

Task1.1

Explain the below concepts with an example in brief.

- **Nosql Databases**

Ans)NoSQL is a non-relational database management system, different from traditional relational database management systems in some significant ways. It is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users). These type of data storing may not require fixed schema, avoid join operations and typically scale horizontally.

NoSQL

- Stands for Not Only SQL
- No declarative query language
- No predefined schema
- Key-Value pair storage, Column Store, Document Store, Graph databases
- Eventual consistency rather ACID property
- Unstructured and unpredictable data
- Prioritizes high performance, high availability and scalability
- BASE Transaction

(A BASE system gives up on consistency.

Basically Available indicates that the system does guarantee availability, in terms of the CAP theorem.

Soft state indicates that the state of the system may change over time, even without input. This is because of the eventual consistency model.

Eventual consistency indicates that the system will become consistent over time, given that the system doesn't receive input during that time.)

Advantages :

High scalability
Distributed Computing
Lower cost
Schema flexibility, semi-structure data
No complicated Relationships

Disadvantages

No standardization
Limited query capabilities (so far)
Eventual consistent is not intuitive to program for

- **Types of Nosql Databases**

There are four general types (most common categories) of NoSQL databases. Each of these categories has its own specific attributes and limitations. There is not a single solutions which is better than all the others, however there are some databases that are better to solve specific problems.

1. Key-value stores

- i. Key-value stores are most basic types of NoSQL databases.
- ii. Designed to handle huge amounts of data.
- iii. Based on Amazon's Dynamo paper.
- iv. Key value stores allow developer to store schema-less data.
- v. In the key-value storage, database stores data as hash table where each key is unique and the value can be string, JSON, BLOB (Binary Large OBJec) etc.
- vi. A key may be strings, hashes, lists, sets, sorted sets and values are stored against these keys.

- vii. For example a key-value pair might consist of a key like "Name" that is associated with a value like "Robin".
- viii. Key-Value stores can be used as collections, dictionaries, associative arrays etc.
- ix. Key-Value stores follow the 'Availability' and 'Partition' aspects of CAP theorem.
- x. Key-Values stores would work well for shopping cart contents, or individual values like color schemes, a landing page URI, or a default account number.

Example of Key-value store DataBase: Redis, Dynamo, Riak. Etc

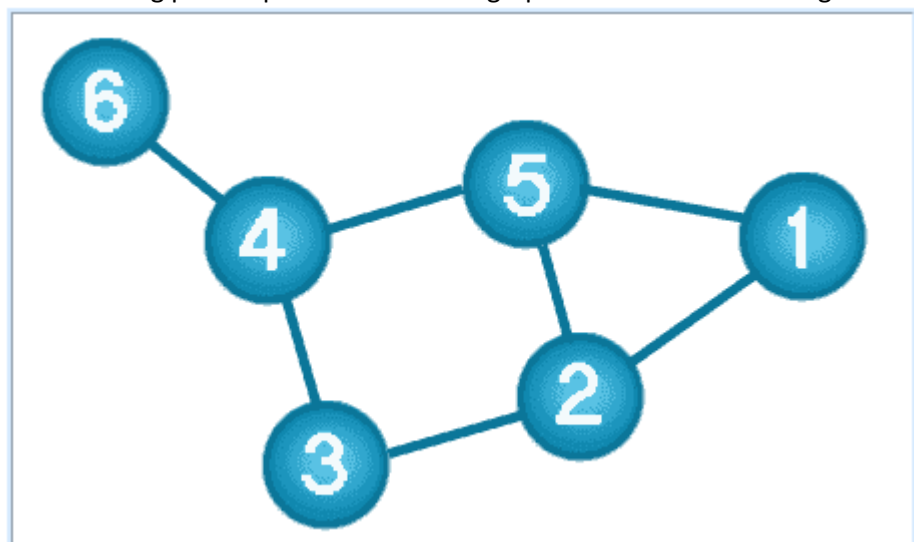
2. Column-oriented

- i. Column-oriented databases primarily work on columns and every column is treated individually.
- ii. Values of a single column are stored contiguously.
- iii. Column stores data in column specific files.
- iv. In Column stores, query processors work on columns too.
- v. All data within each column datafile have the same type which makes it ideal for compression.
- vi. Column stores can improve the performance of queries as it can access specific column data.
- vii. High performance on aggregation queries (e.g. COUNT, SUM, AVG, MIN, MAX).
- viii. Works on data warehouses and business intelligence, customer relationship management (CRM), Library card catalogs etc.

Example of Column-oriented databases: BigTable, Cassandra, SimpleDB etc.

3. Graph

- i. A graph data structure consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices.
- ii. The following picture presents a labeled graph of 6 vertices and 7 edges.



What is a Graph Databases?

- A graph database stores data in a graph.
- It is capable of elegantly representing any kind of data in a highly accessible way.
- A graph database is a collection of nodes and edges
- Each node represents an entity (such as a student or business) and each edge represents a connection or relationship between two nodes.
- Every node and edge are defined by a unique identifier.
- Each node knows its adjacent nodes.
- As the number of nodes increases, the cost of a local step (or hop) remains the same.
- Index for lookups.

Here is a comparison between the classic relational model and the graph model:

Relational model	Graph model
Tables	Vertices and Edges set
Rows	Vertices
Columns	Key/value pairs
Joins	Edges

4. Document oriented

- i. A collection of documents
- ii. Data in this model is stored inside documents.
- iii. A document is a key value collection where the key allows access to its value.
- iv. Documents are not typically forced to have a schema and therefore are flexible and easy to change.
- v. Documents are stored into collections in order to group different kinds of data.
- vi. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

Here is a comparison between the classic relational model and the document model:

Relational model	Document model
Tables	Collections
Rows	Documents
Columns	Key/value pairs
Joins	not available

- **CAP Theorem**

You must understand the CAP theorem when you talk about NoSQL databases or in fact when designing any distributed system. CAP theorem states that there are three basic requirements which exist in a special relation when designing applications for a distributed architecture.

Consistency - This means that the data in the database remains consistent after the execution of an operation. For example after an update operation all clients see the same data.

Availability - This means that the system is always on (service guarantee availability), no downtime.

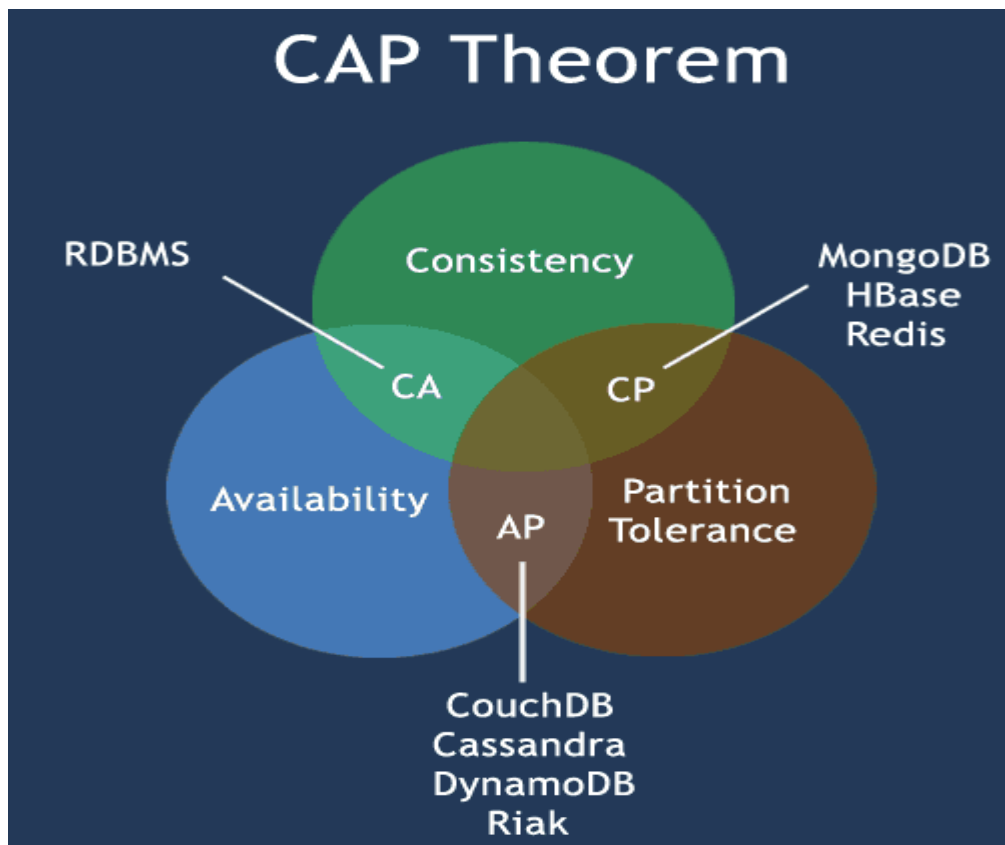
Partition Tolerance - This means that the system continues to function even the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

In theoretically it is impossible to fulfill all 3 requirements. CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements. Therefore all the current NoSQL database follow the different combinations of the C, A, P from the CAP theorem. Here is the brief description of three combinations CA, CP, AP :

CA - Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.

CP - Some data may not be accessible, but the rest is still consistent/accurate.

AP - System is still available under partitioning, but some of the data returned may be inaccurate.



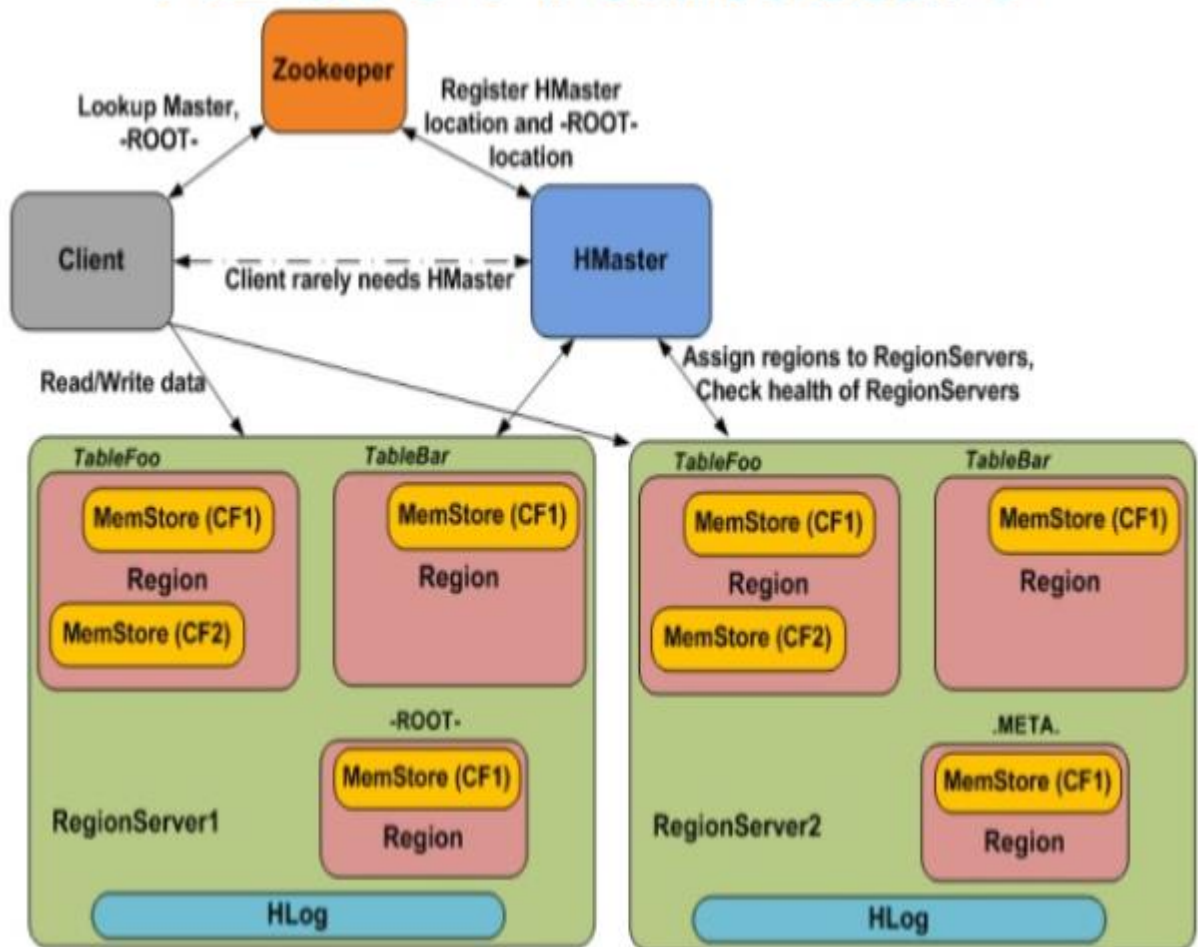
- **HBase Architecture**

HBase is a data model similar to Google's big table that is designed to provide random access to high volume of structured or unstructured data. HBase is an important component of the Hadoop ecosystem that leverages the fault tolerance feature of HDFS. HBase provides real-time read or write access to data in HDFS. HBase can be referred to as a data store instead of a database as it misses out on some important features of traditional RDBMs like typed columns, triggers, advanced query languages and secondary indexes.

HBase is an open-source, column-oriented distributed database system in Hadoop environment. Apache HBase is needed for real-time Big Data applications. The tables present in HBase consist of billions of rows having millions of columns.

Hbase is built for low latency operations, which is having some specific features compared to traditional relational models.

HBase Architecture



HBase provides low-latency random reads and writes on top of HDFS. In HBase, tables are dynamically distributed by the system whenever they become too large to handle (Auto Sharding). The simplest and foundational unit of horizontal scalability in HBase is a Region. A continuous, sorted set of rows that are stored together is referred to as a region (subset of table data). HBase architecture has a single HBase master node (HMaster) and several slaves i.e. region servers. Each region server (slave) serves a set of regions, and a region can be served only by a single region server. Whenever a client sends a write request, HMaster receives the request and forwards it to the corresponding region server.

HBase can be run in a multiple master setup, wherein there is only single active master at a time. HBase tables are partitioned into multiple regions with every region storing multiple table's rows.

Components of Apache HBase Architecture

HBase architecture has 3 important components- HMaster, Region Server and ZooKeeper.

HMaster:

HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster for load balancing. Responsibilities of HMaster –

1. Manages and Monitors the Hadoop Cluster

2. Performs Administration (Interface for creating, updating and deleting tables.)
3. Controlling the failover
4. DDL operations are handled by the HMaster
5. Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.

Region Server:

These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the Hadoop cluster. Region Server runs on HDFS DataNode and consists of the following components –

1. Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted.
2. MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore.
3. Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage.
4. HFile is the actual storage file that stores the rows as sorted key values on a disk.

Zookeeper:

HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquorum will trigger error messages and start repairing failed nodes.

ZooKeeper service keeps track of all the region servers that are there in an HBase cluster-tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts ZooKeeper to get the details of region servers. Various services that Zookeeper provides include –

1. Establishing client communication with region servers.
2. Tracking server failure and network partitions.
3. Maintain Configuration Information
4. Provides ephemeral nodes, which represent different region servers.

HBase Use Cases- When to use HBase

HBase is an ideal platform with ACID compliance properties making it a perfect choice for high-scale, real-time applications. It does not require a fixed schema, so developers have the provision to add new data as and when required without having to conform to a predefined model.

It provides users with database like access to Hadoop-scale storage, so developers can perform read or write on subset of data efficiently, without having to scan through the complete dataset. HBase is the best choice as a NoSQL database, when your application already has a Hadoop cluster running with huge amount of data. HBase helps perform fast read/writes.

- **HBase vs RDBMS**

	Hbase	RDBMS
What is?	An open source and sorted map data built on Hadoop is what we call HBase. Basically, it is column-oriented and horizontally scalable. Moreover, it offers APIs enabling development in practically any programming language. Also, it offers random real-time read/write access to data in the Hadoop File System, as it is a part of the Hadoop ecosystem that.	RDBMS refers to Relational Database Management Systems. Basically, systems like SQL, MS SQL Server, IBM DB2, Oracle, MySQL and Microsoft Access are based on RDBMS. Since it is based on the relational model introduced by E.F. Codd so it is called Relational Database Management System (RDBMS).
Database Type	HBase is the column-oriented database. On defining Column-oriented, each column is a contiguous unit of page.	Whereas, RDBMS is row-oriented that means here each row is a contiguous unit of page.
Schema-type	Schema of HBase is less restrictive; adding columns on the fly is possible.	Schema of RDBMS is more restrictive.
Sparse Tables	HBase is good with the Sparse table.	Whereas, RDBMS is not optimized for sparse tables.
Scale up/ Scale out	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.	However, 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.
Amount of data	While here it does not depend on the particular machine but the number of machines.	In RDBMS, on the configuration of the server, amount of data depends.
Support of	For HBase, there is no built-in support.	And, RDBMS has ACID support.
Data type	HBase supports both structured and nonstructural data.	RDBMS is suited for structured data.
Transaction integrity	In HBase, there is no transaction guaranty.	Whereas, RDBMS mostly guarantees transaction integrity.
JOINS	HBase supports JOINS.	RDBMS does not support JOINS.
Referential integrity	While it comes to referential integrity, there is no in-built support.	And, RDBMS, supports referential integrity.
Features of	<ul style="list-style-type: none"> • HBase is horizontally scalable. • Integrations with Map/Reduce framework. • Moreover, it is possible to refer HBase as a key-value store or column family-oriented database. 	<ul style="list-style-type: none"> • Here, in form of rows and columns, data stores. • By using SQL queries, it also supports virtual tables from where we can retrieve data. • For the purpose of data uniqueness, RDBMS provides a primary key. • Also, it offers referential integrity.

Execute blog present in below link


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

```

acadgild@localhost:~
File Edit View Search Terminal Help
[acadgild@localhost ~]$ jps
29680 HMaster
4512 NameNode
13633 QuorumPeerMain
4805 SecondaryNameNode
5015 ResourceManager
4633 DataNode
13962 Kafka
29882 Jps
29787 HRegionServer
5115 NodeManager
[acadgild@localhost ~]$ hbase shell
2018-08-19 00:31:34,250 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0>

```



The screenshot shows a terminal window titled 'acadgild@localhost:~'. The terminal contains the following text:

```
File Edit View Search Terminal Help
hbase(main):001:0>
hbase(main):001:0> create 'bulk_table','cf1','cf2'
0 row(s) in 10.9590 seconds

=> Hbase::Table - bulk_table
hbase(main):002:0>
```

[illegible]

```
[acadgild@localhost Hbase]$ hdfs dfs -put bulk_data.tsv /hbase_assignment
18/09/07 01:21:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hbase]$ hadoop dfs -ls /hbase_assignment
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/09/07 01:22:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--  1 acadgild supergroup          39 2018-09-07 01:21 /hbase_assignment/bulk_data.tsv
[acadgild@localhost Hbase]$ hadoop dfs -cat /hbase_assignment/bulk_data.tsv
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/09/07 01:22:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1      Amit      4
2      girja     3
3      jatin     5
4      swati     3
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hbase]$ █
```

File Edit View Search Terminal Help

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hbase]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -D
importtsv.columns=HBASE_ROW_KEY,cf1:Name,cf2:Exp bulk_table /hbase_assignment
2018-09-07 01:38:00,096 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
█
```

File Edit View Search Terminal Help

```
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=12618
Total vcore-seconds taken by all map tasks=12618
Total megabyte-seconds taken by all map tasks=12920832
Map-Reduce Framework
  Map input records=4
  Map output records=4
  Input split bytes=126
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=172
  CPU time spent (ms)=2280
  Physical memory (bytes) snapshot=104046592
  Virtual memory (bytes) snapshot=2067746816
  Total committed heap usage (bytes)=32571392
ImportTsv
  Bad Lines=0
File Input Format Counters
  Bytes Read=40
File Output Format Counters
  Bytes Written=0
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hbase]$ █
```

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hbase(main):002:0> scan 'bulk_table'  
ROW COLUMN+CELL  
1 column=cf1:Name, timestamp=1534619900212, value=Amit  
1 column=cf2:Exp, timestamp=1534619900212, value=4  
2 column=cf1:Name, timestamp=1534619900212, value=Girija  
2 column=cf2:Exp, timestamp=1534619900212, value=3  
3 column=cf1:Name, timestamp=1534619900212, value=Jatin  
3 column=cf2:Exp, timestamp=1534619900212, value=5  
4 column=cf1:Name, timestamp=1534619900212, value=Swati  
4 column=cf2:Exp, timestamp=1534619900212, value=3  
4 row(s) in 0.7340 seconds  
hbase(main):003:0> █
```

Task2.1

Create a flume agent that streams data from Twitter and stores in the HDFS.

Ans)I have created a developer account in twitter called “ypkApp”

And under the Keys and Access Tokens I have copied the following information:

1. Consumer Key
2. Consumer Secret Key

And then I have clicked on Create my access token, then I have copied the following information:

1. Access token
2. Access token secret

These four item information I have updated the conf file present in the flume installation conf folder and saved it as tweet.conf

In that file I have also added the Hadoop file System path where I want to store the data fetched from Twitter stream like as follows:

Hdfs://localhost:8020/Pradeep/Assignment6/Tweets

This localhost:8020 I have retrieved this information from core-site.xml file present in Hadoop installation folder.

And in tweet.conf I have also added the keywords that needs to be streamed as follows:

Cricket, Rakhi, Kerala, Uttarakhand

Attaching the conf file which I explained above:

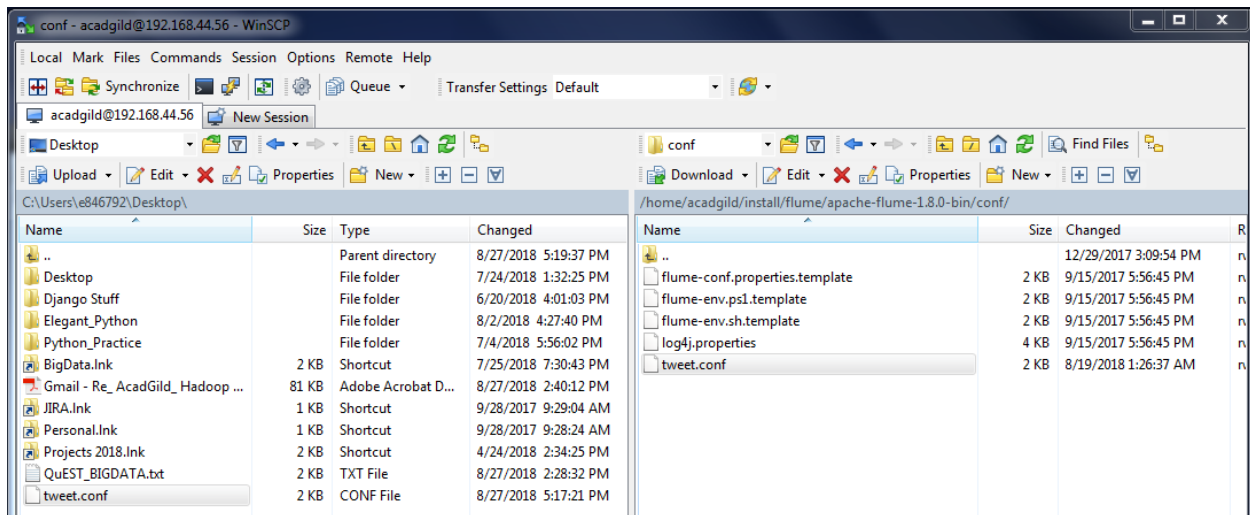


tweet.conf

Now I made sure that Hadoop demons are running as shown below:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
[acadgild@localhost ~]$ jps  
3940 NodeManager  
5767 Jps  
3466 DataNode  
3370 NameNode  
3835 ResourceManager  
3628 SecondaryNameNode  
[acadgild@localhost ~]$
```

I have copied the tweet.conf file into flume installation's conf folder using WinScp:



Then I have created the tweets folder in the following location of the Hadoop file system:

```
Hadoop fs -mkdir /Pradeep/Assignment6/tweets
```

Then I have verified whether folder is created or not with the following command:

```
Hadoop fs -ls /Pradeep/Assignment6/tweets
```

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
[acadgild@localhost ~]$ hadoop fs -ls /Pradeep/Assignment6/tweets/  
18/08/27 17:29:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
[acadgild@localhost ~]$
```

Then I have executed the following flume command to stream the twitter data:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
[acadgild@localhost ~]$ flume-ng agent -n TwitterAgent -f install/flume/apache-flume-1.8.0-bin/conf/tweet.conf
```

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
stem = false  
18/08/27 17:34:38 INFO hdfs.BucketWriter: Creating hdfs://localhost:8020/Pradeep  
/Assignment6/tweets/FlumeData.1535371477477.tmp  
18/08/27 17:34:39 INFO twitter.TwitterSource: Processed 100 docs  
18/08/27 17:34:39 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
ry for your platform... using builtin-java classes where applicable  
18/08/27 17:34:41 INFO twitter.TwitterSource: Processed 200 docs  
18/08/27 17:34:43 INFO twitter.TwitterSource: Processed 300 docs  
18/08/27 17:34:45 INFO twitter.TwitterSource: Processed 400 docs  
18/08/27 17:34:48 INFO twitter.TwitterSource: Processed 500 docs  
18/08/27 17:34:50 INFO twitter.TwitterSource: Processed 600 docs  
18/08/27 17:34:52 INFO twitter.TwitterSource: Processed 700 docs  
18/08/27 17:34:54 INFO twitter.TwitterSource: Processed 800 docs  
18/08/27 17:34:56 INFO twitter.TwitterSource: Processed 900 docs  
18/08/27 17:34:58 INFO twitter.TwitterSource: Processed 1,000 docs  
18/08/27 17:34:58 INFO twitter.TwitterSource: Total docs indexed: 1,000, total s  
kipped docs: 0  
18/08/27 17:34:58 INFO twitter.TwitterSource: 38 docs/second  
18/08/27 17:34:58 INFO twitter.TwitterSource: Run took 26 seconds and processed:  
18/08/27 17:34:58 INFO twitter.TwitterSource: 0.01 MB/sec sent to index  
18/08/27 17:34:58 INFO twitter.TwitterSource: 0.266 MB text sent to index  
18/08/27 17:34:58 INFO twitter.TwitterSource: There were 0 exceptions ignored:  
18/08/27 17:35:01 INFO twitter.TwitterSource: Processed 1,100 docs
```

Then I have stopped the streaming by pressing Ctrl+c

Now I have to check whether streamed data is copied to the said Hadoop file system directory or not:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
[acadgild@localhost ~]$ hadoop fs -ls /Pradeep/Assignment6/tweets/  
18/08/27 17:29:25 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
ry for your platform... using builtin-java classes where applicable  
[acadgild@localhost ~]$ hadoop fs -ls /Pradeep/Assignment6/tweets/  
18/08/27 17:37:03 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
ry for your platform... using builtin-java classes where applicable  
Found 1 items  
-rw-r--r-- 1 acadgild supergroup 1463327 2018-08-27 17:35 /Pradeep/Assignme  
nt6/tweets/FlumeData.1535371477477  
You have new mail in /var/spool/mail/acadgild  
[acadgild@localhost ~]$ hadoop fs -cat /Pradeep/Assignment6/tweets/FlumeData.153  
5371477477
```

Now with cat command I can check the content of the output file as shown below:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
-08-27T17:35:39ZzMinsan ok naman pero minsan Hindi Ano ba talaga hahahahahaha  
a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>  
10340493007312896000  
SMT!!and other boys' groups  
language :  
맞팔해요^^ Let's talk a lot^^F코코로 @10월은 해피ㅋㅋㅋCocoro SMT(2018-08-27T17:35:39Z~RT @S2JS248: 계획대로 되지 않아서 화난 피알오 기범 너무,,,,, https://t.co/mmeg2A1RD4a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>https://pbs.twimg.com/ext_tw_video_thumb/1033920066960601088/pu/img/ABX7w7sd6luwycil.jpg|https://twitter.com/S2JS248/status/1033920345974009856/video/1&10340493007397765120게이머兼プロデューサー兼騎空士兼ラブライバー兼社畜 気に入った絵とかをRTしたりいいねする人INA_ina_donder21(2018-08-27T17:35:39Z~RT @ogufeb: 違う、そうじゃない https://t.co/hwbGhj7huHaha href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>https://pbs.twimg.com/media/DlbodXqVAAAUvIZ.jpgzhttps://twitter.com/ogufeb/status/1033268825532162048/photo/1&10340493007396741120期待すんな所詮ばんつだたすらばんつを眩く bot。何かしら返信機能ついてるからリプ送ってみ？お？antur_bot(2018-08-27T17:35:39Zばんつ被っておおおおおおおおおおおお！！！！！！！！ばんつ被っておおおおおおおおおおお！！！！！！！！三々(\\*ω*)」*ト`コ`ッ!a href="http://twittbot.net/" rel="nofollow">twittbot.net</a>IO/701000  
You have new mail in /var/spool/mail/acadgild  
[acadgild@localhost ~]$
```

As you can see I am able to stream the data from twitter in my AcadGild VM.

I think those special symbols are something related to the separator/delimiter or something related to the file format.