Rapport
« Cloud privé avec OpenStack »

Julien Brun, Maxime Mouchet
Fréderic Pourraz (tuteur)

RT2 2012-13

Remerciements

Nous remercions M. Frédéric Pourraz pour nous avoir permis de réaliser ce projet.
Nous remercions M. Gaëtan Ferez pour nous avoir mis à disposition le matériel et son aide sur celui-ci.
De manière générale nous remercions tous les professeurs pour leurs enseignements durant ces deux années de DUT.

Sommaire

1 Introduction **4**
- 1.1 Choix du sujet . . . 4
- 1.2 Contexte . . . 4
- 1.3 Rappel du cahier des charges . . . 6
- 1.4 Démarche . . . 6

2 Matériel **7**
- 2.1 Serveur dédié OVH . . . 7
- 2.2 BladeCenter HP . . . 7

3 Logiciels **8**
- 3.1 OpenStack . . . 8
- 3.2 Automatisation . . . 9
- 3.3 Haute disponibilité . . . 9
- 3.4 Reverse Proxy . . . 9
- 3.5 Supervision . . . 10

4 Mise en place **11**
- 4.1 Installation de développement sur un serveur dédié . . . 11
- 4.2 Déploiement sur un BladeCenter avec Solaris . . . 11
- 4.3 Déploiement sur un BladeCenter avec CentOS . . . 12

5 Conclusion **16**

A Utilisation du Cloud privé **17**
- A.1 Généralités . . . 17
- A.2 Création d'une instance virtuelle . . . 18

B Administration du Cloud privé **20**
- B.1 Gestion avec le Dashboard . . . 20
- B.2 Gestion en ligne de commande . . . 20
- B.3 Dépannage à l'aide des logs . . . 20

C Installation des systèmes d'exploitation sur les lames **21**
- C.1 Systèmes d'exploitation supportés . . . 21
- C.2 Integrated Lights-Out . . . 21
- C.3 Installation de CentOS . . . 22
- C.4 Installation de Solaris . . . 25

D Diagrammes Réseau et schémas de principe **26**
- D.1 Les différents services d'OpenStack . . . 26
- D.2 Schéma global BladeCenter avec Solaris . . . 27
- D.3 Schéma global BladeCenter avec QEMU et GlusterFS . . . 28

E Scripts d'automatisation **29**
- E.1 Exemple de configuration de MediaWiki avec Juju . . . 29
- E.2 Exemple de configuration avec Puppet . . . 31

F Fichiers de configuration OpenStack **33**
- F.1 Keystone . . . 33
- F.2 Glance . . . 34
- F.3 Nova . . . 36

1 Introduction

1.1 Choix du sujet

La virtualisation et le Cloud Computing sont des solutions d'avenir pour les entreprises grâce à des coûts réduits, une maintenance simplifiée, et une montée en charge aisée. Nous trouvons ces technologies passionnantes et mettre en place un Cloud privé nous permet de les aborder tout en approfondissant les thématiques étudiées durant notre formation.

1.2 Contexte

1.2.1 La virtualisation

Le principe de base de la virtualisation est de permettre le fonctionnement de plusieurs systèmes d'exploitation (Windows, Linux, ...) en simultané sur un ordinateur. Les intérêts sont multiples, on peut citer principalement :

- Augmenter la disponibilité grâce à une redondance et réduire les coûts. Par exemple on peut imaginer avoir deux serveurs mails sur une même machine physique et utiliser l'un ou l'autre en fonction de leur charge ou en cas de panne. On réduit par la même occasion les coûts puisqu'on n'utilisera qu'une seule machine physique pour ces deux serveurs.

- Un dimensionnement plus facile. La virtualisation ajoutant une couche d'abstraction entre le système d'exploitation (logiciel) et le matériel (ordinateur) on peut aisément augmenter la puissance de calcul en ajoutant, par exemple, de la mémoire sur un ordinateur sans impacter le fonctionnement des systèmes virtuels.

- Faciliter les migrations site-à-site. Il est beaucoup plus facile de transférer un serveur d'un site physique à un autre en transférant une image virtuelle par le réseau plutôt qu'en transportant un serveur physique.

Il existe plusieurs logiciels permettant de virtualiser des systèmes, parmi les plus connus on pourra citer Parallels Desktop ou VirtualBox.

1.2.2 Le Cloud Computing

Vue d'ensemble

Ces logiciels conviennent à la virtualisation de quelques machines mais pas à la gestion globale d'une infrastructure, ils se contentent de faire fonctionner plusieurs systèmes d'exploitation sur un seul ordinateur. Ils ne permettent pas d'utiliser des ressources et du stockage présents sur plusieurs ordinateurs depuis un lieu distant et par plusieurs personnes.

On va alors regrouper toutes ces ressources (stockage, ordinateurs, logiciels de virtualisation) via un réseau. On parle de Cloud Computing et plus précisément [1] dans ce cas là d'Infrastructure as a Service (IaaS). L'utilisateur final ne se préoccupe pas de savoir où est physiquement le stockage ou les ordinateurs qui possèdent le logiciel de virtualisation, ni comment ils sont reliés entre eux. Il indique juste au cloud qu'il souhaite tant de machines avec tel système d'exploitation.

IaaS, PaaS, et autres *aaS

Dans le domaine du Cloud Computing il existe plusieurs niveaux d'abstraction, offrant des fonctionnalités plus ou moins haut niveau, on parle de modèles de services. Leur nom prend la forme Sth- as a Service.

Les trois modèles les plus courants sont l'Infrastructure as a Service (IaaS), le Platform as a Service (PaaS), et le Software as a Service (SaaS). Ce dernier représente le plus haut niveau d'abstraction.

1. Voir « IaaS, PaaS, et autres *aaS »ci-dessous.

Dans ce modèle les clients paient un abonnement pour utiliser un logiciel dont ils n'auront pas à gérer l'installation et la maintenance. Les applications courantes sont les Customer Relationship Management (CRM), la visioconférence, et les emails. On peut citer pour exemple Google Apps for Business ou Salesforce Sales Cloud.

Vient ensuite le PaaS, où les applications fonctionnent également sur une plateforme externalisée (dans le Cloud) mais, à la différence du SaaS, elles sont développés par le client.
Cela permet aux entreprises de disposer d'un environnement d'exécution pour leur applications internes rapidement. Par exemple, Heroku, le PaaS de Salesforce est capable d'exécuter de nombreux langages, tels que PHP, Ruby, Node.js, Python, Java, ou Scala. On peut aussi citer Google App Engine ou Microsoft Azure, qui offrent des fonctionnalités similaires.

Enfin vient l'IaaS, qui est le plus bas niveau d'abstraction. Ici le client dispose d'une infrastructure, mise à disposition par le fournisseur de Cloud, où il peut créer des serveurs virtuels à la demande, sans se préoccuper du matériel physique. Il gère donc la configuration des systèmes d'exploitation et des logiciels qui s'y exécutent.

C'est ce dernier modèle que nous avons choisi d'étudier, nous allons donc détailler, dans la section suivante (1.2.3), les différentes solutions existantes avant de voir celle que nous avons retenu.

Le stockage dans le cloud

Dans le cas des IaaS on distingue deux types de stockages dans le cloud : le stockage bloc, et le stockage objet. Le premier est proche du matériel physique, limitant les couches d'abstractions, et offre des performances de haut-niveau. Il est exposé aux instances virtuelles comme un disque physique natif, ce qui permet au système d'exploitation de démarrer dessus.
Le stockage objet, au contraire, est dit « haut-niveau »car il est accessible via une Application Programming Interface (API) et l'utilisateur final ne se préoccupe pas de savoir où, ni comment, ses données seront stockés. Il est généralement plus lent que le stockage bloc mais disponible en plus grande quantité et redondé, il est donc utilisé pour les données qui changent peu (images de machines virtuelles, sauvegardes, photos, ...).

Cloud public vs. Cloud privé

La différence majeure entre un Cloud dit « public », d'un Cloud dit « privé »est le partage de l'infrastructure. Dans le premier cas plusieurs clients se partagent l'infrastructure, tandis que dans le second seul une entreprise utilise l'infrastructure.
Un Cloud public sera toujours situé en dehors de l'entreprise, alors qu'un Cloud privé pourra être installé dans l'entreprise et géré par elle-même. Il en résulte alors un contrôle complet de l'infrastructure et une sécurité accrue.

1.2.3 Panorama des IaaS

Amazon EC2

Amazon a lancé Elastic Compute Cloud (EC2) durant l'été 2006. Il s'agit d'un Cloud public, au sens où l'infrastructure est commune à tout les clients, et accessible à travers une API et une interface Web, moyennant un paiement par heures d'utilisation.

Les instances, allant de 1 à 244Go de RAM et de 1 coeur virtuel à 32 coeurs physiques, sont virtualisés grâce à l'hyperviseur Open Source Xen. EC2 supporte Linux, *BSD, Solaris, et Windows comme systèmes d'exploitation invités.

Le stockage bloc est fourni par Amazon Elastic Block Store (EBS) tandis que le stockage objet est fourni par Amazon Simple Storage Service (S3). Chacun de ces services possède sa propre API, permettant de les utiliser indépendamment les uns des autres.

A titre informatif, EC2 est utilisé, entre autres, par Netflix [2], Instagram, et Shazam, tandis que la plupart des services de stockage en ligne grand public, comme Dropbox, utilisent S3.

Il faut noter que l'IaaS d'Amazon est propriétaire et spécifique à leur infrastructure. Il n'est pas possible de la déployer sur son propre matériel.

CloudStack

CloudStack est un système de création d'IaaS originellement développé par Cloud.com puis racheté par Citrix à l'été 2011. Il appartient désormais à la fondation Apache et est donc Open Source sous la licence correspondante [3].
CloudStack est développé en Java.

2. En 2011, aux heures de pointe, Netflix était responsable de 29% du trafic Internet en Amérique du Nord [10]. D'où la nécessité d'une infrastructure robuste et scalable.
3. La licence Apache

OpenStack

OpenStack a été lancé en Juillet 2010 par Rackspace et la NASA. Son but est de proposer un outil gratuit et libre de création d'IaaS. Le code source du projet (du Python) est disponible sur GitHub [4] et Launchpad [5] sous licence Apache 2.0.
Outre ses deux fondateurs OpenStack est à l'heure actuelle principalement utilisé par HP, AT&T, Intel, et Canonical.

1.3 Rappel du cahier des charges

Objectifs

Dans un premier temps nous voulions mettre en place un Cloud privé sous la forme d'une IaaS avec OpenStack. L'objectif était de permettre à des utilisateurs (des élèves par exemple) de créer des machines virtuelles très rapidement même sur des machines disposant de peu de ressources.

Ensuite nous avons voulu mettre en place une solution pour automatiser la configuration des instances. L'objectif était de permettre à un/des administrateur(s) de modifier une configuration ou de rajouter une application même après qu'une image ait été créée.

Fonctionnalités à implémenter

- Un noeud de virtualisation avec Kernel-based Virtual Machine (KVM)
- Authentification & Autorisations avec un serveur Lightweight Directory Access Protocol (LDAP)
- Interface simple d'utilisation pour la création d'images
- Personnalisation des instances au lancement suivant l'utilisateur (montage des disques, ...)

Perspectives

- Haute-disponibilité, tolérance de pannes
- Monitoring des machines physiques
- Nœuds de virtualisation supplémentaires avec Xen, LinuX Containers (LXC), voir Hyper-V
- Migration automatique des VMs après la chute d'un nœud

1.4 Démarche

Initialement notre projet devait comporter deux grandes étapes : la mise en place d'une installation de développement, et le déploiement sur un BladeCenter. Après cette dernière étape et discussion avec notre tuteur, nous avons décidé de modifier l'installation que nous avions réalisé sur le BladeCenter afin de maximiser les possibilités de virtualisation et de simplifier l'administration.
Au final ce sont donc trois grandes étapes que nous détaillons dans le chapitre 4 avec un accent particulier sur la dernière puisqu'il s'agit la seule installation réellement intéressante.

De plus la majorité de notre travail a été effectuée à deux. Notre sujet étant très pointu et compliqué, nous avons préféré être à deux sur la majorité des tâches afin de pouvoir mettre en commun nos idées et réfléchir ensemble à la résolution des problèmes rencontrés pendant des différentes installations.

Dans la suite de ce rapport nous commencerons par détailler le matériel et les logiciels utilisés, puis nous verrons comment nous les avons assemblés afin d'obtenir notre Cloud privé.

4. https ://github.com/openstack
5. https ://launchpad.net/openstack

2 Matériel

2.1 Serveur dédié OVH

Nous avons réalisé la première partie de notre projet sur un serveur dédié Kimsufi d'OVH. Il était équipé d'un processeur 64 bits à 3,4Ghz avec les extensions VT-x (Intel Core i3), de 8Go de RAM, et de 2x1To de stockage en RAID 0.

Il s'agit de la configuration minimale pour faire fonctionner une installation de développement d'OpenStack.

2.2 BladeCenter HP

Une fois notre projet fonctionnel sur le serveur dédié nous avons déployé une nouvelle installation d'OpenStack sur quatre lames du BladeCenter dont dispose notre département au sein de l'IUT.

La configuration est cette fois-ci beaucoup plus intéressante et permet d'entrevoir les réelles possibilités du Cloud Computing [1].

2.2.1 Configuration matérielle

Nous disposions de trois lames équipées chacune de deux processeurs AMD Opteron 250 (2,4GHz / 64 bits) et de 4Go de RAM, ainsi que d'une lame équipée de deux Opteron 275 (2,2GHz / 64 bits) et de 8Go de RAM. Toutes les lames possèdent deux cartes Fibre Channel (FC) QLogic Q2312 pour le stockage.

2.2.2 Stockage

Les lames ne disposant pas de stockage intégré, ce dernier est assuré pas un Storage area network (SAN) HP StorageWorks relié en FC. Ce dernier fournit une couche d'abstraction au-dessus des disques durs et permet d'exposer plusieurs disques « virtuels » à une lame au travers du contrôleur FC.
Les contrôleurs font ce que l'on appelle du Multipath, c'est à dire qu'ils utilisent plusieurs chemins physiques pour accéder aux disques afin de faire du Failover.

2.2.3 Réseau

Chaque lame possède deux liens à 1Gbit/s que nous avons agrégés afin de faire du Load balancing et du Failover. Cela présente deux avantages : dans le meilleur cas on obtient une bande passante totale de 2Gbit/s, dans le pire cas (chute d'un lien) le trafic continue de fonctionner (à 1Gbit/s).

La séparation des réseaux est permise par l'ajout de tags sur des trames Ethernet avec un numéro de Virtual LAN (VLAN) (encapsulation 802.1Q) et la configuration des ports en mode trunk sur les switchs.

2.2.4 Compatibilité des systèmes d'exploitation

L'installation des systèmes d'exploitation sur les lames fût plus compliquée que prévu, étant donné la configuration bi-processeur et les cartes Fibre Channel.

Après de multiples installations infructueuses, un appel au support HP et la vérification des versions des firmwares nous sommes arrivés à la conclusion que les noyaux Linux récents (> 2.6) sont trop instables. Pour en savoir plus sur les systèmes installables voir l'annexe C.1.

1. Toutefois, bien que le BladeCenter possède une puissance importante, ses composants sont assez anciens et ne supportent pas les technologies de virtualisation matérielle (VT-x, VT-d, AMD-V), ce qui limite le nombre de systèmes invités possibles.

3 Logiciels

3.1 OpenStack

Nous avons choisi OpenStack car des solutions d'IaaS présentés précédemment dans la section 1.2.3 c'est celle qui est soutenu par le plus d'entreprise, et, comme en témoigne le volume des recherches Google, elle possède une communauté très développé sur Internet.

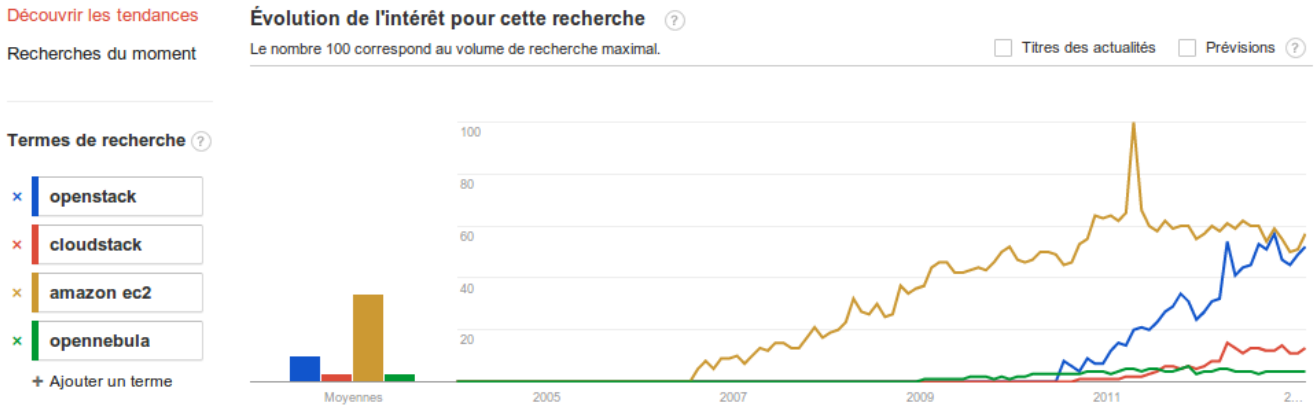


FIGURE 3.1 – Évolution du volume des recherches Google pour les principales IaaS

3.1.1 Les services

OpenStack est constitué de différents services gérant chacun une composante spécifique de l'IaaS.

Keystone

Ce service est indispensable, il permet de coordonner l'authentification et l'accès aux autres services.
Il expose une API HTTP(S) sur deux endpoints, un public sur le port 5000, permettant l'authentification des utilisateurs, et un autre dit privé sur le port 35357 permettant les tâches administratives tels que la gestion des utilisateurs et des terminaisons de services. Il supporte différents backends pour l'authentification, certains permettant la persistance et l'édition des données comme Structured Query Language (SQL) ou LDAP, d'autres en lecture-seule comme Pluggable Authentication Modules (PAM).

Nova

C'est le service qui va gérer les systèmes de virtualisation. Il supporte QEMU, KVM, et LXC au travers de libvirt ainsi que Xen et VMware ESXi en communicant avec leur API respective. Il est chargé de créer/démarrer/arrêter les machines, et de gérer les ressources physiques qui leurs sont allouées. Étant donné qu'il ne fait que donner des instructions aux hyperviseurs il peut-être redémarré sans risque de coupure des instances.
Nova peut aussi gérer le stockage bloc si Cinder n'est pas utilisé, et le réseau si Quantum n'est pas installé.
Il utilise Keystone pour l'authentification.

Quantum

Pour des besoins simples le réseau peut-être géré par nova-network. Pour plus de flexibilité il est possible de le remplacer par Quantum qui offre une API à part entière ainsi que des fonctionnalités avancées comme le tunneling L3 dans L2 et le support des plugins de switchs virtuels (OpenVSwitch, Cisco Nexus, ...).

Glance

Glance s'occupe des images virtuelles [1]. Il enregistre leur caractéristiques et leur emplacements de façon à ce que les autres services n'aient pas à s'en occuper.

Il utilise Keystone pour l'authentification et peut, en plus du système de fichier local, utiliser Swift pour stocker les images.

Dashboard

C'est l'interface graphique qui permet à l'utilisateur final de gérer ses instances virtuelles et le réseau associé, ses volumes, et ses conteneurs (stockage objet) ainsi qu'à l'administrateur de gérer les projets et les utilisateurs.

3.2 Automatisation

Une fois la ou les machines virtualisées, il faut les configurer et installer des applications. On peut effectuer ces tâches à la main mais il existe des solutions libres qui permettent d'automatiser ces tâches.

Juju

Canonical, la société qui édite Ubuntu distribue, sous licence libre, Juju, un service d'orchestration, écrit en Python, qui permet de configurer automatiquement des services. [2] et de créer des relations entre eux. Juju rajoute une couche d'abstraction entre la machine et le service. Il suffit de définir des relations entre les charmes et ils se configurent automatiquement. Par exemple, il suffit d'établir une relation entre un logiciel de Load balancing (comme Varnish) et notre service (MediaWiki par example) pour que la charge soit répartie sur les différents serveur, quelque soit le nombre d'instance du service (ici MediaWiki). Juju s'installe sur l'ordinateur de l'utilisateur et s'utilise en ligne de commande mais il existe une interface web (en beta à l'heure actuel) qui s'installe sur un serveur distant. Juju supporte la plus part des IaaS public du marché (HP, Amazon et Rackspace) ainsi que les IaaS privé basés sur OpenStack. L'annexe E.1 explique comment déployer plusieurs instances de MediaWiki avec quelque dépendances (MySQL et HAProxy).

Juju est très pratique pour déployer rapidement un environnement complet et durable dans le temps. Cependant, il l'est beaucoup moins pour appliquer une configuration avancée sur une multitude de machines. Pour ce genre de besoin, il existe des solutions comme Puppet, Chef ou encore CfEngine qui permettent d'automatiser ces configurations et de les appliquer simultanément sur plusieurs machines.

Puppet

Nous avons choisi d'utiliser Puppet car il possède une communauté très active et la syntaxe de ses scripts de configuration nous semble la plus claire. Puppet marche sur un système de clients-serveur. Chaque machine à configurer, appelée nœud, a le client Puppet installé. Elles doivent s'identifier auprès du serveur, appelé PuppetMaster. Les communications entre le PuppetMaster et ses clients sont chiffrées grâce à des certificats générés par Puppet à son démarrage. Afin de définir les configurations à effectuer sur chaque nœud, on utilise des recettes.(Voir l'annexe E.2 pour une explication concrète d'utilisation d'une recette).

3.3 Haute disponibilité

3.3.1 GlusterFS

GlusterFS est un système de fichier distribué. Il permet de redonder les données stockées dans une grappe de serveur. Il est accessible via un client spécifique ou tout simplement via un point de montage Network File System (NFS).

3.4 Reverse Proxy

Pour les premières installations, nous avons décidé de ne connecter directement aucun service avec l'extérieur pour des raisons de sécurité, de contraintes techniques (une seul IP disponible sur le réseau de l'IUT) et de facilité de mise en œuvre (configuration du pare-feu simplifiée). De cette manière, la zone qui abrite le dashboard

1. Une image est le modèle d'après lequel serons créées les machines.
2. Dans Juju, un service est appelé un "charme". Un charme peut aussi bien être un programme comme Apache ou Varnish ou un service web comme MediaWiki

n'est connectée qu'au LAN interne des lames. De cette manière, il nous faut un moyen de rendre le dashboard accessible depuis l'extérieur. Nous avons choisi de mettre en place un mécanisme de reverse proxy [3].

Un reverse proxy est un serveur qui permet de réécrire les requêtes HTTP afin, par exemple, de cacher à l'utilisateur l'architecture interne du réseau.
Grâce à lui, nous pouvons rediriger les requêtes vers notre serveur web. Un reverse proxy permet aussi de mettre en cache les ressources statiques d'un site pour l'accélérer et soulager les serveurs web ou encore répartir la charge entre plusieurs serveurs.

3.4.1 Apache

Il existe plusieurs programmes qui permettent de faire du reverse proxy. Le mieux, de par sa communauté, la finesse de ses règles, et ses performances est, à notre avis, Varnish [4]. C'est un logiciel libre utilisé par exemple par Facebook. Cependant, nous n'avons pas pu l'utiliser pour les raisons détaillées dans la partie dédiée.

Nous avons donc décidé d'utiliser Apache comme reverse proxy. Apache est le serveur web le plus utilisé [5] dans le monde, à l'heure actuelle. Il propose une fonction de reverse proxy grâce au module mod_proxy. Sa configuration est un peu moins fine que Varnish mais suffisante pour notre utilisation.

3.5 Supervision

Afin d'être sur qu'il n'y aucun problème à la fois sur le réseau et sur les machines, il existe des programmes appelés logiciels ou plates-formes de supervision qui, en s'appuyant sur des protocoles tels qu'Simple Network Monitoring Protocol (SNMP) ou Telnet, vérifient en continu le bon fonctionnement de tous les équipements réseaux et de tous les logiciels installés sur les machines. Ils peuvent vérifier de la place restante sur les disques jusqu'à l'état des interfaces réseaux des switchs, en passant par le nombre de clients connectés à la base MySQL. Lorsque un problème est détecté, une alerte est émise [6].

3.5.1 Shinken

Le marché de la supervision est à la fois occupé par de grand constructeur comme HP avec leur gamme OverView et par des logiciel libre comme Zabbix ou Nagios. Nous avons choisi pour notre part d'utiliser Shinken [7]. Il a l'avantage de s'installer facilement, d'être compatible avec Solaris [8] et d'être compatible avec les plugins Nagios et ainsi de bénéficier de son immense communauté.

3.5.2 Munin

Munin permet de tracer des graphes, de l'utilisation de l'interface réseau au nombre de processus en cours, en passant par le nombre d'entrées dans la table du système de fichiers. Il offre peut de fonctions mais est très simple à configurer et relativement léger.
Il se compose de deux parties : munin-node à installer sur chaque machine à monitorer et munin à proprement parler qui va régulièrement interroger munin-node sur chaque machine afin de récupérer les données et de générer des pages HTML statiques avec les graphes.
Une alternative courante est MRTG.

3. Nous aurions pu décider de translater le port 80 (HTTP) sur l'IP de la zone abritant le serveur web, mais cette solution aurait été moins souple. En effet, la translation ne nous autorise qu'une seule redirection alors que le reverse proxy se base sur les données HTTP et non le port. De cette manière, il nous permet de créer autant de sous domaines que l'on veut et de les rediriger vers n'importe quel IP et port.
4. https ://www.varnish-cache.org/
5. D'après le site netcraft.com, en Février 2013, Apache est utilisé par près de 55% des sites actifs. (IIS de Microsoft et Nginx sont eux à 12%)
6. Soit par mail, soit par SMS... Tout dépend du logiciel de supervision et de sa configuration.
7. http ://www.shinken-monitoring.org/
8. Shinken est écrit en python ce qui le rend disponible pour presque tous les systèmes.

4 Mise en place

4.1 Installation de développement sur un serveur dédié

Nous avons réalisé une première installation d'OpenStack et des services associés sur un serveur dédié afin d'étudier le fonctionnement global. Dans cette section nous ne détaillerons que les configurations des systèmes d'exploitation, et pas l'installation d'OpenStack qui sera abordé dans la section 4.3.

4.1.1 Environnement de test

Afin de simuler plusieurs serveurs physiques nous avons installé Proxmox sur le serveur dédié. Il s'agit d'une distribution Linux associant KVM et OpenVZ à une interface web afin de permettre la création rapide de machines virtuelles.

A l'aide de Proxmox nous avons donc créé deux machines virtuelles fonctionnant sur KVM : *node-controller* et *node-compute*. Après avoir effectué une installation minimale d'Ubuntu Server 12.04 LTS sur les deux machines nous avons installé OpenStack conformément aux instructions du manuel « OpenStack Install and Deploy Manual - Ubuntu »[7].

4.1.2 Conclusion

Cette installation nous a permis de commencer à travailler rapidement car il s'agissait d'une configuration standard et les préliminaires à l'installation d'OpenStack (installation des systèmes d'exploitations, configuration du réseau) nous ont pris très peu de temps. Elle était donc idéale à des fins d'étude et de développement.
Cependant il ne s'agit pas d'une configuration utilisable en production car beaucoup trop lente du fait du peu de ressources disponibles.

4.2 Déploiement sur un BladeCenter avec Solaris

Par souci de clarté, nous avons décidé de donner un nom (hostname) au trois lames que nous utilisons afin de les différencier plus facilement. Les noms sont donnés dans le schéma global (Annexe D.2). Par la suite, lorsque nous parlerons d'une lame en particulier, nous l'appellerons par son nom.

Dans cette section nous ne détaillerons que la configuration des systèmes d'exploitation mais pas la configuration d'OpenStack, qui est détaillé dans la section suivante (4.3)

4.2.1 Architecture

Nous avons choisi d'installer Solaris 11 sur Fuji et Iyo et CentOS 6 sur Raijin.
L'installation des systèmes est détaillé dans les annexes C.3 et C.4.

Réseau

Comme expliqué précédemment, les lames Iyo, Tenjin et Raijin sont connecté en 2Gbit/s (deux interfaces à 1Gbit/s agrégés comme expliqué dans cet article [3]) dans un VLAN dédié et non-relié à l'extérieur. Fuji, quand à elle, a une interface sur le réseau extérieur (réseau de IUT avec accès à internet) et l'autre dans le LAN inter-lames. De cette manière, tous le trafic transite par elle et peut être redirigé suivant les règles du reverse proxy et du pare-feu.

Afin de donner accès à internet au lames du réseau interne, nous avons mis en place une translation d'adresse sur Fuji (NAT). Sur Solaris, le pare-feu par défaut est ipfilter. La commande suivante permet de translater l'adresse locale (192.168.1.0/24) vers l'adresse public (10.102.75.190) :

```
echo 'map net0 192.168.1.0/24 -> 10.102.75.190' >> /etc/ipf/ipf.conf
```

Maintenant que nos machine on accès à internet, il faut qu'on rende disponible depuis l'extérieur les interfaces web des différent services disponible (Le dashboard, le panel Shinken et l'interface de configuration de Puppet). Pour réaliser cela, nous allons utiliser un reverse proxy (expliqué précédemment). Celle-ci renvoie tous le trafic qui arrive à Apache vers la machine qui a l'IP 192.168.1.3 et sur sont port 80.

Particularités liées à l'utilisation de Solaris

Solaris possède son propre gestionnaire de paquet que l'on peut appeler par la commande pkg. Seulement, il y a assez peu de paquet disponible. Une communauté s'est alors créé autour du projet OpenCSW, un gestionnaire de paquet qui contient un très grand nombre [1] de paquet compilés pour Solaris. OpenCSW est disponible à l'adresse : www.opencsw.org.

Certains utilitaires système de Solaris tels que pkg sont écrits en Python et utilisent des librairies en commun avec les services d'OpenStack. Pour éviter tout problème de compatibilité il est nécessaire d'installer les services dans un environnement virtuel (*virtualenv*) à l'aide de la commande :

```
python tools/install_venv.py
```

4.2.2 Conclusion

S'agissant de notre premier contact avec un BladeCenter un certains nombre de concept étaient nouveaux pour nous (FC, SAN, multipath) et nous manquions de recul sur le dimensionnement de notre installation. C'est pourquoi, après discussion avec notre tuteur, nous avons décidé de reprendre l'installation de manière beaucoup plus propre.

En effet bien que Solaris fonctionne parfaitement sur les lames son administration diffère beaucoup des systèmes Linux et la pérennité de l'installation n'est donc pas garantie.
De plus notre configuration réseau nous obligeait à faire du Network Address Translation (NAT) pour accéder aux machines virtuelles et ce n'est pas une configuration idéale, surtout dans le cas où les utilisateurs ont besoins d'accéder à plusieurs ports sur une machine virtuelle.

4.3 Déploiement sur un BladeCenter avec CentOS

Comme dans la section 4.2 nous avons donné des noms (hostname) aux lames afin de les différencier plus facilement que par leur IP. Ces noms sont détaillés dans le tableau ci-dessous ainsi que dans l'annexe D.3

Nous avons cette fois ci voulu réaliser une installation permettant de maximiser les capacités de virtualisation et de résister à la chute éventuel d'un noeud. Pour cela nous nous sommes basés sur ce guide https ://github.com/beloglazov/openstack-centos-kvm-glusterfs que nous avons adapté.

Sur les switchs tout les ports ont été placés en mode trunk et la tagguage des VLANs est réalisé par les systèmes d'exploitations. Les liens ont été agrégés de façon à augmenter la bande passante et à disposer d'un lien de secours.

Dans cette configuration nous distinguerons deux types de nœuds : le *cloud controller* et les noeuds de *compute*. Le premier possèdera tout les services d'OpenStack (y compris nova-compute) et sera chargé de router les paquets des instances sur le réseau public tandis que les seconds ne disposeront que de nova-compute et se contenteront de faire tourner les instances.

Afin de pouvoir migrer les instances entre les lames nous avons mis en place un stockage distribué et partagé entre les machines à l'aide de GlusterFS (voir 3.3.1).

Volume	Taille (Go)	Périphérique
VolGroup/lv_root	32	/dev/mapper/mpathap2
VolGroup/lv_swap	8	/dev/mapper/mpathap2
nova-volumes	200	/dev/mapper/mpathbp1

FIGURE 4.1 – Récapitulatif du stockage sur le Cloud Controller

Volume	Taille (Go)	Périphérique
VolGroup/lv_root	36	/dev/mapper/mpathap2
VolGroup/lv_swap	4	/dev/mapper/mpathap2
vg_gluster	200	/dev/mapper/mpathbp1

FIGURE 4.2 – Récapitulatif du stockage sur le nœuds de *compute*

1. le projet OpenCSW propose 3675 paquet en février 2013

Lame	Interface	VLAN	Addresse IP
Raijin	bond0.2	2	10.102.75.100
	bond0.99	99	
Taijin	bond0.2	2	10.102.75.101
	bond0.99	99	
Iyo	bond0.2	2	10.102.75.102
	bond0.99	99	

FIGURE 4.3 – Récapitulatif de la configuration réseau des lames

4.3.1 Préliminaires

Installation des systèmes d'exploitation

Nous avons choisi d'installer CentOS 6.3 x64 sur les quatre lames. Le procédure d'installation est décrite dans l'annexe C.3.
Il s'agit d'une installation sans Internet car cette fois-ci les liens sont en mode trunk et nous tagguons nous-même les VLANs.

Configuration des dépôts YUM

Les dépôts de paquets par défaut de CentOS ne contiennent pas OpenStack. Il faut donc ajouter le dépot EPEL [2], pour cela :

```
sudo rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
```

4.3.2 Installation du Cloud Controller

Installation du serveur MySQL

```
yum install -y mysql mysql-server

# On change le mot de passe root de mysql
mysqladmin -u root password PASS_MYSQL
```

Installation du service d'identité

Il faut d'abord créer la base dans MySQL :

```
mysql -u root -p

CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'PASS_DBKEYSTONE';
GRANT ALL ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'PASS_DBKEYSTONE';
```

On peut maintenant installer Keystone :

```
yum install -y openstack-utils openstack-keystone
```

Pour configurer Keystone voir l'annexe F.1.
Fin de la configuration :

```
# On crée les tables dans la DB
keystone-manage db_sync

# On mets les bonnes permissions (le fichier de config contient le pass SQL !)
chmod 640 /etc/keystone/keystone.conf
chown -R keystone:keystone /var/log/keystone
chown -R keystone:keystone /var/lib/keystone

# On démarre le service et on l'active au démarrage
service openstack-keystone restart
chkconfig openstack-keystone on
```

Pour la création des utilisateurs de base, services, et endpoints, on peut utiliser le script à l'adresse : `https://github.com/openstack/keystone/blob/90ebf9/tools/sample_data.sh`.

2. Extra Packages for Enterprise Linux. Ensemble de paquets supplémentaires pour RedHat, CentOS, et Scientific Linux.

Installation du services d'image

On commence par créer la base MySQL :

```
mysql -u root -p

CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY 'PASS_DBGLANCE';
GRANT ALL ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'PASS_DBGLANCE';
```

On installe Glance :

```
yum install -y openstack-glance
```

Pour configurer Glance voir l'annexe F.2.
Fin de la configuration :

```
# On crée les tables dans la DB
glance-manage db_sync

# On mets les bonnes permissions
chmod 640 /etc/glance/*.conf
chmod 640 /etc/glance/*.ini
chown -R glance:glance /var/log/glance
chown -R glance:glance /var/lib/glance

# On démarre les services et on les active au démarrage
service openstack-glance-registry restart
service openstack-glance-api restart
chkconfig openstack-glance-registry on
chkconfig openstack-glance-api on
```

Installation de Nova

On crée la base MySQL :

```
mysql -u root -p

CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'nova'@'%' IDENTIFIED BY 'PASS_DBNOVA';
GRANT ALL ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'PASS_DBNOVA';
```

On installe Nova et le serveur AMQP :

```
yum install -y openstack-nova* qpid-cpp-server
```

Pour configurer Nova voir l'annexe F.3.
Fin de la configuration :

```
# On crée les tables dans la DB
nova-manage db sync

# On mets les bonnes permissions
chmod 640 /etc/nova/nova.conf
chown -R root:nova /etc/nova
chown -R nova:nova /var/lib/nova

# On démarre les services et on les active au démarrage
service qpidd restart
service tgtd  restart
service openstack-nova-api          restart
service openstack-nova-cert         restart
service openstack-nova-consoleauth  restart
service openstack-nova-direct-api   restart
service openstack-nova-scheduler    restart
service openstack-nova-volume       restart
service openstack-nova-network      restart
chkconfig qpidd on
chkconfig tgtd  on
```

```
chkconfig openstack-nova-api          on
chkconfig openstack-nova-cert         on
chkconfig openstack-nova-consoleauth  on
chkconfig openstack-nova-direct-api   on
chkconfig openstack-nova-scheduler    on
chkconfig openstack-nova-volume       on
chkconfig openstack-nova-network      on
```

4.3.3 Installation des noeuds de Compute

Installation de Nova

On installe Nova :

```
yum install -y openstack-nova*
```

Pour configurer Nova voir l'annexe F.3.
Fin de la configuration :

```
# On mets les bonnes permissions
chmod 640 /etc/nova/nova.conf
chown -R root:nova /etc/nova
chown -R nova:nova /var/lib/nova

# On démarre les services et on les active au démarrage
service openstack-nova-compute restart
chkconfig openstack-nova-compute  on
```

4.3.4 Conclusion

L'architecture de cette installation est beaucoup plus efficace que les précédentes, elle permets de maximiser les capacités de virtualisation ainsi que d'étendre les fonctionnalités. De plus CentOS étant un système largement documenté l'administration en est simplifié. Il a également été beaucoup plus facile d'installer les logiciels de supervision (Shinken et Munin).

5 Conclusion

La mise en place d'une infrastructure de Cloud Computing demande de nombreuses connaissances dans le domaine de l'informatique et des réseaux. Nous avons appris énormément durant la réalisation de ce projet mais nous avons clairement manquer de recul dans nos choix et avec de l'expérience nous aurions pu être plus efficaces.
Cependant nous avons pu implémenter toutes les fonctionnalités du cahier des charges y compris les fonctionnalités supplémentaires tels que la haute disponibilité et le monitoring, à l'exception de l'authentification avec le LDAP car non réalisable avec le serveur de l'Université de Savoie [1].

Sur la partie matériel nous avons été limités par les capacités du matériel qui, datant de quelques années, ne possède pas toutes les fonctionnalités nécessaire à la virtualisation.

Enfin concernant OpenStack il s'agit d'un projet au potentiel énorme de part sa flexibilité (services indépendants), sa portabilité (Python), et sa communauté (*mailing lists* entre autres) mais qui est encore en développement actif et dont la compatibilité n'est pas assurée d'une version à l'autre.

1. Keystone requiert des droits en écriture et une structure de l'arbre particulière, ce qui n'est évidemment pas possible sur le LDAP de l'Université.

A Utilisation du Cloud privé

A.1 Généralités

A.1.1 Dashboard

Le dashboard d'OpenStack qui offre une interface conviviale pour utiliser et administrer le cloud se trouve à l'adresse http ://10.102.75.100/dashboard .

A.1.2 Projets

Les projets permettent des définir des quotas de ressources pour un groupe d'utilisateur. Un utilisateur peut appartenir à plusieurs projet. Ceux-ci sont configurables dans partie *Admin* du Dashboard.

A.1.3 Images

Les instances sont créés à partir d'images, contenant le système d'exploitation et les éventuels logiciels, qui peuvent être créés ou téléchargés sur Internet. Pour ajouter une image il suffit d'indiquer son URL, son nom, et son format dans le dialogue *Create An Image* de la section *Images & Snapshots* du Dashboard.

A.1.4 Réseau privé vs. Réseau public

Par défaut les instances virtuelles se voient attribué une adresse IP privé (dite fixe car une fois attribuée à une instance elle le restera jusqu'à la destruction de cette dernière) sur le réseau 192.168.100.0/24 et elles ne sont donc accessible qu'à travers le VNC (voir section suivante). Pour y accéder depuis l'extérieur il est possible de leur attribuer une adresse IP publique (dite flottante car elle peut être détaché d'une instance pour aller sur une autre) sur le réseau 10.102.75.0/24 [1] qui est accessible depuis l'IUT ou via le VPN.

Pour assigner une IP flottante à une instance il suffit de choisir *Associate Floating IP* dans le menu déroulant *Actions* dans la liste des instances.

FIGURE A.1 – Association d'une IP flottante à une instance via le dashboard

A.1.5 Groupes de sécurité

Par défaut les instances ne sont pas accessibles depuis l'extérieur. Pour ouvrir certains ports il faut créer un *Security Group* via l'onglet *Access & Security* du Dashboard.

A.1.6 VNC

Le dashboard offre un client VNC qui permet de contrôler (clavier, souris, et écran) une instance virtuelle. Le client utilisent des fonctionnalités d'HTML5, et par conséquent nécessite un navigateur récent. Toutes les

1. En réalité un subnet en 10.102.75.128/27 car les autres IPs sont potentiellement utilisés.

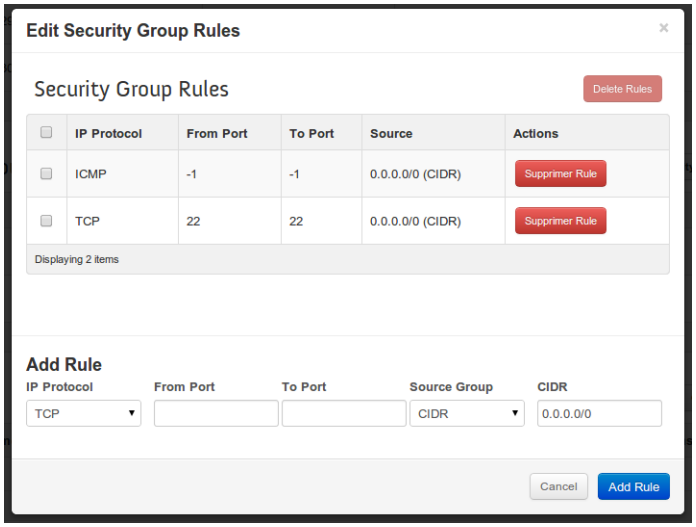


FIGURE A.2 – Édition d'un groupe de sécurité via le dashboard

versions de Firefox et de Chrome (ou autres navigateurs basés sur Webkit) depuis 2012 le supporte. C'est le seul moyen [2] d'accéder aux instances si elles n'ont pas d'IP publique.

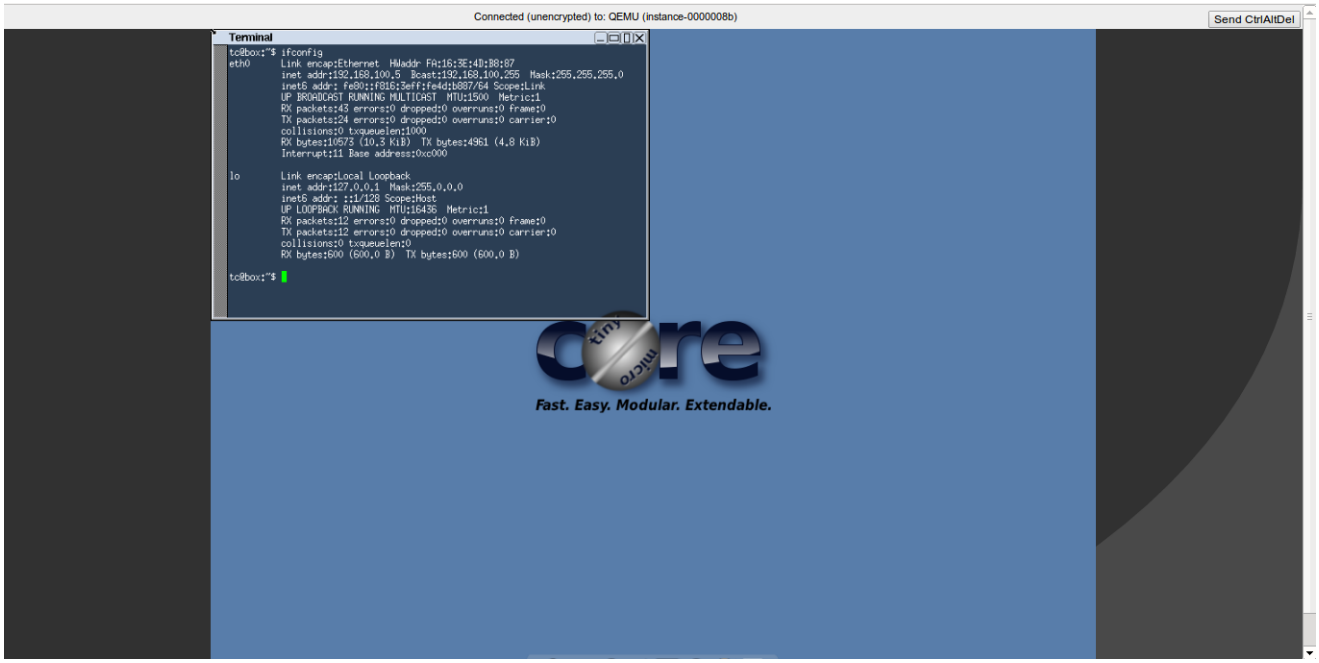


FIGURE A.3 – Session VNC sur une instance TinyCore via le dashboard

A.1.7 Keypair

Si l'image virtuelle le supporte [3] il est possible d'injecter une clé publique à la création de l'instance afin d'y accéder via SSH. Les images ne supportant pas cette méthode d'authentification proposent en général une authentification par mot de passe.

Pour créer ou envoyer une paire de clés il suffit de se rendre dans la partie *Access & Security* du Dashboard puis dans la dernière section, *Keypairs*, de cliquer sur le bouton correspondante (*Create Keypair* ou *Import Keypair*).

A.2 Création d'une instance virtuelle

Pour créer une instance il faut, après s'être identifier, se rendre dans le section *Instances* du menu *Project* à gauche. Si besoin choisir le projet dans la liste déroulante en haut du menu.

2. Il est possible d'accéder aux instances sur leur IP privé via les lames ou fonctionnent OpenStack mais ce n'est évidemment pas faisable pour l'utilisateur final.

3. Pour récupérer la clé publique et d'autres informations, il faut que l'instance interroge l'API metadata à l'adresse http ://169.254.169.254

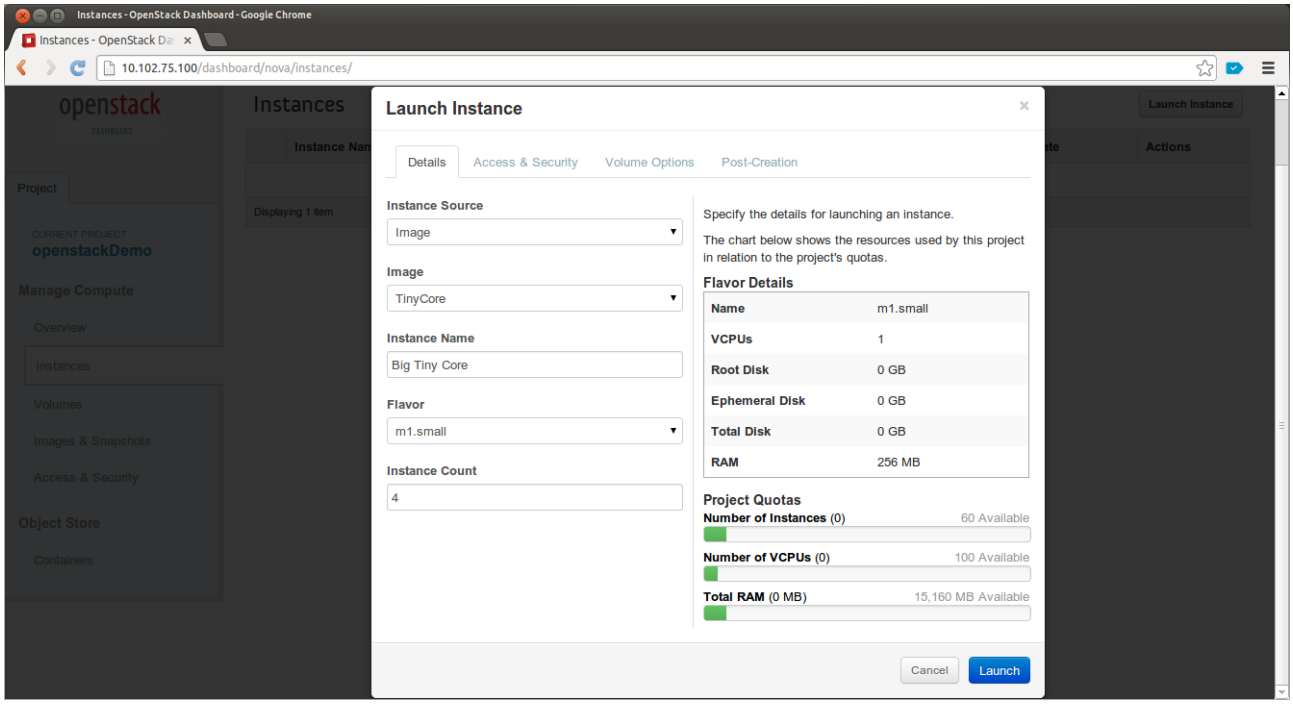


FIGURE A.4 – Exemple de création de quatre instances virtuelles via le dashboard

Ensuite il suffit de cliquer sur *Launch Instance* puis de choisir l'image, le nom, le nombre, et les ressources (*Flavor*). Dans l'onglet *Access & Security* on choisira sa paire de clé si besoin (voir section précédente).
Enfin il suffit de cliquer sur *Launch*.

B Administration du Cloud privé

B.1 Gestion avec le Dashboard

Le Dashboard offre la possibilité de gérer simplement les utilisateurs, les projets, et les instances. La partie *Admin* est accessible aux utilisateurs qui possèdent le rôle correspondant.

B.2 Gestion en ligne de commande

Sur n'importe quelle des lames où fonctionne OpenStack il est possible de tout gérer à l'aide des commandes nova, nova-manage, keystone, glance, et swift. La commande openstack-config permet de modifier les fichiers de configuration en ligne de commande, tandis que la commande openstack-status donne une vue globale des services installés sur la lame.

B.3 Dépannage à l'aide des logs

En cas de problème les logs d'OpenStack sont d'une grande utilité. Ils sont consultables dans /var/log/ à l'exception de ceux du Dashboard pour lequel il faut voir les logs d'Apache dans /var/log/httpd/ . Pour débugger en temps réel une commande utile est tail -f qui permet d'afficher le contenu des logs en même temps qu'ils sont écrits. Un exemple d'utilisation pour visualiser les erreurs survenant avec nova serait :

```
tail -f /var/log/nova/* | grep ERROR
```

Note : pour une administration simplifiée il serait possible de rediriger les logs vers un serveur syslog.

C Installation des systèmes d'exploitation sur les lames

Les lames du BladeCenter ne disposant pas de lecteur optique et l'accès physique n'étant pas toujours faisable, il est possible de réaliser l'installation des systèmes d'exploitation à distance. Nous détaillerons ici les fonctionnalités intégrés aux lames permettant de les gérer à distance, ainsi que l'installation des différents systèmes que nous avons utilisés.
Note : les installations détaillés sont spécifiques au BladeCenter et correspondent à la configuration minimum pour faire fonctionner OpenStack. Des documentations plus exhaustives sont disponibles sur les sites respectifs des OS.

C.1 Systèmes d'exploitation supportés

Systèmes officiellement supportés

D'après HP les systèmes d'exploitations supporté sur les lames BL35p sont :
- Red Hat Enterprise Linux (RHEL) 5
- Suse Linux Enterprise Server (SLES) 10
- VMware ESX 2.5.5
- Solaris 10

Systèmes que nous avons testés

Après de multiples essais nous avons conclus que les systèmes suivants sont stables et supportent le multipath en natif sur les lames :
- Solaris 11 (64 bits)
- CentOS 6.3 (64 bits)
- VMware ESXi 4.1 (dernière version installable, les versions suivantes requièrent les extensions processeurs LAHF et SAHF)

C.2 Integrated Lights-Out

Chaque lame possède un serveur de gestion intégré, Integrated Lights-Out (iLO), permettant de gérer l'alimentation (allumage, extinction, reset), de connecter des médias virtuels, et fournissant un Keyboard, Video, Mouse (KVM) dans le navigateur. Les iLO sont sur un réseau privé en 10.100.100.0/24 qui est accessible depuis un serveur TSE à l'adresse manap.rt.iut-acy.local (10.102.75.190).

C.2.1 Connexion

Serveur TSE

Le serveur TSE manap.rt.iut-acy.local est accessible depuis tout poste de l'IUT à l'adresse 10.102.75.190 :3389. Windows XP,Vista,7 et 8 possèdent un client RDP nommé "Connexion Bureau à distance". Sur Linux on pourra utiliser Remmina qui intègre le tunneling SSH (voir paragraphe suivant).

Depuis l'extérieur (WiFi, maison) il est nécessaire d'utiliser le VPN [1] mais, celui-ci bloquant les ports peu utilisés (pour des raisons de sécurité), il faut également mettre en place un tunnel SSH. Pour ce faire il faut d'abord demander un accès à srv-dev.iut-acy.local à la DSI puis configurer son client Secure SHell (SSH).
Sur Windows on utilisera Putty [2] tandis que sur Linux, à défaut de client gérant le tunneling comme Remmina, on utilisera la commande ssh comme suit pour mapper le port local 10389 sur le port 3389 du serveur TSE :

```
ssh -L 10389:10.102.75.250:3389 login@srv-dev.iut-acy.local
```

1. Le téléchargement et l'installation du client VPN sont détaillés sur vpn.univ-savoie.fr
2. Les instructions de configuration du tunnel SSH sous Windows sont disponibles à l'adresse https ://w3.iut-acy.univ-savoie.fr/institutionnel/service-informatique/faq/tunnel-ssh/

Il suffira alors d'indiquer l'adresse 127.0.0.1 :10389 dans le client RDP pour se connecter.

Le serveur, fonctionnant sous Windows Server 2003, possède Firefox comme navigateur Internet, et plusieurs autres logiciels utiles, comme le client VMWare vCenter pour la gestion d'ESXi.

iLO

Il suffit d'indiquer l'adresse IP correspondant à l'iLO de la lame dans le navigateur et d'entrer ses identifiants.

C.2.2 Principales fonctionnalités

Gestion de l'alimentation

La section *Virtual Power* de l'onglet *Virtual Devices* indique l'état actuel de l'alimentation de la lame et propose six options :

Momentary Press : Équivalent à un appui court sur le bouton Power de la lame. Permet de démarrer ou d'arrêter proprement le système.

Press and Hold : Équivalent à un appui de cinq secondes sur le bouton Power de la lame. Permet d'éteindre la lame si le système ne répond plus.

Cold boot of system : Coupe l'alimentation pendant six secondes puis allume la lame. Permet aux condensateurs de se décharger et ainsi de résoudre certains problèmes matériels.

Reset system : Redémarre la lame.

Manual Override for BL p-Class : Force le mise sous tension de la lame sans tenir compte de la puissance électrique disponible. Peut éteindre les autres serveurs présents dans le même rack.

Automatically Power On Server : Si *Yes*, permet de remettre automatiquement sous tension la lame après une coupure de courant.

Média virtuel et console distante

L'onglet *Remote Console* offre le choix entre plusieurs consoles permettant de prendre le contrôle (clavier, souris, écran) de la lame, tandis que le lien *Virtual Media* de l'onglet *Virtual Devices* permet d'associer une image ISO à un lecteur CD virtuel de la lame.

C.3 Installation de CentOS

C.3.1 Média d'installation

La lame a accès à Internet

Dans le cas où la lame a accès à Internet, on réalisera une installation par le réseau, plus rapide et permettant d'obtenir les dernières versions des paquets. Les images ISOs peuvent être obtenus de plusieurs miroirs [3] mais on préférera celui de l'IN2P3, connecté directement au réseau RENATER et qui offre donc d'excellents débits (de l'ordre de la dizaine de Mo/s).
On téléchargera l'image ISO netinstall à l'adresse suivante :

`http://mirror.in2p3.fr/linux/CentOS/[version]/isos/{i386,x86_64}/`

Le dossier contenant l'image d'installation est quand à lui situé à l'adresse :

`http://mirror.in2p3.fr/linux/CentOS/[version]/os/{i386,x86_64}/`

Par exemple pour CentOS 6.3 x64 les URLs sont :

`http://mirror.in2p3.fr/linux/CentOS/6.3/isos/x86_64/CentOS-6.3-x86_64-netinstall.iso`
`http://mirror.in2p3.fr/linux/CentOS/6.3/os/x86_64/`

La lame n'a pas accès à Internet

Dans le cas où la lame n'aurais pas accès à Internet ou que le lien soit en mode trunk (il est compliqué de configurer les VLANs pendant l'installation) il faut effectuer une installation avec le DVD contenant tout les paquets.
L'image ISO sera préférablement téléchargé depuis le miroir de l'IN2P3 (voir paragraphe précèdent) à cette adresse :

`http://mirror.in2p3.fr/linux/CentOS/[version]/isos/{i386,x86_64}/`

Par exemple pour CentOS 6.3 x64 l'url du DVD est :

`http://mirror.in2p3.fr/linux/CentOS/6.3/isos/x86_64/CentOS-6.3-x86_64-bin-DVD1.iso`

3. La liste à jour de tout les mirroirs de CentOS est disponible à l'adresse http ://www.centos.org/modules/tinycontent/index.php ?id=30

C.3.2 Options de boot

Afin qu'Anaconda, l'installateur de CentOS, détecte correctement les cartes FC et le multipath et que l'installation se fasse en mode texte, il est nécessaire d'entrer la ligne suivante au prompt boot (ECHAP pour y accèder à partir de la version 6) :

```
linux text mpath
```

C.3.3 Configuration générale

Langue et clavier

On choisira la langue que l'on souhaite mais l'Anglais est préférable car les messages d'erreurs y sont souvent plus explicites et on évite les traductions hasardeuses.
Pour le clavier choisir le layout correspondant. Pour les claviers AZERTY on peut utiliser le layout fr ou fr-latin9 (avec la touche Euro).

Méthode d'installation

Image netinstall

Si l'on utilise l'image ISO netinstall il faut choisir "URL" au dialogue "Installation Method".
Dans le dialogue suivant qui demande de configurer la carté réseau on choisira celle qui est relié à Internet. Pour IPv4 on préferera configurer l'adresse en statique. Bien que début 2013 l'IPv6 ne soit pas encore routé sur le réseau de l'IUT son support peut-être laissé activé en prévision du futur.
Si l'on souhaite agréger plusieurs cartes réseaux, on n'en configurera qu'une seule pour l'instant et l'agrégation sera réalisé une fois le système installé.
Dans le dialogue demandant l'URL contenant l'image d'installation on rentrera celle indiquée dans la section C.3.1.

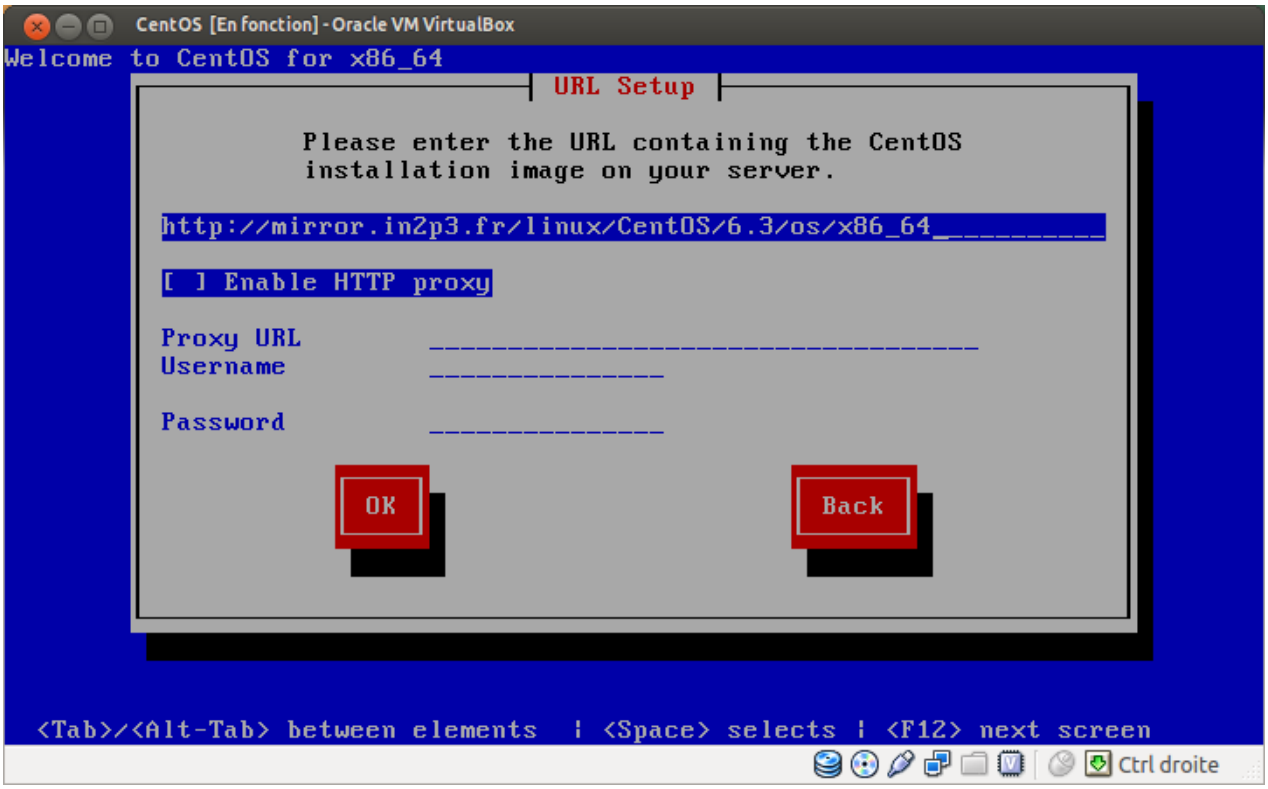


FIGURE C.1 – Exemple de configuration de la source pour CentOS 6.3 x64 en utilisant le mirroir de l'IN2P3

Image DVD

Si l'on utilise l'image ISO DVD on choisira "Local CD/DVD", au dialogue "Installation Method".
Le dialogue suivant demande de configurer la carte réseau. Si il y a des VLANs à configurer, on laissera la configuration automatique (DHCP) échouer et on choisira de configurer le réseau plus tard. Sinon on configurera les adresses IPv4 et IPv6.
Note : Si l'on souhaite agréger plusieurs cartes réseaux, on n'en configurera qu'une seule pour l'instant et agrégation sera réalisé une fois le système installé.

VNC

A partir de la version 5 de CentOS l'installateur propose d'effectuer l'installation via VNC afin de proposer un partitionnement personnaliser ou de modifier les paquets à installer. Étant donné que l'on souhaite installer un système minimal sur du stockage vide on continuera l'installation en mode texte.

Fuseau horaire

Linux sera le seul système installé "en dur" sur la lame, on peut donc laisser l'horloge système en UTC. Attention à bien choisir le bon fuseau horaire afin que le passage à l'heure d'été se fasse correctement et que le timestamp des logs soit cohérent entre les machines.

Mot de passe

Rien de spécial ici, il faut juste choisir un mot de passe sûr. Des sites comme random.org [4] permettent de générer des mots de passes aléatoires.

Partitionnement

On souhaite installer le système sur du stockage vierge, on choisira donc d'utiliser le disque entier. Si plusieurs disques sont présents on sélectionnera uniquement celui où le système résidera. On configurera plus tard le stockage supplémentaire nécessaire aux images et aux instance virtuelles.

Installation et premier redémarrage

Après le partitionnement l'installation se déroule toute seule et l'installateur demande de confirmer le redémarrage à la fin. Il faut bien penser à déconnecter le lecteur virtuel dans iLO afin d'éviter de redémarrer sur l'installation.
A partir de ce moment, et si le réseau est configuré il est possible d'accéder à la machine en SSH.

C.3.4 Post-configuration

Afin d'éviter de perdre la connexion il est préférable de modifier la configuration réseau depuis la console iLO plutôt que via SSH.

Hostname

Pour modifier le hostname il faut éditer la ligne HOSTNAME= dans /etc/sysconfig/network

VLANs

Pour supporter les VLANs il faut tout d'abord charger le module 802.1q :

```
modprobe 8021q
```

Ensuite il faut modifier le fichier de configuration de l'interface principale, exemple pour eth1 dont le fichier est /etc/sysconfig/network-scripts/ifcfg-eth1 :

```
DEVICE="eth1"
HWADDR="00:1A:4B:CE:D6:36"
TYPE="Ethernet"
ONBOOT="yes"
NM_CONTROLLED="no"
BOOTPROTO="none"
IPV6INIT="no"
IPV6_AUTOCONF="no"
```

Puis il faut créer le fichier correspondant à la sous interface, exemple pour le VLAN 2 sur eth1 dont le fichier sera /etc/sysconfig/network-scripts/ifcfg-eth1.2 :

```
DEVICE="eth1.2"
HWADDR="00:1A:4B:CE:D6:36"
TYPE="Ethernet"
ONBOOT="yes"
NM_CONTROLLED="no"
BOOTPROTO="static"
```

4. random.org est un site internet intéressant pour l'obtention de nombres et de chaines aléatoires puisqu'il utilise le bruit atmosphérique pour les générer.

```
VLAN=yes
IPADDR=XXX.XXX.XXX.XXX
NETMASK=XXX.XXX.XXX.XXX
GATEWAY=XXX.XXX.XXX.XXX
```

C.4 Installation de Solaris

C.4.1 Installation

L'installateur de Solaris se présente sous la forme d'une image ISO unique téléchargeable sur le site d'Oracle à l'adresse : http ://www.oracle.com/technetwork/server-storage/solaris11/downloads/index.html .

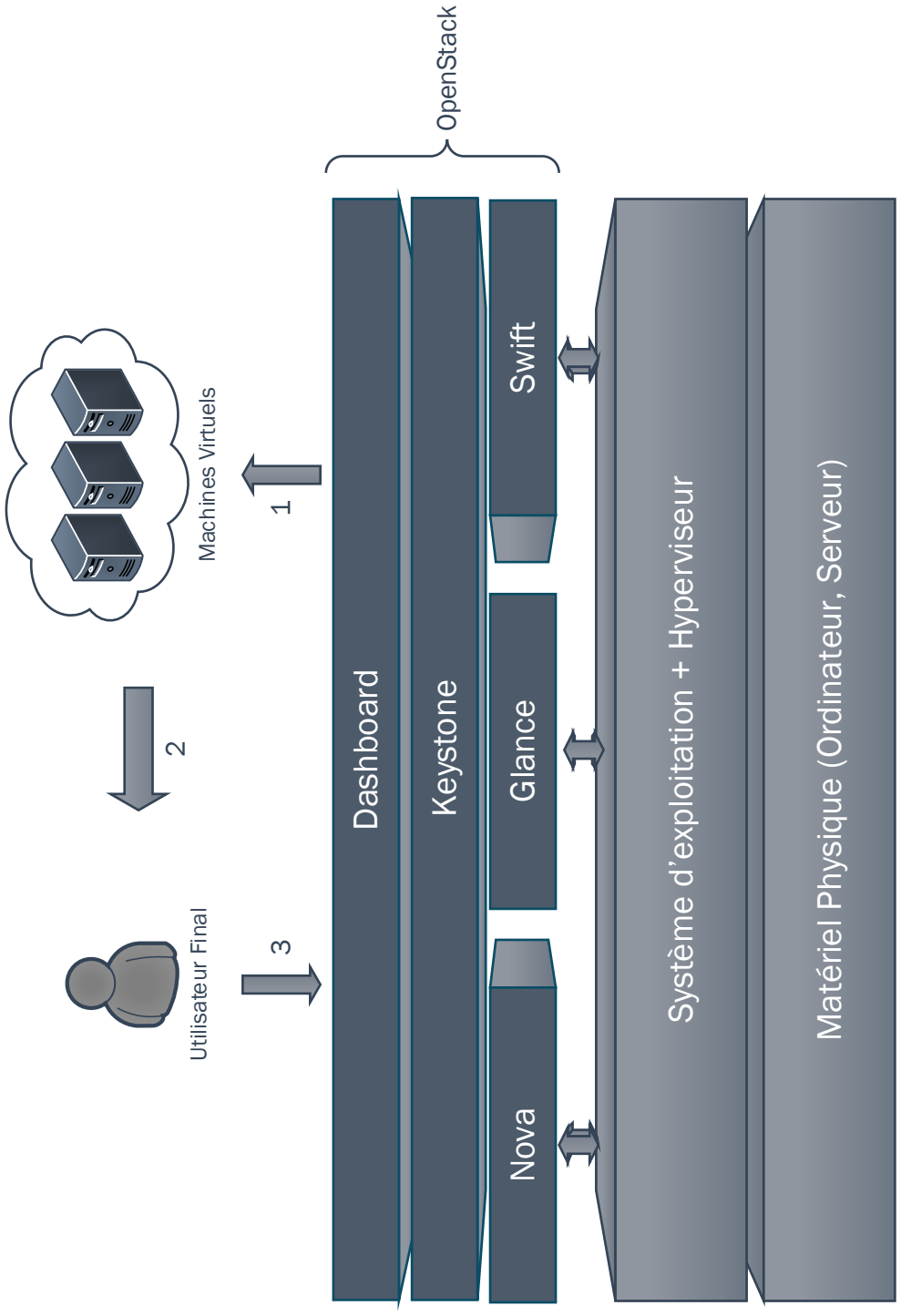
Il n'y as pas de flags de boot à passer, l'installation se fait en mode texte et Solaris reconnaît directement le multipath.
Le reste de l'installation est évident et ne nécessite pas d'explications.

D Diagrammes Réseau et schémas de principe

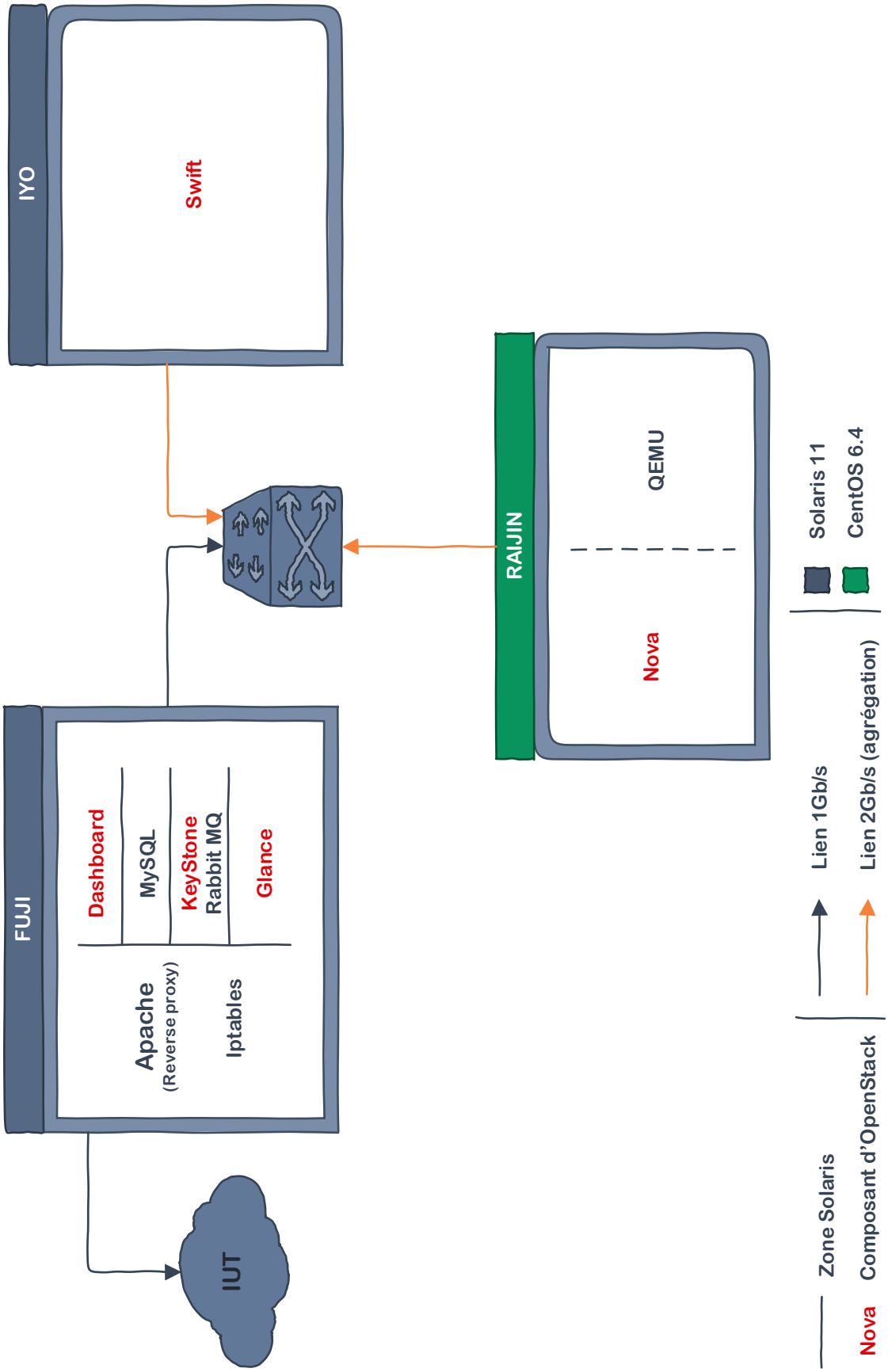
D.1 Les différents services d'OpenStack

1. L'utilisateur demande la création d'une ou plusieurs instance.
2. OpenStack se charge de les crées et retourne les informations pour y accéder à l'utilisateur
3. L'utilisateur peut utiliser son instance

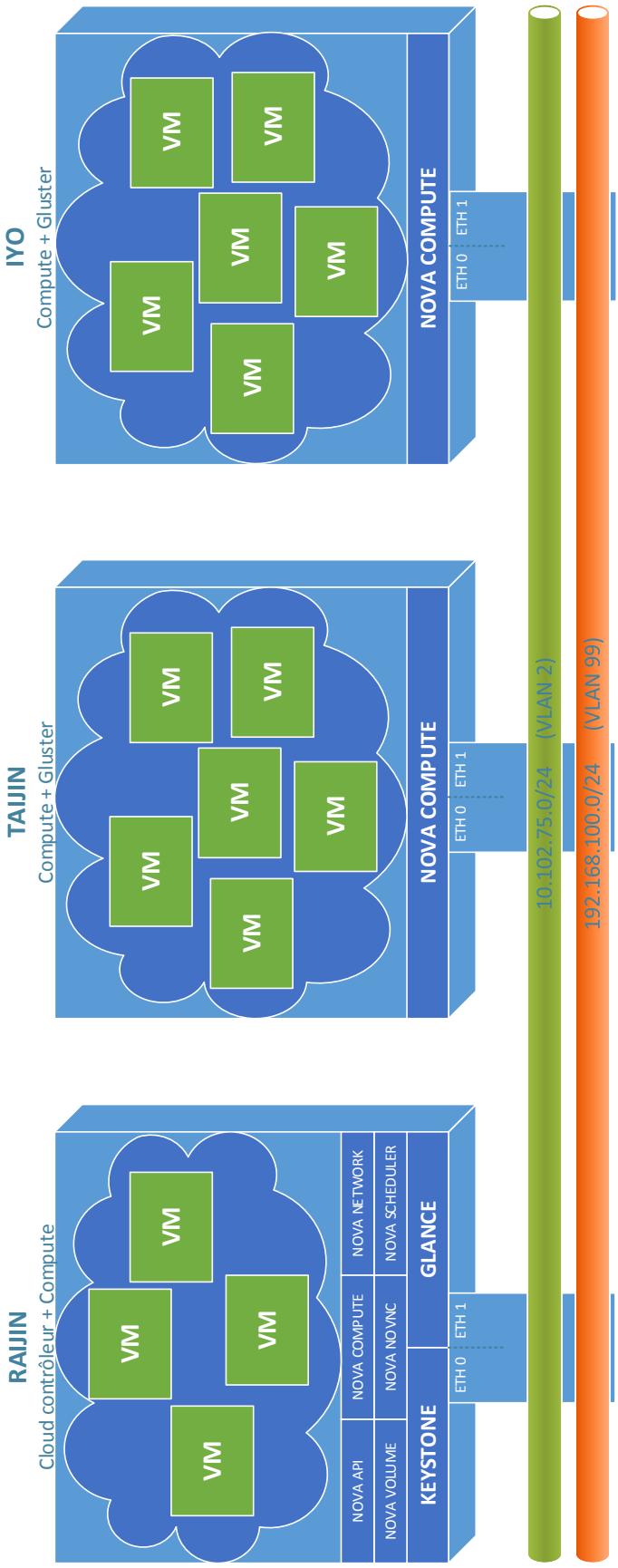
L'utilisateur final ne peut accéder qu'au dashboard.



D.2 Schéma global BladeCenter avec Solaris



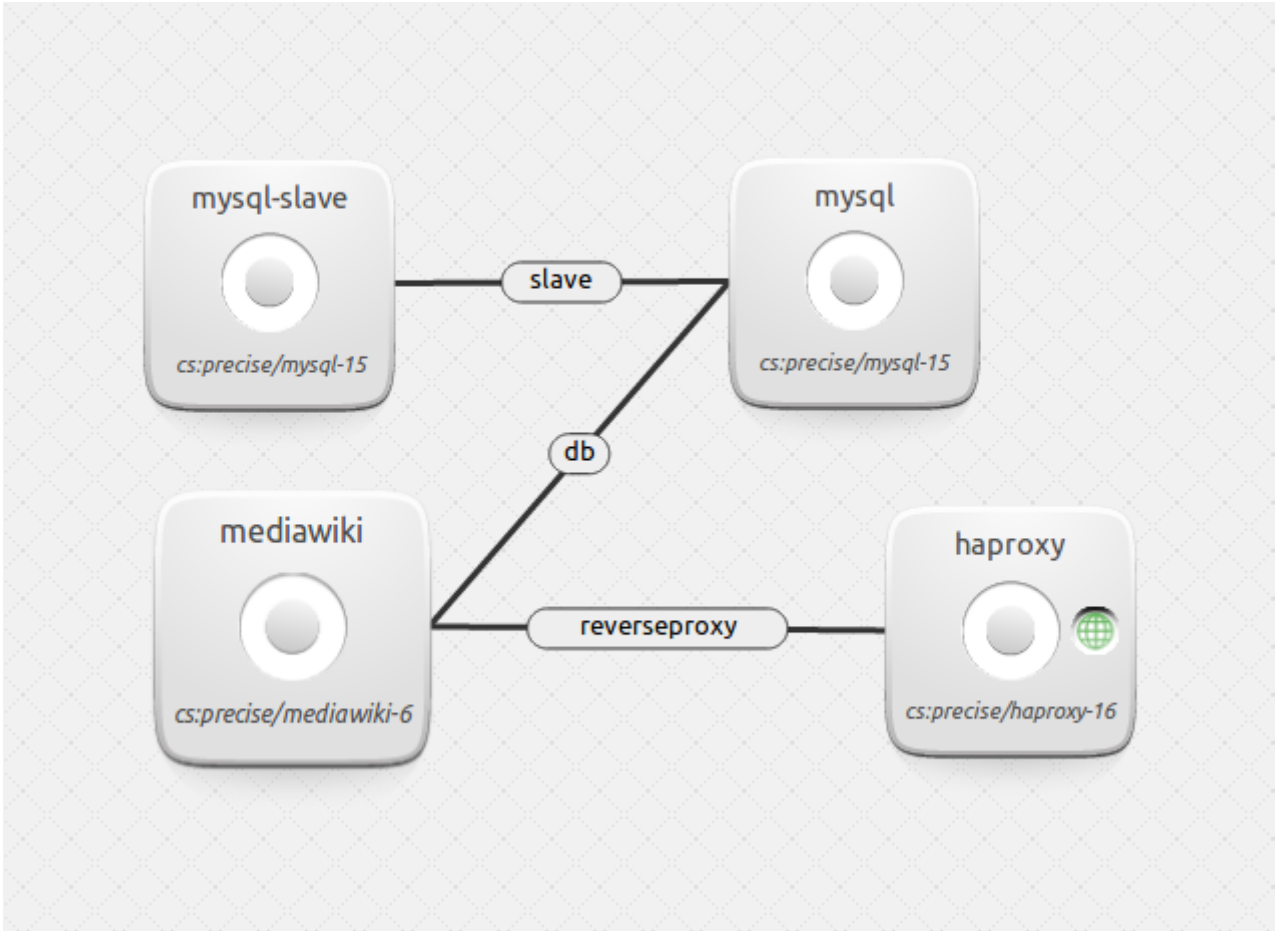
D.3 Schéma global BladeCenter avec QEMU et GlusterFS



E Scripts d'automatisation

E.1 Exemple de configuration de MediaWiki avec Juju

Notre but est de déployer MediaWiki et d'y ajouter quelques services de haute disponibilité. Nous allons donc créer deux serveurs web avec MediaWiki. Ils seront reliés à la même base MySql qui sera redondée avec une autre base esclave. La charge entre les deux serveurs web sera répartie par HAProxy. Le schéma ci-dessous présente la configuration finale vue depuis juju-gui, l'interface web de Juju. Le logo vert en forme de globe représente le monde extérieur.



On commence par installer Juju et ses dépendances (Juju est disponible dans les dépôts officiels d'Ubuntu mais on préfèrera utiliser la version issue de son Personal Package Archives (PPA).

```
sudo apt-get -y install python-software-properties
sudo add-apt-repository ppa:juju/pkgs
sudo apt-get update && sudo apt-get -y install juju charm-tools
```

On initialise ensuite Juju, on renseigne les identifiants de connection au Cloud dans ~/.juju/environments.yaml puis on relance le bootstap. [1]

```
juju bootstrap
# Edition de environments.yaml
juju bootstap
```

On installe maintenant la base MySql maître et MediaWiki. On ajoute ensuite une relation entre eux

1. Voir https ://juju.ubuntu.com/get-started/openstack/ pour des explications sur environments.yaml

```
juju deploy mysql
juju deploy mediawiki
juju add-relation mediawiki:db mysql
```

On ajoute une nouvelle base MySql (qu'on appelle mysql-slave) et on la lie avec la précédente

```
juju deploy mysql mysql-slave
juju add-relation mysql:master mysql-slave:slave
```

On crée une deuxième instance de notre MediaWiki

```
juju add-unit mediawiki
```

Et on ajoute un proxy pour répartir la charge entre les deux MediaWiki

```
juju deploy haproxy
juju add-relation haproxy mediawiki
```

Toute notre infrastructure est créée mais n'est pas encore accessible depuis internet. Par défaut les machines ne peuvent pas communiquer avec l'extérieur. Il faut donc rendre le proxy accessible Il suffit ensuite d'attendre que toutes les machines démarrent et que les ports s'ouvrent

```
juju expose haproxy
juju status
```

La dernière commande affiche un compte-rendu de toutes les machines créées et de tous les services installés. Il faut localiser la partie de HAProxy et regarder les ligne open-ports et public-address. Dans l'exemple ci-dessous, on voit que le port 80 est ouvert et qu'on peut accéder à la machine via l'ip 15.185.120.12. Si ces informations n'ont pas encore apparu, c'est que la machine n'a pas encore fini d'être configurée.

```
[...]
services:
  haproxy:
    charm: cs:precise/haproxy-16
    exposed: true
    relations:
      reverseproxy:
      - mediawiki
    units:
      haproxy/0:
        agent-state: started
        machine: 5
        open-ports:
        - 80/tcp
        public-address: 15.185.120.12
[...]
```

Voilà, notre réseau est terminé et fonctionnel. Si la charge augmente on peut ajouter des serveurs web, des bases mysql esclave, voir des HAProxy avec la commande *juju add-unit*
On peut aussi modifier la configuration des charmes. Par exemple, pour modifier le nom de notre MediaWiki, il suffit de faire *juju set mediawiki name='Lamas Wiki !'*
La liste complète des charmes et de leur configuration est disponible à l'adresse http ://jujucharms.com/charms.
On peut aussi créer ses propres charmes. Il est possible de les développer dans de nombreux langages tels que le Python, le Ruby ou même avec puppet.

E.2 Exemple de configuration avec Puppet

Notre but est de déployer une application sur un parc entier de machines. Nous allons copier automatiquement un fichier sur deux machines.

On considère que PuppetMaster est installé sur le serveur et que les clients ont Puppet installé aussi.

On commence par identifier le client auprès du serveur :
Sur le client

```
sudo puppetd -t -v -w 60
```

Et sur le serveur

```
sudo puppetca --list
  "dummy42" (D4:E7:37:A4:DF:F1:18:16:FA:D1:5B:38:CA:13:37:FB)
```

Cette commande nous montre la liste des machines qui se sont identifiées au serveur. Ici la machine "dummy42" s'est identifiée. On l'enregistre

```
sudo puppetca --sign dummy42
```

On ajoute une deuxième machine (dummy43 par example)

```
sudo puppetca --list
  "dummy43" (D4:E7:37:A4:DF:F1:18:16:FA:D1:5B:38:CA:13:37:FB)
sudo puppetca --sign dummy43
```

Maintenant on n'a plus qu'à activer Puppet au boot de la machine Dans /etc/default/puppet

```
# Start puppet on boot?
START=yes
```

Et de le démarrer en tâche de fond

```
sudo /etc/init.d/puppet start
```

Puppet gère le parc de machines grâce au ficher /etc/puppet/manifests/site.pp. C'est dans celui-ci que l'on définit l'adresse du PupperMaster et le fichier qui contient la définition des machines. On le remplit donc ainsi, en supposant que l'adresse ip du PuppetMaster soit 192.168.1.1.

```
filebucket { 'main': server => '192.168.1.1' }
File { backup => 'main' }

import "node"
```

Vient maintenant la définition des machines. On applique donc à dummy42 et dummy43 la configuration de vim. On écrit ensuite dans /etc/puppet/manifests/node.pp

```
node 'dummy42','dummy43 {
    include pushText
}
```

On crée maintenant notre module. Puppet nomme module, la liste des opérations à effectuer pour compléter une tâche, par exemple, l'installation d'apache ou, en l'occurrence, la copie d'un fichier.

On commence par créer l'arborescence de notre module.

```
mkdir -p /etc/puppet/modules/pushText/manifests
mkdir -p /etc/puppet/modules/pushText/files
```

Puis on définit les actions à faire. On crée un fichier puppet.txt dans /etc qui appartient à root et on lui applique les droits 644. Le fichier sera copié depuis /etc/puppet/module/pushText/puppet.txt

```
class dummy {
        file { "/etc/puppet.txt":
                owner => root,
                group => root,
                mode => 644,
                source => "puppet:///pushText/puppet.txt"
        }
}
```

On redémarre PuppetMaster pour qu'il prenne en compte les modifications

```
sudo service puppetmaster restart
```

Et il ne reste plus qu'à forcer la mise à jour sur les clients.

```
sudo puppetd -t -v
```

Ici, nous avons effectué une tâche très simple, mais puppet nous permet de faire plein de choses beaucoup plus poussées. On peut installer toutes sortes de logiciels tel que Apache, Munin ou encore Wordpress. On peut même installer des modules d'Openstack. Le client Puppet est disponible sur beaucoup de plateformes comme Linux, Windows et Mac. Toutes les recettes sont recensées dans la Forge Puppet [2] et peuvent être installés facilement. Par exemple, la figure ci-dessous montre une partie de la recette d'installation de vim, un célèbre éditeur de textes sous linux.

```
class vim inherits vim::params {
  exec { 'update-alternatives':
    command => 'update-alternatives --set editor /usr/bin/vim.basic',
    unless  => 'test /etc/alternatives/editor -ef /usr/bin/vim.basic',
  }

  file { '/etc/vim/vimrc':
    owner   => 'root',
    group   => 'root',
    mode    => '0644',
    source  => "puppet:///modules/vim/\${::lsbdistcodename}/etc/vim/vimrc",
    notify  => Exec['update-alternatives'],
    require => Package['vim'],
  }

  package { 'vim':
    ensure => present,
  }
}
```

Exemple du manifeste de la recette de vim

2. http ://forge.puppetlabs.com/

F Fichiers de configuration OpenStack

Ne sont inclus ici que le fichiers de configurations que nous avons modifié par rapport à l'installation par défaut d'OpenStack Folsom sur Centos 6.3.

F.1 Keystone

F.1.1 keystone.conf

```
[DEFAULT]
# /etc/keystone/keystone.conf
# 02/03/2013

admin_token = e2e2f51aaf1ba336c736

bind_host    = 0.0.0.0
admin_port   = 35357
public_port  = 5000
compute_port = 8774

# === Logging Options ===
verbose = True
debug   = False

log_file = keystone.log
log_dir  = /var/log/keystone

[sql]
connection = mysql://keystone:xxxxxx@raijin/keystone

[identity]
driver = keystone.identity.backends.sql.Identity

[catalog]
template_file = /etc/keystone/default_catalog.templates
driver = keystone.catalog.backends.sql.Catalog

[token]
driver = keystone.token.backends.sql.Token

[ec2]
driver = keystone.contrib.ec2.backends.sql.Ec2

[ssl]
enable = False

# Par soucis de place l'ensemble des sections [filter:] a été omise,  il suffit de recopier
la partie correspondante du fichier de configuration d'origine
```

F.2 Glance

F.2.1 glance-api.conf

```
[DEFAULT]
# /etc/glance/glance-api.conf
# 03/03/2013

verbose = True
debug = False
log_file = /var/log/glance/api.log

sql_connection = mysql://glance:xxxxxx@raijin/glance
sql_idle_timeout = 3600

workers = 1
backlog = 4096
default_store = file
bind_host = 0.0.0.0
bind_port = 9292

enable_v1_api = True
enable_v2_api = True

# ============ Registry Options ===============================
registry_host = 0.0.0.0
registry_port = 9191
registry_client_protocol = http

# ============ Notification System Options =====================
notifier_strategy = noop

qpid_notification_exchange = glance
qpid_notification_topic = glance_notifications
qpid_host = localhost
qpid_port = 5672
qpid_heartbeat = 5
# Set to 'ssl' to enable SSL
qpid_protocol = tcp
qpid_tcp_nodelay = True

# ============ Filesystem Store Options ========================
filesystem_store_datadir = /var/lib/glance/images/

# ============ Delayed Delete Options ============================
delayed_delete = False
scrub_time = 43200
scrubber_datadir = /var/lib/glance/scrubber

# =============== Image Cache Options =============================
image_cache_dir = /var/lib/glance/image-cache/

[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = xxxxxx

[paste_deploy]
flavor = keystone
config_file = /etc/glance/glance-api-paste.ini
```

F.2.2 glance-api-paste.ini

```
[...]
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
delay_auth_decision = true
admin_tenant_name = service
admin_user = glance
admin_password = xxxxxx
[...]
```

F.2.3 glance-cache.conf

```
[...]
[filter:authtoken]
admin_password = xxxxxx
admin_tenant_name = service
admin_user = glance
[...]
```

F.2.4 glance-registry.conf

```
[...]
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = xxxxxx

[paste_deploy]
flavor = keystone
config_file = /etc/glance/glance-registry-paste.ini
[...]
```

F.2.5 glance-registry.conf

```
[...]
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
admin_tenant_name = service
admin_user = glance
admin_password = xxxxxx
```

F.3 Nova

F.3.1 nova.conf

Note : ne pas oublier de remplacer my_ip et vncserver_proxyclient_address par l'IP de la lame

```
[DEFAULT]
# /etc/nova/nova.conf
# 02/03/2013

# Général
verbose = True
logdir = /var/log/nova
state_path = /var/lib/nova
lock_path = /var/lib/nova/tmp
sql_connection = mysql://nova:xxxxxx@raijin/nova
my_ip = 10.102.75.100

# Hosts
network_host = 10.102.75.100
metadata_host = 10.102.75.100
glance_host = raijin
qpid_hostname = raijin

# Virtualisation
## Drivers
compute_driver = libvirt.LibvirtDriver
libvirt_type = qemu
# libvirt_vif_type=ethernet
# libvirt_use_virtio_for_bridges = true
# libvirt_vif_driver = nova.virt.libvirt.vif.QuantumLinuxBridgeVIFDriver
# linuxnet_interface_driver = nova.network.linux_net.QuantumLinuxBridgeInterfaceDriver

# Réseau
## VLAN Manager
network_manager = nova.network.manager.FlatDHCPManager
fixed_range = 192.168.100.0/24
flat_network_bridge = br100
flat_interface = eth0.99
public_interface = eth0.2
network_size = 256

## DHCP
dhcpbridge = /usr/bin/nova-dhcpbridge
dhcpbridge_flagfile = /etc/nova/nova.conf
force_dhcp_release = False

## Autre
injected_network_template = /usr/share/nova/interfaces.template
libvirt_nonblocking = True
libvirt_inject_partition = -1
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver

# Stockage bloc
iscsi_helper = tgtadm
volumes_dir = /etc/nova/volumes

## Scheduler
compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
scheduler_available_filters=nova.scheduler.filters.standard_filters
scheduler_default_filters=RamFilter

rpc_backend = nova.openstack.common.rpc.impl_qpid
rootwrap_config = /etc/nova/rootwrap.conf
auth_strategy = keystone
```

```
# VNC (All)
vncserver_listen = 0.0.0.0
novncproxy_base_url = http://10.102.75.100:6080/vnc_auto.html
xvpvncproxy_base_url = http://10.102.75.100:6081/console

# VNC (Par host)
vncserver_proxyclient_address = 10.102.75.100

[keystone_authtoken]
admin_tenant_name = service
admin_user = nova
admin_password = xxxxxx
auth_host = raijin
auth_port = 35357
auth_protocol = http
signing_dir = /tmp/keystone-signing-nova
```

Bibliographie

[1] Duncan HARDIE. *How to Get Started Creating Oracle Solaris Zones in Oracle Solaris 11*. Nov. 2011. URL : http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-092-s11-zones-intro-524494.html.

[2] *Juju - Get Started*. Canonical. 2013. URL : https://juju.ubuntu.com/get-started/.

[3] Masoud KALALI. *Configuring Solaris Link Aggregation*. Nov. 2010. URL : http://kalali.me/configuring-solaris-link-aggregation-ethernet-bonding/.

[4] Masoud KALALI. *Writing Solaris SMF service manifest*. Jan. 2011. URL : http://kalali.me/authoring-solaris-service-management-facility-smf-service-manifest/.

[5] *OpenStack Hypervisor support matrix*. URL : https://wiki.openstack.org/wiki/HypervisorSupportMatrix.

[6] *OpenStack Install and Deploy Manual - Red Hat*. OpenStack Foundation. Nov. 2012. URL : http://docs.openstack.org/folsom/openstack-compute/install/yum/content/.

[7] *OpenStack Install and Deploy Manual - Ubuntu*. OpenStack Foundation. Nov. 2012. URL : http://docs.openstack.org/folsom/openstack-compute/install/apt/content/.

[8] John QUAGLIERI. *CentOS 6.2 and libvirt startup issues*. Déc. 2011. URL : http://quags.net/archives/53.

[9] *Red Hat Enterprise Linux 6 Deployment Guide*. Red Hat. 2013. URL : https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/.

[10] Nil SANYAS. *Le streaming capte un tiers du trafic en Europe, et 50% aux USA*. Mai 2011. URL : http://www.pcinpact.com/actu/news/63619-streaming-youtube-netflix-europe-usa.htm.

Glossaire

***BSD** Désigne une famille de systèmes d'exploitation dérivées d'UNIX. 5

Apache Serveur HTTP. C'est le plus utilisé sur Internet. 9

API Application Programming Interface. 5, 8

BladeCenter Originellement nom donné à la gamme des serveur blades d'IBM. Désigne par extension tout les systèmes de serveur lames. 6, 7, 12

CfEngine Logiciel permettant l'automatisation de la configuration de serveurs et de postes de travail écrit en C. 9

Chef Logiciel permettant l'automatisation de la configuration de serveurs et de postes de travail écrit en Ruby. 9

Cloud Computing Désigne l'accès via un réseau à un ensemble de ressources informatiques partagées. 1, 4, 7, 16

Cloud privé Infrastructure de virtualisation dont l'usage est réservé à une entreprise et qui le plus souvent géré et installé par celle-ci. 1, 4–6, 17, 20

Cloud public Infrastructure de virtualisation accessible publiquement moyennant éventuellement un paiement. 5

CloudStack Solution permettant de créer des IaaS créé par Citrix et Cloud.com. 5

coeur En informatique un coeur est une unité de calcul. 5

CRM Customer Relationship Management. 5

Dropbox Service de stockage et de partage de fichiers en ligne. 5

EBS Elastic Block Store. Service de stockage bloc d'Amazon pour son IaaS, EC2. 5

EC2 Elastic Compute Cloud. IaaS d'Amazon. 5

Failover Technique consistant à basculer sur un autre équipement en cas de défaillance. 7

FC Fibre Channel. 7, 12

GlusterFS Système de fichiers distribués. 9

Google App Engine PaaS de Google. Supporte Python, Java, et le Go. 5

Google Apps for Business Suite d'outils pour les entreprises (e-mails, agenda, bureautique) disponible via Internet. 5

HAProxy Logiciel de répartition de charge. 9

Heroku PaaS de Salesforce. Supporte Ruby, Python, Java, Node.js, Clojure, et Scala. 5

Hyper-V Système de virtualisation de Microsoft présent dans Windows Server depuis la version 2008. 6

IaaS Infrastructure as a Service. Modèle du Cloud Computing où une infrastructure, potentiellement externe, est mis à disposition du client. 4–6, 8, 9

iLO Integrated Lights-Out. Système de gestion *out-of-band* d'HP pour ses serveurs. 21

Instagram Application de partage de photos pour iOS et Android. 5

Java Langage de programmation orienté objet. Utilisé pour sa portabilité car la machine virtuelle Java (JVM) sur laquelle il est basé est disponible sur de nombreuses plateformes. D'après l'indice TIOBE, c'est le langage le plus populaire début 2013. 5

Juju Gestionnaire de paquet pour le cloud computing. 9

KVM Keyboard, Video, Mouse. 21

KVM Kernel-based Virtual Machine. Système de virtualisation libre basé sur QEMU et utilisant les extensions processeurs de virtualisation. Très performant et supporte tout les systèmes en tant qu'invités. 6, 8, 11

LDAP Lightweight Directory Access Protocol. Protocole de communication avec les annuaires respectant la norme du même nom. Généralement utilisé pour l'authentification des utilisateurs. 6, 8

licence Apache Licence autorisant la modification et la redistribution du code tout en obligeant le maintien du copyright. 5

Linux Système d'exploitation libre basé sur le noyau éponyme. 4, 5, 11

Load balancing Technique permettant de distribuer la charge sur plusieurs liens afin de maximiser la bande passante. 7, 9

LXC LinuX Containers. Système de virtualisation léger intégré au noyau Linux qui permet de créer des environnements exécution isolés. Consomme très peu de ressources et très performant mais ne supporte que les systèmes Linux. 6, 8

MediaWiki CMS permettant la création de Wiki. 9

Microsoft Azure IaaS et PaaS de Microsoft. 5

Multipath Technique permettant d'utiliser plusieurs chemins physiques pour accéder à du stockage afin d'augmenter la bande passante et de fournir de la redondance. 7

MySQL Serveur de base de données. 9

NAT Network Address Translation. Mécanisme permettant à des équipements possédant une IP privé de communiquer avec des hôtes situés sur un réseau publique. 12

Netflix Service de streaming de films sur Internet. 5

NFS Network File System. Protocole qui permet d'accéder à des fichiers via le réseau. 9

Node.js Logiciel basé sur la machine virtuelle JavaScript V8 de Google permettant d'exécuter du JS côté serveur et ainsi d'écrire des applications web scalables grâce, notamment, aux E/S asynchrones. 5

Open Source Terme s'appliquant aux logiciels dont la licence respecte la possibilité de libre de redistribution et d'accès au code source. 5

OpenStack Solution permettant de créer des IaaS créé par Rackspace et la NASA. 1, 6–9, 11, 16, 20

OVH Hébergeur Français. 7

PaaS Platform as a Service. Modèle du Cloud Computing où un environnement d'exécution est mis à disposition du client pour ses propres applications. 4, 5

PAM Pluggable Authentication Modules. Système permettant d'intégrer différent schémas d'authentification à un système UNIX/Linux de façon transparente pour les applications. 8

PHP PHP : Hypertext Preprocessor. Langage de programmation principalement utilisé avec un serveur HTTP pour la création de sites et d'applications web.. 5

PPA Personal Package Archives. Dépôts non-officiel mise à disposition des développeurs de logiciel libre par la plateforme Launchpad. 29

Puppet Logiciel permettant l'automatisation de la configuration de serveurs et de postes de travail écrit en Ruby. 9

Python Langage de programmation orienté objet. 5, 6, 9, 30

QEMU Logiciel de virtualisation libre pour Linux. 8

Rackspace Hébergeur et fournisseur de solutions de Cloud Computing Américain. Créateur d'OpenStack. 9

RAID 0 Technique consistant à séparer les données. 7

Ruby Langage de programmation orienté objet inspiré du Smalltalk et de Perl. Principalement connu pour la création d'applications web grâce au framework Rails. 5, 30

S3 Simple Storage Service. Service de stockage objet d'Amazon. Offre une stockage virtuellement illimité dans le cloud. 5

SaaS Software as a Service. Modèle du Cloud Computing où l'on paye un abonnement pour utiliser un logiciel qui peut ne pas être présent physiquement sur notre ordinateur. 4, 5

Salesforce Editeur de logiciel Américain très présent dans le domaine du Cloud Computing. 5

Salesforce Sales Cloud Outil de CRM disponible via Internet. 5

SAN Storage area network. 7, 12

Scala Langage de programmation inspiré de Java. 5

service d'orchestration Programme qui permet d'automatiser la coordination et l'organisation de systèmes complexe. 9

Shazam Logiciel de reconnaissance musicale. 5

SNMP Simple Network Monitoring Protocol. Protocole qui permet de gérer et de superviser des équipements réseaux. 10

Solaris Système d'exploitation UNIX développé par Oracle. 5, 12

SQL Structured Query Language. Language permettant d'effectuer des requêtes sur des bases de données.. 8

SSH Secure SHell. Protocole de communication sécurisé permettant le transfert de fichiers entre ordinateurs ou l'administration de serveurs à distance. 21, 24

stockage bloc Dans le cloud computing désigne le stockage qui est proche des disques physiques et qui offre de bonnes performances. 5

stockage objet Dans le cloud computing désigne un type de stockage ne mettant pas l'accent sur la performance mais sur la facilité d'utilisation grâce à une API. 5

switch Équipement réseau de couche 2 permettant de connecter d'autres équipements entre eux. 7, 10, 12

Telnet Protocole permettant d'échanger très simplement des données entre deux ordinateurs. 10

tunnel SSH Un tunnel SSH permet d'encapsuler le trafic IP dans une connexion SSH afin, par exemple, d'accéder à des machines distantes dont l'accès serait bloqué par un pare-feu. 21

Ubuntu Système d'exploitation libre soutenu par Canonical. 9

Varnish Serveur de cache HTTP. 9

VirtualBox Logiciel de virtualisation de bureau développé par Oracle. 4

virtualisation En informatique, la virtualisation consiste à faire fonctionner plusieurs systèmes d'exploitations, ou plusieurs applications isolées les une des autres, sur un seul ordinateur. 1, 4, 6–8

VLAN Virtual LAN. Norme permettant d'isoler des réseaux à la couche 2 en utilisant la norme 801.1q afin d'ajouter un identifiant de réseau virtuel au début de la trame Ethernet. 7

Windows Système d'exploitation de Microsoft. 4, 5, 21

Windows Server Version serveur du système d'exploitation de Microsoft. 22

Xen Logiciel de virtualisation libre. 5, 6, 8

Résumé

Derrière le mot Cloud Computing se trouvent de nombreux thèmes de l'informatique tels que la virtualisation, la haute disponibilité, et la scalabilité. La mise en place d'un Cloud privé est donc un sujet d'étude particulièrement intéressant en Réseaux & Télécommunications.
A l'aide d'OpenStack nous avons mis en place une Infrastructure en tant que Service, ce qui nous as permis de mieux comprendre l'interaction entre les différents service (Réseau, Stockage, Virtualisation). L'installation a été effectué sur des systèmes UNIX (Solaris) et Linux (CentOS, Ubuntu Server) dont les choix sont expliqués dans ce rapport.
En plus des services d'OpenStack nous avons intégrés une gestion de configuration automatique pour les instances virtuelles à l'aide de Puppet. La mise en place de monitoring à l'aide de Shinken et Munin est également détaillé.

Behind what we call Cloud Computing there is a lot of interesting topics like virtualization, high-disponibility, and scalability. This make deploying a Private Cloud an interesting studies subject in Networking & Telecommunications.
We used OpenStack, an open-source software, to create an Infrastructure as a Service, this gave us a better understanding of service's interaction (Networking, Storage, Virtualization). Setup was made on both UNIX (Solaris) and Linux (CentOS, Ubuntu Server) systems whose choice are explained in this report.
In additions of OpenStack services we used Puppet to deliver automated configuration deployment to VMs. Monitoring with Shinken and Munin is also detailled.

Mots clés : Cloud Computing, Virtualisation, Réseau, OpenStack, Linux, UNIX, BladeCenter, Puppet, Juju, Shinken, Monitoring
Keywords : Cloud Computing, Virtualization, Networking, OpenStack, Linux, UNIX, BladeCenter, Puppet, Juju, Shinken, Monitoring