Big Data And Hadoop

Session 18 – Assignment 2

Problem Statement:

Perform the earlier operation using Java program.

The program should be able to check for the presence of 'customer' table.

In case the table exists, it must be dropped and recreated.

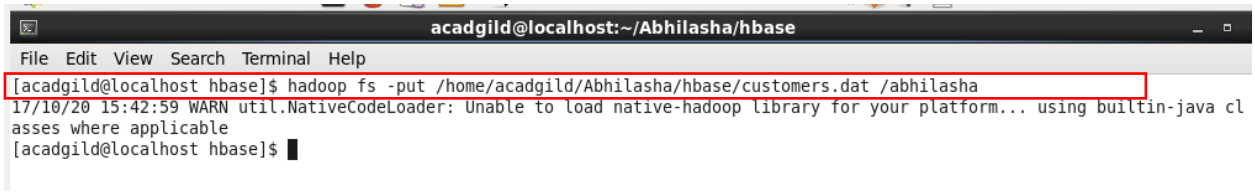
Solution:

A. Place input file in HDFS

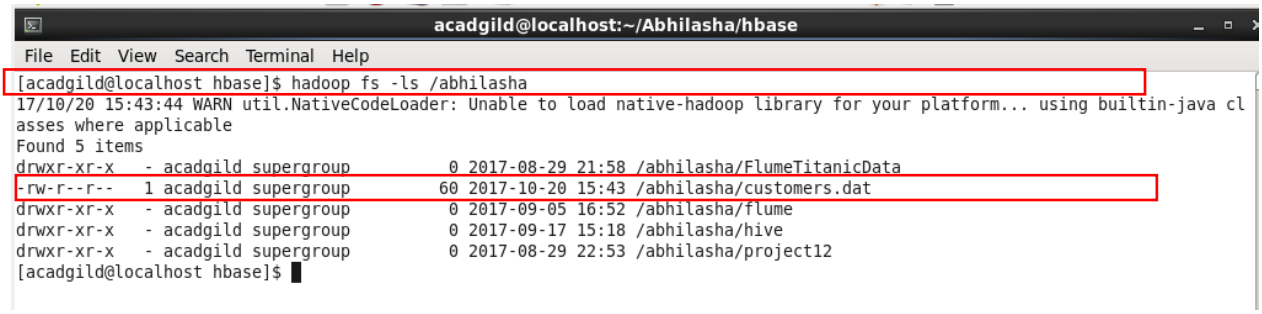
Step 1: The input file is present on local file system at **/home/acadgild/Abhilasha/hbase** as shown below.

Name of the file is customers.dat

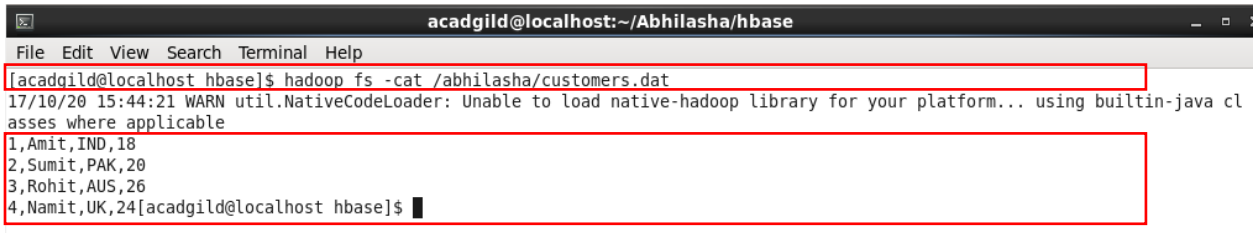
Step 2: We now place the file on HDFS using the **put** command at location **/abhilasha** as follows



Step 3: On listing the directory **/abhilasha**, we find the file with name **customers.dat** placed.



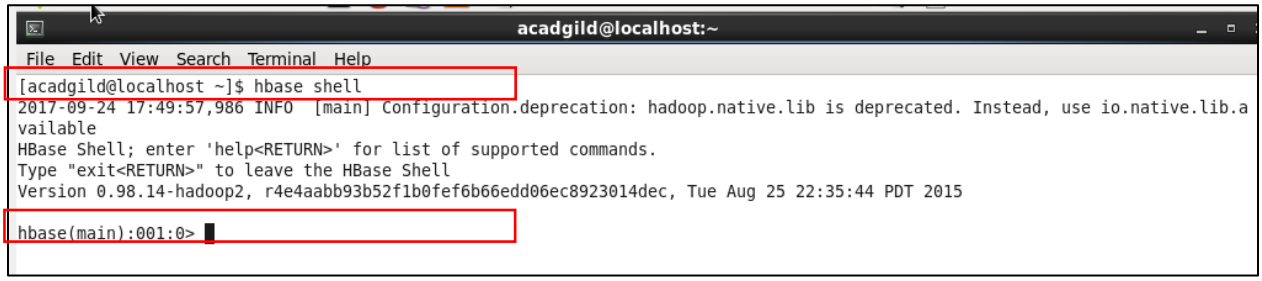
Step 4: We can view the content of this file using **cat** command as follows:



B. Start HBase

Step 1: We first start HBase using the command **start-hbase.sh** as follows:

Step 2: Next we start the hbase shell using the command **hbase shell** as follows:



C. Create table

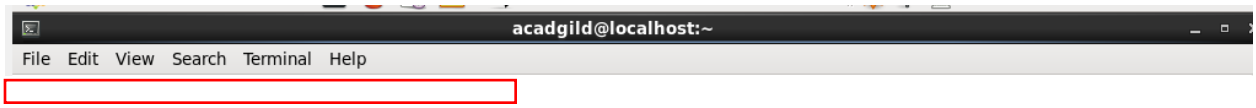
Now we create the table named **customer**. Here, **details** is the name of the column family of the table **customer**.

To create a table, we need to create an instance of HBaseAdmin and then ask it to create the table and with at least one column family name in it. To perform DDL operations, you must use HBaseAdmin class instance.

The code file required to create table is TableCreator.java

a. Use-case 1: Table did not exist already.

In the hbase shell, we use **list** command to check if the table with name **customer** already existed.



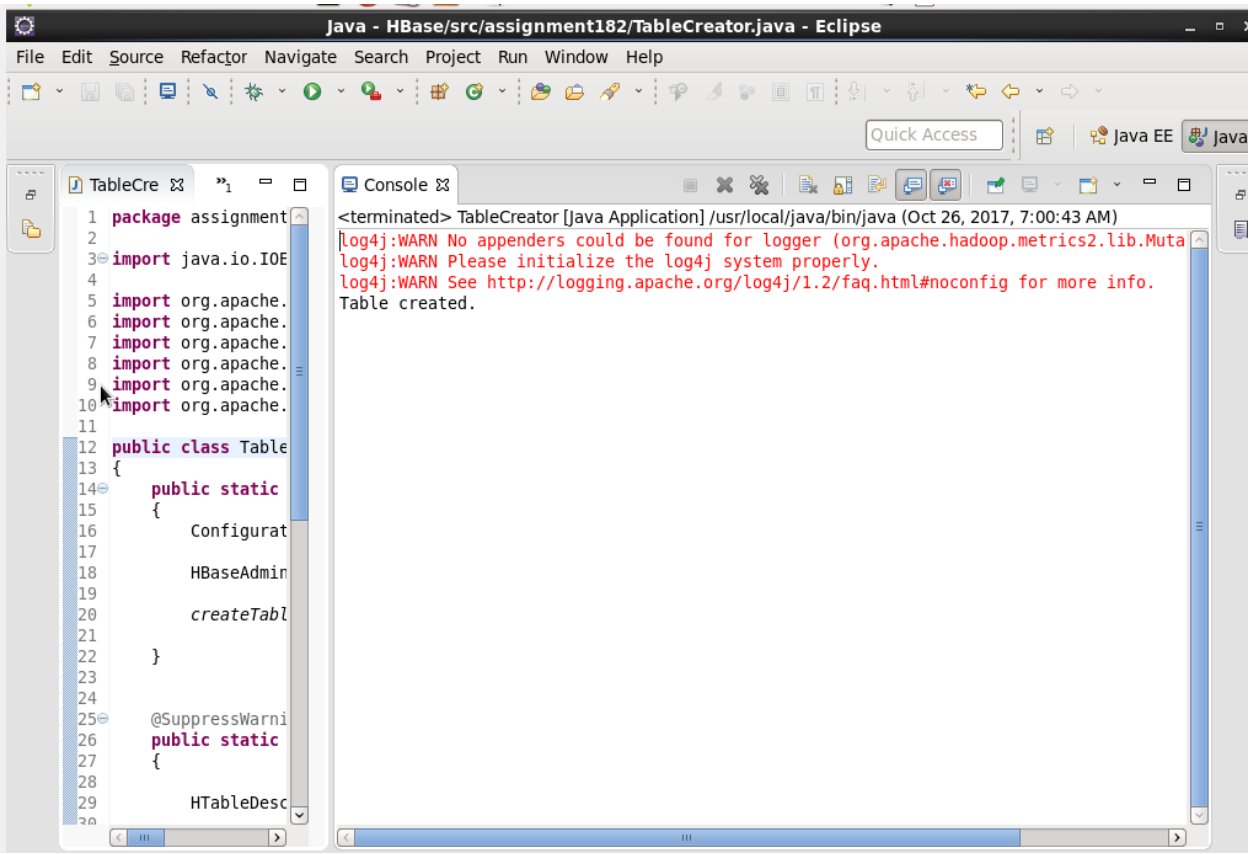
```
hbase(main):006:0> list
TABLE
clicks
htest
2 row(s) in 0.0190 seconds

=> ["clicks", "htest"]
hbase(main):007:0>
```

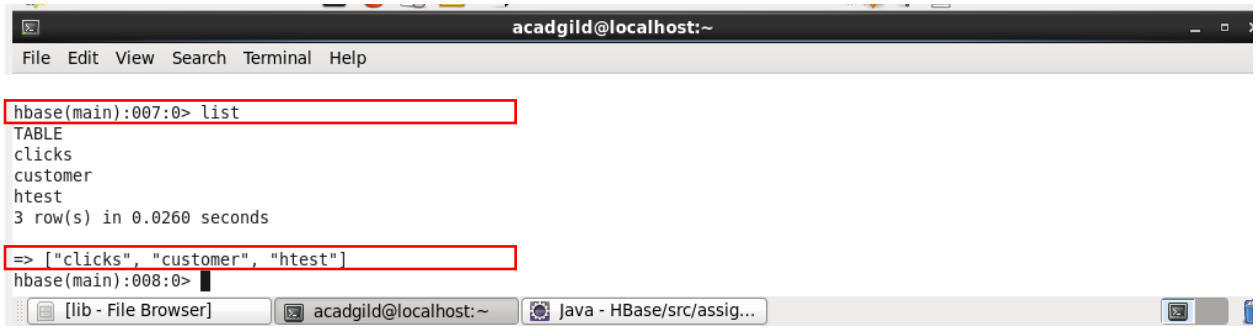
[lib - File Browser] acadgild@localhost:~ Java - HBase/src/assig...

The screen-shot shows that table named **customer** does not exist already.

On executing out code to create table in hbase, we get the following:

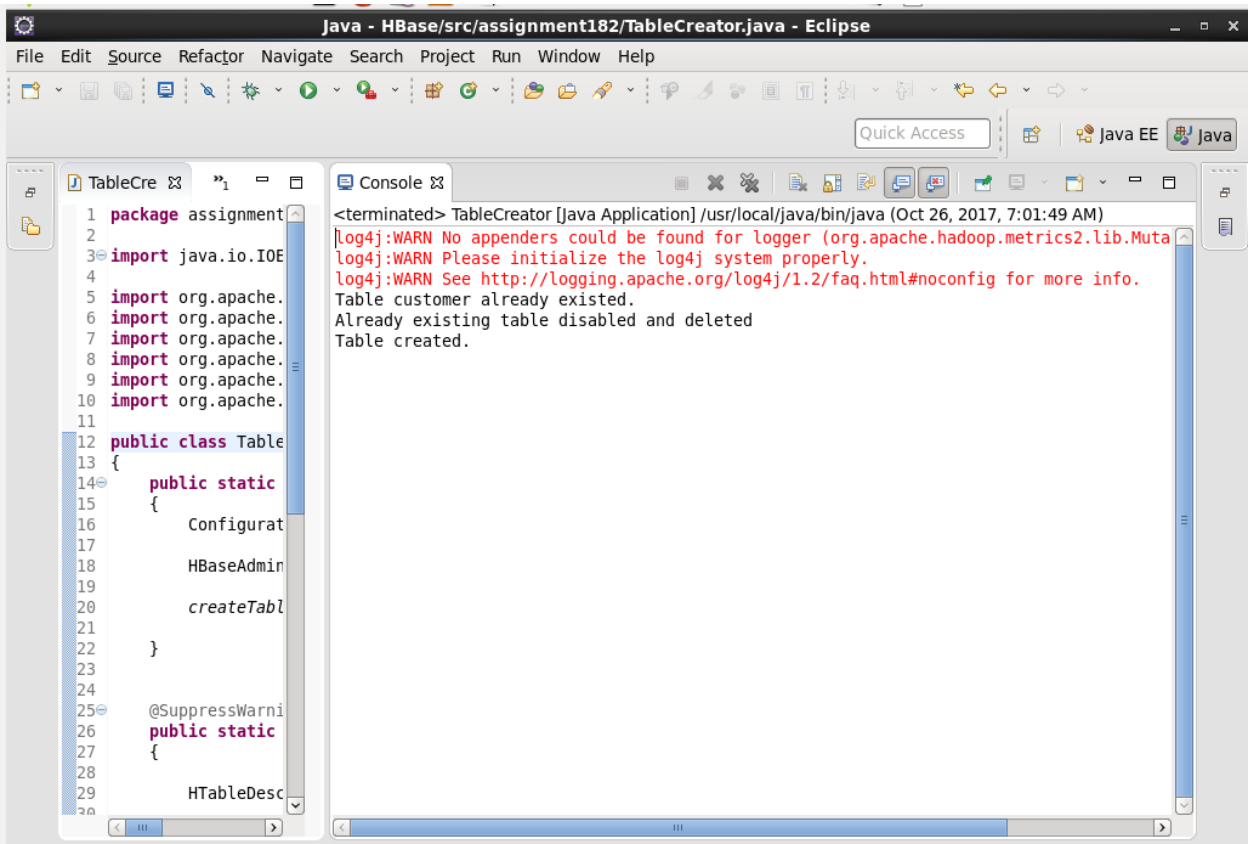


The table that got created now appears in hbase as follows:

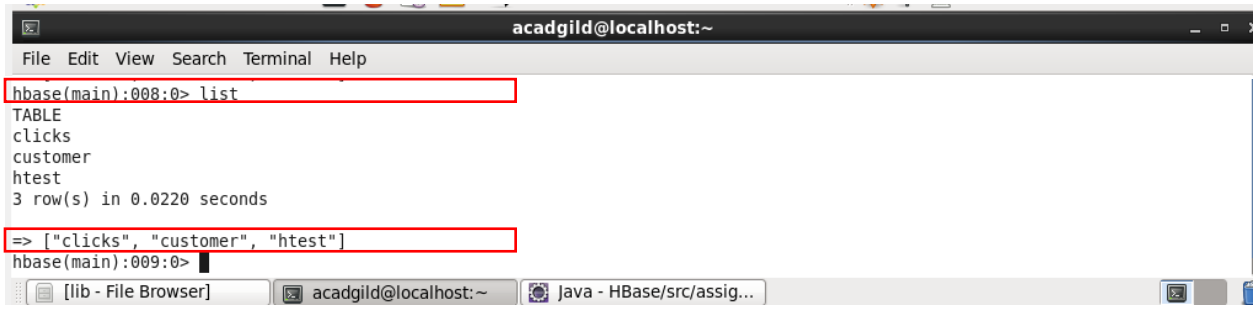


b. Use-case 2: The table already existed.

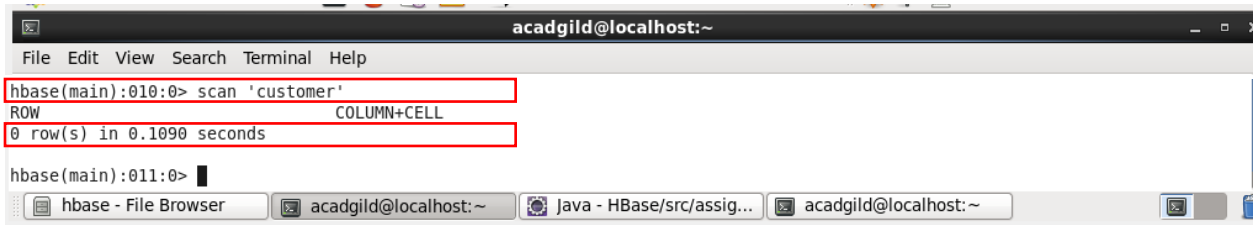
In this case, we disable already existing table and delete it, Then we create a new table. On executing the code, we get:



The newly created table appears in the list of tables in hbase as follows:



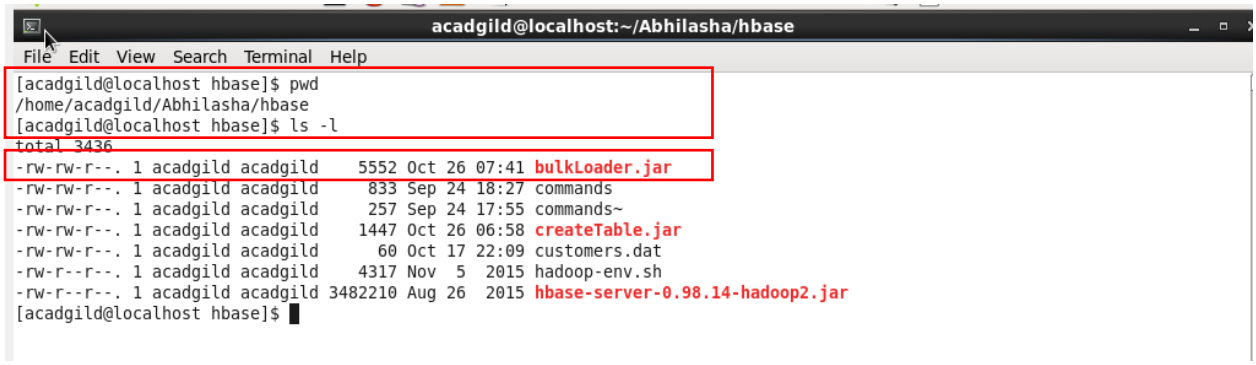
On using the **scan** command, we see that the table has no rows inserted.



D. Bulk insert data into hbase table from a file placed in HDFS.

Step 1: We need to load the HBase library files into the Hadoop classpath this enables the Hadoop client to connect to HBase and get the number of splits. To do so, we execute the following command.
export HADOOP_CLASSPATH=$HBASE_HOME/lib/*

Step 2: The code file used to perform bulk load is **BulkLoader.java.** The jar file created is named **bulkLoader.jar**. It is listed below:

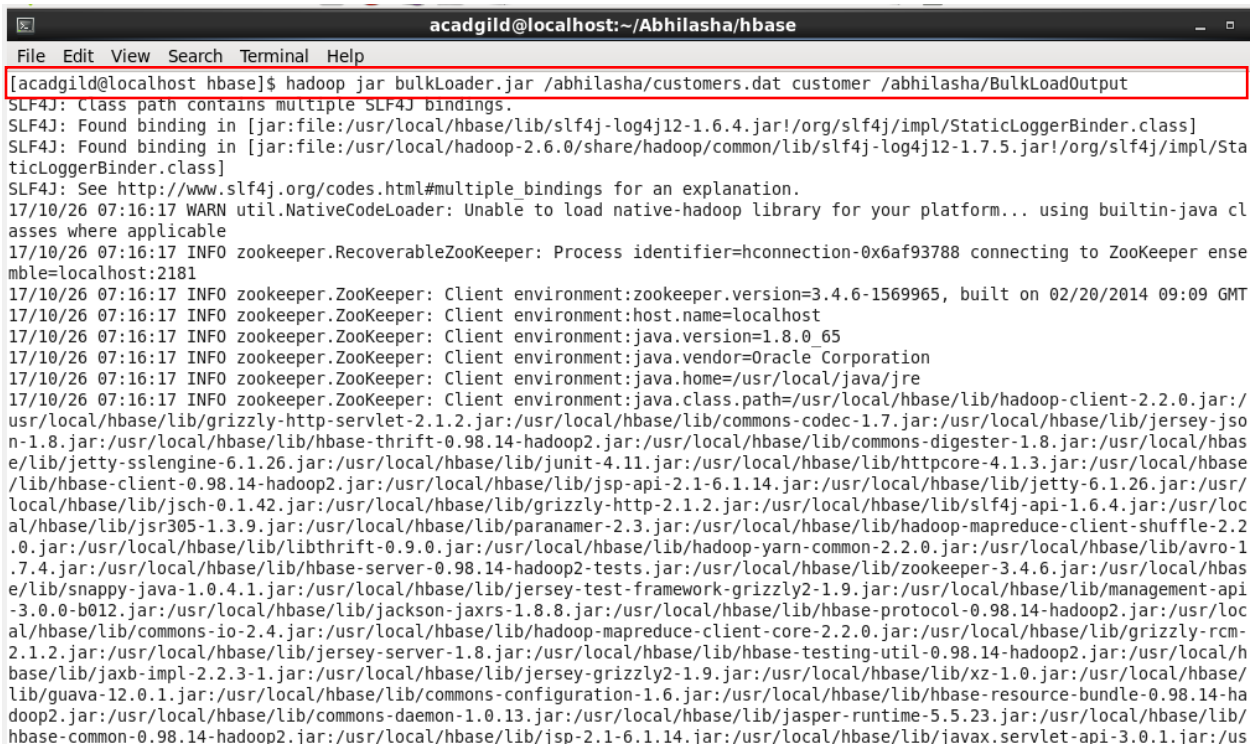


Step 3: Now, run the MapReduce job by following below command to generate the HFiles.
hadoop jar bulkLoader.jar /abhilasha/customers.dat customer /abhilasha/BulkLoadOutput

The jar is placed at /home/acadgild/Abhilasha/bulkLoader.jar
The map-reduce job takes three input parameters:

i. Input File HDFS path : /abhilasha/customers.dat
ii. Table Name : customer
iii. Output Folder HDFS path : /abhilasha/BulkLoadOutput

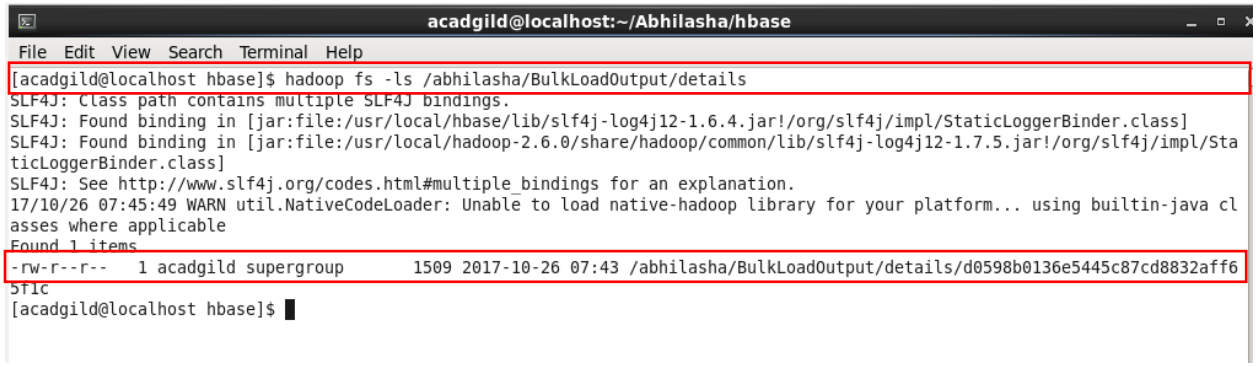
The execution of the above command is shown as follows:



After the job execution, we find the output folder created as follows:

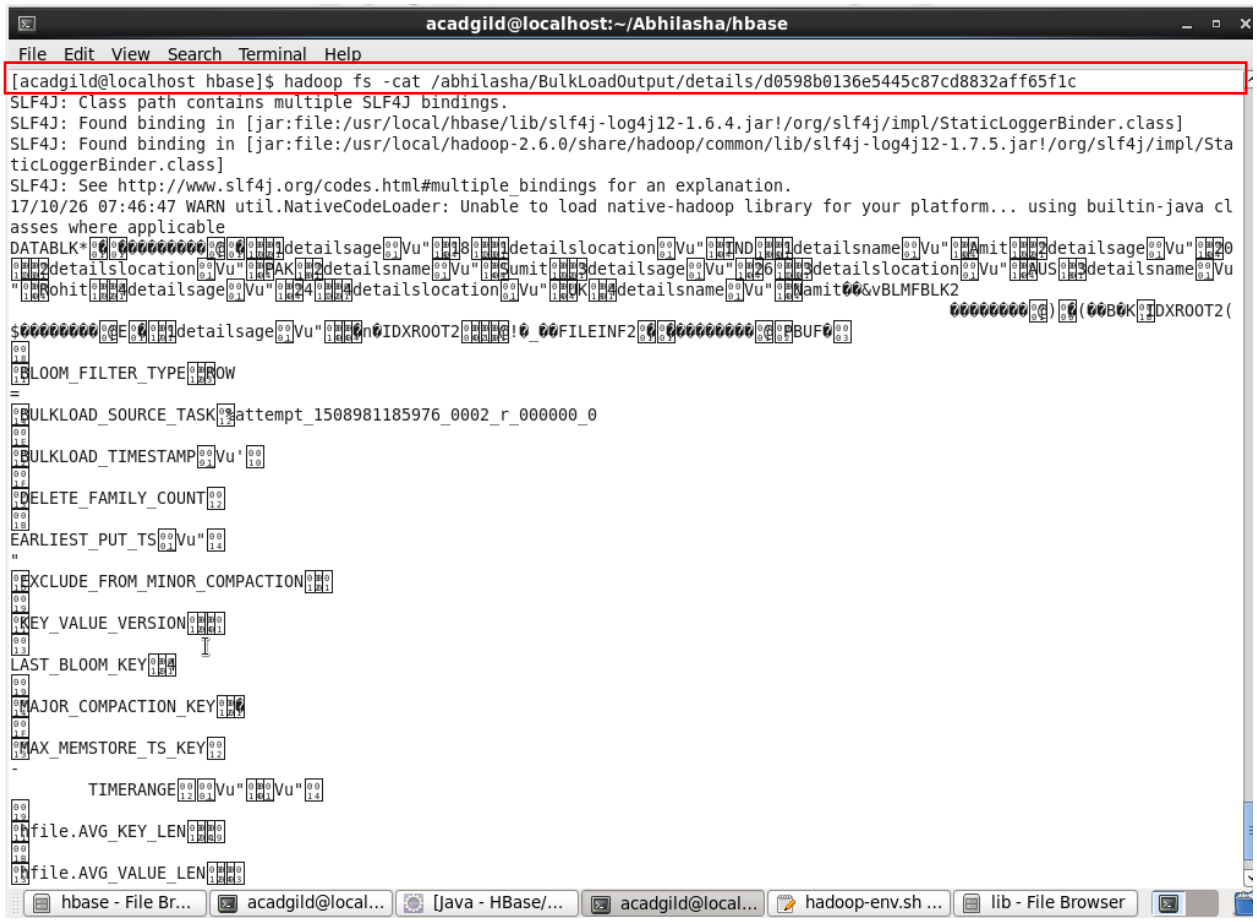
The content of this output folder is as follows:

details folder will contain the a folder with the name of column column and this folder will in turn contain the HFile as follows:



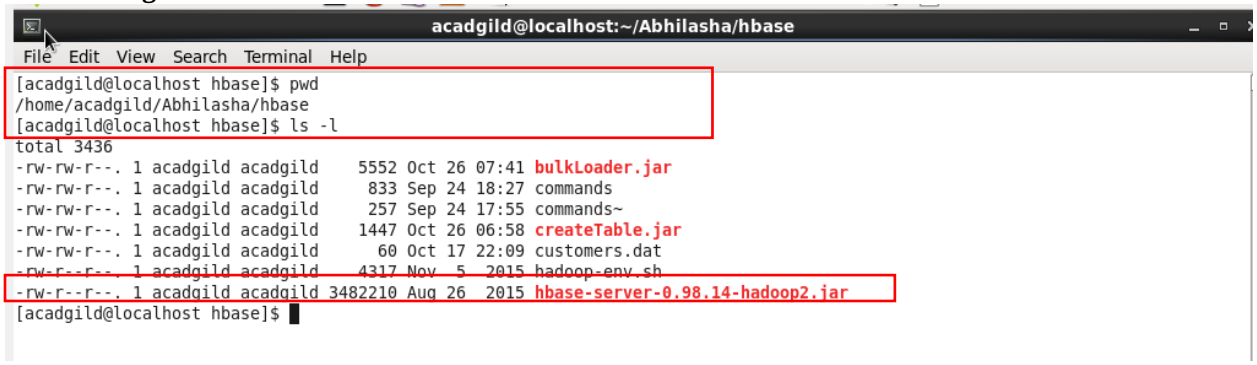
Here, name of the column family is **details.**

The content of the HFile can be seen as follows:



Step 4: After the data has been prepared using HFileOutputFormat, it is loaded into the cluster using completebulkload. This command line tool iterates through the prepared data files, and for each one determines the region the file belongs to. It then contacts the appropriate Region Server which adopts the HFile, moving it into its storage directory and making the data available to clients.
Now, we load the files into HBase by telling the RegionServers where to find them.
The command used is :
hadoop jar /home/acadgild/Abhilasha/hbase/hbase-server-0.98.14-hadoop2.jar completebulkload /abhilasha/BulkLoadOutput/ customer

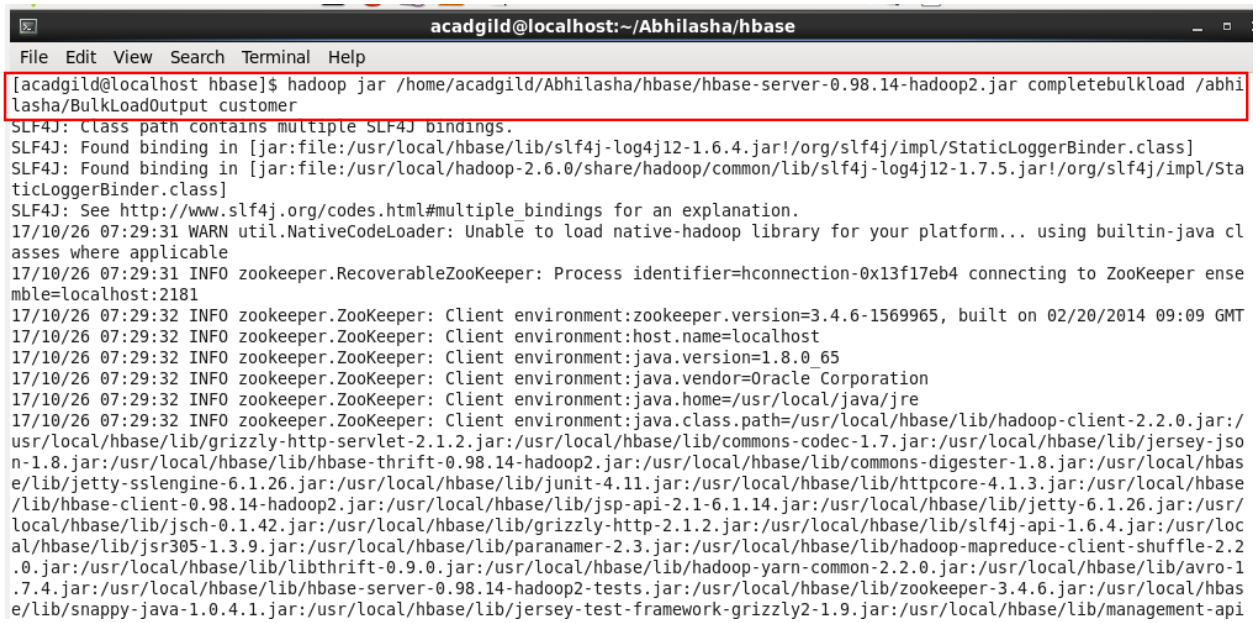
The required jar **hbase-server-0.98.14-hadoop2.jar** is placed in the local system at **/home/acadgild/Abhilasha/hbase** as shown below**.**



The input parameters for complete bulk load are:

i. Output directory HDFS path where HFile resides : /abhilasha/BulkLoadOutput/
ii. Table name : customer

The execution of the above command is shown as follows:



Output of the execution of the above command is the content of the HFile getting loaded into given hbase table as follows: