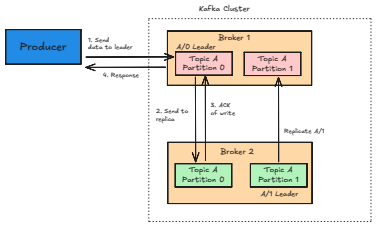
Producers



What does a producer do?

A producer serializes a record, chooses a partition, batches records per partition, then sends those batches over the network to the broker leader, waiting for an acknowledgement.

Choosing a Partition

- If a key is provided, same key goes to the same partition
- If no key, Kafka uses sticky partitioner, which keeps sending to one partition for a while to improve batching

Batching and Sending

- Producers batch messages per partition to reduce network calls
- When batch is full or a short timer expires, it's sent to the broker
- This is what gives kafka high throughput, it turns many small writes into fewer bigger ones

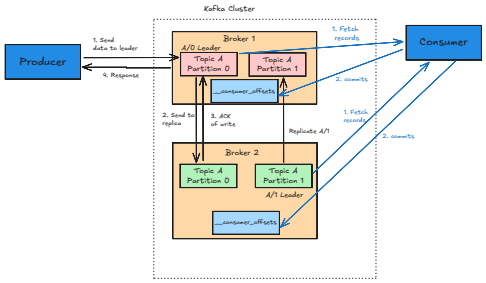
Acknowledgements

- After the broker writes the batch to its log, it sends an acknowledgement:
acks = 0 : don't wait (fast but unsafe)
acks = 1: leader confirms
acks = all: wait for replica (safest)

Reliability and Idempotence

- Producers can retry automatically if a send fails
- Kafka's producer keeps track of sequence number to prevent duplicates

Consumers



What does a consumer do?

A consumer reads data from Kafka topics. It fetches messages from one or more partitions and processes them in order.

How does it read?

- Consumers poll for message in batches, not one by one
- Kafka returns data starting from the last committed offset
- Each record has an offset
- The consumer keeps track of its own position in the log using the offsets, if it restarts, it can resume exactly where it left off

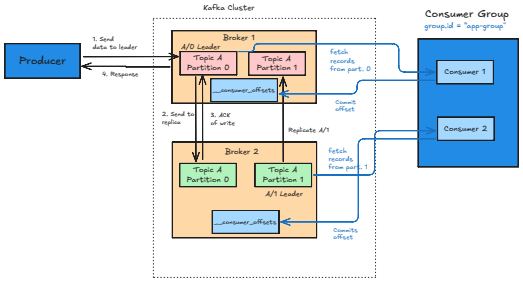
Commit means saving the consumer's current read position. The last offset that was successfully processed

Offsets stored in an internal Kafka topic named __consumer_offsets

Offset management

- Offsets can be automatically committed (at intervals) or manually commited after successfull processing
- Offsets are like checkpoints, commit them too soon, you might skip data, too late, you might reprocess it

Consumers Groups



What are Consumer Groups?

A consumer group is a collection of consumers that work together to consume message from a topic.

Why Consumer groups?

- It makes scaling easy, multiple consumers can work together on the same topic, each taking a slice of partitions
- Instead of having a single consumer handle all messages, each one handles portions of the messages
- Multiple groups can independently process the same topic:
  - One group process orders for fulfillment
  - One group process orders for BI
  - One group archives orders for compliance

Rebalancing

- When group membership changes (consumers leaving or being added to group) Kafka pauses consumption,
Redistributes partitions among active consumers
Resumes reading

Rebalancing happens automatically when consumers appear, disappear or crash. It ensures load is shared, but can cause short pauses

The Poll loop (heartbeat)

The poll loop is the consumer's main engine:
- Poll for new data
- Process records
- Commit offsets
- Repeat

Regular polling sends heartbeats to keep the consumer in the group
If the consumer stops polling for too long, the broker assumets it's dead and rebalances its partitions