# 文章简介

|  |
| --- |
|  |

# kafka简介

|  |
| --- |
| 1. kafka基本结构 2. kafka基本概念 3. kafka设计概念 4. kafka特性   ----------------------------------------------------------------------------------------------------------------------    **1，kafka基本结构**  Kafka是一个开源的，分布式的，高吞吐量的消息系统。随着Kafka的版本迭代，日趋成熟。大家对它的使用也逐步从日志系统衍生到其他关键业务领域。特别是其超高吞吐量的特性，在互联网领域，使用越来越广泛，生态系统也越来的完善。同时，其设计思路也是其他消息中间件重要的设计参考。  Kafka原先的开发初衷是构建一个处理海量日志的框架，基于高吞吐量为第一原则，所以它对消息的可靠性以及消息的持久化机制考虑的并不是特别的完善。0.8版本后，陆续加入了一些复制、应答和故障转移等相关机制以后，才可以让我们在其他关键性业务中使用。  Kafka的运行架构如下图，各组件之间通过TCP协议通信： |
| **2，kafka基本概念**  **（1）Topic：**  主题，或者说是一类消息。类似于RabbitMQ中的queue。可以理解为一个队列。  **（2）Broker：**  一个Kafka服务称之为Broker。Kafka可以集群部署，每一个Kafka部署就是一个Broker。  **（3）Producer & Consumer：**  生产者和消费者。一般消息系统都有生产者和消费者的概念。生产者产生消息，即把消息放入Topic中，而消费者则从Topic中获取消息处理。一个生产者可以向多个Topic发送消息，一个消费者也可以同时从几个Topic接收消息。同样的，一个Topic也可以被多个消费者来接收消息。  **（4）Partition：**  分区，或者说分组。分组是Kafka提升吞吐量的一个关键设计。这样可以让消费者多线程并行接收消息。创建Topic时可指定Parition数量。一个Topic可以分为多个Partition，也可以只有一个Partition。每一个Partition是一个有序的，不可变的消息序列。每一个消息在各自的Partition中有唯一的ID。这些ID是有序的。称之为offset，offset在不同的Partition中是可以重复的，但是在一个Partition中是不可能重复的。越大的offset的消息是最新的。Kafka只保证在每个Partition中的消息是有序的，就会带来一个问题，即如果一个Consumer在不同的Partition中获取消息，那么消息的顺序也许是和Producer发送到Kafka中的消息的顺序是不一致的。这个在后续会讨论。      如果是多Partition，生产者在把消息放到Topic中时，可以决定放到哪一个Patition。这个可以使用简单的轮训方法，也可以使用一些Hash算法。  一个Topic的多个Partition可以分布式部署在不同的Server上，一个Kafka集群。配置项为：num.partitions，默认是1。每一个Partition也可以在Broker上复制多分，用来做容错。详细信息见下面创建Topic一节。    **（5）Consumer Group：**  顾名思义，定义了一组消费者。一般来说消息中间件都有两种模式：队列模式和发布订阅模式。队列模式及每一个消息都会给其中一个消费者，而发布订阅模式则是每个消息都广播给所有的消费者。Kafka就是使用了Consumer Group来实现了这两种模式。  如果所有的消费者都是同一个Consumer Group的话，就是队列模式，每个消息都会负载均衡的分配到所有的消费者。  如果所有的消息者都在不同的Consumer Group的话，就是发布订阅模式，每个消费者都会得到这个消息。  下图是一个Topic，配置了4个Patition，分布在2个Broker上。由于有2个Consumer Group，Group A和Group B都可以得到P0-P3的所有消息，是一个订阅发布模式。两个Group中的Consumer则负载均衡的接收了这个Topic的消息。如果Group中的Consumer的总线程数量超过了Partition的数量，则会出现空闲状态。      **（6）Zookeeper：**  Kafka的运行依赖于Zookeeper。Topic、Consumer、Patition、Broker等注册信息都存储在ZooKeeper中。  **3，kafka设计概念**    **4，kafka特性**  （1）消息持久化    （2）高吞吐量    （3）扩展性    （4）多客户端支持    （5）kafka Streams  kafka在0.10之后版本引入kafka Streams。kafka Streams是一个用java语言实现的用于流处理的jar文件。  （6）轻量级 |

# Windows环境下zookeeper安装配置

|  |
| --- |
| 注：win8/win10环境下，启动报错：javahome is not set需要修改bin/zkEnv.cmd中的javahome配置到自己的java/bin目录，如下示例（注意修改为自己的javahome）：  @echo off REM Licensed to the Apache Software Foundation (ASF) under one or more REM contributor license agreements. See the NOTICE file distributed with REM this work for additional information regarding copyright ownership. REM The ASF licenses this file to You under the Apache License, Version 2.0 REM (the "License"); you may not use this file except in compliance with REM the License. You may obtain a copy of the License at REM REM <http://www.apache.org/licenses/LICENSE-2.0> REM REM Unless required by applicable law or agreed to in writing, software REM distributed under the License is distributed on an "AS IS" BASIS, REM WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. REM See the License for the specific language governing permissions and REM limitations under the License.  REM for sanity sake assume Java 1.6 REM see: <http://java.sun.com/javase/6/docs/technotes/tools/windows/java.html>  REM add the zoocfg dir to classpath set CLASSPATH=%ZOOCFGDIR%  REM make it work in the release SET CLASSPATH=%~dp0..\\*;%~dp0..\lib\\*;%CLASSPATH%  REM make it work for developers SET CLASSPATH=%~dp0..\build\classes;%~dp0..\build\lib\\*;%CLASSPATH%  **set JAVA=D:\java\jdk1.8.0\_77\bin\java  set JAVA\_HOME=D:\java\jdk1.8.0\_77**  set ZOOCFG=%ZOOCFGDIR%\zoo.cfg set ZOOCFGDIR=%~dp0%..\conf set ZOO\_LOG\_DIR=%~dp0%..\logs set ZOO\_LOG4J\_PROP=INFO,CONSOLE @REM setup java environment variables  if not defined JAVA\_HOME (  echo Error: JAVA\_HOME is not set.  goto :eof )  if not defined JAVA (  echo Error: **----"%JAVA\_HOME%"--- is set.but not found JAVA** goto :eof )  参考：<https://www.cnblogs.com/china-baizhuangli/p/8920776.html>  ----------------------------------------------------------------------------------------------------------------------  1 |

# Windows环境下Kafka安装部署

|  |
| --- |
|  |

# Windows环境下Kafka基本命令操作

|  |
| --- |
| 三，kafka的命令行操作：  **1，创建topic**  1.1 新建cmd 窗口 ， cd 到kafka目录bin/windows目录下（D:\Program Files\kafka\_2.12-2.0.0\bin\windows）。  1.2执行: kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test  注：其中 单体环境只有一台broker，最后一个复本个数只能为1 ， test为topic名称。    **2，启动生产者Producer**  2.1 新建cmd窗口， cd 到kafka安装目录bin/windows目录下，  2.2 执行： kafka-console-producer.bat --broker-list localhost:9092 --topic test    **3，启动消费者consumer**  3.1 新建cmd窗口， cd 到kafka安装目录bin/windows目录下，  3.2执行：  kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test  注：老版本的consumer启动命令与新版本不同：  老版本如下：  kafka-console-consumer.bat -- localhost:2181 --topic test  到此kafka单机windows环境安装和启动以及创建topic，启动生产者消费者的操作完成，可以尝试由Producer发送消息到consumer。  计算机生成了可选文字: 命令_毳f毳 sdf asd fsa asdf asd fas asd f f sadf asdf asd 0 asd f dsafasdf --bootstrap- S巳 命令_kafka-console-producer.bat--broker-listchos亡992 >as >df >asdf >as >df >asd >fas >df >asd 〉f >sd 〉f >sadf >sa >df >asdf >asd 〉f >asd 〉f >dsafasdf --toplctest |

# Springboot框架中Kafka整合与快速启动

|  |
| --- |
| 1. **准备工作**   Kafka服务版本为 kafka\_2.12-2.0.0 (Scala)， 也就是2.0.0  SpringBoot版本：1.5.9.RELEASE  提前启动zookeeper，kafka，并且创建一个Topic   1. **快速构建项目** 2. **新建springboot项目**      1. **添加maven依赖**   <dependency>  <groupId>org.springframework.kafka</groupId>  <artifactId>spring-kafka</artifactId>  </dependency>   1. **具体代码实现**   **Application.yml**    KafkaApplication    TestKafkaProducerController    TestConsumer    源码地址：  <https://github.com/JDZW2018/springCloudDemo.git> |

# Linux环境下zookeeper集群安装配置

|  |
| --- |
|  |

# Linux环境下Kafka集群安装配置

|  |
| --- |
|  |

# Kafka核心组件

|  |
| --- |
|  |

# Kafka核心流程分析

|  |
| --- |
|  |

# Kafka命令行操作

|  |
| --- |
|  |

# Kafka API编程

|  |
| --- |
|  |

# Kafka Streams

|  |
| --- |
|  |

# Kafka数据采集应用

|  |
| --- |
|  |

# Kafka与ELK整合应用

|  |
| --- |
|  |

# Kafka与Spark整合应用

|  |
| --- |
|  |