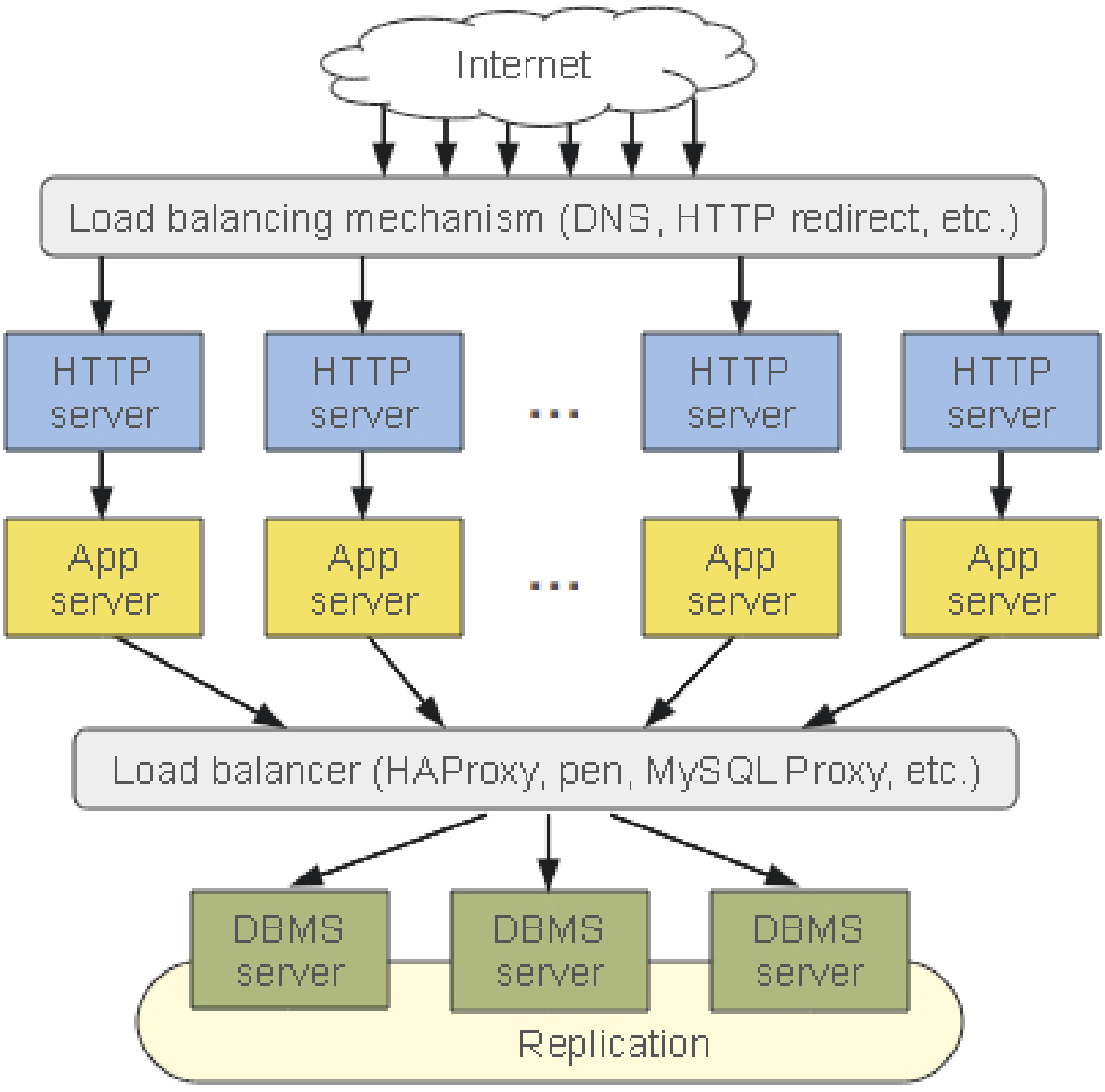

Fig 5.6 – Architecture redondante pour un site web

On retrouvera une architecture technique analogue dans la partie contrôleur d'OpenStack. En effet, OpenStack, comme tout service d'infrastructure discerne deux parties :

- Les serveurs comportant les composants pilotant l'infrastructure : la partie "contrôleur".
- Les serveurs hébergeant les machines virtuelles déployés par la partie contrôleur : la partie "compute".

La partie contrôleur d'OpenStack fonctionne de la manière suivante (Fig. 5.7). Lorsqu'une requête est adressée à la partie contrôleur d'OpenStack :

- Elle transite par le réseau d'administration qui relie les contrôleurs aux hyperviseurs de la partie compute.
- Chaque contrôleur possède une adresse IP propre et une adresse IP virtuelle commune maintenue par le programme keepalived [15] en utilisant le protocole Virtual Router Redundancy Protocol (VRRP). La requête est adressée à cette adresse IP virtuelle commune.
- Sur chaque contrôleur on trouve un proxy HAProxy [14] qui écoute les requêtes entrantes et redirige celles-ci sur les adresses IP propres aux contrôleurs pour effectuer une répartition de charge.
- Finalement, la requête arrive à l'API du module OpenStack concerné et est traitée par celui-ci. Ce processus peut alors effectuer plusieurs actions :
  - Il peut écrire des données dans la base de données répartie sur les contrôleurs.
  - Il peut envoyer un message dans le logiciel de gestion de files de messages (ici RabbitMQ).
  - Il peut faire une requête à un autre module par l'adresse IP virtuelle commune qui suivra à nouveau le même principe de routage.

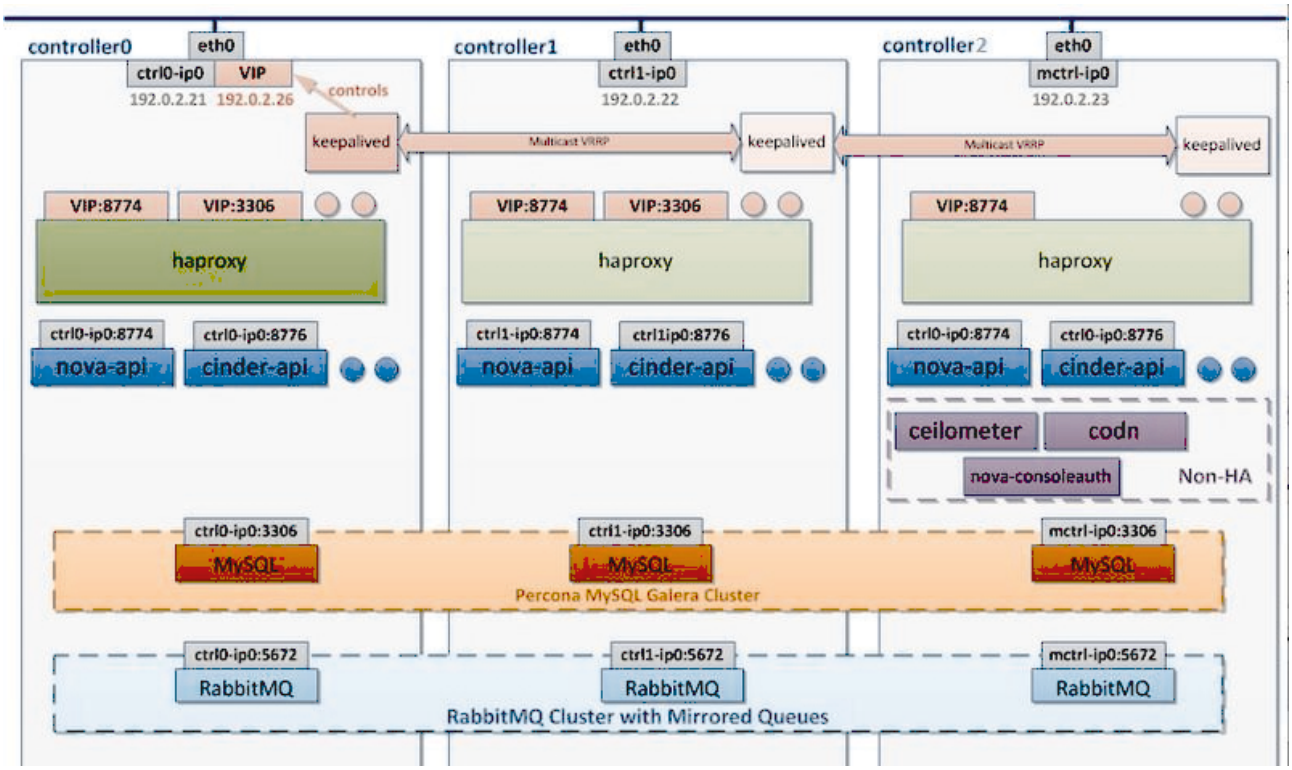

Fig 5.7 – Architecture haute disponibilité des contrôleurs OpenStack

L'architecture de la partie contrôleur d'OpenStack est donc redondée de manière logicielle. Mis à part quelques cas d'exceptions qui sont en cours de correction par la communauté, il s'agit à proprement parler d'une application répartie.

La partie contrôleur respecte bien la philosophie du projet car si un contrôleur venant à faillir il y aurait deux choix :

- Passer un certains nombres d'heures à comprendre quel est le problème pour le réparer.
- Supprimer le contrôleur et en recréer un autre.

La deuxième solution qui est la plus simple et rapide sera employée car, dans cette philosophie, aucun serveur n'est plus important qu'un autre. Il peut donc être entièrement reconstruit de manière très rapide et automatisée.

5.5.2 La partie "compute"

Chaque module dans OpenStack est composé de deux parties :

- La partie résidente sur les contrôleurs qui prend les décisions d'allocation des ressources, car au plus proche des bases de données.
- La partie qui réside sur les hyperviseurs qui exécute les ordres venant de la partie contrôleur et permet entre autre, de configurer le réseau, le stockage et administrer les machines virtuelles.

Il est important de noter que les machines virtuelles hébergeant les applications ne sont pas redondées. Comme déjà exprimé auparavant, il est de la responsabilité des applications d'être redondées de manière logicielle. Certaines applications sont déjà conçues de cette manière (les sites web) mais la plupart des applications historiques ne le sont pas.

Il y a donc deux possibilités dans ce cas :

- Réécrire l'application pour pouvoir l'héberger sur OpenStack : ceci est long, fastidieux et pas forcément possible s'il s'agit d'une application qui n'a pas été développée en interne.
- Ne pas l'héberger sur OpenStack et attendre que les travaux en cours dans la communauté concernant la redondance des machines virtuelles aboutissent.

Pour l'instant, la stratégie adoptée par PSA est la seconde. Comme les ressources humaines et financières sont limitées, il n'est actuellement pas possible de réécrire nos applications internes. Nous avons décidé d'héberger sur OpenStack ce qui est possible pour le moment et continuons à suivre les travaux de la communauté pour implémenter la haute disponibilité dans la partie compute. Nous travaillons aussi à adopter l'automatisation de la couche supérieure des logiciels communs (la partie PaaS). Cette couche PaaS sera utilisée dans les nouveaux projets internes afin de produire des applications réparties.

5.5.3 Le portail web

Horizon (Fig. 5.8) est le projet d'OpenStack servant de portail web. Ce portail est conçu dans le but d'être utilisable et compréhensible par les utilisateurs finaux. La terminologie employée est proche de celle présente dans les portails web des acteurs majeurs tels que Amazon ou Microsoft Azure.

Un utilisateur ou groupe d'utilisateurs se voit attribuer un "projet" dans le portail par les administrateurs. Les administrateurs définissent alors des quotas en termes de ressources pour chaque projet. Si toutes les ressources du projet venaient à être consommées, il faut s'adresser aux administrateurs pour se voir en attribuer davantage. Cependant, dans les limites des quotas, les utilisateurs sont indépendants dans la gestion de leurs ressources. Cette méthode de gestion a l'avantage d'éviter de mettre en place des cycles de validation complexes qui ralentissent la mise à disposition des ressources.

De plus, l'ensemble des communications du portail web avec les différents modules se fait en temps réel et via des requêtes HTTP aux API avec le format de représentation des données standardisé JSON. Le déploiement est donc quasi instantané.

Horizon permet également l'authentification et l'attribution des droits aux utilisateurs. Comme tous les autres modules d'OpenStack, il fait appel au module Keystone pour déterminer les droits des utilisateurs. Keystone permet de gérer un grand nombre de mécanismes d'authentification tels que LDAP, OAuth, SAML ...

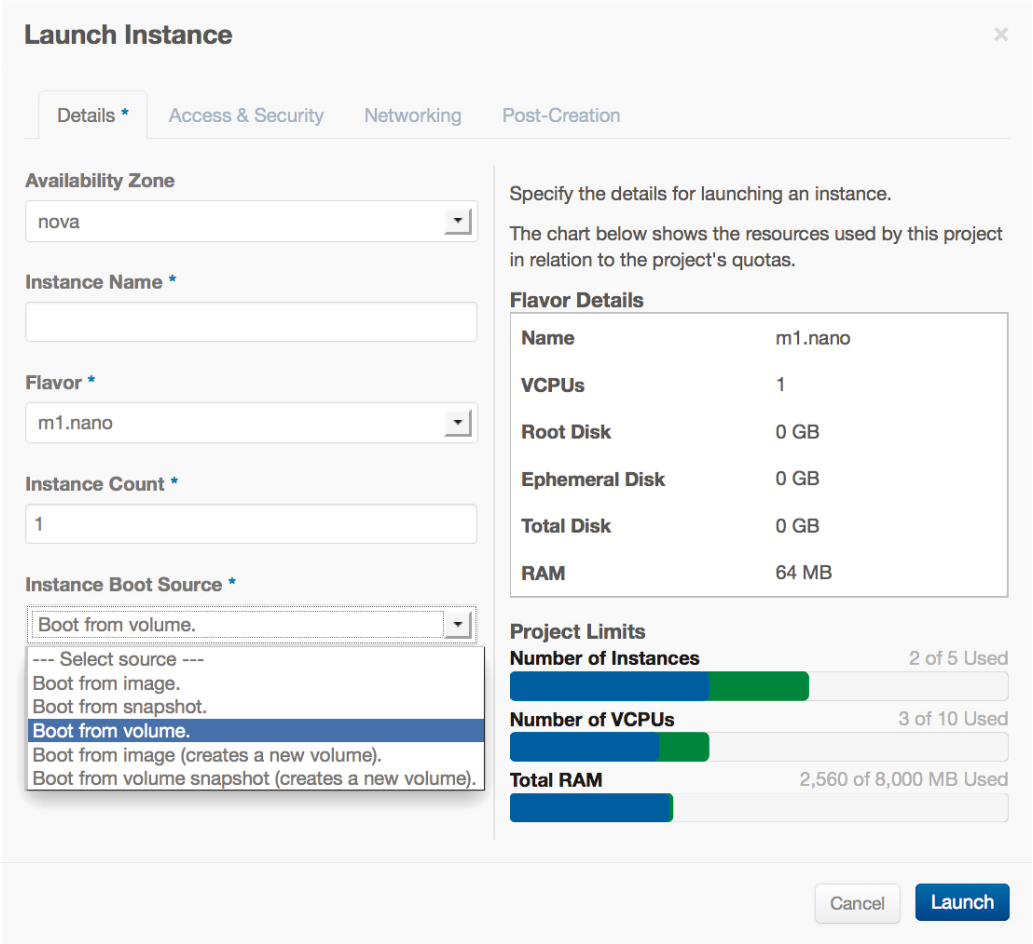

Fig 5.8 – Aperçu de l'interface d'administration Horizon

5.6 Bilan

La conception d'OpenStack fait que la majorité des problèmes que nous avons rencontrés dans le service d'infrastructure actuel chez PSA (FastPath + Portail COMUT) n'y sont pas présents.

Les points forts d'OpenStack par rapport à nos développements internes sont :

- Un unique langage de programmation est utilisé dans l'ensemble du projet : python. Ce langage est beaucoup plus flexible et évolué que le langage de Shell scripts actuellement utilisé dans FastPath.
- Les modules pour gérer les éléments matériels ou logiciels de l'infrastructure sont écrits par des spécialistes, voire par les concepteurs des éléments en question. Les administrateurs n'ont donc plus besoin d'investir autant de temps à développer des outils pour intégrer chaque nouvelle technologie.
- L'ensemble des modules d'OpenStack maintient des données en temps réel et il n'y a donc pas de problèmes de décalage entre ceux-ci.
- L'ensemble des communications inter et intra modules utilise une API et un format de données standardisé.
- Le portail web utilise un langage clair et familier auquel les développeurs utilisant des Clouds publics sont déjà habitués.
- Le mode de gestion des ressources par le portail web permet une plus grande autonomie des utilisateurs et une mise à disposition des ressources accélérée.

Malgré tous les atouts d'OpenStack il reste quand même des challenges à relever afin de réussir l'intégration de celui-ci au sein de l'infrastructure PSA.

Premièrement, il faut choisir une distribution d'OpenStack facilitant le déploiement et la maintenance de celui-ci. L'analyse des stratégies des fournisseurs à court, moyen et long termes est essentielle afin d'assurer la pérennité de la solution. Le choix final du constructeur devrait intervenir fin 2015.

Le deuxième challenge s'articule autour du type de service d'infrastructure. OpenStack laissant aux applications le soin d'assurer la disponibilité et la redondance, il va falloir définir quelles sont les applications susceptibles de migrer sur OpenStack. L'établissement de cette liste va être effectuée à partir de septembre 2015 par les équipes de l'infrastructure avec l'accord des équipes ayant la responsabilité des dites applications. Afin de promouvoir le développement d'applications basées sur ce nouveau paradigme, l'étude et la définition de la cible d'architecture PaaS est aussi en cours.

Cependant, comment va-t-on gérer deux services d'infrastructure différents en parallèle ? OpenStack étant une architecture où, au vue des interactions entre les composants, les limites des responsabilités entre les équipes existantes de l'infrastructure deviennent de plus en plus floues. Dans l'idéal, l'équipe devant gérer OpenStack devra être constituée d'un ensemble de personnes de spécialités différentes mais ayant des compétences transverses. Une des solutions est de rediriger le maximum de ressources sur OpenStack en ne faisant plus évoluer l'ancien service d'infrastructure FastPath. Ce point n'a pas encore été abordé en interne PSA.

De même, doit-on adapter notre portail COMUT actuel pour gérer OpenStack ou doit-on changer nos modes de validation des demandes de ressources ?

Cet ensemble d'interrogations et challenges sont plus liés à des problèmes stratégiques et organisationnels que techniques.

Nous allons à présent rentrer dans les détails techniques de mes travaux sur OpenStack. Je vais rapidement faire un rappel des bases du stockage puis expliquer le fonctionnement et exposer mes travaux sur le module gérant le stockage dans OpenStack : Cinder.

Chapitre

6

Le stockage dans OpenStack : Cinder

6.1 Présentation du stockage SAN

Les réseaux de stockage Storage Area Network (SAN) ont été introduits dans les années 2000 pour répondre aux besoins de croissance et de haute-disponibilité. On est alors passé d'un mode de stockage dédié ou chaque serveur avait ses propres disques à un mode de stockage mutualisé ou les disques sont rassemblés dans une baie et partagés entre plusieurs serveurs.

Un réseau de stockage SAN est ensemble de matériels (switchs, routeurs), redondants permettant l'interconnexion de serveurs et de moyens de stockage. L'ensemble des éléments communique par le protocole Fibre Channel sur un support fibre optique 2, 4, 8 ou 16 Gbps.

Les deux principaux fournisseurs de matériel du réseau SAN sont Brocade et Cisco. PSA utilise exclusivement des switchs et routeurs SAN du constructeur Brocade.

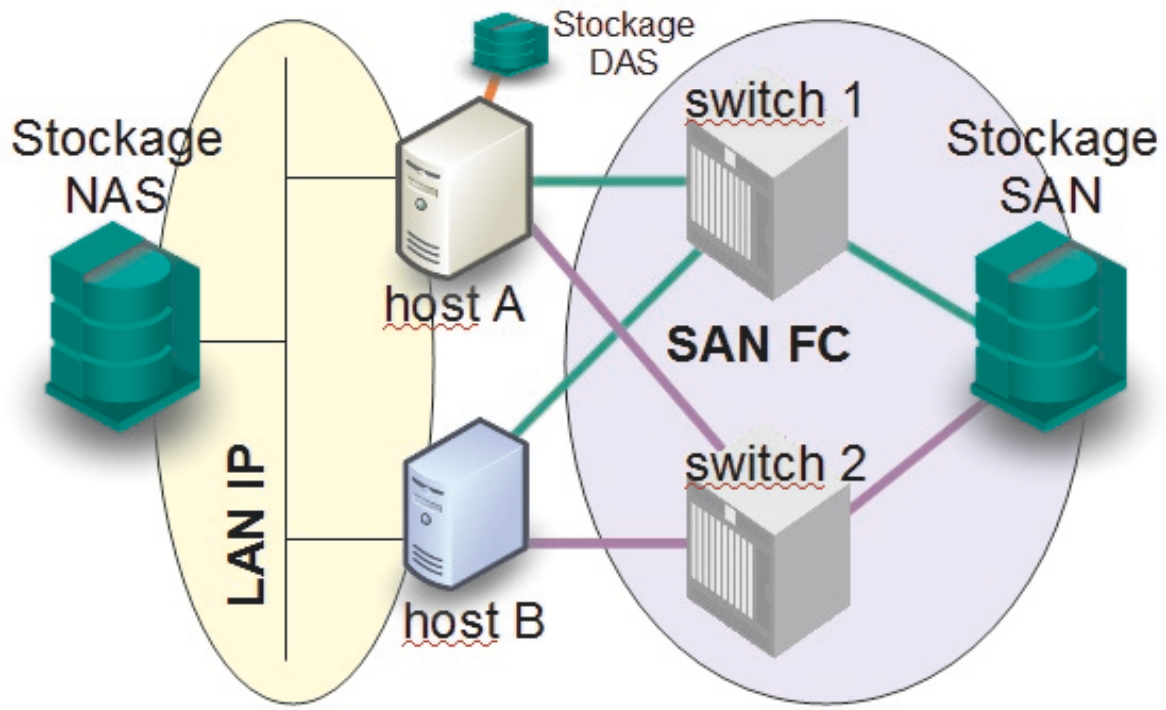

Fig 6.1 – Exemple de réseaux SAN, NAS et DAS

Le SAN est un réseau bien distinct des autres types de réseaux tels que :

- Le Direct Attached Storage (DAS) se caractérise pas l'accès à une baie de stockage directement reliée à un serveur. Les disques ne sont donc pas partagés avec d'autres serveurs. Comme dans environnement SAN, l'accès aux données s'effectue en mode "bloc" via le protocole SCSI.
- Le Network Attached Storage (NAS) se caractérise pas l'accès à une baie de stockage via un réseau LAN IP. L'accès aux données s'effectue cette fois en mode "fichier", grâce aux protocoles de partage CIFS (Windows) ou NFS (Unix).

L'ensemble des switchs, routeurs, serveurs et baies de stockage interconnectés dans un même réseau se nomme une "Fabric". Tous les périphériques connectés au SAN disposent d'attachements à celui-ci par l'intermédiaire de cartes réseaux spécifiques Host Bus Adapater (HBA). Ces cartes possèdent elles-mêmes un ou plusieurs ports ayant chacun un identifiant unique semblable aux adresses MAC du monde IP : le World Wide Name (WWN). Chez PSA, chaque serveur possède deux de ces cartes. Chacune de ces cartes est reliée à un réseau SAN indépendant. Des logiciels dits de "multipathing" permettent d'obtenir une haute disponibilité d'accès aux baies de stockage en utilisant ces deux cartes.

La configuration des switchs, appelée "zoning", permet de définir quel serveur a accès à quelle baie de stockage. Afin de rendre plus lisible la configuration du zoning en place, il est attribué à chaque WWN un nom ou "alias" reflétant le nom de l'équipement à qui il appartient. On définit alors des "zones" constituées d'au moins deux alias. Les zones sont donc les éléments de la configuration du zoning permettant de relier les équipements entre eux.

Les WWN des serveurs ont aussi besoin d'être définis dans la configuration des baies de stockage. Cela permet aux baies de stockage de savoir quel type de système d'exploitation va utiliser les disques, et donc de modifier son comportement en fonction des spécificités de ceux-ci.

6.2 Le stockage SAN chez PSA

Le métier du stockage chez PSA gère environ 10Po de données dont 6Po de SAN et 4Po de NAS. Parmi le stockage SAN on retrouve deux grandes catégories de baies de stockage :

- Les baies haut de gamme EMC VMAX représentant un tiers du stockage.
- Les baies milieu de gamme EMC VNX représentant le reste du stockage.

Les baies haut de gamme sont historiquement utilisées pour les applications très critiques ne tolérant pas de pannes. Une panne d'une de ces baies engendre directement un impact majeur sur les activités du groupe. Ces baies sont mises en relation directe avec les serveurs via le SAN.

Les baies milieu de gamme sont utilisées pour les applications peu critiques et étaient aussi mises en relation directe avec les serveurs. Cependant en 2007, pour faciliter la gestion de ce type de stockage, nous avons introduit l'équipement du fournisseur IBM SAN Volume Controller (SVC). SVC est un équipement SAN de virtualisation du stockage qui peut être comparé à un proxy dans le mode du réseau IP. Les disques des baies EMC VNX sont fournis à SVC qui les rassemble pour en faire de gros espaces de stockage mutualisés appelés des "mdiskgroups". SVC va permettre de découper des disques virtuels (aussi appelés "vdisks") dans ces mdiskgroups et les présenter aux serveurs (Fig. 6.2).

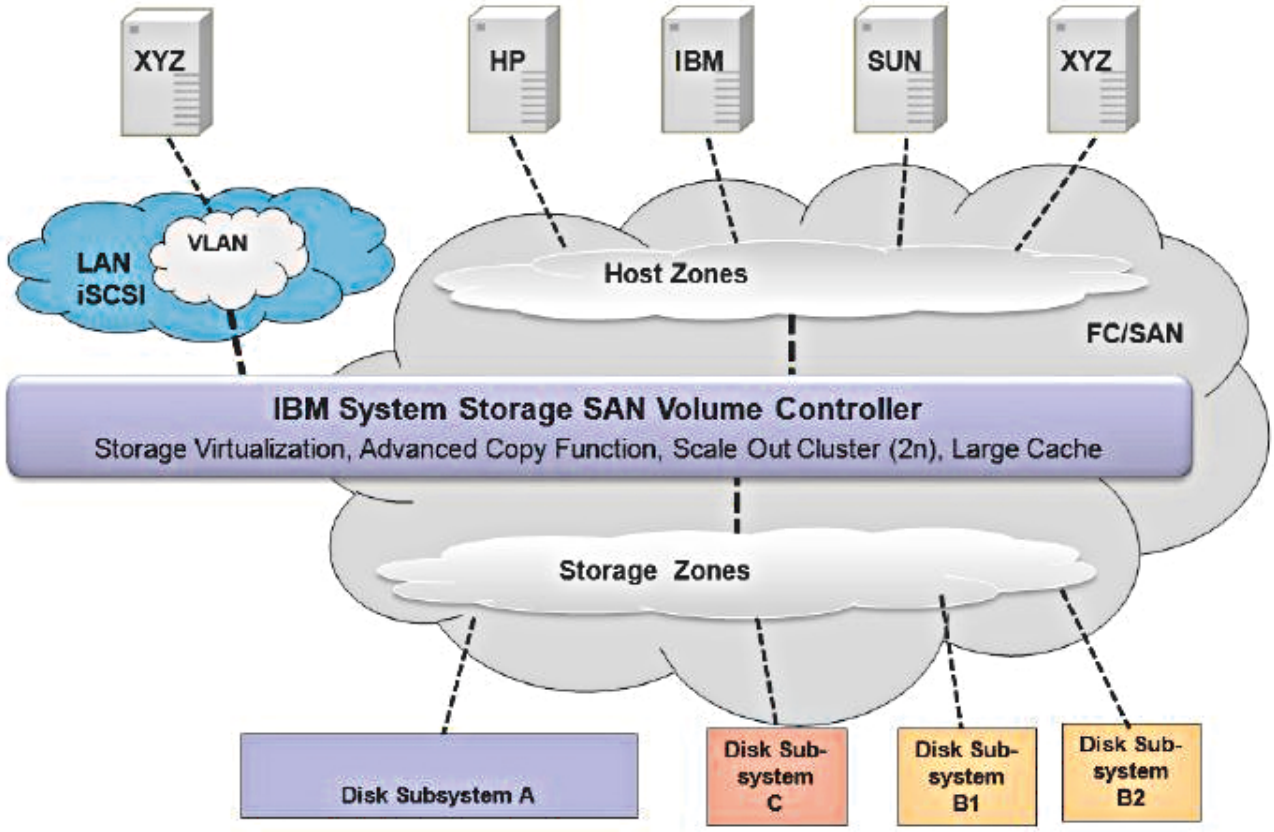

Fig 6.2 – Architecture d'IBM SVC

Les avantages de ce système de virtualisation du stockage sont nombreux :

- Avant l'introduction de SVC, lors du nécessaire renouvellement des baies de milieu de gamme, il fallait faire beaucoup d'opérations manuelles pour migrer les serveurs des anciennes baies aux nouvelles. Chaque disque d'une ancienne baie devait être recopié par le serveur sur un nouveau disque de la nouvelle baie. Cela impliquait donc se synchroniser avec les équipes des systèmes d'exploitation pour faire les manipulations nécessaires. Depuis l'introduction de SVC, comme celui-ci agit comme un proxy, il suffit de déplacer les vdisks de l'ancienne baie à la nouvelle. Les serveurs ne sont donc plus impactés car ils n'ont pas la visibilité des baies en arrière-plan.
- Les baies en arrière-plan de SVC sont interchangeables à souhait car n'étant plus directement visibles des serveurs La dépendance à unique fournisseur de baies étant largement réduite grâce à cette fonctionnalité.
- Les fonctionnalités avancées d'une baie SAN sont présentes dans SVC, réduisant donc la complexité et par la même occasion le coût des baies en arrière-plan.
- Toutes les opérations réalisées auparavant sur les baies sont à présent directement réalisées sur SVC. Le développement d'une automatisation à chaque changement de constructeur de baies n'est donc plus nécessaire.

La charge de développement nécessaire à l'automatisation des baies haut de gamme, de SVC et du réseau SAN est quasiment équivalente. Nous maintenons actuellement environ 20.000 lignes de code Perl pour chacune de ces technologies. Cela demande donc de constantes évolutions en fonction des évolutions et nouvelles fonctionnalités introduites pas ces baies. De plus l'ensemble des scripts n'est maintenu que par deux personnes chez PSA : un collègue et moi.

Les opérations à réaliser sur cet ensemble d'équipements sont variées. Prenons en exemple la simple attribution d'une Logical Unit Number (LUN) à une machine virtuelle. Une LUN est un espace logique de stockage qui peut être distribué sur plusieurs disques physiques de la baie pour redondance. Les étapes nécessaires sont les suivantes :

- Premièrement, il faut modifier la configuration des switchs SAN pour permettre au serveur de communiquer avec la baie. C'est l'étape de zoning impliquant de créer des alias et des zones.
- Puis, viennent les modifications sur la baie :
  - En premier lieu, il est nécessaire de renseigner le serveur dans la baie afin de permettre les adaptations protocolaires nécessaires.
  - Puis, il faut créer une LUN dans la baie.
  - Enfin, on doit relier la LUN au serveur dans la configuration de la baie. On appelle cette étape le "masking".
- Finalement, il faut découvrir la LUN sur le serveur.

Ceci n'est qu'une opération parmi tant d'autres. On peut citer comme opérations élémentaires nécessaires à un service d'infrastructure :

- Sur les switchs SAN :
  - Création et suppression d'alias.
  - Création et suppression de zones.
- Sur les baies :
  - Création et suppression des définitions des serveurs.
  - Création, agrandissement et suppression des LUN.
  - Attribution et retrait des LUN aux serveurs.
- Sur les serveurs :
  - Détection et restitution des LUN.

Comme on peut le constater le nombre d'étapes pour une simple opération requiert des interventions sur des équipements différents. Pour complexifier encore les choses, les scripts gérant les switchs SAN et les baies de stockage sont développés par une équipe, ceux des systèmes d'exploitation par d'autres. Les équipes des systèmes d'exploitation, ne collaborant quasiment pas entre elles, demandent au service en charge du stockage les mêmes informations sous des formats différents.

Un module existe dans OpenStack afin de réaliser l'ensemble de ces étapes de bout en bout de manière cohérente : Cinder. Je vais donc à présent présenter ce module et expliquer comment il résout ces problèmes.

6.3 Présentation de Cinder

A l'origine d'OpenStack, le gestionnaire d'hyperviseurs Nova gérait aussi le réseau et le stockage. Cependant, OpenStack prenant de l'ampleur, il fût décidé que Nova ne gérerait plus que les hyperviseurs. Le projet Neutron fût donc créé pour gérer le réseau et Cinder fût le projet gérant le stockage en mode bloc. Cinder est un projet communautaire ayant le soutien de la majorité des fournisseurs de stockage. On ne compte pas moins de 65 équipements de stockage différents supportés [21].

Cinder gère uniquement le stockage en mode bloc car d'autres projets d'OpenStack existent pour les autres modes de stockage :

- Le projet Swift gère le stockage en mode objet.
- Le projet Manila gère le stockage de type NAS.

Cinder a donc la responsabilité de fournir des modules (ou "drivers") pour plusieurs types équipements :

- Des drivers pour les switchs SAN.
- Des drivers pour les baies de stockage.
- Des drivers pour découvrir le stockage sur les serveurs : il s'agit du sous projet os-brick [27].

Le rôle de Cinder est donc de présenter des LUN, aussi appelées "volumes" dans la terminologie OpenStack, aux machines virtuelles. Afin d'y parvenir il peut utiliser un grand nombre de protocoles différents via le sous projet os-brick. Celui-ci supporte entre autres les protocoles suivants :

- ATA over Ethernet (AOE) utilisant le réseau Ethernet.
- Distributed Replicated Block Device (DRBD) utilisant le réseau Ethernet.
- Internet Small Computer System Interface (ISCSI) utilisant le réseau Ethernet.
- ISCSI Extensions for RDMA (ISER) utilisant le réseau Ethernet.
- Network File System (NFS) utilisant le réseau Ethernet.
- Fibre Channel utilisant le réseau SAN.

Historiquement, le protocole ISCSI est le plus utilisé dans Cinder car les premiers utilisateurs d'OpenStack étaient des hébergeurs web. Ils avaient déjà en place une infrastructure Ethernet éprouvée et efficace. Cela explique la prépondérance de l'utilisation de ISCSI. Le support des réseaux SAN, et donc du protocole Fibre Channel, n'étant arrivé que récemment avec l'émergence de l'utilisation d'OpenStack dans des entreprises industrielles.

6.4 Architecture de Cinder

Afin de standardiser la gestion du grand nombre de protocoles et de fournisseurs différents, Cinder est implémenté sous la forme d'une couche d'abstraction. Cette couche d'abstraction comprend cinq modules principaux (Fig. 6.3) :

- cinder-client
- cinder-api
- cinder-scheduler
- cinder-volume
- cinder-backup

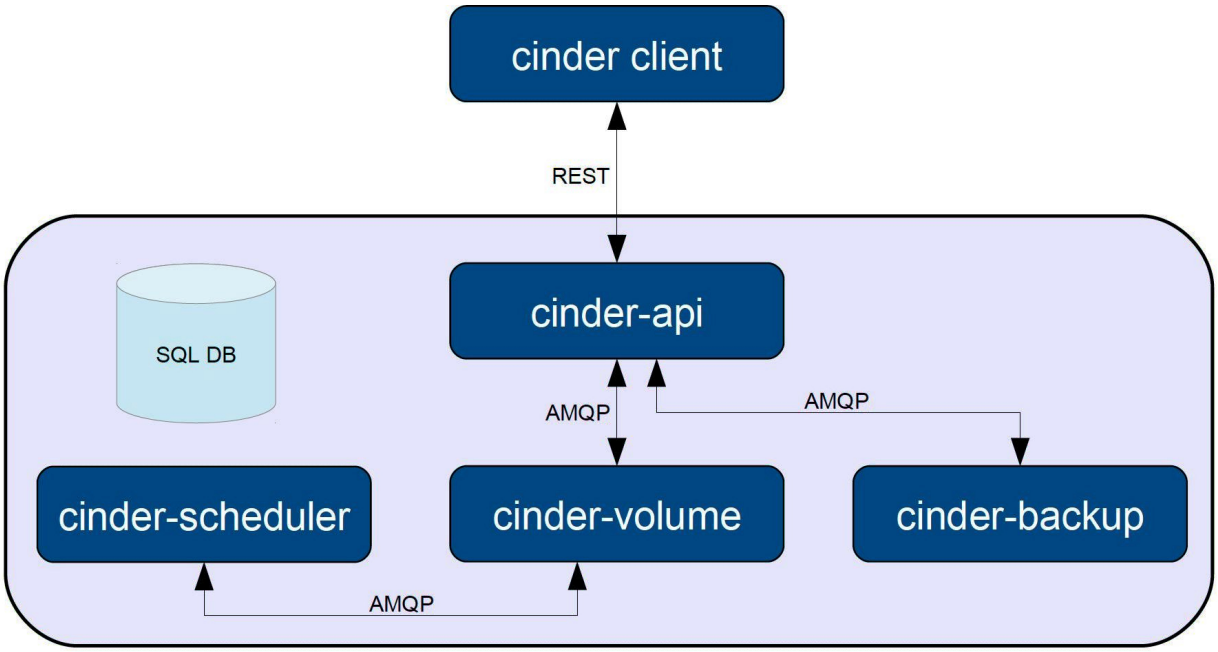

Fig 6.3 – Architecture de Cinder

L'ensemble de ces modules est écrit dans le même langage de programmation (python) et est configuré par un unique fichier texte "cinder.conf".

Ce fichier de configuration est constitué d'un ensemble de sections dont :

- Une section par module de Cinder.
- Une section pour configurer la base de données.
- Une section pour configurer le logiciel de file de messages.
- Une section pour chaque baie.
- Une section pour chaque Fabric SAN.

6.4.1 cinder-client

Le premier composant de Cinder "cinder-client" [20] est un module optionnel. Ce module permet de piloter Cinder en ligne de commande en envoyant des requêtes à l'API standardisée "cinder-api". Voici un exemple de la ligne de commande du client :

```
# cinder help create
usage: cinder create [--snapshot-id <snapshot-id>]
                     [--source-volid <source-volid>] [--image-id <image-id>]
                     [--display-name <display-name>]
                     [--display-description <display-description>]
                     [--volume-type <volume-type>]
                     [--availability-zone <availability-zone>]
                     [--metadata [<key=value> [<key=value> ...]]]
                     <size>

Creates a volume.

Positional arguments:
  <size>                Volume size, in GBs.

Optional arguments:
  --snapshot-id <snapshot-id>
                        Creates volume from snapshot ID. Default=None.
  --source-volid <source-volid>
                        Creates volume from volume ID. Default=None.
  --image-id <image-id>
                        Creates volume from image ID. Default=None.
  --display-name <display-name>
                        Volume name. Default=None.
  --display-description <display-description>
                        Volume description. Default=None.
  --volume-type <volume-type>
                        Volume type. Default=None.
  --availability-zone <availability-zone>
                        Availability zone for volume. Default=None.
  --metadata [<key=value> [<key=value> ...]]
                        Metadata key and value pairs. Default=None.
```

Les requêtes sont envoyées à travers le protocole HTTP au module cinder-api et les données des requêtes sont quant à elles normalisées au format JSON.

Le module cinder-client peut aussi s'utiliser sous forme d'une librairie python d'accès aux fonctions de Cinder :

```
>>> from cinderclient import client
>>> cinder = client.Client('1', $OS_USER_NAME, $OS_PASSWORD,
    $OS_TENANT_NAME, $OS_AUTH_URL)
>>> cinder.volumes.list()
[]
>>> myvol = cinder.volumes.create(display_name="test-vol", size=1)
>>> myvol.id
ce06d0a8-5c1b-4e2c-81d2-39eca6bbfb70
>>> cinder.volumes.list()
[<Volume: ce06d0a8-5c1b-4e2c-81d2-39eca6bbfb70>]
>>> myvol.delete
```

6.4.2 cinder-api

Le composant “cinder-api” fournit une API standardisée [19] permettant d’effectuer des opérations sur le stockage. Il constitue la couche d’abstraction de Cinder et reçoit des requêtes HTTP et des données associées au format JSON de sources diverses :

- Les requêtes peuvent provenir du portail web Horizon quand les utilisateurs créent des volumes.
- Les requêtes peuvent venir de cinder-client quand les administrateurs font des opérations manuelles.
- Finalement des requêtes peuvent provenir d’applications tierces.

Parmi les fonctions utilisables on va trouver un ensemble de fonctions :

- Création, extension, suppression et détails des volumes.
- Création, suppression et détails de snapshots ou backups.
- Création des volumes à partir d’images ou de snapshots.
- Attachement et détachement des volumes sur les hyperviseurs.
- Migration des volumes entre hyperviseurs ou entre baies.
- Transfert des volumes d’un projet à un autre.
- Gestion des types de volumes.
- Gestion des quotas.

Une fois la requête parvenue à cinder-api, celui-ci vérifie la syntaxe de la requête. Si celle-ci est correcte, il dépose les informations de la demande dans une file de messages qui est traitée par le module suivant “cinder-volume”.

6.4.3 cinder-volume

Le module “cinder-volume” gère les demandes en provenance de cinder-api. Il interroge de manière régulière la file de messages pour vérifier si des opérations sont nécessaires. Si une opération est demandée, il vérifie en collaboration avec le module cinder-scheduler qu'elle soit réalisable et effectue cette opération sur les équipements de stockage via les drivers associés. Ce module peut être configuré de deux manières :

- Soit une unique instance de ce module gère plusieurs baies.
- Soit plusieurs instances ont chacune la gestion d'une baie.

Via des drivers propre à chaque matériel, cinder-volume est capable de gérer à la fois la configuration des switchs SAN et la configuration des baies. Ces drivers doivent implémenter l'ensemble des fonctions définies par cinder-api. Cela permet d'avoir via Cinder un ensemble de fonctions ayant la même finalité sur l'ensemble des matériels.

Une fois l'opération effectuée, cinder-volume met les informations nécessaires ainsi que le résultat de l'opération dans la file de messages à destination de cinder-api.

6.4.4 cinder-scheduler

Cinder-scheduler est le module de Cinder dans lequel réside l'intelligence de Cinder. Il a deux fonctions principales :

- Collecter les informations des différents équipements à l'aide de cinder-volume et les stocker dans la base de données.
- Utiliser les données présentes ainsi que les restrictions en place afin de déterminer où et comment placer les demandes de ressources.

Afin de prendre les décisions concernant le placement des ressources, cinder-scheduler implémente deux concepts :

- Un système de filtres pour choisir quels équipements sont capables de répondre aux besoins des demandes.
- Un système de tris afin de classer les équipements éligibles selon des critères définis.

Ces deux systèmes sont configurables et utilisent un système de sous-modules afin d'effectuer leurs fonctions respectives.

Le système de filtres permet de choisir la manière de déterminer où un volume va être créé. On peut choisir actuellement plusieurs filtres :

- AvailabilityZoneFilter : Ce filtre permet de créer un volume dans une AvailabilityZone définie. Dans OpenStack une AvailabilityZone est un groupement logique arbitraire d'équipements.
- CapacityFilter : Ce filtre permet de déterminer si un équipement a encore la capacité physique d'héberger la volumétrie demandée.
- CapabilitiesFilter : Ce filtre permet de spécifier, lors de la création d'un volume, des fonctionnalités nécessaires pour le volume. Si l'équipement ne gère pas ces fonctionnalités il sera exclu.
- AffinityFilter : Ce filtre permet de créer un volume dans le même équipement qu'un autre volume pour avoir une cohérence de placement. La fonction inverse est aussi disponible : créer un volume dans un autre équipement que le volume sélectionné, ceci afin d'améliorer la tolérance aux pannes.
- InstanceLocalityFilter : Dans le même principe que le filtre AffinityFilter, ce filtre permet de choisir un machine virtuelle, déterminer sur quel équipement elle est hébergée et créer un volume au même endroit.
- DriverFilter : Ce filtre permet de définir une fonction personnalisée pour le filtrage. Le driver de l'équipement ou l'administrateur peut utiliser ce filtre via le fichier de configuration.

De la même manière le système de tris a plusieurs options fournies :

- CapacityWeigher : Ce tri retournera une liste triée d'équipements suivant l'espace non réservé de stockage.
- AllocatedCapacityWeigher : Ce tri retournera une liste triée d'équipements suivant l'espace réellement libre. Cela prend en compte le fait que même si des volumes sont créés sur un équipement, ceux-ci peuvent être relativement vide, libérant ainsi de l'espace de stockage. La surallocation est ainsi possible sur un équipement. Elle consiste à allouer plus d'espace logique que d'espace physique disponible sur l'équipement en espérant qu'il ne soit pas consommé.
- ChanceWeigher : Ce tri permet de choisir de manière aléatoire une liste d'équipements.
- GoodnessWeigher : Ce tri permet de remonter une liste triée d'équipements en fonction du nombre d'options supportées parmi un liste de fonctionnalités optionnelles fournies.
- VolumeNumberWeigher : Ce tri permet de remonter une liste triée d'équipements selon le nombre de volumes hébergés sur chacun d'eux.

Lors de la création d'un volume cinder-scheduler suit donc le circuit de décision suivant :

- cinder-scheduler part de la liste de tous les équipements disponibles dans la configuration.
- Il vérifie que les quotas du projet ne sont pas dépassés.
- Puis, il filtre les équipements à l'aide d'un ou plusieurs filtres consécutifs selon la configuration définie.
- Il remonte une liste d'équipements potentiellement utilisables. S'il n'en existe pas la création du volume échoue.
- Cette liste est ensuite triée par le type de tri choisi.
- Finalement, une liste triée d'équipements potentiels est fournie.

Ce système étant hautement configurable, si PSA choisit d'implémenter des règles métiers de placement des ressources et que celles-ci sont actuellement non présentes dans Cinder, cinder-scheduler est l'endroit où faire l'implémentation. En effet, avec seulement quelques lignes de configuration nous pouvons implémenter un filtre ou tri personnalisé en utilisant DriverFilter et GoodnessWeigher. Ceci à condition que les informations servant à ces fonctions soient déjà récoltées et stockées dans la base de données.

6.4.5 cinder-backup

Dans cinder-backup une sauvegarde est une archive des données. A la différence d'un snapshot qui recopie l'ensemble du volume sous forme brute, une sauvegarde copie l'ensemble des fichiers contenus dans un volume. Ces fichiers sont alors rassemblés dans une archive, puis l'archive est sauvegardée dans un système de stockage objet.

Par défaut cinder-backup va donc sauvegarder les données des volumes via le projet Swift qui gère le stockage objet d'OpenStack.

Les sauvegardes complètes et incrémentales sont supportées, cependant il faut que les volumes ne soient pas attachés à une machine pour effectuer la sauvegarde. Cela est actuellement un inconvénient majeur de cette méthode de sauvegarde.

6.4.6 Exemple de la création d'un volume

Résumons les interactions entre les modules en prenant le cas de la création d'un volume (Fig. 6.4).

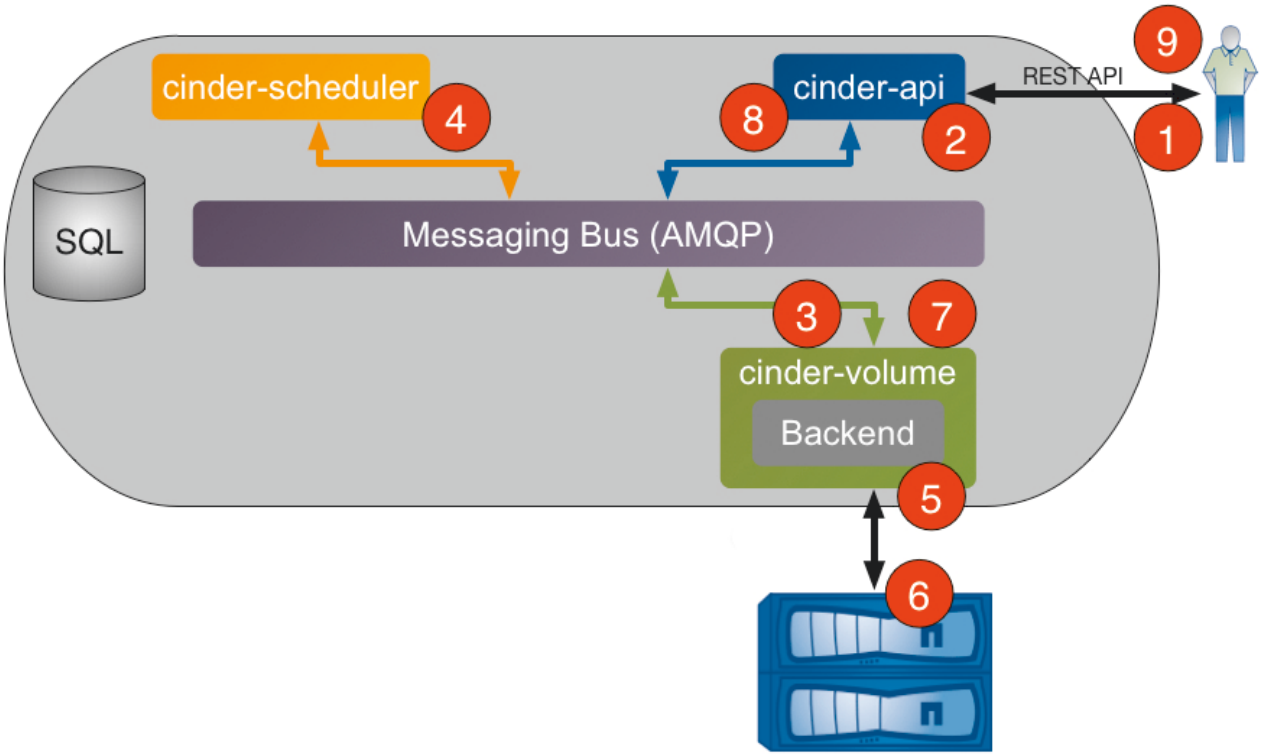

Fig 6.4 – Cinder : requête de création d'un volume

1. Le client envoie la requête de création de volume à travers l'API. Le client peut être la ligne de commande, le portail web Horizon ou un autre processus externe.
2. cinder-api valide la requête puis la dépose dans une file de messages pour être traitée.
3. cinder-volume récupère le message dans la file, vérifie que rien ne manque et dépose la requête dans la file de message.
4. cinder-scheduler récupère le message dans la file, filtre et tri les équipements à l'aide des informations présentent dans la base de données et finalement dépose une liste d'équipements pouvant héberger le volume dans la file de messages.
5. cinder-volume récupère la liste dans la file, puis essaye de créer le volume sur le premier équipement de la liste. Si celui-ci n'est pas disponible, il va tenter de créer le volume sur le deuxième équipement de la liste, et ainsi de suite.
6. Le driver de cinder-volume crée le volume. En fonction du protocole cela peut impliquer aussi des interactions avec des switchs SAN.
7. cinder-volume récupère les informations du volume crée, met à jour la base de données et dépose le tout dans la file de messages.
8. cinder-api lit les informations dans la file de messages, mets en forme les informations et répond au client.
9. Le client reçoit les informations concernant sa demande.

6.5 Application au stockage chez PSA

6.5.1 Les tests effectués

Les tests que j'ai effectués ont commencé début 2014. Cinder a été testé sur plusieurs distributions d'OpenStack dont SUSE Cloud et HP Helion. Ces tests ont été réalisés en collaboration avec M. DURUPT et M. DUCHENOY.

Les tests se sont concentrés sur deux types de baies que nous possédons chez PSA. Les baies de virtualisation de stockage IBM SVC et les baies NETAPP (Table 6.1).

Protocole / Driver	ISCSI	FC + Zoning	NFS
IBM SVC	OK	OK	N/A
NETAPP	OK	N/A	OK

Table 6.1 – Cinder : Matériels testés

Le fonctionnement de IBM SVC a été validé sur les protocoles Fibre Channel (réseau SAN) et le protocole ISCSI (réseau LAN). PSA utilisant exclusivement SVC sur le réseau SAN, les tests se sont concentrés sur SVC et ont permis de tester la gestion automatique de la configuration des switchs SAN (le zoning).

Concernant les baies NETAPP, celles-ci n'étant pas connectées au réseau SAN, seul les protocoles ISCSI et NFS utilisant le réseau LAN IP ont été testés.

Le fonctionnement technique de Cinder pour les protocoles Fibre Channel et ISCSI est similaire. Dans les deux cas Cinder va d'abord créer un volume sur une baie, puis attacher le volume sur le contrôleur OpenStack afin de copier le système d'exploitation sur le volume. Une fois la copie effectuée, il va détacher le volume du contrôleur OpenStack et l'attacher sur l'hyperviseur où la machine virtuelle va être hébergée. Cette étape de copie du système d'exploitation n'est nécessaire que si l'on désire démarrer la machine virtuelle sur un volume d'une baie. Il est possible de faire l'inverse : démarrer une machine virtuelle en hébergeant son système d'exploitation sur les disques internes de l'hyperviseur et donc ne pas utiliser de baie de stockage. Si un volume est destiné à héberger uniquement des données, l'étape d'attachement au contrôleur OpenStack et de recopie du système d'exploitation n'est donc pas nécessaire.

En ce qui concerne l'utilisation du protocole NFS sur une baie NETAPP, le fonctionnent diffère. Cinder ne crée pas de volume directement sur celle-ci. Il exporte en NFS un volume existant définit dans la configuration. Ce volume est exporté soit vers le contrôleur OpenStack, soit vers l'hyperviseur de destination. Un fichier (aussi appelé image disque) est alors créé sur ce volume exporté, puis ce fichier est présenté à la machine virtuelle comme un volume standard. Un volume d'une baie NETAPP peut donc contenir plusieurs images disques étant elles-mêmes des volumes pour les machines virtuelles.

La seule différence technique entre l'utilisation des protocoles Fibre Channel et ISCSI dans Cinder est la gestion du réseau d'accès aux volumes. Au début du projet Cinder les baies ainsi que les hyperviseurs qui utilisaient un de ces protocoles devaient être dans un même réseau. Cela impliquait pour les protocoles ISCSI et NFS, que les baies et hyperviseurs soient dans le même réseau IP, et pour le protocole Fibre Channel que le zoning entre toutes les baies et hyperviseurs soit fait à l'avance.

Cela n'est pas trop problématique pour le réseau IP, mais pour le réseau SAN cela est très fastidieux. Heureusement, avec la sortie de la version Icehouse d'OpenStack en avril 2014, Cinder s'est vu rajouter un composant permettant de gérer automatiquement le zoning des switchs : FCZoneManager ou FCZM (Fig. 6.5).

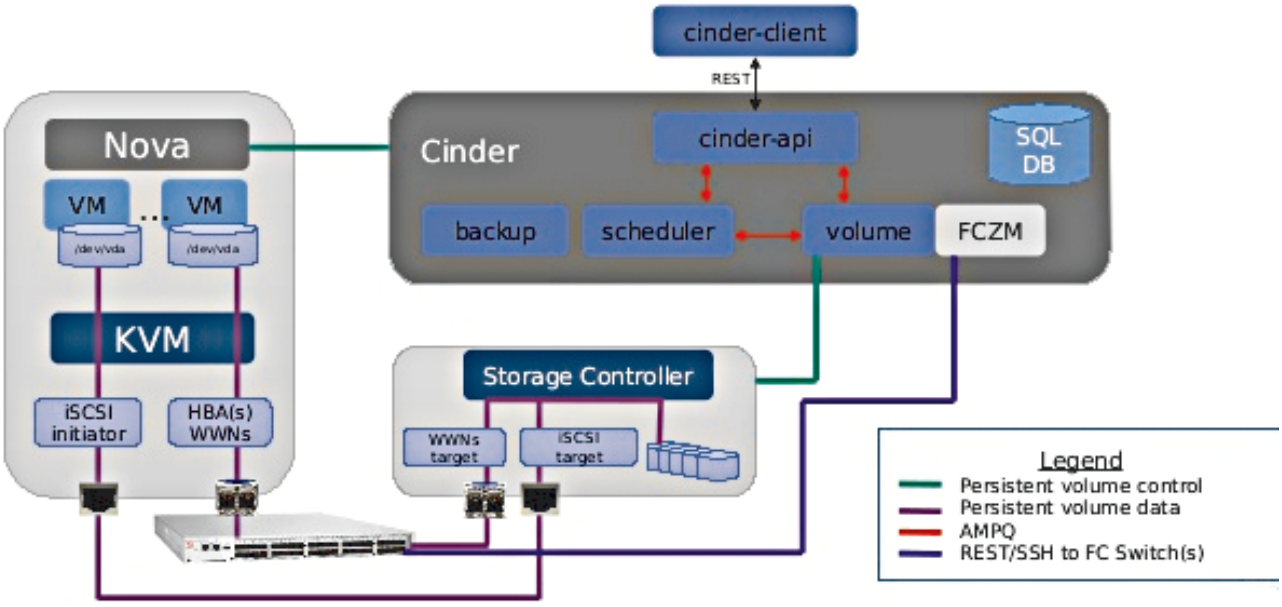

Fig 6.5 – Cinder : Schéma complet multi-protocoles

On retrouvera en annexe les configurations de Cinder nécessaires aux différents protocoles.

- SVC + Fibre Channel (Annexe A.1).
- SVC + ISCSI (Annexe A.2).
- NETAPP + ISCSI (Annexe A.3).
- NETAPP + NFS (Annexe A.4).

Les fonctionnalités basiques ont été testées et validées sur les deux types d'équipements (Table 6.2).

Matériel / Action	IBM SVC	NETAPP
Création d'un volume	OK	OK
Suppression d'un volume	OK	OK
Attachement d'un volume à une machine virtuelle	OK	OK
Détachement d'un volume d'une machine virtuelle	OK	OK

Table 6.2 – Cinder : Tests fonctionnels

6.5.2 Les problèmes rencontrés

Le principal problème rencontré lors des tests est un problème des fournisseurs de distributions d'OpenStack et pas un problème d'OpenStack en lui-même. Un fournisseur peut décider de ne pas inclure certains modules de pilotage de baies (drivers) dans sa distribution OpenStack. Un fournisseur, vendant aussi du matériel, va mettre en avant son propre matériel et donc tester et supporter en priorité les drivers de cinder-volume le concernant. Cela parait parfaitement justifiable d'un point de vue du fournisseur, cependant retirer des drivers sous prétexte qu'ils ne sont pas "supportés" d'un point de vue contractuel est handicapant pour effectuer des tests sur d'autres matériels.

Le deuxième problème est aussi relié aux fournisseurs de distributions et plus particulièrement à la méthode de déploiement des contrôleurs OpenStack. Comme la méthode de déploiement des contrôleurs n'est pour le moment pas standardisée par la communauté, chaque fournisseur a implémenté sa propre manière de déployer un contrôleur. Le fichier de configuration de Cinder (cinder.conf) est dans la majorité des cas automatiquement généré et mis en place sur le contrôleur. Cette automatisation du fichier de configuration ne permet pas forcément de changer tous les paramètres du fichier de configuration. Une édition manuelle de celui-ci est donc nécessaire. Le problème étant que ces modifications seront supprimées par le système de déploiement. Ce système vérifie de manière régulière que les fichiers de configuration d'OpenStack soient identiques à sa référence et les remplace si ceux-ci ont étés modifiés. Il est possible de désactiver cette synchronisation des fichiers de configuration, mais dans ce cas tous les fichiers de configuration d'OpenStack ne seront plus synchronisés, y compris ceux ne faisant pas partie de Cinder.

Lors des tests effectués sur IBM SVC deux problèmes ont étés détectés lors de l'utilisation du protocole Fibre Channel. J'ai donc personnellement créé deux entrées dans le système de traçage des problèmes du projet Cinder : Launchpad.

- Storwise SVC : unexpected removing of FC zoning.
  Bug : `https://bugs.launchpad.net/cinder/+bug/1468913`
  Code Review : `https://review.openstack.org/#/c/196966`
- Change cinder configuration storwize_svc_npiv_compatibility_mode default value to True.
  Bug : `https://bugs.launchpad.net/cinder/+bug/1471749`
  Code Review : `https://review.openstack.org/#/c/198692`

Le premier problème supprimait les zones entre l'hyperviseur et la baie SVC lorsque l'on supprimait un volume de l'hyperviseur. Lorsque ce volume était le dernier visible de l'hyperviseur cela n'était pas un problème. Cependant s'il restait encore d'autres volumes de SVC attachés à l'hyperviseur, la suppression était aussi exécutée, rendant ainsi les volumes encore présents sur l'hyperviseur inaccessibles. Le module gérant SVC ne vérifiait donc pas la présence de volumes attachés sur l'hyperviseur et supprimait dans tous les cas les zones. Le correctif implémente cette vérification et évite la suppression de zones si des volumes de SVC sont encore visibles sur l'hyperviseur.

Le deuxième problème rendait impossible la création de zones entre un hyperviseur et la baie SVC sans que le paramètre “storwize_svc_npiv_compatibility_mode” ne soit présent et activé. Ce paramètre était déclaré comme optionnel dans la documentation mais empêchait cependant les opérations de création de zones sur les switchs SAN. Ce paramètre était un reliquat d'ancien code dans le module SVC et avait été omis d'être supprimé. Le correctif active par défaut ce paramètre dans un premier temps. Si aucune régression n'est identifiée, ce paramètre sera simplement supprimé dans la prochaine version d'OpenStack.

J'ai donc contacté les développeurs du module SVC, collaboré avec eux afin d'expliquer les deux problèmes, proposer des correctifs à apporter au code et valider les correctifs sur notre plateforme de test. Les modifications ont été acceptées et sont à présent implémentées dans Cinder.

6.5.3 Les travaux à venir

Les fonctionnalités basiques étant validées, il restera les fonctionnalités avancées de Cinder à tester.

Un des premiers tests à réaliser concerne la possibilité de migration d'un volume. Un volume pouvant être migré par Cinder d'une baie à une autre ou d'un hyperviseur à un autre. Cette fonctionnalité est utile lorsqu'un désengagement ou une opération de maintenance est requise sur un hyperviseur ou une baie. Cela permet de déplacer tous les ressources présentes sur l'équipement afin de réaliser les opérations nécessaires.

Il me reste aussi la validation du fonctionnement de Cinder sur nos baies haut de gamme VMAX du fournisseur EMC. La même série de tests fonctionnels sera donc effectuée sur ces baies. Lors de la dernière conférence à Paris, j'ai eu l'occasion de prendre contact avec la personne ayant la responsabilité des drivers des équipements EMC. Cette personne sera un bon point d'entrée au cas où je rencontrerais des anomalies lors de mes tests.

Une simplification et normalisation des règles de placement des volumes va aussi être nécessaire. Les règles vont devoir être définies puis mises en place dans Cinder à l'aide de filtres et tris personnalisés.

Finalement, le type de baie, ainsi que le protocole à utiliser pour la plateforme de production de Cinder va devoir être choisi. Cette décision sera basée sur les avantages, les inconvénients ainsi que les coûts de chaque solution.

6.6 Bilan

Cinder est un projet très dynamique dans l'écosystème OpenStack. A chaque sortie d'une nouvelle version d'OpenStack le support des équipements de stockage progresse en termes de fonctionnalité et de diversité. L'implication des fournisseurs de matériels de stockage est grandissante et aujourd'hui ils participent activement à l'évolution de ce projet. De plus, la communauté des développeurs est réactive et facile d'approche, ce qui permet au projet de s'enrichir de nouveaux cas d'usages.

L'architecture très modulaire de Cinder a permis aux différents constructeurs de fournir des modules pour leurs matériels de manière aisée. Le gain de temps pour les administrateurs du domaine de stockage est apparent car ils n'ont plus à écrire eux-mêmes une automatisation à chaque renouvellement de matériel. En effet, le temps d'acquisition des compétences et de développement d'automatisations de nouvelles technologies de stockage peut rapidement devenir un facteur limitant. Si les développements internes sont longs et coûteux, le choix d'un matériel en rupture technologique peut être remis en cause.

Grâce à Cinder, l'introduction d'une nouvelle technologie se résume à une série de tests et validations fonctionnelles. La configuration simplifiée et centralisée de celui-ci évite de nombreux développements internes. L'envergure du projet n'aurait pas été possible sans une standardisation omniprésente des API, formats des données et fonctionnalités fournies. Ces standardisations permettent une intégration simplifiée avec des logiciels externes à OpenStack. Ses logiciels peuvent utiliser Cinder pour fournir des services de stockage car les spécifications de Cinder sont ouvertes au public. Dans le cas de notre service d'infrastructure interne, ceci n'est clairement pas possible car peu de choses sont standardisées et les spécifications, quand elles existent, ne sont pas publiées à l'extérieur.

Bien que ceux-ci ne soient pas totalement terminés, l'ensemble des tests que j'ai réalisés sur Cinder jusqu'à présent confirme que ce projet est mature et prêt à être utilisé à grande envergure sur des environnements de production. Les deux types de baies ainsi que les switchs SAN présents au sein de PSA sont supportés par Cinder. Les interactions de Cinder avec ceux-ci sont comprises et assez proche de ce que réalise notre automatisation actuelle dans FastPath. Il faudra cependant simplifier la multitude de règles internes à notre service d'infrastructure afin d'éviter de s'éloigner des bonnes pratiques.

En résumé Cinder, est un projet essentiel de notre point de vue. Il va permettre d'optimiser l'utilisation de nos ressources existante, réduire nos coûts de développement et accélérer la mise à disposition du stockage pour les utilisateurs.

Chapitre 7

La surveillance de l'infrastructure

7.1 Les concepts

La surveillance des différents éléments de l'infrastructure est cruciale. Plus les infrastructures deviennent complexes, plus il devient difficile de traiter les problèmes individuellement de chaque élément la constituant. Une surveillance centralisée devient donc nécessaire afin de :

- Détecter les pannes physiques.
- Détecter les défaillances logicielles.
- Surveiller les accès aux éléments.
- Remonter des statistiques sur les éléments surveillés.

L'agrégation de ses données de manière centralisée va permettre entre autre de :

- Détecter les problèmes récurrents ou cycliques.
- Classifier les problèmes par importance.
- Identifier les matériels ou logiciels sortant de la norme.
- Corréler les événements entre eux.
- Être plus efficace dans l'analyse des causes racines.

De manière schématique une surveillance est une suite d'événements distincts les uns des autres. Un événement étant constitué lui-même de deux éléments :

- Un horodatage précisant le moment ou l'événement est survenu.
- Un message indiquant le sujet de l'événement.

La problématique principale d'une surveillance centralisée tourne autour des formats des événements ainsi que des méthodes de transport de ceux-ci. En fonction des éléments de l'infrastructure les événements peuvent avoir une multitude de formats différents. Prenons le cas de l'horodatage d'un événement : celui-ci doit être précis et normalisé afin d'être exploitable, or ce n'est pas toujours le cas. Voyons le cas de différents formats non normalisés et analysons leurs problèmes.

- `19:05:32`
  Nous avons l'heure mais quelle est la date ?
- `15/05/15 19:05:32`
  C'est déjà mieux car nous avons une date, mais dans de quel fuseau horaire parlons-nous ?
- `15/05/15 19:05:32 UTC`
  Enfin l'horodatage est assez précis.

Comme on peut le remarquer avoir un horodatage précis demande une certaine rigueur. Cela sans compter que même parmi les horodatages précis il existe plusieurs normes : ISO 8601, Unix Time, ... On supposera ici que les éléments de l'infrastructure sont correctement synchronisés et n'ont pas de décalage de temps.

Le message de l'événement doit lui aussi être précis en précisant entre autre :

- Le processus émettant l'événement.
- Le niveau de sévérité de l'événement.
- Le sujet de l'événement.

Nous allons voir à présent quelques protocoles permettant de fournir un événement complet.

7.2 Les outils à disposition

Afin de remonter des événements de manière précise, il existe des protocoles standardisés d'échange.

7.2.1 Le protocole SNMP

Le premier d'entre eux est le protocole Simple Network Management Protocol (SNMP). SNMP est un protocole réseau standardisé permettant de gérer des équipements via un réseau IP. Ce protocole est très utilisé dans le monde des réseaux IP et SAN afin de surveiller les équipements tel que les switchs, routeurs ou encore les baies de stockage.

Ce protocole a comme principal avantage de standardiser le message de l'événement. Les Management Information Base (MIB) ont le rôle de normaliser les messages. Les MIB sont des fichiers structurés classifiant de manière hiérarchique les messages en leur attribuant des identifiants uniques appelés "object identifier" (OID) (Fig. 7.1).

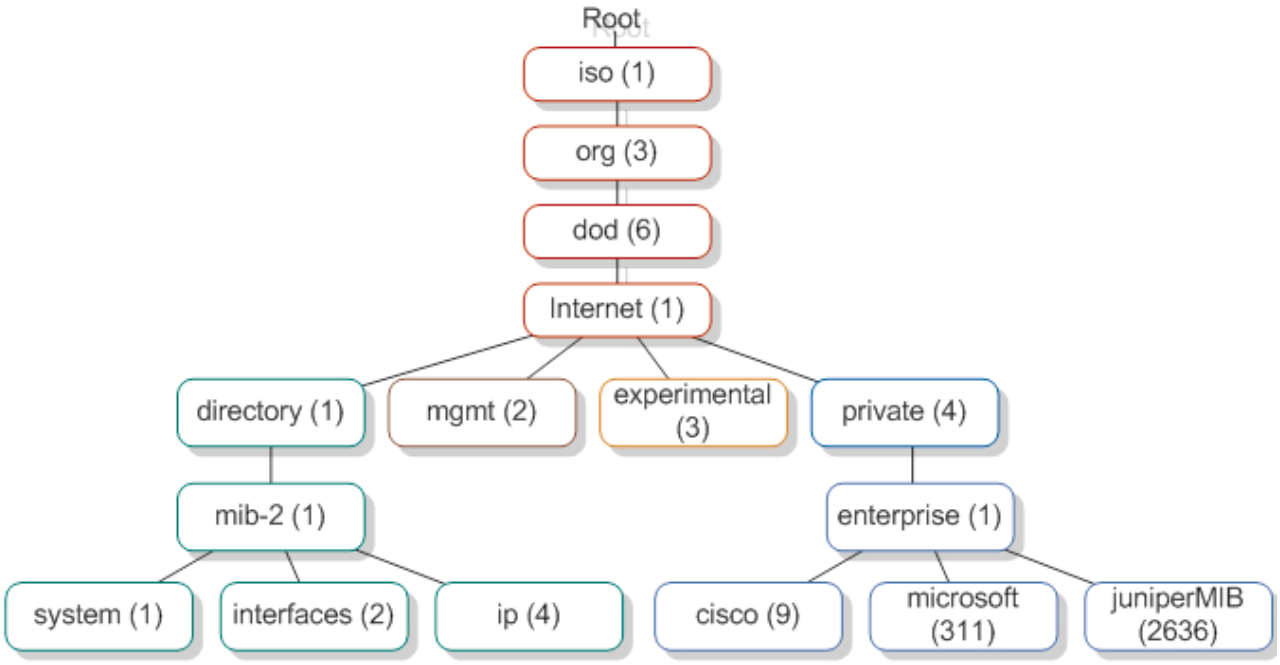

Fig 7.1 – Exemple d'une hiérarchie d'OID

Chaque type d'équipement peut définir une MIB personnalisée tant qu'elle ne rentre pas en conflit avec d'autres OID déjà réservés. Chaque OID est donc une suite de chiffres (ex : 1.3.6.1.4.1.9.8) pouvant être soit un conteneur de sous OID soit un OID final. Un OID final contient une valeur associée qui peut être de différents types tel qu'un entier, une chaîne de caractères, une IP, ...

Cependant SNMP a aussi quelques inconvénients :

- Il ne définit pas comment remonter l'horodatage de l'événement.
- Bien qu'il soit techniquement possible de remonter des informations de SNMP via le protocole Transmission Control Protocol (TCP), la plupart des équipements l'utilisant emploie le protocole User Datagram Protocol (UDP) pour des raisons historiques. Or, UDP est un protocole réseau déconnecté ne validant pas que le paquet réseau soit arrivé à destination. De ce fait, certains événements peuvent être non parvenus à destination sans que l'on puisse s'en rendre compte facilement.
- Une maintenance continue est requise pour maintenir à jour les MIB afin d'ajouter et de retirer des identifiants pour remonter des événements.
- Il s'agit principalement d'un protocole de transport et donc la gestion du stockage des événements est réduite.

De plus, tout événement d'un équipement n'a pas forcément d'OID associé. Sans compter que les applications utilisent très rarement SNMP pour stocker leurs événements. Elles ont tendance à écrire elles-mêmes leurs informations dans des fichiers journaux de logs.

7.2.2 Le protocole SYSLOG

SYSLOG est un protocole de stockage de message normalisé très utilisé dans le monde Unix [32]. Son but est de transporter via un réseau IP des messages générés par une application. Il est basé sur une architecture client-serveur avec la partie serveur qui peut être hébergée soit sur le même serveur que le client, soit sur un serveur tierce (Fig. 7.2).

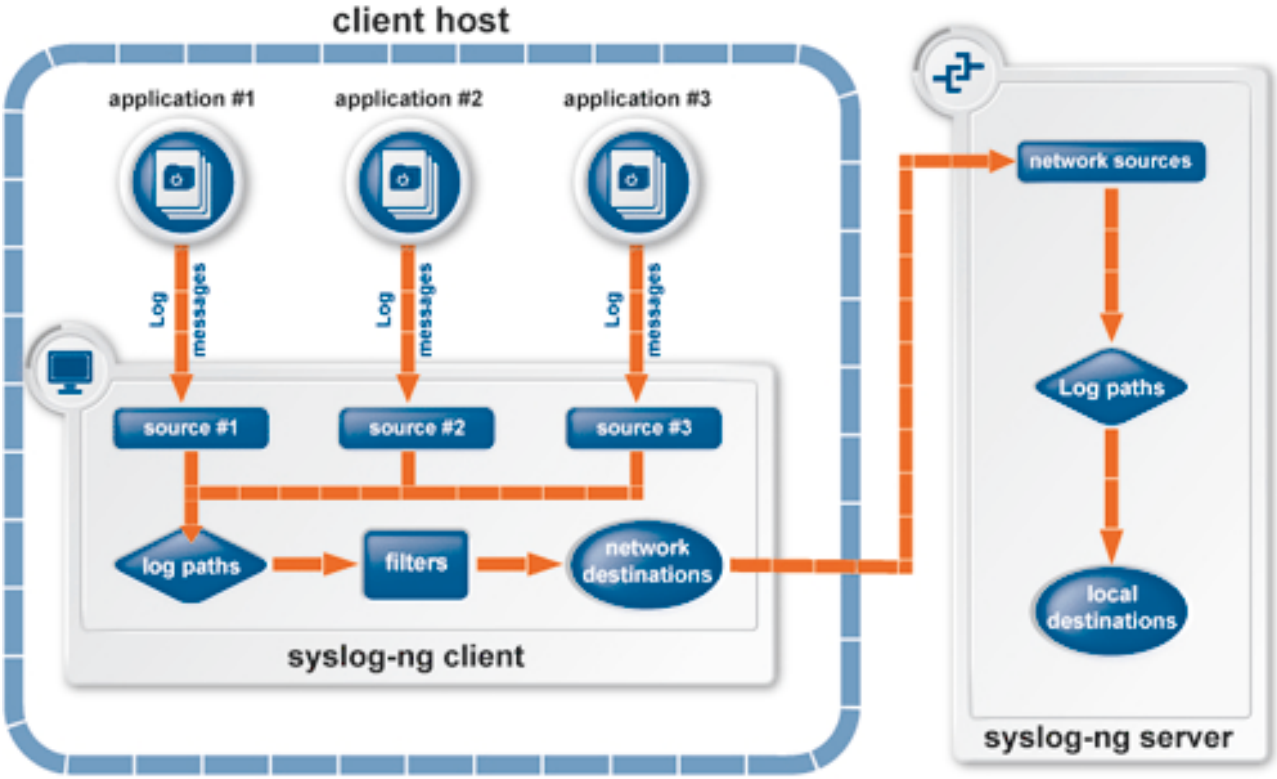

Fig 7.2 – Exemple d'une architecture SYSLOG

Le protocole SYSLOG a l'avantage de bien définir la plupart des éléments d'un événement :

- L'horodatage de l'événement est précis et normé.
- Le processus émettant l'événement est renseigné.
- Le niveau de sévérité de l'événement est renseigné.
- Cependant, comme le sujet de l'événement est libre et donc non défini par un OID le contenu peut être vague.

Parmi les avantages de SYSLOG on trouve aussi :

- L'utilisation des protocoles de transport TCP ou UDP.
- L'utilisation et la maintenance simplifiée comme système de stockage des messages pour les applications.
- SYSLOG étant un composant fondamental des systèmes UNIX. On le retrouve présent sur quasiment tout type de matériels.

SYSLOG permet de stocker les événements dans des fichiers logs, et si l'architecture correcte est en place, de manière centralisée. Cependant, les données s'accumulant, la recherche dans une grande quantité de fichiers textes devient rapidement très lente. Il nous faut donc une solution alternative de stockage aux fichiers de logs, solution que la couche ELK fournit.

7.3 ELK : Une couche multi-usages

7.3.1 Introduction

Les protocoles décrits auparavant sont surtout orientés dans le transfert des événements. Afin de rendre le stockage et l'interrogation de ceux-ci exploitable, il faut rajouter des outils de normalisation, de stockage et de visualisation. A cet effet, une solution a été sélectionnée chez PSA : ELK.

ELK est la combinaison de trois logiciels libres spécialisés dans ces domaines : Elasticsearch [4], Logstash [6] et Kibana [5]. Cette couche logicielle a été choisie car il s'agit d'un projet de logiciel libre qui rencontre un grand succès et ayant un support commercial disponible.

7.3.2 Elasticsearch : Un moteur de recherche et de stockage

7.3.2.1 Présentation

Elasticsearch est logiciel libre créé en 2010 par Shay Banon. Il s'agit d'un logiciel servant de moteur de recherche et de stockage des données. Suite au fort succès rencontré par le projet, la société "Elasticsearch BV" a été créée en 2012 afin de fournir un support commercial à Elasticsearch.

Le projet est basé sur le projet Apache Lucene [3] et est développé en Java. Apache Lucene est une librairie facilitant l'indexation et l'interrogation d'une grande quantité de données. Afin de rendre possible le stockage et l'interrogation d'une grande quantité de donnée de manière rapide, Elasticsearch est conçu comme une application distribuée qui fonctionne sur plusieurs serveurs collaborant entre eux.

7.3.2.2 Architecture

Pour pouvoir stocker ou interroger des données Elasticsearch fournit une API [12] accessible via HTTP et utilisant le format de données JSON. Les méthodes d'accès sont donc très similaires à celles d'OpenStack.

Voici un exemple de requête permettant d'injecter un événement dans Elasticsearch :

```
curl -XPUT 'http://localhost:9200/mon_index/mon_type/1' -d '
{
    "date": "2015-09-03",
    "message": "message de test"
}'
```

Elasticsearch est une application distribuée utilisant le principe de cluster (Fig. 7.3). Un cluster est constitué de 2n+1 nœuds. Chaque nœud peut contenir plusieurs "indexes" qui eux-mêmes sont découpés en plusieurs "shards" répliqués sur différents nœuds pour éviter les pertes de données en cas de perte de nœuds. Les indexes contiennent des "documents" qui sont en réalité des structures de données au format JSON. Dans notre cas, un document correspond à un événement.

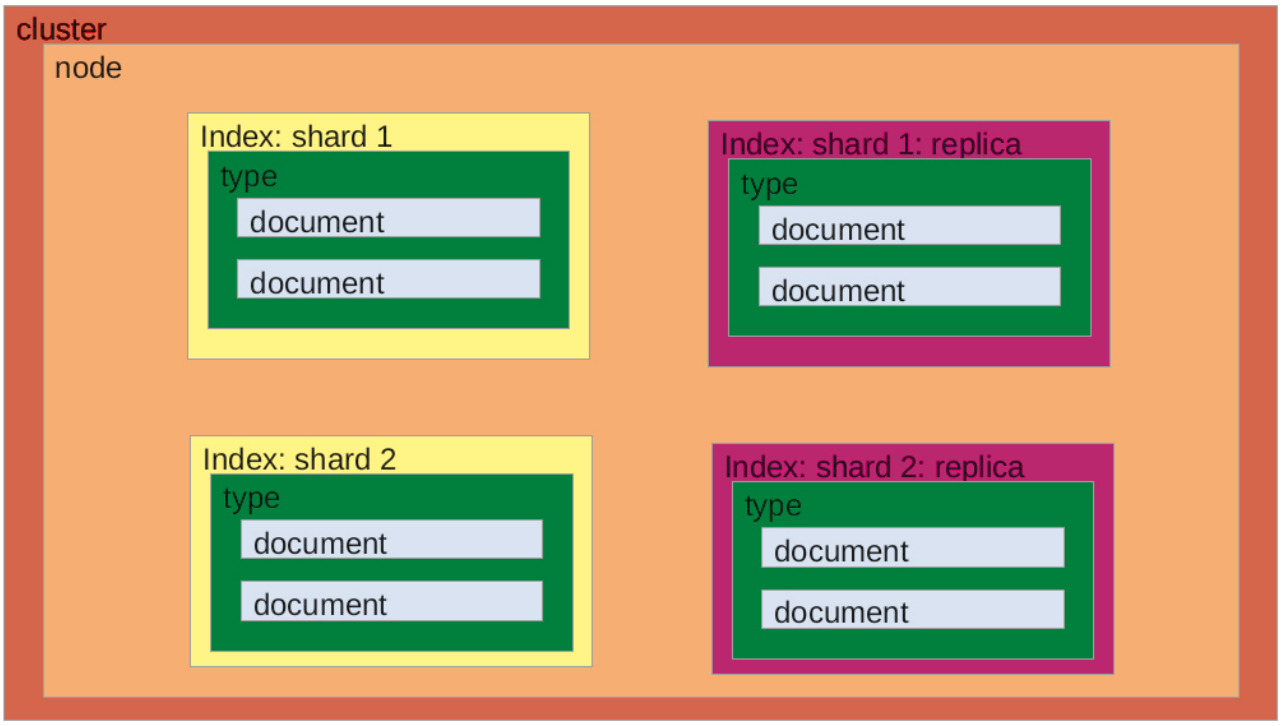

Fig 7.3 – Architecture d'Elasticsearch

7.3.3 Logstash : Le middleware pour les logs

7.3.3.1 Présentation

Logstash est logiciel libre crée en 2009 par Jordan Sissel et Pete Fritchman. Il s'agit d'un logiciel facilitant la gestion des logs et événements. Il est utilisé pour collecter, normaliser et retransférer les logs. Son utilisation aisée et le fait qu'il puisse enregistrer les événements dans Elasticsearch ont réellement fait décoller l'utilisation de la couche ELK pour la surveillance d'événements. Le créateur d'Elasticsearch, Shay Banon, a rapidement réalisé que la combinaison des deux logiciels pouvait être une grande opportunité de créer une entreprise supportant le tout. Ainsi, en 2013, Jordan Sissel fût embauché par Elasticsearch BV.

Le projet est développé en JRuby. JRuby est une implémentation du langage de programmation Ruby utilisant la machine virtuelle Java afin de s'exécuter.

7.3.3.2 Architecture

Logstash est le couteau suisse de la conversion de format d'événements et protocoles. Il est constitué d'une suite de plugins ayant trois responsabilités différentes (Fig. 7.4) :

- La récupération d'événements provenant de différents protocoles / logiciels [8] :
  - La lecture de fichiers de log.
  - L'extraction de données de Redis ou d'Elasticsearch.
  - La récupération de messages du logiciel de file de message RabbitMQ.
  - La récupération de messages utilisant le protocole SYSLOG.
  - La récupération de messages utilisant le protocole SNMP.
- La transformation des événements réceptionnés via des plugins de filtres [7] permettant de :
  - Agréger des événements entre eux.
  - Supprimer des événements.
  - Retoucher et normaliser les horodatages via le plugin "Grok". Grok étant une librairie permettant de découper et structurer, via des expressions régulières, un message qui n'est pas forcément bien structuré à l'origine.
  - Manipuler de manière totalement libre le message de l'événement via le plugin "Grok".
- L'envoi d'événements structurés vers différents protocoles / logiciels [9] :
  - L'écriture de fichiers de log ou de fichiers CSV.
  - L'envoi de données vers Redis ou Elasticsearch.
  - L'écriture de messages dans une file de messages.
  - L'envoi de données via le protocole SYSLOG.

Ceci ne représente que les plugins les plus couramment utilisés. Il en existe des dizaines et si la fonctionnalité qui nous intéresse n'est pas présente, on peut toujours créer un plugin personnalisé de manière assez simple.

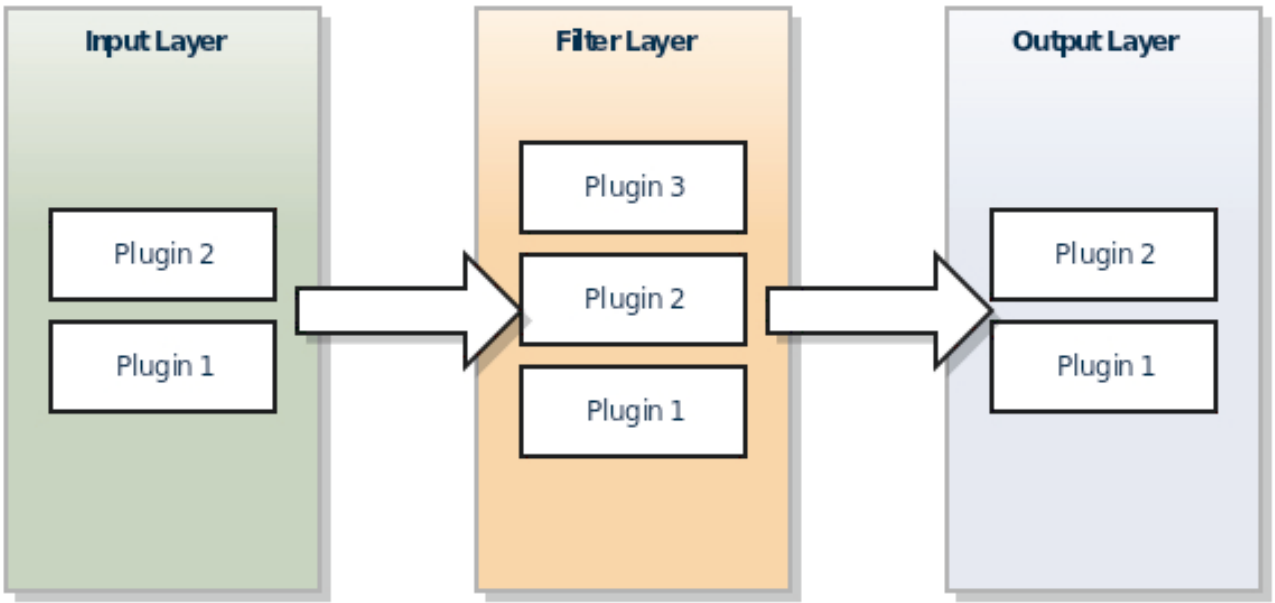

Fig 7.4 – Architecture de Logstash

7.3.4 Kibana : Un portail de consultation

Kibana est logiciel libre créé en 2012 par Rashid Khan qui a aussi été embauché par Elasticsearch BV. Il s'agit d'une interface web facilitant la consultation des données présentes dans Elasticsearch. Cette interface permet de mettre en place un tableau d'indicateurs qui est constitué d'outils d'analyse (graphiques, tableaux, ...) (Fig. 7.5). Cette visualisation synthétique peut se réaliser via différents critères : regroupement par équipement, par sévérité des événements ou par n'importe quel autre champ présent dans un événement.

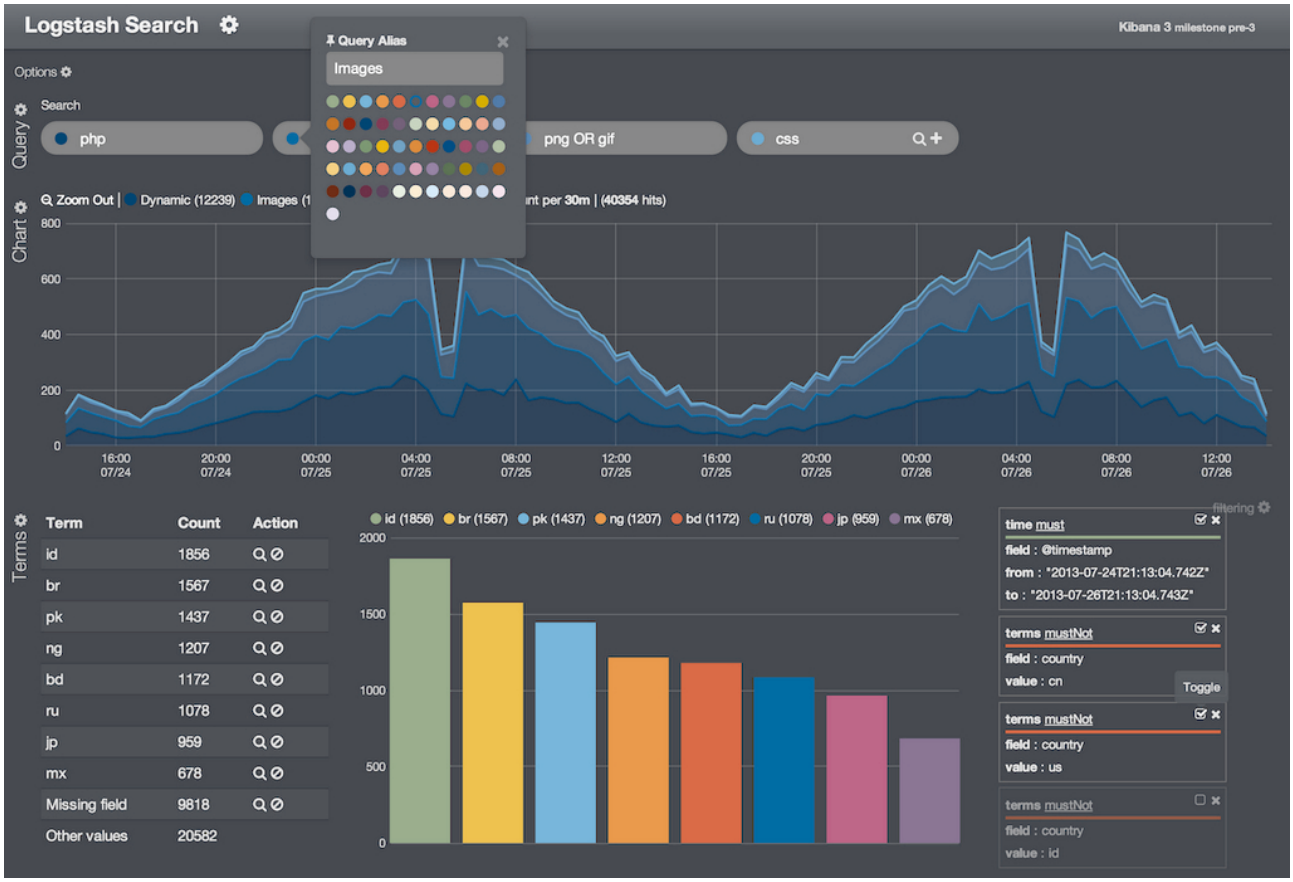

Fig 7.5 – Aperçu de Kibana

7.3.5 Une architecture modulaire

L'architecture des différents composants de la solution ELK définissent quel niveau de redondance et de performance l'ensemble va avoir. La solution la plus simple consiste à mettre en place une instance de chaque logiciel (Fig 7.6).

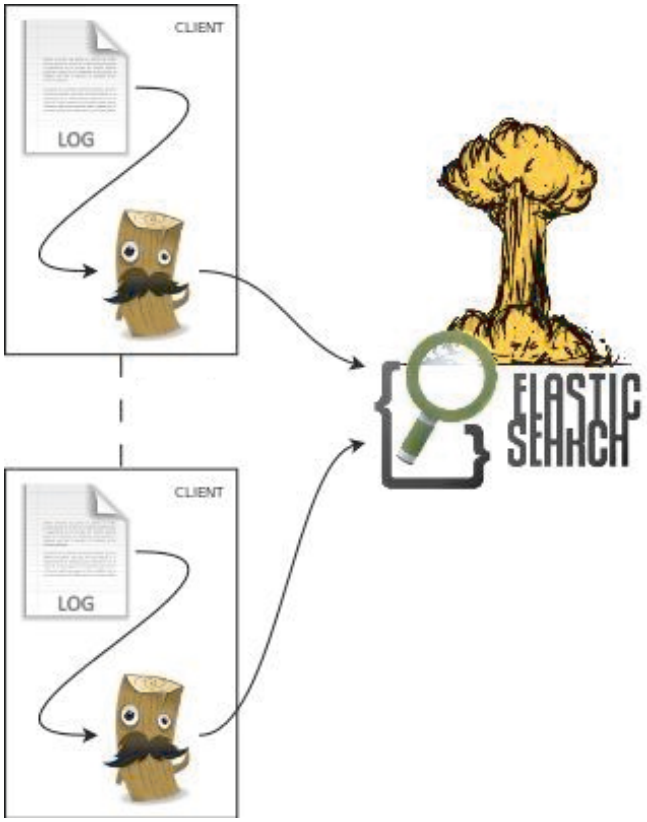

Fig 7.6 – Architecture Simple de ELK

Cette architecture simple est bien appropriée pour effectuer des tests rapidement. Cependant elle n'est pas très adaptée à une architecture de production pour plusieurs raisons :

- On ne peut pas forcément installer Logstash sur tous les équipements. En effet, les baies et switchs SAN sont des équipements propriétaires où l'administrateur n'as pas accès au système sous-jacent.
- Logstash est bien adapté pour être déployé sur des serveurs afin de lire et de retransmettre les logs systèmes ou applicatives. Cependant, si on utilise le protocole SYSLOG cela impliquerait de configurer celui-ci différemment sur chaque équipement afin de définir chaque instance de Logstash comme destination SYSLOG.
- Une seule instance d'Elasticsearch ne permet pas d'avoir une redondance suffisante en cas de panne ou d'intervention de mise à jour nécessaire sur l'instance Elasticsearch. Si Logstash lit des fichiers il peut se mettre en pause et reprendre la transmission une fois Elasticsearch de retour. Par contre si Logstash reçoit des événements via SYSLOG, comme il ne pourra pas les transmettre, il ne les acceptera pas et ceux-ci seront donc définitivement perdus.

Afin de résoudre certains de ces problèmes, on peut mettre en place une architecture différente (Fig. 7.7) :

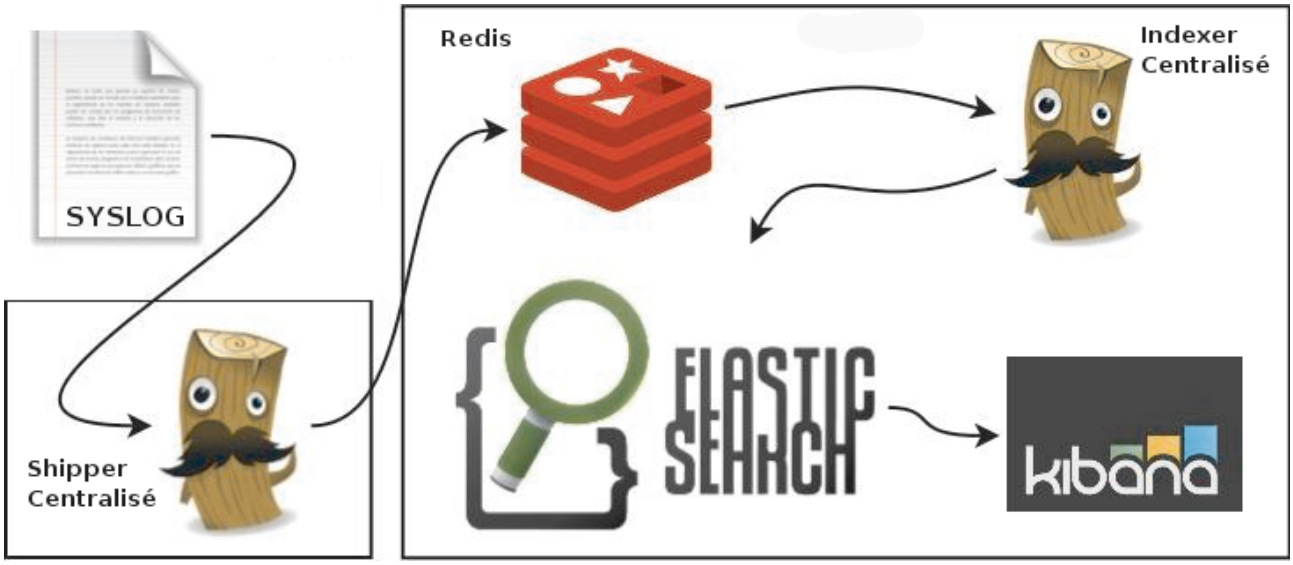

Fig 7.7 – Architecture SYSLOG de ELK

Cette architecture découpe Logstash en deux parties et introduit Redis [11] entre les deux. Redis est un dépôt de structures de données. Il est souvent utilisé comme cache pour des bases de données ou comme logiciel de file de message au même titre que RabbitMQ.

- Une partie "shipper" centralisée ayant la responsabilité de récolter et retransmettre les événements vers Redis. Ces événements peuvent alors provenir d'autres Logstash installés sur des serveurs ou directement arriver via SYSLOG. L'avantage est que chaque équipement utilisant SYSLOG sera configuré de la même manière. Cette partie réalisera le moins de traitement possible sur les événements.
- Une partie "indexer" centralisée ayant la responsabilité de récupérer les événements dans Redis, effectuer les traitements des événements et envoyer le tout vers Elasticsearch.

Cette architecture a le principal avantage de permettre de ne pas perdre de messages en cas d'indisponibilité d'Elasticsearch. En effet, si Elasticsearch venait à être indisponible, l'indexer n'enverrait plus de messages et ne lirait plus de messages en provenance de Redis. Redis continue cependant à se remplir de messages en provenance du "shipper". Le shipper ne se bloquera donc plus et les événements utilisant SYSLOG comme protocole ne seront plus perdu. Au redémarrage d'Elasticsearch l'indexer commencera à traiter les messages en attente dans Redis. Redis étant spécifiquement conçu pour ce type d'utilisation il peut stocker un très grand nombre de messages sans impacter ses performances.

Afin de rendre cette architecture complètement redondante il ne reste plus qu'à multiplier les instances de chaque composants et les relier entre eux (Fig. 7.8).

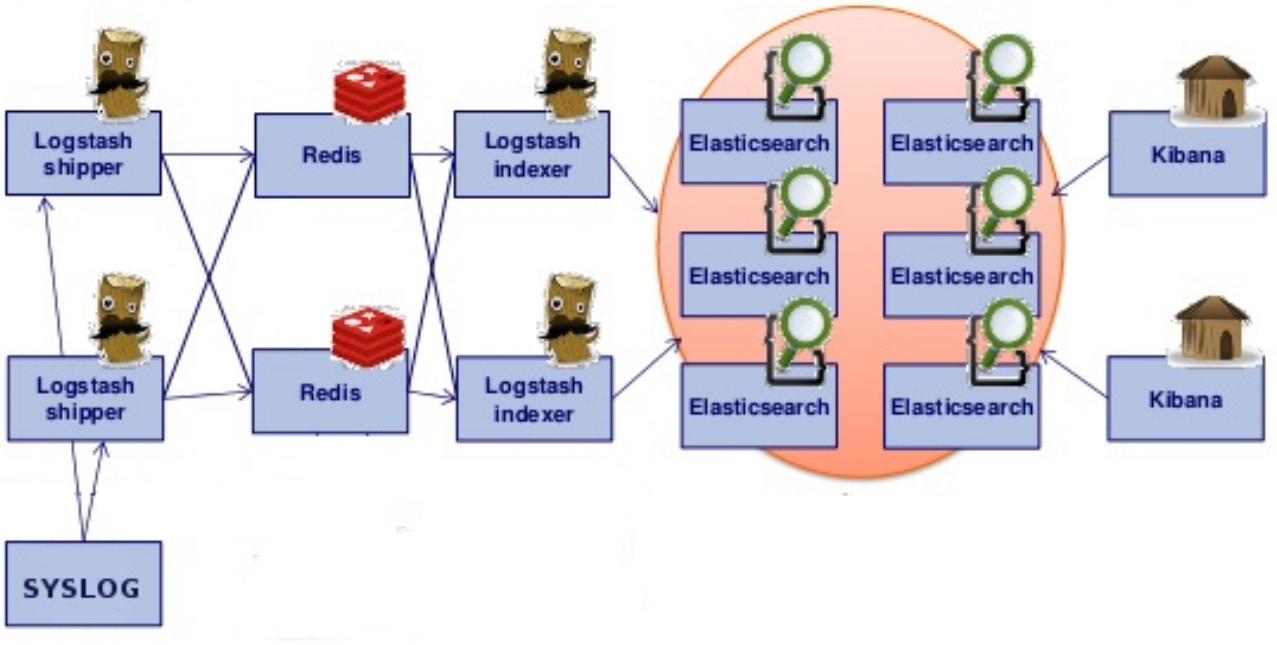

Fig 7.8 – Architecture redondante SYSLOG de ELK

Il reste cependant deux problèmes à résoudre dans cette architecture. Les deux se situant au niveau des points d'entrées SYSLOG dans les shipper Logstash :

- Certains équipements n'acceptent qu'une destination SYSLOG.
- Ceux qui en acceptent plusieurs envoient le même événement aux deux destinations.

Ces problèmes peuvent être résolus en réutilisant des composants de l'architecture des contrôleurs OpenStack (Fig. 5.7). C'est-à-dire utiliser une unique adresse IP virtuelle (VIP) via keepalived et utiliser Linux Virtual Server [10] sur chacun des serveurs hébergeant shipper Logstash afin de répartir la charge. LVS sera choisi pour la simple raison que celui-ci supporte aussi la répartition de charge du protocole UDP alors que HAProxy ne supporte que le protocole TCP.

7.3.6 Bilan

La centralisation des événements est un élément essentiel pour faciliter la détection, la classification et l'analyse de ceux-ci. Parmi les protocoles les plus connus, grâce à sa souplesse et à ses éléments de normalisation, le protocole SYSLOG parait le plus adapté à être utilisé pour transférer des événements.

Une fois le transfert des événements effectués, le couple Logstash et Redis est d'une grande utilité. Il permet de normaliser les événements qui ne le sont pas déjà et enrichir ceux-ci. De plus, via le nombre impressionnant de modules, Logstash permet de récupérer des événements provenant d'autres sources ne supportant pas le protocole SYSLOG. Redis quant à lui permet d'assurer une grande disponibilité de l'architecture et d'éviter les pertes de messages lors d'indisponibilités du moteur de stockage Elasticsearch.

Le stockage d'une grande quantité de données est relativement aisé. Cependant, la consultation d'une grande quantité de données peut être une tâche difficile. Il faut pouvoir répondre aux demandes de recherches dans un temps raisonnable. Il ne serait pas envisageable de stocker une telle quantité dans une base de donnée traditionnelle. Elasticsearch est spécifiquement conçu pour répondre à ce type de cas d'usage. Son architecture interne en fait un moteur de recherche très performant, tout en garantissant la future croissance des données ainsi que la redondance nécessaire à un système de surveillance. Son API simple et standardisée permet d'effectuer des requêtes complexes tout en garantissant des temps de réponses raisonnables pour les utilisateurs.

Le portail web Kibana est un plus appréciable et facilement abordable par des utilisateurs n'étant pas des développeurs. Il permet de générer très rapidement des rapports visuels mettant en exergue les événements importants parmi une très grande quantité de données.

Je vais à présent présenter l'architecture de surveillance en place au sein du service responsable du stockage. Nous allons voir pourquoi le projet de substitution de celui-ci par la couche logicielle ELK a été initié et réalisé. Je vais aussi aborder l'accostage de cette surveillance avec le projet de service d'infrastructure OpenStack.

Chapitre 8

ELK : Surveillance du SAN et du stockage

8.1 La surveillance chez SIXS

Le service SIXS, ayant la responsabilité de la gestion du stockage, a la mission de surveiller les équipements de stockage. Détecter rapidement les problèmes logiciels et matériels est un impératif car les conséquences de problèmes non traités peuvent rapidement aboutir à une situation mettant en péril la disponibilité des applications. Cette surveillance permet donc de planifier des interventions, détecter les problèmes récurrents et effectuer des actions préventives.

Actuellement, notre service a mis en place une surveillance sur trois types d'équipements :

- Les switchs SAN BROCADE.
- Les baies de virtualisation du stockage SAN IBM SVC.
- Les baies de stockage NAS NETAPP.

Les autres équipements utilisés dans notre service sont généralement directement surveillés par le constructeur du matériel.

Afin de surveiller ces équipements, il faut en premier lieu vérifier quels sont les protocoles de transmission d'événements disponibles sur chaque type de matériels. Voici un résumé pour nos trois équipements (Table 8.1) :

Protocole / Équipement	SNMP	SYSLOG	MAIL
Switchs BROCADE	OK	OK	OK
IBM SVC	OK	OK	OK
NETAPP	OK	OK	K0

Table 8.1 – Support des protocoles de surveillance par type de matériel

Historiquement, notre service a choisi deux de ces protocoles pour deux usages différents :

- Le protocole SNMP a été choisi afin de remonter les problèmes les plus critiques.
- L'envoi de mail par les équipements a été désigné pour remonter les événements de plus faible importance ne demandant pas d'intervention immédiate.

8.1.1 Gestion des alertes : SNMP chez PSA

Le protocole SNMP est donc utilisé afin de remonter les événements urgents nécessitant l'ouverture de tickets d'intervention. Les traps SNMP sont donc envoyés à un serveur centralisé ayant la responsabilité d'interpréter les événements et d'ouvrir les tickets d'intervention de manière automatisée. L'application en charge de ces tâche se nomme "OSP" (Outil de Surveillance et de Pilotage des systèmes d'information).

OSP reçoit donc les traps SNMP d'un grand nombre d'équipements différents. A l'aide des OID de ces traps, les événements sont classifiés par types de matériels et donc les tickets d'intervention sont attribués aux services respectifs de ces équipements. Afin de différencier les équipements, les équipes en charge des équipements doivent fournir à cette application une liste d'OID ainsi que les règles d'interprétation de ceux-ci (Fig. 8.1).

Equipement : brocade
Rechercher un terme :

NUMERO	ETAT	OID / MIB	DESCRIPTION ou TITRE INCIDENT	TYPE INCIDENT
1324993230	active		fw-111(3|5) : fabric e-port down : Mode Native	appli=STOCKAGE
1324548297	active		fw-1194 : port state change : Mode Native	appli=STOCKAGE
1305538440	active		fw-1194 : port state change : Access Gateway	appli=STOCKAGE
1324995418	active		fw-1424 : switch status changed : Mode Native	appli=STOCKAGE
1411395500	active		fw-1424 : switch status changed : Access Gateway	appli=STOCKAGE
1324996834	active		BROCADE : Alerte SAN a destination de ESO/AOSS	appli=STOCKAGE
1411397996	active		BROCADE : Alerte SAN a destination de OPER/BE	hard

Fig 8.1 – Exemple de liste de règles dans OSP pour les switchs SAN BROCADE

La centralisation de la gestion des règles permet de ne pas avoir de conflits dans la gestion des OID. Certains équipements de différents fournisseurs peuvent utiliser la même hiérarchie d'OID ce qui peut poser des problèmes. Heureusement, le système de vérification en place dans OSP vérifie que les différentes règles fournies par les différents services ne rentrent pas en conflit.

Le système de règles permettant de classifier les événements est basé sur les expressions régulières (Fig. 8.2). Chaque règle est en fait une expression régulière permettant d'identifier un unique type d'événement en analysant le contenu de la trap SNMP. Si le même OID parvient à OSP pour deux types de matériels complètement différents, on peut toujours distinguer quel est réellement le type d'équipement en se basant, par exemple, sur le nom DNS de celui-ci.

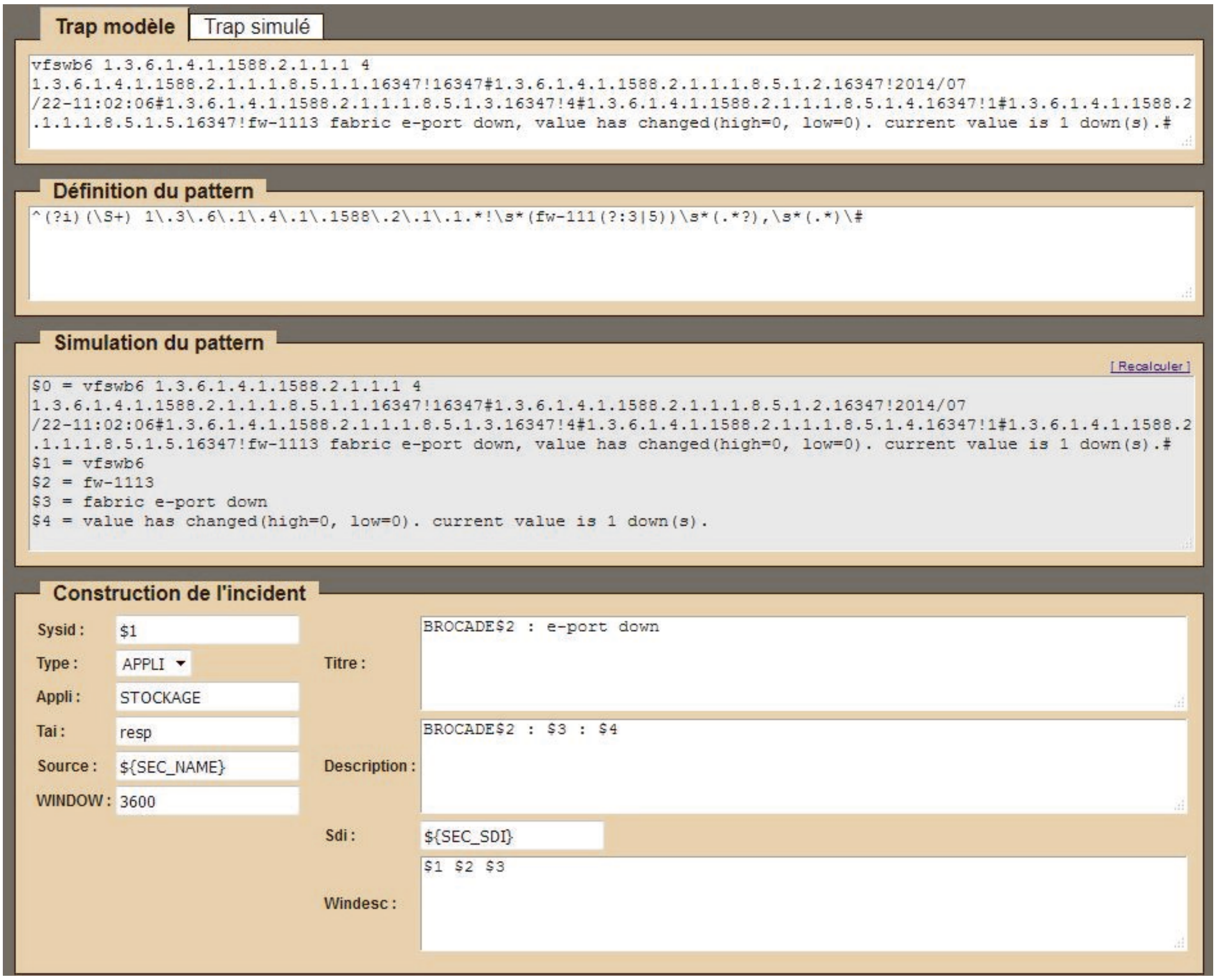

Fig 8.2 – Exemple d'une règle dans OSP

Malgré les avantages d'OSP, celui-ci possède quand même quelques lacunes majeures :

- Il n'existe pas d'interface permettant de voir tous les événements reçus, qu'ils soient traités ou non.
- Admettons que nous ayons 10 traps SNMP avec le même OID et qu'un seul champ varie. Si ce champ permet de distinguer à qui devrait être attribué le ticket d'intervention, il ne sera pas possible de faire une unique règle. L'interface d'OSP ne permettant pas de mettre des conditions du type "si ... alors ... sinon", il sera nécessaire de créer autant de règles que de cas particuliers. Le matériel IBM SVC est le cas typique où nous avons plus de 250 règles à maintenir à cause de ce manque de souplesse. Cela devient donc très rapidement une corvée à maintenir à jour.
- La surveillance basée sur les OID SNMP peut aussi être compliquée par le fait que, les matériels évoluant, les structures des traps SNMP et la hiérarchie des OID peuvent varier très fortement d'une version à l'autre.

8.1.2 Gestion des événements peu critiques : les mails

Certains équipements comme les switchs SAN BROCADE et IBM SVC supportent l'envoi de mail lorsque des événements sont déclenchés. Pour ne pas ouvrir des tickets d'intervention concernant des problèmes bénins, il fût décidé d'utiliser cette fonctionnalité. Les équipements furent donc configurés afin d'envoyer des mails à des listes de diffusion internes.

Ce mécanisme simpliste d'alerte a de nombreux inconvénients :

- Le problème principal de l'utilisation d'envoi de mail pour signaler des événements non critiques est que la probabilité de lecture et de traitement de ces événements est inversement proportionnelle au nombre de personnes à qui le mail a été envoyé. A moins qu'une personne réponde à l'ensemble de la liste de diffusion, on ne sait pas si quelqu'un traite le problème. La traçabilité du traitement d'un problème est donc très mauvaise via ce mode d'alerte.
- Lors des remaniements d'organisation il n'est pas rare que les listes de diffusion soit renommées demandant ainsi la modification de la configuration de nombreux équipements.
- Comme il s'agit d'une méthode séparée de l'envoi par SNMP, il est possible de recevoir un ensemble d'alertes mineures reliées à une alerte majeure, qui elle a un ticket d'intervention. Cela oblige à faire des allers-retours entre deux outils pour traiter le problème original et ses effets de bords.

8.2 Un nouvel outil de surveillance

8.2.1 Enjeux

L'enjeu du nouvel outil de surveillance du stockage est dans un premier temps de complémenter la surveillance actuelle. Une fois l'outil fiabilisé, il remplacera petit à petit l'envoi de mail voire l'envoi d'événements par SNMP. Cet outil est censé être relativement agnostique et réutilisable par d'autres services si ceux-ci sont intéressés par le principe.

Différents facteurs ont convergé rendant nécessaire l'étude d'une solution alternative :

- L'outil de surveillance via SNMP OSP est relativement peu configurable et difficile à maintenir.
- OSP ne permet pas de voir facilement l'historique des événements remontés.
- Comme indiqué auparavant, la surveillance par mail n'étant pas centralisée implique des lourdeurs de gestion.
- Il est prévu que fin 2015 l'envoi de mails anonymes ne soit plus autorisé pour des raisons de sécurité. Or, la plupart de nos équipements de stockage ne supporte pas l'authentification auprès d'un serveur de mail. Une solution nous a été proposé mais ne nous convient pas car complexifiant encore la surveillance. La solution consistait à permettre de manière exceptionnelle l'envoi de mails anonymes depuis des IP clairement identifiés. Or, nous avons des centaines d'équipements, ce qui engendrerait la nécessité de maintenir une grande liste d'IP manuellement.

Pour ces différentes raisons j'ai donc eu la tâche d'étudier une solution alternative afin de simplifier le traitement des problèmes au sein de mon service.

8.2.2 Solution retenue

Après analyse de quelques solutions pouvant remplacer notre système actuel de remontées d'événements, j'ai décidé de m'orienter sur la solution basée sur Elasticsearch, Logstash et Kibana. Cette solution répondant à nos besoins de centralisation, de modularité et de consultation des événements. J'ai aussi décidé d'utiliser le protocole SYSLOG afin de transporter les événements vers ce système, car ce protocole est simple et requiert relativement peu de maintenance.

Au même moment où je commençais à présenter cette solution dans mon service, M. FRICKER eu l'idée similaire d'utiliser cette couche logicielle pour d'autres cas d'usages. Je me suis donc tout naturellement rapproché de lui afin de participer en tant que projet pilote à la mise en place de cette infrastructure au sein de PSA. M. FRICKER, qui est devenu mon tuteur, a la responsabilité de la mise en place de l'infrastructure ELK, promouvoir son utilisation au sein de groupe et conseiller les utilisateurs.

Dans le cadre de mon projet pilote, j'ai donc eu les responsabilités suivantes :

- Automatiser la configuration du protocole SYSLOG sur nos équipements BROCADE, IBM SVC et NETAPP.
- Fournir une configuration pour Logstash pour normaliser les événements de ceux-ci.
- Fournir une personnalisation du portail Kibana pour chaque type d'équipements.
- Développer un portail web permettant non seulement la consultation mais aussi la prise en charge des alertes de manière suivie et centralisée. Ce portail sera nommé "Webalerts".

Voilà de manière synthétique à quoi ressemble l'architecture mise en place (les éléments de redondance n'y figurant pas pour plus de lisibilité) (Fig. 8.3) :

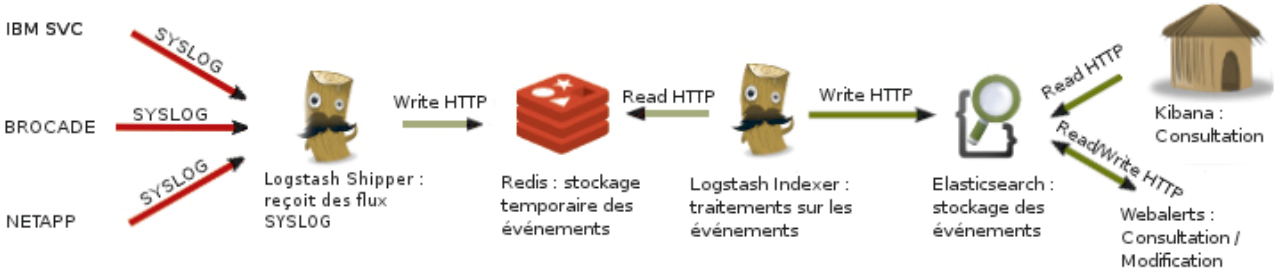

Fig 8.3 – Architecture du nouveau système d'alertes pour le stockage

8.2.3 Mise en place

8.2.3.1 Logstash

Afin de mettre en place la nouvelle solution de surveillance il a fallu dans un premier temps analyser et standardiser les événements envoyés par les équipements via SYSLOG. Logstash a été très utile pour cette tâche.

On retrouvera en annexes une configuration simplifiée de la partie shipper de Logstash (Annexe B.1) et de la partie indexer (Annexe B.2).

On pourra remarquer l'utilisation intensive des expressions régulières dans Logstash. On pourra aussi remarquer qu'avec uniquement sept expressions régulières on parvient à traiter l'ensemble des messages de trois types d'équipements différents. Cela contraste particulièrement avec les centaines de règles dans OSP. L'avantage principal de Logstash est l'utilisation des conditions dans ses fichiers de configuration. La syntaxe des fichiers de configuration est proche d'un langage de programmation, ce qui apporte une grande souplesse dans la manipulation des données.

Le principal challenge rencontré lors de la création du fichier de configuration concernait la gestion des dates dans la partie indexer. Il se trouve que les switchs SAN BROCADE ne précisent pas leur fuseau horaire dans l'événement. J'ai donc dû déterminer le fuseau horaire en fonction de l'emplacement géographique de ceux-ci. Les deux premières lettres d'un nom de switch donne son site d'installation. Grâce à cette indication j'ai pu convertir l'ensemble des événements vers le même fuseau horaire de référence : UTC. Cela est d'un grande aide lors de l'analyse de problème sur plusieurs sites géographiquement répartis. Cela évite la gymnastique de conversion mentale des fuseaux horaires.

Comme précisé dans la structure d'un fichier de configuration de Logstash, on retrouve les trois blocs principaux :

- Le bloc "input" définissant sous quel format arrive l'événement (en l'occurrence SYSLOG utilise le port UDP 514 pour la partie shipper et Redis pour la partie indexer).
- Le bloc "filter" permettant de manipuler les données de l'événement.
- Le bloc "output" convertissant l'événement au format nécessaire afin de l'écrire : Dans notre cas Redis pour la partie shipper et Elasticsearch pour la partie indexer.

8.2.3.2 Elasticsearch

Elasticsearch étant le dépôt de données, il a fallu estimer combien d'événements serait stockés sur la période de rétention définie à 90 jours. Les baies NAS NETAPP étant de loin les plus verbeuses, j'ai réalisé l'estimation suivante.

Estimation réalisée sur la moitié des baies NAS NETAPP sur une durée de 24 heures (Table 8.2) :

Nombre d'événements supprimés car inutiles	$1.4 * 10^6$
Nombre d'événements conservés	$1.7 * 10^4$
Nombre d'événements ne correspondant à aucune expression régulière	3
Taux d'échec de reconnaissance d'événements (%)	$1.76 * 10^{-4}$

Table 8.2 – Estimation du nombre d'événements dans Elasticsearch

Sur la totalité des baies cela représente donc environ 40.000 événements/jour soit 3.600.000 événements sur 90 jours. La taille des données dans Elasticsearch dépendra des mécanismes internes de compression et d'indexation. Avec seulement quatre expressions régulières pour le type de matériel NETAPP on arrive à analyser une écrasante majorité des événements.

8.2.3.3 Kibana

La troisième partie de ce projet consistait à configurer l'interface Kibana afin d'avoir pour chaque type d'équipement une synthèse graphique des événements. Voici un exemple pour le matériel IBM SVC (Fig. 8.4).

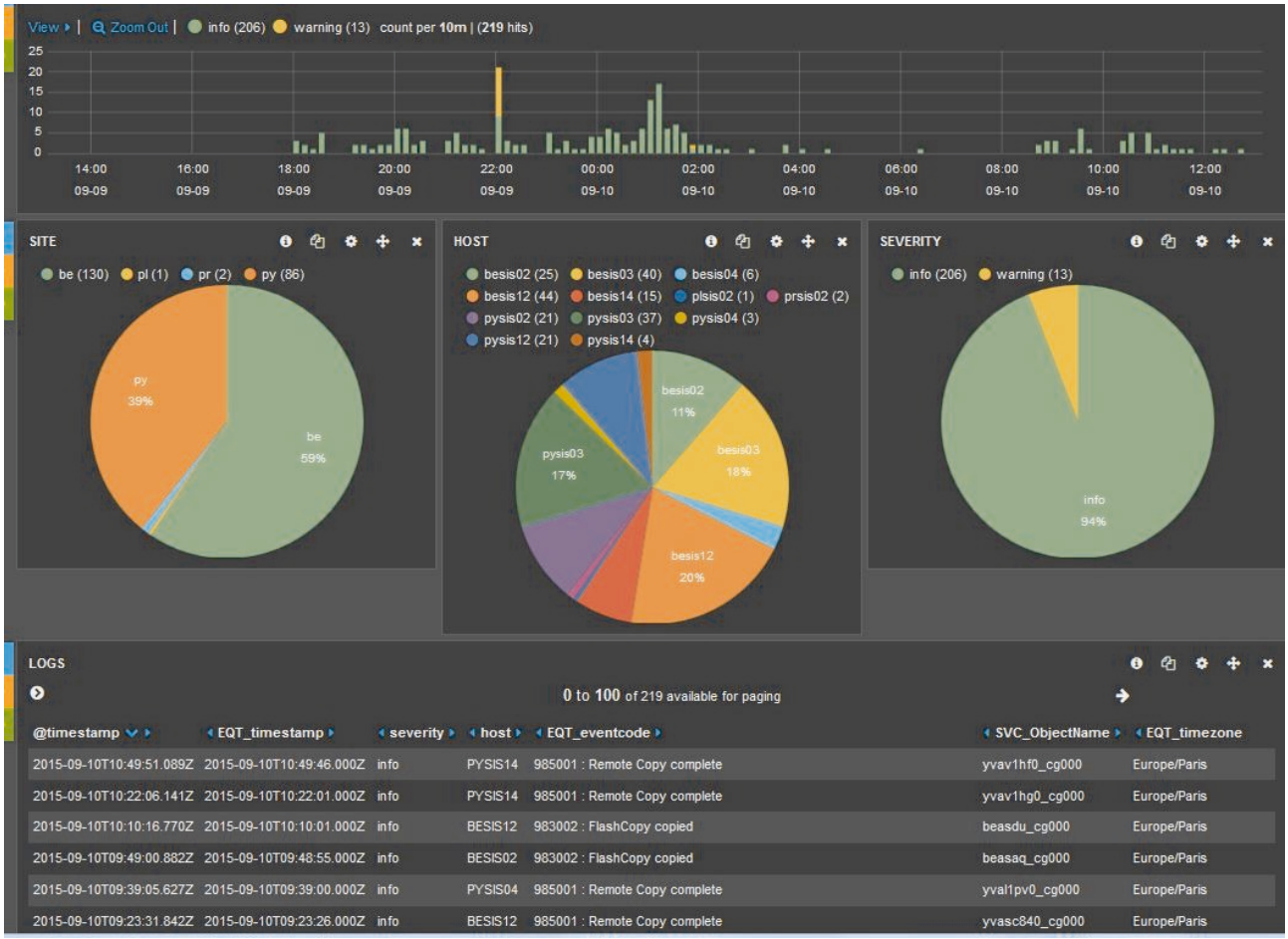

Fig 8.4 – Kibana pour IBM SVC

8.2.3.4 Webalerts

Finalement le dernier besoin était la réalisation d'une interface web minimaliste permettant de consulter et traiter les événements de manière centralisée. J'ai donc créé le site web "Webalerts" dans ce but (Fig. 8.5).

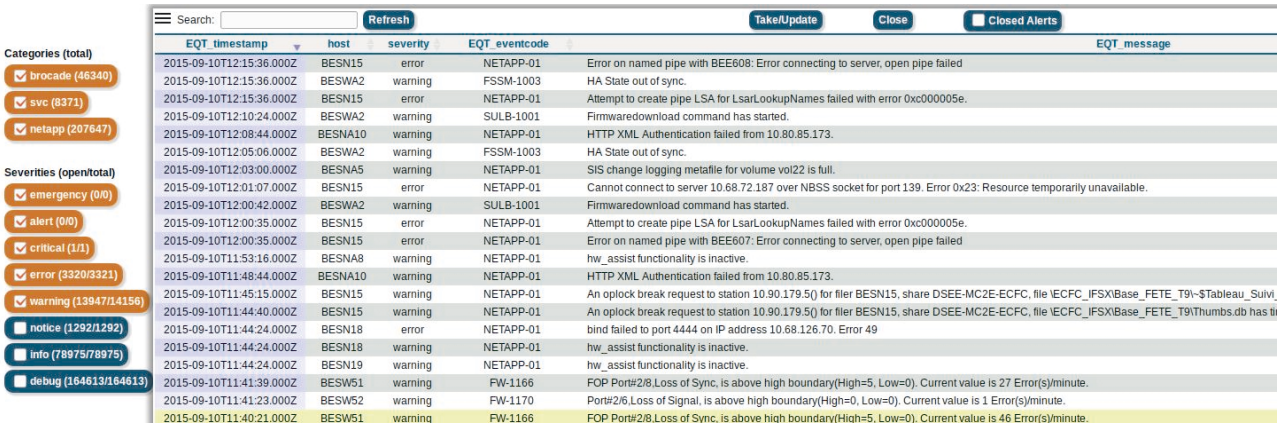

Fig 8.5 – Aperçu de Webalerts

Webalerts est une page web minimaliste permettant de :

- Filtrer les équipements par type.
- Filtrer les alertes par sévérité.
- Filtrer les événements sur le message selon un filtre personnalisé.
- Commenter une ou plusieurs alertes à la fois (Fig. 8.6).
- Clore une ou plusieurs alertes à la fois.
- Visualiser les alertes closes.

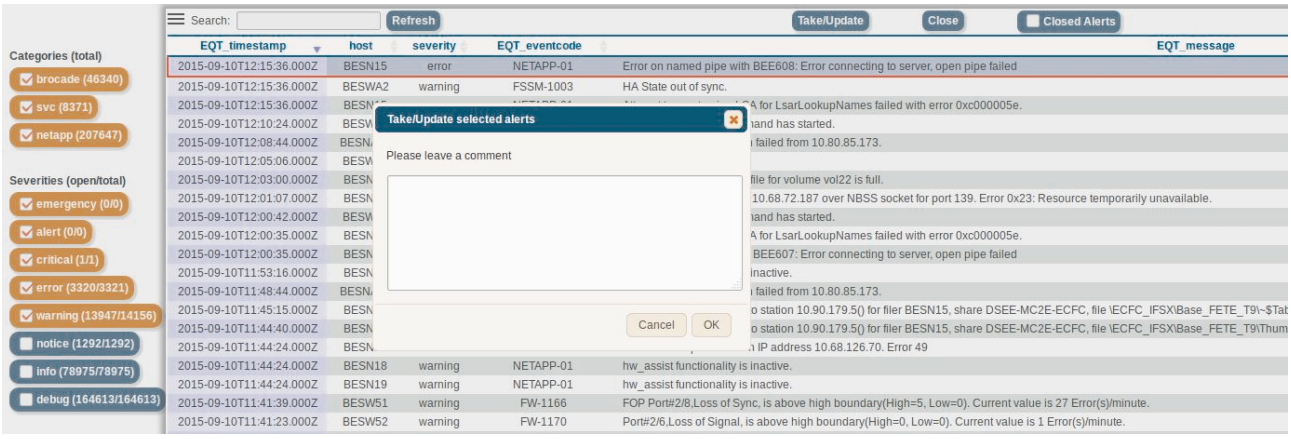

Fig 8.6 – Webalerts : Commentaire sur une alerte

Pour l'instant, cette interface n'est pas encore en production. Les travaux restant consistent à diminuer le nombre d'alertes remontées en retirant les faux positifs ainsi que les messages considérés comme inutiles. Afin de ne pas ralentir la chaîne de traitement des alertes, ce nettoyage d'alertes sera effectué à l'aide d'un traitement périodique. Il consistera à identifier et supprimer ces alertes directement dans Elasticsearch via l'API de celui-ci.

8.3 Bilan et opportunités

La gestion des événements est un domaine dans lequel il n'est pas facile de trouver un juste milieu. Trop d'événements entraîne une lassitude lors du traitement de ceux-ci, trop peu et il est possible de faire l'impasse sur certains qui peuvent avoir de graves conséquences. Même si l'on se place en amont, d'un point de vue architecture celle-ci n'est pas toujours évidente à mettre en place. La multitude des équipements et protocoles rend ardue la centralisation des événements.

Comme aucune solution miracle n'existe, la plupart des systèmes de surveillance en place dans les entreprises résultent d'empilement de nombreuses expérimentations. Afin de tenter de rationaliser la surveillance le projet d'utilisation de la couche ELK dans le domaine du stockage a été initié. Après analyse des différentes solutions et protocoles, Elasticsearch est naturellement apparu comme la solution la mieux adaptée. Bien que l'architecture de cette solution semble complexe, sa configuration est relativement simple à maintenir par rapport à d'autres solutions historiquement présentes dans le groupe comme OSP.

La couche ELK apporte une centralisation renforcée des événements quels que soient leurs niveaux de criticité. Il en résulte une meilleure vision d'ensemble de l'état de l'infrastructure ainsi qu'une optimisation du traitement des événements. Cet ensemble d'outils est facile à adapter aux besoins d'une entreprise comme PSA, tout en restant abordable par des utilisateurs non-initiés. La finesse de configuration de Logstash, la rapidité de recherche dans une grande quantité de données d'Elasticsearch et la convivialité de l'interface web Kibana forment une couche logicielle bien adaptée pour ce cas d'usage.

D'autres cas d'usages sont actuellement déjà exploités chez PSA en utilisant ELK :

- Centralisation et analyse des temps de réponses requêtes de nombreux sites web de PSA : cela a permis de fortes améliorations des temps de réponses.
- Remontée et affichage en temps réel des performances des serveurs.
- Centralisation des logs de l'application en charge du cycle de vie d'un produit (conception véhicule, ...).

Pour résumer, la couche logicielle ELK répond à nos besoins originaux de souplesse de configuration, de redondance de l'infrastructure et de centralisation accentuée des alertes. La propagation de son utilisation en interne de PSA dépendra principalement de l'implication des différentes équipes et de leurs besoins spécifiques. La centralisation des événements n'étant que la première étape nécessaire afin à l'exploitation de ceux-ci. Le principal de la valeur ajoutée se situera dans les usages intelligents des données.

Conclusion

Les évolutions des méthodes de déploiement et de surveillance de l'infrastructure sont des sujets vastes et complexes. Les besoins de fournir de larges infrastructures rapidement et de manière standardisée se font de plus en plus ressentir. Il n'est donc plus approprié de remanier des outils vieillissants et peu adaptés aux nouvelles contraintes. Lors de ce mémoire, j'ai donc tenté d'exposer deux projets majeurs nécessaires au renouvellement de nos outils afin de répondre aux nouvelles attentes de nos utilisateurs.

Ces deux projets sont ambitieux et s'étendent sur une longue durée. Bien qu'ayant débuté ces projets il y a deux années, ces projets sont toujours en phase de test et commencent réellement à aboutir sur des solutions viables pour une utilisation sur des environnements de production. Les longs délais s'expliquent par l'envergure de ces projets. En effet, un grand nombre de domaines de l'infrastructure sont impactés par l'utilisation de ces nouveaux outils. Outre les défis techniques à relever, de nouveaux modes de travail et d'organisation sont nécessaires afin de mener ces projets à terme.

Les problématiques d'organisation sont les plus difficiles à résoudre. Comme les nouveaux outils couvrent un large éventail de technologies, plusieurs domaines de l'infrastructure doivent collaborer entre eux. A l'opposé de la séparation actuelle des équipes, ce type d'outil devrait être sous la responsabilité d'une équipe pluridisciplinaire. Il sera donc nécessaire de redéployer des compétences internes dans une nouvelle équipe, chose ardue en période de restriction budgétaire. Néanmoins, cette transformation va de pair avec les nouveaux modes de gestion techniques de l'infrastructure. L'un sans l'autre serait certainement voué à l'échec comme le montre de nombreux exemple de projets similaires dans d'autres entreprises. Ces projets à longs termes impliquent aussi une période où les anciens et nouveaux modes de gestion seront présents dans l'entreprise, rajoutant ainsi encore de la complexité dans la gestion du personnel et de la diversité dans les solutions à maintenir.

Malgré les difficultés rencontrées, les plausibles cas d'usages de ces nouveaux outils sont variés. On peut imaginer utiliser le service d'infrastructure OpenStack dans de nombreux domaines où l'élasticité des ressources est essentielle tels que :

- Les frontaux des sites web lors de période de fortes charges (promotions et offres diverses) afin de ne pas surdimensionner l'infrastructure pour tenir la charge uniquement présente lors de courtes périodes.
- Les centres de calcul pour la Conception Assisté par Ordinateur (CAO).
- Les domaines en forte croissance tel que le véhicule connecté.
- Le Data Mining des Réseaux Sociaux pour l'analyse des sentiments.
- Les environnements de tests et développements en couplant OpenStack avec une couche logicielle Plateform as a Service (PaaS).

La surveillance basée sur la couche logicielle ELK pourrait aussi être étendue dans son utilisation :

- En surveillant non seulement le domaine du stockage, mais aussi plus de types d'équipements. Il faudrait donc proposer son utilisation à d'autres équipes.
- En centralisant les logs du service d'infrastructure OpenStack dans Elasticsearch. La distribution OpenStack HP Helion utilise déjà ce principe en interne afin de faciliter la visibilité et la résolution de problèmes.
- En centralisant les événements provenant des logs systèmes des systèmes d'exploitation. Ceci permettrait de corréler les événements entre les équipements et les systèmes d'exploitation.
- En remontant les événements provenant des systèmes de sécurité tels que les pare-feux, les annuaires LDAP, ...
- En substituant petit à petit l'ancien système de surveillance (OSP) par cette nouvelle méthode.

Bien que m'étant principalement concentré sur l'utilisation de ces outils dans le domaine du stockage, ces deux projets m'ont permis d'aborder d'autres domaines de spécialités. Ces projets de veille technologique m'ont aidé à aborder des problématiques plus globales dans le monde de l'infrastructure. La portée et la durée de ceux-ci vont continuer à m'apporter des défis quotidiens qui devront être relevés en collaboration avec l'ensemble de l'équipe d'architecture.

> *Don't have good ideas if you aren't willing to be responsible for them.*
>
> Alan Perlis, *Epigrams on Programming, 1982*

Bibliographie

[1] Chef : gestion de configuration. https://www.chef.io/chef.

[2] Cloud Computing. https://fr.wikipedia.org/wiki/Cloud_computing.

[3] Apache Lucene. http://lucene.apache.org.

[4] Elasticsearch. https://www.elastic.co/products/elasticsearch.

[5] Kibana. https://www.elastic.co/products/kibana.

[6] Logstash. https://www.elastic.co/products/logstash.

[7] Logstash Filter Plugins.
https://www.elastic.co/guide/en/logstash/current/filter-plugins.html.

[8] Logstash Input Plugins.
https://www.elastic.co/guide/en/logstash/current/input-plugins.html.

[9] Logstash Output Plugins.
https://www.elastic.co/guide/en/logstash/current/output-plugins.html.

[10] Linux virtual server. http://www.linuxvirtualserver.org.

[11] Redis. http://redis.io.

[12] Elasticsearch REST API.
https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html.

[13] Git : gestion de versiions. https://git-scm.com.

[14] HAProxy : Répartition de charge. http://www.haproxy.org.

[15] Keepalived : Haute disponibilité. http://www.keepalived.org.

[16] MySQL : base de donnée. https://www.mysql.com.

[17] NASA's Nebula Cloud Computing Initiative.
http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120011651.pdf.

[18] Open Source IaaS Community Analysis. http://www.qyjohn.net/?p=3801.

[19] OpenStack Cinder API.
http://developer.openstack.org/api-ref-blockstorage-v2.html.

[20] OpenStack Cinder Client.
http://git.openstack.org/cgit/openstack/python-cinderclient.

[21] OpenStack Cinder Support Matrix.
https://wiki.openstack.org/wiki/CinderSupportMatrix.

[22] OpenStack Community Welcome Guide.
https://www.openstack.org/assets/welcome-guide/OpenStackWelcomeGuide.pdf.

[23] OpenStack : Five years in.
http://fr.slideshare.net/openstack/openstack-five-years-in.

[24] Opening the Rackspace Cloud.
http://www.rackspace.com/blog/opening-the-rackspace-cloud.

[25] OpenStack HP Helion. http://www8.hp.com/us/en/cloud/hphelion-openstack.html.

[26] OpenStack OpenSUSE. https://en.opensuse.org/Portal:OpenStack.

[27] OpenStack os-brick. http://git.openstack.org/cgit/openstack/os-brick.

[28] OpenStack Summit. https://www.openstack.org/summit/openstack-paris-summit-2014.

[29] OpenStack SUSE Cloud. https://www.suse.com/fr-fr/products/suse-cloud.

[30] Principe de Pareto. https://fr.wikipedia.org/wiki/Principe_de_Pareto.

[31] RabbitMQ : file de messages. https://www.rabbitmq.com.

[32] RFC5424 : The Syslog Protocol. http://tools.ietf.org/html/rfc542.

[33] Wireshark : Analyseur de paquet. https://www.wireshark.org.

Annexe

A OpenStack

A.1 Configuration de Cinder : SVC + Fibre Channel

Configuration du driver SVC dans cinder.conf :

```
[SVC-TEST]
# Le driver SVC
volume_driver=cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver
# L'ip de la baie SVC
san_ip = 10.68.199.161
# Le compte d'administration et la clé SSH associée
san_login = admin
san_private_key = /var/lib/cinder/.ssh/id_rsa_svc
# Nom du mdiskgrp sur le SVC
storwize_svc_volpool_name =  BESCL90-STD-01
# Utilisation du protocole Fibre Channel ainsi que du multipathing
storwize_svc_connection_protocol = FC
storwize_svc_multipath_enabled = True
# Option obligatoire suite à un bug : https://bugs.launchpad.net/cinder/+bug/1471749
storwize_svc_npiv_compatibility_mode = True
```

Configuration du driver des Switchs SAN BROCADE pour le zoning automatique dans cinder.conf :

```
# Mettre dans la section [DEFAULT]
zoning_mode=fabric

# Mettre dans la section [fc-zone-manager]
# Le driver des switchs
brcd_sb_connector=\
cinder.zonemanager.drivers.brocade.brcd_fc_zone_client_cli.BrcdFCZoneClientCLI
zone_driver=cinder.zonemanager.drivers.brocade.brcd_fc_zone_driver.BrcdFCZoneDriver
# Politique de zoning
zoning_policy=initiator-target
# Liste des fabric a activer
fc_fabric_names=FAB1

# Exemple de configuration de la fabric 1 de test~:
[FAB1]

# IP d'un des switchs de la fabric
fc_fabric_address=10.68.199.13
# Le compte d'administration et le mot de passe
fc_fabric_user=admin
fc_fabric_password=*******
# WWN du switch
principal_switch_wwn=100000051e0461cc
```

A.2 Configuration de Cinder : SVC + ISCSI

Configuration du driver SVC dans cinder.conf :

```
[SVC-TEST]
# Le driver SVC
volume_driver=cinder.volume.drivers.ibm.storwize_svc.StorwizeSVCDriver
# L'ip de la baie SVC
san_ip = 10.68.199.161
# Le compte d'administration et la clé SSH associée
san_login = admin
san_private_key = /var/lib/cinder/.ssh/id_rsa_svc
# Nom du mdiskgrp sur le SVC
storwize_svc_volpool_name =  BESCL90-STD-01
# Utilisation du protocole ISCSI
storwize_svc_connection_protocol = iSCSI
```

A.3 Configuration de Cinder : NETAPP + ISCSI

Configuration du driver NETAPP dans cinder.conf :

```
[NETAPP-TEST]
# Le driver NETAPP
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
# L'ip, le port et le type de la baie NETAPP
netapp_server_hostname=10.68.124.231
netapp_server_port=80(
netapp_storage_family=ontap_cluster
# Le compte d'administration et le mot de passe associé
netapp_login=admin
netapp_password=*******
# Utilisation du protocole ISCSI
netapp_storage_protocol=iscsi
```

A.4 Configuration de Cinder : NETAPP + NFS

Configuration du driver NETAPP dans cinder.conf :

```
[NETAPP-TEST]
# Le driver NETAPP
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
# L'ip, le port et le type de la baie NETAPP
netapp_server_hostname=10.68.124.231
netapp_server_port=80
netapp_storage_family=ontap_cluster
# Le compte d'administration et le mot de passe associé
netapp_login=admin
netapp_password=*******
# Utilisation du protocole NFS
netapp_storage_protocol=nfs
# Le fichier de configuration définissant quels volumes sont a utiliser pour les exports
nfs_shares_config=/etc/cinder/baie1_shares.conf
```

Annexe

B Elasticsearch

B.1 Configuration de Logstash : partie shipper

```
input {
    # Input of an udp socket for SYSLOG
    udp {
        port    => 514
        workers => 30
        type    => "udp_equipement"
    }
}

filter {
    # Drop empty messages
    if [message] == "" {
        drop { }
    }

    # Pattern matching by constructor
    # Matching based on frequency of hosts types in logs for better performance
    grok {
        # NETAPP
        match => [ "message", "^.*?>%{CISCOTIMESTAMP:EQT_timestamp} \[%{WORD:EQT_host}:
        (\s)?%{DATA:progname}:%{LOGLEVEL:severity}\]: %{WORD:NETAPP_user}
        (@\[%{IP:NETAPP_userip}.*?\])?:.*?:%{WORD:NETAPP_proto}.*?:
        (\[%{IP:NETAPP_userip}.*?\]:)?%{DATA:EQT_message}(\s+)?$" ]
        match => [ "message", "^.*?>%{CISCOTIMESTAMP:EQT_timestamp} \[%{WORD:EQT_host}:
        (\s)?%{DATA:progname}:%{LOGLEVEL:severity}\]:(.*?:)?%{DATA:EQT_message}$" ]
        match => [ "message", "^.*?>%{CISCOTIMESTAMP:EQT_timestamp} \[%{GREEDYDATA:progname}:
        %{LOGLEVEL:severity}\]:(.*?:)? %{WORD:NETAPP_user}(@\[%{IP:NETAPP_userip}.*?\])?:.*?:
        %{WORD:NETAPP_proto}.*?:(\[%{IP:NETAPP_userip}.*?\]:)?%{DATA:EQT_message}(\s+)?$" ]
        match => [ "message", "^.*?>%{CISCOTIMESTAMP:EQT_timestamp} \[%{GREEDYDATA:progname}:
        %{LOGLEVEL:severity}\]:(.*?:)?%{DATA:EQT_message}$" ]

        # BROCADE
        match => [ "message", "^.*?>%{WORD:progname}: .*?20%{DATESTAMP:EQT_timestamp}.*?
```

```
        , \[%{DATA:EQT_eventcode}\],(.* \| .*?,)? %{LOGLEVEL:severity}
        , (.*/)?%{WORD:EQT_host},(.*?,)?\s+%{DATA:EQT_message}$" ]
        match => [ "message", "^.*?>%{WORD:progname}: .*?20%{DATESTAMP:EQT_timestamp}.*?
        , \[%{DATA:EQT_eventcode}\], %{INT},
        , %{LOGLEVEL:severity}, %{WORD:EQT_host},\s+(,\s+)?%{DATA:EQT_message}$" ]

        # SVC
        match => [ "message", "^.*?>%{CISCOTIMESTAMP} %{WORD:EQT_host} %{WORD:progname}:
        .*? Record Type = %{LOGLEVEL:severity}\s+#\s+%{GREEDYDATA:EQT_message}$" ]
    }

    # Find host type based on it's name
    # Matching based on frequency of hosts types in logs for better performance
    if [EQT_host] =~ "[A-Z]{2}[SUX]N[A-Z0-9]{2,4}" {
        # NETAPP
        mutate {
            replace => [ "type", "netapp" ]
        }
    } else if [EQT_host] =~ "[A-Z]{2}S[WD][A-Z0-9]{2,4}" {
        # BROCADE
        mutate {
            replace => [ "type", "brocade" ]
        }
    } else if [EQT_host] =~ "[A-Z]{2}SIS[A-Z0-9]{2,4}" {
        # SVC
        mutate {
            replace => [ "type", "svc" ]
        }
    }
}

output {
    # Output messages to redis
    if "_grokparsefailure" not in [tags] {
        # If the parsing succes we send it to redis
        redis {
            host                => [ "yval0900", "yval0910" ]
            data_type           => "list"
            key                 => "logstash"
            congestion_interval  => 30
            congestion_threshold => 4000000
```

```
        }
    }  else {
        # If the parsing fails we logged it
        file {
            codec => rubydebug
            path  => "/users/sth00/logs/sixs_udp_grokfailure.log"
        }
    }
}
```

B.2 Configuration de Logstash : partie indexer

```
input {
    # Input of redis
    redis {
        host      => "yval0900"
        port      => "6379"
        data_type => "list"
        key       => "logstash"
        codec     => "json"
        threads   => 10
    }
    redis {
        host      => "yval0910"
        port      => "6379"
        data_type => "list"
        key       => "logstash"
        codec     => "json"
        threads   => 10
    }
}

filter {
    # Date normalisation
    translate {
        dictionary  => [
            "CY", "Europe/London",
            "MG", "Europe/London",
            "KA", "Europe/Moscow",
            "PL", "America/Buenos_Aires",
            "PR", "America/Sao_Paulo",
            "NU", "America/Sao_Paulo",
            "SN", "Asia/Shanghai"
        ]
        field       => "EQT_site"
        destination => "EQT_timezone"
        fallback    => "Europe/Paris"
    }
```

```
    mutate {
        add_field => [ "fulltimestamp", "%{EQT_timestamp} %{EQT_timezone}" ]
    }

    date {
        match => [ "fulltimestamp", "YY/MM/dd-HH:mm:ss ZZZ",
                                    "EEE MMM dd HH:mm:ss YYYY ZZZ",
                                    "EEE MMM  d HH:mm:ss YYYY ZZZ",
                                    "MMM dd HH:mm:ss ZZZ",
                                    "MMM  d HH:mm:ss ZZZ",
                                    "YYYY/MM d HH:mm:ss ZZZ",
                                    "YYYY/MM  d HH:mm:ss ZZZ" ]
        target => "temptimestamp"
    }

    mutate {
        replace      => [ "EQT_timestamp", "%{temptimestamp}" ]
        remove_field => [ "temptimestamp", "fulltimestamp" ]
    }
}

output {
    if "_grokparsefailure" not in [tags] {
        # If the parsing succes we send it to elasticsearch
        elasticsearch {
            protocol  => "node"
            host      => "yval0900"
            cluster   => "ELA_TEST"
            node_name => "yval0900_indexer"
            index     => "%{clust}%{idx}-%{+YYYY.MM.dd}"
        }
    } else {
        # DEBUG logging
        file {
            codec => rubydebug
            path  => "/users/sth00/logs/sixs_red_grokfailure.log"
        }
    }
}
```

Table des figures

Liste des tableaux

le cnam

Michaël LABOUEBE

Architecture Cloud Computing chez PSA

le cnam

Résumé :

Les systèmes d'informations deviennent de plus en plus complexes et exigeants. Les infrastructures matérielles et logicielles évoluent à un rythme effréné et les services informatiques doivent donc constamment s'adapter afin de répondre aux exigences de délais de mise à disposition et de continuité de service. Pour accompagner ces évolutions, des nouveaux modes de déploiement d'infrastructures ne cessent d'émerger et l'un d'entre eux est le logiciel de service d'infrastructure OpenStack.

Qu'est-ce qu'un logiciel de service d'infrastructure ? Quels en sont les caractéristiques ? OpenStack est-il capable de répondre aux besoins naissant des services informatiques ? La première partie de ce mémoire tentera donc de répondre à ces questions et traitera de l'administration du stockage dans OpenStack.

Le niveau très poussé d'automatisation des tâches élémentaires par OpenStack exige la mise en œuvre de puissants mécanismes de traçabilité. Ceci afin de faciliter et améliorer l'analyse d'éventuels dysfonctionnements. Les moyens de surveillance ont donc évolué afin d'être plus flexibles et mieux intégrés à ces nouveaux outils. La deuxième partie de ce mémoire abordera un outil de surveillance, par traitement des logs, utilisable dans le contexte d'OpenStack : Elasticsearch.

Mots clés :

PSA, OpenStack, Elasticsearch, Logstash, Logging, Cloud Computing

Abstract :

Information systems are becoming increasingly complex and demanding. Hardware and software infrastructures are evolving at a rapid pace and therefore IT departments must constantly adapt to meet resources provisioning deadlines and business continuity requirements. To deal with these evolutions, new infrastructure deployment architectures emerge every day and one of them is the infrastructure as a service software : OpenStack.

What is "infrastructure as a service" software ? What are its characteristics ? Is OpenStack able to meet the emerging needs of IT services ? The first part of this paper will therefore try to answer these questions and address storage administration within OpenStack.

With the degree of automation in OpenStack, a high level of traceability is required to facilitate problem analysis and solving. As a result, these monitoring tools have evolved to become more flexible and be better integrated within the OpenStack framework. The second part of this paper will therefore explore a log processing tool used to monitor OpenStack : Elasticsearch.

Keywords :

PSA, OpenStack, Elasticsearch, Logstash, Logging, Cloud Computing