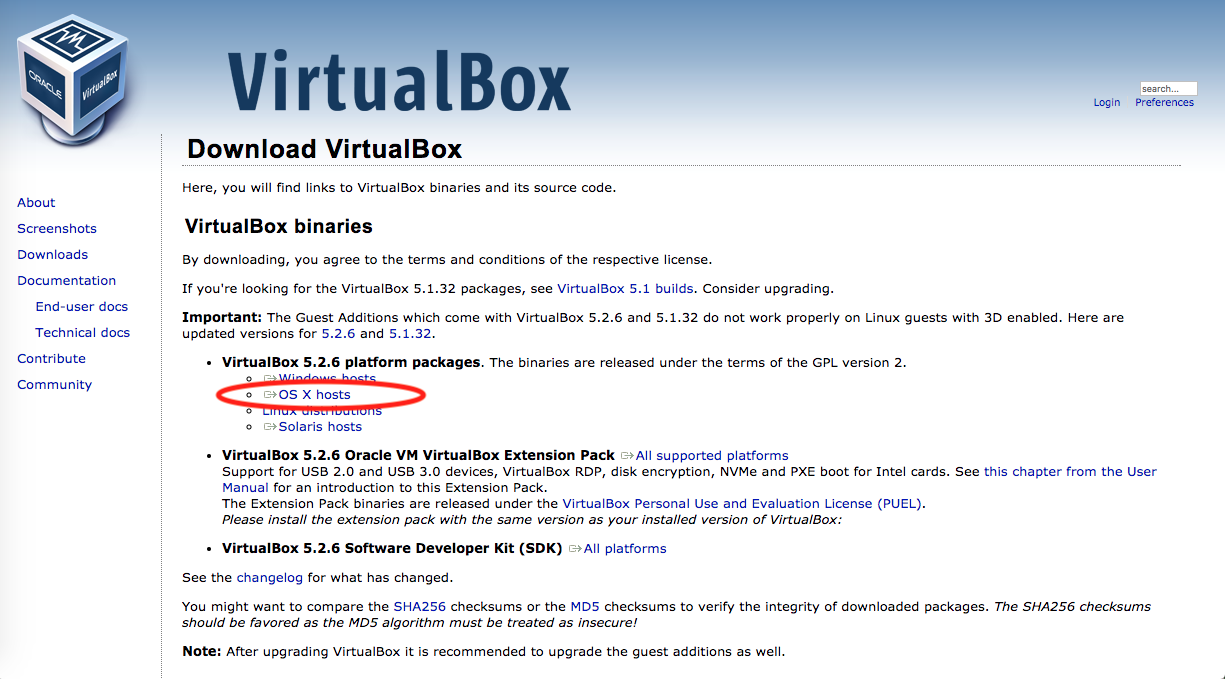
**BIG DATA Project**

Vinh Tuong Hoang – 986023

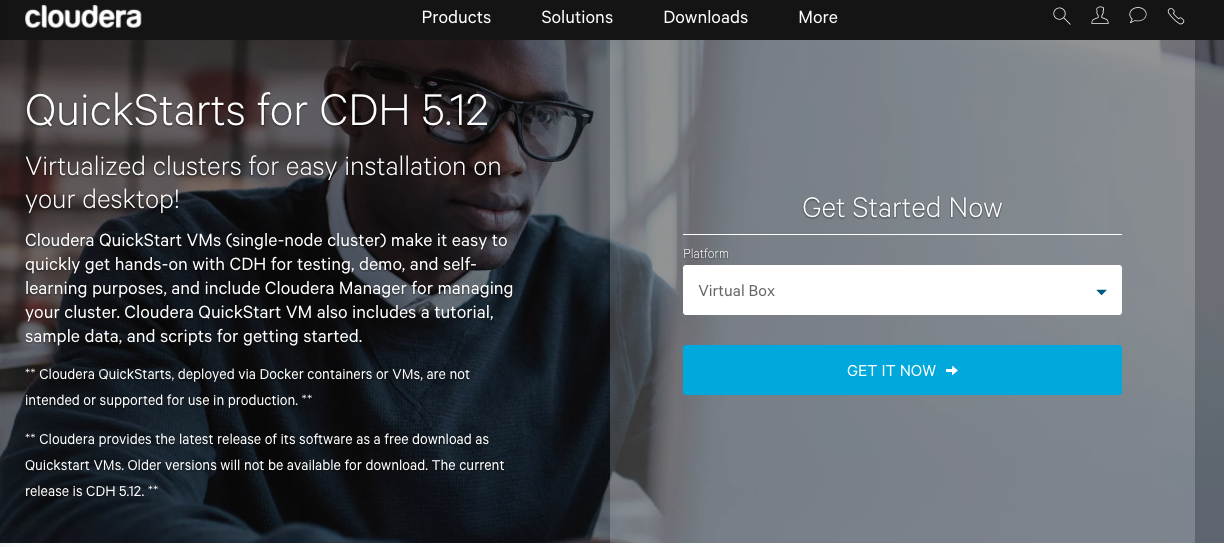
1. Part 1
2. Get and Install Virtual Box

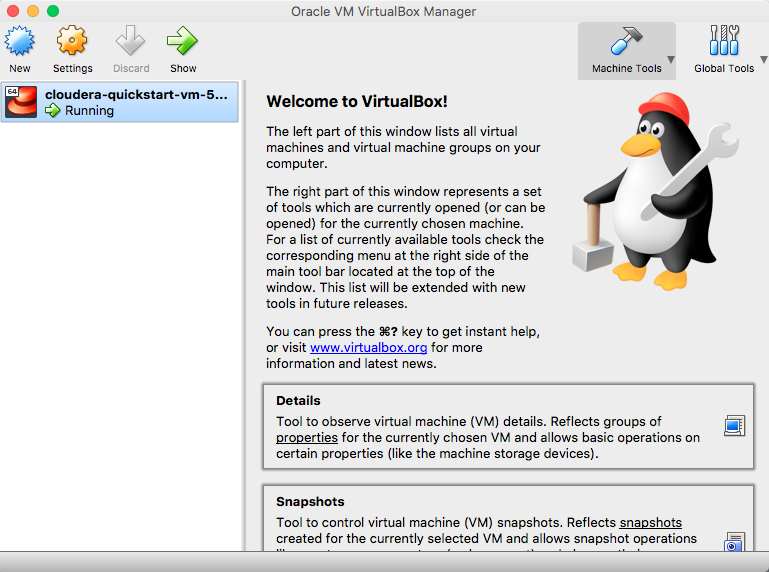
* Download the Mac OS X version for Virtual Box because I worked on my Mac
* Launch VirtualBox-5.2.6-120293-OSX.dmg to install.
* Download and install Virtual Box Extension Pack



1. Install Cloudera

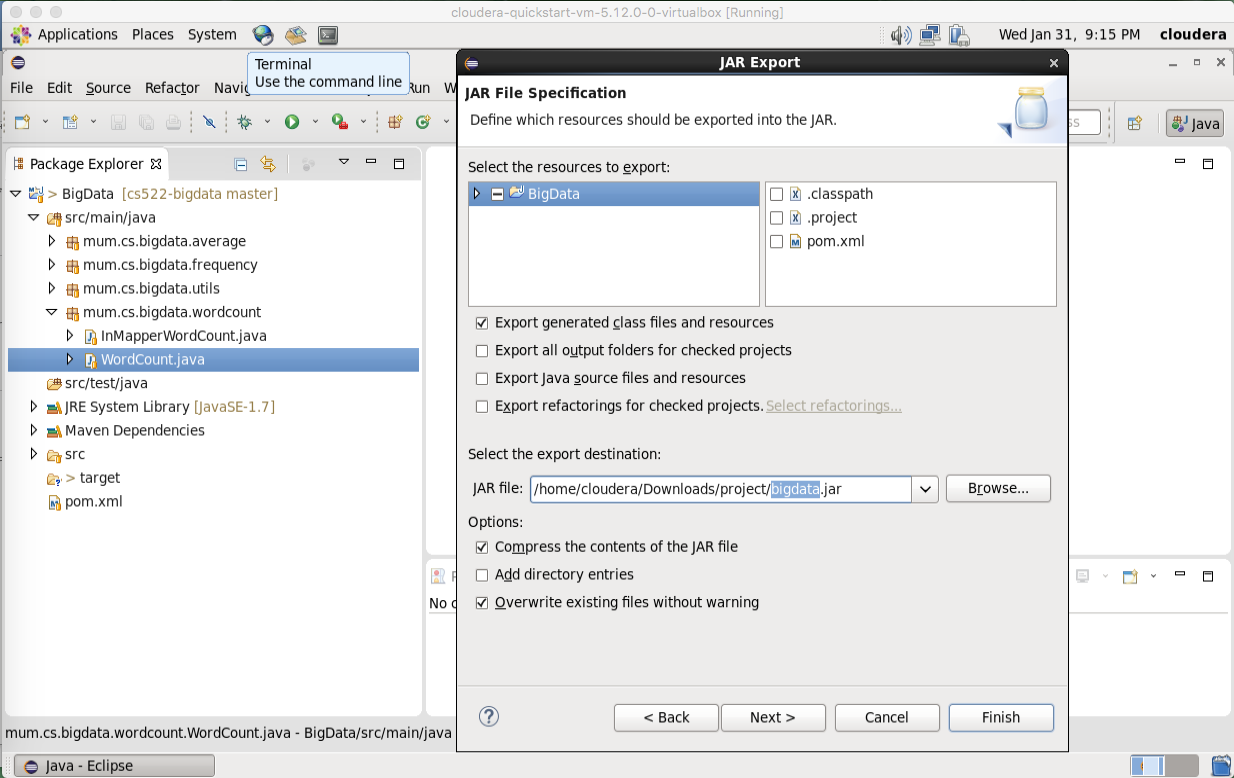
* Go to <https://www.cloudera.com/downloads/quickstart_vms/5-12.html>
* Choose “Virtual Box” platform and download (as in image)
* Extract the downloaded file
* Launch Virtual Box, and choose “File -> Import Alliance”. Browse to file “cloudera-quickstart-vm-5.12.0-0-virtualbox.ovf”, then click “Next” button to process import.
* After successfully imported, there will be a cloudera virtual machine.
* Start the virtual machine



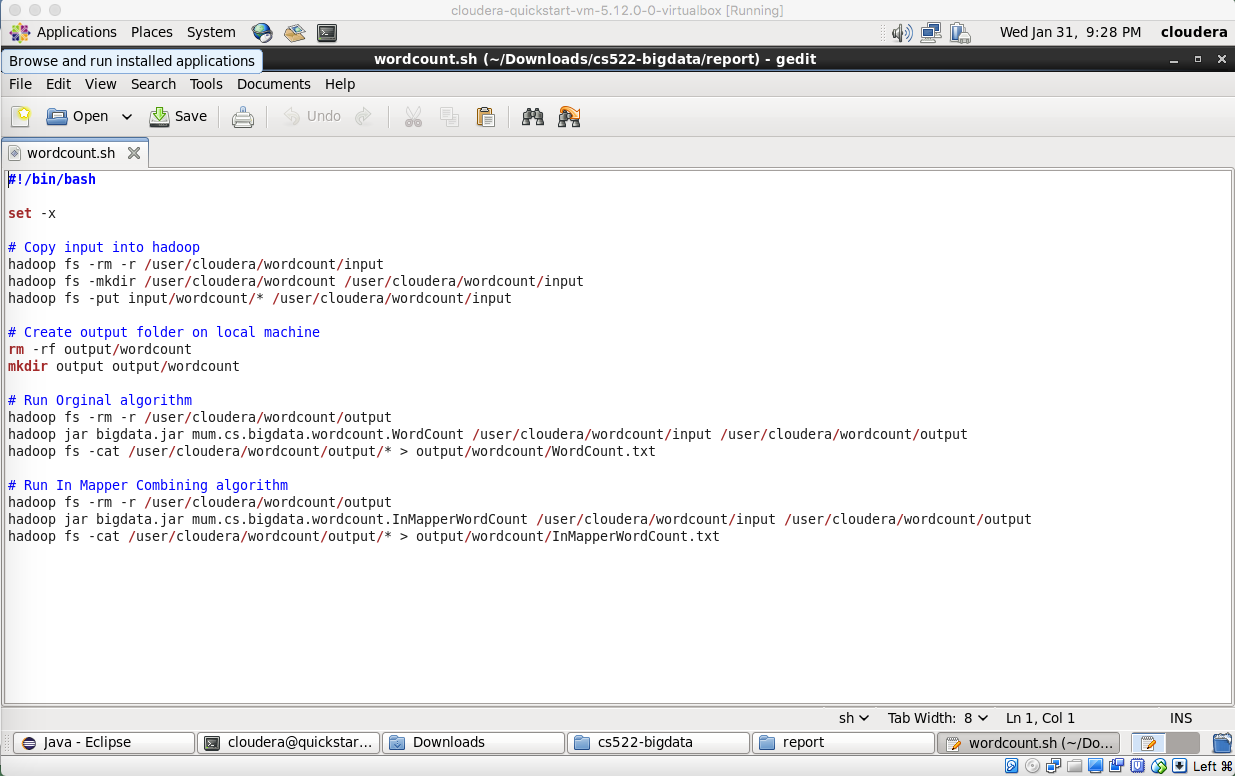


1. Get WordCount example (test and run)

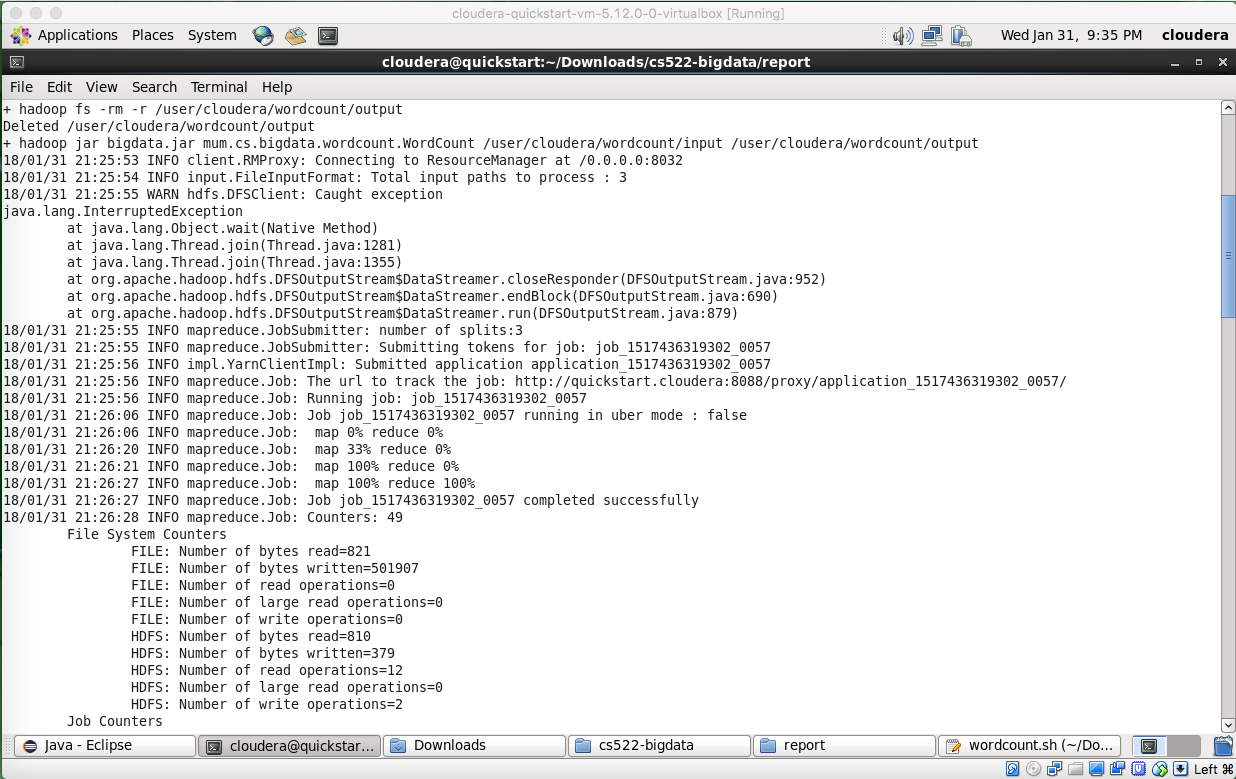
* Inside the virtual machine, open Eclipse and create new project, name “BigData”
* Right click on the project, choose “Configure -> Convert to Maven Project”
* Edit the pom.xml file to add the two following dependencies:
  + <http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common/2.7.3>
  + <http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core/2.7.3>
* Get sample source code from [**https://wiki.apache.org/hadoop/WordCount**](https://wiki.apache.org/hadoop/WordCount)
* Export the project to bigdata.jar as follow
  + Right click on the project, choose “Export”. In the pop up dialog, choose “Jar file” then click button “Next”
  + Choose the location and the name as “bigdata.jar”



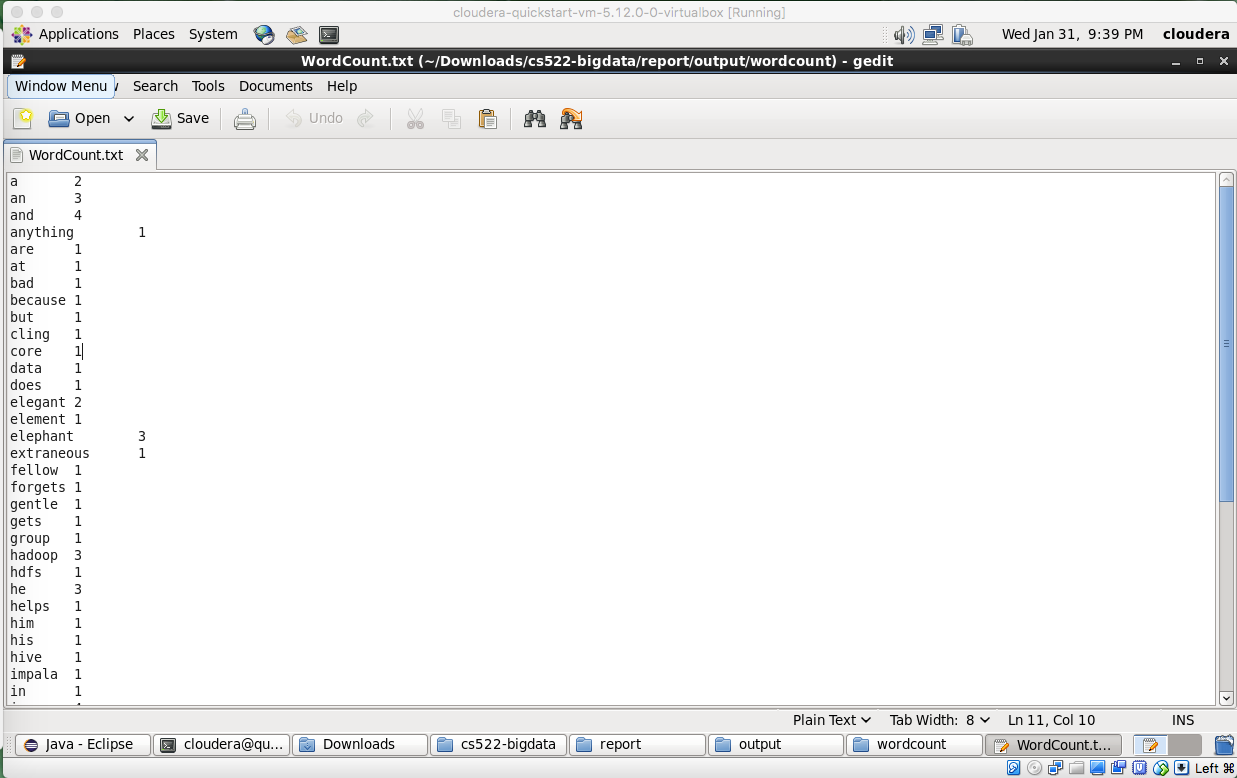
* Open “Terminal app”, create working folder on HDFS, as follow:
* $ sudo su hdfs
* $ hadoop fs -mkdir /user/cloudera
* $ hadoop fs -chown cloudera /user/cloudera
* $ exit
* $ sudo su cloudera
* In “Terminal”, run “wordcount.sh”. See image for detail. Basically, this script file do the following:
  + Copy input data from local to HDFS, located at /user/cloudera/wordcount/input. There are 3 input files, named as wordcount1, wordcount2, wordcount3 (See /input/wordcount/ folder for more detail).
  + Remove output folder on HDFS (if existed)
  + Run Hadoop with bigdata.jar and WordCount class (as in image)
  + The output then is copied from HDFS to local machine



* The console output of the execution is as in the image



* The output is now located in local machine at /output/wordcount

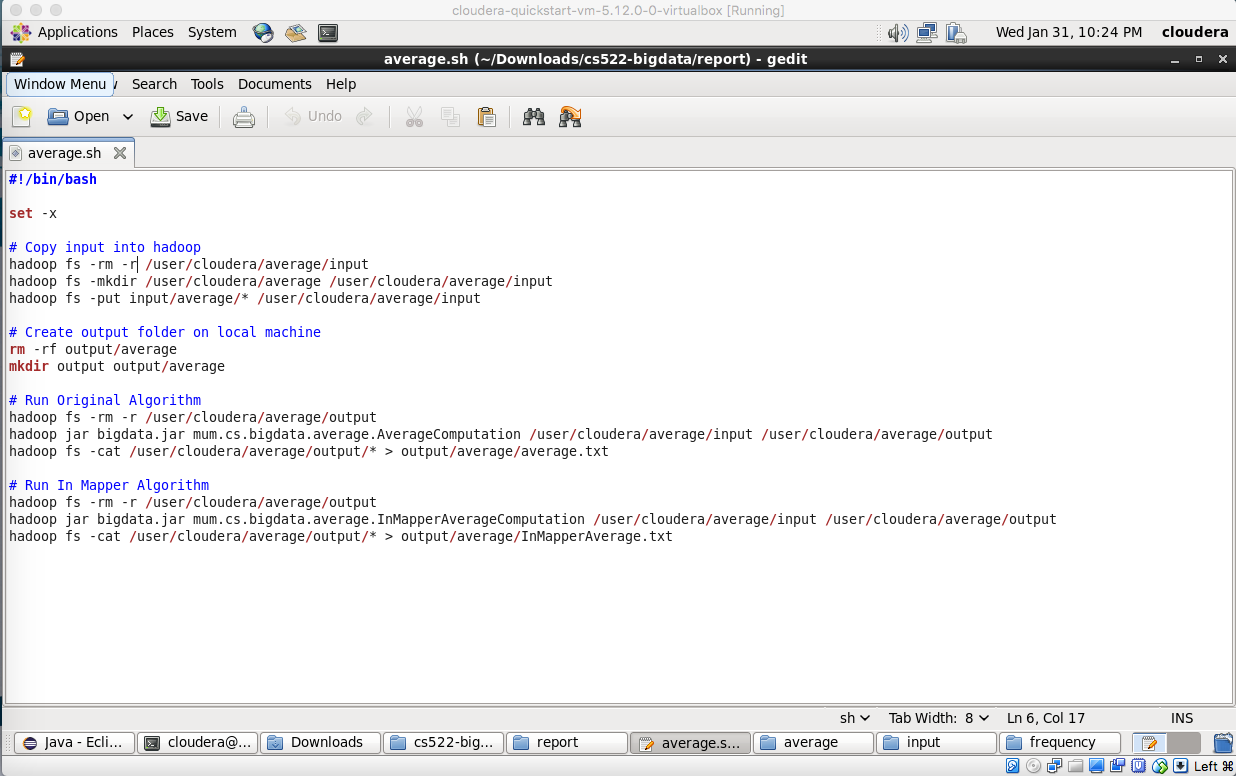


1. InMapperWordCount

* Java classes: See attaches source files (InMapperWordCount.java)
* Input files: Same as above. There are 3 files: wordcount1, wordcount2, wordcount3
* Output files: named as InMapperWordCount.txt, located at /output/wordcount/

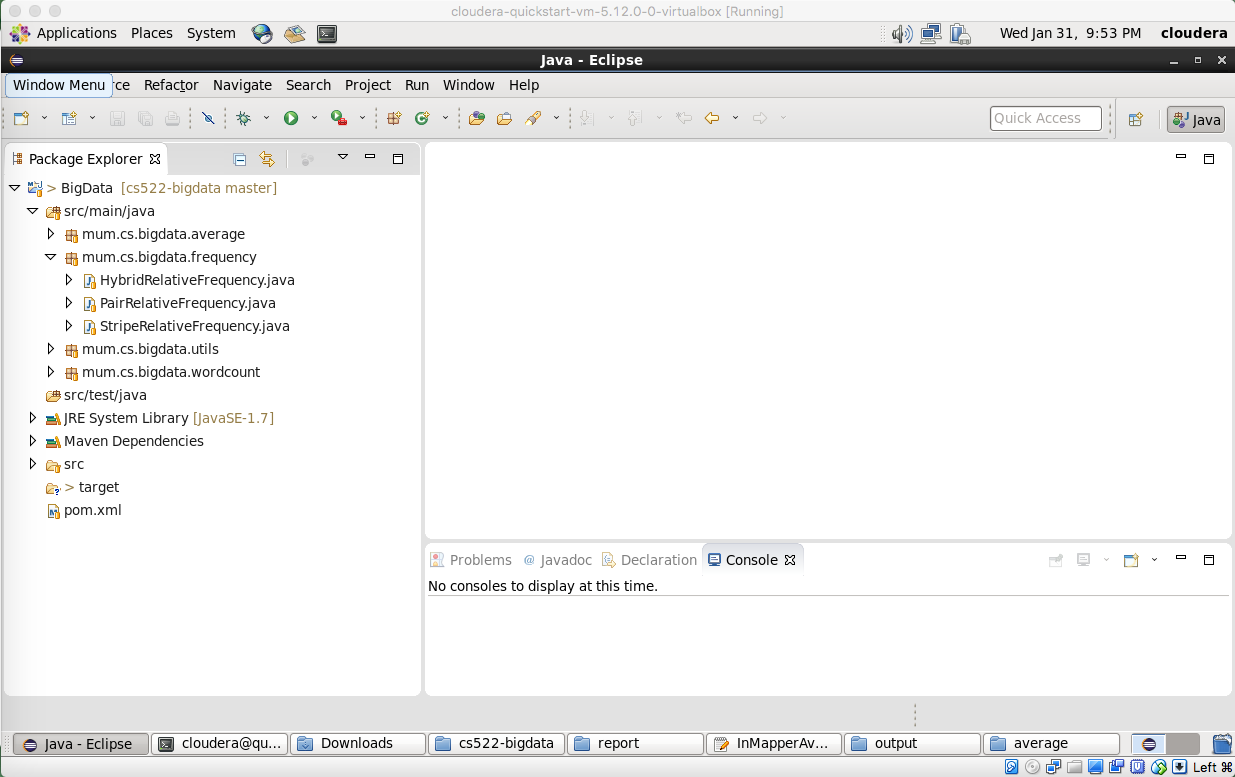
1. Average and In-mapper Average

* Java classes: See attached source files
  + AverageComputation.java: Implements average computation algorithms
  + InMapperAverageComputation.java: Implements in-mapper average computation algorithms.
* Input files are located at /input/average. There is one file access\_log (as provided)
* Output files are located at /output/average. There are 2 files average.txt and InMapperAverage.txt for the two approaches.
* In Terminal, run average.sh to run the two approaches with Hadoop. See the file for more detail.



1. Part 2, 3 & 4
2. Create java classes (See attached source files)

* PairRelativeFrequency.java: Implements the Pair approach algorithms
* StripeRelativeFrequency.java: Implements the Stripe approach algorithms
* HybridRelativeFrequency.java: Implements the Hybrid approach algorithms

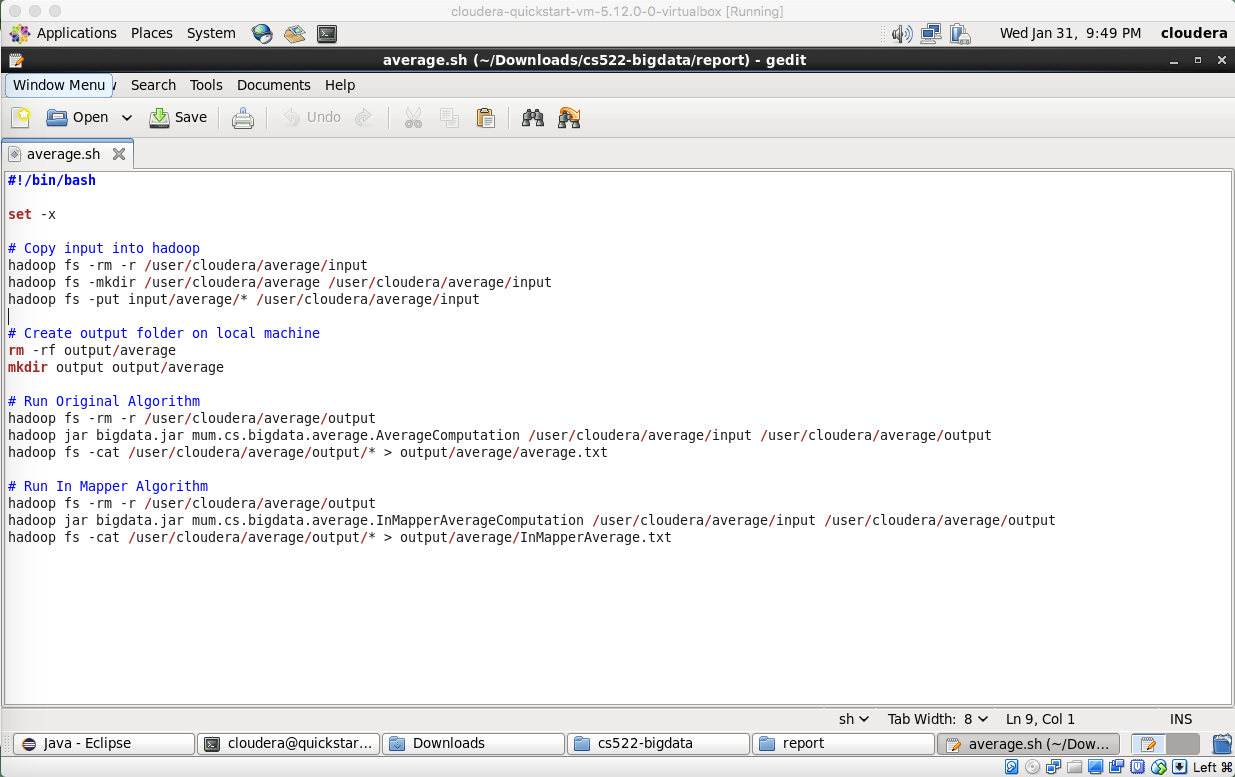


1. Input files:

* Input files are located at /input/frequency, include 2 files: frequency1, frequency2

|  |  |  |
| --- | --- | --- |
| Input File Name | frequency1 | frequency2 |
| Content | 15 91 80 12 19 80 18  17 15 80 18 19 18 | 19 15 80 18 19 18  18 15 18 18 80 18 |

1. Batch file: Run “frequency.sh” in Terminal. See average.sh for more detail.



1. Output is located at /output/frequency/. There are 3 files, stripe.txt, pair.txt and hybrid.txt for the three approaches

|  |  |
| --- | --- |
| **File** | **Content** |
| pair.txt | (12, 18) 0.3333333333333333  (12, 19) 0.3333333333333333  (12, 80) 0.3333333333333333  (15, 12) 0.05555555555555555  (15, 18) 0.4444444444444444  (15, 19) 0.16666666666666666  (15, 80) 0.2777777777777778  (15, 91) 0.05555555555555555  (17, 15) 0.2  (17, 18) 0.4  (17, 19) 0.2  (17, 80) 0.2  (18, 15) 0.25  (18, 19) 0.5  (18, 80) 0.25  (19, 15) 0.14285714285714285  (19, 18) 0.5714285714285714  (19, 80) 0.2857142857142857  (80, 12) 0.1  (80, 18) 0.6  (80, 19) 0.3  (91, 12) 0.2  (91, 18) 0.2  (91, 19) 0.2  (91, 80) 0.4 |
| stripe.txt | 12 [(19, 0.3333333333333333)(18, 0.3333333333333333)(80, 0.3333333333333333)]  15 [(91, 0.05555555555555555)(19, 0.16666666666666666)(18, 0.4444444444444444)(80, 0.2777777777777778)(12, 0.05555555555555555)]  17 [(19, 0.2)(18, 0.4)(80, 0.2)(15, 0.2)]  18 [(19, 0.5)(80, 0.25)(15, 0.25)]  19 [(18, 0.5714285714285714)(80, 0.2857142857142857)(15, 0.14285714285714285)]  80 [(19, 0.3)(18, 0.6)(12, 0.1)]  91 [(19, 0.2)(18, 0.2)(80, 0.4)(12, 0.2)] |
| hybrid.txt | 12 [(19, 0.3333333333333333)(18, 0.3333333333333333)(80, 0.3333333333333333)]  15 [(91, 0.05555555555555555)(19, 0.16666666666666666)(18, 0.4444444444444444)(80, 0.2777777777777778)(12, 0.05555555555555555)]  17 [(19, 0.2)(18, 0.4)(80, 0.2)(15, 0.2)]  18 [(19, 0.5)(80, 0.25)(15, 0.25)]  19 [(18, 0.5714285714285714)(80, 0.2857142857142857)(15, 0.14285714285714285)]  80 [(19, 0.3)(18, 0.6)(12, 0.1)]  91 [(19, 0.2)(18, 0.2)(80, 0.4)(12, 0.2)] |