Producer Delivery Semantics

As we Know, producer write new messages to the leader broker, and the followers asynchronously replicates the data. How will the Producer will know , the data is successfully stored in the Leader Broker or followers are keeping up with the leader. Kafka offers 3 options to denote the numbers of brokers that must receive the records before the producer considers the writes are successful.

- **Async :** Producer sends the message and doesn't waits for success acknowledgement. Once the message is sent, it is considered as success. [Fire and forget approach, but there is not guarantee that server has received the records]

- **Committed to leader :** Producer waits for an acknowledgment from the leader. This approach is slower than Async but it ensures that the data is received successfully by leader. Under this leader will not wait for acknowledgement from its followers. In this case data can also lost, if leader crashes after the acknowledgment, but before the followers have replicated the message.

- **Commit to Leader & Quorum :** Here Producer waits for an acknowledgment from the leader and quorum. This means leader will also wait for the full set of in-sync replicas to acknowledgment for the record.This is the slowest but guarantees that the record is not lost until a single in-sync replica is still alive.

We can choose any one of above, totally depending on our preference.

Consumer Delivery Semantics

A Consumer can only read the message if it have been written to the set of in-sync replicas. There are three ways of providing consistency of the consumer:

- **At most Once [Message may be lost but are never re-delivered]**

In this option, a message is consumed by the consumer and the consumer increments the offset to the broker, but if the consumer crashes it will loose the message as it have incremented the offset.

- **At least Once [Message are never lost but may be never re-delivered]**

In this, a message be delivered more than once, but no message is lost. It doesn't immediately increment the offset , instead it waits for the message processing to be done. So if the consumer crashes after the message processing and not updating the offset, the broker will re-deliver the same old message.

- **Exactly Once [Each Message is delivered once & only once]**

It is hard to achieve, only if the consumer is working with the transactional system. In this option, consumer put the message in processing and updates the offset in one transaction. This ensures that the offset increment will only happen if the transaction is completed. If consumer crashes, while processing the transaction will roll back and offset is not incremented. When the consumer restarts, it can re-read the message as it failed to process last time. This option leads to no data duplication and no data loss but can lead to decreased throughput.