

Kalyan Big Data Projects – Project 5 How To Stream CSV Data Into Phoenix Using Apache Flume

Pre-Requisites of Flume Project:

hadoop-2.6.0
flume-1.6.0
hbase-1.1.2
phoenix-4.7.0
java-1.7

NOTE: Make sure that install all the above components

Flume Project Download Links:

`hadoop-2.6.0.tar.gz` ==> [link](#)

(<https://archive.apache.org/dist/hadoop/core/hadoop-2.6.0/hadoop-2.6.0.tar.gz>)

`apache-flume-1.6.0-bin.tar.gz` ==> [link](#)

(<https://archive.apache.org/dist/flume/1.6.0/apache-flume-1.6.0-bin.tar.gz>)

`hbase-1.1.2-bin.tar.gz` ==> [link](#)

(<https://archive.apache.org/dist/hbase/1.1.2/hbase-1.1.2-bin.tar.gz>)

`phoenix-4.7.0-HBase-1.1-bin.tar.gz` ==> [link](#)

(<https://archive.apache.org/dist/phoenix/phoenix-4.7.0-HBase-1.1/bin/phoenix-4.7.0-HBase-1.1-bin.tar.gz>)

`kalyan-csv-phoenix-agent.conf` ==> [link](#)

(<https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/flume/project5-phoenix-csv/kalyan-csv-phoenix-agent.conf>)

`kalyan-bigdata-examples.jar` ==> [link](#)

(<https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/kalyan/kalyan-bigdata-examples.jar>)

`kalyan-phoenix-flume-4.7.0-HBase-1.1.jar` ==> [link](#)

(<https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/kalyan/kalyan-phoenix-flume-4.7.0-HBase-1.1.jar>)

`json-path-2.2.0.jar` ==> [link](#)

(<http://central.maven.org/maven2/com/jayway/jsonpath/json-path/2.2.0/json-path-2.2.0.jar>)

`commons-io-2.4.jar` ==> [link](#)

(<http://central.maven.org/maven2/commons-io/commons-io/2.4/commons-io-2.4.jar>)

Learnings of this Project:

- We will learn Flume Configurations and Commands
 - Flume Agent
 1. Source (Exec Source)
 2. Channel (Memory Channel)
 3. Sink (Phoenix Sink)
 - Major project in Real Time `Product Log Analysis`
 1. We are extracting the data from server logs
 2. This data will be useful to do analysis on product views
 3. CSV is the output format
 - We can use phoenix to analyze this data
-

1. create "**kalyan-csv-phoenix-agent.conf**" file with below content

```
agent.sources = EXEC
agent.channels = MemChannel
agent.sinks = PHOENIX
```

```
agent.sources.EXEC.type = exec
agent.sources.EXEC.command = tail -F /tmp/users.csv
agent.sources.EXEC.channels = MemChannel
```

```
agent.sinks.PHOENIX.type = org.apache.phoenix.flume.sink.PhoenixSink
agent.sinks.PHOENIX.batchSize = 10
agent.sinks.PHOENIX.zookeeperQuorum = localhost
agent.sinks.PHOENIX.table = users3
agent.sinks.PHOENIX.ddl = CREATE TABLE IF NOT EXISTS users3 (userid BIGINT NOT
NULL, username VARCHAR, password VARCHAR, email VARCHAR, country VARCHAR, state
VARCHAR, city VARCHAR, dt VARCHAR NOT NULL CONSTRAINT PK PRIMARY KEY
(userid, dt))
agent.sinks.PHOENIX.serializer = csv
agent.sinks.PHOENIX.serializer.delimiter = ,
agent.sinks.PHOENIX.serializer.columns = userid,username,password,email,country,state,city,dt
agent.sinks.PHOENIX.channel = MemChannel
```

```
agent.channels.MemChannel.type = memory
agent.channels.MemChannel.capacity = 1000
agent.channels.MemChannel.transactionCapacity = 100
```

2. Copy "**kalyan-csv-phoenix-agent.conf**" file into "\$FUME_HOME/conf" folder

3. Copy "**kalyan-phoenix-flume-4.7.0-HBase-1.1.jar, json-path-2.2.0.jar and commons-io-2.4.jar**" files into "\$FLUME_HOME/lib" folder

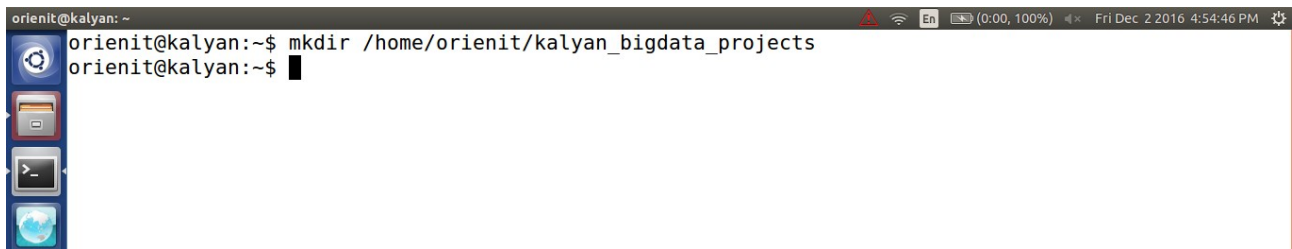
4. Generate Large Amount of Sample JSON data follow this [article](#).

(<http://kalyanbigdatatraining.blogspot.com/2016/12/how-to-generate-large-amount-of-sample.html>)

5. Follow below steps...

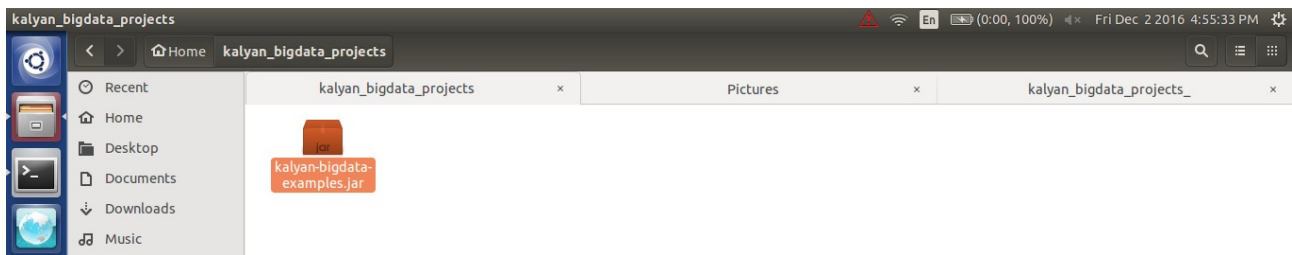
i) Create '**kalyan_bigdata_projects**' folder in **user home** (i.e **/home/orienit**)

Command: `mkdir /home/orienit/kalyan_bigdata_projects`



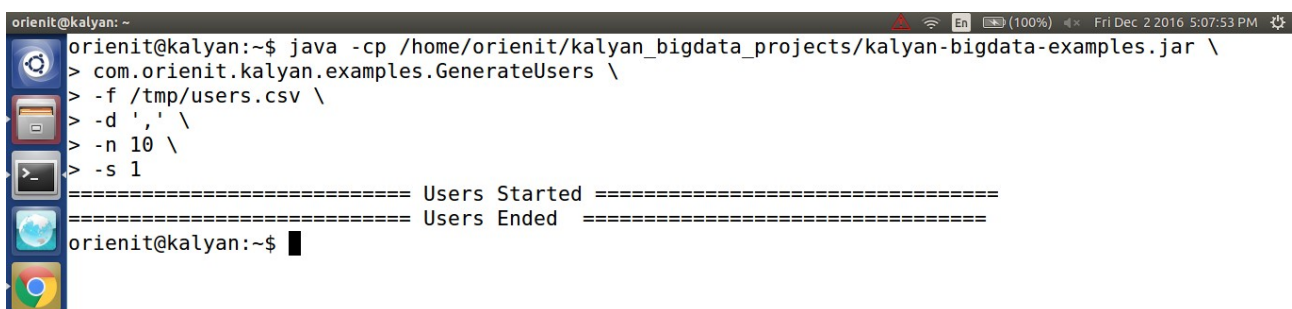
```
orienit@kalyan: ~  
orienit@kalyan:~$ mkdir /home/orienit/kalyan_bigdata_projects  
orienit@kalyan:~$
```

ii) Copy '**kalyan-bigdata-examples.jar**' jar file into '**/home/orienit/kalyan_bigdata_projects**' folder



iii) Execute Below Command to Generate Sample CSV data with 100 lines. Increase this number to get more data ...

```
java -cp /home/orienit/kalyan_bigdata_projects/kalyan-bigdata-examples.jar \  
com.orientit.kalyan.examples.GenerateUsers \  
-f /tmp/users.csv \  
-d ',' \  
-n 100 \  
-s 1
```



```
orienit@kalyan: ~  
orienit@kalyan:~$ java -cp /home/orienit/kalyan_bigdata_projects/kalyan-bigdata-examples.jar \  
> com.orientit.kalyan.examples.GenerateUsers \  
> -f /tmp/users.csv \  
> -d ',' \  
> -n 10 \  
> -s 1  
===== Users Started =====  
===== Users Ended =====  
orienit@kalyan:~$
```

6. Verify the Sample CSV data in Console, using below command

`cat /tmp/users.csv`

```
orienit@kalyan: ~$ cat /tmp/users.csv
1,user1,user1,user1@gmail.com,US,Washington,Seattle,2016-07-02 05:07:48
2,user2,user2,user2@gmail.com,US,Florida,Orlando,2016-07-02 05:07:48
3,user3,user3,user3@gmail.com,US,New York,Little Falls,2016-07-02 05:07:49
4,user4,user4,user4@gmail.com,India,Karnataka,Mangaluru,2016-07-02 05:07:49
5,user5,user5,user5@gmail.com,US,Hawaii,Hanapepe,2016-07-02 05:07:49
6,user6,user6,user6@gmail.com,India,Chennai,Kottur,2016-07-02 05:07:49
7,user7,user7,user7@gmail.com,India,Andhra Pradesh,Kakinada,2016-07-02 05:07:49
8,user8,user8,user8@gmail.com,US,Hawaii,East Honolulu,2016-07-02 05:07:49
9,user9,user9,user9@gmail.com,US,Florida,Hollywood,2016-07-02 05:07:49
10,user10,user10,user10@gmail.com,US,Washington,Bellevue,2016-07-02 05:07:49
orienit@kalyan:~$
```

7. To work with **Flume + Phoenix Integration**, Follow the below steps

i) Start the hbase using below '`start-hbase.sh`' command.

```
orienit@kalyan: ~$ start-hbase.sh
localhost: starting zookeeper, logging to /home/orienit/work/hbase-0.98.4-hadoop2/bin/../logs/hbase-
orienit-zookeeper-kalyan.out
starting master, logging to /home/orienit/work/hbase-0.98.4-hadoop2/logs/hbase-orienit-master-kalya
n.out
localhost: starting regionserver, logging to /home/orienit/work/hbase-0.98.4-hadoop2/bin/../logs/hb
ase-orienit-regionserver-kalyan.out
orienit@kalyan:~$
```

ii. verify the hbase is running or not with "`jps`" command

```
orienit@kalyan: ~$ jps
13904 DataNode
24529 HQuorumPeer
24835 HRegionServer
14259 ResourceManager
24596 HMaster
13749 NameNode
20725 Application
14392 NodeManager
14104 SecondaryNameNode
25486 Jps
7183 org.eclipse.equinox.launcher_1.3.200.v20160318-1642.jar
orienit@kalyan:~$
```

iii. Start the phoenix using below '`sqlline.py localhost`' command.

```
orientit@kalyan: ~$ sqlline.py localhost
Setting property: [incremental, false]
Setting property: [isolation, TRANSACTION_READ_COMMITTED]
issuing: !connect jdbc:phoenix:localhost none none org.apache.phoenix.jdbc.PhoenixDriver
Connecting to jdbc:phoenix:localhost
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/orientit/work/phoenix-4.7.0-HBase-1.1-bin/phoenix-4.7.0-HBase-1.1-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/orientit/work/hadoop-2.6.0/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.
16/10/06 17:25:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Connected to: Phoenix (version 4.7)
Driver: PhoenixEmbeddedDriver (version 4.7)
Autocommit status: true
Transaction isolation: TRANSACTION_READ_COMMITTED
Building list of tables and columns for tab-completion (set fastconnect to true to skip)...
83/83 (100%) Done
Done
sqlline version 1.1.8
0: jdbc:phoenix:localhost>
```

iv. list out all the tables in phoenix using '!tables' command

```
orientit@kalyan: ~$ 0: jdbc:phoenix:localhost> !tables
```

TABLE_CAT	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	REMARKS	TYPE_NAME	SELF_REFERENC
	SYSTEM	CATALOG	SYSTEM TABLE			
	SYSTEM	FUNCTION	SYSTEM TABLE			
	SYSTEM	SEQUENCE	SYSTEM TABLE			
	SYSTEM	STATS	SYSTEM TABLE			

```
0: jdbc:phoenix:localhost>
```

8. Execute the below command to `Extract data from CSV data into Phoenix using Flume`

```
$FLUME_HOME/bin/flume-ng agent -n agent --conf $FLUME_HOME/conf -f $FLUME_HOME/conf/kalyan-csv-phoenix-agent.conf -Dflume.root.logger=DEBUG,console
```

```
orientit@kalyan: ~$ $FLUME_HOME/bin/flume-ng agent -n agent --conf $FLUME_HOME/conf -f $FLUME_HOME/conf/kalyan-csv-phoenix-agent.conf -Dflume.root.logger=DEBUG,console
Info: Sourcing environment configuration script /home/orientit/work/apache-flume-1.6.0-bin/conf/flume-env.sh
Info: Including Hadoop libraries found via (/home/orientit/work/hadoop-2.6.0/bin/hadoop) for HDFS access
Info: Excluding /home/orientit/work/hadoop-2.6.0/share/hadoop/common/lib/slf4j-api-1.7.5.jar from classpath
```

9. Verify the data in console


```
orientit@kalyan: ~
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['state']
2016-10-06 17:48:02,749 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - com.jayway.jsonpat
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['city']
2016-10-06 17:48:02,749 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - com.jayway.jsonpat
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['date']
2016-10-06 17:48:02,750 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - com.jayway.jsonpat
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['userid']
2016-10-06 17:48:02,750 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - com.jayway.jsonpat
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['username']
2016-10-06 17:48:02,750 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - com.jayway.jsonpat
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['password']
2016-10-06 17:48:02,750 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - com.jayway.jsonpat
h.internal.path.CompiledPath.evaluate(CompiledPath.java:47)) Evaluating path: $['email']
```

10. Verify the data in Phoenix , `users3` table is created or not using below command

!tables

```
orientit@kalyan: ~
0: jdbc:phoenix:localhost> !tables
```

TABLE_CAT	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	REMARKS	TYPE_NAME	SELF_REFERENC
	SYSTEM	CATALOG	SYSTEM TABLE			
	SYSTEM	FUNCTION	SYSTEM TABLE			
	SYSTEM	SEQUENCE	SYSTEM TABLE			
	SYSTEM	STATS	SYSTEM TABLE			
		USERS3	TABLE			

```
0: jdbc:phoenix:localhost>
```

11. Execute below command to get the data from phoenix table 'users3'

select count(*) from users3;

select * from users3;

```
orientit@kalyan: ~
0: jdbc:phoenix:localhost> select count(*) from users3;
```

COUNT(1)
10

```
1 row selected (0.07 seconds)
0: jdbc:phoenix:localhost> select * from users3;
```

USERID	USERNAME	PASSWORD	EMAIL	COUNTRY	STATE	CITY
91	user91	user91	user91@gmail.com	India	Chennai	Tamparam
92	user92	user92	user92@gmail.com	US	Florida	South Beach
93	user93	user93	user93@gmail.com	US	New York	Albany
94	user94	user94	user94@gmail.com	India	Chennai	Kottur
95	user95	user95	user95@gmail.com	US	Florida	South Beach
96	user96	user96	user96@gmail.com	India	Telangana	Hyderabad
97	user97	user97	user97@gmail.com	US	Washington	Renton
98	user98	user98	user98@gmail.com	India	Karnataka	Vijayapura
99	user99	user99	user99@gmail.com	India	Andhra Pradesh	Visakhapatn
100	user100	user100	user100@gmail.com	India	Karnataka	Bidar

```
10 rows selected (0.111 seconds)
0: jdbc:phoenix:localhost>
```