

UNIVERSITÉ DE SOUSSE

ECOLE NATIONALE D'INGÉNIEURS DE SOUSSE



المدرسة الوطنية للمهندسين بسوسة
École Nationale d'Ingénieurs de Sousse

Département d'Informatique Industrielle

Mémoire de Projet de Fin d'Études

Présenté en vue de l'obtention du diplôme d'

Ingénieur en Génie Télécommunications
Embarquées

Option : Réseaux de Télécommunications

Plateforme d'automatisation de la configuration
des environnements logiciels "ConfigAutomation"

Réalisé par : Achraf YAAKOUB

Soutenu le **/**/2023 devant le jury :

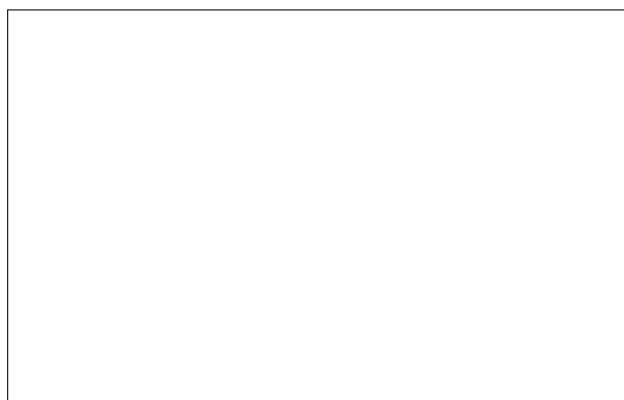
Président : M. ..., ENISo

Rapporteur : M^{me} ..., ENISo

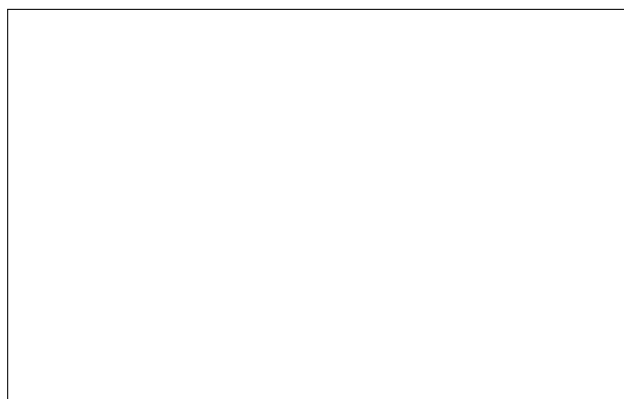
Encadrant Académique : M^{me} Imen KHADHRAOUI, ENISo

Encadrant Professionnel : M. Mohamed Zied TLILI, MEDIANET

SIGNATURES DES ENCADRANTS



Mme. Imen KHADHRAOUI



Mr. Mohamed Zied TLIL

DÉDICACES

Je dédie le résultat de mes 18 ans d'études

À mon père **Abdellaziz**

Pour les conseils inestimables qui m'ont permis de rester sur la bonne voie

À ma mère **Mabrouka**

Pour m'avoir encouragé et stimulé quand j'en avais besoin

À mon frère **Ayoub**

Vous avez toujours été présents pour me motiver et me pousser à devenir la
personne que je suis aujourd'hui

À mes sœurs **Safa et Rihab**

Qui n'ont ménagé aucun effort pour me soutenir tout au long de mes études.
Leur dévouement sans faille a été pour moi une source constante de motivation et
de soutien

À toute ma **Famille**

Il est impossible de trouver les mots pour exprimer à quel point je vous suis
reconnaisant pour le soutien et les encouragements que vous m'avez prodigués.

Je vous dédie ce travail en signe de reconnaissance pour l'amour et la bonté
exceptionnels que vous m'offrez chaque jour.

À mes **Ami(e)s**

Qui mon soutenu durant les heures les plus pénibles

À TOUS QUI M'AIMENT

Que Dieu le tout puissant vous maintienne en bonne santé.

REMERCIEMENTS

Le mérite de cette réalisation incombe à plusieurs personnes que je tiens à remercier sincèrement.

Tout d'abord, je suis reconnaissant envers mon encadrante académique, madame **Imen KHADHRAOUI**, pour son désir de m'inciter à aller plus loin dans ma réflexion et pour la confiance qu'il a placée en moi tout au long du projet.

De même, mon encadrant professionnel, monsieur **Mohamed Zied TLIL**, a été une figure essentielle en m'accueillant dans son milieu de travail chez MEDIANET et en me permettant de collaborer avec lui pour acquérir de nouvelles compétences et connaissances.

Tous les enseignants qui ont participé à mon évolution scientifique durant les années écoulées de formation à l'Ecole Nationale d'Ingénieurs de Sousse.

En fin, Les membres du jury qui ont accepté de participer à l'évaluation de mon travail.

RÉSUMÉ

Dans le cadre de notre projet, nous sommes concentrés sur l'automatisation des processus liés à la gestion des configurations logicielles. Notre objectif principal est d'optimiser les étapes de configuration des logiciels, afin de garantir des déploiements rapides, cohérents et sans erreur.

Grâce à notre plateforme "ConfigAutomation" , MEDIANET peut bénéficier d'une productivité accrue, d'une réduction des erreurs humaines et d'une amélioration de la qualité des déploiements logiciels. Notre approche axée sur l'automatisation des configurations permet d'optimiser les processus de déploiement, d'améliorer la fiabilité des systèmes et de réduire les délais de mise sur le marché.

Mots clés : Configurations logicielles, Ansible, Automatisation, DevOps.

ABSTRACT

In our project, we focused on automating the processes related to software configuration management. Our main goal was to simplify and optimize the software configuration steps to ensure fast, consistent and error-free deployments.

With our "ConfigAutomation" platform, organizations can benefit from increased productivity, reduced human error, and improved software deployment quality. Our configuration automation approach optimizes deployment processes, improves system reliability, and reduces time to market.

Key words : Software configurations, Ansible, Automation, DevOps.

ABRÉVIATIONS

DevOps : Développement et d'Opérations

HTTP : Hypertext Transfer Protocol

HTTPS : Hypertext Transfer Protocol Secure

IaaS : Infrastructure as a Service

IaC : Infrastructure as code

IP : Internet Protocol

JDBC : Java Database Connectivity

JSON : JavaScript Object Notation

JWT : JSON Web Token

PHP : Hypertext Preprocessor

SSH : Secure Shell

UML : Unified Modeling Language

YAML : YAML Ain't Markup Language

TABLE DES MATIÈRES

Introduction Générale	1
1 Contexte général du projet	3
1.1 Organisme d'accueil	3
1.1.1 Présentation de Medianet	3
1.1.2 Services de l'entreprise	4
1.1.3 Principaux clients de MEDIANET	4
1.2 Présentation du projet	6
1.2.1 Cadre général du projet	6
1.2.2 Problématique	6
1.2.3 Objectifs	7
1.2.4 Solutions existantes	8
1.2.5 Critiques des solutions existantes	10
1.2.6 Solution proposée	11
1.3 Méthode de gestion de projet	12
1.3.1 Méthode agile	12
1.4 Suivre de la progression du projet	14
1.5 Outils adoptés pour la gestion de projet	14
2 Etude préalable	16
2.1 Concepts Généraux	16
2.1.1 DevOps	16
2.1.2 Approvisionnement en logiciels	17
2.1.3 Automatisation	17
2.1.4 L'infrastructure en tant que service	18
2.1.5 L'infrastructure en tant que code	18
2.1.6 Comparaison entre IaaS et IaC	19
2.2 L'infrastructure de l'entreprise	19
2.2.1 Services web	19

2.2.2	Services de base de données	20
2.2.3	Services pare-feux	21
2.2.4	Services de monitoring	22
2.2.5	Cloud computing	22
2.3	L'outil de gestion et d'automatisation de la configuration : Ansible .	24
2.3.1	Architecture Ansible	24
2.3.2	Modules	24
2.3.3	Inventaires	25
2.3.4	Playbooks	25
2.3.5	Rôles	26
3	Analyse et spécification des besoins	28
3.1	Les acteurs	28
3.2	Identification des besoins	29
3.2.1	Spécification des besoins fonctionnels	29
3.2.2	Fonctionnalités du système	29
3.2.3	Scénarios des fonctionnalités du système	33
3.2.4	Spécifications des besoins non fonctionnels	39
4	Conception	40
4.1	Architecture physique	40
4.2	Architecture logique	42
4.2.1	Architecture MVC	44
4.3	Modèle de données	45
4.4	Backlog du projet	47
4.5	Planification des sprints	47
5	Réalisation	49
5.1	Choix technique	49
5.1.1	Outil de gestion de configuration	49
5.1.2	Outils de gestion de code source	51
5.1.3	Plateformes, langages et environnements de développement	53
5.2	Démonstration	56
5.2.1	Connexion aux serveurs	56
5.2.2	Développement des rôles Ansible	57
5.2.3	Interfaces réalisées	61
	Conclusion générale et perspectives	67
	Bibliographie	69

LISTE DES FIGURES

1.1	Le logo de Medianet [1]	4
1.2	Foreman Hosts	8
1.3	Foreman configure	8
1.4	Cobbler interface	9
1.5	Cycle de vie de SCRUM [4]	13
1.6	Diagramme de Gantt du projet	14
1.7	Tâches sous Trello	15
1.8	Documents partagés sous Drive	15
2.1	Cycle DevOps [6]	16
2.2	Architecture Ansible. [2]	24
2.3	Exemple fichier Ansible Inventory.	25
2.4	Exemple Playbook pour installation Apache.	25
2.5	Structure du rôle.	26
3.1	Diagramme de cas d'utilisation des principales fonctionnalités.	30
3.2	Diagramme de cas d'utilisation de gestion des configurations.	32
3.3	Diagramme de séquence "Authentification".	34
3.4	Diagramme de séquence "Configuration automatisée des serveurs".	35
3.5	Diagramme de séquence "Consultation des rôles ansible".	37
3.6	Diagramme de séquence "Gestion des Clients".	38
4.1	Diagramme de déploiement.	41
4.2	Architecture logique.	43
4.3	Architecture MVC. [10]	44
4.4	Diagramme de classe.	46
5.1	Génération d'une paire de clés SSH.	56
5.2	Exemple : Copie de la clé publique sur Debian.	56
5.3	Exemple : la connexion SSH avec Debian.	57

5.4	Structure du rôle Apache.	58
5.5	Playbook de configuration d'Apache pour Debian.	59
5.6	Playbook de configuration d'Apache pour RedHat.	59
5.7	Playbook d'installation Apache.	60
5.8	Exécution d'installation Apache.	60
5.9	Interface d'authentification.	61
5.10	Interface d'authentification.	61
5.11	Exemple de configurations.	62
5.12	Exemple de résultats détaillés pour le cas de UFW.	63
5.13	Interface de gestion des clients.	63
5.14	Interface d'ajout d'un client.	64
5.15	Interface de gestion des vhosts.	64
5.16	Interface de consultation des rôles.	65
5.17	Interface de Consultation des services.	65
5.18	Interface de Consultation des logs.	66
5.19	Exemple d'un fichier log.	66

LISTE DES TABLEAUX

1.1	Liste des clients de MEDIANET par secteur d'activité	5
1.2	Comparaison entre Foreman et Cobbler	10
1.3	L'équipe Scrum	13
2.1	Comparaison IaaS vs IaC	19
3.1	Description textuelle du cas d'utilisation des principales fonction- nalités	31
3.2	Description textuelle du cas d'utilisation de "Gestion de la configu- ration des serveurs"	32
4.1	Backlog du projet	47
5.1	Récapitulatif de comparaison des outils de gestion et automatisation de configuration	51
5.2	Récapitulatif de comparaison des outils de gestion de code source .	53

INTRODUCTION GÉNÉRALE

Dans le monde du développement logiciel, la configuration des environnements de développement peut être une tâche fastidieuse et sujette à des erreurs humaines.

Pour automatiser ce processus, de nombreuses entreprises ont recouru à des outils d'infrastructure en tant que code (IaC) pour décrire et déployer l'ensemble de leur infrastructure, y compris leurs environnements de développement. Cependant, la configuration et l'utilisation de ces outils peuvent être complexes et nécessitent souvent une expertise technique approfondie. C'est dans ce contexte que s'inscrit notre projet de fin d'étude intitulé "ConfigAutomation".

L'objectif principal de ce projet est de concevoir et développer une plateforme d'automatisation de la configuration des environnements logiciels. Cette plateforme a pour ambition de simplifier et d'automatiser le processus de création et de gestion des environnements de développement. Elle est conçue pour offrir aux utilisateurs une interface utilisateur intuitive et conviviale. La mise en place de cette plateforme répond aux besoins spécifiques de l'entreprise en matière d'amélioration de son efficacité opérationnelle. En automatisant les processus de configuration, l'entreprise pourra réduire de manière significative le temps et les coûts associés à la gestion des environnements logiciels. De plus, en centralisant et en simplifiant les opérations de configuration, la plateforme permettra aux équipes de développement de se concentrer davantage sur la création de valeur et le développement de nouvelles fonctionnalités. La plateforme d'automatisation de la configuration offrira un large éventail de fonctionnalités, telles que la création rapide d'environnements de développement, la configuration automatique des composants logiciels, la gestion des dépendances, la sauvegarde et la restauration des configurations, et bien d'autres encore. Elle sera conçue de manière modulaire et extensible, afin de pouvoir s'adapter aux besoins spécifiques de l'entreprise et évoluer en fonction des nouvelles exigences.

Le présent rapport est divisé en cinq chapitres. Le premier chapitre introduit le projet en présentant l'organisation d'accueil, les objectifs poursuivis, une analyse des solutions existantes, solution proposée et la méthodologie choisie. Le deuxième chapitre se concentre sur les concepts de base de notre projet. Le troisième chapitre constitue une étude des spécifications des besoins fonctionnels et non fonctionnels. Le quatrième chapitre est consacré à une étude approfondie de l'architecture de la solution que nous proposons. Enfin, Le cinquième et dernier chapitre détaille la réalisation du projet en expliquant les différentes étapes qui ont été nécessaires pour atteindre les objectifs fixés.

CHAPITRE 1

CONTEXTE GÉNÉRAL DU PROJET

Introduction

Ce chapitre est consacré à présenter l'entreprise dans laquelle nous avons mené notre projet, à décrire le contexte général du projet et la problématique qu'il vise à résoudre et à présenter la méthode de travail que nous avons adoptée pour mener bien à ce projet.

1.1 Organisme d'accueil

1.1.1 Présentation de Medianet

Medianet [1] est une entreprise de services en ingénierie informatique, une référence dans le domaine des TIC en Tunisie. Elle exerce actuellement sur le marché local ainsi qu'à l'international. Elle a été créée en 1998 par M. Iheb BEJI sous la forme juridique SARL. Sa mission principale est de créer et développer des portails Web, des sites et des applications mobiles pour ses différents clients, en leur fournissant une variété de solutions adaptées pour réussir leurs stratégies interactives de marketing digital, mettre en œuvre leur e-business et participer à la transformation numérique de leur entreprise. Pour mener bien à ses projets, Medianet se concentre sur quatre étapes essentielles :

- Conception
- Suivi et contrôle
- Marketing digital/Web marketing
- Publicité

Medianet possède une grande expérience dans la conception de solutions numériques pour des clients provenant de divers secteurs du marché :

Banque et finance, assurance, tourisme, e-commerce, santé et éducation.



FIGURE 1.1 – Le logo de Medianet [1]

1.1.2 Services de l'entreprise

Medianet offre une expertise complète dans le domaine digital, qui dépasse largement la simple conception de sites web et d'applications mobiles. Elle met à la disposition de ses clients des experts techniques de marketing pour les accompagner dans la réussite de leurs projets. L'entreprise est responsable de la création d'applications informatiques pour les environnements web et mobile, qu'il s'agisse de sites web spécifiques, de services en ligne ou d'applications multimédias. Dans ce contexte, Medianet s'engage à garantir l'interactivité des produits et des services qu'elle développe.

1.1.3 Principaux clients de MEDIANET

Exceller dans un domaine en pleine expansion est le principal défi que l'entreprise Medianet s'efforce de relever. Depuis ses débuts, Medianet a constamment recherché l'excellence en se positionnant comme un acteur majeur dans un marché en pleine croissance. L'entreprise s'est engagée à cibler spécifiquement une catégorie de clients exigeants, en leur offrant un service de haute qualité.

Pour atteindre cet objectif, Medianet a mis en place une approche axée sur l'innovation et la technologie de pointe. L'entreprise investit continuellement dans la recherche et le développement pour rester à la pointe des dernières avancées de l'industrie. Elle s'appuie sur une équipe hautement qualifiée et passionnée, qui met en œuvre des pratiques de travail rigoureuses et efficaces.

La quête de l'excellence de Medianet se reflète également dans sa culture d'entreprise axée sur la satisfaction du client. L'entreprise accorde une attention particulière aux besoins et aux attentes de ses clients, en veillant à offrir des solutions personnalisées et adaptées à leurs besoins spécifiques. Cette approche orientée client a permis à Medianet de développer des relations solides et durables avec sa clientèle, tout en renforçant sa réputation d'excellence dans le secteur.

En résumé, le principal défi de Medianet est de se démarquer dans un secteur en pleine croissance en offrant un service de haute qualité et en répondant aux besoins spécifiques de sa clientèle. L'entreprise met en œuvre une approche innovante, s'appuie sur des technologies de pointe et cultive une culture d'entreprise

axée sur la satisfaction du client. C'est ainsi qu'elle s'affirme comme un acteur de référence dans son domaine d'activité.

Le tableau 1.1 détaille les différents clients de Medianet, témoignant de notre engagement à servir un large éventail d'entreprises et d'organisations dans divers secteurs d'activité.

TABLEAU 1.1 – Liste des clients de MEDIANET par secteur d'activité

Secteur	Nom du client
Tourisme	Magic Life Hotels and Resorts
Banque et assurance	We Bank – Attijari Bank Banque Zitouna Maghreb Assurance GAT Assurance LOYD Assurance
Nourriture	Président KFC Randa Warda Bidha
Consommateur	Géant Magasin Générale Fatales
Télécommunications et technologies	Ooredoo Tunisie Télécom Simens Tunisie SONY Scoop Tunisie Hannibal TV Al Watanya 1 and 2 Fono Smart Tunisie
Commerce électronique	We Discount Acombien.com
Transport	Audi Tunisie SRT Ennakl Automobile Smart Car SNCFT

1.2 Présentation du projet

1.2.1 Cadre général du projet

Dans un monde de plus en plus numérique, les entreprises doivent faire face à des défis majeurs en matière de gestion des environnements logiciels. La configuration et la mise à jour de ces environnements peuvent être une tâche fastidieuse, coûteuse et risquée, nécessitant souvent une intervention manuelle et une coordination complexe entre les différentes équipes.

De plus, les environnements logiciels sont de plus en plus complexes et hétérogènes, comprenant des systèmes d'exploitation différents, des applications tierces, des bases de données, des serveurs, des pare-feux, des contrôles d'accès, etc. Il est donc impératif pour les entreprises de trouver des solutions qui permettent d'automatiser la configuration et la gestion de ces environnements de manière efficace, sécurisée et évolutive. C'est dans ce contexte que s'inscrit notre projet de fin d'études.

1.2.2 Problématique

La configuration des environnements logiciels peut être complexe, nécessitant la configuration et l'interconnexion de nombreux éléments différents, tels que les serveurs, les réseaux et les applications. De plus, dans les environnements distribués, la configuration peut être répartie sur de nombreux sites géographiques, ce qui peut rendre la gestion manuelle encore plus difficile.

L'automatisation de la configuration peut aider à simplifier ce processus en permettant la définition de modèles de configuration et en les appliquant automatiquement à l'ensemble de l'infrastructure. Cette approche permet de réduire les tâches répétitives associées à la configuration manuelle, améliorant ainsi l'efficacité opérationnelle.

Un autre défi réside dans la visualisation des équipements à configurer. Dans des environnements distribués ou complexes, il peut être difficile d'avoir une vue d'ensemble claire de tous les éléments à configurer. L'automatisation de la configuration permet de centraliser et de visualiser facilement les équipements à configurer, facilitant ainsi la coordination et la planification.

La complexité est également un défi majeur dans la configuration logicielle. L'interconnexion de différents composants et la prise en compte de nombreuses variables peuvent conduire à des erreurs ou à des incohérences lors de la configuration manuelle. L'automatisation garantit une configuration cohérente et précise, minimisant les risques d'erreurs humaines et de dysfonctionnements.

En outre, l'automatisation de la configuration contribue à améliorer la scalabilité des environnements logiciels. À mesure que l'infrastructure se développe en taille et en complexité, la configuration manuelle devient de plus en plus difficile à

gérer efficacement. L'automatisation permet de déployer rapidement de nouvelles configurations et d'adapter l'infrastructure aux besoins changeants, favorisant ainsi la scalabilité.

En résumé, l'automatisation de la configuration des environnements logiciels permet de simplifier les tâches répétitives, de visualiser facilement les équipements à configurer, de réduire les erreurs humaines, de faire face à la complexité de la configuration et de favoriser la scalabilité. Elle offre une approche plus efficace et fiable pour gérer les environnements logiciels, améliorant ainsi l'efficacité opérationnelle et garantissant des configurations cohérentes et fiables.

1.2.3 Objectifs

Les principaux objectifs de notre projet sont :

- Réduire le temps et les coûts associés à la configuration et à la maintenance des systèmes informatiques en automatisant les tâches répétitives et en évitant les erreurs humaines.
- Offrir une interface utilisateur conviviale et facile à utiliser pour les administrateurs système, même pour ceux qui ne disposent pas de compétences techniques avancées.
- Prendre en charge une large gamme de systèmes d'exploitation et de plateformes matérielles, notamment les distributions Linux populaires et les environnements cloud comme AWS, Azure et Google Cloud.
- Fournir une solution évolutive qui peut être utilisée pour gérer des infrastructures de petite à grande échelle, avec la possibilité de gérer des milliers de machines en même temps.
- Assurer la sécurité des configurations en appliquant des bonnes pratiques et des politiques de sécurité lors de l'automatisation.
- Faciliter la collaboration et la gestion des versions en permettant la gestion centralisée des modèles de configuration.
- Améliorer la traçabilité et la documentation en enregistrant automatiquement les configurations appliquées.
- Optimiser les performances en automatisant la configuration des systèmes et en appliquant des paramètres optimisés.
- Réduire la complexité en offrant une vue d'ensemble consolidée de l'ensemble de l'infrastructure.
- Favoriser l'agilité et la flexibilité en permettant des déploiements rapides et reproductibles.

1.2.4 Solutions existantes

Les outils d'administration et configuration des systèmes les plus populaires sont Foreman et Cobbler. Nous décrivons les spécificités de chaque outil dans cette section finissant par établir une comparaison entre eux.

1.2.4.1 Foreman

Foreman est un projet open source qui aide les administrateurs système à gérer les serveurs tout au long de leur cycle de vie, du provisionnement et de la configuration à l'orchestration et à la surveillance (Figure 1.2 et 1.3). La prise en charge du provisionnement permet de contrôler facilement l'installation de nouveaux serveurs et, grâce à la gestion de la configuration (Puppet, Ansible, Chef et Salt sont pris en charge) et d'automatiser facilement les tâches répétitives. [8]

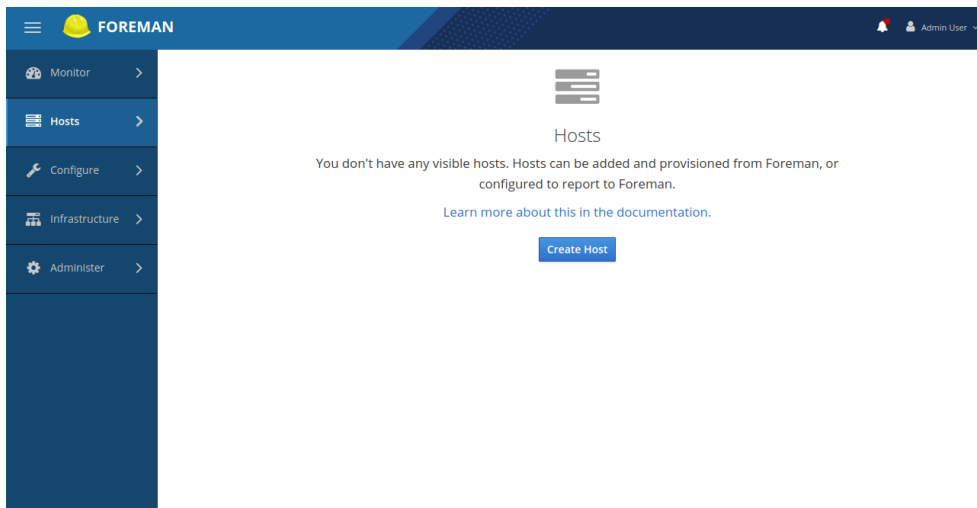


FIGURE 1.2 – Foreman Hosts

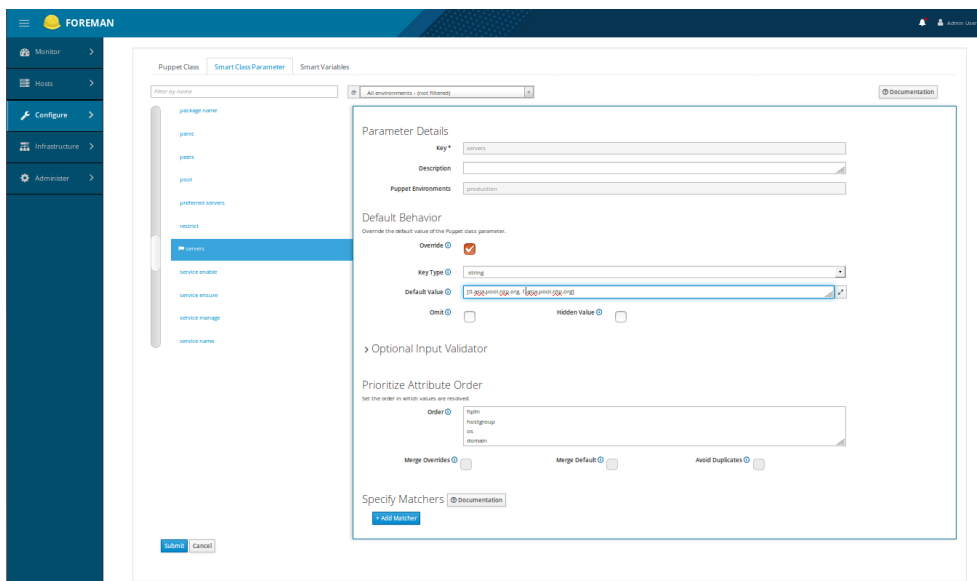


FIGURE 1.3 – Foreman configure

1.2.4.2 Cobbler

Cobbler est un outil qui automatise la configuration des environnements informatiques (Figure 1.4) . Il est spécifiquement conçu pour simplifier et automatiser le processus de déploiement et de gestion des systèmes d'exploitation sur un réseau. Cobbler permet de définir des modèles de configuration, d'effectuer des installations automatisées des systèmes d'exploitation, de gérer les configurations matérielles et logicielles, et d'appliquer des paramètres spécifiques à chaque machine.

En utilisant Cobbler, les administrateurs système peuvent centraliser la gestion des configurations, éviter les tâches manuelles répétitives et minimiser les risques d'erreurs humaines. L'outil offre également des fonctionnalités avancées telles que la gestion des versions, la traçabilité des configurations, la gestion de la sécurité et la facilité de déploiement sur différentes plates-formes matérielles et logicielles.

En résumé, Cobbler est un outil open source d'automatisation de la configuration qui permet aux administrateurs système de simplifier et de centraliser la gestion des environnements informatiques, en offrant des fonctionnalités avancées pour le déploiement et la configuration des systèmes d'exploitation. [9]

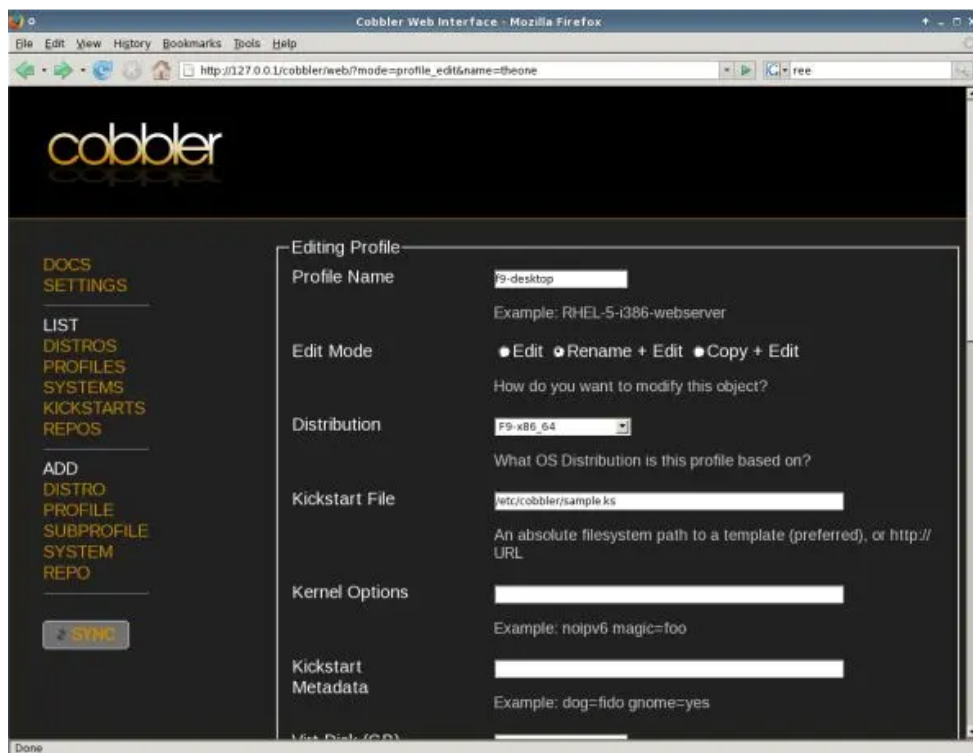


FIGURE 1.4 – Cobbler interface

1.2.5 Critiques des solutions existantes

Le tableau 1.2 donne une description de la comparaison entre Foreman et Cobbler en termes du cout, securité, personnalisation, support multiplateforme, gestion des ressources et Utilisation sur des machines aux capacités limitées.

Foreman est une solution open source, mais cela peut être un inconvénient en termes de sécurité, car il existe un risque potentiel de vulnérabilités du le code. De plus, certaines fonctionnalités de Foreman peuvent avoir une personnalisation et une prise en charge limitées. Foreman offre un support multiplateforme pour les systèmes d'exploitation RedHat et Debian et une large gamme d'applications, ce qui en fait une solution polyvalente pour la gestion des ressources du système informatique. Cependant, son utilisation sur des machines aux capacités limitées peut être difficile.

En ce qui concerne Cobbler, il s'agit également d'une solution open source. Il présente des risques similaires en termes de sécurité du code. Cependant, Cobbler offre une plus grande flexibilité en termes de personnalisation et de configuration. Il prend en charge une variété de systèmes d'exploitation et de plates-formes, et est principalement axé sur la gestion des configurations et du déploiement des systèmes d'exploitation. De plus, Cobbler peut être adapté pour une utilisation sur des machines aux capacités limitées.

TABLEAU 1.2 – Comparaison entre Foreman et Cobbler

Critères	Foreman	Cobbler
Coût	Open source et gratuit	Open source et gratuit
Sécurité	Risque potentiel de vulnérabilités avec le code	Risque potentiel de vulnérabilités avec le code
Personnalisation	Certaines fonctionnalités peuvent avoir une personnalisation et une prise en charge limitées	Personnalisation et configuration plus flexibles
Support multiplateforme	Supporte les systèmes d'exploitation RedHat et Debian et une large gamme d'applications	Supporte une variété de systèmes d'exploitation et de plates-formes
Gestion des ressources	Solution polyvalente pour la gestion des ressources du système informatique	Concentré sur la gestion des configurations et du déploiement des systèmes d'exploitation
Utilisation sur des machines aux capacités limitées	Peut être difficile à utiliser sur des machines aux capacités limitées	Peut être adapté pour une utilisation sur des machines aux capacités limitées

Bien que la concurrence soit plus riche en fonctionnalités et qu'elle ait été testée dans le cadre de divers scénarios, le développement d'un outil interne d'automatisation de l'infrastructure présente toujours un grand intérêt :

- L'extensibilité : Les logiciels libres sont souvent mal documentés ou ne disposent carrément pas de documentation sur le fonctionnement du code ou sur l'architecture qu'il suit.
- Sécurité : Les logiciels libres peuvent contenir des codes malveillants susceptibles de provoquer des fuites de données internes ou de saboter l'environnement hôte. L'utilisation d'un logiciel libre pour effectuer la tâche critique de mise en place des infrastructures peut conduire à la compromission potentielle de tous les environnements gérés.
- Complexité : La plupart des outils qui existent sur le marché sont trop complexes à utiliser, ce qui s'explique par le fait qu'ils sont développés pour répondre à de nombreux besoins à la fois.

En réponse à ces défis, nous avons entrepris le développement de notre propre plateforme d'automatisation de configuration 'ConfigAutomation'.

1.2.6 Solution proposée

"ConfigAutomation" est une solution de gestion de configuration qui permet d'automatiser la configuration et le déploiement des machines virtuelles et physiques. Elle permet aux administrateurs système de gérer efficacement les configurations de leur infrastructure informatique à grande échelle. La solution "ConfigAutomation" offre des fonctionnalités telles que la création de profils de configuration, la gestion des packages, la gestion des utilisateurs et des groupes, la configuration des services et la gestion des fichiers de configuration.

"ConfigAutomation" est facile à installer et à utiliser grâce à son interface utilisateur conviviale et intuitive. Elle prend en charge les principaux systèmes d'exploitation tels que Red Hat et Debian, et peut être utilisée pour gérer des configurations à la fois sur site et dans le cloud. La solution "ConfigAutomation" est entièrement open-source, ce qui permet aux utilisateurs de la personnaliser et de l'adapter à leurs besoins spécifiques.

En utilisant "ConfigAutomation", les administrateurs système peuvent automatiser la configuration de leurs machines, éviter les erreurs humaines et gagner du temps. La solution peut être utilisée pour déployer rapidement de nouveaux systèmes, ainsi que pour appliquer des modifications de configuration à grande échelle. En fin de compte, configAutomation permet aux entreprises de gérer efficacement leur infrastructure informatique et de garantir la cohérence de leurs configurations à grande échelle.

1.3 Méthode de gestion de projet

Une méthode de gestion de projet est une procédure qui rassemble des idées, des procédures et des méthodes. Ces stratégies doivent être utilisées afin de réaliser chaque étape du projet dans un environnement productif et rentable.

Cela incite l'ensemble de l'équipe à procéder à une nouvelle validation de l'ensemble des travaux réalisés. Mais il existe un cadre méthodologique plus souple, appelé méthodologie agile, que Medianet utilise pour construire ses projets.

1.3.1 Méthode agile

La méthodologie Agile est une façon de gérer un projet en le divisant en phases. Cela implique une collaboration continue avec les parties prenantes et une amélioration continue à chaque étape. Une fois que le travail commence, les équipes passent par un processus de planification, d'exécution et d'évaluation. La collaboration continue est essentielle, avec à la fois les membres de l'équipe et les parties prenantes du projet.

Les quatre valeurs principales d'Agile sont :

- Les individus et les interactions plutôt que les processus et les outils.
- Un logiciel fonctionnel plutôt qu'une documentation complète.
- La collaboration avec les clients plutôt que la négociation de contrats.
- Répondre au changement plutôt que suivre un plan.

Dans le cadre de notre projet, nous utiliserons SCRUM [3], l'une des méthodes agiles les plus connues. C'est ce que nous allons découvrir la prochaine sous-section du présent chapitre.

1.3.1.1 Méthode SCRUM

La méthode SCRUM [3] est une variante de la méthodologie agile, qui se caractérise par des processus itératifs et incrémentaux. Nous avons utilisé cette méthode pour améliorer la productivité de notre équipe. Pour ce faire, nous avons découpé le projet en périodes de temps appelées "Sprints", d'une durée de 1 à 4 semaines, avec un rythme constant. Chaque Sprint débute par une estimation, suivie d'une planification opérationnelle, et se termine par une démonstration des réalisations, appelée rétrospective, qui nous permet d'améliorer nos pratiques. Ce cycle, représenté dans la figure 1.5, se répète jusqu'à ce que toutes les tâches de la liste des "Product Backlog" soient accomplies.

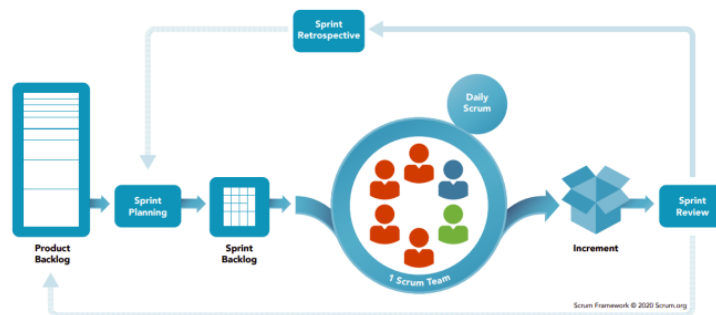


FIGURE 1.5 – Cycle de vie de SCRUM [4]

Rôles des acteurs

La méthode Scrum regroupe généralement trois acteurs principaux que nous avons illustrés dans le tableau 1.3

TABLEAU 1.3 – L'équipe Scrum

Rôle	Description	Acteur
Product owner	Pour le compte du client, définir le périmètre du projet et compiler les fonctionnalités requises par l'utilisateur sous forme de "user stories"	Mohamed Zied TLIL
Scrum master	Responsable de la gestion de projet et de la gestion de l'équipe	Mohamed Zied TLIL
Scrum teams	Réunissant des équipes de développeurs, d'ingénieurs, de designers, etc.	Achraf YAAKOUB

Artefacts

Les artefacts représentent un outil clé de la méthode SCRUM pour faciliter la gestion Projets et équipes. Les trois constantes sont les suivantes :

- **Product Backlog** : est une liste des principaux projets, fonctionnalités, demandes, améliorations et correctifs à apporter. Il est supporté par un « Product Owner » ou un « Product Manager » et agit comme une entrée dans le backlog de sprint.
- **Sprint Backlog** : est une liste d'éléments comprenant des user stories ou des corrections de bugs qui seront développés ou réalisés pendant le cycle de sprint en cours. Il est important de souligner que le backlog de sprint est flexible et peut être modifié au cours du sprint.
- **Increment Product** : est un outil qui permet à l'équipe d'évaluer ce qui a été accompli pendant le sprint.

1.4 Suivre de la progression du projet

Le diagramme de Gantt est un outil qui facilite l'organisation, la planification et le suivi de l'avancement d'un groupe de tâches d'un projet. La figure 1.6 illustre la décomposition en semaine de notre projet. Le projet dure 16 semaines comprenant l'analyse et la spécification et une formation Ansible (5 semaines), le développement des rôles et de la plateforme et la phase de test (9 semaines) et la rédaction du rapport final (2 semaines).

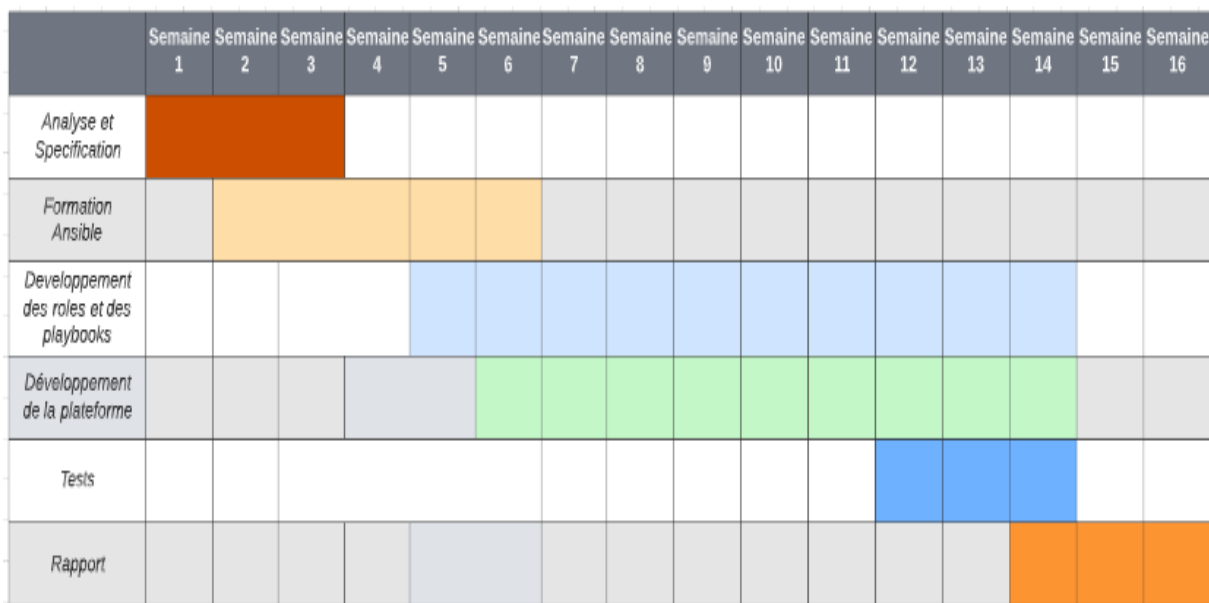


FIGURE 1.6 – Diagramme de Gantt du projet

1.5 Outils adoptés pour la gestion de projet

Pour le bon déroulement des projets et la bonne livraison de nos produits, nous utilisons certains outils de gestion de projet tels que :

- Git : un système de contrôle de version utilisé pour suivre les modifications apportées aux fichiers de code et coordonner le travail de plusieurs personnes sur ces fichiers.
- Trello : Outil open source de gestion de projet. La figure 1.7 Présente les tâches effectuées lors du projet sous Trello.

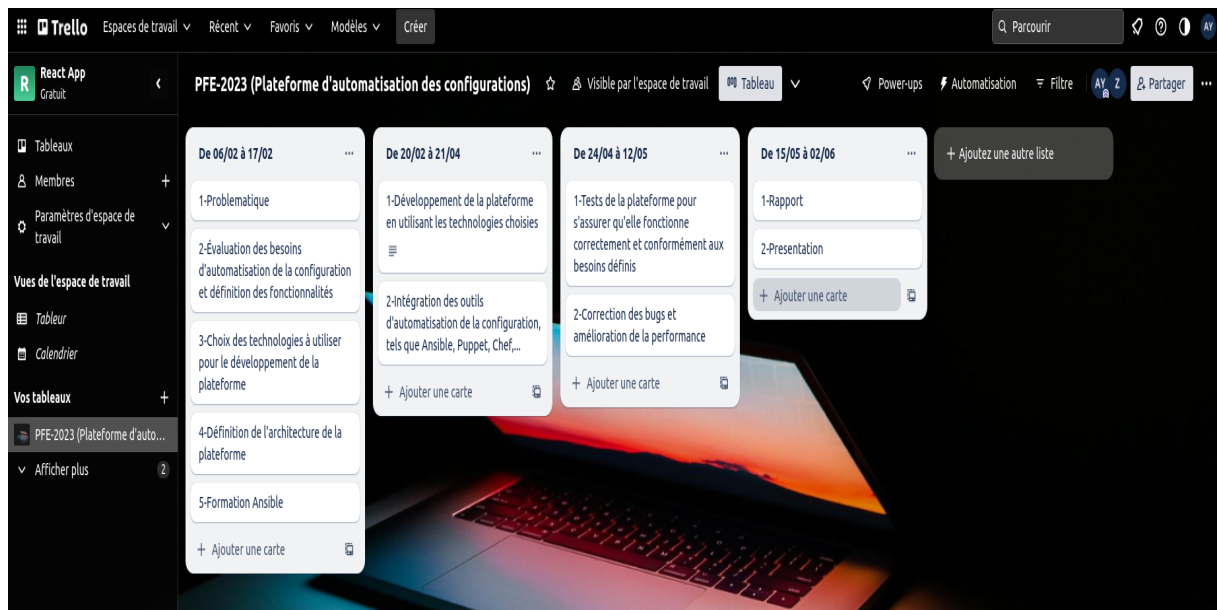


FIGURE 1.7 – Tâches sous Trello

- Drive : Il permet de partager les documents avec les membres de l'équipe comme décrit dans la figure 1.8.

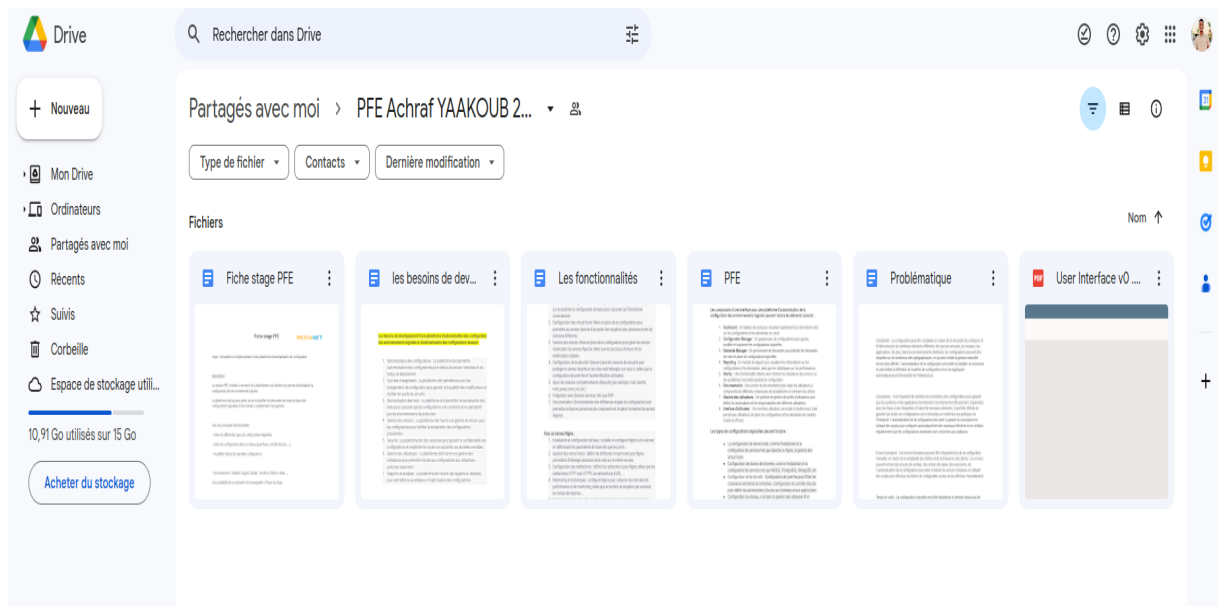


FIGURE 1.8 – Documents partagés sous Drive

Conclusion

Ce chapitre a débuté par une brève présentation de l'entreprise d'accueil Medianet. Ensuite, nous avons exposé le cadre général, la problématique, l'existant et ses limites et la solution proposée. Puis, nous avons décrit la philosophie de travail et de gestion de projet adoptée par l'entreprise. Enfin, on a listé les outils utilisés pour la gestion de ce projet.

Introduction

Ce chapitre est consacré à décrire les concepts clés sur lesquels repose notre projet en relations avec DevOps, Automatisation, L'infrastructure en tant que code et les outils d'automatisations.

2.1 Concepts Généraux

Dans le cadre d'une étude approfondie de notre solution, nous allons présenter différents concepts :

2.1.1 DevOps

DevOps [5] est un ensemble de philosophies , de pratiques et d'outils qui aident les organisations à fournir de meilleurs produits plus rapidement en intégrant plus facilement les fonctions de développement et d'exploitation. Le terme DevOps est une combinaison des termes développement et opérations, et vise à représenter une approche collaborative ou partagée dans laquelle les équipes de développement d'applications et d'exploitation informatique d'une entreprise exécutent des tâches. Il est associé à des principes définis dans la Figure 2.1.

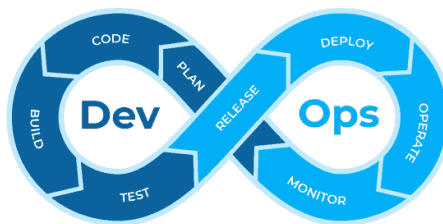


FIGURE 2.1 – Cycle DevOps [6]

2.1.2 Approvisionnement en logiciels

L'approvisionnement en logiciels fait référence au processus consistant à rendre les logiciels disponibles et prêts à être utilisés dans un environnement informatique particulier. Il implique l'installation, la configuration et la mise en place d'applications logicielles sur différents systèmes matériels, réseaux et serveurs.

Voici quelques exemples d'approvisionnement en logiciels :

- Installation et configuration du système d'exploitation sur un nouvel ordinateur ou serveur.
- Installation et configuration d'un logiciel de gestion de base de données sur un serveur.
- Mise en place d'un serveur web avec le logiciel et la configuration nécessaires.
- Installation et configuration d'un logiciel d'application tel qu'un logiciel de gestion de la relation client (CRM) ou un logiciel de planification des ressources de l'entreprise (ERP) sur un serveur ou un réseau.
- Fournir des mises à jour logicielles ou des correctifs aux systèmes existants.

Globalement, l'approvisionnement en logiciels est le processus qui consiste à s'assurer que les applications logicielles sont correctement installées, configurées et prêtes à fonctionner dans un environnement spécifique.

L'approvisionnement en logiciels peut être effectué manuellement, mais il peut aussi être automatisé.

2.1.3 Automatisation

L'automatisation fait référence à l'utilisation de technologies et de logiciels pour effectuer des tâches et des processus automatiquement, sans nécessiter d'intervention humaine.

Dans le contexte du développement de logiciels et des opérations informatiques, l'automatisation fait référence à l'utilisation d'outils, de scripts et de flux de travail pour automatiser des tâches telles que la création, le test, le déploiement et la gestion d'applications logicielles et de ressources d'infrastructure. L'automatisation peut aider les équipes à améliorer la vitesse, l'efficacité et la qualité de leurs processus de développement de logiciels, tout en réduisant le risque d'erreur humaine.

Voici quelques exemples d'automatisation dans le développement de logiciels et les opérations informatiques :

- Les outils de gestion de la configuration, qui automatisent le déploiement et la gestion des ressources d'infrastructure à l'aide de fichiers de configuration codés.

- Les outils de surveillance et d’alerte, qui détectent automatiquement les problèmes et les erreurs dans les applications logicielles et les ressources d’infrastructure et y répondent.

L’approvisionnement en logiciels est un processus qui bénéficie grandement de l’automatisation, car il est hautement répétitif, mais aussi délicat et sujet aux erreurs.

2.1.4 L’infrastructure en tant que service

L’infrastructure en tant que service (IaaS) est un modèle commercial qui fournit une infrastructure informatique telle que des ressources informatiques, de stockage et de mise en réseau sur une base de paiement à l’utilisation sur Internet. Vous pouvez utiliser IaaS pour demander et provisionner les ressources nécessaires pour exécuter des applications et des systèmes informatiques. Vous êtes responsable du déploiement, de la maintenance et du support de votre application, tandis que le fournisseur IaaS gère l’infrastructure physique. L’infrastructure en tant que service vous offre un contrôle flexible sur les ressources informatiques de manière rentable.

2.1.5 L’infrastructure en tant que code

L’infrastructure en tant que code (IaC) est une pratique de développement logiciel qui consiste à définir et à gérer l’infrastructure et les ressources informatiques à l’aide de fichiers de configuration et d’outils basés sur le code. Cette approche permet aux équipes d’automatiser la création, le déploiement et la gestion des ressources d’infrastructure telles que les serveurs, les réseaux, le stockage et les politiques de sécurité.

L’IaC traite l’infrastructure comme s’il s’agissait d’un code logiciel, en utilisant des systèmes de contrôle de version et des cadres de test pour s’assurer que les changements apportés à l’infrastructure sont effectués de manière contrôlée et reproductible. En définissant l’infrastructure comme du code, les équipes peuvent facilement reproduire et faire évoluer les environnements, réduire l’erreur humaine et améliorer la vitesse et la souplesse de leurs processus de développement et d’exploitation.

Terraform, CloudFormation et Ansible sont quelques exemples d’outils IaC. Ces outils permettent aux développeurs et aux équipes d’exploitation de définir les ressources d’infrastructure à l’aide d’un langage déclaratif de haut niveau, puis de déployer et de gérer automatiquement ces ressources dans des environnements cloud tels qu’Amazon Web Services (AWS), Microsoft Azure ou Google Cloud Platform (GCP).

L'IaC fait passer le concept d'automatisation au niveau supérieur. L'écriture manuelle de scripts pour installer et configurer un logiciel est en soi une tâche répétitive qui prend beaucoup de temps et qui est sujette aux erreurs des développeurs. Cela ouvre la porte au développement d'outils logiciels qui gèrent l'aspect répétitif du provisionnement automatisé et laissent l'aspect personnalisé à instruire à l'aide d'un langage de configuration de haut niveau tel que YAML.

2.1.6 Comparaison entre IaaS et IaC

Le tableau 2.1 donne une description de la comparaison entre IaaS et IaC.

TABLEAU 2.1 – Comparaison IaaS vs IaC

IaaS (Infrastructure as a Service)	IaC (Infrastructure as Code)
Le fournisseur de cloud gère l'infrastructure	L'infrastructure est gérée par le code
Les utilisateurs ont un contrôle limité sur l'infrastructure	Les utilisateurs ont un contrôle total sur l'infrastructure
Les utilisateurs doivent gérer manuellement les ressources	Les ressources sont gérées automatiquement
Les coûts sont basés sur l'utilisation des ressources	Les coûts sont basés sur le code et l'automatisation
L'extensibilité dépend de la capacité du fournisseur de cloud	L'extensibilité est illimitée

2.2 L'infrastructure de l'entreprise

L'infrastructure informatique joue un rôle vital dans les entreprises modernes. Il se compose d'un ensemble de composants technologiques fondamentaux qui prennent en charge les opérations quotidiennes, les communications, le stockage des données et la sécurité. Cette infrastructure comprend généralement des services Web, des bases de données, des pare-feu, services de monitoring et d'autres éléments critiques nécessaires au bon fonctionnement de l'entreprise.

2.2.1 Services web

Un service web est un logiciel qui permet de gérer les requêtes HTTP, afin de fournir des contenus dynamiques aux utilisateurs. Les serveurs web tels que Nginx et Apache sont largement utilisés pour héberger des sites web et des applications web.

2.2.1.1 Varnish

Varnish est un serveur de mise en cache qui stocke les pages Web dans le cache pour servir rapidement les utilisateurs. Il est généralement utilisé en conjonction

avec un serveur Web tel que Nginx ou Apache, et stocke les pages Web dans la RAM pour un accès rapide. Varnish est conçu pour améliorer les performances des sites Web très chargés et réduire la charge du serveur.

2.2.1.2 Apache

Apache est un serveur web populaire qui peut être utilisé pour servir des pages web statiques ou dynamiques. Il est également largement utilisé pour exécuter des scripts PHP et est compatible avec de nombreux modules et extensions pour répondre à différents besoins.

2.2.1.3 Nginx

Nginx est un serveur web léger et performant qui peut être utilisé pour servir du contenu statique ou dynamique. Il est également utilisé comme proxy inverse pour équilibrer la charge entre les serveurs ou pour gérer des connexions SSL.

2.2.2 Services de base de données

Les services de base de données sont utilisés pour stocker, gérer et manipuler des données. Les bases de données peuvent être divisées en deux grandes catégories :

2.2.2.1 Les bases de données relationnelles (SQL)

Les bases de données relationnelles sont les plus courantes et sont basées sur le modèle de données relationnelles. Les données sont stockées dans des tables, où les enregistrements contiennent des valeurs pour des colonnes spécifiques. Les relations entre les tables sont établies en définissant des clés primaires et étrangères. MySQL, MariaDB, PostgreSQL sont des exemples de systèmes de gestion de bases de données relationnelles. Les bases de données relationnelles sont utilisées dans de nombreuses applications, notamment les systèmes de gestion de contenu, les systèmes de gestion de la relation client (CRM) et les systèmes de gestion des stocks.

MySQL

MySQL est un système de gestion de base de données relationnelle open source développé par Oracle Corporation. Il est largement utilisé dans les sites Web et les applications Web. MySQL prend en charge le langage SQL et fournit des fonctionnalités avancées telles que la réplication, la haute disponibilité et l'évolutivité.

MariaDB

MariaDB est un système de gestion de base de données open-source, dérivé de MySQL. Il est compatible avec MySQL et prend en charge le langage SQL. MariaDB offre des fonctionnalités telles que la réplication, la haute disponibilité et la scalabilité.

PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle gratuit et open source. Il prend en charge le langage SQL et fournit des fonctionnalités avancées telles que les transactions ACID, la réplication, l'indexation de texte intégral, les fonctions stockées et les vues matérialisées.

2.2.2.2 Les bases de données non relationnelles (NoSQL)

Les bases de données non relationnelles, quant à elles, stockent les données dans un format différent de celui des tables relationnelles, généralement à l'aide de documents, de graphiques ou de paires clé-valeur. Les bases de données non relationnelles sont généralement utilisées pour les applications qui stockent de grandes quantités de données non structurées et nécessitent une évolutivité horizontale. MongoDB, Elasticsearch sont des exemples de systèmes de gestion de bases de données non relationnelles. Les bases de données non relationnelles sont également appelées NoSQL.

MongoDB

MongoDB est un système de gestion de base de données non relationnelles open-source et gratuit. Il stocke des données dans des documents BSON (Binary JSON) et prend en charge des fonctionnalités telles que la réplication, la haute disponibilité, la scalabilité et les index géospatiaux.

Elasticsearch

Elasticsearch [7] est un système open source de recherche et d'analyse de données distribuées basé sur Apache Lucene. Il stocke les données dans des index de documents JSON et prend en charge des fonctionnalités telles que la recherche en texte intégral, la recherche géospatiale, la réplication et la haute disponibilité.

2.2.3 Services pare-feux

Les services pare-feux sont utilisés pour protéger les réseaux informatiques contre les attaques malveillantes. Les pare-feux peuvent être configurés pour blo-

quer ou autoriser les connexions réseau en fonction de divers critères tels que les adresses IP source et destination, les ports, les protocoles .

2.2.3.1 UFW

UFW (Uncomplicated Firewall) est un pare-feu pour les systèmes basés sur Linux qui permet aux administrateurs système de configurer des règles de pare-feu à l'aide d'une syntaxe simplifiée. Il permet de bloquer ou d'autoriser les connexions basées sur les ports, les adresses IP et les protocoles, ainsi que les connexions à des services spécifiques tels que SSH, HTTP, HTTPS UFW est couramment utilisé pour protéger les serveurs Web contre les attaques malveillantes et les tentatives d'intrusion.

2.2.4 Services de monitoring

Les services de monitoring sont utilisés pour surveiller les performances des systèmes informatiques et détecter les problèmes potentiels. Parmi les exemples de services de monitoring, on peut citer Ganglia.

2.2.4.1 Ganglia

Ganglia est un système de surveillance distribué et évolutif pour la surveillance des ressources système et des applications. Il est utilisé dans les environnements informatiques hautes performances tels que les grilles de calcul et les centres de données. Ganglia utilise une architecture client-serveur pour envoyer des données de surveillance à un ou plusieurs serveurs de surveillance. Les données sont stockées dans une base de données distribuée et peuvent être consultées en temps réel via une interface Web. Ganglia prend également en charge les alertes et les rapports pour faciliter la détection des problèmes et la planification des capacités.

2.2.5 Cloud computing

Le cloud computing a révolutionné la gestion de l'infrastructure informatique des entreprises. Il permet d'accéder à distance à des ressources informatiques via Internet, telles que le stockage, les serveurs et les applications. Les entreprises peuvent ajuster rapidement leurs ressources en fonction de leurs besoins, sans investir dans une infrastructure coûteuse. Le cloud offre une accessibilité accrue, favorisant la collaboration et la mobilité des employés. La sécurité est une priorité, avec des mesures de protection avancées pour préserver la confidentialité des données. En résumé, le cloud computing offre aux entreprises une solution flexible, rentable et évolutive pour gérer leur infrastructure informatique. L'entreprise utilise souvent des fournisseurs de services cloud de premier plan tels que Google Cloud

Platform (GCP), Microsoft Azure et Amazon Web Services (AWS) pour répondre à leurs besoins spécifiques en matière d'infrastructure et de services informatiques.

2.2.5.1 Google Cloud Platform (GCP)

Google Cloud Platform est l'un des principaux fournisseurs de services cloud, proposant des solutions complètes aux entreprises. GCP est connu pour ses capacités d'analyse de données, son expertise en IA et sa vaste infrastructure mondiale. Les entreprises utilisent GCP pour une évolutivité élevée, une haute disponibilité et des performances optimales. GCP fournit également des services tels que le stockage, les bases de données, la mise en réseau et les outils de développement, offrant une suite complète pour créer et gérer des applications cloud. En mettant l'accent sur l'innovation et la technologie de pointe, GCP est un choix populaire pour les entreprises à la recherche d'une solution cloud robuste et fiable.

2.2.5.2 Amazon Web Services (AWS)

Amazon Web Services a été l'un des pionniers du cloud et reste l'un des fournisseurs les plus populaires. AWS fournit une large gamme de services cloud, notamment l'informatique, le stockage, la mise en réseau, la sécurité, etc. AWS se distingue par son échelle mondiale, offrant une haute disponibilité et une excellente évolutivité. Les entreprises bénéficient de la flexibilité d'AWS pour créer, déployer et gérer leurs applications plus facilement et plus efficacement. AWS propose également des fonctionnalités avancées de sécurité, de conformité et de gestion des coûts, ce qui en fait un choix populaire pour les entreprises de toutes tailles, des startups aux grandes entreprises, dans de nombreux secteurs.

2.2.5.3 Microsoft Azure

Microsoft Azure est une plate-forme complète de cloud computing fournie par Microsoft. Azure se distingue par son intégration étroite avec les produits Microsoft tels que Windows Server et SQL Server, ce qui facilite la migration des applications existantes vers le cloud. Azure propose une large gamme de services, notamment le calcul, le stockage, l'IA, l'IoT... . Les entreprises apprécient la flexibilité d'Azure pour faire évoluer les ressources selon les besoins, ainsi que son écosystème de services complémentaires, tels que la sécurité, l'analyse et la gestion des identités. Avec sa large portée mondiale et ses capacités pour les entreprises de toutes tailles, Azure est un choix solide pour les entreprises qui cherchent à tirer parti du cloud.

2.3 L'outil de gestion et d'automatisation de la configuration : Ansible

L'infrastructure décrite dans la section précédente est configuré avec ansible. Nous détaillons dans ce qui suit l'architecture de cet outil ainsi que ces concepts.

2.3.1 Architecture Ansible

Ansible est un moteur d'automatisation simple qui automatise le cloud, le provisionnement, gestion de configuration, déploiement d'applications, orchestration de services et que de nombreuses autres fonctionnalités. Dans cette section, nous donnerons un aperçu du fonctionnement d'Ansible. La figure 2.2 présente l'architecture Ansible.

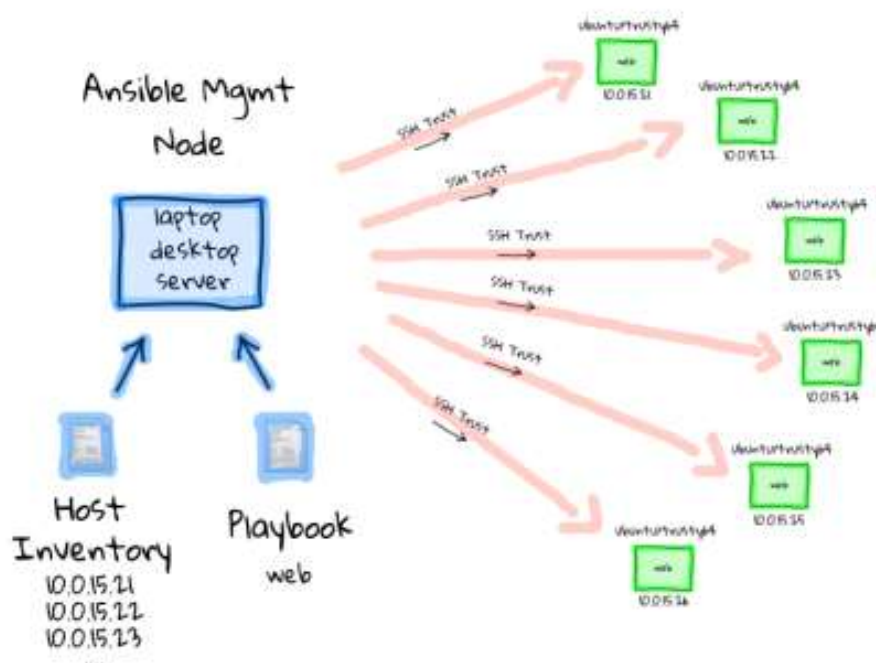


FIGURE 2.2 – Architecture Ansible. [2]

2.3.2 Modules

Ansible fonctionne en se connectant aux nœuds et en poussant des applets, Ceux-ci sont appelés "Ansible Modules". Ces programmes sont écrits sous forme de modèles. Une ressource pour l'état souhaité du système. Ansible exécute ensuite les modules (via SSH par défaut) et supprimez-le lorsque vous avez terminé. La bibliothèque de modules peut résider sur n'importe quel ordinateur et ne nécessite aucun serveur, démon ou base de données.

2.3.3 Inventaires

L'inventaire fournit une liste d'hôtes (serveurs gérés) sur lesquels Ansible s'exécutera Tâche. Il peut également être utilisé pour regrouper des hôtes et configurer des variables hôtes et groupes. La figure 2.3 montre un exemple de fichier d'inventaire.

```
1 ---
2 [webservers]
3 35.238.65.194 ansible_user=achrafyaakoubpfe
4 34.136.104.27 ansible_user=achrafyaakoubpfe
5
6 [dbservers]
7 34.29.170.63 ansible_user=achrafyaakoubpfe
8 34.123.240.162 ansible_user=achrafyaakoubpfe
```

FIGURE 2.3 – Exemple fichier Ansible Inventory.

2.3.4 Playbooks

Les playbooks sont le langage de configuration d'Ansible utilisé pour effectuer toutes les tâches et étapes de configuration ou pour appliquer des politiques sur des nœuds distants. Les playbooks utilisent des modules Ansible pour effectuer des actions. Les playbooks sont écrits en YAML, un langage simple et lisible par l'homme, et développés avec l'anglais de base comme langue de texte. Les playbooks sont plus susceptibles d'être conservés dans un système de contrôle de version pour maintenir les configurations du système distant. Les playbooks sont utilisés pour tout, de la simple configuration et gestion des nœuds distants aux déploiements avancés impliquant des mises à jour, la surveillance des serveurs, l'équilibrage de charge, etc. La figure 2.4 présente un exemple de playbook pour l'installation d'Apache.

```
---
- name: install and start Apache webserver
  hosts: webservers
  user: root

  tasks:
    name: install httpd
    yum: name=httpd state=present

    name: start httpd
    service: name=httpd state=running
```

FIGURE 2.4 – Exemple Playbook pour installation Apache.

2.3.5 Rôles

Un rôle Ansible est une unité organisationnelle dans Ansible qui encapsule les tâches, les fichiers de configuration et d'autres éléments nécessaires pour atteindre un objectif spécifique. Il permet de définir une structure modulaire pour automatiser des tâches complexes et répétables.

2.3.5.1 Structure du rôle

La figure 2.5 montre la structure d'un rôle ansible.

```
myrole/
├─ tasks/
│   └─ main.yml
│   └─ ...
├─ handlers/
│   └─ main.yml
│   └─ ...
├─ templates/
│   └─ template1.j2
│   └─ ...
├─ files/
│   └─ file1
│   └─ ...
├─ vars/
│   └─ main.yml
│   └─ ...
├─ defaults/
│   └─ main.yml
│   └─ ...
├─ meta/
│   └─ main.yml
└─ README.md
```

FIGURE 2.5 – Structure du rôle.

Chaque répertoire a un rôle spécifique dans la structure du rôle :

- Le répertoire 'tasks' contient les tâches principales du rôle.
- Le répertoire 'handlers' contient les gestionnaires d'événements du rôle.
- Le répertoire 'templates' contient les modèles de fichiers à générer.
- Le répertoire 'files' contient les fichiers à copier sur les systèmes cibles.
- Le répertoire 'vars' contient les variables spécifiques au rôle.
- Le répertoire 'defaults' contient les valeurs par défaut des variables.
- Le répertoire 'meta' contient les métadonnées du rôle.

Conclusion

Dans ce chapitre, nous introduisons les concepts clés utilisés pour mettre en œuvre notre système. Puis, nous avons aussi présenté l'infrastructure de l'entreprise. Nous avons clôturé par un découvert de l'outil d'automatisation Ansible.

CHAPITRE 3

ANALYSE ET SPÉCIFICATION DES BESOINS

Introduction

Ce chapitre est dédié à la description de la phase d'analyse et spécifications des besoins fonctionnels et non fonctionnels en utilisant des modèles conceptuels. Il met l'accent sur les besoins de notre projet. C'est dans ce contexte que nous avons énuméré les acteurs et les cas d'utilisations présentant les principales tâches de chaque acteur. Ainsi nous donnons quelques scénarios détaillés. Dans ce chapitre aussi nous citons les besoins fonctionnels et non fonctionnels de notre projet finissons par planifier les prochains sprints.

3.1 Les acteurs

Avant d'énumérer les besoins de notre système, nous allons d'abord identifier les principaux acteurs impliqués. Ces acteurs sont :

- **Administrateur système** : Cet acteur est l'utilisateur principal du système, responsable de la configuration et de la maintenance des serveurs. Il utilisera notre solution pour configurer les serveurs Debian 10, Debian 11, RedHat 8 et RedHat 9.
- **Système** : Il s'agit de notre solution avec laquelle nous allons effectuer les configurations. Le système sera responsable de l'automatisation et de la gestion des configurations des serveurs.
- **Serveurs** : Les serveurs Debian 10, Debian 11, RedHat 8 et RedHat 9 sont également considérés comme des acteurs dans notre système. Ils sont les destinataires des configurations automatisées et doivent être configurés de manière cohérente et efficace.

3.2 Identification des besoins

Le système proposé doit répondre à deux types d'exigences : fonctionnelles et non fonctionnelles. Cette section identifie ces exigences.

3.2.1 Spécification des besoins fonctionnels

Les différents besoins de notre système sont :

- Authentification : Authentification de l'Admin pour accéder à la plateforme.
- Gestion des configurations des serveurs : Création, modification et suppression de configurations logicielles.
- Exécution automatisée des configurations : Déclenchement de l'exécution des configurations en fonction de différents événements, tels que des déclencheurs temporels ou des actions spécifiques.
- Gestion des environnements cibles (Clients) : Définition des environnements cibles (par exemple, des serveurs ou des machines virtuelles) sur lesquels les configurations doivent être déployées.
- Consultation des services : Permet à l'admin de visualiser les services qu'il peut utiliser à configurer les serveurs.
- Consultation des logs : Permet à l'admin de consulter les logs des exécutions passées pour le suivi et le dépannage.
- Gestion des vhosts : Permet à l'admin de gérer les configurations des virtual hosts pour les serveurs web.
- Consultation des rôles Ansible : Permet à l'admin de visualiser la liste des rôles Ansible disponibles dans la plateforme.

3.2.2 Fonctionnalités du système

Dans cette section, nous détaillons les diagrammes de cas d'utilisations afin de présenter les fonctionnalités de notre système. Puis nous détaillons quelques unes.

3.2.2.1 Diagramme de cas d'utilisation général

Nous allons dans cette section décrire les principales fonctionnalités offertes par notre système de gestion de configuration. Après qu'un administrateur IT s'est authentifié, il dispose d'un accès complet à toutes les fonctionnalités du système. Il peut gérer les différents paramètres de configuration, créer, modifier et supprimer

des machines virtuelles, ainsi que gérer les utilisateurs et leurs rôles. En outre, l'administrateur peut consulter les logs d'activité pour surveiller les actions des utilisateurs et détecter les éventuelles tentatives de violation de la sécurité.

Toutes les fonctionnalités du système sont détaillées dans la figure 3.1 :

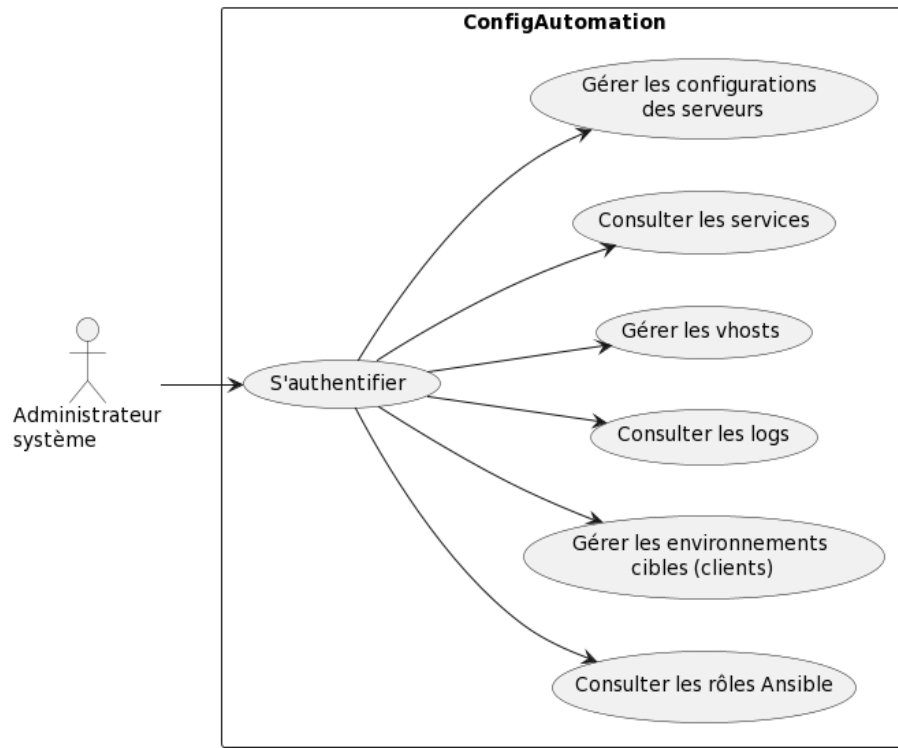


FIGURE 3.1 – Diagramme de cas d'utilisation des principales fonctionnalités.

Le tableau 3.1 illustre la description textuelle du cas d'utilisation des principales fonctionnalités à travers lequel l'administrateur peut générer plusieurs fonctionnalités.

TABLEAU 3.1 – Description textuelle du cas d'utilisation des principales fonctionnalités

Intitulé	Diagramme de cas d'utilisation des principales fonctionnalités
ID	UC-01
Acteur principal	Administrateur système
Acteur secondaire	Aucun
Description	Ce cas d'utilisation permet à l'administrateur système d'accéder aux principales fonctionnalités de la plateforme d'automatisation des configurations une fois qu'il est authentifié.
Pré-condition	L'administrateur est authentifié sur la plateforme.
Post-condition	L'administrateur peut utiliser les principales fonctionnalités de la plateforme.
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur accède à la page principale de la plateforme. 2. L'administrateur visualise les services disponibles pour la configuration des serveurs. 3. L'administrateur gère les configurations des serveurs en créant, modifiant ou supprimant des configurations logicielles. 4. L'administrateur déclenche l'exécution automatisée des configurations en fonction de différents événements, tels que des déclencheurs temporels ou des actions spécifiques. 5. L'administrateur gère les environnements cibles (Clients) en définissant les environnements sur lesquels les configurations doivent être déployées. 6. L'administrateur consulte les logs des exécutions passées pour le suivi et le dépannage. 7. L'administrateur gère les configurations des virtual hosts pour les serveurs web. 8. L'administrateur visualise la liste des rôles Ansible disponibles dans la plateforme.
Scénario d'erreur	En cas de problème, le système affiche un message d'erreur.

3.2.2.2 Gestion de Configuration des Serveurs

La gestion de la configuration des serveurs est une fonctionnalité clé de notre plateforme d'automatisation. Elle permet aux administrateurs de créer, modifier, supprimer et afficher de manière centralisée les configurations logicielles du serveur. Cette fonctionnalité est conçue pour simplifier la gestion de la configuration, assurant ainsi la cohérence et la fiabilité de l'environnement logiciel déployé sur le serveur. Le diagramme de cas d'utilisation de la figure 3.2 illustre cette fonctionnalité.

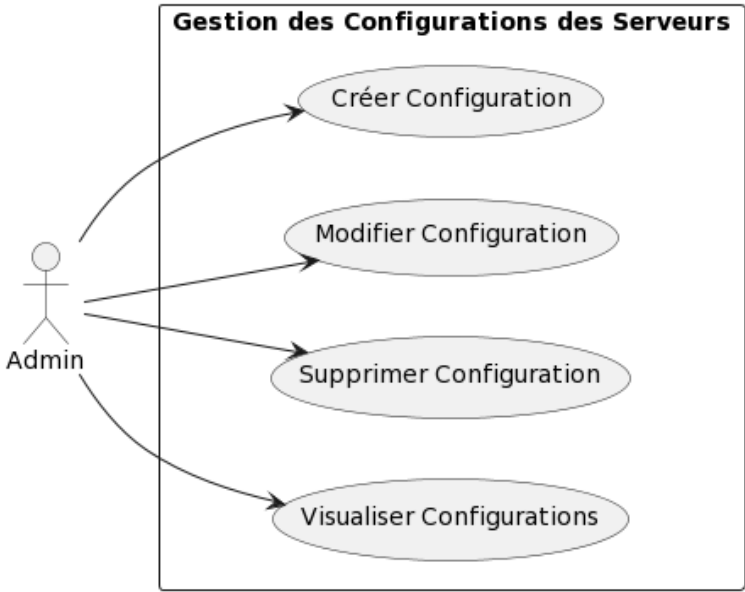


FIGURE 3.2 – Diagramme de cas d’utilisation de gestion des configurations.

Le tableau 3.2 illustre la description textuelle du cas d’utilisation "Gestion de la configuration des serveurs" à travers lequel l’administrateur peut générer les configurations nécessaires.

TABEAU 3.2 – Description textuelle du cas d’utilisation de "Gestion de la configuration des serveurs"

Intitulé	Gestion de la configuration des serveurs
ID	UC-02
Acteur principal	Administrateur système
Acteur secondaire	Aucun
Description	Permet à l’administrateur système de créer, modifier, supprimer et afficher les configurations logicielles des serveurs.
Pré-condition	L’administrateur est authentifié sur la plateforme.
Post-condition	Les configurations logicielles sont créées, modifiées, supprimées ou affichées selon les actions de l’administrateur.
Scénario nominal	1.Le système affiche l’interface de création de configuration. 2. L’administrateur sélectionne le serveur à configurer. 3. L’administrateur choisi les configurations . 4. L’administrateur valide la création de la configuration. 5. Le système enregistre la configuration logicielle. 6. Le système affiche un message de succès.
Scénario d’erreur	En cas de problème, le système affiche un message d’erreur.

3.2.3 Scénarios des fonctionnalités du système

Dans cette section, nous détaillons par les diagrammes de séquence quelques cas d'utilisation afin de présenter la relation et l'interaction entre les composants et le déroulement pas à pas de chaque fonctionnalité.

3.2.3.1 Authentification

Lorsqu'un utilisateur souhaite accéder à notre système, il doit passer par le processus d'authentification sécurisé mis en place grâce à Spring Security. Cette technologie offre une protection robuste contre les tentatives d'accès non autorisées.

Lorsqu'un utilisateur soumet son nom d'utilisateur et son mot de passe, ces informations sont envoyées au serveur via une requête d'authentification. Le module Spring Security intervient alors pour vérifier ces informations par rapport à celles stockées dans la base de données. Si les informations fournies sont correctes, l'authentification est considérée comme réussie.

En cas d'authentification réussie, le serveur génère un jeton JWT (JSON Web Token) contenant des informations d'identification telles que le nom d'utilisateur et les rôles d'utilisateur. Ce jeton est ensuite renvoyé au client, qui le stocke localement. Le client utilise ce jeton pour inclure les informations d'identification dans les requêtes ultérieures afin d'accéder aux ressources protégées du système.

En revanche, si l'authentification échoue en raison de mauvaises informations d'identification, Spring Security génère une réponse d'erreur appropriée qui est renvoyée au client. Le client affiche ensuite le message d'erreur à l'utilisateur, l'informant que l'authentification a échoué et lui demandant de fournir des informations d'identification valides.

L'utilisation de Spring Security dans notre processus d'authentification garantit une couche supplémentaire de sécurité en appliquant des mécanismes tels que le hachage des mots de passe, la protection contre les attaques par force brute et la gestion des sessions.

En résumé, grâce à Spring Security, notre système offre une authentification sécurisée en vérifiant les informations d'identification fournies par l'utilisateur et en générant des jetons JWT pour l'accès ultérieur aux ressources protégées. Cela permet de garantir que seuls les utilisateurs autorisés peuvent accéder à notre système et d'assurer la confidentialité et l'intégrité des données.

Le diagramme de séquence de la figure 3.2 illustre le processus d'authentification.

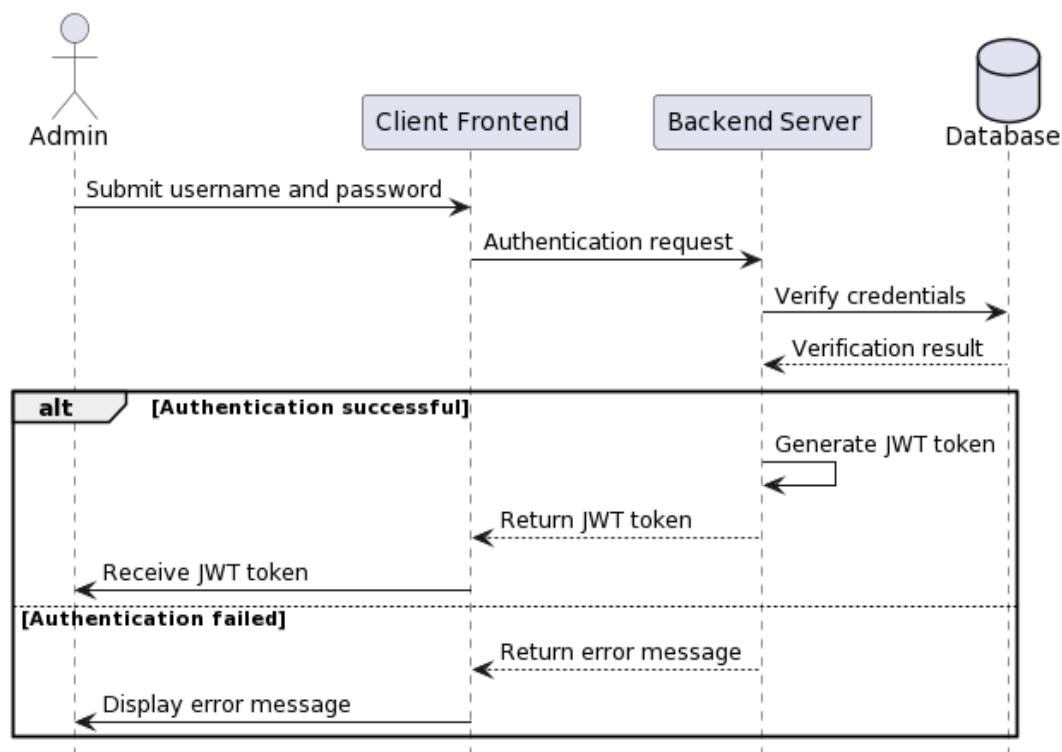


FIGURE 3.3 – Diagramme de séquence “Authentication”.

3.2.3.2 Configuration automatisée des serveurs

Lorsque l’administrateur souhaite effectuer des configurations automatisées des serveurs, il se connecte à l’interface utilisateur (Client Frontend) en fournissant ses informations d’identification. L’interface utilisateur reçoit ces informations et les transmet au serveur backend pour vérification.

Le serveur backend joue un rôle clé dans ce processus. Il communique avec la base de données pour vérifier les informations d’identification de l’administrateur. Cette étape garantit que seuls les administrateurs authentifiés ont accès aux fonctionnalités de configuration des serveurs.

Une fois que l’administrateur est authentifié avec succès, le processus de configuration automatisée des serveurs peut commencer. L’administrateur utilise l’interface utilisateur pour sélectionner un serveur à configurer. Le serveur à configurer peut être un serveur externe qui n’appartient pas à notre plateforme.

L’interface utilisateur communique alors avec la base de données pour récupérer les détails et les paramètres de configuration du serveur sélectionné. Ces informations sont essentielles pour personnaliser la configuration et garantir que les rôles Ansible appropriés sont appliqués.

Une fois les détails et les paramètres récupérés, l’interface utilisateur accède au dossier des playbooks qui contient les rôles Ansible nécessaires pour la configuration. Ces rôles Ansible sont ensuite transmis au serveur à configurer.

Le serveur à configurer, qui peut être un serveur distant, reçoit les rôles Ansible et procède à la mise à jour de ses configurations en fonction des instructions

fournies. Une fois la configuration terminée, le serveur envoie une confirmation à l'interface utilisateur pour indiquer que la mise à jour des configurations a été effectuée avec succès.

Il convient de noter que tout au long de ce processus, l'authentification de l'administrateur est essentielle pour garantir la sécurité et la confidentialité des opérations de configuration des serveurs. Seuls les administrateurs authentifiés peuvent accéder à cette fonctionnalité et effectuer des modifications sur les serveurs.

En résumé, le scénario de configurations automatisées des serveurs implique l'authentification de l'administrateur, la récupération des détails et des paramètres de configuration du serveur sélectionné, l'application des rôles Ansible appropriés, la mise à jour des configurations sur le serveur à configurer, et la confirmation de la réussite de la mise à jour. Tout cela est rendu possible grâce à une communication fluide entre le client frontend, le serveur backend, la base de données et le serveur à configurer. La figure 3.4 décrit cette fonctionnalité.

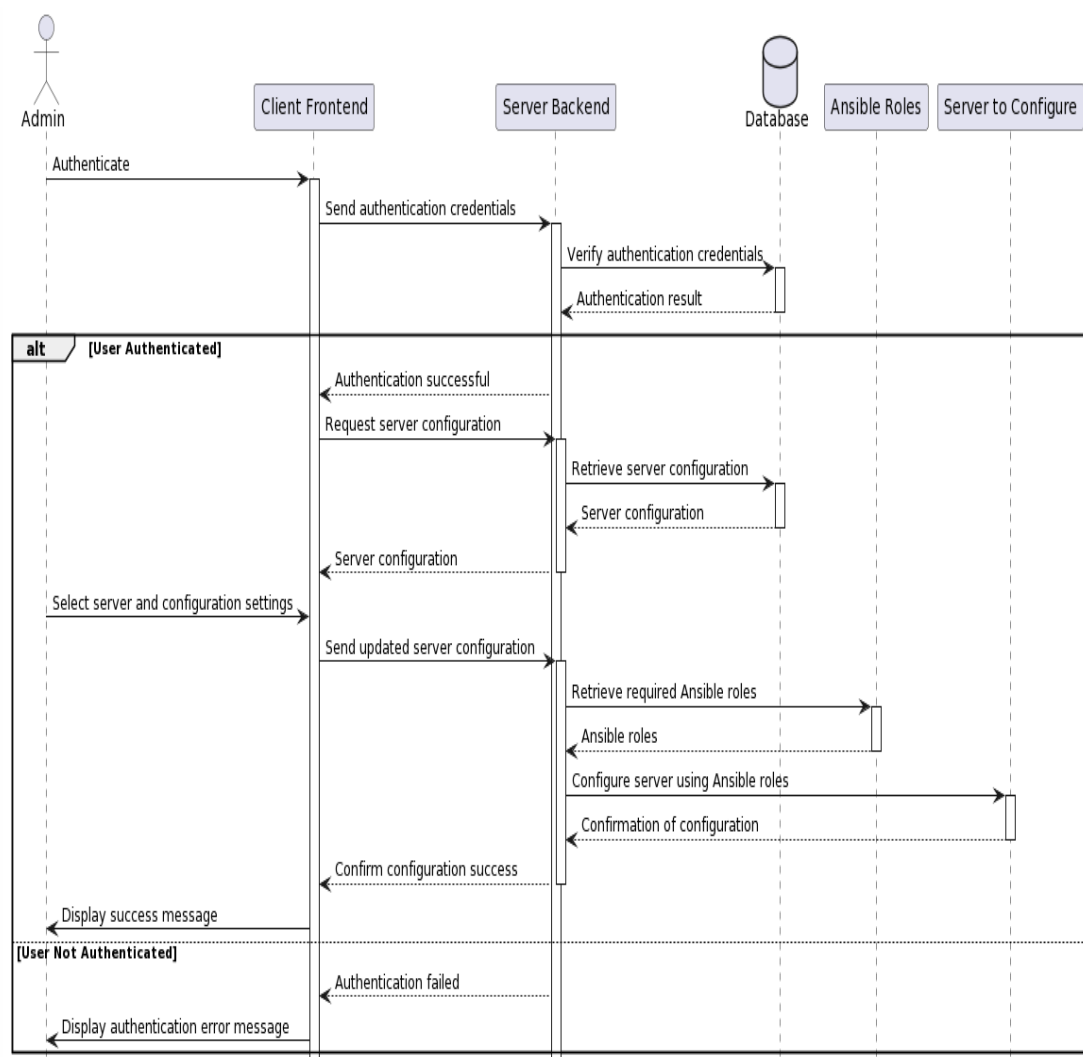


FIGURE 3.4 – Diagramme de séquence “Configuration automatisée des serveurs”.

3.2.3.3 Consultation des rôles ansible

Dans ce scénario, le diagramme représente le processus de l'administrateur accédant à la page de consultation des rôles et visualisant les rôles Ansible disponibles. Il est important de noter que l'administrateur doit d'abord s'authentifier avant de pouvoir accéder à cette fonctionnalité.

La séquence commence par l'administrateur initiant le processus en accédant à la page de consultation des rôles via le Client Frontend. Le Client Frontend envoie ensuite une demande au Backend Server, indiquant le besoin de la liste des rôles.

À la réception de la demande, le Backend Server vérifie d'abord l'authentification de l'administrateur. Si l'administrateur n'est pas authentifié, le processus s'arrête et un message d'erreur est renvoyé au Client Frontend.

Si l'administrateur est authentifié, le Backend Server communique avec la base de données pour récupérer les informations sur les rôles. La base de données répond en fournissant la liste des rôles au Backend Server.

Ensuite, le Backend Server récupère les rôles Ansible disponibles en communiquant avec la source désignée, qui est un dossier contenant les rôles. Le participant "Roles" représente ce processus de récupération.

Une fois que le Backend Server a obtenu les rôles, il renvoie la liste des rôles au Client Frontend. Le Client Frontend affiche ensuite la liste des rôles à l'administrateur, lui permettant de visualiser les rôles Ansible disponibles.

Si l'administrateur souhaite effectuer une recherche par nom de rôle, il entre le nom dans le Client Frontend. Le Client Frontend envoie ensuite une demande de recherche au Backend Server, spécifiant le nom du rôle recherché.

Le Backend Server communique avec la base de données pour effectuer la recherche par nom de rôle. La base de données répond en renvoyant les rôles correspondant au nom spécifié.

Enfin, le Backend Server renvoie les rôles filtrés au Client Frontend, qui les affiche à l'administrateur comme le montre la figure 3.5.

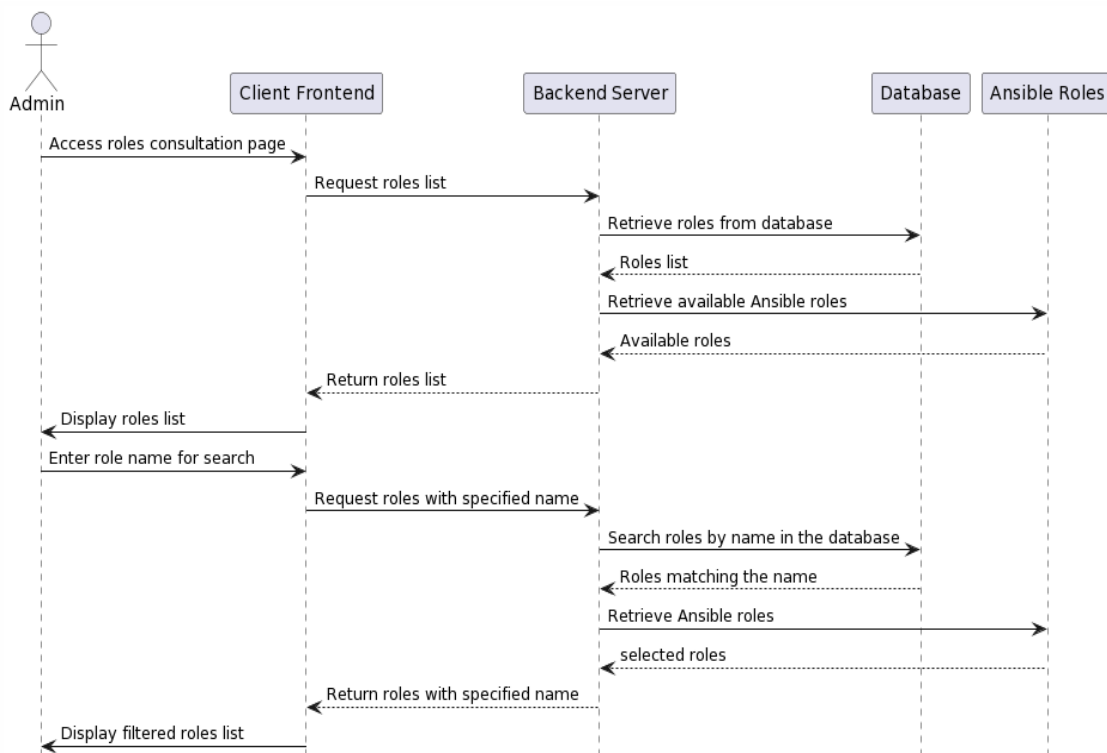


FIGURE 3.5 – Diagramme de séquence “Consultation des rôles ansible”.

3.2.3.4 Gestion des clients

La gestion des clients permet aux administrateurs d’ajouter, de modifier, de supprimer et de consulter les informations des clients enregistrés sur notre plateforme. (Figure 3.6)

Le scénario commence par L’administrateur qui se connecte à notre plateforme via le client frontend. Une fois authentifié, l’administrateur accède à la page de gestion des clients.

Lorsque l’administrateur souhaite ajouter un client, il envoie une demande au serveur backend, qui à son tour communique avec la base de données pour ajouter les informations du client. Une fois l’ajout effectué avec succès, le serveur renvoie une confirmation d’ajout au client frontend, qui à son tour l’affiche à l’administrateur.

De même, lors de la modification ou de la suppression d’un client, l’administrateur envoie une demande au serveur, qui interagit avec la base de données pour mettre à jour ou supprimer les informations du client. Le serveur renvoie ensuite une confirmation à l’administrateur via le client frontend.

Enfin, lorsque l’administrateur souhaite consulter les détails d’un client spécifique, il envoie une demande au serveur, qui récupère les informations du client à partir de la base de données. Le serveur renvoie ensuite les détails du client au client frontend, qui les affiche à l’administrateur.

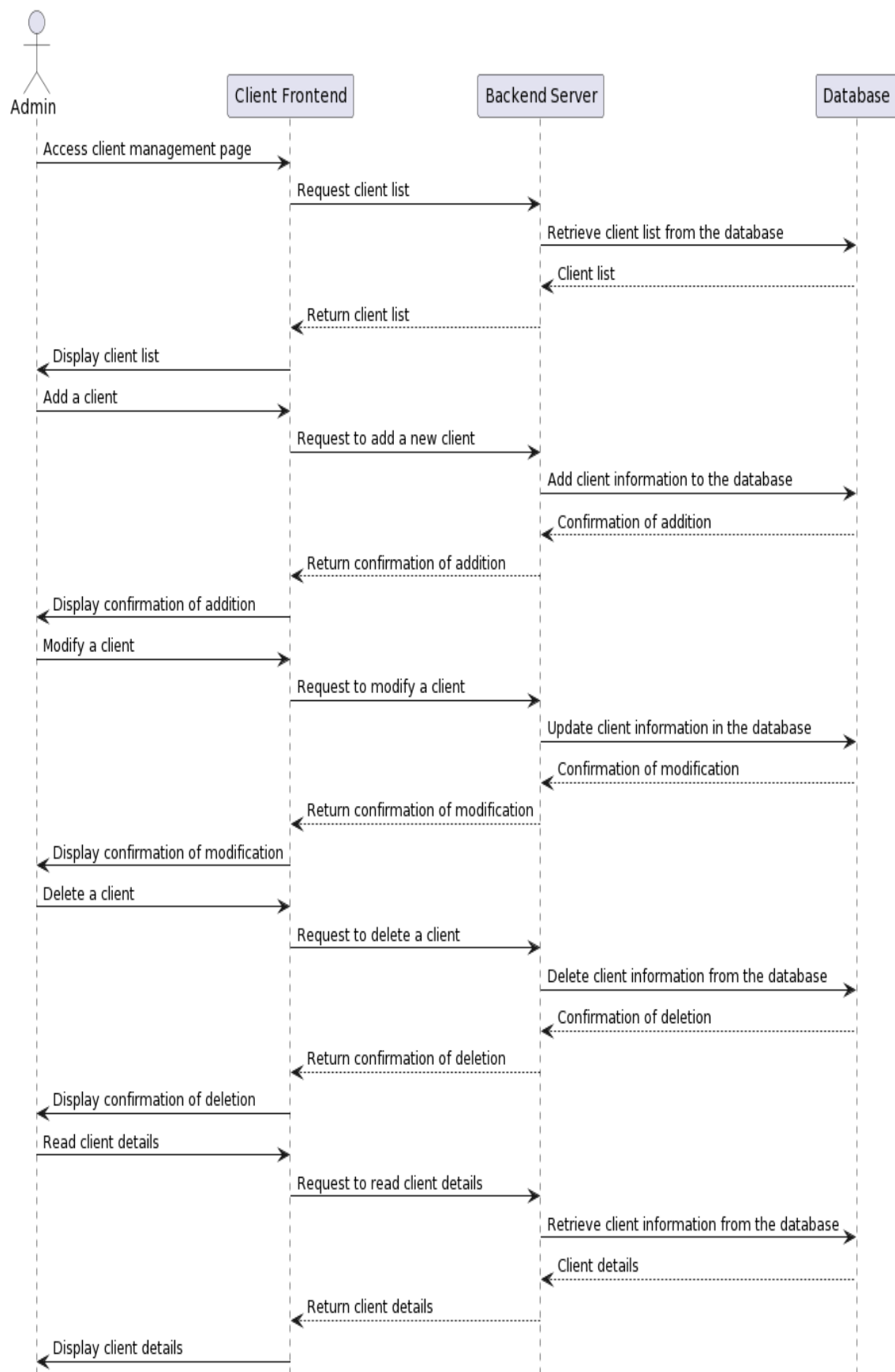


FIGURE 3.6 – Diagramme de séquence “Gestion des Clients”.

3.2.4 Spécifications des besoins non fonctionnels

Les besoins non fonctionnels diffèrent des besoins fonctionnels en ce qu'ils spécifient les contraintes à respecter pour assurer le bon fonctionnement du système conformément aux exigences requises.

- **Fiabilité** : La plateforme doit être disponible en tout temps et ne doit pas planter ou causer de pertes de données. Les sauvegardes doivent être effectuées régulièrement pour assurer la récupération des données en cas de sinistre.
- **Sécurité** : La plateforme doit être protégée contre les attaques externes et les intrusions, ainsi que les utilisations malveillantes internes. Les données sensibles doivent être cryptées et protégées par des niveaux d'accès appropriés.
- **Traçabilité** : La plateforme doit être capable de fournir des rapports détaillés sur l'utilisation, les erreurs et les incidents, afin de permettre l'amélioration continue de la qualité et de la sécurité du système.
- **Facilité d'utilisation** : La solution doit être facile à utiliser, intuitive et ergonomique pour les utilisateurs, avec une documentation claire et accessible.
- **Scalabilité** : La plateforme doit être capable de s'adapter à l'évolution des besoins et de la taille de l'entreprise, sans avoir besoin d'une refonte complète.

Conclusion

Dans ce chapitre, nous identifions les cas d'utilisation et les diagrammes de séquence les plus importants, accompagnés de la spécification des besoins fonctionnels et non fonctionnels, pour faciliter la compréhension du fonctionnement de notre solution.

Introduction

Après avoir étudié les spécifications et les exigences, une étude conceptuelle est requise. Le présent chapitre décrit les aspects architecturaux du projet via des diagrammes pour mettre en évidence le fonctionnement de notre solution. Il termine par le "product Backlog" et la planification des sprints à venir.

4.1 Architecture physique

L'architecture physique de notre solution repose sur plusieurs composants clés qui travaillent ensemble comme présenté dans le diagramme illustré par la figure 4.1 pour faciliter la gestion et l'exécution automatisée des configurations logicielles.

Au niveau de l'interface utilisateur, nous avons une interface Angular qui a été développée pour offrir une expérience conviviale aux utilisateurs. Cette interface communique avec le serveur backend via des requêtes HTTP. Le serveur backend est basé sur Spring Boot et fournit des services RESTful pour gérer les requêtes provenant de l'interface utilisateur. Il est responsable de la gestion des opérations métier, de l'interaction avec la base de données MySQL et de l'exécution de scripts Bash pour appeler les rôles Ansible.

Les rôles Ansible jouent un rôle clé dans notre solution. Ansible est un outil d'automatisation open source qui nous permet de provisionner l'infrastructure, gérer la configuration et déployer des applications. Les rôles Ansible sont définis en utilisant la syntaxe YAML et encapsulent des ensembles réutilisables de tâches et de configurations. Le serveur backend utilise Ansible en exécutant des commandes ou des playbooks Ansible à l'aide de scripts Bash. La communication entre Ansible et les serveurs à configurer se fait via SSH, assurant ainsi une connexion sécurisée.

La base de données MySQL joue un rôle important dans notre solution. Elle stocke les données relatives aux clients (serveurs), aux journaux et aux services. Le serveur backend interagit avec la base de données en utilisant le protocole JDBC pour récupérer ou enregistrer des informations. Cette interaction permet de gérer les données liées aux clients et de garantir la cohérence et la persistance des informations.

En ce qui concerne la configuration des serveurs, nous avons un noeud "Servers to Configure" qui représente les serveurs sur lesquels les configurations sont appliquées. Les différents composants de ce noeud représentent les distributions OS spécifiques à chaque serveur, notamment Debian 10, Debian 11, Red Hat 8 et Red Hat 9.

Dans l'ensemble, cette architecture permet une gestion efficace des configurations logicielles en automatisant les processus et en facilitant la communication entre les différentes composantes de notre solution.

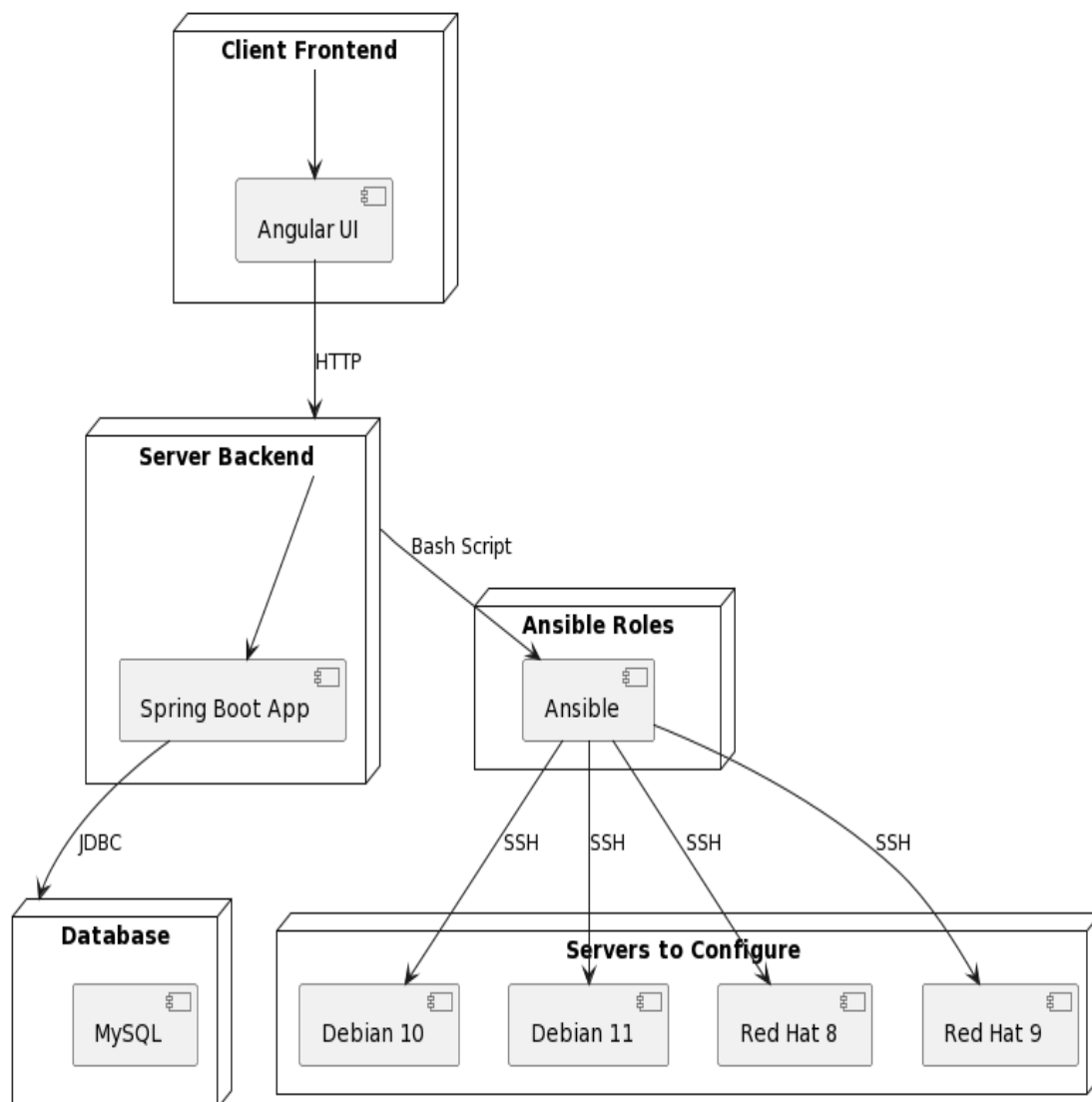


FIGURE 4.1 – Diagramme de déploiement.

4.2 Architecture logique

L'architecture logique de notre plateforme repose sur une approche en couches, où chaque couche joue un rôle spécifique dans le fonctionnement global du système comme le montre la figure 4.2. Nous avons une architecture basée sur le modèle MVC (Modèle-Vue-Contrôleur), qui favorise la séparation des préoccupations et la modularité de notre application.

La couche de présentation est représentée par l'interface utilisateur Angular, qui offre une expérience conviviale aux utilisateurs et leur permet d'interagir avec l'application. Cette interface communique avec la couche métier via des contrôleurs Spring Boot.

La couche métier contient le contrôleur Spring Boot, qui agit comme une passerelle entre l'interface utilisateur et les autres composants du système. Il reçoit les demandes de l'interface utilisateur, coordonne les actions nécessaires et communique avec les services métier. Les services métier, quant à eux, implémentent la logique métier de notre application et effectuent des opérations telles que la gestion des configurations des serveurs, l'exécution automatisée des configurations, la gestion des environnements cibles, etc. Les modèles représentent les données manipulées par l'application et sont utilisés pour stocker et récupérer des informations de la base de données.

Une nouveauté dans notre architecture est l'introduction d'un dossier spécifique pour stocker les rôles Ansible. Ces rôles sont des ensembles réutilisables de tâches et de configurations qui facilitent le provisionnement d'infrastructure, la gestion de configuration et le déploiement d'applications. Ils sont référencés à partir du dossier des rôles Ansible dans la couche métier de notre application. Cela permet une gestion plus flexible des rôles et facilite leur réutilisation.

Enfin, la couche d'accès aux données est représentée par une base de données MySQL, qui stocke les informations relatives aux clients, aux configurations, aux environnements cibles, etc. Les services métier interagissent avec la base de données pour récupérer ou enregistrer les données nécessaires.

En résumé, notre architecture logique suit une approche en couches avec une séparation claire des préoccupations. Nous utilisons le modèle MVC pour structurer notre application et avons introduit un dossier distinct pour stocker les rôles Ansible. Cela nous permet de développer une plateforme robuste, modulaire et facilement extensible pour la gestion et l'exécution automatisée des configurations logicielles.

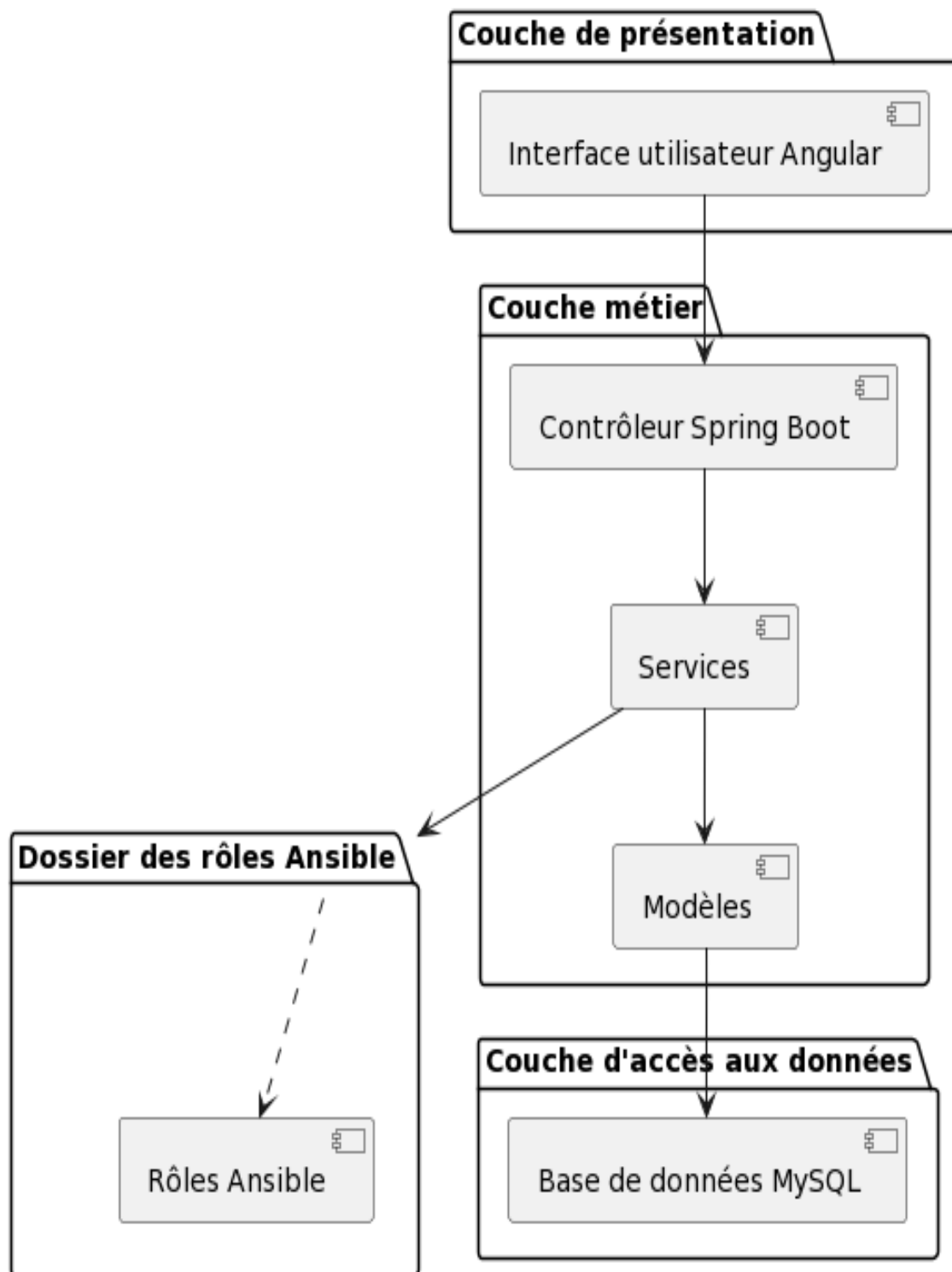


FIGURE 4.2 – Architecture logique.

4.2.1 Architecture MVC

L'architecture Modèle-Vue-Contrôleur (MVC) est un modèle de conception largement utilisé dans le développement de logiciels, en particulier dans les applications basées sur une interface utilisateur graphique. Elle vise à séparer la logique métier, la présentation des données et la gestion des interactions utilisateur pour faciliter la maintenabilité, la réutilisabilité et l'extensibilité du système présentées par la figure 4.3.

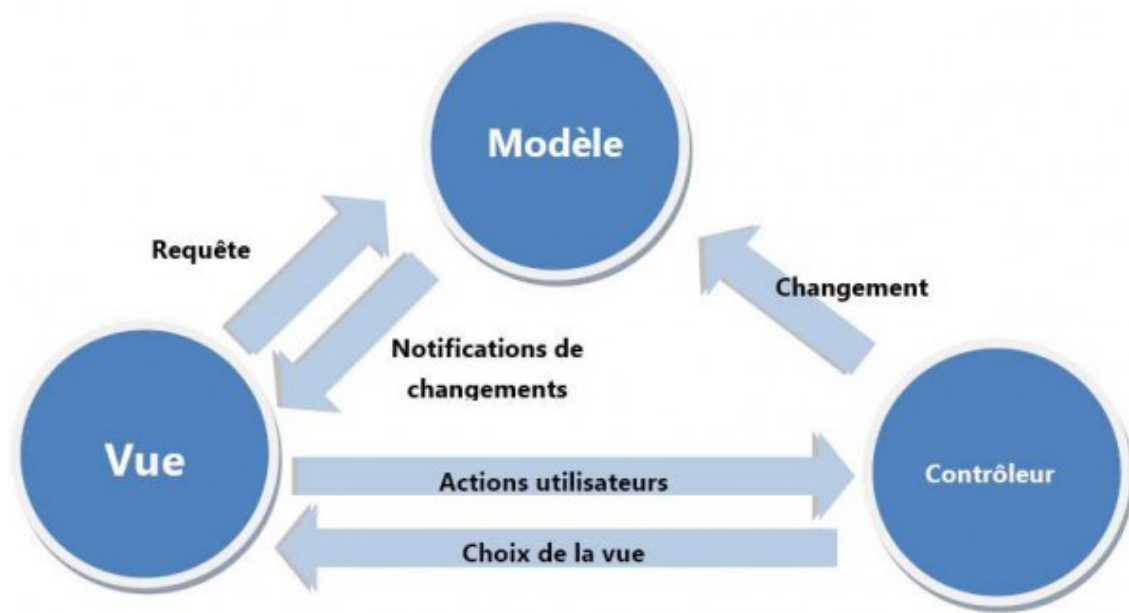


FIGURE 4.3 – Architecture MVC. [10]

- **Modèle** : Le modèle représente la logique métier et les données du système. Il gère la récupération, la manipulation et la persistance des données. Dans notre cas, le modèle peut être représenté par les classes et les services responsables de la gestion des configurations, des clients, des rôles Ansible, etc.
- **Vues** : La vue est responsable de la présentation des données au niveau de l'interface utilisateur. Elle affiche les informations au format approprié pour l'utilisateur et peut permettre des interactions avec le système. Dans notre plateforme, les vues peuvent être représentées par les pages web ou les interfaces graphiques utilisées pour créer, modifier ou afficher les configurations, les clients, les rôles Ansible, etc.
- **Contrôleur** : Le contrôleur agit en tant qu'intermédiaire entre le modèle et la vue. Il reçoit les actions et les événements de l'utilisateur à travers la vue, traite ces actions en conséquence et met à jour le modèle si nécessaire. Dans notre plateforme, les contrôleurs peuvent être représentés par les classes qui gèrent les requêtes et les interactions utilisateur, en appelant les services appropriés du modèle et en mettant à jour les vues en conséquence.

4.3 Modèle de données

Le diagramme de classes fournit une représentation visuelle des différentes entités présentes dans le schéma de base de données ConfigAutomationDB (Figure 4.4). Il illustre les relations et les attributs associés à chaque classe.

La classe "BDD" représente une base de données avec des attributs tels que "bddID" (identifiant de la base de données), "name" (nom de la base de données), "type" (type de base de données) et "version" (version de la base de données).

La classe "Client" représente un client ou un serveur dans le système. Elle est caractérisée par des attributs tels que "clientID" (identifiant du client), "name" (nom du client), "ipaddress" (adresse IP du client) et "users" (utilisateurs associés au client). Un client peut être lié à plusieurs bases de données, prendre en charge plusieurs distributions, utiliser plusieurs langages et utiliser plusieurs services.

La classe "Distribution" représente une distribution de système d'exploitation telle que Debian, Red Hat, etc. Elle possède des attributs tels que "distributionID" (identifiant de la distribution), "name" (nom de la distribution) et "version" (version de la distribution).

Les associations entre les classes sont les suivantes :

- Un client peut prendre en charge plusieurs distributions, également sous forme de relation de composition (1 à plusieurs).
- Un client peut être associé à plusieurs bases de données, ce qui est représenté par une relation de composition (1 à plusieurs).
- Un client peut utiliser plusieurs langages, exprimé par une relation de composition (1 à plusieurs).
- Un client peut utiliser plusieurs services, encore une fois à travers une relation de composition (1 à plusieurs).

Ces associations reflètent les relations entre les différentes entités du schéma ConfigAutomationDB, permettant ainsi de modéliser les connexions entre les clients, les bases de données, les distributions, les langages et les services.

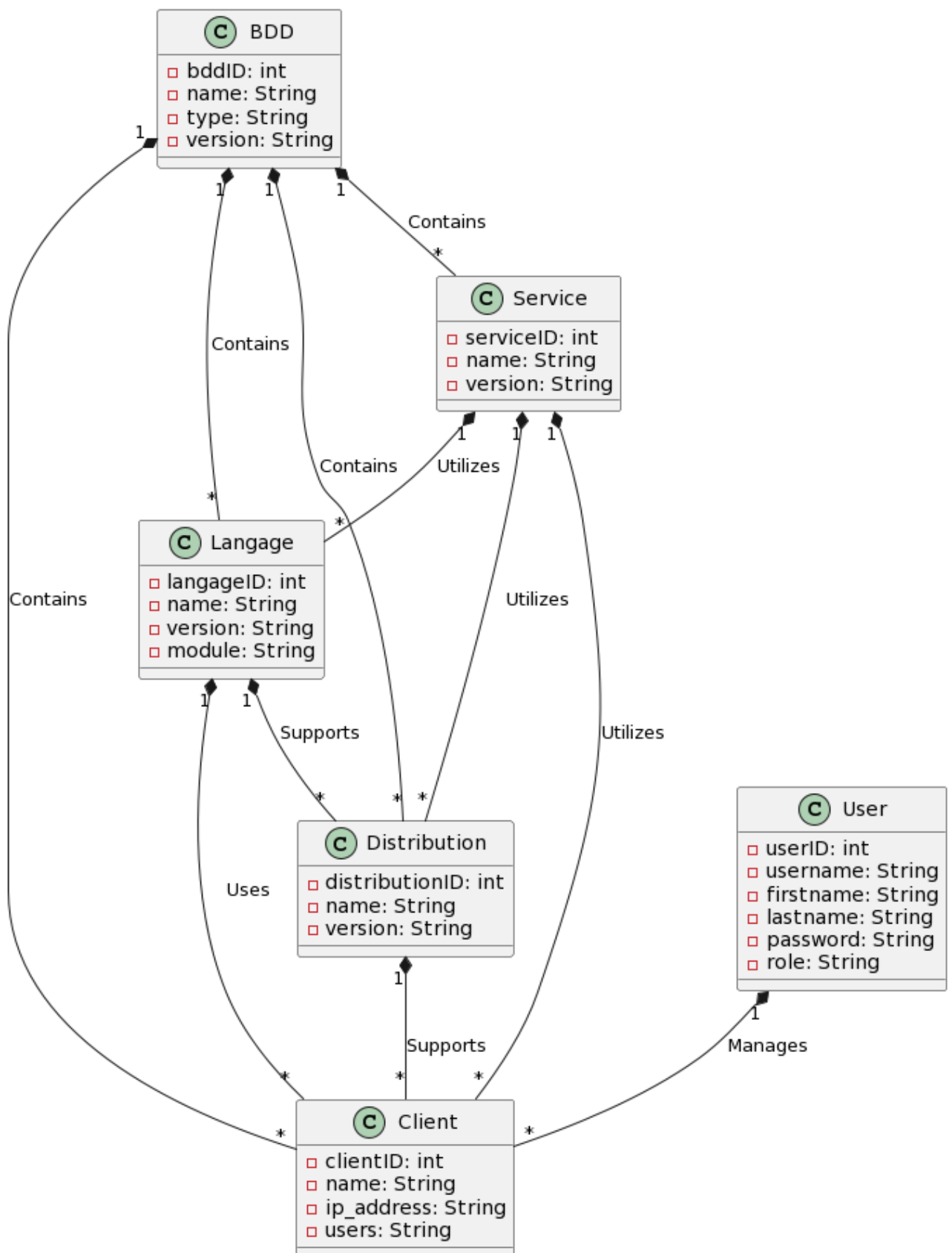


FIGURE 4.4 – Diagramme de classe.

4.4 Backlog du projet

Voici le tableau 4.1 récapitulant les fonctionnalités nécessaires, les histoires d'utilisateurs correspondantes et la difficulté de chaque fonctionnalité. Ce tableau assure la conformité du produit avec les exigences demandées.

TABLEAU 4.1 – Backlog du projet

ID	Fonctionnalité	User Story	Difficulté
1	Développement des rôles Ansible	En tant qu'administrateur, je veux avoir des rôles Ansible prêts à l'emploi pour automatiser les configurations des systèmes cibles.	Moyenne
2	Développement Backend	En tant qu'administrateur, je veux pouvoir créer, modifier et supprimer des configurations logicielles via des API backend.	Élevée
3	Développement Frontend	En tant qu'administrateur, je veux pouvoir gérer les configurations des serveurs à travers une interface utilisateur conviviale.	Moyenne
4	Exécution automatisée des configurations	En tant qu'administrateur, je veux pouvoir déclencher l'exécution automatisée des configurations en fonction d'événements spécifiques ou de planifications régulières.	Élevée
5	Tests, correction de bugs et finalisation	En tant qu'administrateur, je veux que la plateforme soit testée de manière approfondie, que les bugs soient corrigés et que la plateforme soit finalisée pour une utilisation en production.	Moyenne

4.5 Planification des sprints

La planification des sprints est une étape cruciale dans la méthodologie Scrum, car elle permet de découper le projet en itérations concises et ciblées. Chaque sprint est un cycle de développement délimité dans le temps, généralement d'une durée de deux à quatre semaines.

Pour notre projet, nous avons choisi de diviser le travail en quatre sprints principaux :

- **Sprint 1 : Développement des rôles Ansible**

Ce sprint sera dédié au développement des rôles Ansible nécessaires pour la configuration des systèmes cibles. Nous créerons les différents rôles en utilisant les bonnes pratiques d'Ansible. Une fois les rôles développés, nous les déploierons sur les serveurs correspondants, assurant ainsi une automatisation efficace des configurations.

- **Sprint 2 : Développement Backend**

Dans ce sprint, nous nous concentrerons sur le développement du backend de la plateforme. Notre objectif principal sera de mettre en place l'architecture backend en utilisant les meilleures pratiques. Nous configurerons également les serveurs nécessaires pour héberger notre application. Ensuite, nous créerons les API de base pour la gestion des configurations des serveurs, permettant ainsi à l'administrateur d'effectuer les opérations de création, de modification et de suppression des configurations.

- **Sprint 3 : Développement Frontend**

L'objectif principal de ce sprint est de développer la partie frontend de la plateforme d'automatisation des configurations. Nous allons créer la structure de base de l'application frontend et configurer l'environnement de développement. Ensuite, nous allons implémenter les composants de l'interface utilisateur nécessaires pour la gestion des configurations des serveurs. Cela comprendra la création, la modification et la suppression des configurations logicielles, ainsi que la visualisation des configurations existantes.

- **Sprint 4 : Fonctionnalités complémentaires et finalisation**

Dans ce dernier sprint, nous nous concentrerons sur le développement des fonctionnalités complémentaires de la plateforme. Cela comprendra notamment l'exécution automatisée des configurations, permettant à l'administrateur de déclencher les configurations en fonction d'événements spécifiques ou de planifications régulières. Nous consacrerons également du temps aux tests approfondis, à la correction des bugs et à la finalisation de la plateforme d'automatisation des configurations.

Conclusion

Le présent chapitre était dédié à l'analyse conceptuelle du projet. Dans un premier temps, il explique l'architecture physique et logique du projet. Il décrit ensuite le diagramme de classe de notre plateforme.

Introduction

Après avoir traité la partie conceptuelle, nous avons commencé la partie réalisation du projet, nous avons montré les différentes techniques utilisées pour installer notre système, puis nous décrivons l'implémentation et la réalisation de la solution.

5.1 Choix technique

5.1.1 Outil de gestion de configuration

Les outils de gestion de configuration sont utilisés pour contrôler les changements, les mises à jour et les modifications apportées à l'infrastructure d'une entreprise. Ces outils gèrent l'équilibre requis dans les grands environnements de serveurs, effectuent une surveillance continue et garantissent que l'infrastructure est configurée selon les spécifications correctes, éliminant les risques et réduisant le temps de résolution des incidents. Ces fonctionnalités facilitent la collaboration au sein de l'entreprise en éliminant les problèmes de contrôle de version et en garantissant la cohérence des modifications. Enfin, les outils de gestion de la configuration sont essentiels pour éviter des changements excessifs entre les serveurs et l'infrastructure, et peuvent en fait économiser du temps et de l'argent pour corriger les erreurs.

5.1.1.1 Ansible

Ansible est un moteur d'automatisation open source qui automatise le provisionnement des logiciels, la gestion de la configuration et le déploiement des applications. Ansible fournit une automatisation informatique simple, mettant fin aux tâches répétitives et permettant aux équipes DevOps de se concentrer sur des tâches plus stratégiques. Ansible simplifie le déploiement, la gestion de la configuration et l'orchestration des applications, le tout à partir d'un seul système. Pour atteindre cet objectif, Ansible est conçu pour être un outil simple, fiable et convivial avec un minimum de dépendances. De plus, son architecture sans agent garantit sa sécurité et réduit les coûts du réseau.

5.1.1.2 Chef

Chef est un outil de gestion de la configuration et une plate-forme d'automatisation conçus pour simplifier les tâches de provisionnement, de configuration et de maintenance des serveurs d'entreprise. Il transforme l'infrastructure en code, la rendant flexible, versionnable, lisible et testable, quelle que soit la plate-forme sur laquelle elle s'exécute ou la taille du réseau [10]. Écrit en Ruby et Erlang, il peut s'intégrer à diverses plates-formes cloud, notamment Rackspace, Internap, Amazon EC2, Google Cloud Platform, OpenStack, SoftLayer et Microsoft Azure. Comme son nom l'indique, Chef nécessite l'écriture d'une série de recettes qui décrivent une série de ressources qui doivent être dans un certain état : un package à installer, un service à exécuter ou un fichier à écrire. .

5.1.1.3 Puppet

Puppet est un outil de gestion de configuration logicielle Open Source qui assure un moyen standard de livrer et d'exploiter les logiciels, quel que soit l'endroit où ils s'exécutent. L'utilisateur doit déclarer le statut final des applications déployées et de l'infrastructure provisionnée au moyen d'un langage facile d'accès. Compatible avec Unix et Windows, Puppet est livré avec son propre langage déclaratif pour décrire la configuration du système. Il existe deux éditions de l'outil : Open Source et professionnelle. Cette dernière inclut un GUI, des API et des outils en ligne de commande pour la gestion des nœuds. L'objectif de Puppet est de permettre de partager, de tester et d'appliquer des changements à travers un Datacenter, tout en garantissant visibilité et reporting pour la prise de décision et la conformité. Son véritable objectif est de mettre en œuvre une manière élastique et standard d'automatiser le déploiement et la mise en production des applications.

5.1.1.4 Comparaison et synthèse

Le tableau synthétise la comparaison des gestionnaires de configuration présentés préalablement, selon les critères établis.

TABLEAU 5.1 – Récapitulatif de comparaison des outils de gestion et automatisation de configuration

Caractéristiques	Ansible	Chef	Puppet
Langage	YAML	Ruby	DSL basé sur Ruby
	Facile à lire et à écrire	Apprentissage plus complexe	Syntaxe spécifique
Sans Agent	Oui	Non	Non
Orchestration	Oui	Oui	Oui
Gestion des configurations	Oui	Oui	Oui
Extensibilité	Modules personnalisés	Ressources personnalisées	Modules personnalisés
Communauté	Vaste et active	Active	Active
Plateformes prises en charge	Multiplateforme	Multiplateforme	Multiplateforme

L'avantage d'Ansible par rapport aux autres outils de gestion de configuration est son aspect sans agent, c'est-à-dire qu'il ne nécessite aucune configuration préalable pour contrôler une machine. De plus, sa communauté est plus grande et de nouvelles options sont ajoutées chaque mois. Ainsi, par rapport à Chef et Puppet, Ansible reste le plus adapté et le plus facile à mettre en œuvre.

5.1.2 Outils de gestion de code source

La gestion du code source (SCM) est essentielle pour la gestion des changements logiciels. Son objectif principal est d'améliorer la qualité et la rapidité de livraison des changements de code. Les outils de SCM offrent des fonctionnalités telles que le suivi, la visibilité, la collaboration et le contrôle tout au long du cycle de vie du développement, ce qui permet aux développeurs travaillant sur des projets complexes d'être plus créatifs et libres. De plus, le SCM protège les fichiers source contre les anomalies et permet à toutes les équipes de connaître les auteurs des changements effectués et à quel stade. Dans cette étude, nous nous concentrons sur le choix de l'outil GitLab, déjà utilisé en interne, par rapport à ses concurrents BitBucket et GitHub, afin de consolider notre choix de gestionnaire de code source.

5.1.2.1 GitLab

GitLab est une plateforme web de gestion de référentiel Git qui offre une gamme complète de fonctionnalités, y compris un wiki intégré et des outils de suivi des problèmes. Il fournit une gestion centralisée des référentiels Git, permettant aux utilisateurs d'avoir un contrôle total sur leurs projets et référentiels. Écrit principalement en Ruby, avec des composants supplémentaires en Go, GitLab propose des fonctionnalités avancées telles que des contrôles d'accès granulaires, des revues de code, un suivi des problèmes, des flux d'activités, des wikis et une intégration continue. En décembre 2016, il comptait 1 400 contributeurs Open Source et était largement utilisé par des grandes entreprises telles que Sony, IBM, CERN et la NASA, pour n'en citer que quelques-unes. GitLab est reconnu comme une plateforme solide et fiable pour la gestion du code source et a acquis une réputation dans la communauté des développeurs.

5.1.2.2 GitHub

GitHub est un service de référentiel web hébergé offrant toutes les fonctionnalités de gestion de code source, tout en garantissant un espace aux développeurs pour stocker leurs projets et concevoir des logiciels en parallèle. GitHub fournit des fonctions de collaboration, de contrôle d'accès, des Wikis et des outils simples de gestion de tâches pour projets. Conçu par des développeurs pour les développeurs, GitHub propose une interface graphique et un bureau web ainsi qu'une intégration mobile. Il ne se borne pas au développement logiciel : son aspect ouvert et « réseau social » est fondamental. Il permet de faire une copie du projet public d'une autre personne et de modifier ses fonctionnalités tout en visualisant le travail et les profils de chacun.

5.1.2.3 BitBucket

BitBucket est un service web hébergé qui facilite la gestion de projets logiciels utilisant les systèmes de contrôle de version Mercurial ou Git. Il propose des plans commerciaux ainsi que des options gratuites, offrant aux utilisateurs un nombre illimité de référentiels privés et jusqu'à 5 utilisateurs. Grâce à sa flexibilité, BitBucket simplifie la collaboration entre les équipes de développement. Parmi ses fonctionnalités clés, on retrouve l'extraction de requêtes, les permissions de branches et les discussions en ligne, qui favorisent une gestion efficace du code. BitBucket est écrit en Python, en utilisant le framework web Django. Il permet aux équipes de livrer et de partager du code de meilleure qualité plus rapidement. Sa capacité d'adaptation, particulièrement dans un environnement commercial, garantit aux développeurs une vitesse et une fiabilité optimales, peu importe l'endroit où le projet est situé.

5.1.2.4 Comparaison et synthèse

Le tableau synthétise la comparaison des gestionnaires de code source présentés préalablement, selon les critères établis.

TABLEAU 5.2 – Récapitulatif de comparaison des outils de gestion de code source

Caractéristiques	GitLab	GitHub	Bitbucket
Type de dépôt	Self-hosted	Cloud-based	Cloud-based
	Peut être hébergé localement	Hébergé par le fournisseur	Hébergé par le fournisseur
Gestion de version	Oui	Oui	Oui
Contrôle d'accès	Oui	Oui	Oui
Intégration continue	Oui	Oui	Oui
Suivi des problèmes	Oui	Oui	Oui
Documentation	Oui	Oui	Oui
Wikis	Oui	Oui	Oui
Tableaux de bord	Oui	Oui	Oui
Support des plug-ins	Oui	Oui	Oui
Open source	Oui	Non	Non

L'outil GitLab est open source, de plus gratuit pour la version communauté. Cette raison nous a poussés de le choisir sans hésitation. D'une autre, son intégration de fonctionnalité d'intégration continue et suivi de problèmes permettra dans le futur d'abonder l'utilisation des outils à part.

5.1.3 Plateformes, langages et environnements de développement

5.1.3.1 Angular

Angular est un framework de développement Web open source basé sur TypeScript qui fournit une architecture modulaire et des composants réutilisables. Il simplifie la création d'applications Web interactives et dynamiques grâce à une gestion avancée de la liaison de données bidirectionnelle. Angular est également compatible avec les appareils mobiles et dispose d'une grande communauté de développeurs et d'un riche écosystème de modules complémentaires. Dans l'ensemble, Angular offre une meilleure structure de code, une productivité plus élevée, une compatibilité mobile et une communauté active, ce qui en fait un choix populaire pour développer des applications Web performantes et évolutives.

5.1.3.2 Spring Boot

Spring Boot est un framework de développement d'applications web Java qui simplifie considérablement le processus de création d'applications robustes et évolutives. Il offre une configuration intelligente par défaut et facilite la mise en place de l'infrastructure nécessaire à une application web. Grâce à son approche basée sur des conventions plutôt que des configurations, Spring Boot permet de démarrer rapidement un projet. Cela en fait un choix populaire pour le développement d'applications web en Java.

5.1.3.3 MySQL

MySQL est un système de gestion de base de données relationnelle open source populaire. Il se distingue par sa stabilité, sa fiabilité et sa capacité à traiter efficacement de grandes quantités de données. MySQL offre de bonnes performances pour les opérations de lecture et prend en charge les transactions ACID. Il est flexible et fournit de nombreuses fonctionnalités telles que les index et les clés étrangères. En conclusion, MySQL est un choix solide pour stocker et gérer des données dans le développement d'applications Web.

5.1.3.4 Postman

Postman est une application qui permet de tester les API à l'aide d'une interface utilisateur graphique. Parmi les avantages de Postman, citons la fonction de collecte et la possibilité de créer différents environnements de test. Postman est un outil convivial qui aide à optimiser le temps lors de l'exécution des tests.

5.1.3.5 IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré (IDE) puissant et convivial, principalement utilisé pour le développement de logiciels en Java. Il offre une interface utilisateur intuitive et une large gamme de fonctionnalités avancées qui améliorent la productivité des développeurs. L'IDE propose des fonctionnalités telles que la complétion de code intelligente, la navigation facile entre les fichiers, le refactoring automatique et l'analyse statique du code. En résumé, IntelliJ IDEA facilite le processus de développement en fournissant un ensemble complet d'outils pour les développeurs Java.

5.1.3.6 Lucidchart

Lucidchart est une plateforme en ligne facile à utiliser pour créer et partager facilement des graphiques et des visualisations. Avec une large gamme de formes, d'icônes et de modèles prédéfinis, Lucidchart facilite la création de diagrammes professionnels. Il fournit également des fonctionnalités de collaboration en temps

réel qui permettent aux équipes de travailler ensemble efficacement. En conclusion, Lucidchart est un outil pratique pour créer des diagrammes et des visualisations de manière rapide et collaborative.

5.1.3.7 Docker

Docker est une plateforme open-source qui simplifie le déploiement et l'exécution d'applications dans des conteneurs. Les conteneurs Docker fournissent un environnement isolé et léger pour exécuter des applications, ce qui facilite la portabilité et l'évolutivité. Docker permet aux développeurs de créer des images conteneurisées contenant toutes les dépendances nécessaires pour exécuter une application, ce qui facilite le déploiement sur différentes machines et environnements. Grâce à sa simplicité d'utilisation et à sa large adoption, Docker est devenu un outil essentiel dans le développement et le déploiement d'applications.

5.1.3.8 Overleaf

Overleaf est une plateforme en ligne qui facilite l'édition et la collaboration de documents LaTeX. Il offre une interface conviviale pour créer, modifier et compiler des documents LaTeX en temps réel. Overleaf est largement utilisé dans le domaine académique et scientifique pour la création de documents professionnels tels que des articles de recherche, des thèses et des rapports.

5.1.3.9 YAML

YAML (YAML Ain't Markup Language) est un langage de sérialisation de données simple et lisible par les humains. Il est couramment utilisé pour la configuration de logiciels, le stockage de données structurées et l'échange de données entre applications. En tant qu'utilisateur, vous avez utilisé YAML dans les rôles Ansible, un outil d'automatisation informatique. YAML est utilisé pour définir la configuration et les tâches des rôles Ansible.

5.1.3.10 Script Bash

Un script Bash est un fichier contenant une séquence de commandes et d'instructions destinées à être exécutées dans l'interpréteur de commandes Bash. Bash est un shell Unix populaire, largement utilisé dans les systèmes d'exploitation basés sur Linux. Il est utilisé pour appeler les rôles depuis notre backend Spring Boot. Le script Bash nous a permis d'automatiser le processus d'appel et d'exécution des rôles, simplifiant ainsi la gestion et le déploiement de votre application.

5.2 Démonstration

La sections précédente ont défini divers outils qui seront présentés dans les captures d'écran suivantes. Ces images fournissent une démonstration complète des fonctionnalités offertes par chaque outil et couvrent tous les aspects du projet.

5.2.1 Connexion aux serveurs

Dans notre projet, quatre serveurs distants sont utilisés pour le test. Il existe plusieurs méthodes pour se connecter, dans notre cas, la connexion aux quatre serveurs se fait en utilisant le protocole SSH [11].

Chaque serveur, qu'il s'agisse de Debian 10, Debian 11, Red Hat 8 ou Red Hat 9, dispose d'une adresse IP unique à laquelle nous établissons une connexion SSH.

- Génération d'une paire de clés SSH (Figure 5.1) : Tout d'abord, nous devons générer une paire de clés SSH composée d'une clé privée et d'une clé publique. Cela se fait généralement à l'aide de la commande "ssh-keygen".

```
yaakoub@yaakoub:~$ ssh-keygen -t rsa -C "achrafyaakoubpfe@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yaakoub/.ssh/id_rsa): VMs
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in VMs
Your public key has been saved in VMs.pub
The key fingerprint is:
SHA256:NeL800Mhd+6Re4kCsQDm7oiVltPbBK01BtYQz/Ur++U achrafyaakoubpfe@gmail.com
The key's randomart image is:
+---[RSA 3072]---+
|  o.  .          |
|  = . .          |
|  o + . = o .    |
|  . o o o = + .  |
|  ..S.. . +     |
|  + = ..o+.o ..o |
|  *oo ... o+.o.. |
|  .o+. + . +.. . |
|  ..o+. . . E    |
+---[SHA256]-----+
```

FIGURE 5.1 – Génération d'une paire de clés SSH.

- Copie de la clé publique sur les serveurs (Figure 5.2) : La clé publique doit être copiée sur chaque serveur avec lequel nous souhaitons nous connecter.

```
yaakoub@yaakoub:~$ ssh-copy-id -i /home/yaakoub/.ssh/gcp-vm achrafyaakoubpfe@34.173.6.106
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/yaakoub/.ssh/gcp-vm.pub"
The authenticity of host '34.173.6.106 (34.173.6.106)' can't be established.
ED25519 key fingerprint is SHA256:58CaIgJiZxtIT0JfMEP5j9QEs2p9sa4PrawWtGTMVuk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
```

FIGURE 5.2 – Exemple : Copie de la clé publique sur Debian.

- Vérification de la connexion SSH : Une fois que la clé publique est copiée sur les serveurs, nous pouvons tester la connexion SSH.

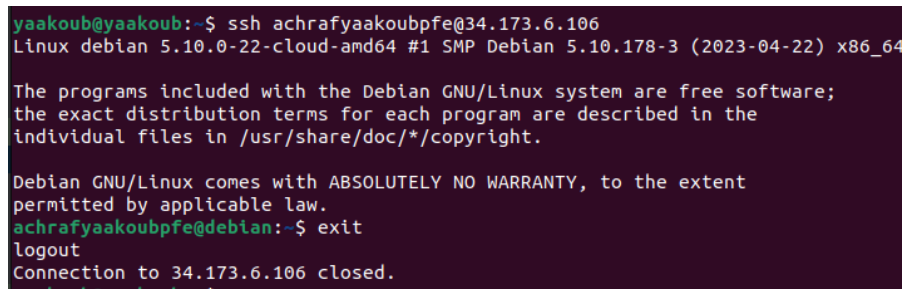
A screenshot of a terminal window showing an SSH connection. The prompt is 'yaakoub@yaakoub:~\$' and the command is 'ssh achrafyaakoubpfe@34.173.6.106'. The terminal output shows the Debian login banner, including the version 'Linux debian 5.10.0-22-cloud-amd64 #1 SMP Debian 5.10.178-3 (2023-04-22) x86_64', a notice about free software, and a disclaimer. The user then types 'exit' and the session ends with 'logout' and 'Connection to 34.173.6.106 closed.'

FIGURE 5.3 – Exemple : la connexion SSH avec Debian.

5.2.2 Développement des rôles Ansible

Le développement des rôles Ansible a été une partie essentielle de notre projet, nous permettant de configurer et de gérer nos serveurs de manière automatisée et reproductible. En utilisant la structure Ansible Galaxy et en suivant les bonnes pratiques de développement, nous avons pu créer des rôles personnalisés pour chaque composant de notre infrastructure. Ces rôles ont permis d’automatiser des tâches complexes et de garantir la cohérence et la fiabilité de notre système. Grâce à notre approche modulaire et à la collaboration avec notre équipe, nous avons pu développer une solution d’automatisation solide et flexible, contribuant ainsi à l’efficacité et à la stabilité de notre plateforme.

5.2.2.1 Les rôles développés

Nous avons développé plusieurs rôles Ansible dans le cadre de notre plateforme de configuration automatisé. Ces rôles sont conçus pour simplifier l’installation et la configuration de différents composants de notre infrastructure. Voici ces rôles :

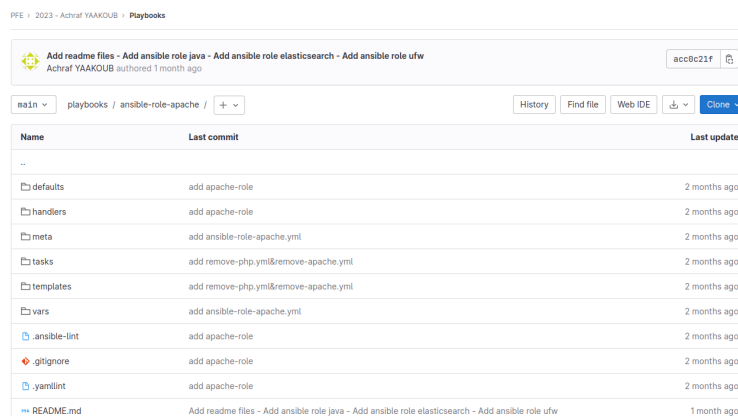
- Varnish : ce rôle facilite l’installation et la configuration du cache HTTP Varnish.
- Apache : ce rôle permet d’installer et de configurer le serveur web Apache.
- Nginx : ce rôle facilite l’installation et la configuration du serveur web Nginx.
- PostgreSQL : ce rôle est dédié à l’installation et à la configuration du serveur de base de données PostgreSQL.
- MySQL : ce rôle permet d’installer et de configurer le serveur de base de données MySQL.
- MariaDB : ce rôle est conçu pour l’installation et la configuration du serveur de base de données MariaDB.

- Elasticsearch : ce rôle simplifie le déploiement et la configuration du moteur de recherche Elasticsearch.
- MongoDB : ce rôle facilite l'installation et la configuration de la base de données MongoDB.
- PHP : ce rôle est dédié à l'installation et à la configuration du langage de programmation PHP.
- Java : ce rôle permet d'installer et de configurer l'environnement Java.
- Python : ce rôle facilite l'installation et la configuration du langage de programmation Python.
- UFW : ce rôle permet de gérer le pare-feu UFW (Uncomplicated Firewall).
- Ganglia : ce rôle simplifie le déploiement et la configuration du système de monitoring Ganglia.

Ces rôles ont été développés de manière modulaire et configurable, ce qui permet de les réutiliser facilement pour différents scénarios d'installation et de configuration. Ils contribuent à l'automatisation et à la gestion efficace de notre infrastructure. On va mentionner un exemple des rôles (Apache) dans la section suivante.

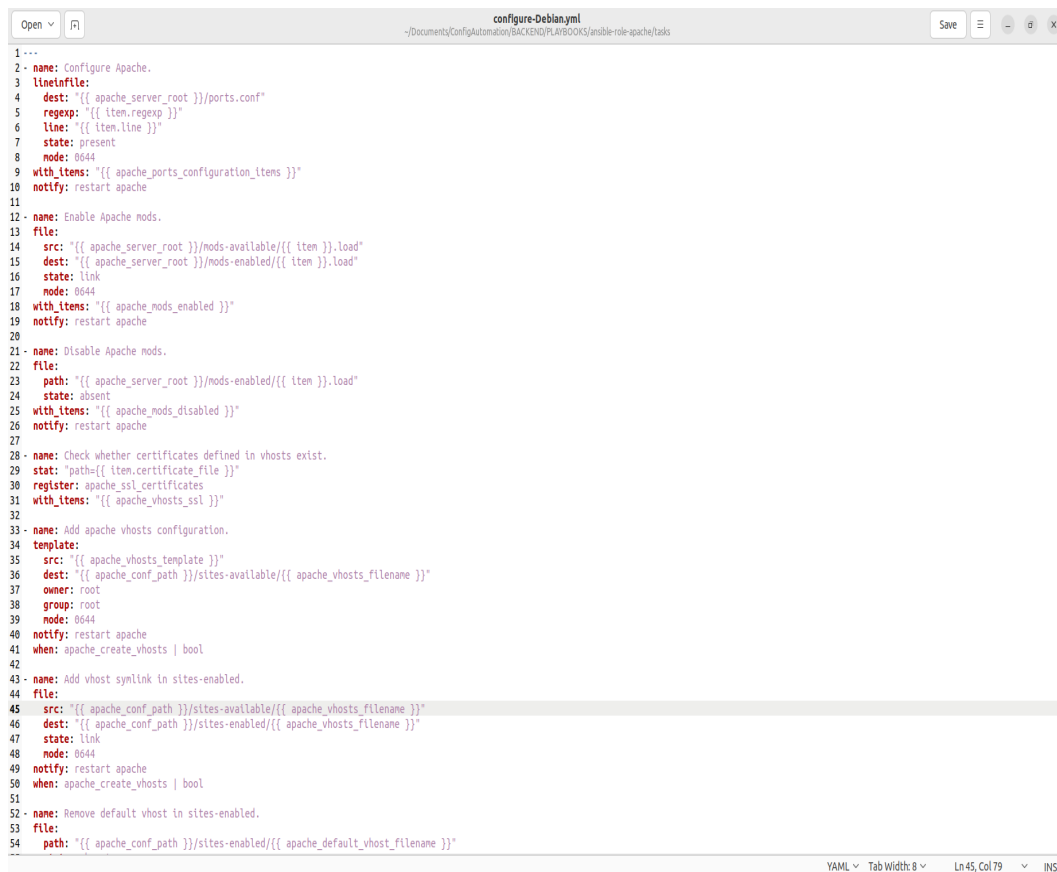
5.2.2.2 Apache Rôle

L'un des rôles clés que nous avons développés est le rôle Apache (Figure 5.4) , qui nous a permis de configurer et de gérer le serveur web Apache sur nos machines. Grâce à ce rôle, nous avons pu déployer rapidement et facilement des sites web, gérer les virtual hosts, et assurer la sécurité et les performances de nos applications web.



Name	Last commit	Last update
..		
defaults	add apache-role	2 months ago
handlers	add apache-role	2 months ago
meta	add ansible-role-apache.yml	2 months ago
tasks	add remove-php.yml&remove-apache.yml	2 months ago
templates	add remove-php.yml&remove-apache.yml	2 months ago
vars	add ansible-role-apache.yml	2 months ago
.ansible-lint	add apache-role	2 months ago
.gitignore	add apache-role	2 months ago
.yamllint	add apache-role	2 months ago
README.md	Add readme files - Add ansible role java - Add ansible role elasticsearch - Add ansible role ufw	1 month ago

FIGURE 5.4 – Structure du rôle Apache.

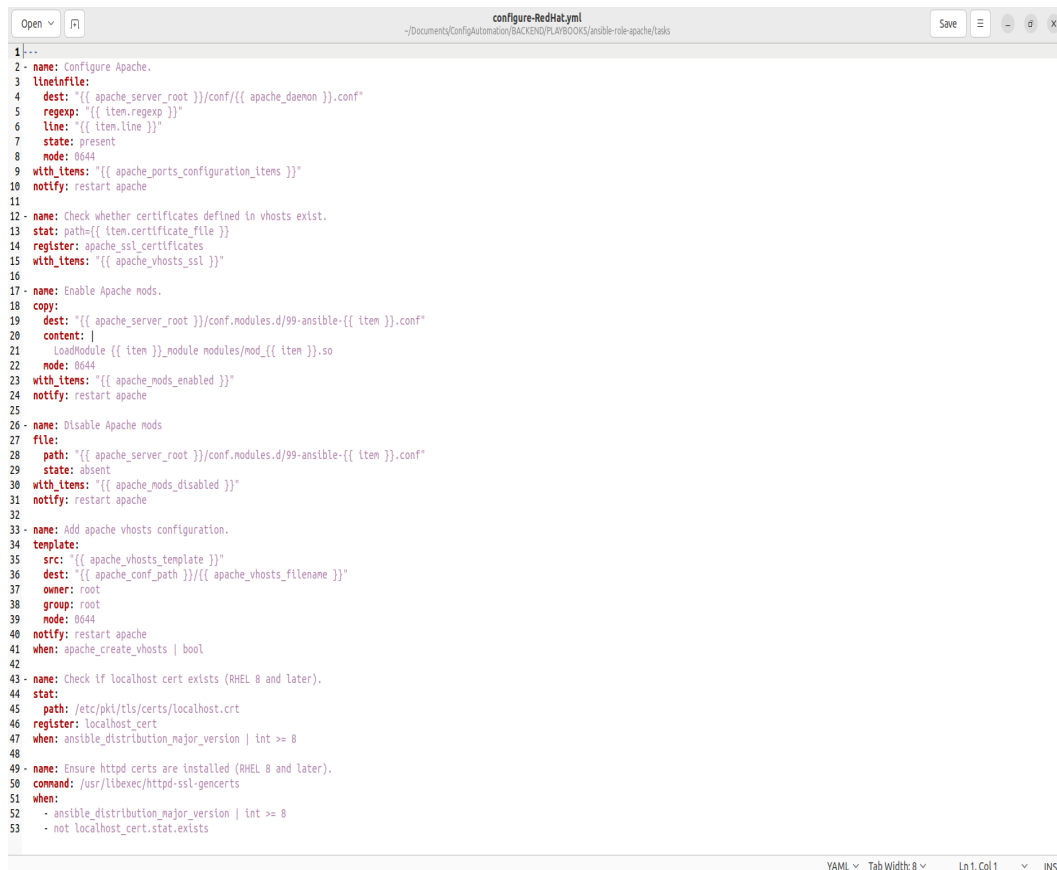


```

1 ---
2 - name: Configure Apache.
3   lineinfile:
4     dest: "{{ apache_server_root }}/ports.conf"
5     regexp: "{{ item.regexp }}"
6     line: "{{ item.line }}"
7     state: present
8     node: 0644
9   with_items: "{{ apache_ports_configuration_items }}"
10  notify: restart apache
11
12 - name: Enable Apache mods.
13   file:
14     src: "{{ apache_server_root }}/mods-available/{{ item }}.load"
15     dest: "{{ apache_server_root }}/mods-enabled/{{ item }}.load"
16     state: link
17     node: 0644
18   with_items: "{{ apache_mods_enabled }}"
19   notify: restart apache
20
21 - name: Disable Apache mods.
22   file:
23     path: "{{ apache_server_root }}/mods-enabled/{{ item }}.load"
24     state: absent
25   with_items: "{{ apache_mods_disabled }}"
26   notify: restart apache
27
28 - name: Check whether certificates defined in vhosts exist.
29   stat: path={{ item.certificate_file }}"
30   register: apache_ssl_certificates
31   with_items: "{{ apache_vhosts_ssl }}"
32
33 - name: Add apache vhosts configuration.
34   template:
35     src: "{{ apache_vhosts_template }}"
36     dest: "{{ apache_conf_path }}/sites-available/{{ apache_vhosts_filename }}"
37     owner: root
38     group: root
39     node: 0644
40   notify: restart apache
41   when: apache_create_vhosts | bool
42
43 - name: Add vhost symlink in sites-enabled.
44   file:
45     src: "{{ apache_conf_path }}/sites-available/{{ apache_vhosts_filename }}"
46     dest: "{{ apache_conf_path }}/sites-enabled/{{ apache_vhosts_filename }}"
47     state: link
48     node: 0644
49   notify: restart apache
50   when: apache_create_vhosts | bool
51
52 - name: Remove default vhost in sites-enabled.
53   file:
54     path: "{{ apache_conf_path }}/sites-enabled/{{ apache_default_vhost_filename }}"

```

FIGURE 5.5 – Playbook de configuration d’Apache pour Debian.



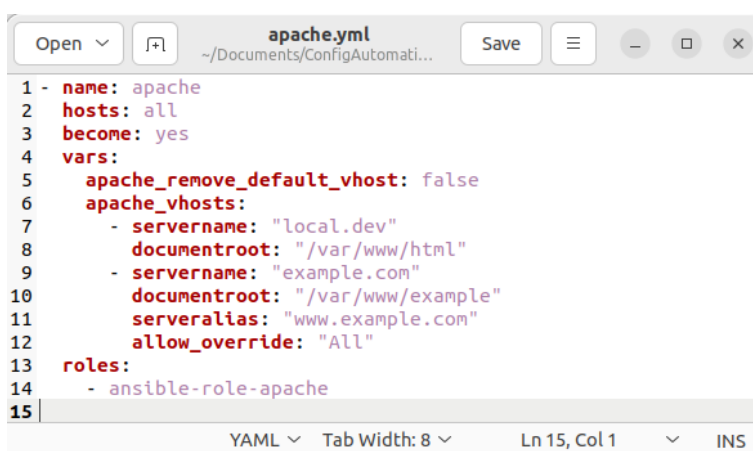
```

1 ---
2 - name: Configure Apache.
3   lineinfile:
4     dest: "{{ apache_server_root }}/conf/{{ apache_daemon }}.conf"
5     regexp: "{{ item.regexp }}"
6     line: "{{ item.line }}"
7     state: present
8     node: 0644
9   with_items: "{{ apache_ports_configuration_items }}"
10  notify: restart apache
11
12 - name: Check whether certificates defined in vhosts exist.
13   stat: path={{ item.certificate_file }}"
14   register: apache_ssl_certificates
15   with_items: "{{ apache_vhosts_ssl }}"
16
17 - name: Enable Apache mods.
18   copy:
19     dest: "{{ apache_server_root }}/conf.modules.d/99-ansible-{{ item }}.conf"
20     content: |
21       LoadModule {{ item }}_module modules/mod_{{ item }}.so
22     node: 0644
23   with_items: "{{ apache_mods_enabled }}"
24   notify: restart apache
25
26 - name: Disable Apache mods
27   file:
28     path: "{{ apache_server_root }}/conf.modules.d/99-ansible-{{ item }}.conf"
29     state: absent
30   with_items: "{{ apache_mods_disabled }}"
31   notify: restart apache
32
33 - name: Add apache vhosts configuration.
34   template:
35     src: "{{ apache_vhosts_template }}"
36     dest: "{{ apache_conf_path }}/{{ apache_vhosts_filename }}"
37     owner: root
38     group: root
39     node: 0644
40   notify: restart apache
41   when: apache_create_vhosts | bool
42
43 - name: Check if localhost cert exists (RHEL 8 and later).
44   stat:
45     path: /etc/pki/tls/certs/localhost.crt
46   register: localhost_cert
47   when: ansible_distribution_major_version | int >= 8
48
49 - name: Ensure httpd certs are installed (RHEL 8 and later).
50   command: /usr/libexec/httpd-ssl-gencerts
51   when:
52     - ansible_distribution_major_version | int >= 8
53     - not localhost_cert.stat.exists

```

FIGURE 5.6 – Playbook de configuration d’Apache pour RedHat.

Dans notre déploiement automatisé, nous avons utilisé Ansible pour configurer et déployer nos serveurs. Grâce à nos rôles personnalisés développés avec la structure Ansible Galaxy, nous avons pu installer des logiciels, configurer les serveurs et gérer les virtual hosts de manière cohérente et reproductible. Notre processus de déploiement impliquait l'exécution d'un playbook Ansible, "apache.yml" (Figure 5.7), qui orchestrait l'exécution des différentes tâches et assurait la stabilité de notre infrastructure. L'ajout de virtual hosts à notre serveur Apache était une étape essentielle, réalisée à l'aide de tâches spécifiques dans le playbook. En automatisant ces étapes, nous avons gagné du temps et garanti la fiabilité de nos déploiements. L'utilisation d'Ansible et de nos rôles personnalisés nous a permis de configurer rapidement nos serveurs et de maintenir une infrastructure stable et cohérente.



```

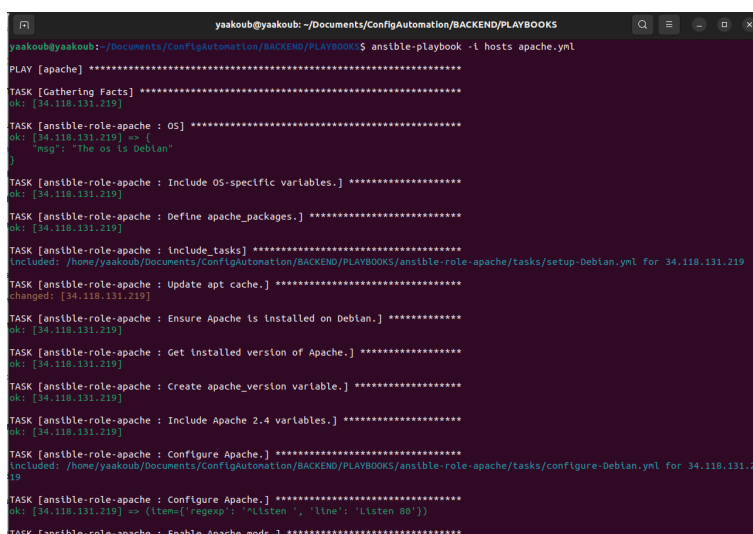
1 - name: apache
2 hosts: all
3 become: yes
4 vars:
5     apache_remove_default_vhost: false
6     apache_vhosts:
7         - servername: "local.dev"
8           documentroot: "/var/www/html"
9         - servername: "example.com"
10          documentroot: "/var/www/example"
11          serveralias: "www.example.com"
12          allow_override: "All"
13 roles:
14     - ansible-role-apache
15

```

FIGURE 5.7 – Playbook d'installation Apache.

Pour test ce Playbook avant l'intégration dans notre plateforme, nous utilisons la commande suivante : "ansible-playbook -i hosts apache.yml" .

La figure 5.8 montre la résultat de l'exécution.



```

yaakoub@yaakoub: ~/Documents/ConfigAutomation/BACKEND/PLAYBOOKS
yaakoub@yaakoub:~/Documents/ConfigAutomation/BACKEND/PLAYBOOKS$ ansible-playbook -i hosts apache.yml

PLAY [apache] *****
TASK [gathering Facts] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : OS] *****
ok: [34.118.131.219] => {
  "msg": "The os is Debian"
}

TASK [ansible-role-apache : Include OS-specific variables.] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : Define apache_packages.] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : Include_tasks] *****
included: /home/yaakoub/Documents/ConfigAutomation/BACKEND/PLAYBOOKS/ansible-role-apache/tasks/setup-Debian.yml for 34.118.131.219

TASK [ansible-role-apache : Update apt cache.] *****
changed: [34.118.131.219]

TASK [ansible-role-apache : Ensure Apache is installed on Debian.] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : Get installed version of Apache.] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : Create apache_version variable.] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : Include Apache 2.4 variables.] *****
ok: [34.118.131.219]

TASK [ansible-role-apache : Configure Apache.] *****
included: /home/yaakoub/Documents/ConfigAutomation/BACKEND/PLAYBOOKS/ansible-role-apache/tasks/configure-Debian.yml for 34.118.131.219

TASK [ansible-role-apache : Configure Apache.] *****
ok: [34.118.131.219] => {items: {'regex': 'Listen', 'line': 'Listen 80'}}

TASK [ansible-role-apache : Enable Apache mods.] *****

```

FIGURE 5.8 – Exécution d'installation Apache.

5.2.3 Interfaces réalisées

Dans cette partie, nous allons présenter les différentes interfaces réalisées de notre plateforme "ConfigAutomation"

5.2.3.1 Authentification

L'interface d'authentification (Figure 5.9) de notre plateforme de déploiement automatisé constitue le point d'entrée sécurisé pour les administrateurs. Elle leur permet de se connecter en fournissant leurs identifiants, tels que leur nom d'utilisateur et leur mot de passe, afin de vérifier leur authentification. Une fois connectés, les administrateurs ont accès à toutes les fonctionnalités de la plateforme, telles que la gestion des configurations, l'exécution automatisée, la consultation des services, la gestion des environnements cibles, la consultation des logs et la gestion des virtual hosts.

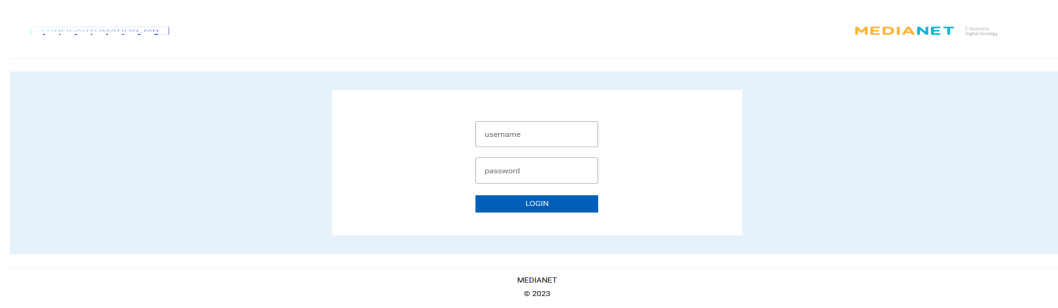


FIGURE 5.9 – Interface d'authentification.

5.2.3.2 Tableau de bord

L'interface du tableau de bord (Figure 5.10) centralise toutes les fonctionnalités de notre plateforme.

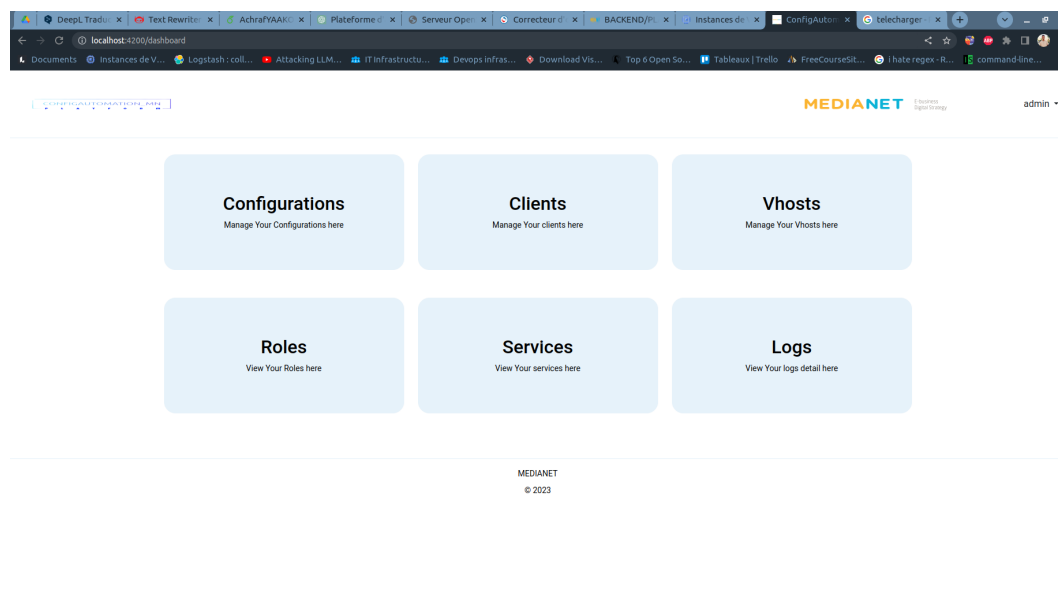


FIGURE 5.10 – Interface d'authentification.

5.2.3.3 Configuration automatisée des serveurs

L'interface de gestion des configurations (Figure 5.11) de notre plateforme offre aux administrateurs la possibilité de choisir le serveur cible sur lequel ils souhaitent déployer leurs configurations logicielles. À travers une interface conviviale, les administrateurs peuvent sélectionner parmi les serveurs disponibles, tels que les serveurs Debian 10, Debian 11, Red Hat 8 et Red Hat 9. Une fois le serveur cible choisi, notre système basé sur Spring Boot effectue un appel aux rôles Ansible correspondants en fonction de la configuration sélectionnée. Par exemple, si l'administrateur choisit de déployer une configuration basée sur Apache, notre système fera appel au rôle Ansible dédié à Apache pour installer et configurer le serveur web sur le serveur cible. De même, pour d'autres configurations telles que Nginx, PostgreSQL, Elasticsearch, MongoDB, MySQL, MariaDB, Python, UFW, Ganglia, Varnish, PHP et Java, notre système sélectionnera les rôles Ansible appropriés en fonction des besoins spécifiques de l'administrateur. Cette approche permet une gestion flexible et modulaire des configurations, en assurant que seuls les rôles nécessaires sont appelés pour chaque serveur cible, réduisant ainsi la complexité et optimisant les performances du déploiement des configurations logicielles. Une fois l'exécution des configurations terminée, une fenêtre s'ouvre dans l'interface (Figure 10.12), affichant les résultats détaillés de l'opération, y compris les logs correspondants.

The screenshot shows the 'Configurations' page of the MEDIANET platform. The page has a header with the MEDIANET logo and a user profile 'admin'. The main content area is titled 'Configurations' and includes a 'Back to Dashboard' button. Below this, there are several dropdown menus and checkboxes for configuring the server. The 'Select a client' dropdown is set to 'GCP'. The 'Select an IP address and user' dropdown is set to '34.118.131.219 - achrafy...'. The 'OS distribution' dropdown is set to 'Debian' and the 'OS version' dropdown is set to '10'. The 'select service' dropdown is set to 'Apache' and the 'select service version' dropdown is set to 'latest version'. The 'select language' dropdown is set to 'PHP' and the 'select language version' dropdown is set to '7.2'. The 'Modules' dropdown is set to 'MySQL'. There is a 'Select Other Language' button. Below these, there are checkboxes for 'Enable the mod_proxy_fcgi Apache module' (checked), 'Add Server Cache' (unchecked), 'Add Firewall' (checked), and 'Add Monitoring Tool' (checked). At the bottom right, there is a 'configure' button.

FIGURE 5.11 – Exemple de configurations.

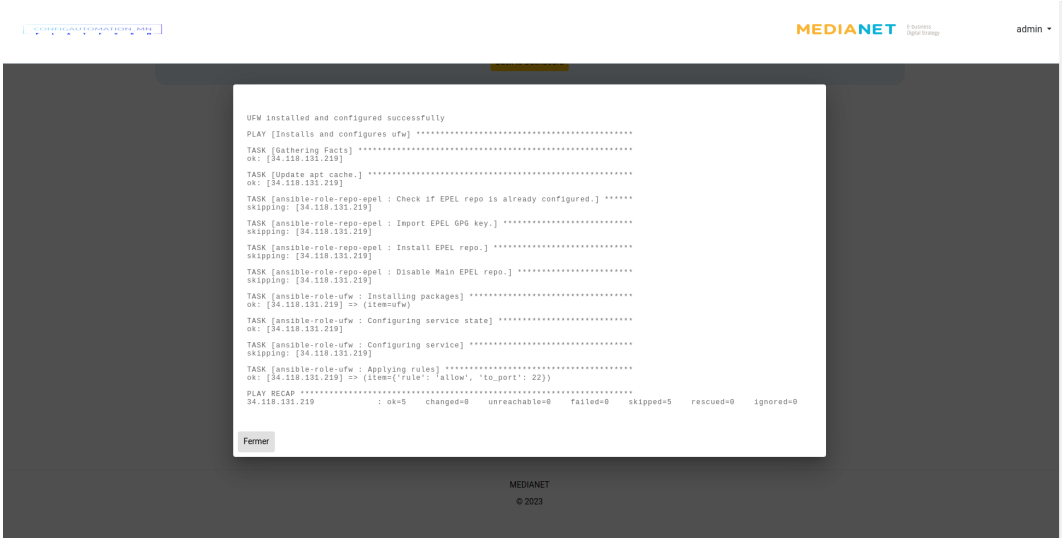


FIGURE 5.12 – Exemple de résultats détaillés pour le cas de UFW.

5.2.3.4 Gestion des clients

L’interface de gestion des clients (Figure 5.13) permet à l’administrateur de gérer les clients (serveurs) cibles sur lesquels les configurations doivent être déployées.

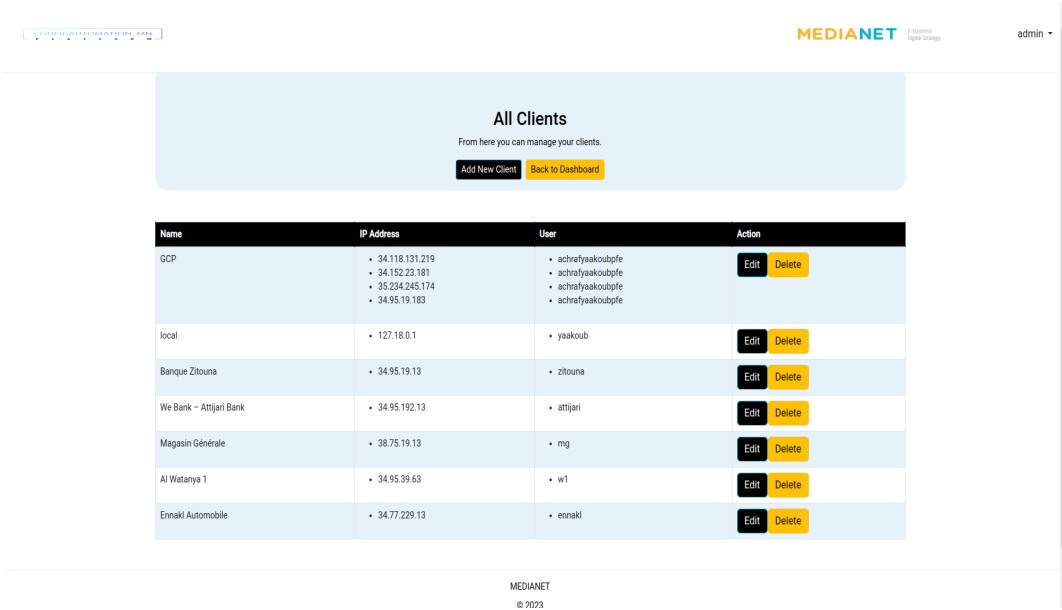


FIGURE 5.13 – Interface de gestion des clients.

The screenshot shows the 'Add Client' interface within the MEDIANET admin panel. The header includes the MEDIANET logo, the tagline 'Cloudless Digital Strategy', and an 'admin' user indicator. The main content area features a central white box with three input fields: 'name', 'ip_address', and 'users'. Below these fields are two buttons: 'Add Client' (black) and 'Back to Clients' (yellow). The footer displays 'MEDIANET © 2023'.

FIGURE 5.14 – Interface d’ajout d’un client.

5.2.3.5 Gestion des vhosts

L’interface de gestion des vhosts (Figure 5.15) vous permet de configurer facilement les virtual hosts pour vos serveurs web. Vous pouvez définir les noms de domaine, les adresses IP associées, les répertoires racines et d’autres paramètres spécifiques à chaque virtual host. Cette interface intuitive vous permet d’ajouter, de modifier et de supprimer des vhosts en quelques clics, offrant ainsi une flexibilité maximale dans la configuration de vos serveurs web. Grâce à cette fonctionnalité, vous pouvez facilement gérer et organiser vos sites web en fonction de vos besoins spécifiques.

The screenshot displays the 'All Vhosts' management interface in the MEDIANET admin panel. The header is consistent with the previous figure. The main content area has a light blue background with the title 'All Vhosts' and the subtitle 'From here you can manage your vhosts'. Below this are two buttons: 'Add New vhost' (black) and 'Back to Dashboard' (yellow). A section titled 'Select a client' shows 'GCP' as the selected client. To its right, a field 'Select an IP address and user' contains '34.152.23.181 - achrafya...'. Below these are two buttons: 'verify server connectivity' and 'Get Virtual Hosts'. A list of virtual hosts is shown below, including 'debian-11.northamerica-northeast1-a.c.vms-pfe-380010.internal', 'example.com', and 'local.dev'. The footer shows 'MEDIANET © 2023'.

FIGURE 5.15 – Interface de gestion des vhosts.

5.2.3.6 Consultation des rôles

L’interface de consultation (Figure 5.16) des rôles permet à l’administrateur de visualiser la liste des rôles Ansible disponibles dans la plateforme.

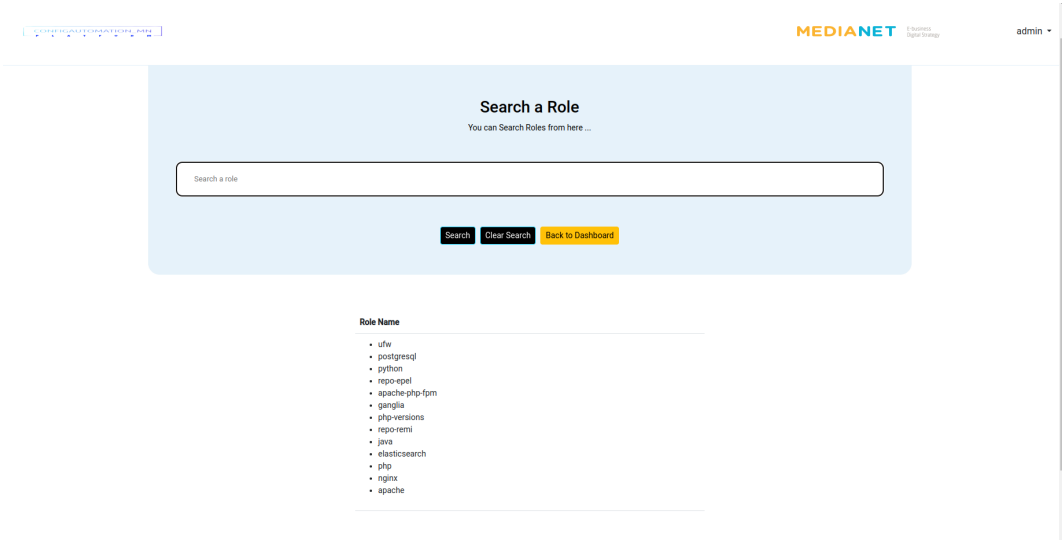


FIGURE 5.16 – Interface de consultation des rôles.

5.2.3.7 Consultation des services

L’interface de consultation des services (Figure 5.17) vous permet d’avoir une vision globale des différents services disponibles dans notre plateforme. Vous pouvez consulter les bases de données (BDD), les serveurs web, les langages de programmation et les systèmes d’exploitation (OS) pris en charge par notre solution. Cette interface vous fournit des informations détaillées sur chaque service, telles que les versions et les configurations spécifiques.

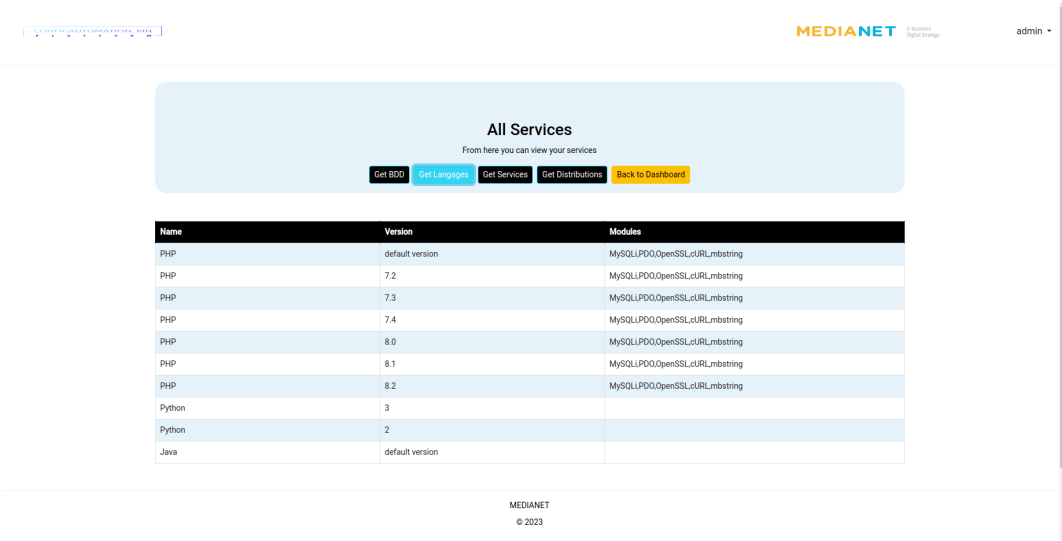


FIGURE 5.17 – Interface de Consultation des services.

5.2.3.8 Consultation des logs

L'interface de consultation des logs (Figure 5.18) permet à l'administrateur de télécharger les logs des exécutions passées, offrant ainsi un suivi détaillé et des informations de dépannage pour les configurations réalisées (Figure 5.19).

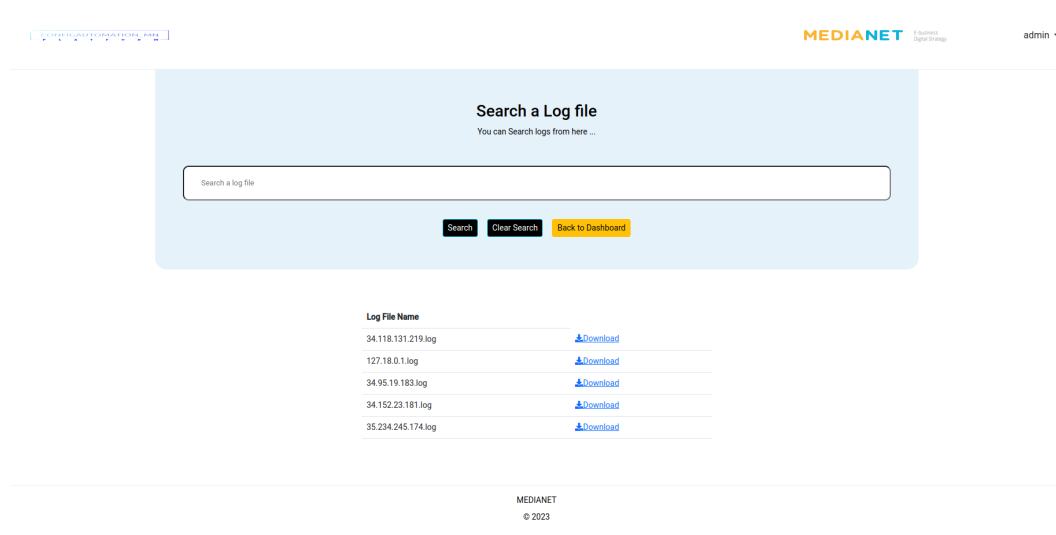


FIGURE 5.18 – Interface de Consultation des logs.



FIGURE 5.19 – Exemple d'un fichier log.

Conclusion

Dans ce chapitre, nous avons recensé les technologies utilisées au cours de ce cours. Nous avons par la suite détaillé les étapes effectuées avec les résultats attendus. Enfin, nous avons terminé avec quelques captures de nos interfaces réalisées.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Face à l'évolution constante des produits développés par les entreprises, notre projet 'ConfigAutomation' a été couronné de succès en développant une solution complète d'automatisation des configurations. Grâce à l'utilisation de la méthodologie Scrum et à la planification des sprints, nous avons pu structurer notre travail de manière efficace et itérative, ce qui a permis d'obtenir des résultats tangibles à chaque étape.

Nous avons pu répondre aux besoins des entreprises en leur offrant une plateforme conviviale et puissante pour gérer les configurations de leurs systèmes. Notre approche centrée sur l'automatisation a permis d'améliorer la productivité et d'éliminer les erreurs humaines dans le processus de configuration.

Cependant, notre projet ne s'arrête pas là. Nous envisageons plusieurs perspectives pour continuer à évoluer et à répondre aux besoins futurs des entreprises dans le domaine de l'automatisation des configurations.

Premièrement, nous prévoyons d'étendre notre solution en intégrant des fonctionnalités avancées telles que la personnalisation des configurations, afin de permettre aux utilisateurs de configurer la plateforme selon leurs besoins spécifiques. Cela offrira une plus grande adaptabilité et une meilleure correspondance aux processus métier des entreprises.

Deuxièmement, nous nous engageons à améliorer la performance et la stabilité de notre solution. Nous continuerons à optimiser les performances, à réduire les temps de traitement et à garantir une expérience utilisateur fluide et fiable. Nous accorderons également une attention particulière à la résolution des problèmes identifiés et à la mise en place d'une infrastructure solide.

Troisièmement, nous prévoyons d'explorer des opportunités d'intégration avec d'autres outils et technologies utilisés dans le domaine de la gestion des configurations. Cela permettra une meilleure interopérabilité et une expérience utilisateur plus harmonieuse en tirant parti des synergies avec d'autres acteurs de l'écosystème DevOps.

Enfin, nous engageons à maintenir notre avantage concurrentiel en suivant de près les nouvelles technologies émergentes et les tendances du secteur. Cela nous permettra d'innover en permanence et de proposer des fonctionnalités avancées qui répondent aux défis actuels et futurs de l'automatisation des configurations.

BIBLIOGRAPHIE

- [1] Medianet, <https://www.medianet.tn/fr/> [Consulté le 05/03/2023].
- [2] Ansible Architecture, <https://www.devopsschool.com/blog/understanding-ansible-architecture-using-diagram/> [Consulté le 15/02/2023].
- [3] Scrum, <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf?zoom=100> [Consulté le 05/03/2023].
- [4] Cycle de vie de SCRUM, <https://www.scrum.org/resources/scrum-framework-poster> [Consulté le 05/03/2023].
- [5] DevOps, <https://www.redhat.com/en/topics/devops> [Consulté le 06/03/2023].
- [6] Cycle DevOps, <https://www.plunge.cloud/blog/approche-devops> [Consulté le 06/03/2023].
- [7] Elasticsearch, <https://www.elastic.co/fr/elasticsearch/> [Consulté le 10/04/2023].
- [8] Foreman, <https://www.theforeman.org/introduction.html> [Consulté le 06/03/2023].
- [9] Cobbler, <https://cobbler.github.io/> [Consulté le 13/03/2023].
- [10] MVC, (<https://mouradev.wordpress.com/2015/10/29/le-pattern-mvc/>) [Consulté le 13/05/2023].
- [11] SSH, (<https://guide.ubuntu-fr.org/server/openssh-server.html>) [Consulté le 23/02/2023].