RÉPUBLIQUE DU SÉNÉGAL

Un Peuple – Un But – Une Foi



MINISTERE DE L'ENSEIGNEMENT SUPERIEUR, DE LA RECHERCHE ET DE l'INNOVATION



UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

****

FACULTE DES SCIENCES ET TECHNIQUES

****

DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE SECTION INFORMATIQUE



MÉMOIRE DE MASTER INFORMATIQUE

Spécialité : Réseaux et Télécommunications

Option : Réseaux, Systèmes et Services

Sujet : Optimisation du système d'hébergement web de l'ADIE

Présenté et Soutenu publiquement par :

Mr CHEIKH MADICKE NDIAYE

Encadré par Dr. Chérif DIALLO

Le : 29 / 05 / 2021

Jury :

Président :	Pr. Cheikh Ahmadou Bamba GUEYE	Professeur Titulaire, UCAD
Examinateur :	Dr. Abdourahmane RAIMY	Maître de Conférences, UCAD
Examinateur :	Dr. Mohamed Ould DEYE	Maître Assistant, UCAD
Examinateur :	Pr. Ibrahima NIANG	Professeur Titulaire, UCAD

Année universitaire 2019/2020

Remerciements

Au terme de ce projet de fin d'étude, mes vifs remerciements sont dédiés à tous ceux qui ont contribué, directement ou indirectement à l'élaboration de ce projet de fin d'étude.

En Premier lieu, je remercie Monsieur Cheikh BAKHOUM directeur générale de l'ADIE, pour m'avoir donné l'opportunité d'effectuer ce stage à l'ADIE.

Je remercie également M. Moustapha KANE directeur des réseaux, systèmes et télécommunications de m'avoir accueilli dans sa direction.

Mes remerciements s'adressent également à M. Alassane NDOYE, chef de division plateformes et serveurs, de m'avoir accueilli dans sa division où j'ai eu à séjourner.

Je tiens aussi à remercier M. Alioune Badara NIMAGA, ingénieur systèmes, de m'avoir proposé ce sujet de mémoire et d'avoir bien voulu encadrer mes travaux.

Sans oublier M. Patrick Emanuel BADJI, responsable hébergement web pour son support et ses précieux conseils et M. Cheikh Sadibou THIAW, chef de division études et architectures pour ses conseils et sa disponibilité.

Je remercie en particulier mon Encadrant académique Pr Cherif DIALLO.

J'exprime ma gratitude à Pr Bamba GUEYE, Dr Fatoumata BALDE KASSE ainsi que tout le corps professoral de la section informatique.

Mon dernier mot s'adresse à tous les membres du jury pour l'honneur qu'ils me feront d'examiner ce travail.

Dédicaces

Pour leur amour et soutien inconditionnel, je dédie ce mémoire :

A mon cher père Amadou Ravane NDIAYE, ma chère mère Thiané NIANG.

A mes adorables frères et sœurs particulièrement Thiara NDIAYE pour leur amour et assistance.

A la famille DIOP qui m á chaleureusement supporté et encouragé tout au long de mon parcours en particuliers Tonton Maguette et Lamine DIOP (Pito).

A ma meilleure amie et complice Mame Fatou NIANG qui m á soutenu dans toute situation et toute sa famille.

A tous mes amis, camarades de promotion et collègues avec qui j ái travaillé et partagé des moments de bonheur.

Glossaire

API : Application Programming Interface

CLI : Command-line interface

CSS : Cascading Style Sheets

E/S : Entrée/Sortie

FTP : File Transfer Protocol

HA : High Availability

HPA : Horizontal Pod Autoscaler

HTTP : HyperText Transfer Protocol

HTTPS : HyperText Transfer Protocol Secure

LXC : Linux Containers

NFS : Network File System

NTP : (Network Time Protocol)

OS : Operating System

RAM : Random Access Memory

RDP : Remote Desktop Protocol

REST : Representational State Transfer

TCP : Transmission Control Protocol

TLS : Transport Layer Security

URL : Uniform Resource Locator

VM : Virtual Machine

VPS : Virtual Private Server

Table des matières

Remerciements 2
Dédicaces 3
Glossaire 4
Liste des Figures 8
Liste des Tableaux 10
Introduction Générale 11
1. Chapitre 1 : Présentation de l'ADIE 13
1.1. Historique 13
1.2 Missions 14
1.3 Réalisations et projets 15
1.3.1 Réalisations 15
1.3.2 Projets 17
1.4 Organisation 17
2 Chapitre 2 : Etat de l'art sur l'hébergement web 21
2.1 Introduction 21
2.2 Définitions 21
2.2.1 Les sites web 21
2.2.2 Les applications web 22
2.2.3 Les serveurs web 22
2.3 Types d'hébergements 26
2.3.1 Hébergement partagé 26
2.3.2 Hébergement dédié 26
2.3.3 Hébergement VPS 27
2.3.4 Hébergement cloud 28
2.4 Critères de performance 29
2.4.1 Les entrées-sorties 29
2.4.2 La mémoire vive 29
2.4.3 La puissance de calcul 29
2.4.4 La bande passante 29
2.4.5 La disponibilité 30

2.5 Conclusion 30
3 Chapitre 3 : Gestion de ressources dans l'hébergement 31
3.1 Introduction 31
3.2 Les environnements d'exécutions 31
3.2.1 Les serveurs physiques 31
3.2.2 La virtualisation 32
3.2.3 La conteneurisation 34
3.3 Les architectures de service 37
3.3.1 Architectures en mode tiers 37
3.3.2 Architectures en mode cluster 38
3.4 Les modèles de gestion 41
3.4.1 Gestion manuelle 42
3.4.2 Gestion automatisée 42
3.5 Conclusion 42
4 Chapitre 4 : Proposition de solution 43
4.1 Introduction 43
4.2 Etude de l'existant 43
4.2.1 Environnements partagés 44
4.2.2 Environnements dédiés 45
4.3 Solution proposée 46
4.3.1 Présentation de la solution 46
4.3.2 Description fonctionnelle 46
4.4 Conclusion 52
5 Chapitre 5 : Etude conceptuelle 54
5.1 Introduction 54
5.2 Choix technologique 54
5.2.1 Plateforme d'automatisation 54
5.2.2 Technologie de conteneurisation 56
5.2.3 Orchestrateur de conteneurs 59
5.3 Architectures proposées 64
5.3.1 Niveau applicatif 64

5.3.2 Niveau infrastructure 65
5.4 Conclusion 67
6 Chapitre 6 : Tests de validation 69
6.1 Introduction 69
6.2 Environnement de test 69
6.2.1 Serveur manager 69
6.2.2 Workers nodes 72
6.2.3 Serveur de stockage 73
6.3 Mise en œuvre et tests 73
6.3.1 Déploiement de Moodle 73
6.4 Conclusion 80
Bibliographie 82
Webographie 83

Liste des Figures

Figure 1-1: Aperçu des services de l'intranet 17
Figure 1-2: Organigramme de l'ADIE 20
Figure 2-1: Fonctionnement de l'hébergement web 25
Figure 2-2: Hébergement mutualisé 26
Figure 2-3: Hébergement dédié 27
Figure 2-4: Hébergement VPS 28
Figure 2-5: Hébergement cloud 28
Figure 3-1: Environnements d'exécution 31
Figure 3-2: Hyperviseur de type 1 32
Figure 3-3: Hyperviseur de type 2 : source 33
Figure 3-4: Architecture distribuée 38
Figure 3-5: Clusters haute disponibilité et de basculement 39
Figure 3-6: Cluster d'équilibrage de charge 40
Figure 3-7: Cluster HA et d'équilibrages de charge 40
Figure 3-8: Cluster de traitements distribués 41
Figure 4-1: Architecture du système d'hébergement Web de l'ADIE 44
Figure 4-2 : Architecture Fonctionnelle de notre solution 49
Figure 5-1: Architecture d'Ansible 55
Figure 5-2: Architecture de Docker 57
Figure 5-3: Cluster Kubernetes 60
Figure 5-4: Architecture Applicative de notre solution 65
Figure 5-5: Architecture Technique de la solution 66
Figure 6-1: Architecture de test 69
Figure 6-2: Composants du cluster 73
Figure 6-3: Ressources du cluster 74
Figure 6-4: Image de conteneur Moodle 74
Figure 6-5: Image Conteneur Mariadb 75
Figure 6-6: Espace d'administration 75
Figure 6-7: Exécution du programme de déploiement 76
Figure 6-8: Vue du déploiement depuis Rancher 76
Figure 6-9: Configuration de l'autoscaling
Figure 6-10: Ressources allouées 77
Figure 6-11: Taux de mémoire initialement utilisé 77
Figure 6-12: Interface d'accueil de la plateforme Moodle déployée 77
Figure 6-13: taux de mémoire utilisé lors de la connexion 78
Figure 6-14: Interface d'administration de la plateforme Moodle déployée 78
Figure 6-15: Taux de mémoire utilisé après login 78
Figure 6-16: Etat avant migration 79

Figure 6-17: Phase de migration des instances 79
Figure 6-18: Etat après migration 79
Figure 6-19: Sites Drupal déployés 80

Liste des Tableaux

Tableau 5-1: Comparaison entre différents outils d'automatisation 54
Tableau 5-2: Comparaison entre différents outils de conteneurisation..... 56
Tableau 5-3: Comparaison d'outils d'orchestration..... 59

Introduction Générale

Dans une dynamique d'accompagner la transformation digitale au sein de l'administration sénégalaise, l'Agence De l'Informatique de l'Etat a mis à disposition des structures étatiques un service d'hébergement web avec d'importantes quantités de ressources, mais avec l'explosion des applications métiers et grands public ces dernières années, dans un contexte où le télétravail est de plus en plus adopté et la dématérialisation des procédures administratives en plein expansion, le service fait face à une rapide augmentation de la demande qui a fini par traduit à de nouveaux besoins en terme de ressources.

Face à cette situation où les ressources jouent un rôle essentiel dans cette démarche d'accompagnement, une utilisation non optimale de ces dernières est notée. En effet, par anticipation aux pics de charges, des ressources sont réservées pour chaque application hébergée alors que les pics sont de nature évènementielle avec de rares périodes de hausses. De même, dans la prise en charge des exigences des sites web développé avec les dernières versions de CMS, de nouveaux environnements sont déployés alors que les ressources dédiées aux environnements déjà existant ne sont pas entièrement exploitées.

Ainsi, dans le cadre de notre mémoire l'agence nous a donnée comme sujet « Optimisation du système d'hébergement web de l'ADIE », qui a pour objectif de proposer une solution permettant de rationnaliser l'utilisation des ressources de calculs, ce qui nous a amené à repartir le sujet en cinq chapitres où dans un premier nous avons présenté l'historique, les missions, les réalisations, les projets et l'organisation de la structure d'accueil. Puis dans un deuxième chapitre nous avons fait un état de l'art sur l'hébergement pour mieux cerner le domaine du travail. Ensuite au troisième chapitre nous nous sommes focalisé sur les techniques de gestion des ressources dans l'hébergement web. Après cela, dans un quatrième chapitre nous sommes passé à la proposition de solution en faisant d'abord une étude de l'existant pour en ressortir ses limites, ce qui nous a permis de proposer

une solution. Au cinquième chapitre nous avons fait l'analyse conceptuelle, les choix des technologique et proposer des architectures. Pour finir, dans un sixième chapitre nous avons effectué des tests pour confirmer la validité et la fiabilité de la solution proposée.

1.Chapitre 1 : Présentation de l'ADIE

L'Agence De l'Informatique de l'Etat (ADIE) est une structure autonome chargée de mettre en œuvre la politique d'informatisation de l'Etat du Sénégal. Elle a pour mission principale de doter l'Administration d'un dispositif cohérent de traitement et de diffusion de l'information, répondant aux normes internationales en matière de qualité, de sécurité, de performance et de disponibilité. A travers elle, le Gouvernement du Sénégal a adopté une démarche visant à améliorer le service public en s'appuyant sur les Technologies de l'Information et de la Communication (TIC). Dans cette optique, l'ADIE développe un ensemble de produits et services pour contribuer à la bonne gouvernance par la promotion d'une Administration numérique efficace et efficiente au service du citoyen.

1.1. Historique

De la Section Informatique du Bureau Organisation et Méthodes (BOM) à l'Agence De l'Informatique de l'Etat, l'ADIE s'est toujours tournée vers l'avenir en mettant tout en œuvre pour que l'Administration s'adapte aux différentes mutations technologiques. Avec une vision futuriste, l'ADIE a progressivement introduit le numérique dans l'Administration Sénégalaise. L'évolution institutionnelle de l'ADIE est la suivante :

- Jusqu'en 1987 : la politique informatique de l'Etat du Sénégal était prise en charge par la Section Informatique du Bureau Organisation et Méthodes (BOM), notamment à travers la coordination du Comité National Informatique (CNI).
- Novembre 1987 : création de la Délégation à l'Informatique avec pour missions essentielles l'animation, la coordination et le contrôle de l'informatisation des administrations et des organes du secteur parapublic.
- Juin 2001 : création de la Direction Informatique de l'Etat (DIE) avec un

renforcement des missions et de nouvelles orientations, notamment : le renouvellement et le renforcement des équipements informatiques et réseaux de l'Administration ; le déploiement de réseaux locaux dans les bâtiments administratifs ; l'étude et le développement d'applications transversales.

- 2004 : création de l'Agence de l'Informatique de l'Etat (ADIE) pour donner plus d'impulsion, d'autorité et de moyens aux activités d'informatisation ; rendre un service de qualité aux usagers en apportant des solutions appropriées fondées sur la proximité et la réactivité ; rendre le secteur plus attentif à la notion de performance et de résultats.

1.2 Missions

Les missions dévolues à l'Agence sont :

- La modernisation de l'Administration Sénégalaise par la dématérialisation des procédures administratives ;
- La rationalisation des dépenses informatiques de l'Etat en mutualisant et en harmonisant les choix technologiques des services de l'Administration ;
- L'édification d'une infrastructure nationale de réseaux pour l'interconnexion des structures de l'Etat ;
- La mise à disposition d'un système d'information fiable pour un suivi efficace de l'action gouvernementale ;
- La coordination de la mise en place d'un cadre législatif et réglementaire propice au développement des technologies de l'information et de la communication.
- La réduction de la fracture numérique et l'exclusion sociale par la généralisation de l'accès aux tic.

Elle participe également à la définition de la stratégie de l'administration

électronique, communément dénommée « e-gouvernement », de l'Etat du Sénégal en vue :

- De doter l'Etat d'un système d'information et d'outils d'aide à la prise de décision
- De fournir aux citoyens et aux entreprises une interface décentralisée d'accès à l'Administration ;
- De pérenniser et sécuriser les archives de l'Etat en dotant celui-ci d'une mémoire électronique ;
- De définir des indicateurs de performances des systèmes d'information mis en place, et d'en assurer le suivi et l'évaluation ;
- D'évaluer l'impact des investissements réalisés dans le domaine de l'informatique ;
- De contribuer à la bonne gouvernance notamment par la promotion de la télé démocratie.

1.3 Réalisations et projets

1.3.1 Réalisations

- Réseaux et télécoms
  - 4500 KM de Fibre Optique
  - 1337 Bâtiments Connectés
  - 7000 Comptes Gouvernementaux créés
  - Plus de 60 BTS installées sur le territoire national

- Plateformes
  - 60 Sites Web Harmonisés
  - 30 Plateformes de gestion électronique déployés
  - 750 procédures administratives documentées pour faciliter l'accès à l'information administrative

- Plus de 25 procédures dématérialisées
- Système de gestion des RH déployé à la fonction publique
- Système de géolocalisation
- Plateforme Optica intégrée avec l'outil de gestion de projet
- Plateforme de vidéoconférence
- Plateforme archivage électronique
- Plateforme de télépaiement

➢ Confiance Numérique
- Mise en place d'une Infrastructure de Gestion des Clés et reconnaissance internationale
- Renforcement des équipements de sécurité
- Formation des agents en ISO 27001
- Participation aux sessions de la Commission Nationale de Cryptologie
- Réalisation de l'audit de sécurité de l'Intranet
- Elaboration d'une politique de sécurité de SI de l'Intranet
- Lancement d'un C-SIRT
- Etude de faisabilité pour la mise en place d'un SOC et recherche de financement

➢ Référentiels règlementaires et normatifs
- Lois sur les sociétés de l'information
- Charte de normalisation des sites web
- Homologation des plateformes de CMS pour l'hébergement
- Conduite du changement
- Mise en place équipe support aux utilisateurs
- Mise en place de Chargés de compte pour chaque Ministère pour le recueil des besoins et un suivi rapproché

- Formation en informatique de base
- Formation effectuée pour tous les services déployés

➢ Services de l'Intranet

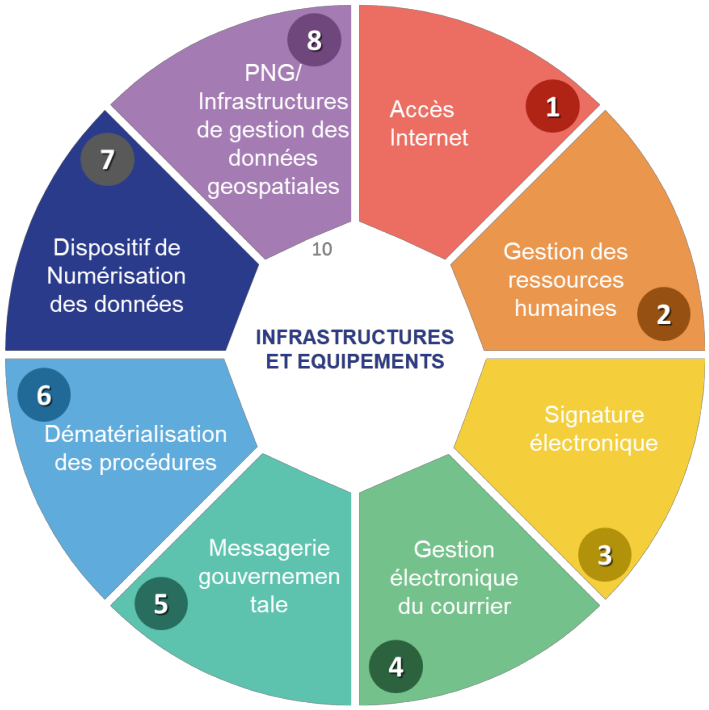


Figure 1-1: Aperçu des services de l'intranet

1.3.2 Projets

➢ Projet d'appui structurel à la stratégie d'aménagement numérique du territoire

➢ Smart Sénégal

➢ Dématérialisation

➢ Fichier Unifié des données du personnel de l'Etat

➢ Mon Service Public

1.4 Organisation

L'agence travaille avec des équipes et des dirigeants pour développer le potentiel collectif et individuel tout en améliorant les performances de l'Administration via des stratégies de leadership. Sur ce, l'ADIE est composé de 53 cadres (Réseaux et

Télécoms, Développeurs, énergies...), 15 managers et 65 techniciens (Administratifs, Maintenance, Réseaux et Télécoms, Développeurs, Energies...) répartis comme suit :

- La Direction générale assure la coordination et fixe les objectifs. Elle comprend : Le Directeur Général (DG), le Secrétaire Général (SG), les chargés de programmes et le coordonnateur du bureau des projets (PMO).
- L'Agence Comptable a pour mission de mobiliser les ressources de l'ADIE, de payer les dépenses, de conserver les valeurs et les fonds, de gérer la trésorerie, de tenir la comptabilité générale et d'établir les états financiers de synthèse.
- L'Audit et Contrôle de Gestion assure la performance de l'entreprise dans le respect des obligations légales et des procédures internes et participe au pilotage et à la prise de décision. Elle est chargée d'harmoniser les procédures comptables et financières et de superviser la clôture des comptes.
- La Cellule de passation des marchés est chargée de veiller à la qualité des dossiers de passation de marchés ainsi qu'au bon fonctionnement de la commission des marchés de l'ADIE.
- La Commission des marchés est chargée de l'exécution et du suivi des activités d'ouverture des plis, d'évaluation des offres et d'attribution provisoire des marchés l'ADIE conformément aux dispositions du code des marchés publics.
- La Direction de Solidarité Numérique (DSN) a pour mission générale de lutter contre la Fracture Numérique et l'exclusion sociale grâce à une intégration des TIC à l'école, à travers le recyclage utilitaire et durable des équipements informatiques et la formation des enseignants et des couches vulnérables.
- La Direction des Réseaux, des Systèmes et des Télécoms (DRST) est chargée de l'administration, de la maintenance des infrastructures systèmes,

réseaux et télécoms. Elle est également chargée de la gestion du parc informatique de l'Etat, du support aux utilisateurs, de la mise en production et de l'exploitation des systèmes d'information.

- La Direction des Services de l'Ingénierie (DSI) est chargée de la maîtrise d'ouvrage, de la maîtrise d'œuvre des projets et de la veille technologique. Elle assure l'accompagnement des structures administratives dans la mise en œuvre de leurs projets TIC et s'occupe du développement, de l'intégration et de la commercialisation des applications/produits/services.
- La Direction des Relations Extérieures, du Marketing, de la Formation et de la Communication (DRMFC) a pour mission de promouvoir les produits, les services, les réalisations et l'image de l'ADIE. Elle est chargée de renforcer les relations de l'Agence avec les structures administratives. Elle contribue également à la mise en œuvre de la politique de renforcement des capacités des agents de l'Etat. Elle est aussi en charge de définir et d'adopter une démarche pour la rétribution de services.
- La Direction de la Sécurité des Systèmes d'Information (DSSI) gère l'élaboration et la mise en œuvre des politiques de sécurité des systèmes d'information de l'Etat. Elle assure la définition des normes et règles de sécurité, la mise en place et l'administration des équipements de sécurité.
- La Direction Administrative et Financière (DAF) prend en charge la gestion comptable et financière, la gestion du personnel, l'intendance qui comprend notamment la gestion des locaux, des approvisionnements et des véhicules et la conduite des procédures administratives et juridiques.
- La Direction de la Logistique et de la Maintenance (DLM) est chargée de concevoir, d'organiser et de définir les meilleures stratégies de gestion des processus, du suivi des acquisitions et de leurs entreposages, de leur distribution. Elle assure le suivi des constructions et de l'entretien des ouvrages génie civil, la maintenance des véhicules et autres matériels roulants, l'entretien des matériels de bureau, l'approvisionnement en

carburant pour les véhicules et matériels pour l'énergie. Elle est chargée de la maintenance des équipements informatiques, de télécommunications et de production d'énergie (transformateurs, groupes électrogènes, onduleurs, régulateurs, climatiseurs ...), ainsi que des matériels et outils de maintenance du réseau de l'intranet.

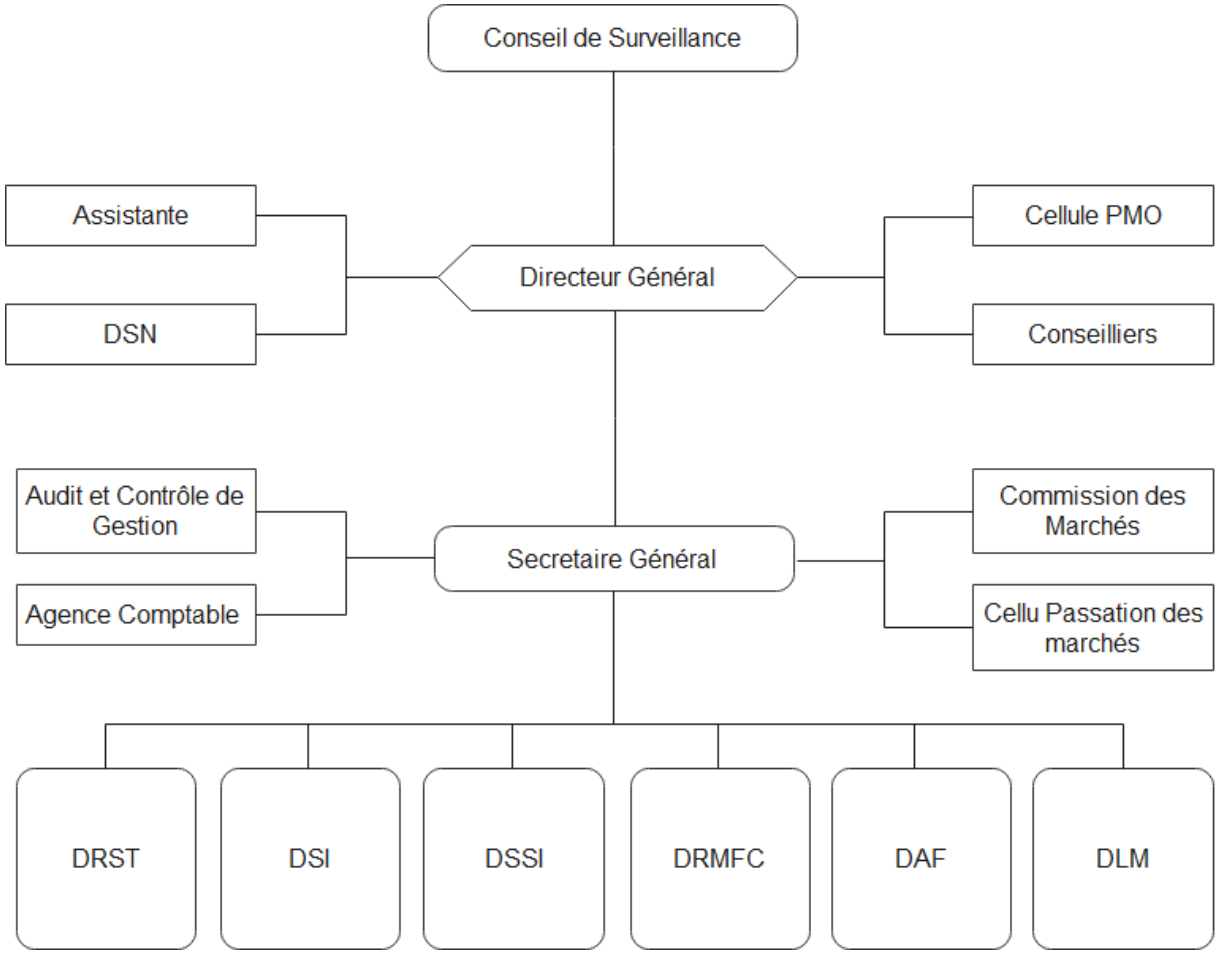


Figure 1-2: Organigramme de l'ADIE

2 Chapitre 2 : Etat de l'art sur l'hébergement web

2.1 Introduction

Dans ce chapitre, nous allons voir les généralités de l'hébergement web en le définissant ainsi que ses terminologies fondamentales. Puis faire un revu sur les différents types d'hébergement et finir par voir l'ensemble de critères qui font la performances d 'un 'hébergement web.

2.2 Définitions

L'hébergement web peut être défini comme l'acte de mettre sur un serveur un site web ou une application web pour permettre son accès à travers internet. L'entité qui fournit le service est appelée hébergeurs, il se charge de mettre à disposition un ensemble de ressources informatique, de maintenir le système où se repose le service, d'assurer la qualité de service à la clientèle et de garantir la sécurité des sites et applications web hébergés.

2.2.1 Les sites web

Un site web est un ensemble de documents html rattachés entre eux par le biais de liens hypertextes. Ils ont pour rôle principale de distribuer des informations et sont organisés en différentes catégories :

- Les sites vitrines sont utilisés pour présenter des informations sur une entreprise, des produits ou des services.
- Les sites e-commerce sont des boutiques en ligne permettant l'achat ou la vente de produits ou services via le web.
- Les blogs sont des sites personnels utilisés pour publier périodiquement et régulièrement des articles.
- Les médias sociaux sont des sites utilisés pour le partage d'informations

entre les différents utilisateurs.

2.2.2 Les applications web

Une applications web quant à elle est un site web avec une orientation métier, elle facilite le travail au quotidien de ses utilisateurs. Contrairement aux logiciels classiques, une application web n'a pas besoin d'être installée sur les terminaux des utilisateurs.

2.2.3 Les serveurs web

Un serveur web fait référence à des composants logiciels et matériels qui fonctionnent ensemble pour répondre à des requêtes du web. Au niveau des composants matériels, un serveur web est un ordinateur qui stocke les fichiers qui composent un site ou une application web et se charge de répondre aux demandes effectuées par un client distant généralement un navigateur web en s'appuyant sur les entrées-sorties (E/S), La mémoire vive (RAM) et la puissance de calcul (CPU).

- Les E/S représentent la vitesse à laquelle les fichiers sont transférées pour être exécutées ou traitées. La vitesse des entrées-sortie d'un hébergement déterminera la rapidité avec laquelle les données nécessaires à la création et à l'affichage du site Web, ainsi que les instructions d'assemblage, seront transmises à la RAM.
- La RAM se charge de stocker temporairement et de transmettre ces données. Cette mémoire est l'intermédiaire de ces processus car une grande partie du traitement de données requis pour exécuter un site Web est extrêmement répétitive. La RAM stocke donc ces données répétitives de manière à pouvoir les livrer à des vitesses très élevées lorsque nécessaire. La quantité de mémoire vive dont un hébergement dispose déterminera combien de ces processus peuvent être exécutés simultanément.

- Le CPU est la ressource qui fournit réellement la puissance du serveur pour traiter les données et exécuter les instructions qui lui sont transmises par la RAM. La quantité de puissance du processeur déterminera la vitesse à laquelle les données d'un site Web sont analysées, distribuées et affichées.

Au niveau des composants logiciels, un serveur web contient différents fragments qui contrôlent la façon dont les utilisateurs peuvent accéder aux fichiers hébergés. On trouvera à minima un serveur http qui est un logiciel qui comprend le protocole HTTP et les URL.

- Le protocole HTTP ou HyperText Transfer Protocol littéralement « protocole de transfert hypertexte » est un protocole de communication client-serveur développé pour le World Wide Web. Il peut fonctionner sur n'importe quelle connexion fiable, dans les faits on utilise le protocole TCP comme couche de transport. Un serveur HTTP utilise alors par défaut le port 80 et le port 443 pour HTTPS (avec S pour secured, soit « sécurisé ») qui est la variante sécurisée par l'usage des protocoles Transport Layer Security (TLS). Les clients HTTP les plus connus sont les navigateurs Web permettant à un utilisateur d'accéder à un serveur contenant les données. (Wikipédia).
- Une URL ou Uniform Resource Locator ou localisateur uniforme de ressource en français est une adresse d'une ressource donnée, unique sur le Web. En théorie, chaque URL valide pointe vers une ressource unique. Ces ressources peuvent être des pages HTML, des documents CSS, des images, etc. En pratique, il y a quelques exceptions : les URL peuvent pointer vers une ressource qui n'existe plus ou qui a été déplacée. La ressource représentée par l'URL et l'URL elle-même sont gérées par le serveur web. Une URL se compose de différents fragments dont certains sont obligatoires et d'autres optionnels.

- http:// correspond au protocole. Ce fragment indique au navigateur le protocole qui doit être utilisé pour récupérer le contenu. Généralement, ce protocole sera HTTP ou sa version sécurisée : HTTPS. Le « Web » fonctionne autour de ces deux protocoles mais le navigateur peut parfois gérer d'autres protocoles comme mailto: (qui permet d'ouvrir un client de messagerie électronique) ou ftp: qui permet de transférer des fichiers. Ne soyez pas surpris donc si vous rencontrez ces autres protocoles.

- www.exemple.com correspond au nom de domaine. Il indique le serveur web auquel le navigateur s'adresse pour échanger le contenu. À la place du nom de domaine, on peut utiliser une adresse IP, ce qui sera moins pratique (et qui est donc moins utilisé sur le Web).

- :80 correspond au port utilisé sur le serveur web. Il indique la « porte » technique à utiliser pour accéder aux ressources du serveur. Généralement, ce fragment est absent car le navigateur utilise les ports standards associés aux protocoles (80 pour HTTP, 443 pour HTTPS). Si le port utilisé par le serveur n'est pas celui par défaut, il faudra l'indiquer.

- /chemin/vers/monfichier.html est le chemin, sur le serveur web, vers la ressource. Aux débuts du Web, ce chemin correspondait souvent à un chemin « physique » existant sur le serveur. De nos jours, ce chemin n'est qu'une abstraction qui est gérée par le serveur web, il ne correspond plus à une réalité « physique ».

- ?clé1=valeur1&clé2=valeur2 sont des paramètres supplémentaires fournis au serveur web. Ces paramètres sont construits sous la forme d'une liste de paires de clé/valeur dont chaque élément est séparé par une esperluette (&). Le serveur web pourra utiliser ces paramètres pour effectuer des actions supplémentaires avant d'envoyer la ressource.

Chaque serveur web possède ses propres règles quant aux paramètres. Afin de les connaître, le mieux est de demander au propriétaire du serveur.

- #QuelquePartDansLeDocument correspond à une ancre, celle-ci désigne un endroit donné de la ressource. Une ancre représente, en quelque sorte, un marque-page à l'intérieur de la ressource. Ajouter une ancre à une URL permet au navigateur d'afficher la ressource à l'endroit de ce marque page. Pour un document HTML, par exemple, le navigateur défilera la page jusqu'au niveau de l'ancre. Pour un document audio ou vidéo, le navigateur ira se placer à l'instant représenté par l'ancre. On notera également que la partie de l'URL située après le # n'est jamais envoyée au serveur avec la requête.

Pour publier un site web, on distingue les serveurs web statiques et les serveurs web dynamiques.

- Le serveur web statique est composé d'un ordinateur (matériel) et d'un serveur HTTP (logiciel). Il envoie les fichiers hébergés tels quels vers le navigateur.
- Le serveur web dynamique quant à lui possède d'autres composants logiciels, tels qu'un serveur d'applications et une base de données. Il met à jour les fichiers hébergés avant de les envoyer au navigateur via HTTP.

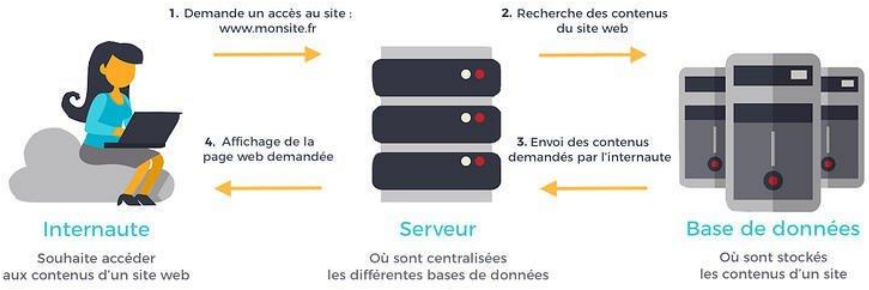


Figure 2-1: Fonctionnement de l'hébergement web [1]

2.3 Types d'hébergements

Pour héberger un site ou une application, différent type d'hébergement sont mise à la disposition du client. Ils sont définis en fonction du besoin du site ou de l'application hébergé pour son bon fonctionnement. On retrouve l'hébergement partagé, l'hébergement VPS, l'hébergement dédié et l'hébergement cloud.

2.3.1 Hébergement partagé

L'hébergement partagé, aussi appelé hébergement mutualisé est un modèle ou plusieurs sites sont hébergés sur un même serveur et se répartissent l'allocation des ressources. C'est une solution qui est particulièrement adaptée pour les site taille petite et moyenne comme les sites vitrines et les sites à faible trafic moyennement inférieur à 1000 visites par jour. Les aspects techniques et sécuritaires comme la maintenance et la sauvegarde sont en principe assurés par l'hébergeur. Par contre, l'hébergement mutualisé n'offre pas des performances adaptées pour les sites et application web à fort trafic, car les ressources du serveur étant partagées, les performances sont parfois affectées par la croissance des sites hébergés.



Figure 2-2: Hébergement mutualisé [2]

2.3.2 Hébergement dédié

L'hébergement dédié, plus précisément hébergement par serveur physique dédié. C'est une solution particulière, destinée aux applications et aux sites

hautement performants. Contrairement à l'hébergement mutualisé, ce l'hébergement dédié permet au site ou à l'application hébergé d'exploiter entièrement et de manière optimale la configuration du serveur et de ses ressources comme le CPU et la RAM. Il permet aussi aux clients un contrôle total sur l'architecture, ce qui signifie qu'ils peuvent personnaliser les systèmes de sécurité, les systèmes d'exploitation etc. C'est le modèle idéal pour les sites avec plus de 100000 visiteurs par mois. Les plans d'hébergement dédiés sont parmi les plus coûteux, compte tenu du matériel, des services gérés et du support technique de pointe.

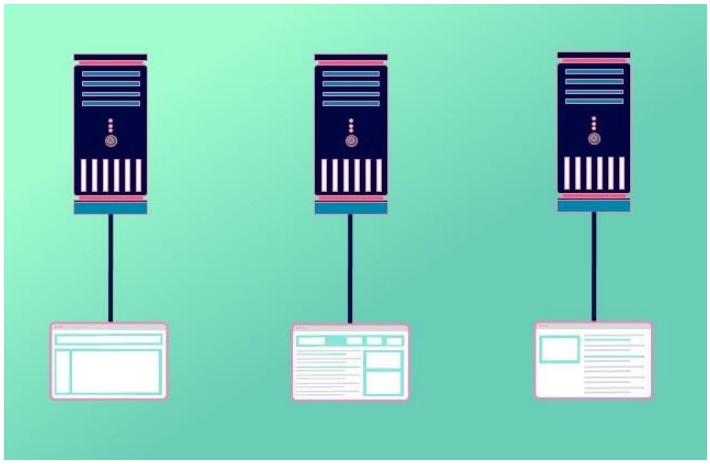


Figure 2-3: Hébergement dédié [3]

2.3.3 Hébergement VPS

L'hébergement VPS ou hébergement sur serveur privé virtuel comme son nom l'indique, il se repose sur des machines virtuelles qui agissent comme des serveurs dédiés indépendants. Il constitue un juste milieu entre l'hébergement partagé et l'hébergement dédié du fait qu'il divise un serveur physique en plusieurs autres serveurs virtuels, dans lesquels chaque site ou application est hébergé de manière dédiée. Chacun d'eux utilise un système d'exploitation qui lui est propre et est en mesure de redémarrer indépendamment des autres. Le VPS permet aux utilisateurs de gérer leur serveur depuis la racine de la machine. Il devient alors possible d'administrer son serveur de façon indépendante et sécurisée, de la même manière qu'un serveur dédié. Avec ce model il est possible d'attribuer au site ou à

l'application hébergé la quantité de ressources souhaités, à savoir du CPU processeur, de l'espace mémoire ou et de l'espace disque.

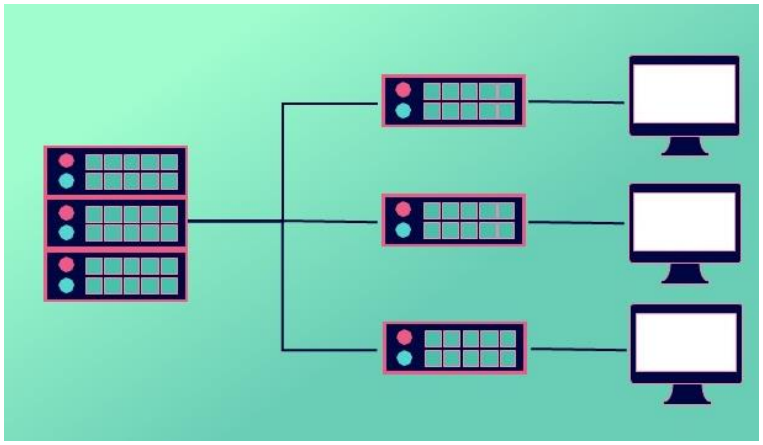


Figure 2-4: Hébergement VPS [4]

2.3.4 Hébergement cloud

L'hébergement cloud peut être considéré comme un hébergement VPS qui ne s'appuyant pas sur un seul serveur physique mais plutôt cluster de serveurs qui s'exécute dans le cloud. Chaque serveur du cluster stocke une copie à jour de votre site web. Lorsqu'un des serveurs est trop occupé, le cluster redirige automatiquement le trafic vers un autre. En conséquence, l'hébergement en nuage ne présente aucun temps d'arrêt, car un cluster du serveur peut toujours répondre aux demandes des visiteurs de votre site web.

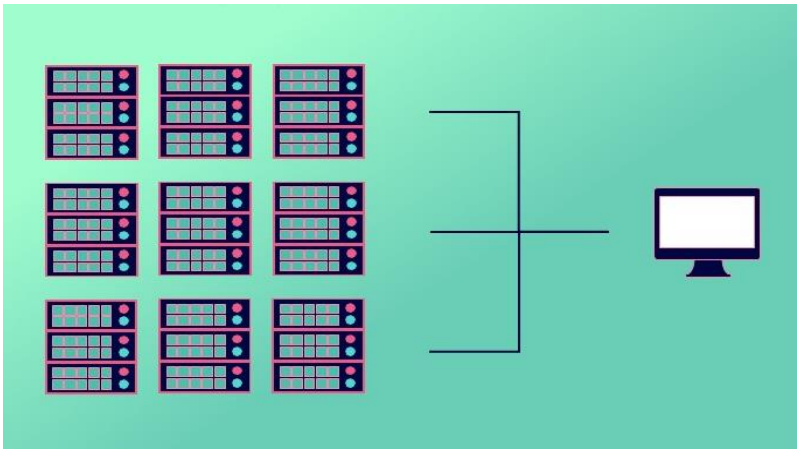


Figure 2-5: Hébergement cloud [5]

2.4 Critères de performance

La performance d'un hébergement web est définit principalement selon les critères suivant : les entrées-sorties E/S, la quantité de mémoire RAM, la charge CPU, la bande passante et le taux de disponibilité.

2.4.1 Les entrées-sorties

Les entrées-sorties E/S représentent la vitesse à laquelle les fichiers sont transférées pour être exécutées ou traitées. La vitesse des entrées-sortie d'un hébergement déterminera la rapidité avec laquelle les données nécessaires à la création et à l'affichage du site Web, ainsi que les instructions d'assemblage, seront transmises à la RAM.

2.4.2 La mémoire vive

La mémoire RAM ou mémoire vive se charge de stocker temporairement les données et les requêtes répétitives de manière à pouvoir les livrer à des vitesses très élevées lorsque nécessaire. La quantité de mémoire RAM dont un hébergement dispose déterminera le nombre de processus pouvant être exécutés simultanément.

2.4.3 La puissance de calcul

Le CPU est la ressource qui fournit la puissance de traitement des données et de l'exécution des instructions transmises par la RAM. Ce qui permet de déterminer la vitesse à laquelle les données d'un site Web seront analysées, distribuées et affichées. La charge CPU pour l'hébergement d'un site est déterminé selon le nombre de requête sur l'ensemble des pages du site et leur degré de complexité.

2.4.4 La bande passante

La bande passante est la quantité de données qui peut être envoyée sur le réseau en simultané. Également appelée transfert de données, la bande passante

indique la rapidité avec laquelle les visiteurs d'un site peuvent accéder aux informations, plus la bande passante est élevée, plus le nombre de personnes pouvant consulter le site simultanément est élevé. Il faut donc s'assurer qu'elle soit compatible avec les prévisions de trafic, en prenant en compte les éventuels pics de fréquentation et le type de contenus proposés.

2.4.5 La disponibilité

La disponibilité fait référence au temps pendant lequel un site web fonctionne activement pour les utilisateurs sur une période donnée. Exprimée en taux de disponibilité, elle donne le taux de réussite des connexions au site web. Une indisponibilité correspond à la situation où le navigateur affiche un message d'erreur lors de l'accès à une page du site web. Il s'agit en fait généralement d'une indisponibilité du serveur d'hébergement. Les problèmes de disponibilité se produisent le plus souvent à l'occasion de brutales montées en charge. Pour avoir le taux de disponibilité les fournisseurs détermine le ratio du temps disponible divisé par le temps total puis faire le calcul en incréments mensuels ou annuels. Même si 100% de disponibilité est l'objectif, l'industrie considère la disponibilité de 99,999% comme une haute disponibilité.

2.5 Conclusion

L'objectif de ce chapitre était de présenter de façon globale l'hébergement web, cependant nous avons défini ce qui est un site web, une application web et un serveur web et expliquer leurs fonctionnements. De même, nous avons présenté les différents type d'hébergement en exposant leurs avantages et leurs inconvénients et dans quels cas ils doivent chacun être utilisé. Par rapport aux critères de performances, nous vues qu'ils tournent aux tours des entrées-sorties, de la puissance de calcul, de la bande passante et de la disponibilité que nous avons chacun présenté.

3 Chapitre 3 : Gestion de ressources dans l'hébergement

3.1 Introduction

A ce chapitre nous allons faire une présentation sur la gestion des ressources dans l'hébergement web en nous accentuant sur les éléments fondamentaux qui définissent la façon d'utiliser ces dernières, notamment l'environnement d'exécution, l'architecture de déploiement et le modèle de gestion.

3.2 Les environnements d'exécutions

Les environnement d'exécution tournent aujourd'hui autour de trois : les serveurs physiques, les machines virtuelles et les conteneurs application qui sont représentés sur la figure suivante :

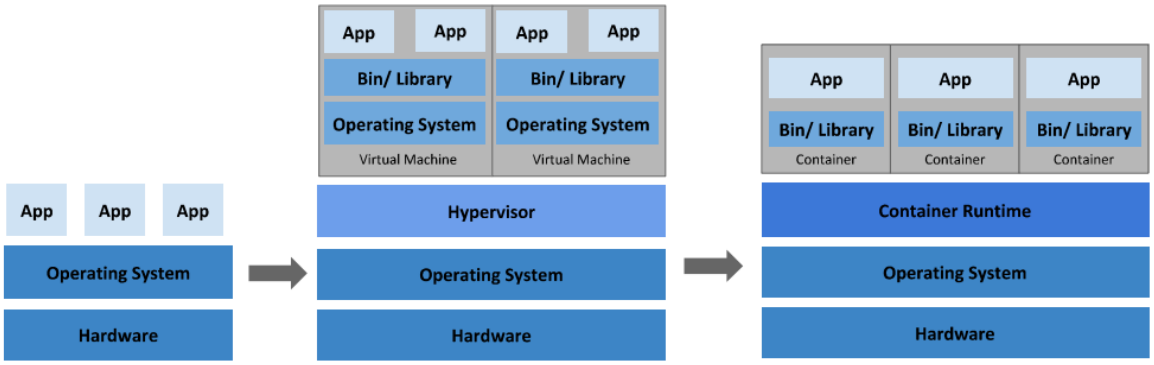


Figure 3-1: Environnements d'exécution [6]

3.2.1 Les serveurs physiques

Au début, les applications étaient exécutées sur des serveurs physiques. Il n'y avait pas de moyen de définir des limites de ressource pour les applications dans un serveur physique, ce qui provoquait des problèmes d'allocation de ressources. Par exemple, si plusieurs applications s'exécutent sur un serveur physique, il y a des cas où une application prendrait la plupart des ressources et, par conséquent les autres applications seraient sous-performances. Une solution serait d'exécuter chaque application sur un serveur physique différent. Mais cela n'a pas évolué car

les ressources étaient sous-utilisées et il était couteux pour les organisations de maintenir de nombreux serveurs physiques. Comme alternative, la virtualisation a été introduite.

3.2.2 La virtualisation

En tant que solution, la virtualisation, permet d'exécuter sur le processeur d'un seul serveur physique plusieurs machines virtuelles (VM). Chaque machine virtuelle dispose de son propre système d'exploitation (appelé système invité ou Guest OS) qui utilise les ressources matérielles d'un système hôte sous-jacent. Avec la virtualisation, la couche d'abstraction entre la machine physique et le système virtuel est créée par un hyperviseur. Un hyperviseur est un logiciel qui permet à plusieurs systèmes invités de fonctionner sur un seul système hôte. Il gère les ressources matérielles fournies par la machine physique, telles que le CPU, la RAM, l'espace disque et les périphériques, et les répartissent entre les machines virtuelles. Il existe deux types différents d'hyperviseurs, l'hyperviseur de type 1 et l'hyperviseur de type 2.

3.2.2.1 Les types d'hyperviseurs

L'hyperviseur de type 1 se basé directement sur la couche matérielle du serveur. Au démarrage d'une machine physique, ces hyperviseurs prennent directement le contrôle du matériel. Un maximum de ressources peut être alloué aux machines virtuelles.

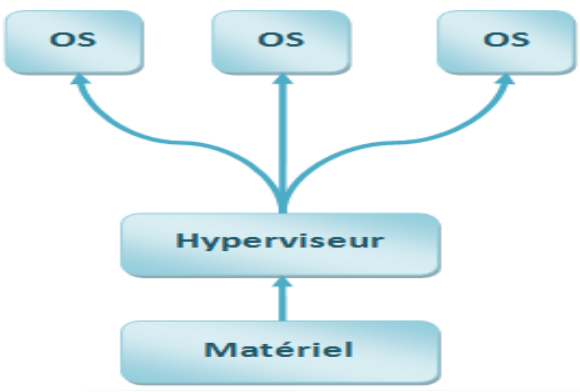


Figure 3-2: Hyperviseur de type 1 [7]

L'hyperviseur de type 2, par contre, fonctionnent dans le système d'exploitation de l'hôte et utilisent les pilotes périphériques fournis par le système pour accéder au matériel. Ce type d'hyperviseur ne peut pas fournir autant de ressources matérielles que les hyperviseurs de type 1 puisqu'ils sont installés sur un système d'exploitation, lui-même consommateur de ressources.

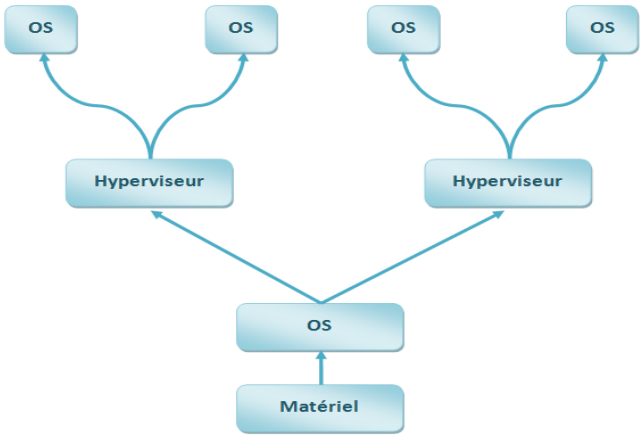


Figure 3-3: Hyperviseur de type 2 [8]

3.2.2.2 Les avantages de la virtualisation

La virtualisation permet d'isoler les applications entre des machines virtuelles et fournit un niveau de sécurité car les informations d'une application ne sont pas librement accessibles par une autre application. Elle permet aussi une meilleure utilisation des ressources dans un serveur physique et permet une meilleure évolutivité car une application peut être ajoutée et mise à jour facilement, réduit les couts matériels, et bien plus encore. Par contre avec la virtualisation, pour chaque VM on recrée intégralement un serveur exécutant tous les composants, y compris son propre système d'exploitation avec ses pilotes, ses fichiers binaires ou bibliothèques, en plus de matériel virtualisé. Ce qui amène à ce que la couche OS consomme à lui seul énormément de ressources alors qu'au final, ce qui nous intéresse c'est la ou les applications qui vont tourner dessus. Le fait d'accéder aussi aux ressources de façon virtuelle affaiblie les performances. D'où l'avènement de la conteneurisation pour palier à ces problèmes.

3.2.3 La conteneurisation

Avec la virtualisation l'isolation des environnements se fait au niveau matérielles (CPU/RAM/Disque) avec un accès virtuel aux ressources de l'hôte via un hyperviseur. Par contre avec la conteneurisation l'isolation se fait au niveau du système d'exploitation. Son principe est de faire tourner des applications isolées des uns aux autres dans des environnement partageants le noyau de l'hôte à partir duquel ils sont déployés. Un conteneur possède son propre système de fichiers, son processeur, sa mémoire, son espace de processus, etc. Contrairement une machine virtuelle, un conteneur n'inclus pas d'OS, puisse qu'il s'appuie sur les fonctionnalités de l'os de la machine Hôte. Ce qui le rend beaucoup plus léger que la VM. Par conséquent, il est donc plus facile à migrer et à télécharger, plus rapide à sauvegarder et à restaurer et permet aux serveurs d'héberger beaucoup plus d'application que s'il s'agissait d'un VM. La conteneurisation se repose principalement sur deux fonctionnalités qu'offre le noyau linux, les cgroups qui permettent de limiter l'utilisation des ressources et les namespaces qui permettent d'isoler les processus et une couche intermédiaire appelé runtime qui joue le rôle de contrôleur. Ainsi, il n'est pas nécessaire d'avoir recours à un hyperviseur pour gérer des conteneurs.

3.2.3.1 Les fonctionnalités de la conteneurisation

Les cgroups permettent de limiter la quantité de mémoire de processeur utilisée par un conteneur. Il existe plusieurs types de cgroups tels que :

- **CPUSET :** assigne des processeurs individuels et des nœuds de mémoire à des groupes de contrôle
- **CPU :** planifie un accès aux ressources du processeur
- **CPUACCT :** génère des rapports sur les ressources du processeur utilisées
- **DEVICES :** autorise ou refuse l'accès aux périphériques
- **NET_PRIO :** permet de définir la priorité du trafic réseau

- **MEMORY :** définit la limite d'utilisation de la mémoire
- **BLKIO :** limite de la vitesse E/S (lecture/écriture) sur périphériques de type bloc (ex : disque dur)
- **PID :** limite le nombre de processus

Les namespaces ou espaces de nom en français, donnent à chaque conteneur sa propre vue unique du système, limitant ainsi leur accès aux ressources système sans que le processus en cours ne soit au courant des limitations. Il existe différents types de namespaces :

- **PID :** fournit un ensemble indépendant d'identifiants de processus (PID). Il s'agit d'une structure hiérarchique dans laquelle le namespace parent peut afficher tous les PID des namespaces enfants. Lorsqu'un nouvel namespace est créé, le premier processus obtient le PID 1 et constitue une sorte de processus init de ce namespace. Cela signifie également que si on tue ce processus PID 1 alors on mettra immédiatement fin à tous les processus de son namespace PID et à tous ses descendants.
- **IPC :** empêche la communication avec les autres processus, plus simplement il interdit l'échange d'informations avec les autres processus.
- **NET :** crée une pile réseau entièrement nouvelle, y compris : un ensemble privé d'adresses IP, sa propre table de routage, liste de socket, table de suivi des connexions, pare-feu et autres ressources liées au réseau.
- **MOUNT :** monte un système de fichier propre au processus qui est différent du système de fichier de la machine hôte. Vous pouvez ainsi monter et démonter des systèmes de fichiers sans que cela n'affecte le système de fichiers hôte.
- **USER :** fournit à la fois l'isolation des privilèges et la séparation des identifications d'utilisateurs entre plusieurs ensembles. Il permet par exemple de donner un accès root dans le conteneur sans qu'il soit root sur la machine hôte.

- **UTS :** associe un nom d'hôte et de domaine au processus pour avoir son propre hostname.

Les contrôleurs gèrent un ensemble de fonctionnalités pour les conteneurs :

- Les interactions des conteneurs avec l'OS
- La sécurité par la gestion de privilèges et de ressources
- L'exécution et la suppression des conteneurs
- L'accessibilité des conteneurs à travers des API et CLI

Une image est un package qui inclut tout ce qui est nécessaire à l'exécution d'une application à savoir :

- Le code
- L'exécution
- Les variables d'environnement
- Les bibliothèques
- Les fichiers de configuration

3.2.3.2 Les avantages de la conteneurisation

Comparé à un système de virtualisation classique où chaque Machine Virtuelle (VM) exécute son propre système d'exploitation (OS) et peut augmenter l'utilisation de ressources de système, la technologie de conteneurs à plusieurs avantages :

- Une création et déploiement d'applications agiles avec une facilité et efficacité accrues de la création d'images de conteneurs par rapport à l'utilisation d'images de VM.
- Une consommation faible de ressources : Les containers partagent les ressources avec le système d'exploitation, ce qui permet de les rendre plus efficaces. Les containers permettent de diminuer le temps de démarrage, diminuer l'espace de stockage et la consommation du processeur, par

rapport au système traditionnel de machines virtuelles.

- Une accélération du déploiement des applications : Les containers disposent d'un environnement d'exécution minimal pour les applications, réduisant leur taille et permettant un déploiement plus rapide.
- Une amélioration de la portabilité des applications : La portabilité des containers permet d'éliminer un nombre important de bugs provoqués par les changements d'environnement d'exécution et les systèmes propriétaires.
- Une gestion des ressources simplifiée

Certes les conteneurs promettent évolutivité, flexibilité et facilité, mais ne conviennent pas à toutes les situations. A mesure qu'augmente le nombre de conteneurs, la gestion se complexifient ce qui peut entrainer des temps d'arrêt.

3.3 Les architectures de service

Dans l'hébergement web on distingue généralement l'architecture de type 1 tiers, l'architecture de type N tiers et l'architecture en cluster.

3.3.1 Architectures en mode tiers

3.3.1.1 Architectures 1-tiers

L'architecture 1-tiers ou mono-serveur est une architecture ou un site ou une application et l'ensemble de ses prérequis sont installé sur un seul serveur physique ou virtuel. Cette architecture est simple mais naturellement moins évolutive.

3.3.1.2 Architectures N-tiers

L'architecture N-tiers ou distribuée est une architecture qui suit généralement une conception à trois niveaux qui divise l'architecture en couches. Cela consiste à distribuer les services sur plusieurs serveurs. La conception permet l'évolutivité grâce à l'inclusion de serveurs supplémentaires à chacune de ces couches.

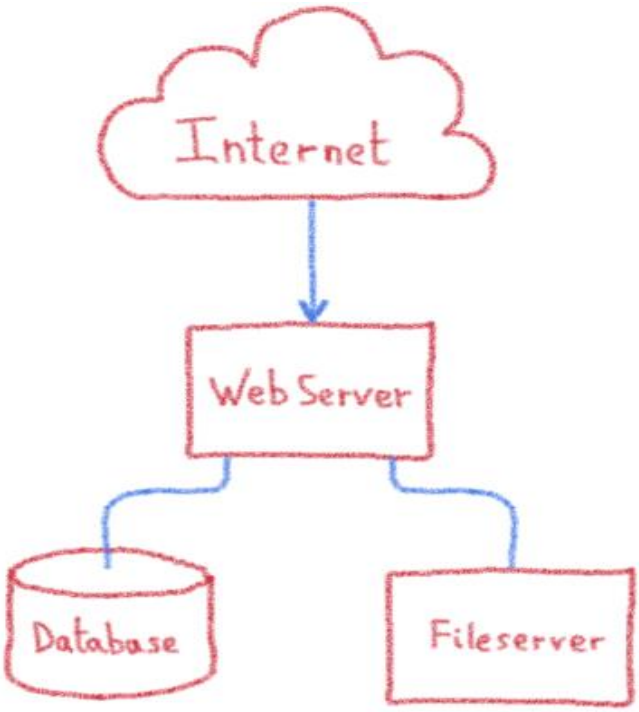


Figure 3-4: Architecture distribuée[9]

3.3.2 Architectures en mode cluster

Les clusters de serveurs sont généralement déployés pour améliorer les performances et la disponibilité par rapport à celles d'un seul ordinateur. Ainsi, les clusters permettent de profiter d'une ressource de traitement de données centralisée et aident à résoudre des opérations complexes plus efficacement avec une vitesse de traitement beaucoup plus rapide. Les programmes exécutés sur les clusters de serveurs reposent sur une API standard : Message Passing Interface. Cette API assure la communication entre les divers processus répartis sur les nœuds par le biais de messages. En cas de défaillance de l'un des serveurs, le logiciel de clustering isole le système en question. Lorsque les ressources sont partagées entre plusieurs tâches, si un serveur est surchargé, les tâches sont partagées avec un autre serveur. Les clusters peuvent être classés en deux catégories, les clusters ouverts où tous les nœuds sont accessibles directement via internet et les clusters fermés qui se trouvent derrière des passerelles pour offrir une meilleure sécurité.

3.3.2.1 Clusters de haute disponibilité

Clusters haute disponibilité (HA) et de basculement - Ces modèles de cluster créent la disponibilité des services et des ressources dans une méthode interrompue utilisant la redondance implicite du système. L'idée de base de cette forme de cluster est que si un nœud tombe en panne, les applications et les services peuvent être mis à la disposition d'autres nœuds. Ces types de clusters servent de base pour les missions critiques, les courriers électroniques, les fichiers et les serveurs d'applications.

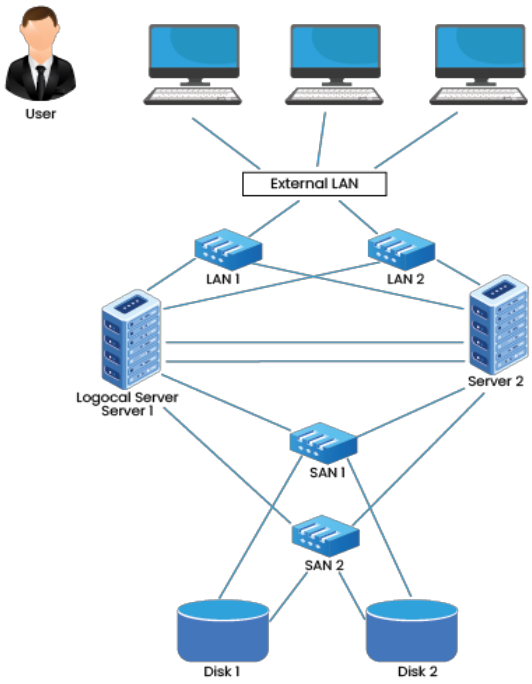


Figure 3-5: Clusters haute disponibilité et de basculement [10]

3.3.2.2 Clusters d'équilibrage de charges

Ces clusters sont utilisés pour répartir les charges de travail sur plusieurs ordinateurs. Dans ce modèle de cluster, tous les nœuds sont responsables du suivi des commandes, et si un nœud échoue, les demandes sont réparties entre tous les nœuds disponibles. Une telle solution est généralement utilisée sur les fermes de serveurs Web.

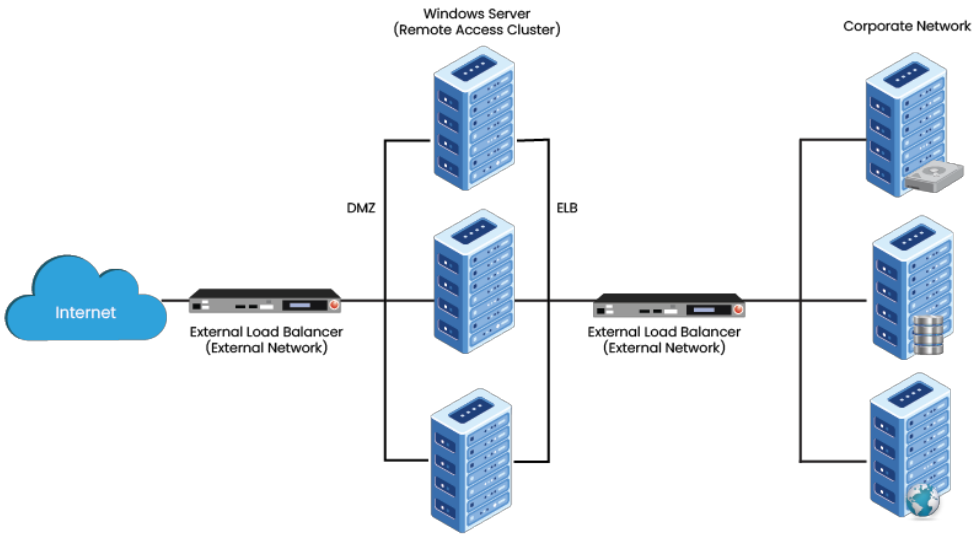


Figure 3-6: Cluster d'équilibrage de charge [11]

3.3.2.3 Clusters de haute disponibilité et d'équilibrage de charges

Clusters HA et d'équilibrage de charge - Ce modèle de cluster combine les fonctionnalités des deux clusters, ce qui se traduit par une disponibilité et une évolutivité accrues des services et des ressources. Ce type de cluster est couramment utilisé pour les serveurs de messagerie électronique, Web, d'actualités et FTP.

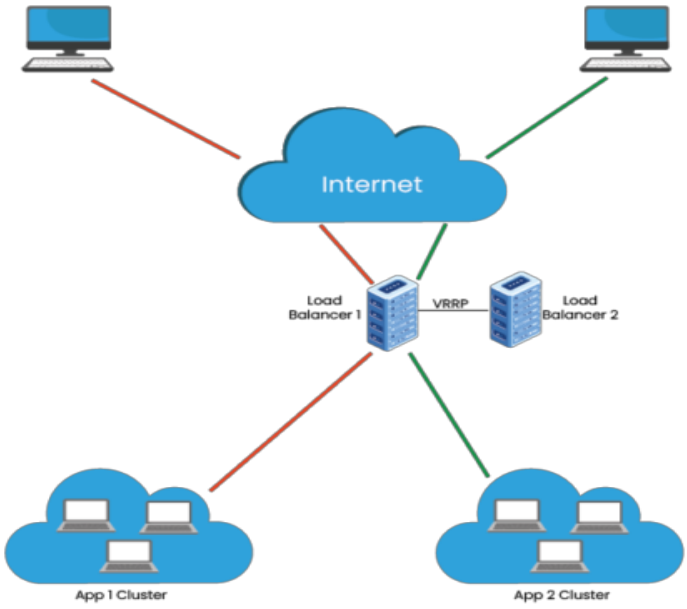


Figure 3-7: Cluster HA et d'équilibrages de charge [12]

3.3.2.4 Clusters de traitements distribués et parallèles

Clusters de traitement distribués et parallèles - Ce modèle de cluster améliore la disponibilité et les performances des applications qui ont de grandes tâches de calcul. Une grande tâche de calcul est divisée en tâches plus petites et répartie entre les stations. Ces clusters sont généralement utilisés pour le calcul scientifique ou l'analyse financière qui nécessitent une puissance de traitement élevée.

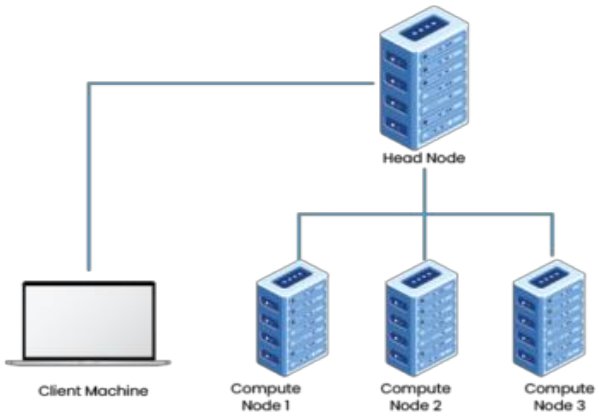


Figure 3-8: Cluster de traitements distribués [13]

3.4 Les modèles de gestion

Dans les environnements d'exécution les ressources ne sont pas illimitées et sont distingué les ressources de types compressibles qui sont "inépuisables" du moment qu'on a le temps pour attendre qu'ils soient de nouveau disponibles (ex : CPU, Réseau). En cas de surcharge, on peut toujours avoir un accès limité à ces ressources (Throttling), mais avec un risque de latence et de famine. On a aussi les ressources de types non compatibles qui sont limitées, si toutes les ressources disponibles sont consommées l'application ne pourra plus en avoir plus (ex : Mémoire, Disque.), En cas de surcharge, certains processus peuvent être tués (kill OOM), si le système a besoin de libérer des ressources. Ainsi différents modèles de gestion de ces ressources à travers les environnements sont définis, à savoir une gestion manuelle et une gestion automatisée.

3.4.1 Gestion manuelle

La gestion manuelle qui consiste à ajouter et à supprimer manuellement des ressources aux environnements ou à ajouter d'autres environnements en cas de besoins. Mais cette façon de gérer présente beaucoup de limites car pour ajouter ou supprimer des ressources une intervention humaine est nécessaire. Ce qui amené à faire recoure à l'automatisation.

3.4.2 Gestion automatisée

L'automatisation limite les interactions humaines ou en les remplaçant par des systèmes informatiques qui utilisent des logiciels pour effectuer les tâches, avec l'objectif de réduire les coûts, la complexité et le risque d'erreur. En général, l'automatisation concerne une tâche unique. C'est ce qui la différencie de l'orchestration, qui décrit comment automatiser un processus ou un workflow constitué de nombreuses étapes réalisées sur plusieurs systèmes disparates. Dans ce cas l'orchestration est utilisés pour automatiser les processus de gestion des ressources avec de l'autoscaling horizontal ou verticale. La différence entre un hyperviseur et un orchestrateur, c'est qu'un hyperviseur permet sur une machine d'avoir plusieurs systèmes (VM) alors pour qu'un orchestrateur permet de grouper plusieurs machines en un système logique.

3.5 Conclusion

Ce chapitre nous a permis de cerner la gestion des ressources dans l'hébergement web en passant par les environnements d'exécutions où nous avons présenté l'évolution des environnements physiques jusqu'aux conteneurs applicatifs, par les architectures de services en présentant celles en mode tiers et celles en mode clusters et finir par les différents modèles de gestion avec la présentation de la gestion manuelle des environnements et la gestion automatisé qui a facilité les choses surtout avec l'orchestration.

4 Chapitre 4 : Proposition de solution

4.1 Introduction

Dans ce chapitre nous allons faire une étude sur le système existant pour déterminer les causes qui sont à l'origine de l'adoption de ces techniques qui provoquent une utilisation non optimale des ressources et de proposer une solution qui répond aux attentes.

4.2 Etude de l'existant

Le système existant permet d'offrir deux types services d'hébergement. Un service basé sur le type mutualisé où les ressources d'un même serveur sont partagées entre plusieurs sites web et un service basé sur le type VPS où pour chaque application les ressources d'un serveur lui sont entièrement dédiées. Le système se repose sur une architecture à trois zones dont :

- Une zone d'administration qui n'est accessible qu'à partir du réseau privé pour permettre le déploiement des sites et des applications et l'administration des environnements en toute sécurité.
- Une zone d'exploitation qui permet un accès public en passant par un proxy qui se charge de relayer les requêtes entrantes vers les serveurs concernés.
- Une zone cœur qui loge les serveurs où sont déployer les sites et les applications hébergés.

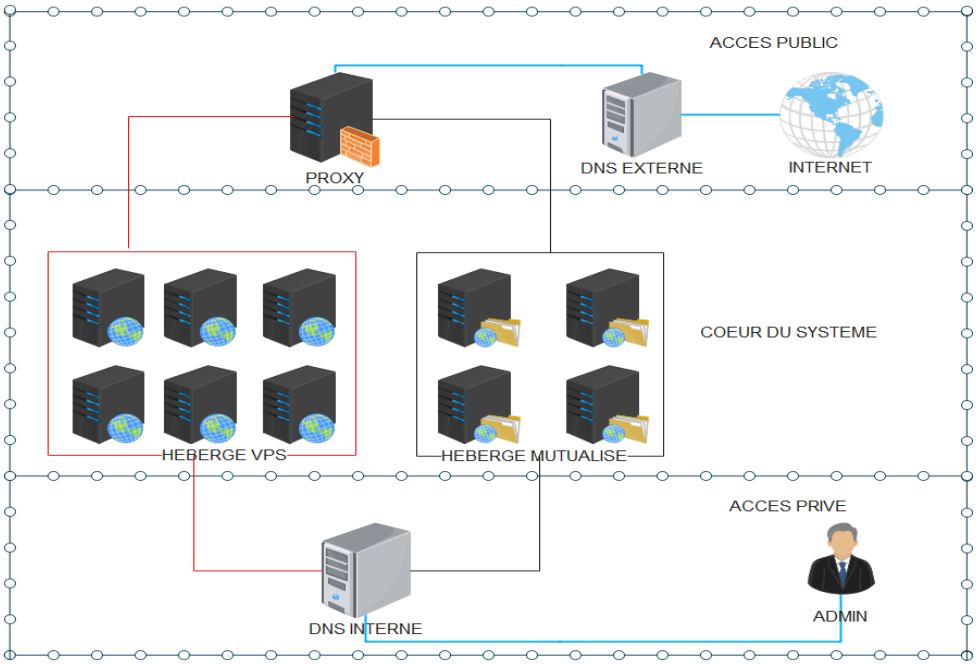


Figure 4-1: Architecture du système d'hébergement Web de l'ADIE

4.2.1 Environnements partagés

Les environnements partagés tournent sur des machines virtuelles qui intègrent chacune un service web, un service de transfert de fichiers et un service de base de données et dispose d'une quantité importante de ressources (Stockage, RAM et CPU) dans le but d'héberger un très grand nombre de sites web. Par contre le couplage fort qui existe entre les sites hébergés et le système d'exploitation de la machine hôte, limite la possibilité de prendre en charge diverse dépendances sur un même environnement, cela dû au fait que certains processus ne peuvent cohabiter ensemble. Ce qui a amené à déployer plusieurs environnements pour prendre en charge toutes les exigences en terme de dépendances des sites héberger.

Cet hébergement se fait en trois étapes :

- La création d'un espace d'hébergement, qui consiste à choisir le serveur qui répond au mieux aux exigences du site à héberger et la création d'un compte utilisateur, d'une base de données et d'un répertoire qui loge le site. Ce répertoire sera relié au répertoire personnel de l'utilisateur pour permettre avec les paramètres du compte utilisateur d'y accéder via FTP. Ensuite, un

VIRTUALHOST sera créé pour y renseigner le nom de domaine et le chemin absolu du répertoire créé pour la redirection des requêtes.

- Une fois l'espace d'hébergement disponible, le nom de domaine sera déclaré au niveau du DNS INTERNE pour permettre au client de déployer son site et à l'équipe de sécurité d'effectuer les tests de vulnérabilités.
- Après les tests, si les exigences en terme de sécurité sont respectées, la publication du site est autorisée et le nom de domaine sera cette fois déclaré au niveau du DNS EXTERNE avec adresse IP public.

4.2.2 Environnements dédiés

Les environnements dédiés sont aussi des machines virtuelles mais pour l'augmentation et la réduction de ressources (scalabilité verticale) et pour la réplication des environnements (scalabilité horizontale) les hyperviseurs utilisés ne permettent pas d'effectuer ces opérations à chaud. Ce qui entraine des arrêts de services qui peuvent impacter sur la disponibilité des applications. Par ailleurs, le fait de se baser aussi sur une architecture de service 1-tiers, limite les possibilités de répliquer des applications, ce qui risque provoquer des disfonctionnements. Pour éviter cela, chaque environnement est dimensionné selon les besoins de l'application hébergée en tenant compte les ressources nécessaires pour supporter les charges durant les périodes de pics, plus des exigences de la couche OS pour le bon fonctionnement du système. Ce qui est à la base du surdimensionnement des environnements.

Le processus d'hébergement d'application est le suivant :

- La création d'un environnement de déploiement, qui consiste à créer un serveur virtuel en y installant les prérequis nécessaires.
- Le déploiement de l'application via le protocole SSH ou RDP par le client.
- Le scan de vulnérabilités et la transmission du rapport au client pour d'éventuels correctifs.

- La publication de l'application en déclarant le nom de domaine au niveau du DNS EXTERNE.

4.3 Solution proposée

4.3.1 Présentation de la solution

Sous la lumière des études faites sur l'existant, nous avons compris que d'une manière générale les limites exposées se résident dans l'utilisation de machines virtuelles comme environnements d'hébergement. Ce qui n'offre pas une grande flexibilité pour l'adaptabilité des environnements en fonction des besoins et la portabilité des sites et applications hébergés. De même l'architecture de service 1-tiers sur laquelle on se repose le système, ne permet pas l'extensibilité horizontale des environnements. Ce qui nous amène à proposer comme solution **un système d'hébergement web automatisé, basé sur des conteneurs**. Cette solution a pour objectifs :

- L'utilisation de conteneurs va nous permettre de supprimer le couplage fort qui existe entre les systèmes d'exploitation des serveurs hôtes et les applications qui y sont hébergés, ainsi nous pourrons déployer des sites et applications sur un même serveur quel que soit les dépendances et de profité de la flexibilité et la portabilité des conteneurs pour faciliter la gestion.
- L'automatisation quant à elle va nous permettre de prendre une approche de dimensionnement automatique basée sur des tendances de trafic réelles qui ne met des ressources en service que si elles sont nécessaires et cela nous évitera les surdimensionnements et de faciliter les taches d'administration.

4.3.2 Description fonctionnelle

La description fonctionnelle va nous permettre de mieux comprendre comment fonctionne la solution proposée. Elle va se faire en présentant les différents acteurs et leurs rôles, les données manipulées, les traitements à effectuer, les composants techniques et les fonctionnalités du système.

4.3.2.1 Acteurs et rôles

Un acteur représente une entité jouant un rôle défini par une ou plusieurs fonctions, cette entité est externe au système mais elle est en interaction avec celui-ci. Les acteurs pouvant interagir avec notre système sont les suivants :

- Les webmasters, qui se chargent de déployer et de maintenir les applications et sites web au sein du système.
- L'administrateur, qui a pour rôle de définir la politique de gestion des applications et sites web déployés pour garantir leur performance et leur haute disponibilité et pour rationaliser l'allocation des ressources.
- Les clients finaux, ils sont les exploitant des applications et des sites hébergés à partir d'internet.

4.3.2.2 Données manipulées

Ce sont les données utilisées par le système pour conteneuriser, héberger et gérer des applications et des sites web concernés. Elles sont pour la plupart des fichiers descriptifs. Nous avons :

- Les fichiers d'images, où sont décrites les procédures à suivre pour créer les images d'une application ou d'un site web.
- Les fichiers de déploiement qui décrivent les procédures de déploiements des applications et des sites web à héberger ainsi que les règles de gestion de ces derniers.
- Les images d'applications et de sites qui seront utilisées pour instancier des applications et des sites web qui seront exécutés sous forme de conteneurs.
- Les conteneurs seront utilisés pour gérer les ressources

4.3.2.3 Traitements à effectuer

Ceux sont les actions à réaliser pour atteindre ces objectifs. Elles sont regroupées en trois grandes fonctionnalités qui sont la conteneurisation, le déploiement et

l'administration :

- La conteneurisation qui permet de packager les sites et les applications à héberger avec leurs dépendances et de leurs images qui seront utilisées dans le déploiement. Elle se fait en trois phases :
    - **Phase 1 :** Description des étapes à suivre et mise à disposition des composants nécessaires pour la construire de l'images de l'application ou du site web à héberger.
    - **Phase 2 :** Création de l'image en suivant les instructions décrites précédemment dans un fichier
    - **Phase 3 :** Mise en stock dans un registre l'image créée d'images
- Le déploiement qui permet d'exécuter les sites et les applications et de les attribuer les ressources minimales pour leur bon fonctionnement. Il se fait en trois étapes :
    - **Phase 1 :** Description de la procédure de déploiement
    - **Phase 2 :** Instanciation de l'application à partir de son image
    - **Phase 3 :** Exécution de l'application dans un environnement propice

- L'administration qui permet de maintenir les entités déployés dans les états souhaités et de les fournir les ressources nécessaires en cas de besoins. Elle se fait en trois étapes :
    - **Phase 1 :** Description des règles à suivre sur la façon d'allouer des ressources à l'application
    - **Phase 2 :** Application des règles soumises
    - **Phase 3 :** Contrôle de l'application des règles établies

4.3.2.4 Composants techniques

Ici nous allons identifier les composants essentiels de notre système. Elle se fait en fonction des besoins pour la partie conteneurisation et des besoins pour la partie automatisation pour ensuite les modéliser sur une architecture fonctionnelle pour

mieux expliquer le fonctionnement.

- Pour la partie conteneurisation nous avons besoin des éléments suivants :
  - D'une plateforme de conteneurisation pour la création d'images.
  - D'un registre d'images pour la conservation de ces dernières.
- Pour la partie automatisation les besoins :
  - D'un orchestrateur pour l'exécution des taches de déploiement et pour la gestion des gestions.
  - D'un plateforme d'automatisation pour l'envoi de fichiers descriptifs (images, déploiement et administration) à la plateforme de conteneurisation et à l'orchestrateur.

La figure suivante représente l'architecture fonctionnelle du système :

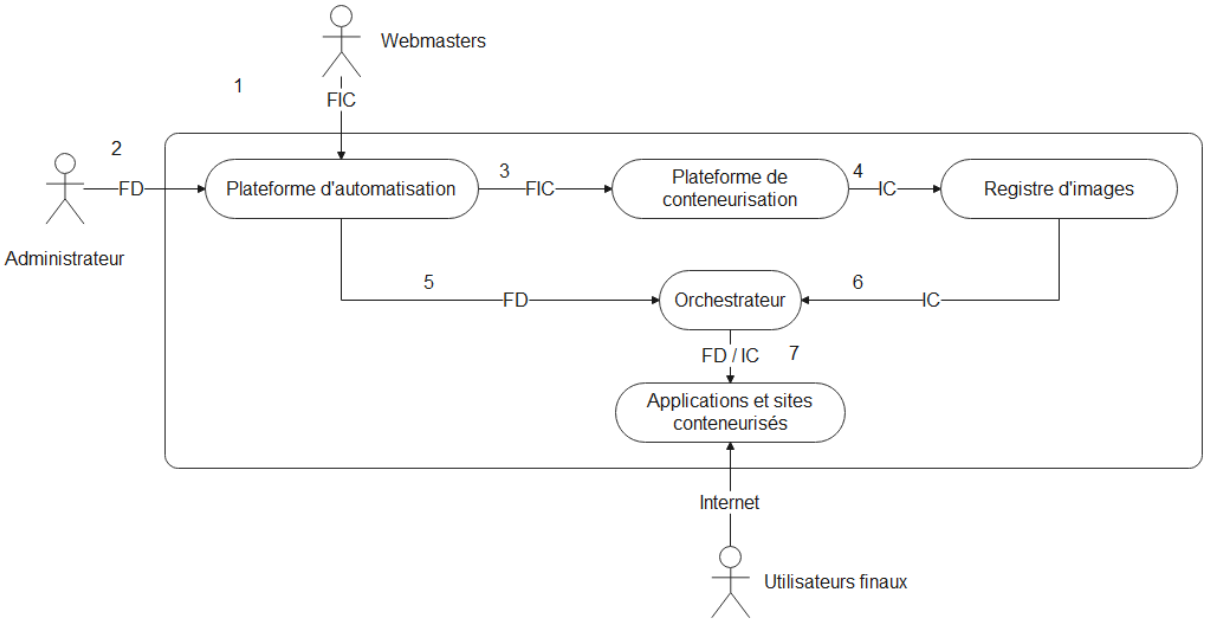


Figure 4-2 : Architecture Fonctionnelle de notre solution

- FIC : Fichiers de description d'images et composants
- FD : Fichiers de déploiement
- IC : Images d'applications ou de sites web conteneurisés
- → : Action vers la flèche

1. Le webmaster envoie les fichiers qui décrivent l'image de son application et les composants.
2. L'administrateur envoie les fichiers de déploiement.
3. La plateforme d'automatisation relaie les fichiers d'images à la plateforme de conteneurisation.
4. La plateforme de conteneurisation conteneurise l'application et créée son image puis l'envoie vers le registre d'images.
5. La plateforme d'automatisation envoie les fichiers de déploiement à l'orchestrateur.
6. L'orchestrateur se base sur les fichiers de déploiement pour identifier et récupérer l'image de l'application à déployer.
7. L'orchestrateur suit les instructions dans les fichiers de déploiement, instancie l'application à partir de l'image récupérée, l'exécute, lui allouer les ressources nécessaires, maintenir son bon fonctionnement et l'exposer sur internet.

4.3.2.5 Fonctionnalités du système

Nous allons ici présenter les fonctionnalités du système pour de clarifier leurs et déterminer les actions qui devront être réalisées. Pour leur illustration nous avons opté pour le formalisme suivant :

Numéro de fonctionnalité	Nom de la fonctionnalité
But	But de la fonctionnalité
Acteurs	Acteurs participant au cas d'utilisation
Scénario	Séquence d'actions normales associées à la fonctionnalité

- Création d'espace d'hébergement

1	Création d'espace d'hébergement
But	Mettre à disposition un espace d'hébergement
Acteur	Administrateur
Scénario	Créer un projet, lui attribuer un nom, un quota en ressource (CPU-RAM) et un nom de domaine. Puis créer un compte utilisateur avec un login et mot de passe et lui joindre au projet.

- Création de modèles de base

2	Création de modèles de base
But	Faciliter la conteneurisation
Acteur	Administrateur
Scénario	Créer un répertoire qui va recevoir les fichiers qui compose le Template prédéfini et d'y charger ce dernier pour le rendre accessible aux clients

- Déploiement d'application ou de site

3	Déploiement d'application ou de site
But	Exécuter une application ou un site web
Acteurs	Webmaster
Scénario	Le webmaster choisir le model qui l'intéresse, renseigner les informations demandées y joindre les fichiers de déploiements en cas de nécessité, importer les fichiers nécessaires et lancer le déploiement. La plateforme se charge d'envoyer les fichiers de déploiements à l'orchestrateur, qui a son tour va effectuer celui-ci

- Gestion des pics de charge

5	Gestion des pics de charge
But	Rationnaliser l'utilisation des ressources
Acteur	Orchestrateur
Scénario	Collecter les métriques de consommation et se référer aux fichiers de déploiements ou sont indiqué le taux de consommation de CPU et de RAM auxquels une entité hébergée doit être UP-Scaler ou DOWN-Scaler par rapport au nombre de réplication maximum ou minimum autorisé.

- Gestion de mises à jour

6	Gestion de mises à jour
But	Maintenance sans interruption de service
Acteur	Administrateur
Scénario	Migrer les conteneurs se logeant sur le nœud à mettre à jour vers un autre nœud, le détacher du cluster, faire les mises à jour nécessaires et le réintégré dans le cluster.

4.4 Conclusion

Dans ce chapitre nous avons fait une analyse sur le fonctionnement du système d'hébergement web actuel. Suite à cela, nous en avons ressorties ses limites qui se résident dans l'utilisation de machines virtuelles comme environnements d'exécution et dans l'adoption d'architecture 1 tiers comme architecture. Ainsi nous avons terminé ce chapitre en proposant une solution qui s'oriente sur l'utilisation de conteneurs comme environnements d'exécution et sur de l'orchestration pour faciliter et automatiser la gestion des ressources et que nous avons d'ailleurs

présentés.

5 Chapitre 5 : Etude conceptuelle

5.1 Introduction

Ce chapitre qui concerne l'étude conceptuelle, va nous permettre en s'appuyant sur les composants du système vu dans le chapitre précédant, de faire le choix des technologies à utiliser selon des critères définit. Puis proposer une architecture au niveau applicatif comme au niveau infrastructure qui vont nous servir dans la partie mise en œuvre.

5.2 Choix technologique

Pour chacune des technologies à étudier nous allons en sélectionner les trois qui répondent au mieux à nos attentes avec divers critères pour en choisir celle qui nous convient le mieux.

5.2.1 Plateforme d'automatisation

Nous avons porté notre choix sur Ansible comme outils d'automatisation en faisant une étude sur les critères suivants :

- Facilité d'utilisation qui joue un rôle déterminant dans notre processus de création de Template.
- Type d'architecture pour une communication facilite avec les entités concernées.
- Modelé de configuration pour une installation et configuration simple de l'outil.

Critères	Ansible	Puppet	Chef
Utilisation	Facile	Difficile	Difficile
Architecture	sans agent	maître-agent	maître-agent
Configuration	Pull	Pull	Pull et Push

Tableau 5-1: Comparaison entre différents outils d'automatisation

5.2.1.1 Présentation d'Ansible

Ansible est un logiciel Open Source de gestion de configuration qui permet d'automatiser des tâches avec des scripts d'automatisation. Il fonctionne sous un environnement Linux et écrit en langage Python et permet de décrire la configuration à appliquer à nos différents serveurs au travers de recettes écrites en YAML. Ces recettes contiennent une série de taches qui seront lancées les unes après les autres. Les taches utilisent des modules internes à Ansible qui permettent de décrire les opérations à effectuer et les conditions de lancement.

5.2.1.1.1 Architecture de Ansible

Dans Ansible, il existe deux classes de serveurs : le nœud maître (master) et les nœuds esclaves (slaves). Le nœud maître est une machine sur laquelle est installé l'outil Ansible et les nœuds esclaves sont les machines cibles sur lesquelles le nœud maitre ordonne l'exécution de tâches Ansible. Ansible fonctionne en se connectant à vos nœuds en SSH et en y poussant de petits programmes, appelés modules. Ces modules sont définis dans un fichier nommé le Playbook. Le nœud maître, se base sur un fichier d'inventaire qui fournit la liste des hôtes sur lesquels les modules Ansible doivent être exécutés.

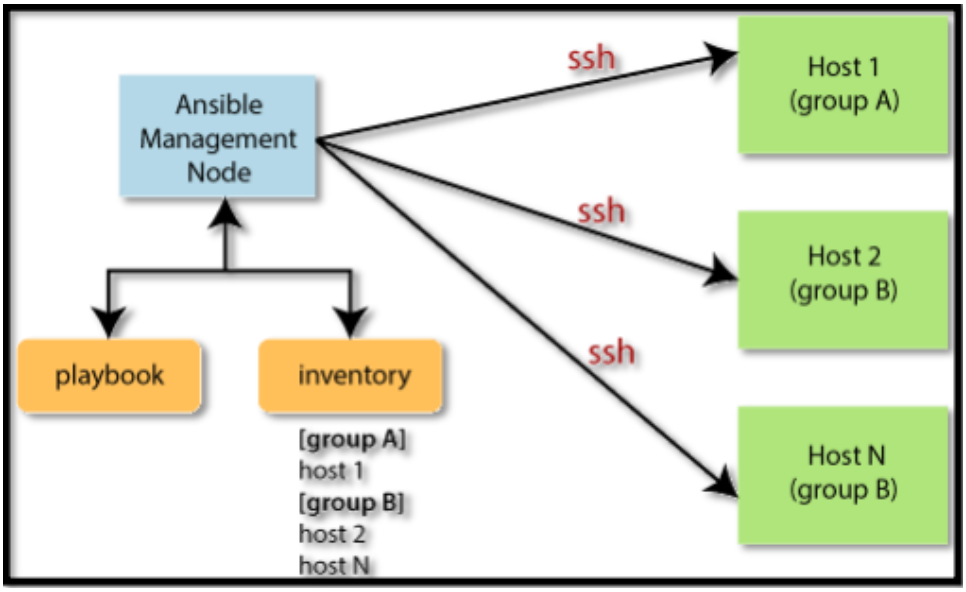


Figure 5-1: Architecture d'Ansible [14]

5.2.1.1.2 Avantages d'Ansible

Simple : l'automatisation est lisible par l'homme ne nécessitant ainsi pas de connaissance ou compétence particulier en codage. Ainsi les équipes sont beaucoup plus productive.

Installation facile : il n'y a aucun agent à exploiter et utilise OpenSSh et WinRM (Voir notre article sur l'orchestration WinRM).

Facile à prendre en main : les scénarios Ansible permet de décrire plusieurs états et tous les modules Ansible fonctionne en sens. Ansible agit seulement s'il a une action à réaliser. Un tableau de bord visuel est intégré avec des éléments de vérification et des notifications améliorée.

Pour les équipes : gain de temps, plus de productivité, automatiser les tâches redondantes et répétitives, limite les erreurs de déploiement et améliore la qualité de travail en équipe.

Pour les entreprises : meilleure utilisation des ressources en l'attribuant à l'innovation, faire des économies, faire face à la concurrence avec un déploiement rapide et sécurisé des logiciels.

5.2.2 Technologie de conteneurisation

Notre choix sur la technologie de conteneurisation s'est porté Docker en évaluant par rapport aux critères suivants :

- La facilité d'utilisation
- Le type d'architecture supporté.
- La disponibilité en terme d'images de base

Critères	Rocket	Docker	LXC
Utilisation	Moyen	Facile	Difficile
Architecture	Monolithique	Micro-service + Mono	Micro-service + Mono
Images	Moyennement	Très bien	Faiblement

Tableau 5-2: Comparaison entre différents outils de conteneurisation

5.2.2.1 Présentation de Docker

Docker est un logiciel libre qui permet d'empaqueter une application et ses dépendances et de l'exécuter dans un environnement isolé appelé conteneur. Cette isolation permet d'exécuter plusieurs conteneurs simultanément sur un hôte donné. Fondamentalement, Docker étend LXC avec le Docker daemon qui est une API (Application Programming Interface) installée sur le noyau du système d'exploitation. Depuis la version 0.9 en 2014, Docker n'utilise plus LXC comme environnement d'exécution par défaut. Il l'a remplacé par sa propre bibliothèque Libcontainer, implémentée en Go, un langage de programmation orientée service.

5.2.2.1.1 Architecture de Docker

Docker utilise une architecture client-serveur. Le client Docker communique avec le Docker daemon, qui s'occupe du travail de construction, d'exécution et de distribution de conteneurs. Le client et le Docker daemon peuvent s'exécuter sur un même système, comme un client Docker peut aussi être connecté à un Docker daemon distant. Le client et le Docker daemon communiquent à l'aide d'une API REST, via des sockets UNIX ou une interface réseau. Un autre client Docker est le Docker Compose, qui permet de travailler avec des applications constituées d'un ensemble de conteneurs.

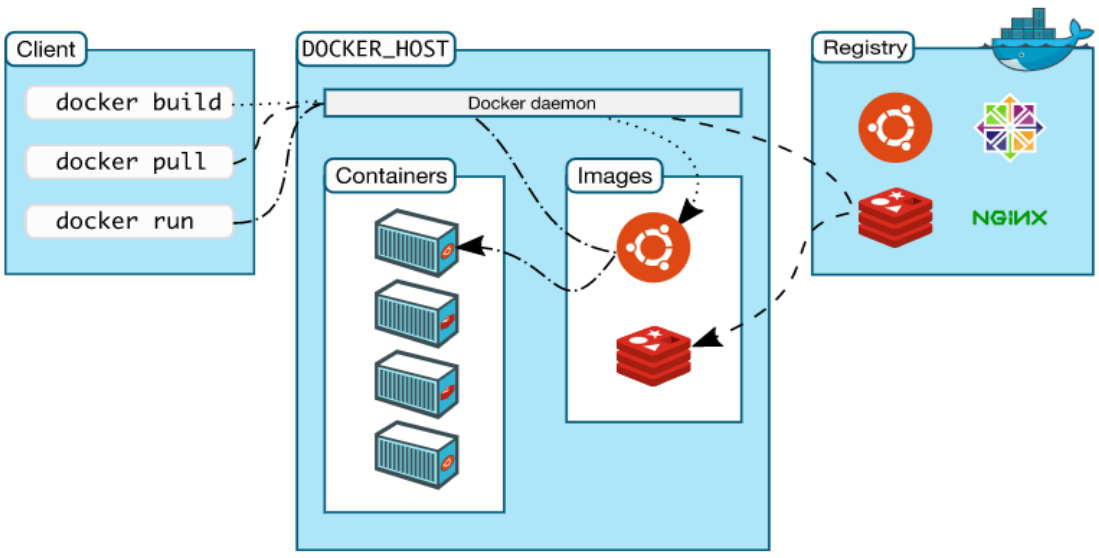


Figure 5-2: Architecture de Docker [15]

- Le Docker daemon (dockerd) écoute les requêtes de l'API Docker et gère les objets Docker tels que les images, les conteneurs, les réseaux et les volumes. Un démon peut également communiquer avec d'autres démons pour gérer les services Docker.
- Le client Docker (docker) est le principal moyen par lequel de nombreux utilisateurs de Docker interagissent avec Docker. Lorsque vous utilisez des commandes telles que docker run, le client envoie ces commandes à dockerd, qui les exécute. La commande docker utilise l'API Docker.
- Le registre Docker stocke les images Docker. Docker Hub est un registre public que tout le monde peut utiliser, et Docker est configuré pour rechercher des images sur Docker Hub par défaut. Vous pouvez même exécuter votre propre registre privé.

5.2.2.1.2 Objets Docker

Lorsqu'on utilise Docker des images, des conteneurs, des réseaux, des volumes, des plugins et d'autres objets sont créés et utilisés.

- Une image est un modèle en lecture seule avec des instructions pour créer un conteneur Docker. Souvent, une image est *basée sur* une autre image, avec quelques personnalisations supplémentaires. Un fichier Dockerfile peut être pour définir avec une syntaxe simple les étapes nécessaires pour créer l'image et l'exécuter. Chaque instruction d'un Dockerfile crée un calque dans l'image. Lorsque le Dockerfile est modifié pour reconstruire l'image, seuls les calques qui ont été modifiés sont reconstruits. Cela fait partie de ce qui rend les images docker si légères, petites et rapides par rapport aux autres technologies.
- Un conteneur est une instance exécutable d'une image. Il peut être créée, démarré, arrêté, déplacé ou supprimé à l'aide de l'API ou de la CLI Docker et être connecté à un ou plusieurs réseaux, attaché à du stockage ou même créer une nouvelle image en fonction de son état actuel.

5.2.3 Orchestrateur de conteneurs

C'est l'élément central de notre système et nous avons choisi Kubernetes comme orchestrateur de conteneurs en se basant sur les critères suivants.

- Le type d'environnement adapté
- Les technologies supportées.
- La disponibilité de supports

Critères	Kubernetes	Swarm	Mesos
Environnement	Production	Développement/test	Production
Technologie	Multiplateforme	Docker	Multiplateforme
Communauté	Forte	Moyenne	Faible

Tableau 5-3: Comparaison d'outils d'orchestration

5.2.3.1 Présentation de Kubernetes

Nous avons choisi Kubernetes comme orchestrateur de notre système. C'est une plateforme open source d'orchestration de conteneurs créé par Google. Il permet d'automatiser le déploiement et la gestion de services conteneurisés en utilisant des méthodes de prédictibilité, de scalabilité et de haute disponibilité. Principalement compatible avec Docker, Kubernetes peut fonctionner avec n'importe quel système de conteneur conforme au standard Open Container Initiative en termes de formats d'images et d'environnements d'exécution. De par son caractère open source, Kubernetes peut aussi être utilisé librement par n'importe qui, n'importe où. Kubernetes se repose sur une architecture a plusieurs concepts et abstractions, certaines existaient déjà auparavant, d'autres lui sont spécifiques. La principale abstraction sur laquelle repose Kubernetes est le cluster, à savoir un groupe de nœuds exécutant Kubernetes. Il peut s'agir de machines physiques ou virtuelles. Les nœuds quant à eux exécutent des pods. Un pod représente une instance d'une application ou d'un processus en cours d'exécution sur Kubernetes, et se constitue d'un ou plusieurs conteneurs.

5.2.3.2 Cluster Kubernetes

Le terme « cluster kubernetes » désigne un déploiement fonctionnel de Kubernetes. Un cluster Kubernetes comprend deux principaux composants : le plan de contrôle ou Master node et les machines de calcul ou workers node. Chaque nœud est son propre environnement Linux, Il peut s'agir d'une machine physique ou virtuelle. La communication entre le master et les workers (ainsi qu'à l'intérieur des nodes) est permise via un réseau privé interne au cluster, où chacune des ressources possède sa propre adresse IP afin d'être identifiée.

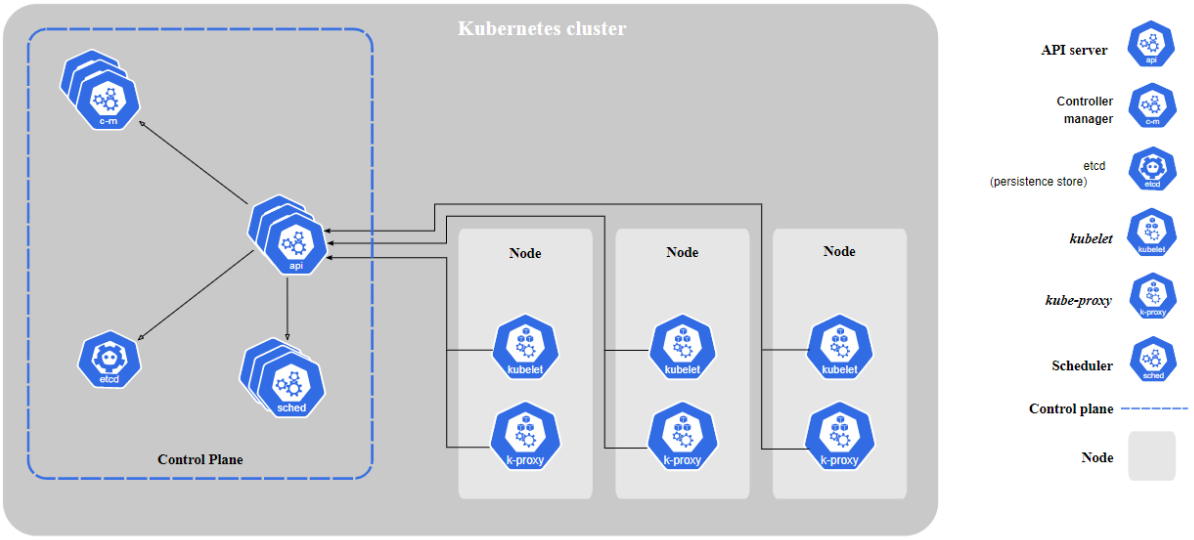


Figure 5-3: Cluster Kubernetes [16]

Le contrôle Plane est le centre névralgique d'un cluster Kubernetes. Il contient les composants Kubernetes qui contrôlent le cluster, ainsi que des données sur l'état et la configuration du cluster. Ces composants s'assurent que suffisamment de conteneurs peuvent fonctionner avec les ressources nécessaires. Le plan de contrôle est en contact permanent avec les machines de calcul. Dans le plan de contrôle on retrouve :

- **L'apiserver** permet d'interagir avec le cluster Kubernetes. C'est une API REST qui est le point d'entrée pour envoyer des commandes au cluster. Il communique avec les différents composants des nœuds du cluster pour

appliquer l'état souhaité par l'utilisateur.

- **Le scheduler** joue le rôle de planificateur, il prend en compte les besoins en ressources (processeur ou mémoire) d'une application et planifie son attribution au nœud de calcul adéquat. C'est à lui de s'assurer que la charge de travail est répartie uniformément sur l'ensemble du cluster.
- **Le controller-manager** supervise les nœuds qui quittent et rejoignent le cluster, et s'assure que l'état actuel du cluster est toujours en accord avec l'état souhaité. En cas de défaillance d'un nœud, il sa charge instancier à nouveau les pods sur les nœuds restants pour qu'ils aient toujours le nombre de répliques voulu.
- **L'etcd** est une base de données clé-valeur qui comprend les données de configuration et les informations sur l'état du cluster. C'est également là que sont stockées les informations d'identification requises pour authentifier les requêtes envoyez à l'API.

Les nœuds de calcul forment une plateforme de déploiement unique pour les ressources Kubernetes du cluster. Ils hébergent plusieurs pods systèmes qui leur permettent de communiquer avec les masters, et d'exécuter des applications utilisateur dans des pods. Les composant d'un nœud de calcul sont :

- **Le kubelet** c'est une petite application qui communique avec le plan de contrôle. Il interagit avec l'API et applique la configuration des ressources stockées dans l'ETCD sur le nœud. Il se charge de communiquer également au master l'état du nœud où il se trouve. Lorsque le plan de contrôle envoie une requête vers un nœud, le kubelet exécute l'action. Il s'assure également du bon fonctionnement des conteneurs dans un pod. Comme il est en charge de la gestion des pods, l'agent Kubelet est également présent sur les maîtres.
- **Le kube-proxy** c'est un proxy réseau qui maintient les règles réseau sur les nœuds. Ces règles permettent une communication réseau vers les Pods depuis des sessions réseau à l'intérieur ou à l'extérieur du cluster. Le service

Proxy de Kubernetes agit comme un répartiteur de charge. Il achemine le trafic réseau et permet d'exposer les services à l'extérieur du cluster.

5.2.3.3 Objets Kubernetes

Certaines abstractions représentant l'état d'un cluster et de ses composants (applications et processus conteneurisés déployés, leurs ressources réseau et disque associées, ainsi que d'autres informations sur les activités du cluster) par des objets de l'API Kubernetes ; Les objets de base de Kubernetes incluent :

- **Pod :** c'est l'unité atomique de déploiement dans Kubernetes. Il regroupe un ou plusieurs conteneurs d'applications, qui partagent une même adresse IP, le même espace partagé ainsi que les mêmes ports. Ils sont également exécutés sur le même nœud qui les héberge. Les pods sont liés au nœud qui les déploie. Ils y demeurent jusqu'à leur expiration ou leur suppression. Dans le cas d'une défaillance du nœud, de nouveaux pods possédant les mêmes propriétés que les précédents seront déployés sur d'autres nœuds disponibles. Il est possible de connecter les pods à un système de stockage persistant afin d'exécuter des applications avec état.
- **Service :** c'est une abstraction qui définit un ensemble logique de pods et une politique par laquelle y accéder. Même si les pods ont leur propre adresse IP à travers le cluster, ces IP ne sont pas exposées à l'extérieur de Kubernetes. En tenant compte du fait que les pods peuvent être détruits ou remplacés par d'autres pods, il faut pouvoir permettre à l'ensemble des pods de se découvrir automatiquement entre eux. Pour répondre à cette problématique Kubernetes regroupe les pods dans des services. Chaque service a sa propre adresse IP (unique pour le cluster) et expose un ou plusieurs ports afin de recevoir du trafic.
- **HorizontalPodAutoscaler (HPA) :** il permet de mettre à l'échelle le nombre de pods disponibles dans un cluster pour pouvoir gérer la charge de travail (ou workload) d'une application. Il détermine le nombre de pods nécessaires

en fonction des seuils de ressources définies et crée ou supprime des pods basés sur des ensembles de ces seuils. Dans la plupart des cas, ces mesures sont l'utilisation du CPU et de la RAM, mais il est également possible de spécifier vos mesures personnalisées (trafic ou autres). L'HPA vérifie en permanence les mesures de CPU et de mémoire générées par le metrics-server installé sur le cluster Kubernetes. Si l'un des seuils spécifiés est atteint, l'outil met à jour le nombre de réplicas de pod à l'intérieur du contrôleur de déploiement. Après avoir mis à jour le nombre de réplicas de pod, le contrôleur de déploiement va augmenter ou diminuer le nombre de pods jusqu'à ce que le nombre de réplicas corresponde au nombre souhaité. Si on souhaite utiliser des mesures personnalisées pour définir des règles sur la façon dont le HPA gère la mise à l'échelle de modules, le cluster doit être lié à une base de données chronologiques (time-series database) contenant les mesures souhaitées utiliser.

- **Ingress :** c'est un objet Kubernetes qui gère l'accès externe aux services dans un cluster, généralement du trafic HTTP. Il peut être configuré pour donner aux services des URLs accessibles de l'extérieur, un équilibrage du trafic de charge externe, la terminaison SSL/TLS et un hébergement virtuel basé sur le nom. Un contrôleur d'Ingress est responsable de l'exécution de l'Ingress, généralement avec un load-balancer (équilibreur de charge).

5.2.3.4 Fonctionnalités de Kubernetes

Kubernetes s'appuie sur les objets de base pour fournir des fonctionnalités supplémentaires. Parmi ses fonctionnalités on retrouve :

- **ReplicaSet :** Un ReplicaSet (ensemble de réplicas en français) a pour but de maintenir un ensemble stable de Pods à un moment donné. Cet objet est souvent utilisé pour garantir la disponibilité d'un certain nombre identique de Pods. Un ReplicaSet va atteindre son objectif en créant et supprimant des Pods pour atteindre le nombre de réplicas désirés.

Quand un ReplicaSet a besoin de créer de nouveaux Pods, il utilise alors son Pod template.

- **Requests :** C'est la quantité de ressource (cpu ou mémoire) demandée par un container. Les Requests sont évalués au moment du scheduling, par conséquent la somme de toutes les requests de mémoire ne peut pas dépasser la mémoire totale allouable (allocatable) du nœud. En ce qui concerne le cpu, la somme des requests ne peut pas dépasser la taille du cpu du nœud. Le scheduler choisi le nœud le plus adéquat pour le pod en fonction de ses requests. Ainsi la quantité de ressource demandée sera allouée au conteneur/pod, il s'agit donc d'une "soft limit".
- **Limits :** C'est le seuil maximal de ressources consommées que peut atteindre un pod. C'est donc une "hard limit" qui influence le cgroup. La valeur de limits est considérée au runtime par le Kubelet. Un conteneur ou pod qui dépasse ses limits de mémoire sera automatiquement tué, il peut être redémarré ou pas par la suite. Si on dépasse les limits de CPU alors le pod va être "throttle".
- **Volumes :** ils permettent de persister les données et fichiers des applications car dans un conteneur ils sont éphémères, ce qui présente certains problèmes. En effet, lorsqu'une défaillance du conteneur se produit, Kubelet va tenter de le redémarrer mais les données contenues dans celui-ci seront perdues. De plus, lorsque plusieurs conteneurs s'exécutent ensemble dans un pod, il est souvent nécessaire de partager les fichiers (et données) entre eux.

5.3 Architectures proposées

5.3.1 Niveau applicatif

Au niveau applicatif, l'architecture que nous proposons illustre la solution sous l'angle logiciels, ce qui va donner une meilleure vue du système et la façon dont communiquent les différents composants.

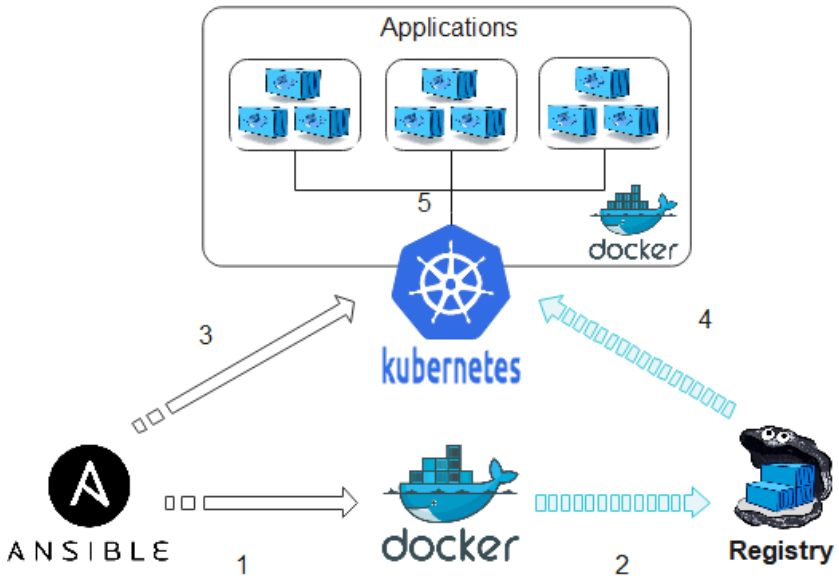


Figure 5-4: Architecture Applicative de notre solution

1. Phase d'envoi de fichiers de création d'images.
2. Phase de création et de registration d'images.
3. Phase d'envoi de fichier de déploiements.
4. Phase d'instanciation d'applications ou de sites.
5. Phase de déploiement et de gestion.

5.3.2 Niveau infrastructure

A ce niveau, l'infrastructure technique que nous proposons se base sur les critères tels que :

- La modularité pour que chaque élément du système puisse être facilement évolutif et de manière indépendante.
- La résilience pour être en capacité de conserver une application fonctionnelle dans n'importe quelle situation.
- La scalabilité pour qu'elle puisse être en capacité d'absorber de la charge croissante.

Ainsi notre système est divisé en différentes zones indépendantes. Celles nécessitant de la haute performance et/ou de la haute disponibilité regroupent un ensemble de serveurs mutualisés et dans chacune de ces zones le choix de trois

serveurs au minimum est justifié par le fait d'éviter que si un serveur tombe ou est en maintenance de reposer toute la charge sur un seul serveur. En plus, pour des raisons de sécurité une séparation des canaux de communication entre les différentes zones est optée d'où l'existence d'un réseau d'accès, d'un réseau de management et d'un réseau de stockage. La figure suivante présente l'architecture technique proposé pour notre nouveau système d'hébergement :

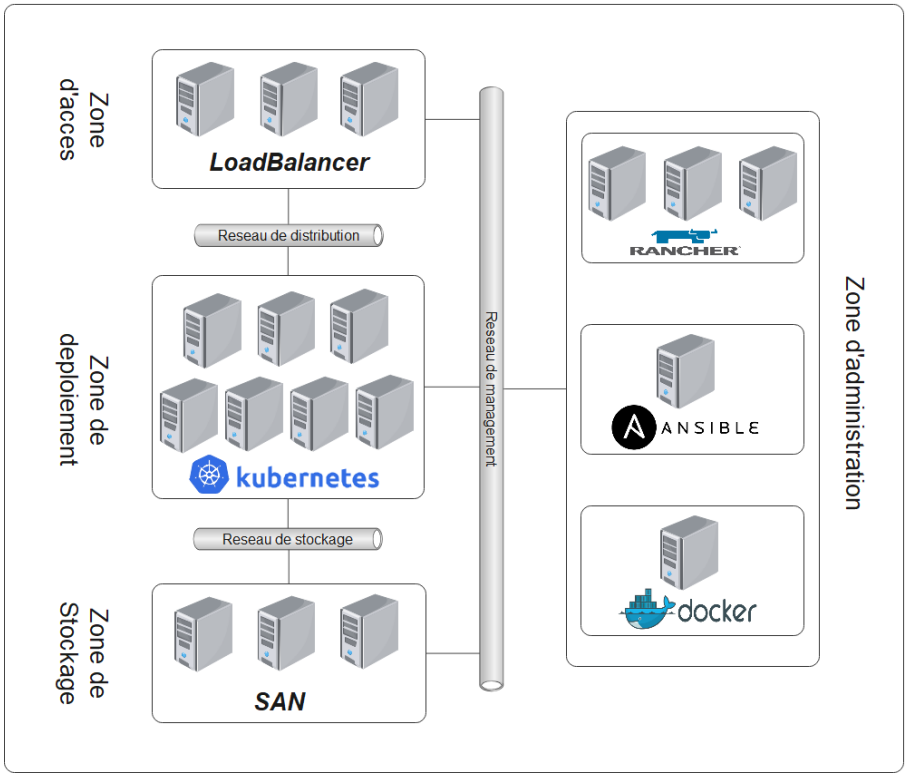


Figure 5-5: Architecture Technique de la solution

- La zone de déploiement est le cœur de notre système, elle est constituée d'un ensemble de serveurs qui forment le cluster kubernetes auquel seront logés les sites et applications conteneurisés. Ces serveurs sont organisés en deux groupes, un groupe pour les Masters Node et un autre pour les Workers Node. Cette zone est la seule qui communique avec l'ensemble des zones du système. A ce niveau, en plus de la prise en compte de la haute disponibilité et de la performance, le choix du nombre de master porté sur trois est aussi justifié du fait que pour déployer une application chaque nœud master doit donner son avis sur son emplacement ainsi avec un nombre

impair de master le risque de choisir à la fois plusieurs nœuds worker qui pousse à une réélection d'un nœud de déploiement sera supprimé.

- La zone de stockage regroupe les serveurs de stockages qui sont configuré en pool de ressources. Ces ressources seront utilisées sous forme volumes par les applications et sites web déployés dans la zone de déploiement pour persister leurs données. Elle communique avec la zone de déploiement grâce au réseau de stockage qui n'est destiné que pour cela à des fins de performance.

- La zone de d'administration est composée du serveur d'automatisation et du registre d'image qui sont utilisés dans le processus de déploiement d'applications et de création de modèles de base et de cluster RANCHER partir duquel sont administrés le cluster kubernetes, ses composants, les entités hébergées ainsi les comptes utilisateurs et les ressources du système. Cette zone communique avec l'ensemble des zones du système

- La zone d'accès permet aux internautes d'accéder aux sites et aux applications hébergés. Elle est composée d'un ensemble qui se chargent de loadbalancer les requêtes entrantes à travers une adresse IP virtuelle. Cette redondance des serveurs est faite pour absorber le trafic et pour garantir la haute disponibilité de cette fonction qui joue le rôle de porte d'entrée. La zone est connectée au réseau de management pour son administration et reliées à la zone de déploiement par le réseau de distribution pour permettre sa communication avec les Masters Node à qui sont transmet les requêtes entrantes.

5.4 Conclusion

Le but de ce chapitre était de faire le choix des principales technologies qui composent notre système. Après études, les technologies retenues sont : Ansible

comme plateforme d'automatisation, Docker comme technologie de conteneurisation et Kubernetes comme orchestrateur. Suite à cela, deux architectures sont proposées, une au niveau applicatif qui montre comment communiquent en elles ces technologies choisies et une autre au niveau infrastructure qui est divisée en cinq zones pour le rendre le système modulable, scalable et sécurisé.

6 Chapitre 6 : Tests de validation

6.1 Introduction

Ce chapitre nous permet de tester la solution proposer afin de pouvoir déterminer si elle répond aux besoins. Pour cela, nous allons en premier lieu définir un environnement de test, d'y déployer des applications puis les tests.

6.2 Environnement de test

Notre environnement de test sera composé de cinq serveurs :

- Un serveur d'administration qui va jouer le rôle de gestionnaire de cluster, de plateforme d'automatisation et de registre d'images.
- Un master node et deux workers nodes qui vont constituer notre cluster Kubernetes.
- Un serveur de stockage qui va regrouper le service de stockage et le service de gestion de fichiers.

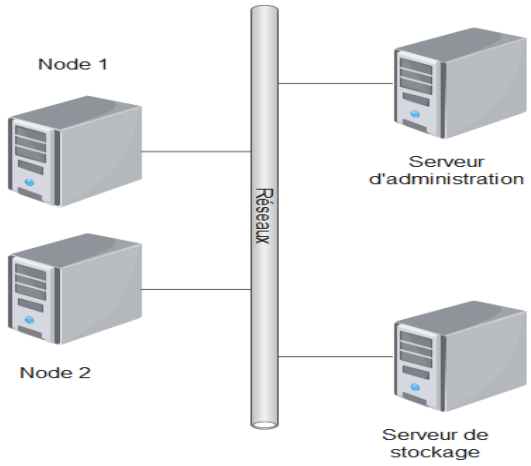


Figure 6-1: Architecture de test

6.2.1 Serveur manager

Il représente le point focal de notre infrastructure en jouant les rôles de gestionnaire de cluster, de plateforme d'automatisation, de registre d'images et de master Node. Les prérequis pour l'installation des services de base sont les suivants

6.2.1.1 Gestionnaire de cluster

Comme vu dans le chapitre précédent nous nous basons sur Rancher pour gérer notre cluster. Rancher peut être installé sur un seul nœud ou sur un cluster Kubernetes pour sa haute disponibilité. Dans notre cas nous allons l'installer sur un seul nœud pour effectuer nos tests. Les exigences pour le nœud sur lequel Rancher sera installé sont présentées suivant les exigences logicielles, matérielles et réseau.

- Exigences logicielles :
  - Distribution Linux moderne
  - Runtime Docker
  - Kubernetes
  - Package NTP (Network Time Protocol)
- Exigences matérielles
  - 2 CPU
  - 4 Go RAM
  - 20 Go Stockage
- Exigences réseau
  - IP statique
  - Ports ouverts pour le trafic entrant : 80 et 443
  - Ports ouverts pour le trafic sortant : 22, 443, 2376 et 6443

6.2.1.2 Plateforme d'automatisation

Ansible a était choix comme notre plateforme d'automatisation et pour son installation les prérequis sont les suivants :

- Exigences logicielles :
  - Distribution Linux
  - Python (version 2.7, version 3.5 et supérieures)
  - Ansible

- Exigences matérielles :
    - 1 CPU
    - 2 Go RAM
    - 20 Go Stockage
- Exigences réseau
    - IP statique
    - Ports ouverts pour le trafic : 22

6.2.1.3 Registre Docker

L'installation de docker en tant que registre d'images nécessite les prérequis suivants :

- Exigences logicielles :
    - Distribution Linux, Windows et Mac
    - Docker Engine
- Exigences matériels :
    - 1 CPU
    - 4 Go RAM
    - 60 Go Stockage
- Exigences réseau
    - IP statique
    - Ports ouverts pour le trafic entrant : 5000

6.2.1.4 Master node

- Exigences logicielles :
    - Distribution Linux
    - Container Runtime
    - Pas de swap
- Exigences matériels :
    - 2 CPU

- 4 Go RAM
- 40 Go Stockage

➢ Exigences réseau

- IP statique
- Ports ouverts pour le trafic : 6443, 2379, 2380, 10250, 10251 et 10252

6.2.2 Workers nodes

6.2.2.1 Node 1

➢ Exigences logicielles :

- Distribution Linux
- Container Runtime
- Pas de swap

➢ Exigences matériels :

- 2 CPU
- 2 Go RAM
- 20 Go Stockage

➢ Exigences réseau

- IP statique
- Ports ouverts pour le trafic : 10250 et 30000-32767

6.2.2.2 Node 2

➢ Exigences logicielles :

- Distribution Linux
- Container Runtime
- Pas de swap

➢ Exigences matériels :

- 2 CPU
- 2 Go RAM

- 20 Go Stockage

➢ Exigences réseau

- IP statique
- Ports ouverts pour le trafic : 10250 et 30000-32767

6.2.3 Serveur de stockage

6.2.3.1 Serveur de fichiers

➢ Exigences logicielles :

- Distribution Linux
- NFS

6.2.3.2 Service de transfert de fichiers

➢ Exigences logicielles :

- Distribution Linux
- FTP

6.3 Mise en œuvre et tests

Dans cette partie nous allons mettre en œuvre notre solution en déployant une application de cours en ligne avec la plateforme Moodle et un site web puis effectuer les tests sur l'adaptabilité automatique des environnements, la migration des applications et la cohabitation des services.

6.3.1 Déploiement de Moodle

6.3.1.1 Environnement de travail

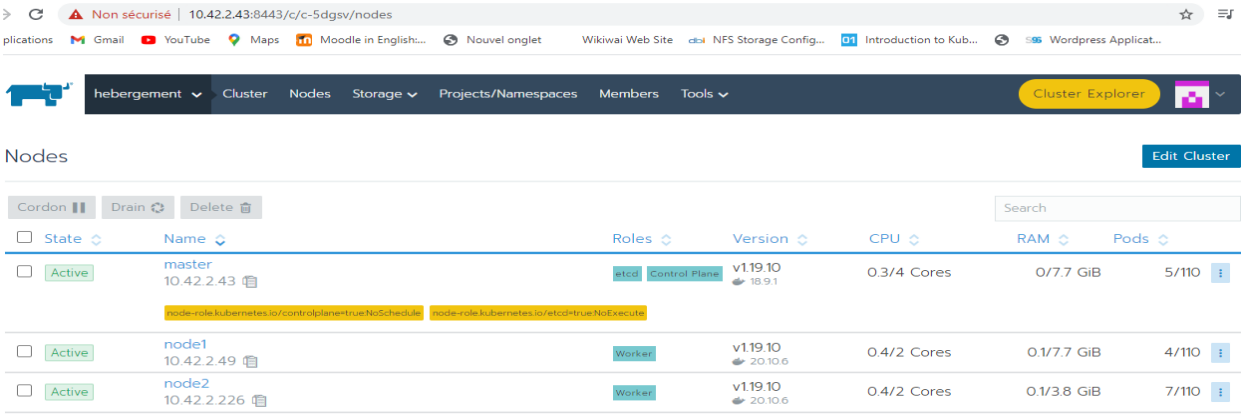


Figure 6-2: Composants du cluster

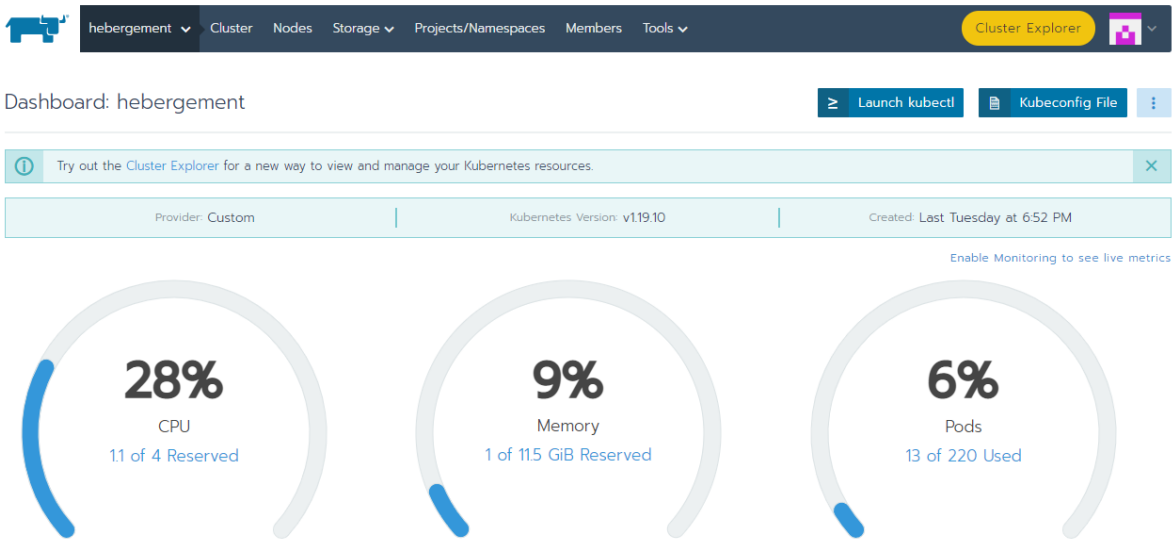


Figure 6-3: Ressources du cluster

6.3.1.2 Partie conteneurisation

Comme expliqué précédemment, pour effectuer un déploiement nous devons avant tout disposer des images conteneurisées à partir desquelles créer l'application à héberger. Pour le cas de Moodle nous aurons besoin de l'image de moodle et d'une image de base de données. Ce qui nous amené à soit créer nos propres à partir de fichiers Dockerfile ou à soit les récupérer si elles existent dans Docker Hub qui est le registre public Docker, ce qui est le cas.

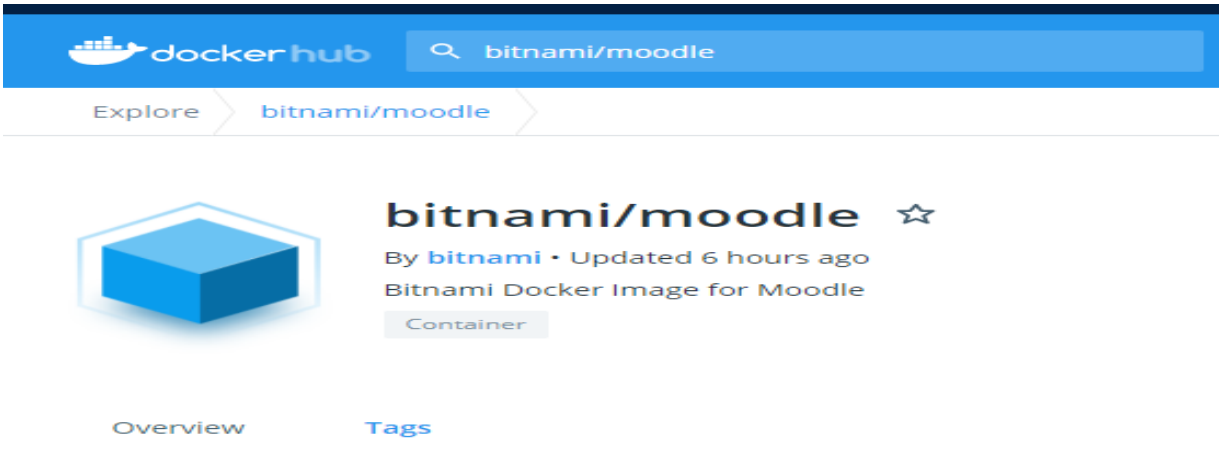


Figure 6-4: Image de conteneur Moodle

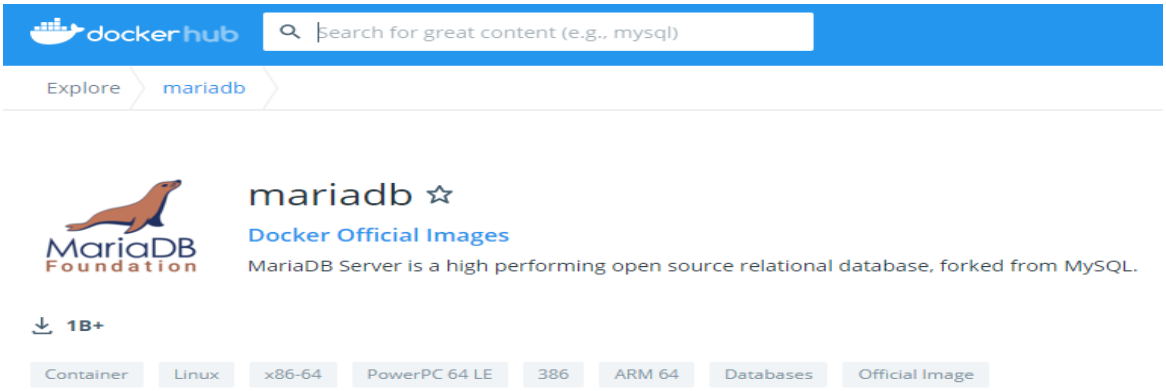


Figure 6-5: Image Conteneur Mariadb

6.3.1.3 Partie déploiement

Etape 1 :

Tout d'abord, dans notre plateforme de gestion « Rancher » nous avons créé un espace d'administration nommé « retel » pour la prise en compte de l'application à déployer qui va porter le même nom :

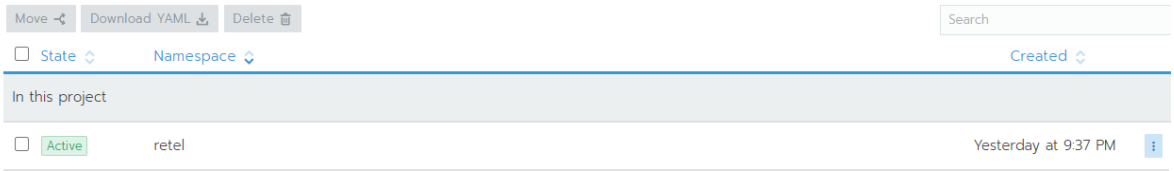


Figure 6-6: Espace d'administration

Etape 2 :

Ensuite, à partir de la plateforme d'automatisation nous avons lancé le programme de déploiement qui s'est chargé d'exécuter les taches de déploiement avec comme paramètres : le nom de l'application le nom de domaine, le nombre minimum et maximum de réplicas, les images sources, les ressources minimales à disposer, les seuils à ne pas dépasser et les périodes à autoscaler l'application.

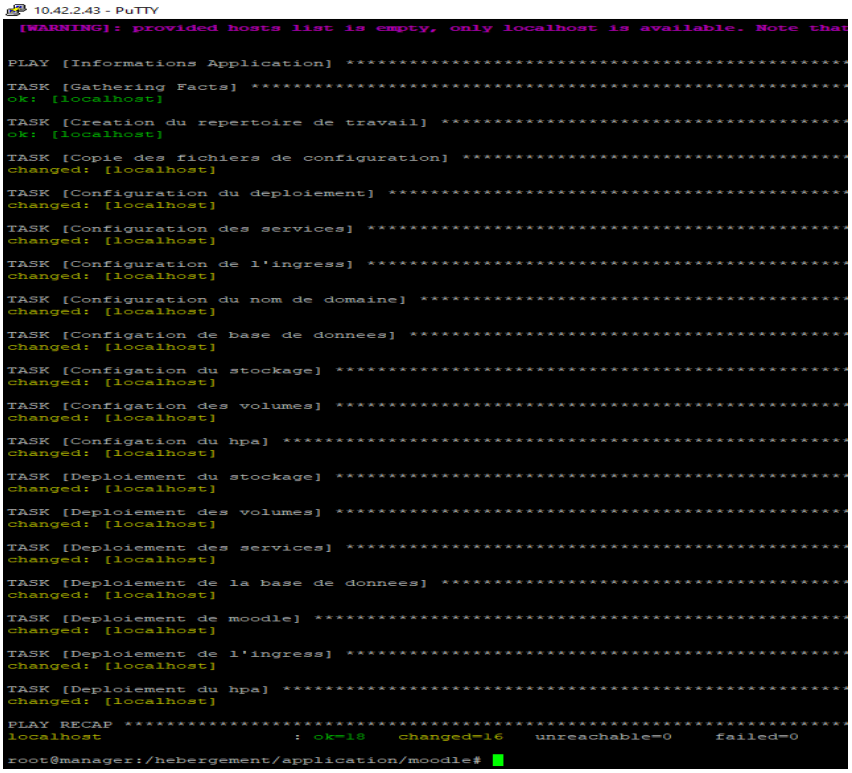


Figure 6-7: Exécution du programme de déploiement

Etape 3 :

Le déploiement devient ainsi visible à partir de la plateforme de gestion avec l'instance de Moodle et celle de la base de données :

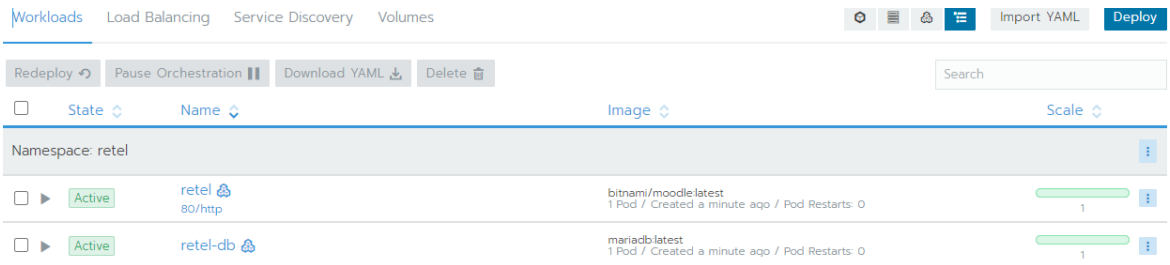


Figure 6-8: Vue du déploiement depuis Rancher

6.3.1.4 Tests pour l'adaptabilité des environnements

Pour ces tests, nous avons défini à 1 le nombre minimum et à 3 le nombre maximum de réplicas, mettre à 1 Go la quantité de mémoire requis et 2 Go la quantité à ne pas dépasser et fixer à 5 % le taux de référence pour scaler l'application.

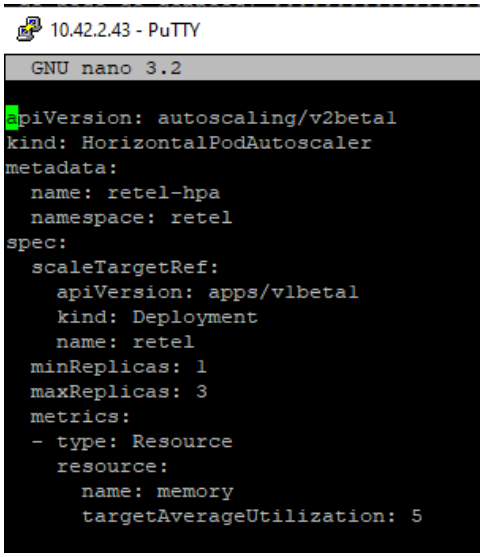


Figure 6-9: Configuration de l'autoscaling

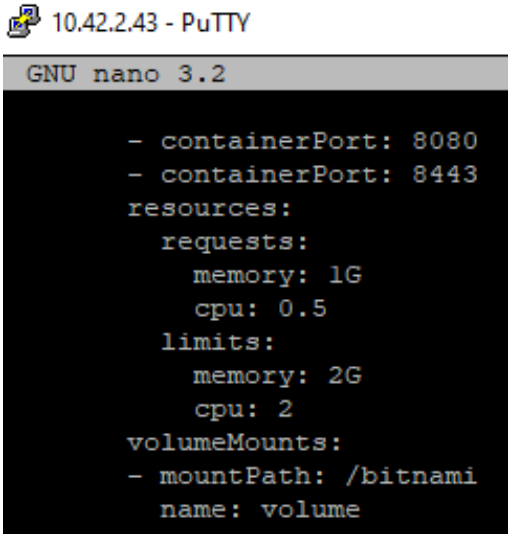


Figure 6-10: Ressources allouées

Etape 1 :

À l'état initiale, sans aucune connexion à la plateforme Moodle déployée, le taux d'utilisation de la mémoire est à 1 % et le nombre d'instance à 1 aussi :

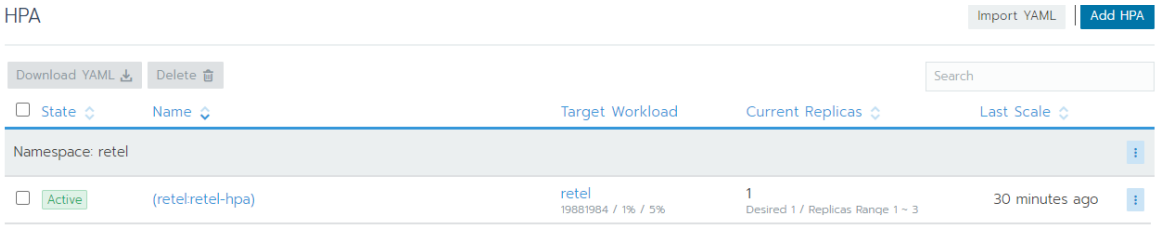


Figure 6-11: Taux de mémoire initialement utilisé

Etape 2 :

Dès que nous nous sommes connecté sur la plateforme sans se loguer, le taux d'utilisation de la mémoire est passé à 5 % et le nombre d'instance à 2 :

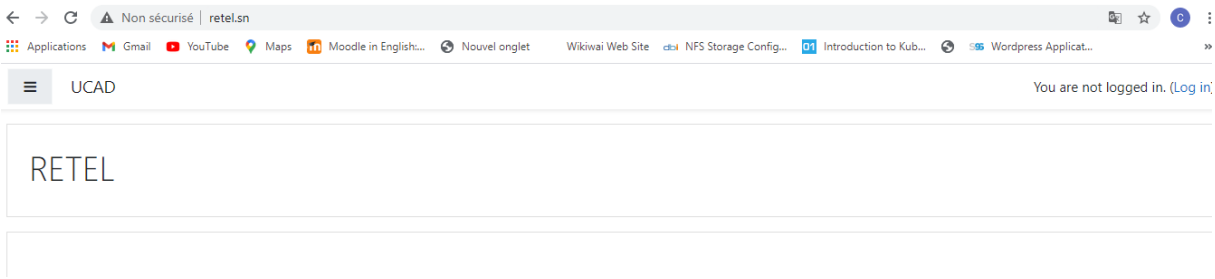


Figure 6-12: Interface d'accueil de la plateforme Moodle déployée

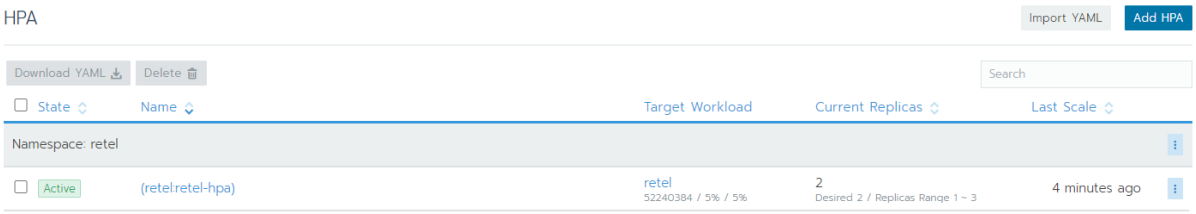


Figure 6-13: taux de mémoire utilisé lors de la connexion

Etape 3 :

Puis quand nous nous sommes logué, le taux d'utilisation de la mémoire est allé à 9 % et le nombre d'instance à 3 :

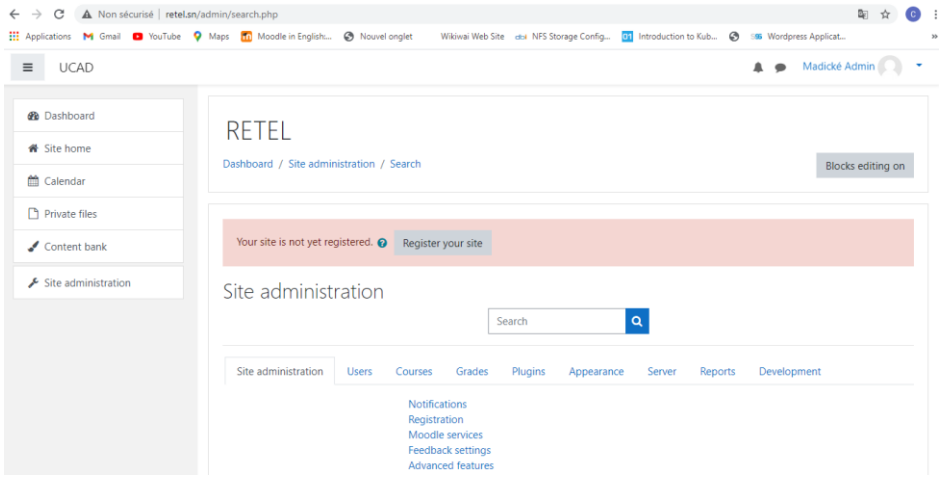


Figure 6-14: Interface d'administration de la plateforme Moodle déployée

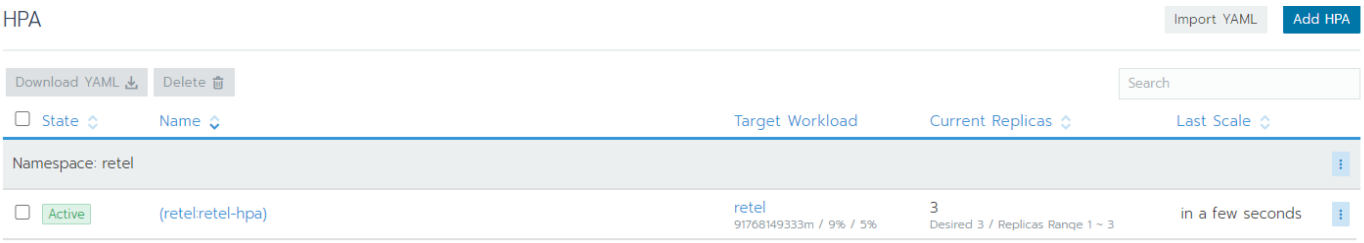


Figure 6-15: Taux de mémoire utilisé après login

6.3.1.5 Tests de migration

Pour la maintenance du système avec des mises jour fréquents des serveurs hôtes, une migration à chaud des applications d'un environnement à l'autre aide à éviter des disfonctionnements et arrêts de services. Cette fonctionnalité est testée en effectuant la migrant les instances de la plateforme Moodle se trouvant dans

node2 vers node1.

Etape 1 :

À l'état initiale 2 instances tournent dans node 2 et 1 dans node1 :

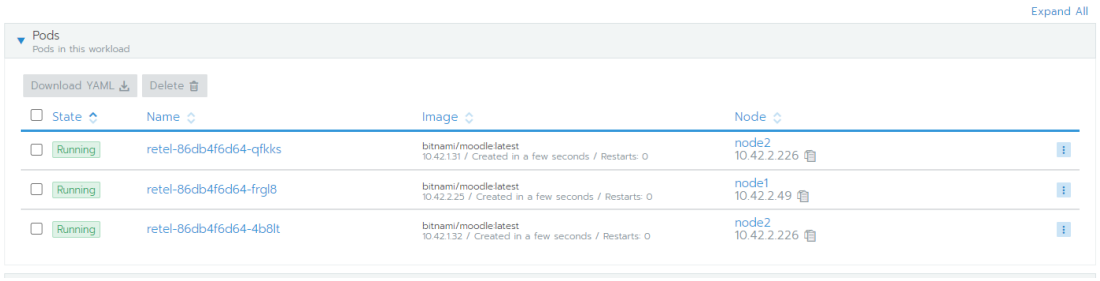


Figure 6-16: Etat avant migration

Etape 2 :

Puis après drainage, toutes les 3 instances comment à tourner dans node1 sans interruption de services :

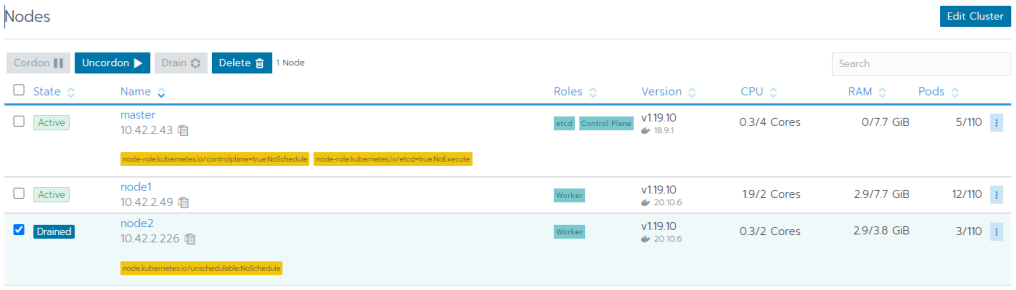


Figure 6-17: Phase de migration des instances

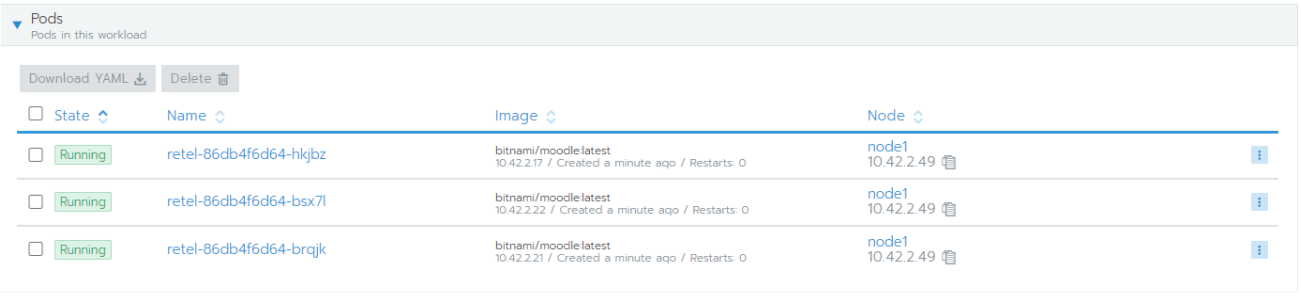


Figure 6-18: Etat après migration

6.3.1.6 Tests de compatibilité

Nous allons faire ces tests en déployant Drupal 7, 8 et 9 qui avec l'actuel système ne peuvent pas tourner ensemble sur un même. La version 7 a comme nom « site1 », la version 8 « site2 » et la version 9 « site3 » :

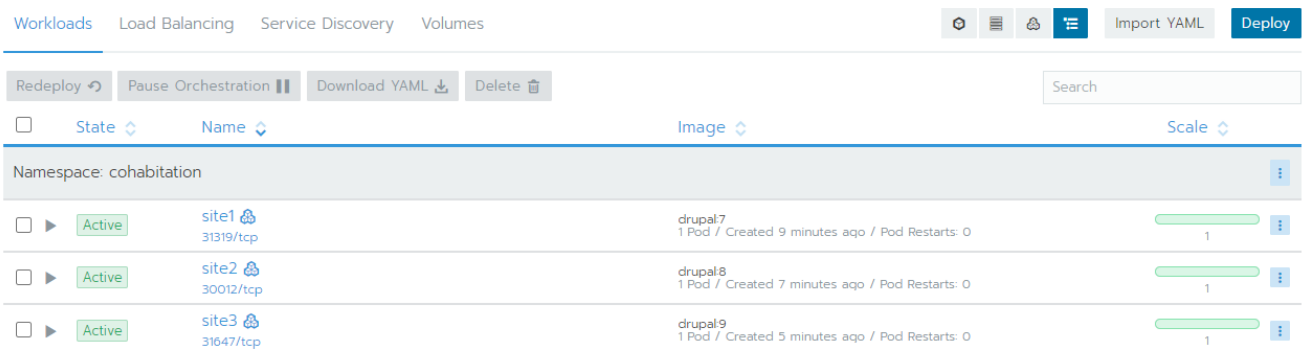


Figure 6-19: Sites Drupal déployés

6.4 Conclusion

Ce dernier chapitre nous a permis de confirmer que notre solution répond bien aux attentes en terme d'adaptabilité des environnements en fonction de la charge, de gestion facile des mises a jours et de prise en charge de toute application ou site web avec n'importes quelles dépendances. Les tests sont effectués un environnement de quatre serveurs avec une plateforme de cours en ligne Moodle et des site web Drupal.

Conclusion Générale

Le but de ce mémoire était de proposer une solution qui va permettre de rationaliser l'utilisation des ressources du système d'hébergement web de l'Agence de l'Informatique de l'Etat. Cependant, après avoir présenté l'agence et fait un état de l'art sur l'hébergement web où nous avons donné les définitions des terminologies du domaine et présenter les types d'hébergements existants. Ensuite nous avons fait un revu sur les techniques actuelles de gestion de ressources dans l'hébergement web qui s'est articulée autour des environnements d'exécution, des architectures de service et des modèles gestion qui par la suite nous ont servis de références pour mener notre étude sur le système existant et de ressortir les limites.

Ces dernières se résidants dans l'utilisation de machines virtuelles comme environnements d'exécution et du le fait de se baser sur une architecture de service à 1-tiers nous a amené à proposer comme solution de mettre en place un nouveau **système d'hébergement** qui est **automatisé** et qui utilise des **conteneurs** comme environnements d'exécution. Cette solution avait pour objectifs d'adapter automatiquement les environnements en fonction de la charge, de permettre la cohabitation des entités hébergés et permettre une migration à chaud des environnements pour faciliter la gestion des mises à jour. Après une mise en place d'une infrastructure de tests basée sur quatre nœuds accompagnée d'une série de tests, les résultats obtenus ont répondus à nos attentes et nous ont permis de valider la solution proposée qui en offre la possibilité d'optimiser le système d'hébergement web de l'ADIE par une rationalisation de l'utilisation des ressources.

Bibliographie

Kubernetes : Gérez la plateforme de déploiement de vos applications conteneurisées, Edition ENI, Octobre 2019

Docker en production : Mise en œuvre en cluster sur des microservices, Edition ENI, Décembre 2020

Mémoire : Partage Efficace Des Ressources De Calcul Dans Le Nuage Informatique, Par Achraf Labidi, Montréal Le 11 Juillet 2017

Thèse : Performance et gestion de ressources dans un cloud multi-virtualisé Mathieu Bacou 12 mai 2020

Livre Blanc : Orchestration Définitions, Cas d'usage, Choix et déploiement, Sécurité, Retours d'Expérience Réalisé par le Groupe de Travail Orchestration-Automatisation du CRiP Pilote : Hugues Fondeux Assistance éditoriale : Renaud Bonnet, Juin 2013

Article : Hébergement d'applications Web dans le cloud AWS, Septembre 2017

Mémoire : Une Plateforme D'orchestration De Conteneurs Docker Tolérante Aux Fautes Byzantines, Par Gor Mack Diouf Avril 2019

Thèse : Architecture pour les systèmes de déploiement logiciel à grande échelle : prise en compte des concepts d'entreprise et de stratégie Noelle Merle, Jan 2006

Webographie

https://www.hebergement-sites.fr/hebergement-applications-sur-le-web/
https://waytolearnx.com/2018/09/difference-entre-serveur-dapplication-et-serveur-web.html
https://www.twaino.com/blog/creation-site-web/hebergement-web
https://fr.wikipedia.org/wiki/Virtualisation#Diff%C3%A9rentes_techniques
https://www.ionos.fr/digitalguide/serveur/configuration/la-virtualisation/
https://www.ionos.fr/digitalguide/serveur/know-how/conteneurs-informatiques-virtualisation-sans-emulation/
https://www.esds.co.in/blog/cluster-computing-definition-architecture-and-algorithms/
https://www.lebigdata.fr/cluster-definition#:~:text=Enfin%2C%20nous%20%C3%A9voquerons%20la%20notion,fonctionnant%20comme%20un%20seul%20syst%C3%A8me
https://www.data-transitionnumerique.com/cluster-informatique-grappe-big-data/#2-2_-_Caracteristiques_du_cluster_computing
https://blog.wescale.fr/2018/09/14/k8s-gestion-de-ressources/
https://www.redhat.com/fr/topics/containers/what-is-kubernetes
https://platform9.com/blog/kubernetes-monitoring-webinar-takeaways-from-platform9-product-manager-chris-jones/
https://www.zdnet.com/article/what-kubernetes-really-is-and-how-orchestration-redefines-the-data-center/
https://docs.docker.com/
https://docs.ansible.com/
https://kubernetes.io/docs/
https://rancher.com/docs/
https://www.youtube.com/c/xavki-linux
https://www.youtube.com/c/Cookieconnect%C3%A9/playlists

[1] https://blog.simplebo.fr/hs-fs/hubfs/Editorial/Blog/Article/49.%20H%C3%A9bergement%20site%20web/Schema-explication-serveur-hebergement-site-web.jpg?width=800&name=Schema-explication-serveur-hebergement-site-web.jpg
[2] https://www.lafabriquedunet.fr/wp-content/uploads/2019/07/vps-vs-dedicated-server.png
[3] https://www.twaino.com/wp-content/uploads/2020/09/36-Hebergement-web-dedie-1.jpg
[4] https://www.twaino.com/wp-content/uploads/2020/09/34-L-hebergement-Web-VPS.jpg
[5] https://www.twaino.com/wp-content/uploads/2020/09/38-un-conteneur.jpg
[6] https://d33wubrfki0l68.cloudfront.net/26a177ede4d7b032362289c6fccd448fc4a91174/eb693/images/docs/container_evolution.svg
[7] https://www.it-connect.fr/wp-content-itc/uploads/2013/06/hyperviseur-type1.png
[8] https://imgsite.compufirst.com/images/https:/www.it-connect.fr/wp-content-itc/uploads/2013/06/hyperviseur-type2.pngwww/images/main/pages_magasins/le-lab/hyperviseur/hyperviseur-t2.png
[9] https://blog.ovh.com/fr/blog/wp-content/uploads/2019/06/9EC316F8-09D0-4F78-8DF1-E7FDBED8E93C-367x510.png
[10] https://www.esds.co.in/blog/wp-content/uploads/2020/12/HA-Cluster.png
[11] https://www.esds.co.in/blog/wp-content/uploads/2020/12/Load-Balancing.png
[12] https://www.esds.co.in/blog/wp-content/uploads/2020/12/5-660x467.png
[13] https://www.esds.co.in/blog/wp-content/uploads/2020/12/6-660x467.png
[14] https://devopssec.fr/documents/architecture-devsecops-automatisator.pdf
[15] https://docs.docker.com/engine/images/architecture.svg
[16] https://d33wubrfki0l68.cloudfront.net/2475489eaf20163ec0f54ddc1d92aa8d4c87c96b/e7c81/images/docs/components-of-kubernetes.svg