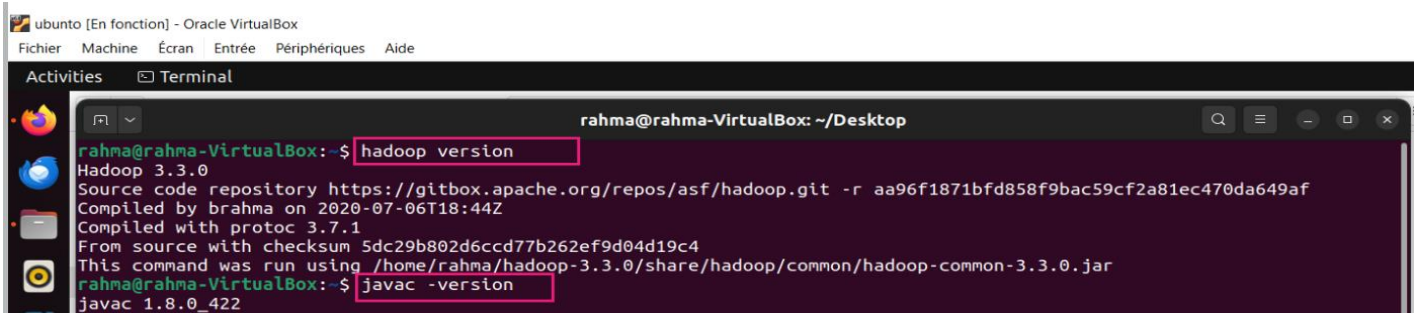


1/ Vérifier que Hadoop et Java sont correctement installés.

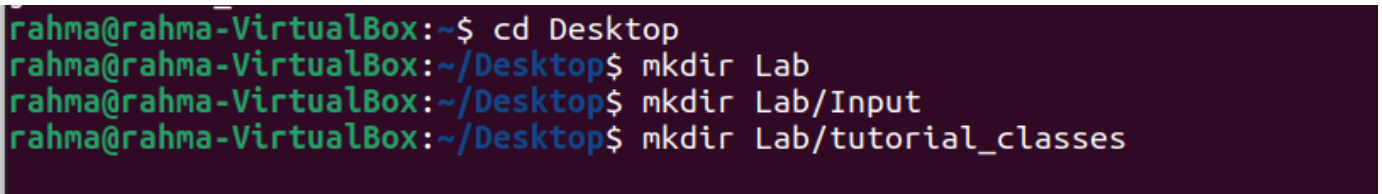
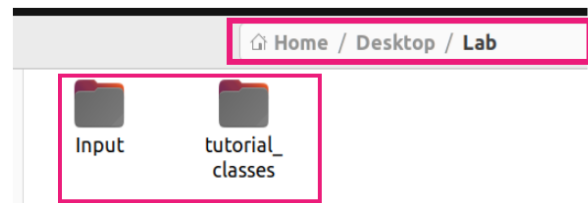
```
hadoop version  
javac -version
```



A terminal window titled 'rahma@rahma-VirtualBox: ~/Desktop' showing the output of 'hadoop version' and 'javac -version'. The 'hadoop version' command outputs: 'Hadoop 3.3.0', 'Source code repository https://gitbox.apache.org/repos/asf/hadoop.git -r aa96f1871bfd858f9bac59cf2a81ec470da649af', 'Compiled by brahma on 2020-07-06T18:44Z', 'Compiled with protoc 3.7.1', 'From source with checksum 5dc29b802d6ccd77b262ef9d04d19c4', and 'This command was run using /home/rahma/hadoop-3.3.0/share/hadoop/common/hadoop-common-3.3.0.jar'. The 'javac -version' command outputs: 'javac 1.8.0\_422'.

2/ Créer un répertoire " Lab" sur le bureau avec deux sous-répertoires "Input" et "tutorial\_classes".

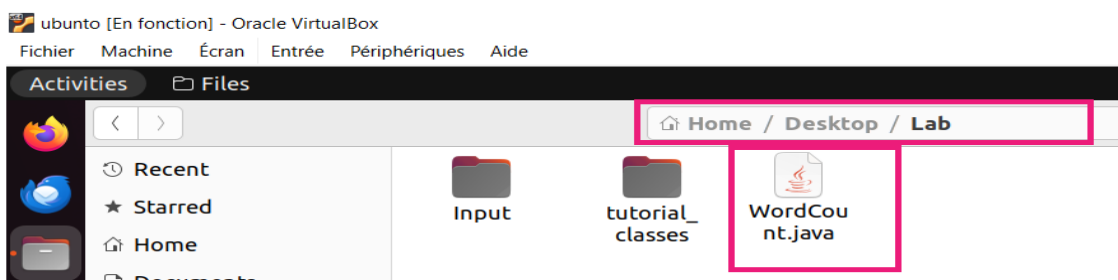
```
cd Desktop  
mkdir Lab  
mkdir Lab/Input  
mkdir Lab/tutorial_classes
```




A terminal window showing the execution of the following commands: 'cd Desktop', 'mkdir Lab', 'mkdir Lab/Input', and 'mkdir Lab/tutorial\_classes'.

3/créer le fichier "WordCount.java" dans le répertoire "Lab".

🔧 Taper : `touch WordCount.java`



 File : WordCount.java

```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

// WordCount

public class WordCount {

    // Classe TokenizerMapper

    public static class TokenizerMapper

        extends Mapper<Object, Text, Text, IntWritable>{
```

```
private final static IntWritable one = new IntWritable(1);
```

```
private Text word = new Text();
```

#### // Méthode map

```
public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {

    StringTokenizer itr = new StringTokenizer(value.toString());

    while (itr.hasMoreTokens()) {

        word.set(itr.nextToken());

        context.write(word, one);

    } } }
```

#### // Classe IntSumReducer

```
public static class IntSumReducer

    extends Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,

                        Context context

                        ) throws IOException, InterruptedException {

        int sum = 0;
```

```
for (IntWritable val : values) {  
  
    sum += val.get(); }  
  
    result.set(sum);  
  
    context.write(key, result);  
  
}}
```

### // Méthode principale

```
public static void main(String[] args) throws Exception {  
  
    Configuration conf = new Configuration();  
  
    Job job = Job.getInstance(conf, "word count");  
  
    job.setJarByClass(WordCount.class);  
  
    job.setMapperClass(TokenizerMapper.class);  
  
    job.setCombinerClass(IntSumReducer.class);  
  
    // Optionnel, pour réduire le trafic de données  
  
    job.setReducerClass(IntSumReducer.class);  
  
    job.setOutputKeyClass(Text.class);  
  
    job.setOutputValueClass(IntWritable.class);  
  
    // Chemin d'entrée  
  
    FileInputFormat.addInputPath(job, new Path(args[0]));
```

```
// Chemin de sortie

FileOutputFormat.setOutputPath(job, new Path(args[1]));

//execution

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

////////////////////////////////////fin class //////////////////////////////////////
```

## Une explication du programme WordCount.

### 1. Classe WordCount

- **Fonctionnalité** : Contient la méthode main, qui initialise la configuration, définit les classes Mapper et Reducer, et gère les chemins d'entrée et de sortie.

### 2. Classe TokenizerMapper

- **Fonctionnalité** :
  - **Méthode map** : Divise chaque ligne en mots à l'aide de StringTokenizer. Chaque fois qu'un mot est trouvé, une occurrence de 1 lui est attribuée, permettant ainsi de compter le nombre total d'occurrences de chaque mot

### 3. Classe IntSumReducer

- **Fonctionnalité** :
  - **Méthode reduce** : Prend un mot (clé) et un ensemble de valeurs (occurrences). Calcule la somme des occurrences pour chaque mot et écrit le mot avec son total dans le contexte, produisant ainsi le résultat final du comptage.

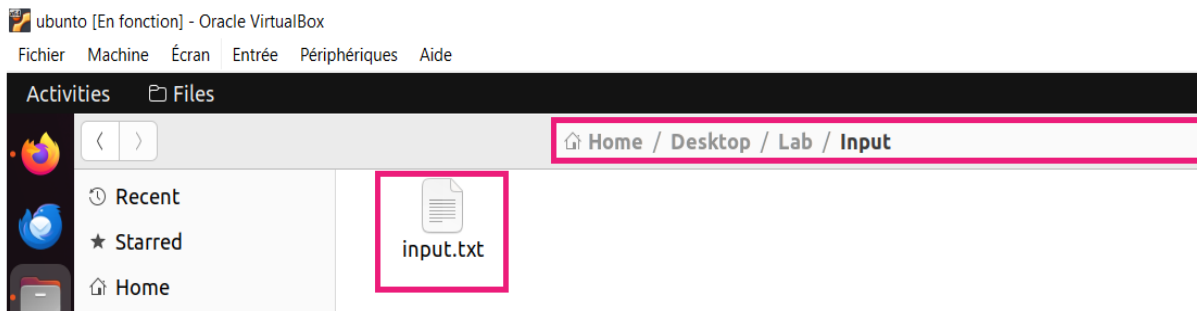
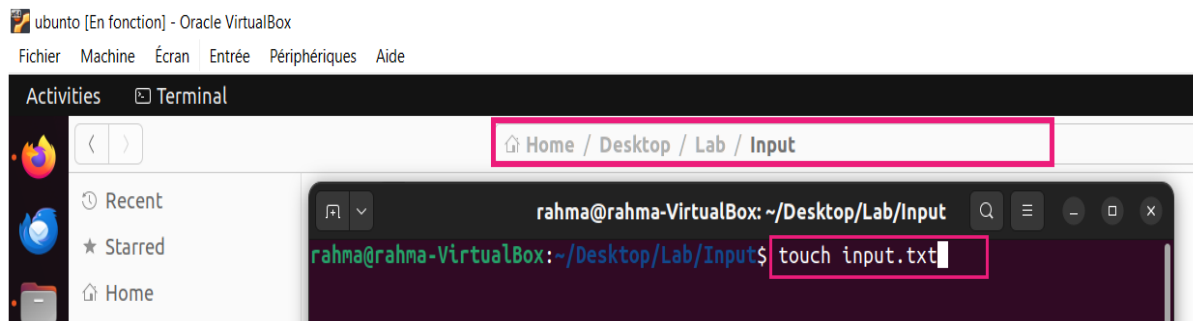
### 4. Méthode main


- **Fonctionnalité** :
  - Configure le job MapReduce, spécifie les classes Mapper et Reducer, définit les types de clés et de valeurs, et gère les chemins d'entrée et de sortie. Lance le job et gère la sortie du programme en fonction de l'exécution complète du traitement.

4/Ajouter le fichier "input.txt" dans le répertoire "Lab/Input".

✚ Taper :

```
cd Desktop  
cd Lab  
cd Input  
touch input.txt
```



 File: **input.txt**

tunis

ariana

ben arous

manouba

nabeul

zaghouan

bizerte

béja

jendouba

le kef

siliana

kairouan

kasserine

sidi bouzid

sousse

monastir

mahdia

sfax

gabès

médenine

tataouine

tozeur

kebili

gafsa

tunis

ariana

ben arous

manouba

nabeul

zaghouan

bizerte

béja

jendouba

le kef

siliana

kairouan

kasserine

sidi bouzid

sousse

monastir

mahdia

sfax

gabès

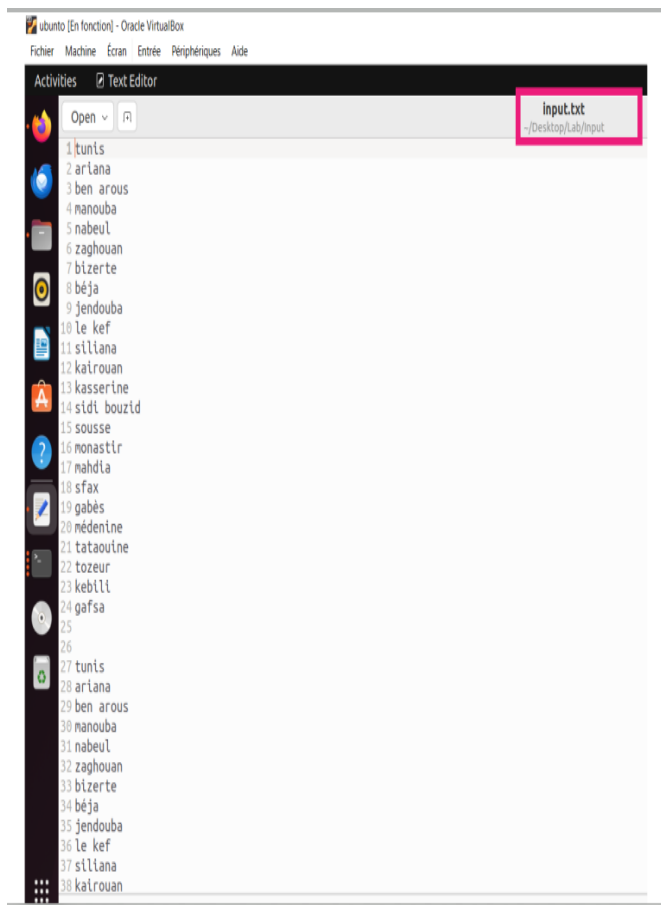
médenine

tataouine

tozeur

kebili

gafsa



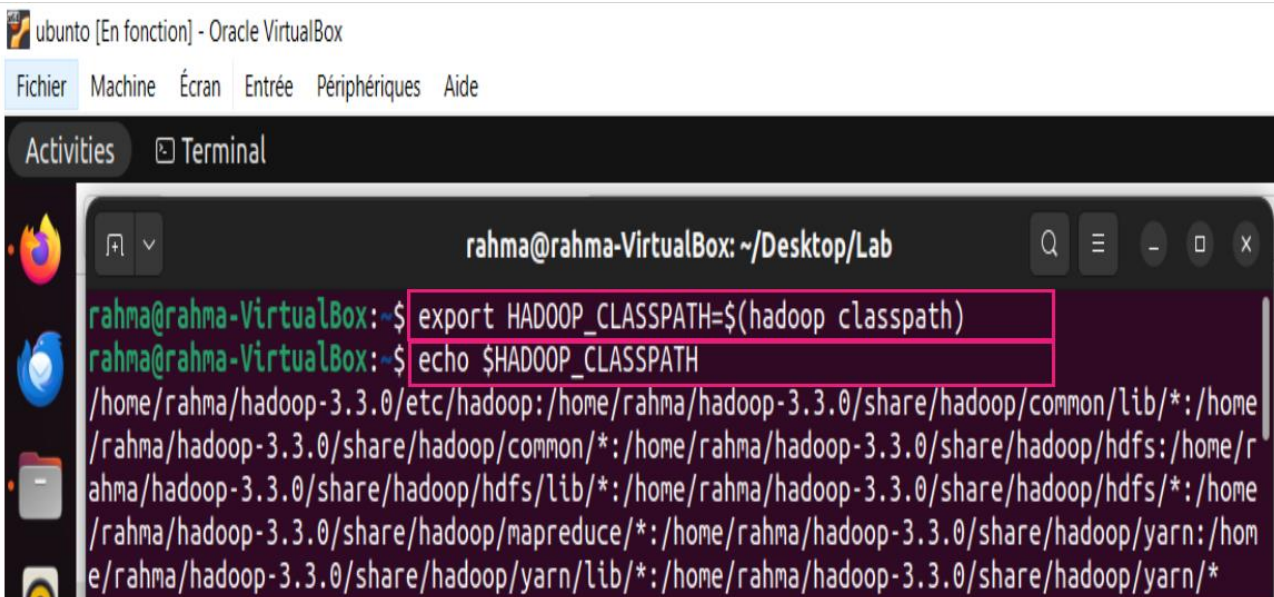


5/ Exporter le classpath Hadoop dans la variable d'environnement.

```
export HADOOP_CLASSPATH=$(hadoop classpath)
```

```
echo $HADOOP_CLASSPATH
```

==> Cette commande configure la variable d'environnement HADOOP\_CLASSPATH pour inclure les chemins nécessaires aux bibliothèques Hadoop, afin de faciliter l'exécution des programmes Hadoop.

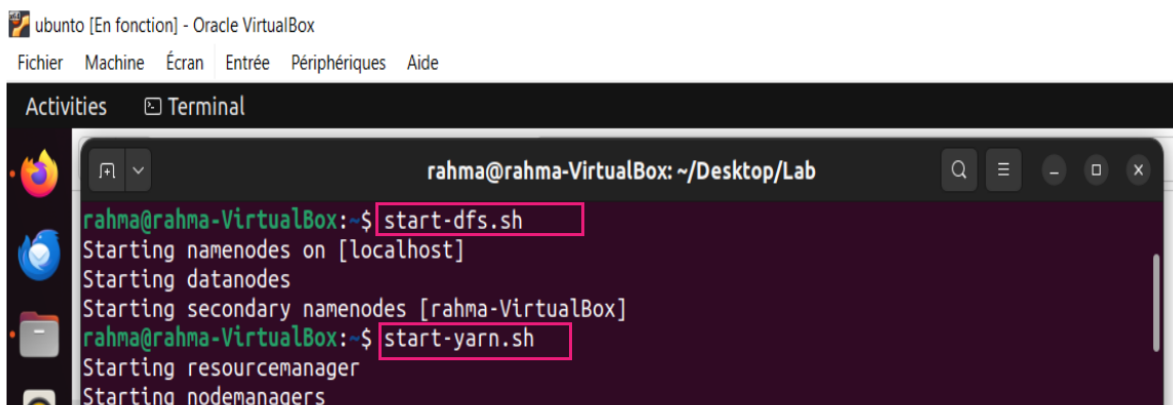


The screenshot shows a terminal window titled 'rahma@rahma-VirtualBox: ~/Desktop/Lab'. The user has entered two commands: `export HADOOP_CLASSPATH=$(hadoop classpath)` and `echo $HADOOP_CLASSPATH`. The output of the echo command is a long string of Hadoop classpath entries, including paths for `etc/hadoop`, `share/hadoop/common/lib`, `share/hadoop/common`, `share/hadoop/hdfs`, `share/hadoop/hdfs/lib`, `share/hadoop/hdfs`, `share/hadoop/mapreduce`, `share/hadoop/yarn`, and `share/hadoop/yarn/lib`.

🚀 Démarrer les services DFS et YARN dans Hadoop

====> `start-dfs.sh`

====> `start-yarn.sh`



The screenshot shows a terminal window titled 'rahma@rahma-VirtualBox: ~/Desktop/Lab'. The user has entered two commands: `start-dfs.sh` and `start-yarn.sh`. The output of `start-dfs.sh` is: `Starting namenodes on [localhost]`, `Starting datanodes`, and `Starting secondary namenodes [rahma-VirtualBox]`. The output of `start-yarn.sh` is: `Starting resourcemanager` and `Starting nodemanagers`.

6/ Créer les répertoires nécessaires sur le système de fichiers HDFS.

```
hadoop fs -mkdir /WordCountTutorial
```

=====> Cette commande crée un répertoire nommé /WordCountTutorial dans le système de fichiers Hadoop (HDFS).

```
hadoop fs -mkdir /WordCountTutorial/Input
```

=====> Cette commande crée un dossier *Input* dans le dossier *WordCountTutorial* dans le système de fichiers Hadoop (HDFS).

```
cd Desktop
```

```
hadoop fs -put Lab/Input/input.txt /WordCountTutorial/Input
```

=====> Cette commande copie le fichier input.txt depuis le répertoire local Lab/Input/ vers le répertoire /WordCountTutorial/Input du système de fichiers HDFS.

```
rahma@rahma-VirtualBox:~$ hadoop fs -mkdir /WordCountTutorial
rahma@rahma-VirtualBox:~$ hadoop fs -mkdir /WordCountTutorial/Input
```

```
rahma@rahma-VirtualBox:~/Desktop$ hadoop fs -put Lab/Input/input.txt /WordCountTutorial/Input
```

7/Vérifier l'existence des répertoires et fichiers sur l'interface web de Hadoop.

Hadoop NameNode ==> <http://localhost:9870>

=====> Allez à localhost:9870 depuis le navigateur, ouvrez "Utilitaires → Parcourir le système de fichiers" et vous devriez voir les répertoires et fichiers que nous avons placés dans le système de fichiers.

Started:	Wed Nov 13 19:50:54 +0100 2024
Version:	3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled:	Mon Jul 06 19:44:00 +0100 2020 by brahma from branch-3.3.0
Cluster ID:	CID-9dbadc4e-7b0d-4fa9-9b84-8954e5d79e38
Block Pool ID:	BP-1999807259-127.0.1.1-1729126664024

The screenshot shows the Hadoop web interface with the following details:

- Browser tabs: Restore Session, Namenode information, All Applications, Browsing HDFS.
- Address bar: localhost:9870/explorer.html#/WordCountTutorial
- Navigation bar: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, Utilities.
- Page title: Browse Directory
- Search bar: /WordCountTutorial, Go!
- Buttons: Folder, Upload, Download, Refresh.
- Filter: Show 25 entries
- Search: Search: [input field]
- Table headers: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, Name.
- Table data:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	rahma	supergroup	0 B		0	0 B	Input

The screenshot shows the Hadoop web interface with the following details:

- Browser tabs: Restore Session, Namenode information, All Applications, Browsing HDFS.
- Address bar: localhost:9870/explorer.html#/WordCountTutorial/Input
- Navigation bar: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, Utilities.
- Page title: Browse Directory
- Search bar: /WordCountTutorial/Input, Go!
- Buttons: Folder, Upload, Download, Refresh.
- Filter: Show 25 entries
- Search: Search: [input field]
- Table headers: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, Name.
- Table data:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	rahma	supergroup	385 B		1	128 MB	input.txt
- Footer: Showing 1 to 1 of 1 entries, Previous, 1, Next.

The screenshot shows the 'File information - input.txt' dialog box with the following details:

- Buttons: Download, Head the file (first 32K), Tail the file (last 32K).
- Block information: Block 0 (selected).
- Block ID: 1073741825
- Block Pool ID: BP-1999807259-127.0.1.1-1729126664024
- Generation Stamp: 1001
- Size: 385
- Availability: rahma-VirtualBox
- Close button.

8/ Compiler le code Java WordCount.java et créer un fichier jar.

```
cd Desktop
cd Lab
javac -classpath $HADOOP_CLASSPATH -d tutorial_classes WordCount.java
```

=====> Cette commande compile les fichiers Java avec le classpath Hadoop et place les classes compilées dans le dossier tutorial\_classes.

```
jar -cvf WordCount.jar -C tutorial_classes .
```

=====> Mettre les fichiers de sortie dans un seul fichier jar.

```
rahma@rahma-VirtualBox:~/Desktop$ cd Lab
rahma@rahma-VirtualBox:~/Desktop/Lab$ javac -classpath $HADOOP_CLASSPATH -d tutorial_classes WordCount.java
rahma@rahma-VirtualBox:~/Desktop/Lab$ jar -cvf WordCount.jar -C tutorial_classes .
added manifest
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 754)(deflated 56%)
adding: WordCount$IntSumReducer.class(in = 1739) (out= 739)(deflated 57%)
adding: WordCount.class(in = 1491) (out= 814)(deflated 45%)
```



9/ Exécuter le programme WordCount en utilisant le fichier jar sur Hadoop.

```
hadoop jar WordCount.jar WordCount /WordCountTutorial/Input
/WordCountTutorial/Output
```

=====> Cette commande exécute le programme **WordCount** en utilisant WordCount.jar, en prenant les données d'entrée depuis /WordCountTutorial/Input et en écrivant les résultats dans /WordCountTutorial/Output dans HDFS.

```
rahma@rahma-VirtualBox:~/Desktop/Lab$ hadoop jar WordCount.jar WordCount /WordCountTutorial/Input /WordCountTutorial/Output
2024-11-13 20:46:58,304 INFO client.DefaultNoHARMFatOverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2024-11-13 20:47:00,073 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2024-11-13 20:47:00,186 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/rahma/.staging/job_1731523957744_0001
2024-11-13 20:47:01,324 INFO input.FileInputFormat: Total input files to process : 1
2024-11-13 20:47:01,635 INFO mapreduce.JobSubmitter: number of splits:1
2024-11-13 20:47:02,429 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1731523957744_0001
2024-11-13 20:47:02,429 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-11-13 20:47:03,422 INFO conf.Configuration: resource-types.xml not found
2024-11-13 20:47:03,423 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-11-13 20:47:04,881 INFO impl.YarnClientImpl: Submitted application application_1731523957744_0001
2024-11-13 20:47:05,226 INFO mapreduce.Job: The url to track the job: http://rahma-VirtualBox:8088/proxy/application_1731523957744_0001/
2024-11-13 20:47:05,230 INFO mapreduce.Job: Running job: job_1731523957744_0001
2024-11-13 20:47:32,702 INFO mapreduce.Job: Job job_1731523957744_0001 running in uber mode : false
2024-11-13 20:47:32,706 INFO mapreduce.Job: map 0% reduce 0%
2024-11-13 20:47:47,340 INFO mapreduce.Job: map 100% reduce 0%
2024-11-13 20:47:59,660 INFO mapreduce.Job: map 100% reduce 100%
2024-11-13 20:48:00,717 INFO mapreduce.Job: Job job_1731523957744_0001 completed successfully
2024-11-13 20:48:01,132 INFO mapreduce.Job: Counters: 54
```

```
File System Counters
  FILE: Number of bytes read=359
  FILE: Number of bytes written=527959
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=505
  HDFS: Number of bytes written=245
  HDFS: Number of read operations=8
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=11350
  Total time spent by all reduces in occupied slots (ms)=9466
  Total time spent by all map tasks (ms)=11350
  Total time spent by all reduce tasks (ms)=9466
  Total vcore-milliseconds taken by all map tasks=11350
  Total vcore-milliseconds taken by all reduce tasks=9466
  Total megabyte-milliseconds taken by all map tasks=11622400
  Total megabyte-milliseconds taken by all reduce tasks=9693184
```

**Map-Reduce Framework**

```
Map input records=51
Map output records=54
Map output bytes=598
Map output materialized bytes=359
Input split bytes=120
Combine input records=54
Combine output records=27
Reduce input groups=27
Reduce shuffle bytes=359
Reduce input records=27
Reduce output records=27
Spilled Records=54
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=384
CPU time spent (ms)=2000
Physical memory (bytes) snapshot=329822208
Virtual memory (bytes) snapshot=4972486656
Total committed heap usage (bytes)=170004480
Peak Map Physical memory (bytes)=216195072
Peak Map Virtual memory (bytes)=2482933760
Peak Reduce Physical memory (bytes)=113627136
Peak Reduce Virtual memory (bytes)=2489552896
```

**Shuffle Errors**

```
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
```

**File Input Format Counters**

```
Bytes Read=385
```

**File Output Format Counters**

```
Bytes Written=245
```

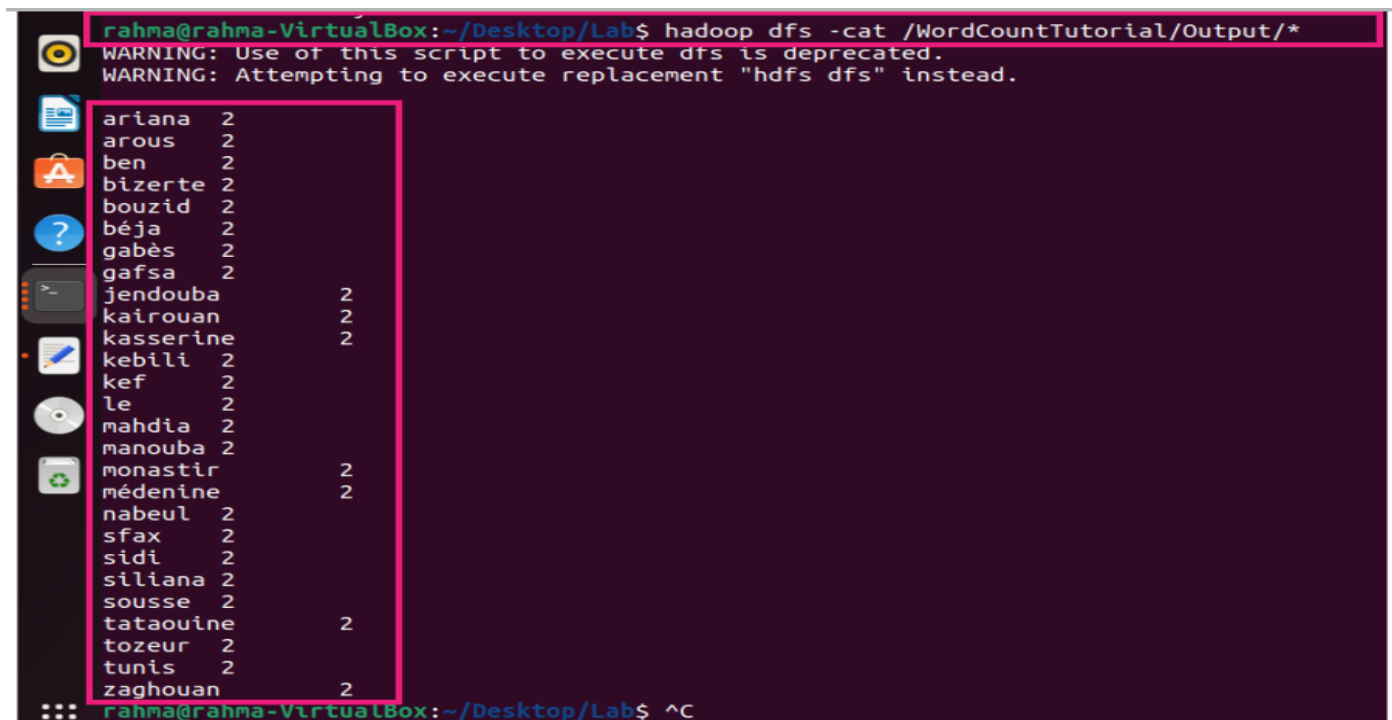
10/ Afficher les résultats de l'exécution à partir du répertoire de sortie HDFS.

```
cd Desktop
```

```
cd Lab
```

```
hadoop dfs -cat /WordCountTutorial/Output/*
```

=====> Cette commande affiche le contenu de tous les fichiers dans le répertoire /WordCountTutorial/Output dans le système de fichiers Hadoop (HDFS).



```
rahma@rahma-VirtualBox:~/Desktop/Lab$ hadoop dfs -cat /WordCountTutorial/Output/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.
ariana 2
arous 2
ben 2
bizerte 2
bouzid 2
béja 2
gabès 2
gafsa 2
jendouba 2
kairouan 2
kasserine 2
kebili 2
kef 2
le 2
mahdia 2
manouba 2
monastir 2
médenine 2
nabeul 2
sfax 2
sidi 2
siliana 2
sousse 2
tataouine 2
tozeur 2
tunis 2
zaghouan 2
rahma@rahma-VirtualBox:~/Desktop/Lab$ ^C
```

#### + Remarque

Pour exécuter cette commande "hadoop dfs -cat /WordCountTutorial/Output/\*", il faut d'abord démarrer les services DFS et YARN dans Hadoop :

```
start-dfs.sh
```

```
start-yarn.sh
```