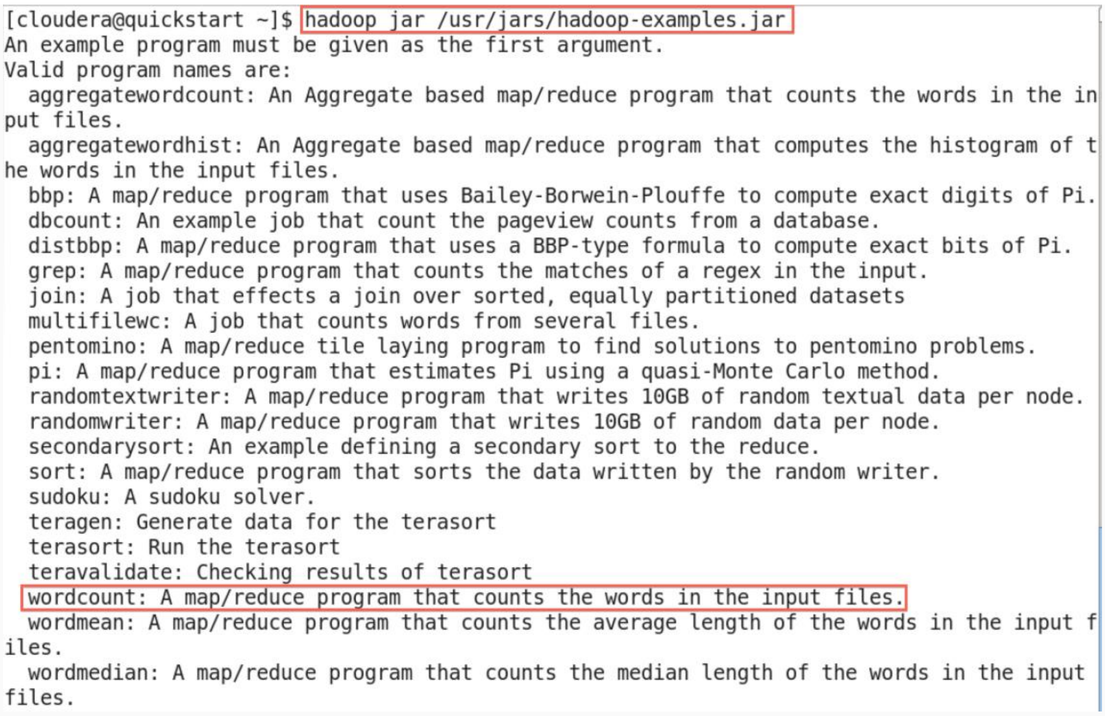
WORDCOUNT WITH HADOOP

By the end of this activity, you will be able to:

- Execute the WordCount application.
- Copy the results from WordCount out of HDFS.

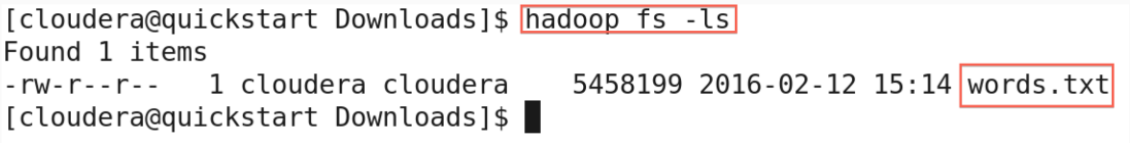
1. Open a terminal shell**.** Start the Cloudera VM in VirtualBox, if not already running, and open a terminal shell. Detailed instructions for these steps can be found in the previous Readings.
2. **See example MapReduce programs.** Hadoop comes with several example MapReduce applications. You can see a list of them by running *hadoop jar /usr/jars/hadoop-examples.jar.* We are interested in running WordCount.



```
[cloudera@quickstart ~]$ hadoop jar /usr/jars/hadoop-examples.jar
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the in
put files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of t
he words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input f
iles.
  wordmedian: A map/reduce program that counts the median length of the words in the input
files.
```

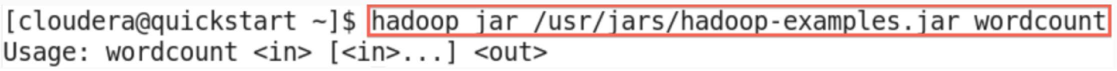
The output says that WordCount takes the name of one or more input files and the name of the output directory. Note that these files are in HDFS, not the local file system.

3. **Verify the input file exists.** In the previous Reading, we downloaded the complete works of Shakespeare and copied them into HDFS. Let's make sure this file is still in HDFS so we can run WordCount on it. Run *hadoop fs -ls*



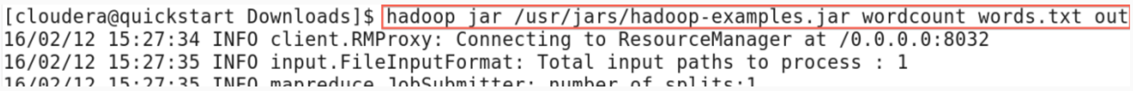
```
[cloudera@quickstart Downloads]$ hadoop fs -ls
Found 1 items
-rw-r--r--   1 cloudera cloudera    5458199 2016-02-12 15:14 words.txt
[cloudera@quickstart Downloads]$
```

4. **See WordCount command line arguments.** We can learn how to run WordCount by examining its command-line arguments. Run *hadoop jar /usr/jars/hadoop-examples.jar wordcount*.



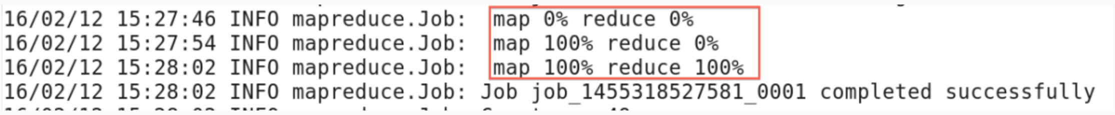
```
[cloudera@quickstart ~]$ hadoop jar /usr/jars/hadoop-examples.jar wordcount
Usage: wordcount <in> [<in>...] <out>
```

5. **Run WordCount.** Run WordCount for words.txt: *hadoop jar /usr/jars/hadoop-examples.jar wordcount words.txt out*



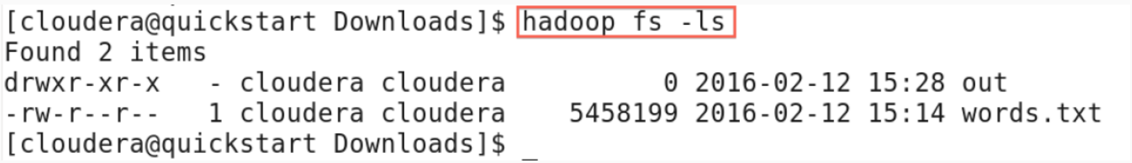
```
[cloudera@quickstart Downloads]$ hadoop jar /usr/jars/hadoop-examples.jar wordcount words.txt out
16/02/12 15:27:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/02/12 15:27:35 INFO input.FileInputFormat: Total input paths to process : 1
16/02/12 15:27:35 INFO mapreduce.JobSubmitter: number of splits:1
```

As WordCount executes, the Hadoop prints the progress regarding Map and Reduce. When the WordCount is complete, both will say *100%*.



```
16/02/12 15:27:46 INFO mapreduce.Job:  map 0% reduce 0%
16/02/12 15:27:54 INFO mapreduce.Job:  map 100% reduce 0%
16/02/12 15:28:02 INFO mapreduce.Job:  map 100% reduce 100%
16/02/12 15:28:02 INFO mapreduce.Job: Job job_1455318527581_0001 completed successfully
```

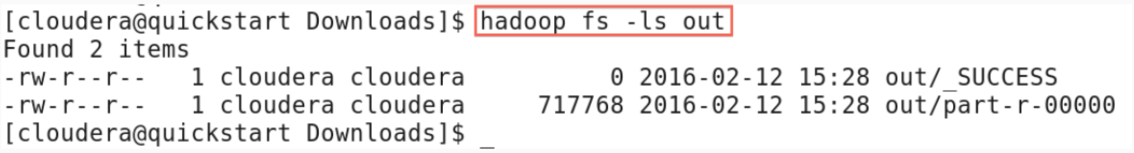
6. **See WordCount output directory.** Once WordCount is finished, let's verify the output was created. First, let's see that the output directory, *out,* was created in HDFS by running *hadoop fs –ls*



```
[cloudera@quickstart Downloads]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - cloudera cloudera          0 2016-02-12 15:28 out
-rw-r--r--   1 cloudera cloudera    5458199 2016-02-12 15:14 words.txt
[cloudera@quickstart Downloads]$ _
```

We can see there are now two items in HDFS: *words.txt* is the text file that we previously created, and *out* is the directory created by WordCount.

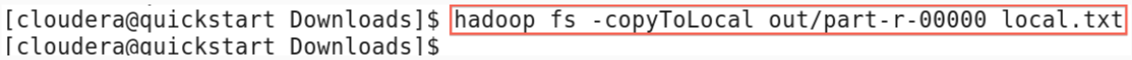
7. **Look inside output directory.** The directory created by WordCount contains several files. Look inside the directory by running *hadoop –fs ls out*



```
[cloudera@quickstart Downloads]$ hadoop fs -ls out
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2016-02-12 15:28 out/_SUCCESS
-rw-r--r--   1 cloudera cloudera     717768 2016-02-12 15:28 out/part-r-00000
[cloudera@quickstart Downloads]$ _
```

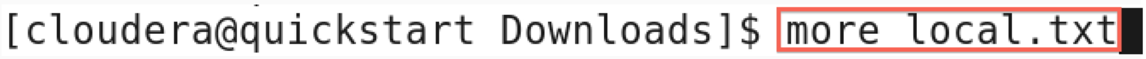
The file *part-r-00000* contains the results from WordCount. The file _SUCCESS means WordCount executed successfully.

8. **Copy WordCount results to local file system.** Copy *part-r-00000* to the local file system by running *hadoop fs –copyToLocal out/part-r-00000 local.txt*



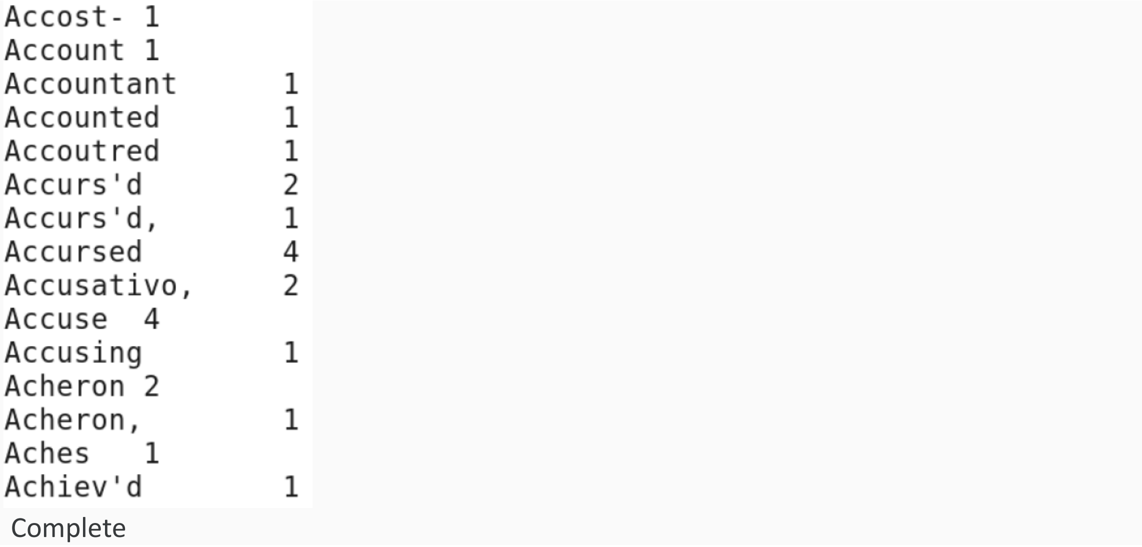
```
[cloudera@quickstart Downloads]$ hadoop fs -copyToLocal out/part-r-00000 local.txt
[cloudera@quickstart Downloads]$
```

9. **View the WordCount results.** View the contents of the results: *more local.txt*



```
[cloudera@quickstart Downloads]$ more local.txt
```

Each line of the results file shows the number of occurrences for a word in the input file. For example, *Accuse* appears four times in the input, but *Accusing* appears only once.



```
Accost- 1
Account 1
Accountant      1
Accounted       1
Accoutred       1
Accurs'd        2
Accurs'd,       1
Accursed        4
Accusativo,     2
Accuse  4
Accusing        1
Acheron 2
Acheron,        1
Aches   1
Achiev'd        1
```

Complete