Hadoop Word Cound using Docker

Open the Docker terminal and run the following commands:

1. Start the container:

```
C:\Users\asus> docker run -p 9870:9870 -p 8088:8088 -it
--name=testHadoop macio232/hadoop-pseudo-distributed-mode
```

Or

```
C:\Users\asus> docker container start -i testHadoop
```

2. Navigate to the Hadoop Data Directory

```
root@2a78d9e418fb:/# cd /home/hadoop/data
```

3. Confirm You Are in the Correct Directory

```
root@2a78d9e418fb:/home/hadoop/data# pwd
```

4. Create a .java file for word count

```
root@2a78d9e418fb:/home/hadoop/data# vi WordCount.java
```

Copy and paste the following java code to the WordCount.java file

```
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;

import java.io.IOException;
import java.util.StringTokenizer;

public class WordCount {

    public static class TokenizerMapper
         extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        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);
                }
            }
        }

        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);
            }
        }

        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);
            job.setReducerClass(IntSumReducer.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);
        }
      }
```

Summary of Commands:

i: Enter insert mode.
Esc: Exit insert mode (back to normal mode).
:wq: Save and quit.
:q!: Quit without saving

5. Compile the Java Code:

```
/home/hadoop/data# javac -classpath `hadoop classpath` -d . WordCount.java
```

6. Package the compiled classes into a JAR file:

```
/home/hadoop/data# jar cf wordcount.jar WordCount*.class
```

7. Create a directory for the input data inside

```
/home/hadoop/data# mkdir input
```

8. Create a sample text file:

```
/home/hadoop/data# echo "Hello Hadoop Hello Docker" > input/file01.txt
```

9. Put the input data into HDFS

```
/home/hadoop/data# hdfs dfs -mkdir -p /user/hadoop/input
/home/hadoop/data# hdfs dfs -put ./input/* /user/hadoop/input/
```

10. Run the Hadoop job using:

```
/home/hadoop/data# hadoop jar wordcount.jar WordCount /user/hadoop/input
/user/hadoop/output
```

11. After the job completes, view the results:

```
/home/hadoop/data# hdfs dfs -cat /user/hadoop/output/part-r-00000
```

1.