Session 12 – Oozie and Flume Assignment1
Streaming Twitter Data Using Flume

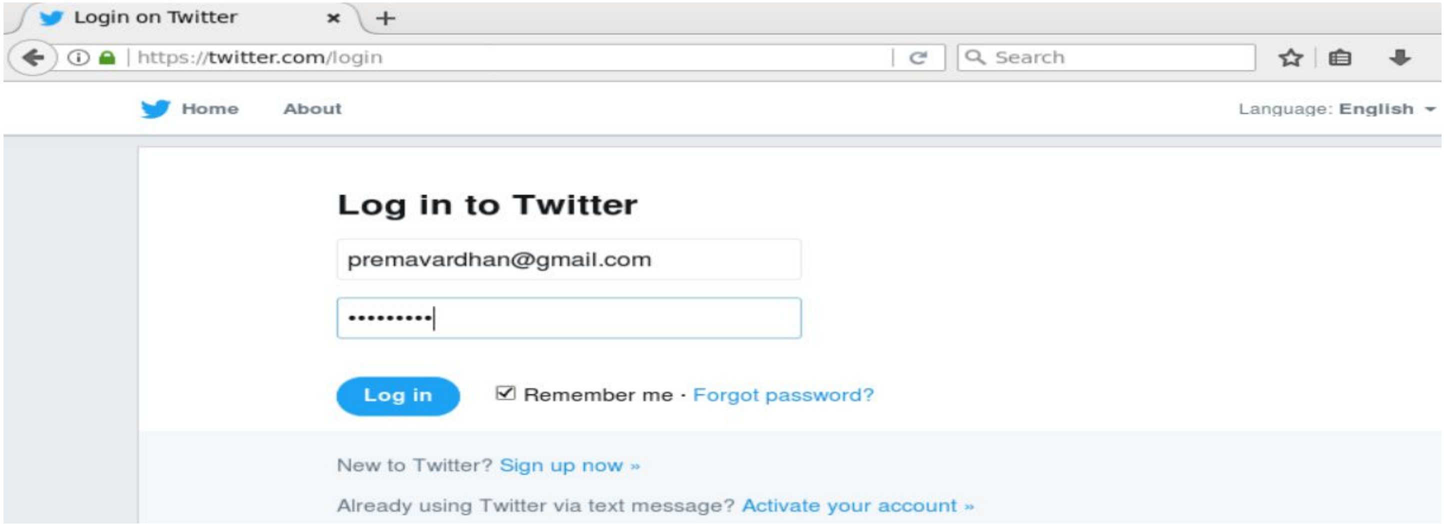
Streaming Twitter Data

To stream data to our database from twitter we should have the following pre-requisites.

- Twitter account.
- Hadoop cluster.

If both prerequisites are available we can move to our further step.

Step 1:
Loging to the twitter account



Log in to Twitter

premavardhan@gmail.com

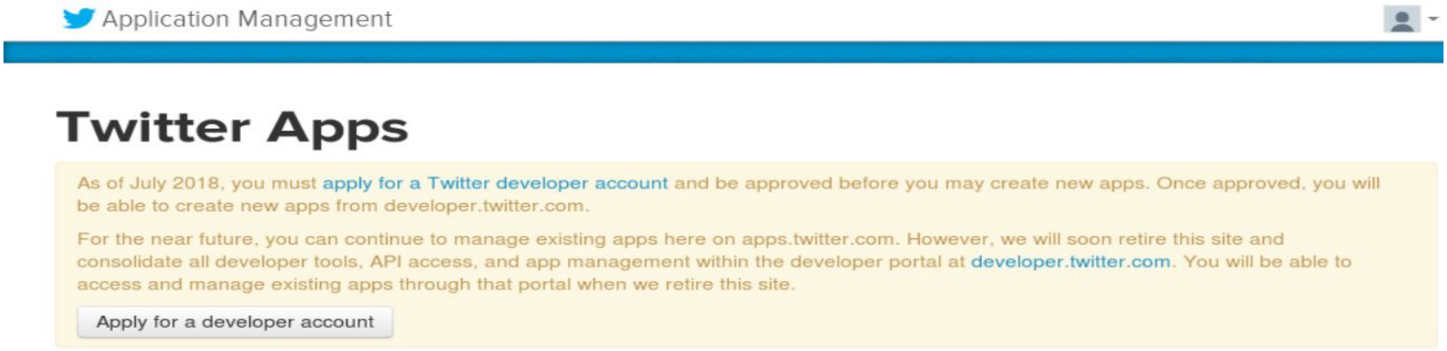
••••••••••

New to Twitter? Sign up now »

Already using Twitter via text message? Activate your account »

Step 2:
Go to the following link and click the 'create new app' button.
https://apps.twitter.com/app

You get following picture.

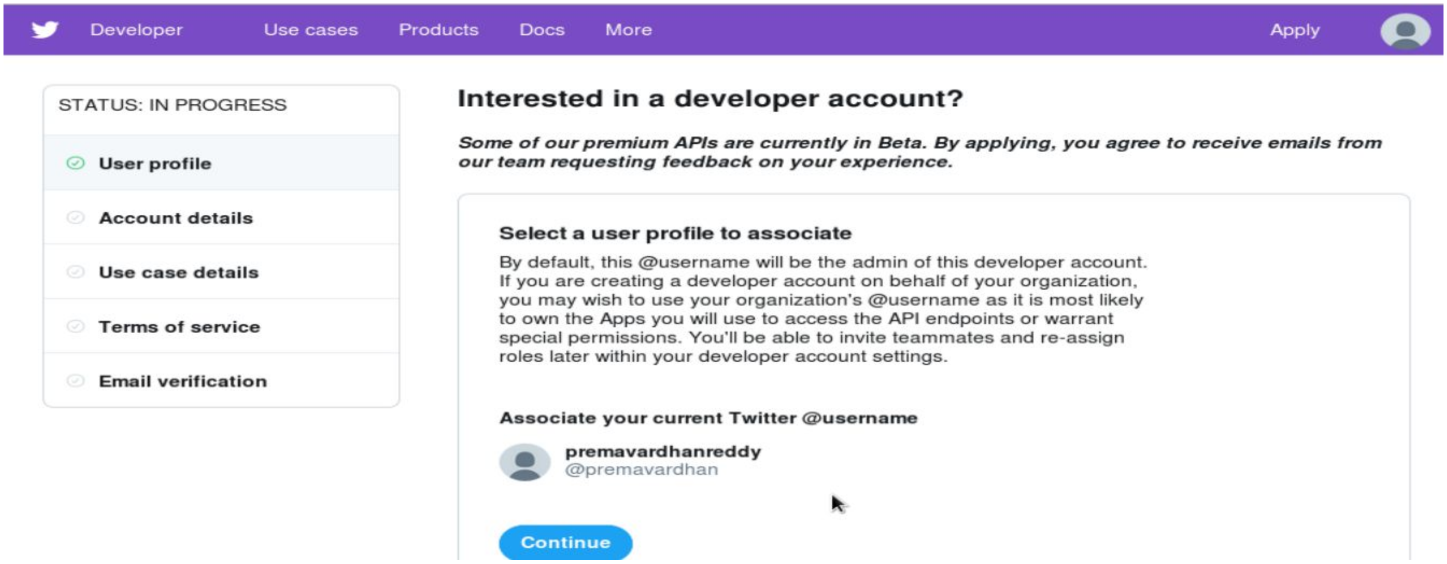


Twitter Apps

As of July 2018, you must apply for a Twitter developer account and be approved before you may create new apps. Once approved, you will be able to create new apps from developer.twitter.com.

For the near future, you can continue to manage existing apps here on apps.twitter.com. However, we will soon retire this site and consolidate all developer tools, API access, and app management within the developer portal at developer.twitter.com. You will be able to access and manage existing apps through that portal when we retire this site.

Click on the 'Apply for a development account.



Interested in a developer account?

Some of our premium APIs are currently in Beta. By applying, you agree to receive emails from our team requesting feedback on your experience.

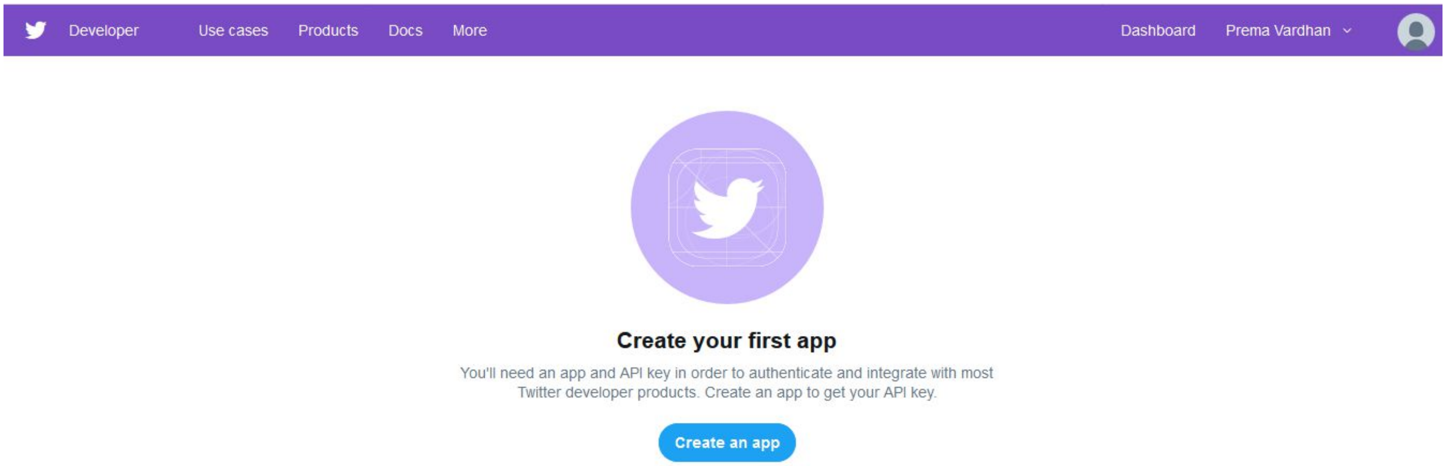
Select a user profile to associate

By default, this @username will be the admin of this developer account. If you are creating a developer account on behalf of your organization, you may wish to use your organization's @username as it is most likely to own the Apps you will use to access the API endpoints or warrant special permissions. You'll be able to invite teammates and re-assign roles later within your developer account settings.

Associate your current Twitter @username

premavardhanreddy
@premavardhan

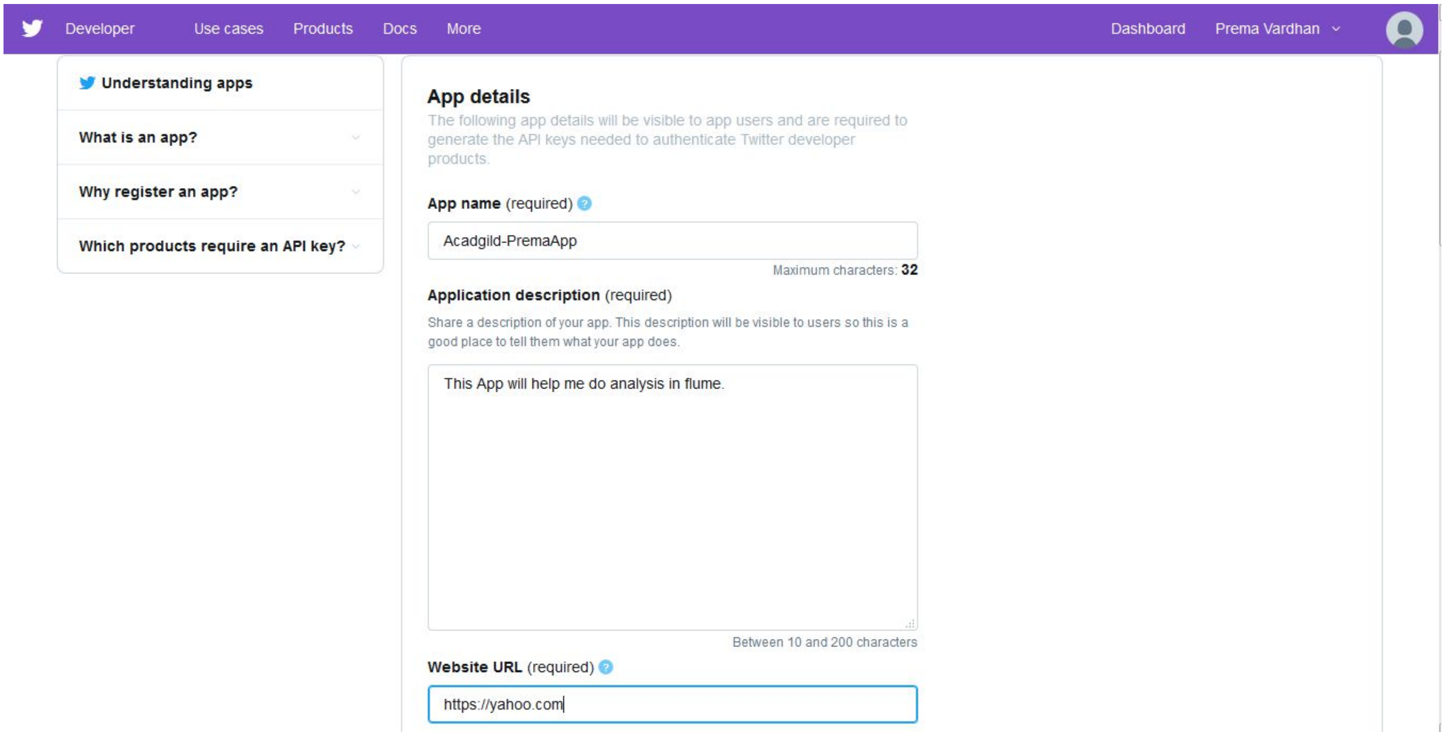
Now click on continue button.



Create your first app

You'll need an app and API key in order to authenticate and integrate with most Twitter developer products. Create an app to get your API key.

Step 3:
Enter the necessary details.



Understanding apps

What is an app?

Why register an app?

Which products require an API key?

App details

The following app details will be visible to app users and are required to generate the API keys needed to authenticate Twitter developer products.

App name (required)

Acadgild-PremaApp

Maximum characters: **32**

Application description (required)

Share a description of your app. This description will be visible to users so this is a good place to tell them what your app does.

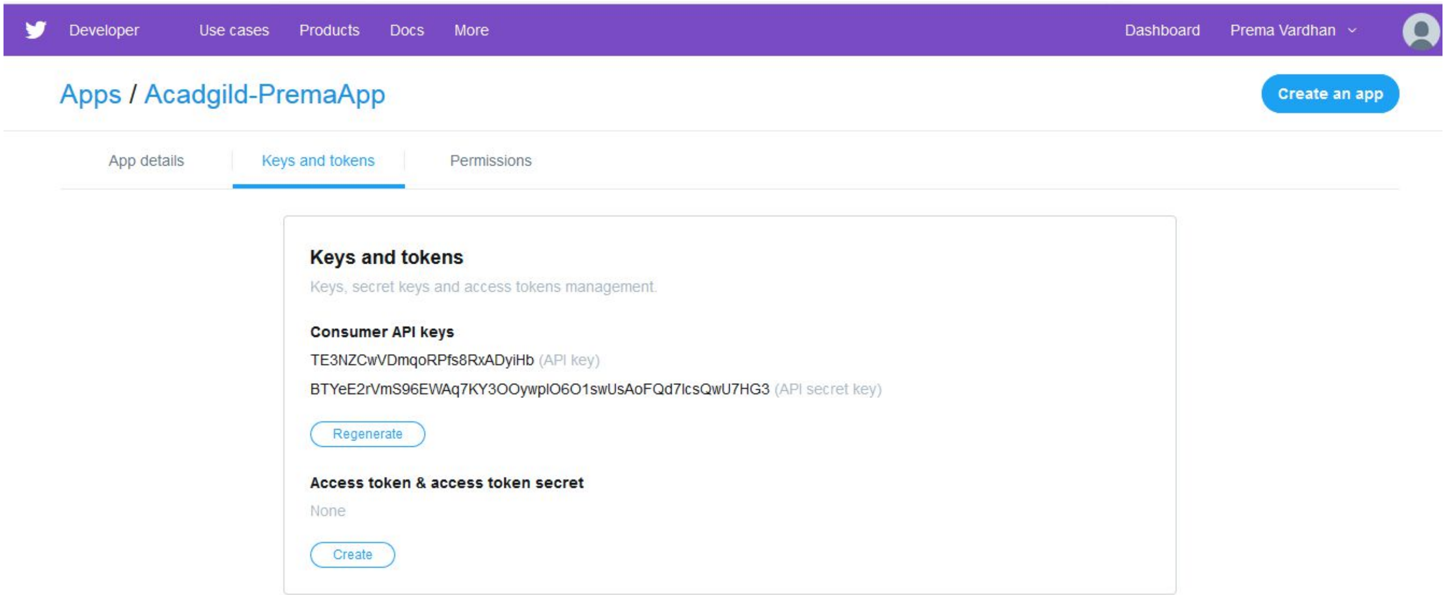
This App will help me do analysis in flume.

Between 10 and 200 characters

Website URL (required)

https://yahoo.com

Step 4:
Select the ‘keys and Access Token’ tab.



Apps / Acadgild-PremaApp

Keys and tokens

Keys, secret keys and access tokens management.

Consumer API keys

TE3NZCwVDmqoRPfs8RxADyiHb (API key)

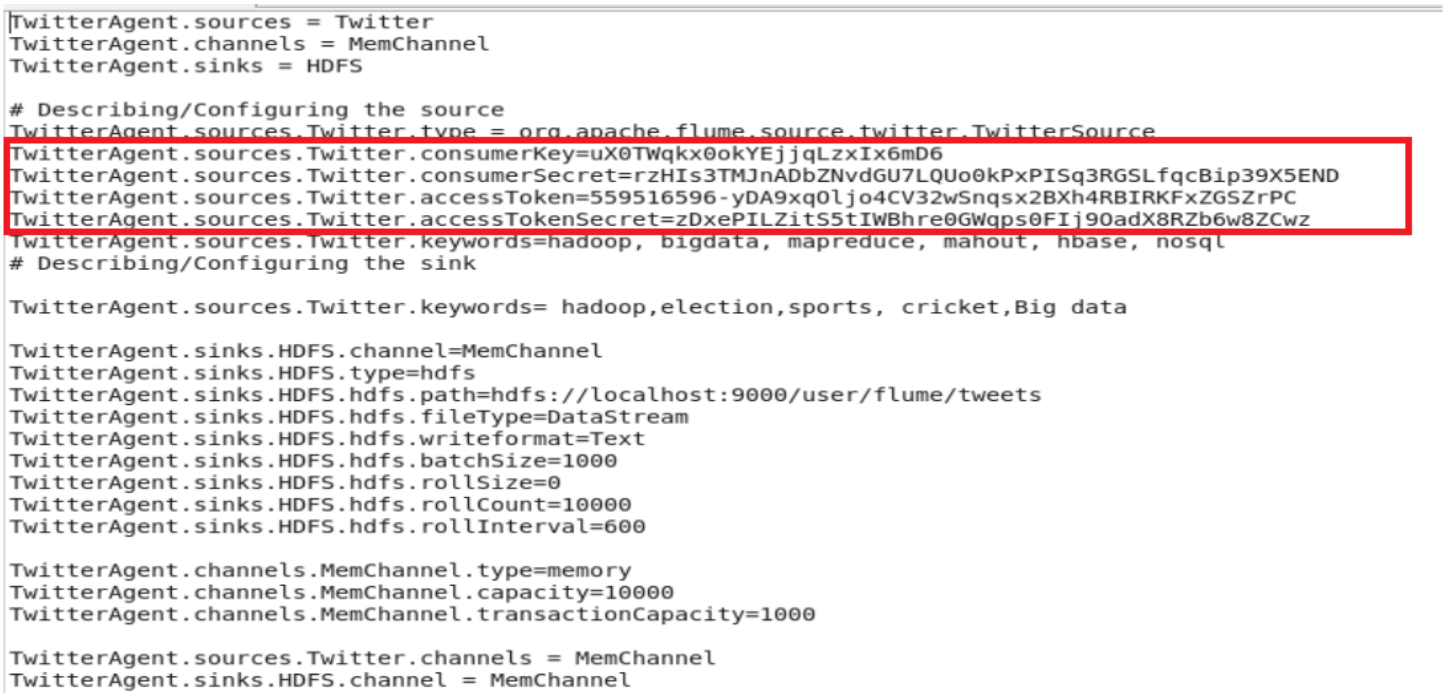
BTYeE2rVmS96EWAq7KY3OOywplO6O1swUsAoFQd7lcsQwU7HG3 (API secret key)

Access token & access token secret

None

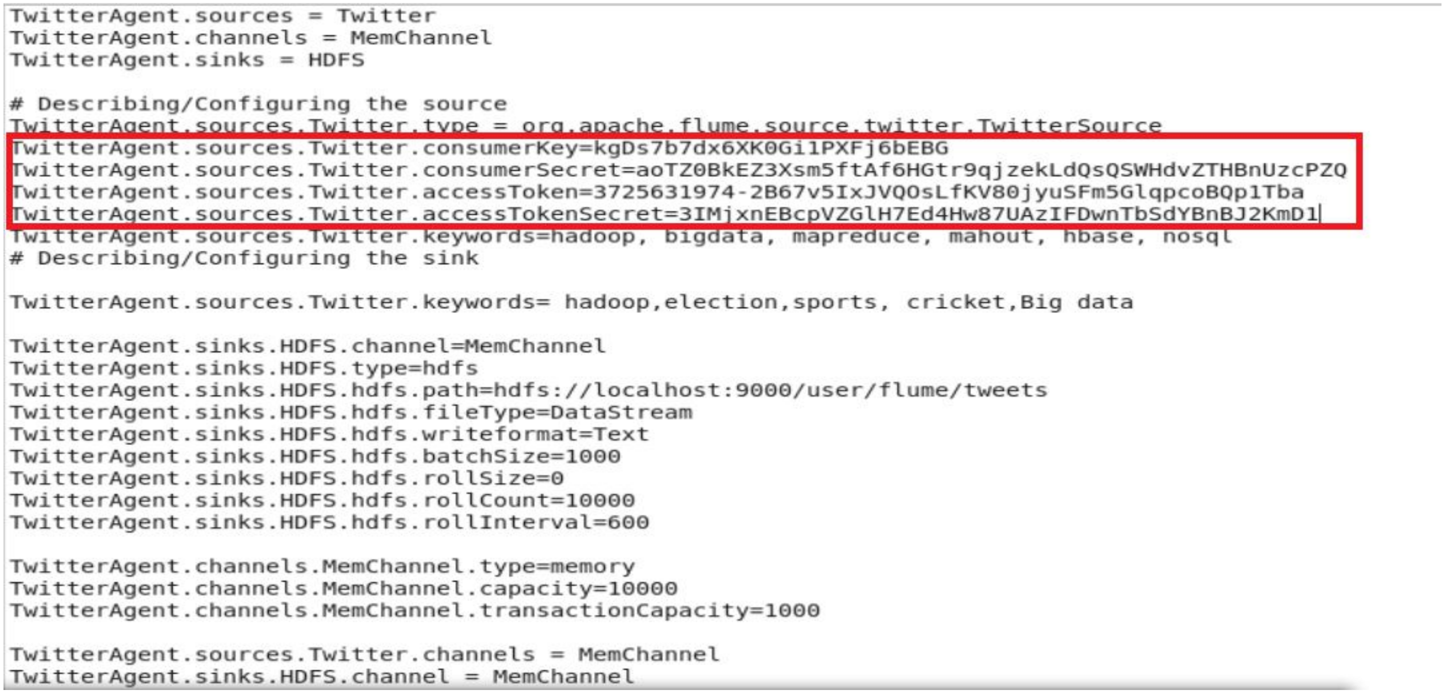
Step 5:
Copy the consumer key and the consumer secret key:

Step 6:
Copy the Flume configuration code from the below link and pass it in the newly created file:
https://drive.google.com/open?id=0B1QaXx7tpw3Sb3U4LW9SWINidkk



```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

# Describing/Configuring the source
TwitterAgent.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource
TwitterAgent.sources.Twitter.consumerKey=uX0TWqkx0okYEjjqLzxIx6mD6
TwitterAgent.sources.Twitter.consumerSecret=rzHIs3TMJnADbZNvdGU7LQUo0kPxPISq3RGSLfqcBip39X5END
TwitterAgent.sources.Twitter.accessToken=559516596-yDA9xqOljo4CV32wSnqsx2BXh4RBIRKFxZGSZrPC
TwitterAgent.sources.Twitter.accessTokenSecret=zDxePILZitS5tIWBhre0GWqps0FIj9OadX8RZb6w8ZCwz
TwitterAgent.sources.Twitter.keywords=hadoop, bigdata, mapreduce, mahout, hbase, nosql
# Describing/Configuring the sink

TwitterAgent.sources.Twitter.keywords= hadoop,election,sports, cricket,Big data

TwitterAgent.sinks.HDFS.channel=MemChannel
TwitterAgent.sinks.HDFS.type=hdfs
TwitterAgent.sinks.HDFS.hdfs.path=hdfs://localhost:9000/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.sinks.HDFS.hdfs.rollInterval=600

TwitterAgent.channels.MemChannel.type=memory
TwitterAgent.channels.MemChannel.capacity=10000
TwitterAgent.channels.MemChannel.transactionCapacity=1000

TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sinks.HDFS.channel = MemChannel
```

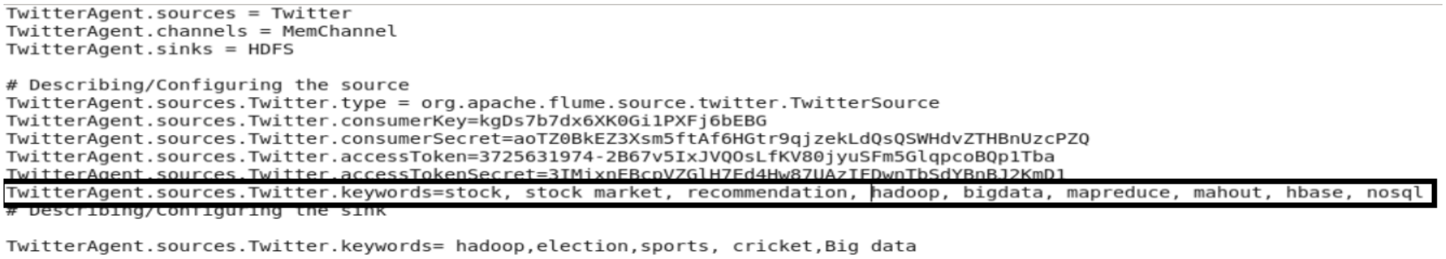
Step 7:
Change the twitter api keys with the keys generated as we were created the twitter app.



```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

# Describing/Configuring the source
TwitterAgent.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource
TwitterAgent.sources.Twitter.consumerKey=kgDs7b7dx6XK0Gi1PXFj6bEBG
TwitterAgent.sources.Twitter.consumerSecret=aoTZ0BkEZ3Xsm5ftAf6HGtr9qjzekLdQsQSWHdvZTHBnUzcPZQ
TwitterAgent.sources.Twitter.accessToken=3725631974-2B67v5IxJVQOsLfKV80jyuSFm5GlqpcoBQp1Tba
TwitterAgent.sources.Twitter.accessTokenSecret=3IMjxnEBcpVZGlH7Ed4Hw87UAzIFDwnTbSdYBnBJ2KmD1
TwitterAgent.sources.Twitter.keywords=hadoop, bigdata, mapreduce, mahout, hbase, nosql
# Describing/Configuring the sink

TwitterAgent.sources.Twitter.keywords= hadoop,election,sports, cricket,Big data

TwitterAgent.sinks.HDFS.channel=MemChannel
TwitterAgent.sinks.HDFS.type=hdfs
TwitterAgent.sinks.HDFS.hdfs.path=hdfs://localhost:9000/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.sinks.HDFS.hdfs.rollInterval=600

TwitterAgent.channels.MemChannel.type=memory
TwitterAgent.channels.MemChannel.capacity=10000
TwitterAgent.channels.MemChannel.transactionCapacity=1000

TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sinks.HDFS.channel = MemChannel
```

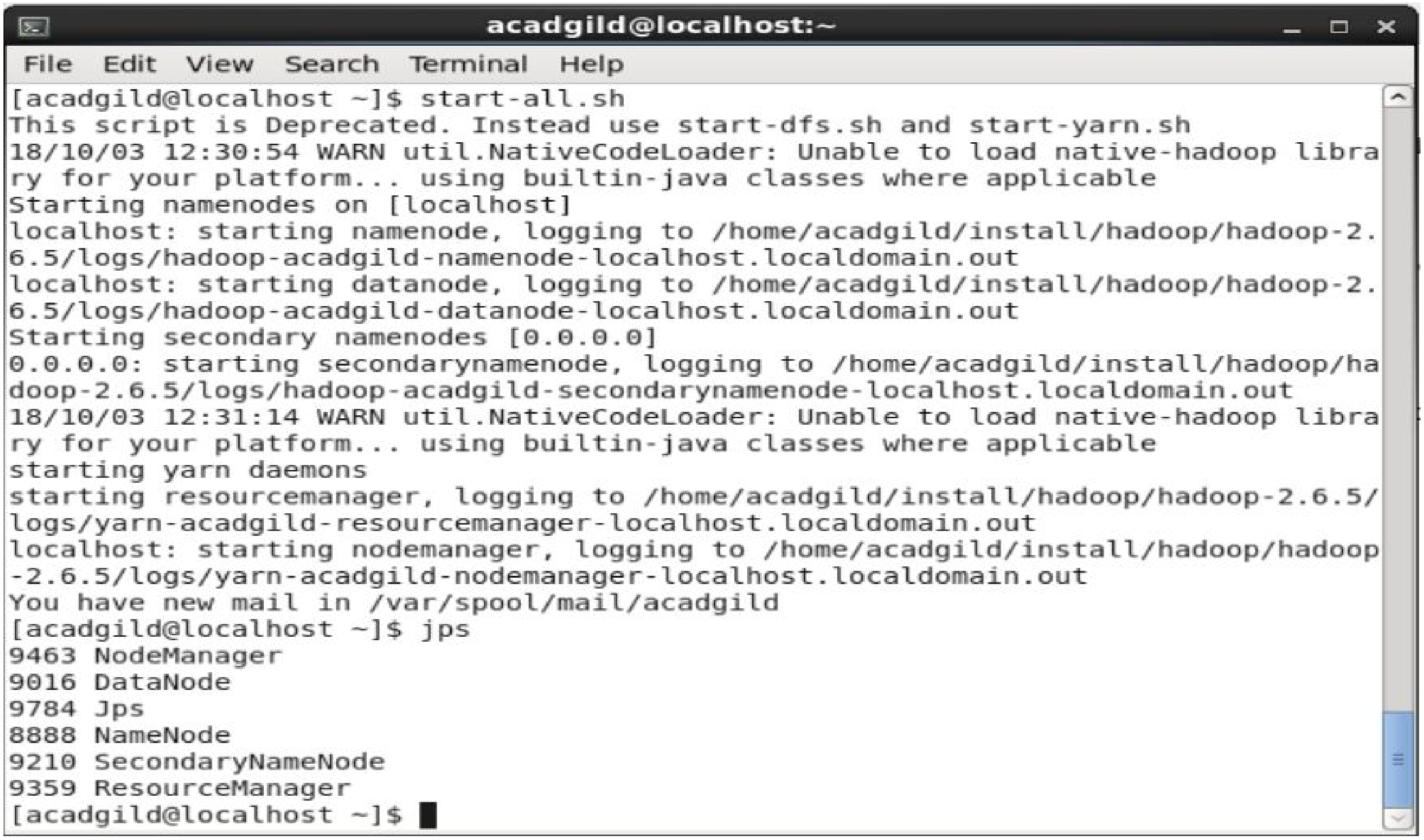
Step 8:
We have to decide which keywords tweet data to be collected from the twitter application. So, you can change the keywords in the TwitterAgent.source.Twitter.keywords command.

In our example, we are fetching tweet data related to stock market, recommendation, Hadoop, election, sports cricket and Big data.



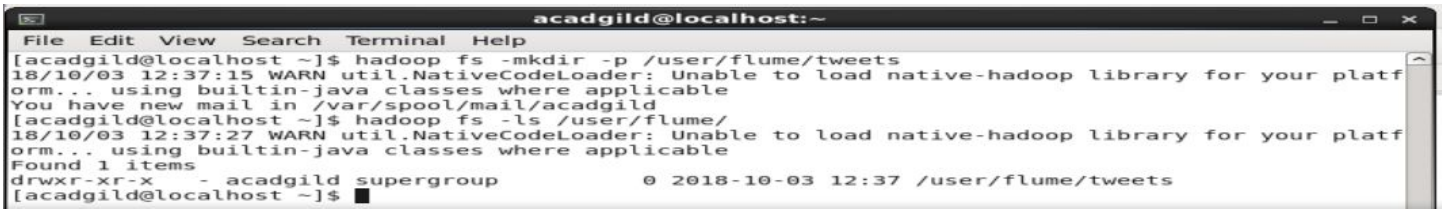
```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

# Describing/Configuring the source
TwitterAgent.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource
TwitterAgent.sources.Twitter.consumerKey=kgDs7b7dx6XK0Gi1PXFj6bEBG
TwitterAgent.sources.Twitter.consumerSecret=aoTZ0BkEZ3Xsm5ftAf6HGtr9qjzekLdQsQSWHdvZTHBnUzcPZQ
TwitterAgent.sources.Twitter.accessToken=3725631974-2B67v5IxJVQOsLfKV80jyuSFm5GlqpcoBQp1Tba
TwitterAgent.sources.Twitter.accessTokenSecret=3IMixnFBcpVZGlH7Ed4Hw87UAzIFDwnTbSdYBnBJ2KmD1
TwitterAgent.sources.Twitter.keywords=stock, stock market, recommendation, hadoop, bigdata, mapreduce, mahout, hbase, nosql
# Describing/Configuring the sink

TwitterAgent.sources.Twitter.keywords= hadoop,election,sports, cricket,Big data
```

Step 9:
Open a new terminal and start all the Hadoop daemons, before running the flume command to fetch the twitter data. Use the 'jps' command to see the running Hadoop daemons.



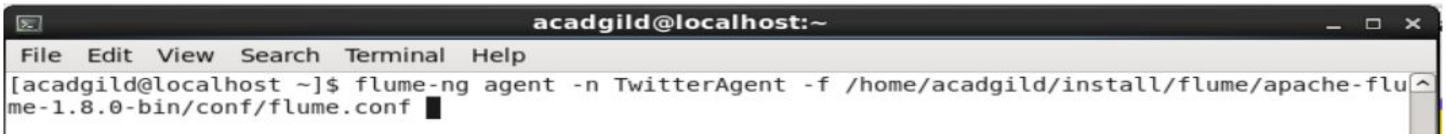
```
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/10/03 12:30:54 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.
6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.
6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/ha
doop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
18/10/03 12:31:14 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/
logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop
-2.6.5/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ jps
9463 NodeManager
9016 DataNode
9784 Jps
8888 NameNode
9210 SecondaryNameNode
9359 ResourceManager
[acadgild@localhost ~]$ █
```

Step 10:
Now create a new directory inside HDFS path, where the Twitter tweet data should be stored.



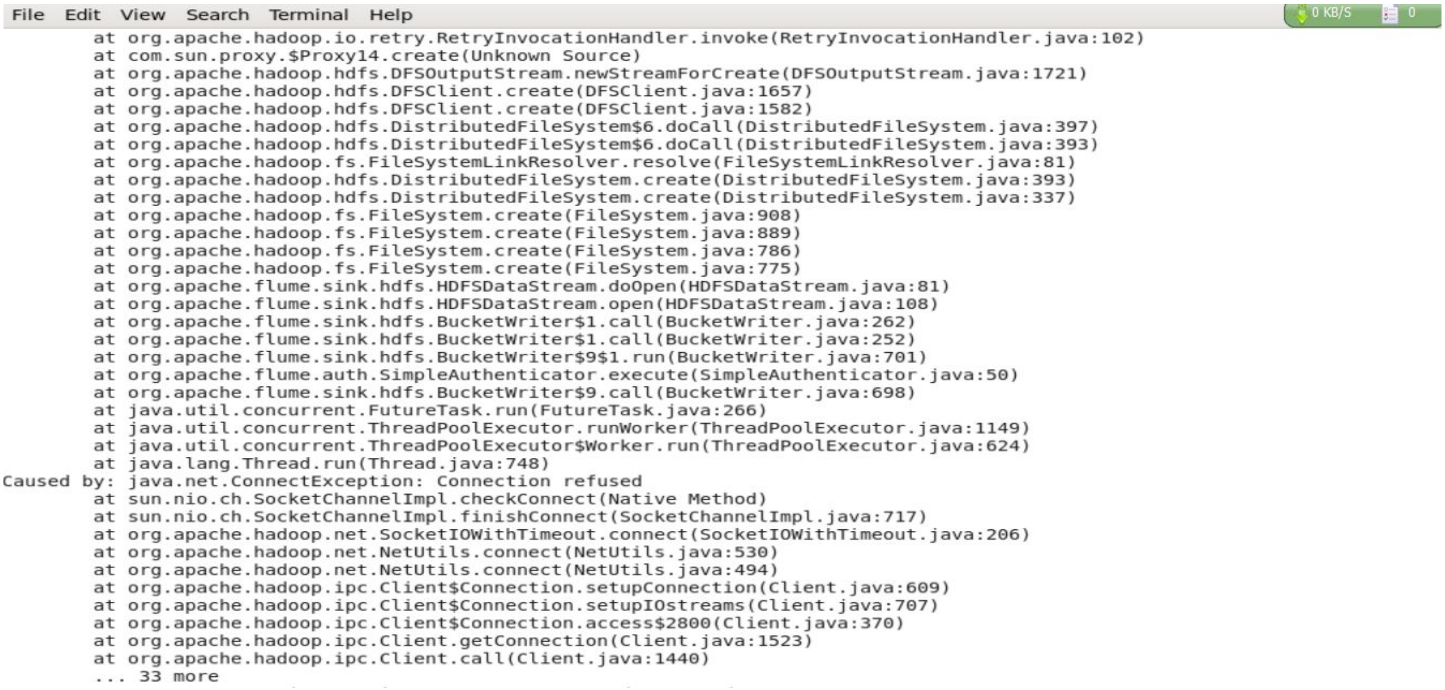
```
[acadgild@localhost ~]$ hadoop fs -mkdir -p /user/flume/tweets
18/10/03 12:37:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -ls /user/flume/
18/10/03 12:37:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x   - acadgild supergroup          0 2018-10-03 12:37 /user/flume/tweets
[acadgild@localhost ~]$ █
```

Step 11:
For fetching the data from Twitter, Use the below command to fetch the twitter tweet data into the HDFS cluster path.
$flume-ng agent –n TwitterAgent –f <location of created/edited conf file>



```
[acadgild@localhost ~]$ flume-ng agent -n TwitterAgent -f /home/acadgild/install/flume/apache-flu
me-1.8.0-bin/conf/flume.conf █
```

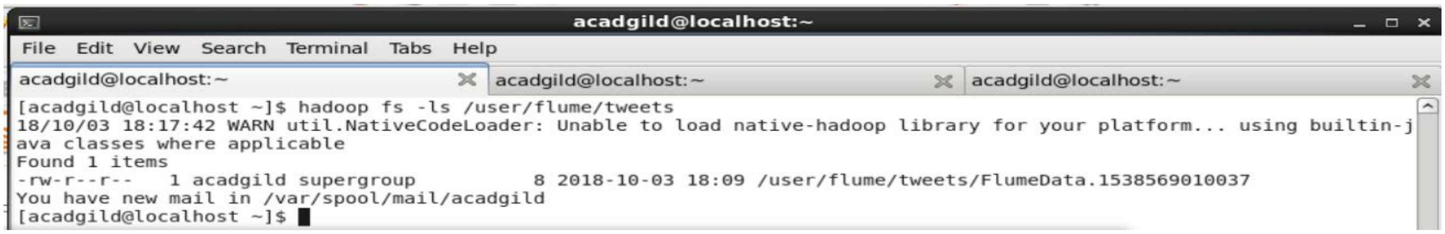
Step 12:
The above command will start fetching data from Twitter and streams it into the HDFS given path.



```
        at org.apache.hadoop.io.retry.RetryInvocationHandler.invoke(RetryInvocationHandler.java:102)
        at com.sun.proxy.$Proxy14.create(Unknown Source)
        at org.apache.hadoop.hdfs.DFSOutputStream.newStreamForCreate(DFSOutputStream.java:1721)
        at org.apache.hadoop.hdfs.DFSClient.create(DFSClient.java:1657)
        at org.apache.hadoop.hdfs.DFSClient.create(DFSClient.java:1582)
        at org.apache.hadoop.hdfs.DistributedFileSystem$6.doCall(DistributedFileSystem.java:397)
        at org.apache.hadoop.hdfs.DistributedFileSystem$6.doCall(DistributedFileSystem.java:393)
        at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java:81)
        at org.apache.hadoop.hdfs.DistributedFileSystem.create(DistributedFileSystem.java:393)
        at org.apache.hadoop.hdfs.DistributedFileSystem.create(DistributedFileSystem.java:337)
        at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:908)
        at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:889)
        at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:786)
        at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:775)
        at org.apache.flume.sink.hdfs.HDFSDataStream.doOpen(HDFSDataStream.java:81)
        at org.apache.flume.sink.hdfs.HDFSDataStream.open(HDFSDataStream.java:108)
        at org.apache.flume.sink.hdfs.BucketWriter$1.call(BucketWriter.java:262)
        at org.apache.flume.sink.hdfs.BucketWriter$1.call(BucketWriter.java:252)
        at org.apache.flume.sink.hdfs.BucketWriter$9$1.run(BucketWriter.java:701)
        at org.apache.flume.auth.SimpleAuthenticator.execute(SimpleAuthenticator.java:50)
        at org.apache.flume.sink.hdfs.BucketWriter$9.call(BucketWriter.java:698)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
Caused by: java.net.ConnectException: Connection refused
        at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
        at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:717)
        at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:206)
        at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:530)
        at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:494)
        at org.apache.hadoop.ipc.Client$Connection.setupConnection(Client.java:609)
        at org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:707)
        at org.apache.hadoop.ipc.Client$Connection.access$2800(Client.java:370)
        at org.apache.hadoop.ipc.Client.getConnection(Client.java:1523)
        at org.apache.hadoop.ipc.Client.call(Client.java:1440)
        ... 33 more
```

Step 13:
Once the tweet data started it into the given HDFS path we can use 'Ctrl+c' command to stop the streaming process.

Step 14:
To check the contents of the tweet data we can use the following command:
hadoop fs –ls /user/flume/tweets



```
[acadgild@localhost ~]$ hadoop fs -ls /user/flume/tweets
18/10/03 18:17:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j
ava classes where applicable
Found 1 items
-rw-r--r--   1 acadgild supergroup          8 2018-10-03 18:09 /user/flume/tweets/FlumeData.1538569010037
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ 
```

We can see that the all the streaming tweets are get stored in the FlumeData.xxx file in hadoop.