To start Hadoop Commands:

1. su - hadoop
2. sudo service ssh start
3. ssh localhost
4. start-all.sh
5. hdfs dfs -mkdir word_count_ex #created directory in hadoop cluster

Link:
https://medium.com/javarevisited/word-count-example-using-hadoop-and-java-8ef3d665e331

Step 1: Create Word Count Mapper file

Create WC_Mapper.java file

```
 GNU nano 4.8                                                    WC_Mapper.java

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements 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,OutputCollector<Text,IntWritable> output,
                    Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer  tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()){
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }

}
```

Step 2: Create Word Count Reducer file

Create WC_Reducer.java file

```
 GNU nano 4.8                                  WC_Reducer.java                                  Mo
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WC_Reducer  extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
                       Reporter reporter) throws IOException {
        int sum=0;
        while (values.hasNext()) {
            sum+=values.next().get();
        }
        output.collect(key,new IntWritable(sum));
    }
}
```

Step 3: Create Word Count Runner file

Create WC_Runner.java file

```
 GNU nano 4.8                                  WC_Runner.java
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
    public static void main(String[] args) throws IOException{
        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

Step 4: Create txt file and push it into hdfs

Create input.txt file

Using the same input.txt from python wordcount exercise.

- cd ..
- cp word_count/file1.txt word_count_java

Step 5: Now create a folder called input on hdfs using the following commands:

- `hadoop fs -mkdir /word_count_java`
- `hadoop fs -mkdir /word_count_java/input`

Step 6: Now to push the created input.txt file you can type the following command : `hadoop fs -put word_count_java/file1.txt /word_count_java/input`

Step 7: Compile the Java Classes

- `javac -classpath `hadoop classpath` -d wordcount_classes WC_Mapper.java WC_Reducer.java WC_Runner.java`

```
hadoop@EDITH:~$ cd word_count_java
hadoop@EDITH:~/word_count_java$ javac -classpath `hadoop classpath` -d wordcount_classes WC_Mapper.java WC_Reducer.java WC_Runner.jav
a
```

- `jar -cvf wordcount.jar -C wordcount_classes/ .`

```
hadoop@EDITH:~/word_count_java$ jar -cvf wordcount.jar -C wordcount_classes/ .
added manifest
adding: WC_Mapper.class(in = 1859) (out= 757)(deflated 59%)
adding: WC_Reducer.class(in = 1531) (out= 605)(deflated 60%)
adding: WC_Runner.class(in = 1440) (out= 713)(deflated 50%)
```

Step 8: Run the project

`hadoop jar wordcount.jar WC_Runner /word_count_java/input/file1.txt /word_count_java/output`

```
hadoop@EDITH:~/word_count_java$ hadoop jar wordcount.jar WC_Runner /word_count_java/input/file1.txt /word_count_java/output
2024-10-01 21:17:22,878 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-01 21:17:22,971 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-01 21:17:23,203 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inte
rface and execute your application with ToolRunner to remedy this.
2024-10-01 21:17:23,224 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.stagi
ng/job_1727761516095_0004
2024-10-01 21:17:23,427 INFO mapred.FileInputFormat: Total input files to process : 1
2024-10-01 21:17:23,516 INFO mapreduce.JobSubmitter: number of splits:2
2024-10-01 21:17:23,650 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1727761516095_0004
2024-10-01 21:17:23,650 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-01 21:17:23,765 INFO conf.Configuration: resource-types.xml not found
2024-10-01 21:17:23,766 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-10-01 21:17:23,816 INFO impl.YarnClientImpl: Submitted application application_1727761516095_0004
2024-10-01 21:17:23,864 INFO mapreduce.Job: The url to track the job: http://EDITH.localdomain:8088/proxy/application_1727761516095_0
004/
2024-10-01 21:17:23,865 INFO mapreduce.Job: Running job: job_1727761516095_0004
2024-10-01 21:17:27,943 INFO mapreduce.Job: Job job_1727761516095_0004 running in uber mode : false
2024-10-01 21:17:27,945 INFO mapreduce.Job:  map 0% reduce 0%
2024-10-01 21:17:33,015 INFO mapreduce.Job:  map 100% reduce 0%
2024-10-01 21:17:37,050 INFO mapreduce.Job:  map 100% reduce 100%
2024-10-01 21:17:38,076 INFO mapreduce.Job: Job job_1727761516095_0004 completed successfully
2024-10-01 21:17:38,133 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=242
                FILE: Number of bytes written=830199
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=598
                HDFS: Number of bytes written=98
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
```

```
                Map output materialized bytes=248
                Input split bytes=206
                Combine input records=36
                Combine output records=18
                Reduce input groups=11
                Reduce shuffle bytes=248
                Reduce input records=18
                Reduce output records=11
                Spilled Records=36
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=51
                CPU time spent (ms)=1120
                Physical memory (bytes) snapshot=747839488
                Virtual memory (bytes) snapshot=9129893888
                Total committed heap usage (bytes)=635437056
                Peak Map Physical memory (bytes)=265916416
                Peak Map Virtual memory (bytes)=2745475072
                Peak Reduce Physical memory (bytes)=223100928
                Peak Reduce Virtual memory (bytes)=3643244544
        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=392
        File Output Format Counters
                Bytes Written=98
```

`Worked!`

Step 9: See the output

`hdfs dfs -cat /word_count_java/output/part-00000`

```
hadoop@EDITH:~/word_count_java$ hdfs dfs -cat /word_count_java/output/part-00000
Hi      1
bagiya  9
cynddia 4
ganesh  2
harini  4
hello   1
krishna 2
saanji  3
sholini 4
shrumo  2
suruthi 4
```