--- Prelimanary steps: install Kafka and Zookeeeper and run them

a) Download and install Kafka, see
https://kafka.apache.org/downloads
and https://kafka.apache.org/quickstart
Zookeeper comes included with Kafka, we will use Zookeeper rather than KRaft.

You can set an environment variable, e.g.,
$KAFKA_HOME = /usr/local/etc/kafka

If needed, configure Kafka properties in
$KAFKA_HOME/server.properties (e.g., set listeners and advertised.listeners)

By default, clients connect to Zookeeper on port 2181 and to Kafka on port 9092.

Let's start Zookeeper and Kafka.

b) start Zookeeper
```
$ zookeeper-server-start $KAFKA_HOME/zookeeper.properties
```

alternatively,
```
$ zkServer start
```

c) start Kafka
```
$ kafka-server-start $KAFKA_HOME/server.properties
```

Once the Kafka server has successfully launched, you will have a basic Kafka environment running and ready to use.

Let's use Kafka CLI tools to create a topic, publish and consume some messages to/from topic and delete it.

1) Create a topic, which is named test, has 3 partitions and is non-replicated (only one broker per partition)
```
$ kafka-topics --create --topic test \
--bootstrap-server localhost:9092 \
--replication-factor 1 --partitions 3
```

You cannot specify a replication factor greater than the number of available brokers.

You can get details about the topic just created:
```
$ kafka-topics --describe --topic test \
--bootstrap-server localhost:9092
```

A Kafka client communicates with the Kafka broker via the network for writing or reading messages.
The broker will store the received messages in a durable and fault-tolerant manner.

Run the console producer client to write a few messages into your

topic.
By default, each line you enter will result in a separate message being written to the topic.

2) Write some messages into your topic

```
$ kafka-console-producer --bootstrap-server localhost:9092 \
--topic test
> <insert here your message>
```

You can stop the producer client with Ctrl-C.

You can run multiple producers from different terminal windows, e.g.,

```
$ kafka-console-producer --bootstrap-server localhost:9092 \
--topic test
> msg 1
> msg 2

$ kafka-console-producer --bootstrap-server localhost:9092 \
--topic test
> msg A
> msg B
```

3) Read messages from your topic
To read all historical messages and future ones from the topic (all topic partitions), use the --from-beginning option

```
$ kafka-console-consumer --bootstrap-server localhost:9092 \
--topic test --from-beginning
```

If you do not specify --from-beginning, the consumer will read only future messages.

You can stop the consumer client with Ctrl-C.

To read messages from a given offset (e.g., 2) and a specific topic partition (e.g., 1)

```
$ kafka-console-consumer --bootstrap-server localhost:9092 \
--topic test --offset 2 --partition 1
```

Offset number starts from 0 and partition number starts from 0.

Messages by default will not display metadata information (e.g., partition or key).

You can run multiple consumers from different terminal windows.

4) Produce messages with key
By default messages sent to a Kafka topic will result in messages with null keys.
To specify the key, use the properties parse.key and key.separator in the producer.
In the following example, the separator between the key and the value is :

```
$ kafka-console-producer --bootstrap-server localhost:9092 \
--topic test --property parse.key=true --property key.separator=:
>course:sdcc
>university: Tor Vergata
```

5) Consume messages with key
By default, the console consumer shows only the value of the Kafka message. To show also the key, use the formatter kafka.tools.DefaultMessageFormatter and the properties print.timestamp=true print.key=true print.value=true:

```
$ kafka-console-consumer --bootstrap-server localhost:9092 \
--topic test \
--formatter kafka.tools.DefaultMessageFormatter \
--property print.timestamp=true \
--property print.key=true --property print.value=true \
--from-beginning
```

6) Create a group of consumers.
You cannot have more consumers in a group than partitions in your Kafka topic.
Let's use the topic test which has 3 partitions.

Launch a consumer in a consumer group, named my-group.

```
$ kafka-console-consumer --bootstrap-server localhost:9092 \
--topic test --group my-group
```

Open two new terminal windows and launch a second and third consumer belonging to the same consumer group my-group.

Each consumer in the consumer group my-group will be assigned to a partition.
Produce some messages in the topic.

```
$ kafka-console-producer --bootstrap-server localhost:9092 \
--topic test
>first message
>second message
>third message
>fourth message
```

Each consumer will show only the messages produced on the partition that are assigned to it.

7) Some useful commands:

To list topics

```
$ kafka-topics --list --bootstrap-server localhost:9092
```

To delete a topic

```
$ kafka-topics --delete --bootstrap-server localhost:9092 \
--topic test
```

Topic deletion must be enabled (see the delete.topic.enable setting in Kafka configuration file server.properties), otherwise then the topics will be "marked for deletion" but will not be deleted.

You can delete a topic also using Zookeeper shell, after having stopped Kafka server (see below).

Go to Kafka logs directory (e.g., $ cd /usr/local/var/lib/kafka-logs) and delete the topic directory.
Then, connect to Zookeeper, list the topics, remove the topic entry from Zookeeper (e.g., test), and exit Zookeeper using Ctrl-C.

```
$ zookeeper-shell localhost:2181
ls /brokers/topics
deleteall /brokers/topics/test
```

Using the same procedure, you can also reset __consumer_offsets topic in Kafka with Zookeeper.
In this case, use from Zookeeper shell:

```
deleteall /brokers/topics/__consumer_offsets
```

You should never do that in production because there is a good chance to skip messages in consumers.

8) To tear down the Kafka environment:

Stop the producer and consumer clients with Ctrl-C.

Stop the Kafka broker using either Ctrl-C or

```
$ kafka-server-stop
```

Stop Zookeeper using either

```
$ zookeeper-server-stop
```

or

```
$ zkServer stop
```