Hadoop WordCount Complete Guide

==========

Step 1: Start HDFS Services

---

start-dfs.sh

Step 2: Start YARN Services

---

start-yarn.sh

Step 3: WordCount Java Program

```
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 org.apache.hadoop.util.GenericOptionsParser;

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 tokenizer = new StringTokenizer(value.toString());
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.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();

    GenericOptionsParser parser = new GenericOptionsParser(conf, args);
```

```
        String[] remainingArgs = parser.getRemainingArgs();

        if (remainingArgs.length != 2) {
            System.err.println("Usage: wordcount <input> <output>");
            System.exit(2);
        }

        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(remainingArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(remainingArgs[1]));

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

Step 4: Compile the Program

```
javac -classpath $(hadoop classpath) -d WordCountClasses WordCount.java
jar -cvf wordcount.jar -C WordCountClasses/ .
```

Step 5: Upload Input to HDFS

```
hdfs dfs -mkdir /input
hdfs dfs -put /local/path/to/input_file /input
```

Step 6: Run the MapReduce Job

---

hadoop jar wordcount.jar WordCount /input /output

Step 7: View Output

---

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

Step 8: Stop Hadoop

---

stop-yarn.sh

stop-dfs.sh