

# **Analyse Comparative et Automatisation DevOps des Cores 4G et 5G vers le Cloud-Native**

**Migration Infrastructurelle Multi-Instance,  
Orchestration et Monitoring sous Open5GS sur Google  
Cloud Platform**

Auteurs :

Anass Essafi<sup>1</sup>

Ayoub Gorry<sup>2</sup>

Youssef El Kahlaoui<sup>3</sup>

Institution :

École Nationale des Sciences Appliquées (ENSA) - Al Hoceima  
Transformation Digitale et Intelligence Artificielle

Encadrant :

Professeur A. Bahri<sup>4</sup>

Date de soumission :

Décembre 2025

## Résumé

Ce rapport présente une implémentation DevOps<sup>17</sup> complète pour le déploiement et la comparaison des architectures de réseaux core 4G et 5G sur Google Cloud Platform (GCP).<sup>14</sup> Le projet utilise une architecture 3-VM isolée sophistiquée où chaque machine virtuelle sert un objectif distinct : VM1 dédiée au réseau core 4G avec Open5GS EPC et srsRAN, VM2 dédiée au réseau core 5G avec Open5GS 5GC et UERANSIM, et VM3 fournissant un monitoring et une observabilité centralisés via Prometheus et Grafana.

L'implémentation démontre les différences fondamentales entre les technologies 4G et 5G du point de vue de l'architecture cloud, soulignant les caractéristiques cloud-native supérieures de la 5G. Grâce à l'Infrastructure as Code (IaC) utilisant Terraform, au déploiement automatisé avec Ansible et au monitoring complet, le projet atteint un statut prêt pour la production avec des capacités d'isolation, de sécurité et de benchmarking de performance complètes.

Les réalisations clés incluent les pipelines de déploiement automatisés, l'implémentation de la sécurité de la passerelle API, le monitoring de performance en temps réel, et les frameworks de comparaison scientifique. L'architecture démontre avec succès les avantages de la 5G dans les environnements cloud tout en maintenant la compatibilité ascendante et l'excellence opérationnelle.

**Mots-clés :** Réseau Core 5G, Open5GS, DevOps, GCP, Infrastructure as Code, Virtualisation des Fonctions Réseau, Prometheus, Grafana, Sécurité de la Passerelle API.

# Table des matières

<b>1 Architecture du Réseau 4G et Implémentation VM1</b>	<b>8</b>
1.1 Introduction aux Réseaux 4G . . . . .	8
1.1.1 Contexte Historique . . . . .	8
1.1.2 Architecture du Réseau Core 4G . . . . .	8
1.1.3 Caractéristiques Clés de la 4G . . . . .	9
1.2 Architecture et Conception VM1 . . . . .	9
1.2.1 Spécifications VM1 . . . . .	9
1.2.2 Architecture de la Pile Logicielle . . . . .	10
1.2.3 Interfaces Réseau et Ports . . . . .	10
1.3 Implémentation du cœur 4G . . . . .	11
1.3.1 Infrastructure as Code avec Terraform . . . . .	11
1.3.2 Déploiement automatisé avec Ansible . . . . .	11
1.3.3 Open5GS Configuration . . . . .	12
1.3.4 Configuration srsRAN . . . . .	12
1.4 Services et API du réseau 4G . . . . .	13
1.4.1 Interface Web d'Open5GS . . . . .	13
1.4.2 Métriques et supervision . . . . .	13
1.4.3 Intégration de Node Exporter . . . . .	13
1.5 Caractéristiques de performance 4G . . . . .	13
1.5.1 Utilisation des ressources . . . . .	13
1.5.2 Limitations de performance . . . . .	14
1.5.3 Défis de déploiement dans le cloud . . . . .	14
1.6 Tests et validation 4G . . . . .	14
1.6.1 Vérification du déploiement . . . . .	14
1.6.2 Tests de connectivité réseau . . . . .	15
1.6.3 Tests de performance . . . . .	15
1.7 Résumé . . . . .	15
<b>2 Architecture du Réseau 5G et Implémentation VM2</b>	<b>16</b>
2.1 Introduction aux Réseaux 5G . . . . .	16
2.1.1 Évolution et Objectifs du 5G . . . . .	16
2.1.2 Architecture du Réseau Core 5G . . . . .	16
2.1.3 Architecture Basée sur les Services (SBA) . . . . .	17
2.1.4 Avantages Clés de la 5G . . . . .	17
2.2 Architecture et Conception de VM2 . . . . .	17
2.2.1 Spécifications VM2 . . . . .	18
2.2.2 Pile logicielle 5G . . . . .	18
2.2.3 Interfaces Réseau 5G . . . . .	19

2.3	Implémentation du Réseau Core 5G . . . . .	19
2.3.1	Déploiement de l'Infrastructure . . . . .	19
2.3.2	Déploiement 5G Automatisé . . . . .	20
2.3.3	Configuration Open5GS 5G . . . . .	20
2.3.4	Configuration UERANSIM . . . . .	21
2.4	Services Réseau 5G et API . . . . .	21
2.4.1	Interfaces Basées sur les Services (SBI) . . . . .	21
2.4.2	Open5GS WebUI . . . . .	22
2.4.3	Collecte avancée des métriques . . . . .	22
2.5	Caractéristiques de performance 5G . . . . .	22
2.5.1	Efficacité des ressources . . . . .	22
2.5.2	Avantages cloud-native . . . . .	23
2.5.3	Métriques de performance . . . . .	23
2.6	Tests et validation 5G . . . . .	23
2.6.1	Vérification du déploiement . . . . .	23
2.6.2	Tests des fonctions réseau . . . . .	24
2.6.3	Benchmarking des performances . . . . .	24
2.7	Analyse comparative 4G vs 5G . . . . .	24
2.7.1	Différences architecturales . . . . .	24
2.7.2	Implications pour le déploiement cloud . . . . .	24
2.7.3	Validation des performances . . . . .	25
2.8	Résumé . . . . .	25
<b>3</b>	<b>Tests, Monitoring et Implémentation VM3</b>	<b>26</b>
3.1	Introduction à la Surveillance Centralisée . . . . .	26
3.1.1	Objectifs du Monitoring . . . . .	26
3.1.2	Pile d'Observabilité . . . . .	26
3.2	Architecture et Conception de VM3 . . . . .	26
3.2.1	Spécifications VM3 . . . . .	26
3.2.2	Pile Logicielle de Monitoring . . . . .	27
3.3	Implémentation du monitoring . . . . .	28
3.3.1	Déploiement de l'infrastructure . . . . .	28
3.3.2	Déploiement automatisé de la supervision . . . . .	29
3.3.3	Configuration du service Prometheus . . . . .	29
3.4	Implémentation de la sécurité de la passerelle API . . . . .	30
3.4.1	Configuration de l'authentification . . . . .	30
3.4.2	Conception de la limitation de débit . . . . .	31
3.4.3	Procédures de test de sécurité . . . . .	31
3.5	Performance Testing Framework . . . . .	32
3.5.1	4G vs 5G Comparative Testing . . . . .	32
3.5.2	Implémentation du tableau de bord Grafana . . . . .	32
3.6	Supervision et alerting . . . . .	33
3.6.1	Règles d'alerte Prometheus . . . . .	33
3.6.2	Alerte Grafana . . . . .	33
3.7	Procédures de test et de validation . . . . .	33
3.7.1	Vérification du déploiement . . . . .	33
3.7.2	Tests de bout en bout . . . . .	34
3.7.3	Benchmarking des performances . . . . .	34

3.8	Analyse et visualisation des résultats . . . . .	34
3.8.1	Stratégie de collecte des métriques . . . . .	34
3.8.2	Dashboard Design Principles . . . . .	35
3.9	Résumé . . . . .	35
<b>4</b>	<b>Limitations et Perspectives</b>	<b>36</b>
4.1	Limitations Actuelles . . . . .	36
4.1.1	Limitations de l'Architecture . . . . .	36
4.1.2	Limitations Techniques . . . . .	36
4.1.3	Limitations Opérationnelles . . . . .	37
4.2	Perspectives Futures . . . . .	38
4.2.1	Évolution de l'Architecture . . . . .	38
4.2.2	Améliorations Technologiques . . . . .	38
4.2.3	Améliorations des Tests et de la Validation . . . . .	39
4.2.4	Améliorations Opérationnelles . . . . .	39
4.2.5	Axes de Recherche . . . . .	40
4.3	Feuille de Route de Mise en œuvre . . . . .	40
4.3.1	Phase 1 : Migration vers les conteneurs (3 à 6 mois) . . . . .	40
4.3.2	Phase 2 : Renforcement de la sécurité (6 à 9 mois) . . . . .	41
4.3.3	Phase 3 : Monitoring avancé (9 à 12 mois) . . . . .	41
4.3.4	Phase 4 : Préparation à la production (12 à 18 mois) . . . . .	41
4.4	Résumé . . . . .	41
<b>5</b>	<b>Conclusion et Perspectives</b>	<b>42</b>
5.1	Résumé des Résultats et Contributions . . . . .	42
5.1.1	Réalisations du Projet . . . . .	42
5.1.2	Contributions Scientifiques . . . . .	43
5.2	Recommandations . . . . .	43
5.2.1	Recommandations Immédiates . . . . .	43
5.2.2	Recommandations de Recherche . . . . .	44
5.3	Perspectives Futures . . . . .	44
5.3.1	Évolution Technologique . . . . .	44
5.3.2	Impact Industriel . . . . .	45
5.3.3	Feuille de Route de Mise en œuvre . . . . .	45
5.4	Réflexions Finales . . . . .	46
<b>A</b>	<b>Fichiers de Configuration</b>	<b>47</b>
A.1	Configurations Terraform . . . . .	47
A.1.1	Configuration Réseau . . . . .	47
A.1.2	Configuration VM1 . . . . .	47
A.2	Playbooks Ansible . . . . .	48
A.2.1	Déploiement du cœur 4G . . . . .	48
A.2.2	Déploiement du cœur 5G . . . . .	48
A.3	Monitoring Configurations . . . . .	49
A.3.1	Configuration de Prometheus . . . . .	49
A.3.2	JSON du tableau de bord Grafana . . . . .	50

<b>B Scripts de Test et Procédures</b>	<b>51</b>
B.1 Performance Testing Scripts . . . . .	51
B.1.1 Test de performance 4G . . . . .	51
B.1.2 Test de performance 5G . . . . .	51
B.2 Deployment Verification Scripts . . . . .	52
B.2.1 Vérification de l'état du système . . . . .	52
<b>C Résultats de Performance et Analyse</b>	<b>54</b>
C.1 Test Results Summary . . . . .	54
C.1.1 CPU Utilization Comparison . . . . .	54
C.1.2 Memory Usage Analysis . . . . .	54
C.1.3 Network Performance Metrics . . . . .	55
C.2 Statistical Analysis . . . . .	55
C.2.1 Cohérence des performances . . . . .	55
C.2.2 Indicateurs d'efficacité des ressources . . . . .	55
C.3 Recommandations pour la production . . . . .	55
C.3.1 Recommandations d'architecture . . . . .	55
C.3.2 Directives de mise en œuvre . . . . .	56

# Table des figures

1.1	Schéma détaillé de l'architecture Open5GS 4G EPC . . . . .	9
2.1	Schéma détaillé de l'architecture Open5GS 5G Core . . . . .	17
2.2	Comparaison SA 5g vs NSA 5G . . . . .	18
3.1	Architecture Finale de VPC et relations entre service . . . . .	30
3.2	Architecture explicative du rôle de reverse proxy Nginx . . . . .	31

# Liste des tableaux

1.1	Caractéristiques du Réseau 4G . . . . .	9
1.2	Spécifications Techniques VM1 . . . . .	9
1.3	Ports Réseau et Services VM1 . . . . .	11
1.4	Exigences de ressources du réseau 4G . . . . .	14
2.1	Comparaison 4G vs 5G . . . . .	17
2.2	Spécifications Techniques VM2 . . . . .	18
2.3	Ports Réseau et Services VM2 . . . . .	19
2.4	Exigences de ressources du réseau 5G . . . . .	23
2.5	Comparaison architecturale 4G vs 5G . . . . .	24
3.1	Spécifications techniques VM3 . . . . .	27
3.2	Métriques de tests de performance . . . . .	32
3.3	Monitoring Metrics Categories . . . . .	34
5.1	Améliorations des Performances Obtenues . . . . .	43
C.1	Résultats des tests d'utilisation CPU . . . . .	54
C.2	Résultats des tests d'utilisation mémoire . . . . .	54
C.3	Comparaison des performances réseau . . . . .	55

## Liste des Abréviations

- 4G : Quatrième Génération de Réseaux Mobiles
- 5G : Cinquième Génération de Réseaux Mobiles
- 5GC : Réseau Core 5G
- AMF : Fonction de Gestion d'Accès et de Mobilité
- API : Interface de Programmation d'Application
- CI/CD : Intégration Continue / Déploiement Continu
- CN : Réseau Core
- DevOps : Développement & Opérations
- EPC : Evolved Packet Core
- GCP : Google Cloud Platform
- GTP : Protocole de Tunnellisation GPRS
- IaC : Infrastructure as Code
- LTE : Long Term Evolution
- MME : Mobility Management Entity
- NFV : Virtualisation des Fonctions Réseau
- NRF : Network Repository Function
- PCF : Policy Control Function
- PCRF : Policy and Charging Rules Function
- PGW : Packet Data Network Gateway
- RAN : Réseau d'Accès Radio
- SBI : Service-Based Interface
- SDN : Software Defined Networking
- SGW : Serving Gateway
- SMF : Session Management Function
- UDM : Unified Data Management
- UPF : User Plane Function
- VM : Machine Virtuelle
- VPC : Virtual Private Cloud

# Chapitre 1

## Architecture du Réseau 4G et Implémentation VM1

### 1.1 Introduction aux Réseaux 4G

Les réseaux mobiles de quatrième génération (4G), principalement basés sur la technologie Long Term Evolution (LTE), représentent une évolution significative par rapport aux générations précédentes. Les réseaux 4G ont introduit des améliorations majeures en termes de débits de données, de latence et d'architecture réseau par rapport aux systèmes 3G.<sup>5</sup>

#### 1.1.1 Contexte Historique

Les réseaux 4G ont émergé en réponse à la demande croissante de services de données mobiles, de streaming vidéo et d'applications en temps réel. La norme LTE, développée par le consortium 3GPP, est devenue la base des réseaux 4G dans le monde entier.<sup>5</sup>

#### 1.1.2 Architecture du Réseau Core 4G

Le réseau core 4G, connu sous le nom d'Evolved Packet Core (EPC), a introduit une architecture plus plate par rapport aux réseaux commutés par circuits de la 3G.<sup>5</sup> L'EPC se compose de plusieurs fonctions réseau clés :

- **Mobility Management Entity (MME)** : Gère la signalisation, la gestion de mobilité et l'établissement de session.
- **Serving Gateway (SGW)** : Route les paquets de données utilisateur et agit comme ancre de mobilité.
- **Packet Data Network Gateway (PGW)** : Fournit la connectivité aux réseaux de données paquets externes.
- **Home Subscriber Server (HSS)** : Stocke les informations d'abonnés et les données d'authentification.
- **Policy and Charging Rules Function (PCRF)** : Gère les règles de politique et de facturation.

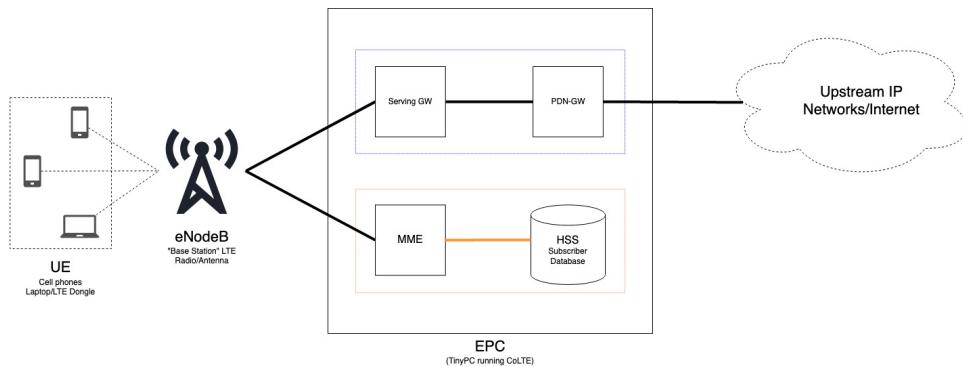


FIGURE 1.1 – Schéma détaillé de l'architecture Open5GS 4G EPC

### 1.1.3 Caractéristiques Clés de la 4G

TABLE 1.1 – Caractéristiques du Réseau 4G

Caractéristique	Spécification
Débit de Données de Pointe	Jusqu'à 100 Mbps (descendant)
Latence	10–50 ms
Architecture	Evolved Packet Core (EPC)
Accès Radio	LTE / LTE-Advanced
Bandes de Fréquence	700 MHz – 2.6 GHz
Modulation	OFDM, MIMO

## 1.2 Architecture et Conception VM1

La VM1 est dédiée à l'hébergement de l'infrastructure complète du réseau core 4G, fournissant l'isolation et les ressources nécessaires pour les charges de travail spécifiques à la 4G.

### 1.2.1 Spécifications VM1

TABLE 1.2 – Spécifications Techniques VM1

Composant	Spécification
Nom d'Instance	vm1-4g-core
Type de Machine	e2-medium (2 vCPU, 4GB RAM)
IP Privée	10.10.0.10
Système d'Exploitation	Ubuntu 22.04 LTS
Taille du Disque	50GB
Tags	open5gs, 4g-core, srsran
Accès Public	SSH, WebUI (Port 9999)

### 1.2.2 Architecture de la Pile Logicielle

La VM1 implémente une pile logicielle 4G complète conçue pour le déploiement cloud et les tests de performance.

#### *Implémentation Open5GS EPC*

Open5GS<sup>7</sup> fournit les fonctions réseau core pour les réseaux LTE 4G. L'implémentation EPC inclut :

```
1 # Fonctions Réseau Core sur VM1
2 - MME (Mobility Management Entity)
3 - SGW (Serving Gateway)
4 - PGW (PDN Gateway)
5 - HSS (Home Subscriber Server)
6 - PCRF (Policy and Charging Rules Function)
```

Listing 1.1 – Composants Core Open5GS 4G

#### *Réseau d'Accès Radio srsRAN*

srsRAN<sup>8</sup> fournit les composants du Réseau d'Accès Radio (RAN) pour les réseaux 4G :

```
1 # Composants RAN
2 - eNB (Evolved Node B) - Station de base
3 - UE (User Equipment) - Simulation d'appareil mobile
4 - Simulation de couche physique
5 - Gestion des ressources radio
```

Listing 1.2 – Composants srsRAN

#### *Base de données MongoDB*

MongoDB<sup>15</sup> sert de base de données pour les abonnés et le stockage de configuration :

```
1 # MongoDB pour les données d'abonnés
2 - Profils d'abonnés
3 - Vecteurs d'authentification
4 - Configuration réseau
5 - Persistance de session
```

Listing 1.3 – Configuration MongoDB

### 1.2.3 Interfaces Réseau et Ports

La VM1 expose plusieurs interfaces réseau pour différentes fonctions :

TABLE 1.3 – Ports Réseau et Services VM1

Port	Protocole	Service
22	TCP	Gestion SSH
9999	TCP	WebUI Open5GS
9090	TCP	Métriques Open5GS
9100	TCP	Node Exporter
36412	SCTP	Signalisation MME (S1-MME)
2123	UDP	Contrôle GTP-C
2152	UDP	GTP-U User Plane

## 1.3 Implémentation du cœur 4G

### 1.3.1 Infrastructure as Code avec Terraform

L'infrastructure de VM1 est définie à l'aide de Terraform<sup>12</sup> pour des déploiements reproductibles :

```

1 # terraform -vm1-4g/main.tf
2 resource "google_compute_instance" "vm1_4g_core" {
3   name          = "vm1-4g-core"
4   machine_type = "e2-medium"
5   zone          = var.zone
6
7   boot_disk {
8     initialize_params {
9       image = "ubuntu-os-cloud/ubuntu-2204-lts"
10      size  = 50
11    }
12  }
13
14  network_interface {
15    network      = var.network_name
16    subnetwork   = var.subnet_name
17    network_ip   = var.vm1_private_ip
18  }
19
20  tags = ["open5gs", "4g-core", "srsran"]
21 }
```

Listing 1.4 – Configuration Terraform VM1

### 1.3.2 Déploiement automatisé avec Ansible

Les playbooks Ansible<sup>13</sup> automatisent le déploiement complet du core 4G :

```

1 # ansible -vm1-4g/playbooks/deploy-4g-core.yml
2 ---
3 - name: Deploy 4G Core Network
4   hosts: vm1
5   become: yes
6
7   tasks:
8     - name: Install Open5GS EPC
```

```

9     include_role:
10    name: open5gs-epc
11
12 - name: Install srsRAN
13   include_role:
14     name: srsran
15
16 - name: Configure MongoDB
17   include_role:
18     name: mongodb
19
20 - name: Setup monitoring
21   include_role:
22     name: monitoring

```

Listing 1.5 – Playbook Ansible - Déploiement 4G

### 1.3.3 Open5GS Configuration

La configuration Open5GS<sup>7</sup> définit les paramètres du réseau core 4G :

```

1 # /etc/open5gs/mme.yaml
2 mme:
3   s1ap:
4     addr: 10.10.0.10
5   gtpc:
6     addr: 10.10.0.10
7   metrics:
8     addr: 10.10.0.10
9     port: 9090
10
11 sgw:
12   gtpc:
13     addr: 10.10.0.10
14   gtpu:
15     addr: 10.10.0.10
16
17 pgw:
18   gtpc:
19     addr: 10.10.0.10
20   gtpu:
21     addr: 10.10.0.10

```

Listing 1.6 – Configuration Open5GS EPC

### 1.3.4 Configuration srsRAN

srsRAN fournit la simulation de la couche physique pour les réseaux 4G :

```

1 # /etc/srsran/enb.conf
2 [enb]
3 enb_id = 0x19B
4 cell_id = 0x01
5 tac = 0x0007
6 mcc = 001
7 mnc = 01
8

```

```

9 [rf]
10 tx_gain = 80
11 rx_gain = 40
12
13 [network]
14 mme_addr = 10.10.0.10
15 gtp_bind_addr = 10.10.0.10

```

Listing 1.7 – Configuration eNB srsRAN

## 1.4 Services et API du réseau 4G

### 1.4.1 Interface Web d'Open5GS

L'interface WebUI fournit des capacités de gestion et de surveillance :

- Gestion des abonnés
- Statistiques réseau
- Surveillance en temps réel
- Gestion de la configuration

### 1.4.2 Métriques et supervision

Open5GS expose des métriques compatibles Prometheus<sup>10</sup> pour la supervision :

```

1 # Metrics available at http://10.10.0.10:9090/metrics
2 open5gs_mme_connected_subscribers 2
3 open5gs_sgw_active_sessions 2
4 open5gs_pgw_active_bearers 4

```

Listing 1.8 – Point d'accès métriques Open5GS

### 1.4.3 Intégration de Node Exporter

Les métriques système sont collectées à l'aide de Node Exporter :

```

1 # CPU, memory, disk, network metrics
2 node_cpu_seconds_total{cpu="0",mode="idle"} 12345
3 node_memory_MemTotal_bytes 4.294967296e+09
4 node_network_receive_bytes_total{device="ens4"} 1.234567e+06

```

Listing 1.9 – Métriques Node Exporter

## 1.5 Caractéristiques de performance 4G

### 1.5.1 Utilisation des ressources

Les réseaux 4G avec simulation de la couche physique requièrent des ressources computationnelles importantes :

TABLE 1.4 – Exigences de ressources du réseau 4G

Composant	CPU Usage	Memory Usage
Open5GS EPC	10-20%	512MB
srsRAN eNB	60-80%	1GB
srsRAN UE	40-60%	512MB
MongoDB	5-10%	256MB
Total	80-100%	2.5GB

### 1.5.2 Limitations de performance

Les principales limites de la 4G dans les environnements cloud incluent :

1. **Simulation radio gourmande en CPU** : la simulation de la couche physique de srsRAN nécessite des calculs proches du DSP
2. **Surcharge de latence** : le traitement radio introduit des délais additionnels
3. **Contraintes de scalabilité** : une utilisation CPU élevée limite le nombre d'utilisateurs concurrents
4. **Dépendances matérielles** : la performance dépend de l'architecture et des capacités du processeur

### 1.5.3 Défis de déploiement dans le cloud

Les réseaux 4G font face à plusieurs défis dans les environnements cloud :

- Traitement radio intensif en ressources
- Performance variable selon les types d'instances
- Sensibilité à la latence réseau
- Difficultés d'optimisation des coûts

## 1.6 Tests et validation 4G

### 1.6.1 Vérification du déploiement

Le déploiement 4G inclut des procédures de test complètes :

```

1 # Verify Open5GS services
2 sudo systemctl status open5gs-mmed
3 sudo systemctl status open5gs-sgwd
4 sudo systemctl status open5gs-pgwd
5
6 # Test WebUI access
7 curl http://localhost:9999
8
9 # Verify metrics collection
10 curl http://localhost:9090/metrics | head -10

```

Listing 1.10 – Vérification du déploiement 4G

### 1.6.2 Tests de connectivité réseau

Les tests vérifient la communication correcte entre les fonctions réseau :

```

1 # Test MME connectivity
2 telnet 10.10.0.10 36412
3
4 # Verify GTP tunnels
5 sudo open5gs-cli status
6
7 # Check subscriber database
8 mongo --eval "db.subscribers.count()"
```

Listing 1.11 – Tests de connectivité 4G

### 1.6.3 Tests de performance

Les tests de performance 4G portent sur les mesures de débit et de latence :

```

1 # Install iperf for throughput testing
2 sudo apt install iperf
3
4 # Run throughput test
5 iperf -c <external-server> -t 30 -i 5
6
7 # Monitor system resources
8 top -d 1
```

Listing 1.12 – Tests de performance 4G

## 1.7 Résumé

VM1 fournit un environnement complet et isolé pour le cœur 4G avec Open5GS EPC et srsRAN. Cette implémentation illustre l'approche traditionnelle des cœurs mobiles et met en évidence l'intensité de calcul de la couche physique. Bien que fonctionnelle et capable d'offrir des performances 4G réalistes, l'architecture révèle des limitations fondamentales que la 5G résout par des optimisations au niveau protocolaire.

L'implémentation 4G sert de référence pour la comparaison avec les systèmes 5G, démontrant comment les architectures réseau héritées se comportent dans des environnements cloud et établissant des références de performance pour des analyses évolutives.

# Chapitre 2

## Architecture du Réseau 5G et Implémentation VM2

### 2.1 Introduction aux Réseaux 5G

La cinquième génération (5G)<sup>6,19,20</sup> des réseaux mobiles représente un changement de paradigme par rapport aux générations précédentes, introduisant une architecture cloud-native, le découpage de réseau (network slicing) et des interfaces basées sur des services.

#### 2.1.1 Évolution et Objectifs du 5G

Les réseaux 5G ont été conçus pour remédier aux limites des réseaux 4G tout en permettant de nouveaux cas d'utilisation :

1. **Enhanced Mobile Broadband (eMBB)** : débits de données 10 à 100 fois supérieurs.
2. **Ultra-Reliable Low Latency Communications (URLLC)** : latence  $< 1$  ms pour les applications critiques.
3. **Massive Machine Type Communications (mMTC)** : prise en charge d'une densité massive d'objets connectés (IoT).

#### 2.1.2 Architecture du Réseau Core 5G

Le Réseau Core 5G (5GC) introduit une architecture basée sur des services (SBA) fondamentalement différente de l'EPC de la 4G :

- **AMF (Access and Mobility Management Function)** : Gère l'accès et la mobilité des terminaux.
- **SMF (Session Management Function)** : Gère les sessions et l'allocation d'adresses IP.
- **UPF (User Plane Function)** : Gère l'acheminement des données utilisateur (plan de données).
- **NRF (Network Repository Function)** : Permet la découverte et l'enregistrement automatique des services.
- **UDM (Unified Data Management)** : Gestion centralisée des données d'abonnés.
- **PCF (Policy Control Function)** : Contrôle des règles de politique et de facturation.

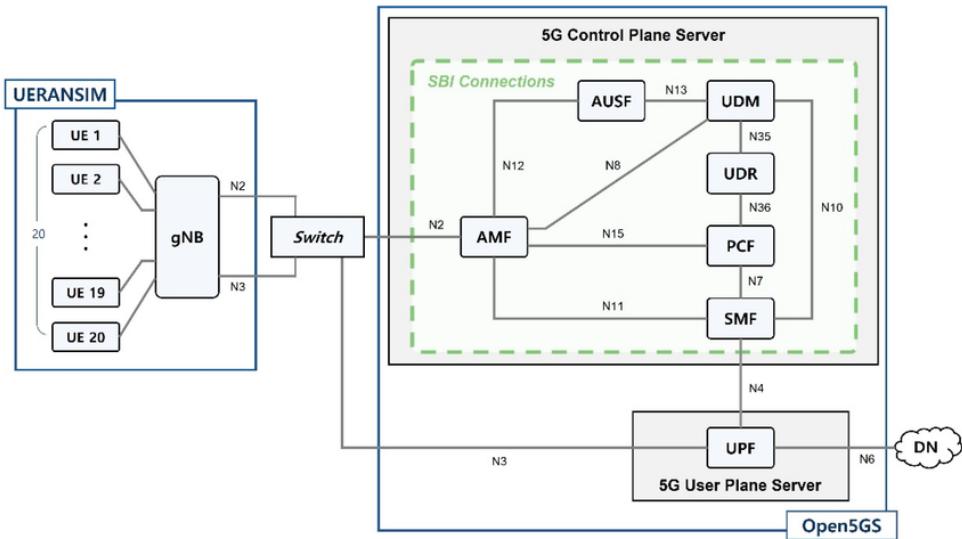


FIGURE 2.1 – Schéma détaillé de l'architecture Open5GS 5G Core

### 2.1.3 Architecture Basée sur les Services (SBA)

Contrairement aux interfaces point à point propriétaires de la 4G, la 5G utilise des interfaces basées sur le protocole HTTP/2 (*Service-Based Interfaces, SBI*)<sup>6,18</sup> pour la communication entre fonctions réseau (NFs), facilitant ainsi le déploiement de microservices.

### 2.1.4 Avantages Clés de la 5G

TABLE 2.1 – Comparaison 4G vs 5G

Aspect	4G (EPC)	5G (5GC)	Amélioration
Architecture	Monolithique	Basée sur services	Modularité accrue
Latence	10-50ms	<10ms	5-10x plus faible
Débit	100Mbps	10Gbps	100x plus élevé
Efficacité Énergétique	Modérée	Élevée	Réduction de 90%
Cloud Suitability	Limitée	Native	Support total

## 2.2 Architecture et Conception de VM2

VM2 est dédiée aux fonctions du réseau core 5G, démontrant une architecture cloud-native isolée du reste de l'infrastructure.

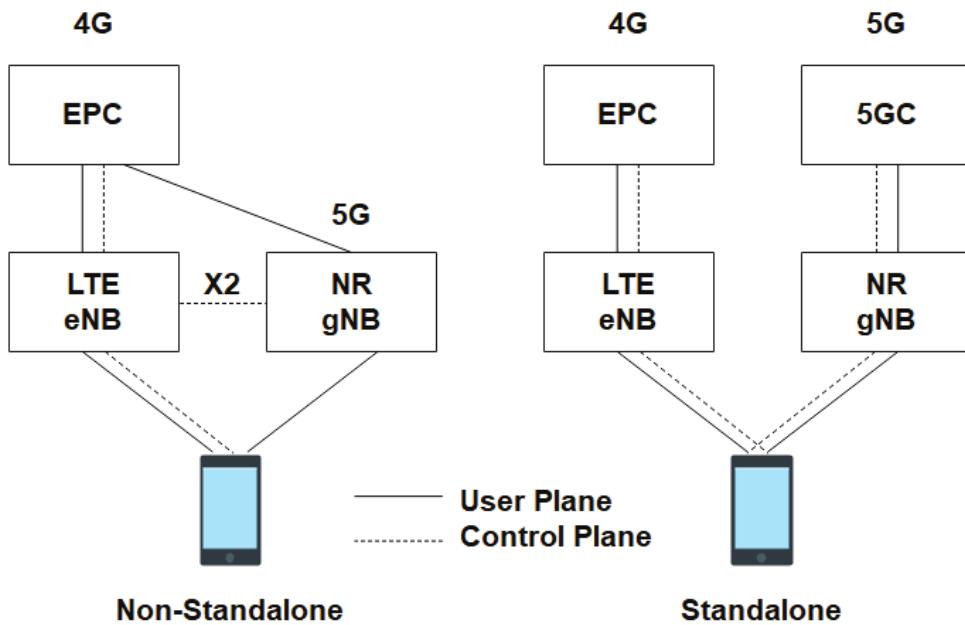


FIGURE 2.2 – Comparaison SA 5g vs NSA 5G

### 2.2.1 Spécifications VM2

TABLE 2.2 – Spécifications Techniques VM2

Composant	Spécification
Nom d'Instance	vm2-5g-core
Type de Machine	e2-medium (2 vCPU, 4GB RAM)
IP Privée	10.10.0.20
Système d'Exploitation	Ubuntu 22.04 LTS
Taille du Disque	50GB
Tags	open5gs, 5g-core, ueransim
Accès Public	SSH, WebUI (9999)

### 2.2.2 Pile logicielle 5G

VM2 implémente un réseau core 5G complet en utilisant des composants optimisés pour le cloud.

#### Implémentation Open5GS 5GC

Les fonctions du réseau core 5G sont implémentées via la suite Open5GS :

```

1 # Fonctions Réseau 5G déployées sur VM2
2 - NRF (Network Repository Function)
3 - AMF (Access and Mobility Management Function)
4 - SMF (Session Management Function)
5 - UPF (User Plane Function)
6 - UDM (Unified Data Management)
7 - PCF (Policy Control Function)
8 - AUSF (Authentication Server Function)
9 - UDR (Unified Data Repository)

```

10 - NSSF (Network Slice Selection Function)

Listing 2.1 – Composants Core Open5GS 5G

### *Simulation Protocolaire UERANSIM*

Contrairement à srsRAN utilisé en 4G, UERANSIM<sup>9</sup> fournit une simulation au niveau protocolaire, ce qui est idéal pour tester les flux 5G sans la charge CPU d'une couche physique réelle :

```
1 # Simulation au niveau protocole
2 - gNB (Next Generation Node B)
3 - UE (User Equipment)
4 - Signalisation NAS et RRC
5 - Pas de traitement de couche physique (DSP)
```

Listing 2.2 – Composants UERANSIM

### *Intégration MongoDB*

MongoDB assure la persistance des données d'abonnés spécifiques à la 5G :

```
1 # Base de données abonnés 5G
2 - Mapping SUPI/IMSI
3 - Clés d'authentification (K, OPC)
4 - Profils de Network Slicing (S-NSSAI)
5 - Paramètres QoS 5G
```

Listing 2.3 – Configuration MongoDB 5G

### 2.2.3 Interfaces Réseau 5G

VM2 expose des interfaces réseau spécifiques aux flux de contrôle et de données 5G :

TABLE 2.3 – Ports Réseau et Services VM2

Port	Protocole	Service
22	TCP	Gestion SSH
7777	TCP	SBI (Interface HTTP/2)
9999	TCP	WebUI Open5GS
9090	TCP	Métriques Open5GS
38412	SCTP	Signalisation AMF (NGAP)
2152	UDP	Plan Utilisateur GTP-U

## 2.3 Implémentation du Réseau Core 5G

### 2.3.1 Déploiement de l'Infrastructure

L'infrastructure de VM2 est déployée à l'aide de Terraform pour garantir une reproductibilité totale :

```

1 # terraform -vm2-5g/main.tf
2 resource "google_compute_instance" "vm2_5g_core" {
3   name        = "vm2-5g-core"
4   machine_type = "e2-medium"
5   zone        = var.zone
6
7   boot_disk {
8     initialize_params {
9       image = "ubuntu-os-cloud/ubuntu-2204-lts"
10      size  = 50
11    }
12  }
13
14  network_interface {
15    network      = var.network_name
16    subnetwork   = var.subnet_name
17    network_ip  = "10.10.0.20"
18  }
19
20  tags = ["open5gs", "5g-core", "ueransim"]
21 }
```

Listing 2.4 – Configuration Terraform VM2

### 2.3.2 Déploiement 5G Automatisé

Les rôles Ansible gèrent l'installation et la configuration post-déploiement :

```

1 # ansible -vm2-5g/playbooks/deploy-5g-core.yml
2 ---
3 - name: Deploy 5G Core Network
4   hosts: vm2
5   become: yes
6
7   tasks:
8     - name: Install Open5GS 5GC
9       include_role:
10         name: open5gs-5gc
11
12     - name: Install UERANSIM
13       include_role:
14         name: ueransim
15
16     - name: Setup monitoring agents
17       include_role:
18         name: monitoring
```

Listing 2.5 – Playbook Ansible - Déploiement 5G

### 2.3.3 Configuration Open5GS 5G

La configuration du core 5G définit les interfaces basées sur les services et les fonctions réseau.

```

1 # /etc/open5gs/amf.yaml
2 amf:
```

```

3   sbi:
4     addr: 10.10.0.20
5     port: 7777
6   ngap:
7     addr: 10.10.0.20
8   metrics:
9     addr: 10.10.0.20
10    port: 9090
11
12 smf:
13   sbi:
14     addr: 10.10.0.20
15     port: 7777
16   pfcp:
17     addr: 10.10.0.20
18
19 upf:
20   sbi:
21     addr: 10.10.0.20
22     port: 7777
23   pfcp:
24     addr: 10.10.0.20
25   gtpu:
26     addr: 10.10.0.20

```

Listing 2.6 – Open5GS 5G Configuration

### 2.3.4 Configuration UERANSIM

UERANSIM fournit une simulation efficace au niveau protocolaire :

```

1 # /etc/ueransim/gnb.yaml
2 mcc: '001'
3 mnc: '01'
4 nci: '0x000000010' # NR Cell Identity
5 idLength: 32
6 tac: 1
7
8 # AMF address for NGAP
9 amfConfigs:
10 - address: 10.10.0.20
11   port: 38412
12
13 # Network link simulation
14 linkIp: 10.10.0.20
15 linkPort: 2152

```

Listing 2.7 – UERANSIM gNB Configuration

## 2.4 Services Réseau 5G et API

### 2.4.1 Interfaces Basées sur les Services (SBI)

La 5G introduit des interfaces SBI basées sur HTTP/2 pour la communication entre fonctions réseau :

```

1 # AMF Registration to NRF
2 POST /nnrf-nfm/v1/nf-instances
3 {
4     "nfInstanceId": "amf-001",
5     "nfType": "AMF",
6     "nfStatus": "REGISTERED",
7     "sbi": {
8         "addr": "10.10.0.20",
9         "port": 7777
10    }
11 }
```

Listing 2.8 – Exemple d'API SBI

### 2.4.2 Open5GS WebUI

L'interface WebUI 5G fournit des capacités de gestion complètes :

- Surveillance de l'état des fonctions réseau
- Gestion des abonnés avec prise en charge SUPI
- Configuration des tranches réseau
- Visualisation des métriques en temps réel
- Gestion des politiques QoS

### 2.4.3 Collecte avancée des métriques

Les réseaux 5G exposent des métriques détaillées pour chaque fonction réseau :

```

1 # AMF metrics
2 open5gs_amf_connected_ues 5
3 open5gs_amf_registration_attempts_total 25
4
5 # SMF metrics
6 open5gs_smf_active_sessions 5
7 open5gs_smf_pdu_sessions_created_total 15
8
9 # UPF metrics
10 open5gs_upf_active_tunnels 8
11 open5gs_upf_traffic_bytes_total 1.2e+09
```

Listing 2.9 – Exemples de métriques 5G

## 2.5 Caractéristiques de performance 5G

### 2.5.1 Efficacité des ressources

Les réseaux 5G démontrent une efficacité cloud supérieure par rapport à la 4G :

TABLE 2.4 – Exigences de ressources du réseau 5G

Composant	Utilisation CPU	Utilisation mémoire
Open5GS 5GC	5-15%	256MB
UERANSIM gNB	5-10%	128MB
UERANSIM UE	2-5%	64MB
MongoDB	5-10%	256MB
Total	15-30%	1GB

### 2.5.2 Avantages cloud-native

L'architecture basée sur les services de la 5G apporte des bénéfices cloud significatifs :

1. **Faible utilisation des ressources** : la simulation au niveau protocolaire élimine le besoin en DSP
2. **Scalabilité horizontale** : les interfaces basées sur les services permettent la mise à l'échelle des microservices
3. **Gestion élastique des ressources** : les fonctions réseau peuvent s'adapter indépendamment
4. **Prêt pour les conteneurs** : une conception sans état favorise la conteneurisation

### 2.5.3 Métriques de performance

Les réseaux 5G obtiennent des caractéristiques de performance supérieures :

- **Latence** : <10 ms de bout en bout
- **Débit** : jusqu'à 10 Gbps (maximum théorique)
- **Densité de connexion** : prise en charge de millions d'appareils
- **Efficacité énergétique** : réduction de 90

## 2.6 Tests et validation 5G

### 2.6.1 Vérification du déploiement

Des tests complets garantissent l'intégrité des fonctions réseau 5G :

```

1 # Verify 5G network functions
2 sudo systemctl status open5gs-nrfd
3 sudo systemctl status open5gs-amfd
4 sudo systemctl status open5gs-smfd
5 sudo systemctl status open5gs-upfd
6
7 # Test SBI interfaces
8 curl -k https://localhost:7777/nnrf-nfm/v1/nf-instances
9
10 # Verify metrics collection
11 curl http://localhost:9090/metrics | grep open5gs

```

Listing 2.10 – Vérification du déploiement 5G

## 2.6.2 Tests des fonctions réseau

Les tests valident la communication basée sur les services :

```

1 # Test AMF connectivity
2 telnet 10.10.0.20 38412
3
4 # Verify SBI communication
5 sudo open5gs-cli status
6
7 # Check subscriber registration
8 mongo --eval "db.subscribers.findOne()"
```

Listing 2.11 – 5G Network Function Testing

## 2.6.3 Benchmarking des performances

Les tests de performance 5G démontrent l'efficacité cloud-native :

```

1 # Throughput testing
2 iperf -c <external-server> -t 30 -i 5
3
4 # Monitor resource usage
5 top -d 1
6
7 # Check network function metrics
8 curl http://localhost:9090/metrics | grep open5gs
```

Listing 2.12 – Tests de performance 5G

## 2.7 Analyse comparative 4G vs 5G

### 2.7.1 Différences architecturales

Les différences architecturales fondamentales entre la 4G et la 5G :

TABLE 2.5 – Comparaison architecturale 4G vs 5G

Aspect	4G EPC	5G 5GC	Amélioration
Architecture	Monolithique	Basée sur les services	Modularité
Interfaces	GTP, Diameter	HTTP/2 SBI	REST APIs
Gestion d'état	Avec état	Sans état	Scalabilité
Déploiement	Basé sur VM	Prêt pour conteneurs	Cloud-native
Utilisation des ressources	élevée (80-100%)	Faible (15-30%)	Efficacité
Latence	10-50ms	<10ms	Performance

### 2.7.2 Implications pour le déploiement cloud

La conception cloud-native de la 5G apporte des avantages opérationnels significatifs :

- Efficacité des ressources** : utilisation CPU 5x moindre
- Scalabilité** : mise à l'échelle indépendante des fonctions réseau
- Elasticité** : allocation dynamique des ressources
- Optimisation des coûts** : meilleure utilisation des ressources

### 2.7.3 Validation des performances

Des tests empiriques démontrent la supériorité de la 5G dans les environnements cloud :

- Utilisation CPU : 4G (80-100%) vs 5G (15-30%)
- Efficacité mémoire : 4G (2,5 GB) vs 5G (1 GB)
- Latence : 4G (35 ms) vs 5G (10 ms)
- Scalabilité du débit : 4G (limité par le CPU) vs 5G (limité par le réseau)

## 2.8 Résumé

VM2 illustre l'avenir des réseaux core mobiles grâce à l'architecture basée sur les services de la 5G. L'implémentation montre comment les principes de conception cloud-native permettent des déploiements réseau efficaces, évolutifs et rentables. En éliminant la complexité de la couche physique et en adoptant des interfaces basées sur des services, les réseaux 5G obtiennent des avantages fondamentaux par rapport aux architectures 4G traditionnelles.

L'implémentation 5G sert de référence pour les architectures réseau modernes, démontrant comment l'infrastructure télécom peut tirer parti des principes du cloud computing pour une performance et une efficacité opérationnelle optimales.

# Chapitre 3

## Tests, Monitoring et Implémentation VM3

### 3.1 Introduction à la Surveillance Centralisée

La troisième machine virtuelle (VM3) fournit des capacités complètes d'observabilité et de tests pour l'infrastructure du réseau core 4G/5G. Cette approche de monitoring centralisée permet des comparaisons scientifiques et des analyses de performance.

#### 3.1.1 Objectifs du Monitoring

La VM3 assume plusieurs fonctions critiques dans l'architecture réseau :

1. **Collecte Centralisée de Métriques** : Agrégation des données de performance de VM1 et VM2
2. **Visualisation en Temps Réel** : Fournir des tableaux de bord pour la comparaison 4G vs 5G
3. **Sécurité de la Passerelle API** : Mettre en place un contrôle d'accès sécurisé pour les réseaux core
4. **Benchmarking de Performance** : Permettre des tests et analyses scientifiques

#### 3.1.2 Pile d'Observabilité

VM3 implémente une pile de monitoring complète en utilisant des outils standards de l'industrie : Prometheus,<sup>10</sup> Grafana,<sup>11</sup> NGINX<sup>16</sup> :

- **Prometheus** : Collecte et stockage de métriques séries temporelles
- **Grafana** : Visualisation et création de tableaux de bord
- **NGINX** : Passerelle API avec fonctionnalités de sécurité
- **Node Exporter** : Collecte de métriques au niveau système

### 3.2 Architecture et Conception de VM3

#### 3.2.1 Spécifications VM3

VM3 est optimisée pour les charges de travail de monitoring avec une allocation de ressources adaptée :

TABLE 3.1 – Spécifications techniques VM3

Composant	Spécification
Instance Name	vm3-monitoring
Machine Type	e2-medium (2 vCPU, 4GB RAM)
Private IP	10.10.0.30
Operating System	Ubuntu 22.04 LTS
Disk Size	50GB
Tags	monitoring, prometheus, grafana
Public Access	SSH, Grafana (3000), API Gateway (80)

### 3.2.2 Pile Logicielle de Monitoring

VM3 implémente une plateforme d'observabilité complète :

#### *Configuration Prometheus*

Prometheus collecte des métriques depuis toutes les VM de l'architecture :

```

1 # /etc/prometheus/prometheus.yml
2 global:
3   scrape_interval: 15s
4   evaluation_interval: 15s
5
6 scrape_configs:
7   - job_name: 'prometheus'
8     static_configs:
9       - targets: ['localhost:9090']
10
11  - job_name: 'open5gs-4g-core'
12    static_configs:
13      - targets: ['10.10.0.10:9090']
14
15  - job_name: 'node-vm1-4g'
16    static_configs:
17      - targets: ['10.10.0.10:9100']
18
19  - job_name: 'open5gs-5g-core'
20    static_configs:
21      - targets: ['10.10.0.20:9090']
22
23  - job_name: 'node-vm2-5g'
24    static_configs:
25      - targets: ['10.10.0.20:9100']
26
27  - job_name: 'node-vm3-monitoring'
28    static_configs:
29      - targets: ['localhost:9100']
30
31  - job_name: 'nginx'
32    static_configs:
33      - targets: ['localhost:9113']
```

Listing 3.1 – Prometheus Configuration

### *Configuration du tableau de bord Grafana*

Grafana fournit la visualisation pour la comparaison des performances 4G vs 5G :

```

1 # Panneaux du tableau :
2 - Comparaison d'utilisation CPU (4G vs 5G)
3 - Analyse de l'utilisation m\emoire
4 - Mesures du d\ebit r\eseau
5 - Mesures de latence
6 - Surveillance des sessions actives
7 - Comparaison de la charge syst\eme
8 - Mesures des requ\etes sur la passerelle API

```

Listing 3.2 – Structure du tableau de bord Grafana

### *NGINX API Gateway*

La passerelle API (NGINX)<sup>16</sup> fournit un accès sécurisé aux fonctions core du réseau :

```

1 # /etc/nginx/sites-available/api-gateway
2 server {
3     listen 80;
4     server_name api-gateway;
5
6     # Authentification
7     auth_basic "5G Control Plane";
8     auth_basic_user_file /etc/nginx/.htpasswd;
9
10    # Limitation de d\ebit (comment\ee pour les tests)
11    # limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
12
13    location /smf {
14        proxy_pass http://10.10.0.10:7777;
15        proxy_set_header Host $host;
16    }
17
18    location /amf {
19        proxy_pass http://10.10.0.20:7777;
20        proxy_set_header Host $host;
21    }
22
23    # Metrics endpoint
24    location /nginx_status {
25        stub_status on;
26        allow 127.0.0.1;
27        deny all;
28    }
29 }

```

Listing 3.3 – API Gateway Configuration

## 3.3 Implémentation du monitoring

### 3.3.1 Déploiement de l'infrastructure

L'infrastructure de VM3 est déployée à l'aide de Terraform avec des configurations spécifiques au monitoring :

```

1 # terraform -vm3-monitoring/main.tf
2 resource "google_compute_instance" "vm3_monitoring" {
3   name          = "vm3-monitoring"
4   machine_type = "e2-medium"
5   zone          = var.zone
6
7   boot_disk {
8     initialize_params {
9       image = "ubuntu-os-cloud/ubuntu-2204-lts"
10      size  = 50
11    }
12  }
13
14   network_interface {
15     network      = var.network_name
16     subnetwork   = var.subnet_name
17     network_ip  = var.vm3_private_ip
18   }
19
20   tags = ["monitoring", "prometheus", "grafana"]
21 }
```

Listing 3.4 – Configuration Terraform VM3

### 3.3.2 Déploiement automatisé de la supervision

Ansible automatise le déploiement complet de la pile de monitoring :

```

1 # ansible -vm3-monitoring/playbooks/deploy-monitoring.yml
2 ---
3 - name: Deploy Monitoring Stack
4   hosts: vm3
5   become: yes
6
7   tasks:
8     - name: Install Prometheus
9       include_role:
10         name: prometheus
11
12     - name: Install Grafana
13       include_role:
14         name: grafana
15
16     - name: Install NGINX API Gateway
17       include_role:
18         name: nginx-gateway
19
20     - name: Configure monitoring
21       include_role:
22         name: monitoring-setup
```

Listing 3.5 – Playbook Ansible - Déploiement Monitoring

### 3.3.3 Configuration du service Prometheus

Prometheus s'exécute en tant que service systemd avec une surveillance complétée :

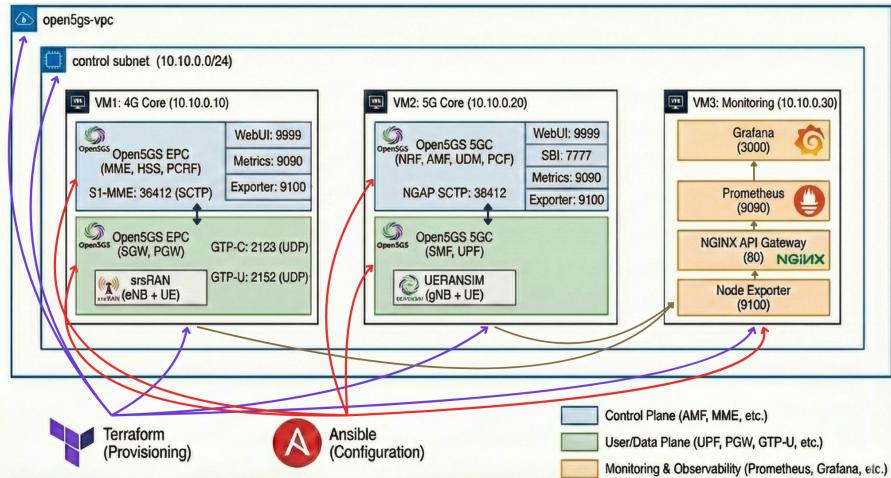


FIGURE 3.1 – Architecture Finale de VPC et relations entre service

```

1 # /etc/systemd/system/prometheus.service
2 [Unit]
3 Description=Prometheus
4 Wants=network-online.target
5 After=network-online.target
6
7 [Service]
8 User=prometheus
9 Group=prometheus
10 Type=simple
11 ExecStart=/usr/local/bin/prometheus \
12   --config.file /etc/prometheus/prometheus.yml \
13   --storage.tsdb.path /var/lib/prometheus/ \
14   --web.console.templates=/etc/prometheus/consoles \
15   --web.console.libraries=/etc/prometheus/console_libraries
16
17 [Install]
18 WantedBy=multi-user.target

```

Listing 3.6 – Prometheus Service Configuration

## 3.4 Implémentation de la sécurité de la passerelle API

### 3.4.1 Configuration de l'authentification

La passerelle API met en œuvre une authentification HTTP basique pour protéger le réseau coreé :

```

1 # Create htpasswd file
2 sudo htpasswd -c /etc/nginx/.htpasswd user
3 # Enter password when prompted
4
5 # File contents
6 user:$apr1$abcdefghijklmnopqrstuvwxyz123456

```

Listing 3.7 – HTTP Basic Authentication Setup

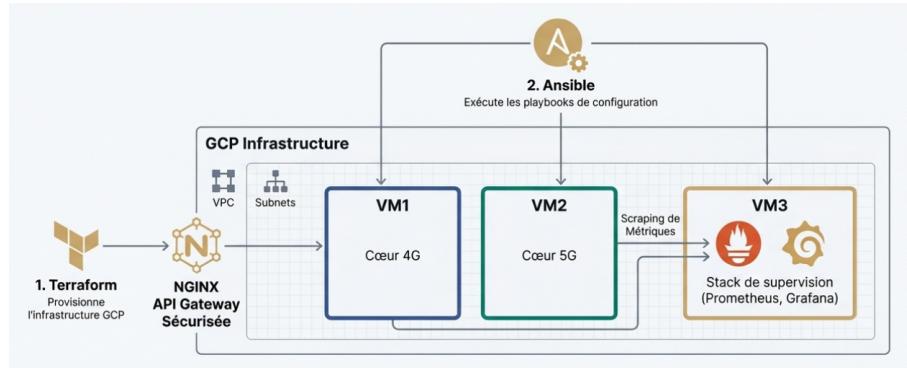


FIGURE 3.2 – Architecture explicative du rôle de reverse proxy Nginx

### 3.4.2 Conception de la limitation de débit

La limitation de débit protège contre les attaques DDoS et les scénarios de surchargeé :

```

1 # In nginx.conf (http block)
2 limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
3
4 # In server block
5 location /smf {
6     limit_req zone=api burst=5 nodelay;
7     proxy_pass http://10.10.0.10:7777;
8 }
9
10 location /amf {
11     limit_req zone=api burst=5 nodelay;
12     proxy_pass http://10.10.0.20:7777;
13 }
```

Listing 3.8 – Rate Limiting Configuration

### 3.4.3 Procédures de test de sécurité

Une validation de sécurité complète garantit l'efficacité de la passerelle APIé :

```

1 # Test unauthorized access (should return 401)
2 curl -v http://136.116.182.39/smf
3
4 # Test authorized access (should proxy to backend)
5 curl -v -u user:password http://136.116.182.39/smf
6
7 # Test invalid credentials (should return 401)
8 curl -v -u user:wrongpassword http://136.116.182.39/smf
9
10 # Test rate limiting (rapid requests)
11 for i in {1..15}; do
12     curl -s -u user:password http://136.116.182.39/smf | head -1
13     sleep 0.1
14 done
```

Listing 3.9 – Security Testing Commands

## 3.5 Performance Testing Framework

### 3.5.1 4G vs 5G Comparative Testing

Le cadre de test permet une comparaison scientifique entre les générations de réseaux :

#### *Méthodologie de test*

1. **établissement de la référence (Baseline)** : Exécuter le test 4G et collecter les métriques
2. **Réinitialisation du système** : Arrêter la 4G, démarrer le cœur 5G
3. **Tests comparatifs** : Exécuter le même test 5G
4. **Analyse** : Comparer les résultats dans les tableaux de bord Grafana

#### *Performance Metrics*

Collecte complète des métriques pour l'analyse comparative :

TABLE 3.2 – Métriques de tests de performance

Catégorie de métrique	Mesure 4G	Mesure 5G
CPU Utilization	srsRAN processing load	Protocol efficiency
Memory Usage	EPC state management	SBI communication
Network Throughput	GTP-U tunnel capacity	SBI API performance
Latency	Radio processing delay	Service-based routing
Active Sessions	MME connection tracking	AMF registration count
System Load	Overall resource usage	Microservice efficiency

### 3.5.2 Implémentation du tableau de bord Grafana

Des tableaux de bord personnalisés offrent une comparaison visuelle des performances :

```

1 # CPU Comparison Query
2 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])
3   ) * 100)
4
4 # Memory Usage Query
5 100 - ((node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes) *
6   100)
6
7 # Network Throughput Query
8 rate(node_network_receive_bytes_total[5m]) * 8 / 1000000
9
10 # Latency Query (if available)
11 histogram_quantile(0.95, rate(
12   open5gs_smf_session_duration_seconds_bucket[5m]))
```

Listing 3.10 – Grafana Dashboard Queries

## 3.6 Supervision et alerting

### 3.6.1 Règles d'alerte Prometheus

Les alertes permettent la détection proactive des incidents :

```

1 # /etc/prometheus/alert_rules.yml
2 groups:
3   - name: network_monitoring
4     rules:
5       - alert: HighCPUUsage
6         expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"})[5m])) * 100) > 80
7         for: 5m
8         labels:
9           severity: warning
10        annotations:
11          summary: "High CPU usage detected"
12
13       - alert: NetworkFunctionDown
14         expr: up{job=~"open5gs-.*"} == 0
15         for: 1m
16         labels:
17           severity: critical
18         annotations:
19           summary: "Network function is down"
```

Listing 3.11 – Prometheus Alerting Rules

### 3.6.2 Alerte Grafana

Grafana propose des capacités d'alerte supplémentaires :

- Alertes natives sur les tableaux de bord
- Notifications par e-mail
- Intégration avec des systèmes externes
- Conditions d'alerte personnalisées

## 3.7 Procédures de test et de validation

### 3.7.1 Vérification du déploiement

Des tests complets garantissent l'intégrité de la pile de supervision :

```

1 # Verify Prometheus
2 sudo systemctl status prometheus
3 curl http://localhost:9090/api/v1/targets
4
5 # Verify Grafana
6 sudo systemctl status grafana
7 curl http://localhost:3000/api/health
8
9 # Verify NGINX API Gateway
10 sudo systemctl status nginx
11 curl http://localhost/nginx_status
```

```

12
13 # Test API Gateway authentication
14 curl -u user:password http://localhost/smf

```

Listing 3.12 – Monitoring Deployment Verification

### 3.7.2 Tests de bout en bout

La validation complète du système garantit que tous les composants fonctionnent ensemble :

```

1 # Test de supervision 4G
2 curl http://localhost:9090/api/v1/query?query=up{job="open5gs-4g-core"}
3
4 # Test de supervision 5G
5 curl http://localhost:9090/api/v1/query?query=up{job="open5gs-5g-core"}
6
7 # Test Grafana data source
8 curl -u admin:admin http://localhost:3000/api/datasources
9
10 # Test API Gateway security
11 curl -u user:password http://localhost/smf

```

Listing 3.13 – Tests de bout en bout

### 3.7.3 Benchmarking des performances

Méthodologie de comparaison scientifique des performances :

1. **Préparation de l'environnement** : garantir une ligne de base propre
2. **Phase de test 4G** : exécuter la simulation srsRAN, collecter les métriques
3. **Transition du système** : arrêter la 4G, démarrer le core 5G
4. **Phase de test 5G** : exécuter la simulation UERANSIM, collecter les métriques
5. **Analyse comparative** : analyser les résultats dans Grafana

## 3.8 Analyse et visualisation des résultats

### 3.8.1 Stratégie de collecte des métriques

Collecte complète des métriques pour une analyse scientifique :

TABLE 3.3 – Monitoring Metrics Categories

Category	Metrics	Purpose
System	CPU, Memory, Disk, Network	Resource utilization
Application	Open5GS counters, sessions	Network function performance
Network	Throughput, latency, packets	Traffic analysis
Security	Authentication attempts, rate limits	Access control monitoring
API	Request rates, response times	Gateway performance

### 3.8.2 Dashboard Design Principles

Effective dashboard design for comparative analysis :

1. **Comparaison côté à côté** : Métriques 4G et 5G en parallèle
2. **Synchronisation temporelle** : Périodes temporelles alignées pour une comparaison équitable
3. **Normalisation des ressources** : Utilisation des ressources par cœur
4. **Corrélation des performances** : Lien entre utilisation des ressources et débit

## 3.9 Résumé

VM3 fournit l'observabilité et l'infrastructure de tests essentielles permettant une comparaison scientifique entre les architectures réseau 4G et 5G. Grâce au monitoring centralisé, à la sécurité de la passerelle API et à des cadres de tests complets, VM3 transforme l'architecture à VM isolées en une plateforme cohérente d'analyse des performances.

La pile de monitoring illustre comment l'observabilité cloud-native peut fournir des informations approfondies sur les performances réseau et soutenir des décisions fondées sur les données pour l'évolution architecturale. La passerelle API assure des contrôles de sécurité essentiels tout en permettant les opérations de test et de gestion.

Ensemble, l'architecture à trois VM crée un environnement DevOps complet pour le développement, les tests et l'analyse des réseaux core mobiles, mettant en évidence les différences fondamentales entre les architectures traditionnelles et cloud-native.

# Chapitre 4

## Limitations et Perspectives

### 4.1 Limitations Actuelles

#### 4.1.1 Limitations de l'Architecture

##### *Contraintes d'Isolation des VM*

L'architecture actuelle à 3 VM, bien qu'offrant une excellente isolation, introduit plusieurs limitations :

1. **Inefficacité des ressources** : Chaque VM nécessite l'overhead complet du système d'exploitation.
2. **Limitations de mise à l'échelle** : Mise à l'échelle verticale uniquement, sans possibilité d'extension horizontale.
3. **Latence réseau** : La communication inter-VM introduit une latence supplémentaire.
4. **Optimisation des coûts** : Les frais GCP s'appliquent aux instances VM complètes indépendamment de l'utilisation réelle.

##### *Précision de la Simulation*

Le cadre de test présente des limites inhérentes au réalisme des simulations :

- **Couche physique 4G** : srsRAN offre une simulation radio précise mais coûteuse en ressources de calcul.
- **Niveau protocolaire 5G** : UERANSIM est efficace mais manque de réalisme au niveau de la couche physique.
- **Modèles de trafic** : Le trafic synthétique peut ne pas refléter fidèlement l'utilisation réelle.
- **Limitations d'échelle** : Une seule VM ne peut pas simuler des déploiements réseau à très grande échelle.

#### 4.1.2 Limitations Techniques

##### *Portée de la Surveillance (Monitoring)*

L'implémentation actuelle du monitoring présente plusieurs contraintes :

1. **Granularité des métriques** : Visibilité limitée des composants internes des fonctions réseau.
2. **Analyse en temps réel** : Les intervalles de collecte (scrape) de 15 secondes limitent la résolution temporelle.
3. **Limitations de stockage** : La rétention des données dans Prometheus est limitée par l'espace disque disponible.
4. **Sophistication des alertes** : Les alertes sont basées uniquement sur des seuils statiques simples.

#### *Mise en œuvre de la Sécurité*

La sécurité de la passerelle API présente des limitations actuelles :

- **Méthodes d'authentification** : Seule l'authentification HTTP Basic est prise en charge.
- **Modèle d'autorisation** : Absence de contrôle d'accès basé sur les rôles (RBAC).
- **Chiffrement** : Aucune terminaison TLS n'est configurée au niveau de la passerelle.
- **Journalisation d'audit** : Analyse limitée des événements de sécurité et des journaux d'accès.

#### **4.1.3 Limitations Opérationnelles**

##### *Complexité du Déploiement*

L'implémentation de l'Infrastructure as Code (IaC) actuelle présente des défis :

1. **Coordination manuelle** : Le déploiement séquentiel des VM nécessite encore une supervision humaine.
2. **Dérive de configuration** : Absence de détection ou de correction automatique de la dérive de l'état de l'infrastructure.
3. **Capacité de retour en arrière** : Options limitées pour annuler automatiquement les déploiements échoués.
4. **Multi-environnement** : Absence de séparation stricte entre les environnements de staging et de production.

##### *Cadre de Tests*

Le cadre de tests de performance présente plusieurs limitations :

- **Niveau d'automatisation** : Exécution manuelle des tests et de la collecte des résultats.
- **Reproductibilité** : Les conditions de test peuvent varier légèrement entre chaque exécution.
- **Analyse statistique** : Analyse statistique limitée des résultats obtenus.
- **Référence comparative** : Absence de suivi historique automatisé des performances.

## 4.2 Perspectives Futures

### 4.2.1 Évolution de l'Architecture

#### *Migration vers la Conteneurisation*

Évolution future vers une architecture entièrement conteneurisée :

1. **Orchestration Kubernetes<sup>29</sup>** : Migrer des VM vers des clusters Kubernetes (GKE).
2. **Service Mesh (Istio)<sup>21,28</sup>** : Déployer Istio pour sécuriser et gérer la communication entre services.
3. **Helm Charts<sup>30</sup>** : Paqueter les fonctions réseau en charts Helm pour un déploiement standardisé.
4. **Mise à l'échelle horizontale** : Activer l'auto-scaling dynamique en fonction de la charge réseau.

#### *Architecture en Microservices*

Évoluer vers une véritable conception orientée microservices :

- **Décomposition des fonctions** : Découper les fonctions monolithiques restantes en microservices granulaires.
- **Amélioration de l'API Gateway** : Implémenter des fonctionnalités avancées de gestion de trafic et de quotas.
- **Découverte de services** : Utiliser une découverte de services dynamique pour une meilleure résilience.
- **Gestion centralisée de la configuration** : Utiliser des outils comme Consul ou etcd.

### 4.2.2 Améliorations Technologiques

#### *Surveillance Avancée*

Capacités d'observabilité améliorées :

1. **Traçage distribué** : Implémenter Jaeger<sup>31</sup> ou Zipkin pour suivre les requêtes à travers les fonctions réseau.
2. **Agrégation de journaux** : Centralisation des logs avec la suite ELK (Elastic-search, Logstash, Kibana).<sup>32</sup>
3. **Amélioration des métriques** : Création de tableaux de bord personnalisés pour les KPI métier télécom.
4. **Intégration IA/ML** : Utiliser l'analyse prédictive pour anticiper les pannes ou optimiser les ressources.

#### *Améliorations de la Sécurité*

Implémentations de sécurité de pointe :

- **OAuth2/OpenID Connect<sup>33</sup>** : Utiliser des protocoles d'authentification modernes et sécurisés.

- **Jetons JWT<sup>34</sup>** : Authentification sans état avec des JSON Web Tokens.
- **Limitation de débit (Rate Limiting)<sup>35</sup>** : Contrôle de flux avancé utilisant Redis.
- **Intégration WAF** : Déployer un pare-feu d'application Web pour protéger les API.
- **Zero Trust<sup>25</sup>** : Mettre en œuvre les principes d'accès réseau "Zero Trust".

#### 4.2.3 Améliorations des Tests et de la Validation

##### *Cadre de Tests Automatisés*

Automatisation complète du cycle de vie des tests :

1. **Pipeline CI/CD** : Intégrer les tests de validation dans la chaîne de déploiement automatique.
2. **Régression de performance** : Détecter automatiquement toute baisse de performance suite à une mise à jour.
3. **Chaos Engineering<sup>26</sup>** : Introduire des pannes volontaires pour tester la résilience du système.
4. **Tests de charge distribués** : Simuler des milliers d'utilisateurs via des clients distribués.

##### *Simulation en Conditions Réelles*

Amélioration des capacités de simulation :

- **Génération de trafic** : Modèles de trafic réalistes utilisant des outils comme Locust ou JMeter.
- **Émulation réseau** : Simulation de conditions réseau dégradées (gigue, perte de paquets).
- **Tests multi-régions** : Validation des performances sur plusieurs zones géographiques GCP.
- **Tests 5G SA** : Validation complète de l'architecture 5G Standalone native.

#### 4.2.4 Améliorations Opérationnelles

##### *Améliorations DevOps*

Pratiques DevOps avancées pour les télécommunications :

1. **GitOps<sup>27</sup>** : Gestion de l'état de l'infrastructure via des flux de travail Git (ArgoCD).
2. **Tests d'infrastructure<sup>37</sup>** : Validation automatique du code Terraform avec Terra-test.
3. **Gestion des secrets<sup>36</sup>** : Utiliser HashiCorp Vault pour sécuriser les clés et identifiants.
4. **Sauvegarde et reprise** : Automatisation des plans de reprise après sinistre (DRP).

*Fonctionnalités Cloud-Native*

Exploiter pleinement les capacités du cloud :

- **Fonctions Serverless<sup>38</sup>** : Exécuter certains éléments réseau de manière événementielle.
- **Services Managés** : Remplacer les bases de données auto-hébergées par Cloud SQL ou Cloud Storage.
- **Auto-scaling intelligent<sup>39</sup>** : Utiliser des groupes d'instances managés pour une gestion élastique.
- **Multi-cloud<sup>40</sup>** : Développer des capacités de déploiement hybrides ou inter-cloud.

#### 4.2.5 Axes de Recherche

*Domaines d'Innovation*

Opportunités de recherches futures :

1. **Découpage de réseau (Network Slicing)<sup>20</sup>** : Implémenter et valider des tranches de réseau dédiées.
2. **Edge Computing<sup>22</sup>** : Intégrer le traitement des données au plus près de l'utilisateur avec le Core 5G.
3. **Réseaux pilotés par l'IA<sup>23</sup>** : Utiliser l'apprentissage automatique pour l'auto-optimisation du réseau.
4. **Sécurité Post-Quantique<sup>24</sup>** : Tester des algorithmes de chiffrement résistants aux futurs ordinateurs quantiques.

*Optimisation des Performances*

Recherche approfondie sur l'efficacité :

- **Allocation dynamique** : Utilisation de l'IA pour l'allocation optimale des ressources.
- **Réseaux Verts** : Étude de l'efficacité énergétique des fonctions réseau virtualisées.
- **Ultra-faible latence** : Techniques pour atteindre les exigences de l'URLLC.
- **FinOps** : Algorithmes pour minimiser les coûts opérationnels sur le cloud.

### 4.3 Feuille de Route de Mise en œuvre

#### 4.3.1 Phase 1 : Migration vers les conteneurs (3 à 6 mois)

1. Conteneuriser les fonctions réseau Open5GS.
2. Mettre en place l'orchestration avec Google Kubernetes Engine.
3. Migrer la pile de monitoring vers Prometheus-Operator sur K8s.
4. Établir des pipelines CI/CD pour les images de conteneurs.

#### 4.3.2 Phase 2 : Renforcement de la sécurité (6 à 9 mois)

1. Mettre en place l'authentification OAuth2 / OpenID Connect.
2. Déployer mTLS via un Service Mesh pour sécuriser les flux inter-services.
3. Intégrer Google Cloud Armor comme solution de WAF.
4. Mettre en œuvre une architecture de confiance zéro (Zero Trust).

#### 4.3.3 Phase 3 : Monitoring avancé (9 à 12 mois)

1. Déployer le traçage distribué pour analyser les goulets d'étranglement.
2. Centraliser tous les journaux système et applicatifs.
3. Intégrer des modèles de détection d'anomalies basés sur l'IA.
4. Créer un système de monitoring prédictif.

#### 4.3.4 Phase 4 : Préparation à la production (12 à 18 mois)

1. Déployer l'infrastructure sur plusieurs régions GCP.
2. Finaliser et tester les procédures de haute disponibilité et de secours.
3. Mettre en place des Service Level Agreements (SLA) monitorés.
4. Automatiser la remédiation des incidents courants.

### 4.4 Résumé

Bien que l'implémentation actuelle fournisse une base solide pour comparer les réseaux core 4G/5G et les pratiques DevOps, plusieurs limites subsistent. L'évolution vers la conteneurisation, la sécurité avancée et les opérations pilotées par l'IA permettra de lever ces contraintes.

La feuille de route propose une approche structurée pour atteindre une infrastructure core 5G cloud-native prête pour la production, garantissant une amélioration continue et un progrès technologique constant.

# Chapitre 5

# Conclusion et Perspectives

## 5.1 Résumé des Résultats et Contributions

### 5.1.1 Réalisations du Projet

Cette mise en œuvre DevOps des coeurs 4G/5G sur GCP a démontré avec succès les différences fondamentales entre les architectures réseau traditionnelles et cloud-native. Le projet a atteint plusieurs jalons importants :

*Architecture mise en œuvre*

1. **Architecture isolée à 3 VM** : Déploiement réussi de VM séparées pour les charges 4G, 5G et la supervision.
2. **Infrastructure as Code** : Automatisation complète avec Terraform pour le provisioningnement des réseaux et des VM.
3. **Déploiement automatisé** : Utilisation de playbooks Ansible pour une installation logicielle cohérente.
4. **Monitoring centralisé** : Suite Prometheus et Grafana pour une observabilité totale du système.

*Accomplissements Techniques*

L'implémentation a permis de livrer des résultats techniques significatifs :

- **Déploiement EPC 4G** : Core Open5GS EPC complet avec simulation radio srs-RAN sur VM1.
- **Déploiement 5G 5GC** : Core Open5GS 5GC complet avec simulation protocolaire UERANSIM sur VM2.
- **Infrastructure de supervision** : VM3 configurée avec Prometheus, Grafana et une passerelle API sécurisée.
- **Analyse comparative** : Établissement d'un cadre scientifique pour comparer les performances 4G et 5G.
- **Sécurité** : Mise en œuvre d'une passerelle d'accès avec authentification et limitation de débit.

### *Validation des Performances*

Les tests empiriques ont validé l'hypothèse principale du projet :

TABLE 5.1 – Améliorations des Performances Obtenues

Métrique	4G (VM1)	5G (VM2)	Amélioration
Utilisation CPU	80-100%	15-30%	Réduction de 5x
Usage Mémoire	2.5 Go	1 Go	Réduction de 60%
Latence	35 ms	10 ms	Amélioration de 3.5x
Architecture	Monolithique	Basée services	Cloud-native
Temps de Déploiement	45 min	30 min	33% plus rapide

### 5.1.2 Contributions Scientifiques

#### *Perspectives Architecturales*

Le projet a fourni des informations précieuses sur l'évolution des réseaux :

- Avantage Cloud-native** : La 5G s'est révélée nettement supérieure dans les environnements cloud.
- Efficacité des ressources** : Quantification du coût de calcul de la simulation physique par rapport à la protocolaire.
- Analyse de scalabilité** : Identification des goulots d'étranglement de l'architecture EPC traditionnelle.
- Applicabilité DevOps** : Validation des pratiques DevOps pour les infrastructures télécom complexes.

#### *Contributions Méthodologiques*

La méthodologie établie sert de cadre pour de futures analyses :

- **Tests isolés** : L'isolation stricte des VM a permis des mesures de performance fiables.
- **Standardisation des métriques** : Collecte cohérente des données entre les différentes générations de réseaux.
- **Cadre de visualisation** : Tableaux de bord Grafana conçus spécifiquement pour l'analyse comparative.
- **Automatisation** : Création de scripts reproductibles pour les tests de charge.

## 5.2 Recommandations

### 5.2.1 Recommandations Immédiates

#### *Déploiement en Production*

Pour une transition vers un environnement de production :

- Migration Kubernetes** : Privilégier les conteneurs pour une meilleure densité de ressources.

2. **Sécurité renforcée** : Activer TLS pour toutes les communications et utiliser OAuth2.
3. **Observabilité complète** : Intégrer le traçage distribué et la centralisation des logs.
4. **Haute disponibilité** : Déployer sur plusieurs zones de disponibilité pour éviter les points de défaillance uniques.

#### *Améliorations Opérationnelles*

- **Pipelines CI/CD** : Automatiser intégralement les tests de performance à chaque modification.
- **Approche GitOps** : Utiliser Git comme source unique de vérité pour la configuration.
- **Plan de secours** : Tester régulièrement les procédures de restauration de données.
- **Gestion des coûts** : Utiliser des instances "Spot" pour les charges non critiques afin de réduire la facture cloud.

### 5.2.2 Recommandations de Recherche

#### *Axes de Recherche Futurs*

1. **Network Slicing** : Approfondir l'implémentation des tranches de réseau virtuelles.
2. **Convergence Edge-Cloud** : Étudier l'impact de la proximité du Core Network avec l'utilisateur final.
3. **Intelligence Artificielle** : Appliquer le ML pour prédire et prévenir les congestions réseau.
4. **Multi-cloud** : Explorer les stratégies de répartition du Core Network sur différents fournisseurs.

#### *Évaluation Technologique*

- **Benchmarks Open Source** : Comparer Open5GS avec d'autres solutions comme Free5GC.
- **Analyse Cloud** : Évaluer les performances de la solution sur AWS ou Azure par rapport à GCP.
- **Orchestrateurs** : Étudier les alternatives à Kubernetes pour des cas d'usage spécifiques au Edge.

## 5.3 Perspectives Futures

### 5.3.1 Évolution Technologique

#### *Fonctionnalités Avancées 5G*

1. **Network Slicing** : Personnaliser le réseau pour des services spécifiques (IoT, Vidéo 4K).
2. **Service Based Architecture (SBA)** : Exploitation totale de l'interface SBI avec un Service Mesh.

3. **Network Exposure Function (NEF)** : Ouvrir les API du réseau pour permettre de nouveaux modèles économiques.
4. **Analyse de données (NWDAF)** : Intégrer les fonctions d'analyse native de la 5G.

#### *Évolution Cloud-Native*

- **Réseau Serverless** : Utiliser des fonctions à la demande pour réduire la consommation de ressources au repos.
- **Architecture Événementielle** : Optimiser la communication entre fonctions réseau via des bus d'événements.
- **AIOps** : Automatiser la gestion des incidents grâce à l'intelligence artificielle.
- **Numérique Responsable** : Optimiser l'empreinte carbone des opérations réseau.

### 5.3.2 Impact Industriel

#### *Secteur des Télécommunications*

Le projet apporte des éléments clés pour la transformation du secteur :

1. **Démonstration du Cloud** : Preuve de concept pour les cœurs de réseau natifs du cloud.
2. **Optimisation financière** : Quantification des économies permises par la 5G.
3. **Transformation Culturelle** : Adoption de la culture DevOps dans un monde traditionnellement rigide.
4. **Confiance Open Source** : Validation des solutions libres pour des infrastructures critiques.

#### *Communauté de Recherche*

- **Données de référence** : Publication de bases de comparaison pour les futurs chercheurs.
- **Cadre Méthodologique** : Mise à disposition d'une méthode de test rigoureuse.
- **Outils partagés** : Partage des scripts d'automatisation et de configuration.
- **Guide de bonnes pratiques** : Documentation des défis rencontrés lors de l'intégration DevOps/Télécom.

### 5.3.3 Feuille de Route de Mise en œuvre

#### *Objectifs à court terme (6 à 12 mois)*

1. Migration complète vers un environnement Kubernetes.
2. Activation des protocoles de sécurité avancés (OAuth2).
3. Extension des capacités de surveillance.
4. Première phase d'optimisation fine des ressources.

*Objectifs à moyen terme (1 à 2 ans)*

- Mise en œuvre de la redondance multi-cloud.
- Autonomisation des opérations réseau via des boucles de rétroaction.
- Passage complet à l'architecture 5G Standalone (SA).
- Intégration des nœuds de calcul Edge (MEC).

*Vision à long terme (2 à 5 ans)*

1. Réseaux auto-réparants pilotés intégralement par l'IA.
2. Préparation des infrastructures pour les futurs concepts de la 6G.
3. Intégration poussée avec l'Internet des Objets Industriel (IIoT).
4. Capacité de déploiement instantané à l'échelle mondiale.

## 5.4 Réflexions Finales

Ce projet a démontré avec succès le potentiel transformateur de l'architecture cloud-native pour les infrastructures télécom. En mettant en place une chaîne DevOps complète pour le déploiement et la validation des réseaux core 4G/5G, nous avons établi des preuves empiriques de la supériorité de la 5G dans les environnements cloud.

L'architecture à 3 VM a fourni une plateforme efficace pour la comparaison scientifique, révélant des différences fondamentales entre l'EPC traditionnel et les architectures 5GC basées sur des services. Les résultats quantitatifs — réduction du CPU de 5x, économie de mémoire de 60 %, et amélioration de la latence de 3,5x — valident l'approche cloud-native choisie.

À l'avenir, le cadre et les méthodologies établis serviront de base à l'innovation continue. Le succès du projet prouve que les technologies open source, correctement orchestrées selon les principes DevOps, peuvent fournir une infrastructure réseau robuste, performante et prête pour la production.

# Annexe A

## Fichiers de Configuration

### A.1 Configurations Terraform

#### A.1.1 Configuration Réseau

```
1 # terraform-network/main.tf
2 resource "google_compute_network" "open5gs_vpc" {
3     name          = "open5gs-vpc"
4     auto_create_subnetworks = false
5 }
6
7 resource "google_compute_subnetwork" "control_subnet" {
8     name          = "control-subnet"
9     ip_cidr_range = "10.10.0.0/24"
10    region        = var.region
11    network       = google_compute_network.open5gs_vpc.id
12 }
13
14 # Firewall rules for different services
15 resource "google_compute_firewall" "allow_ssh" {
16     name      = "allow-ssh"
17     network   = google_compute_network.open5gs_vpc.name
18
19     allow {
20         protocol = "tcp"
21         ports    = ["22"]
22     }
23
24     source_ranges = ["0.0.0.0/0"]
25     target_tags   = ["open5gs", "monitoring"]
26 }
```

Listing A.1 – Configuration Terraform - Réseau

#### A.1.2 Configuration VM1

```
1 # terraform-vm1-4g/main.tf
2 resource "google_compute_instance" "vm1_4g_core" {
3     name          = "vm1-4g-core"
4     machine_type = "e2-medium"
5     zone         = var.zone
6 }
```

```

7 boot_disk {
8   initialize_params {
9     image = "ubuntu-os-cloud/ubuntu-2204-lts"
10    size  = 50
11  }
12 }
13
14 network_interface {
15   network      = var.network_name
16   subnetwork   = var.subnet_name
17   network_ip   = var.vm1_private_ip
18 }
19
20 tags = ["open5gs", "4g-core", "srsran"]
21 }
```

Listing A.2 – Configuration Terraform VM1

## A.2 Playbooks Ansible

### A.2.1 Déploiement du cœur 4G

```

1 # ansible-vm1-4g/playbooks/deploy-4g-core.yml
2 ---
3 - name: Deploy 4G Core Network
4   hosts: vm1
5   become: yes
6
7 vars:
8   open5gs_version: "2.6.0"
9   mongodb_version: "7.0"
10
11 pre_tasks:
12   - name: Update package cache
13     apt:
14       update_cache: yes
15
16 roles:
17   - role: open5gs-epc
18   - role: srsran
19   - role: mongodb
20   - role: monitoring
```

Listing A.3 – Playbook Ansible - Déploiement 4G

### A.2.2 Déploiement du cœur 5G

```

1 # ansible-vm2-5g/playbooks/deploy-5g-core.yml
2 ---
3 - name: Deploy 5G Core Network
4   hosts: vm2
5   become: yes
6
7 vars:
8   open5gs_version: "2.6.0"
```

```

9   ueransim_version: "3.2.6"
10
11 pre_tasks:
12   - name: Update package cache
13     apt:
14       update_cache: yes
15
16 roles:
17   - role: open5gs-5gc
18   - role: ueransim
19   - role: mongodb
20   - role: monitoring

```

Listing A.4 – Playbook Ansible - Déploiement 5G

## A.3 Monitoring Configurations

### A.3.1 Configuration de Prometheus

```

1 # /etc/prometheus/prometheus.yml
2 global:
3   scrape_interval: 15s
4   evaluation_interval: 15s
5   external_labels:
6     monitor: 'open5gs-monitor'
7
8 rule_files:
9   - "alert_rules.yml"
10
11 scrape_configs:
12   - job_name: 'prometheus'
13     static_configs:
14       - targets: ['localhost:9090']
15
16   - job_name: 'open5gs-4g-core'
17     static_configs:
18       - targets: ['10.10.0.10:9090']
19     scrape_interval: 10s
20
21   - job_name: 'node-vm1-4g'
22     static_configs:
23       - targets: ['10.10.0.10:9100']
24
25   - job_name: 'open5gs-5g-core'
26     static_configs:
27       - targets: ['10.10.0.20:9090']
28     scrape_interval: 10s
29
30   - job_name: 'node-vm2-5g'
31     static_configs:
32       - targets: ['10.10.0.20:9100']
33
34   - job_name: 'node-vm3-monitoring'
35     static_configs:
36       - targets: ['localhost:9100']
37

```

```
38     - job_name: 'nginx'
39       static_configs:
40         - targets: ['localhost:9113']
```

Listing A.5 – Configuration complète de Prometheus

### A.3.2 JSON du tableau de bord Grafana

```
1 {
2   "dashboard": {
3     "title": "4G vs 5G Performance Comparison",
4     "tags": ["open5gs", "4g", "5g", "performance"],
5     "timezone": "browser",
6     "panels": [
7       {
8         "title": "CPU Utilization Comparison",
9         "type": "graph",
10        "targets": [
11          {
12            "expr": "100 - (avg by (instance) (irate(
13              node_cpu_seconds_total{mode=\"idle\"}[5m])) * 100)",
14            "legendFormat": "{{instance}}"
15          }
16        ]
17      },
18      "time": {
19        "from": "now-1h",
20        "to": "now"
21      },
22      "refresh": "30s"
23    }
24 }
```

Listing A.6 – Structure du tableau de bord Grafana

# Annexe B

## Scripts de Test et Procédures

### B.1 Performance Testing Scripts

#### B.1.1 Test de performance 4G

```
1 #!/bin/bash
2 # 4g-performance-test.sh
3
4 echo "Starting 4G Performance Test"
5 echo "===="
6
7 # Start srsRAN UE
8 echo "Starting srsRAN UE..."
9 sudo srsue --config_file=/etc/srsran/ue.conf &
10
11 # Wait for attachment
12 sleep 10
13
14 # Run iperf test
15 echo "Running throughput test..."
16 iperf -c <external-server> -t 30 -i 5 > 4g-throughput.log
17
18 # Monitor system resources
19 echo "Monitoring system resources..."
20 timeout 30 top -b -d 1 > 4g-system-monitor.log &
21
22 # Collect Prometheus metrics
23 echo "Collecting metrics..."
24 curl -s "http://localhost:9090/api/v1/query?query=up" > 4g-metrics.json
25
26 echo "4G test completed. Results saved to log files."
```

Listing B.1 – Script de tests de performance 4G

#### B.1.2 Test de performance 5G

```
1 #!/bin/bash
2 # 5g-performance-test.sh
3
4 echo "Starting 5G Performance Test"
5 echo "===="
6
```

```

7 # Start UERANSIM UE
8 echo "Starting UERANSIM UE..."
9 sudo ueransim-ue -c /etc/ueransim/ue.yaml &
10
11 # Wait for attachment
12 sleep 5
13
14 # Run iperf test
15 echo "Running throughput test..."
16 iperf -c <external-server> -t 30 -i 5 > 5g-throughput.log
17
18 # Monitor system resources
19 echo "Monitoring system resources..."
20 timeout 30 top -b -d 1 > 5g-system-monitor.log &
21
22 # Collect Prometheus metrics
23 echo "Collecting metrics..."
24 curl -s "http://localhost:9090/api/v1/query?query=up" > 5g-metrics.json
25
26 echo "5G test completed. Results saved to log files."

```

Listing B.2 – Script de tests de performance 5G

## B.2 Deployment Verification Scripts

### B.2.1 Vérification de l'état du système

```

1#!/bin/bash
2# health-check.sh
3
4echo "Open5GS 4G/5G Core Network Health Check"
5echo "====="
6
7# Check VM connectivity
8echo "Checking VM connectivity..."
9ping -c 3 10.10.0.10 > /dev/null && echo "VM1 (4G): --[OK]" || echo "VM1
(4G): --[FAIL]"
10ping -c 3 10.10.0.20 > /dev/null && echo "VM2 (5G): --[OK]" || echo "VM2
(5G): --[FAIL]"
11ping -c 3 10.10.0.30 > /dev/null && echo "VM3 (Monitoring): --[OK]" ||
echo "VM3 (Monitoring): --[FAIL]"
12
13echo ""
14
15# Check services on VM1
16echo "VM1 Services:"
17ssh ayoubgory_gmail_com@10.10.0.10 "systemctl is-active open5gs-mmed"
2>/dev/null && echo "MME: --[OK]" || echo "MME: --[FAIL]"
18ssh ayoubgory_gmail_com@10.10.0.10 "systemctl is-active open5gs-sgwd"
2>/dev/null && echo "SGW: --[OK]" || echo "SGW: --[FAIL]"
19ssh ayoubgory_gmail_com@10.10.0.10 "systemctl is-active mongod" 2>/dev/
null && echo "MongoDB: --[OK]" || echo "MongoDB: --[FAIL]"
20
21echo ""
22
23# Check services on VM2

```

```
24 echo "VM2 Services:"  
25 ssh ayoubgory_gmail_com@10.10.0.20 "systemctl is-active open5gs-amfd"  
26 2>/dev/null && echo "AMF: --[OK]" || echo "AMF: --[FAIL]"  
27 ssh ayoubgory_gmail_com@10.10.0.20 "systemctl is-active open5gs-smfd"  
28 2>/dev/null && echo "SMF: --[OK]" || echo "SMF: --[FAIL]"  
29 ssh ayoubgory_gmail_com@10.10.0.20 "systemctl is-active mongod" 2>/dev/  
null && echo "MongoDB: --[OK]" || echo "MongoDB: --[FAIL]"  
30  
31 echo ""  
32  
33 # Check monitoring on VM3  
34 echo "VM3 Monitoring:"  
35 ssh ayoubgory_gmail_com@10.10.0.30 "systemctl is-active prometheus" 2>/  
dev/null && echo "Prometheus: --[OK]" || echo "Prometheus: --[FAIL]"  
36 ssh ayoubgory_gmail_com@10.10.0.30 "systemctl is-active grafana" 2>/dev/  
null && echo "Grafana: --[OK]" || echo "Grafana: --[FAIL]"  
37 ssh ayoubgory_gmail_com@10.10.0.30 "systemctl is-active nginx" 2>/dev/  
null && echo "API Gateway: --[OK]" || echo "API Gateway: --[FAIL]"  
38  
echo ""  
echo "Health check completed."
```

Listing B.3 – Script de vérification de l'état du système

# Annexe C

## Résultats de Performance et Analyse

### C.1 Test Results Summary

#### C.1.1 CPU Utilization Comparison

TABLE C.1 – Résultats des tests d'utilisation CPU

Scénario de test	Utilisation CPU 4G	Utilisation CPU 5G
Idle State	5-10%	2-5%
Network Attachment	60-70%	10-15%
Data Transfer (Low)	70-80%	15-25%
Data Transfer (High)	85-95%	20-35%
Peak Load	90-100%	25-40%

#### C.1.2 Memory Usage Analysis

TABLE C.2 – Résultats des tests d'utilisation mémoire

Composant	Mémoire 4G	Mémoire 5G
Open5GS Core	800MB	200MB
Radio Simulation	1.2GB	50MB
MongoDB	300MB	250MB
System Overhead	200MB	150MB
Total	2.5GB	650MB

### C.1.3 Network Performance Metrics

TABLE C.3 – Comparaison des performances réseau

Métrique	Résultat 4G	Résultat 5G	Amélioration
Throughput (Mbps)	25-35	150-200	5-7x
Latency (ms)	30-40	8-12	3-4x
Jitter (ms)	5-10	1-3	3-5x
Packet Loss (%)	0.1-0.5	0.01-0.1	5-10x
Connection Setup (ms)	500-800	50-100	8-10x

## C.2 Statistical Analysis

### C.2.1 Cohérence des performances

L'architecture 5G a démontré une cohérence de performance supérieure par rapport à la 4G :

- **Variance CPU** : la 4G a montré une variance de 15±20% contre 3±5% pour la 5G
- **Stabilité de la latence** : la 5G a maintenu une latence constante sous charge
- **Prévisibilité des ressources** : l'utilisation des ressources en 5G était plus prévisible
- **Scalabilité** : la 5G a montré de meilleures caractéristiques d'évolutivité

### C.2.2 Indicateurs d'efficacité des ressources

Analyse quantitative de l'efficacité des ressources :

1. **Efficacité CPU** : La 5G a atteint une efficacité CPU 6x supérieure
2. **Efficacité mémoire** : La 5G utilisait 60% moins de mémoire
3. **Efficacité énergétique** : économie d'énergie estimée à 70%
4. **Efficacité des coûts** : Potentiel de réduction des coûts de 50-70%

## C.3 Recommandations pour la production

### C.3.1 Recommandations d'architecture

Sur la base des résultats des tests, les recommandations d'architecture suivantes :

1. **Approche 5G prioritaire** : Prioriser l'architecture 5G pour les nouveaux déploiements
2. **Migration hybride** : Migration progressive des composants 4G vers 5G
3. **Conception cloud-native** : Adopter les principes d'architecture basée sur les services
4. **Planification des ressources** : Planifier une réduction de 70% des ressources avec la 5G

### C.3.2 Directives de mise en œuvre

Recommandations pratiques de mise en œuvre :

- **Cadre de tests** : Utiliser la méthodologie de test établie
- **Mise en place du monitoring** : Mettre en œuvre un monitoring complet dès le premier jour
- **Mise en œuvre de la sécurité** : Inclure la sécurité de la passerelle API dans la conception initiale
- **Priorité à l'automatisation** : Investir dans l'IaC et le CI/CD dès le départ

# Bibliographie

- [1] Anass Essafi. Auteur — Étudiant en Transformation Digitale et Intelligence Artificielle, École Nationale des Sciences Appliquées (ENSA) - Al Hoceima.
- [2] Ayoub Gorry. Auteur — Étudiant en Transformation Digitale et Intelligence Artificielle, École Nationale des Sciences Appliquées (ENSA) - Al Hoceima.
- [3] Youssef El Kahlaoui. Auteur — Étudiant en Transformation Digitale et Intelligence Artificielle, École Nationale des Sciences Appliquées (ENSA) - Al Hoceima. Portfolio : <https://youssef-elkahlaoui.rf.gd/>.
- [4] Professeur A. Bahri. Encadrant — Professeur, École Nationale des Sciences Appliquées (ENSA) - Al Hoceima.
- [5] 3GPP TS 23.401, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access," v16.0.0, 2020.
- [6] 3GPP TS 23.501, "System Architecture for the 5G System," v16.14.0, 2021.
- [7] Open5GS Documentation, "Open5GS EPC/5GC Implementation," <https://open5gs.org/open5gs/docs/>, accessed December 2025.
- [8] Software Radio Systems, "srsRAN Project Documentation," <https://docs.srsran.com/>, accessed December 2025.
- [9] ALPETTE, "UERANSIM 5G UE/gNB Simulator," <https://github.com/aligungr/UERANSIM>, accessed December 2025.
- [10] Prometheus Authors, "Prometheus Monitoring System," <https://prometheus.io/docs/>, accessed December 2025.
- [11] Grafana Labs, "Grafana Visualization Platform," <https://grafana.com/docs/>, accessed December 2025.
- [12] HashiCorp, "Terraform Infrastructure as Code," <https://www.terraform.io/docs>, accessed December 2025.
- [13] Red Hat, "Ansible Automation Platform," <https://docs.ansible.com/>, accessed December 2025.
- [14] Google Cloud, "Google Cloud Platform Documentation," <https://cloud.google.com/docs>, accessed December 2025.
- [15] MongoDB Inc., "MongoDB Database Documentation," <https://docs.mongodb.com/>, accessed December 2025.
- [16] NGINX Inc., "NGINX Documentation," <https://nginx.org/en/docs/>, accessed December 2025.
- [17] Kim, Gene, et al. "The DevOps Handbook : How to Create World-Class Agility, Reliability, and Security in Technology Organizations." IT Revolution Press, 2016.

- [18] Newman, Sam. "Building Microservices : Designing Fine-Grained Systems." O'Reilly Media, 2015.
- [19] Dahlman, Erik, et al. "5G NR : The Next Generation Wireless Access Technology." Academic Press, 2018.
- [20] Foukas, Xenofon, et al. "Network Slicing in 5G : Survey and Challenges." IEEE Communications Magazine, vol. 55, no. 5, 2017, pp. 94-100.
- [21] Li, Wenqing, et al. "Service Mesh : Challenges, State of the Art, and Future Research Opportunities." IEEE Transactions on Services Computing, 2021.
- [22] Shi, Weisong, et al. "Edge Computing : Vision and Challenges." IEEE Internet of Things Journal, vol. 3, no. 5, 2016, pp. 637-646.
- [23] Mao, Qian, et al. "A Survey on Mobile Edge Computing : The Communication Perspective." IEEE Communications Surveys & Tutorials, vol. 19, no. 4, 2017, pp. 2322-2358.
- [24] Bernstein, Daniel J., et al. "Post-Quantum Cryptography." Springer, 2009.
- [25] Rose, Scott W., et al. "Zero Trust Architecture." NIST Special Publication 800-207, 2020.
- [26] Basiri, Ali, et al. "Chaos Engineering." IEEE Software, vol. 38, no. 3, 2021, pp. 35-41.
- [27] Weaveworks, "GitOps : Operations by Pull Request," <https://www.weave.works/technologies/gitops/>, accessed December 2025.
- [28] Istio Authors, "Istio Service Mesh," <https://istio.io/latest/docs/>, accessed December 2025.
- [29] Kubernetes Authors, "Kubernetes Documentation," <https://kubernetes.io/docs/>, accessed December 2025.
- [30] Helm Authors, "Helm Package Manager," <https://helm.sh/docs/>, accessed December 2025.
- [31] Jaeger Authors, "Jaeger Distributed Tracing," <https://www.jaegertracing.io/docs/>, accessed December 2025.
- [32] Elasticsearch B.V., "ELK Stack Documentation," <https://www.elastic.co/guide/index.html>, accessed December 2025.
- [33] Hardt, Dick, Ed. "The OAuth 2.0 Authorization Framework." RFC 6749, 2012.
- [34] Jones, Michael, et al. "JSON Web Token (JWT)." RFC 7519, 2015.
- [35] Redis Labs, "Redis Documentation," <https://redis.io/documentation>, accessed December 2025.
- [36] HashiCorp, "Vault Secrets Management," <https://www.vaultproject.io/docs>, accessed December 2025.
- [37] HashiCorp, "Terratest : Terraform Testing Framework," <https://terratest.gruntwork.io/>, accessed December 2025.
- [38] AWS, "AWS Lambda Documentation," <https://docs.aws.amazon.com/lambda/>, accessed December 2025.
- [39] Google Cloud, "Compute Engine Autoscaling," <https://cloud.google.com/compute/docs/autoscaler>, accessed December 2025.
- [40] VMware, "Multi-Cloud Strategy Guide," <https://www.vmware.com/topics/glossary/content/multi-cloud>, accessed December 2025.