--- Prelimanary steps: install Kafka and run it

a) Download and install Kafka
See
https://kafka.apache.org/downloads
https://kafka.apache.org/quickstart

We will use Kraft rather than Zookeeper.

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

If needed, configure Kafka properties in the file $KAFKA_HOME/
server.properties.

Key properties for KRaft mode:
- Ensure process.roles is set to broker and kafka:
process.roles=broker,kafka
Ensure kafka.metrics.reporters is set correctly for monitoring.
- KRaft mode uses listeners and advertised.listeners to configure
how Kafka communicates.
By default, Kafka in Kraft mode runs on port 9092. You can set the
listeners in the server.properties file:
listeners=PLAINTEXT://localhost:9092

b) Start Kafka with
$ kafka-server-start $KAFKA_HOME/config/server.properties

Once Kafka starts, you'll have a Kafka cluster with a single broker
and running in KRaft mode.

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 named test with 3 partitions and is a replication
factor of 1 (no replication, only one broker per partition)
```
$ kafka-topics --create --topic test \
  --bootstrap-server localhost:9092 \
  --replication-factor 1 --partitions 3
```

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

To get details about the topic:
```
$ kafka-topics \
  --bootstrap-server localhost:9092 \
  --describe \
  --topic test \
```

2) Produce messages to the topic
A Kafka producer communicates with the Kafka broker via the network
for writing messages.
The broker stores the received messages in a durable and fault-
tolerant manner.

Run the console producer to write a few messages into your topic.

```
$ kafka-console-producer \
  --bootstrap-server localhost:9092 \
  --topic test
```

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

```
Type your messages (each line is a separate message)
>msg 1
>msg 2
>msg 3
```

To stop the producer, press Ctrl-C.

You can open additional terminal windows and run multiple producers:

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

3) Consume messages from the topic

To read all historical messages and future ones from the topic (all topic partitions), use the --from-beginning flag

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

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

To stop the consumer, press Ctrl-C.

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

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

Note: offset numbering starts from 0 and partition numbering starts from 0.

You can run multiple consumers from different terminal windows.
If you run three producers and three consumers reading from different partitions, you will see that the messages produced are distributed among the partitions in a round-robin manner.

4) Produce messages with a key
By default, messages are sent with a null key.
To specify a 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=:
```

Then, type messages:
```
>course:sdcc
>university:Tor Vergata
```

5) Consume messages with a key
By default, the console consumer shows only the value of the Kafka message, not metadata such as the partition or key. To show both the key and value, use the formatter kafka.tools.DefaultMessageFormatter together with the properties to print timestamps, keys, and values (print.timestamp=true print.key=true print.value=true):

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

6) Use a consumer group.
Consumers within the same group share partitions. If there are more consumers than the number of partitions of a topic, then some consumers will remain inactive.

We will 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 more terminal windows and run two additional consumers in the same consumer group my-group.
Each consumer in the group will be assigned to different partitions.
Produce messages in the topic.

Run producers; each consumer will display the messages from its assigned partition.

7) Other useful commands:

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

Describe topics
$ kafka-topics --bootstrap-server localhost:9092 --describe

Check partition assignment among consumers in the same group
$ kafka-consumer-groups \
  --bootstrap-server localhost:9092 \
  --describe --group my-group

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

Note: topic deletion must be enabled (set delete.topic.enable=true in server.properties), otherwise the topics will be "marked for deletion" but will not be deleted.

8) Tear down the Kafka environment:

Stop producers and consumers with Ctrl-C.

Stop Kafka:
$ kafka-server-stop

To delete data from your local Kafka environment, including messages:
$ rm -rf /usr/local/var/lib/kraft-combined-logs
(the path may change)