easy Former
La formation à 360°

Ansible

Automatisation MySQL

Référence : EF-TEST-TEST

Auteur(s) :
Yann BENHAMRON

Destinataire(s) :
Easyformer

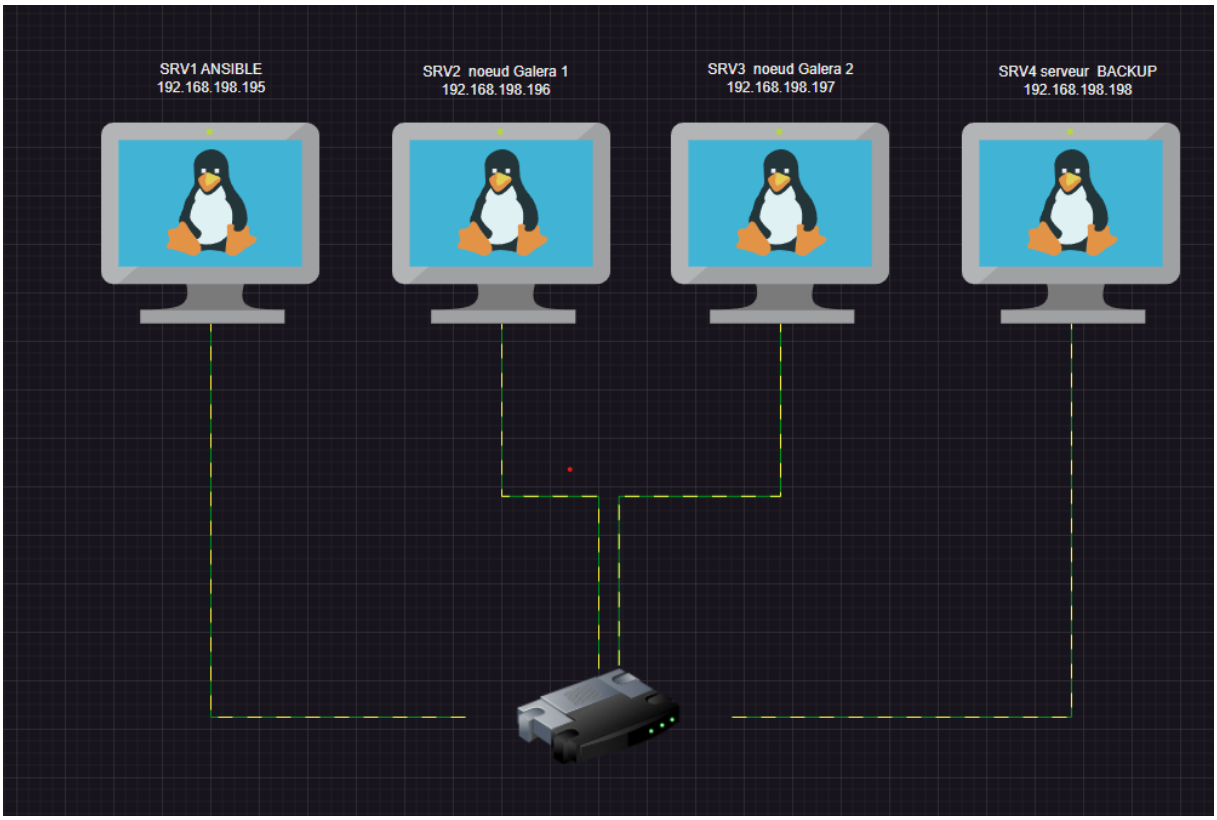
Date de modification : 13/02/24

Version : 1

Sommaire

page

 Date : 13/02/24



1 Enoncé

Pour cet exercice, nous allons élaborer un cas pratique qui englobe :

- La configuration d'un cluster Galera pour MariaDB sur deux nœuds à l'aide d'Ansible
- La création automatisée de bases de données et de tables
- La mise en place d'une stratégie de sauvegarde automatisée avec mysqldump et cron.

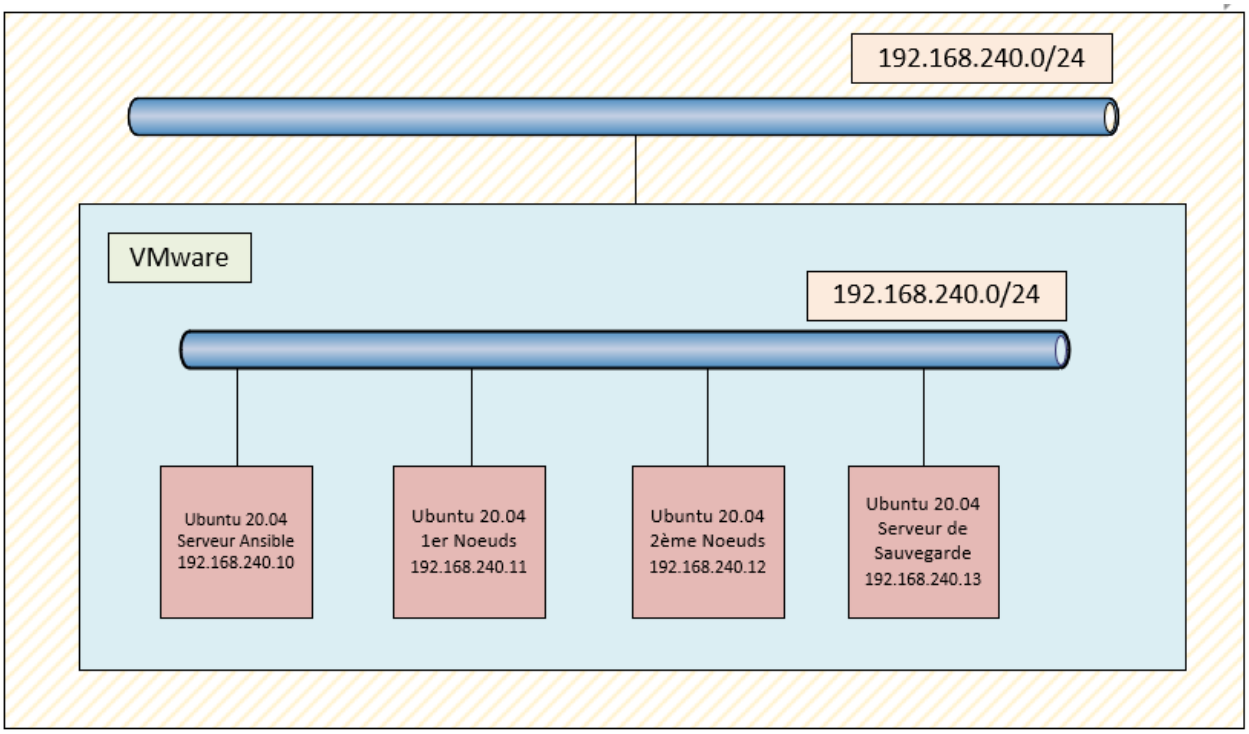
2 Objectifs

Les Objectifs sont les suivantes :

1. Configurer un cluster Galera pour MariaDB sur deux nœuds en utilisant Ansible pour l'installation et la configuration.
2. Créer des bases de données et des tables de manière automatisée sur les deux nœuds du cluster.
3. Automatiser les sauvegardes avec mysqldump et cron sur un serveur distant, en s'exécutant toutes les 5 minutes.

3 Configuration Requise

Serveur Ansible : 192.168.240.10
Premier nœud du cluster : 192.168.240.11
Deuxième nœud du cluster : 192.168.240.12
Serveur de sauvegarde : 192.168.240.13



Suivre les étapes suivantes :

1. Configuration du Cluster Galera avec Ansible
    a. Définir un playbook Ansible pour installer MariaDB et configurer le cluster Galera sur les deux nœuds (192.168.240.11 et 192.168.240.12).
2. Création Automatisée de Bases de Données et de Tables
    a. Définir un playbook Ansible qui exécute les commandes SQL pour créer les éléments suivants sur chaque nœud du cluster
    b. Etablir une clé de relation entre les tables Utilisateurs et Connexions dans votre base de données galera_cluster_db

Base de données : galera_cluster_db						
Table 1 : Utilisateurs				**Table 2 : Connexions**		
Id (INT, clé primaire, auto-incrémente)	Nom (VARCHAR)	Email (VARCHAR)	Date inscription (DATE)	Connexion_id (INT, clé primaire, auto-incrément)	Utilisateur_id (INT, clé étrangère vers Utilisateurs.id)	Timestamp (DATETIME).
1010	Yann	yann@hotmail.com	05/02/24	1	1030	2024-02-07 10:00:00
1020	David	david@hotmail.com	06/02/24	2	1010	2024-02-07 11:00:00
1030	Dorian	dorian@hotmail.com	07/02/24	3	1020	2024-02-07 12:00:00

Pour établir une clé de relation entre les tables **Utilisateurs** et **Connexions** dans votre base de données **galera_cluster_db**, vous utilisez la colonne **Utilisateur_id** dans la table **Connexions** comme **clé étrangère** qui pointe vers la colonne **Id** de la table **Utilisateurs**. Cette relation permet de lier chaque enregistrement de connexion à un utilisateur spécifique dans la base de données.
Une clé étrangère est un concept clé dans le domaine des bases de données relationnelles, utilisé pour maintenir l'intégrité des données et établir une relation logique entre deux tables.

3. Automatisation des Sauvegardes avec MysqlDumb et Cron
    a. Définir un playbook Ansible pour concevoir un script shell. Ce script emploiera mysqldump pour effectuer des sauvegardes de galera_cluster_db sur chaque nœud, en stockant ces sauvegardes dans des fichiers distincts au sein d'un dossier /etc/SAVE situé sur le serveur de sauvegarde à l'adresse 192.168.240.13.
    b. Configurer une tâche cron sur le serveur de sauvegarde pour exécuter ce script toutes les 5 minutes.

4 Consignes Additionnelles

Pour garantir l'efficacité de la stratégie de sauvegarde, il est recommandé de tester la restauration d'une base de données à partir des fichiers de sauvegarde et de documenter chaque étape du TP, permettant ainsi une reproduction ou une vérification aisée par d'autres.

5 Livrables

À la fin du TP, vous devez soumettre les éléments suivants :

- Playbook Ansible pour l'installation et la configuration de MariaDB et du cluster Galera.
- Playbook Ansible pour la création de bases de données et de tables.
- Playbook Ansible qui exécute les commandes SQL pour créer la base de données, les tables, le contenue des tables et la clé étrangère qui permet de faire la relation des deux tables
- Script de sauvegarde et configuration cron pour les sauvegardes automatiques.
- Documentation détaillée des étapes, commandes SQL utilisées pour la création des tables, et instructions pour la vérification de l'exercice.

6 Procédure

6.1 Création des Vms avec vagrant (Provider Vmware)

6.1.1 Prérequis :

Vagrant installé sur votre machine Vagrant et Vagrant_VMware_Utility
Plugin Vagrant VMware Desktop installé : vagrant plugin install vagrant-vmware-desktop

Étapes :
Configuration du Vagrantfile :

Assurez-vous que les ressources de chaque machine virtuelle (RAM, CPU) sont adaptées à vos besoins.

Ouvrez un terminal dans le répertoire où se trouve le Vagrantfile.
Exécutez la commande suivante pour créer le Vagrantfile

```
vagrant init waounde221/ubuntu
```

Exécutez la commande suivante pour créer les machines virtuelles :

```
vagrant up --provider vmware_desktop
```

Connexion aux VMs :

Une fois la création terminée, utilisez la commande suivante pour vous connecter à une VM spécifique (remplacez X par le numéro de la VM) :
Copy code

```
vagrant ssh srvX ou ssh vagrant@IPdela VM
```

Provisionnement automatique :

Les VMs seront automatiquement mises à jour, le port 22 sera ouvert, et la connexion SSH en tant que root sera autorisée.
Personnalisation (si nécessaire) :

Si vous souhaitez personnaliser davantage les machines virtuelles, modifiez le script de provisionnement dans le Vagrantfile en conséquence. Vous pouvez également utiliser un script sh personnalisé avec Vagrant pour automatiser et configurer des tâches spécifiques selon vos besoins.
Arrêt des VMs :
Pour arrêter les VMs, utilisez la commande :

```
vagrant halt
```

Suppression des VMs :
Si vous souhaitez supprimer complètement les VMs, utilisez la commande :

```
vagrant destroy -f
```

Vous pouvez personnaliser les paramètres du vagrantfile selon vos besoins spécifiques avant de lancer la création des VMs.

Ce Vagrantfile permet de créer 4 VM avec l'image box waounde221/ubuntu
L'user par default de vagrant est : user ➔ vagrant password ➔ vagrant

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Définition des ressources pour chaque machine virtuelle
RAM = 2048
CPU = 2
# Configuration générale de Vagrant
Vagrant.configure("2") do |config|
  # Boucle pour créer 4 machines virtuelles
  (1..4).each do |i|
    config.vm.define "srv#{i}" do |srv|
      # Configuration de la box Vagrant pour chaque machine virtuelle
      srv.vm.box = "waounde221/ubuntu"
      # Attribution d'un nom d'hôte unique à chaque machine virtuelle
      srv.vm.hostname = "srv#{i}"
      # Configuration du fournisseur de la machine virtuelle (VMware Desktop)
      srv.vm.provider "vmware_desktop" do |v|
        # Activation de l'interface graphique
        v.gui = true
        # Spécification de la mémoire pour chaque machine virtuelle
        v.memory = RAM
        # Spécification du nombre de CPU pour chaque machine virtuelle
        v.cpus = CPU
      end
# Provisionnement pour mettre à jour les 4 VMs, ouvrir le port 22 et autoriser la connexion SSH en tant que root
      srv.vm.provision "shell", inline: <<-SHELL
        # Mise à jour du système
        sudo apt update && sudo apt full-upgrade -y
        # Modification du fichier de configuration SSH pour autoriser la connexion en tant que root
        sudo sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
        # Redémarrage du service SSH
        sudo systemctl restart sshd
        # Création d'un mot de passe pour l'utilisateur root (remplacez 'vagrant' par votre mot de passe)
        sudo sh -c 'echo "root:vagrant" | chpasswd'
      SHELL
    end
  end
end
```

6.2 Configuration du server

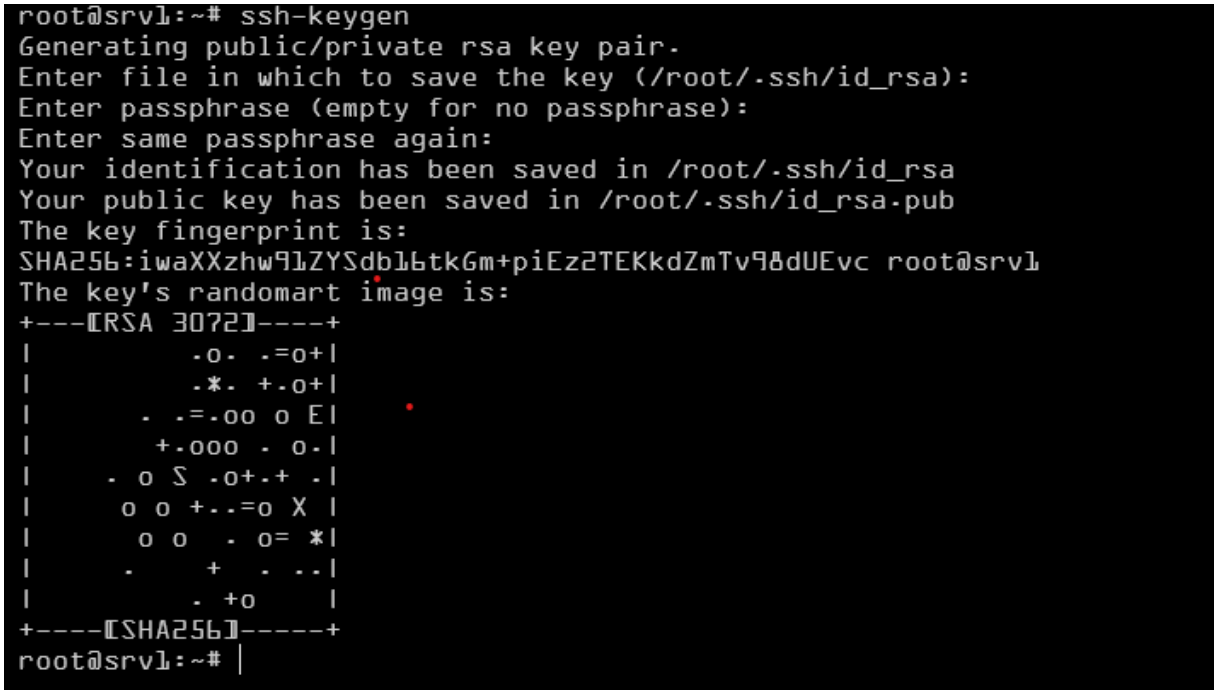
6.2.1 Installation d'ansible

```
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install python3 -y
sudo apt install ansible -y
sudo apt install ansible-core -y
```

6.2.2 Génération de clé SSH

La clé SSH est une paire de clés cryptographiques utilisées pour sécuriser les communications entre deux parties via le protocole SSH (Secure Shell). Cette paire de clés se compose de deux parties complémentaires : la clé privée et la clé publique.
La commande **SSH-KEYGEN -B 2048** est utilisée pour générer une paire de clés SSH (Secure Shell) sur un système Unix ou Linux.
Voici ce que font les différents composants de cette commande :
SSH-KEYGEN : C'est l'utilitaire en ligne de commande pour la génération, la gestion et la conversion de clés SSH.
-B 2048 : Cela spécifie la taille de la clé, en bits. Dans ce cas, la taille de la clé est définie à 2048 bits. La taille de la clé est liée à la sécurité : des clés plus longues offrent généralement une meilleure sécurité

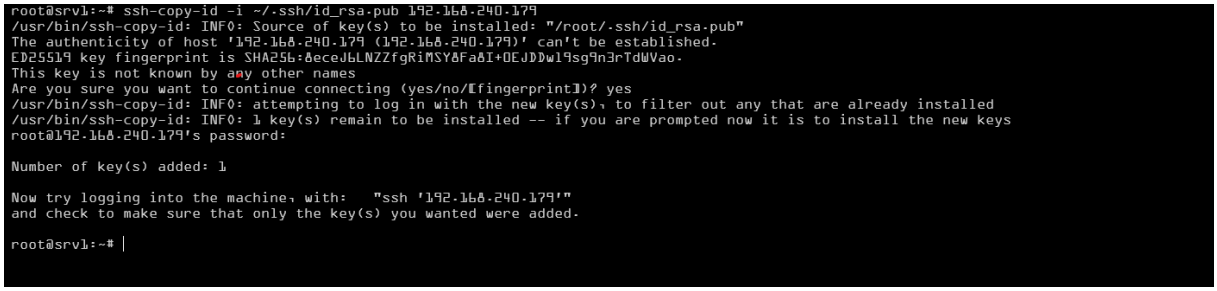
```
ssh-keygen -b 2048
```



6.2.3 Partage de clé ssh sur les deux serveurs Galera et sur le serveur Backup

Le partage de clés SSH fait référence à la distribution de la clé publique d'une paire de clés SSH à d'autres systèmes ou utilisateurs afin de permettre des connexions sécurisées
La commande ssh-copy-id est utilisée pour installer la clé publique d'un utilisateur sur une machine distante, permettant ainsi une connexion sans mot de passe à cette machine.
SSH-COPY-ID : C'est l'utilitaire qui copie la clé publique de l'utilisateur sur la machine distante.
-I ~/.SSH/ID_RSA.PUB : Cette option spécifie le chemin de la clé publique que vous souhaitez installer sur la machine distante.
192.168.240.180 : C'est l'adresse IP de la machine distante à laquelle vous souhaitez copier la clé publique.

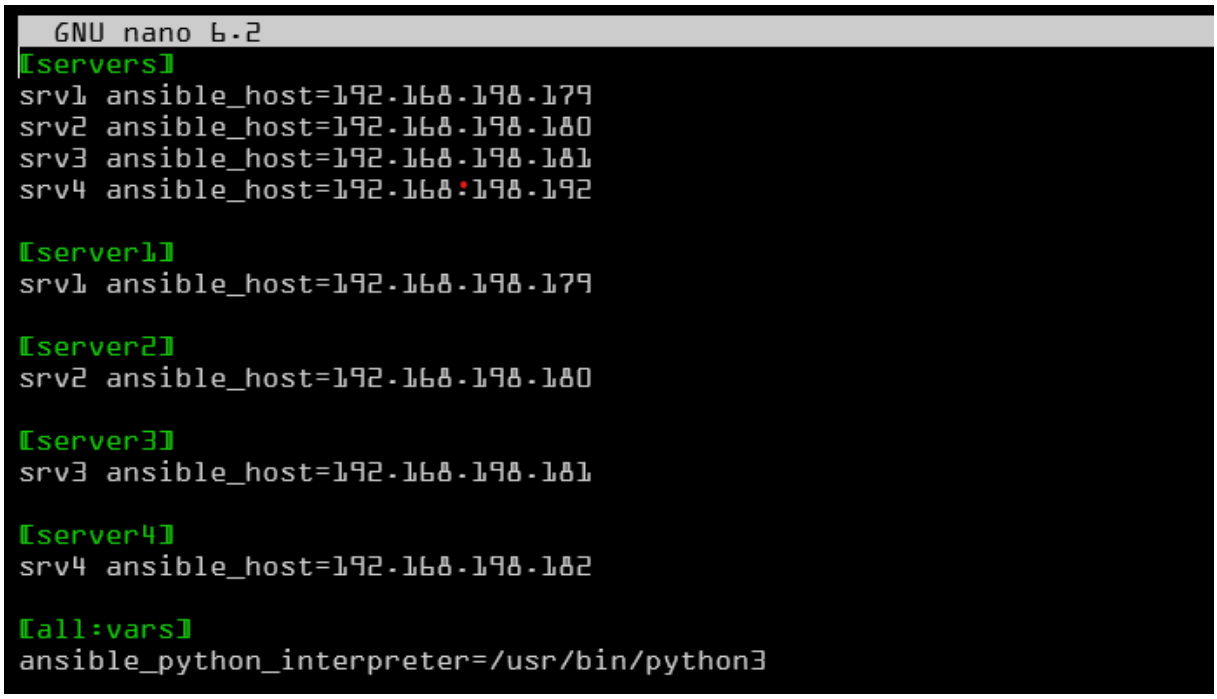
```
ssh-copy-id -i ~/.ssh/id_rsa.pub 192.168.240.180
```



6.2.4 Configuration du ficher hosts d'ansible

Le fichier d'inventaire qui spécifie les hôtes sur lesquels les tâches Ansible doivent être exécutées. C'est un fichier de configuration qui répertorie les machines cibles et peut également inclure des informations supplémentaires telles que les utilisateurs SSH, les ports, les groupes d'hôtes, et d'autres variables.

```
Nano /etc/ansible/hosts
```



6.3 Création du cluster avec Galera

6.3.1 Création du rôle galera

Nous allons créer notre rôle avec ansible-galaxy pour se faciliter le travail on va se déplacer dans le répertoire des rôles ansibles.

```
cd /etc/ansible/roles/
```

```
ansible-galaxy init galera
```

```
root@srv1:/etc/ansible/roles# ansible-galaxy init galera
- Role galera was created successfully
root@srv1:/etc/ansible/roles#
```

```
ansible all -m ping -u root
```

Pour tester la connectivité avec les hôtes et de vérifier que Ansible peut les atteindre et exécuter des commandes de base.

```
root@srv1:/etc/ansible/roles# ansible all -m ping -u root
srv4 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
srv3 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
srv1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
srv2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
root@srv1:/etc/ansible/roles#
```

6.3.2 Configuration des serveurs Galera

Configurez le fichier de configuration Galera (galera.cnf) pour spécifier les options de cluster, les détails sur le serveur actuel et les autres serveurs du cluster

BINLOG_FORMAT=ROW : Spécifie le format de journal binaire à utiliser. Ici, ROW indique que les modifications aux données seront enregistrées sous forme de lignes modifiées.
DEFAULT-STORAGE-ENGINE=INNODB : Définit le moteur de stockage par défaut à InnoDB, qui est recommandé pour être utilisé avec Galera Cluster.
INNODB_AUTOINC_LOCK_MODE=2 : Configure le mode de verrouillage pour les colonnes auto-incrémentées dans InnoDB. La valeur 2 signifie que le verrouillage est optimisé pour les performances dans un environnement multi-maître.
BIND-ADDRESS=0.0.0.0 : Spécifie que MySQL écoute sur toutes les interfaces réseau. Cela permet aux autres nœuds du cluster d'accéder à ce nœud par son adresse IP.
WSREP_ON=ON : Active la prise en charge de Galera Cluster pour ce serveur MySQL.
WSREP_PROVIDER=/USR/LIB/GALERA/LIBGALERA_SMM.SO : Indique le chemin vers la bibliothèque partagée de Galera Cluster.

WSREP_CLUSTER_NAME="TEST_CLUSTER" : Donne un nom au cluster Galera, permettant aux nœuds de se joindre au même cluster.
WSREP_CLUSTER_ADDRESS="GCOMM://192.168.240.180,192.168.240.181" : Indique les adresses IP des nœuds du cluster. Dans cet exemple, le cluster est composé de deux nœuds, avec les adresses IP 192.168.240.180 et 192.168.240.181.
WSREP_SST_METHOD=RSYNC : Spécifie la méthode de transfert de données lors de la synchronisation. Ici, rsync est utilisé.
WSREP_NODE_ADDRESS="192.168.240.180" : L'adresse IP de ce nœud du cluster.
WSREP_NODE_NAME="srv2" : Un nom unique pour identifier ce nœud au sein du cluster.

nano / galera/templates/galera2.cnf.j2

Configuration du fichier galera2.cnf.j2 qui sera copier avec le playbook dans le serveur 2 /etc/mysql/conf.d/galera.cnf
nous allons créez le fichier galera2.cnf.j2 pour notre premier nœud srv2 et on va faire de même pour le fichier galera3.cnf.j2 pour le srv3 .

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.240.180,192.168.240.181"
# Galera Synchronization Configuration
wsrep_sst_method=rsync
# Galera Node Configuration
wsrep_node_address="192.168.240.180"
wsrep_node_name="srv2"a
```

Configuration de galera3.cnf.j2

nano / galera/templates/galera3.cnf.j2

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.240.180,192.168.240.181"
# Galera Synchronization Configuration
wsrep_sst_method=rsync
# Galera Node Configuration
wsrep_node_address="192.168.240.181"
wsrep_node_name="srv3"
```

6.3.3 Création du playbook galera

Ce playbook Ansible déploie un cluster MariaDB avec Galera sur plusieurs nœuds. Je vais expliquer les différentes parties du playbook :
Déployer le cluster MariaDB

hosts : servers: Spécifie que les tâches de ce bloc seront exécutées sur tous les hôtes du groupe servers définis dans l'inventaire Ansible.

become: yes: Indique que les tâches doivent être exécutées en tant que superutilisateur (root).

gather_facts: false: Désactive la collecte automatique de faits par Ansible. Cela peut être utile pour les playbooks qui n'ont pas besoin d'informations détaillées sur les hôtes avant l'exécution des tâches.

Tâches :
Installer MariaDB : Utilise le module Ansible apt pour installer le paquet mariadb-server.
Installer le paquet python3-mysqldb : Installe le paquet python3-mysqldb nécessaire pour utiliser MySQL avec Python.
Installer Galera-4 : Utilise le module Ansible apt pour installer le paquet galera-4.

2. Configuration de Galera sur les nœuds 2 et 3
hosts: server2 et hosts: server3: Ces blocs spécifient que les tâches suivantes doivent être exécutées respectivement sur server2 et server3.

nano /etc/ansible/roles/galera/tasks/galera.yml

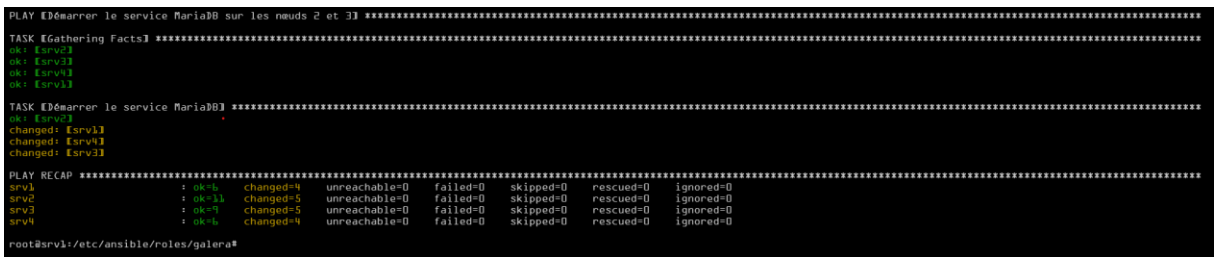
```
# Déployer le cluster MariaDB
- name: Déployer le cluster MariaDB
  hosts: servers
  become: yes
  gather_facts: false
  tasks:
    - name: Installer MariaDB
      apt:
        name: mariadb-server
        state: present

    - name: Arrêter le service MariaDB
      systemd:
        name: mysql
        state: stopped

    - name: Installer le paquet python3-mysqldb
      apt:
        name: python3-mysqldb
        state: present

    - name: Installer Galera-4
      apt:
        name: galera-4
        state: present

# Configuration de Galera sur le nœud 2
- name: Configurer MariaDB pour Galera sur le nœud 2
  hosts: server2
  tasks:
    - name: Créer le répertoire /etc/mysql/conf.d s'il n'existe pas
      file:
        path: "/etc/mysql/conf.d"
        state: directory
        mode: 0755  # Les permissions peuvent être ajustées selon vos besoins

    - name: Copier Galera sur le nœud 2
      template:
        src: "/etc/ansible/roles/galera/templates/galera2.cnf.j2"
        dest: "/etc/mysql/conf.d/galera.cnf"
```

```
# Configuration de Galera sur le nœud 3
- name: Configurer MariaDB pour Galera sur le nœud 3
  hosts: server3
  tasks:
    - name: Créer le répertoire /etc/mysql/conf.d s'il n'existe pas
      file:
        path: "/etc/mysql/conf.d"
        state: directory
        mode: 0755  # Les permissions peuvent être ajustées selon vos besoins

    - name: Copier Galera sur le nœud 3
      template:
        src: "/etc/ansible/roles/galera/templates/galera3.cnf.j2"
        dest: "/etc/mysql/conf.d/galera.cnf"

# Exécuter la commande sur le nœud 2
- name: Exécuter la commande sur le nœud 2
  hosts: server2
  become: yes
  tasks:
    - name: Lancer la commande shell
      command: |
        galera_new_cluster

# Démarrer le service MariaDB sur les nœuds 2 et 3
- name: Démarrer le service MariaDB sur les nœuds 2 et 3
  hosts: servers
  become: yes
  tasks:
    - name: Démarrer le service MariaDB
      systemd:
        name: mysql
        state: started
```

ansible-playbook /etc/ansible/roles/galera/tasks/galera.yml

L'exécution de la commande ansible-playbook /etc/ansible/roles/galera/tasks/galera.yml

lance un playbook Ansible dédié à la configuration cluster Galera pour MariaDB sur deux nœuds



6.4 Créer des bases de données et des tables

6.4.1 Création du playbook create_db_tables.yml

Ce playbook Ansible automatise la création d'une base de données, de tables et l'insertion de données dans ces tables sur le serveur server2.

nano /etc/ansible/roles/galera/tasks/create_db_tables.yml

```
- name: Création Automatisée de Bases de Données et de Tables
  hosts: server2
  become: yes
  vars:
    db_name: 'dbz'
    db_user: 'root'
    db_password: 'vagrant'

  tasks:
    - name: Debug - Afficher les variables
      debug:
        var: item
      with_items:
        - db_name
        - db_user
        - db_password

    - name: Créer la base de données {{ db_name }}
      mysql_db:
        name: "{{ db_name }}"
        state: present
        login_user: "{{ db_user }}"
        login_password: "{{ db_password }}"
      register: db_creation_result

    - name: Créer la table Utilisateurs
      mysql_query:
        login_db: "{{ db_name }}"
        login_user: "{{ db_user }}"
        login_password: "{{ db_password }}"
        query: >
          CREATE TABLE IF NOT EXISTS Utilisateurs (
            Id INT AUTO_INCREMENT PRIMARY KEY,
            Nom VARCHAR(255),
            Email VARCHAR(255),
            Date_inscription DATE
          )
      when: db_creation_result.changed
```

```
- name: Ajouter les utilisateurs dans la table Utilisateurs
  mysql_query:
    login_db: "{{ db_name }}"
    login_user: "{{ db_user }}"
    login_password: "{{ db_password }}"
    query: >
      INSERT INTO Utilisateurs (Nom, Email, Date_inscription) VALUES
      ('goku', 'goku@dbz.sn', NOW()),
      ('gohan', 'gohan@dbz.sn', NOW()),
      ('vegeta', 'vegeta@dbz.sn', NOW())
  when: db_creation_result.changed

- name: Créer la table Connexions
  mysql_query:
    login_db: "{{ db_name }}"
    login_user: "{{ db_user }}"
    login_password: "{{ db_password }}"
    query: >
      CREATE TABLE IF NOT EXISTS Connexions (
        Connexion_id INT AUTO_INCREMENT PRIMARY KEY,
        Utilisateur_id INT,
        Utilisateur_nom VARCHAR(255),
        Utilisateur_email VARCHAR(255),
        Timestamp DATETIME,
        FOREIGN KEY (Utilisateur_id) REFERENCES Utilisateurs(Id)
      )
  when: db_creation_result.changed

- name: Afficher les utilisateurs avant la création de la table Connexions
  mysql_query:
    login_db: "{{ db_name }}"
    login_user: "{{ db_user }}"
    login_password: "{{ db_password }}"
    query: >
      SELECT Nom, Email FROM Utilisateurs
  register: users_result
  when: db_creation_result.changed

- name: Debug - Afficher les résultats de la sélection des utilisateurs
  debug:
    var: users_result.stdout_lines
  when: db_creation_result.changed

- name: Créer les connexions dans la table Connexions
  mysql_query:
    login_db: "{{ db_name }}"
    login_user: "{{ db_user }}"
    login_password: "{{ db_password }}"
    query: >
      INSERT INTO Connexions (Utilisateur_id, Utilisateur_nom, Utilisateur_email, Timestamp)
      SELECT Id, Nom, Email, NOW() FROM Utilisateurs
  when: db_creation_result.changed
```

ansible-playbook /etc/ansible/roles/galera/tasks/create_db_tables.yml

L'exécution de la commande ansible-playbook /etc/ansible/roles/galera/tasks/create_db_tables.yml lance un playbook Ansible dédié à la création automatique de bases de données et de tables dans un environnement Galera.

```
root@srv1:~# ansible-playbook /etc/ansible/roles/galera/tasks/create_db_tables.yml

PLAY [Création Automatisée de Bases de Données et de Tables] ******************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************************************************************
ok: [srv2]

TASK [Debug - Afficher les variables] *****************************************************************************************************************************
ok: [srv2] => (item=db_name) => {
    "ansible_loop_var": "item",
    "item": "db_name"
}
ok: [srv2] => (item=db_user) => {
    "ansible_loop_var": "item",
    "item": "db_user"
}
ok: [srv2] => (item=db_password) => {
    "ansible_loop_var": "item",
    "item": "db_password"
}

TASK [Créer la base de données dbz] *******************************************************************************************************************************
changed: [srv2]

TASK [Créer la table Utilisateurs] ********************************************************************************************************************************
changed: [srv2]

TASK [Ajouter les utilisateurs dans la table Utilisateurs] ********************************************************************************************************
changed: [srv2]

TASK [Créer la table Connexions] **********************************************************************************************************************************
changed: [srv2]

TASK [Afficher les utilisateurs avant la création de la table Connexions] *****************************************************************************************
ok: [srv2]

TASK [Debug - Afficher les résultats de la sélection des utilisateurs] ********************************************************************************************
ok: [srv2] => {
    "users_result.stdout_lines": "VARIABLE IS NOT DEFINED!"
}

TASK [Créer les connexions dans la table Connexions] **************************************************************************************************************
changed: [srv2]

PLAY RECAP ********************************************************************************************************************************************************
srv2                       : ok=9    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

6.4.2 Vérification du cluster galera

mysql -u root -proot -e "SHOW STATUS LIKE 'wsrep_cluster_size'"

MYSQL : C'est le client en ligne de commande pour MySQL.

-U ROOT : Cela spécifie l'utilisateur MySQL à utiliser lors de la connexion, dans ce cas, "root".

-PROOT : CELA SPECIFIE LE MOT DE PASSE A UTILISER LORS DE LA CONNEXION, DANS CE CAS, "ROOT". NOTEZ QUE cette méthode de spécification du mot de passe sur la ligne de commande peut être un risque de sécurité dans un environnement partagé.

-E "SHOW STATUS LIKE 'WSREP_CLUSTER_SIZE'" : Cela spécifie la requête SQL à exécuter. Dans ce cas, la requête est "SHOW STATUS LIKE 'wsrep_cluster_size'",

Cette commande exécute une requête SQL pour afficher la taille du cluster Galera.

La commande si on est déjà connecté à la base de donnais est : 'wsrep_cluster_size';

```
root@srv2:~# mysql -u root -proot -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 2     |
+--------------------+-------+
root@srv2:~#
```

mysql -u root -p

connexion à la base de donnais avec le user root

```
root@srv2:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 95
Server version: 10.6.16-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

show databases ;

Est utilisée pour afficher la liste de toutes les bases de données disponibles sur le serveur MySQL auquel vous êtes connecté

```
MariaDB [(none)]> show databases ;
+--------------------+
| Database           |
+--------------------+
| dbz                |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.000 sec)
```

USE dbz;

Est utilisée pour sélectionner une base de données spécifique avec laquelle vous souhaitez interagir

```
MariaDB [(none)]> USE dbz;
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
```

SELECT * FROM Utilisateurs;

Est une requête SQL qui demande à MariaDB d'afficher toutes les lignes et colonnes de la table appelée "Utilisateurs" dans la base de données actuellement sélectionnée, qui est "dbz" dans votre cas.

```
MariaDB [dbz]> SELECT * FROM Utilisateurs;
+----+--------+---------------+------------------+
| Id | Nom    | Email         | Date_inscription |
+----+--------+---------------+------------------+
|  1 | goku   | goku@dbz.sn   | 2024-02-10       |
|  3 | gohan  | gohan@dbz.sn  | 2024-02-10       |
|  5 | vegeta | vegeta@dbz.sn | 2024-02-10       |
+----+--------+---------------+------------------+
3 rows in set (0.000 sec)
```

SELECT * FROM Connexions;

Est une requête SQL qui demande à MariaDB d'afficher toutes les lignes et colonnes de la table "Connexions" dans la base de données actuellement sélectionnée, qui est "dbz" dans votre cas.

```
MariaDB [dbz]> SELECT * FROM Connexions;
+-------------+---------------+----------------+------------------+---------------------+
| Connexion_id | Utilisateur_id | Utilisateur_nom | Utilisateur_email | Timestamp           |
+-------------+---------------+----------------+------------------+---------------------+
|           1 |             1 | goku           | goku@dbz.sn      | 2024-02-10 16:57:49 |
|           3 |             3 | gohan          | gohan@dbz.sn     | 2024-02-10 16:57:49 |
|           5 |             5 | vegeta         | vegeta@dbz.sn    | 2024-02-10 16:57:49 |
+-------------+---------------+----------------+------------------+---------------------+
3 rows in set (0.000 sec)
```

show status like 'wsrep_cluster_status';

Est une requête SQL spécifique à Galera Cluster dans MariaDB/MySQL. Elle permet de vérifier l'état du cluster Galera, indiquant si le nœud actuel est en cours de synchronisation avec d'autres nœuds du cluster.

```
MariaDB [dbz]> show status like 'wsrep_cluster_status';
+----------------------+---------+
| Variable_name        | Value   |
+----------------------+---------+
| wsrep_cluster_status | Primary |
+----------------------+---------+
1 row in set (0.001 sec)
```

show variables like 'default_storage_engine';

Est une requête SQL qui permet de consulter la valeur actuelle de la variable système default_storage_engine dans le serveur MariaDB/MySQL.

```
MariaDB [dbz]> show status like 'wsrep_local_state_comment';
+---------------------------+--------+
| Variable_name             | Value  |
+---------------------------+--------+
| wsrep_local_state_comment | Synced |
+---------------------------+--------+
1 row in set (0.001 sec)
```

show status like 'wsrep_local_state_comment';

Est une requête SQL spécifique à Galera Cluster dans MariaDB/MySQL. Elle permet d'obtenir des informations détaillées sur l'état local du nœud dans le cluster Galera.
SYNCED: Le nœud a rejoint le cluster et est synchronisé.

```
MariaDB [dbz]> show status like 'wsrep_local_state_comment';
+---------------------------+--------+
| Variable_name             | Value  |
+---------------------------+--------+
| wsrep_local_state_comment | Synced |
+---------------------------+--------+
1 row in set (0.001 sec)
```

show status like 'wsrep_local_recv_queue_avg';

Est une requête SQL spécifique à Galera Cluster dans MariaDB/MySQL. Elle permet d'obtenir des informations sur la taille moyenne de la file de réception locale du nœud dans le cluster Galera.

```
MariaDB [dbz]>  show status like 'wsrep_local_recv_queue_avg';
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| wsrep_local_recv_queue_avg | 0.05  |
+----------------------------+-------+
1 row in set (0.001 sec)
```

6.5 Configuration du backup dans le server backup

Playbook ansible pour automatiser les sauvegardes avec MySQL dump et cron sur le serveur backup

nano /etc/ansible/roles/galera/tasks/backup.yml

```
# Automatisation des Sauvegardes avec Mysqldump et Cron
- name: Automatisation des Sauvegardes avec Mysqldump et Cron
  hosts: server4
  become: true
  vars:
    backup_folder: "/etc/backuptpsql"
    cron_schedule: "*/5 * * * *"

  tasks:
    - name: Create Backup Folder
      file:
        path: "{{ backup_folder }}"
        state: directory
        mode: '0755'

    - name: Create Backup Script
      template:
        src: "/etc/ansible/roles/galera/templates/backup_script.sh.j2"
        dest: "{{ backup_folder }}/backup_script.sh"

    - name: Set Permissions on Backup Script
      file:
        path: "{{ backup_folder }}/backup_script.sh"
        mode: '0755'

    - name: Configure Cron Job
      cron:
        name: "mysql_backup"
        job: "{{ backup_folder }}/backup_script.sh"
        minute: "{{ cron_schedule.split(' ')[0] }}"
        hour: "{{ cron_schedule.split(' ')[1] }}"
        day: "{{ cron_schedule.split(' ')[2] }}"
        month: "{{ cron_schedule.split(' ')[3] }}"
        weekday: "{{ cron_schedule.split(' ')[4] }}"
        user: "root"
```

6.5.1 Script de backup

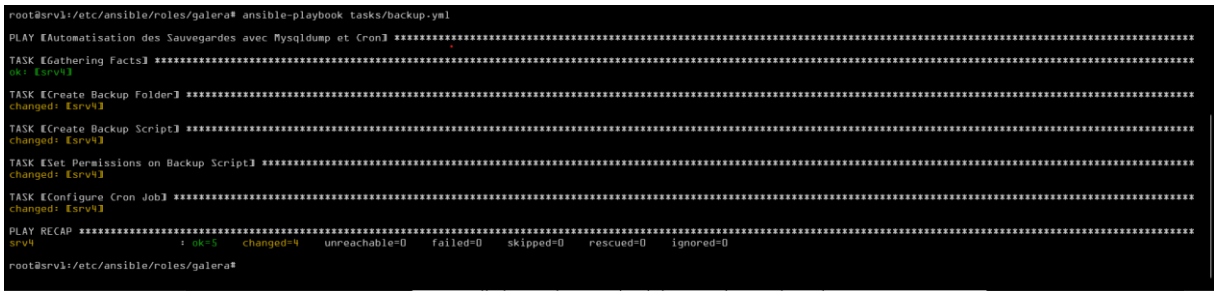
Ce script réalise des sauvegardes de la base de données MySQL spécifiée en utilisant mysqldump et conserve uniquement les deux sauvegardes les plus récentes dans le dossier spécifié.

nano /etc/ansible/roles/galera/templates/backup_script.sh.j2

```
#!/bin/bash
# Définir le dossier de sauvegarde
backup_folder="{{ backup_folder }}"
# Définir le format de date pour les fichiers de sauvegarde
date_format=$(date +"%Y%m%d_%H%M%S")
# Définir les identifiants MySQL
mysql_user="root"
mysql_password="vagrant"
mysql_host="localhost"
mysql_database="dbz"
# Créer une sauvegarde en utilisant mysqldump
mysqldump -u"${mysql_user}" -p"${mysql_password}" -h"${mysql_host}"
"${mysql_database}" > "${backup_folder}/backup_${date_format}.sql"
# Conserver les fichiers avec l'extension .sql
for file in "${backup_folder}"/*.sql; do
  # Vérifier si le fichier a une extension .sql
  if [[ -f "$file" ]]; then
    # Conserver les deux sauvegardes les plus récentes, supprimer les autres
    ls -t "${backup_folder}"/*.sql | tail -n +3 | xargs -I {} rm "{}"
  fi
done
```

6.5.2 Excution playbook de backup

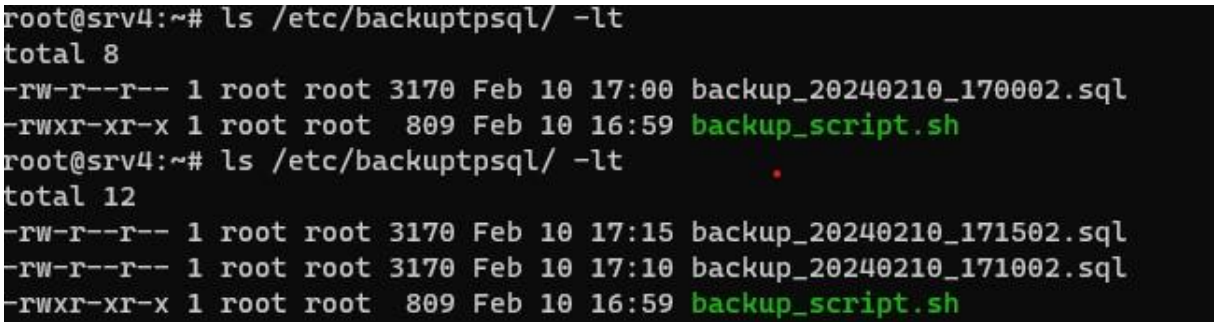
ansible-playbook /etc/ansible/roles/galera/tasks/backup.yml



6.5.1 Vérification des Backup

ls /etc/backuptpsql/ lt

-lt: Options de tri.
-l : Affiche les détails de chaque fichier ou répertoire, y compris les permissions, le nombre de liens, le propriétaire, le groupe, la taille, et la date de dernière modification.
-t : Trie les fichiers et répertoires en fonction de la date et de l'heure de la dernière modification, du plus récent au plus ancien.



7 Annex

```
root@srv1:~# tree /etc/ansible/roles/galera/
/etc/ansible/roles/galera/
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   ├── backup.yml
│   ├── create_db_tables.yml
│   ├── galera.yml
│   └── main.yml
├── templates
│   ├── backup_script.sh.j2
│   ├── galera2.cnf.j2
│   ├── galera3.cnf.j2
│   └── vars.yml
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

7 directories, 15 files
root@srv1:~#
```

Vagrantfile

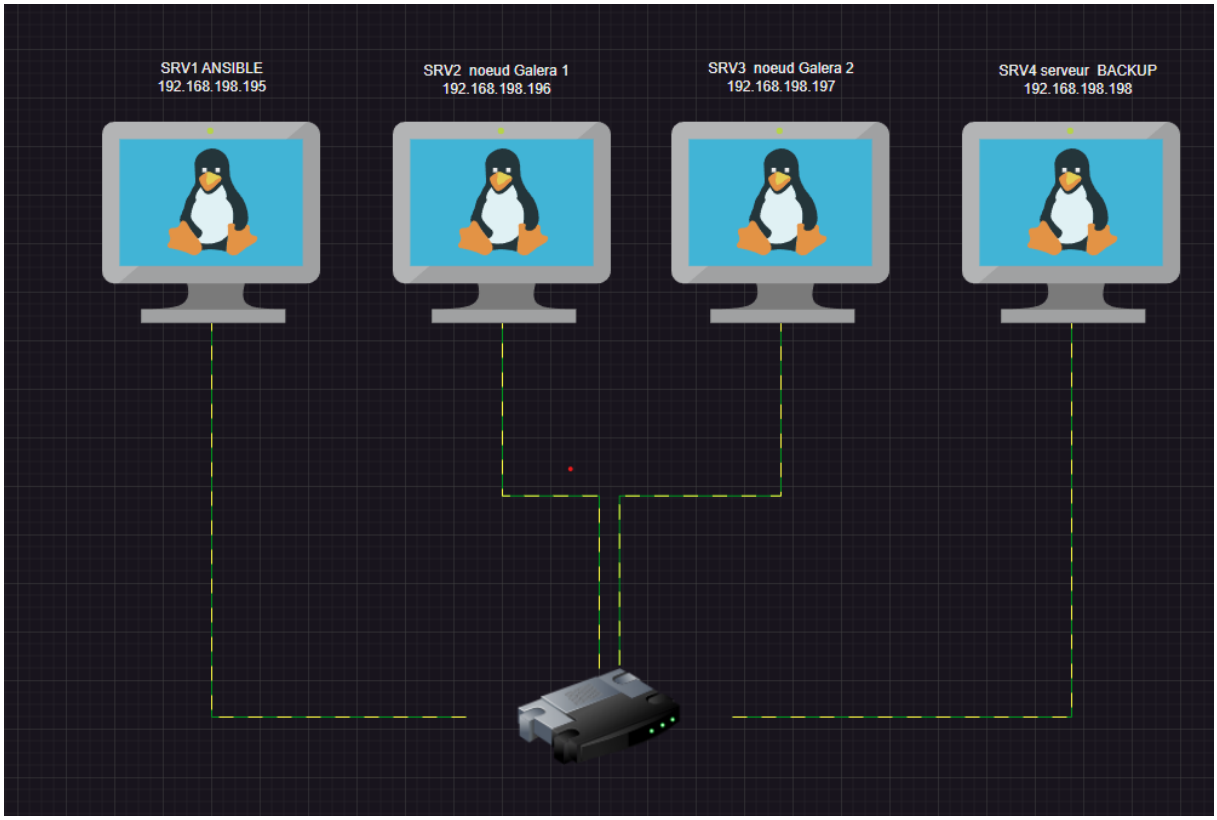
```
# -*- mode : ruby -*-
# vi: set ft-ruby :

RAM = 2048
CPU = 2


Vagrant.configure("2") do |config|
  (1..4).each do |i|
    config.vm.define "srv#{i}" do |srv|
      srv.vm.box = "waounde221/ubuntu"
      srv.vm.hostname = "srv#{i}"
      srv.vm.provider "vmware_desktop" do |v|
        v.gui = true
        v.memory = RAM
        v.cpus = CPU
      end
    end
  end
end
```

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  PORTS

PS D:\GitHub\Nouveau dossier\azds> vagrant up --provider vmware_desktop
Bringing machine 'srv1' up with 'vmware_desktop' provider...
Bringing machine 'srv2' up with 'vmware_desktop' provider...
Bringing machine 'srv3' up with 'vmware_desktop' provider...
Bringing machine 'srv4' up with 'vmware_desktop' provider...
==> srv1: Cloning VMware VM: 'waounde221/ubuntu'. This can take some time...
```

Nano /etc/ssh/sshd_config

Pour activer le port 22 et le PermitRootLogin

```
  GNU nano 6.2                                                    /etc/ssh/sshd_config *
#       $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```



Commande Vagrant

vagrant --version: Affiche la version actuelle de Vagrant installée sur votre système.

vagrant list-commands: Affiche une liste de toutes les commandes Vagrant disponibles.

vagrant global-status: Affiche l'état global de toutes les machines Vagrant sur votre système, indépendamment du répertoire de travail actuel.

vagrant global-status --prune: Actualise l'état global en supprimant les références aux machines Vagrant qui n'existent plus.

vagrant global-status -h: Affiche l'aide pour la commande vagrant global-status.

vagrant init waounde221/ubuntu: Initialise un nouveau fichier Vagrantfile dans le répertoire actuel, basé sur la box spécifiée.

vagrant init waounde221/ubuntu -m: Initialise un nouveau fichier Vagrantfile avec des options spécifiées pour la box.

vagrant plugin install vagrant-vmware-desktop: Installe le plugin Vagrant pour la prise en charge de VMware Desktop.

vagrant plugin list: Affiche la liste des plugins Vagrant installés.

vagrant up --provider=vmware_workstation: Démarre la machine Vagrant en utilisant le fournisseur VMware Workstation.

vagrant halt: Arrête la machine Vagrant en douceur.

vagrant destroy -f: Détruit complètement la machine Vagrant, y compris toutes les ressources associées, sans confirmation.

vagrant box add waounde221/ubuntu https://app.vagrantup.com/waounde221/boxes/ubuntu : Ajoute une nouvelle box Vagrant à partir d'un fichier téléchargé depuis un lien spécifié.

vagrant box list: Affiche la liste des box Vagrant disponibles en local.

vagrant box remove waounde221/ubuntu: Supprime une box Vagrant.

vagrant package --output waounde221/ubuntu: est utilisée pour créer une nouvelle box Vagrant en empaquetant une machine virtuelle existante.

Cette commande générera une nouvelle box Vagrant en utilisant la machine virtuelle actuellement configurée dans votre environnement Vagrant. La box résultante sera enregistrée dans un fichier avec le nom spécifié, "waounde221/ubuntu" dans cet exemple. Vous pouvez ajuster le nom du fichier de sortie selon vos besoins.