- kafka
  - kafaka分布式消息系统。将消息持久化到磁盘，默认保存一周
  - 概念
    - 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
  - kafka集群搭建
    - 1.上传解压
    - 2.配置../config/server.properties
      - broker.id配置每个broker信息
      - log.dirs 配置真实数据存储位置
      - zookeeper.connect=node3:2181,node4:2181,node5:2181
    - 每台broker中都要启动../bin/kafka-server-start.sh ../config/server.properties(写脚本)
  - 命令
    - 查看集群有哪些topic：./kafka-topics.sh --zookeeper node3:2181,node4,node5 --list
    - 创建topic：./kafka-topics.sh --zookeeper node3:2181,node4,node5 --create --topic t0425 --partitions 3 --replication-factor 3
    - 控制台当做生产者：./kafka-console-producer.sh --broker-list node1:9092,node2:9092,node3:9092 --topic t0425
    - 控制台当做消费者：./kafka-console-consumer.sh --zookeeper node3:2181,node4:2181,node5:2181 --topic t0425
    - 查看topic的描述信息：./kafka-topics.sh --zookeeper node3:2181,node4,node5 --describe --topic t0425
    - 查看producer生产消息的最大位置：./kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list node1:9092 --topic t0425 --time -1
      - -1表示查询这个topic各个分区当前最大的消息位移值（注意这里的位移不是consumer端的位移，而是指消息在每个分区的位置）
      - -2表示获取当前各个分区的最小位移
    - 删除topic：./kafka-topics.sh --zookeeper node3:2181,node4:2181,node5:2181 --delete --topic t0425
  - kafka的leader均衡机制
    - 当partition的的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
  - 删除topic
    - 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***
  - SparkStreaming+kafka
    - Receiver模式
      - 采用Receiver接收器模式，需要有一个线程一直被占用接收数据
      - receiver模式使用kafka高级api读取kafka消息，使用zookeeper来管理offset，consumer自己去zookeeper中维护
      - 怎么解决receiver丢失数据
        - 开启WAL（Write Ahead Log）预写日志，需要设置checkpoint
        - 开启WAL机制可将接收来的数据的级别降级存储
        - 开启WAL会将接收来的数据备份到HDFS中一份，再去zookeeper中更新offset
      - receiver模式并行度？
        - spark.streaming.blockInterval=200ms来调节
        - 并行度（DStream中RDD的partition的个数）= batchInterval/blockInterval
        - 如何提高并行度：降低spark.streaming.blockInterval的值，最低不能低于50ms
    - Direct模式
      - Direct模式是将kafka看成存储数据的地方，直接去读取，使用的是kafka中低阶api消费消息
      - 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)
      - Direct模式并行度
        - 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
      - 自己手动管理offset
        - 在zookeeper中自己管理offset
        - 使用mysql来管理
        - 使用HBase来管理
    - 参数
      - saprk.streaming.backpressure.enabled
      - spark.streaming.blockInterval
      - spark.streaming.receiver.maxRate
      - spark.streaming.receiver.writeAheadLogEnable
      - spark.streaming.stopGracefullyOnShutdown
      - spark.streaming.kafka.maxRatePerPartition