Licence Professionnelle :

Administration de systèmes, réseaux et applications à base de logiciels libres

IUT Nancy Charlemagne

Cloud Computing

Tuteur : M. Lucas Nussbaum

Groupe : Vincent Kherbache, Mohamed Moussalih, Yannick Kuhn, Allan Lefort

Sommaire

1. Introduction

Indéniablement, la technologie de l'internet se développe de manière exponentielle depuis sa création. Actuellement, une nouvelle "tendance" à fait son apparition dans le monde des IT (Technologies de l'information et de la communication), il s'agit du cloud computing. Cette technologie, s'appuyant sur le WEB 2.0, offre des occasions aux sociétés de réduire les coûts d'exploitation des logiciels par leurs utilisations directement en ligne. Divers fournisseurs comme Google, Amazon, IBM offrent une vaste gamme de services de Cloud Computing. Cette technologie vient juste d'éclore, elle est au début de son exploitation mais déjà plusieurs acteurs majeurs cités précédemment adoptent leurs propres stratégies de pionnier qui déterminera l'utilisation du cloud computing des entreprises souhaitant investir.

De plus, on remarque aussi que des plus petits acteurs se battent pour une part de marché.

Beaucoup d'entreprises restent cependant sceptiques sur le cloud computing. La principale raison est l'intégrité et la sécurité des données car les DSI (direction des systèmes d'informations) restent frileuses de penser que leur données critiques sont dans un endroit incontrôlé et souvent inconnu.

Enfin, notons la position de l'open source dans cette technologie car aujourd'hui, on ne parle plus de serveur sans virtualisation et donc de XEN et KVM qui ont été choisi par la plupart des exploitants. Sans compter que la plupart des logiciels développés pour le cloud computing sont open source.

2. Le Cloud Computing

2.1. Qu'est-ce que le Cloud Computing?

Le cloud computing se traduit littéralement par "informatique dans les nuages", faisant référence aux technologies d'internet qui est souvent représenté schématiquement par un nuage. C'est un concept abstrait qui regroupe plusieurs technologies servant à délivrer des services. Son but est de pousser les entreprises à externaliser les ressources numériques qu'elles stockent. Ces ressources offrant des capacités de stockage et de calcul, des logiciels de gestion de messagerie, et d'autres services sont mis à disposition par des sociétés tierces et accessibles, grâce à un système d'identification, via un PC et une connexion à Internet.

Historique :

Le cloud computing n'est pas nouveau, il est exploité depuis les années 2000, les changements qui ont permis l'apparition du cloud computing sont nombreux. Ainsi on peut cité l'apparition du SaaS (Software as a Service), le produit délivré par le cloud.

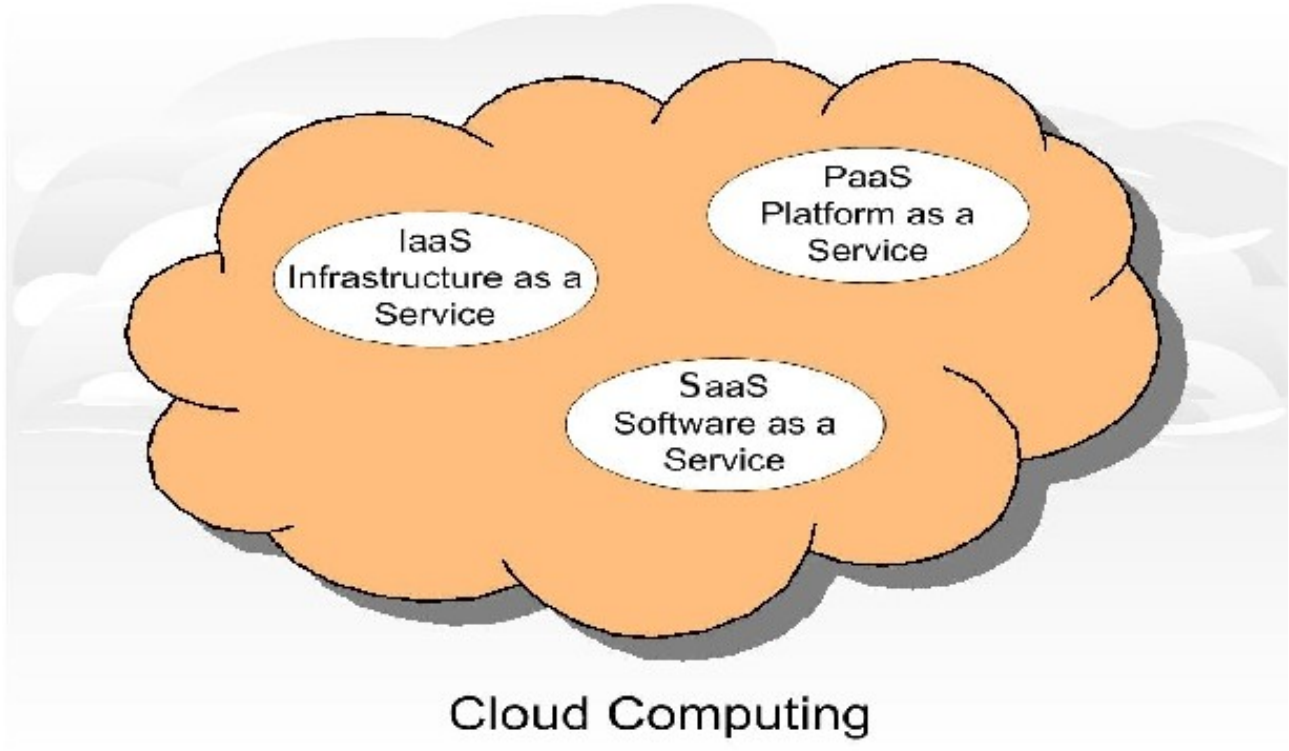
Puis il y a le concept de virtualisation qui permet une mutualisation des serveurs et offre donc une mise en production simplifiée et un meilleur ratio d'utilisation des ressources.

Le cloud computing est donc la juxtaposition de ces technologies pour passer à la vitesse supérieure sur l'exploitation de données à travers Internet.

Le concept du Cloud Computing a été mis en œuvre en 2002 par Amazon, un leader du e-business, pour absorber la charge importante des commandes faites sur leur site au moment des fêtes de Noël.

Récemment, d'autres acteurs comme Google et Microsoft proposent à leur tour des services similaires.

2.2. Les différents services



Le Cloud computing est composé de trois services, que nous allons exposer.

2.2.1. Iaas (Infrastructure as a Service)

Il s'agit de la mise à disposition, à la demande, de ressources d'infrastructures dont la plus grande partie est localisée à distance dans des Datacenters.

L'IaaS permet l'accès aux serveurs et à leurs configurations pour les administrateurs de l'entreprise.

Le client a la possibilité de louer des clusters, de la mémoire ou du stockage de données. Le coût est directement lié au taux d'occupation. Une analogie peut être faîte avec le mode d'utilisation des industries des commodités (électricité, eau, gaz) ou des Télécommunications.

- Avantage : grande flexibilité, contrôle total des systèmes (administration à distance par SSH ou Remote Desktop, RDP), qui permet d'installer tout type de logiciel métier.
- Inconvénient : besoin d'administrateurs système comme pour les solutions de serveurs classiques sur site.

Les cibles sont les responsables d'infrastructures informatiques. **Amazon EC2** est le principal qui propose ce genre d'infrastructures. **Eucalyptus** est un exemple d'infrastructure.

2.2.2. Paas (Plateform as a Service)

Il s'agit des plateformes du nuage, regroupant principalement les serveurs mutualisés et leurs systèmes d'exploitation. En plus de pouvoir délivrer des logiciels en mode SaaS, le PaaS dispose d'environnements spécialisés au développement comprenant les langages, les outils et les modules nécessaires.

L'avantage est que ces environnements sont hébergés par un prestataire basé à l'extérieur de l'entreprise ce qui permet de ne disposer d'aucune infrastructure et de personnel de maintenance et donc de pouvoir se consacrer au développement.

- Avantage : le déploiement est automatisé, pas de logiciel supplémentaire à acheter ou à installer.
- Inconvénient : limitation à une ou deux technologies (ex. : Python ou Java pour Google AppEngine, .NET pour Microsoft Azure, propriétaire pour force.com). Pas de contrôle des machines virtuelles sous-jacentes. Convient uniquement aux applications Web.

Les cibles sont les développeurs. **Google App Engine** est le principal acteur proposant ce genre d'infrastructures.

2.2.3. Saas (Software as a Service)

Concept consistant à proposer un abonnement à un logiciel plutôt que l'achat d'une licence. On oublie donc le modèle client-serveur et aucune application n'est installée sur l'ordinateur, elles sont directement utilisables via le navigateur Web.

L'utilisation reste transparente pour les utilisateurs, qui ne se soucient ni de la plateforme, ni du matériel, qui sont mutualisés avec d'autres entreprises.

Le Saas remplace l'ASP, aussi appelé fournisseur d'applications hébergées ou FAH, ou application service provider en anglais ou ASP, qui est une entreprise qui fournit des logiciels ou des services informatiques à ses clients au travers d'un réseau.

Deux principales différences avec l'ASP traditionnel sont qu'une simple interface web est utilisée côté client dans tous les cas (pas de client lourd), et que le SaaS propose une seule instance de logiciel qui évolue indépendamment des clients.

Avec l'arrivé du Haut débit, les logiciel en mode SaaS deviennent utilisables sans problèmes.

- Avantage : plus d'installation, plus de mise à jour (elles sont continues chez le fournisseur), plus de migration de données etc. Paiement à l'usage. Test de nouveaux logiciels avec facilité.
- Inconvénient : limitation par définition au logiciel proposé. Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel. Réactivité des applications Web pas toujours idéale.

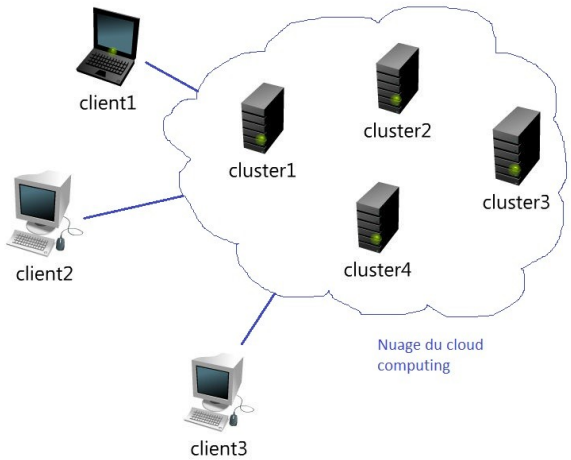
Les cibles sont les utilisateurs finaux. On retrouvera des entreprises comme **Sales Force** ou **Diva** (projet lancé par une équipe de l'école d'ingénieur EPITECH).

2.2.4 Avantages et Inconvénients des services

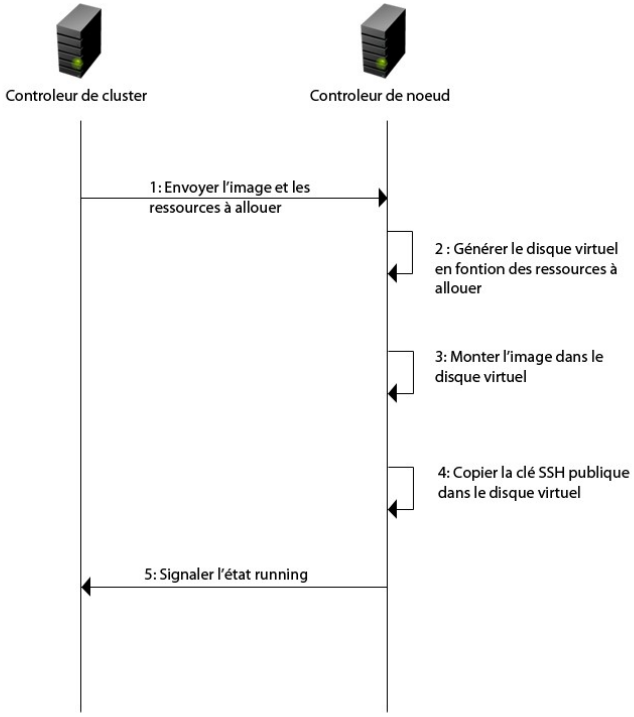
	Avantage	inconvénient
Saas	-pas d'installation -plus de licence -migration	-logiciel limité -sécurité -dépendance des prestataire
Paas	-pas d'infrastructure nécessaire -pas d'installation -environnement hétérogène	-limitation des langages -pas de personnalisation dans la configuration des machines virtuelles
Iaas	-administration -personnalisation -flexibilité d'utilisation	-sécurité -besoin d'un administrateur système

2.3. Cloud Computing et clusters

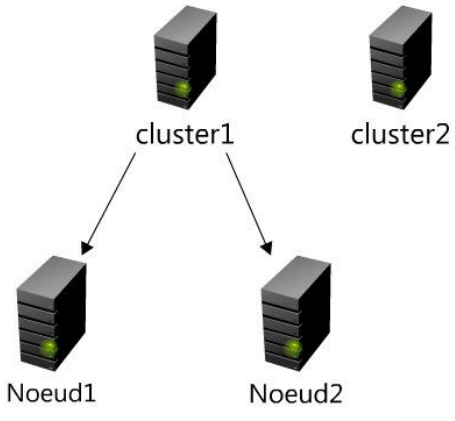
Le but du cloud computing est de construire un nuage de clusters, c'est à dire d'inter-connecter un ensemble de machines sur un réseau défini. Les utilisateurs peuvent ensuite déployer des machines virtuelles dans ce nuage, ce qui leur permet d'utiliser un certain nombre de ressources. Par exemple de l'espace disque, de la mémoire vive, ou encore du CPU (processeur).



Cette infrastructure, en allant plus dans le détail, est constituée de clusters et de nœuds. Les clusters servent à gérer l'interface entre les nœuds et l'utilisateur. Ainsi, lorsqu'on déploie une machine virtuelle sur un cluster, le cluster va créer une instance, qui se matérialisera par l'utilisation des ressources dans les nœuds. Voici un schéma UML récapitulatif qui décrit les étapes de déploiement d'une image.



L'utilisateur final disposera enfin d'un accès SSH sur la machine virtuelle. Pour lui, l'utilisation des ressources sera transparente. Ce sont des administrateurs réseaux qui lui délivreront ses ressources en fonction de ses besoins.



2.4. Avantages et inconvénients du Cloud Computing

Avantages :

✔Un démarrage rapide :

Le cloud computing permet de tester le business plan rapidement, à coûts réduits et avec facilité.

✔L'agilité pour l'entreprise :

Résolution des problèmes de gestion informatique simplement sans avoir à vous engager à long terme.

✔Un développement plus rapide des produits :

Réduisons le temps de recherche pour les développeurs sur le paramétrage des applications.

✔Pas de dépenses de capital :

Plus besoin des locaux pour élargir vos infrastructures informatiques

Inconvénients :

✔La bande passante peut faire exploser votre budget :

La bande passante qui serait nécessaire pour mettre cela dans le Cloud est gigantesque, et les coûts seraient tellement importants qu'il est plus avantageux d'acheter le stockage nous-mêmes plutôt que de payer quelqu'un d'autre pour s'en charger.

✔Les performances des applications peuvent être amoindries :

Un Cloud public n'améliorera définitivement pas les performances des applications.

✔La fiabilité du Cloud :

Un grand risque lorsqu'on met une application qui donne des avantages compétitifs ou qui contient des informations clients dans le Cloud,

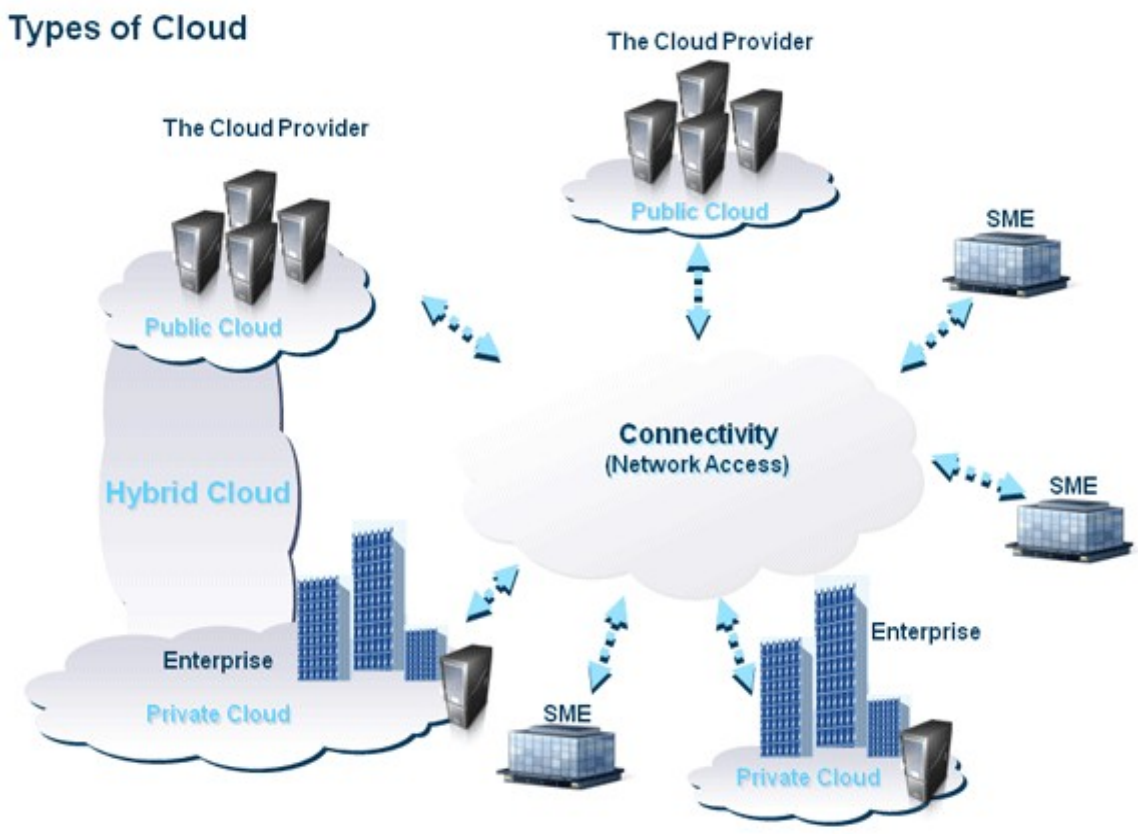
✔Taille de l'entreprise :

Si votre entreprise est grande alors vos ressources sont grandes, ce qui inclut une grande consommation du cloud. vous trouverez peut-être plus d'intérêt à mettre au point votre propre Cloud plutôt que d'en utiliser un externalisé. Les gains sont bien plus importants quand on passe d'une petite consommation de ressources à une consommation plus importante.

2.5. Types de Cloud Computing

Le concept de Cloud Computing est encore en évolution. On peut toutefois dénombrer trois types de Cloud Computing :

- le cloud privé (ou interne) : réseau informatique propriétaire ou un centre de données qui fournit des services hébergés pour un nombre limité d'utilisateurs.
- le cloud public (ou externe) : prestataire de services qui propose des services de stockage et d'applications Web pour le grand public. Ces services peuvent être gratuits ou payants. Exemples de clouds publics : Amazon Elastic Compute Cloud (EC2), Sun Cloud, IBM's Blue Cloud, Google AppEngine And Windows Azure Services Platform.
- le cloud hybride (interne et externe) : un environnement composé de multiples prestataires internes et externes. Un exemple, IBM avait conclu un partenariat avec Juniper Networks. Cette association a permit à Big Blue de déployer son offre de cloud hybride. Ainsi les entreprises qui utilisent ce service peuvent faire basculer, par un simple glisser-déposer, des applications hébergées dans un nuage privé interne vers un nuage public sécurisé.



2.6. Cloud Computing et OpenSource

Le cloud computing représente un nouveau défi dans le monde informatique, plusieurs solutions sont proposées : des solutions propriétaires et des solutions open source.

Beaucoup des solutions sont avant tout des solutions d'infrastructures permettant une gestion simplifiée d'architectures matérielles, les plus complexes. Dans ce domaine l'open source joue un rôle important dans les solutions de cloud computing. Voici quelques outils open source :

➔**Eucalyptus** : Eucalyptus est compatible avec Amazon Web Services. Il est intégré dans la distribution Linux Ubuntu 9.04 en tant qu'outils de "cloud computing". Eucalyptus peut s'installer facilement sur la majorité des distributions linux: Debian, CentOS...

➔**OpenNebula** : OpenNebula 1.2 supporte les plate-formes de virtualisation Xen et KVM ainsi que le service «on-demand» d'Amazon EC2. Parmi ses fonctionnalités : gestion centralisée des machines virtuelles et des ressources physique, répartition de charges, extension des capacités par ajout de serveurs physiques.

Pour la partie pratique, nous allons tester ces deux solutions et ensuite faire un bilan en montrant les avantages et les inconvénients de ces solutions.

3. Mise en place du Cloud Computing

3.1. Introduction

Afin de tester nos solutions, nous avons à disposition nos ordinateurs personnels ainsi que ceux de l'IUT. Nous avons déployer ces solutions en local, sur le réseau de la salle ASRALL et ensuite sur une plateforme appelée **Grid5000**.

Grid5000 est un projet d'informatique instrumental visant la réunion de 5000 « cœurs » de processeurs à l'échelle nationale (actuellement 9 sites sont impliqués). Cette plateforme est utilisée dans le cadre de projets expérimentaux, dans le domaine de la recherche.

On retrouve parmis ces 9 sites :

–Bordeaux

–Grenoble

–Lille

–Lyon

–Nancy

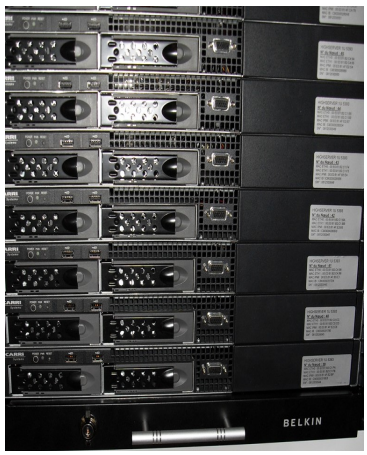
–Orsay

–Rennes

–Sophia

–Toulouse

Lors de notre projet, nous avons eu l'occasion de visiter le site de Nancy. Les serveurs se situent dans un laboratoire de recherche en informatique, appelé le LORIA. C'est grâce à notre tuteur de projet, Mr. Lucas Nussbaun, chercheur dans ce laboratoire, qu'on a pu utiliser cette plateforme. Voici quelques images des serveurs :



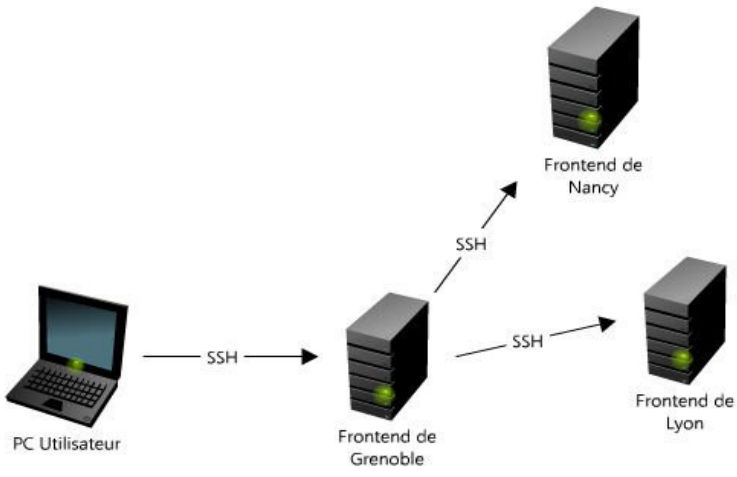
Comment utiliser Grid5000 ?

Grid5000, comme dit précédemment, est un ensemble de clusters réunis sur 9 sites. Ces sites sont reliés par un réseau haut débit, basé sur de la fibre optique, et appelé **Renater**. Les nouveaux membres s'inscrivent sur un wiki, à cette adresse : https://www.grid5000.fr. Ce sont ensuite des administrateurs qui jugent, en fonction du projet à réaliser, s'ils peuvent y travailler ou non.

Voici une photographie prise sur le site de Nancy. On peut y voir des câbles rouges, qui sont des câbles à fibre optique qui sont directement reliés au réseau **Renater**.

On peut se connecter à Grid5000 via le protocole SSH, sur le Front-End de la plateforme, qui se situe à Grenoble. De là, on peut seulement choisir sur quel site on souhaite se connecter, par exemple Lyon, ou Nancy dans notre cas.

Voici un schéma récapitulatif, décrivant ce procédé :



Maintenant nous allons voir ***les étapes pour déployer une machine virtuelle sur Grid5000***.

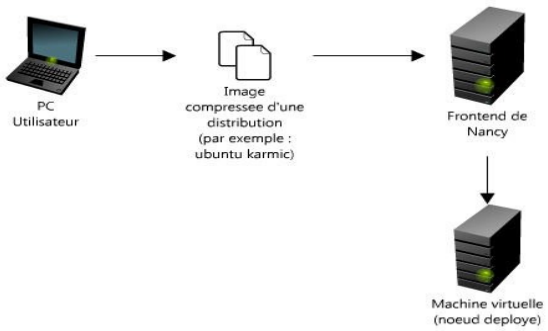
–On réserve des nœuds sur le site que l'on souhaite, à l'aide d'un utilitaire nommé OAR, pendant un certain temps (2 heures par exemple). Cet utilitaire est un ordonnanceur qui gère la réservation, ainsi que la terminaison de ces derniers une fois la fin de la connexion au Front-End du site, ou une fois le temps écoulé de la réservation.

–Ensuite on envoie une image compressée d'une distribution (par exemple : Ubuntu Karmic serveur) via SSH sur le même site.

–On déploie cette distribution sur le Front-End du site avec un utilitaire nommé Kadeploy3.

–On obtient une machine virtuelle où on peut se connecter en SSH.

Voici un schéma récapitulatif.



A présent, on peut directement se connecter en SSH sur cette machine depuis n'importe quel site de Grid5000, en utilisant la commande :

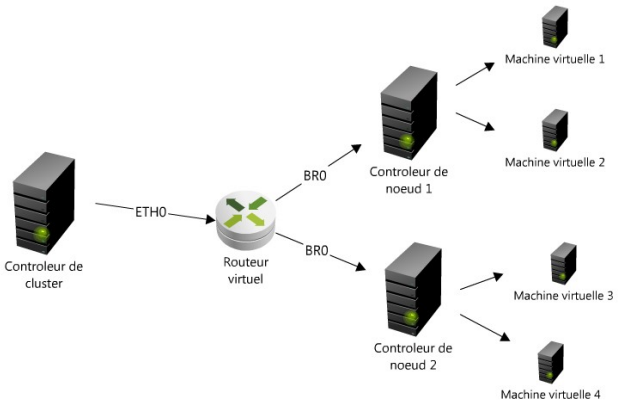
```
$ ssh root@node.site.grid5000.fr
```

Puis en donnant comme mot de passe : **grid5000**.

3.2. Eucalyptus

C'est la solution OpenSource la plus rependue de cloud computing. Eucalyptus est un accronyme de « Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems », qui se traduit littéralement par « Utilitaire d'Architecture informatique élastique pour relier vos programmes à des systèmes fonctionnels ». Il est compatible à **Amazon Web Services**, qui permet d'utiliser son **cloud computing privé sur un cloud computing public**. Cette solution est directement installable par les paquets de Ubuntu Karmic Serveur, que nous avons tester.

Nous allons voir un schéma qui va nous permettre d'expliquer comment se compose le nuage du cloud avec Eucalyptus.



Pour construire notre nuage, on va :

–d'abord créer un/plusieurs contrôleurs de clusters. Ils permettront de déployer les machines virtuelles dans le nuage, et d'allouer les ressources.

–Ensuite on va créer un/plusieurs contrôleurs de nœuds. Il représenteront la puissance de calcul du nuage. On va créer des machines virtuelles sur ces machines là.

Les contrôleurs de nœuds seront liés sur le(s) contrôleur(s) de nœuds par un ou plusieurs bridge(s). Eucalyptus utilise la librairie **libvirt** pour cela. Les machines virtuelles seront déployées avec un hyperviseur : soit **KVM**, soit **Xen**.

Nous allons voir quelles sont les étapes nécessaires à la construction du nuage.

•Dans un premier temps, on installera le(s) contrôleur(s) de cluster. Pour cela on installe **ubuntu karmic serveur** sur la/les machines. Puis on installe les paquets d'**eucalyptus-cc (contrôleur de cluster), eucalyptus-sc (contrôleur de stockage), et eucalyptus-walrus (réseau)**. On précisera lors de l'installation le nom du contrôleur de cluster, ainsi que les adresses IP à utiliser.

• On installera ensuite le(s) contrôleur(s) de nœud. Pour cela on installe **ubuntu karmic serveur** sur la/les machines. Puis on installe les paquets d'**eucalyptus-nc**.

• Sur le(s) contrôleur(s) de nœud, on mets en place une interface virtuelle (bridge br0). Le script permettant de faire sa est présent dans l'annexe 6.2 (**script 3, annexes**).

• Par défaut, un utilisateur eucalyptus est crée sur le contrôleur de nœud, avec un mot de passe inconnu. Il faut donc lui changer son mot de passe, afin qu'on puisse synchroniser les contrôleurs de nœuds, avec les contrôleurs de cluster. Pour cela, on se connecte sur le contrôleur de cluster, puis **on envoie la clé publique SSH sur tous les controleurs de** nœuds.

• Ensuite vient une **étape d'enregistrement sur les contrôleurs de clusters**. Pour cela on a 3 programmes qu'il suffit de lancer. Il s'agit d'eucalyptus-cc-registration, eucalyptus-sc-registration et eucalyptus-walrus-registration. Ces processus se chargeront de créer des certificats (fichiers .pem) qui permettront la sécurité du réseau dans le nuage.

• L'étape suivante sera d'enregistrer les contrôleurs de nœuds, c'est à dire d'envoyer ces fichiers .pem sur les contrôleurs de nœuds. Pour cela , on va dire au contrôleur de cluster de **découvrir les nœuds** présents sur le même réseau. Puis lorsqu'on enregistrera un contrôleur de nœud, le contrôleur de cluster lui enverra les certificats.

Maintenant que le nuage est créer, nous allons voir comment déployer les images.

–La première étape est d'aller **sur un contrôleur de cluster**, puis d'y ajouter des images. En allant dans la documentation d'eucalyptus, on a pu en télécharger des toutes faites (par exemple ubuntu). Ensuite un script se charge d'ajouter cette image à l'environnement d'eucalyptus (voir **script 2**, **annexes**, dans la section **10**). Ce script nous renverra un numéro EMI-xxxxxxxx. Ce numéro est consultable aussi au travers de la commande

```
$ euca-describe-images
```

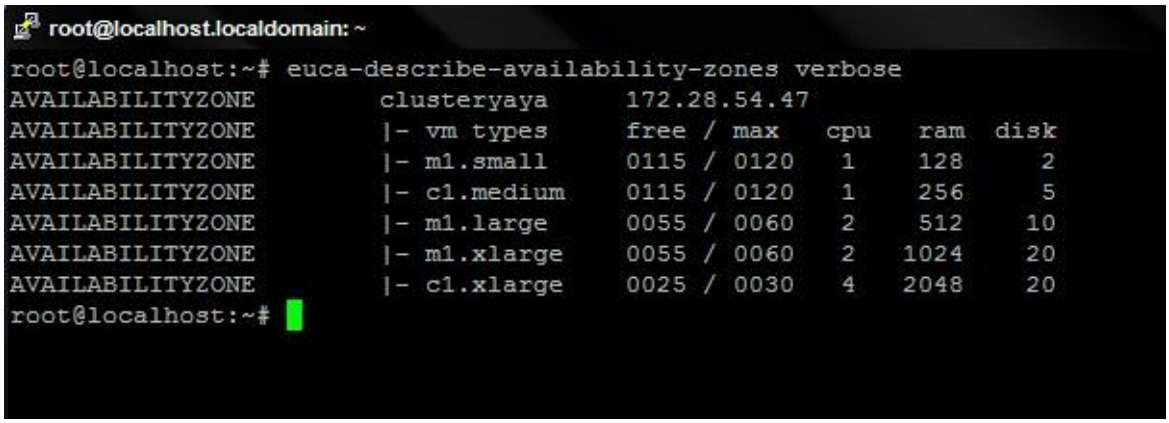
–L'étape suivante est de créer une clé SSH privée à notre environnement. Pour cela on va utiliser la commande *euca-add-keypairs*.

–On donnera les autorisations nécessaires à la connexion, grâce à la commande :

```
$ euca-authorize default -P tcp -p 22 -s 0.0.0.0/0
```

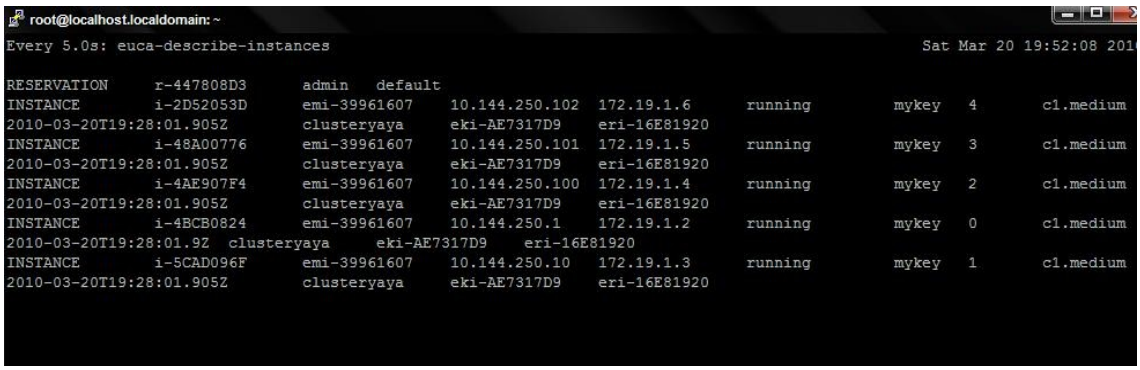
–On peut enfin déployer notre image, en utilisant notre image et l'identifiant de la clé SSH. La dernière option de la commande permet de préciser les types de ressources que l'on souhaite utiliser (grâce à une commande, on peut consulter les différents types)

```
$ euca-run-instances $EMI -k mykey -t c1.medium
```



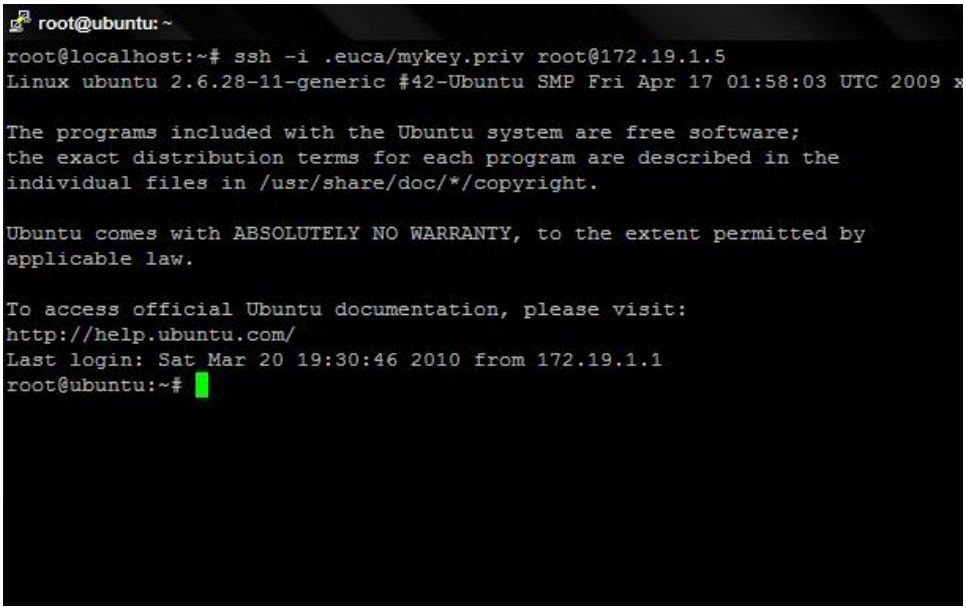
Dans notre cas, on va utiliser 256 Mo de RAM, 5 Go de disque et 1 processeur.

–On peut consulter les machines virtuelles déployées.



Les machines passent de l'état pending (en attente) à l'état running (prête). Lors de cette étape, le contrôleur de cluster se charge d'envoyer l'image à déployer sur le contrôleur de cluster, déploie cette image avec les bonne ressources à allouer, puis créer lors du déploiement une clé SSH publique, qu'il va mettre dans le disque virtuel. Une fois ceci fait, on peut se connecter dessus avec la clé SSH privée.

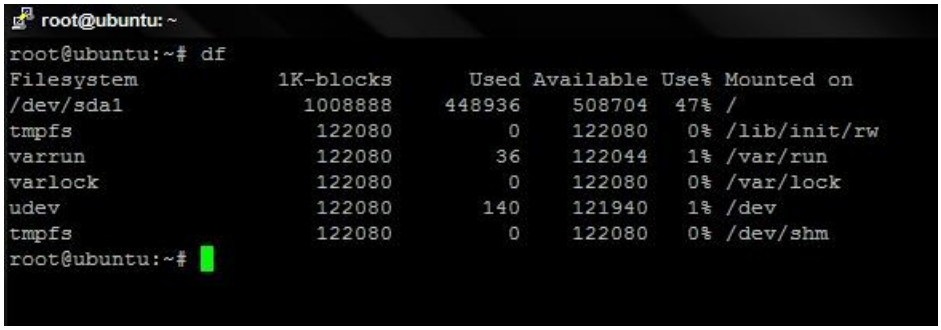
On a précisé dans la commande ssh l'adresse IP virtuelle, puis la clé SSH privée. On voit qu'on arrive bien à avoir cet accès.



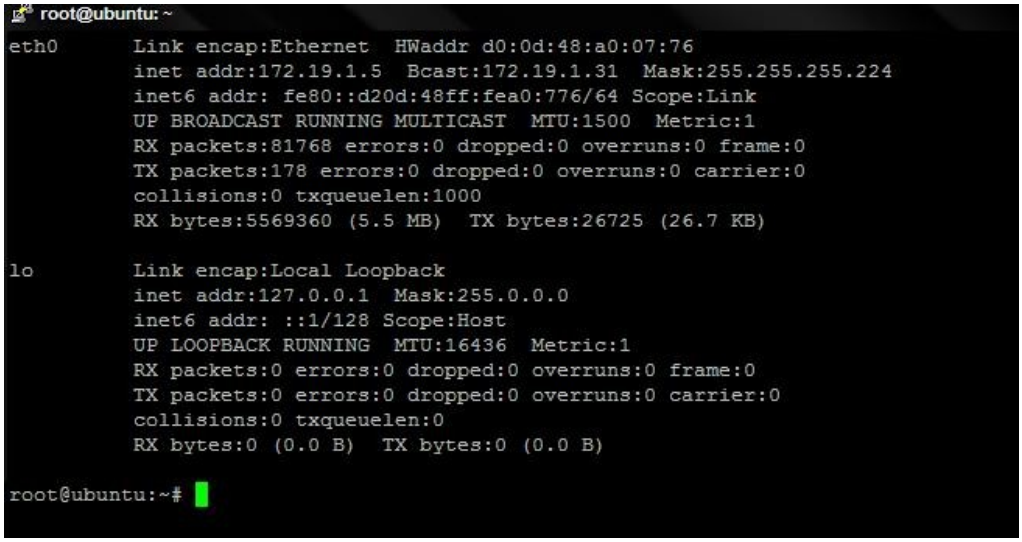
On peut enfin vérifier qu'on a les bonnes ressources allouées, et qu'on dispose de la bonne adresse IP.

Voici ce qui représente nos 256 Mo de RAM qu'on avait demandé

Voici ce qui représente nos 5 Go d'espace disque demandé



Et enfin notre adresse IP virtuelle



Malgré cela, on a été confronter à beaucoup de problèmes concernant cette solution ! On va donc faire un récapitulatif de ceux-là, puis voir comment nous les avons résolues.

- Un problème est arrivé lors de l'étape d'enregistrement. On arrivait pas à ce que le contrôleur de cluster nous crée les certificats (fichiers .pem). Pour le résoudre, il a fallu consulter le fichier de logs : /var/log/eucalyptus/cc-registration.log. Nous avons pu voir qu'il manquait des paquets à installer (wget et rsync).
- Un problème est arrivé lors de l'étape du déploiement de l'image sur le contrôleur de nœud.

  "Error : partition-to-disk image conversion command failed"

  Pour résoudre ce problème nous avons consulté le fichier de log /var/log/eucalyptus/nc.log sur le contrôleur de nœud.

  Nous avons pu voir qu'il manquait un paquet à installer (parted).

- Une fois la machine virtuelle passée en mode « running », on a rencontré un problème au niveau de la clé SSH. On arrivait pas à se connecter à la machine avec la clé privée. On nous demandait un mot de passe au lieu de pouvoir s'y connecter directement.

  $ ssh -i mykey.priv root@IP_machine_virtuelle

  password :

  Pour résoudre ce problème nous avons vérifié que la clé publique est bien transmise sur la machine virtuelle lors du déploiement. Nous avons monté le disque virtuel en se connectant sur le contrôleur de nœud.
  $ sfdisk -l instances/admin/numéro_instance/disk (lister les partitions des disques monté sur le disque virtuel)

$ mount -o loop, offset=$((63*512)) instances/admin/numéro_instance/disk /tmp/disk
(la commande permet de monter la partition virtuelle sur le contrôleur de nœud, dans un répertoire temporaire)
après consultation du répertoire : /tmp/disk/root/.ssh nous avons remarqué que le fichier authorized_keys était vide.

Nous avons donc chercher quel était le script qui copier cette clé SSH, qui était un fichier nommé /usr/share/eucalyptus/addkey.pl. En le consultant, nous avons remarqué que cette clé (publique) n'était pas copiée, lorsqu'on configurait eucalyptus dans un mode de configuration spécifique.

Nous nous somme donc interrogé à sa configuration pour changer les options possibles.

C'est en regardant la documentation, que nous avons remarqué que l'option VNET_MODE était fixée en mode MANAGED_NO_VLAN par défaut, et que la clé SSH n'était pas copiée dans ce mode en particulier. Nous avons donc essayé de passer ce mode dans une autre option possible : SYSTEM.

Cette modification n'a pas résolu le problème donc nous avons été sur plusieurs forums, ce qui nous as conduit à regarder si des bugs ont été corrigées sur des nouvelles versions. Nous avons vérifier la version du programme euca2ools qui permet de créer les clés SSH. Cette version n'était pas à jour car elle était en version 1.0 alors qu'il existait déjà une version 1.2.

C'est grâce à cette dernière modification que nous avons réussi à nous connecter via la clé SSH privée sur la machine virtuelle !

En conclusion, on peut dire d'Eucalyptus qu'il n'est pas encore au point. Il faut beaucoup de patience pour réussir à résoudre tous les problèmes qui se posent lors de son installation et que ce n'est pas un système si simple à installer que ça en à l'air.

3.3. OpenNebula

OpenNebula est une solution open-source pour construire facilement n'importe quel type de nuage :

◆**privé**

◆**public**

◆**hybride**

OpenNebula a été conçu pour être intégré à n'importe quel réseau et solution de stockage.

OpenNebula gère le **stockage**, le **réseau** et les **technologies de virtualisation** afin de permettre la mise en place dynamique de services multi-niveaux (des groupes de machines virtuelles interconnectées) sur les infrastructures distribuées, en combinant les ressources des machines physiques et des nuages à distance (machines virtuelles), en fonction des politiques d'allocation.

Fonctionnement d'OpenNebula

OpenNebula effectue les opérations suivantes :

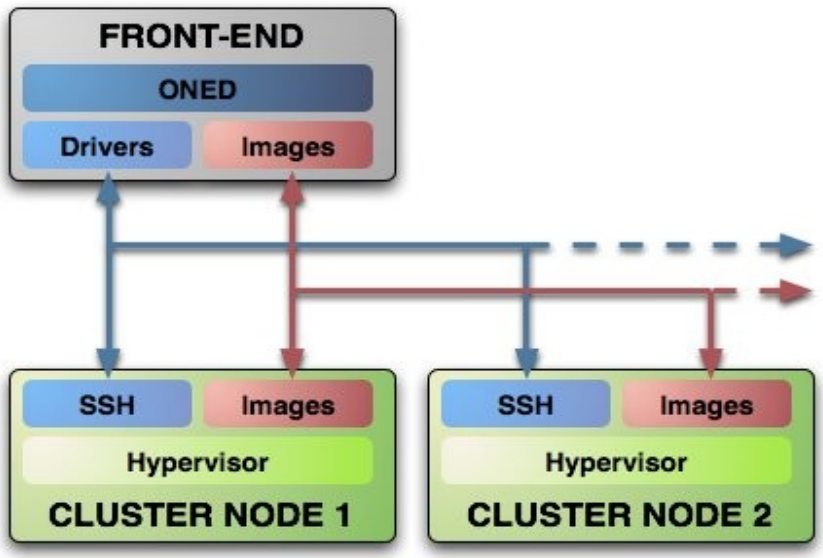
- **Gère les réseaux virtuels.** Le réseaux virtuels interconnectent des machines virtuelles entre elles. Chaque Réseaux virtuel intègre une description.
- **Crée des machines virtuelles.** Cette description est ajouté à la base de données.
- **Déploie des machines virtuelles.** Selon la politique d'attribution, l'ordonnanceur décide l'emplacement du lancement des machines virtuelles.
- **Gère les images des machines virtuelles.** Avant l'exécution, les images des machines virtuelles sont transférés à l'hôte et les images de disque de swap sont créés. Après l'exécution, les images peuvent être déplacées.
- **Gère les machines virtuelles lancées.** Lorsque les machines virtuelles sont démarrés, elles sont interrogés périodiquement afin de connaître leur consommation et leur état. Il est également possible d'éteindre, de suspendre, d'arrêter ou de migrer les machines virtuelles.

Les principales composantes d'un Cloud Computing avec OpenNebula sont les suivantes :

- **Hyperviseur:** Gestionnaire de virtualisation sur le cluster qui permet de voir et de modifier l'état des machines virtuelles.

- **Virtual Infrastructure Manager:** Assure la gestion du réseau, le cycle de vie des machines virtuelles et la tolérance aux pannes.
- **Scheduler** (planificateur) : Politiques d'équilibrage de la charge de travail des machines virtuelles.

Pour mettre en place la solution OpenNebula, il faut une architecture dite de 'cluster', c'est à dire un Front-End avec un ou plusieurs nœuds.



Les éléments de base d'un système OpenNebula sont les suivants :

- **Front-end.** Exécute OpenNebula et les services du cluster.
- **Nodes (nœuds).** Fournit les ressources nécessaires pour les machines virtuelles.
- **Image repository.** Tout support de stockage qui contient les images de base des machines virtuelles.
- **OpenNebula daemon.** Il est le service de base du système. Il gère le cycle de vie du VMS et les sous-systèmes du cluster (réseau, stockage et hyperviseurs)
- **Drivers.** Programmes utilisés par le processeur pour l'interfaçage avec un sous-système spécifique, par exemple un système de stockage de fichiers.
- **Oneadmin.** Est l'administrateur du Cloud qui effectue des opérations sur les machines virtuelles, les réseaux virtuels, les nœuds ou les utilisateurs.
- **Utilisateurs.** Utilisent OpenNebula pour créer et gérer leurs propres machines virtuelles et réseaux virtuels.

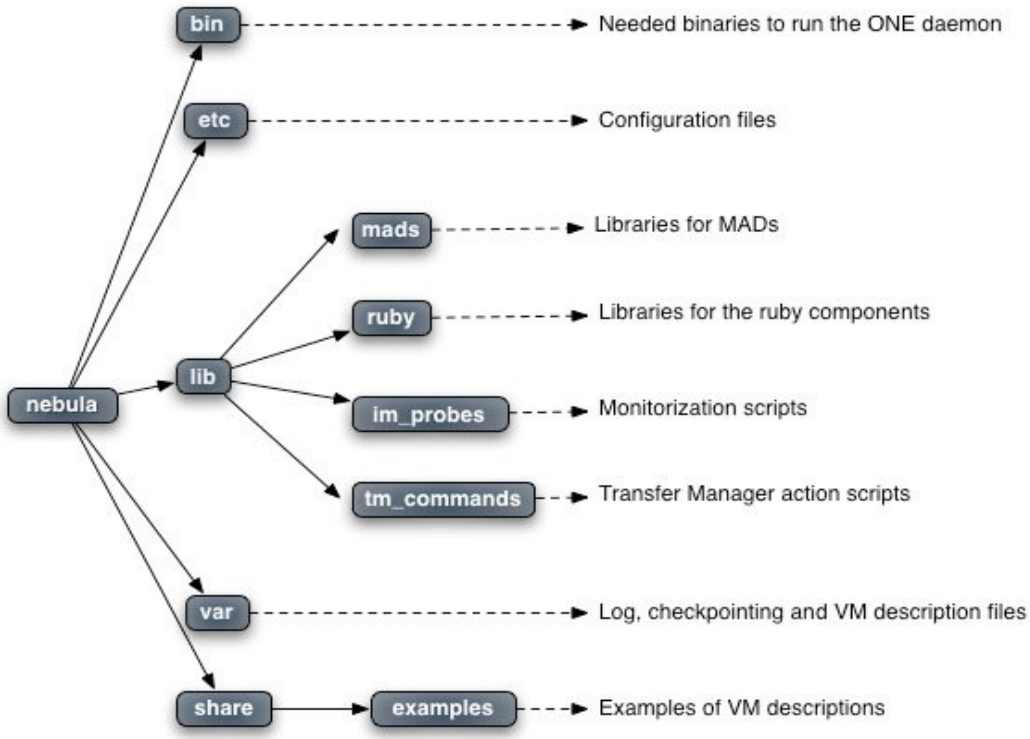
OpenNebula utilise la librairie **libvirt**. Les machines virtuelles seront déployées avec un hyperviseur : soit **KVM**, soit **Xen,** soit **Vmware**.

Installation

On peut installer OpenNebula de deux manières :

✗ Soit par paquet en le téléchargeant sur le site officiel.
✗ Soit en utilisant simplement 'apt-get install opennebula', et pour le nœud : 'apt-get install opennebula-node'.

Une fois installé, voici une schéma récapitulatif des répertoires sur le front-end :



Un utilisateur est automatiquement créé à la fin de l'installation (oneadmin).

Front-End ↔ griffon-6
Nœud ↔ griffon-7

La première étape est d'ajouter des nœuds à notre Front-End :

```
onehost add griffon-7 im_kvm vmm_kvm tm_ssh
```

Les arguments im_kvm vmm_kvm tm_ssh sont définis dans le fichier de configuration /etc/nebula/one.conf et renseignent respectivement les outils kvm et ssh.

➔ On peut y ajouter le nombre de nœuds que l'on souhaite. Plus Il y a de nœuds connectés au Front-End et plus il y aura de ressources disponibles pour déployer les machines virtuelles.

Après avoir installé le paquet « opennebula-node » sur les nœuds, il faut maintenant synchroniser la communication entre le Front-End et les nœuds en ajoutant la clef publique rsa pour la connexion ssh de l'utilisateur 'oneadmin' du Front-End dans la liste des clefs autorisés sur le/les noeud(s) déployé(s).

La clef apparaît automatiquement (dans le terminal) lors de l'installation, il suffit de la copier en remplaçant « localdomain » par l'IP/DNS du Front-End (ici « griffon-6 ») :

```
sudo  tee /var/lib/one/.ssh/authorized_keys << EOT
ssh-rsaIwAAAQEA...rzMqHw==oneadmin@griffon-6
EOT
```

Pour tester si la synchronisation est validé, utiliser cette commande, coté Front-End :

```
sudo -u oneadmin ssh griffon-7
```

L'installation est maintenant terminée.

Pour lister les nœuds connectés :

```
onehost list
```

```
root@localhost:/tmp# onehost list
  HID NAME                    RVM   TCPU   FCPU   ACPU    TMEM    FMEM STAT
    0 griffon-7                 0    800    800    800 1646949 1605041   on
```

Il faut préparer un réseau virtuel pour nos futures machines. Il suffit de créer un fichier de la forme :

```
NAME   = "Réseau Virtuel"
TYPE   = FIXED
BRIDGE = virbr0
LEASES = [IP=10.144.120.120]
```

NAME → Nom que l'on définit à notre réseau virtuel
TYPE → Pour fixer une IP ou une plage d'adresse IP
BRIDGE → Interface des futurs machines virtuelles
LEASES → Adresses IP des machines virtuelles

Pour créer/ajouter un réseau virtuel :

onevnet create <nom-fichier>

Puis, récapitulatif des réseaux virtuels :

onevnet list

```
root@localhost:/tmp# onevnet list
 NID NAME              TYPE BRIDGE
   0 Reseau Virtuel    Fixed virbr0
```

On peut obtenir des détails sur un réseau virtuel :

onevnet show <name-reseau_ou_id>

```
root@localhost:/tmp# onevnet show 0
NID             : 0
UID             : 0
Network Name    : Reseau Virtuel
Type            : Fixed
Bridge          : virbr0

....: Template :....
        BRIDGE=virbr0
        LEASES=IP=10.144.120.120
        NAME=Reseau Virtuel
        TYPE=FIXED

....: Leases :....
IP = 10.144.120.120 MAC = 00:03:0a:90:78:78  USED = 0 VID = -1
```

De la même manière que la création du réseau virtuel, il faut maintenant ajouter les machines définies sous forme de fichiers :

```
NAME   = machine-virtuelle-1
    CPU    = 1
    MEMORY = 512
    OS      = [ BOOT   = hd ]
    DISK   = [
       source   = "/tmp/debian.5-0.x86.img",
       target   = "sda",
       readonly = "no" ]
    DISK   = [
       type     = "swap",
       size     = 1024,
       target   = "sdb"]
    NIC    = [ NETWORK = "Reseau Virtuel" ]
```

BOOT → Choix du périphérique de démarrage.
NAME → Nom de la machine virtuelle.
CPU → Nombre de CPU alloué à la machine virtuelle.
MEMORY → Mémoire allouée à la machine virtuelle.
OS → Emplacement de l'OS installé dans l'image.
(1) DISK → Emplacement de l'image source.
(2) DISK → Création d'un autre disque swap.
NIC → Réseau virtuel utilisé

Démarrer enfin le déploiement de la machine virtuelle :

onevm create <nom-fichier>

Voici les étapes du déploiement :

- **Création d'un répertoire coté nœud pour l'emplacement de l'image.**
- **Copie de l'image du Front-End dans le répertoire du nœud.**
- **Montage de l'image et du swap.**
- **Initialisation du swap.**
- **La machine virtuelle BOOT.**
- **Génération d'un fichier de déploiement coté Front-End.**
- **Copie de ce fichier sur le nœud.**
- **Connexion au réseau virtuel.**
- **La machine virtuel est démarré => état RUNNING.**

Pour voir toutes les machines ainsi que leurs états :

```
onevm list
```

```
root@localhost:/tmp# onevm list
  ID     NAME STAT CPU     MEM        HOSTNAME        TIME
   0 machine- pend   0       0                00 00:00:17
root@localhost:/tmp# onevm list
  ID     NAME STAT CPU     MEM        HOSTNAME        TIME
   0 machine- prol   0       0       griffon-7 00 00:01:09
```

```
root@localhost:/tmp# onevm list
  ID     NAME STAT CPU     MEM        HOSTNAME        TIME
   0 machine- runn   0  524288       griffon-7 00 00:06:23
```

On peut également voir les logs de chaque machine virtuelle dans /var/log/one/<n°ID>.log. (voir annexes).

A la fin du projet, nous sommes arrivés à déployer des machines virtuelles comme nous le montre les images ci-dessus mais nous n'arrivons pas à y accéder en ssh. Nous n'arrivons pas à pinger la machine virtuelles.

```
root@localhost:/tmp# ping 10.144.120.120
PING 10.144.120.120 (10.144.120.120) 56(84) bytes of data.
From 10.147.255.254 icmp_seq=7 Destination Host Unreachable
From 10.147.255.254 icmp_seq=8 Destination Host Unreachable
From 10.147.255.254 icmp_seq=9 Destination Host Unreachable
From 10.147.255.254 icmp_seq=10 Destination Host Unreachable
^C
--- 10.144.120.120 ping statistics ---
10 packets transmitted, 0 received, +4 errors, 100% packet loss, time 9006ms

root@localhost:/tmp# ssh 10.144.120.120
ssh: connect to host 10.144.120.120 port 22: No route to host
```

Sur le Front-End, on peut avoir des informations sur les machines virtuelles :

```
root@localhost:/tmp# onevm show 2
VID           : 2
UID           : 0
STATE         : ACTIVE
LCM STATE     : RUNNING
DEPLOY ID     : one-2
MEMORY        : 524288
CPU           : 0
LAST POLL     : 1269423353
START TIME    : 03/24 09:09:11
STOP TIME     : 01/01 00:00:00
NET TX        : 0
NET RX        : 0

....: Template :....
    CPU             : 1
    DISK            : READONLY=no,SOURCE=/tmp/debian.5-0.x86.img,TARGET=sda
    DISK            : SIZE=1024,TARGET=sdb,TYPE=swap
    MEMORY          : 512
    NAME            : machine-virtuelle-1
    NIC             : BRIDGE=virbr0,IP=10.144.120.120,MAC=00:03:0a:90:78:78,NETWORK=Reseau Virtuel,VNID=0
    OS              : BOOT=hd
```

Sur le nœud, on voit bien le bridge avec les interfaces virtuelles des machines :

```
virbr0    Link encap:Ethernet  HWaddr 2e:96:38:0f:e2:9b
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

vnet0     Link encap:Ethernet  HWaddr 2e:96:38:0f:e2:9b
          inet6 addr: fe80::2c96:38ff:fe0f:e29b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2763 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:143832 (143.8 KB)

vnet1     Link encap:Ethernet  HWaddr 76:a8:65:69:37:63
          inet6 addr: fe80::74a8:65ff:fe69:3763/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2215 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:115336 (115.3 KB)

vnet2     Link encap:Ethernet  HWaddr 36:73:ec:0a:c6:b5
          inet6 addr: fe80::3473:ecff:fe0a:c6b5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:630 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:32916 (32.9 KB)
```

Nous avons tester des commandes sur le nœud :

```
root@localhost:/var/lib/one/1/images# virsh list
Connecting to uri: qemu:///system
 Id Name                 State
----------------------------------
  1 one-0                running
  2 one-1                running
  3 one-2                running

root@localhost:/var/lib/one/1/images# ps aux | grep kvm
root     17183  0.0  0.0   6068   636 pts/0    R+   09:33   0:00 grep kvm
root     19116  0.0  0.0 653760 14060 ?        Sl   08:03   0:03 /usr/bin/kvm -S -M pc-0.11 -m 512 -smp 1 -name one-0 -uuid
 9ec63a2f-4ea9-0b53-96fd-9f7c44bb30e9 -nographic -monitor unix:/var/run/libvirt/qemu/one-0.monitor,server,nowait -boot c -d
rive file=/var/lib/one//0/images/disk.0,if=scsi,index=0,boot=on -drive file=/var/lib/one//0/images/disk.1,if=scsi,index=1 -
net nic,macaddr=00:03:0a:90:78:78,vlan=0,name=nic.0 -net tap,fd=14,vlan=0,name=tap.0 -serial none -parallel none -usb
root     25379  0.0  0.0 653760 14056 ?        Sl   08:22   0:03 /usr/bin/kvm -S -M pc-0.11 -m 512 -smp 1 -name one-1 -uuid
 bf555154-bd2a-1aa8-64fd-f6b85068707e -nographic -monitor unix:/var/run/libvirt/qemu/one-1.monitor,server,nowait -boot c -d
rive file=/var/lib/one//1/images/disk.0,if=scsi,index=0,boot=on -drive file=/var/lib/one//1/images/disk.1,if=scsi,index=1 -
net nic,macaddr=00:03:0a:90:78:78,vlan=0,name=nic.0 -net tap,fd=15,vlan=0,name=tap.0 -serial none -parallel none -usb
root     31358  0.1  0.0 653760 14064 ?        Sl   09:14   0:01 /usr/bin/kvm -S -M pc-0.11 -m 512 -smp 1 -name one-2 -uuid
 1407a9a2-1fd4-6ab5-3a43-797f8949fedb -nographic -monitor unix:/var/run/libvirt/qemu/one-2.monitor,server,nowait -boot c -d
rive file=/var/lib/one//2/images/disk.0,if=scsi,index=0,boot=on -drive file=/var/lib/one//2/images/disk.1,if=scsi,index=1 -
net nic,macaddr=00:03:0a:90:78:78,vlan=0,name=nic.0 -net tap,fd=16,vlan=0,name=tap.0 -serial none -parallel none -usb
root@localhost:/var/lib/one/1/images# socat STDIO UNIX-CONNECT:/var/run/libvirt/qemu/one-1.monitor
```

```
root@localhost:/var/lib/one/1/images# socat STDIO UNIX-CONNECT:/var/run/libvirt/qemu/one-1.monitor
```

Problèmes rencontrés :

•Il faut que la virtualisation soit supporté par le PC.

- egrep '^flags.*(vmx|svm)' /proc/cpuinfo >/dev/null && echo supporte || echo non_supporte

- Attention à la taille coté nœuds
  - Avec Grid5000, le répertoire /tmp a beaucoup d'espace, il faut donc monter l'espace disponible de
  - /tmp dans le répertoire ou est copié l'image (/var/lib/one sur le nœud)
  - Montage du répertoire
    - mv /var/lib/one /tmp/one
    - mount - - bind /tmp/one /var/lib/one
    - OU
    - ln -s /tmp/one /var/lib/one
- Problème de copie de l'image du Front-End vers le nœud
  - Script exécutant la copie : /usr/lib/one/tm_commands/ssh/tm_clone.sh
  - Changer le premier argument (SRC) par l'emplacement de l'image à copier (/tmp/debian.5-0.x86.img)
- Problème de permission
  - Il faut rajouter l'utilisateur oneadmin au groupe libvirtd
- Attention au fichier de création de la machine virtuelle (sda, sda1, sdb, sdb1 …..)

3.4. Comparaison des deux solutions

	Eucalyptus	**Opennebula**
Installation et mise en œuvre	Simple	Très simple et très rapide
Configuration	Compliquée	Moyen
Lancement d'une machine virtuelle	Beaucoup de commandes pour le lancement	Seulement en 2 commandes
Communauté	Très forte	Présente
Infrastructure minimum	1 frontend et 1 nœud	1 frontend et 1 nœud
Hyperviseurs supportés	Xen, Kvm, Vmware	Xen, Kvm et Vmware
Difficultés rencontrés	De nombreuses difficultés rencontrés, qui ont été difficilement résolues	De gros problèmes rencontrés avec quelques solutions, mais pas efficaces

L'installation et la mise en œuvre des deux solutions Open Source a été simple dans l'ensemble mais on peut noter plus de difficultés pour la solution Eucalyptus avec un fichier de configuration plus complet.

Lancement d'une machine virtuelle :

La mise en œuvre d'une machine virtuelle en utilisant la solution d'Eucalyptus demande beaucoup de commandes, nous avons transmis ces derniers sous formes de scripts pour une utilisation simplifié.

Dans le cas d'OpenNebula, seul deux commandes suffisent pour lancer une machine virtuelle.

La communauté d'Eucalyptus est très présente, nous pouvons le voir sur leur site officiel avec de nombreuses questions posées et nombreux forums. Au contraire OpenNebula a une communauté réduite avec quelques questions et très peu de réponses.

La mise en œuvre des deux solutions demande un infrastructure minimale qui consiste à avoir un Front-End (controleur de nœud) et un ou plusieurs nœuds. C'est à dire avoir une infrastructure dite 'cluster'. On peut également ajouter que l'ajout de plusieurs nœuds permet d'avoir plus de ressources.

Les deux solutions Open Source supportent différents *Hyperviseurs,* qui sont les mêmes pour les deux solutions. Ils supportent tous les deux : KVM, XEN, VMWARE.

Durant l'installation, la mise en œuvre et la configuration des deux solutions nous avons rencontré de *nombreuses difficultés* au niveau d'Eucalyptus qui ont été résolue après des longues recherches et de nombreux tests.

Par contre, au niveau d'OpenNebula nous avons eu plusieurs problèmes qui ont été résolu partiellement mais nous restons bloqué, malgré plusieurs semaines de recherches, sur le fait d'impossibilité d'accéder aux machines virtuelles.

4. Conclusion

Ce projet nous a aider à travailler en groupe, à nous aider les uns les autres face à de nombreuses difficultés.

Nous avons approfondis nos connaissances dans le domaine de la virtualisation.

Les nombreux déploiements et recherches que nous étions amenés à faire nous ont largement familiarisés avec les solutions de virtualisation existantes et malgré le manque de documentation et l'absence de communauté Francophone autour de ces premières solutions, le Cloud Computing a un avenir très prometteur.

5. Bibliographie

http://fr.wikipedia.org/wiki/Informatique_dans_les_nuages

http://en.wikipedia.org/wiki/Cloud_computing

https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home

http://www.eucalyptus.com/

http://open.eucalyptus.com/

http://open.eucalyptus.com/wiki/EucalyptusImageManagement_v1.6

http://open.eucalyptus.com/wiki/EucalyptusUserImageCreatorGuide_v1.6

https://help.ubuntu.com/community/UEC

http://www.opennebula.org/start

https://help.ubuntu.com/community/OpenNebula

6. Annexes

6.1. Répartition des tâches

Notre projet a été réparti sur environ huit semaines (~ 160h). Durant quatre semaines, nous avons travaillé tous les quatre ensemble afin de résoudre les premiers problèmes persistants. Nous avons principalement travaillé sur les principes du Cloud Computing puis sur les parties pratique (Eucalyptus et OpenNebula).

Après ces quatre semaines, nous avons décider de répartir les tâches de la façon suivante :

Mohammed et Yannick -------> Eucalyptus

Vincent et Allan -------> OpenNebula

6.2 Scripts pour eucalyptus

SCRIPT 1 : Déploiement de nœuds sur Grid5000 avec envoie de l'image à déployer sur la machine virtuelle, envoie de scripts sur le Front-End et envoie de clé SSH.

```
if [ "${1}" = "1" ]
then
        echo "reservation pour le controleur cluster ..."
        oarsub -I -t deploy -l nodes=1,walltime=12
elif [ "${1}" = "2" ]
then
        echo "recopie le oarnode pour le controleur de cluster ..."
        cat $OAR_NODEFILE | uniq > cluster.txt
        echo "deploiement du controleur de cluster ..."
        kadeploy3 -e ubuntu-karmic -f $OAR_NODEFILE
elif [ "${1}" = "3" ]
```

```
then
	echo "reservation pour le controleur noeud ..."
	oarsub -I -t deploy -l nodes=8,walltime=12
elif [ "${1}" = "4" ]
then
	echo "recopie le oarnode pour les controleurs de noeud ..."
	cat $OAR_NODEFILE | uniq > noeuds.txt
	echo "deploiement des controleurs de noeud (les paquets nescessaires sont deja installés
dessus) ..."
	kadeploy3 -e ubuntu-karmic-nc2 -f $OAR_NODEFILE
elif [ "${1}" = "5" ]
then
	echo "creation des br0 sur les controleurs de noeuds"
	./creerbr0.sh
	echo "envoi des fichiers sur le cluster ..."
	for machine in `cat cluster.txt`
	do
			ssh-copy-id -i ~/.ssh/id_rsa.pub root@$machine
			scp euca2ools-1.2-bzr254-src.tar.gz euca-ubuntu-9.04-x86_64.tar.gz
scriptcluster.sh noeuds.txt root@$machine:
			ssh root@$machine "chmod a+x scriptcluster.sh"
			ssh root@$machine
	done
elif [ "${1}" = "6" ]
then
	echo "connection au controleur de cluster ..."
	for machine in `cat cluster.txt`
	do
			ssh root@$machine
	done
else
	echo "------------------------------------------------------------------------"
	echo " - Vous devez specifier un nombre, pour dire quelle etape vous voulez faire "
	echo "------------------------------------------------------------------------"
	echo
	echo " ./deploi 1 :: reservation du controleur de cluster"
	echo " ./deploi 2 :: deploiement du cc"
	echo " ./deploi 3 :: reservation du controleur de noeuds"
	echo " ./deploi 4 :: deploiement du,des nc"
	echo " ./deploi 5 :: creation du br0, envoi des scripts sur le cc"
	echo " ./deploi 6 :: connection au controleur de cluster ..."
	echo "------------------------------------------------------------------------"
fi
```

SCRIPT 2 : A lancer sur le Front-End (installation, configuration, lancement de machine virtuelle)

```
# 1) on installe les paquets sur le controleur de cluster
apt-get install eucalyptus-cloud eucalyptus-cc eucalyptus-walrus eucalyptus-sc
apt-get install wget rsync vim

# 2) on envoie les clés SSH sur les controleurs de noeuds
for machine in `cat noeuds.txt`
do
        sudo -u eucalyptus ssh-copy-id -i ~eucalyptus/.ssh/id_rsa.pub eucalyptus@$machine
done

# 3) on ajoute le nom du contrôleur de cluster dans la conf d'eucalyptus
sudo tee -a /etc/eucalyptus/eucalyptus.conf <<EOF
        CC_NAME="clusteryaya"
EOF

# 4) les étapes d'enregistrements des différents services : controleur de cluster,
#     controleur de stockage, et réseau.
tail /etc/eucalyptus/eucalyptus.conf
sleep 1
/etc/init.d/eucalyptus-cc-registration start
sleep 1
/etc/init.d/eucalyptus-sc-registration start
sleep 1
/etc/init.d/eucalyptus-walrus-registration start

# 5) on découvre les contrôleurs de nœuds
euca_conf --no-rsync --discover-nodes

# 6) on supprime euca2ools (version 1.0)
apt-get remove euca2ools

# 7) on télécharge les credentials
mkdir -p .euca;
euca_conf --get-credentials .euca/mycreds.zip;
unzip .euca/mycreds.zip;
. eucarc;

# 8) on installe euca2ools (version 1.2)
tar -xvf euca2ools-1.2-bzr254-src.tar.gz
make -C  euca2ools-1.2-bzr254-src
make install -C  euca2ools-1.2-bzr254-src

# 9) on vérifie que euca2ools fonctionne
```

```
euca-describe-availability-zones verbose;

# 10) on ajoute une image de distribution à l'environnement d'eucalyptus
tar zxvf euca-ubuntu-9.04-x86_64.tar.gz
euca-bundle-image -i euca-ubuntu-9.04-x86_64/kvm-kernel/vmlinuz-2.6.28-11-generic --kernel
true
euca-upload-bundle -b ubuntu-kernel-bucket -m /tmp/vmlinuz-2.6.28-11-generic.manifest.xml
EKI=$(euca-register ubuntu-kernel-bucket/vmlinuz-2.6.28-11-generic.manifest.xml | grep
"^IMAGE" | awk '{print $2}') && echo $EKI

euca-bundle-image -i euca-ubuntu-9.04-x86_64/kvm-kernel/initrd.img-2.6.28-11-generic --ramdisk
true
euca-upload-bundle -b ubuntu-ramdisk-bucket -m /tmp/initrd.img-2.6.28-11-generic.manifest.xml
ERI=$(euca-register ubuntu-ramdisk-bucket/initrd.img-2.6.28-11-generic.manifest.xml | grep
"^IMAGE" | awk '{print $2}') && echo $ERI

euca-bundle-image -i euca-ubuntu-9.04-x86_64/ubuntu.9-04.x86-64.img --kernel $EKI --ramdisk
$ERI
euca-upload-bundle -b ubuntu-image-bucket -m /tmp/ubuntu.9-04.x86-64.img.manifest.xml
EMI=$(euca-register ubuntu-image-bucket/ubuntu.9-04.x86-64.img.manifest.xml | grep "^IMAGE"
| awk '{print $2}') && echo $EMI

# 11) on ajoute les clés SSH, autorise le port 22, et déploie une machine
euca-add-keypair yaya > yaya.priv;
chmod 0600 yaya.priv;
euca-describe-groups;
euca-authorize default -P tcp -p 22 -s 0.0.0.0/0;
euca-run-instances -k yaya $EMI -t c1.medium

# 12) on regarde quelles sont les instances lancées
watch -n 5 euca-describe-instances
```

SCRIPT 3 : Création des bridges sur les nœuds

```
# lance le script creerbr0 sur chaque contrôleur de nœud
for machine in `cat noeuds.txt`
do
	ssh root@$machine "./creerbr0"
done


# voici le script creerbr0 sur les contrôleurs de nœud
interface=eth0
bridge=br0
sudo sed -i "s/^iface $interface inet \(.*\)$/iface $interface inet manual\n\nauto br0\niface $bridge
inet \1/" /etc/network/interfaces
sudo tee -a /etc/network/interfaces <<EOF
	bridge_ports $interface
	bridge_fd 9
	bridge_hello 2
	bridge_maxage 12
	bridge_stp off
EOF
sudo /etc/init.d/networking restart
```

6.3 OpenNebula

Logs du déploiement :

Principale erreur rencontré :

Tue Mar 23 08:36:01 2010 [DiM][I]: New VM state is ACTIVE.
Tue Mar 23 08:36:02 2010 [LCM][I]: New VM state is PROLOG.
Tue Mar 23 08:36:02 2010 [TM][I]: tm_clone.sh: localhost.localdomain:/tmp/debian.5-0.x86.img griffon-18:/var/lib/one//0/images/disk.0
Tue Mar 23 08:36:02 2010 [TM][I]: tm_clone.sh: DST: /var/lib/one//0/images/disk.0
Tue Mar 23 08:36:02 2010 [TM][I]: tm_clone.sh: Creating directory /var/lib/one//0/images
Tue Mar 23 08:36:02 2010 [TM][I]: tm_clone.sh: Executed "ssh griffon-18 mkdir -p /var/lib/one//0/images".
Tue Mar 23 08:36:02 2010 [TM][I]: tm_clone.sh: Cloning /tmp/debian.5-0.x86.img
Tue Mar 23 08:39:36 2010 [TM][I]: tm_clone.sh: Executed "scp /tmp/debian.5-0.x86.img griffon-18:/var/lib/one//0/images/disk.0".
Tue Mar 23 08:39:36 2010 [TM][I]: tm_clone.sh: Executed "ssh griffon-18 chmod a+w /var/lib/one//0/images/disk.0".
Tue Mar 23 08:39:36 2010 [TM][I]: tm_mkswap.sh: Creating 1024Mb image in /var/lib/one//0/images/disk.1
Tue Mar 23 08:39:36 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 mkdir -p /var/lib/one//0/images".
Tue Mar 23 08:39:36 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 dd if=/dev/zero of=/var/lib/one//0/images/disk.1 bs=1 count=1 seek=1024M".
Tue Mar 23 08:39:36 2010 [TM][I]: tm_mkswap.sh: Initializing swap space
Tue Mar 23 08:39:36 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 /sbin/mkswap /var/lib/one//0/images/disk.1".
Tue Mar 23 08:39:36 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 chmod a+w /var/lib/one//0/images/disk.1".
Tue Mar 23 08:39:36 2010 [LCM][I]: New VM state is BOOT
Tue Mar 23 08:39:36 2010 [VMM][I]: Generating deployment file: /var/lib/one/0/deployment.0
Tue Mar 23 08:39:36 2010 [VMM][I]: Command: scp /var/lib/one/0/deployment.0 griffon-18:/var/lib/one//0/images/deployment.0
Tue Mar 23 08:39:37 2010 [VMM][I]: Copy success
Tue Mar 23 08:40:07 2010 [VMM][I]: Connecting to uri: qemu:///system
Tue Mar 23 08:40:07 2010 [VMM][I]: error: Failed to create domain from /var/lib/one//0/images/deployment.0
Tue Mar 23 08:40:07 2010 [VMM][I]: error: monitor socket did not show up.: Connection refused
Tue Mar 23 08:40:07 2010 [VMM][I]: ExitCode: 1
Tue Mar 23 08:40:07 2010 [VMM][E]: Error deploying virtual machine: Failed to create

domain from /var/lib/one//0/images/deployment.0
Tue Mar 23 08:40:08 2010 [LCM][I]: Fail to boot VM.
Tue Mar 23 08:40:09 2010 [DiM][I]: New VM state is FAILED
Tue Mar 23 08:41:33 2010 [DiM][I]: New VM state is DONE.

Déploiement réussi :

Tue Mar 23 09:11:31 2010 [DiM][I]: New VM state is ACTIVE.

Tue Mar 23 09:11:32 2010 [LCM][I]: New VM state is PROLOG.

Tue Mar 23 09:11:32 2010 [TM][I]: tm_clone.sh: localhost.localdomain:/tmp/debian.5-0.x86.img griffon-18:/var/lib/one//4/images/disk.0

Tue Mar 23 09:11:32 2010 [TM][I]: tm_clone.sh: DST: /var/lib/one//4/images/disk.0

Tue Mar 23 09:11:32 2010 [TM][I]: tm_clone.sh: Creating directory /var/lib/one//4/images

Tue Mar 23 09:11:32 2010 [TM][I]: tm_clone.sh: Executed "ssh griffon-18 mkdir -p /var/lib/one//4/images".

Tue Mar 23 09:11:32 2010 [TM][I]: tm_clone.sh: Cloning /tmp/debian.5-0.x86.img

Tue Mar 23 09:15:02 2010 [TM][I]: tm_clone.sh: Executed "scp /tmp/debian.5-0.x86.img griffon-18:/var/lib/one//4/images/disk.0".

Tue Mar 23 09:15:02 2010 [TM][I]: tm_clone.sh: Executed "ssh griffon-18 chmod a+w /var/lib/one//4/images/disk.0".

Tue Mar 23 09:15:02 2010 [TM][I]: tm_mkswap.sh: Creating 1024Mb image in /var/lib/one//4/images/disk.1

Tue Mar 23 09:15:02 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 mkdir -p /var/lib/one//4/images".

Tue Mar 23 09:15:03 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 dd if=/dev/zero of=/var/lib/one//4/images/disk.1 bs=1 count=1 seek=1024M".

Tue Mar 23 09:15:03 2010 [TM][I]: tm_mkswap.sh: Initializing swap space

Tue Mar 23 09:15:03 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 /sbin/mkswap /var/lib/one//4/images/disk.1".

Tue Mar 23 09:15:03 2010 [TM][I]: tm_mkswap.sh: Executed "ssh griffon-18 chmod a+w /var/lib/one//4/images/disk.1".

Tue Mar 23 09:15:03 2010 [LCM][I]: New VM state is BOOT

Tue Mar 23 09:15:03 2010 [VMM][I]: Generating deployment file: /var/lib/one/4/deployment.0

Tue Mar 23 09:15:03 2010 [VMM][I]: Command: scp /var/lib/one/4/deployment.0 griffon-18:/var/lib/one//4/images/deployment.0

Tue Mar 23 09:15:03 2010 [VMM][I]: Copy success

Tue Mar 23 09:15:04 2010 [VMM][I]: Connecting to uri: qemu:///system

Tue Mar 23 09:15:04 2010 [VMM][I]: ExitCode: 0

Tue Mar 23 09:15:04 2010 [LCM][I]: New VM state is RUNNING

Tue Mar 23 09:15:31 2010 [VMM][I]: Connecting to uri: qemu:///system

Tue Mar 23 09:15:31 2010 [VMM][I]: ExitCode: 0

Tue Mar 23 09:15:31 2010 [VMM][I]: Monitor Information:

CPU : -1

Memory: 524288

Net_TX: -1

Net_RX: -1