



ÉCOLE NATIONALE SUPÉRIEURE D'ÉLECTROTECHNIQUE, D'ÉLECTRONIQUE,
D'INFORMATIQUE, D'HYDRAULIQUE ET DES TÉLÉCOMMUNICATIONS
ENSEEIH

3SN IBDIOT

RAPPORT DE PROJET

Automatisation du Déploiement de Spark avec Ansible et Terraform

Membres du groupe :

Salma EL KHATRI

Sifeddine DEKKAKI

Encadrant:

DJOMGWE TEABE Boris

Année académique 2024/2025

Contents

1	Étape 1: Construction de l'infrastructure avec Terraform	2
1.1	Objectif	2
1.2	Démarche	2
1.3	Configuration réseau	3
2	Étape 2: Automatisation du déploiement de Spark avec Ansible	4
2.1	Objectif	4
2.2	Démarche	4
3	Étape 3: Validation du déploiement avec l'application WordCount	5
3.1	Objectif	5
3.2	Démarche	5

Introduction

Ce projet vise à automatiser le déploiement d'une infrastructure Big Data basée sur Apache Spark. L'objectif est d'utiliser Terraform pour la création de l'infrastructure et Ansible pour le déploiement et la configuration de Spark. Une application de test, WordCount, sera utilisée pour valider le bon fonctionnement du système.

Objectifs

- Automatiser le déploiement d'une infrastructure Big Data basée sur Apache Spark.
- Utiliser Terraform pour créer une infrastructure composée de machines virtuelles sur plusieurs hôtes physiques.
- Configurer et déployer Apache Spark sur ces machines en utilisant Ansible.
- Valider l'installation via l'exécution d'une application de test, WordCount.

1 Étape 1: Construction de l'infrastructure avec Terraform

1.1 Objectif

Créer et configurer quatre machines virtuelles (VMs) sur deux machines physiques en utilisant Terraform et KVM avec libvirt.

1.2 Démarche

1. Préparation des machines physiques :

- Installation de KVM, libvirt, et Terraform sur les deux machines hôtes.
- Configuration d'un pont réseau pour permettre la connectivité entre les VMs.

2. Configuration Terraform :

Création du fichier main.tf : pour définir les ressources (VMs, réseaux).

- Définition du fournisseur Terraform :** Terraform est configuré pour utiliser le provider libvirt, qui permet de gérer les machines virtuelles sous KVM. Le champ uri définit l'accès à l'hyperviseur local via qemu:///system.
- Définition des variables :** On définit le nombre de machines virtuelles à créer et on leur assigne des adresses IP statiques aux VMs pour garantir une communication réseau stable.
- Création des volumes de disques pour les VMs :** Terraform utilise une image Ubuntu Cloud (.img) comme base pour les VMs, la technologie qcow2 permet de créer des disques différenciés pour économiser de l'espace disque.
- Configuration du cloud-init pour l'initialisation des VMs :** Cloud-init est utilisé pour automatiser la configuration initiale des machines virtuelles. Chaque VM reçoit un hostname et un FQDN, un utilisateur ubuntu avec accès sudo, et une clé SSH pour l'authentification sans mot de passe.

L'accès SSH par mot de passe est activé (`ssh_pwauth: true`), et un mot de passe par défaut (`ubuntu:ubuntu`) est défini.

- (e) **Déploiement des machines virtuelles** : Chaque VM dispose de 2 Go de RAM et 2 vCPUs. Le disque cloud-init est attaché pour initialiser la VM.

3. Déploiement :

Exécution des commandes Terraform

- `terraform init`
- `terraform apply`

```
libvirt_volume.vm_volume[1]: Creating...
libvirt_volume.vm_volume[0]: Creating...
libvirt_cloudinit_disk.vm_cloudinit_disk[0]: Creation complete after 2s [id=/var/
/lib/libvirt/images/vm-0-cloudinit.iso;c644fa1a-022e-4d3e-98b5-3ce32ad07102]
libvirt_cloudinit_disk.vm_cloudinit_disk[1]: Creation complete after 2s [id=/var/
/lib/libvirt/images/vm-1-cloudinit.iso;237f435a-860d-4252-8a46-9d3d423fcf40]
libvirt_volume.vm_volume[1]: Creation complete after 0s [id=/var/lib/libvirt/ima
ges/vm1_disk]
libvirt_volume.vm_volume[0]: Creation complete after 0s [id=/var/lib/libvirt/ima
ges/vm0_disk]
libvirt_domain.vm[1]: Creating...
libvirt_domain.vm[0]: Creating...
libvirt_domain.vm[1]: Creation complete after 2s [id=a510e116-9511-4c3a-8b9b-d3c
42ce5d2d7]
libvirt_domain.vm[0]: Creation complete after 2s [id=58c3dd61-f745-4bbd-9281-237
bdcf8d719]
Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
```

Figure 1: Déploiement du terraform avec succès

Résultat : Quatre VMs créées avec succès, réparties sur deux machines physiques.

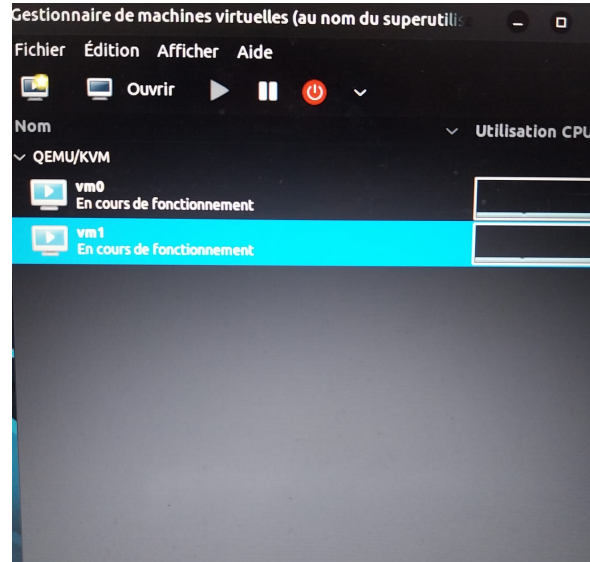


Figure 2: virt-manager pour la première machine physique

1.3 Configuration réseau

Chaque machine virtuelle est configurée avec une adresse IP statique assignée via Terraform. Le VPN Hamachi est utilisé pour établir une communication sécurisée entre les machines hôtes et les VMs.

1. Installation du VPN Hamachi:

```
wget https://vpn.net/installers/logmein-hamachi_2.1.0.203-1_amd64.deb
sudo dpkg -i logmein-hamachi_2.1.0.203-1_amd64.deb
sudo apt-get install -f # Pour résoudre Les dépendances
```

2. Démarrer et connecter Hamachi pour chaque VM :

```
sudo systemctl start logmein-hamachi
sudo hamachi login
sudo hamachi join selsa-infra infra
```

2 Étape 2: Automatisation du déploiement de Spark avec Ansible

2.1 Objectif

Installer et configurer Apache Spark automatiquement sur les VMs créées.

2.2 Démarche

1. Configuration d'Ansible

- Création d'un fichier inventory listant les adresses IP des VMs.
- Écriture d'un playbook Ansible pour :
 - Installer les dépendances Java(version 17).
 - Télécharger et déployer Apache Spark à partir du site officiel, le décompresse, et crée un lien symbolique pour faciliter l'accès à Spark.
 - Configurer les fichiers essentiels de Spark en modifiant les fichiers de configuration de Spark (notamment spark-env.sh) pour définir les paramètres nécessaires, comme l'adresse du master Spark et le nombre de cœurs par worker.

2. Exécution :

- Commande pour exécuter les playbooks :

```
ansible-playbook -i inventory.ini spark_install.yml
```

Cette commande permet à Ansible de se connecter aux VMs listées dans le fichier inventory.ini et d'exécuter les tâches définies dans le playbook spark_install.yml.

```
Fichier Edition Affichage Recherche Terminal Aide
root@n7student:~/terraform_project# ansible-playbook -i inventory.ini spark_install.yml

PLAY [Set up Spark cluster] *****

TASK [Gathering Facts] *****
ok: [vn3]
ok: [vn2]
ok: [vn1]
ok: [vn0]

TASK [Update apt cache] *****
changed: [vn3]
changed: [vn2]
changed: [vn0]
changed: [vn1]

TASK [Install Java 17] *****
changed: [vn2]
changed: [vn3]
changed: [vn1]
changed: [vn0]

TASK [Download Spark] *****
changed: [vn3]
```

```

Fichier  Edition  Affichage  Recherche  Terminal  Aide
TASK [Download Spark] *****
changed: [vn3]
changed: [vn2]
changed: [vn1]
changed: [vn0]

TASK [Extract Spark] *****
changed: [vn3]
changed: [vn2]
changed: [vn0]
changed: [vn1]

TASK [Create a symbolic link for Spark] *****
changed: [vn3]
changed: [vn2]
changed: [vn1]
changed: [vn0]

TASK [Set Spark environment variables in /etc/environment] *****
changed: [vn2] => (item=SPARK_HOME=/opt/spark)
changed: [vn3] => (item=SPARK_HOME=/opt/spark)
changed: [vn1] => (item=SPARK_HOME=/opt/spark)
changed: [vn0] => (item=SPARK_HOME=/opt/spark)
changed: [vn2] => (item=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/spark/bin)

TASK [Gathering Facts] *****
ok: [vn2]
ok: [vn3]
ok: [vn1]

TASK [Configure Spark workers] *****
changed: [vn1]
changed: [vn3]
changed: [vn2]

TASK [Start Spark workers] *****
changed: [vn2]
changed: [vn3]
changed: [vn1]

PLAY RECAP *****
vn0  : ok=11  changed=9  unreachable=0  failed=0  skipped=0
    : rescued=0  ignored=0
vn1  : ok=11  changed=9  unreachable=0  failed=0  skipped=0
    : rescued=0  ignored=0
vn2  : ok=11  changed=9  unreachable=0  failed=0  skipped=0
    : rescued=0  ignored=0
vn3  : ok=11  changed=9  unreachable=0  failed=0  skipped=0
    : rescued=0  ignored=0

```

- Résultat : Spark installé et configuré avec succès sur toutes les VMs.

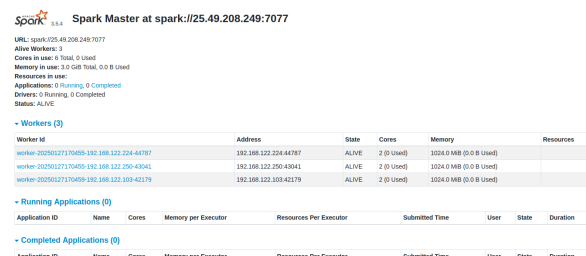


Figure 3: Interface graphique du master du spark

3 Étape 3: Validation du déploiement avec l'application WordCount

3.1 Objectif

Tester le bon fonctionnement de Spark en exécutant un job Spark WordCount.

3.2 Démarche

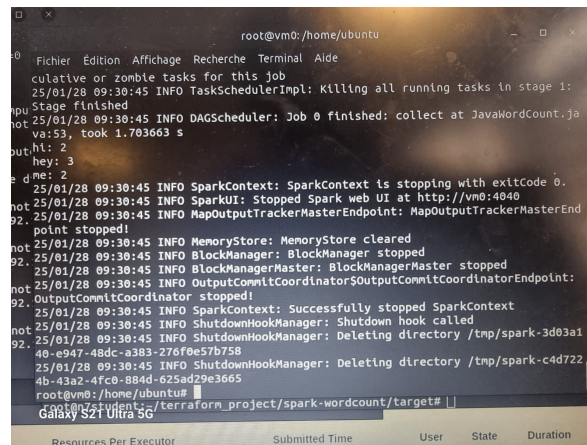
1. Préparation :

- Création d'un fichier texte d'exemple fichier.txt.
- Développement de l'application WordCount en Java avec Maven.

- Création d'un projet Maven avec Spark en dépendance.
- Implémentation du job Spark en Java pour lire un fichier texte, compter les occurrences des mots et écrire le résultat dans un fichier de sortie.
- Compilation du projet pour générer un JAR exécutable.

2. Exécution:

- Commande pour soumettre le job Spark :
`spark-submit --master spark://25.49.208.249:7077 --class com.example.WordCount target/wordcount.jar fichier.txt output.txt`
- Résultat : Le job s'exécute avec succès et génère un fichier de sortie contenant le comptage des mots.



```

root@vm0:/home/ubuntu
Fichier Edition Affichage Recherche Terminal Aide
25/01/28 09:30:45 INFO TaskSchedulerImpl: Killing all running tasks in stage 1:
culative or zombie tasks for this job
25/01/28 09:30:45 INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 1.703663 s
Output:
hey: 3
done: 2
25/01/28 09:30:45 INFO SparkContext: SparkContext is stopping with exitCode 0.
25/01/28 09:30:45 INFO SparkUI: Stopped Spark web UI at http://vm0:4040
25/01/28 09:30:45 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
25/01/28 09:30:45 INFO MemoryStore: MemoryStore cleared
25/01/28 09:30:45 INFO BlockManager: BlockManager stopped
25/01/28 09:30:45 INFO BlockManagerMaster: BlockManagerMaster stopped
25/01/28 09:30:45 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
25/01/28 09:30:45 INFO SparkContext: Successfully stopped SparkContext
25/01/28 09:30:45 INFO ShutdownHookManager: Shutdown hook called
25/01/28 09:30:45 INFO ShutdownHookManager: Deleting directory /tmp/spark-3d03a14b-e947-48dc-a383-276f0e57b758
25/01/28 09:30:45 INFO ShutdownHookManager: Deleting directory /tmp/spark-c4d7224b-43a2-4fc0-884d-625ad29e3665
root@vm0:/home/ubuntu#

```

Conclusion

Le projet a permis d'automatiser avec succès le déploiement d'un cluster Apache Spark. Terraform a simplifié la création de l'infrastructure, et Ansible a assuré une configuration cohérente et reproductible. L'application WordCount a confirmé la fiabilité du système. Ce projet peut être étendu pour inclure des échelles plus larges ou d'autres outils Big Data.