

KAFKA CONSUMER

Đơn vị: Công ty CP Giáo dục và Công nghệ QNET

QNET JOINT STOCK COMPANY

Address: 14th Floor, VTC Online Tower
18 Tam Trinh Street. Hoang Mai District
Hanoi, Vietnam



Quality Network for Education and Technology

Kafka Consumer

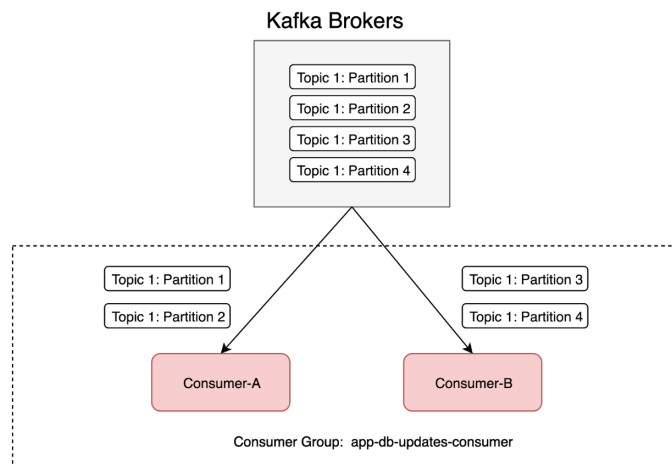
Learning Objective

- Overview of Consumer and Consumer Group and Partitions Rebalance
- Poll loop and It's functioning
- Configuring Consumers
- Commit and Offset
- Rebalance Listeners
- Consuming Records with Specific Offsets

Kafka Consumer

Overview

Kafka Consumer Group

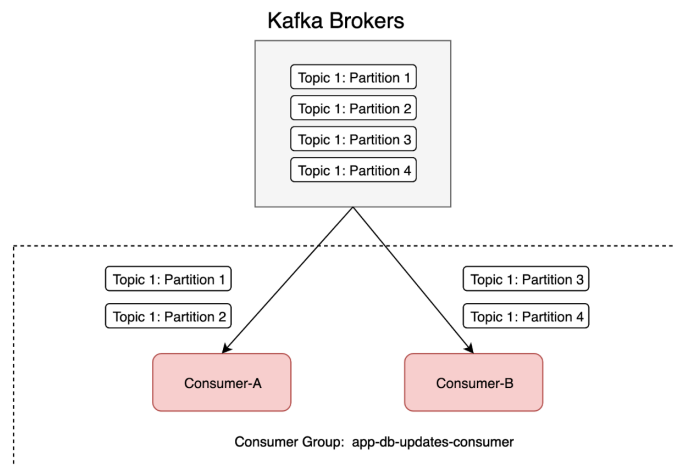


- Kafka Consumer: is application that consumes messages (generated by Producers) by subscribing to a topic
- Kafka consumers are typically part of a consumer group. When multiple consumers are subscribed to a topic and belong to the same consumer group, each consumer in the group will receive messages from a different subset of the partitions in the topic

Kafka Consumer

Overview

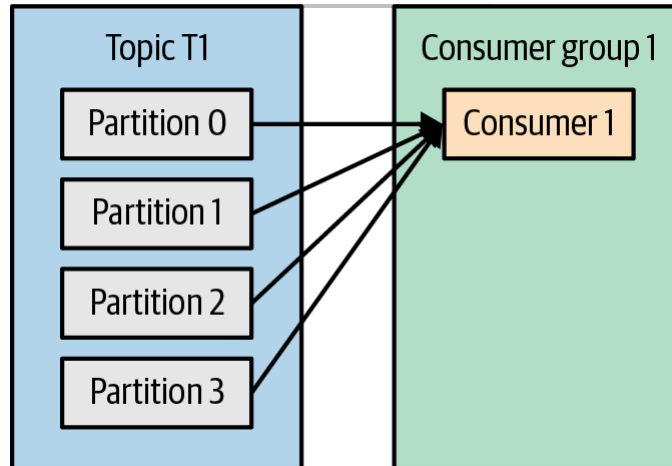
Kafka Consumer Group



- Consumers maintain membership in a consumer group and ownership of the partitions assigned to them by sending *heartbeats* to a Kafka broker designated as the *group coordinator* (this broker can be different for different consumer groups).
- The heartbeats are sent by a background thread of the consumer, and as long as the consumer is sending heartbeats at regular intervals, it is assumed to be alive.
- If the consumer stops sending heartbeats for long enough, its session will timeout and the group coordinator will consider it dead and trigger a rebalance. If a consumer crashed and stopped processing messages, it will take the group coordinator a few seconds without heartbeats to decide it is dead and trigger the rebalance. During those seconds, no messages will be processed from the partitions owned by the dead consumer.
- When closing a consumer, the consumer will notify the group coordinator that it is leaving, and the group coordinator will trigger a rebalance immediately, reducing the gap in processing.

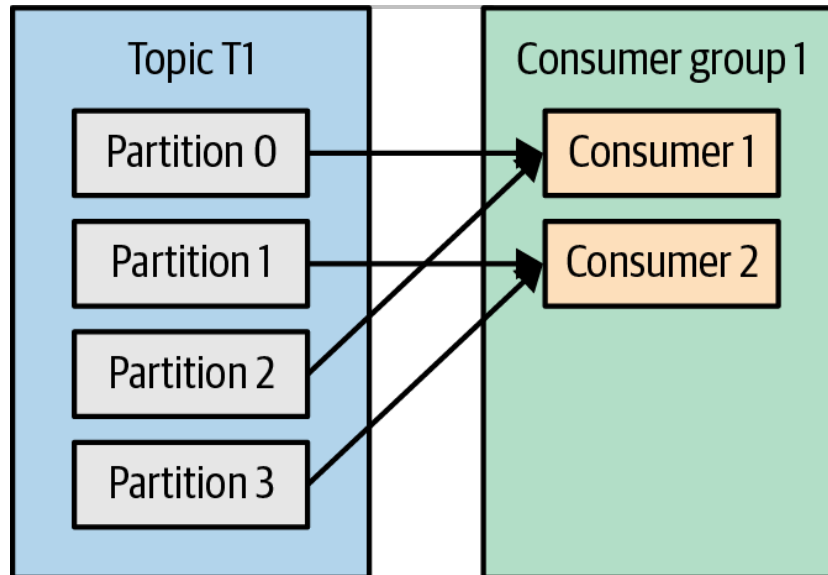
Kafka Consumer

One consumer group with four partitions



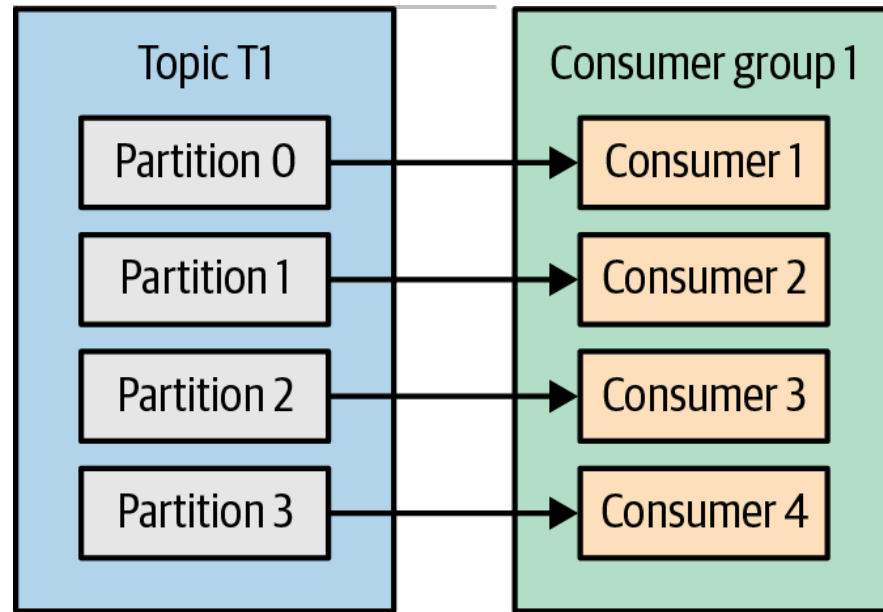
Kafka Consumer

Four partitions split to two consumers in a group



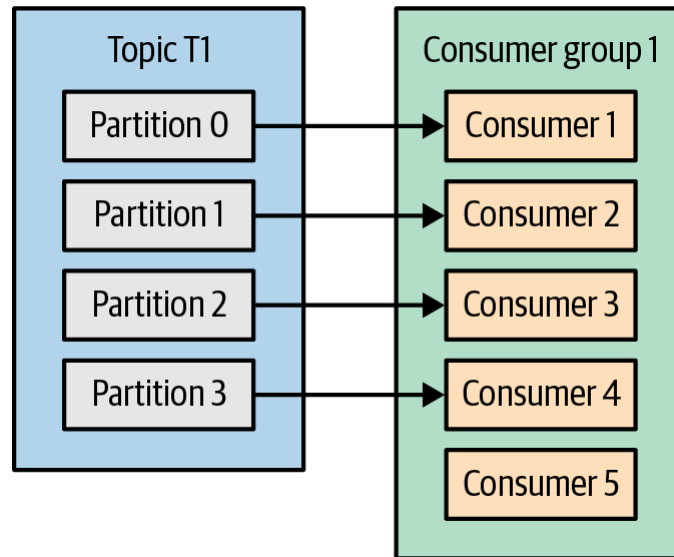
Kafka Consumer

Four consumers in a group with one partition each



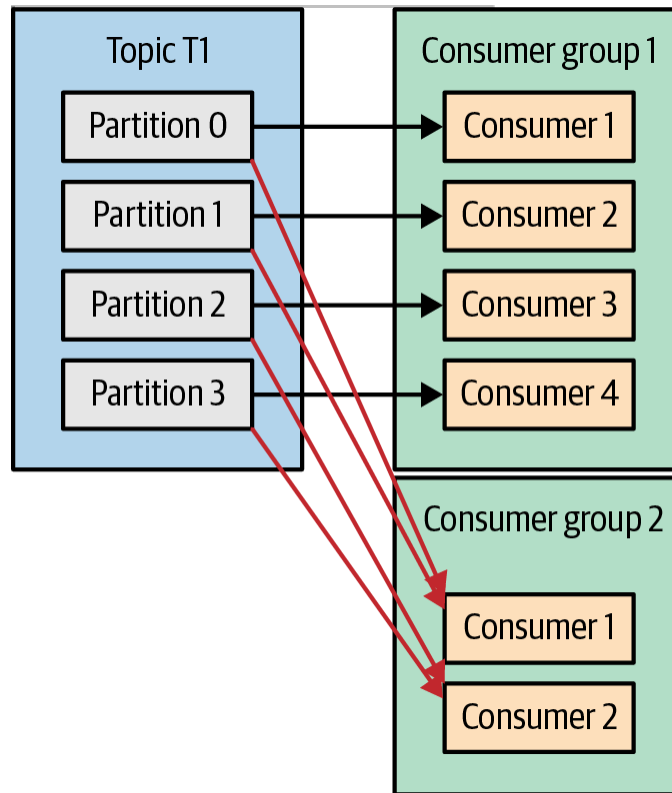
Kafka Consumer

More consumers in a group than partitions



Kafka Consumer

Add new consumer group



Kafka Consumer

Consumer Groups and Partition Rebalance

- Consumers in a consumer group share ownership of the partition in the topics they subscribe to.
- Reassignment of partitions to consumer group happens when:
 - Add a new consumer to a group
 - A consumer shutdowns or crashes
 - A consumer leaves the group
 - Topic that the consumer group is consuming is modified
 - A new partition is added
- The process that move partition ownership from one consumer group to another is called rebalance. Rebalances are important because they provide the consumer group with high availability and scalability (allowing us to easily and safely add and remove consumers), but in the normal course of events they can be fairly undesirable.

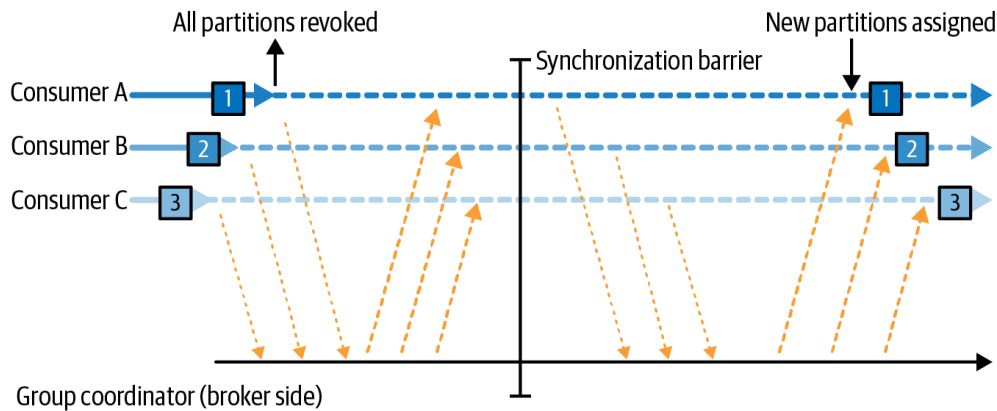
Kafka Consumer

Two types of rebalance

- Edge rebalance
- Cooperative rebalance

Kafka Consumer

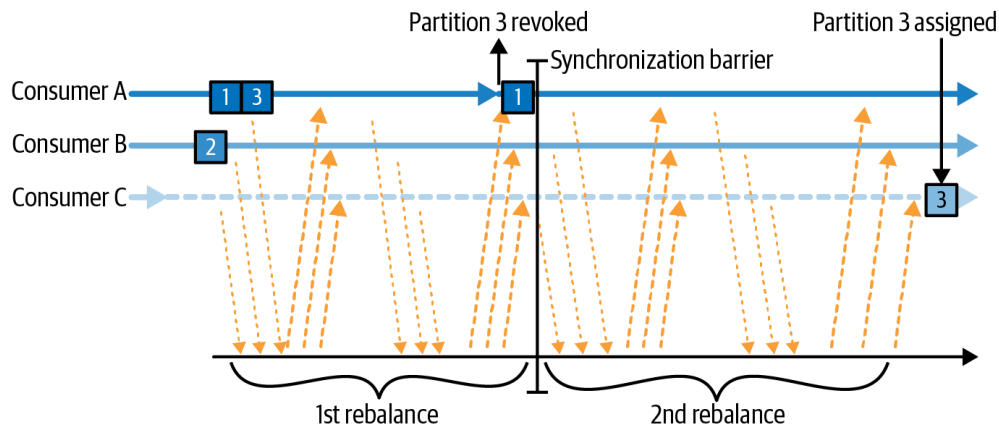
Edge Rebalances



- All consumer will:
 - Stop consuming
 - Give up their ownership of all partitions
 - Rejoin the consumer group
 - Get a brand-new partition reassignment

Kafka Consumer

Cooperative rebalances

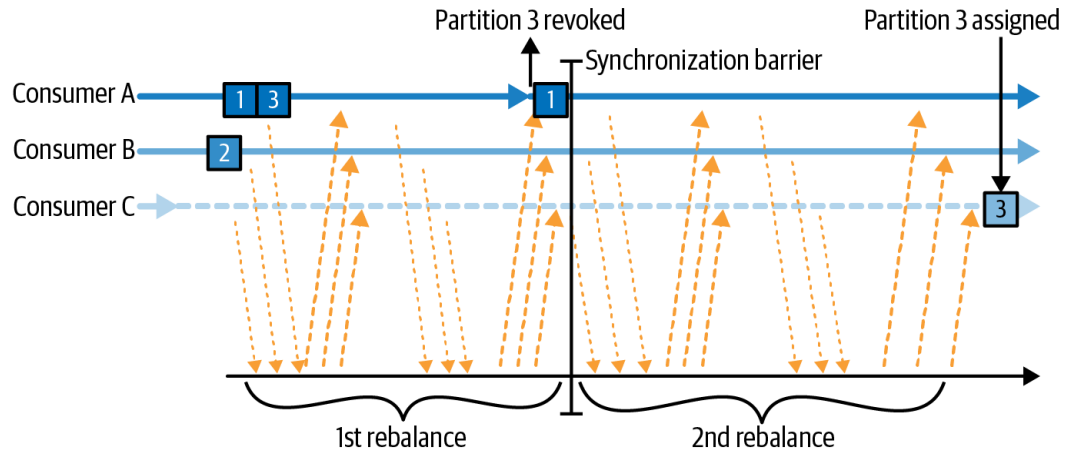


- Reassigning a small subset of the partitions from one consumer to another, allow consumers to continue consuming records from all the partitions that are not reassigned.

Kafka Consumer

Cooperative rebalances

- Cooperative rebalances process:
 - Consumer group leader informs all the consumers that they will lose the ownership of a subset of their partitions. Then the consumers stop consuming from these partitions and give their ownership in them.
 - The consumer group leader assigns these now orphaned partitions to their new owners. This incremental approach may take a few iterations until a stable partition assignment is archived.



Kafka Consumer

Creating a new Consumer

- Create `KafkaConsumer` instance then pass `Properties` instance with properties to consumer.
- There mandatory properties:
 - `Bootstrap.servers`: the connection string to a Kafka cluster
 - `Key.deserializer`: specifying classes that turn Java objects in message key to byte arrays
 - `Value.deserializer`: specifying classes that turn Java objects in message content to byte arrays
 - (optional)

Kafka Consumer

Creating a Consumer

```
Properties props = new Properties();
    props.put("bootstrap.servers", "broker1:9092,broker2:9092");
    props.put("group.id", "CountryCounter");
    props.put("key.deserializer",
        "org.apache.kafka.common.serialization.StringDeserializer");
    props.put("value.deserializer",
        "org.apache.kafka.common.serialization.StringDeserializer");
KafkaConsumer<String, String> consumer =
    new KafkaConsumer<String, String>(props);
```


Kafka Consumer

Subscribing to Topics

- Once we create a consumer, the next step is to subscribe to one or more topics. The `subscribe()` method takes a list of topics as a parameter, so it's pretty simple to use:

```
consumer.subscribe(Collections.singletonList("customerCountries"));
```

Here we simply create a list with a single element: the topic name `customerCountries`.

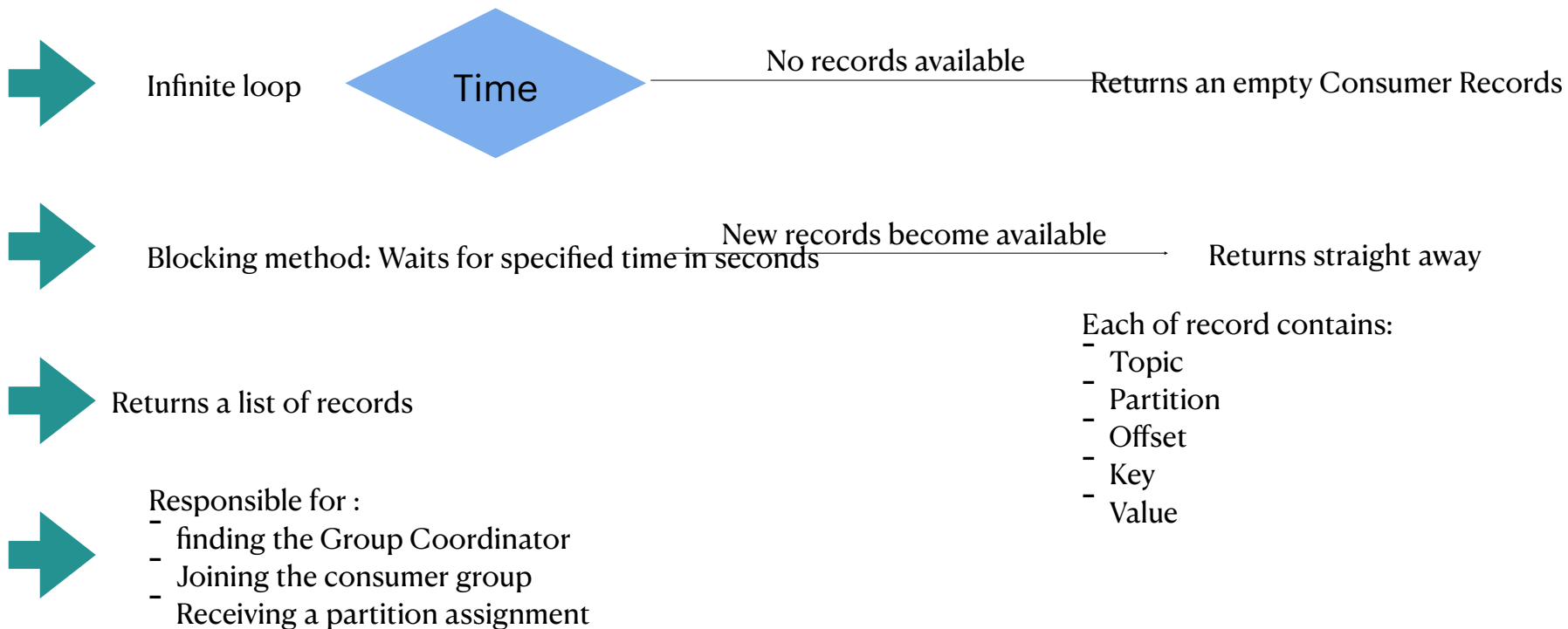
- It is also possible to call `subscribe` with a regular expression.

For example, to subscribe to all test topics, we can call:

```
consumer.subscribe(Pattern.compile("test.*"));
```

Kafka Consumer

Poll loop



Kafka Consumer

Configuring Consumers

These parameters may impact on memory use, performance, and reliability of the producers

`fetch.min.bytes` `fetch.max.wait.ms` `max.partition.fetch.bytes` `enable.auto.commit`

`Auto.offset.reset` `partition.assignment.strategy`

`receive.buffer.bytes` `send.buffer.bytes` `max.poll.records`

Kafka Consumer


Kafka Consumer Configurations

fetch.min.b

- To specify the minimum amount of data that it wants to receive from the broker when fetching records
- If the new record contain fewer bytes than min.fetch.bytes, broker will wait until more messages are available before sending the records
- Recommended to set it higher than the default. In the scenario when consumer is using too much CPU (when not much data available)

Kafka Consumer

Kafka consumer configurations

- 
- Controls how long to wait in case id does not have enough data
 - Max value is 500ms
 - To reduce latency due to this wait, configure a lower value

Kafka Consumer

Kafka consumer configurations

max.partition.fetch

- Controls the maximum number of bytes server will return per partition
- Default: 1MB

Kafka Consumer

Kafka consumer configurations

session.timeout.

- Controls how long a consumer can be considered alive without sending a heartbeat
- Default: 3s
- After expiry of this time, group coordinator will trigger rebalance
- Higher value:
 - Reduce the chance of accidental rebalance
 - Takes longer to detect and recovery from failure
- Lower value:
 - Quick detection and recovery from failure
 - May cause unwanted rebalances

Kafka Consumer

Kafka consumer configurations

auto.offset.reset

- Controls the behavior of the consumer when for a partition it doesn't have a committed offset or committed offset is invalid
- Default: Latest
- Consumer will start reading from the newest records, if there is no valid offset
- Can be set to "earliest". Consumer will start reading from the oldest records, if there is no valid offset

Kafka Consumer

Kafka consumer configurations

enable.auto.com

- Controls automatic commit
- Default: True
- Set to false if you want to control when offsets are committed

Kafka Consumer

Kafka consumer configurations

partition.assignment.strate

- To choose a partition-assignment strategy
- 2 strategies
 - Range - Default : Assigns a consecutive subset of partitions from each topic consumer is subscribed
 - Round robin: takes all the partitions from all subscribed topics and assigns them to consumers sequentially

Kafka Consumer

Kafka consumer configurations

max.poll.records

- Controls the maximum number of records that a single call to poll() will return
- Enables controlling the amount of data application need to process in the polling loop

Kafka Consumer

Kafka consumer configurations

receive.buffer.bytes

- Size of the TCP receive buffer
- OS defaults will be used, if its set to -1

Kafka Consumer

Kafka consumer configurations

send.buffer.bytes

- Size of the TCP receive buffer
- OS defaults will be used, if its set to -1

Kafka Consumer

Commit and Offset

Commit

Action of updating the current position of the consumer in the partition

Offset

Current position of a consumer/producer in each partition is called Offset
Each message in a partition is assigned a sequential id called an offset that uniquely identifies each message within the partition

Offset which tells how many messages have been produced or consumed

Kafka Consumer

Commit and Offset

How does consumer commit offsets?

It sends a message to Kafka, which updates a special `__consumer_offsets` topic with the committed offset for each partition.

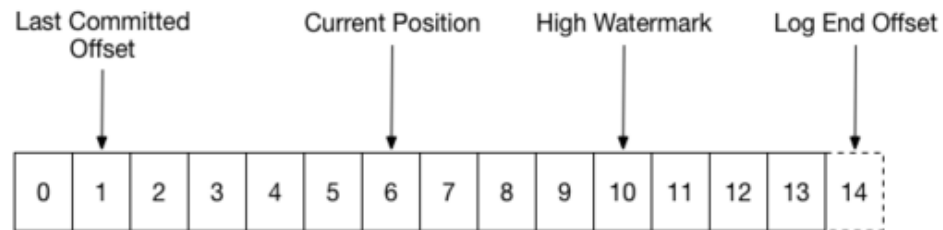


Figure 2: The Consumer's Position in the Log

DISCUSSION



Quality Network for Education and Technology

XIN CHÂN THÀNH CẢM ƠN!