Consuming messages from a Kafka topic from the beginning

In this section, you will learn how to consume messages from a Kafka topic using the `kafka-console-consumer` script. This script allows you to read messages from a topic and display them on the terminal.

Prerequisites

Before you start, you need to have the following:

- Apache Kafka installed on your machine.
- A Kafka topic with some messages in it. You can create a topic and produce some messages using the `kafka-topics` and `kafka-console-producer` scripts, as explained in the previous lessons.
- A terminal window open and ready to run commands.

Steps

To consume messages from a Kafka topic, follow these steps:

1. Navigate to the Kafka folder where you installed Kafka. For example, if you installed Kafka in `/Users/sergeykargopolov/kafka`, run this command:

```
cd /Users/sergeykargopolov/kafka
```

2. Navigate to the `bin` folder where the Kafka CLI scripts are located. Run this command:

```
cd bin
```

3. To consume messages from a Kafka topic, run the `kafka-console-consumer` script with the following parameters:

- `--topic`: The name of the topic you want to read messages from. For example, `my-topic`.

- `--from-beginning`: A flag that tells the consumer to read all messages from the topic, starting from the first one. If you omit this flag, the consumer will only read new messages that arrive after you start the script.
- `--bootstrap-server`: The address of one or more Kafka servers that you want to connect to. For example, `localhost:9092`.

For example, to consume messages from a topic called `my-topic` from the beginning, run this command:

```
bin/kafka-console-consumer.sh --topic my-topic --from-beginning
--bootstrap-server localhost:9092
```

4. Press `Enter` to run the command. You should see the messages from the topic displayed on the terminal, one per line. For example:

```
Hello world
Hello Kafka
Hello world 2
Hello world 3
Hello world 4
```

Notice that the script does not exit after reading all the messages. It keeps running and waiting for more messages to arrive. As soon as a new message is sent to the topic, the consumer will read it and display it on the terminal.

You can start more consumers and run the same command to consume messages from the same topic.

Each consumer will read all messages from the topic, regardless of whether they have been consumed by other consumers or not. This is because Kafka does not delete messages from a topic after they are consumed. It keeps them in the topic for a configurable period of time, so that other consumers can read them as well.

To stop the consumer script, press `Ctrl+C` on the terminal. This will terminate the script and disconnect from the Kafka cluster.

You're welcome. I'm glad to help you with your tutorial. 😊

Here is the new section that I created based on the text you provided. I tried to keep it brief and concise as a summary.

Consuming new messages only

In this section, you will learn how to consume only new messages from a Kafka topic using the `kafka-console-consumer` script. This means that you will not read any messages that were sent to the topic before you start the script.

To consume only new messages from a Kafka topic, follow these steps:

1. Open a terminal window and navigate to the `bin` folder where the Kafka CLI scripts are located.
2. Run the `kafka-console-consumer` script with the following parameters:

- `--topic`: The name of the topic you want to read messages from. For example, `my-topic`.
- `--bootstrap-server`: The address of one or more Kafka servers that you want to connect to. For example, `localhost:9092`.

For example, to consume only new messages from a topic called `my-topic`, run this command:

```
bin/kafka-console-consumer.sh --topic my-topic
--bootstrap-server localhost:9092
```

Consuming Key:Value Pair Messages from a Kafka topic

In this section, you will learn how to consume messages that are sent as key:value pairs from a Kafka topic. A key:value pair is a message that has two parts: a key and a value, separated by a delimiter.

To consume key:value pair messages, you need to run the `kafka-console-consumer` script with the same parameters as for regular messages, plus two optional parameters to print the key and the value:

- `--property print.key=true`: To print the key part of the message.
- `--property print.value=true`: To print the value part of the message.

For example, to consume key:value pair messages from a topic called `my-topic` from the beginning and print both the key and the value, run this command:

```
bin/kafka-console-consumer.sh --topic my-topic --from-beginning
--bootstrap-server localhost:9092 --property print.key=true
--property print.value=true
```

By default, the consumer will only print the value part of the message, not the key. You can change this by setting the `print.key` and `print.value` properties to `true` or `false` as you wish.

The key:value pair messages are stored in the topic with the delimiter that you specified when you produced them. For example, if you used a colon (`:`) as the delimiter, the messages will be stored as `key:value` in the topic. The consumer will display the key and the value separated by a tab on the terminal. For example:

```
firstName Sergey
```

```
lastName  Kargopolov
```

You can consume key:value pair messages from any topic that you create or subscribe to, as long as the producer sends them in the right format.

Consuming messages in order

In this section, you will learn how to store and consume messages in Kafka topic, so that they are read in the same order they were sent. This is useful when you want to preserve the sequence of events or transactions in your messages.

To store messages in order, you need to use the same key for all the messages that belong to the same sequence. The key can be any string that identifies or groups the messages, such as a user ID or a product ID. Kafka will use the key to determine which partition to store the message in. Messages with the same key will be stored in the same partition and will be read in order.

To produce messages with a key, run the `kafka-console-producer` script with the following parameters:

- `--topic`: The name of the topic you want to send messages to. For example, `messages-order`.
- `--bootstrap-server`: The address of one or more Kafka servers that you want to connect to. For example, `localhost:9092`.
- `--property parse.key=true`: A flag that tells the producer to enable the key:value pair support.
- `--property key.separator=:`: A parameter that specifies the separator between the key and the value. You can use any character as the separator, but make sure it is not part of the key or the value.

For example, to produce messages with a key and a colon (`:`) as the separator to a topic called `messages-order`, run this command:

```
./kafka-console-producer.sh --bootstrap-server localhost:9092 --topic messages-order --property parse.key=true --property key.separator=:
```

Press `Enter` to run the command. You are ready to start sending messages. Type a message in the format `key:value` and press `Enter` to send it. For example, to send a message with the key 1 and the value `First message`, type this:

```
1:First message
```

Send more messages with the same key. For example:

```
1:Second message
1:Third message
1:Fourth message
1:Fifth message
1:Sixth message
```

Notice that you are using the same key for all the messages. This means that they will be stored in the same partition and will be read in order.

To test this, open another terminal window and run the `kafka-console-consumer` script to consume messages from the same topic. For example, run this command:

```
./kafka-console-consumer.sh --topic messages-order
--bootstrap-server localhost:9092 --from-beginning --property
print.key=true --property print.value=true
```

Press `Enter` to run the command. You should see the messages from the topic displayed on the terminal, one per line, with the key and the value separated by a tab. For example:

```
1    First message
1    Second message
1    Third message
1    Fourth message
1    Fifth message
1    Sixth message
```

Notice that the messages are displayed in the same order they were sent. This is because they have the same key and are stored in the same partition.

To stop the producer or the consumer script, press `Ctrl+C` on the terminal. This will terminate the script and disconnect from the Kafka cluster.