# Mise en place d'un projet Hadoop MapReduce avec Docker

---

**Objectifs du projet**

- Déployer un **cluster Hadoop** (1 master + 2 slaves) avec **Docker**.
- Configurer l'environnement de développement sur **Windows 10** avec IntelliJ IDEA.
- Développer un programme **MapReduce (WordCount)** en Java.
- Exécuter et tester le job MapReduce en local puis sur le **cluster Hadoop**.

---

## 1. Preparation de Hadoop

### 🔷 Étape 1 : Téléchargement de l'image Hadoop depuis DockerHub

Avant toute chose, on télécharge l'image préconfigurée contenant Hadoop, Spark, Kafka et HBase :

docker pull zaidelfid/hadoop-spark-kafka-hbase:v2

### 🔷 Étape 2 : Création du réseau Docker

On crée un réseau bridge qui permettra aux conteneurs (master et slaves) de communiquer :

docker network create --driver=bridge hadoop

### 🔷 Étape 3 : Création et lancement des conteneurs

On crée un **master** et deux **slaves** :

**Master :**

docker run -itd --net=hadoop -p 50070:50070 -p 8088:8088 -p 7077:7077 -p 16010:16010 --name hadoop-master --hostname hadoop-master zaidelfid/hadoop-spark-kafka-hbase:v2

**Slave 1 :**

docker run -itd -p 8040:8042 --net=hadoop --name hadoop-slave1 --hostname hadoop-slave1 zaidelfid/hadoop-spark-kafka-hbase:v2

**Slave 2 :**

docker run -itd -p 8041:8042 --net=hadoop --name hadoop-slave2 --hostname hadoop-slave2 zaidelfid/hadoop-spark-kafka-hbase:v2

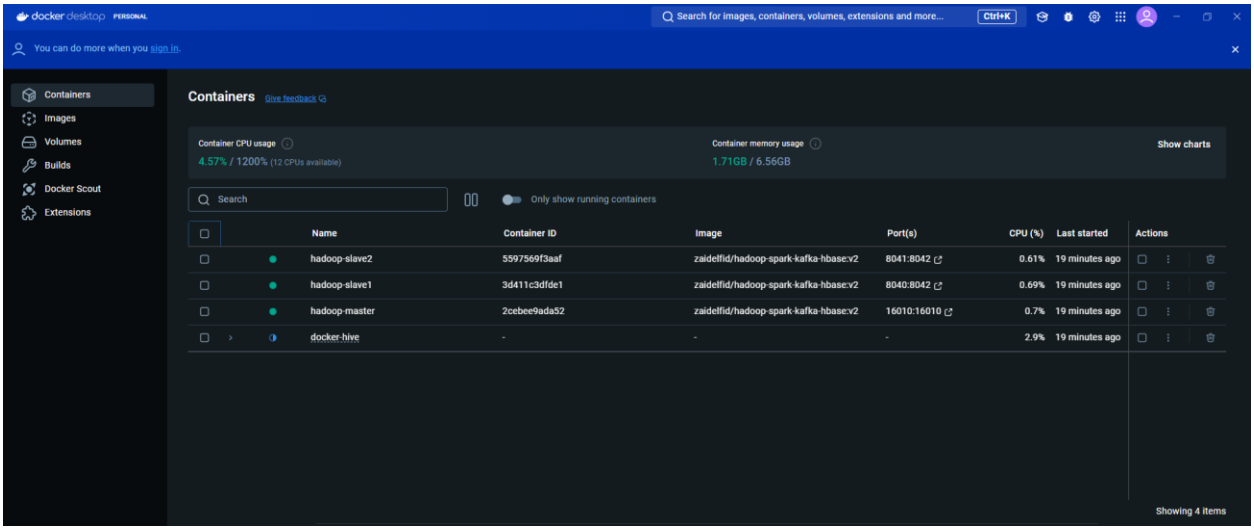
### 🔷 Étape 4 : Accès au conteneur master

Une fois les conteneurs lancés, on entre dans le master :

docker exec -it hadoop-master bash

Puis on démarre Hadoop :

./start-hadoop.sh




## 2. Configuration sur Windows

- Ajout de winutils.exe et hadoop.dll pour la compatibilité.
- Variables d'environnement :
    - HADOOP_HOME=C:\winutils
    - JAVA_HOME=C:\Program Files\Java\jdk1.8.0_xx

| | |
|---|---|
| HADOOP_HOME | C:\winutils |
| JAVA_HOME | C:\Program Files\Java\jdk-11.0.16\bin |

---

## 3. Développement du projet MapReduce

### 3.1. Création du projet Maven

- **GroupId :** hadoop.mapreduce
- **ArtifactId :** wordcount
- **Version :** 1.0

### 3.2. Dépendances Maven

Ajout dans pom.xml :

<dependencies>

```
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>2.7.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-core</artifactId>
    <version>2.7.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-hdfs</artifactId>
    <version>2.7.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-common</artifactId>
    <version>2.7.2</version>
  </dependency>
</dependencies>
```

**3.3. Implémentation des classes**

- **Mapper :** découpe le texte en mots et émet (mot, 1)
- **Reducer :** agrège les valeurs pour chaque mot
- **Runner :** configure et lance le job

---

## 4. Tests locaux

### 4.1. Fichier d'entrée

src/main/resources/input/file.txt

Hello Wordcount!

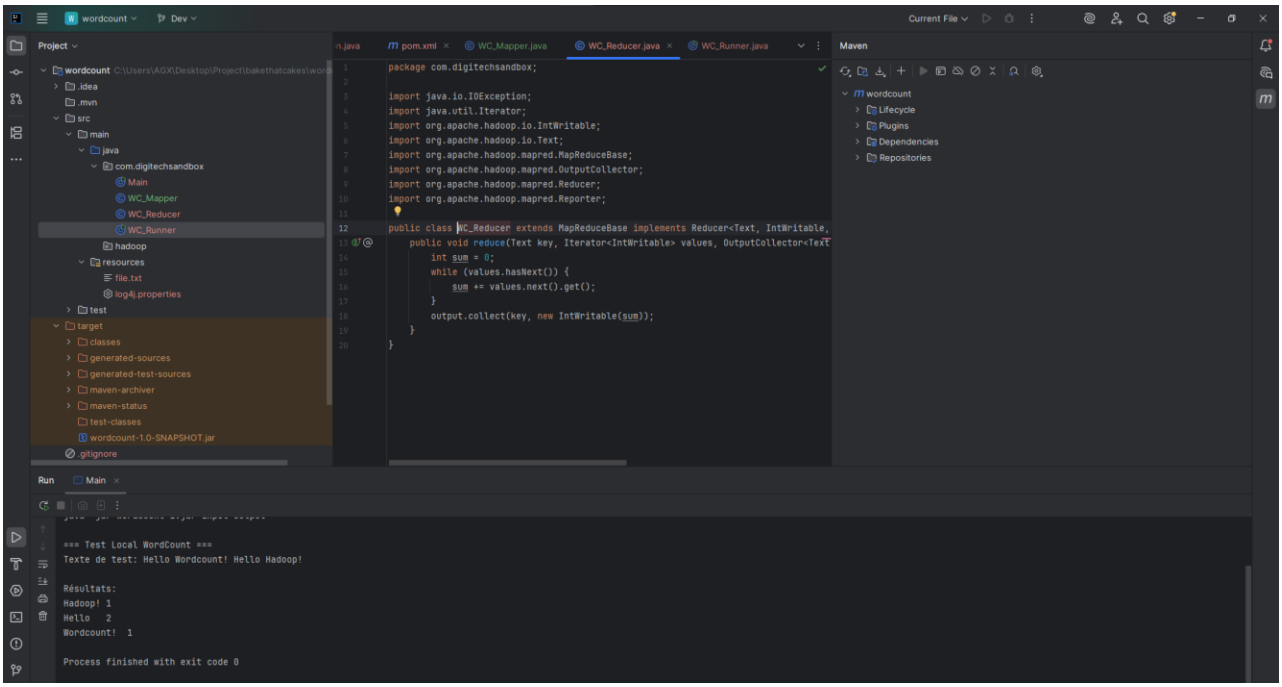
Hello Hadoop!

### 4.2. Résultats obtenus (part-r-00000)

Hadoop!  1

Hello  2

Wordcount! 1




## 7. Exécution sur le cluster Hadoop

### 5.1 Compilation et génération du JAR

mvn package install

docker cp target/wordcount-1.0-SNAPSHOT.jar hadoop-master:/root/ wordcount-1.0-SNAPSHOT.jar

```
PS C:\Users\AGX\Desktop\Project\bakethatcakes\wordcount> docker cp target/wordcount-1.0-SNAPSHOT.jar hadoop-master:/root/wordcount-1.0-SNAPSHOT.jar
Successfully copied 8.7kB to hadoop-master:/root/wordcount-1.0-SNAPSHOT.jar
```

### 5.2. Exécution du job

hadoop jar wordcount-1.0-SNAPSHOT.jar com.digitechsandbox.WC_Runner input output

**REALISER PAR :**
**YASSIR EL GHRISSI**