How often old log will be collected:

retention.ms

segment.ms

segment.bytes

For a specific partition, the position always moves forward even if old segment is reclaimed and new log segment is created. Logic position is not equivalent to file position. An index file maintains mapping between these two.

Listeners: most important configuration in server.properties, without specifying this. Producer and consumer will not be able to connect to broker. Each listener has to use a different protocol. All producers and consumers can share the same port

Delete topic command will not take effect immediately, it only marks topic for deletion. delete.topic.enable=true

Producer send will not directly throw exception, the callback registered will show error details. Sometimes it will wait for “max.block.ms”, eg: metadata cannot be retrieved

Topic can be auto created on first publishing if non-exists (use default configuration in server.properties)

partition is assigned only after first polling, and seek is not possible until partition is assigned

consumed data will not be re-consumed without seeking to beginning, even if position is not committed

Control data fetch size: kafka cannot directly control how much messages fetch per polling

However there is some work around that provides inaccurate control : setting max.partition.fetch.bytes (if less then max.request.size, consumer will report an error on polling)

seekToBeginning/seek are able to go beyond the commit position after first polling.

Commit to position that behind current commit position will resulting in no data fetched by next polling

<https://cwiki.apache.org/confluence/display/KAFKA/FAQ>

Why does my consumer never get any data?

By default, when a consumer is started for the very first time, it ignores all existing data in a topic and will only consume new data coming in after the consumer is started. If this is the case, try sending some more data after the consumer is started. Alternatively, you can configure the consumer by setting auto.offset.reset to "earliest" for the new consumer in 0.9 and "smallest" for the old consumer.

How soon consumer failover will happen?

When a consumer leaves the cluster, some other consumer will take over its partitions. This is called repartition and consumer does repartition check on each polling.

session.timeout.ms determines how soon a consumer is considered dead after it is not responding

make sure your polling timeout (ie: consumer.poll(timeout)) + sleep interval between polling > session.timeout.ms, then next polling result will reflect the reassigned partitions

It seems that the heartbeat interval: heartbeat.interval.ms has no effect

Note that new consumer joining the cluster will not trigger repartition