# SPARK-STREAMING WITH CONFLUENT KAFKA TEST(LOCAL) (POC)

## METHOD:-1

## KAFKA set up:

* Download community kafka zip from confluent website.



* Unzip it and navigate to etc/kafka/server.properties
* Amend the listeners to point to localhost port



* Set CONFLUENT\_HOME and PATH



* Set up the confluent CLI and the path

curl -sL --http1.1 https://cnfl.io/cli | sh -s -- latest

export PATH=$(pwd)/bin:$PATH

* Set up OpenJDK 11 and javapath
* Then start the kafka local services by using the following command.

**confluent local services start**

* **It will spin up the below applications as per the screenshot below.**

**Text

Description automatically generated**

* **Test producer & Consumer using below commands from console**

**kafka-console-producer --topic PricingView --broker-list localhost:9092**

**kafka-console-consumer --topic PricingView --bootstrap-server localhost:9092 --from-beginning**

* **Once test is done execute the below command to stop and destroy kafka instance**

confluent local services stop

confluent local destroy

**Note:- Once kafka local services are up and running, spark streaming has been used to transform and consume the messages from producer which has been written in python.**

**SPARK SETUP IN PYCHARM:**

* Set up pycharm projects folder for the spark streaming projects.
* Set up the python and spark interpreter by installing required packages.

Graphical user interface, text

Description automatically generated

* Download Spark from Apache spark website and unzip it. Import below mention libraries (py4j & pyspark zip to the project structure.

Graphical user interface, text, application

Description automatically generated

* Start the kafka application on local host as mentioned earlier.
* Create a Kafka producer .py program to connect to kafka producer running on localhost to produce json messages to kafka topic constantly.

Text

Description automatically generated

* Create a spark-streaming .py program to connect to the kafka topic to consume and transform the streaming messages produced to the topic. Text

  Description automatically generated
* Below screenshot showing aggregation of the streaming messages which has been printed to the console.

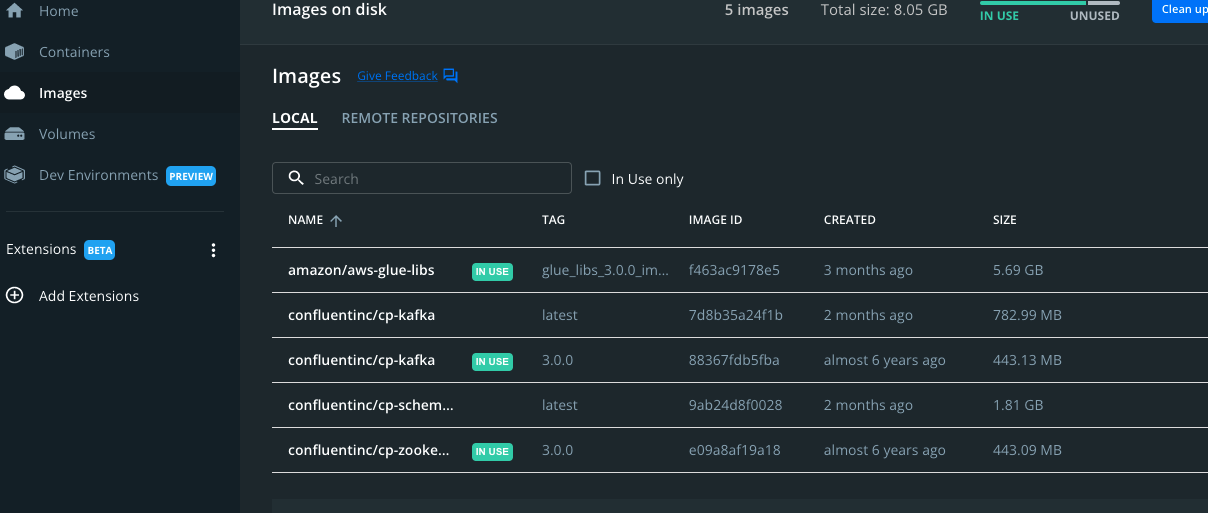
Text

Description automatically generated

**METHOD:-2**

**Using docker container:**

* Download docker desktop.
* And then download aws-glue-lib, cp-kafka and cp-zookeeper docker images from docker hub.

****

* Use Docker pull command to pull the images mentioned above.
* Then run zookeeper, kafka and gluelib using docker run command which will create respective containers.

docker run -itd -p 8888:8888 -p 4040:4040 -e DISABLE\_SSL="true" -v ~/.aws:/root/.aws:ro --name glue\_jupyter amazon/aws-glue-libs:glue\_libs\_3.0.0\_image\_01 /home/glue\_user/jupyter/jupyter\_start.sh

docker run -d \

--net=host \

--name=zookeeper \

-e ZOOKEEPER\_CLIENT\_PORT=32181 \

-e ZOOKEEPER\_TICK\_TIME=2000 \

confluentinc/cp-zookeeper:3.0.0

docker run -d \

--net=host \

--name=kafka \

-e KAFKA\_ZOOKEEPER\_CONNECT=localhost:32181 \

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://localhost:29092 \

confluentinc/cp-kafka:3.0.0

**Test that the broker is working fine**

**i. Create a topic**

docker run --net=host --rm confluentinc/cp-kafka:3.0.0 kafka-topics --create --topic testtopic --partitions 1 --replication-factor 1 --if-not-exists --zookeeper localhost:32181

**You should see**

Created topic "foo".

**ii. Verify that the topic is created successfully**

docker run --net=host --rm confluentinc/cp-kafka:3.0.0 kafka-topics --describe --topic testtopic --zookeeper localhost:32181

**You should see**

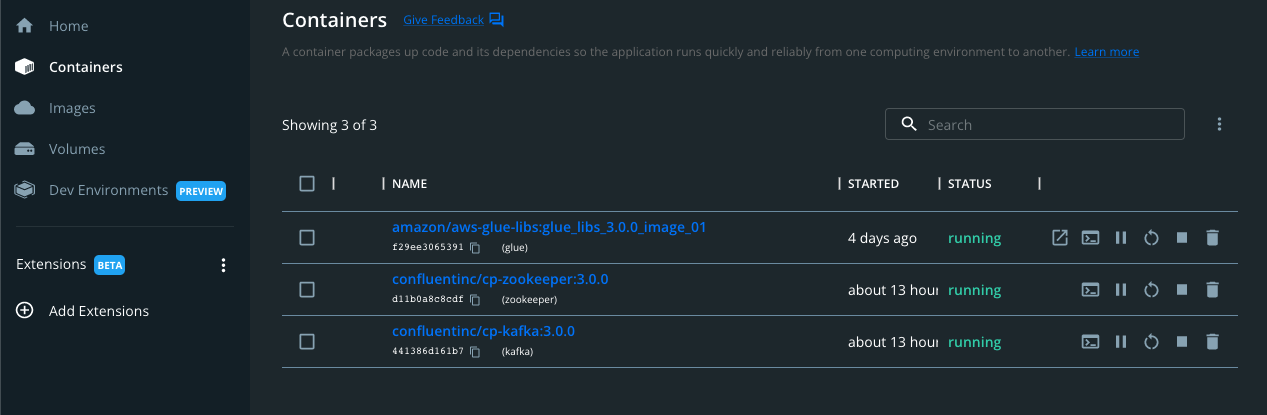
Topic: testtopic PartitionCount:1 ReplicationFactor:1 Configs:

Topic: foo Partition: 0 Leader: 1001 Replicas: 1001 Isr: 1001

docker run --net=host --rm confluentinc/cp-kafka:3.0.0 bash -c "seq 42 | kafka-console-producer --broker-list localhost:29092 --topic testtopic && echo 'Produced 42 messages.'"

docker run --net=host --rm confluentinc/cp-kafka:3.0.0 kafka-console-consumer --bootstrap-server localhost:29092 --topic testtopic --new-consumer --from-beginning --max-messages 42

* It will bring the below containers as per the screenshot

****

* **Now configure the containers in remote connection in VS code**

**Text, application, chat or text message

Description automatically generated**

* **Then open the Glue container repo inside VScode by attaching to it.**
* **Write Glue streaming .py code and producer.py code to write messages to kafka topic and then process an store it in S3 using glue-spark streaming libraries.**

**Text

Description automatically generated**

**Text

Description automatically generated**