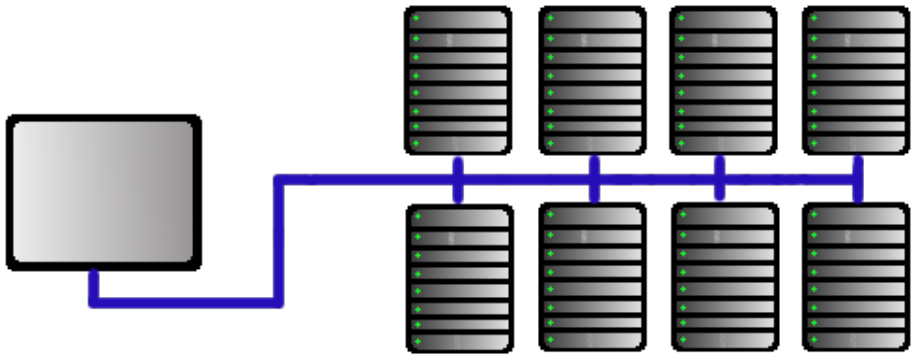
IUT de Nancy Charlemagne

Projet Tuteuré

Gestion centralisée de machines virtuelles

Auteurs :
Augustin Bocca
Mathieu Lamouroux
Sébastien Michaux
Julien Tournois

Tuteur :
Lucas Nussbaum



25 mars 2012

Résumé

Remerciements

Avant tout développement sur cette expérience, il apparait opportunt de commencer ce rapport par dess remerciements, à ceux qui nous on beaucoup appris au cours de cette période, et également à ceux qui ont eu la gentillesse de faire de ce projet un moment très profitable.
Aussi nous remercions Lucas Nussbaum, notre maître de stage qui nous a accompagné avec beaucoup de patience et de pédagogie.

Table des matières

Chapitre 1

Introduction

1.1 Présentation du projet

Mettre en place, évaluer et comparer différents outils permettant de gérer de manière centralisée et automatisée une infrastructure basée sur des machines virtuelles : Ganeti, OpenXenManager, virt-manager, Archipel...

1.2 Introduction à la virtualisation

1.2.1 Machine virtuelle

Une machine virtuelle est un conteneur isolé capable d'exécuter ses propres système d'exploitation et applications. Une machine virtuelle se comporte exactement comme un ordinateur physique et contient ses propres processeurs, mémoire RAM, disque dur et carte d'interface réseau virtuels.Une machine virtuelle a pour but de générer sur une même machine un ou plusieurs environnements d'exécution applicative. On en distingue deux types d'application : d'une part la virtualisation par le biais d'un hyperviseur jouant le rôle d'émulateur de système (PC ou serveur), d'autre part la virtualisation applicative qui permet de faire tourner un application sur un poste client quelque soit le système sous-jacent.

1.2.2 Hyperviseur

La machine virtuelle avec hyperviseur est utilisée pour générer au dessus d'un système d'exploitation serveur, une couche logicielle sous la forme d'un émulateur permettant de créer plusieurs environnements d'exécution serveur. Cet émulateur se place comme un niveau supplémentaire qui se greffe sur le système d'origine.

1.2.3 Au commencement, la virtualisation des mainframes

La virtualisation a été mise en œuvre pour la première fois il y a plus de 30 ans par IBM pour partitionner logiquement des mainframes en machines virtuelles distinctes. Ces partitions permettaient un traitement « multitâche », à savoir l exécution simultanée de plusieurs applications et processus. Étant donné que les mainframes consommaient beaucoup de ressources en même temps, le partitionnement constituait un moyen naturel de tirer pleinement parti de l investissement matériel.

1.2.4 Histoire de la virtualisation

La virtualisation est un concept qui a été mis au point pour la première fois dans les années 1960 pour permettre la partition dune vaste gamme de matériel mainframe et optimiser l'utilisation du matériel. De nos jours, les ordinateurs basés sur l'architecture x86 sont confrontés aux mêmes problèmes de rigidité et de sous-utilisation que les mainframes dans les années 1960. VMware a inventé la virtualisation pour la plate-forme x86 dans les années 1990 afin de répondre notamment aux problèmes de sous-utilisation, et a surmonté les nombreux défis émergeant au cours de ce processus. Aujourd'hui, VMware est devenu le leader mondial de la virtualisation x86 avec plus de 190 000 clients, dont la totalité des membres du classement Fortune 100.

1.2.5 Enjeux de la virtualisation

Actuellement, les entreprises rencontrent des besoins qui ne sont pas couverts.

Au niveau de la sécurité, les entreprises souhaiteraient isoler les services sur des serveurs différents. Pour la maintenance, il serait utile d'améliorer des services tels que la disponibilité, la migration,la redondance,la flexibilité ou le temps de réponse. Il serait également bienvenu de tester, déléguer l'administration d'un système ...

Une des solutions pour répondre à ces besoins serait d'acquérir davantage de plateformes de travail.

La multiplications des serveurs pose cependant un certain nombre de problème, augmenter sans cesse son parc informatique est impossible pour plusieurs raisons :

- Tout d'abord au niveau écologique cela entrainerait un surplus de déchets électronique,une consommation d'energie directe et de l'énergie utilisée pour refroidir les salles serveur.
- Au niveau de la surface utilisée, les salles machine seraient vite encombrées, puis apparaitra des problèmes tel que la nuisance sonore, le manque de puissance pour alimenter les salles serveur.
- Au niveau économique les couts d'achat, de recyclage, de fonctionnement, de maintenance seraient trop chère. La mise en place de serveur de virtualisation est une solution pour résoudre ces problèmes.

Le but de la virtualisation est de donner un environnement système au programme pour qu'il croie être dans un environnement matériel. Pour cela, une machine virtuelle est utilisée. Ainsi, plusieurs environnements d'exécution sont créés sur une seule machine, dont chacun émule la machine hôte. L'utilisateur pense posséder un ordinateur complet pour chaque système d'exploitation alors que toutes les machines virtuelles sont isolées entre elles.

FIGURE 1.1 – Logo de Grid5000

1.3 Présentation de Grid5000

Aujourd'hui, grâce à Internet, il est possible d'interconnecter des machines du monde entier pour traiter et stocker des masses de données. Cette collection hétérogène et distribuée de ressources de stockage et de calcul a donné naissance à un nouveau concept : les grilles informatiques.

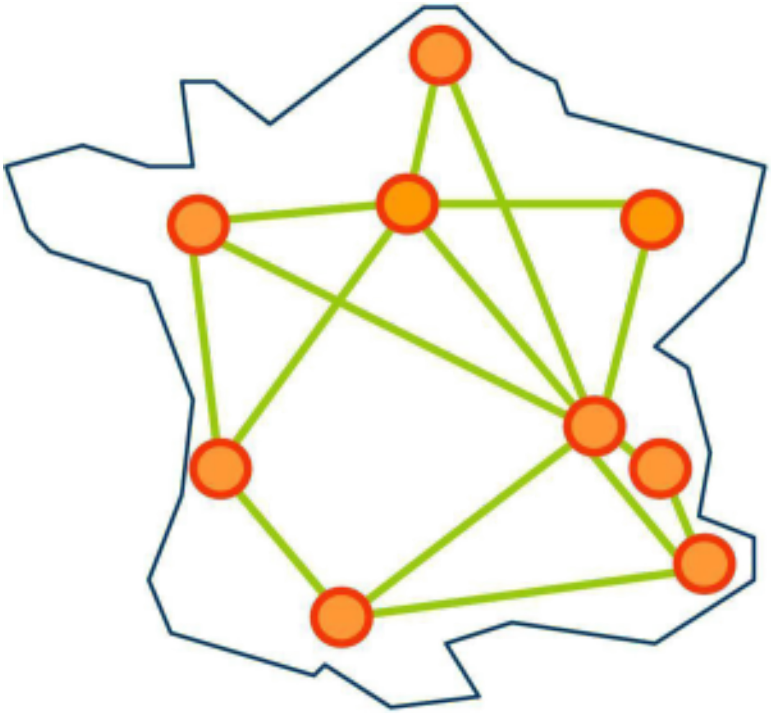
L'idée de mutualiser les ressources informatiques vient de plusieurs facteurs, évolution de la recherche en parallélisme qui, après avoir étudié les machines homogènes, s'est attaquée aux environnements hétérogènes puis distribués ; besoins croissants des applications qui nécessitent l'utilisation toujours plus importante de moyens informatiques forcément répartis.

La notion de grille peut avoir plusieurs sens suivant le contexte : grappes de grappes, environnements de type GridRPC (appel de procédure à distance sur une grille)., réseaux pair-à-pair, systèmes de calcul sur Internet, etc... Il s'agit d'une manière générale de systèmes dynamiques, hétérogènes et distribués à large échelle. Un grand nombre de problématiques de recherche sont soulevées par les grilles informatiques. Elles touchent plusieurs domaines de l'informatique :algorithmique, programmation, intergiciels, applications, réseaux.

L'objectif de GRID'5000 est de construire un instrument pour réaliser des expériences en informatique dans le domaine des systèmes distribués à grande échelle (GRID).

Cette plate-forme, ouverte depuis 2006 aux chercheurs de la communauté grille, regroupe un certain nombre de sites répartis sur le territoire national. Chaque site héberge une ou plusieurs grappes de processeurs. Ces grappes sont alors interconnectées via une infrastructure réseau dédiée à 10 Gb/s fournie par RENATER. À ce jour, GRID'5000 est composé de 9 sites : Lille, Rennes, Orsay, Nancy, Bordeaux, Lyon, Grenoble, Toulouse et Nice.

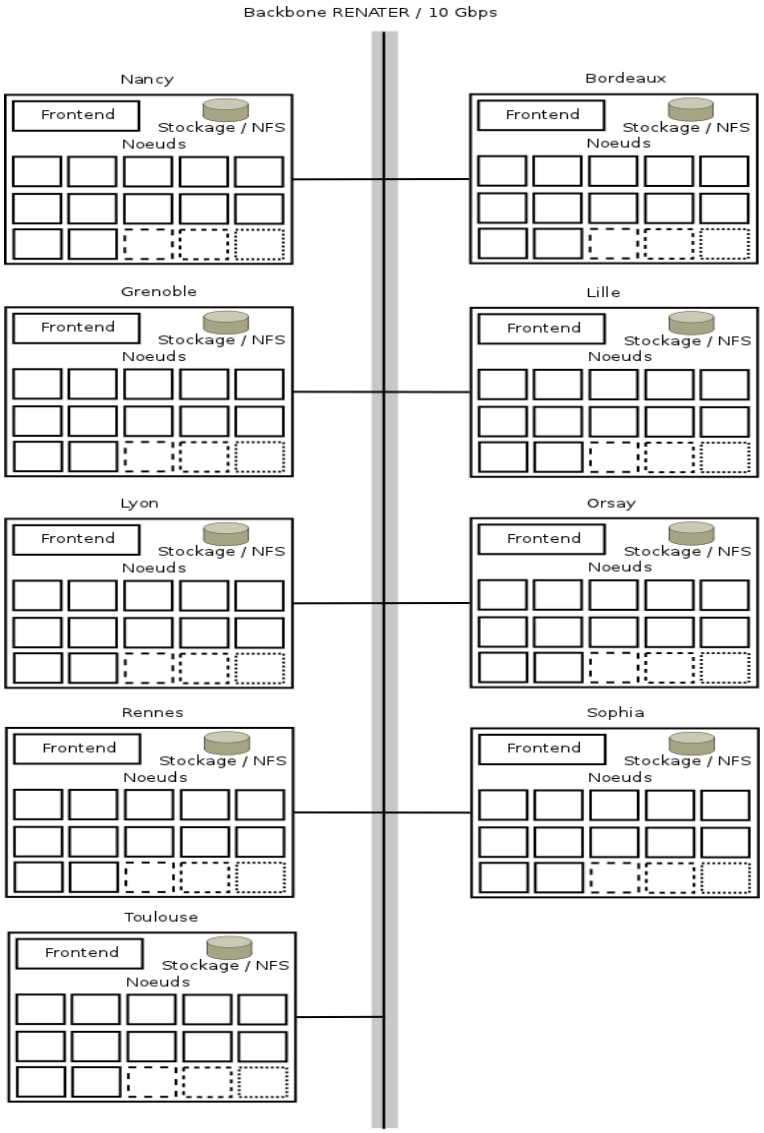
Début 2007, GRID'5000 regroupait plus de 2500 processeurs et près de 3500 cœurs.



Répartition des sites

1.3.1 Infrastructure des sites

Chaque site héberge :
- un frontend, serveur permettant d'accéder aux clusters disponibles ,
- un serveur de données, pour centraliser les données utilisateurs ,
- plusieurs clusters, c'est-à-dire des grappes de machines homogènes, appelées noeuds (nodes).



Architecture Grid5000

L'utilisateur de Grid 5000 accède à chaque site par son frontend en utilisant le protocole SSH.
Commande :

```
ssh utilisateur@access.grid5000.fr
```

Sur tous les serveurs du site, un répertoire home, local à chaque site, est monté avec NFS 2 . A partir du frontend, il est possible d'accéder aux machines des clusters en exectuant des réservations à l'aide de la commande :

```
oarsub
```

Gràce à notre tuteur, M. Lucas Nussbaum nous avons pu visiter la salle serveurs du site de Nancy située au Loria, ainsi qu'une présentation de la plate-forme (matériel utilisé, connexions réseau, administration).

Une description détaillée du site de Nancy est disponible sur le site de Grid 5000.

1.3.2 Réseau

Les sites et toutes les machines qu'ils comprennent sont interconnectés par RENATER 3 en 10Gbits/s. De plus, chaque site peut disposer de plusieurs réseaux locaux 4 :

– réseau en ethernet, 1 Gb/s
– réseaux hautes performances (Infiniband 20 Gb/s ou 10 Gb/s, et Myrinet 20 Gb/s)

1.3.3 Environnement logiciel

Tous les serveurs de Grid 5000 fonctionnent sous Debian GNU/Linux. A partir du frontend, l'utilisateur peut réserver des machines en utilisant la suite de logiciels OAR dédiée à la gestion de ressources de clusters, et déployer ses propres images de systèmes à l'aide des outils kadeploy. Il y a deux types de réservation :
– par défaut, pour des besoins de calcul avec OpenMPI ;
– pour le déploiement d'environnements (deploy).

1.4 Répartition des tâches

Nous avons commencé par prendre en main Grid5000 durant les 2 premières semaines du projet. Pour ce faire nous avons suivi avec soin les tutoriels mis à notre disposition sur le site www.grid5000.fr.

Une fois les manipulations de bases bien assimilées. Nous nous sommes divisés en 2 sous-groupes pour tester les différents outils du projet :
– Julien et Augustin se sont chargés de Ganeti et archipel.
– Sébatien et Mathieu pour OpenXenManager et virt-manager.

Chapitre 2

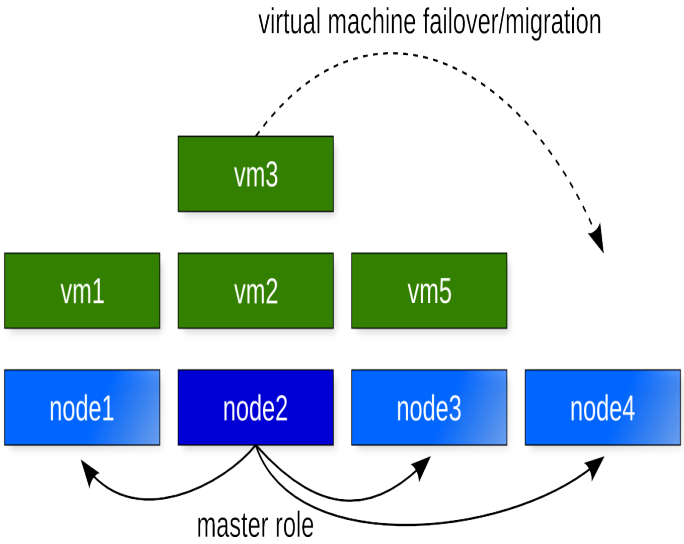
Ganeti

2.1 Introduction

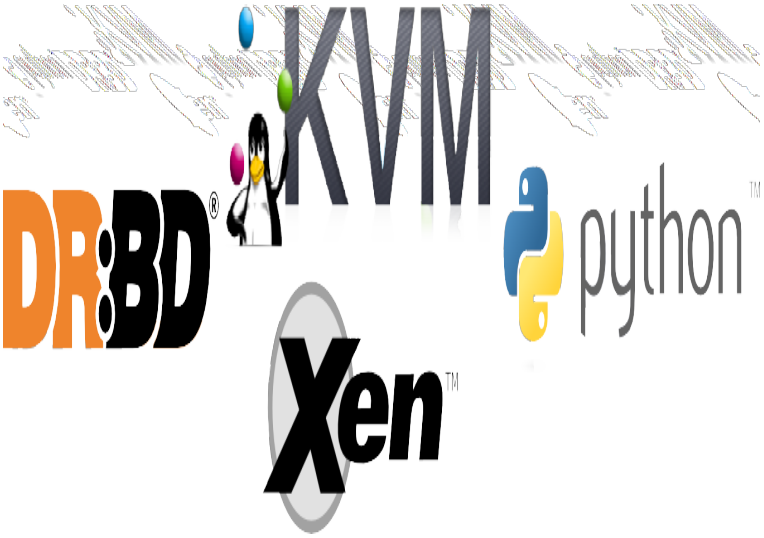
Ganeti est un outil de gestion de machines virtuelles se basant sur les technologies de virtualisation existantes comme XEN et KVM et LXC.
Ganeti nécessite un logiciel de virtualisation pré-installé sur les serveurs afin de pouvoir fonctionner. Une fois installé, l'outil prendra en charge la partie gestion des instances virtuelles (Xen DomU), par exemple, la gestion de création de disque, l'installation du système d'exploitation (en coopération avec les scripts d'installation du système d'exploitation spécifique), et le démarrage, l'arrêt, le basculement entre les systèmes physiques. Il a été conçu pour faciliter la gestion de cluster de serveurs virtuels et de fournir une récupération rapide et simple.

Ganeti est un un manager de cluster de machine virtuelles. Il combine la virtualization et la réplication en temps réel de disque. Ganeti offre un plateforme de haute disponibilité. Ce que Ganeti peut faire d'autre :

- Migration en "live" des instance
- Souplesse face aux pannes (Redondance des données avec DRBD)
- "Cluster balancing"
- Facilité pour les réparation et les changements matériel
- Possibilité de superviser simultanément entre 1 et environ 200 hôtes physiques.



Ganeti utilse Python, Xen, KVM, DRDB, LVM, SAN, socat et Haskell. Développé par Google depuis Aout 2007



Ce Projet est sous licence GNU GPLv2.
Site du projet :

```
1 http://code.google.com/p/ganeti/
```

Terminologie :
- Node : Un hôte physiques
- Instance : Un machine virtuelles
- Cluster : Un groupe de node supervisés
- Job : Une opération de ganeti

2.2 Installation

2.2.1 Modification des sources

Nous avons intallé ganeti à partir de la branche testing de debian. Pour des raisons techniques le système est squeeze. Pour cela il faut ajouter les sources de testing dans le fichier /etc/apt/sources.list :

```
##Wheezy
deb http://ftp.fr.debian.org/debian/ wheezy main contrib non-free
deb-src http://ftp.fr.debian.org/debian/ wheezy main contrib non-free

## wheezy security
deb http://security.debian.org/ wheezy/updates main contrib non-free
deb-src http://security.debian.org/ wheezy/updates main contrib non-free
```

IL faut ensuite créer le fichier de préférence de apt dans le répertoire /etc/apt/apt.conf.d. Nous avons appelé le fichier 80default-distrib (le nom du fichier est libre). Il faut ajouter cette ligne au fichier qui défini la branche stable comme la branche par défaut :

```
APT::Default-Release "stable";
```

2.2.2 Mise à jour et installation

On peut enfin alors mettre le système à jour et installer ganeti :

```
apt-get update && apt-get dist-upgrade -q -y --force-yes
apt-get -t testing install -q -y --force-yes ganeti2 ganeti-htools ganeti-instance-
    debootstrap
```

2.3 Configuration

La configuration est l'étape la plus complexe.

2.3.1 Configuration du fichier hosts

Dans le fichier hosts il faut renseigner l'adresse et le nom complet du node primaire de cette manière :

```
172.16.68.10   griffon-10.nancy.grid5000.fr griffon-10
```

2.3.2 Copie des fichier du noyau

Dans /etc/boot copier les fichiers vmlinuz-2.6.32-5-xen-amd64 et initrd.img-2.6.32-5-xen-amd64 :

```
cp vmlinuz-2.6.32-5-xen-amd64 vmlinuz-2.6-xenU
cp initrd.img-2.6.32-5-xen-amd64 initrd.img-2.6-xenU
```

2.3.3 Création du bridge xen-br0

Bien que nous utilisions eth0 comme bridge, la configuration de xen-br0 dans le fichier /etc/network/interfaces est obligatoire pour l'initialisation du cluster.

2.3.4 Création du LVM

Ganeti requière un LVM d'au moins 20Go pour fonctionner.
Sur les neud de grid5000 il est possible de créer un tel LVM sur la partition /dev/sd5.

```
umount /dev/sda5
pvcreate /dev/sda5
vgcreate xenvg /dev/sda5
```

On a créé un VG qui se nome xenvg sur /dev/sda5.

2.3.5 Edition de /usr/share/ganeti/os/debootstrap/common.sh

Il est nécessaire d'éditer ce fichier pour que ganeti puisse créer des instances :

Par defaut le mirroir utilisé par ganeti est http ://ftp.us.debian.org/debian/. Sur Grid5000 les depots US sont bloqués. Il faut donc indiquer les depots francais. On indique aussi l'adresse du proxy de grid5000. On peut choisir la version de Debian que l'on souhaite installer. Ici nous avons opter pour squeeze. L'architecture que nous choisi est amd64, car l'hote la supporte. La variable : EXTRA_PKGS permet d'installer des paquets supplémentaires.

2.3.6 Configuration et initialisation du cluster

L'initialisation du cluster se fait avec la commande gnt-cluster init clusterX

```
#initialisation du cluster
gnt-cluster init --no-drbd-storage --nic-parameters link=eth0 cluster1
```

Ici nous avons précisé les options –no-drbd-storage –nic-parameters link=eth0. La première permet d'utiliser ganeti sans utiliser la haute disponibilité. La seconde permet de préciser un autre bridge, et d'utiliser eth0 plutôt que xen-br0

Enfin il faut renseigner le inird et le root_path, cela est nécessaire pour la création des instances :

```
gnt-cluster modify --hypervisor-parameter xen-pvm:initrd_path='/boot/initrd.img-2.6-xenU
    '
gnt-cluster modify --hypervisor-parameter xen-pvm:root_path='/dev/xvda1'
```

2.4 Utilisation des nodes

2.4.1 Ajouter un node

Il est possible d'ajouter un node à tout moment :

```
root@griffon-8: gnt-node add griffon-78.nancy.grid5000.fr

-- WARNING --
Performing this operation is going to replace the ssh daemon keypair
on the target machine (griffon-78.nancy.grid5000.fr) with the ones of the current one
and grant full intra-cluster ssh root access to/from it

Unable to verify hostkey of host griffon-78.nancy.grid5000.fr:
30:cb:8f:ec:16:6a:3b:f5:0c:2a:de:a6:4c:1d:00:19. Do you want to accept
it?
y/[n]/?: y
2012-03-12 07:38:09,239: MainThread Authentication to griffon-78.nancy.grid5000.fr via
    public key failed, trying password
root password:
Mon Mar 12 07:38:16 2012 - INFO: Node will be a master candidate
```

2.4.2 Reconfigurer un node

Il aussi possible de reconfigurer un node deja présent :

```
root@griffon-8: gnt-node add --readd griffon-78.nancy.grid5000.fr

Unable to verify hostkey of host griffon-78.nancy.grid5000.fr:
6c:10:44:28:e2:2c:fc:7f:d4:5e:a3:bd:83:2c:b2:97. Do you want to accept
it?
```

```
y/[n]/?: y
Mon Mar 12 07:39:36 2012 - INFO: Readding a node, the offline/drained flags were reset
Mon Mar 12 07:39:36 2012 - INFO: Node will be a master candidate
```

2.4.3 Roles des nodes et opérations

Les différents nodes ainsi que leurs rôles :

- Master node : Utilise ganeti-masterd, rapi, noded and confd. Peut accueillir des instances. Toutes les opération de supervision s'effectuent sur ce node.
- Master candidates : Possède un copie complete de la configuration, peut prendre le rôle de Master.Utilise ganeti-confd and noded. Peut accueillir des instances.
- Regular node : Ne peuvent pas devenir Master et ne possedent qu'une partie de la configuration. Peut accueillir des instances.
- Offline node : Ces nodes sont hors-ligne. Ne peut pas accueillir des instances.

Promouvoir un neud au rang de master :

Il faut d'abord revoquer le rang de master du node principal, sur un node master-candidate :

```
gnt-cluster master-failover

root@griffon-81: gnt-instance list
Failure: prerequisites not met for this operation:
This is not the master node, please connect to node 'griffon-8.nancy.grid5000.fr' and
    rerun the command

root@griffon-81: gnt-cluster master-failover

root@griffon-81: gnt-node list
Node                        DTotal DFree MTotal MNode MFree Pinst Sinst
griffon-8.nancy.grid5000.fr 283.2G 283.2G 16.0G 965M 14.8G 0     0
griffon-78.nancy.grid5000.fr 283.2G 283.2G 16.0G 965M 14.8G 0    0
griffon-81.nancy.grid5000.fr 283.2G 283.2G 16.0G 965M 14.8G 0    0
```

On voit que le node griffon-81 n'était pas master avant l'utilisation de la commande. Ensuite il est possible d'executer les commandes master.

Passer un neud en master-candidate :

```
root@griffon-81: gnt-node modify -C yes griffon-8.nancy.grid5000.fr
Modified node griffon-8.nancy.grid5000.fr
 - master_candidate -> True
 - drained -> False
```

Passer un node en status drained :

```
root@griffon-81: gnt-node modify -D yes griffon-8.nancy.grid5000.fr
Modified node griffon-8.nancy.grid5000.fr
 - master_candidate -> False
 - drained -> True
```

Passer un node en offline :

```
root@griffon-81: gnt-node modify -O yes griffon-8.nancy.grid5000.fr
Modified node griffon-8.nancy.grid5000.fr
 - master_candidate -> False
 - offline -> True
```

Passer un node en mode regular (remise à zero de tous les flags) :

```
root@griffon-81: gnt-node modify -O no -D no -C no griffon-8.nancy.grid5000.fr
Mon Mar 12 08:26:01 2012 - INFO: Ignoring request to unset flag master_candidate,
    already unset
Mon Mar 12 08:26:01 2012 - INFO: Ignoring request to unset flag drained, already unset
Mon Mar 12 08:26:01 2012 - INFO: Auto-promoting node to master candidate
Mon Mar 12 08:26:01 2012 - WARNING: Transitioning node from offline to online state
    without using re-add. Please make sure the node is healthy!
Modified node griffon-8.nancy.grid5000.fr
 - master_candidate -> True
 - offline -> False
```

Le node est de nouveau en master-candidate comme à l'origine.

2.4.4 Supprimer un node :

```
root@griffon-81: gnt-node list
Node                      DTotal DFree MTotal MNode MFree Pinst Sinst
griffon-8.nancy.grid5000.fr 283.2G 282.2G 16.0G 965M 14.7G 1     0
griffon-78.nancy.grid5000.fr 283.2G 283.2G 16.0G 965M 14.8G 0    0
griffon-81.nancy.grid5000.fr 283.2G 283.2G 16.0G 965M 14.8G 0    0

root@griffon-81: gnt-node remove griffon-78.nancy.grid5000.fr

root@griffon-81: gnt-node list
Node                      DTotal DFree MTotal MNode MFree Pinst Sinst
griffon-8.nancy.grid5000.fr 283.2G 282.2G 16.0G 965M 14.7G 1     0
griffon-81.nancy.grid5000.fr 283.2G 283.2G 16.0G 965M 14.8G 0    0
```

Le node griffon-78 à bien été effacer du cluster.

2.4.5 Manipulation du stockage :

Faire la liste des volumes sur lesquels sont les instances :

```
root@griffon-81: gnt-node volumes
Node                      PhysDev  VG    Name                                      Size
    Instance
griffon-81.nancy.grid5000.fr /dev/sda5 xenvg 4b328fb6-1cc9-4599-9523-2c6a8cf7b861.disk0
    1000M instance4
griffon-8.nancy.grid5000.fr /dev/sda5 xenvg 8a85c87b-39fa-4e88-886b-a60c8706cc65.disk0
    1000M instance2
griffon-8.nancy.grid5000.fr /dev/sda5 xenvg d84f3842-04d5-4951-aded-b5ea76f19681.disk0
    1000M instance1
griffon-8.nancy.grid5000.fr /dev/sda5 xenvg ddfea3fc-d7b1-4a06-ae37-3094b1f0de11.disk0
    1000M instance3
```

Il est possible de lancer une reparation sur les volume de stockage :

```
root@griffon-81:~# gnt-node repair-storage griffon-8.nancy.grid5000.fr lvm-vg xenvg
Mon Mar 12 09:56:23 2012 Repairing storage unit 'xenvg' on griffon-8.nancy.grid5000.fr
    ...
```

Cela équivau à vgreduce –removemissing.

2.5 Utilisation des instances

2.5.1 Ajouter une instance

```
root@graphene-11: gnt-instance add -n graphene-11.nancy.grid5000.fr -o debootstrap+
    default -t plain -s 1000 instance1

Sat Mar 10 16:52:35 2012 * disk 0, vg xenvg, name 99370e22-9421-40d6-8d9d-59bb6ecfa959.
    disk0
Sat Mar 10 16:52:35 2012 * creating instance disks...
Sat Mar 10 16:52:36 2012 adding instance instance1 to cluster config
Sat Mar 10 16:52:36 2012 - INFO: Waiting for instance instance1 to sync disks.
Sat Mar 10 16:52:37 2012 - INFO: Instance instance1's disks are in sync.
Sat Mar 10 16:52:37 2012 * running the instance OS create scripts...
Sat Mar 10 16:54:05 2012 * starting instance...

root@graphene-11: gnt-instance list
Instance Hypervisor OS              Primary_node              Status Memory
instance1 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr running 128M
```

L'instance est bien créer sur le neud. Il est possible à partir du maitre de créer des instances sur n'importe quels neud d'un cluster. Il aussi possible de créer une instance primaire sur un noeud et une instance secondaire sur un autre.

2.5.2 Supprimer une instance

```
root@graphene-11: gnt-instance remove instance1
This will remove the volumes of the instance instance1 (including
mirrors), thus removing all the data of the instance. Continue?
y/[n]/?: y

root@graphene-11: gnt-instance list
Instance Hypervisor OS Primary_node Status Memory
```

L'instance à bien été supprimée. Cette commande supprime l'instance quelque soit le ou les neud où elle a été créer.

Arret et demarrage d'une instance

```
root@graphene-11: gnt-instance list
Instance Hypervisor OS              Primary_node              Status Memory
instance1 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr running 128M
instance2 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr running 128M

root@graphene-11: gnt-instance shutdown instance2
Waiting for job 21 for instance2...

root@graphene-11: gnt-instance list
Instance Hypervisor OS              Primary_node              Status    Memory
instance1 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr running 128M
instance2 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr ADMIN_down -
```

```
root@graphene-11: gnt-instance startup instance2
Waiting for job 26 for instance2...

root@graphene-11: gnt-instance list
Instance Hypervisor OS              Primary_node               Status Memory
instance1 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr running 128M
instance2 xen-pvm  debootstrap+default graphene-11.nancy.grid5000.fr running 128M
```

Le status de instance2 est de nouveau "running" ce qui signifie qu'elle est en fonctionnement.

Interroger les instances :

```
root@graphene-11: gnt-instance info instance1

Instance name: instance1
UUID: 3c3bd5ac-a261-4cba-a7f3-6cc74e49ce4e
Serial number: 2
Creation time: 2012-03-10 17:06:56
Modification time: 2012-03-10 17:07:04
State: configured to be up, actual state is up
  Nodes:
    - primary: graphene-11.nancy.grid5000.fr
    - secondaries:
  Operating system: debootstrap+default
  Allocated network port: None
  Hypervisor: xen-pvm
    - blockdev_prefix: default (sd)
    - bootloader_args: default ()
    - bootloader_path: default ()
    - initrd_path: default (/boot/initrd.img-2.6-xenU)
    - kernel_args: default (ro)
    - kernel_path: default (/boot/vmlinuz-2.6-xenU)
    - root_path: default (/dev/sda1)
    - use_bootloader: default (False)
  Hardware:
    - VCPUs: 1
    - memory: 128MiB
    - NICs:
      - nic/0: MAC: aa:00:00:d8:c6:8a, IP: None, mode: bridged, link: xen-br0
  Disk template: plain
  Disks:
    - disk/0: lvm, size 1000M
      access mode: rw
      logical_id: xenvg/3fe11555-edcd-40dc-bf63-f3fb749825bb.disk0
      on primary: /dev/xenvg/3fe11555-edcd-40dc-bf63-f3fb749825bb.disk0 (254:0)
```

Cette commande édite les informations relatives à l'instance.

Import et export d'instances :

Export :

```
root@graphene-100: gnt-backup export -n graphene-143.nancy.grid5000.fr instance1
Sun Mar 11 14:06:04 2012 Shutting down instance instance1
Sun Mar 11 14:08:07 2012 Creating a snapshot of disk/0 on node graphene-100.nancy.
    grid5000.fr
Sun Mar 11 14:08:08 2012 Starting instance instance1
Sun Mar 11 14:08:09 2012 Exporting snapshot/0 from graphene-100.nancy.grid5000.fr to
    graphene-143.nancy.grid5000.fr
Sun Mar 11 14:08:13 2012 snapshot/0 is now listening, starting export
Sun Mar 11 14:08:17 2012 snapshot/0 is receiving data on graphene-143.nancy.grid5000.fr
Sun Mar 11 14:08:17 2012 snapshot/0 is sending data on graphene-100.nancy.grid5000.fr
```

```
Sun Mar 11 14:08:22 2012 snapshot/0 sent 14M, 2.8 MiB/s
Sun Mar 11 14:08:38 2012 snapshot/0 finished receiving data
Sun Mar 11 14:08:38 2012 snapshot/0 finished sending data
Sun Mar 11 14:08:38 2012 Removing snapshot of disk/0 on node graphene-100.nancy.grid5000
    .fr
Sun Mar 11 14:08:39 2012 Finalizing export on graphene-143.nancy.grid5000.fr
Sun Mar 11 14:08:40 2012 Removing old exports for instance instance1
```

L'instance à bien été exporté dans graphene-143.nancy.grid5000.fr

Il est tout à fait possible d'exporter une instance sans la redémarer en utilisant l'option : –noshutdown.

Import :

```
root@graphene-100: gnt-instance remove instance1
This will remove the volumes of the instance instance1 (including
mirrors), thus removing all the data of the instance. Continue?
y/[n]/?: y

root@graphene-100: gnt-instance list
Instance Hypervisor OS              Primary_node               Status Memory
instance2 xen-pvm  debootstrap+default graphene-100.nancy.grid5000.fr running 128M
instance3 xen-pvm  debootstrap+default graphene-143.nancy.grid5000.fr running 128M
```

Instance1 à été supprimée du cluster.

```
root@graphene-100: gnt-backup import -n graphene-100.nancy.grid5000.fr --src-node=
    graphene-143.nancy.grid5000.fr -t plain instance1
Sun Mar 11 14:22:29 2012 * disk 0, vg xenvg, name a4cc7447-5ed7-4417-b222-d33a0c2842a0.
    disk0
Sun Mar 11 14:22:29 2012 * creating instance disks...
Sun Mar 11 14:22:30 2012 adding instance instance1 to cluster config
Sun Mar 11 14:22:30 2012 - INFO: Waiting for instance instance1 to sync disks.
Sun Mar 11 14:22:30 2012 - INFO: Instance instance1's disks are in sync.
Sun Mar 11 14:22:30 2012 * running the instance OS import scripts...
Sun Mar 11 14:22:30 2012 Exporting disk/0 from graphene-143.nancy.grid5000.fr to
    graphene-100.nancy.grid5000.fr
Sun Mar 11 14:22:34 2012 disk/0 is now listening, starting export
Sun Mar 11 14:22:37 2012 disk/0 is receiving data on graphene-100.nancy.grid5000.fr
Sun Mar 11 14:22:37 2012 disk/0 is sending data on graphene-143.nancy.grid5000.fr
Sun Mar 11 14:22:42 2012 disk/0 sent 34M, 6.0 MiB/s, 19%, ETA 23s
Sun Mar 11 14:23:08 2012 disk/0 finished sending data
Sun Mar 11 14:23:14 2012 disk/0 finished receiving data
```

On importe instance1 depuis graphene-143.nancy.grid5000.fr.

```
root@graphene-100: gnt-instance list
Instance Hypervisor OS              Primary_node               Status    Memory
instance1 xen-pvm  debootstrap+default graphene-100.nancy.grid5000.fr ADMIN_down -
instance2 xen-pvm  debootstrap+default graphene-100.nancy.grid5000.fr running 128M
instance3 xen-pvm  debootstrap+default graphene-143.nancy.grid5000.fr running 128M
```

L'instance à bien été importée.

```
root@graphene-100: gnt-instance startup instance1
Waiting for job 32 for instance1...

root@graphene-100: gnt-instance list
Instance Hypervisor OS              Primary_node               Status Memory
instance1 xen-pvm  debootstrap+default graphene-100.nancy.grid5000.fr running 128M
instance2 xen-pvm  debootstrap+default graphene-100.nancy.grid5000.fr running 128M
instance3 xen-pvm  debootstrap+default graphene-143.nancy.grid5000.fr running 128M
```

L'instance est fonctionnelle sur graphene-100.nancy.grid5000.fr

IL est aussi possible d'importer une instance étrangère à ganeti dont le disque est deja dans un LVM, sans avoir à le recopier.

```
gnt-instance add -t plain -n HOME_NODE ... \
  --disk 0:adopt=lv_name[,vg=vg_name] INSTANCE_NAME
```

Connexion à la console d'un instance :

```
gnt-instance console instance5
```

Une fois connecté à la console, l'utilisateur par defaut est root et il n'y pas de mots de passe.

Pour ce projet toutes nos instances sont des Debian 6.0. Evidemment il est possible de créer des instances autres que Debian. Nous avons créer les instances à partir de debootstrap, d'autres méthodes existe. On peut par exemple créer des instances à partir d'une image ISO. Ce qui permet d'avoir une grande variété d'instances.

Chapitre 3

OpenXENManager

3.1 Présentation

XenseMaking Project développe un client lourd, ainsi qu'un client web, pour manager XenServer. C'est un clone du XenCenter, qui fonctionne avec Linux, BSD, Windows et MacOSX, alors que le XenCenter ne fonctionne qu'avec Windows. OpenXenManager/OpenXenCenter un le client lourd qui permet de manager XenServer.Il a été développé en Python avec pygtk et gtk-vnc. Les fonctionnalités actuellement implémentées sont les suivantes :

- monitoring des machines virtuelles - accès à la console des machines virtuelles
- opérations d'administration (démarrage, arrèt, reboot, ...)
- création de machines virtuelles

3.2 Installation

Pour l'installation nous avons besoin des paquets suivant :

```
1 apt-get install subversion bzip2 python-glade2 python-gtk-vnc shared-mime-info graphviz
```

On télécharge la dernière version d'openxenmanager dans le dépot subversion

```
1 svn co https://openxenmanager.svn.sourceforge.net/svnroot/openxenmanager openxenmanager
```

On se déplace dans le répertoire trunk :

```
1 cd openxenmanager/trunk
```

Finalement on lance openxenmanager avec la commande suivante

```
1 python window.py
```

Une interface graphique d'openxenmanger apparait.

3.3 Utilisation

Pour l'instant nous n'avons toujours pas réussi à utiliser ce logiciel. DU a diverses difficultées (pas de service distant sous squeeze et plus de xen lors de la migration sous unstable).

Chapitre 4

Xen Cloud Platform

4.1 installation

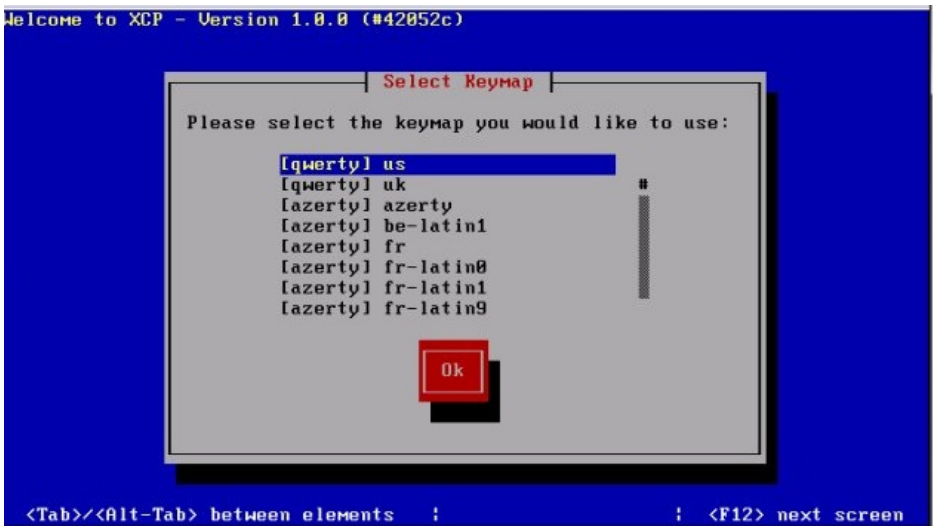


FIGURE 4.1 – On choisit le type de clavier

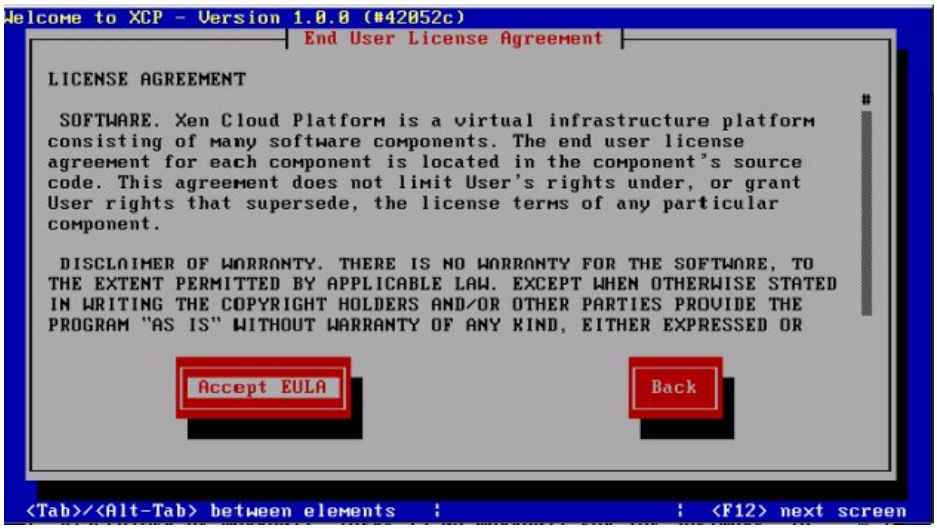


FIGURE 4.2 – On accepte la licence

FIGURE 4.3 – Choix du disque d'installation

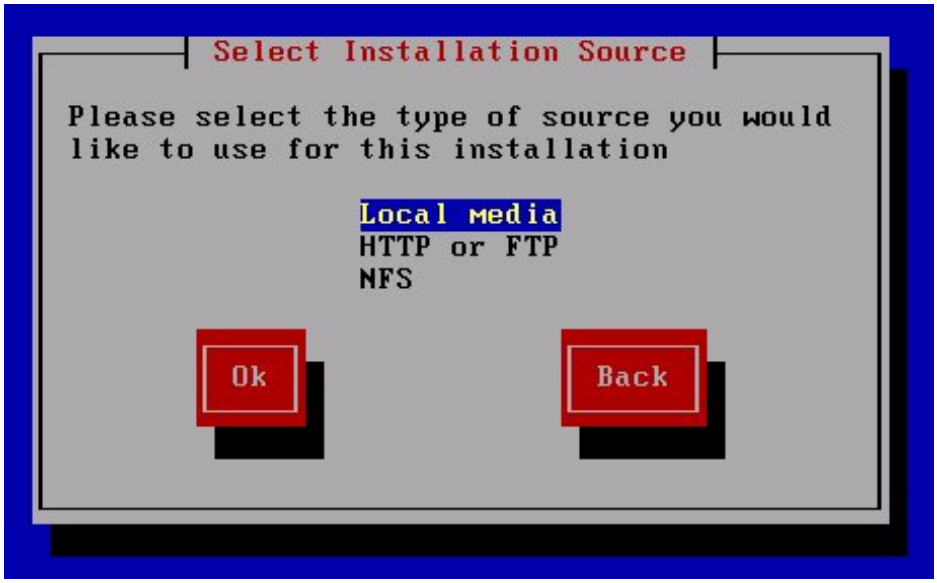


FIGURE 4.4 – Choix de la source d'installation

FIGURE 4.5 – Paquets additionnels

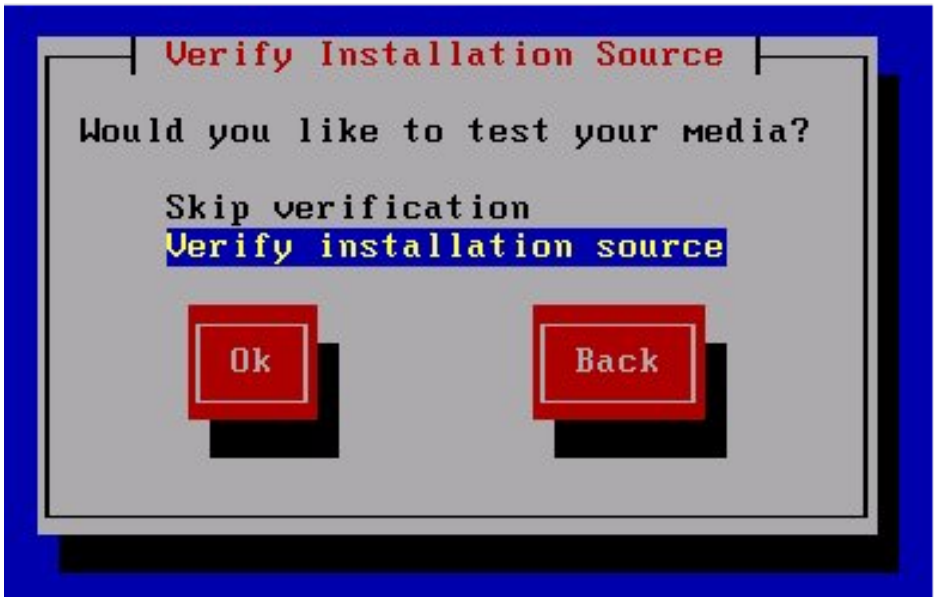


FIGURE 4.6 – Vérification de la source d'installation

FIGURE 4.7 – Choix du password

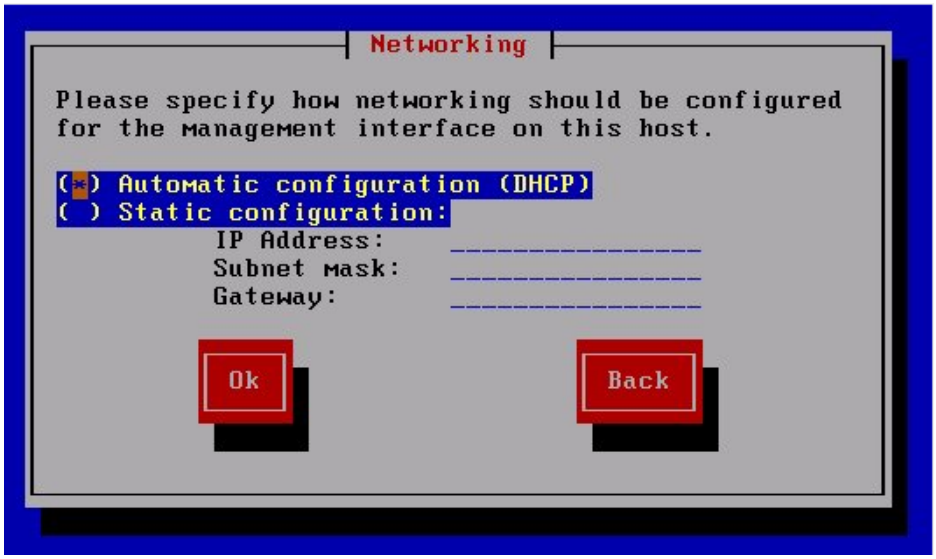


FIGURE 4.8 – Configuration du réseau

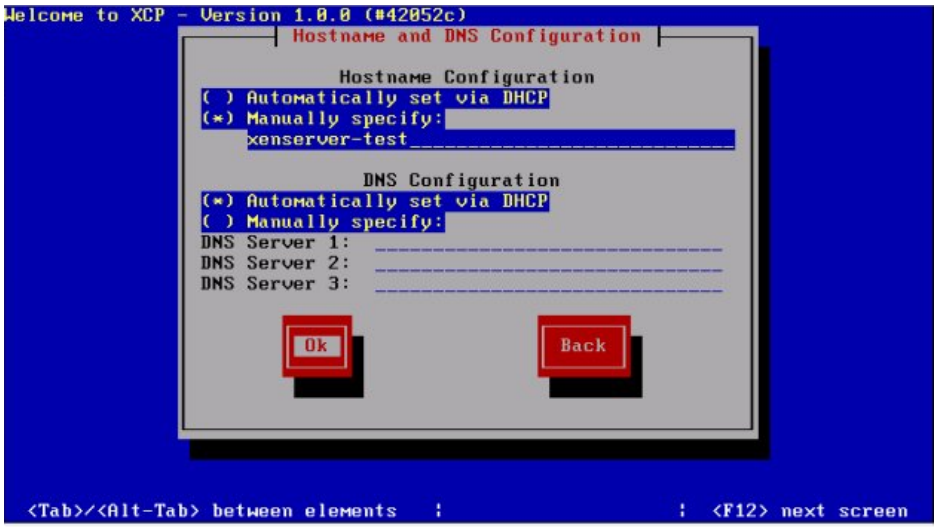


FIGURE 4.9 – Configuration du DNS

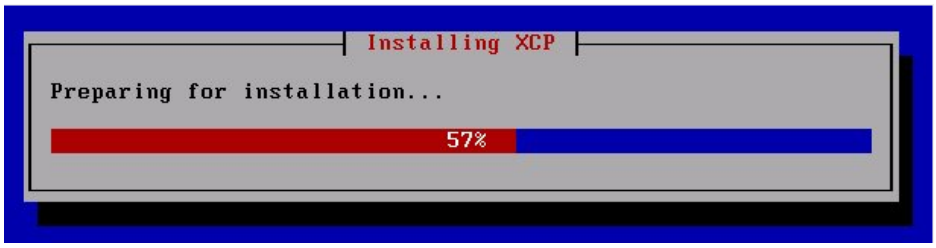


FIGURE 4.10 – Installation

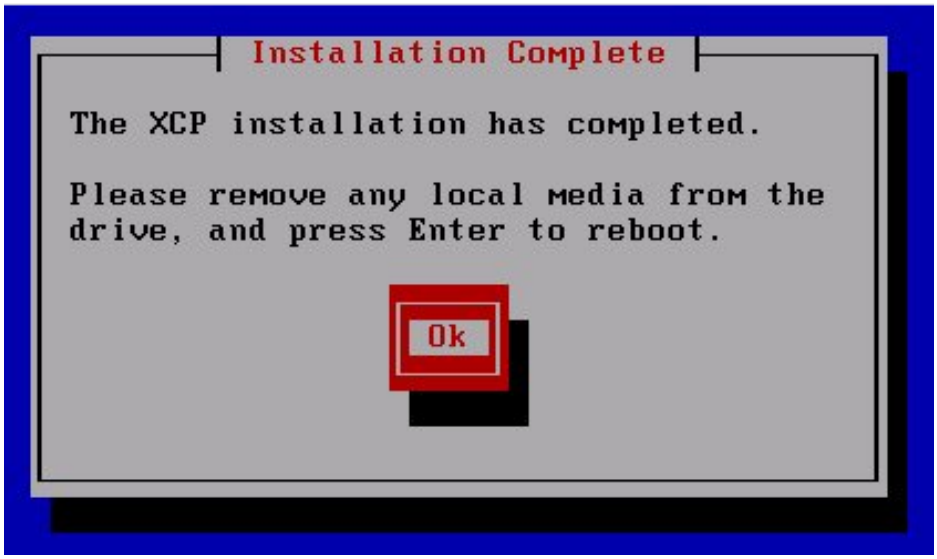


FIGURE 4.11 – Installation complète

Chapitre 5

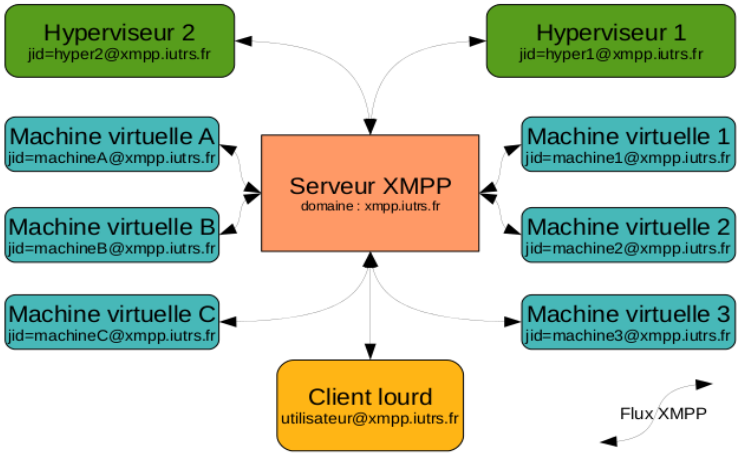
Archipel Project

5.1 Introduction

Archipel repose totalement sur libvirt comme API de gestion des machines virtuelles. De fait, l'outil s'impose d'être de plus haut niveau. Pour la gestion des utilisateurs, on peut directement utiliser un serveur XMPP existant pour authentifier les personnes.

5.2 Architecture

L'utilisation d'un serveur XMPP peut sembler étrange. L'application devra envoyer et recevoir des informations avec les hyperviseurs et les machines virtuelles. Plutôt que de réécrire un n-ième protocole, de l'implémenter, de devoir le débugger et le sécuriser, l'équipe d'Archipel avec XMPP a fait le choix de la réutilisation. Ce protocole est défini par des RFC, et possède de nombreuses implémentations.



Il suffit donc d'avoir un client adapté pour les hyperviseurs et pour les machines virtuelles. Les machines virtuelles, à un instant donné étant localisées sur un serveur, c'est au niveau de l'hyperviseur que reposera la création des n instances de clients pour chacune des n machines virtuelles. Au niveau de l'application de l'utilisateur, il y a également un client XMPP pour que les messages soient transmis aux entités.
En terme logiciel, un agent est installé sur chaque hyperviseur. Cet agent est scindé en deux parties :

– une incluant un client XMPP chargé de se connecter au serveur XMPP pour lui signaler sa présence.
– l'autre, plus complexe, est chargée de créer autant d'agents qu'il y a de machines virtuelles définies, chacun de ces agents intègre également un client XMPP.

Du coté du client utilisateur, l'interface intègre un client XMPP. Toutes les communications bénéficient donc des fonctionnalités du serveur XMPP, par exemple le roster (liste des contacts autorisés) qui donne une certaine sécurité dans les dialogues entre entités. On « ajoute » donc un contact dans son roster, qui peut être soit une machine virtuelle ou un hyperviseur. Lorsque dans l'interface client, on demande un démarrage de la VM (« Virtual Machine », en français machine virtuelle), un stanza (un message XMPP) est envoyé vers le serveur XMPP. Celui-ci vérifie que l'émetteur est dans le roster du destinataire, et lui transmettra alors le message s'il est autorisé. L'agent, coté machine virtuelle, va recevoir ce message et l'interpréter, donc lancer via l'API libvirt le signal start .
Les agents, qu'il s'agisse de ceux liés à un hyperviseur ou à une machine virtuelle, exploitent la librairie libvirt via le binding python. En particulier, c'est la partie events de libvirt qui est utilisée. De cette manière, même si une interaction parallèle est effectuée sur les hyperviseurs (par exemple, l'utilisation de virsh ou de virt-manager pour démarrer une entité, ajouter un nouveau réseau, ou créer une nouvelle VM par un script), les agents seront informés et pourront remonter le nouvel état via un message XMPP. Ce choix laisse la porte ouverte à une intégration fine dans un système existant : si vous aviez vos propres scripts virsh pour créer des machines virtuelles, l'adoption d'Archipel ne vous force nullement à renoncer à votre travail déjà en place !

5.3 Haute disponibilité et montée en charge

Dans les solutions de gestions d'hyperviseurs, un critère souvent retenu est la Haute Disponibilité (HA). Mais tout le monde ne met pas les mêmes éléments derrière ces mots. En terme de HA au niveau des machines virtuelles, Archipel se repose complètement et uniquement sur libvirt. Au niveau d'Archipel lui même, la haute disponibilité concerne l'échange des messages entre les entités, donc la haute disponibilité du serveur XMPP lui même. Une utilisation d'une solution cluster de serveurs XMPP répond donc à la demande. Ce mode cluster permet aussi de se rassurer au niveau de la montée en charge du serveur car la multiplication des VM et des hyperviseurs va augmenter le nombre de stanza échangés.
Au niveau des messages échangés, la notification de toutes les entités n'est pas nécessaire, ni voulue, pour des raisons de charge de messages. Une solution consisterait à stocker du coté de l'agent, les entités et les messages qu'elles doivent recevoir, mais il faudrait alors maintenir des états. L'architecture a recours à un mécanisme très intéressant du coté du serveur XMPP. Il s'agit de la partie PubSub (PUBlish- SUBscribe) : cela fonctionne comme les groupes de multicast au niveau réseau IP. Les entités qui souhaitent diffuser des informations à plusieurs entités publient via une entrée dans le service PubSub du serveur, et les entités qui souhaitent recevoir les messages pour cette entrée s'y abonnent. Ainsi, c'est encore une fois le serveur XMPP qui va faire le travail de diffuser aux entités abonnées les messages. Cela donne un flux optimal de messages, et simplifie la programmation des agents.

5.4 Sécurité

Dans une telle plate-forme, la gestion de la sécurité est importante. L'utilisation de XMPP en tant que protocole permet déjà d'être rassuré sur l'intégrité des messages échangés : XMPP est issue d'un processus de validation et de standardisation via les RFC et est implémenté depuis de nombreuses années.
La sécurité qui permet de savoir quel utilisateur peut dialoguer avec une entité repose d'abord sur le roster de celle-ci, car comme le roster est stocké coté serveur XMPP, c'est sans doute la

meilleure solution qui puisse être retenue. Il est donc possible de définir clairement quel utilisateur peut dialoguer avec une machine virtuelle ou un hyperviseur donnés, et également de savoir qui a accès à une entité donnée. Archipel va encore plus loin dans la gestion de la sécurité. Comme ce sont des stanza qui définissent les demandes entre les entités, et que cela se traduit par des appels à l'API de libvirt, un filtrage coté agent est en place, il permet d'autoriser ou non certaines actions. Ainsi, actuellement, 110 rôles possibles sont définis et il est possible d'attribuer à un utilisateur, l'accès ou non à chacun de ces rôles. Par exemple, on peut donner à un webmestre, le droit d'accéder à la « console VNC » de sa machine virtuelle, ainsi que l'action start et stop. Ou à un développeur, le droit de prendre des instantanés d'une machine virtuelle, et de les restaurer.
Dans un soucis de séparer les utilisateurs des entités machines virtuelles et hyperviseurs, on peut aller encore plus loin (et s'intégrer encore mieux dans le système d'information). Les serveurs XMPP sachant dialoguer ensemble, via une autorisation explicite au niveau du serveur pour autoriser le S2S (Server to Server), il est possible d'utiliser un autre serveur XMPP existant pour les utilisateurs (comme cela, ils peuvent utiliser par exemple une authentification liée à leur ENT). Les messages transitent alors par les deux serveurs successivement avant d'arriver aux entités coté Archipel.
En terme de sécurité, là encore, Archipel se base sur l'existant, et ajoute de la finesse dans la délégations de droits.

5.5 Fonctionnalités

Archipel étant encore jeune, les fonctionnalités les plus évidentes ont été mises en place. Le système de module permet d'ajouter aisément de nouvelles fonctions. Tout ce qui est nécessaire à un travail quotidien existe.
Actuellement, la plupart des fonctions disponibles via l'interface virt-manager sont déjà en place : définition d'une nouvelle VM, manipulations du réseau et du stockage, accès à la console VNC, gestions des snapshots, etc... Les opérations de migration sont également prises en charge. Les nouvelles possibilités sont celles qu'on attend d'un logiciel de management d'hyperviseurs : reporting sur l'état de hyperviseur, création de nouvelles machines à partir de flux RSS (VMCast), planifications de taches, gestions des droits des différents utilisateurs, création d'une machine avec détection automatique du serveur le moins chargé, etc... Les développeurs sont pleins d'idées, il même est prévu, par exemple, un module de facturation !
La mise en place d'Archipel est très simple : il faut installer un serveur XMPP avec une configuration correcte, puis installer sur chaque agent, un agent écrit en python. Cet agent, une fois correctement configuré, va se cloner pour s'exécuter pour toutes les machines virtuelles, et va joindre le serveur XMPP. A partir de ce moment là, Archipel est exploitable. Les machines virtuelles déjà en place bénéficient des fonctionnalités d'Archipel (sauf au niveau de la gestion disque et migration). On ne peut pas faire beaucoup plus simple.
Et pour couronner le tout, l'application client n'est pas un client lourd standard. Tout le client est en fait une application HTML5, en Javascript. C'est ce qui déconcerte généralement les premiers utilisateurs : il n'y a pas de serveur web où l'application s'exécute. Il faut juste un serveur web pour stocker les fichiers (dans certains cas et avec le navigateur chrome uniquement, il est même possible d'exécuter l'application depuis les fichiers stockés en local). Une fois chargé, c'est uniquement le Javascript qui fait tourner tout le client. Le framework utilisé est Cappuccino, qui donne un aspect très MacOsX. Le client XMPP y est également intégré. Donc quand vous utilisez votre client, vous dialoguez directement vers le serveur jabber !

5.6 Conclusion

Archipel s'intègre bien dans le courant UNIX : Keep It Simple, Stupid. D'autres solutions préfèrent construire des monstres d'infrastructure. Se baser sur libvirt permet de s'assurer d'avoir la main via virsh ou virt-manager, en cas de soucis de client, d'agent, ou même si le serveur XMPP

pose problème, cela permet de dormir tranquille.
Cependant Archipel a encore quelques faiblesses. A force d'exploiter la robustesse de XMPP, peu de serveurs implémentent tout ce qui est nécessaire. Ejabberd est pour l'instant le seul serveur XMPP qui est recommandé. Archipel a également mis en évidence un certain nombre de bugs dans libvirt. En particulier la gestion de Xen et de Vmware dans libvirt reste problématique, et rend inopérant Archipel (la balle est dans le camp des développeurs de libvirt). Enfin Archipel propose déjà un certain nombre de fonctionnalités qui le rend largement utilisable, et il reste de la place pour l'innovation.
La mise en place n'affecte pas l'existant, elle est relativement aisée à effectuer, ce qui permet de faire une migration en douceur.
Enfin, Archipel répond a un besoin récurrent mais simple : pouvoir donner un accès restreint à certaines machines pour certains utilisateurs. La délégation de droits est simple à effectuer. De plus, avec une interface conviviale et sans client lourd, en HTML5, il n'y a pas de problème pour donner un accès à des non informaticiens.

5.7 scripts

```
#Ajout des sources testing pour installer la derniere version de archipel
echo "## wheezy security" >> /etc/apt/sources.list
echo "deb http://security.debian.org/ wheezy/updates main contrib non-free" >> /etc/apt/sources.list
echo "deb-src http://security.debian.org/ wheezy/updates main contrib non-free" >> /etc/apt/sources.list
echo " " >> /etc/apt/sources.list
echo "#wheezy" >> /etc/apt/sources.list
echo "deb http://ftp.fr.debian.org/debian/ wheezy main contrib non-free" >> /etc/apt/sources.list
echo "deb-src http://ftp.fr.debian.org/debian/ wheezy main contrib non-free" >> /etc/apt/sources.list

#Ajout du fichier de preference
echo "APT::Default-Release \"stable\";" > /etc/apt/apt.conf.d/80default-distrib

#installation des composants utils archipel
apt-get install -t testing ejabberd erlang-dev erlang-xmerl build-essential erlang-tools python-setuptools -y --force-yes

#configuration de ejabber
cd /usr/local/src/xmlrpc-1.13/src
make

cd /usr/local/src
cp -a xmlrpc-1.13 /usr/lib/erlang/lib/

cd /usr/local/src/ejabberd-modules/ejabberd_xmlrpc/trunk/
sh build.sh
cp ebin/ejabberd_xmlrpc.beam /usr/lib/ejabberd/ebin/

cd /usr/local/src/ejabberd-modules/mod_admin_extra/trunk/
sh build.sh
cp ebin/mod_admin_extra.beam /usr/lib/ejabberd/ebin/

#suppression du fichier de conf de base
rm /etc/ejabberd/ejabberd.cfg
cp /usr/local/src/ejabberd.cfg /etc/ejabberd/
```

```
#!/bin/bash

################################################
#------------Installation de Archipel------------#
#_______________________________________________#

#chargement des noeuds reserve
list_nodes="$HOME/script_base/list_nodes"
list_node=‘cat $HOME/script_base/list_nodes‘

#sauvegarde le 1er noeud qui sera le maitre
sed -n "1 p" $list_nodes > $HOME/archipel/archipel_manager
archipel_manager="$HOME/archipel/archipel_manager"
archipel_man=‘cat $HOME/archipel/archipel_manager‘

#sauvegarde des noeud esclaves
cat $list_nodes | grep -v $archipel_man > $HOME/archipel/archipel_slaves
archipel_slaves="$HOME/archipel/archipel_slaves"
archipel_slav=‘cat $HOME/archipel/archiel_slaves‘

echo $USER > username

echo "####Archipel Manager########"
cat $archipel_manager
echo "############################"
echo "#####Archipel Slaves########"
cat $archipel_slaves
echo "############################"

#copie des fichier nssaisaire  la configuration de archipel
while read line
do
    scp -r sources/* root@$line:/usr/local/src
    scp -r username root@$line:/usr/local/src
    scp -r config_archipel.sh root@$line:/usr/local/src
done < $list_nodes

#lancement de la configuration de archipel sur le noeud maitre
taktuk -l root -f $archipel_manager broadcast exec [ sh /usr/local/src/config_archipel.
    sh ]

#scp archipel_master.sh root@$archipel_man:/root/

#lancement du script d'installation de ganeti sur le manager
#taktuk -l root -f $archipel_manager broadcast exec [ sh archiepl_master.sh ]

#while read line
#do
#    scp ganeti_slave.sh root@$line:/root/
#    taktuk -l root -f $line broadcast exec [ sh archipel_slave.sh ]
#done < archipel_slaves
```

Chapitre 6

Virt-manager

6.1 Pré-requis et considérations pour les hôtes

Divers facteurs doivent être considérés avant de créer des hôtes virtualisés.

Performance

Les hôtes virtualisés doivent être déployé et configuré en fonction de leurs tâches prévues. Certains systèmes (par exemple, les hôtes ou sont hébergés des serveur de base de données) ont besoin de performances plus élevées que d'habitude ; Les hôtes peuvent exiger plus de CPU ou de mémoire attribué en fonction de leur rôle,et de l'utilisation futur qu'il pourrait avoir. projeté la charge du système.

Stockage

Certains hôtes peuvent avoir besoin d'une plus grande priorité d'accès au stockage, de disques plus rapides, ou peuvent exiger un accès exclusif à des zones de stockage. La quantité de stockage utilisée par les hôtes doit être régulièrement surveillée et prise en compte lors du déploiement et le maintien de stockage.

Mise en réseau et l'infrastructure du réseau

En fonction de notre environnement, certains hôtes pourraient exiger des liens réseau plus rapides que d'autres hôtes. La bande passante ou de latence sont souvent des facteurs à prendre en compte lors du déploiement et le maintenance des hôtes.

6.2 Installation côté serveur

Cette partie est facile, un simple apt-get install suffit. Nous installons le paquet qui communique avec Xen et remonte les informations au client virt-manager.

```
1 apt-get install libvirt-bin
```

Du coté de Xen, nous devons vérifier qu'il peut communiquer avec libvirt.

Libvirt accède aux données de Xen via un socket unix. La configuration consiste à activer cette option dans Xen et à relancer les services. Nous éviterons ainsi l'erreur *libvirtError : internal error failed to connect to xend* dont on trouve peu d'explication sur le net.
Un autre bug rencontré nécessite la correction d'un lien symbolique et l'installation de qemu.

```
1 # Installation de qemu-dm pour crer des machines virtuelles en mode full virtualis
2 apt-get install xen-qemu-dm-4.0 -y
3
```

```
# Correction d'un bug de qemu qui invalidait un lien symbolique
mkdir /usr/lib64/xen
mkdir /usr/lib64/xen/bin
cd /usr/lib64/xen/bin
ln -s /usr/lib/xen-4.0/bin/qemu-dm
```

On édite le fichier de configuration xen

```
nano /etc/xen/xend-config.sxp
```

on active, ou on rajoute la ligne suivante

```
(xend-unix-server yes)
```

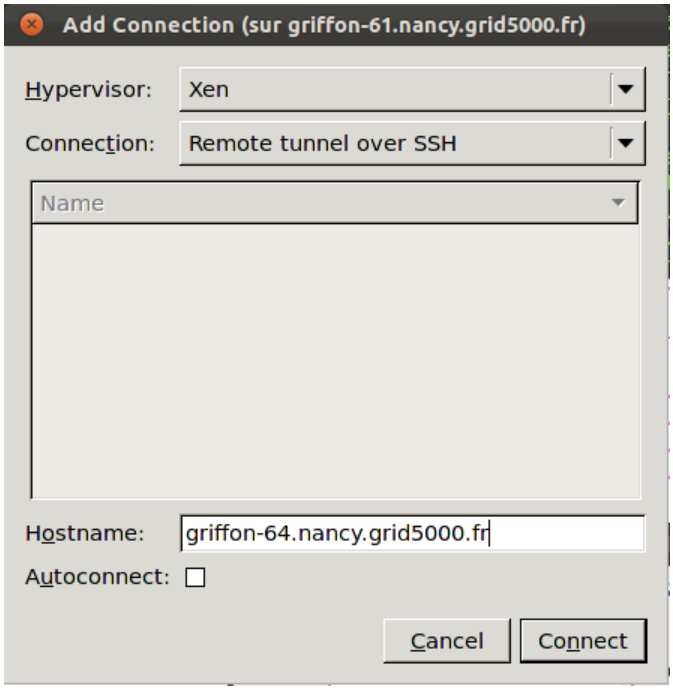
Enfin on relance le service xen avec /etc/init.d/xend restart

6.3 Installation côté client

Pour gérer nos serveurs, nous installons virt-manager avec la commande suivante :

```
apt-get install virt-manager
```

Au premier lancement, seule la connexion locale est active. Pour ajouter un nouveau noeud, on peut passer par le menu *File>Add Connection* auquel cas il faut modifier le type pour le positionner sur *xen+ssh* et renseigner le nomd du noeud.



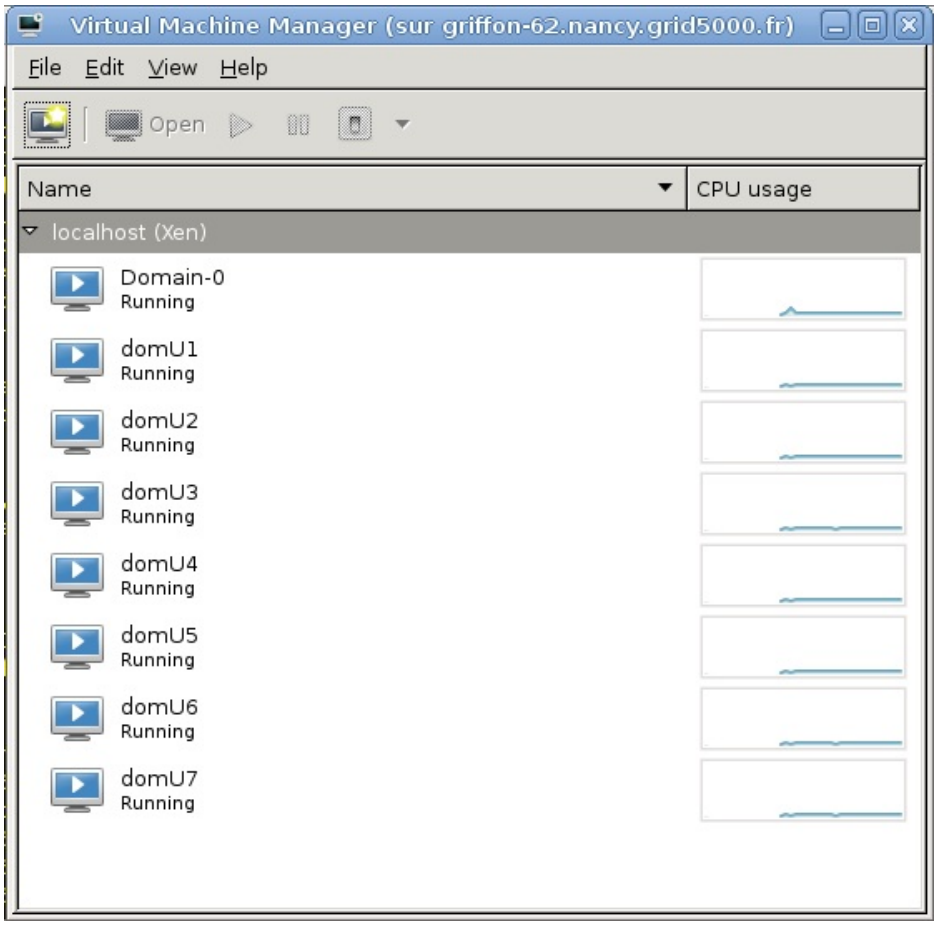
Virt-manager étant un outil graphique, ses fichiers de configurations sont gérés par gconf. Ainsi, pour automatiser l'ajout des différents noeuds nous avons crée un script qui réécrit le fichier */.gconf/apps/virt-manager/connections/%gconf.xml*

```
for node in $(cat $list_nodes)
do
    echo '<li type="string">' >> $fichier
    echo "<stringvalue>xen+ssh://root@$node/</stringvalue>" >> $fichier
    echo '</li>' >> $fichier
done
```

Ce fichier est au format xml et contient pour chaque noeud une entrée. Ici, les connexions sont gérées au-dessus d'un tunnel ssh. Commme nous ne pouvions pas accéder, depuis l'extérieur, aux noeuds de Grid5000, nous avons du installer virt sur les noeuds eux-mêmes (qui sont donc à la fois clients et serveurs) et y accéder via le X Forwarding de la connexion ssh.

6.4 Création d'hôtes virtualisés avec virt-manager

Pour commencer on démarre virt-manager, puis on lance le gestionnaire de machines virtuelles à partir du menu en cliquant sur l'icone en forme de pc.

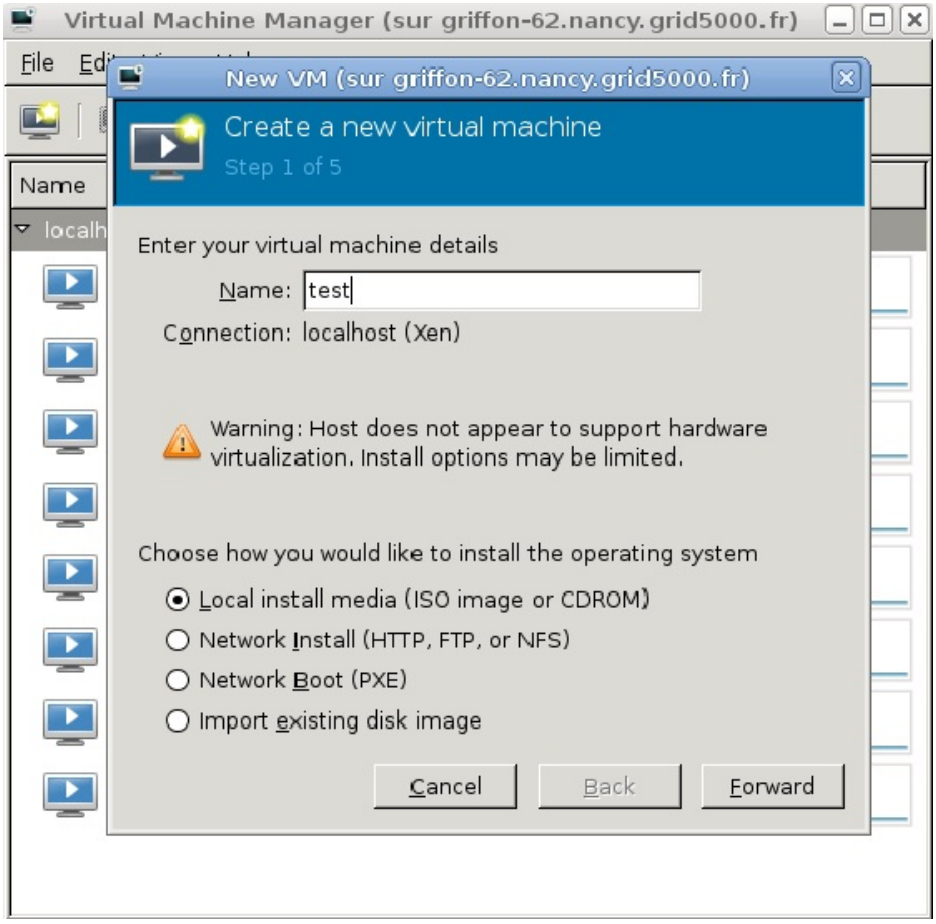


La fenêtre du gestionnaire de machine virtuelle nous autorise à en créer de nouvelles.
On clique sur création de nouvelle machine virtuelle pour faire apparaître l'assistant qui va nous aider pour élaborer notre hôte. L'assistant décompose la création en cinq étapes :

- La localisation et la configuration des supports d'installation.
- La configuration de la mémoire et les options de CPU.
- La configuration du stockage de l'invité.
- La configuration réseau, l'architecture, et d'autres paramètres matériels.

Le processus de création d'hôte commence avec la sélection d'un nom et le type d'installation.

Localisation et Configuration des supports d'installation.



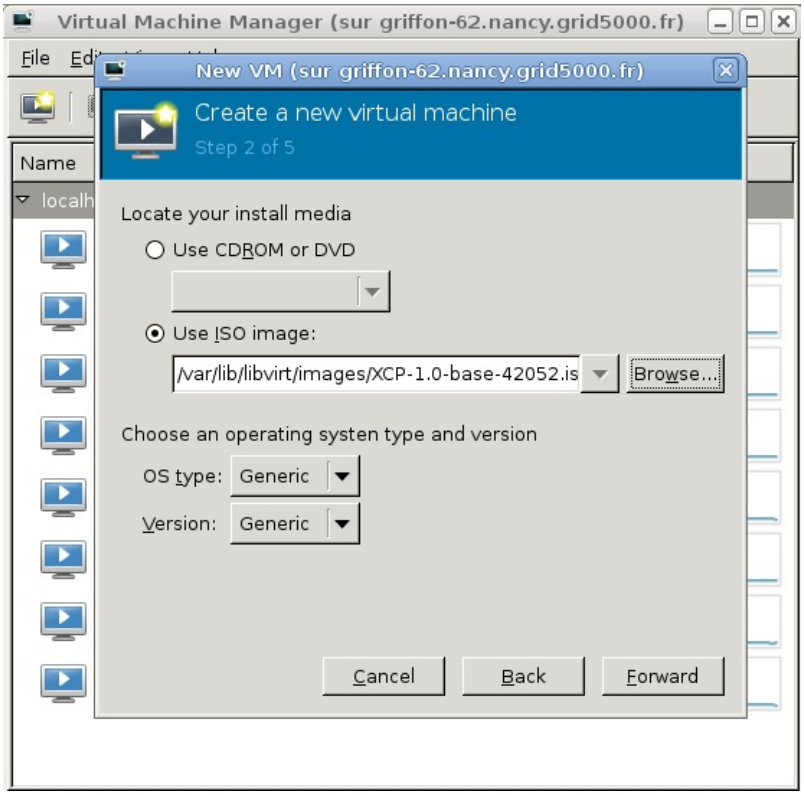
Local install media(ISO image or CDROM) : Cette méthode utilise un CD-ROM,DVD ou une image iso.

Network install : Cette méthode utilise le réseau pour installer le système d'exploitation.

Network Boot : Ceci permet de résupérer un environement exécutable directement depuis le réseau et donc sans installation nécessaire.

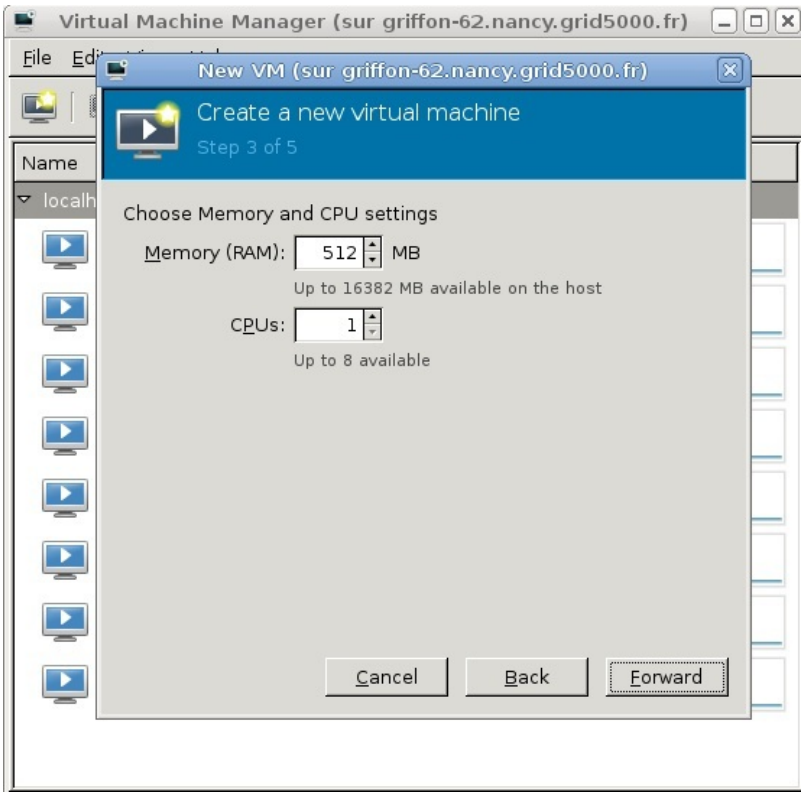
Import existing disk image : Cette méthode nous permet de créer un nouvelle hôte et d'y importer une image disque.

La prochaine étape consiste à configurer l'installation.
On configure le type de système d'exploitation et sa version qui sera installé, cela dépend de la méthode d'installation que l'on a choisie.

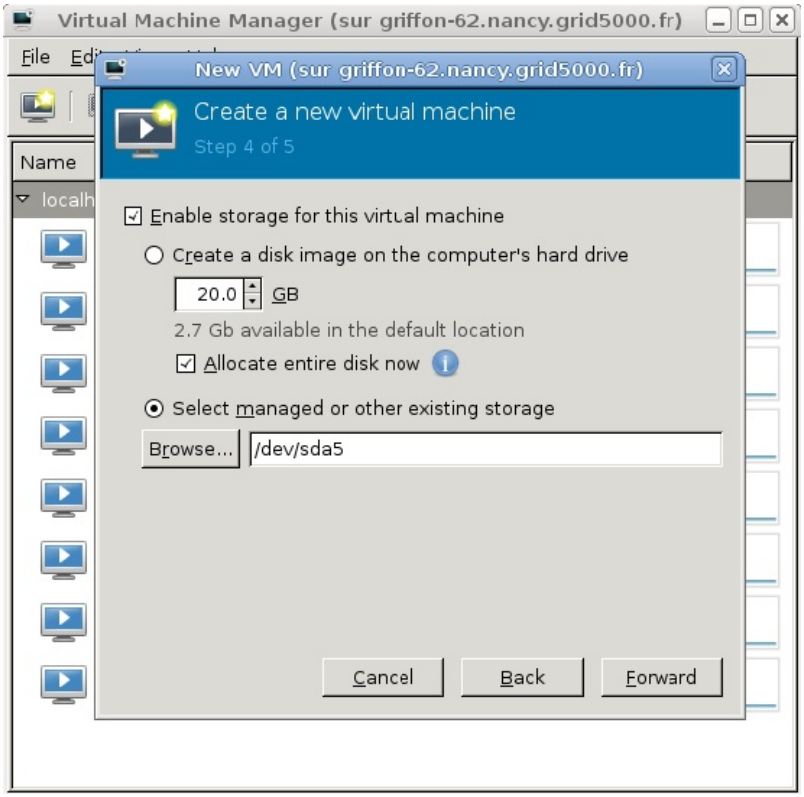


Configuration du CPU et de la mémoire

La prochaine étape consiste à configurer le nombre de CPU et la quantité de mémoire à allouer à la machine virtuelle. L'assistant indique le nombre de processeurs et la quantité de mémoire que l'on peut lui allouer.



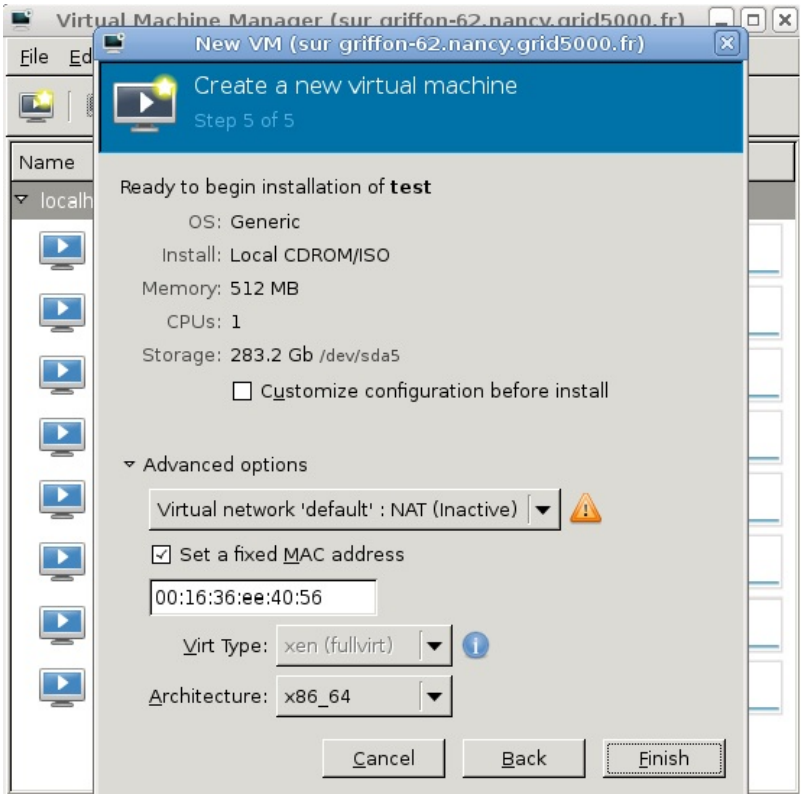
Configuration de l'espace de stockage



Si l'on a choisi d'importer une image de disque existante au cours de la première étape, virt-manager va sauter cette étape. On doit attribuer un espace suffisant pour notre machine virtuelle et toutes les applications que l'hôte a besoin.

Configuration réseau et architecture

On vérifie les paramètres de la machine virtuelle et on clique sur Terminer lorsqu'on est satisfait, cela permettra de créer l'hôte avec les paramètres réseau par défaut, le type de virtualisation, et l'architecture.



6.4.1 Controle d'hotes distants

On peut facilement manager des hotes distants une fois ceux-ci enregistrés dans virt-manager. Cette opération peut être fastidieuse car virt étant un programme graphique, crée un script pour automatiser l'ajout de plusieurs dizaines de noeuds s'est avéré plus compliqué que prévu. Cherchant désespérément un fichier de configuration, nous avons fini par le trouver en lançant une recherche dans tout le système sur les noms des noeuds ajoutés à virt. Le fichier est donc géré par *gconf* en voici un exemple

```
<?xml version="1.0"?>
<gconf>
  <entry name="autoconnect" mtime="1332579669" type="list" ltype="string">
    <li type="string">
      <stringvalue>xen:///</stringvalue>
    </li>
  </entry>
  <entry name="uris" mtime="1332579669" type="list" ltype="string">
    <li type="string">
      <stringvalue>xen:///</stringvalue>
    </li>
  </entry>
```

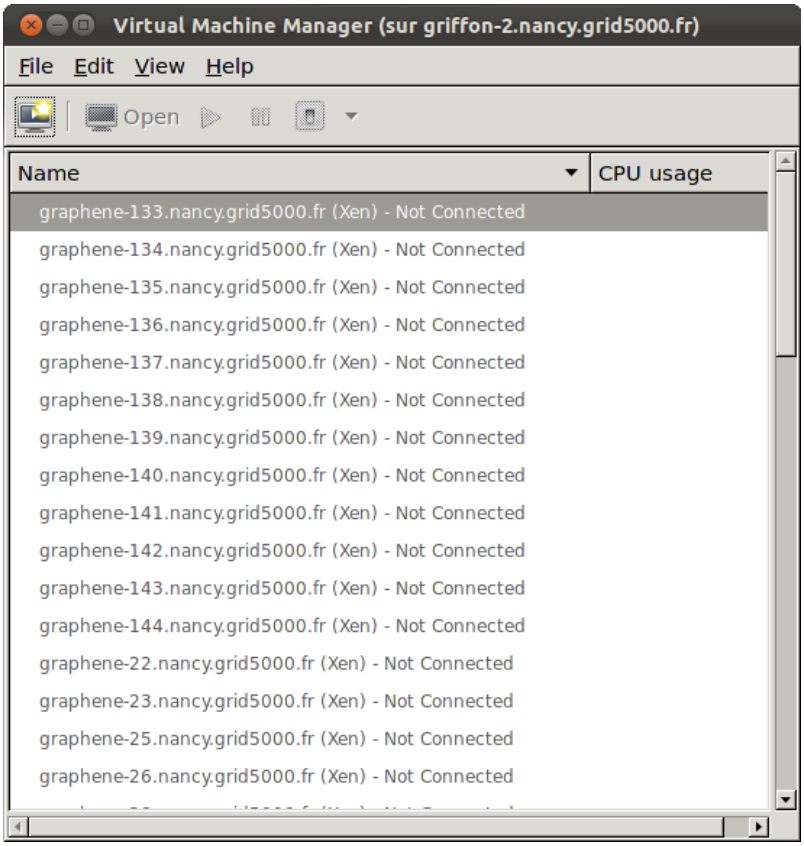
```
13 </gconf>
```

Listing 6.1 – Fichier .gconf/apps/virt-manager/connections/%gconf.xml

Après l'installation de virt, le seul lien enregistré est le lien local. Pour ajouter une connexion vers un nouvel hote xen, on peut utiliser l'interface graphique. Cependant, dans notre cas, il était assez fastidieux de renseigner manuellelement chacun des noeuds surtout lors des tests à grande échelle. C'est pourquoi nous avons fait un script (voir annexe B.10 en page 58) donc voici la boucle d'ajout des noeuds :

```
1 #Pour chaque noeud  rserv , on ajoute une entre dans le fichier
2 for node in $(cat $list_nodes)
3 do
4     echo '<li type="string">' >> $fichier
5     echo "<stringvalue>xen+ssh://root@$node/</stringvalue>" >> $fichier
6     echo '</li>' >> $fichier
7 done
```

Le script de base ayant configuré chacun des noeuds de la même manière, ils peuvent tous être utilisés pour gérer l'ensemble du réseau, une fois le script exécuté.



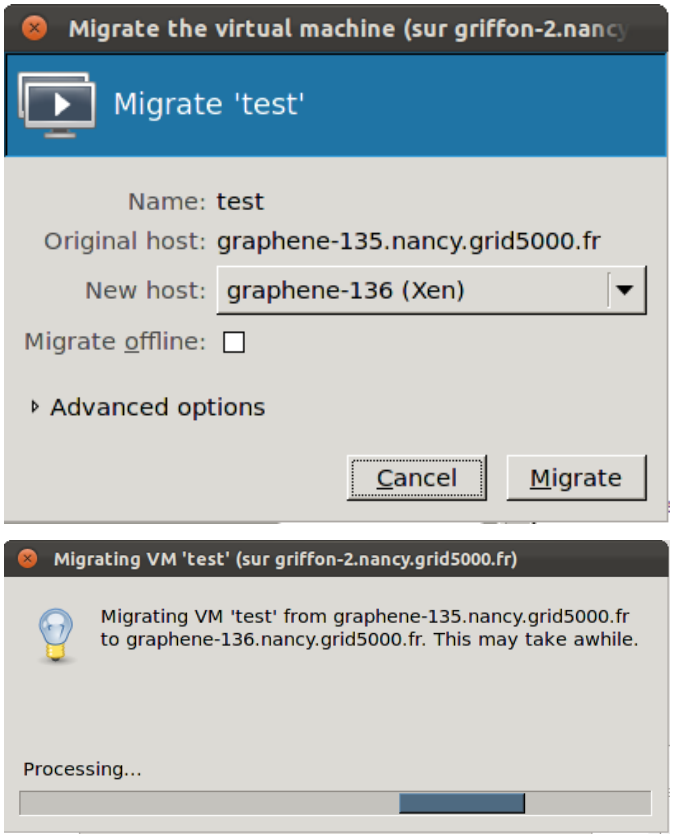
Un simple clic sur un noeud établit la connexion puisque les mêmes clefs ssh ont étées copiées partout et le fichier *known_hosts* a également été répliqué pour éviter de confirmer l'ajout d'un hôte inconnue (opération qui devait se faire en ligne de commande car non gérée par virt).
On est ensuite en mesure de créer une machine sur n'importe quel noeud, la procédure étant la même que pour une installation locale, nous n'allons pas la détailler à nouveau.

6.4.2 Migration de machines

Une fonctionnalité intéressate d'un gestionnaire de machines virtuelles est la migration des systèmes invités vers un autre hôte pour prévoir des opérations de maintenance, remplacement,...

Encore une fois virt-manger possède un outil graphique afin d'assister cette étape. Pour commencer, en effectuant un click droit sur une machine virtuelle on obitent un menu contextuel que nous allons détaillé au fur et à mesure.

En sélectionnant l'option *Migrate* l'assistant demande alors sur quel autre noeud (préalamblement ajouté à virt) nous souhaitons transférer la machine virtuelle. L'opération peut prendre plusieures minutes en fonctions des capacités des machines et du réseau.



Annexe A

Sources

www.grid5000.fr : le wiki disponible sur le site internet de grid5000 fut principale source de renseignements pour le démarrage du projet.

www.loria.fr/∼*lnussbau*/ : nous avons pu y consulter des anciens projets sur Grid5000 ce qui nous a permis d'avoir un premier aperçu de ses possibilités.

Annexe B

Scripts

B.1 Scripts spécifiques à l'environnement de Grid5000

```ruby
#!/usr/bin/ruby -w
# -*- coding: utf-8 -*-
puts "----------------------------------------"
puts "Souhaitez vous reserver des noeuds?(y/n)"
loop do
  test = gets.chomp
  if test.eql?("n")
    puts "######################"
    puts "#sortie du programme#"
    puts "######################"
    break;
  end
  if test.eql?("y")
    #Rservation de machines
    puts "---------Script de reservation-----------"
    puts "Choisir un nombre de noeud:"
    noeuds = gets.chomp.to_i
    puts "Choisir un temps de reservation(HH:MM:SS):"
    temps = gets
    puts "\nVous avez reserve #{noeuds} noeuds pour une duree de #{temps}"
    puts "----------------------------------------"
    puts "Voulez-vous mettre les noeuds dans un vlan separe(y/n)"
loop do
  test = gets.chomp
  if test.eql?("n")
    exec "oarsub -I -t deploy -n'virtu' -l slash_22=1+nodes=#{noeuds},walltime=#{temps}"
    break;
  end
      if test.eql?("y")
        puts "----------------------------------------"
        exec "oarsub -I -t deploy -n'virtu' -l {\"type='kavlan-local'\"}/vlan=1+/slash_22
            =1+nodes=#{noeuds},walltime=#{temps}"
        break;
      end
    end
    break;
  end
end
```

Listing B.1 – Réservation de noeuds

```ruby
# -*- coding: utf-8 -*-
#!/usr/bin/ruby -w

#Deployer une image cree
puts "Voulez vous deployer une image?(y/n)"
loop do
  test = gets.chomp
  if test.eql?("n")
    break;
  end
  if test.eql?("y")
    #choix de l'image
    puts "-----------------------------------------------"
    puts "image disponibles:"
    puts `ls | grep .env`
    puts "-----------------------------------------------"
    puts "Choix de la distibution(tout saisir):"
    debian = gets.chomp
    puts "-----------------------------------------------"
    puts "Voulez-vous deploye dans un vlan separe(y/n)"
    loop do
      test = gets.chomp
      if test.eql?("n")
        puts "----------------------------------------"
        exec"kadeploy3 -f $OAR_FILE_NODES -a #{debian} -k $HOME/.ssh/id_rsa.pub"
        break;
      end
      if test.eql?("y")
        puts "-----------------------------------------------"
        vlan = `kavlan -V`
        exec"kadeploy3 -f $OAR_FILE_NODES -a #{debian} --vlan #{vlan} -d -k $HOME/.ssh/
            id_rsa.pub"
        break;
      end
    end
    break;
  end
end

#########################################################

#Deploiment de l'environement sur les noeuds reserves
puts "Voulez vous deployer un environement?(y/n)"
loop do
  test = gets.chomp
  if test.eql?("n")
    puts "######################"
    puts "#sortie du programme#"
    puts "######################"
    break;
  end
  if test.eql?("y")
    #choix de la version a deployer
    puts "-----------------------------------------------"
    puts "distributions disponibles:"
    puts `kaenv3 -l | cut -d - -f1 | uniq | tail -n +3`
    puts "-----------------------------------------------"
    puts "Choix de la distibution:"
```

```
58      debian = gets.chomp
59      puts "-----------------------------------------------"
60      puts "version de la distribution:"
61      puts `kaenv3 -l | grep #{debian}| cut -d ' ' -f1`
62      debian = debian+"-x64-"
63      puts "-----------------------------------------------"
64      puts "Choix de la version de la distribution a deployee (sans #{debian}):"
65      version = gets.chomp
66      version = debian+version
67      puts "version utilisee:"
68      puts version
69      puts "-----------------------------------------------"
70      puts "Voulez-vous deploye dans un vlan separe(y/n)"
71      loop do
72        test = gets.chomp
73        if test.eql?("n")
74          #choix de la version  deployer
75          puts "-----------------------------------------------"
76          exec"kadeploy3 -e #{version} -f $OAR_FILE_NODES -k $HOME/.ssh/id_rsa.pub"
77          break;
78        end
79        if test.eql?("y")
80          puts "-----------------------------------------------"
81          vlan = `kavlan -V`
82          exec"kadeploy3 -e #{version} -f $OAR_FILE_NODES --vlan #{vlan} -k $HOME/.ssh/
                id_rsa.pub"
83          break;
84        end
85      end
86      break;
87    end
88  end
```

Listing B.2 – Déploiement des environnements sur les noeuds réservés

B.2 Scripts pour le déploiement de ganeti

```
#!/bin/bash

#######################################################
#------------------Configuration minimale--------------#
#______________________________________________________#

config_ssh () {
#ajoute les ligner permettant la connexion a un kvlan
Echo "Host *-*-kavlan-1 *-*-kavlan-1.*.grid5000.fr"
echo "ProxyCommand ssh -a -x kavlan-1 nc -q 0 %h %p"
echo "Host *-*-kavlan-2 *-*-kavlan-2.*.grid5000.fr"
echo "ProxyCommand ssh -a -x kavlan-2 nc -q 0 %h %p"
echo "Host *-*-kavlan-3 *-*-kavlan-3.*.grid5000.fr"
echo "ProxyCommand ssh -a -x kavlan-3 nc -q 0 %h %p"
}

#Verification et configuration SSH de l'utilisateur
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
##On regarde si le fichier .ssh existe
echo "---"
echo "Verification de la configuration ssh de l'utilisateur "$USER

trouver=0

if [ -f $HOME/.ssh/config ]
then
	trouver=1
	for line in $(cat $HOME/.ssh/config)
	do
		if [ $line = "kavlan-$vlan" ]
		then
			trouver=2
		fi
	done
else
	echo "Erreur. Aucun fichier, .ssh/config, n'a pas t trouv..."
	echo " Arrt  de l'installation."
	exit
fi
if [ $trouver -eq 1 ]
then
	echo "Ajout des lignes de 'transparence'"
	config_ssh >> $HOME/.ssh/config
fi

echo "-------------------------------------------"
echo "Vos noeuds sont dans un kavlan?(y/n)"
read choix
if [ $choix == "n" ]
then
	#suppression des know_host
  rm $HOME/.ssh/known_hosts
	#Recuperation des noeuds reserves
  cat $OAR_FILE_NODES | uniq > $HOME/script_base/list_nodes
  list_nodes="$HOME/script_base/list_nodes"

```

```
57    echo "-----------------------------------"
58    echo "Liste des machines reservee:"
59    cat $list_nodes
60    echo "-----------------------------------"
61    echo "Copie des clees SSH vers toutes les machines."
62    cat $HOME/.ssh/id_dsa.pub >> $HOME/.ssh/authorized_keys
63    for node in $(cat $list_nodes)
64    do
65        scp $HOME/.ssh/id_* root@$node:~/.ssh/
66             taktuk -l root -s -m $node broadcast exec [ 'cat ~/.ssh/id_dsa.pub >> ~/.ssh/
                  authorized_keys' ]
67    done
68    echo "-----------------------------------"
69
70  else
71      #recuperation des noeuds reserves
72    kavlan -l > list_nodes_kavlan
73    list_nodes_kavlan="$HOME/script_base/list_nodes_kavlan"
74
75    echo "-----------------------------------"
76    echo "Liste des machines reservee:"
77    cat $list_nodes_kavlan
78    echo "-----------------------------------"
79    echo "Copie des clees SSH vers toutes les machines."
80    for node in $(cat $list_nodes_kavlan)
81    do
82    scp $HOME/.ssh/id_rsa* root@$node:~/.ssh/
83    done
84    echo "-----------------------------------"
85
86    #changement des mdp root
87    taktuk -l root -f $list_nodes_kavlan broadcast exec [ 'echo -e "pttvirtu\npttvirtu" |
          passwd root' ]
88  fi
```

Listing B.3 – Configuration générale des noeuds

```
1   #
2
3   # Copyright (C) 2007, 2008, 2009 Google Inc.
4   #
5   # This program is free software; you can redistribute it and/or modify
6   # it under the terms of the GNU General Public License as published by
7   # the Free Software Foundation; either version 2 of the License, or
8   # (at your option) any later version.
9   #
10  # This program is distributed in the hope that it will be useful, but
11  # WITHOUT ANY WARRANTY; without even the implied warranty of
12  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
13  # General Public License for more details.
14  #
15  # You should have received a copy of the GNU General Public License
16  # along with this program; if not, write to the Free Software
17  # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
18  # 02110-1301, USA.
19
20  CLEANUP=( )
21
22  log_error() {
```

```
  echo "$@" >&2
}


get_api5_arguments() {
  GETOPT_RESULT=$*
  # Note the quotes around `$TEMP': they are essential!
  eval set -- "$GETOPT_RESULT"
  while true; do
    case "$1" in
      -i|-n) instance=$2; shift 2;;

      -o) old_name=$2; shift 2;;

      -b) blockdev=$2; shift 2;;

      -s) swapdev=$2; shift 2;;

      --) shift; break;;

      *)  log_error "Internal error!" >&2; exit 1;;
    esac
  done
  if [ -z "$instance" -o -z "$blockdev" ]; then
    log_error "Missing OS API Argument (-i, -n, or -b)"
    exit 1
  fi
  if [ "$SCRIPT_NAME" != "export" -a -z "$swapdev" ]; then
    log_error "Missing OS API Argument -s (swapdev)"
    exit 1
  fi
  if [ "$SCRIPT_NAME" = "rename" -a -z "$old_name" ]; then
    log_error "Missing OS API Argument -o (old_name)"
    exit 1
  fi
}

get_api10_arguments() {
  if [ -z "$INSTANCE_NAME" -o -z "$HYPERVISOR" -o -z "$DISK_COUNT" ]; then
    log_error "Missing OS API Variable:"
    log_error "(INSTANCE_NAME HYPERVISOR or DISK_COUNT)"
    exit 1
  fi
  instance=$INSTANCE_NAME
  if [ $DISK_COUNT -lt 1 -o -z "$DISK_0_PATH" ]; then
    log_error "At least one disk is needed"
    exit 1
  fi
  if [ "$SCRIPT_NAME" = "export" ]; then
    if [ -z "$EXPORT_DEVICE" ]; then
      log_error "Missing OS API Variable EXPORT_DEVICE"
    fi
    blockdev=$EXPORT_DEVICE
  elif [ "$SCRIPT_NAME" = "import" ]; then
    if [ -z "$IMPORT_DEVICE" ]; then
      log_error "Missing OS API Variable IMPORT_DEVICE"
    fi
    blockdev=$IMPORT_DEVICE
  else
```

```
    blockdev=$DISK_0_PATH
  fi
  if [ "$SCRIPT_NAME" = "rename" -a -z "$OLD_INSTANCE_NAME" ]; then
    log_error "Missing OS API Variable OLD_INSTANCE_NAME"
  fi
  old_name=$OLD_INSTANCE_NAME
}


format_disk0() {
  # Create one big partition, and make it bootable
  # some versions of sfdisk need manual specification of
  # head/sectors for devices such as drbd which don't
  # report geometry
  sfdisk -H 255 -S 63 --quiet --Linux "$1" <<EOF
0,,L,*
EOF
}

map_disk0() {
  blockdev="$1"
  filesystem_dev_base=`kpartx -l -p- $blockdev | \
                       grep -m 1 -- "-1.*$blockdev" | \
                       awk '{print $1}'`
  if [ -z "$filesystem_dev_base" ]; then
    log_error "Cannot interpret kpartx output and get partition mapping"
    exit 1
  fi
  kpartx -a -p- $blockdev > /dev/null
  filesystem_dev="/dev/mapper/$filesystem_dev_base"
  if [ ! -b "$filesystem_dev" ]; then
    log_error "Can't find kpartx mapped partition: $filesystem_dev"
    exit 1
  fi
  echo "$filesystem_dev"
}

unmap_disk0() {
  kpartx -d -p- $1
}

cleanup() {
  if [ ${#CLEANUP[*]} -gt 0 ]; then
    LAST_ELEMENT=$((${#CLEANUP[*]}-1))
    REVERSE_INDEXES=$(seq ${LAST_ELEMENT} -1 0)
    for i in $REVERSE_INDEXES; do
      ${CLEANUP[$i]}
    done
  fi
}

trap cleanup EXIT

DEFAULT_FILE="/etc/default/ganeti-instance-debootstrap"
if [ -f "$DEFAULT_FILE" ]; then
    . "$DEFAULT_FILE"
fi

# note: we don't set a default mirror since debian and ubuntu have
```

```
# different defaults, and it's better to use the default

# only if the user want to specify a mirror in the defaults file we
# will use it, this declaration is to make sure the variable is set
: ${MIRROR:="http://ftp.fr.debian.org/debian/"}
: ${PROXY:="http://proxy:3128/"}
: ${SUITE:="squeeze"}
: ${ARCH:="amd64"}
: ${EXTRA_PKGS:=""}
: ${CUSTOMIZE_DIR:="/etc/ganeti/instance-debootstrap/hooks"}
: ${VARIANTS_DIR:="/etc/ganeti/instance-debootstrap/variants"}
: ${GENERATE_CACHE:="yes"}
: ${CLEAN_CACHE:="14"} # number of days to keep a cache file
if [ -z "$OS_API_VERSION" -o "$OS_API_VERSION" = "5" ]; then
  DEFAULT_PARTITION_STYLE="none"
else
  DEFAULT_PARTITION_STYLE="msdos"
fi
: ${PARTITION_STYLE:=$DEFAULT_PARTITION_STYLE} # disk partition style

CACHE_DIR="/var/cache/ganeti-instance-debootstrap"

SCRIPT_NAME=$(basename $0)

if [ -f /sbin/blkid -a -x /sbin/blkid ]; then
  VOL_ID="/sbin/blkid -o value -s UUID"
  VOL_TYPE="/sbin/blkid -o value -s TYPE"
else
  for dir in /lib/udev /sbin; do
    if [ -f $dir/vol_id -a -x $dir/vol_id ]; then
      VOL_ID="$dir/vol_id -u"
      VOL_TYPE="$dir/vol_id -t"
    fi
  done
fi

if [ -z "$VOL_ID" ]; then
  log_error "vol_id or blkid not found, please install udev or util-linux"
  exit 1
fi

if [ -z "$OS_API_VERSION" -o "$OS_API_VERSION" = "5" ]; then
  OS_API_VERSION=5
  GETOPT_RESULT=`getopt -o o:n:i:b:s: -n '$0' -- "$@"`
  if [ $? != 0 ] ; then log_error "Terminating..."; exit 1 ; fi
  get_api5_arguments $GETOPT_RESULT
elif [ "$OS_API_VERSION" = "10" -o "$OS_API_VERSION" = "15" ]; then
  get_api10_arguments
else
  log_error "Unknown OS API VERSION $OS_API_VERSION"
  exit 1
fi

if [ -n "$OS_VARIANT" ]; then
  if [ ! -d "$VARIANTS_DIR" ]; then
    log_error "OS Variants directory $VARIANTS_DIR doesn't exist"
    exit 1
  fi
  VARIANT_CONFIG="$VARIANTS_DIR/$OS_VARIANT.conf"
```

```
  if [ -f "$VARIANT_CONFIG" ]; then
    . "$VARIANT_CONFIG"
  else
    if grep -qxF "$OS_VARIANT" variants.list; then
      log_error "ERROR: instance-debootstrap configuration error"
      log_error " Published variant $OS_VARIANT is missing its config file"
      log_error " Please create $VARIANT_CONFIG or unpublish the variant"
      log_error " (by removing $OS_VARIANT from variants.list)"
    else
      log_error "Unofficial variant $OS_VARIANT is unsupported"
      log_error "Most probably this is a user error, forcing a wrong name"
      log_error "To support this variant please create file $VARIANT_CONFIG"
    fi
    exit 1
  fi
fi
```

```
#!/bin/bash

################################################
#------------Installation de Ganeti-------------#
#_____________________________________________#

#chargement des noeuds reserve
list_nodes="$HOME/script_base/list_nodes"
list_node=‘cat $HOME/script_base/list_nodes‘

#sauvegarde le 1er noeud qui sera le maitre
sed -n "1 p" $list_nodes > $HOME/ganeti/ganeti_manager
ganeti_manager="$HOME/ganeti/ganeti_manager"
ganeti_man=‘cat $HOME/ganeti/ganeti_manager‘

#sauvegarde des noeud esclaves
cat $list_nodes | grep -v $ganeti_man > $HOME/ganeti/ganeti_slaves
ganeti_slaves="$HOME/ganeti/ganeti_slaves"
ganeti_slav=‘cat $HOME/ganeti/ganeti_slaves‘


echo "######Ganeti Manager########"
cat $ganeti_manager
echo "############################"
echo "######Ganeti Slaves########"
cat $ganeti_slaves
echo "############################"

#  rcupre  l’adresse du cluster1 pour l’envoyer au node x.x.x.1
g5k-subnets -i -o ip_list.txt
ipnode=‘head -1 ip_list.txt‘
echo $ipnode > ipcluster

#recupere l’ip de la gateway x.x.x.254
ipgateway=‘head -254 ip_list.txt | tail -1‘
echo $ipgateway > ipgateway

#  rcupre  l’ip du reseau
g5k-subnets -a > tempipreseau
ipnetwork=‘head -1 tempipreseau | cut -d’/’ -f1‘
echo $ipnetwork > ipnetwork

```

```
#envoi via ssh au node
while read node
do
    scp common.sh node-add.sh ganeti_slaves ganeti_master.sh ganeti_slave.sh ipgateway
        ipnetwork ipcluster tempipreseau ip_list.txt root@$node:/root/
done < $list_nodes

#suppression des fichiers
rm ipgateway ipnetwork ipcluster tempipreseau ip_list.txt

#lancement du script d'installation de ganeti sur le manager
taktuk -l root -f $ganeti_manager broadcast exec [ sh ganeti_master.sh ]

#idem sur les eclaves
taktuk -l root -f $ganeti_slaves broadcast exec [ sh ganeti_slave.sh ]

mkdir keys
cd keys
scp root@$ganeti_man:/root/.ssh/authorized_keys .
mv authorized_keys authorized
tail -n 1 authorized > authorized_keys && rm authorized
scp root@$ganeti_man:/root/.ssh/id_dsa* .
taktuk -l root -f $ganeti_manager broadcast exec [ rm /usr/lib/ganeti/tools/setup-ssh ]
scp setup-ssh root@$ganeti_man:/usr/lib/ganeti/tools/

while read slavenode
do
    scp id_dsa* authorized_keys root@$slavenode:/root/
done < $ganeti_slaves
rm id_dsa*
taktuk -l root -f $ganeti_slaves broadcast exec [ cp /root/authorized_keys /root/.ssh/ ]
```

Listing B.4 – Installation et configuration de ganeti

```
#!/bin/bash

##VERIFIER QUE postinstall.sh et ganeti.sh soient dans le meme dossier puis executer
    postinstall.sh (Sur le frontend)!

#Ajout des sources testing pour installer la derniere version de ganeti
echo "## wheezy security" >> /etc/apt/sources.list
echo "deb http://security.debian.org/ wheezy/updates main contrib non-free" >> /etc/apt/
    sources.list
echo "deb-src http://security.debian.org/ wheezy/updates main contrib non-free" >> /etc/
    apt/sources.list
echo " " >> /etc/apt/sources.list
echo "#wheezy" >> /etc/apt/sources.list
echo "deb http://ftp.fr.debian.org/debian/ wheezy main contrib non-free" >> /etc/apt/
    sources.list
echo "deb-src http://ftp.fr.debian.org/debian/ wheezy main contrib non-free" >> /etc/apt
    /sources.list

#Ajout du fichier de preference
echo "APT::Default-Release \"stable\";" > /etc/apt/apt.conf.d/80default-distrib

#Installation de ganeti
apt-get update && apt-get dist-upgrade -q -y --force-yes
apt-get -t testing install -q -y --force-yes ganeti2 ganeti-htools ganeti-instance-
    debootstrap
```

```
20
21 echo "Ajout du node dans /etc/hosts"
22 hostname=‘cat /etc/hostname‘
23
24 #recuperation des variables
25
26 #ip du node
27 ifconfig eth0 > troll
28 ipnode=‘head -2 troll | tail -1 | cut -d’:’ -f2 | cut -d’ ’ -f1‘
29 echo $ipnode " " $hostname >> /etc/hosts
30
31 #ip du broadcast
32 ipbroadcast=‘head -2 troll | tail -1 | cut -d’B’ -f2 | cut -d’:’ -f2 | cut -d’ ’ -f1‘
33
34 #ip du masque de sous rseau
35 ipmask=‘head -2 troll | tail -1 | cut -d’M’ -f2 | cut -d’:’ -f2 | cut -d’ ’ -f1‘
36
37 #ip du reseau
38 ipnetwork=‘head -1 ipnetwork‘
39
40 #ip de la passerelle
41 ipgateway=‘head -1 ipgateway‘
42
43 #ajout de cluster1 dans dans /etc/hosts
44 echo "ajout de cluster1 dans /etc/hosts"
45 ipcluster=‘cat ipcluster‘
46 echo $ipcluster "cluster1" >> /etc/hosts
47
48 #Dans /boot/ creer des liens symboliques :
49 cd /boot
50 cp vmlinuz-2.6.32-5-xen-amd64 vmlinuz-2.6-xenU
51 cp initrd.img-2.6.32-5-xen-amd64 initrd.img-2.6-xenU
52
53 #Pour le moment changera surement.
54 echo "creation du LVM"
55 umount /dev/sda5
56 pvcreate /dev/sda5
57 vgcreate xenvg /dev/sda5
58
59 #Creation du bridge xen-br0
60 echo " " >> /etc/network/interfaces
61 echo "auto xen-br0" >> /etc/network/interfaces
62 echo "iface xen-br0 inet static" >> /etc/network/interfaces
63 echo "address" $ipnode >> /etc/network/interfaces
64 echo "netmask " $ipmask >> /etc/network/interfaces
65 echo "network" $ipnetwork >> /etc/network/interfaces
66 echo "gateway" $ipgateway >> /etc/network/interfaces
67 echo "broadcast" $ipbroadcast>> /etc/network/interfaces
68 echo "bridge_ports eth0" >> /etc/network/interfaces
69 echo "bridge_stp off" >> /etc/network/interfaces
70 echo "bridge_fd 0" >> /etc/network/interfaces
71
72 #Suppression des ligne de eth0
73 sed -i ’9d’ /etc/network/interfaces
74 sed -i ’9d’ /etc/network/interfaces
75
76 #Redemarage du network
77 /etc/init.d/networking stop
78 /etc/init.d/networking start
```

```
79
80 #supression des fichier temporaires et copie de common.sh
81 cd /root/
82 rm troll ipcluster ipgateway ipnetwork
83 rm /usr/share/ganeti/os/debootstrap/common.sh
84 mv common.sh /usr/share/ganeti/os/debootstrap/
85
86 #initialisation du cluster
87 gnt-cluster init --master-netdev xen-br0 --no-drbd-storage --nic-parameters link=eth0
       cluster1
88
89 #et verifier
90 gnt-node list
91
92 #reiseigner le inird pour les instances
93 gnt-cluster modify --hypervisor-parameter xen-pvm:initrd_path='/boot/initrd.img-2.6-xenU
       '
94 gnt-cluster modify --hypervisor-parameter xen-pvm:root_path='/dev/xvda1'
```

Listing B.5 – Installation et configuration du master node

```
1  #!/bin/bash
2
3  ##VERIFIER QUE postinstall.sh et ganeti.sh soient dans le meme dossier puis executer
       postinstall.sh (Sur le frontend)!
4  #Ajout des sources testing pour installer la derniere version de ganeti
5  echo "## wheezy security" >> /etc/apt/sources.list
6  echo "deb http://security.debian.org/ wheezy/updates main contrib non-free" >> /etc/apt/
       sources.list
7  echo "deb-src http://security.debian.org/ wheezy/updates main contrib non-free" >> /etc/
       apt/sources.list
8  echo " " >> /etc/apt/sources.list
9  echo "#wheezy" >> /etc/apt/sources.list
10 echo "deb http://ftp.fr.debian.org/debian/ wheezy main contrib non-free" >> /etc/apt/
       sources.list
11 echo "deb-src http://ftp.fr.debian.org/debian/ wheezy main contrib non-free" >> /etc/apt
       /sources.list
12
13 #Ajout du fichier de preference
14 echo "APT::Default-Release \"stable\";" > /etc/apt/apt.conf.d/80default-distrib
15
16 #Installation de ganeti
17 apt-get update && apt-get dist-upgrade -y --force-yes
18 apt-get -t testing install -y --force-yes ganeti2 ganeti-htools ganeti-instance-
       debootstrap
19
20 echo "Ajout du node dans /etc/hosts"
21 hostname=`cat /etc/hostname`
22
23 #recuperation des variables
24
25 #ip du node
26 ifconfig eth0 > troll
27 ipnode=`head -2 troll | tail -1 | cut -d':' -f2 | cut -d' ' -f1`
28 echo $ipnode " " $hostname >> /etc/hosts
29
30 #ip du broadcast
31 ipbroadcast=`head -2 troll | tail -1 | cut -d'B' -f2 | cut -d':' -f2 | cut -d' ' -f1`
32
```

```
#ip du masque de sous rseau
ipmask=‘head -2 troll | tail -1 | cut -d’M’ -f2 | cut -d’:’ -f2 | cut -d’ ’ -f1‘

#ip du reseau
ipnetwork=172.16.64.0

#ip de la passerelle
ipgateway=‘head -1 ipgateway‘
ipgateway=10.0.1.254

#ajout de cluster1 dans dans /etc/hosts
echo "ajout de cluster1 dans /etc/hosts"
ipcluster=‘cat ipcluster‘
echo $ipcluster "cluster1" >> /etc/hosts

#Dans /boot/ creer des liens symboliques :
cd /boot
cp vmlinuz-2.6.32-5-xen-amd64 vmlinuz-2.6-xenU
cp initrd.img-2.6.32-5-xen-amd64 initrd.img-2.6-xenU

#Pour le moment changera surement.
echo "creation du LVM"
umount /dev/sda5
pvcreate /dev/sda5
vgcreate xenvg /dev/sda5

#Creation du bridge xen-br0
echo " " >> /etc/network/interfaces
echo "auto xen-br0" >> /etc/network/interfaces
echo "iface xen-br0 inet static" >> /etc/network/interfaces
echo "address" $ipnode >> /etc/network/interfaces
echo "netmask " $ipmask >> /etc/network/interfaces
echo "network" $ipnetwork >> /etc/network/interfaces
echo "gateway" $ipgateway >> /etc/network/interfaces
echo "broadcast" $ipbroadcast>> /etc/network/interfaces
echo "bridge_ports eth0" >> /etc/network/interfaces
echo "bridge_stp off" >> /etc/network/interfaces
echo "bridge_fd 0" >> /etc/network/interfaces

#Suppression des ligne de eth0
sed -i ’9d’ /etc/network/interfaces
sed -i ’9d’ /etc/network/interfaces

#Redemarage du network
/etc/init.d/networking stop
/etc/init.d/networking start

#Supression des fichier temporaires et copie de common.sh
cd /root/
rm troll ipcluster ipgateway ipnetwork
rm /usr/share/ganeti/os/debootstrap/common.sh
mv common.sh /usr/share/ganeti/os/debootstrap/
```

Listing B.6 – Installation et configuration des noeuds esclaves

```
#!/bin/bash
ganeti_slaves="/root/ganeti_slaves"


```

```
while read node
do
    gnt-node add $node
done < $ganeti_slaves

echo "Combien d'instances par node ?? "
read nb
b=$nb
while read line
do
    for i in `seq $a $nb`;
    do
        echo "10.0.1.$i instance$i" >> /etc/hosts
        gnt-instance add -n $line -o debootstrap+default -t plain -s 2000 instance$i
    done
    nb=`echo $(($nb + $b))`
    a=`echo $(($a + $b))`
done < ganeti_slaves
```

Listing B.7 – Ajout des noeuds au cluster et création des instances

B.3 Scripts pour le déploiement de virt-manager

```
#!/bin/bash

# On  rcupre  la liste des noeuds  rservs  et on les stocke dans list_nodes
cat $OAR_FILE_NODES | sort -u > $HOME/list_nodes
list_nodes="$HOME/list_nodes"

# Rappel des machines  rservs
echo "----------------------------------"
echo "Liste des machines reservee:"
cat $list_nodes
echo "----------------------------------"
echo "Copie des fichiers vers toutes les machines."
for node in $(cat $list_nodes)
do
    # Copie des clefs ssh sur chaque noeud pour faciliter les communications
  scp $HOME/.ssh/id_rsa* root@$node:~/.ssh/
    # Copie du script d'ajout de 7 vm au noeud
  scp $HOME/vm.sh root@$node:~/
    # Copie du script d'installation de virt-manager
  scp $HOME/virt-install.sh root@$node:~/
    # Copie de la liste des noeuds sur chacun des noeuds
  scp $HOME/list_nodes root@$node:~/
    # Copie d'un fichier de configuration de xend
  scp $HOME/xend-config.sxp root@$node:~/
    # Copie du script ajoutant automatiquement les noeuds  virt-manager
  scp $HOME/ajout_noeuds.sh root@$node:~/
done
echo "----------------------------------"

# Mise   jour des paquets sur les noeuds
echo "Mise a jour des noyaux"
taktuk -l root -f $list_nodes broadcast exec [ apt-get update ]
taktuk -l root -f $list_nodes broadcast exec [ apt-get upgrade -y ]
taktuk -l root -f $list_nodes broadcast exec [ apt-get dist-upgrade -y ]

# Changement des mdp root
taktuk -l root -f $list_nodes broadcast exec [ 'echo -e "pttvirtu\npttvirtu" | passwd
    root' ]

# On s'assure que les droits sont bien positionns sur les diffrents scripts
taktuk -l root -f $list_nodes broadcast exec [ chmod 777 vm.sh ]
taktuk -l root -f $list_nodes broadcast exec [ chmod 777 virt-install.sh ]

echo "Lancement des scripts d'initialisation"
taktuk -l root -f $list_nodes broadcast exec [ bash vm.sh ]
taktuk -l root -f $list_nodes broadcast exec [ bash virt-install.sh ]

# On ajoute la clef aux clefs autorisees sur chaque noeud
taktuk -l root -f $list_nodes broadcast exec [ cat .ssh/id_rsa.pub >> .ssh/
    authorized_keys ]

# Une fois que taktuk s'est connect sur tous les noeuds, le fichier known_host contient
    les signatures de tous les noeuds.
# On le copie donc partout pour viter de valider lors de l'ajout d'une nouvelle
    connexion (ce qui provoquait un message d'erreur sous virt-manager)
for node in $(cat $list_nodes)
```

```
do
  scp $HOME/.ssh/known_hosts root@$node:~/.ssh/known_hosts
done
```

Listing B.8 – Configuration générale et copie des scripts pour l'installation de virt-manager

```
#!/bin/bash
##########################################
## Script d'installation de virt-manager ##
##########################################

# On s'assure qu'on est dans le home de root
cd /root

# Mise a jour du systme
#apt-get update
#apt-get upgrade -y

# Installation de libvirt pour communiquer avec le dmon xend
apt-get install libvirt-bin -y

# Installation de qemu-dm pour crer des machines virtuelles en mode full virtualis
apt-get install xen-qemu-dm-4.0 -y

# Correction d'un bug de qemu qui invalidait un lien symbolique
mkdir /usr/lib64/xen
mkdir /usr/lib64/xen/bin
cd /usr/lib64/xen/bin
ln -s /usr/lib/xen-4.0/bin/qemu-dm

# Activation des sockets Unix
# (ncessaire pour la communication entre libvirt et xend)
# echo "(xend-unix-server yes)" >> /etc/xen/xend-config.sxp

# Maintenant on dploie un fichier complet plutot que d'ajouter la ligne ci-dessus
cp $HOME/xend-config.sxp /etc/xen/xend-config.sxp

# Redmarrage de xend pour application des changements
/etc/init.d/xend restart

# Installation de virt-manager
apt-get install virt-manager -y

##############
# Optionnel #
##############
# Lancement du script pour le dploiement de 7 machines virtuelles
#./vm.sh
```

Listing B.9 – Installation et configuration de virt-manager

```
#!/bin/bash
############################################################
# Virt-manager se configure via le gestionnaire gconf.
# On a donc opt pour la  rcriture  complte  du fichier
# %gconf.xml
# Il est ncessaire d'attendre 1 min aprs la fermeture
# de virt-manager pour que le chemin de gconf soit cre
############################################################
```

```
9
10 # On  rcupre  la liste des noeuds  rservs  ( copi  par scp)
11 list_nodes="$HOME/list_nodes"
12 echo "Cration du fichier de config"
13
14 fichier=".gconf/apps/virt-manager/connections/%gconf.xml"
15
16 # Sauvegarde de l'ancien fichier de configuration
17 mv $fichier .gconf/apps/virt-manager/connections/%gconf.xml.BACKUP
18
19 #  Cration  du nouveau fichier
20 touch $fichier
21 echo ".....OK"
22 echo " Gnration  de l'entete"
23 echo '<?xml version="1.0"?>' >> $fichier
24 echo '<gconf>' >> $fichier
25 echo ' <entry name="autoconnect" mtime="1332080750" type="list" ltype="string">
26 ' >> $fichier
27 echo ' <li type="string">' >> $fichier
28 echo '<stringvalue>xen:///</stringvalue>' >> $fichier
29 echo '</li>' >> $fichier
30 echo '</entry>' >> $fichier
31 echo '<entry name="uris" mtime="1332080863" type="list" ltype="string">' >> $fichier
32 echo ".....OK"
33 echo "ajout des noeuds"
34
35 #Pour chaque noeud  rserv , on ajoute une entre dans le fichier
36 for node in $(cat $list_nodes)
37 do
38     echo '<li type="string">' >> $fichier
39     echo "<stringvalue>xen+ssh://root@$node/</stringvalue>" >> $fichier
40     echo '</li>' >> $fichier
41 done
42
43 echo ".....OK"
44 echo " Gnration  de la fin du fichier"
45 echo '<li type="string">' >> $fichier
46 echo ' <stringvalue>xen:///</stringvalue>' >> $fichier
47 echo '</li>' >> $fichier
48 echo '</entry>' >> $fichier
49 echo '</gconf>' >> $fichier
50 echo ".....OK"
51 echo "Vous pouvez relancer virt-manager"
```

Listing B.10 – Ajout des noeuds réservés à virt-manager