**生产者配置参数：**

**acks**：指定了必须有多少个分区副本收到消息，生产者才会认为消息写入是成功的。默认为**acks=1**

**acks=0** 如果设置为 0，则 Producer 不会等待服务器的反馈。该消息会被立刻添加到 socket buffer 中并认为已经发送完成。在这种情况下，服务器是否收到请求是没法保证的，并且参数**retries**也不会生效（因为客户端无法获得失败信息）。每个记录返回的 offset 总是被设置为-1。

**acks=1** 如果设置为 1，leader 节点会将记录写入本地日志，并且在所有 follower 节点反馈之前就先确认成功。在这种情况下，如果 leader 节点在接收记录之后，并且在 follower 节点复制数据完成之前产生错误，则这条记录会丢失。

**acks=all** 如果设置为 all，这就意味着 leader 节点会等待所有同步中的副本（ISR）确认之后再确认这条记录是否发送完成。只要至少有一个同步副本存在，记录就不会丢失。这种方式是对请求传递的最有效保证。acks=-1 与 acks=all 是等效的。

**buffer.memory**：用来设置 Producer 缓冲区大小。

**compression.type**：Producer 生成数据时可使用的压缩类型。默认值是 none(即不压缩)。可配置的压缩类型包括：**none**、**gzip**、**snappy** 、**lz4** 或 **zstd**。压缩是针对批处理的所有数据，所以批处理的效果也会影响压缩比（更多的批处理意味着更好的压缩）。

**retries**：用来设置发送失败的重试次数。

**batch.size**：用来设置一个批次可占用的内存大小。

**linger.ms**：用来设置 Producer 在发送批次前的等待时间。

**client.id**：Kafka 服务器用它来识别消息源，可以是任意字符串。

**max.in.flight.requests.per.connection**：用来设置Producer在单个连接上能够发送的未响应请求的个数。**设置此值是1表示kafka broker在响应请求之前client不能再向同一个broker发送请求。**默认值为5。如果设置1，可以**避免生产者发送消息乱序，虽然吞吐量降低了，但是安全性得到了提升，要权衡业务场景配置**。（比如生产者发送两条顺序消息1,2，都是异步发送，同步发送性能低，如果2成功，1因为网络问题重试发送成功，1就到2后面，乱序了）。

**timeout.ms**：用来设置 Broker 等待同步副本返回消息确认的时间，与 **acks** 的配置相匹配。

**request.timeout.ms**：Producer 在发送数据时等待服务器返回响应的时间。

**metadata.fetch.timeout.ms**：Producer 在获取元数据时（如：分区的 Leader 是谁）等待服务器返回响应的时间。

**max.block.ms**：该配置控制 **KafkaProducer.send()** 和**KafkaProducer.partitionsFor()** 允许被阻塞的时长。这些方法可能因为缓冲区满了或者元数据不可用而被阻塞。用户提供的序列化程序或分区程序的阻塞将不会被计算到这个超时。

**max.request.size**：请求的最大字节数。

**receieve.buffer.bytes**：TCP 接收缓冲区的大小。

**send.buffer.bytes**：TCP 发送缓冲区的大小。

**producer.type**：该参数指定了在后台线程中消息的发送方式是同步的还是异步的，默认是sync的方式，即producer.type=sync。如果设置成异步的模式，即producer.type=async，但是这样会增加丢失数据的风险。如果需要确保消息的可靠性，必须要将producer.type设置为sync。

**queue.buffering.max.ms** 默认值：5000。启用异步模式时，producer缓存消息的时间。比如我们设置成1000时，它会缓存1s的数据再一次发送出去，这样可以极大的增加broker吞吐量，但也会造成时效性的降低。

**queue.buffering.max.messages** 默认值：10000。启用异步模式时，producer缓存队列里最大缓存的消息数量，如果超过这个值，producer就会阻塞或者丢掉消息。

**queue.enqueue.timeout.ms** 默认值：-1。当达到上面参数时producer会阻塞等待的时间。如果设置为0，buffer队列满时producer不会阻塞，消息直接被丢掉；若设置为-1，producer会被阻塞，不会丢消息。

**batch.num.messages** 默认值：200。启用异步模式时，一个batch缓存的消息数量。达到这个数值时，producer才会发送消息。（每次批量发送的数量）

**消费者配置参数：**

**bootstrap.servers** - Broker 集群地址，格式：ip1:port,ip2:port...，不需要设定全部的集群地址，设置两个或者两个以上即可。

**group.id** - 消费者隶属的消费者组名称，如果为空会报异常，一般而言，这个参数要有一定的业务意义。

**fetch.min.bytes** - 消费者获取记录的最小字节数。Kafka 会等到有足够的数据时才返回消息给消费者，以降低负载。

**fetch.max.wait.ms** - Kafka 需要等待足够的数据才返回给消费者，如果一直没有足够的数据，消费者就会迟迟收不到消息。所以需要指定 Broker 的等待延迟，一旦超时，直接返回数据给消费者。

**max.partition.fetch.bytes** - 指定了服务器从每个分区返回给消费者的最大字节数。默认为 1 MB。

**session.timeout.ms** - 指定了消费者的心跳超时时间。如果消费者没有在有效时间内发送心跳给群组协调器，协调器会视消费者已经消亡，从而触发分区再均衡。默认为 3 秒。

**auto.offset.reset**- 指定了消费者在读取一个没有偏移量的分区或偏移量无效的情况下，该如何处理。

**latest** - 表示在偏移量无效时，消费者将从最新的记录开始读取分区记录。

**earliest** - 表示在偏移量无效时，消费者将从起始位置读取分区记录。

**enable.auto.commit** - 指定了是否自动提交消息偏移量，默认开启。

**partition.assignment.strategy**- 消费者的分区分配策略。

**Range** - 表示会将主题的若干个连续的分区分配给消费者。

**RoundRobin** - 表示会将主题的所有分区按照轮询方式分配给消费者。

**client.id** - 客户端标识。

**max.poll.records** - 用于控制单次能获取到的记录数量。

**receive.buffer.bytes** - 用于设置 Socket 接收消息缓冲区（SO\_RECBUF）的大小，默认值为 64KB。如果设置为-1，则使用操作系统的默认值。

**send.buffer.bytes** - 用于设置 Socket 发送消息缓冲区（SO\_SNDBUF）的大小，默认值为 128KB。与 receive.buffer.bytes 参数一样，如果设置为-1，则使用操作系统的默认值。

**Broker配置参数：**

**replication.factor 的作用是设置每个分区的副本数**。**replication.factor** 是主题级别配置； **default.replication.factor** 是 broker 级别配置。

副本数越多，数据可靠性越高；但由于副本数增多，也会增加同步副本的开销，可能会降低集群的可用性。一般，建议设为 3，这也是 Kafka 的默认值。

**不完全的选主**

**unclean.leader.election.enable** 用于控制是否支持不同步的副本参与选举 Leader。**unclean.leader.election.enable** 是 broker 级别（实际上是集群范围内）配置，默认值为 true 。

如果设为 true，代表着**允许不同步的副本成为主副本**（即不完全的选举），那么将**面临丢失消息，消息不一致的风险**；

如果设为 false，就要**等待原先的主副本重新上线**，从而降低了可用性。

从Kafka 0.11.0.0版本开始**unclean.leader.election.enable**参数的默认值由原来的**true**改为**false**

**最少同步副本**

**min.insync.replicas 控制的是消息至少要被写入到多少个副本才算是“已提交”**。**min.insync.replicas** 是主题级别和 broker 级别配置。

尽管可以为一个主题配置 3 个副本，但还是可能会出现只有一个同步副本的情况。如果这个同步副本变为不可用，则必须在可用性和数据一致性之间做出选择。Kafka 中，消息只有被写入到所有的同步副本之后才被认为是已提交的。但如果只有一个同步副本，那么在这个副本不可用时，则数据就会丢失。

如果要确保已经提交的数据被已写入不止一个副本，就需要把最小同步副本的设置为大一点的值。

注意：要确保 **replication.factor** > **min.insync.replicas**。如果两者相等，那么只要有一个副本挂机，整个分区就无法正常工作了。我们不仅要改善消息的持久性，防止数据丢失，还要在不降低可用性的基础上完成。推荐设置成 **replication.factor = min.insync.replicas + 1**。

**auto.create.topics.enable** 默认值为true,生产者向一个尚未创建的topic发送消息时，会自动创建一个num.partitions（默认值为1）个分区和default.replication.factor（默认值为1）个副本的对应topic。不过我们一般不建议将**auto.create.topics.enable**参数设置为true，因为这个参数会影响topic的管理与维护

**log.retention.check.interval.ms** 周期性检测,删除不符合保留条件的日志分段文件的间隔时间，默认值为300,000，即5分钟。

**log.retention.bytes** 默认值是-1，表示无穷大，参数配置的是日志文件的总大小，而不是单个的日志分段的大小，一个日志文件可以包含多个日志分段。

**broker.id** broker的id

broker.id.generation.enable和reserved.broker.max.id来配合生成新的broker.id。

**broker.id.generation.enable**参数是用来配置是否开启自动生成broker.id的功能，默认情况下为true，即开启此功能。自动生成的broker.id是有一个基准值的，即自动生成的broker.id必须超过这个基准值，这个基准值通过**reserved.broker.max.id**参数配置，默认值为1000，也就是说默认情况下自动生成的broker.id从1001开始。

**replica.lag.time.max.ms** 默认值（10000），即10s，当ISR中的一个follower副本滞后leader副本的时间超过参数值时即判定为副本失效，需要将此follower副本剔出除ISR之外。

**auto.leader.rebalance.enable** 默认true，开启分区自动平衡

如果某个Partition的Leader挂掉，该Partition副本所在的Broker会成为这Partition的Leader，这样会造成Leade分布不均衡，Leader多的Broker读写压力都会比较大。如果开启了auto.leader.rebalance.enable，则当原来挂掉的Broker恢复正常以后，可以夺回Leader。

**举例：**

当 partition 1 的 leader，就是 broker.id = 1 的节点挂掉后，那么 leader 0 或 leader 2 成为 partition 1 的 leader，那么 leader 0 或 leader 2 会管理两个 partition 的读写，性能会下 降，当 leader 1 重新启动后，如果开启了 leader 均衡机制，那么 leader 1 会重新成为 partition 1 的 leader，降低 leader 0 或 leader 2 的负载。

**leader.imbalance.per.broker.percentage** 默认是10%，执行优先副本选举比例阈值，超过就执行优先副本选举

所有Broker中leader的不平衡比例，若是超过这个数值，会对分区进行重新的平衡。Broker的分区不平衡比例=非优先副本的leader个数/分区总数。只有当auto.leader.rebalance.enable设置为true时才有效。建议将auto.leader.rebalance.enable设置为false，避免集群自动执行优先副本选举，因为这样不可控。最好是手动使用path-to-json-file的方式小批量执行优先副本选举。

**leader.imbalance.check.interval.seconds** 默认300，即5分钟，参数是自动执行优先副本的选举动作周期时长

**compression.type**默认值为producer，Broker对消息的压缩格式，支持gzip、snappy、 lz4、zstd和uncompressed。也可以配置成producer，表示以Producer端设置的压缩方式为准。

**delete.topic.enable**（默认值为true） 是否支持使用管理工具删除Topic，默认为true，支持删除。