Rapport de projet : Automatisation d'une infrastructure Big Data

Introduction et Avancement

Infrastructure Terraform

Terminé

- L'infrastructure générée avec Terraform et libvirt/KVM se déploie sur une seule machine physique. (Description ici)
- 4 VMs, IPs statiques
- Réseau NAT, avec DHCP et DNS pour avoir des noms de vm plutot que des IPs
- Disque virtuel : ubuntu cloud image
- Initialistation des VMs avec cloudinit

Non Terminé

- Pas de solution de lien entre 2 réseau de VMs sur 2 machines physique différentes

Déploiment Ansbile

Terminé

- Spark peut être déployé sur les 4 VMs en suivant les consignes ici
- Le namenode HDFS / master Spark est la VM0, les autres sont les slaves

Structure et rôles des fichiers

Répertoire Terraform

Le répertoire `terraform/` contient les fichiers nécessaires à Terraform :

- `terraform.tf` : Spécification du provider terraform (Libvirt ici)
- `main.tf` : Fichier principal de configuration des machines virtuelles et réseau virtuel
- `cloud_init.cfg` : Fichier cloudinit, permet de configurer chaque VM au moment de sa création
- `network_config_dhcp.cfg` et `network_config_static.cfg` : Fichier cloudinit network pour spécifier les paramètres et cartes réseau des VMs
- `sshkeys/` : Clés ssh correspondantes à chaque machines virtuelles
- `rmsshkeys.sh` : Script pour supprimer les clés SSH connues correspondantes aux VMs pour éviter les conflits à la création / destruction des VMs

Répertoire Ansible

Le répertoire `ansible/` contient les fichiers pour automatiser l'installation et la configuration des services :

- `ansible.cfg` : Spécifie l'user pour se connecter par ssh
- `inventory.ini` : Définit les machines cibles via leurs IPs
- `ssh.yml` : Playbook qui envoie les clés ssh aux VMs correspondantes
- `spark.yml` : Playbook principal pour installer Spark et ses dépendances
- `submit.yml` : Soumission des tâches Spark via `submit.sh`
- `files/` : Contient les fichiers à envoyer via les playbooks :
- `submit.sh` : script qui **lance** les *namenode*, *master*, *datanode* et *workers* HDFS / Spark lance le **spark-sbumit** puis **termine** les processus
- `core-site.xml`, `hdfs-site.xml`, `hadoop-env.xml` : Configuration de

l'environnement HDFS

- `spark-env.sh` : Configuration de l'environnement Spark
- `hadoop_slaves` : Définit les esclaves HDFS / Spark
- `tp2/tpSpark/` : Dossier contenant l'application **WordCount**

Déploiment de l'infrastructure

Phase 1 : Terraform

Important : Vous aurez peut-être besoin de télecharger l'image ubuntu ici. A placer dans le répertoire `terraform/`

Supprimer le réseau libvirt par défaut

Lancer la commande

```
$ virsh net-destroy default
```

Création de l'infrastructure

1. Se placer dans le répertoire `terraform/` .
2. Lancer les commandes

```
$ terraform plan
$ terraform apply
```

Vérification de l'infrastructure

```
$ virsh list
```

Phase 2 : Déploiement Ansible

A savoir / Important

- Se placer dans répertoire `ansible/`
- Les VMs redémarrent après 30s depuis leur premier lancement, il se peut que les VMs redémarrent pendant l'éxecution d'un playbook. Pour vérifier que toutes les VMs sont opérationnelles :

```
$ ansible all -i inventory.ini -m ping
```

Lancement des playbooks

Commande générale :

```
$ ansible-playbook -i inventory.ini file
```

A envoyer dans l'ordre:

1. Playbook `ssh.yml`
2. Playbook `spark.yml`
3. Playbook `submit.yml`

La description des fichiers est faite ici

Phase 3 : Validation

Le répertoire `~/spark/tp2/tpSpark/resultwc` est le dossier contenant le résultat du WordCount avec Spark.

Vous ne verrez **AUCUN** worker, datanode, master, namenode en fonctionnement après un `jps` car le script `submit.sh` lancé dans le playbook `submit.yml` lance les processus, lance le spark-submit puis éteint les processus. (Voir `submit.sh`)

Phase 4 : Extension multi-physique (non réalisée)

Cette phase, initialement prévue, vise à étendre l'infrastructure au-delà de l'hôte local KVM. Elle inclurait :

- La répartition des nœuds sur plusieurs serveurs physiques.
- La configuration d'un réseau VxLAN.
- Une gestion distribuée des données et des tâches, adaptée à des environnements multi-serveurs.