Big Data and Hadoop

Project 1.3: Twitter Sentiment Analysis

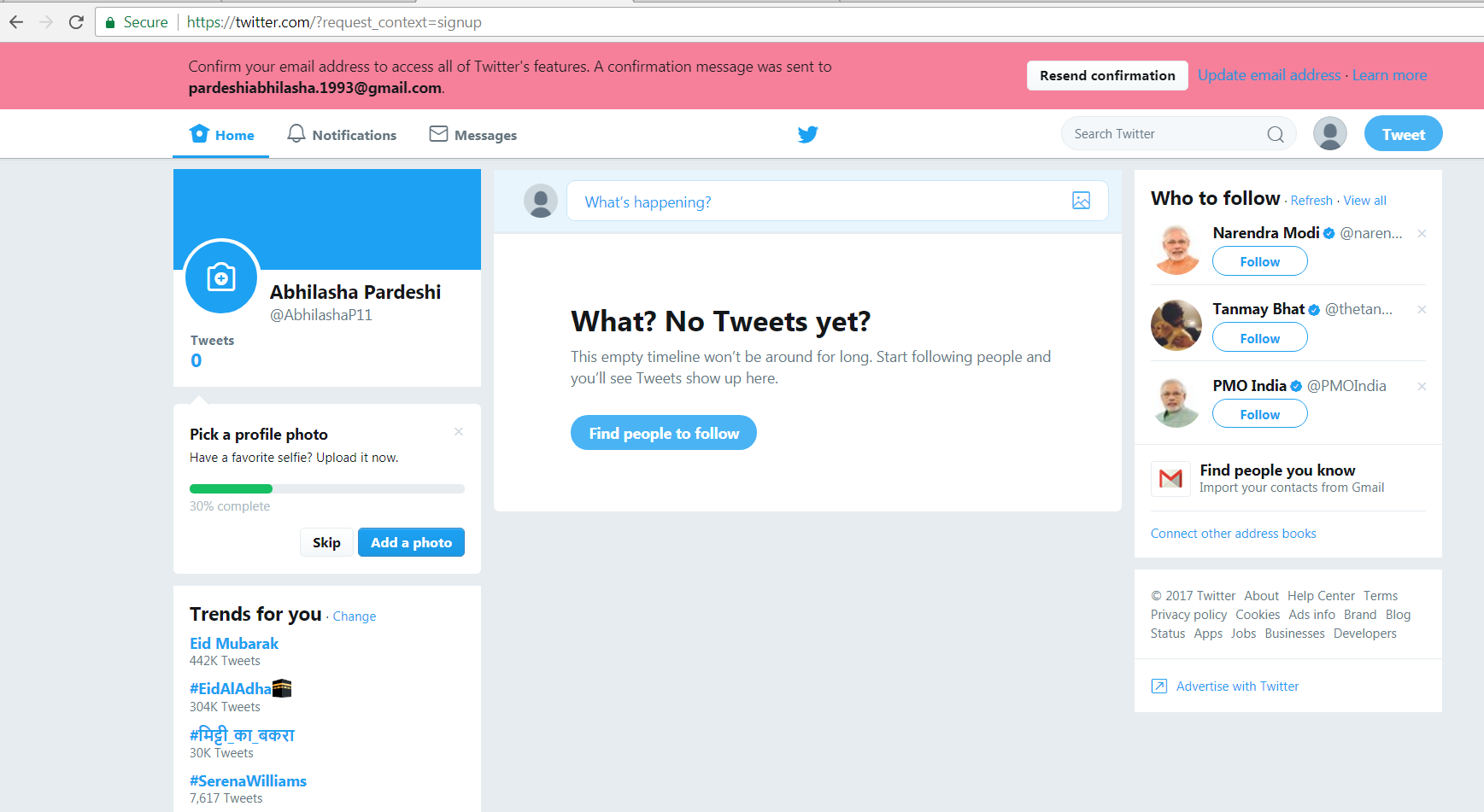
**Problem Statement:**

To stream data from Twitter and perform sentiment analysis on the data.

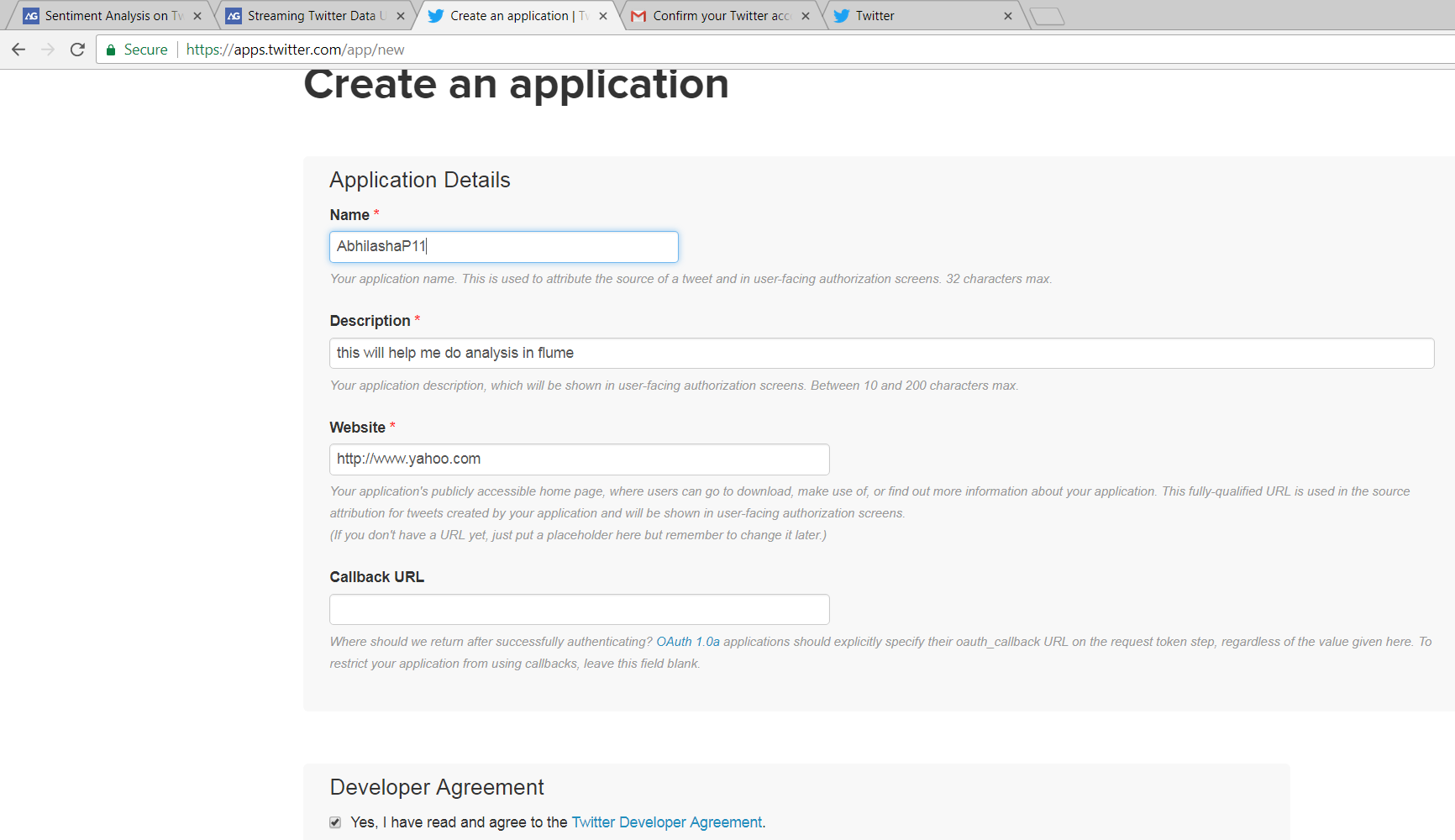
**Solution:**

**Part A**: Create a twitter app.

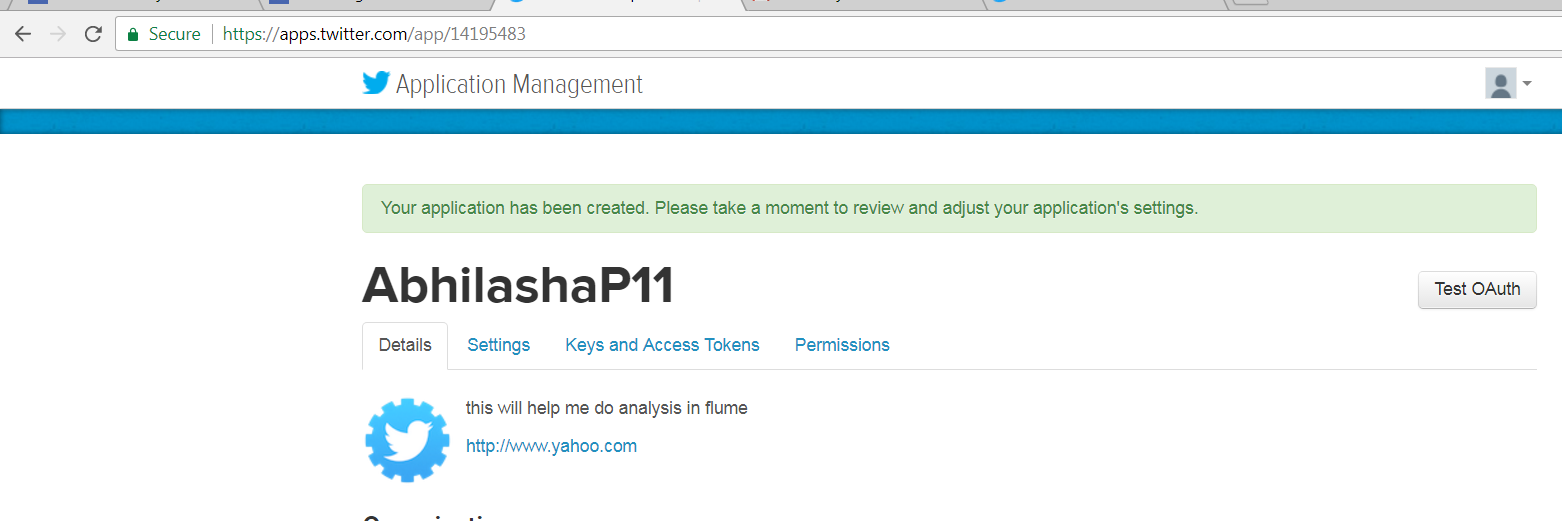
* In order to stream data to our database from twitter we should have Twitter account as the pre-requisite.
* Hence, created a twitter account and logged into it. The snapshot shows the logged in twitter account.



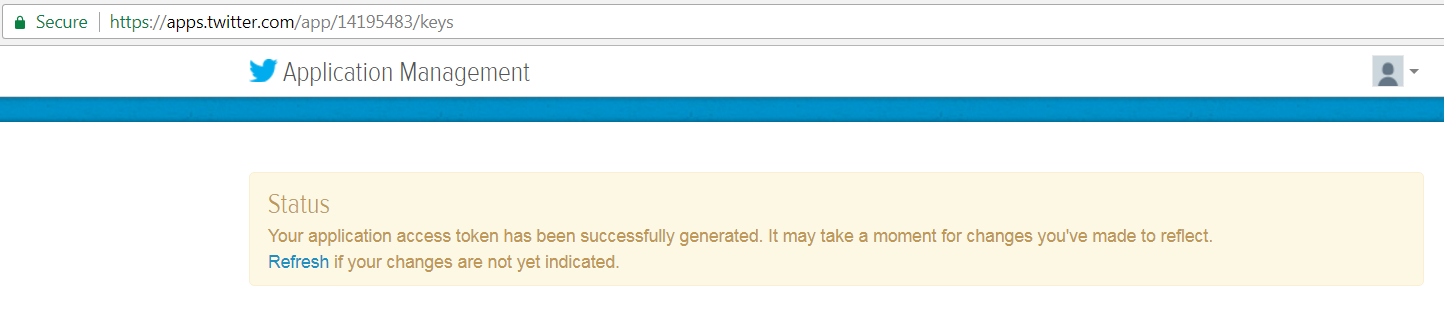
* After the login, we need to create a twitter application. Click on **Create New App**. The following window appears. We fill in the necessary details.



* The following snap shot shows the application is created.



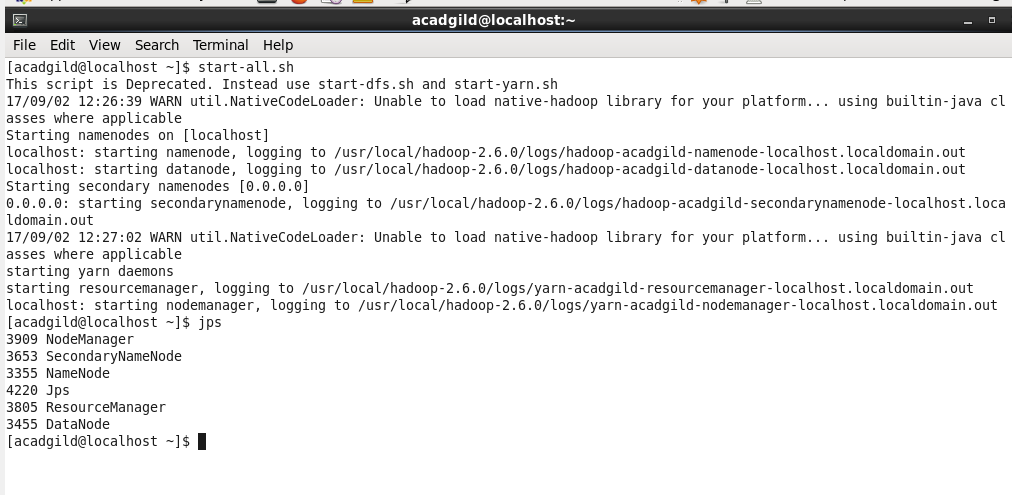
* Next we need to generate access key and token, hence, we click on Keys And Access Tokens tab. The following snapshot shows a message mentioning that access key and token have been generated.



* Then we also get the consumer key and consumer secret code.
* All the 4 parameters i.e., access key, access token, consumer key, consumer token are later used while configuring streaming of data from twitter.

**Part B:** Using flume to extract data from twitter into HDFS.

* We start HDFS using **start-all.sh** command, as shown below



We also use **jps** command to verify if Hadoop cluster has started.

* We are using flume to fetch data from twitter and store it in HDFS for further analysis.
* Flume Agent: Is an independent Java virtual machine daemon process which receives the data (events) from clients and transports to the subsequent destination (sink or agent).
* The configuration file used is flume.conf, which configures the flume agent.
* Flume agent is made of three parts

1. Source: Is the component of Flume agent which receives data from the data generators say, twitter, facebook, weblogs from different sites and transfers this data to one or more channels in the form of Flume event.
2. Channel: Once, the Flume source receives an Event, it stores this data into one or more channel and buffers them till they are consumed by sinks. It acts as a bridge between the source and sinks. These channels are implemented to handle any number of sources and sinks.
3. Sink: It stores the data into the centralized stores like HDFS.

In our use-case, source is the twitter agent, sink is HDFS and the channel used is memory channel.

* Properties of source defined in it are as follows:

1. type = com.cloudera.flume.source.TwitterSource

This is to specify that we are using twitter as source. Instead of using the out of the box apache twitter source, we have used cloudera twitter source as the default source was not functioning as expected.

ii. Authorization attributes. These include consumerKey, consumerSecret, accessToken, accessTokenSecret. These values are obtained from twitter account as mentioned earlier.

iii. keywords = hadoop, primeMinister, Food, Travel, Trek, election, sports, cricket, big data

Used to identify the tweets to read.

* Properties of the channel defined in it are as follows:

1. type = memory

This is to specify the we are using memory channel

ii. capacity = 1000  
 The maximum number of events stored in the channel

iii. transactionCapacity = 1000

The maximum number of events the channel will take from a source or give to a sink per transaction

* Properties of the sink defined in it are as follows:

1. type = hdfs

This is to specify that hdfs is the sink

ii. path = /abhilasha/flume/twitter

The path on HDFS where the data will be transferred

iii. batchSize = 1000

Number of events written to file before it is flushed to HDFS

iv. rollSize = 0

File size to trigger roll, in bytes

v. rollInterval = 600

Number of seconds to wait before rolling current file

vi. rollCount = 1000

Number of events written to file before it rolled

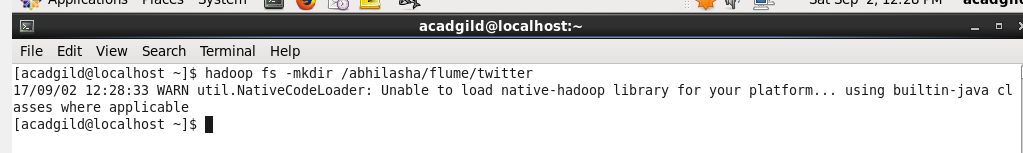
vii. writeFormat=Text

Format for sequence file records. One of “Text” or “Writable”

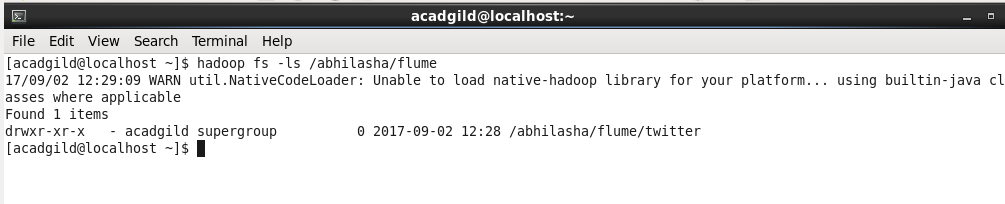
viii. fileType = DataStream

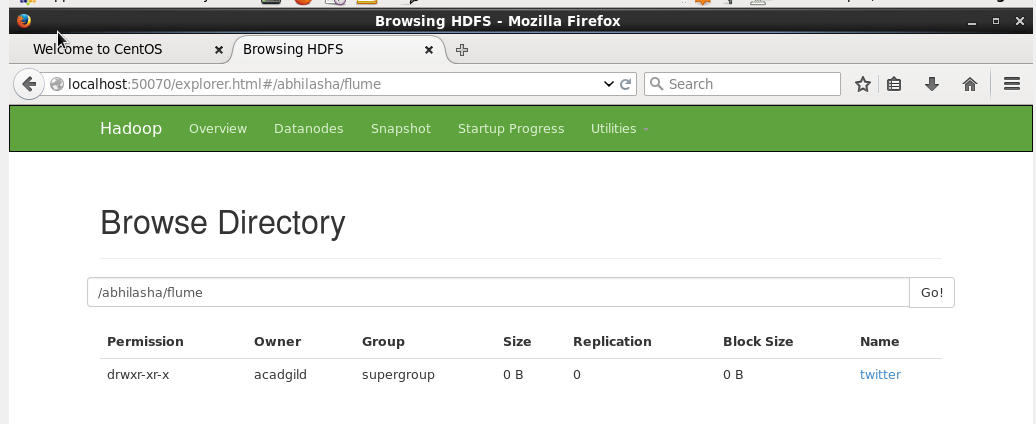
File format: currently SequenceFile, DataStream or CompressedStream

* Now that we have configured the flume.conf, we are all set to start using flume to fetch twitter streaming data into HDFS.
* We create a folder in HDFS where we want the data extracted from twitter, to be stored. We use **mkdir** command for the same. The destination folder is ‘/abhilasha/flume/twitter’. This is also specified in the flume.conf file.



* ls command is used to confirm creation of destination folder.

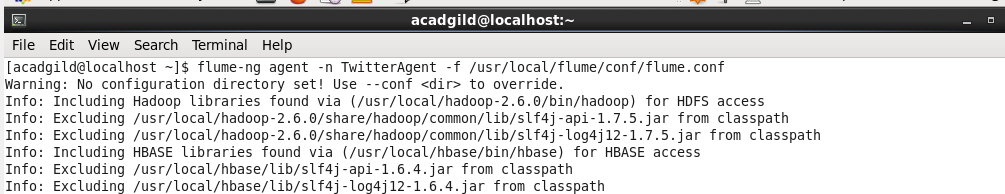


We can also verify it from the HDFS UI

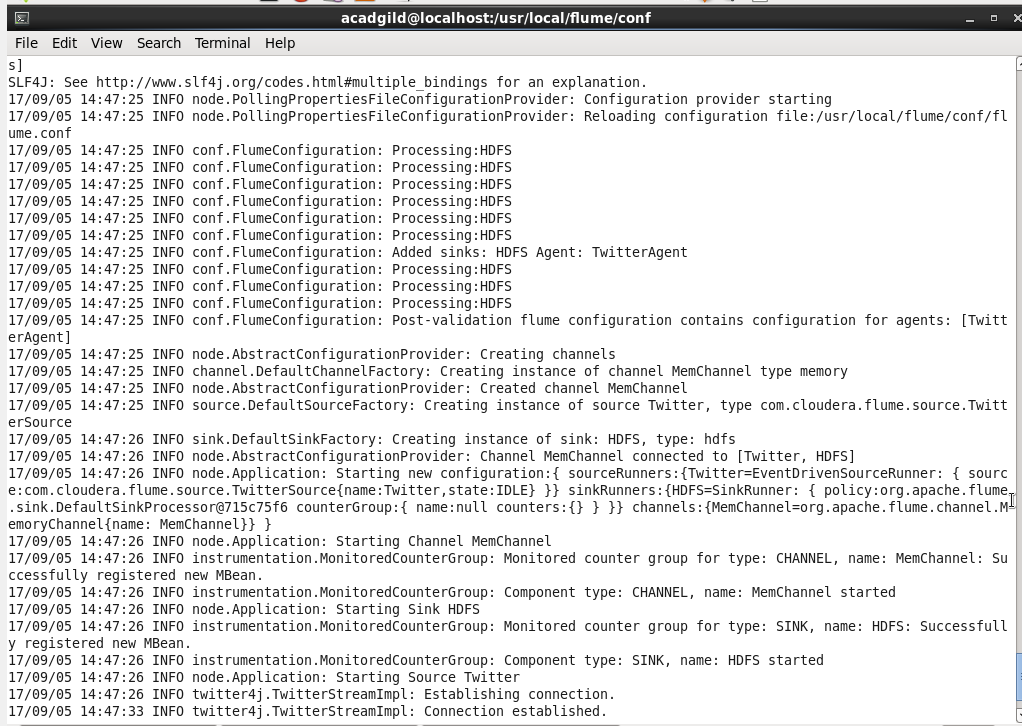
* Next is starting flume to start reading the data. The command used is

**flume-ng agent -n <Agent Name> -f <location of created/edited conf file>**

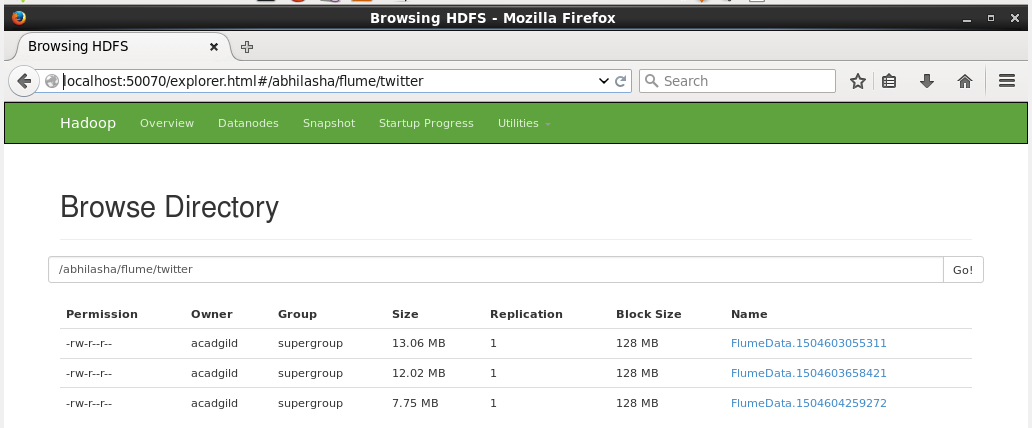
Conf file is ‘/usr/local/flume/conf/flume.conf’



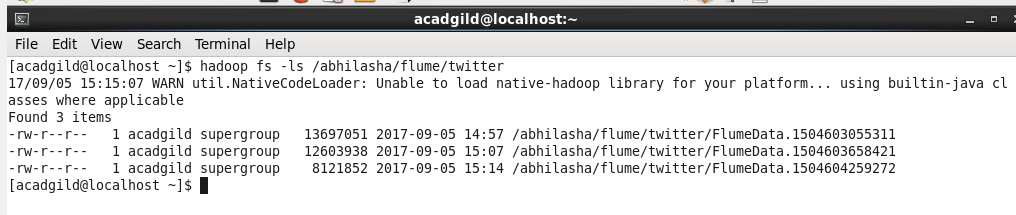
* The snapshot below shows that flume has started and has initiated reading data from twitter and dumping it in sink that is HDFS.



* The data extracted from twitter is stored in hdfs at the location specified in conf file. This is seen from the UI of HDFS as follows:



ls command can also be used to verify the same as follows:

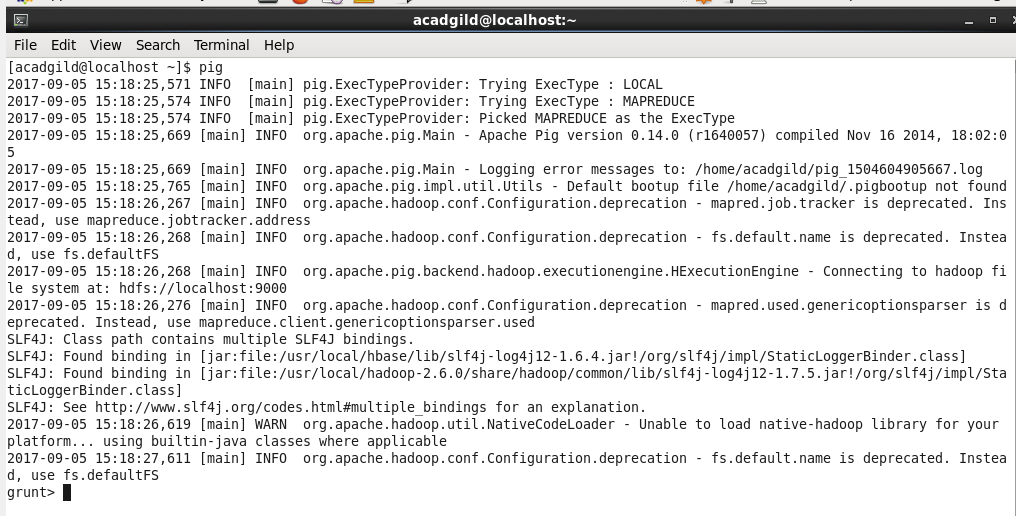


* Performing **cat** command on one of the datasets, we get the following output. The data is in Json format.



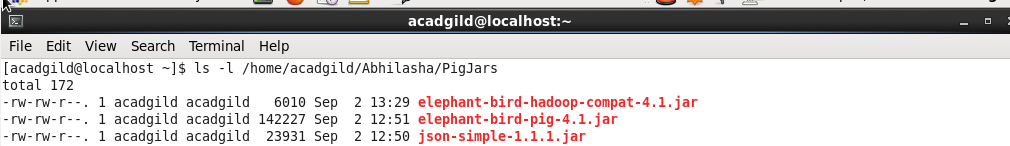
**Part C**: Perform analysis on data from twitter using pig.

* First we need to start pig in map-reduce mode using pig command as follows:

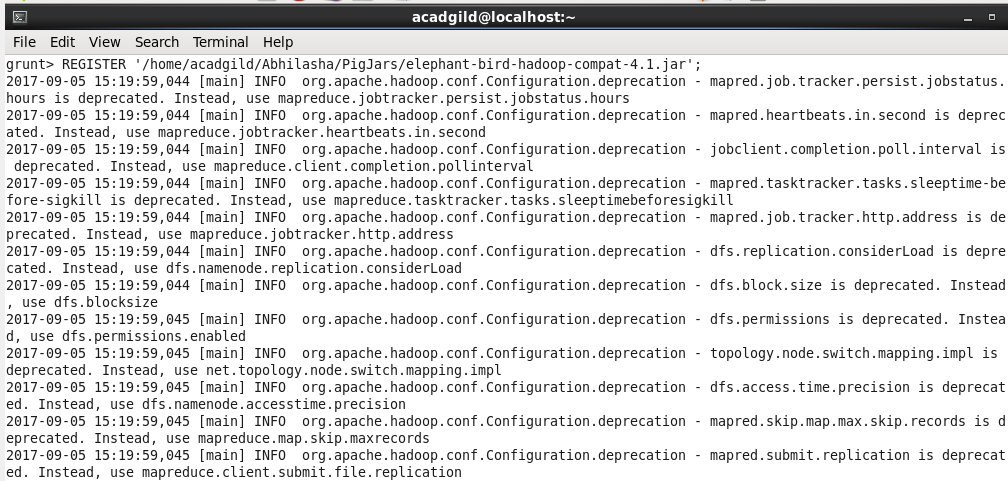


The grunt shell is now started.

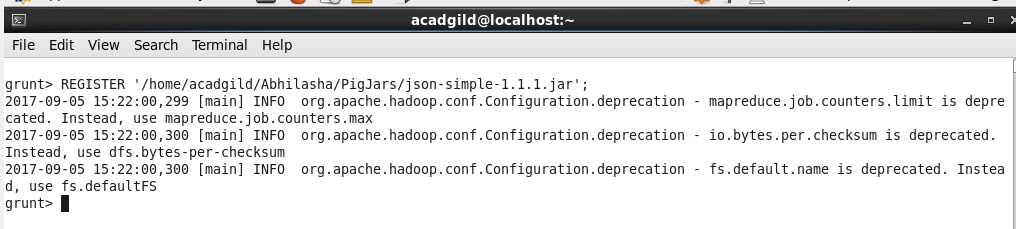
* As the data from twitter is in Json format and not regular text, we need some extra jars to parse json data. These are stored at **/home/acadgild/Abhilasha/PigJars.**



* We register these jars in pig by executing the **REGISTER** command as follows:

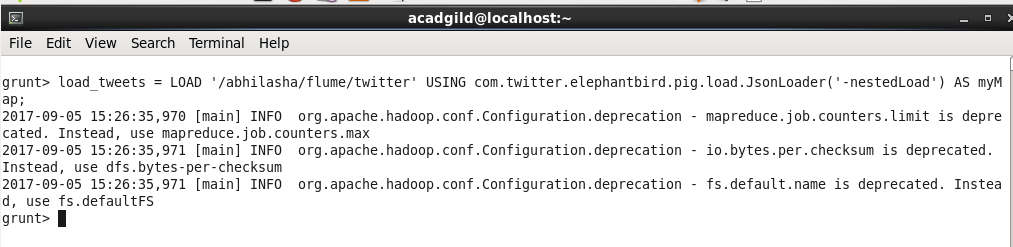




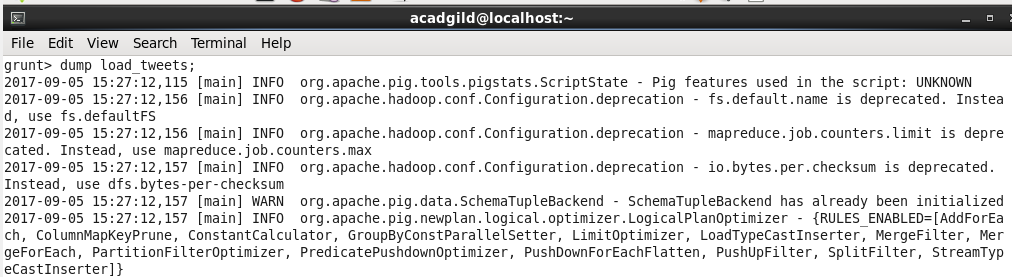


**Analysis 1:**

Step 1: Load the data in pig using **LOAD** command as follows. We specify the path in HDFS from where data is to be loaded. The path mentioned is **/abhilasha/flume/twitter.**  This will load all the data files present in this directory into pig.



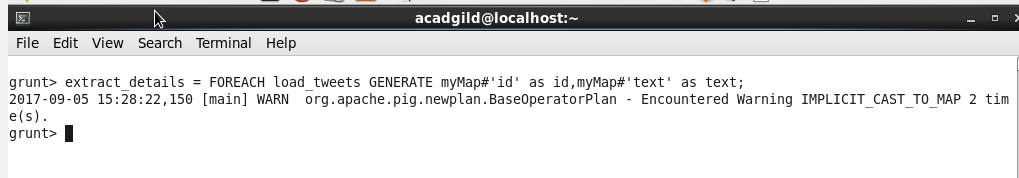
We use the **dump** command to see the data on console, that is to dump the relation, whose name is specified, on console.



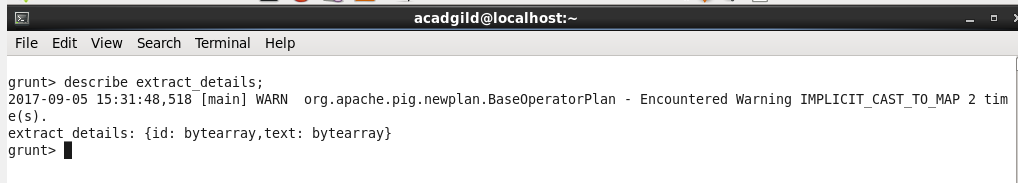
Snippet of the dump is



Step2: Now, we extract the **id**and the **tweet text** from the above tweets. The Pig statement used for the same is as shown in the snapshot. We use **FOREACH** to iterate over all the tweets and extract id and text from it. Tweet id and text is extracted in the relation **extract\_details.**

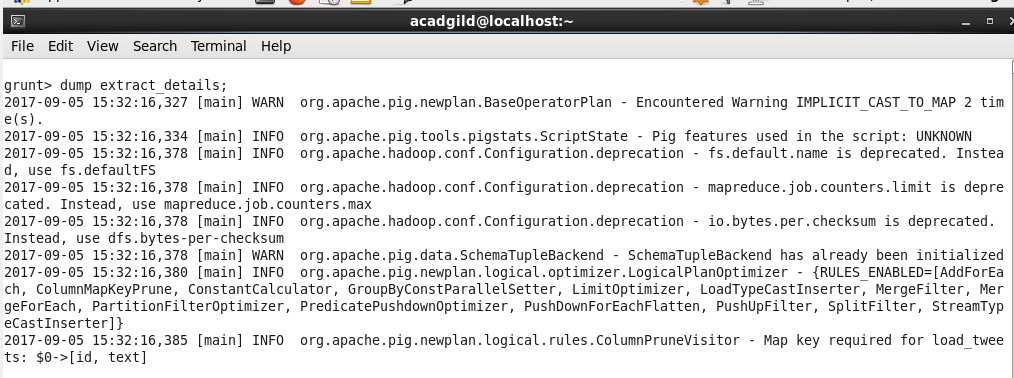


Using **describe** command, we can get the schema of the relation as follows:

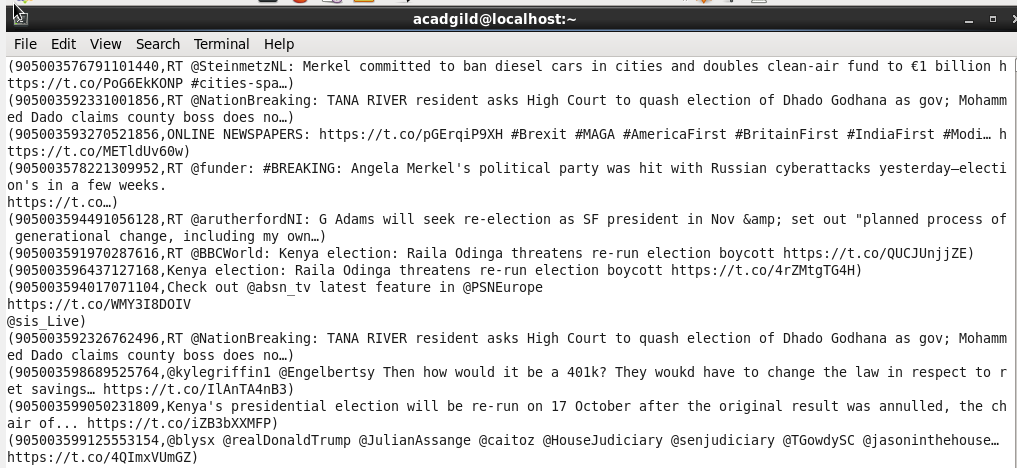


The fields of **extract\_details** relation are id and text, both byte arrays, as mentioned in output of describe command.

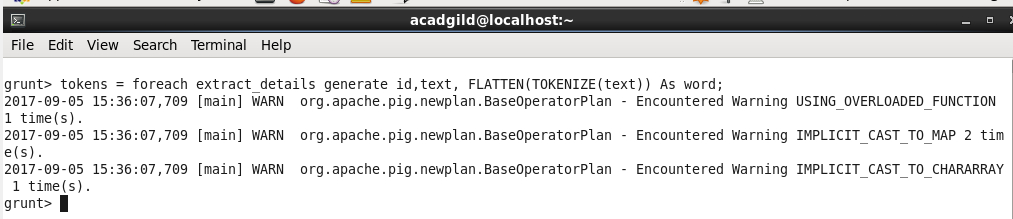
Performing dump **extract\_details,** as shown below



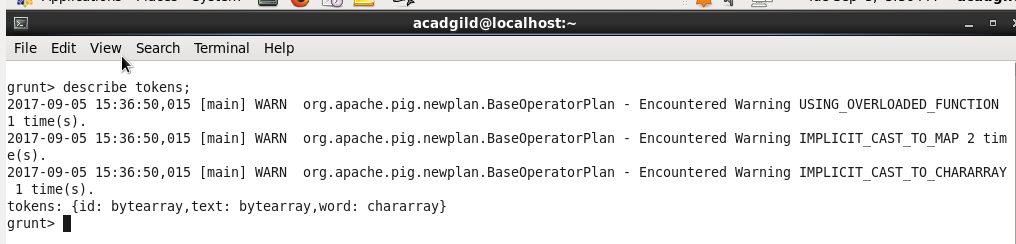
Snippet of output of dump of **extract\_details** is



Step 3: Next is to extract words from text using **TOKENIZE.** We also use **FLATTEN** on the output of tokenize. **FLATTEN** will remove all the words from the nesting of bags/tuples and provide a flat structure. We have also used the alias for it as **word.**

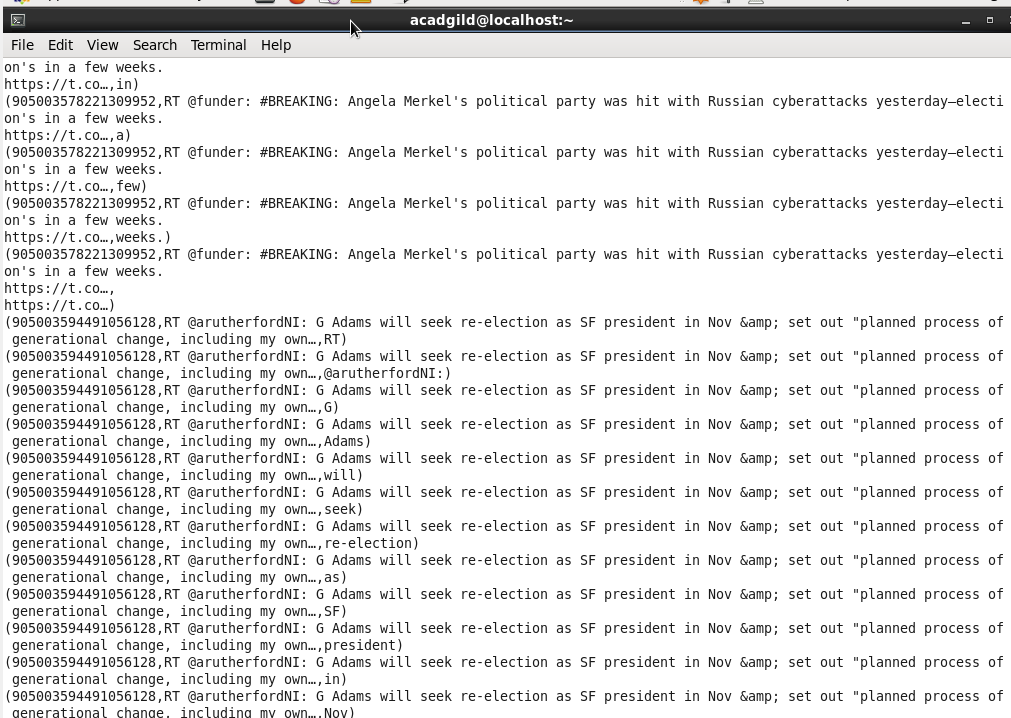


Using **describe** command, we can get the schema of the relation **tokens** as follows:



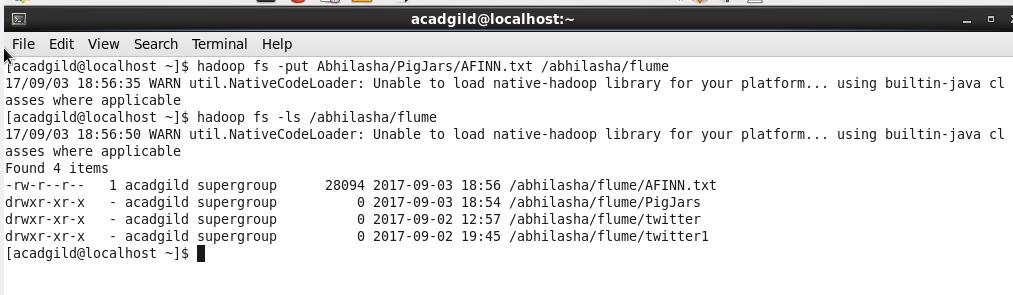
**tokens** relation has records in the format : id, text, word, as mentioned in the schema.

Performing dump of tokens, following is the snippet, every record contains id, text and the word after tokenization. Here, the text is divided into words.

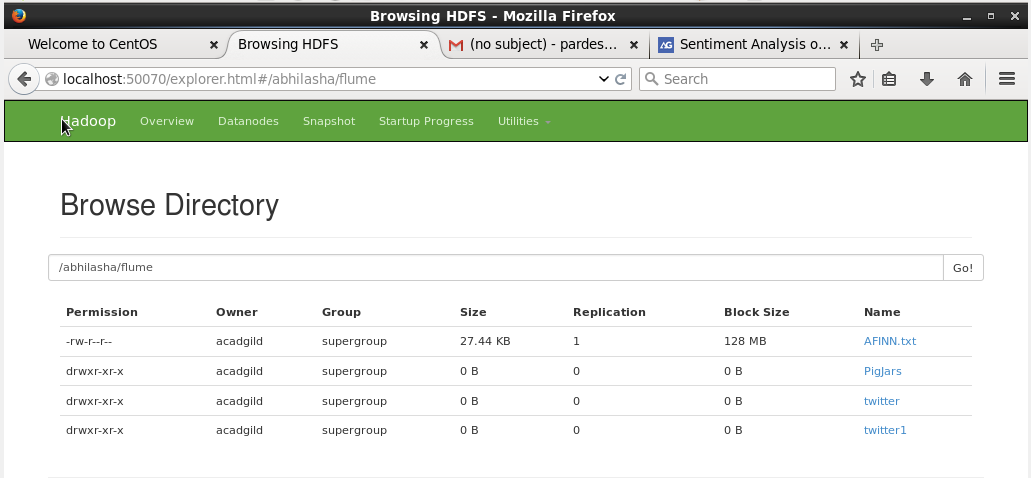


Step 4: Now, we have to analyze the sentiment of the tweets by using the words in the text. We will rate the word as per its meaning from +5 to -5 using the dictionary AFINN. The AFINN is a dictionary which consists of 2500 words which are rated from +5 to -5 depending on their meaning.

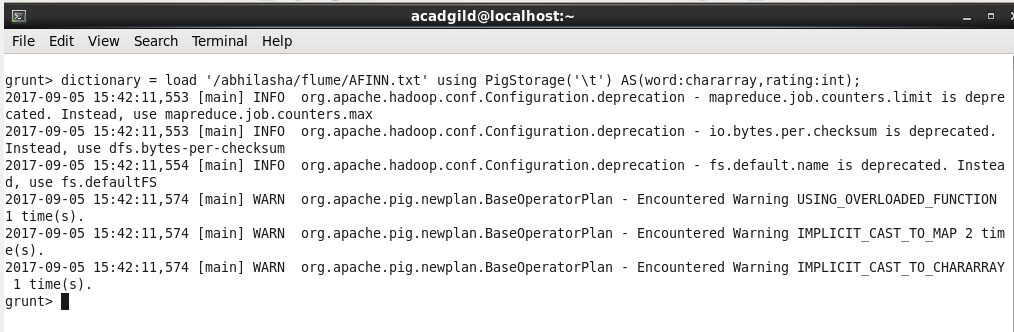
So, we will put the downloaded AFINN dictionary file into HDFS using **put** command as follows:



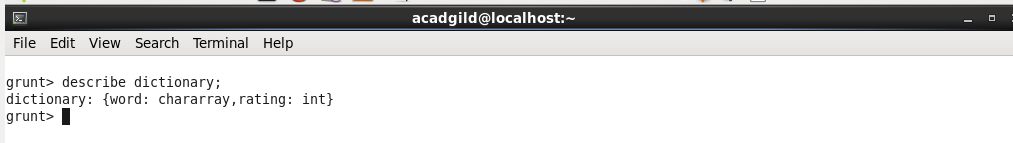
Here, AFINN file was placed locally at **/home/acadgild/Abhilasha/PigJars.** We put it in HDFS at /abhilasha/flume. After executing **put** command, we verified the presence on AFINN file on HDFS using **ls** command, as seen in above screen shot. We can also view it from UI of HDFS as follows:



Step 5: Load AFINN.txt into pig. We have also mentioned its schema, the tuple is made of word and rating.



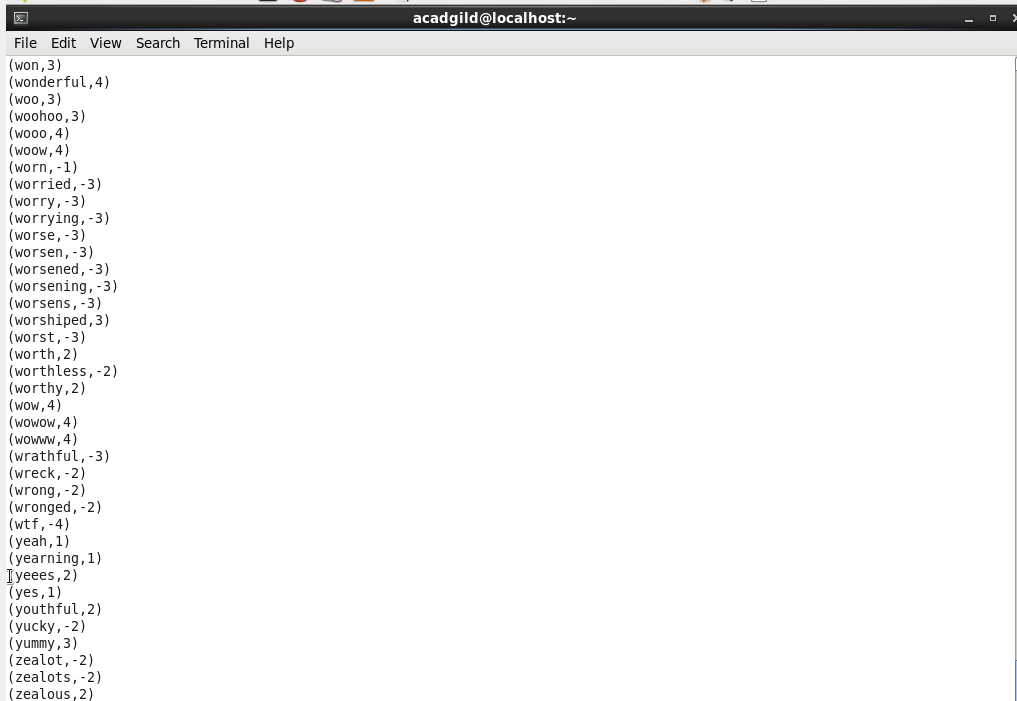
**Describe** command also verifies its schema.



Snippet of the dump of the relation dictionary is as follows:



Dump output:



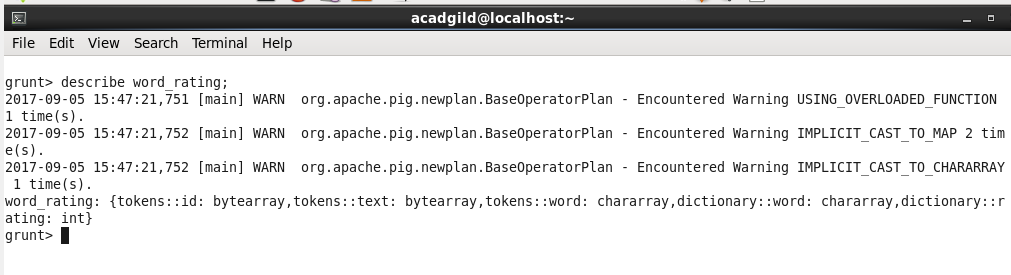
Step 6: Next, we perform a join of the relations **tokens** and **dictionary**, as we need the rating for each word in **tokens** relation. The using **'replicated'** tells Pig to use the fragment-replicate algorithm to execute this join. The second input listed in the join (in this case, **dictionary**) is always the input that is loaded into memory.

Also, we have used left outer join, as there would be words in **tokens** relation that do not have entry in **dictionary** relation.

The syntax for the same is as follows:

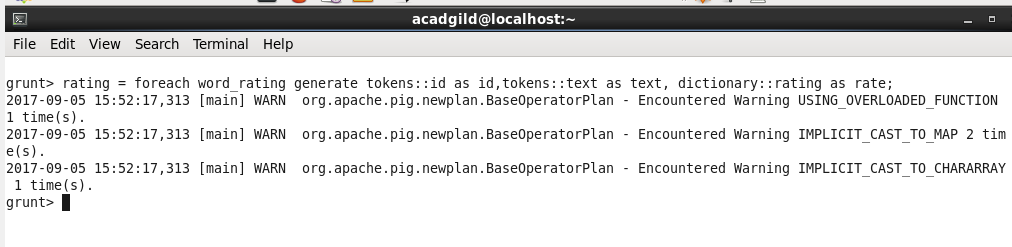


**Describe** command specifies its schema as follows:

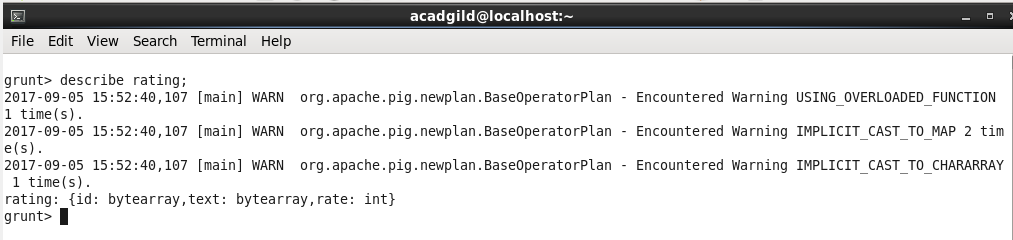


Step 7:  The **word\_rating** has joined the **tokens (**consists of id, tweet text, word**)**statement and the **dictionary**(consists of word, rating).

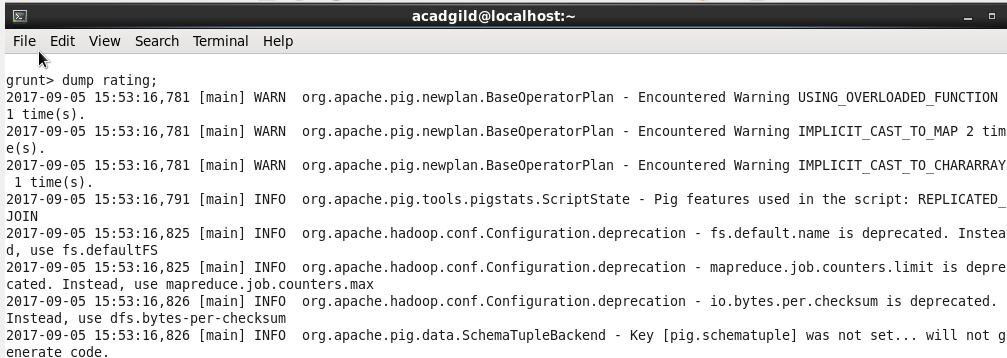
Now we will extract the **id, tweet text**and **word rating (**from the dictionary**)**by using the below relation



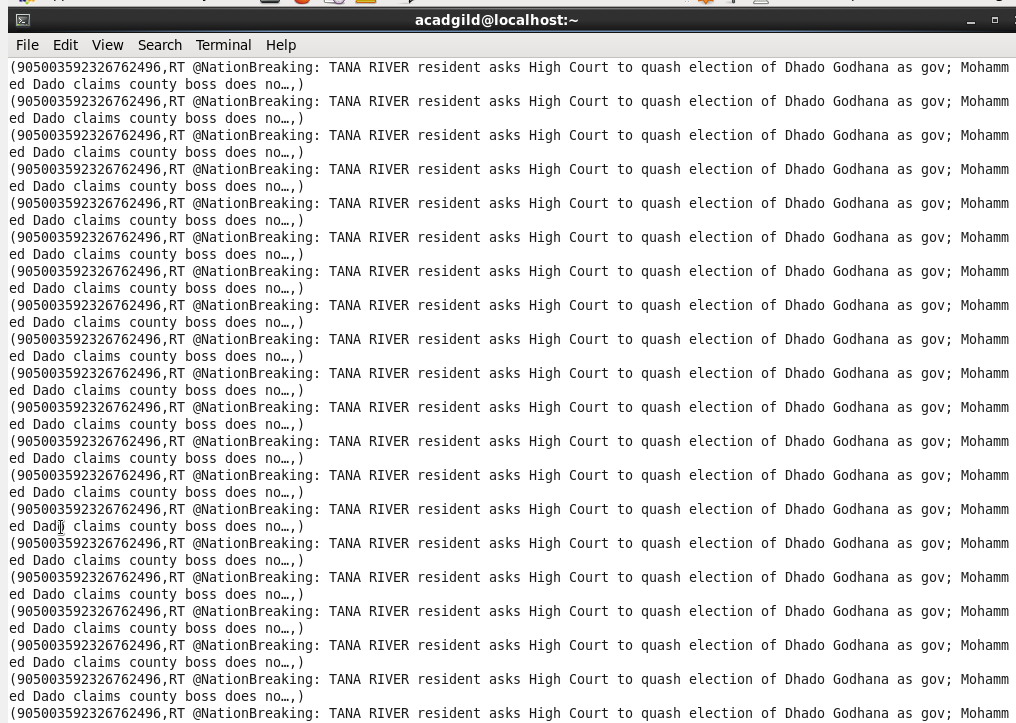
**Describe** command specifies its schema as follows:



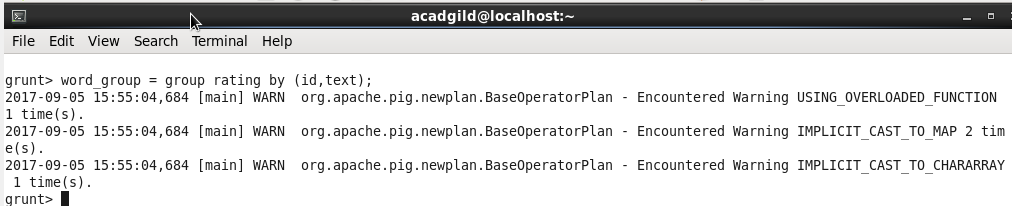
Dumping the relation rating as follows:



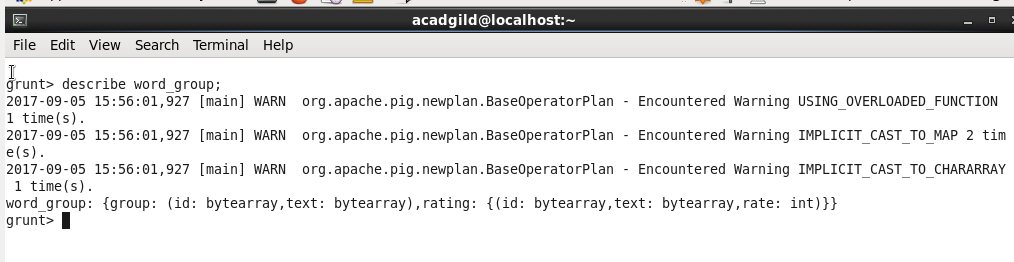
Snippet of the dump:



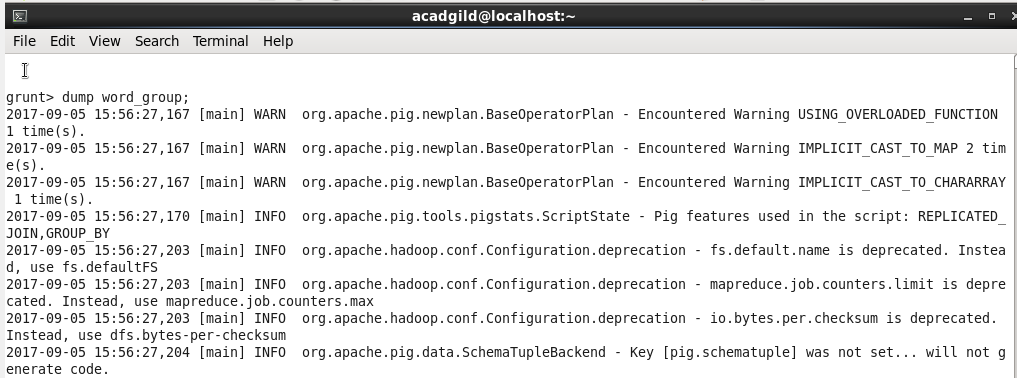
Step 8: Now, we perform grouping on relation **rating** by id and text fields as follows:



**Describe** command mentions the schema of the relation as follows:



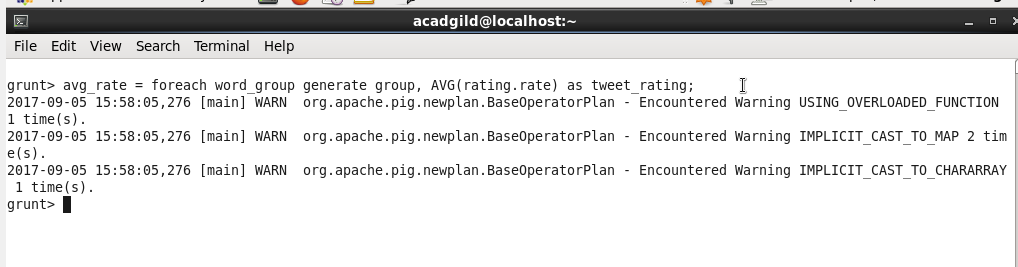
Dumping the relation as follows:



Snippet of the dump is as follows:

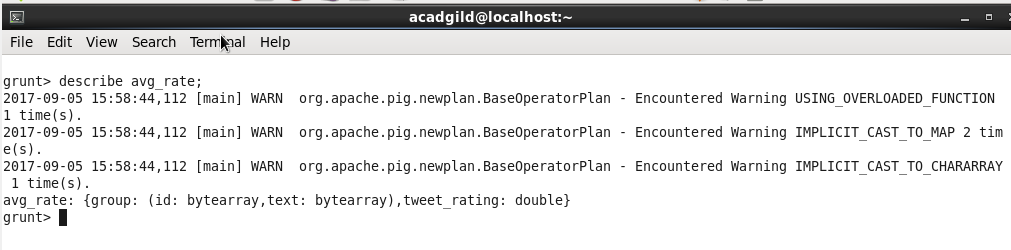


Step 9: Now, we calculate the average rating of tweet using the rating for each word as follows:

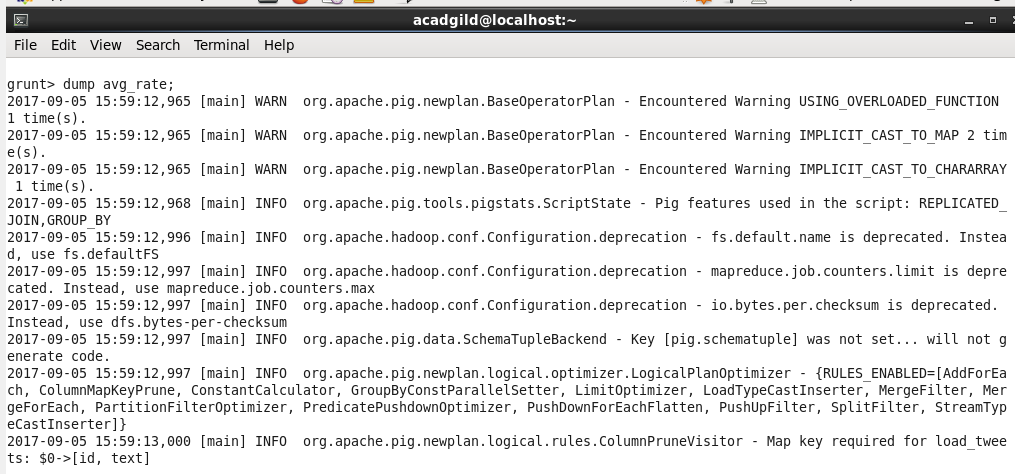


We used the **AVG** function for it.

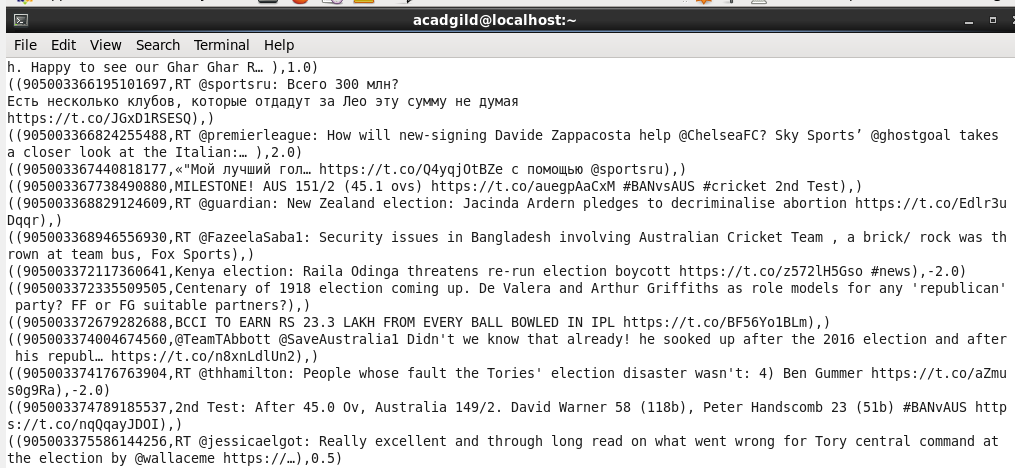
**Describe** command mentions the schema of the relation as follows:



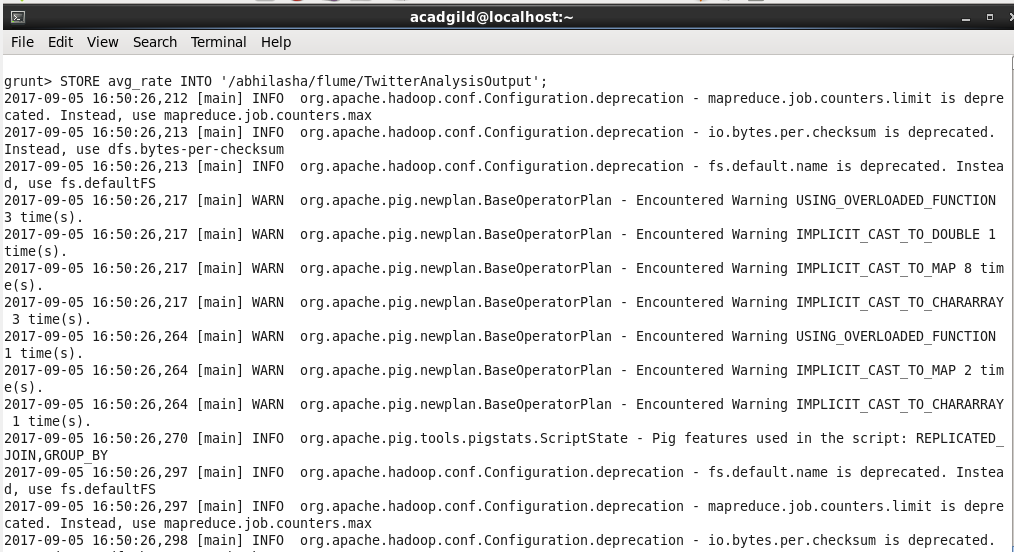
Dumping the relation as follows:



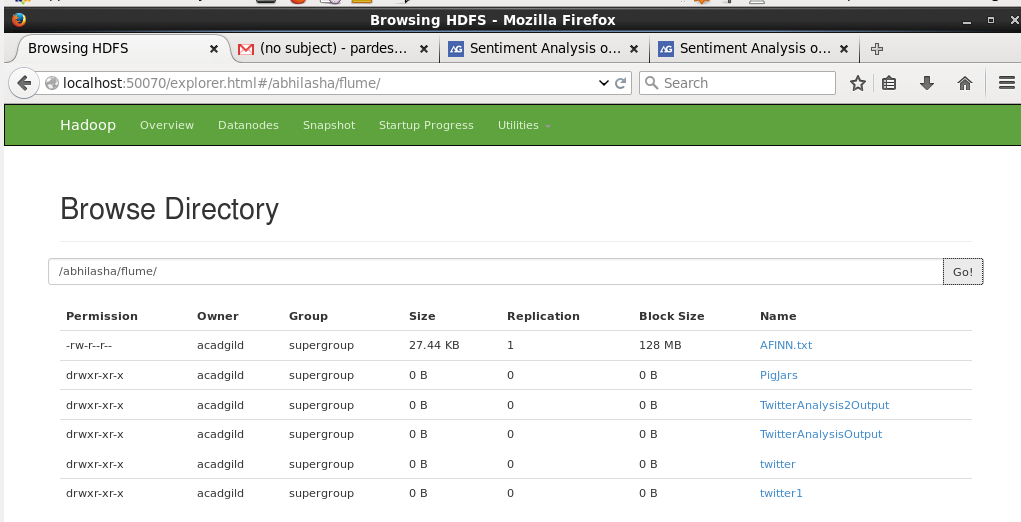
Snippet of the dump is:

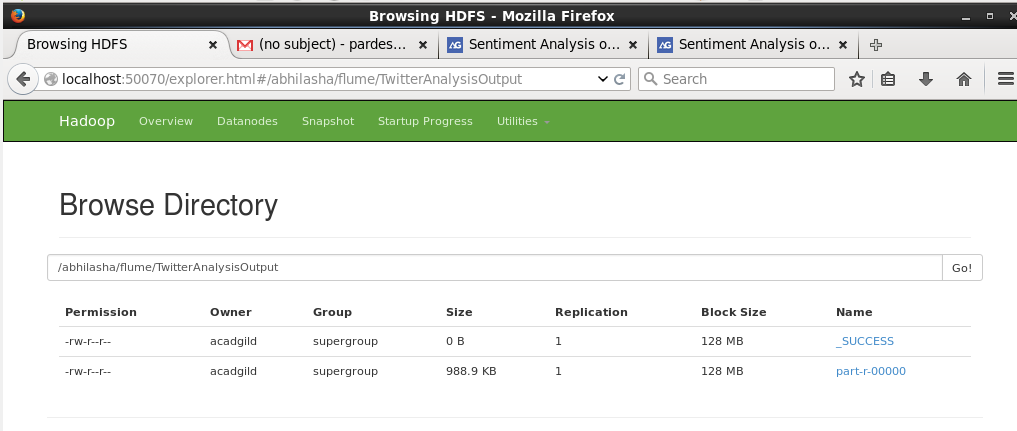


Step 10: Store the output into a file on HDFS. We use **STORE** command and specify the destination folder name as follows. The destination path mentioned is **/abhilasha/flume/TwitterAnalysisOutput.**



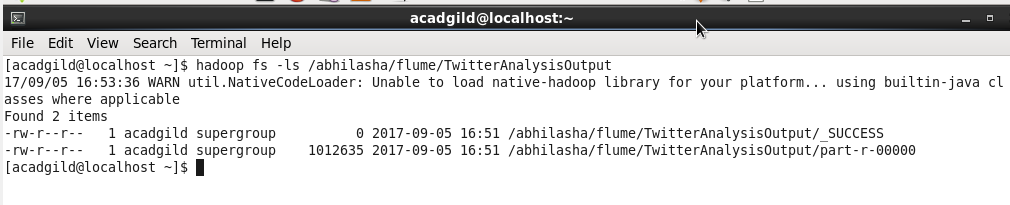
We can verify the result of **STORE** through HDFS UI as follows:



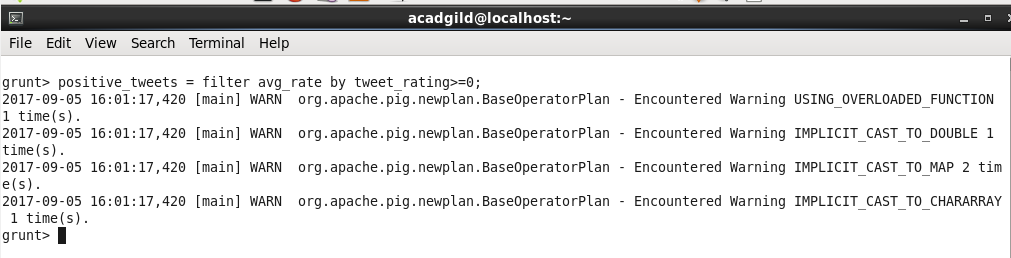


In the output folder, part-r-00000 file contains the result.

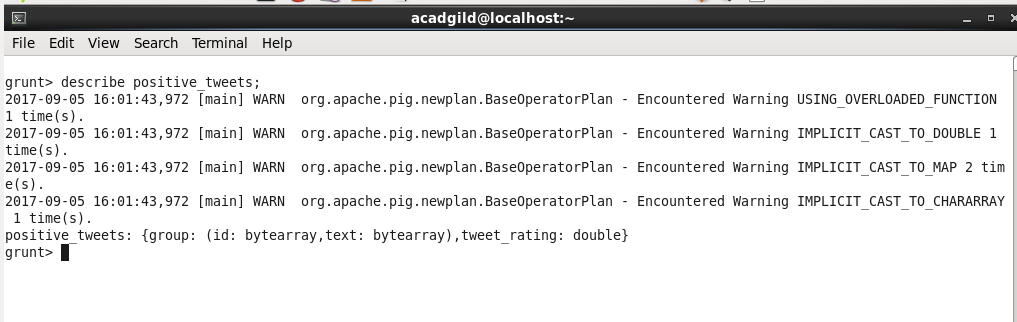
We can also see the output file using **ls** command as follows:



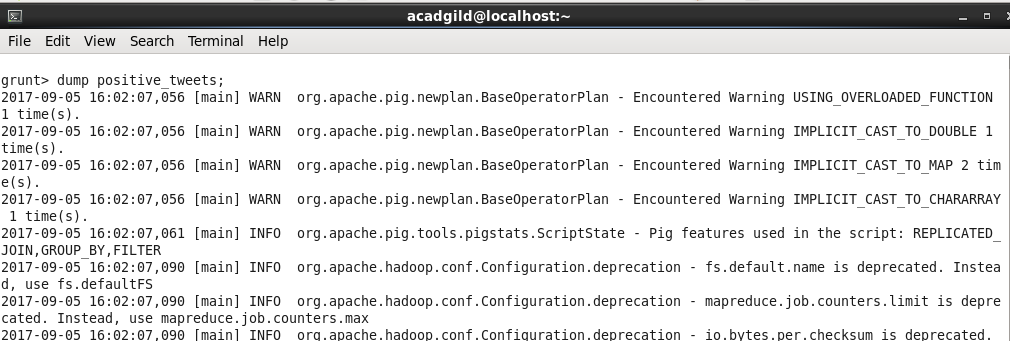
Step 11: Now, we filter the result to get only those tweets that have average rating greater than 0 as follows:



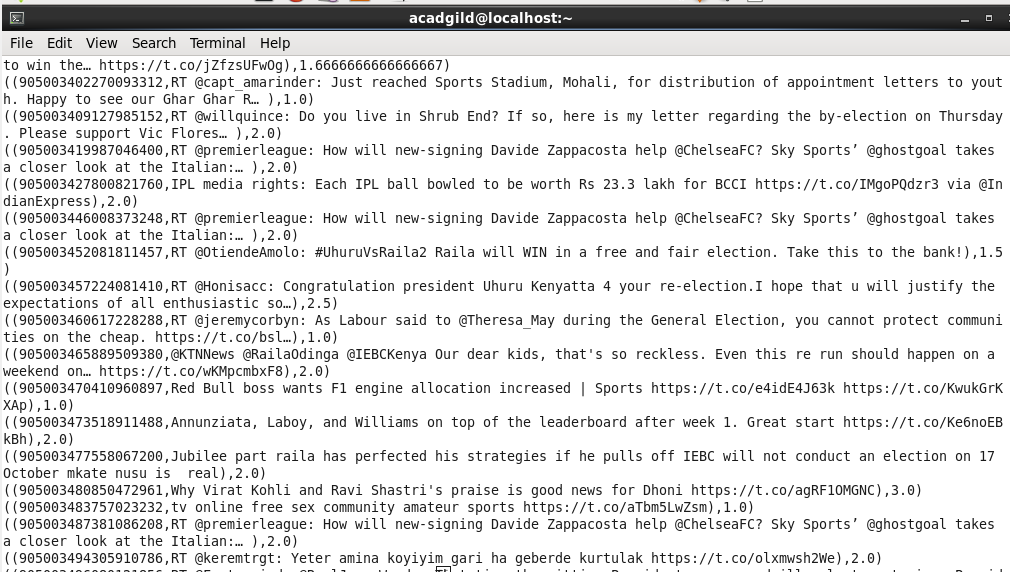
**Describe** command specifies schema as follows:



Dumping the relation as follows:

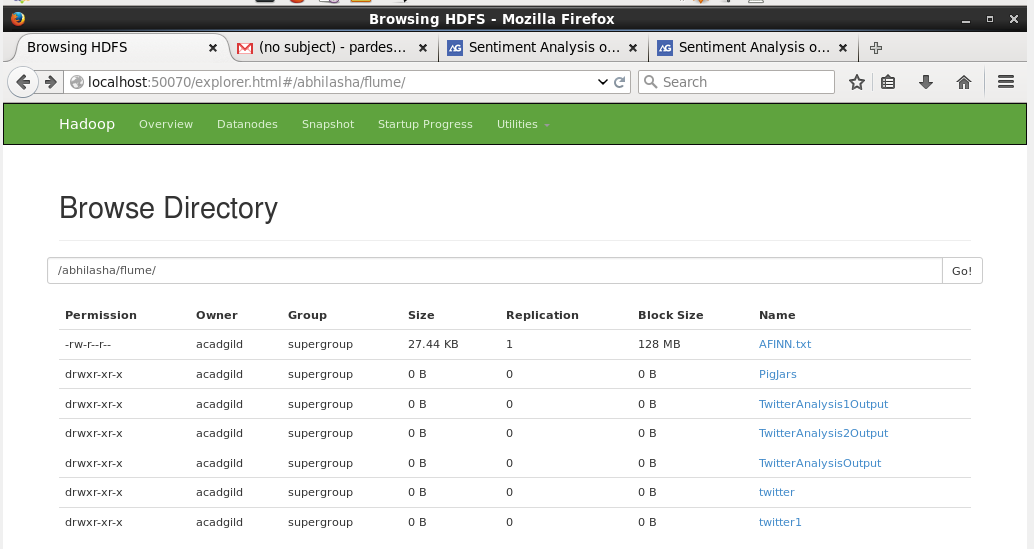


Snippet of the dump is as follows:

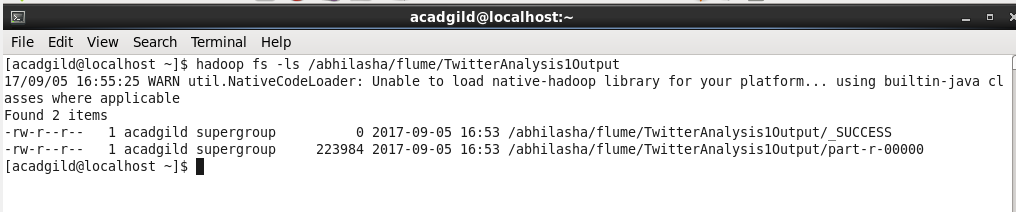


Step 12: Store the output into a file on HDFS. We use **STORE** command and specify the destination folder name as follows. The destination path mentioned is **/abhilasha/flume/TwitterAnalysis1Output.** The command used is **STORE positive\_tweets INTO /abhilasha/flume/TwitterAnalysis1Output.**

We can verify the result of **STORE** through HDFS UI as follows:



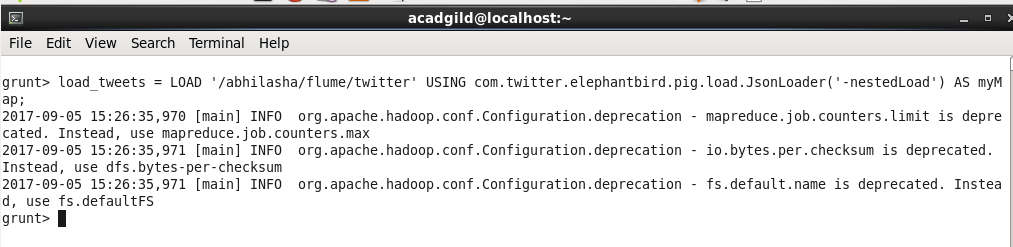
We can also verify if using **ls** command as follows:



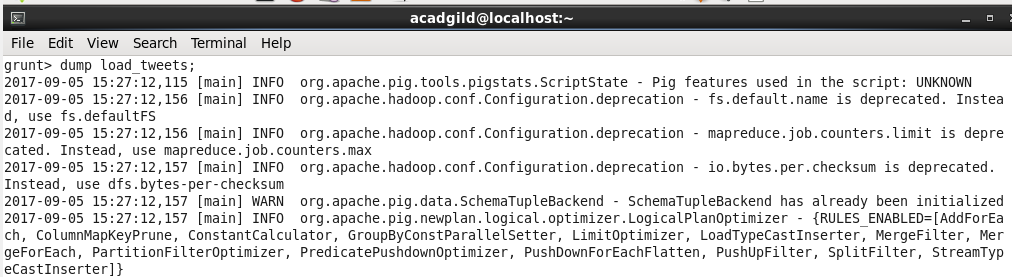
In the output folder, part-r-00000 file contains the result.

**Analysis 2:**

Step 1: Load the data in pig using **LOAD** command as follows. We specify the path in HDFS from where data is to be loaded. The path mentioned is **/abhilasha/flume/twitter.**  This will load all the data files present in this directory into pig.



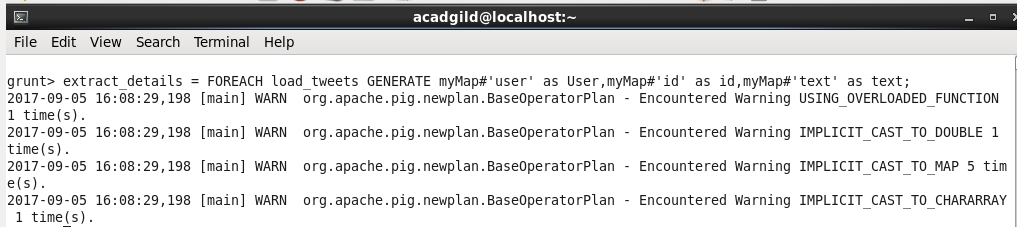
We use the **dump** command to see the data on console, that is to dump the relation, whose name is specified, on console.



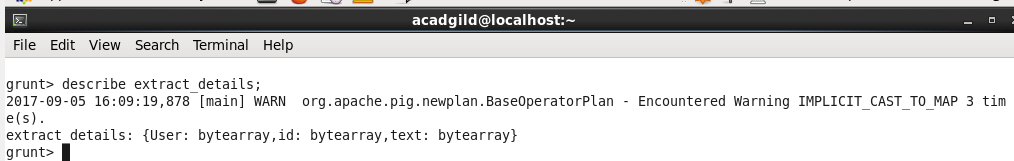
Snippet of the dump is



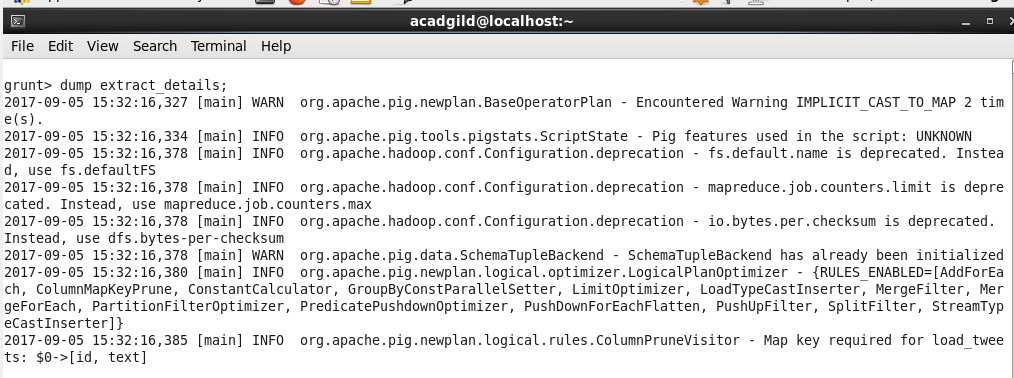
Step2: Now, we extract tweeted user\_details, id and the tweet text from the above tweets. The Pig statement used for the same is as shown in the snapshot. We use **FOREACH** to iterate over all the tweets and extract the required data in the relation **extract\_details.**



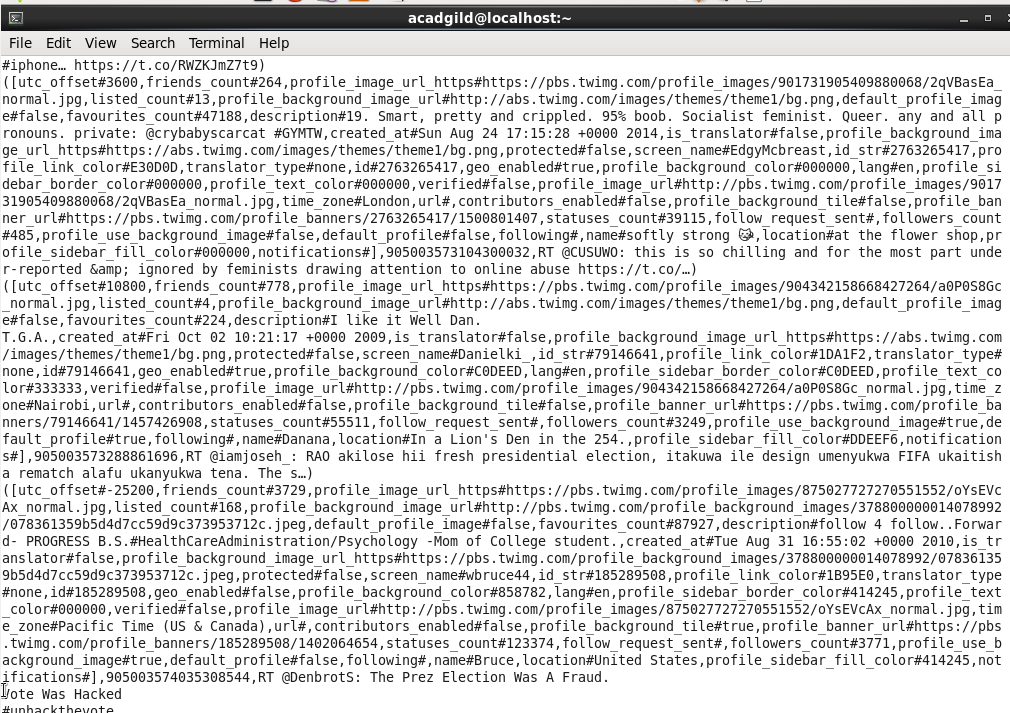
Using **describe** command, we can get the schema of the relation as follows:



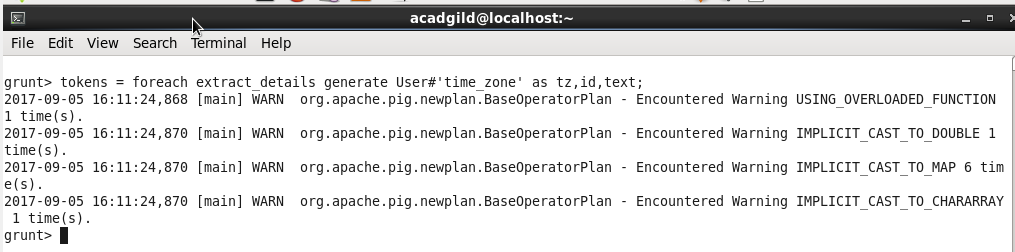
Performing dump of **extract\_details** as follows:



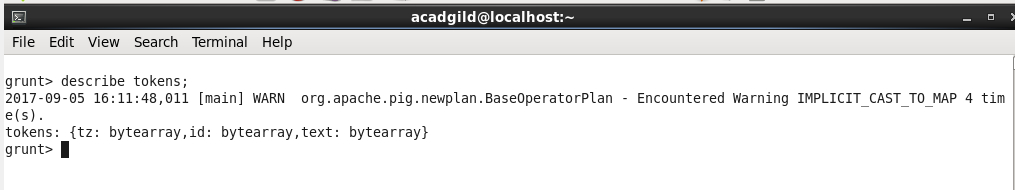
Snippet of output of dump of **extract\_details** is



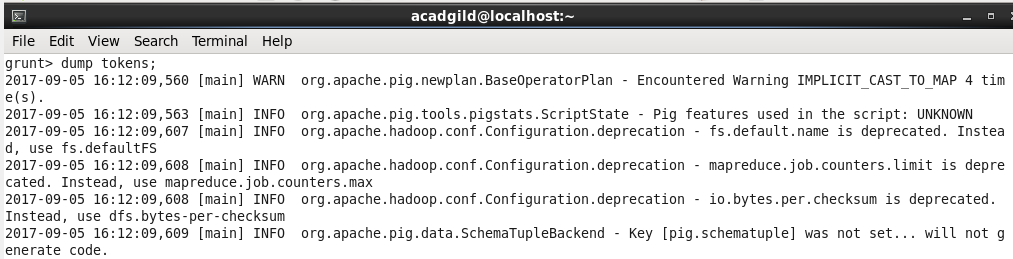
Step 3:  Time\_zone is part of the user details and id, tweet are extracted separately from the tweet. Hence, we now extract id, text and only time\_zone from all the user details as follows:



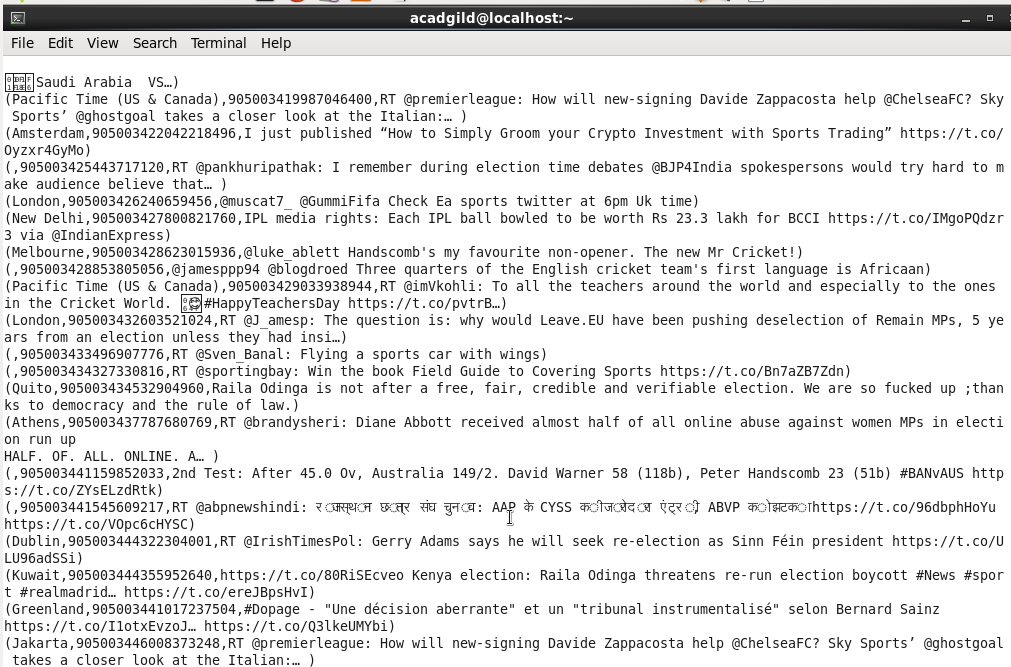
Using **describe** command, we can get the schema of the relation **tokens** as follows:



Performing dump of **tokens**



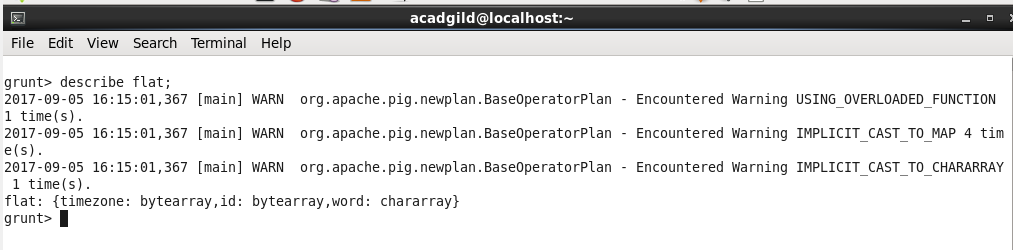
Snippet of the dump is as follows:



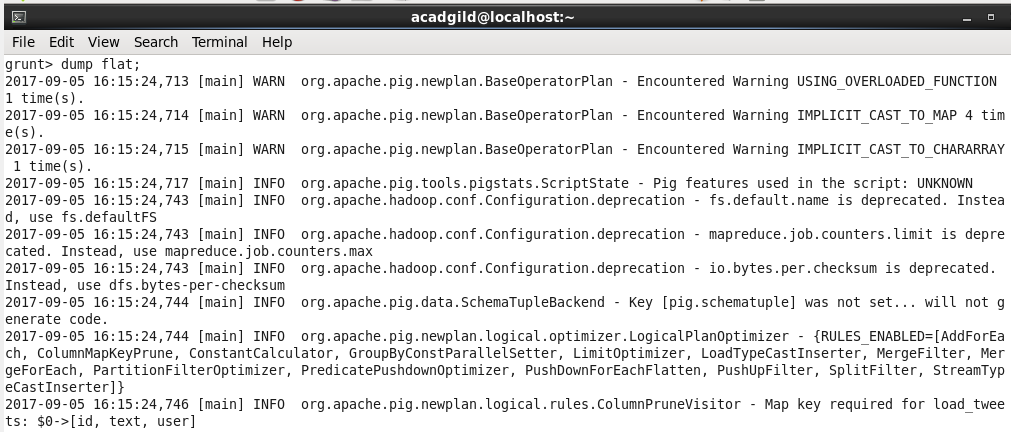
Step 4: Next is to extract words from text using **TOKENIZE.** We also use **FLATTEN** on the output of tokenize. **FLATTEN** will remove all the words from the nesting of bags/tuples and provide a flat structure. We have also used the alias for it as **word.**



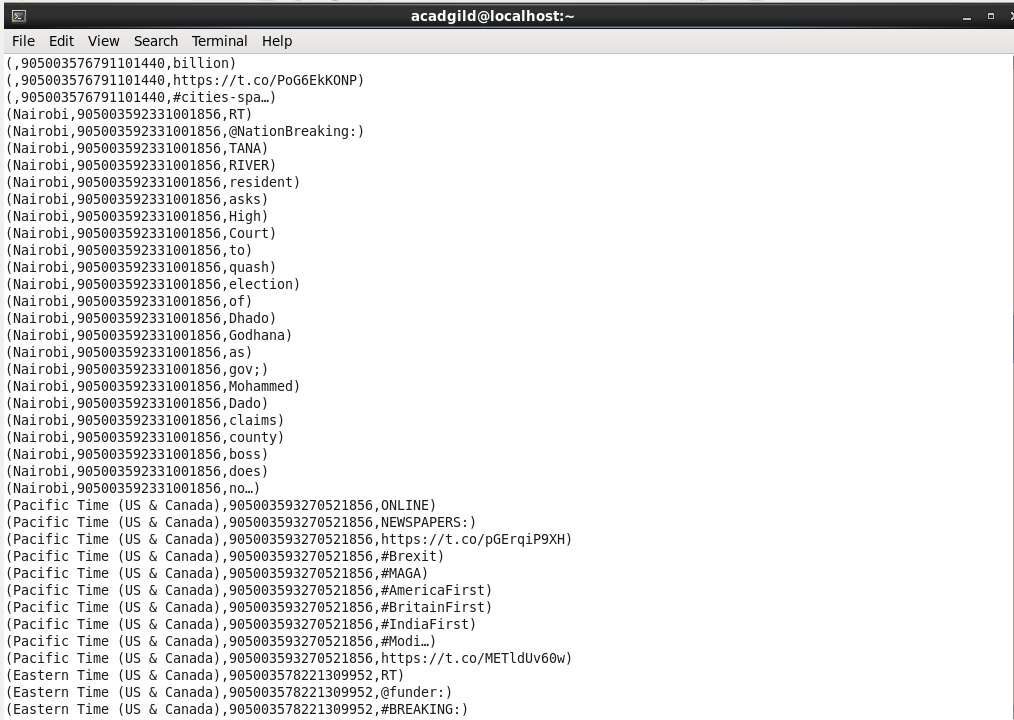
Using **describe** command, we can get the schema of the relation **flat** as follows:



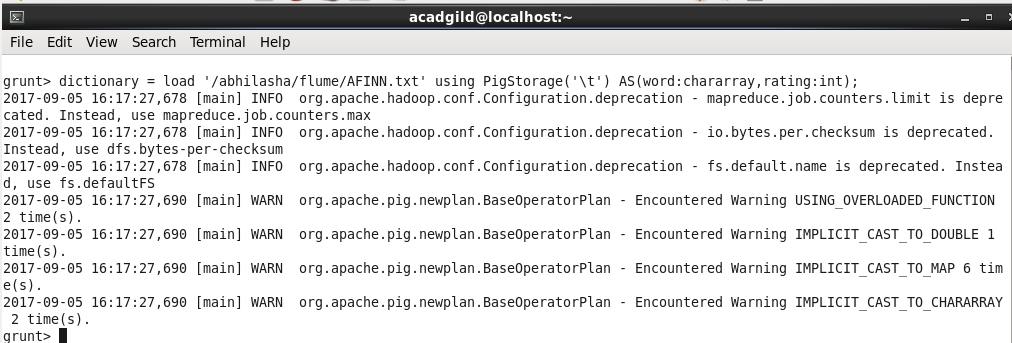
Performing dump of **flat** as follows:



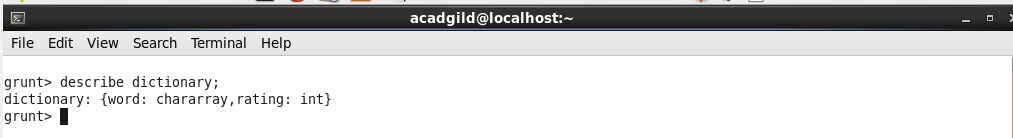
Snippet of the dump is:



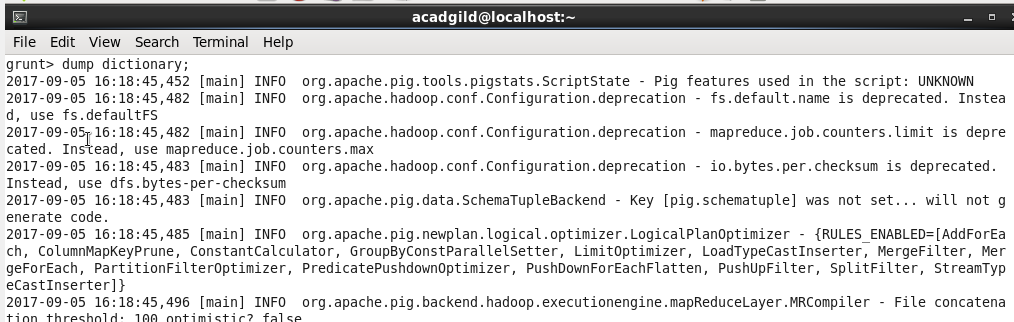
Step 5: As seen earlier, AFINN.txt is already present in HDFS at **/abhilasha/flume.** We now load it in pig as follows. We have also mentioned its schema, the tuple is made of word and rating.



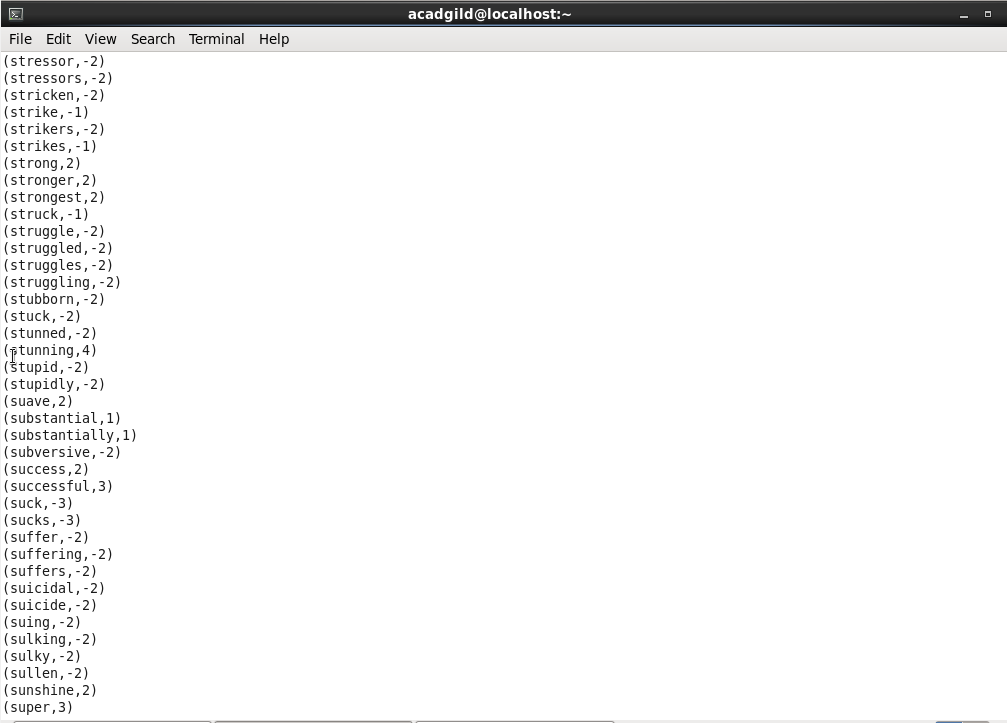
**Describe** command also verifies its schema.



Dumping the relation dictionary is as follows:

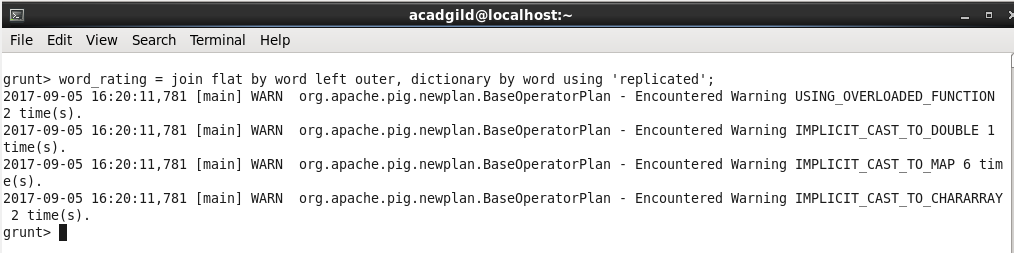


Dump output:

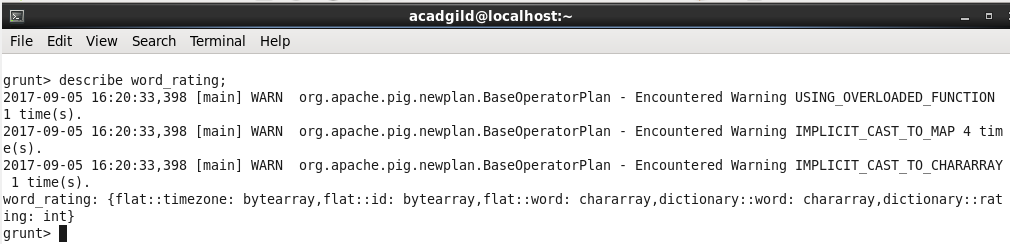


Step 6: Next, we perform a join of the relations **tokens** and **dictionary**, as we need the rating for each word in **tokens** relation. The using **'replicated'** tells Pig to use the fragment-replicate algorithm to execute this join. The second input listed in the join (in this case, **dictionary**) is always the input that is loaded into memory.

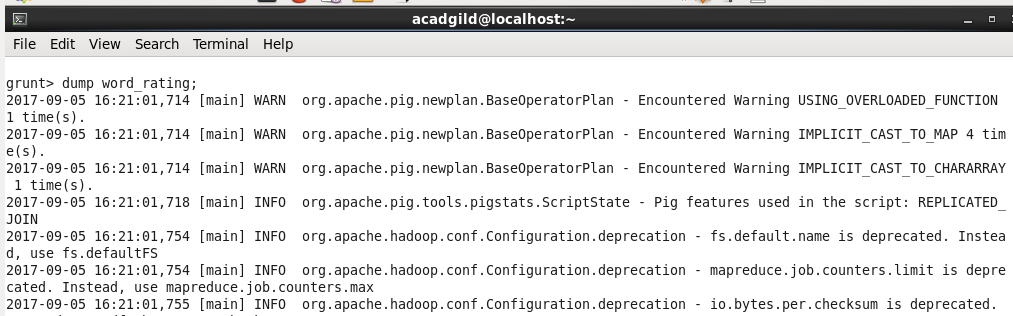
The syntax for the same is as follows:



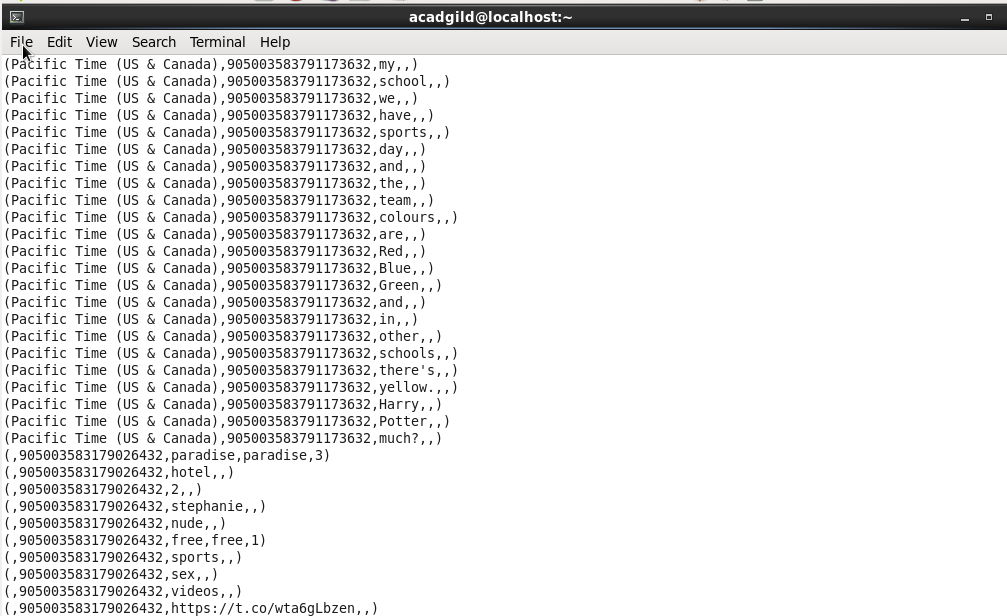
 The schema of the **word\_rating** relation, which has the joined content of both the flat relation, containing the time\_zone, id, word and the dictionary rating for the word is as follows:



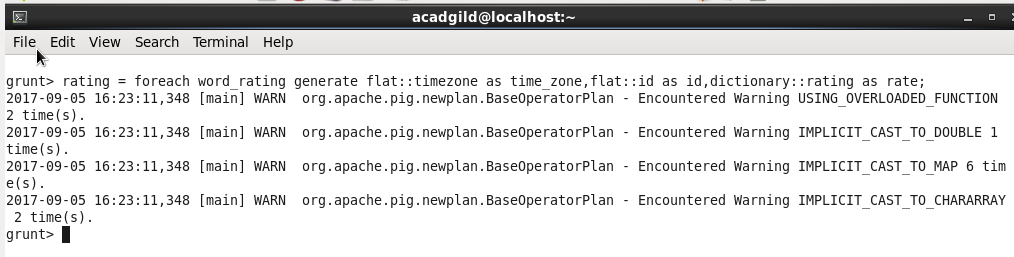
Performing dump of the relation:



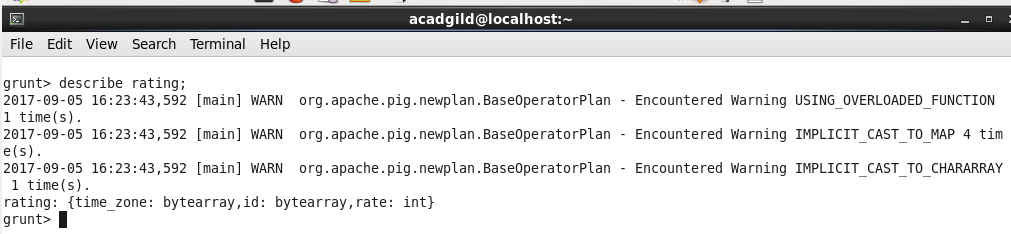
Dump output:



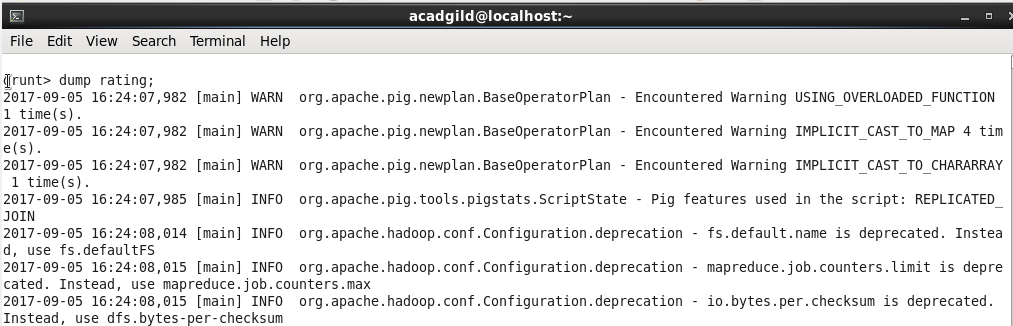
Step 7: Now we extract the time\_zone, id and the rating of the word in a relation and name it as rating.



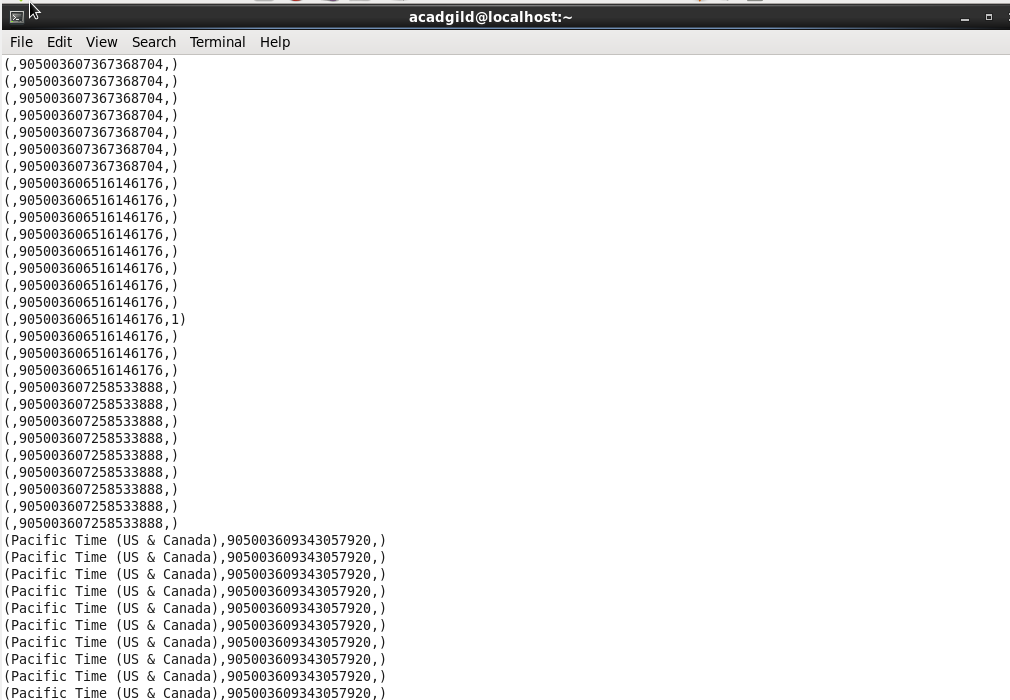
**Describe** command specifies its schema as follows:



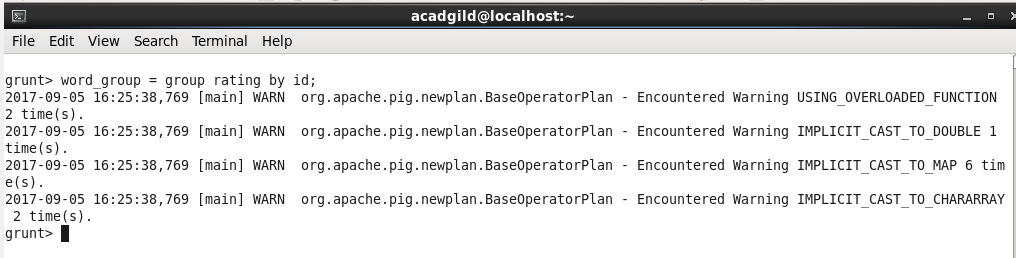
Performing dump of relation:



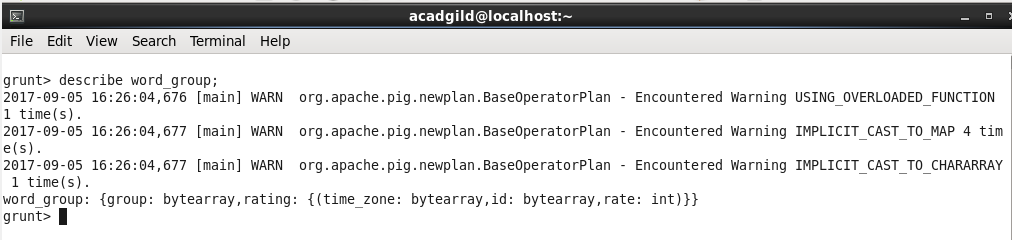
Snippet of dump is:



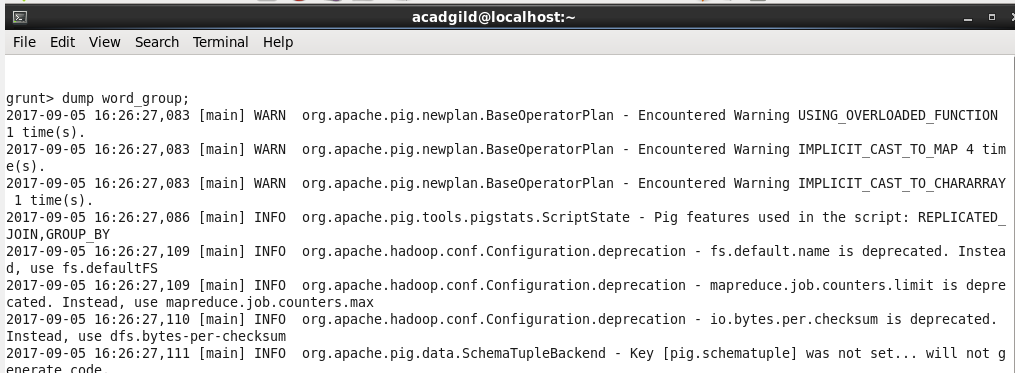
Step 8: Now, we group the records by id as follows:



**Describe** command specifies its schema as follows:



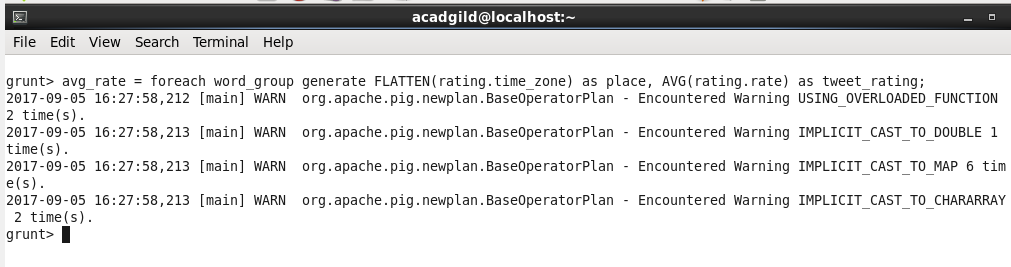
Performing dump of the relation:



Dump output:

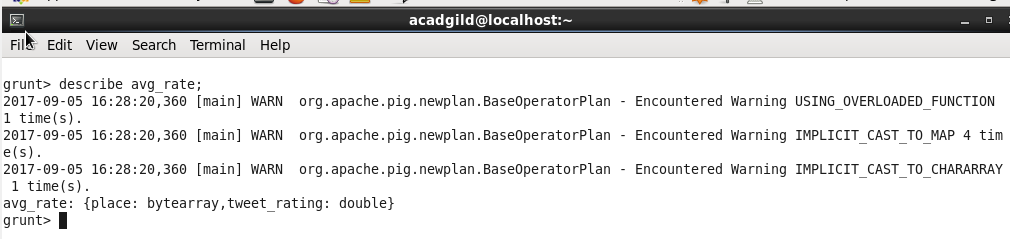


Step 9: Now, we calculate the average rating of tweet using the rating for each word as follows:

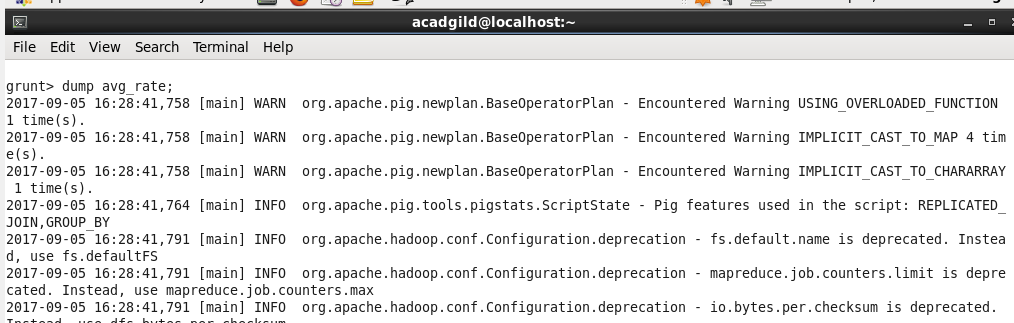


We used the **AVG** function for it.

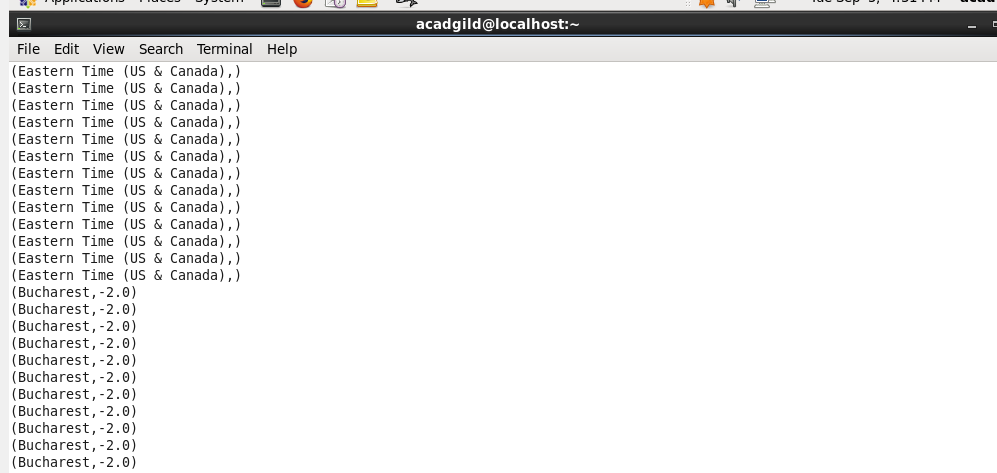
**Describe** command mentions the schema of the relation as follows:



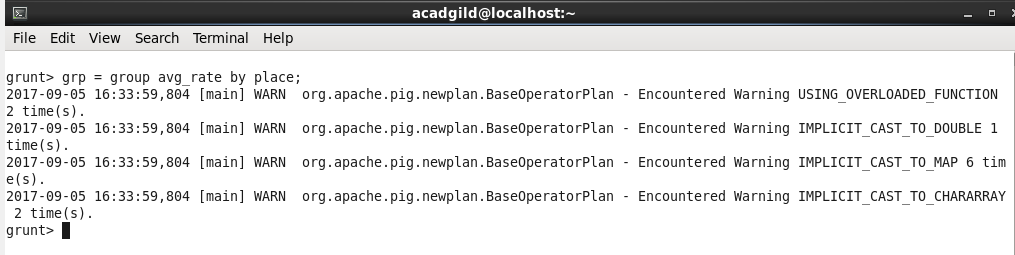
Dumping the relation as follows:



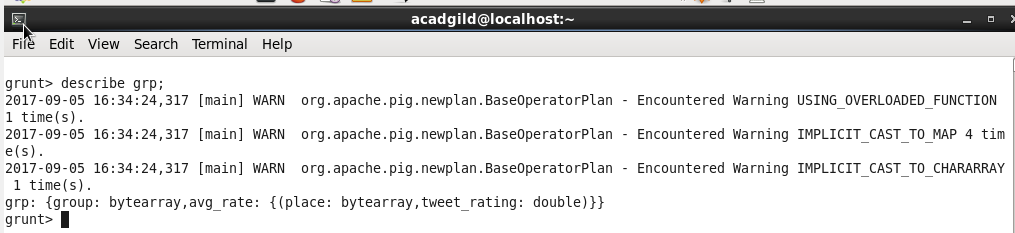
Snippet of the dump is:



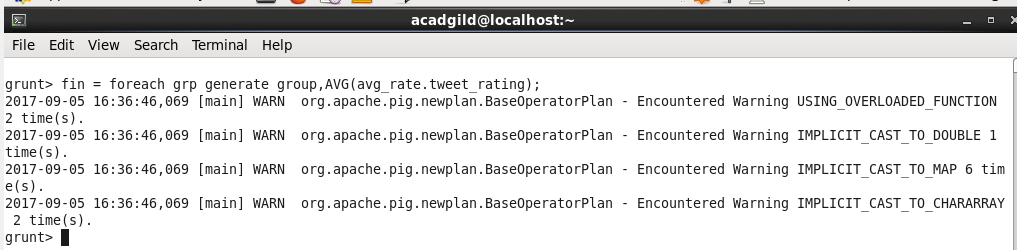
Step 10: Group the relation by place as follows:



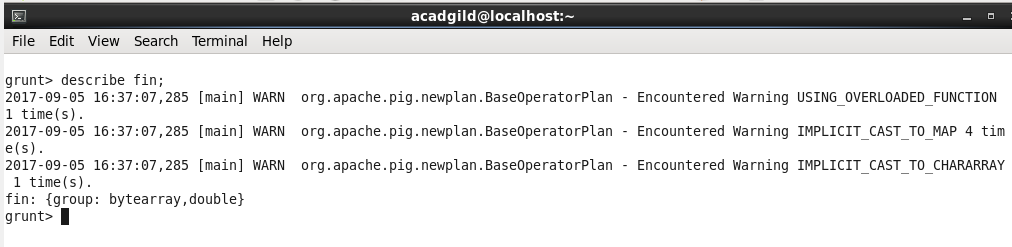
**Describe** command mentions the schema of the relation as follows:



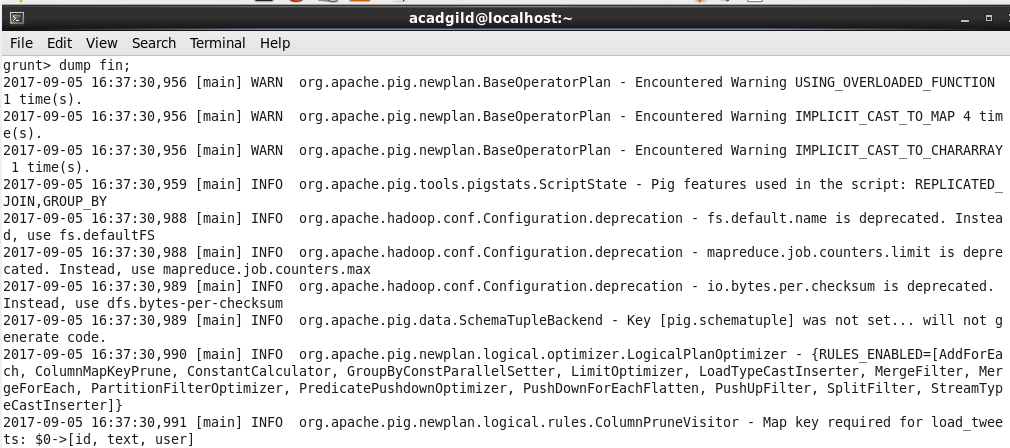
Step 11: We have time\_zones and the rating of the tweets of that time\_zone. So, we can now perform Average operation on the rating to get the time\_zone and the average rating of the people for the topic.



**Describe** command mentions the schema of the relation as follows:



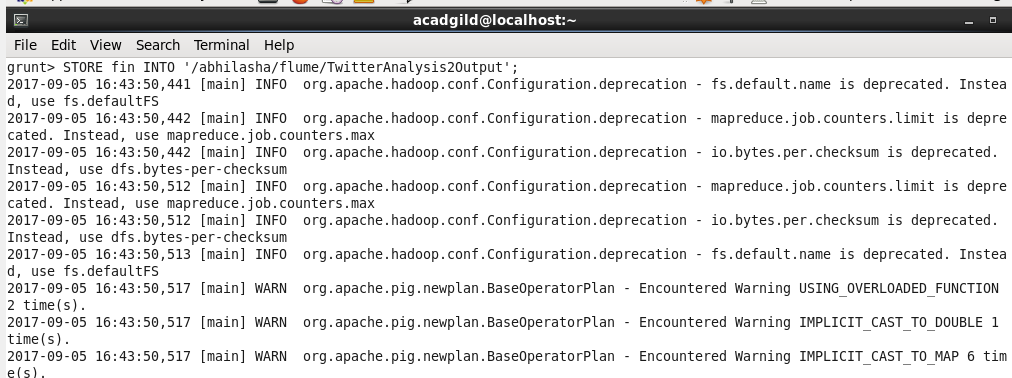
Dumping the relation as follows:



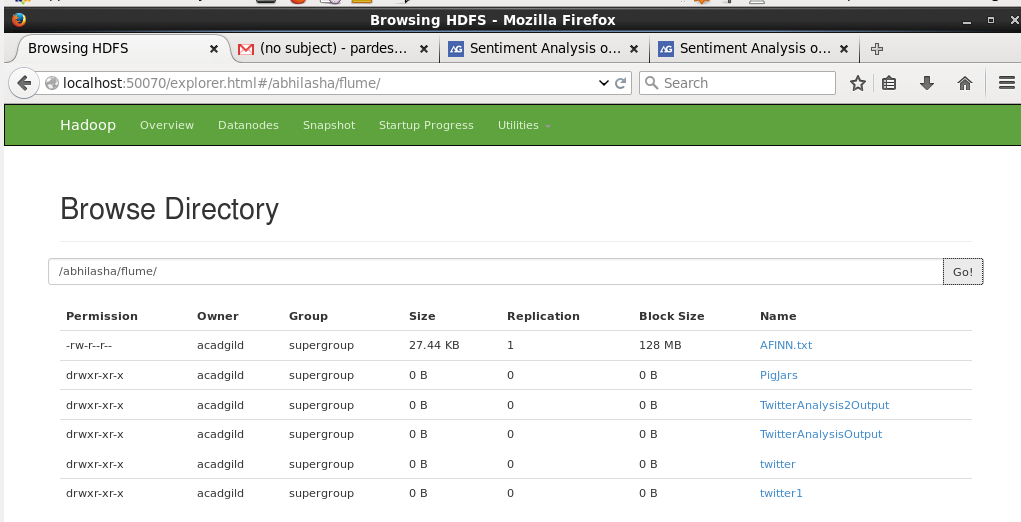
Snippet of the dump is:



Step 12: Store the output into a file on HDFS. We use **STORE** command and specify the destination folder name as follows. The destination path mentioned is **/abhilasha/flume/TwitterAnalysis2Output.**



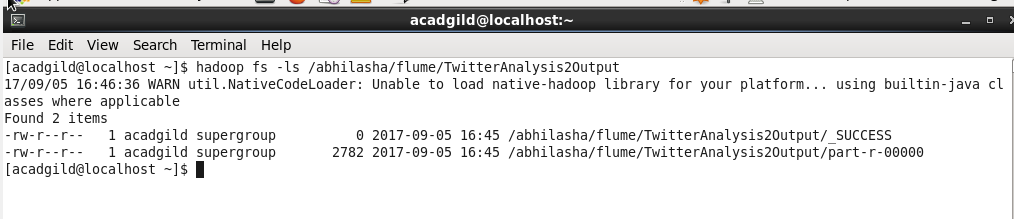
We can verify the result of **STORE** through HDFS UI as follows:



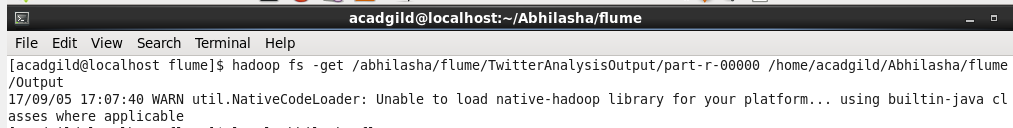


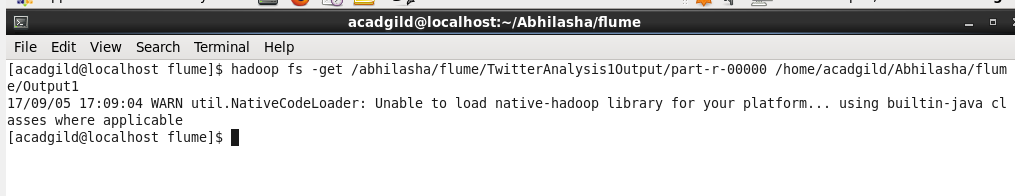
In the output folder, part-r-00000 file contains the result.

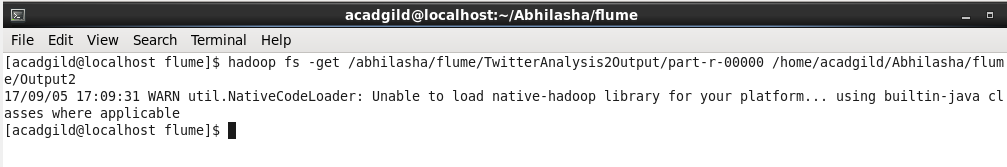
We can also see the output file using **ls** command as follows:



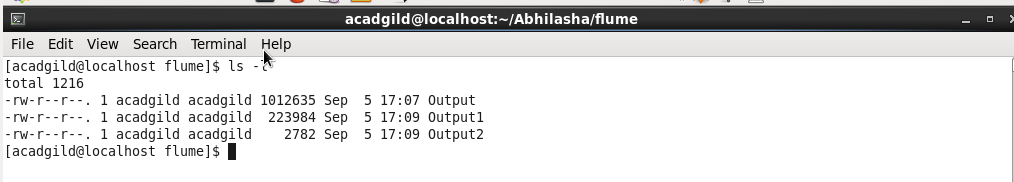
* We now get all the three output files from HDFS into local file system using **get** command as follows:







* These files on local system can be seen using **ls** command as follows:



We have renamed the files while getting them to local system.

The file **part-r-00000** from first output folder **TwitterAnalysisOutput** is renamed as Output.

The file **part-r-00000** from first output folder **TwitterAnalysis1Output** is renamed as Output1.

The file **part-r-00000** from first output folder **TwitterAnalysis2Output** is renamed as Output2.