ANSIBLE

Voici un projet de déploiement d'infrastructure web automatisée via ansible.

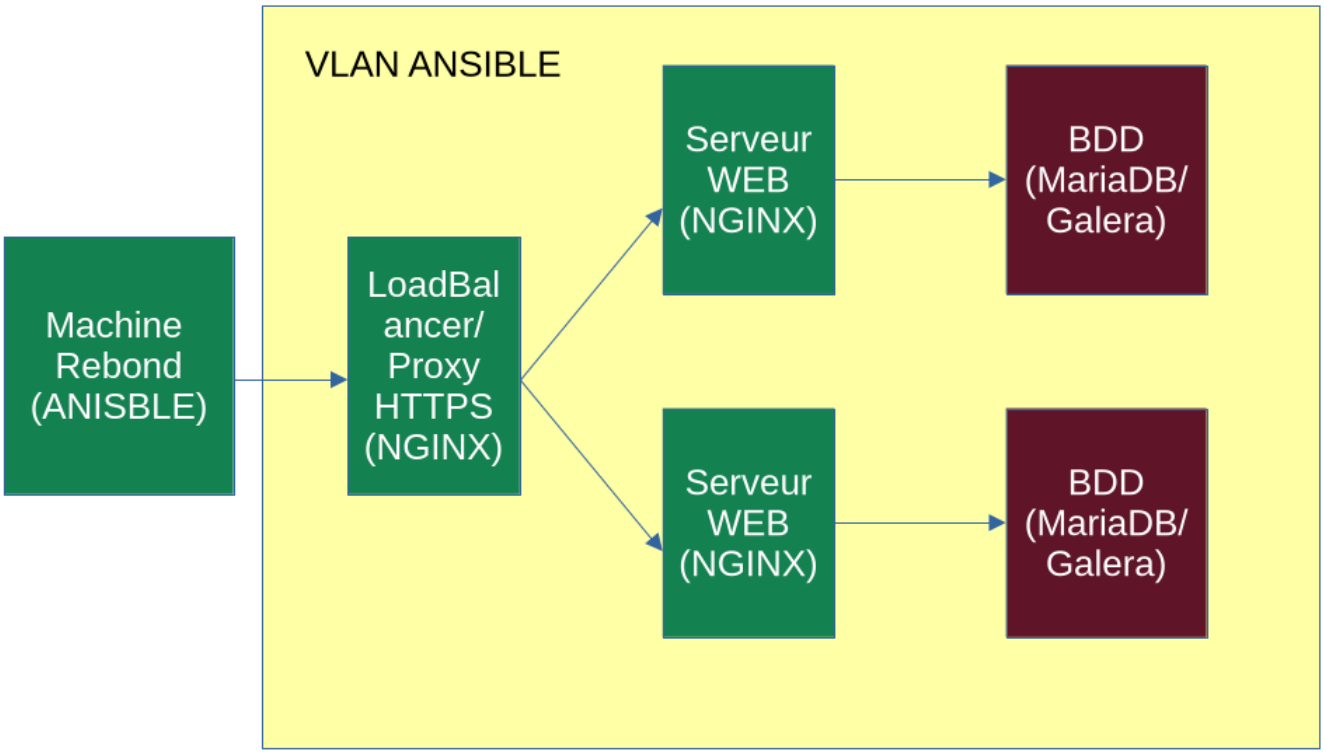
Ansible est une plateforme permettant d'automatiser la gestion et la configuration d'actifs dans une infrastructure.

Contexte du projet

Le but de ce projet, va être de déployer de manière automatisée une infrastructure web redondante. Voici les différentes étapes :

1. Création d'un script déployant automatiquement des conteneurs sur l'infrastructure.
2. Configuration d'un loadbalancer entre deux serveurs web.
3. Configuration de deux serveurs Web
4. Configuration de deux serveurs de base de données

Schéma de l'infrastructure



Script de déploiement

Le script ci-dessous permet de vérifier les ID des conteneurs déjà présents et d'en créer de nouveaux avec les ID suivants.

```
#!/bin/bash
#Initialisation de J pour les adresses IP
j=2
for i in {1..5}
do
    #Liste des conteneurs existants
    value=$(qm list | tail -1 | cut -d " " -f8)
    pct=$((value + i))

    #Création des conteneurs et ajout de la clé de la machine rebond
    pct create $pct local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst --
storage local-lvm --rootfs 8 --hostname "slave-$i" --memory 1024 --cores 1 -
```

```
-net0 name=eth0,bridge=vmbr3,tag=20,ip="XX.XX.XX.$j/24",gw="XX.XX.XX.XX" --
ssh-public-keys id_ed25519.pub
    #Démarrage des conteneurs
    pct start $pct
    j=$((j+1))
done

    #Execution du playbook ansible sur la machine rebond
    pct exec 104 -- sh -c "ansible-playbook -i
/home/rebond/ansible/inventory.yml /home/rebond/ansible/playbook.yml --key-
file /home/rebond/.ssh/id_ed25519"
```

Dans ce script, une seule infrastructure est possible. Il est néanmoins possible de récupérer l'ip du dernier conteneur et créer les conteneurs sur les IP suivantes qui seront libres :

```
#Récupération de la dernière IP dispo
j=$(pct exec 111 ip addr show | grep eth0 | grep inet | cut -d " " -f6 | cut
-d "/" -f1 | cut -d "." -f4)
```

Rôles Ansible

Lorsque l'on utilise ansible, il est possible de créer des rôles. Le rôle va s'organiser en arborescence de dossiers comprenant :

- Task : liste des actions d'installations et de déploiements
- Handlers : actions qui s'effectuent en attente d'une autre
- Templates: comprend les fichiers de configuration à envoyer
- Defaults et Var : comprend les variables que l'on veut utiliser dans les autres fichiers.

Inventaire

Je commence par lister les conteneurs à utiliser par catégorie

```
[all]
XX.XX.XX.XX
XX.XX.XX.XX
XX.XX.XX.XX
XX.XX.XX.XX
XX.XX.XX.XX

[all:vars]
ansible_user=root
ansible_become=yes
```

```
#Accepte automatiquement le fingerprint SSH
ansible_ssh_common_args='-o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null'


[proxy]
XX.XX.XX.XX


[web]
XX.XX.XX.XX
XX.XX.XX.XX


[web:vars]
ansible_user=root
ansible_become=yes

[bdd]
bdd1 ansible_host=XX.XX.XX.XX
bdd2 ansible_host=XX.XX.XX.XX
```

Rôle NGINX Proxy :

Task:

Dans ce fichier, le point important est la création du certificat permettant de chiffrer les connexions entrantes depuis le proxy.

```
---
# tasks file for nginx-reverse
- name: Installer NGINX
  apt:
    name: nginx
    state: present
    update_cache: yes
  notify: Redemarrer NGINX

# Création du fichier de certificat
- name: Creation du repertoire de certificats
  file:
    path: "/etc/ssl/private"
    state: directory
    mode: '0775'
```

```
# Création du fichier de certificat
- name: Creation du repertoire de certificats
  file:
    path: "/etc/ssl/certs"
    state: directory
    mode: '0755'

# Génération du certificat
- name: Generation du certificat
  command: openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/autosigne.key -out /etc/ssl/certs/autosigne.crt -subj
"/C=FR/ST=Paris/L=Paris/O=homelab/OU=IT/CN=localhost"

# Copie de la configuration
- name: Copier la configuration
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
  notify: Redemarrer NGINX

# Activation de NGINX
- name: Demarrer et activer NGINX
  service:
    name: nginx
    state: started
    enabled: yes
```

Templates:

```
events {
    worker_connections 1024;
}

http {
    upstream backend {
        server XX.XX.XX.XX:80;
        server XX.XX.XX.XX:80;
    }

    server {
        listen 80;
        server_name localhost;

        return 301 https://$host$request_uri;
    }
```

```
    server {
        listen 443 ssl;
        server_name localhost;

        # SSL configuration
        ssl_certificate /etc/ssl/certs/autosigne.crt;
        ssl_certificate_key /etc/ssl/private/autosigne.key;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384';

        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

Handlers :

```
# handlers file for nginx-reverse
- name: Redemarrer NGINX
  service:
    name: nginx
    state: restarted
```

Rôle NGINX :

Task :

```
# tasks file for nginx
---
- name: Installer NGINX
  apt:
    name: nginx
    state: present
    update_cache: yes
  notify: Redémarrer NGINX

- name: Copier la configuration NGINX depuis le template
```

```
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
  notify: Redémarrer NGINX

- name: Copier le fichier index.html
  template:
    src: index.html
    dest: /var/www/html/index.html

- name: Changement de permission du dossier web
  file:
    path: '/var/www/html'
    owner: www-data
    group: www-data
    mode: '0755'

- name: Démarrer et activer NGINX
  service:
    name: nginx
    state: started
    enabled: yes
```

Templates :

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 1024;
}

http {
   server {
    listen 80;
    server_name {{ nginx_server_name }};

    location / {
        root /var/www/html;
        index index.html index.htm;
    }
   }
}
```

Handlers :

```
# handlers file for nginx
- name: Redémarrer NGINX
  service:
    name: nginx
    state: restarted
```

Rôle base de données :

Task :

```
---
# tasks file for galera
- name: Mettre à jour le cache APT
  apt:
    update_cache: yes
    cache_valid_time: 3600

# Installation de pip et pymysql nécéssaire pour les paquets galera et mysql
- name: Installation de pip3
  apt:
    name:
      - python3-pip

- name: Installation de pymysql
  pip:
    name: pymysql
    executable: pip3
    extra_args: --break-system-packages

- name: Installation des paquets Mariadb et Galera
  apt:
    name:
      - rsync
      - mariadb-server
      - galera-4

- name: Définir le mot de passe root MySQL
  mysql_user:
    name: root
    host: localhost
    password: "changeme"
    priv: "*.*:ALL,GRANT"
```

```
  become: yes

- name: Configuration du serveur MariaDB pour Galera
  template:
    src: 'galera.cnf.j2'
    dest: '/etc/mysql/mariadb.conf.d/galera.cnf'
  notify: Redemarrer Galera

#Récupération des ip dans l'inventaire
- name: Démarrer MariaDB sur le premier nœud
  command: galera_new_cluster
  when: inventory_hostname == groups['bdd'][0]

- name: Démarrer MariaDB sur les autres nœuds
  service:
    name: mariadb
    state: started
    enabled: yes
  when: inventory_hostname != groups['bdd'][0]
```

Templates :

```
[mysqld]
# Activation de Galera
wsrep_on = ON
wsrep_provider = /usr/lib/galera/libgalera_smm.so
wsrep_cluster_name = "Galera_Cluster_IT-Connect"
wsrep_cluster_address = "gcomm://XX.XX.XX.XX,XX.XX.XX.XX"
wsrep_node_address = "{{ ansible_host }}"
wsrep_node_name = "{{ inventory_hostname }}"
wsrep_sst_method = rsync
wsrep_sst_auth = "root:changeme"

# Réplication et format des logs
binlog_format = ROW
default_storage_engine = InnoDB
innodb_autoinc_lock_mode = 2
innodb_force_primary_key = 1

# Autoriser les connexions sur toutes les interfaces
bind-address = 0.0.0.0

# Logs
log_error = /var/log/mysql/error-galera.log
```

Handlers :

```
---
# handlers file for galera

- name: Redémarrer MariaDB
  service:
    name: mariadb
    state: restarted
```

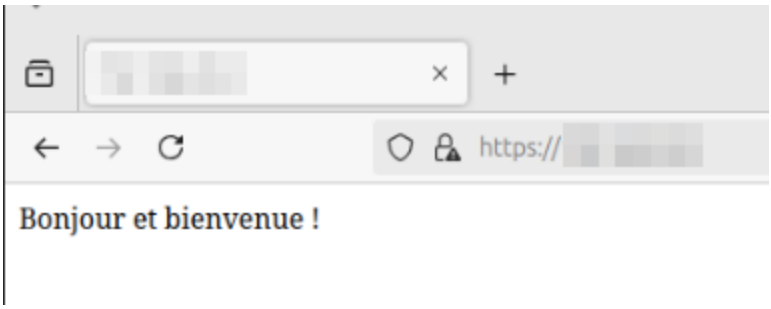
Playbook :

```
- name: Mise a jour des conteneur
  hosts: all
  gather_facts: no
  become: yes

  roles:
    - update

- name: Déploiement de nginx-reverse sur le conteneur
  hosts: proxy
  gather_facts: no
  become: yes

  roles:
    - nginx-reverse


- name: Déploiement de nginx sur les conteneurs
  hosts: web
  gather_facts: no
  become: yes

  roles:
    - nginx

- name: Déploiement de galera
  hosts: bdd
  gather_facts: yes
  become: yes

  roles:
    - galera
```

Résultat :

Web :

Nous avons bien une connexion chiffrée et redirigée vers les serveurs web:



Base de données

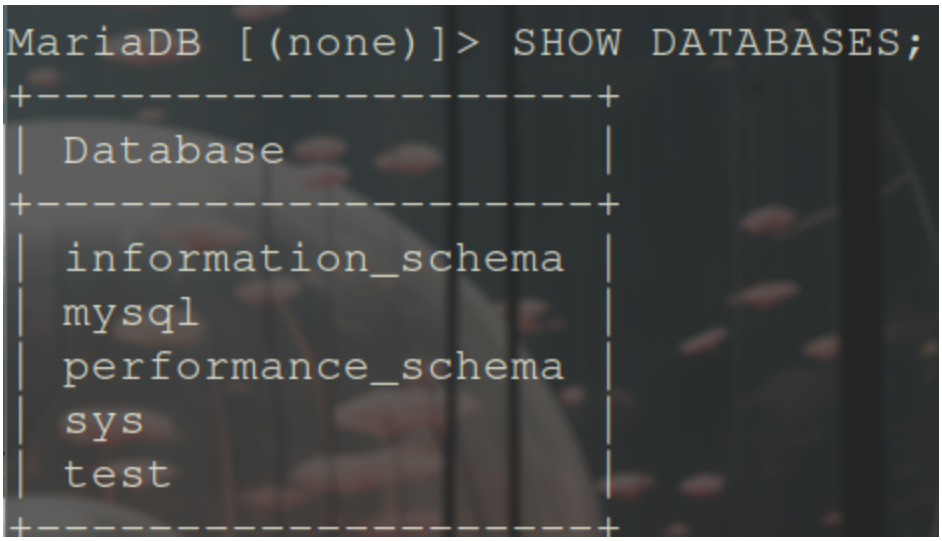
Notre cluster de bases de données réplique bien les données entre les deux nœuds du cluster.

```
root@slave-4:~# mysql -e "SHOW STATUS LIKE 'wsrep_cluster_size';"
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 2     |
+--------------------+-------+
```

Serveur 1

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.000 sec)
```

Serveur 2


MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database |
+--------------------+
| information_schema |
| mysql |
| performance_schema |
| sys |
| test |
+--------------------+