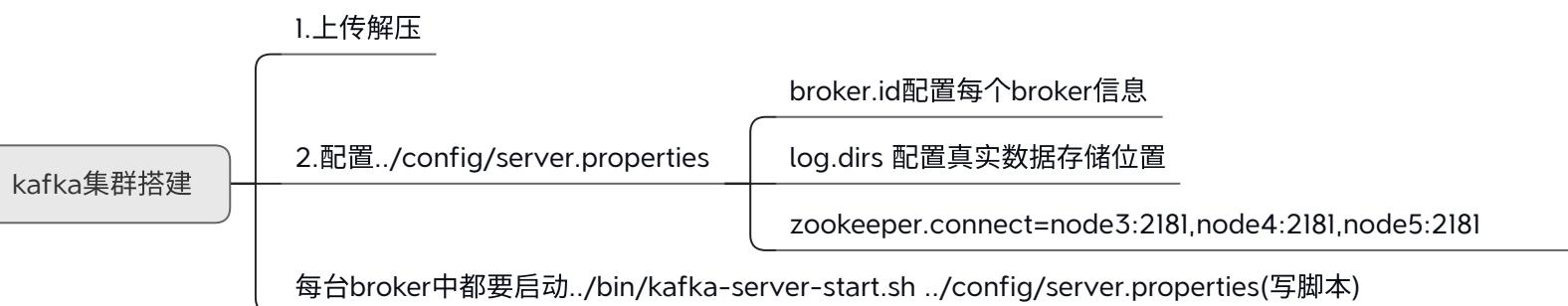


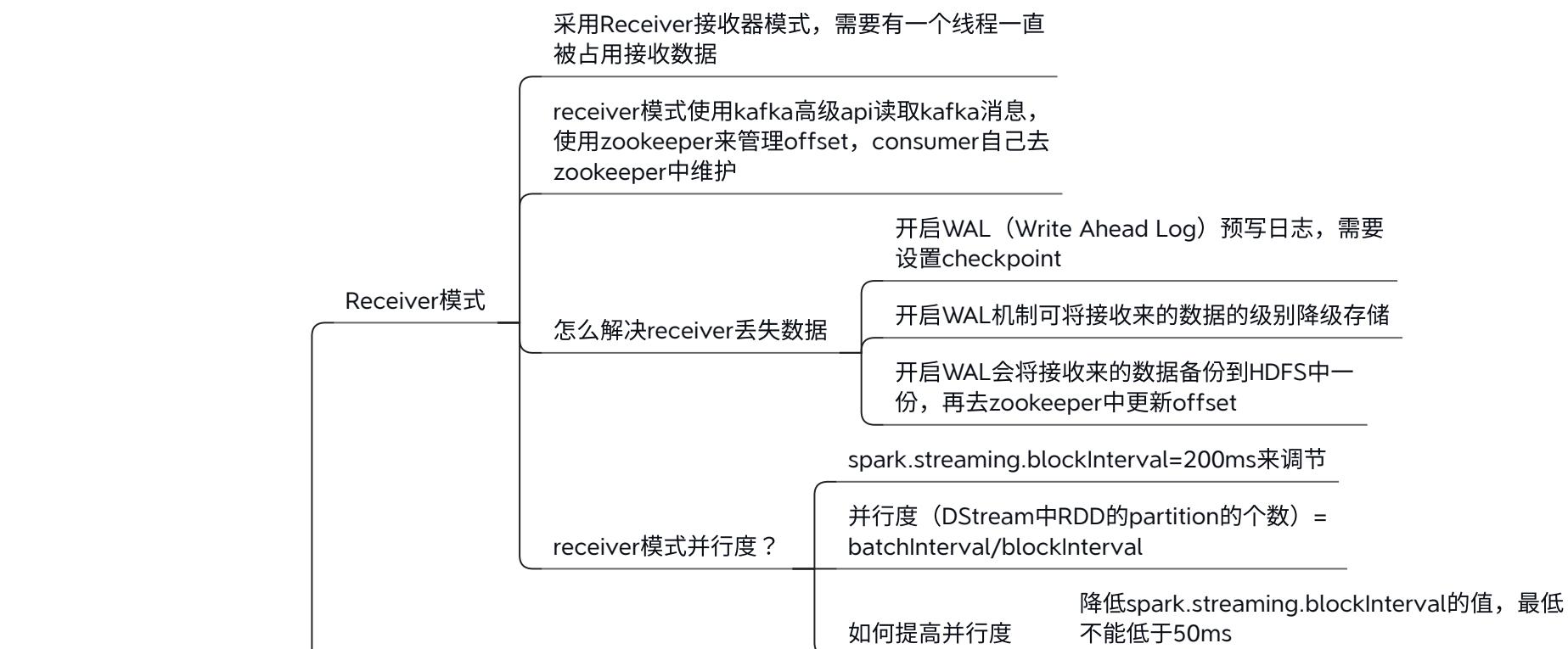
kafka分布式消息系统。将消息持久化到磁盘，默认保存一周

broker	组成kafka集群的节点，负责消息读写，存储，broker之间没有主从关系，一个broker可以管理多个partition
producer	消息生产者，producer自己决定写往topic中的哪个partition, 1) hash 2) 轮询
consumer	消息消费者，消费者自己维护消费者偏移量，consumer有各自的消费者组，不同的组之间消费相同的topic互不影响。相同的消费者组内的消费者，消费同一个topic数据，这个topic中相同的消息只能被消费一次
topic	topic是一类消息/消息队列，topic由partition组成，创建的时候可以指定。
partition	partition是组成topic的单元，在broker中，每个partition有副本，创建topic的时候可以指定。每个partition由一个broker来管理
zookeeper	存储原数据，consumer消费偏移量+broker信息+partition信息+topic信息



当partition的leader挂掉之后，会按照副本优先去寻找新的leader，由新的broker来管理partition
当挂掉的leader重新启动，partition会重新由原来的broker管理

- 1.命令删除topic，当前topic被标记删除，默认一周后被删除
2.清除broker上当前topic的数据目录
3.去zookeeper中删除当前topic的原数据 rmr /broker/topics/topic***
4.zookeeper中删除被标记已删除的topic信息 rmr /admin/delete_topics/topic***



spark自己管理offset，默认在内存中，如果设置了checkpoint，那么offset在checkpoint中也有一份

从checkpoint中恢复原数据：
JavaStreamingContext.getOrCreate(
checkpointDir, JavaStreamingContextFactory)

与读取的kafka中的topic的partition个数一致

如何提高？增加读取的topic的partition个数

在zookeeper中自己管理offset

自己手动管理offset

使用mysql来管理

使用HBase来管理

spark.streaming.backpressure.enabled

spark.streaming.blockInterval

spark.streaming.receiver.maxRate

spark.streaming.receiver.writeAheadLogEnable

spark.streaming.stopGracefullyOnShutdown

spark.streaming.kafka.maxRatePerPartition