Apache Flume:

- Apache Flume is a tool for data ingestion in HDFS. It collects, aggregates and transports large amount of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS.
- Flume is a highly reliable & distributed.
- The main idea behind the Flume's design is to capture streaming data from various web servers to HDFS.
- It has simple and flexible architecture based on streaming data flows.
- It is fault-tolerant and provides reliability mechanism for Fault tolerance & failure recovery.

Advantages of Apache Flume:

- Flume is scalable, reliable, fault tolerant and customizable for different sources and sinks.
- Apache Flume can store data in centralized stores (i.e data is supplied from a single store) like HBase & HDFS.
- Flume is horizontally scalable.

- If the read rate exceeds the write rate, Flume provides a steady flow of data between read and write operations.
- Flume provides reliable message delivery. The transactions in Flume are channel-based where two transactions (one sender & one receiver) are maintained for each message.
- Using Flume, we can ingest data from multiple servers into Hadoop.
- It gives us a solution which is reliable and distributed and helps us in collecting, aggregating and moving large amount of data sets like Facebook, Twitter and e-commerce websites.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.
- It supports a large set of sources and destinations types.

Streaming / Log Data?

- The data produced by various data sources like applications servers, social networking sites, cloud servers and enterprise servers usually require to be analyzed. This data will be usually in the form of log files or events.

- Log file – A file that lists all the events/actions occurring in an operating system is a log file. For instance, every request made to the server by the web server is listed in log files.

- On harvesting such log data, the information obtained is about -
  the application performance and locate various software and hardware failures.
  the user behavior and derive better business insights.

- The traditional method of transferring data into the HDFS system is to use the **put** command.

- Moving the logs produced by multiple servers to Hadoop environment is the main challenge in handling the log data.

- Hadoop File System Shell provides commands to insert data into Hadoop and read from it. The data can be inserted using put command as:

$ hadoop fs –put /path of the required file /path in HDFS where to save the file

Problem with put Command:

put command of Hadoop can be used to transfer data from these sources to HDFS. But, it has some of the drawbacks −

•Using **put** command, only one file at a time can be transferred, while the data is generated at a much higher rate by the generators. And hence the analysis made on old data would be less accurate. A solution is required to transfer the real time data.
•In **put** command, the data is needed to be packaged and should be ready for the upload. As the data is generated continuously by the webservers, packing would be a difficult task.

•In this case, a solution is required to overcome the drawbacks of put command and can transfer without delay, the "streaming data" from data generators to centralized stores (especially HDFS).

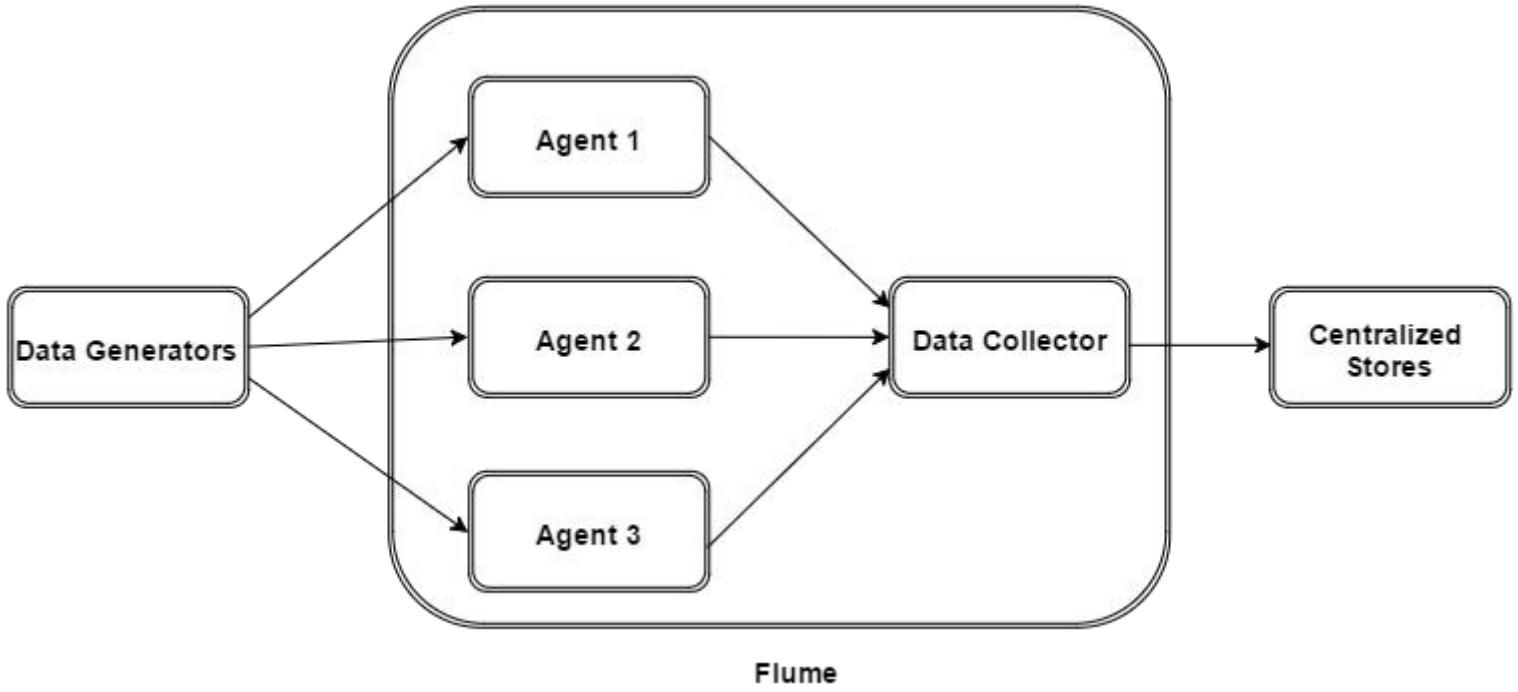
Problem with HDFS:

•In HDFS, the file exists as a directory entry and the length of the file will be considered as zero till it is closed. For instance, if a source is writing data into HDFS and the network was interrupted in the middle of the operation (without closing the file), then the data written in the file will be lost.

•To transfer the log data into HDFS, a reliable, configurable, and maintainable system is required.

Apache Flume Architecture:

- The basic architecture of Apache Flume is shown in the below illustration. As shown, the data is generated by the **data generators** like Facebook, Twitter etc. The individual Flume **agents** running on them collect the data. Then the data is collected by a **data collector**, (also an agent) from the agents and then the data is aggregated and pushed into a centralized source like HDFS or HBase.



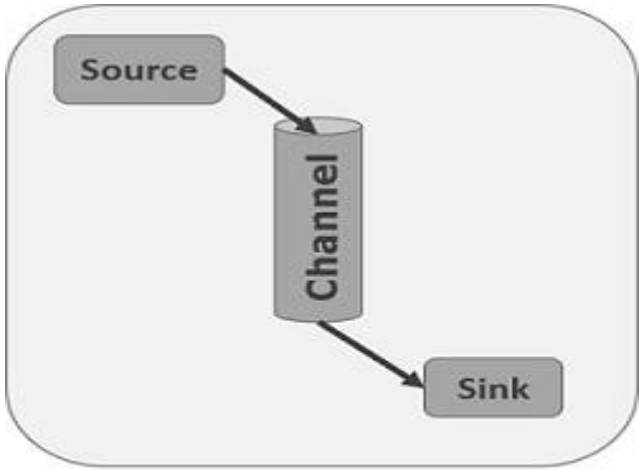
Apache Flume Event

- The basic unit of data transported inside the Apache Flume is known as an **event**. A payload of byte array has to be transported along with the optional headers from the source to the destination. The structure of a typical Apache Flume event is as follows:

Flume event

Apache Flume Agent

- An **agent** is an independent daemon process (JVM) in Apache Flume. The data (events) from the clients is received by the agent and then forwarded to other agents and forwards it to its next destination (sink or agent). Apache Flume can have more than one agent. The structure of a typical Apache Flume agent is as follows:



Flume Agent

A Flume Agent contains three main components namely, source, channel, and sink.

Source

•The component of the Apache Flume Agent which receives the data from data generators and transfers the data to one more channels as Flume events is known as a **Source**.

•Several types of sources are supported by Apache Flume and a specified data generator send events to each source

•**Example** − Avro source, Thrift source, twitter 1% source etc.

Channel

•A transient store that receives the events from the source and buffers them till they are consumed by sinks is known as a Channel. A channel is a bridge between the source and the sinks.

•These channels are fully transactional and they can work with any number of sources and sinks.

•Example − JDBC channel, File system channel, Memory channel, etc.

Sink

•A sink stores the data into centralized stores like HBase and HDFS. The data (events) from the channels are consumed by Sink and is delivered to the destination, might be another agent or the central stores.

•Example − HDFS sink

A Flume agent with one flow

Additional Components of Apache Flume Agent:

There are few other components of Apache Flume Agent that enable in transferring the events from data generators to centralized stores. They are:

Interceptors
Interceptors are used to alter/inspect flume events which are transferred between source and channel.

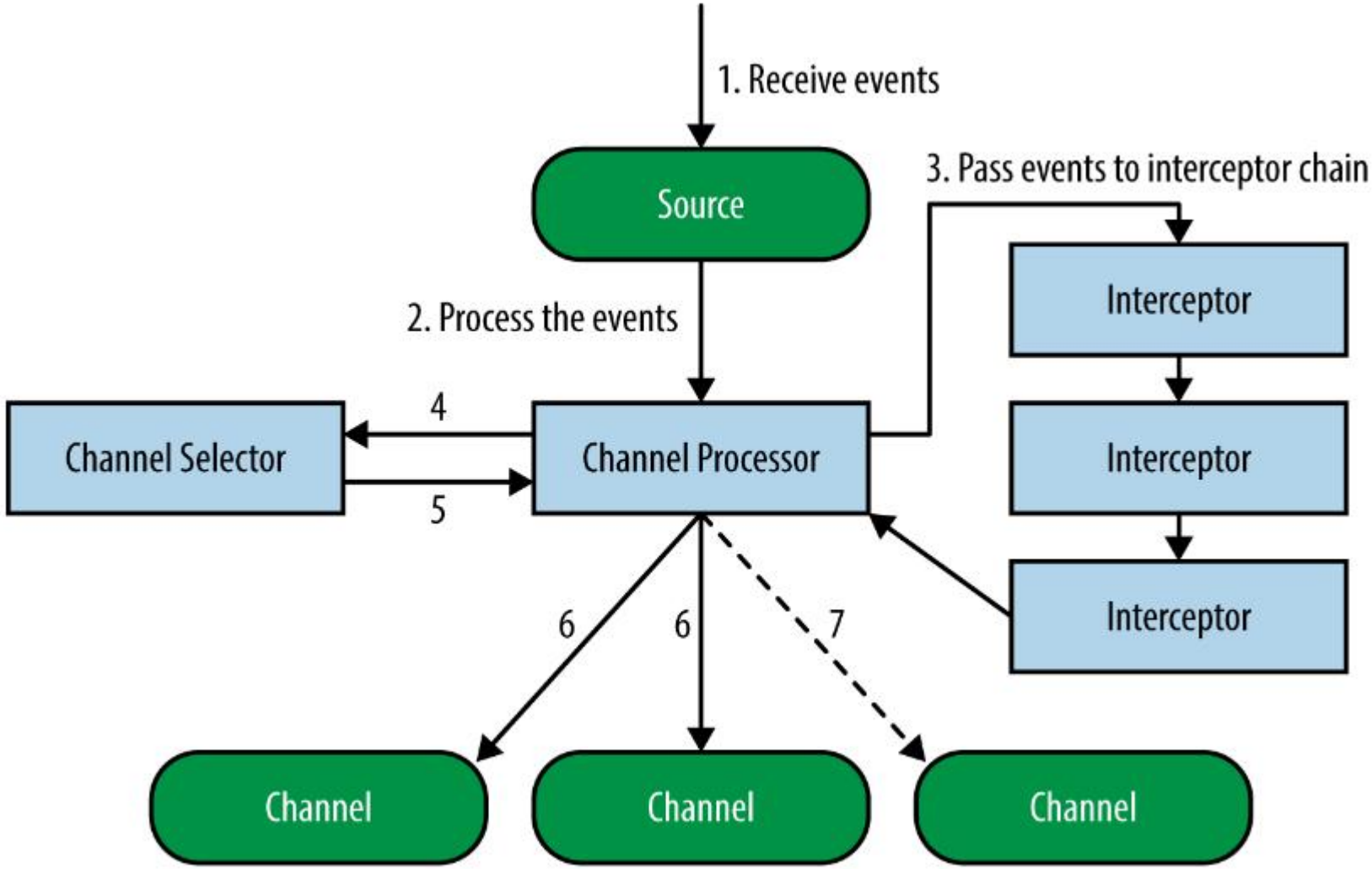
Channel Selectors
In case of multiple channels, to determine the channel to be opted to transfer the data, Channel Selector is used. Channel selectors are of two types -

Default channel selectors − These are also known as replicating channel selectors they replicates all the events in each channel.

Multiplexing channel selectors − These decides the channel to send an event based on the address in the header of that event.

Sink Processors
To raise a particular sink from the selected group of sinks a Sink Processor is used. The failover path for the sinks is created by using Sink Processors. They are also used to load balance events from a channel across multiple sinks.

- Sources writes the data into channels using channel processors, interceptors and selectors.
- Each and every source has its own channel processor, which takes the task given by the source and then passes that task or events to one or more interceptors.
- Interceptors read the event and modify or drop the event based on some criteria. We can have multiple interceptors which are called in the order in which they are defined. This can be called as chain-of-responsibility design pattern.
- Then we pass that list of events generated by interceptor chain to channel selector. The selectors decide which channels attached to this source each event be written to.
- They apply some filtering type of criteria about which channels are required and optional.
- A failure when writing a required channel causes the processor to throw a “ChannelException” error which indicates the source to retry the event. If a failure happens to optional channel, then it is ignored.
- Once all the events are written successfully then the processor indicates success to the source and sends an acknowledgement to the source that sent the event and continues to accept more events.


1. Receive events
Source
3. Pass events to interceptor chain
Interceptor
2. Process the events
4
Channel Selector
Channel Processor
Interceptor
5
Interceptor
6
6
7
Channel
Channel
Channel

Example of Apache Flume:

- We are using single source-channel-sink. We configure the flume agent using java properties file. The configuration controls the types of sources, sinks, and channels that are used, as well as how they are connected together.

- First we need to list the sources, sinks and channels for the given agent which we are using, and then point the source and sink to a channel.

- A source instance can specify multiple channels, but a sink instance can only specify one channel.

<Agent>.sources = <Source>

<Agent>.sinks = <Sink>

<Agent>.channels = <Channel1> <Channel2>

 # To point the source and sink to the channel

<Agent>.sources.<Source>.channels = <Channel1> <Channel2> ...

<Agent>.sinks.<Sink>.channel = <Channel1>

Then we need to set the properties of each source, sink and channel.

 # properties for sources

<Agent>.sources.<Source>.<someProperty> = <someValue>

 # properties for channels

<Agent>.channel.<Channel>.<someProperty> = <someValue>

 # properties for sinks

<Agent>.sinks.<Sink>.<someProperty> = <someValue>

- Each component i.e. Source, Channel, Sink has its own set of properties. We need to set the property “type” for every component in Flume.

Example:

First we need to write the java properties file as,

mamoon@mamoon-VirtualBox:~/apache-flume-1.8.0-cdh5.3.2-bin$ cd conf/

mamoon@mamoon-VirtualBox :~/apache-flume-1.8.0-cdh5.3.2-bin/conf$ cat flume.conf

agent1.sources = source1

agent1.channels = Channel1

agent1.sinks = Sink1

```
agent1.sources.source1.type = exec

agent1.sources.source1.command = cat /home/myflume

agent1.sources.source1.channels = Channel1


agent1.sinks.Sink1.type = hdfs

agent1.sinks.Sink1.channel = Channel1

agent1.sinks.Sink1.hdfs.path = hdfs://localhost:9000/flume-00001

agent1.sinks.Sink1.filetype = DataStream


agent1.channels.Channel1.type = memory
```

- Here “agent1” is the name of the agent and we are using ‘exec’ source. The sink is HDFS sink which means we are writing the data into HDFS.

- DataStream means it will not write any metadata, only actual data will be collected.

- Now execute the flume configuration file as,

mamoon@mamoon-VirtualBox :~/apache-flume-1.8.0-cdh5.3.2-bin$ bin/flume-ng agent --conf ./conf/ -f conf/flume.conf -n agent1 -Dflume.root.logger=DEBUG,console

Flume-ng—flume executable file

-conf—location of configuration directory

-f—location of configuration file

-n—name of the agent

We are using -Dflume.root.logger=DEBUG,console so that if any problem occurs it will be written on console.

Now open other terminal and check for hdfs://localhost:9000/flume-00001.

mamoon@mamoon-VirtualBox :~$ hdfs dfs -ls /flume-00001

Found 1 items

-rw-r--r-- 1 mamoon supergroup 200 2019-12-13 05:05 /flume-00001/FlumeData.1513170317306

Now we have got the data into HDFS which was mentioned by source using “cat /home/myflume”.

Example: flume using “spooling directory” source.

First create flume configuration file,

mamoon@mamoon-VirtualBox :~/apache-flume-1.8.0-cdh5.3.2-bin/conf$ cat Flume1.conf

agent1.sinks = hdfs-sink1

agent1.sources = source1

agent1.channels = fileChannel1

agent1.channels.fileChannel1.type = file

agent1.channels.fileChannel.capacity = 2000

agent1.channels.fileChannel.transactionCapacity = 100

agent1.sources.source1.type = spooldir

#Spooldir in my case is /home/hadoop/Desktop/flume_sink

agent1.sources.source1.spoolDir = /home/mamoon/myflume

agent1.sources.source1.fileHeader = false

agent1.sources.source1.fileSuffix = .COMPLETED

agent1.sinks.hdfs-sink1.type = hdfs

agent1.sinks.hdfs-sink1.hdfs.path = hdfs://localhost.localdomain:9000/flume_sink

agent1.sinks.hdfs-sink1_1.hdfs.batchSize = 1000

agent1.sinks.hdfs-sink1.hdfs.rollSize = 2684

agent1.sinks.hdfs-sink1.hdfs.rollInterval = 0

agent1.sinks.hdfs-sink1.hdfs.rollCount = 5000

agent1.sinks.hdfs-sink1.hdfs.writeFormat=Text

agent1.sinks.hdfs-sink1.hdfs.fileType = DataStream

agent1.sources.source1.channels = fileChannel1

agent1.sinks.hdfs-sink1.channel = fileChannel1
Now start the agent as,

mamoon@mamoon-VirtualBox :~/apache-flume-1.8.0-cdh5.3.2-bin$ flume-ng agent -n agent1 -f /home/hdadmin/apache-flume-1.5.0-cdh5.3.2-bin/conf/Flume1.conf

Now copy some files in spoolDir, they will be automatically being stored in HDFS.

mamoon@mamoon-VirtualBox :~$ cat data2.txt

(1,2)

(5,3)

mamoon@mamoon-VirtualBox :~$ cp data2.txt Desktop/flume_sink

mamoon@mamoon-VirtualBox :~$ cd Desktop/flume_sink/

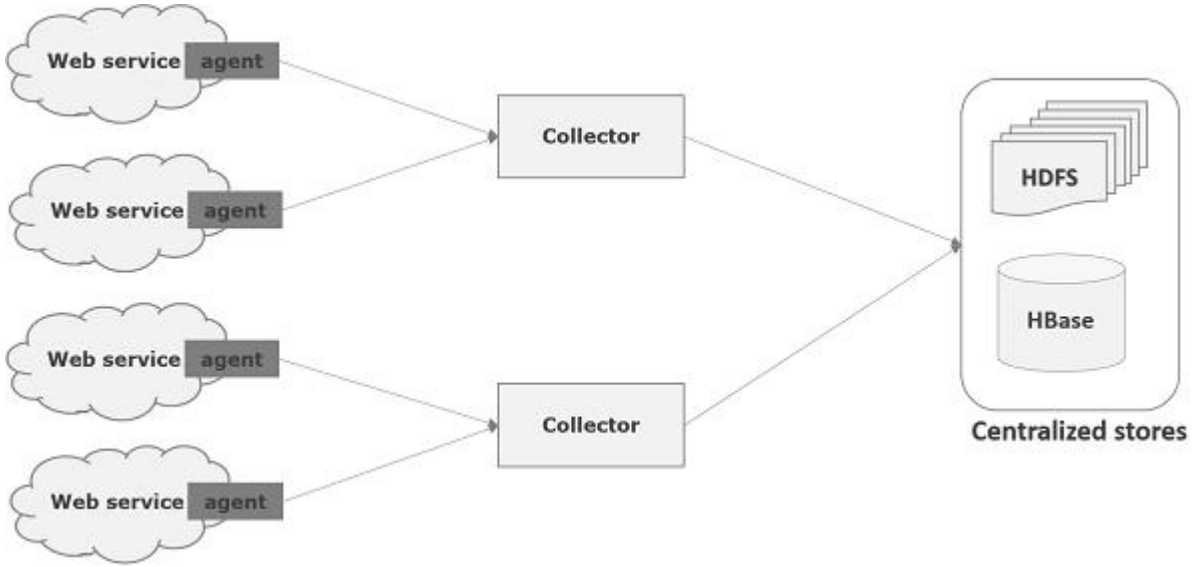
mamoon@mamoon-VirtualBox :~/Desktop/flume_sink$ ls

data2.txt.COMPLETED
We have copied ‘data2.txt’ into our spool directory and after that its status have been changed into ‘completed’ state. Now check it in HDFS.

mamoon@mamoon-VirtualBox :~/Desktop/flume_sink$ hdfs dfs -ls /flume_sink

Data Flow in Apache Flume:

- The framework used for moving log data into HDFS is Apache Flume.
- The log servers generate events and log data and also have agents running on them, which receive the data.
- An immediate node, Collector, collects the data from the agents. Finally the data is aggregated and pushed into a centralized store such as HBase or HDFS.
- The data flow of Apache Flume is depicted as:



Multi-hop Flow

•There can be multiple agents within a Apache Flume, and an event before reaching the final destination may travel through more than one agent. This is known as multi-hop flow.

Fan-out Flow

•The dataflow from one source to multiple channels is known as fan-out flow. It is of two types −

Replicating − The data flow where the data will be replicated in all the configured channels.
Multiplexing − The data flow where the data will be sent to a selected channel which is mentioned in the header of the event.

Fan-in Flow

•The data flow in which the data will be transferred from many sources to one channel is known as fan-in flow.

Failure Handling in Flume:

For each event in Apache Flume, two transactions take place – one at sender side and the other at receiver side. The events are send by the sender to the receiver. On receiving the data, the receiver commits the transaction and a “received” signal is sent to the sender. On receiving the signal, the sender commits its transaction. The sender commits the transaction only on receiving the signal from receiver.

Flume configuration and Twitter Sentiment Analysis using Apache Flume

1.Download Flume and extract it with following command:

```
$ tar -xvzf apache-flume-1.8.0-bin.tar.gz
```

2**. Download the flume-sources-1.0-SNAPSHOT.jar from following link:**

https://drive.google.com/file/d/0B_t6uqPmWadsdWJNQ0NjaXBUYUk/view?usp=sharing

3.Paste flume-sources-1.0-SNAPSHOT.jar in flume lib folder

```
cp flume-sources-1.0-SNAPSHOT.jar $FLUME_HOME/lib
```

4.Add it to the flume class path as shown below in the conf/flume-env.sh file . Add Java home path too.

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
FLUME_CLASSPATH="/home/mamoon/apache-flume-1.8.0-bin/lib/flume-sources-1.0-SNAPSHOT.jar"
```

Note: The jar contains the java classes to pull the Tweets and save them into HDFS.

5. Download consumerKey, consumerSecret, accessTokenSecret from https://apps.twitter.com/ which can be accessed from your twitter developer account by creating a simple app. See here how to create a twitter app:

https://www.youtube.com/watch?v=xqSp7060Gj0

6. Create flume-twitter.conf file in conf folder and paste given lines:

Note: Here change consumerKey, ConsumerSecret, accessToken, accessTokenSecret

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS
TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sources.Twitter.consumerKey = xxxxxxxxxxxxxxxxxxxxxxxx
TwitterAgent.sources.Twitter.consumerSecret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
TwitterAgent.sources.Twitter.accessToken = xxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
TwitterAgent.sources.Twitter.accessTokenSecret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

This is where you give keywords to be fetched from twitter. Replace spark, flink with your desired keyword.

```
TwitterAgent.sources.Twitter.keywords = covid-19, Maharastra
TwitterAgent.sinks.HDFS.channel = MemChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path =  /user/flume/tweets
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000
TwitterAgent.channels.MemChannel.type = memory
 TwitterAgent.channels.MemChannel.capacity = 10000
TwitterAgent.channels.MemChannel.transactionCapacity = 10000
```

The consumerKey, consumerSecret, accessToken and accessTokenSecret have to be replaced with those obtained from https://dev.twitter.com/apps. And, TwitterAgent.sinks.HDFS.hdfs.path should point to the NameNode and the location in HDFS where the tweets will go to.

The TwitterAgent.sources.Twitter.keywords value can be modified to get the tweets for some other topic like football, movies etc.

7. **Add following to .bashrc file**

$ gedit ~/.bashrc

FLUME VARIABLES START

export FLUME_HOME=/home/mamoon/apache-flume-1.8.0-bin
export FLUME_CONF_DIR=$FLUME_HOME/conf
export FLUME_CLASSPATH=$FLUME_CONF_DIR
export PATH="$FLUME_HOME/bin:$PATH"

FLUME VARIABLES END

8. Run following command to reload new .bashrc file

$source ~/.bashrc

9. Rename these 3 files in lib folder of Flume. (All you need to do just change the extention of these files from .jar to .org)

twitter4j-core-3.0.3.jar twitter4j-media-support-3.0.3.jar twitter4j-stream-3.0.3.jar
to
twitter4j-core-3.0.3.org twitter4j-media-support-3.0.3.org twitter4j-stream-3.0.3.org

10. Make Directories in HDFS using following Command:

$ hdfs dfs –mkdir /user/flume/tweets

11. Run Flume and collect data into HDFS.

Start Hadoop first. $start-all.sh

Start flume using the below command

(Assuming you are inside '/home/mamoon/apache-flume-1.8.0-bin')

bin/flume-ng agent -n TwitterAgent --conf ./conf/-f --conf-file conf/flume-twitter.conf -Dflume.root.logger=DEBUG,console

After a couple of minutes the Tweets should appear in HDFS.

Visit localhost:50070 in your browser

Navigate to localhost:50070/explorer.html#/user/flume/tweets/

12. To access raw data retrieved from Twitter, go to GUI of Hadoop Administrator:

http://localhost:50070/explorer.html#/

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Overview 'localhost:9000' (active)

Started:	Sun Mar 25 11:27:34 +0530 2018
Version:	2.9.0, r756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled:	Tue Nov 14 04:45:00 +0530 2017 by arsuresh from branch-2.9.0
Cluster ID:	CID-d75f57a6-30e8-4bd4-98c7-4c6f11441602
Block Pool ID:	BP-635614031-172.26.81.223-1515997263502

Summary

Security is off.

Safe mode is ON. The reported blocks 0 needs additional 548 blocks to reach the threshold 0.9990 of total blocks 549. The number of live datanodes 0 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.