

Kalyan Big Data Projects – Project 8

How To Stream CSV 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)
(<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)
(<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)
(<http://mirror.fibergrid.in/apache/hive/hive-1.2.1/apache-hive-1.2.1-bin.tar.gz>)

`kalyan-csv-hive-agent.conf` ==> [link](https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/flume/project8-hive-csv/kalyan-csv-hive-agent.conf)
(<https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/flume/project8-hive-csv/kalyan-csv-hive-agent.conf>)

`kalyan-bigdata-examples.jar` ==> [link](https://github.com/kalyanhadooptraining/kalyan-bigdata-realtime-projects/blob/master/kalyan/kalyan-bigdata-examples.jar)
(<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. CSV is the output format
 - We can use hive to analyze this data
-

1. create "**kalyan-csv-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.csv
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 = users1
agent.sinks.HIVE.serializer = DELIMITED
agent.sinks.HIVE.serializer.delimiter = ","
agent.sinks.HIVE.serializer.fieldnames=userid,username,password,email,country,state,city,dt
agent.sinks.HIVE.channel = MemChannel

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

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

3. Copy "**kalyan-bigdata-examples.jar**" file into "**\$FLUME_HOME/lib**" folder

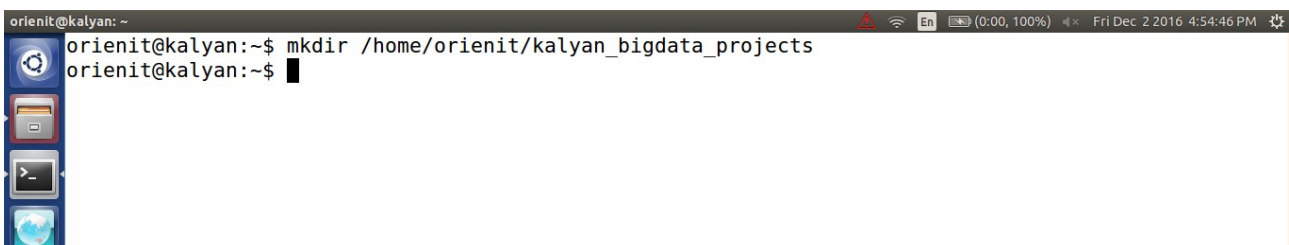
4. Generate Large Amount of Sample CSV data follow this [article](http://kalyanbigdatatraining.blogspot.com/2016/12/how-to-generate-large-amount-of-sample.html).

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

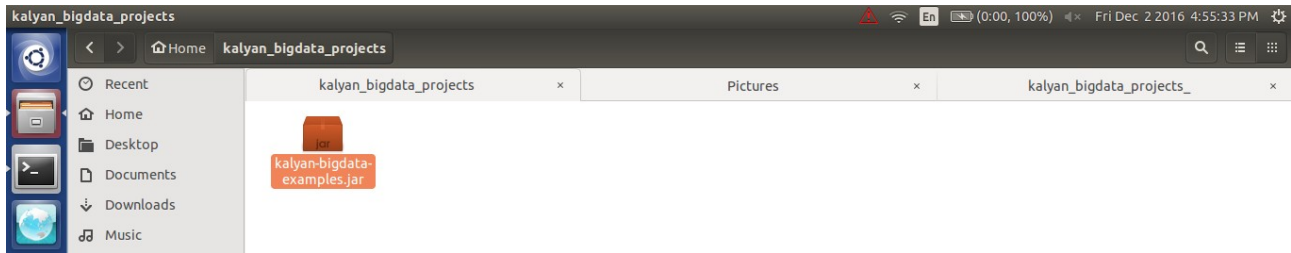
5. Follow below steps...

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

Command: `mkdir /home/orienit/kalyan_bigdata_projects`

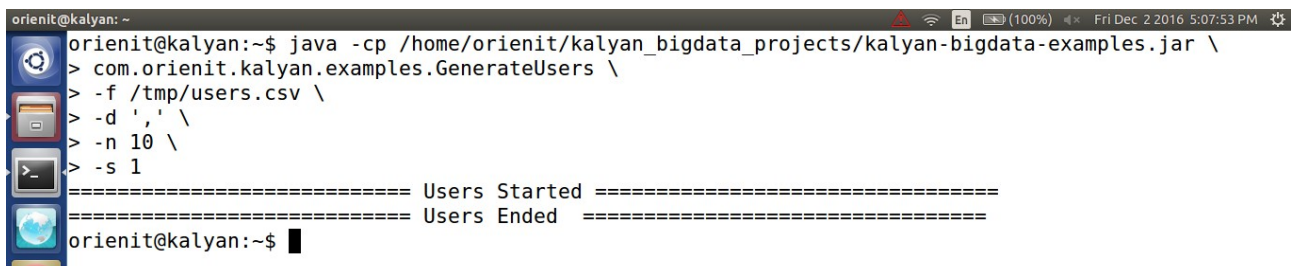


ii) Copy '**kalyan-bigdata-examples.jar**' jar file into '**/home/orienit/kalyan_bigdata_projects**' folder



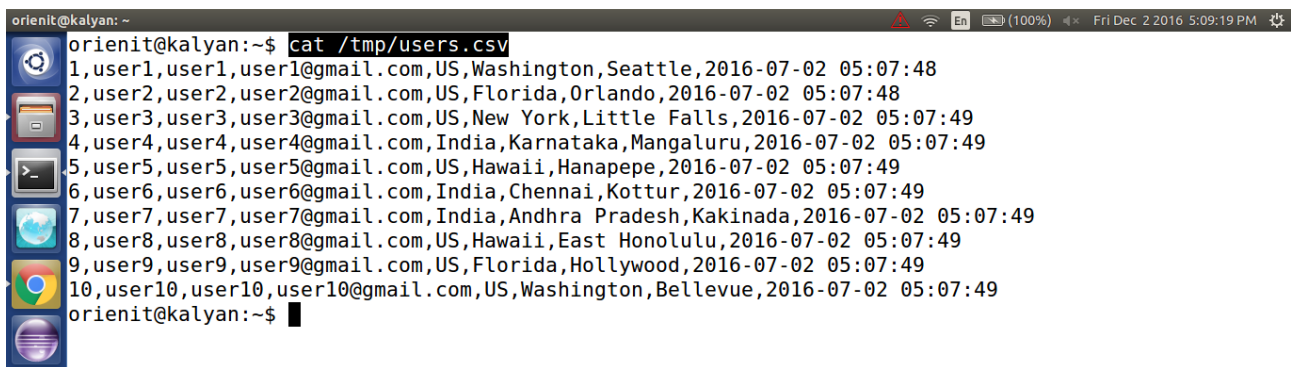
iii) Execute Below Command to Generate Sample CSV data with 100 lines. Increase this number to get more data ...

```
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
```



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

```
cat /tmp/users.csv
```



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

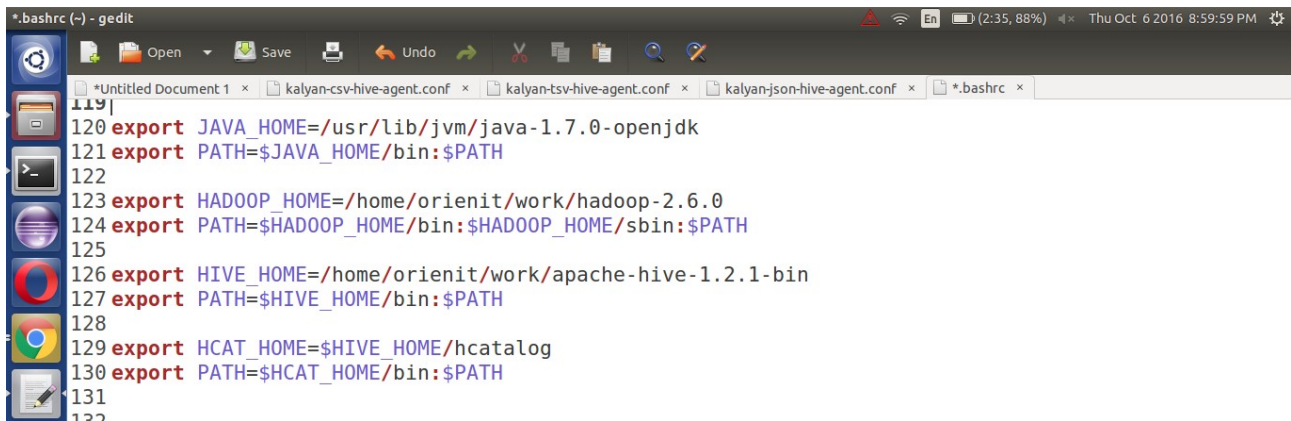
Follow this [article](#) to work with below procedure.

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
```

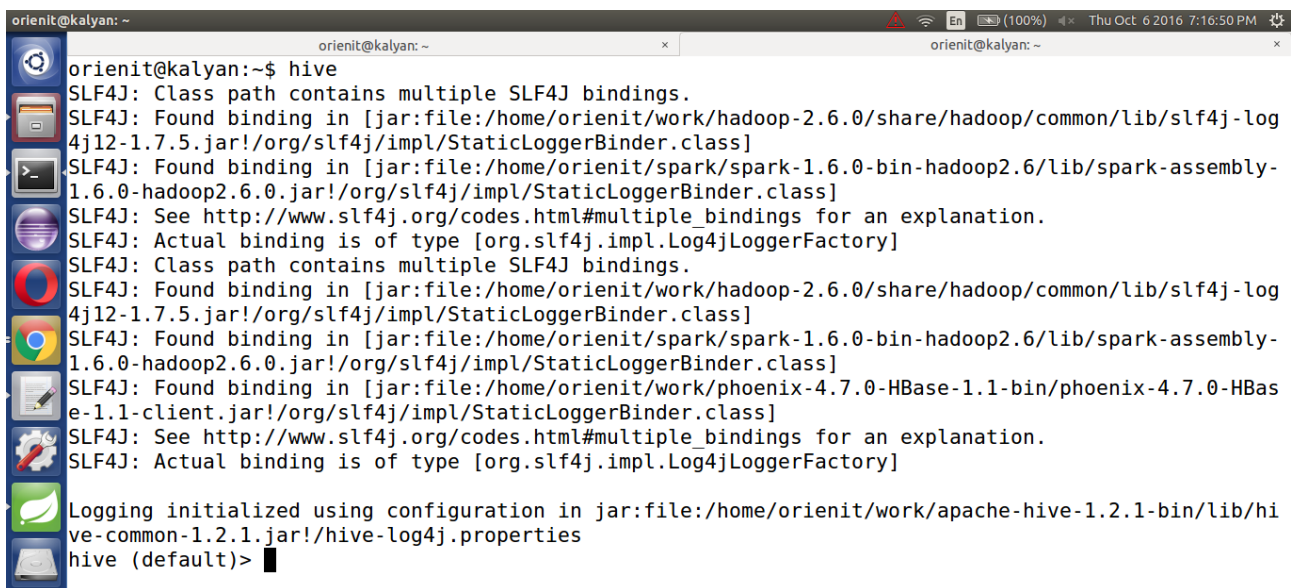
```
export HCAT_HOME=$HIVE_HOME/hcatalog
export PATH=$HCAT_HOME/bin:$PATH
```



```
*.bashrc (-) - gedit
119|
120 export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
121 export PATH=$JAVA_HOME/bin:$PATH
122
123 export HADOOP_HOME=/home/orienit/work/hadoop-2.6.0
124 export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
125
126 export HIVE_HOME=/home/orienit/work/apache-hive-1.2.1-bin
127 export PATH=$HIVE_HOME/bin:$PATH
128
129 export HCAT_HOME=$HIVE_HOME/hcatalog
130 export PATH=$HCAT_HOME/bin:$PATH
131
132
```

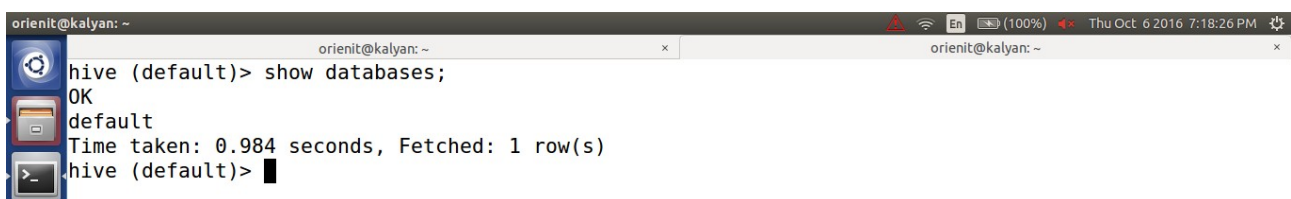
ii. reopen the Terminal

iii. start the hive using 'hive' command.



```
orienit@kalyan: ~
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)> █
```

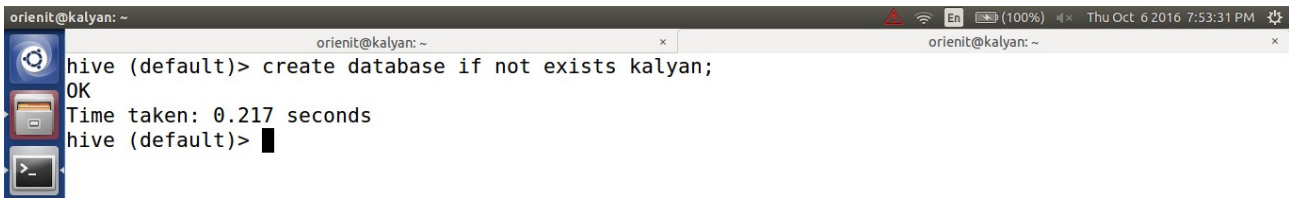
iv. list out all the databases in hive using 'show databases;' command



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

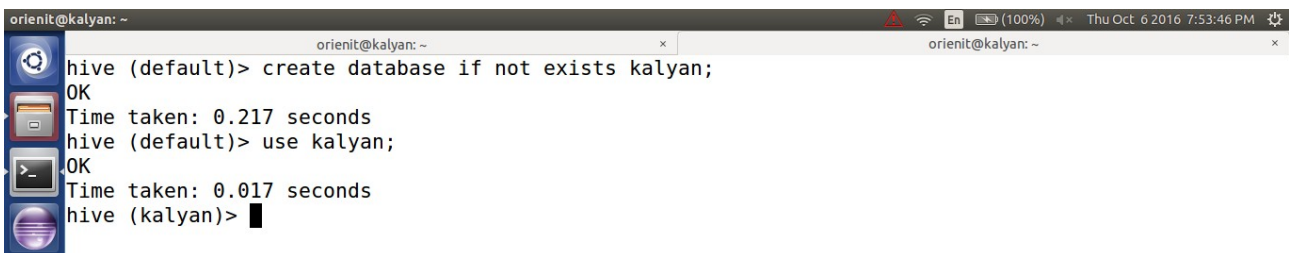
v. create a new database (**kalyan**) in hive using below command.

create database if not exists kalyan;



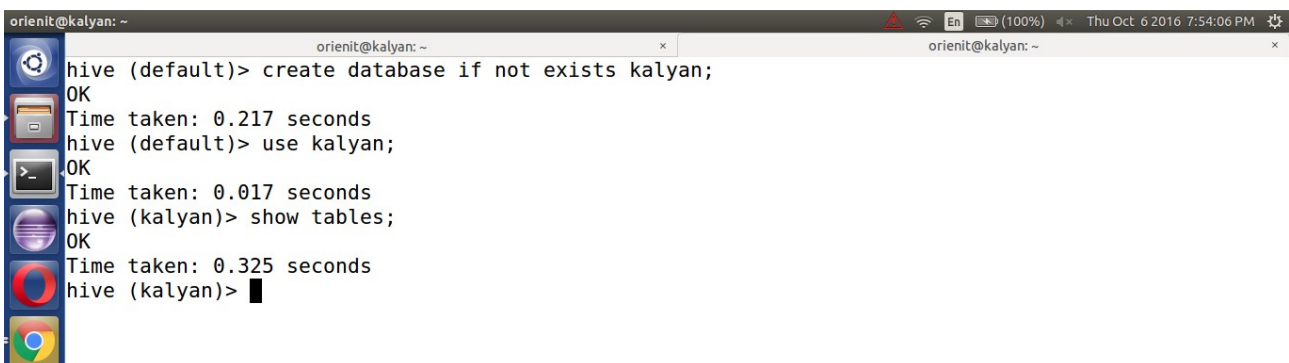
```
orientit@kalyan: ~
hive (default)> create database if not exists kalyan;
OK
Time taken: 0.217 seconds
hive (default)>
```

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



```
orientit@kalyan: ~
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.



```
orientit@kalyan: ~
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 '**users1**' table in kalyan database using below command.

```
CREATE TABLE IF NOT EXISTS kalyan.users1 (  
  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;
```



```

orienit@kalyan: ~
hive (kalyan)> CREATE TABLE IF NOT EXISTS users1 (
>     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: 0.77 seconds
hive (kalyan)>

```

ix. Display the data from 'users1' table using below command

select * from users1;

```

orienit@kalyan: ~
hive (kalyan)> CREATE TABLE IF NOT EXISTS users1 (
>     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: 0.77 seconds
hive (kalyan)> select * from users1;
OK
Time taken: 0.397 seconds
hive (kalyan)>

```

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

hive --service metastore

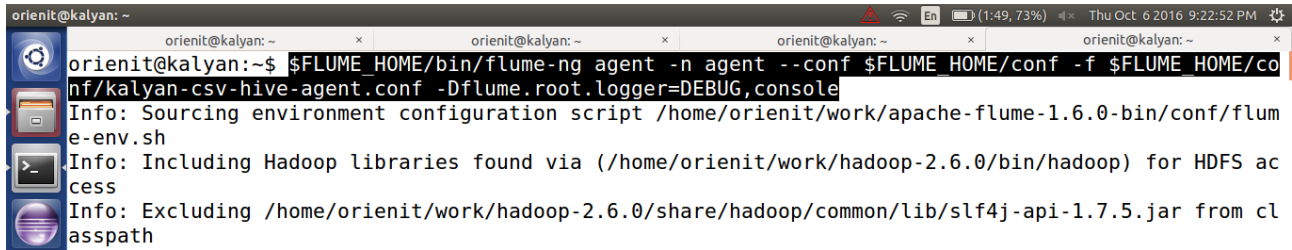
```

orienit@kalyan: ~
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]

```

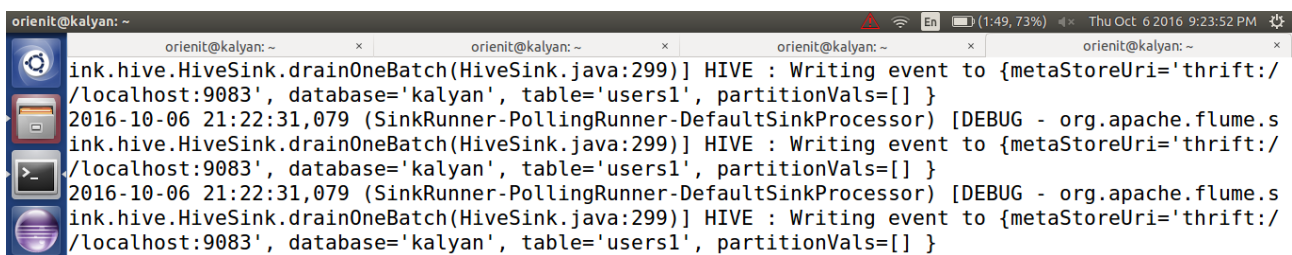
8. Execute the below command to `Extract data from CSV data into Hive using Flume`

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



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

9. Verify the data in console




```
ink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift://localhost:9083', database='kalyan', table='users1', partitionVals=[] }
2016-10-06 21:22:31,079 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - org.apache.flume.sink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift://localhost:9083', database='kalyan', table='users1', partitionVals=[] }
2016-10-06 21:22:31,079 (SinkRunner-PollingRunner-DefaultSinkProcessor) [DEBUG - org.apache.flume.sink.hive.HiveSink.drainOneBatch(HiveSink.java:299)] HIVE : Writing event to {metaStoreUri='thrift://localhost:9083', database='kalyan', table='users1', partitionVals=[] }
```

10. Verify the data in Hive

Execute below command to get the data from hive table 'users1'

```
select * from users1;
```



```
h1ve (kalyan)> select * from users1;
OK
91      user91  user91  user91@gmail.com      India  Karnataka  Ballari  2016-45-06  12:45:42
92      user92  user92  user92@gmail.com      US     New York   Hornell  2016-45-06  12:45:42
93      user93  user93  user93@gmail.com      India  Andhra Pradesh  Tirupati  2016-45-06
12:45:43
94      user94  user94  user94@gmail.com      US     New York   Hornell  2016-45-06  12:45:43
95      user95  user95  user95@gmail.com      US     Washington  Auburn  2016-45-06  12:45:43
96      user96  user96  user96@gmail.com      US     New York   Hornell  2016-45-06  12:45:43
97      user97  user97  user97@gmail.com      US     Florida Indian Creek  2016-45-06  12:45:43
98      user98  user98  user98@gmail.com      India  Karnataka  Bagalkot  2016-45-06
12:45:43
99      user99  user99  user99@gmail.com      US     New York   Canandaigua  2016-45-06
12:45:43
100     user100  user100  user100@gmail.com     US     New York   Little Falls  2016-45-06
12:45:43
Time taken: 0.34 seconds, Fetched: 10 row(s)
h1ve (kalyan)>
```