Conception et automatisation d'une infrastructure hospitalière sécurisée en approche DevOps

Sommaire

INTRODUCTION

Mon parcours professionnel est atypique, marqué par une reconversion profonde et pleinement assumée. Après une première carrière au Mexique, j'ai décidé de bâtir un nouveau projet en France, plus aligné avec mes aspirations personnelles et professionnelles. Accompagné par un conseiller en évolution professionnelle, j'ai longuement hésité entre le droit — que j'avais étudié — et l'informatique, un domaine que je découvrais et qui m'a rapidement captivé par sa logique, ses défis et ses perspectives.

C'est en participant à des cours proposés par la mairie, puis en suivant des formations en ligne en autonomie (OpenClassrooms, documentation technique, tutoriels, TP personnels), que j'ai découvert les bases de l'informatique : comprendre ce qu'est un BIOS, une adresse IP, apprendre à utiliser les premières commandes Linux, installer un système, résoudre des problèmes simples. Petit à petit, chaque nouvelle notion était une victoire. Cette progression a renforcé ma motivation et m'a convaincu de faire de l'informatique plus qu'un choix de reconversion : un projet de vie durable.

J'ai alors suivi un parcours structuré de formation professionnelle :
• Technicien d'Assistance Informatique (TAI) : les bases du support et de la maintenance
• Technicien Supérieur Systèmes et Réseaux (TSSR) : l'administration réseau, les services Windows/Linux, la sécurité
• Et aujourd'hui, Administrateur Systèmes DevOps chez M2i Formation, où j'approfondis l'automatisation, le cloud et l'orchestration.

Au fil de ce chemin, j'ai acquis des compétences concrètes et transversales :
• Configuration de réseaux (VLAN, DHCP, DNS)
• Administration de serveurs Linux et Windows (Active Directory, scripts Bash, PowerShell)
• Virtualisation avec Proxmox, Hyper-V, VirtualBox
• Sécurisation avec pfSense, segmentation réseau, gestion des accès
• Automatisation avec Ansible, déploiement CI/CD avec Jenkins et GitHub Actions
• Conteneurisation avec Docker, orchestration avec Kubernetes
• Supervision avec Prometheus et Grafana
• Et exploration des environnements Cloud Azure et Terraform

Pour structurer ce **projet professionnel**, j'ai choisi de présenter les compétences acquises en deux grandes parties : d'abord l'administration systèmes, puis l'approche DevOps. Ce découpage n'a pas pour but d'opposer ces deux dimensions, mais au contraire de montrer leur complémentarité.

L'administration systèmes représente la base solide sur laquelle toute infrastructure repose : configuration des réseaux, gestion des serveurs, sécurité, segmentation… Ces fondations sont essentielles pour garantir stabilité et fiabilité.

L'approche DevOps, quant à elle, vient enrichir cette base en apportant des outils d'automatisation, de collaboration, d'intégration et de déploiement continu. Grâce à cette approche, les équipes systèmes et développement travaillent ensemble, dans une logique fluide et agile.

Cette division est donc pédagogique : elle me permet d'exposer les étapes de construction de **mon projet professionnel d'infrastructure hospitalière**, tout en montrant comment les compétences d'un administrateur moderne doivent s'étendre aux pratiques DevOps pour répondre aux besoins réels du terrain.

Je tiens à remercier les équipes pédagogiques de EPIE Formation, Laser Formation, M2i Formation, ainsi que toutes les personnes — formateurs, conseillers, proches — qui ont contribué, par leur accompagnement ou leur confiance, à transformer ma reconversion en un projet concret et ambitieux.

PARTIE 1 – ADMINISTRATION SYSTÈMES

La première étape de la conception d'une infrastructure hospitalière consiste à bâtir une base solide, stable et sécurisée. Cela correspond aux missions fondamentales d'un administrateur systèmes : planifier, déployer, configurer et tester une architecture réseau cohérente, capable de répondre aux besoins de l'organisation.

Dans cette partie, j'ai simulé un service informatique hospitalier complet, intégrant un pare-feu pfSense, une segmentation réseau stricte, un serveur Windows Server 2022 assurant les rôles essentiels (Active Directory, DNS, DHCP), ainsi que l'intégration de postes clients au domaine.

Des tests fonctionnels et de sécurité ont été menés pour valider le bon fonctionnement de l'infrastructure, garantir l'isolation entre les réseaux, la gestion centralisée des utilisateurs, et la disponibilité des services.

L'objectif est de reproduire fidèlement un environnement professionnel, structuré comme on pourrait le retrouver dans un hôpital réel : sécurisation des flux sensibles, gouvernance des comptes, stabilité des services essentiels.

Cette base servira de fondation technique pour les étapes d'automatisation, de supervision et de migration cloud, qui seront abordées dans la Partie 2 dédiée à l'approche DevOps.

a) 1.1 Organisation du Service IT

L'infrastructure de l'hôpital repose sur un service informatique interne structuré, capable de répondre aux besoins quotidiens des utilisateurs tout en préparant la transition vers une automatisation avancée et une éventuelle migration vers le cloud. Cette organisation repose sur une répartition claire des rôles et des responsabilités.

Objectifs du Service IT :

- Assurer la disponibilité, la sécurité et la maintenance des systèmes et réseaux
- Gérer les utilisateurs, les postes de travail, les serveurs et les services
- Déployer progressivement des solutions d'automatisation, de supervision et de virtualisation

- Préparer l'ouverture vers des environnements cloud ou hybrides à long terme

Tableau récapitulatif des rôles :

Rôle	Responsabilités	Technologies utilisées
Responsable IT (DSI / CIO)	Supervision globale du SI, stratégie d'évolution, gestion des ressources, sécurité	ITIL, DevOps, gestion de projet, outils de supervision
Administrateur Systèmes & Réseaux	Administration des serveurs, gestion des réseaux, pare-feu, Active Directory	Windows Server, Linux (Debian), pfSense, Proxmox, VMware
Technicien Support IT	Assistance aux utilisateurs, installation et maintenance des postes, gestion des tickets	Windows 10/11, Linux, outils Helpdesk, Active Directory
Développeur Web & Logiciels	Développement et maintenance des applications internes (prise de RDV, accès patient)	PHP, Python, Node.js, API REST, MariaDB
Responsable Sécurité (RSSI)	Protection du réseau, surveillance, mise en place de politiques de sécurité	pfSense, Suricata, SIEM, Zabbix, antivirus, Pentest
Ingénieur DevOps / Cloud	(Ajouté à terme) Automatisation des déploiements, gestion du cloud et du CI/CD	Ansible, Terraform, Jenkins, Docker, Kubernetes, Azure, AWS

Remarque :
Dans le cadre de ce projet, certains rôles sont regroupés ou simulés afin de démontrer l'ensemble des compétences techniques attendues, en particulier celles de l'administrateur systèmes et de l'ingénieur DevOps.

b) 1.2 Installation et configuration de pfSense

L'objectif de cette étape est d'installer et configurer **pfSense**, un pare-feu open source, pour sécuriser l'infrastructure hospitalière simulée. Il permettra de :

- Segmenter les réseaux internes (admin, patient, médical, DMZ, sécurité)
- Protéger les flux sensibles
- Fournir des services réseau essentiels (DHCP, pare-feu, NAT, etc.)
- Centraliser la gestion du trafic

Définition des réseaux

Réseau	Plage IP	Utilisation
WAN	NAT (VirtualBox)	Accès à Internet (mises à jour/tests)
LAN (Admin)	192.168.1.0/26	Administration, serveurs, AD
Patient	192.168.1.64/26	Postes patients
Médical	192.168.1.128/26	Soignants / dossiers médicaux
DMZ	192.168.1.192/27	Serveurs web accessibles
Sécurité	192.168.1.224/27	Supervision, sauvegardes, firewall

Création de la machine virtuelle pfSense

Paramètres recommandés :

- **RAM** : 2048 Mo
- **CPU** : 2 vCPU
- **Disque** : 10 Go (ZFS)
- **Cartes réseau** :
    - em0 → WAN (NAT)
    - em1 → LAN (192.168.1.1/26)
    - em2 → DMZ
    - em3 → Médical
    - em4 → Patient
    - em5 → Sécurité

Installation de pfSense

1. Démarrer la VM depuis VirtualBox
2. Sélectionner **Install pfSense**
3. Clavier : **French (France)**
4. Type de partition : **ZFS recommandé**
5. Laisser l'installation se terminer
6. Retirer l'ISO et redémarrer

Configuration des interfaces réseau

Depuis la console de pfSense, affecter les interfaces comme suit :

- em0 → WAN

- em1 → LAN → 192.168.1.1/26
- em2 → DMZ → 192.168.1.193/27
- em3 → Médical → 192.168.1.129/26
- em4 → Patient → 192.168.1.65/26
- em5 → Sécurité → 192.168.1.225/27

Accès à l'interface Web & activation SSH

- Accès Web : https://192.168.1.1
- Identifiants initiaux : admin / pfsense
- Changer le mot de passe dès la première connexion
- Activer l'accès SSH :
    - Modifier /etc/rc.conf et ajouter : sshd_enable="YES"
    - Démarrer le service : service sshd start

Configuration DHCP et pare-feu

DHCP :

- Activer le DHCP sur chaque interface souhaitée (ex : LAN)
- Plage exemple : 192.168.1.10 → 192.168.1.50
- Passerelle : 192.168.1.1

Pare-feu :

- Définir des règles spécifiques par interface
- Exemple :
    - Autoriser l'accès du réseau Médical vers AD
    - Bloquer les connexions directes entre Patient et DMZ

Résultats attendus

- pfSense opérationnel
- Interfaces correctement assignées et fonctionnelles
- Accès Web sécurisé et SSH actif
- DHCP attribue des adresses IP sur les bons réseaux
- Les règles de pare-feu respectent la segmentation définie

c) 1.3 Définition des réseaux (LAN, DMZ, Médical, Patient, Sécurité)

Dans une infrastructure hospitalière, la segmentation réseau est essentielle pour garantir la sécurité, la performance et la clarté de l'organisation du trafic. Chaque réseau a un usage bien défini, avec des droits d'accès, des règles de communication et des priorités spécifiques.

Objectifs de la segmentation réseau

- Isoler les flux critiques (ex. : données médicales) des flux utilisateurs (ex. : patients)
- Sécuriser les services exposés à Internet (DMZ)
- Protéger les serveurs internes via un réseau d'administration (LAN)
- Réduire la surface d'attaque globale
- Permettre une gestion fine des règles pare-feu avec pfSense

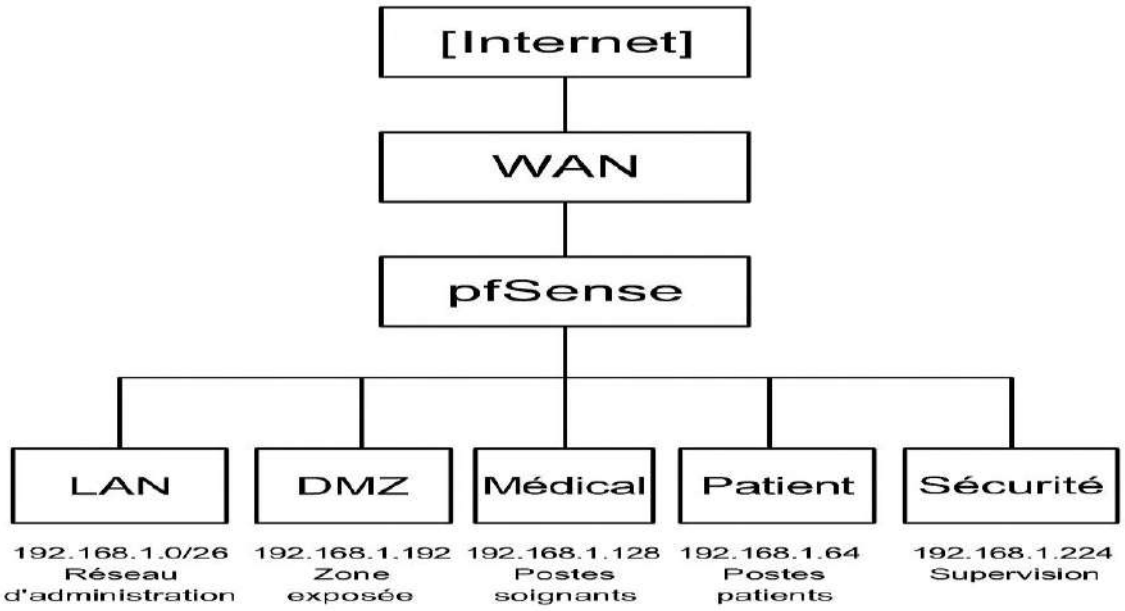
Réseaux définis dans pfSense

Nom du réseau	Plage IP	Utilisation	Exemples de services / équipements
LAN (Admin)	192.168.1.0/26	Administration, services internes	Serveur AD, DNS, DHCP, Ansible, Jenkins, SSH
Patient	192.168.1.64/26	Accès Internet uniquement pour les patients	Postes patients, portail captif, navigateur
Médical	192.168.1.128/26	Applications et postes médicaux	Dossiers patients, postes soignants, ERP
DMZ	192.168.1.192/27	Services publics accessibles depuis Internet	Site Web, interface RDV, API externe
Sécurité	192.168.1.224/27	Supervision et sécurité	Zabbix, Prometheus, Grafana, pfSense, backups

Règles de sécurité appliquées entre les réseaux

de	Vers	Accès autorisé ?	Justification
Médical	LAN (AD/DNS)	✅ Oui	Accès aux services d'authentification et DNS
Patient	Médical	❌ Non	Isolation stricte pour protéger les données
Patient	Internet (WAN)	✅ Oui	Navigation web autorisée
DMZ	LAN	❌ Non (sauf exceptions)	Isolation du réseau interne
LAN	Tous les réseaux	✅ Oui (admin only)	Accès complet pour gestion et supervision
Sécurité	Tous les réseaux	✅ Oui (lecture seule)	Monitoring et collecte de logs centralisée

Plan logique de l'infrastructure



Tous les réseaux sont reliés à pfSense, qui joue le rôle de routeur, pare-feu et serveur DHCP. Il assure le filtrage des flux selon les règles de sécurité définies.

Résultats attendus

- Chaque réseau est correctement identifié, segmenté et isolé
- Les flux sont autorisés ou bloqués selon les besoins réels
- La base réseau est sécurisée, stable, et prête à accueillir les services applicatifs

d) 1.4 Installation de Windows Server 2022 (AD, DNS, DHCP)

Objectif

Déployer un serveur Windows Server 2022 assurant trois rôles clés :

- **Active Directory (AD)** pour la gestion centralisée des utilisateurs,
- **DNS** pour la résolution des noms internes,
- **DHCP** pour l'attribution automatique des adresses IP sur le réseau LAN.

Ces services assurent la cohérence, la sécurité et l'évolutivité de l'infrastructure réseau hospitalière.

Configuration de la machine virtuelle

Ressource	Valeur
Système	Windows Server 2022 Standard (interface graphique)
IP statique	192.168.1.10
Passerelle	192.168.1.1 (pfSense)
DNS	192.168.1.10 (auto-référence)
Nom d'hôte	WS2022-HOSPITAL

Installation des rôles (AD, DNS, DHCP)

Depuis **PowerShell en tant qu'administrateur**, installer les rôles suivants :

```
powershell
```

```
Install-WindowsFeature -Name AD-Domain-Services, DNS, DHCP -
IncludeManagementTools
```

Une fois les rôles installés, ils apparaissent dans le **Server Manager**.

Promotion en contrôleur de domaine

Création du domaine : **hopital.local**

```
powershell
```

```
Install-ADDSForest `
  -DomainName "hopital.local" `
  -InstallDNS `
  -NoRebootOnCompletion:$false `
  -Force:$true
```

Le serveur redémarre automatiquement après la promotion.

Vérification du bon fonctionnement

Après redémarrage, lancer les commandes suivantes :

```
powershell
```

```
Get-ADDomainController -Discover
```

```
Get-Service | Where-Object { $_.Name -match "ADWS|DNS|NETLOGON" }
```

```
Get-DnsServerZone
```

Le domaine doit être actif et les services essentiels démarrés.

Création d'un utilisateur dans Active Directory

Création d'un utilisateur **Jean Dupont** :

```
powershell
```

```
New-ADUser -Name "Jean Dupont" `
  -GivenName "Jean" `
  -Surname "Dupont" `
  -SamAccountName "jdupont" `
  -UserPrincipalName "jdupont@hopital.local" `
  -Path "CN=Users,DC=hopital,DC=local" `
  -AccountPassword (ConvertTo-SecureString "Password123!" -AsPlainText -Force) `
  -Enabled $true
```

Vérification de l'utilisateur :

```
powershell
Get-ADUser -Filter "SamAccountName -eq 'jdupont'"
```

Configuration du serveur DHCP

Création d'une **étendue DHCP** pour le LAN :

```
powershell
Add-DhcpServerv4Scope `
  -Name "Scope-LAN" `
  -StartRange 192.168.1.20 `
  -EndRange 192.168.1.50 `
  -SubnetMask 255.255.255.192 `
  -State Active
```

Définition des options réseau :

```
powershell
Set-DhcpServerv4OptionValue -ScopeId 192.168.1.0 -OptionId 3 -Value 192.168.1.1    # Passerelle
Set-DhcpServerv4OptionValue -ScopeId 192.168.1.0 -OptionId 6 -Value 192.168.1.10   # DNS
```

Tests depuis un poste client

1. Démarrer une VM cliente (Windows 10 ou 11)
2. Configurer la carte réseau en **obtenir une adresse IP automatiquement**
3. Lancer PowerShell et exécuter les commandes suivantes :

```
powershell
ipconfig /release
ipconfig /renew
nslookup hopital.local
```

Résultats attendus :

- Le poste reçoit une IP du serveur DHCP
- Le nom de domaine est résolu via le DNS local
- L'utilisateur peut être ajouté au domaine

Résumé des résultats

- Le serveur est promu en **contrôleur de domaine**
- Les services **AD, DNS, DHCP** sont actifs
- Un utilisateur a été créé avec succès
- Les postes clients reçoivent une configuration réseau fonctionnelle automatiquement

e) 1.5 Intégration des postes clients au domaine

L'intégration des postes clients dans le domaine Active Directory **hopital.local** permet une gestion centralisée des comptes, des autorisations, des stratégies de sécurité (GPO), et des ressources partagées comme les imprimantes ou les dossiers réseau. Cette opération est un pilier de toute infrastructure professionnelle bien administrée.

Objectifs

- Intégrer les postes clients (Windows 10/11) au domaine Active Directory
- Vérifier la communication avec le serveur AD/DNS
- Tester l'attribution automatique des adresses IP via DHCP
- Authentifier un utilisateur du domaine sur le poste client

Configuration réseau du poste client

Élément	Valeur
Type de machine	Windows 10 ou Windows 11
Adresse IP	Attribuée automatiquement (DHCP)
DNS principal	192.168.1.10 (Serveur AD)
Passerelle	192.168.1.1 (pfSense)

À vérifier via PowerShell sur le poste client :

```
powershell
ipconfig /all
ping 192.168.1.10
nslookup hopital.local
```

Rejoindre le domaine Active Directory

Méthode graphique :

1. Ouvrir **Paramètres > Système > Informations système**
2. Cliquer sur **"Renommer ce PC (avancé)"**
3. Choisir **"Domaine"** et saisir : hopital.local
4. Entrer les identifiants AD : Administrateur / mot de passe
5. Redémarrer le poste

OU via PowerShell :

```
powershell
Add-Computer -DomainName "hopital.local" -Credential hopital\Administrateur
-Restart
```

Après redémarrage, le poste est intégré au domaine.

Connexion d'un utilisateur du domaine

Depuis l'écran de connexion Windows :

- Cliquer sur **"Autre utilisateur"**
- Entrer :
    - Nom d'utilisateur : hopital\jdupont
    - Mot de passe : Password123!

L’utilisateur est authentifié sur le domaine avec succès.

Vérifications depuis le serveur AD

Sur le serveur Windows Server 2022 :

```
powershell
```

```
Get-ADComputer -Filter "Name -eq 'NomDuPosteClient'"
```

```
Get-EventLog -LogName Security -Newest 10
```

Le poste client apparaît dans l’Active Directory et ses connexions sont tracées dans les journaux de sécurité.

Résultats obtenus

Étape testée	Statut
Réception d’une IP via DHCP	✅
Résolution DNS vers le domaine hopital.local	✅
Intégration du poste via GUI ou PowerShell	✅
Connexion utilisateur du domaine	✅
Visibilité du poste dans AD	✅

f) 1.6 Tests d’accès et de sécurité inter-réseaux

Une infrastructure hospitalière gère des flux sensibles : données médicales, connexions administratives, accès patients… Il est donc essentiel de tester la communication entre les réseaux afin de s’assurer que **seules les connexions légitimes sont autorisées** et que chaque segment est isolé selon ses besoins.

Objectifs des tests

- Vérifier la bonne **isolation entre les réseaux** (Médical, Patient, DMZ, LAN, Sécurité)
- Valider le bon fonctionnement des **règles de pare-feu pfSense**

- Tester les accès autorisés (ex : Médical vers AD) et interdire les accès non légitimes (ex : Patient vers Médical)
- Préparer l'environnement à l'audit de sécurité ou à l'intégration future d'un IDS/IPS

Réseaux concernés et cas de test

Source	Destination	Comportement attendu	Justification
Patient	Internet (WAN)	✅ Autorisé	Accès Web autorisé pour les patients
Patient	Médical	❌ Bloqué	Protection des données médicales
Médical	LAN (AD/DNS)	✅ Autorisé	Accès aux services centraux
Médical	Patient	❌ Bloqué	Confidentialité et protection mutuelle
DMZ	LAN	❌ Bloqué	Isolement strict des services exposés
LAN	Tous réseaux	✅ Autorisé	Administration globale
Sécurité	Tous réseaux	✅ Lecture uniquement	Supervision passive (Zabbix, Prometheus…)

Méthodologie de test

Outils utilisés :

- ping pour tester la connectivité IP
- tracert (Windows) ou traceroute (Linux) pour suivre le chemin réseau
- telnet ou nmap pour tester les ports
- Navigateur Web ou curl pour tester HTTP
- Connexions RDP ou SSH si activées

Exemples de tests :

Depuis un poste dans le réseau Patient :

```
bash
ping 192.168.1.10            # vers AD (doit échouer)
ping 8.8.8.8                 # vers Internet (réussi)
curl http://site-hopital.dmz  # test d'accès à la DMZ
```

Depuis un poste dans le réseau Médical :

```
bash
ping 192.168.1.10            # vers AD (réussi)
nslookup hopital.local       # test de résolution DNS
ping 192.168.1.65            # vers Patient (doit échouer)
```

Vérification des règles dans pfSense

Depuis l'interface Web :

1. Aller dans **Firewall > Rules**
2. Contrôler les règles sur chaque interface :
    - **Patient** : blocage total sauf Internet
    - **Médical** : accès limité au LAN uniquement
    - **DMZ** : accès entrant restreint
    - **Sécurité** : accès en lecture seule (ex. monitoring)
3. Activer le **logging** pour suivre les tentatives de connexion

Résultats des tests

Test effectué	Résultat	Conforme ?
Patient → Internet	✅ Réussi	✅
Patient → Médical	❌ Bloqué	✅
Médical → AD/DNS	✅ Réussi	✅
Médical → Patient	❌ Bloqué	✅
DMZ → LAN	❌ Bloqué	✅

LAN → Tous réseaux	✅ Réussi	✅
Sécurité → Monitoring réseau	✅ Réussi	✅

Conclusion de la partie Administration Systèmes

Ce projet m'a permis de consolider l'ensemble des compétences fondamentales attendues d'un administrateur systèmes dans un environnement réaliste, structuré et critique.

J'ai appris à penser comme un professionnel de la production IT, en construisant une **architecture réseau claire, segmentée et sécurisée**, tout en assurant la gestion des utilisateurs via un serveur Windows Server bien configuré.

Points clés maîtrisés

- Installation et configuration de **pfSense** avec plusieurs interfaces et règles personnalisées
- Mise en place d'un serveur Windows Server 2022 avec **Active Directory, DNS, DHCP**
- Conception d'une **infrastructure logique, sécurisée et documentée**
- Réalisation de **tests inter-réseaux** pour valider l'efficacité des règles de sécurité

Réflexion personnelle

Ce travail sur l'infrastructure hospitalière m'a permis de passer d'une approche théorique à une mise en pratique concrète, complète et cohérente.
J'ai non seulement appliqué des compétences techniques (réseaux, sécurité, services Windows), mais surtout appris à les organiser et à les structurer comme le ferait un professionnel en entreprise.

Chaque étape – de la configuration du pare-feu pfSense à la gestion centralisée des utilisateurs avec Active Directory – m'a confronté à des situations réelles : prise de décision, résolution de problèmes, validation par des tests. C'est dans ce processus que j'ai véritablement progressé.

Je me rends compte que je suis désormais capable de **concevoir, sécuriser et administrer une infrastructure réseau de bout en bout**, en m'appuyant sur des outils professionnels, une logique rigoureuse et une documentation structurée.

Cette première partie représente bien plus qu'un exercice technique : c'est une démonstration concrète que je peux assumer des responsabilités opérationnelles en tant qu'administrateur systèmes. Elle constitue la fondation solide qui me permettra d'aborder la suite avec assurance : **l'automatisation, l'orchestration, le cloud et la culture DevOps**.

.

PARTIE 2 – APPROCHE DEVOPS & AUTOMATISATION

Après avoir mis en place une infrastructure stable, sécurisée et fonctionnelle dans la partie *Administration Systèmes*, l'étape suivante consiste à **automatiser, industrialiser et fiabiliser** les déploiements et la gestion des services. Cette démarche s'inscrit dans la culture **DevOps**, qui vise à rapprocher les équipes systèmes et développement grâce à des outils, des pratiques agiles, et des chaînes de déploiement automatisées (CI/CD).

Dans cette partie, j'ai utilisé plusieurs technologies complémentaires :

- **Ansible** pour l'automatisation de la configuration (Windows et Linux),
- **Terraform** pour définir l'infrastructure sous forme de code (*Infrastructure as Code*),
- **Docker** pour la conteneurisation de services,
- **Jenkins** ou **GitHub Actions** pour l'intégration continue,
- Et **Prometheus/Grafana** pour la supervision des services et des ressources.

Objectifs de cette approche DevOps

- Réduire les tâches répétitives et les erreurs humaines
- Gagner du temps lors des déploiements
- Assurer la reproductibilité des environnements
- Rendre l'infrastructure évolutive, scalable et prête pour le cloud
- Appliquer les principes de l'intégration et du déploiement continus (CI/CD)

Structure de la Partie 2

N°	Titre	Objectif principal
2.1	Configuration d'Ansible pour la gestion de Windows Server	Automatiser l'installation des rôles AD, DNS, DHCP

2.2	Automatisation de la promotion AD et de la création d'utilisateurs	Déployer le domaine hopital.local via playbooks
2.3	Automatisation des mises à jour Windows avec Ansible	Maintenir la sécurité des serveurs
2.4	Introduction à Terraform	Décrire et automatiser l'infrastructure
2.5	Conteneurisation + supervision (Docker, Prometheus, Grafana)	Gérer les services légers et surveiller l'état du système
2.6	Intégration continue avec Jenkins ou GitHub Actions	Déployer automatiquement les applications hospitalières
2.7	Préparation à la migration vers le Cloud (Azure / AWS)	Penser la résilience et la scalabilité à long terme

1) 2.1 Introduction à l'automatisation

Objectifs de cette phase

L'automatisation est un pilier fondamental du métier d'**administrateur systèmes DevOps**. Elle permet de :

- Accélérer les déploiements tout en réduisant les erreurs humaines
- Standardiser la configuration des serveurs et des services
- Rendre l'infrastructure reproductible grâce au concept **Infrastructure as Code (IaC)**
- Garantir la cohérence entre les environnements (développement, test, production)
- Préparer la montée en charge vers le **cloud** et les environnements **conteneurisés**

Dans ce projet hospitalier simulé, j'ai choisi de mettre en œuvre une automatisation complète des tâches critiques à l'aide d'outils modernes utilisés dans le monde DevOps.

Outils utilisés

Outil	Rôle dans le projet

Ansible	Automatiser la configuration de Windows Server (AD, DNS, DHCP)
WinRM	Protocole de communication entre Ansible (Linux) et les machines Windows
Terraform *(à venir)*	Déploiement automatique de l'infrastructure (réseaux, VMs)
Jenkins / GitHub Actions	Intégration continue / déploiement automatisé
Docker	Conteneurisation de services et tests en environnements isolés
Prometheus / Grafana	Supervision des ressources et des services

Environnement de test

L'environnement a été conçu pour simuler un cadre professionnel tout en restant maîtrisé techniquement :

Machine Linux (Debian) – Machine de gestion Ansible

- **IP** : 192.168.1.29
- **Rôle** : machine maître (master)
- **Services** : Ansible, SSH, client WinRM

Machine Windows Server 2022 – Cible à configurer

- **IP** : 192.168.1.10
- **Rôle** : serveur cible à configurer
- **Services à automatiser** : AD DS, DNS, DHCP

Réseau utilisé

- **Plage** : LAN – 192.168.1.0/26
- **Communication** : assurée via pfSense
- **Règles de pare-feu** : port **5985** (WinRM HTTP) autorisé

Résumé de la stratégie d'automatisation

L'approche adoptée repose sur une progression contrôlée et modulaire :

1. **Configurer la connexion Ansible ↔ Windows** (via WinRM)

2. **Créer des playbooks Ansible** modulaires pour chaque rôle (AD, DNS, DHCP…)
3. **Vérifier les déploiements** avec des tests automatisés (Ansible) ou manuels (PowerShell)
4. **Étendre progressivement** la démarche aux outils complémentaires :
    - Docker pour la conteneurisation
    - Terraform pour l'IaC
    - Jenkins/GitHub Actions pour l'intégration continue
    - Prometheus/Grafana pour la supervision

2) 2.2 Configuration d'Ansible (Inventaire, WinRM)

Objectifs de cette étape

Avant de pouvoir automatiser l'installation de rôles et services Windows, il est nécessaire de configurer **Ansible** pour qu'il puisse se connecter à distance à une machine **Windows Server** via **WinRM (Windows Remote Management)**.

Cette étape permet à une machine Linux d'administrer un environnement Windows dans une logique multiplateforme.

Pré-requis

Équipement	Détails
Ansible	Installé sur une VM Linux (ex. : Debian 12)
Windows Server	Windows Server 2022 GUI – IP : 192.168.1.10
Communication	Port **5985** (HTTP WinRM) ouvert dans le pare-feu

Activation de WinRM sur Windows Server

Sur la machine **Windows Server 2022**, ouvrir PowerShell en administrateur et exécuter :

```
powershell
# Activer WinRM
winrm quickconfig

# Autoriser les connexions non chiffrées (test interne uniquement)
```

```
winrm set winrm/config/service '@{AllowUnencrypted="true"}'


# Activer l'authentification par mot de passe
winrm set winrm/config/service/Auth '@{Basic="true"}'


# Ouvrir le port 5985 dans le pare-feu
New-NetFirewallRule -Name "WinRM" -DisplayName "WinRM HTTP" -Protocol TCP -LocalPort 5985 -Action Allow
```

À ce stade, le service WinRM est opérationnel et prêt à recevoir des connexions depuis Ansible.

Création de l'inventaire Ansible

Sur la machine Linux (Debian) avec Ansible installé :

```
bash
sudo nano /etc/ansible/hosts
Ajouter les lignes suivantes :
ini
CopierModifier
[windows]
192.168.1.10


[windows:vars]
ansible_user=Administrateur
ansible_password=D@vid0907
ansible_connection=winrm
ansible_port=5985
ansible_winrm_transport=basic
ansible_winrm_server_cert_validation=ignore
```

⚠️ *Remarque : ce mot de passe doit être protégé. En production, on utiliserait* ***Ansible Vault*** *pour chiffrer ces informations.*

Tester la connexion Ansible → Windows

Depuis la machine Linux :

```
bash
ansible windows -m win_ping
```

Résultat attendu :

```
json
192.168.1.10 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

En cas d'échec :

- Vérifier l'adresse IP cible
- Vérifier l'ouverture du port 5985 (pfSense, pare-feu Windows)
- Vérifier que l'utilisateur a les droits administratifs
- Vérifier que WinRM est actif

Résultats obtenus

Étape	Statut
Activation de WinRM sur Windows	✅ OK
Règle de pare-feu ouverte (port 5985)	✅ OK
Inventaire Ansible correctement configuré	✅ OK
Connexion Ansible → Windows testée avec win_ping	✅ Réussi

3) 2.3 Déploiement automatisé (AD, DNS, DHCP)

Automatiser l'installation des services fondamentaux d'un serveur d'entreprise :

- **Active Directory Domain Services (AD DS)**
- **DNS Server**
- **DHCP Server**

Cette automatisation permet de préparer un **contrôleur de domaine prêt à l'emploi**, sans avoir à effectuer une configuration manuelle fastidieuse et sujette à erreur. Elle s'inscrit dans une logique **Infrastructure as Code** (IaC), garante de reproductibilité.

Outils et technologies utilisés

Élément	Description
Ansible	Outil principal d'automatisation, utilisé depuis une machine Debian
WinRM	Protocole de communication avec la machine Windows cible (192.168.1.10)
Module Ansible	win_feature – pour installer les rôles Windows à distance

Playbook Ansible : install_windows_server.yaml

```
yaml
---
- name: Installation des rôles AD, DNS et DHCP
  hosts: windows
  tasks:

    - name: Installer le rôle Active Directory Domain Services
      win_feature:
        name: AD-Domain-Services
        state: present
      register: ad_result

    - name: Installer le rôle DNS Server
      win_feature:
        name: DNS
        state: present
      register: dns_result

    - name: Installer le rôle DHCP Server
      win_feature:
        name: DHCP
        state: present
      register: dhcp_result

    - name: Redémarrer si nécessaire
```

```
      win_reboot:

      when: ad_result.reboot_required or dns_result.reboot_required or
dhcp_result.reboot_required
```

Commande d'exécution

```
bash

ansible-playbook install_windows_server.yaml
```

Résultat attendu

Lors de l'exécution du playbook, Ansible doit afficher :

```
yaml

TASK [Installer le rôle Active Directory Domain Services] => changed:
[192.168.1.10]

TASK [Installer le rôle DNS Server] => changed: [192.168.1.10]

TASK [Installer le rôle DHCP Server] => changed: [192.168.1.10]

TASK [Redémarrer si nécessaire] => changed: [192.168.1.10]
```

Après redémarrage, les services **AD, DNS et DHCP** sont installés et prêts à être configurés.

Vérifications

Sur le serveur Windows (ou via un module Ansible), exécuter :

```
powershell

Get-WindowsFeature | Where-Object { $_.Installed -eq $true }
```

Les services suivants doivent apparaître comme "Installé" :

- AD-Domain-Services
- DNS
- DHCP

Conclusion

Grâce à ce **playbook simple et modulaire**, l'installation des rôles critiques est **entièrement automatisée**.
Le serveur devient immédiatement prêt à remplir son rôle central dans l'infrastructure :

- Annuaire centralisé (Active Directory)
- Résolution DNS interne
- Distribution des adresses IP (DHCP)

Avantages :

- Gain de temps
- Moins d'erreurs
- Reproductibilité
- Industrialisation de la configuration

4) 2.4 Promotion en contrôleur de domaine et création des utilisateurs AD

Objectifs

Une fois les rôles AD, DNS et DHCP installés, l'étape suivante consiste à :

- **Promouvoir automatiquement** le serveur Windows en **contrôleur de domaine (DC)** pour le domaine hopital.local
- **Créer des utilisateurs Active Directory** de manière automatisée avec Ansible

Cette phase permet de déployer un **annuaire centralisé** et une **gestion normalisée des comptes utilisateurs**.

Outils utilisés

Outil	Rôle
Ansible	Automatisation des tâches sur Windows Server
WinRM	Accès distant au serveur Windows depuis Ansible
win_domain	Module Ansible pour promouvoir un serveur en DC
win_domain_user	Module Ansible pour créer un utilisateur AD

Playbook 1 – Promotion en Contrôleur de Domaine

Fichier : configure_active_directory.yaml

```
yaml
---
- name: Promotion en contrôleur de domaine
  hosts: windows
  tasks:
```

```
    - name: Promouvoir le serveur en DC
      win_domain:
        dns_domain_name: hopital.local
        safe_mode_password: "D@vid0907"
      register: domain_result

    - name: Redémarrer le serveur après promotion
      win_reboot:
      when: domain_result.reboot_required
```

Le mot de passe "safe mode" est utilisé pour les restaurations AD. En production, il doit être chiffré avec Ansible Vault.

Commande d'exécution

```
bash
ansible-playbook configure_active_directory.yaml
```

Le serveur est promu automatiquement en **contrôleur de domaine hopital.local**.

Playbook 2 – Création d'un utilisateur Active Directory

Fichier : create_ad_user.yaml

```
yaml
---
- name: Création d'un utilisateur dans Active Directory
  hosts: windows
  tasks:

    - name: Créer un utilisateur Jean Dupont
      win_domain_user:
        name: "jdupont"
        password: "P@ssword123"
        firstname: "Jean"
        surname: "Dupont"
        upn: "jdupont@hopital.local"
        path: "CN=Users,DC=hopital,DC=local"
        state: present
```

Commande de déploiement

```
bash
```

```
ansible-playbook create_ad_user.yaml
```

Vérification de l'utilisateur

Depuis PowerShell (local ou distant via Ansible) :

```
powershell
```

```
Get-ADUser -Filter "SamAccountName -eq 'jdupont'"
```

Résultat attendu : l'utilisateur **jdupont** est présent dans l'Active Directory et peut se connecter à un poste client du domaine.

Résultats obtenus

Étape	Statut
Promotion du serveur en contrôleur de domaine	✅ OK
Redémarrage post-promotion	✅ OK
Création automatisée de l'utilisateur AD	✅ OK
Vérification de l'utilisateur dans AD	✅ OK

Bénéfices de cette automatisation

- Déploiement **rapide et reproductible** du domaine AD
- **Standardisation** des comptes utilisateurs
- **Réduction des erreurs** manuelles
- Base solide pour appliquer des **GPO**, scripts de connexion, politiques de sécurité…

5) 2.5 Mises à jour automatisées

Objectif

Dans un environnement de production, maintenir les systèmes à jour est indispensable pour :

- **Garantir la sécurité** grâce aux patchs critiques
- **Assurer la stabilité** et la compatibilité des services

- **Réduire les interventions manuelles** grâce à l'automatisation

Dans ce projet, Ansible est utilisé pour déclencher les mises à jour critiques et de sécurité de **Windows Server 2022**, avec un redémarrage automatique si nécessaire.

Outils utilisés

Outil	Rôle
Ansible	Orchestration de la tâche d'update
win_updates	Module Ansible dédié à la mise à jour de Windows
WinRM	Connexion distante entre la machine Ansible et le serveur cible

Playbook Ansible : windows_update.yaml

```
yaml
---
- name: Mettre à jour Windows Server automatiquement
  hosts: windows
  tasks:

    - name: Appliquer les mises à jour critiques et de sécurité
      win_updates:
        category_names:
          - SecurityUpdates
          - CriticalUpdates
        reboot: yes
      register: update_status

    - name: Afficher le nombre de mises à jour appliquées
      debug:
        msg: "Nombre de mises à jour installées : {{
update_status.installed_update_count }}"
```

Commande d'exécution

```
bash
CopierModifier
ansible-playbook windows_update.yaml
```

Résultat attendu

Extrait typique du retour Ansible :

```
yaml

TASK [Appliquer les mises à jour critiques et de sécurité] => changed: [192.168.1.10]

TASK [Afficher le nombre de mises à jour appliquées]

ok: [192.168.1.10] => {

    "msg": "Nombre de mises à jour installées : 5"

}
```

Si un redémarrage est requis, **Ansible l'exécutera automatiquement**.

Vérifications complémentaires

Depuis PowerShell, sur le serveur mis à jour :

```
powershell

Get-HotFix | Sort-Object InstalledOn -Descending | Select-Object -First 5
```

Les dernières mises à jour doivent correspondre à la date d'exécution du playbook.

Conclusion

L'automatisation des mises à jour Windows avec Ansible permet :

- Un **gain de temps** considérable
- Une **meilleure régularité** dans l'application des correctifs
- Une **sécurité renforcée** contre les vulnérabilités système
- Une **base stable pour les environnements critiques** comme les infrastructures hospitalières

Cette méthode peut être intégrée dans une tâche planifiée via **Jenkins, CRON ou GitHub Actions** pour un **maintien en conditions opérationnelles (MCO)** régulier.

6) 2.6 Supervision avec Prometheus et Grafana

Objectif

Mettre en place une solution de **supervision open source** pour :

- Suivre en temps réel l'état des serveurs (CPU, RAM, disque, etc.),
- Détecter rapidement les anomalies ou pannes,

- Visualiser les métriques sur des dashboards clairs,
- Renforcer la **fiabilité** de l'infrastructure hospitalière.

Outils utilisés

Outil	Rôle
Prometheus	Collecte et stockage des métriques
Node Exporter	Agent sur chaque serveur Linux à superviser
Grafana	Interface graphique pour visualiser les données
(optionnel) Windows Exporter	Supervision d'un serveur Windows

Environnement de test

- **VM Linux** (ex : Debian, IP : 192.168.1.29)
  → héberge **Prometheus** et **Grafana**
- **Node Exporter** installé sur la même VM
- (Facultatif) Windows Exporter sur Windows Server

Étapes d'installation

1. Installation de Prometheus

```
bash

sudo useradd --no-create-home --shell /bin/false prometheus

sudo mkdir /etc/prometheus /var/lib/prometheus

# Télécharger la dernière version

wget https://github.com/prometheus/prometheus/releases/latest/download/prometheus-*.tar.gz

# Décompression et configuration

sudo cp prometheus.yml /etc/prometheus/
```

2. Installation de Node Exporter

```
bash

wget https://github.com/prometheus/node_exporter/releases/latest/download/node_exporter-*.tar.gz

tar xvf node_exporter-*.tar.gz

cd node_exporter-*

./node_exporter &
```

3. Configuration de Prometheus

Extrait du fichier prometheus.yml :

```
yaml
global:
  scrape_interval: 15s


scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
```

Lancer Prometheus :

```
bash
./prometheus --config.file=/etc/prometheus/prometheus.yml
```

4. Installation de Grafana

```
bash
sudo apt install -y grafana
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Accès : http://localhost:3000 (admin / admin)

Résultat

- **Prometheus** collecte les données toutes les 15 secondes
- **Grafana** affiche les courbes : CPU, mémoire, réseau, charge
- Alertes personnalisables (ex : charge > 80 %)

Vérification

Dans Grafana :

- Créer un **dashboard** basé sur "Node Exporter Full"
- Ajouter des panels (CPU, RAM, Filesystem)

Conclusion

La supervision avec Prometheus & Grafana permet :

- Une **visibilité temps réel** sur l'infrastructure

- Un meilleur **diagnostic en cas d'incident**
- Une base pour l'**observabilité** dans une logique DevOps complète

Cette solution est **légère**, **open source** et **extensible** (cloud, Kubernetes, containers).

2.7 Introduction à Terraform (Infrastructure as Code)

Objectif

Terraform est un outil d'**Infrastructure as Code (IaC)** qui permet de :

- Décrire toute l'infrastructure (machines, réseaux, ressources) sous forme de **code versionné**
- **Créer, modifier ou détruire** automatiquement des ressources
- **Garantir la reproductibilité** et la traçabilité de l'environnement
- **Préparer une migration vers le cloud** (Azure, AWS, GCP) ou déployer localement (VirtualBox, Proxmox...)

Dans ce projet hospitalier, **Terraform marque le point de bascule vers une vraie démarche DevOps**, où l'infrastructure devient un **composant géré par le code**, versionné et automatisable.

Outils utilisés

Outil	Rôle
Terraform	Déploiement de l'infrastructure (VMs, réseaux...)
VirtualBox / Proxmox	Plateformes de test en local pour valider les modules
Azure CLI / GCP SDK *(optionnel)*	Préparation à une extension vers le cloud

Structure du projet Terraform

Afin de rendre l'infrastructure **modulaire et maintenable**, les fichiers Terraform sont organisés ainsi :

```
bash
```

```
terraform/
```

```
├── main.tf                # Configuration principale
├── variables.tf           # Déclaration des variables
├── outputs.tf             # Sorties utiles (ex : IP, nom des VMs)
├── modules/               # Modules réutilisables
│   ├── vm_linux/
│   ├── vm_windows/
│   └── network/
```

Extrait simplifié de main.tf

```
hcl
provider "virtualbox" {}

module "network" {
  source = "./modules/network"
}

module "vm_linux" {
  source       = "./modules/vm_linux"
  vm_name      = "ansible-manager"
  vm_ip        = "192.168.1.29"
}

module "vm_windows" {
  source       = "./modules/vm_windows"
  vm_name      = "windows-server"
  vm_ip        = "192.168.1.10"
}
```

Étapes de base d'utilisation

```
bash
# 1. Initialiser le projet Terraform
terraform init

# 2. Voir les ressources à créer
terraform plan
```

```
# 3. Créer l'infrastructure
terraform apply
```

Résultat : les **VMs et réseaux sont déployés automatiquement** depuis les fichiers .tf.

Vérifications

- Les VMs sont bien créées dans **VirtualBox** ou **Azure**
- Les **adresses IP** sont correctement assignées
- Ansible peut se connecter automatiquement à la VM **Windows** (WinRM opérationnel)
- Le réseau est fonctionnel et prêt à accueillir les services

Conclusion : le début du DevOps réel

L'utilisation de Terraform marque une **évolution majeure dans le projet** :
On ne configure plus manuellement, **on déclare l'infrastructure comme du code**.

Cela correspond pleinement aux principes DevOps :

- **Automatisation** des environnements
- **Versioning** du code d'infrastructure (Git)
- **Déploiement reproductible**, documenté, modifiable à tout moment
- **Interopérabilité cloud / on-premise** garantie

Cette base technique ouvre désormais la voie à :

- La **conteneurisation avec Docker**
- L'**intégration continue (CI/CD) avec Jenkins ou GitHub Actions**
- La **supervision centralisée avec Prometheus et Grafana**

7) 2.8 Migration vers le Cloud & CI/CD DevOps (Terraform, AKS, GitHub Actions, Docker, Kubernetes)

Transformer l'infrastructure hospitalière simulée en une **infrastructure cloud-native, automatisée, sécurisée et scalable**, en appliquant une démarche **DevOps complète et professionnelle**.

Il ne s'agit pas uniquement de déplacer les services vers le cloud, mais de :

- Définir l'infrastructure **comme du code (IaC)** avec Terraform
- **Conteneuriser les applications** avec Docker
- **Orchestrer le déploiement** avec Kubernetes (AKS)
- **Automatiser le cycle CI/CD** avec Jenkins et GitHub Actions
- **Superviser les services** avec Prometheus & Grafana
- **Sécuriser les accès** avec Vault et RBAC

Stack DevOps utilisée

Catégorie	Outils utilisés
Infrastructure as Code (IaC)	Terraform + Azure CLI
Provisionnement Cloud	RG, VNet, AKS, VMs, NSG, Storage, ServiceBus (via Terraform)
Conteneurisation	Docker + Docker Hub
Orchestration	Kubernetes (AKS), Helm, kubectl
CI/CD	GitHub Actions, Jenkins (pipeline build → push → deploy)
Automatisation config	Ansible (post-deployment Windows & Linux)
Supervision	Prometheus + Grafana
Secrets & sécurité	HashiCorp Vault ou Azure Key Vault
Observabilité cloud	Azure Monitor + Grafana Cloud *(optionnel)*

1. Déploiement cloud avec Terraform

```
bash
```

```
terraform apply
```

Provisionnement complet :

- Resource Group, Virtual Network, Subnets
- Cluster AKS, VMs Linux & Windows, Azure Storage
- Sécurité réseau (NSG), Service Bus, Key Vault

Modularité assurée : le même code Terraform peut être adapté à AWS ou GCP.

```
hcl
```

```
module "aks" {
```

```
  source              = "./modules/aks"
  cluster_name        = "hopital-cluster"
  node_count          = 2
  resource_group_name = module.resource_group.name
}
```

2. Conteneurisation avec Docker

L'application hospitalière (Flask, PHP, MariaDB) est packagée dans des conteneurs Docker.

```
bash
docker build -t vareladavid/hopital-app .
docker push vareladavid/hopital-app
```

Les images sont publiées automatiquement sur Docker Hub à chaque push.

3. CI/CD automatisé avec GitHub Actions & Jenkins

- **GitHub Actions** déclenche le pipeline dès qu'un commit est poussé : build → push → déploiement automatique sur Kubernetes.

```
yaml
jobs:
  deploy:
    steps:
      - name: Build and Push
        run: |
          docker build -t vareladavid/hopital-app .
          docker push vareladavid/hopital-app

      - name: Deploy to AKS
        run: |
          az aks get-credentials --name hopital-cluster
          kubectl apply -f k8s/
```

- **Jenkins** peut aussi être utilisé en complément pour la planification et la gestion des pipelines complexes.

4. Orchestration avec Kubernetes (AKS)

- Les conteneurs sont **répliqués, load-balancés et supervisés**

- Secrets et paramètres sont injectés avec **Helm** ou **kubectl**
- Déploiements mis à jour sans interruption

```
bash
```

```
kubectl apply -f k8s/deployment.yaml
```

5. Supervision avec Prometheus & Grafana

- **Prometheus** collecte les métriques de tous les pods & nodes Kubernetes
- **Grafana** propose des dashboards dynamiques et alertes (email, Slack…)

```
bash
```

```
kubectl port-forward svc/grafana 3000:3000
```

6. Sécurité et gestion des secrets

- **Vault** (ou Azure Key Vault) stocke en sécurité les mots de passe AD, MySQL, tokens API…
- **Traefik/NGINX Ingress** assure le HTTPS et le routage interne
- **RBAC Kubernetes + NSG Azure** sécurisent l'accès aux services

7. Résultats obtenus

Composant	Statut
Infrastructure cloud provisionnée	✅ via Terraform
Applications conteneurisées	✅ via Docker
Déploiement automatisé	✅ via GitHub Actions / Jenkins
Cluster Kubernetes opérationnel	✅ via AKS
Supervision centralisée	✅ via Grafana/Prometheus
Secrets sécurisés	✅ via Vault/Key Vault
Reproductibilité et scalabilité	✅ Totale

Conclusion de la Partie 2 – Approche DevOps & Cloud

Cette deuxième partie du projet représente bien plus qu'un simple exercice technique : elle marque mon entrée dans une démarche DevOps complète et professionnelle.

À travers des outils comme Ansible, Terraform, Docker, GitHub Actions et Kubernetes, j'ai appris à ne plus simplement configurer des serveurs, mais à :

- **Automatiser chaque déploiement,**
- **Versionner toute l'infrastructure,**
- **Documenter les étapes de manière claire,**
- **Superviser l'ensemble du système pour en garantir la fiabilité.**

J'ai compris la force de l'Infrastructure as Code, du CI/CD, de la conteneurisation et de l'orchestration cloud.
Mais surtout, j'ai pu mettre en œuvre des pratiques réelles d'entreprise, en intégrant des notions de sécurité, de scalabilité et de résilience.

Cette étape est un tournant dans mon parcours : je ne suis plus uniquement un technicien,
je deviens acteur de la transformation numérique, capable de concevoir, déployer et maintenir une infrastructure moderne de bout en bout.

CONCLUSION GÉNÉRALE

Ce projet m'a permis de franchir une étape décisive dans mon parcours professionnel. En mettant en place une infrastructure **complète, sécurisée, automatisée et orientée Cloud**, j'ai pu mobiliser concrètement les outils DevOps essentiels : **Ansible, Terraform, Docker, GitHub Actions, Kubernetes, Grafana, Prometheus**, entre autres.

Mais au-delà de l'aspect technique, j'ai compris que le **DevOps n'est pas un objectif à atteindre**, c'est une **démarche continue**.
Les outils évoluent, les environnements deviennent hybrides, multi-cloud, voire intégrés à des systèmes intelligents.
Dans ce contexte, **l'apprentissage est permanent**, et c'est précisément ce qui me motive dans ce métier.

Aujourd'hui, je suis capable de :

- Concevoir une **architecture réseau** stable et sécurisée,
- Déployer des systèmes **Windows et Linux** en environnement professionnel,
- **Automatiser** le cycle de vie complet d'une infrastructure,
- Mettre en place une **chaîne CI/CD moderne**,

- Préparer et accompagner la **migration vers le Cloud**.

Mais je suis aussi prêt à **continuer à évoluer**, à intégrer de **nouvelles pratiques**, à apprendre de mes erreurs et à contribuer à des **projets réels et complexes** en entreprise.

Ce rapport n'est pas une fin,
c'est le **début d'un véritable parcours professionnel** dans l'univers du DevOps et de l'infrastructure moderne.

REMERCIEMENTS

Je tiens à exprimer ma **profonde gratitude** à toutes les personnes qui m'ont accompagné tout au long de ce projet et de mon parcours de reconversion.

Merci à **l'ensemble de l'équipe pédagogique de M2i Formation**, pour leur disponibilité, leur patience et la qualité des enseignements transmis. Leur accompagnement m'a permis de développer des **compétences techniques solides** et de me projeter avec confiance dans le métier d'**Administrateur Systèmes DevOps**.

Je remercie également mes **collègues de formation et de stage**, pour leur collaboration, leur esprit d'équipe et leur soutien constant. Travailler à leurs côtés m'a permis de progresser, de partager des idées, et de surmonter chaque difficulté avec motivation.

Un grand merci à la **Région Île-de-France**, ainsi qu'aux dispositifs de **formation professionnelle**, qui m'ont permis d'accéder à ce parcours et de construire un projet d'avenir structurant et porteur de sens.

Enfin, je souhaite saluer toutes les personnes – **formateurs, conseillers, techniciens, intervenants** – qui ont croisé mon chemin et qui, chacun à leur manière, ont contribué à faire de cette reconversion un projet **concret, ambitieux et humainement riche**.

🔗 **Annexes – Liens vers mes profils professionnels**

- 💻 **GitHub** : https://github.com/VARELAdavidhugo
  (Projets, code source, playbooks Ansible, scripts Terraform…)
- 🐳 **Docker Hub** : https://hub.docker.com/u/vareladavid
  (Images Docker de l'application hospitalière et autres services conteneurisés)

- 👔 **LinkedIn** : https://www.linkedin.com/in/david-varela-devops
  (Parcours, formation, certifications, contact professionnel)

Ressources complémentaires

Ces ressources m'ont accompagné tout au long de mon apprentissage et m'ont permis de structurer mes connaissances techniques et pratiques en DevOps, Cloud et administration systèmes :

Sites et plateformes de référence

- **OpenClassrooms** – Formations certifiantes IT et DevOps
  https://openclassrooms.com
- **Documentation Ansible (RedHat)**
  https://docs.ansible.com
- **Documentation Terraform (HashiCorp)**
  https://developer.hashicorp.com/terraform/docs
- **GitHub Actions – Docs officielles**
  https://docs.github.com/actions
- **Kubernetes – Documentation officielle**
  https://kubernetes.io/fr/docs/
- **Docker Docs**
  https://docs.docker.com
- **Grafana + Prometheus – Monitoring Guide**
  https://grafana.com/docs/grafana/latest/

Livres et lectures recommandés

- **"The DevOps Handbook"** – Gene Kim, Jez Humble
  (Excellent pour comprendre la culture DevOps et l'organisation)
- **"Infrastructure as Code"** – Kief Morris
  (Approche moderne de la gestion d'infrastructure automatisée avec Terraform, Ansible, etc.)
- **"Linux Administration: A Beginner's Guide"** – Wale Soyinka
  (Base solide pour l'administration Linux)
- **"Apprendre à programmer avec Python"** – Gérard Swinnen
  (Très bon pour débuter en scripting Python, utile en automatisation)

Chaînes YouTube utiles

- **TechWorld with Nana** *(DevOps, Kubernetes, Terraform, Jenkins)*
  https://www.youtube.com/@TechWorldwithNana
- **NetworkChuck** *(Cloud, hacking, certifs, café ☕)*
  https://www.youtube.com/c/NetworkChuck

- **Grafikart.fr** *(dev, Docker, Git, Linux, PHP, projets concrets)*
  https://www.youtube.com/@Grafikart

Vocabulaire technique essentiel – Administration Systèmes & DevOps

Terme / Outil	Définition simple	Utilisation dans le projet
VM (Machine Virtuelle)	Ordinateur simulé dans un hôte (via VirtualBox, Proxmox, Azure, etc.)	Création des serveurs (Windows, Linux), des clients, et du pare-feu pfSense
pfSense	Pare-feu open source complet avec interfaces, NAT, règles et DHCP	Sécurisation et segmentation du réseau hospitalier
AD (Active Directory)	Service d'annuaire centralisé pour gérer utilisateurs, groupes, GPO	Authentification, autorisations et structure du domaine hopital.local
DNS (Domain Name System)	Fait le lien entre noms et adresses IP	Résolution des noms internes : hopital.local, ws2022, etc.
DHCP	Attribution automatique des adresses IP aux clients	Configuration via Windows Server et pfSense
Ansible	Automatisation des tâches (configurations, mises à jour, installations)	Déploiement automatique de rôles, utilisateurs, mises à jour Windows
Playbook	Script YAML d'Ansible listant des tâches à exécuter	Exemple : install_windows_server.yaml, create_ad_user.yaml
Terraform	Infrastructure as Code (IaC) pour créer serveurs, réseaux, cloud	Provisionnement automatique de ressources Azure
WinRM	Protocole de gestion à distance pour Windows	Communication entre Ansible (Linux) et Windows Server
Docker	Conteneurisation d'applications avec image portable et exécutable	Création d'un container de l'application hospitalière
Image Docker	Modèle d'un container (application, dépendances, système)	Publication sur Docker Hub : vareladavid/hopital-app
Docker Hub	Registre public pour stocker et partager des images Docker	Stockage de l'image de l'app Flask ou PHP pour l'hôpital

Kubernetes (AKS)	Orchestrateur de containers (AKS = Azure Kubernetes Service)	Exécution automatique des services conteneurisés en haute dispo
CI/CD	Intégration & déploiement continus (pipeline automatisé)	Avec GitHub Actions ou Jenkins
Jenkins	Serveur d'automatisation pour exécuter les pipelines CI/CD	Build, test, déploiement de l'application hospitalière
Git / GitHub	Outil de versionnage de code et plateforme de partage	Stockage du code source, intégration dans le pipeline
Grafana	Interface graphique pour surveiller l'infrastructure avec tableaux de bord	Affichage des métriques Prometheus
Prometheus	Collecte des métriques système et réseau	Suivi de l'activité des containers et serveurs
Vault / Azure Key Vault	Stockage sécurisé des secrets, identifiants, mots de passe	Sécurisation des accès aux bases de données, AD, API
NSG (Network Security Group)	Pare-feu sur Azure pour restreindre les ports et IPs	Sécurisation des ressources cloud
Cluster	Groupe de machines (VMs ou containers) qui travaillent ensemble	Kubernetes avec plusieurs nœuds (AKS)
Helm	Gestionnaire de paquets pour Kubernetes (déploiements simplifiés)	Déploiement rapide et versionné de services
RBAC	Contrôle d'accès basé sur les rôles (souvent utilisé en Kubernetes)	Limitation des droits utilisateurs dans le cluster
YAML	Format de fichier texte utilisé dans Ansible, Kubernetes, GitHub Actions, etc.	Écriture des playbooks, déploiements, pipelines CI/CD
Monitoring / Supervision	Observation des performances système et déclenchement d'alertes	Prometheus, Grafana, logs système, Zabbix

Infrastructure as Code (IaC)	Définir son infrastructure dans des fichiers texte déployables automatiquement	Terraform, Ansible
Pipeline	Ensemble de tâches automatisées qui s'exécutent dans un ordre précis	Exemple : build → push → déploiement sur Kubernetes
Conteneurisation	Isolation d'une application avec son environnement dans un container Docker	Déploiement léger et réutilisable de l'application hospitalière

Dédicace

À **ma fille**,
ma plus grande source de motivation, de courage et d'amour,
à qui je dédie chaque pas de ce parcours.

À **ceux qui ne sont plus là**,
mais qui continuent de vivre en moi,
dans chaque décision, chaque réussite, chaque combat.

Et à **toutes celles et ceux qui, comme moi, ont choisi de changer de vie**,
de relever un défi, parfois seul, parfois en silence,
et de prouver qu'il **n'est jamais trop tard pour apprendre, progresser**,
et construire un avenir plus digne, plus libre, plus humain.