CLOUD COMPUTING

Yohan PARENT, Maxime LEMAUX
Cyprien FORTINA, Hyacinthe CARTIAUX

LP ASRALL
Année universitaire 2010-2011

Université Nancy 2
IUT Nancy-Charlemagne

Nous remercions Lucas Nussbaum, notre tuteur, et Sébastien Badia.

Yohan pour les cafés et sa connexion Internet et Hyacinthe pour les croissants.
Ainsi que Maxime et Cyprien pour leur bonne humeur...

Table des matières

1. Introduction

Le Cloud Computing (informatique dans le nuage) est une technique permettant de gérer des ressources (serveurs) et d'adapter très rapidement une infrastructure à des variations de charge de manière totalement transparente pour l'administrateur et les utilisateurs.

Dans le nuage, le client n'a pas connaissance de la complexité de la gestion matérielle derrière son environnement logiciel.

FIGURE 1.1.: Un nuage, source Wikimedia Commons.

Notre projet vise à déployer des solutions de Cloud Computing sur plusieurs sites de Grid 5000.

1.1. Le Cloud Computing

Le Cloud Computing est un concept :
- déporter sur une infrastructure distante les applications et les données ;
- abstraire la gestion de l'infrastructure et des ressources matérielles aux clients.

Il existe plusieurs types de cloud computing classés selon leur utilisation :
- **SaaS** : Software as a Service, le fournisseur maintient des applications que le client utilisé de manière totalement transparente. Par exemple, les applications Google (doc, reader, gmail, etc).
- **PaaS** : Platform as a Service, le client maintient uniquement ses applications alors que le fournisseur maintient les serveurs et l'infrastructure logicielle (bases de données, sécurité, stockage). Par exemple : Microsoft Azure.
- **IaaS** : Infrastructure as a Service. On met à disposition des ressources composées par une infrastructure virtualisée, dont la plus grande partie est localisée à distance dans des Datacenters.

Notre projet tuteuré porte sur cette dernière forme de cloud : l'**IaaS**.

Un cloud (ou nuage) désigne un ensemble de serveurs de virtualisation interconnectés.

L'apport des solutions de cloud provient de l'« élasticité », il s'agit de simplifier la gestion d'un parc de serveurs de virtualisation offrant des ressources matérielles, et de permettre d'installer, de configurer et de migrer les machines virtuelles à la volée en fonction des besoins.

Nous allons comparer plusieurs solutions de cloud sous GNU/Linux :
- **Eucalyptus**
- **Open Nebula**
- **Open Stack**

Ces solutions n'ont d'intérêt que pour une infrastructure de taille importante (plusieurs dizaines de serveurs). Nous avons réalisé nos expérimentations sur Grid 5000.

1.2. Grid 5000

Pour réaliser notre projet tuteuré, nous utilisons la plate-forme Grid 5000 [1].
Grid 5000 est une infrastructure distribuée sur 9 sites en France, pour la recherche dans les systèmes parallélisés à grande échelle et les systèmes distribués.

Grid 5000 vise a réunir 5000 processeurs sur 9 sites en France. Cette plate-forme est utilisée dans le cadre de projets expérimentaux, dans le domaine de la recherche dans les systèmes informatiques distribués et à grande échelle.

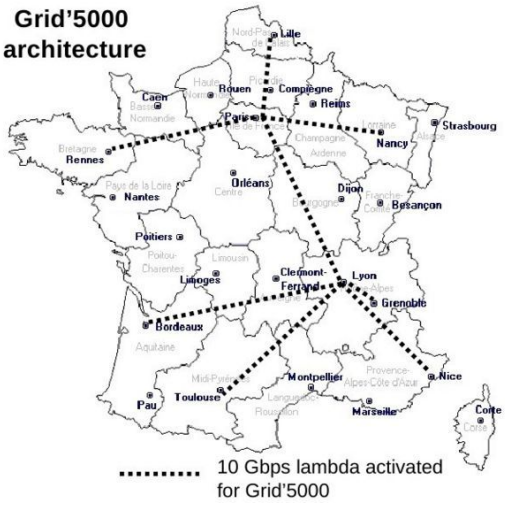

FIGURE 1.2.: Architecture de Grid 5000, source grid5000.fr

1.2.1. 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 nœuds (nodes).

1. Site de grid5000, présentation, documentation, tutoriels, etc

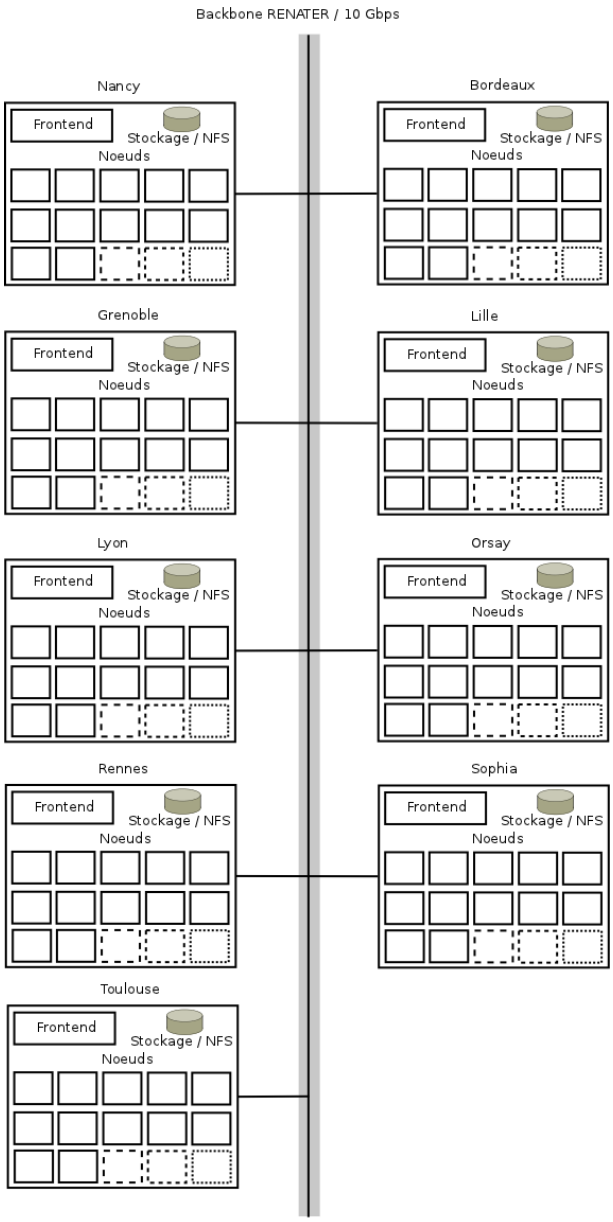

FIGURE 1.3.: Vue très simplifiée de Grid 5000

L'utilisateur de Grid 5000 accède à chaque site par son frontend en utilisant SSH.
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 effectuant des réservations.

Nous avons pu visiter la salle serveurs du site de Nancy située au Loria. Notre tuteur, M. Lucas Nussbaum et M. Sébatien Badia, nous ont fait 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.

2. NFS : système de fichiers en réseau permettant de partager des données principalement entre systèmes UNIX.

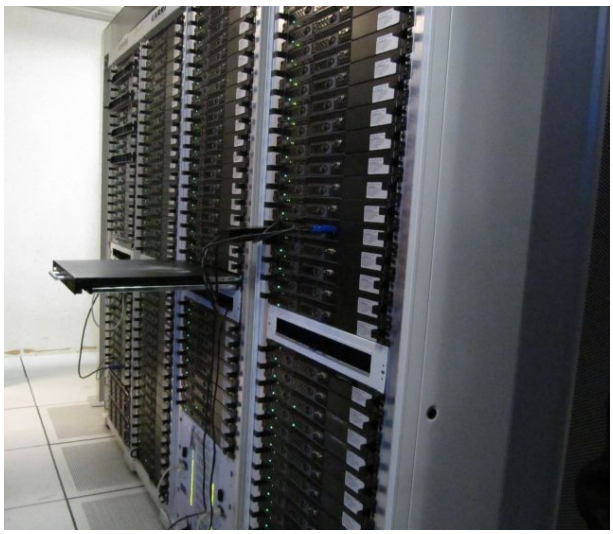

FIGURE 1.4.: Armoires de serveurs de Nancy

FIGURE 1.5.: Quelques serveurs du site de Nancy

1.2.2. 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*).

Dans le cadre de notre projet, nous effectuons uniquement des réservations de type *deploy*. La commande oarsub nous permet de réserver des nœuds sur un site (en créant un job). Voici un exemple d'utilisation d'oarsub, pour réserver 10 nœuds pendant 3 heures en déploiement.

```
$ oarsub -I -t deploy -l nodes=5,walltime=3
```

Après réservation, oarsub ouvre un shell dans lequel des variables d'environnements sont définies comme $OAR_FILE_NODE, qui est le nom d'un fichier avec la liste des nœuds réservés, ou $OAR_JOB_ID.

```
$ cat $OAR_FILE_NODES | sort -u
paramount-24.rennes.grid5000.fr
paramount-8.rennes.grid5000.fr
paramount-9.rennes.grid5000.fr
parapluie-19.rennes.grid5000.fr
parapluie-26.rennes.grid5000.fr
```

```
$ echo $OAR_JOB_ID
380540
```

Pour supprimer le job et libérer les nœuds, on utilise la commande oardel.

```
$ oardel 380540
```

Kadeploy permet de déployer des environnements personnalisés sur les noeuds réservés à l'aide d'une commande simple. Lorsque la réservation est terminée, le noeud est automatiquement restauré avec un environnement Debian.

Pour déployer un environnement sur tous les noeuds réservés, il faut utiliser la commande kadeploy3

```
$ kadeploy3 -e lenny-x64-base -f $OAR_FILE_NODES
```

1.2.3. 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)

TABLE 1.1.: Réseau de production de chaque site

Site	Réseau de production
Bordeaux	172.16.0.0/20
Grenoble	172.16.16.0/20
Lille	172.16.32.0/20
Lyon	172.16.48.0/20
Nancy	172.16.64.0/20
Orsay	172.16.80.0/20
Rennes	172.16.96.0/20
Toulouse	172.16.112.0/20
Sophia	172.16.128.0/20
DAS-3	172.16.144.0/20
Reims	172.16.160.0/20
Luxembourg	172.16.176.0/20
Porto Alegre	172.16.192.0/20
Orléans	172.16.208.0/20

Selon le site, le serveur dhcp accessible par l'interface réseau de production attribuera une adresse comprise dans les réseaux ci-dessus.

TABLE 1.2.: Réseau pour les machines virtuelles sur chaque site

Site	Réseau	Passerelle	DHCP
Bordeaux	10.128.0.0/14	10.131.255.254	10.131.255.253
VPN nets	10.132.0.0/14	n/a	n/a
Lille	10.136.0.0/14	10.139.255.254	10.139.255.253
Lyon	10.140.0.0/14	10.143.255.254	10.143.255.253
Nancy	10.144.0.0/14	10.147.255.254	10.147.255.253
Orsay I	10.148.0.0/14	10.151.255.254	10.151.255.253
Orsay II	10.152.0.0/14	10.155.255.254	10.155.255.253
Rennes	10.156.0.0/14	10.159.255.254	10.159.255.253
Toulouse	10.160.0.0/14	10.163.255.254	10.163.255.253
Sophia	10.164.0.0/14	10.167.255.254	10.167.255.253
DAS-3	10.168.0.0/14	10.171.255.254	10.171.255.253
Luxembourg	10.172.0.0/14	10.175.255.254	10.175.255.253
Porto Alegre	10.176.0.0/14	10.179.255.254	10.179.255.253
Grenoble	10.180.0.0/14	10.183.255.254	10.183.255.253

Chaque site dispose donc de classes d'adresses IP réservées aux machines virtuelles, celles ci sont routables sur le réseau de grid 5000.
De plus, chaque site dispose d'une passerelle avec le nom d'hôte virtual-gw et d'un serveur dhcp.

3. grid5000.fr : Network
4. grid5000.fr : Network interlink

Pour vérifier que les IP de machines virtuelles sont bien routées, il est possible de tester uniquement les passerelles. Le script suivant effectue ce test pour tous les sites.

```
#!/bin/bash

for i in `cat listes_des_sites` ; do
  ping -c 1 virtual-gw.$i > /dev/null
  if [ $? == 0 ] ; then
    echo $i >> sites-virtual-gw
    echo $i : OK
  else
    echo $i : KO
  fi
done
```

Pour utiliser le serveur DHCP[5] et obtenir automatiquement une IP pour une machine virtuelle, il faut que l'adresse mac de l'interface réseau soit préfixée par *00 :16 :3e*.

A l'avenir, grid5000.fr : KaVLAN, outil de gestion de VLAN sur Grid 5000 sera disponible sur tous les sites de Grid 5000 pour permettre de réserver des IP et d'utiliser simplement des VLAN. Cette solution sera alors à privilégier pour simplifier la configuration réseau et le déploiement d'un cloud.

1.3. Objectifs et répartition du travail

Les objectifs à l'issue du projet sont de produire :
- des images d'environnements, des scripts et documentations pour déployer rapidement des solutions de cloud sur Grid 5000 ;
- des expériences à grande échelle
- des retours d'expériences

Afin de travailler plus efficacement, nous avons mis en place de nouveaux outils en combinant les outils Google Code et Google Groups :
- un dépôt SVN ;
- un wiki ;
- une mailing list ;
- un répertoire partagé via Dropbox.

Nous avons également appris LaTeXpour rédiger notre rapport.

FIGURE 1.6.: Notre projet sur Google Code

5. grid5000.fr : Virtual Network Configuration (DHCP)

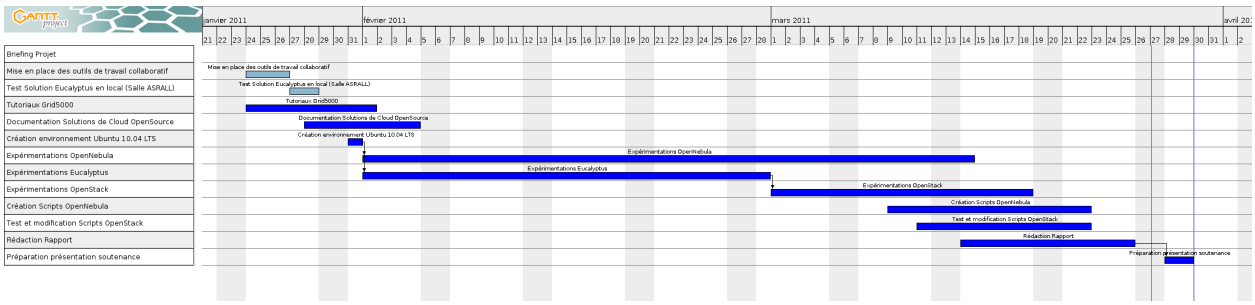

FIGURE 1.7.: Diagramme de Gantt de l'organisation de notre projet

Lors de la première semaine, nous avons mis en place nos outils de travail collaboratif, et commencé à découvrir l'environnement offert par Grid 5000 en suivant des tutoriels. Nous avons également configuré nos machines (SSH) pour être plus efficace. Pour cette tâche, nous avons écrit quelques scripts, un premier pour copier nos clés ssh d'un site vers tous les autres, et un deuxième pour synchroniser le répertoire personne (/home/user/) vers d'autres sites. Ces scripts sont en annexe.

Arrivé à la deuxième semaine, nous nous sommes partagés le travail équitablement : Cyprien et Yohan se sont occupés de Eucalyptus et OpenStack, Maxime et Hyacinthe se sont concentrés sur OpenNebula. Les premiers tests de Eucalyptus ont été effectués sur des machines en salle 501, et un environnement minimal Ubuntu Lucid a été créé sur le site de Nancy de Grid 5000. Cet environnement a servi de base pour le déploiement de Eucalyptus et OpenNebula.

Pendant la 3ème semaine, nous avons créé une image disque contenant une installation de Debian Squeeze à l'aide de qemu, ainsi que les environnements pour déployer Eucalyptus et OpenNebula sur Grid 5000. Nous déployons pour la première fois des machines virtuelles ttylinux sur un cloud OpenNebula, mais nous rencontrons des problèmes de configuration réseau. De plus, les environnements ne sont pas encore totalement fonctionnels.

Dans la 4ème semaine, Yohan et Cyprien ont recréé une image de Ubuntu Lucid Enterprise Cloud depuis un accès physique aux machines du site de Nancy en espérant résoudre leurs problèmes. Du coté de OpenNebula, le travail se concentre sur la configuration réseau (bridge et DHCP pour les machines virtuelles) et l'amélioration des environnements (problème d'espace disque disponible dans la partition racine des nœuds pour les images de machines virtuelles).

La 5ème semaine est celle des vacances, Yohan et Cyprien arrêtent de travailler sur Eucalyptus et se lancent sur OpenStack. Nous commençons le rapport dans ses grands axes pour se faire une idée du contenu de ce dernier.

Pendant la 6ème semaine, Yohan et Cyprien travaillent toujours à corriger les problèmes de configuration réseau pour les machines virtuelles sous OpenStack. L'autre groupe finalise les environnements pour OpenNebula, et commence à écrire des scripts pour automatiser le déploiement d'un cloud sur plusieurs sites de Grid 5000 et la configuration réseau des machines virtuelles.

Lors de la 7ème semaine, les problèmes concernant OpenStack ne sont toujours pas réglés (apparemment liés à la configuration réseau) et nous finalisons notre travail sur OpenNebula. Les principaux problèmes viennent de notre manque de prévision par rapport à l'hétérogénéité de Grid 5000. En effet, OpenNebula se base par défaut sur la technologie KVM qui nécessite un support matériel de la virtualisation, or le support n'est pas activé sur tous les clusters.

La 8ème semaine a été essentiellement dédiée à l'écriture du rapport et à la réalisation de la présentation.

2. Eucalyptus

2.1. Introduction

Eucalyptus est un ensemble d'outil disponible sous Licence BSD[1] qui permet de mettre en place une infrastructure de type cloud computing sur une grappe de serveurs. Cet outil est donc open source et a été développé par une équipe de chercheurs de l'université de Californie à partir de 2007. Son nom fait référence à l'acronyme « Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems » qui peut se traduire en « Utilitaire d'Architecture informatique élastique pour relier vos programmes à des systèmes fonctionnels ».

Depuis 2009, il est intégré dans la distribution GNU/Linux Ubuntu 9.04 Jaunty Jackalope. Eucalyptus est devenu très populaire et est considéré comme l'une des principales solutions open-source pour la mise en place de systèmes de type *cloud*. Cependant, il est un peu délaissé actuellement au profit de la solution OpenStack.

Il existe également un packaging appelé Ubuntu Enterprise Cloud, UEC en abrégé, réalisé par Canonical et fourni avec Ubuntu Server Edition. UEC comprend Eucalyptus avec un certain nombre d'autres logiciels open source pour faciliter l'installation et la configuration de ce genre d'architecture.

2.1.1. Composants d'Eucalyptus

Une configuration de cloud fondée sur Eucalyptus se compose de cinq types de composants principaux.

Contrôleur de nœud (NC) Le rôle du nœud est d'héberger KVM[2], il sert ainsi d'hyperviseur pour les machines virtuelles qui sont déployées. Les machines virtuelles fonctionnant sur l'hyperviseur sont appelées des «instances». Eucalyptus permet aussi d'utiliser d'autres types d'hyperviseurs comme XEN[3], mais Canonical a fait le choix de privilégier KVM.

Le contrôleur de nœud fonctionne sur chaque nœud et est chargé de vérifier le cycle de vie des instances en cours d'exécution sur le nœud. Il interagit avec le système d'exploitation et l'hyperviseur en cours d'exécution sur le nœud d'un côté et le contrôleur de cluster (CC) de l'autre côté. Le contrôleur interroge le système d'exploitation s'exécutant sur le nœud de afin de découvrir les ressources physiques du nœud (le nombre de cœurs, la taille de la mémoire, l'espace disque disponible et aussi de s'informer sur l'état des instances VM en cours d'exécution sur le nœud et propage ces données au contrôleur de cluster.)

1. La licence BSD est une licence libre utilisée pour la distribution de logiciels.`http://www.freebsd.org/copyright/license.html/`
2. KVM (Kernel-based Virtual Machine) est une solution de virtualisation libre pour Linux. Elle fonctionne sur les architectures x86 disposant des technologies Intel VT ou AMD SVM (AMD-V).
3. Xen est un logiciel libre de virtualisation, plus précisément un hyperviseur de machine virtuelle. Son développement a débuté sous la forme d'un projet de recherche de l'université de Cambridge au Royaume-Uni. La société XenSource a par la suite été créée et en a poursuivi le développement. Xen permet de faire fonctionner plusieurs systèmes d'exploitation virtuels (invités) sur une seule machine hôte.

Contrôleur de cluster (CC) Ce contrôleur sert à déployer et gérer les différents contrôleurs de nœuds. Il sert également à gérer la mise en place du réseau entre les instances des différents nœuds. C'est lui qui communique l'ensemble des informations au contrôleur du cloud.

Il a 4 fonctions principales :

- Il reçoit les requêtes de déploiement des instances du contrôleur de cloud .
- Il décide sur quel contrôleur de nœud les instances seront déployées.
- Il contrôle le réseau virtuel entre les instances.
- Il collecte les informations des contrôleurs de nœuds enregistrés et les rapporte au contrôleur de cluster.

Contrôleur de stockage Walrus (WS3) Il est appelé : Walrus Storage Controller (WS3). WS3 fournit un service de stockage persistant simple à l'aide des API REST [4] et SOAP [5] et compatible avec les API S3. [6]

Il assure 3 fonctions principales :

- Le stockage des images de machines virtuelles.
- Le stockage des images prises en fonctionnement à un instant précis.
- Stocker des fichiers et les service en utilisant l'API S3.

En fait, WS3 peut être considéré comme un simple système de stockage de fichiers.

Contrôleur de stockage (SC) Ce contrôleur fournit un service de stockage persistant pour les instances. C'est similaire au service «Elastic Block Storage (EBS) [7]» d'Amazon.

Il a donc 3 fonctions essentielles :

- La création de dispositifs EBS persistants.
- Fournir le système de stockage de blocs aux instances.
- Autoriser la création d'images des volumes pour permettre leurs sauvegardes.

Contrôleur du cloud (CLC) C'est un programme Java qui sert de Frontend à l'infrastructure. Il offre, d'un côté une interface de gestion et de contrôle conforme aux services ec2/s3, une véritable boîte à outils complète, et de l'autre côté, permet d'interagir avec les autres composants de l'infrastructure. Il est également possible d'avoir une interface web qui permet aux utilisateurs de gérer certains aspects de l'infrastructure.

Il a 3 rôles principaux :

- Surveiller la disponibilité des ressources sur les différentes composantes de l'infrastructure du cloud.
- L'arbitrage des ressources - C'est à dire décider quel cluster sera utilisé pour la virtualisation des instances.
- Monitorer les instances en cours d'exécution.

En résumé, le CLC a une connaissance approfondie de la disponibilité et l'utilisation des ressources dans le nuage et permet de surveiller son état.

L'exemple d'une architecture avec l'ensemble des composants d'Eucalyptus :

4. REST (Representational State Transfer) est une manière de construire une application pour les systèmes distribués comme le World Wide Web. Le terme a été inventé par Roy Fielding en 2000. REST n'est pas un protocole ou un format, c'est un style d'architecture, c'est le style architectural original du Web.

5. SOAP (ancien acronyme de Simple Object Access Protocol) est un protocole de RPC orienté objet bâti sur XML. Il permet la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur. Le transfert se fait le plus souvent à l'aide du protocole HTTP, mais peut également se faire par un autre protocole, comme SMTP.

6. Simple Storage Service, un service de stockage via service web d'Amazon. Amazon S3 est un service de stockage pour Internet. Il est pensé pour simplifier l'accès à l'informatique à l'échelle du web, pour les développeurs. Amazon S3 offre une simple interface de services web à utiliser pour stocker et récupérer n'importe quelle quantité de données, à tout moment, de n'importe où sur le web. Il permet aux développeurs d'accéder à la même infrastructure hautement évolutive, fiable, sûre, rapide, et peu coûteuse qu'Amazon utilise pour faire fonctionner son propre réseau mondial de sites. Le service vise à maximiser les avantages d'échelle et les transmettre aux développeurs.

7. Le stockage extensible (Amazon EBS) fournit des volumes de stockage en mode bloc à utiliser avec les instances AmazonEC2. Les volumes EBS Amazon sont des stockages hors instances qui persistent indépendamment de la vie d'une instance. Amazon Elastic Block Store fournit des volumes de stockage hautement disponible et fiable qui peuvent être annexés à une instance Amazon EC2 en cours d'exécution et exposés comme un périphérique au sein de l'instance. EBS Amazon est spécialement adapté aux applications qui nécessitent une base de données, un système de fichier ou un accès à un stockage brut en mode bloc.

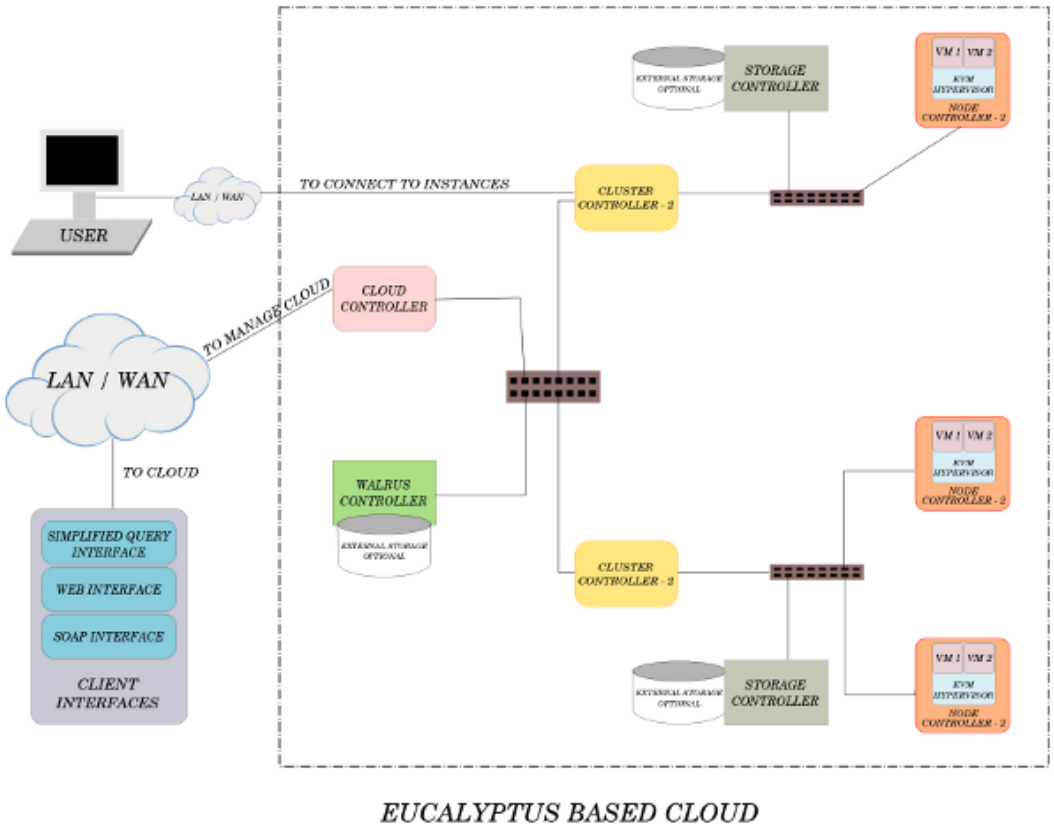

FIGURE 2.1.: Composants Eucalyptus, Eucalyptus Beginner's Guide - UEC Edition.

2.1.2. Un fonctionnement modulaire

Ce qui fait la force d'une architecture fondée sur Eucalyptus est sa grande modularité.

En effet, les composants Eucalyptus ont des interfaces bien définies (via WSDL [8], car se sont des services web) et peuvent donc être facilement échangés pour des composants personnalisés.

Eucalyptus est flexible et peut être installé sur une configuration très minime. Pourtant, il peut être installé sur des milliers de cœurs et des téraoctets de stockage. Et il peut le faire comme une superposition à une infrastructure existante.

Le fonctionnement en sous-ensemble qui forme chacun un cluster distinct permet de faciliter son déploiement sur une infrastructure multi-sites comme c'est le cas pour le Grid5000. L'ensemble étant ensuite géré au niveau du contrôleur du cloud, le Walrus assurant la communication avec les contrôleurs de stockages.
Ce fonctionnement est parfaitement illustré sur le schéma ci-dessous.

8. Le WSDL décrit une Interface publique d'accès à un Service Web, notamment dans le cadre d'architectures de type SOA (Service Oriented Architecture). C'est une description fondée sur le XML qui indique « comment communiquer pour utiliser le service »

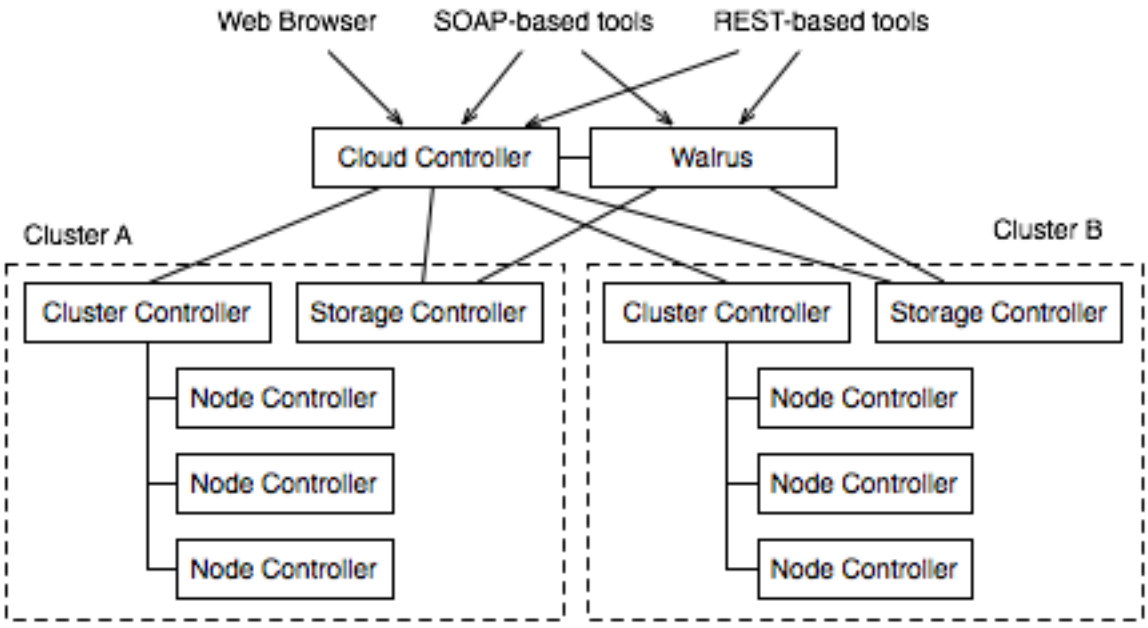

FIGURE 2.2.: Architecture Eucalyptus, Documentation officiel Eucalyptus.

Sur le schéma, on constate que le contrôleur de cloud (CLC) et le «Walrus» sont des composants de haut niveau, avec un de chaque dans une installation de cloud computing. Ensuite les autres composants sont réparties sur les différents clusters que l'on veut utiliser.

Il existe 3 façons principales de déployer Eucalyptus :

– Installation à partir des sources.
– Installation à partir des paquets.
– Installation à l'aide de la distribution Ubuntu Enterprise Cloud.

Nous avons testé ces 3 types d'installation et nous allons vous les détailler dans la partie suivante.

2.1.3. Interface Web

Un des gros point fort d'Eucalyptus, c'est qu'il inclut une interface web pour la gestion des utilisateurs, du cloud et des machines virtuelles. Cette interface est très simple d'utilisation et intuitive et nous avons pu la tester en salle ASRALL. Sur Grid5000 nous n'avons pas pu l'utiliser car elle est hébergée sur le contrôleur de cloud, que nous n'arrivons pas à contacter comme précisé plus loin dans la partie "Problème rencontrés", mais le service web est pourtant fonctionnel car nous avons installé le navigateur Lynx pour nous connecter à l'interface web et cela fonctionne.

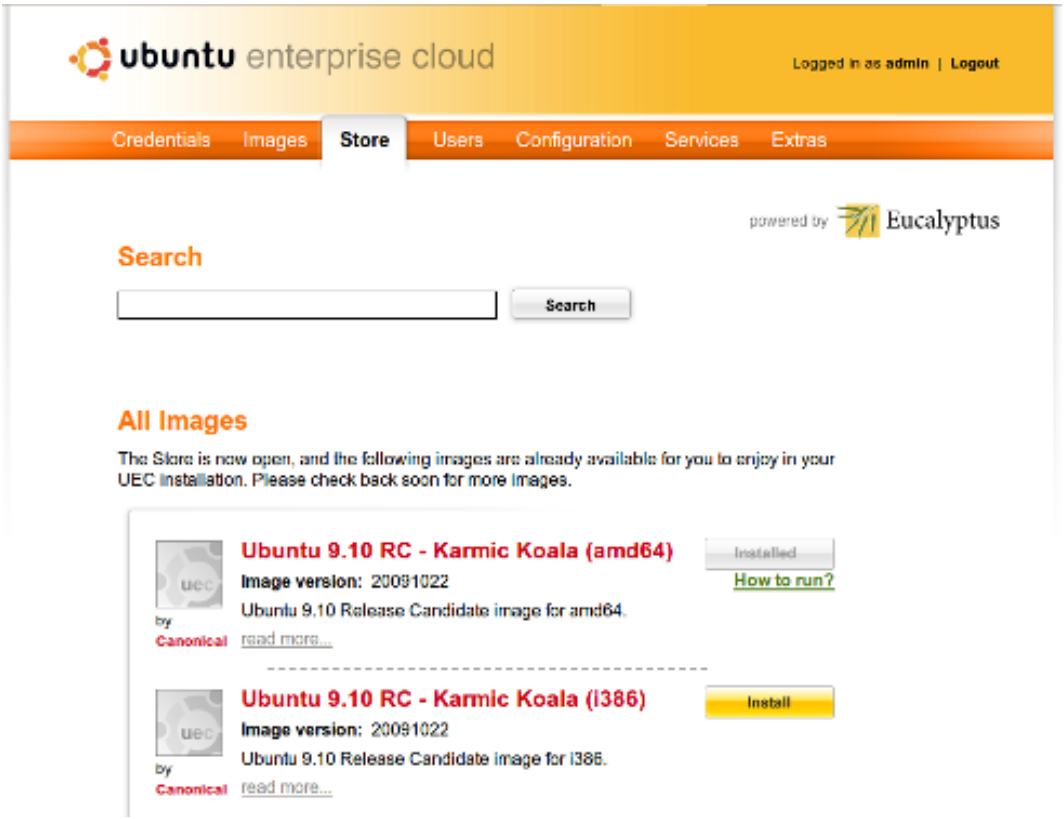

FIGURE 2.3.: Source :UEC PackageInstall`https://help.ubuntu.com/community/UEC/PackageInstall`

2.2. Installation

2.2.1. La configuration matérielle recommandée

L'installation des composants suivants :
- le contrôleur de cloud (clc)
- le contrôleur de cluster (cc)
- le walrus
- le contrôleur de stockage (sc)

demande au minimum la configuration matérielle suivante :
- CPU 1GHz
- RAM 2GB
- Disque Dur 5400rpm IDE
- Espace Disque 40GB
- Réseau 100Mbps

Les nœuds demandent quant à eux, la configuration minimum suivante :
- CPU VT extensions
- RAM 1GB
- Disque Dur 5400rpm IDE
- Espace Disque 40GB
- Réseau 100Mbps

Le point clé est surtout le support de la virtualisation puisque ce sont eux qui accueilleront les instances des machines virtuelles.

2.2.2. Installation fondée sur Ubuntu Enterprise Cloud

La façon la plus simple d'installer Eucalyptus est d'utiliser la distribution Ubuntu Server qui intègre de base une série d'outils dont Eucalyptus et qui porte le nom : «Ubuntu Enterprise Cloud (UEC)».
La société de Mark Shuttleworth, Canonical LTD, a développé une série de services à destination des professionnels afin que ces derniers développent leur solution d'informatique distribuée. Baptisé Ubuntu Enterprise Cloud (UEC), cette solution se base sur la technologie open source d'Eucalyptus qui permet aux entreprises de déployer leur propre solution de cloud computing tout en communicant avec l'interface de programmation

d'Amazon Elastic Compute Cloud (EC2).
En intégrant UEC directement au sein d'Ubuntu Server Edition, Canonical devient le premier éditeur à proposer une solution de Linux professionnelle qui s'ouvre sur la programmation sur nuages de serveur.
UEC se complète d'outils permettant de contrôler, de déployer et de gérer ces serveurs, l'objectif principal de Canonical est de faciliter au maximum le déploiement des solutions de cloud computing.
Nous avons en effet constaté la facilité d'installation d'Eucalyptus grâce à cette distribution. La première étape consiste simplement à télécharger l'image ISO [9] de Ubuntu Server sur le site officiel Ubuntu.
Ensuite, il suffit de se laisser guider par l'interface d'installation.
L'installation est illustrée par les captures d'écran suivantes, elle se résume en 5 étapes :

- Sélection "Installation Ubuntu Enterprise Cloud".
- Choix de l'adresse IP du contrôleur de cloud.
- Choix des différents composants à installer sur la même machine. (Pour le test, nous n'avons utilisé que 2 machines).
- Choix du nom du cluster pour le contrôleur de cluster.
- Choix d'une plage d'adresses IP pour le réseau des instances.

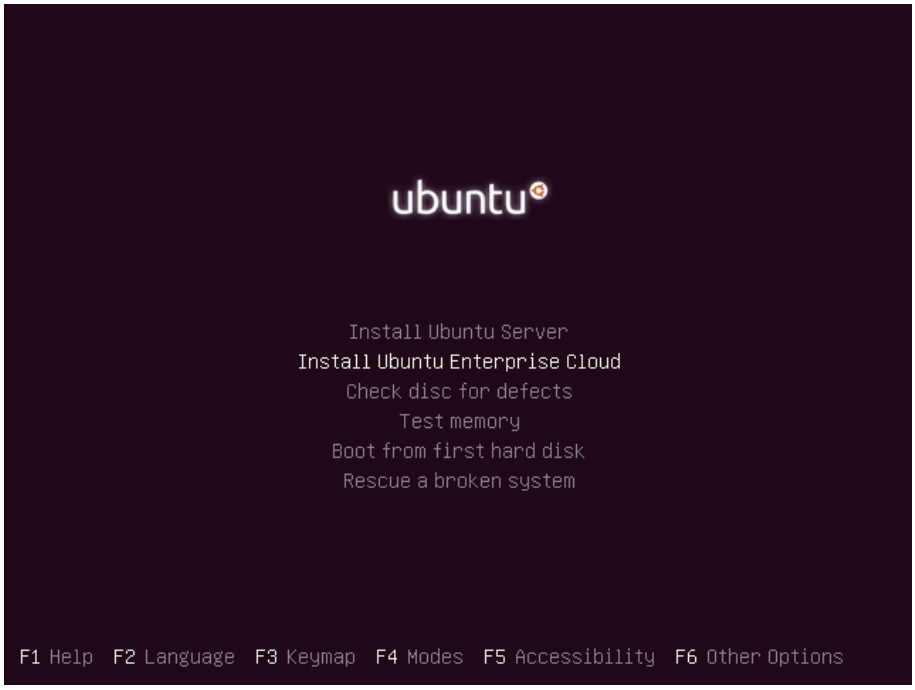

FIGURE 2.4.: Installation UEC - Écran d'accueil, Documentation officiel Eucalyptus.

9. Une image ISO est une image d'un disque (CD, DVD ou disque dur) sous forme de fichier, créer avec un logiciel de gravure.

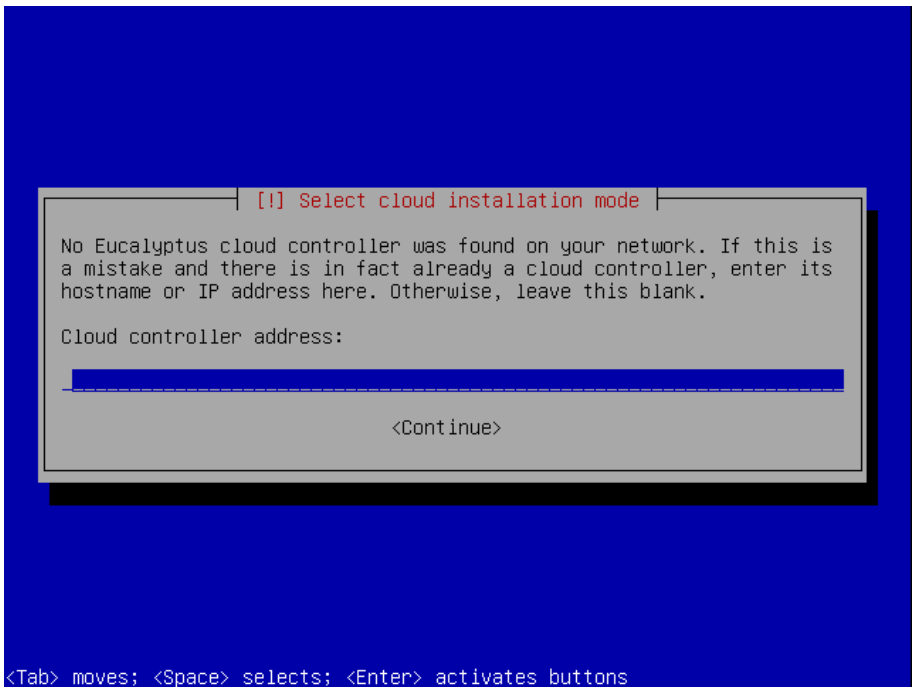

FIGURE 2.5.: Installation UEC - Étape 1, Documentation officiel Eucalyptus.

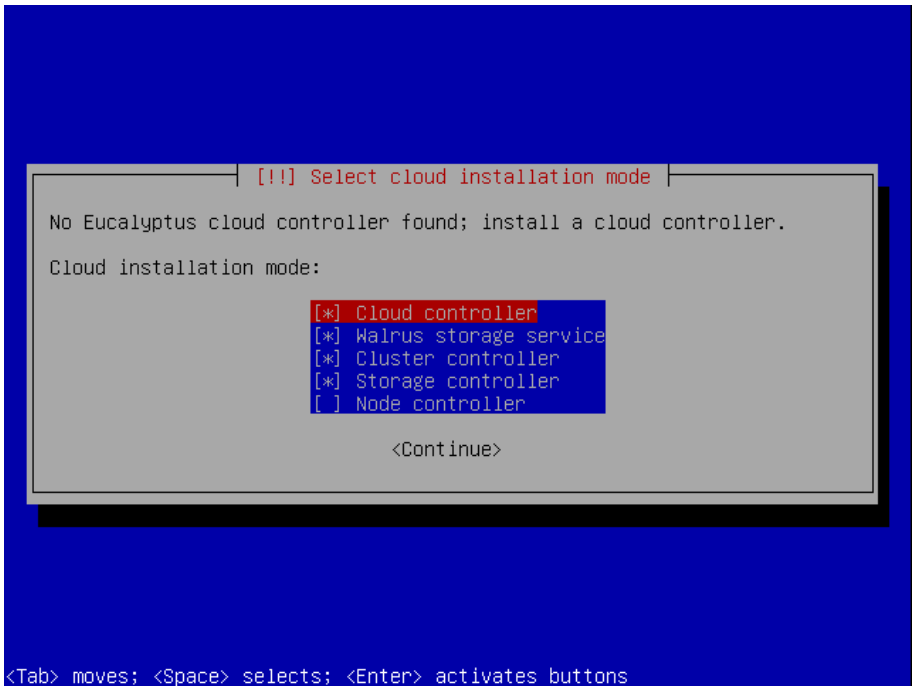

FIGURE 2.6.: Installation UEC - Étape 2, Documentation officiel Eucalyptus.

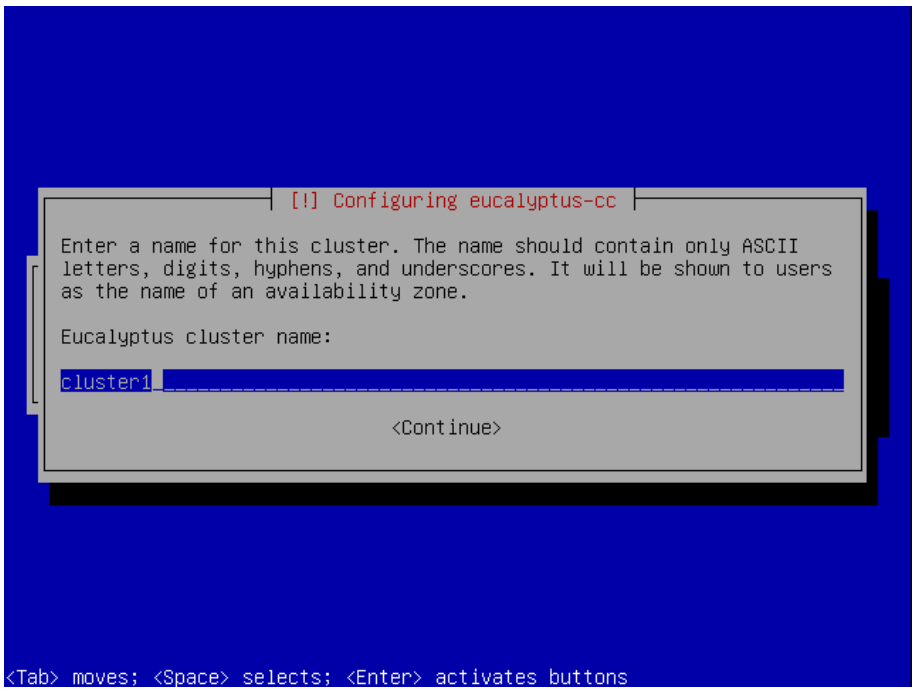

FIGURE 2.7.: Installation UEC - Étape 3, Documentation officiel Eucalyptus.

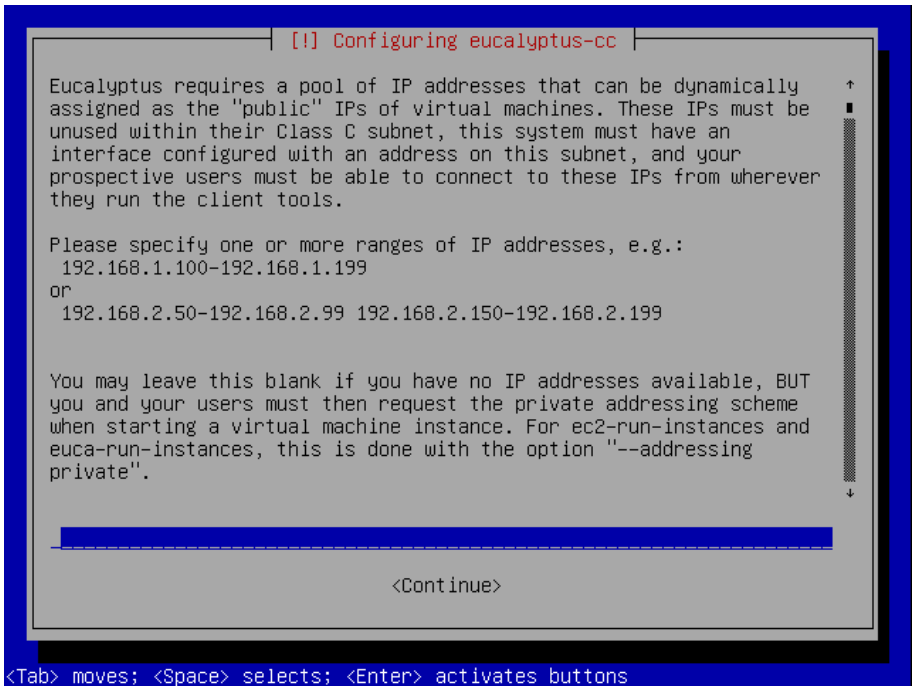

FIGURE 2.8.: Installation UEC - Étape 4, Documentation officiel Eucalyptus.

Après avoir installé le contrôleur de cloud, il suffit de refaire la même installation d'Ubuntu server sur chaque contrôleur de nœud. A l'exception, qu'il suffira à l'étape 2, de choisir "Node Controller".

L'installation a été testée sur deux machines en salle ASRALL et fonctionnait parfaitement, ce qui montre qu'Eucalyptus est utilisable tant que nous ne sommes pas sur une architecture particulière et que nous suivons à la lettre l'installation classique. Les choses se sont nettement compliquées quand nous sommes passés à la phase d'expérimentation sur Grid 5000. Nous ne détaillerons pas ici la configuration puisque ce sera la même méthode que sur Grid 5000, sur laquelle nous allons revenir plus en détail afin de préciser les difficultés rencontrées.

2.2.3. Installation à partir des paquets

Cette installation consiste à partir de l'environnement Ubuntu Server 10.04 LTS Ludic Lynx d'installer les différents paquets qui permettent l'installation et l'utilisation d'Eucalyptus. Nous avons également testé cette installation en local de la même manière que précédemment mais nous ne détaillerons que l'installation sur Grid 5000, ce qui était l'objectif principal du projet.

Nous commençons par déployer l'environnement Ubuntu Server. Puis, nous mettons à jour le système pour partir de la version la plus récente sur l'ensemble des machines que nous voulons utiliser pour constituer notre cloud.

La première chose à penser est que si le noyau est mis à jour et que nous voulons utiliser la dernière version, il faut redémarrer l'environnement, le soucis, c'est que le fichier de description de ce dernier contient en dur le nom du noyau à charger, pour remédier à cela, nous avons utilisé un lien symbolique, ainsi nous pouvions facilement utiliser et donc booter sur le noyau mis à jour.

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

Sur le Front end Installation des paquets nécessaires eucalyptus-cc et eucalyptus-cloud :

```
$ sudo apt-get install eucalyptus-cloud eucalyptus-cc eucalyptus-walrus eucalyptus-sc
```

Au cours de cette étape, nous répondons à quelques questions lors de l'installation des différents paquets, comme le nom du cluster et la plage d'adresse Ip du réseau.

Sur les nœuds Le paquet eucalyptus-nc est nécessaire.

```
$ sudo apt-get install eucalyptus-nc
```

C'est la seule étape d'installation pour les nœuds. Il convient alors d'effectuer la configuration du bridge à l'aide du script suivant :

```
1  interface=eth0
2  bridge=br0
3  sudo sed -i "s/^iface $interface inet \(.*\)$/iface $interface inet manual\n\nauto br0\niface
       $bridge inet \1/" /etc/network/interfaces
4  sudo tee -a /etc/network/interfaces <<EOF
5          bridge_ports $interface
6          bridge_fd 9
7          bridge_hello 2
8          bridge_maxage 12
9          bridge_stp off
10 EOF
11 sudo /etc/init.d/networking restart
```

Enfin, par défaut, un utilisateur «eucalyptus» est créé sans mot de passe, il faut donc lui changer celui-ci de la manière suivante :

```
$ sudo passwd eucalyptus
```

Cela va permettre de s'y connecter du Front end pour y installer les clés SSH.

```
$ sudo -u eucalyptus ssh-copy-id -i ~eucalyptus/.ssh/id_rsa.pub eucalyptus@<IP_OF_NODE>
```

Le mot de passe de l'utilisateur «eucalyptus» n'étant plus nécessaire, nous pouvons le supprimer pour des questions de sécurité.

```
$ sudo passwd -d eucalyptus
```

La configuration des services L'utilisateur «eucalyptus» du contrôleur de cloud doit avoir accès en SSH au Walrus, au contrôleur de stockage et au contrôleur de cluster. Il faut donc envoyer la clé SSH publique sur les différents contrôleurs de la manière suivante.
Dans un premier temps, créer un mot de passe temporaire pour l'utilisateur «eucalyptus» sur la cible :

```
$ sudo passwd eucalyptus
```

Ensuite, publier la clé SSH publique du contrôleur de cloud :

```
$ sudo -u eucalyptus ssh-copy-id -i /var/lib/eucalyptus/.ssh/id_rsa.pub eucalyptus@<IP_OF_NODE
    >
```

Enfin, supprimer le mot de passe pour des questions de sécurité :

```
$ sudo passwd -d eucalyptus
```

Il est donc désormais possible de se connecter sur l'ensemble des contrôleurs en SSH. Nous avons testé que cela fonctionnait bien.

Il s'agit ensuite de vérifier que la configuration des différents services est correcte en éditant les fichiers de configuration suivants : */etc/eucalyptus/eucalyptus-cc.conf* et */etc/eucalyptus/eucalyptus-ipaddr.conf* Le principal ici est de vérifier les noms des contrôleurs et leurs adresses IP.

La publication des services Cette étape consiste simplement à démarrer l'ensemble des services du cloud.

- Walrus :

```
$ sudo start eucalyptus-walrus-publication
```

- Contrôleur de Cluster :

```
$ sudo start eucalyptus-cc-publication
```

- Contrôleur de stockage :

```
$ sudo start eucalyptus-sc-publication
```

- Contrôleur de nœud :

```
$ sudo start eucalyptus-nc-publication
```

Enfin, sur le contrôleur de cluster et le contrôleur de cloud, on lance l'écoute pour découvrir les nœuds :

```
$ sudo start uec-component-listener
```

Puis, on vérifie que l'ensemble fonctionne, en observant le fichier de log, si la découverte se fait correctement, nous obtenons le résultat suivant :

```
1 cat /var/log/eucalyptus/registration.log
2 2010-04-08 15:46:36-05:00 | 24243 -> Calling node cluster1 node 10.1.1.75
3 2010-04-08 15:46:36-05:00 | 24243 -> euca_conf --register-nodes returned 0
4 2010-04-08 15:48:47-05:00 | 25858 -> Calling walrus Walrus 10.1.1.71
5 2010-04-08 15:48:51-05:00 | 25858 -> euca_conf --register-walrus returned 0
6 2010-04-08 15:49:04-05:00 | 26237 -> Calling cluster cluster1 10.1.1.71
7 2010-04-08 15:49:08-05:00 | 26237 -> euca_conf --register-cluster returned 0
8 2010-04-08 15:49:17-05:00 | 26644 -> Calling storage cluster1 storage 10.1.1.71
9 2010-04-08 15:49:18-05:00 | 26644 -> euca_conf --register-sc returned 0
```

Sur le grid5000, nous n'avons jamais réussi à avoir cela, pourtant en local, lors de nos tests en salle ASRALL, nous n'avons pas rencontré de difficultés particulières.

2.2.4. Installation à partir des sources

Il est également possible d'installer directement Eucalyptus à partir des sources disponibles sur le site officiel. Eucalyptus est compatible avec les distributions suivantes :

- CentOS 5
- Debian squeeze
- OpenSUSE 11
- Fedora 12

Nous avons testé cette manière de l'installer sur l'environnement Debian pour essayer de régler le problème de non reconnaissance du contrôleur de cloud qui est détaillé dans la partie suivante. Cette installation à partir des sources ne présente aucun aspect particulier mais nécessite les pré-requis suivants pour se dérouler correctement :

- Compilateurs C
- Java Developer Kit (SDK) version 1.6 ou supérieur
- Apache 1.6.5
- Fichiers de développement libc
- Fichiers de développement pthreads
- Fichiers de développement libvirt
- Fichiers de développement Axis2C et rampart (inclus dans l'archive Eucalyptus obtenu sur le site officiel)
- Fichiers de développement Curl
- Fichiers de développement openssl
- Fichiers de développement zlib

Malheureusement, cette installation n'a pas non plus réglé le problème.

2.3. Problèmes rencontrés lors de l'installation et la configuration

L'ensemble des outils ne sont pas installés par défaut Lors de l'installation des paquets et de la publication des clés SSH, nous avons eu des messages d'erreur qui expliquait un problème de synchronisation, confirmés lors du test de la commande :

```
$ sudo euca_conf --discover-nodes
```

En étudiant, les logs du contrôleur de cloud, nous constatons rapidement qu'il manque certains paquets : rsync et wget. Un simple :

```
$ sudo apt-get install rsync wget
```

a résolu ce petit soucis.

Aucun nom pour le contrôleur de cloud Un autre problème qui nous a occupé une bonne semaine est le fait que peu importe la manière dont on entrait le nom d'hôte du contrôleur de cloud, il reprenait toujours le nom «localhost.localdomain».
Nous avons donc essayé de refaire notre environnement de départ en vérifiant l'ensemble des paramètres de configuration et en utilisant kaconsole3, qui permet d'obtenir une console et d'observer le boot de son environnement. Nous avons étudié différents fichiers de logs lors de l'installation et la configuration, et nous avons constaté un bug d'Eucalyptus : il arrive parfois que les fichiers de logs ne soient pas générés correctement durant l'installation. En recherchant sur le site officiel, et plus précisément sur le bug tracker, nous observons que c'est un bug encore non corrigé.
Il est donc assez difficile de débuguer un outil sans log!! Nous nous sommes alors connecté sur le salon IRC de eucalyptus hébergé sur le serveur Freenode, mais nous n'obtenons pas plus d'informations.
Enfin, nous sollicitons l'aide de Sébastien Badia, et il constate que le problème est en fait un bug sur le site de Nancy. Comme à son habitude, très efficace, il le résout dès le lendemain et nous pouvons donc reprendre nos tests. Le soucis des logs n'est quant à lui toujours pas réglé.

Le contrôleur de cloud est introuvable Le principal problème rencontré et qui nous a empêché de poursuivre avec Eucalyptus est qu'une fois les différentes étapes d'installation et de configuration réalisées, au moment d'utiliser le cloud et de déployer des images, le contrôleur de cloud est introuvable! En effet, nous obtenons l'erreur suivante :

```
ERROR: you need to be on the CLC host and the CLC needs to be running.
```

Nous avons d'abord pensé que les services étaient mal lancés. Nous avons donc testé de les redémarrer, et nous n'avons constaté aucun problème, ceux-ci fonctionnant bien. Mais en observant de plus près et en particulier les logs de démarrage des services... quand ils existent (cf. le problème précédent), nous constatons qu'il arrive que le contrôleur de cloud s'arrête brutalement sans raison apparente.

Nous constatons également une erreur lors de l'installation des paquets lorsqu'il démarre les instances, il y a le message suivant :

```
Unknow instance :
```

Cette erreur vient du fait qu'il lance les services de la mauvaises façon, puisque Ubuntu utilise désormais upstart [10].

Après avoir parcouru la documentation, le bug tracker et fait de nombreux tests, nous arrivons à la déduction que cela pourrait être un problème lié à la version de Java installée.

Nous avons donc testé d'installer le paquet officiel de Java : Sun-Java6-jre et même la version OpenSource : openjdk-6-jre mais cela n'a rien changé.

Nous avons également essayé de recréer un environnement Ubuntu Server à partir de l'accès physique au cluster du Grid5000 dans les locaux du LORIA [11], grâce à Sebastien Badia...mais en vain, cela n'a pas réglé le soucis.

Nous avons poursuivi les expérimentations et les analyses pour essayer d'isoler au mieux le problème. Je tiens à préciser que nous avons essayé d'obtenir de l'aide de la communauté Eucalyptus et la seule réponse concrète que nous avons eu est que c'est sans doute le fait de notre plate forme particulière et que l'on peut toujours essayer de débugger le code source en Java.

En essayant plusieurs manipulations, nous nous confrontons constamment à ce type d'erreur :

10. Upstart est un remplaçant du daemon init qui se base sur les événements. Il a été écrit par Scott James Remnant, un employé de Canonical Ltd. Upstart fonctionne de manière asynchrone. De la même manière qu'il gère le lancement des tâches et daemons au démarrage et leur arrêt à l'arrêt de la machine, il les supervise pendant que le système tourne.
11. Laboratoire Lorrain de Recherche en Informatique et ses Applications

```
[error:0215]
java.lang.NoClassDefFoundError: groovy/lang/GroovyObject
   at java.lang.ClassLoader.defineClass1(Native Method)
   at java.lang.ClassLoader.defineClass(ClassLoader.java:634)
   at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
   at java.net.URLClassLoader.defineClass(URLClassLoader.java:277)
   at java.net.URLClassLoader.access$000(URLClassLoader.java:73)
   at java.net.URLClassLoader$1.run(URLClassLoader.java:212)
   at java.security.AccessController.doPrivileged(Native Method)
   at java.net.URLClassLoader.findClass(URLClassLoader.java:205)
   at java.lang.ClassLoader.loadClass(ClassLoader.java:321)
   at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:294)
   at java.lang.ClassLoader.loadClass(ClassLoader.java:266)
   at com.eucalyptus.bootstrap.SystemBootstrapper.init(SystemBootstrapper.java:128)
Caused by: java.lang.ClassNotFoundException: groovy.lang.GroovyObject
   at java.net.URLClassLoader$1.run(URLClassLoader.java:217)
   at java.security.AccessController.doPrivileged(Native Method)
   at java.net.URLClassLoader.findClass(URLClassLoader.java:205)
   at java.lang.ClassLoader.loadClass(ClassLoader.java:321)
   at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:294)
   at java.lang.ClassLoader.loadClass(ClassLoader.java:266)
   ... 12 more
[error:0390] Service exit with a return value of 1
```

Eucalyptus est censé fonctionner et être intégré à la distribution Ubuntu Server, mais son déploiement sur Grid 5000 est étonnamment difficile. Au final, nous n'avons pas réussi à nous sortir de ce problème, Lucas Nussbaum nous a donc conseillé de passer à l'étude d'OpenStack.

3. OpenStack

3.1. Introduction

OpenStack est un projet de cloud computing privé et public sous licence Apache [1]. Initialement développé par la NASA, l'agence gouvernementale qui a en charge la majeure partie du programme spatial civil des États-Unis [2]. et Rackspace Cloud, un fournisseur de plate-forme de cloud computing [3] à partir de juillet 2010, ces deux sociétés ont ensuite été rejointes par Cloud.com, Citrix Systems, Dell, enStratus, NTT Data, PEER 1, RightScale, Cloudkick, Zenoss, Limelight, Scalr, AMD, Intel, Spiceworks, Canonical et Cisco pour le développement d'OpenStack.
L'objectif d'OpenStack est de rendre le Cloud simple à mettre en œuvre et très extensible. Ubuntu, l'une des distributions Linux les plus populaires, a annoncé que sa version 11.04, baptisée Natty Narwhal, supportera nativement OpenStack.

1. La licence Apache est une licence de logiciel libre et open source.`http://www.apache.org/licenses/LICENSE-2.0.html`
2. La National Aeronautics and Space Administration.`http://www.nasa.gov/`
3. Rackspace Cloud :`http://www.rackspace.com/cloud/`

3.1.1. Aperçu des partenaires d'OpenStack

FIGURE 3.1.: Source : OpenStack Compute Administration Guide `http://docs.openstack.org/openstack-compute/admin/content/`

3.1.2. Composants d'OpenStack

OpenStack Swift OpenStack Swift permet de créer un service de stockage dans une architecture de cloud computing. Il permet de gérer une large capacité de stockage évolutive avec une redondance ainsi que le basculement entre les différents objets de stockage.

OpenStack Nova Compute OpenStack Compute permet d'administrer un contrôleur pour une application IaaS. Cela permet de gérer des machines virtuelles pour des utilisateurs ou des groupes. On peut l'utiliser pour configurer le réseau des machines virtuelles d'un projet.

OpenStack Imaging Service OpenStack Imaging Service est un système de récupération et de recherche d'images de machines virtuelles.

Exemple de l'utilisation des trois applications :

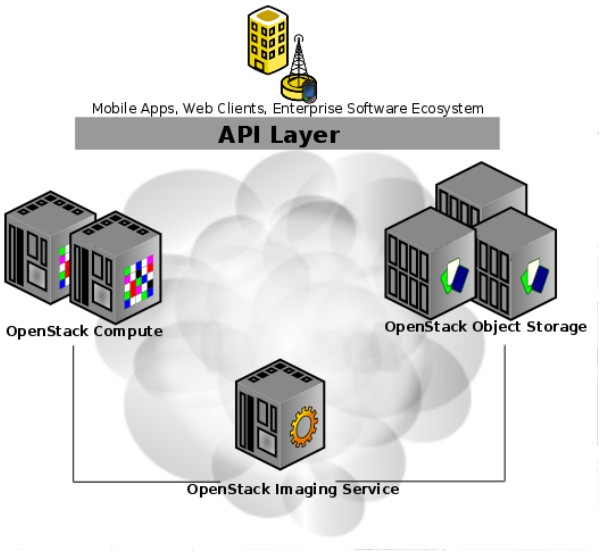

FIGURE 3.2.: Source : OpenStack Compute Administration Guide `http://docs.openstack.org/openstack-compute/admin/content/`

Pour notre projet nous utilisons OpenStack Nova Compute.

3.2. OpenStack Nova Compute

3.2.1. Prérequis

Hardware : Les composants d'OpenStack sont destinés à fonctionner sur du matériel standard.

Système d'exploitation : OpenStack tourne actuellement sur Ubuntu, la version à utiliser pour les grands déploiements est, de préférence, la 10.04 LTS. Les membres de la communauté OpenStack ont testé d'installer Nova sur CentOS et RHEL, ils ont documenté leurs démarches sur le wiki d'OpenStack. Pour ces autres distribution GNU/Linux, une installation sur CentOS 6 semble la solution la plus viable car elle ne souffre pas de problèmes de dépendances.

Réseaux : Un débit de 1000 Mbps est suggéré. Pour OpenStack Compute, le réseau est configuré sur des installations multi-noeuds entre les machines physiques sur un seul sous-réseau.

Base de données : Pour OpenStack Compute, nous pouvons installer soit PostgreSQL ou MySQL comme base de données lors du processus d'installation d'OpenStack Compute.

3.2.2. Version actuelle et versions à venir de Nova Compute

OpenStack est un projet très récent, la dernière version baptisée "Bexar" est sortie le 3 Février 2011, la prochaine baptisée "Cactus" est prévu pour le 15 Avril 2011.

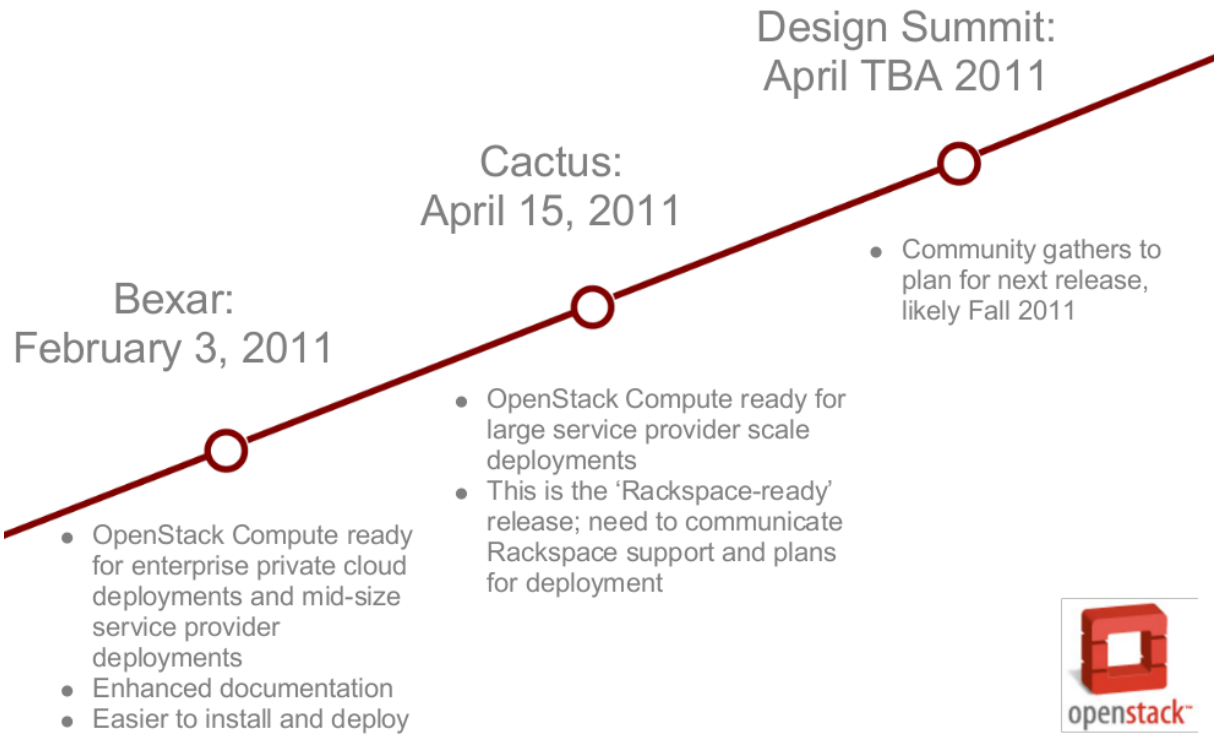

FIGURE 3.3.: Source : OpenStack Compute Administration Guide `http://docs.openstack.org/openstack-compute/admin/content/`

3.2.3. Installation de Nova Compute sur un seul serveur

Installation scriptée

Pour nos premiers pas sur Nova Compute nous nous sommes basés sur le wiki d'OpenStack[4]. Nous avons donc essayé la première installation que le wiki nous propose : une installation via un script sur un seul serveur. Nous avons d'abord lu la documentation technique qui présente ce script construit en trois étapes principales, en fonction de l'argument que l'on donne au script :

- "branch" crée un accès au Bazaar d'OpenStack[5]. Si par exemple nous exécutons la commande suivante :

```
# ./nova.sh branch lp:nova/bexar
```

  Nous obtenons un accès au Bazaar de Bexar[6].
- "install" Installe les paquets pour l'installation d'une base, la création des utilisateurs et la création d'un fichier de configuration prêt à être utilisé pour faciliter l'installation d'OpenStack.
- "run" Démarre le réseau utilisé par Nova.

Le but de ce script est de créer une architecture de Nova de ce type :

4. `http://wiki.openstack.org/InstallInstructions`
5. Bazaar (bzr en ligne de commande) est un système de gestion de versions libre sponsorisé par Canonical Ltd. Il appartient à la catégorie des systèmes de gestion de version décentralisés `http://fr.wikipedia.org/wiki/Bazaar_%28logiciel%29`
6. Bexar est la version d'OpenStack sortie le 3 Février 2011

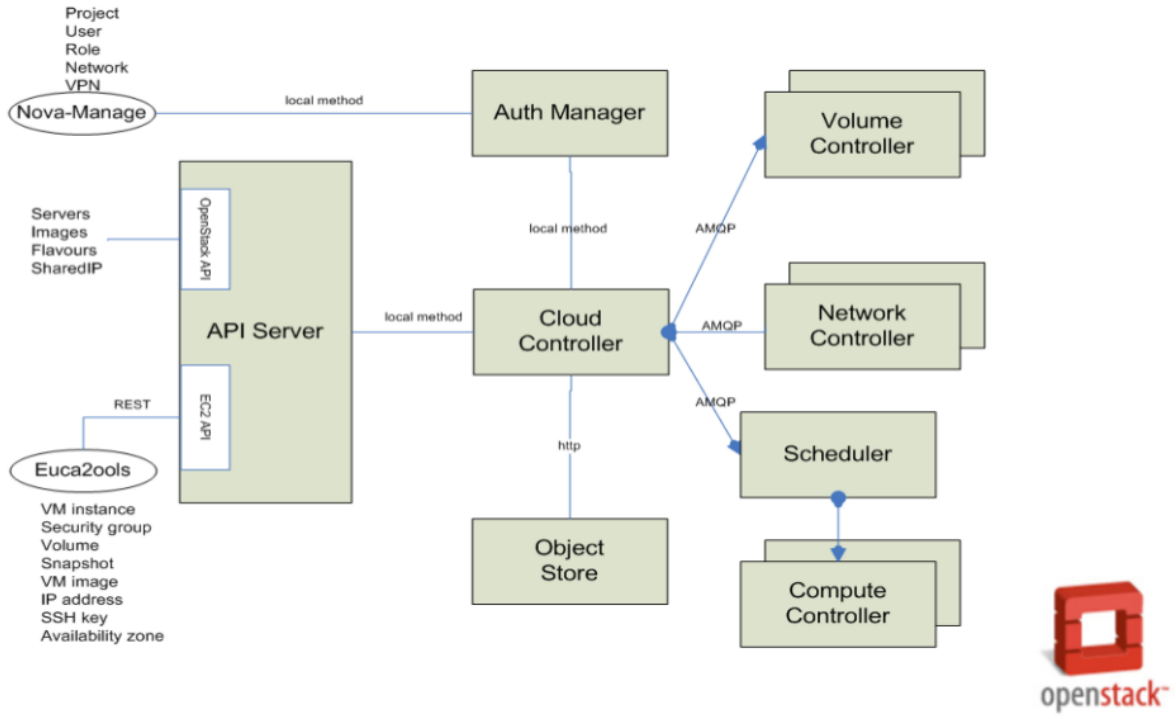

FIGURE 3.4.: Source : OpenStack Tutorial IEEE CloudCom `http://salsahpc.indiana.edu/CloudCom2010/slides/PDF/tutorials/OpenStackTutorialIEEECloudCom.pdf`

Nous avons donc testé cette installation qui s'est bien déroulée, mais pour pouvoir déployer des machines virtuelles après l'utilisation du script, nous devons configurer le réseaux utilisé par nos machines virtuelles. Nous avons ensuite abandonné l'installation d'OpenStack Compute car nous voulions utilisé une architecture avec un contrôleur et un nœud. Nous pouvons toutefois conclure que l'installation sur un seul serveur nous permet de tester rapidement l'installation d'OpenStack Compute.

3.2.4. Installation sur plusieurs serveurs manuellement

Installation du contrôleur

Obtention des paquets

OpenStack est écrit en Python, il faut donc installer au préalable les python-software-properties.
Nous utilisons Ubuntu 10.04 Lucid Lynx pour l'installation d'Openstack. Ubuntu ne supporte pas encore nativement Openstack, il faut donc que l'on rajoute le PPA[7] de Nova : `https://launchpad.net/~nova-core/+archive/trunk`

On rentre donc les commandes suivantes :

```
# apt-get install python-software-properties
# add-apt-repository ppa:nova-core/trunk
# apt-get update
```

On installe les principaux paquets de Nova :

```
$ sudo apt-get install python-greenlet python-mysqldb python-nova nova-common nova-doc nova-
    api nova-network nova-objectstore nova-scheduler nova-compute euca2ools unzip
```

7. Les Personal Package Archives (abrégés PPA) sont des dépôts de paquets logiciels offerts aux individus et aux équipes de développeurs désireux de proposer facilement leurs logiciels pour les utilisateurs d'Ubuntu..`http://doc.ubuntu-fr.org/ppa`

Configuration du fichier nova.conf

Le fichier principal de configuration de Nova se nomme nova.conf, il se situe dans /etc/nova/

Aperçu de nova.conf :

```
1 --sql_connection=mysql://root:nova@<CC_ADDR>/nova
2 --s3_host=<CC_ADDR>
3 --rabbit_host=<CC_ADDR>
4 --ec2_host=<CC_ADDR>
5 --ec2_url=http://<CC_ADDR>:8773/services/Cloud
6 --network_manager=nova.network.manager.VlanManager
7 --fixed_range=<network/prefix>
8 --network_size=<# of addrs>
```

- <CC_ADDR> correspond à l'adresse IP du contrôleur
- –sql_connection : Là ou se situe la base de données
- –s3_host : Là ou se situe le service qui contient les images des machines virtuelles
- –rabbit_host : Là ou se situe le serveur de messagerie rabbit AMQP
- –ec2_url : Là ou se situe l'interface nova-api
- –network_manager : Correspond au type de communication qu'utilise le contrôleur pour parler aux machines virtuels. Il en existe 3.
  - nova.network.manager.FlatManager
  - FlatDHCPManager
  - nova.network.manager.VlanManager
- –fixed_range=<network/prefix> : Correspond au réseaux des machines virtuelles
- –network_size=<# of addrs> : Correspond au nombre d'adresses IP utilisées pour héberger nos machines virtuelles

On crée le groupe et l'utilisateur nova pour ne pas avoir de problèmes de droits :

```
# addgroup nova
# chown -R root:nova /etc/nova
# chmod 644 /etc/nova/nova.conf
```

Installation et configuration de MySql

```
# bash
# MYSQL_PASS=nova
# cat <<MYSQL_PRESEED | debconf-set-selections
# mysql-server-5.1 mysql-server/root_password password $MYSQL_PASS
# mysql-server-5.1 mysql-server/root_password_again password $MYSQL_PASS
# mysql-server-5.1 mysql-server/start_on_boot boolean true
# MYSQL_PRESEED
```

On installe le serveur MySql :

```
apt-get install -y mysql-server
\end{listing}
\\
On modifie le fichier /etc/mysql/my.cnf :
\begin{lstlisting}
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart
```

On crée la base de données NOVA avec les droits :

```
# mysql -uroot -p $MYSQL_PASS -e 'CREATE DATABASE nova;'
# mysql -uroot -p $MYSQL_PASS -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION
    ;"
# mysql -uroot -p $MYSQL_PASS -e "SET PASSWORD FOR 'root'@'%' = PASSWORD('$MYSQL_PASS');"
```

Installation du Nœud

Installation de l'environnement du Nœud Pour installer l'environnement du Nœud on répète les deux premières étapes de l'installation du contrôleur à savoir :

- L'obtention des paquets
- La configuration du fichier nova.conf

Configuration du réseau On doit créer un pont entre le réseau de Grid5000 (sur eth0) et notre réseau de machines virtuelles sur le nœud, pour cela on doit installer le paquet bridge-utils et éditer le fichier "interfaces" situé dans /etc/network/

```
# The loopback network interface
auto lo
iface lo inet loopback

# Networking for NOVA
auto br100

iface br100 inet dhcp
        bridge_ports    eth0
        bridge_stp      off
        bridge_maxwait  0
        bridge_fd       0
```

On redémarre le service *networking* pour que nos modifications soient prises en compte.

Mise en place de l'environnement pour Nova Ici, on va créer l'environnement dont Nova a besoin pour fonctionner :

- On va synchroniser la base de donnée
- On va définir l'utilisateur principale de Nova
- On va créer le projet de l'utilisateur
- On termine par créer le réseau utilisé par Nova

```
# /usr/bin/python /usr/bin/nova-manage db sync
# /usr/bin/python /usr/bin/nova-manage user admin <user_name>
# /usr/bin/python /usr/bin/nova-manage project create <project_name> <user_name>
# /usr/bin/python /usr/bin/nova-manage network create <project-network> <number-of-networks-in
    -project> <IPs in project>
```

Création des certifications de Nova Pour que l'on puisse créer des machines virtuelles, nous avons besoin des certifications de Nova :

```
# mkdir -p /root/creds
# /usr/bin/python /usr/bin/nova-manage project zipfile \$NOVA_PROJECT \$NOVA_PROJECT_USER /
    root/creds/novacreds.zip
```

On extrait les certifications et on les ajoute à notre environnement :

```
# unzip /root/creds/novacreds.zip -d /root/creds/
# cat /root/creds/novarc >> ~/.bashrc
# source ~/.bashrc
```

On redémarre les services Pour que Nova utilise notre configuration, nous devons redémarrer les services qui lui sont associés :

```
libvirtd restart; service nova-network restart; service nova-compute restart; service nova-api
     restart; service nova-objectstore restart; service nova-scheduler restart
```

Accès aux machines virtuelles depuis le Nœud Pour pouvoir communiquer avec nos machines virtuelles depuis le nœud, on doit autoriser certains protocoles sur nos machines virtuelles (dans notre cas, nous allons autoriser le protocole SSH et le ping).
SSH est basé sur le protocole TCP connecté sur le port 22.
Le ping est une requête icmp.
On configure ces accès avec ces commandes :

```
# euca-authorize -P icmp -t -1:-1 default
# euca-authorize -P tcp -p 22 default
```

Démarrage des machines virtuelles

Préparation au déploiement Pour cette partie, nous avons préparé une image d'Ubuntu 10.10 sous forme d'archive sur le nœud qui nous servira d'image de déploiement pour nos machines virtuelles. Le lancement et la gestion des machines virtuelles se fait via les outils Euca2ools. [8]

Pour pouvoir utiliser cette image, nous devons rentrer la commande suivante :

```
# uec-publish-tarball ubuntu1010-UEC-localuser-image.tar.gz VM1 x86_64
```

Les deux arguments correspondent respectivement au nom de notre machine virtuelle et à l'architecture qu'elle va utiliser. Cette commande nous retourne trois références : emi, eri et eki, il faut utiliser la valeur de emi pour les prochaines étapes de déploiements.
Nous voulons communiquer en ssh avec nos machines virtuelles donc il faut créer une paire de clé :

```
# euca-add-keypair mykey > mykey.priv
# chmod 0600 mykey.priv
```

Déploiement des machines virtuelles Maintenant que toutes les étapes ont été réalisées, nous pouvons déployer nos machines virtuelles basées sur l'image d'Ubuntu précédemment citée :

```
# euca-describe-images
# euca-run-instances $emi -k mykey -t m1.tiny
```

8. http://open.eucalyptus.com/wiki/Euca2oolsGuide_v1.1

3.2.5. Installation scriptée sur plusieurs serveurs

Présentation des scripts Les développeurs d'OpenStack ont prévu des scripts d'installation pour le contrôleur et le nœud. Le principal soucis des scripts est que cela ne nous explique pas l'installation et la configuration de Nova en détail. Pour l'installation par scripts, on doit lancer le script du contrôleur, en annexe. Celui-ci installe les paquets de Nova et configure le fichier nova.conf ainsi que l'environnement euca2ools.

Ensuite, on doit copier le nova.conf sur le nœud et on peut lancer le script d'installation de l'environement du nœud, en annexe.

3.3. Problèmes rencontrés

Choisir le type d'installation d'OpenStack Le site d'Openstack nous propose plusieurs types d'installation pour Nova :

- installer Nova sur une seule machine
- installer Nova sur plusieurs machines avec les outils de déploiements d'OpenStack
- installer Nova sur plusieurs machines manuellement
- installer Nova sur plusieurs machines par des scripts

Un des premiers problèmes que nous avons rencontré était de choisir le type d'installation de Nova. Notre but était de créer un système du même type que Eucalyptus : c'est à dire un contrôleur et un nœud, mais au vu des problèmes que nous a posé ce type de solution pour Eucalyptus, nous avons voulu essayer une installation sur une seule machine.
Le problème est que OpenStack propose plusieurs types d'installation sur une seule machine :

- installation via des scripts
- installation via les packages
- installation via l'utilitaire Bexar

Ces différentes installations présentent l'avantage de pouvoir installer Nova rapidement mais il n'y a aucune explication sur ce que les scripts installent et sur la configuration de Nova.
Pour mieux comprendre l'installation de Nova, nous avons donc choisi une installation manuelle sur plusieurs serveurs.

Problèmes de PPA et de création de bridge Lors de l'installation de Nova Compute, nous avons rencontré plusieurs problèmes. L'ajout du PPA nous renvoyait un message d'erreur de ce type :

```
# add-apt-repository ppa:nova-core/trunk
Error reading https://launchpad.net/api/1.0/~nova-core/+archive/trunk: <urlopen error [Errno
    110] Connection timed out>
```

Nous avons essayé d'ajouter le PPA sur une autre image et il n'y avait pas ce message. Après avoir comparé les fichiers de configuration d'APT sur les deux environnements, nous nous sommes rendu compte que le PPA avait bien été ajouté : même avec ce message d'erreur, le PPA est ajouté. Le second problème que nous avons rencontré se situe lors de la création du brigde. Après avoir édité le fichier /etc/network/interfaces, nous devions redémarrer le service networking mais nous avons eu le message d'erreur suivant :

```
1  # /etc/init.d/networking restart
2   * Reconfiguring network interfaces...                                    Ignoring
        unknown interface eth0=eth0.
3  Internet Systems Consortium DHCP Client V3.1.3
4  Copyright 2004-2009 Internet Systems Consortium.
5  All rights reserved.
6  For info, please visit https://www.isc.org/software/dhcp/
7
8  SIOCSIFADDR: No such device
9  br100: ERROR while getting interface flags: No such device
10 br100: ERROR while getting interface flags: No such device
11 Bind socket to interface: No such device
12 Failed to bring up br100.
```

Notre bridge nous permet de relier une interface réseau physique à une interface réseau virtuelle, pour pouvoir le créer nous avons installé le paquet bridge-utils qui permet de créer et gérer des bridges. Même avec ce paquet notre bridge n'était pas créé... Nous avons étudié les journaux du système avec les commandes syslog [9] et modprobe [10] qui nous ont indiqué qu'il manquait un module [11] intitulé "bridge-module" au niveau du noyau.

Un petit schéma pour mieux comprendre :

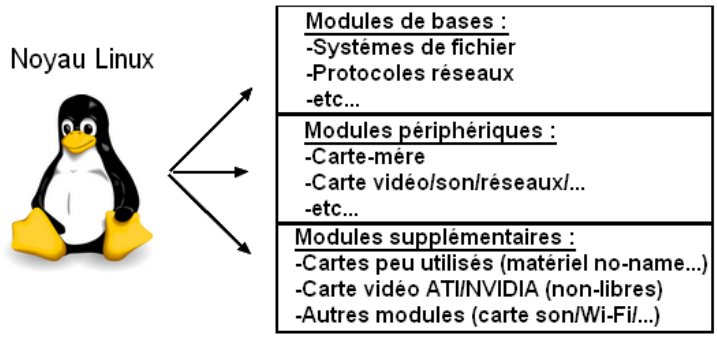

FIGURE 3.5.: Source : Les modules Linux `http://doc.ubuntu-fr.org/tutoriel/tout_savoir_sur_les_modules_linux`

Vu que nous étions en retard au niveau de notre planning, nous n'avons pas essayé de corriger cette erreur et nous avons donc essayé de créer un bridge sur un autre environnement, ce qui a fonctionné. En fait le principal problème était de créer un environnement Kadeploy Ubuntu sous lequel nous pouvions créer un bridge et ajouter le PPA.

La mise en réseau des machines virtuelles Lorsque nous déployons nos machines virtuelles sur Grid 5000, elles n'ont pas d'adresse IP. Si nous éxécutons la commande suivante :

```
# euca-describe-instances
```

Nous obtenons le résultat suivant :

```
# euca-describe-instances
RESERVATION r-kow5gx5o toto default
INSTANCE i-00000001 ami-7b006df8 networking mykey (toto, localhost.localdomain) 0 m1
    .tiny 2011-03-25T09:21:08Z nova
```

Sur Grid5000, les machines virtuelles sont sur un réseau réservé exclusivement pour la virtualisation. Ce réseau utilise un DHCP qui attribue des addresses IP aux machines virtuelles si elles comportent une adresse mac avec un préfixe valide. Pour résoudre notre problème nous supposons que nous devons créer nos propres images de machines virtuelles en local pour leur attribuer un préfixe mac. Vu le manque de temps ce problème n'a pas été résolu pour l'instant.

9. Syslog est un protocole définissant un service de journaux d'événements d'un système informatique.`http://fr.wikipedia.org/wiki/Syslog`
10. modprobe permet de charger des modules du noyau Linux dynamiquement`http://doc.ubuntu-fr.org/tutoriel/tout_savoir_sur_les_modules_linux`
11. Un module est un morceau de code permettant d'ajouter des fonctionnalités au noyau : pilotes de périphériques matériels, protocoles réseaux, etc...`http://doc.ubuntu-fr.org/tutoriel/tout_savoir_sur_les_modules_linux`

4. OpenNebula

4.1. Introduction

OpenNebula est un projet Open Source de cloud computing de type IaaS. Le projet a été lancé en 2005, la première version stable est sortie en 2008. Le projet est sous licence Apache 2 [1]. Le projet a pour but la gestion de machines virtuelles à grande échelle sur des infrastructures distribuées ou de cluster, et supporte plusieurs technologies d'hyperviseur : Xen, KVM et VMware. OpenNebula permet aussi de combiner les infrastructures locales et publiques, ce qui offre une grande modularité aux environnements hébergés.
OpenNebula est intégré dans Debian Sid, Ubuntu Natty et OpenSuse. OpenNebula 2.2 Bêta 1 est sortie le 02/06/2011. Dans le cadre du projet, nous avons utilisé la dernière version stable disponible, la 2.0 parue le 24/10/2010.

La documentation, assez fournie, mais parfois assez brute de fonderie, est disponible sur le site officiel :
- documentation générale : `http://opennebula.org/documentation:documentation`
- référence : `http://opennebula.org/documentation:references`

FIGURE 4.1.: Aperçu des différents partenaires, utilisateurs et projets vedettes de OpenNebula

1. La licence Apache est une licence de logiciel libre et open source.`http://www.apache.org/licenses/LICENSE-2.0.html`

4.2. Fonctionnement

Dans le cadre du projet, nous utilisons OpenNebula pour créer un cloud privé. Pour déployer ce type de cloud, OpenNebula adopte une architecture classique de type cluster-like avec un frontend et un ou plusieurs nœuds qui exécutent et hébergent les machines virtuelles, avec un réseau physique reliant le frontend aux nœuds.

Frontend

Le frontend est la machine qui gère toutes les autres : les nœuds ou les VM. Toute la gestion du cloud se fait sur cette machine.

FIGURE 4.2.: Illustration du frontend

Le frontend permet de gérer tous les service du cloud, notamment celui qui gère le cycle de vie des VM ou encore les sous-système comme les réseaux virtuels, le stockage et les nœuds.
L'administration du cloud se fait avec le compte utilisateur *oneadmin*. Pour la création d'une VM, il faut au minimum un fichier *.one* de définition de cette VM. Dans ce fichier on trouve notamment le chemin de l'image(fichier img, qcow2...), la spécification des interfaces réseaux, les informations de contextualisation, etc.

Nœuds

Les nœuds sont les machines qui hébergent les machines virtuelles, c'est à dire des serveurs hôtes de virtualisation.

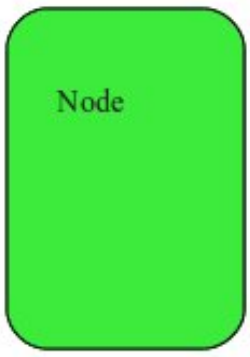

FIGURE 4.3.: Illustration d'un nœud

Chaque nœud dispose des trois éléments suivants :
- un hyperviseur, c'est l'hyperviseur qui permet la virtualisation de plusieurs machines virtuelles sur une seule machine physique. Les hyperviseurs utilisés par OpenNebula sont Xen, KVM et VMware ;
- un bridge, le bridge permet de relier les interfaces réseaux virtuelles des VM au l'interface réseaux physique de la node ;
- un serveur ssh, OpenNebula utilise ssh notamment pour copier les images des machines virtuelles.

Fonctionnement

Dans un premier temps, on doit renseigner le frontend en lui indiquant les nœuds qu'il va pourvoir utiliser. Pour la gestion des nœuds on utilise la commande **onehost**. Cette commande permet d'ajouter des nœuds au cloud, de les lister ou de les supprimer.

Ajouter un nœud au cloud :

```
$ onehost add node01 im_kvm vmm_kvm tm_ssh
```

Cette commande permet d'ajouter le nœud *node01* au cloud, im_kvm et vmm_kvm précise que l'on utilise kvm comme hyperviseur et tm_ssh indique que l'on copie les images des VM par ssh. Ces trois paramètres sont en fait le nom des *drivers* que l'ont souhaitent utiliser.

Lister les nodes :

```
$ onehost list
```

Supprimer une node.

```
$ onehost delete node01
```

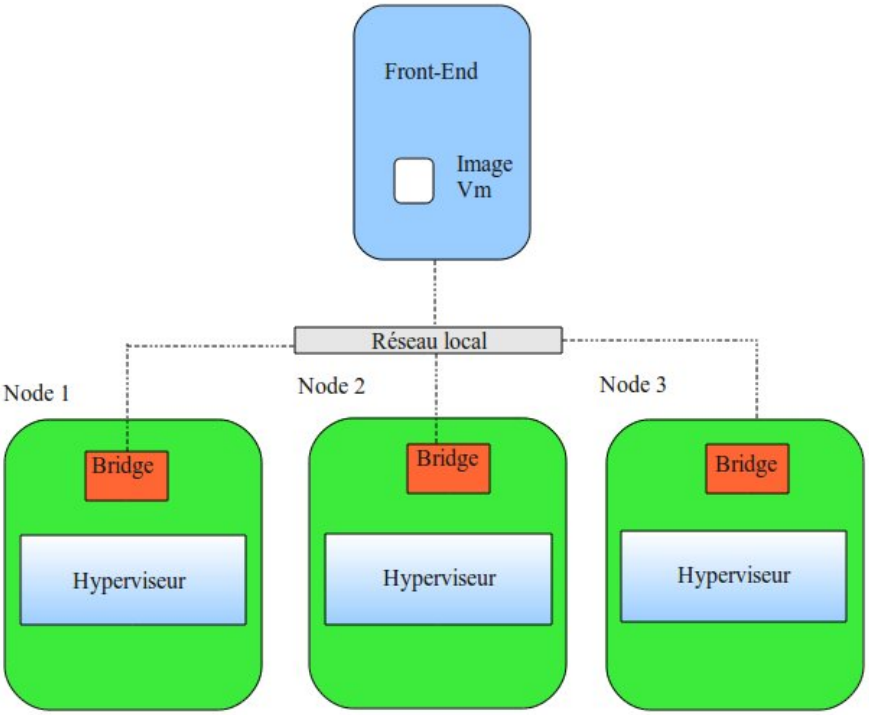

FIGURE 4.4.: Une structure composée d'un frontend et de nœuds

Après avoir pris connaissance des nœuds, on peut alors déployer des VM. Pour la gestion des VM, on utilise la commande **onevm**. Cette commande permet de créer des VM, de les lister, de donner des informations complémentaires ou de les supprimer.

Création d'une VM :

```
$ onevm create machine_virtuelle.one
```

La répartition des VM sur les nœuds se fait de manière automatique, OpenNebula va déployer la VM sur un nœud disposant de suffisamment de ressources pour l'héberger. machine_virtuelle.one est un fichier descriptif de la VM, vous trouverez plus d'information sur ce fichier dans l'annexe « *"Création de machines virtuelles et fichiers de définition"* ».

Lister les VM :

```
$ onevm list
```

Récuperer des information sur une VM :

```
$ onevm show ID_VM
```

L'ID de la VM est donné par la commande onevm list.

Supprimer une VM :

```
$ onehost delete ID_VM
```

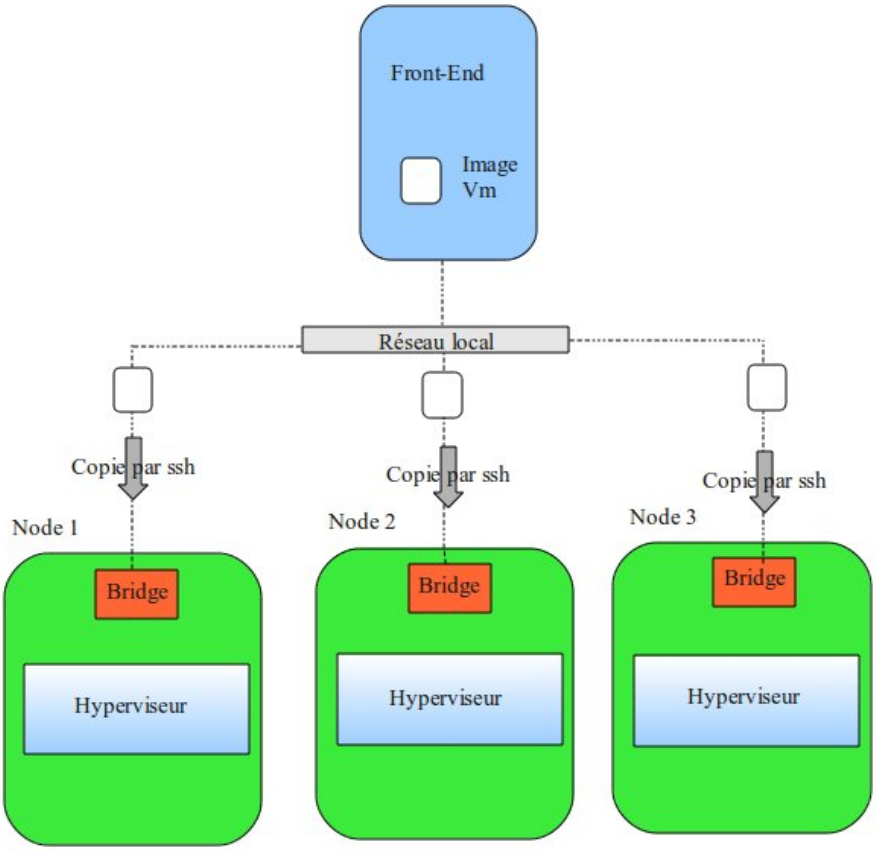

FIGURE 4.5.: Déploiement de VMs par SSH sur les nœuds

Pendant le déploiement d'une VM sur une node, une image disque est copiée du frontend vers un nœud, une interface réseau virtuelle est créée et attachée au bridge puis la VM est démarrée.

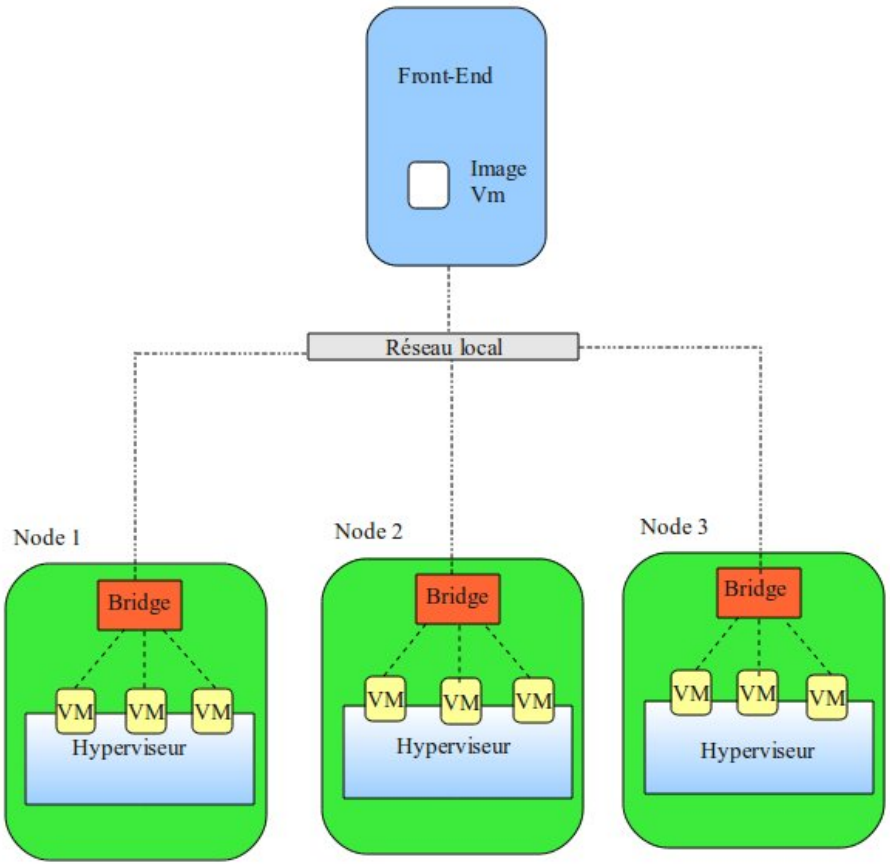

FIGURE 4.6.: Déploiement finalisé

Pour aller plus loin, d'autres commandes sont disponibles :
– **oneimage** : permet d'enregistrer des images de machines virtuelles avec leur configuration ;
– **onevnet** : permet d'administrer des réseaux virtuels ;
– **onecluster** : permet de regrouper des machines pour former des clusters.

4.3. Installation

Pour déployer OpenNebula 2 nous avons décidé d'utiliser comme support une Ubuntu Lucid Lynx 10.04 LTS. Pour cela nous sommes parti d'une image de Ubuntu Karmic disponible sur le Grid que nous avons mise à jour et fait passer en Lucid. Nous avons décidé d'utiliser cette distribution car elle supportait plus facilement Eucalyptus et permettait également d'installer OpenNebula via un PPA. Les deux groupes sont donc parti de cette image mise à jour comme support.

4.3.1. Installation du Frontend

Il existe plusieurs façons d'installer OpenNebula : par une archive disponible sur le site officiel ou par les paquets disponibles pour Ubuntu. Dans le cadre du projet nous avons donc décidé de passer par les paquets, ce qui nous a rendu la tâche d'installation beaucoup plus aisée (aucun problème de dépendances, pré et post-configuration automatique, etc.)

Les fichiers README inclus aux paquets Debian *opennebula*[2] et *opennebula-node*Fichier README du paquet Debian *opennebula-node* nous ont été d'une très grande aide pour l'installation.

Premièrement, OpenNebula 2 est packagé dans un PPA pour Ubuntu Lucid, nous avons donc ajouté le PPA dans le fichier /etc/apt/sources.list :

```
deb http://ppa.launchpad.net/opennebula-ubuntu/ppa/ubuntu intrepid main
```

Puis nous avons installé le paquet opennebula :

```
#apt-get install opennebula
```

2. Fichier README du paquet Debian *opennebula*

Voici la liste des dépendance du paquet opennebula : *libruby1.8 libsqlite3-ruby libsqlite3-ruby1.8 libxmlrpc-c3 libxmlrpc-core-c3 opennebula-common ruby ruby1.8*

Attention, il se peut qu'une dépendance nécessaire pour contextualiser les machines virtuelles soit manquante : *mkisofs*

4.3.2. Configuration du Frontend

Lors de l'installation, un compte utilisateur oneadmin est créé. C'est depuis ce compte que toutes les commandes pour gérer le Cloud sont lancées. Il nous faut maintenant le configurer en éditant le fichier */var/lib/one/one_ auth*.

```
# su - oneadmin
$ vi $HOME/.one/one_auth
```

Dans ce fichier, entrez la ligne suivante en changeant le mot de passe :

```
oneadmin:<password>
```

Dans un premier temps, nous avions oublié de créer ce fichier, ce qui occasionnait l'erreur suivante :

```
/usr/lib/one/ruby/OpenNebula.rb:77:in `initialize': ONE_AUTH file not present (RuntimeError)
        from /usr/lib/one/ruby/client_utilities.rb:239:in `new'
        from /usr/lib/one/ruby/client_utilities.rb:239:in `get_one_client'
        from /usr/bin/onehost:343
```

Par défaut, le service opennebula ne démarre pas automatiquement. Nous avons donc fait les actions suivantes.

Rendre exécutable le script /etc/init.d/opennebula :

```
# chmod +x /etc/init.d/opennebula
```

Ajouter le service en démarrage automatique pour tous les runlevels :

```
update-rc.d opennebula defaults
```

Nous avons par la suite configuré le serveur SSH pour désactiver la vérification strictes des clés des serveurs présentes dans le fichier *$HOME/.ssh/known_ hosts*. Par défaut, SSH est en mode "ask", ce qui demande une interaction de la part de l'utilisateur, causant problème pour l'exécution de scripts automatisés.

Dans le fichier *$HOME/.ssh/config*

```
Host *
   StrictHostKeyChecking no
```

Par défaut, lors de la copie via *scp* d'une image disque de machine virtuelle du frontend vers un nœud, OpenNebula tente de se connecter à son serveur SSH local et demande le mot de passe. Pour supprimer cette interaction, il faut ajouter la clé publique du compte oneadmin au fichier *authorized_ keys*.

```
# su - oneadmin
$ cd $HOME/.ssh/
$ cat id_rsa.pub >> authorized_keys
```

Après avoir suivi toute ces étapes, nous avons enregistré un environnement sous le nom *lucid-opennebula-frontend* pour pouvoir la redéployer avec kadeploy3 et travailler sur le Grid.

4.3.3. Installation d'un nœud

Comme pour le contrôleur, nous avons ajouté le dépot PPA correspondant :

```
deb http://ppa.launchpad.net/opennebula-ubuntu/ppa/ubuntu intrepid main
```

Puis nous avons installé le paquet opennebula-node

```
# aptitude install opennebula-node
```

Il faut aussi installer les paquets pour le serveur ssh et la manipulation de bridge si nécessaire (*bridge-utils*).

```
#aptitude install libvirt0 openssh-server
```

4.3.4. Configuration d'un nœud

Vérifications

Chaque nœud est un serveur de virtualisation, devant disposer d'un hyperviseur. Il nous faut vérifier que l'hyperviseur choisi fonctionne correctement sur les nœuds, dans notre cas nous utilisons KVM [3]. Pour cela, il faut que le processeur supporte la virtualisation (vmx pour Intel, svm pour AMD, normalement disponible sur tous les nœuds de grid5000).

```
# egrep '(vmx|svm)' /proc/cpuinfo
```

Il faut vérifier que les modules noyau kvm et kvm-intel ou kvm-amd sont bien chargés :

```
# lsmod | grep "kvm\(.*\)"
```

Configuration SSH

Pour autoriser les connexions SSH du contrôleur vers la node, il faut copier la clé publique SSH du compte oneadmin du frontend dans le fichier *authorized_keys* du compte oneadmin de la node. La documentation Debian nous indique les commandes suivantes :

```
root@controller> su - oneadmin
oneadmin@controller> cat $HOME/.ssh/id_rsa.pub
<COPY YOUR CONTROLLER SSH PUBKEY>

root@node01> su - oneadmin
oneadmin@node01> vi $HOME/.ssh/authorized_keys
<PAST YOUR CONTROLLER SSH PUBKEY>
```

Arrivé à ce point, un nœud est correctement configuré. Nous avons enregistré l'environnement sous le nom *lucid-opennebula-node*.

Il reste cependant à créer un bridge, et à y attacher l'interface réseau ethernet principale. OpenNebula reliera les interfaces réseaux virtuelles de chaque machine virtuelle à ce bridge, leur permettant de communiquer avec le réseau.

4.3.5. Problèmes rencontrés pour l'installation et la configuration des environnements

Il nous a été difficile de créer un environnement fonctionnant bien sur tous les sites de Grid 5000 à cause de quelques particularités, nous allons les expliquer ici.

Accès au dépôt PPA

Nous avons rencontré des problèmes lors de l'ajout des PPA, en effet un problème de proxy nous empêchait de télécharger les paquets de OpenNebula. Nous avons donc ouvert un bug dans le Bugzilla pour demander son ouverture.

Dépôt PPA Ubuntu obsolète, passage à Debian SID

Lors de nos tests, nous avons rencontrés des bugs de OpenNebula. Après recherche, ceux-ci étaient déjà corrigés dans une version plus récente de OpenNebula. Nous sommes alors passés à Debian SID et nous avons enregistré deux nouveaux environnements finalisés et fonctionnels : *sid-opennebula-frontend* et *sid-opennebula-node*.

Configuration réseau des nœuds : création du bridge et interface réseau

L'interface utilisée pour le réseau de production change d'un site à l'autre. Une des solutions est d'ajouter un fichier de configuration réseau dans la *postinstall* [4] de notre environnement. Il est alors possible d'utiliser la variable NET_DEV (interface réseau principale) afin de créer une configuration réseau universelle incluant un bridge. Cependant, cette solution était en cours de déploiement [5].
Nous alors avons choisi une solution de contournement, configurer le réseau dans le fichier */etc/rc.local* (cf. Annexes). Ce script est exécuté après tous les autres scripts dans tous les runlevel. Il détecte l'interface réseau principale utilisée, puis créer un bridge nommé *br0* et y attache cette interface avec la commande *brctl*.

3. KVM : Kernel-based Virtual Machine, technologie de virtualisation intégrée au noyau Linux.
4. grid5000.fr : Prepost
5. Rapport de bug sur le bugzilla concernant la variable NET_DEV

Problèmes d'espace disque disponible

L'espace disque disponible à la racine de chaque serveur est uniquement de quelques Go, ce qui occasionne rapidement des problèmes d'espace disque lors du déploiement de multiples machines virtuelles volumineuses sur le même nœud.

Nous avons résolu le problème rapidement en ajoutant un script /etc/rc.bindhome (cf. Annexes) pour contourner ce problème. Celui-ci monte avec l'option *bind* le répertoire /tmp/one dans le répertoire où sont placées les images disques (/var/lib/one).

Ce script doit obligatoirement être exécuté avant celui lançant le service opennebula dans le runlevel 2.

4.4. Création de machines virtuelles et fichiers de définition

Pour la création de VM (commande *onevm create*) et de réseaux (commande *onevnet create*), il faut créer avant tout une image disque avec un système installé et des fichiers de définitions.

4.4.1. Image disque de machine virtuelle

Création d'une image disque

Nous avons utilisé deux solutions, la première, faire une installation classique avec qemu, et la deuxième, utiliser vmbuilder.

Qemu Avec qemu-img, on crée une image de 4Go au format qcow2. Le format qcow2 apporte des avantages sur le format *raw* dont la compression.

```
qemu-img create debian_squeeze.qcow2 4G
```

On lance ensuite qemu, avec cette image disque et l'iso du CD d'installation de Debian pour faire une installation standard.

```
qemu-system-x86_64 -hda debian_squeeze.img -cdrom debian-6.0.0-amd64.iso -boot once=d -m 512
```

VM Builder VMBuilder[6] est un utilitaire écrit en Python permettant de créer rapidement des images minimales de machines virtuelles. Il gère Debian et Ubuntu, cependant le support de Debian ne semble pas à jour et son utilisation est beaucoup plus simple pour Ubuntu.
Pour nos essais, nous avons donc créé une image de Ubuntu Lucid de cette façon :

```
ubuntu-vm-builder kvm lucid --arch amd64 --addpkg=openssh-server --rootpass=grid5000
```

Les premiers paramètres sont l'hyperviseur cible (kvm), puis la distribution (ubuntu lucid).
Nous ajoutons ensuite un paquet : le serveur openssh.
L'option rootpass permet de spécifier un mot de passe root.

Configuration

En premier lieu, il faut supprimer les fichiers de règles udev concernant les interfaces réseaux (situées dans */etc/udev/rules.d*). Celles-ci fixent les interfaces réseaux en fonction des adresses MAC, ce qui est problématique pour une image de machine virtuelle générique.
Il faut ensuite permettre la contextualisation de la machine virtuelle pour OpenNebula. Il s'agit d'une fonctionnalité permettant de personnaliser les machines virtuelles, en leur attachant un nouveau disque disposant d'un système de fichiers au format ISO 9660. Celui-ci contient un script init.sh pour paramétrer la machine virtuelle et des fichiers complémentaires (fichier de variables context.sh, clés SSH)
Attention, quelque soit le périphérique cible donné par OpenNebula pour l'iso de contexte, le système détecte un lecteur optique à cause du système de fichier, et le nomme sr0 (/dev/sr0).

```
1 mount -t iso9660 /dev/sr0 /mnt
2
3 if [ -f /mnt/context.sh ]; then
4   . /mnt/init.sh
5 fi
6
7 umount /mnt
```

6. SitedeVMBuilder:https://launchpad.net/vmbuilder

4.4.2. Fichier de définition de machine virtuelle

Ce fichier permet de définir une machine virtuelle. On peut ensuite l'instancier avec la commande *onevm create fichier.one*. Il contient des variables générales. Celles-ci sont listées dans la documentation de OpenNebula[7].

```
NAME   = myUbuntu
CPU    = 0.5
MEMORY = 400

DISK   = [
 source   = "/var/lib/one/lucid/ubuntu-lucid.qcow2",
 target   = "sda",
 readonly = "no",
 driver = "qcow2" ]

OS = [ ARCH = "x86_64" ]

NIC = [ BRIDGE = "br0" ]

FEATURES = [ acpi="no" ]

CONTEXT = [
 files = "/var/lib/one/lucid/init.sh /var/lib/one/.ssh/id_rsa.pub",
 id = "$VMID",
 target = "hdc",
 root_pubkey = "id_rsa.pub",
 username = "oneadmin",
 user_pubkey = "id_rsa.pub"
]
```

- NAME : nom de la VM
- CPU : nombre de processeurs, 0.5 correspond à 50% d'un processeur
- MEMORY : mémoire allouée en Mo
- DISK : définition de l'image disque du système, du nom du périphérique dans /dev, etc. Il faut aussi spécifier une variable driver pour utiliser un autre format d'image (comme qcow2).
- NIC : définition du réseau, possibilité d'utiliser un réseau virtuel existant de OpenNebula (NETWORK), ou de définir uniquement une interface avec un bridge auquel relier l'interface de la VM.
- OS : information sur l'OS contenu dans l'image disque, il est nécessaire de préciser l'architecture.
- CONTEXT : section permettant la contextualisation de la machine virtuelle.

La contextualisation de la machine virtuelle est une fonction essentielle de configuration. Elle permet de spécifier un script à exécuter au lancement de la VM, et de lui passer des paramètres. Pour ce faire, les fichiers contenus dans la variable files sont archivés dans un fichier iso. Ce fichier iso est monté au lancement de la VM, et le script init.sh est appelé. Ce script dispose en variable d'environnement de toutes les variables définies dans le fichier de définition de la VM. Il va alors fixer lui même l'adresse IP et la configuration réseau, les clés SSH, etc.

4.4.3. Fichier de définition de réseau virtuel

Il est possible (et facultatif), de créer un réseau virtuel dans OpenNebula, pour lui déléguer la gestion de la configuration et réseau et l'attribution automatique d'IP à chaque VM. La commande à utiliser pour créer le réseau est *onevnet create mon_reseau.net*

```
NAME            = "BigNetwork"
TYPE            = "RANGED"
BRIDGE          = "br0"
NETWORK_SIZE    = 250
NETWORK_ADDRESS= "10.144.0.0"
```

Ce fichier de définition est très court et définit une configuration réseau pour un groupe de VM.

- NAME nomme le réseau.
- TYPE peut avoir deux valeurs : RANGED et FIXED. RANGED permet de définir un bloc d'IP et les deux variables suivantes en dépendent. La valeur FIXED permet de définir une liste d'adresses IP.
- NETWORK_SIZE, nombre de machines dans le réseau
- NETWORK_ADDRESS, adresse du réseau à utiliser

Avec cette configuration, OpenNebula va attribuer une IP dans le réseau 10.144.O.0 à chaque VM utilisant le réseau *BigNetwork*.

Voici un exemple pour un réseau de type FIXED.

7. opennebula.org : Virtual Machine Definition File 2.0

```
NAME = "Small network"
TYPE = FIXED

BRIDGE = br0
LEASES = [ IP="192.168.0.5"]
LEASES = [ IP="192.168.0.6"]
LEASES = [ IP="192.168.0.7"]
```

Les IP disponibles sont listées dans LEASES.

4.5. Déploiement automatisé sur Grid 5000

Nous avons écrit des scripts pour automatiser chaque étape du déploiement d'un cloud OpenNebula. Le prérequis est d'avoir transféré deux environnements et les enregistré avec kaenv3 sur tous les sites. Ces deux environnements se nomment :

- **sid-opennebula-frontend** : frontend OpenNebula
- **sid-opennebula-node** : nœud OpenNebula

Tous les scripts suivants sont fournis dans les annexes. L'idéal est de les enregistrer et de les exécuter dans un répertoire identique.

4.5.1. oargridsubON.sh : réservation de nœuds sur plusieurs sites

Ce script permet de réserver des noeuds sur plusieurs sites et différents cluster. Il lit le fichier *cluster-test* dans le répertoire courant pour avoir la liste des cluster et le nombre de nœuds souhaités par cluster.
Voici un exemple de ce fichier :

```
sagittaire;1
griffon;3
paradent;10
pastel;5
violette;3
```

Chaque ligne est un couple associant le nom d'un cluster ou d'un site à un nombre de nœuds. Il est possible de changer la valeur de la variable *DURATION*
En résultat, il lance la commande *oargridstat*. Celle-ci retourne un **GRID_JOB_ID** en cas de réussite.
Notez qu'il est possible de préciser des propriétés nécessaires aux nœuds souhaités. Il est ainsi possible de spécifier que nous souhaitons uniquement des nœuds supportant la virtualisation. La syntaxe de la commande *oargridsub* ressemblerait donc à ceci :

```
oargridsub -w \'02:00:00\' violette:rdef=/nodes=3:prop=virtual='amd-v' OR virtual='ivt',pastel
    :rdef=/nodes=5:prop=virtual='amd-v' OR virtual='ivt',paradent:rdef=/nodes=10:prop=virtual
    ='amd-v' OR virtual='ivt',griffon:rdef=/nodes=3:prop=virtual='amd-v' OR virtual='ivt',
    sagittaire:rdef=/nodes=1:prop=virtual='amd-v' OR virtual='ivt' -t deploy
```

Cependant, nous n'avons pas pu tester cette fonctionnalité.

4.5.2. kadeployON.sh : déploiement avec kadeploy3 sur les machines réservées

Ce script prend en unique paramètre le **GRID_JOB_ID**. Il établit une liste des machines réservées dans le fichier tmp_gridstat. Il déploie ensuite avec *kadeploy3* un environnement pour le frontend OpenNebula sur une machine, et un environnement pour les noeuds sur toutes les autres. Toutes les commandes sont lancées par SSH sur les sites auxquels appartiennent les nœuds visés.

4.5.3. pingON.sh : vérification de la connexion réseau des machines déployées

Ce script affiche une liste des nœuds déployés et leur état. Il est très souvent nécessaire d'attendre plusieurs minutes pour que tous les nœuds récupèrent une adresse IP par DHCP.

4.5.4. configureON.sh : configuration automatique du frontend OpenNebula

Ce script se contente d'ajouter tous les nœuds OpenNebula au cloud, en exécutant sur le frontend une série de commandes "textitonehost create node im_kvm vmm_kvm tm_ssh. Il est nécessaire de modifier la valeur de la variable **sshpubkey** par le chemin de votre clé publique SSH. Lors de la première connexion SSH au frontend en root, il sera nécessaire de taper le mot de passe : **grid5000**

4.5.5. creationvmON.rb : création des fichiers de base pour une VM

Ce script permet de créer une archive contenant l'image de la VM, le fichier de description de la VM et le script init.sh.

Contextualisation et init.sh La difficulté est la configuration réseau qui n'est pas fixe sur tous les sites (réseaux pour les machines virtuelles et passerelles) ce qui pose problème pour un déploiement sur plusieurs sites. Auparavant, nous utilisions les capacités de gestion de réseau intégrées à OpenNebula pour l'attribution automatique des IP (vnet), et nous définissions la passerelle dans le script de contexte (init.sh). Mais chaque site possède sa propre configuration réseau spécifique, et OpenNebula n'est pas assez souple pour gérer le cas d'un Cloud distribué sur plusieurs sites hétérogènes.
La solution que nous avons trouvé est d'utiliser le DHCP de chaque site. Le DHCP attribue une IP automatiquement aux machines virtuelles, et donnent également le reste de la configuration réseau comme la passerelle par défaut. Pour cela nous avons modifié le script init.sh pour spécifier une adresse mac correspondant à l'ID de la machine virtuelle et pour qu'il fasse une requête DHCP sur le réseau de production.

4.5.6. deployvmON.sh : déploiement des machines virtuelles

Ce script interactif copie l'archive générée par le script creationvmON.rb sur le frontend et déploie les VM en exécutant sur une série de commandes *onevm create ficher.one*.

5. Comparatif

	Eucalyptus	OpenStack	OpenNebula
Stockage	Très bien : un contrôleur principal *walrus* et des contrôleurs de stockage sur chaque nœud.	Bien : juste l'essentiel, espace de stockage partagé.	Neutre : par défaut, copie des machines virtuelles par SSH, gestion d'un répertoire partagé via NFS et possibilité d'utiliser une solution complémentaire comme un système de fichiers distribué.
Hyperviseur	Xen, KVM	Xen, KVM	Xen, KVM, VMware
Réseau	Limité, non prévu pour un déploiement sur un réseau complexe.	Sans opinion, nos tests ne nous ont pas permis d'émettre un avis.	Avantages et inconvénients, automatisé pour un réseau simple, manuel via contextualisation sur un réseau complexe.
Orientation	Cloud public et privé	Cloud public et privé	Cloud privé
Sécurité	Bien, chaque contrôleur est authentifié par clé SSH et fichiers de permission pour authentifier toutes les transactions.	Très bien, authentification utilisateur par clé, utilisation des Euca2ools.	Neutre, prévu pour des utilisateurs de confiance (besoin d'accès au frontend pour l'administration via libvirt ou les commandes one*)
Installation	Problématique, dépend de l'environnement réseau et matériel, difficulté en environnement hétérogène.	Facile, installation automatisée et documentée.	Manuelle, installation facile sur les distributions supportées (dont Debian et Ubuntu).
Maturité	Aboutie, solution intégrée à Ubuntu Server, produit complet avec une interface de gestion web fonctionnelle.	Jeune, mais prometteur. Soutenu par de grands acteurs de différents secteurs (informatique, aéronautique, etc).	Avancée, deuxième version stable, solution supportée par Debian.
Documentation	Correcte, complète mais pas toujours à jour.	Excellente, site bien fourni et facile d'accès avec à la fois un wiki contenant l'essentiel et une documentation officielle disponible et très détaillée.	Complète, documentations, références de tous les fichiers de configuration, exemples. Manque d'aide sur un environnement complexe.
Communauté	Fermée et peu accueillante, équipe de chercheurs peu disponible pour répondre aux problèmes.	Accueillante, la communauté ne cesse de grandir et est toujours prête à répondre aux questions.	Sympathique, de nombreuses présentations disponibles, et aide sur le canal IRC *#opennebula@freenode*.
Extensibilité	Difficile, Eucalyptus s'appuie sur le SDK Java et le code est peu accessible. La solution est compatible avec les API Amazon EC2.	Sans opinion, OpenStack est écrit en Python et est à première vue simple d'accès.	Excellente, OpenNebula est écrit en Ruby de façon élégante, et facilement extensible via une architecture à base de *drivers*.
Flexibilité	Très bonne, tant que l'on reste dans une architecture classique, sa grande modularité est un atout qui permet de facilement le déployer sur une architecture multi-sites et permet également de l'étendre à la demande.	Bonne, OpenStack se décline en deux versions : une version orientée stockage et une version orientée applications, ce qui permet d'adapter le cloud à ses besoins.	Excellente, au prix d'une conception très épurée en n'intégrant que le minimum. OpenNebula s'avère très flexible et donne beaucoup de possibilités à l'utilisateur en lui laissant des possibilités de configurations très proches de celles des hyperviseurs supportés.

Simplicité	Mitigé, lorsque l'on reste dans un cadre d'utilisation classique, la solution est simple à mettre en œuvre, en revanche, sur une architecture complexe, le déploiement devient un casse-tête.	Bonne, en effet, grâce à la documentation et les scripts d'installation fortement commentés, tout se passe bien en suivant le pas-à-pas du wiki.	Moyenne, les fonctions de base sont simples à utiliser, le passage à un environnement complexe l'est moins.

Eucalyptus est une solution mature et qui permet l'installation d'une infrastructure de type cloud assez facilement tant que l'on reste dans les clous de son fonctionnement. Cependant, le but du cloud computing est de faciliter l'évolution ce qu'Eucalyptus ne permet pas. C'est donc pour cela qu'il est actuellement délaissé pour d'autres solutions.
OpenNebula est une autre solution de cloud, flexible et mature. Sa conception est très épurée, et laisse une grande liberté à l'administrateur qui souhaiterait déployer cette solution au prix d'un effort d'intégration poussé au réseau et à d'autres solutions complémentaires pour le stockage. Dans cette optique, OpenNebula est une solution adaptée à Grid 5000.
OpenStack est une solution encore jeune, mais qui a un potentiel très important par rapport à son architecture et sa communauté ainsi que le support de ses partenaires. Il s'agit donc d'une solution à surveiller car nous pensons qu'elle peut devenir la référence des solutions de cloud computing libre.

6. Conclusion

L'objectif de ce projet était de déployer et d'évaluer les différentes solutions libre permettant de mettre en place un cloud privé sur la plateforme Grid 5000.

Cet objectif est partiellement atteint. En effet, nous avons pu tester trois solutions de cloud : Eucalyptus, OpenNebula et OpenStack.

Ce projet étant très ambitieux, nous nous sommes vite heurtés à de nombreux problèmes, que ce soit dû aux solutions de cloud ou à leur intégration sur le Grid 5000, notamment en ce qui concerne le réseau.

Tous ces problèmes nous ont montré la complexité d'utiliser une telle plate-forme, et leurs résolutions nous a souvent retardé mais nous ont appris à investiguer.

Ce projet nous a également apporté une expérience de travail en équipe sur un projet de longue durée, en le découpant en tâches et en répartissant le travail, ce qui nous a préparé au travail dans le monde professionnel et à notre stage.

Ce projet a été pour nous une chance de découvrir un environnement aussi complexe que Grid 5000, ce qui nous a permis d'acquérir de l'expérience en administration systèmes et réseaux et d'approfondir nos connaissances dans le domaine de la virtualisation et du cloud computing.

7. Bibliographie

L'ensemble des définitions ont été trouvés principalement sur le site Wikipédia : http://fr.wikipedia.org/

Les documentations officielles des solutions testées :
- Eucalyptus : http://open.eucalyptus.com/
- UEC : https://help.ubuntu.com/community/UEC
- OpenStack : http://www.openstack.org/
- OpenNebula : http://opennebula.org/

Le livre "book eucalyptus beginners guide uec edition 1" rédigé par Johnson D., Kiran Murari, Murthy Raju, Suseendran RB., Yogesh Girikumar, le 25 mai 2010 disponible au format pdf en téléchargement sur les sites de distributions d'ebooks comme celui-ci : http://ebookbrowse.com/search/eucalyptus-beginners-guide-uec-edition1

Les sites officiels suivants :
- Debian http://www.debian.org/
- Ubuntu-fr http://ubuntu-fr.org/
- Ubuntu http://www.ubuntu.com

Le site de Grid 5000 en particulier pour ses nombreux tutoriels : https://www.grid5000.fr/
Le site personnel de Pierre Riteau, thésard a l'université de Rennes qui a testé le déploiement de Nimbus sur Grid 5000 http://perso.univ-rennes1.fr/pierre.riteau/ et les slides techniques de sa participation au challenge Grid 5000 de l'année 2010 : https://www.grid5000.fr/school2010/slides/Challenge_Riteau.pdf

Presse :
- ISGTW http://www.thedigitalscientist.org/feature/feature-reading-sky-computing
- ERCIM News Octobre 2010http://ercim-news.ercim.eu/en83/special/large-scale-cloud-computing-research

Utilisation de launchpad [1] : launchpad.net/vmbuilder
OpenNebula Installation and Configuration Guide, fait par Marian Mihailescu, thésard au département de sciences de l'université de Singapour, rédigé le 11 août 2010 http://marianmi.comp.nus.edu.sg/2010/08/opennebula-installation-and-configuration-guide.php

1. Launchpad est la plate-forme de développement logiciel de Canonical.

A. Annexe : Diagramme de Gantt

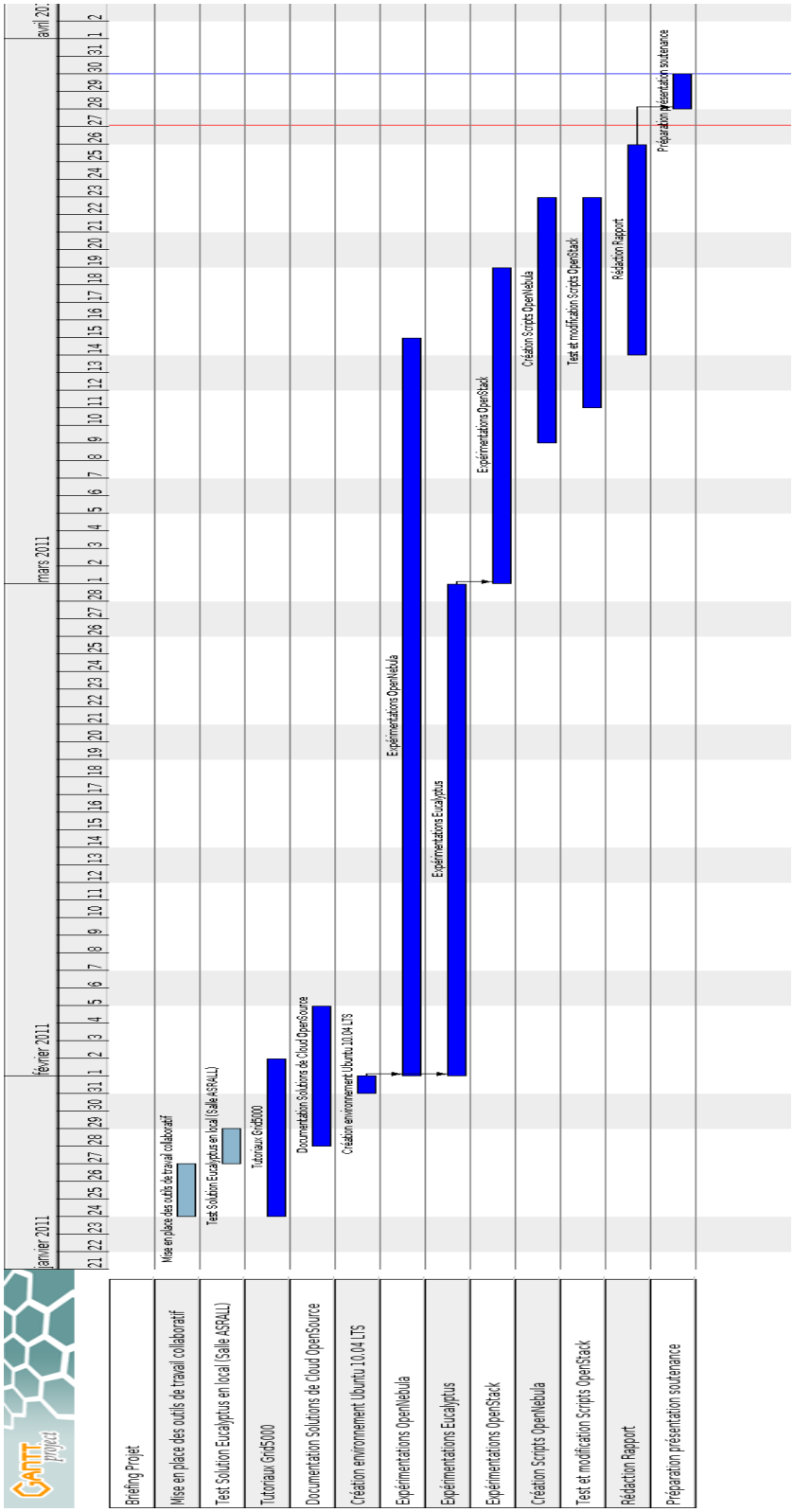

FIGURE A.1.: Diagramme de Gantt de l'organisation de notre projet

B. Annexe : Commandes de base pour la soumission de job

Cet annexe est un *mémos* que nous avons écrit en suivant les tutoriels sur le site de Grid 5000, il en est par conséquent inspiré et moins complet.

Il s'agit d'un résumé non exhaustif des commandes de base pour réserver des nœuds sur Grid 5000.

B.1. État des cluster

Voir les ressources et nœuds occupés et disponibles :

```
frontend: disco nancy
```

B.2. Informations sur les nœuds

oarnodes est aussi un outil en ligne de commande. Il permet d'afficher les propriétés des nœuds.

B.3. Information sur les jobs

Pour tout ce qui concerne les jobs en cours, il faut utiliser la commande oarstat.
Voir tous les jobs (soumissions) :

```
frontend: oarstat
```

Voir un job spécifique :

```
frontend: oarstat -f -j $OAR_JOB_ID
```

Voir tous les jobs d'un utilisateur :

```
frontend: oarstat -u login
```

B.4. Soumission de job

Pour tout ce qui concerne la soumission de job, il faut utiliser la commande oarsub.

Pour soumettre un job :

```
frontend: oarsub -I
```

Le serveur vous affiche le job correspondant à votre demande.
Pour soumettre un job sur deux nœud :

```
frontend: oarsub -I -l nodes=2
```

Par défaut, vous serez connecté sur le premier nœud, mais vous pouvez toujours utiliser cette commande pour vous connecter à un autre nœud.

```
frontend: oarsub -C $OAR_JOB_ID
```

Pour réserver des nœuds à partir d'une date pour une durée limitée :

```
frontend: oarsub -r '2010-03-24 17:30:00' -l nodes=2,walltime=0:10:00
```

Pour terminer un job :

```
fontend: oardel $OAR_JOB_ID
```

B.5. Autres commandes

Pour voir les nœuds :

```
node|frontend: sort -u $OAR_NODEFILE
```

Pour se connecter à un notre nœud :

```
    node: oarsh another-hostname
```

Pour se déconnecter et finir le job :

```
node: exit
frontend: oardel $OAR_JOB_ID
\end

Pour lire la sortie stdout (ou stderr d'un noeud) :

\begin{lstlisting}
frontend: tail -f OAR.OAR_JOB_ID.stdout
frontend: tail -f OAR.OAR_JOB_ID.stderr
```

B.6. Utiliser plusieurs clusters (oargridsub)

Pour réserver plusieurs nœuds sur des clusters différents :

```
frontend: oargridsub cluster1:rdef="/nodes=2",cluster2:rdef="/nodes=1",cluster3:rdef="nodes=1"
```

Pour visualiser un job :

```
frontend: oargridstat -w -l gridJobId
```

Pour terminer un job :

```
frontend: oargriddel gridJobId
```

C. Annexe : Déployer et enregistrer un environnement avec kadeploy3

Cet annexe est un *mémos* que nous avons écrit en suivant les tutoriels sur le site de Grid 5000, il en est par conséquent inspiré et moins complet.

Nous allons résumer ici l'enregistrement et le déploiement d'environnements sur Grid 5000.

C.1. Réserver des noeuds pour le déploiement

Réserver un noeud, en interactif pour 3 heures :

```
frontend: oarsub -I -t deploy -l nodes=1,walltime=3
```

Lister les noeuds réservés :

```
frontend: cat $OAR_FILE_NODES
```

C.2. kaenv3 : gestion des environnements kadeploy

Lister tous les environnements disponibles :

```
frontend: kaenv3 -l
```

Lister les environnements partagés par un utilisateur :

```
frontend: kaenv3 -l -u user
```

Afficher les informations d'un environnement :

```
frontend: kaenv3 -p lenny-x64-base -u deploy
```

C.3. kadeploy3

Déployer un environnement sur les nœuds réservés :

```
frontend: kadeploy3 -e lenny-x64-base -f $OAR_FILE_NODES
```

Pour copier le fichier /.ssh/authorized_keys (défaut) dans /root sur les environnements déployés, utiliser l'option -k

Obtenir une console sur un nœud :

```
frontend: kaconsole3 -m node.site.grid5000.fr
```

Se loguer sur un nœud en SSH (mot de passe par défaut : grid5000)

```
frontend: ssh root@node.site.grid5000.fr
```

C.4. kaenv3 : Créer un nouvel environnement à partir d'un environnement personnalisé

Création de l'archive depuis le frontend ou sur le noeud :

Attention, avant d'enregistrer un environnement, vous devez supprimer le fichier de règles udev fixant les adresses mac des interfaces réseau.

```
node: rm /etc/udev/rules.d/*_persistent-net.rules
```

1ère méthode :

```
frontend: ssh root@node tgz-g5k > path_to_myimage.tgz
```

2ème méthode :

```
node: tgz-g5k login@frontend:path_to_myimage.tgz
```

Création de l'environnement :

```
frontend:kaenv3 -p lenny-x64-base -u deploy > mon_environnement.env
```

Puis on modifie le fichier mon_environnement.env

```
name : mylenny
version : 4
description : my lenny
author : maxime.lemaux@gmail.com
tarball : /home/mlemaux/archive.tgz|tgz
postinstall : /grid5000/postinstalls/etch-x64-base-2.0-post.tgz|tgz|traitement.ash /rambin
kernel : /boot/vmlinuz-2.6.26-2-amd64
initrd : /boot/initrd.img-2.6.26-2-amd64
fdisktype : 83
filesystem : ext3
environment_kind : linux
demolishing_env : 0
```

On enregistre notre environnement pour pouvoir le déployer avec kadeploy3 et le partager avec les autres utilisateurs.

```
frontend: kaenv3 -a mon_environnement.env
```

D. Annexe : Scripts de synchronisation pour Grid 5000

D.1. sendsshkeys.sh

```
#!/bin/bash

# Automatically send internal keys to all sites
# Execute "ssh-keygen" on one site, add the new generated public key to known host
# Then, use this script, first arg is your user name, second is the precedent site
# Example : ./sendsshkeys.sh hcartiaux lille

user=$1
host=$2

sites="bordeaux grenoble lille lyon orsay rennes sophia toulouse"

pubkey="id_rsa.pub"
privkey="id_rsa"

echo Retrieving authorized_keys file
scp $user@$host.g5k:/home/$user/.ssh/authorized_keys .
scp $user@$host.g5k:/home/$user/.ssh/$pubkey .
scp $user@$host.g5k:/home/$user/.ssh/$privkey .


if [[ "$?" != "0" ]]
then
  echo Failed, exiting
  exit 1
fi

for site in $sites
do

    if [[ "$host" == "$site" ]]
    then
      echo Skipping site $site
      continue
    fi

    echo Sending files to $site
    scp authorized_keys $pubkey $privkey $user@$site.g5k:/home/$user/.ssh/

    if [[ "$?" != "0" ]]
    then
      echo "$site : FAILED"
    else
      echo "$site : DONE"
    fi

done
```

D.2. synctosite.sh

```
#!/bin/bash

site=$1
user=`whoami`

if [[ "$site" == "" ]]
  then
  echo "synctosite synchronize your home directory with rsync to the site of your choice"
  echo "Usage : synctosite.sh {bordeaux,grenoble,lille,lyon,orsay,rennes,sophia,toulouse}"
fi

echo "Some files might be deleted on $1. Are you sure ? [n]"

read check

if [[ "$check" == "y" ]]
  then

  cd ~
  rsync --delete -avz --exclude .ssh/known_hosts /home/$user/ $site.grid5000.fr:/home/$user/

fi
```

E. Annexe : Configuration des environnements OpenNebula

E.1. Configuration réseau : /etc/rc.local

```
#!/bin/sh
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

ifconfig br0

if [ "x$?" != "x0" ] ; then

        export NET_DEV=`netstat -rn | grep ^0.0.0.0 | awk '{print $8}'`

        /etc/init.d/networking stop

        # /sbin/dhclient -r $NET_DEV -pf /var/run/dhclient.$NET_DEV.pid
        /usr/sbin/brctl addbr br0
        /usr/sbin/brctl addif br0 $NET_DEV
        /sbin/ifconfig $NET_DEV up
        /sbin/dhclient br0

fi

exit 0
```

E.2. /etc/rc.bindhome

```
#!/bin/bash

mv /var/lib/one /tmp
mkdir /var/lib/one
mount --bind /tmp/one /var/lib/one
```

F. Annexe : Scripts de déploiement automatisé pour OpenNebula

F.1. oargridsubON.sh

```
#!/bin/bash
# set -x

ARGS=""
DURATION="02:00:00"

for line in `cat cluster-test`
do
  SITE=`echo $line | awk 'BEGIN {FS=";"} ; { print $1 }'`
  NBR=`echo $line | awk 'BEGIN {FS=";"} ; { print $2 }'`
  echo SITE $SITE
  echo NBR $NBR
# ARGS=$SITE:rdef="/nodes=$NBR:prop=virtual='amd-v' OR virtual='ivt'",$ARGS
  ARGS=$SITE:rdef="/nodes=$NBR",$ARGS
done

set -x

oargridsub -w \'$DURATION\' $ARGS -t deploy
```

F.2. kadeployON.sh

```
#!/bin/bash

# set -x

JID=$1
USER=`whoami`

oargridstat -w -l $JID | sed '/^$/d' | sort -u > tmp_gridstat

frontend=`cat tmp_gridstat | head -n 1`
site_frontend=`echo $frontend | awk 'BEGIN { FS="." } ; {print $2 }'`

echo "== éDploiement du frontend =="
echo frontend $frontend
ssh $USER@$site_frontend "kadeploy3 -e sid-opennebula-frontend -u hcartiaux -m $frontend" &> /dev/null 2> /dev/null


echo "== éDploiement des nodes =="
for machine in `tail -n +2 tmp_gridstat | uniq`
do
  site=`echo $machine | awk 'BEGIN { FS="." } ; {print $2 }'`
  echo node $machine
  ssh $USER@$site "kadeploy3 -e sid-opennebula-node -u hcartiaux -m $machine" &> /dev/null 2> /dev/null

done
```

F.3. pingON.sh

```
#!/bin/bash

for i in `cat tmp_gridstat` ; do
```

```
  ping -c 1 $i 2> /dev/null > /dev/null

  if [ "x$?" == "x0" ] ; then
    echo $i : OK
  else
    echo $i : KO
  fi

done
```

F.4. configureON.sh

```
#!/bin/bash

set -x

frontend=`cat tmp_gridstat | head -n 1`

sshpubkey="/home/hcartiaux/.ssh/id_dsa.pub"

scp $sshpubkey root@$frontend:/root/.ssh/authorized_keys
scp root@$frontend:/var/lib/one/.ssh/authorized_keys .
cat $sshpubkey >> authorized_keys

scp authorized_keys root@$frontend:/var/lib/one/.ssh/authorized_keys

for machine in `tail -n +2 tmp_gridstat | uniq`
do

  ssh oneadmin@$frontend "onehost add $machine im_kvm vmm_kvm tm_ssh"

done
```

F.5. creationvmON.rb

```
#!/usr/bin/ruby

puts "Ce script permet de éégnrer une archive conteant l'image de la vm, le fichier de
    édfinition et le script init.sh.\nAttention le nom choisi pour les vm sera le nom de l'
    archive.\n\n"


puts "Chemin de l'image à deploier (Format de l'image .img,qcow2,rar):"
image = gets.strip

puts "Nom des machines virtuelles :"
nom = gets.strip

puts "Utilisation CPU :"
cpu = gets.strip

puts "Utilisation émmoire :"
memoire = gets.strip

puts "Machine virtuel 32 bit ou 64 bit ? (32/64)"
format = gets.strip

puts "Chemin du script init.sh"
init = gets.strip

`mkdir #{nom}`

File.new("#{nom}/#{nom}.one", "w+")
template = File.open("#{nom}/#{nom}.one", "w")
template.write("NAME = #{nom}-$VMID
CPU = #{cpu}
MEMORY = #{memoire}\n
DISK = [
source = \"#{image}\",
target = \"sda\",\n")
```

```
nomImage=`basename #{image}`
nomImage =~ /\.(.*)$/
if $1 != "img"
  template.write("readonly = \"no\",\ndriver = \"#{$1}\"]\n")
else
  template.write("readonly = \"no\"]\n")
end

if format == "64"
  template.write("OS = [ ARCH = \"x86_64\" ]\n")
end

template.write("NIC = [ BRIDGE = \"br0\" ]\n
FEATURES = [ acpi=\"no\" ]\n
CONTEXT = [
 hostname = \"$NAME\",
 files = \"/var/lib/one/#{nom}/init.sh /var/lib/one/.ssh/id_rsa.pub\",
 target = \"hdc\",
 root_pubkey = \"id_rsa.pub\",
 username = \"oneadmin\",
 user_pubkey = \"id_rsa.pub\"
]")

`cp #{init} #{nom}; cp #{image} #{nom}`
`tar cfvz #{nom}.tar.gz #{nom}; rm -R #{nom}`
```

F.6. init.sh

```
#!/bin/bash

#Script ébas sur l'exemple fourni par le paquet OpenNebula

#Fonction qui éègnre une adresse Mac en fonction de L'ID de la VM que fourni OpenNebula

if [ -f /mnt/context.sh ]
then
  . /mnt/context.sh
fi

Hexa=`printf "%x\n" $ID`
TailHexa=`expr length $Hexa`

if [ $ID -lt 256 ]
then
  if [ $TailHexa -eq 1 ]
  then
    Mac=00:16:3e:00:00:0$Hexa
  else
    Mac=00:16:3e:00:00:$Hexa
  fi
else
  if [ $TailHexa -eq 3 ]
  then
    HexaFin=${Hexa:1}
   HexaDeb=${Hexa:0:1}
   Mac=00:16:3e:00:0$HexaDeb:$HexaFin
  else
   HexaFin=${Hexa:2}
   HexaDeb=${Hexa:0:2}
   Mac=00:16:3e:00:$HexaDeb:$HexaFin
  fi
fi

ifconfig eth0 up
ifconfig eth0 hw ether $Mac
ifconfig eth0 up
dhclient eth0

if [ -f /mnt/$ROOT_PUBKEY ]; then
  mkdir -p /root/.ssh
  cat /mnt/$ROOT_PUBKEY >> /root/.ssh/authorized_keys
  chmod -R 600 /root/.ssh/
fi
```

```
46
47 if [ -n "$USERNAME" ]; then
48   useradd -s /bin/bash -m $USERNAME
49   if [ -f /mnt/$USER_PUBKEY ]; then
50     mkdir -p /home/$USERNAME/.ssh/
51     cat /mnt/$USER_PUBKEY >> /home/$USERNAME/.ssh/authorized_keys
52     chown -R $USERNAME:$USERNAME /home/$USERNAME/.ssh
53     chmod -R 600 /home/$USERNAME/.ssh/authorized_keys
54   fi
55 fi
```

F.7. deployvmON.sh

```
1  #!/bin/bash
2
3  echo "Attention ce script est fait pour édployer la archive éégnr par le script creationvmON.
       rb.\nLe nom de l'archive, du dossier et du fichier descriptif de la VM doivent avoir le
       êmme nom.\nExemple : Ubntu.tar.gz contient le érpertoire Ubuntu et dans ce érpertoire il y
       a le fichier Ubuntu.one\n"
4
5  echo "Chemin de l'archive :"
6  read archive
7  frontend=`cat tmp_gridstat | head -n 1`
8  rsync -avz --progress --partial $archive oneadmin@$frontend:.
9  ssh oneadmin@$frontend "tar xvf /var/lib/one/$1"
10
11 nomVM=`basename $archive | cut -d"." -f1`
12
13 echo "Combien de machine virtuelle voulez-vous ?"
14 read nombreVM
15 for i in `seq 1 $nombreVM`;
16 do
17         ssh oneadmin@$frontend "onevm create /var/lib/one/$nomVM/$nomVM.one"
18 done
```

G. Annexe : Scripts d'installation de Nova

G.1. Script d'installation sur un seul serveur

Source : https://github.com/vishvananda/novascript/raw/master/nova.sh

```
#!/usr/bin/env bash
DIR=`pwd`
CMD=$1
if [ "$CMD" = "branch" ]; then
    SOURCE_BRANCH=${2:-lp:nova}
    DIRNAME=${3:-nova}
else
    DIRNAME=${2:-nova}
fi

NOVA_DIR=$DIR/$DIRNAME

if [ ! -n "$HOST_IP" ]; then
    # NOTE(vish): This will just get the first ip in the list, so if you
    #             have more than one eth device set up, this will fail, and
    #             you should explicitly set HOST_IP in your environment
    HOST_IP=`LC_ALL=C ifconfig  | grep -m 1 'inet addr:'| cut -d: -f2 | awk '{print $1}'`
fi

INTERFACE=${INTERFACE:-eth0}
FLOATING_RANGE=${FLOATING_RANGE:-10.6.0.0/27}
FIXED_RANGE=${FIXED_RANGE:-10.0.0.0/24}
USE_LDAP=${USE_E_MYSQL:-0}
MYSQL_PASS=${MYSQL_PASS:-nova}
TEST=${TEST:-0}
USE_LDAP=${USE_LDAP:-0}
# Use OpenDJ instead of OpenLDAP when using LDAP
USE_OPENDJ=${USE_OPENDJ:-0}
# Use IPv6
USE_IPV6=${USE_IPV6:-0}
LIBVIRT_TYPE=${LIBVIRT_TYPE:-qemu}
NET_MAN=${NET_MAN:-VlanManager}
# NOTE(vish): If you are using FlatDHCP on multiple hosts, set the interface
#             below but make sure that the interface doesn't already have an
#             ip or you risk breaking things.
# FLAT_INTERFACE=eth0

if [ "$USE_MYSQL" == 1 ]; then
    SQL_CONN=mysql://root:$MYSQL_PASS@localhost/nova
else
    SQL_CONN=sqlite:///$NOVA_DIR/nova.sqlite
fi

if [ "$USE_LDAP" == 1 ]; then
    AUTH=ldapdriver.LdapDriver
else
    AUTH=dbdriver.DbDriver
fi

if [ "$CMD" == "branch" ]; then
    sudo apt-get install -y bzr
    if [ ! -e "$DIR/.bzr" ]; then
        bzr init-repo $DIR
    fi
    rm -rf $NOVA_DIR
    bzr branch $SOURCE_BRANCH $NOVA_DIR
    cd $NOVA_DIR
    mkdir -p $NOVA_DIR/instances
    mkdir -p $NOVA_DIR/networks
    exit
fi

# You should only have to run this once
```

```
if [ "$CMD" == "install" ]; then
    sudo apt-get install -y python-software-properties
    sudo add-apt-repository ppa:nova-core/trunk
    sudo apt-get update
    sudo apt-get install -y dnsmasq-base kpartx kvm gawk iptables ebtables
    sudo apt-get install -y user-mode-linux kvm libvirt-bin
    sudo apt-get install -y screen euca2ools vlan curl rabbitmq-server
    sudo apt-get install -y lvm2 iscsitarget open-iscsi
    sudo apt-get install -y socat unzip
    echo "ISCSITARGET_ENABLE=true" | sudo tee /etc/default/iscsitarget
    sudo /etc/init.d/iscsitarget restart
    sudo modprobe kvm
    sudo /etc/init.d/libvirt-bin restart
    sudo modprobe nbd
    sudo apt-get install -y python-twisted python-mox python-ipy python-paste
    sudo apt-get install -y python-migrate python-gflags python-greenlet
    sudo apt-get install -y python-libvirt python-libxml2 python-routes
    sudo apt-get install -y python-netaddr python-pastedeploy python-eventlet
    sudo apt-get install -y python-novaclient python-glance python-cheetah
    sudo apt-get install -y python-carrot python-tempita python-sqlalchemy


    if [ "$USE_IPV6" == 1 ]; then
        sudo apt-get install -y radvd
        sudo bash -c "echo 1 > /proc/sys/net/ipv6/conf/all/forwarding"
        sudo bash -c "echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra"
    fi

    if [ "$USE_MYSQL" == 1 ]; then
        cat <<MYSQL_PRESEED | debconf-set-selections
mysql-server-5.1 mysql-server/root_password password $MYSQL_PASS
mysql-server-5.1 mysql-server/root_password_again password $MYSQL_PASS
mysql-server-5.1 mysql-server/start_on_boot boolean true
MYSQL_PRESEED
        apt-get install -y mysql-server python-mysqldb
    fi
    mkdir -p $DIR/images
    wget -c http://images.ansolabs.com/tty.tgz
    tar -C $DIR/images -zxf tty.tgz
    exit
fi

NL=`echo -ne '\015'`

function screen_it {
    screen -S nova -X screen -t $1
    screen -S nova -p $1 -X stuff "$2$NL"
}

if [ "$CMD" == "run" ] || [ "$CMD" == "run_detached" ]; then

  cat >$NOVA_DIR/bin/nova.conf << NOVA_CONF_EOF
--verbose
--nodaemon
--dhcpbridge_flagfile=$NOVA_DIR/bin/nova.conf
--network_manager=nova.network.manager.$NET_MAN
--my_ip=$HOST_IP
--public_interface=$INTERFACE
--vlan_interface=$INTERFACE
--sql_connection=$SQL_CONN
--auth_driver=nova.auth.$AUTH
--libvirt_type=$LIBVIRT_TYPE
NOVA_CONF_EOF

    if [ -n "$FLAT_INTERFACE" ]; then
        echo "--flat_interface=$FLAT_INTERFACE" >>$NOVA_DIR/bin/nova.conf
    fi

    if [ "$USE_IPV6" == 1 ]; then
        echo "--use_ipv6" >>$NOVA_DIR/bin/nova.conf
    fi

    killall dnsmasq
    if [ "$USE_IPV6" == 1 ]; then
       killall radvd
    fi
```

```
140     screen -d -m -S nova -t nova
141     sleep 1
142     if [ "$USE_MYSQL" == 1 ]; then
143         mysql -p$MYSQL_PASS -e 'DROP DATABASE nova;'
144         mysql -p$MYSQL_PASS -e 'CREATE DATABASE nova;'
145     else
146         rm $NOVA_DIR/nova.sqlite
147     fi
148     if [ "$USE_LDAP" == 1 ]; then
149         if [ "$USE_OPENDJ" == 1 ]; then
150             echo '--ldap_user_dn=cn=Directory Manager' >> \
151                 /etc/nova/nova-manage.conf
152             sudo $NOVA_DIR/nova/auth/opendj.sh
153         else
154             sudo $NOVA_DIR/nova/auth/slap.sh
155         fi
156     fi
157     rm -rf $NOVA_DIR/instances
158     mkdir -p $NOVA_DIR/instances
159     rm -rf $NOVA_DIR/networks
160     mkdir -p $NOVA_DIR/networks
161     if [ ! -d "$NOVA_DIR/images" ]; then
162         ln -s $DIR/images $NOVA_DIR/images
163     fi
164
165     if [ "$TEST" == 1 ]; then
166         cd $NOVA_DIR
167         python $NOVA_DIR/run_tests.py
168         cd $DIR
169     fi
170
171     # create the database
172     $NOVA_DIR/bin/nova-manage db sync
173     # create an admin user called 'admin'
174     $NOVA_DIR/bin/nova-manage user admin admin admin admin
175     # create a project called 'admin' with project manager of 'admin'
176     $NOVA_DIR/bin/nova-manage project create admin admin
177     # create a small network
178     $NOVA_DIR/bin/nova-manage network create $FIXED_RANGE 1 32
179
180     # create some floating ips
181     $NOVA_DIR/bin/nova-manage floating create `hostname` $FLOATING_RANGE
182
183     # convert old images
184     $NOVA_DIR/bin/nova-manage image convert $DIR/images
185
186     # nova api crashes if we start it with a regular screen command,
187     # so send the start command by forcing text into the window.
188     screen_it api "$NOVA_DIR/bin/nova-api"
189     screen_it objectstore "$NOVA_DIR/bin/nova-objectstore"
190     screen_it compute "$NOVA_DIR/bin/nova-compute"
191     screen_it network "$NOVA_DIR/bin/nova-network"
192     screen_it scheduler "$NOVA_DIR/bin/nova-scheduler"
193     screen_it volume "$NOVA_DIR/bin/nova-volume"
194     screen_it ajax_console_proxy "$NOVA_DIR/bin/nova-ajax-console-proxy"
195     sleep 2
196     # export environment variables for project 'admin' and user 'admin'
197     $NOVA_DIR/bin/nova-manage project zipfile admin admin $NOVA_DIR/nova.zip
198     unzip -o $NOVA_DIR/nova.zip -d $NOVA_DIR/
199
200     screen_it test "export PATH=$NOVA_DIR/bin:$PATH; . $NOVA_DIR/novarc"
201     if [ "$CMD" != "run_detached" ]; then
202       screen -S nova -x
203     fi
204 fi
205
206 if [ "$CMD" == "run" ] || [ "$CMD" == "terminate" ]; then
207     # shutdown instances
208     . $NOVA_DIR/novarc; euca-describe-instances | grep i- | cut -f2 | xargs euca-terminate-
            instances
209     sleep 2
210     # delete volumes
211     . $NOVA_DIR/novarc; euca-describe-volumes | grep vol- | cut -f2 | xargs -n1 euca-delete-
            volume
212     sleep 2
213 fi
```

```
214
215 if [ "$CMD" == "run" ] || [ "$CMD" == "clean" ]; then
216     screen -S nova -X quit
217     rm *.pid*
218 fi
219
220 if [ "$CMD" == "scrub" ]; then
221     $NOVA_DIR/tools/clean-vlans
222     if [ "$LIBVIRT_TYPE" == "uml" ]; then
223         virsh -c uml:///system list | grep i- | awk '{print \$1}' | xargs -n1 virsh -c uml:///
                system destroy
224     else
225         virsh list | grep i- | awk '{print \$1}' | xargs -n1 virsh destroy
226     fi
227 fi
```

G.2. Script d'installation du contrôleur

Source : https://github.com/dubsquared/OpenStack-NOVA-Installer-Script/raw/master/nova-CC-install-v1.1.sh

```
1  #!/bin/bash
2
3  # Copyright (c) 2010 OpenStack, LLC.
4  # Licensed under the Apache License, Version 2.0 (the "License");
5  # you may not use this file except in compliance with the License.
6  # You may obtain a copy of the License at
7
8  #    http://www.apache.org/licenses/LICENSE-2.0
9
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the License is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
13 # implied.
14
15 # See the License for the specific language governing permissions and
16 # limitations under the License.
17
18 # Written by Wayne A. Walls (dubsquared) with the amazing help of Jordan Rinke (JordanRinke),
     Vish Ishaya (vishy),
19 # and a lot of input from the fine folks in #openstack on irc.freenode.net!
20
21 # Please contact script maintainers for questions, comments, or concerns:
22 # Wayne -> wayne@openstack.org
23 # Jordan -> jordan@openstack.org
24
25 # This script is intended to be ran on a fresh install on Ubuntu 10.04 64-bit.  Once ran with
26 # the appropiate varibles, will produce a fully functioning Nova Cloud Contoller.  I am
     working on
27 # getting this working on all flavors of Ubuntu, and eventually RPM based distros.  Please
     feel free
28 # to reach out to script maintainers for anything that can be done better.  I'm pretty new to
     this scripting business
29 # so I'm sure there is room for improvement!
30
31 #Usage:  bash nova-CC-installer.sh
32
33 #This is a Linux check
34 if [ `uname -a | grep -i linux | wc -l` -lt 1 ]; then
35   echo "Not Linux, not compatible."
36   exit 1
37 fi
38
39 #Compatible OS Check
40     DEB_OS=`cat /etc/issue | grep -i 'ubuntu'`
41     RH_OS=`cat /etc/issue | grep -i 'centos'`
42     if [[ ${#DEB_OS} -gt 0 ]] ; then
43         echo "Valid OS, continuing..."
44         CUR_OS="Ubuntu"
45     elif [[ ${#RH_OS} -gt 0 ]] ; then
46         echo "Unsupported OS, sorry!"
47         CUR_OS="CentOS"
48         exit 1
49     else
```

```
50        echo "Unsupported OS, sorry!"
51          CUR_OS="Unknown"
52          exit 1
53      fi
54      echo $CUR_OS detected!
55
56  #Set up log file for debugging
57  LOGFILE=/var/log/nova/nova-install.log
58  mkdir /var/log/nova
59  touch /var/log/nova/nova-install.log
60
61  echo "Nova Cloud Controller install script v1.0"
62  echo
63  echo "Setting up the Nova cloud controller is a multi-step process.  After you seed
        information, the script will take over and finish off the install for you. Full log of
        commands will be availible at /var/log/nova/nova-install.log"
64  echo
65
66  read -p "Press any key to continue..." -n1 -s
67  echo
68
69  #Setting up sanity check function
70  valid_ipv4(){
71   newmember=$(echo $1 | egrep '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$');
72   members=$(echo $newmember | tr "." "\n");
73   memcount=$(echo $newmember | tr "." "\n" | wc -l);
74   if [ $memcount -ne 4 ]; then
75    echo "fail";
76    exit;
77   fi
78   for i in $members; do
79    if [ $i -lt 0 ] || [ $i -gt 255 ]; then
80     echo "fail";
81     exit;
82    fi;
83   done;
84   echo "success";
85  }
86
87  #Enforcing root user to execute the script
88  USER=root
89  USER2=`whoami`
90
91          if [ $USER != $USER2 ]; then
92              echo "Please run this as the user, 'root'!"; exit 1
93         else
94              echo
95  fi
96
97  echo
98
99  echo
100 echo "Step 1: Setting up the database."
101 #mySQL only for now, will update to include others in next release
102 echo
103
104 echo "mySQL User Config"
105 echo "#################"
106 echo
107
108 #Setting up mySQL root password, and verify
109 while true; do
110         read -s -p "Desired mySQL Password:  " MYSQL_PASS
111         echo "";
112         read -s -p "Verify password: " MYSQL_PASS2
113         echo "";
114
115         if [ $MYSQL_PASS != $MYSQL_PASS2 ]
116         then
117                 echo "Passwords do not match... try again.";
118                 continue;
119         fi
120         break;
121 done
122
123 cat <<MYSQL_PRESEED | debconf-set-selections
```

```
mysql-server-5.1 mysql-server/root_password password $MYSQL_PASS
mysql-server-5.1 mysql-server/root_password_again password $MYSQL_PASS
mysql-server-5.1 mysql-server/start_on_boot boolean true
MYSQL_PRESEED
echo

echo "Step 2: Setting up the controller configuration."
echo "This includes setting the S3 IP, RabbitMQ IP, Cloud Controller IP, and mySQL host IP"

echo "Nova Cloud Controller Configuration"
echo "###################################"
sleep 1

#Configuring S3 Host IP
set -o nounset
echo

debug=":"
debug="echo"

echo

#Grabs the first real IP of ifconfig, and set it as the default entry
default=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/g
    ' | tr -d " "`

while true; do
read -p "S3 Host IP (Default is $default -- ENTER to accept):" -e t1
if [ -n "$t1" ]
then
  S3_HOST_IP="$t1"
else
  S3_HOST_IP="$default"
fi
if [ $(valid_ipv4 $S3_HOST_IP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
   continue
fi
break;
done;

echo
echo " S3 Host IP set as \"$S3_HOST_IP\""

#Configuring RabbitMQ IP
set -o nounset
echo

debug=":"
debug="echo"

echo

#default=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/
    g' | tr -d " "`
while true; do
read -p "RabbitMQ Host IP (Default is $default -- ENTER to accept): " -e t1
if [ -n "$t1" ]
then
  RABBIT_IP="$t1"
else
  RABBIT_IP="$default"
fi

if [ $(valid_ipv4 $RABBIT_IP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
    continue
fi
break;
done;

echo
echo " RabbitMQ Host IP set as \"$RABBIT_IP\""
echo
echo "There is an issue bypassing the rabbit package splash screen, so installing here and
    allowing you to proceed.   There is currenly no \
```

```
way to background/preseed this, so it will output to terminal..."

echo
sleep 5
apt-get -y install rabbitmq-server
echo

#Configuring Cloud Controller Host IP
set -o nounset
echo

debug=":"
debug="echo"

echo

#default=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/
    g' | tr -d " "`
while true; do
read -p "Cloud Controller Host IP (Default is $default -- ENTER to accept): " -e t1
if [ -n "$t1" ]
then
  CC_HOST_IP="$t1"
else
  CC_HOST_IP="$default"
fi

if [ $(valid_ipv4 $CC_HOST_IP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
    continue
fi
break;
done;

echo
echo " Cloud Controller Host IP set as \"$CC_HOST_IP\""

#Configuring mySQL Host IP
set -o nounset
echo

debug=":"
debug="echo"


echo

#default=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/
    g' | tr -d " "`
while true; do
read -p "mySQL Host IP (Default is $default -- ENTER to accept): " -e t1
if [ -n "$t1" ]
then
  MYSQL_HOST_IP="$t1"
else
  MYSQL_HOST_IP="$default"
fi

if [ $(valid_ipv4 $MYSQL_HOST_IP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
    continue
fi
break;
done;

echo
echo "mySQL Host IP set as \"$MYSQL_HOST_IP\""

echo

echo "Step 3:  Building the network for your controller."

echo "Here you will set the network range that ALL your projects will reside in.  This is
    typically \
a large block such as a /12.  You will also choose how many IPs in this block are availible
    for use."
```

```
269 |
270 | echo
271 |
272 | echo "Nova␣Network␣Setup"
273 | echo "####################"
274 | echo
275 |
276 | #Set network range and sanity check
277 |
278 | while true; do
279 | read -p "Controller␣network␣range␣for␣ALL␣projects␣(normally␣x.x.x.x/12):" FIXED_RANGE
280 | IP=$(echo $FIXED_RANGE | awk -F/ '{print $1}');
281 | CIDR=$(echo $FIXED_RANGE | awk -F/ '{print $2}' | perl -pe 'if(!($_>=1 && $_<=32)){undef $_
      ;}');
282 | if [ $(valid_ipv4 $IP) == "fail" ] || [ -z "$CIDR" ]
283 | then
284 |  echo "You␣failed␣at␣entering␣a␣correct␣range,␣try␣again"
285 |  continue
286 | fi;
287 |
288 | break;
289 | done;
290 |
291 | while true; do
292 | read -p "Total␣amount␣of␣usable␣IPs␣for␣ALL␣projects:" NETWORK_SIZE
293 |   if [ ! "$(␣echo␣$NETWORK_SIZE␣|␣egrep␣'^[0-9]{1,9}$'␣)" ];
294 |     then echo "Not␣a␣valid␣entry,␣please␣try␣again"
295 |   continue
296 | fi
297 | break;
298 | done;
299 |
300 | echo
301 |
302 | echo "Step␣4:␣␣Creating␣a␣project␣for␣Nova"
303 |
304 | echo "You␣will␣choose␣an␣admin␣for␣the␣project,␣and␣also␣name␣the␣project␣here.␣\
305 | Also,␣you␣will␣build␣out␣the␣network␣configuration␣for␣the␣project."
306 | sleep 1
307 | echo
308 |
309 | read -p "Nova␣project␣user␣name:" NOVA_PROJECT_USER
310 | read -p "Nova␣project␣name:" NOVA_PROJECT
311 |
312 | while true; do
313 | read -p "Desired␣network␣+␣CIDR␣for␣project␣(normally␣x.x.x.x/24):" PROJECT_CIDR
314 |   if [ ! "$(␣echo␣$PROJECT_CIDR␣|␣egrep␣
      '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\/[0-9]{1,2}$'␣)" ];
315 |     then echo "You␣failed␣at␣entering␣a␣correct␣range,␣try␣again"
316 |    continue
317 | fi
318 | break;
319 | done;
320 |
321 | while true; do
322 | read -p "How␣many␣networks␣for␣project:" NOVA_NETWORK_NUMBER
323 | if [ ! "$(␣echo␣$NOVA_NETWORK_NUMBER␣|␣egrep␣'^[0-9]{1,3}$'␣)" ];
324 |    then echo "You␣have␣not␣entered␣a␣valid␣network␣number,␣try␣again"
325 |   continue
326 | fi
327 | break;
328 | done;
329 |
330 | while true; do
331 | read -p "How␣many␣availible␣IPs␣per␣project␣network:" IPS_PER_NETWORK
332 | if [ ! "$(␣echo␣$IPS_PER_NETWORK␣|␣egrep␣'^[0-9]{1,9}$'␣)" ];
333 |    then echo "You␣have␣not␣entered␣amount␣of␣IPs"
334 |   continue
335 | fi
336 | break;
337 | done;
338 |
339 | echo
340 |
341 | echo "Preparing␣setup␣of␣br100"
342 | echo "############################"
```

```
343 echo
344
345 #Backup original network file
346 cp /etc/network/interfaces /root/interfaces.ORIG
347
348 LOCALIP=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/g
      ' | tr -d "␣"`
349
350 while true; do
351 read -p "Please␣enter␣your␣local␣server␣IP␣(Default␣is␣$LOCALIP␣--␣ENTER␣to␣accept):" -e t1
352 if [ -n "$t1" ]
353 then
354   LOCALIP="$t1"
355 else
356   LOCALIP="$LOCALIP"
357 fi
358 if [ $(valid_ipv4 $LOCALIP) == "fail" ]; then
359     echo "You␣have␣entered␣an␣invalid␣IP␣address,␣please␣try␣again."
360    continue
361 fi
362 break;
363 done;
364
365 echo
366 BROADCAST=`ifconfig -a | egrep '.*inet ' | head -n 1 | perl -pe 's/.*Bcast:(.+).*Mask.*/$1/g'
      | tr -d "␣"`
367
368 while true; do
369 read -p "Please␣enter␣your␣broadcast␣IP␣(Default␣is␣$BROADCAST␣--␣ENTER␣to␣accept):" -e t1
370 if [ -n "$t1" ]
371 then
372   BROADCAST="$t1"
373 else
374   BROADCAST="$BROADCAST"
375 fi
376 if [ $(valid_ipv4 $BROADCAST) == "fail" ]; then
377     echo "You␣have␣entered␣an␣invalid␣IP␣address,␣please␣try␣again."
378    continue
379 fi
380 break;
381 done;
382
383 echo
384
385 NETMASK=`ifconfig -a | egrep '*inet '| head -n 1 | perl -pe 's/.*Mask:/$1/g'| tr -d "␣"`
386
387 while true; do
388 read -p "Please␣enter␣your␣netmask␣(Default␣is␣$NETMASK␣--␣ENTER␣to␣accept):" -e t1
389 if [ -n "$t1" ]
390 then
391   NETMASK="$t1"
392 else
393   NETMASK="$NETMASK"
394 fi
395 if [ $(valid_ipv4 $NETMASK) == "fail" ]; then
396     echo "You␣have␣entered␣an␣invalid␣IP␣address,␣please␣try␣again."
397    continue
398 fi
399 break;
400 done;
401
402 echo
403
404 GATEWAY=`ip route | awk '/default/{print $3}'`
405
406 while true; do
407 read -p "Please␣enter␣your␣gateway␣(Default␣is␣$GATEWAY␣--␣ENTER␣to␣accept):" -e t1
408 if [ -n "$t1" ]
409 then
410   GATEWAY="$t1"
411 else
412   GATEWAY="$GATEWAY"
413 fi
414 if [ $(valid_ipv4 $GATEWAY) == "fail" ]; then
415     echo "You␣have␣entered␣an␣invalid␣IP␣address,␣please␣try␣again."
416    continue
```

```
fi
break;
done;

echo

NAMESERVER=`cat /etc/resolv.conf| awk '/nameserver/{print $2}'`

while true; do
read -p "Please enter your default nameserver (Default is $NAMESERVER --- ENTER to accept):" -e
     t1
if [ -n "$t1" ]
then
  NAMESERVER="$t1"
else
  NAMESERVER="$NAMESERVER"
fi
if [ $(valid_ipv4 $NAMESERVER) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
   continue
fi
break;
done;

echo "At this point, you've entered all the information needed to finish deployment of your
    controller!"
echo
echo "Feel free to get some coffee, you have earned it!"
sleep 5

echo
echo "Entering auto-pilot mode..."
echo

#Package installation function, and sanity check

echo "Installing packages"
echo "###################"

install_package() {
  for packages in $@
  do
    printf "%-40s" "Installing package '$packages' ... "
    if dpkg -l | egrep "^ii.* $packages " &>> $LOGFILE
    then
     echo "Already installed";
     continue;
    fi
    apt-get install -y $packages &>> $LOGFILE
    if dpkg -l | egrep "^ii.* $packages " &>> $LOGFILE
    then
     echo "ok";
    else
     echo "Failed";
     #exit -1
    fi
  done
#  return 0;
}

REQUIRED_PACKAGES="python-software-properties"
install_package ${REQUIRED_PACKAGES}
#add-apt-repository ppa:nova-core/ppa &>> $LOGFILE
add-apt-repository ppa:nova-core/release &>> $LOGFILE
apt-get update &>> $LOGFILE
REQUIRED_PACKAGES="python-mysqldb mysql-server nova-api nova-network nova-objectstore nova-
    scheduler nova-compute unzip vim euca2ools"
#REQUIRED_PACKAGES="mysql-server bzr nova-common nova-doc python-mysqldb python-greenlet
    python-nova nova-api nova-network nova-objectstore nova-scheduler nova-compute unzip vim
    euca2ools dnsmasq open-iscsi kpartx kvm gawk iptables ebtables user-mode-linux kvm libvirt
    -bin screen iscsitarget euca2ools vlan curl python-twisted python-sqlalchemy python-mox
    python-greenlet python-carrot python-daemon python-eventlet python-gflags python-libvirt
    python-libxml2 python-routes"
install_package ${REQUIRED_PACKAGES}

echo "Finalizing mySQL setup"
```

```
485 echo "#########################"
486 echo
487
488 sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
489 service mysql restart &>> $LOGFILE
490 echo
491
492 mysql -uroot -p$MYSQL_PASS -e "CREATE DATABASE nova;"
493 mysql -uroot -p$MYSQL_PASS -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;"
494 mysql -uroot -p$MYSQL_PASS -e "SET PASSWORD FOR 'root'@'%' = PASSWORD('$MYSQL_PASS');"
495 echo "...done..."
496 echo
497
498 echo "Setting up Nova configuration files"
499 echo "#######################################"
500
501
502 # Passing info into config files
503 cat >> /etc/nova/nova.conf << NOVA_CONF_EOF
504 --s3_host=$S3_HOST_IP
505 --rabbit_host=$RABBIT_IP
506 --cc_host=$CC_HOST_IP
507 --ec2_url=http://$S3_HOST_IP:8773/services/Cloud
508 --fixed_range=$FIXED_RANGE
509 --network_size=$NETWORK_SIZE
510 --FAKE_subdomain=ec2
511 --routing_source_ip=$CC_HOST_IP
512 --verbose
513 --sql_connection=mysql://root:$MYSQL_PASS@$MYSQL_HOST_IP/nova
514 --network_manager=nova.network.manager.FlatManager
515 NOVA_CONF_EOF
516
517 echo "Initializing database"
518 nova-manage db sync &>> $LOGFILE
519 sleep 1
520 echo "...done..."
521 echo
522
523 /usr/bin/python /usr/bin/nova-manage user admin $NOVA_PROJECT_USER &>> $LOGFILE
524 /usr/bin/python /usr/bin/nova-manage project create $NOVA_PROJECT $NOVA_PROJECT_USER &>>
        $LOGFILE
525 /usr/bin/python /usr/bin/nova-manage network create $PROJECT_CIDR $NOVA_NETWORK_NUMBER
        $IPS_PER_NETWORK &>> $LOGFILE
526 echo "...done..."
527
528 echo
529 echo "Generating Nova credentials"
530 echo "###########################"
531
532 mkdir -p /root/creds
533 /usr/bin/python /usr/bin/nova-manage project zipfile $NOVA_PROJECT $NOVA_PROJECT_USER /root/
        creds/novacreds.zip &>> $LOGFILE
534 sleep 3
535 unzip -d /root/creds /root/creds/novacreds.zip &>> $LOGFILE
536 . /root/creds/novarc
537 cat /root/creds/novarc >> ~/.bashrc
538 sed -i.bak "s/127.0.0.1/$CC_HOST_IP/g" /root/creds/novarc
539 echo "...done..."
540 echo
541
542 echo "Creating br100 bridge device"
543 echo "############################"
544 echo
545
546 cat > /etc/network/interfaces << NOVA_BR100_CONFIG_EOF
547 # The loopback network interface
548 auto lo
549 iface lo inet loopback
550
551 auto br100
552 iface br100 inet static
553         bridge_ports eth0
554         bridge_stp off
555         bridge_maxwait 0
556         bridge_fd 0
557         address $LOCALIP
```

```
558         netmask $NETMASK
559         broadcast $BROADCAST
560         gateway $GATEWAY
561         dns-nameservers $NAMESERVER
562 NOVA_BR100_CONFIG_EOF
563 echo "...done..."
564 echo
565
566 echo "Restarting networking"
567 /etc/init.d/networking restart
568 echo "...done..."
569 echo
570
571 echo "###################NOTE#####NOTE#####NOTE#####NOTE#####NOTE#####NOTE###################"
572 echo "#Be sure to source your credential file into your environment after config changes#"
573 echo "##############################e.g. source /root/creds/novarc##############################"
574 sleep 1
575 echo
576
577 #Sourcing creds file and restarting all Nova services
578 . /root/creds/novarc
579 sleep 1
580 #21Jan2011 - Init scripts updated to use upstart
581 restart libvirt-bin &>> $LOGFILE; restart nova-network &>> $LOGFILE; restart nova-compute &>>
        $LOGFILE; restart nova-api &>> $LOGFILE; restart nova-objectstore &>> $LOGFILE; restart
        nova-scheduler &>> $LOGFILE
582 #restart libvirt-bin &>> $LOGFILE; service nova-api restart &>> $LOGFILE; service nova-
        scheduler restart &>> $LOGFILE; service nova-network restart &>> $LOGFILE; service nova-
        objectstore restart &>> $LOGFILE; service nova-compute restart &>> $LOGFILE
583
584 echo "Ensure all five Nova services are running"
585 echo "############################################"
586 ps aux | grep -i nova
587 sleep 5
588
589 echo "Setup default ICMP and SSH access to your future VMs"
590 echo "###################################################"
591
592 echo "Allowing ping and SSH to your running instances"
593 euca-authorize -P icmp -t -1:-1 default &>> $LOGFILE
594 euca-authorize -P tcp -p 22 default &>> $LOGFILE
595 echo "...done..."
596
597 echo "#################################################################################"
598 echo "#You /MUST/ re-source your 'novarc' to use the API commands since the#"
599 echo "##script cannot pass the source information out of it's own process###"
600 echo "#################################################################################"
601 echo
602
603 #Only one dnsmasq process starts, supposed to two running at different priorities.  This fixes
        that...possible bug.
604 killall dnsmasq
605 service nova-network restart &>> $LOGFILE
606
607 #Needed for KVM to initialize, VMs run in qemu mode otherwise and is very slow
608 chgrp kvm /dev/kvm
609 chmod g+rwx /dev/kvm
610 echo
611 echo 'The next thing you are going to want to do it get a VM to test with.  You can find a
        test VM how-to, and read more about custom image creation at "http://nova.openstack.org/
        adminguide/multi.node.install.html" and "http://wiki.openstack.org/GettingImages'
612 echo
613 echo 'Enjoy your new private cloud!'
614
615
616 #If you run into any problems, please feel free to contact the script maintainers.
617 #You can also get assistance by stopping by irc.freenode.net - #openstack, sending a message
        to the OpenStack Users maillist - #openstack@lists.launchpad.net, or posting at "https://
        answers.launchpad.net/openstack"
```

G.3. Script d'installation du nœud

Source : `https://github.com/dubsquared/OpenStack-NOVA-Installer-Script/`.

```
#!/bin/sh

# Copyright (c) 2011 OpenStack, LLC.
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at

#    http://www.apache.org/licenses/LICENSE-2.0

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.

# See the License for the specific language governing permissions and
# limitations under the License.

# Written by Wayne A. Walls (dubsquared) with the amazing help of Jordan Rinke (JordanRinke), Vish Ishaya (vishy),
# and a lot of input from the fine folks in #openstack on irc.freenode.net!

# Please contact script maintainers for questions, comments, or concerns:
# Wayne  ->  wayne@openstack.org
# Jordan ->  jordan@openstack.org

# This script is intended to be ran on a fresh install on Ubuntu 10.04 64-bit.  Once ran with
# the appropiate varibles, will produce a fully functioning Nova Cloud Contoller.  I am working on
# getting this working on all flavors of Ubuntu, and eventually RPM based distros.  Please feel free
# to reach out to script maintainers for anything that can be done better.  I'm pretty new to this scripting business
# so I'm sure there is room for improvement!

#Usage:  bash nova-NODE-installer.sh

#This is a Linux check
if [ `uname -a | grep -i linux | wc -l` -lt 1 ]; then
  echo "Not Linux, not compatible."
  exit 1
fi

#Compatible OS Check
    DEB_OS=`cat /etc/issue | grep -i 'ubuntu'`
    RH_OS=`cat /etc/issue | grep -i 'centos'`
    if [[ ${#DEB_OS} -gt 0 ]] ; then
        echo "Valid OS, continuing..."
        CUR_OS="Ubuntu"
    elif [[ ${#RH_OS} -gt 0 ]] ; then
        echo "Unsupported OS, sorry!"
        CUR_OS="CentOS"
        exit 1
    else
      echo "Unsupported OS, sorry!"
        CUR_OS="Unknown"
        exit 1
    fi
    echo $CUR_OS detected!

#Set up log file for debugging
LOGFILE=/var/log/nova/nova-node-install.log
mkdir /var/log/nova
touch /var/log/nova/nova-node-install.log

#Setting up sanity check function
valid_ipv4(){
 newmember=$(echo $1 | egrep '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$');
 members=$(echo $newmember | tr "." "\n");
 memcount=$(echo $newmember | tr "." "\n" | wc -l);
 if [ $memcount -ne 4 ]; then
  echo "fail";
  exit;
 fi
 for i in $members; do
  if [ $i -lt 0 ] || [ $i -gt 255 ]; then
   echo "fail";
```

```
    exit;
   fi;
 done;
 echo "success";
}

echo "Installing required packages"
echo "##############################"

apt-get install -y python-software-properties
add-apt-repository ppa:nova-core/release
#add-apt-repository ppa:nova-core/trunk
apt-get update
apt-get install -y nova-compute python-mysqldb

#Configuring S3 Host IP
set -o nounset
echo

debug=":"
debug="echo"

echo

#Grabs the first real IP of ifconfig, and set it as the default entry
#read -p "What is the IP address of your NOVA CONTROLLER? " default
#echo

#Configuring Cloud Controller Host IP
read -p "What is the IP address of your NOVA CONTROLLER? " default

set -o nounset
echo

debug=":"
debug="echo"

echo

while true; do
read -p "NOVA Controller IP (Default is $default -- ENTER to accept): " -e t1
if [ -n "$t1" ]
then
  CC_HOST_IP="$t1"
else
  CC_HOST_IP="$default"
fi

if [ $(valid_ipv4 $CC_HOST_IP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
    continue
fi
break;
done;

echo
echo " Cloud Controller Host IP set as \"$CC_HOST_IP\""

#default=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/
    g' | tr -d " "`

while true; do
read -p "S3 Host IP (Default is $default -- ENTER to accept):" -e t1
if [ -n "$t1" ]
then
  S3_HOST_IP="$t1"
else
  S3_HOST_IP="$default"
fi
if [ $(valid_ipv4 $S3_HOST_IP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
   continue
fi
break;
done;

```

```
148 echo
149 echo "␣S3␣Host␣IP␣set␣as␣\"$S3_HOST_IP\""
150
151 #Configuring RabbitMQ IP
152 set -o nounset
153 echo
154
155 debug=":"
156 debug="echo"
157
158 echo
159
160 while true; do
161 read -p "RabbitMQ␣Host␣IP␣(Default␣is␣$default␣---␣ENTER␣to␣accept):␣" -e t1
162 if [ -n "$t1" ]
163 then
164   RABBIT_IP="$t1"
165 else
166   RABBIT_IP="$default"
167 fi
168
169 if [ $(valid_ipv4 $RABBIT_IP) == "fail" ]; then
170     echo "You␣have␣entered␣an␣invalid␣IP␣address,␣please␣try␣again."
171     continue
172 fi
173 break;
174 done;
175
176 echo
177 echo "␣RabbitMQ␣Host␣IP␣set␣as␣\"$RABBIT_IP\""
178 echo
179
180 #Configuring mySQL Host IP
181 set -o nounset
182 echo
183
184 debug=":"
185 debug="echo"
186
187
188 echo
189
190 while true; do
191 read -p "mySQL␣Host␣IP␣(Default␣is␣$default␣---␣ENTER␣to␣accept):␣" -e t1
192 if [ -n "$t1" ]
193 then
194   MYSQL_HOST_IP="$t1"
195 else
196   MYSQL_HOST_IP="$default"
197 fi
198
199 if [ $(valid_ipv4 $MYSQL_HOST_IP) == "fail" ]; then
200     echo "You␣have␣entered␣an␣invalid␣IP␣address,␣please␣try␣again."
201     continue
202 fi
203 break;
204 done;
205
206 echo
207 echo "mySQL␣Host␣IP␣set␣as␣\"$MYSQL_HOST_IP\""
208
209 echo
210
211 echo "mySQL␣User␣Config"
212 echo "####################"
213 echo
214
215 #Setting up mySQL root password, and verify
216 while true; do
217 read -s -p "Enter␣mySQL␣password␣on␣controller␣node:␣" MYSQL_PASS
218         echo "";
219         read -s -p "Verify␣password:␣" MYSQL_PASS2
220         echo "";
221
222         if [ $MYSQL_PASS != $MYSQL_PASS2 ]
223         then
```

```
echo "Passwords do not match... try again.";
                continue;
        fi
break;
done

echo "Setting up Nova configuration files"
echo "###################################"
echo

#Info to be passed into /etc/nova/nova.conf

cat >> /etc/nova/nova.conf << NOVA_CONF_EOF
--s3_host=$S3_HOST_IP
--rabbit_host=$RABBIT_IP
--cc_host=$CC_HOST_IP
--ec2_url=http://$S3_HOST_IP:8773/services/Cloud
--sql_connection=mysql://root:$MYSQL_PASS@$MYSQL_HOST_IP/nova
--network_manager=nova.network.manager.FlatManager
NOVA_CONF_EOF
echo "...done..."
echo

echo "Setting up br100"
echo "################"
echo
LOCALIP=`/sbin/ifconfig -a | egrep '.*inet ' | head -n 1|perl -pe 's/.*addr:(.+).*Bcast.*/$1/g
    ' | tr -d " "`


while true; do
read -p "Please enter your local server IP (Default is $LOCALIP -- ENTER to accept):" -e t1
if [ -n "$t1" ]
then
  LOCALIP="$t1"
else
  LOCALIP="$LOCALIP"
fi
if [ $(valid_ipv4 $LOCALIP) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
   continue
fi
break;
done;

echo
BROADCAST=`ifconfig -a | egrep '.*inet ' | head -n 1 | perl -pe 's/.*Bcast:(.+).*Mask.*/$1/g'
    | tr -d " "`

while true; do
read -p "Please enter your broadcast IP (Default is $BROADCAST -- ENTER to accept):" -e t1
if [ -n "$t1" ]
then
  BROADCAST="$t1"
else
  BROADCAST="$BROADCAST"
fi
if [ $(valid_ipv4 $BROADCAST) == "fail" ]; then
    echo "You have entered an invalid IP address, please try again."
   continue
fi
break;
done;

echo

NETMASK=`ifconfig -a | egrep '*inet '| head -n 1 | perl -pe 's/.*Mask:/$1/g'| tr -d " "`

while true; do
read -p "Please enter your netmask (Default is $NETMASK -- ENTER to accept):" -e t1
if [ -n "$t1" ]
then
  NETMASK="$t1"
else
  NETMASK="$NETMASK"
fi
```

```
298 if [ $(valid_ipv4 $NETMASK) == "fail" ]; then
299     echo "You have entered an invalid IP address, please try again."
300     continue
301 fi
302 break;
303 done;
304
305 echo
306
307 GATEWAY=`ip route | awk '/default/{print $3}'`
308
309 while true; do
310 read -p "Please enter your gateway (Default is $GATEWAY -- ENTER to accept):" -e t1
311 if [ -n "$t1" ]
312 then
313   GATEWAY="$t1"
314 else
315  GATEWAY="$GATEWAY"
316 fi
317 if [ $(valid_ipv4 $GATEWAY) == "fail" ]; then
318     echo "You have entered an invalid IP address, please try again."
319     continue
320 fi
321 break;
322 done;
323
324 echo
325
326 NAMESERVER=`cat /etc/resolv.conf| awk '/nameserver/{print $2}'`
327
328 while true; do
329 read -p "Please enter your default nameserver (Default is $NAMESERVER -- ENTER to accept):" -e
      t1
330 if [ -n "$t1" ]
331 then
332   NAMESERVER="$t1"
333 else
334   NAMESERVER="$NAMESERVER"
335 fi
336 if [ $(valid_ipv4 $NAMESERVER) == "fail" ]; then
337     echo "You have entered an invalid IP address, please try again."
338     continue
339 fi
340 break;
341 done;
342
343 echo
344 while true; do
345 read -p "Please enter the IP where nova-api lives: " NOVA_API_IP
346
347 if [ $(valid_ipv4 $NOVA_API_IP) == "fail" ]; then
348     echo "You have entered an invalid IP address, please try again."
349     continue
350 fi
351 break;
352 done;
353
354 cat > /etc/network/interfaces << NOVA_BR100_CONFIG_EOF
355 # The loopback network interface
356 auto lo
357 iface lo inet loopback
358
359 auto br100
360 iface br100 inet static
361         bridge_ports eth0
362         bridge_stp off
363         bridge_maxwait 0
364         bridge_fd 0
365         address $LOCALIP
366         netmask $NETMASK
367         broadcast $BROADCAST
368         gateway $GATEWAY
369         dns-nameservers $NAMESERVER
370 NOVA_BR100_CONFIG_EOF
371
372 echo
```

```
373 echo "Bouncing services"
374 echo "###################"
375 /etc/init.d/networking restart; restart libvirt-bin; service nova-compute restart
376 echo "...done..."
377
378 #Needed for KVM to initialize, VMs run in qemu mode otherwise and is very slow
379 chgrp kvm /dev/kvm
380 chmod g+rwx /dev/kvm
381
382 #Any server that does /NOT/ have nova-api running on it will need this rule for UEC images to
      get metadata info
383 iptables -t nat -A PREROUTING -d 169.254.169.254/32 -p tcp -m tcp --dport 80 -j DNAT --to-
      destination $NOVA_API_IP:8773
```