Virtualisation et Cloud Computing

RAPPORT DE PROJET - GROUPE 7

Master1 [Sécurité Informatique]

Membre du groupe N°7 :

Nom	**Prénom**
OUIYA	Jean Paul
OUEDRAOGO	Teegwende Gyslain
SAWADOGO	Eulodie Carelle W.
TEONSA	Mohamed
ZOMA	W. Rodrigue

Enseignant: Dr SOMDA

Année Académique : 2024 - 2025

PLAN

INTRODUCTION

Présentation des Outils et Technologies

Conception de l'infrastructure

Mise en oeuvre Technique

Tests et Validation

CONCLUSION

INTRODUCTION

Dans le cadre du cours de Virtualisation et Cloud Computing, nous avons travaillé en groupe sur un projet visant à déployer une infrastructure virtuelle à l'aide de Vagrant et de scripts Bash. Ce projet nous a permis de simuler un environnement cloud local et de mieux comprendre les notions de déploiement automatisé, de gestion des machines virtuelles et de supervision d'une architecture distribuée.

I. Présentation des Outils et Technologies

1) Vagrant et VirtualBox

Vagrant est un outil open-source qui permet de créer, configurer et gérer des environnements de machines virtuelles de manière simple, reproductible et portable. Il utilise un fichier de configuration (Vagrantfile) pour décrire l'infrastructure désirée (nombre de machines, IPs, ressources, scripts de provisioning, etc.). Il permet de lancer plusieurs machines virtuelles avec des configurations précises en une seule commande.
VirtualBox est également un logiciel de virtualisation open-source développé par Oracle. Il permet de créer et d'exécuter plusieurs machines virtuelles (VM) sur un même hôte physique, chacune avec son propre système d'exploitation, ses ressources matérielles simulées (processeur, RAM, disque, carte réseau, etc.).

2) Bash et le Provisioning

Dans le compte ce projet, Bash est utilisé pour automatiser le **provisioning** des machines virtuelles, c'est-à-dire leur configuration automatique après création. Chaque rôle (serveur web, base de données, etc.) a son script dédié :

- lb.sh pour Nginx
- web1.sh et web2.sh pour Apache
- db-master.sh et db-slave.sh pour MySQL
- monitoring.sh pour le monitoring
- client.sh pour la machine cliente

3) nginx (équilibreur de charge)

Nginx est un serveur web léger et performant qui peut aussi agir comme **un équilibreur de charge** (*load balancer*). Il est largement utilisé pour répartir efficacement le trafic réseau entre plusieurs serveurs backend afin d'améliorer la disponibilité et les performances des applications web.
Dans ce projet, la machine virtuelle lb joue le rôle de point d'entrée unique de l'infrastructure.

Elle a les responsabilités suivantes :

- Recevoir les requêtes du client via navigateur.
- Répartir dynamiquement le trafic vers les serveurs web1 et web2 pour équilibrer la charge.
- Améliorer la tolérance aux pannes : si un serveur web tombe en panne, l'autre peut continuer à servir les pages.

4) Apache (Serveurs Web)

Apache HTTP Server, couramment appelé Apache, est un des serveurs web.
Il permet de diffuser du contenu web tel que des pages HTML, des images ou des scripts aux clients via le protocole HTTP. Ce serveur, libre, bénéficie d'un large support au sein des distributions Linux, ce qui en fait un choix privilégié pour de nombreux déploiements web.
Dans le contexte de ce projet, les machines web1 et web2 sont déployées avec Apache afin de :

- Servir une page d'accueil statique permettant d'identifier le serveur en cours de consultation.
- Répondre aux requêtes HTTP acheminées par le load balancer Nginx, installé sur la machine lb.

5) MySQL – Réplication Maître-Esclave

La réplication dans un système de gestion de base de données consiste à copier automatiquement les données d'un serveur principal vers un ou plusieurs serveurs secondaires. Ce mécanisme est particulièrement utile pour garantir la disponibilité, la tolérance aux pannes**.**
Dans le modèle maître-esclave de MySQL :

- Le serveur maître (master) est le point central où les opérations d'écriture (INSERT, UPDATE, DELETE) sont effectuées.
- Le serveur esclave (slave) reçoit une copie en temps quasi réel des modifications du maître et ne sert qu'aux opérations de lecture.

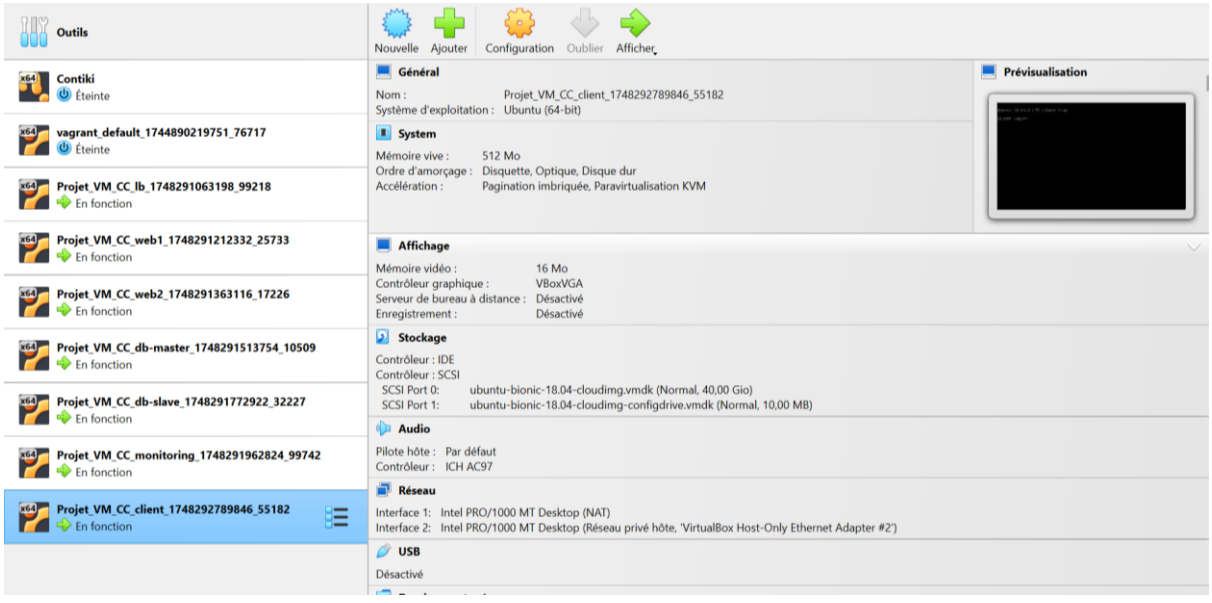
6) Prometheus et Grafana

Prometheus est une solution de monitoring open-source conçue pour collecter en temps réel des métriques provenant d'applications et de serveurs. Son fonctionnement repose sur un modèle de collecte de données en mode *pull*, où il interroge régulièrement des agents appelés *exporters* qui fournissent des informations détaillées sur les ressources système (CPU, mémoire, disque) ainsi que sur des services spécifiques, tels que MySQL.
Grafana est une plateforme de visualisation de données qui permet de concevoir des tableaux de bord interactifs et personnalisés à partir des métriques recueillies par Prometheus. Elle simplifie l'analyse visuelle de la performance et de l'état des systèmes grâce à des graphiques, des jauges et des alertes configurables selon les besoins.

II. Conception de l'Infrastructure

❖ Architecture Cible et plan d'adressage IP

Ceci est un aprecu de toutes les 7 machines virtuelles (VM) demarrées apres le lancement de la commande vagrant up sous l'invide de commande sur le repertoire ou sont stockés nos fichiers vagrantfile et les differents fichiers de provisionning de ces VM.



Adresse Réseau étant 192.68.56.0/24, nous avons procédé à une allocation statique des VM comme suit :

- **Client (192.168.56.16)**

Sert à **tester la connectivité** et l'accès à l'application via le Load Balancer.

- **Équilibreur de charge - lb (192.168.56.10)**

Utilise **Nginx** pour distribuer les requêtes HTTP entrantes vers les serveurs web web1 et web2 en fonction de la charge, assurant ainsi la répartition et la haute disponibilité.

- **Web1 (192.168.56.11) et Web2 (192.168.56.12)**

Hébergent les serveurs web Apache qui servent des pages HTML statiques. Ils reçoivent le trafic distribué par le Load Balancer.

- db-m**aster (192.168.56.13)**

Base de données MySQL principale qui gère les opérations d'écriture. Elle enregistre les modifications dans un journal binaire pour les répliquer vers le serveur esclave.

- db-s**lave (192.168.56.14)**

Base de données MySQL répliquée en lecture seule, synchronisée automatiquement avec la base maître pour assurer la redondance et la tolérance aux pannes.

-**Monitoring (192.168.56.15)**

Installe et exécute **Prometheus** pour la collecte des métriques et **Grafana** pour la visualisation. Permet de superviser l'ensemble du cluster en temps réel.

III. Mise en œuvre technique

La mise en œuvre technique du projet repose sur l'automatisation complète du déploiement d'une infrastructure multi-nœuds à l'aide de Vagrant et de scripts Bash pour le provisioning. Cette automatisation garantit la cohérence, la reproductibilité et la rapidité du déploiement.

1) Le Vagrantfile : Cœur de l'orchestration

Le fichier Vagrantfile permet de décrire de manière déclarative les 7 machines virtuelles nécessaires à notre infrastructure :

- Attribution des noms d'hôtes,
- Affectation des adresses IP statiques dans le sous-réseau 192.168.56.0/24,
- Définition des scripts de provisioning spécifiques à chaque machine,
- Spécification des ressources systèmes (RAM, CPU).

Cette structure modulaire permet de gérer indépendamment chaque VM tout en maintenant leur interdépendance logique.

2) Équilibreur de charge – lb (192.168.56.10)

Le Load Balancer (lb) joue un rôle d'optimisation de la charge du trafic entrant. Il agit comme un répartiteur, acheminant les requêtes HTTP reçues vers l'un des deux serveurs web (**web1** ou **web2**)

3) Serveurs Web – web1 (192.168.56.11) et web2 (192.168.56.12)

Les serveurs web1 et web2 constituent la couche applicative de l'infrastructure. Ils sont chargés de servir les pages web statiques aux utilisateurs finaux via le protocole HTTP, à travers le Load Balancer (lb). Leur duplication garantit la tolérance aux pannes

4) Bases de données db-master (192.168.56.13) et db-slave (192.168.56.14)

Les machines db-master et db-slave constituent le système de gestion de base de données MySQL de l'infrastructure. Leur objectif est de démontrer la mise en œuvre d'un mécanisme de réplication maître-esclave, qui permet :

- La redondance des données,
- La tolérance aux pannes,
- La séparation des charges de lecture/écriture dans un environnement complexe.

Dans cette architecture :

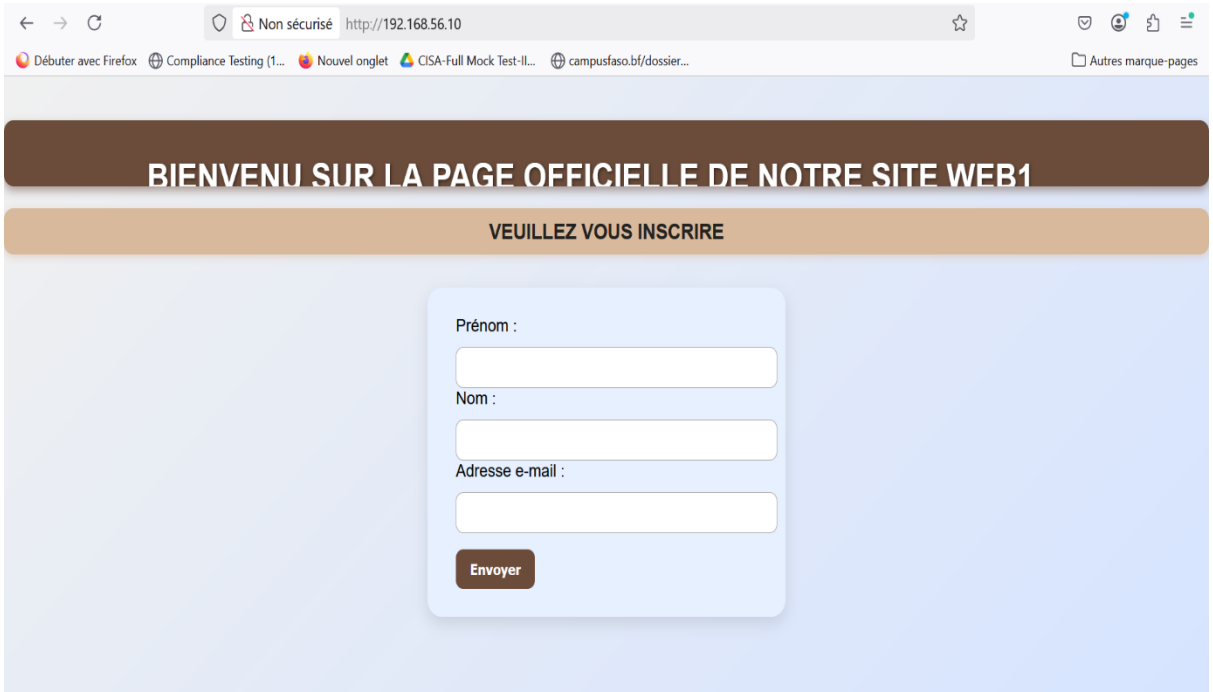
- **db-master** est responsable des écritures et mises à jour de la base de données.
- **db-slave** réplique automatiquement ces modifications pour garantir une copie fidèle et en temps réel.

IV. Tests et Validation

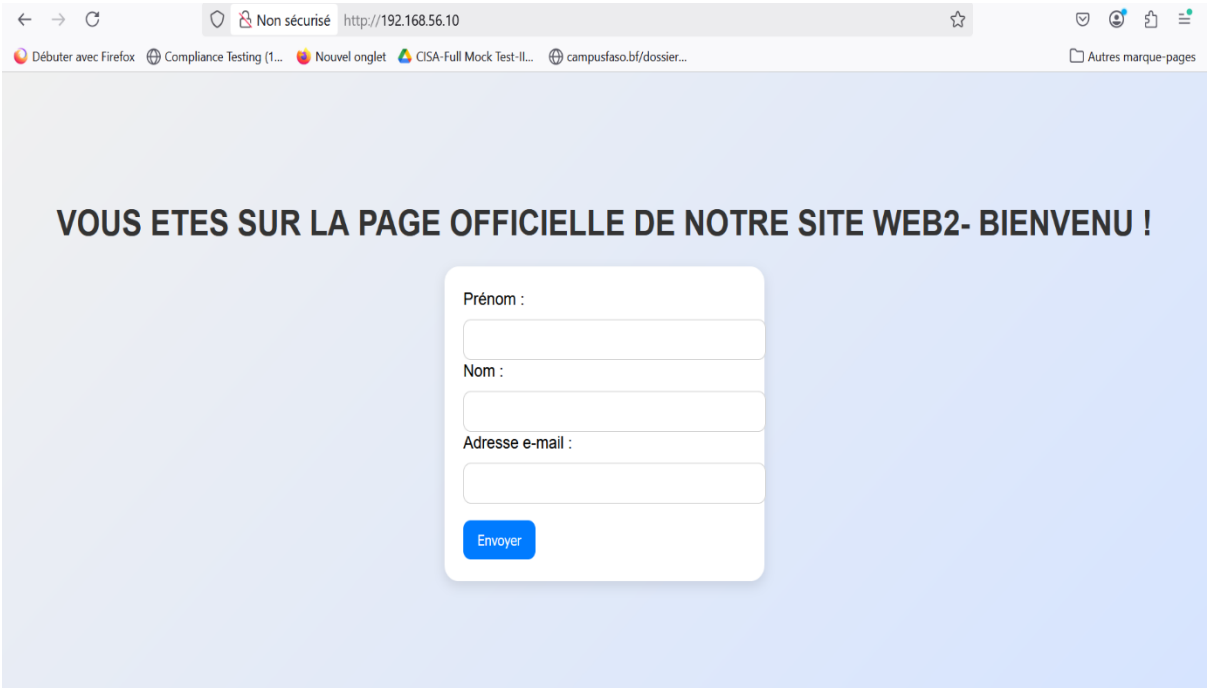
1) Accès web via le Load Balancer

Via un navigateur de la machine hôte : Nous tapons l'adresse IP de la VM lb : 192.68.56.10
Nous obtenons la page d'accueil du Web1. En actualisant nous basculons sur le Web2.

Le Web1

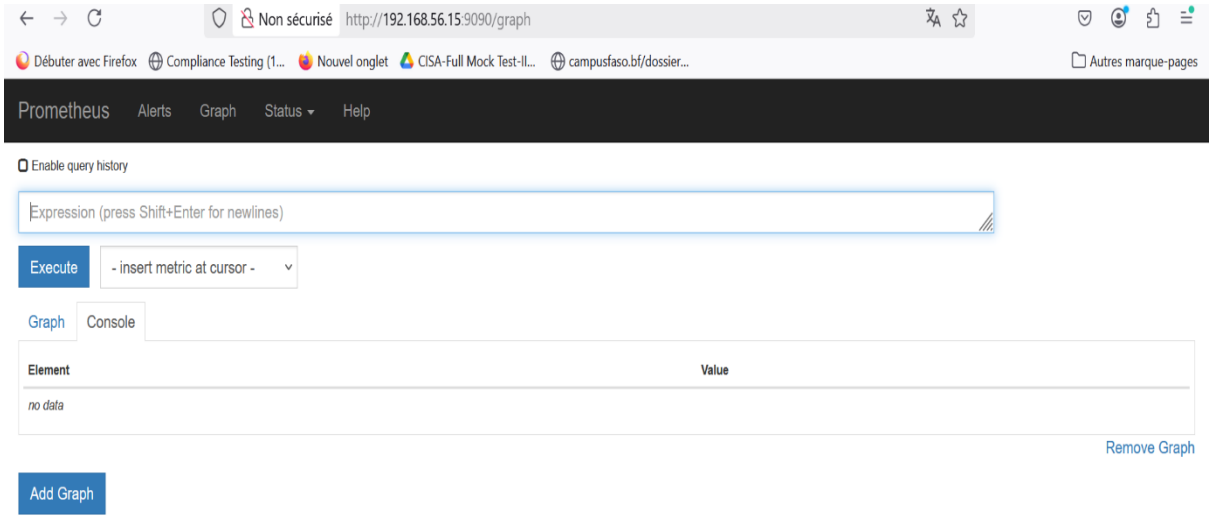


le web2

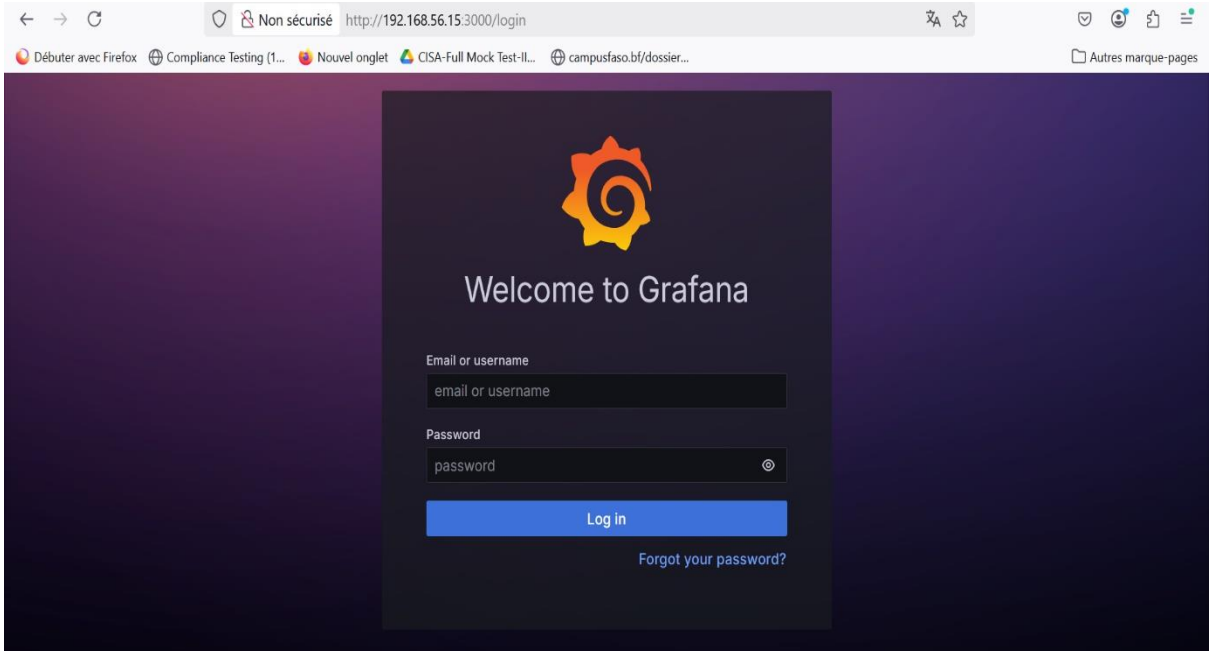


2) Supervision avec Prometheus et Grafana

Rappelons que l'adresse IP de la VM monitoring est : **192.168.56.15**
Pour accéder à Prometheus : Rendons-nous à un navigateur web et saisons le lien **http://192.168.56.15:9090**



Le lien d'accès à la plateforme Grafana : http://192.168.56.15:3000
Via un navigateur web, nous obtenons :



3) Tests sur les bases de données

Tester les bases de données db-master et db-slave est crucial pour assurer la cohérence des données répliquées entre les deux serveurs. Ces tests garantissent que toutes les modifications effectuées sur le master sont correctement

répercutées sur le slave, sans perte ni erreur. Ils permettent également de vérifier la capacité du système à gérer un basculement en cas de panne, assurant ainsi une continuité de service. De plus, ils évaluent les performances du slave, notamment pour la répartition des requêtes en lecture et la rapidité de réplication. Par ailleurs, ces tests renforcent la sécurité des échanges entre les bases, protégeant les informations sensibles. Sans eux, la fiabilité et la disponibilité de l'architecture seraient mises en danger.

a. Ecriture des données sur db-master ;

Nous allons commencer par créer une base de données nommée **Test_BD** pour disposer d'un environnement dédié. Avant cela, nous vérifierons si cette base existe déjà et, si c'est le cas, nous la sélectionnerons afin d'éviter les erreurs liées à une création redondante. Ensuite, nous procéderons à la création des tables nécessaires pour structurer les données.
Une fois les tables en place, nous insérerons les données initiales pour alimenter la base. Les requêtes sont organisées de manière claire et séquentielle :

- **Création de la base de données Test_BD**
  DROP DATABASE IF EXISTS Test_BD; # Suppression de la base de données **Test_BD** si elle existe
  CREATE DATABASE Test_BD ; # Création d'une nouvelle base de donnés **Test_BD**

- **Sélection de la base de données Test_BD**
  USE Test_BD ;

- **Création d'une table simple(Etudiant)**

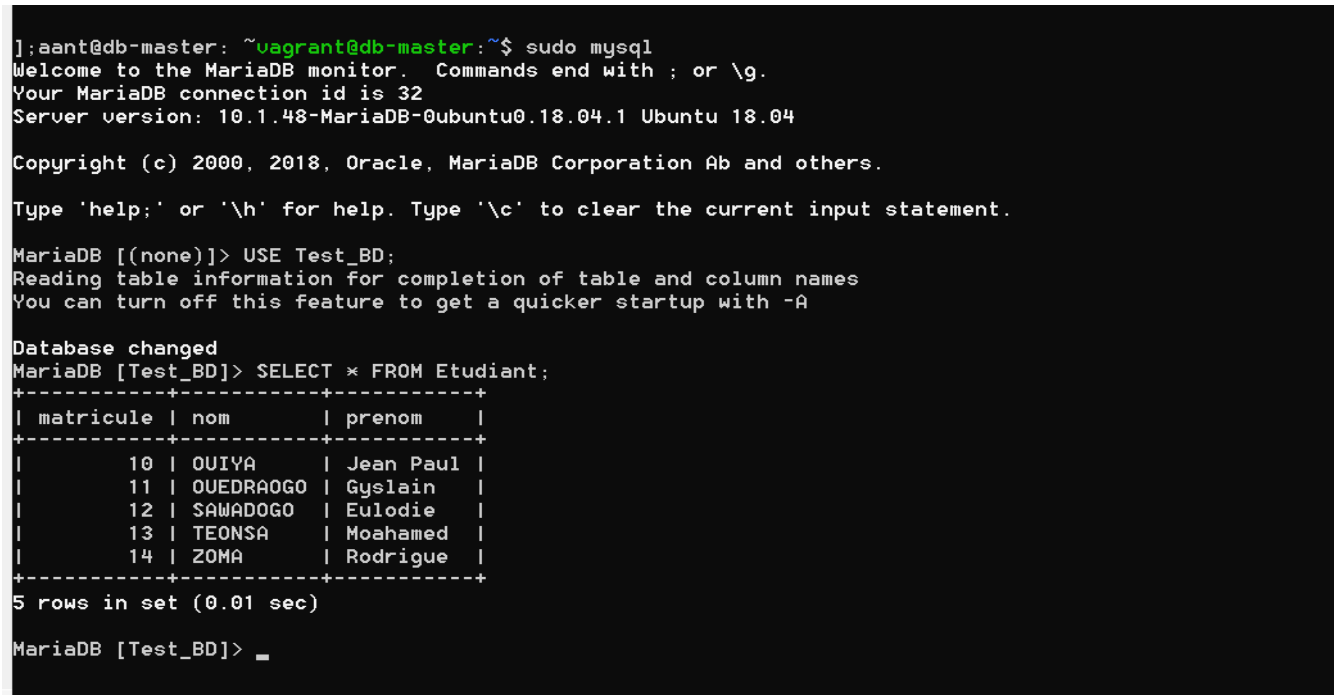
```
CREATE  TABLE Etudiant (
   matricule VARCHAR(20) CLÉ PRIMAIRE,
   nom VARCHAR(50) NOT NULL,
   prenom VARCHAR(50) NOT NULL,
);
```

- **Insertion des données dans la table Etudiant**

```
INSERT INTO Etudiant (matricule, nom, prenom) VALUES
('10', 'OUIYA', 'Jean Paul'),
('11', 'OUEDRAOGO', 'Gyslain'),
```

('12', 'SAWADOGO', 'Eulodie'),
('13', 'TEONSA', 'Moahamed'),
('14', 'ZOMA', 'Rodrigue');

Pour afficher les données sur la db-master :
SELECT * FROM Etudiant ;
Le resultat sur la db-master est présenté comme suit :



b. Vérification des données répliquées sur db-slave

Pour vérifier que les données enregistrées sur la base db-master sont correctement répliquées sur la db-slave, nous allons nous connecter à cette dernière, puis exécuter une requête d'affichage du contenu de la table *Etudiant*, à l'aide des commandes suivantes :

- Connexion à la machine virtuelle db-slave

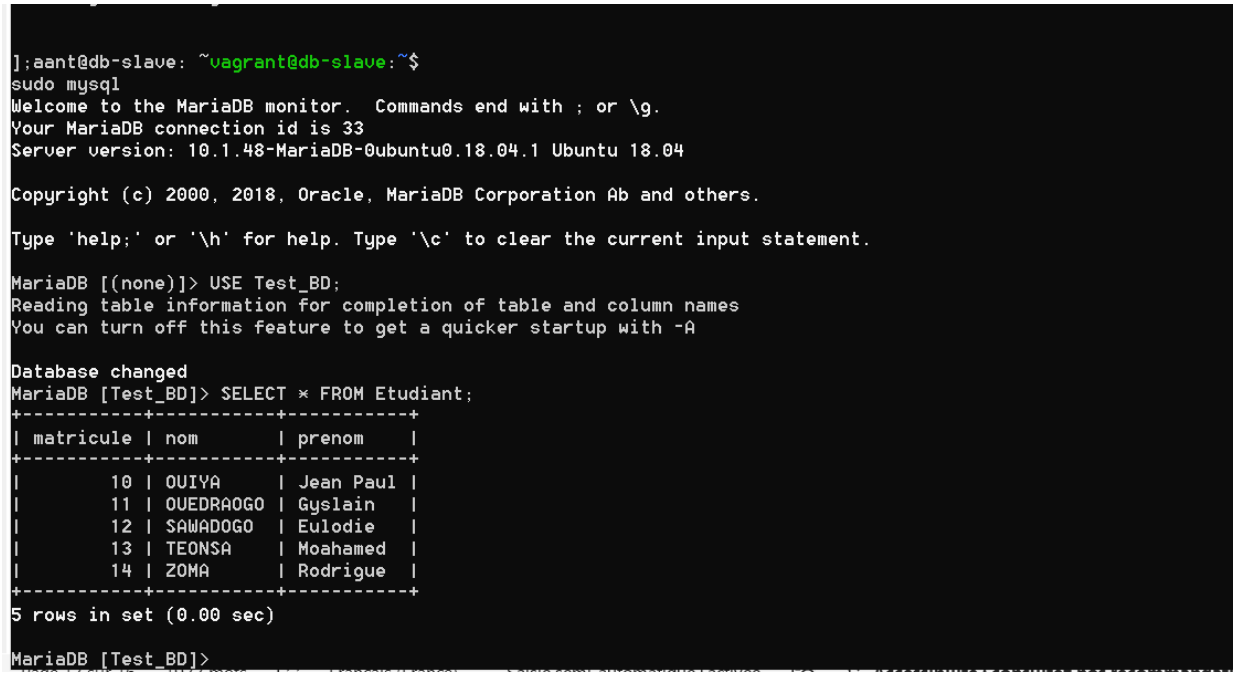
Vagrant ssh db-slave

- Accès à MySQL en mode super utilisateur
  sudo mysql

- Selection de la base de données Test_BD
  USE Test_BD

- Affichage du contenu de la table Etudiant

SELECT * FROM Etudiant



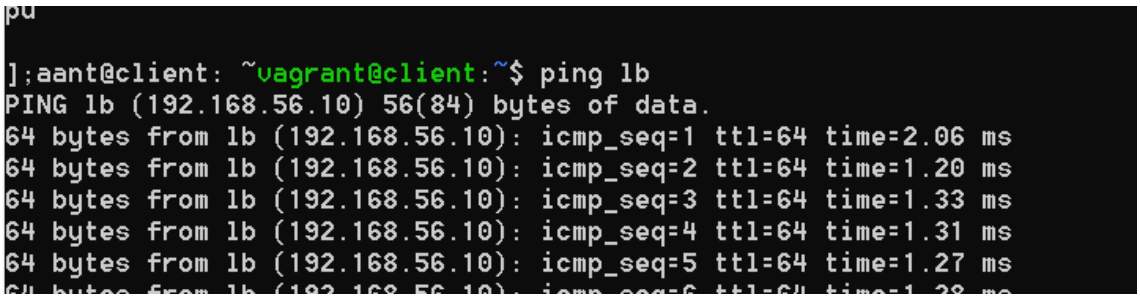
4) Tests depuis la machine client

Tester la connectivité depuis la machine virtuelle client permet de vérifier l'accessibilité réseau et fonctionnelle des services déployés sur l'ensemble des autres machines de de notre architecture. Cette démarche garantit que chaque composant est correctement configuré et que les communications inter-machines s'effectuent conformément selon les exigences du système.

Pour ce faire nous allons nous connecter à la machine cliente via ssh : **vagrant ssh client**

a. Connectivité sur la VM lb (Load balancer)

:~ping lb ping 192.168.56.10



b. Connectivité sur la VM web1

:~ping web1 ping 192.168.56.11

```
];aant@client: ~vagrant@client:~$ ping web1
PING web1 (192.168.56.11) 56(84) bytes of data.
64 bytes from web1 (192.168.56.11): icmp_seq=1 ttl=64 time=1.22 ms
64 bytes from web1 (192.168.56.11): icmp_seq=2 ttl=64 time=1.39 ms
64 bytes from web1 (192.168.56.11): icmp_seq=3 ttl=64 time=1.42 ms
64 bytes from web1 (192.168.56.11): icmp_seq=4 ttl=64 time=0.978 ms
64 bytes from web1 (192.168.56.11): icmp_seq=5 ttl=64 time=1.08 ms
64 bytes from web1 (192.168.56.11): icmp_seq=6 ttl=64 time=1.23 ms
64 bytes from web1 (192.168.56.11): icmp_seq=7 ttl=64 time=1.40 ms
64 bytes from web1 (192.168.56.11): icmp_seq=8 ttl=64 time=1.06 ms
64 bytes from web1 (192.168.56.11): icmp_seq=9 ttl=64 time=1.24 ms
64 bytes from web1 (192.168.56.11): icmp_seq=10 ttl=64 time=1.14 ms
```

c. Connectivité sur la VM web2

:~ping web2 ping 192.168.56.12

```
];aant@client: ~vagrant@client:~$ ping web2
PING web2 (192.168.56.12) 56(84) bytes of data.
64 bytes from web2 (192.168.56.12): icmp_seq=1 ttl=64 time=1.26 ms
64 bytes from web2 (192.168.56.12): icmp_seq=2 ttl=64 time=1.30 ms
64 bytes from web2 (192.168.56.12): icmp_seq=3 ttl=64 time=1.19 ms
64 bytes from web2 (192.168.56.12): icmp_seq=4 ttl=64 time=1.10 ms
64 bytes from web2 (192.168.56.12): icmp_seq=5 ttl=64 time=1.34 ms
64 bytes from web2 (192.168.56.12): icmp_seq=6 ttl=64 time=1.41 ms
64 bytes from web2 (192.168.56.12): icmp_seq=7 ttl=64 time=1.07 ms
64 bytes from web2 (192.168.56.12): icmp_seq=8 ttl=64 time=1.04 ms
```

d. Connectivité sur la VM db-master

:~ping db-master ping 192.168.56.13

```
];aant@client: ~vagrant@client:~$ ping db-master
PING db-master (192.168.56.13) 56(84) bytes of data.
64 bytes from db-master (192.168.56.13): icmp_seq=1 ttl=64 time=1.00 ms
64 bytes from db-master (192.168.56.13): icmp_seq=2 ttl=64 time=1.14 ms
64 bytes from db-master (192.168.56.13): icmp_seq=3 ttl=64 time=1.03 ms
64 bytes from db-master (192.168.56.13): icmp_seq=4 ttl=64 time=1.04 ms
64 bytes from db-master (192.168.56.13): icmp_seq=5 ttl=64 time=1.28 ms
64 bytes from db-master (192.168.56.13): icmp_seq=6 ttl=64 time=1.13 ms
64 bytes from db-master (192.168.56.13): icmp_seq=7 ttl=64 time=1.13 ms
64 bytes from db-master (192.168.56.13): icmp_seq=8 ttl=64 time=0.926 ms
64 bytes from db-master (192.168.56.13): icmp_seq=9 ttl=64 time=1.22 ms
```

e. Connectivité sur la VM db-slave

:~ping db-slave ou ping 192.168.56.14

```
];aant@client: ~vagrant@client:~$ ping db-slave
PING db-slave (192.168.56.14) 56(84) bytes of data.
64 bytes from db-slave (192.168.56.14): icmp_seq=1 ttl=64 time=0.577 ms
64 bytes from db-slave (192.168.56.14): icmp_seq=2 ttl=64 time=1.10 ms
64 bytes from db-slave (192.168.56.14): icmp_seq=3 ttl=64 time=1.25 ms
64 bytes from db-slave (192.168.56.14): icmp_seq=4 ttl=64 time=1.26 ms
64 bytes from db-slave (192.168.56.14): icmp_seq=5 ttl=64 time=1.05 ms
64 bytes from db-slave (192.168.56.14): icmp_seq=6 ttl=64 time=1.30 ms
```

f. Connectivité sur la VM monitoring

:~ ping monitoring ou ping 192.168.56.15

```
];aant@client: ~vagrant@client:~$ ping monitoring
PING monitoring (192.168.56.15) 56(84) bytes of data.
64 bytes from monitoring (192.168.56.15): icmp_seq=1 ttl=64 time=1.09 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=2 ttl=64 time=1.28 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=3 ttl=64 time=1.25 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=4 ttl=64 time=1.35 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=5 ttl=64 time=1.05 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=6 ttl=64 time=1.32 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=7 ttl=64 time=1.15 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=8 ttl=64 time=1.10 ms
64 bytes from monitoring (192.168.56.15): icmp_seq=9 ttl=64 time=1.19 ms
```

CONCLUSION

Ce projet nous a permis d'appliquer de manière concrète les notions vues en cours, notamment la gestion d'un cluster de machines virtuelles avec Vagrant et Bash. En construisant cette infrastructure virtuelle, nous avons pu mieux comprendre les enjeux liés à l'automatisation, à la configuration système, à la supervision et à l'interconnexion des services. Ce travail en groupe a également renforcé notre esprit de collaboration et notre capacité à résoudre des problèmes techniques en équipe.