Rapport Technique Administration Réseaux - 2TL2 - Groupe 3

Louis Arys Geoffrey Brogniet Martin Perdaens
Jean-Michaël Tang

13 mars 2020

Table des matières

Responsable de mission et bilan (13 mars 2020)

Louis Arys

Bilan : Pour le moment, tout se déroule plutôt bien, nous n'avons pas rencontré de soucis lors de la prise de contact (c'est la première fois que nous travaillons tous ensemble). Au niveau de l'implémentation du projet, nous arrivons à bien nous répartir le travail, et en cas de soucis, on peut compter sur les autres pour nous donner un coup de main. Nous nous complétons bien au niveau de nos forces et nos faiblesses. Donc pour moi le bilan est très positif !

1 Introduction

Vous trouverez ci-après les différentes spécifications techniques des différentes technologies que nous utiliserons dans ce projet. Elles sont suceptibles d'évoluer ou d'être étoffées selon l'évolution de celui-ci.

2 Méthodologie

Nous utilisons plusieurs outils :

— `Trello` : permet de nous répartir le travail, de mettre les ressources importantes et les liens de manière claire dans un tableau de ressources, ainsi qu'une version synthétisée des consignes

— `Github` : permet de centraliser le code, d'éviter les conflits lorsque nous travaillons ensemble sur le code. Nous l'utilisons aussi pour faire le wiki de notre groupe.

— `Overleaf` : pour éditer des rapports à plusieurs en LaTeX

3 DNS

Nous avons décidé d'utiliser un container docker dans lequel tournera un serveur DNS Bind9. La raison de ce choix est la suivante : Bind9 est le serveur DNS le plus répandu sous Linux. Il y aura donc moyen d'accéder facilement à la documentation, et en cas de soucis, nous pourrons plus facilement trouver des solutions à nos problèmes. De plus, le fonctionnement de ce serveur est relativement simple à prendre en main.

Niveau d'avancement : A l'heure actuelle, nous avons implémenté un serveur Bind9 très basique et nous l'avons testé sur un réseau docker local avec 2 autres containers Ubuntu.

4 Database

Nous avons décidé d'utiliser une database de type mysql car nous avons tous un minimum de connaissance dans les bases de données relationnelles. De plus, une base de données relationnelle correspondra exactement aux besoins d'une entreprise comme WoodyToys, nécessitant de pouvoir gérer des clients, des stocks, ...

Pour le choix de la base de données en elle-même, nous avons arrêté notre choix sur **MariaDB**. Ce choix découle de deux choses : d'une, mysql est une base de données facile à prendre en main et intuitive. De deux, MariaDB est un fork libre de mysql, et est actuellement quasiment pareil à celui-ci. Cela nous permet d'utiliser des outils *Open-Source*, ce qui nous tient à coeur.

Niveau d'avancement : Nous n'avons pas encore implémenté la database.

5 Websites

Nous avons décidé de partir sur une architecture très classique. Pour le **Front-end**, nous sommes partis sur du pur *HTML/JS/CSS*. Cela permettra de très rapidement obtenir quelques pages afin que le client puisse utiliser les fonctionnalités du Back-End.

Pour le **Back-end**, nous nous sommes décidés d'implémenter un serveur Nodejs, pouvant être agrémenté si besoin est d'*EJS* pour faire du templating. Nous avons fait ce choix car nous avions de l'expérience dans l'implémentation de ce type de serveur. NodeJS est également efficace pour le routing et le déployement d'API.

Niveau d'avancement : Nous avons actuellement implémenté un serveur très basique renvoyant un hello-world au navigateur. Ce serveur a, bien entendu, été containerisé et testé.

6 Host

Nous avons choisi d'implémenter un container très basique, contenant un noyau Ubuntu, et permettant, actuellement, d'effectuer des ping sur d'autres machines, afin de pouvoir tester le serveur DNS. Ce container sera étendu au besoin pour nous permettre de tester nos différents services. À terme, ce container servira aussi à émuler les machines se trouvant chez WoodyToys.

Niveau d'avancement : terminé pour les besoins actuels du projet.

7 VoIP

Au terme de cette semaine de début de projet, nous ne pouvons nous exprimer sur le VoIP pour la raison simple que le cours abordant le sujet n'a pas encore eu lieu. Nous pouvons juste dire que nous le placerons sur un VPS à part car nous voulons séparer les différents services les uns des autres.

8 Firewall

Au terme de cette semaine, nous ne pouvons pas encore nous prononcer sur le firewall vu que la sécurité n'est pas encore de mise.

9 Problèmes rencontrés

9.1 Pour le DNS

Il a fallu comprendre comment marchait Bind9 ainsi que la manière de le containeriser proprement. Mais pour le moment tous les problèmes rencontrés ont été résolus.

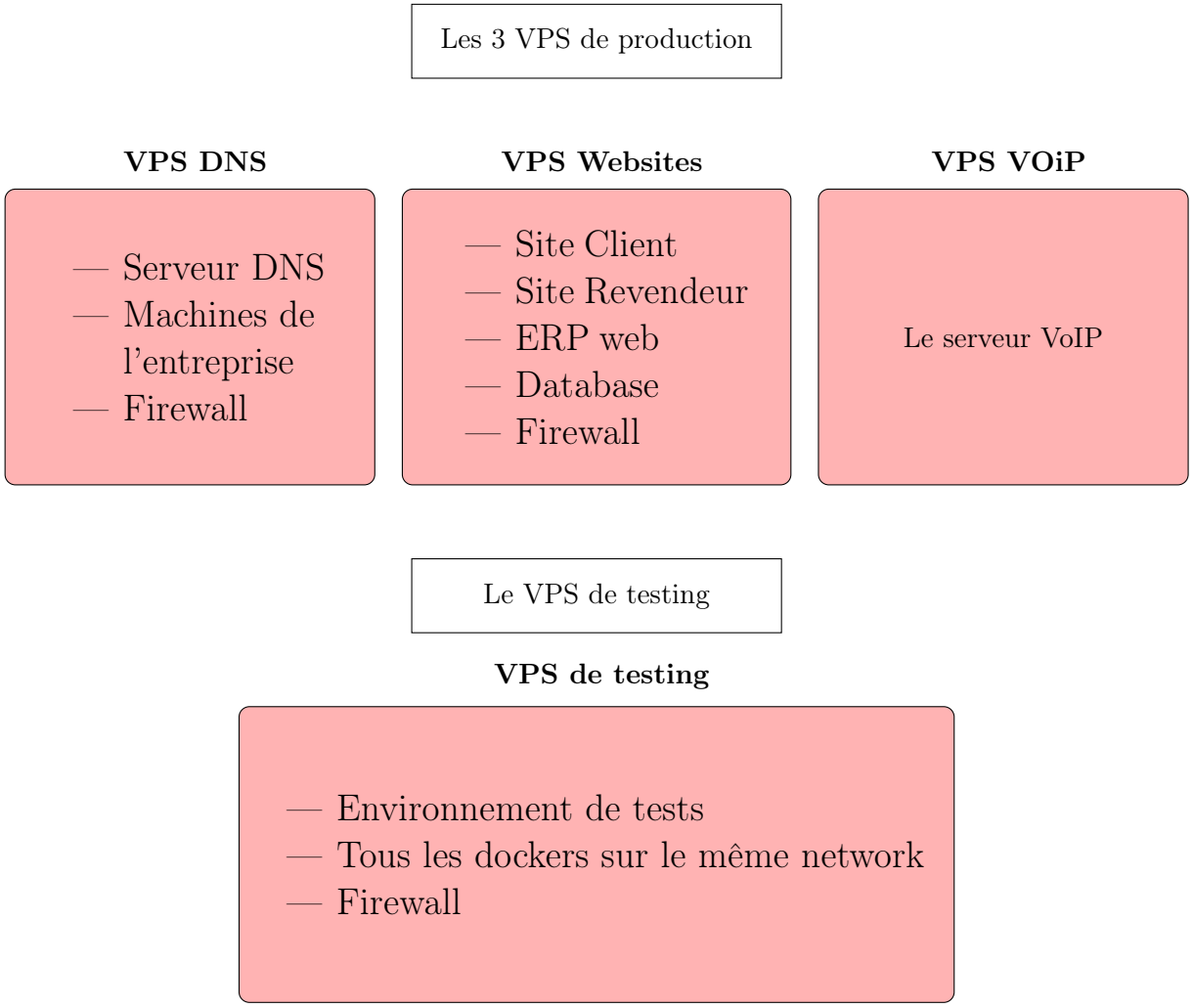
9.2 Pour le serveur NodeJS

Pas de problèmes rencontrés jusqu'à maintenant

9.3 Pour les outils d'organisation

Il a fallu un petit temps pour les prendre en mains pour ceux qui ne les avaient pas encore utilisés. Cela a été vrai surtout pour Overleaf(LaTeX) qui demande un petit temps pour le prendre en main.

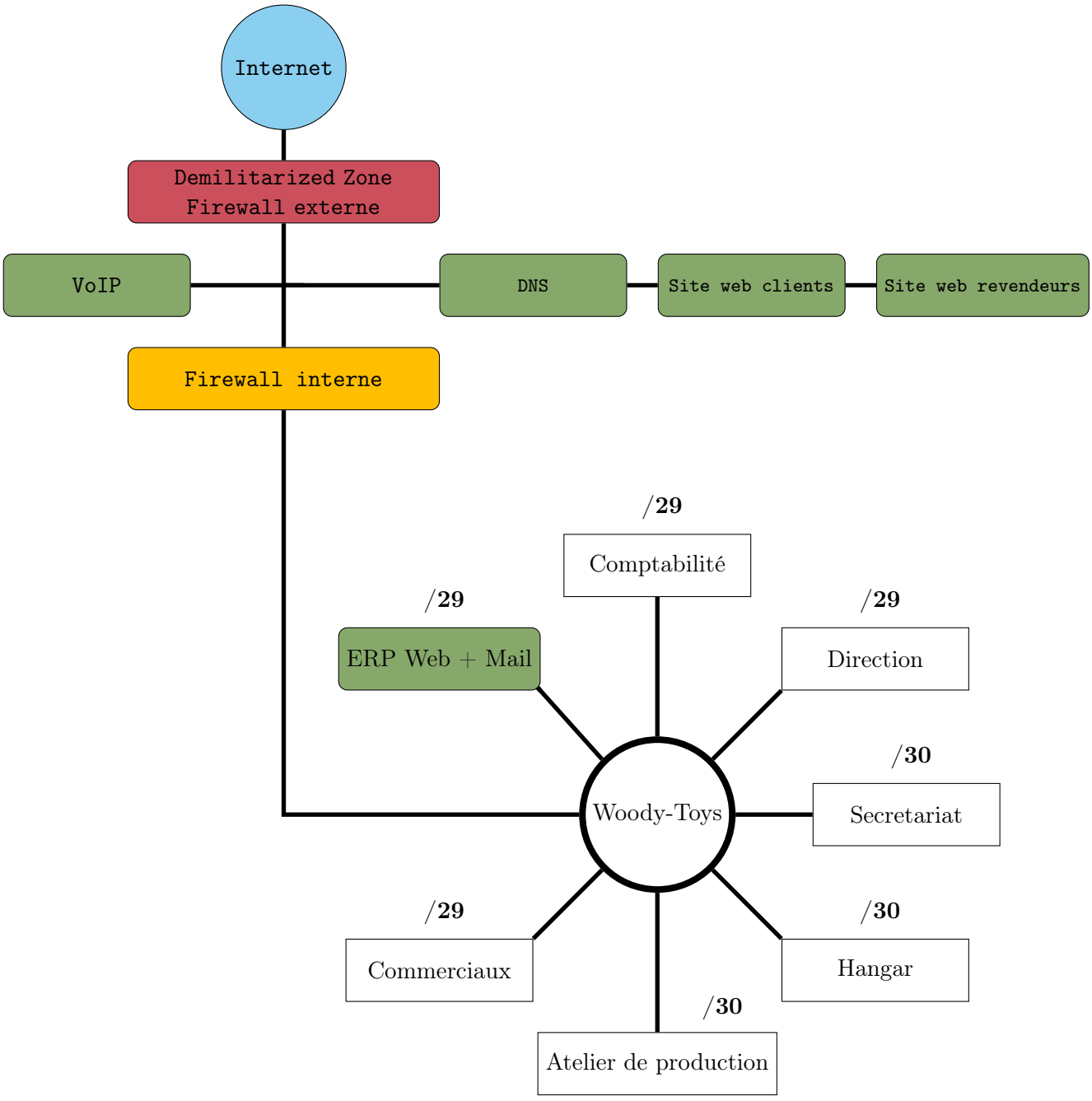
Schéma du prototype



Le schéma ci-dessus représente la manière dont seront répartis les différents services sur nos 4 VPS. Notre choix a été de séparer au maximum les différents services que nous devons implémenter : le DNS dans le premier, les sites webs et la database dans le deuxième, et la VoIP dans le troisième. Cela a pour but de simplifier notre architecture au maximum, et d'avoir un schéma simple à comprendre. De plus, cela permettra d'éviter la perte de plusieurs services d'un coup en cas d'attaques.

Le quatrième VPS vient du fait que nous soyons un groupe de 4, ce qui nous donnes l'avantage de pouvoir l'utiliser comme environnement de test, pour pouvoir déployer plus facilement les parties nouvellement implémentées sur les VPS de production.

Schéma de Woody-Toys



Notre architecture se base sur une architecture incluant deux firewalls et une DMZ(Demilitarised Zone). Le but de cette architecture est de séparer les ressources accessible de l'extérieur de celles ne devant être seulement disponible en interne. Cela permet de mettre deux couches de sécurité à notre architecture : un premier firewall permettant de se protéger des attaques de l'extérieur, mais laissant passer toutes les demandes/requêtes venant de l'internet. Un second, effectuant un filtrage stricte des requêtes allant vers l'intérieur, et empêchant ainsi n'importe qui de rentrer dans le

réseau interne de l'entreprise. C'est aussi pourquoi nous avons mis le serveur `ERP Web` à l'intérieur, car il n'est pas nécessaire qu'il soit accessible de l'extérieur.