# **Running a java program in Hadoop**

1. Navigate to a Working Directory

| cd ~ mkdir hadoop\_projects cd hadoop\_projects |
| --- |

1. Create the Java File:

| nano WordCount.java |
| --- |

1. Create a New File (e.g., input.txt):

| nano input.txt |
| --- |

1. Paste the Java Code into WordCount.java and save the file:

| import java.io.IOException; 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;  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 {  String[] tokens = value.toString().split("\\s+");  for (String token : tokens) {  word.set(token);  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);  } } |
| --- |

1. Compile the Java Program:

| javac -classpath $(hadoop classpath) -d . WordCount.java |
| --- |

1. Create a JAR File

| jar cf wordcount.jar WordCount\*.class |
| --- |

1. Start Hadoop Services

| start-dfs.sh start-yarn.sh |
| --- |

1. Create an Input Directory in HDFS:

| hdfs dfs -mkdir -p /user/1910776142/input |
| --- |

1. Upload Your Input File to HDFS:

| hdfs dfs -put input.txt /user/1910776142/input |
| --- |

1. Run the Hadoop Job:

| hadoop jar wordcount.jar WordCount /user/1910776142/input /user/1910776142/output |
| --- |

1. Check the Output:

| hdfs dfs -cat /user/1910776142/output/part-r-00000 |
| --- |

**Note : In case of mapred-site.xml configuration error follow the steps below**

1. Edit mapred-site.xml:

| nano $HADOOP\_HOME/etc/hadoop/mapred-site.xml |
| --- |

1. Add the following properties if they are not already present, and ensure they are pointing to the correct Hadoop distribution directory.

| <configuration>  <property>  <name>yarn.app.mapreduce.am.env</name>  <value>HADOOP\_MAPRED\_HOME=/path/to/your/hadoop/directory</value>  </property>  <property>  <name>mapreduce.map.env</name>  <value>HADOOP\_MAPRED\_HOME=/path/to/your/hadoop/directory</value>  </property>  <property>  <name>mapreduce.reduce.env</name>  <value>HADOOP\_MAPRED\_HOME=/path/to/your/hadoop/directory</value>  </property> </configuration> |
| --- |

1. Check Environment Variables:

| echo $HADOOP\_HOME echo $HADOOP\_MAPRED\_HOME |
| --- |

1. If they are not set, add the following lines to your shell profile and reload it.

| export HADOOP\_HOME=/path/to/your/hadoop/directory export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME source ~/.bashrc |
| --- |

1. Restart Hadoop Services:

| stop-dfs.sh stop-yarn.sh start-dfs.sh start-yarn.sh |
| --- |

1. Run the Hadoop Job Again:

# **Hadoop Word Cound using Docker**

1. Start the container:

docker start hadoop

1. Get inside the container:  
   docker exec -it hadoop /bin/bash
2. Start hadoop services:  
   start-all.sh
3. Navigate to the Data Directory: Since you mounted the current directory to /home/hadoop/data (while installation), navigate there

cd /home/hadoop/data

1. Create the WordCount.java File: Use a text editor like vi to create the Java file:  
   vi WordCount.java

Code:

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

1. Compile the Java Code:  
   javac -classpath `hadoop classpath` -d . WordCount.java
2. Package the compiled classes into a JAR file:  
   jar cf wordcount.jar WordCount\*.class
3. Create a directory for the input data inside /home/hadoop/data:  
   mkdir input
4. Create a sample text file:  
   echo "Hello Hadoop Hello Docker" > input/file01.txt
5. Put the input data into HDFS  
   hdfs dfs -mkdir -p /user/hadoop/input   
   hdfs dfs -put ./input/\* /user/hadoop/input/
6. Run the Hadoop job using:  
   hadoop jar wordcount.jar WordCount /user/hadoop/input /user/hadoop/output
7. After the job completes, view the results:  
   hdfs dfs -cat /user/hadoop/output/part-r-00000