Université Sultan Moulay Slimane
EST – Beni Mellal
Département : ITG
Pr. Siham BAKKOURI

Filières : DUT – IDIA & SIR – S3
Module : Big Data & BD NoSQL (M236)
Année universitaire : 2025/2026

Atelier N° 01 : Installation et configuration de Hadoop avec Docker

Objectif de l'atelier

- Installer et configurer Hadoop sous Windows à l'aide de Docker et Docker Compose.
- Comprendre l'architecture de l'environnement Hadoop (HDFS, YARN, MapReduce).
- Démarrer et tester un cluster Hadoop distribué.
- Manipuler HDFS via la ligne de commande et via interface Web.
- Exécuter et analyser un job MapReduce d'exemple.

Lien de l'atelier

PARTIE 1 – Installation et préparation de l'environnement

1. Installation de WSL 2 (Windows Subsystem for Linux)

Docker Desktop utilise WSL 2 comme moteur pour exécuter les conteneurs Linux sur Windows.

→ Étapes d'installation

- Ouvrez *PowerShell* en mode **Administrateur** et exécutez :

```
≥ wsl --install
```

Cela installe automatiquement la fonctionnalité WSL 2 et une distribution Linux (par défaut Ubuntu).

- Une fois terminé

- Redémarrez votre ordinateur.
- Vérifiez la version installée :

```
≥ wsl --status
```

- **Si besoin, activez WSL 2 comme version par défaut**

≥ ***wsl --set-default-version 2***

2. **Installation de Docker Desktop**

   - Télécharger Docker Desktop à partir de ce lien : **https://docs.docker.com/desktop/setup/install/windows-install/**

- Les étapes

   1. Téléchargez l'installeur pour Windows.
   2. Lors de l'installation, **cochez la case "*Use WSL 2 instead of Hyper-V*"**.
   3. Une fois installé, redémarrez votre machine.
   4. Démarrez **Docker Desktop**.
   5. Vérifiez que le démon fonctionne.

- Vérification (PowerShell)

≥ ***docker --version***
≥ ***docker compose version***

PARTIE 2 – Création du projet Hadoop

1. **Créer un dossier de travail**

≥ ***cd C:***
≥ ***mkdir hadoop_docker***
≥ ***cd hadoop_docker***

2. **Télécharger les images Docker Hadoop**

→ Exécutez ces commandes une seule fois pour récupérer les images du cluster Hadoop :

≥ ***docker pull bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8***
≥ ***docker pull bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8***
≥ ***docker pull bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8***
≥ ***docker pull bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8***
≥ ***docker pull bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8***

Ces cinq images Docker sont toutes nécessaires, car Hadoop fonctionne selon une architecture distribuée composée de plusieurs services qui collaborent entre eux. Chaque image correspond à un composant précis d'Hadoop, et sera exécutée dans un conteneur séparé pour simuler un vrai cluster.

- **NameNode** : c'est le "cerveau" du système HDFS. Il garde la carte du système de fichiers (où sont stockés les blocs, quelles machines les possèdent, etc.).

- **DataNode** : stocke réellement les données (les blocs de fichiers). Plusieurs DataNodes peuvent exister pour répartir le stockage.
- **ResourceManager** : gère la répartition des ressources dans le cluster (c'est lui qui décide où et comment exécuter les tâches).
- **NodeManager** : exécute les tâches sur chaque nœud de calcul selon les ordres du ResourceManager.
- **HistoryServer** : conserve l'historique des traitements MapReduce (les logs, les résultats terminés, etc.).

Chaque composant doit être lancé dans un conteneur séparé, car :

- Chaque service a son rôle propre et son processus indépendant.
- Cela reproduit la vraie architecture Hadoop où chaque nœud tourne sur une machine différente.
- Docker permet de séparer et isoler ces services, tout en les faisant communiquer entre eux à travers un réseau virtuel.

Remarque

- Une **image** Docker est comme un modèle ou une "recette" d'un service.
- Un **conteneur** est l'instance vivante de cette image (le service réellement en cours d'exécution).

Exemple : l'image `bde2020/hadoop-namenode` sert à créer un conteneur qui jouera le rôle du NameNode dans notre cluster.

Ces images contiennent **Hadoop 3.2.1** configuré avec **Java 8**.

3. Téléchargez le fichier docker-compose.yml à partir de ce lien, puis placez-le dans ***C:\hadoop_docker***.

PARTIE 3 – Démarrage du cluster Hadoop

1. Lancer le cluster

→ Dans PowerShell, dans le dossier du projet :

```
≥ docker compose up -d
```

→ Vérifiez les conteneurs actifs :

```
≥ docker ps
```

- Vous devez voir
  - namenode
  - datanode
  - resourcemanager
  - nodemanager

- historyserver

PARTIE 4 – Utilisation en ligne de commande

→ Ouvrir un terminal dans le conteneur NameNode

≥ `docker exec -it namenode bash`

→ Commandes de base HDFS

≥ `hdfs dfs -mkdir /input`
≥ `hdfs dfs -ls /`
≥ `hdfs dfs -chmod -R 777 /input`
≥ `hdfs dfs -chown -R root:root /input`
≥ `hdfs dfs -ls /`
≥ `echo "Bonjour Hadoop" > /tmp/test.txt`
≥ `hdfs dfs -put /tmp/test.txt /input`
≥ `hdfs dfs -ls /input`
≥ `hdfs dfs -cat /input/test.txt`

→ Vérifier l'état du cluster

≥ `hdfs dfsadmin -report`

→ Fichiers de configuration importants

Voici les fichiers clés (dans `/opt/hadoop-3.2.1/etc/hadoop/` ou dans `/etc/hadoop/`) :

Fichier	Rôle principal
core-site.xml	Contient la configuration générale du système de fichiers (ex: fs.defaultFS)
hdfs-site.xml	Configure HDFS (réplication, chemins des données, NameNode...)
yarn-site.xml	Configure le gestionnaire de ressources YARN
mapred-site.xml	Configure le framework MapReduce
workers	Liste des DataNodes

Pour consulter un fichier :

≥ `cat /opt/hadoop-3.2.1/etc/hadoop/core-site.xml`
≥ `cat /etc/hadoop/core-site.xml`

PARTIE 5 – Interfaces Web Hadoop

Service	URL locale	Description

NameNode	http://localhost:9870	Explorer le système de fichiers HDFS
ResourceManager	http://localhost:8088	Suivre les jobs MapReduce
HistoryServer	http://localhost:8188	Voir l'historique des jobs terminés

Depuis ton navigateur

- Accède à **NameNode UI** pour explorer le contenu de HDFS (/input, /output).
- Accède à **ResourceManager UI** pour suivre le job WordCount.
- Accède à **HistoryServer UI** pour consulter le rapport final.

→ **Quitte le conteneur** :

≥ `exit`

→ **Arrêter le cluster**

≥ `docker compose down`

→ **Supprimer les volumes et conteneurs**

≥ `docker compose down -v`

PARTIE 6 – Exemple MapReduce

→ Exemple ***WordCount***

≥ `hdfs dfs -mkdir /wordcount_input`

≥ `echo "Hadoop est un framework distribué. Hadoop est puissant." > text.txt`

≥ `hdfs dfs -put text.txt /wordcount_input`

≥ `hdfs dfs -rm -r /wordcount_output  # Supprime le dossier de sortie s'il existe`

≥ `hadoop jar /opt/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar wordcount /wordcount_input /wordcount_output`

→ Afficher le résultat

≥ `hdfs dfs -ls /wordcount_output`
≥ `hdfs dfs -cat /wordcount_output/part-r-00000`