STEP 1: START HADOOP (Always first)

Open **CMD as Administrator**

cd c:\hadoop-3.3.0\sbin

hdfs namenode -format

start-all.cmd

STEP 2: Create the Java program

Open **Notepad**

Copy–paste your **WordCount.java** code

Save it as **WordCount.java** inside this folder C:\HadoopWork\src

```
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.LongWritable;
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;
public class WordCount {
  public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      context.write(key, new IntWritable(sum));
```

```
        }
    }
    public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Word Count");

        // Critical for Windows

        job.setJar("wordcount.jar");

        job.setJarByClass(WordCount.class);

        job.setMapperClass(Map.class);

        job.setReducerClass(Reduce.class);

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

    }

}
```

STEP 3: Create input file (words.txt)

Go to: C:\HadoopWork\src

Create a new text file named:words.txt

Write this inside the file and save:

```
Hadoop, is cool.

Hadoop is great!

HADOOP is fast.
```

STEP 4: Create input directory in HDFS

```
cd c:\HadoopWork\src
```

Run:

```
hdfs dfs -mkdir /wc_input
```

This creates an input folder in HDFS.

STEP 5: Copy input file to HDFS

hdfs dfs -put words.txt /wc_input

STEP 6: Compile the Java program

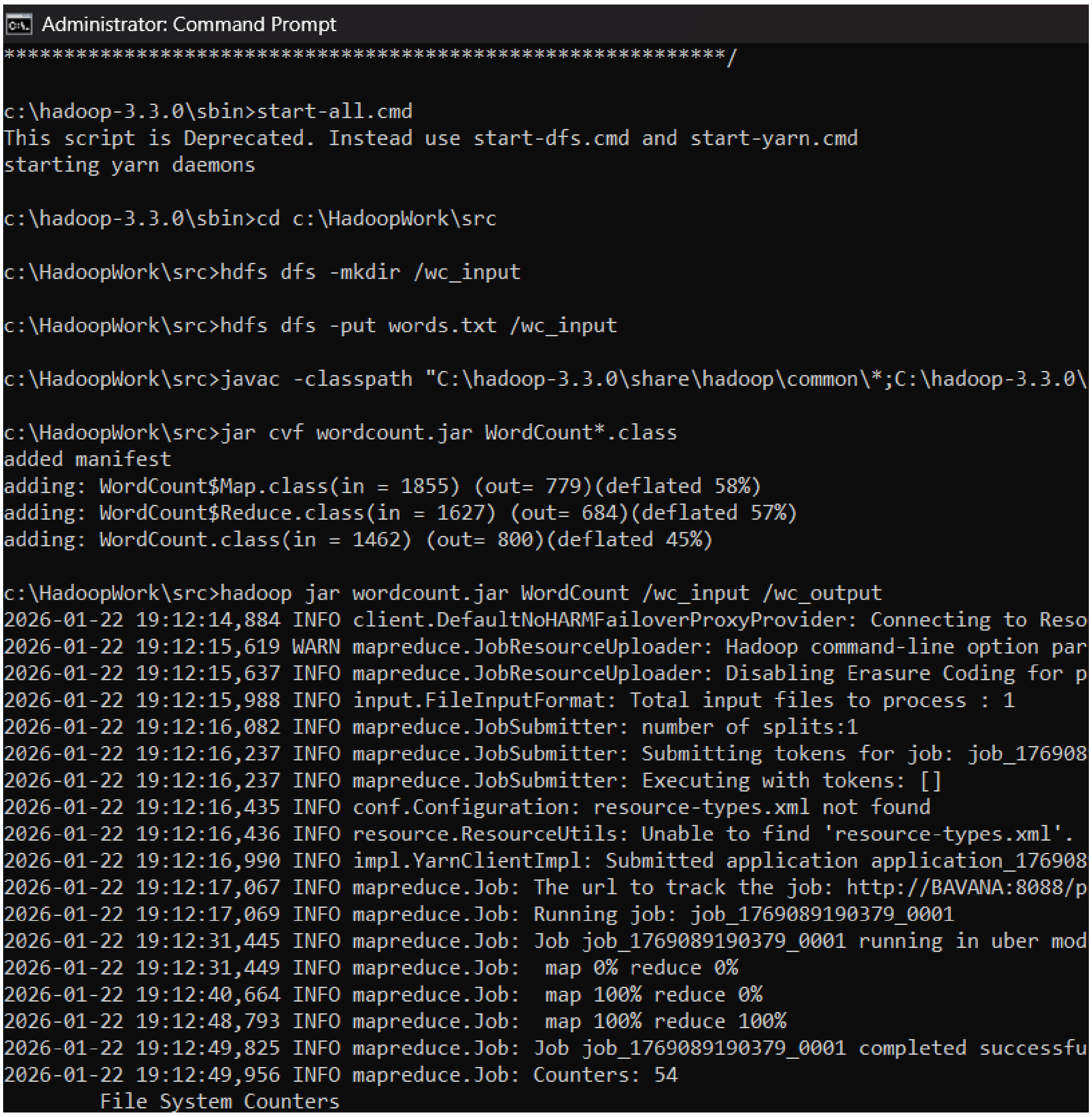
javac -classpath "C:\hadoop-3.3.0\share\hadoop\common\*;C:\hadoop-3.3.0\share\hadoop\mapreduce\*;C:\hadoop-3.3.0\share\hadoop\common\lib\*" WordCount.java

STEP 7: Create JAR file

jar cvf wordcount.jar WordCount*.class

STEP 8: Run WordCount job

hadoop jar wordcount.jar WordCount /wc_input /wc_output

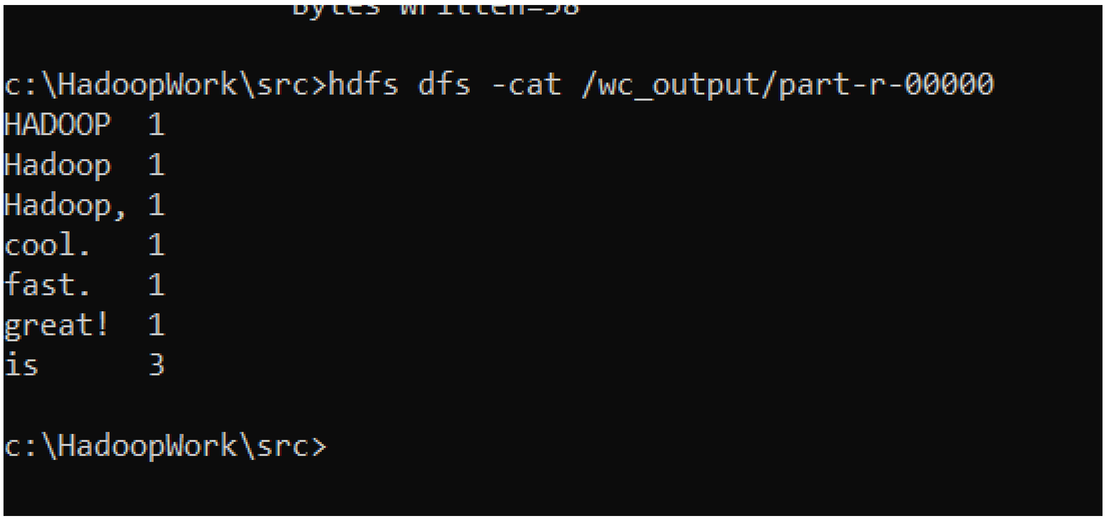


```
Administrator: Command Prompt
************************************************************/

c:\hadoop-3.3.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

c:\hadoop-3.3.0\sbin>cd c:\HadoopWork\src

c:\HadoopWork\src>hdfs dfs -mkdir /wc_input

c:\HadoopWork\src>hdfs dfs -put words.txt /wc_input

c:\HadoopWork\src>javac -classpath "C:\hadoop-3.3.0\share\hadoop\common\*;C:\hadoop-3.3.0\

c:\HadoopWork\src>jar cvf wordcount.jar WordCount*.class
added manifest
adding: WordCount$Map.class(in = 1855) (out= 779)(deflated 58%)
adding: WordCount$Reduce.class(in = 1627) (out= 684)(deflated 57%)
adding: WordCount.class(in = 1462) (out= 800)(deflated 45%)

c:\HadoopWork\src>hadoop jar wordcount.jar WordCount /wc_input /wc_output
2026-01-22 19:12:14,884 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to Reso
2026-01-22 19:12:15,619 WARN mapreduce.JobResourceUploader: Hadoop command-line option par
2026-01-22 19:12:15,637 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for p
2026-01-22 19:12:15,988 INFO input.FileInputFormat: Total input files to process : 1
2026-01-22 19:12:16,082 INFO mapreduce.JobSubmitter: number of splits:1
2026-01-22 19:12:16,237 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_176908
2026-01-22 19:12:16,237 INFO mapreduce.JobSubmitter: Executing with tokens: []
2026-01-22 19:12:16,435 INFO conf.Configuration: resource-types.xml not found
2026-01-22 19:12:16,436 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2026-01-22 19:12:16,990 INFO impl.YarnClientImpl: Submitted application application_176908
2026-01-22 19:12:17,067 INFO mapreduce.Job: The url to track the job: http://BAVANA:8088/p
2026-01-22 19:12:17,069 INFO mapreduce.Job: Running job: job_1769089190379_0001
2026-01-22 19:12:31,445 INFO mapreduce.Job: Job job_1769089190379_0001 running in uber mod
2026-01-22 19:12:31,449 INFO mapreduce.Job:  map 0% reduce 0%
2026-01-22 19:12:40,664 INFO mapreduce.Job:  map 100% reduce 0%
2026-01-22 19:12:48,793 INFO mapreduce.Job:  map 100% reduce 100%
2026-01-22 19:12:49,825 INFO mapreduce.Job: Job job_1769089190379_0001 completed successfu
2026-01-22 19:12:49,956 INFO mapreduce.Job: Counters: 54
        File System Counters
```

STEP 9: View output

hdfs dfs -cat /wc_output/part-r-00000

EXPECTED OUTPUT



```
                Bytes Written=58

c:\HadoopWork\src>hdfs dfs -cat /wc_output/part-r-00000
HADOOP  1
Hadoop  1
Hadoop, 1
cool.   1
fast.   1
great!  1
is      3

c:\HadoopWork\src>
```