Industrialisation du déploiement de machines virtuelles

Nicolas Olive

le cnam

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL DE TOULOUSE

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

SPECIALITE : INFORMATIQUE

OPTION : SYSTEMES D'INFORMATIONS

par

OLIVE, Nicolas

Industrialisation du déploiement de machines virtuelles

Soutenu le 15 janvier 2014

JURY

PRESIDENT : **Professeur Anne Wei**

MEMBRES : **Monsieur Thierry Millan**
Monsieur Bertrand Raiff
Monsieur Nicolas Cros
Monsieur François Sarron

Remerciements

Ce mémoire est l'aboutissement de plusieurs années d'études au sein du CNAM.

Tout d'abord, je tiens à remercier l'ensemble du corps enseignant de Toulouse, dont Monsieur Raiff qui a suivi mon étude et a été moteur dans l'aboutissement de sa réalisation. Je souhaite également remercier le Président du jury ainsi que ses membres.

Ce mémoire est la concrétisation de mon expérience professionnelle, pour cela je remercie globalement la société CLS et plus spécifiquement Monsieur Cros. C'est en effet dans cette société que j'ai commencé le cursus Ingénieur du CNAM dès le début de mon emploi en tant que technicien bureautique. Dans une relation gagnant-gagnant, et avec le support des plusieurs personnes qui se reconnaitrons, j'ai pu obtenir un poste d'ingénieur et avoir une responsabilité qui me permet de m'épanouir aujourd'hui professionnellement.

Table des matières

Introduction

La société CLS est spécialisée dans la localisation et la collecte de données satellite. Pour répondre à ces activités, elle dispose d'applications avec des engagements de disponibilités 24h/24, 7j/7 avec certains de ses clients. Pour maintenir ce niveau de service, la direction informatique définit, conçoit et assure le bon fonctionnement des infrastructures hébergeant ces applications.

Avec son centre de données en forte évolution lié à l'augmentation constante de l'activité, donc du nombre de serveurs, la direction informatique se retrouve confrontée à des problématiques de gestions et de déploiements de serveurs, majoritairement virtuels. Le choix de l'étude d'une solution d'industrialisation a été soumis dans le domaine de la virtualisation, car il s'agit de la technologie en forte croissance et dont la récente utilisation permet d'entrevoir de nouvelles possibilités.

Dans une première partie, nous présenterons le contexte dans lequel j'ai réalisé ce projet. Nous ferons ensuite une analyse de la problématique et des besoins exprimés par l'équipe système, que nous poursuivrons avec un état des lieux et une explication sur le choix de la solution pour répondre aux besoins, à savoir l'utilisation de modèle au format OVF. Pour la réalisation du projet, elle sera décrite en quatre sous parties : la définition d'un socle technique, la réalisation d'une maquette, la mise en place organisationnelle et enfin la mise en production de la solution. Avant de conclure, nous présenterons les résultats obtenus ainsi que des perspectives sur la technologie OVF.

I Contexte

En premier lieu, nous allons présenter CLS (Collecte Localisation Satellites) qui est la société dans laquelle je travaille et dans laquelle j'ai réalisé mon mémoire. Dans un second temps, nous nous intéresserons brièvement aux différentes applications de CLS. Enfin, nous décrirons l'organisation de CLS avant d'en présenter son système d'information.

I.1 Collecte Localisation Satellites

CLS, filiale du Centre National d'Études Spatiales (CNES), est né en 1986 du regroupement d'actionnaires suivant : le CNES, l'IFREMER (Institut Français de Recherche pour l'Exploitation de la MER) et de banques françaises comme illustré par la figure 1.

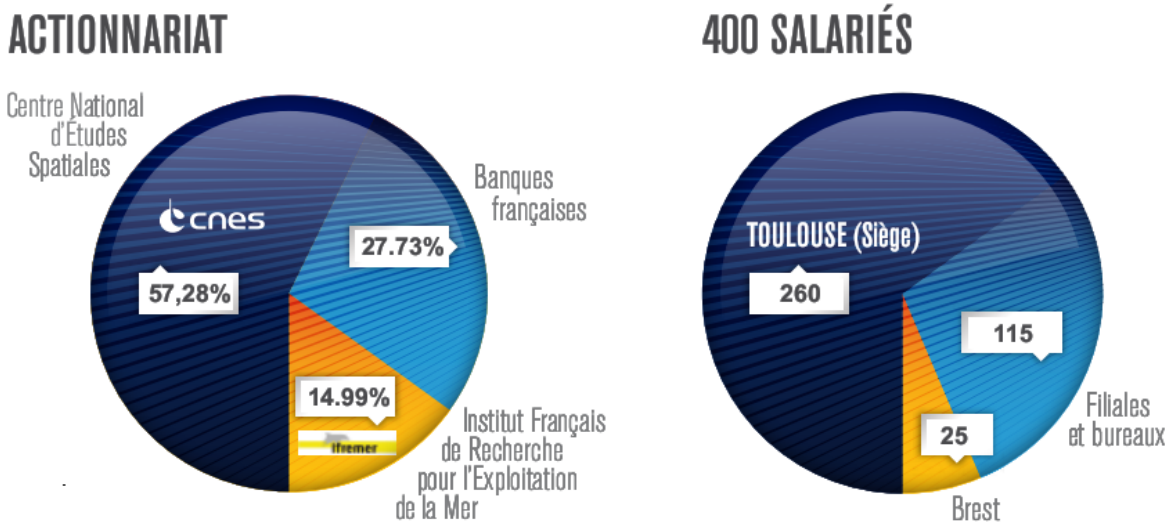

Figure 1 : CLS en chiffre

CLS compte environ 400 salariés dont 260 présents au siège qui est situé en périphérie de Toulouse, à Ramonville Saint-Agne. CLS possède des filiales en France, et est également représenté internationalement avec des filiales aux Etats-Unis, au Pérou, en Indonésie et en Espagne. Des bureaux annexes servant de relais pour les utilisateurs locaux sont également implantés dans certains endroits stratégiques comme au Japon, en Russie, en Corée du Sud, au Chili ou encore en Australie.

La figure 2 représente CLS à l'échelle mondiale.

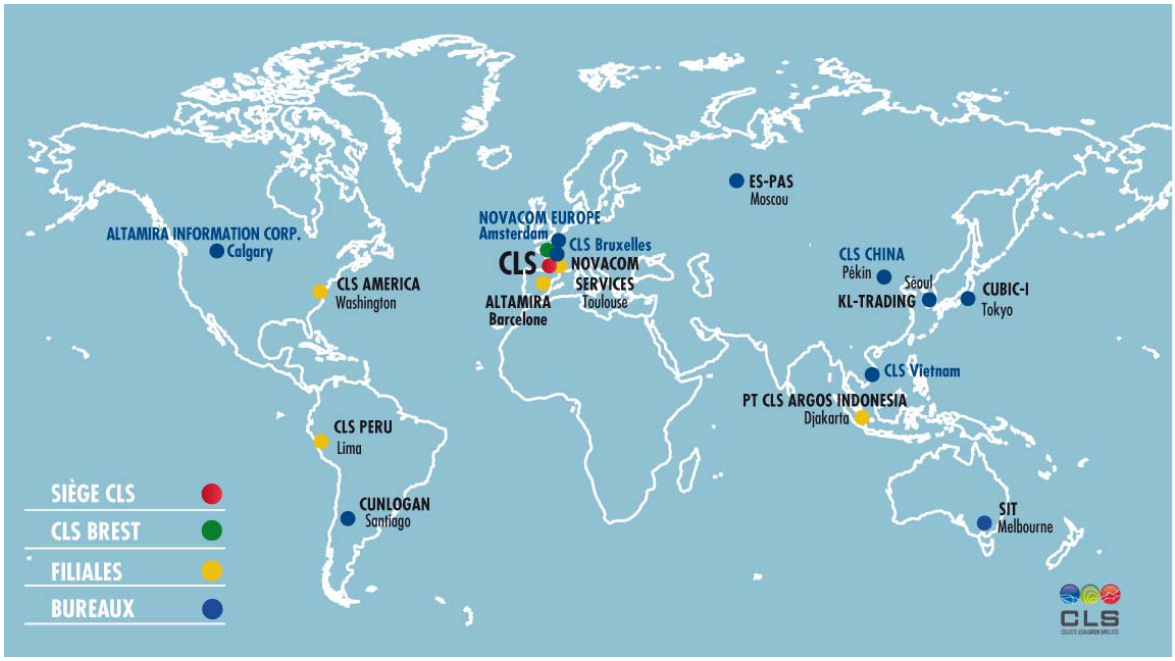

Figure 2 : CLS à l'échelle mondiale

I.1.1 Présentation du CNES

Avec 57% de ses parts détenues par le CNES, CLS est une filiale du CNES. Créé sous l'impulsion du Général de Gaulle, le 19 décembre 1961, le CNES est chargé d'élaborer et de proposer au gouvernement la stratégie spatiale française et de la mettre en œuvre. De 1961 à 1981, le CNES va être le moteur de l'Europe spatiale. Durant ces années, les structures indispensables à un programme spatial sont mises en place : lanceurs, satellites, ensemble de lancements, centres d'opérations et réseau de stations de contrôle, laboratoires. Parallèlement, une industrie spatiale compétente et dynamique voit le jour en France. C'est au sein de l'Agence Spatiale Européenne (ESA) que le CNES a contribué à la création de la fusée Ariane, et devient de ce fait une grande agence, où de nombreux programmes à vocation internationale lui sont confiés. En effet, le CNES représente la France à l'ESA et il recadre avec succès ses activités sur un programme national ambitieux orienté vers les applications. En 1977, les directives du gouvernement mettent l'accent sur les missions prioritaires de l'établissement qui sont notamment de qualifier le plus rapidement possible la fusée Ariane et lancer sa production. Cet objectif est atteint le 24 décembre 1979 avec le lancement réussi d'Ariane 1 de la base de Kourou. Une autre mission

prioritaire est l'étude interne d'un programme national de satellites d'observation de la Terre. Ce sera le projet du satellite Spot dont l'étude est réalisée au Centre Spatial de Toulouse. Les activités du CNES en faveur de la recherche scientifique ne diminuent pas mais les réductions de budget orientent les réalisations vers des programmes en coopération et vers des expériences embarquées sur des satellites de la NASA, de l'URSS et de l'ESA. On retiendra deux projets marquants qui sont Argos et TOPEX-Poséidon. C'est le projet Argos qui a abouti à la création de la filiale CLS.

Le CNES est composé du Centre Spatial de Toulouse (CST), de Kourou en Guyane française et d'Evry. Les activités du centre de Toulouse sont le management des projets, les études de recherche et technologie, la gestion des centres d'opération pour les mises à poste et la gestion en orbite, la mise en œuvre des moyens informatiques et d'études mathématiques et les supports administratifs, logistiques et de communication. Il regroupe environ 1500 personnes dont une majorité d'ingénieurs et cadres.

Le centre de Kourou est dénommé "port spatial" de l'Europe, depuis lequel les lanceurs Ariane ou depuis peu Soyouz et Vega sont envoyés dans l'espace. Le site de Kourou possède une position géographique exceptionnelle, proche de l'équateur, qui autorise des lancements vers l'est (en bénéficiant de la vitesse d'entraînement de la Terre) ou au nord dans des conditions maximales de sécurité. En effet, le lanceur ne survole aucune terre avant 4000 km.

Enfin, la Direction des Lanceurs s'installe dans la ville nouvelle d'Évry en 1974 à la fermeture de Brétigny-sur-Orge. Elle assure le développement des lanceurs Ariane et elle prépare l'avenir en travaillant sur les nouvelles générations de lanceurs et de systèmes de propulsion.

I.1.2 Présentation de l'IFREMER

L'IFREMER, l'Institut Français de Recherche pour l'Exploitation de la MER, est un établissement public à caractère industriel et commercial sous la tutelle du ministère de l'Écologie, de l'Énergie, du Développement durable et de la Mer et du ministère de l'Alimentation, de l'Agriculture et de la Pêche. Ses missions principales sont de connaître, d'évaluer et de mettre en valeur les ressources des

océans et permettre leur exploitation durable. Elles sont aussi d'améliorer les méthodes de surveillance, de prévision d'évolution, de protection et de mise en valeur du milieu marin et côtier. Elles favorisent également le développement économique du monde maritime.

L'IFREMER entretient des relations internationales à travers diverses coopérations. En effet, l'Ifremer participe activement aux travaux de l'Union Européenne (programmes de la direction générale de la Recherche et de la direction générale de la Pêche) et au Marine Board de la Fondation Européenne de la Science (ESF). Il contribue aux programmes internationaux de recherche (étude du climat, de l'environnement et de la biodiversité). Il anime aussi de nombreux accords bilatéraux (Japon, États-Unis, Canada, Australie, pays européens).

L'IFREMER est présent dans 26 sites répartis sur tout le littoral métropolitain et dans les DOM-TOM. L'Institut est composé de 5 centres (Boulogne, Brest, Nantes, Toulon et Tahiti), d'un siège social (Issy-les-Moulineaux) et d'une vingtaine de départements de recherche rattachés à ces centres.

I.1.3 Présentation de CLS

CLS, Collecte Localisation Satellites, est spécialisée dans la localisation et la collecte de données satellites, l'observation et la surveillance des océans mais également la géolocalisation de mobiles terrestres par satellite. CLS fournit à ses clients des services opérationnels et des solutions clés en main, issus de l'exploitation de systèmes spatiaux mondiaux, articulés autour de trois grands axes :

- **La surveillance environnementale**
    - Surveiller l'océan.
    - Préserver les grandes espèces migratrices.
    - Définir et évaluer des règlementations publiques.
    - Soutenir les politiques industrielles maritimes.

- **La gestion durable des ressources marines**
    - Administrer une pêche raisonnée.
    - Lutter contre la pêche illicite, non déclarée, non réglementée.
    - Modéliser les écosystèmes marins et mettre en place un aménagement des pêches.
    - Aider au développement des pêcheries locales.
- **La sécurité maritime**
    - Suivre les navires à l'échelle du globe.
    - Surveiller des zones d'intérêt stratégique.
    - Déclencher et coordonner des actions en mer.

I.2 Les applications

Comme le montre la figure 3, une grande partie du résultat d'exploitation de CLS est réalisée par la collecte et localisation de données et l'océanographie spatiale.

Figure 3 : Répartition du résultat d'exploitation

Alors que l'océanographie spatiale est utilisée principalement dans le cadre de la surveillance environnementale, la collecte et localisation est, elle, beaucoup plus diversifiée. En quelques mots, voici quelques exemples d'applications des systèmes exploités ou opérés par CLS :

- Argos : Système de localisation et de collecte de données par satellite dédié à l'étude et à la protection de l'environnement de notre planète. CLS traite les données des 21 000 balises Argos réparties sur les mers et continents du globe. L'entreprise décode les signaux reçus et les transmet à ses clients. Ainsi, un biologiste peut depuis son bureau observer la migration de l'espèce qu'il étudie, ou encore un ministère des affaires maritimes peut suivre l'ensemble des mouvements d'une flotte de pêche et un patron d'armement peut surveiller les pérégrinations de ses bateaux de fret où qu'ils soient dans le monde.
- Iridium : CLS est qualifiée VAR[1] Iridium, elle est donc habilitée à traiter, intégrer et valoriser des données Iridium. L'application CLS propose ainsi de la géolocalisation en temps réel avec une couverture totale du globe terrestre en s'appuyant sur la constellation Iridium équipées de 66 satellites. CLS utilise Iridium en voie montante et descendante ce qui permet d'avoir une localisation régulière ou à la demande. Elle est utilisée dans le cadre de la pêche pour la surveillance des ressources halieutiques, mais aussi pour le suivi des courses de bateaux comme le Vendée Globe, la Solitaire du Figaro ou encore la très célèbre Route du Rhum.
- CLS est une entreprise spécialisée en Océanographie Spatiale. Pour ce faire elle opère 80 instruments embarqués à bord de près de 80 satellites. Grâce à ces données, les 90 océanographes de l'entreprise sont capables de mesurer le niveau moyen des mers du globe au millimètre près. Ces données sont indispensables aux climatologues et météorologues du monde entier. En dehors de ce paramètre, la température, les courants

[1] Value Added Reseller, Revendeur à valeur ajoutée

ainsi que la couleur de l'eau sont envoyés aux scientifiques du monde entier.

- Radar : Pour compléter les deux familles de systèmes précédemment citées, dans les années 2000, CLS a voulu se doter de la capacité de surveiller les activités maritimes. Elle possède ainsi la capacité d'acquisition et de traitement d'images satellites haute résolution et peut donc offrir à ses clients des solutions de surveillance en temps quasi réel, de jour comme de nuit. L'interprétation experte des images satellites radar acquises par CLS permet de lutter contre la pêche illégale, de surveiller le trafic maritime, de lutter contre les actions illégales mais également de mesurer la hauteur des vagues, ou encore de « voir » le vent et soutenir ainsi l'industrie éolienne offshore.

I.3 Organigramme

Nous allons présenter ici l'organisation interne de CLS, qui est divisée en sept directions.

I.3.1 Organisation générale de CLS

Outre les directions administratives telles que la Direction Générale (DG), la Direction Financière (DF) et la Direction des Ressources Humaines (DRHC), nous allons présenter les directions métiers de CLS.

Comme le montre la figure 4, la Direction Technique (DT) forme un axe sur lequel reposent les « business units» de la diffusion commerciale des produits à savoir les directions Collecte et Localisation (DCL), Océanographie Spatiale (DOS) et Application Radar (DAR).

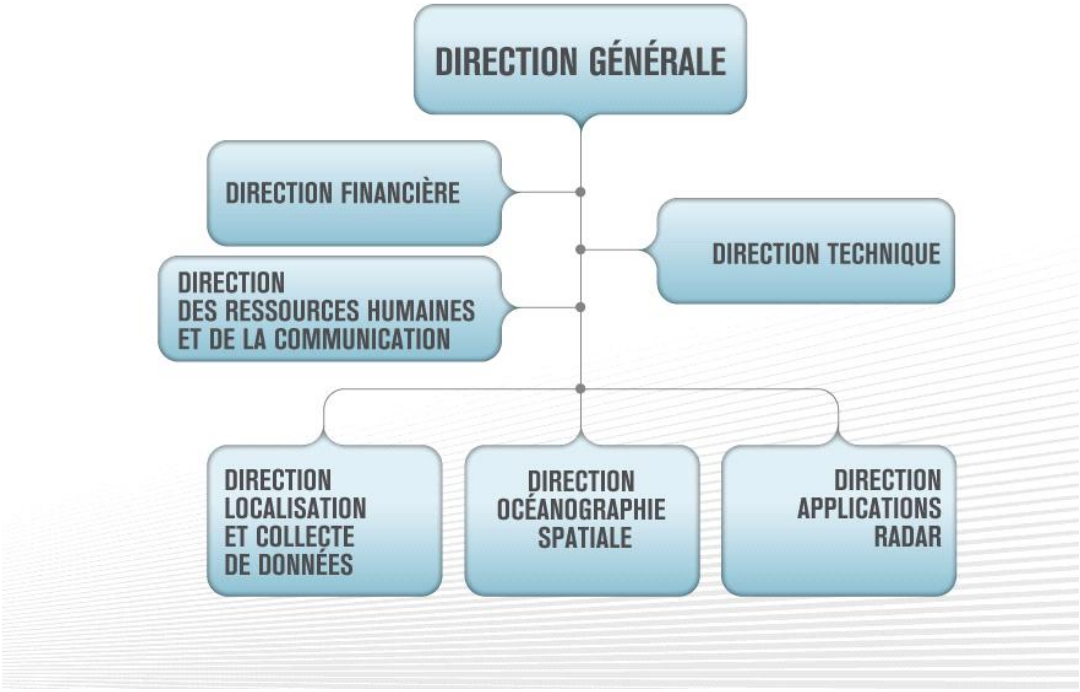

Figure 4 : Organigramme général de CLS

La DCL développe les activités technico-commerciales. On retrouve l'aspect commercial des trois pôles stratégiques de CLS à savoir, la surveillance environnementale, la gestion des ressources marines et la sécurité maritime.

La DOS développe des activités scientifiques d'études océanographiques. Entre autre, le traitement de la télémesure scientifique des satellites, la géodésie spatiale et la modélisation des écosystèmes marins.

En ce qui concerne la DAR, c'est une direction qui est principalement implantée à Brest. Elle regroupe des activités fondées sur l'acquisition, le traitement et l'interprétation de l'imagerie radar satellitaire haute résolution.

La DT, quant à elle, est chargée de la conception, du développement, de la maintenance des systèmes mais également de leur exploitation (assurer le maintien en condition opérationnelle des applications 24h/24, 7j/7). Pour y parvenir, cette direction est divisée en 4 divisions, composées elles-mêmes de départements. La figure 5 représente l'organigramme de cette direction :

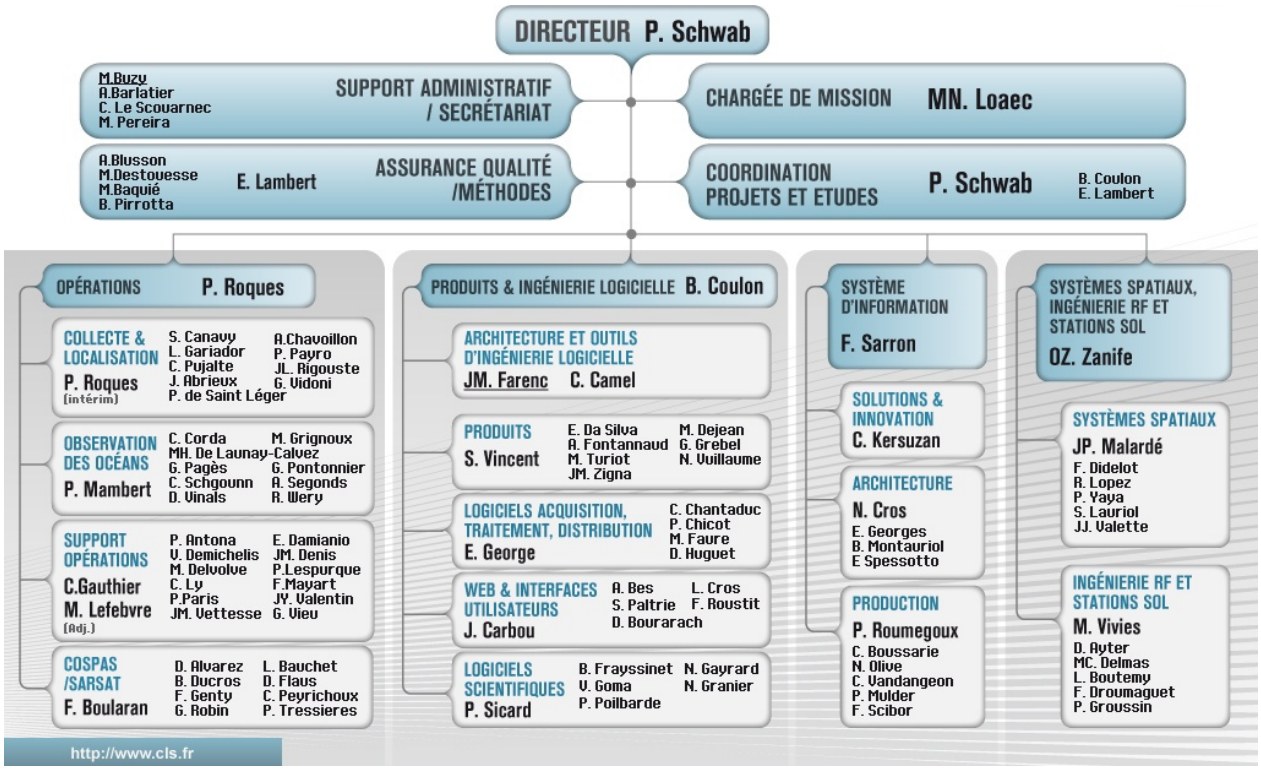

Figure 5 : Organigramme de la DT

I.3.2 La Division Système d'Information

Le rôle de la Division Système d'Information (DSI) est de définir, concevoir, et mettre en œuvre le système d'information à CLS, à travers trois départements :

- L'équipe « solutions et innovation » qui étudie les solutions innovantes qui pourraient être utilisées dans le futur par CLS.
- L'équipe « architecture » qui définit et met en place les infrastructures du système d'information.
- L'équipe « production » qui maintient et assure le bon fonctionnement du système d'information.

J'occupe actuellement le poste d'Ingénieur Système dans le département Production. Ma fonction dans l'équipe peut être décrite par plusieurs rôles : le principal est l'exploitation de l'infrastructure virtuelle, c'est-à-dire de mettre en œuvre les moyens garantissant une disponibilité maximale de celle-ci. A ce titre, j'effectue une analyse des anomalies, assure un suivi qualité, gère les évolutions et également les mises en production. Ma seconde fonction est l'exploitation de

l'outil de supervision de notre système d'information. Enfin, chaque membre de l'équipe est le référent système d'une ou plusieurs applications. Pour ma part ce sont les applications Iridium et Catsat.

C'est dans ce contexte que j'ai réalisé mon mémoire.

I.4 Le Système d'Information

I.4.1 Généralité

Pour héberger ces applications et rendre le service opérationnel à nos clients, nous disposons de plusieurs centres de données, appelés « Datacenter » (DC). A CLS-Toulouse, nous disposons de deux DC :

- Deux salles serveurs dans les locaux de CLS-Toulouse
- Une salle Disaster Recovery au CNES interconnectée au DC de CLS, via une liaison fibre optique dédiée

Nous disposons également de deux DC en dehors des locaux de CLS-Toulouse :

- Un DC à CLS America
- Un DC à CLS Brest

Cette infrastructure est nécessaire afin de répondre aux contrats de services en place avec nos clients pour la majorité de nos applications. Pour les applications les plus critiques, le taux de disponibilité de l'application doit être supérieur ou égal à 99,5%, ce qui correspond à moins de 4 heures d'arrêt autorisé par mois. De ce fait, les ressources informatiques sont importantes pour réaliser les activités de CLS. Nous décomptons environ 500 serveurs, physiques et virtuels confondus, de production et de développement pour environ 250 To de données stockées, fonctionnant sous les systèmes d'exploitation Linux, OpenVMS, Solaris, Tru64 et Windows.

I.4.2 La virtualisation

I.4.2.1 Définition

Afin d'introduire la notion de virtualisation, voici quelques définitions de la virtualisation et d'hyperviseur, issues du CNRTL[2] et de Wikipédia :

« virtuel : se dit des éléments (terminaux, mémoire, …) d'un système informatique considérés comme ayant des propriétés différentes de leurs caractéristiques physiques »

« Techniques matérielles et/ou logicielles qui permettent de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines physiques distinctes. »

« En informatique, un hyperviseur est une plate-forme de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps. »

I.4.2.2 Historique

Les premières solutions de virtualisation sont historiquement apparues au sein des systèmes mainframes de chez IBM à la fin des années soixante. Ce n'est que plus tard, dans les années quatre-vingt-dix, que la virtualisation s'est développée avec la progression des performances des processeurs et l'apparition pour le grand public d'émulateurs de consoles de jeux sur les architectures de type PC. A partir des années deux mille, les premières solutions sur ces architectures font leur apparition, permettant la virtualisation de système d'exploitation. Cet attrait du grand public et des entreprises pour la virtualisation fait que les deux acteurs majeurs de processeurs ont ajouté de nouvelles instructions visant à améliorer les performances des solutions de virtualisation (VT-x chez Intel et AMD-V chez AMD). Dans ce sens, les acteurs principaux de la virtualisation continuent de faire progresser la virtualisation, offrant des solutions toujours plus performantes.

---

[2] Centre National de Ressources Textuelles et Lexicales, portail de ressources linguistiques informatisées du CNRS

I.4.2.3 Les solutions techniques

Nous allons aborder les principales techniques de virtualisation :

Isolateur : Un isolateur (ou virtualisation par cloisonnement) est un logiciel permettant d'isoler l'exécution des applications dans des contextes ou bien zones d'exécution. L'isolateur permet ainsi de faire tourner plusieurs fois la même application dans un mode multi-instance (plusieurs instances d'exécution) même si elle n'était pas conçue pour cela. Cette solution est très performante mais les environnements virtualisés ne sont pas complètement isolés, on ne peut donc pas vraiment parler de virtualisation de systèmes d'exploitation. En effet, le matériel et le système d'exploitation sont les mêmes pour tout le monde. Ce type de virtualisation est essentiellement utilisé sur les systèmes Unix, Linux et BSD.

Exemples : Linux-vServer, chroot

Hyperviseur de type 2 : Un hyperviseur de type 2 est un logiciel qui s'exécute sur le système d'exploitation hôte et permet de lancer un ou plusieurs systèmes d'exploitation invités (machines virtuelles). La figure 6 représente ce fonctionnement :

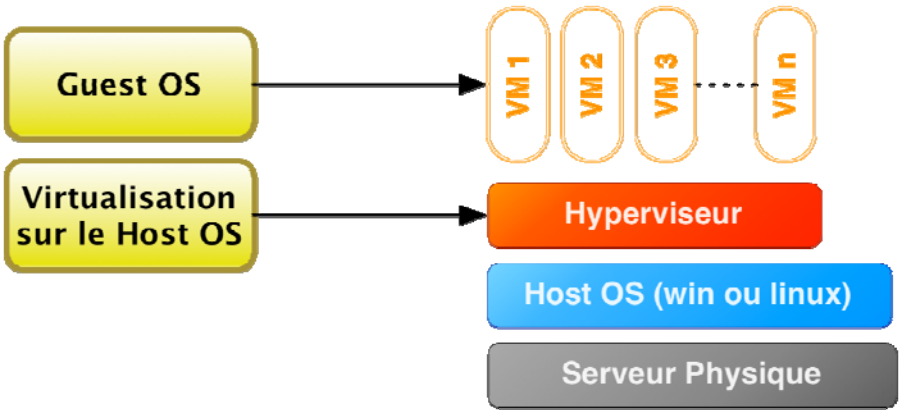

Figure 6 : Hyperviseur de type 2

L'hyperviseur installé sur le système d'exploitation se charge de créer un environnement complet simulant du faux matériel pour les systèmes invités, ces derniers croient dialoguer directement avec le matériel réel. On appelle également cette solution une « virtualisation complète ». Cette solution permet de faire

cohabiter plusieurs systèmes d'exploitation hétérogènes sur une même machine grâce à une isolation complète, mais cette isolation a un coût en performance. Ce type de virtualisation est toutefois limité aux systèmes d'exploitations prévus sur la même architecture matérielle que le processeur physique du serveur hôte. C'est ce qui le différencie d'un émulateur[3], qui va jusqu'à simuler le microprocesseur, et donc d'avoir la possibilité d'héberger une architecture matérielle différente de celle du CPU hôte mais les performances sont considérablement réduites par rapport à la virtualisation.

Exemples : Microsoft VirtualPC, Oracle VM VirtualBox, VMware Player

Hyperviseur de type 1 : Cette technique de virtualisation « native », aussi appelée « Bare-Metal », est une solution qui s'exécute directement sur la plate-forme matérielle. La figure 7 représente ce fonctionnement :

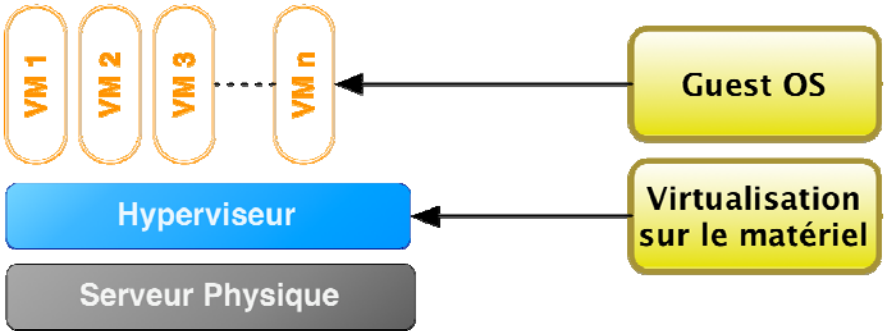

Figure 7 : Hyperviseur de type 1

C'est un noyau système très léger et optimisé pour gérer des systèmes d'exploitation invités à l'architecture matérielle sous-jacente. Les systèmes invités ont conscience de s'exécuter dans un environnement virtualisé : ils sont capables d'interagir avec l'hyperviseur. Actuellement, il s'agit de la méthode de virtualisation d'infrastructure la plus performante.

Cette solution, appelée également paravirtualisation, fournit un environnement virtuel (interface et ressources) censé représenter une architecture réelle. Ainsi,

[3] L'émulation sert à utiliser des logiciels ou OS prévus sur un autre type de matériel.
Toutes les instructions, qui sont à la base incompatibles avec le matériel, sont transformées à la volée pour pouvoir être effectuées.

un système d'exploitation développé pour une architecture particulière, s'exécute de façon native, sans aucune modification, dans une machine virtuelle qui virtualise complètement cette architecture. Concrètement, les machines virtuelles partagent les ressources physiques du serveur (disques, mémoires, CPU, carte réseaux, …).

Exemples : VMWare ESXi, Citrix XenServer, Microsoft Hyper-V

I.4.2.4 La solution VMWare

I.4.2.4.1 Généralités

Je vais vous présenter la solution de virtualisation qui sera traitée dans ce mémoire car c'est celle qui est déployée à CLS : VMWare.

VMWare est une société américaine créée en 1998, aujourd'hui filiale d'EMC, qui conçoit des solutions de virtualisation d'infrastructures informatiques d'entreprise. VMware s'impose rapidement comme le plus grand fournisseur de solutions sur ce marché avec un chiffre d'affaires de 4,61 milliards de dollars en 2012 et plus de 480 000 clients et 55 000 partenaires.

VMWare décline son offre d'hyperviseur type 1 en deux gammes « ESX et ESXi ». Un serveur ESX/ESXi est un système d'exploitation optimisé pour gérer les accès des machines virtuelles à l'architecture matérielle. Son rôle est d'héberger des machines virtuelles et de permettre l'exécution de celles-ci. La dernière version majeure est la version 5.5 (septembre 2013) qui est présente uniquement en ESXi, la gamme ESX ayant été abandonnée depuis la version 4.1.

La différence entre les deux gammes est la présence d'un système d'exploitation Linux (appelé Service Console) permettant la gestion de l'ESX, qui a disparu dans la gamme ESXi afin de sécuriser et d'optimiser les ressources mais également l'espace utilisé par celui-ci (le système peut maintenant démarrer sur une clé USB). La console affiche un minimum d'opérations, s'approchant visuellement d'un BIOS (cf ci-dessous la figure 8).

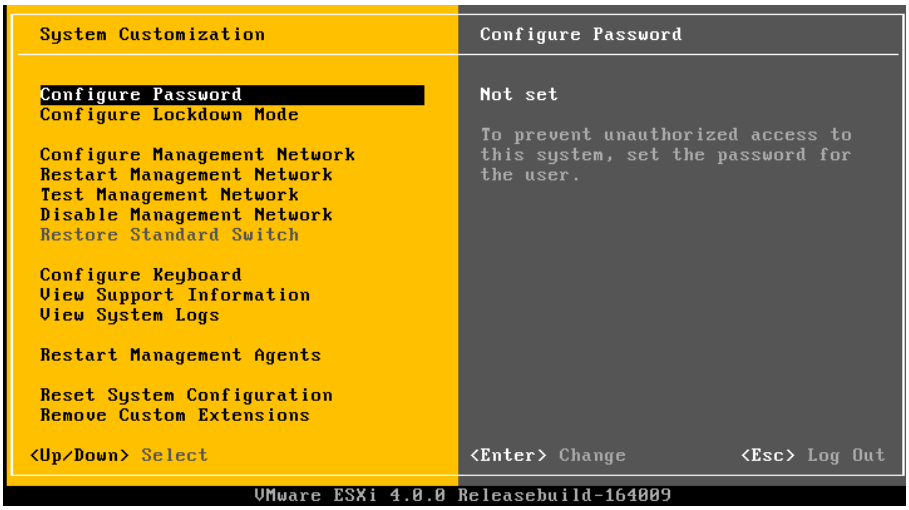

Figure 8 : Interface ESXi

VMWare dispose de fonctionnalités de gestion et d'administration qui ont simplifié l'administration des serveurs. Ces fonctionnalités, reprises à ce jour en majorité par les concurrents, ont fortement contribué au succès et à la démocratisation de la virtualisation dans les entreprises :

- **vCenter** : Console de gestion centralisée. Elle permet de gérer l'ensemble des serveurs VMWare ESXi et machines virtuelles depuis une interface graphique, et de ce fait, d'accéder aux consoles des machines virtuelles.

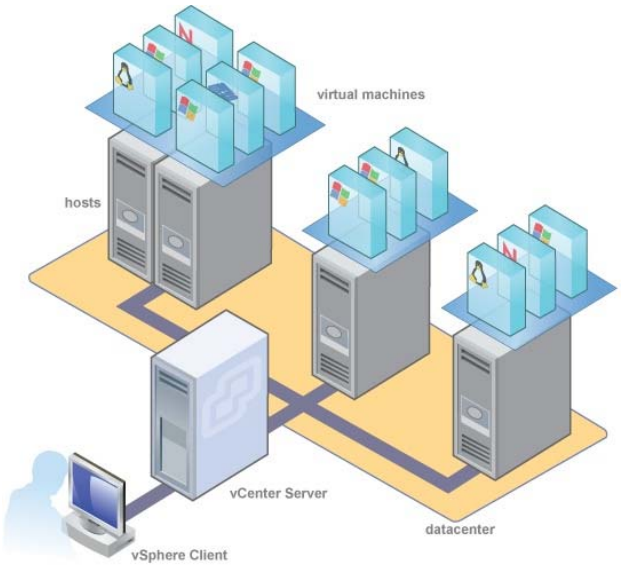

Figure 9 : vCenter

- **VMotion** : Cette option permet de migrer à chaud, donc sans interruption de service, une machine virtuelle d'un serveur ESX vers un autre. Cette opération est possible si les serveurs hôtes utilisent des microprocesseurs compatibles et que l'espace de stockage est partagé entre les serveurs. Techniquement, VMotion déplace le contenu de la mémoire d'un serveur hôte vers un autre.

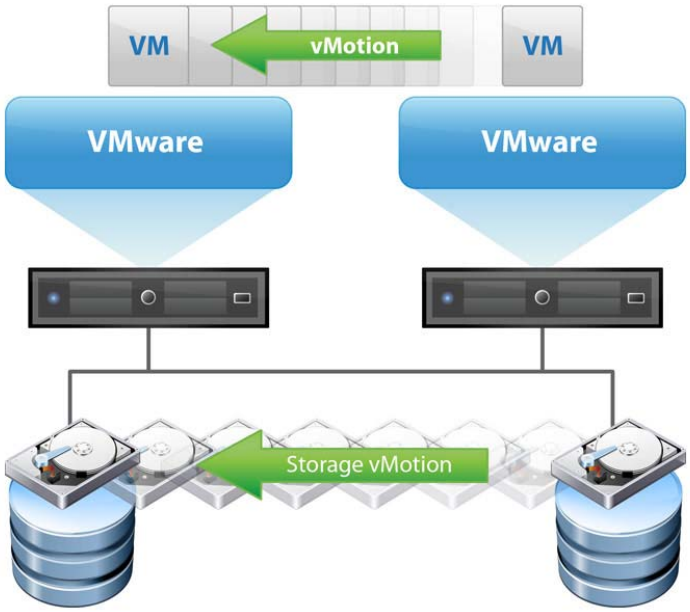

Figure 10 : vmotion et storage vmotion

- **Storage VMotion** : Concept identique à VMotion mais pour le stockage. Option très utile pour des opérations de réorganisation du stockage, ou la migration des VM vers de nouveaux espaces.
- **High Availability (HA)** : Lorsqu'un ESX tombe en panne, l'ensemble des machines virtuelles associées et faisant partie du cluster HA vont automatiquement et immédiatement redémarrer sur un autre ESX membre du cluster. Ceci permet de réduire le temps d'indisponibilité des applications.

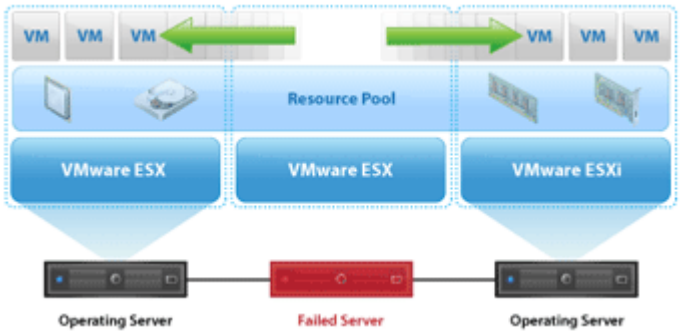

Figure 11 : High Availability

- **Distributed Resource Scheduler (DRS) / Distributed Power Management (DPM)** : Ces deux composants permettent de proposer une bonne répartition des machines virtuelles en fonction de l'utilisation des ressources de la plateforme. DRS, en s'appuyant sur VMotion, va jouer le rôle de répartiteur des machines virtuelles entre ESX en analysant l'utilisation mémoire et processeur des machines virtuelles. Pour sa part, DPM permet de réduire la consommation électrique de la plateforme, en consolidant les machines virtuelles sur les ESX. DPM est couplé avec DRS pour éviter de surcharger les ESX.
- **Fault Tolerance (FT)** : La tolérance de panne apporte de la très haute disponibilité. Même en cas d'arrêt brutal d'un ESX, il n'y a pas d'arrêt des machines virtuelles, donc pas d'interruption de service. FT crée une copie conforme d'une machine donnée. Tout ce qui se passe sur cette machine, appelée machine virtuelle primaire, est copié en miroir dans une machine virtuelle secondaire. En revanche, les contraintes sont nombreuses : consommateur en ressources et techniquement à ce jour un seul vCPU peut être attribué par machine virtuelle.

I.4.2.4.2 Gestion des ressources

Nous venons de voir les fonctionnalités qu'offre la virtualisation, nous allons maintenant nous intéresser à la gestion des ressources. Ces concepts sont à prendre en compte pour une meilleure compréhension du fonctionnement de la virtualisation. Un hyperviseur permet de virtualiser les principaux composants d'un serveur physique afin de pouvoir partager ces ressources. Quatre composants

majeurs d'un serveur sont concernés : le processeur, la mémoire, le stockage et le réseau.

- Le processeur est le seul composant du serveur qui n'est pas masqué par la couche de virtualisation. La machine virtuelle voit donc le type et le modèle du processeur physique du serveur sur lequel il s'exécute.

  On définit sur des machines virtuelles un nombre de vCPU, l'hyperviseur s'occupe de répartir la charge sur les différents cœurs.

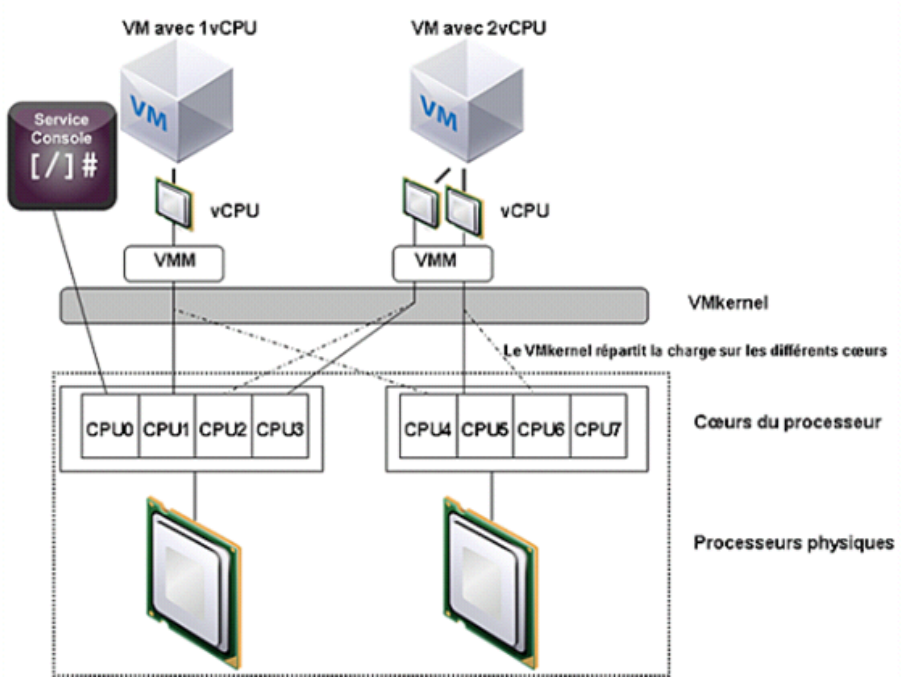

Figure 12 : Gestion des processeurs

- La mémoire est une ressource dont VMWare a pu tirer parti au mieux. En premier lieu, nous allons voir que la virtualisation permet «d'économiser » la consommation mémoire avec diverses technologies. De plus, la sur-allocation mémoire est possible : on peut avoir plus de mémoire dans les machines virtuelles qu'il n'y a de mémoire physique dans le serveur. Lorsque la sur-allocation a lieu, VMWare utilise différentes techniques pour récupérer de l'espace mémoire d'une ou plusieurs VM : le TPS, le ballooning et le swap.

  - **Le TPS (Transparent Page Sharing)** : les VM partageant la même mémoire physique, il est fort probable que certaines pages mémoires soient identiques. Le TPS supprime les pages mémoires

qui sont redondantes, et ce, de manière transparente (les VM n'ont pas conscience de partager une page mémoire avec d'autres VM). La figure 13 représente ce fonctionnement, les pages mémoires représentées en orange sont identiques, elles ne seront consommées qu'une fois sur la mémoire physique :

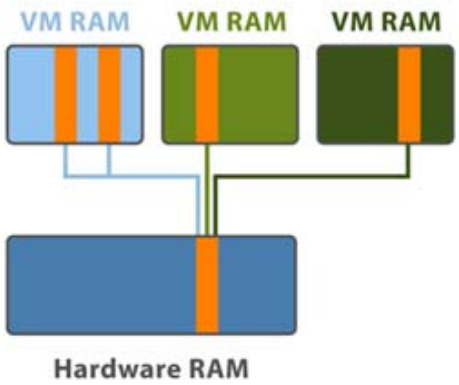

Figure 13 : TPS

- **Le Ballooning** est le fait de prendre de la mémoire disponible à une machine virtuelle pour la donner à une autre qui en a besoin. Pour cela, l'hyperviseur fait la différence entre la mémoire consommée par le système d'exploitation et la mémoire utilisée (active). Ensuite, le driver du balloon (vmmemctl) donne l'instruction de « gonfler » (comme un ballon) la mémoire ce qui oblige le système d'exploitation à libérer les pages mémoire non utiles et à les déplacer vers le disque. La figure 14 représente le fonctionnement du ballooning :

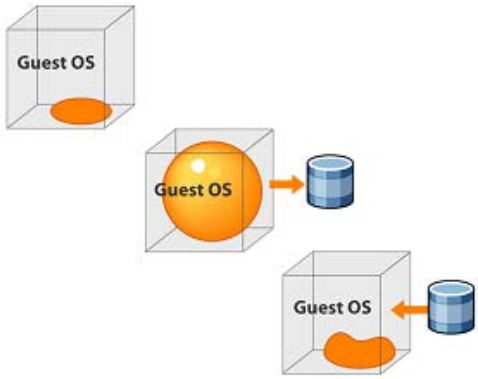

Figure 14 : Ballooning

- **Le Swap** : le concept est identique à celui appliqué sur un système linux. Un fichier de swap (vswp) de la taille de la mémoire de la VM va être créé. Cette technologie est employée par l'hyperviseur en dernier recours si le TPS et le ballooning ont été insuffisants.

- Le stockage : l'hyperviseur a accès aux disques sur serveur physique ou à un stockage partagé. L'espace de stockage est nommé « datastore » : c'est un volume formaté dans le format vmfs, propriétaire à VMWare. Cela permet d'avoir un modèle de stockage uniforme, pour y stocker les machines virtuelles.

  - La connexion des VM au réseau est basée sur des switchs virtuels nommés vSwitch. Concrètement, le vmkernel va jouer le rôle de répartir les composants physiques du réseau aux switchs virtuels.

I.4.2.5 La virtualisation au sein de CLS

Comme de nombreuses entreprises, CLS a fait le choix d'implanter la virtualisation dans son infrastructure. C'est à l'occasion du projet Disaster Recovery en 2006 qu'ont eu lieu les premiers déploiements. Outre les nouvelles fonctionnalités techniques, la virtualisation a permis de rationnaliser nos centres de données, tant en consommation énergétique qu'en espace physique et de consolider en optimisant l'utilisation de nos serveurs. En effet, l'acquisition de serveurs physiques a été exponentielle avec l'apparition de serveurs à faible hauteur avec des performances toujours améliorées grâce à l'arrivée des nouveaux processeurs, avec par exemple l'arrivée des multi-cœurs et le 64 bits. En général, avant l'utilisation de la virtualisation les serveurs ont été « sous-exploités ». Ce n'est pas un constat spécifique à CLS mais un constat global dans le monde informatique. D'après E.Maillet, 80% des serveurs avaient un taux d'utilisation moyen inférieur à 10%. La virtualisation est arrivée à point nommé et a permis de faire face à l'accroissement des demandes tout en répondant aux problèmes répandus dans les DSI en consolidant et rationalisant l'infrastructure.

Depuis, la virtualisation a permis la mise en place de nombreuses applications et l'augmentation significative du nombre de serveurs dans notre infrastructure, aujourd'hui majoritairement virtuels. De plus, la virtualisation a apporté de la

souplesse dans l'administration globale des serveurs. Les tâches et rôles ont été modifiés : les administrateurs sont moins confrontés à des problématiques matérielles mais plus systèmes.

En ce qui concerne notre infrastructure virtuelle, nous disposons de 30 serveurs dédiés à la production et au développement pour environ 350 machines virtuelles. Vous retrouverez sur la figure 15 le schéma d'architecture concernant l'environnement de production du site de Toulouse, découpé en deux grandes familles : les ESX hébergeant les moyens communs et ceux hébergeant les applications.

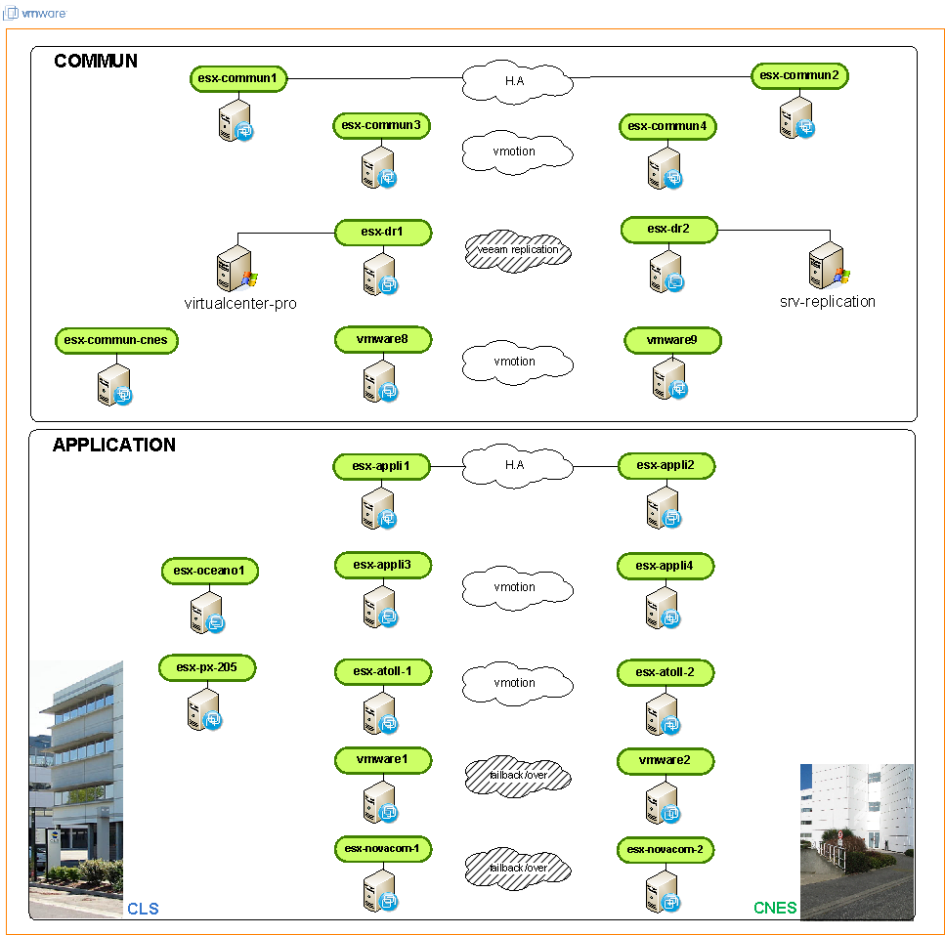

Figure 15 : Infrastructure virtuelle de production

I.4.3 La démarche ITIL

Les préconisations ITIL (Information Technology Infrastructure Library) ont été suivies pour l'évolution de notre système d'information. J'ai souhaité de ce fait utiliser cette démarche pour ce projet d'industrialisation.
Avant tout, il est nécessaire de vous présenter les concepts d'ITIL.

I.4.3.1 Généralités

Créé par le gouvernement britannique dans les années quatre-vingt comme une initiative d'amélioration de l'efficacité, la valeur ITIL a été rapidement identifiée par les entreprises et les organisations à travers le monde. Aujourd'hui, son rôle essentiel pour l'amélioration de la gestion des services est largement documenté. ITIL est constitué d'un ensemble d'ouvrages, basés sur les meilleures pratiques et dont les auteurs sont des experts internationaux publics et privés. ITIL en est à ce jour à la version 3 qui intègre cinq étapes dans son cycle de vie.

I.4.3.2 Les étapes du cycle de vie ITIL v3

La première étape de la démarche ITIL se transcrit par un premier ouvrage sur la stratégie des services (Service Strategy), c'est-à-dire la stratégie permettant de fournir des services créateurs de valeur. Autour, s'articulent les trois livres correspondant aux grandes phases de la vie d'un service : la conception du service (Service Design), la transition (Service Transition), et l'exploitation du service (Service Operation). Enfin, nous trouvons autour de ceux-ci la partie Amélioration continue du service (Continual Service Improvement). La figure 16 représente le cycle de vie des cinq chapitres d'ITIL v3.

Figure 16 : Cycle de vie ITIL v3

L'architecture des publications principales d'ITIL v3 est basée sur le cycle de vie du service où chaque livre est représenté par une « phase ». La définition selon ITIL d'un service est : « Un service représente un moyen de fournir une plus-value à un client, en produisant les résultats que ce dernier attend, sans qu'il n'ait à en porter lui-même les coûts et les risques spécifiques ».

I.4.3.3 ITIL au sein de CLS

Cette démarche a été utilisée pour l'amélioration de notre système d'information. Dans un premier temps, en 2006, notre solution de supervision, Eyes Of Network, a été mise en place en suivant les préconisations ITIL (gestion des alertes, gestion de la capacité, gestion de la disponibilité, etc.). Ensuite, en 2008, l'installation d'un outil de helpdesk, (gestion des événements, gestion des incidents, gestion des problèmes, gestion des changements, gestion des mises en production, etc.) et de gestion de parc (gestion des actifs et des configurations), ISILOG, est venue conforter cette démarche. Les trois grandes phases du cycle de vie ITIL v3 ont été ainsi parcourues. Conjointement, la Division des Opérations a travaillé plus spécifiquement sur la phase « Conception des Services » avec la mise en place des contrats de services formalisés au sein d'un SLA (Service Level Agreement) avec nos clients (internes et externes) pour la plupart de nos applications à ce jour.

II Problématique et besoins

Nous venons de voir dans quel contexte ce mémoire va être réalisé. Ce que nous allons voir maintenant est la problématique et les besoins exposés par le sujet du mémoire.

II.1 Problématique

Pour assurer les activités de CLS, des configurations informatiques (serveurs, logiciels, équipements de stockage et réseau) doivent être imaginées, validées puis mises en place à CLS ou chez nos clients. La virtualisation n'a pas modifié le comportement de l'équipe système qui s'occupe des installations et des configurations. De ce fait, toute la puissance qu'offre cette technologie n'est pas exploitée. Il est important de souligner que l'arrivée de la virtualisation dans notre infrastructure a bousculé l'évolution constante prévue, avec une forte croissance en termes de nombre de serveurs : en trois ans, nous avons triplé le nombre de serveurs applicatifs. Une des conséquences directes de cette forte hausse est la manière de procéder pour l'installation d'un serveur. Jusqu'à l'arrivée de la virtualisation, pour l'installation d'un serveur physique, l'équipe système avait divers moyens pour mettre en place les configurations :

- Installation du système d'exploitation via l'image disque (ISO) ou le CD-ROM pour ensuite travailler sur sa configuration à partir d'une check-list préétablie.
- Installation du système d'exploitation via l'image disque (ISO) ou le CD-ROM pour ensuite travailler sur sa configuration « à la carte ».
- par d'autres moyens de déploiement (par exemple la solution RDP avait été mise en place pour faciliter le déploiement d'applications).

Cette habitude a été dans un premier temps conservée pour l'environnement virtuel. Plusieurs voies alternatives sont apparues avec les nouvelles possibilités qu'offre la virtualisation sans qu'il n'y ait de règles précises mises en place, comme :

- le déploiement par l'action de cloner une machine virtuelle existante et d'en modifier ensuite la configuration.

- par la création d'une machine virtuelle via un modèle prédéfini antérieurement.

La figure 17 représente ces nouvelles alternatives :

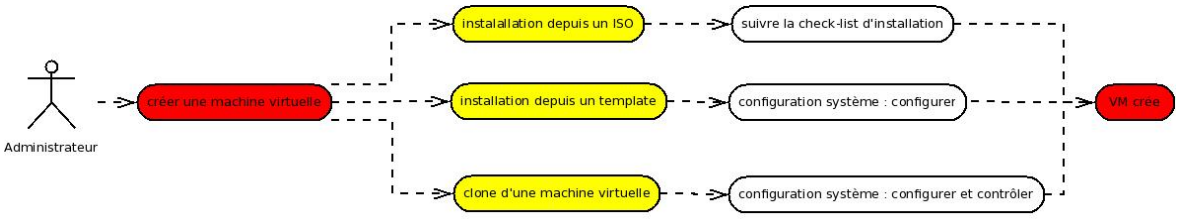

Figure 17 : Méthode de création d'une VM

Nous venons de rendre compte des différentes alternatives qui se sont naturellement mises en place dans l'équipe afin d'installer et de configurer des serveurs virtuels. Seulement, nous nous sommes rendus compte que ces solutions n'étaient pas optimales. La solution d'installation depuis une image disque a deux défauts : elle est très consommatrice en temps et surtout il existe un risque très fort d'avoir des installations non identiques si elle n'est pas associée à un outil de gestion des configurations. La solution de clonage d'une machine virtuelle a le mérite d'apporter un gain de temps assez important car elle permet d'outrepasser l'étape longue et fastidieuse d'installation du système d'exploitation mais elle est peu fiable. Le fait de cloner la machine virtuelle implique, par exemple, que les deux serveurs auront la même adresse IP, le même nom, etc. En outre, le problème majeur est que l'on conserve toute l'ancienne vie de la machine virtuelle source, en plus de la rigueur nécessaire dans la reconfiguration afin qu'un oubli dans un fichier de configuration quelconque ne se transforme pas en conflits sur le réseau. Bien sûr, des solutions de reconfiguration existent et permettent de simplifier cette étape. Par exemple, l'outil Sysprep pour les serveurs du constructeur Microsoft permet, au premier démarrage, d'effectuer les étapes de configuration les plus communes. Il existe des outils équivalents dans le monde Linux, comme par exemple kickstart chez RedHat. Ces solutions d'installations depuis un modèle associé à un outil de configuration semblent être les solutions les plus adaptées. Nous en avions mis quelques unes en place afin de faciliter nos déploiements, dans l'objectif d'économiser du temps pour les prochains déploiements. Seulement, sans y associer un processus de gestion de ces modèles, nous avons constaté que les bénéfices étaient presque nuls. En effet,

nos modèles ne sont pas cadrés car ce n'était qu'une installation de base mise de côté afin d'éviter les étapes rébarbatives d'installation. En fonctionnant de la sorte, nous ne sommes pas prêts à prendre en compte l'évolution des configurations qui doivent êtres adaptées au fil du temps et des évolutions. De plus, si on utilise les nouvelles possibilités qu'offre la virtualisation, la solution de clonage ou de modèle, il faut de toute manière avoir une machine virtuelle « propre », spécifique. A partir d'une installation nouvelle, sans qu'elle ait « vécue ». C'est la seule manière d'obtenir la garantie que les machines virtuelles commenceront leur cycle de vie au même niveau. Un autre constat, nous n'avons aucune gestion de nos systèmes d'exploitation au sein de notre système d'information. Nous essayons de déployer au maximum des distributions choisies par l'équipe système, sans regard sur les versions de celles-ci. Cette gestion, qui est inexistante aujourd'hui dans l'équipe, est une action que nous souhaitons mettre en œuvre.

Illustrons par un exemple concret l'intérêt d'industrialiser le déploiement de machines virtuelles. Nous avons rencontré des problèmes avec notre première application 100% virtuelle : Iridium. Un problème d'arrondis sur les valeurs dans l'insertion des positions des balises Iridium dans la base de données a sérieusement impacté notre production. Les clients recevaient des positions erronées. Après de longues investigations des différentes équipes (exploitation, intégration, de développement et système), celles-ci ont été incapables de trouver l'origine du problème. Etait-ce :

- lié au système d'exploitation ?
- lié à l'application ?
- ou lié à la virtualisation ?

CLS a fait appel à une société extérieure pour investiguer ce problème. Leur première demande a été de leur fournir la configuration système et applicative initiale, celle qui avait été validée en recette. Pour la version applicative, il n'y a pas eu de problème car un processus est en place pour la gestion des versions, ce qui n'est pas le cas pour le système. La conclusion est que nous avons été incapables de leur fournir le système tel qu'il a été livré deux ans auparavant, celui validé lors de la recette.

II.2 Besoins

II.2.1 Industrialiser

Le processus d'industrialisation qui est en cours de mise en place dans l'équipe système vise à maitriser ces installations et ces évolutions. Comme nous l'avons vu, l'objectif est, par exemple, d'être en mesure de reproduire une configuration à l'identique deux ans plus tard ou encore de garantir qu'une telle configuration n'a pas évolué et qu'elle est conforme à celle qui a été validée lors de la recette.

Commençons tout d'abord par faire la synthèse des enjeux de l'industrialisation. Ce terme est répandu dans le monde de l'informatique d'aujourd'hui, nous allons voir de quoi il s'agit. En voici la définition brute issue du CNRTL :

« Processus complexe qui permet d'appliquer à un secteur, à une branche de l'économie, des techniques et des procédés industriels qui apportent rationalisation et hausse de productivité »

Nous pouvons résumer l'industrialisation en deux idées directrices qui seront au cœur des enjeux de mon mémoire :

- Processus de fabrication
- Forte productivité de travail

En pratique, l'industrialisation dans le domaine informatique se traduit concrètement par la mise en place d'outils et de procédures afin de faciliter et de contrôler le déploiement, dans notre cas, de machines virtuelles. En effet, l'objectif est d'assurer la production des applications et de garantir qu'à tout moment, on peut reconstruire l'application de production à partir de ses sources. Cela se résume à disposer d'un réel processus de « fabrication » permettant de garantir la conformité et la qualité des services produits.

II.2.2 Homogénéiser

Comme déjà développé dans la problématique, l'objectif est d'homogénéiser notre manière de travailler, afin que chaque membre de l'équipe ait la même façon de procéder. Il est devenu indispensable de mettre en place une nomenclature

unique. L'équipe système attend la mise en place d'un socle technique, de versions validées par l'équipe pour les utilisateurs et les clients.

De plus, les installations se réalisent généralement avec la dernière version stable du système d'exploitation concerné, d'où l'apparition d'écarts de configuration dans notre parc. L'objectif est de protéger l'environnement de production et ses services en mettant en place des vérifications lors de l'implémentation des changements. Cette gestion des configurations vise à maitriser notre infrastructure. Concrètement, cela consisterait à empaqueter une application avec un système d'exploitation certifié, ce qui réduirait les problèmes de compatibilité entre une application et son système d'exploitation.

ITIL préconise que la gestion des configurations soit la base de référence afin de fournir la représentation la plus fidèle possible du système d'information et en particulier de son infrastructure en identifiant, contrôlant, et maintenant à jour et en vérifiant les versions de tous les éléments de configuration existants. La mise en place à CLS du couple d'outils Isilog pour la gestion de parc et Ocsinventory pour l'inventaire automatique a permis d'avoir une base de données unique des configurations (CMDB). Dans la même optique, nous avons besoin de gérer les profils et les versions de nos systèmes qui nous permettront de suivre et de maitriser les évolutions de notre système d'information.

II.2.3 Objectifs

Il est primordial de se fixer des objectifs réalistes et précis. Pour cela, nous devrons bâtir le projet afin qu'il corresponde à notre propre contrainte. Ainsi, il faut connaître les améliorations attendues par rapport aux faiblesses du Système d'Information actuel et à ses problématiques.

Que va apporter la nouvelle solution ? Il sera possible de mesurer les bénéfices à tous les niveaux et devront être clairement identifiés :

- Mise à disposition plus rapide et plus efficiente des mises en production.
- Garantie que les utilisateurs peuvent utiliser le service dès sa mise en place.
- Taux d'erreurs plus faible, simplification de l'exploitation au quotidien.

- Reconstruction possible d’un environnement identique à celui validé en recette.

Il conviendra de mettre en place un système simple à utiliser, convivial et rapide. L’objectif est que chaque membre de l’équipe, quel que soit son domaine de compétences, puisse facilement déployer des machines virtuelles rentrant dans le périmètre CLS défini par l’équipe.

III Etude préliminaire

Après avoir décrit la problématique et les besoins à couvrir, considérons maintenant la description des étapes préliminaires qui ont permis d'aboutir à une solution. Cette réalisation est décomposée en trois parties. Dans la première, nous allons effectuer un état des lieux de notre infrastructure. Ensuite, dans la deuxième, nous délimiterons le périmètre du projet. Enfin, dans la dernière partie, nous étudierons les préconisations ITIL pour la réussite du projet.

III.1 Etat des lieux

III.1.1 Parc serveurs virtuels

Avant d'entreprendre la réalisation du projet, il est indispensable d'effectuer un état des lieux de notre parc virtuel à CLS. Nous nous sommes limités aux machines virtuelles référencées dans le centre de données de CLS-Toulouse. Notre CMDB (Configuration Management Database) identifie environ 250 machines virtuelles. Suite à ces informations, la première analyse (comme le montre le tableau 1) révèle que l'on peut découper notre parc virtuel en deux grandes familles de systèmes d'exploitation : Linux et Windows.

Système d'exploitation	Nombre de VM
Linux Server	191
Windows Server	33

Tableau 1 : Répartition des OS

Nous avons fait le choix d'ajouter un degré de granularité, c'est à dire en spécifiant les distributions dans leurs versions majeures. L'objectif est de ressortir seulement les distributions les plus déployées. De ce fait, les distributions ayant moins de dix machines virtuelles ne sont pas représentées dans le tableau 2 ci-dessous. Nous n'ajoutons pas de degré supplémentaire jugeant l'intérêt mineur. Par exemple, on aurait pu ajouter la répartition des updates : la distribution Redhat Entreprise Linux AS 4 regroupe toutes les updates existantes, c'est-à-dire de 1 à 8.

Distribution | Version majeure *	Nombre de VM
RedHat Enterprise Linux AS 4	73
Centos 5.x	59
RedHat Enterprise Linux 5.x	25
Windows 2003 Server	22
RedHat Enterprise Linux AS 3	13
Debian 5.x	11

Tableau 2 : Répartitions des distributions

** Les familles (VM <10) ne sont pas représentées*

Cet audit préalable fait ressortir que notre parc est assez hétérogène avec six versions majeures différentes déployées, majoritairement basées sur des systèmes Linux qui représentent environ deux cents machines virtuelles, soit 80% du parc.

III.1.2 Définitions de classes

L'état des lieux a permis de donner la composition de notre infrastructure virtuelle actuelle. La deuxième étape est de délimiter le périmètre du projet. Au cours de mes travaux, j'ai défini trois classes de machines virtuelles, décrites ci-dessous.

III.1.2.1 La classe système

J'ai nommé la première classe de machines virtuelles «système». En effet, cette famille est la base d'une configuration. Elle inclut le système d'exploitation supporté par l'équipe système de CLS, configuré selon nos préconisations, avec en plus, les outils nécessaires à une bonne exploitation lors de sa livraison. Les versions des systèmes et l'interopérabilité avec notre infrastructure seraient de ce fait validées. Par exemple, l'installation d'une machine virtuelle reposant sur une distribution Linux mais optimisée : les paquets concernant la suite bureautique ou le serveur graphique ne seront pas installés par défaut, car ils sont inutiles au bon fonctionnement d'un serveur web. De plus, les outils utilisés pour l'exploitation quotidienne seraient déployés : le client d'inventaire, de supervision, de

sauvegarde, etc. Enfin, les fichiers de configuration système seraient prédéfinis (configuration réseau, ...).

D'après le référentiel ITIL, ce mode d'implémentation peut se faire après un test exhaustif des éléments qui le composent mais permet de garantir que tous les éléments sont à niveau et validés.

III.1.2.2 La classe applicative

A ce jour, les équipes de développement créent une application sur un système préinstallé que l'équipe système leur fournit selon les besoins exprimés. Ensuite, elles en modifient généralement la configuration système avec le concours des administrateurs systèmes pour l'adapter aux besoins de leurs applications.

On peut imaginer que les développeurs utilisent une machine virtuelle de type « système » pour effectuer leurs installations et validations. Les équipes de développement et d'intégration effectueraient les modifications nécessaires. Par exemple, en ajoutant des paquets indispensables au bon fonctionnement de leur application. Lors de la recette, on certifiera que l'application fonctionne avec la configuration de livraison : configuration système avec par exemple version java 1.6.3. Un autre bénéfice est que nous pourrions également mettre rapidement à disposition d'autres machines virtuelles d'une application donnée « prêtes à l'emploi ».

III.1.2.3 La classe service

Il existe également des machines virtuelles ayant un rôle précis, généralement déployées à la chaîne. En effet, cela peut être intéressant pour les serveurs communs : serveurs web, serveurs d'annuaire centralisé, serveurs DNS[4], etc. Cette solution est celle que l'on retrouve majoritairement chez les éditeurs. Par exemple, la solution de serveurs collaboratifs « Zimbra » met à disposition un modèle qui est pré-empaqueté dans un système d'exploitation validé. Les utilisateurs de la solution n'ont plus à se préoccuper de l'intégration de la solution

[4] Domain Name Server : serveur de nom dont le rôle est de traduire un nom d'adresse par une adresse IP

sur leurs systèmes d'exploitation (compatibilité, dépendances, évolution, obsolescence, etc.).

III.2 ITIL et le déploiement

En me documentant sur les préconisations ITIL, j'ai pu relever que la phase principale concernant mon mémoire est la « Transition des services ». Je vais dans un premier temps en définir son rôle :

« La Transition des services » : cette phase est l'interface centrale entre la conception des services et l'exploitation des services. Son objectif est de définir et contrôler les composants des services de l'infrastructure afin d'assurer la précision et la fiabilité de cette information. On va s'assurer que tous les composants des services ou des produits sont identifiés, maintenus et rattachés à une configuration de référence et que tous les changements portant sur ces composants seront gérés.

Trois processus de cette phase vont être primordiaux pour le bon déroulement de notre projet :

- Gestion des changements : l'objectif est de s'assurer que les changements sont enregistrés, évalués, autorisés, priorisés, planifiés, testés, réalisés, documentés et vérifiés.
- Gestion des configurations et des actifs des services : l'objectif est de fournir et maintenir un modèle logique de l'infrastructure informatique, des services informatiques qui y sont liés et des différents composants informatiques (physiques, logiques, etc.).
- Gestion des mises en productions et des déploiements : l'objectif est de garantir le déploiement dans l'environnement de production.

La notion de configuration de référence fait son apparition et apporte entre autres :

➔ Une meilleure connaissance de la production des services

➔ Une meilleure analyse des impacts des changements

➔ Une aide à la résolution des incidents et problèmes

➔ Une aide à l'industrialisation et à la standardisation des infrastructures

III.3 Solutions envisagées

Nous avons identifié les besoins auxquels doit répondre la solution choisie. Il s'agit maintenant de présenter les différentes solutions disponibles, en révélant les avantages et inconvénients respectifs.

III.3.1 Outils de gestion des configurations

La première solution est d'utiliser un outil de gestion des configurations de serveurs. Il s'agit d'une solution où, depuis un serveur maître, nous pouvons gérer de manière centralisée le déploiement des paquets du système d'exploitation sur les serveurs esclaves. Ce genre de solution correspond à ce qui avait été mis en place quelques années auparavant à CLS : la solution RDP (Rapid Deployment software) de la société HP, aujourd'hui abandonnée. Il existe également des solutions comme PXEBoot+TFTP que l'on a utilisé pour notre ferme de calcul, ou lié à kickstart par exemple pour Linux, solution que l'on a utilisée pour nos serveurs de base de données. Depuis, des solutions plus abouties sont apparues, comme par exemple Puppet, Chef ou encore FAI (Fully Automatic Installation). L'avantage de ces solutions est qu'elles sont également fonctionnelles pour un environnement non virtuel. Elles permettent également des déploiements plus poussés comme par exemple le BIOS d'un serveur ou les firmwares des composants. L'inconvénient est qu'elles sont complexes à prendre en main, demandent une certaine période d'adaptation et qu'elles ne sont, pour l'instant, fonctionnelles que sur des systèmes d'exploitation Linux.

III.3.2 Modèles de machine virtuelle

Les modèles de machine virtuelle que nous appelons également « template » est également un moyen intéressant de répondre à nos besoins. En effet, la virtualisation a apporté des fonctionnalités intéressantes : à partir d'un modèle nous pouvons déployer des machines virtuelles assez facilement. Deux possibilités distinctes existent pour la création de templates :

- La première possibilité est l'utilisation de solutions permettant de générer des machines virtuelles à partir de configurations préétablies. Par exemple, VMware Studio ou encore Suse Studio sont les solutions les plus abouties à ce jour. Il s'agit de solution de gestion de configuration comme citée ci-dessus, mais orientée virtualisation. L'avantage de cette possibilité est que tout est décrit dans un profil mais il est aussi un inconvénient, car la configuration s'effectue à « l'aveugle », c'est-à-dire qu'à partir d'un système d'exploitation vierge on doit décrire les paquets supplémentaires ou les scripts post-installation. Cette possibilité manque de souplesse dans l'administration et dans les possibilités.
- La seconde possibilité est une création personnalisée avec une conversion de la machine en modèle une fois la machine virtuelle validée. L'avantage de cette possibilité est que le résultat de notre configuration est visible directement. L'inconvénient est que l'on perd l'obligation de description et donc de traçabilité. Cette solution peut être très intéressante si elle est associée à des procédures rigoureuses. De plus, un modèle est consommateur en termes d'espace disque rendant cette solution contraignante. Heureusement, les éditeurs majeurs dans le monde de la virtualisation ont bien compris l'enjeu de ce type de solution et ont donc décidé de mettre à disposition un modèle standard.

IV Solution choisie

La solution d'utiliser un outil de gestion des configurations a été écartée, bien que le potentiel de ces solutions soit intéressant mais le maintien de ces solutions est complexe. En effet, ce type de solution est lourd à mettre en place et à gérer. A ce titre, la solution d'utiliser les modèles de machines virtuelle est apparue comme la plus viable par rapport à nos besoins. De plus, au cours de mes recherches, j'ai appris l'existence d'un format ouvert créé en 2007 par les acteurs majeurs de la virtualisation (VMWare, XenSource, ...) : le format OVF : Open Virtual machine Format. Ce format tend à s'imposer comme la norme pour les modèles de machines virtuelles.

IV.1 Le format OVF

Ce format est apparu et a été accepté en 2007 par le DMTF (Distributed Management Task Force). Les spécifications sont à ce jour définies dans la version 2.0.1, et plus récemment ratifiées par l'organisme ANSI (American National Standards Institute). Il est à noter que la version prise en compte dans cette étude est la version 1.1.0, la version 2.0.1 datant de 2013. Le besoin d'un standard pour la distribution de machines virtuelles sur des plates-formes de virtualisation est né de la rapide adoption d'infrastructures virtuelles dans les entreprises.

Un OVF est un mécanisme de transport de modèle de machine virtuelle, concrètement une « image maître » de la version de l'application. Son rôle est de permettre la distribution de machines virtuelles en assurant l'interopérabilité des principales solutions de virtualisation du marché. Ce format est également nécessaire afin d'automatiser et sécuriser l'installation, la configuration et l'exécution de ces applications sur n'importe quelle plateforme de virtualisation.

Ce format est basé sur le standard XML (Extensible Markup Language), il décrit les caractéristiques d'une machine virtuelle, aussi bien techniques (format des disques durs, nombre de processeurs,...) qu'administratives (compatibilité, licence,...). Il peut également contenir une signature numérique pour assurer l'intégrité d'une machine virtuelle ainsi que des informations facilitant son déploiement dans un nouvel environnement.

IV.2 Intérêt du format OVF

Le format OVF est une solution attractive qui permet de mettre ensemble une application et un système d'exploitation sur lequel l'application est certifiée. C'est-à-dire qu'elle passerait de la phase de développement à celle de tests d'intégration puis de production sous la forme d'un ensemble pré-configuré, pré-assemblé, sans aucune dépendance extérieure. L'application peut être facilement transmise par un revendeur, le tout, à l'intérieur d'une machine virtuelle.

De plus, construire une application dans un environnement virtuel est plus simple et plus économique que de fournir une application dans un environnement matériel : l'application est préinstallée avec le système d'exploitation qu'elle utilise (réduisant d'autant les phases de tests de compatibilité entre l'application et le matériel, ainsi que le passage par les certifications). Peu importe que son utilisation soit interne ou externe, le format OVF permet de simplifier de manière importante le cycle de gestion à travers l'adoption d'un ensemble de procédés standardisés.

IV.3 Possibilité du format OVF

A l'heure actuelle, les applications ne contiennent généralement qu'une seule machine virtuelle, alors que les applications d'entreprise utilisent une architecture orientée service avec plusieurs tiers, où chaque tiers contient une ou plusieurs machines. Le modèle de machine virtuelle unique n'est donc pas suffisant pour distribuer un service multi-tiers. De ce fait, un OVF peut être composé d'une ou plusieurs machines virtuelles. Prenons par exemple le cas de la revente d'une solution hébergée sur un serveur. On pourrait imaginer que ce serveur accueille plusieurs machines virtuelles, composant le « service » fourni au client. Une machine virtuelle pour l'application, une pour la supervision de l'application, une pour la sauvegarde, … La figure 18 représente ce type d'OVF.

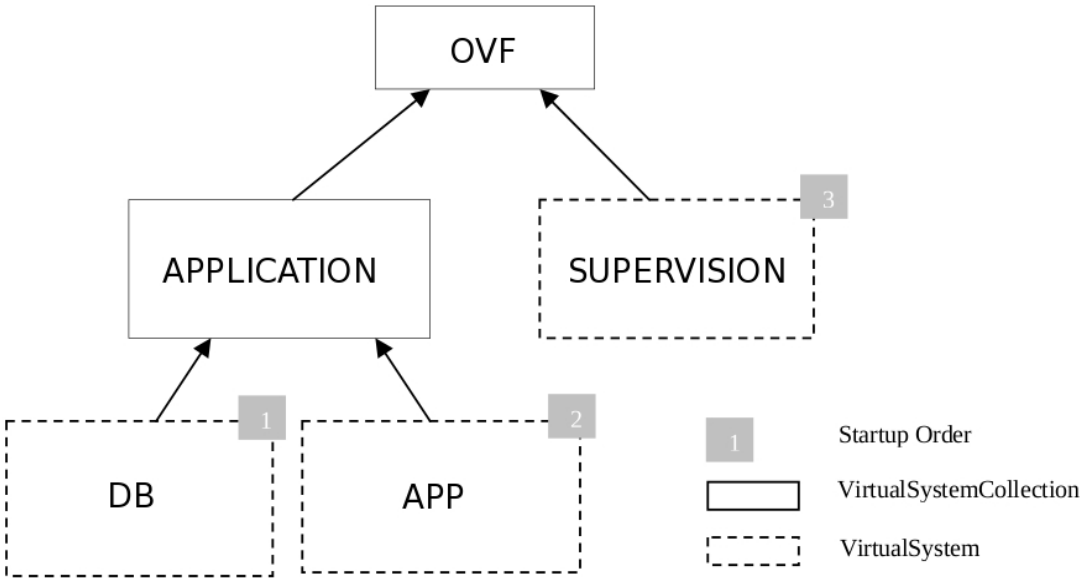

Figure 18 : Exemple OVF multi VM

Les machines virtuelles sont assemblées dans un format dédié à l'exécution avec leurs images disques et leur données de configuration, et ce, pour un hyperviseur donné. Ces formats d'exécution sont optimisés pour l'exécution, non pour la distribution. De plus, il faut prendre en compte que les applications nécessitent l'optimisation lors de la phase d'installation des parties réseaux et des autres caractéristiques spécifiques et physiques au client.

Pour une distribution de logiciels efficace, un certain nombre de caractéristiques supplémentaires deviennent critiques, comme la portabilité, l'indépendance de la plateforme, la vérification, la signature, la gestion des versions, et les termes de la licence d'utilisation.

La spécification OVF est une spécification neutre vis à vis de l'hyperviseur, elle se doit d'être performante, extensible aux futures évolutions des plateformes et ouverte pour en assurer son indépendance. Elle définit les principes d'assemblage et de distribution des applications virtuelles composées d'une ou plusieurs machines virtuelles.

OVF n'est cependant pas la solution miracle au problème de la portabilité des machines virtuelles, en particulier :

- OVF ne garantit pas que la solution cible sera capable de lire le format dans lequel est stocké le disque dur de la VM. Néanmoins, en pratique, les

différents formats de disque sont convertibles. Une telle garantie est envisagée pour une version future de la norme. Il sera nécessaire d'imposer un format ouvert pour le stockage du disque afin d'avoir une réelle interopérabilité entre toutes les solutions commerciales et Open Source, présentes et à venir.

- OVF ne garantit pas que l'application elle-même soit compatible d'un produit à l'autre. Par exemple, une machine virtuelle sur laquelle n'est installé que le pilote pour le matériel présenté par VMWare ne tournera jamais sur l'hyperviseur de Xen.

En résumé, OVF est un format facilitant l'échange et la distribution de machines virtuelles entre les différentes solutions commerciales, mais le seul respect de ce standard par un produit de virtualisation n'assure en rien que ce produit pourra lire n'importe quelle VM exportée sous ce format.

A ce jour, l'industrie de la virtualisation a majoritairement adopté ce format comme le montre le tableau 3. Cette adoption donne un élan très positif à ce projet. A noter qu'il ne manque qu'un acteur majeur : Microsoft.

Produit	Support OVF	Date
VMware	ESX 3.5	Décembre 2007
VirtualBox	2.2.0	Avril 2009
AbiCloud Cloud Computing Platform Appliance Manager	0.7.0	Juin 2009
Red Hat Enterprise Virtualization	2.2	Mars 2010
OpenNode Cloud Platform	1.1	Novembre 2010
Oracle VM	3.0	Aout 2011
Microsoft Hyper-V	N/A	N/A

Tableau 3 : OVF et distributeurs

IV.4 Paquetage OVF

IV.4.1 Structure

Un paquetage OVF est composé de plusieurs fichiers :

- Un fichier « OVF descriptor » avec l'extension .ovf
- Un ou plusieurs fichiers image disque (exemple .vmdk)
- Un fichier « OVF manifest » avec l'extension .mf (optionnel)

- Un fichier « OVF certificate » avec l'extension .cert (optionnel)
- Un ou plusieurs fichiers ressources (exemple image ISO)

Un des avantages d'un paquetage OVF est la compression des données qui permet d'archiver simplement des versions de modèles. A noter, la possibilité de distribuer dans un seul fichier utilisant le format TAR. Dans ce cas, l'extension est .ova mais il n'y a pas de gain d'espace supplémentaire, l'intérêt est que les fichiers qui composent l'OVF soient stockés dans un seul fichier.

Prenons l'exemple de la figure 19, fourni par le DMTF, où l'on dispose d'une comparaison en taille des fichiers et de stockage entre un modèle VMWare, un OVF et un OVA.

	VMware Format	OVF Format	OVA Format
Files	LinuxBasedAppliance.nvram LinuxBasedAppliance.vmdk LinuxBasedAppliance-s001.vmdk LinuxBasedAppliance-s002.vmdk LinuxBasedAppliance.vmsd LinuxBasedAppliance.vmx LinuxBasedAppliance.vmxf	LinuxBasedAppliance.ovf LinuxBasedAppliance-0.vmdk LinuxBasedAppliance-1.vmdk LinuxBasedAppliance-2.vmdk	LinuxBasedAppliance.ova
Total size	251MB using thin provisioning 4000MB using thick provisioning	132MB	132MB

Figure 19 : Comparaison des tailles des formats

Ceci est prometteur mais nous aurons l'occasion d'effectuer notre propre comparaison plus tard.

IV.4.2 Conformité (OVF Descriptor)

Les spécifications définissent trois niveaux de conformité des OVF descriptor, le niveau 1 étant le niveau le plus élevé :

- Conformité niveau 1 : L'OVF Descriptor utilise seulement les sections, éléments et attributs qui sont définis dans la spécification.

- Conformité niveau 2 : L'OVF Descriptor utilise des sections, éléments et attributs personnalisés qui ne sont pas définis dans la spécification et toutes les extensions sont facultatives.

- Conformité niveau 3 : L'OVF Descriptor utilise des sections, éléments et attributs personnalisés qui ne sont pas définis dans la spécification et au moins une extension est définie.

IV.4.3 Enveloppe (OVF Descriptor)

L'enveloppe est l'élément qui décrit l'ensemble des métadonnées pour la/les machines virtuelles ; il peut être vu comme la structure du paquetage OVF.

L'enveloppe contient un premier niveau de définition :

- Indication de version, défini par les XML URL (XML Namespace Prefixes)

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope vmw:buildId="build-260188" xmlns="http://schemas.dmtf.org/ovf/envelope/1"
xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1" xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData" xmlns:vmw="http://www.vmware.com/schema/ovf"
xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

- La liste des références (les fichiers associés) qui font partie du paquetage OVF.

```
  <References>
    <File ovf:href="template-centos5-64-12-ovf-disk1.vmdk" ovf:id="file1" ovf:size="1101370880"/>
  </References>
```

Chaque élément « File » peut être identifié sous la forme ovf :id.

ovf :href spécifie le fichier qui contient une URL mais qui est interprété comme le chemin relatif du descripteur OVF. La taille du fichier référencé peut être spécifiée en utilisant ovf :size (bytes). Il existe la possibilité de compresser le fichier en utilisant gzip, il faut pour cela préciser la compression ovf :compression.

- Content Element

La configuration d'une machine virtuelle dans un paquetage OVF est représentée par la balise « VirtualSystem » ou « VirtualSystemCollection ». « VirtualSystem » est utilisé dans le cas d'une VM unique, à contrario de « VirtualSystemCollection » utilisé dans le cas de multiple VM.

```
<VirtualSystem ovf:id="vm">
    <Info>A virtual machine</Info>
    <Name>template-centos55_64bits</Name>
```

```
  <SomeSection>
          <!—Additional section content →
 </SomeSection>
</VirtualSystem>
```

- Virtual Hardware Description

    - Virtual Hardware section

La description du matériel virtuel est base sur la classe CIM[5].

```
<VirtualHardwareSection>
    <Info>Virtual hardware requirements</Info>
    <System>
      <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
      <vssd:InstanceID>0</vssd:InstanceID>
      <vssd:VirtualSystemIdentifier>template-centos55_64bits</vssd:VirtualSystemIdentifier>
      <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
    </System>
    <Item>
      <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
      <rasd:Description>Number of Virtual CPUs</rasd:Description>
      <rasd:ElementName>1 virtual CPU(s)</rasd:ElementName>
      <rasd:InstanceID>1</rasd:InstanceID>
      <rasd:ResourceType>3</rasd:ResourceType>
      <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
    </Item>
    <Item>
      <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
      <rasd:Description>Memory Size</rasd:Description>
      <rasd:ElementName>256MB of memory</rasd:ElementName>
      <rasd:InstanceID>2</rasd:InstanceID>
      <rasd:ResourceType>4</rasd:ResourceType>
      <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
    </Item>
    <Item>
      <rasd:Address>0</rasd:Address>
      <rasd:Description>SCSI Controller</rasd:Description>
      <rasd:ElementName>scsiController0</rasd:ElementName>
      <rasd:InstanceID>3</rasd:InstanceID>
      <rasd:ResourceSubType>lsilogic</rasd:ResourceSubType>
      <rasd:ResourceType>6</rasd:ResourceType>
    </Item>
    <Item ovf:required="false">
      <rasd:AddressOnParent>0</rasd:AddressOnParent>
      <rasd:AutomaticAllocation>false</rasd:AutomaticAllocation>
      <rasd:Description>Floppy Drive</rasd:Description>
      <rasd:ElementName>floppy0</rasd:ElementName>
      <rasd:InstanceID>4</rasd:InstanceID>
      <rasd:ResourceType>14</rasd:ResourceType>
    </Item>
    <Item>
      <rasd:AddressOnParent>0</rasd:AddressOnParent>
      <rasd:ElementName>disk1</rasd:ElementName>
```

[5] Common Information Model (modèle de données unifié) est un standard qui définit comment des éléments administrés dans un environnement informatique peuvent être représentés sous la forme d'un ensemble d'objets cohérents et d'un ensemble de relations entre ces objets.

```
    <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
    <rasd:InstanceID>5</rasd:InstanceID>
    <rasd:Parent>3</rasd:Parent>
    <rasd:ResourceType>17</rasd:ResourceType>
   </Item>
   <Item>
    <rasd:AddressOnParent>2</rasd:AddressOnParent>
    <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
    <rasd:Connection>Reseau-bleu</rasd:Connection>
    <rasd:Description>E1000 ethernet adapter on &amp;quot;Reseau-bleu&amp;quot;</rasd:Description>
    <rasd:ElementName>ethernet0</rasd:ElementName>
    <rasd:InstanceID>6</rasd:InstanceID>
    <rasd:ResourceSubType>E1000</rasd:ResourceSubType>
    <rasd:ResourceType>10</rasd:ResourceType>
   </Item>
  </VirtualHardwareSection>
```

Nous allons examiner les différents types d'éléments qui composent la Virtual Hardware Section.

- Vssd :VirtualSystemType décrit le type de système virtuel. Par exemple, dans le cadre d'une machine virtuelle VMware de 4ème génération on utilisera « vmx-4 », pour Xen 3ème génération on utilisera « xen-3 ».

```
    <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
```

- Les caractéristiques du matériel virtuel (cpu, mem, cdrom,…) sont décrites dans une séquence d'éléments "Item". Un item, s'il est décrit, est obligatoire. L'option ovf :required= « false » peut être utilisée dans le cas d'un matériel optionnel.

```
   <Item ovf:required="false">
    <rasd:AddressOnParent>0</rasd:AddressOnParent>
    <rasd:AutomaticAllocation>false</rasd:AutomaticAllocation>
    <rasd:Description>Floppy Drive</rasd:Description>
    <rasd:ElementName>floppy0</rasd:ElementName>
    <rasd:InstanceID>4</rasd:InstanceID>
    <rasd:ResourceType>14</rasd:ResourceType>
   </Item>
```

Exemple de description de la mémoire :

```
   <Item>
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:Description>Memory Size</rasd:Description>
    <rasd:ElementName>256MB of memory</rasd:ElementName>
    <rasd:InstanceID>2</rasd:InstanceID>
    <rasd:ResourceType>4</rasd:ResourceType>
    <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
   </Item>
```

On peut également fournir des gammes avec l'option ovf :bound qui doit être utilisée dans un élément « item ». Une gamme est composée d'une valeur :

minimum (min), normal (normal) et maximum (max). Il est recommandé de démarrer par une valeur normale et d'ajuster la valeur selon le besoin.

- De métadonnées de base concernant les disques, le réseau :

DiskSection décrit les méta-informations à propos des disques virtuels dans le paquetage OVF. Les disques virtuels ainsi que leurs métadonnées sont décrits en dehors du matériel virtuel pour faciliter les échanges entre les VM et paquetage OVF.

```
<DiskSection>
   <Info>Virtual disk information</Info>
   <Disk ovf:capacity="10" ovf:capacityAllocationUnits="byte * 2^30" ovf:diskId="vmdisk1" ovf:fileRef="file1"
ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized"
ovf:populatedSize="2731409408"/>
 </DiskSection>
```

NetworkSection décrit les réseaux logiques utilisés dans le paquetage OVF.

```
<NetworkSection>
   <Info>The list of logical networks</Info>
   <Network ovf:name="Reseau-bleu">
     <Description>The Reseau-bleu network</Description>
   </Network>
 </NetworkSection>
```

IV.5 Outils pour créer des OVF

Le format OVF étant une technologie récente, peu d'outils existent à ce jour. VMWare a été un des premiers acteurs et surtout un acteur majeur dans la technologie OVF. Depuis l'adoption de ce format par l'industrie, VMWare fournit toutes ses applications virtuelles dans ce format (vma, vCenter, …).

Le DMTF a fait un constat : dans la pratique, les OVF sont crées de deux manières :

- En utilisant un éditeur texte ou un outil de création de XMl pour écrire le descripteur, puis d'assembler les fichiers et images requises.
- En exportant une machine virtuelle. Le fichier OVF « brut » peut être ensuite modifié pour devenir un nouvel OVF. (améliorer la portabilité, options de configuration supplémentaires, etc.)

Les éditeurs ont fourni plusieurs alternatives pour créer/générer des modèles au format OVF. Elles sont présentées dans les prochains paragraphes.

IV.5.1 VMWare Studio

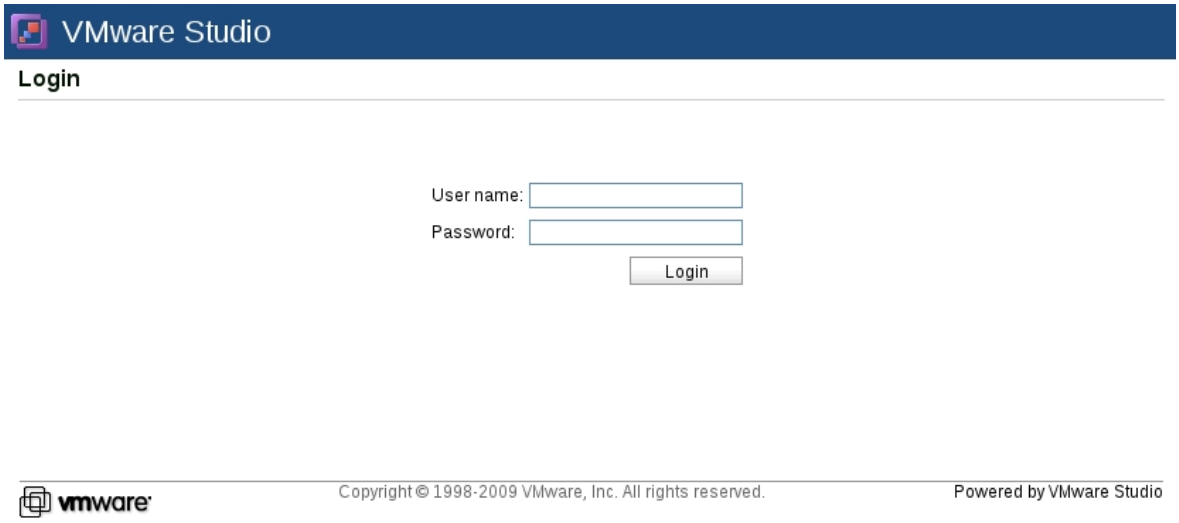

Figure 20 : Interface VMWare Studio

VMWARE Studio aide à créer, configurer et déployer des machines virtuelles. Cet outil permet de générer, en respectant le standard OVF, des vApps depuis une interface web. Entre autre, cet outil permet de :

- gérer les versions de profil des machines virtuelles créées ;
- afficher une licence (ou message d'aide) au démarrage de la vApp ;
- faciliter la configuration via une interface web, adaptable à nos besoins ;
- créer des utilisateurs et des groupes ;

L'outil VMware studio permet de paramétrer une machine virtuelle à partir de l'image disque ISO du DVD de la distribution retenue. On peut ajouter des paquets RPM, des archives « maison » (au format tar), faire exécuter des scripts au cours de l'installation, au premier démarrage ou même à chaque démarrage. Le principal avantage de cet outil est que tout peut se gérer à partir d'une interface web. L'inconvénient est que la personnalisation reste limitée car elle doit être effectuée à partir de l'image ISO, agrémentée d'ajouts de packages et de scripts de post-installation. Le principe est de générer la machine virtuelle depuis l'outil

avec toutes les personnalisations stockées dans un profil. Cette solution est un outil de gestion de configuration qui permet de générer des machines virtuelles au format OVF.

IV.5.2 VMWare OVF Tool

OVF Tool est un utilitaire en ligne de commande qui permet d'importer et d'exporter des machines virtuelles au format OVF. Dans un premier temps, il faut créer sa machine virtuelle maîtresse, comme pour un modèle, afin de la convertir dans le format OVF. L'avantage de cette solution est qu'elle est en ligne de commande ; elle peut donc être utile pour automatiser le déploiement de machines virtuelles par la création d'un script.

IV.5.3 Virtual Center

Récemment, depuis la version 4.x de vCenter, il est possible d'effectuer un certain nombre d'actions relatives à la gestion des OVF directement depuis l'outil de gestion centralisée :

- déployer un OVF
- générer à partir d'une machine virtuelle un modèle au format OVF depuis l'outil.

Cette solution est intégrée nativement dans la solution vCenter et propose une interface intuitive.

V Réalisation du projet

Nous venons de décrire les principes de fonctionnement du format OVF et ses intérêts. Nous allons maintenant détailler une utilisation de ce format dans le cadre de notre projet. Dans la première partie, nous allons définir le socle technique. Ensuite, nous détaillerons la mise en place de la maquette. Dans une troisième partie, nous décrirons l'organisation mise en place afin de gérer les modèles. Enfin, la dernière partie détaillera les mises en production des trois classes de modèles définies lors de l'étude préliminaire.

V.1 Socle technique

Avant de mettre en place la solution dite « technique », il faut au préalable définir une configuration de référence. En effet, ITIL préconise, dans la gestion des configurations, de définir une configuration de base qui sera ensuite validée dans le processus de gestion des mises en production et qui pourra être revue au cours du processus de la gestion des changements. Pour y parvenir, il faut que trois processus travaillent ensemble. Bien que chaque processus ITIL soit indépendant, dans le cas présent, leur mise en commun va apporter de l'aide au contrôle du système d'information. En effet, il va permettre de planifier les mises en production de nouvelles versions de système d'exploitation (majeures ou mineures) ou de patchs de sécurité. On va pouvoir tracer ces changements et n'autoriser que des versions testées et approuvées dans notre système d'information.

V.1.1 Configuration de base

Concrètement, cela se traduit par la mise en place d'un socle technique, c'est-à-dire la définition d'une machine virtuelle type :

- Un système d'exploitation validé et supporté par la suite par l'équipe système
- L'installation et la configuration de base des composants techniques nécessaires aux applications. Par exemple, nous n'installons pas un

système avec les options d'installation par défaut : l'interface graphique sur un serveur linux est souvent inutile et non optimisée pour notre besoin.

- Définir les outils internes nécessaires à une bonne exploitation (outil d'inventaire, de supervision, …)
- Définir les configurations types (configuration réseau, …)
- Mettre à disposition la « version » de celle-ci, identifiable et restituable par notre outil d'inventaire.

L'objectif est de s'assurer que seules les versions autorisées et testées des machines virtuelles sont mises à disposition, tout en répondant aux besoins définis par notre expérience et par les exploitants.

Comment ?

- Etudier notre infrastructure actuelle pour déterminer notre manière de travailler.
- Interviewer les membres de l'équipe pour identifier les outils utiles. Par exemple, un paquet perl doit il être par défaut dans le template ?
- Créer la machine type à partir des éléments recueillis.

V.1.2 Définition du socle technique

En reprenant l'état des lieux effectué dans l'étude préliminaire, nous avons dégagé plusieurs types de machines virtuelles. Il y a une volonté de la direction informatique d'industrialiser et de standardiser mais sans être prisonnier d'une solution. L'objectif est donc de définir un socle technique, un référentiel de systèmes d'exploitation supportés à CLS. Nous nous limiterons dans cette étude à la famille la plus représentée, Linux. Après une réunion avec les membres de l'équipe système, il a été décidé et acté que nous ne révolutionnerons pas notre architecture. Les deux distributions Linux supportées à CLS en tant que serveur sont les distributions CentOS et Debian que je vous présente brièvement ci-dessous :

- CentOS (The Community ENTerprise Operating System) est une distribution Linux principalement destinée aux serveurs. Tous ses paquets, à l'exception du logo, sont des paquets compilés à partir des sources de la distribution Red Hat Enterprise Linux (RHEL), éditée par la société Red Hat. Elle est donc quasiment identique et se veut 100% compatible d'un point de vue binaire. En effet, la RHEL ne peut être obtenue que par l'achat d'une licence d'utilisation auprès de Red Hat ou de ses revendeurs. Pour vulgariser, on peut donc considérer la CentOS comme une version « gratuite » de RHEL. Dans un contexte de réduction des coûts, CentOS est l'OS retenu au détriment de RHEL qui jusqu'à ce jour était le système d'exploitation historique de CLS.
- Debian est une distribution GNU/Linux non commerciale, lancée en 1993 par Ian Murdock avec le soutien de la Free Software Foundation ; elle a pour principal but de fournir un système d'exploitation composé uniquement de logiciels libres. Debian est une alternative fiable à CentOS.

Ces deux systèmes d'exploitation seront les « bases » des futurs et/ou potentiels modèles. Actuellement, CentOS est en version majeure 6.x et Debian en 6.x. Ce seront donc les deux premiers modèles au format OVF créés. Ces deux systèmes d'exploitation sont des systèmes supportés par la virtualisation VMware.

V.1.3 Gestion des versions

Les différentes distributions qui donneront lieu à un modèle OVF ont été choisies. Cependant, il faut également définir la granularité. Sur ce point, on a choisi de créer un modèle par version majeure de distribution. De plus, nous avons vu que les modèles systèmes sont la base des futurs modèles services et applicatifs. Pour être plus précis, la figure 21 représente l'exemple de ce que pourrait être le modèle système CentOS 5 : les versions du modèle et leurs interopérabilités avec les autres types de modèles.

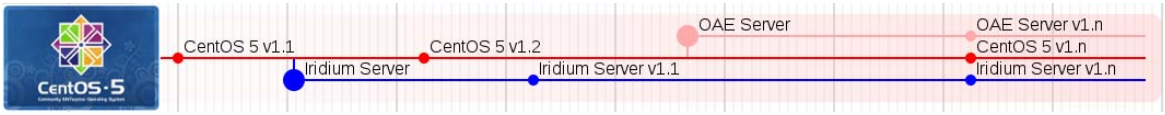

Figure 21 : Gestion des versions

Les distributions disposent d'une nomenclature respective pour fournir leur numéro de version. Pour CentOS, cette information est disponible dans le fichier « /etc/redhat-release », alors que pour la distribution Debian, on obtient la version de la release dans le fichier « /etc/debian_version ».

Par exemple, voici le contenu du fichier « /etc/redhat-release » d'un serveur CentOS :

```
CentOS release 5.5 (Final)
```

Nous avons fait le choix de suivre ce mode de fonctionnement pour tracer la version de nos modèles au format OVF. Le choix s'est arrêté sur la présence d'un fichier « /etc/cls-release » dont le contenu serait ainsi normalisé :

```
NOM release num_version (champ-libre)
```

Sur un modèle CentOS 5, le fichier « /etc/cls-release » aurait la forme :

```
CLS_CENTOS5 release 1.0 (x86_64)
```

ITIL préconise qu'un actif du service informatique doit être identifiable (enregistrement et restitution des informations) dans notre CMBD. Actuellement, dans notre infrastructure physique nos éléments sont identifiés via un numéro d'inventaire (code barre), associé ensuite aux caractéristiques techniques et à son numéro de série. Dans l'infrastructure virtuelle, nos éléments sont identifiés via un « tag », avec la nomenclature « VM-NOM », ce qui permet d'identifier rapidement que l'élément est virtuel et non physique. Pour les caractéristiques techniques, nous n'avons actuellement pas de moyen de remonter cette information. Pour ce faire, il faut qu'on puisse identifier le numéro de version de notre configuration. Cependant, notre outil d'inventaire ne remonte pas de façon native cette information. Il va falloir adapter notre outil d'inventaire ou trouver une alternative afin que cette information soit facilement exploitée et stockée dans notre CMDB.

V.1.4 Choix de la version hardware

Une étape importante à la création de la machine virtuelle est le choix de sa «virtual hardware version» (VHV). En effet, il existe plusieurs versions liées aux évolutions des serveurs ESX, des capacités et fonctionnalités des machines virtuelles. Cela a son importance car la compatibilité du modèle dépendra de versions d'ESX. En effet, la « VHV 7 » n'est compatible qu'à partir de la version 4 d'ESX/ESXi. Il sera alors impossible le déployer sur des ESX 3.5.

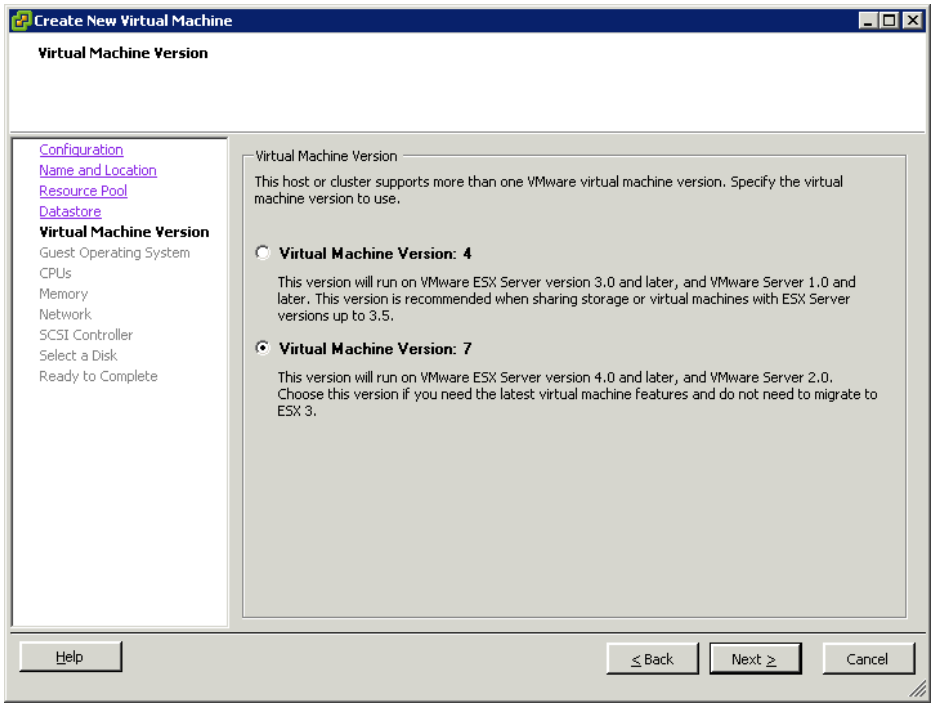

Figure 22 : Virtual Machine Version

Vous trouverez dans le tableau 4 une vue simplifiée des compatibilités de la gamme ESX/ESXi avec les versions des machines virtuelles.

Tableau 4 : Compatibilité ESX / VH

Produits	Virtual Hardware version
ESXi 5.x	8
ESXi/ESX 4.x	7
ESX 3.x	4

Nous avons fait le choix de créer nos templates en VHV 7. En effet, cette version introduit de nouvelles fonctionnalités : il est possible de créer une machine virtuelle avec 8 vCPU et 255Go de mémoire vive et de rajouter à chaud des composants tels que la mémoire et le processeur. Elle apporte également de nouvelles compatibilités pilotes (réseau avec vmxnet3, pilotes ISCSI, …)

V.2 Maquette

Nous allons créer tout d'abord une maquette qui sera notre premier modèle. La sélection d'un sous-ensemble représentatif pour le modèle système portera sur la famille la plus représentée dans notre infrastructure, à savoir Linux. Le choix de la distribution suit également la même logique, comme défini dans le socle technique, nous allons utiliser la distribution CentOS.

V.2.1 Création de l'image maîtresse

Avant de commencer l'installation du système d'exploitation, il existe plusieurs pré-requis que nous allons déterminer.

V.2.1.1 Choix de l'architecture

Il faut au préalable définir le choix de l'architecture. De nos jours, il existe principalement deux possibilités, un système installé en 32 bits et un en 64 bits. Depuis un peu plus de vingt ans, les processeurs x86 effectuaient des instructions 32 bits mais depuis quelques années l'évolution matérielle a permis aux nouveaux processeurs de gérer des instructions 64 bits (x86-64). Ces processeurs sont de nos jours les standards pour les serveurs, ceux dont disposent nos ESX. Après un inventaire de ceux-ci, nous disposons de plusieurs gammes : Xeon X7460, E730, X7350 et AMD Opteron 8358 et 2384. Ces processeurs supportent bien sûr des systèmes 32 bits (legacy mode) mais le système 64 bits a plusieurs avantages :

- Sans technologie ajoutée (par exemple, la pagination prise en charge par le processeur), les processeurs 32 bits ne peuvent pas adresser plus de 4Go de mémoire vive, contrairement aux processeurs 64 bits.

- Doublement des registres (emplacement mémoire à un processeur), cela offre en théorie la possibilité de transmettre deux fois plus d'informations dans le même temps.

Notre premier modèle système sera donc dans une architecture 64 bits, outre le fait que ce soit l'architecture du présent et de l'avenir, c'est le type d'architecture aujourd'hui utilisé par l'équipe système, hors contre-indication applicative.

V.2.1.2 Type d'installation

Comme nous l'avons vu dans la problématique, l'objectif est d'avoir une installation nouvelle, « propre ». L'installation ne doit pas être celle par défaut, mais une installation allégée, optimisée. Si des paquets supplémentaires sont nécessaires, ils pourront être installés et documentés dans la note d'installation du modèle. Pour ce faire, nous optons pour l'option «installation minimale » existant depuis peu sur les distributions Linux les plus célèbres. La définition issue du site internet de CentOS pour ce type d'installation est la suivante et nous conforte dans ce choix : « Le but est d'installer une CentOS système qui a un minimum de paquets nécessaires pour avoir un système fonctionnel. ».

V.2.1.3 Gestion de parc

Au préalable, nous avons fait le choix de créer un fichier « /etc/cls-release » contenant les informations du modèle. L'objectif est maintenant de pouvoir remonter cette information dans notre outil d'inventaire et de gestion de parc. La modification de l'agent ocsinventory est assez complexe car l'on doit recompiler l'agent pour toute modification, ce qui rend la maintenance future compliquée. A chaque nouvelle version, nous serions obligés de compiler l'agent, en espérant que les nouvelles versions ne révolutionneront pas le fonctionnement. Nous avons fait le choix de créer un paquet au format RPM[6] (ou DPKG pour la distribution

[6] RPM Package Manager est un système de gestions de paquets logiciels utilisé sur certaines distributions Linux, dont CentOS

Debian) pour les différents modèles et versions qui seront de ce fait automatiquement remontés par notre outil d'inventaire sans modification de l'agent ou du serveur.

Pour la maquette, j'ai créé le paquet RPM qui sera par la suite installé dans le modèle. Vous trouverez en annexe 1 la notice de création du paquet.

Afin d'en tester le bon fonctionnement, j'ai installé le paquet sur un serveur de test. Celui-ci génère un fichier « /etc/cls-release » à l'installation. Le résultat est cohérent à nos attentes, disponible dans l'interface d'ocsinventory dans la rubrique « software ». La figure 23 représente ce résultat.

Figure 23 : Résultat du paquet cls-release dans ocsinventory

Outre le modèle et sa version déployée sur un serveur, on pourra par la suite avoir des données intéressantes pour les statistiques, comme par exemple la répartition des versions de modèle au sein de notre système d'information.

V.2.1.4 Installation

Maintenant que les bases sont définies, la première étape avant l'installation consiste à récupérer l'image disque au format ISO sur le site officiel de la distribution. Pour la maquette, nous avons utilisé l'image « CentOS-6.0-x86_64-bin-DVD». La première étape a été la création de la machine virtuelle et de ses caractéristiques :

```
RAM : 512 Mo
1 vCPU
HDD1 : 10Go ( / et swap )
Réseau : Reseau-bleu
```

Ces caractéristiques ne sont pas figées, elles pourront être modifiées par la suite après le déploiement du modèle. En effet, lors de la configuration, tout est prévu afin que nous n'ayons pas de difficulté à modifier ces caractéristiques :

- Kernel SMP (symmetric shared memory multiprocessor) afin de pouvoir ajouter facilement des vCPU.

- Système de fichier formaté en LVM [7] afin de pouvoir étendre facilement l'espace disque en cas de besoin.
- Configuration réseau par défaut en DHCP[8] afin de pouvoir effectuer les premières administrations sur notre réseau bureautique.

Une fois l'installation minimale réalisée, j'ai suivi la check-list « installation d'un serveur linux » afin de passer toutes les étapes validées par l'équipe. En effet, nous disposons d'une check-list post installation d'un serveur Linux qui nous permet, lors de l'installation d'un système, d'avoir une cohérence dans nos installations, même si celle-ci ne garantit pas à ce jour des systèmes identiques. La figure 24 représente les étapes principales entre la création de la machine virtuelle jusqu'au modèle.

[7] Logical Volume Management (gestion par volumes logiques) est une méthode et un logiciel de découpage, concaténation et d'utilisation des espaces de stockages

[8] Dynamic Host Configuration Protocol, est un protocole réseau dont le rôle est d'assurer la configuration réseau automatiquement

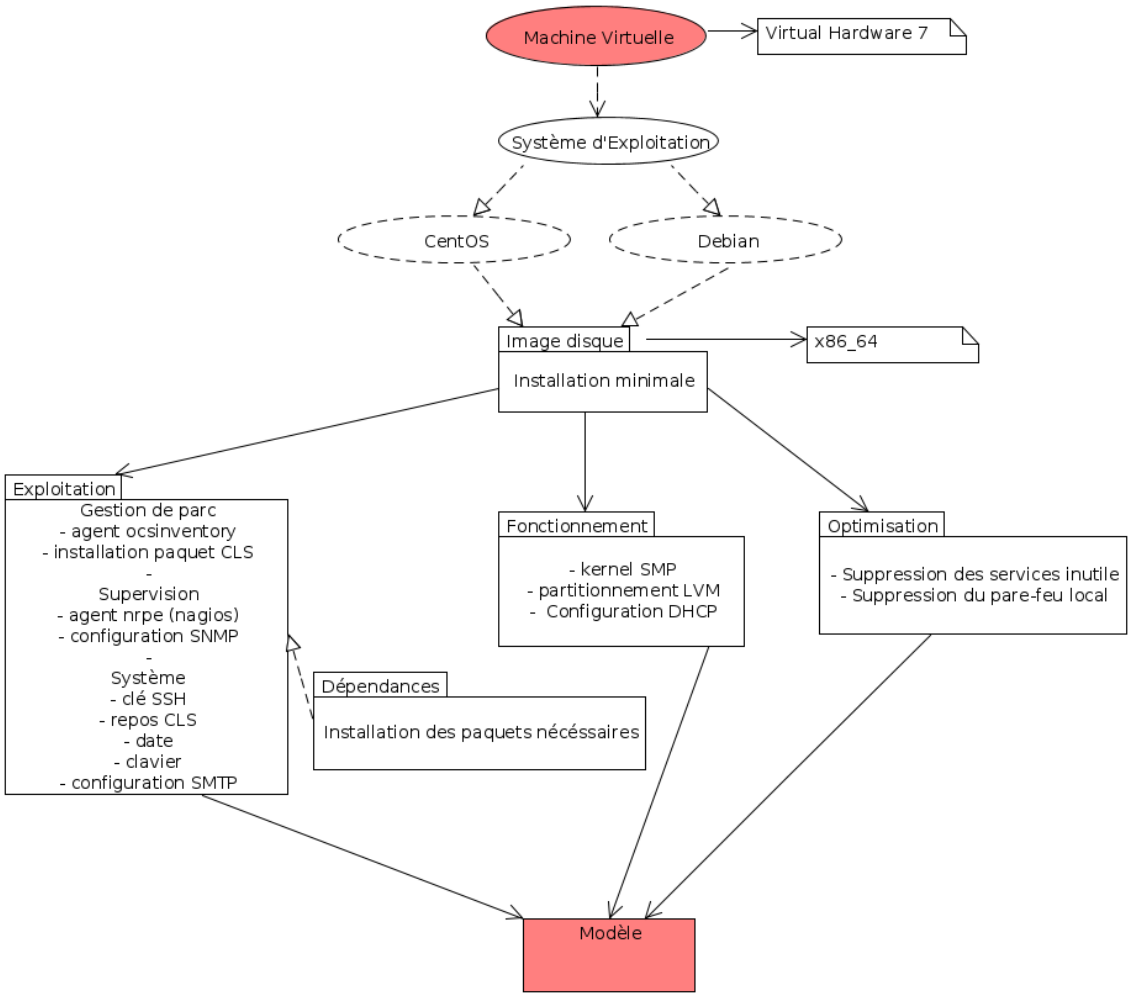

Figure 24 : Schéma de création du modèle

Ci-dessous le résumé des étapes :

- Mise à jour du système
- Paquets supplémentaires
- Suppression des services inutiles
- Déploiement des clés ssh
- Ajout du repository CLS
- Configuration SNMP
- Configuration date via ntp
- Modification du clavier
- Configuration de la messagerie (smtp)
- Suppression du pare-feu local

- Configuration de l'agent de supervision
- Configuration de l'agent d'inventaire
- Installation et configuration des Vmware Tools
- Configuration réseau

De plus, nous avons mis un message d'accueil au démarrage de la machine virtuelle permettant une aide à la configuration post-déploiement. Par exemple, voici le message pour la maquette :

```
##########################################
#         TEMPLATE CENTOS 6              #
#                                        #
#         Pour installer ce template veuillez    #
#         lire le fichier suivant :      #
#         /root/INSTALL_README.txt       #
#                                        #
##########################################

(c) CLS 2011. DT/SI
```

A la fin de la création du système, nous conservons l'état des paquets installés ainsi que le numéro de leurs versions respectives. Pour obtenir cette information, une commande existe qui diffère selon les distributions.

Vous trouverez plus en détail l'installation système et le détail des paquets installés dans l'annexe 2.

V.2.1.5 Validation

Une fois la machine virtuelle installée, les membres de l'équipe système ont contrôlé et validé si la machine virtuelle correspondait bien aux besoins. Chacun, dans ses domaines de compétences et responsabilités respectifs, a vérifié l'état du système d'exploitation et les outils associés. Une fois les validations effectuées, la création du modèle OVF est possible. La figure 25 représente l'enchainement des étapes pour la création de la maquette.

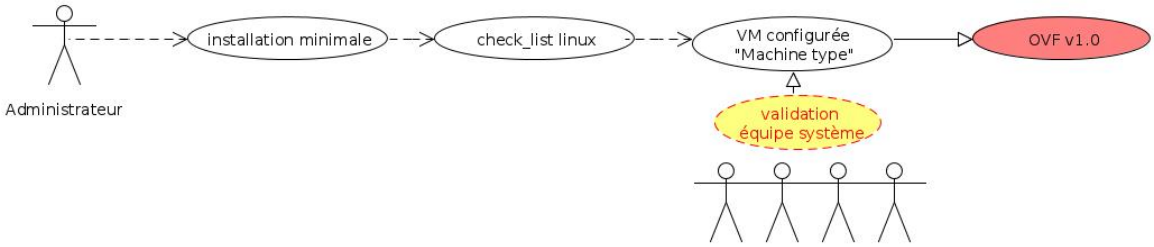

Figure 25 : Création d'un OVF système

Nous verrons dans un prochain chapitre l'organisation mise en place lors la création et validation d'un modèle.

V.2.2 Création OVF

Maintenant que notre système est validé, nous allons convertir notre machine virtuelle au format OVF. Pour la création, nous avons envisagé d'utiliser plusieurs outils comme OVF Tool, VMWare Studio et vCenter.

Tableau 5 : Outils de création OVF

	Interface graphique	Création vm	Déploiement vm
OVF Tool	✗	✓	✓
VMWare Studio	✓	✓	✓
vCenter	✓	✓	✓

La solution retenue est la plus simple, celle intégrée dans l'outil de gestion centralisée, le vCenter. OVF Tool est un bon compromis qui pourra servir si l'on souhaite par exemple automatiser par script le déploiement. Nous avons également testé VMWare Studio, qui ne nous a pas entièrement satisfaits, notamment par le manque de souplesse de l'outil mais également dans le manque de visibilité sur le résultat final.

La figure 26 représente l'écran du vCenter pour la création du modèle OVF depuis une machine virtuelle :

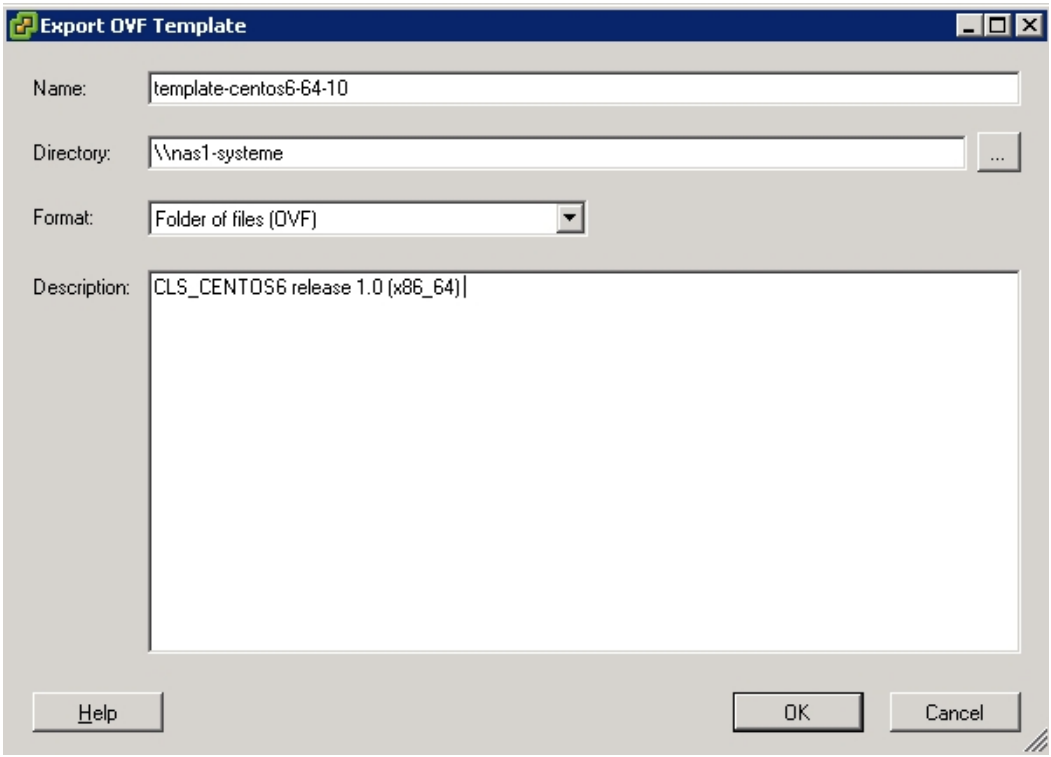

Figure 26 : Export via vCenter

Le modèle OVF généré est « brut ». On va ensuite éditer le fichier OVF (.ovf / XML) pour compléter les informations qui apparaitront lors du déploiement :

```
<ProductSection>
   <Info>test</Info>
            <Product>CLS Centos 6 (64bits)</Product>
            <Vendor>CLS</Vendor>
            <Version>1.0</Version>
            <FullVersion>1.0</FullVersion>
</ProductSection>
```

Il faut également mettre à jour le fichier « manifest » (.mf) sinon le déploiement sera impossible lors du contrôle d'intégrité. Pour obtenir la signature du fichier nous avons utilisé la commande sha1sum et nous avons reporté le résultat dans le fichier manifest.

```
sha1sum template-centos6-64-10.ovf
8d9e266928a23dbd9f1d32e30100de093c089dba  template-centos6-64-10.ovf
```

Comme nous l'avons vu, il n'y a pas de difficulté particulière dans la création à proprement dite d'un OVF depuis une machine virtuelle.

V.2.3 Test déploiement OVF

Nous avons testé le déploiement de notre modèle OVF dans différentes solutions de virtualisation afin de valider la portabilité des modèles. Dans un premier temps, nous avons testé le déploiement sur des hyperviseurs de type 2, potentiellement utile pour fournir des maquettes. Ensuite, sur des hyperviseurs de type 1, pour terminer sur une conclusion sur la portabilité des modèles OVF.

V.2.3.1 Déploiement sur des hyperviseurs type 2

Déploiement sur VirtualBox

Oracle VirtualBox prend officiellement en charge le format OVF. On peut, de ce fait, importer le modèle OVF directement et les paramètres importés correspondent bien à nos attentes, comme le montre la figure 27.

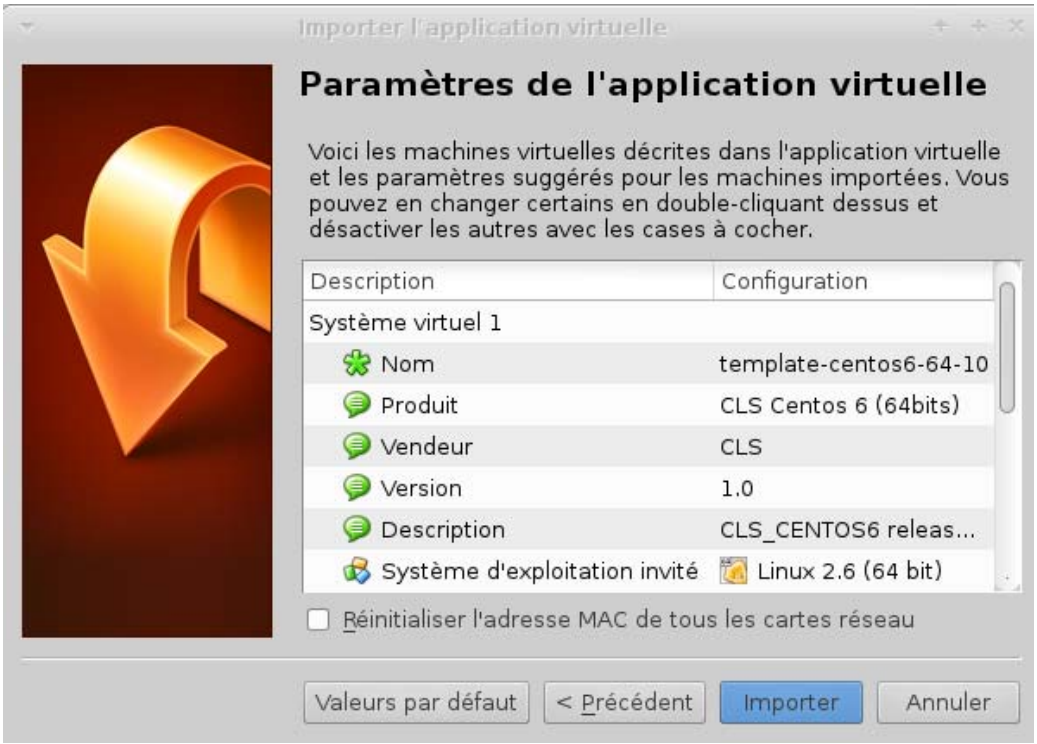

Figure 27 : OVF sous Oracle VirtualBox

Une fois importée, la machine virtuelle démarre correctement.

Déploiement sur Qemu-KVM

Comme précédemment, on crée une nouvelle machine virtuelle avec comme option « importer une image disque existante » et on sélectionne notre OVF. Après la fin de l'importation, la machine virtuelle démarre mais pas correctement. Elle reste bloquée au contrôle des partitions, malgré plusieurs tentatives et

déploiements. Ceci est dû au format du disque de l'OVF, à savoir vmdk qui est un format propriétaire de VMWare. Qemu-kvm est compatible avec ce format, mais il ne s'agit pas de son format natif. Nous allons donc convertir le format du disque afin de corriger le problème. Qemu-kvm met à disposition un outil qui permet de convertir des formats des autres solutions vers un format compatible avec sa solution. Par exemple, on va convertir notre modèle OVF au format de la solution, à savoir qcow2. D'autres formats de conversion sont possibles comme par exemple le format brut (raw) :

```
 [root@px-olive test]# virt-convert -v -i ovf -D qcow2 template-centos5-64-12-
ovf.ovf
Sortie générée au format « virt-image » pour template-centos55_64bits/
Conversion du disque « template-centos5-64-12-ovf-disk1.vmdk » vers le type
qcow2...
Done.
```

Une fois réalisé, la machine virtuelle démarre correctement comme le montre la figure 28.

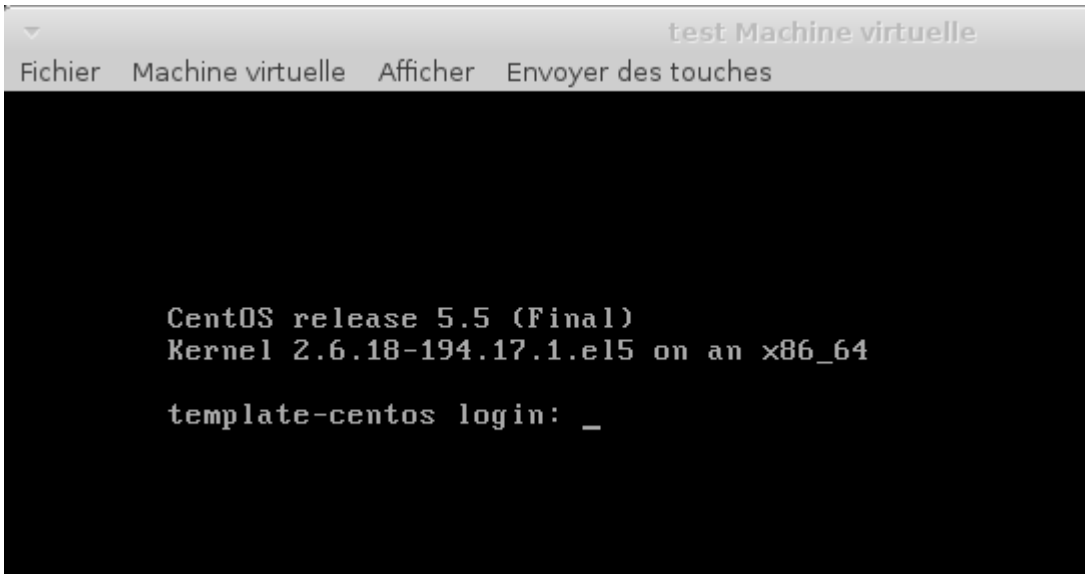

Figure 28 : Résultat sous qemu-kvm

V.2.3.2 Déploiement sur des hyperviseurs de type 1

Déploiement sur la solution VMWare

Nous avons testé le déploiement de notre modèle OVF dans notre infrastructure. Le déploiement diffère peu de la création d'une machine virtuelle. La figure 29

représente le premier écran qui est un résumé du modèle. Les modifications apportées dans le fichier OVF sont bien présentes.

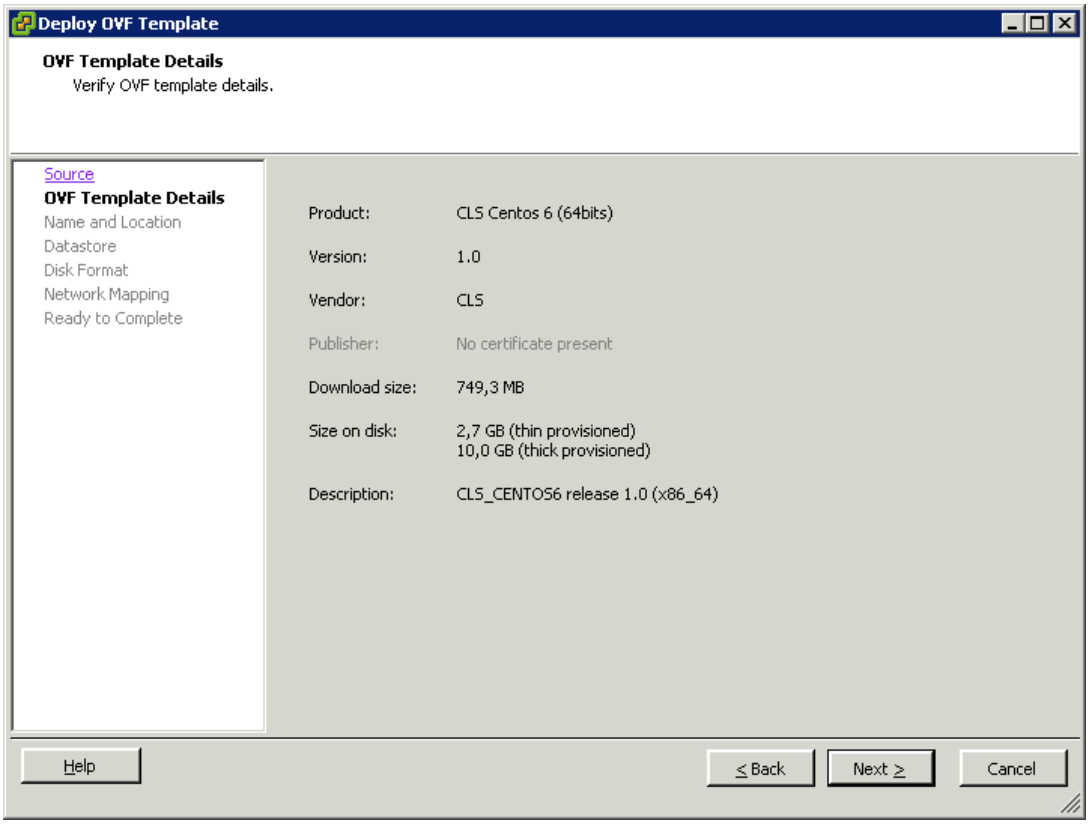

Figure 29 : Déploiement OVF sur VMWare

Après avoir sélectionné les informations de déploiements les plus courantes, comme le nom de la machine virtuelle, le serveur ESX de destination, l'emplacement disque et son format, son réseau, etc. le résultat du déploiement correspond à nos attentes. Ce n'est pas une surprise dans le sens où l'on déploie le modèle OVF vers la même solution que la création.

Nous avons également testé le déploiement du modèle OVF sur différents ESX, dans une autre version. Le résultat correspond à nos attentes.

Déploiement sur la solution XenServer

XenServer prend en charge le format OVF, seulement l'importation de notre modèle OVF échoue, comme le montre la figure 30.

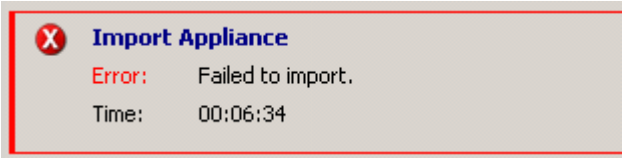

Figure 30 : Importation échouée sous XenServer

Le problème est identique à celui rencontré lors du test de portabilité sur Qemu-kvm, à savoir le format du disque de la machine virtuelle. Une fois converti, cela fonctionne.

V.2.3.3 Conclusion sur la portabilité testée

On peut conclure que la portabilité d'un modèle OVF entre les différentes plates-formes de virtualisation est possible mais pas native. En effet, même si les différentes solutions sont capables d'interpréter de la même manière le descripteur OVF, on se rend compte que le format du disque, qui contient les données brutes, est l'élément qui va poser des problèmes. Chaque éditeur a encore à ce jour un format propriétaire. Comme vu précédemment, on peut utiliser des outils pour convertir le disque dans les différents formats. Une solution serait de fournir l'OVF avec les différentes images disques.

La portabilité ne sera donc totale que si un format d'image disque unique existe.

V.3 Organisation

V.3.1 Gestion des modèles

Nous avons décidé de gérer la vie de nos modèles comme le préconise ITIL. C'est-à-dire que la création d'un modèle OVF, d'une mise à jour, nécessite le passage par le processus de gestion des changements. La figure 31 représente le processus de Gestion des Déploiements et Mises en Production au sens ITIL. Dans notre organisation, concrètement cela va se traduire par la création d'une « demande de changement » dans notre outil ISILOG.

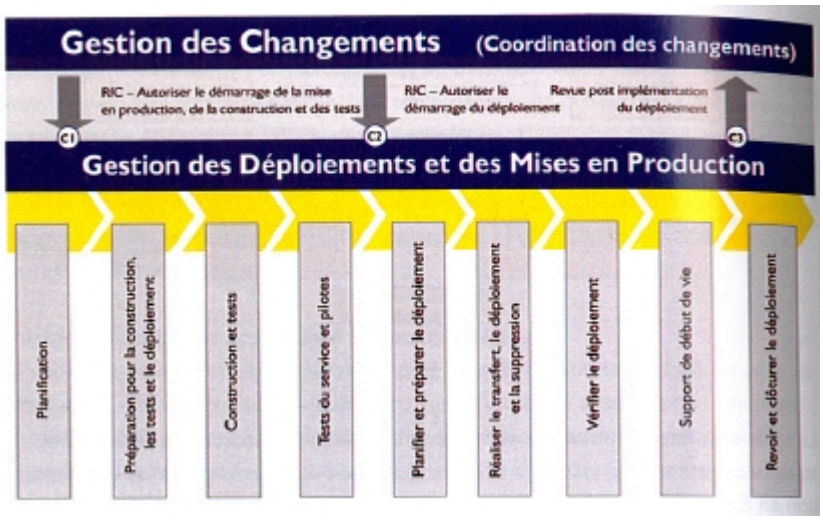

Figure 31 : Gestion des changements

En procédant de la sorte, chaque équipe technique aura une action dédiée, reprenant le schéma proposé par ITIL. Ces actions permettront de valider le modèle OVF. La figure 32 nous montre un exemple de demande de changement via notre outil de gestion de tickets, à savoir Isilog.

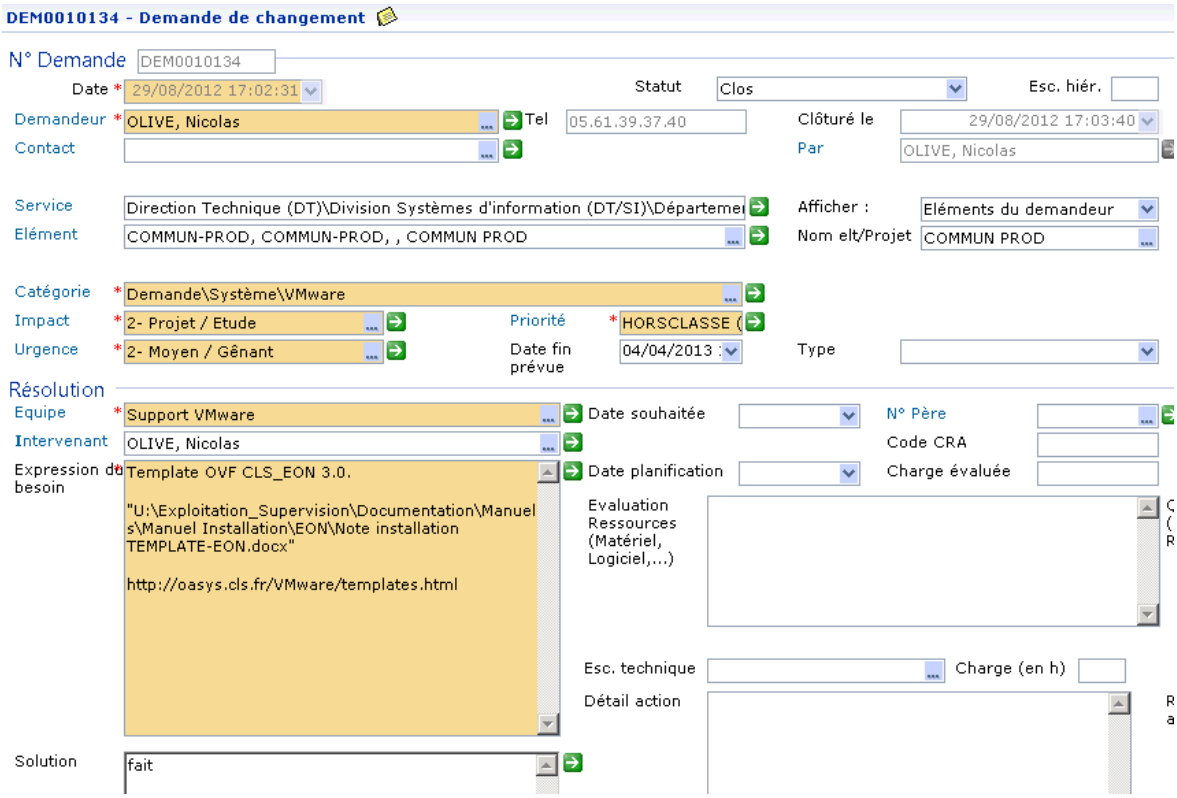

Figure 32 : Exemple Demande de Changement

V.3.2 Stockage des OVF

Comme nous l'avons vu précédemment, le format OVF permet un gain de place par rapport à un modèle classique, voire à une machine virtuelle. Dans l'exemple précédent, le modèle OVF fait 749Mb, contre 2,7Go pour le modèle et 10Go pour la machine virtuelle. De ce fait, nous avons un espace de stockage de 5.5 To qui fait office de stockage centralisé nous permettant de stocker tous nos modèles ainsi que leurs versions.

L'arborescence type définie contient :

- Un répertoire « archives » : avec tous les modèles OVF anciennes versions que nous n'utiliserons plus pour les nouvelles machines mais pouvant être utilisés en cas de besoin.
- Un répertoire par modèle OVF dans la dernière version disponible à déployer pour toute nouvelle machine virtuelle.

Nomenclature :

nom-version

Par exemple, la figure 33 représente l'arborescence que l'on aurait dans le répertoire courant.

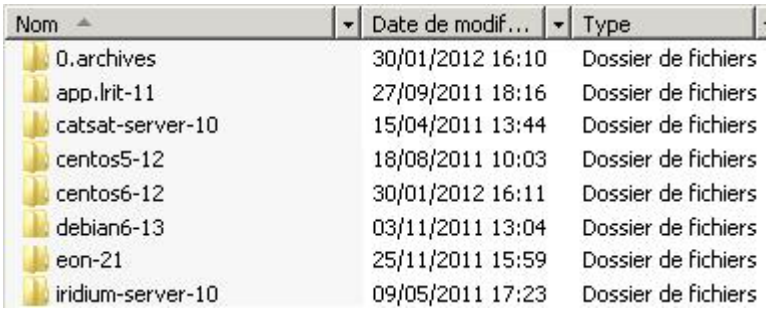

Figure 33 : exemple d'arborescence

V.3.3 Interface web

Une fois l'emplacement des modèles déterminé, nous avons souhaité donner un accès plus simple pour tous les membres de l'équipe amenés à utiliser ces

modèles. Notre objectif est d'avoir une interface claire et lisible sur nos modèles disponibles. L'idée d'une liste dans une page internet est la solution choisie.

Nous disposons d'une machine virtuelle dédiée à l'équipe-système où nous centralisons nos informations « équipe ». Ce portail regroupe des informations utiles à l'équipe, des scripts utiles à l'exploitation, exécutables depuis le portail (script qui vérifie l'existence d'un email, modification de mot de passe LDAP, comptes VPN, etc.).

Nous aurons donc une page dédiée aux modèles, avec la version actuelle (stable) et les anciennes, disponibles en cas de nécessité. Un automontage a été mis en place sur le serveur afin qu'il puisse accéder à la baie de stockage contenant les modèles, ce qui permet donc d'accéder aux OVF.

Le lien internet fourni (URI) peut être directement utilisé dans le vCenter pour le déploiement comme le montre la figure 34.

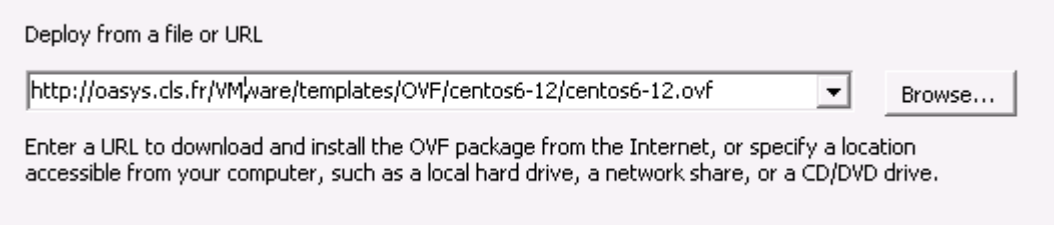

Figure 34 : Exemple utilisation URI

La figure 35 représente cette page internet accessible à l'équipe système.

Templates OVF

Mis à jour le 25/11/2011

Centos 6

Stable

CLS_CENTOS6 release 1.2 (x86_64)	template-centos6-64-1.2.ovf

Archives

CLS_CENTOS6 release 1.1 (x86_64)	template-centos6-64-1.1.ovf

Debian 6

Stable

CLS_DEBIAN6 release 1.3 (x86_64)	debian6-64-13.ovf

Archives

Pas d'archives

Centos 5

Stable

CLS_CENTOS5 release 1.2 (x86_64)	template-centos5-64-12-ovf.ovf

Archives

CLS_CENTOS5 release 1.1 (x86_64)	template-centos5-64-11-ovf.ovf
CLS_CENTOS5 release 1.0 (x86_64)	no OVF file

Eyes Of Network

Stable

CLS_EON release 2.1 (EON_3.0)	template-eon-21.ovf

Archives

CLS_EON release 2.0 (EON_3.0)	template-eon-20.ovf
CLS_EON release 1.2 (EON_2.2)	template-eon-12.ovf
CLS_EON release 1.1 (EON_2.2)	template-eon-11.ovf
CLS_EON release 1.0 (EON_2.2)	template-eon-10-ovf.ovf

OAE

Stable

OAE-SERVER release 1.0	template-OAE-1.0.ovf
Argos-OAE release 1.1	argos-oae-1.1.ovf

Figure 35 : Page Web Templates OVF

V.4 Mise en Production

Après la création de la maquette et la mise en place de l'organisation, nous allons pouvoir effectuer les premières mises en production en utilisant ces modèles, définies par le périmètre du projet. L'objectif est la mise en production d'un modèle par classe de machines virtuelles.

V.4.1 Le modèle système

Comme expliqué dans le chapitre V.1.3, nous disposons pour ce type de modèle d'une version majeure de modèle par version majeure de distribution. Les versions mineures seront, quant à elles, revues selon les exigences (mises à jour importantes de sécurité, etc.).

La création des modèles système s'est effectuée comme ceci :

- Création d'un modèle pour CENTOS en version 6, 64 bits, basé en VHV version 7
- Création d'un modèle pour DEBIAN en version 6, 64 bits, basé en VHV version 7

Nous avons pris la décision qu'il n'y aurait pas de déploiement massif sur tout notre parc concernant les modèles OVF systèmes. En effet, l'impact potentiel et le risque d'un tel déploiement ont été jugés trop importants. Nous allons fournir, à partir d'aujourd'hui pour tout nouvel environnement demandé sur l'infrastructure virtuelle, un des modèles disponibles à CLS répondant au socle technique défini : le modèle Debian ou CentOS. Le déploiement dans notre infrastructure s'effectuera au fil du temps : l'objectif est raisonnable.

V.4.2 Le modèle applicatif

Nous avons choisi de prendre comme sous ensemble représentatif l'application Iridium. En effet, elle a été l'application qui a fait l'expérience de l'émergence et des nouveautés de la virtualisation, il s'agit de la première application 100% virtuelle à CLS.

Comme expliqué dans la problématique, l'application Iridium a été confrontée à des problèmes, qui ont entre autre donné naissance à ce mémoire. Iridium a eu des soucis avec des problèmes d'arrondis. L'origine du problème n'a pas été détectée mais le constat est que le système n'a pas pu être dédouané. De plus, des problèmes de stabilité et de performance sont présents depuis l'augmentation du nombre de balises suivies par notre centre.

Son implémentation en production remonte à 2007. Cette application est composée de trois configurations :

- La configuration de QT (Qualification Technique) composée de deux machines virtuelles : iridium-qt et iridium-db-qt. Cette configuration est essentiellement opérée par les équipes de développement.
- La configuration de QO (Qualification Opérationnelle) composée de trois machines virtuelles : iridium-sfr-qo, iridium-trm-qo et iridium-db-qo. Cette configuration est essentiellement opérée par les équipes d'exploitation.
- La configuration de PROD (Production) composée de six machines virtuelles : iridium-sfr, iridium-trm, iridium-db, iridium-sfr-2, iridium-trm-2 et iridium-db-stby. Cette configuration est essentiellement opérée par les équipes système.

La figure 36 représente le schéma d'architecture de la production, incluant son mode Disaster Recovery.

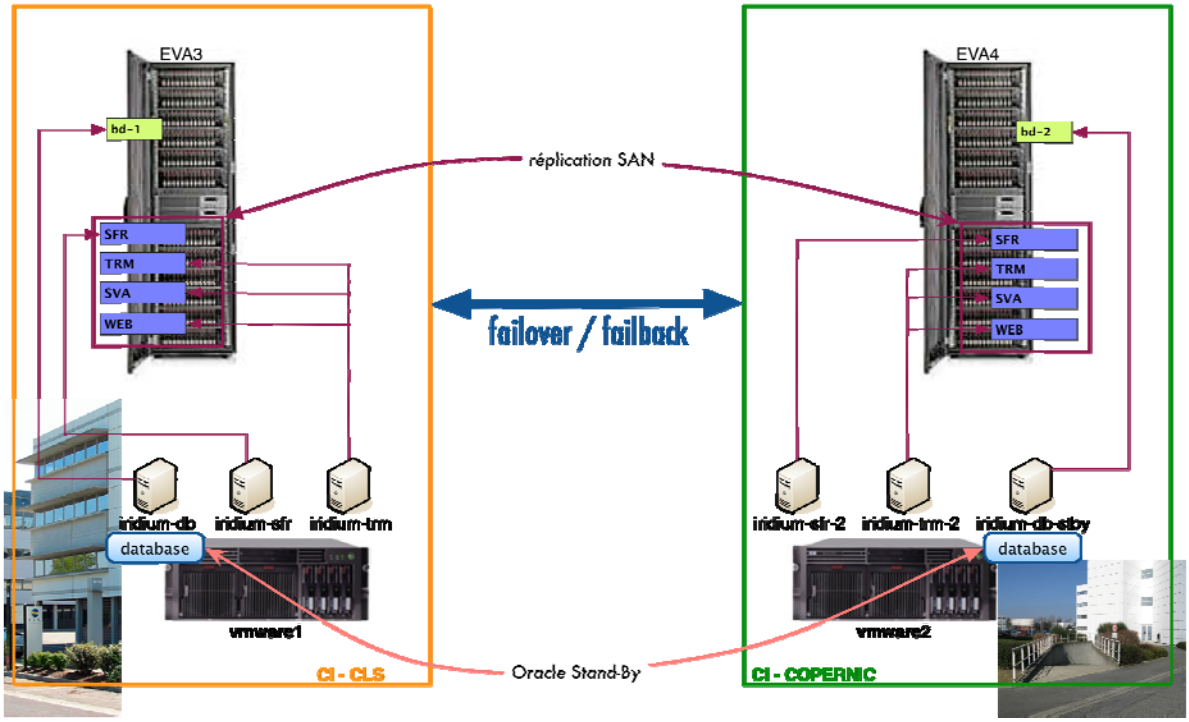

Figure 36 : Architecture PROD Iridium

Les machines virtuelles hébergeant l'application Iridium sont actuellement sous la distribution « Redhat AS 4 u5 » sur une architecture 32 bits. Nous sommes incapables de certifier que le système est identique entre la QT, la QO et la PROD. Bien au contraire, de fortes convergences ont été ressorties entre les trois environnements : la présence de paquets dans l'environnement de production et non présents dans les autres environnements, versions différentes de paquets, comptes locaux différents, etc.

On m'a confié de ce fait, en collaboration avec les équipes Produits et Opération, la migration de l'application Iridium vers CentOS, dans une architecture 64 bits. L'objectif étant de stabiliser l'application et d'avoir des environnements identiques. Les contraintes du projet sont fortes : il s'agit d'une application H24, le droit à l'erreur est faible, surtout avec les problèmes passés.

Pour résumer la création du modèle OVF Iridium, nous avons créé en premier lieu un modèle CentOS dans la version 5. Deux raisons à cela, les serveurs ESX hébergeant l'application Iridium sont en version 3.5, la VHV doit être donc en version 4 maximum, contrairement au modèle système créé plus tôt. De plus, les équipes Iridium sont réfractaires à une évolution supérieure à une version majeure du système d'exploitation, sachant à juste titre que la version 6 de CentOS a peu

de vécu en exploitation chez nous, et surtout le passage en 64 bits représente déjà à leurs yeux une étape importante. Après la création du modèle système CentOS 5 qui a servi de base pour ce modèle, nous avons suivi la note d'installation initiale du projet Iridium qui date de 2006. Je vous invite à consulter l'annexe 3 pour avoir le détail de cette installation.

Le début du projet avait été fixé au 1er février 2011 et clôturé fin août 2011. Vous trouverez les tâches définies dans le diagramme de GANTT représenté par la figure 37.

Tâches

TPÉ	Nom	Démarré	Terminé	Travail	État d'avancement	Assigné à
1	création template	Feb 1	Feb 21	15d	100%	nolive
2	validation équipe-système	Feb 22	Mar 7	10d	100%	equipe-systeme
3	validation équipe-intégration	Mar 8	Mar 21	10d	100%	equipe-integration
4	déploiement QT	Mar 22	Mar 22	1d	100%	nolive
5	mise en production QT	Mar 23	Mar 23	1d	100%	nolive, equipe-integration
6	période de validation QT	Mar 23	May 4	30d	100%	equipe-integration
7	déploiement QO	May 4	May 5	1d	100%	nolive
8	mise en production QO	May 5	May 5	1d	100%	nolive, equipe-exploitation
9	période de validation QO	May 6	Jun 16	30d	100%	equipe-exploitation
10	déploiement PROD	Jun 30	Jun 30	1d	100%	nolive
11	mise en production PROD secours	Jul 4	Jul 4	1d	100%	nolive, equipe-exploitation
12	période de validation PROD secours	Jul 4	Jul 5	1d	100%	equipe-exploitation
13	mise en production PROD primaire	Jul 6	Jul 6	1d	100%	nolive, equipe-exploitation
14	période de validation PROD primaire	Jul 6	Aug 17	30d	100%	equipe-exploitation
15	clôture projet	Aug 31	Aug 31	1d	100%	nolive

Figure 37 : Tâches du projet Migration Iridium vers CentOS 64 bits

Pour expliquer le déroulement du projet, comme le démontre la figure 38, le modèle a été créé et une première machine virtuelle de test a été livrée à l'équipe intégration du projet Iridium. Une fois validée, nous avons pu créer le modèle OVF. A partir de celui-ci, nous avons fourni la machine de QT, à savoir « iridium-qt ». Une fois que les équipes de développement et d'intégration Iridium ont testé et validé que l'applicatif fonctionnait correctement sur ce système, nous avons alors déployé les machines de QO, puis de PROD.

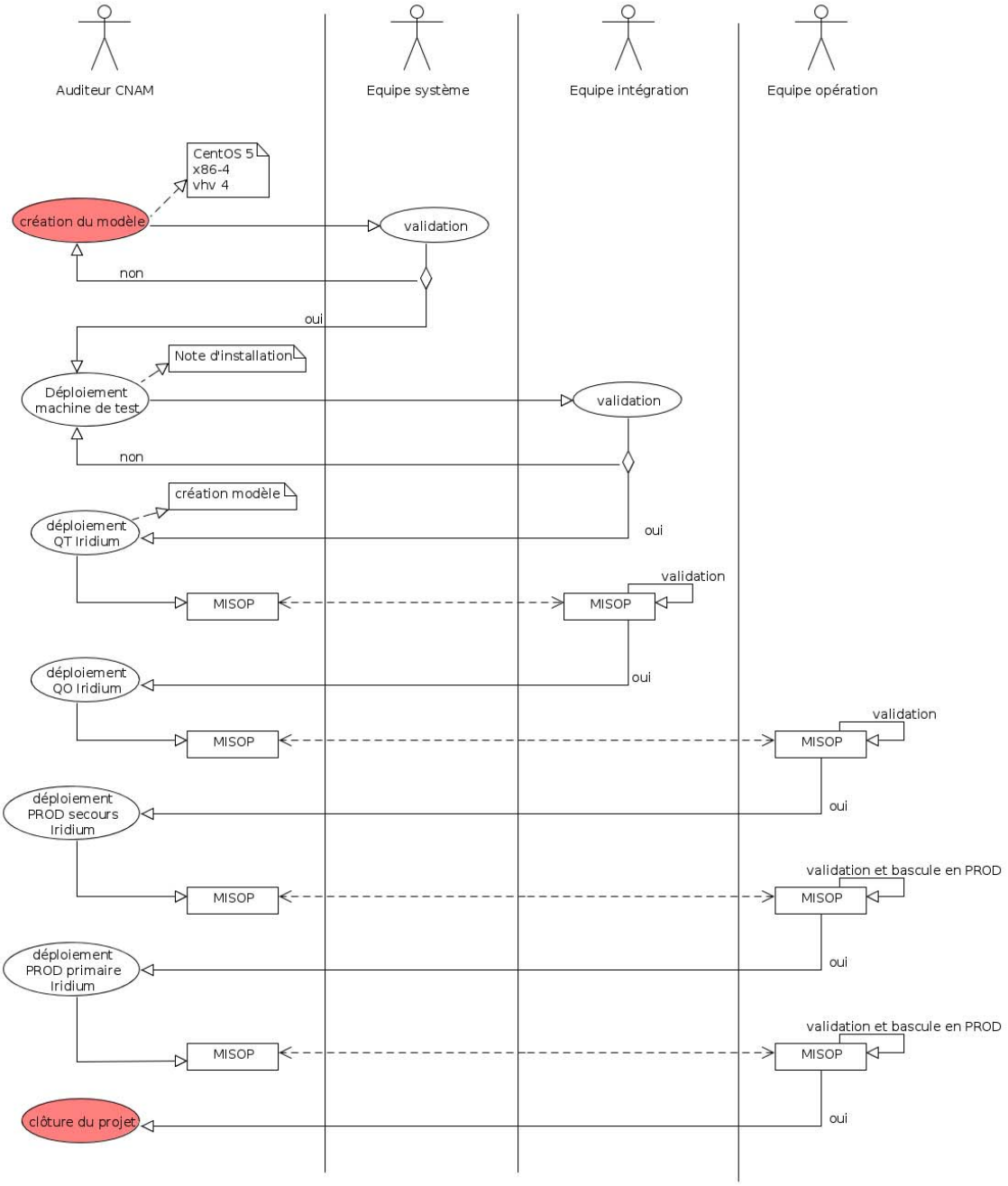

Figure 38 : Logigramme Iridium

Le projet s'est déroulé correctement et sans anicroches. Cela a été une grande satisfaction pour les équipes, puisque l'objectif a été atteint sans impact majeur. Les avantages sont multiples :

- Avantages immédiats :
    - Configuration système identique entre la QT / QO / PROD
    - Installations tracées et contrôlées

- Avantages futurs :
    - Evolutions futures simplifiées : par exemple, il est prévu de séparer les services WEB et TRM qui sont actuellement hébergés sur le même serveur. Concrètement, cela correspond au déploiement d'un nouveau serveur iridium : le déploiement sera plus rapide et conforme aux autres serveurs de l'application.
    - Restauration rapide en cas de problème.
    - Déploiements rapides dans nos centres Iridium hors CLS-Toulouse.

V.4.3 Le modèle service

J'ai choisi de prendre comme sous-ensemble représentatif la solution de supervision dont je suis responsable à CLS, à savoir Eyes Of Network. Il s'agit d'une solution « OpenSource » qui imbrique plusieurs modules du monde de la supervision sur un système d'exploitation et certifie que les modules interagissent correctement. La figure 39 représente l'imbrication des modules de supervision :

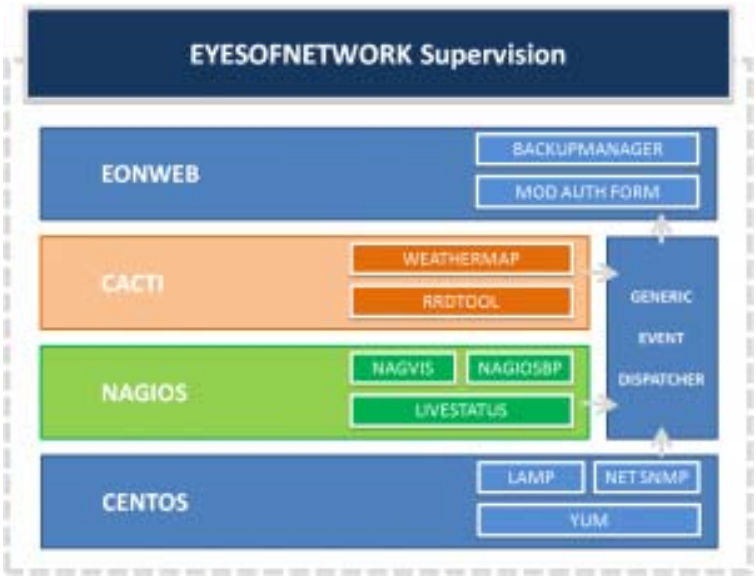

Figure 39 : Eyes Of Network

La société éditrice de la solution, APX, ne fournit pas cette solution au format OVF. Aussi, à CLS, l'installation a lieu par l'image disque ISO qui est ensuite complétée par une configuration de la solution par rapport à notre environnement (compte/plugins métiers/…).

Actuellement, nous avons plusieurs serveurs de supervision déployés dans le groupe CLS, représenté par la figure 40.

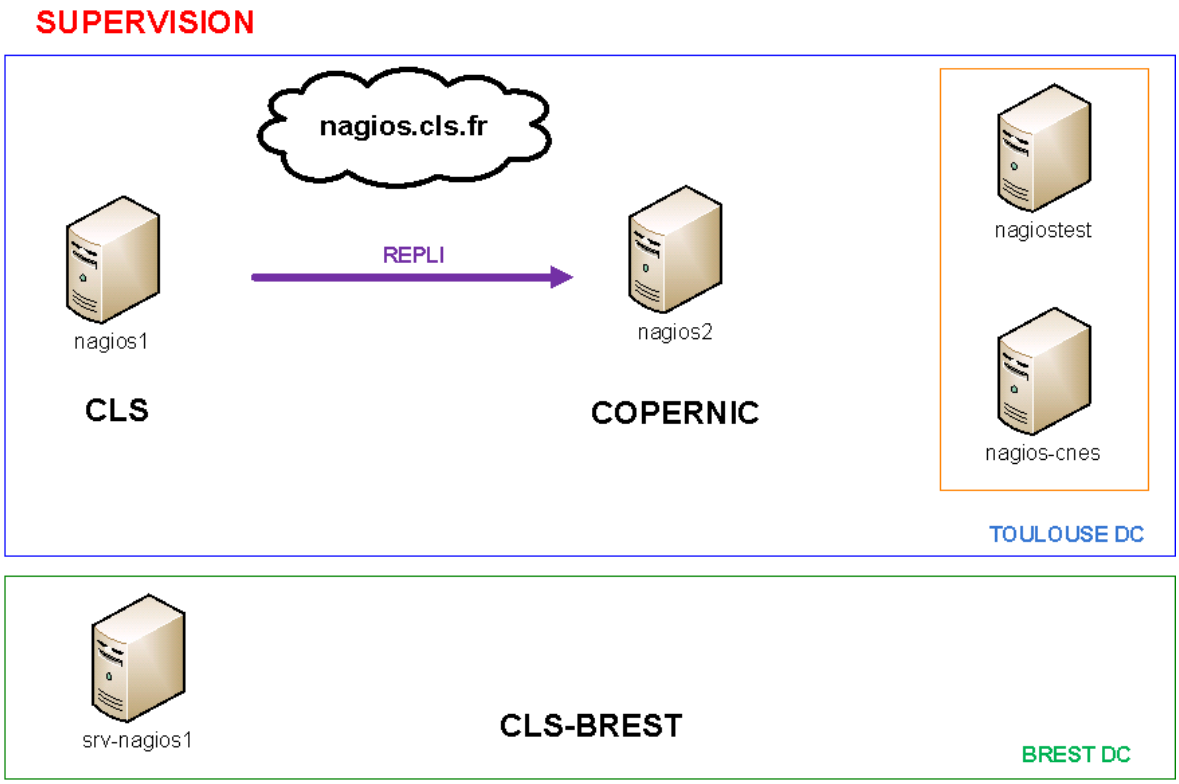

Figure 40 : Supervision de CLS GROUP

- Quatre serveurs (dont trois virtuels) au sein de CLS-TOULOUSE
    - nagios1 et nagios2 pour la PRODUCTION
    - nagiostest pour la QT et QO
    - nagios-cnes utilisé spécifiquement sous un réseau distinct
- Un serveur (virtuel) pour CLS-BREST :
    - srv-nagios1

L'enjeu d'avoir un modèle OVF pour cette solution va être décrit ci-dessous en deux parties. Tout d'abord le gain de temps pour les mises à jour de la solution de notre supervision à CLS. De plus, nous pourrons fournir pour nos applications hébergées chez nos clients (en dehors du DC de CLS), une machine virtuelle de supervision préconfigurée. L'idée est de fournir une application clé en main, supervision comprise. Le modèle OVF va permettre :

- Le déploiement multiple.
- La gestion des versions fournies au client
- Les possibilités d'évolutions avec retour arrière rapide possible

V.4.3.1 Mise à jour

La création du modèle de supervision nous a permis la mise à jour multiple et rapide de notre parc de supervision. Nous sommes partis de l'image disque au format ISO fourni par APX et nous avons suivi la note d'installation initiale.

Le constat est que les mises à jour des serveurs de supervision ont été beaucoup plus rapides que d'habitude. En effet, une fois le modèle déployé afin que notre premier serveur (nagiostest) soit mis à jour, nous avons pu valider les procédures de migration (export et import de la configuration par exemple). Le déploiement pour les autres serveurs a été rapide et sans impact. Ainsi, nagios-cnes et nagios2 ont pu être mis à jour dans la même journée. Par contre, la mise à jour du serveur physique nagios1, a été plus longue et fastidieuse que le déploiement du modèle. En effet, nous avons dû reprendre l'installation depuis le début.

V.4.3.2 Exemple de supervision attachée

La première application « offshore » profitant d'une solution de supervision intégrée est le projet SEAS-OI. La station SEAS-OI est un ensemble de réception satellitaire financé par l'Europe, l'Etat, le Conseil Régional de la Réunion, l'IRD et l'Université de la Réunion. Sa vocation première est d'améliorer dans la région Océan Indien la disponibilité des images satellitaires de type optique (Spot) et radar (Envisat et Radarsat-2) à des fins de recherche, de développements institutionnels et de formation. Cette station, installée dans le sud de la Réunion, permettra de collecter ces images dans un rayon de 2500km couvrant ainsi 12,6 millions de km² d'océan.

Le système (cf figure 41) regroupe les moyens nécessaires pour assurer :

- La définition des images et la commande d'image, ou demande d'acquisition, auprès des opérateurs satellites.

- La réception de la télémesure.
- La production des images et leur validation.
- Leur distribution aux utilisateurs.

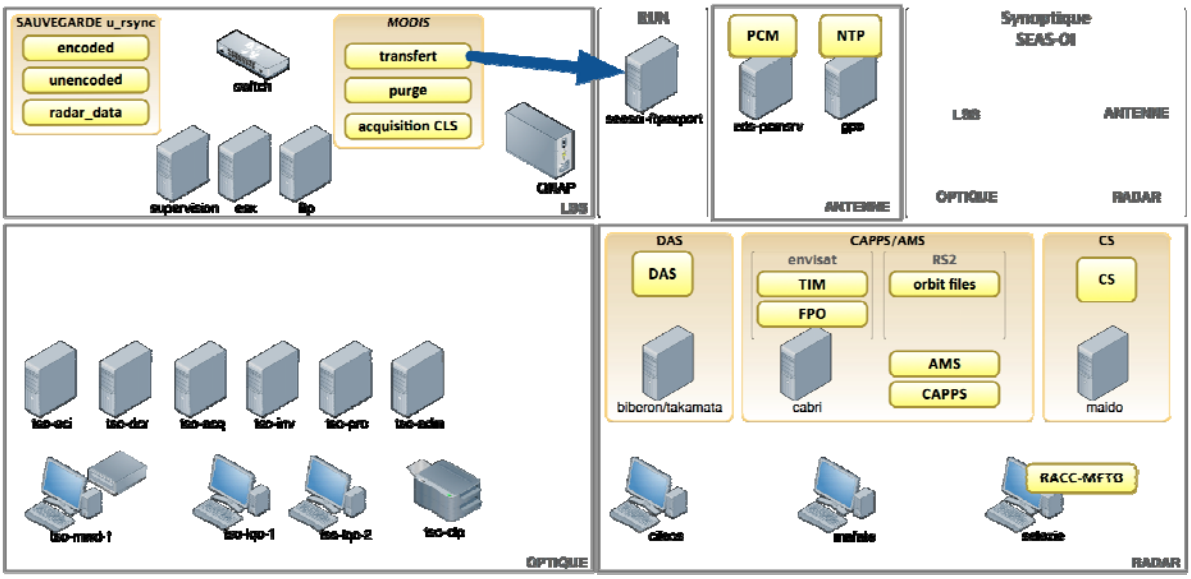

Figure 41 : Architecture SEAS-OI

CLS a assuré la fourniture et l'installation des équipements station et continuera avec le support aux opérations. Pour ma part, j'étais en charge de l'installation et la configuration de l'ESX mais également de la création de la machine virtuelle de supervision, dans le module LSS, depuis le modèle OVF.

VI Bilan du projet

VI.1 Etat actuel du projet

L'utilisation des modèles OVF est en place depuis maintenant plusieurs mois et ce format a été adopté par l'équipe système pour la création de machines virtuelles. Comme vu auparavant, nous disposons à ce jour de plusieurs modèles, par famille avec différentes versions pour chacun.

Nous allons faire un état des lieux quelques mois après l'adoption du projet. Nous ne prenons en compte que les modèles déployés dans notre DC de CLS-Toulouse.

- Les modèles **systèmes** : sur les deux systèmes d'exploitation utilisés à CLS, trois modèles ont été créés en versions majeures. En effet, pour la distribution CentOS, il existe deux modèles dans les deux dernières versions majeures du système d' exploitation (5 et 6). Le déploiement est assez important, comme le montrent les chiffres du tableau 6 et la répartition des modèles, figure 42.

Tableau 6 : Bilan modèles systèmes

Modèle	Versions mineures	Déploiements
CLS_CENTOS5 release 1.x (x86_64)	3	41
CLS_CENTOS6 release 1.x (x86_64)	4	63
CLS_DEBIAN6 release 1.x (x86_64)	5	14
TOTAL	12	118

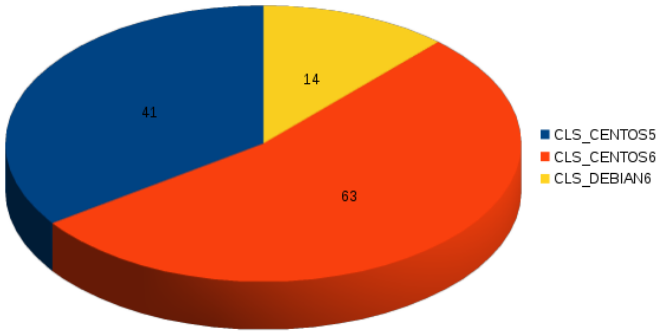

Figure 42 : Répartition modèles systèmes

- Les modèles **applicatifs** : parmi les nombreuses applications exploitées par CLS, quatre modèles ont été créés en versions majeures mais peu ont été déployés (cf Tableau 7). Par contre, comme point positif, les deux applications principales de CLS utilisent les modèles OVF, symbole de l'acception des modèles OVF.

Tableau 7 : Bilan modèles applicatifs

Modèle	Versions mineures	Déploiements
CLS_ARGOS-SERVER release 1.x	3	9
CLS_IRIDIUM-SERVER release 1.x	2	13
CLS_CATSAT-SERVER release 1.x	1	2
CLS_THEMIS-SERVER release 1.x	3	1
TOTAL	9	25

- Les modèles **services** : deux modèles ont été créés (cf Tableau 8). Le plus important est celui de notre solution de supervision qui est déployé à ce jour dans notre infrastructure mais également chez certains de nos clients.

Tableau 8 : Bilan modèles services

Modèle	Versions mineures	Déploiements
CLS_EON release 1.x (EON_2.2)	3	1
CLS_EON release 2.x (EON_3.0)	3	4

CLS_EON release 3.x (EON_3.1)	3	4
CLS_EON release 4.x (EON_4.0)	3	4
CLS_OAE release 1.x (x86_64)	1	8
TOTAL	13	21

VI.2 Bilan

Le constat est que le déploiement des modèles **systèmes** est fort. En effet, 118 machines virtuelles de notre parc informatique à CLS-Toulouse qui en compte environ 400 à ce jour, sont issues des modèles créés lors de ce projet. Ce chiffre valide la pertinence du projet et démontre également la forte évolution de notre système d'information. Le modèle le plus déployé est le modèle CentOS, lequel, à CLS reste dans l'esprit la relève de la distribution historique avant ce projet, à savoir RedHat.

Le faible déploiement des machines virtuelles **applicatives** correspond au fort changement que cela implique : conception plus longue, validation par le développement, l'intégration, l'exploitation et le système. Le déploiement du modèle pour le projet Iridium en est la démonstration.

Les machines virtuelles **services** ont été un bon moyen de mesurer les bénéfices du projet. En effet, notre solution de supervision est majoritairement virtuelle. On a pu migrer trois serveurs dans une nouvelle version en un jour, contre trois jours pour le serveur physique. Le gain de temps et l'assurance d'un système identique démontrent que les objectifs initiaux sont atteints. De plus, de nouvelles possibilités comme le déploiement offshore de notre solution a permis d'élargir le périmètre du projet.

VI.3 Perspectives

Le format OVF est une technologie récente créée grâce aux nouvelles possibilités qu'offre la virtualisation. Cependant, comme nous l'avons vu, le format n'est pas encore mature. De nouvelles possibilités sont à venir, notamment un format ouvert pour le stockage, qui reste un frein à ce jour pour l'inter-operatibilité des modèles entre les différentes solutions de virtualisation. Il existe tout de même des outils permettant la conversion des formats de stockage entre les acteurs mais le gain serait énorme s'il était réalisable nativement.

Une autre ouverture est l'arrivée du Cloud Computing qui est en train de bouleverser les habitudes. En effet, certains éditeurs acceptent maintenant en entrée le format OVF qui doit s'adapter à ces nouveaux besoins. Nous disposons à ce jour de quelques machines virtuelles dans le cloud afin d'effectuer des tests dans les différents continents mais il s'agit de machines « prêtes à l'emploi », vierges de toute modification liées à notre métier. On peut très bien imaginer déployer via la technologie OVF notre serveur de supervision afin d'avoir une vue sur nos équipements publics depuis l'extérieur.

Comme expliqué dans le chapitre 4.1, la version des spécifications du format OVF étudiée dans ce mémoire est la 1.1.0, la 2.1.0 est disponible depuis 2013. Cette version apporte notamment l'amélioration du support de la configuration réseau via l'OVF et une plus grande sécurité (il est possible de chiffrer le contenu d'un paquetage OVF). Je vous propose de consulter l'annexe 4 pour avoir un résumé des nouveautés / possibilités de cette version.

De plus, il serait avantageux d'étudier la combinaison de nos modèles OVF avec un outil de gestion de configuration qui pourrait être complémentaire. De même, des outils de gestion de configuration spécifique à la virtualisation, comme la solution Vagrant, méritent qu'on s'y intéresse.

Enfin, un nouveau chantier s'ouvre : la virtualisation sur deux ans de notre application principale et historique : Argos. Au moment d'écrire ces lignes, la majorité des modules ont été migrés sur les serveurs virtuels. Ceci comprend la montée de version du système d'exploitation et le changement d'architecture vers le 64 bits comme l'application Iridium. Mais la contrainte supplémentaire est que la

configuration de secours se situe à CLS-America, dans le centre de données à Washington. Une fois la configuration migrée et validée dans nos locaux, nous pourrons facilement déployer outre atlantique la configuration de secours grâce au format OVF.

Cette vision « CLS Groupe » pourra être étendue à nos autres centres car, depuis quelques années la société CLS implante des locaux partout dans le monde (Brésil, Indonésie, etc.).

Conclusion

A travers ce mémoire, nous avons montré que le format OVF répond aux besoins des entreprises, notamment dans les grandes lignes de l'industrialisation. En effet, cette technologie permet de pouvoir obtenir des configurations système homogènes, de pouvoir obtenir rapidement la version du modèle déployé et donc d'en connaitre sa configuration. De plus, utiliser des modèles permet un gain de temps pour le déploiement de nouveaux systèmes et donc de leur validation. Néanmoins, à ce jour, le format OVF n'est pas encore arrivé à maturité. De ce fait, son évolution sera intéressante à suivre car elle est liée à la virtualisation, technologie en plein essor qui offre de très grandes possibilités.

En interne à CLS, l'utilisation des modèles au format OVF pour le déploiement de machines virtuelles est désormais ancrée dans les habitudes de l'équipe système. Le gain de temps et la facilité dans le déploiement ont contribué à ce succès. Au-delà des bénéfices interne, nous avons vu que CLS va pouvoir, à moindre coût, tirer les avantages des modèles, avec par exemple son utilisation pour les applications offshores.

A titre personnel, ce mémoire m'a permis d'approfondir un sujet spécifique et de proposer une solution dans un domaine technologique très récent et peu documenté dont toutes les possibilités n'ont pas encore été exploitées. Son application dans un contexte d'opérationnalité très fort a été également une étape importante dans la crédibilité du projet. Ces dernières années au CNAM m'ont permis d'être capable de mener ce type d'étude et de m'épanouir professionnellement.

Bibliographie

Ouvrages imprimés

NAWROCKI Christian. Les Services Informatiques. Stratégie – Alignement – Transformation. DUNOD, 2004, 218 p.

MAILLE Eric. VMware vSphere 4. Mise en place d'une infrastructure virtuelle. ENI Editions, 2010, 419 p.

LOWE Scott. Mastering VMware vSphere 5. SYBEX, 2011, 742 p.

Michael Kresse. Découvrir ITIL v3. Service Management - Pocket book. Serview, 2009, 200p.

Articles de périodiques électroniques

DMTF. *Open Virtualization Format White Paper*. DSP2017, 2009, 1.0.0. Disponible sur : <http://www.dmtf.org/>.

DMTF. *Open Virtualization Format Specification*. DSP0243, 2010, 1.1.0. Disponible sur : <http://www.dmtf.org/>.

DMTF. *Open Virtualization Format Specification*. DSP0243, 2013, 2.1.0. Disponible sur : <http://www.dmtf.org/>.

Sites web

CNRS/ATILF. *Centre National Ressources Textuelles Linguistique*. CNRTL, 2012. Disponible sur : <http://www.cnrtl.fr>.

Wikimedia Foundation. *Wikipédia L'Encyclopédie libre*. WIKIPEDIA, 2012. Disponible sur : <http://fr.wikipedia.org>.

DELBRAYELLE Pascal. *ITIL France*. 2012. Disponible sur : <http://www.itilfrance.com>.

Table des annexes

Annexe 1
Création des paquets CLS-RELEASE

Debian

Création de paquet

Une petite introduction pour présenter l'environnement de gestion des paquets (Dpkg et aptitude)

1. **Création environnement**

apt-get install equivs

2. **Création sources (SOURCES)**

Equivs-control CLS_DEBIAN6

```
Section: misc
Priority: optional
Standards-Version: 3.6.2
Package: cls-debian6
Version: 1.4
Maintainer: CLS <equipe.systeme@cls.fr>
Architecture: all
Files: cls-release /etc/cls-release
Description: /etc/cls-release contient la version du template OVF
```

3. **Creation DEV (RPMS)**

equivs-build CLS_DEBIAN6

CentOS

Une petite introduction pour présenter l'environnement de gestion des paquets (yum et rpm)

4. **Création environnement**

   - Installer le package rpm-build

   - [root@template-centos6 ~]# ls rpmbuild/

     BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

5. **Création sources (SOURCES)**

- Création de /etc/cls–release est :

```
NOM release num_version (libre)
```

Exemple : CLS_CENTOS6 release 1.2 (x86_64)

- tar cvf template.tar /etc/cls-release
- gzip template.tar

6. Création .spec (SPECS)

```
[root@template-centos6 SPECS]# cat CLS_CENTOS6.spec
Summary: Version du template OVF
Name: CLS_CENTOS6
Version: 1.2
Release: 1
License: GPL
Source: CLS_CENTOS6_12.tar.gz
#NoSource: 0
BuildRoot: %{_tmppath}/%{name}
Vendor: CLS
# Following are optional fields
#URL: http://www.example.net/tmp/
#Distribution: Red Hat Contrib-Net
#Patch: tmp-%{version}.patch
#Prefix: /usr/local
BuildArch: x86_64
#Requires: mondo >= 2.2.7
#Obsoletes:
#BuildRequires:

%description
/etc/cls-release contient la version du template OVF

%prep
%setup -c tmp
#%patch

%install
%__cp -a . "${RPM_BUILD_ROOT-/}"

%clean
[ "$RPM_BUILD_ROOT" != "/" ] && rm -rf "$RPM_BUILD_ROOT"

%files
%defattr(-,root,root)
/etc/cls-release
%config

%changelog
* Thu Oct 23 2008 root <root@vrssi01t>
- hh
```

7. Creation RPM (RPMS)

rpmbuild -bb CLS_CENTOS5.spec

Annexe 2
Notes d'installation du template CentOS 6

Création de la VM de base

Configuration VMWare

```
RAM : 512 Mo
1 vCPU
HDD1 : 10Go ( / et swap )
Réseau : Reseau-bleu
```

Déploiement

Installation minimale à partir de l'iso « CentOS-6.0-x86_64-bin-DVD».

Configuration Système

Post install

Mise à jour du système

```
yum update
```

Paquets supplémentaires

Ces paquets sont nécessaires étant des pré-requis aux installations suivantes (tsm, vmware-tools, ocs, nagios, etc…)

```
yum install compat-libstdc++-33.x86_64
yum install openssh-clients
yum install xinetd
yum install perl-CPAN
yum install ftp
yum install dmidecode
yum install mlocate
yum install rsync
yum install gcc
yum install kernel-devel
yum install make
yum install wget
```

Suppression services

```
chkconfig --del auditd
```

Clé ssh

Ajout des clés (ex : tsm)

```
authorized keys
```

Repos CLS

Création du fichier /etc/yum.repos.d/CentOS-CLS.repo

```
[base]
name=CentOS-$releasever - Base
baseurl=http://centos.cls.fr/$releasever/os/$basearch/

# updates
[update]
name=CentOS-$releasever - Updates
baseurl=http://centos.cls.fr/$releasever/updates/$basearch/

# Extras
[extras]
name=CentOS-$releasever - Extras
baseurl=http://centos.cls.fr/$releasever/extras/$basearch/
```

SNMP

Installation des paquets

```
yum install net-snmp net-snmp-libsnet-snmp-utils
```

Modification du fichier : /etc/snmp/snmpd.conf

```
 rocommunity et syscontact et syslocation
```

Modifier également /etc/init.d/snmp pour baisser le niveau de log :

```
OPTIONS="-LS0-6d -Lf /dev/null -p /var/run/snmpd.pid"
en
OPTIONS="-LS0-4d -Lf /dev/null -p /var/run/snmpd.pid"
```

Démarrage auto :

```
chkconfig snmpd on
```

Date

NTP

Installation du paquet

```
yum install ntp
```

Modification du /etc/ntp.conf :

```
server date.cls.fr prefer
server date2.cls.fr
```

Démarrage auto :

```
chkconfig ntpd on
```

Localtime

```
cp /usr/share/zoneinfo/GMT /etc/localtime
```

modifier /etc/sysconfig/clock :

```
ZONE="Etc/GMT"
UTC=True
ARC=false
```

Locale

Modification du fichier /etc/sysconfig/i18n

```
LANG="en_US.UTF-8"
SUPPORTED="en_US.UTF-8:en_US:en:fr_FR.UTF-8:fr_FR:fr"
SYSFONT="latarcyrheb-sun16"
```

Postfix

Modification du fichier /etc/postfix/main.cf

```
relayhost = mail.cls.fr
```

Firewall

SELinux

Modification du fichier /etc/selinux/config :

```
SELINUX=disabled
```

IPtables

Suppression du démarrage :

```
chkconfig --del ip6tables
chkconfig --del iptables
```

Nagios

Création compte :

```
groupadd -g 598 nagios
useradd -u 617 -d /usr/local/nagios -g nagios nagios
passwd nagios
```

Création arborescence :

```
mkdir -p /usr/local/nagios
```

Installation nagios-plugins :

```
tar xvzf nagios-plugins-1.4.15.tar.gz
./configure --prefix=/usr/local/nagios --with-nagios-user=nagios --with-nagios-group=nagios
 make all
```

```
make install
```

Installation de l'agent nrpe :

```
./configure --prefix=/usr/local/nagios --with-nrpe-user=nagios --with-nrpe-group=nagios --enable-command-args --disable-ssl
make all
mkdir /usr/local/nagios/bin
cp src/nrpe /usr/local/nagios/bin/
cp nrpe.cfg /usr/local/nagios/etc
```

Configuration nrpe / xinetd

```
cp nrpe /etc/xinetd.d/
vi /etc/services
service xinetd reload
```

On remet les droits :

```
chown -Rv nagios:nagios /usr/local/nagios/
```

OCS Inventory

Installation des parquets cpan :

```
Net ::SSLeay
Net ::IP
XML ::Simple
XML::Parser
Lib WWW Perl
```

Installation de l'agent :

```
perl Makefile.PL
make
make install
```

Vmware Tools

```
./vmware-install.pl
```

Configuration réseau

Modification du fichier /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE="eth0"
BOOTPROTO=dhcp #static
#IPADDR=X.X.X.X
#NETMASK=255.255.0.0
#BROADCAST=X.X.255.255
#NETWORK=X.X.0.0
NM_CONTROLLED="yes"
ONBOOT="yes"
```

Modification du fichier /etc/sysconfig/network

```
NETWORKING=yes
NETWORKING_IPV6=no
```

```
HOSTNAME=template-centos6
#GATEWAY=X.X.X.X
```

Modification du fichier /etc/hostname

```
template-centos6.cls.fr
```

Template

Création du fichier /etc/motd

```
        ##########################################
        #         TEMPLATE CENTOS 6              #
        #                                        #
        # Pour installer ce template veuillez    #
        # lire le fichier suivant :              #
        # /root/INSTALL_README.txt               #
        #                                        #
        ##########################################

(c) CLS 2011. DT/SI
```

Création du fichier /root/INSTALL_README.txt

```
-------------------------------------------------
What to do à l'install du template centos minimal
-------------------------------------------------

Changer le nom de la VM dans les fichiers suivants:
---------------------------------------------------
        vi /etc/sysconfig/network
        vi /etc/hostname
        vi /etc/hosts

Modifier la configuration reseau dans:
--------------------------------
        vi /etc/sysconfig/network-scripts/ifcfg-eth0
        vi /etc/sysconfig/network

Modifier le Tag OCS dans le fichier:
------------------------------------
        vi /etc/ocsinventory-agent/ocsinventory-agent.cfg

Modifier l'heure de la cron (pour pas que toutes les VM se mettent a jour en meme temps):
-----------------------------------------------------------------------------------------
        vi /etc/cron.d/ocsinventory-agent

Pour mettre en place une rotation des logs prendre exemple sur :
----------------------------------------------------------------
        /etc/logrotate.d/syslog


Regenerer une clé ssh:
----------------------
        ssh-keygen -t dsa

Enlever le Message a l'entrée:
------------------------------
        rm /etc/motd
```

Création du fichier /etc/cls-release

```
CLS_CENTOS6 release 1.0 (x86_64)
```

Détails

Nombre de paquets : 243

```
perl-version-0.77-115.el6.x86_64
setup-2.8.14-10.el6.noarch
perl-Pod-Simple-3.13-115.el6.x86_64
basesystem-10.0-4.el6.noarch
perl-5.10.1-115.el6.x86_64
ncurses-base-5.7-3.20090208.el6.x86_64
net-snmp-5.5-27.el6_0.1.x86_64
ntpdate-4.2.4p8-2.el6.x86_64
ntp-4.2.4p8-2.el6.x86_64
bash-4.1.2-3.el6.x86_64
ppl-0.10.2-11.el6.x86_64
info-4.13a-8.el6.x86_64
mpfr-2.4.1-6.el6.x86_64
popt-1.13-7.el6.x86_64
libgomp-4.4.4-13.el6.x86_64
libcom_err-1.41.12-3.el6.x86_64
glibc-headers-2.12-1.7.el6_0.5.x86_64
libsepol-2.0.41-3.el6.x86_64
gcc-4.4.4-13.el6.x86_64
glib2-2.22.5-5.el6.x86_64
make-3.81-19.el6.x86_64
shadow-utils-4.1.4.2-8.el6.x86_64
xinetd-2.3.14-29.el6.x86_64
nspr-4.8.6-1.el6.x86_64
perl-ExtUtils-ParseXS-2.2003.0-115.el6.x86_64
perl-ExtUtils-MakeMaker-6.55-115.el6.x86_64
libattr-2.4.44-4.el6.x86_64
ftp-0.17-51.1.el6.x86_64
…
```

Post Install pour l'OVF 1.1

Voici les modifications à effectuer lors de la création de la prochaine version du template :

Annexe 3
Notes d'installation du template Iridium

Création de la VM de base

Déploiement

Déploiement à partir du template OVF « CLS_CENTOS5 release 1.2 (x86_64) ».

Configuration VMWare

```
RAM : 4 Go (réservé)
2 vCPU (dont 4Ghz réservé)
Destination: esx-dev5 / Baie EVA7
HDD1 : 10Go ( / et swap )
HDD2 : 70Go ( /opt/iridium )
Réseau : Reseau-bleu
```

Configuration Système

Configuration réseau et 1er démarrage

```
/etc/sysconfig/network
/etc/hostname
/etc/hosts
/etc/sysconfig/network-scripts/ifcfg-eth0

/etc/ocsinventory-agent/ocsinventory-agent.cfg

ssh-keygen -t dsa
```

Création du HDD2 (/opt/iridium)

Clone du sdb d'iridium-qt (/opt/iridium):

```
vmkfstools –I
```

Ajout dans le fstab :

```
LABEL=/iridium /opt/iridium ext3 defaults,acl 1 2
```

Création des comptes

```
grep ^i /etc/shadow
grep ^i /etc/passwd
```

+ajout des comptes iridium et iriftp dans /etc/group

visudo

```
visudo
```

Ajout des lignes

```
# IRIDIUM
i_root ALL = NOPASSWD : /opt/iridium/scripts/vip*
i_root ALL = NOPASSWD : /opt/iridium/scripts/TARcreate.sh
i_root ALL = NOPASSWD : /opt/iridium/scripts/start_SFTP.sh
i_root ALL = NOPASSWD : /opt/iridium/scripts/start_FTP.sh
i_root ALL = NOPASSWD : /bin/chown * /opt/iridium/*
i_root ALL = NOPASSWD : /bin/chgrp * /opt/iridium/*
i_root ALL = NOPASSWD : /bin/chmod * /opt/iridium/*
i_root ALL = NOPASSWD : /usr/bin/setfacl * /opt/iridium/*
i_root ALL = NOPASSWD : /sbin/shutdown
i_root ALL = NOPASSWD : /sbin/reboot
i_root ALL = NOPASSWD : /sbin/service
i_root ALL = NOPASSWD : /bin/su - i_*
# i_root peut executer n importe quelle commande de n importe quel user du groupe iridium
i_root ALL = (%iridium) NOPASSWD : ALL
# i_hal peut executer n importe quelle commande de n importe quel user du groupe iridium
i_hal ALL = (%iridium) NOPASSWD : ALL
```

init.d

```
scp /etc/init.d/iridium root@iridium-qt-64:/etc/init.d/iridium
chkconfig iridium on
```

Httpd

```
yum install httpd
```

Rsh

```
yum install rsh rsh-server
chkconfig rsh on
chkconfig rexec on
```

nano

```
yum install nano
```

FTP

vsftpd

```
yum install vsftpd
chkconfig --level 345 vsftpd  on
chkconfig --list vsftpd
service vsftpd start
```

Proftpd et Sftp

Pré-requis

```
yum install gcc-c++.x86_64
```

```
cd /tmp/
wget --passive-ftp ftp://ftp.proftpd.org/distrib/source/proftpd-1.3.2.tar.gz
tar xvfz proftpd-1.3.2.tar.gz
cd proftpd-1.3.2/
./configure --sysconfdir=/etc
make
make install
cd ..
rm -fr proftpd-1.3.2
```

```
ln -s /usr/local/sbin/proftpd /usr/sbin/proftpd
```

Ajout du groupe iriftp

```
# groupadd –g 621 iriftp
```

Mettre en place des ACLs sur /opt/iridium/TRM

Modifier /etc/fstab :

```
LABEL=/iridium        /opt/iridium            ext3    defaults,acl     1 2
```

Attention, sur iridium-trm, /opt/iridium/TRM est un FS spécifique.
Puis démonter / remonter le FS /opt/iridium

```
mount -o acl,remount /opt/iridium
```

Modifier /etc/shells

Modifier /etc/shells pour y ajouter

```
/bin/ftponly
```

Modifier limits.conf

Modifier le contenu de /etc/security/limits.conf :

```
i_trm         hard    nofile          2048
```

```
i_trm        soft    nofile          2048
```

LOGS

Création d'un répertoire dédié

Créer le répertoire pour les logs sur le serveur cible :

```
mkdir –p /var/log/proftpd
```

Modification syslogd

Modifier /etc/sysconfig/syslog

```
SYSLOGD_OPTIONS="-m 0 -a /data/chroot/dev/log"
```

Relancer :

```
service syslog restart
```

Logrotate

Créer le /etc/logrotate.d/proftpd :

```
/var/log/proftpd/xferlog {
    compress
    missingok
    postrotate
            /usr/bin/killall -HUP proftpd
    endscript
}

/var/log/proftpd/proftpd.syslog /var/log/proftpd/*.log {
    compress
    missingok
    postrotate
       /usr/bin/killall -HUP proftpd
    endscript
}
```

Ajouter le demarrage dans /etc/init.d/iridium

Modifier /etc/init.d/iridium pour rajouter dans start_TRM() les lignes

```
[ -x /opt/iridium/scripts/start_SFTP.sh] &&  /opt/iridium/scripts/start_SFTP.sh start
[-x /opt/iridium/scripts/start_FTP.sh ]&& /opt/iridium/scripts/start_FTP.sh start
```

Pour cela, il faut installer la version 1.5 de /etc/init.d/iridium (dans CVS/Iridium/iridium.sh)

Messagerie

Installation des paquetages

```
yum install cyrus-imapd.x86_64 cyrus-imapd-utils.x86_64 cyrus-sasl-md5.x86_64
```

Adaptation du système Linux

Modifier le mot de passe de l'utilisateur cyrus :

```
passwd cyrus (mot de passe cyrus)
cp /etc/aliases /etc/postfix/aliases
postalias /etc/postfix/aliases
chkconfig --add postfix
chkconfig --level 2345 cyrus-imapd on
chkconfig --level 2345 saslauthd on
```

Adaptation pour que n'importe quel email se terminant par @thorium.cls.fr confié à ce serveur soit déposé dans une seule BAL :
Modifier /etc/postfix/main.cf :

```
myhostname = iridium-qt-64.cls.fr
mydomain = thorium.cls.fr
myorigin = $myhostname
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain, clsamerica.com
virtual_alias_domains = iridium.cls.fr
virtual_alias_maps = hash:/etc/postfix/virtual, regexp:/etc/postfix/virtual.regexp
local_recipient_maps =
unknown_local_recipient_reject_code = 550
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mailbox_transport = cyrus
fallback_transport = cyrus
```

Modifier /etc/postfix/virtual.regexp :

```
#
# Toutes les adresses @thorium.cls.fr
# vont dans la BAL i_smtp
/^([-a-zA-Z0-9_.]+)@thorium.cls.fr$/ i_smtp
/^([-a-zA-Z0-9_.]+)@thorium.clsamerica.com$/ i_smtp
#
# idem pour i_smtp-dev
/^([-a-zA-Z0-9_.]+)@dev.thorium.cls.fr$/ i_smtp-dev
/^([-a-zA-Z0-9_.]+)@dev.thorium.clsamerica.com$/ i_smtp-dev
#
# idem pour i_smtp-qt
/^([-a-zA-Z0-9_.]+)@qt.thorium.cls.fr$/ i_smtp-qt
/^([-a-zA-Z0-9_.]+)@qt.thorium.clsamerica.com$/ i_smtp-qt
#
# idem pour i_smtp-qo
/^([-a-zA-Z0-9_.]+)@qo.thorium.cls.fr$/ i_smtp-qo
/^([-a-zA-Z0-9_.]+)@qo.thorium.clsamerica.com$/ i_smtp-qo
#
# Adresses "classiques" en @iridium.cls.fr
/^sbdmail@iridium.cls.fr$/ sbdmail
/^sbdmail-dev@iridium.cls.fr$/ sbdmail-dev
/^sbdmail-qt@iridium.cls.fr$/ sbdmail-qt
/^sbdmail-qo@iridium.cls.fr$/ sbdmail-qo
```

Attention à exécuter postmap /etc/postfix/virtual.regexp après chaque modification du fichier /etc/postfix/virtual (sinon, le .db n'est pas mis à jour et du coup postfix utilise l'ancienne version). Et un restart de postfix n'y change rien...
Idem pour le fichier /etc/postfix/virtual (et la commande postmap /etc/postfix/virtual).

Démarrage des services :

```
service postfix start
service cyrus-imapd start
```

```
service saslauthd  start
```

Paquets supp

GPG

```
.configure
make
make install
```

```
gpg: WARNING: unsafe permissions on configuration file `/opt/iridium/TRM/i_trm/.gnupg/gpg.conf'
gpg: WARNING: unsafe enclosing directory permissions on configuration file
`/opt/iridium/TRM/i_trm/.gnupg/gpg.conf'
```

Modif des droits du fichier de conf :

```
[i_trm@iridium-qt ~]$ chmod 600 /opt/iridium/TRM/i_trm/.gnupg/gpg.conf
[i_trm@iridium-qt ~]$ chmod 700 /opt/iridium/TRM/i_trm/.gnupg
```

```
i_trm@iridium-qt ~]$ /opt/iridium/TRM/config/spears/encrypt.sh 0cerbere1 spears.test@iridium.com
/opt/iridium/TRM/data/trm/spears-2/working/SPEAR1-IST-CLSAT-20110414085449-0018927.cmd
gpg: Sorry, no terminal at all requested - can't get input
```

Modification des scripts, apparemment l'option tty n'existe plus (gpg --help)
qo : gpg (GnuPG) 1.2.6
qt centos : gpg (GnuPG) 1.4.11

-> /opt/iridium/TRM/config/spears/encrypt.sh

```
echo $1 | gpg --always-trust --passphrase-fd 0  --sign --encrypt  --no-tty  -r $2 $3
```

modification du script en :

```
echo $1 | gpg --always-trust --passphrase-fd 0  --sign --encrypt -r $2 $3
```

Création du Template OVF 1.0

/opt/iridium

Désactivation du montage de /opt/iridium

```
vi /etc/fstab
```

Services

Désactivation de certains services

```
chkconfig iptables off
chkconfig nfslock off
chkconfig rpcgssd off
```

Annexe 4
Spécifications du format OVF 2.1.0

La spécification OVF 1.0 a fourni un standard pour le conditionnement et la distribution de machines virtuelles permettant pour répondre à un besoin des fournisseurs de logiciels. Nous avons vu dans le chapitre 4.1 que le format OVF se veut être un format ouvert, sécurisé, portable et extensible. OVF 1.0 a été largement adoptée par l'industrie.

OVF 2.0 est une évolution logique, qui apporte un ensemble amélioré de fonctions, rendant son utilisation applicable en cas d'utilisation dans le cloud computing. Les améliorations les plus importantes incluent une amélioration du support de la configuration réseau via l'OVF et une plus grande sécurité (il est possible de chiffrer le contenu d'un paquetage OVF). Les nouvelles fonctionnalités :

- Amélioration du support pour la configuration réseau
- Cryptage de paquet pour une livraison sûre
- Nouvelles options de déploiement
- Disques partagés
- Ordre de démarrage des équipements
- Mécanismes avancés pour transmettre les données à l'hôte
- Mise à jour des schémas CIM

Liste des figures et tableaux

Glossaire

ANSI	American National Standards Institute
CIM	Common Information Model
CLS	Collecte Localisation Satellites
CNAM	Conservatoire National des Art et Métiers
CNES	Centre National d'Études Spatiales
CNRTL	Centre National de Ressources Textuelles et Lexicales
CMDB	Configuration Management DataBase
CST	Centre Spacial de Toulouse
DC	Data Center
DAR	Direction Radar
DCL	Direction Collecte et Localisation
DF	Direction Financière
DG	Direction Générale
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
DMTF	Distributed Management Task Force
DO	Direction des Opérations
DOS	Direction Océanographique Spatiale
DPM	Distributed Power Management
DRHC	Direction des Ressources Humaine et Communication
DRS	Distributed Resource Scheduler
DSI	Division Système Information
DT	Direction Technique
ESA	European Space Agency
FT	Fault Tolerance
HA	High Availability
IFREMER	Institut Français de Recherche pour l'Exploitation de la Mer
ITIL	Information Technology Infrastructure Library
LDAP	Lightweight Directory Access Protocol
LRIT	Long Range Identification & Tracking
LVM	Logical Volume Management
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
NTP	Network Time Protocol
OS	Operating System
OVF	Open Virtualization Format
QO	Qualification Opérationnelle
QT	Qualification Technique
RDP	Rapid Deployment Package
RPM	RPM Package Manager
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol

SMP	Symmetric shared Memory multiProcessor
SSH	Secure Shell
UTC	*Temps universel coordonné*
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VAR	Value Added Reseller
VHV	Virtual Hardware Version
VPN	Virtual Private Network
XML	Extensible Markup Language

Industrialisation du déploiement de machines virtuelles

Mémoire d'Ingénieur C.N.A.M., Toulouse 2013

RESUME

La société CLS est spécialisée dans la localisation et la collecte de données par satellite, l'observation et la surveillance des océans. Pour répondre à ces activités, elle met en place pour ses clients des applications avec des engagements de disponibilités 24h/24, 7j/7. Au sein de la direction technique, la division informatique définit, conçoit et assure le bon fonctionnement des infrastructures hébergeant ces applications.

Afin d'améliorer la disponibilité du système d'information et la qualité des infrastructures, majoritairement virtuelles, la division informatique souhaite industrialiser ses activités. Aussi, des problématiques concernant la gestion et le déploiement de machines virtuelles se posent. Les objectifs principaux de cette étude sont d'homogénéiser le déploiement, rendre la mise à disposition d'environnement rapide et efficace, tout en garantissant leur redéploiement à l'identique au fil du temps.

Ce mémoire traite de la manière dont l'industrialisation du déploiement de machines virtuelles a été mise en œuvre, de l'analyse des besoins à la réalisation technique, en décrivant chacune des étapes du projet.

Mots clés : Virtualisation, VMWare, déploiement, industrialisation, ITIL, OVF.

SUMMARY

CLS offers satellite services in location, environmental data collection and ocean observations. CLS set up applications with availabilities 24/7 for its customers to respond the best to such activities. Within the technical direction, the IT department defines, designs and ensures the good operating of infrastructures which host these applications.

In order to improve the availability of the information system, mostly virtual, and the quality of the infrastructures, the IT department wishes to industrialize these activities. Therefore, there can be issues concerning the management and the deployment of virtual machines. The main objectives of this study are to make the deployment coherent, fast and effective while guarantying it identically and throughout the years.

This report deals with the way industrialization of virtual machines deployment was applied, the analysis of the needs to technical realization while describing each step of the project.

Key words : Virtualization, VMWare, deployment, industrialization, ITIL, OVF.