Kalyan Big Data Projects – Project 9
How To Stream JSON Data Into Hive Using Apache Flume

Pre-Requisites of Flume + Hive Project:

hadoop-2.6.0
flume-1.6.0
hive-1.2.1
java-1.7

NOTE: Make sure that install all the above components

Flume + Hive 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)

\`apache-hive-1.2.1-bin.tar.gz\` ==> link
(http://mirror.fibergrid.in/apache/hive/hive-1.2.1/apache-hive-1.2.1-bin.tar.gz)

\`kalyan-json-hive-agent.conf\` ==> link
(https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/flume/project9-hive-json/kalyan-json-hive-agent.conf)

\`kalyan-bigdata-examples.jar\` ==> link
(https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/kalyan/kalyan-bigdata-examples.jar)

Learnings of this Project:

- We will learn Flume Configurations and Commands
- Flume Agent
  1. Source (Exec Source)
  2. Channel (Memory Channel)
  3. Sink (Hive 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 Hive to analyze this data

---

1. create "**kalyan-json-hive-agent.conf**" file with below content

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

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

agent.sinks.HIVE.type = hive
agent.sinks.HIVE.hive.metastore = thrift://localhost:9083
agent.sinks.HIVE.hive.database = kalyan
agent.sinks.HIVE.hive.table = users2
agent.sinks.HIVE.serializer = JSON
agent.sinks.HIVE.channel = MemChannel

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

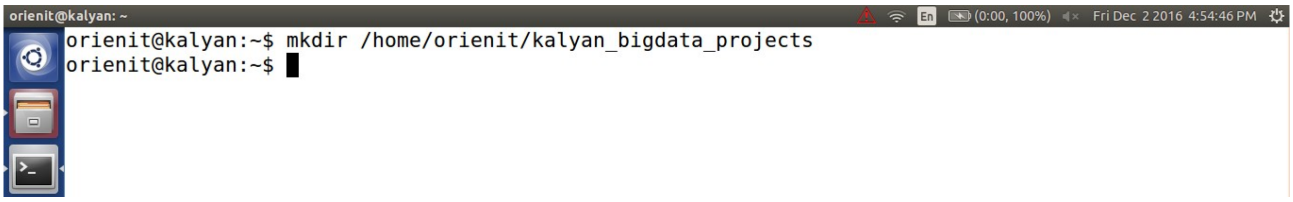
2. Copy "**kalyan-json-hive-agent.conf**" file into "**$FUME_HOME/conf**" folder

3. Copy "**kalyan-bigdata-examples.jar**" file 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)

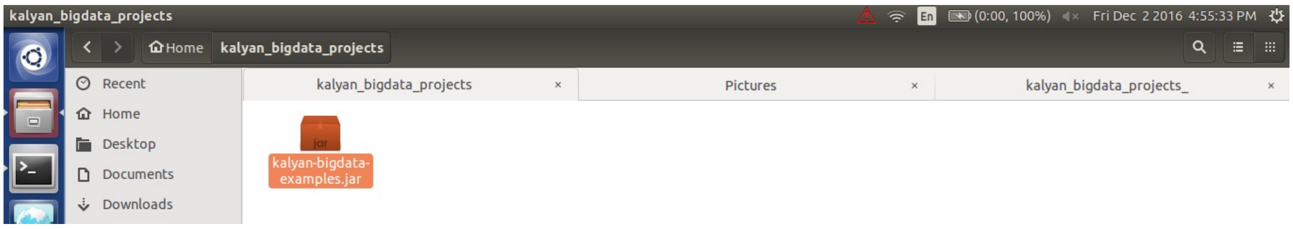
5. Follow below steps...

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



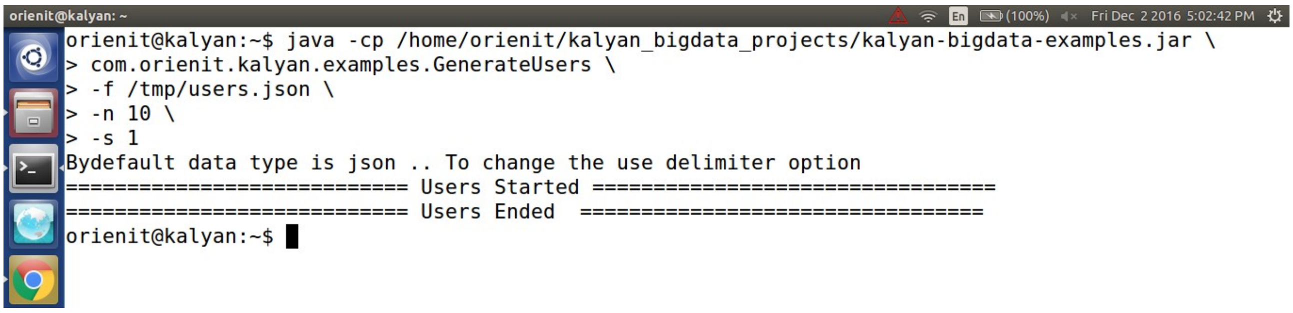
```
orienit@kalyan:~$ mkdir /home/orienit/kalyan_bigdata_projects
orienit@kalyan:~$ 
```

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



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

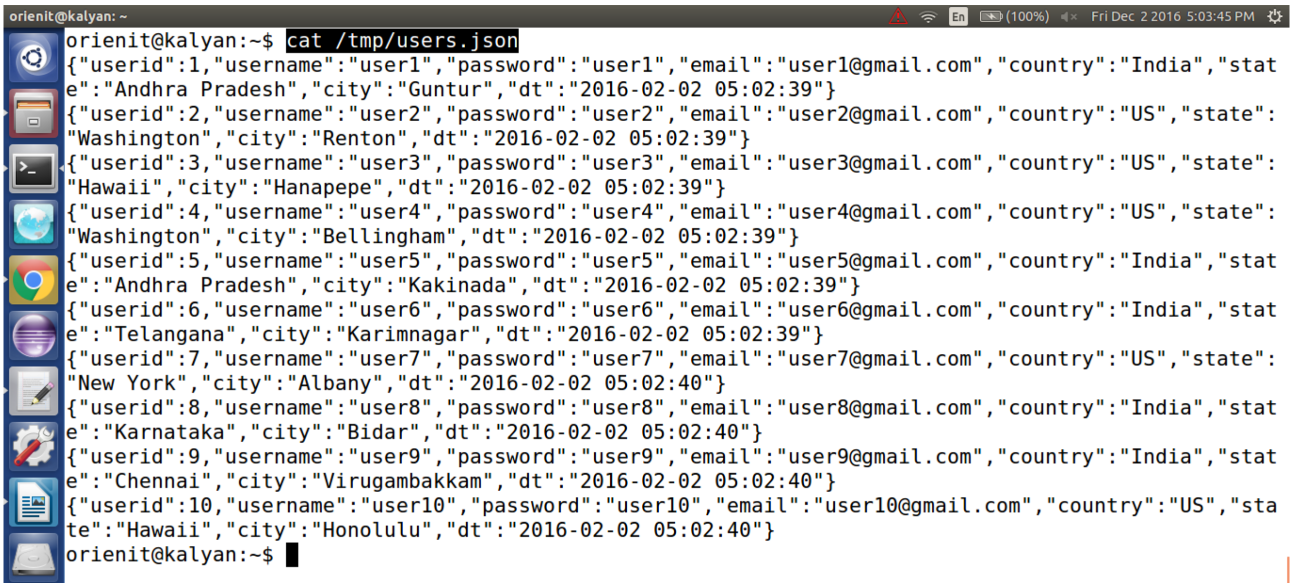
```
java -cp /home/orienit/kalyan_bigdata_projects/kalyan-bigdata-examples.jar \
com.orienit.kalyan.examples.GenerateUsers \
-f /tmp/users.json \
-n 100 \
-s 1
```



```
orienit@kalyan:~$ java -cp /home/orienit/kalyan_bigdata_projects/kalyan-bigdata-examples.jar \
> com.orienit.kalyan.examples.GenerateUsers \
> -f /tmp/users.json \
> -n 10 \
> -s 1
Bydefault data type is json .. To change the use delimiter option
============================ Users Started =================================
============================ Users Ended  =================================
orienit@kalyan:~$ 
```

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

```
cat /tmp/users.json
```



```
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:~$ 
```

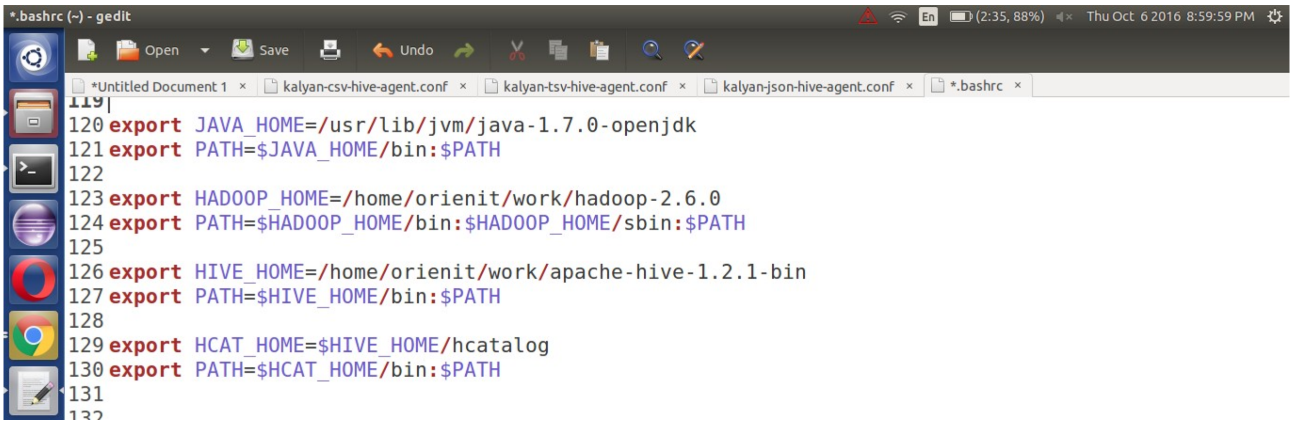
7. To work with **Flume + Hive Integration**, Follow the below steps

Follow this aritcle to work with below procedure.

Refer: http://kalyanbigdatatraining.blogspot.in/2016/10/how-to-work-with-acid-functionality-in.html

i) update '**~/.bashrc**' file with below changes

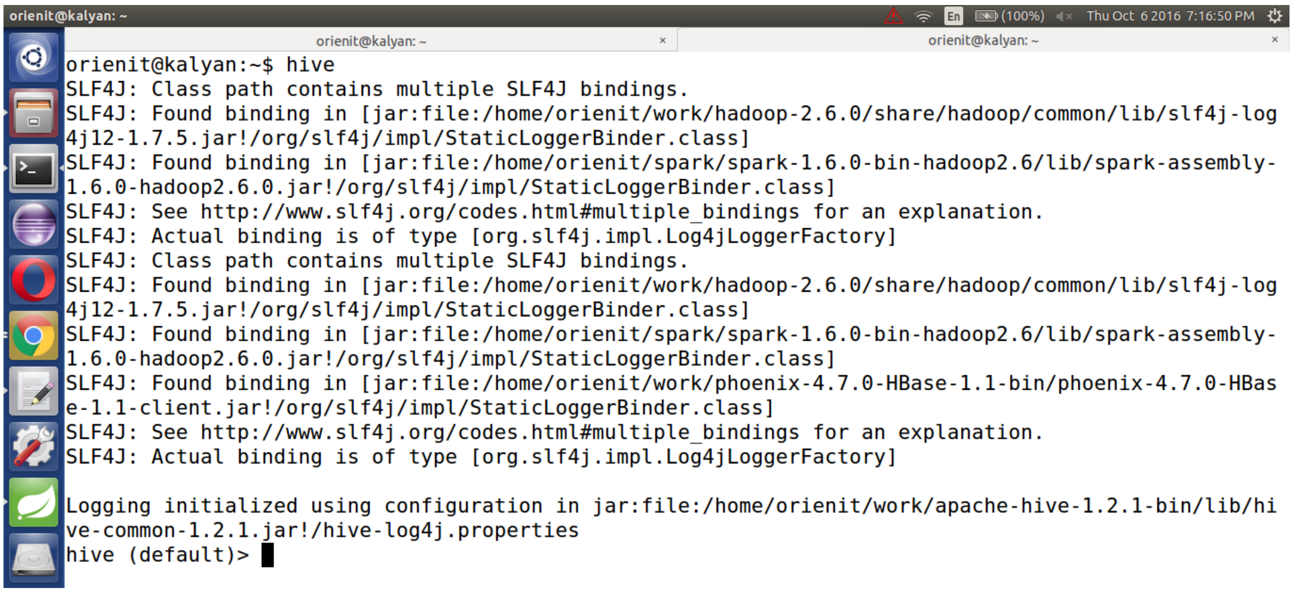
```
export HIVE_HOME=/home/orienit/work/apache-hive-1.2.1-bin
export PATH=$HIVE_HOME/bin:$PATH

export HCAT_HOME=$HIVE_HOME/hcatalog
export PATH=$HCAT_HOME/bin:$PATH
```


*.bashrc (~) - gedit
Thu Oct 6 2016 8:59:59 PM
Open
Save
Undo
*Untitled Document 1
kalyan-csv-hive-agent.conf
kalyan-tsv-hive-agent.conf
kalyan-json-hive-agent.conf
*.bashrc

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
export PATH=$JAVA_HOME/bin:$PATH

export HADOOP_HOME=/home/orienit/work/hadoop-2.6.0
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

export HIVE_HOME=/home/orienit/work/apache-hive-1.2.1-bin
export PATH=$HIVE_HOME/bin:$PATH

export HCAT_HOME=$HIVE_HOME/hcatalog
export PATH=$HCAT_HOME/bin:$PATH
```

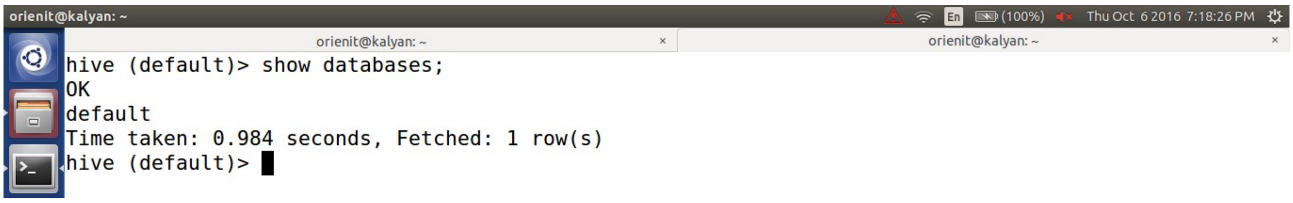
ii. reopen the Terminal

iii. start the hive using '**hive**' command.



```
orienit@kalyan:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
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: Found binding in [jar:file:/home/orienit/spark/spark-1.6.0-bin-hadoop2.6/lib/spark-assembly-
1.6.0-hadoop2.6.0.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]
SLF4J: Class path contains multiple SLF4J bindings.
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: Found binding in [jar:file:/home/orienit/spark/spark-1.6.0-bin-hadoop2.6/lib/spark-assembly-
1.6.0-hadoop2.6.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
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: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/orienit/work/apache-hive-1.2.1-bin/lib/hi
ve-common-1.2.1.jar!/hive-log4j.properties
hive (default)>
```

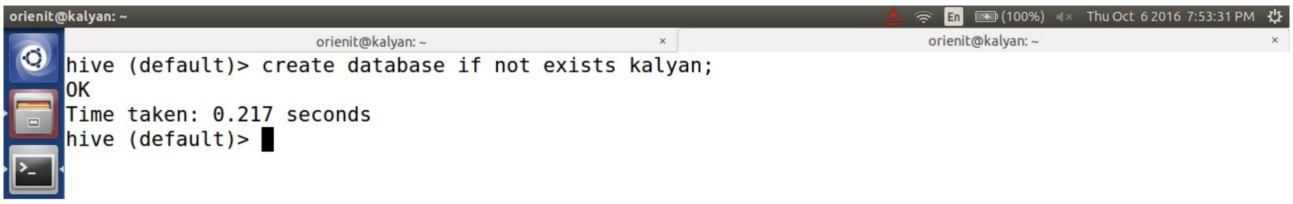
iv. list out all the databases in hive using '**show databases;**' command



```
hive (default)> show databases;
OK
default
Time taken: 0.984 seconds, Fetched: 1 row(s)
hive (default)>
```

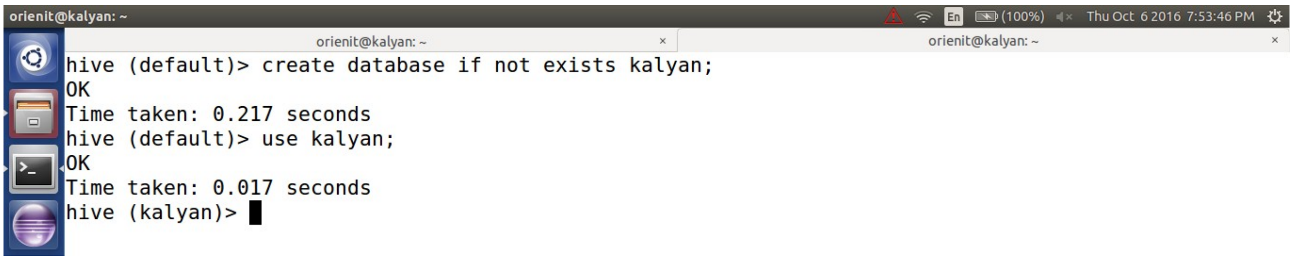
v. create a new database (**kalyan**) in hive using below command.

create database if not exists kalyan;



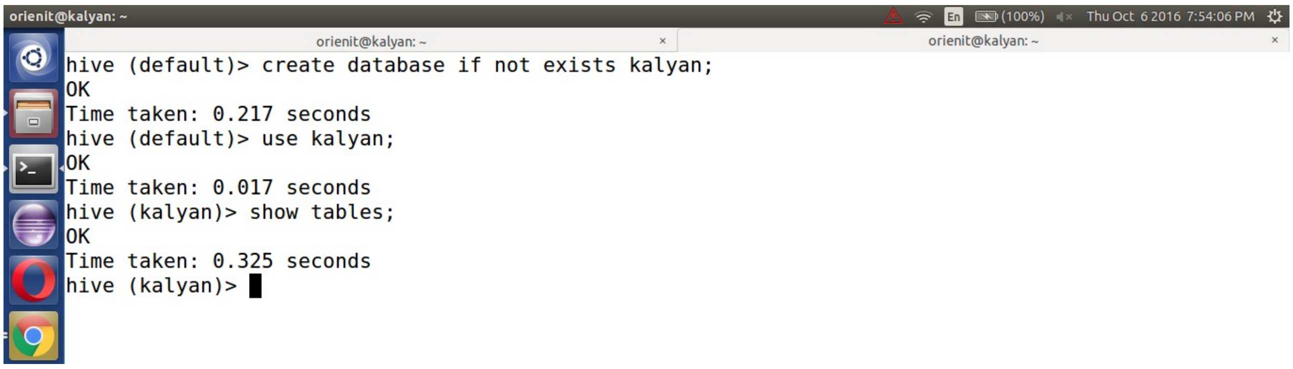
```
hive (default)> create database if not exists kalyan;
OK
Time taken: 0.217 seconds
hive (default)>
```

vi. use kalyan database using '**use kalyan;**' command



```
hive (default)> create database if not exists kalyan;
OK
Time taken: 0.217 seconds
hive (default)> use kalyan;
OK
Time taken: 0.017 seconds
hive (kalyan)>
```

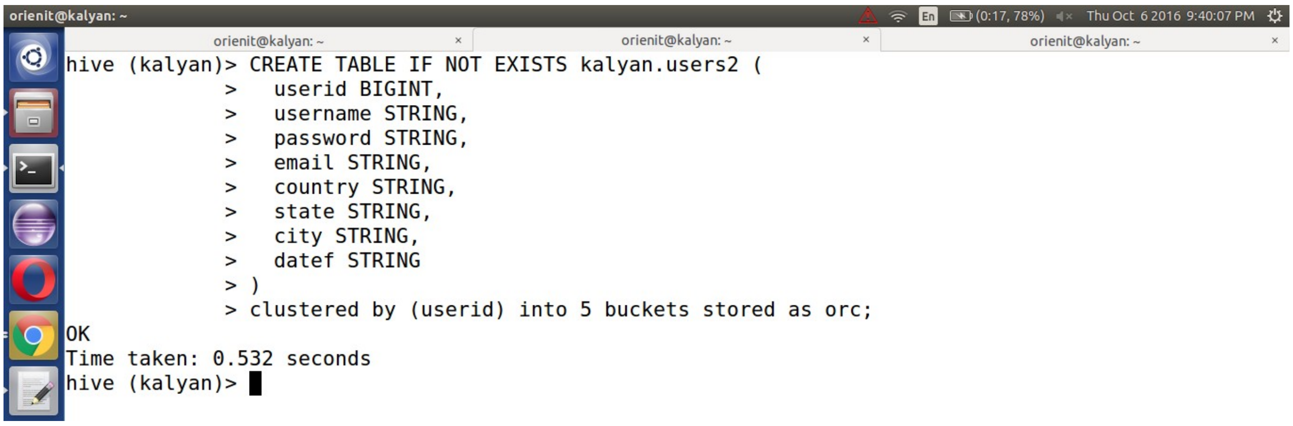
vii. list out all the tables in kalyan database using '**show tables;**' command.



```
hive (default)> create database if not exists kalyan;
OK
Time taken: 0.217 seconds
hive (default)> use kalyan;
OK
Time taken: 0.017 seconds
hive (kalyan)> show tables;
OK
Time taken: 0.325 seconds
hive (kalyan)>
```

viii. create '**users2**' table in **kalyan** database using below command.

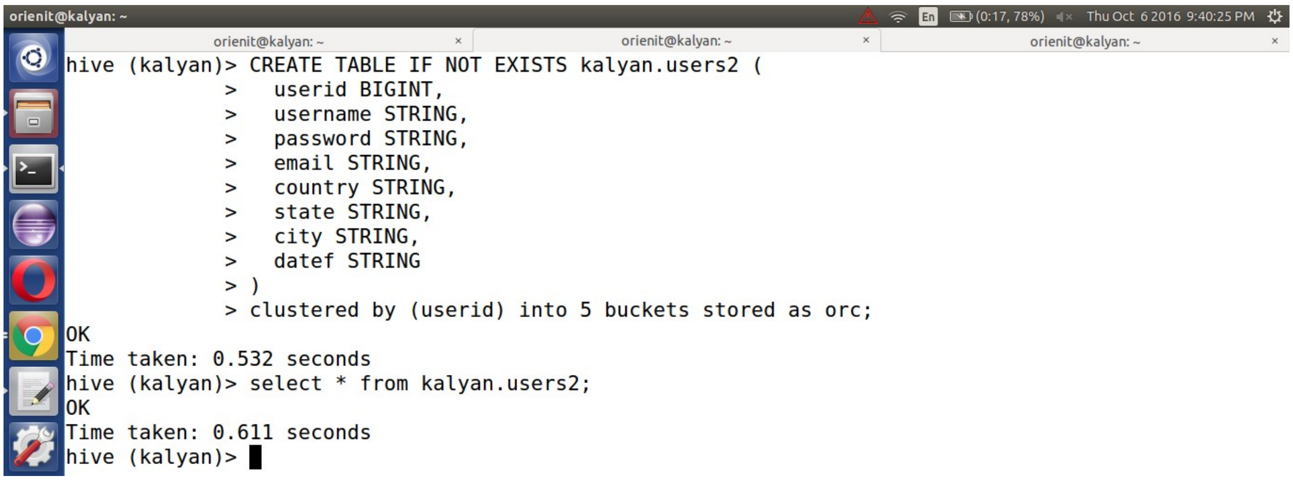
```
CREATE TABLE IF NOT EXISTS kalyan.users2 (
 userid BIGINT,
 username STRING,
 password STRING,
 email STRING,
 country STRING,
 state STRING,
 city STRING,
 dt STRING
)
clustered by (userid) into 5 buckets stored as orc;
```



```
hive (kalyan)> CREATE TABLE IF NOT EXISTS kalyan.users2 (
             >   userid BIGINT,
             >   username STRING,
             >   password STRING,
             >   email STRING,
             >   country STRING,
             >   state STRING,
             >   city STRING,
             >   datef STRING
             > )
             > clustered by (userid) into 5 buckets stored as orc;
OK
Time taken: 0.532 seconds
hive (kalyan)>
```

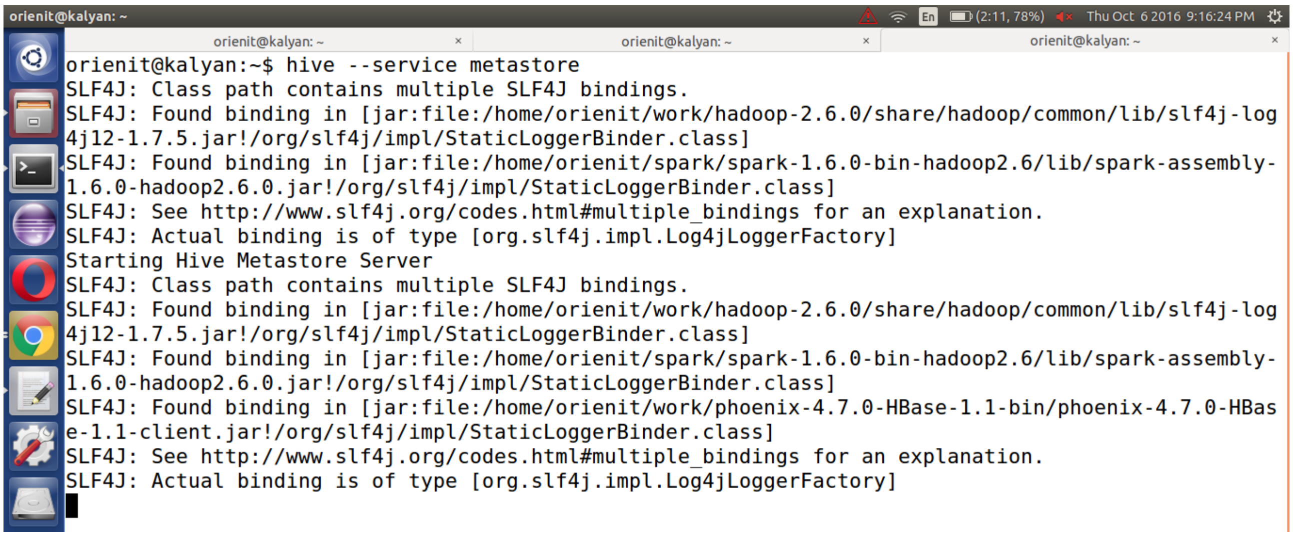
ix. Display the data from 'users2' table using below command

select * from users2;



```
hive (kalyan)> CREATE TABLE IF NOT EXISTS kalyan.users2 (
             >   userid BIGINT,
             >   username STRING,
             >   password STRING,
             >   email STRING,
             >   country STRING,
             >   state STRING,
             >   city STRING,
             >   datef STRING
             > )
             > clustered by (userid) into 5 buckets stored as orc;
OK
Time taken: 0.532 seconds
hive (kalyan)> select * from kalyan.users2;
OK
Time taken: 0.611 seconds
hive (kalyan)>
```

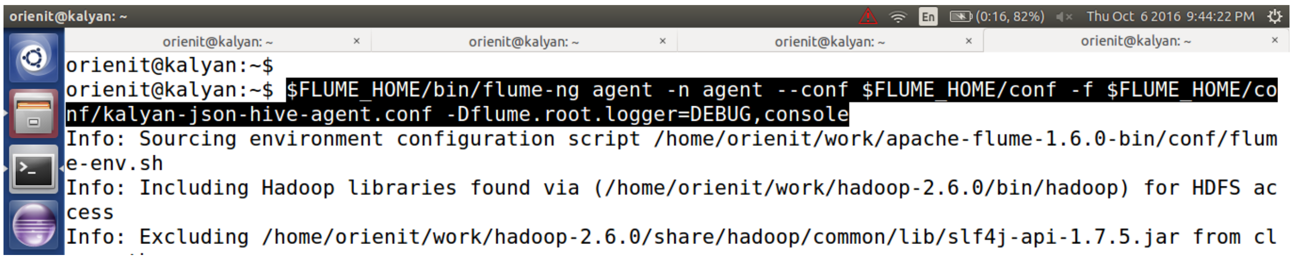
x. start the hive in external metastore db mode using below command
hive --service metastore



```
orienit@kalyan:~$ hive --service metastore
SLF4J: Class path contains multiple SLF4J bindings.
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: Found binding in [jar:file:/home/orienit/spark/spark-1.6.0-bin-hadoop2.6/lib/spark-assembly-
1.6.0-hadoop2.6.0.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]
Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
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: Found binding in [jar:file:/home/orienit/spark/spark-1.6.0-bin-hadoop2.6/lib/spark-assembly-
1.6.0-hadoop2.6.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
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: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

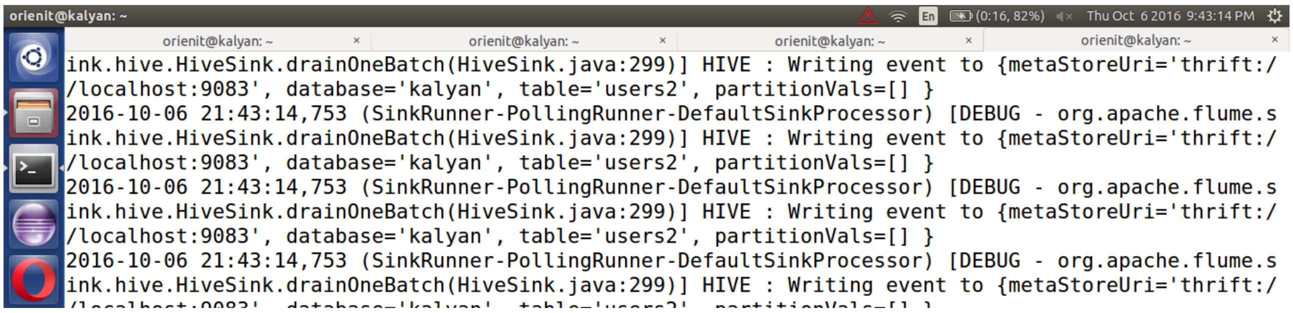
8. Execute the below command to `**Extract data from JSON data into Hive using Flume**`

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



```
orienit@kalyan:~$
orienit@kalyan:~$ $FLUME_HOME/bin/flume-ng agent -n agent --conf $FLUME_HOME/conf -f $FLUME_HOME/co
nf/kalyan-json-hive-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
```

9. Verify the data in console

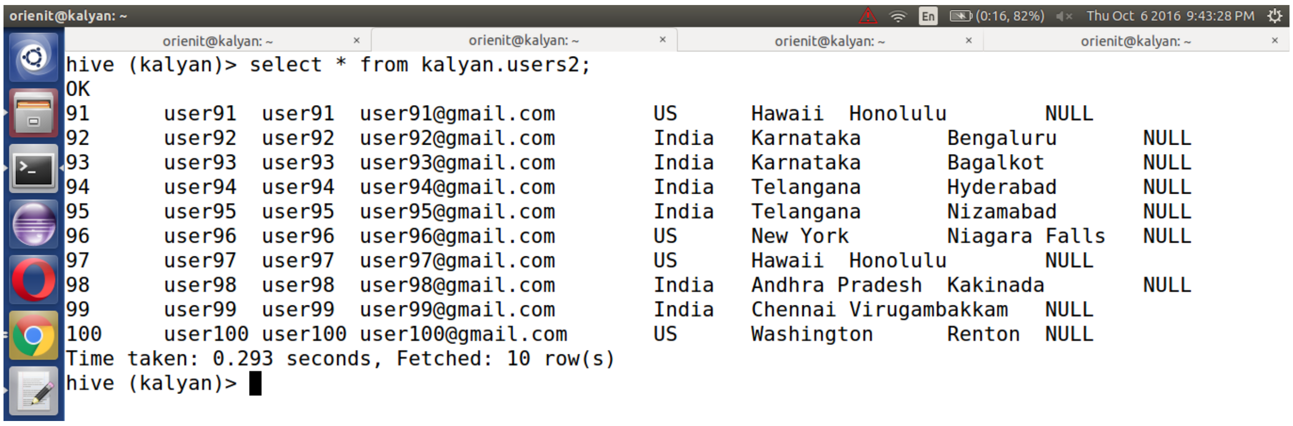


```
ink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift:/
/localhost:9083', database='kalyan', table='users2', partitionVals=[] }
2016-10-06 21:43:14,753 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - org.apache.flume.s
ink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift:/
/localhost:9083', database='kalyan', table='users2', partitionVals=[] }
2016-10-06 21:43:14,753 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - org.apache.flume.s
ink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift:/
/localhost:9083', database='kalyan', table='users2', partitionVals=[] }
2016-10-06 21:43:14,753 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - org.apache.flume.s
ink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift:/
```

10. Verify the data in Hive

Execute below command to get the data from hive table 'users2'

select * from users2;



```
hive (kalyan)> select * from kalyan.users2;
OK
91      user91  user91  user91@gmail.com        US      Hawaii  Honolulu        NULL
92      user92  user92  user92@gmail.com        India   Karnataka       Bengaluru       NULL
93      user93  user93  user93@gmail.com        India   Karnataka       Bagalkot        NULL
94      user94  user94  user94@gmail.com        India   Telangana       Hyderabad       NULL
95      user95  user95  user95@gmail.com        India   Telangana       Nizamabad       NULL
96      user96  user96  user96@gmail.com        US      New York        Niagara Falls   NULL
97      user97  user97  user97@gmail.com        US      Hawaii  Honolulu        NULL
98      user98  user98  user98@gmail.com        India   Andhra Pradesh  Kakinada        NULL
99      user99  user99  user99@gmail.com        India   Chennai Virugambakkam   NULL
100     user100 user100 user100@gmail.com       US      Washington      Renton  NULL
Time taken: 0.293 seconds, Fetched: 10 row(s)
hive (kalyan)> 
```