Kalyan Big Data Projects – Project 6
How To Stream JSON 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-json-phoenix-agent.conf` ==> link (https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/flume/project6-phoenix-json/kalyan-json-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. JSON is the output format
- We can use Phoenix to analyze this data

---

1. create "**kalyan-json-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.json
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 = users2
agent.sinks.PHOENIX.ddl = CREATE TABLE IF NOT EXISTS users2 (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 = json
agent.sinks.PHOENIX.serializer.columnsMapping = {"userid":"userid", "username":"username",
"password":"password", "email":"email", "country":"country", "state":"state", "city":"city",
"dt":"dt"}
agent.sinks.PHOENIX.serializer.partialSchema = true
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-json-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

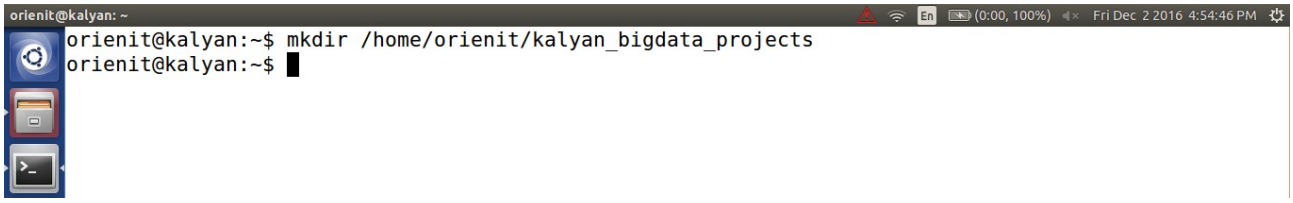
4. Generate Large Amount of Sample JSON data follow this article.

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

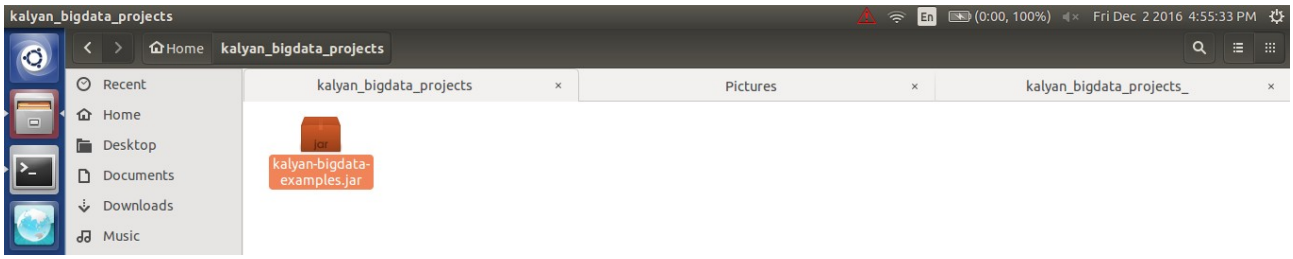
5. Follow below steps...

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

Command: *mkdir /home/orienit/kalyan_bigdata_projects*

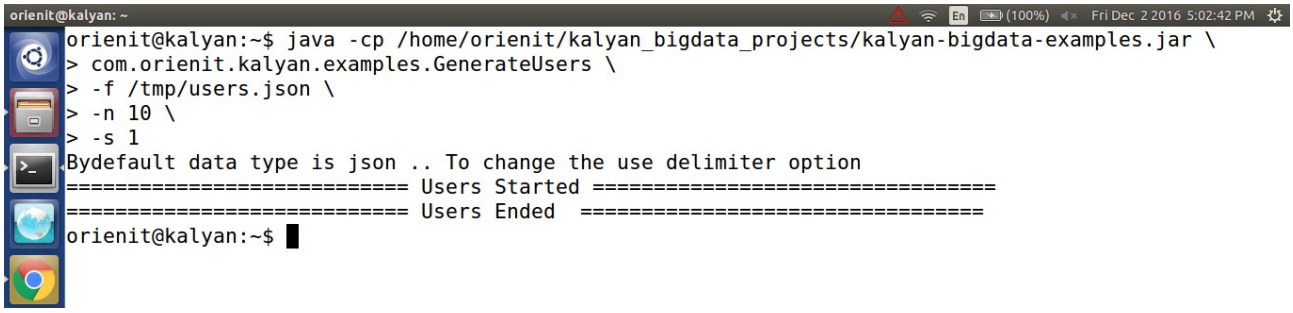


ii) Copy '**kalyan-bigdata-examples.jar**' jar file into '**/home/orienit/kalyan_bigdata_projects**' folder



iii) Execute below command to Generate Sample JSON data with 100 lines. Increase this number to get more data ...

```
java -cp /home/orienit/kalyan_bigdata_projects/kalyan-bigdata-examples.jar \
com.orienit.kalyan.examples.GenerateUsers \
-f /tmp/users.json \
-n 100 \
-s 1
```



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

cat /tmp/users.json

```
orienit@kalyan: ~
orienit@kalyan:~$ cat /tmp/users.json
{"userid":1,"username":"user1","password":"user1","email":"user1@gmail.com","country":"India","stat
e":"Andhra Pradesh","city":"Guntur","dt":"2016-02-02 05:02:39"}
{"userid":2,"username":"user2","password":"user2","email":"user2@gmail.com","country":"US","state":
"Washington","city":"Renton","dt":"2016-02-02 05:02:39"}
{"userid":3,"username":"user3","password":"user3","email":"user3@gmail.com","country":"US","state":
"Hawaii","city":"Hanapepe","dt":"2016-02-02 05:02:39"}
{"userid":4,"username":"user4","password":"user4","email":"user4@gmail.com","country":"US","state":
"Washington","city":"Bellingham","dt":"2016-02-02 05:02:39"}
{"userid":5,"username":"user5","password":"user5","email":"user5@gmail.com","country":"India","stat
e":"Andhra Pradesh","city":"Kakinada","dt":"2016-02-02 05:02:39"}
{"userid":6,"username":"user6","password":"user6","email":"user6@gmail.com","country":"India","stat
e":"Telangana","city":"Karimnagar","dt":"2016-02-02 05:02:39"}
{"userid":7,"username":"user7","password":"user7","email":"user7@gmail.com","country":"US","state":
"New York","city":"Albany","dt":"2016-02-02 05:02:40"}
{"userid":8,"username":"user8","password":"user8","email":"user8@gmail.com","country":"India","stat
e":"Karnataka","city":"Bidar","dt":"2016-02-02 05:02:40"}
{"userid":9,"username":"user9","password":"user9","email":"user9@gmail.com","country":"India","stat
e":"Chennai","city":"Virugambakkam","dt":"2016-02-02 05:02:40"}
{"userid":10,"username":"user10","password":"user10","email":"user10@gmail.com","country":"US","sta
te":"Hawaii","city":"Honolulu","dt":"2016-02-02 05:02:40"}
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: ~
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: ~
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.

```
orienit@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/orienit/work/phoenix-4.7.0-HBase-1.1-bin/phoenix-4.7.0-HBas
e-1.1-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/orienit/work/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log
4j12-1.7.5.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 platfor
m... 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

```
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 JSON data into Phoenix using Flume**`

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

```
orienit@kalyan:~$ $FLUME_HOME/bin/flume-ng agent -n agent --conf $FLUME_HOME/conf -f $FLUME_HOME/co
nf/kalyan-json-phoenix-agent.conf -Dflume.root.logger=DEBUG,console
Info: Sourcing environment configuration script /home/orienit/work/apache-flume-1.6.0-bin/conf/flum
e-env.sh
Info: Including Hadoop libraries found via (/home/orienit/work/hadoop-2.6.0/bin/hadoop) for HDFS ac
cess
Info: Excluding /home/orienit/work/hadoop-2.6.0/share/hadoop/common/lib/slf4j-api-1.7.5.jar from cl
asspath
Info: Excluding /home/orienit/work/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar fro
m classpath
Info: Including HBASE libraries found via (/home/orienit/work/hbase-1.1.2/bin/hbase) for HBASE acce
```

9. Verify the data in console

```
orienit@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 , `**users2**` table is created or not using below command

!tables

```
orienit@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  |          |            |                |
|            |             | USERS2     | TABLE         |          |            |                |
+------------+-------------+------------+---------------+----------+------------+----------------+
0: jdbc:phoenix:localhost>
```

11. Execute below command to get the data from phoenix table '**users2**'

select count(*) from users2;
select * from users2;

```
0: jdbc:phoenix:localhost> select count(*) from users2;
+-----------+
| COUNT(1)  |
+-----------+
| 10        |
+-----------+
1 row selected (0.034 seconds)
0: jdbc:phoenix:localhost> select * from users2;
+---------+-----------+-----------+--------------------+---------+-----------------+-------------+
| USERID  | USERNAME  | PASSWORD  |       EMAIL        | COUNTRY |      STATE      |     CITY    |
+---------+-----------+-----------+--------------------+---------+-----------------+-------------+
| 91      | user91    | user91    | user91@gmail.com   | US      | Hawaii          | Honolulu    |
| 92      | user92    | user92    | user92@gmail.com   | India   | Karnataka       | Bengaluru   |
| 93      | user93    | user93    | user93@gmail.com   | India   | Karnataka       | Bagalkot    |
| 94      | user94    | user94    | user94@gmail.com   | India   | Telangana       | Hyderabad   |
| 95      | user95    | user95    | user95@gmail.com   | India   | Telangana       | Nizamabad   |
| 96      | user96    | user96    | user96@gmail.com   | US      | New York        | Niagara Fal |
| 97      | user97    | user97    | user97@gmail.com   | US      | Hawaii          | Honolulu    |
| 98      | user98    | user98    | user98@gmail.com   | India   | Andhra Pradesh  | Kakinada    |
| 99      | user99    | user99    | user99@gmail.com   | India   | Chennai         | Virugambakk |
| 100     | user100   | user100   | user100@gmail.com  | US      | Washington      | Renton      |
+---------+-----------+-----------+--------------------+---------+-----------------+-------------+
10 rows selected (0.055 seconds)
0: jdbc:phoenix:localhost>
```