République Tunisienne

Ministère de l'Enseignement Supérieur

École Supérieure Privée d'Ingénierie et de Technologies

Rapport de Stage

1ère année du cycle d'ingénieur en Informatique

Automatisation de l'hébergement des applications web sur le Cloud

---

Réalisé par : Khalil Benlahmer
Centre National de l'Informatique

Encadrant professionnel : M. Nathir Mine Nourhen

Année Universitaire 2025–2026

Remerciements

La réalisation de ce rapport de stage n'aurait pas été possible sans le soutien et l'accompagnement de nombreuses personnes, à qui je souhaite exprimer ma profonde gratitude.

Je tiens tout d'abord à remercier le **Centre National de l'Informatique (CNI)** de m'avoir accueilli au sein de son établissement et de m'avoir offert l'opportunité d'effectuer ce stage dans un environnement professionnel enrichissant.

J'adresse mes sincères remerciements à mon encadrant professionnel,

Résumé

Dans le cadre de ma formation en informatique à l'École Supérieure Privée d'Ingénierie et de Technologies (ESPRIT), j'ai effectué un stage au sein du Centre National de l'Informatique (CNI), portant sur l'automatisation de l'hébergement et du déploiement des applications web dans un environnement Cloud virtualisé.

L'objectif de ce projet était de mettre en place une solution permettant d'automatiser et de fiabiliser le processus de déploiement d'une application web, tout en réduisant les interventions manuelles et les risques d'erreurs. Pour cela, une infrastructure virtualisée a été mise en place avec VMware Workstation et Ubuntu Server, tandis qu'Ansible a été utilisé pour automatiser l'installation, la configuration et le déploiement des différents composants. L'environnement repose notamment sur Apache HTTP Server, PHP 8.5, MariaDB 11.8, Symfony, ainsi que Git et GitHub pour la gestion du code source.

La solution développée permet d'automatiser les principales étapes du déploiement, depuis la préparation de l'environnement jusqu'à la mise en production de l'application. Afin de renforcer sa fiabilité, plusieurs mécanismes ont également été intégrés, notamment la journalisation des opérations dans les fichiers 'deployment.log' et 'error.log', ainsi qu'une gestion avancée des erreurs basée sur les blocs 'block', 'rescue' et 'always'. Un mécanisme de rollback automatique permet également de restaurer l'état précédent de l'application en cas d'échec du déploiement. Des pages d'erreur personnalisées pour les erreurs HTTP 404, 500 et 503 ont été configurées au niveau d'Apache. Enfin, un système de notifications par courrier électronique informe l'administrateur de la réussite ou de l'échec du déploiement.

Cette approche permet d'améliorer la rapidité, la fiabilité, la traçabilité et la reproductibilité du processus de déploiement, tout en facilitant les opérations de maintenance et la gestion des incidents.

Ce stage m'a permis de renforcer mes compétences en administration des systèmes Linux, virtualisation, Cloud Computing et automatisation des infrastructures, tout en mettant en pratique les bonnes pratiques de déploiement dans un environnement professionnel.

Mots-clés :

Cloud Computing, Virtualisation, VMware Workstation, Ubuntu Server, Ansible, Automatisation, Apache HTTP Server, PHP, MariaDB, Symfony, Git, GitHub, Déploie-

ment automatisé, Journalisation, Rollback, Gestion des erreurs.

Abstract

Dans le cadre de ma formation en informatique à l'École Supérieure Privée d'Ingénierie et de Technologies (ESPRIT), j'ai effectué un stage au sein du Centre National de l'Informatique (CNI).

L'objectif principal de ce stage était de concevoir et de mettre en œuvre une solution automatisée permettant d'assurer l'hébergement et le déploiement d'applications web dans un environnement Cloud virtualisé.

Pour atteindre cet objectif, plusieurs technologies ont été utilisées, notamment **VMware Workstation** pour la virtualisation, **Ubuntu Server** comme système d'exploitation, **Ansible** pour l'automatisation de l'infrastructure, **Apache HTTP Server** comme serveur web, **PHP 8.5** pour l'exécution de l'application Symfony, **MariaDB 11.8** comme système de gestion de bases de données, ainsi que **Git et GitHub** pour la gestion des versions du code source.

La solution proposée permet d'automatiser le processus de déploiement d'une application web, depuis l'installation des logiciels nécessaires jusqu'à la configuration des différents services. Cette approche permet de réduire les interventions manuelles, de limiter les erreurs de configuration et d'améliorer la rapidité, la fiabilité et la reproductibilité des déploiements.

Ce stage m'a permis de renforcer mes connaissances en administration des systèmes Linux, en virtualisation, en Cloud Computing et en automatisation des infrastructures, tout en acquérant une expérience professionnelle enrichissante.

Mot-clés :

Cloud Computing, Virtualisation, VMware, Ubuntu Server, Ansible, Apache HTTP Server, PHP, MariaDB, Git, GitHub, Symfony, Déploiement automatisé.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

API	Application Programming Interface
CNI	Centre National de l'Informatique
CPU	Central Processing Unit
DBaaS	Database as a Service
DNS	Domain Name System
ESXi	Elastic Sky X Integrated
FaaS	Function as a Service
Git	Global Information Tracker
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IaaS	Infrastructure as a Service
IP	Internet Protocol
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
PaaS	Platform as a Service
PHP	PHP : Hypertext Preprocessor
RAM	Random Access Memory
SaaS	Software as a Service
SSH	Secure Shell
VM	Virtual Machine
VMware	Virtual Machine Software
YAML	YAML Ain't Markup Language

Introduction générale

L'évolution rapide des technologies de l'information et de la communication a profondément transformé les méthodes de développement, de déploiement et d'administration des applications informatiques. Aujourd'hui, les entreprises recherchent des solutions capables de réduire les coûts d'exploitation, d'améliorer la disponibilité des services et de simplifier la gestion des infrastructures. Dans ce contexte, le Cloud Computing s'est imposé comme une solution incontournable grâce à sa flexibilité, son évolutivité et sa capacité à fournir des ressources informatiques à la demande.

Parallèlement au développement du cloud, les outils d'automatisation des infrastructures connaissent une adoption croissante. Ils permettent d'automatiser les tâches répétitives d'administration système, de réduire les erreurs humaines et d'assurer un déploiement rapide, fiable et reproductible des applications.

C'est dans ce contexte que s'inscrit le présent stage, effectué au sein du Centre National de l'Informatique (CNI). Ce projet avait pour objectif de concevoir et de mettre en œuvre une solution permettant d'automatiser l'hébergement et le déploiement d'une application web Symfony dans un environnement virtualisé.

Pour atteindre cet objectif, plusieurs technologies ont été utilisées, notamment VMware Workstation pour la virtualisation, Ubuntu Server comme système d'exploitation, Ansible pour l'automatisation des tâches d'administration, Apache HTTP Server comme serveur web, PHP pour l'exécution de l'application Symfony, MariaDB comme système de gestion de base de données ainsi que Git et GitHub pour la gestion du code source.

Le présent rapport est organisé en quatre chapitres :

- Le premier chapitre présente une étude générale sur le Cloud Computing, son architecture, ses modèles de services ainsi que ses principaux avantages.
- Le deuxième chapitre est consacré à la virtualisation, à ses principes de fonctionnement ainsi qu'aux principales solutions utilisées dans ce domaine.
- Le troisième chapitre décrit l'étude de la solution proposée, l'environnement de travail ainsi que l'architecture générale du projet.
- Enfin, le quatrième chapitre présente la réalisation pratique du projet, les différentes étapes de configuration de l'environnement ainsi que l'automatisation du déploiement de l'application web à l'aide d'Ansible.

1 Étude générale sur le Cloud Computing

1.1 Introduction

L'évolution rapide des technologies de l'information a profondément changé la manière dont les entreprises utilisent les ressources informatiques. Les infrastructures traditionnelles nécessitent souvent des investissements importants en matériel, en maintenance et en administration.

Afin de répondre aux besoins croissants en puissance de calcul, stockage et disponibilité des services, le Cloud Computing est apparu comme une approche moderne permettant d'accéder à des ressources informatiques à la demande.

Le Cloud Computing offre une grande flexibilité grâce à la virtualisation, l'automatisation et la mutualisation des ressources. Il permet aux entreprises de déployer rapidement leurs applications tout en optimisant les coûts d'exploitation.

1.2 Historique du Cloud Computing

Le concept du Cloud Computing trouve son origine dans les années 1960 avec les premiers travaux sur le partage des ressources informatiques.

Dans les années 1990, l'apparition d'Internet et des technologies de virtualisation a permis l'évolution vers des infrastructures distribuées.

Au début des années 2000, plusieurs entreprises ont commencé à proposer des services Cloud accessibles via Internet. Parmi les acteurs majeurs figurent Amazon Web Services, Microsoft Azure et Google Cloud Platform.

Aujourd'hui, le Cloud Computing constitue un élément essentiel de la transformation numérique des entreprises.

1.3 Définition du Cloud Computing

Selon le National Institute of Standards and Technology (NIST), le Cloud Computing est un modèle permettant un accès réseau pratique et à la demande à un ensemble

partagé de ressources informatiques configurables.

Ces ressources peuvent être :

— des serveurs ;
— du stockage ;
— des bases de données ;
— des réseaux ;
— des applications.

L'utilisateur peut exploiter ces ressources sans avoir besoin de gérer directement l'infrastructure physique.

1.4 Caractéristiques principales du Cloud Computing

Le Cloud Computing repose sur cinq caractéristiques principales.

1.4.1 Libre-service à la demande

L'utilisateur peut créer, modifier ou supprimer des ressources informatiques automatiquement selon ses besoins.

1.4.2 Accès réseau étendu

Les services Cloud sont accessibles depuis différents équipements à travers Internet.

1.4.3 Mutualisation des ressources

Les ressources physiques sont partagées entre plusieurs utilisateurs grâce aux mécanismes de virtualisation.

1.4.4 Élasticité rapide

Le Cloud permet d'adapter automatiquement les ressources selon la charge de travail.

1.4.5 Service mesurable

Les ressources utilisées peuvent être surveillées et facturées selon la consommation réelle.

1.5 Architecture du Cloud Computing

L'architecture Cloud est généralement organisée en plusieurs couches.

1.5.1 Couche physique

Cette couche représente l'ensemble des ressources matérielles :

- serveurs physiques ;
- systèmes de stockage ;
- équipements réseau.

1.5.2 Couche de virtualisation

La virtualisation permet de créer plusieurs machines virtuelles à partir d'un même serveur physique.

Elle est assurée par un hyperviseur permettant :

- la création des machines virtuelles ;
- l'allocation des ressources ;
- l'isolation des environnements.

1.5.3 Couche de services

Cette couche fournit les services accessibles aux utilisateurs :

- Infrastructure ;
- Plateforme ;
- Applications.

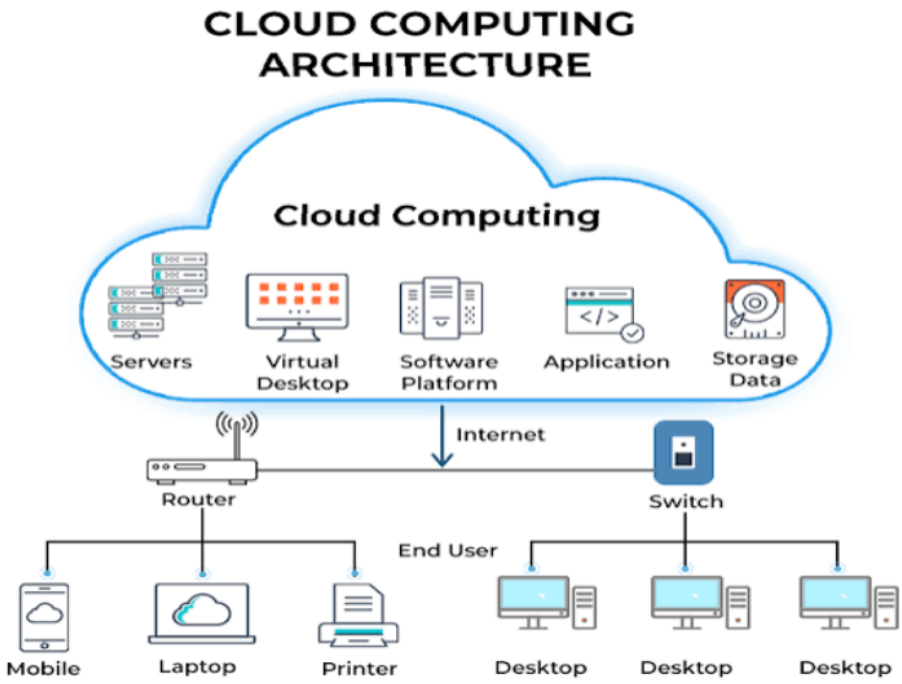

FIGURE 1.1 – Architecture générale du Cloud Computing

1.6 Les modèles de services Cloud

Le Cloud Computing propose plusieurs modèles de services.

1.6.1 Infrastructure as a Service (IaaS)

L'IaaS fournit une infrastructure informatique virtuelle comprenant :

- des machines virtuelles ;
- du stockage ;
- des réseaux virtuels.

L'utilisateur possède un contrôle important sur le système d'exploitation et les applications.

Exemples :

- Amazon EC2 ;
- Microsoft Azure Virtual Machines.

1.6.2 Platform as a Service (PaaS)

Le modèle PaaS fournit une plateforme complète permettant aux développeurs de créer et déployer leurs applications sans gérer l'infrastructure.

Il comprend généralement :

— environnement d'exécution ;
— outils de développement ;
— bases de données.

1.6.3 Software as a Service (SaaS)

Le SaaS permet d'utiliser directement une application accessible via Internet.

L'utilisateur n'a pas besoin d'installer ou de maintenir l'application.

Exemples :

— messagerie électronique ;
— suites bureautiques en ligne.

1.6.4 Database as a Service (DBaaS)

Le DBaaS fournit une base de données gérée dans le Cloud.

Le fournisseur assure :

— l'installation ;
— les sauvegardes ;
— la maintenance.

1.6.5 Function as a Service (FaaS)

Le FaaS permet l'exécution de fonctions informatiques sans gérer les serveurs.

Cette approche est utilisée dans les architectures Serverless.

1.7 Les modèles de déploiement Cloud

1.7.1 Cloud public

Le Cloud public est accessible à plusieurs utilisateurs. Les ressources sont hébergées par un fournisseur externe.

1.7.2 Cloud privé

Le Cloud privé est réservé à une seule organisation. Il offre un meilleur contrôle et une sécurité renforcée.

1.7.3 Cloud hybride

Le Cloud hybride combine un Cloud privé et un Cloud public.

Il permet de conserver certaines données sensibles en interne tout en utilisant les ressources publiques.

1.7.4 Multi-Cloud

Le Multi-Cloud consiste à utiliser plusieurs fournisseurs Cloud afin d'améliorer la disponibilité et réduire la dépendance envers un seul acteur.

1.8 Avantages du Cloud Computing

Les principaux avantages du Cloud sont :

- Réduction des coûts d'infrastructure ;
- Flexibilité et évolutivité ;
- Déploiement rapide des applications ;
- Haute disponibilité des services ;
- Accès aux ressources depuis n'importe quel endroit.

1.9 Limites et défis du Cloud Computing

Malgré ses avantages, le Cloud présente certains défis :

- dépendance à la connexion Internet ;
- problèmes de confidentialité des données ;
- risques liés à la sécurité ;
- dépendance envers le fournisseur Cloud.

1.10 Sécurité dans le Cloud

La sécurité représente un aspect essentiel dans les environnements Cloud.

Les principales mesures utilisées sont :

- chiffrement des données ;
- authentification forte ;
- contrôle des accès ;
- sauvegardes régulières ;
- surveillance des activités.

1.11 Virtualisation : base du Cloud Computing

La virtualisation constitue la technologie fondamentale permettant l'utilisation efficace des ressources matérielles.

Elle permet de créer plusieurs environnements virtuels indépendants sur un même serveur physique.

Dans notre projet, la virtualisation est réalisée avec VMware Workstation afin de créer plusieurs machines virtuelles utilisées pour le déploiement automatisé de l'application web.

1.12 Conclusion

Le Cloud Computing représente aujourd'hui une solution majeure pour les entreprises souhaitant moderniser leurs infrastructures informatiques.

Grâce à ses différents modèles de services et de déploiement, il permet d'obtenir des ressources flexibles, évolutives et accessibles à la demande.

La virtualisation constitue l'élément fondamental de cette technologie. Le chapitre suivant sera consacré à l'étude de la virtualisation ainsi qu'aux différentes solutions utilisées dans les environnements professionnels.

2 La virtualisation

2.1 Introduction

La virtualisation est une technologie fondamentale dans les infrastructures informatiques modernes. Elle permet de créer plusieurs environnements informatiques virtuels indépendants à partir d'une seule machine physique.

Grâce à cette technologie, les entreprises peuvent optimiser l'utilisation de leurs ressources matérielles, réduire les coûts d'exploitation et faciliter l'administration des systèmes.

La virtualisation représente également une composante essentielle du Cloud Computing, car elle permet aux fournisseurs Cloud de proposer des ressources informatiques dynamiques et évolutives.

2.2 Définition de la virtualisation

La virtualisation consiste à utiliser une couche logicielle appelée hyperviseur afin de simuler des ressources matérielles virtuelles.

Cette technologie permet d'exécuter plusieurs systèmes d'exploitation différents sur un même serveur physique tout en garantissant une isolation entre les environnements.

Chaque environnement virtuel créé est appelé :

— Machine Virtuelle (Virtual Machine ou VM) ;

2.3 Principe de fonctionnement

Le fonctionnement de la virtualisation repose sur trois éléments principaux :

— **Machine physique (Host)** : serveur réel contenant les ressources matérielles.
— **Hyperviseur** : logiciel permettant la création et la gestion des machines virtuelles.
— **Machines virtuelles (Guests)** : environnements virtuels contenant leurs propres systèmes d'exploitation et applications.

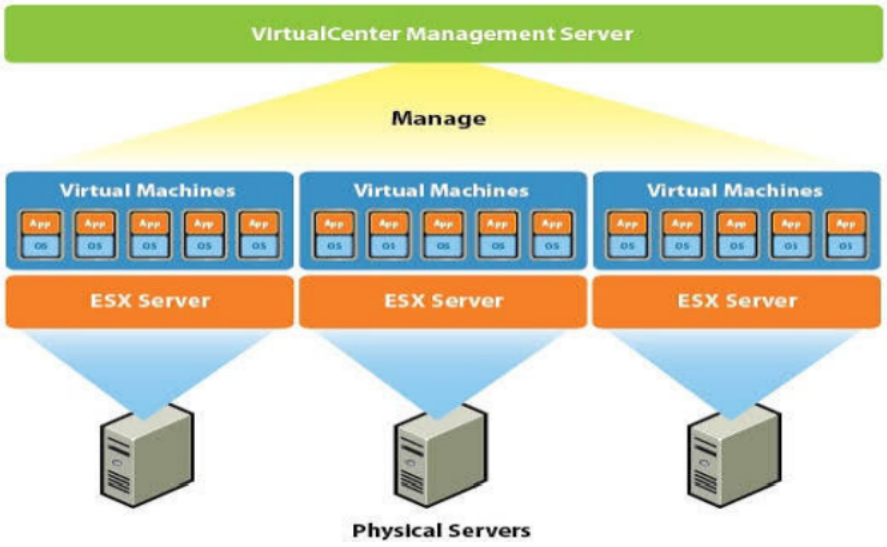

FIGURE 2.1 – Architecture générale de la virtualisation

2.4 Les types d'hyperviseurs

Un hyperviseur est un composant logiciel permettant d'exécuter plusieurs machines virtuelles sur un même serveur.

2.4.1 Hyperviseur de type 1 (Bare Metal)

L'hyperviseur de type 1 fonctionne directement sur le matériel physique sans nécessiter un système d'exploitation intermédiaire.

Il offre de meilleures performances et est principalement utilisé dans les centres de données professionnels.

Exemples :

— VMware ESXi ;
— Microsoft Hyper-V Server ;
— Xen.

2.4.2 Hyperviseur de type 2 (Hosted)

L'hyperviseur de type 2 fonctionne au-dessus d'un système d'exploitation existant.

Il est généralement utilisé pour les environnements de développement, de test et d'apprentissage.

Exemples :

— VMware Workstation ;
— Oracle VirtualBox ;
— Parallels Desktop.

2.5 Comparaison entre les hyperviseurs Type 1 et Type 2

Critère	Type 1	Type 2
Installation	Directement sur le matériel	Au-dessus d'un système d'exploitation
Performance	Très élevée	Moins élevée
Utilisation	Production et Data Centers	Développement et tests
Exemples	VMware ESXi, Hyper-V	VMware Workstation, VirtualBox

TABLE 2.1 – Comparaison des types d'hyperviseurs

2.6 Les solutions VMware

VMware est l'un des principaux acteurs dans le domaine de la virtualisation.

Ses solutions sont largement utilisées dans les infrastructures professionnelles.

2.6.1 VMware Workstation

VMware Workstation est un hyperviseur de type 2 permettant de créer et d'exécuter plusieurs machines virtuelles sur un ordinateur physique.

Dans le cadre de ce projet, VMware Workstation a été utilisé afin de créer un environnement de test composé de plusieurs machines virtuelles Ubuntu.

2.6.2 VMware ESXi

VMware ESXi est un hyperviseur de type 1 destiné aux infrastructures professionnelles.

Il permet d'installer directement une plateforme de virtualisation sur des serveurs physiques.

Ses principales fonctionnalités sont :

- gestion centralisée des machines virtuelles ;
- allocation dynamique des ressources ;
- haute disponibilité ;
- migration des machines virtuelles.

2.6.3 VMware vCenter

VMware vCenter est une plateforme d'administration permettant de gérer plusieurs serveurs ESXi depuis une interface centralisée.

Il offre :

— supervision des ressources ;
— gestion des machines virtuelles ;
— automatisation des opérations ;
— suivi des performances.

2.7 Les composants d'une machine virtuelle

Une machine virtuelle possède plusieurs composants virtuels.

2.7.1 Processeur virtuel (vCPU)

Le processeur virtuel représente une partie des capacités du processeur physique attribuée à la machine virtuelle.

2.7.2 Mémoire virtuelle (RAM)

La mémoire virtuelle correspond à une quantité de mémoire RAM allouée depuis la machine physique.

2.7.3 Stockage virtuel

Le stockage virtuel est généralement représenté sous forme de fichiers contenant les disques virtuels des machines.

2.7.4 Réseau virtuel

La virtualisation réseau permet aux machines virtuelles de communiquer entre elles ou avec des réseaux externes.

Dans VMware, plusieurs modes réseau existent :

— NAT ;
— Bridge ;
— Host-only.

2.8 Avantages de la virtualisation

La virtualisation présente plusieurs avantages :

2.8.1 Optimisation des ressources

Elle permet d'utiliser efficacement les ressources matérielles en exécutant plusieurs machines virtuelles sur un même serveur.

2.8.2 Réduction des coûts

La diminution du nombre de serveurs physiques permet de réduire :

- les coûts matériels ;
- la consommation énergétique ;
- les coûts de maintenance.

2.8.3 Isolation des environnements

Chaque machine virtuelle fonctionne indépendamment des autres.

Une panne dans une machine virtuelle n'affecte pas directement les autres.

2.8.4 Facilité de déploiement

Les machines virtuelles peuvent être facilement :

- copiées ;
- sauvegardées ;
- restaurées ;
- déplacées.

2.9 Virtualisation et Cloud Computing

La virtualisation constitue la base technique du Cloud Computing.

Les fournisseurs Cloud utilisent des technologies de virtualisation afin de fournir :

- des serveurs virtuels ;
- du stockage à la demande ;
- des réseaux virtuels ;
- des environnements isolés.

Dans notre projet, la virtualisation permet de simuler une infrastructure Cloud privée composée de plusieurs machines virtuelles :

- une machine de contrôle Ansible ;
- un serveur d'application Symfony ;
- un serveur de base de données MariaDB.

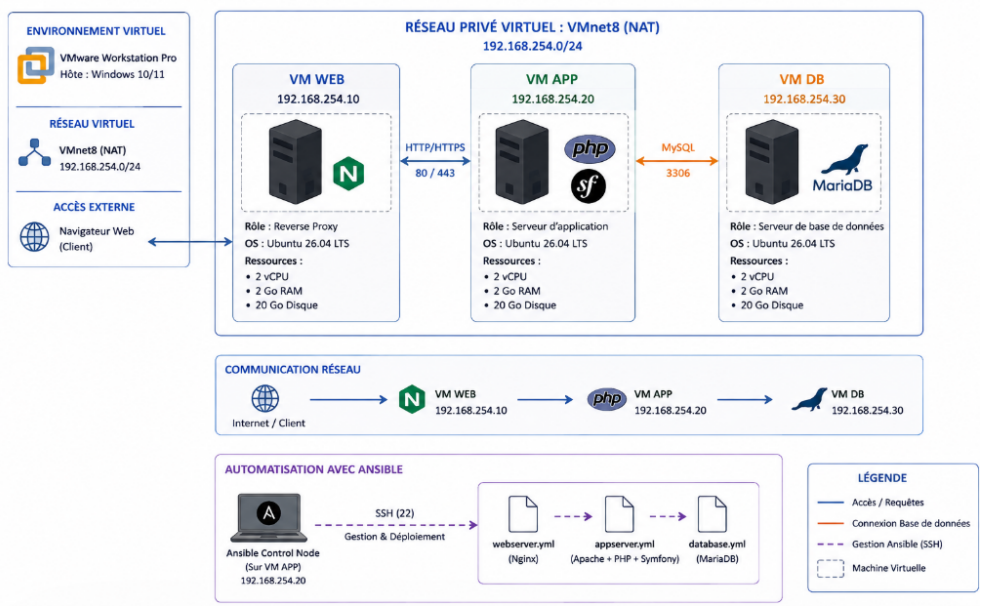

FIGURE 2.2 – Infrastructure virtuelle utilisée dans le projet

2.10 Comparaison entre infrastructure physique et virtuelle

Critère	**Physique**	**Virtuelle**
Matériel	Un serveur par application	Plusieurs VM sur un serveur
Déploiement	Long	Rapide
Gestion	Administration individuelle	Gestion centralisée
Flexibilité	Limitée	Très élevée

TABLE 2.2 – Comparaison infrastructure physique et virtuelle

2.11 Conclusion

La virtualisation représente une technologie essentielle permettant d'optimiser les ressources informatiques et de construire des infrastructures flexibles.

Grâce aux hyperviseurs tels que VMware, il est possible de créer des environnements isolés adaptés aux besoins des entreprises.

Dans le chapitre suivant, nous présenterons l'étude et la conception de la solution proposée, ainsi que l'environnement matériel et logiciel utilisé pour réaliser ce projet.

3 Étude et conception de la solution

3.1 Introduction

Après avoir présenté les concepts fondamentaux du Cloud Computing et de la virtualisation, ce chapitre est consacré à l'étude du contexte du projet, à l'analyse des besoins ainsi qu'à la conception de la solution proposée.

L'objectif principal de ce projet est de mettre en place une solution permettant d'automatiser le déploiement d'une application web Symfony dans un environnement virtualisé en utilisant l'outil d'automatisation Ansible.

Cette solution vise à simplifier les opérations d'administration système, à réduire les interventions manuelles, à améliorer la fiabilité des configurations et à garantir un processus de déploiement reproductible.

Le chapitre présente tout d'abord l'organisme d'accueil et le contexte du stage. Ensuite, les besoins fonctionnels et non fonctionnels sont identifiés. Enfin, l'architecture de la solution ainsi que la conception du processus de déploiement sont présentées.

3.2 Présentation de l'organisme d'accueil

3.2.1 Présentation du Centre National de l'Informatique

Le Centre National de l'Informatique (CNI) est un établissement public tunisien spécialisé dans le domaine des technologies de l'information et de la communication.

Dans le cadre de ses activités, le CNI accompagne les organismes publics dans leurs projets informatiques et contribue à la mise en place et à l'exploitation de solutions numériques.

Ses activités couvrent notamment plusieurs domaines liés aux systèmes d'information et aux infrastructures informatiques, tels que :

- le développement et la maintenance des applications informatiques ;
- l'hébergement des applications et des systèmes d'information ;
- l'administration des infrastructures informatiques ;
- la gestion des systèmes et des réseaux ;
- la sécurité des systèmes d'information ;
- l'accompagnement des organismes publics dans leurs projets de transformation

numérique.

Grâce à ses compétences techniques et à ses infrastructures, le CNI participe au développement et à la modernisation des services numériques destinés aux organismes publics.

FIGURE 3.1 – Centre National de l'Informatique

3.2.2 Rôle de l'organisme d'accueil dans le projet

Dans le cadre de ce stage, le CNI constitue l'environnement professionnel dans lequel le projet d'automatisation a été étudié et réalisé.

Le projet s'inscrit dans une problématique d'administration et d'hébergement d'applications web. L'objectif est notamment d'étudier comment l'automatisation peut faciliter les opérations de déploiement et permettre de reproduire plus facilement une configuration serveur.

Cette démarche s'inscrit dans une logique de modernisation des méthodes d'administration des infrastructures informatiques.

3.3 Contexte du stage

Avec l'évolution des applications web et l'augmentation des besoins en ressources informatiques, les opérations d'installation, de configuration et de déploiement deviennent de plus en plus nombreuses.

Dans une approche traditionnelle, l'administrateur doit effectuer manuellement plusieurs opérations sur les serveurs :

- installation du système d'exploitation ;
- installation du serveur web ;
- installation de PHP et de ses extensions ;
- installation et configuration du système de gestion de base de données ;
- création des bases de données et des utilisateurs ;
- récupération du code source ;
- installation des dépendances de l'application ;
- configuration du serveur web ;
- configuration des permissions ;
- redémarrage et vérification des services.

Lorsque ces opérations sont répétées sur plusieurs serveurs, elles peuvent devenir longues et entraîner des différences de configuration entre les environnements.

L'automatisation permet donc de définir les différentes opérations sous forme de scripts et de playbooks pouvant être exécutés de manière reproductible.

3.4 Problématique

Le déploiement manuel d'une application web présente plusieurs limites, notamment en matière de temps d'exécution, de reproductibilité et de risque d'erreur humaine.

La problématique principale de ce projet est donc la suivante :

Comment automatiser le déploiement d'une application web Symfony dans un environnement virtualisé tout en garantissant une configuration fiable, rapide et reproductible ?

Pour répondre à cette problématique, une solution reposant sur la virtualisation VMware et l'automatisation avec Ansible a été proposée.

3.5 Objectifs du projet

3.5.1 Objectif général

L'objectif général du projet est de concevoir et de mettre en place une infrastructure permettant d'automatiser l'installation, la configuration et le déploiement d'une application web Symfony dans un environnement virtualisé.

3.5.2 Objectifs spécifiques

Les objectifs spécifiques du projet sont les suivants :

— créer un environnement virtualisé avec VMware Workstation ;
— mettre en place plusieurs machines virtuelles Ubuntu ;
— configurer la communication réseau entre les machines ;
— mettre en place une communication SSH sécurisée ;
— installer et configurer Ansible ;
— définir un inventaire des machines administrées ;
— automatiser l'installation du serveur web Apache ;
— automatiser l'installation et la configuration de PHP ;
— automatiser l'installation de MariaDB ;
— automatiser la création de la base de données ;
— récupérer automatiquement le code source depuis GitHub ;
— installer automatiquement les dépendances avec Composer ;
— configurer l'application Symfony ;
— configurer le serveur Apache pour l'application ;
— vérifier le bon fonctionnement de l'application après déploiement ;
— enregistrer les résultats du processus de déploiement.

3.6 Analyse de l'existant

Avant la mise en place de la solution automatisée, le déploiement d'une application web nécessite plusieurs opérations réalisées manuellement.

L'administrateur doit notamment installer les différents composants, configurer les services, récupérer le code source et effectuer les différentes vérifications nécessaires.

Le processus traditionnel peut être résumé comme suit :

1. installation du système d'exploitation ;
2. configuration du réseau ;
3. installation d'Apache ;
4. installation de PHP ;
5. installation de MariaDB ;
6. configuration de la base de données ;
7. récupération du projet depuis GitHub ;
8. installation des dépendances avec Composer ;
9. configuration de Symfony ;
10. configuration du VirtualHost Apache ;
11. redémarrage des services ;
12. vérification du fonctionnement de l'application.

3.6.1 Limites de l'approche manuelle

Cette méthode présente plusieurs limites :

— temps de déploiement important ;
— répétition de tâches similaires ;
— risque d'erreurs humaines ;
— difficulté à reproduire exactement une configuration ;
— risque d'oubli d'une étape ;
— difficulté de maintenir plusieurs serveurs avec une configuration identique.

Ces limites justifient l'utilisation d'une solution d'automatisation.

3.7 Analyse des besoins

L'analyse des besoins permet de déterminer les fonctionnalités que la solution doit fournir ainsi que les contraintes auxquelles elle doit répondre.

3.7.1 Besoins fonctionnels

La solution doit permettre :

— d'administrer plusieurs serveurs à distance ;

— de vérifier la disponibilité des serveurs ;
— d'installer automatiquement les logiciels nécessaires ;
— de configurer le serveur web ;
— de configurer le serveur de base de données ;
— de récupérer le code source depuis un dépôt Git ;
— d'installer les dépendances de l'application ;
— de déployer l'application Symfony ;
— de configurer les permissions nécessaires ;
— de redémarrer les services ;
— de vérifier le résultat du déploiement ;
— d'enregistrer les informations relatives au déploiement.

3.7.2 Besoins non fonctionnels

La solution doit également respecter plusieurs exigences non fonctionnelles.

— **Fiabilité :** les opérations doivent être exécutées de manière contrôlée afin de limiter les erreurs ;
— **Reproductibilité :** une même configuration doit pouvoir être reproduite sur plusieurs machines ;
— **Automatisation :** le nombre d'interventions manuelles doit être réduit ;
— **Sécurité :** les communications entre le serveur de contrôle et les serveurs cibles doivent utiliser SSH ;
— **Maintenabilité :** les tâches doivent être organisées dans des fichiers Ansible faciles à modifier ;
— **Évolutivité :** l'architecture doit permettre d'ajouter d'autres serveurs si nécessaire.

3.8 Solution proposée

Afin de répondre aux besoins identifiés, une solution basée sur l'automatisation avec Ansible a été développée.

La solution repose sur une infrastructure virtuelle composée de plusieurs machines Ubuntu exécutées avec VMware Workstation.

L'architecture comprend trois machines principales :

— un serveur de contrôle Ansible ;
— un serveur d'application ;
— un serveur de base de données.

Le serveur Ansible joue le rôle de machine de contrôle. Il contient l'inventaire ainsi que les différents playbooks permettant d'automatiser les opérations.

Le serveur d'application héberge Apache, PHP et l'application Symfony.

Le serveur de base de données héberge MariaDB et la base utilisée par l'application.

3.9 Architecture générale de la solution

L'infrastructure mise en place est composée de trois machines virtuelles connectées à travers un réseau privé VMware.

La figure suivante présente l'architecture générale de la solution.

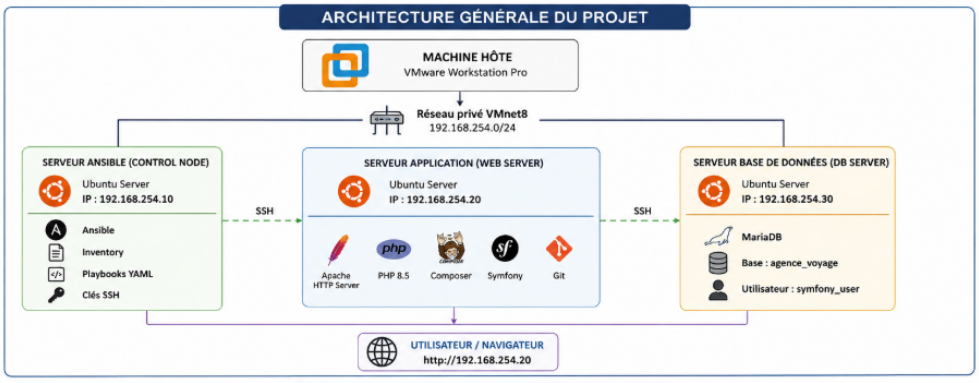

FIGURE 3.2 – Architecture générale de la solution

Le serveur Ansible communique avec les serveurs cibles à travers le protocole SSH.

Le serveur d'application communique ensuite avec le serveur de base de données afin d'accéder aux données nécessaires au fonctionnement de l'application Symfony.

3.9.1 Description des machines

Les machines virtuelles utilisées sont présentées dans le tableau suivant.

Machine	Adresse IP	Rôle
Ansible Control	192.168.254.10	Serveur de contrôle
Application Server	192.168.254.20	Apache + PHP + Symfony
Database Server	192.168.254.30	MariaDB

TABLE 3.1 – Machines virtuelles utilisées

3.10 Architecture logique

L'architecture logique peut être divisée en plusieurs niveaux.

3.10.1 Couche de contrôle

Cette couche est représentée par le serveur Ansible.

Elle contient :

— l'inventaire ;
— les playbooks ;
— les variables de configuration ;
— les tâches d'automatisation.

3.10.2 Couche applicative

Cette couche correspond au serveur d'application.

Elle contient principalement :

— Apache HTTP Server ;
— PHP ;
— Composer ;
— l'application Symfony.

3.10.3 Couche données

Cette couche correspond au serveur de base de données.

Elle contient :

— MariaDB ;
— la base de données utilisée par Symfony ;
— l'utilisateur de connexion à la base de données.

3.11 Environnement matériel

L'environnement matériel utilisé pour réaliser le projet est constitué d'un ordinateur hôte permettant l'exécution simultanée de plusieurs machines virtuelles.

Les ressources nécessaires comprennent notamment :

— un processeur compatible avec la virtualisation matérielle ;
— une quantité de mémoire RAM suffisante pour exécuter plusieurs machines virtuelles ;
— un espace de stockage suffisant pour les systèmes virtuels ;
— une connexion réseau permettant aux machines virtuelles de communiquer.

3.12 Environnement logiciel

3.12.1 VMware Workstation

VMware Workstation est utilisé comme plateforme de virtualisation.

Il permet de créer, configurer et exécuter les différentes machines virtuelles utilisées dans le cadre du projet.

3.12.2 Ubuntu Server

Ubuntu est utilisé comme système d'exploitation des machines virtuelles.

Il fournit l'environnement Linux nécessaire à l'installation des différents services et outils utilisés dans le projet.

3.12.3 Ansible

Ansible est l'outil principal utilisé pour l'automatisation.

Il permet d'exécuter des tâches d'administration sur les serveurs distants à partir d'une machine de contrôle.

Les configurations sont principalement décrites dans des fichiers YAML appelés *playbooks*.

3.12.4 Apache HTTP Server

Apache est utilisé comme serveur web.

Il reçoit les requêtes HTTP des utilisateurs et les transmet à l'application Symfony.

3.12.5 PHP

PHP constitue l'environnement d'exécution utilisé par l'application Symfony.

Plusieurs extensions PHP nécessaires au fonctionnement de l'application sont également installées.

3.12.6 Composer

Composer est utilisé pour gérer les dépendances PHP du projet Symfony.

Lors du déploiement, la commande `composer install` permet d'installer les bibliothèques définies par le projet.

3.12.7 MariaDB

MariaDB est utilisé comme système de gestion de base de données relationnelle.

Il stocke les données utilisées par l'application Symfony.

3.12.8 Git et GitHub

Git est utilisé pour la gestion du code source.

Le projet est hébergé dans un dépôt GitHub et peut être récupéré automatiquement par le serveur d'application lors du déploiement.

3.13 Configuration réseau

Les machines virtuelles communiquent à travers un réseau privé configuré dans VMware.

La configuration utilisée est la suivante :

```
Ansible      : 192.168.254.10
Application  : 192.168.254.20
Database     : 192.168.254.30
```

Cette configuration permet au serveur Ansible d'accéder aux serveurs cibles et permet également au serveur d'application de communiquer avec le serveur de base de données.

3.13.1 Communication SSH

Ansible utilise SSH pour exécuter les tâches sur les machines distantes.

Une clé SSH est configurée sur le serveur Ansible afin de permettre une connexion automatisée.

Le principe de communication est :

Serveur Ansible → SSH → Serveur Application / Serveur Base de données

3.14 Méthodologie adoptée

Une démarche progressive a été adoptée pour réaliser le projet.

Les principales étapes sont :

1. étude des concepts liés au Cloud Computing et à la virtualisation ;
2. préparation de l'environnement VMware ;
3. création des machines virtuelles ;
4. configuration du réseau ;
5. configuration des connexions SSH ;
6. installation d'Ansible ;
7. création de l'inventaire ;
8. développement des playbooks ;
9. automatisation de la configuration des serveurs ;
10. automatisation du déploiement de Symfony ;
11. réalisation des tests ;
12. analyse des résultats.

3.15 Conception du déploiement

Le processus de déploiement a été conçu afin de réduire au maximum les interventions manuelles.

Le serveur Ansible constitue le point de départ du processus.

Après le lancement du playbook, Ansible établit les connexions SSH avec les serveurs cibles et exécute les différentes tâches définies.

Le processus général est illustré par la figure suivante.

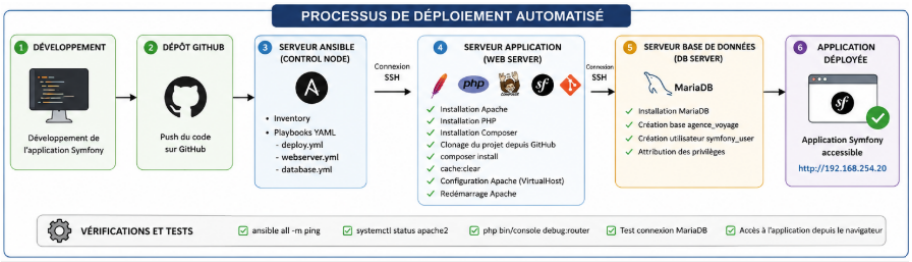

FIGURE 3.3 – Processus de déploiement automatisé

Les principales étapes sont :

1. vérification de la disponibilité des serveurs ;
2. connexion SSH aux machines cibles ;
3. installation des paquets nécessaires ;
4. configuration du serveur web ;
5. configuration de la base de données ;
6. récupération du code source depuis GitHub ;
7. installation des dépendances avec Composer ;
8. configuration de Symfony ;
9. configuration des permissions ;
10. redémarrage d'Apache ;
11. vérification du fonctionnement de l'application ;
12. enregistrement du résultat du déploiement.

3.16 Organisation du projet Ansible

L'organisation du projet Ansible est la suivante :

ansible-project/

inventory

playbooks/

webserver.yml

database.yml

deploy.yml

Cette organisation permet de séparer les différentes responsabilités.

- `inventory` : contient la liste des serveurs cibles ;
- `webserver.yml` : automatise la configuration du serveur web ;
- `database.yml` : automatise la configuration de MariaDB ;
- `deploy.yml` : automatise le déploiement de l'application.

3.17 Inventaire Ansible

L'inventaire permet à Ansible de connaître les machines qu'il doit administrer.

L'inventaire utilisé dans le projet peut être représenté comme suit :

```
1 [webservers]
2 app1 ansible_host=192.168.254.20
3
4 [databases]
5 db1 ansible_host=192.168.254.30
```

Les serveurs sont ainsi regroupés selon leur rôle.

Le groupe `webservers` correspond au serveur d'application tandis que le groupe `databases` correspond au serveur MariaDB.

3.18 Conception des playbooks

Afin de faciliter la maintenance et l'organisation du projet, les tâches ont été réparties dans plusieurs playbooks.

3.18.1 Playbook webserver.yml

Le playbook `webserver.yml` est responsable de la configuration du serveur d'application.

Il permet notamment :

- d'installer Apache ;
- d'installer PHP ;
- d'installer les extensions PHP nécessaires ;
- de démarrer le service Apache ;
- de configurer l'environnement nécessaire à Symfony.

3.18.2 Playbook database.yml

Le playbook `database.yml` est destiné au serveur de base de données.

Il permet :

- d'installer MariaDB ;
- d'installer les dépendances nécessaires à Ansible ;
- de créer la base de données ;
- de créer l'utilisateur de base de données ;
- d'attribuer les privilèges nécessaires.

3.18.3 Playbook deploy.yml

Le playbook `deploy.yml` constitue le cœur du processus de déploiement de l'application.

Il permet notamment :

- de récupérer le projet depuis GitHub ;
- de préparer le répertoire de l'application ;
- d'installer les dépendances avec Composer ;
- de configurer l'application Symfony ;
- de définir les permissions ;
- de configurer Apache ;
- de redémarrer le serveur web ;
- de vérifier le résultat du déploiement.

3.19 Cycle de fonctionnement de la solution

Le fonctionnement global de la solution peut être résumé en plusieurs phases.

3.19.1 Phase 1 : Préparation

Les machines virtuelles sont créées et configurées dans VMware.

Le réseau est ensuite configuré afin de permettre la communication entre les différentes machines.

3.19.2 Phase 2 : Configuration

Le serveur Ansible est configuré avec l'inventaire et les clés SSH.

Les playbooks nécessaires à l'installation des différents services sont ensuite préparés.

3.19.3 Phase 3 : Déploiement

Le playbook principal est exécuté depuis le serveur Ansible.

Ansible récupère le code source, installe les dépendances et configure l'application.

3.19.4 Phase 4 : Validation

Une fois le déploiement terminé, différents tests sont effectués afin de vérifier :

— la disponibilité des serveurs ;
— le fonctionnement d'Apache ;
— l'accès à l'application Symfony ;
— la communication avec MariaDB.

3.20 Flux de fonctionnement de l'application

Après le déploiement, le traitement d'une requête utilisateur suit plusieurs étapes.

1. l'utilisateur envoie une requête HTTP ;
2. Apache reçoit la requête ;
3. Apache transmet la requête à l'application Symfony ;
4. Symfony traite la requête ;
5. Symfony interroge MariaDB si des données sont nécessaires ;
6. MariaDB retourne les données demandées ;
7. Symfony génère la réponse ;
8. Apache retourne la réponse au navigateur.

La figure suivante présente l'architecture interne du traitement d'une requête Symfony.

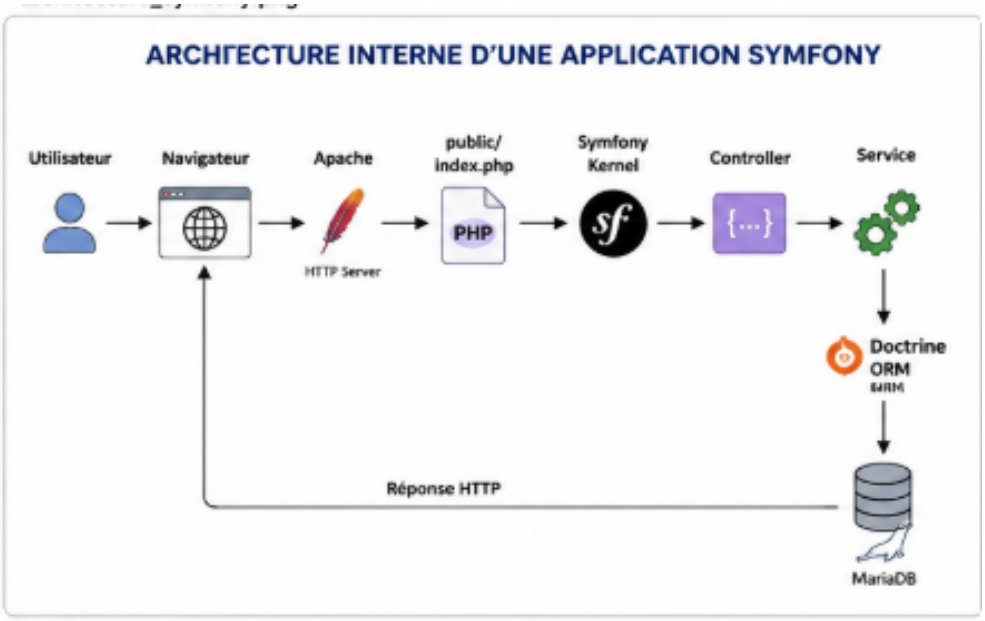

FIGURE 3.4 – Architecture interne du traitement d'une requête Symfony

3.21 Chronologie du déploiement

Le déploiement automatisé suit une chronologie précise.

1. lancement du playbook depuis le serveur Ansible ;
2. vérification de la connexion aux serveurs ;
3. préparation du serveur d'application ;
4. récupération du projet depuis GitHub ;
5. installation des dépendances Composer ;
6. configuration de Symfony ;
7. configuration du VirtualHost Apache ;
8. application des permissions ;
9. redémarrage d'Apache ;
10. vérification de l'application ;
11. enregistrement du résultat du déploiement.

3.22 Conclusion

Dans ce chapitre, nous avons présenté le contexte général du projet, l'organisme d'accueil ainsi que les besoins auxquels la solution doit répondre.

L'analyse de l'existant a permis d'identifier les principales limites du déploiement manuel, notamment le temps nécessaire, le risque d'erreur humaine et la difficulté de reproduire une configuration identique.

Une solution basée sur VMware Workstation et Ansible a ainsi été conçue. Elle repose sur un serveur de contrôle Ansible et deux serveurs cibles destinés respectivement à l'hébergement de l'application Symfony et à la gestion de la base de données MariaDB.

La conception présentée dans ce chapitre constitue la base de la réalisation pratique. Le chapitre suivant sera consacré à la mise en oeuvre de cette solution, à la configuration des machines virtuelles, au développement des playbooks Ansible et aux tests du déploiement automatisé.

4 Réalisation, tests et validation de la solution

4.1 Introduction

Après avoir présenté l'étude et la conception de la solution, ce chapitre présente la mise en œuvre pratique de l'environnement ainsi que la réalisation du déploiement automatisé de l'application Symfony.

Il présente également les différentes améliorations apportées au playbook Ansible, notamment la journalisation, la gestion des erreurs, le rollback automatique, les notifications par e-mail et la configuration des pages d'erreur Apache.

Enfin, des tests sont réalisés afin de vérifier le bon fonctionnement et la fiabilité de la solution développée.

4.2 Mise en œuvre de l'environnement

4.2.1 Création et configuration des machines virtuelles

Trois machines virtuelles Ubuntu ont été créées avec VMware Workstation :

- une machine de contrôle Ansible ;
- un serveur destiné à héberger l'application Symfony ;
- un serveur dédié à la base de données MariaDB.

Les machines ont été configurées afin de pouvoir communiquer entre elles sur le réseau virtuel VMware.

4.2.2 Configuration du réseau

Les machines virtuelles ont été connectées au réseau VMnet8 en mode NAT.

Les adresses IP utilisées sont :

Machine	Adresse IP	Rôle
Ansible	192.168.254.10	Contrôle
APP	192.168.254.20	Application Web
DB	192.168.254.30	Base de données

TABLE 4.1 – Configuration réseau des machines virtuelles

4.3 Configuration d'Ansible

Ansible a été installé sur la machine de contrôle.

Une authentification SSH par clé publique a ensuite été configurée afin de permettre à Ansible d'administrer les serveurs distants sans intervention manuelle.

La communication entre les différentes machines a été vérifiée avec la commande :

```
ansible all -m ping
```

Le résultat obtenu a confirmé que les serveurs distants étaient accessibles depuis le nœud de contrôle.

4.4 Déploiement automatisé de l'application

Le déploiement de l'application Symfony a été automatisé à l'aide du playbook `deploy.yml`.

Les principales opérations automatisées sont :

1. mise à jour du système ;
2. installation des dépendances nécessaires ;
3. installation et configuration d'Apache et PHP ;
4. récupération du projet depuis GitHub ;
5. création et configuration du fichier `.env` ;
6. installation des dépendances avec Composer ;
7. configuration des permissions ;
8. nettoyage et génération du cache Symfony ;
9. configuration du serveur Apache ;
10. redémarrage des services ;
11. vérification de l'accessibilité de l'application.

4.5 Améliorations du processus de déploiement

Afin de rendre la solution plus fiable et plus facile à superviser, plusieurs fonctionnalités supplémentaires ont été intégrées au playbook.

4.5.1 Journalisation du déploiement

Deux fichiers de journalisation ont été mis en place :

— `deployment.log` pour enregistrer le déroulement du déploiement ;
— `error.log` pour enregistrer les erreurs rencontrées.

Ces fichiers permettent de conserver un historique des opérations et facilitent le diagnostic en cas de problème.

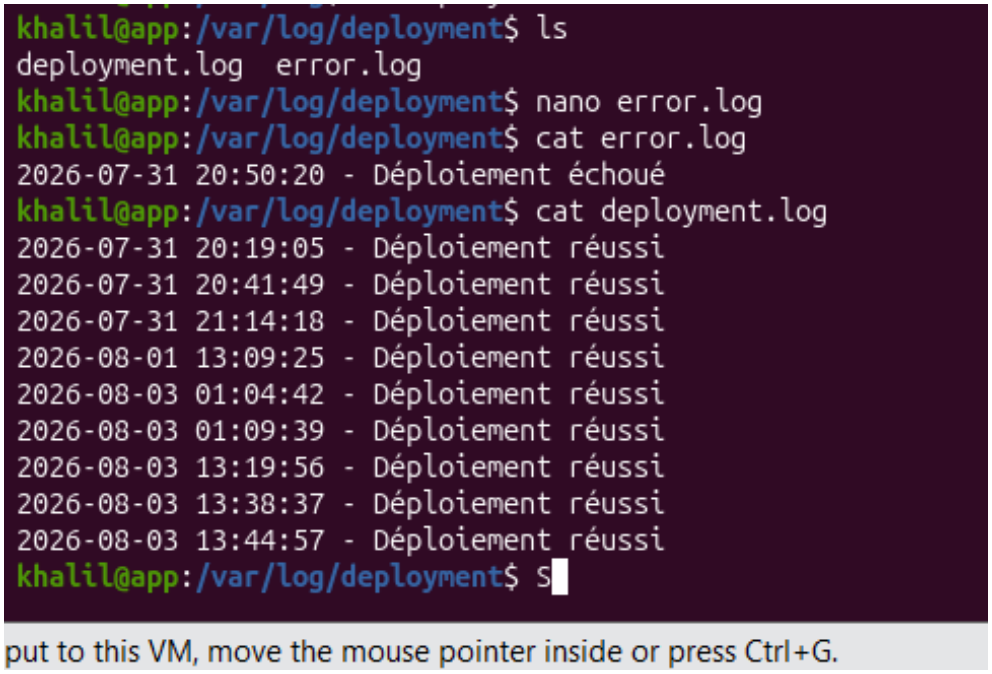

FIGURE 4.1 – Journal du déploiement

4.5.2 Gestion des erreurs avec block, rescue et always

Une gestion structurée des erreurs a été intégrée au playbook à l'aide des blocs `block`, `rescue` et `always`.

Le bloc `block` regroupe les opérations principales du déploiement. En cas d'échec, le bloc `rescue` permet d'exécuter les opérations prévues pour gérer l'erreur. Le bloc `always` permet quant à lui d'exécuter certaines opérations finales indépendamment du résultat.

Cette organisation permet d'améliorer la robustesse du processus de déploiement.

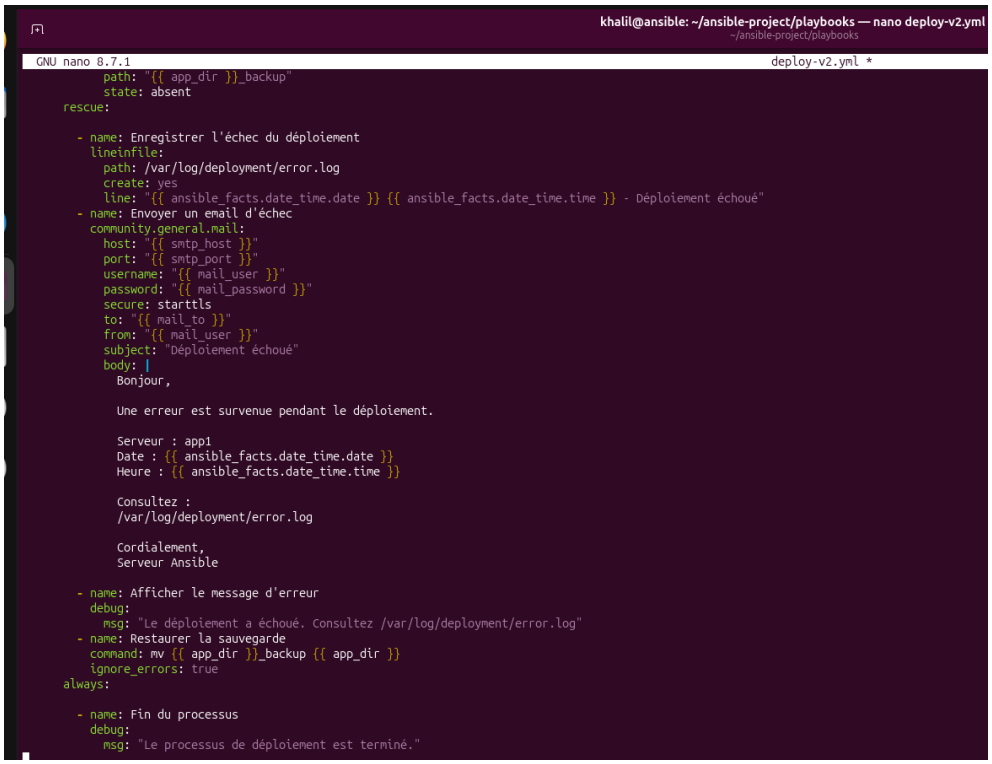

FIGURE 4.2 – Gestion des erreurs avec block, rescue et always

4.5.3 Rollback automatique

Un mécanisme de rollback automatique a été ajouté afin de permettre une restauration de l'état précédent de l'application lorsqu'une erreur critique survient durant le déploiement.

Cette fonctionnalité permet de limiter les interruptions de service et de revenir à un état fonctionnel de l'application.

```
        Cordialement,
        Serveur Ansible

  - name: Afficher le message d'erreur
    debug:
      msg: "Le déploiement a échoué. Consultez /var/log/deployment/error.log"
  - name: Restaurer la sauvegarde
    command: mv {{ app_dir }}_backup {{ app_dir }}
    ignore_errors: true
always:

  - name: Fin du processus
    debug:
      msg: "Le processus de déploiement est terminé."
```

FIGURE 4.3 – Mécanisme de rollback automatique

4.5.4 Notifications par e-mail

Un système de notifications par e-mail a également été intégré au processus de déploiement.

Une notification permet d'informer l'administrateur de l'état du déploiement, notamment en cas de réussite ou d'échec.

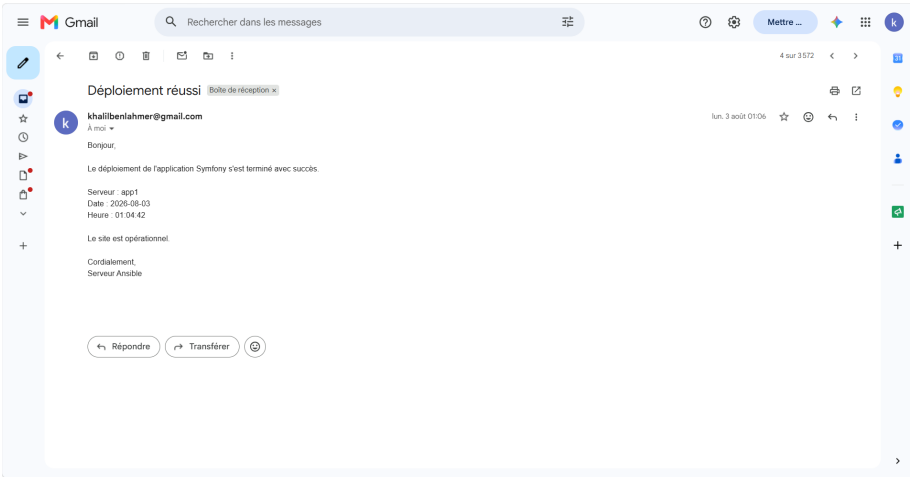

FIGURE 4.4 – Notification e-mail du déploiement

4.5.5 Pages d'erreur personnalisées

Afin d'améliorer la gestion des erreurs côté serveur, des pages personnalisées ont été configurées dans Apache.

Les erreurs prises en charge sont :

- erreur 404 : page non trouvée ;
- erreur 500 : erreur interne du serveur ;
- erreur 503 : service temporairement indisponible.

Cette configuration permet d'afficher des pages adaptées lorsqu'une erreur survient lors de l'accès à l'application.

```
khalil@app:/var/www/html$ cd errors
khalil@app:/var/www/html/errors$ ls
404.html  500.html  503.html
khalil@app:/var/www/html/errors$ cat 404.html
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<title>Page introuvable</title>
<style>
body{
    background:#f4f4f4;
    font-family:Arial,sans-serif;
    text-align:center;
    padding-top:100px;
}
.container{
    width:700px;
    margin:auto;
}
h1{
    font-size:60px;
    color:#d9534f;
}
p{
    font-size:20px;
    color:#555;
}
a{
    text-decoration:none;
    color:#007BFF;
}
</style>
</head>
<body>

<div class="container">

<h1>404</h1>

<h2>Application introuvable</h2>

<p>
```

FIGURE 4.5 – Pages d'erreur personnalisées

4.6 Tests et validation

Après la mise en œuvre de la solution, plusieurs tests ont été réalisés afin de vérifier le bon fonctionnement du processus automatisé.

4.6.1 Test du déploiement

Le playbook a été exécuté depuis le serveur de contrôle Ansible.

Le résultat de l'exécution a permis de vérifier que les différentes tâches ont été réalisées correctement et que le serveur d'application a été configuré automatiquement.

4.6.2 Vérification de l'application

Après le déploiement, l'application Symfony a été accessible depuis un navigateur Web.

Cette vérification confirme le bon fonctionnement de l'ensemble des composants nécessaires au fonctionnement de l'application.

4.6.3 Vérification de la journalisation

Les fichiers `deployment.log` et `error.log` ont été consultés afin de vérifier que les opérations et les éventuelles erreurs étaient correctement enregistrées.

4.6.4 Test de la gestion des erreurs et du rollback

Des tests ont été effectués afin de vérifier le comportement du playbook lorsqu'une erreur survient durant le déploiement.

Le mécanisme de gestion des erreurs permet d'exécuter les opérations prévues dans le bloc `rescue`, tandis que le mécanisme de rollback permet de restaurer l'état précédent de l'application.

4.6.5 Vérification des notifications

La réception de la notification par e-mail a été vérifiée afin de confirmer que l'administrateur est correctement informé de l'état du déploiement.

4.7 Résultats obtenus

Les tests réalisés ont permis de confirmer les résultats suivants :

- déploiement automatisé de l'application Symfony ;
- réduction des interventions manuelles ;
- configuration reproductible du serveur ;
- journalisation des opérations ;
- gestion structurée des erreurs ;
- possibilité de rollback en cas d'échec ;
- notification de l'administrateur par e-mail ;
- gestion personnalisée des erreurs HTTP ;
- application accessible après le déploiement.

4.8 Conclusion

La réalisation et les tests effectués ont permis de valider la solution d'automatisation proposée. L'utilisation d'Ansible a permis de simplifier le déploiement de l'application Symfony et de rendre le processus plus rapide, fiable et reproductible.

Les mécanismes de journalisation, de gestion des erreurs, de rollback, de notification et de gestion des pages d'erreur ont permis de renforcer la robustesse de la solution et

d'améliorer son suivi.

Conclusion générale

Ce stage réalisé au sein du Centre National de l'Informatique (CNI) m'a permis de découvrir et d'approfondir plusieurs domaines essentiels de l'informatique moderne, notamment le Cloud Computing, la virtualisation, l'administration des systèmes Linux ainsi que l'automatisation des infrastructures.

L'objectif principal de ce projet était de concevoir et mettre en place une solution permettant d'automatiser le déploiement d'une application web Symfony dans un environnement virtualisé.

Pour atteindre cet objectif, une infrastructure composée de plusieurs machines virtuelles a été mise en place avec VMware Workstation. Cette infrastructure comprend un serveur de contrôle Ansible, un serveur d'application hébergeant l'application Symfony ainsi qu'un serveur de base de données MariaDB.

L'utilisation d'Ansible a permis d'automatiser les différentes étapes du déploiement, notamment l'installation des services nécessaires, la configuration des serveurs, la récupération du code source depuis GitHub, l'installation des dépendances avec Composer et la configuration du serveur Apache.

Grâce à cette approche, les opérations qui étaient auparavant réalisées manuellement peuvent désormais être exécutées automatiquement, de manière rapide, fiable et reproductible. Cette automatisation permet également de réduire les erreurs humaines et de faciliter la maintenance de l'infrastructure.

Sur le plan technique, ce projet m'a permis d'améliorer mes compétences dans plusieurs domaines :

- la virtualisation avec VMware Workstation ;
- l'administration des systèmes Linux Ubuntu ;
- la configuration des services Apache, PHP et MariaDB ;
- l'utilisation d'Ansible pour l'automatisation ;
- la gestion du code source avec Git et GitHub ;
- le déploiement d'applications web Symfony.

Au-delà des aspects techniques, ce stage m'a également permis de développer des compétences importantes telles que l'autonomie, l'analyse des problèmes, la recherche de solutions et l'organisation du travail dans un contexte professionnel.

Comme perspectives d'amélioration, plusieurs évolutions peuvent être envisagées afin de rendre la solution encore plus complète :

- l'intégration de Docker afin de conteneuriser l'application ;

— l'utilisation de Kubernetes pour l'orchestration des conteneurs ;
— la mise en place d'une chaîne CI/CD avec Jenkins ou GitHub Actions ;
— l'ajout d'outils de supervision comme Grafana et Prometheus ;
— l'utilisation d'une plateforme Cloud publique ou privée pour un déploiement en production.

En conclusion, ce projet a permis de démontrer l'importance de l'automatisation dans la gestion des infrastructures modernes. L'association de la virtualisation VMware et de l'outil Ansible constitue une solution efficace pour simplifier le déploiement des applications web et préparer les environnements aux pratiques DevOps actuelles.

Bibliographie

[1] T. Erl, *Cloud Computing : Concepts, Technology and Architecture*. Prentice Hall, 2013.

[2] J. Keating, *Ansible for DevOps*. Leanpub, 2016.

[3] M. Portnoy, *Virtualization Essentials*. Sybex, 2012.

[4] "Ansible documentation," https://docs.ansible.com, 2026.

[5] "Symfony documentation," https://symfony.com/doc, 2026.