**Aim** : Using Pig Tool and creating Pig Latin Scripts

**CREATING AND EXECUTING PIG LATIN SCRIPT**

**What is Pig in Hadoop?**

Pig is a scripting platform that runs on Hadoop clusters designed to process and analyze large datasets. Pig is extensible, self-optimizing, and easily programmed.

Programmers can use Pig to write data transformations without knowing Java. Pig uses both structured and unstructured data as input to perform analytics and uses HDFS to store the results.

**Components of Pig**

There are two major components of the Pig:

● Pig Latin script language

● A runtime engine

**Pig Latin script language:**

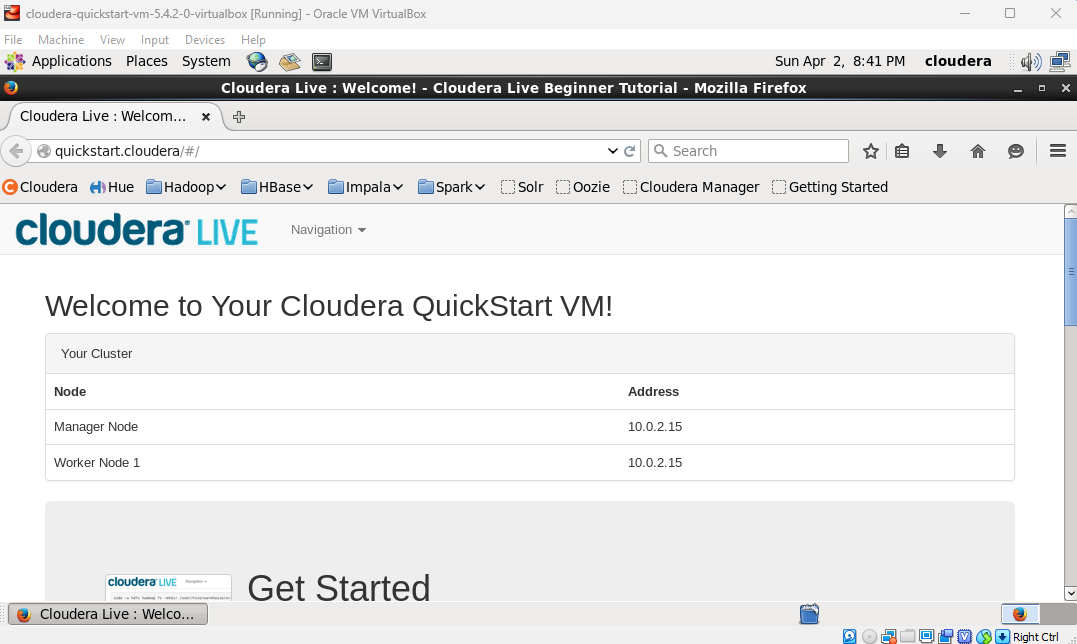
The Pig Latin script is a procedural data flow language. It contains syntax and commands that can be applied to implement business logic. Examples of Pig Latin are LOAD and STORE.

**A runtime engine:**

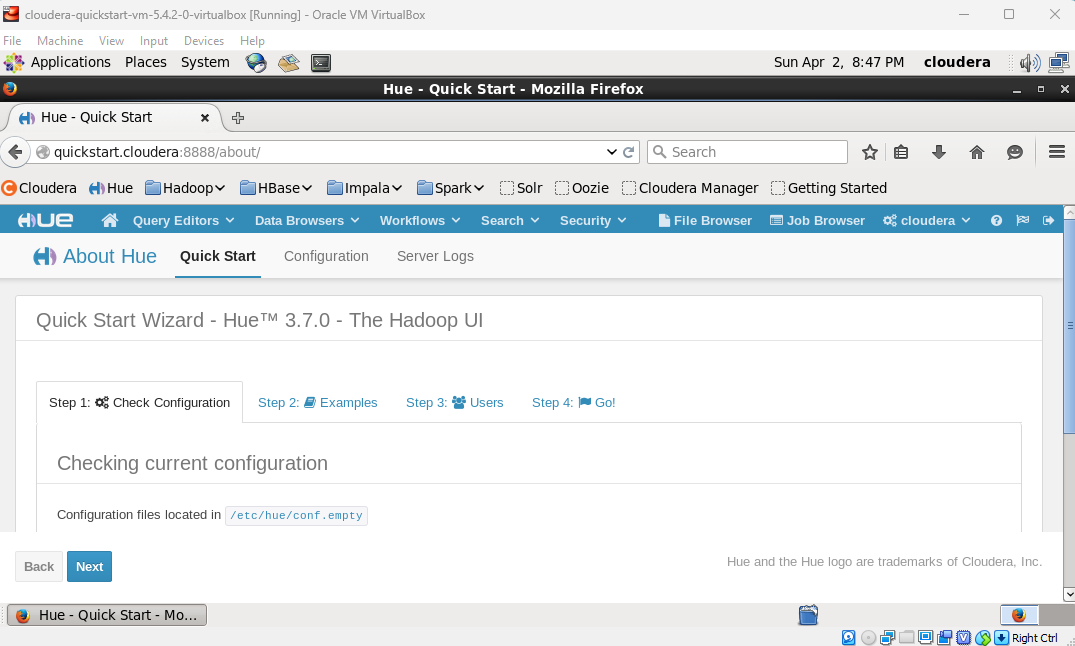
The runtime engine is a compiler that produces sequences of MapReduce programs. It uses HDFS to store and retrieve data. It is also used to interact with the Hadoop system (HDFS and MapReduce).

The runtime engine parses, validates, and compiles the script operations into a sequence of MapReduce jobs.

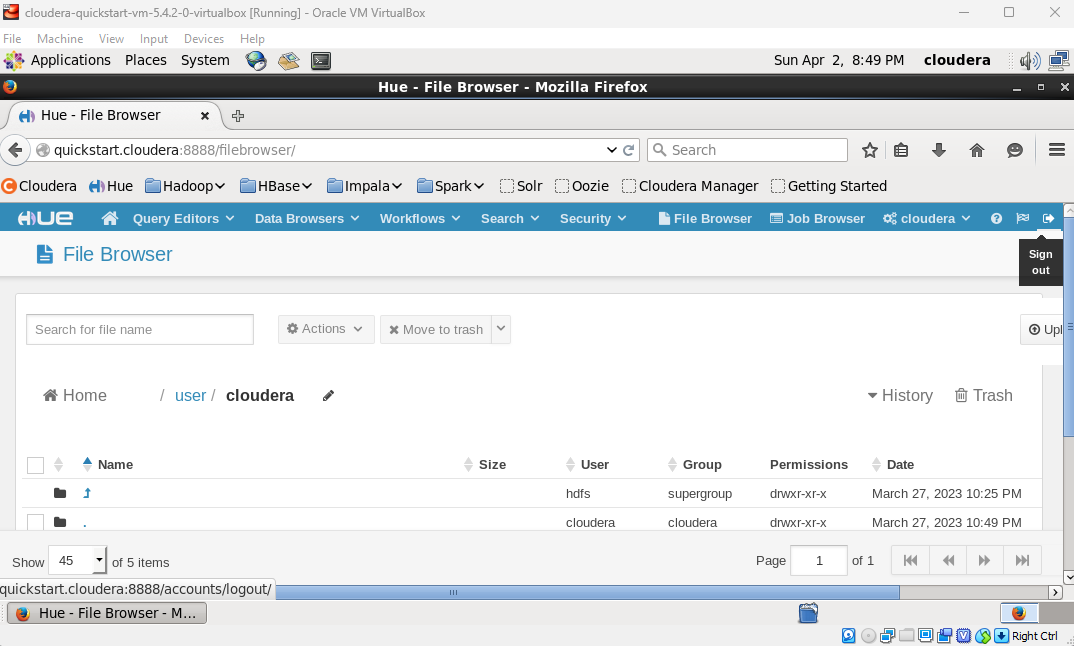
* **Open cloudera browser**

****

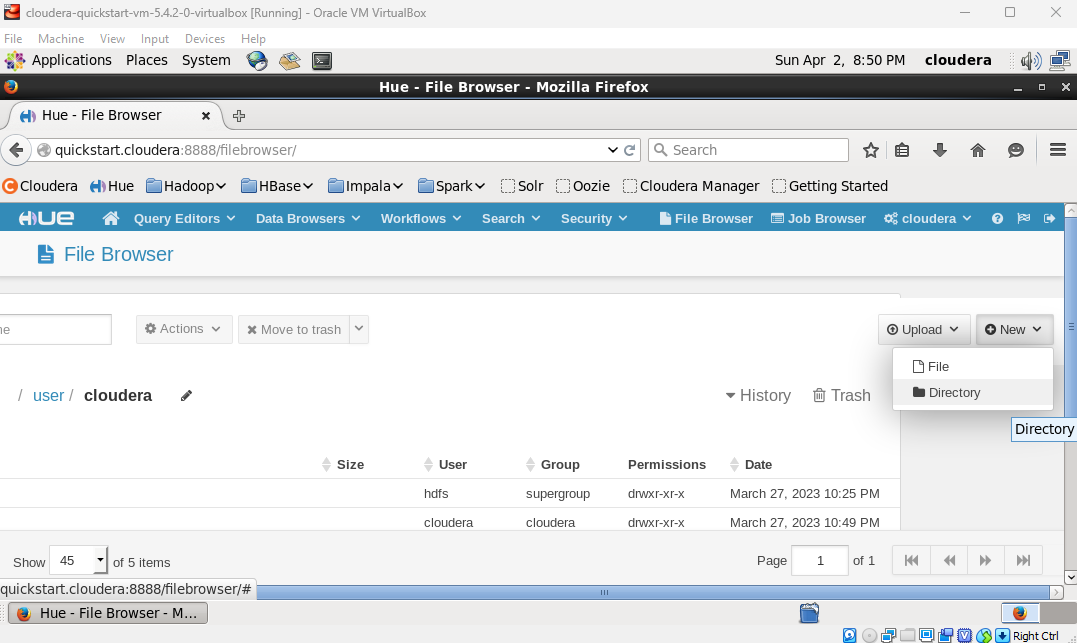
* **Locate Hue and get logged in with username cloudera and password cloudera.**

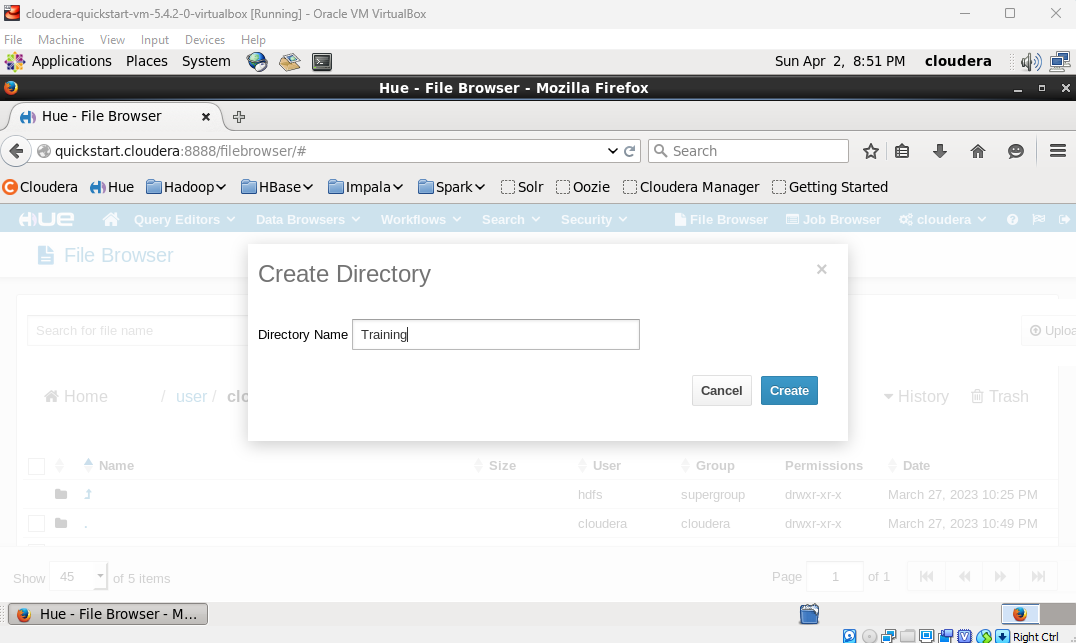
****

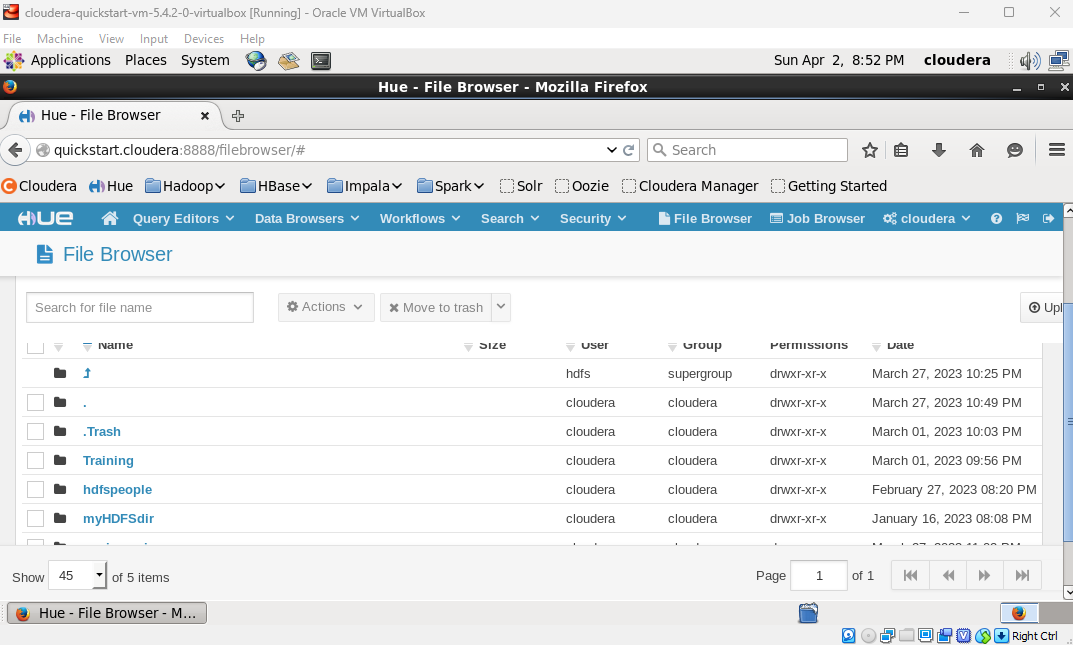
* **Locate file Browser and check for directory user/cloudera.**

****

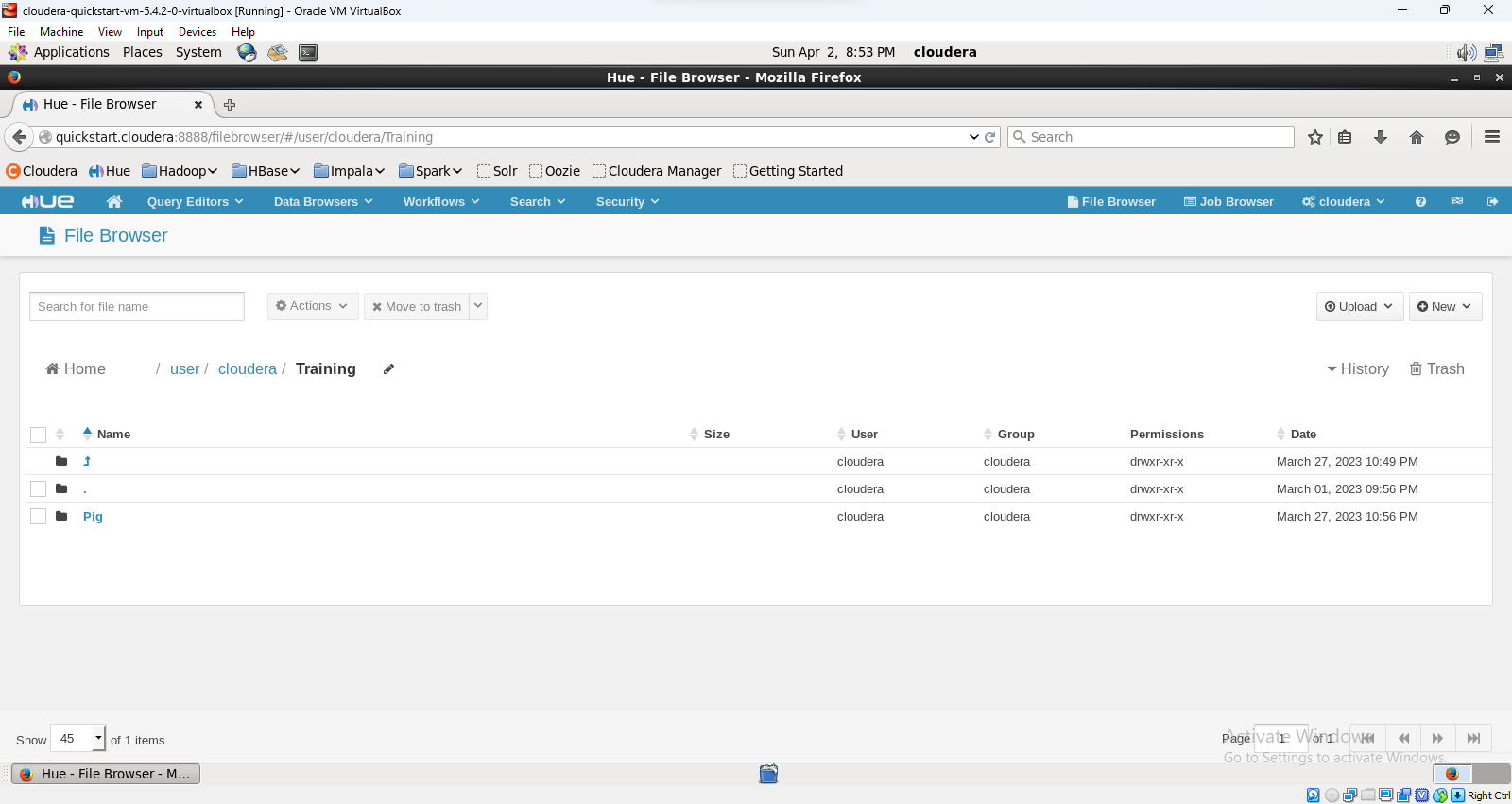
* **Create a new directory named ‘training’.**

****

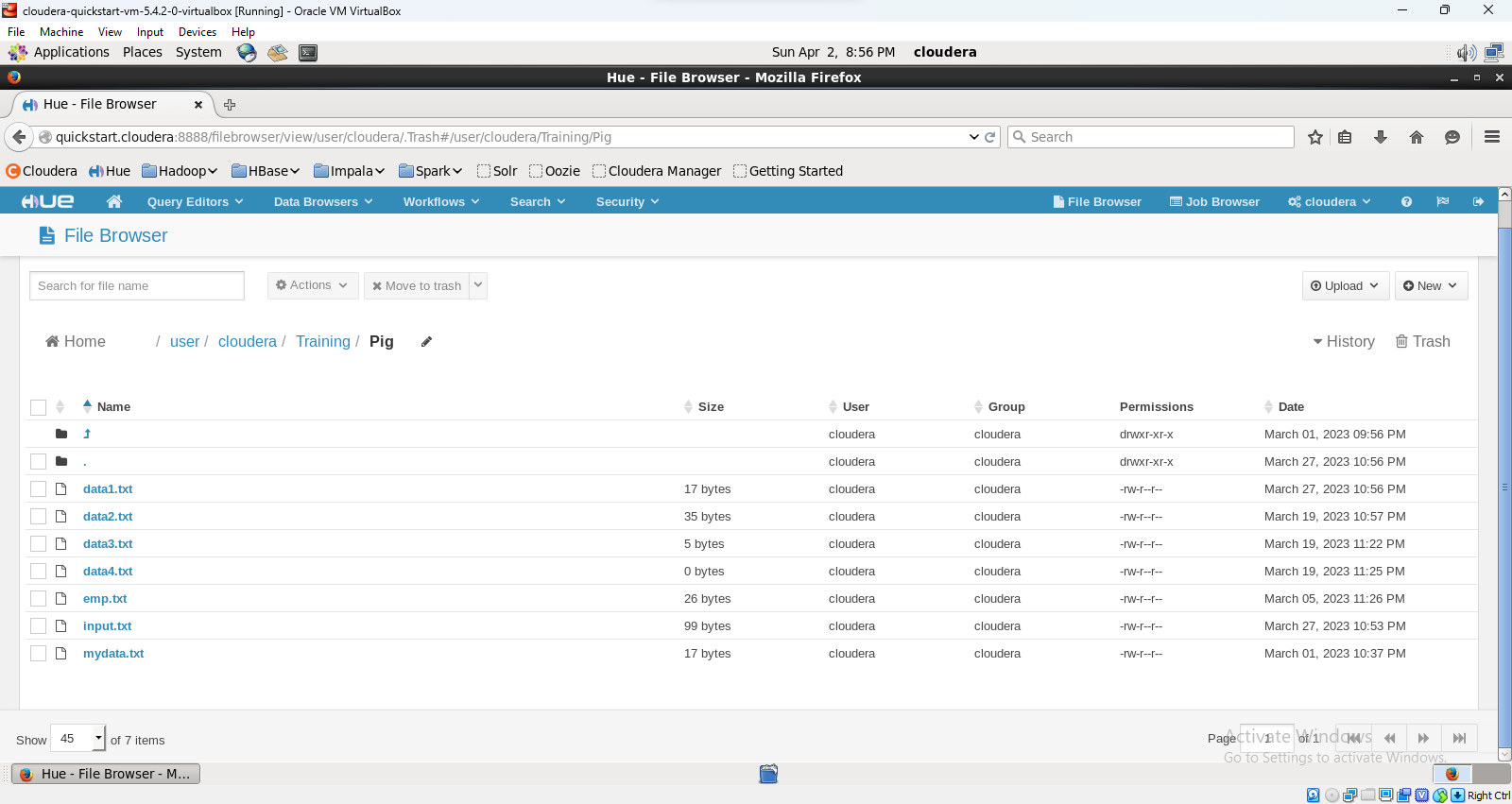
****

****

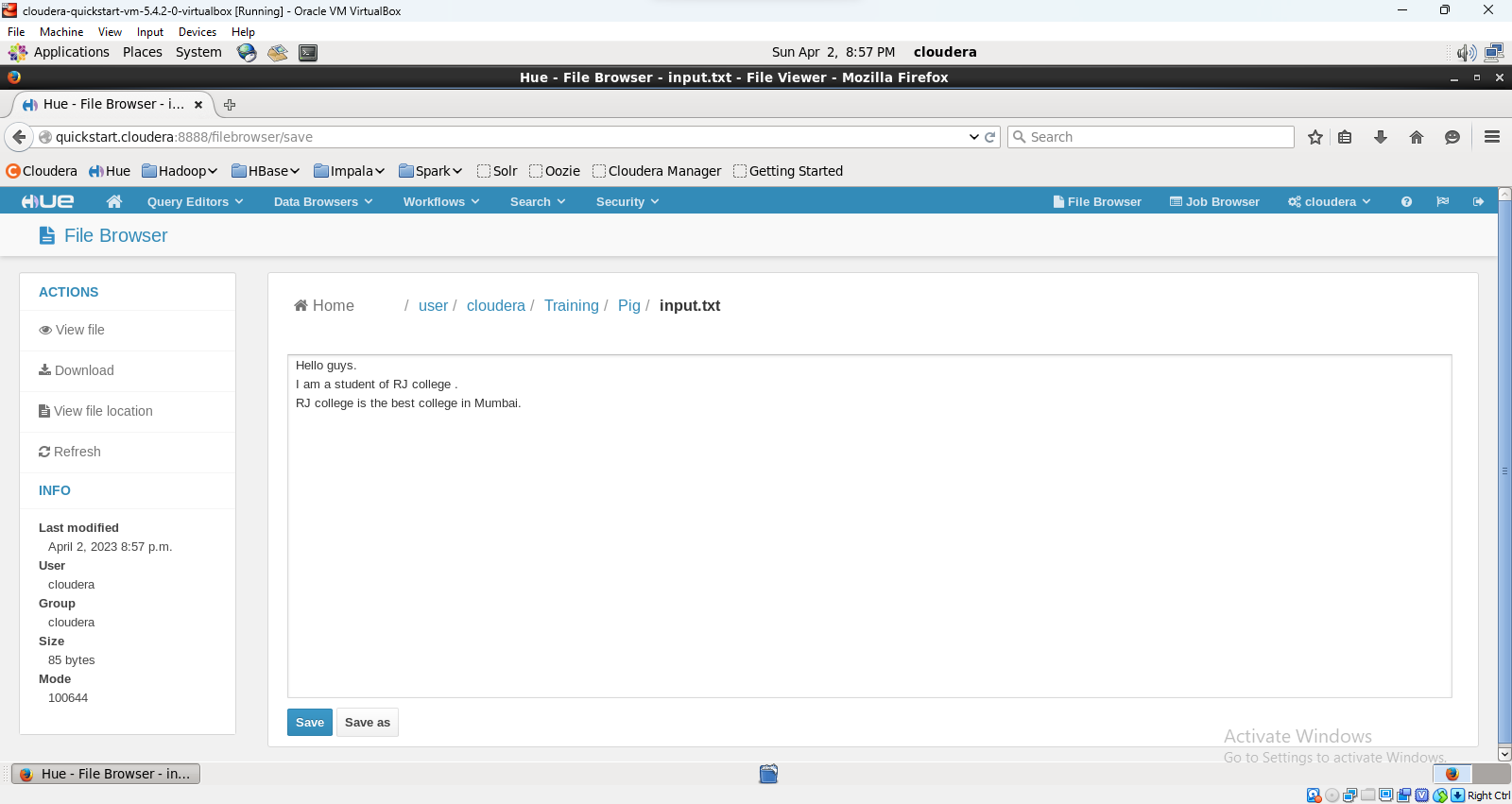
* **Create the pig directory inside the training directory.**

****

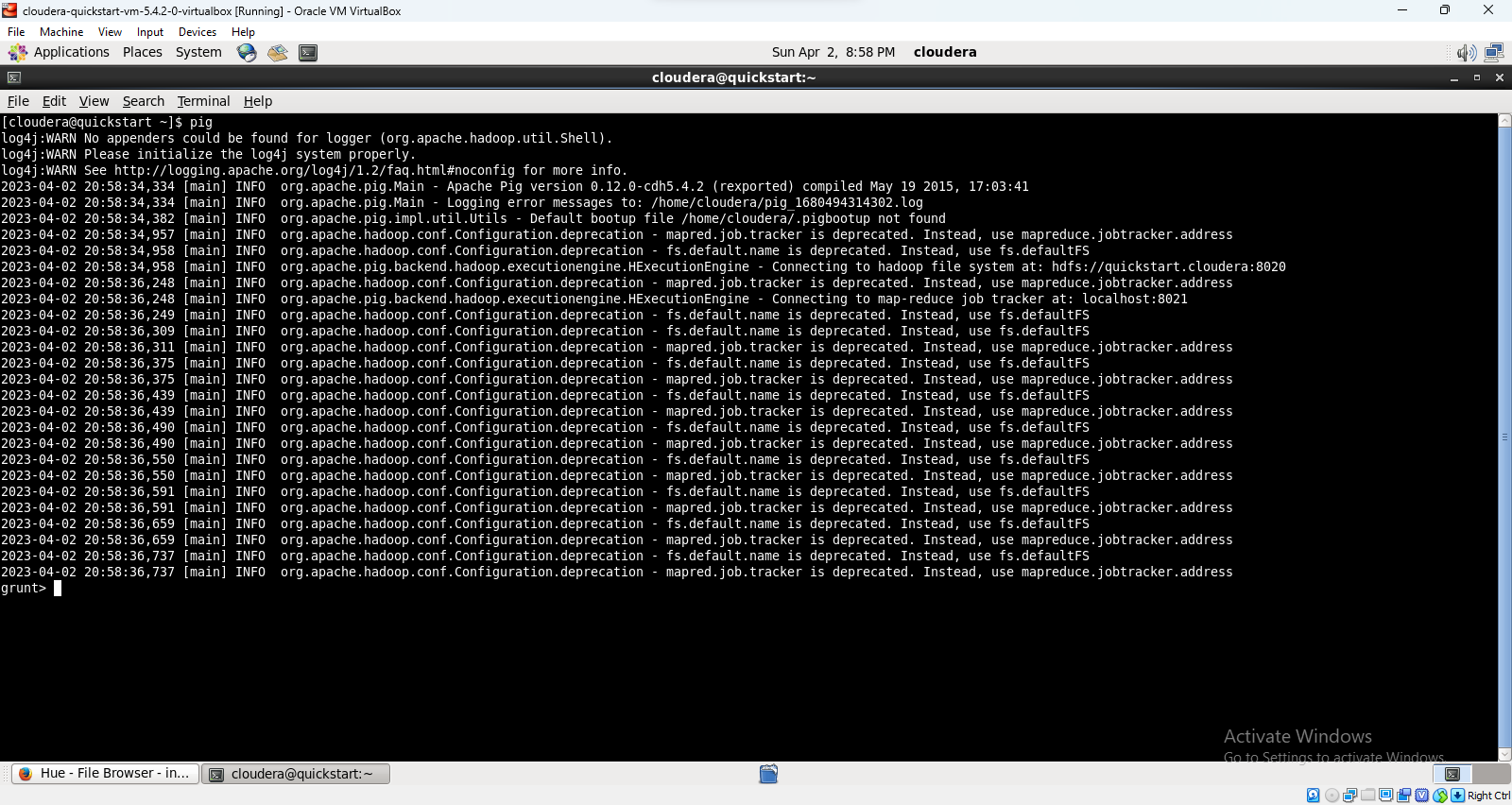
* **Create an input.txt file in the pig directory.**

****

* **Add some text into the input.txt and save the file.**

****

* **Open terminal and start pig tool by executing pig command.**

****

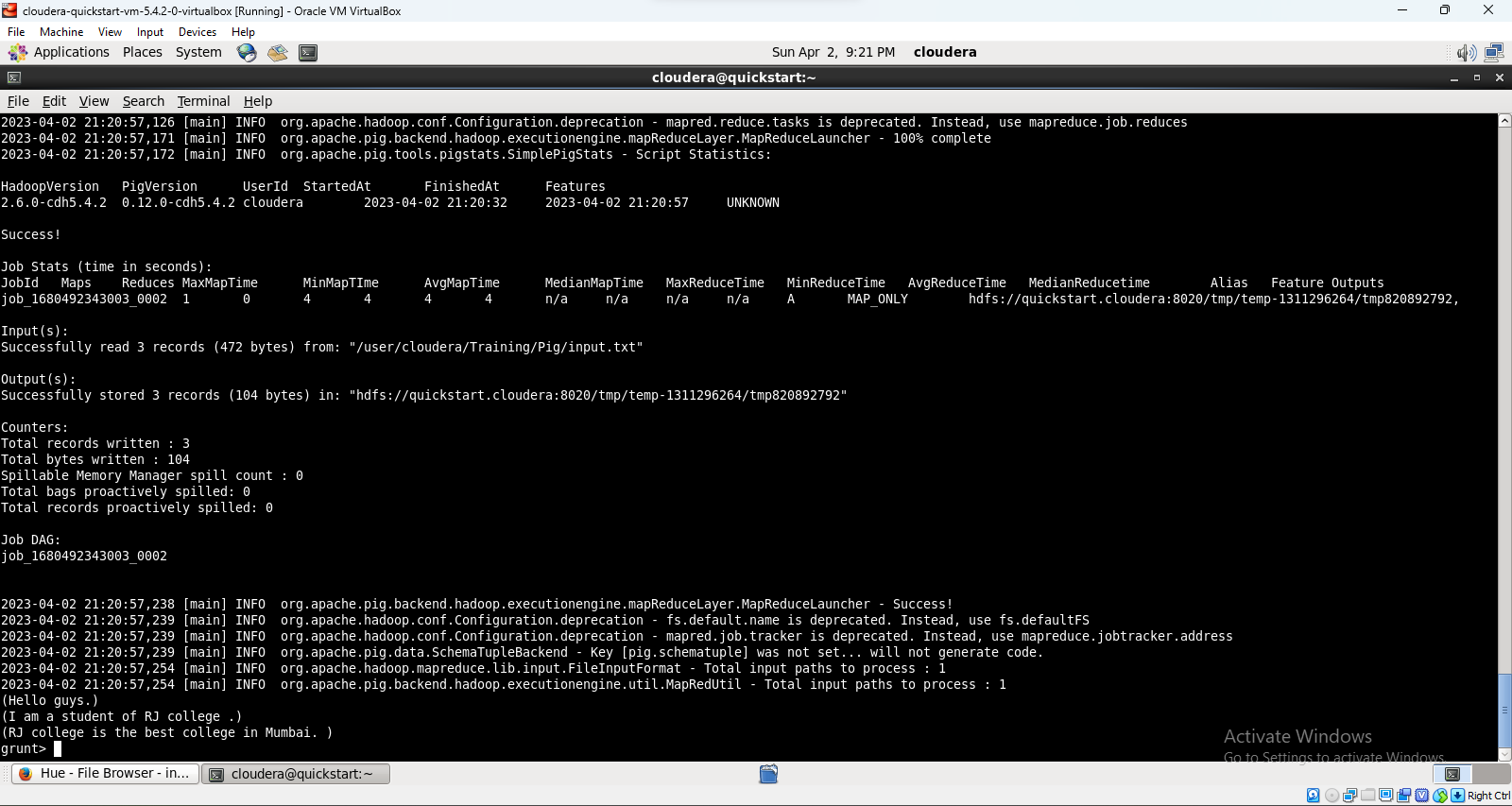
* **Load input.txt file in input variable.**

A = LOAD ‘/user/cloudera/Training/pig/input.txt’ AS (f1:chararray);

****

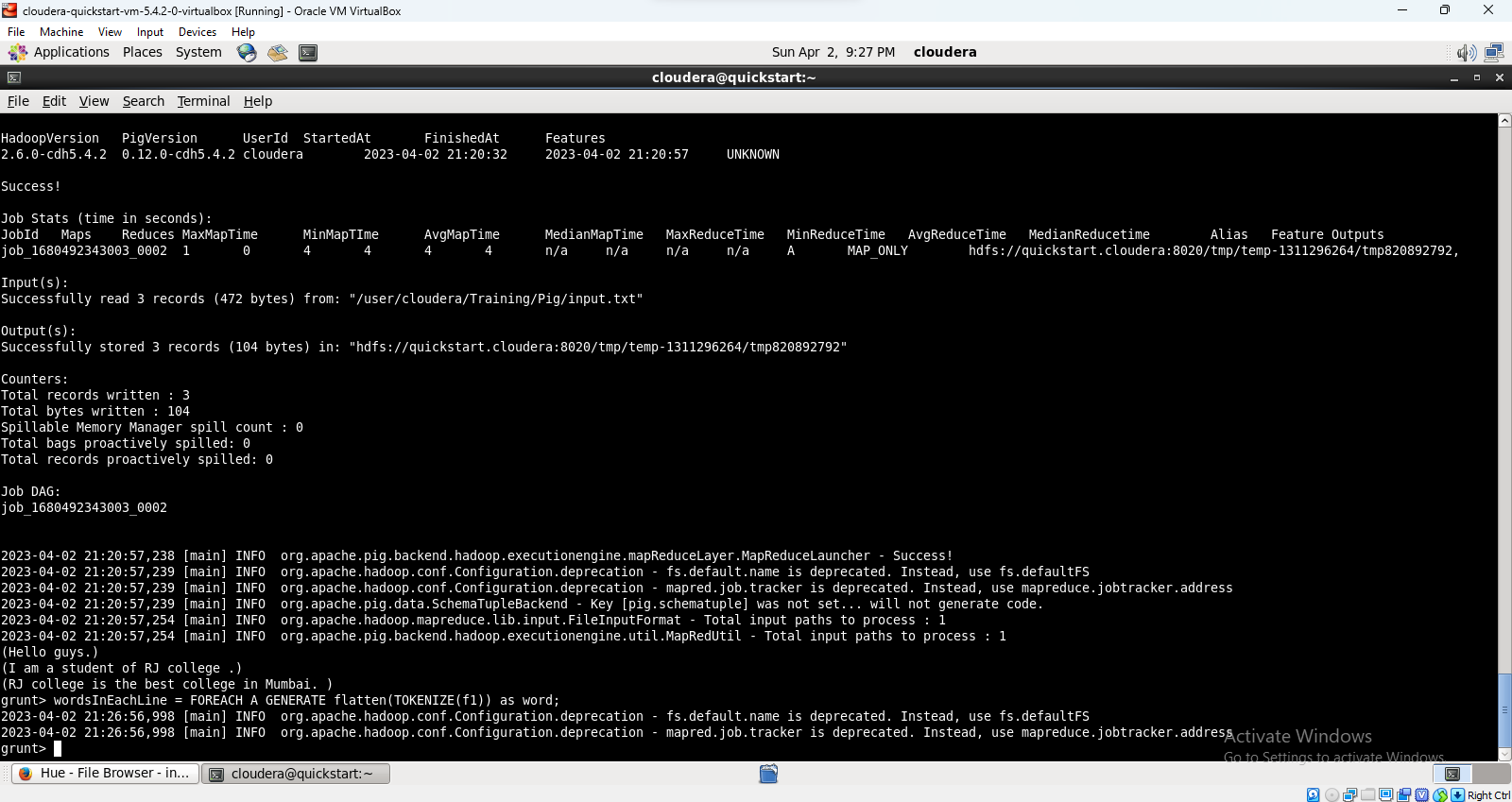
* **Display the contents of the input variable.**

**DUMP A**

****

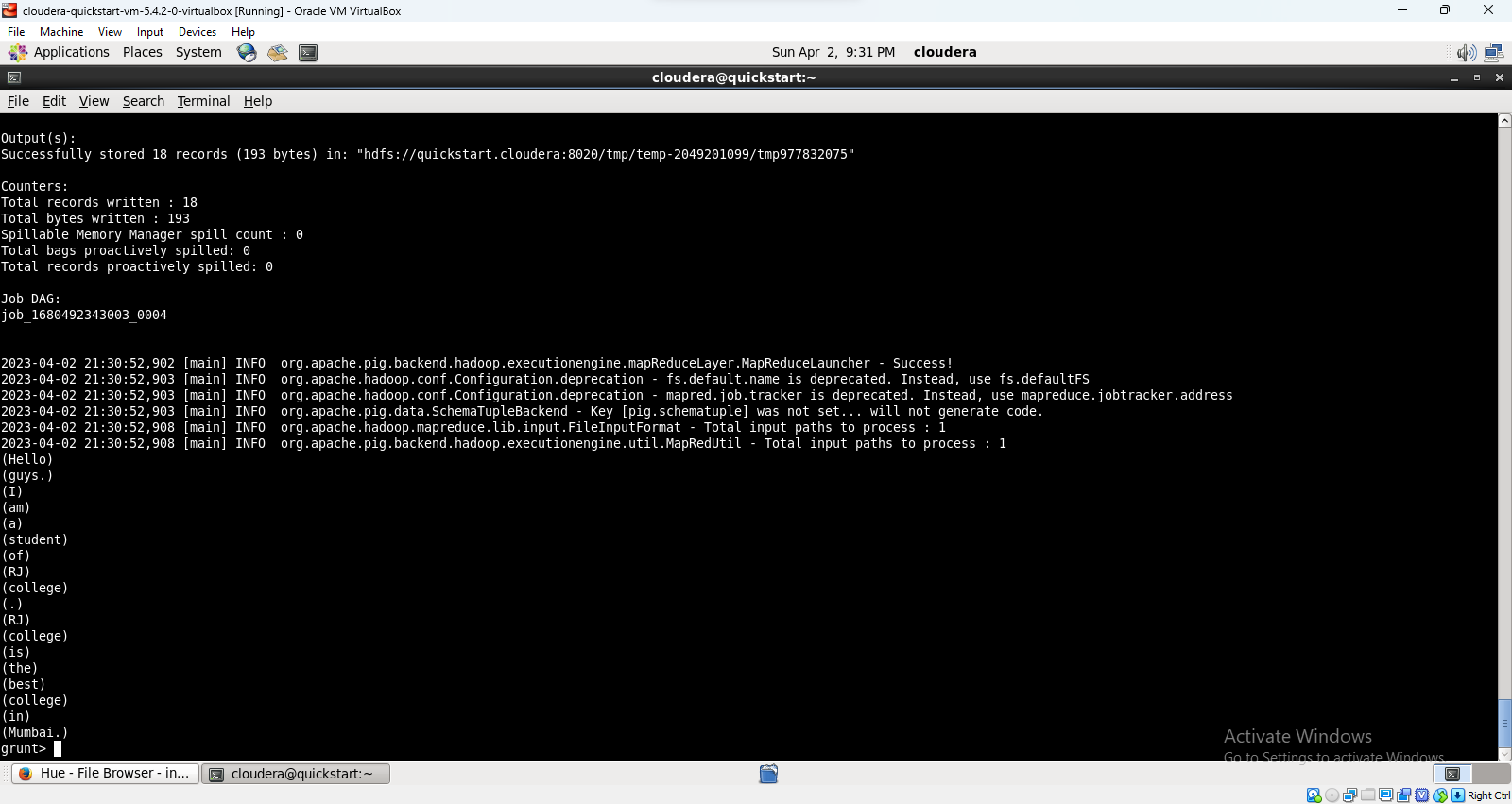
* Tokenize the text that is stored in variable input.

wordsInEachLine = FOREACH A GENERATE flatten(TOKENIZE(f1)) as word;



dump wordsInEachLine;





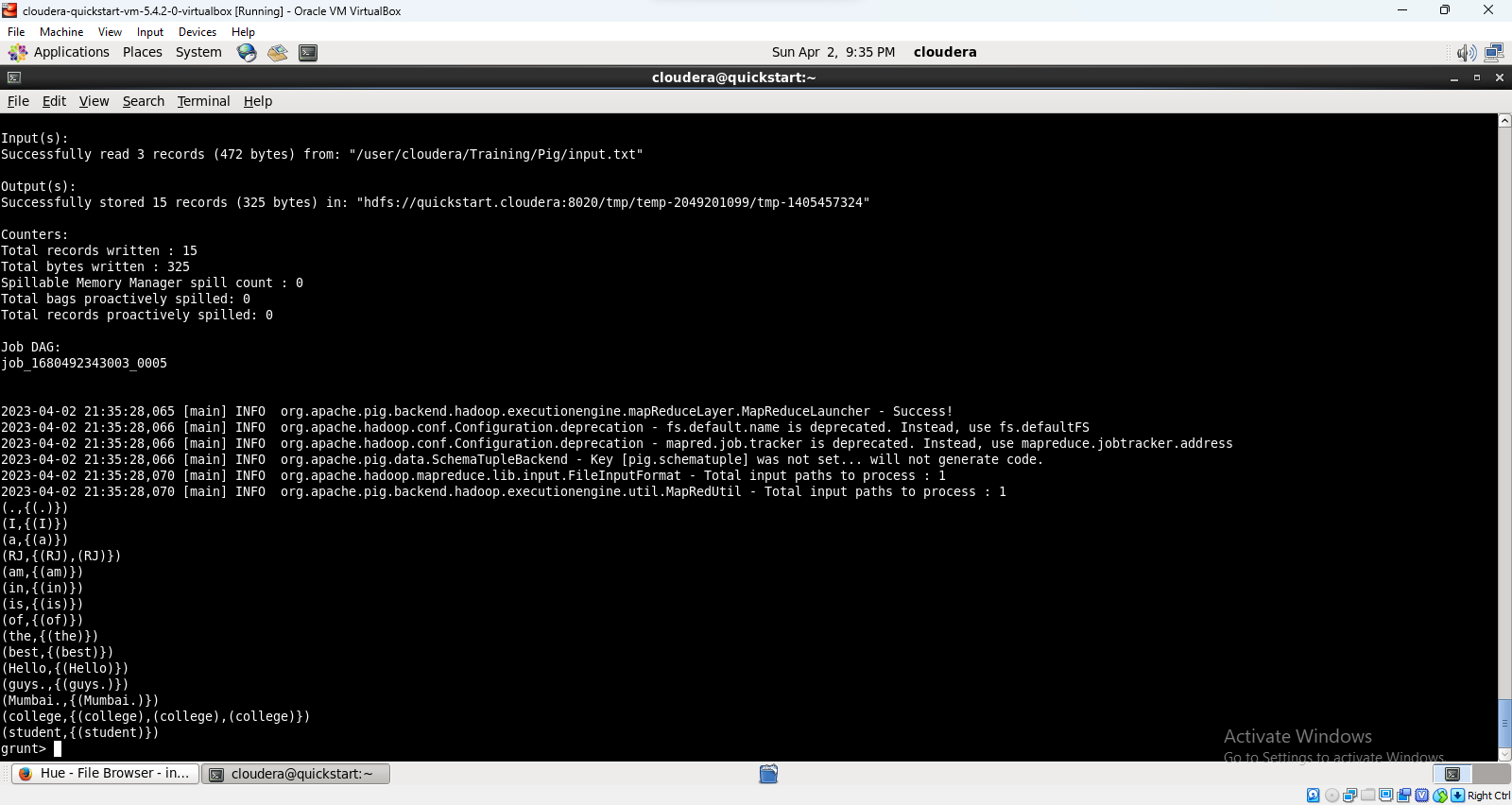
* Group all similar words.

groupedWords = group wordsInEachLine by word;



dump groupedWords;





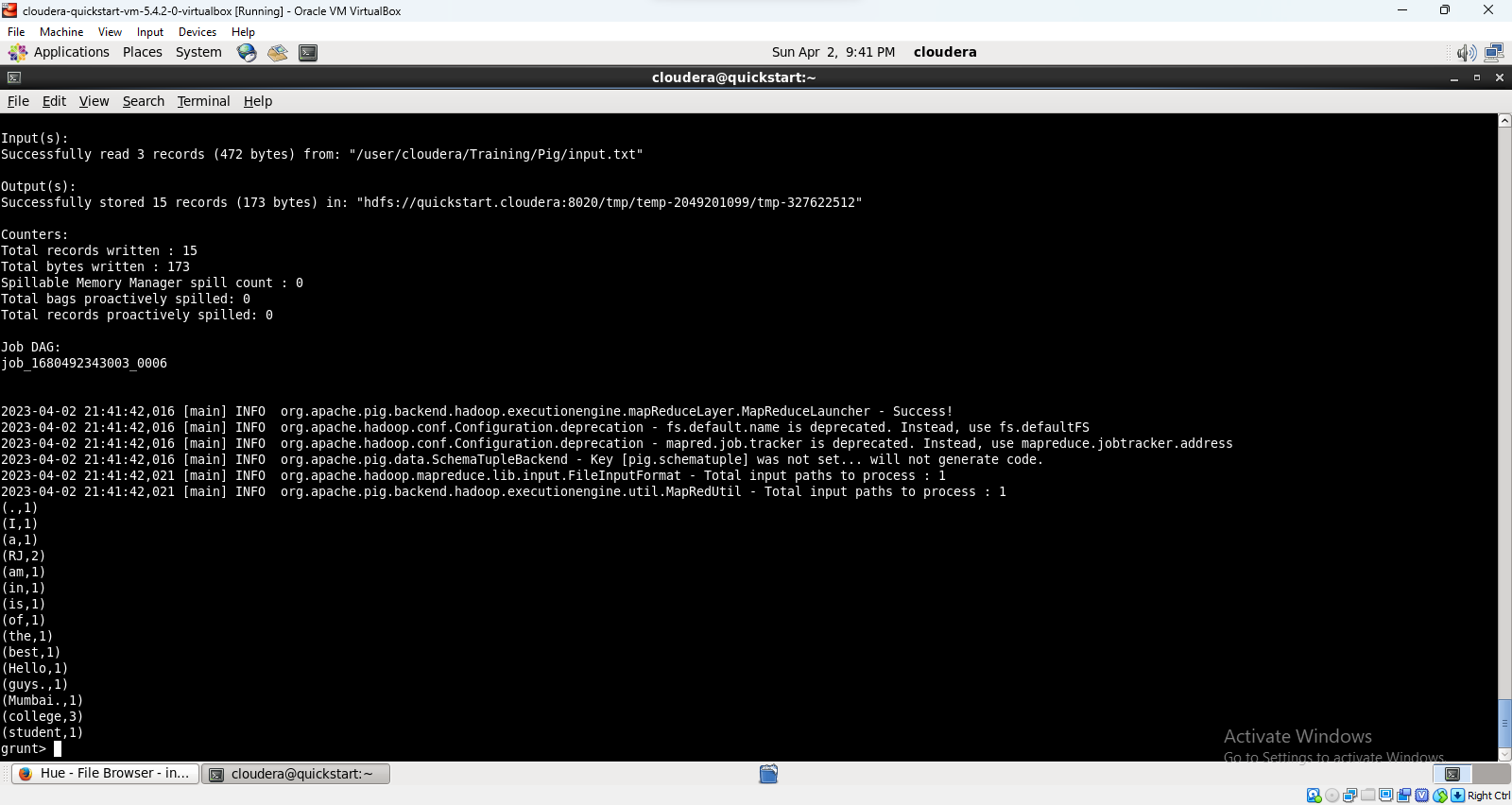
* Count the number of occurrences of each word.

countedWords = FOREACH groupedWords GENERATE group, COUNT(wordsInEachLine);



dump countedWords;





**Creating pig script**

/\*

Wordcountex.pig

Counting the occurrences of word

\*/

–Execute this script in mapreduce mode

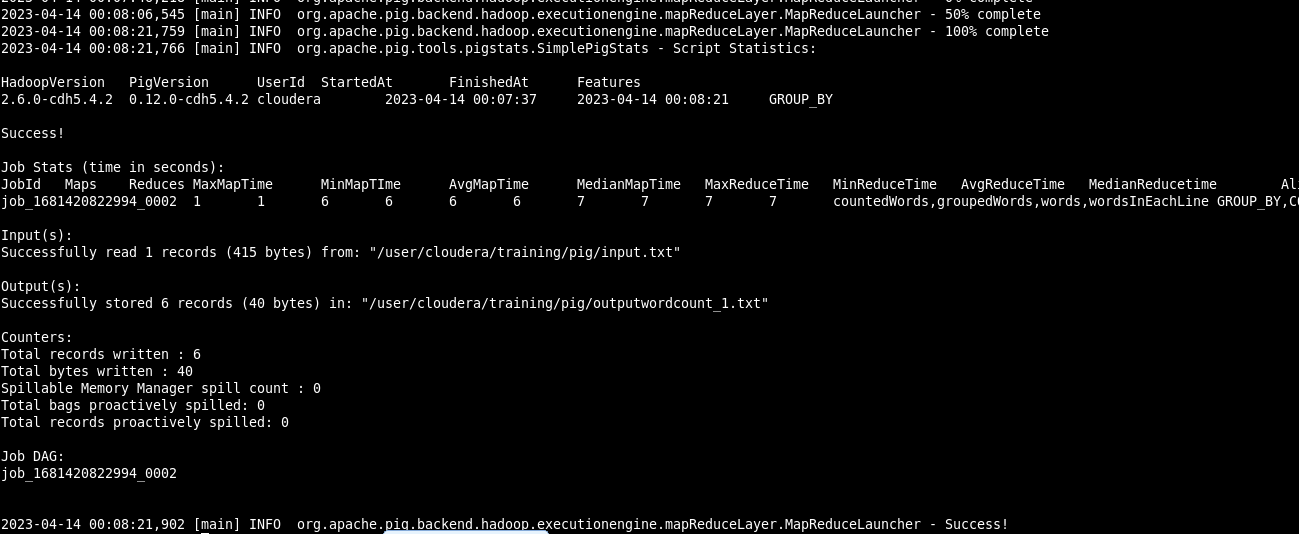
words = LOAD '/user/cloudera/training/pig/input.txt' AS (line:chararray);

wordsInEachLine = FOREACH words GENERATE flatten(TOKENIZE(line)) as word;

groupedWords = group wordsInEachLine by word;

countedWords = foreach groupedWords generate group, COUNT(wordsInEachLine);

store countedWords into '/user/cloudera/training/pig/outputwordcount\_1.txt' using PigStorage(',');



Ex 2 : Loading matrix in a pig variable.

Mydata.txt

1 2 3

4 5 6

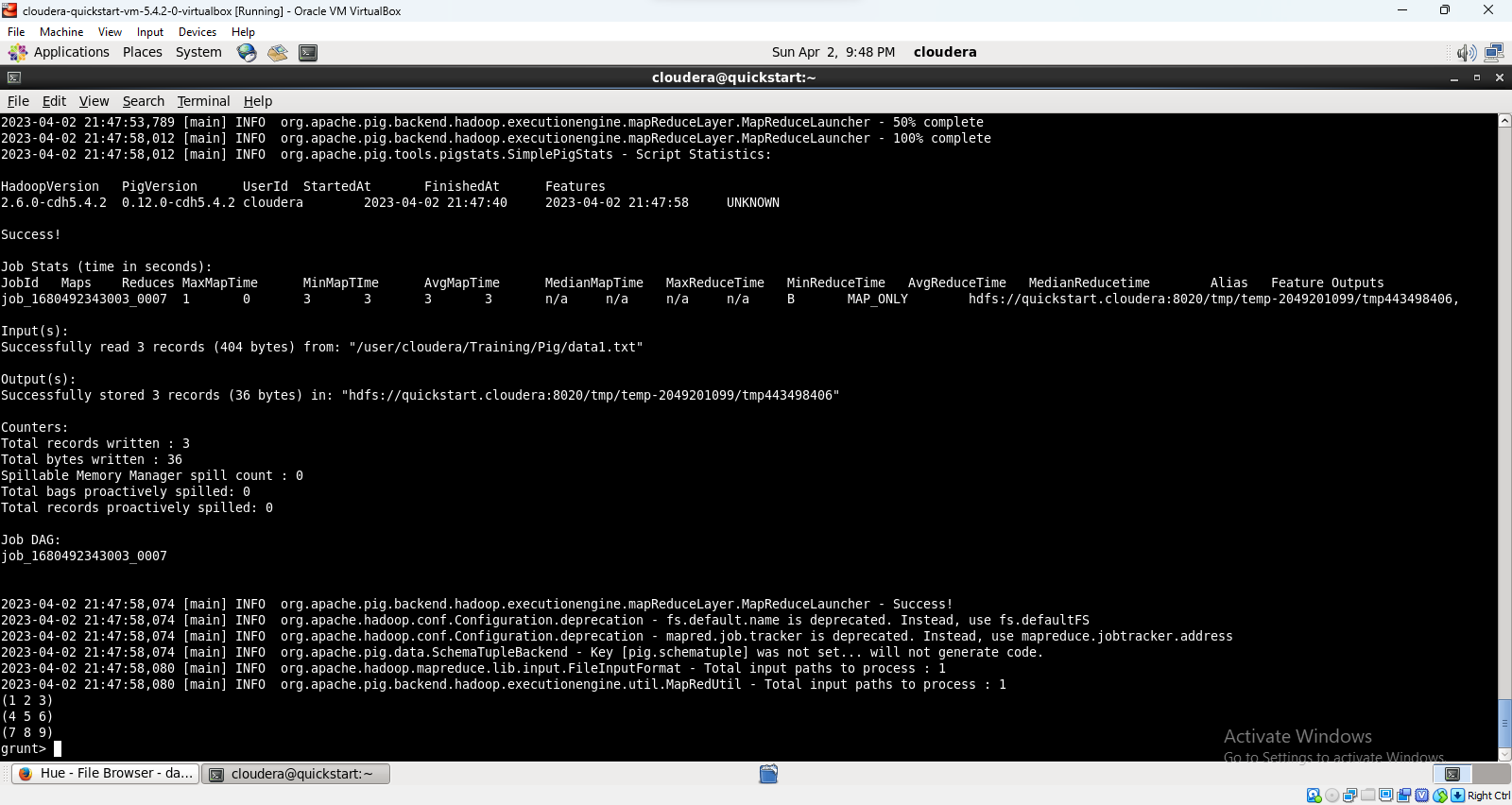
7 8 9

B=LOAD ‘/user/cloudera/Training/pig/mydata.txt’ AS (c1:int,c2:int,c3:int);



DUMP B;





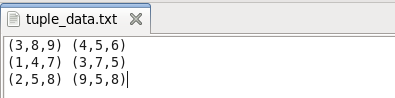
Example 4: Creating pig script and loading tuple data.

1. Create a relation named ‘tupledata’ and enter the following data.

(3,8,9) (4,5,6)

(1,4,7) (3,7,5)

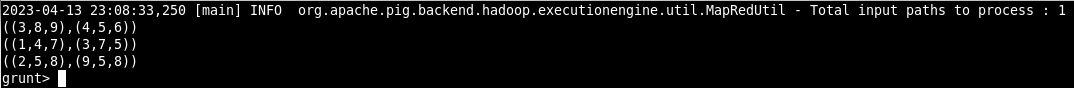
(2,5,8) (9,5,8)

****

* 1. Load tupledata into a pig variable named ‘inputtuple’.

inputtuple = LOAD 'tupledata' AS (t1:tuple(t1a:int, t1b:int,t1c:int),t2:tuple(t2a:int,t2b:int,t2c:int));

DUMP inputtuple;

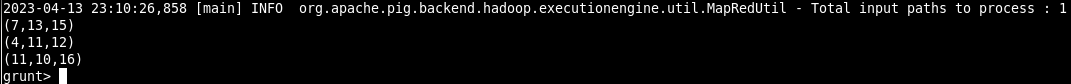


1. Add the columns with a similar index of each tuple and store the result in addout.out file.

//try following cmd

addition = FOREACH A GENERATE t1a+t2a,t1b+t2b,t1c+t2c;

store addition into ‘addout.out’;



A = LOAD 'data1' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

B = LOAD 'data2' AS (b1:int,b2:int);

DUMP B;

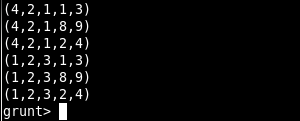
(2,4)

(8,9)

(1,3)

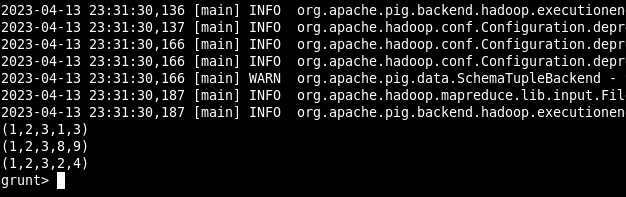
X = CROSS A, B;

DUMP X;



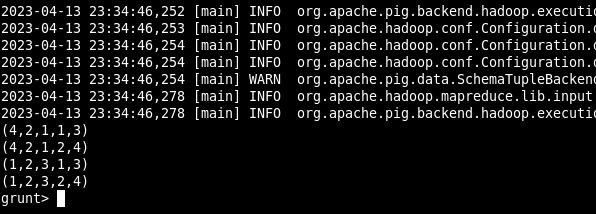
d= DISTINCT A;

f1 = FILTER X BY (a1 == 1);



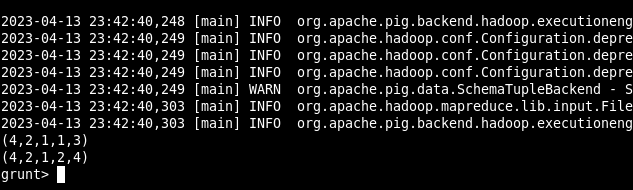
Q. Filter all rows for 2nd column val+4th column val=5th column val

F2 = FILTER X BY (a2+b1 == b2);

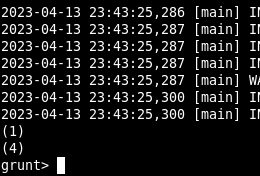


Q. Filter all rows for 2nd column val+4th column val=5th column val and also check for 1st column val as 4

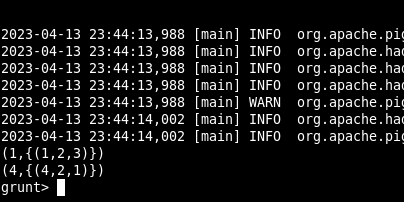
f1 = FILTER X BY (f2+f4 == f5) AND (f1==4);



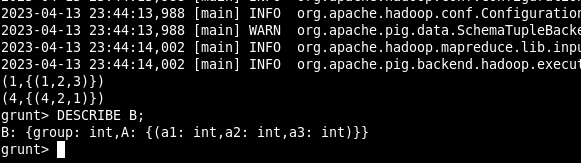
XX = FOREACH A GENERATE f1;



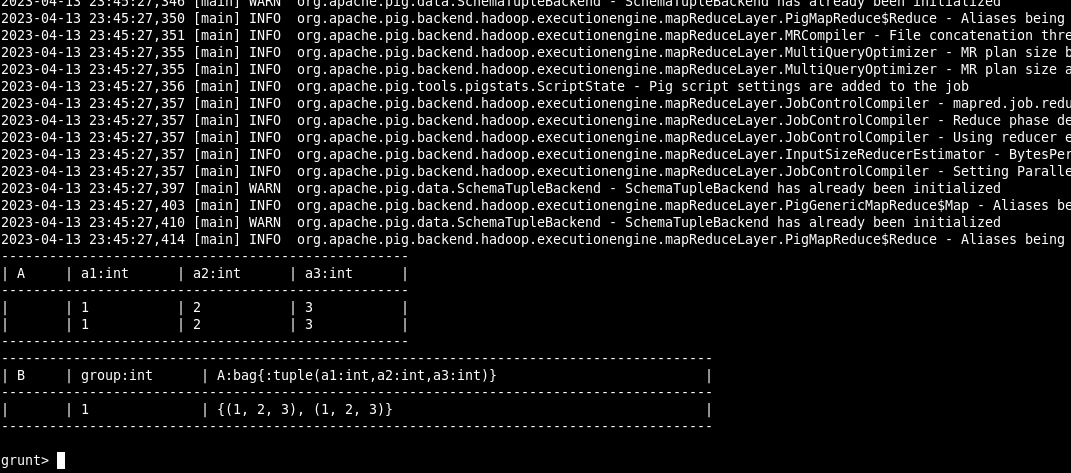
B = GROUP A BY f1



DESCRIBE B



ILLUSTRATE B



1. Create the file named ‘data’ for the following data.

1,2,3

4,2,1

8,3,4

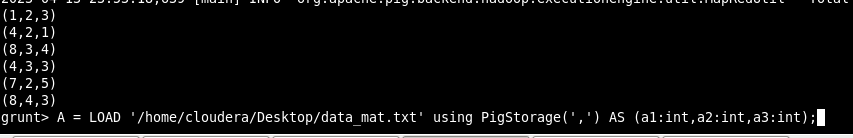
4,3,3

7,2,5

8,4,3

1. Load the contents of ‘data’ into relation A.

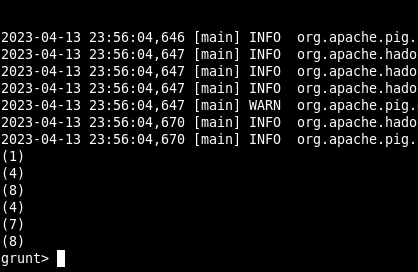
A = LOAD 'data' AS (a1:int,a2:int,a3:int);



1. Display the elements of the first column.

X = FOREACH A GENERATE f1;

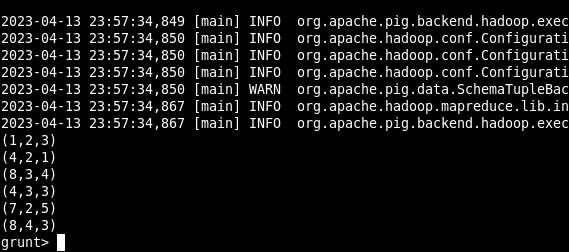
DUMP X



1. Display all tuples of relation A.

X = FOREACH A GENERATE \*;

DUMP X;



1. Display the first two columns of relation A.

X = FOREACH A GENERATE  a1,a2;

DUMP X;

