Master of Science HES-SO in
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication (TIC)

Agility in Open Source virtualized infrastructure

Fait par

Benoît Chalut

Sous la direction de

Prof. Gérald Litzistorf

Dans le laboratoire de Transmission de Données à la **H**aute **E**cole du **P**aysage d'**I**ngénieurie et d'**A**rchitecture

Expert externe : Mr. Sébastien Pasche, Security and System Engineer à leShop.ch

Genève, HES-SO//Master, 17/06/2013

1 Enoncé

MSE | MASTER OF SCIENCE IN ENGINEERING

Agility in Open Source virtualized infrastructure

Responsable	Litzistorf Gerald
MRU	TIC / hepia
Profils technologiques concernés	TIC / Réseaux d'entreprises et sécurité IT
Entreprise	LeShop.ch
Résumé	Une infrastructure de virtualisation comprend diverses briques (hyperviseurs, machines virtuelles, systèmes de stockage, réseau, ...) appelées à devoir évoluer suite à de nouveaux besoins opérationnels, à des mises à jour, ? Cette étude vise à définir un processus de gestion dans un environnement Linux Open Source basé sur les bonnes pratiques garantissant une grande souplesse d'utilisation
Cahier des charges	Partie 1 : Live-CD Etude des outils Fedora (cobler, livecd-tool, lazaar, ?) pour construire des live-cd Réalisation d'un live-cd contenant un environnement de virtualisation lié à des machines virtuelles se trouvant sur un système de stockage nfs ou iSCSI Partie 2 : Gestion centralisée Etudier les possibilités pour une gestion centralisée des configurations dépendantes de chaque hyperviseur (VM à démarrer, configuration réseau, ?) via un dépôt central GIT et des rpm Etablir avec l'entreprise mandante http://www.leshop.ch les étapes du projet. Définir les priorités et les tests unitaires Identifier les points pouvant faire l'objet d'une étude complémentaire dans la thèse de Master
Connaissances préalables	Module de base ITSec
Mots-clés	TIC Sécurité; TIC VIrtualisation

2 Avant-propos

2.1 Remerciements

Je remercie :

- Monsieur Gérald Litzistorf en sa qualité de professeur responsable pour m'avoir suivi tout au long de ce projet, ainsi que la précieuse aide qu'il m'a apportée dans la rédaction de ce document.
- Monsieur Sébastien Pasche en sa qualité d'expert pour son encadrement et ses précieux conseils.
- Monsieur Khaled Basbous en sa qualité d'assistant de laboratoire et camarade de classe qui a su m'éclairer sur de nombreux problèmes techniques.

3 Analyse:

3.1 Introduction

Depuis quelques années avec l'apparition du Cloud Computing, la gestion des infrastructures virtualisées est devenue un élément central à la bonne marche des Data-Centers. Ces centres sont de plus en plus importants et hébergent de plus en plus de serveurs physiques et virtuels. A l'heure actuelle, il existe de nombreuses solutions Open-Source qui permettent le management des VMs au sein des hyperviseurs tels que OpenNebula ou oVirt. Ces outils se focalisent essentiellement sur le déploiement automatique des VMs, en leurs indiquant un hyperviseur hôte ou en utilisant la Live-Migration pour rendre ces VMs toujours disponibles. Mais ces managers n'intègrent pas le déploiement automatique des éléments centraux de notre Data-Center qui sont les hyperviseurs. Comme la technologie KVM utilisée pour la virtualisation sous Linux est poussée par une grande communauté open-source, ces modules sont alors en constante évolution.

3.2 Périmètre du projet

Mon projet consiste en l'élaboration d'une infrastructure de déploiement automatique d'hyperviseurs, permettant d'avoir une certaine agilité dans la mise à jour de ces éléments. Il va faire intervenir plusieurs outils tels que la création de LiveCD ainsi que le système de boot via PXE. Toute l'infrastructure sera basée sur Fedora 18 qui est la dernière version de cette distribution Linux. Le but final est de déployer un hyperviseur contenant tous les packages utiles à la virtualisation, s'administrant grâce au protocole ssh via CLI donc sans serveur Xorg. Dans ce travail, je vais essentiellement explorer la fonctionnalité du projet, et ne vais par conséquent pas aborder les problèmes liés à la sécurité.

3.3 Composition d'un hyperviseur

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 seul 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).

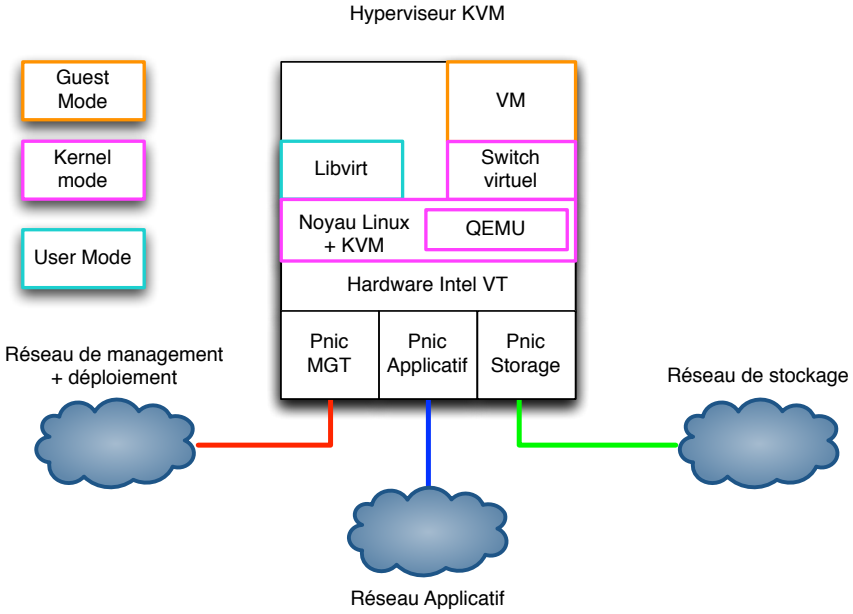


Figure 1 : Virtualisation sous Linux

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

- **KVM** arbitre les accès des VM aux processeurs et à la RAM physiques, il est directement intégré au Noyau GNU/Linux.
- **QEMU** sert de solution d'émulation, donc va simuler à la VM un matériel virtualisé (Exp : Processeur virtualisé, Mémoire RAM virtuelle,…).
- Le module **libvirt** envoie des instructions à KVM pour instancier une VM, la stopper ou la redémarrer.
- Carte réseau physique (**Pnic**) de **Management** : Connecter au réseau de management pour le transfert de la distribution, et commander le déploiement des VM.
- **Pnic applicative** : Connecter au réseau applicatif, pour que la VM communique avec le monde extérieur.
- **Pnic de stockage** : Connecter au réseau de stockage, qui héberge les données des VM.
- Le **Switch virtuel** : Fait transiter le flux de données de l'interface réseau virtuel de la VM vers la Pnic Applicative.

Les modes d'exécution des processus Linux[1] :

- Kernel mode[2] : Le processus a accès à l'ensemble du Noyau Linux, c'est à dire qu'il a accès à l'ensemble des instructions et des adresses mémoires, il a les privilèges root.
- User mode[3] : C'est un mode avec un jeu d'instruction et des doits restreints que la majorité de processus utilisent.
- Guest mode : Nouveau mode rendu possible grâce à l'extension VT-x Intel. Il va permettre à la VM de lancer des interruptions sur le matériel virtualisé. La VM est vue comme un processus depuis le Noyau Linux de l'hyperviseur.

[1] http://www.linuxjournal.com/magazine/linux-kvm-learning-tool
[2] http://www.linfo.org/kernel_mode.html
[3] http://www.linfo.org/kernel_mode.html

3.4 Etat de l'art

3.4.1 Qu'est ce qu'une distribution

Une distribution Linux est un ensemble d'éléments regroupant :

- Le Noyau GNU/Linux : Le cœur du système qui fait le lien entre le matériel et les applications, en faisant des appels systèmes en utilisant les différents modes comme expliqués ci-dessus. Il est vu comme un ordonnanceur afin de gérer les différents processus et la gestion de la mémoire.
- Des paquets ou packages: Fichiers compressés permettant d'installer une application contenant :
  - Les fichiers exécutables ou le code source
  - Les fichiers de configuration
  - La documentation de l'application
  - La liste des dépendances, qui va indiquer quelles sont les bibliothèques ou les applications pré-requises
- Un fichier Kickstart[4] : Fichier texte qui va conditionner l'installation de la distribution, constitué de quatre parties (voir un exemple de Fichier Kickstart en Annexe B):
  - 1ère partie : Configuration du noyau, par exemple pour définir la langue du système ou la configuration du firewall
  - 2ème partie : Partie « %pre » installation, où l'on peut insérer un script qui va s'exécuter avant le démarrage de l'installation pour parcourir par exemple les différentes unités de stockages
  - 3ème partie : La liste des packages à installer afin de créer une distribution contenant uniquement les applications qui nous intéressent
  - 4ème partie : Partie « %post » installation, où l'on peut insérer un script s'exécutant une fois le noyau et les paquets installés. Par exemple on configure nos montages NFS ou la configuration de la connexion SSH

3.4.2 L'entreposage des paquets

Afin de créer une distribution personnalisée, nous devons aller puiser en local ou sur internet les packages à greffer. Cet entrepôt s'appelle un « dépôt » ou « repository ». La présence d'un dépôt local va permettre de diminuer le temps de chargement des paquets. Durant le déploiement, le chargement des paquets peut se faire via le protocole HTTP.

3.4.3 Composition et installation d'une distribution officielle

3.4.3.1 2 types d'utilisation:

Nous avons deux possibilités de déployer une distribution Linux, soit en utilisant un LiveCD ou soit en l'installant via un DVD.

D'après Fedora : « Le live CD a pour but de rendre disponible chaque version de Fedora en "Live" (sans devoir l'installer). »[5] :

Le LiveCD va copier une distribution complète en RAM, ce qui ne nécessite aucune installation. Tandis qu'avec un DVD, nous copions le FileSystem allégé sur le disque dur, puis

[4] http://fedoraproject.org/wiki/Anaconda/Kickstart

[5] http://doc.fedora-fr.org/wiki/Fedora_Live_CD#Objectifs_et_fonctionnalit.C3.A9s

nous installons les différents paquets en fonctions d'un fichier Kickstart ou à l'aide du GUI Fedora Installer. (Voir ci-dessous le détail de l'installation)
Le FileSystem[6] : La partition principale de notre système d'exploitation contenant la structure des dossiers où vont se loger les différents fichiers, dossiers des applications possédant leurs permissions.

3.4.3.2 Téléchargement des Fichiers

Lien vers le téléchargement du LiveCD :
http://mirror2.hs-esslingen.de/fedora/linux/releases/18/Live/x86_64/Fedora-18-x86_64-Live-Desktop.iso

Lien vers le téléchargement du DVD :
http://mirror2.hs-esslingen.de/fedora/linux/releases/18/Fedora/x86_64/iso/Fedora-18-x86_64-DVD.iso
Les deux fichiers sont des fichiers ISO. Ils correspondent à l'image virtuelle d'un CD-ROM ou DVD-ROM physiques.

3.4.3.3 Arborescence des ISO du le LiveCD et du DVD

Afin de comprendre comment s'effectue une installation du DVD et du LiveCD, j'ai analysé l'arborescence des 2 ISO.
Taille ISO Fedora-DVD : 4.6 Go
Taille ISO Fedora-LiveCD : 0.9Go

- **Contenu ISO Fedora 18 Live-CD Officiel:**

http://fedoraproject.org/wiki/LiveOS_image

Pour l'analyser, j'ai monté l'ISO grâce à la commande Mount :

```
Documents/Fedora-18-x86_64-Live-Desktop/
!(Mount)
├──── [BOOT]
│     └──── Bootable_NoEmulation.img
├──── EFI
│     └──── BOOT
├──── GPL
├──── isolinux
│     ├──── boot.cat
│     ├──── efiboot.img
│     ├──── initrd0.img
│     ├──── isolinux.bin
│     ├──── isolinux.cfg
│     ├──── macboot.img
│     ├──── memtest
│     ├──── vesamenu.c32
│     └──── vmlinuz0
└──── LiveOS
      ├──── livecd-iso-to-disk
      ├──── osmin.img
      ├──── squashfs.img
```

[6] http://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/s1-filesystem-fhs.html : Début du §2.2

- **Contenu ISO Fedora 18 DVD Officiel:**

```
Documents/Fedora-18-x86_64-DVD/
├──── [BOOT]
│     └──── Bootable_NoEmulation.img
├──── EFI
│     ├──── BOOT
│     └──── TRANS.TBL
├──── images
│     ├──── efiboot.img
│     ├──── macboot.img
│     ├──── pxeboot
│     └──── TRANS.TBL
├──── isolinux
│     ├──── boot.cat
│     ├──── boot.msg
│     ├──── grub.conf
│     ├──── initrd.img
│     ├──── isolinux.bin
│     ├──── isolinux.cfg
│     ├──── memtest
│     ├──── splash.png
│     ├──── TRANS.TBL
│     ├──── upgrade.img
│     ├──── vesamenu.c32
│     └──── vmlinuz
├──── LiveOS
│     ├──── squashfs.img
├──── Packages
├──── repodata
```

Les fichiers en gras sont les fichiers de bases de l'installation.

3.4.3.4 Analyse du contenu du dossier ISOLINUX

Dossier **ISOLINUX** : Ce dossier est commun aux 2 fichiers ISO, il contient l'ensemble des Boot Loader qui vont permettre de monter un CD-ROM afin de booter dessus.

- vmlinuz[7] : Image compressée du Noyau Linux, qui sera chargée dans la RAM afin d'être exécutée.
- initrd.img : Initial RAM-Disk est un FileSystem compressé appelé Dracut[8]. Il contient des modules (drivers) qui vont être ajoutés au Noyau tels que le pilote de partition ext3. Il est chargé au démarrage du PC dans la RAM, afin de pouvoir monter les FileSystem réels.

3.4.3.5 Comparaison de l'arborescence du initrd

Pour comparer les arborescences, nous allons décompresser les fichiers initrd.img situés dans */isolinux* du LiveCD et du DVD.

Pour cela, j'ai suivi les explications : https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/6.2_Release_Notes/installation.html

[7] http://www.linfo.org/vmlinuz.html : Définition du noyau vmlinuz

[8] http://fedoraproject.org/wiki/Dracut#Technical_details : Fonctionnalités de Dracut

- Arborescence initrd du DVD :

```
├── bin -> usr/bin
├── dev
├── etc
├── init -> /usr/lib/systemd/systemd
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── shutdown
├── sys
├── sysroot
├── tmp
├── usr
└── var
```

- Arborescence du LiveCD

```
├── bin -> usr/bin
├── dev
├── etc
├── init -> /usr/lib/systemd/systemd
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── shutdown
├── sys
├── sysroot
├── tmp
├── usr
└── var
```

Nous voyons que les deux arborescences sont identiques. Afin de rentrer dans les détails des structures, j'ai effectué un diff :

```
diff -r  initrd/live/ initrd/dvd/ > /home/Share/Analysis/diff-initrd.txt
```

Voici un extrait :

```
Only in /home/deploy/initrd/dvd/lib/modules/3.6.10-4.fc18.x86_64/kernel/fs:
hfs
Only in /home/deploy/initrd/dvd/lib/modules/3.6.10-4.fc18.x86_64/kernel/fs:
hfsplus
```

Nous voyons que notre initrd dans le DVD possède des modules supplémentaires tels que les drivers de montages de FileSystem HFS[9] proposés par Apple. Ceci nous montre déjà que l'installation via DVD intègre un plus grand nombre de drivers de communication.

3.4.3.6 Ananlyse du squashfs

Un ISO Fedora LiveCD ou DVD est constitué d'un fichier primordial qui est : squashfs.img situé dans ***/LiveOS.***

[9] http://fr.wikipedia.org/wiki/Hierarchical_File_System

Ce fichier va contenir le FileSystem final, c'est à dire la partition principale de notre système d'exploitation contenant tous les dossiers où se logent les différents fichiers des applications. Cette partition est au format ext3.
https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/s1-filesystem-fhs.html

Taille du SquashFS.img :
DVD : 200 Mo
LiveCD : 800 Mo

On analyse le contenu de cette partition ext3 :

```
mount -t squasfs squasfs.img /mnt
```

Sur le **DVD** :

```
ext3-DVD/
├── bin -> usr/bin
├── dev
├── etc
├── firmware -> lib/firmware
├── fonts.scale
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── lost+found
├── mnt
├── modules -> lib/modules
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── sys
├── tmp
├── usr
└── var
```

Sur le **LiveCD** :

```
ext3-Live/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

A première vue, il n'y a pas de grandes différences entre la racine du LiveCD et du DVD.
Analyse du dossier ***/bin*** contenant les binaires des applications :
1546 Fichiers pour LiveCD
466 Fichiers pour le DVD
En faisant un « diff » entre ces deux fichiers nous voyons que le LiveCD possède les paquets tels que :

- Gnome
- qemu-kvm
- OpenOffice

Le FileSystem (Squashfs) du LiveCD qui va être copié en RAM au moment du boot du PC, contient déjà tous les paquets permettant d'utiliser pleinement Fedora18 et est déjà complétement configuré, comme par exemple le choix de la langue ainsi que les paquets déjà installés.

Dans notre DVD-Fedora 18, notre FileSystem ne possède qu'un nombre limité de paquets.
Nous aurons une phase d'installation et de personnalisation du Noyau grâce à l'installer Fedora.
Cet installer est géré par l'outil Anaconda qui permet soit d'interpréter un Fichier Kickstart ou soit de paramétrer grâce à un GUI les différents paramètres :

3.4.3.7 Etape du déploiement

Phases d'installation de Fedora à partir du DVD :

1. Extraction du vmlinuz et initrd.img
2. Copie du FileSystem initrd et vmlinuz dans la RAM (Dracut opérationnel)
3. Extraction du FileSystem à partir du squashfs.img grâce à Dracut
4. Copie du FileSystem Squashfs dans la RAM
5. Exécution du Fedora installer
6. L'utilisateur entre les différentes informations de configuration (exp : réseau, partitionnement des disques)
7. Copie du FileSystem sur le disque dur
8. Installation des paquets à partir du dépôt local contenu dans l'ISO ***/Packages*** dans le FileSystem

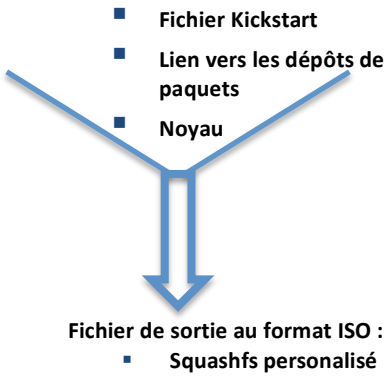
Le fait d'installer Fedora sur le disque dur via le DVD-Fedora18 va permettre d'avoir une installation persistante et d'utiliser Anaconda comme interpréteur de Kickstart afin de personnaliser notre système.

Phases d'installation de Fedora à partir du LiveCD:
Les étapes 1 à 4 sont identiques

5. Exécution de Fedora grâce au Squashfs complet copié en RAM

3.4.4 Création d'un LiveCD via la génération d'un ISO

Deux techniques de déploiement seront utilisées à travers ce projet. La première consiste à générer un fichier ISO contenant notre distribution personnalisée sous forme de LiveCD. A travers cette méthode nous allons recréer un LiveCD comme le LiveCD officiel (Voir 3.4.3) mais contenant uniquement les paquets et la configuration qui nous intéresse.
Génération d'un LiveCD :



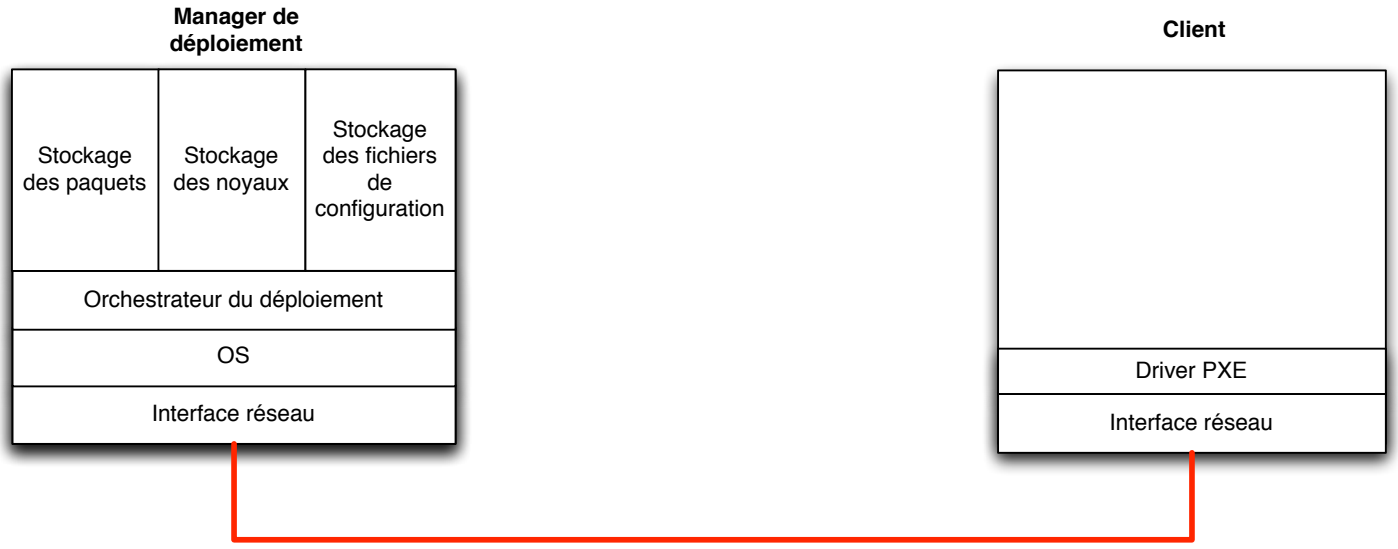
Ce fichier ISO a la même arborescence que le LiveCD officiel, mais le FileSystem Squashfs.img contient uniquement les paquets et la configuration listés dans le fichier Kickstart.

3.4.5 Via le réseau

Cette technique va permettre de déployer à distance des serveurs physiques basés sur l'amorçage PXE. Cette méthode sera vue comme une installation grâce au DVD-Fedora 18 (explication au §3.4.3). Le but est de transférer et d'installer une distribution personnalisée à travers le réseau.

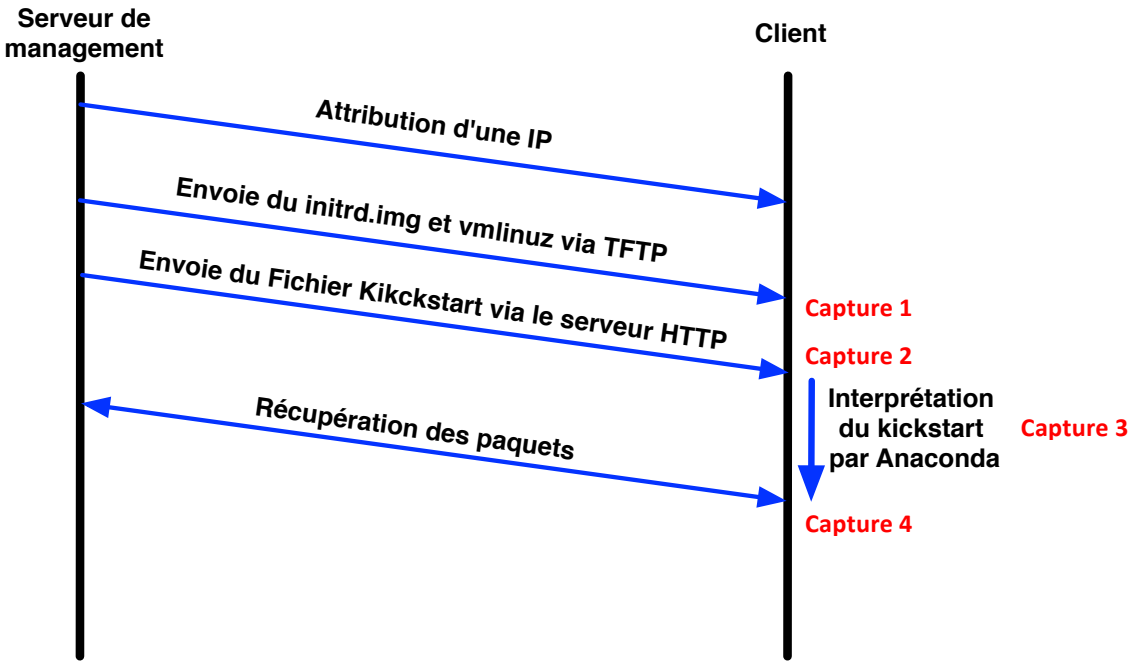
- Un Manager va contenir :
  - Un serveur DHCP
  - Un serveur TFTP
  - Un dépôt local
  - Un serveur HTTP pour envoyer les fichiers de configuration tel que le fichier Kickstart

Vue générale de notre infrastructure :



3.4.5.1 Echanges entre le serveur et le client :

Voir §3.4.5.2 pour matérialiser sous forme de texte ce qui est affiché au moment du Boot.



3.4.5.2 Captures de boot au format texte

Voici un extrait de la séquence de Boot, au moment de l'installation d'une distribution personnalisée via PXE. La méthode pour capturer le séquençage du boot est disponible en **Annexe C**.

Capture 1 : Récupération de vmlinuz et initrd.img

```
PXELINUX 3.61 2008-02-03  Copyright (C) 1994-2008 H. Peter Anvin

Loading /images/fc18-
x86_64/vmlinuz.....................................................
...................

Loading /images/fc18-
x86_64/initrd.img..................................................
.....................................................................
.....................................................................
.............................................ready.
```

Capture 2 : Récupération du fichier Kickstart

```
[    4.834138] 8139cp 0000:00:03.0: eth0: link up, full-duplex, lpa
0x05E1

dracut-initqueue[297]: % Total    % Received % Xferd  Average Speed
Time    Time     Time  Current


dracut-initqueue[297]: Dload  Upload   Total   Spent    Left  Speed


dracut-initqueue[297]: 0     0    0     0    0     0      0      0 --:--
:-- --:--:-- --:--:--     0
100  4077  100  4077    0     0  40747      0 --:--:-- --:--:-- --:--:--
41181
```

Capture 3 : Analyse du Kickstart au niveau de la partie « Pre » et configuration

```
anaconda 18.37.11 for Fedora 18 started.[1;24r[H[2;41H

Running pre-installation scripts

Starting automated install

Generating updated storage configuration

storage configuration....


================================================================================

Install hub


1) [x] Timezone settings    2) [x] Install Destination
(Europe/Zurich timezone)           (Automatic partitioning selected)

[x] Set root password
(Password is set.)

================================================================================
```

Capture 4 : Récupération + installation des paquets et exécution de la partie « Post »

```
Installing openssh-clients (263/422)
Installing ruby (345/422)
Installing libvirt-daemon (406/422)
Installing qemu-kvm (421/422)
Installing libvirt (422/422)
Performing post-install setup tasks
```

3.5 Outils de déploiement : LiveCD-Creator

3.5.1 Fonctionnement de LiveCD-Creator[10]

Une fois que l'on a édité le fichier Kickstart, cet outil CLI va permettre de générer un ISO contenant un LiveCD personnalisé (Voir §3.4.4).
LiveCD-Creator est un outil simple à utiliser, nécessitant quelques paramètres :

- L'emplacement du fichier Kickstart :

```
--config= «chemin jusqu'au fichier ks »
```

[10] http://doc.fedora-fr.org/wiki/Création_de_Live_CD/DVD_et_de_Live_USB : Description de l'outil LiveCD-Creator

- Le nom du LiveCD ainsi que celui du fichier iso:

```
--fslabel= «nom du LiveCD »
```

- L'emplacement des fichiers temporaires

```
--cache= «chemin du stockage des fichiers temporaires»
```

Voici un exemple de la commande complète :

```
livecd-creator --config=/home/deploy/files/ks_2.cfg
--fslabel=Fedora18-mini --cache=/var/tmp
```

Cet outil va en réalité créer un système d'exploitation constitué du noyau Fedora 18 ainsi que des différents paquets qu'il a téléchargé.
Pour une application réelle de cet outil, un scénario applicatif est décrit dans le §4.2.

3.6 Outils de déploiement : Cobbler[11]

L'outil Cobbler va être le chef d'orchestre de notre déploiement. Il est basé sur le système d'amorçage PXE. Il va gérer les différents services utiles durant le déploiement à travers le réseau :

- Le serveur DHCP
- Le serveur DNS
- Le serveur TFTP
- Le serveur HTTP
- La création de dépôts locaux

Par exemple pour le serveur DHCP, on va éditer un fichier dhcp.template en mentionnant notre range IP.

3.6.1 Déploiement d'une distribution personnalisée pour chaque client

Grâce à la gestion dynamique des services énoncée ci-dessus, nous pouvons envoyer une configuration de distribution personnalisée à chaque client. Comme par exemple, ils auront un nom et une adresse IP qui leurs sont propres.

Afin de gérer la personnalisation de la distribution, Cobbler intègre différents objets :

- Distro : Regroupe l'emplacement des fichiers de base du noyau tel que les boots loaders « initdr.img »
- Profile : Attribue un fichier Kickstart et un dépôt local à une distro. Pour personnaliser chaque client, j'ai créé un Kickstart générique contenant des variables.
  Exemple :

```
network  --bootproto=static --device=$stoIF --gateway=$stoGAT
--ip=$stoIP  --netmask=$stoSUB --onboot=yes --nodns
```

[11] http://www.cobblerd.org/manuals/2.2.3/2_-_Cobbler_Quickstart_Guide.html : Installation et configuration de l'outil Cobbler

- System : Chaque « System » va représenter notre client. On va lui greffer un profile, une adresse MAC afin d'indiquer au server PXE-TFTP quel client a le droit de booter via PXE. De plus, on va attribuer des valeurs aux variables du fichier Kickstart, ce qui va créer un fichier Kickstart par client. Exemple :

```
cobbler system add --name=sys_fc18_client2 --profile=fc18_generic --dns-
name=client2.cirrus --interface=p33p1 --mac=c8:60:00:73:a4:a4 --
ksmeta="stoIF=eth1 stoGAT=10.2.0.1 stoIP=10.2.4.102 stoSUB=255.255.0.0
mgtIF=eth0 mgtGAT=192.168.1.1 mgtIP=192.168.1.82 mgtSUB=255.255.255.0"
```

3.7 Scénarios

3.7.1 Scénario 1 : Création d'un ISO personnalisé grâce à LiveCD-Creator

Dans notre cas, le but sera :

- de générer un LiveCD déployant le kernel Fedora.
- d'ajouter tous les packages permettant le montage automatique d'un partage NFS.

3.7.2 Scénario 2 : Déploiement de Fedora 18 grâce à Cobbler

Déployer un hyperviseur via le réseau. Dans un premier temps ce scénario va permettre une maîtrise de l'outil Cobbler. Pour se faire, tout comme le scénario précédent, je vais :

- déployer un Fedora 18 à travers le réseau
- automatiser le montage d'un partage NFS

Cette fois-ci le déploiement de tous les éléments se fera à travers le réseau. Ce système va permettre de déployer automatiquement un hyperviseur avec une configuration individualisée tels que la configuration réseau. Voir schéma **page 24**.

3.7.3 Scénario 3 : Déploiement et management d'un hyperviseur

Déployer automatiquement deux hyperviseurs grâce à notre manager Cobbler (**Etape 1 page 27**) et gérer le déploiement de VM grâce à OpenNebula (**Etape 2 page 27**)
Nous aurons besoin des éléments suivants:

- Manager : Constituer de Cobbler et OpenNebula
- Unité de stockage : Intégrant un partage NFS pour le stockage des VM
- Deux hyperviseurs

Les fonctionnalités :

- Déployer un hyperviseur sans intervention extérieur :
  - Configuration du firewall applicatif de l'hyperviseur pour autoriser les connexions entrantes VNC afin d'utiliser l'affichage du bureau à distance
  - Envoie des fichiers de configuration pour libvirt et qemu
  - Permettre au manager de se connecter aux différents hyperviseurs via SSH
  - Connexion SSH avec une authentification par clés SSH entre les hyperviseurs pour les besoins de la Live Migration
  - Les hyperviseurs doivent avoir une unité de stockage commune
- Le manager grâce à Cobbler doit pouvoir:
  - Envoyer une configuration personnalisée à chaque hyperviseur
  - Gérer dynamiquement le serveur DNS, car dès qu'un nouvel hyperviseur est déployé il faudra rentrer l'équivalence FQDN - IP dans les fichiers de configuration du DNS
  - Déployer des VM sur les hyperviseurs
- La distribution sera directement copiée sur le disque dur, et donc sera persistante

4 Réalisation

4.1 Scénario 1 : Création d'un ISO personnalisé grâce à LiveCD-Creator

4.1.1 Installation de LiveCD-Creator

```
yum install livec-tools
```

4.1.2 Edition du fichier Kickstart

Paramètres du fichier Kickstart :

- Le clavier : Français-Suisse
- Le Noyau Linux (vmlinuz) sera reprit de l'ISO officiel fournit par le DVD Fedora (§3.4.3)
- L'adresse IP sera acquise via DHCP
- Le mot de passe root sera : rootroot
- La langue du système sera en anglais
- Partition de 6GB
- L'URL du dépôt sera : http://mirror.switch.ch/ftp/mirror/fedora/linux/releases/18/Everything/x86_64/os

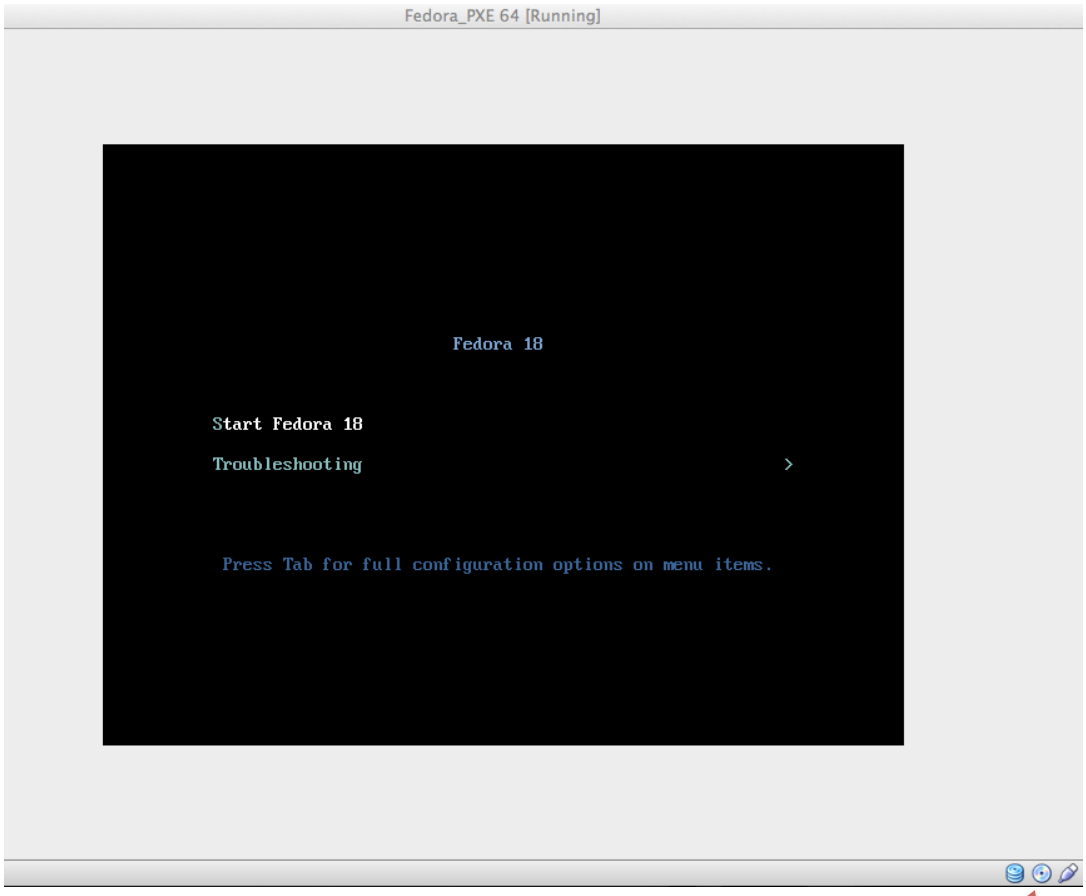
Pour la liste des paquets installés dans l'ISO, voir le fichier Kickstart en Annexe A.
Le script de post-installation va s'exécuter afin de :

- Créer un utilisateur « user »
- Créer /media/user_nfs, qui est le dossier qui va permettre de monter le partage NFS
- Permettre l'édition du fichier fstab, afin de rendre permanent le montage NFS :

```
echo '192.168.1.16:/home/ben/data_nfs/data_user /media/user_nfs nfs rw 0
0' >> /etc/fstab
```

4.1.3 Test du fichier ISO

J'utilise Virtualbox pour tester la distribution contenue dans l'ISO. Je crée une VM dans laquelle j'installe mon LiveCD au format ISO.



Print screen 1: Démarrage du LiveCD

LiveCD ISO monté avec la VM

```
         Starting Security Auditing Service...
         Starting NTP client/server...
         Starting Login Service...
         Starting System Logging Service...
[  OK  ] Started System Logging Service.
         Starting D-Bus System Message Bus...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Security Auditing Service.
[  OK  ] Started NTP client/server.
[  OK  ] Started Login Service.
[  OK  ] Started firewalld - dynamic firewall daemon.
         Starting Network Manager...
         Starting Authorization Manager...
[  OK  ] Started Authorization Manager.
[  OK  ] Started Network Manager.
         Starting Network Manager Wait Online...
[  OK  ] Started Network Manager Wait Online.
[  OK  ] Reached target Network.
         Starting OpenSSH server daemon...
         Starting Sendmail Mail Transport Agent...
         Starting RPC bind service...
[  OK  ] Reached target Remote File Systems (Pre).
         Mounting /media/user_nfs...
[  OK  ] Started RPC bind service.
```

Print screen 2 : Démarrage de Fedora

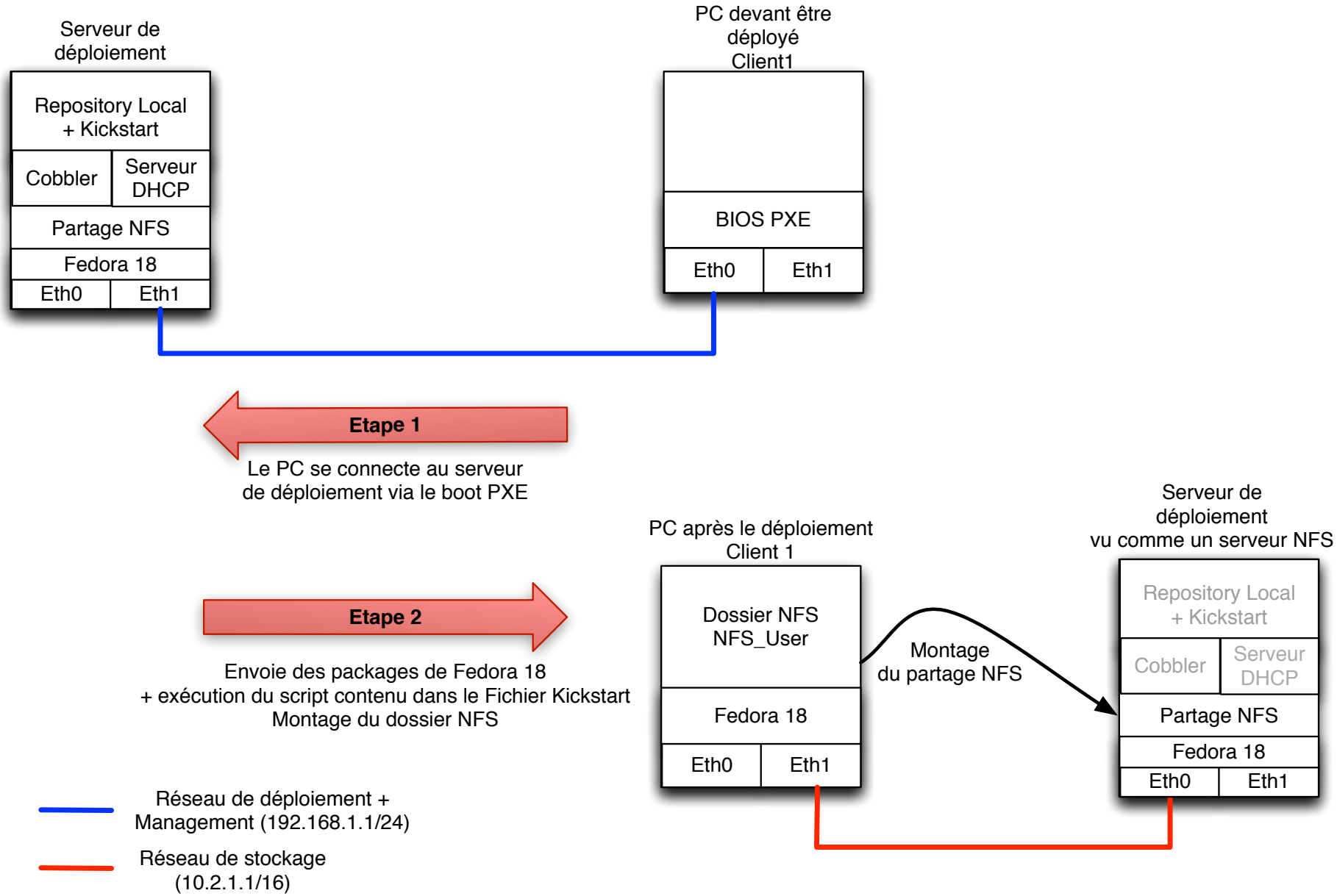
4.1.4 Conclusion

La prise en main de LiveCD-Creator a été très aisée, cet outil m'a permis de comprendre comment se déroule l'installation d'un système d'exploitation, ainsi que le fonctionnement des fichiers Kickstart. Cette méthode pour personnaliser Fedora 18 en générant un fichier ISO est très pratique pour installer le système sur de nombreux postes identiques. Car par la suite, on peut aisément graver notre ISO sur un CD-Rom afin de le déployer sur des PC physiques.

Cette méthode ne sera pas utilisée dans la suite du projet car il faudra déployer des hyperviseurs avec des configurations différentes sans intervention humaines. De plus, pour tester la fonctionnalité de notre fichier Kickstart nous devons attendre la fin de la génération de l'ISO. Par soucis d'agilité, nous allons donc nous tourner vers une infrastructure de déploiement à travers le réseau en utilisant l'outil Cobbler.

4.2 Scénario 2 : Déploiement de Fedora 18 grâce à Cobbler

4.2.1 Schéma



4.2.2 Caractéristique du serveur de déploiement

Notre serveur de déploiement va servir de serveur de stockage NFS. Il va héberger l'outil Cobbler qui va englober (voir les détails de l'installation via le réseau §3.4.5) :

- Le serveur DHCP
- Un serveur TFTP
- Un serveur Web pour héberger le dépôt local
- 2 cartes Ethernets :
  - Une carte (**Eth0**) pour le réseau de stockage (10.1.2.3.67)
  - Une carte (**Eth1**) pour le réseau de management (192.168.1.1)
- Un serveur NFS

4.2.3 Mise en place de Cobbler[12]

Installation :

```
yum install cobbler
```

Dans le fichier : ***/etc/cobbler/settings***
Nous allons paramétrer le fait que Cobbler gère le service DHCP automatiquement, puis rentrer l'adresse IP de notre serveur TFTP, appelé « next_serveur ». Dans notre cas nous allons mettre 192.168.1.1.

Edition du fichier de template « dhcp.template » situé ***/etc/cobbler/***
Il servira de fichier de configuration au moment du démarrage du service DHCP.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
     option routers             192.168.1.1;
     option domain-name-servers 192.168.1.1,192.168.1.1;
     option subnet-mask         255.255.255.0;
     filename                   "/pxelinux.0";
     default-lease-time         21600;
     max-lease-time             43200;
     range dynamic-bootp        192.168.1.50-80
     next-server                $next_server;
}
```

Démarrage du service Cobbler

```
systemctl start cobblerd.service
```

Pour démarrer l'outil afin qu'il amorce tous les services annexes tels que le service DHCP et HTTP.
Exécution de la commande :

```
cobbler sync
```

[12] http://www.cobblerd.org/manuals/2.2.3/2_-_Cobbler_Quickstart_Guide.html : Mise en place rapide de Cobbler

4.2.4 Importation de la distribution déployée

Pour se constituer un dépôt local, j'ai choisi d'importer une distribution DVD de Fedora18 : http://mirror.switch.ch/ftp/mirror/fedora/linux/releases/18/Fedora/x86_64/iso/Fedora-18-x86_64-DVD.iso
Montage de la distribution :

```
mount -t iso9660 -o loop,ro /home/deploy/Download/Fedora-18-x86_64-
DVD.iso /mnt
```

Cobbler va importer cette distribution dans son espace situé ***/var/www/cobbler***

```
cobbler import --name=fedora18 --arch=x86_64 --path=/mnt
```

Grâce à cette commande, Cobbler va transférer tous les paquets contenus dans ce DVD afin de se constituer son repository local et obtenir tous les fichier du noyau de Fedora 18.

4.2.5 Elaboration du fichier Kickstart

Tout comme le scénario précédent, il a fallu écrire un nouveau fichier Kickstart (voir Annexe B). Il reprend quasiment les mêmes informations que le précèdent. Comme notre **Client1** possède deux cartes Ethernets (stockage et management) il faut hiérarchiser la mise sous tension de ces dernières au démarrage du PC. Dans le fichier Kickstart la carte Eth0 du PC Client, possède le paramètre « on_boot = NO ». Cela permet à la carte Eth1 d'être la seule à envoyer la requête au serveur NFS.
Une fois la phase de démarrage achevée, Fedora active la carte Eth0 grâce à la commande « ifup eth0 » intégrée dans le script de démarrage. Grâce à cette astuce nous pouvons monter sans problème le partage NFS.

4.2.6 Création du Profil

Pour réellement déployer la distribution nous devons créer un profil qui regroupe :

- Le chemin du fichier Kickstart
- Et la distribution précédemment ajoutée

Ce profil est le nom que va avoir la distribution au niveau du menu du Boot PXE.

4.2.7 Déploiement du poste Client

Dans le Bios du poste client nous allons indiquer que le Boot PXE s'effectue en premier.
Voir les étapes du Boot PXE au §3.4.5

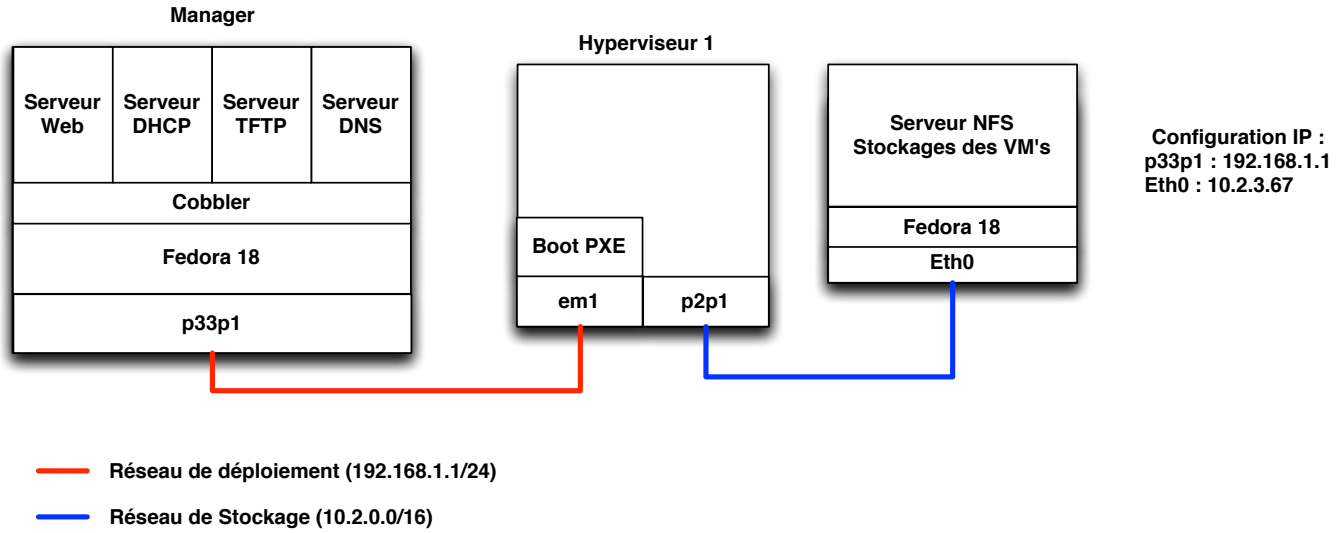
4.2.8 Conclusion

A travers ce scénario, j'ai pu mettre en place le déploiement automatique via le réseau d'une distribution Fedora 18 grâce à l'outil Cobbler. Il sera très utile par la suite car en rajoutant quelques paquets et instructions au fichier Kickstart tels que les paquets de virtualisation et les fichiers de configurations de libvirt. On pourra déployer automatiquement des hyperviseurs qui pourront être gérés par OpenNebula. Dans ce scénario l'utilisateur doit manuellement choisir le profil à déployer. Dans le prochain scénario Cobbler pourra déployer des profils automatiquement en fonction des adresses mac des PC clients qui l'interrogent.

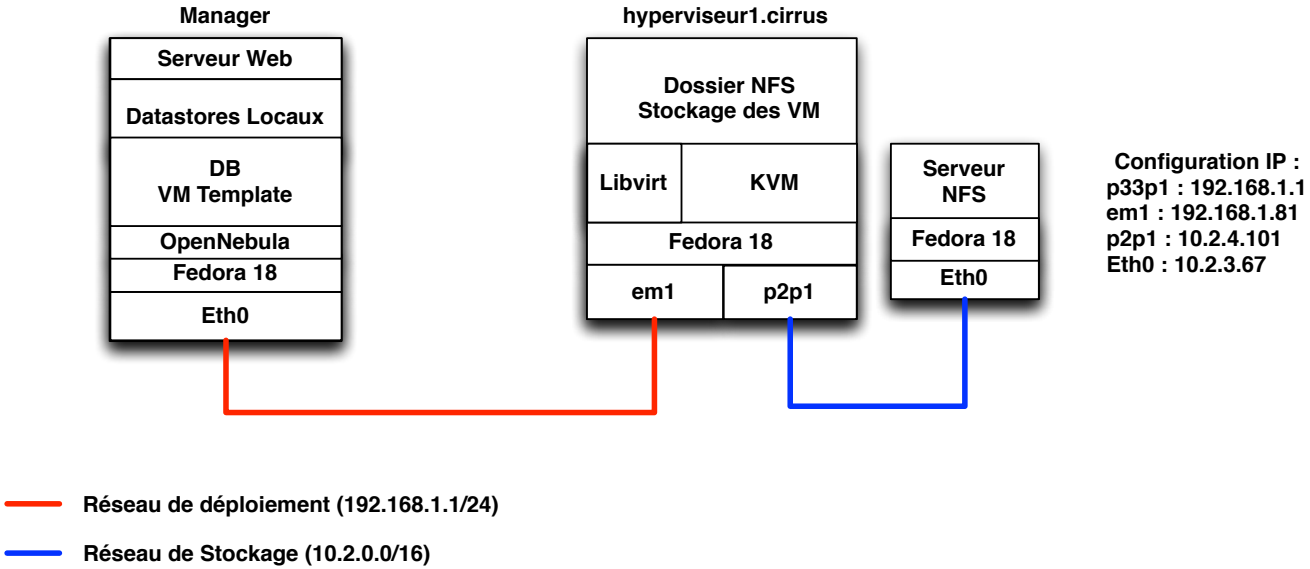
4.3 Scénario 3 : Déploiement complet d'hyperviseur

4.3.1 Déroulement du déploiement

Etape 1 : Voici la topologie global, l'hyperviseur n'a pas encore d'adresse IP attribuée



Etape 2 : Notre hyperviseur est opérationnel, notre manager se mue en manager de VM grâce à OpenNebula



4.3.2 Héritage du scénario précédent

Par rapport au scénario précédent, nous allons reprendre la même configuration de base, mais en rajoutant un serveur DNS, en recréant un nouveau fichier Kickstart adapté à nos besoins et en supprimant les « profiles et systems ».

4.3.3 Configuration du serveur DNS

Cobbler va permettre de gérer notre serveur DNS, mais au préalable il faut avoir téléchargé le paquet **named.** Cobbler mettra dynamiquement à jour les fichiers de zone DNS au moment où un hyperviseur est déployé. Pour la mise à jour des zones Cobbler redémarre le service named. Notre zone DNS est .cirrus

Modifier dans le fichier ***/etc/cobbler/settings*** :

```
manage_dns: 1           #Indique à Cobbler qu'il gére le service dns
manage_forward_zones: [cirrus]      #Le nom de la zone DNS
manage_reverse_zones: [192.168.1]   #Les adresses IP de la zone
```

Configurer le fichier de zone comportant les corrélations IP-DNS dans ***/etc/cobbler/zone.template***

```
\$TTL 300
@                       IN      SOA     $cobbler_server. nobody.example.com.
(
                                        $serial   ; Serial
                                        600         ; Refresh
                                        1800         ; Retry
                                        604800       ; Expire
                                        300          ; TTL
                                        )

                        IN      NS      $cobbler_server.

manager IN A 192.168.1.1 #Ajouter l'entrée DNS persistante pour le manager
$host_record      #Variable qui va être remplie par cobbler pour ajouter les
                  #nouveaux hyperviseurs déployés
```

4.3.4 Création du fichier Kickstart

Voici la première partie du fichier Kickstart contenant la configuration du noyau. Il n'aura pas de valeur fixe mais aura des variables. Je me suis basé sur un Kickstart de base présent dans Cobbler que j'ai modifié. Le texte en gras indique les lignes que j'ai ajoutées :

```
#platform=x86, AMD64, or Intel EM64T
# System authorization information
auth  --useshadow  --enablemd5
# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --all --initlabel
# Use text mode install
text
# Firewall configuration (Ouverture des ports pour VNC et Live Migration)
firewall --enabled --service=vnc-server --port=5000-5999:tcp,5000-5999:udp,49000-
50000:tcp
# Run the Setup Agent on first boot
firstboot --disable
# System keyboard (Clavier Suisse)
keyboard 'fr_CH'
```

```
# System language
lang en_US
#add user (Ajout d'un utilisateur pour OpenNebula)
user --name=oneadmin --password=oneadmin --uid=10000 --shell=/bin/bash --
homedir=/var/lib/one
# Use network installation
url --url=$tree
# Network configuration (Configuration des cartes storages et management)
network  --bootproto=static --device=$mgtIF --gateway=$mgtGAT --ip=$mgtIP  --
netmask=$mgtSUB --onboot=no  --nodns
network  --bootproto=static --device=$stoIF --gateway=$stoGAT --ip=$stoIP  --
netmask=$stoSUB --onboot=yes --nodns
# Reboot after installation
reboot
#Root password (Password contenu dans le fichier /etc/cobbler/settings)
rootpw --iscrypted $default_password_crypted
# SELinux configuration
selinux --enable
# Do not configure the X Window System
skipx
# System timezone
timezone Europe/Zurich
# Install OS instead of upgrade
install
# Clear the Master Boot Record
zerombr
# Allow anaconda to partition the system as needed
autopart
```

Liste des paquets à installer :

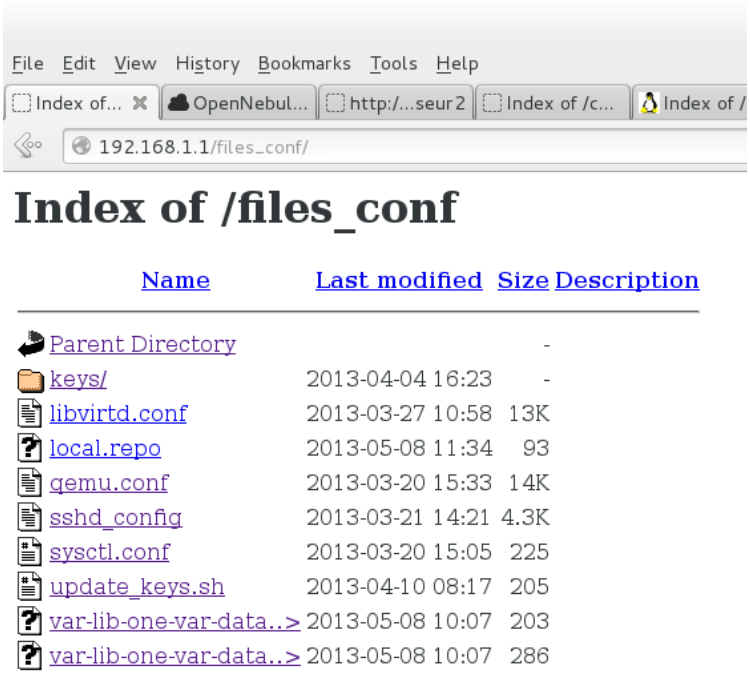
```
%packages
@Core
@Base
@Virtualization
bash
kernel
passwd
policycoreutils
chkconfig
authconfig
rootfiles
grub2
efibootmgr
nfs-utils
net-tools
nano
wget
ruby
ruby-irb
ruby-libs
kvm
%end
```

Les paquets en gras sont ceux que j'ai ajoutés en fonction :

- Des prérequis d'OpenNebula[13] (ruby)
- Des besoins de la virtualisation (kvm et @Virtualisation)
- De la connexion NFS (nfs-utils)

[13] http://opennebula.org/documentation:archives:rel3.8:ignc : Prérequis de l'installation

Voici la dernière partie du Kickstart, qui contient le script que j'ai écrit permettant la configuration de l'hyperviseur, en fonction des besoins du scénario.
Au niveau de cette partie, l'hyperviseur qui est déployé va rechercher les différents fichiers de configuration présents sur le serveur HTTP du manager :



Partie %post du fichier Kickstart détaillé:

```
%post
#!/bin/bash
```

- L'hyperviseur se connecte au serveur NFS :

```
#######Connexion NFS######
mkdir /var/lib/one/var/datastores/
wget -P /etc/systemd/system/ http://192.168.1.1/files_conf/var-lib-one-var-datastores-
0.mount
wget -P      /etc/systemd/system/ http://192.168.1.1/files_conf/var-lib-one-var-
datastores-0.automount
systemctl daemon-reload
systemctl enable var-lib-one-var-datastores-0.automount
```

Pour se connecter, nous n'utilisons pas le fichier fstab, mais le montage NFS va se faire via le service mount. Pour se faire, j'ai dû créer deux fichiers, .mount et .automount. Les fichiers de services sont appelés « Files Units »[14].
Voici le fichier .mount :

```
[Unit]
Description=Mount NFS Storage for VM
Wants=network.target rpcbind.service
After=network.service rpcbind.service

[Mount]
What=10.2.3.67:/home/deploy/share_nfs
Where=/var/lib/one/var/datastores/0
Type=nfs
Options=noatime
TimeoutSec=15
StandardOutput=syslog
StandardError=syslog
```

[14] https://bbs.archlinux.org/viewtopic.php?id=150208 : Création du Files Unit pour le automount

Ce fichier recense tous les paramètres utiles au montage du partage NFS.
Nous avons un fichier .automount qui va automatiser le montage du partage NFS :

```
[Unit]
Description=NFS Storage Automount
Wants=network.target rpcbind.service
After=network.service rpcbind.service

[Automount]
Where=/var/lib/one/var/datastores/0

[Install]
WantedBy=multi-user.target
```

Les deux fichiers de services doivent posséder le nom du chemin de montage du NFS donc ***var-lib-one-var-datastores-0.automount*** et ***var-lib-one-var-datastores-0.mount***
Pour démarrer le service au boot de l'hyperviseur, il faut « enable » le service .automount.

- Configuration de libvirt et qemu en fonction des recommandation d'OpenNebula[15] :

```
##Configuration Libvirtd and qemu
wget -P /etc/libvirt/ -N http://192.168.1.1/files_conf/libvirtd.conf
wget -P /etc/libvirt/ -N http://192.168.1.1/files_conf/qemu.conf
wget -P /etc/ -N http://192.168.1.1/files_conf/sysctl.conf
```

- Configuration de l'utilisateur pour la communication avec OpenNebula

```
###Manage User
/usr/sbin/groupadd -g 10000 cloud
/user/sbin/usermod -a -G cloud,root oneadmin
/bin/chown -R oneadmin:cloud /var/lib/one
/bin/chown oneadmin:cloud /var/run/libvirt/libvirt-sock
echo 'oneadmin      ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers
```

- Envoie du fichier de configuration du serveur SSH

```
##Config SSH server
cd /etc/ssh/
wget -N http://192.168.1.1/files_conf/sshd_config

##Create Directory SSH
mkdir /var/lib/one/.ssh
/bin/chown -R oneadmin:cloud /var/lib/one/
```

- Création d'une connexion ssh avec authentification par clés SSH:

Pour que nos hyperviseurs puissent recevoir des instructions SSH du manager, il faut qu'il récupère la clé publique du manager stockée dans le fichier **authorized_keys** sur le serveur HTTP.

```
####Recover Script Update authorized_key
mkdir /var/lib/one/script
cd /var/lib/one/script
wget -N http://192.168.1.1/files_conf/update_keys.sh
/bin/chmod +x update_keys.sh
```

- Partage des clés SSH des hyperviseurs :

Pour la live-migration, les hyperviseurs doivent communiquer entres eux via SSH, donc pour se faire, chaque hyperviseur va générer et ajouter sa clé publique dans le fichier authorized_keys

```
###Generate Public Keys
mkdir /mnt/keys
mount -t nfs 192.168.1.1:/var/www/html/files_conf/keys /mnt/keys
```

[15] http://opennebula.org/documentation:archives:rel3.8:kvmg#kvm_configuration : Configuration de libvirt et qemu

```
/bin/su - oneadmin -c "ssh-keygen -q -t rsa -N '' -f /var/lib/one/.ssh/id_rsa"
/bin/cat /var/lib/one/.ssh/id_rsa.pub >> /mnt/keys/authorized_keys
cd /var/lib/one/.ssh
wget -N http://192.168.1.1/files_conf/keys/config
```

- Récupération du fichier authorized_keys

```
####Recover Public KEY
wget -N http://192.168.1.1/files_conf/keys/authorized_keys
/bin/chmod 700 /var/lib/one/.ssh
/bin/chmod 600 /var/lib/one/.ssh/authorized_keys
/bin/chown -R oneadmin:cloud /var/lib/one/
```

Activation de la carte réseau management, pour garantir que la carte storage est la seule à envoyer une requête NFS

```
#########ifup second PNIC
cd /etc/rc.d/
echo '#!/bin/bash' > rc.local
echo '/etc/sysconfig/network-scripts/ifup $mgtIF' >> rc.local
/bin/chmod +x /etc/rc.d/rc.local

%end
```

4.3.5 Création du profil

Création du profil qui va contenir la « distro » et le fichier Kickstart générique :

```
cobbler profile add --name=fc18_generic --distro=fc18-x86_64
--kickstart=/var/www/html/Kickstart_perso/ks_generic.ks
```

4.3.6 Création du système

Dans cette partie nous allons définir la configuration de chaque hyperviseur. Les valeurs commençant par un « $ » dans le Kickstart vont être remplacées par des valeurs fixes :

Nom	NomSystème	NomInterfaceMGT	MacAddressMGT	AddressIPMGT	NetMaskMGT	GatewayMGT	DNSServerMGT	NomInterfaceSTO
hyperviseur1.cirrus	sys_hyperviseur1	em1	e0:cb:4e:25:2f:34	192.168.1.81	255.255.255.0	192.168.1.1	192.168.1.1	p2p1
hyperviseur2.cirrus	sys_hyperviseur2	p33p1	c8:60:00:73:a4:a4	192.168.1.82	255.255.255.0	192.168.1.1	192.168.1.1	p16p2

Nous allons créer un système par hyperviseur, nos configurations seront différenciées grâce à l'adresse Mac de leurs cartes Ethernets de management.

```
cobbler system add --name=sys_fc18_hyperviseur1 --profile=fc18_generic -
-dns-name=hyperviseur1.cirrus --interface=em1 --ip-address=192.168.1.81
--mac=e0:cb:4e:25:2f:34 --ksmeta="stoIF=p2p1 stoIP=10.2.4.101
mgtIF=p16p1 mgtIP=192.168.1.81 stoDNS=10.2.0.1 mgtDNS=192.168.1.1"
```

- --profile : Indique que l'on se greffe au profile précédent
- --dns-name : Le nom de l'hyperviseur. Cobbler va injecter ce nom dans le fichier de zone DNS
- --interface : Le nom de la carte réseau qui va servir au déploiement
- --ip-address : L'adresse IP qui sera corrélée au nom dans le fichier de zone DNS

- --mac : Au moment où cette adresse MAC fera une requête DHCP auprès du serveur PXE, l'hyperviseur pourra charger ce « system »
- --ksmeta : Valeurs qui seront injecter dans le fichier Kickstart

4.3.7 Configuration du manager OpenNebula sur le PC de déploiement

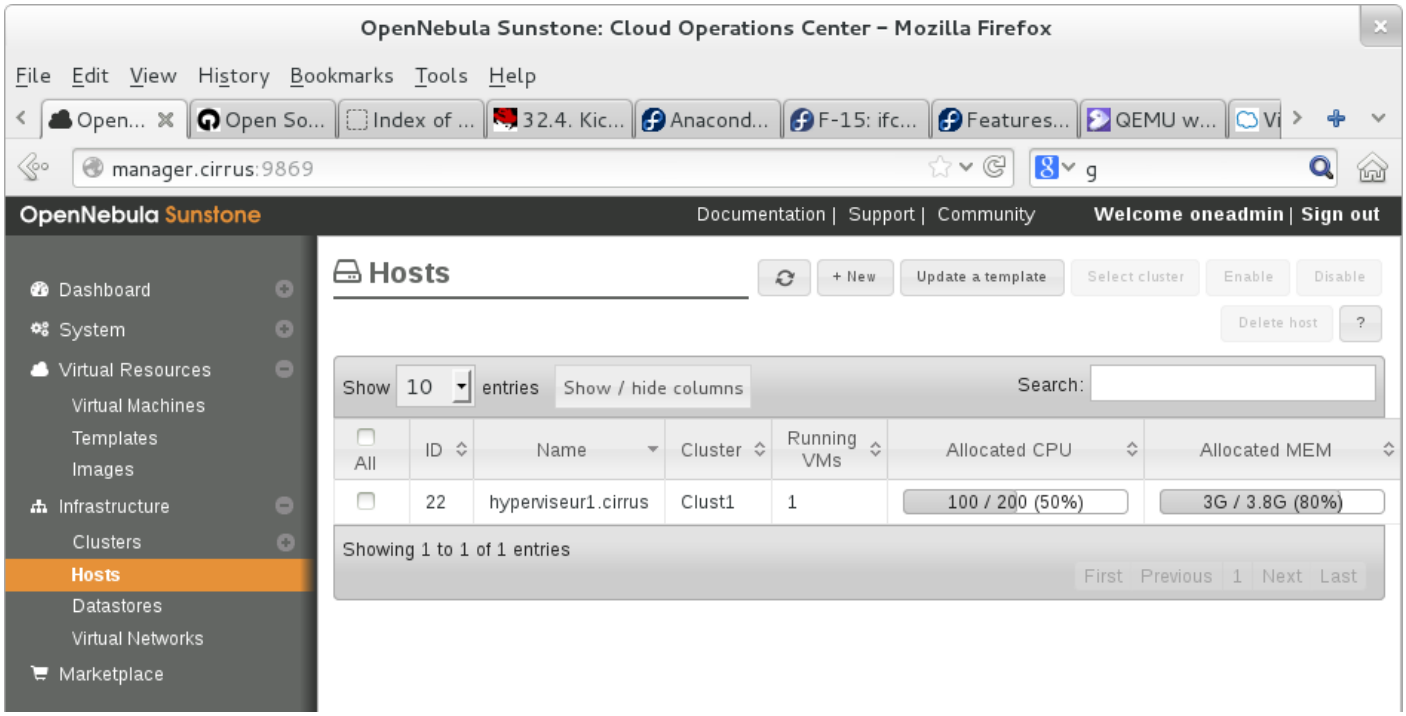
J'ai suivi le document http://www.tdeig.ch/kvm/Chalut_RTB.pdf en page 60.

4.3.8 Déploiement de l'hyperviseur

1. Mise sous tension du PC client (futur hyperviseur)
2. Configuration du BIOS, pour définir le boot prioritaire PXE
3. Le PC demande une adresse IP au près du serveur de déploiement
4. Le déploiement commence
5. Au bout de 4 minutes, l'hyperviseur est opérationnel

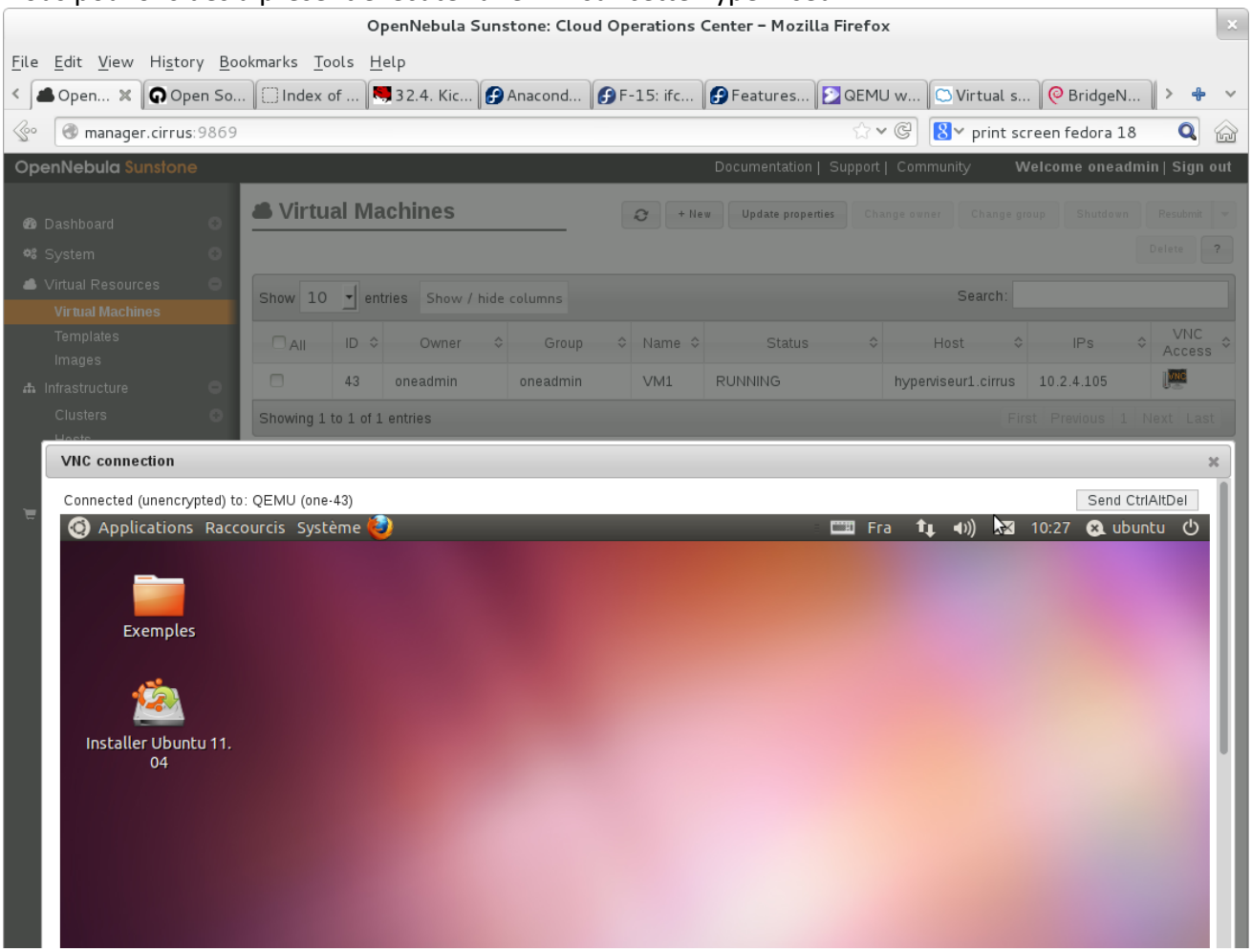
4.3.9 Management de l'infrastructure virtualisée

Comme « très bien » expliqué dans le travail de Bachelor sur le Cloud Management http://www.tdeig.ch/kvm/Chalut_RTB.pdf, nous pouvons gérer l'hyperviseur depuis le serveur web OpenNebula accessible depuis : http://manager.cirrus :9869



4.3.10 Déploiement d'une VM

Nous pouvons dès à présent exécuter une VM sur cette hyperviseur :



4.3.11 Conclusion

A travers ce scénario, j'ai pu mettre en œuvre une solution de déploiement personnalisée pour différents hyperviseurs. Cette solution Cobbler permet de faire une gestion dynamique du parc de machines physiques, grâce à la mise à jour de la zone DNS. Au niveau du projet, le déploiement passe encore par de nombreuses manipulations, donc un script résumant toutes ces étapes pourrait automatiser la création de systemes.

5 Conclusion

A travers ce projet, j'ai pu améliorer mes connaissances dans la compréhension de l'installation de distributions officielles ou personnalisées par moi même. J'ai exploré le fait d'installer un nouveau serveur physique à travers le réseau. Grâce à ces techniques de déploiement, nous pouvons créer une distribution correspondant exactement à nos besoins. Ce que se traduit par une taille de distribution plus petite, et contenant uniquement les packages qui nous sont utiles.
Voici ci-dessous quelques bonnes pratiques pour mener à bien le déploiement à distance.

5.1 Bonnes pratiques

Voici un résumé de bonnes pratiques à appliquer dans le déploiement :

- Pour la copie de dépôts publiques en vue de créer un dépôt local, il est préférable de prendre les serveurs allemands, car ils offrent une bande passante plus élevée que les autres (environ 1Gb/s) : http://mirrors.fedoraproject.org/publiclist/Fedora/18/
- Pour monter un partage NFS, il est préférable d'utiliser les Files Unit automout. Cette technique facilite le déploiement.
- Dans la partie Scripting du fichier, il est préférable de transférer des fichiers configuration plutôt que d'utiliser la fonction « echo >> » qui va aller écrire dans les fichiers. A l'aide du transfert complet de fichier nous pouvons utiliser le versionning grâce à un système GitHub

5.2 Perspectives futures

Ce projet d'approfondissement, va déboucher sur un Travail de Master donc voici quelques pistes qui vont être explorées par la suite :

- Créer un script qui va piloter le déploiement. Par exemple si on décide de déployer un hyperviseur déjà en production, dans le cas où des VMs sont actives et hébergées dessus, il faudra utiliser la live migration pour les migrer chez son voisin
- Utiliser Cobbler pour déployer un LiveCD :
  http://www.cobblerd.org/manuals/2.2.3/5/12_-_Booting_Live_CDs.html
  On pourra choisir soit de déployer un LiveCD ou une distribution gravée sur le disque dur
- Utiliser GitHub pour utiliser un système de versionning pour gérer les fichiers Kickstart et de configurations
- Depuis peu, RedHat vient de packager OpenStack[16], c'est une autre approche de gestion et de déploiement de VM qui pourrait être abordée dans ce travail
- Explorer la gestion des flux réseaux entre les VM et l'extérieur de l'hyperviseur grâce à Open vSwitch

6 Bibliographie

- http://fedoraproject.org/wiki/Anaconda/Kickstart : Liste des différentes options des fichiers Kickstart
- http://doc.fedora-fr.org/wiki/Fedora_Live_CD#Objectifs_et_fonctionnalit.C3.A9s : Analyse du contenu du LiveCD officiel Fedora 18
- http://www.linfo.org/vmlinuz.html : Analyse détaillé du Noyau Linux compressé « vmlinuz »
- https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/s1-filesystem-fhs.html : Explication du File System principal
- http://doc.fedora-fr.org/wiki/Création_de_Live_CD/DVD_et_de_Live_USB : Application pratique de l'outil « livecd-creator »
- http://www.cobblerd.org/manuals/2.2.3/2_-_Cobbler_Quickstart_Guide.html : Application pratique de l'outil « Cobbler »
- http://www.tdeig.ch/kvm/Chalut_RTB.pdf : Travail de Bachelor de Mr Benoît Chalut pour installer le manager OpenNebula

[16] http://docs.openstack.org/trunk/openstack-compute/install/yum/content/

7 Annexes

Afin d'avoir une structure des fichiers Kickstart j'ai installé le package **spin-kickstarts**, qui regroupe des exemples de fichiers Kickstarts dans ***/usr/share/spin-kickstarts/.***
Les parties en gras indiquent les parties que j'ai modifiées.

A Fichiers Kickstart du scénario 1

```
#platform=x86, AMD64, or Intel EM64T
#version=DEVEL
# Install OS instead of upgrade
install
# Firewall configuration
firewall --disabled
# Keyboard layouts
keyboard 'fr_CH'
# Use hard drive installation media
cdrom
# Network information
network  --bootproto=dhcp --device=eth0
# Reboot after installation
reboot
# Root password
rootpw --iscrypted $1$KLs.1eca$qGEp1w8UK43OsWfJgFo8W1
# System timezone
timezone Europe/Luxembourg
# System authorization information
auth  --useshadow  --passalgo=md5
# Use graphical install
graphical
firstboot --disable
# System language
lang en_US
# SELinux configuration
selinux --enabled
# Do not configure the X Window System
skipx
# System bootloader configuration
bootloader --location=mbr
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all --initlabel
# Disk partitioning information
part / --fstype="ext4" --size=6000

repo --name=update --baseurl=http://mirror.switch.ch/ftp/mirror/fedora/linux/releases/18/Everything/x86_64/os

%packages
@Core
@base
@admin-tools
@hardware-support
anaconda-runtime
bash
nano
kernel
passwd
```

```
policycoreutils
chkconfig
authconfig
rootfiles
grub-efi
grub2
efibootmgr
nfs-utils
net-tools
%end

%post --interpreter /bin/bash
#!/bin/bash
/usr/sbin/useradd user -m
echo -n 'user1'|passwd user --stdin
echo 'bonjour'
echo 'Connexion au nfs Server'
mkdir /media/user_nfs
#mount -t nfs 192.168.1.16:/home/ben/data_nfs/data_user /media/#user_nfs
echo '192.168.1.16:/home/ben/data_nfs/data_user /media/user_nfs nfs rw 0 0' >> /etc/fstab
%end
```

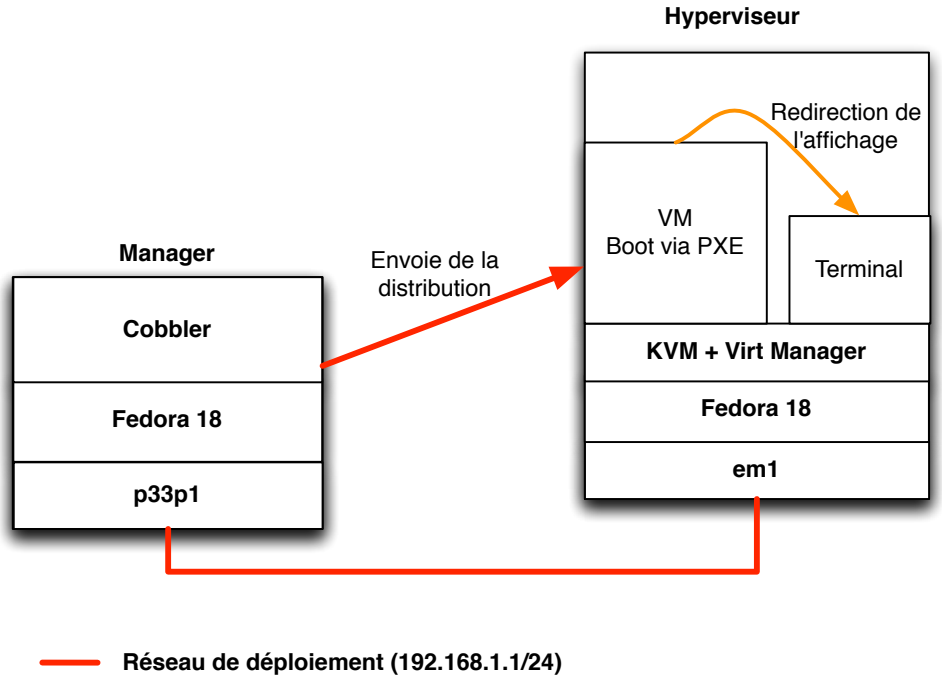
B Fichiers Kickstart du scénario 2

```
# System authorization information
auth  --useshadow  --enablemd5
# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --all --initlabel
# Use text mode install
text
# Firewall configuration
firewall --enabled
# Run the Setup Agent on first boot
firstboot --disable
# System keyboard
keyboard 'fr_CH'
# System language
lang en_US
#add user
user --name=oneadmin --password=oneadmin --uid=10000 --shell=/bin/bash --homedir=/var/lib/one
# Network information
network  --bootproto=static --device=em1 --gateway=10.2.0.1 --ip=10.2.4.101  --onboot=yes --netmask=255.255.0.0 --
nameserver=10.2.0.1
network  --bootproto=static --device=p2p1 --gateway=192.168.1.1 --onboot=no --ip=192.168.1.81  --
netmask=255.255.255.0
# Reboot after installation
reboot
#Root password
rootpw --iscrypted $default_password_crypted
# SELinux configuration
selinux --enabled
# Do not configure the X Window System
skipx
# System timezone
timezone Europe/Zurich
# Install OS instead of upgrade
```

```
install
# Clear the Master Boot Record
zerombr
# Allow anaconda to partition the system as needed
autopart

%pre
$SNIPPET('log_ks_pre')
$SNIPPET('kickstart_start')
#$SNIPPET('pre_install_network_config')
# Enable installation monitoring
$SNIPPET('pre_anamon')
%end

%packages
@Core
bash
kernel
passwd
policycoreutils
chkconfig
authconfig
rootfiles
grub2
efibootmgr
nfs-utils
nano
wget
ruby
ruby-irb
ruby-libs
%end

%post
#!/bin/bash
#######Connexion NFS######
mkdir /var/lib/one/var/datastores/
echo '10.2.3.67:/home/deploy/share_nfs /var/lib/one/var/datastores/0        nfs defaults        0        0' >> /etc/fstab
##########ifup second PNIC
cd /etc/rc.d/
echo '#!/bin/bash' > rc.local
echo '/etc/sysconfig/network-scripts/ifup p2p1' >> rc.local
/bin/chmod +x /etc/rc.d/rc.local

%end
```

C Capture de la séquence de boot

Pour capturer sous forme de texte les différentes informations du boot affichées dans le §3.4.5, j'ai déployé une VM via PXE. Le flux de texte affiché à l'écran par cette VM au démarrage du boot PXE, a pu être redirigé vers une console gérée par virt-manager sur l'hyperviseur.



Pour indiquer à la VM que la sortie de son affichage doit se faire via le terminal il faut :

- Editer le fichier ***/etc/cobbler/pxe/pxesystem.template*** sur le manager[17] :
  - Au début du fichier : serial 0 9600
  - Après la variable $append_line : console=tty0 console=ttyS0,9600n8
    On indique que la sortie de l'affichage est située sur le port série **tty0**. Ce fichier pxesystem.template sera le premier fichier transmis à la VM avant les fichiers vmlinuz et initrd.
- Comme la VM sera gérée via le virt-manager, dans les priorités de boot, il faut mettre en premier le boot PXE.
- Créer sous Cobbler un system qui possède comme MAC, l'adresse Ethernet de la carte virtuelle de la VM. Pour information : Le fait de déployer une VM ne change rien à la configuration de Cobbler.
- Sur l'hyperviseur, au moment où la VM démarre, dans un terminal exécuter :
  virsh console « nom de la VM » > boot.txt
- Une fois le déploiement terminé, nous pouvons ouvrir le fichier texte et voir les différentes étapes de l'installation.

[17] http://blather.michaelwlucas.com/archives/638