Remette le bon titre : prise en main d'un environnement de Cloud : OpenStack

RAPPORT DE PROJET
"Cloud et virtualisation avec OpenStack"

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

Tutrice : BOUZIANE Hinde

2015 - 2016

Table des matières

: installation sur machines personnelles
Utilisation sur une plateforme distribuée existante
La plateforme Grid5000
OpenStack sur Grid5000 : XP....
Déploiement

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 avions une réunion hebdomadaire avec madame BOUZIANE pour l'informer de notre avancement et lui demander des conseils.

Nous voulons aussi remercier l'équipe Grid5000 qui nous à permis de réaliser ce projet sur leur plateforme distribuée par l'intermédiaire de notre tutrice. ~~Ce qui nous~~ à offert la possibilité de réaliser ce projet dans un environnement plus réaliste et plus adapté à la notion de cloud computing.

Reformuler "Ce qui.. " pas en début de phrase

1. Introduction

Texte cité en italique

Essayez d'éviter cette forme de texte espacé

> «Le Cloud Computing est l'ensemble des disciplines, technologies et modèles commerciaux utilisés pour délivrer des capacités informatique (logiciels, plateformes, matériels) comme un service à la demande.» [Cloud Computing le guide complet, auteurs Sylvain Caicoya et Jean-George Saury, édition Micro Application, 2011]

En faire une référence

Le Cloud (ou en français ,virgule l'informatique dans les nuages) est un mot derrière lequel se cachent de nombreuses choses. De ce fait aborder tous les aspects du Cloud en un seul paragraphe est une chose impossible. Un des services fournis par le Cloud qui nous intéresse en particulier est le SaaS (Software as a Service, en français : applications prêtes à l'emploi).

L'ancienne présentation était incomplète mais dans la bonne direction

> «Il s'agit de la mise à disposition d'un logiciel non pas sous la forme d'un produit que le client installe en interne sur ses serveurs mais en tant qu'application accessible à distance comme un service... L'utilisation est ainsi transparente pour les utilisateurs, qui ne se soucient ni de la plateforme, ni de sa mise à jour, ni de son administration.» [Cloud Computing le guide complet, auteurs Sylvain Caicoya et Jean-George Saury, édition Micro Application, 2011]

Référence

Par exemple Gmail, Dropbox, ou encore ShareLatex sont des Saas.

Dans cette partie, nous analyserons plus en détail les fonctionnalités fournis par un outil permettant de déployer un cloud : OpenStack. Nous présenterons aussi la problématique et les objectifs précis de notre projet, ainsi que notre organisation du projet.

1.1 L'environnement de cloud OpenStack

> «OpenStack est un ensemble de logiciels open source permettant de déployer des infrastructures de cloud computing (infrastructure en tant que service). La technologie possède une architecture modulaire composée de plusieurs projets corrélés (Nova, Swift, Glance...) qui permettent de contrôler les différentes ressources des machines virtuelles telles que la puissance de calcul, le stockage ou encore le réseau inhérents au centre de données sollicité.» [https ://fr.wikipedia.org/wiki/OpenStack]

Vous allez utiliser plus tard le terme contrôleur alors que vous ne avait pas parlé

OpenStack permet de déployer un environnement de cloud de type IaaS (Infrastructure as a Service, en français : infrastructure prête à l'emploi). Un environnement de cloud de type IaaS permet à l'utilisateur d'accéder à des machines physiques et la plupart du temps de créer des machines virtuelles avec les ressources choisies sur lesquelles il pourra installer le système d'exploitation et les applications qu'il souhaite. Par exemple, OVH, Online, FirstHeberg sont des entreprises offrant ce service à travers des offres de location de serveurs virtuels.

OpenStack est à ce jour composé de 17 logiciels que nous ne pouvons vous présenter intégralement. On s'intéressera à un seul composant d'OpenStack : Nova.

Vous pouvez citer des fonctionnalités

«Nova dirige le cycle de vie des instances de calcul dans un environnement OpenStack. Les responsabilités de Nova englobent la création, l'ordonnancement et le démantelement des machines virtuelles sur demande.» [http ://www.openstack.org/software/releases/liberty/components/nova]

Sans améliorer le texte, Il n'a pas été dit que OpenStack offrait Linux ....

~~Sur les systèmes linux, Nova utilise la plateforme de virtualisation KVM intégrée au noyau et permettant la simulation de plusieurs systèmes d'exploitation (ou machines virtuelles) sur une même machine physique. Nova est la brique élémentaire d'OpenStack, il est suffisant pour créer un environnement de cloud~~. En effet il permet de créer un ensemble de machines virtuelles connectées au réseau publique de la machine hôte et installer un système d'exploitation ainsi que des serveurs ssh sur ces machines virtuelles. Par la suite l'utilisateur peut donc se connecter à ces machines et installer les applications qu'il souhaite, et ainsi exécuter ~~ses~~ applications à distance.

La seconde partie du paragraphe est moins technique et est plus claire et suffisante

Des

1.2 Problématique et objectifs du projet

Les objectifs de ce projet sont :

- ~~«~~la prise en main ~~d'un environnement de Cloud IaaS~~ : OpenStack https ://www.openstack.org ; permettant d'acquérir des compétences en virtualisation, administration de plusieurs machines virtuelles, installation et exécution d'applications sur ces machines.
- mettre en oeuvre un système d'automatisation de l'installation d'applications (un ensemble de programmes) sur des machines virtuelles et de leur exécution.~~»~~

[~~Sujet TER L3 : Prise en main d'un environnement de Cloud~~]

~~A ces objectifs ci-dessus, nous pouvons associer la problématique :~~

Comment utiliser OpenStack pour déployer un ensemble d'applications p~~rêtes à l'emploi~~ ?

Contenu à réorganiser et à reformuler avec vos propres phrases

Cette phrase est ç qui devrait paraître en premier et ensuite décomposer en sous objectifs

1.3 Organisation du projet

Ce texte est bien adapté pour la partie objectifs

Afin de mieux déterminer les tâches à effectuer, nous avons réalisé un diagramme de Gantt (~~disponible en annexe figure 6.1 et 6.2~~).

~~On~~ peut voir que nous avons découpé le projet en plusieurs étapes qui sont :

- Le déploiement et la prise en main d'OpenStack. Cette étape consiste à installer OpenStack sur une plateforme quelconque, à comprendre les différents outils q~~ue nous procure OpenStack~~, à créer et configurer un ensemble de machines virtuelles ~~avec OpenStack~~.
- Comprendre la procédure d'installation et d'exécution d'une application sur une machine virtuelle OpenStack. ~~Cette étape consiste à installer et exécuter une application quelconque sur une machine virtuelle~~ (~~par~~ exemple une application client-serveur).
- Automatiser le déploiement d'un ensemble de machines virtuelles et l'installation et l'exécution d'un ensemble d'applications d~~essus.~~

Ce sont exactement les objectifs ! En plus clair

Offerts

Une ou plusieurs

Répétition inutile

Sur ces machines

Tout au long du projet nous nous sommes réunis en moyenne deux après-midi par semaine pour travailler le projet. Nous avions aussi une réunion hebdomadaire avec notre tutrice de projet afin de l'informer de nos avancements et de lui demander des conseils.

~~À la suite de ce chapitre introductif~~, nous présenterons la prise en main de l'environnement OpenStack à travers nos premières expériences. Ensuite, nous développerons le travail effectué notamment sur la conception de l'outil permettant le déploiement d'applications sur des machines virtuelles. Enfin nous terminerons par un bilan sur ce projet de cloud computing à travers OpenStack.

Le reste du rapport est organisé comme suit : le chapitre 2 ....., ...

2. Prise en main ~~de l'environnement~~ d'OpenStack

Mettre en place, créer ? Et administrer ... : via une interface graphique ou une API (détailler l'acronyme).

L'interface graphique ..... Appelée dashboard

L'API .....

Vous avez utilisé ...

Il existe deux solutions pour ~~gérer so~~n cloud avec Openstack. La première est l'utilisation du dashboard (module horizon) (en français tableau de bord) qui propose une interface utilisateur basée sur des pages web pour gérer les differents services (Nova, Neutron, ...). La deuxième est de passer par une interface de programmation applicative (souvent désignée par le terme API pour Application Programming Interface). C'est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Afin d'atteindre notre but d'automatisation, nous avons choisi d'utiliser l'API offrant plus d'outils pour y parvenir. Toutefois nous avons aussi utilisé l'interface graphique pour ..... Un aperçu de cette interface se trouve en annexe ...

~~Il est à noté que le dashboard horizon fournit une vue d'ensemble de notre cloud. Il est disponible en annexe page 20 figure 6.8.~~

2.1 Première expérience / Installation d'OpenStack sur nos machines

Titre à améliorer (voir notes plan)

Introduire le "quoi faire" avant le "avec quoi".

~~«DevStack est un ensemble de scripts et d'outils pour déployer rapidement un cloud OpenStack.~~

~~Les objectifs :~~

Inutile, la suite est bien et claire

- ~~Construire rapidement des environnements de développement OpenStack sur un environnement compatible tel que Ubuntu ou Fedora.~~
- ~~...~~
- ~~Permettre aux développeurs de se lancer dans OpenStack pour qu'ils puissent contribuer efficacement sans avoir à maitriser OpenStack dans son ensemble.»~~

~~[https ://github.com/openstack-dev/devstack/README.md]~~

Pour déployer OpenStack sur nos machines, nous avons utilisé DevStack, ~~qui est maintenu par une communauté de développeurs OpenStack sous la forme d'un projet github~~. DevStack ~~peut~~ installer OpenStack de différentes manières (liste non exhaustive) : Mettre le lien github ou ne rien dire

Permet d'installer

- Sur une machine ou un réseau de machines (~~des machines qu'on appelle plus communément des nœuds~~). Appelées noeuds
- Sur une machine virtuelle ou sur une machine physique (~~bare metal , en français : système à nu).~~ Répétition avec premier point ?
- Avec les modules d'OpenStack installés par défaut ou choisis.
- Avec différentes configurations réseau. Deux derniers points pas clairs. Est ce que vous voulez dire installation par défaut ou personnalisée (exemple : configuration réseau) ?

Nous avons choisi ..... La raison est ....

Déconseillé

Machine physique

Machines physiques

Et alors ? Quel est le problème pour vous ?

Il est ~~conseillé de ne pa~~s utiliser DevStack sur l~~es systèmes à nu~~ mais sur des machines virtuelles car DevStack modifie la configuration réseau et la configuration des droits d'accès aux fichiers sur ~~le système hôte~~. Nos objectifs avec DevStack étaient de déployer un environnement de cloud OpenStack sur nos machines de manière individuelle dans un premier temps puis de déployer OpenStack sur un réseau comprenant l'ensemble de nos machines.

Faisant face à divers difficultés lors de l'installation pour certains et lors du déploiement pour d'autres, nous avons décidé avec notre tutrice d'arrêté le déploiement de DevStack sur nos machines. À la place nous nous sommes tourné vers la possibilité de déployer OpenStack sur un réseau de machines distantes.

L'utilisation d'un environnement OpenStack dédié à une plateforme distribuée existante nommée Grid5000 et présentée dans la section suivante

2.2 Déploiement d'OpenStack sur un réseau distant

Améliorer avec suggestion dans plan

2.2.1 Présentation grid5000

Inutile. La description d'après en ajoutant le lien est suffisante.

«Grid'5000 est un environnement de test versatile et à grande échelle pour mener des experiences de recherches dans tous les domaines de l'informatique, avec un soin particulier pour l'informatique parrallèle et distribué comme le "Cloud", le calcul à haute performance ou encore le "Big Data".» [https ://www.grid5000.fr]

La plate-forme Grid'5000 est une grille informatique, c'est-à-dire une infrastructure partagée sur différents sites géographiques. Grid'5000 est constitué de dix sites répartis principalement en France (Grenoble, Lyon, Toulouse, Bordeaux, Lille, Rennes, Reims, Nantes, Nancy, Luxembourg).

Répétition.
Différents sites ... : Grenoble .....

Ne pas oublier : initiative de l'INRIA

Le Grid'5000 (ou G5K) est donc une plate-forme nationale répartie sur différents sites. Elle dispose de 1171 nœuds uniquement physiques. La totalité des nœuds représentent 2218 processeurs, soit pratiquement 8000 cœurs. Les sites géographiques sont reliés par une fibre optique 10 Gbits/s faisant partie du réseau RENATER qui relie les différentes universités et centres de recherche français.

Remettre la figure ici et dans le texte (voir Figure ...)

Nous pouvons voir l'architecture réseau de Grid'5000 en annexe sur la figure 6.3, page 15.

Toute simplement : le point d'entrée de Grid5000 est access.grid5000.fr et se fait via ssh
pas besoin de parler du sud et nord', etc,

Afin d'accéder à l'infrastructure G5K, il faut passer par un des deux accès principaux : access-north ou access-south. Une fois la connexion SSH établie, il est possible d'accéder aux différents sites. La connexion se fait obligatoirement via une authentification par clé SSH. À partir de chaque site, il est possible d'y effectuer des réservations de noeuds sur celui-ci. Les différents sites proposent plusieurs clusters de machines homogènes.

Détail inutile. Les plus curieux iront voir sur le site

2.2.2 Présentation XP5K-OpenStack

Adapter le titre à la suggestion dans la plan

«XP5K est une bibliothèque légère écrite en Ruby permettant d'aider les utilisateurs de Grid'5000 à préparer leur expérimentations en faisant appel à l'interface de programmation Grid'5000 pour :

Consulter le statut

— Faire des demandes de réservation, avoir le statut des réservations, supprimer une réservation...
— Créer des rôles (chaque nœud est dédié à un rôle, par exemple : serveur puppet, contrôleur, unités de calcul), sans avoir à connaître le nom des nœuds qu'on utilise.

Utilisés

— Faire un ou plusieurs déploiement sur des nœuds spécifiés par une réservation ou un rôle.

...»
[https ://github.com/pmorillon/xp5k-openstack/blob/master/README.md]

«Cette méthode d'installation d'OpenStack sur Grid'5000 offre à l'utilisateur des outils courants pour :
— Reserver des ressources et manipuler des réservations.
— Déployer des systèmes d'exploitation sur les noeuds...

Inutile et répétition

— Installer un serveur Puppet.
— Manipuler des modules Puppet maintenus par Openstack.»
[https ://github.com/pmorillon/xp5k-openstack/blob/master/README.md]

XP5K-OpenStack est une interface de programmation conçue pour déployer OpenStack sur Grid'5000. XP5K-OpenStack utilise d'une part l'interface de programmation XP5K, permettant de manipuler les réservations et les nœuds sur Grid'5000, d'autre part l'outil Puppet configuré avec les modules Puppet d'OpenStack qui à partir d'un nœud central (le maitre ou Puppet master) permet le déploiement d'OpenStack sur un ensemble de nœuds esclaves. De calcul plutôt qu'esclave ?

Noeud maitre (dit puppet master)

Terminer avec une transition : un mot sur son utilisation dans le projet

2.2.3 Déploiement et exécution d'applications sur XP5K-OpenStack

Maintenant que nous connaissons les bases sur XP5K-OpenStack, nous pouvons commencer à réfléchir aux moyens d'installer et d'éxécuter une application sur un cloud, cela permettra de lister les étapes nécessaires et identifier lesquels nous pourront automatiser par la suite.

On le sais donc' tout simplement, une fois connecté sur grid5000 et XP5k.. Importé sur la grille ..

Vu que nous travaillons dessus, il est en premier lieu nécessaire de se connecter à Grid'5000 par un des deux accès puis sur un site (nous ne traitons pas du multi-site). Ensuite si cela n'est pas déjà le cas, il faut importer XP5K-OpenStack pour pouvoir en faire usage, en effet, il n'est pas directement fournit dans la grille. Si XP5K est déjà présent, il faut l'éxécuter afin que celui ci nous mettent en place Openstack et donc notre cloud (cette étape est plutôt longue, environ une trentaine de minutes à être réalisé). Simplement : durée de cette étape ~30 minutes

Contradiction avec phrase précédente

À partir de maintenant, nous avons tous les outils pour créer/gérer le cloud. Il nous faut créer et paramétrer les machines virtuelles sur lesquels nous allons déployer nos applications. Cela fait, il ne nous manque plus qu'à importer les fichiers nécessaires à l'installation ou directement l'éxécutable sur les machines virtuelles précédemment créées. Il est à noter que nous utiliseront des images Unix sur nos machines virtuels, les applications doivent donc être portable ou du moins compatible. La dernière étape étant simplement d'installer et d'éxécuter la ou les applications.

Reformuler sans le nous

des

Phrase incomplète

Déjà dit ou sinon vous utilisez les mêmes termes pour dir deux choses différentes ? Ça ne peut pas marcher

Et ???? L'exécution consiste à .... Ssh sur une machine virtuelle et exécution de la commande lançant l'application ....

2.3 Discussion : quelles sont les parties qu'il faut automatiser ?

Bien qu'il soit techniquement possible de tout automatiser, certaines parties sont plus avantageuses que d'autres. Tout ce qui est lié à Grid'5000 étant plus spécifique à notre situation, l'automatisation de cette partie semble moins intéressante et moins au coeur du projet. Par contre, rendre le paramétrage et la création des machines virtuelles ainsi que l'installation et l'éxécution d'application automatique est l'essentiel de notre projet. Nous nous sommes donc concentrés à réaliser cette tâche.

Finir avec une phrase de transition vers le chapitre suivant

3. Un outil pour automatiser le déloiement et l'exécution d'applications sur OpenStack

Pour la réalisation d'un outil automatisant ....., nous avons choisi une implémentation basée sur des scripts ....

~~Afin de réaliser cette tâche~~, nous avons décidé de créer des "scripts" écrits en Bash et en Ruby. Un script est un code interprété qui à la manière d'un script au théatre édicte une sucession d'actions à réaliser. Les scripts ~~nou~~s servent à exécuter et coordonner les comamndes de Nova automatiquement. Dans le cadre de ce projet, les scripts servent à ..... (Mais pas que nova et ici ça mérite de rappeler le rôle de nova)

Dans un premier temps nous verrons l'aspect algorithmique puis ensuite l'aspect technique des scripts développés.

3.1 Description générale

La première étape du processus d'automatisation est la création d'un fichier de configuration structuré, contenant les informations sur les machines virtuelles ainsi que sur les applications devant y être installés~~.~~ Ce fichier est l'entrée d'un script .... (Sans saut de ligne)

~~Une fois que ce fichier de configuration est créé, alors on démarre l~~e script qui analyse ce fichier ~~et organise les informations en tableau~~. Ce tableau est constitué de plusieurs niveaux, le premier représentant les machines virtuelles et le second niveau représentant les applications liées à une machine virtuelle.

Répétition inutile

~~Le script parcourt ensuite le tableau et pour chaque machine virtuelle il commence par appeler le script de création d'une machine virtuelle, puis pour chaque application il appelle le script d'installation et d'exécution d'une application.~~

L'inverse !

Utiliser les références latex

Vous trouverez en annexe ~~page 18, figure 6.6 l'algorithme du script principal, page 16, figure 6.4~~ l'algorithme du script de création d'une machine virtuelle~~, et page 17, figure 6.5~~ l'algorithme d'installation et d'exécution d'une application sur un machine virtuelle.

3.2 Technologies utilisées

Que des technologies que vous connaissez déjà ? C'est léger
Vous pouvez meme supprimer cette section et la grouper avec l'implémentation

Pour réaliser notre projet nous avons dû prendre en main différents outils de base :

— SSH : pour ~~naviguer de notre machine personnelle ver~~s les machines du réseau Grid'5000 et entre les différentes machines virtuelles.

sur grid5000 et lancer des commandes à distance

Inutile. Libérer de la place pour l'essentiel

~~«SSH signifie Secure SHell. C'est un protocole qui permet de faire des connexions sécurisées (i.e. chiffrées) entre un serveur et un client SSH.» [http ://formation-debian.via.ecp.fr/ssh.html]~~

— SCP : ~~pour transférer nos fichiers de notre machine personnelle à notre réportoire sur un site géographique du réseau Grid'5000, et par la suite transférer ces fichiers sur nos machines virtuelles.~~

Pour le transfert sécurisé de fichier sur machines distantes

Détail acronyme toujours apres sa première utilisation, pas plus loin

«Secure copy (SCP) désigne un transfert sécurisé de fichiers entre deux ordinateurs utilisant le protocole de communication SSH» [https ://fr.wikipedia.org/wiki/Secure_copy]

— JSON : pour permettre à l'utilisateur de spécifier l'architecture qu'il souhaite mettre en place avec la configuration des applications et des machines virtuelles, le tout dans une syntaxe simple d'utilisation, et pouvoir récupérer ces informations dans des variables de manière structurée sous forme de tableau et de dictionnaires de données.

«JSON (JavaScript Object Notation – Notation Objet issue de JavaScript) est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou générable par des machines.» [http ://www.json.org/json-fr.html]

3.3 Implémentation

Nous avons développé .... Scripts : enumerate ou itemize nom du script puis rôle

VM_launcher est le premier script que nous avons développé. Il a pour but d'automatiser la création d'une VM, lui associer une adresse IP publique et ouvrir une plage de port pour la communication réseau. Pour cela, nous avons créé un script écrit en Bash exécutant la commande "nova boot" pour créé une machine virtuelle, 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 ouvrir des ports. Pour ?

Sur laquelle un utilisateur souhaite déployer une application

APP_installer vient dans la continuité de VM_launcher. Après avoir créer une VM nous souhaitons en effet y déployer au moins un programme. C'est le rôle d'APP_installer il installe et démarre un programme. Afin d'y parvenir, nous récupérons les adresses des VM via le controller, puis pour chaque IP récupéré nous copions les fichiers nécessaires à l'installation grâce à la commande SCP, puis installons et démarrons le programme comme si nous étions dans un terminal.

En particulier, il récupère ....

Application à la place de programme,

Comme si .... phrase inutile

Enfin pour permettre le déploiement automatique, nous avons créé le script 'construc' en ruby. C'est le script qui parcourt le fichier de configuration JSON, et appelle VMSetup et appSetup, les scripts légérement modifié de VM_launcher et APP_installer qui ont été adapté pour le déploiement automatique.

Suivre la même forme qu'avant

Pourquoi ne pas avoir mis VMsetup directement .... ?

3.4 Expérimentation

Outre la vérification ou tests de bon fonctionnement de notre outil, ... Nous avons ....

Lors de l'éxécution de nos scripts, il est nécessaire de vérifier si chaque étape s'est déroulée correctement. Les plus importantes étant, dans un premier temps, la création des machines virtuelles et celle des adresses IP ainsi que l'association de ces dernières aux machines, et dans un second temps, l'installation et l'éxécution des programmes.

Pour les premières étapes à vérifier, Nova nous fournit deux commandes. La première : nova floating-ip-list, permet de voir la liste des IP créées. La seconde : nova list, permet de voir les machines virtuelles créées ainsi que leurs caractéristiques, notamment si une IP a été liée à une machine.

Ne jamais supposer ce genre de choses !! Vous avez bien tester l'exécution d'applications

Pour les autres étapes, nous supposons que les programmes s'installent bien ou qu'ils aient eux-mêmes prévu ce qu'ils doivent faire en cas d'échec (désintallation, nouvelle essai, etc...). Il est tout de même possible de vérifier qu'un programme est présent sur une VM en se connectant à celle-ci et en regardant par soi-même.

Si vous avez implémenté cette fonctionnalité, il faut le dire dès le départ (dans l'algo) c'est un plus

Une application

Pour vérifier l'éxécution ~~d'un programme,~~ il y a principalement deux façons de faire, la plus simple étant de récupérer un résultat retourné ou un fichier de log. L'autre façon étant de se connecter à la machine virtuelle et de vérifier les processus actifs (par exemple : ps -ef | grep le nomDuProg).

Une fonctionnalité de votre outil. Pas dans la partie expérimentations

3.5 Améliorations

Possibles

Il est possible d'utiliser screen

Pour ~~naviguer~~ d'une machine virtuelle à l'autre~~, l'utilisateur doit savoir utiliser les commandes SSH. Au lieu de cela, on pourrait faciliter la navigation entre les machines virtuelles~~ en utilisant le programme Screen.

> «Screen (GNU Screen) est un « multiplexeur de terminaux » permettant d'ouvrir plusieurs terminaux dans une même console, de passer de l'un à l'autre et de les récupérer plus tard.» [https ://doc.ubuntu-fr.org/screen]

Screen permet d'afficher plusieurs terminaux virtuels sur une même fenêtre, comme on peut le voir sur l'image en annexe page 19 figure 6.7. L'idée serait d'utiliser cette fonctionnalité pour afficher sur le même écran différents terminaux ~~virtuel~~s correspondant chacun a une machine virtuelle sur laquelle s'exécute l'application. Screen permet l'association de plusieurs terminaux virtuels dans des sessions, et la possibilité de détacher ou de rattacher une session. L'idée serait d'utiliser cette fonctionnalité pour permettre à l'utilisateur de naviguer entre les applications. En étant attaché a qu'une seule session l'utilisateur n'a accès qu'aux machines virtuelles sur lesquelles s'exécute l'application et en passant sur une autre session il n'aura accès qu'aux machibes virtuelles de cette application. Enfin Screen est configurable avec des raccourcis pour exécuter toutes les tâches spécifiés précédemment. On peut donc facilement créer un ensemble de raccourcis pour faciliter l'utilisation de notre p~~rogramme.~~

Trop long
A résumer

Outil

Pour configurer le déploiement de ~~ses~~ applications avec notre pr~~ogramme~~, l'utilisateur doit manipuler un fichier de configuration au format JSON. ~~L'architecture de notre fichier de configuration~~ JSON ne permet par l'installation d'applications complexes qui prendraient par exemple plusieurs paramètres ou qui auraient ~~par exemple~~ un ensemble de machines virtuelles qui communiqueraient entre elles de manière plus complexe que le cas d'une application client serveur. ~~Nous pourrions améliorer l'architecture de notre fichier JSON pour prendre en compte ces difficultés cependant notre fichier JSON deviendrait plus complexe et moins facile à utiliser.~~ Une solution pour configurer de manière plus simple le déploiement d'application serait d'utiliser le logiciel Puppet.

Ce format ne permet pas facilement de décrire ....

Cas

Répétition à supprimer

> ~~« C'est un outil permettant de définir et de mettre en œuvre des configurations. ... Il faut installer et configurer sur chaque machine que l'on veut « puppetiser » un client, qui se réfèrera au puppetmaster, machine sur laquelle seront stockés les manifests, qui contiennent les spécifications » [http ://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-112/Les-sysadmins-jouent-a-la-poupee]~~

A supprimer

Puppet permet de configurer et de déployer des applications complexes sur des machines virtuelles en les 'puppetisant', ~~les fichiers de configuration seront présents sur un seul noeud appelé le puppetmaster. Puppet permet la réutilisation de configurations grâce à un système de classe que l'on peut affecter aux machines. La machine bénificiera alors de la configuration de la classe. De plus il gère la diversité des configuration~~s grâce à des templates s'adaptant à la configuration de la machine. Mais Puppet reste difficile d'utilisation pour un utilisateur. Cependant Puppet offre la possibilité de déplacer les informations conçernant la configuration des machines dans une base de donnée. L'idée serait donc de créer une application graphique facile d'utilisation pour l'utilisateur et qui modifiera une base de données dans laquelle Puppet ira chercher les informations pour configurer les noeuds.

Supprimer.

Reformuler et faire simple. Ça veut dire quoi puppetisant

4. Conclusion

Rappel de l'objectif, ou dire, dans le cadre de notre projet, nous avons développé .....

Après quelques difficultés ....

Après quelques difficultés liés à l'utilisation de DevStack, nous avons réussi ~~à nous intégrer~~ à l'~~environnement de~~ Grid'5000. Petit à petit, nous avons réalisé des scripts réalisant des parties de notre objectif, pour arriver à la fin à un script global qui permet d'automatiser le déploiement d~~e~~ une ou plusieurs applications sur une ou plusieurs machines virtuelles.

Plateforme

Les premiers scripts que nous avions développé étant en Bash, et l'environnement d'OpenStack étant en Python, nous avons décidé de les adaptés pour avoir plus de cohérence.
Python étant un langage plus évolué que Bash il permet aussi l'utilisation de bibliothèque liées à OpenStack, ce qui rend aussi son utilisation plus facile. Le script ruby 'construct' qui permet de parcourir le fichier de configuration JSON est également à refaire en Python.
Ces nouveaux scripts Python sont actuellement en développement.

L'utilisation de Grid'5000 nous a permis d'acquérir d'importantes notions sur les infrastructures informatiques ~~à grandes échelles.~~ Distribuee (vous avez pas abordé un problème à grande échelle)

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

5. Ressources documentaires

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

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

Nova https://github.com/openstack/nova

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

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

xp5k-openstack https://github.com/pmorillon/xp5k-openstack

xp5k https://github.com/pmorillon/xp5k

SSH http://formation-debian.via.ecp.fr/ssh.html

SCP http://cc.in2p3.fr/docenligne/134/fr

JSON http://www.json.org/json-fr.html

Puppet http://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-112/Les-sysadmins-jouent-

Screen https://www.gnu.org/software/screen/manual/screen.html [1]

1. LaTeX

6. Annexes

Où est le code source ? Pensez à le commenter.

Le contenu de l'annexe est à mettre dans le meme ordre que les citations

Untitled Gantt Project 10 avr. 2016

Tâches 2

Nom	Date de début	Date de fin
Installation + Prise en main OpenStack	18/01/16	23/02/16
Doc openstack/devstack	18/01/16	10/02/16
Deploiement single machine	18/01/16	10/02/16
Getting started grid5000	11/02/16	16/02/16
TP XP5K-openStack Grid5000	16/02/16	23/02/16
Procedure installation execution application	02/03/16	08/03/16
Mise en place application client serveur	02/03/16	08/03/16
Automatisation	09/03/16	29/03/16
Script déploiement application VM	09/03/16	15/03/16
Script création VM	09/03/16	15/03/16
Creation script nettoyage	16/03/16	22/03/16
Creation application serveur FTP	16/03/16	22/03/16
Automatisation déploiement application	16/03/16	22/03/16
Conception fichier utilisateur JSON	23/03/16	29/03/16
Diagramme de GANTT *Elaboration d'un diagramme de GANTT*	11/02/16	16/02/16
Redaction rapport	11/02/16	17/04/16
Plan du rapport	11/02/16	16/02/16
Redaction 1ere ébauche de rapport	17/02/16	05/04/16
Redaction 2eme ebauche de rapport	06/04/16	12/04/16
Finaliser rapport	13/04/16	17/04/16

FIGURE 6.1 – Liste des tâches de notre diagramme de gantt

Choisir l'un Des deux diagrammes

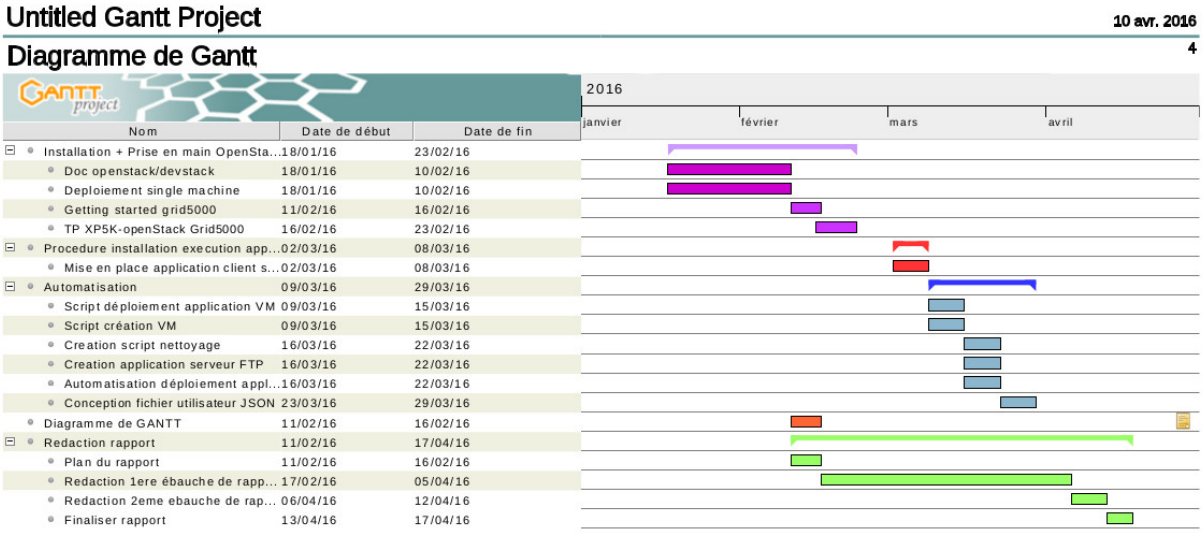
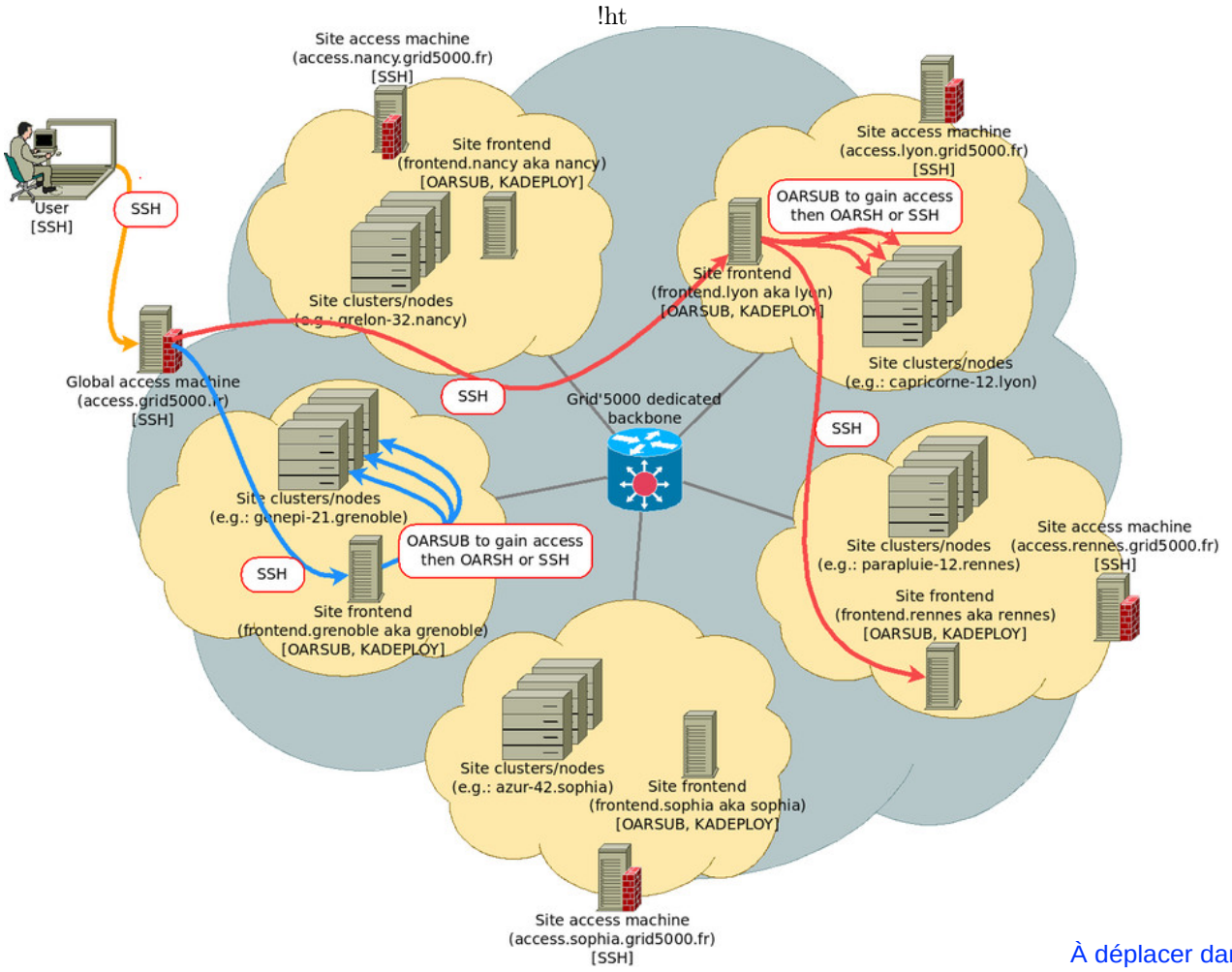


FIGURE 6.2 – L'arbre des tâches de notre diagramme de gantt

!ht



À déplacer dans le rapport.

FIGURE 6.3 – Architecture de Grid'5000

source : www.grid5000.fr/mediawiki/index.php/Getting_Started

Sur la meme ligne que le titre

```
Algorithme :  creationVM
Entrée : - tailleVM : taille de la machine virtuelle è créer
         - nomVM : nom de la machine virtuelle à créer
Localisation : les commandes sont exécutées au niveau du controlleur,
là où OpenStack est installé et où on peut faire appel aux commandes
Nova.

Debut :

      Verification tailleVM valide.

      Vérification nomVM valide ( nom n'étant pas déjà utilisé par
une autre machine virtuelle )

      Ajout de la clé de chiffrement dans la configuration Nova.

      Appel de la commande Nova pour créer la machine virtuelle.

      Appel de la commande Nova pour créer une IP publique.

      Appel de la commande Nova pour récupérer l'adresse IP créée.

      Appel de la commande Nova pour attribuer l'adresse IP à la machine virtuelle.

      Connection à la machine virtuelle.

      Mise à jour des briques logicielles de base.

      Fin de la connection

Fin algorithme.
```

Simplifier et mettre les algos dans le texte principal

FIGURE 6.4 – Algorithme de création d'une machine virtuelle

```
Algorithme : Déploiement et exécution d'une application
             sur une machine virtuelle.

Données : - nomVM : nom d'une machine virtuelle
                    déjà existante.
  - nomAPP : nom de l'application à installer.
  - typeAPP : type d'application.
                     L'appliction est-elle un client ?
                     un serveur ? Ou ni l'un ni l'autre ?
  - pathAPP : chemin vers le repertoire contenant
                     les fichiers nécéssaires au
                     fonctionnement de l'application.
  - portAPP : port sur lequel l'application va écouter.
  - serveurAPP : nom de la VM jouant le rôle de serveur
de l'application dans le cas une application cliente, dans le cas d'une
application de type serveur ou autre,l'argument serveurAPP n'apparait pas
dans l'algorithme.

Localisation : les commandes sont exécutées au niveau du controlleur,
               là où OpenStack est installé et où on peut faire appel
               aux commandes Nova.

Debut :
       Vérification de la validité des arguments.
       On récupère l'IP de la machine virtuelle sur laquelle
          on va installer l'appliction en faisant appel à
          la commande Nova
       Si l'application est de type cliente alors
          On récupère l'IP du serveur en faisant appel à Nova.
       Copie des fichiers présents dans le répertoire
         à l'adresse pathAPP vers la machine virtuelle sur
         laquelle on va installer l'application.
       On se connecte à la machine virtuelle sur laquelle
          on va installer l'application.
       On lance l'installation de l'application.
       Si l'application est de type cliente :
          On execute l'application à laquelle
          on donne en argument l'adresse IP du serveur.
       Sinon
          On exécute l'application sans argument.
Fin algorithme
```

Forme du texte à améliorer

Enlever tous les on. Et citer des actions : récupérer ....

FIGURE 6.5 – Algorithme d'installation d'une application sur une machine virtuelle

```
Algorithme : Création de machines virtuelles,
             installation et exécution d'applications sur
             ces machines virtuelles
Données : - tabVM : un tableau de dictionnaires repésentant
                    les machines virtuelles.
            Chaque machine virtuelle est représentée par :
              - tailleVM : une taille de machine virtuelle.
              - nomVm : un nom de machine virtuelle.
              - APPS : un tableau de dictionnaires représentant
                       les applications à installer sur la
                       machine virtuelle.

Debut :

Parcours de tabVM : pour chaque machine virtuelle faire
        faire appel au script de création de machine virtuelle avec
        pour arguments nomVM et tailleVM.

        Parcours de APPS : pour chaque application faire

            faire appel au script d'installation et d'execution
            d'application sur une machine virtuelle avec pour
            arguments le nom de la machine virtuelle et les
            attributs de l'application.

Fin algorithme
```

En donnant des noms aux scripts, ça sera plus simple

Exécuter ...

FIGURE 6.6 – Algorithme création de machines virtuelles, intallation et exécution d'applications sur ces machines virtuelles

Vérifier la lisibilité après impression

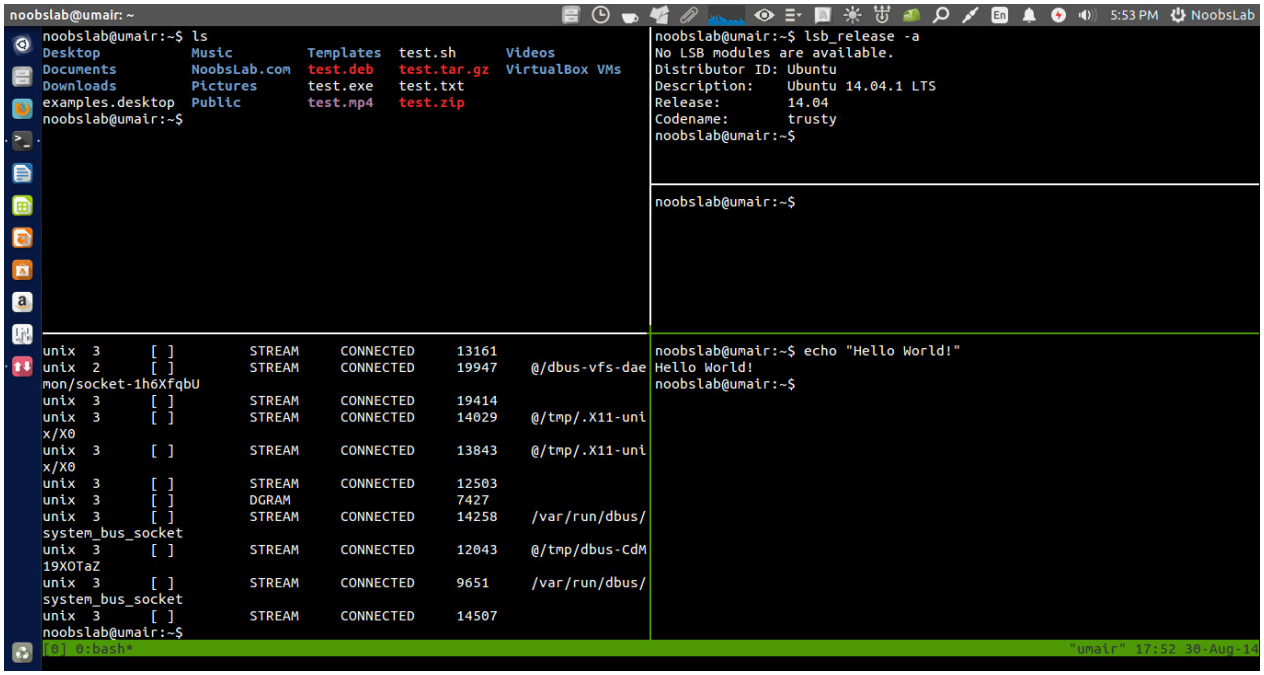


FIGURE 6.7 – Affichage de plusieurs terminaux virtuels sur une même fenêtre avec Screen

source : www.noobslab.com/2014/08/split-ubuntugnome-terminal-screen-and.html

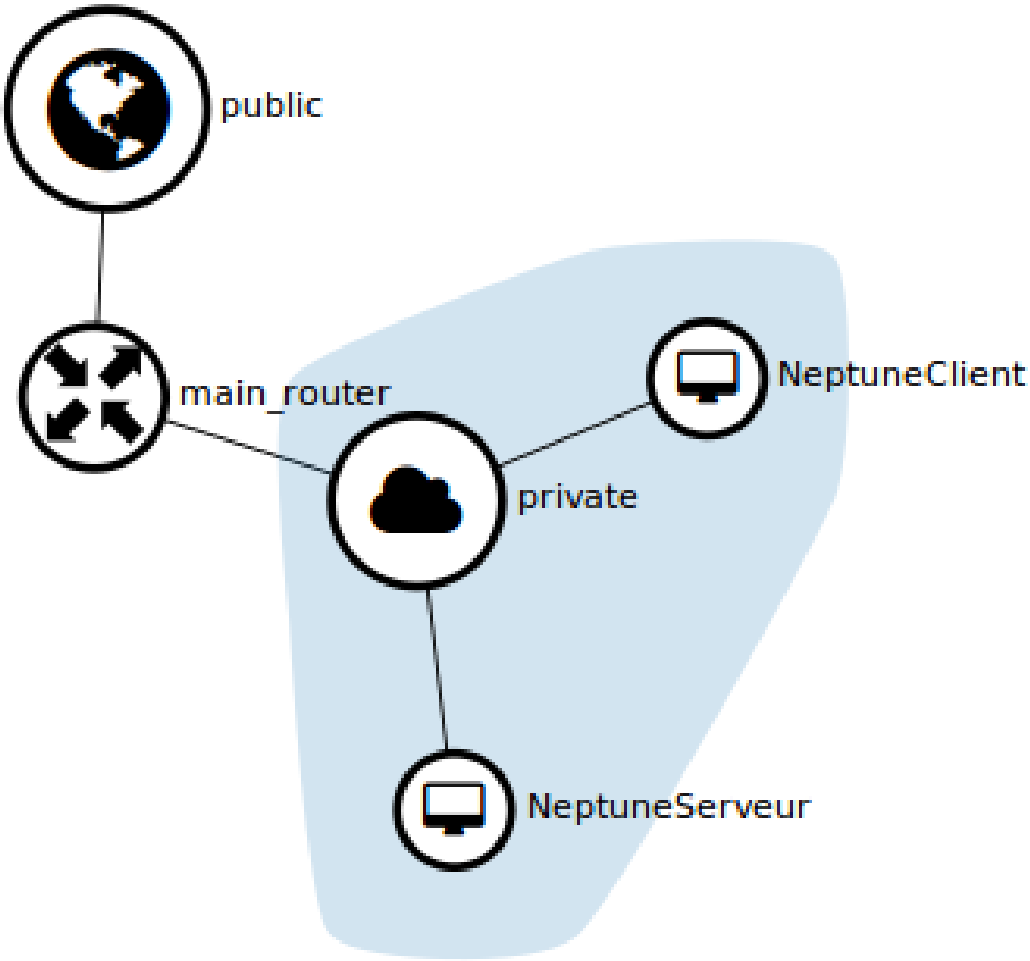


FIGURE 6.8 – Vue du cloud offerte par Horizon