Use Cases :

1. publish message to kafka topic and subsdcribe the same message from the topic
2. read message from a kafka filestream and populate kafka topic
3. read from a kafka topic and update a table

\* yum install community-mysql\*

\* service mysqld restart

confluent kafka setup:( linux )

\* wget http://packages.confluent.io/archive/4.1/confluent-oss-4.1.0-2.11.tar.gz

\* tar -xzvf confluent-oss-4.1.0-2.11.tar.gz

\* cd /opt/confluent-4.1.0/bin

start zookeeper server:

\* cd /opt/confluent-4.1.0/bin

\* ./zookeeper-server-start ../etc/kafka/zookeeper.properties

\* ( if port 0.0.0.0/0.0.0.0:2181 is seen in output press ctrl-c and run below)

./zookeeper-server-start -daemon ../etc/kafka/zookeeper.properties

run netstat command to ensure zookeeper running in 2181

netstat -anp|grep 2181

tcp6 0 0 :::2181 :::\* LISTEN 15418/java

start kafka server:

\* ./kafka-server-start ../etc/kafka/server.properties

\* (if socket connections on 0.0.0.0:9092. is seen in output press ctrl-c and run below)

\* ./kafka-server-start -daemon ../etc/kafka/server.properties

\* 5c)run netstat command to ensure kafka running in 9092

\* netstat -anp|grep 9092

\* tcp6 0 0 :::9092 :::\* LISTEN 15656/java

6) create a topic:

./kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic poctopic

Created topic "poctopic".

7) start producer in a terminal

7a)cd /opt/confluent-4.1.0/bin

7b) ./kafka-console-producer --broker-list localhost:9092 --topic poctopic

1. start kafka consumer in another terminal

8a)cd /opt/confluent-4.1.0/bin

8b)[root@prathyusha bin]#./kafka-console-consumer --bootstrap-server localhost:9092 --topic poctopic --from-beginning

**Test kafka connect source**

the purpose of kafka connect source is to act as a publisher by reading data from a source ( could be file) and put it on a kafka topic ( its a type of kafka streaming)

kafka conenct source steps:

1) edit&save file /opt/confluent-4.1.0/etc/kafka/connect-file-source.properties

name=local-file-source

connector.class=FileStreamSource

tasks.max=1

file=/opt/confluent-4.1.0/test.txt

topic=poctopic

2 ) cd /opt/confluent-4.1.0/

edit test.txt and put some content there

3) cd /opt/confluent-4.1.0/bin

./connect-standalone ../etc/kafka/connect-standalone.properties ../etc/kafka/connect-file-source.properties

4) check on consumer it should apear with the content of test.txt in json format

5) again change the test.txt save and go back and check in consumer windo it should reaper with edited changes.

**test kafka connect sink**

step 1)export CLASSPATH=$CLASSPATH:.:/opt/confluent-4.1.0//share/java/kafka-connect-jdbc/mysql-connector-java.jar:/opt/confluent-4.1.0//share/java/kafka-connect-jdbc/kafka-connect-jdbc-4.1.0.jar

step 2)cd /opt/confluent-4.1.0/bin

step 3)./schema-registry-start ../etc/schema-registry/schema-registry.properties

step 4) open /opt/confluent-4.1.0//etc/schema-registry/connect-avro-standalone.properties and change rest port to rest.port=18083

step 5) creat a file at kafka-connect-jdbc/sink-mysql-jdbc.properties with bel content

name=test-sink

connector.class=io.confluent.connect.jdbc.JdbcSinkConnector

tasks.max=1

topics=poc

connection.url=jdbc:mysql://127.0.0.1:3306/pocTest?user=root

auto.create=true

step 6) open a putty terminal ,cd /opt/confluent-4.1.0/bin

step 7)./connect-standalone ../etc/schema-registry/connect-avro-standalone.properties ../etc/kafka-connect-jdbc/sink-mysql-jdbc.properties

( this ensures that consumer is runnign and making connectivity to mysql on database pocTest. it must hang.)

step 8)open a putty terminal ,cd /opt/confluent-4.1.0/bin

step 9)./kafka-avro-console-producer

--broker-list localhost:9092 --topic orders

--property value.schema='{"type":"record","name":"myrecord","fields":[{"name":"id","type":"int"},

{"name":"name", "type": "string"}, {"name":"city", "type": "int"}, {"name":"zip",

"type": "int"}]}'

this eill hang to get an input

{"id": 999, "product": "foo", "quantity": 100, "price": 50} ( press enter)

step 10) go to terminal which was opened by point 6) ( should see something like below without error)

[2018-04-27 11:26:05,158] INFO Checking table:poc exists for product:MySQL schema:null catalog: (io.confluent.connect.jdbc.sink.DbMetadataQueries:50)

[2018-04-27 11:26:05,163] INFO product:MySQL schema:null catalog:pocTest -- table:poc is absent (io.confluent.connect.jdbc.sink.DbMetadataQueries:60)

[2018-04-27 11:26:05,165] INFO Creating table:poc with SQL: CREATE TABLE `poc` (

`product` VARCHAR(256) NOT NULL,

`quantity` INT NOT NULL,

`price` FLOAT NOT NULL,

`id` INT NOT NULL) (io.confluent.connect.jdbc.sink.DbStructure:91)

[2018-04-27 11:26:05,509] INFO Querying column metadata for product:MySQL schema:null catalog:pocTest table:poc (io.confluent.connect.jdbc.sink.DbMetadataQueries:84)

[2018-04-27 11:26:05,573] INFO Updating cached metadata -- DbTable{name='poc', columns={product=DbTableColumn{name='product', isPrimaryKey=false, allowsNull=false, sqlType=12}, quantity=DbTableColumn{name='quantity', isPrimaryKey=false, allowsNull=false, sqlType=4}, price=DbTableColumn{name='price', isPrimaryKey=false, allowsNull=false, sqlType=7}, id=DbTableColumn{name='id', isPrimaryKey=false, allowsNull=false, sqlType=4}}} (io.confluent.connect.jdbc.sink.metadata.TableMetadataLoadingCache:49)

go to database terminal where you opened mysql and check row is create

mysql

use pocTest

mysql> show tables

-> ;

+-------------------+

| Tables\_in\_pocTest |

+-------------------+

| poc |

+-------------------+

1 row in set (0.00 sec)

> select \* from poc

-> ;

+---------+----------+-------+-----+

| product | quantity | price | id |

+---------+----------+-------+-----+

| foo | 100 | 50 | 999 |