

**UNIVERSITÉ JOSEPH KI-ZERBO
(UJKZ)**

**INSTITUT BURKINABÈ DES ARTS ET MÉTIERS
(IBAM)**



**TP : Déploiement d'une Infrastructure Multi-Machines
avec Vagrant et Bash** « https://github.com/Mensan2024/V_projet »

Membre du groupe :

1. ZOUNDI JEAN PHILIPPE
2. OUEDRAOGO R. Daouda
2. BOGNINI Benjamine
4. CODJIA M. Moréno

Sous la supervision de :

Dr Flavien

Année 2024-2025

PLAN

1. Introduction

1.1. Contexte du projet

1.2. Objectifs généraux et pédagogiques

1.3. Technologies utilisées (Vagrant, VirtualBox, Bash, Apache, MySQL, Nginx, Prometheus, Grafana)

2. Description et architecture du projet

2.1. Présentation de l'architecture globale

2.2. Architecture réseau et schéma de topologie

2.3. Rôles des machines

2.4. Organisation et structure des répertoires du projet

3. Mise en place de l'environnement

3.1. Installation des prérequis : Vagrant, VirtualBox, Git

3.2. Création du fichier **Vagrantfile** et explication de sa structure

3.3. Écriture et organisation des scripts de provisioning Bash pour chaque rôle

4. Déploiement de l'infrastructure

4.1. Lancement de l'infrastructure avec **vagrant up**

4.2. Vérification de l'état des machines (**vagrant status**)

4.3. Vérification de la connectivité réseau (**ping, ssh, host**)

5. Configuration des services

5.1. Serveur web (web1 et web2)

5.2. Load Balancer

5.2.1. Installation et configuration de Nginx

5.2.2. Mise en œuvre de l'équilibrage de charge (round-robin)

5.3. Infrastructure MySQL

5.3.1. Installation de MySQL sur db-master et db-slave

5.3.2. Mise en place de la réplication maître-esclave (configuration, utilisateurs, synchronisation)

5.4. Supervision et Monitoring

5.4.1. Installation de Prometheus + **node_exporter**

5.4.2. Installation de Grafana

5.5. Tests et validation avec le client

5.5.1. Test sur l'interface de connexion du Client

5.5.2. Test de répartition des charges

5.5.3. Test de connexion sur les serveurs web1 et web2

5.5.4. Test de connexion aux bases de données

5.5.5. Test du fonctionnement du serveur Monitoring

5.5.6. Test du remplissage du formulaire sur les serveurs

5.5.7. Test de résilience lors des arrêts des serveurs

5.5.8. Test de vérification de la réplication entre le db-master et le db-slave

5.5.9. Test de consommation des nœuds en pourcentage

5.5.10. Test de consommation de la mémoire

6. Conclusion

6.1. Bilan du projet

6.2. Compétences techniques et transversales acquises

6.3. Limites éventuelles et perspectives d'amélioration

6.4. Lien vers le dépôt Git du projet (code source, README, scripts, etc.)

7. Bibliographie

Table des matières

1. INTRODUCTION	5
1.1. Contexte du projet	5
1.2 Objectifs généraux et pédagogiques	5
1.3. Technologies utilisées	5
2. Description de l'architecture	6
2.1 Présentation de l'architecture globale	6
2.2 Architecture réseau	6
2.3 Rôles des machines	7
2.4 Organisation et structure des répertoires du projet	7
3. Mise en place de l'environnement	8
3.1. Installation des prérequis : Vagrant, VirtualBox, Git	8
3.1.1. Vagrant	8
3.2 Création du fichier Vagrantfile et explication de sa structure	10
3.3 Écriture et organisation des scripts de provisioning Bash pour chaque rôle	11
4. Déploiement de l'infrastructure	12
4.1 Lancement de l'infrastructure avec vagrant up	12
4.2 Vérification de l'état des machines (vagrant status)	13
4.3 Vérification de la connectivité réseau (ping, ssh, host)	13
5. Configuration des services	14
5.1. Serveurs Web (web1, web2)	14
5.1.1 Installation d'Apache	14
5.2.1 Déploiement d'une page d'une page de test personnalisée	14
5.2 Load Balancer	14
5.3 Déploiement de la base de données db-master et db-slave dans le Vagrantfile	16
5.4 Supervision et Monitoring	24
5.5 Tests et validation avec le client	30
6. Conclusion	38
7. Bibliographie	40

TABLE DES FIGURES

Figure 1 : l'architecture globale du projet	6
Figure 2 : Tableau montrant les correspondances des adresses IP par machine	7
Figure 3 : Visualisation des dossiers créer sur à travers notre PowerShell.....	8
Figure 4 : Version de Vagrant utilisé	9
Figure 5 : Version du VirtualBox utilisé	10
Figure 6: version de git utilisée	10
Figure 7 : répertoire scripts contenant nos dossiers.....	12
Figure 8 : Les scripts sont remplis avec nos contenues	12
Figure 9 : Tous les machine sont démarrées sans problème	13
Figure 10 : Ping avec chaque IP pour vérifier les connexions.....	13
Figure 11 : affichage des contenus de nos pages web	14
<i>Figure 12 : Lb en fonctionnement.....</i>	14
Figure 13 : Interface graphique du serveur Web 2	15
Figure 14 : Interface graphique du serveur Web 1	15
Figure 15 : projetdB	17
Figure 16 : Affichage de la table actuelle.....	17
Figure 17 : Insertion d'un nouvel étudiant	18
Figure 18: capture mysql installé	18
Figure 19 : Ecoute sur toutes les interfaces réseaux.....	19
Figure 20 : Capture fichier de config mysql sur db-slave	19
Figure 21 : Vérification de la réplication.....	20
Figure 22 : La réplication fonctionne	21
Figure 23 : fichier sql crée.....	22
Figure 24 : Insertion d'un nouvel étudiant	23
Figure 25 : test de réplication	23
Figure 26 : fichier de configuration prometheus.yml	24
Figure 27 : interface de prometheus	26
Figure 28 : Interface gafrana	27
Figure 29 : création de source de données.....	28
Figure 30 : la répartition de charge entre web1 et web2	30
Figure 31 : Contenu du server web1.....	31
Figure 32 : Contenu du server web2.....	31
Figure 33 : Test de connexion sur db-master et db-slave	31
Figure 34 : Test sur le serveur	32
Figure 35 : code html du site	32
Figure 36 : Test de saisie	32
Figure 37 : Réplication effective	35
<i>Figure 38 : Réponse du serveur web1</i>	
Figure 39 : consommation en pourcentage des nœuds du load balancer.....	36
Figure 40 : consommation du CPU	37
<i>Figure 41 : Consommation de la mémoire pour le db master</i>	37

1. INTRODUCTION

1.1. Contexte du projet

La transformation numérique a profondément modifié les pratiques informatiques contemporaines. Face à la montée en puissance des services numériques, les entreprises et les professionnels de l'informatique doivent relever des défis croissants en matière de flexibilité, de performance, de sécurité et de résilience. Dans ce contexte, la virtualisation s'impose comme une solution stratégique : elle permet d'optimiser l'usage des ressources matérielles, de réduire les coûts d'infrastructure et de déployer rapidement des environnements complexes.

C'est dans cette logique que s'inscrit notre projet. Réalisé dans le cadre du cours de Virtualisation et Cloud Computing, il consiste à concevoir et à mettre en œuvre une infrastructure virtualisée multi-machines, simulant un environnement de production réaliste, modulaire et évolutif.

1.2 Objectifs généraux et pédagogiques

Ce travail pratique poursuit plusieurs objectifs pédagogiques clairs :

- Comprendre les principes de la virtualisation et de l'orchestration d'environnements complexes à l'aide d'outils open-source ;
- Maîtriser l'écriture de scripts d'automatisation en Bash, afin de provisionner rapidement et efficacement les machines ;
- Mettre en œuvre des services critiques dans un environnement virtualisé : équilibrage de charge, base de données distribuée, supervision et monitoring ;
- Tester la résilience et la tolérance aux pannes dans un environnement isolé mais réaliste, afin de reproduire les contraintes d'un système en production.

1.3. Technologies utilisées

Notre projet repose exclusivement sur des solutions open-source, garantissant une approche accessible, flexible et pédagogique.

Afin de structurer notre démarche, ce rapport est organisé comme suit : nous commencerons par une description détaillée de l'architecture de l'infrastructure. Ensuite, nous expliquerons la mise en place de l'environnement, le déploiement automatique des services, puis la configuration de chaque composant fonctionnel. Enfin, nous terminerons par une série de tests de validation sur le client avant de conclure sur les compétences acquises et les perspectives d'amélioration.

2. Description de l'architecture

2.1 Présentation de l'architecture globale

L'objectif principal de notre projet est de simuler un environnement de production complet, cohérent et interconnecté, capable de démontrer les principes de base de la virtualisation, de l'automatisation et de la résilience des services. Pour cela, nous avons conçu une architecture multi-machines qui reflète les standards actuels des infrastructures déployées en entreprise.

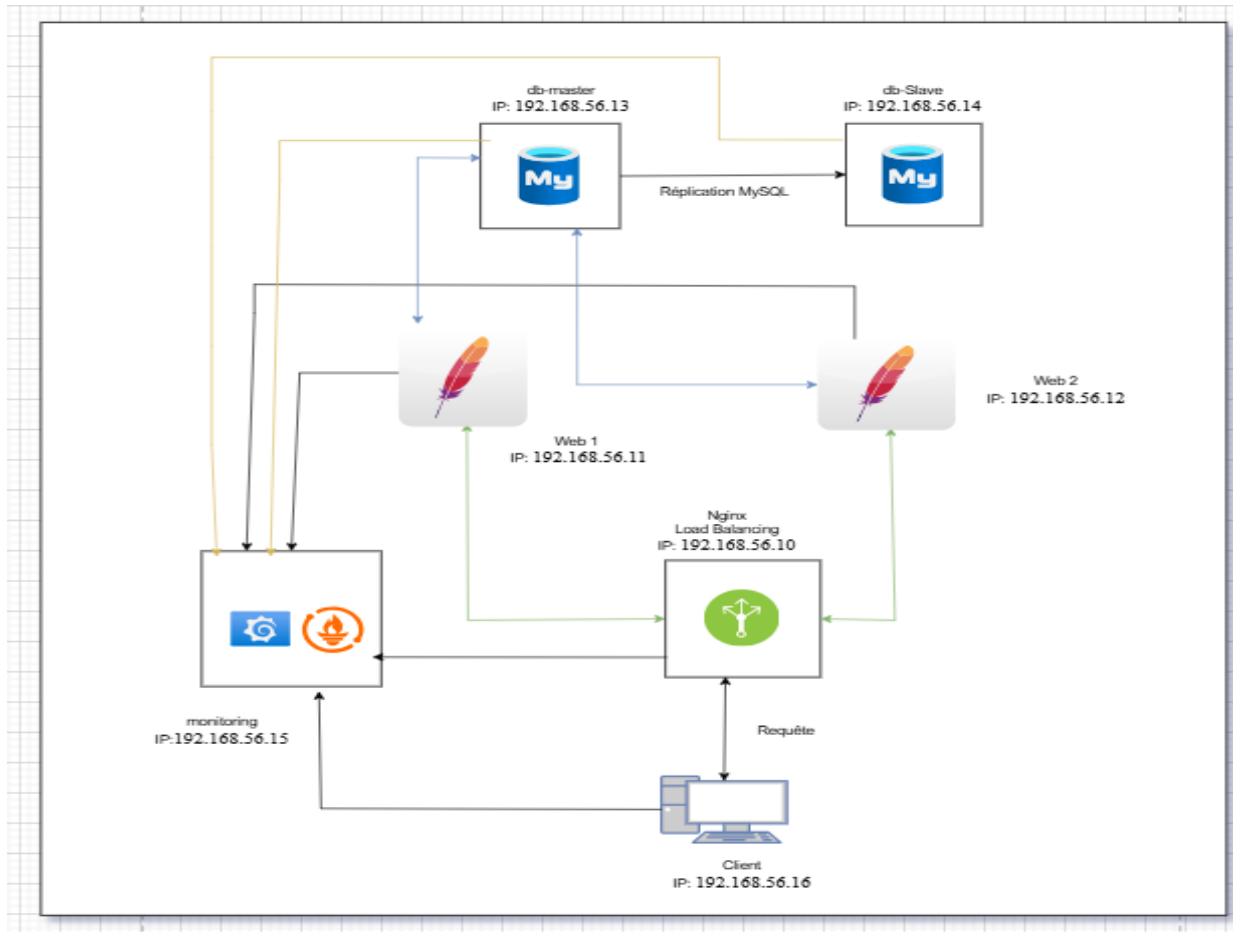


Figure 1 : l'architecture globale du projet

2.2 Architecture réseau

L'ensemble des machines virtuelles sont reliées via un réseau privé en mode host-only configuré sur le sous-réseau **192.168.56.0/24**. Chaque nœud dispose d'une adresse IP statique, ce qui facilite les connexions inter-machines et l'automatisation des configurations.

Voici un aperçu de la table de correspondance des adresses IP par machine :

Nom de la machine	Rôle	Adresse IP
lb	Load Balancer (Nginx)	192.168.56.10
web1	Serveur Web (Apache)	192.168.56.11
web2	Serveur Web (Apache)	192.168.56.12
db-master	Base de données MySQL Maître	192.168.56.13
db-slave	Base de données MySQL Esclave	192.168.56.14
monitoring	Supervision (Prometheus + Grafana)	192.168.56.15
client	Poste de test et validation	192.168.56.16

Figure 2 : Tableau montrant les correspondances des adresses IP par machine

2.3 Rôles des machines

- **lb** (Load Balancer) : Ce nœud utilise Nginx comme répartiteur de charge HTTP. Il distribue intelligemment les requêtes entrantes vers **web1** et **web2**, garantissant un partage équitable de la charge et une haute disponibilité.
- **web1** et **web2** (Serveurs Web) : Ils hébergent l'application web via le serveur Apache2. Ces serveurs sont identiques et leur contenu est synchronisé via les scripts de provisioning, assurant la redondance.
- **db-master** (Serveur MySQL maître) : Il centralise les écritures de la base de données. Sa configuration permet la réplication des données vers l'esclave.
- **db-slave** (Serveur MySQL esclave) : Ce serveur reçoit les mises à jour de la base depuis le maître. Il peut être utilisé pour des lectures, des sauvegardes ou comme bascule en cas de panne du maître.
- **Monitoring** (Serveur de supervision) : Il héberge Prometheus pour la collecte des métriques système, et Grafana pour leur visualisation graphique à travers des tableaux de bord dynamiques.
- **Client** (Machine de test) : Ce poste permet de simuler l'expérience utilisateur finale, de tester les services (ping, requêtes web, accès aux bases de données), et d'observer le comportement du système en conditions réelles.

2.4 Organisation et structure des répertoires du projet

Pour assurer une bonne organisation de notre infrastructure virtuelle, nous avons adopté une structure de répertoires claire, logique et modulaire. Chaque élément du projet est rangé dans un emplacement précis en fonction de son rôle :

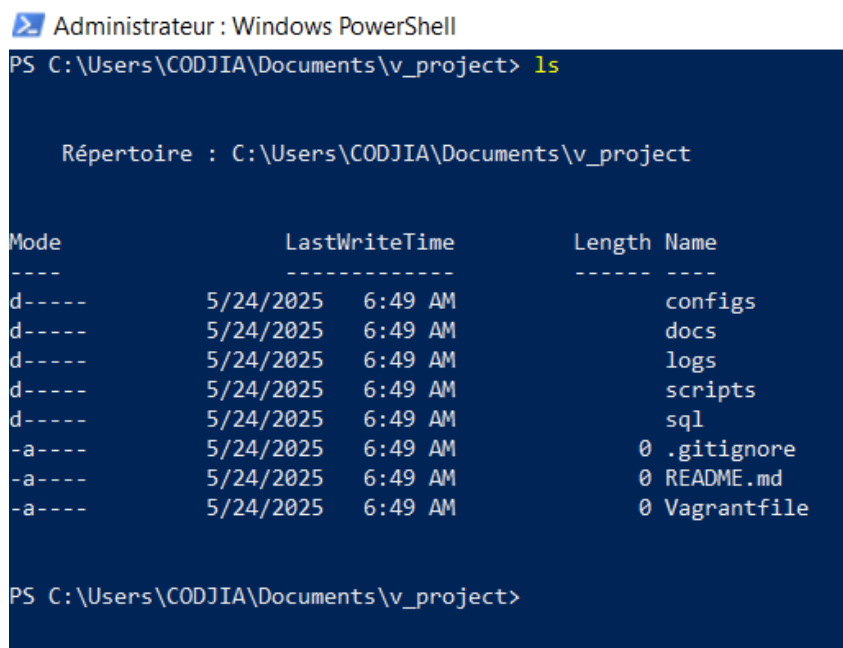
- **.Vagrant** : est essentiel pour que Vagrant puisse suivre et gérer l'état des environnements virtuels créés dans un projet donné. Sans ce fichier, Vagrant ne saurait pas quelle machine est associée au projet, ni son état.

- **configs/** : contient les fichiers de configuration personnalisés pour les services comme Prometheus ou Grafana.
- **docs/** : documentation du projet, avec les captures d'écran des étapes importantes.
- **logs/** : fichiers journaux pour suivre les éventuelles erreurs lors du provisioning.
- **scripts/** : contient tous les scripts de provisioning Bash, un par rôle (web, db, lb...).
- **sql/** : fichiers SQL utilisés pour initialiser la base de données ou tester la réplication
- **gitignore** : Il permet d'exclure des fichiers ou dossiers spécifiques des suivis de Git (évitant qu'ils ne soient commités/pushés).
- **Vagrantfile** : fichier de configuration principal qui décrit l'ensemble de l'infrastructure.

Étape 1 :

Dans PowerShell, on n'a tapé ceci : « cd \$HOME\Documents » pour aller dans notre dossier "Documents"

Étape 2 : Créer notre dossier de projet et sa structure



```

Administrateur : Windows PowerShell
PS C:\Users\CODJIA\Documents\v_project> ls

Répertoire : C:\Users\CODJIA\Documents\v_project

Mode                LastWriteTime         Length Name
----                -
d-----         5/24/2025   6:49 AM             configs
d-----         5/24/2025   6:49 AM             docs
d-----         5/24/2025   6:49 AM             logs
d-----         5/24/2025   6:49 AM            scripts
d-----         5/24/2025   6:49 AM             sql
-a-----         5/24/2025   6:49 AM             0 .gitignore
-a-----         5/24/2025   6:49 AM             0 README.md
-a-----         5/24/2025   6:49 AM             0 Vagrantfile

PS C:\Users\CODJIA\Documents\v_project>

```

Figure 3 : Visualisation des dossiers créés sur à travers notre PowerShell

3. Mise en place de l'environnement

3.1. Installation des prérequis : Vagrant, VirtualBox, Git

3.1.1. Vagrant

Vagrant est un outil d'orchestration qui permet de définir et déployer facilement des environnements virtualisés reproductibles via un fichier de configuration (**Vagrantfile**).

Étapes d'installation :

- Se rendre sur le site officiel : <https://www.vagrantup.com/downloads>
- Télécharger la version correspondant à votre système.
- Installer Vagrant comme un logiciel classique (Next > Next > Finish).
- Vérifier l'installation via un terminal ou PowerShell (en mode administrateur) à travers cette commande : `vagrant --version`.

Pour ce projet, nous avons installé vagrant 2.4.5

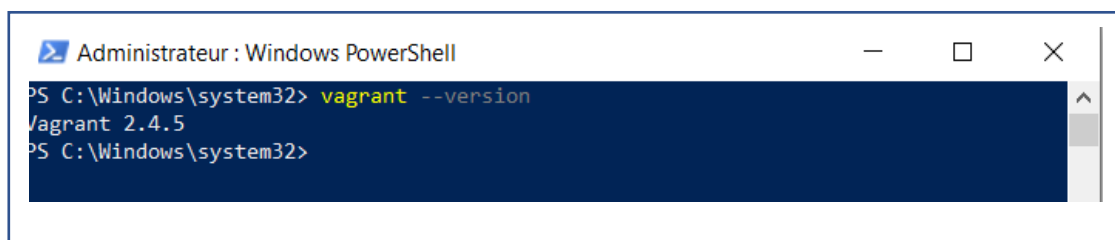


Figure 4 : Version de Vagrant utilisé

*VirtualBox

VirtualBox est un hyperviseur permettant d'exécuter des machines virtuelles sur un système hôte. Il est nécessaire pour que Vagrant puisse créer et gérer des VM localement.

Étapes d'installation :

- Aller sur le site officiel : <https://www.virtualbox.org/>
- Télécharger la version adaptée à votre système d'exploitation (Windows, macOS, Linux).
- Suivre l'assistant d'installation en acceptant les options par défaut.
- Vérifier que l'installation est réussie en lançant VirtualBox.

Pour notre projet, nous avons utilisé VirtualBox version 6.1.50



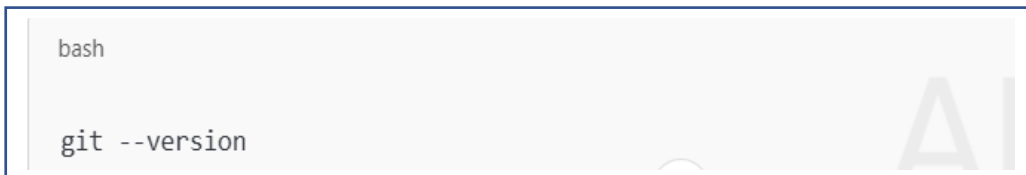
Figure 5 : Version du VirtualBox utilisé

*Git

Système de contrôle de version, utilisé pour versionner notre code et collaborer efficacement.

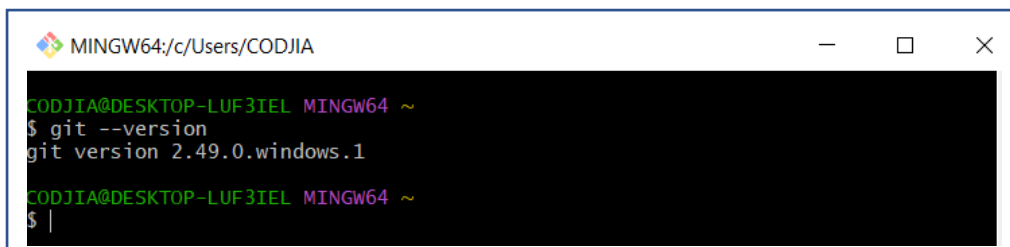
Sous Windows :

- a) Rendez-vous sur le site officiel :
<https://git-scm.com/download/win>
Le téléchargement démarre automatiquement.
- b) Une fois le fichier **.exe** téléchargé, lancez-le.
- c) Lors de l'installation, vous pouvez laisser la configuration par défaut (Next → Next...).
- Vous pouvez aussi cocher l'option "Git Bash Here" pour faciliter l'usage via clic droit.
- d) Une fois installé, ouvrez Git Bash (une interface de terminal) pour l'utiliser.
- e) Pour vérifier l'installation :



```
bash
git --version
```

Pour ce projet, nous avons utilisé git version 2.49.0



```
MINGW64; c:/Users/CODJIA
CODJIA@DESKTOP-LUF3IEL MINGW64 ~
$ git --version
git version 2.49.0.windows.1
CODJIA@DESKTOP-LUF3IEL MINGW64 ~
$ |
```

Figure 6: version de git utilisée

3.2 Création du fichier Vagrantfile et explication de sa structure

Le fichier Vagrantfile est un fichier de configuration écrit en **Ruby**, utilisé par **Vagrant** pour décrire l'environnement de machines virtuelles à créer. Il est le cœur de notre infrastructure car il décrit la configuration des 7 machines virtuelles, leurs adresses IP, leurs rôles respectifs et les scripts de provisioning à exécuter.

Voir l'aperçu de notre fichier Vagrantfile avec quelques descriptions :

```

1 # -*- mode: ruby -*-
2 # vi: set ft=ruby :
3
4 Vagrant.configure("2") do |config|
5   config.vm.box = "ubuntu/bionic64"
6
7   # Machine 1 : Load Balancer
8   config.vm.define "lb" do |lb|
9     lb.vm.hostname = "lb.local"
10    lb.vm.network "private_network", ip: "192.168.56.10"
11    lb.vm.provider "virtualbox" do |vb|
12      vb.name = "lb"
13      vb.memory = 1024
14    end
15    lb.vm.provision "shell", path: "scripts/setup_lb.sh"
16  end
17
18  # Machine 2 : Serveur web 1
19  config.vm.define "web1" do |web1|
20    web1.vm.hostname = "web1.local"
21    web1.vm.network "private_network", ip: "192.168.56.11"
22    web1.vm.provider "virtualbox" do |vb|
23      vb.name = "web1"
24      vb.memory = 1024
25    end
26    web1.vm.provision "shell", path: "scripts/setup_web.sh"
27  end
28
29  # Machine 3 : Serveur web 2
30  config.vm.define "web2" do |web2|
31    web2.vm.hostname = "web2.local"
32    web2.vm.network "private_network", ip: "192.168.56.12"
33    web2.vm.provider "virtualbox" do |vb|
34      vb.name = "web2"
35      vb.memory = 1024
36    end
37  end

```

```

# Machine 4 : DB Master
config.vm.define "db-master" do |db_master|
  db_master.vm.hostname = "db-master.local"
  db_master.vm.network "private_network", ip: "192.168.56.13"
  db_master.vm.provider "virtualbox" do |vb|
    vb.name = "db-master"
    vb.memory = 1024
  end
  db_master.vm.synced_folder "C:/Users/CODRIA/Documents/v_project", "/home/vagrant/shared"
  db_master.vm.provision "shell", path: "scripts/setup_db_master.sh"
end

# Machine 5 : DB Slave
config.vm.define "db-slave" do |slave|
  slave.vm.hostname = "db-slave.local"
  slave.vm.network "private_network", ip: "192.168.56.14"
  slave.vm.provider "virtualbox" do |vb|
    vb.name = "db-slave"
    vb.memory = 1024
  end
  slave.vm.provision "shell", path: "scripts/setup_db_slave.sh"
end

# Machine 6 : Monitoring
config.vm.define "monitoring" do |monitoring|
  monitoring.vm.hostname = "monitoring.local"
  monitoring.vm.network "private_network", ip: "192.168.56.15"
  monitoring.vm.provider "virtualbox" do |vb|
    vb.name = "monitoring"
    vb.memory = 1024
  end
  monitoring.vm.provision "shell", path: "scripts/setup_monitoring.sh"
end

```

```

# Machine 7 : Client (interface graphique avec autologin)
config.vm.define "client" do |client|
  client.vm.hostname = "client.local"
  client.vm.network "private_network", ip: "192.168.56.16"
  client.vm.provider "virtualbox" do |vb|
    vb.name = "client"
    vb.memory = 1024
  end
  client.vm.provision "shell", inline: <<-SHELL
  sudo apt-get -y update
  sudo apt-get -y upgrade
  sudo apt-get install -y xfce4 lightdm lightdm-gtk-greeter firefox net-tools htop

  if ! id "client" && dev/null; then
    sudo useradd -m -s /bin/bash client
    echo 'client:1234' | sudo chpasswd
    sudo usermod -aG sudo client
  fi

  sudo groupadd -f autologin
  sudo usermod -aG autologin client

  sudo mkdir -p /etc/lightdm/lightdm.conf.d
  echo "[SeatDefaults]"
autologin-user-client
autologin-user-timeout=0
user-session=xfce
" | sudo tee /etc/lightdm/lightdm.conf.d/50-myconfig.conf

  sudo debconf-set-selections <<< "lightdm shared/default-x-display-manager select lightdm"
  sudo dpkg-reconfigure -f noninteractive lightdm
  sudo chown -R client:client /home/client

  SHELL
end

```

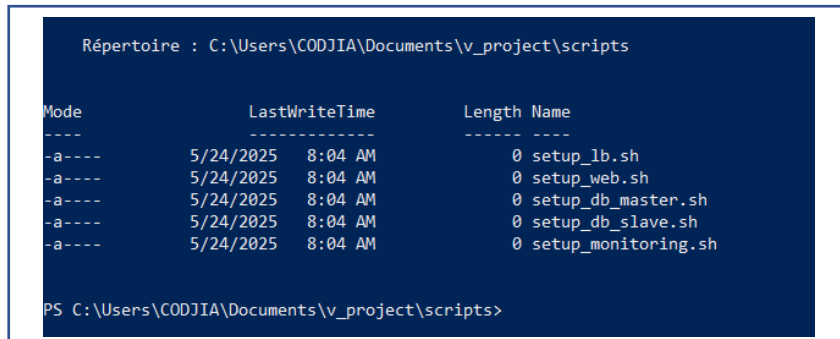
3.3 Écriture et organisation des scripts de provisioning Bash pour chaque rôle

Notre dossier scripts doit contenir les fichiers suivants :

- scripts/
- |—— setup_db_slave.sh
- |—— setup_db_master.sh
- |—— setup_web.sh
- |—— setup_monitoring.sh
- |—— setup_lb.sh

Pour le faire, nous allons d'abord créer tous les fichiers .sh par cette commande dans notre PowerShell :

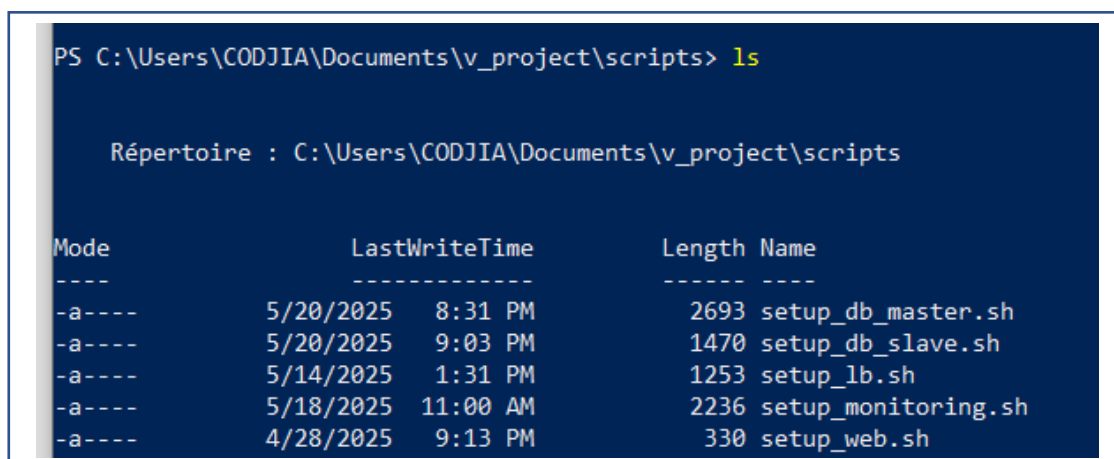
```
foreach ($file in  
"setup_lb.sh","setup_web.sh","setup_db_master.sh","setup_db_slave.sh","setup_monitoring.s  
h") { New-Item -ItemType File -Name $file -Force}
```



```
Répertoire : C:\Users\CODJIA\Documents\v_project\scripts  
  
Mode                LastWriteTime         Length Name  
----                -  
-a-----         5/24/2025   8:04 AM              0 setup_lb.sh  
-a-----         5/24/2025   8:04 AM              0 setup_web.sh  
-a-----         5/24/2025   8:04 AM              0 setup_db_master.sh  
-a-----         5/24/2025   8:04 AM              0 setup_db_slave.sh  
-a-----         5/24/2025   8:04 AM              0 setup_monitoring.sh  
  
PS C:\Users\CODJIA\Documents\v_project\scripts>
```

Figure 7 : répertoire scripts contenant nos dossiers

Maintenant, nous allons injecter directement le contenu de chaque script dans les fichiers .sh via PowerShell.



```
PS C:\Users\CODJIA\Documents\v_project\scripts> ls  
  
Répertoire : C:\Users\CODJIA\Documents\v_project\scripts  
  
Mode                LastWriteTime         Length Name  
----                -  
-a-----         5/20/2025   8:31 PM         2693 setup_db_master.sh  
-a-----         5/20/2025   9:03 PM         1470 setup_db_slave.sh  
-a-----         5/14/2025   1:31 PM         1253 setup_lb.sh  
-a-----         5/18/2025  11:00 AM         2236 setup_monitoring.sh  
-a-----         4/28/2025   9:13 PM          330 setup_web.sh
```

Figure 8 : Les scripts sont remplis avec nos contenues

4. Déploiement de l'infrastructure

4.1 Lancement de l'infrastructure avec vagrant up

Nous avons ouvert notre PowerShell et nous sommes rendus dans notre dossier « **PS C:\Users\CODJIA\Documents\v_project>** » puis exécuter la commande « **vagrant up** » cela va Créer et démarrer chaque machine virtuelle définie dans notre Vagrantfile et Configura automatiquement chaque VM avec son script associé (installation, configuration réseau, services, etc.).

4.2 Vérification de l'état des machines (vagrant status)

```
PS C:\Users\CODJIA\Documents\v_project\scripts> vagrant status
Current machine states:

lb                running (virtualbox)
web1              running (virtualbox)
web2              running (virtualbox)
db-master         running (virtualbox)
db-slave          running (virtualbox)
monitoring        running (virtualbox)
```

Figure 9 : Tous les machine sont démarrées sans problème

4.3 Vérification de la connectivité réseau (ping, ssh, host)

```
Envoi d'une requête 'Ping' 192.168.56.10 avec 32 octets de données :
Réponse de 192.168.56.10 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.10 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.10 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.10 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.56.10:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 0ms, Moyenne = 0ms
PS C:\Users\CODJIA\Documents\v_project\scripts> ping 192.168.56.11
Envoi d'une requête 'Ping' 192.168.56.11 avec 32 octets de données :
Réponse de 192.168.56.11 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.11 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.11 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.11 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.56.11:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 0ms, Moyenne = 0ms
PS C:\Users\CODJIA\Documents\v_project\scripts> ping 192.168.56.12
Envoi d'une requête 'Ping' 192.168.56.12 avec 32 octets de données :
Réponse de 192.168.56.12 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.12 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.12 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.12 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.56.12:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 1ms, Moyenne = 0ms
PS C:\Users\CODJIA\Documents\v_project\scripts> ping 192.168.56.13
Envoi d'une requête 'Ping' 192.168.56.13 avec 32 octets de données :
Réponse de 192.168.56.13 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.13 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.13 : octets=32 temps<1ms TTL=64
Réponse de 192.168.56.13 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.56.13:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 0ms, Moyenne = 0ms
PS C:\Users\CODJIA\Documents\v_project\scripts>
```

- Les vérifications de la connexion par ssh

vagrant@web1:~\$

vagrant@web2:~\$

vagrant@lb:~\$

vagrant@db-master:~\$

vagrant@db-slave:~\$

5. Configuration des services

5.1. Serveurs Web (web1, web2)

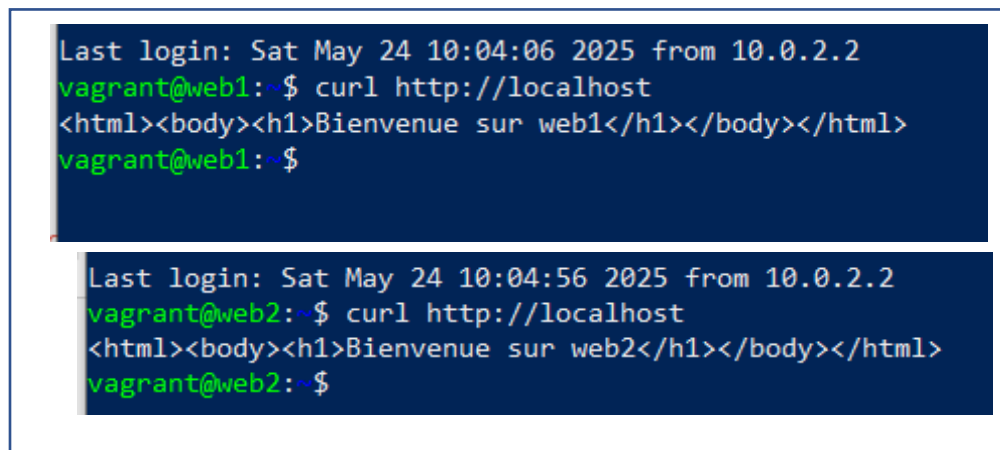
Les serveurs web constituent la couche applicative de l'infrastructure. Ils sont directement sollicités par le Load Balancer **lb** et doivent être fonctionnels en permanence. Grâce à la duplication de service sur **web1** et **web2**, le système garantit une meilleure résilience et une répartition de la charge.

5.1.1 Installation d'Apache

Chaque machine **web1** et **web2** exécute le script de provisionnement suivant (**setup_web.sh**) déjà créé.

5.1.2 Déploiement d'une page d'une page de test personnalisée

On va ensuite ajouter une page HTML personnalisée. Ce qui revient à écrire dans le fichier **setup_web.sh** :



```
Last login: Sat May 24 10:04:06 2025 from 10.0.2.2
vagrant@web1:~$ curl http://localhost
<html><body><h1>Bienvenue sur web1</h1></body></html>
vagrant@web1:~$

Last login: Sat May 24 10:04:56 2025 from 10.0.2.2
vagrant@web2:~$ curl http://localhost
<html><body><h1>Bienvenue sur web2</h1></body></html>
vagrant@web2:~$
```

Figure 11 : affichage des contenus de nos pages web

5.2 Load Balancer

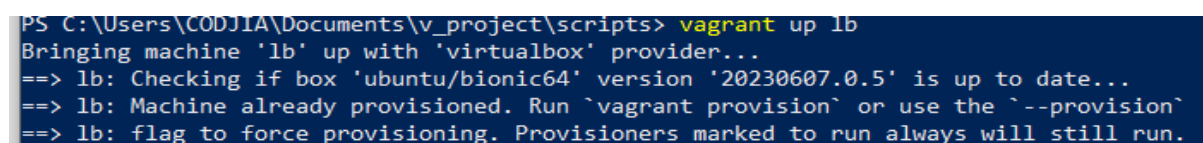
Le serveur **lb** (Load Balancer) utilise Nginx, un serveur HTTP performant capable de faire office de répartiteur de charge (reverse proxy).

5.2.1 Installation et configuration de Nginx

Le script de provisionnement **setup_lb.sh** déjà créé permet d'installer et configurer automatiquement Nginx.

Ensuite lançons la machine **lb** avec **vagrant up lb** pour vérification.

Une fois la machine **lb** lancée avec **vagrant up lb** on n'a :



```
PS C:\Users\CODJIA\Documents\v_project\scripts> vagrant up lb
Bringing machine 'lb' up with 'virtualbox' provider...
==> lb: Checking if box 'ubuntu/bionic64' version '20230607.0.5' is up to date...
==> lb: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> lb: flag to force provisioning. Provisioners marked to run always will still run.
```

Figure 12 : Lb en fonctionnement

5.2.2 Mise en œuvre et vérification de l'équilibrage de charge

Vérification du bon fonctionnement de notre lb en se connectant à travers son IP : 192.168.56.10.

Automatiquement on n'est connecté sur le web2 :

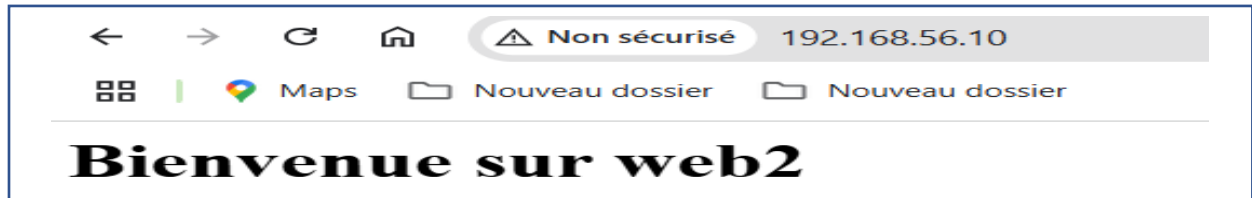


Figure 13 : Interface graphique du serveur Web 2

En actualisant la page web on passe au web1 :

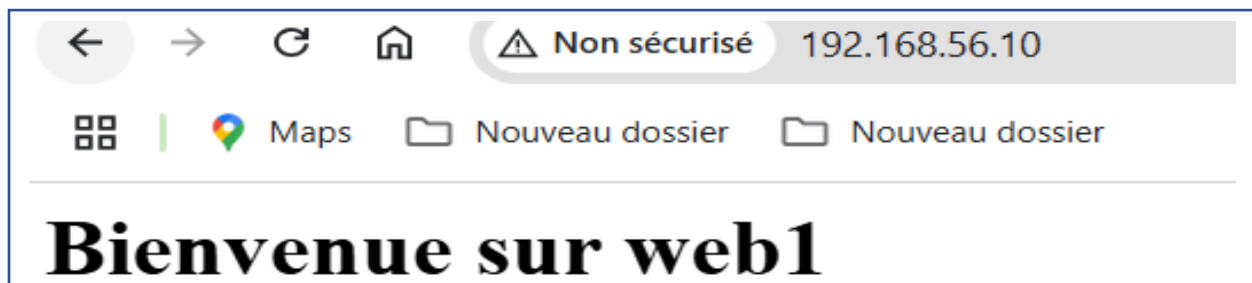


Figure 14 : Interface graphique du serveur Web 1

Test de résilience

Nous avons testé la résilience du système en éteignant web1 et en vérifiant que l'application reste disponible via le load balancer car lorsque le serveur web 1 est éteint le serveur 2 continu de travailler avec le load balancer.

5.3 Infrastructure MySQL

Déploiement de la base de données db-master et db-slave dans le Vagrantfile

```
# Machine 4 : DB Master
config.vm.define "db-master" do |db_master|
  db_master.vm.hostname = "db-master.local"
  db_master.vm.network "private_network", ip: "192.168.56.13"
  db_master.vm.provider "virtualbox" do |vb|
    vb.name = "db-master"
    vb.memory = 1024
  end
  db_master.vm.synced_folder "C:/Users/CODJIA/Documents/v_project", "/home/vagrant/shared"
  db_master.vm.provision "shell", path: "scripts/setup_db_master.sh"
end

# Machine 5 : DB Slave
config.vm.define "db-slave" do |slave|
  slave.vm.hostname = "db-slave.local"
  slave.vm.network "private_network", ip: "192.168.56.14"
  slave.vm.provider "virtualbox" do |vb|
    vb.name = "db-slave"
    vb.memory = 1024
  end
  slave.vm.provision "shell", path: "scripts/setup_db_slave.sh"
end
```

5.3.1 Installation de mysql sur db-master et db-slave

Pendant le déploiement des serveurs db-master et db-slave mysql serveur est installé avec les variables ci-dessous qui se trouve dans le fichier « mysql_dump_transfer.sh » du dossier dossier sql

```
# Variables
MYSQL_ROOT_PASSWORD="@dmin25"
REPLICATION_USER="user_replica"
REPLICATION_PASSWORD="user@replica"
BIND_ADDRESS="0.0.0.0"
DB_NAME="projetdb"
CONFIG_FILE="/etc/mysql/mysql.conf.d/mysqld.cnf"
```

Le mot de passe du super administrateur mysql avec tous les privilèges

Un utilisateur de réplication et son mot de passe associé qui sera utilisé par le slave pour se connecter au master et recevoir les mises à jour.

L'adresse **0.0.0.0** signifie que MySQL acceptera les connexions depuis toutes les interfaces réseau, y compris les autres machines. Utile pour permettre au slave d'accéder au master à distance.

Le nom de la base de données qui sera créée : projetdb

a) Création de la base de données projetdb sur db-master

Après l'installation de mysql avec tout le paramétrage nécessaire nous avons procédé à la création de la base de données projetdb sur dB-master avec différentes tables.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| projetdb |
| sys |
+-----+
5 rows in set (0.00 sec)
```

Figure 15 : projetdb

Nous avons ensuite copié le contenu de notre fichier « script_sql_projet.sql » présent dans notre dossier sql.

Ce script crée une base de données relationnelle simple et cohérente :

- 1 Professeur enseigne plusieurs cours.
- 1 Étudiant peut suivre plusieurs cours.
- **ETUDIANT_COURS** fait le lien entre eux avec des infos utiles (paiement, note...).

Étape 1 : dans le répertoire où se trouve notre fichier nous avons exécuté notre commande : **mysql -u root -p projetdb < /vagrant/scripts/script_sql_projet.sql**

On n'est invité à entrer le mot de passe : @dmin25

Cela va créer toutes les tables et insérer toutes les données en une seule fois.

Maintenant, il faut se connecter à mysql du db-master pour vérifier si les données ont été enregistrer.

b) Test pour s'assurer que l'insertion d'étudiants dans la table étudiant fonctionne

```
mysql> SELECT * FROM ETUDIANT;
+-----+-----+-----+-----+-----+
| N_ETUD | NOM_ETUD | PRENOM_ETUD | DATE_INSCRIP | EMAIL |
+-----+-----+-----+-----+-----+
| 1 | Zoundi | philippe | 2024-09-01 | philippe.zoundi@example.com |
| 2 | moreno | codja | 2024-09-02 | codja.moreno@example.com |
| 3 | bendja | Sophie | 2024-09-02 | sophie.bendja@example.com |
| 4 | Ouedraogo | rapouyouga | 2024-09-02 | rapougouya.ouedraogo@example.com |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Figure 16 : Affichage de la table actuelle

c) Insertion d'un nouvel étudiant

Pour ajouter cet étudiant à ta table ETUDIANT, on va exécuter cette requête SQL dans MySQL :

```
INSERT INTO ETUDIANT (N_ETUD, NOM_ETUD, PRENOM_ETUD, DATE_INSCRIP, EMAIL)
VALUES (5, 'KONE', 'JACOB', '2025-05-20', 'jacob.kone@example.com');
```

```
mysql> SELECT * FROM ETUDIANT;
```

N_ETUD	NOM_ETUD	PRENOM_ETUD	DATE_INSCRIP	EMAIL
1	Zoundi	philippe	2024-09-01	philippe.zoundi@example.com
2	moreno	codja	2024-09-02	codja.moreno@example.com
3	bendja	Sophie	2024-09-02	sophie.bendja@example.com
4	Ouedraogo	rapouyouga	2024-09-02	rapougouya.ouedraogo@example.com
5	KONE	JACOB	2025-05-20	jacob.kone@example.com

```
5 rows in set (0.00 sec)

mysql>
```

Figure 17 : Insertion d'un nouvel étudiant

5.3.2 Mise en place de la réplication maître-esclave (Configuration, utilisateurs, synchronisation)

Le chemin vers le fichier de configuration de mysql.

/etc/mysql/mysql.conf.d/mysqld.cnf

```
vagrant@db-master:/etc/mysql/mysql.conf.d$ mysql --version
mysql Ver 14.14 Distrib 5.7.42, for Linux (x86_64) using Editline wrapper
vagrant@db-master:/etc/mysql/mysql.conf.d$
```

Figure 18: capture mysql installé

Configurer le fichier par cette commande

```
vagrant@db-master:~$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

5.3.2.1 Pour la réplication MySQL, voici les prochaines lignes à ajouter dans ce même fichier

[mysqld]

server-id = 1

log_bin = /var/log/mysql/mysql-bin.log

binlog_do_db = nom_de_ta_base

bind-address = 0.0.0.0

```
[mysqld]
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
binlog_do_db = projetdb
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
binlog_do_db = projetdb
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
binlog_do_db = projetdb
#
# * Basic Settings
#
user                = mysql
pid-file            = /var/run/mysqld/mysqld.pid
socket              = /var/run/mysqld/mysqld.sock
port                = 3306
basedir             = /usr
datadir             = /var/lib/mysql
tmpdir              = /tmp
lc-messages-dir     = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address = 0.0.0.0
```

Figure 19 : Ecoute sur toutes les interfaces réseaux

Cette ligne permet à MySQL d'écouter sur toutes les interfaces réseau, ce qui est nécessaire pour que les autres machines (comme le **db-slave**) puissent se connecter pour la réplication.

On fera les mêmes configurations sur le db-slave

```
[mysqld]
server-id = 2
relay-log = /var/log/mysql/mysql-relay-bin.log
server-id = 2
```

Figure 20 : Capture fichier de config mysql sur db-slave

NB :

Le server-id de **db-master**=1

Le server-id de **db-slave**=2

Ce qui montre le maître et l'esclave

5.3.2.2 On doit redémarrer les deux serveurs pour valider la réplication et vérifier si la réplication marche :

```
vagrant@db-master:~$ sudo systemctl restart mysql

vagrant@db-master:~$
vagrant@db-master:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.42-0ubuntu0.18.04.1-log (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| mysql-bin.000001 |      154 | projetdb     |                   |                   |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Figure 21 : Vérification de la réplication

Le **SHOW MASTER STATUS;** affiche désormais :

- File : **mysql-bin.000001** → le fichier de log binaire actif
- Position : **154** → la position actuelle dans le fichier binaire
- Binlog_Do_DB : **projetdb** → seule cette base est répliquée

J'ai aussi changé le mot de passe de la réplication sur ma base de données.

CHANGE MASTER TO

MASTER_HOST='192.168.56.13',

MASTER_USER='user_replica',

MASTER_PASSWORD='replica_pass',

MASTER_LOG_FILE='mysql-bin.000001',

MASTER_LOG_POS=154;

La réplication fonctionne correctement maintenant. Voici les signes qui confirment cela :

```
mysql> SHOW SLAVE STATUS\G
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: 192.168.56.13
      Master_User: user_replica
      Master_Port: 3306
      Connect_Retry: 60
      Master_Log_File: mysql-bin.000002
      Read_Master_Log_Pos: 154
      Relay_Log_File: mysql-relay-bin.000003
      Relay_Log_Pos: 367
      Relay_Master_Log_File: mysql-bin.000002
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
      Seconds_Behind_Master: 0
      Master_SSL_Enabled: 0
```

Figure 22 : La réplication fonctionne

Voici les signes qui confirment cela :

Éléments clés à vérifier :

- **Slave_IO_Running: Yes** → La connexion au master est active.
- **Slave_SQL_Running: Yes** → Le slave applique bien les événements.
- **Slave_IO_State: Waiting for master to send event** → Tout est synchronisé, il attend simplement de nouveaux événements.
- **Seconds_Behind_Master: 0** → Le slave est à jour avec le master.

Sur le **master**, on va se connecter à MySQL :

```
mysql -u root -p
```

Et on insère des données dans la base projetdb, par exemple :

```
USE projetdb;
CREATE TABLE test_replication (id INT PRIMARY KEY, nom VARCHAR(50));
INSERT INTO test_replication VALUES (1, 'Test depuis master');
```

5.3.2.3 Sauvegarde de la configuration de la base de données du db-master et transfert vers le db-salve

- ✓ Exécuter la commande sur le db-master

```
FLUSH TABLES WITH READ LOCK;
```

Pour verrouiller temporairement les tables en lecture seule et empêcher de nouvelles écritures pendant qu'on fait une copie (dump) de la base de données.

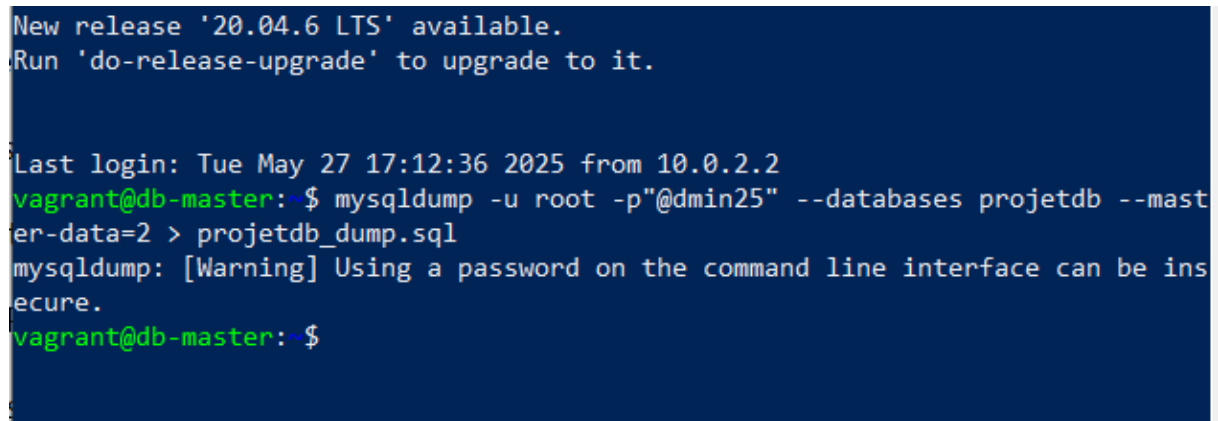
5.3.2.4 Exécution du script mysql_dump_transfer.sh

Étape 1 : Lançons la sauvegarde (mysqldump) pendant que le verrou est actif

Dans notre machine **db-master**, sur un autre terminal (ou session SSH), exécutons le script (ou la commande) qui va faire le dump :

```
mysqldump -u root -p"@dmin25" --databases projetdb --master-data=2 >
projetdb_dump.sql
```

- Cette commande crée un fichier SQL avec les données de notre base verrouillée.
- L'option **--master-data=2** ajoute dans le dump les informations de position de la réplication.



```
New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

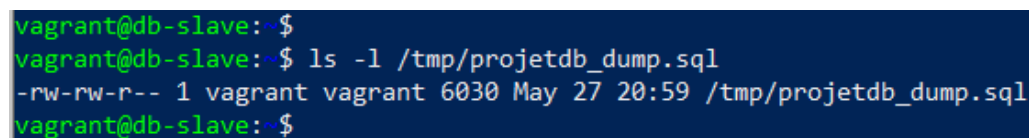
Last login: Tue May 27 17:12:36 2025 from 10.0.2.2
vagrant@db-master:~$ mysqldump -u root -p"@dmin25" --databases projetdb --master-data=2 > projetdb_dump.sql
mysqldump: [Warning] Using a password on the command line interface can be insecure.
vagrant@db-master:~$
```

Figure 23 : fichier sql crée

Étape 2 : Transférer le fichier dump vers le serveur esclave (db-slave)

Toujours dans un terminal (sur le db-master), lançons :

```
scp projetdb_dump.sql user@192.168.56.14:/tmp/
```



```
vagrant@db-slave:~$
vagrant@db-slave:~$ ls -l /tmp/projetdb_dump.sql
-rw-rw-r-- 1 vagrant vagrant 6030 May 27 20:59 /tmp/projetdb_dump.sql
vagrant@db-slave:~$
```

Cela va créer un dump SQL propre et à jour qui sera transféré vers l'esclave, et va inclure automatiquement les bonnes infos de réplication.

Dans le répertoire du projet il s'agit de projetdb_dump.sql

5.3.2.5 Test de réplication sur le db-master et le db-slave

Nous allons vérifier que l'insertion de l'étudiant SAWADOGO Arouna dans db-master a été répliquée sur db-slave en tapant cette commande dans db-master pour insérer Arouna :

```
INSERT INTO ETUDIANT (N_ETUD, NOM_ETUD, PRENOM_ETUD, DATE_INSCRIP, EMAIL)
```

```
VALUES (6, 'SAWADOGO', 'Arouna', CURDATE(), 'arouna.sawadogo@example.com');
```

```
vagrant@db-master: ~
mysql> K, 1 row affected (0.08 sec)
mysql>
mysql> INSERT INTO ETUDIANT (N_ETUD, NOM_ETUD, PRENOM_ETUD, DATE_INSCRIP, EMAIL) VALUES (6, 'SAWADOGO', 'Arouna', CURDATE(), 'arouna.sawadogo@example.com');
mysql>
mysql> select * from ETUDIANT
-> ;
+-----+-----+-----+-----+-----+
| N_ETUD | NOM_ETUD | PRENOM_ETUD | DATE_INSCRIP | EMAIL |
+-----+-----+-----+-----+-----+
| 1 | Zoundi | philippe | 2024-09-01 | philippe.zoundi@example.com |
| 2 | moreno | codja | 2024-09-02 | codja.moreno@example.com |
| 3 | bendja | Sophie | 2024-09-02 | sophie.bendja@example.com |
| 4 | Ouedraogo | rapouyouga | 2024-09-02 | rapougouya.ouedraogo@example.com |
| 5 | KONE | JACOB | 2025-05-20 | jacob.kone@example.com |
| 6 | SAWADOGO | Arouna | 2025-05-27 | arouna.sawadogo@example.com |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 24 : Insertion d'un nouvel étudiant

Vérifions notre réplication sur db-slave maintenant

On va se connecter à db-slave et taper cette commande

```
SELECT * FROM ETUDIANT WHERE NOM_ETUD='SAWADOGO' AND PRENOM_ETUD='Arouna';
```

La réplication fonctionne bien car on :

```
vagrant@db-slave: ~
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 1
Master_UUID: 778ba09f-39fc-11f0-bd8d-023b7bb73b2d
Master_Info_File: /var/lib/mysql/master.info
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
Master_Retry_Count: 86400
Master_Bind:
Last_IO_Error_Timestamp:
Last_SQL_Error_Timestamp:
Master_SSL_Crl:
Master_SSL_Crlpath:
Retrieved_Gtid_Set:
Executed_Gtid_Set:
Auto_Position: 0
Replicate_Rewrite_DB:
Channel_Name:
Master_TLS_Version:
1 row in set (0.00 sec)

mysql> SELECT * FROM ETUDIANT WHERE NOM_ETUD='SAWADOGO' AND PRENOM_ETUD='Arouna';
+-----+-----+-----+-----+-----+
| N_ETUD | NOM_ETUD | PRENOM_ETUD | DATE_INSCRIP | EMAIL |
+-----+-----+-----+-----+-----+
| 6 | SAWADOGO | Arouna | 2025-05-27 | arouna.sawadogo@example.com |
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 25 : test de réplication

5.4 Supervision et Monitoring

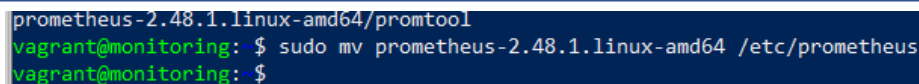
5.4.1 Installation de Prometheus et Node Exporter

L'outil de monitoring Prometheus, couplé à Node Exporter, a été déployé sur la machine dédiée au monitoring (IP : 192.168.56.15).

Voici les étapes principales de notre installation :

5.4.1.1 Télécharger et installer Prometheus à travers cette commande :

```
wget https://github.com/prometheus/prometheus/releases/download/v2.48.1/prometheus-2.48.1.linux-amd64.tar.gz
tar -xvzf prometheus-2.48.1.linux-amd64.tar.gz
sudo mv prometheus-2.48.1.linux-amd64 /etc/prometheus
```

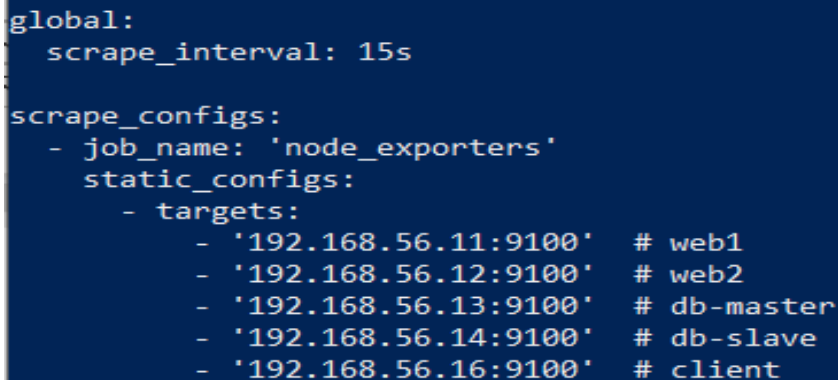


```
prometheus-2.48.1.linux-amd64/promtool
vagrant@monitoring: $ sudo mv prometheus-2.48.1.linux-amd64 /etc/prometheus
vagrant@monitoring: $
```

5.4.1.2 Créer un fichier de configuration prometheus.yml et l'éditer avec nano :

```
Sudo nano /etc/prometheus/prometheus.yml
```

Et y insérer quelque chose comme :



```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporters'
    static_configs:
      - targets:
        - '192.168.56.11:9100' # web1
        - '192.168.56.12:9100' # web2
        - '192.168.56.13:9100' # db-master
        - '192.168.56.14:9100' # db-slave
        - '192.168.56.16:9100' # client
```

Figure 26 : fichier de configuration *prometheus.yml*

5.4.1.3 Lancer Prometheus

Installons d'abord *Node Exporter* sur tous les machines avec :

Pour cela, nous avons ajouté un **script de provisioning** pour que Node Exporter soit installé dès le lancement des machines.

Nous pouvons vérifier la présence du Node Exporter sur chaque machine par cette commande :

```
ls /usr/local/bin/node_exporter
```

```
vagrant@db-slave:~$ ls /usr/local/bin/node_exporter
/usr/local/bin/node_exporter
vagrant@db-slave:~$
```

Lançons Prometheus

/usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml

```
vagrant@monitoring:/etc/prometheus$ sudo systemctl restart prometheus
vagrant@monitoring:/etc/prometheus$ sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2025-05-28 19:38:48 UTC; 46s ago
     Main PID: 9374 (prometheus)
        Tasks: 9 (limit: 1151)
      CGroup: /system.slice/prometheus.service
              └─9374 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/var/lib/p

May 28 19:38:48 monitoring prometheus[9374]: ts=2025-05-28T19:38:48.761Z caller=main.go:1150 level=info fs_type=EXT4_SU
May 28 19:38:48 monitoring prometheus[9374]: ts=2025-05-28T19:38:48.762Z caller=main.go:1153 level=info msg="TSDB start
May 28 19:38:48 monitoring prometheus[9374]: ts=2025-05-28T19:38:48.762Z caller=main.go:1335 level=info msg="Loading co
May 28 19:38:48 monitoring prometheus[9374]: ts=2025-05-28T19:38:48.766Z caller=main.go:1372 level=info msg="Completed
May 28 19:38:48 monitoring prometheus[9374]: ts=2025-05-28T19:38:48.766Z caller=main.go:1114 level=info msg="Server is
May 28 19:38:48 monitoring prometheus[9374]: ts=2025-05-28T19:38:48.767Z caller=manager.go:163 level=info component="ru
May 28 19:38:56 monitoring prometheus[9374]: ts=2025-05-28T19:38:56.229Z caller=compact.go:567 level=info component=tsd
May 28 19:38:56 monitoring prometheus[9374]: ts=2025-05-28T19:38:56.239Z caller=head.go:1345 level=info component=tsdb
May 28 19:38:56 monitoring prometheus[9374]: ts=2025-05-28T19:38:56.241Z caller=checkpoint.go:101 level=info component=
May 28 19:38:56 monitoring prometheus[9374]: ts=2025-05-28T19:38:56.278Z caller=head.go:1307 level=info component=tsdb
lines 1-18/18 (END)
```

Sur le monitoring, nous avons tapé « sudo apt-get install -y grafana »

```
vagrant@monitoring:/etc/prometheus$ # Installe Grafana
vagrant@monitoring:/etc/prometheus$ sudo apt-get install -y grafana
Reading package lists... Done
Building dependency tree
Reading state information... Done
grafana is already the newest version (12.0.1).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
vagrant@monitoring:/etc/prometheus$
vagrant@monitoring:/etc/prometheus$ # Démarre le service Grafana
vagrant@monitoring:/etc/prometheus$ sudo systemctl start grafana-server
vagrant@monitoring:/etc/prometheus$
vagrant@monitoring:/etc/prometheus$ # Active le service pour qu'il démarre automatiquement au boot
vagrant@monitoring:/etc/prometheus$ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
vagrant@monitoring:/etc/prometheus$
vagrant@monitoring:/etc/prometheus$ # Vérifie que Grafana tourne bien
vagrant@monitoring:/etc/prometheus$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2025-05-28 18:11:36 UTC; 1h 54min ago
     Docs: http://docs.grafana.org
     Main PID: 827 (grafana)
        Tasks: 13 (limit: 1151)
      CGroup: /system.slice/grafana-server.service
              └─827 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-
```

Pour accéder aux interfaces web :

1. Depuis notre client (l'ordinateur hôte, pas la VM), on ouvre un navigateur.
2. On tape dans la barre d'adresse :

<http://192.168.56.15:9090/targets> Pour Prometheus :

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.56.12:9100/metrics	UP	instance="192.168.56.12:9100" job="nodes"	7.61s ago	43.790ms	
http://192.168.56.13:9100/metrics	UP	instance="192.168.56.13:9100" job="nodes"	14.997s ago	113.911ms	
http://192.168.56.14:9100/metrics	UP	instance="192.168.56.14:9100" job="nodes"	466.000ms ago	93.443ms	
http://192.168.56.15:9100/metrics	UP	instance="192.168.56.15:9100" job="nodes"	12.474s ago	76.365ms	
http://192.168.56.10:9100/metrics	UP	instance="192.168.56.10:9100" job="nodes"	2.509s ago	35.167ms	
http://192.168.56.11:9100/metrics	UP	instance="192.168.56.11:9100" job="nodes"	1.961s ago	23.775ms	

prometheus (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
----------	-------	--------	-------------	-----------------	-------

Figure 27 : interface de prometheus

Requête sur Prometheus pour avoir des informations sur le metrique consommation de CPU

$100 * (1 - avg(rate(node_cpu_seconds_total\{mode="idle"\} [5m])) by (host))$

- Pour Grafana (une fois installé et démarré) :

http://192.168.56.15:3000

5.4.1.4 Configuration initiale de Grafana

- Accède à l'interface web Grafana :
http://<IP_de_ton_serveur>:3000
- Connecte-toi avec le login par défaut :
 - user: admin
 - password: admin

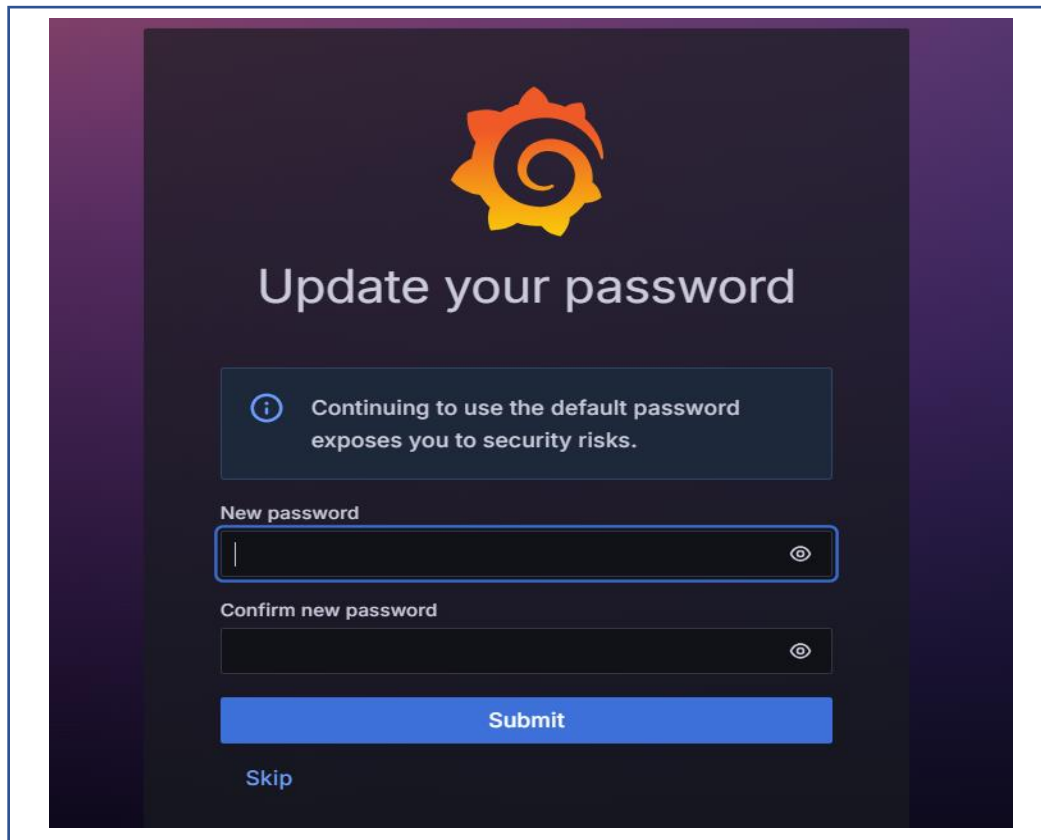


Figure 28 : Interface gafrana

5.4.1.5 Création d'un dashboard de visualisation des métriques

Une fois Grafana installé et lancé, nous avons procédé à la création d'un dashboard personnalisé pour surveiller les performances de nos machines virtuelles.

Étapes réalisées :

1. Ajout de la source de données Prometheus :
 - Depuis l'interface Grafana, aller dans "Configuration > Data Sources".
 - Ajouter une nouvelle source de données de type Prometheus.
 - Renseigner l'URL de Prometheus : <http://localhost:9090>.

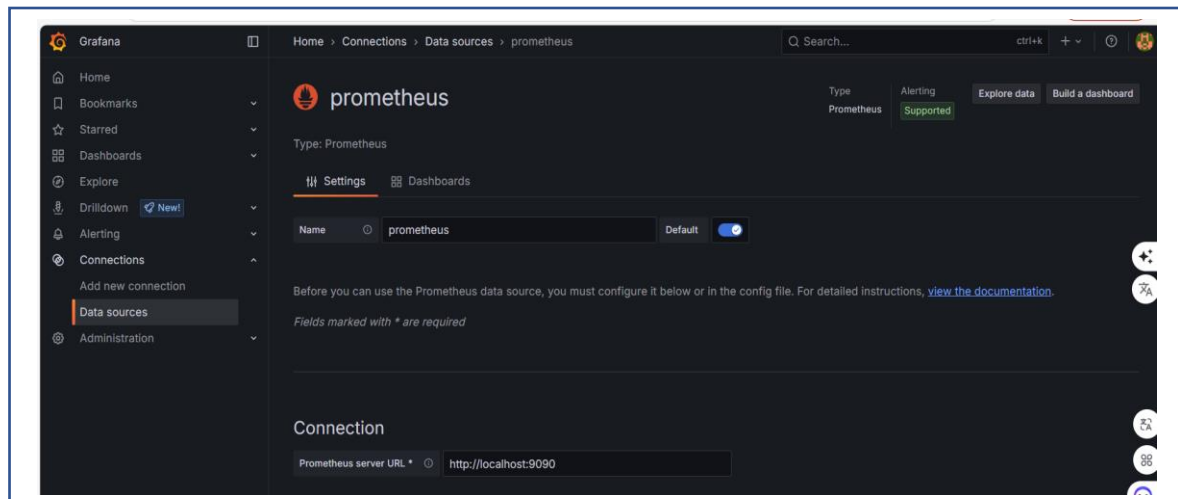


Figure 29 : création de source de données

5.4.1.6 Création d'un nouveau dashboard :

Créer un dashboard personnalisé

Dans Grafana → menu de gauche → "+" → Dashboard

Clique sur Add new panel

Dans le champ Query, entre une métrique comme :

- **node_cpu_seconds_total**
- **node_memory_MemAvailable_bytes**
- **node_network_receive_bytes_total**

Choix du type de visualisation : graph, gauge, bar gauge, etc.

Clique sur Apply

Répète pour d'autres panels (CPU, RAM, etc.)

Clique sur Save dashboard

- Cliquer sur "Create > Dashboard".
- Ajouter un nouveau panneau (panel).

Ajout de panels avec des métriques pertinentes :

- CPU usage :
- Requête PromQL :

rate(node_cpu_seconds_total{mode="user"}[1m])

- Utilisation de la RAM :

(node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes

- Espace disque disponible :

node_filesystem_avail_bytes / node_filesystem_size_bytes

Personnalisation :

- Choix de types de visualisation (graph, gauge, bar gauge, stat, etc.).
- Attribution de noms, unités (% , MB, etc.), couleurs et seuils pour chaque panel.

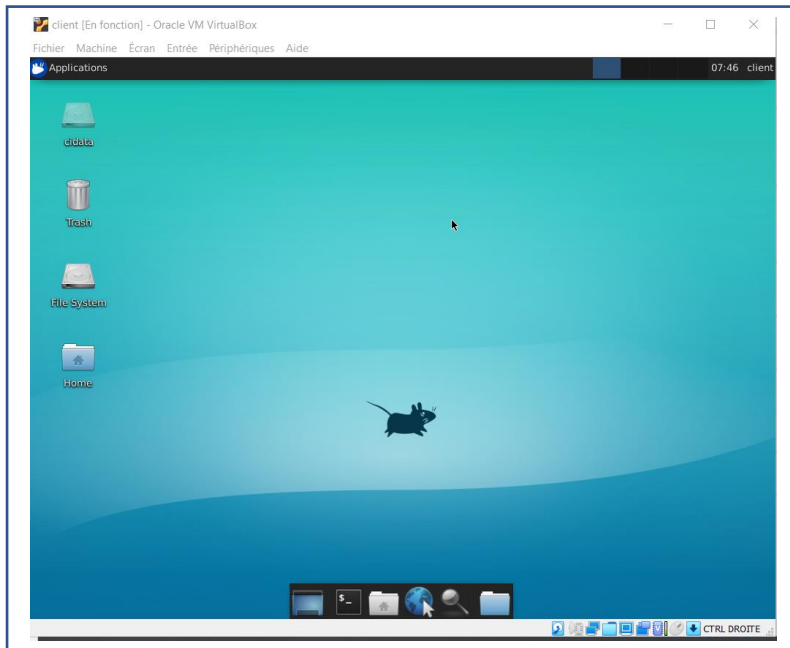
Sauvegarde du dashboard :

- Nom du dashboard : **Monitoring Infrastructure**.
- Partage possible par lien ou export JSON.

Grâce à cette mise en place, nous disposons désormais d'un système de supervision visuel et réactif, permettant de détecter en temps réel les variations de performances ou les incidents potentiels au sein de notre infrastructure.

5.5 Tests et validation avec le client

5.5.1 Test sur Interface de connexion du client



5.5.2 Test de répartition de charge

Curl `http://192.168.56.10` pour se connecter sur le lb et vérifier le fonctionnement de la répartition des charges entre web1 et web2

```
clientuser@client:~$ curl http://192.168.56.10
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web1.</h1>
    <p>Adresse IP = 192.168.56.11</p>
  </body>
</html>
clientuser@client:~$ curl http://192.168.56.10
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web2.</h1>
    <p>Adresse IP = 192.168.56.12</p>
  </body>
</html>
clientuser@client:~$
```

Figure 30 : la répartition de charge entre web1 et web2

5.5.3 Test sur les serveurs web

```
clientuser@client:~$ curl http://192.168.56.11
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web1.</h1>
    <p>Adresse IP = 192.168.56.11</p>
  </body>
</html>
clientuser@client:~$ █
```

Figure 31 : Contenu du server web1

```
clientuser@client:~$ curl http://192.168.56.12
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web2.</h1>
    <p>Adresse IP = 192.168.56.12</p>
  </body>
</html>
```

Figure 32 : Contenu du server web2

5.5.4 Test de connexion aux bases de données

Test de connexion sur db-master et db-slave

```
clientuser@client:~$ curl http://192.168.56.13
curl: (7) Failed to connect to 192.168.56.13 port 80: Connection refused
clientuser@client:~$ curl http://192.168.56.14
curl: (7) Failed to connect to 192.168.56.14 port 80: Connection refused
clientuser@client:~$ █
```

Figure 33 : Test de connexion sur db-master et db-slave

Les deux serveurs de base de données n'écotent pas sur le port 80 donc la commande Curl <http://192.168.56.13/14> est refusée

Nous allons effectuer le test sur le port d'écoute des deux bases de données à savoir le port 3306 avec la commande :

```
Nc -zv 192.168.56.13 3306
Nc -zv 192.168.56.14 3306
```



```

clientuser@client:~$ nc -zv 192.168.56.13 3306
Connection to 192.168.56.13 3306 port [tcp/mysql] succeeded!
clientuser@client:~$ nc -zv 192.168.56.14 3306
Connection to 192.168.56.14 3306 port [tcp/mysql] succeeded!
clientuser@client:~$ █

```

Figure 34 : Test sur le serveur

5.5.5 Test sur le serveur de monitoring

```

clientuser@client:~$ curl http://192.168.56.15

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2016-11-16
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>

```

Figure 35 : code html du site

Nous constatons que les tests de connectivité que les tests de connectivité sont concluants.

5.5.6 Test de remplissage du formulaire sur les serveurs web

Nous allons nous connecter à partir du client sur le serveur web1 et remplir un formulaire

En saisissant l'adresse <http://192.168.56.11/formulaire.php>

The image shows two screenshots of a web browser. The top screenshot displays a form titled "Ajout d'un étudiant(web1)" with the following fields: "Nom" (filled with "KOURAOGO"), "Prénom" (filled with "MOUASE"), "Email" (filled with "mouase.kouraogo@test.com"), and "Date d'inscription" (filled with "05 / 25 / 2025" and a calendar icon). There is an "Ajouter" button at the bottom. The bottom screenshot shows the same browser window with the URL "192.168.56.11/traitement.php" and the message "Étudiant ajouté avec succès !".

Figure 36 : Test de saisie

5.5.7 Test de résilience (arrêt d'un web ou du maître, observation du comportement)

```
clientuser@client:~$ ssh clientuser@192.168.56.13
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-212-generic x86_64)
```

```
clientuser@db-master:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.42-0ubuntu0.18.04.1-log (Ubuntu)
```

Nous avons éteint le serveur web2 pour effectuer le test

```
PS D:\projets_virtualisation> vagrant status
Current machine states:

lb               running (virtualbox)
web1             running (virtualbox)
web2             poweroff (virtualbox)
db-master        poweroff (virtualbox)
db-slave         poweroff (virtualbox)
monitoring       poweroff (virtualbox)
client           running (virtualbox)
```

```
clientuser@client:~$ curl http://192.168.56.10
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web1.</h1>
    <p>Adresse IP = 192.168.56.11</p>
  </body>
</html>
clientuser@client:~$ curl http://192.168.56.10
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web1.</h1>
    <p>Adresse IP = 192.168.56.11</p>
  </body>
</html>
clientuser@client:~$ █
```

Nous allons éteindre maintenant le serveur web1

```
PS D:\projets_virtualisation> vagrant status
Current machine states:

lb               running (virtualbox)
web1             poweroff (virtualbox)
web2             running (virtualbox)
db-master        poweroff (virtualbox)
db-slave         poweroff (virtualbox)
monitoring       poweroff (virtualbox)
client           running (virtualbox)
```

```

clientuser@client:~$ curl http://192.168.56.10
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web2.</h1>
    <p>Adresse IP = 192.168.56.12</p>
  </body>
</html>
clientuser@client:~$ curl http://192.168.56.10
<html>
  <body>
    <h1>Bonjour, vous etes sur le serveur web2.</h1>
    <p>Adresse IP = 192.168.56.12</p>
  </body>
</html>
clientuser@client:~$ █

```

En conclusion si l'un des serveurs web tombe en panne, l'autre reste toujours fonctionnel

5.5.8 Test de Vérification et de réplication du db-master et réplication sur db-slave

```
mysql> use projetdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * FROM ETUDIANT;
```

N_ETUD	NOM_ETUD	PRENOM_ETUD	DATE_INSCRIP	EMAIL
1	Zoundi	philippe	2024-09-01	philippe.zoundi@example.com
2	moreno	codja	2024-09-02	codja.moreno@example.com
3	bendja	Sophie	2024-09-02	sophie.bendja@example.com
4	Ouedraogo	rapouyouga	2024-09-02	rapougouya.ouedraogo@example.com
5	KONE	JACOB	2025-05-20	jacob.kone@example.com
6	ROUAMBA	OSEE	2025-05-21	ose.rouamba@example.com
7	BELEM	Abdou rasmane	2025-05-21	rasmane.belem@example.com
8	PALE	Lynda	2025-05-21	lynda.pale@example.com
9	Dupont	marcelus	2025-05-21	Test@example.com
10	TANKOANO	Abdou kader	2025-05-22	abdou.tankoano@example.com
11	KONE	Mimi	2025-05-22	mimi.kone@example.com
12	KABORE	ISSOUF	2025-05-22	issouf.kabore@example.com
13	TRAORE	Moussa	2025-05-22	moussa.traore@example.com
14	SAWADOGO	Arouna	2025-05-25	arouna.sawadogo@example.com
15	KOURAOGO	MOUASE	2025-05-25	mouase.kouraogo@test.com

On observe que kouraogo mouase a bien été ajouté à db-master

Maintenant observons sur le db-slave

```

clientuser@client:~$ ssh clientuser@192.168.56.14
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-213-generic x86_64)

```

```
clientuser@db-slave:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.42-0ubuntu0.18.04.1 (Ubuntu)

mysql> use projetdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * FROM ETUDIANT;
```

N_ETUD	NOM_ETUD	PRENOM_ETUD	DATE_INSCRIP	EMAIL
1	Zoundi	philippe	2024-09-01	philippe.zoundi@example.com
2	moreno	codja	2024-09-02	codja.moreno@example.com
3	bendja	Sophie	2024-09-02	sophie.bendja@example.com
4	Ouedraogo	rapouyouga	2024-09-02	rapougouya.ouedraogo@example.com
5	KONE	JACOB	2025-05-20	jacob.kone@example.com
6	ROUAMBA	OSEE	2025-05-21	ose.rouamba@example.com
7	BELEM	Abdou rasmane	2025-05-21	rasmane.belem@example.com
8	PALE	Lynda	2025-05-21	lynda.pale@example.com
9	Dupont	marcelus	2025-05-21	Test@example.com
10	TANKOANO	Abdou kader	2025-05-22	abdou.tankoano@example.com
11	KONE	Mimi	2025-05-22	mimi.kone@example.com
12	KABORE	ISSOUF	2025-05-22	issouf.kabore@example.com
13	TRAORE	Moussa	2025-05-22	moussa.traore@example.com
14	SAWADOGO	Arouna	2025-05-25	arouna.sawadogo@example.com
15	KOURAOGO	MOUASE	2025-05-25	mouase.kouraogo@test.com

Figure 37 : Réplication effective

Nous constatons que la réplication est effective sur db-slave pour kouraogo Mouase

5.5.9 Test de consommation des nœuds en pourcentage

Dans la fenêtre graph on a exécuté la commande up qui afficher 1 pour chaque cible en ligne.



Exécution de la commande : $100 * (1 - \text{avg}(\text{rate}(\text{node_cpu_seconds_total}\{\text{mode}=\text{"idle"}\}[5\text{m}]))$ by (instance)) pour obtenir la consommation en pourcentage des différents nœuds toutes les 5m.

☐ Use local time ☐ Enable query history ☒ Enable autocomplete ☒ Enable highlighting ☒ Enable linter

Q 100 * (1 - avg(rate(node_cpu_seconds_total{mode="idle"}[5m])) by (instance))

Table Graph

< Evaluation time >

No data queried yet

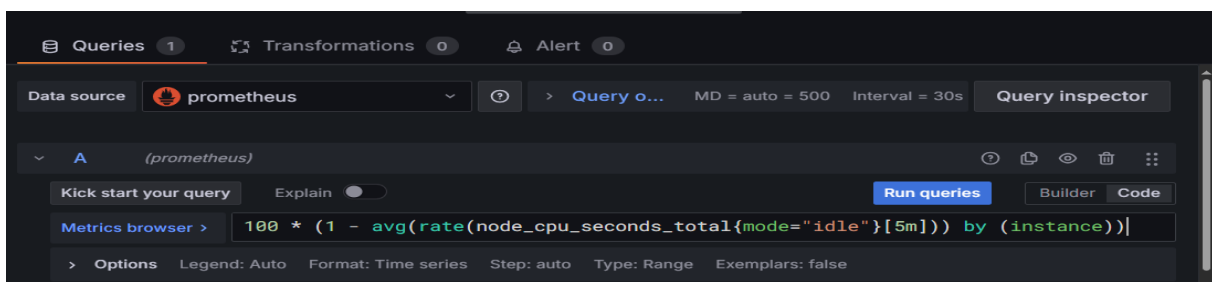
Dans cet exemple il s'agit du load balancer



Figure 38 : consommation en pourcentage des nœuds du load balancer

Exécution du commande:

100 * (1 - avg(rate(node_cpu_seconds_total{mode="idle"}[5m])) by (instance))



Elle donne le pourcentage moyen d'utilisation du CPU sur chaque machine surveillée, calculé sur les 5 dernières minutes.

Résultat de la commande de vérification de consommation après l'exécution

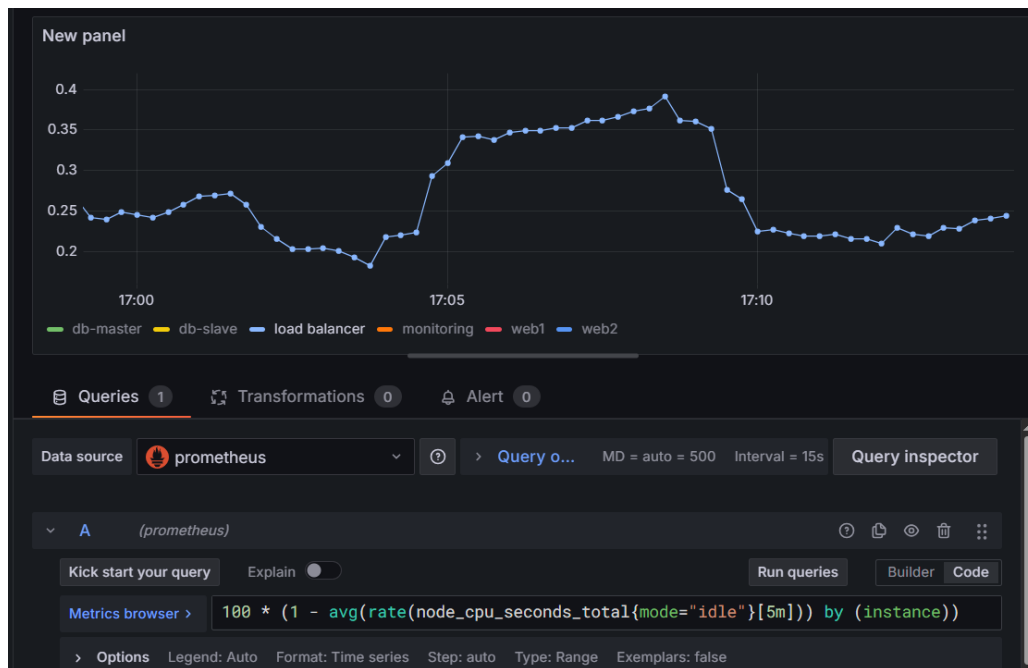


Figure 39 : consommation du CPU

5.5.10 Test de la consommation de la mémoire

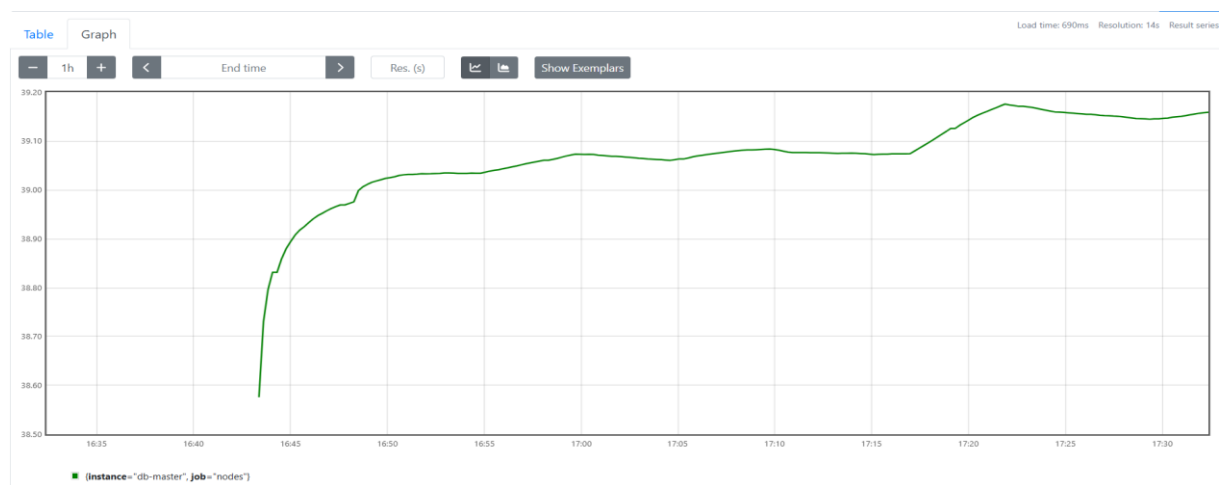
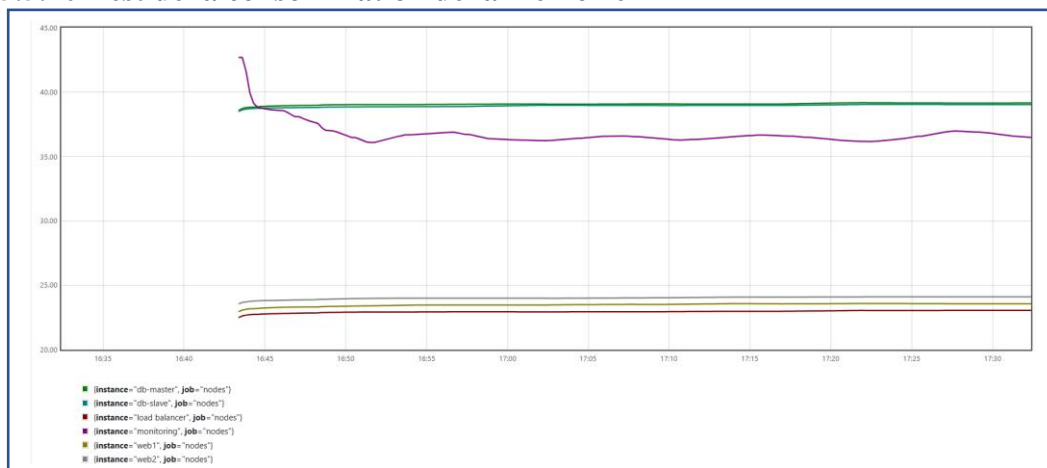


Figure 40 : Consommation de la mémoire pour le db master

6. Conclusion

6.1 Bilan du projet

La réalisation de ce projet d'infrastructure virtuelle multi-machines a permis de mettre en œuvre de manière concrète les concepts clés abordés en cours de virtualisation et cloud computing. Grâce à l'utilisation combinée de Vagrant pour l'orchestration et de Bash pour l'automatisation, nous avons réussi à construire une infrastructure complète, modulaire et fonctionnelle intégrant des composants critiques tels que la supervision, le load balancing et la réplication de base de données.

Cette approche a favorisé une compréhension approfondie des mécanismes d'interconnexion entre les services, de la gestion des dépendances, ainsi que des enjeux liés à la résilience, la redondance et la performance des systèmes virtualisés.

6.2 Compétences techniques et transversales acquises

Ce projet nous a permis de développer et renforcer plusieurs compétences :

- Techniques :
 - Déploiement automatisé d'infrastructures virtualisées
 - Configuration d'un système de répartition de charge avec Nginx
 - Mise en place d'un système de monitoring avec Prometheus et Grafana
 - Configuration d'une réplication MySQL maître-esclave
 - Résolution de problèmes liés à la connectivité, aux dépendances logicielles et à l'intégrité des services
- Transversales :
 - Travail collaboratif et gestion de version via Git
 - Capacité d'analyse et d'adaptation face aux imprévus techniques
 - Documentation technique rigoureuse et structuration claire du code
 - Gestion de projet en mode agile : planification, test, validation

6.3 Limites éventuelles et perspectives d'amélioration

Malgré les résultats satisfaisants, certaines limites peuvent être identifiées :

- L'infrastructure ne dispose pas encore d'un mécanisme de scalabilité automatique, ce qui pourrait être envisagé via un outil comme Ansible, Docker Swarm ou Kubernetes.
- Les tests de résilience ont été réalisés manuellement ; une intégration continue avec des scripts de test automatisés renforcerait la robustesse.
- Le système de supervision pourrait être enrichi avec des alertes proactives, une journalisation plus fine, ou l'ajout de Loki pour la gestion des logs.

À terme, le projet pourrait évoluer vers un déploiement en environnement cloud public (AWS, Azure, GCP), afin de tester sa portabilité et sa flexibilité à plus grande échelle.

6.4 Lien vers le dépôt Git

Le code source du projet, accompagné des scripts Bash, des fichiers de configuration Vagrant et des captures d'écran de validation, est disponible à l'adresse suivante :

Lien GitHub du projet :

“https://github.com/Mensan2024/V_projet”

7. Bibliographie

1. Rapport de projet de Virtualisation des étudiants de Master 1 ISIE 2025
2. Le site officiel de Vagrant : documentation technique et tutoriels de référence.
3. Le site officiel de [MySQL](https://dev.mysql.com/doc/) : documentation sur la réplication, la gestion des utilisateurs et la sécurité.
4. Le site officiel de Grafana : déploiement, dashboards, intégration avec Prometheus.
5. Le site officiel de Prometheus : collecte de métriques, alerte et configuration.
6. [LinuxCommand.org](https://linuxcommand.org/) : Documentation sur la syntaxe Bash et les scripts shell.
7. WILLIAM, T. (2020). Scripts Bash pour administrateurs Linux : du débutant à l'expert. Libre Linux Press.
8. Formation OpenClassrooms (2023). Déployer son infrastructure avec Vagrant. OpenClassrooms.com
9. Réseau des grandes écoles d'ingénieurs (2023). La virtualisation au service de l'efficacité énergétique des data centers. Étude de cas interne.
10. CAIRE, C. (2022). Surveillance et performances des infrastructures IT. Guide technique chez Eyrolles.
11. Documentation Apache HTTP Server : <https://httpd.apache.org/docs/>
12. Documentation Nginx officielle : <https://nginx.org/en/docs/>
13. TUTOSI, L. (2022). L'art de l'automatisation système sous Linux. En ligne, revue LinuxTech.
14. Cours de Cloud Computing et Virtualisation (2024). Département Informatique, Université Virtuelle de Côte d'Ivoire.

