

1.	Prerequisites:	2
2.	Start Kafka	3
3.	Basic Kafka Operations - CLI (Topic) – 30 Mins	4
4.	Basic Kafka Operations - CLI (Producer & Consumer) – 30 Mins	10
5.	Understanding Streaming Processing – 30 Minutes.	17
6.	Understand KSQL -30 Minutes	21

Hardware:

8 GB RAM , 30 GB HDD , Centos 7 or above OS. Access to internet.

Software Inventory:

- Zookeeper Version: apache-zookeeper-3.8.0-bin.tar
- Apache kafka : 2.13-3.2.1
- JDK 11.0.16
- Eclipse for Linux. (Any Latest version for JEE Development)
- Status : Color is Verified

Last Updated: June 29 - 2024.

1. Prerequisites:

Using docker s:

Start docker container : kafkao and verify it

Open a terminal.

#docker ps

```
(base) Henrys-MacBook-Air:~ henrypotsangbam$ docker ps
```

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS
0945477d457b	ehenry0227/learning:spark-kafka-raw	"/bin/bash"	34 seconds ago	Up 33 secon
ds	0.0.0.0:6062->6062/tcp, 0.0.0.0:8081->8081/tcp	kafka0		

```
(base) Henrys-MacBook-Air:~ henrypotsangbam$
```

Software:

- [kafka 2.13-3.2.1.tgz](#)
- jdk 11.0.1
- apache-zookeeper-3.6.3-bin.tar.gz
- confluent-7.2.2.tar

2. Start Kafka

Start kafka node

The following scripts will start a zookeeper along with a broker.

```
#sh /opt/scripts/startABroker.sh
```

Lab Installation completes End here.

3. Basic Kafka Operations - CLI (Topic) – 30 Mins

In this lab you will be able to create a topic and perform some operations to understand the information about topic like partition and replication.

You need to start the broker using startABroker.sh. The script should be in /opt/scripts folder

```
#sh startABroker.sh
```

```
#jps
```

```
[root@tos scripts]# sh startABroker.sh
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/./conf/zoo.cfg
Starting zookeeper ... STARTED
Started Successfully
[root@tos scripts]# jps
11665 Jps
11646 Kafka
11375 QuorumPeerMain
[root@tos scripts]#
```

Once the Kafka broker is started, we can verify that it is working by performing some simple operations against the broker; creating a test topic etc.

Create and verify details about topic:

```
# /opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --
replication-factor 1 --partitions 1 --topic test
```

5 Kafka – Administration

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --replication-factor 1 --partitions 1 --topic test
Created topic test.
```

```
# /opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --
replication-factor 1 --partitions 12 --topic IBM
```

```
# /opt/kafka/bin/kafka-topics.sh --bootstrap-server kafka0:9092 --describe --topic
test
```

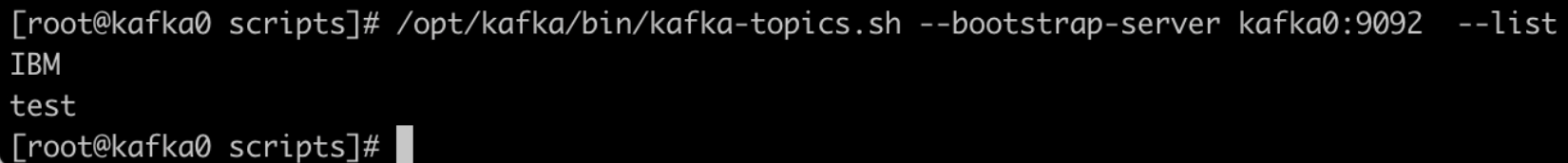
```
#!/opt/kafka/bin/kafka-topics.sh --bootstrap-server kafka0:9092 --describe --topic
IBM
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --replication-factor 1 --partitions 12 --topic IBM
Created topic IBM.
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --bootstrap-server kafka0:9092 --describe --topic test
Topic: test      TopicId: 9Rreu1aUR6uHfz3GHnYhtA PartitionCount: 1      ReplicationFactor: 1      Configs: segment.bytes=1073741824
  Topic: test      Partition: 0      Leader: 0      Replicas: 0      Isr: 0
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --bootstrap-server kafka0:9092 --describe --topic IBM
Topic: IBM       TopicId: eY3Tl3N_T6iYY_ZzV70Pjg PartitionCount: 12      ReplicationFactor: 1      Configs: segment.bytes=1073741824
  Topic: IBM       Partition: 0      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 1      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 2      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 3      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 4      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 5      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 6      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 7      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 8      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 9      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 10     Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 11     Leader: 0      Replicas: 0      Isr: 0
[root@kafka0 scripts]#
```

Verify the no of partition in the output above.

list and describe topic.

```
#/opt/kafka/bin/kafka-topics.sh --bootstrap-server kafka0:9092 --list
```



```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --bootstrap-server kafka0:9092 --list
IBM
test
[root@kafka0 scripts]#
```

List and describe Topics

What does the tool do?

This tool lists the information for a given list of topics. If no topics are provided in the command line, the tool queries zookeeper to get all the topics and lists the information for them. The fields that the tool displays are - topic name, partition, leader, replicas, isr.

How to use the tool?

List only single topic named "test" (prints only topic name)

```
#/opt/kafka/bin/kafka-topics.sh --list --bootstrap-server kafka0:9092 --topic test
```

List all topics (prints only topic names)

```
#/opt/kafka/bin/kafka-topics.sh --list --bootstrap-server kafka0:9092
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --list --bootstrap-server kafka0:9092 --topic test
test
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --list --bootstrap-server kafka0:9092
IBM
test
[root@kafka0 scripts]#
```

Describe only single topic named "test" (prints details about the topic)

```
#/opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server kafka0:9092 --topic test
```

Describe all topics (prints details about the topics)

```
#/opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server kafka0:9092
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server kafka0:9092 --topic test
Topic: test      TopicId: 9Rreu1aUR6uHfz3GHnYhtA PartitionCount: 1      ReplicationFactor: 1      Configs: segment.bytes=1073741824
  Topic: test      Partition: 0      Leader: 0      Replicas: 0      Isr: 0
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server kafka0:9092
Topic: test      TopicId: 9Rreu1aUR6uHfz3GHnYhtA PartitionCount: 1      ReplicationFactor: 1      Configs: segment.bytes=1073741824
  Topic: test      Partition: 0      Leader: 0      Replicas: 0      Isr: 0
Topic: IBM       TopicId: eY3Tl3N_T6iYY_ZzV70Pjg PartitionCount: 12      ReplicationFactor: 1      Configs: segment.bytes=1073741824
  Topic: IBM       Partition: 0      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 1      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 2      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 3      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 4      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 5      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 6      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 7      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 8      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 9      Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 10     Leader: 0      Replicas: 0      Isr: 0
  Topic: IBM       Partition: 11     Leader: 0      Replicas: 0      Isr: 0
[root@kafka0 scripts]#
```

We will understand the output in details later.

Create Topics

What does the tool do?

By default, Kafka auto creates topic if "auto.create.topics.enable" is set to true on the server. This creates a topic with a default number of partitions, replication factor and uses Kafka's default scheme to do replica assignment. Sometimes, it may be required that we would like to customize a topic while creating it. This tool helps to create a topic and also specify the number of partitions, replication factor and replica assignment list for the topic.

How to use the tool?

create topic with default settings

```
#/opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --topic
topic1 --partitions 2 --replication-factor 1
```

```
[root@tos scripts]# /opt/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic to
pic1 --partitions 2 --replication-factor 1
Created topic "topic1".
[root@tos scripts]#
```

Create a topic with replication 2.

```
#/opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --topic
topic2 --partitions 2 --replication-factor 2
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --topic topic1 --partitions 2 --replication-factor 1
Created topic topic1.
[root@kafka0 scripts]# /opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --topic topic2 --partitions 2 --replication-factor 2
Error while executing topic command : Replication factor: 2 larger than available brokers: 1.
[2023-07-03 15:53:18,020] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorException: Replication factor: 2 larger than available brokers: 1.
(kafka.admin.TopicCommand$)
[root@kafka0 scripts]#
```

As shown above, it generates an error. Since there is only a single broker. It will fix later.

-----Lab CLI completes End here-----

4. Basic Kafka Operations - CLI (Producer & Consumer) – 30 Mins

In this lab we will send message to the broker and consumer message using the kafka inbuilt commands.

You need to complete the previous lab before proceeding ahead.

You need to start the broker using startABroker.sh if not done earlier. The script should be in /opt/scripts folder

```
#sh startABroker.sh
```

```
#jps
```

```
[root@tos scripts]# sh startABroker.sh
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
Started Successfully
[root@tos scripts]# jps
11665 Jps
11646 Kafka
11375 QuorumPeerMain
[root@tos scripts]#
```

Sent message to **test** topic: Open a console to send message to the topic, test. Enter some text as shown below.

```
# /opt/kafka/bin/kafka-console-producer.sh --broker-list kafka0:9092 --topic test
```

Test Message 1

Test Message 2

^D

```
#
```

```
[root@kafka0 scripts]#
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-console-producer.sh --broker-list kafka0:9092 --topic test
```

```
>Test Message 1
```

```
>Test Message 2
```

```
>
```

Consume messages from a test topic: As soon as you enter the following script in a separate terminal, you should be able to consume the messages that we have type in the producer console.

```
# /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka0:9092 --topic test --from-beginning
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka0:9092 --topic test --from-beginning
```

```
Test Message 1
```

```
Test Message 2
```

DumplogSegment

You can also view the message publish to a topic from a log file.

```
/opt/kafka/bin/kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files \
/opt/data/kafka-logs/test-0/00000000000000000000.log | head -n 12
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files /opt/data/kafka-logs/test-0/00000000000000000000.log | head -n 12
Dumping /opt/data/kafka-logs/test-0/00000000000000000000.log
Starting offset: 0
baseOffset: 0 lastOffset: 0 count: 1 baseSequence: 0 lastSequence: 0 producerId: 0 producerEpoch: 0 partitionLeaderEpoch: 0 isTransactional: false isControl: false deleteHorizonMs: OptionalLong.empty position: 0 CreateTime: 1688379972311 size: 82 magic: 2 compresscodec: none crc: 1391971847 invalid: true
| offset: 0 CreateTime: 1688379972311 keySize: -1 valueSize: 14 sequence: 0 headerKeys: [] payload: Test Message 1
baseOffset: 1 lastOffset: 1 count: 1 baseSequence: 1 lastSequence: 1 producerId: 0 producerEpoch: 0 partitionLeaderEpoch: 0 isTransactional: false isControl: false deleteHorizonMs: OptionalLong.empty position: 82 CreateTime: 1688379975484 size: 82 magic: 2 compresscodec: none crc: 2251353974 invalid: true
| offset: 1 CreateTime: 1688379975484 keySize: -1 valueSize: 14 sequence: 1 headerKeys: [] payload: Test Message 2
[root@kafka0 scripts]#
```

Determine the offset of the messages.
CLI to list offset number.

```
#/opt/kafka/bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list
kafka0:9092 --topic test
```

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list kafka0:9092 --topic test
test:0:2
[root@kafka0 scripts]#
```

Partition 0 offset 2.

Sending Message with Key and Value

```
#/opt/kafka/bin/kafka-console-producer.sh --broker-list kafka0:9092 --topic IBM --  
property "parse.key=true" --property "key.separator="
```

Enter

WAS:600

DB2:700

ESB:900

```
[root@kafka0 scripts]# /opt/kafka/bin/kafka-console-producer.sh --broker-list kafka0:9092 --topic IBM --property "parse.key=true" --property "key.separator="
">WAS:600
>DB2:700
>ESB:900
>
```

```
#/opt/kafka/bin/kafka-console-consumer.sh --topic IBM --bootstrap-server  
kafka0:9092 --from-beginning \  
--property print.key=true \  
--property key.separator=":"
```

```
[root@kafka0 /]# /opt/kafka/bin/kafka-console-consumer.sh --topic IBM --bootstrap-server kafka0:9092 --from-beginning \  
> --property print.key=true \  
> --property key.separator=":"  
WAS:600  
ESB:900  
DB2:700
```

Let us understand consumer group load balancing concepts.

- Create a topic with 2 partitions – **city_info**.
- Initiate 2 consumers with same consumer Group
- Send 6 messages to the topic
- Verify load get distributed among the 2 consumers.

```
#/opt/kafka/bin/kafka-topics.sh --create --bootstrap-server kafka0:9092 --topic  
city_info --partitions 2 --replication-factor 1
```

Initiate 2 consumers with same consumer Group
Open in a separate 2 terminals.


```
#/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka0:9092 --topic  
city_info --consumer-property group.id=citygroup
```

```
#/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka0:9092 --topic  
city_info --consumer-property group.id=citygroup
```

- Produce about 6 messages to the topic

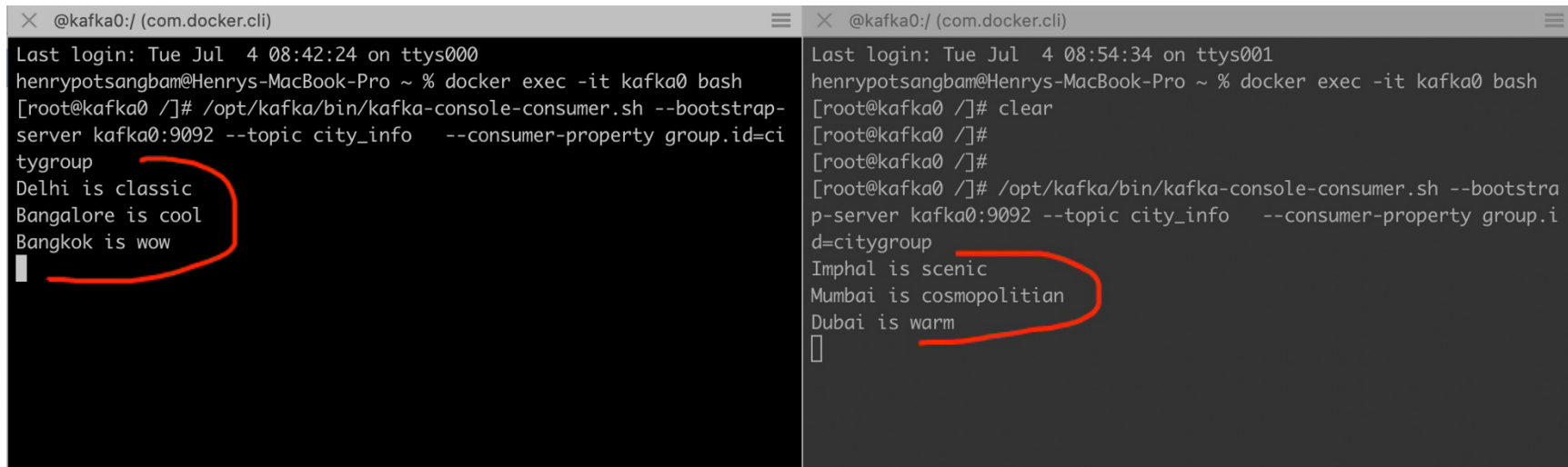
`#/opt/kafka/bin/kafka-console-producer.sh --broker-list kafka0:9092 --topic city_info`

```
created topic city_info.  
[root@kafka0 scripts]# /opt/kafka/bin/kafka-console-producer.sh --broker-list kafka0:9092 --topic city_info  
>Imphal is scenic  
>Delhi is classic  
>Mumbai is cosmopolitan  
>Bangalore is cool  
>Dubai is warm  
>Bangkok is wow  
>
```



- load get distributed among the 2 consumers.

16 Kafka – Administration



```
@kafka0:/ (com.docker.cli)
Last login: Tue Jul  4 08:42:24 on ttys000
henrypotsangbam@Henrys-MacBook-Pro ~ % docker exec -it kafka0 bash
[root@kafka0 /]# /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka0:9092 --topic city_info --consumer-property group.id=citygroup
Delhi is classic
Bangalore is cool
Bangkok is wow
█

@kafka0:/ (com.docker.cli)
Last login: Tue Jul  4 08:54:34 on ttys001
henrypotsangbam@Henrys-MacBook-Pro ~ % docker exec -it kafka0 bash
[root@kafka0 /]# clear
[root@kafka0 /]#
[root@kafka0 /]#
[root@kafka0 /]# /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka0:9092 --topic city_info --consumer-property group.id=citygroup
Imphal is scenic
Mumbai is cosmopolitan
Dubai is warm
█
```

-----Lab CLI ends here-----

5. Understanding Streaming Processing – 30 Minutes.

In this lab, following actions will be performed.

- Kstream creation from a Topic
- Perform transformation using flatMapValues function
- Perform aggregation method
- Conversion from stream to KTable.

Logic:

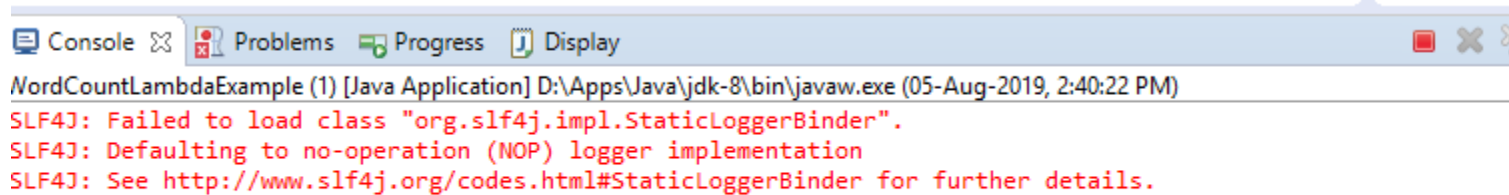
Stream from the input topic, streams-plaintext-input is created and perform transformation to it and the output is written to a topic, streams-wordcount-output.

Transformation logic:

Parse the text inserted in the input topic and count the occurrence of each word.
Calculate word count for the complete execution.

Execute program as shown below.

Start this example application in eclipse IDE. Double click on the eclipse icon in the Desktop.



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for Console, Problems, Progress, and Display. The console text reads: "WordCountLambdaExample (1) [Java Application] D:\Apps\Java\jdk-8\bin\javaw.exe (05-Aug-2019, 2:40:22 PM)". Below this, three lines of red text indicate an error: "SLF4J: Failed to load class 'org.slf4j.impl.StaticLoggerBinder'.", "SLF4J: Defaulting to no-operation (NOP) logger implementation", and "SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details."

Start the console producer. You can then enter input data by writing some line of text, followed by ENTER:

```
* #
* # hello kafka streams<ENTER>
* # all streams lead to kafka<ENTER>
* # join kafka summit<ENTER>
* #
* # Every line you enter will become the value of a single Kafka message.
```

```
# sh kafka-console-producer.sh --broker-list kafka0:9092 --topic streams-plaintext-
input
```

```
(base) [root@tos ~]# kafka-console-producer --broker-list tos.hp.com:9092 --topic streams-plaintext-input1
>Hello
>finally its seems to be workinhg
>Hey
>is it working now
>Great It working
>
```

Inspect the resulting data in the output topic. * You should see output data similar to below. Please note that the exact output * sequence will depend on how fast you type the above sentences. If you type them

- * slowly, you are likely to get each count update, e.g., kafka 1, kafka 2, kafka 3.
- * If you type them quickly, you are likely to get fewer count updates, e.g., just kafka 3.
- * This is because the commit interval is set to 10 seconds. Anything typed within
- * that interval will be compacted in memory.

```
# sh kafka-console-consumer.sh --topic streams-wordcount-output --from-beginning \
    --bootstrap-server kafka0:9092 \
    --property print.key=true \
    --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer
```

```
(base) [root@tos logs]# kafka-console-consumer --topic streams-wordcount-output1
--from-beginning \
> --bootstrap-server tos.hp.com:9092 \
> --property print.key=true \
> --property value.deserializer=org.apache.kafka.co
mmmon.serialization.LongDeserializer
hello 1
finally 1
its 1
seems 1
to 1
be 1
workinhg 1
hey 1
is 1
it 1
working 1
now 1
great 1
it 2
working 2
```

Once you're done with your experiments, you can stop this example via `Ctrl-C`.

----- Lab Ends Here -----

6. Understand KSQL -30 Minutes

Start ksqlDB's server

ksqlDB is packaged with a startup script for development use. We'll use that here.

When you're ready to run it as a service, you'll want to manage ksqlDB with something like `systemd`.

```
#/opt/ksqldb/bin/ksql-server-start /opt/ksqldb/etc/ksqldb/ksql-server.properties
```

If you are able to successfully start the KSQL DB server, you should get the following output.

```
[2022-02-15 16:17:02,735] INFO ksqlDB API server listening on http://0.0.0.0:8088 (io.confluent.ksql.rest.server.KsqlRestApplication:405)
```

```
=          _      _   _    _   _  
= | | _____ -| | - \ |__ )  
= | | / /_/_/_\`' | | | | |_- \  
= | | < \ \ ( | | | | | |_) |  
= |_|\_\_/___/\_,_|_|____/|___/  
=           |_|_  
=  
=       The Database purpose-built  
=       for stream processing apps
```

Copyright 2017-2021 Confluent Inc.

```
Server 0.23.1 listening on http://0.0.0.0:8088
```

To access the KSQL CLI, run:

```
ksql http://0.0.0.0:8088
```

```
[2022-02-15 16:17:02,813] INFO Server up and running (io.confluent.ksql.rest.server.KsqlServerMain:92)
```

```
[2022-02-15 16:17:07,390] INFO Successfully submitted metrics to Confluent via secure endpoint (io.confluent.support.metrics.submitters.ConfluentSubmitter:146)
```

Start ksqlDB's interactive CLI

ksqlDB runs as a server which clients connect to in order to issue queries.

Run this command to connect to the ksqlDB server and enter an interactive command-line interface (CLI) session.

```
#/opt/ksqldb/bin/ksql http://0.0.0.0:8088
```

```
[root@kafka0 ksqldb]# /opt/ksqldb/bin/ksql http://0.0.0.0:8088
```

[illegible]

Copyright 2017-2021 Confluent Inc.

CLI v0.23.1, Server v0.23.1 located at <http://0.0.0.0:8088>

Server Status: RUNNING

Having trouble? Type 'help' (case-insensitive) for a rundown of how things work!

ksql>

```
#show topics;
```

```
ksql> show topics;
```

Kafka Topic	Partitions	Partition Replicas
default_ksql_processing_log	1	1
test	1	1
topic1	2	1

The topics listed above depends on the number of topics in your kafka system.

Execute the following commands in the above CLI.

```
CREATE STREAM riderLocations (profileId VARCHAR, latitude DOUBLE, longitude DOUBLE)
  WITH (kafka_topic='locations', value_format='json', partitions=1);
```

Insert 3 messages

```
INSERT INTO riderLocations (profileId, latitude, longitude) VALUES ('c2309eec', 37.7877, -122.4205);
INSERT INTO riderLocations (profileId, latitude, longitude) VALUES ('18f4ea86', 37.3903, -122.0643);
INSERT INTO riderLocations (profileId, latitude, longitude) VALUES ('4ab5cbad', 37.3952, -122.0813);
INSERT INTO riderLocations (profileId, latitude, longitude) VALUES ('8b6eae59', 37.3944, -122.0813);
INSERT INTO riderLocations (profileId, latitude, longitude) VALUES ('4a7c7b41', 37.4049, -122.0822);
INSERT INTO riderLocations (profileId, latitude, longitude) VALUES ('4ddad000', 37.7857, -122.4011);
```


Fetch all the records.

```
select * from riderLocations
```

----- Lab Ends Here -----