المدرسة الحسنية للأشغال العمومية
ECOLE HASSANIA DES TRAVAUX PUBLICS

Rapport: Projet Linux

Automatisation par *Ansible* - Réplication de *MySQL*

Réalisation:

✧ BENHADDOU Lahcen

✧ ERRAZI Fatima-Zahra

✧ AJALE Saad

Sommaire

Introduction

Le cahier de charge demande la réplication bidirectionnelle de *MySQL* via *Ansible* pour automatiser le maintien des données entre un Desktop et deux serveurs *Linux*.

Pour assurer cette automatisation, nous avons créer un menu qui englobe plusieurs options à savoir: la réplication monodirectionnelle et bidirectionnelle de *MySQL*, effectuer les différentes modifications dans les bases de données de *MySQL* en se servant d'*Ansible*.

Présentation du menu

Le menu s'affiche comme suit:

L'option1: Elle sert à configurer la réplication bidirectionnelle entre le nombre désiré de machines.

L'option2:Elle sert à configurer la réplication monodirectionnelle entre le nombre désiré de machines.

L'option3: Elle sert à apporter n'importe quelle types de modification au niveau de MySQL sans avoir à connaitre la syntaxe des commandes MySQL.

L'option4: Elle permet l'installation d'Ansible, MySQL et de Python dans la machine à partir de laquelle on effectue les configurations.

L'option5: Elle sert à quitter le menu et revenir au terminal.

Valeur ajoutée

✧ Menu:

Menu interactif dédié à n'importe quel utilisateur.

✧ Conformité au cas réel:

La réplication bidirectionnelle MySQL entre n'importe quel nombre désiré de machines.

La réplication monodirectionnelle MySQL entre n'importe quel nombre désiré de machines.

✧ Évolutivité:

Les codes sont basés sur la création des fonctions ce qui simplifiera toute susceptible modification et évolution.

✧ Maintenance:

Nous avons opté pour l'utilisation de chemins relatifs au répertoire de notre projet afin de garantir le bon fonctionnement de notre application sur n'importe quelle machine, sans nécessiter de modifications fréquentes dans le script principal qui contient souvent

les chemins des scripts.

✧ Optimisation (de la mémoire et du temps d'exécution):

La logique du code ne crée pas pour toute machine à configurer des scripts qui sont propre à elle, mais plutôt on se base sur un nombre fixe de scripts pour quel que soit le nombre de machines.

✧ Simplification de l'utilisation:

Le menu avec toutes ses options est dédié à tout type d'utilisateurs sans nécessité de prérequis préalables.

✧ Traitement des possibilités d'erreurs:

L'entrée d'une adresse IP incorrecte n'influence pas sur la configuration des autres machines. Et l'entrée d'un nombre inexistant pour le choix des options redonne la main de la saisie d'un nombre valide sans aucune erreur.

Outils de travail

- ✓ Oracle VM VirtualBox : un logiciel de virtualisation open source qui permet aux utilisateurs de créer et d'exécuter des machines virtuelles sur leurs systèmes.
- ✓ Ansible : est un outil d'automatisation informatique en code source ouvert, utilisée pour simplifier les tâches informatiques répétitives en définissant les taches dans des fichiers YAML.
- ✓ Secure Shell (SSH) : est un protocole de réseau sécurisé permettant à un utilisateur d'accéder à des systèmes distants de manière sécurisée et d'y gérer des services
- ✓ Ms Dos : acronyme de Microsoft Disk Operating System, est un système d'exploitation en ligne de commande développé par Microsoft.
- ✓ SHELL : une interface en ligne de commande entre l'utilisateur et le noyau du système d'exploitation Linux.

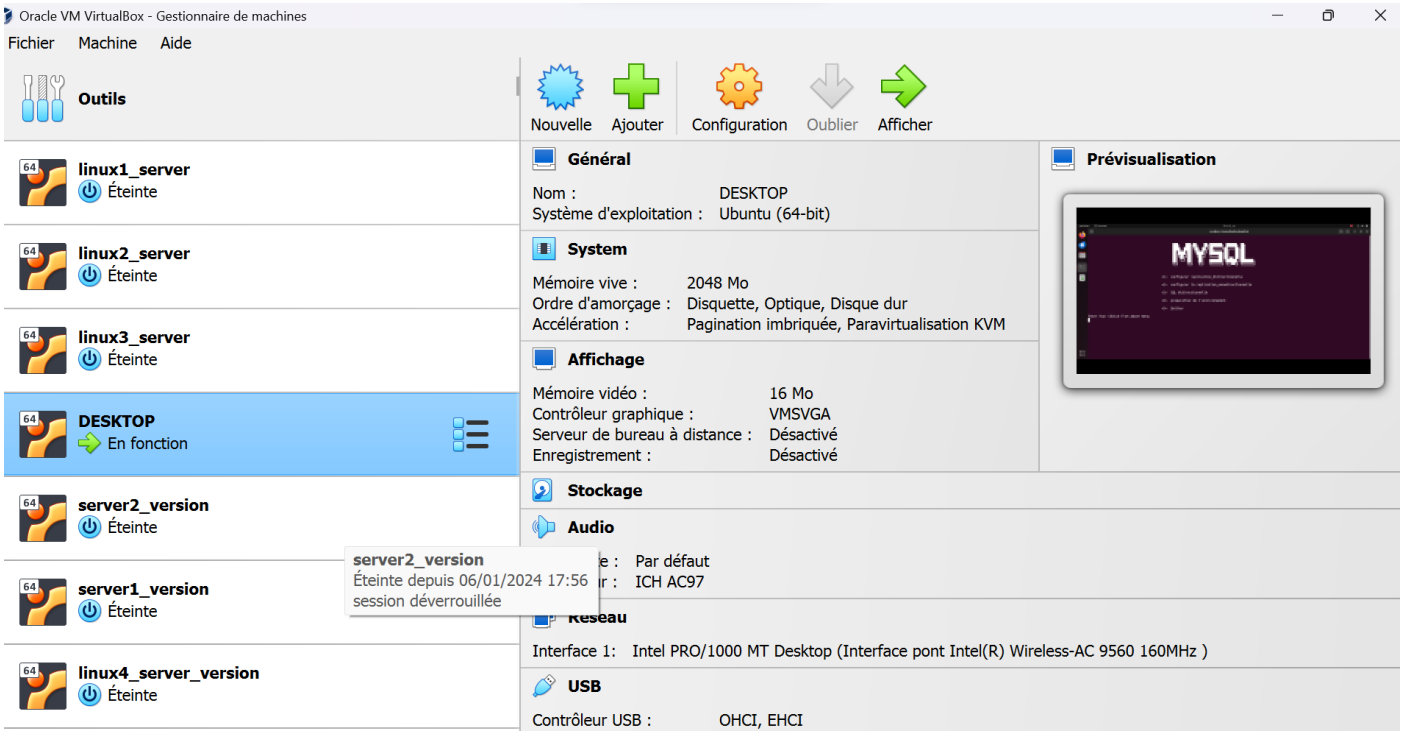
- ✓ Python : est un langage de programmation de haut niveau.
- ✓ MySQL : est un système de gestion de base de données.

- ✓ YAML :(YAML Ain't Markup Language ou YAML Ain't a Markup Language) est un format de sérialisation de données léger, lisible par l'homme et facile à écrire. Il est souvent utilisé pour la configuration de fichiers et l'échange de données entre langages de programmation.

Préparation de l'environnement de travail

1. Création des machines virtuelles Linux

En utilisant VirtualBox, nous créons le nombre de machines nécessaire pour cette configuration.



2. Installation et configuration de SSH dans les machines

Étant connecté en root, on exécute successivement les commandes suivantes pour mettre à jour le package des outils à utiliser :

```
root@vm:/home/dba# apt update
```

```
root@vm:/home/dba# apt upgrade
```

```
root@vm:/home/dba# apt install -y net-tools
```

Ensuite, on utilise la commande suivante pour l'installer:

```
root@vm:/home/dba# apt install -y openssh-server
```

3. Installation d'Ansible sur la machine assurant la configuration

La commande est la suivante:

```
root@vm:~# apt install -y ansible
```

4. Configuration des fichiers d'Ansible

Pour accéder au fichier inventaire d'Ansible où on ajoute les adresses IP des machines distantes, on utilise le chemin suivant:

```
dba@vm:~$ nano /etc/ansible/hosts
```

5. Intallation de MySQL

La commande d'installation de MySQL est :

```
root@vm:~# apt install -y mysql-server
```

Notons que dans le projet nous avons utilisé un script préparant l'environnement d'une manière automatique.

Étapes de travail

✧ Tester et découvrir le fonctionnement d'Ansible à travers l'exécution de simples scripts dans la machine distante.

✧ Configurations manuelles des réplications monodirectionnelle et bidirectionnelle de MySQL entre deux machines.

✧ Extension de la réplication monodirectionnelle à plusieurs machines.

✧ Essai d'extension de la réplication bidirectionelle par application de la méthode anneau entre 3 machines.

✧ Détection de la complexité de cette application et présence de conflits.

✧ Automatisation initiale de la détection de bin-log et position par gtid_mode.

✧ Mise en application de la méthode multisource pour étendre la réplication bidirectionnelle à plusieurs machines.

✧ Création des scripts de commandes MySQL pour faciliter toute

modification susceptible.

✧ Rassembler les scripts et création du menu.

Réalisation

Le script de base de ce projet repose sur le menu suivant :

```
ch=0
until [ "$ch" -eq 5 ]; do
clear
design
    echo -e "                                   <1>  configurer replication_bidirectionnelle\n\n
   <3>  SQL bidirectionnelle\n\n                                    <4>  preparation de l'environnement\n\n
    echo -e "Enter Your Choice from above menu: "
    read -r ch
    clear

    if [ $ch -gt 5 ] || [ $ch -eq 0 ]; then
        echo "<-> veillez choisir un chiffre entre 1 et 5\n"
    fi

    case $ch in
    1) replication_bidirectionnelle;;
    2) replication_monodirectionnelle ;;
    3) sql ;;
    4) environment_configuration ;;
    esac

done
```

Nous avons opté pour l'utilisation de la variable "project directory" en tant que chemin absolu de notre projet, ce choix vise à minimiser les modifications nécessaires dans les chemins d'autres fichiers en cas de déplacement du projet.

Le menu nous conduit vers l'une des fonctions implémentées :

✧ Replication Monodirectionnelle

```
replication_monodirectionnelle() {
read -p "<-> Entrez le nom de la base de données : " DB_NAME
read -p "<-> Entrez l'adresse ip du maitre : " master_ip
ssh-copy-id dba@"$master_ip"
configuration_environnement_machine $master_ip
configuration_master_master
echo -e "<-> combien de machines esclaves voulez vous connecter ?\n"
read -r nserver
nserver=$(expr $nserver + 1)
i=1
#myipadress=$(ifconfig | grep -oE 'inet [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+'
  | awk '{print $2}'| head -n 1)
until [ "$i" -eq "$nserver" ]; do
        echo -e "<-> donner l'adresse ip N° $i :"
        read -r ipadress
        ssh-copy-id dba@"$ipadress"
        configuration_environnement_machine $ipadress
        i=$(expr $i + 1)
        configuration_slave_master
        configuration_slave_slave
done

}
```

Cette fonction sollicite de l'utilisateur l'adresse IP du maître, ainsi que le nombre d'esclaves et leurs adresses IP. À l'aide d'une boucle "until", elle configure chaque machine en utilisant les fonctions configuration _master _master, configuration _master _slave, configuration _slave _master et `configuration _slave _slave`. Chacune de ces fonctions ajuste les fichiers exécutables en utilisant un playbook implémenté en langage YAML.

Cette fonction modifie aussi l'inventaire pour l'adapter à chaque machine distante après la configuration.

✧ Réplication bidirectionnelle

```
14
15 project_directory="/home/dba/Desktop/final"
16 replication_bidirectionnelle() {
17 read -p "<-> Entrez le nom de la base de données : " DB_NAME
18 read -p "<-> Entrez l'adresse ip de la premiere machine : " master_ip
19 ssh-copy-id dba@"$master_ip"
20 configuration_environnement_machine $master_ip
21 configuration_master_master
22 echo -e "<-> combien de machines voulez vous connecter ?\n"
23 read -r nserver
24 nserver=$(expr $nserver + 1)
25 i=1
26 #myipadress=$(ifconfig | grep -oE 'inet [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+'
   | awk '{print $2}'| head -n 1)
27 until [ "$i" -eq "$nserver" ]; do
28         i=$(expr $i + 1)
29         echo -e "<-> donner l'adresse ip N° $i :"
30         read -r ipadress
31         ssh-copy-id dba@"$ipadress"
32         configuration_environnement_machine $ipadress
33         configuration_slave_master
34         configuration_slave_slave
35         yml_file="logfile.yml"
36         logfile_inventory="$project_directory/$yml_file"
37         resultfile="/result.txt"
38         file_result="        dest: ${project_directory}${resultfile}"
39         sed -i "27s|.*|$file_result|" $logfile_inventory
40         ansible-playbook $logfile_inventory
41         configuration_master_slave
42 done
43
44 }
```

La fonction bidirectionnelle ressemble à la fonction monodirectionnelle, à la différence qu'elle intègre la configuration multisource en utilisant la fonction `configuration_master_slave` avec l'utilisation de canaux. Cela intervient après l'obtention des informations des journaux bin log de la machine configurée initialement en tant qu'esclave.

✧ configuration_slave_master

```
84 configuration_slave_master() {
85 config_file="config.sh"
86 result="$project_directory/$config_file"
87 chainedb="DB_NAME="
88 DBch="${chainedb}${DB_NAME} "
89 sed -i "3s/.*/$DBch/" $result
90 chaineid="id=$i"
91 sed -i "2s/.*/$chaineid/" $result
92 UTILISATEUR_MYSQL="UTILISATEUR_MYSQL=slave1"
93 sed -i "4s/.*/$UTILISATEUR_MYSQL/" $result
94
95 yml_file="inventory.yml"
96 inventory="$project_directory/$yml_file"
97 file_copy="        src: $result"
98 sed -i "13s|.*|$file_copy|" $inventory
99 file_copy="        bash /home/dba/$config_file"
100 sed -i "23s|.*|$file_copy|" $inventory
101 file_copy="        src: ${project_directory}/sql.sh"
102 sed -i "18s|.*|$file_copy|" $inventory
103 file_delete="        path: /home/dba/$config_file"
104 sed -i "26s|.*|$file_delete|" $inventory
105
106 chainebi="[servers]\nserver1 ansible_host="
107 resultat="${chainebi}${ipadress} "
108 echo -e "$resultat" > /etc/ansible/hosts
109 ansible-playbook $inventory
110
111 }
```

La fonction `configuration_slave_master` modifie les paramètres du fichier `config.sh` en ajoutant l'adresse IP, l'ID du slave, et le nom de la base de données pour la réplication.

✧ configuration_master_master

```
112 configuration_master_master(){
113 config_file="config.sh"
114 result="$project_directory/$config_file"
115 chainedb="DB_NAME="
116 DBch="${chainedb}${DB_NAME} "
117 sed -i "3s/.*/$DBch/" $result
118 chaineid="id=1"
119 sed -i "2s/.*/$chaineid/" $result
120 UTILISATEUR_MYSQL="UTILISATEUR_MYSQL=slave1"
121 sed -i "4s/.*/$UTILISATEUR_MYSQL/" $result
122
123 yml_file="inventory.yml"
124 inventory="$project_directory/$yml_file"
125 file_copy="        src: $result"
126 sed -i "13s|.*|$file_copy|" $inventory
127 file_copy="        bash /home/dba/$config_file"
128 sed -i "23s|.*|$file_copy|" $inventory
129 file_copy="        src: ${project_directory}/sql.sh"
130 sed -i "18s|.*|$file_copy|" $inventory
131 file_delete="        path: /home/dba/$config_file"
132 sed -i "26s|.*|$file_delete|" $inventory
133
134
135 chainebi="[servers]\nserver1 ansible_host="
136 resultat="${chainebi}${master_ip} "
137 echo -e "$resultat" > /etc/ansible/hosts
138 ansible-playbook $inventory
139
140 }
```

Cette fonction modifie les paramètres du fichier `config.sh` en ajoutant l'identifiant, l’adresse ip du master et le nom de la base de données pour la réplication.

- ✧ configuration_slave_slave

```
142 configuration_slave_slave() {
143 chainebi="[servers]\nserver1 ansible_host="
144 resultat="${chainebi}${master_ip} "
145 echo -e "$resultat" > /etc/ansible/hosts
146
147 yml_file="logfile.yml"
148 logfile_inventory="$project_directory/$yml_file"
149 resultfile="/result.txt"
150 file_result="        dest: ${project_directory}${resultfile}"
151 sed -i "27s|.*|$file_result|" $logfile_inventory
152 ansible-playbook $logfile_inventory
153
154 logpos_file="result.txt"
155 nom_du_fichier="$project_directory/$logpos_file"
156 BINLOG_POSITION=$(tail -n 1 "$nom_du_fichier")
157 BINLOG_FILE_NAME=$(head -n 1 "$nom_du_fichier")
158
159 slave_file="slave.sh"
160 result="$project_directory/$slave_file"
161 master_valeur_File="valeur_File=$BINLOG_FILE_NAME"
162 sed -i "3s/.*/$master_valeur_File/" $result
163 master_valeur_Position="valeur_Position=$BINLOG_POSITION"
164 sed -i "4s/.*/$master_valeur_Position/" $result
165 slave_channel="channel=channel1"
166 sed -i "5s/.*/$slave_channel/" $result
167 #myipadress=$(ifconfig | grep -oE 'inet [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+' | awk '{print $2}'| head -n 1)
168 chaineipmaster="ipmaster="
169 masteripch="${chaineipmaster}${master_ip} "
170 sed -i "2s/.*/$masteripch/" $result
171
172 yml_file="inventory.yml"
173 inventory="$project_directory/$yml_file"
174 file_copy="        src: $result"
175 sed -i "13s|.*|$file_copy|" $inventory
176 file_copy="        bash /home/dba/$slave_file"
177 sed -i "23s|.*|$file_copy|" $inventory
178 file_delete="        path: /home/dba/$slave_file"
179 sed -i "26s|.*|$file_delete|" $inventory
180
181 chainebi="[servers]\nserver1 ansible_host="
182 resultat="${chainebi}${ipadress} "
```

La fonction configuration_slave_slave modifie les paramètres du fichier slave.sh en ajoutant l'adresse IP du maître et le nom du canal.

✧ configuration_master_slave`

```
186 configuration_master_slave() {
187 slave_file="result.txt"
188 nom_du_fichier="$project_directory/$slave_file"
189 BINLOG_POSITION=$(tail -n 1 "$nom_du_fichier")
190 BINLOG_FILE_NAME=$(head -n 1 "$nom_du_fichier")
191 slave_file="slave.sh"
192 result="$project_directory/$slave_file"
193 master_valeur_File="valeur_File=$BINLOG_FILE_NAME"
194 sed -i "3s/.*/$master_valeur_File/" $result
195 master_valeur_Position="valeur_Position=$BINLOG_POSITION"
196 sed -i "4s/.*/$master_valeur_Position/" $result
197 slave_channel="channel=channel$i"
198 sed -i "5s/.*/$slave_channel/" $result
199 chaineipmaster="ipmaster="
200 ipadressch="${chaineipmaster}${ipadress} "
201 sed -i "2s/.*/$ipadressch/" $result
202
203 yml_file="inventory.yml"
204 inventory="$project_directory/$yml_file"
205 file_copy="        src: $result"
206 sed -i "13s|.*|$file_copy|" $inventory
207 file_copy="        bash /home/dba/$slave_file"
208 sed -i "23s|.*|$file_copy|" $inventory
209 file_delete="        path: /home/dba/$slave_file"
210 sed -i "26s|.*|$file_delete|" $inventory
211
212 chainebi="[servers]\nserver1 ansible_host="
213 resultat="${chainebi}${master_ip} "
214 echo -e "$resultat" > /etc/ansible/hosts
215 ansible-playbook $inventory
216 }
```

La fonction `configuration_master_slave` modifie le fichier `slave.sh` en ajoutant l'adresse IP de chaque esclave et le nom du canal.

Les quatre fonctions, configuration _master _master, configuration _master _slave, configuration _slave _master,et configuration _slave _slave, modifient les fichiers d'inventaire pour y ajouter l'adresse IP de la machine destinée chaque configuration respective.

✧ configuration de l'environnement

```
69 configuration_environnement_machine(){
70         ipadress=$1
71         echo "<-> Voulez-vous configurer cette machine? (y/n)"
72         read response
73         if [ "$response" == "y" ]; then
74         chainebi="[servers]\nserver1 ansible_host="
75         resultat="${chainebi}${ipadress} "
76         echo -e "$resultat" > /etc/ansible/hosts
77         configuremachine_file="env.yml"
78         inventoryconfig="$project_directory/$configuremachine_file"
79         ansible-playbook $inventoryconfig
80         fi
81
82 }
```

La fonction de configuration de l'environnement de la machine installe les outils nécessaires sur la machine configuratrice, tels que MySQL, Ansible et SSH, dans le cas où le configurateur ne dispose pas de ces bibliothèques.

✧ Sql

```
218 sql() {
219 slave_file="sql.sh"
220 sql_file="$project_directory/$slave_file"
221 bash $slave_file
222 }
223
```

La fonction SQL exécute le fichier `sql.sh`, lequel déclenche à son tour un menu détaillé pour chaque requête MySQL.

Autres fichiers

Les fichiers playbook servent cependant à configurer la machine distante en exécutant une tâche grâce au transfert des fichiers exécutables. Ils assurent le processus d'envoi, d'exécution, puis de suppression ultérieure de ces fichiers sur la machine distante.

```yaml
---

- name: Exécution de commandes
  hosts: servers
  remote_user: dba
  become: true
  become_flags: "-S"
  vars:
    ansible_become_pass: "test"

  tasks:
    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Install MySQL Server
      apt:
        name: mysql-server
        state: present

    - name: Install Python
      apt:
        name: python3
        state: present
```

```yaml
---
- name: Exécution de commandes
  hosts: servers
  remote_user: dba
  become: true  # Utiliser le privilège d'administration (sudo)
  become_flags: "-S"

  vars:
    ansible_become_pass: "test"
  tasks:
    - name: Copier le fichier
      copy:
        src: /home/dba/Desktop/final/slave.sh
        dest: /home/dba
        mode: 0755
    - name: Copier le fichier sql.sh
      copy:
        src: /home/dba/Desktop/final/sql.sh
        dest: /home/dba
        mode: 0755
    - name: Exécution
      shell: |
        bash /home/dba/slave.sh
    - name: Supprimer un fichier
      ansible.builtin.file:
        path: /home/dba/slave.sh
        state: absent
```

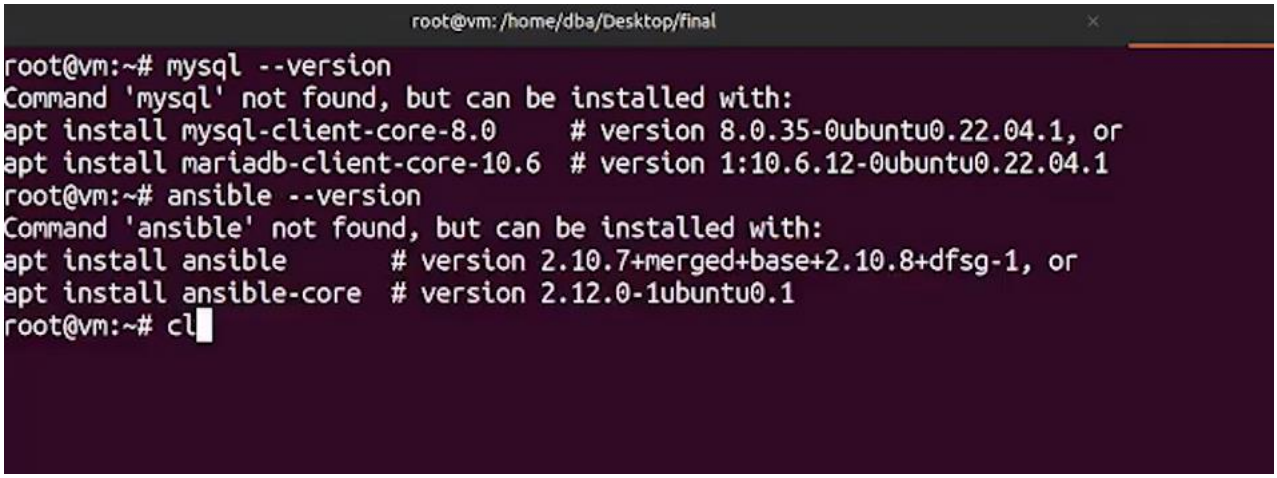
```yaml
---
- name: Exécution de commandes et récupération de résultats
  hosts: servers
  remote_user: dba
  become: true
  become_flags: "-S"

  vars:
    ansible_become_pass: "test"
  tasks:
    - name: Exécuter le script shell sur la machine distante
      shell: |
        MYSQL_USER="root"
        MYSQL_PASSWORD="test"
        SHOW_MASTER_STATUS=$(mysql -u"$MYSQL_USER" -p"$MYSQL_PASSWORD" -e "SHOW MASTER STATUS" --skip-column-names)
        BINLOG_FILE_NAME=$(echo "$SHOW_MASTER_STATUS" | awk '{print $1}')
        BINLOG_POSITION=$(echo "$SHOW_MASTER_STATUS" | awk '{print $2}')
        FIRST_LINE=$(echo "$BINLOG_FILE_NAME" | head -n 1)
        echo "$FIRST_LINE"
        echo "$BINLOG_POSITION"
      register: resultat_commande

    - name: Stocker les résultats localement
      local_action:
        module: copy
        content: "{{ resultat_commande.stdout }}"
        dest: /home/dba/Desktop/final/result.txt
```

Application

Un exemple d'exécution de tout le processus

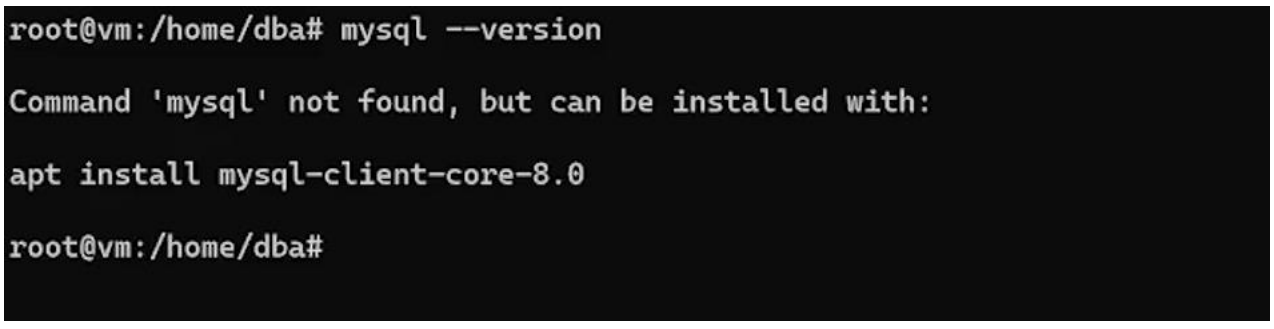
- Entamons l'application à partir d'un Desktop

- MySQL et Ansible ne sont pas installés sur la machine



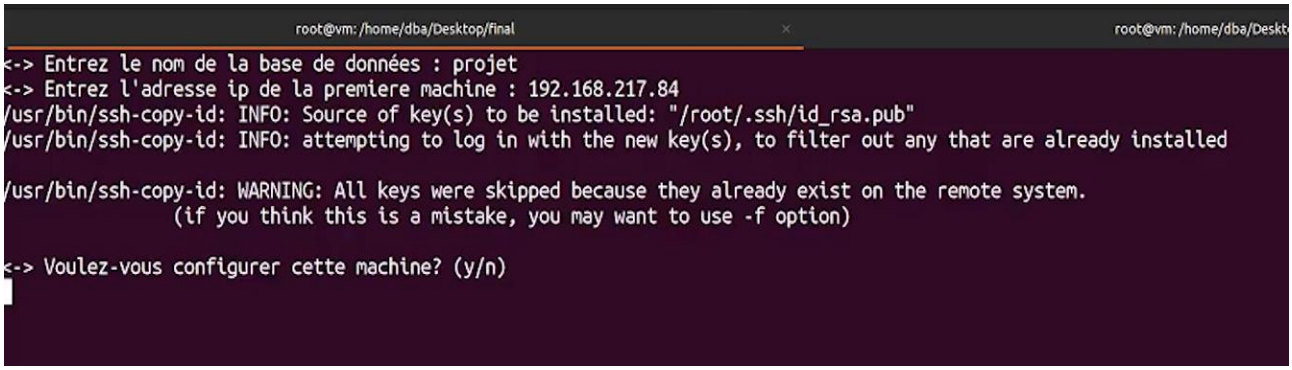
- En choisissant l'option 4 préparation de l'environnement, ansible et MySQL sont installés

- Vérification que MySQL n'est pas installé sur les autres machines dans le but de montrer que les scripts utilisé peuvent la faire



- On revient au menu sur le Desktop et on choisit le type de configuration souhaitée

- Cet écran s’affiche pour demander si on veut configurer(installation MySQL et Python) la machine qu’on vient d’ajouter son adresse IP



- Il demande combien de machines on veut configurer.

  On entre l’adresse ip de la machine

  distante

- En revenant à la machine distante, on exécute un script shell ‘sql.sh’ qui est envoyé par ansible, le menu suivant s’affiche :

- En manipulant les options proposées dans le menu on peut effectuer tout type de modifications .

Erreurs

➢ Autorisation du clé public:

Lors des tests avec Ansible pour la configuration du projet, une erreur est survenue lors de la tentative d'envoi de fichiers vers les serveurs cibles.

L'erreur spécifique était liée à "une défaillance lors de la transmission des fichiers avec le playbook".

Pour résoudre cette erreur, nous avons d'abord confirmé que la clé SSH était correctement configurée sur les serveurs cibles en utilisant la commande ssh-copy-id. Cela assure que la clé publique est autorisée sur chaque machine distante.

➢ Authentification privilégiée avec Ansible:

nous avons également rencontré une problématique liée à l'exécution de tâches en tant que sudo sur les machines distantes. Cette difficulté est apparue lorsque nous avons tenté de configurer des tâches nécessitant des privilèges

Pour résoudre ce problème, nous avons identifié que l'ajout des modules become et become_method dans nos playbooks était nécessaire. Ces modules permettent à Ansible de s'authentifier en tant qu'utilisateur sudo sur les machines distantes, autorisant ainsi

l'exécution de tâches avec des privilèges

➢ Gestion des journaux binaires:

Lors de la configuration initiale de la réplication bidirectionnelle MySQL, nous avons rencontré une difficulté liée à des conflits dans la gestion des journaux binaires (log bin). Cette problématique a engendré erreur fatale, impactant directement la synchronisation bidirectionnelle entre nos serveurs MySQL.

Pour résoudre ces conflits, nous avons opté pour l'utilisation de GTID. En activant GTID, chaque transaction a été identifiée de manière unique, simplifiant la gestion des journaux binaires.

➢ Libération des canaux lors de la réplication multi source:

Au cours de la mise en place de la réplication bidirectionnelle multisource, nous avons fait face à un problème majeur : les canaux de réplication étaient occupés par d'autres machines, entravant ainsi notre capacité à établir une réplication stable entre nos sources MySQL.

Face à l'occupation des canaux de réplication par d'autres machines lors de notre configuration de réplication bidirectionnelle multisource, nous avons entrepris une démarche

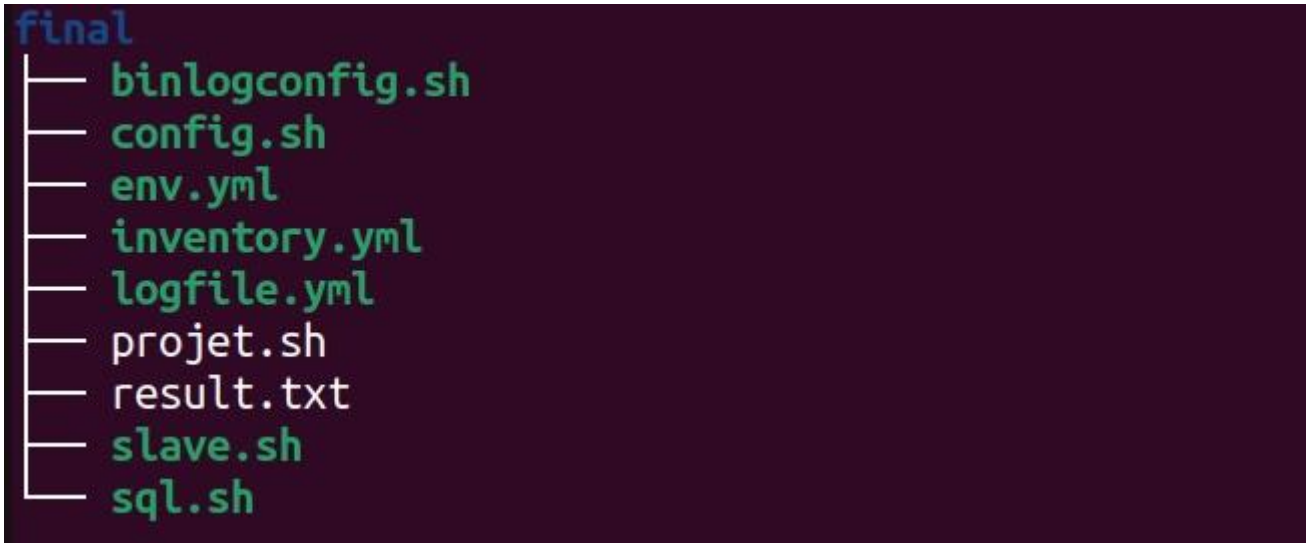
résolue pour résoudre ce conflit. Optant pour la solution radicale, nous avons utilisé la commande MySQL RESET SLAVE ALL pour réinitialiser complètement la configuration de réplication. Cette action a permis de libérer les canaux occupés et d'effacer toute configuration existante, établissant ainsi une base propre et prête à accueillir notre réplication bidirectionnelle multisource. Cette approche concise a démontré son efficacité en résolvant rapidement le problème d'occupation des canaux, facilitant ainsi une réinitialisation complète et une transition réussie vers une configuration optimale

➢ Connextion au cmd (ssh):

Dans le processus de connexion à SSH pour accéder à la commande (cmd), nous avons rencontré une erreur persistante de connexion, créant des obstacles à l'exécution fluide des opérations via SSH.

En utilisant la commande ssh-keygen -f /root/.ssh/known_hosts -R, nous avons supprimé l'entrée de l'hôte problématique dans le fichier known_hosts. Cette action a permis de réinitialiser la connexion SSH et d'éliminer les obstacles liés à l'erreur de connexion.

Annexe


final
binlogconfig.sh
config.sh
env.yml
inventory.yml
logfile.yml
projet.sh
result.txt
slave.sh
sql.sh