Big Data Hadoop 'Session 11: ADVANCE HBASE Assignment 1'

Task 1

Answer in your own words with exam ple.

1.What is NoSQL data base?

- NoSQL database is design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats.
- NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built.
- NoSQL databases are especially useful for working with large sets of distributed data.
- Data can be stored in a schema-less or free-form fashion.

2. Types of Nosql Databases?

Eg

- ***Document databases*** *(e.g. CouchDB, MongoDB)*
- ***Key-value stores*** *(e.g. Redis, Riak)*
- ***Wide column stores*** *(e.g. HBase, Cassandra)*
- ***Graph databases*** *(e.g. Neo4j)*

3. CAP Theorem?

Consistency means that data is the same across the cluster, so data can be read or write from/to any node and get the same data.

Availability means the ability to access the cluster even if a node in the cluster goes down.

Partition tolerance means that the cluster continues to function even if there is a "partition" (communication break) between two nodes (both nodes are up, but can't communicate)

In order to get both availability and partition tolerance, you have to give up consistency. Consider if we have two nodes, X and Y, in a master-master setup. Now, there is a break between network communication between X and Y, so they can't sync updates. At this point we can either:

A) Allow the nodes to get out of sync (giving up consistency), or

B) Consider the cluster to be "down" (giving up availability)

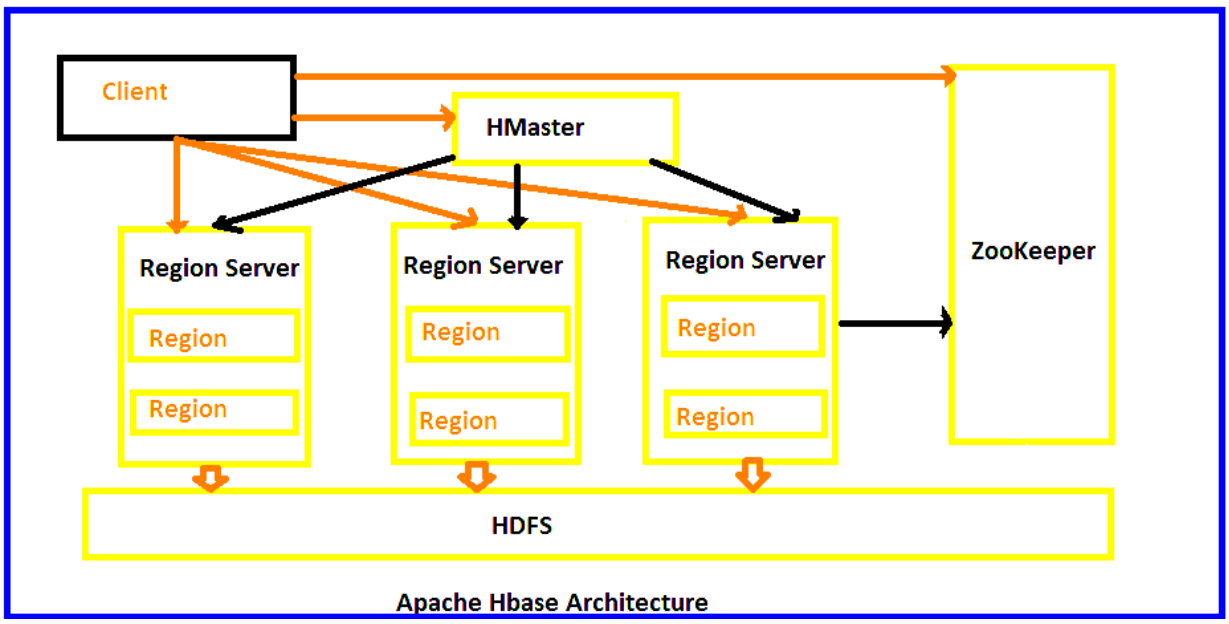
All the combinations available are:

CA - data is consistent between all nodes - as long as all nodes are online - and data can be read/write from any node and be sure that the data is the same, but if you ever develop a partition between nodes, the data will be out of sync (and won't re-sync once the partition is resolved).

CP - data is consistent between all nodes, and maintains partition tolerance (preventing data desync) by becoming unavailable when a node goes down.

AP - nodes remain online even if they can't communicate with each other and will resync data once the partition is resolved, but you aren't guaranteed that all nodes will have the same data (either during or after the partition)

4. HBase Architecture?



HBase architecture consists mainly of four components as follows:

1) HMaster
   - It is master server to monitor all Region Server instances present in the cluster and acts as an interface for all the metadata changes.
   - HMaster assigns regions to region servers.
   - It has features like controlling load balancing and failover to handle the load over nodes present in the cluster.
2) HRegionserver
   - When Region Server receives writes and read requests from the client, it assigns the request to a specific region, where actual column family resides.
   - Communicate with the client and handle data-related operations.
3) HRegions
   - HRegions are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families.
   - It contains multiple stores, one for each column family.
   - It consists of mainly two components, which are Memstore and Hfile.
4) Zookeeper

- Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization
- Clients communicate with region servers via zookeeper.
- Master and HBase slave nodes (region servers) registered themselves with ZooKeeper. The client needs access to ZK(zookeeper) quorum configuration to connect with master and region servers.

4. HBase vs RDBMS ?

HBase	RDBMS
HBase is the column-oriented database.	R DBMS is row-oriented.
HBase is less restrictive on defining scheme, adding columns on the fly is possible	Schema of RDBMS is more restrictive about schema it much be pre defined
HBase supports scale out. It means while we need memory processing power and more disk, we need to add new servers to the cluster rather than upgrading the present one.	RDBMS supports scale up. That means while we need memory processing power and more disk, we need upgrade same server to a more powerful server, rather than adding new servers.
HBase supports both structured and nonstructural type of data.	RDBMS is suited for only structured data.

Task2

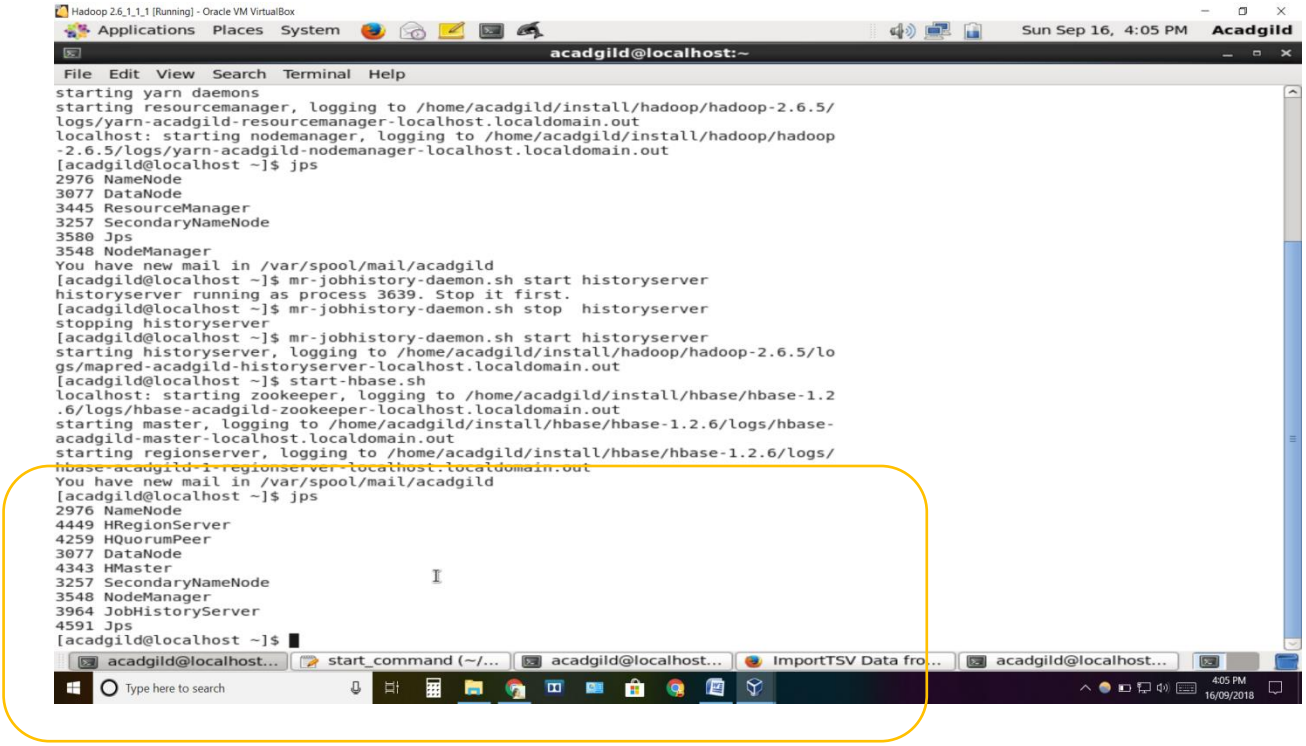
Execute blog present in below link

https://acadgild.com/blog/importtsv-data-from-hdfs-into-hbase/

Steps to Practical Execution

Start all the Hadoop and HBase daemons using below command

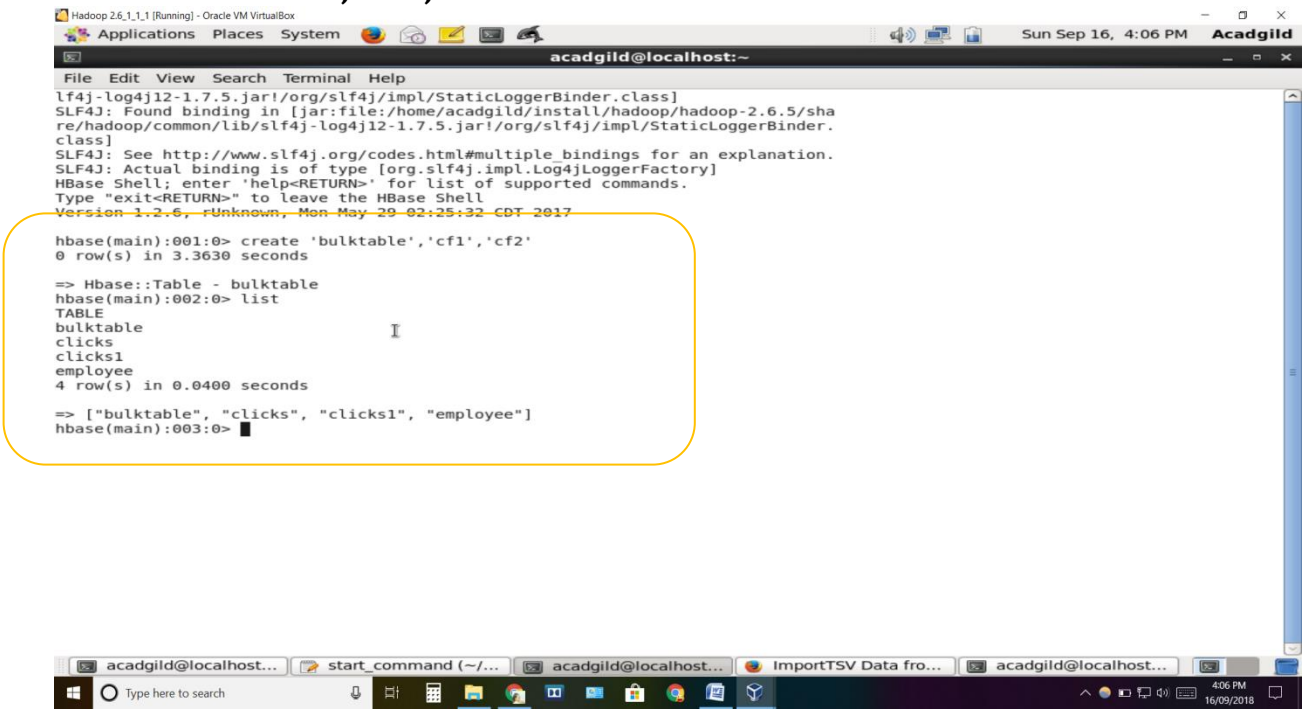
1) start-all.sh
2) start mr-historyserver-daemon.sh.
3) start HMaster Hstart-Hbase.sh



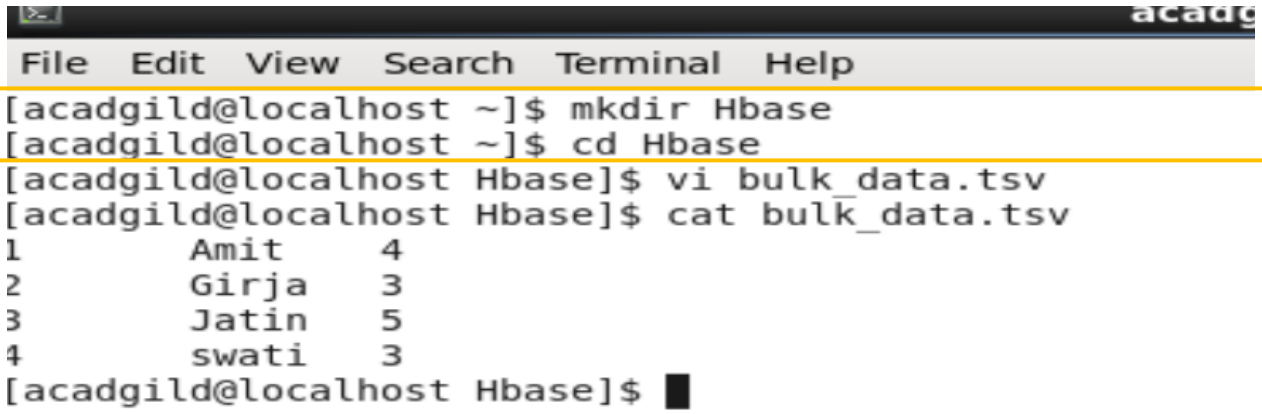
Step1:

Inside Hbase shell give the following command to create table along with 2 column family.

Create 'bulktable', 'cf1', 'cf2'



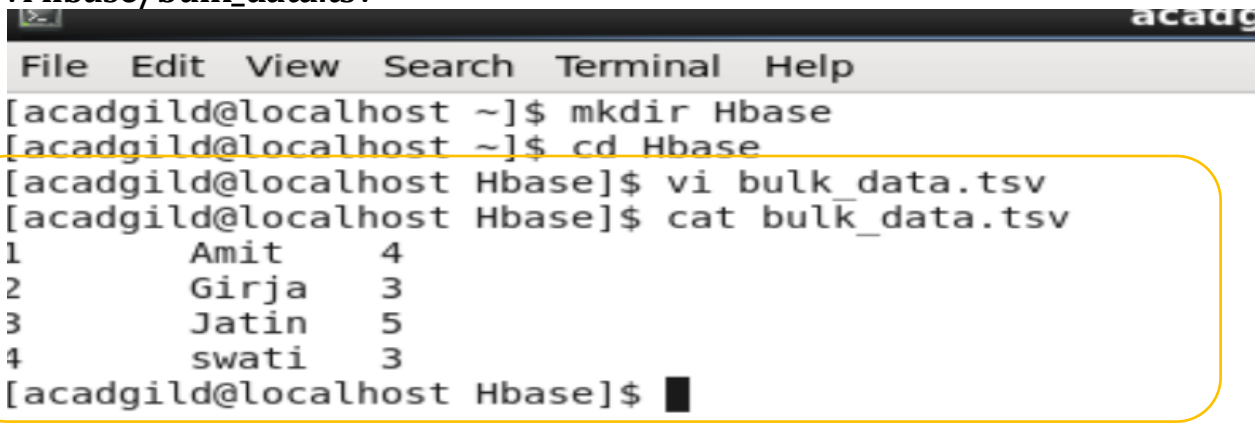
Step2 : make a directory for Hbase in the local drive



Step3:

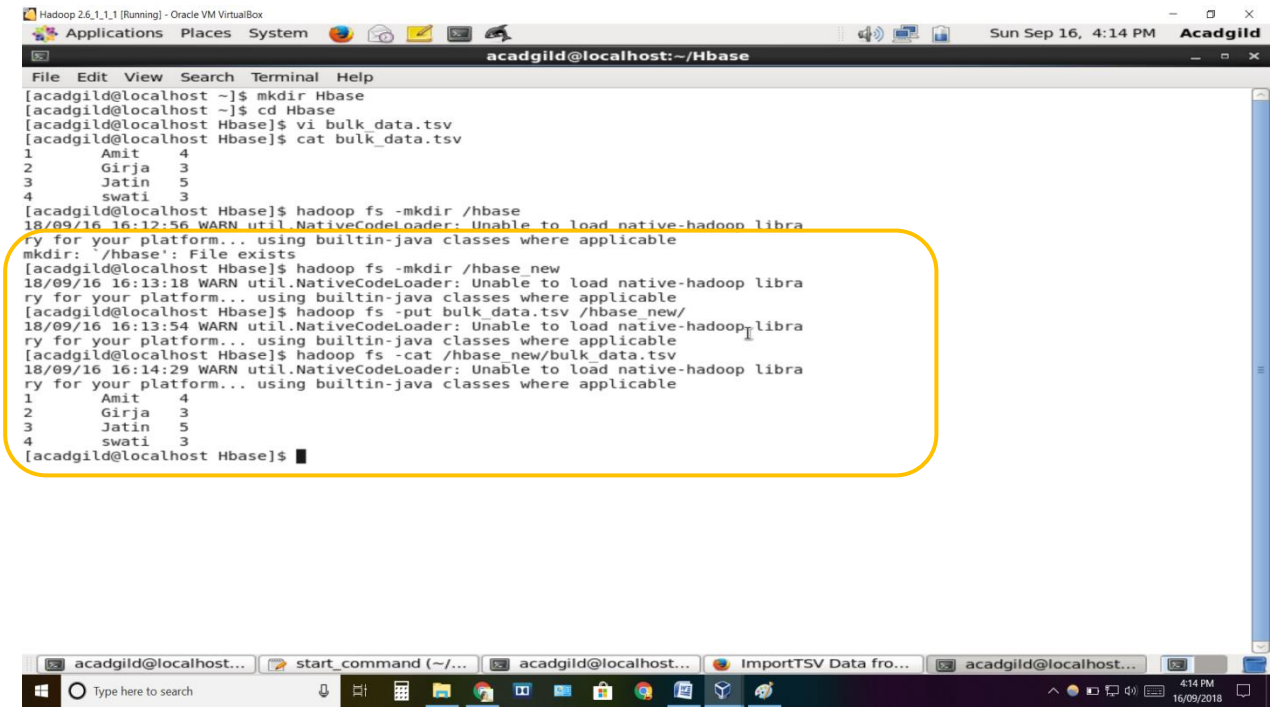
Create a file inside the HBase directory named bulk_data.tsv with tab separated data inside using below command in terminal.

vi hbase/bulk_data.tsv



Step4: copy the data inside HDFS using below command

1) hadoop fs -mkdir /hbase_new
2) hadoop fs -put bulk_data.tsv /hbase_new/
3) hadoop fs -cat /hbase_new/**bulk_data.tsv**



Step5:

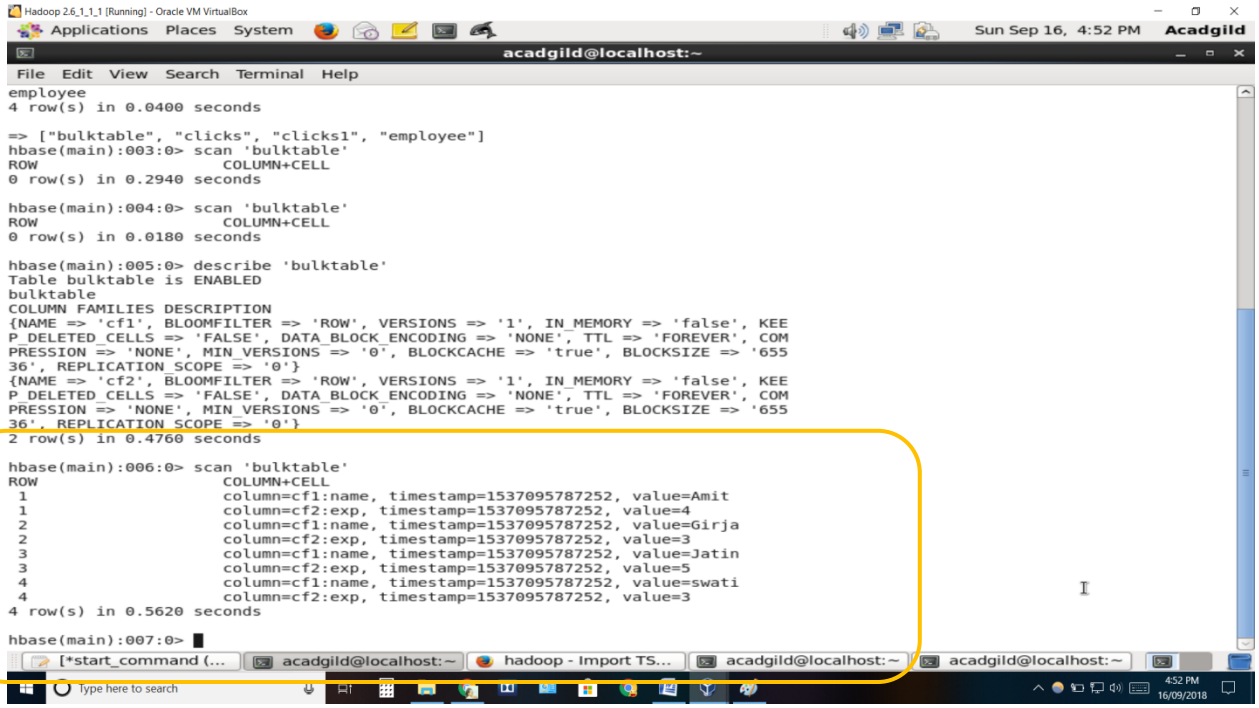
After the data is present now in HDFS.In terminal, we give the following command along with arguments<tablename> and <path of data in HDFS>

Command:

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv –Dimporttsv.columns=HBASE_ROW_KEY,cf1:name,cf2:exp bulktable /hbase/bulk_data.tsv

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv –Dimporttsv.columns = HBASE_ROW_KEY,cf1,cf2 bulktab
le /hbase_new/bulk_data.tsv
ERROR: No columns specified. Please specify with -Dimporttsv.columns=...
Usage: importtsv -Dimporttsv.columns=a,b,c <tablename> <inputdir>
```

Step6: Scan ‘bulkdata’


Hadoop 2.6_1_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Sun Sep 16, 4:52 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
employee
4 row(s) in 0.0400 seconds
=> ["bulktable", "clicks", "clicks1", "employee"]
hbase(main):003:0> scan 'bulktable'
ROW COLUMN+CELL
0 row(s) in 0.2940 seconds
hbase(main):004:0> scan 'bulktable'
ROW COLUMN+CELL
0 row(s) in 0.0180 seconds
hbase(main):005:0> describe 'bulktable'
Table bulktable is ENABLED
bulktable
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEE
P_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '655
36', REPLICATION_SCOPE => '0'}
{NAME => 'cf2', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEE
P_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '655
36', REPLICATION_SCOPE => '0'}
2 row(s) in 0.4760 seconds
hbase(main):006:0> scan 'bulktable'
ROW COLUMN+CELL
1 column=cf1:name, timestamp=1537095787252, value=Amit
1 column=cf2:exp, timestamp=1537095787252, value=4
2 column=cf1:name, timestamp=1537095787252, value=Girja
2 column=cf2:exp, timestamp=1537095787252, value=3
3 column=cf1:name, timestamp=1537095787252, value=Jatin
3 column=cf2:exp, timestamp=1537095787252, value=5
4 column=cf1:name, timestamp=1537095787252, value=swati
4 column=cf2:exp, timestamp=1537095787252, value=3
4 row(s) in 0.5620 seconds
hbase(main):007:0>
[*start_command (...
acadgild@localhost:~
hadoop - Import TS...
acadgild@localhost:~
acadgild@localhost:~
Type here to search
4:52 PM
16/09/2018