BDC Practical No. 03

To implement a word count application using the MapReduce API.

SOURCE CODE

Dependencies :

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>3.3.3</version>
</dependency>

<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-core</artifactId>
    <version>3.3.3</version>
</dependency>
```

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);
        }
    }
}
```

WC_Reducer.java

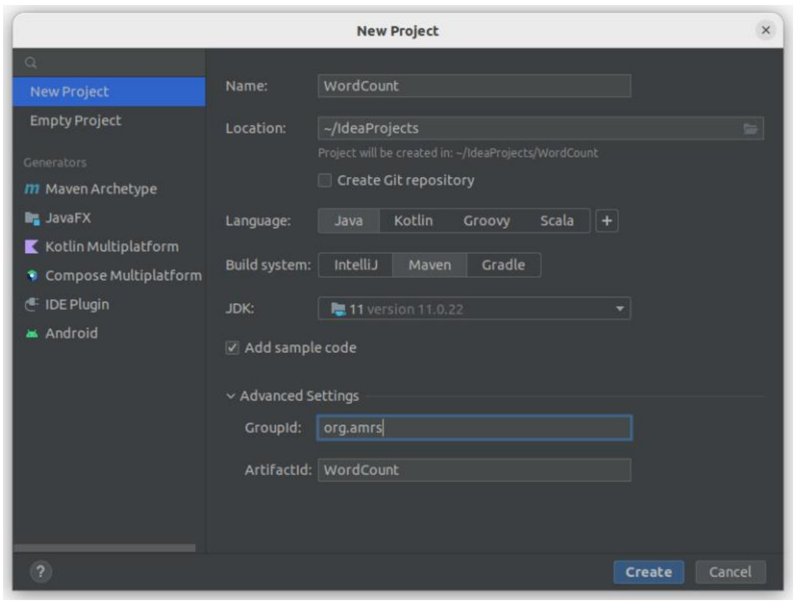
```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
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));
    }
}
```

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);
    }
}
```

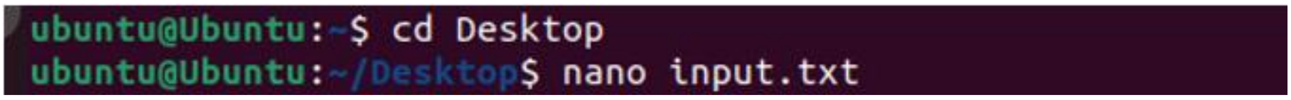
---

1) Setting up Java Environment.



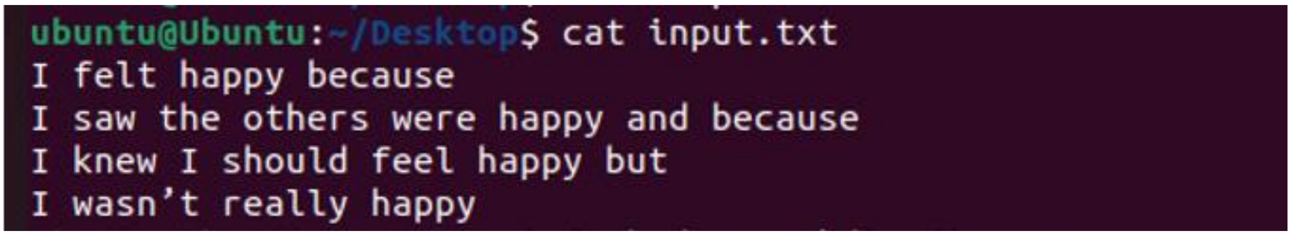
Name: WordCount
Location: ~/IdeaProjects
Project will be created in: ~/IdeaProjects/WordCount
Create Git repository
Language: Java Kotlin Groovy Scala
Build system: IntelliJ Maven Gradle
JDK: 11 version 11.0.22
Add sample code
Advanced Settings
GroupId: org.amrs
ArtifactId: WordCount

2) **Creating Input data file to the ubuntu desktop.**



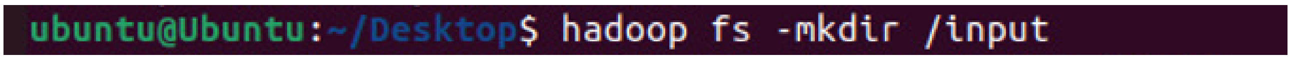
```
ubuntu@Ubuntu:~$ cd Desktop
ubuntu@Ubuntu:~/Desktop$ nano input.txt
```

3) **Displaying contents of the input text file.**



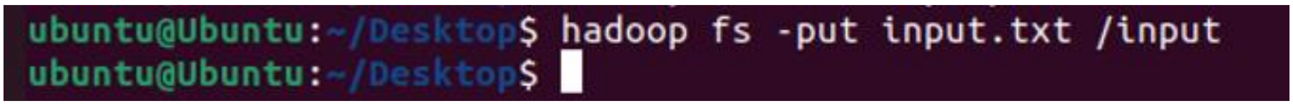
```
ubuntu@Ubuntu:~/Desktop$ cat input.txt
I felt happy because
I saw the others were happy and because
I knew I should feel happy but
I wasn’t really happy
```

4) **Creating Input directory to the hadoop path.**

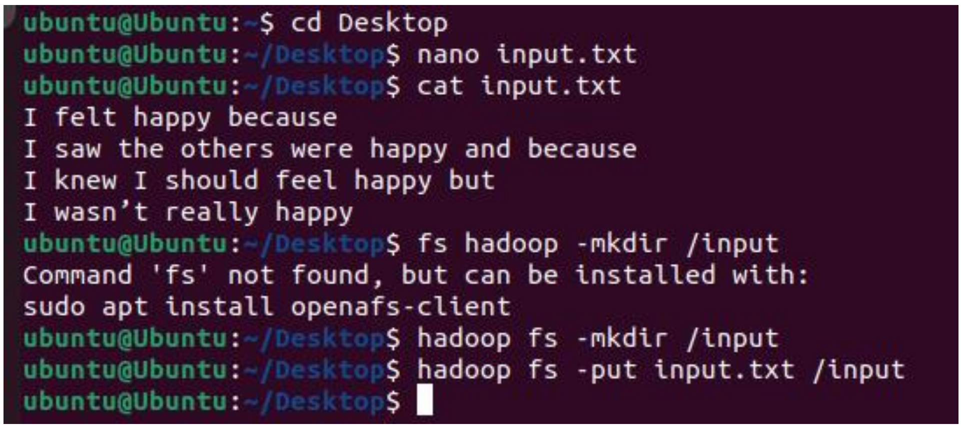


```
ubuntu@Ubuntu:~/Desktop$ hadoop fs -mkdir /input
```

5) **Copy & paste input.txt file to the input path in hadoop from desktop.**

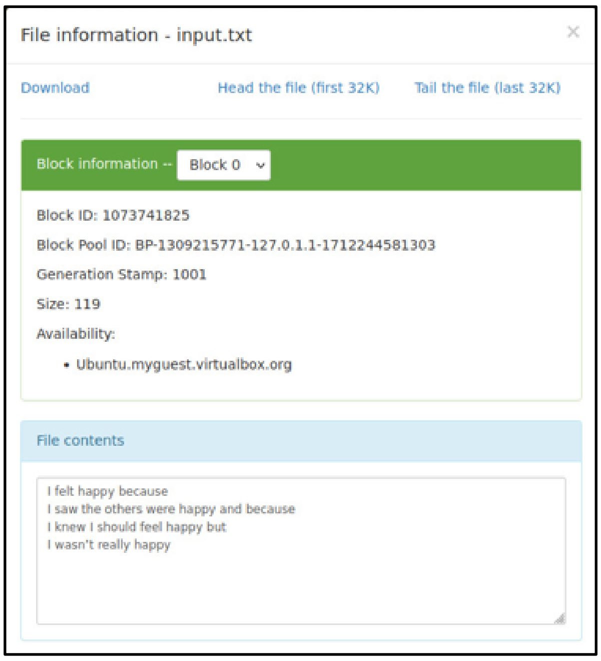


```
ubuntu@Ubuntu:~/Desktop$ hadoop fs -put input.txt /input
ubuntu@Ubuntu:~/Desktop$ 
```



```
ubuntu@Ubuntu:~$ cd Desktop
ubuntu@Ubuntu:~/Desktop$ nano input.txt
ubuntu@Ubuntu:~/Desktop$ cat input.txt
I felt happy because
I saw the others were happy and because
I knew I should feel happy but
I wasn’t really happy
ubuntu@Ubuntu:~/Desktop$ fs hadoop -mkdir /input
Command 'fs' not found, but can be installed with:
sudo apt install openafs-client
ubuntu@Ubuntu:~/Desktop$ hadoop fs -mkdir /input
ubuntu@Ubuntu:~/Desktop$ hadoop fs -put input.txt /input
ubuntu@Ubuntu:~/Desktop$ 
```

Before Word Count - Input.txt



File information - input.txt

Download   Head the file (first 32K)   Tail the file (last 32K)

Block information -- Block 0

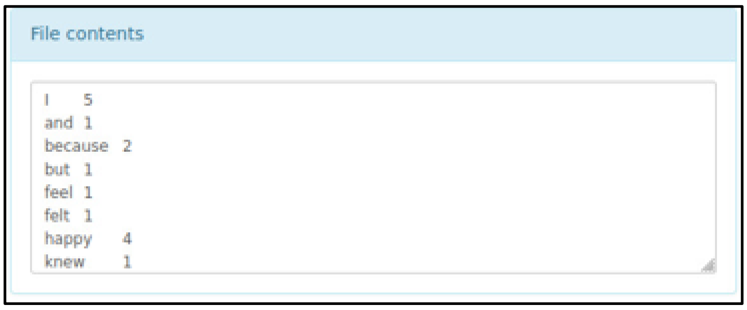
Block ID: 1073741825
Block Pool ID: BP-1309215771-127.0.1.1-1712244581303
Generation Stamp: 1001
Size: 119
Availability:

- Ubuntu.myguest.virtualbox.org

File contents

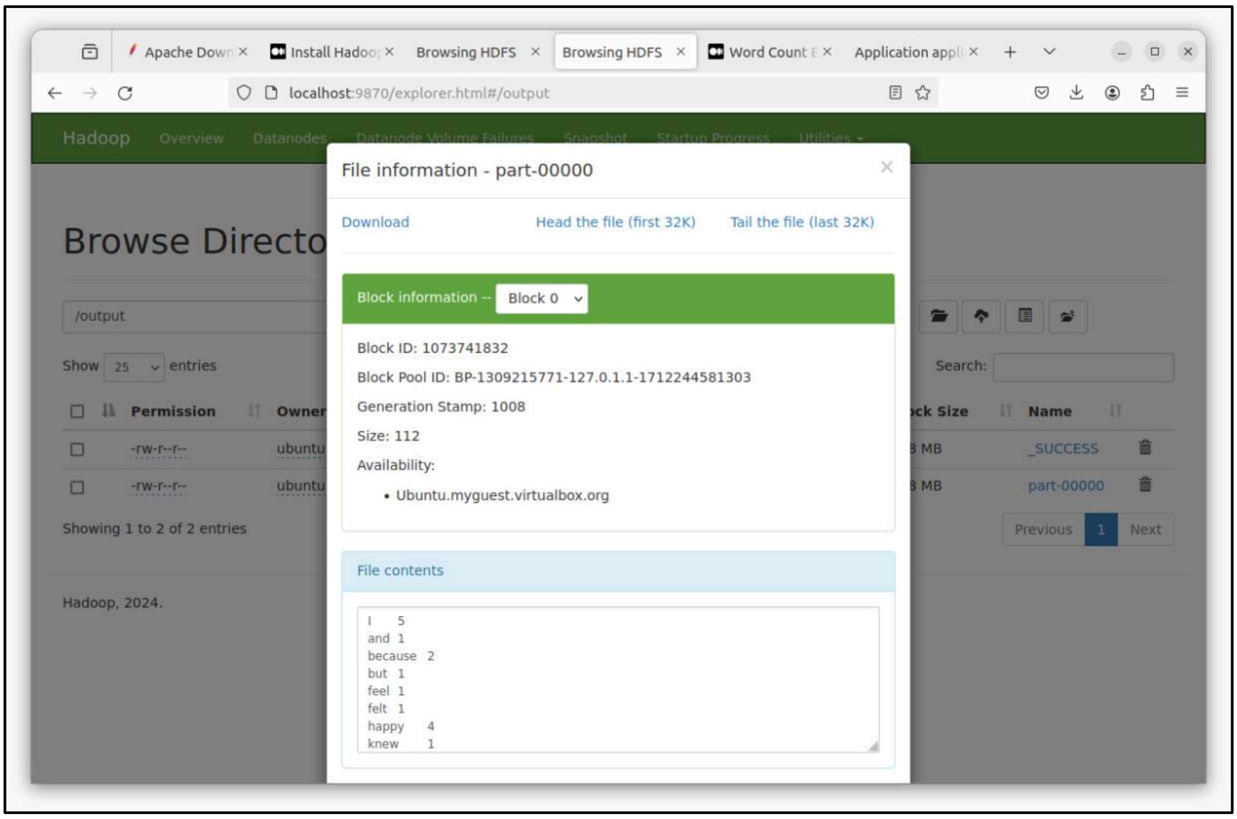
```
I felt happy because
I saw the others were happy and because
I knew I should feel happy but
I wasn't really happy
```

After Word Count - Input.txt



File contents

```
I     5
and  1
because  2
but  1
feel  1
felt  1
happy    4
knew     1
```



localhost:9870/explorer.html#/output

File information - part-00000

Download   Head the file (first 32K)   Tail the file (last 32K)

Block information -- Block 0

Block ID: 1073741832
Block Pool ID: BP-1309215771-127.0.1.1-1712244581303
Generation Stamp: 1008
Size: 112
Availability:

- Ubuntu.myguest.virtualbox.org

File contents

```
I     5
and  1
because  2
but  1
feel  1
felt  1
happy    4
knew     1
```

Permission	Owner	Block Size	Name
-rw-r--r--	ubuntu	MB	_SUCCESS
-rw-r--r--	ubuntu	MB	part-00000

Showing 1 to 2 of 2 entries

Hadoop, 2024.