h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·SO GENÈVE

Haute Ecole Spécialisée
de Suisse occidentale

Travail de Bachelor 2012

Cloud Management

Professeur : Mr. Gérald Litzistorf
Etudiant : Mr. Benoît Chalut
E-mail : benoitchalut@gmail.com

Date : Juin 2012

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Printemps 2012
Session de bachelor

SECURITE DES SYSTEMES D'INFORMATION

CLOUD MANAGEMENT

Descriptif :

La société http://www.smartbee.ch cherche une solution de gestion de son Cloud privé capable de supporter les hyperviseurs ESXi – KVM, la gestion réseau VLAN – firewall et la haute disponibilité (vMotion – LiveMigration – vStorage – NAS iSCSI)

Travail demandé :

1. Evaluer les solutions http://www.ovirt.org/ et http://opennebula.org/
   Identifier les fonctions intéressantes
   Comparer (avantages - inconvénients) de ces 2 solutions
   Estimation = 2-3 semaines

2. Etudier la solution http://openvswitch.org/
   Préciser les avantages par rapport à solution réseau classique (TAP) de KVM
   Configuration VLAN
   Configuration NIC bonding
   Estimation = 3 semaines

3. Configurer live migration pour KVM → http://www.linux-kvm.org/page/Migration
   Utiliser le module QNAP iSCSI du labo
   Indiquer les différentes variantes
   http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/chap-Virtualization-KVM_live_migration.html
   Mesure de performances
   Estimation = 2 semaines

4. Thème à choix si le temps le permet

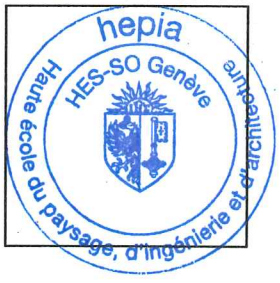
Sous réserve de modification en cours du travail de Bachelor

Candidat :
M. CHALUT BENOIT-GEORGES
Filière d'études : Télécommunications
Domaine de formation TIC

Professeur(s) responsable(s) :
Litzistorf Gérald

En collaboration avec : SmartBee
Travail de bachelor soumis à une convention de stage en entreprise : non
Travail de bachelor soumis à un contrat de confidentialité : non

Timbre de la direction



h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Printemps 2012
Session de bachelor

Résumé : CLOUD MANAGEMENT

Le but de mon travail de Bachelor a été de concevoir une architecture de Cloud IaaS (Infrastructure as a Service) Open-Source basée sur un environnement virtualisé. Le prestataire de service va accueillir plusieurs clients d'horizons différents au sein de son Cloud, qui possédera les caractéristiques suivantes:

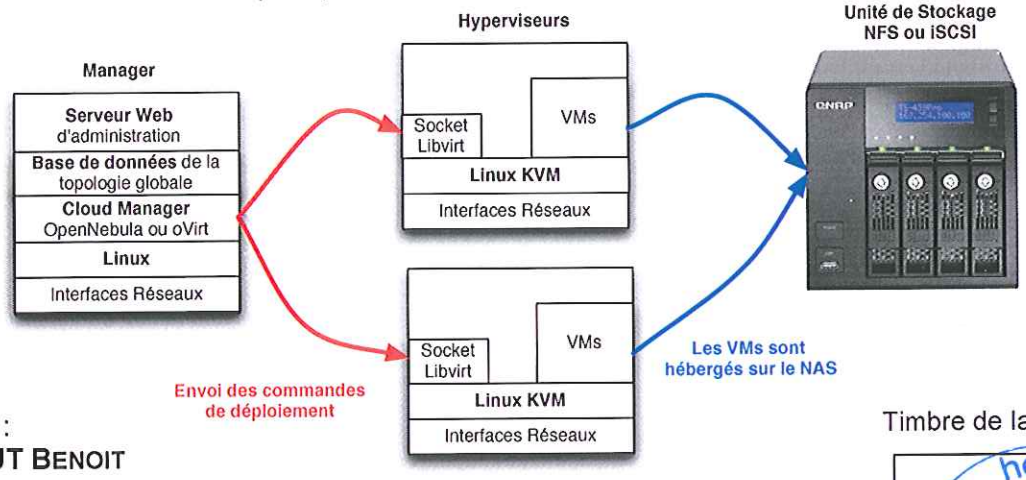
- Virtualisation : Linux-KVM
- Isolation des VMs : Utilisation des VLANs
- NAS : QNAP ts-459 Pro II
- Live Migration : Migration d'une VM d'un hyperviseur à l'autre sans interruption de services
- Outils de Management : Manager pour l'instanciation des VMs, gestion des ressources

Au cours de mon travail j'ai étudié trois technologies Open Source permettant de gérer mon infrastructure virtualisée :

- oVirt (Open Virtualisation) :
  - Permet de déployer des VMs sur des hyperviseurs (KVM)
  - Interface GUI intuitive via une interface web
- OpenNebula :
  - Possède les mêmes fonctions de base que oVirt
  - Permet l'intégration d'hyperviseur ESXi
  - Peut être également administré via CLI
  - Permet l'attribution d'un VLAN ID à chaque VMs déployées
- Open vSwitch :
  - S'intègre au noyau Linux et offre une compatibilité avec OpenNebula et KVM
  - Alternative au module Bridge de KVM
  - S'administre via CLI sur l'hyperviseur
  - Permet la propagation des VLANs sur un switch physique norme 802.1Q

Ces trois technologies m'ont permis de me plonger dans le monde Open-Source. oVirt, de par sa simplicité de mise en place m'a permis de comprendre les différents éléments d'une architecture IaaS. Tandis que grâce à OpenNebula et Open vSwitch, j'ai pu approfondir les problématiques liées à l'élaboration d'une solution fiable et sécurisée, malgré que le déploiement n'ait pas toujours été trivial.

Au cours des 8 semaines du travail de Bachelor, j'ai pu mettre en œuvre plusieurs scénarios au cours desquels j'intégrais de plus en plus d'éléments liés à l'élaboration d'une topologie sécurisée. Mes différentes infrastructures se sont basées sur le schéma de principe ci-dessous:



Diplômant :
M. CHALUT BENOIT
Classe : TE3
Filière d'études : Télécommunications
Ingénierie des Technologies de l'Information

Timbre de la direction

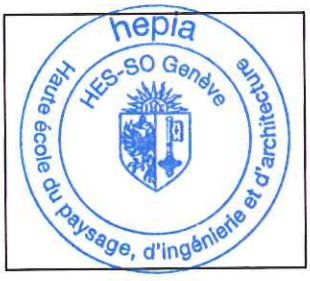


Table des matières :

1 Enoncé 2

2 Résumé 3

3 Avant-propos 6
3.1 Remerciements 6
3.2 Règles typographiques 6
3.3 Lexique 6

4 Analyse: 7
4.1 Introduction 7
4.2 Différents types de Cloud 7
4.2.1 Qui dit Cloud dit virtualisation 8
4.3 Le Cloud management 8
4.4 Contexte pratique au sein de SmartBee 10
4.4.1 Schéma de principe du réseau de la société SmartBee 10
4.5 Contraintes 10
4.5.1 Solutions utilisées 11
4.6 Technologies n°1 : oVirt 11
4.6.1 oVirt Engine Manager 11
4.6.2 oVirt Node Hyperviseur 12
4.7 Technologie n°2 : OpenNebula 13
4.7.1 Le Manager (Front-end) 13
4.7.2 Les hyperviseurs (Host) 14
4.8 Communication réseau des VMs 14
4.8.1 NAT (Network Address Translation) 14
4.8.2 Bridge 14
4.8.3 Bridge KVM 15
4.8.4 Les VLANs 15
4.9 Technologie n°3 : Open vSwitch 16
4.10 Solutions de stockages 17
4.10.1 Cible iSCSI sous Fedora 17
4.10.2 NAS (Network Area Storage) QNAP TS-459 Pro II 17
4.11 Live Migration KVM 19
4.12 Scénarios proposés 20
4.12.1 Déploiement de oVirt avec Windows Server 2008 virtualisé 20
4.12.2 Mise en place d'une VM sous KVM grâce à OpenNebula 21
4.12.3 Isolation des VMs grâce aux VLANs layer 1 gérés par Open vSwitch 22
4.12.4 Scénario final n°4: Isolation des VMs + Live Migration + NAS + Supervision 23

5 Réalisation 25
5.1 Scénario n°1 : Déploiement de oVirt avec Windows Server 2008 virtualisé 25
5.1.1 Schéma global de l'infrastructure oVirt 25
5.1.2 Introduction 25
5.1.3 Installation de la cible iSCSI 25
5.1.4 Déploiements du Manager (oVirt Engine) 26
5.1.5 Déploiement de l'hyperviseur (oVirt Node) 27
5.1.6 Mise en place du Datacenter 28
5.1.7 Déploiement de la VM Windows Server 28
5.1.8 Configuration du rôle Active Directory 32
5.1.9 Intégration d'un poste client au domaine 32
5.2 Conclusion à propos de la solution oVirt 33
5.3 2e Scénario : Déploiement d'un VM avec OpenNebula 34
5.3.1 Schéma global de OpenNebula 34
5.3.2 Introduction : 34
5.3.3 Activation du partage NFS du QNAP 34

5.3.4 Installation de OpenNebula sur le manager (front.cumulus) 35
5.3.5 Installation de KVM sur l'hyperviseur (host.cumulus) 35
5.3.6 Communication entre le Manager et l'hyperviseur KVM 35
5.3.7 Stockage et Datastores 36
5.3.8 Configuration de OpenNebula 36
5.3.9 Déploiement de l'infrastructure 37
5.3.10 Affichage de la console de la VM avec vncwiever 38
5.4 Scénario n°3 : Isolation des VMs 39
5.4.1 Schéma Global 39
5.4.2 Introduction 39
5.4.3 Installation de Open vSwitch 39
5.4.4 Démarrage du module Open vSwitch 40
5.4.5 Déploiement des 4 VMs 40
5.4.6 Connexion des 4 VMs au bridge et configuration des VLANs 40
5.4.7 Automatisation de l'attribution des VLANs 41
5.4.8 Configuration du Switch Cisco 24 ports Catalyst 2900 41
5.4.9 Tests 42
5.5 Scénario final n°4: Isolation des VMs + Live Migration + Supervision 43
5.5.1 Schéma global 43
5.5.2 Switch physique 43
5.5.3 Utilisation du NFS au lieu du iSCSI 44
5.5.4 Installation du deuxième hyperviseur 44
5.5.5 Partage des clés SSH pour la Live Migration 44
5.5.6 Configuration de Open vSwitch 44
5.5.7 Configuration des VLANs 44
5.5.8 Déploiement des VMs 46
5.5.9 Configurations des serveurs DHCP et Samba sous Windows Server 2008 46
5.5.10 Configuration de la VM de Management 47
5.5.11 Les Tests 48
5.5.1 Analyses des risques du scénario 51
5.6 Conclusion à propos de la solution OpenNebula 51

6 Difficultés rencontrées 52
6.1 Scénario 1 : oVirt 52
6.1.1 Bibliothèque ISO 52
6.1.2 Console d'affichage Spice sous Windows 7 52
6.1.3 Gestion des cartes réseaux physiques sur l'hyperviseur 52
6.1.4 Gestion des réseaux virtuelles 53
6.2 Scénario 2 : Déploiement d'une VM via OpenNebula 53
6.2.1 Installation du manager 53
6.2.2 Installation de l'Hyperviseur 53
6.2.3 Gestion du Datastore de l'hyperviseur 53
6.2.4 Communication Manager <-> Hyperviseur 53
6.3 Scénario 3 : Isolation des VMs avec Open vSwitch 54
6.3.1 Exécution de Open vSwitch 54
6.3.2 Greffes des vnets au bridge 54
6.3.3 Port Trunk 54
6.4 Scénario 4 : 54
6.4.1 Réponses aux requêtes Pings 54

7 Conclusion 55

8 Déroulement du projet 56

9 Liens & références 56

A Annexes 57

3 Avant-propos

3.1 Remerciements

Je tiens à remercier Monsieur Gérald Litzistorf, pour ses précieux conseils qui m'ont permis de mener à bien ce travail de Bachelor. Je remercie également la société SmartBee, d'avoir proposé ce sujet de Bachelor fort intéressant, ainsi que toute la classe TE3.

3.2 Règles typographiques

Voici quelques règles typographiques utilisées tout au long du rapport :

- ***Chemins des fichiers*** : En gras + italiques
- Liens internet : Bleus + soulignés

3.3 Lexique

- Le terme VM : Signifie « Machine Virtuelle » ou « Virtual Machine »
- Le terme PC : Signifie un ordinateur physique

4 Analyse:

4.1 Introduction

Depuis quelques années, le Cloud Computing a fait son apparition au sein des réseaux informatiques. Cette technologie a pour but de dématérialiser les éléments informatiques. C'est-à-dire, que les serveurs ne sont plus stockés au sein même des entreprises, mais sont gérés par une société spécialisée dans le Cloud Computing. Ces sociétés vont proposer des services à leurs clients. Cette solution permet aux entreprises clientes de ne plus se soucier de la maintenance de leurs serveurs et des backups, car elles vont uniquement se connecter aux plateformes situées dans le Cloud.

Mon Projet de Bachelor va se baser sur la mise en place d'un Cloud privé permettant de supporter plusieurs types d'hyperviseurs, une gestion des VLAN entre les VMs au sein d'un environnement Open-Source. Une étude sera également menée au niveau des moyens de stockage. Mon Travail de Bachelor se fait en collaboration avec la société SmartBee, qui est spécialisée dans la prestation de services externalisés, elle va donc fournir à ces clients des services tels que des domaines Active Directory, ou des services de messagerie.

4.2 Différents types de Cloud

Le Cloud Computing a changé radicalement la manière de gérer le réseau informatique des entreprises. Auparavant les sociétés possédaient au sein de leurs bâtiments de vastes locaux informatiques dédiés à l'hébergement de leurs infrastructures informatiques. Elles s'occupaient de A à Z de la gestion des serveurs.
Avec la démocratisation d'Internet et de l'augmentation des débits proposés aux abonnés, de nouvelles sociétés se créèrent en se spécialisant dans l'hébergement des serveurs physiques, afin de proposer à leurs clients des services des bases de données ou de gestions de VMs. Donc de plus en plus d'entreprises choisissent d'externaliser leurs infrastructures, afin de ne plus se soucier de la gestion physique de leurs salles serveurs, elles n'ont besoin que d'une connexion VPN, ou d'une ligne louée jusqu'au fournisseur de Cloud. Grâce à la philosophie de Cloud, les clients ne payent plus que les ressources consommées.
Ils ont plusieurs types de gestion de leurs ressources virtuelles à disposition :

IaaS[1] (Infrastructure as a Service) : Le fournisseur Cloud s'occupe de la gestion des nœuds de virtualisations de ces clients ainsi que des serveurs physiques. Le client n'a qu'à s'occuper de gérer ces VMs, en implémentant les programmes qu'il veut. Ce projet de Bachelor va se baser sur cette variante.

Saas[2] (Software as a Service) : Le fournisseur propose à ces clients des services tel que de messageries basé sur des systèmes de Webmail.
Durant le travail de Bachelor, nous allons nous placer au niveau du fournisseur de Cloud et allons voir les différentes contraintes liés à une infrastructure virtualisée, tant au niveau de l'isolation des VM avec l'utilisation de VLAN, que de la gestion des unités de stockages.

[1] http://fr.wikipedia.org/wiki/Infrastructure_as_a_service
[2] http://fr.wikipedia.org/wiki/Logiciel_en_tant_que_service

4.2.1 Qui dit Cloud dit virtualisation

Au sein d'une architecture virtualisée, chaque serveur va exécuter plusieurs systèmes d'exploitation. Cette solution va permettre une optimisation des ressources matérielles, car un serveur ne gère plus un système d'exploitation, mais une multitude. Pour créer ce type de serveurs, nous devons rajouter une couche de virtualisation au système d'exploitation primaire (système installé juste au dessus de la couche matériel). Pour ce projet, nous allons rester dans un environnement Open-Source, donc le module qui sera rajouté au noyau Linux sera KVM (Kernel-based Virtual Machine).

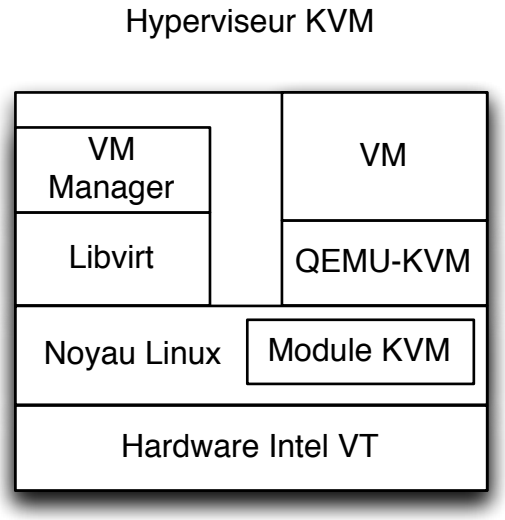


Figure 1 : Virtualisation sous Linux

La **Figure 1** est composée de plusieurs couches :

- la couche **QEMU-KVM,** va servir de solution d'émulation, donc va simuler à la VM un processeur virtualisé
- Le module **libvirt** va permettre d'envoyer des instructions à KVM pour instancier une VM, la stopper ou la redémarrer
- Le PC supportant les couches de virtualisation est un hyperviseur.

4.3 Le Cloud management

Le centre névralgique du Cloud est le Datacenter, il est basé sur une infrastructure virtualisée. C'est-à-dire qu'elle va regrouper des dizaines de serveurs, qui vont virtualiser des centaines de VMs. Il serait impensable d'administrer chaque hyperviseur un par un. C'est pour cela que différentes solutions existent pour administrer les différents hyperviseurs de manière centralisée.

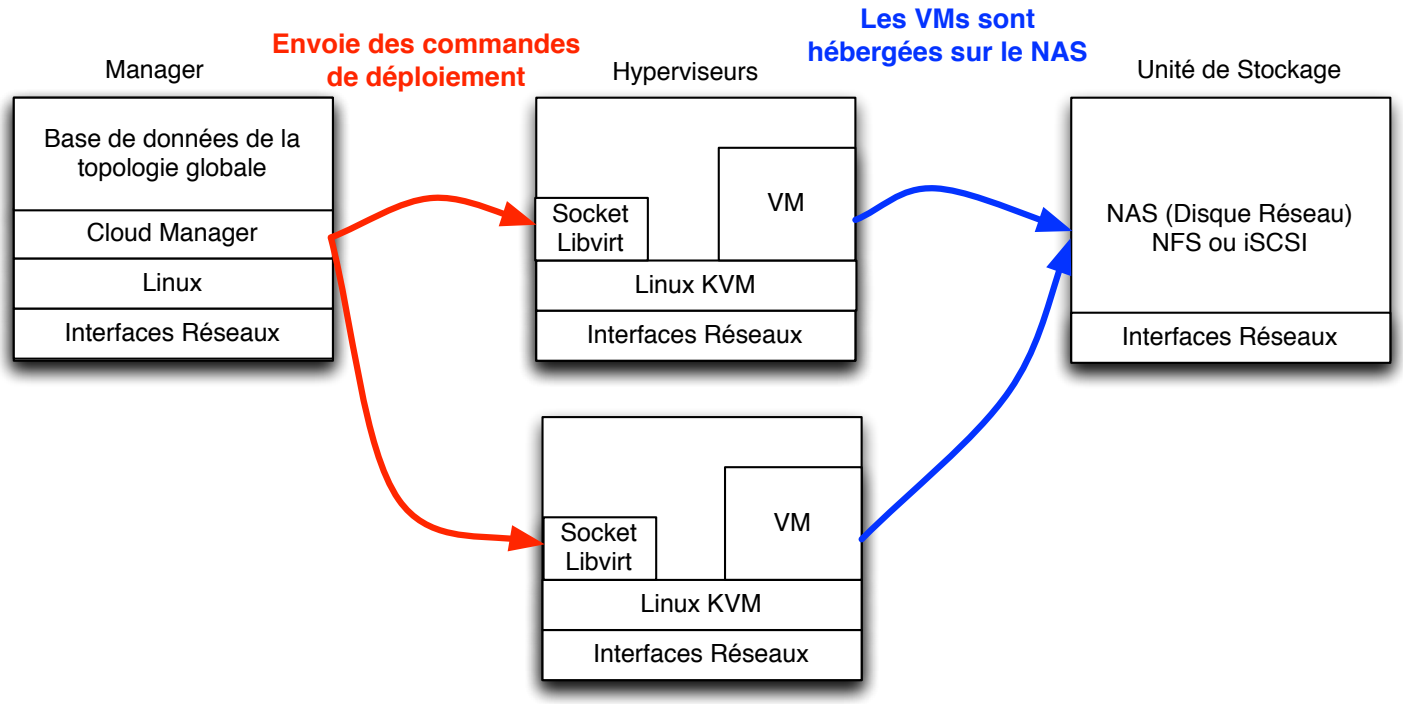


Figure 2: Schéma global d'une infrastructure virtualisée

Cette topologie (Figure 2) est composée de plusieurs éléments :

- Un élément central appelé **Manager**, qui va gérer l'exécution des VMs, ainsi que contrôler les ressources disponibles au sein de notre réseau.
- Des **hyperviseurs** qui vont exécuter les VMs.
- **L'unité de stockage** va contenir les disques utilisés par les VMs. Il intègre plusieurs protocoles, dans mon projet j'ai utilisé l'iSCSI et le NFS (Système de fichiers réseaux). Cette unité sera matérialisée dans un premier temps par un **PC Fedora16**, puis par la suite par un **QNAP ts-459 Pro II**.

4.4 Contexte pratique au sein de SmartBee

4.4.1 Schéma de principe du réseau de la société SmartBee

Ce schéma simule le réseau de la société SmartBee :

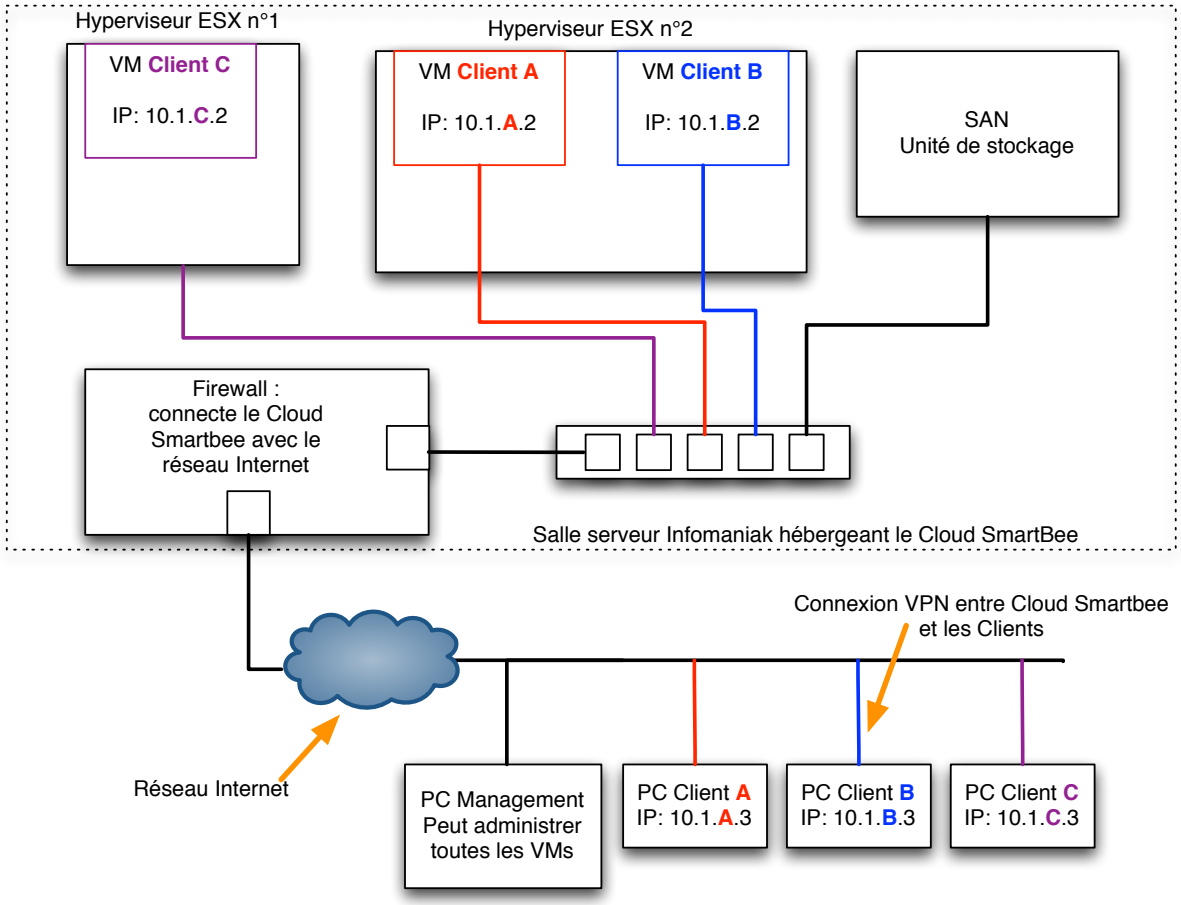


Figure 3: Schéma de principe du réseau SmartBee

- L'entreprise SmartBee héberge ses serveurs auprès de la société Infomaniak.
- Ses différents clients viennent se connecter à des VMs propres à chacun d'eux par l'intermédiaire d'un tunnel VPN.
- Un système de management va monitorer chaque VM, afin de mesurer les charges CPU, la RAM ou encore la gestion de l'anti-virus.

4.5 Contraintes

- Le développement du projet doit se faire essentiellement avec des outils **Open-Source**.

- Une isolation entre les VMs doit être mise en place, du fait que chaque client se connectant par son tunnel VPN doit pouvoir interroger uniquement ces VMs.
- Chaque VM doivent être administrées via un réseau de management commun à toutes.
- Le systèmes de stockage doit être externalisé et centralisé. Tous les disques durs des VMs doivent être stockés sur une unité de stockage à part entière.

4.5.1 Solutions utilisées

Pour mener à bien ce projet, je vais étudier trois technologies, toutes basées sur de l'Open-Source, la première est **oVirt**[3] (Open Virtualisation), qui permet d'administrer plusieurs nœuds de virtualisation. La deuxième s'appelle **OpenNebula**[4], cette solution permet de gérer un Cloud privé basé sur une gestion approfondie des services réseaux tel que les VLANs. Pour créer une isolation entre les VMs, nous devons mettre en place **Open vSwitch**[5]. Comme le réseau doit être isolé au sein du Cloud, un serveur DNS devra être déployé pour que la résolution FQDN puisse se faire entre les différents serveurs.

4.6 Technologies n°1 : oVirt

Dans un premier temps, la solution mise en place pour créer une infrastructure virtualisée est oVirt. Cette solution est basée sur le management d'hyperviseurs basés sur KVM. Toute l'administration des nœuds de virtualisations se fait via une interface web. Cette solution gère plusieurs types de protocoles de stockage comme le partage NFS et iSCSI. Deux éléments importants vont caractériser notre mise en place. Voir le schéma global au **§5.1.1**:

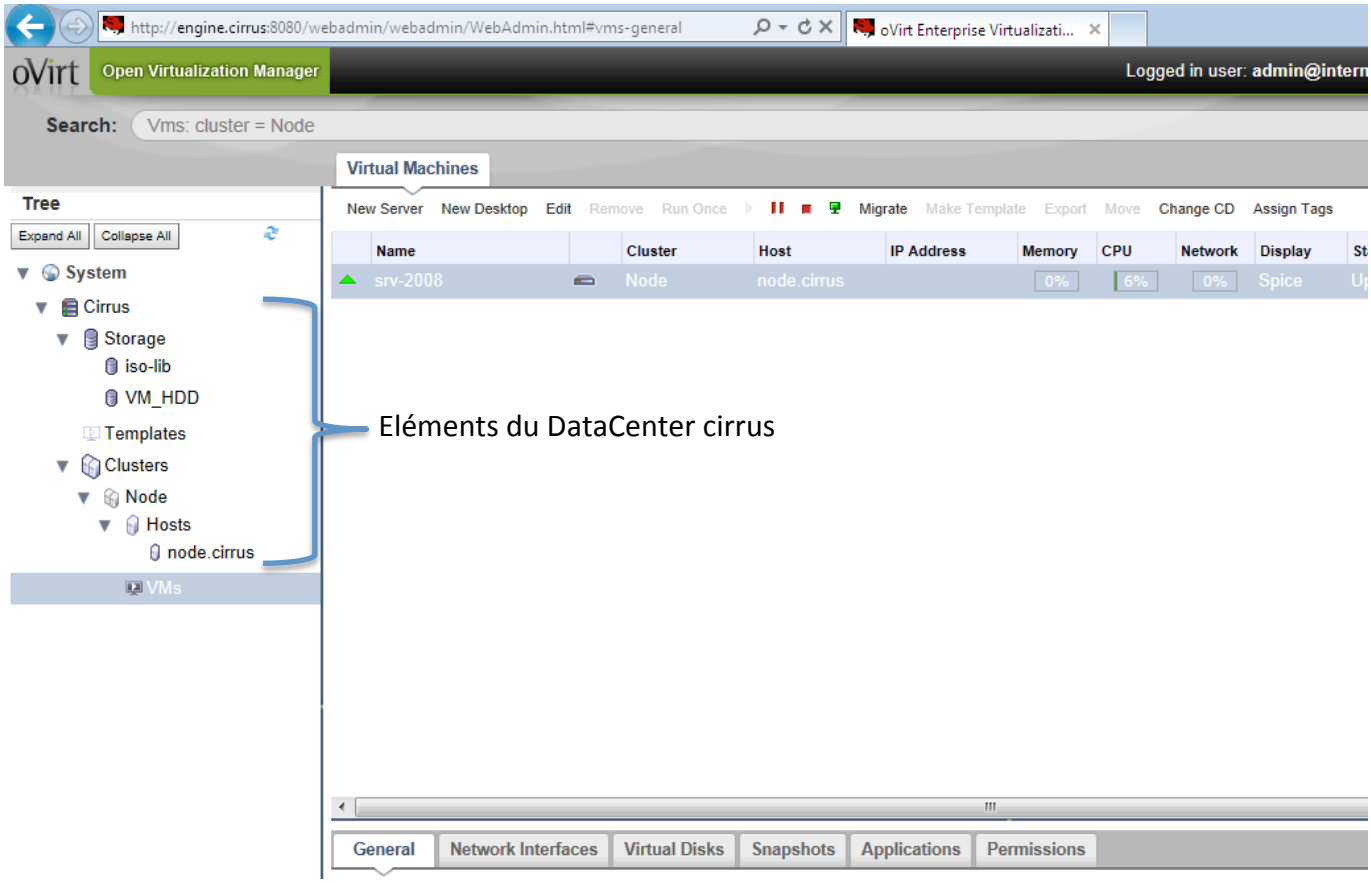
4.6.1 oVirt Engine Manager

Il est vu comme vSphère sous VMware. C'est lui qui va tenir à jour les bases de données qui englobent tous les nœuds de virtualisation, ainsi que la liste des images ISO. Il possède une interface web, atteignable depuis un navigateur internet, afin de gérer les nœuds et les unités de stockages. Il fonctionne sur la distribution Fedora 16.
Il possède comme services :

- Deux bases de données, une pour la bibliothèque ISO et l'autre pour l'architecture globale.
- Client SSH, pour communiquer avec l'hyperviseur afin d'instancier les VMs et monitorer les ressources.
- Possède un dossier partagé en NFS, qui contient toutes les images ISO, servant à installer un VM.
- Serveur web, pour administrer le Datacenter via une interface GUI, basé sur le moteur JAVA JBOSS.
- Utilitaire d'installation pour déployer l'hyperviseur de façon automatique, grâce à un script.
- Serveur Spice, pour pouvoir ouvrir une console d'affichage directement sur une VM.

[3] http://ovirt.org/w/images/4/4a/OVirt-SCALE10x-20120122.pdf
[4] http://www.opennebula.org/documentation:rel3.4:intro
[5] http://openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=README;hb=HEAD

- Gestion du Datacenter : Le Datacenter est l'ensemble des éléments qui vont constituer notre infrastructure virtualisée tels que les hyperviseurs et les unités de stockage.



Print screen 1 : Apperçu de l'interface web

4.6.2 oVirt Node Hyperviseur

Cet élément est l'hyperviseur, donc notre plate-forme de virtualisation. Il est formé par :

- Distribution basée sur Fedora16, appelé **oVirt Node.**
- Serveur SSH pour recevoir les commandes du manager.
- Implémentation des protocoles de stockages iSCSI ou NFS pour stocker les VMs sur un NAS.
- Configurations de base se fait via une interface TUI (Text User Interface), afin d'administrer les cartes réseaux physiques, ainsi que les mots de passe.
- Intègre le module KVM.
- Système Read-Only donc impossible d'intégrer des modules supplémentaires à la distribution.

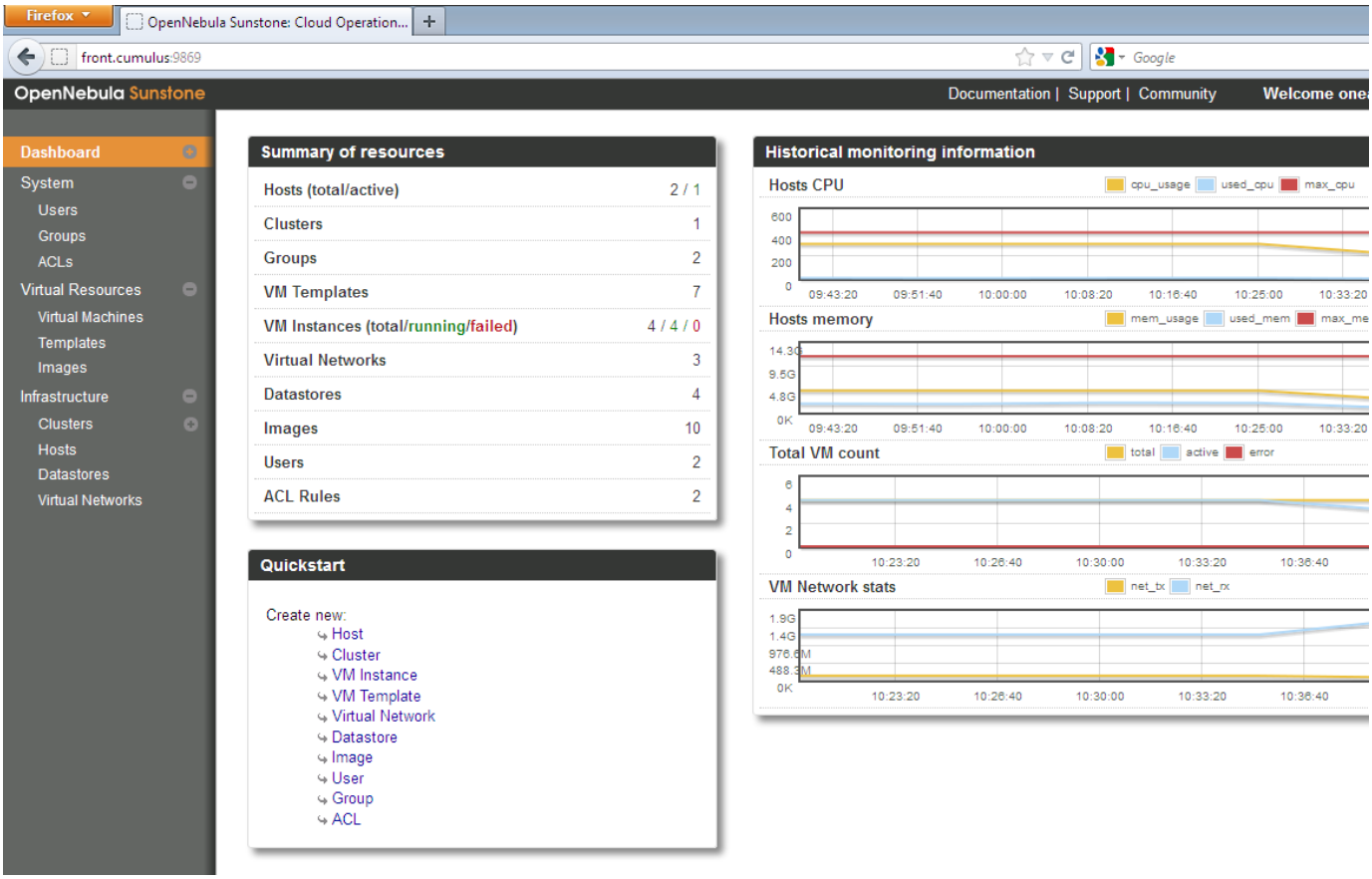
4.7 Technologie n°2 : OpenNebula

La deuxième technologie est OpenNebula. Il s'agit d'un outil de management permettant d'administrer un Datacenter hétérogène, c'est-à-dire qu'il peut être composé à la fois d'hyperviseurs KVM, ESX chez VMware ou même de Hyper-V chez Microsoft. OpenNebula est vu comme le chef d'orchestre de notre Cloud, du fait qu'il va gérer les nœuds de virtualisation, les réseaux inter-VM, ainsi que les unités de stockage disponibles. Il peut être déployé sur toutes sortes de distributions Linux.

4.7.1 Le Manager (Front-end)

Le manager est appelé « Front-end », et l'hyperviseur « Host ». Le manager possède :

- Un client SSH, qui va transférer vers l'hyperviseur les fichiers ISO à monter avec la VM ou le disque cloné.
- Le module Libvirt, utile pour déployer le modèle de la VM via le socket Libvirt de l'hyperviseur. Ce modèle est généré à partir d'un fichier XML, il regroupe diverses informations sur la quantité de mémoire allouée à la VM, des disques montés, ainsi que les interfaces réseaux virtuelles connectées à cette VM, le module est appelé **one**.
- Une base de donnée MySQL, qui regroupe la topologie globale de notre infrastructure.
- Administration **CLI**[6] ou **GUI** via le serveur web **Sunstone.**



Print screen 2 : Apperçu de l'interface web Sunstone

[6] http://opennebula.org/documentation:rel3.4:cli

4.7.2 Les hyperviseurs (Host)

L'hyperviseur peut être un système KVM, il possède :

- Un serveur SSH pour recevoir les fichiers ISO envoyés par le manager.
- Un client NFS ou iSCSI pour communiquer avec le NAS afin de stocker les VMs.
- L'hyperviseur peut être basé sur n'importe quelle distribution Linux intégrant KVM.

4.8 Communication réseau des VMs

KVM propose deux modes de communications entre les VMs est le réseau global :

4.8.1 NAT (Network Address Translation)

La VM est confinée dans un réseau IP au sein de l'hyperviseur. Pour sortir sur le réseau, l'hyperviseur effectue une action de NAT. Il est donc difficile d'interroger une VM server depuis le réseau externe.

4.8.2 Bridge

Les VMs s'intègrent directement dans le réseau externe. La carte réseau physique de l'hyperviseur est transparente. Ce mode est utile si les VMs hébergées par l'hyperviseur sont des serveurs, car on pourra directement les intégrer depuis le réseau externe. Le point de sortie vers l'extérieur se fait grâce à l'interface physique.

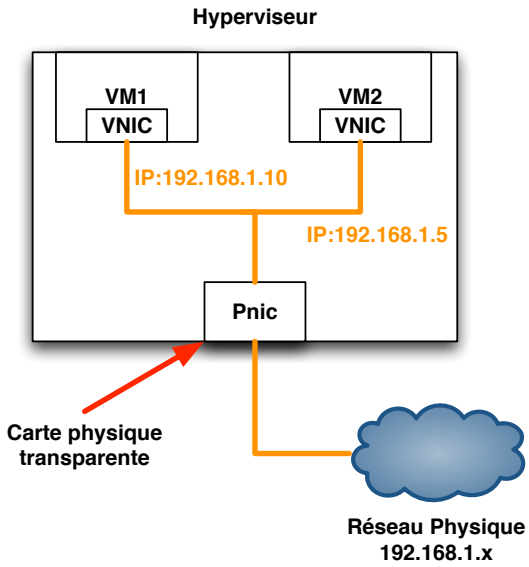


Figure 4: Mode bridge

4.8.3 Bridge KVM

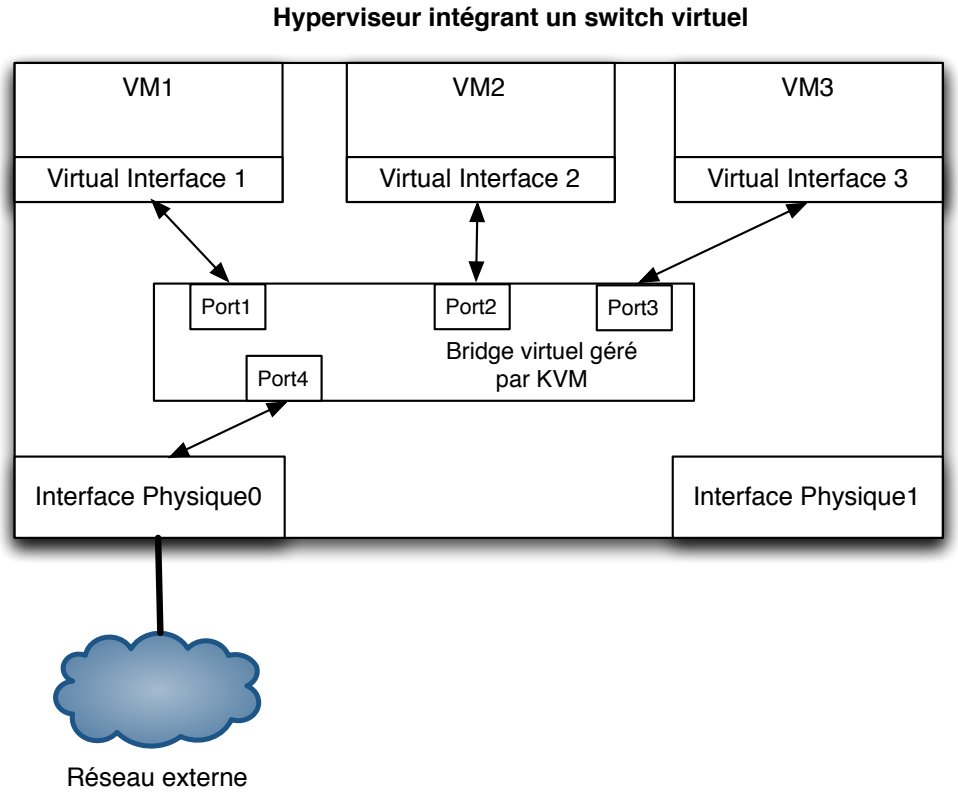


Figure 5: Shéma de principe du bridge sous KVM

Au lancement d'une VM, le module KVM va créer une carte réseau virtuelle (Virtual Interface). Cette interface va être connectée à un switch virtuel. Chez KVM, il se nomme **Bridge**. Il va travailler au niveau Ethernet, et chaque carte virtuelle est vue comme un port du Bridge. Il est directement intégré au noyau Linux pour augmenter la rapidité de commutation des paquets. Au sein d'un Cloud, plusieurs VMs possédées par des clients différents se côtoient. Il est donc essentiel de faire un cloisonnement par VLAN. Le commutateur KVM[7] ne prend pas en compte les VLANs. Pour palier à ce problème, il faut utiliser le module Open vSwitch, qui lui permet la gestion des VLANs.

4.8.4 Les VLANs

Les Virtuals LANs sont des réseaux virtuels au sein d'un switch physique. Ils permettent de séparer les flux, afin de garantir un cloisonnement et gérer plus finement notre réseau. Chaque VLAN est matérialisé par un ID. Il existe trois types de VLANs :

- **VLAN de niveau 1** (Tag des ports du switch): Il faut inclure les différents ports du switch, qui appartiendront à tel ou tel VLAN.
- **VLAN de niveau 2** (Au niveau Mac Adresse) : Il faut indiquer quelles MAC adresses vont être incluses dans tel ou tel VLAN, peu importe le numéro de port du switch auquel sont connectés les PCs.
- **VLAN de niveau 3** (Au niveau des adresses IPs) : Fonctionne comme le niveau 2, mais l'intégration dans les VLANs se fait via l'adresse IP.

[7] http://www.linux-kvm.org/page/Networking

Pour interconnecter plusieurs switchs entre eux, on configure sur chacun d'entres eux un port **Trunk** qui va permettre de propager les VLANs.

Pour connaître l'appartenance de la trame Ethernet à tel VLAN, le switch va modifier l'entête de la trame en ajoutant un champs appelé **Tag** correspondant au numéro du VLAN, basé sur la norme **802.1Q**.

adresse MAC dst.	adresse MAC src.	**Tag (inséré)**	Len/Etype	Data	**FCS (modifié)**

Figure 6 : Schéma d'une trame Ethernet

Au niveau des PCs connectés au switch, ils ne vont pas voir l'entête Ethernet modifiée du fait que lorsque la trame sort du switch pour aller à un PC au sein du même VLAN, le switch va supprimer le champ **Tag** et recalculer le checksum (**FCS**).

4.9 Technologie n°3 : Open vSwitch

4.9.1.1 Introduction

Open vSwitch est un module qui va se substituer au module **Bridge** de KVM, et va permettre de pouvoir gérer les VLANs au sein de notre hyperviseur. L'administration du bridge se fait par lignes de commandes à l'intérieur de l'hyperviseur. Il va s'installer au niveau du noyau. Trois éléments importants constituent Open vSwitch au sein de l'hyperviseur :

- `ovs-vswitchd` : Le service qui va gérer le module Open vSwitch intégré au noyau, afin d'aiguiller les paquets entre les différentes interfaces (physiques ou virtuelles) au sein de l'hyperviseur.
- `ovsdb-server` : C'est une base de donnée, qui va stocker tous les éléments réseaux raccordés aux différents bridges. `ovs-vswitchd` va effectuer des requêtes à cette base, afin aiguiller chaque paquets.
- `ovs-vsctl` : Cette commande va permettre d'administrer le bridge, elle va permettre de venir greffer une nouvelle interface, ou d'attribuer les VLANs ID. Elle va interagir directement avec la base de donnée.

4.9.1.2 Les commandes de bases

- Ajout d'un nouveau bridge nommé « **br1** » :

```
sudo ovs-vsctl add-br br1
```

- Ajout d'une interface virtuelle avec le numéro de **VLAN 2** au bridge « **br1** »

```
sudo ovs-vsctl add-port br1 vnet3 tag=2
```

- Ajout d'un port Trunk propageant les VLANs 2 et 5

```
sudo ovs-vsctl add-port br1 eth3 trunks=2,5
```

- Pour visualiser toute la configuration des bridges

```
sudo ovs-vsctl show
```

4.10 Solutions de stockages

A travers ce projet, deux unités de stockages ont été mises en place.

4.10.1 Cible iSCSI sous Fedora

Une cible iSCSI basée sur Fedora16, installée sur un PC dédié à cette tâche. Le serveur iSCSI écoute sur le port 3260. Le déploiement est basé sur un fichier de configuration qui contient le nom de l'espace de stockage (LUN) et les autorisations d'accès.

4.10.2 NAS (Network Area Storage) QNAP TS-459 Pro II

Pour calquer au mieux la réalité, nous avons commandé un NAS. Cette unité de stockage va pouvoir gérer la cible iSCSI ainsi que les partages réseaux NFS, utilisés pour héberger les disques virtuels des VMs.

Ce NAS n'a rien à envier aux vraies solutions professionnelles, du fait qu'il peut intégrer quatre disques durs SATA 600 pouvant atteindre des vitesses d'écritures de 6 Gbit/s. Pour notre installation, nous avons choisi quatre disques de 600 Gbits Western Digital VelociRaptor.

Photos 1: Face avant du QNAP

Caractéristiques techniques[8] :

Capacité de stockage	4 disques Western Digital VelociRaptor 600GB Sata6 (6Gb/sec)
Processeur	Intel Atom D525 1,8 GHz Dual Core
Mémoire RAM	1GB de base, peut aller jusqu'à 3GB
Services de stockage	iSCSI, NFS
Système d'exploitation	Environnement QNAP basé sur un noyau Linux 2.6
Administration	Serveur web intégré, afin de configurer les services réseaux, monitorer les ressources utilisées. Accès aux fichiers stockés sur les disques depuis le web
Ecran frontal	Est utile pour afficher les adresses IP utilisées

Photos 2: Face arrière du QNAP

Tout au long du projet, j'ai utilisé dans le NAS QNAP un unique disque dur de 600GB

[8] http://www.clubic.com/disque-dur-memoire/nas/article-465800-1-qnap-ts-459-pro-ii.html

4.11 Live Migration KVM

Ce procédé permet de transférer une VM active d'un hyperviseur à l'autre. Cette action permet :

- D'équilibrer les charges entre tous les hyperviseurs. Par exemple, si un hyperviseur possède une VM qui commence à être plus sollicité, on va alors décider de la migrer vers un autre hyperviseur sous exploité.
- Une non-interruption de service au moment où la VM est migrée. L'utilisateur ne va pas subir de déconnexion au moment du transfert vers un autre hyperviseur.

Les Prérequis :

- Tous les hyperviseurs doivent partagés le même espace de stockage.
- Ils doivent avoir les mêmes configurations de bridges virtuels entres les deux hyperviseurs.
- La connexion SSH sans mot de passe doit se faire entre les deux hyperviseurs.

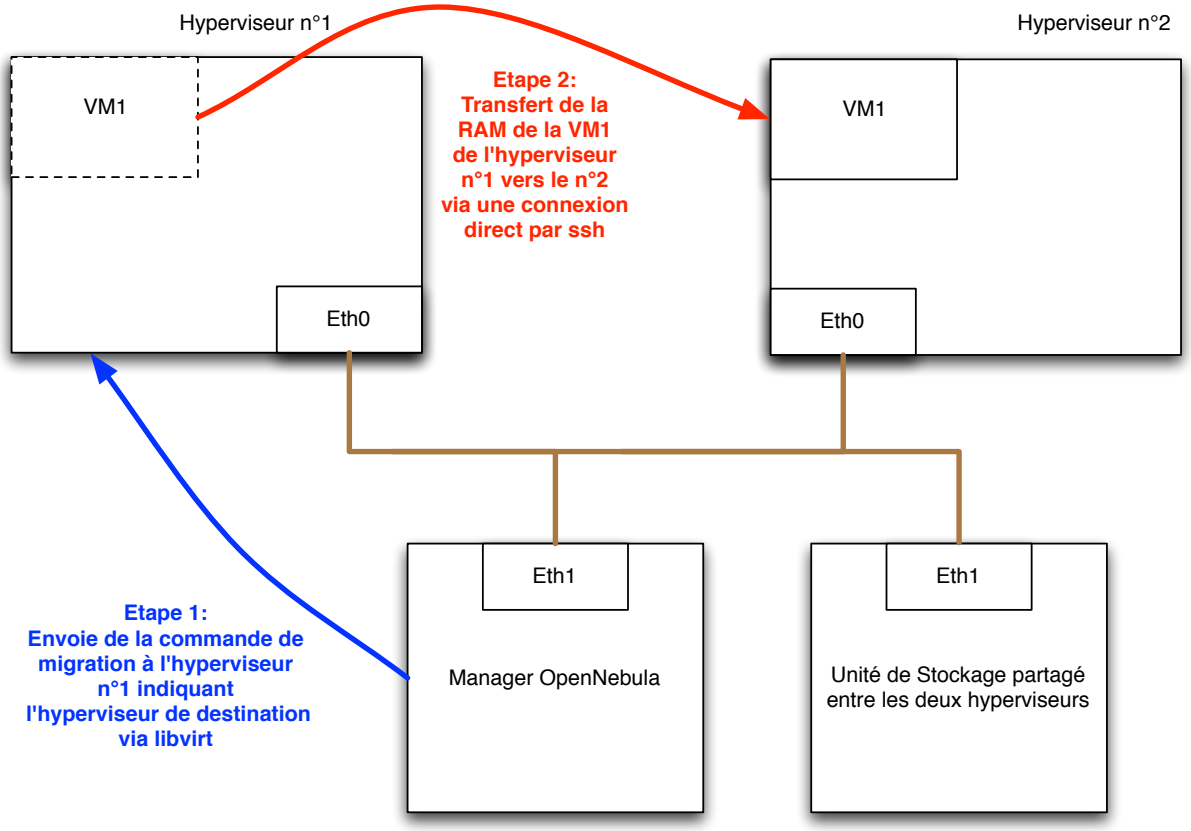


Figure 7 : Schéma de principe de la Live Migration

Le fonctionnement comme expliqué sur la **Figure 7**:

- Etape 1 : Une connexion SSH est établie entre le manager et l'hyperviseur.
- Etape 2 : La Live Migration effectue un transfert de la mémoire RAM via une connexion SSH entre les deux hyperviseurs.
- Etape 3 : Une fois le transfert de la RAM effectué, la VM est stoppée sur l'hyperviseur n°1 et activée sur l'hyperviseur n°2.

4.12 Scénarios proposés

Je vais vous proposer quatre scénarios qui vont mettre en place les différentes technologies énoncées plus haut :

4.12.1 Déploiement de oVirt avec Windows Server 2008 virtualisé

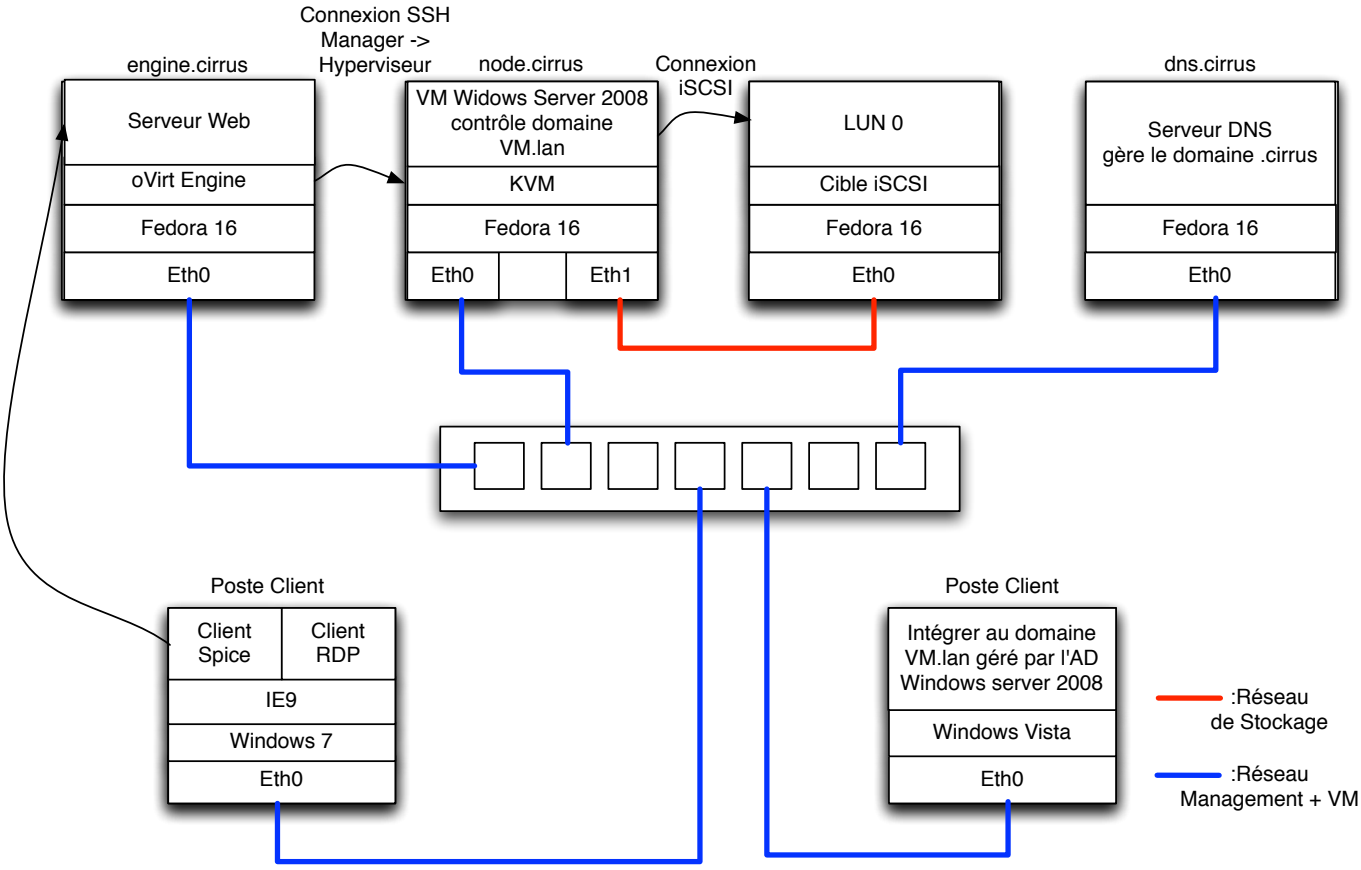


Figure 8: Schéma bloc du scénario 1

Ce scénario (Figure 8) va permettre de tester le déploiement d'une VM depuis un manager vers un hyperviseur distant, basée sur la technologie oVirt. Le but est de configurer un manager (oVirt Engine) et un hyperviseur KVM (oVirt-Node), la VM exécutées sera stockée sur une unité iSCSI au sein d'un réseau physique à part entière (Réseau Rouge). Cette VM va gérer un domaine Active directory, afin de créer un véritable environnement professionnel.
Notre scénario va être constitué :

- D'un **manager** oVirt Engine, déployé à l'aide d'un utilitaire fourni par oVirt.
- D'un **hyperviseur** KVM, fonctionnant sous Fedora16 déployé à l'aide d'oVirt-Node.
- D'une **unité de stockage iSCSI**, qui sera un PC sous Fedora16.
- D'un **poste client** fonctionnant sous **Windows 7** avec un navigateur IE9, pour se connecter au serveur web du manager, afin de lancer des VMs. Il intègrera le client SPICE pour afficher le terminal des VMs. Par la suite on utilisera un client RDP pour se connecter au Server 2008.
- D'une **VM Windows Server 2008**, permettant de créer un environnent professionnel grâce au domaine Active Directory.
- D'un **PC Windows Vista**, **intégré au domaine VM.lan** qui sera géré par la VM Windows Server 2008.

4.12.2 Mise en place d'une VM sous KVM grâce à OpenNebula

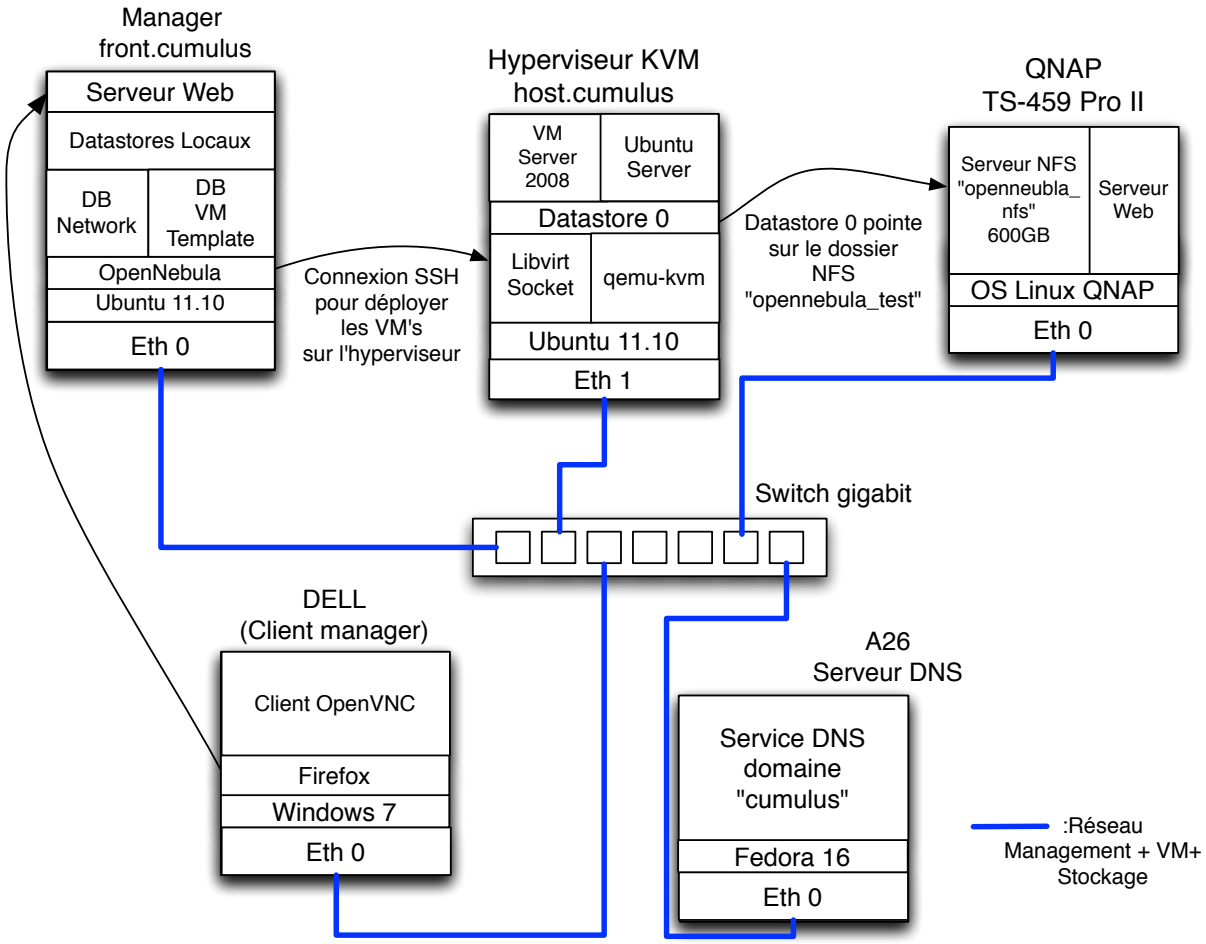


Figure 9: Schéma de principe du Scénario 2

A travers ce deuxième scénario, nous allons mettre en place la solution OpenNebula. Il a pour but de tester, tout comme le scénario précédent, le déploiement de VMs grâce à la technologie OpenNebula. Nous allons nous intéresser tout particulièrement à l'installation des différents éléments tels que le manager et l'hyperviseur. Notre scénario va être constitué :

- D'un manager qui va instancier les VMs sur l'hyperviseur, qui supportera le module OpenNebula.
- D'un hyperviseur possédant le module KVM afin d'exécuter les VMs, ainsi qu'un serveur SSH.
- D'une unité de stockage externe QNAP pour le stockage physique des VMs.
- D'un serveur DNS gérant la zone « cumulus ».

4.12.3 Isolation des VMs grâce aux VLANs layer 1 gérés par Open vSwitch

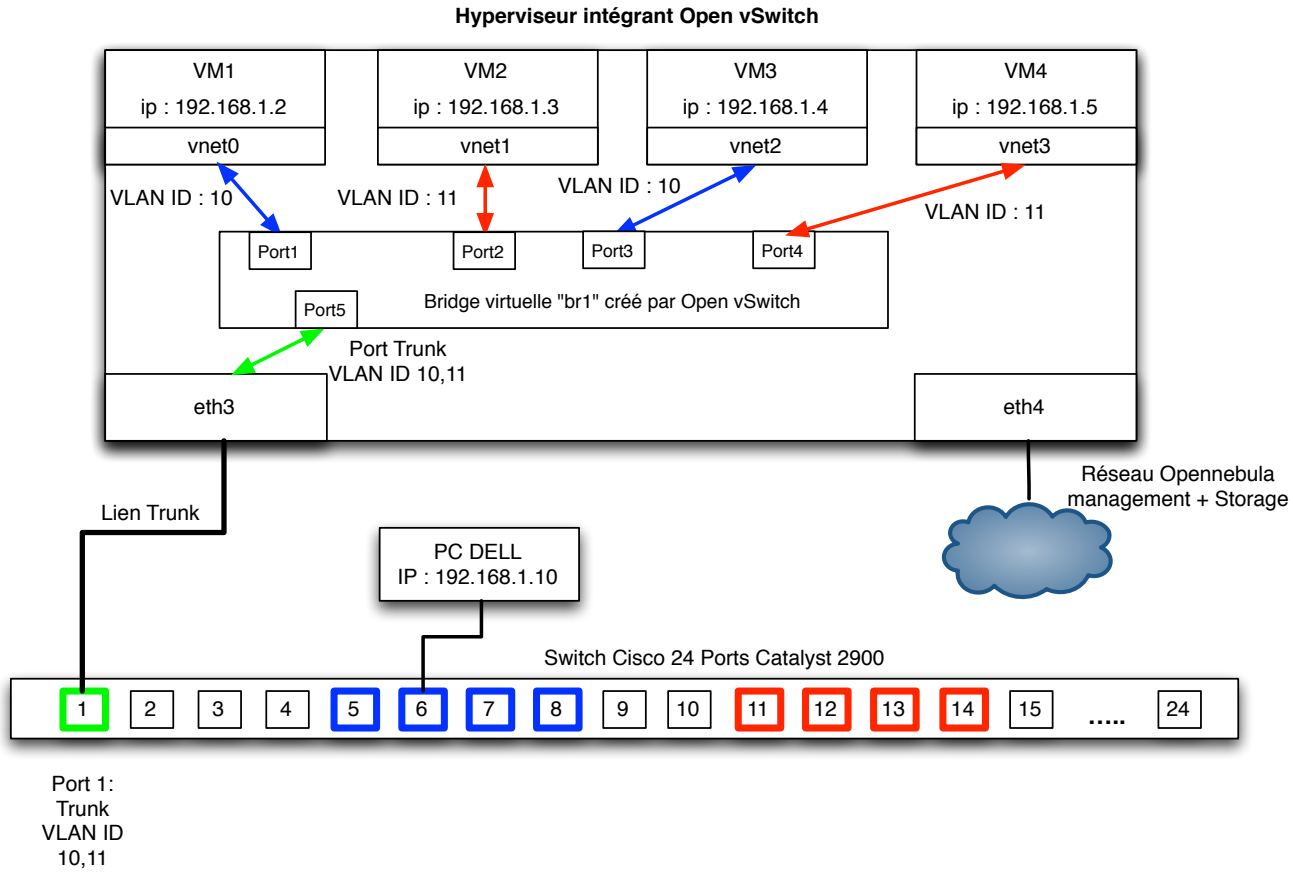


Figure 10 : Schéma de principe du Scénario 3

A travers ce troisième scénario, le but est de cloisonner les VMs en les intégrant dans des VLANs (Voir Figure 10).

- L'objectif est de mettre en œuvre le cloisonnement des VMs grâce à des VLANs gérés par Open vSwitch, et de voir la compatibilité avec les VLANs standards gérés par des switchs physiques.
- Quatre VMs vont être déployées sur notre hyperviseur KVM, qui intégrera le bridge virtuel géré par Open vSwitch.
- Deux VMs seront dans le VLAN 10 et les deux autres dans le VLAN 11.
- Une interface réseau sera dédiée au trafic des VMs, elle sera paramétrée en mode **Trunk**, afin de propager vers un switch Cisco Catalyst 2900.
- Ce switch aura quatre ports connectés au VLAN 10 et quatre au VLAN 11.
- Des tests de Ping seront effectués pour tester le cloisonnement. Un PC DELL sera connecté au switch Cisco pourra pinger tels ou tels VMs en fonction du port où il sera connecté.

4.12.4 Scénario final n°4: Isolation des VMs + Live Migration + NAS + Supervision

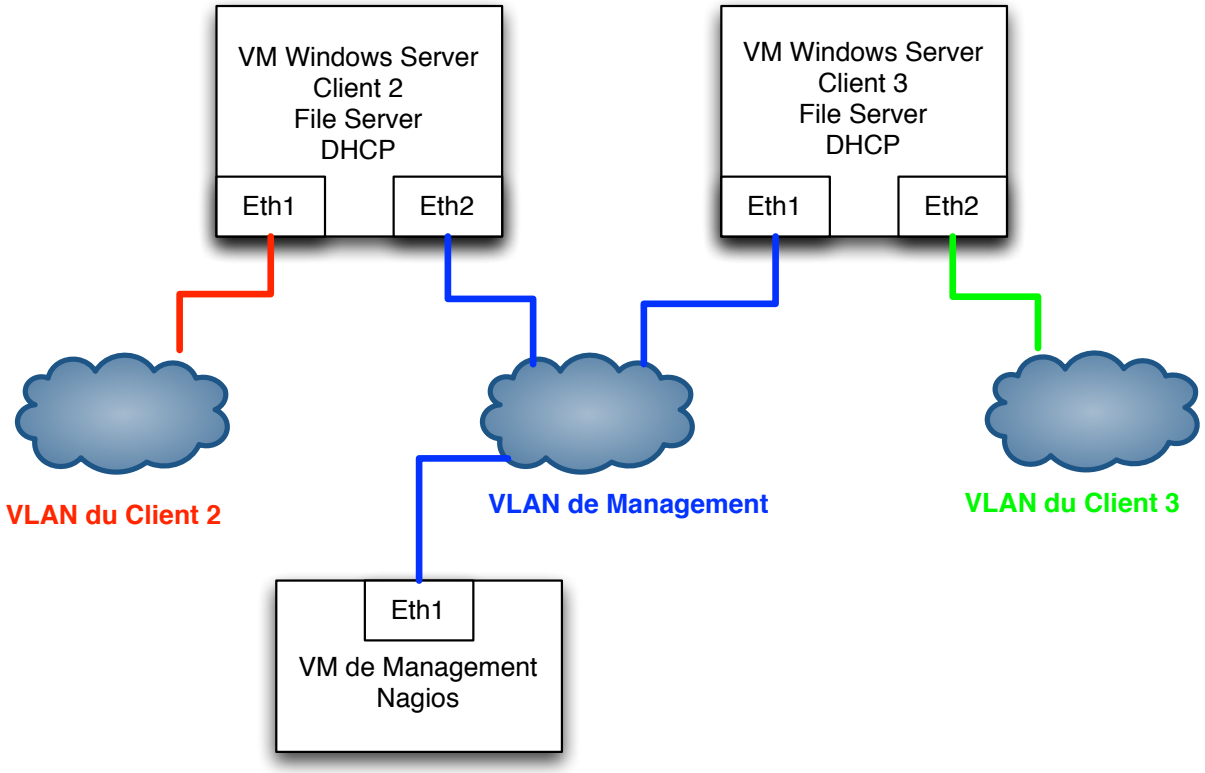


Figure 11 : Schéma de principe du Scénario final 4

Les objectifs de ce scénario sont:

- De permettre à deux clients de se connecter à leurs File Servers (Samba) respectifs gérés par deux VMs, tout en étant isolés de l'autre.
- De monitorer l'état des VMs.
- De rendre possible la Live Migration grâce aux deux hyperviseurs en fonction.

Le scénario sera basé sur:

- La création de deux VMs Windows Server 2008, qui intègreront chacune un serveur DHCP et un File Server (Samba) pour chacun des clients qui viendront se connecter. J'ai choisi Windows Server 2008 pour recréer un environnement professionnel. Chaque VM va posséder deux cartes réseaux virtuelles, une destinée au réseau du client (Vert ou Rouge), et l'autre réservée au réseau de Management (Bleu). Cette deuxième interface connectée au réseau de management sera essentielle pour que Nagios puisse contrôler l'état des Serveurs Windows 2008.
- La création d'une VM basée sur Ubuntu Desktop 10.04, qui va servir à exécuter le service de supervision NAGIOS afin de connaître l'état des VMs opérationnelles. Elle possèdera une interface réseau reliée au réseau de management.

Cloisonner les VMs aux seins de différents VLANs (Voir le Scénario n°3 pour la configuration de Open vSwitch), trois VLANs seront créés :

- VLAN 1 : VLAN de Management : Il regroupe les éléments de OpenNebula (host.cumulus, front.cumulus, PC Management, dns.cumulus) et les interfaces Management de chaque VM.

- VLAN 2 : VLAN du Client 2. Il regroupe une interface virtuelle de la VM du client 2 et le PC de ce client. Grâce à ce procédé, le client pourra uniquement interroger sa VM et sera confiné dans son réseau.
- VLAN 3 : VLAN du Client 3 : Même but que le VLAN du Client 2, mais appliqué à la VM du Client3

Test du scénario :

- Pour tester mon scénario, j'ai configuré un serveur DHCP au sein de chaque VM qui est en relation avec un client (VM Client2, VM Client3). Chaque PC Client (Vert et Rouge voir schéma ci-dessus) connecté à son VLAN se verra attribuer une adresse IP via le serveur DHCP de sa VM.
- J'ai également configuré un serveur Samba au sein des VMs Windows 2008, car les clients vont se connecter à un service en particulier auprès de leur VM respective.
- A travers mon réseau de Management, j'utilise le logiciel de supervision Nagios installé sur ma VM Ubuntu, il va tester grâce au service de Ping si les éléments névralgiques du réseau sont toujours en vie (VMs Windows Server 2008, dns.cumulus, les deux hyperviseurs).

5 Réalisation

5.1 Scénario n°1 : Déploiement de oVirt avec Windows Server 2008 virtualisé

5.1.1 Schéma global de l'infrastructure oVirt

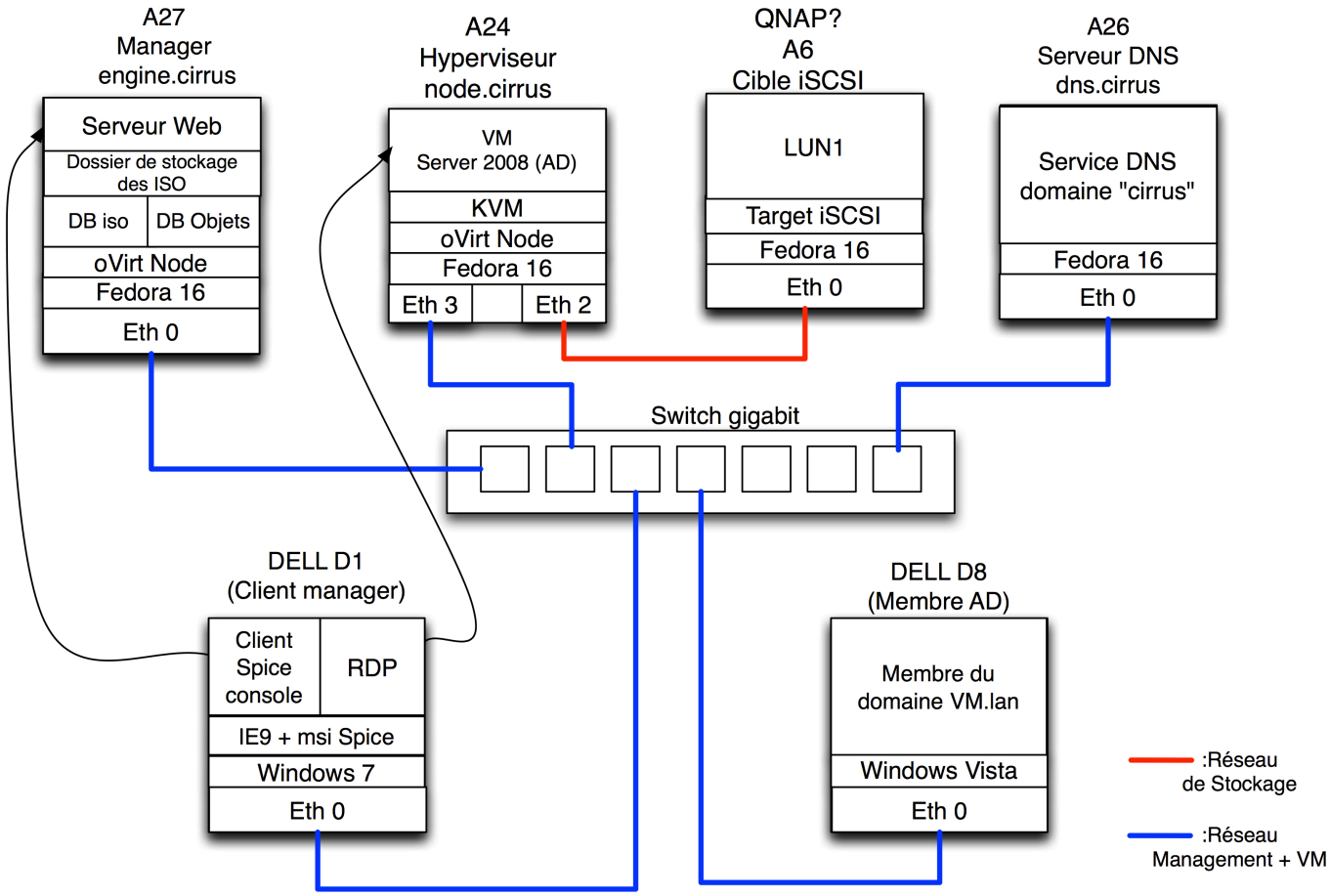


Figure 12: Shéma bloc du scenario n°1 sous oVirt

5.1.2 Introduction

A travers ce chapitre, je vais vous montrer les différentes étapes qui ont permis de réaliser le scénario n°1 basé sous oVirt. La **Figure 12** ci-dessus montre l'environnement réel dans lequel c'est déroulé le scénario. Vous trouverez en **Annexe A.3** la liste des différents PCs utilisés.

5.1.3 Installation de la cible iSCSI

Pour installer ma cible j'ai suivi le laboratoire **Labo_iSCSI_IN3.pdf** :

- Installation du service iSCSI intégrant la gestion de la cible.
- Création d'un Logical Volume de 200 Gbits, dans mon cas cette allocation de stockage correspondra à mon LUN 1, je l'ai appelé LV_iscsi.

J'ai édité le fichier de configuration iSCSI se trouvant dans ***/etc/tgt/targets.conf*** en rajoutant ces quelques lignes :

```
<target iqn.2012-04.cirrus:iscsi>   ←Nom de la cible
backing-store /dev/vg/LV_iscsi      ←Chemin du Logical Group
initiator-address any               ←Aucune restriction de
                                    connexion
LUN 1                               ←Numéro du LUN
</target>
```

Démarrage de la cible, la commande va directement utiliser le fichier de configuration édité ci-dessus :

```
tgt-admin   -e -f
```

Le serveur iSCSI va écouter sur le port 3260. D'après notre topologie, le LUN 1 va servir de disque réseau à l'hyperviseur, c'est-à-dire que le disque des VMs sera hébergé sur le disque de la cible iSCSI. Pour respecter les bonnes pratiques, j'ai décidé de le séparer physiquement du réseau de management. Sur notre schéma (Figure 12), j'ai matérialisé ce réseau sur le schéma global par un trait rouge.

5.1.4 Déploiements du Manager (oVirt Engine)

Pour déployer le manager j'ai suivi le document[9] proposé par oVirt, à partir de la **page 18**. Le manager va fonctionner sur un PC fonctionnant sous Fedora 16. L'installation est basée sur un script fournit par oVirt, qui va automatiser le déploiement des différents services.

- Déploiement du script d'installation

```
sudo engine-setup
```

Le script va recueillir différentes informations, pour configurer les éléments du manager :

- Les ports d'écoutes du serveur web en mode non sécurisé et sécurisé : 8080 et 8443.
- Le nom de domaine du PC hébergeant le manager : engine.cirrus.
- Le mot de passe d'administration global de l'utilisateur « admin » : root.
- L'utilitaire va déployer une base de donnée, le mot de passe est : root.
- Le type de stockage que nous allons utiliser pour héberger les disques des VM: ISCSI.
- Il va également déployer un partage réseau (NFS) pour stocker la bibliothèque d'ISO. permettant d'installer l'OS des VMs. Il sera créé au chemin ***/data/iso***, et aura comme nom de partage **iso-lib**.
- Le script va ouvrir certains ports du firewall.
- L'utilitaire résume les paramètres de l'installation. Passé cette étape, oVirt va déployer les deux bases de données. Le programme va lancer les services liés au serveur web. Le serveur est basé sur le serveur d'application JBoss.

[9] http://www.ovirt.org/w/images/a/a9/OVirt-3.0-Installation_Guide-en-US.pdf

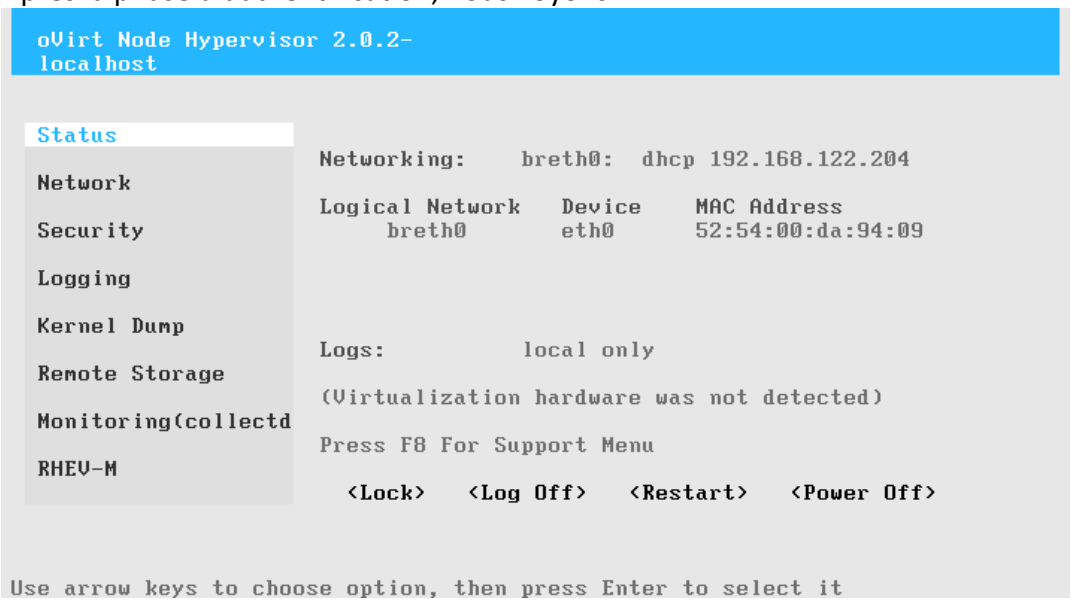
5.1.5 Déploiement de l'hyperviseur (oVirt Node)

L'hyperviseur est installé grâce à une distribution dédiée, appelée **oVirt Node** basée sur Fedora 16. La version utilisée est la **2.3.0-1.0.fc16.**
Il va intégrer tous les modules nécessaires à la virtualisation, ainsi que les différents services permettant de communiquer avec le manager via SSH. La configuration est rendue très aisée grâce à une interface TUI (Text User Interface).
La distribution peut être téléchargée de depuis :
http://ovirt.org/releases/stable/binary/ovirt-node-iso-2.3.0-1.0.fc16.iso

L'installation du système d'exploitation se fait en utilisant les paramètres proposés par le document d'installation[10] en **page 38**.

La configuration de l'hyperviseur s'est effectuée en suivant la **page 40** du document oVirt[11].
Pour la partie suivante, les captures d'écran sont tirées de la présentation sur oVirt-node (http://www.ovirt.org/wp-content/uploads/2011/11/ovirt-node.pdf), sachant que je n'ai pas pu capturer les fenêtres TUI.

Après la phase d'authentification, nous voyons :



Print screen 3: Vu d'ensemble du menu d'oVirt Node

C'est dans ce menu (Print-screen 3) que nous allons pouvoir configurer les adresses IPs comme indiqué dans la Figure 12.
Eth3 aura comme IP : 10.1.2.91 et Eth2 192.168.1.4, car comme sous VMware, pour connecter un hyperviseur à deux réseaux physiques, nous devons constituer deux plages d'adresses IP différentes.

Dans le menu « Security » nous devons activer le server SSH pour que le manager puisse communiquer avec l'hyperviseur.

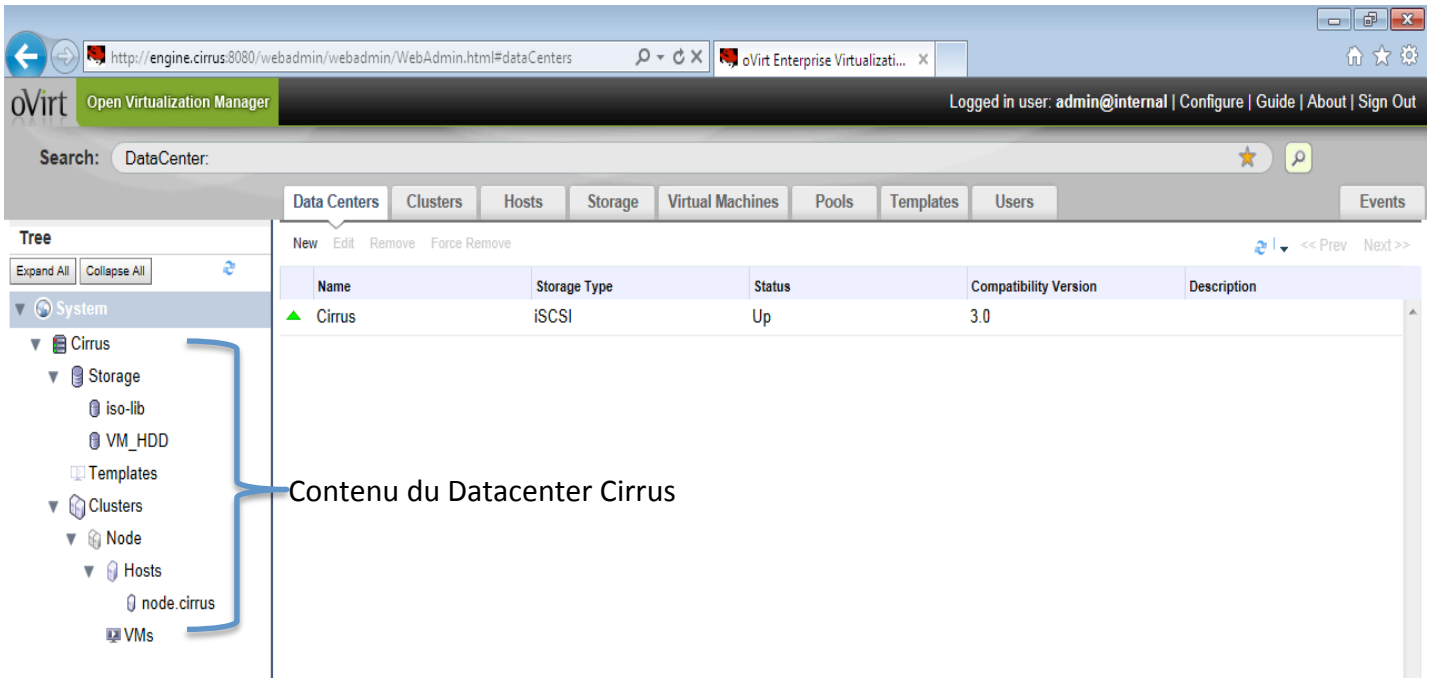
[10] http://www.ovirt.org/w/images/a/a9/OVirt-3.0-Installation_Guide-en-US.pdf
[11] http://www.ovirt.org/w/images/a/a9/OVirt-3.0-Installation_Guide-en-US.pdf

5.1.6 Mise en place du Datacenter

Nous allons nous connecter au serveur web :
Sur le PC client Windows 7 depuis IE9 :

- Installation du plugin client SPICE (RHEV Spice Client.msi) afin de pouvoir visualiser la console d'affichage des VM exécutées.
- Rentrer l'URL de l'interface d'administration : http://engine.cumulus:8080.
- Une fois authentifier sur le portail Administrateur, redirection vers l'interface de management.

Par défaut, oVirt crée un Datacenter appelé « Default », nous devons le supprimer. Pour en déployer un nouveau, j'ai suivi le document oVirt[12] en **page 61**. Dans mon cas le nouveau Datacenter s'appelle Cirrus (Voir Print-screen 4 ci-dessous).



Print screen 4: Contenu global du datacenter Cirrus

D'après le Print-screen 4, il est constitué :

- De deux unités de stockage, notre disque iSCSI (VM_HDD) et notre bibliothèque d'iso (iso-lib).
- D'un cluster, qui va centraliser tous les nœuds de virtualisation disponible. Dans ce scénario nous avons un seul nœuds présent : node.cirrus.

5.1.7 Déploiement de la VM Windows Server

A travers ce scénario, je vais déployer une VM Windows Server 2008 sur mon hyperviseur «node.cirrus». Windows Server va héberger un domaine Active Directory. Un PC client fonctionnant sous Windows Vista va se connecter au domaine Active Directory.

[12] http://www.ovirt.org/w/images/a/a9/OVirt-3.0-Installation_Guide-en-US.pdf

5.1.7.1 Enrichissement la bibliothèque d'ISO

Le dossier ***/data/iso*** va contenir toutes les images ISO, et sera partagé en NFS pour pouvoir transférer ces images disques entre le manager et l'hyperviseur au moment de l'instanciation d'une VM. Depuis le manager j'ai exécuté la commandes iso-uploader présente dans le package d'installation de oVirt-Engine afin de transférer les ISOs dans le dossier ***/data/iso*** :

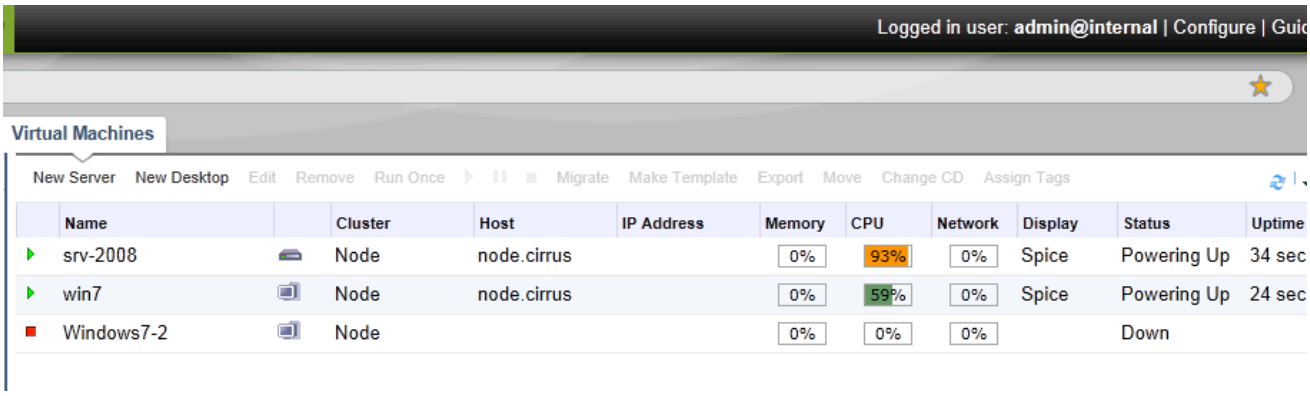
```
ovirt-iso-uploader --iso-domain=iso-lib upload
/chemin de l'iso/Windows_Server2008_32Bits
```

Une fois l'ISO correctement copié, il faut remettre à jour les droits du dossier de la bibliothèque (***/data/iso***)

```
sudo chmod 0755 /data/iso
```

5.1.7.2 Création de la VM Windows Server

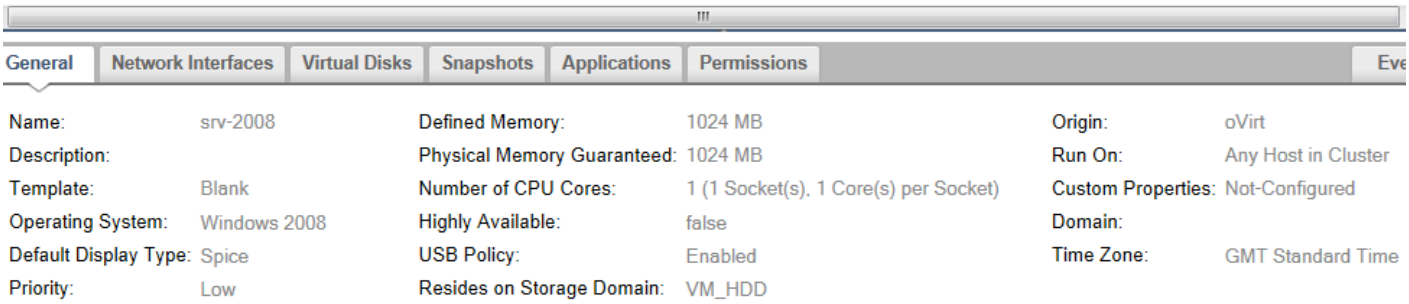
Depuis le serveur web nous pouvons instancier la VM Windows Server 2008 (srv-2008)



Print screen 5: Aperçu des VMS exécutées

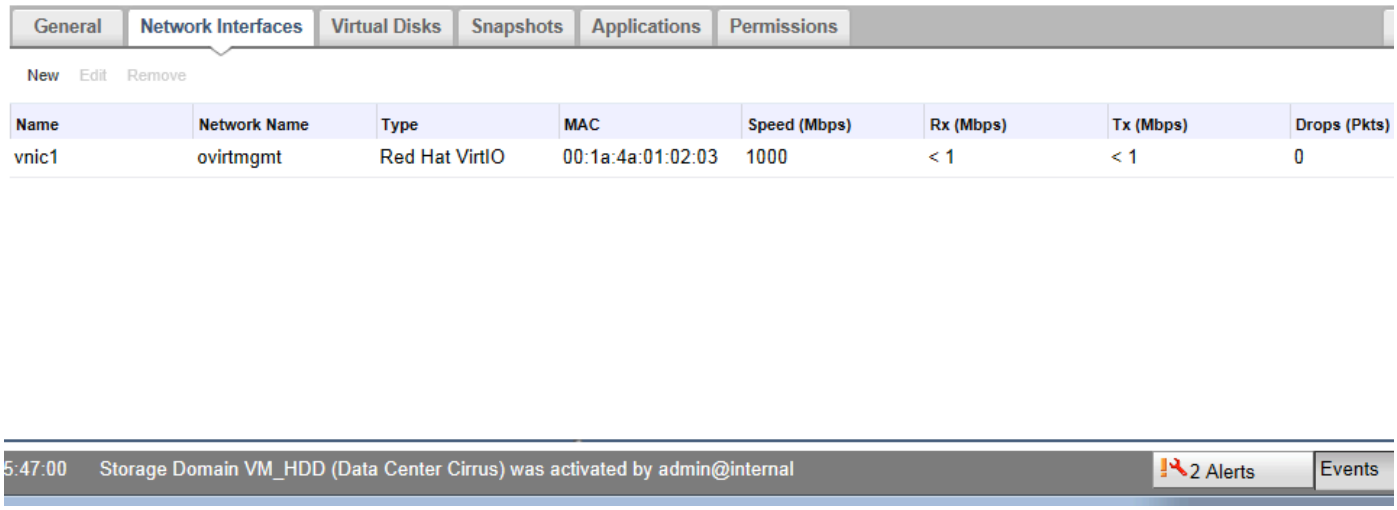
Dans le menu VM affiché ci-dessus (Print-screen 5), nous pouvons voir les VM mises en route sur notre hyperviseur, ainsi que les charges des processeurs propres à chaque VM.

- Voici les caractéristiques générales de la VM Windows Server :



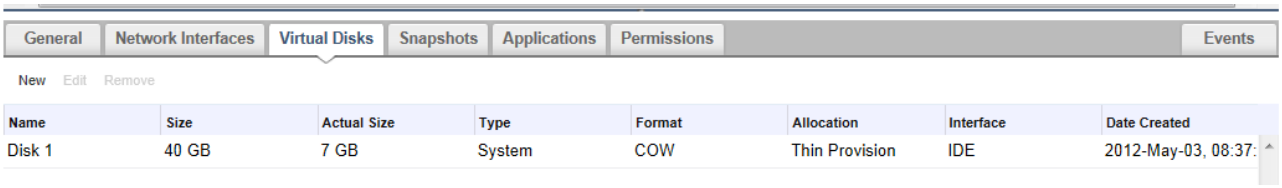
Print screen 6: Caractéristiques générales

- La VM possède une interface réseau :



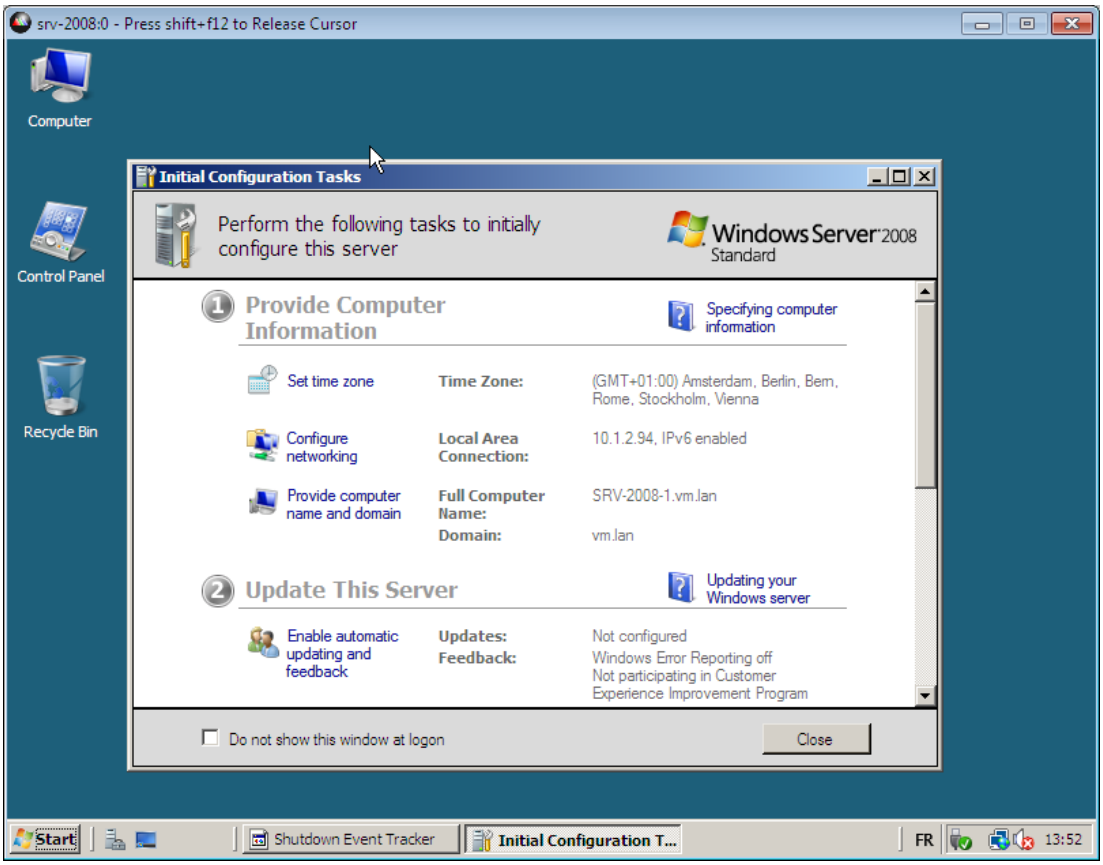
Print screen 7: Interface réseau de la VM

- La VM Windows Server possède un disque virtuel de 40 GB :



Print screen 8: Disque virtuel de la VM

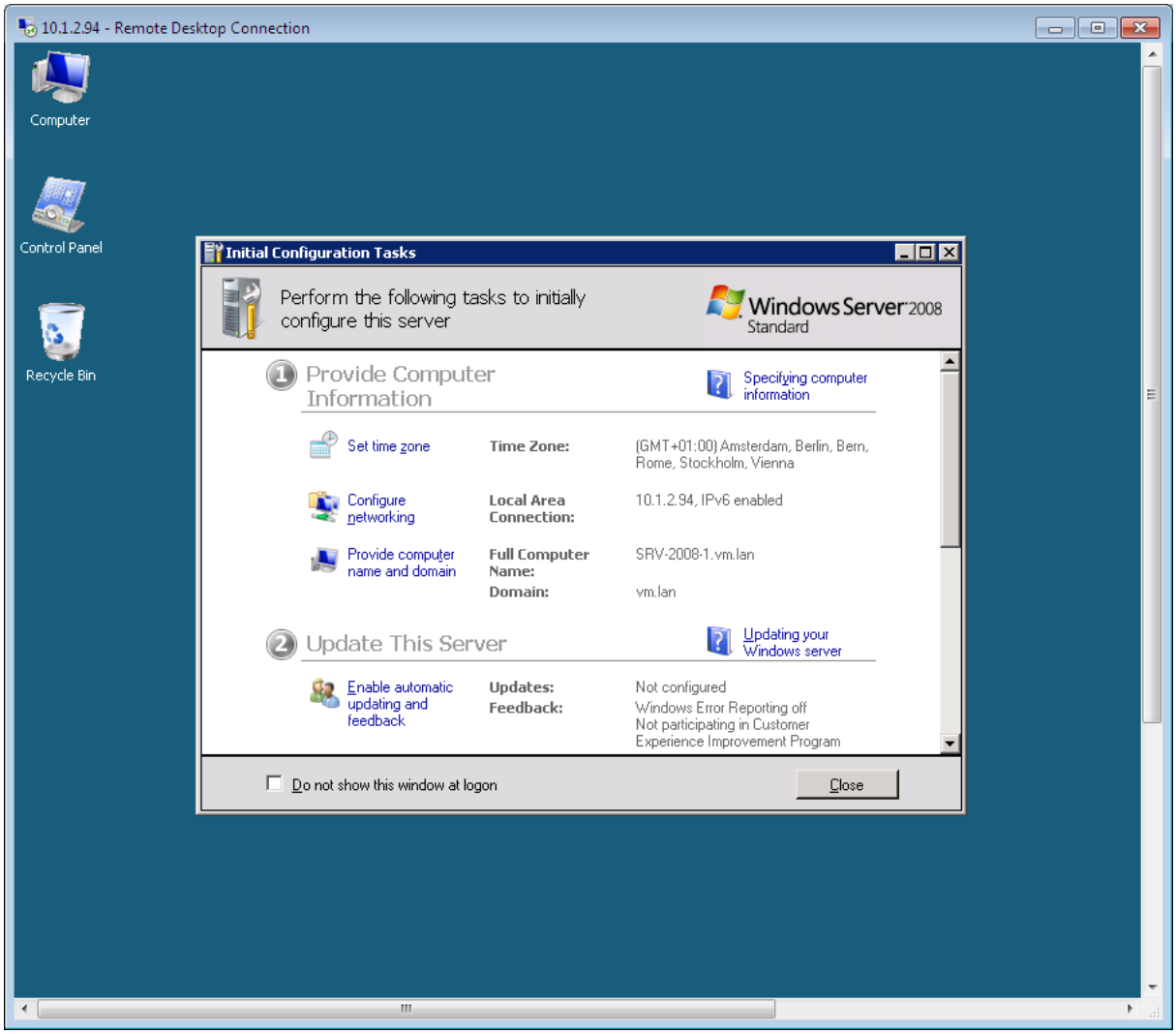
Exécution de la VM Windows 2008
Exécuter la console SPICE pour afficher l'installation de Windows :



Print screen 9: Console SPICE

Une fois l'installation de Windows terminée, nous devons lui attribuer une adresse IP : **10.1.2.95**

- Pour les Administrateurs qui préfèrent utiliser le RDP, je l'ai également activé



Print screen 10: Connexion RDP

5.1.8 Configuration du rôle Active Directory

Pour configurer mon Active Directory, j'ai suivi la démarche expliquée sur ce site : http://syskb.com/installer-active-directory-sur-windows-server-2008/
Mon domaine s'appelle : VM.lan

5.1.9 Intégration d'un poste client au domaine

Concernant le PC Windows Vista, voici les étapes pour l'intégrer au domaine VM.lan :

- Indiquer le serveur DNS, qui est la VM Windows Server 2008 (10.1.2.95).
- Saisir le nom de domaine auquel le PC sera rattaché
- Redémarrer l'ordinateur
- Sur la page d'authentification, rentrer : **VM.lan\nom d'utilisateur présent dans le domaine** et son mot de passe.

5.2 Conclusion à propos de la solution oVirt

La solution oVirt m'a permis de mettre sur pied une infrastructure virtualisée assez facilement grâce à l'utilitaire d'installation présent dans le package oVirt-Engine pour l'installation du manager. De plus, le déploiement de l'hyperviseur est simple à mettre en œuvre grâce notamment à la distribution oVirt-Node intégrant tous les éléments liés à la virtualisation sous KVM, ainsi qu'à la présence d'une interface TUI permettant de configurer les éléments de base.
L'interface web rend le monitoring des ressources utilisées par les VMs sur l'hyperviseur beaucoup plus claire. Depuis cette interface, nous pouvons donner l'ordre de création de nouvelles VMs et de lancer la console d'affichage afin d'interagir directement avec la VM grâce au Plugin Spice intégré au navigateur Internet. Cet outil d'administration est comparable à vSphere chez VMware.
Ma seconde partie de travail de Bachelor doit se baser au niveau de l'isolation des VMs. Sous oVirt, nous pouvons créer des réseaux virtuels mais chaque réseau virtuel doit être affilié à une carte Ethernet physique différente. Donc avec cette méthode, pour créer une isolation entre chaque VM, nous devrions installer une carte physique pour chaque VM, ce qui serait impensable dans un univers professionnel où certains hyperviseurs en exécutent plusieurs dizaines simultanément.
Il faudrait mettre en place un système de VLAN, mais actuellement le manager de oVirt ne le permet pas. De plus nous ne pouvons pas intégrer de couches supplémentaires au sein de l'hyperviseur, du fait que la distribution oVIrt Node bloque tout le système en read-only. C'est à cause de ces différentes limitations que je vais me tourner vers la solution OpenNebula qui intègre des fonctionnalités plus avancées au niveau de la gestion des réseaux, avec la possibilité d'implémenter la solution basée sur OpenvSwitch.

5.3 2e Scénario : Déploiement d'un VM avec OpenNebula

5.3.1 Schéma global de OpenNebula

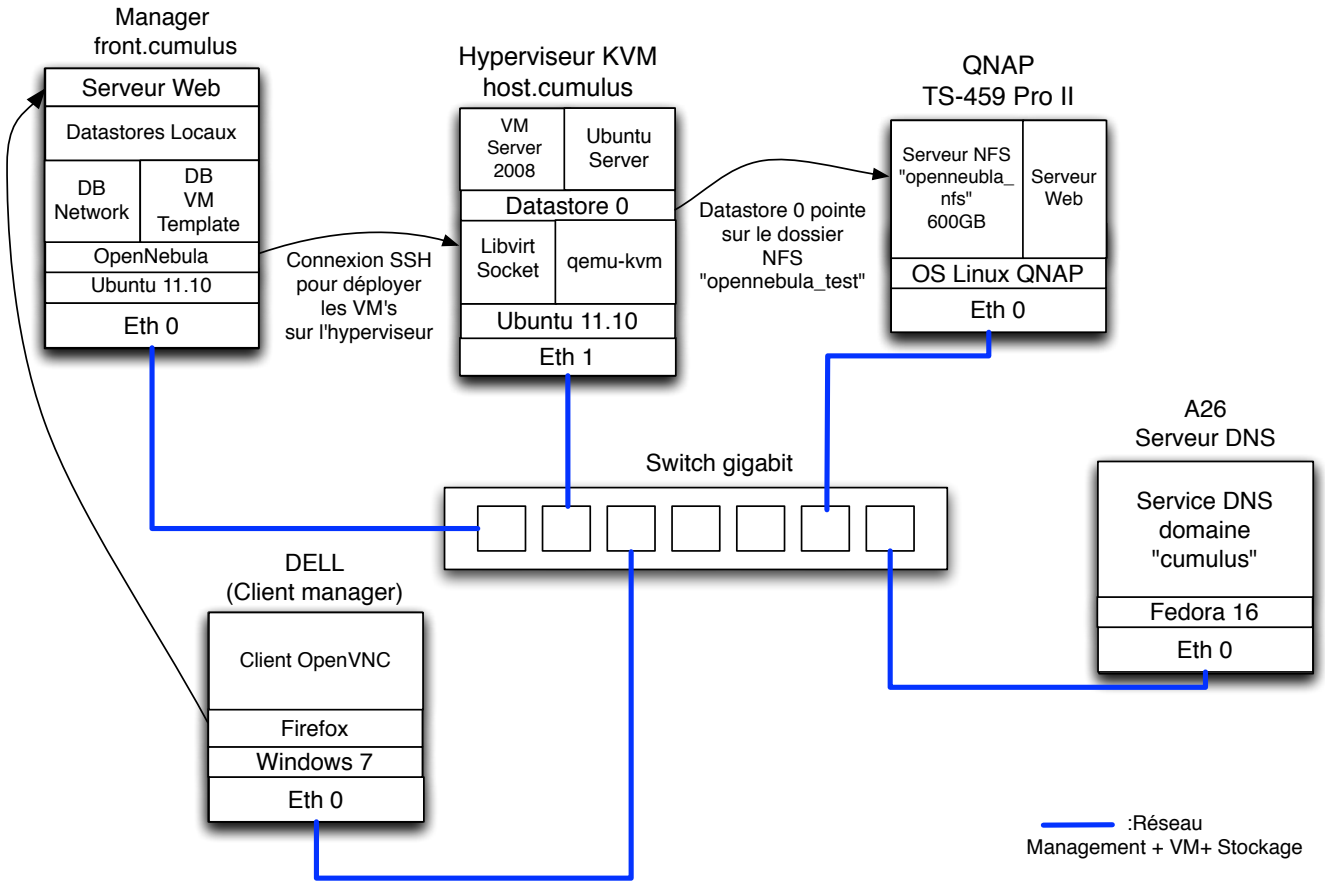


Figure 13 : Schéma complet de l'infrastructure OpenNebula

5.3.2 Introduction :

A travers ce deuxième scénario, nous allons mettre en place l'infrastructure OpenNebula.

- Le manager et l'hyperviseur seront basés sur Ubuntu Server 11.10 64Bits.
- La VM déployée sera un Ubuntu Desktop 11.04 x86.
- La mise en place de l'infrastructure est un peu plus complexe qu'avec oVirt, sachant qu'il n'y a pas d'utilitaire d'installation.

5.3.3 Activation du partage NFS du QNAP

Pour notre scénario les VMs exécutées par l'hyperviseur seront stockées sur le dossier ***nfs-opennebula/test*** du QNAP. Pour cela il faut activer le service NFS dans le QNAP, accessible depuis l'interface web du NAS. Je n'ai appliqué aucune restriction d'accès au dossier NFS. La configuration NFS se fait via un assistant intégré au serveur web.

5.3.4 Installation de OpenNebula sur le manager (front.cumulus)

Comme nous montre la **figure 13**, le manager va être déployé sur le PC front.cumulus. Pour le déploiement, j'ai étudié un document fourni par « CloudBlab »[13]. Au sein de ce PC, un service appelé « one » va gérer la création de VM, et l'autre « sunstone-server », va gèrer l'interface GUI accessible via le serveur web. Pour mener à bien l'installation nous devons effectuer un certain nombre de tâches:

- L'installation du Ubuntu Server doit intégrer le client SSH.
- Créer une arborescence précise au niveau du système de fichier qui sera commun au manager et à l'hyperviseur. Le programme OpenNebula va s'installer dans : ***/var/lib/one/.*** Sachant, que par la suite le déploiement des VMs se fait par l'intermédiaire de script, il faut un certain standard au niveau de la structure des fichiers.
- Un utilisateur et un groupe commun à tous les éléments doivent être créés. L'utilisateur est « **oneadmin** » avec le mot de passe « **oneadmin** » et le groupe « **cloud** ». Il va servir d'administrateur. Son dossier personnel va se trouver ***/var/lib/one/.***
- De nombreux programmes sont requis pour pouvoir exécuter OpenNebula. Entre autre nous devons installer un serveur MySQL, pour la gestion des bases de données, afin de pouvoir rendre compatible OpenNebula avec ESX.
- L'installation du programme se fait via le fichier **opennebula-3.4.1.tar.gz** qu'il faudra recompiler pour intégrer l'utilisation de la base MySQL .

Toutes les commandes de l'installation sont en **Annexe A.4**. Le fichier de configuration doit être édité pour prendre en compte les protocoles de communication entre l'hyperviseur et le manager, ainsi que les mots de passe (oned.conf voir **A.4.7** des annexes).

5.3.5 Installation de KVM sur l'hyperviseur (host.cumulus)

Pour mettre en place l'hyperviseur au moment de l'installation de Ubuntu 11.10 server, il faut choisir d'installer le module de virtualisation basé sur KVM et le serveur SSH. Pour déployer le manager, j'ai suivi le document[14] créé par « Cloudblab » en **page 5**.
Tout comme avec le manager il faut respecter quelques étapes :

- Créer l'utilisateur « oneadmin » et le groupe « cloud ».
- Créer la même arborescence de fichier que dans le manager : **/var/lib/one**
- Charger les modules complémentaires aux modules KVM, afin que le manager puisse communiquer avec l'hyperviseur.
- Modifier les droits d'accès au socket Libvirt, pour accepter les commandes du manager.

5.3.6 Communication entre le Manager et l'hyperviseur KVM

La communication entre l'hyperviseur et le manager se fait via une connexion SSH. Les commandes sont transmises via l'intermédiaire de la socket de communication Libvirt. Au moment du déploiement d'une nouvelle VM, le manager qui possède les Templates des VMs

[13] opennebula-r-3-4-and-vmware-esxi5-0-using-sharedvmware-and-ssh-transfer-drivers.pdf
[14] buidling-a-hybrid-cloud-with-opennebula.pdf

tels que les caractéristiques CPU ou le fichier ISO, va transmettre ces fichiers via le protocole SSH.
Sachant que le déploiement va faire intervenir des scripts implémentés dans OpenNebula, il ne faut pas que le mot de passe soit demandé entre le manager et l'hyperviseur. Pour cela, nous devons indiquer au serveur SSH (Hyperviseur) que le client SSH est autorisé à ce connecter automatiquement.

Figure 14 : Transfert de clé pour autoriser le client à se connecter sans mot de passe

Le détail des commandes pour la génération de la clé et l'envoi au serveur se trouve en **A.5.4 des annexes**. Une fois le transfert effectué, nous devons rentrer une fois le mot de passe administrateur du serveur SSH, pour intégrer la clé au fichier **authorized_key**.
Pour tester la connexion sans mot de passe, on peut exécuter la commande sur le manager :

```
ssh oneadmin@host.cumulus
```

5.3.7 Stockage et Datastores

5.3.7.1 Datastores sur le manager

Le manager va posséder en local les Datastores qui vont héberger les modèles de VMs avec leurs paramètres et les unités de stockages à greffer (Exp : Quelle types de disques ou quels fichiers ISO à monter avec les VMs).Ils sont stocker au chemin ***/var/lib/one/var/datastores/***

5.3.7.2 Stockage des VMs sur l'hyperviseur

Comme nous montre la **figure 13**, le stockage des VMs se fait sur l'unité de stockage QNAP. Pour stocker les VMs, l'hyperviseur va utiliser le protocole NFS. Sur notre QNAP le dossier **opennebula-nfs** est partagé en NFS, et l'hyperviseur monte ce dossier partagé au chemin ***/var/lib/one/var/datastores/0***. Ce chemin doit être standard du fait que le script du manager va déployer les VMs à cet emplacement. Nous avons dû installer un client NFS à l'hyperviseur, et éditer le fichier ***/etc/fstab***, pour indiquer le chemin du partage NFS sur le QNAP.

5.3.8 Configuration de OpenNebula

5.3.8.1 Edition des fichiers de configuration sur le manager

Edition du fichier ***~/etc/oned.conf***

Dans ce fichier nous allons lui indiquer :

- Les paramètres de la base MySQL
- Les drivers de communication à utiliser avec KVM (SSH)
- Le chemin du Datastore de l'hyperviseur (/var/lib/one/var/datastores)

Voir les modifications en **A.4.7** des annexes

Pour exécuter le manager :

```
one start
```

5.3.8.2 Installation du serveur Web Sunstone-Server

Pour installer l'interface GUI gérée par le serveur web **Sunstone**, j'ai suivi le document proposé par OpenNebula[15]. Via cette interface GUI, nous pouvons gérer le déploiement des VMs, les Templates et l'accès à la console via l'utilitaire NoVNC. Par défaut le serveur web écoute sur le port **9869**. On peut éditer les différentes options dans le fichier ***/etc/one/sunstone-server.conf***

Pour démarrer le serveur web

```
sunstone-server start
```

Pour démarrer le service de monitoring, afin de visualiser les graphiques de charges CPU et réseaux :

```
oneacctd start
```

5.3.9 Déploiement de l'infrastructure

Tout d'abord, il faut se connecter au serveur web : http://IP_MANAGER:9869

Pour construire notre Datacenter, nous devons :

- Créer un cluster qui va regrouper tous les hyperviseurs et les datastores du manager.
- Connecter l'hyperviseur au manager en utilisant le menu **Host**. L'ajouter en rentrant son nom de domaine (host.cumulus), et le paramètre KVM et Dummy.
- On crée un datastore sur le manager, qui possède les protocoles de transfert SSH.
- On va éditer deux images (disques des VM), une qui correspondra au fichier ISO et l'autre à un DATABLOCK de 10GB.
- On va éditer les Templates, ce fichier va contenir toutes les informations à propos des VMs comme la quantité de RAM qui va être attribuée, le nombre d'interfaces réseaux, le port sur lequel le serveur VNC de la VM va écouter (Voir **A.7.1 des annexes**).
- Instancier la VM basée sur le Template précédemment édité.
- A ce moment là, le manager va copier les images sur l'hyperviseur au chemin ***/var/lib/one/datastores/0***, comme vu au point **5.3.7.2** le chemin pointe sur le dossier NFS du QNAP.

[15] http://opennebula.org/documentation:rel3.4:sunstone

5.3.10 Affichage de la console de la VM avec vncwiever

Une fois le transfert effectué, on peut ouvrir **vncviewer** et se connecter à la VM grâce au serveur VNC intégrer à l'hyperviseur

```
vncviewer IP_DE_HYPERVISEUR:N° PORT DE LA VM
```

5.4 Scénario n°3 : Isolation des VMs

5.4.1 Schéma Global

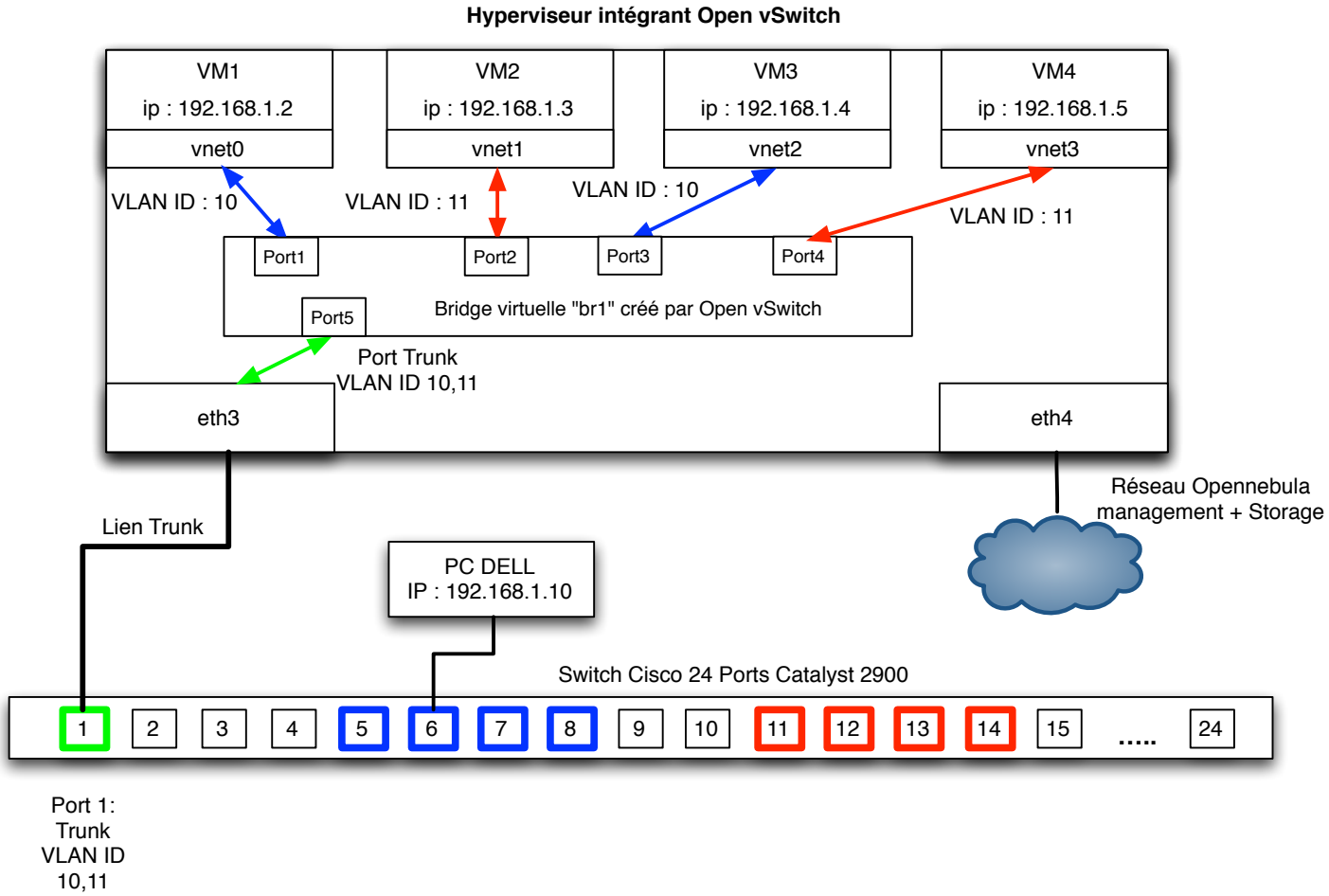


Figure 15: Réseau de VLAN

5.4.2 Introduction

A travers ce scénario (Voir **Figure 15**), nous allons mettre en œuvre une topologie réseau basée sur les VLANs.

5.4.3 Installation de Open vSwitch

Open vSwitch est un module qui va s'installer au niveau du noyau Linux. Nous allons l'installer sur l'hyperviseur host.cumulus, donc nous garderons la topologie globale du scénario précédent. Plusieurs étapes sont nécessaires pour l'installation de Open vSwitch sur l'hyperviseur (la démarche complète se trouve en **A.6 des annexes**)[16]:

- Installer les programmes requis pour l'installation d'Open vSwitch.
- Télécharger Open vSwitch v1.4.1.
- Compiler le programme afin d'installer les différents modules au bon endroit dans le noyau.
- Exécuter le script d'installation.

[16] http://openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=INSTALL.Linux;hb=HEAD

- Créer un dossier dans : ***/usr/local/etc/openvswitch*** qui va contenir la base de donnée.
- Exécuter la création de la base de donnée.
- Comme Open vSwitch va remplacer l'outil de base de KVM bridge, il faut intégrer un mode de compatibilité[17], en déchargeant le module « bridge » et en le remplaçant par les modules Open vSwitch : openvswith_mod.ko et brcompat_mod.ko.
- Comme OpenNebula doit interagir avec Open vSwitch, il ne faut pas qu'il y ait de demande de mot de passe pour exécuter la commande « ovs-vsctl ». Donc modifier le fichier ***/etc/sudoers***.

5.4.4 Démarrage du module Open vSwitch

A chaque démarrage de l'hyperviseur, le module Open vSwitch ne démarre pas automatiquement sachant qu'il est remplacé par le module natif à KVM : **bridge**. Alors j'ai créer un script : start_deamon_open_vswitch.sh. Ce script est à exécuter à chaque démarrage de l'hyperviseur (Voir **A.6.8** des annexes).

5.4.5 Déploiement des 4 VMs

Pour déployer les VMs via OpenNebula, il faut respecter quelques manipulations :

- Exécuter sur l'hyperviseur le script de démarrage de Open vSwitch:

```
sudo sh start_deamon_ovs.sh
```

- Créer le bridge virtuel appelé « br1 » avec les commandes de Open vSwitch

```
sudo ovs-vsctl add-br br1
```

En nous basant sur le scénario précèdent, nous allons déployer 4 VMs Ubuntu Desktop basées sur le Template en **A.7.1**. L'ordre d'instanciation est important du fait que la première VM va posséder une carte réseau virtuelle « vnet0 » et la suivante « vnet1 ». Le fait de respecter cet ordre sera plus simple par la suite pour paramétrer les VLANs. Une fois nos 4 VMs en mode « Running », nous allons paramétrer les adresses IPs pour chacune d'entre elles en se connectant via la console **vncviewer**. (Voir configuration IP : **Figure 15**)

5.4.6 Connexion des 4 VMs au bridge et configuration des VLANs

Par défaut toutes les cartes virtuelles « vnet » sont rattachées au bridge « br1 ». Pour pouvoir attribuer un ID de VLAN à chaque interface il faut :

- Déconnecter toutes les interfaces virtuelles du « br1 » :

```
sudo ovs-vsctl del-port vnet0      --A faire pour tout
                                   les vnets[0 à 3]
```

- Rajouter chaque « vnet » au bridge « br1 » avec le tag souhaité :

```
sudo ovs-vsctl add-port br1 vnet0 tag=10
```

[17] http://openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=INSTALL.KVM;hb=HEAD

```
sudo ovs-vsctl add-port br1 vnet2 tag=10
```

```
sudo ovs-vsctl add-port br1 vnet3 tag=11
```

```
sudo ovs-vsctl add-port br1 vnet1 tag=11
```

Chaque interface virtuelle est configurée dans un VLAN, mais pour l'instant aucune VM ne peut pinger un PC à l'extérieur de l'hyperviseur, sachant qu'il n'y a pour le moment aucune interface physique connectée au bridge.

- Configurer l'interface « eth3 » en mode Trunk

```
sudo ovs-vsctl add-port br1 eth3 trunks=10,11
```

5.4.7 Automatisation de l'attribution des VLANs

Pour attribuer plus aisément les tags aux « vnets », j'ai mis au point un script. Il va déconnecter et reconnecter les vnets en ajoutant le VLAN ID (Le code source en **A.6.9**). Auparavant il faut que le bridge soit créé :

```
sudo ./vlan.sh br1 vnet0 tag=10 vnet1 tag=11 vnet2
tag=10 vnet3 tag=11 eth3 trunks=10,11
```

- Le 1er argument est obligatoirement le nom du bridge.
- Les suivants vont par paires il y a à chaque fois le nom de l'interface virtuelle et l'attribut que l'on veut lui rajouter (tag ou trunk).
- Une fois l'attribution effectuée, nous avons à l'écran le contenu du bridge.

5.4.8 Configuration du Switch Cisco 24 ports Catalyst 2900

Afin de propager les VLANs à l'extérieur de notre hyperviseur, nous devons chaîner un switch Cisco.

- Le port n°1 du switch sera connecter au port Trunk « eth3 ».
- Les ports 5 à 8 seront dans le VLAN 10.
- Les ports 11 à 14 seront dans le VLAN 11.

Pour ce faire, il faut se connecter au switch Cisco via le port COM du PC DELL sous Windows 7, au port « console » du switch, grâce à un câble RS232. La marche à suivre en détaillé dans le protocole de laboratoire de Mr Jenny « Interworking VLAN ». La configuration du switch est **A.6.10**.

5.4.9 Tests

Une fois la configuration terminée, voici les tests effectués pour valider notre configuration :

- Quand le PC Dell est connecté au port du switch Cisco dans le VLAN 10, il peut pinger les VMs 1 et 3, et vice versa.
- Quand le PC Dell est connecté au port du switch Cisco dans le VLAN 11, il peut pinger les VMs 2 et 4, et vice versa.
- Mais la VM 2 par exemple ne peut pinger la VM 3.

5.5 Scénario final n°4: Isolation des VMs + Live Migration + Supervision

Voici les différentes étapes de la mise en œuvre de mon scénario final regroupant au sein d'une même infrastructure tous les éléments vus au cours des scénarios précédents.

5.5.1 Schéma global

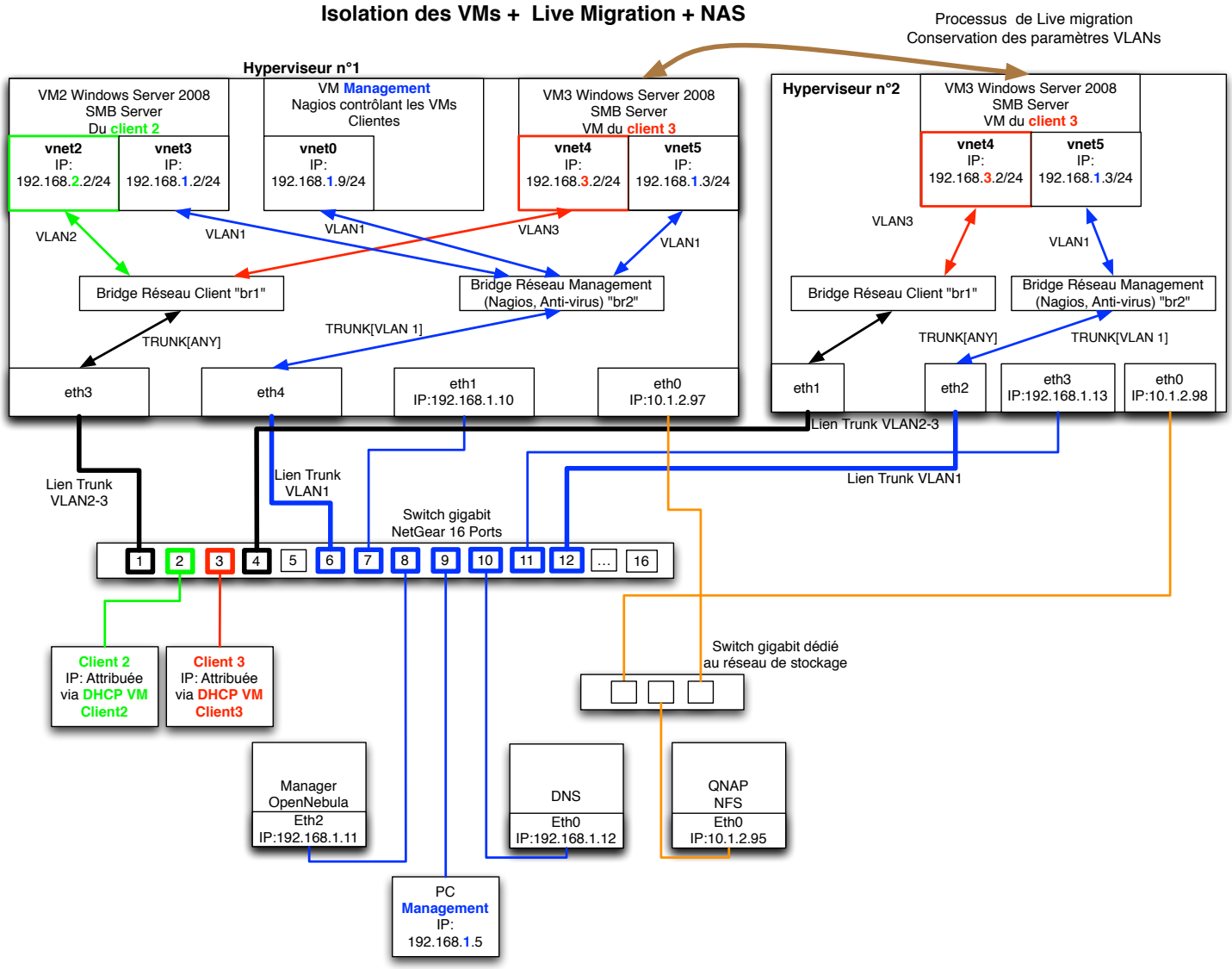


Figure 16 : Schéma global du scénario final

5.5.2 Switch physique

Contrairement au scénario n°3, j'ai abandonné le switch Cisco **Catalyst 2900** 100 Mb/s, pour utiliser un switch 1000 Mb/s 16 ports **Netgear GS116E**. J'ai effectué ce changement de matériel car comme illustré sur la **figure 16**, le switch physique est l'élément central de mon réseau, il va donc devoir aiguiller un trafic élevé, c'est pour cela que j'ai décidé d'en utiliser un avec un débit maximum.

5.5.3 Utilisation du NFS au lieu du iSCSI

Pour pouvoir appliquer mon scénario avec le mode **live migration**, je n'ai pas utilisé le protocole iSCSI pour la communication entre les deux hyperviseurs et le NAS, mais le protocole NFS. Comme ces deux hyperviseurs doivent pointer sur le même partage réseau, il nous faut un protocole qui puisse gérer des accès simultanés à un même stockage. Deux initiateurs iSCSI ne peuvent pas accéder au même LUN en même temps.

5.5.4 Installation du deuxième hyperviseur

J'ai configuré mon deuxième hyperviseur KVM (host2.cumulus) comme le premier, en utilisant les paramètres IP de la **figure 16**.

5.5.5 Partage des clés SSH pour la Live Migration

J'ai partagé les clés publiques destinées à la connexion SSH, pas seulement entre les hyperviseurs et le manager, mais également directement entre les hyperviseurs, pour qu'ils puissent communiquer directement entres eux. (Voir Annexe **A.5.4**)

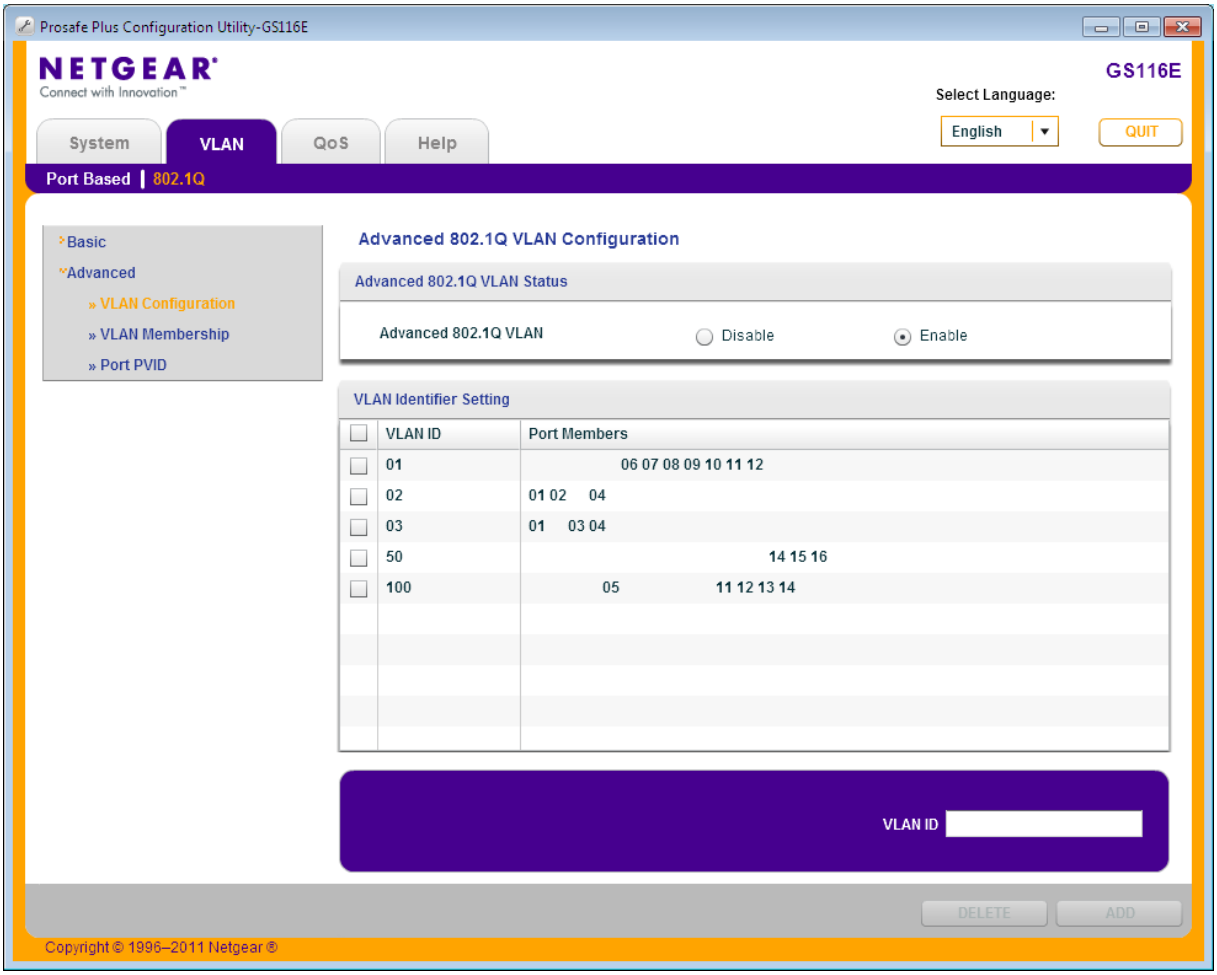
5.5.6 Configuration de Open vSwitch

J'ai créé mes deux bridges (br1 et br2) en lignes de commandes sur les deux hyperviseurs, en reliant les cartes réseaux physiques (Voir Scénario n°3). Cette configuration redondante est essentielle pour la Live Migration, du fait qu'une VM doit retrouver les mêmes configurations de bridges en passant d'un hyperviseur à l'autre.

5.5.7 Configuration des VLANs

- Au niveau du switch Netgear :

La configuration des VLANs se fait via un utilitaire GUI installé sur Windows 7 fournit par Netgear, le mot de passe est : password. J'ai configuré mes VLANs comme sur la **figure 16**. J'ai dû également paramétrer des ports trunks redondants, nécessaires pour la Live Migration afin de recréer les mêmes connexions aux VLANs entre les deux hyperviseurs (Voir Print-screen 11 ci-dessous).



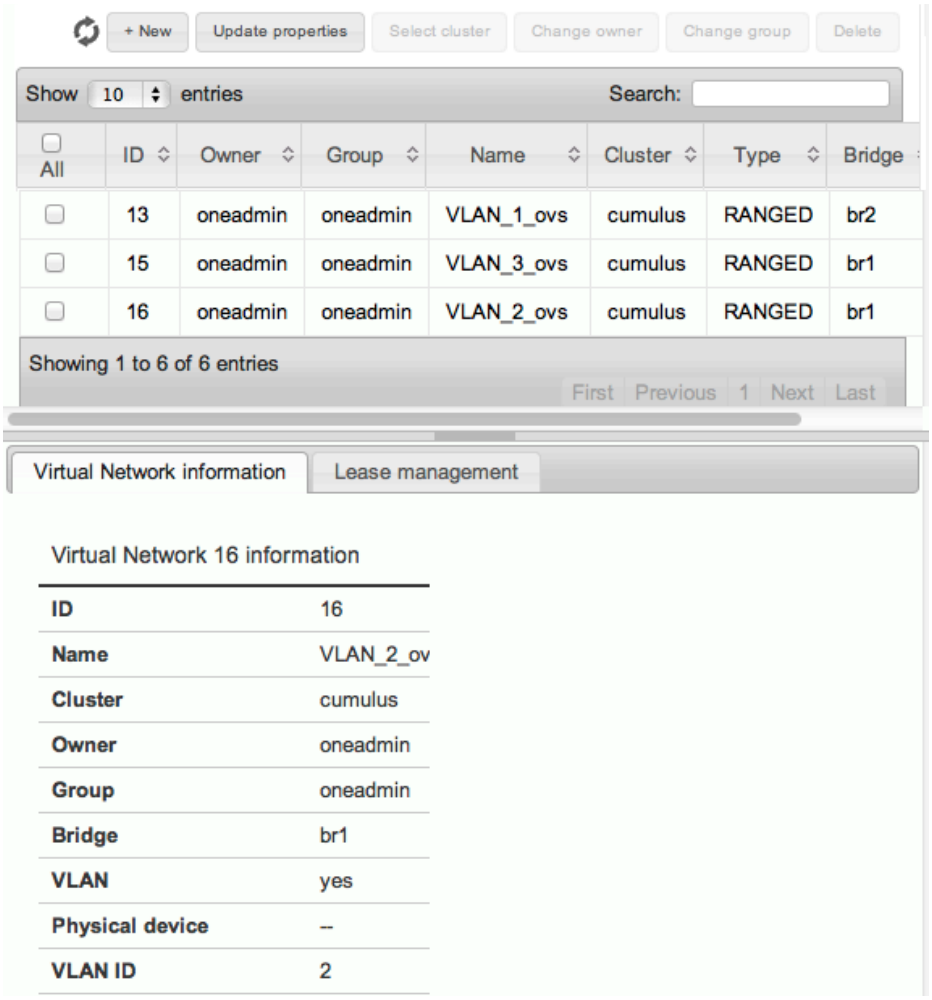
Print screen 11 : Utilitaire de configuration du switch Netgear

	VLAN1	VLAN2	VLAN3
Fonction	Réseau de Management	Réseau du Client 2	Réseau du Client 3
Plage Réseau	192.168.1.1/24	192.168.2.1/24	192.168.3.1/24
Couleur sur schéma	Bleu	Vert	Rouge
N° de ports	6.7.8.9.10.11.12	1.2.4	1.3.4

Tableau 1: Résumé des VLANs

- Au niveau des VMs :

Pour ce scénario, la configuration des VLANs ne se fait plus directement sur l'hyperviseur via les commandes Open vSwitch, car cette méthode est trop éphémère du fait qu'elle se base sur le nom des interfaces virtuelles.
Mais comme j'utilise le procédé de Live Migration, les interfaces virtuelles (vnet) vont disparaître d'un hyperviseur pour se recréer dans l'autre, il faudrait donc reconfigurer les attributions des VLANs ID aux vnets après chaque migration.
Pour pallier à ce problème, j'ai dû paramétrer les VLANs ID au niveau du template de la VM, je lui indique par exemple qu'une interface virtuelle sera connectée au bridge br1 et possèdera le VLAN ID 2. Pour cela il faut créer depuis l'interface GUI de OpenNebula un Virtual Network par VLAN :



Print screen 12: Template configuration réseau

Le Print-screen ci-dessus indique la configuration que va prendre les interfaces virtuelles utilisant ce template. Par exemple, toutes les interfaces virtuelles des VMs qui vont prendre le template VLAN_2_ovs :

- Auront le **VLAN ID n° 2**.
- Seront connectées au bridge **br1**.

Grâce à ce principe, une VM déployée pour la première fois ou tout simplement migrée va reprendre la configuration du template, d'où l'importance de mettre les mêmes noms de bridges au sein de tous les hyperviseurs.

5.5.8 Déploiement des VMs

J'ai déployé mes VMs en suivant les templates des **Annexes A.7.2**.

5.5.9 Configurations des serveurs DHCP et Samba sous Windows Server 2008

Pour configurer le serveur DHCP des Windows Server, j'ai suivi le tutoriel fournit par le site : http://www.toutwindows.com/ws2008r2_dhcp.shtml

Pour le partage SMB, j'ai partagé un dossier appelé SRV_FILE sur chacun des serveurs. J'ai également créé un utilisateur qui pourra se connecter au partage. L'utilisateur client2, pour le partage sur la VM_Client2, et un autre client3.

	VM_Client2	VM_Client3
Plage adresses IP DHCP	192.168.2.30-60	192.168.3.100-150
Utilisateur permettant d'accéder au partage	client2	client3
Adresses IP	192.168.2.2	192.168.3.2

Tableau 2: Résumé des fonctionnalité des Windows Server 2008

5.5.10 Configuration de la VM de Management

La VM de management basée sur un Ubuntu Desktop 10.04 va être notre nœud central de supervision. Le contrôle des états de tous les éléments va être assuré par le service Nagios. Ce service va intégrer un serveur web pouvant être interrogé depuis notre réseau de management.
Pour la configuration IP voir **Figure 16.**
J'ai intégré son nom « vm-mgt » au serveur DNS, ainsi, nous pourront interroger le serveur web via son nom de domaine : http://vm-mgt/nagios3

Installation du package Nagios

```
sudo apt-get install nagios
```

Utilisateur : nagiosadmin Mot de passe : nagios

5.5.10.1 Modification de la base de temps

Une fois l'installation terminée, je vais modifier la base de temps, car par défaut Nagios va contrôler l'état de mes serveurs toutes les 2 minutes. Cet intervalle est trop élevé pour mes tests. Je vais choisir 30 secondes, il faut aller éditer le fichier : ***/etc/nagios3/nagios.cfg***
A l'emplacement **Interval_lenght = 30** au lieu de 60.

5.5.10.2 Configuration des fichiers hosts

Pour indiquer à Nagios quels hôtes il doit contrôler, nous devons le faire manuellement en éditant des fichiers .cfg dans ***/etc/nagios3/conf.d/.*** J'ai choisi de créer un fichier par hôte.
De manière générale ce fichier va contenir le nom de l'hôte, son adresse IP, et les services à contrôler.

Hôtes contrôlés par Nagios	Services testés
dns.cumulus	Check-host-alive Check-DNS
front.cumulus	Check-host-alive Check-http
host.cumulus	Check-host-alive
host2.cumulus	Check-host-alive
VM-Client2	Check-host-alive
VM-Client3	Check-host-alive

Tableau 3: Résumé des services testés

Les services testés :

- **Check_host_alive.** Cette commande va envoyer un Ping et va attendre une réponse de l'hôte pour afficher son statut **UP**
- **Check-DNS** : Nagios va demander au serveur DNS quelle adresse IP possède ce nom de domaine (Pour ma part je lui demande de faire le test pour la résolution front.cumulus)
- **Check-http** : Nagios va envoyer une requête http au serveur front.cumulus sur le port du serveur web OpenNebula 9869

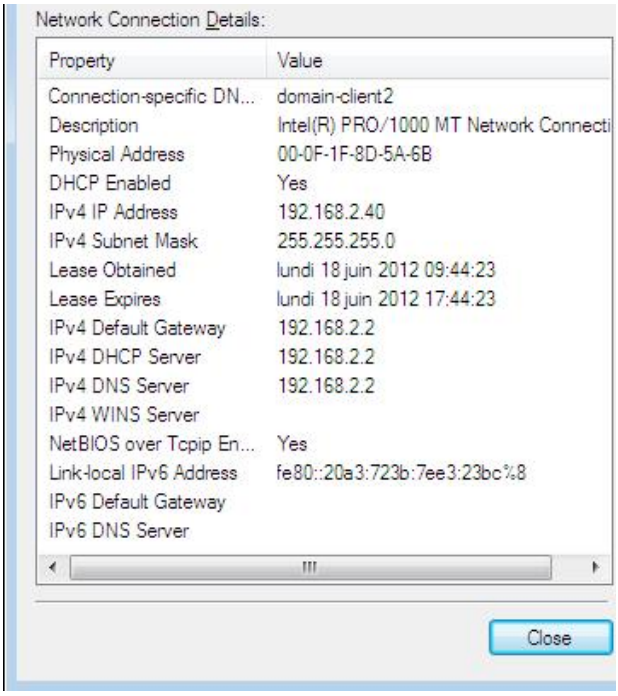
Pour les VM-Client2 et 3 Nagios test leurs interfaces de management, sachant qu'il ne possède pas de lien avec les réseaux des clients. Grâce cette architecture, le « réseau client » sera totalement isolé.

5.5.11 Les Tests

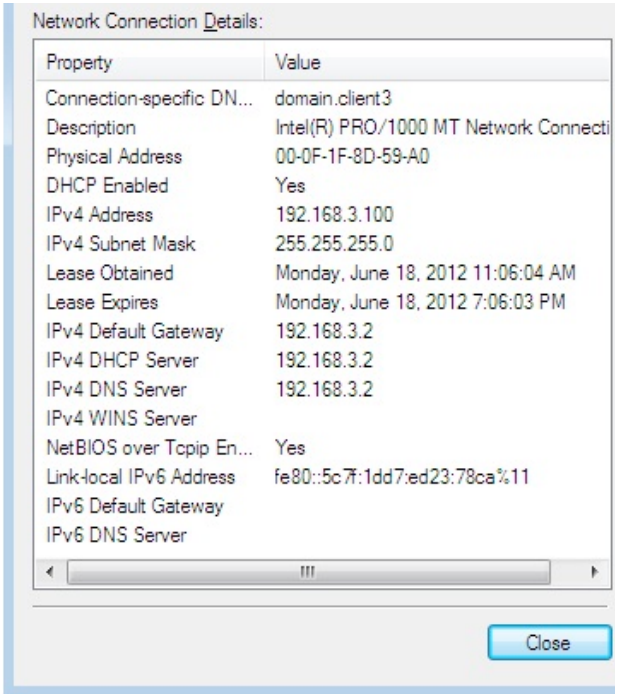
Pour valider le scénario j'ai effectué plusieurs tests :

5.5.11.1 Tests de cloisonnement des réseaux des Clients

Pour tester l'isolation des « réseaux clients », j'utilise les serveurs DHCP de mes deux VMs.
Comme PCs clients j'utilise deux PC Dell, configuré en DHCP.
Au moment où je connecte le PC DELL du Client2 sur le port n°2 du Switch je reçois une IP dans la plage 192.168.2.30-60. Et quand le PC Dell du client3 est connecté au port n°3 du Switch je reçois une IP dans la plage 192.168.3.100-150.
Grâce à ce test, j'ai pu confirmer que les réseaux des clients sont bien cloisonnés, du fait que depuis le réseau du client2 au moment de l'échange DHCP, je ne reçois aucune trame émanant du réseau du client3.



Print screen 14 : Configuration IP PC Client 2



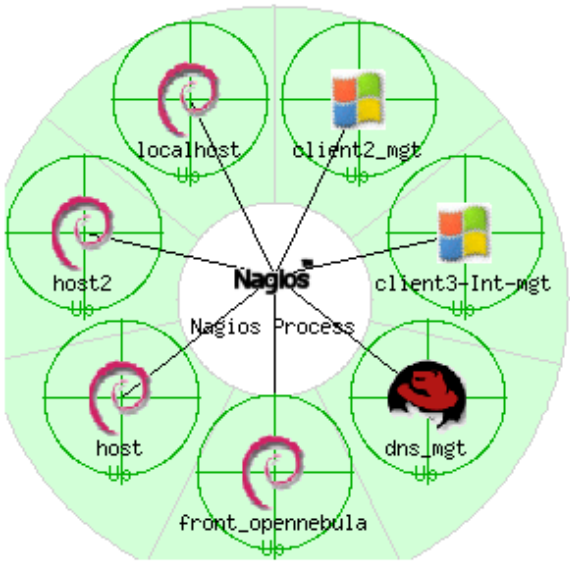
Print screen 14 : Configuration IP PC Client 3

5.5.11.2 Tests au niveau des serveurs SMB

Au moment où le PC client2 a reçu une adresse DHCP, je peux interroger mon serveur SMB à l'adresse : \\192.168.2.2 Et je rentre comme utilisateur et mot de passe : client2
Idem pour le partage du client3, utilisateur mot de passe : client3

5.5.11.3 Test de supervision

Une fois tous les fichiers de configuration pour chaque hôte créés, nous pouvons démarrer le service Nagios, depuis le PC de Management nous allons interroger le serveur web, http://vm-mgt/nagios3 ou http://192.168.1.9/nagios3
Voici un Print-Screen de la Status map générée par Nagios :



Print screen 15 : Status Map Nagios

Voici la page qui résume l'état de tous mes éléments contrôlés par Nagios :

Host Status Details For All Host Groups

Host	Status	Last Check	Duration	Status Information
client2_mgt	UP	2012-06-18 11:31:15	0d 2h 59m 58s	PING OK - Paquets perdus = 0%, RTA = 4.03 ms
client3-Int-mgt	UP	2012-06-18 11:31:15	0d 2h 33m 28s+	PING OK - Paquets perdus = 0%, RTA = 5.04 ms
dns_mgt	UP	2012-06-18 11:31:35	4d 19h 1m 34s	PING OK - Paquets perdus = 0%, RTA = 0.39 ms
front_opennebula	UP	2012-06-18 11:31:15	2d 23h 42m 6s	PING OK - Paquets perdus = 0%, RTA = 3.88 ms
host	UP	2012-06-18 11:31:25	0d 2h 33m 28s+	PING OK - Paquets perdus = 0%, RTA = 2.54 ms
host2	UP	2012-06-18 11:31:45	0d 2h 33m 28s+	PING OK - Paquets perdus = 0%, RTA = 0.49 ms
localhost	UP	2012-06-18 11:31:15	5d 20h 50m 17s	PING OK - Paquets perdus = 0%, RTA = 0.03 ms

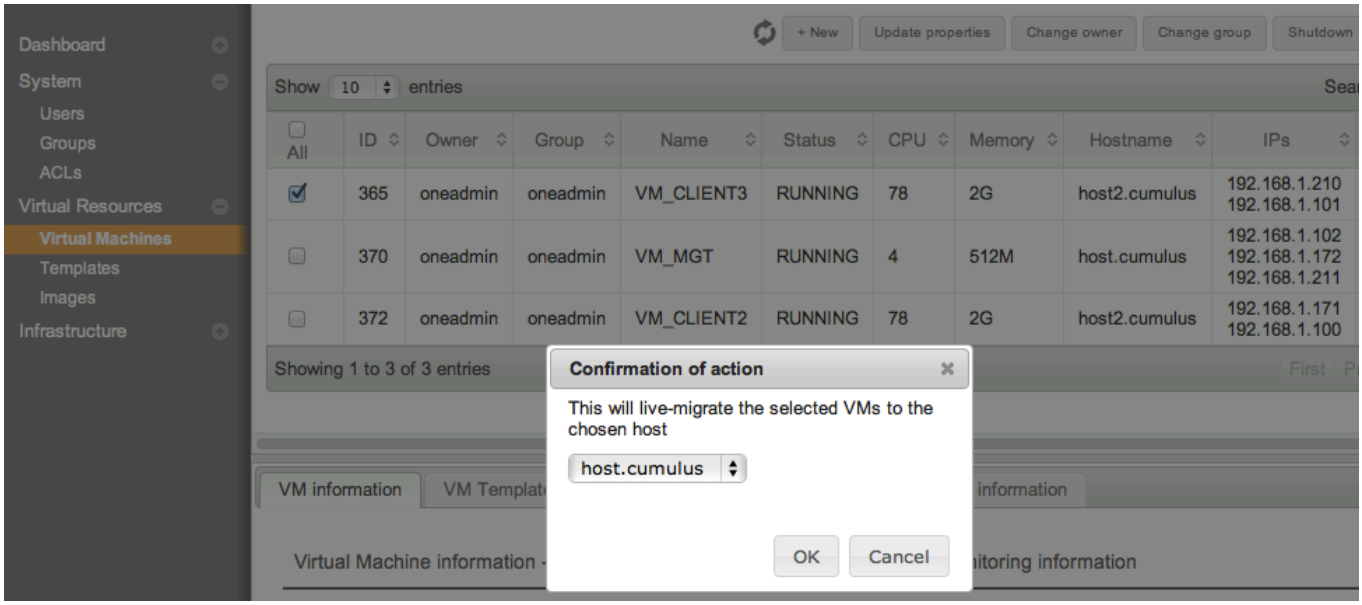
7 Matching Host Entries Displayed

Print screen 16 : Listes des éléments supervisés par Nagios

Grâce à l'outil de supervision, nous pouvons consulter l'état des VMs clientes par l'intermédiaire de leurs interfaces de management.

5.5.11.4 Test de Live Migration

Comme tout est configuré pour la Live Migration nous pouvons faire un test qui consiste à déplacer la VM d'un hyperviseur à l'autre. La Live Migration se commande depuis l'interface web de OpenNebula, dans Virtual Machine :



Print screen 17 : Menu Live Migration de l'interface GUI OpenNebula

On choisit la VM que l'on veut migrer vers l'hyperviseur désiré. Le manager s'occupe de stopper le processus KVM de l'un pour le ré-exécuter sur l'autre hyperviseur. Pour transférer la RAM de la VM de l'un à l'autre.

- Voici le temps que prend la Live Migration de la **VM Windows server 2008 Client3** de l'hyperviseur **host2.cumulus** vers **host.cumulus**

```
Mon Jun 18 11:36:01 2012 [LCM][I]: New VM state is MIGRATE
Mon Jun 18 11:36:10 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:36:12 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:36:12 2012 [VMM][I]: Successfully execute network driver operation: pre.
Mon Jun 18 11:36:35 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:36:35 2012 [VMM][I]: Successfully execute virtualization driver operation: migrate.
Mon Jun 18 11:36:35 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:36:35 2012 [VMM][I]: Successfully execute network driver operation: clean.
Mon Jun 18 11:36:35 2012 [VMM][I]: post: Executed "sudo /usr/local/bin/ovs-vsctl set Port vnet3 tag=3"
Mon Jun 18 11:36:35 2012 [VMM][I]: post: Executed "sudo /usr/local/bin/ovs-vsctl set Port vnet4 tag=1"
Mon Jun 18 11:36:35 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:36:35 2012 [VMM][I]: Successfully execute network driver operation: post.
Mon Jun 18 11:36:35 2012 [LCM][I]: New VM state is RUNNING
```

Print screen 18 : Log Live Migration VM Windows Server 2008

D'après le fichier Log, la Live Migration prends environ de 34 secondes. Pendant ce temps, aucune interruption de service n'est détectée par Nagios.

- Voici le temps que prend la Live Migration de la VM de Management Ubuntu Desktop

```
Mon Jun 18 11:56:19 2012 [LCM][I]: New VM state is MIGRATE
Mon Jun 18 11:56:21 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:56:30 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:56:30 2012 [VMM][I]: Successfully execute network driver operation: pre.
Mon Jun 18 11:56:56 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:56:56 2012 [VMM][I]: Successfully execute virtualization driver operation: migrate.
Mon Jun 18 11:56:56 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:56:56 2012 [VMM][I]: Successfully execute network driver operation: clean.
Mon Jun 18 11:56:56 2012 [VMM][I]: post: Executed "sudo /usr/local/bin/ovs-vsctl set Port vnet2 tag=1".
Mon Jun 18 11:56:56 2012 [VMM][I]: post: Executed "sudo /usr/local/bin/ovs-vsctl set Port vnet3 tag=2".
Mon Jun 18 11:56:56 2012 [VMM][I]: post: Executed "sudo /usr/local/bin/ovs-vsctl set Port vnet4 tag=3".
Mon Jun 18 11:56:56 2012 [VMM][I]: ExitCode: 0
Mon Jun 18 11:56:56 2012 [VMM][I]: Successfully execute network driver operation: post.
Mon Jun 18 11:56:56 2012 [LCM][I]: New VM state is RUNNING
```

Print screen 19 : Log Live Migrtation VM Management

Le temps pour migrer cette VM, tout comme la précédente, il aura fallu environ 37 secondes.

Pour l'utilisateur, le fait que la VM soit sur le host.cumulus ou sur le host2.cumulus n'a aucun impact sur la façon d'utiliser le serveur web Nagios ou le partage SMB, car les configurations réseaux sont identiques sur les deux hyperviseurs.

5.5.1 Analyses des risques du scénario

Dans le dernier scénario, le choix d'avoir intégrer deux vnets aux deux VM_Client peut dans un premier temps montrer une faille de sécurité, car si le mode routing est activé sous Windows Server, on pourrait accéder au réseau de management. Pour cela, il faut restreindre au maximum les droits laissés au client. Dans mon cas le client utilise uniquement des services de la VM Windows Server, tel que le serveur DHCP ou le File Server. Il n'a pas d'accès direct au terminal de la VM.
Si un client change son adresse IP pour essayer d'aller interroger la VM de l'autre client, il n'aura aucun moyen d'aller vers l'autre VM car j'ai mis en place un cloisonnement par VLAN, chaque client possède son propre VLAN.

5.6 Conclusion à propos de la solution OpenNebula

La solution OpenNebula mise en œuvre tout au long des scénarios n°2, 3, 4, m'a permis de créer une infrastructure :

- Basée sur la virtualisation Open-Source grâce à KVM.
- Garantissant une isolation des VMs. Chaque client va pouvoir interroger sa VM qui lui est attribuée, ceci grâce aux bridges virtuels gérés par Open vSwitch.
- Garantissant une haute disponibilité sans interruption de service, au moment où une VM est transférée d'un hyperviseur à un autre grâce à la Live Migration.
- Permettant de contrôler l'état (disponibilité) de tous les éléments névralgiques tels que les VMs des clients, le serveur DNS, les hyperviseurs, ceci grâce au réseau de management et au logiciel de supervision Nagios.
- Basée sur l'évolution du système de stockage, car en passant au projet OpenNebula, j'ai abandonné le PC sous Fedora 16 hébergeant ma cible iSCSI, pour intégrer l'unité de stockage QNAP ts-459 Pro II.

J'ai également réussi à intégrer un hyperviseur ESXi 5.0[18] au manager OpenNebula. Je ne l'ai pas expliqué dans mes scénarios car le temps m'était compté. Le point faible réside dans le fait que fait nous ne pouvons pas nous affranchir de vSphere pour administrer l'ESXi. En ce qui concerne du Live Migration entre deux ESXi, je n'ai pas mis en œuvre ce scénario, mais d'après la documentation d'OpenNebula, le système vMotion[19] de VMware est nécessaire.

6 Difficultés rencontrées

6.1 Scénario 1 : oVirt

6.1.1 Bibliothèque ISO

- Symptôme

J'ai eu quelques problèmes pour peupler la bibliothèque d'ISO, au moment j'uploadais mes images ISO au sein du Manager. Les images étaient chargées physiquement, mais la base de données n'arrivait pas à parcourir le dossier, car le dossier était verrouillé après l'upload.

- Solution

Pour pallier à ce problème après chaque upload, il faut changer les droits du dossier de la bibliothèque ISO.

```
sudo chmod 0755 /data/iso
```

6.1.2 Console d'affichage Spice sous Windows 7

- Symptôme

J'ai rencontré quelques problèmes pour afficher le terminal d'une VM sous Windows 7 avec Internet Explorer 8.

- Solution

Il a fallu rajouter un plug-in à IE8 fourni par Red Hat (RHEV Spice Client.msi).

6.1.3 Gestion des cartes réseaux physiques sur l'hyperviseur

- Symptôme

Pour la gestion du réseau applicatif et du réseau de stockage, j'ai dû installer une carte réseau dédiée au trafic des VMs et du management (communication Manager <-> Hyperviseur) et une autre au trafic de stockage. Au début, j'ai intégré les deux cartes dans le même réseau, mais il y avait beaucoup de problèmes de pertes de connexions.

- Solution

Si il y a deux cartes réseaux sur un même PC, il faut que les deux cartes soient dans des réseaux différents, afin d'éviter les problèmes de routages. Le LAN de stockage est dans le réseau 192.168.1.x et le LAN de management dans le réseau 10.1.2.9x.

[18] opennebula-r-3-4-and-vmware-esxi5-0-using-sharedvmware-and-ssh-transfer-drivers

[19] http://opennebula.org/documentation:rel3.4:evmwareg

6.1.4 Gestion des réseaux virtuelles

- Symptôme

Au moment de greffer une interface physique à un réseau virtuel l'hyperviseur plantait, et il m'était impossible d'administrer l'hyperviseur depuis le manager.

- Solution

Par palier à ce problème, il faut au sein de l'interface graphique mettre l'hyperviseur en mode « Maintenance » afin que les modifications de configuration puissent être effectuées.

Pour maîtriser correctement le produit oVirt, afin de créer un infrastructure complète, j'ai passé environ 8 jours à travailler sur ce projet.

6.2 Scénario 2 : Déploiement d'une VM via OpenNebula

6.2.1 Installation du manager

Pour installer le manager, il faut créé une arborescence au niveau des fichiers particulière, ainsi qu'un nom d'utilisateur précis. Un certain nombre de programmes sont requis pour installer OpenNebula. Pour mener à bien l'installation il faut suivre le **paragraphe A.4** des annexes.

6.2.2 Installation de l'Hyperviseur

J'ai rencontré quelques problèmes pour configurer l'hyperviseur KVM, les problèmes se sont situés au niveau de la configuration de la socket libvirt. J'ai dû modifier les droits d'accès, afin qu'elle accepte les commandes manager venant de l'extérieur voir le **paragraphe A.5 des annexes**.

6.2.3 Gestion du Datastore de l'hyperviseur

Sur notre hyperviseur j'ai créé un dossier qui va contenir les disques de la VM, ce dossier doit se trouver à la place ***/var/lib/one/var/datastores/0***, mais en réalité se dossier va permettre de monter le dossier NFS hébergé sur le QNAP. Il faut respecter exactement ce chemin, car le Manager possède une variable qui indiquera au script de déploiement où aller mettre la VM.

6.2.4 Communication Manager <-> Hyperviseur

- Symptôme

La communication entre l'hyperviseur et le Manager se fait via le protocole SSH. Aucun mot de passe ne doit être demandé entre les deux éléments. Il faut que le Manager soit reconnu par l'hyperviseur.

- Solution

Le manager génère une clé Publique, et la transmet à l'hyperviseur afin que ce dernier puisse l'intégrer dans un dossier **Authorized_keys**.

J'ai passé beaucoup de temps à installer OpenNebula (environ 18 jours), entre le temps de recenser de la documentation, de comprendre la topologie globale au niveau des Datastores, et de mettre en place les différentes entités.

6.3 Scénario 3 : Isolation des VMs avec Open vSwitch

6.3.1 Exécution de Open vSwitch

Comme Open vSwitch ne se démarre pas au démarrage du système, j'ai dû créer un script qui va exécuter les commandes de connexions à la base de données. Voir **§5.4.4**

6.3.2 Greffes des vnets au bridge

La première VM que va déployer OpenNebula sur l'hyperviseur, KVM va attribuer comme nom de vnet : vnet0, puis si une deuxième VM arrive avec une interface virtuelle, il va lui attribuer vnet1. Comme la connexion au bridge se fait par nom d'interface, pour respecter notre schéma de VLAN, il faut que l'ordre d'exécution des VMs restent le même.
EXP : Si la VM1 doit toujours avoir le vnet numéro 0. Il faudra l'exécuter en premier.

6.3.3 Port Trunk

- Symptôme

Comme dans mon schéma le port Trunk de l'hyperviseur est une interface physique, j'ai eu des problèmes pour propager les trames 802.1Q sur le switch Cisco.

- Solution

Le problème venait de la carte réseau en elle même (), du fait qu'elle n'était pas compatible avec la norme 802.1Q. Donc j'ai dû utiliser une carte Intel PCI-Express double ports.
Note pour le professeur : La carte réseau **ZyXEL GNb80-T** 1 port n'est pas compatible avec le port trunk. Seule la carte **Intel PCI-Express double ports** peut servir de ports Trunks

6.4 Scénario 4 :

6.4.1 Réponses aux requêtes Pings

- Symptôme

Une fois mes VMs Windows Server 2008 installée, il m'était impossible de les pinger, les deux serveurs ne répondaient pas au Pings.

- Solution

Pour résoudre ce problème, j'ai dû activer dans le pare-feu de Windows la règle entrante autorisant la réponse aux requêtes Ping : File and printer (Echo Request – ICMPv4-In).

7 Conclusion

A travers ce projet de Bachelor qui s'est déroulé sur 8 semaines, j'ai pu implémenter deux solutions de management d'infrastructures virtualisées. La mise en œuvre de ces deux projets m'a permis de mieux comprendre les enjeux d'une telle infrastructure centralisée, tant au niveau du cloisonnement des VMs, que de la haute disponibilité des ressources, ainsi que de la supervision des différentes entités.

La technologie oVirt m'a fait découvrir les différents éléments nécessaires à la création d'un environnement virtualisé contrôlé depuis un seul point. Même si cette solution ne remplissait pas toutes les conditions pour mener à bien mon projet, elle utilise dans les grandes lignes les mêmes éléments que OpenNebula, soit un manager, des hyperviseurs et une unité de stockage. J'ai trouvé que le déploiement c'est fait de manière générale assez aisément grâce à oVirt-Engine-Setup, qui automatise le processus d'installation du manger. Une fois configurée, l'interface web rappelle l'outil vSphere chez VMware. Le déploiement des hyperviseurs KVM se fait via la distribution oVirt-Node qui pourrait s'apparenter à ESXi, du fait que lorsque l'installation est terminée, nous obtenons une interface TUI pour le configurer. Ovirt m'a également fait découvrir la distribution Fedora que je ne connaissais pas avant de commencer le travail de Bachelor. J'ai passé environ deux semaines à analyser et configurer cette technologie.

La technologie OpenNebula avec l'intégration d'Open vSwitch au sein des différents hyperviseurs a été le point central de mon travail de Bachelor. L'installation de OpenNebula n'a pas été aussi simple que celle d'oVirt, car de nombreux programmes liés à la compilation de cette solution ont été nécessaires. De plus, il faut également porter une attention toute particulière à recréer les mêmes comptes utilisateurs entre les différentes entités de l'infrastructure (hyperviseurs, manager). Grâce à OpenNebula, j'ai pu acquérir des connaissances dans l'utilisation d'un Ubuntu Server via CLI. J'ai pu aussi découvrir l'installation du module KVM, et ainsi, mieux comprendre les différents éléments intervenant dans la virtualisation. J'ai pu voir que même au sein d'une infrastructure virtualisée nous pouvons gérer nos réseaux comme au niveau d'un véritable réseau physique, avec l'intégration de bridges gérés par Open vSwitch intégrant les VLANs.

L'étude de ces deux projets m'a également permis d'approfondir mes connaissances du monde Open-Source, qui est composé de programmes rivalisant avec les grands noms de la virtualisations tel que VMware. De grands géants de l'informatique utilisent OpenNebula[20], comme IBM ou DELL. L'avantage de ces solutions Open-Source est que des centaines de développeurs dévoués contribuent à leur évolution.
Actuellement, de plus en plus d'entreprises externalisent leurs ressources informatiques, pour les transférer dans le Cloud. Ces infrastructures centralisées et dématérialisées, de part leurs emplacements délocalisés, soulèvent de réelles questions tant au niveau de l'identité numérique, que de la confidentialité des données. Il reste encore beaucoup de travail dans la gestion de ces gigantesques Datacenters, afin de garantir au client que ses données sont en sureté, et qu'elles ne seront pas divulguées à autrui.

[20] http://opennebula.org/users:users

8 Déroulement du projet

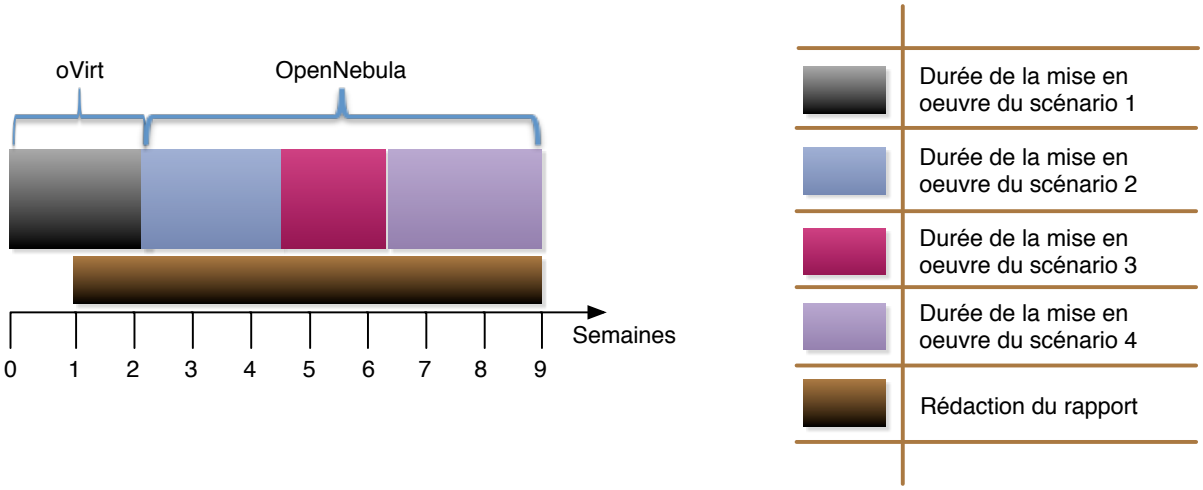


Figure 17 : Déroulement du Travail de Bachelor

9 Liens & références

- Installation d'oVirt :
  http://www.ovirt.org/w/images/a/a9/OVirt-3.0-Installation_Guide-en-US.pdf
  Ce document résume clairement les différentes étapes de mise en place la solution oVirt
- Déploiement d'Active Directory sur Windows Server 2008 :
  http://syskb.com/installer-active-directory-sur-windows-server-2008/
  Ce site internet explique les différentes étapes nécessaires à la mise en place d'un Active Directory
- Pour le déploiement de OpenNebula j'ai consulté tout d'abord la documentation fournie par OpenNebula : http://www.opennebula.org/documentation:rel3.4
  Il résume les différents éléments utilisés dans OpenNebula, il manque de précision au niveau du processus d'installation
- Pour palier à ce problème, j'ai découvert deux documents fournis par http://cloudblab.files.wordpress.com:

  Le premier m'a permis de déployer le manager **Page 2 à 8**:
  http://cloudblab.files.wordpress.com/2012/05/opennebula-r-3-4-and-vmware-esxi5-0-using-sharedvmware-and-ssh-transfer-drivers1.pdf
  Le deuxième m'a permis de déployer KVM sur l'hyperviseur **Page 5**:
  http://cloudblab.files.wordpress.com/2012/04/builing-a-hybrid-cloud-with-opennebula.pdf
  J'ai modifié tous les chemins ***/srv/cloud/one*** par ***/var/lib/one***, afin que cela coïncide avec l'installation du manager
- Pour l'installation de Open vSwitch, j'ai suivi le document fournis par Open vSwitch afin d'installer Open vSwitch sur l'hyperviseur : http://openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=INSTALL.Linux;hb=HEAD
- Compatibilité entre Open vSwitch et KVM, j'ai suivi le document :
  http://openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=INSTALL.KVM;hb=HEAD
- Pour déployer le scénario n°3, j'ai suivi l'exemple fourni par Open vSwitch : http://openvswitch.org/support/config-cookbooks/vlan-configuration-cookbook/

A Annexes

A Annexes 57
A.1 Revirement de situation 57
A.2 Configuration de Base du NAS 57
A.3 Listes des différents PC utilisés sous oVirt 59
A.4 Installation du Manager OpenNebula sur Ubuntu 11.10x64 61
A.5 Installation de l'hyperviseur avec le module KVM 62
A.6 Installation de Open vSwitch sur un hyperviseur KVM 64
A.7 Templates des VMs déployer par OpenNebula 68

A.1 Revirement de situation

Après une discussion avec Mr.Pasche le 24 Avril, la solution OpenStack initialement mentionnée dans le cahier des charges ne semble pas vraiment adaptée à la société SmartBee. OpenStack est utilisé pour gérer une infrastructure de Cloud basé sur plusieurs centaines de serveurs physiques. Cette solution est lourde à mettre en place et à maintenir à niveau. Pour la société SmartBee, une solution d'infrastructure virtualisée serait bien mieux adaptée. La distribution « oVirt » est une solution ressemblant énormément à vCenter, mais qui a l'avantage d'être open source. Pour créer un Cloud privé de petite taille la solution OpenNebula sera également mise en place.

A.2 Configuration de Base du NAS

A.2.1 Formatage des disques

Durant la première mise sous tension, le système nous propose la façon de formater nos disques, soit en RAID, ou soit en mode Single (Utilisation d'un seul disque dans la baie).

Pour faire défiler les choix, il faut utiliser le bouton « SELECT » puis le bouton « ENTER » permet de valider son choix. Une fois le choix validé, il faut attendre que le NAS formate les disques.

Indique quel disque est activité

A.2.2 Configuration Réseau

Une fois le formatage terminé, il faut brancher le câble réseau dans le port Ethernet Eth0.

Pour configurer les paramètres IP, le plus simple est d'obtenir une adresse via un serveur DHCP, car la configuration statique est longue et fastidieuse, du fait qu'il ne possède que deux boutons physiques. Grâce à son écran frontal, nous verrons instantanément quelle adresse le serveur lui a attribué.

Pour rentrer dans le menu de configuration, il faut appuyer pendant deux secondes sur « ENTER ».
Puis avec ce même bouton, rentrer dans le menu TCP/IP.
Faire défiler les options avec « SELECT » jusqu'à arriver au choix Network Settings.
Sélectionner ce menu, et choisir DHCP. Le QNAP obtient une adresse via le serveur DHCP, et va afficher l'adresse obtenue sur son écran.

A.2.3 Administration du QNAP pour l'installation du serveur NFS

Une fois que les paramétrages réseaux, sont effectués, nous allons pouvoir nous connecter au serveur web du QNAP, pour démarrer le service de partage NFS, ainsi que de lui attribuer une adresse IP statique.

A.3 Listes des différents PC utilisés sous oVirt

Le Manager :

- PC : A27
- Nom : engine.cirrus
- Interfaces réseaux : 1 carte Ethernet travaillant au gigabit (Eth0)
- Adresse IP Eth0 : 10.1.2.90
- OS : Fedora16
- Utilisateur : user
- Mot de passe : rootroot

L'hyperviseur :

- PC : A24
- Nom : node.cirrus
- Interfaces réseaux : 2 cartes Ethernets travaillant au gigabit (Eth2, Eth3)
- Adresse IP Eth3 : 10.1.2.91
- Adresse IP Eth2 : 192.168.1.5
- OS : Distribution oVirt-Node-2.3.0-1.0.fc16 basé sur Fedora16[21]
- Utilisateur : admin
- Mot de passe : root

Cible iSCSI:

- PC : A6
- Nom : storage
- Interfaces réseaux : 1 carte Ethernet travaillant au gigabit (p1p2)

[21] http://ovirt.org/releases/stable/binary/ovirt-node-iso-2.3.0-1.0.fc16.iso

- Adresse IP p1p2 : 192.168.1.4
- OS : Fedora16
- Utilisateur : user
- Mot de passe : rootroot

Serveur DNS:

- PC : A26
- Nom : dns.cirrus
- Interfaces réseaux : 1 carte Ethernet travaillant au gigabit (Eth0)
- Adresse IP Eth0 : 10.1.2.92
- OS : Fedora16
- Utilisateur : user
- Mot de passe : rootroot

VM Windows Server 2008:

- PC : VM-Windows-Server
- Nom : SRV-2008
- Interfaces réseaux : 1 carte Ethernet travaillant au gigabit
- Adresse IP Eth0 : 10.1.2.94
- OS : Windows Server 2008 Standard Edition x86
- Utilisateur : Administrator
- Mot de passe : R00tr00t

Client Manager:

- PC : Dell D1
- Nom : dellD1
- Interfaces réseaux : 1 carte Ethernet travaillant au gigabit
- Adresse IP Eth0 : Attribué par DHCP
- OS : Windows 7 + IE9
- Utilisateur : Albert
- Mot de passe : admin
- Particularité : Gère le domaine VM.lan

Membre du domaine VM.lan:

- PC : Dell D8
- Nom : DellD8
- Interfaces réseaux : 1 carte Ethernet travaillant au gigabit
- Adresse IP Eth0 : 10.1.2.95
- OS : Windows Vista
- Utilisateur : Albert
- Mot de passe : admin

A.4 Installation du Manager OpenNebula sur Ubuntu 11.10x64

Voici les commandes à exécuter pour déployer OpenNebula sur le manager :

A.4.1 Création de l'utilisateur et des dossiers

sudo groupadd -g 10000 cloud
sudo useradd -u 10000 -m oneadmin -d /var/lib/one -s /bin/bash -g cloud
sudo passwd oneadmin --- Mettre oneadmin comme mot de passe
sudo chown -R oneadmin:cloud /var/lib/one
sudo adduser oneadmin admin

A.4.2 Téléchargement du package OpenNebula et décompression

su –l oneadmin
Télécharger l'archive opennebula-3.4.1.tar.gz sur http://downloads.opennebula.org/
tar xzf opennebula-3.4.1.tar.gz
cd opennebula-3.4.1/

A.4.3 Installation des programmes requis pour l'installation

sudo apt-get install libcurl3 libmysqlclient16 libruby1.8 libsqlite3-ruby libsqlite3-ruby1.8 mysql-common ruby
sudo apt-get install libsqlite3-dev libxmlrpc-c3-dev g++ libssl-dev ruby-dev
sudo apt-get install libopenssl-ruby
sudo apt-get install pkg-config
sudo apt-get install libxml2-dev libmysqlclient-dev libmysql++-dev libsqlite3-ruby libexpat1-dev
sudo apt-get install libc6 libgcc1 libpassword-ruby libsequel-ruby libsqlite3-0 libssl0.9.8 libstdc++6 libxml2 libxmlrpc-c3-0 libxmlrpc-core-c3-0
sudo apt-get install ruby rubygems libmysql-ruby libsqlite3-ruby libamazonec2-ruby
sudo apt-get install rake rubygems libxml-parser-ruby1.8 libxslt1-dev genisoimage scons
sudo gem install nokogiri rake xmlparser
sudo apt-get install opennebula-common
sudo apt-get install mysql-server
Au moment de l'utilitaire d'installation de MySQL server mettre comme mot de passe général « root »

A.4.4 Création de la base de donnée MySQL

mysql –u root -p
CREATE USER 'oneadmin'@'localhost' IDENTIFIED BY 'oneadmin';
CREATE DATABASE opennebula;
GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin' IDENTIFIED BY 'oneadmin';
quit;

Exécuter la commande
scons sqlite=no mysql=yes ---Compilation de OpenNebula avec l'intégration de MySQL

A.4.5 Installation de OpenNebula 3.4.1

./install.sh -u oneadmin -g cloud -d /var/lib/one

A.4.6 Création des variables d'environnements

vi ~/.bash_profile ---Création du fichier regroupant les variables d'environnement

Contenu du fichier :
export ONE_LOCATION=/srv/cloud/one
export ONE_AUTH=$ONE_LOCATION/.one/one_auth
export ONE_XMLRPC=http://localhost:2633/RPC2
export
PATH=$ONE_LOCATION/bin:/usr/local/bin:/var/lib/gems/1.8/bin/:/var/lib/gems/1.8/:$PATH

Application des variables :
source ~/.bash_profile

Création du fichier possédant les mot de passe global :
mkdir ~/.one
echo "oneadminoneadmin" > ~/.one/one_auth

A.4.7 Edition du fichier oned.conf

Pour réaliser le scénario 1 :

Commenter la partie « sqlite »
Décommenter la partie MYSQL

Décommenter les Drivers :

IM_MAD = [
name = "im_kvm",
executable = "one_im_ssh",
arguments = "-r 0 -t 15 kvm"]

VM_MAD = [
name = "vmm_kvm",
executable = "one_vmm_exec",
arguments = "-t 15 -r 0 kvm",
default = "vmm_exec/vmm_exec_kvm.conf",
type = "kvm"]

Donner le chemin du datastore de l'Hyperviseur :
PATH=/var/lib/var/datastores/

A.5 Installation de l'hyperviseur avec le module KVM

Voici les commandes à exécuter pour KVM sur l'hyperviseur :

Décommenter dans ***/etc/sysctl.conf*** :
net.ipv4.ip_forward = 1 - --pour autoriser le nat entre les VMs

sudo apt-get install nfs-common ---Installation du client NFS

sudo mkdir -p /var/lib/one/var/datastores/0 ---Création du dossier Datastore qui va pointer sur le dossier NFS du QNAP

A.5.1 Création de l'utilisateur et des dossiers

sudo groupadd -g 10000 cloud
sudo useradd -u 10000 -g cloud -m oneadmin -s /bin/bash
sudo usermod -d /var/lib/one oneadmin
sudo usermod -a -G cloud,admin oneadmin
sudo passwd oneadmin ---Mettre comme mot de passe **oneadmin**
sudo chown oneadmin:cloud /var/lib/one

A.5.2 Montage du dossier NFS

Editer le fichier ***/etc/fstab*** pour indiquer que le dossier ***nfs-opennebula/test*** du QNAP sera monté sur le dossier ***/var/lib/one/datastores/0/*** de l'hyperviseur :
Adresse_IP_QNAP:/nfs-opennebula/test /var/lib/one/var/datastores/0 nfs rw 0 0
sudo mount /var/lib/one/var/datastores/0

A.5.3 Configuration de KVM

sudo apt-get install ubuntu-vm-builder ruby
sudo chown :cloud /var/run/libvirt/libvirt-sock

Editer le fichier **libvirtd.conf** et **qemu.conf** pour autoriser le manager à communiquer avec via la socket libvirt

vi /etc/libvirt/libvirtd.conf

listen_tcp = 1
unix_sock_group = "cloud"
unix_sock_rw_perms = « 0777 »

vi /etc/libvirt/qemu.conf

vnc_listen = "0.0.0.0"
user = oneadmin
group = cloud
dynamic_ownership = 0

sudo adduser oneadmin kvm
sudo /etc/init.d/libvirt-bin restart --- Rédemarrer libvirt
sudo libvirtd –d --- Demarrer libvirt-socket

A.5.4 Connexion SSH sans mot de passe

- Manipulation à faire sur le manager :

1. Génération de la clé publique grâce à l'algorithme RSA :

su –l oneadmin
ssh-keygen
Appuyer 3 fois sur ENTER

2. Modification des fichiers de configuration du client SSH

cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
vi ~/.ssh/config
Host *
StrictHostKeyChecking no

3. Envoie de la clé publique au serveur SSH (Hyperviseur : host.cumulus)

Envoie de la clé SSH sur l'hyperviseur pour éviter de rentrer le mot de passe :
ssh-copy-id -i ~/.ssh/id_rsa.pub oneadmin@host.cumulus
Rentrer le mot de passe de l'utilisateur oneadmin, ici oneadmin

4. Tester la connexion sans mot de passe :

ssh oneadmin@host.cumulus

A.6 Installation de Open vSwitch sur un hyperviseur KVM

A.6.1 Les programmes requis

sudo apt-get install python-simplejson python-qt4 python-zope.interface python-twisted-conch automake autoconf gcc uml-utilities pkg-config libtool

A.6.2 Création du dossier d'installation et obtention de l'archive

mkdir ~/openvswitch_install
cd openvswitch_install
wget http://openvswitch.org/releases/openvswitch-1.4.1.tar.gz
tar xzf openvsitch-1.4.1.tar.gz
cd openvsitch-1.4.1/

A.6.3 Exécution du script d'installtion et compilation

./boot.sh
./configure –with-linux=/lib/modules/**'uname –r'**/build
« 'uname –r' » indique la version du Kernel pour ma part : 3.0.0-12-server
make

Se mettre en root
sudo –s
make install

A.6.4 Chargement des modules dans le noyau

insmod datapath/linux/openvswith_mod.ko
rmmod bridge (décharcher le module « bridge » de base du noyau)
insmod datapath/linux/brcompat_mod.ko (comptatibilité avec « bridge »)

A.6.5 Création de la base donnée

mkdir -p /usr/local/etc/openvswitch
ovsdb-tool create /usr/local/etc/openvswitch/conf.db vswitchd/vswitch.ovsschema

ovsdb-server /usr/local/etc/openvswitch/conf.db –-
remote=punix:/usr/local/var/run/openvswitch/db.sock –-
remote=db:Open_vSwitch,manager_options –-pidfile –-detach

A.6.6 Compatibilité avec OpenNebula

Modifier le fichier ***/etc/sudoers***
Rajouter la ligne :
%cloud ALL=NOPASSWD: /usr/local/bin/ovs-vsctl

A.6.7 Démarrage de Open vSwitch

sudo ovs-brcompatd --pidfile --detach
sudo sudo ovs-vsctl –-no-wait init
sudo ovs-vswitchd –pidfile –detach

A.6.8 Scripts de démarrage

Comme la configuration n'est pas persistante, il faut exécuter le script :
start_deamon_ovs.sh, à démarrage de l'hyperviseur.

Détail du script :

```
! /bin/bash
echo "--------------------------"
echo "--Demarrage Open Vswitch--"
echo "--------------------------"

PATH=/var/lib/one/openvswitch_install/openvswitch-1.4.1/datapath/linux

/sbin/insmod $PATH/openvswitch_mod.ko

/usr/local/sbin/ovsdb-server      /usr/local/etc/openvswitch/conf.db      --
remote=punix:/usr/local/var/run/openvswitch/db.sock                       --
remote=db:Open_vSwitch,manager_options --pidfile --detach

/usr/local/bin/ovs-vsctl --no-wait init
/usr/local/sbin/ovs-vswitchd --pidfile --detach

/sbin/rmmod bridge
/sbin/insmod $PATH/brcompat_mod.ko
/bin/sleep 2
/usr/local/sbin/ovs-brcompatd    --appctl=/usr/local/sbin/ovs-vswitchd    --
vsctl=/usr/local/bin/ovs-vsctl --pidfile --detach
```

A.6.9 Script d'attribution des VLANs

Pour attribuer les attributs aux différentes interfaces, éxécuter le script :
vlan.sh nom_du_bridge nom_interface attribut ….

Détail du script :

```
! /bin/bash
##############################
#####README##################
## Ce script ajoute les interfaces physiques
## et virtuelles au bridge géré par Open vSwitch en attribuant un tag VLAN
à chaque interface

## 1er argument : Nom du bridge gérer par Open vSwitch
## Pour configurer une interface
## on rentre 2 arguments : un pour le nom de la carte et le 2e pour
l'attribut
```

```
## EXP:
## sudo vlan.sh br1 vnet0 tag=10 eth3 trunks=10,11
## ###################################

boolean=0

index=0

for arg in $@
do
tab_arg[index]=$arg
echo ${tab_arg[$index]}
((index++))
done
index=0

echo "Le bridge s'appelle:  "${tab_arg[0]}

for ((var2=1; var2<${#tab_arg[*]} ;var2++));
do

if [ "$boolean" = "0" ]; then
      echo "Carte Ethernet:  "${tab_arg[$var2]}
      echo "La fonction de la carte:  "${tab_arg[$var2+1]}
      boolean=1
      ovs-vsctl -- --if-exists del-port ${tab_arg[$var2]}
      ovs-vsctl        add-port        ${tab_arg[0]}        ${tab_arg[$var2]}
${tab_arg[$var2+1]}
else
      boolean=0
fi
done

echo "------------------------------"
echo "--Topologie du" ${tab_arg[0]}"------------"
echo "------------------------------"
ovs-vsctl show
```

A.6.10 Configuration du switch Cisco Catalyst 2900

```
Current configuration : 1579 bytes
!
version 12.1
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname CISCO_02
!
enable password labo
!
username cisco password 0 labo
ip subnet-zero
!
vtp mode transparent
!
spanning-tree mode pvst
no spanning-tree optimize bpdu transmission
spanning-tree extend system-id
```

```
!
vlan 10-11
!
vlan 66
 name Users
vlan 99
 name Management
!
interface FastEthernet0/1
 switchport trunk allowed vlan 10,11
 switchport mode trunk
!
interface FastEthernet0/2
!
interface FastEthernet0/3
!
interface FastEthernet0/4
!
interface FastEthernet0/5
 switchport access vlan 10
!
interface FastEthernet0/6
 switchport access vlan 10
!
interface FastEthernet0/7
 switchport access vlan 10
!
interface FastEthernet0/8
 switchport access vlan 10
!
interface FastEthernet0/9
!
interface FastEthernet0/10
!
interface FastEthernet0/11
 switchport access vlan 11
!
interface FastEthernet0/12
 switchport access vlan 11
!
interface FastEthernet0/13
 switchport access vlan 11
!
interface FastEthernet0/14
 switchport access vlan 11
!
interface FastEthernet0/24
!
interface Vlan1
 ip address 10.1.0.31 255.255.0.0
 no ip route-cache
!
ip default-gateway 10.1.0.1
ip http server
!
line con 0
line vty 0 4
 password labo
 login
line vty 5 15
```

```
 password labo
 login
!
!
end
```

A.7 Templates des VMs déployer par OpenNebula

A.7.1 Scénario 2 et 3

Ubuntu Desktop 11.04 x86 :

```
CPU="1"
DISK=[
 BUS="ide",
 DRIVER="raw",
 IMAGE="HDD",
 IMAGE_UNAME="oneadmin",
 TARGET="hda" ]
DISK=[
 BUS="ide",
 DRIVER="file:",
 IMAGE="CD_Ubuntu_Desktop",
 IMAGE_UNAME="oneadmin",
 TARGET="sdb" ]
GRAPHICS=[
 LISTEN="0.0.0.0",
 TYPE="vnc" ]
INPUT=[
 BUS="usb",
 TYPE="tablet" ]
MEMORY="512"
NAME="Ubuntu_10.04"
NIC=[
 MODEL="e1000",
 NETWORK="openvswitch",
 NETWORK_UNAME="oneadmin" ]
OS=[
 ARCH="i686",
 BOOT="hd" ]
RAW=[
 TYPE="kvm" ]
TEMPLATE_ID="26"
```

A.7.2 Scénario 4

Template VM Client 2 :

```
CPU="1"
DISK=[
 BUS="ide",
 DRIVER="raw",
 IMAGE="OS_Win2008_Client2",
```

```
  IMAGE_UNAME="oneadmin",
  TARGET="hda" ]
FEATURES=[
  ACPI="yes" ]
GRAPHICS=[
  KEYMAP="fr-ch",
  LISTEN="0.0.0.0",
  TYPE="vnc" ]
INPUT=[
  BUS="usb",
  TYPE="tablet" ]
MEMORY="2048"
NAME="VM_Temp_Client2"
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_2_ovs",
  NETWORK_UNAME="oneadmin" ]
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_1_ovs",
  NETWORK_UNAME="oneadmin" ]
OS=[
  ARCH="i686",
  BOOT="hd" ]
RAW=[
  TYPE="kvm" ]
TEMPLATE_ID="37"
```

Template VM Client 3 :

```
CPU="1"
DISK=[
  BUS="ide",
  DRIVER="raw",
  IMAGE="OS_Win2008_Client3",
  IMAGE_UNAME="oneadmin",
  TARGET="hda" ]
FEATURES=[
  ACPI="yes" ]
GRAPHICS=[
  LISTEN="0.0.0.0",
  TYPE="vnc" ]
INPUT=[
  BUS="usb",
  TYPE="tablet" ]
MEMORY="2048"
NAME="VM_Temp_Client3"
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_3_ovs",
```

```
  NETWORK_UNAME="oneadmin" ]
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_1_ovs",
  NETWORK_UNAME="oneadmin" ]
OS=[
  ARCH="i686",
  BOOT="hd" ]
RAW=[
  TYPE="kvm" ]
TEMPLATE_ID="38"
```

Template VM de Supervision Nagios :

```
CPU="1"
DISK=[
  BUS="ide",
  DRIVER="raw",
  IMAGE="Full_HDD4",
  IMAGE_UNAME="oneadmin",
  TARGET="hda" ]
FEATURES=[
  ACPI="yes" ]
GRAPHICS=[
  KEYMAP="fr-ch",
  LISTEN="0.0.0.0",
  TYPE="vnc" ]
INPUT=[
  BUS="usb",
  TYPE="tablet" ]
MEMORY="512"
NAME="Full_Ubuntu"
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_1_ovs",
  NETWORK_UNAME="oneadmin" ]
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_2_ovs",
  NETWORK_UNAME="oneadmin" ]
NIC=[
  MODEL="e1000",
  NETWORK="VLAN_3_ovs",
  NETWORK_UNAME="oneadmin" ]
OS=[
  ARCH="i686",
  BOOT="hd" ]
RAW=[
  TYPE="kvm" ]
TEMPLATE_ID="28"
```