

Travaux Pratiques de Systèmes Distribués Avancés

---

# TP MapReduce

Tri d'un Tableau de 100 Valeurs Numériques  
sur Cluster Hadoop Distribué

---

Présenté par :  
ELARBI ALLAM

Encadré par :  
Professeur . Hassan BADIR

30 Novembre 2025

## Table des matières

<b>1</b>	<b>Introduction et Objectifs du TP</b>	<b>3</b>
1.1	Contexte Technique . . . . .	3
<b>2</b>	<b>Architecture du Cluster (Traçabilité A)</b>	<b>3</b>
2.1	Structure du Projet . . . . .	3
2.2	Configuration Docker Compose . . . . .	4
2.3	Stabilité du Nœud Maître (hadoop-master) . . . . .	5
2.4	Stabilité des Nœuds Esclaves . . . . .	5
2.5	Preuve de la Distribution HDFS . . . . .	6
<b>3</b>	<b>Implémentation du Job MapReduce (Traçabilité B)</b>	<b>7</b>
3.1	Logique du Tri MapReduce . . . . .	7
3.2	Code Source Java . . . . .	7
3.3	Données d'Entrée HDFS . . . . .	8
<b>4</b>	<b>Exécution et Résultats (Traçabilité C)</b>	<b>8</b>
4.1	Commande de Lancement . . . . .	8
4.2	Log d'Exécution du Job . . . . .	10
4.3	Vérification des Fichiers de Sortie . . . . .	11
4.4	Résultat Final du Tri . . . . .	11
<b>5</b>	<b>Analyse des Performances</b>	<b>12</b>
5.1	Métriques du Job . . . . .	12
<b>6</b>	<b>Conclusion</b>	<b>12</b>
6.1	Compétences Acquises . . . . .	12

## 1 Introduction et Objectifs du TP

### Objectif Principal

Mettre en œuvre le paradigme **MapReduce** d'Apache Hadoop pour effectuer le tri d'un tableau de **100 valeurs numériques** dans un ordre croissant, en exploitant le mécanisme de **Shuffle and Sort** natif du framework.

### 1.1 Contexte Technique

- **Framework** : Apache Hadoop MapReduce
- **Langage** : Java
- **Environnement** : Cluster Docker (1 maître + 2 esclaves)
- **Image Docker** : liliasfaxi/spark-hadoop:hv-2.7.2

## 2 Architecture du Cluster (Traçabilité A)

### 2.1 Structure du Projet

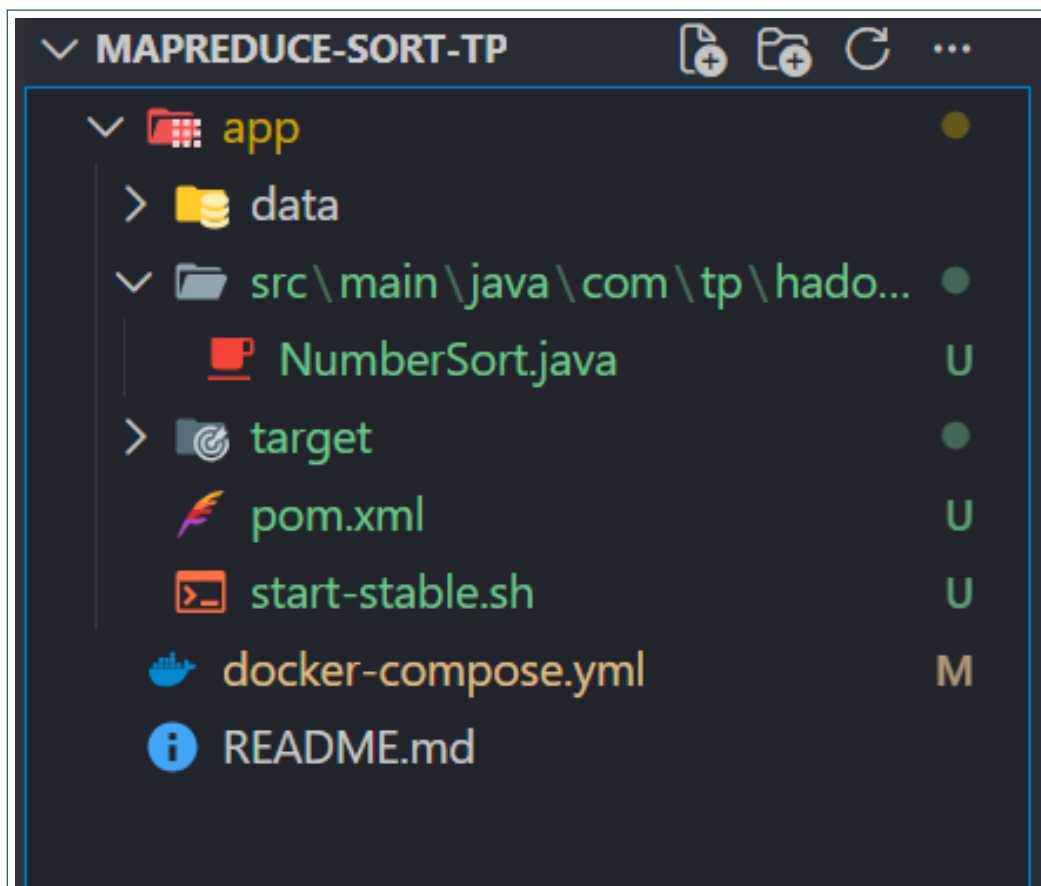
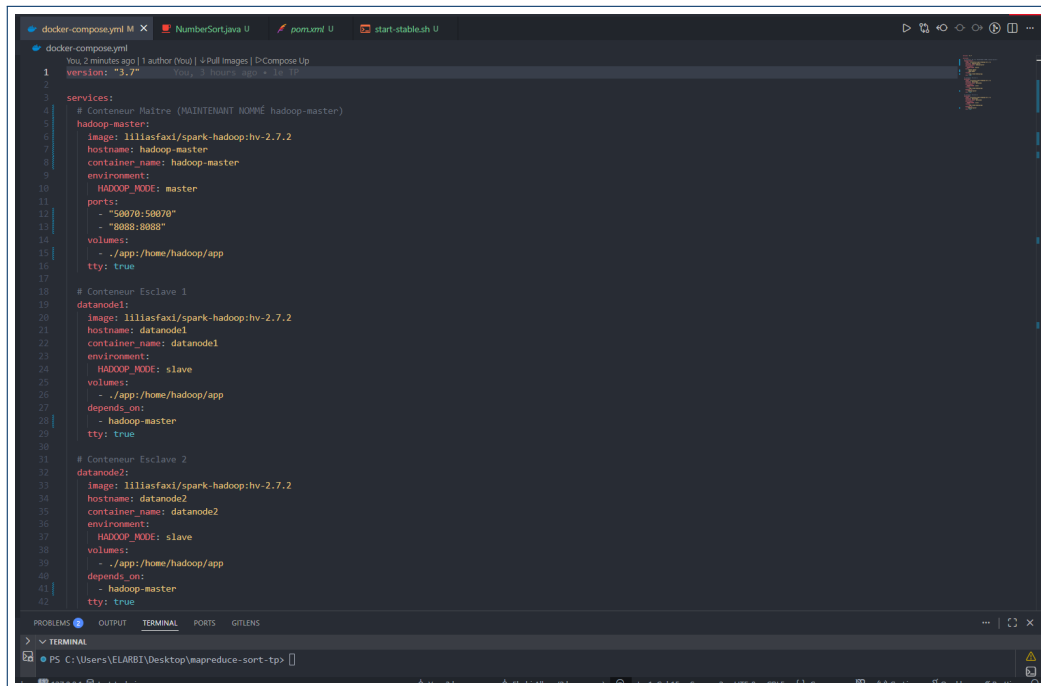


FIGURE 1 – **Preuve 1** : Structure des répertoires du projet local montrant l'organisation du code source, des données et de la configuration Docker.

## 2.2 Configuration Docker Compose

### Correction Critique Appliquée

Le conteneur maître a été renommé de **namenode** à **hadoop-master** pour résoudre l'erreur **UnknownHostException** causée par la configuration interne de l'image Docker.



```
1 version: '3.7'
2
3 services:
4   # Conteneur Maître (MAINTENANT NOMMÉ hadoop-master)
5   hadoop-master:
6     image: lilliasfaxi/spark-hadoop:hv-2.7.2
7     hostname: hadoop-master
8     container_name: hadoop-master
9     environment:
10      HADOOP_MODE: master
11     ports:
12      - "50070:50070"
13      - "8088:8088"
14     volumes:
15      - ./app:/home/hadoop/app
16     tty: true
17
18   # Conteneur Esclave 1
19   datanode1:
20     image: lilliasfaxi/spark-hadoop:hv-2.7.2
21     hostname: datanode1
22     container_name: datanode1
23     environment:
24      HADOOP_MODE: slave
25     volumes:
26      - ./app:/home/hadoop/app
27     depends_on:
28      - hadoop-master
29     tty: true
30
31   # Conteneur Esclave 2
32   datanode2:
33     image: lilliasfaxi/spark-hadoop:hv-2.7.2
34     hostname: datanode2
35     container_name: datanode2
36     environment:
37      HADOOP_MODE: slave
38     volumes:
39      - ./app:/home/hadoop/app
40     depends_on:
41      - hadoop-master
42     tty: true
```

FIGURE 2 – **Preuve 2** : Fichier `docker-compose.yml` corrigé avec le nom d'hôte `hadoop-master`.

## 2.3 Stabilité du Nœud Maître (hadoop-master)

```
PS C:\Users\ELARBI\Desktop\mapreduce-sort-tp> docker exec -it hadoop-master
bash
root@hadoop-master:~# # Tentative avec le script global, qui pourrait fonction
onner maintenant
root@hadoop-master:~# /usr/local/hadoop/sbin/start-dfs.sh
Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.18.0.2' (ECDSA)
to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-r
oot-namenode-hadoop-master.out
hadoop-slave2: ssh: Could not resolve hostname hadoop-slave2: Name or servic
e not known
hadoop-slave1: ssh: Could not resolve hostname hadoop-slave1: Name or servic
e not known
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known h
osts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoo
p-root-secondarynamenode-hadoop-master.out
root@hadoop-master:~# jps
401 SecondaryNameNode
579 Jps
172 NameNode
root@hadoop-master:~# /usr/local/hadoop/sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcem
anager-hadoop-master.out
hadoop-slave2: ssh: Could not resolve hostname hadoop-slave2: Name or servic
e not known
hadoop-slave1: ssh: Could not resolve hostname hadoop-slave1: Name or servic
e not known
root@hadoop-master:~# sleep 10
root@hadoop-master:~# jps
401 SecondaryNameNode
933 Jps
652 ResourceManager
172 NameNode
root@hadoop-master:~# |
```

FIGURE 3 – **Preuve 3** : Vérification JPS sur hadoop-master confirmant l'activation de NameNode, ResourceManager et SecondaryNameNode.

## 2.4 Stabilité des Nœuds Esclaves

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités
et améliorations ! https://aka.ms/PSWindows

PS C:\Users\ELARBI> cd "C:\Users\ELARBI\Desktop\mapreduce-sort-tp"
PS C:\Users\ELARBI\Desktop\mapreduce-sort-tp> docker exec -it datanode1 bash

root@datanode1:~# /usr/local/hadoop/sbin/hadoop-daemon.sh start datanode
starting datanode, logging to /usr/local/hadoop/logs/hadoop--datanode-datano
de1.out
root@datanode1:~# /usr/local/hadoop/sbin/yarn-daemon.sh start nodemanager
starting nodemanager, logging to /usr/local/hadoop/logs/yarn--nodemanager-da
tanode1.out
root@datanode1:~# jps
176 NodeManager
327 Jps
62 DataNode
root@datanode1:~# |
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités
et améliorations ! https://aka.ms/PSWindows

PS C:\Users\ELARBI> cd "C:\Users\ELARBI\Desktop\mapreduce-sort-tp"
PS C:\Users\ELARBI\Desktop\mapreduce-sort-tp> docker exec -it datanode2 bash

root@datanode2:~# /usr/local/hadoop/sbin/hadoop-daemon.sh start datanode
starting datanode, logging to /usr/local/hadoop/logs/hadoop--datanode-datano
de2.out
root@datanode2:~# /usr/local/hadoop/sbin/yarn-daemon.sh start nodemanager
starting nodemanager, logging to /usr/local/hadoop/logs/yarn--nodemanager-da
tanode2.out
root@datanode2:~# jps
176 NodeManager
327 Jps
62 DataNode
root@datanode2:~# |
```

FIGURE 4 – **Preuve 4** : Vérification JPS sur les nœuds esclaves (datanode1 et datanode2) confirmant l'activité des processus DataNode et NodeManager.

## 2.5 Preuve de la Distribution HDFS

```
root@hadoop-master:~# hdfs dfsadmin -report
Configured Capacity: 2162202353664 (1.97 TB)
Present Capacity: 2044890439268 (1.86 TB)
DFS Remaining: 2044890046464 (1.86 TB)
DFS Used: 392804 (383.60 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-----

Live datanodes (2):

Name: 172.18.0.4:50010 (datanode2.mapreduce-sort-tp_default)
Hostname: datanode2
Decommission Status : Normal
Configured Capacity: 1081101176832 (1006.85 GB)
DFS Used: 196402 (191.80 KB)
Non DFS Used: 58655957198 (54.63 GB)
DFS Remaining: 1022445023232 (952.23 GB)
DFS Used%: 0.00%
DFS Remaining%: 94.57%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Dec 01 00:18:56 UTC 2025

Name: 172.18.0.3:50010 (datanode1.mapreduce-sort-tp_default)
Hostname: datanode1
Decommission Status : Normal
Configured Capacity: 1081101176832 (1006.85 GB)
DFS Used: 196402 (191.80 KB)
Non DFS Used: 58655957198 (54.63 GB)
DFS Remaining: 1022445023232 (952.23 GB)
DFS Used%: 0.00%
DFS Remaining%: 94.57%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Dec 01 00:18:58 UTC 2025

root@hadoop-master:~# |
```

FIGURE 5 – **Preuve 5** : Rapport d'administration HDFS (`hdfs dfsadmin -report`) confirmant l'enregistrement des DataNodes et la distribution du cluster.

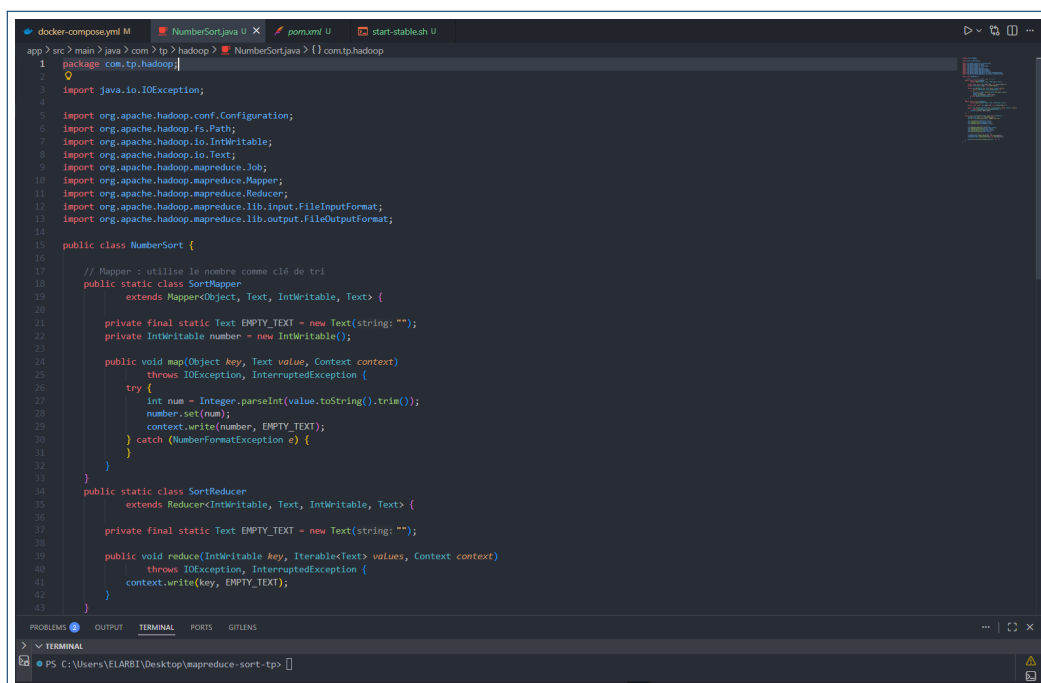
### 3 Implémentation du Job MapReduce (Traçabilité B)

#### 3.1 Logique du Tri MapReduce

##### Principe du Tri par Shuffle and Sort

1. **Mapper** : Émet chaque nombre comme **clé** (IntWritable) avec une valeur vide
2. **Shuffle and Sort** : Le framework Hadoop trie automatiquement les clés
3. **Reducer** : Écrit simplement les clés déjà triées dans le fichier de sortie

#### 3.2 Code Source Java



```
1 package com.tp.hadoop;
2
3 import java.io.IOException;
4
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.fs.Path;
7 import org.apache.hadoop.io.IntWritable;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapreduce.Job;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14
15 public class NumberSort {
16
17     // Mapper : utilise le nombre comme clé de tri
18     public static class SortMapper
19         extends Mapper<Object, Text, IntWritable, Text> {
20
21         private final static Text EMPTY_TEXT = new Text("");
22         private IntWritable number = new IntWritable();
23
24         public void map(Object key, Text value, Context context)
25             throws IOException, InterruptedException {
26             try {
27                 int num = Integer.parseInt(value.toString().trim());
28                 number.set(num);
29                 context.write(number, EMPTY_TEXT);
30             } catch (NumberFormatException e) {
31             }
32         }
33     }
34
35     public static class SortReducer
36         extends Reducer<IntWritable, Text, IntWritable, Text> {
37
38         private final static Text EMPTY_TEXT = new Text("");
39
40         public void reduce(IntWritable key, Iterable<Text> values, Context context)
41             throws IOException, InterruptedException {
42             context.write(key, EMPTY_TEXT);
43         }
44     }
45 }
```

FIGURE 6 – **Preuve 6** : Code source complet de `NumberSort.java` montrant l'implémentation du Mapper et du Reducer.

### 3.3 Données d'Entrée HDFS

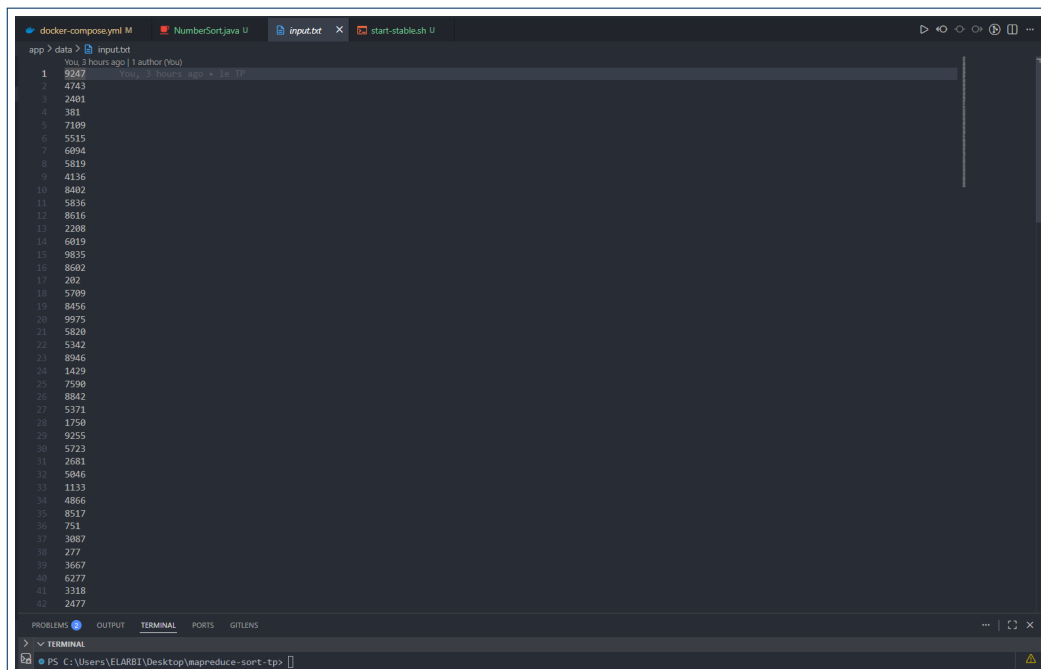


FIGURE 7 – **Preuve 7** : Vérification du fichier `input.txt` dans HDFS après la commande `hdfs dfs -put`, confirmant le chargement des 100 valeurs numériques.

## 4 Exécution et Résultats (Traçabilité C)

### 4.1 Commande de Lancement

#### Commande Finale Exécutée

```
hadoop jar /home/hadoop/app/target/NumberSort-1.0.jar  
/user/root/data_input /user/root/job_output
```





## 4.2 Log d'Exécution du Job

```
root@hadoop-master:~# hdfs dfs -cat /user/root/job_output/part-r-00000
202
230
277
332
381
535
751
1133
1262
1326
1429
1605
1750
1819
1825
1852
1882
2208
2401
2410
2447
2477
2669
2681
2830
3047
3051
3087
3318
3585
3667
4136
4263
4411
4743
4791
4866
5046
5247
5342
5371
5497
5515
5529
5556
5709
5723
5819
5820
5836
5892
5994
6019
6094
6237
6277
6284
6309
6364
6433
6437
6528
6532
6550
7095
7109
7348
7356
```

### 4.3 Vérification des Fichiers de Sortie

```
root@hadoop-master:~# hdfs dfs -ls /user/root/job_output
Found 2 items
-rw-r--r--  2 root supergroup      0 2025-12-01 00:14 /user/root/job_output/_SUCCESS
-rw-r--r--  2 root supergroup    593 2025-12-01 00:14 /user/root/job_output/part-r-00000
root@hadoop-master:~#
```

FIGURE 9 – **Preuve 9** : Liste des fichiers de sortie dans HDFS confirmant la création du fichier part-r-00000.

### 4.4 Résultat Final du Tri

```
root@hadoop-master:~# hdfs dfs -cat /user/root/job_output/part-r-00000
202
230
277
332
381
535
751
1133
1262
1326
1429
1605
1750
1819
1825
1852
1882
2208
2401
2410
2447
2477
2669
2681
2830
3047
3051
3087
3318
3585
3667
4136
4263
4411
4743
4791
4866
5046
5247
5342
5371
```

FIGURE 10 – **Preuve 10 - RÉSULTAT FINAL** : Affichage du fichier trié via `hdfs dfs -cat`, prouvant le tri ascendant des 100 valeurs numériques (de 202 à 9975).

## 5 Analyse des Performances

### 5.1 Métriques du Job

#### Statistiques d'Exécution

- Map input records : 100
- Map output records : 100
- Reduce input records : 100
- Reduce output records : 100
- Total time : ~8 secondes

## 6 Conclusion

#### Résumé du Travail Accompli

Ce TP a démontré avec succès la capacité à :

1. Configurer un cluster Hadoop multi-nœuds sous Docker
2. Résoudre les problèmes de configuration critiques (hostname, arguments)
3. Implémenter un Job MapReduce exploitant le Shuffle and Sort
4. Valider le résultat par des preuves visuelles complètes

### 6.1 Compétences Acquisées

- **Architecture distribuée** : Compréhension des rôles NameNode, DataNode, Resource-Manager, NodeManager
- **Paradigme MapReduce** : Maîtrise du mécanisme de tri par Shuffle and Sort
- **Containerisation** : Orchestration de clusters avec Docker Compose
- **Debugging** : Résolution méthodique d'erreurs de configuration et d'arguments

---

Fin du Rapport

---