🔁 log.cleanup.policy in Kafka

Kafka topics retain messages for a certain duration or based on the policy set. The log.cleanup.policy defines **how Kafka deletes or compacts old data** in a topic.

Kafka provides **two primary cleanup policies**:

1. delete (Default)

- **Description**: Kafka will **retain messages for a configured period** (retention.ms) and then **delete them**.
- **Use case**: Use this when **every event is important**, like in transaction logs, logs that represent events over time, or in auditing.

2. compact

- **Description**: Kafka **retains only the latest message for each key**. Older messages with the same key are **removed**, keeping only the most recent version.
- **Use case**: Used for **stateful data**, like database change logs, where only the latest value per key matters — like **counters**, **user profiles**, or **settings**.

🏦 1. Banking Service — cleanup.policy = delete

Scenario: Transactions Performed by Customers

Key (custID)	Value (Transaction)
1001	deposit
1002	withdraw
1001	withdraw
1001	deposit

- **Policy Used**: delete (default)
- **Why?**:
    - Each transaction is **important and must be preserved**, at least for a retention period.
    - You want to **see the history of all customer actions**.
    - Compacting would **remove older transactions** for the same key (custID), which is **not acceptable** in financial logs.

Behavior:

- Kafka keeps all these messages **temporarily**.
- Once retention time is exceeded, old messages **are deleted**.
- **All events are stored**, allowing full audit/history.

📊 2. Aggregation Operations — cleanup.policy = compact

Scenario: Word Counts in Real-Time

Key (Word)	Value (Count)
kafka	1
spring	1
kafka	2
kafka	5

- **Policy Used**: compact
- **Why?**
  - You only need the **latest count per word**.
  - Earlier values (e.g., 1, 2) for "kafka" are **obsolete** once "kafka: 5" arrives.
  - Compaction **keeps storage small and data relevant**.

Behavior:

- Kafka **keeps only the latest record per key**.
- For "kafka", only the message kafka: 5 remains.
- The **state is always up-to-date**, which is great for key-value stores, caches, and materialized views.

🆚 Comparison Table

Feature	delete	compact
Retains all versions?	Yes, until retention time	No, keeps only latest per key
Use Case	Audit logs, financial transactions	Caches, counters, user profiles
Storage requirement	Higher (all messages kept temporarily)	Lower (old versions removed)
Suitable for streaming history?	Yes	No (history is overwritten per key)
Example	Banking transactions	Word count aggregation

✅ **Summary**

- **Use delete** when **event history matters**, and each message is important (e.g., every customer transaction).
- **Use compact** when **you only care about the latest value** per key (e.g., running totals, latest user info).