DOCUMENTATION TECHNIQUE

Infrastructure as Code

Terraform + Ansible + Proxmox

Projet: Déploiement automatisé de VMs

Auteur: Mamadou CAMARA

Date: Février 2026

SOMMAIRE :

1. Vue d’ensemble du projet

Ce projet met en œuvre une infrastructure automatisée utilisant Terraform pour le provisioning de machines virtuelles sur Proxmox VE, et Ansible pour la configuration des services applicatifs. L’objectif est de déployer rapidement et de manière reproductible des serveurs web avec Apache et des bases de données MariaDB.

1.1 Technologies utilisées

- Terraform v1.14.3 : Infrastructure as Code pour le provisioning
- Proxmox VE : Plateforme de virtualisation
- Ansible : Gestion de configuration et déploiement d’applications
- Ubuntu 22.04/24.04 : Système d’exploitation des VMs
- Apache2 : Serveur web
- MariaDB : Système de gestion de base de données

1.2 Objectifs du projet

- Automatiser le déploiement de 10 machines virtuelles
- Standardiser la configuration des serveurs
- Réduire le temps de provisioning de plusieurs heures à quelques minutes
- Assurer la reproductibilité de l’infrastructure
- Faciliter la scalabilité horizontale

2. Architecture de l'infrastructure

2.1 Schéma d'architecture

L'infrastructure est composée des éléments suivants :

- Serveur Proxmox VE (172.16.X.Y) : Hôte de virtualisation
- 10 VMs Ubuntu (VM-Mamadou-0 à VM-Mamadou-9)
- Template VM (ID 9001) : Image de base pour le clonage
- Réseau : Bridge vmbr0 (172.16.X.0/24)
- Stockage : local pour les disques des VMs

2.2 Spécifications des VMs

Composant	Valeur
CPU	2 cores
RAM	4098 MB (~4 GB)
Disque	36 GB
Réseau	virtio (Bridge vmbr0)
OS	Ubuntu (clone du template 9001)
IP	Assignée via DHCP (ex: 172.16.X.Z)

2.3 Flux de déploiement

Le processus de déploiement suit le workflow suivant :

1. `Terraform plan` → Analyse des ressources à créer
2. `Terraform apply` → Création des 10 VMs par clonage
3. Récupération des IPs des VMs créées
4. `ansible-playbook -i hosts.ini install_apache.yml` → Installation Apache
5. `ansible-playbook -i hosts.ini install_mariadb.yml` → Installation MariaDB
6. Vérification des services déployés

3. Configuration Terraform

3.1 Structure du projet

Le projet Terraform est organisé comme suit :

```
terraform/
├── main.tf              # Configuration principale
├── variables.tf         # Déclaration des variables
├── terraform.tfvars     # Valeurs des variables
├── .terraform.lock.hcl  # Verrouillage des providers
├── terraform.tfstate    # État de l'infrastructure
└── terraform.tfstate.backup  # Backup de l'état
```

3.2 Fichier main.tf

Le fichier **`main.tf`** définit les ressources à provisionner :

```
terraform {
  required_providers {
    proxmox = {
      source  = "bpg/proxmox"
      version = ">=0.50.0"
    }
  }
}

provider "proxmox" {
  endpoint  = var.pm_api_url
  api_token = "${var.pm_user}=${var.pm_password}"
  insecure  = true
}

resource "proxmox_virtual_environment_vm" "vm" {
  count = var.vm_count

  name      = "VM-Mamadou-${count.index}"
  node_name = var.node

  cpu {
```

```
    cores = var.vm_cores
  }

  memory {
    dedicated = var.vm_memory
  }

  disk {
    datastore_id = var.storage
    size         = var.vm_disk_size
    interface    = "scsi0"
  }

  network_device {
    model  = "virtio"
    bridge = var.bridge
  }

  clone {
    vm_id = 9001
  }
}
```

3.3 Variables Terraform

Variable	Description	Valeur
pm_api_url	URL de l'API Proxmox	https://172.16.X.Y:8006/api2/json
pm_user	Token API utilisateur	Token à récupérer sur Proxmox
pm_password	Secret du token API	Secret à récupérer lors de la création du token
node	Nœud Proxmox cible	pve
vm_count	Nombre de VMs à créer	10
vm_memory	RAM par VM (MB)	4098
vm_cores	CPU cores par VM	2
vm_disk_size	Taille disque (GB)	36

storage	Datastore Proxmox	local-lvm
bridge	Bridge réseau	vmbr0

4. Configuration Ansible

4.1 Fichier d'inventaire (hosts.ini)

Le fichier **`hosts.ini`** définit les hôtes cibles pour Ansible :

```
[webservers]
VM-Mamadou-0 ansible_host=IP-VM crée \
  ansible_user=mamadou \
  ansible_private_key_file=~/.ssh/id_ed25519
```

Note : Pour gérer les 10 VMs, ajoutez les autres machines (VM-Mamadou-1 à VM-Mamadou-9) avec leurs IPs respectives.

4.2 Playbook Apache (install_apache.yml)

Ce playbook installe et configure Apache2 :

```
---
- name: Installer Apache sur la VM Ubuntu
  hosts: webservers
  become: yes
  tasks:

    - name: Mettre à jour la liste des paquets
      apt:
        update_cache: yes

    - name: Installer Apache
      apt:
        name: apache2
        state: present

    - name: S'assurer que le service Apache est démarré
      service:
        name: apache2
        state: started
        enabled: yes

    - name: Créer une page HTML de test
```

```
    copy:
      dest: /var/www/html/index.html
      content: "<h1>Bienvenue sur Apache !</h1>"
      owner: www-data
      group: www-data
      mode: '0644'
```

4.3 Playbook MariaDB (install_mariadb.yml)

Ce playbook installe et sécurise MariaDB :

```
---
- name: Installer et configurer MariaDB
  hosts: webservers
  become: yes
  tasks:

    - name: Mettre à jour la liste des paquets
      apt:
        update_cache: yes

    - name: Installer MariaDB
      apt:
        name: mariadb-server
        state: present

    - name: Installer PyMySQL
      pip:
        name: PyMySQL
        executable: pip3

    - name: Configurer le mot de passe root
      mysql_user:
        name: root
        password: "VotreMotDePasse"
        login_unix_socket: /var/run/mysqld/mysqld.sock
```

5. Guide de déploiement

5.1 Prérequis

- Terraform installé (v1.14+)
- Ansible installé (v2.9+)
- Accès API Proxmox configuré
- Template VM (ID 9001) préparé dans Proxmox
- Clés SSH configurées (~/.ssh/id_ed25519)
- Connectivité réseau vers Proxmox et les VMs

5.2 Étape 1 : Préparation du template Proxmox

Créez un template Ubuntu dans Proxmox avec l'ID 9001 :

```
# Télécharger l'image Ubuntu Cloud
wget https://cloud-images.ubuntu.com/jammy/current/\
  jammy-server-cloudimg-amd64.img

# Créer la VM template
qm create 9001 --name ubuntu-template --memory 2048 \
  --cores 1 --net0 virtio,bridge=vmbr0

# Importer le disque
qm importdisk 9001 jammy-server-cloudimg-amd64.img \
  local-lvm

# Attacher le disque
qm set 9001 --scsihw virtio-scsi-pci \
  --scsi0 local-lvm:vm-9001-disk-0

# Configurer cloud-init
qm set 9001 --ide2 local-lvm:cloudinit
qm set 9001 --boot c --bootdisk scsi0
qm set 9001 --serial0 socket --vga serial0
qm set 9001 --ciuser mamadou
qm set 9001 --sshkeys ~/.ssh/id_ed25519.pub
```

```
# Convertir en template
qm template 9001
```

5.3 Étape 2 : Initialisation Terraform

```
# Se placer dans le dossier Terraform
cd terraform/

# Initialiser Terraform
terraform init
```

```
mamadou@CAMARA-M:~/Projet-Terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of bpg/proxmox from the dependency lock file
- Using previously-installed bpg/proxmox v0.93.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

```
# Vérifier la configuration
terraform validate

# Visualiser le plan d'exécution
terraform plan
```

```
mamadou@CAMARA-M:~/Projet-Terraform$ terraform plan
proxmox_virtual_environment_vm.vm[5]: Refreshing state... [id=110]
proxmox_virtual_environment_vm.vm[4]: Refreshing state... [id=105]
proxmox_virtual_environment_vm.vm[0]: Refreshing state... [id=101]
proxmox_virtual_environment_vm.vm[3]: Refreshing state... [id=104]
proxmox_virtual_environment_vm.vm[2]: Refreshing state... [id=111]
proxmox_virtual_environment_vm.vm[9]: Refreshing state... [id=112]
proxmox_virtual_environment_vm.vm[1]: Refreshing state... [id=108]
proxmox_virtual_environment_vm.vm[7]: Refreshing state... [id=107]
proxmox_virtual_environment_vm.vm[8]: Refreshing state... [id=109]
proxmox_virtual_environment_vm.vm[6]: Refreshing state... [id=106]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are
needed.
mamadou@CAMARA-M:~/Projet-Terraform$
```

5.4 Étape 3 : Déploiement des VMs

```
# Appliquer la configuration
terraform apply
```

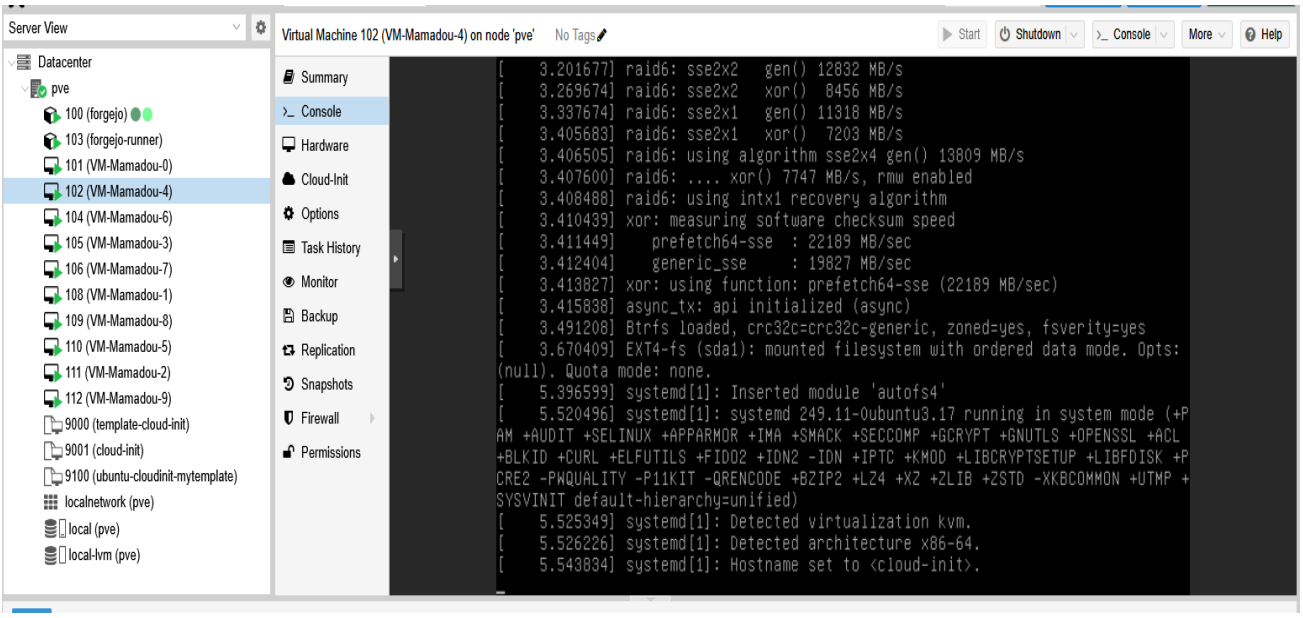
```
# Confirmer avec : yes

# Attendre la fin du provisioning (~5-10 minutes)
```

```
proxmox_virtual_environment_vm.vm[3]: Still creating... [01m36s elapsed]
proxmox_virtual_environment_vm.vm[3]: Still creating... [01m44s elapsed]
proxmox_virtual_environment_vm.vm[6]: Still creating... [01m44s elapsed]
proxmox_virtual_environment_vm.vm[7]: Still creating... [01m44s elapsed]
proxmox_virtual_environment_vm.vm[4]: Still creating... [01m44s elapsed]
proxmox_virtual_environment_vm.vm[4]: Still creating... [01m54s elapsed]
proxmox_virtual_environment_vm.vm[3]: Still creating... [01m54s elapsed]
proxmox_virtual_environment_vm.vm[6]: Still creating... [01m54s elapsed]
proxmox_virtual_environment_vm.vm[7]: Still creating... [01m54s elapsed]
proxmox_virtual_environment_vm.vm[7]: Still creating... [02m04s elapsed]
proxmox_virtual_environment_vm.vm[6]: Still creating... [02m04s elapsed]
proxmox_virtual_environment_vm.vm[4]: Still creating... [02m04s elapsed]
proxmox_virtual_environment_vm.vm[3]: Still creating... [02m04s elapsed]
proxmox_virtual_environment_vm.vm[6]: Creation complete after 2m13s [id=104]
proxmox_virtual_environment_vm.vm[7]: Creation complete after 2m13s [id=106]
proxmox_virtual_environment_vm.vm[3]: Still creating... [02m14s elapsed]
proxmox_virtual_environment_vm.vm[4]: Still creating... [02m14s elapsed]
proxmox_virtual_environment_vm.vm[3]: Creation complete after 2m16s [id=105]
proxmox_virtual_environment_vm.vm[4]: Creation complete after 2m14s [id=102]

Apply complete! Resources: 4 added, 1 changed, 0 destroyed.
mamadou@CAMARA-M:~/Projet-Terraform$
```

5.5 Étape 4 : Configuration Ansible

Récupérez les IPs des VMs créées et mettez à jour hosts.ini :

```
# Lister les VMs dans Proxmox
qm list | grep VM-Mamadou

# Ou vérifier dans l'interface web Proxmox
```




```
# Puis éditer hosts.ini
nano ansible/hosts.ini
```

5.6 Étape 5 : Exécution des playbooks Ansible

```
# Se placer dans le dossier Ansible
cd ansible/

# Tester la connectivité
ansible webservers -i hosts.ini -m ping

# Installer Apache sur toutes les VMs
ansible-playbook -i hosts.ini install_apache.yml
```

```
mamadou@CAMARA-M:~/Projet-Terraform$ ansible-playbook -i hosts.ini install_apache.yml

PLAY [Installer Apache sur la VM Ubuntu] ***************************************************************************

TASK [Gathering Facts] *********************************************************************************************
[WARNING]: Host 'VM-Mamadou-0' is using the discovered Python interpreter at '/usr/bin/python3.10', but future installation of another Python interpreter could cause a differ
ent interpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference_appendices/interpreter_discovery.html for more information.
ok: [VM-Mamadou-0]

TASK [Mettre à jour la liste des paquets] **************************************************************************
[WARNING]: Failed to update cache after 1 retries due to , retrying
[WARNING]: Sleeping for 1 seconds, before attempting to refresh the cache again
[WARNING]: Failed to update cache after 2 retries due to , retrying
[WARNING]: Sleeping for 2 seconds, before attempting to refresh the cache again
[WARNING]: Failed to update cache after 3 retries due to , retrying
[WARNING]: Sleeping for 4 seconds, before attempting to refresh the cache again
[WARNING]: Failed to update cache after 4 retries due to , retrying
[WARNING]: Sleeping for 8 seconds, before attempting to refresh the cache again
changed: [VM-Mamadou-0]

TASK [Installer Apache] ********************************************************************************************
ok: [VM-Mamadou-0]

TASK [S'assurer que le service Apache est démarré et activé] *******************************************************
ok: [VM-Mamadou-0]

TASK [Créer une page HTML de test] *********************************************************************************
ok: [VM-Mamadou-0]

PLAY RECAP *********************************************************************************************************
VM-Mamadou-0               : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

mamadou@CAMARA-M:~/Projet-Terraform$
```

```
# Installer MariaDB
ansible-playbook -i hosts.ini install_mariadb.yml
```

5.7 Étape 6 : Vérification

```
# Tester Apache
curl http://Ip-VM
# Devrait afficher : Bienvenue sur Apache !

# Vérifier MariaDB
ansible webservers -i hosts.ini -m shell \
  -a "systemctl status mariadb" -b
```

6. Commandes essentielles

6.1 Commandes Terraform

Commande	Description
terraform init	Initialise le projet et télécharge les providers
terraform plan	Affiche les changements à apporter
terraform apply	Applique les modifications
terraform destroy	Détruit toute l'infrastructure
terraform state list	Liste les ressources dans l'état
terraform show	Affiche l'état actuel
terraform validate	Valide la syntaxe des fichiers
terraform fmt	Formate les fichiers .tf
terraform output	Affiche les outputs définis
terraform refresh	Synchronise l'état avec la réalité

6.2 Commandes Ansible

Commande	Description
ansible all -m ping	Test de connectivité
ansible-playbook playbook.yml	Exécute un playbook
ansible-playbook -C playbook.yml	Mode dry-run (simulation)
ansible-playbook -v playbook.yml	Mode verbeux
ansible-playbook --check playbook.yml	Vérifie sans appliquer
ansible webservers -a "uptime"	Exécute une commande
ansible-inventory --list	Liste l'inventaire
ansible-doc apt	Documentation d'un module
ansible-playbook --tags apache	Exécute uniquement certains tags
ansible-vault encrypt file	Chiffre un fichier sensible

6.3 Commandes Proxmox (qm)

Commande	Description
qm list	Liste toutes les VMs
qm status <vmid>	Statut d’une VM
qm start <vmid>	Démarre une VM
qm stop <vmid>	Arrête une VM
qm shutdown <vmid>	Arrêt propre d’une VM
qm destroy <vmid>	Supprime une VM
qm clone <vmid> <newid>	Clone une VM
qm config <vmid>	Affiche la config d’une VM
qm terminal <vmid>	Console série de la VM
pvesh get /nodes/pve/qemu	API : liste VMs

7. Dépannage

7.1 Problèmes Terraform

Erreur : API connection failed

```
# Vérifier l'accès API
curl -k https://172.16.X.Y:8006/api2/json

# Vérifier les credentials dans terraform.tfvars
# Vérifier que le token API est valide dans Proxmox
```

Erreur : Template 9001 not found

```
# Vérifier l'existence du template
qm list | grep 9001

# Créer le template si nécessaire (voir section 5.2)
```

État Terraform corrompu

```
# Restaurer depuis le backup
cp terraform.tfstate.backup terraform.tfstate

# Ou synchroniser avec l'infrastructure réelle
terraform refresh
```

7.2 Problèmes Ansible

Erreur : SSH connection failed

```
# Vérifier la connectivité
ping IP VM

# Tester SSH manuellement
ssh -i ~/.ssh/id_ed25519 mamadou@ip-VM

# Vérifier les permissions de la clé
chmod 600 ~/.ssh/id_ed25519
```

Erreur : sudo password required

```
# Configurer sudo sans mot de passe
ssh mamadou@IP-VM
```

```
echo "mamadou ALL=(ALL) NOPASSWD:ALL" | \
  sudo tee /etc/sudoers.d/mamadou
```

Erreur : PyMySQL module not found

```
# Le playbook MariaDB installe automatiquement PyMySQL
# Si erreur persiste, installer manuellement :
ansible webservers -i hosts.ini -m apt \
  -a "name=python3-pip state=present" -b
ansible webservers -i hosts.ini -m pip \
  -a "name=PyMySQL" -b
```

7.3 Problèmes réseau

VM sans adresse IP

```
# Vérifier le service DHCP
# Vérifier la configuration cloud-init du template
# Redémarrer la VM
qm shutdown <vmid> && qm start <vmid>
```

Conflit d'IP

```
# Vérifier les IPs utilisées
nmap -sn 172.16.X.0/24

# Configurer des IPs statiques dans cloud-init si nécessaire
```

Annexes

A. Arborescence complète du projet

```
projet/
├── terraform/
│   ├── main.tf
│   ├── variables.tf
│   ├── terraform.tfvars
│   ├── .terraform.lock.hcl
│   ├── terraform.tfstate
│   └── terraform.tfstate.backup
├── ansible/
│   ├── hosts.ini
│   ├── install_apache.yml
│   └── install_mariadb.yml
└── README.md
```

B. Checklist de déploiement

- ☐ Template Proxmox créé (ID 9001)
- ☐ Clés SSH générées et configurées
- ☐ Token API Proxmox créé et testé
- ☐ Terraform initialisé (terraform init)
- ☐ Variables configurées dans terraform.tfvars
- ☐ Plan Terraform vérifié (terraform plan)
- ☐ Infrastructure déployée (terraform apply)
- ☐ IPs des VMs récupérées
- ☐ Fichier hosts.ini mis à jour
- ☐ Connectivité Ansible testée (ping)
- ☐ Playbook Apache exécuté
- ☐ Playbook MariaDB exécuté
- ☐ Services vérifiés et fonctionnels

C. Ressources et liens utiles

- Documentation Terraform : https://terraform.io/docs
- Provider Proxmox : https://registry.terraform.io/providers/bpg/proxmox
- Documentation Ansible : https://docs.ansible.com
- Proxmox VE : https://pve.proxmox.com/wiki/Main_Page
- Ubuntu Cloud Images : https://cloud-images.ubuntu.com

D. Bonnes pratiques

- Toujours versionner le code Terraform et Ansible (Git)
- Ne jamais committer les fichiers .tfstate ou terraform.tfvars
- Utiliser Ansible Vault pour les secrets (mots de passe MariaDB)
- Tester les playbooks en mode --check avant application
- Documenter les changements d’infrastructure
- Maintenir des backups réguliers du tfstate
- Utiliser des tags Git pour les versions stables
- Automatiser les déploiements avec CI/CD si possible

Conclusion

Ce projet démontre la puissance de l'Infrastructure as Code en combinant Terraform pour le provisioning et Ansible pour la configuration. L'automatisation complète permet de déployer 10 serveurs web avec Apache et MariaDB en quelques minutes, de manière reproductible et documentée.

Les avantages de cette approche incluent :

- Rapidité de déploiement : passage de plusieurs heures à ~15 minutes
- Reproductibilité : infrastructure identique à chaque déploiement
- Scalabilité : ajout de VMs en modifiant simplement vm_count
- Documentation vivante : le code est la documentation
- Versioning : suivi des changements via Git
- Récupération rapide : reconstruction complète en cas de problème

Ce document constitue une base solide pour étendre l'infrastructure avec d'autres services (nginx, PostgreSQL, Docker, Kubernetes) ou pour migrer vers un environnement cloud (AWS, Azure, GCP) en adaptant simplement les providers Terraform.

Fin de la documentation technique