

# LAB Stream 1-2 Custom Word Count Example

Welcome to the session Stream 1-2 lab. The work for this lab is done in `~/kafka-training/stream-lab2`. In this lab, you are going to run a custom Word Count Example. You must write the java code. You create two new Kafka topics, one called `word-count-input` and the other `word-count-output`. You will then add some data to the input topic, start a consumer on the output topic so you can watch it, and then run the Word Count Demo.

## Create Kafka Topics

- Create a topic named `word-count-input` with 1 partition and a replication factor of 1.
- Create a topic named `word-count-output` with 1 partition and a replication factor of 1.

## ACTION - REVIEW `create-topics.sh`

`~/kafka-training/stream-lab2/bin/create-topics.sh`

```
#!/usr/bin/env bash
cd ~/kafka-training

## Create input topic
kafka/bin/kafka-topics.sh --create \
  --replication-factor 1 \
  --partitions 1 \
  --topic word-count-input \
  --zookeeper localhost:2181

## Create output topic
kafka/bin/kafka-topics.sh --create \
  --replication-factor 1 \
  --partitions 1 \
  --topic word-count-output \
  --zookeeper localhost:2181

## List created topics
kafka/bin/kafka-topics.sh --list \
  --zookeeper localhost:2181
```

## ACTION - START ZooKeeper and Kafka Broker if needed.

Only 1 broker is necessary.

## ACTION - RUN `create-topics.sh` as follows:

```
$ cd ~/kafka-training/stream-lab2/bin
$ ./create-topics.sh
Created topic "word-count-input".
Created topic "word-count-output".
word-count-input
word-count-output
```

## Add data to the topic

Now add data to the input topic by starting a console producer and entering data.

### **ACTION - REVIEW** `start-producer-console-input.sh`

`~/kafka-training/stream-lab2/bin/start-producer-console-input.sh`

```
#!/usr/bin/env bash
cd ~/kafka-training

## Producer
kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic streams-plaintext-input
```

### **ACTION - RUN** `start-producer-console-input.sh`

```
$ cd ~/kafka-training/stream-lab2
$ ./start-producer-console-input.sh
>
```

### **ACTION - Enter Data**

Enter the following 3 lines of data. Ctrl-C to stop the console.

```
> stock streams MSFT
> stock data AAPL
> stock streams RHT
```

### **ACTION - Validate Data**

Run a console consumer against the input topic to verify the data is there.

`~/kafka-training/stream-lab2/start-consumer-console-input.sh`

```
#!/usr/bin/env bash
cd ~/kafka-training

## Input Consumer
kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic streams-plaintext-input --from-beginning
```

### **ACTION - RUN** `start-consumer-console-input.sh`

```
$ cd ~/kafka-training/stream-lab2/bin
$ ./start-consumer-console-input.sh
> stock streams MSFT
> stock data AAPL
> stock streams RHT
```

```
<Ctrl-C>
Processed a total of 3 messages
```

Above, stops the consumer.

## Run a console consumer against the output.

Run another consumer against the output topic so we can see the work that the Word Count Demo does. Notice a few things about this consumer. These match the way the topic is populated by the demo.

- It has a message formatter specified.
- It has key and value deserializers.

**ACTION - EDIT** `~/kafka-training/stream-lab2/bin/start-consumer-console-output.sh` , follow instructions in file.

`~/kafka-training/stream-lab2/start-consumer-console-output.sh`

```
#!/usr/bin/env bash
cd ~/kafka-training

## Output Consumer
kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 \
  --topic word-count-output \
  --from-beginning \
  --formatter kafka.tools.DefaultMessageFormatter \
  --property print.key=true \
  --property print.value=true \
  --property
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer \
  --property
value.deserializer=org.apache.kafka.common.serialization.LongDeserializer
```

**ACTION - RUN** `start-consumer-console-output.sh`

```
$ cd ~/kafka-training/stream-lab2/bin
$ ./start-consumer-console-output.sh
```

There will be no output until the demo code runs. Keep this window open so you can see it as the demo runs.

## Create and Run the Custom Word Count

**ACTION - EDIT** `~/kafka-training/stream-lab2/WordCount.java` , follow instructions in file.

**ACTION - RUN** `WordCount`

Run the java main.

**Consumer Console Output after running run-word-count-demo.sh**

```
stock 1
streams 1
msft 1
stock 2
data 1
aapl 1
stock 3
streams 2
rht 1
```

## Conclusion Custom Word Count example

The word count is the simplest example of stream processing of an input topic and writing results to a different then you created Kafka Producer in Java that uses the Kafka replicated topic to send records. You sent records with the Kafka Producer using async and sync send methods.

## Review Custom Word Count example

### How could you change the code to not be case-insensitive?

Remove the `.toLowerCase()` from step 2

### How would you simplify the code by using the builder syntax?

Chain calls together

## Related content

- [What is Kafka?](#)
- [Kafka Architecture](#)
- [Kafka Topic Architecture](#)
- [Kafka Consumer Architecture](#)
- [Kafka Producer Architecture](#)
- [Kafka Architecture and low level design](#)
- [Kafka and Schema Registry](#)
- [Kafka and Avro](#)
- [Kafka Ecosystem](#)
- [Kafka vs. JMS](#)
- [Kafka versus Kinesis](#)
- [Kafka Tutorial: Using Kafka from the command line](#)
- [Kafka Tutorial: Kafka Broker Failover and Consumer Failover](#)
- [Kafka Tutorial](#)
- [Kafka Tutorial: Writing a Kafka Producer example in Java](#)
- [Kafka Tutorial: Writing a Kafka Consumer example in Java](#)
- [Kafka Architecture: Log Compaction](#)

## About Cloudurable

We hope you enjoyed this article. Please provide [feedback](#). Cloudurable provides [Kafka training](#), [Kafka consulting](#), [Kafka support](#) and helps [setting up Kafka clusters in AWS](#).