Kafka Fundamentals Brain Teasers: M3: Fundamentals of the Fundamentals

Overview

Key concepts from Module 3 are:

- Producers
- Brokers
- Consumers
- ZooKeeper
- Decoupling of producers and consumers
- Topics
- Partitions and segments
- Logs
- Offset to write the next message produced to
- Consumer offsets
- Records
- Replication
- Partitioning strategies
- How consumers access messages
- Consumer groups

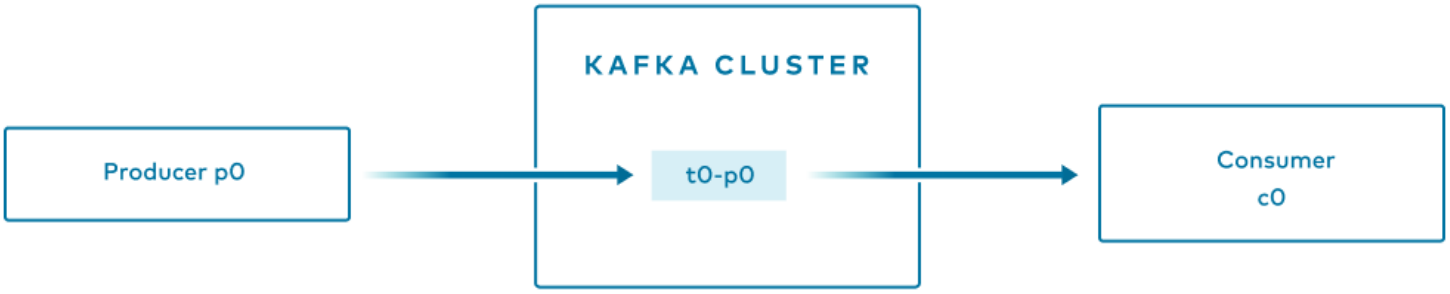
Here's the quick quiz on Module 3 (https://forms.gle/RDc84FbPeJ2CwCRP9) from the Online Talk Series.

Problem #3A: How Do Producers Connect to Consumers?

Suppose you have a Kafka cluster; one producer, p_0, producing to topic t_0, with one partition; and one consumer c_0, reading from topic t_0. c_0 read what p_0 produced. This implies p_0 and c_0 are related. What is the nature of this relationship? If you add another producer p_1 (that also produces to topic t_0), must you also add another consumer (e.g., c_1) to read messages from p_1?

Solution #3A: How Do Producers Connect to Consumers?

This is one of the most basic scenarios possible—you may be playing with Kafka for fun or to get started, but this setup is not recommended in production. We have a single topic with only one partition, along with a producer and a consumer. Because there is only one partition, all messages that the producer produces will be written to that partition, p_0. And because there is only a single consumer reading from t_0 (note that c_0 had to subscribe to topic t_0), that consumer will be assigned to read from the sole partition.



As a result, it seems that p_0 and c_0 are related. However, that is only due to the constraints of the problem (or, on a larger scale, by chance). Should we add another producer, p_1, which produces to the same topic, we do *not* need to add another consumer. The new producer p_1 would write to the sole partition that exists. Consumer c_0 is assigned to consume from the topic t_0. Thus:

- The sole partition would have messages that came from both producers mixed up within the partition
- Consumer c_0 would read messages from both producers in the order that they arrived at that partition

This brings us to the key takeaway—producers and consumers are decoupled. Producers write to logs; consumers read from logs. Consumers don't know which producers produced what messages they are reading; producers don't know which consumers (if any) will read what messages they produce. If you want to scale up production or consumption to improve performance of either, you can just add more producers or consumers and not worry about what's going on on the other end.

Sometimes, people think there is a direct relationship between producers and consumers, perhaps because of how other systems and programming paradigms tie input and output together, but in Kafka, these two entities are independent.

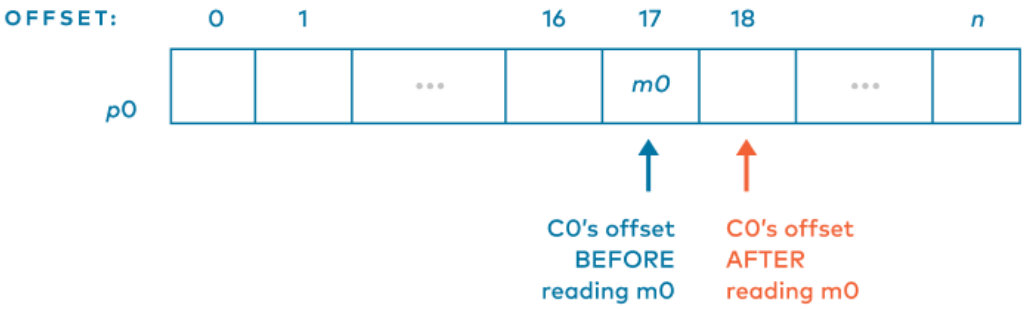
Problem #3B: Single Consumption or Multiple Consumption?

Suppose you have a Kafka cluster; one producer, p_0, producing to topic t_0, with one partition; and one consumer c_0, reading from topic t_0. Independent of the last question, suppose consumer c_0 read message m_0. Could another consumer c_{10}, in another consumer group, also consume message m_0?

Solution #3B: Single Consumption or Multiple Consumption?

c_0 has read a given message, m_0. Let's back up to dissect how this happened:

- First, some producer had to have written m_0 to our sole partition.
- Consumer c_0 had to have been assigned to read from that sole partition.
- Consumer c_0 keeps track of a consumer offset for the sole partition, which is the offset of which message it will read next. Say that c_0's consumer offset for our lone partition was 17, and m_0 was at offset 17.
- So, c_0 reads from offset 17 and gets m_0. Since c_0 has read from offset 17, it advances its offset for the sole partition to 18.



Notice that the message at offset 17 was not removed from the partition. **Kafka logs are not queues; messages don't get removed.** It is best to think of consuming as *reading* messages.

Going back to the question, let's assume that our other consumer c_{10} is assigned to consume from the same partition. Message m_0 is still there, so whenever c_{10} has an offset of 17 and asks for messages, it will indeed read m_0. This gets back to the key Kafka distinguishing feature of multiple consumption, powered by the fact that events are stored.

c_{10} was in a different *consumer group* from c_0. Why would c_{10} consume a message that c_0 already consumed? Each consumer group is doing something different with the same data. Maybe c_0 consumed the message to accomplish something that demands an immediate response, like retrieving a mobile food order, prints an order slip that goes to a restaurant's kitchen, and ensures the preparation of your lunch so that it's waiting for you in 15 minutes on the dot. Maybe c_{10} is going through all lunch orders received and analyzing them to see what all food was eaten during lunch at the restaurant. Perhaps it's working with some other applications or systems to make sure that inventory is in good shape by dinnertime. In the next problem, we dive deeper into consumer groups.