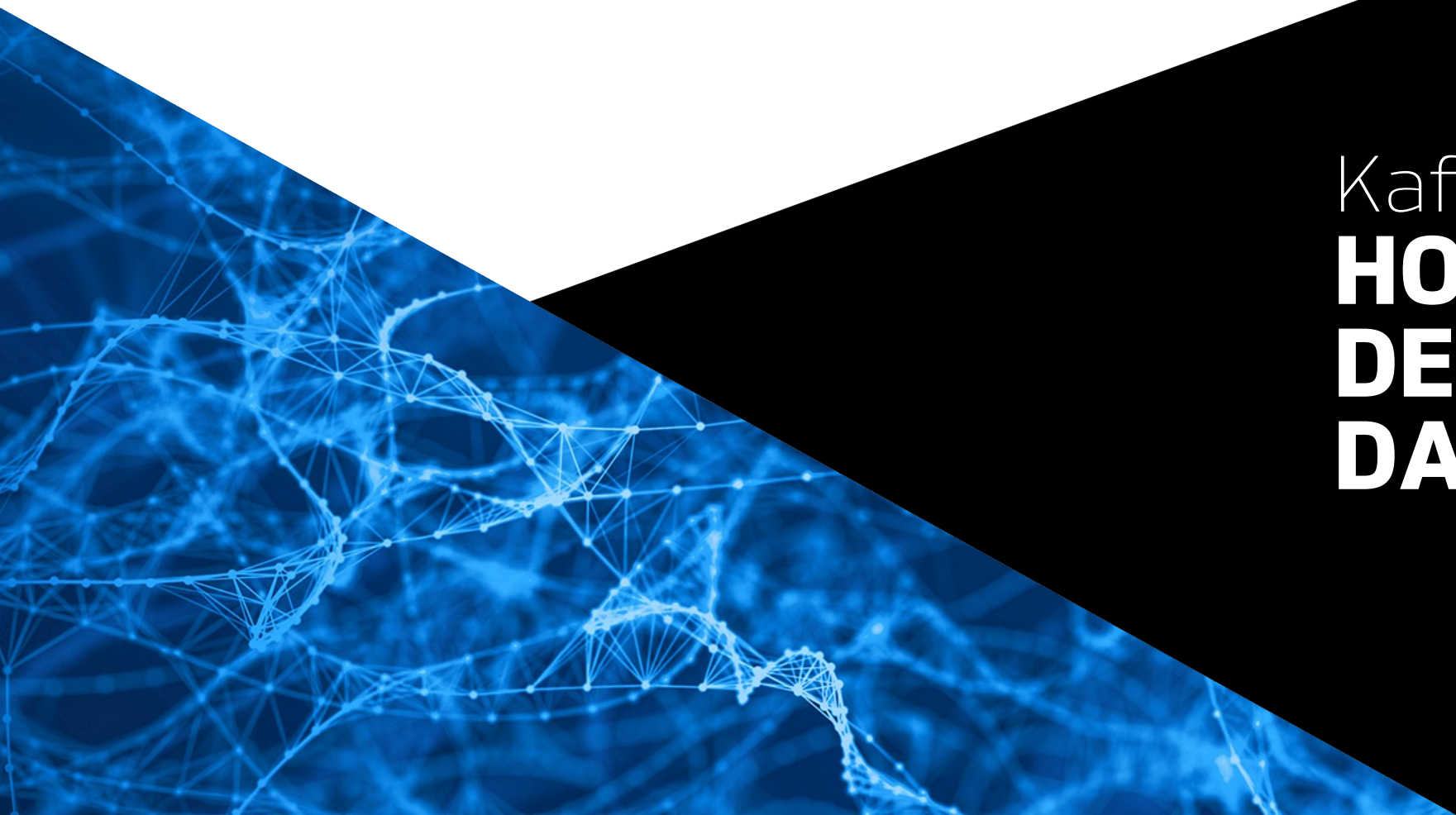
DEEPSHORE



Kafka: HOW TO DELETE DATA

IMPORTANT THINGS TO KNOW ABOUT: KAFKA DATA DELETION

- **Kafka is able to delete old messages** from a topic.
- The **Kafka** design **does not include targeted deletion of single messages**.
- **Targeted deletion of single messages should be handled outside of Kafka (on consumer side)**.
- However, **Kafka provides mechanisms** to support some control over the process **of data deletion**.

IMPORTANT THINGS TO KNOW ABOUT: KAFKA DATA DELETION

- In detail, these options are quite complicated to understand.
- There are **different configurations** that will **affect each other**.
- Let's try to understand what happens under the hood...

01

KAFKA CLEANUP POLICY

KAFKA CLEANUP POLICY

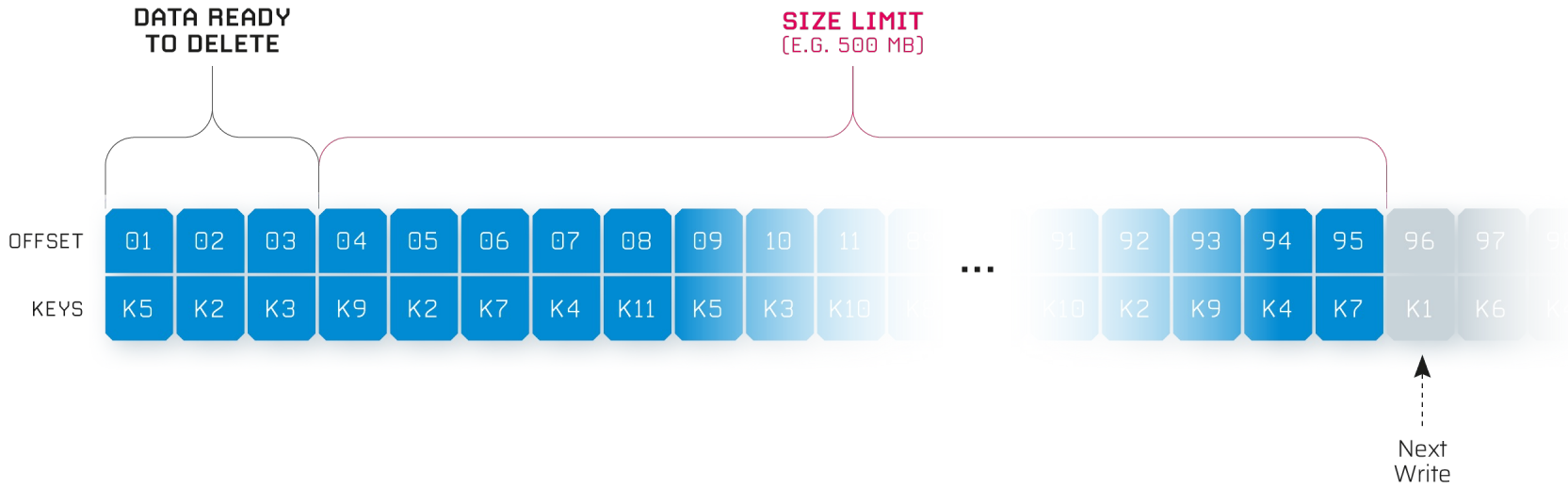
- The concept of **deleting data from Kafka is called „cleanup policy“**.
- The cleanup policy determines how Kafka handles the removal of messages from the log segments on disk.
- There are **two different approaches** of handling such data:
  - **Delete**
  - **Compact**
  - or both at the same time.

KAFKA CLEANUP POLICY – DELETE

- When the **„Delete“ cleanup policy** is set for a topic, Kafka will **delete messages based on** their individual **retention time or log size**.
- The **default setting is time-tiered** compaction with a retention period of 7 days (inherited from the Broker default).

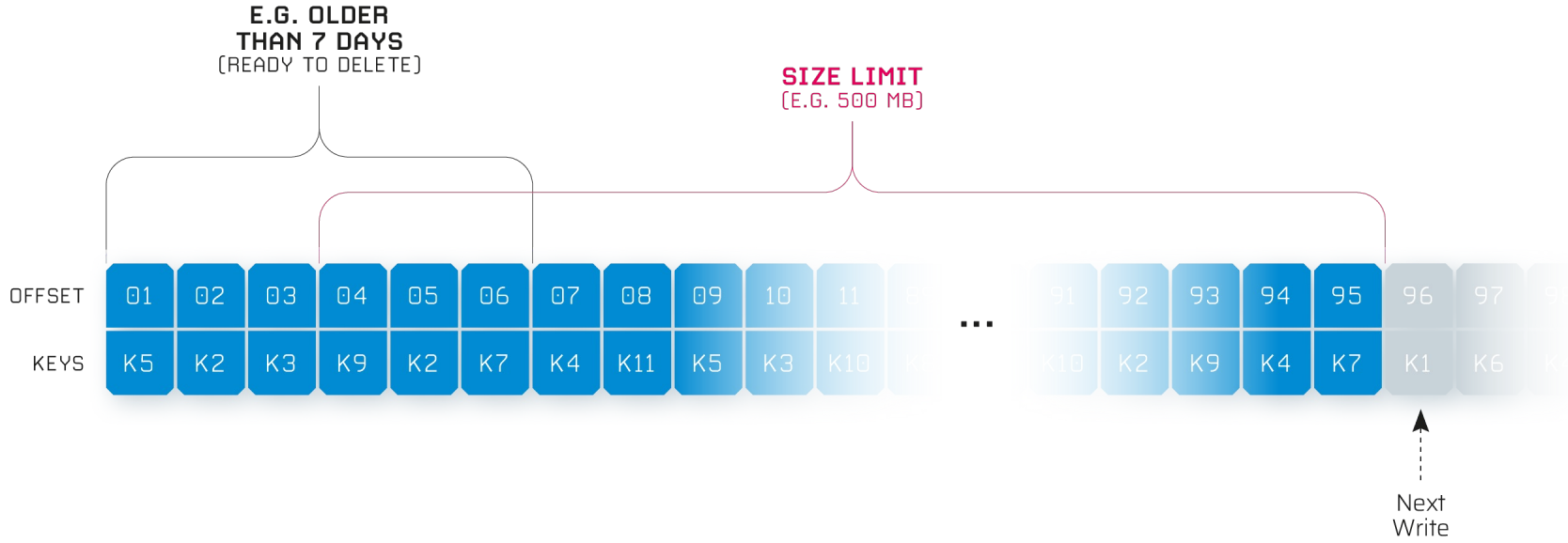
KAFKA CLEANUP POLICY – DELETE (SIZE)

- When using **size-tiered**, the ***log.retention.bytes*** configuration **determines the maximum size** in bytes that a Kafka topic's log can grow before old log segments start to be deleted.
- **It is applied** at a **per-partition** level.



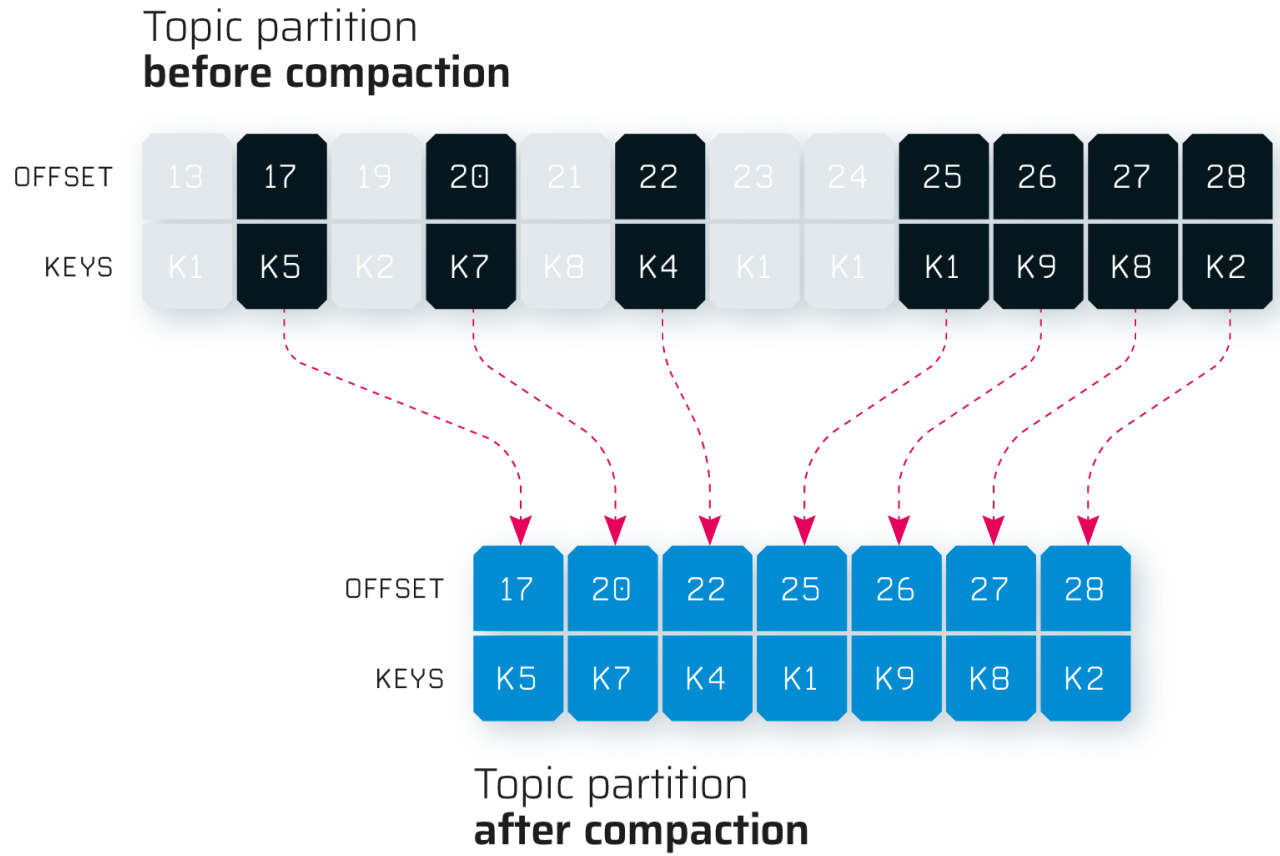
KAFKA CLEANUP POLICY – DELETE (SIZE)

- **By default, *log.retention.bytes* is turned off.**
- You can **combine time-tiered and size-tiered** configurations.
- In such a case, Kafka will use the first trigger point from either limit.



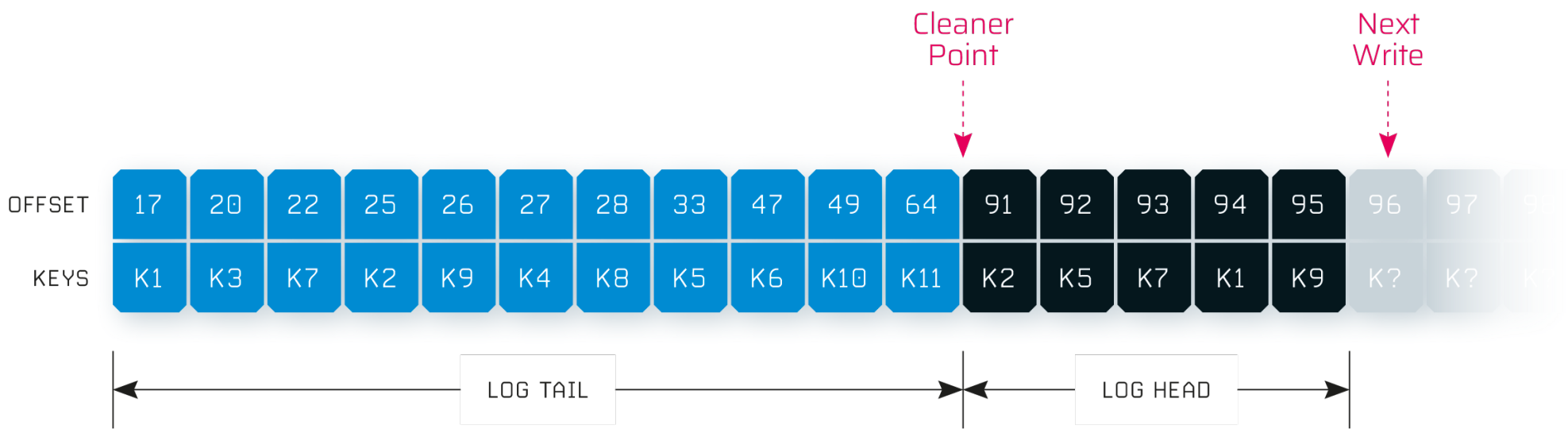
KAFKA CLEANUP POLICY – COMPACT

- Compaction ensures that the **log retains** a compacted representation for **each Key**.
- Only the **latest value for each key** is preserved.
- **It is applied** on a **per partition** level.



DEEPSHORE

A TOPIC AFTER COMPACTION



KAFKA CLEANUP POLICY – DELETE & COMPACT

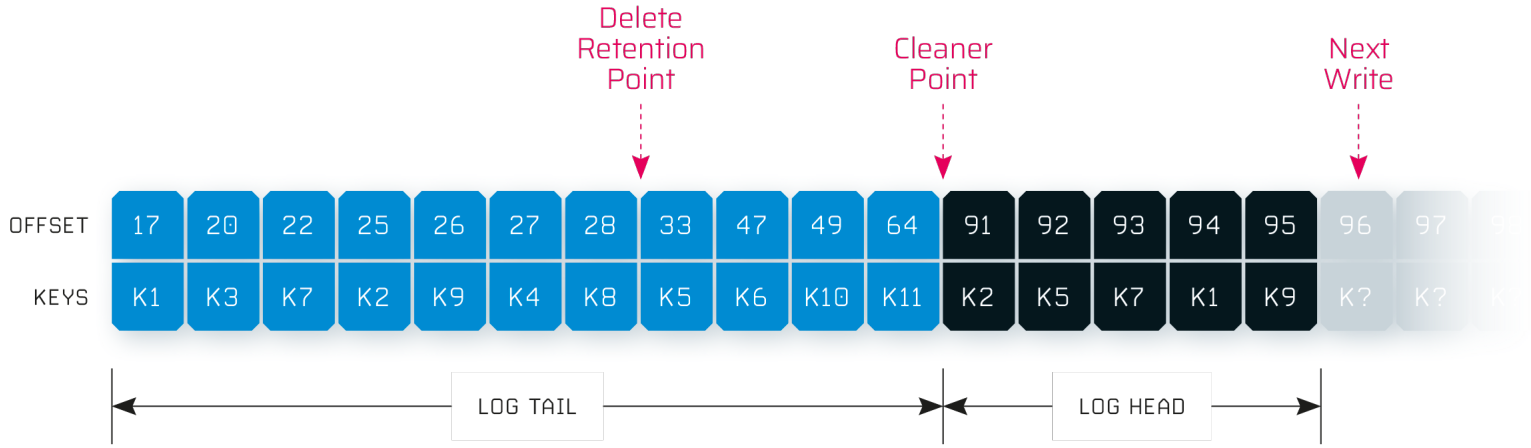
- You **can combine** the **DELETE & COMPACT** configurations.
- In such a case, **Kafka will use both policies at the same time**, e.g., keeping a message per key as long as the retention period has not expired.
- When the retention period has been reached, a message can be deleted, even if there is no other message with the same key.

02

KAFKA TOMBSTONE

KAFKA – TOMBSTONE EVENT

- A message **with a key** and a **null payload** is called a **TOMBSTONE**.
- It will be treated as **a delete-request for a messages with the same key**.
- This **delete marker will cause any prior message with that key to be removed**.
- The point in time at which tombstones are no longer retained is marked as the ***delete retention point***.



KAFKA – TOMBSTONE EVENT

- The **consumer sees all tombstones** as long as it reaches the head of a log within a period less than the topic configurated ***delete.retention.ms*** (the default is 24 hours).
- **Compaction is performed periodically in the background**, but it is not 100% predictable.
  - The number of cleaner threads are configurable through ***log.cleaner.threads*** configuration (default = 1).
- The cleaner thread chooses the log with the highest dirty ratio first
  → **dirty ratio = number of bytes in the head / total number of bytes in the log (tail + head)**.

KAFKA – TOMBSTONE EVENT

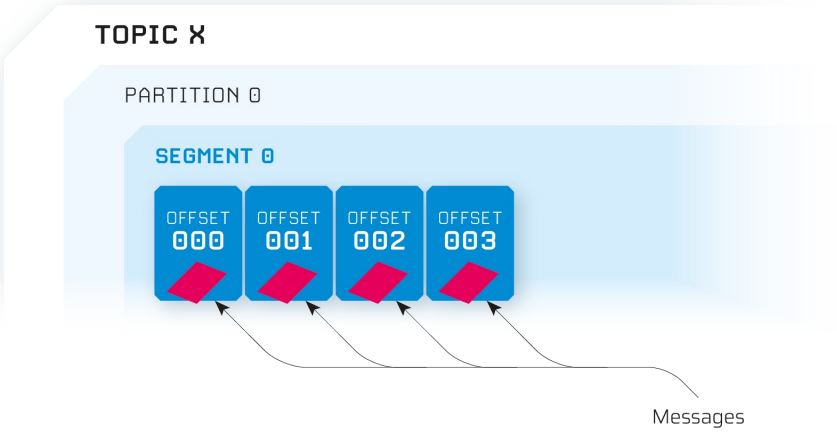
- Also consider: ***min.cleanable.dirty.ratio*** (default = 0,5)
  - **This threshold also influences compaction**. Meaning a topic/partition file will only be compacted if at least 50% of its entries are dirty. **Anything below that, the thread does not perform compaction**.
- Topic config ***min.compaction.lag.ms*** (default = 0) defines the minimum time period that must pass, before a message can be compacted.
- To set delay to start compacting records after they are written use topic config ***log.cleaner.min.compaction.lag.ms*** (default = 0) . The setting gives consumers time to get every record.

03

KAFKA SEGMENTS

KAFKA – ON DISK

- **My messages reached the end of their retention period, but they are still there? Why?**
- Kafka does not organize or store single messages on disk. Instead, **messages are collected** and stored in what is called **"segment"** files.
  - **Segments are specific to each partition**, and therefore, each topic.

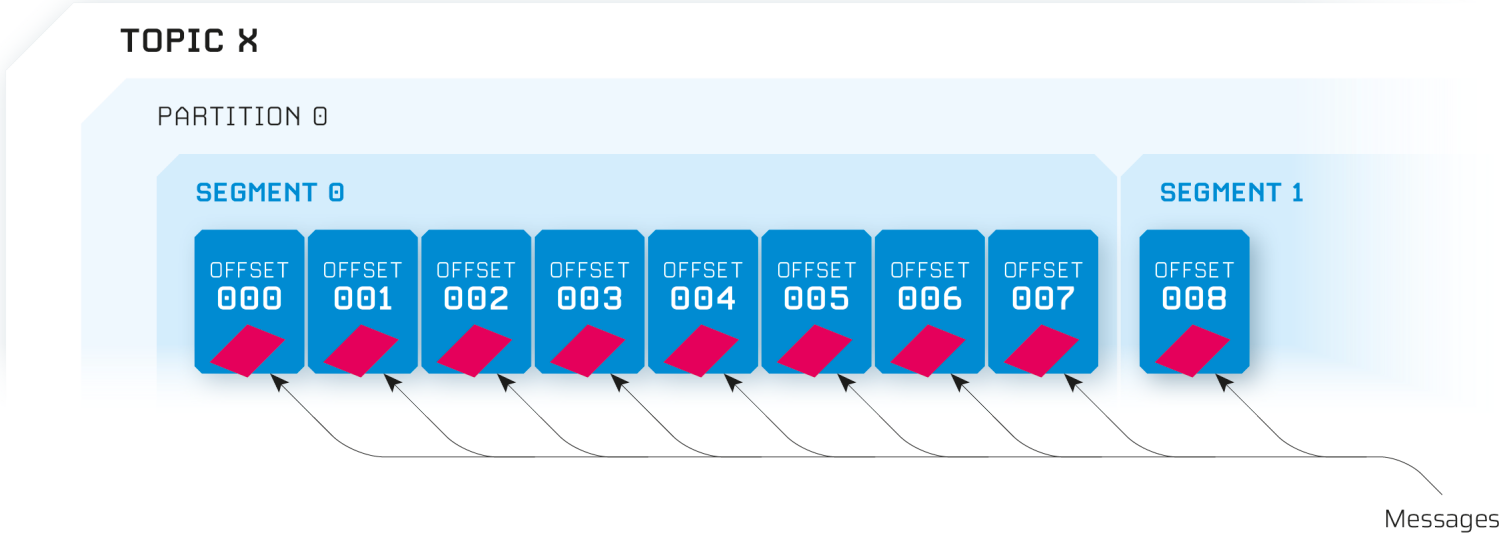


KAFKA – ON DISK

- **Data will always be deleted in bulk**, defined by the size of the segment.
- You can **control the segment creation** by using some Kafka configuration parameters (on topic level):
  - The **maximum age of messages** in a segment file (***segment.ms*** – default is 7 days).
  - The **maximum size of a segment** file (***segment.bytes***).
- If either of these is exceeded, the broker will start a new segment.

KAFKA – ON DISK

- Segments will be **deleted only if every single message is deletable** (expired retention or tombstone event).



KAFKA – ON DISK

- **When data seems to live forever, a known and sophisticated error might be on the producer side**.
  - ***CreateTime*** is the default timestamp assigned to a message by the producer.
  - Kafka uses ***log.retention.hours*** (default is 168 hours) together with ***CreateTime*** to determine when to delete a log file. A log file will only be deleted if the **latest *timestamp* of any record in that log file is older than 168 hours**.
  - If a producer sends a message with an incorrect ***timestamp*** (e.g., 01.01.2100), the segment and its messages will not be deleted within a reasonable timeframe.

04

TAKE AWAY...

TAKE AWAY...

- **Kafka offers no functionality to delete single messages.**
  - The lowest level of control for deletion is on a per-key basis (aka **tombstone**).
  - **Kafka will not delete a message immediately** when it is "ready to delete."
  - It is not possible to predict the exact point in time since **deletion is a background job**.
  - Triggering the job depends on different configurations.
  - Even if the job is running, **it might skip segments** due to other topic configurations.
- **You should never build a business case that relies on the physical deletion of data from Kafka.**

Who said:

DELETING DATA FROM KAFKA IS EASY?

CONTACT

Deepshore GmbH · Van-der-Smissen-Straße 9, 22767 Hamburg
Telefon +49 40 46664-296 · Fax +49 40 46664-299
E-Mail info@deepshore.de · **www.deepshore.de**