RAPPORT DE PROJET
"Cloud et virtualisation avec Openstack"

Tutrice : BOUZIANE Hinde

Groupe :
CULTY Alexandre,
BENAIS Charles,
BRESSAND Jérémy,
ROGLIANO Théo

Ordre alphabétique
Tutrice après les auteurs

2015 - 2016

Table des matières

Remerciements

Nous tenons tout d'abord à remercier madame BOUZIANE Hinde, notre tutrice de projet, qui nous a permis de réaliser ce projet dans de bonnes conditions. En effet, nous prenions régulièrement des réunions avec madame BOUZIANE pour discuter de notre avancement et lui demander des conseils.

Nous voulons aussi remercier l'équipe ~~de~~ Grid5000 qui nous ~~à~~ permis de réaliser ce projet sur leur s~~ystème e~~~~n nous donnant accès à leurs serveurs grâce à~~ la demande de notre tutrice.

Plateforme distribuée par l'intermédiaire de ...

Pas de saut de ligne

Ceci ~~qui~~ nous a permis de réaliser ce projet dans un environnement plus réaliste et plus adapté à la notion de cloud computing.

Introduction

Définition incomplète. Complétez et ne changez rien quand vous citez une définition existante

> «~~Le cloud computing, c'est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau.~~»
> [Wikipédia]

Le cloud est né à la suite de la multiplication des centres de données, de l'expension de la vitesse des communications et de la vitesse des calculs ainsi que de la maitrise de la virtualisation.

L'intérêt du cloud tien~~s dans~~s le fait que les serveurs sont loués à la demande, et que la facturation soit fait à l'usage. Autrement dit, le client paye uniquement l'utilisation réelle qu'il en a fait, ~~pour les caractéristiques précises qu'il peut modifier à tout moment.~~ Un autre intêret du cloud computing est qu'il permet de s'affranchir des serveurs d'une entreprise ainsi que de leurs entretient, car tous les services sont accessible via Internet et sont interconnectés.

D'où vient cette affirmation ?

~~Pour répondre aux différents besoins des clients~~ on distingue trois grands services de cloud :

Grandes catégories de cloud :

— IaaS ~~(~~ Infrastructure as a Service~~)~~ qui donne accès à des machines virtuelles (que nous appelerons par la suite des VM) sur lesquelles l'utilisateur peut installer son propre système d'exploitation et faire évoluer les caractéristiques du serveur à la demande.

— PaaS : Platform as a Service, l'utilisateur installe ses propres applications.

Parenthèses pareil que pour le premier

— SaaS : Software as a Service, les applications sont directement mis à disposition des utilisateurs, celui-ci ne s'occupe en rien de la gestion du serveur.

PaaS Incomplet ! Sur quoi ? L'utilisateur dispose toujours de machines virtuelle, mais avec un environnement et il y installe des applications

Améliorer la définition de SaaS

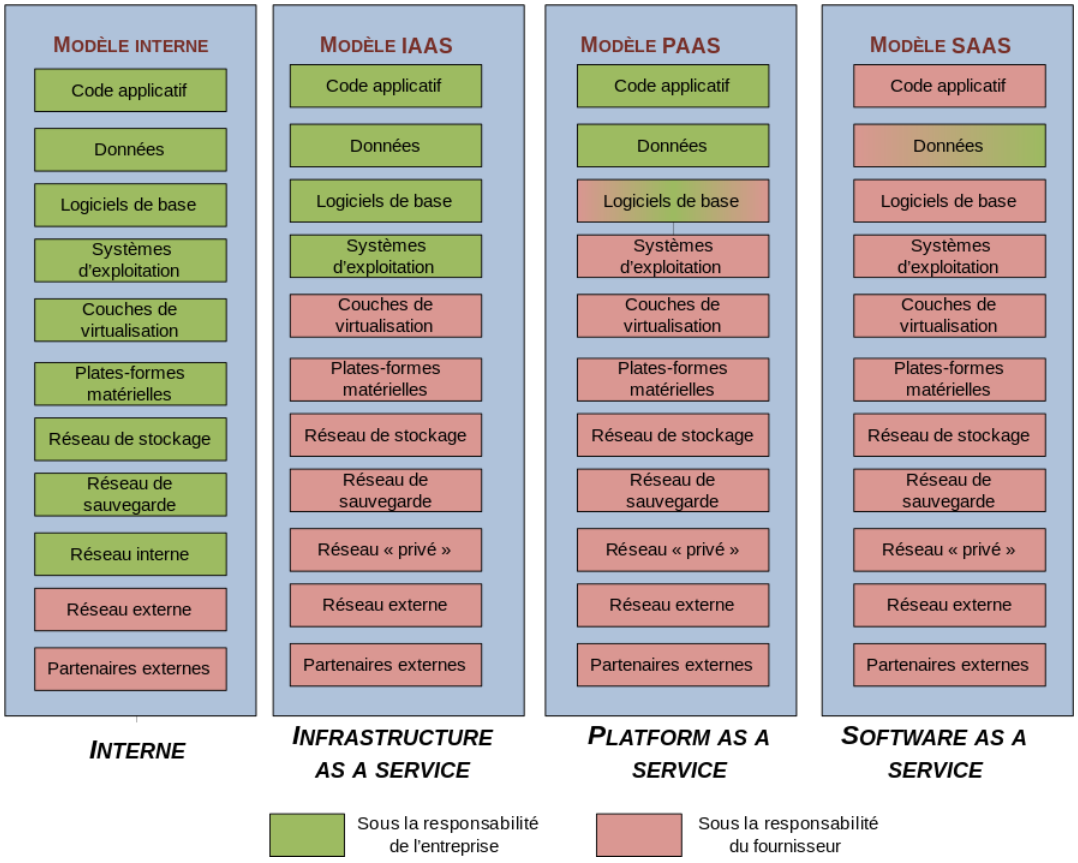

FIGURE 1 – Les responsabilités selon le service cloud [Wikipédia]

Pour ce projet nous avons travaillé avec OpenStack qui permet de déployer des IaaS à l'aide de ses différents composants logiciels que voici :

- Glance : le gestionnaire d'image, il permet la découverte, l'envoi et la distribution d'image disque vers les VM.
- Keystone : le service d'identité, il fournit un annuaire central contenant la liste des services et la liste des utilisateurs ainsi que leurs rôles et autorisations.
- Horizon : l'interface de gestion via navigateur.
- Neutron : permet de gérer et manipuler les réseaux et l'adressage IP.
- Nova : le module que nous avons le plus manipulé, son but est de gérer les ressources de calcul des infrastructures.

Ce paragraphe doit être avant les détails d'openstack

Est où a été

De

L'objectif de ce projet ~~était~~ de prendre en main un environnement cloud de type IaaS à travers OpenStack, d'acquérir des compétences en virtualisation et d'administration de machines virtuelles, ainsi que de mettre en oeuvre un système d'~~automatisation d'installation~~ d'applications sur des machines virtuelles.

Automatisant le

Déploiement et ....

~~Dans la première partie nous présenterons ...~~, puis en seconde partie nous développerons le travail effectué, enfin dans la dernière partie nous exposerons les problèmes rencontrés.

La suite de ce chapitre présente .... Et partie aussi à modifier

Éviter le "nous" (sauf pour des choix et pour pointer un travail fait par vous) et passer à la forme passive ou utiliser la forme : ce chapitre, cette section ou ce rapport présente...

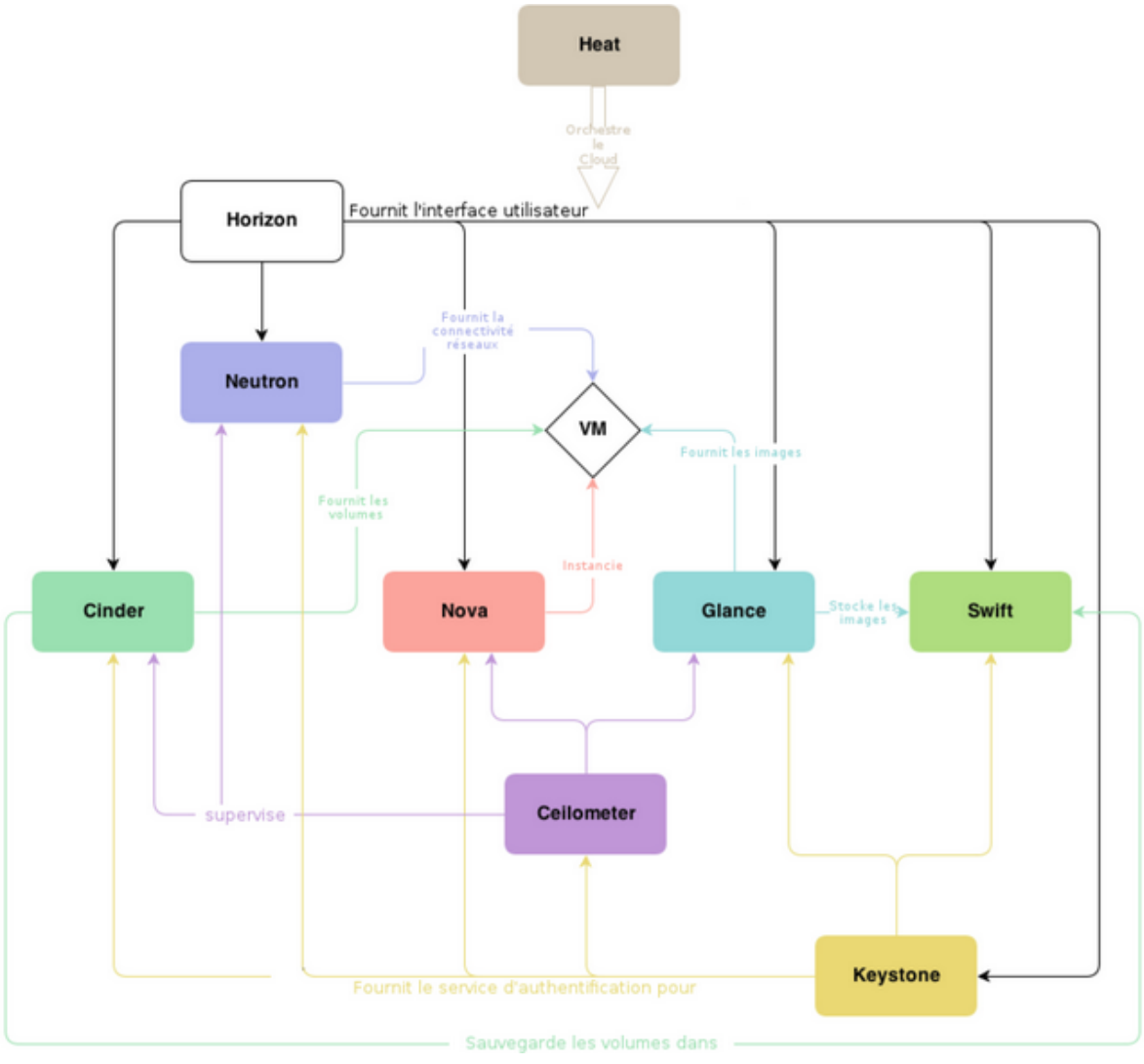

FIGURE 2 – Architecture des Modules

De quoi ? Et référencée où ?

Aussi figure trop technique pour une introduction. Vous pouvez simplifier ou trouver une autre sans oublier de citer la référence

1. Applications

Avant de donner le nom d'un outil, introduire ce que vous voulez présenter et pourquoi . Balancer un nom d'un outil ou d'une technologie sans savoir où on va rendra non seulement le rapport dur à lire mais désagréable. Ne pas oublier que vous vous adressez à des lecteurs qui n'on aucune idée de votre travail, voir ne sont pas du tout du domaine

1.1 XP5K-OpenStack

Puppet est un outil de configuration système fiable et évolutif écrit dans un language déclaratif qui lui est propre (le Puppet Language). Il permet l'automatisation des tâches souvent effectuées par le sys admin. OpenStack-Puppet offre la possibilité de déployer et configurer l'ensemble des composants d'Openstack nécéssaires au sys admin.

XP5K est une librairie écrite en Ruby aidant à la création de scripts d'éxperimentation.

XP5K-Openstack étend XP5K pour deployer un environnement Openstack sur Grid5000 basé sur les modules ci-dessus.

Screen est une amélioration pour l'ergonomie.. Elle est à faire paraître en dernier en justifiant pourquoi son utilisation serait un plus

1.2 Multiprocessing avec screen

Screen est un mutliplexeur de terminal. Il permet d'exécuter des processus en parallèle dans différents terminaux virtuels, qu'on appelle des screens. Il permet aussi d'organiser les terminaux virtuels, de rassembler des screens partageants des affinités dans ce qu'on appelle une session de screen. Le parcours de l'utilisateur entre les différents terminaux est facilité de part la possibilité de déttacher et de rattacher une session de screen, ainsi que par la possibilité de diviser la fenêtre d'affichage afin d'afficher plusieurs terminaux virtuels en même temps.

On peut automatiser une suite d'instructions exécutees par screen en lui passant en paramètre un fichier d'instruction.

2. Travail réalisé

Introduire ce qui va etre présenté. Les objectifs et ensuite le comment

2.1 Mise en place

L'installation d'OpenStack nécessitant des droits sudo, et n'ayant pas la possibilité de les avoir à la faculté, nous avons donc utilisé nos propres machines dans un premier temps.

2.1.1 DevStack

Dans les premières semaines qui ont débuté le projet, nous avons d'abord commencé par utilisé DevStack, un projet qui a pour but de rendre l'installation d'OpenStack plus facile en executant qu'un seul script qui permet de télécharger et configurer tout ce qu'il faut pour utiliser OpenStack. Malheureusement ce projet n'est pas fiable à 100% comme il est indiqué sur leur site et certain on en parti réussi à s'en servir, d'autre non, mais nous avons quand même tous acquis de nouvelles connaissances sur OpenStack et son fonctionnement.

Suite à ces problèmes bloquant, madame BOUZIANE nous à propose~~r~~ d'utiliser OpenStack sur Grid5000.

2.1.2 Grid5000 Ajouter le cite en référence et ajouter un dessin

Grid5000 est un réseau de ressources distribuées supporté par l'INRIA et le CNRS qui permet à des chercheurs en informatique d'effectuer des tests notamment sur le calcul distribué et parallèle, y compris le cloud computing et le big data.

Tout d'abord, nous avons commencé par réalis~~é~~ le tutoriel "Getting Started" qui nous a permis de nous familiariser avec ~~le système de~~ Grid5000, notamment sur les règles d'utilisation, les réservations de jobs et l'architecture générale du réseau. La plateforme

Ensuite, nous avons suivi le tutoriel "OpenStack Deployment", ce qui nous a permis de prendre en main OpenStack plus facilement et de créer nos premières machines virtuelles.

2.2 Nos premiers scripts

En premier lieu, nous avons commencé par faire des petits scripts de tests pour vérifier l'acquisiton de nos nouvelles connaissances. Ces scripts n'ont pas été conservés.

2.2.1 VM_launcher

VM_launcher est le premier script qui a vu le jour. Ce script a pour but d'automatisé la création d'une VM, lui associé une adresse IP publique et ouvrir une série de port pour la communication. C'est en effet la 1ère étape de notre projet.

Pour celà, nous avons crée un script écrit en bash concaténant la commande "nova boot" créant une machine, la commande "nova floating-ip-create/add" créant et associant une IP à la machine nouvellement créé, et enfin la commande "nova secgroup-rule-add" pour dévérouiller des ports. Ce script étant lancé sur le controller via rake. (Script entier en annexe 1)

2.2.2 APP_installer

APP_installer vient dans la continuité de VM_launcher. Après avoir créer une VM nous souhaitons en effet y faire tourner au moins un programme. C'est le role d'APP_installer de mettre en place et lancer le programme.

Afin d'y parvenir, nous récupérons les adresses des VM via le controller, puis pour chaque IP récupéré copions les fichiers nécessaires à l'installation grâce à scp puis installions et lancions le programme comme si nous étions dans un terminal. (Script en annexe 2)

2.2.3 Clean

Le meilleur pour la fin. L'utilité de ce script n'est même pas à démontrer. Il sert tout simplement à tout remetre à zéro. En effet, les erreurs et les test étant fréquent en phase d'écriture, nous avons vite eu besoin de ce script.

Il détruit tous les logs générés par un rm, détruit les VM une à une grâce à "nova delete" et enfin détruit les IP générées par "nova floating-ip-delete".(Script entier en annexe 3)

2.3 Notre programme principal

Avant de lancer le programme, l'utilisateur doit se connecter au point d'accès grid5000 avec ses identifants, se connecter au site de son choix (Grenoble, Rennes ..), transferer l'ensemble de nos scripts sur le frontend, transferer les fichiers des applications qu'il souhaite installer dans le dossier apps et configurer correctement le fichier utilisateur config.json. Ensuite il lui suffit de lancer le script ruby construct.rb.

2.3.1 Automatisation d'un déploiement d'application

Afin de rendre automatique le déploiement d'une application par un client, nous mettons à sa disposition un programme qui permet de générer un fichier JSON. Ce fichier contient une liste de machines virtuelles avec pour chacune d'elles des attributs comme son nom, ses caractéristiques techniques, ou encore le nombre de VM identiques à celle-ci que le client veut. Chaque VM contient aussi une liste des applications à installer avec des attributs comme son nom, son type (client/serveur), le port si c'est un serveur, ou le nom de la VM serveur si c'est un client.

Ce fichier JSON est ensuite télécharge sur Grid5000 pour être analyser par un programme ruby qui permet de parcourir le JSON et d'appeller le script de création d'une VM, puis le script d'installation d'une application sur une VM.

2.3.2 Affichage de nos VM grace a screen : TODO

Une fois que notre programme est fini, l'utilisateur peut alors lancer la commande screen -ls pour voir l'ensemble de ses applications qui sont détachées. Il peut alors rattacher l'application de son choix et naviguer entre les différentes VM faisant parti de l'application en utilisant le raccourci CTRL+a "

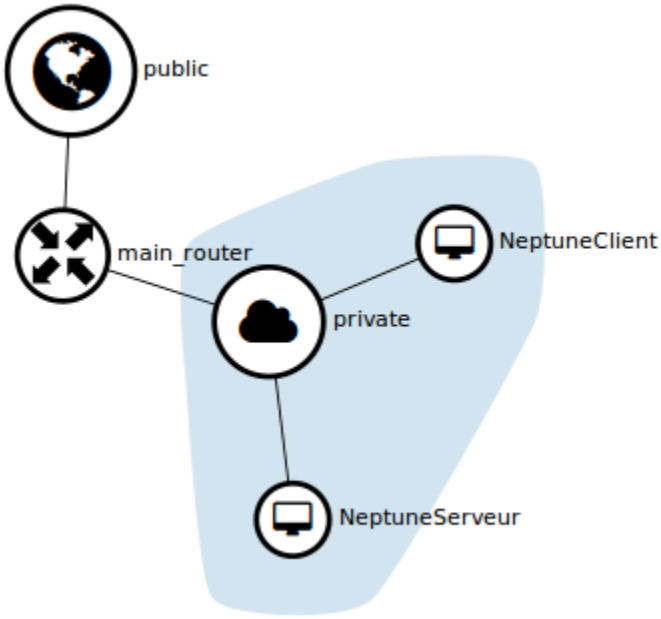

FIGURE 2.1 – Topologie du Network

3. Problèmes rencontrés

Toujours introduire ce que vous allez présenter. Puis cette partie sera dans le dernier chapitre ou en fin du chapitre réalisation. Aussi, les difficultés sont des points à citer et non des sections.

3.1 DevStack

Les premières semaines du TER, nous avons commencé par travailler OpenStack en utilisant DevStack, qui permet d'installer différents modules de base OpenStack sur une unique machine.

3.1.1 Installation et configuration

DevStack requiert une configuration minimale, ainsi que des prérequis de configuration (systeme, network).

Il n'a pas été aisé de mettre en place cette configuration, et, quand bien même une configuration DevStack ait été correctement mise en place et fonctionnelle, celle-ci n'à pour durée de vie qu'une session ! Incomplet : il manque la difficulté en question

3.2 OpenStack sur Grid5000

Chacun rencontrant des difficultés lors du déploiement DevStack Mme. BOUZIANE à pu obtenir un accès au reseau de Grid5000 afin de pouvoir utiliser ~~les scénarios~~ de déploiement d'OpenStack existants.

Si le terme scénario n'a pas été introduit et n'est pas utile, pas besoin de le citer

~~3.2.1 Découverte~~

Ce n'est pas une difficulté

~~Tout d'abord même si celà n'a pas vraiment été un problème mais plutôt une étape nécessaire, nous avons dû comprendre comment le déploiement d'Openstack s'opère et les nouveaux outils mis à notre disposition (comme la commande rake).~~

~~3.2.2 Choix des languages~~

Ce n'est pas une difficulté

~~Voulant écrire des scripts, nous nous sommes naturellement orientés vers le language Bash. Nous nous sommes vite rendu compte que celà n'eut pas été le choix le plus judicieux de par la pénibilité syntaxique et les limitations du language. C'est une des raisons qui nous a poussé à passer progressivement à des scripts en ruby. L'autre raison étant que xp5k est en partie écrit en ruby.~~

Non présenté comme une difficulté.

Possible de le mettre à la fin de la réalisation pour dire qu'avant d'arriver à obtenir les résultats, vous êtes passés par .....

Ou bien dans la discussion avant la réalisation

3.2.3 ~~Récupération de variable et flux~~

Lors de nos scripts nous accédons à de nombreuses machines et, notamment, entrons et sortons de nombreuses fois dans notre "controller". Nous avons d'abord utilis~~er l~~a commande rake (effet de découverte) puis nous nous sommes vite appuyé sur ssh pour pouvoir récupérer des résultats ou des varibles d'environnement via des flux (pipe).

De plus, nous n'avons pu récupérer certaines informations sans devoir les faire afficher dans le terminal et les récupérer par une regexp. Celà est une voie d'amélioration future.

~~3.2.4 Copie de fichier~~

Ce n'est pas une difficulté

~~Une des difficultés pour l'écriture des scripts est le fait qu'ils nous aient impossible de copier un fichier exterieur à grid5000 vers grid5000 en étant au sein de celui ci. Nous sommes donc partis du principe qu'en prérequis le client copie les fichiers de son laptop vers grid5000.~~

3.2.5 IDE

Est ce vraiment une difficulté ?

Conséquence de la difficultée précédente, vu qu'il est peu aisé de copier des fichiers vers grid5000 en permanence, nous utilisons les éditeurs de textes (vim et nano) mis à notre dispostion sur le cluster pour des petites modifications lors des phases de tests. Malheuresement, ceux-ci sont assez peu ergonomique et plutôt contraignant sur la durée.

3.2.6 XP5K-OpenStack

Dire pourquoi vous l'avez fait et si ca à causé des difficultés pour la réalisation, sinon à supprimer

Nous avons regarde le fonctionnement d'XP5K-Openstack et nous avons eu du mal a comprendre vu le nombre de scripts le composant et les dependances entre ces scripts.

3.2.7 Screen

Nous avons eu beaucoup de mal à maitriser cet outil pour plusieurs raisons. Première difficulté comprendre la différence entre un screen et une session de screen qui est aussi appelée screen par abus de langage dans la documentation officielle et les forums. D'autre part il n'existe pas de tutoriel exhaustif pour la creation de fichier d'instruction screen. Notamment pour l'automatisation du lancement d'une commande bash dans un screen nous avons consulté de nombreux sites sans succes.

Conclusion

La fin de ce paragraphe est ce qui doit paraître en premier

Après un début difficile sur DevStack, nous avons réussi à nous intégrer à l'environnement de Grid5000. Petit à petit, nous avons réalisé des scripts réalisant des parties de notre objectif comme le script de création de VM, pour arriver à la fin à un script global qui permet d'automatiser le déploiement d'application sur une ou plusieurs VM.

Ce n'est pas un retard, car sans ca vous n'aurez pas compris OpenStack sur G5k

~~Notre rythme de travail d'environ 8 heures par semaine à travailler en groupe nous à permis de rattraper le retard pris des premieres semaines passés sur DevStack.~~

Ce projet nous à permis d'acquérir de réelles compétences et d'importantes notions dans le domaine du cloud computing et d'OpenStack dans un contexte où le cloud computing semble être omniprésent.

Ressources documentaires

Devstack : `http://docs.openstack.org/developer/devstack/`

Openstack : `http://www.openstack.org/`

Grid5000 : `https://www.grid5000.fr/`

Wikipédia : `https://fr.wikipedia.org/wiki/Cloud_computing/`

Screen `https://www.gnu.org/software/screen/manual/screen.html`

Annexes

```
lastlinelastline
ADR=`rake roles:show | grep 'controller' | grep -o -E '[^: ]*\.grid5000\.fr'`;
echo "*Adresse du controller* > $ADR";

#On s'y connecte pour fetch la liste des IP_VM
IPs=`ssh root@$ADR 'source openstack-openrc.sh && nova floating-ip-list' | cut -d '|' -f 3 | grep -o -E '(([0-9]{1,3}\.){3}[0-9]{1,3})'`;
echo "*IP VMs* > $IPs";

for IP in $IPs; do
echo "VM : $IP";

scp -p -r ../Serveur debian@$IP: ; #/!\ Les applications sont presentes sur le frontend
ssh debian@$IP "sudo apt-get -y update; sudo apt-get -y install gcc make; cd Serveur/chat/; make";
# Serveur/chat ou Serveur/FTP

echo 'Apps installees';

done
```

Listing 3.1 – Installateur d'application : On récupère l'adresse du controlleur

```
lastlinelastline
IPs=`ssh root@$ADR 'source openstack-openrc.sh && nova floating-ip-list' | cut -d '|' -f 3 | grep -o -E '(([0-9]{1,3}\.){3}[0-9]{1,3})'`;
echo "*IP VMs* > $IPs";

for IP in $IPs; do
echo "VM : $IP";

scp -p -r ../Serveur debian@$IP: ; #/!\ Les applications sont presentes sur le frontend
ssh debian@$IP "sudo apt-get -y update; sudo apt-get -y install gcc make; cd Serveur/chat/; make";
# Serveur/chat ou Serveur/FTP

echo 'Apps installees';

done
```

Listing 3.2 – Installateur d'application : On se connecte au controlleur et on récupère les adresses ip des VM

```
lastlinelastline
scp -p -r ../Serveur debian@$IP: ; #/!\ Les applications sont presentes sur le frontend
ssh debian@$IP "sudo apt-get -y update; sudo apt-get -y install gcc make; cd Serveur/chat/; make";
# Serveur/chat ou Serveur/FTP

echo 'Apps installees';
```

```
done
```

Listing 3.3 – Installateur d'application : On se connecte a une VM et on installe l'application

3.2.8 Automatiser creation VM

```
#!/bin/bash

#$>frontend
ERR_ARGS=85

if [ $# -ne 2 ]  # Correct number of arguments passed to script?
then
  echo "Usage: `basename $0` vm_name vm_size"
  exit $ERR_ARGS
fi

case $2 in
    xs | tiny | small | medium | large | xlarge )
        echo "Flavor : $2";;
    * )
 echo "Usage: xs | tiny | small | medium | large | xlarge : $2";
 exit $ERR_ARGS;;
esac


echo '#|--------------------------+';
echo '#|        VM_LAUNCHER       |';
echo '#|--------------------------+';

rake cmd cmd="echo '#### START_RAKE ####';
source openstack-openrc.sh;

echo '#### CREATION VM1 ####';
nova boot --flavor m1.$2 --image 'Debian Jessie 64-bit' --nic net-id=\$(
    neutron net-show -c id -f value private) --key_name demo $1;


echo '#### AJOUTE IP PUBLIQUE ####';
IP_PUB=\`nova floating-ip-create public | grep -o -E '(([0-9]{1,3}\.)
    {3}[0-9]{1,3})'\`;
nova add-floating-ip $1 \$IP_PUB;
echo \$IP_PUB;

echo '#### MODIFIE DROITS ####';
VAR_RULE=\`(nova secgroup-list-rules default | grep -o 10000)\`;

if [ \$VAR_RULE -ne 10000 ]
then
 nova secgroup-add-rule default tcp 10000 10100 0.0.0.0/0;
fi
 " host=controller;


#./APP_installer.sh;
```

Listing 3.4 – VM launcher

3.2.9 Automatiser le deploiement d'openstack sur grid5000

```
login=$1;
site=$2;
```

Listing 3.5 – Deploiement : On recupère le login et le site

```
scp -r -p DOSSIER_APPLICATIONS/ $login@access.grid5000.fr:$site
```

Listing 3.6 – Deploiement : On copie les fichier de l'app vers le frontend

```
ssh $login@access.grid5000.fr
ssh $site
```

Listing 3.7 – Deploiement : On se connecte au site

```
screen rake run
```

Listing 3.8 – Deploiement : On lance l'installation d'openstack