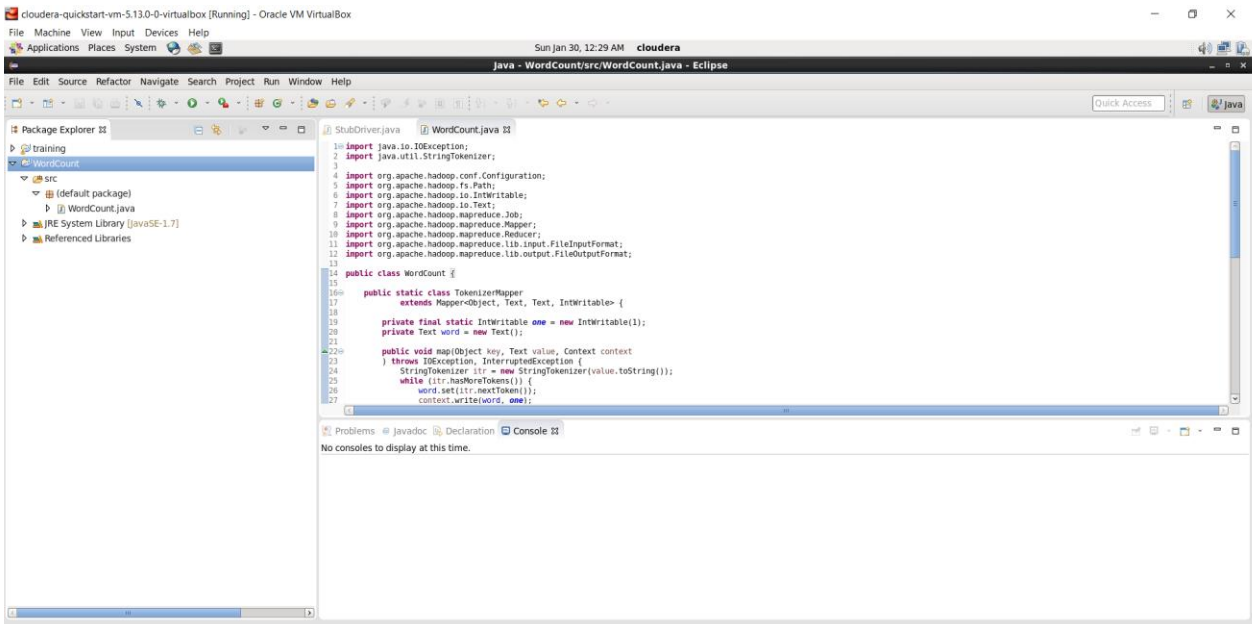
CSCE 5300 INTRODUCTION TO BIG DATA AND DATA SCIENCE

NAME : Eswara Reddy Thimmapuram

Use case Mapper Reduce word count process

- Firstly, the input is given in a specified size and then the input is split into different parts. Then the next job is to convert input block into string. So the mapper takes starting address and ending address as input.
- Then the mapper converts input into string and tokenizes into words. The mapper will then append 1 to each word for first mapper output and so on. The output of mapper would be key-value pairs. After mapper process the next jobs are sorting and shuffling.
- In next phase, all words come to their respective places in sorting phase and in shuffling phase all similar words go to similar reducer. Then the reducer will sum up values by using keys. The output of reducer will be the final result which is the sum of words that are present in the input file.

WordCount Program

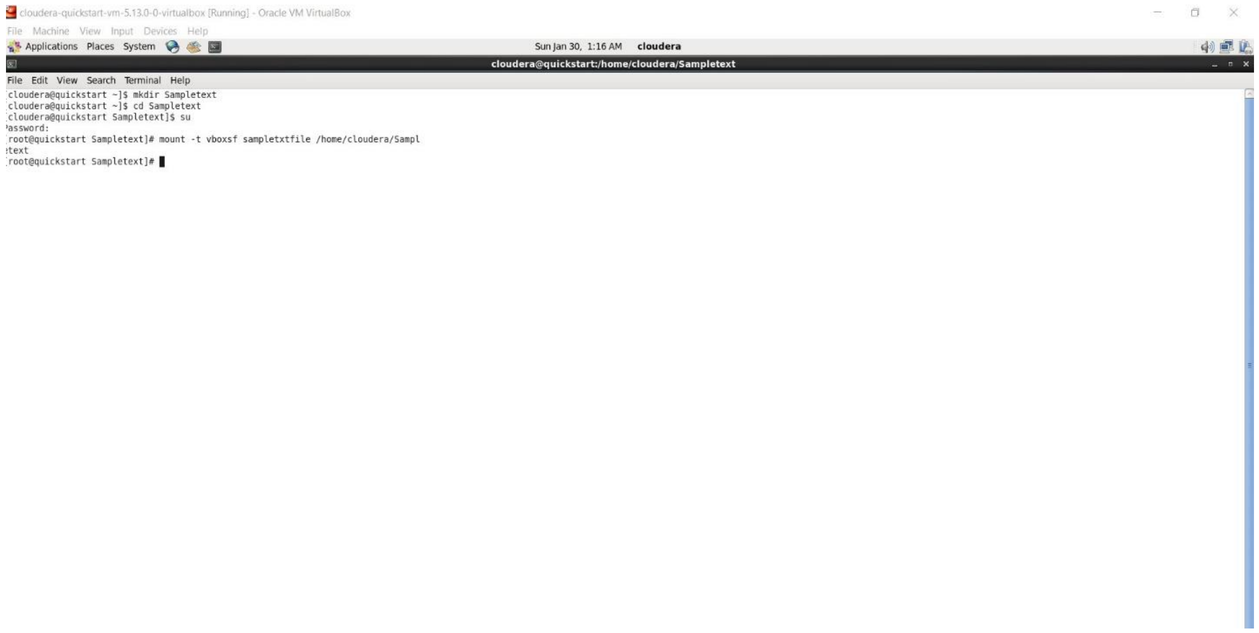

cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sun Jan 30, 12:29 AM cloudera
Java - WordCount/src/WordCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
training
WordCount
src
(default package)
WordCount.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
StubDriver.java
WordCount.java
Problems Javadoc Declaration Console

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

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 {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
```

```
No consoles to display at this time.
```

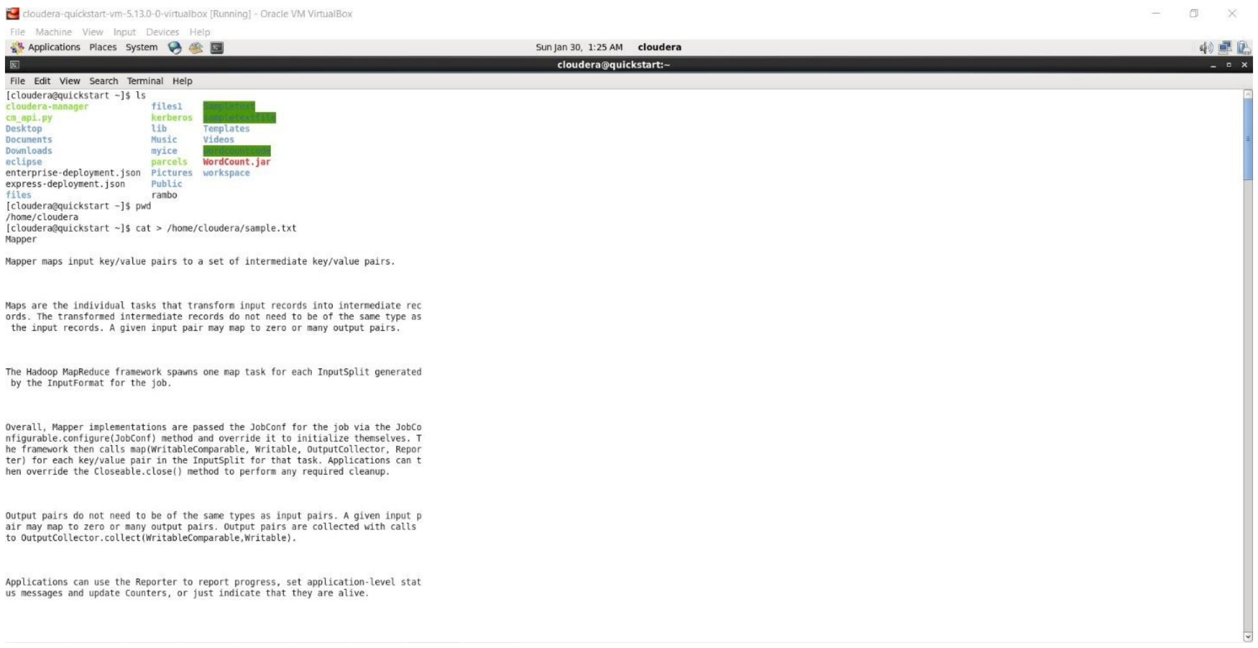
- Firstly we create new java project WordCount and import all header files and will add all required external jar files. Then we create new class named WordCount .

- Then we extend the mapper class by creating TokenizerMapper and it takes arguments like key-in,value-in,key-out and value-out. Then we create word variable to store word and one variable to store value.
- Then we create map function which takes arguments like key-in as key, textual content as value variable and context to emit result to next stage.
- Then we StringTokenizer which divides string into tokens and store each of them in itr variable.
- Then we define reducer class as IntSumReducer which takes four inputs. Then we create result variable to store how many number of times the key is appearing.
- Then we write reduce function which takes word as key and value as value and if the word repeats then it will be added to the sum value.
- Finally the reducer gives the final output of the program which is the total sum of words that are present in input file.


cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sun Jan 30, 1:16 AM cloudera
cloudera@quickstart:/home/cloudera/Sampletext
File Edit View Search Terminal Help

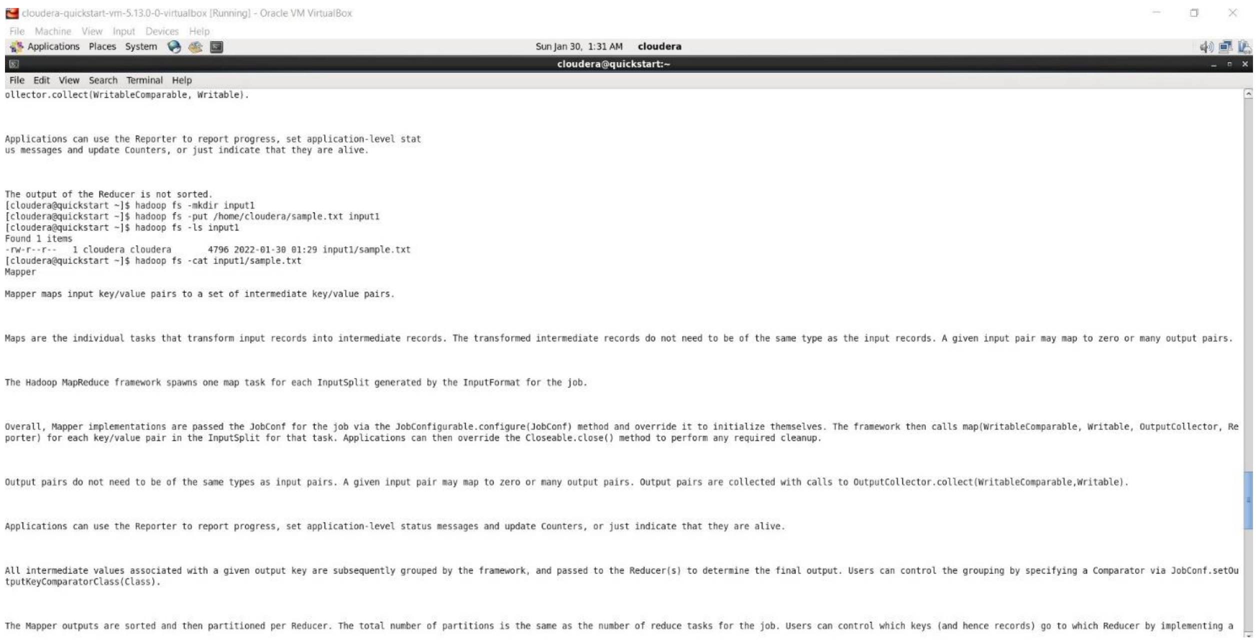
```
cloudera@quickstart ~]$ mkdir Sampletext
cloudera@quickstart ~]$ cd Sampletext
cloudera@quickstart Sampletext]$ su
'assword:
root@quickstart Sampletext]# mount -t vboxsf sampletxtfile /home/cloudera/Sampl
:text
root@quickstart Sampletext]# █
```

- Then we create Sampletext file to store the input we have to give to the program.



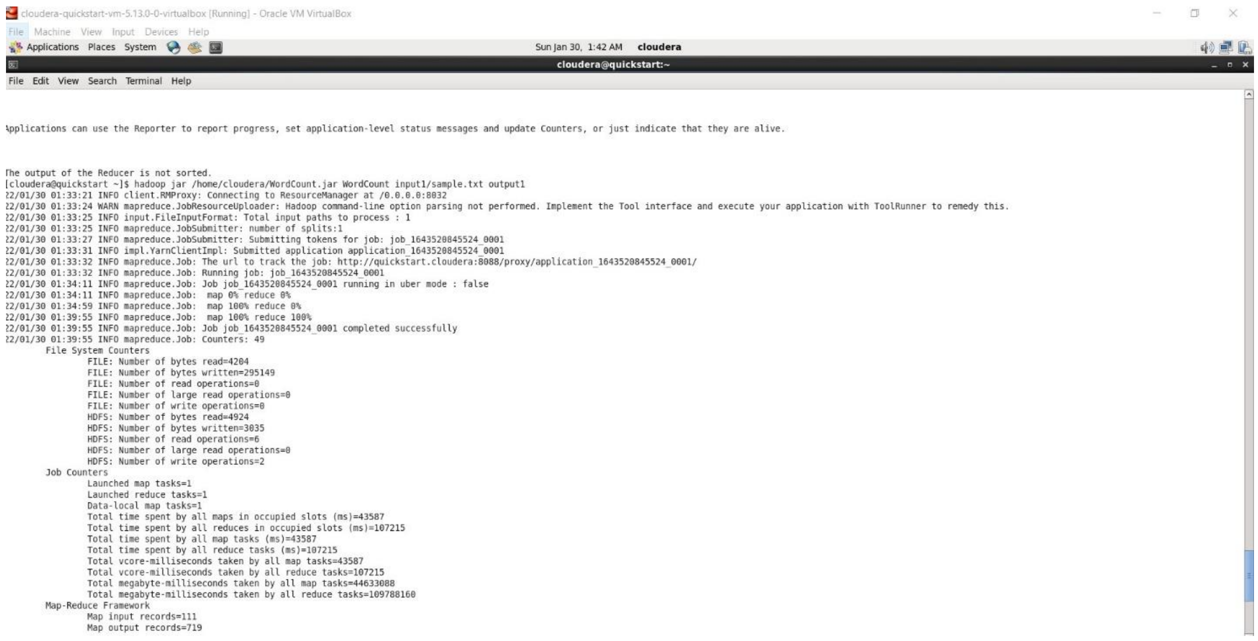
```
[cloudera@quickstart ~]$ ls
cloudera-manager            files1    Sampletext
cm_api.py                   kerberos  Sampletextfile
Desktop                     lib       Templates
Documents                   Music     Videos
Downloads                   myice     wordcountcode
eclipse                     parcels   WordCount.jar
enterprise-deployment.json  Pictures  workspace
express-deployment.json     Public
files                       rambo
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ cat > /home/cloudera/sample.txt
Mapper

Mapper maps input key/value pairs to a set of intermediate key/value pairs.



Maps are the individual tasks that transform input records into intermediate rec
ords. The transformed intermediate records do not need to be of the same type as
 the input records. A given input pair may map to zero or many output pairs.



The Hadoop MapReduce framework spawns one map task for each InputSplit generated
 by the InputFormat for the job.



Overall, Mapper implementations are passed the JobConf for the job via the JobCo
nfigurable.configure(JobConf) method and override it to initialize themselves. T
he framework then calls map(WritableComparable, Writable, OutputCollector, Repor
ter) for each key/value pair in the InputSplit for that task. Applications can t
hen override the Closeable.close() method to perform any required cleanup.



Output pairs do not need to be of the same types as input pairs. A given input p
air may map to zero or many output pairs. Output pairs are collected with calls
to OutputCollector.collect(WritableComparable,Writable).



Applications can use the Reporter to report progress, set application-level stat
us messages and update Counters, or just indicate that they are alive.
```

- Then we use cat command to store the input into sampletext file.



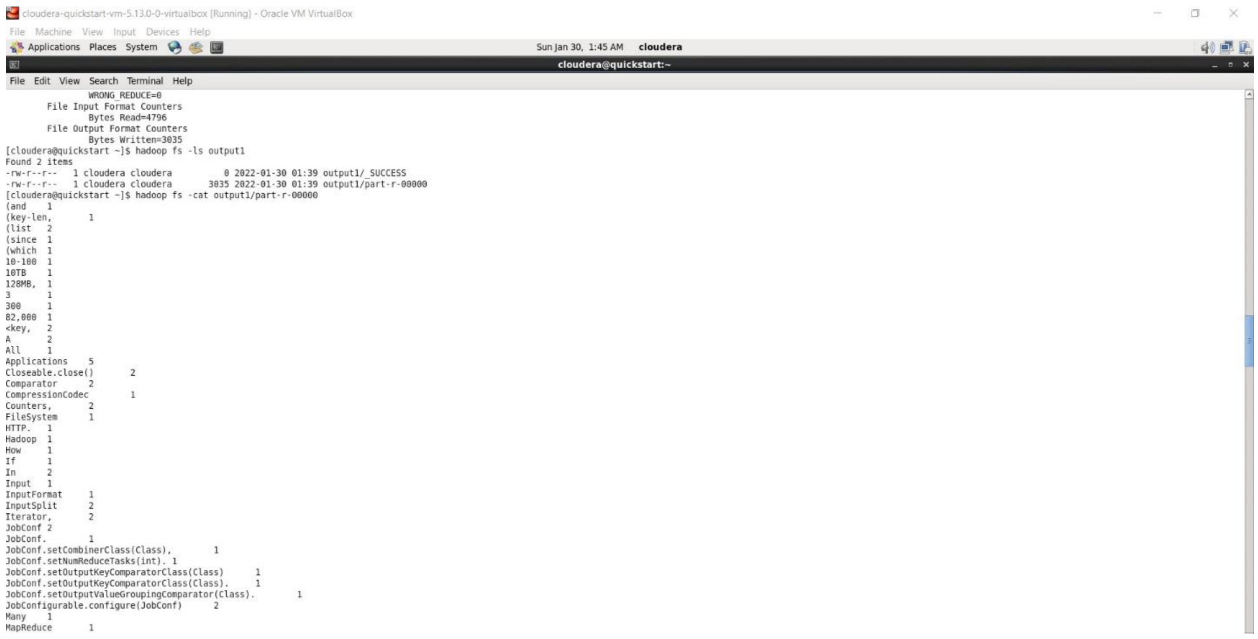
```
ollector.collect(WritableComparable, Writable).



Applications can use the Reporter to report progress, set application-level stat
us messages and update Counters, or just indicate that they are alive.



The output of the Reducer is not sorted.
[cloudera@quickstart ~]$ hadoop fs -mkdir input1
[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/sample.txt input1
[cloudera@quickstart ~]$ hadoop fs -ls input1
Found 1 items
-rw-r--r--   1 cloudera cloudera       4796 2022-01-30 01:29 input1/sample.txt
[cloudera@quickstart ~]$ hadoop fs -cat input1/sample.txt
Mapper

Mapper maps input key/value pairs to a set of intermediate key/value pairs.



Maps are the individual tasks that transform input records into intermediate records. The transformed intermediate records do not need to be of the same type as the input records. A given input pair may map to zero or many output pairs.



The Hadoop MapReduce framework spawns one map task for each InputSplit generated by the InputFormat for the job.



Overall, Mapper implementations are passed the JobConf for the job via the JobConfigurable.configure(JobConf) method and override it to initialize themselves. The framework then calls map(WritableComparable, Writable, OutputCollector, Re
porter) for each key/value pair in the InputSplit for that task. Applications can then override the Closeable.close() method to perform any required cleanup.



Output pairs do not need to be of the same types as input pairs. A given input pair may map to zero or many output pairs. Output pairs are collected with calls to OutputCollector.collect(WritableComparable,Writable).



Applications can use the Reporter to report progress, set application-level status messages and update Counters, or just indicate that they are alive.



All intermediate values associated with a given output key are subsequently grouped by the framework, and passed to the Reducer(s) to determine the final output. Users can control the grouping by specifying a Comparator via JobConf.setOu
tputKeyComparatorClass(Class).



The Mapper outputs are sorted and then partitioned per Reducer. The total number of partitions is the same as the number of reduce tasks for the job. Users can control which keys (and hence records) go to which Reducer by implementing a
```

- Then we create input1 file and use put command to copy file from local system to Hadoop system.



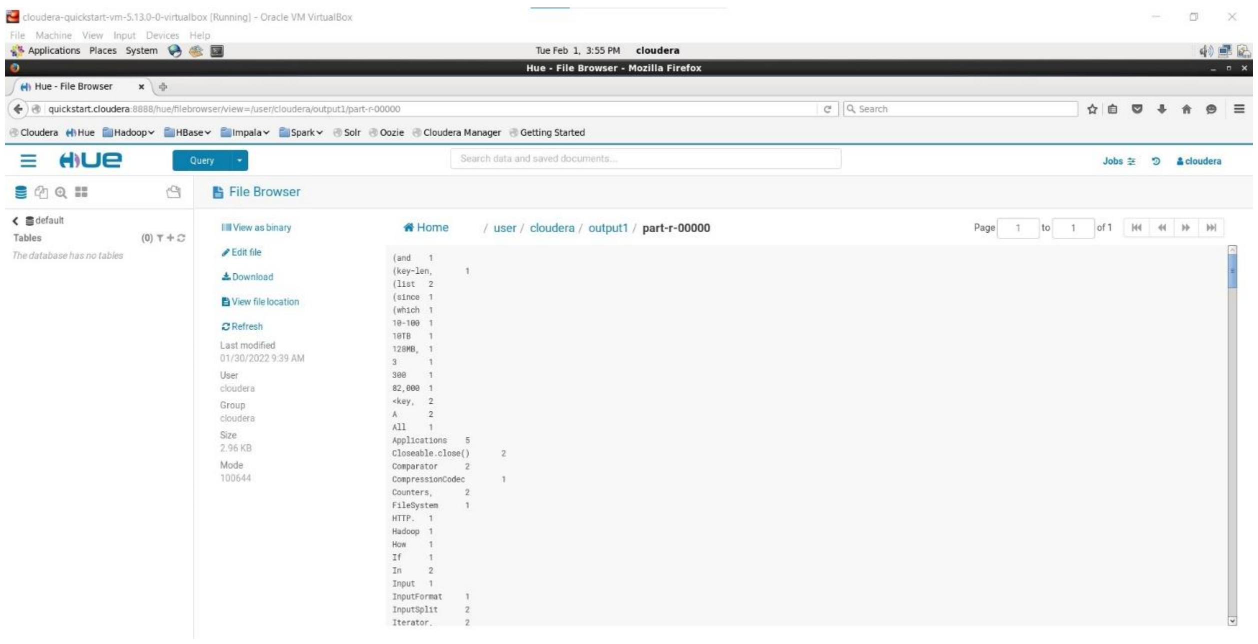
```
Applications can use the Reporter to report progress, set application-level status messages and update Counters, or just indicate that they are alive.


The output of the Reducer is not sorted.
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount input1/sample.txt output1
22/01/30 01:33:21 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/01/30 01:33:24 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/01/30 01:33:25 INFO input.FileInputFormat: Total input paths to process : 1
22/01/30 01:33:25 INFO mapreduce.JobSubmitter: number of splits:1
22/01/30 01:33:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1643520845524_0001
22/01/30 01:33:31 INFO impl.YarnClientImpl: Submitted application application_1643520845524_0001
22/01/30 01:33:32 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1643520845524_0001/
22/01/30 01:33:32 INFO mapreduce.Job: Running job: job_1643520845524_0001
22/01/30 01:34:11 INFO mapreduce.Job: Job job_1643520845524_0001 running in uber mode : false
22/01/30 01:34:11 INFO mapreduce.Job:  map 0% reduce 0%
22/01/30 01:34:59 INFO mapreduce.Job:  map 100% reduce 0%
22/01/30 01:39:55 INFO mapreduce.Job:  map 100% reduce 100%
22/01/30 01:39:55 INFO mapreduce.Job: Job job_1643520845524_0001 completed successfully
22/01/30 01:39:55 INFO mapreduce.Job: Counters: 49
	File System Counters
		FILE: Number of bytes read=4204
		FILE: Number of bytes written=295149
		FILE: Number of read operations=0
		FILE: Number of large read operations=0
		FILE: Number of write operations=0
		HDFS: Number of bytes read=4924
		HDFS: Number of bytes written=3035
		HDFS: Number of read operations=6
		HDFS: Number of large read operations=0
		HDFS: Number of write operations=2
	Job Counters
		Launched map tasks=1
		Launched reduce tasks=1
		Data-local map tasks=1
		Total time spent by all maps in occupied slots (ms)=43587
		Total time spent by all reduces in occupied slots (ms)=107215
		Total time spent by all map tasks (ms)=43587
		Total time spent by all reduce tasks (ms)=107215
		Total vcore-milliseconds taken by all map tasks=43587
		Total vcore-milliseconds taken by all reduce tasks=107215
		Total megabyte-milliseconds taken by all map tasks=44633088
		Total megabyte-milliseconds taken by all reduce tasks=109788160
	Map-Reduce Framework
		Map input records=111
		Map output records=719
```

- Then we run the mapreduce job command and then the execution continues.



```
		WRONG_REDUCE=0
	File Input Format Counters
		Bytes Read=4796
	File Output Format Counters
		Bytes Written=3035
[cloudera@quickstart ~]$ hadoop fs -ls output1
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2022-01-30 01:39 output1/_SUCCESS
-rw-r--r--   1 cloudera cloudera       3035 2022-01-30 01:39 output1/part-r-00000
[cloudera@quickstart ~]$ hadoop fs -cat output1/part-r-00000
(and	1
(key-len,	1
(list	2
(since	1
(which	1
10-100	1
10TB	1
128MB,	1
3	1
300	1
82,000	1
<key,	2
A	2
All	1
Applications	5
Closeable.close()	2
Comparator	2
CompressionCodec	1
Counters,	2
FileSystem	1
HTTP.	1
Hadoop	1
How	1
If	1
In	2
Input	1
InputFormat	1
InputSplit	2
Iterator,	2
JobConf	2
JobConf.	1
JobConf.setCombinerClass(Class),	1
JobConf.setNumReduceTasks(int).	1
JobConf.setOutputKeyComparatorClass(Class)	1
JobConf.setOutputKeyComparatorClass(Class).	1
JobConf.setOutputValueGroupingComparator(Class).	1
JobConfigurable.configure(JobConf)	2
Many	1
MapReduce	1
```

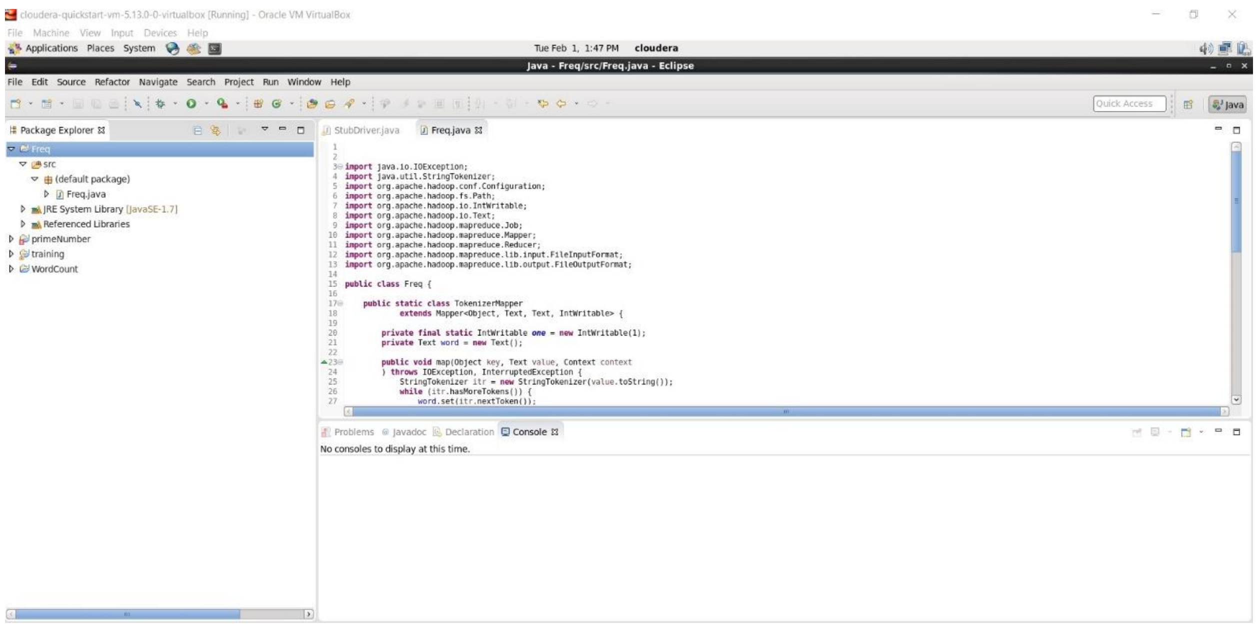
- Finally, we list the files in the output and use cat command to display the output which is the number of times the words are repeated in given file.



```
(and	1
(key-len,	1
(list	2
(since	1
(which	1
10-100	1
10TB	1
128MB,	1
3	1
300	1
82,000	1
<key,	2
A	2
All	1
Applications	5
Closeable.close()	2
Comparator	2
CompressionCodec	1
Counters,	2
FileSystem	1
HTTP.	1
Hadoop	1
How	1
If	1
In	2
Input	1
InputFormat	1
InputSplit	2
Iterator.	2
```

- Here is the visualized version of wordcount program

Frequency of words with letter ‘b’

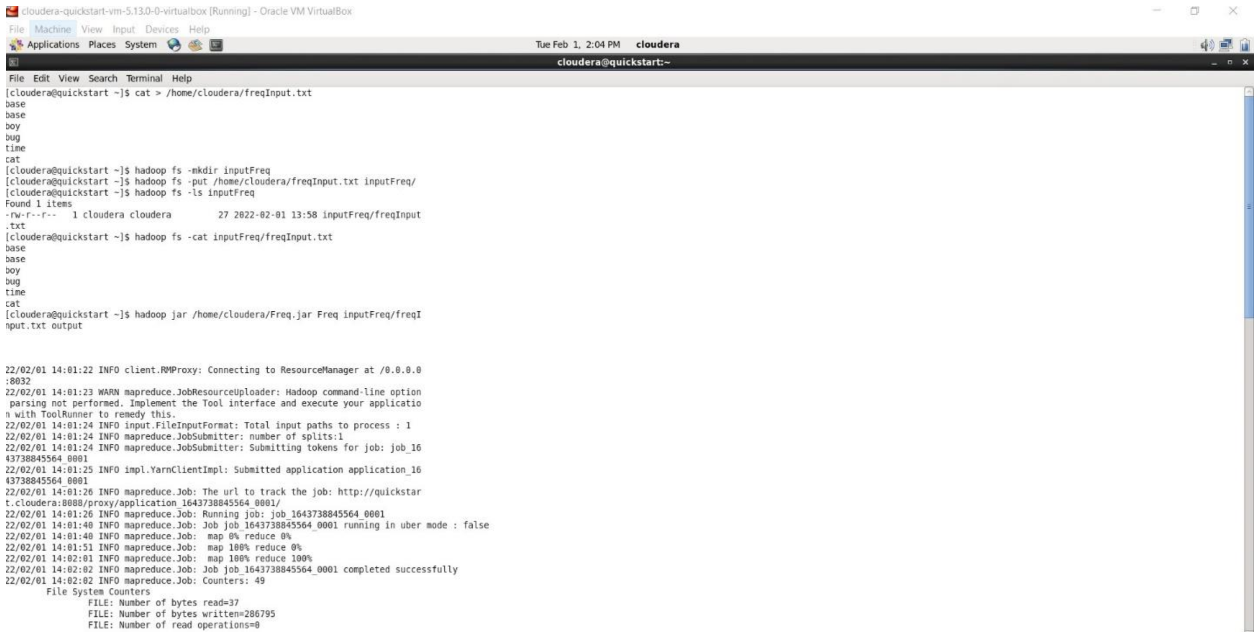


```java


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

    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());
```

No consoles to display at this time.

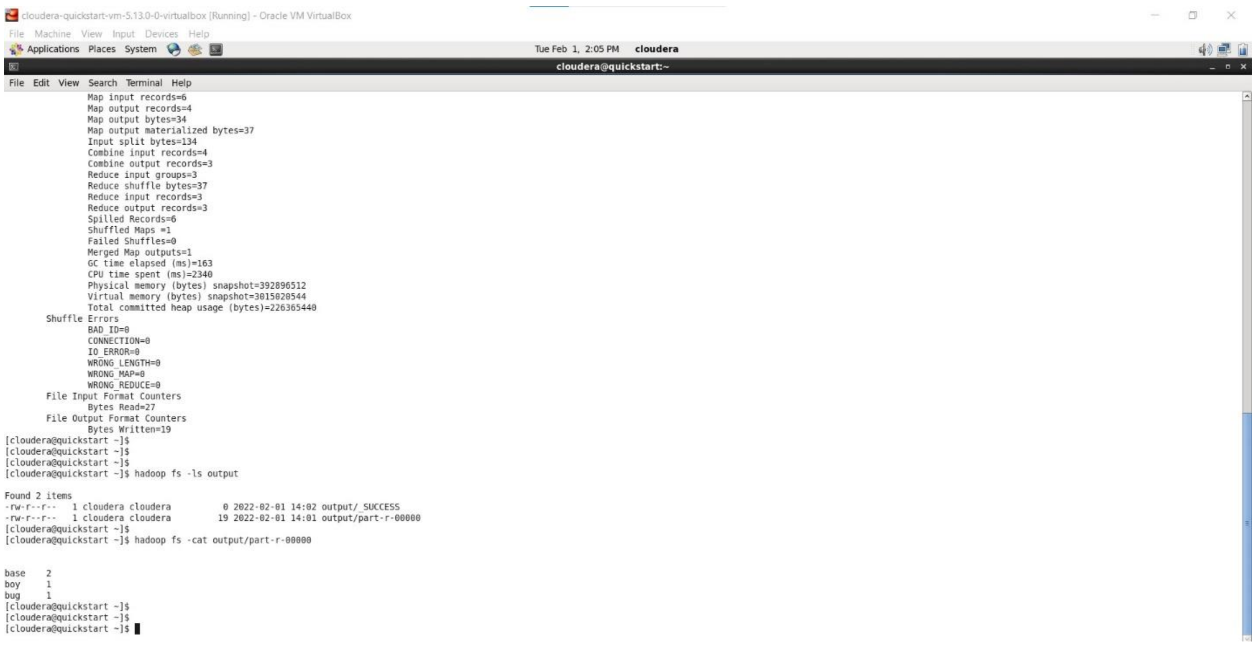
- Firstly we create a java project named Freq and then import all the required header files and add all external jar libs to the program.

- Then we extend the mapper class by creating TokenizerMapper and it takes arguments like key-in,value-in,key-out and value-out. Then we create word variable to store word and one variable to store value.
- Then we create map function which takes arguments like key-in as key, textual content as value variable and context to emit result to next stage.
- Then we StringTokenizer which divides string into tokens and store each of them in itr variable.
- Then we define reducer class as IntSumReducer which takes four inputs. Then we create result variable to store how many number of times the key is appearing.
- Then we write logic which adds words which starts with letter b and gives output to the reducer.


cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Feb 1, 2:04 PM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help

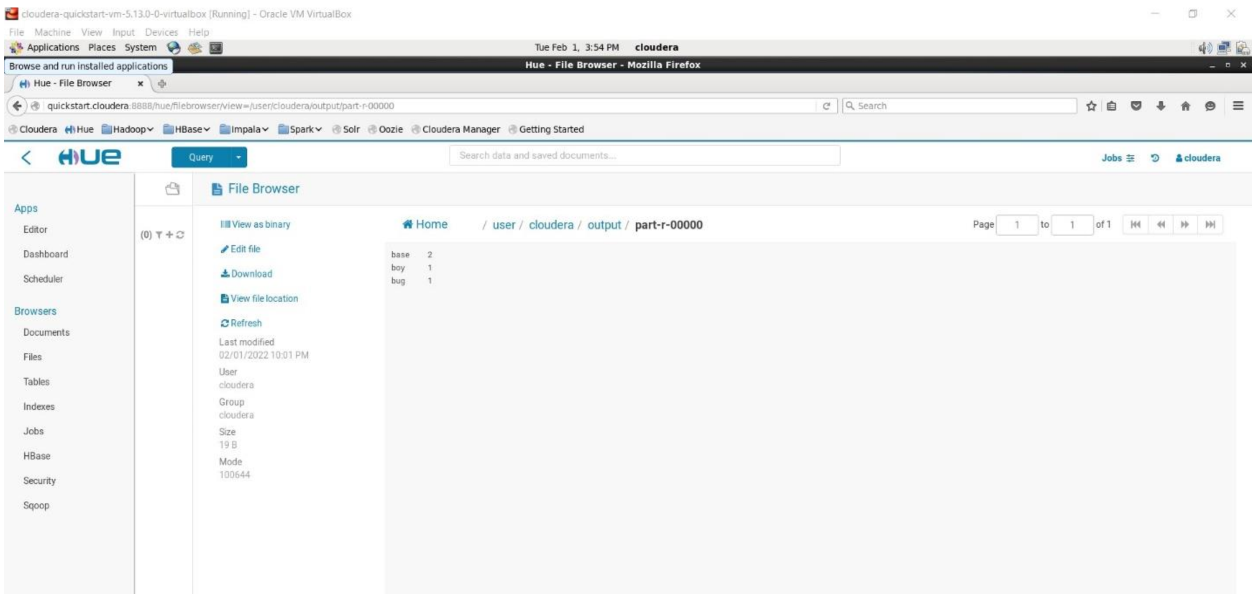
```
[cloudera@quickstart ~]$ cat > /home/cloudera/freqInput.txt
base
base
boy
bug
time
cat
[cloudera@quickstart ~]$ hadoop fs -mkdir inputFreq
[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/freqInput.txt inputFreq/
[cloudera@quickstart ~]$ hadoop fs -ls inputFreq
Found 1 items
-rw-r--r--   1 cloudera cloudera         27 2022-02-01 13:58 inputFreq/freqInput
.txt
[cloudera@quickstart ~]$ hadoop fs -cat inputFreq/freqInput.txt
base
base
boy
bug
time
cat
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Freq.jar Freq inputFreq/freqI
nput.txt output



22/02/01 14:01:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
22/02/01 14:01:23 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
22/02/01 14:01:24 INFO input.FileInputFormat: Total input paths to process : 1
22/02/01 14:01:24 INFO mapreduce.JobSubmitter: number of splits:1
22/02/01 14:01:24 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
43738845564_0001
22/02/01 14:01:25 INFO impl.YarnClientImpl: Submitted application application_16
43738845564_0001
22/02/01 14:01:26 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1643738845564_0001/
22/02/01 14:01:26 INFO mapreduce.Job: Running job: job_1643738845564_0001
22/02/01 14:01:40 INFO mapreduce.Job: Job job_1643738845564_0001 running in uber mode : false
22/02/01 14:01:40 INFO mapreduce.Job:  map 0% reduce 0%
22/02/01 14:01:51 INFO mapreduce.Job:  map 100% reduce 0%
22/02/01 14:02:01 INFO mapreduce.Job:  map 100% reduce 100%
22/02/01 14:02:02 INFO mapreduce.Job: Job job_1643738845564_0001 completed successfully
22/02/01 14:02:02 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=37
                FILE: Number of bytes written=286795
                FILE: Number of read operations=0
```

- Then we create a file and add required input to it using cat command. Then after we import files from local to Hadoop using put command.
- Then we display input using cat command. Then we execute map reduce job and execution continues.



```
                Map input records=6
                Map output records=4
                Map output bytes=34
                Map output materialized bytes=37
                Input split bytes=134
                Combine input records=4
                Combine output records=3
                Reduce input groups=3
                Reduce shuffle bytes=37
                Reduce input records=3
                Reduce output records=3
                Spilled Records=6
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=163
                CPU time spent (ms)=2340
                Physical memory (bytes) snapshot=392896512
                Virtual memory (bytes) snapshot=3015020544
                Total committed heap usage (bytes)=226365440
        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=27
        File Output Format Counters
                Bytes Written=19
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls output

Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2022-02-01 14:02 output/_SUCCESS
-rw-r--r--   1 cloudera cloudera         19 2022-02-01 14:01 output/part-r-00000
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -cat output/part-r-00000


base    2
boy     1
bug     1
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
```

- Finally, we list the files in the output and use cat command to display the output which is the number of times the words with letter 'b' are repeated in given file.



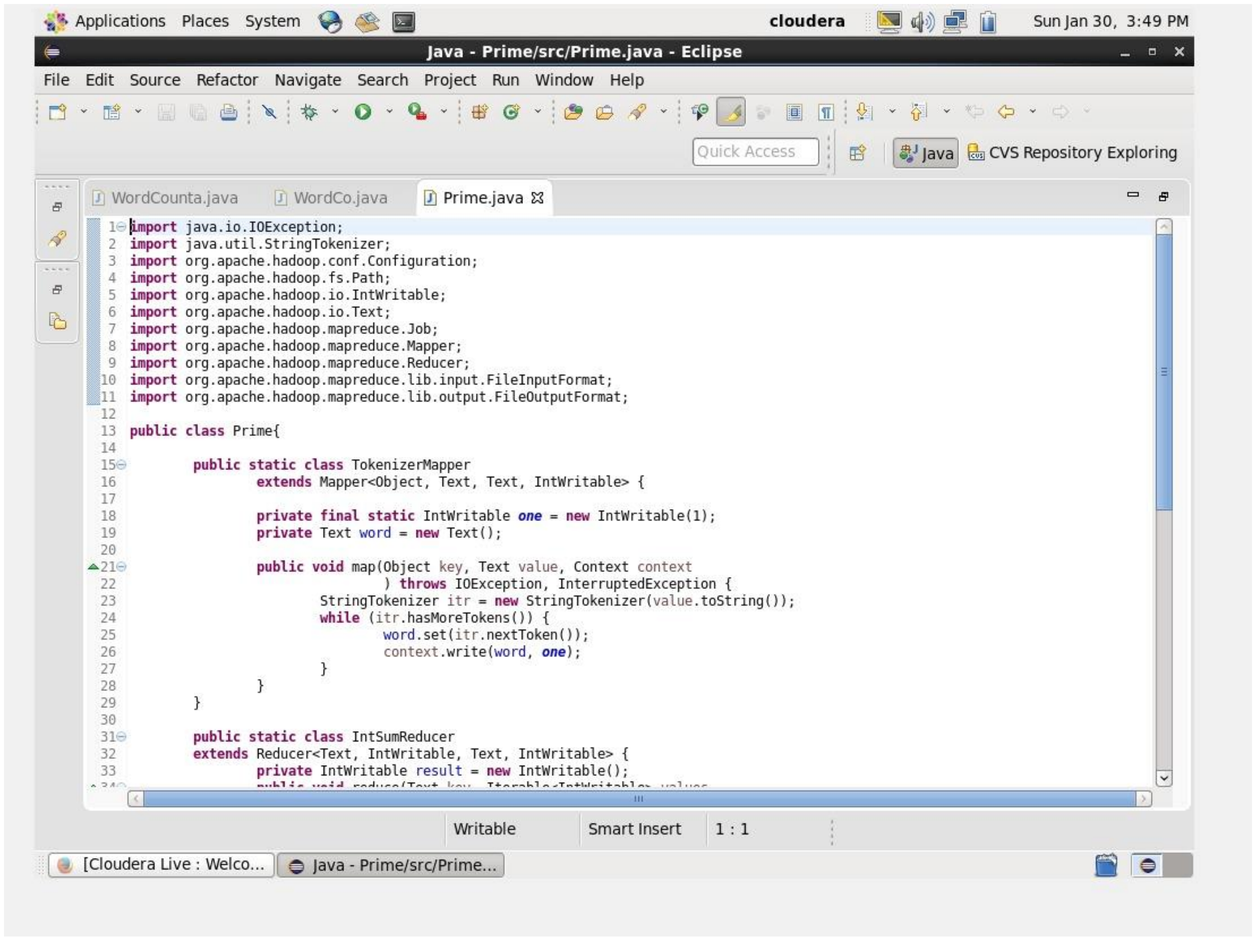
Hue - File Browser - Mozilla Firefox
quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera/output/part-r-00000
File Browser
View as binary
Edit file
Download
View file location
Refresh
Home / user / cloudera / output / part-r-00000

Last modified 02/01/2022 10:01 PM
User cloudera
Group cloudera
Size 19 B
Mode 100644

```
base    2
boy     1
bug     1
```

- Here is visualized version of frequency of b words file

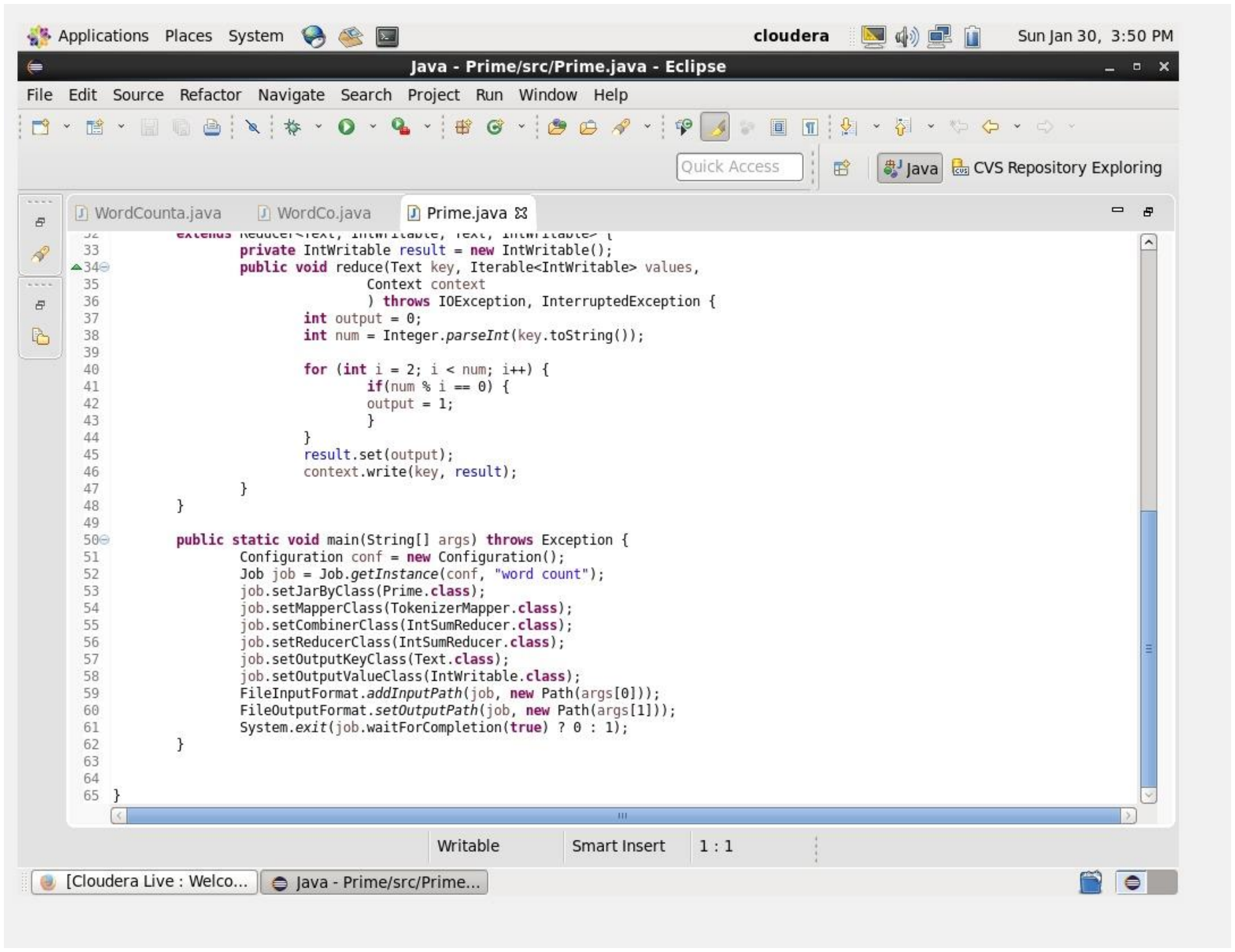
Prime Number Program



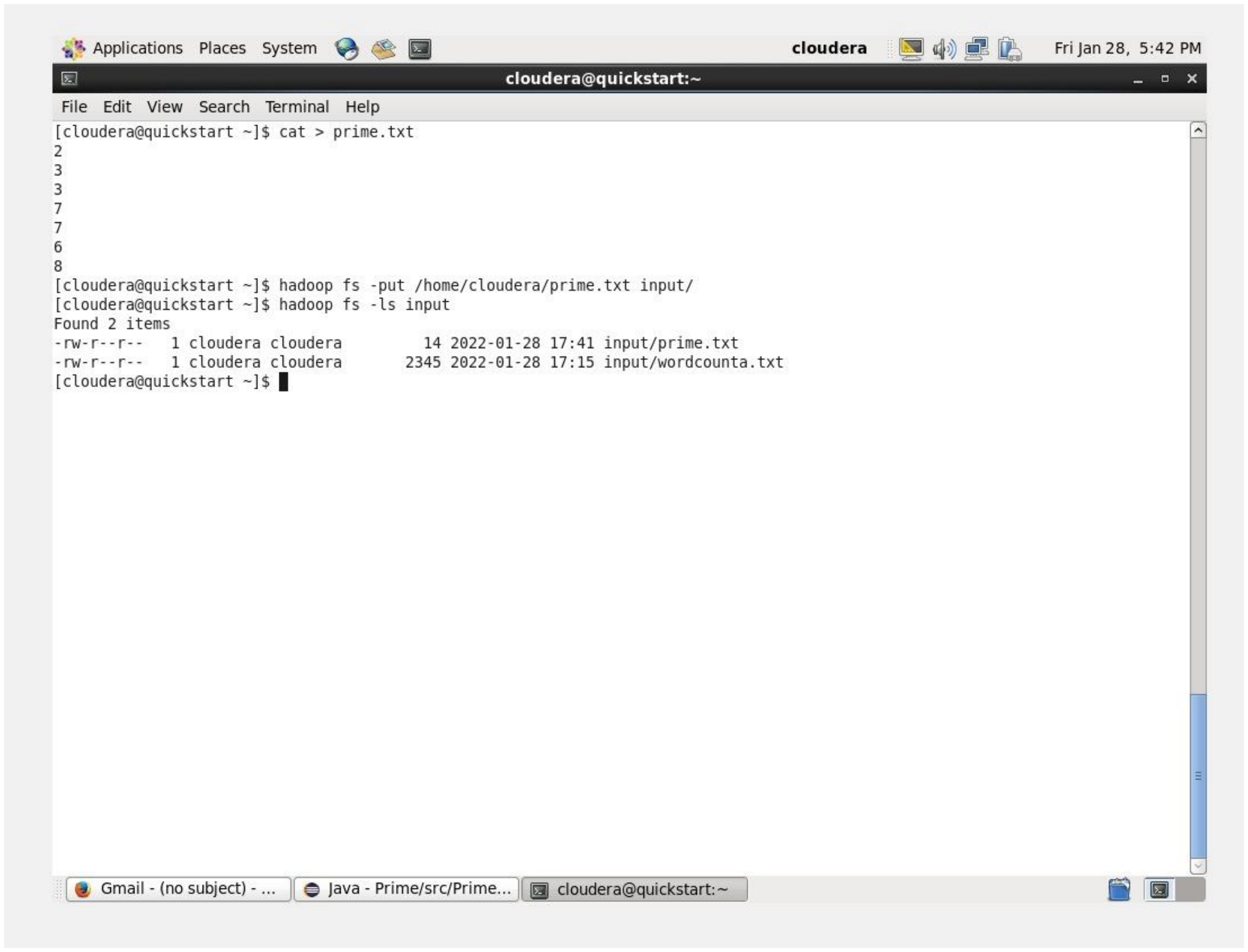
Applications Places System
cloudera
Sun Jan 30, 3:49 PM
Java - Prime/src/Prime.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
CVS Repository Exploring
WordCounta.java
WordCo.java
Prime.java
Writable
Smart Insert
1 : 1
[Cloudera Live : Welco...
Java - Prime/src/Prime...

```java
import java.io.IOException;
import java.util.StringTokenizer;
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 Prime{

        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
```

- Firstly we create a java project named prime and then import all the required header files and add all external jar libs to the program.
- Then we extend the mapper class by creating TokenizerMapper and it takes arguments like key-in,value-in,key-out and value-out. Then we create word variable to store word and one variable to store value.
- Then we create map function which takes arguments like key-in as key, textual content as value variable and context to emit result to next stage.
- Then we StringTokenizer which divides string into tokens and store each of them in itr variable.
- Then we define reducer class as IntSumReducer which takes four inputs. Then we create result variable to store how many number of times the key is appearing.
- Then we write logic which adds words which starts with letter b and gives output to the reducer.

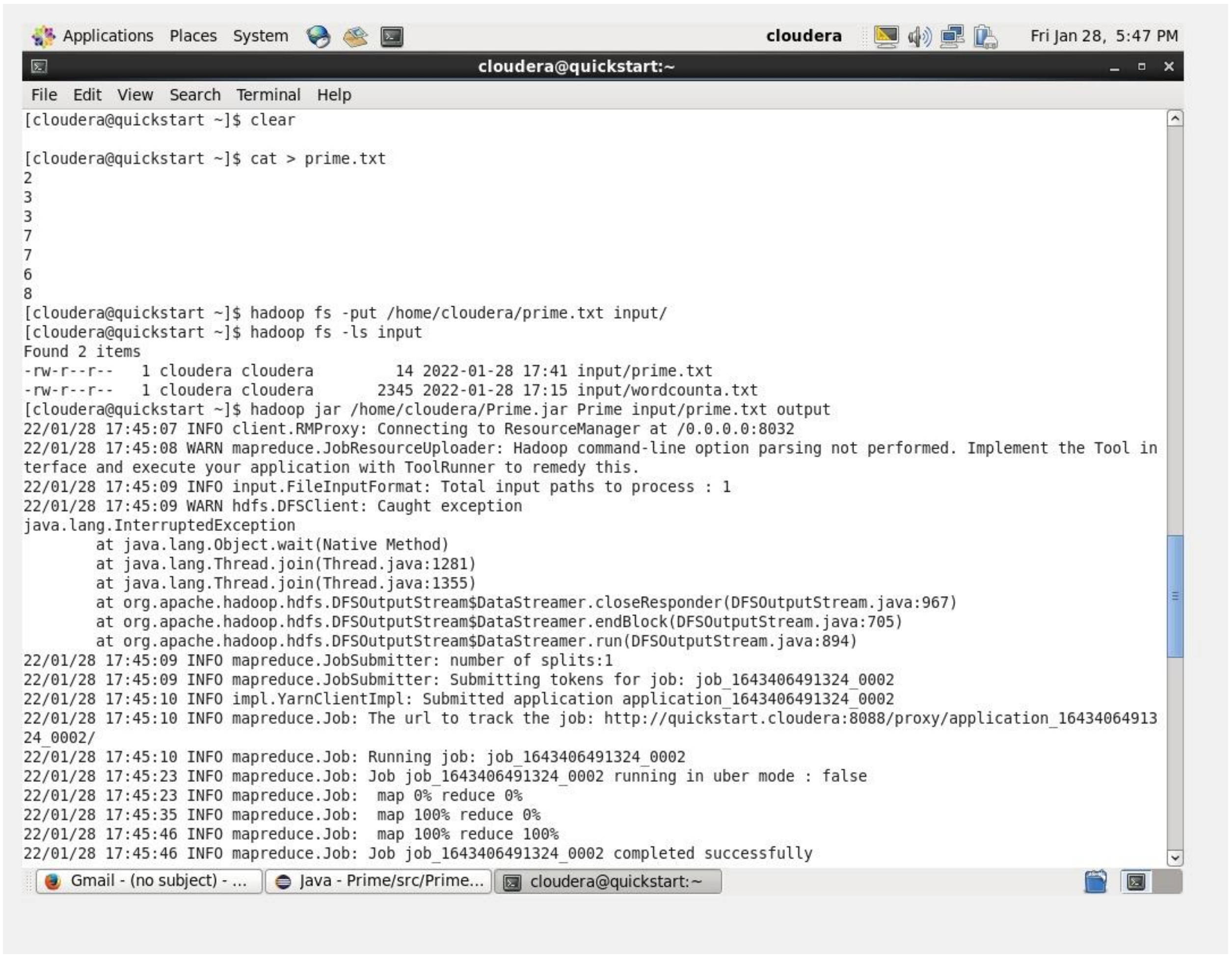

Applications Places System
cloudera
Sun Jan 30, 3:50 PM
Java - Prime/src/Prime.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
CVS Repository Exploring
WordCounta.java
WordCo.java
Prime.java
Writable
Smart Insert
1 : 1
[Cloudera Live : Welco...
Java - Prime/src/Prime...

```
        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 output = 0;
                        int num = Integer.parseInt(key.toString());

                        for (int i = 2; i < num; i++) {
                                if(num % i == 0) {
                                output = 1;
                                }
                        }
                        result.set(output);
                        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(Prime.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);
        }


}
```


Applications Places System
cloudera
Fri Jan 28, 5:42 PM
cloudera@quickstart:~
File Edit View Search Terminal Help
Gmail - (no subject) - ...
Java - Prime/src/Prime...
cloudera@quickstart:~

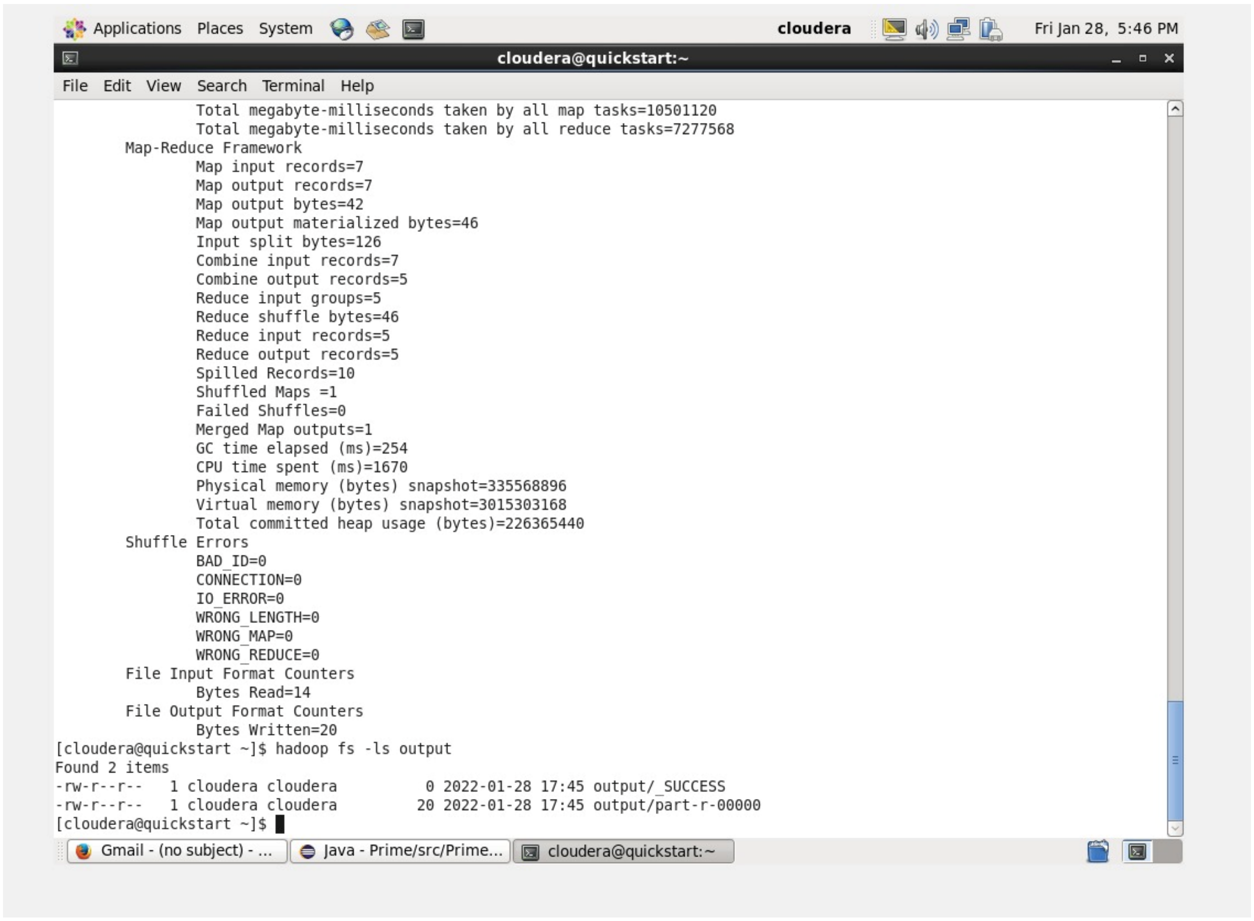
```
[cloudera@quickstart ~]$ cat > prime.txt
2
3
3
7
7
6
8
[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/prime.txt input/
[cloudera@quickstart ~]$ hadoop fs -ls input
Found 2 items
-rw-r--r--   1 cloudera cloudera         14 2022-01-28 17:41 input/prime.txt
-rw-r--r--   1 cloudera cloudera       2345 2022-01-28 17:15 input/wordcounta.txt
[cloudera@quickstart ~]$ 
```

- Then I have added input to file using cat command and listed the files using ls command.


Applications Places System
cloudera
Fri Jan 28, 5:47 PM
cloudera@quickstart:~
File Edit View Search Terminal Help
Gmail - (no subject) - ...
Java - Prime/src/Prime...
cloudera@quickstart:~

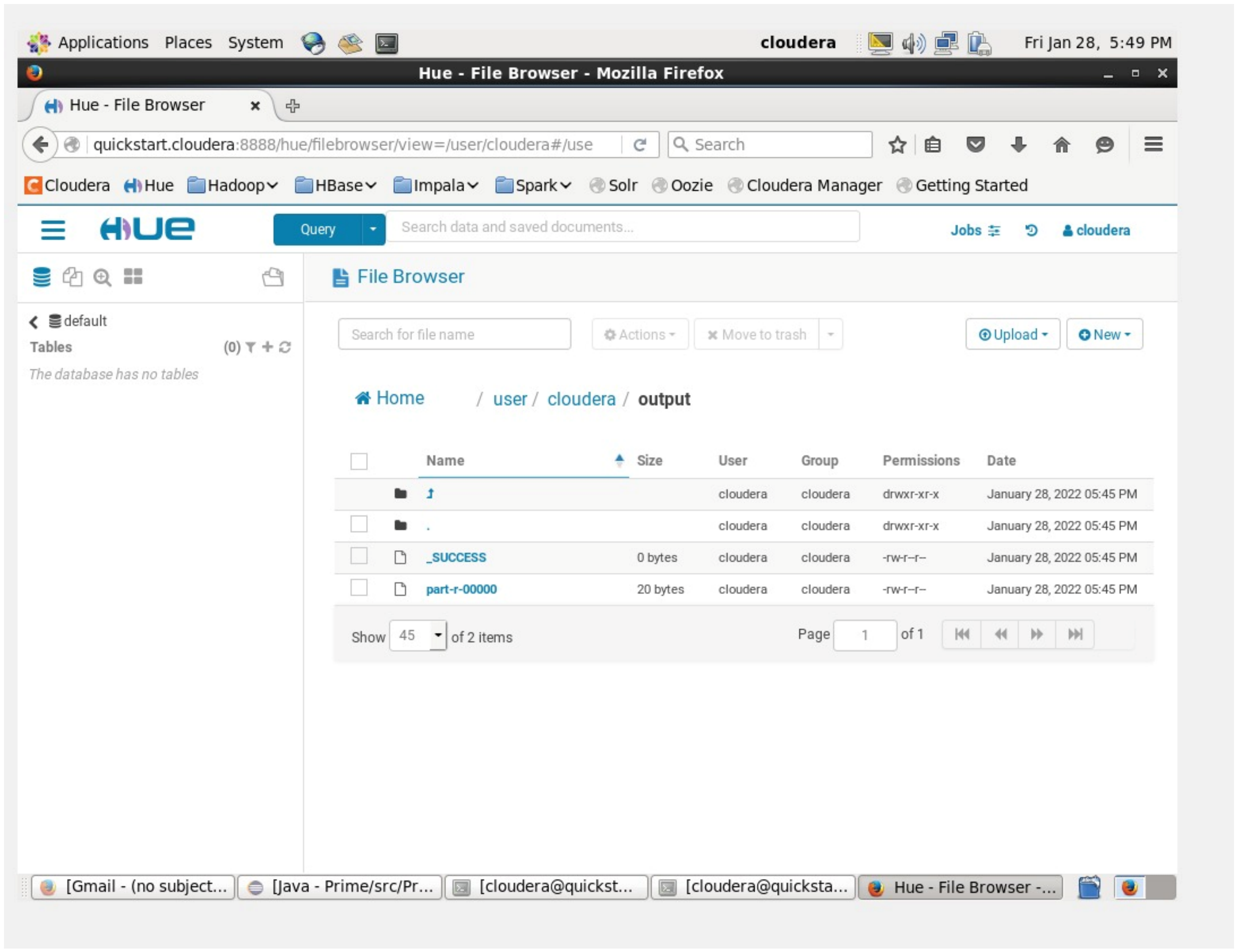
```
[cloudera@quickstart ~]$ clear

[cloudera@quickstart ~]$ cat > prime.txt
2
3
3
7
7
6
8
[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/prime.txt input/
[cloudera@quickstart ~]$ hadoop fs -ls input
Found 2 items
-rw-r--r--   1 cloudera cloudera         14 2022-01-28 17:41 input/prime.txt
-rw-r--r--   1 cloudera cloudera       2345 2022-01-28 17:15 input/wordcounta.txt
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Prime.jar Prime input/prime.txt output
22/01/28 17:45:07 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/01/28 17:45:08 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool in
terface and execute your application with ToolRunner to remedy this.
22/01/28 17:45:09 INFO input.FileInputFormat: Total input paths to process : 1
22/01/28 17:45:09 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
22/01/28 17:45:09 INFO mapreduce.JobSubmitter: number of splits:1
22/01/28 17:45:09 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1643406491324_0002
22/01/28 17:45:10 INFO impl.YarnClientImpl: Submitted application application_1643406491324_0002
22/01/28 17:45:10 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_16434064913
24_0002/
22/01/28 17:45:10 INFO mapreduce.Job: Running job: job_1643406491324_0002
22/01/28 17:45:23 INFO mapreduce.Job: Job job_1643406491324_0002 running in uber mode : false
22/01/28 17:45:23 INFO mapreduce.Job:  map 0% reduce 0%
22/01/28 17:45:35 INFO mapreduce.Job:  map 100% reduce 0%
22/01/28 17:45:46 INFO mapreduce.Job:  map 100% reduce 100%
22/01/28 17:45:46 INFO mapreduce.Job: Job job_1643406491324_0002 completed successfully
```

- Then I have run mapreduce job and exceuted the code.


Applications Places System
cloudera
Fri Jan 28, 5:46 PM
cloudera@quickstart:~
File Edit View Search Terminal Help
Gmail - (no subject) - ...
Java - Prime/src/Prime...
cloudera@quickstart:~

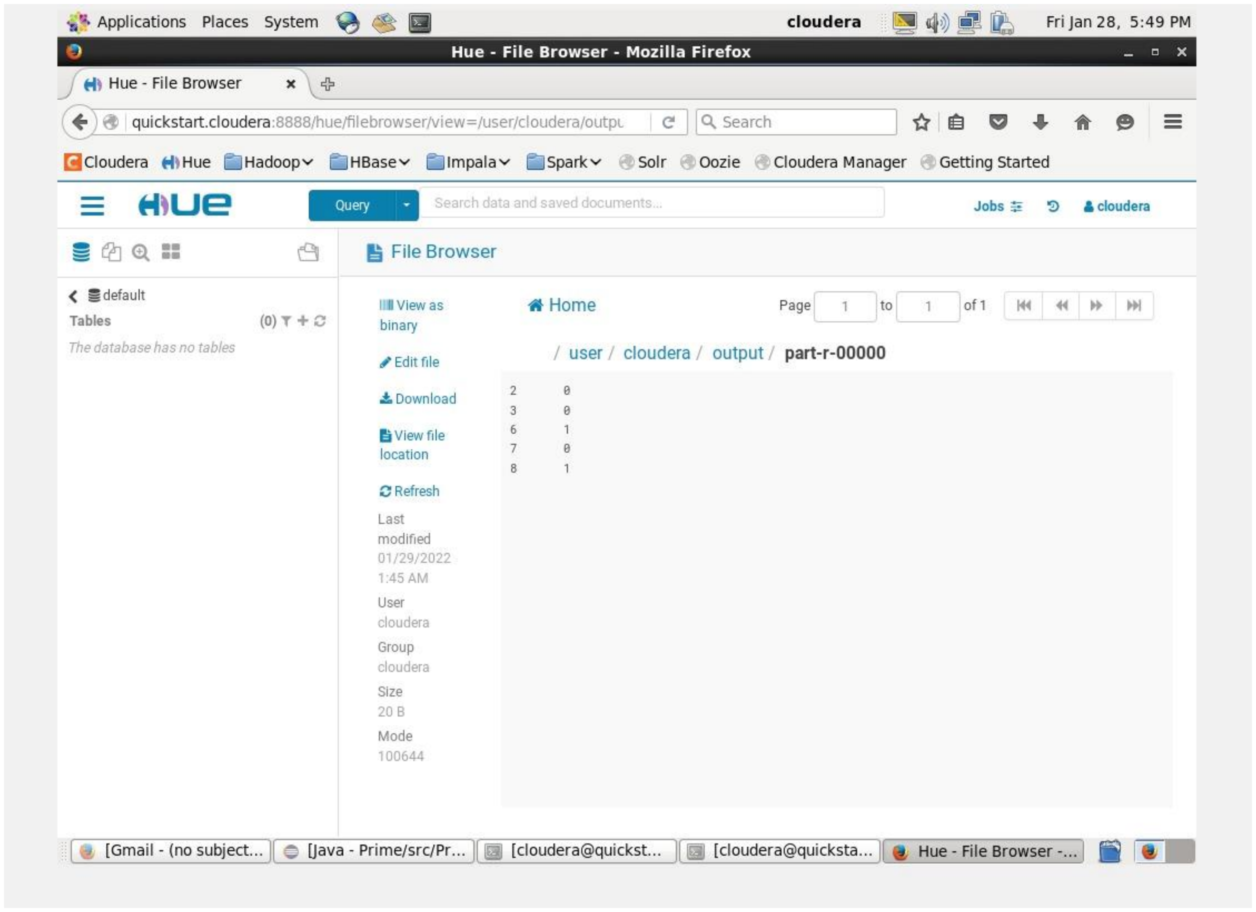
```
                Total megabyte-milliseconds taken by all map tasks=10501120
                Total megabyte-milliseconds taken by all reduce tasks=7277568
        Map-Reduce Framework
                Map input records=7
                Map output records=7
                Map output bytes=42
                Map output materialized bytes=46
                Input split bytes=126
                Combine input records=7
                Combine output records=5
                Reduce input groups=5
                Reduce shuffle bytes=46
                Reduce input records=5
                Reduce output records=5
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=254
                CPU time spent (ms)=1670
                Physical memory (bytes) snapshot=335568896
                Virtual memory (bytes) snapshot=3015303168
                Total committed heap usage (bytes)=226365440
        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=14
        File Output Format Counters
                Bytes Written=20
[cloudera@quickstart ~]$ hadoop fs -ls output
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2022-01-28 17:45 output/_SUCCESS
-rw-r--r--   1 cloudera cloudera         20 2022-01-28 17:45 output/part-r-00000
[cloudera@quickstart ~]$
```

Then I displayed the output using ls command and it gives prime numbers as 1 and others as 0.


Applications Places System
cloudera
Fri Jan 28, 5:49 PM
Hue - File Browser - Mozilla Firefox
Hue - File Browser
quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera#/use
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Query
Search data and saved documents...
Jobs
cloudera
default
Tables (0)
The database has no tables
File Browser
Search for file name
Actions
Move to trash
Upload
New
Home / user / cloudera / output

	Name	Size	User	Group	Permissions	Date
	↥		cloudera	cloudera	drwxr-xr-x	January 28, 2022 05:45 PM
	.		cloudera	cloudera	drwxr-xr-x	January 28, 2022 05:45 PM
	_SUCCESS	0 bytes	cloudera	cloudera	-rw-r--r--	January 28, 2022 05:45 PM
	part-r-00000	20 bytes	cloudera	cloudera	-rw-r--r--	January 28, 2022 05:45 PM

Show 45 of 2 items
Page 1 of 1
[Gmail - (no subject...
[Java - Prime/src/Pr...
[cloudera@quickst...
[cloudera@quicksta...
Hue - File Browser -...



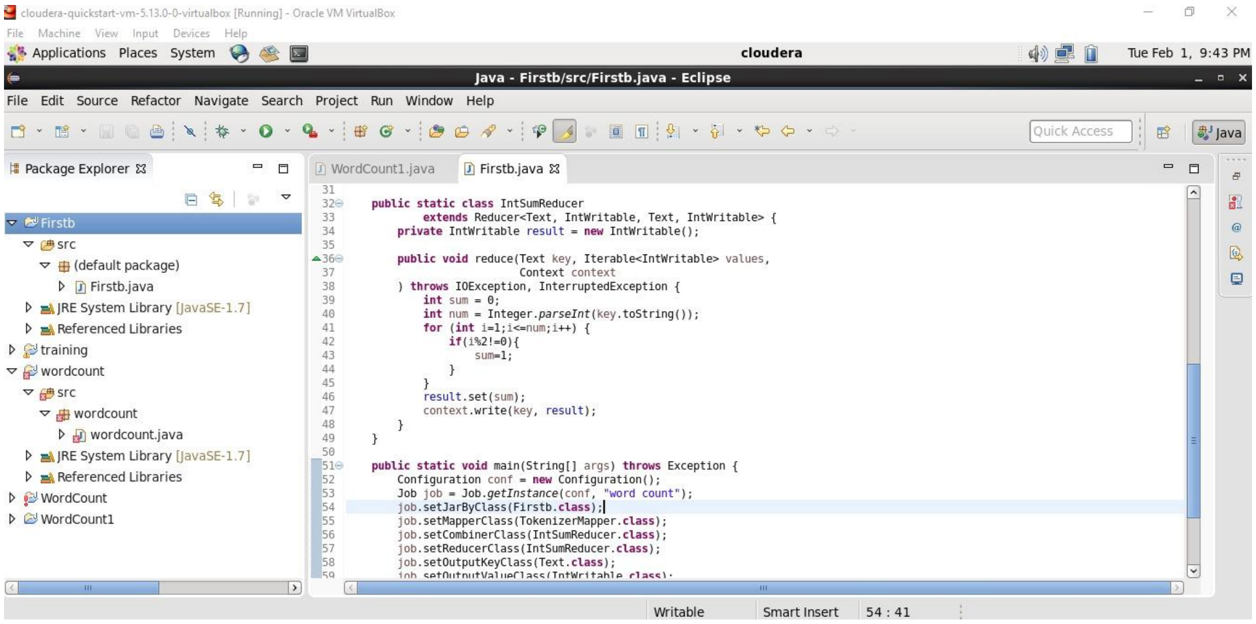
Applications Places System
cloudera
Fri Jan 28, 5:49 PM
Hue - File Browser - Mozilla Firefox
quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera/outpu
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Query
Search data and saved documents...
Jobs
cloudera
default
Tables (0)
The database has no tables
File Browser
View as binary
Edit file
Download
View file location
Refresh
Home
Page 1 to 1 of 1
/ user / cloudera / output / part-r-00000

```
2	0
3	0
6	1
7	0
8	1
```

Last modified 01/29/2022 1:45 AM
User cloudera
Group cloudera
Size 20 B
Mode 100644

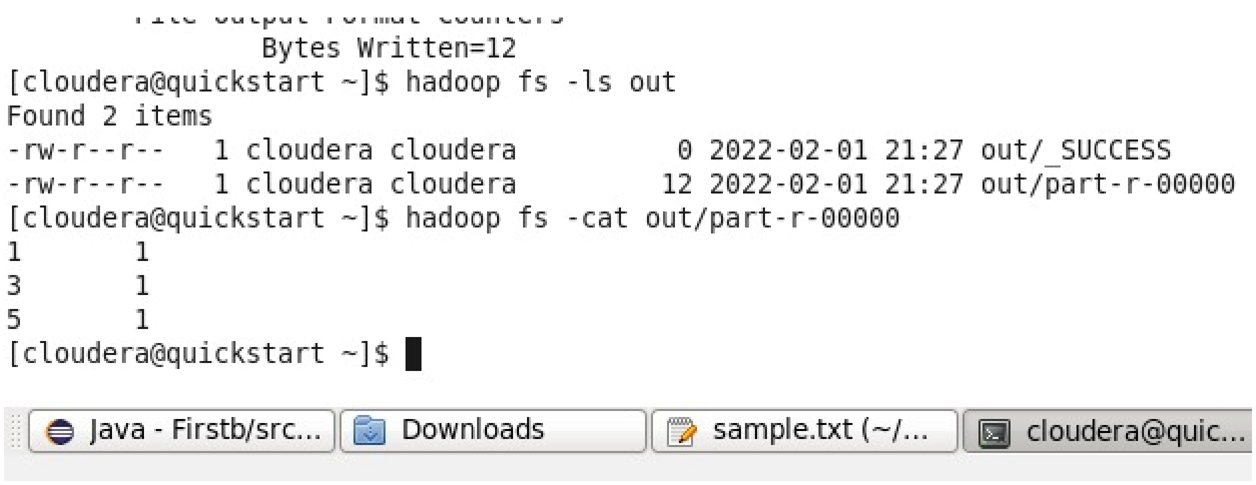
Here is visualized version of prime number program

Odd number Program



cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera
Tue Feb 1, 9:43 PM
Java - Firstb/src/Firstb.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
Firstb
src
(default package)
Firstb.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
training
wordcount
src
wordcount
wordcount.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
WordCount
WordCount1
WordCount1.java
Firstb.java
Writable
Smart Insert
54 : 41

```java

    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;
            int num = Integer.parseInt(key.toString());
            for (int i=1;i<=num;i++) {
                if(i%2!=0){
                    sum=1;
                }
            }
            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(Firstb.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
```



```
                Bytes Written=12
[cloudera@quickstart ~]$ hadoop fs -ls out
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2022-02-01 21:27 out/_SUCCESS
-rw-r--r--   1 cloudera cloudera         12 2022-02-01 21:27 out/part-r-00000
[cloudera@quickstart ~]$ hadoop fs -cat out/part-r-00000
1       1
3       1
5       1
[cloudera@quickstart ~]$ 
```