Infrastructure as Code (IaC) avec Terraform

Olivier Martin

Janvier 2023

Premier TD de virtualisation pour le module R4.08, BUT Informatique deuxième année.

Travail en binôme en salle de machines.

Compétence travaillée : Installer, configurer, mettre à disposition, maintenir en conditions opérationnelles des infrastructures, des services et des réseaux et optimiser le système informatique d'une organisation – niveau 2 : Déployer des services dans une architecture réseau – Utiliser des serveurs et des services réseaux virtualisés.

Pré-requis :
- Utilisation de la ligne de commande Unix (shell Bash)
- Notions de virtualisation
- Notions sur les réseaux TCP/IP

Matériel ou logiciels utilisés :
- Accès au serveur Proxmox de l'Université depuis un navigateur web.
- Client SSH, par exemple PuTTY ou MobaXterm (optionnel)
- Terraform, Ansible (sur le poste principal ou sur une VM)

Table des matières

1 Terraform, un outil pour l'IaC

Terraform est un outil de IaC, *Infrastructure as Code*. Il permet de décrire une infrastructure virtuelle à partir de fichiers de configuration. Il est principalement dédié à la création d'infrastructures sur le cloud, mais il existe aussi des *plug-ins* pour Proxmox, et c'est ce que nous allons utiliser.

Dans ce TD, nous allons donc déployer plusieurs VM avec Terraform à partir du *template* créé précédemment, puis nous installerons un service sur ces VM. Ce sera encore Wordpress puisque nous avons déjà les fichiers Ansible. Tout ceci se fera avec un minimum d'intervention une fois les fichiers de configuration écrits.

La comparaison entre Vagrant et Terraform est inévitable :
- tous deux ont le même éditeur
- tous deux permettent de configurer des machines virtuelles
- tous deux utilisent un fichier de configuration déclaratif, c'est à dire qui exprime le résultat attendu et pas les étapes pour y arriver (ce n'est pas l'utilisateur mais l'outil qui décide des étapes pour arriver au résultat)
- tous deux utilisent des *plug-ins* pour se connecter à des fournisseurs de virtualisation et offrent ainsi une certaine portabilité des environnements virtuels créés.

Les principales différences sont que Vagrant concerne plutôt des installations sur le poste local, et qu'il utilise des "boxes" ; quant à Terraform il déploie des VM sur des serveurs distants, et surtout il conserve une trace de l'état de l'infrastructure virtuelle qu'il a déployée, ce qui évite de tout relancer quand un paramètre change.

On utilise Terraform de la façon suivante :
- Création d'un fichier de configuration nommé `main.tf`
- Initialisation de l'environnement et téléchargement des *plug-ins* avec `terraform init` ; les données et *plug-ins* sont enregistrées dans un répertoire local nommé `.terraform`
- Appel de `terraform plan` : l'outil compare l'état actuel de l'infrastructure avec l'état désiré et prévoit un plan d'action. Pour cela il peut contacter le fournisseur du service ou utiliser des informations sauvegardées localement. Lors du premier appel, tout est à créer.
- Appel de `terraform apply` : l'outil exécute le plan d'action et sauvegarde le nouvel état de l'infrastructure dans un fichier local.
- À chaque changement du fichier de définition de l'infrastructure il faut appeler à nouveau `terraform plan` puis `terraform apply`. L'appel à `terraform plan` affiche une liste des modifications qui vont être faites.
- L'infrastructure virtualisée et le fichier d'état sont supprimés avec `terraform destroy`.

L'ordre des appels à la commande terraform et leurs interactions avec les fichiers de configuration ou de sauvegarde, avec le repo de plug-ins et avec le serveur de virtualisation sont donnés à la figure 1.

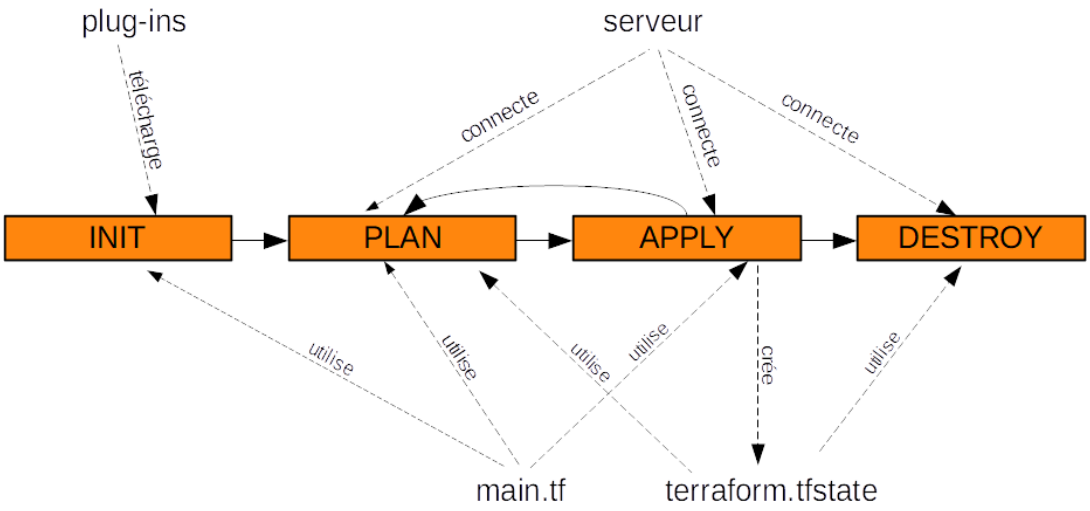

FIGURE 1 – Ordre des appels à la commande `terraform`.

2 Déroulement du TD, en fonction du proxy

Vous allez configurer Terraform sur votre poste de travail. À l'heure où ce document est écrit il y a deux possibilités :

- Cas 1 : on peut se connecter à Proxmox depuis les VM qu'il héberge. Dans ce cas vous installerez Terraform sur une VM et tout le TP se fera entièrement depuis cette VM. Un *template* de VM avec Ubuntu 22.04 Desktop est fourni sur Proxmox.
- Cas 2 : les VM ne peuvent pas se connecter à Proxmox. Dans ce cas vous téléchargerez Terraform et vous l'exécuterez depuis vos postes Windows (c'est juste un fichier exécutable à télécharger, il n'y a rien à installer). Par contre les postes Windows ne peuvent pas communiquer avec les VM, il faudra donc appeler Ansible depuis une VM (comme pour le cas 1, vous pouvez utiliser le *template* fourni)

3 Ressources

Pour ce TD vous disposez des ressources suivantes :

Le patron que vous avez créé à la séance précédente ; c'est lui qui va servir à instancier les machines avec Terraform. Si vous n'avez pas pu faire ce patron (*template*), utilisez `r408-template-00` (login et mot de passe `etudiant`/`etudiant`)

Le patron pour la machine *desktop*, tel que décrit à la section 4

Un fichier de paramètres pour Terraform, qui sera décrit à la section 5

Les fichiers Ansible pour installer Wordpress sur les VM gérées par Terraform. Reprenez les fichiers du TP1, disponibles à l'adresse `https://infoiut.jfanne.fr/IMG/zip/tp1.zip`.

4 Création de la VM desktop

Un *template* avec le système Ubuntu 22.04 Desktop (disposant donc de l'interface graphique) est disponible sur le serveur Proxmox. Il se nomme `r408-desktop-template-00`. L'identifiant et mot de passe de l'utilisateur sont : `etudiant/etudiant`.

Après avoir cloné le *template* ("Full clone"), pensez à renommer la machine comme indiqué dans le document "template.pdf" du TP1. Il est fortement suggéré de changer le mot de passe de l'utilisateur pour éviter qu'un autre groupe se connecte à votre VM, par erreur ou par malice.

Il faudra aussi ajouter vos identifiant et mot de passe Campus 3 aux paramètres de proxy à plusieurs endroits :

- dans le fichier `/etc/apt/apt.conf.d/10proxy` pour l'appel aux commandes `apt`
- dans `/etc/environment` pour les diverses commandes shell qui utilisent les variables `http_proxy` et `https_proxy`
- dans l'interface de Ubuntu (engrenages en haut à droite du bureau, puis recherchez "proxy")
- dans l'interface de Firefox.

Profitez-en pour installer Ansible avec
`sudo apt update && sudo apt install ansible`

5 Le fichier main.tf de Terraform

Installez Terraform sur le système de votre choix, en tenant compte des remarques de la section 2.

Le fichier `main.tf` est donné en annexe, et dans les fichiers du TD. Il se décompose en quatre blocs :

- Un bloc `terraform` qui indique les fournisseurs utilisés dans le fichier.
- Un bloc `locals` qui contient des variables locales, on a regroupé la plupart des options que vous devrez modifier pour utiliser le fichier.
- Un bloc `provider` pour le fournisseur "proxmox" : ce bloc contient les options du fournisseur "Telmate/proxmox", en particulier l'adresse de connexion au serveur. Notez qu'il s'agit de l'adresse de connexion à l'API web de pilotage du serveur et pas à son interface graphique.
- Un bloc de définition d'une ressource. Dans notre exemple on définit une ressource du type "proxmox_vm_qemu" qui se nomme "tp_servers". Ce type de ressource est fourni par le *plug-in* "Telmate/proxmox", et tous les paramètres des VM sont spécifiques au *plug-in*. La portabilité de Terraforme est en fait assez limitée.

Dans la variable locale "proxmox_api" vous indiquerez l'adresse de l'API web de pilotage du serveur Proxmox. Cette variable est utilisée dans le bloc `provider`

Dans la variable "template_name" vous indiquerez le nom de votre *template* pour Ubuntu Server, celui que vous avez créé la semaine précédente. La variable "vm_memory" indique la mémoire allouée à la VM. Ces deux paramètres sont utilisés dans le bloc `resource`.

Vous devrez aussi modifier la valeur du champ "name" du bloc `resource`. Dans ce bloc on demande de créer deux VM, indiqué par le champ "count". Les VM créées sont numérotées et leur nom de machine contient leur numéro. Intégrez votre numéro de groupe dans le nom de VM, en conservant la partie `$count.index + 1` qui les numérote 1 et 2.

Il faut fournir à l'outil vos identifiant et mot de passe pour se connecter au serveur. On ne met pas les identifiants dans le fichier pour des questions de sécurité. Vous allez les indiquer à travers des variables d'environnement, comme à la figure 2. Attention, sous Windows remplacez 'export' par 'env' et enlevez les guillemets autour des chaînes de caractères.

```
terraform_proxmox$ export PM_USER="[illegible]@IUTC3"
terraform_proxmox$ export PM_PASS="[illegible]"
terraform_proxmox$ terraform plan
```

FIGURE 2 – Utilisation de variables d'environnement pour les paramètres de connexion à Proxmox

Appelez `terraform plan` et vérifiez que la connexion au serveur fonctionne et que les ressources à créer correspondent à ce que vous attendez. Il y a beaucoup plus de paramètres que ceux que vous avez fournis, Terraform et le *plug-in* "Telmate/proxmox" utilisent des valeurs par défaut pour toutes les valeurs non fournies.

Appelez `terraform init` pour instancier les variables, tapez 'yes' à l'invite et attendez. Les VM doivent être créées et démarrées par l'outil. Vous pouvez suivre l'avancement dans l'interface de Proxmox et dans le terminal.

Nous allons maintenant modifier les paramètres des variables : 1024 Mo de RAM suffisent pour l'utilisation que nous allons en faire. Modifiez le fichier main.tf, relancez `terraform plan`, vérifiez que la mémoire des deux machines va être changée, appelez `terraform apply` et attendez que les VM redémarrent avec la nouvelle configuration. Vous pouvez vérifier la mémoire disponible sur les VM avec la commande `free -h`.

6 Provisionnement des VM

Le provisionnement des VM n'est pas intégré à Terraform comme il l'était avec Vagrant, nous allons donc appeler Ansible par nous-mêmes. Pour cela il faut que la machine qui exécute Ansible puisse se connecter aux VM à provisionner, nous allons donc commencer par créer des clés de cryptage et les utiliser avec SSH.

Vous avez besoin de l'adresse des VM que vous avez créées, connectez-vous y pour récupérer leurs adresses IP. C'est la seule étape manuelle de cette installation de système. Il y a des mécanismes pour récupérer les adresses IP des VM instanciées par Terraform, si vous disposez d'un peu de temps à la fin de ce TD vous pourrez essayer de le faire à titre d'exercice (!)

6.1 Préparation à la connexion

Commencez par vérifier que la machine sur laquelle vous avez installé Ansible peut se connecter aux VM :
`ssh <user>@<ip-vm>`

Vous allez ensuite créer une paire de clés RSA avec la commande `ssh-keygen`, tapez sur "entrée" à chaque invite. Les clés sont enregistrées dans le répertoire `/.ssh` sous les noms de `id_rsa` et `id_rsa.pub`. La première est une clé privée qui ne doit pas être divulguée, la deuxième est une clé publique que vous pouvez diffuser, et c'est cette dernière qui va être communiquée aux VM.

Copiez ensuite la clé public sur chacune des VM avec la commande
`ssh-copy-id <user>@<ip-vm>`

Vérifiez ensuite que la connexion fonctionne avec la commande
`ssh -i ~/.ssh/id\_rsa <user>@<ip-vm>`

Il est désormais possible de se connecter aux VM depuis votre machine grâce aux clés RSA. Ansible va utiliser cette fonctionnalité et vous n'aurez pas besoin d'indiquer le mot de passe dans les fichiers de configuration.

Puisque Ansible ne connaît pas le mot de passe, il faut aussi qu'il puisse devenir root sur les VM sans avoir à fournir un mot de passe. Pour cela, connectez-vous à chaque VM et tapez `sudo visudo`. Vous allez maintenant éditer le fichier de configuration de la commande Sudo : Retrouvez la ligne qui correspond à votre utilisateur ou au groupe 'sudo' dans le fichier et modifier la fin de ligne pour qu'elle indique `NOPASSWD:ALL`
Sauvegardez le fichier, quittez l'éditeur, déconnectez-vous et reconnectez-vous à la machine puis testez les modifications avec `sudo ls`. Cette commande ne doit pas vous demander de mot de passe.

6.2 Adaptation des fichiers playbook

Il y a quelques changements à faire aux fichiers playbook :

- Dans `playbook_sql.yaml`, le package "python-pymysql" n'existe plus sous Ubuntu 22.04, il a été remplacé par "python3-pymysql". Modifiez le fichier en conséquence.
- Dans `envvar.yaml`, il faut mettre à jour l'adresse IP de la base de données et il faut indiquer des identifiants corrects à la rubrique "proxy".
- Dans `playbook_web.yaml`, pour la tâche "Fichier de configuration pour Apache" il faut indiquer un chemin correct pour le fichier `wordpress.conf` (depuis le répertoire où vous exécutez "ansible-playbook")
- Dans `playbook_web.yaml`, pour la tâche "Création du fichier de configuration de Wordpress" on copie de la VM vers la VM et on doit ajouter le paramètre
  `remote_src: yes`
- Dans `playbook_web.yaml` encore, la tâche "Ajouter les lignes key/salt au fichier de config" a besoin des informations de proxy ; ajoutez un paramètre
  `environment: " proxy_env "`
  à l'ensemble de la tâche (même niveau d'indentation que 'name')

6.3 Provisionnement des VM avec Ansible

À cette étape, vous disposez des fichiers playbook, de l'utilitaire Ansible, de deux VM, d'un moyen de se connecter à chaque VM sans entrer de mot de passe, d'un compte sur chaque machine qui peut utiliser 'sudo' sans mot de passe. Il ne vous reste plus qu'à lancer Ansible pour exécuter toutes les tâches d'installation décrites dans les playbooks :

```
ansible-playbook ./playbook\_web -i <ip-vm-1>,
```
et
```
ansible-playbook ./playbook\_sql -i <ip-vm-2>,
```

Attention à la virgule après l'adresse des VM, ce n'est pas une faute de frappe mais un moyen d'indiquer à Ansible qu'après l'option '-i' on lui passe une liste de machines et pas un nom de fichier contenant une liste de machines.

Si tout se passe bien ouvrez l'adresse `http://<ip-vm-1>/` dans un navigateur web, la page de configuration de Wordpress doit apparaître.

7 Pour aller plus loin

Ce déploiement avec Terraform n'est pas aussi automatisé ni aussi sécurisé que celui que nous avons fait avec Vagrant la semaine passée. Vous pouvez si vous le souhaitez réfléchir et proposer des solutions aux points suivants :

- Comment récupérer automatiquement les IP des VM (une suggestion : utiliser Avahi/Bonjour pour accéder aux machines en utilisant leur nom, et récupérer ainsi leur IP)
- Comment créer un réseau privé entre les VM et utiliser ce réseau pour communiquer entre Apache et MySQL.

(Si l'administration de l'IUT le permet, les premières solutions et les solutions les plus élégantes pourront se voir attribuer des points bonus sur la note du module.)

8 Conclusion

Dans ce TD nous avons utilisé un outil d'*Infrastructure as Code* pour déployer une série de VM, et pour les mettre à jour. Nous avons ensuite utilisé un outil de provisionnement pour installer un service sur les VM précédemment créées.

Les techniques vues aujourd'hui servent fréquemment dans l'industrie, nous n'avons fait que survoler les possibilité qu'elles offrent.

Ce document est mis à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Annexes

A Fichier main.tf

```
terraform {
    required_providers {
        proxmox = {
        source = "Telmate/proxmox"
        version = "2.9.10"
        }
    }
}

# on regroupe les paramètres àadapter ici
locals {
  # point d'accès de l'API Proxmox
  proxmox_api = "https://proxmox-iutc3.unicaen.fr/api2/json"
  # proxmox_api = "https://192.168.1.101:8006/api2/json"

  proxmox_node = "pve1"

  # nom du template àcloner pour créer les VM
  template_name = "r408-template"
  # nom de la VM ou des VMs àmodifier dans le bloc "resource"...

  #paramètres de la VM
  vm_memory = 2048
}

provider "proxmox" {
    pm_api_url = local.proxmox_api
        pm_tls_insecure = true
}


resource "proxmox_vm_qemu" "tp_servers" {
  desc = "Deploiement de VMs Ubuntu sur Proxmox"
  count = 2
  name = "r408-vm${count.index + 1}-om"
  target_node = local.proxmox_node
  clone = local.template_name
  full_clone = true
  define_connection_info=true
  pool="INFO"

  cores = 1
  sockets = 1
```

```
  memory = local.vm_memory
  cpu = "host"
  scsihw = "virtio-scsi-pci"
  bootdisk = "scsi0"

  disk {
    slot = 0
    size = "16G"
    type = "scsi"
    storage = "info"
  }

  network {
    model = "virtio"
    bridge = "vmbr0"
  }
# network {
# model = "virtio"
# bridge = "vmbr10"
# }
}
```