Kafka搭建

<https://www.cnblogs.com/biehongli/p/7767710.html>

<https://kafka.apache.org/quickstart> //官方快速部署文档

<https://www.cnblogs.com/hei12138/p/7805475.html>

kafka的前言知识：

1：Kafka是什么？

在流式计算中，Kafka一般用来缓存数据，Storm通过消费Kafka的数据进行计算。kafka是一个生产-消费模型。**Producer：**生产者，只负责数据生产，生产者的代码可以集成到任务系统中。   
数据的分发策略由producer决定，默认是defaultPartition  Utils.abs(key.hashCode) % numPartitions  
**Broker：**当前服务器上的Kafka进程,俗称拉皮条。只管数据存储，不管是谁生产，不管是谁消费。  
在集群中每个broker都有一个唯一brokerid，不得重复。  
**Topic:** 目标发送的目的地，这是一个逻辑上的概念，落到磁盘上是一个partition的目录。partition的目录中有多个segment组合(index,log)  
 一个Topic对应多个partition[0,1,2,3]，一个partition对应多个segment组合。一个segment有默认的大小是1G。  
每个partition可以设置多个副本(replication-factor 1),会从所有的副本中选取一个leader出来。所有读写操作都是通过leader来进行的。  
特别强调，和mysql中主从有区别，mysql做主从是为了读写分离，在kafka中读写操作都是leader。  
**ConsumerGroup：**数据消费者组，ConsumerGroup可以有多个，每个ConsumerGroup消费的数据都是一样的。  
可以把多个consumer线程划分为一个组，组里面所有成员共同消费一个topic的数据，组员之间不能重复消费。

2：Apache Kafka是一个开源消息系统，由Scala写成。是由Apache软件基金会开发的一个开源消息系统项目。

3：Kafka是一个分布式消息队列：生产者、消费者的功能。它提供了类似于JMS的特性，但是在设计实现上完全不同，此外它并不是JMS规范的实现。

4：Kafka对消息保存时根据Topic进行归类，发送消息者称为Producer,消息接受者称为Consumer,此外kafka集群有多个kafka实例组成，每个实例(server)成为broker。

5：无论是kafka集群，还是producer和consumer都依赖于zookeeper集群保存一些meta信息，来保证系统可用性。

6：Kafka核心组件：Topic ：消息根据Topic进行归类；Producer：发送消息者；Consumer：消息接受者；broker：每个kafka实例(server)；Zookeeper：依赖集群保存meta信息。   
7：消息系统的核心作用就是三点：解耦，异步和并。

8：kafka生产数据时的分组策略？

默认是defaultPartition  Utils.abs(key.hashCode) % numPartitions。  
上文中的key是producer在发送数据时传入的，produer.send(KeyedMessage(topic,myPartitionKey,messageContent))。

9：kafka如何保证数据的完全生产？  
ack机制：broker表示发来的数据已确认接收无误，表示数据已经保存到磁盘。  
    0：不等待broker返回确认消息。  
    1：等待topic中某个partition leader保存成功的状态反馈。  
  -1：等待topic中某个partition 所有副本都保存成功的状态反馈。

10：broker如何保存数据？  
在理论环境下，broker按照顺序读写的机制，可以每秒保存600M的数据。主要通过pagecache机制，尽可能的利用当前物理机器上的空闲内存来做缓存。  
当前topic所属的broker，必定有一个该topic的partition，partition是一个磁盘目录。partition的目录中有多个segment组合(index,log)。

11：如何保证kafka消费者消费数据是全局有序的？  
　　 伪命题,  
　　 如果要全局有序的，必须保证生产有序，存储有序，消费有序。由于生产可以做集群，存储可以分片，消费可以设置为一个consumerGroup，要保证全局有序，就需要保证每个环节都有序。只有一个可能，就是一个生产者，一个partition，一个消费者。这种场景和大数据应用场景相悖。

**快速搭建kafka**

前提环境：安装jdk

**[第1步：下载代码](https://kafka.apache.org/quickstart" \l "quickstart_download)**

[下载](https://www.apache.org/dyn/closer.cgi?path=/kafka/2.0.0/kafka_2.11-2.0.0.tgz" \o "卡夫卡下载) 2.0.0版本并解压缩它。

shell > tar zxf kafka\_2.12-2.0.0.tar.gz

shell > cd kafka\_2.12-2.0.0

**[第2步：启动服务器](https://kafka.apache.org/quickstart" \l "quickstart_startserver)**

Kafka使用[ZooKeeper，](https://zookeeper.apache.org/)因此如果您还没有ZooKeeper服务器，则需要先启动它。您可以使用与kafka一起打包的便捷脚本来获取快速且脏的单节点ZooKeeper实例

shell > bin/zookeeper-server-start.sh config/zookeeper.properties

如需在后台启动 在开头加nohup 末尾加 & 例：nohup bin/xxx &

注释：你也可以下载 zookeeper的源码包来启动zookeeper不用默认kafka里的zookeeper启动脚本

Shell > tar zxf zookeeper-3.4.12.tar.gz

Shell > cd zookeeper-3.4.12

Shell > bin/zkServer.sh start

现在启动Kafka服务器：

Shell > bin/kafka-server-start.sh config/server.properties

注释：先启动zookeeper，在启动kafka

**[第3步：创建主题](https://kafka.apache.org/quickstart" \l "quickstart_createtopic)**

让我们创建一个名为“test”的主题，它只包含一个分区，只有一个副本：

shell > bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

如果我们运行list topic命令，我们现在可以看到该主题：

shell > bin/kafka-topics.sh --list --zookeeper localhost:2181

test

或者，您也可以将代理配置为在发布不存在的主题时自动创建主题，而不是手动创建主题。

**[第4步：发送一些消息](https://kafka.apache.org/quickstart" \l "quickstart_send)**

Kafka附带一个命令行客户端，它将从文件或标准输入中获取输入，并将其作为消息发送到Kafka集群。默认情况下，每行将作为单独的消息发送。

运行生产者，然后在控制台中键入一些消息以发送到服务器。

shell > bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test

> This is a message

**[第5步：启动消费者](https://kafka.apache.org/quickstart" \l "quickstart_consume)**

Kafka还有一个命令行使用者，它会将消息转储到标准输出。

shell > bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

This is a message

如果您在不同的终端中运行上述每个命令，那么您现在应该能够在生产者终端中键入消息并看到它们出现在消费者终端中。