Rapport final de Projet
Nuage pour sites Web

IMA5 2015-2016

février 2016

Thibaut Scholaert **Jérémie Denéchaud**

SOMMAIRE

Introduction

Dans le cadre de notre projet de dernière année à l'école d'ingénieur Polytech Lille, nous avons l'opportunité de travailler sur une solution info-nuagique pour créer et héberger les sites web de l'école.

L'objectif est de permettre à tout étudiant, association ou personnel encadrant de créer facilement un site web sur les serveurs de l'école. L'architecture utilisée doit être légère, robuste et documentée pour remplacer, à terme, l'infrastructure existante.

Le **cahier des charges** précis est rappelé dans une première partie. Nous développerons ensuite le **travail effectué** dans une seconde partie. Nous aborderons, pour finir, les **difficultés rencontrées** et les solutions apportées.

1. Rappel du cahier des charges

Le but du projet est de mettre en place une infrastructure info-nuagique permettant de créer rapidement et simplement un site web pour un utilisateur n'ayant pas nécessairement de connaissance en informatique.

La solution recherchée doit pouvoir remplacer celle existante et donc au moins supporter la charge actuelle des sites existants. Plus précisément, les serveurs hébergent actuellement environ 2000 sites web dont 200 sont considérés comme actifs. Le matériel à utiliser n'est pas imposé, ni défini. Nous nous efforçons donc de garder une infrastructure qui s'adapte à différentes échelles. Le choix des technologies à utiliser est libre mais l'architecture finale doit remplir des contraintes de légèreté et de performance.

L'accès à l'ensemble des serveurs doit se faire à travers une unique adresse IPv4 et IPv6 routée. L'utilisateur doit pouvoir gérer son site et y ajouter du contenu avec au moins autant de facilité que ce que propose le système actuel (FTP). La gestion des certificats pour un accès en HTTPS doit aussi être prise en compte. Différents modules doivent être mis à la disposition de l'utilisateur comme par exemple un module php. Le nom du site est fixé par l'utilisateur et doit être implémenté très simplement.

2. Travaux effectués

2.1. Choix technologiques

2.1.1. Docker et les processus isolés

L'idée initiale était l'utilisation de **Docker**, une technologie récente utilisant les conteneurs. Cette technologie offre une grande légèreté dans son utilisation mais également dans son déploiement. Dans ses applications on peut trouver des serveurs Web tel qu'Apache ou Nginx, des modules additionnels (PHP, Ruby on rails, etc.) mais aussi des solutions de serveur virtuels complets (OVH). Cette solution proposée en open source nous a semblé un choix pertinent pour embarquer un serveur Web dans un conteneur virtuel.

Nous sommes néanmoins en lieu de nous demander quels vont être les avantages de Docker par rapport à une virtualisation classique. Tout d'abord, un conteneur n'embarque pas de système d'exploitation supplémentaire, il utilise le noyau du système sur lequel il s'exécute. Nous avons donc une grande rapidité d'exécution mais également une perméabilité accrue entre la machine hôte et le conteneur. Docker étant récent et attrayant, on trouve beaucoup de documentation à son sujet. Il existe par exemple une base de données de conteneurs pour diverses applications. On peut citer quelques entreprises ayant intégré ce système comme Amazon sur AWS (Amazon Web Services), Google sur Google Compute ou encore OVH et DigitalOcean.

Nous avons donc rapidement décidé, en suivant les recommandations de nos encadrants, d'utiliser Docker. Durant les premières semaines de travail, nous nous sommes renseignés sur cet outil et sur son fonctionnement. Nous l'avons installé sur une machine et réalisé de multiples tests. Docker utilise donc le noyau de la machine hôte et les conteneurs Linux LXC notamment pour leur système d'isolation.

2.1.2. Supervision des conteneurs

Le but était ensuite de savoir comment utiliser au mieux Docker, l'adapter à nos besoins et l'intégrer dans notre système. Pour cela l'utilisation d'Openstack nous a été initialement suggérée pour la supervision de notre système. Nous nous sommes donc renseigné à ce sujet auprès de Monsieur Redon, Alexandre Chojnacki un intervenant extérieur avec qui nous avons eu cours et Thomas Maurice, IMA diplômé en 2015 utilisant Docker et Openstack quotidiennement. De ces différentes entrevues nous en avons conclu que l'utilisation d'Openstack était **trop lourde pour être pertinente** pour notre utilisation. Nous avons donc arrêté nos recherches à ce sujet et décidé de mettre cet outil de côté. Nous créons nous-mêmes tous les scripts nécessaires au déploiement automatique de site.

2.1.3. Machines virtuelles Xen

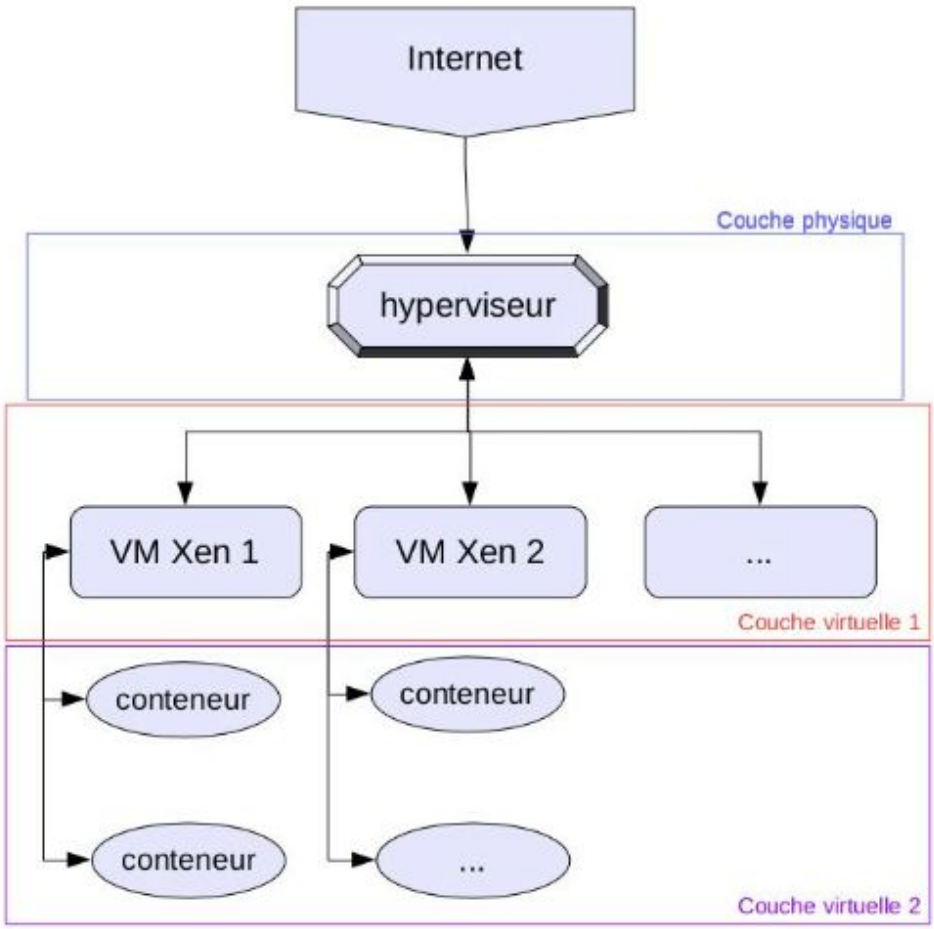
Suite à ces discussions et à d'autres recherches nous avons également discuté de l'utilisation d'une couche intermédiaire de virtualisation pour une meilleure isolation et résilience en cas de défaillance. Il existe plusieurs outils tels que xen, kvm ou vmware. Nous nous sommes néanmoins rapidement tournés vers une solution à base de paravirtualisation, l'hyperviseur de machine virtuelle (VM) Xen. Cet outil nous permet de créer plusieurs VMs sur lesquels nous avons des conteneurs Docker.

Nous avons initialement travaillé sur les machines de la salle projet pour réaliser nos différents tests sur Docker et Xen séparément. Un nouvel élément est venu changer notre façon de tester. L'annexe 1 précise ce point là.

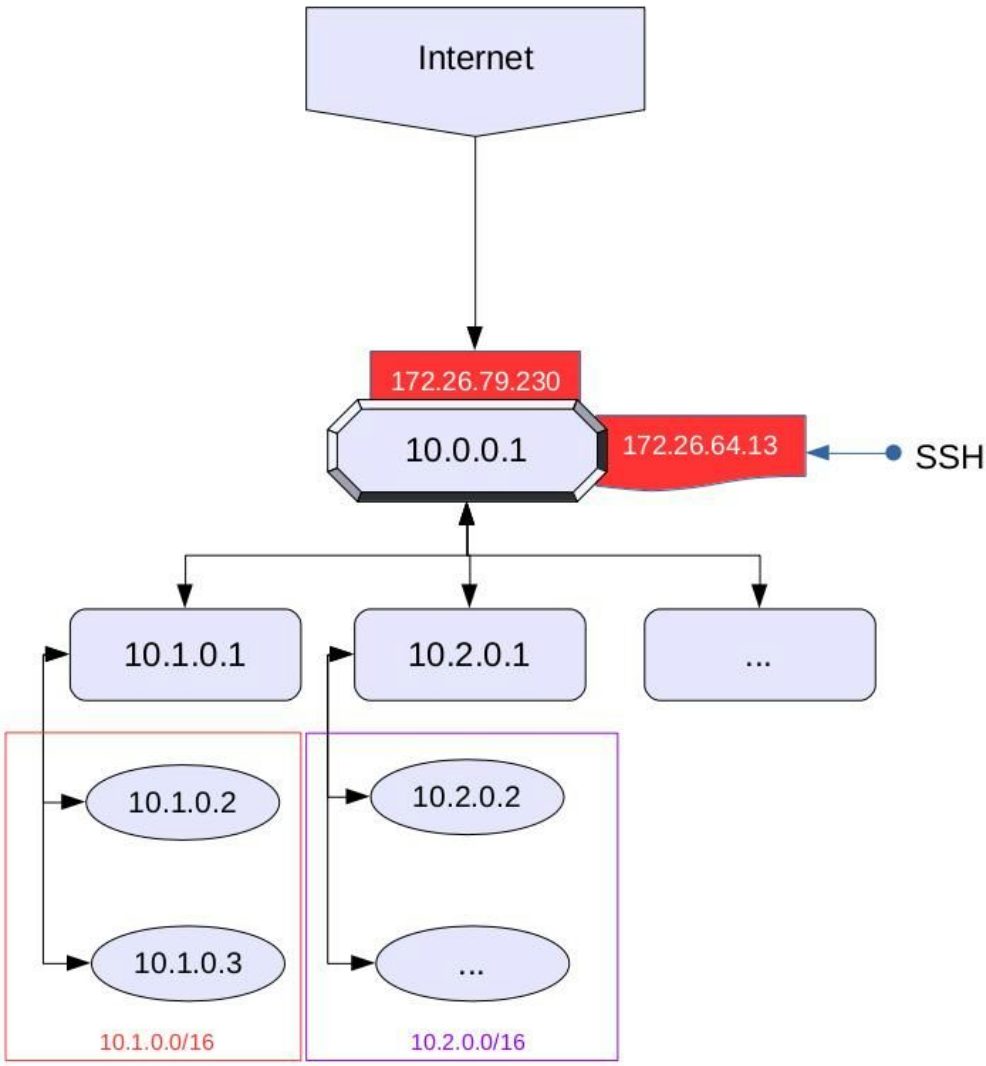
2.2. Back-end

2.2.1. Architecture réseau

Notre problématique sur l'architecture réseau est de faire communiquer les conteneurs avec un serveur principal qui joue le rôle de *proxy*. Par défaut, docker possède son propre réseau et son bridge en 172.17.0.0/16. Ce réseau est paramétrable dans une certaine mesure mais pour notre utilisation nous avons dû créer notre propre réseau. Plus d'informations sur la mise en place de ce réseau sont disponibles dans la section 'Problèmes Rencontrés'.

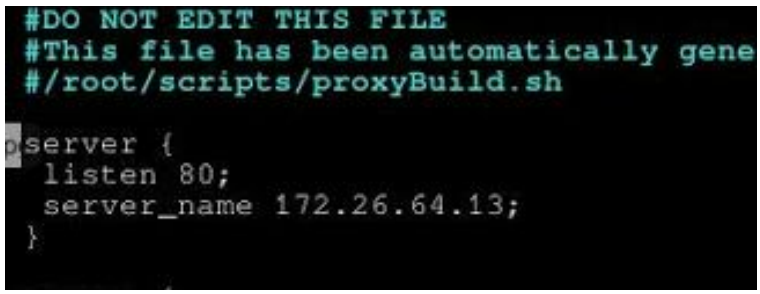


L'architecture comporte un serveur principal dénommé l'*hyperviseur*. Dessus, des machines virtuelles Xen (VM) sont manuellement installées. Enfin, sur chaque VM tournent environ 250 conteneurs Docker au maximum. Nous avons choisi un sous-réseau de classe A pour un maximum de flexibilité, pour que notre architecture puisse être facilement adaptée à une autre échelle. Une mascarade est nécessaire pour permettre aux conteneurs d'accéder à internet. Ci-dessus la maquette représente les trois couches de notre architecture. Voici maintenant un autre schéma illustrant les plages d'adresse ip allouées.



2.2.2. Proxy inverse

Nous avons choisi de mettre sur l'hyperviseur un proxy **Nginx**. Ce choix a été motivé par la simplicité de configuration et les bonnes performance de celui-ci. Ce proxy permet de transférer une requête extérieure au bon serveur en fonction du *header*. Voici un exemple de configuration pour un site :



On écoute sur le port 80 de l'hyperviseur. Ceci permet de répondre au cahier des charges en ne rendant accessible qu'une adresse ip, et uniquement les ports web (80 et 443). On spécifie un nom de serveur puis on indique à Nginx d'analyser l'entête de la requête HTTP. On trie ici les requêtes ayant pour *HOST* : a.domain.tld. On donne ensuite l'adresse du conteneur qui possède le serveur pour ce nom de domaine. Pour construire une telle configuration nous utilisons un script. Voici un résumé du fonctionnement du proxy inverse, adapté à notre configuration

2.2.3. SSH inverse

Nous nous sommes renseignés sur le ssh inverse pour qu'un utilisateur puisse déposer des fichiers sur son site. La difficulté majeure réside dans la complexité de notre réseau. En effet, une connexion extérieure doit passer par le proxy de l'école, par notre hyperviseur, pour ensuite atteindre un conteneur.

Nous avons effectué un test fructueux de connexion à un conteneur (*dock0*) à partir d'une machine du réseau weppes (*titus10*). Voici le déroulement du processus :

- Connexion de dock0 à notre hyperviseur en créant un pont
  - ssh -R **10102**:*dock0*:22 root@*hyperviseur*

- Connexion de *titus10* à notre hyperviseur
  - ssh root@*hyperviseur*

- Connexion de *titus10* au pont créé précédemment
  - ssh **-p 10102** user@*dock0*

L'utilisateur *user* a été créé au préalable pour représenter l'utilisateur ayant les droits d'accès au dossier contenant les fichiers web. Ce processus nécessite cependant un accès à notre hyperviseur et se déroule en deux étapes.

2.2.4. Base de données

Il est nécessaire d'établir une base de données pour construire la configuration du proxy inverse. Celle-ci va également nous permettre de garder une trace sur le nombre de domaines par nom d'utilisateur. On se sert également de cette base pour afficher des informations sur l'interface Web décrite plus en détail dans la partie correspondante. Ci-dessous sont regroupées les informations importantes à propos de l'unique table de cette base. Le *login* a été pensé pour être celui de polytech, de 8 lettres, en prenant en compte les doublons qui peuvent ajouter jusqu'à 2 chiffres supplémentaires. La clé primaire aurait pu se faire sur les adresses des conteneurs mais notre façon de les créer nous a poussé à choisir le nom de domaine.

login	**vlan**	**docker_ip**	**domain**
varchar(10)	int	varchar(15)	varchar(50), **PK**

2.2.5. Scripts

Cinq scripts ont été réalisés pour effectuer les tâches demandées.

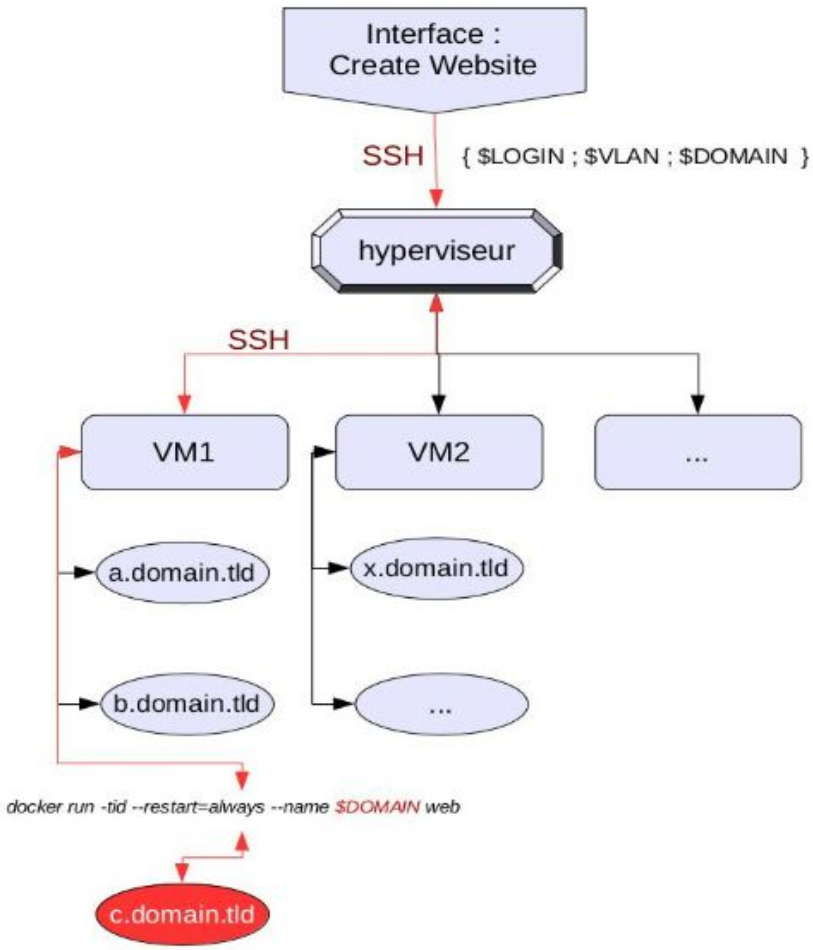
- **Création**

Usage: create.sh USER VLAN DOMAIN_NAME

Le script *create.sh* permet de se connecter à une machine virtuelle depuis l'hyperviseur afin d'y créer un conteneur avec les paramètres renseignés par l'utilisateur. Dans un premier temps, le script vérifie si l'utilisateur n'a pas déjà 3 sites hébergés et si le nom de domaine n'est pas déjà utilisé grâce à une requête SQL à la base de données décrites précédemment. Ensuite, on insert une nouvelle ligne dans la table avec un champ vide pour l'adresse IP du conteneur. On crée alors un nouveau conteneur portant comme nom le domaine spécifié. Ceci permettra de l'identifier facilement à l'avenir.

docker run -tid --restart=always --name $DOMAIN web

Voici un schéma de l'éxécution du script, dans le contexte décrit précédemment.



- **Suppression**

Usage: delete.sh DOMAIN_NAME

La suppression s'occupe d'arrêter et détruire un conteneur, ainsi que de nettoyer la base de données de la ligne correspondante.

- **Information sur le conteneur**

Usage: fetchinfo.sh DOMAIN_NAME

Ce script se situe sur la machine virtuelle. Il permet de récupérer aisément les informations sur un conteneur. On se concentre ici tout particulièrement sur l'adresse IPv4. Ce script

- **Vérification des adresses IP**

Usage: checkIP.sh

Ce script passe en revue chaque entrée de la table de notre base de données. Il vérifie pour chaque nom de domaine si l'adresse IP associée est correcte. En effet si le conteneur est arrêté et relancé, son adresse IP allouée dynamiquement par le démon Docker peut avoir changé. Ce cas intervient par exemple si un autre conteneur est créé pendant l'interruption. On s'assure également que lors d'un changement d'adresse, tous les conteneurs concernés changent en même temps. Ceci évite des problèmes d'accès ou d'incohérences. Ce script a donc pour vocation d'être une tâche régulière, exécutée par exemple dans un *cron* toutes les minutes.

- **Construction du proxy**

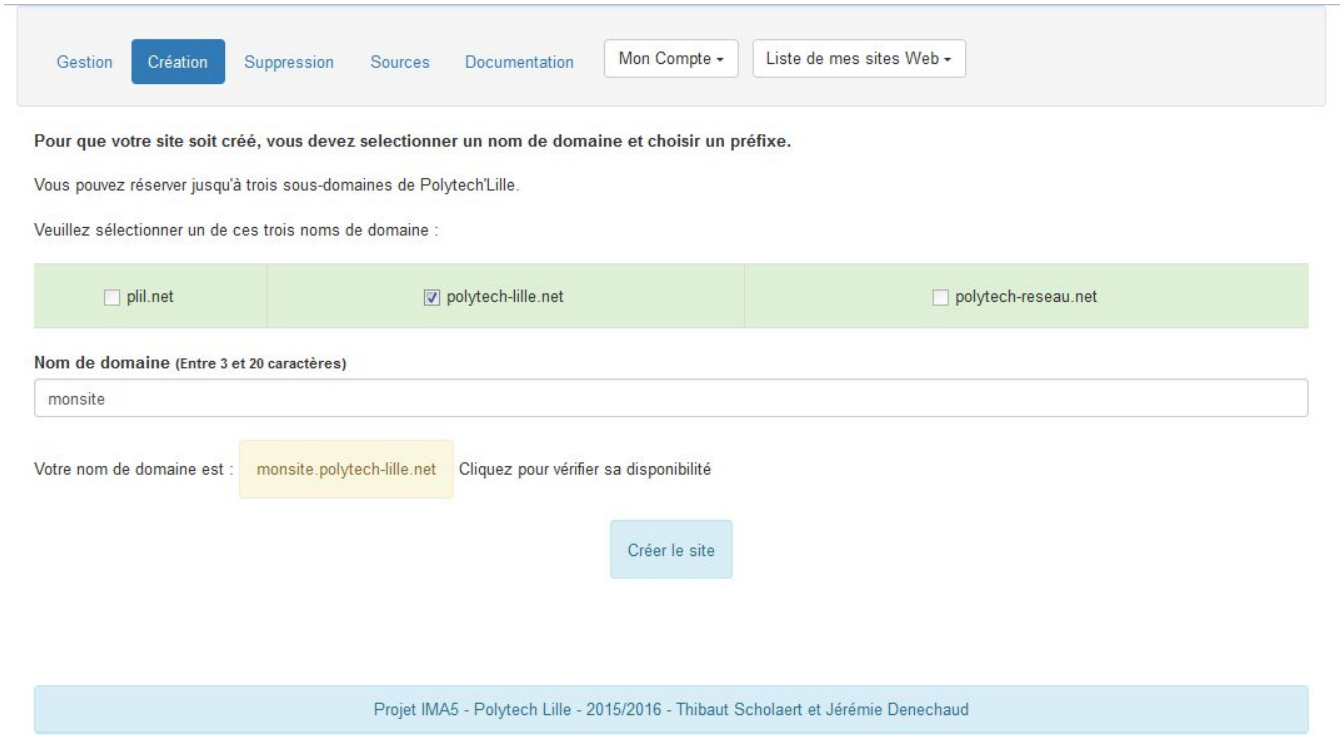
Usage: proxyBuild.sh

La construction de la configuration du proxy inverse est automatisée par ce script. Il écrase la configuration précédente par une configuration avec les adresses IP mises à jour grâce au script précédent. Il lit donc simplement la base de données après chaque mise à jour, reconstruit une configuration et recharge le démon du proxy. Il a donc lui aussi tout intérêt à s'exécuter chaque minute.

2.3. Front-end

2.3.1. Interface

Nous avons bien entendu créé une interface Web qui permet à l'utilisateur d'utiliser tout ce système, à savoir créer, supprimer, gérer ses sites Web école. Nous avons choisi de créer une interface possédant une unique page afin de n'en avoir qu'une à charger sans compter la page d'authentification. Pour cela, nous avons choisi d'utiliser le framework **AngularJS** qui nous permet d'obtenir un site dynamique visuellement et bien rangé. Les différentes rubriques (création, suppression, gestion, etc) sont accessibles via un menu qui permet de naviguer avec fluidité. Pour l'esthétique, nous avons opté pour **bootstrap**. Nous avons donc une interface qui offre une navigation fluide et rapide, en voici un petit aperçu :



Grâce à quelques fonctions javascript supplémentaires, nous pouvons contrôler que l'utilisateur rentre bien un nom de domaine valide. Lorsqu'il veut créer un site, il ne peut sélectionner qu'un seul nom de domaine parmi les 3 disponibles : plil.net, polytech-lille.net et polytech-reseau.net. De plus, lorsqu'il rentre le préfixe du sous domaine qu'il veut, il ne peut y entrer que des chiffres et des lettres.

L'interface Web comporte donc 5 onglets :

- **Gestion** : Cet onglet permet à l'utilisateur d'obtenir la liste de ses sites web et l'adresse à laquelle il doit se connecter pour pouvoir gérer son site.
- **Création** : Cet onglet permet à l'utilisateur de créer son site. Il doit choisir entre les 3 noms de domaine disponibles et doit rentrer un sous-domaine uniquement avec des chiffres et des lettres. Le dynamisme de la page fait qu'il ne peut pas créer de site tant que son nom de domaine complet n'est pas valide.
- **Suppression** : Cet onglet permet à l'utilisateur de supprimer un de ses sites web. Il y trouvera la liste de ses sites et pourra sélectionner celui à supprimer.
- **Sources** : Cet onglet permet à l'utilisateur de récupérer ses sources, à savoir le contenu du dossier dans lequel son site web est (*/usr/share/nginx/html*). Il n'a qu'à cliquer sur le bouton "récupérer vos fichiers sources" et une archive contenant le contenu de ce dossier sera créée et téléchargée.
- **Documentation** : Cet onglet permet à l'utilisateur d'être guidé dans la création, la gestion, la suppression de son site.

Il y a également un menu déroulant avec des liens directs vers les URL des sites web de l'utilisateur et un autre menu pour se déconnecter et avoir les information sur le compte.

Nous avons ensuite besoin de PHP pour se connecter à la base de données utilisateur et ainsi avoir accès aux informations utilisateur (VLAN, login, etc). Mais également pour l'affichage dynamique des données de l'utilisateur, à savoir principalement la liste de ses sites web, mais également pour l'exécution des scripts depuis l'interface Web.

Cependant, les fonctionnalités "Sources" et "Gestion" ne sont pas fonctionnelles car elle nécessitent le bon fonctionnement du reverse ssh côté serveur.

2.3.2. Intégration

Nous avons donc deux scripts principaux qui sont exécutés directement depuis l'interface Web, à savoir le script de création et le script de suppression.

Actuellement, il faut savoir que cette interface Web est hébergée sur le serveur Web d'une machine virtuelle que nous avons créé en projet de Protocoles Réseaux Avancés. Cette machine virtuelle possède l'adresse IP publique 193.48.57.164 et se trouve sur un serveur de l'école. Elle est donc dans le réseau interne de Polytech et a accès à notre serveur via son adresse privée (172.26.64.13). L'interface Web communique donc avec notre serveur par un tunnel SSH via un échange de clé RSA. Lorsque l'utilisateur veut créer ou supprimer un site depuis l'interface, le code PHP établit une connexion SSH avec notre serveur physique Rift et exécute le script.

Dans la même idée, au chargement de la page, une connexion est établie et le code PHP exécute une requête sur la base de données qui contient les information sur les noms de domaine afin de récupérer tous les sites Web de l'utilisateur qui est connecté. On récupère alors cette liste en brut, chaque nom de domaine étant séparé par un saut de ligne. Grâce à la fonction PHP *explode*, on stocke ces données dans un tableau afin d'avoir accès à chacune d'elle précisément.

Pour le script de création, son exécution nécessite 3 paramètres : Login, VLAN et nom de domaine. Depuis l'interface, le Login et le VLAN sont récupérés sur la base de donnée utilisateur et le nom de domaine est donné lorsque l'utilisateur poste son formulaire. Pour la suppression, le script nécessite en paramètre le nom de domaine, il est également donné lorsque l'utilisateur clique sur le nom de domaine qu'il souhaite supprimer.

L'idée par la suite serait, pour simplifier le fonctionnement et éviter la connexion ssh entre l'interface Web et le serveur physique, d'héberger l'interface Web directement sur le serveur physique qui aurait une adresse routée.

3. Problèmes rencontrés

3.1. Architecture réseau

3.1.1. Bridge Docker personnalisé

Un de nos plus grand problème a été de mettre en place proprement le réseau 10.0.0.0/8 en adéquation avec Docker. Le bridge par défaut de Docker a rapidement montré ses limites pour notre utilisation. Le plus gros désavantage est que l'on ne peut pas définir une plage d'ip personnalisée pour les conteneurs. Nous nous sommes ensuite tournés vers l'outil de Docker pour créer notre propre réseau. Par défaut le réseau *bridge* est utilisé.

Cependant les fonctionnalités les plus intéressantes pour nous n'étaient toujours pas présentes. Le nouveau réseau était bien en 10.0.0.0/8, avec une plage d'ip personnalisable (e.g. *--ip-range=10.2.0.0/16*) mais il présentait les problèmes suivants :

- Impossible de définir une passerelle déjà existante
- Impossible d'utiliser un bridge déjà existant

En effet Docker crée forcément un nouveau bridge *docker0* qui prend en adresse ip la passerelle spécifiée. Or la notre existe déjà (*10.0.0.1*) et est essentielle pour converger les différents réseaux. Cela menait donc a un problème de taille qui nous a poussé à ne pas utiliser l'outil *docker network*, mais plutôt de créer un bridge avec les paramètres voulus, et de simplement indiquer au démon Docker de l'utiliser.

3.2. Vlans

Il avait été évoqué initialement de séparer le réseau en trois vlans pour distinguer les sites des associations, des étudiants et du personnel encadrant. Si le besoin d'isolation supplémentaire se manifeste effectivement à l'avenir, il sera possible d'isoler une plage entière d'adresses ip grâce à des règles *iptables*. Notre réseau est en effet découpé de telle sorte qu'une règle sur chaque VM éliminant les paquets venant des réseaux 10.X.0.0/16 (avec X non nul) aurait le même effet que de mettre en place des Vlans. Les machines ne pourrait recevoir de paquets que de la passerelle ou d'une machine d'administration située en 10.0.0.0/16. Nous n'avons pas mis la priorité sur cette fonctionnalité.

3.3. Accident sur notre hyperviseur

Une mauvaise manipulation lors de la mise en place de l'interface utilisateur nous a mené à devoir réinstaller complètement notre hyperviseur. Nous pouvions encore faire des sauvegardes des fichiers importants et créer une liste des paquets essentiels. Cette démarche nous a permis par la suite de créer une documentation complète pour mettre en place notre architecture. Cette documentation est disponible dans l'annexe de notre page wiki.

Cette mauvaise expérience nous a également permis de récupérer une version plus récente de Docker. Celle-ci nous débloquait sur un problème majeur : il était désormais possible d'utiliser l'option *--default-gateway* sur le démon. Ceci nous a grandement débloqué pour la suite du projet puisque les conteneurs avait maintenant accès à internet. Nous avons pu concevoir et tester nos conteneurs. La réinstallation de l'hyperviseur a été bien plus rapide que la première installation. Cela nous a fait revoir ce qui nous paraissait évident et cela a été globalement bénéfique.

3.4. Outils de test

Les outils employés pour tester le réseaux sont multiples : *ping, curl, wget, nc*. Cependant lors des tests du proxy inverse beaucoup de frustration est née du comportement de certains outils. Lorsque nous testions l'accessibilité d'un site, nous avons eu un problème par rapport à *curl* qui interprétait l'adresse ip comme était un nom de domaine à résoudre. Ensuite, par notre manque de connaissance du fonctionnement du cache du proxy polytech nous avons eu des résultats très étranges.

Une lecture des fichiers de log et une meilleure connaissances des outils employés aurait empeché cette perte de temps et cette frustration inutile.

3.5. Transfert de fichiers

Les contraintes sur cette problématique étaient fortes. Il faut en effet que l'utilisateur puisse se connecter en toute simplicité, par exemple avec FileZilla. Bien que nous ayons réussi une connexion en ssh inverse, le processus n'a pu être automatisé. Le problème réside dans l'écoute des ports sur notre hyperviseur, nous n'avons pas réussi à rendre accessible les ports souhaités.

Les droits d'accès ont aussi été un problème, puisqu'il fallait restreindre les accès des dossiers contenant les pages web, sans perturber le fonctionnement du serveur Nginx. Plusieurs solutions ont été explorer pour contourner le problème. Nous avons par exemple imaginer monter un volume partager dès la création du conteneur. Cette solution semble la plus réaliste mais ne résout pas les problèmes d'accès en ssh inverse.

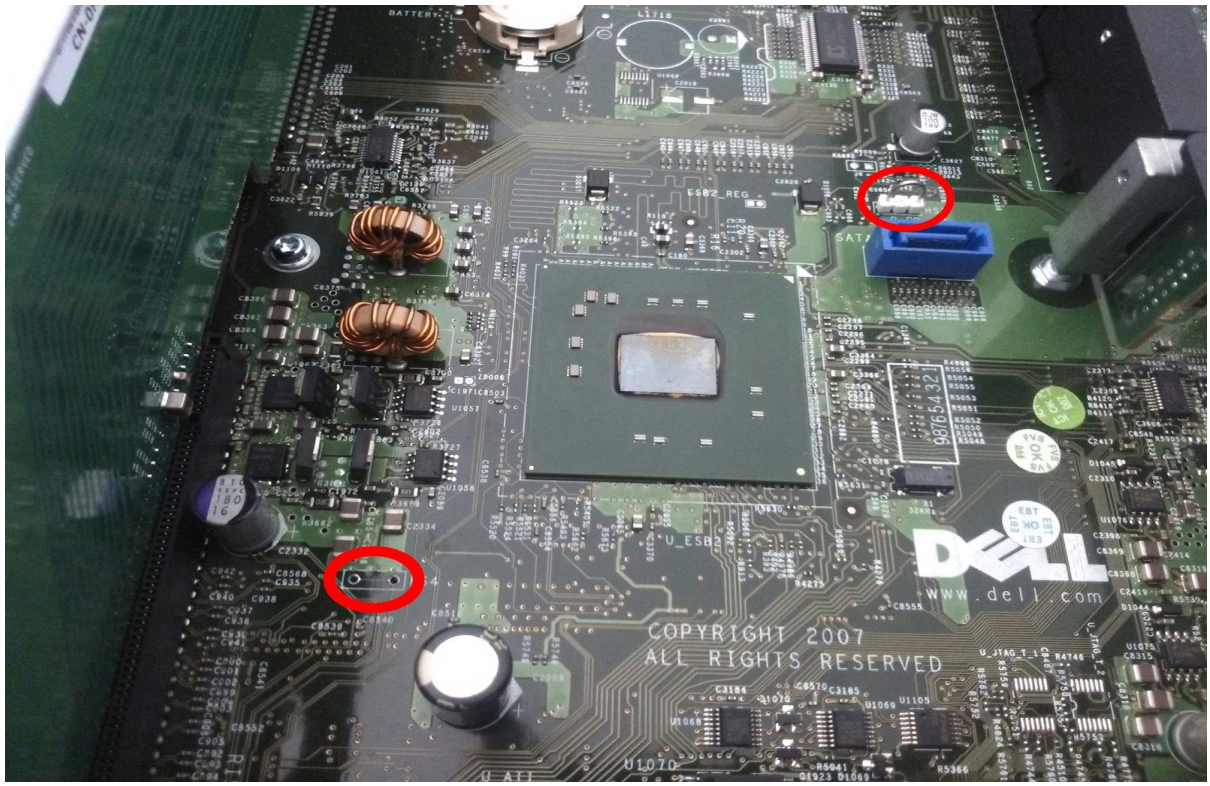
Conclusion

Ce projet a été une bonne occasion de découvrir en détail une technologie actuelle du milieu de la virtualisation : Docker. Le sujet pousse à découvrir en profondeur une multitude de domaines, tant en réseau qu'en système ou encore en web. Ceci est très stimulant et la prise de compétence est immense.

Nous n'avons malheureusement pas atteint tous les objectifs du cahier des charges. Néanmoins, le travail réalisé constitue une base solide pour achever le projet et nous espérons que la documentation et l'ensemble des sources fournis permettront de finir ce projet.

Annexe

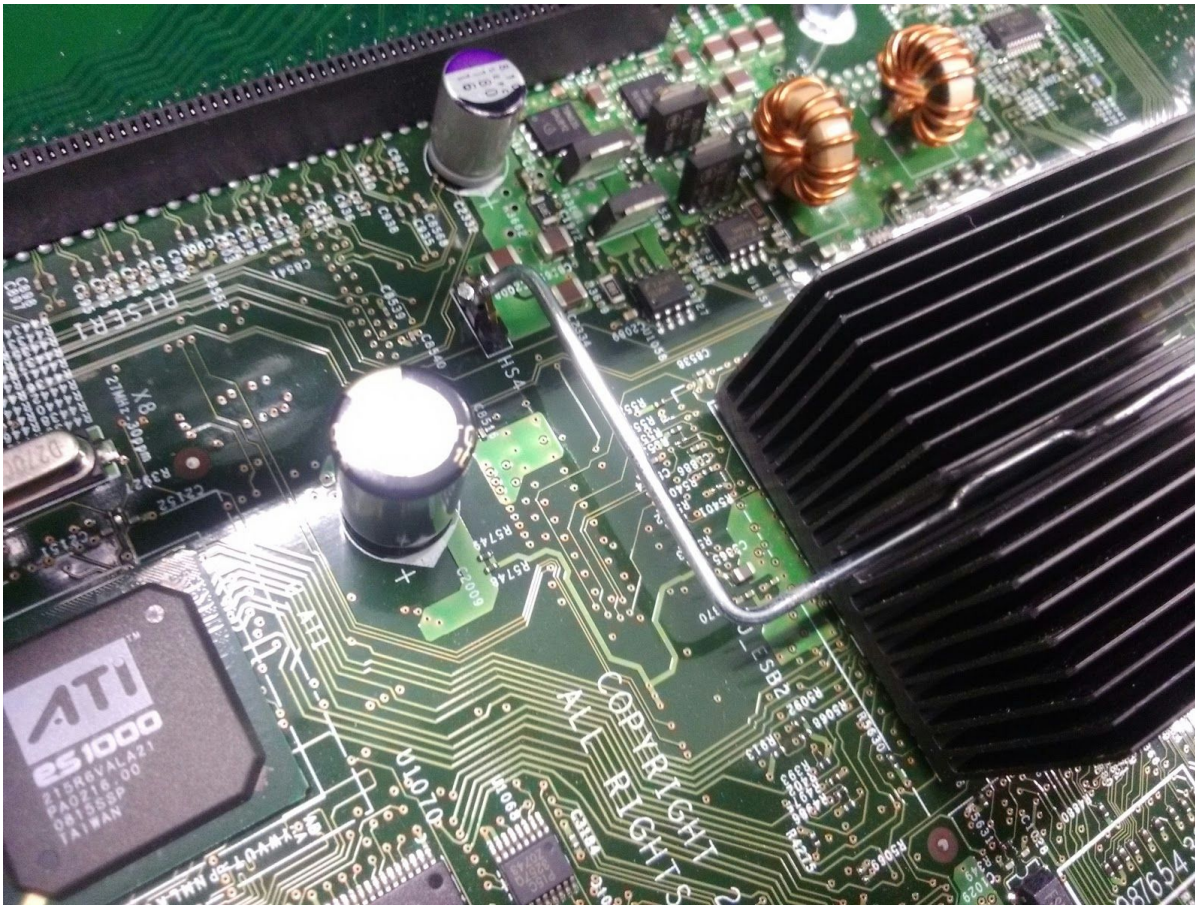
1. Un nouveau serveur



Nous avons reçu une machine DELL Poweredge 2950. Ce serveur était à notre entière disposition à la condition de réparer le dissipateur thermique d'un des composants.

En effet on peut voir sur l'image ci-dessus que l'une des anses aidant à maintenir le dissipateur en place s'est détachée. Après avoir évoqué plusieurs solutions, nous avons créé une anse similaire à celle déjà existante en soudant deux broches. Le résultat est des plus satisfaisants et le dissipateur est à nouveau maintenu correctement.

Nous avons procédé ensuite à la réinstallation complète de la machine sous Debian 8 (Jessie).


ATI
ES1000
COPYRIGHT
ALL RIGHTS


DELL