Télécom SudParis
Année scolaire 2016/2017

Projet Cassiopée

Développement d'une solution de Paas

Encadrant: Aurélien Guerson

Florian Martin
Aurélien BETTINI

I Présentation du projet

Aujourd'hui, le cloud computing est de plus en plus présent et permet à n'importe quelle personne de posséder une machine virtuelle pour déployer différentes applications (sites web, applications web, services de stockage…etc). De tels services de cloud se basent sur des technologies de virtualisation et de multiples services assurant leur fonctionnement. L'objectif de ce Cassiopée est de concevoir une infrastructure fournissant des conteneurs Linux[1] gratuitement et à la demande aux étudiants et aux associations pour héberger leurs sites web, leurs données, ou d'autres applications. Cette solution sera offerte à tous les étudiants par l'association MiNET qui gère le réseau de la résidence étudiante de TMSP.

De plus, ce Cassiopée s'inscrit dans une démarche d'amélioration car cette infrastructure remplacera l'ancienne l'infrastructure d'hébergement de MiNET, qui ne répondait plus aux besoins des utilisateurs.

[1] Méthode de conteneurisation permettant d'exécuter plusieurs processus et de les isoler les uns des autres sur une même machine

Présentation de la solution

1) Fonctionnalités envisagées

a) Interface web

Un étudiant du campus de TMSP peut demander la création d'un serveur privé virtuel via une interface web. Cette interface web lui permet dans un premier temps de s'inscrire. Une fois cela effectué, un mail lui est envoyé avec son mot de passe.

L'interface web permet à un utilisateur authentifié de gérer sa machine virtuelle :

- en choisissant un système d'exploitation pré-configuré avec certaines applications préinstallées.
- en choisissant son nom de domaine en *.hosting.minet.net.
- en gérant les paramètres de sa machine (réseau, clés ssh)
- et l'état de celle-ci (la démarrer, l'arrêter, la redémarrer, la détruire).

Une section du site est réservée à des tutoriels pour permettre aux utilisateurs peu initiés de comprendre les fondamentaux du fonctionnement d'un serveur virtuel. En effet, les machines virtuelles sont des environnements linux. Ils ne peuvent donc être modifiés qu'en ligne de commande et à distance.

Enfin, l'interface web est protégée contre le top 10 des attaques du site OWASP et les entrées utilisateurs sont filtrées. Par exemple, seuls les caractères alphanumériques sont autorisés pour le nom de domaine. Les valeurs entrées par l'utilisateur sont également vérifiées par les scripts python expliqués ci-dessous.

b) Services réseau

Notre infrastructure d'hébergement repose sur différents services réseau :

- Un serveur DNS, qui s'occupe de la zone *.hosting.minet.net. Autrement dit, une requête passée au DNS de l'association MiNET, qui s'occupe de la zone *.minet.net, sera redirigée vers notre serveur DNS. Cette configuration a nécessité des ouvertures de ports sur le pare-feu de MiNET et de la DISI. Nous aurions pu ajouter les entrées directement sur le serveur DNS de MiNET mais nous ne voulions pas qu'Ansible possède les accès root sur une des machines les plus sensibles de MiNET. Un utilisateur peut ainsi choisir son nom de domaine avec un suffixe en hosting.minet.net.

- Un proxy inverse qui à partir du nom de domaine redirige vers l'ip privée du conteneur et force l'utilisation de TLS 1.2 pour les sites web. Nous avons également choisi d'utiliser un proxy inverse dédié, plutôt que de donner les accès root à ansible sur le proxy inverse de MiNET, les certificats TLS y étant stockés. Nous utilisons Let's Encrypt pour disposer gratuitement d'un certificat TLS signé par l'autorité de certification Let's Encrypt. Le certificat est automatiquement renouvelé tous les 3 mois pour le maintenir valide

- Une base de données PostgreSQL pour stocker les informations des conteneurs et des utilisateurs.
- Un serveur DHCP qui fournit les adresses IP aux conteneurs.
- Un pare-feu iptables qui agit comme un routeur pour le réseau des conteneurs.

- Une machine virtuelle sur laquelle est installée Ansible. Les différents services sont gérés par cette machine virtuelle.
- Une machine virtuelle hébergeant l'application web développée en Django.

c) <u>Automatisation avec Ansible</u>

Pour déployer un serveur privé virtuel, il est nécessaire de configurer les différents services réseau. Ces tâches sont effectuées par des playbooks Ansible, que nous avons développés :

- Il faut ajouter le nom de domaine demandé sur le serveur DNS dédié à l'infrastructure d'hébergement.
- Il faut ajouter l'entrée du nom de domaine sur le proxy inverse.
- Les machines virtuelles se trouvent sur un réseau naté derrière un pare-feu. Il faut donc ajouter des règles iptables sur celui-ci pour permettre à l'utilisateur de se connecter en ssh sur son conteneur. Il faut également s'assurer que ces règles sont persistantes, en cas de redémarrage. Il faut enfin également ouvrir certains ports si l'utilisateur choisit le template ftp que nous verrons plus loin.

- Il faut ajouter la clé ssh de l'utilisateur pour qu'il puisse se connecter en ssh. Celle-ci pouvant contenir certains caractères spéciaux tels qu'un espace, un slash ou un plus, il faut les échapper[2] pour ne pas planter nos scripts.
- Enfin, il faut ajouter l'adresse MAC du conteneur sur le serveur DHCP pour que celui-ci obtienne une adresse IP, une gateway et l'adresse d'un serveur DNS.

Toutes ces tâches sont exécutées par ansible lorsqu'un utilisateur demande la création d'un conteneur. Il pourra ensuite demander l'ouverture d'autres ports sur le pare-feu, ajouter une autre clé ssh ou tout simplement start/stop/restart son conteneur quand il le souhaite depuis l'interface web. Ces demandes sont également gérées par des playbooks Ansible.

L'ajout de ces entrées devait également être réversible. Il fallait donc ajouter toutes ces entrées intelligemment pour permettre l'utilisation d'expressions régulières lors de leurs suppressions.

Pour finir, ces données sont stockées dans une base de données pour nous permettre de retrouver l'utilisateur associé à une adresse IP par exemple.

d) Infrastructure réseau

Afin de configurer le réseau, nous avons dû configurer des équipements CISCO.
2 réseaux ont été déployés afin d'assurer le fonctionnement de l'infrastructure :

- 192.168.200.0/22 (1022 adresses ip disponibles) : sous-réseau pour les LXC. Lorsqu'un LXC est créé, il obtient une IP dans ce sous-réseau.
- 192.168.104.0/24 : sous-réseaux pour les différents services de l'infrastructure.

Enfin, nous utilisons des adresses IP publiques pour des machines essentielles :

- 157.159.40.58 : proxy inverse[3]
- 157.159.40.59 : pare-feu assurant la connexion entre les LXC et internet
- 157.159.40.56 : serveur DNS maître de la zone *.hosting.minet.net

e) LXC

Afin de proposer plusieurs services aux utilisateurs, nous leur proposons de choisir à la création de leur machine un template afin que le service choisi (exemple : Wordpress) soit pré-configuré et sécurisé.

- Ubuntu Light

Ce template fournit un conteneur Ubuntu Xenial avec le minimum de paquets installé et configuré (openssh, apache2). La connexion ssh par mot de passe ou sur l'utilisateur root est désactivée et le nom de domaine est accessible en https.

[2] Faire en sorte que certains caractères spéciaux ne soient pas interprétés par un langage de programmation (Ex: '/' en python)
[3] Rev-proxy: permet de rediriger du trafic http/https d'un hôte ayant une adresse publique à des hôtes ayant des adresses dans un sous-réseau privé.

- Wordpress

Ce template fournit un site wordpress hébergé par apache2 prêt à être installé par l'utilisateur, il est basé sur le template Ubuntu Light. L'utilisateur n'a plus qu'à se rendre sur son nom de domaine pour finaliser l'installation. Wordpress est configuré à l'aide des bonnes pratiques de sécurité recommandées par le blog de *Korben*. La base de données, les tables et l'utilisateur de wordpress commencent par un préfixe de 8 caractères générés aléatoirement, rendant ainsi les injections SQL bien plus difficiles. Wordpress est également configuré pour mettre à jour sa version et ses plugins régulièrement. Enfin, sa base de données est sauvegardée chaque semaine sur le conteneur de l'utilisateur, facilitant ainsi les restaurations. L'utilisateur est bien entendu prévenu.

- FTP

Ce template permet de fournir un serveur ftp sécurisé. Il est basé sur le template Ubuntu light et remplit deux objectifs:

- Fournir un espace de stockage pour les utilisateurs, disponible partout dans le monde
- Proposer un moyen simple aux utilisateurs de créer un site web en leur proposant un serveur FTP pour déposer les fichiers de leur site web.

vsftpd

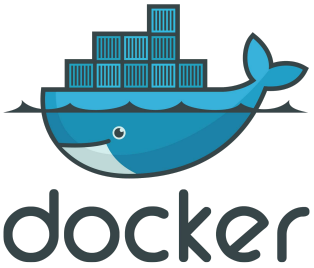
La configuration du serveur FTP est effectuée à la première connexion de l'utilisateur sur la machine. Celle-ci assure les étapes suivantes:

- Création d'un utilisateur dédié aux échanges entre le serveur FTP. Cet utilisateur possède le minimum de privilèges nécessaires sur le conteneur.
- Création d'un dossier "stockage"
- Création d'un dossier "www" étant lié au dossier /var/www dans lequel se situe tous les fichiers du site web tournant sur le conteneur.
- Création du certificat TLS auto signé pour chiffrer les échanges entre le client et le serveur.

- Openvpn

Ce template permet à un conteneur de créer des devices TUN et TAP, qui sont par défaut interdits sur les conteneurs. L'utilisateur n'a plus qu'à ajouter les redirections de port et à générer les clés nécessaires à la configuration d'openvpn.

- Docker



Ce template permet à un utilisateur d'utiliser des conteneurs Docker dans son conteneur LXC, un comportement qui est sinon par défaut interdit par LXC et Apparmor.

f) Sécurisation du système hôte

La conteneurisation est un procédé d'isolation de processus, qui partage alors le même noyau. Par conséquent, l'isolation entre conteneurs est donc moins forte qu'entre machines virtuelles, puisque celles-ci utilisent des noyaux différents, ainsi que du hardware virtualisé.

Par conséquent, nous avons étudié les mécanismes de sécurité proposés par LXC et par le noyau Linux. Nous devons nous protéger :

- Des attaques Denial-Of-Service sur l'hôte ou sur les autres LXC, en allouant seulement une partie de la RAM à chaque conteneur, 1 seul CPU sur les 8 disponibles grâce aux Control Groups et 10Go d'espace disque dans un volume LVM. Certaines ressources comme les files de message Posix ou les descripteurs de fichiers sont plus difficiles à partager.
- des kernel panics, le noyau étant partagé par le système et tous les LXC, ceux-ci peuvent potentiellement passer des appels systèmes malveillants.
- Des 0days sur le noyau linux.
- De l'exécution de code arbitraire.
- De la lecture/modification de fichiers sur l'hôte.
- De l'utilisateur qui cherchera à sortir de son conteneur pour obtenir un accès sur l'hôte.

Après quelques recherches, nous nous sommes rapidement rendu compte qu'il faudrait nous intéresser :

- Aux capabilities, aux namespaces et aux control groups, qui sont des fonctionnalités du noyau Linux permettant de limiter les ressources d'un processus etc.
- au patch de sécurité proposer par Grsecurity
- au filtrage des appels systèmes par Seccomp, nous interdisons 44 appels systèmes dangereux sur les conteneurs à l'aide d'une blacklist (ex : umount, kexec_load etc.)
- à Apparmor qui permet d'associer à chaque programme un profil de sécurité qui restreint ses accès au système d'exploitation.

g) Sécurisation du réseau, les attaques ARP

Nous devons nous protéger des attaques ARP sur le réseau des conteneurs. Pour cela nous avons mis en place 2 protections :

- Les conteneurs ne peuvent pas communiquer entre eux grâce à l'utilisation d'interfaces virtuelles macvlan en mode private, proposée par LXC. Ils ne

peuvent ainsi pas se faire passer pour la passerelle auprès des autres conteneurs.
- Le commutateur virtuel raccordant tous les LXC est configuré avec des règles ebtables, le trafic est accepté sur ce commutateur si et seulement si les adresses MAC et les adresses IP dans la trame correspondent.

2) Les outils utilisés

Afin de réaliser les programmes gérants l'automatisation des tâches de création, d'archivage, de suppression ...etc , nous utilisons du **Python** couplé avec le logiciel **Ansible**[4], permettant d'automatiser les différentes tâches nécessaires aux fonctionnements des serveurs. Nous utilisons aussi le gestionnaire de tâches du système d'exploitation linux (**Crontab**) afin de planifier l'exécution régulière des programmes **Python** et **Ansible**.

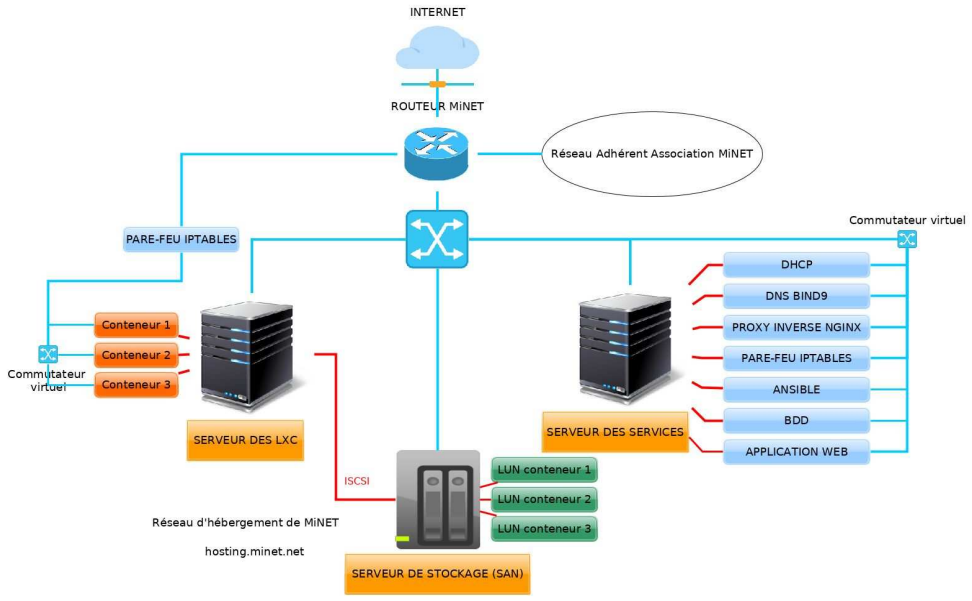
Différents programmes utilisé et configuré mais non programmé par nos soins:
- Bases de données: **Postgresql**
- Application de résolution de noms de domaines: **Bind9**
- Pare-feu: **Iptables**
- Conteneurs: **LXC non privilégiés**
- Stockage des LXC: **LVM** sur de **l'Iscsi** over **Infiniband**
- VPN de management: **Openvpn**
- Sécurité des conteneurs: **Apparmor et Seccomp**
- Proxy inverse : **Nginx**
- Certificat TLS pour le proxy inverse : **Let's encrypt**

[4] Ansible est un logiciel pour la configuration et la gestion des ordinateurs et serveurs.

3) Mise en place

Phase 1:

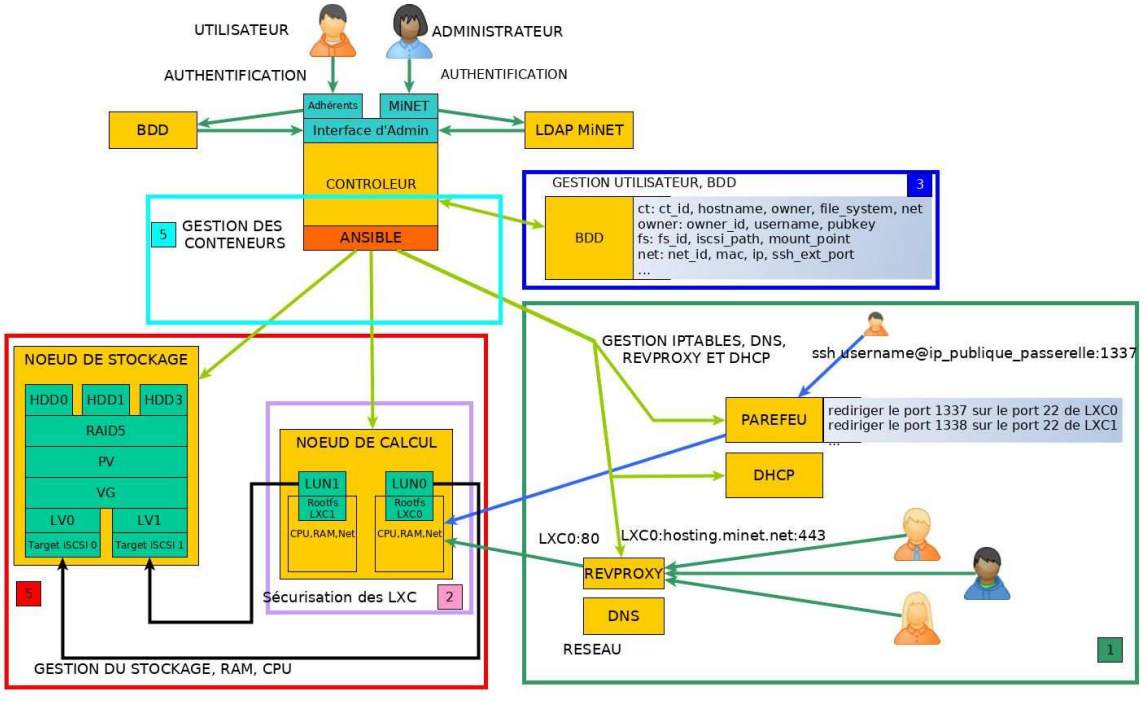
Pour l'infrastructure de la solution d'hébergement, nous avons utilisé 3 serveurs physiques



Un serveur de stockage, de grande capacité (8 To) pour stocker les disques durs que nous hébergeons, qui facilite les snapshots ou sauvegardes des disques durs.

Un serveur de calcul puissant (8 coeurs, 48 Go de RAM), ces ressources sont partagées entre les différents conteneurs que nous hébergerons.

Un dernier serveur physique pour héberger les différents services réseaux du sous-réseau d'hébergement, (proxy inverse nginx, DNS etc.). Nous avons isolé ces services sur ce serveur physique pour ne pas consommer les ressources destinées aux conteneurs pour de la virtualisation lourde.



Phase 2 et 3 :

Nous avions prévu de fournir une interface web à l'utilisateur pour gérer certaines tâches liées à l'administration de son conteneur, en plus d'un accès terminal via ssh, par exemple démarrer, arrêter ou redémarrer son conteneur. Nous souhaitions également que l'utilisateur soit autonome quant à son réseau : pour le choix de son nom de domaine, et les redirections de ports sur le réseau naté.

Nous avons pour cela développé les playbooks Ansible.

Les requêtes de l'utilisateur, par exemple l'ouverture d'un port, sont stockées dans la base de données, récupérées par des scripts python puis exécutées par des playbook Ansible.
Nous espérions ainsi être capable de déployer un conteneur et son site web en moins de 5 minutes.

Phase 4 :

Sécurisation des conteneurs, expliquée plus haut.

Phase 5 :

Rédiger une charte pour les utilisateurs protégeant l'association MiNET. Un brouillon de cette charte se trouve en annexe.

II Fonctionnement de l'infrastructure d'hébergement

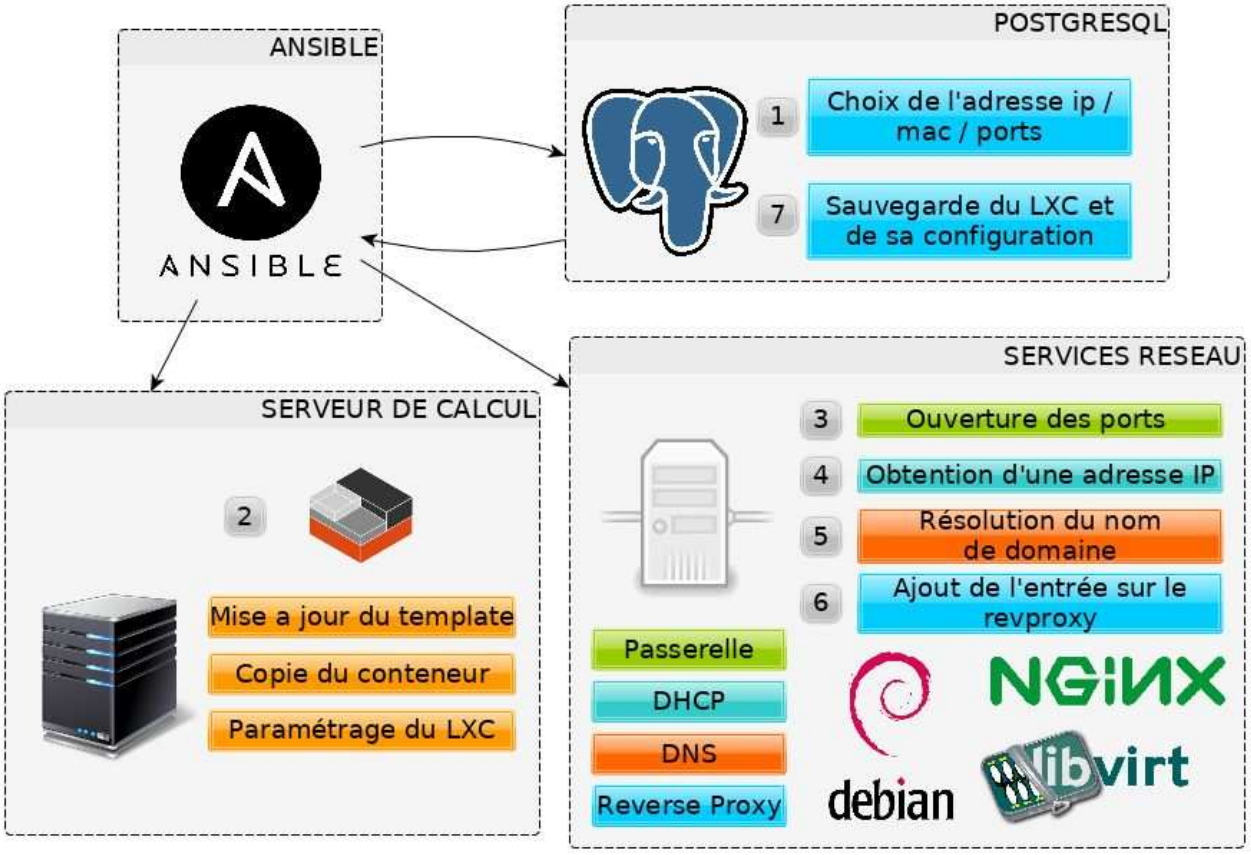
Les fonctionnalités principales de l'infrastructure sont celles permettant de gérer le cycle de vie des conteneurs. Il y a 3 fonctionnalités principales:

- Programme de déploiement d'un conteneur
- Programme d'archivage d'un conteneur
- Programme de destruction d'un conteneur
-

1) Création et déploiement d'un conteneurs

Afin de déployer un conteneur, lorsque cela est demandé par un client via l'interface, la chaîne de fonctionnement est la suivante:

1. Lorsque le client soumet le formulaire pour créer un conteneur, ce formulaire est vérifié afin de s'assurer que le client ne puisse pas avoir plus de 1 conteneur. Si la vérification est validée, la table lxc_temp_creation est complétée avec une ligne possédant les paramètres nécessaires à la création du conteneur.
2. Une cron, tâche s'exécutant régulièrement (toutes les 30 secondes) va exécuter le programme de création de conteneurs. Ce programme vérifie une nouvelle fois la validité des données avant d'exécuter le playbook Ansible de création.
3. Le playbook se décompose en plusieurs étapes:
    a. Vérification du fonctionnement de toutes les machines nécessaires au déploiement du conteneur.
    b. Requête dans la base de données afin de choisir l'adresse ip , la mac ainsi que ports attribués au conteneur.
    c. Ansible se connecte sur le nœud de calcul afin de mettre à jour le conteneur template choisi par le client afin qu'il soit le plus sécurisé possible lors de son déploiement. Le conteneur template est ensuite copié et paramétré. Le paramétrage dépend du template choisi par le client.
    d. Ansible se connecte sur le pare-feu de l'infrastructure afin d'ouvrir les ports du conteneur. De base, seul le port 22 est ouvert. D'autres ports peuvent être ouverts suivant le template choisi.
    e. Ansible se connecte le serveur DHCP afin de modifier la configuration de celui-ci et d'assurer que le conteneur puisse obtenir une IP et se connecter au réseau.
    f. Ansible se connecte au serveur DNS pour ajouter l'entrée DNS **nomconteneur.hosting.minet.net.** L'adresse IP du conteneur pourra être résolu via le nom de domaine précédent.
    g. Ansible se connecte au serveur de proxy inverse permettant de proposer une connexion https au site web présent sur l'infrastructure du hosting.



4. A la fin de l'exécution du playbook ansible, l'application va enregistrer les données du conteneur dans la table lxc de la BDD. L'entrée de la table lxc_temp_creation est ensuite supprimée.

2) Archivage d'un conteneur

Lorsqu'un client demande la suppression de son conteneur. Le conteneur n'est pas détruit instantanément. En effet, afin de se prémunir de tous problèmes, le conteneur n'est pas supprimé afin de garder une trace de celui-ci pendant 1 mois.

Le conteneur est à la place archivé. Les étapes sont les suivantes:

- Le conteneur est stoppé
- L'entrée du conteneur dans la table lxc est déplacée vers adherent_lxc_archive pour qu'il n'y ait plus accès en interface web. La date de destruction est modifiée à la date archivage + 30 jours.
- La colonnes archived de la table lxc_network est passée à TRUE pour indiquer que l'ip est encore utilisée par le conteneur archivé.

3) Destruction d'un conteneur

Une cron vérifie si il existe des conteneurs dont la date de destruction est supérieur à la date actuelle. Si cela est la cas, la cron va exécuter le playbook de destruction ansible. Ce playbook va supprimer tous les paramètres relatifs au conteneur à détruire sur tous les services (dhcp, rev-proxy, dns, passerelle, base de données). Le conteneur est enfin supprimé du noeud de calcul.

III Planification

1) Analyse et comparaison avec les objectifs initiaux

Etude de la faisabilité et choix des solutions:

- Nous avons passé plus de temps sur le rackage des serveurs physiques que prévu. En effet, le serveur de stockage utilisé était très lourd et les modifications de certaines de ces cartes nous ont obligé à le racker / dé-racker.
- Mise en place des KVM pour les services réseaux sur l'infrastructure de MiNET : Nous pensions avoir de l'expérience sur les applications implémentées pour les services DNS, DHCP… etc mais cela a pris plus de temps que prévu.
-

Nous avons également fait fasse à des problèmes que nous avons mis du temps à résoudre :

- Automatisation de la création des LXC avec Ansible : Le temps perdu sur cette partie vient principalement des phases de tests / validations afin de vérifier la sécurité de l'infrastructure. De plus, nous avons changé de vision en cours de route concernant les programmes développés en bash et python.
- Développement de l'application web pour gérer les containers et les utilisateurs : Django est un framework facile à prendre en main. Cependant, certaines fonctionnalités à implémenter telles que l'authentification, la gestion propre des formulaires nécessitent un certain temps d'apprentissage.
- Sécurisation du système et du réseau : Certaines solutions que nous souhaitions utiliser telles que Grsecurity ne sont plus open-source. Afin de s'assurer que le système soit maintenable dans le futur, nous avons changé de solution.

2) Difficultés rencontrées et solutions

Au cours du projet, nous avons pu rencontrer des problèmes nous amenant à revoir nos objectifs, notre planning ainsi que les solutions utilisées. Ceux-ci sont présentés ci-dessous:

- **Problème de Climatisation**

Notre infrastructure de hosting a été déployée au sein de l'infrastructure de l'association MiNET. Cela nous rend dépendants des aléas pouvant intervenir sur cette infrastructure. Nous avons fait fasse à 3 interruptions de services car le système de climatisation de la salle serveur dans laquelle étaient nos machines a subi des dysfonctionnements. En effet, le système de refroidissement gelait. Lors de ces incidents, la climatisation n'est plus utilisable le temps que celle-ci dégèle ou qu'une intervention soit effectuée. Du fait du nombre de machines dans les baies de la salle serveur, si toutes les machines et services continuaient de fonctionner, la température ambiante pourrait atteindre les 35°-40°C. Notre infrastructure n'étant pas un service critique de l'infrastructure de l'association MiNET, nous étions obligés d'éteindre nos 3 machines, le temps de régler les problèmes.

1ère interruption de services: Nous avons éteint les 3 machines. Cependant, l'infrastructure ayant été récemment implémentée lors de l'incident, nous ne savions pas dans quel ordre nous devions éteindre les services. Le problème majeur est venu du fait que le serveur de stockage est relié par le réseau au serveur de calcul. Si une coupure du serveur de stockage a lieu alors que ses disques sont montés sur le serveur de calcul, ses partitions de stockages (lvm thin-pool) pourraient se corrompre. Nous avons, durant ce premier incident, éteint le serveur de stockage en même temps que le serveur de calcul. Par conséquent, nous avons perdu toutes les données des conteneurs que nous avions testés. De plus, nous avons perdu 1 journée à réparer le système pour refaire fonctionner l'infrastructure.

Solution : Nous avons mis en place une procédure d'extinction des services pour être certains qu'un redémarrage des machines ne nécessiterait pas de réparations de notre part, et nous avons documenté les méthodes de restauration. Le logiciel d'interconnexion a été re-paramétré afin qu'il se lance automatiquement.

2ème interruptions: Les précautions prises précédemment nous ont permis de refaire fonctionner l'infrastructure rapidement.

3ème interruptions: Une panne de climatisation s'est produite au milieu de la nuit, la température ambiante a dépassé les 37°C. Notre serveur de calcul s'est donc arrêté à cause de la chaleur, ce qui a corrompu les volumes de stockage. Cependant, nous avons été capables de retrouver toutes les données dans la journée.

- **Problème avec le LDAP de la DISI**

Nous souhaitions utiliser le LDAP de la DISI comme moyen d'authentifications pour que les personnes scolarisés de TMSP puissent se connecter avec leurs identifiants DISI. Cependant, cette méthode d'authentification est compliquée à mettre en place avec Django. Nous avons passé une semaine sur cette implémentation mais sans succès.

Après cet échec, nous avons décidé d'utiliser une méthode d'authentification standard : L'utilisateur s'enregistre avec une adresse appartenant à l'école, un mail est envoyé pour lui confirmer son inscription, puis celui-ci peut se connecter.

- **Défaut de batterie sur une des cartes RAID**

Une des batteries du module BBU d'une des cartes RAID était en défaut. Le serveur physique que nous utilisions étant customisé, nous n'avions aucune documentation sur la batterie. A l'aide d'un voltmètre et le peu documentation trouvée sur cette carte RAID, nous avons tout de même réussi à changer la batterie.

- **Grsecurity,** une modification augmentant la sécurité pour le noyau n'est plus libre depuis Avril 2017

Nous avions prévu de patcher le noyau linux du serveur de calcul avec Grsecurity, distribué sous la licence publique générale GNU version 2. Malheureusement, ce patch de sécurité est maintenant commercialisé par **Open Source Security Inc.** Le patch n'est donc plus disponible pour les dernières versions du noyau. Nous comptions utiliser le noyau patché avec Grsecurity et compilé par Debian dans ses dépôts backport, mais nous pensons que ce noyau ne sera bientôt plus disponible dans les dépôts, le patch n'étant plus libre. Par conséquent nous n'utilisons pas ce patch de sécurité.

- **Nous avons revu 3 fois les scripts Ansible**

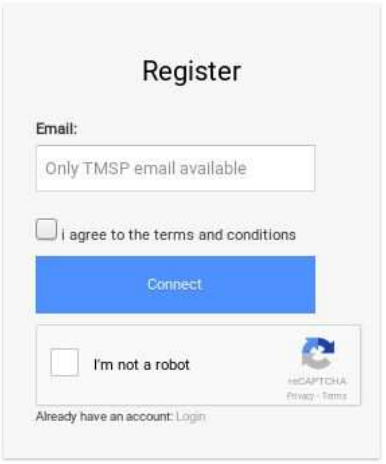
Nous souhaitons tout d'abord tout exécuté avec Ansible, c'est-à-dire récupérer les données dans la base de données, vérifier qu'elles sont correctes pour enfin exécuter les tâches nécessaires sur chaque serveur. Nous nous sommes rapidement rendu compte que le langage utilisé par ansible (yaml) ne se prêtait pas à toutes ses tâches.

Nous sommes ensuite parti sur des scripts en bash pour exécuter les playbooks, et des API pythons pour interagir avec la base de données. Tout marchait à merveille jusqu'au moment où nous nous sommes intéressés à la centralisation des logs. Il était difficile à la fois de rediriger tous les stdout d'exécution, et de les parser avec des headers de date, de PID etc. Plus généralement, mélanger 2 langages de programmation est une mauvaise idée.

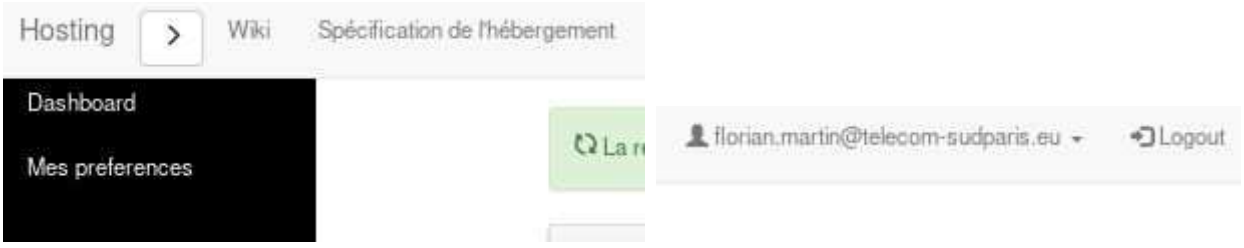
Nous avons donc fini par tout faire en python2. Malgré quelques difficultés pour les appels systèmes et la centralisation des logs, les scripts et les logs sont maintenant propres, administrables et maintenables.

IV Resultats

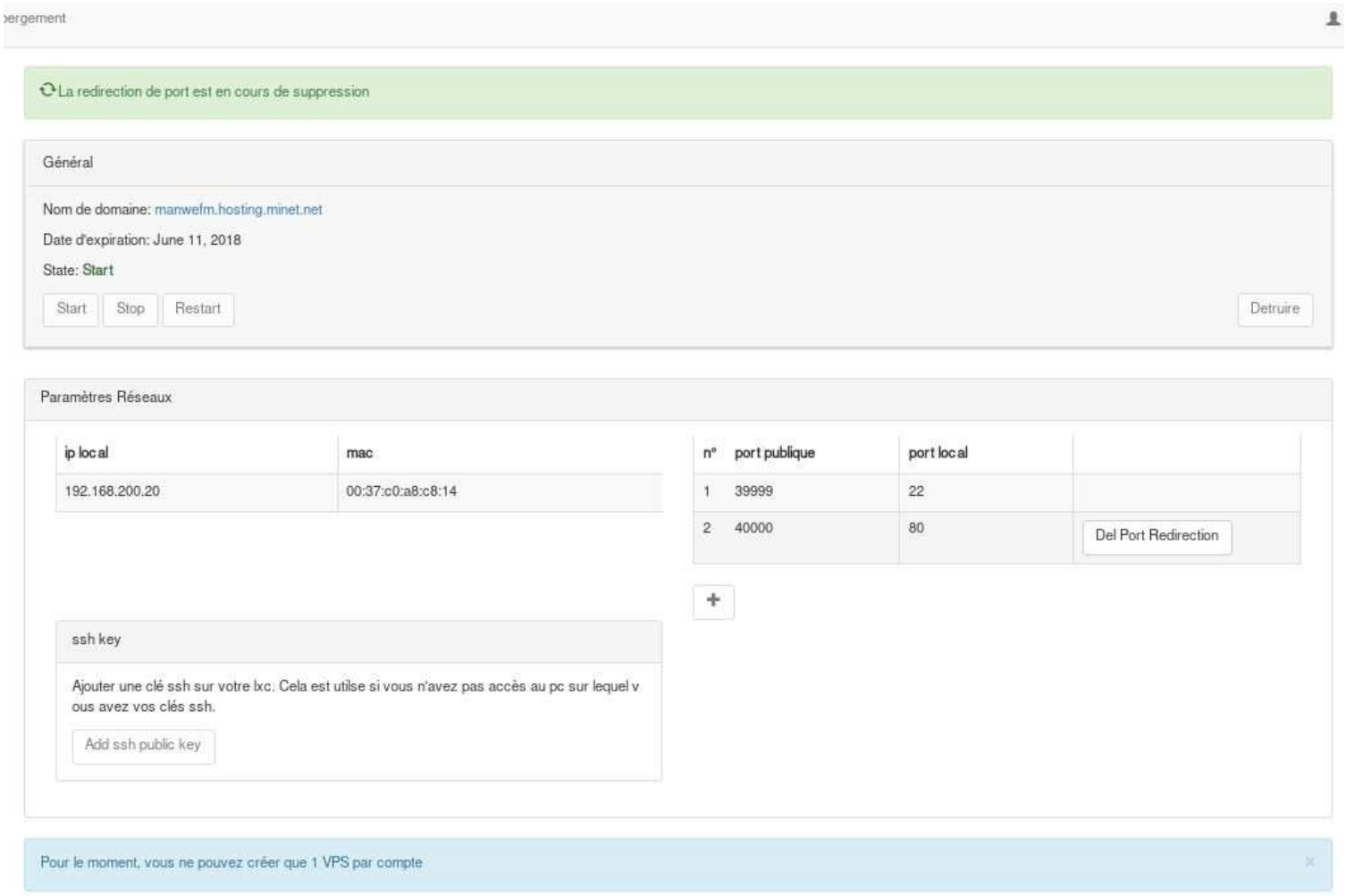
Les étudiants peuvent s'enregistrer sur le site pour créer leur LXC. La page d'enregistrement est la suivante.



La navigation sur le site est intuitive et facilitée par la barre de menu située sur le haut, complétée par une barre latérale qui peut être fermée ou ouverte.



L'utilisateur à accès après connexion à son tableau de bord pour gérer son conteneur une fois qu'il en a créé un.



Voici quelques statistiques de l'infrastructure:

- Temps moyen de création d'un conteneur avec le template wordpress: 3min30sec
- Temps moyen de création d'un conteneur standard, ne possèdent que ubuntu: 3min
- Temps moyen pour effectuer une tâche (ajouter un port etc.): 30 sec

Nous avons effectué 191 commits pour un total de 3939 lignes de code pour les scripts Ansible, 80 commits pour 24937 lignes de code pour le site web, code Django compris.

Annexes

Nous avons complété un wiki au cours du projet pour faciliter la maintenance de cette infrastructure dans les années à venir :
https://wiki.minet.net/wiki/hebergement/hebergement_lxc

Charte d'Hébergement de conteneurs sur les

serveurs de l'association MiNET

Service fourni aux adhérents de l'association

La présente charte vise à :
- Définir les conditions d'hébergements de conteneurs LXC par MiNET
- Définir les règles applicables à respecter afin de bénéficier du service d'hébergement

L'hébergeur

Le conteneur sera hébergé sur les machines de l'association MiNET. Les contenus et hébergements sur ces machines seront validés par les membres de ladite aéssociation. En cas de non-respect des règles énoncées dans cette charte, le conteneur pourra être mis hors ligne (partiellement ou totalement). La remise en ligne sera effectuée qu'après que le contenu soit de nouveau jugé conforme à la présente charte.

Responsabilité

L'association MiNET ne pourra en aucun cas être tenue responsable du contenu des conteneurs hébergés. En adhérant à cette charte, le responsable du conteneur assume l'entière responsabilité (civile et pénale) des informations mises en ligne. En cas de diffusion de contenu ne respectant pas les lois et règlements ou pouvant porter un préjudice quelconque, le responsable sera invité à interrompre immédiatement la diffusion des informations incriminées dès que ces faits auront été portés à sa connaissance. Dans le cas où aucune suite n'aurait été donnée à cette invitation, l'association MiNET se réserve le droit de suspendre l'hébergement du conteneur concerné.

Règles techniques particulières relatives à l'hébergement du conteneur

- Le conteneur et son contenu doit être maintenu à jour (date de 15 jours maximum). L'association MiNET se réserve le droit de mettre à jour le système du conteneur et son contenu sinon sinon.
- Les sites web hébergés sur le conteneur doivent comporter le nom de l'auteur, ainsi qu'une adresse électronique de contact
- Le conteneur ne peut pas être utilisé à des fins commerciales
- Le responsable ne doit pas effectuer des manœuvres qui auraient pour but de méprendre les autres utilisateurs sur son indentité
- L'utilisateur doit rester abonné à la mailing-list de sécurité visant à lui demander de mettre à jour son conteneur en cas de vulnérabilités mineures comme majeures.
- L'utilisation de logiciels P2P est interdite

Cadre juridique général

Le contenu du conteneur se doit de respecter les lois en vigueur, mais également les obligations suivantes (ceci constitue une liste non exhaustive) :
- Le respect de la loi sur les informations nominatives
- Le respect du droit d'auteur (droit moral, patrimonial, de diffusion)
- L'abstention de porter atteinte à la vie privée ou au droit à l'image d'autrui
- L'absence de diffusion d'informations non vérifiées ou présentant un caractère délictueux
- Le respect des exigences de la loi «Informatique et Libertés »
- L'absence de propos à caractère raciste, xénophobe
- Toute forme d'apologie du crime, du racisme, d'invitation à la haine raciale est interdite
- Tous propos injurieux ou diffamatoires sont interdits
- Tous contenus à caractère pornographique sont interdits

Sanctions

En cas de non-respect des règles énoncées dans cette charte, le conteneur concerné pourra être mis totalement ou partiellement hors-ligne par les membres de l'association MiNET, La remise en ligne ne sera effectuée qu'après que le contenu soit de nouveau jugé conforme à la présente charte.

Tout utilisateur n'ayant pas respecté les dispositions du présent règlement est passible, suivant la gravité et la nature de l'infraction :
- d'un rappel à l'ordre
- d'interdiction temporaire ou définitive
- d'accéder aux conteneurx et aux services hébergés
- d'une radiation de l'association MiNET
- de sanctions internes à TSP & TEM pouvant aller jusqu'à la convocation devant le conseil de discipline ;
- de poursuites pénales prévues par la loi

et ce sans que le responsable du site ne puisse prétendre à aucun dédommagement d'aucune sorte.

Arrêt de service :
Le rôle de l'association MINET est limité à la mise à disposition d'espace disque et des possibilités techniques du centre serveur. Cette mise à disposition peut être interrompue :

- Immédiatement : en cas de force majeure, en cas d'opérations urgentes de maintenance, en cas d'arrêt de fourniture des prestations d'interconnexion au réseau par la DISI, quel qu'en soit le motif, en cas de non-respect de la présente charte ou de plaintes émanants de tiers.
- Après respect d'un préavis : en cas d'opérations prévisibles nécessaires à la maintenance du serveur (déplacement d'un serveur, maintenance préventive, augmentation des capacités des machines, mise à niveau des logiciels...)

Règles particulières

Les administrateurs ou membres du Bureau de l'association MiNET qui, par leur fonction, possèdent des droits plus étendus leur permettant d'avoir accès à des informations confidentielles sont tenus de respecter le secret professionnel et les consignes de sécurité émises par le président de l'association MiNET. Ils doivent s'abstenir de toute intervention susceptible de compromettre la sécurité et le fonctionnement des conteneurs des utilisateurs.

Prise d'effet - Validité

La présente charte prend effet dès signature par le responsable du conteneur et est valide pendant toute la durée de l'hébergement du conteneur sur les serveurs de l'association MiNET.

Signature du responsable du conteneur