Rapport projet : Automation of Spark Deployment with Ansible and Terraform

Najwa Harrak
Oumaima El Bekkai

Département Sciences du Numérique - 3e année - Parcours IBDIOT
2024-2025

Table des matières

1 Introduction :

Le but principal de ce projet est d'automatiser le déploiement d'une infrastructure Big Data sur Apache Spark. Pour créer un environnement à la fois scalable et performant, nous avons opté pour Terraform pour gérer l'infrastructure, et Ansible pour installer et configurer Spark sur un ensemble de machines virtuelles. Ce déploiement se fait sur un cluster réparti sur deux machines physiques utilisant la plateforme de virtualisation KVM, ce qui assure une architecture à la fois flexible et facilement extensible.

Pour vérifier que l'installation et la configuration de Spark étaient réussies, nous avons utilisé une application de test, WordCount, qui permet de valider le bon fonctionnement du cluster en traitant un fichier texte et en comptant le nombre d'occurrences de chaque mot.

2 Architecture de l'infrastructure

L'infrastructure se compose de trois machines virtuelles (VM) réparties sur deux hôtes physiques. Chaque machine virtuelle joue un rôle spécifique dans le cluster Apache Spark. L'une des machines est dédiée au rôle de nœud maître (Master Node), tandis que les deux autres sont des nœuds travailleurs (Worker Nodes) qui exécutent les tâches de calcul.

Le nœud maître est responsable de la coordination des tâches entre les nœuds travailleurs et gère les ressources du cluster. Les nœuds travailleurs, quant à eux, sont chargés de l'exécution des jobs distribués et du stockage des données.

Sur le premier hôte, nous avons déployé deux machines virtuelles : une pour le nœud maître et une pour un nœud travailleur. Sur le second hôte, la autres machine virtuelle joue le role d'un travailleur. Cette configuration permet d'assurer une bonne répartition de la charge de travail tout en optimisant la résilience du cluster.

3 Build the Infrastructure

Le fichier de configuration **main.tf** contient l'ensemble des instructions nécessaires pour créer et configurer les machines virtuelles, les volumes de stockage et le réseau sur deux hôtes physiques distincts. Ce déploiement est effectué via le fournisseur libvirt, qui permet la gestion des machines virtuelles sur un environnement KVM.

Le fichier **main.tf** commence par la définition des providers, qui sont utilisés pour interagir avec les ressources de l'infrastructure. Deux providers libvirt sont configurés :

- Le provider par défaut pour l'hôte physique A, qui gère la virtualisation sur le premier serveur.
- Un provider avec alias *host_b* pour l'hôte physique B, permettant de déployer des machines virtuelles sur ce second hôte via SSH. Cette configuration permet une gestion centralisée de l'ensemble de l'infrastructure répartie sur plusieurs hôtes.

De plus, des providers pour template et cloudinit sont utilisés pour faciliter la gestion des configurations d'initialisation des VMs.

Certaines variables sont définies pour rendre la configuration flexible, telles que la quantité de mémoire (1 Go) et le nombre de CPU pour chaque machine virtuelle. D'autres variables, comme le domaine et les adresses IP des machines, sont également définies pour personnaliser les ressources créées.

4 Automate the Deployment of Spark

L'automatisation du déploiement d'Apache Spark sur l'infrastructure mise en place se fait à l'aide d'un playbook Ansible. Ansible est un outil d'automatisation qui permet de déployer, configurer et gérer des applications sur des machines distantes. Le playbook que nous avons créé couvre l'installation de Java (nécessaire pour exécuter Spark), le téléchargement et l'installation de Spark, ainsi que la configuration de l'environnement Spark sur les nœuds du cluster (Master et Workers).

4.1 Installation des Prérequis :

La première étape consiste à installer Java (OpenJDK 11), car Apache Spark nécessite Java pour fonctionner. Le playbook utilise le module apt d'Ansible pour s'assurer que Java est installé et que le cache des paquets est à jour. Ensuite, le playbook crée le répertoire /opt, si nécessaire, pour y installer Spark, et vérifie que le répertoire a les bonnes permissions.

4.2 Configuration des Variables d'Environnement :

Nous téléchargeons la version 3.3.2 d'Apache Spark depuis le dépôt officiel d'Apache .Ensuite, nous définissons les variables d'environnement nécessaires à Spark. Un script est créé dans le répertoire /etc/profile.d/ pour que ces variables soient automatiquement chargées à chaque session, ce qui garantit que les chemins de Spark sont correctement ajoutés à la variable PATH.

4.3 Configuration de Spark pour le Cluster :

Pour configurer Spark, nous devons renommer le fichier spark-env.sh.template en spark-env.sh, et définir la variable $JAVA_HOME. Le playbook s'assure que ce fichier existe et est correcte
$env.sh pour qu'il pointe vers l'installation de Java sur le système.$

4.4 Configuration du Master et des Workers :

La configuration du Master se fait en définissant la variable $SPARK_MASTER_HOST$ dans le fichie
$env.sh et en spécifiant la liste des Workers dans le fichier slaves sur le nœud master.$

4.5 Démarrage du Cluster Spark :

Une fois Spark installé et configuré, nous vérifions si un processus de Master ou de Worker est déjà en cours d'exécution et, si nécessaire, nous les arrêtons avant de les redémarrer. Cela garantit que l'environnement est dans un état propre avant

le lancement des services. Après avoir vérifié que le Master n'est pas déjà en cours d'exécution, nous le démarrons, ainsi que les Workers.

4.6 Test du Cluster Spark avec l'Application WordCount :

Pour valider que Spark fonctionne correctement, un fichier de test (test.txt) est copié sur le nœud master. Ensuite, l'application WordCount est lancée sur le cluster Spark pour vérifier que le traitement des données se fait correctement. Les résultats sont affichés dans la sortie standard. Le résultat de l'exécution de l'application WordCount est ensuite affiché pour vérifier que tout fonctionne correctement.