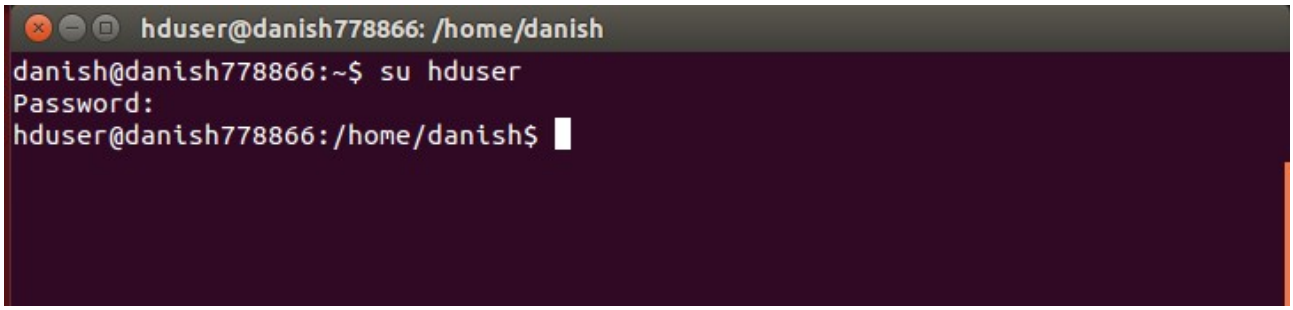
Running the WordCount program

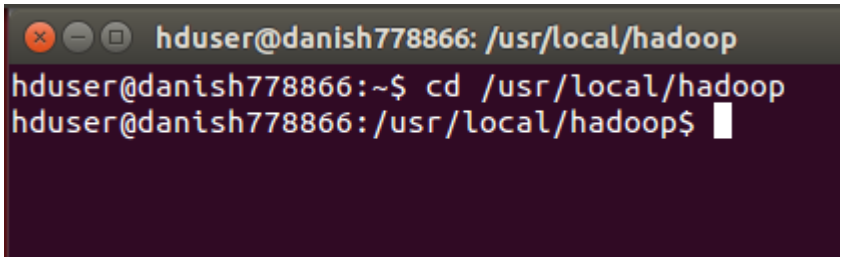
Open a ***terminal*** and follow the steps listed down below

1. Switch to 'hduser' using the command ***su hduser***



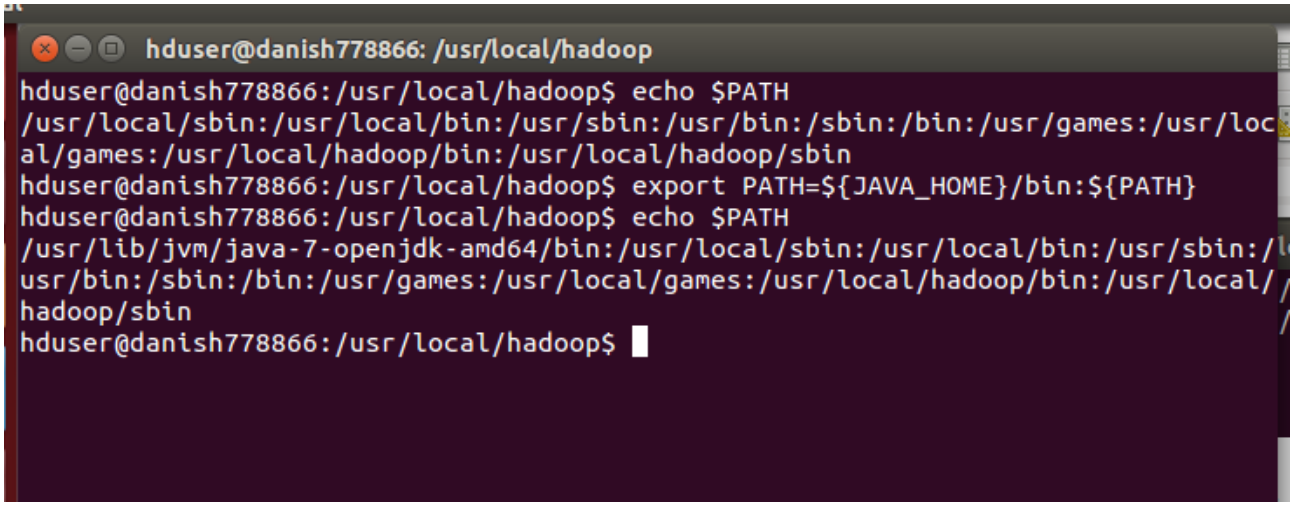
2. Change your working directory to the directory where hadoop is installed. If you have followed the proper hadoop installation steps, you can use the following command:

cd /usr/local/hadoop



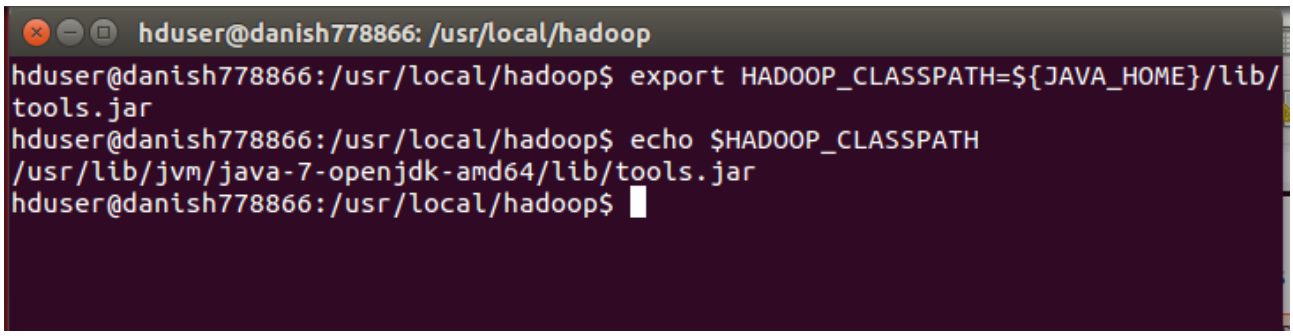
3. Append the 'Java Home' path to the path environment variable by using the following command

export PATH=${JAVA_HOME}/bin:${PATH}



4. Set the ***HADOOP_CLASSPATH*** environment variable as follows

export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar



5. Open ***gedit*** Text Editor and copy the following ***WordCount*** program. Save it as ***WordCount.java***

Note: Save the file in ***/usr/local/hadoop*** directory

```
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 {
```

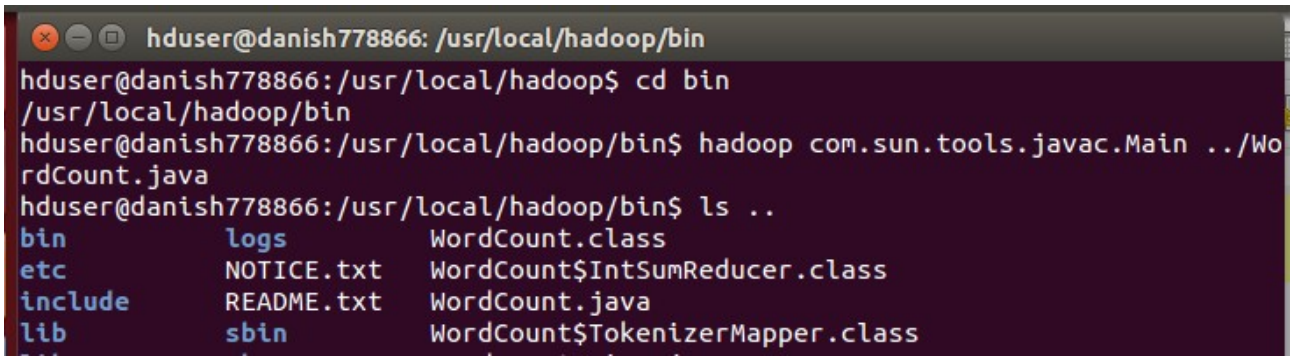
```
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}

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);
 }
}
```

6. Change directory to ***bin*** directory and compile the WordCount program as follows

cd bin

hadoop com.sun.tools.javac.Main ../WordCount.java



7. Go back to ***/usr/local/hadoop*** directory and create a ***jar*** file consisting of the compiled WordCount class files

cd ..

jar cf wc.jar WordCount*.class

```
hduser@danish778866: /usr/local/hadoop
hduser@danish778866:/usr/local/hadoop/bin$ cd ..
hduser@danish778866:/usr/local/hadoop$ ls
bin      libexec      README.txt       WordCount$IntSumReducer.class
etc      LICENSE.txt  sbin             WordCount.java
include  logs         share            WordCount$TokenizerMapper.class
lib      NOTICE.txt   WordCount.class
hduser@danish778866:/usr/local/hadoop$ jar cf wc.jar WordCount*.class
hduser@danish778866:/usr/local/hadoop$ ls
bin      libexec      README.txt  WordCount.class
etc      LICENSE.txt  sbin        WordCount$IntSumReducer.class
include  logs         share       WordCount.java
lib      NOTICE.txt   wc.jar      WordCount$TokenizerMapper.class
hduser@danish778866:/usr/local/hadoop$
```

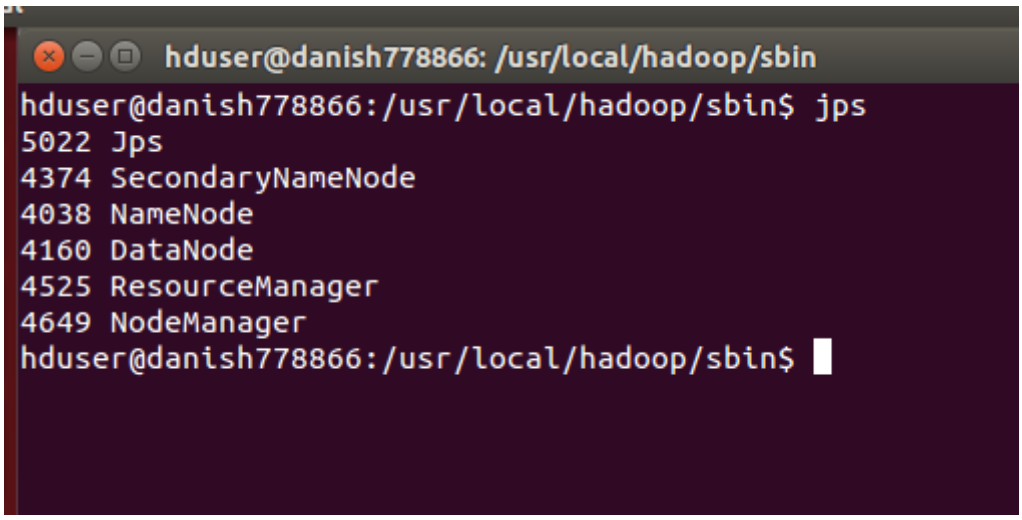
8. Change directory to ***sbin*** and start hadoop

cd sbin

./start-all.sh

```
hduser@danish778866: /usr/local/hadoop/sbin
hduser@danish778866:/usr/local/hadoop$ cd sbin
/usr/local/hadoop/sbin
hduser@danish778866:/usr/local/hadoop/sbin$ ./start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
15/09/16 07:56:20 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-na
menode-danish778866.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-da
tanode-danish778866.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hd
user-secondarynamenode-danish778866.out
15/09/16 07:56:39 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resource
manager-danish778866.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-n
odemanager-danish778866.out
hduser@danish778866:/usr/local/hadoop/sbin$
```

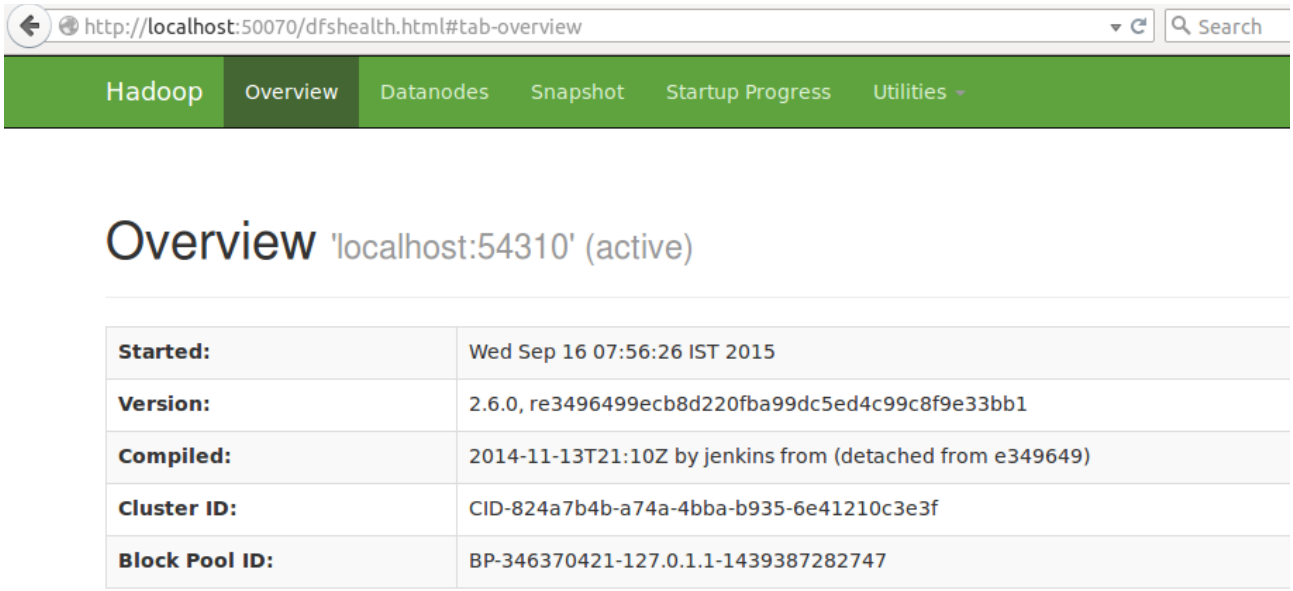
9. Run the ***jps*** command to check if all the nodes have started



10. Open the ***Web UI*** of Hadoop as follows

- Open a browser

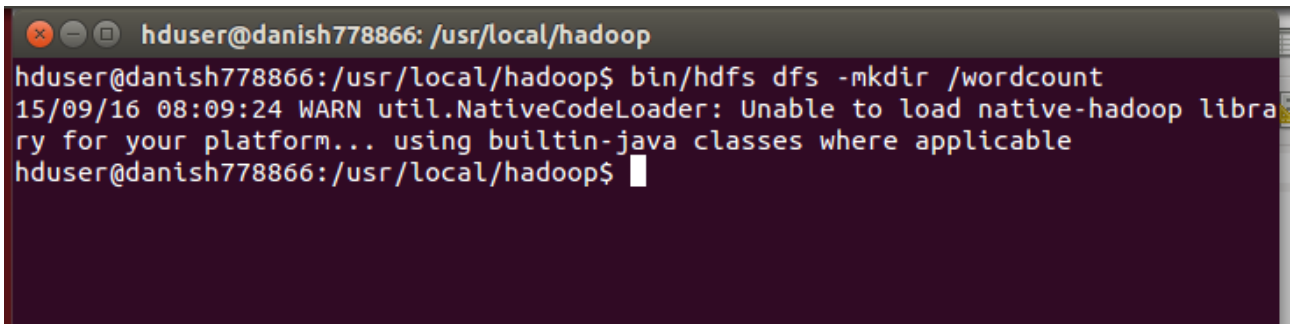
- Go to the URL ***http://localhost:50070***



11. Create a directory in the Hadoop Distributed File Systes (hdfs) to store the input and output files on the WordCount program

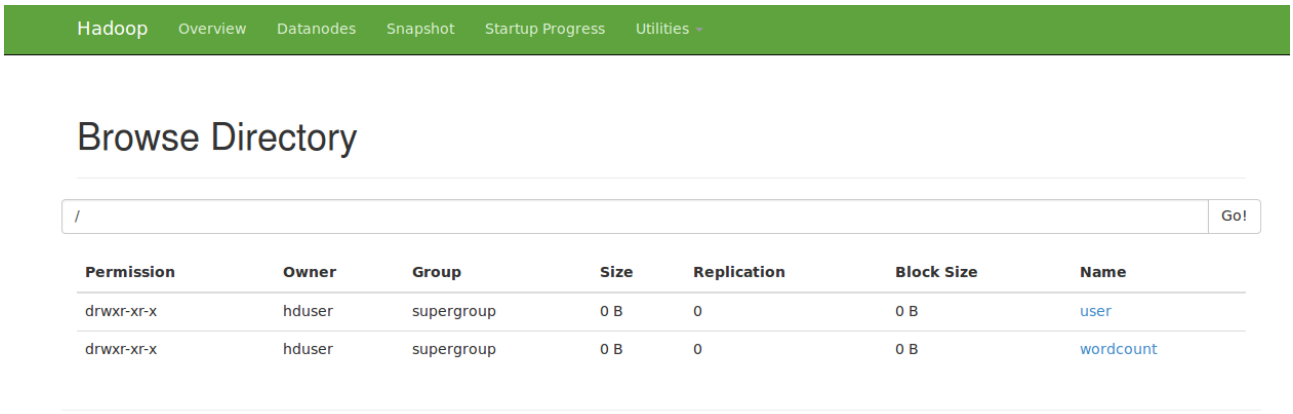
Note: Working directory should be ***/usr/local/hadoop***

bin/hdfs dfs -mkdir /wordcount

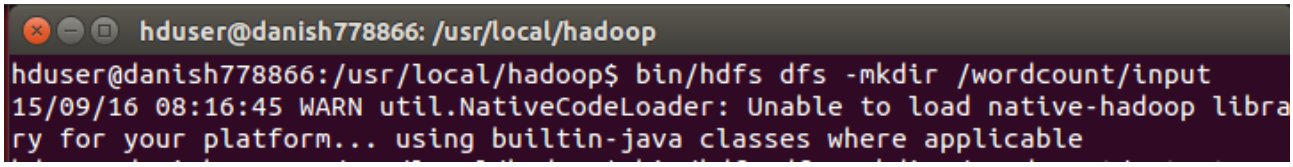


You can verify the created directory using the Web UI as follows:

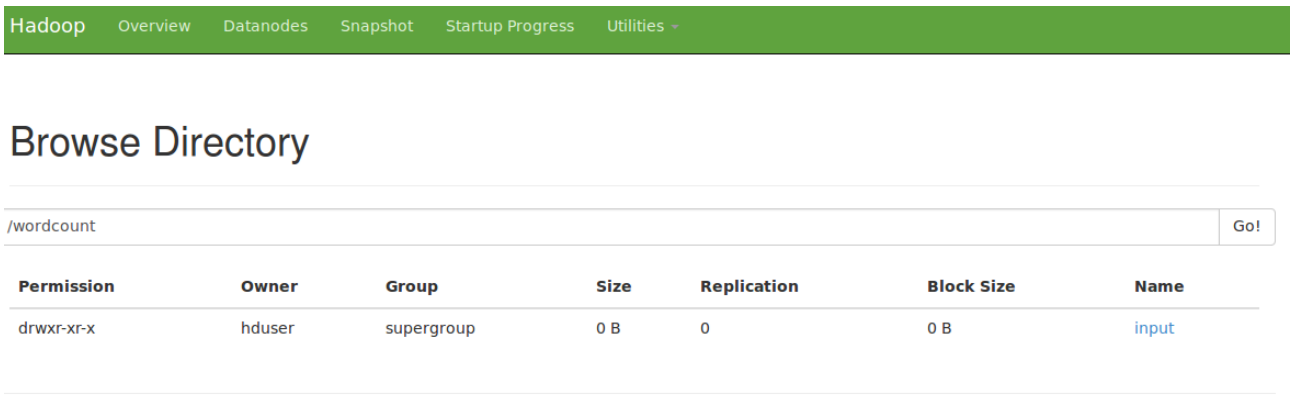
- Go to the Web UI
- Click on ***Utilities*** in the Navigation bar
- Click on ***Browse File System***
- There should be a directory named ***wordcount***



12. Create a directory for storing input files



Verify the created directory using the Web UI



13. Create input files ***file01*** and ***file02*** with the following content

Note: It would be better to create the input files in the ***/usr/local/hadoop*** directory itself. Run the command ***sudo gedit file01*** and save the following content in the file (working directory should be ***/usr/local/hadoop***)

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment.

It is part of the Apache project sponsored by the Apache Software Foundation.

Hadoop makes it possible to run applications on systems with thousands of nodes involving thousands of terabytes.

Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating uninterrupted in case of a node failure.

This approach lowers the risk of catastrophic system failure, even if a significant number of nodes become inoperative.

Hadoop was inspired by Google's MapReduce, a software framework in which an application is broken down into numerous small parts. Any of these parts (also called fragments or blocks) can be run on any node in the cluster.

Doug Cutting, Hadoop's creator, named the framework after his child's stuffed toy elephant.

The current Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper.

The Hadoop framework is used by major players including Google, Yahoo and IBM, largely for applications involving search engines and advertising.

The preferred operating systems are Windows and Linux but Hadoop can also work with BSD and OS X.

Similarly, create the file ***file02*** with the following content

Doug Cutting, Cloudera's Chief Architect, helped create Apache Hadoop out of necessity as data from the web exploded, and grew far beyond the ability of traditional systems to handle it.

Hadoop was initially inspired by papers published by Google outlining its approach to handling an avalanche of data, and has since become the de facto standard for storing, processing and analyzing hundreds of terabytes, and even petabytes of data.Apache Hadoop is 100% open source, and pioneered a fundamentally new way of storing and processing data.

Instead of relying on expensive, proprietary hardware and different systems to store and process data, Hadoop enables distributed parallel processing of huge amounts of data across inexpensive, industry-standard servers that both store and process the data, and can scale without limits.

With Hadoop, no data is too big.

And in today's hyper-connected world where more and more data is being created every day, Hadoop's breakthrough advantages mean that businesses and organizations can now find value in data that was recently considered useless.

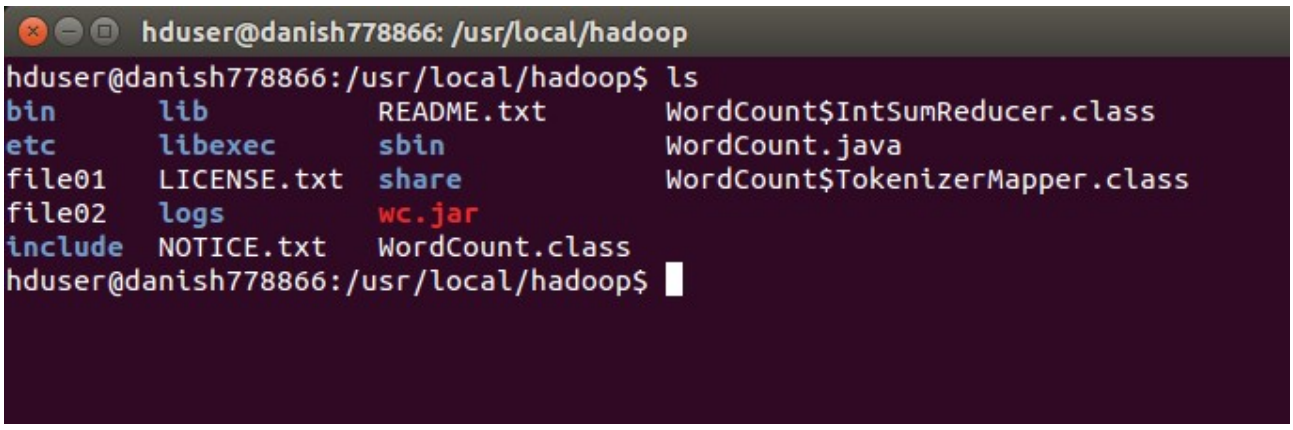
Hadoop can handle all types of data from disparate systems: structured, unstructured, log files, pictures, audio files, communications records, email – just about anything you can think of, regardless of its native format.

Even when different types of data have been stored in unrelated systems, you can dump it all into your Hadoop cluster with no prior need for a schema.

In other words, you don't need to know how you intend to query your data before you store it; Hadoop lets you decide later and over time can reveal questions you never even thought to ask.

By making all of your data useable, not just what's in your databases, Hadoop lets you see relationships that were hidden before and reveal answers that have always been just out of reach.

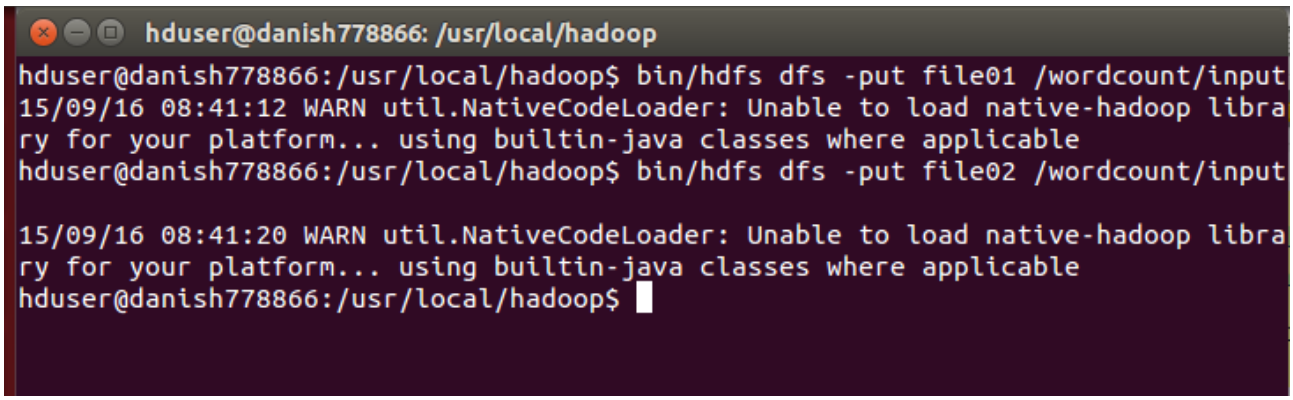
You can start making more decisions based on hard data instead of hunches and look at complete data sets, not just samples.



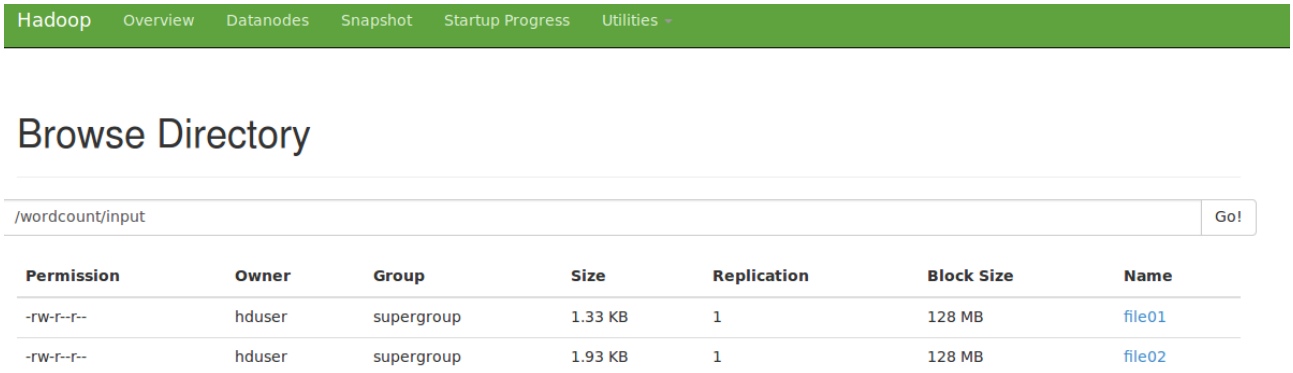
14. Put the input files into ***hdfs*** by using the following command

bin/hdfs dfs -put file01 /wordcount/input

bin/hdfs dfs -put file02 /wordcount/input



You can verify this using the Web UI



15. Change working directory to ***bin*** and execute hadoop on the input files, output will be generated in the specified output directory

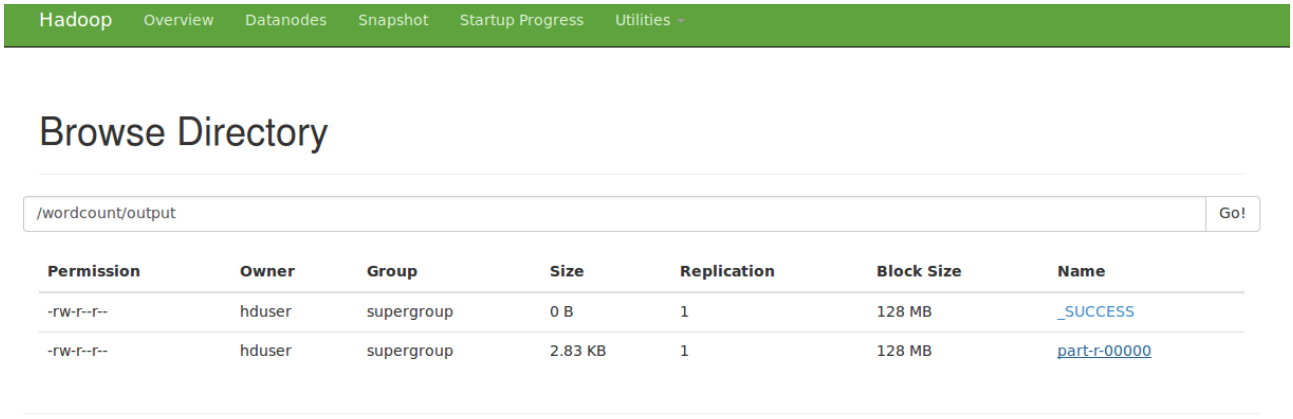
hadoop jar ../wc.jar WordCount /wordcount/input /wordcount/output

This will create a directory named ***output*** inside the ***wordcount*** directory in hdfs. The output file will be present in this directory

```
@danish778866: /usr/local/hadoop/bin

hduser@danish778866:/usr/local/hadoop$ cd bin
/usr/local/hadoop/bin
hduser@danish778866:/usr/local/hadoop/bin$ hadoop jar ../wc.jar WordCount /wordc
ount/input /wordcount/output
15/09/16 08:49:22 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
15/09/16 08:49:23 INFO Configuration.deprecation: session.id is deprecated. Inst
ead, use dfs.metrics.session-id
15/09/16 08:49:23 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName
=JobTracker, sessionId=
15/09/16 08:49:23 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
15/09/16 08:49:23 INFO input.FileInputFormat: Total input paths to process : 2
15/09/16 08:49:23 INFO mapreduce.JobSubmitter: number of splits:2
15/09/16 08:49:24 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_lo
cal238647808_0001
15/09/16 08:49:24 INFO mapreduce.Job: The url to track the job: http://localhost
:8080/
15/09/16 08:49:24 INFO mapred.LocalJobRunner: OutputCommitter set in config null
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=8662
                HDFS: Number of bytes written=2898
                HDFS: Number of read operations=25
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=5
        Map-Reduce Framework
                Map input records=20
                Map output records=532
                Map output bytes=5469
                Map output materialized bytes=4499
                Input split bytes=220
                Combine input records=532
                Combine output records=344
                Reduce input groups=314
                Reduce shuffle bytes=4499
                Reduce input records=344
                Reduce output records=314
                Spilled Records=688
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=10
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=1031798784
        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=3343
        File Output Format Counters
                Bytes Written=2898
hduser@danish778866:/usr/local/hadoop/bin$
```

16. Verify the generated output:

Using the Web UI

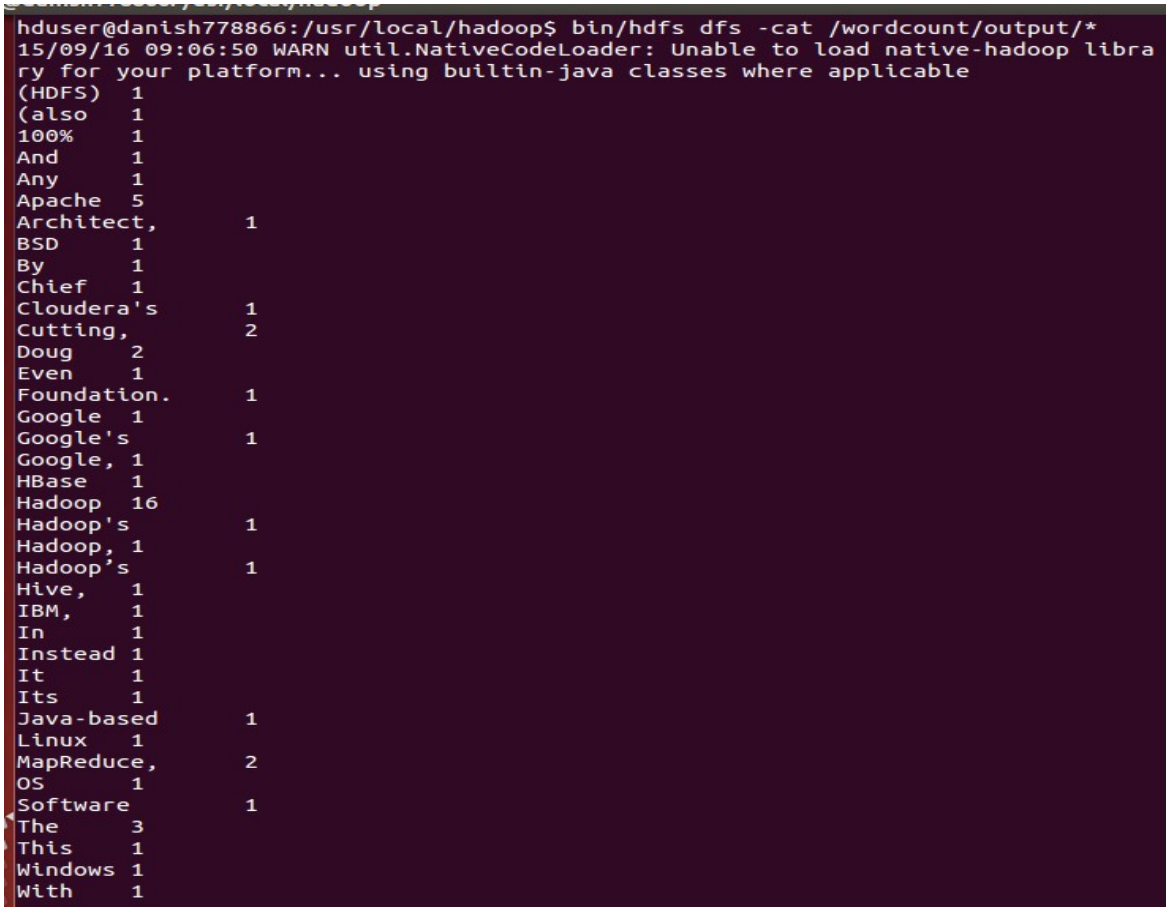


Click on the ***part-r-00000*** file and then on ***Download***. This will trigger a download and the file will be downloaded into your computer. You can open the file and verify the results.

Using CLI

Issue the following command (working directory should be /usr/local/hadoop)

bin/hdfs dfs -cat /wordcount/output/*



17. Stop ***hadoop*** using the following command once you are done

cd sbin

./stop-all.sh