# 常用概念

<https://blog.csdn.net/u010020099/article/details/82290403>

名词 解释

Producer 消息的生成者

Consumer 消息的消费者

ConsumerGroup 消费者组，可以并行消费Topic中的partition的消息

Broker 缓存代理，Kafka集群中的一台或多台服务器统称broker.

Topic Kafka处理资源的消息源(feeds of messages)的不同分类，逻辑上的分组。

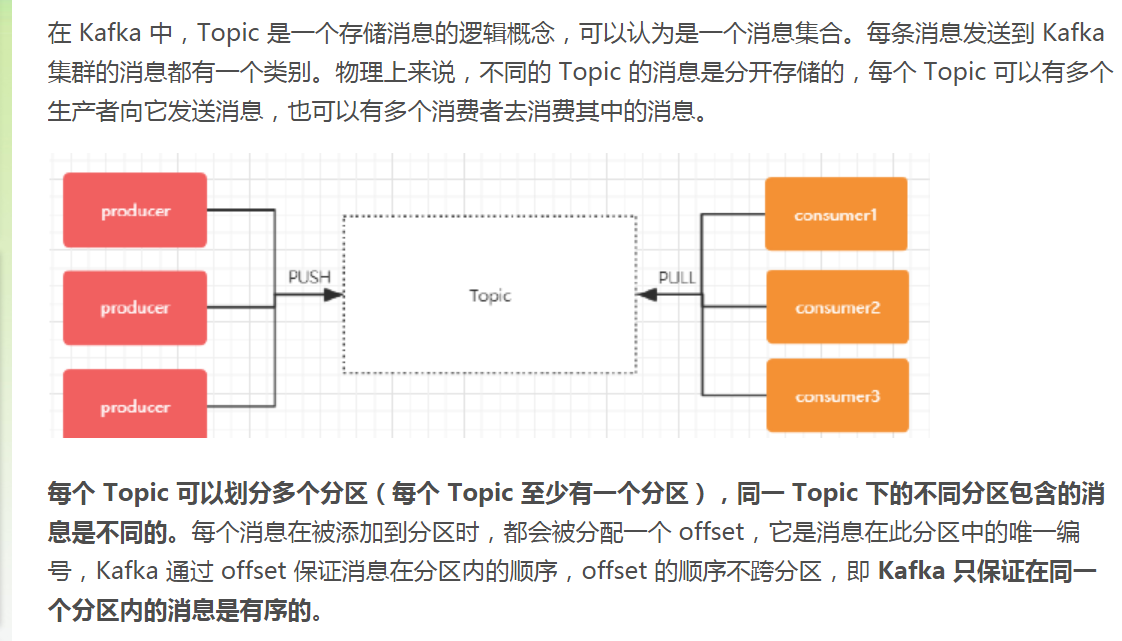
Partition Topic物理上的分组，一个topic可以分为多个partion,每个partion是一个有序的队列。partion中每条消息都会被分配一个 有序的Id(offset)，用以标识消息的顺序和唯一性。

Message 消息，是通信的基本单位，每个producer可以向一个topic（主题）发布一些消息

Producers 消息和数据生成者，向Kafka的一个topic发布消息的 过程叫做producers

Consumers 消息和数据的消费者，订阅topic并处理其发布的消费过程叫做consumers





1. 注意kafka压缩包不要下载源码包，而是下载Binary downloads。
2. Kafka启动之前，必须先启动zookeeper.
3. Kafka的分布式服务器是有主从区别的，通过命令可以查看，

./bin/zkServer.sh status 启动服务，查看服务状态

常见的两种状态是

Mode: follower 从服务器

Mode: leader 主服务器

还有一个模式是Mode: standalone 单机模式。

注：至少2服服务器启动，才能查看主从状态！！

# kafka+zookeeper在linux上的部署

1. kafka依赖于zookeeper,所以先要部署zookeeper.

先把zookeeper的解压文件放到/usr/local 上，修改conf文件夹下的“zoo\_sample.cfg”文件为zoo.cfg,内容如下修改：

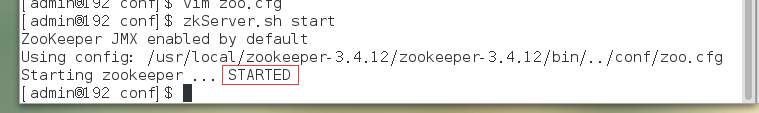
添加server.1=服务器ip:2888:3888 表示zookeeper服务器的地址，如果需要添加多个zookeeper服务器则添加其他服务器的ip ,像server.2=xxx，server.3=xxx类推。

然后修改dataDir=/usr/local/zookeeper-3.4.12/zookeeper-3.4.12\logs,这个路径表示是zookeeper日志等数据的存储地址，所以我们要创建logs这个文件夹。

然后再logs文件夹中添加myid文件，执行linux命令

echo 1 >> myid,这个文件用来标识zookeeper服务器的唯一性，所以如果你需要添加多个zookeeper服务器，需要创建值不同的myid文件 如 echo 2 >> myid。

修改完，进入bin文件夹下执行zkServer.sh start 启动zookeeper，当然也可以配置环境变量后在任何路径下执行zkServer.sh start。



显示如上图表示启动成功。

2.部署kafka，先把解压文件移到/usr/local，修改config文件下的server.properties。修改

broker.id为之前zookeeper中创建myid设置的值，我们之前设置的是1.

修改log.dirs=/user/local/kafka\_2.11-2.4.0/logs 并在相应的路径创建logs文件夹

修改listeners=PLAINTEXT://192.168.1.6:9092,其中的ip为kafka服务器的ip,默认使用9092端口。

修改zookeeper.connect=zookeeper服务器ip:2181,表示kafka通过2181端口连接zookeeper，如果需要启动多台zookeeper服务器，则在后面用逗号分隔其他服务器的ip+端口。到此所有配置结束。

3.kafka常用命令

kafka-server-start.sh server.properties 启动kafka

kafka-topics.sh --create --zookeeper 192.168.44.129:2181 --replication-factor 1 --partitions 3 --topic test

创建名为test的topic.有3个分区，副本因子(replication-factor)为1,值每个分区1个副本。

这里的ip填zookeeper服务所在的ip,zookeeper一般使用2181端口。

删除topic



测定的删除topic很复杂，具体参考

<https://blog.csdn.net/qq_35803514/article/details/87164081>

<https://www.cnblogs.com/ulysses-you/p/10232746.html>

kafka-topics.sh --list --zookeeper ip:2181

查看所有的topic，ip为zookeeper服务器的ip

kafka-console-producer.sh --broker-list ip:9092 --topic test

生产者生产消息(用localhost没用) kafka服务器的ip，其实一般来说单服务器的话kafka和zookeeper都放在一个服务器上。

kafka-console-consumer.sh --bootstrap-server ip:9092 --topic test --from-beginning 消费者消费消息（查看消息） 其中bootstrap-server就是kafka的broker节点 ip是kafka服务器的ip

## 清空kafka+zookeeper的所有数据

<https://blog.csdn.net/qq_35803514/article/details/87164081>

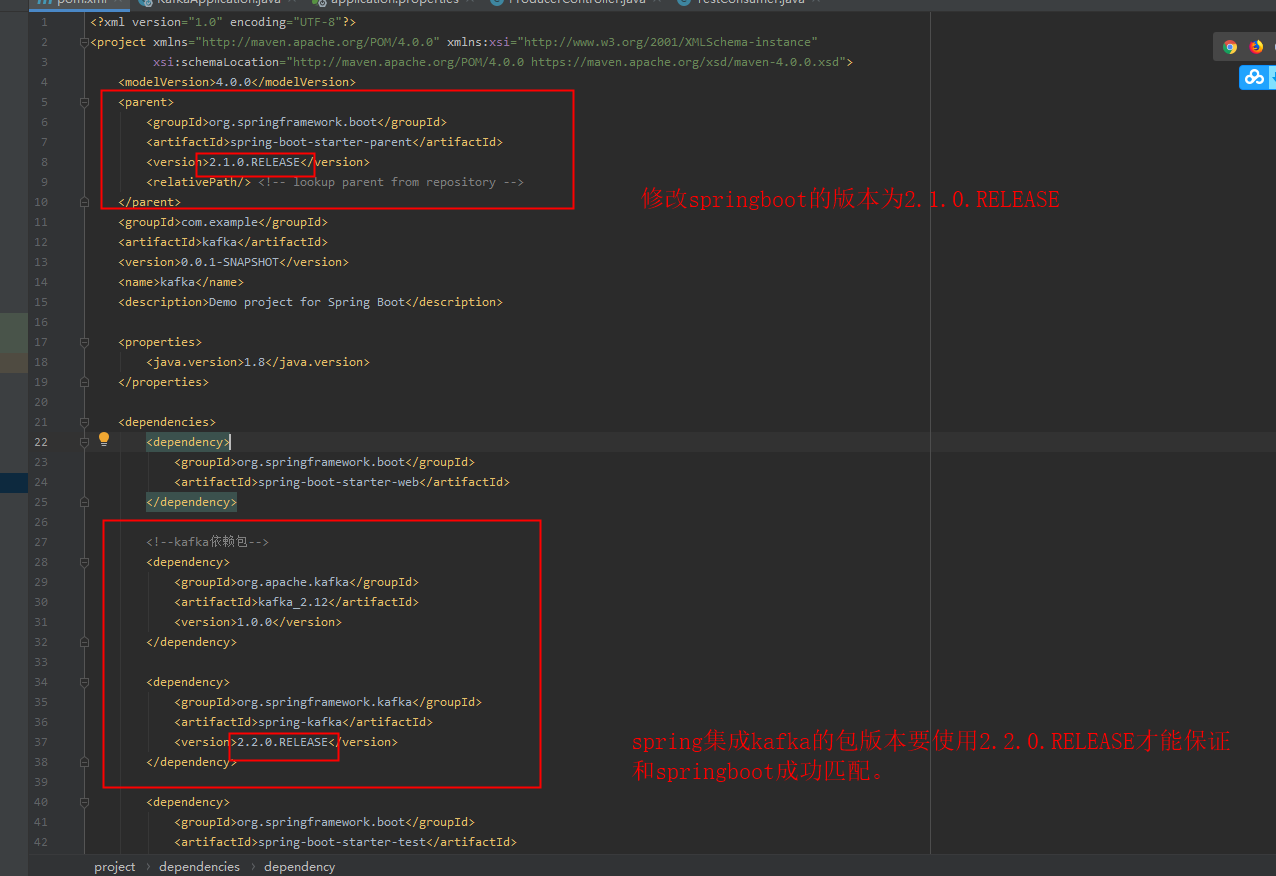
<https://www.cnblogs.com/sherrykid/p/5813148.html>

1. 删除zookeeper下的所有数据。 可用可视化工具实现
2. 清空kafka的logs文件夹。 手动清空即可。

# Kafka+zookeeper+springboot集成

先确保kafka服务已经成功启动

1. idea创建一个springboot项目。
2. 修改pom文件如下



依赖的源码如下

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <!--kafka依赖包-->  
 <dependency>  
 <groupId>org.apache.kafka</groupId>  
 <artifactId>kafka\_2.12</artifactId>  
 <version>1.0.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.kafka</groupId>  
 <artifactId>spring-kafka</artifactId>  
 <version>2.2.0.RELEASE</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 <exclusions>  
 <exclusion>  
 <groupId>org.junit.vintage</groupId>  
 <artifactId>junit-vintage-engine</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
</dependencies>

1. application.properties配置如下

server.port=3370  
spring.application.name=qhplan-application  
spring.kafka.bootstrap-servers=192.168.1.6:9092  
spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer  
spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer  
spring.kafka.producer.batch-size=4096  
spring.kafka.producer.buffer-memory=40960  
spring.kafka.consumer.group-id=test  
spring.kafka.consumer.enable-auto-commit=true  
spring.kafka.consumer.auto-commit-interval=100  
spring.kafka.consumer.key-serializer=org.apache.kafka.common.serialization.StringDeserializer  
spring.kafka.consumer.value-serializer=org.apache.kafka.common.serialization.StringDeserializer  
spring.kafka.consumer.bootstrap-servers=192.168.1.6:9092  
spring.kafka.consumer.auto-offset-reset=earliest

Demo的话把kafka服务器的端口和ip配对就行了，我的是192.168.1.6:9092，对应kafka配置文件server.properties中的listeners属性，其他的参数具体可参考。

<https://blog.csdn.net/fenglibing/article/details/82117166>

1. 添加消息生成者类

@RestController  
@RequestMapping("/kafka")  
public class ProducerController {  
 protected final Logger logger = LoggerFactory.*getLogger*(this.getClass());  
  
 @Autowired  
 private KafkaTemplate<String, String> kafkaTemplate;  
  
 @RequestMapping(value = "/send", method = RequestMethod.*GET*)  
 public String sendKafka(@RequestParam("message") String message) {  
 try {  
 logger.info("kafka的消息={}", message);  
 System.*out*.println(kafkaTemplate);  
 kafkaTemplate.send("test", "key", message);  
 logger.info("发送kafka成功.");  
 return "successs";  
 } catch (Exception e) {  
 logger.error("发送kafka失败", e);  
 return "failure";  
 }  
 }  
  
}

KafkaTemplate.send(“test”,”key”,message);可以生产一条消息，其中第一个参数test表示生产的消息存于名为test的topic中,第二个参数+offset唯一标识一个topic中的message,message就是生产的消息，详细的最后展示消息生产时会介绍。

1. 创建消费类

@Component  
public class TestConsumer {  
 @KafkaListener(topics = "test")  
 public void listen(ConsumerRecord<?, ?> record) throws Exception {  
 System.*out*.printf("topic = %s, offset = %s, value = %s \n", record.topic(), record.key(), record.value());  
 }  
}

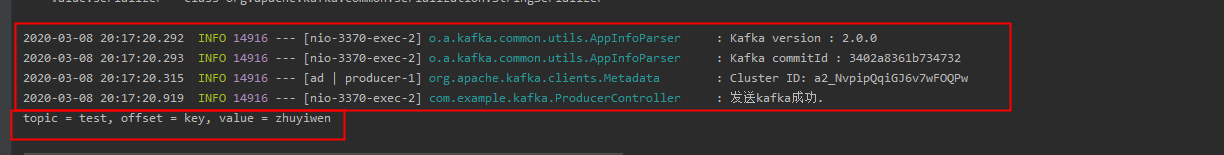
使用方式就是通过KafkaListener注解，topics标识出消费的topic的名称即可，我们生产类中用的事topic。这里注意了，topic是通过kafka命令在kafka服务器上创建的，即

kafka-console-producer.sh --broker-list 192.168.1.6:9092 --topic test，而不是在

执行KafkaTemplate.send(“test”,”key”,message)时创建的。

1. 启动项目，访问localhost:3370/kafka/send?message=zhuyiwen,页面显示success.

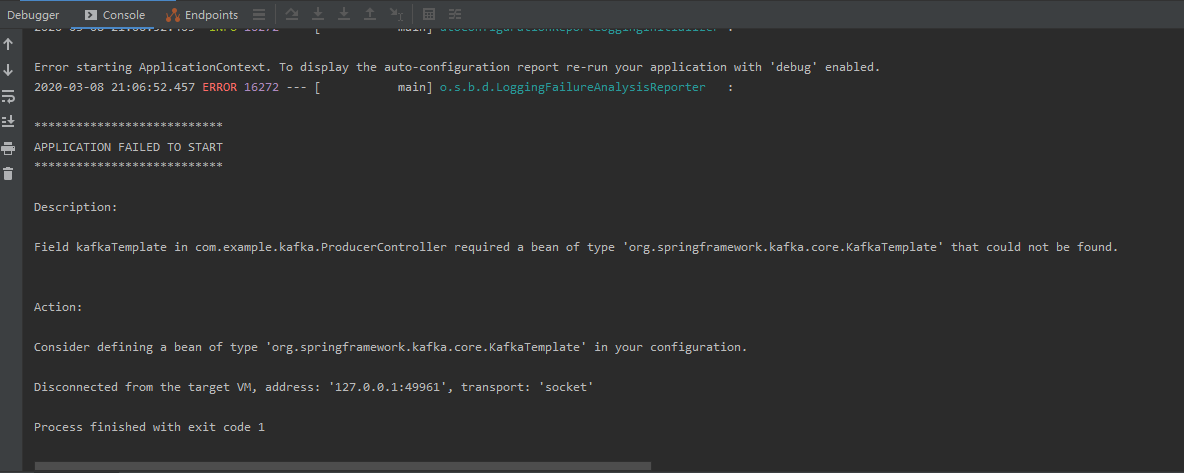
控制台显示如下：



上面的框是生产消息的输出结果，下面的框就是消费类消费的输入结果。

1. 如果不成功可能的原因：

如果报下图错误



是因为springboot版本跟kafka版本不匹配导致，文档中设置的版本亲测可以启动成功，springboot和kafka版本对应详细如下：

<https://blog.csdn.net/lzx1991610/article/details/100777040>

如果启动成功，但是请求url后控制台报错

Connection to node -1 could not be established. Broker may not be available

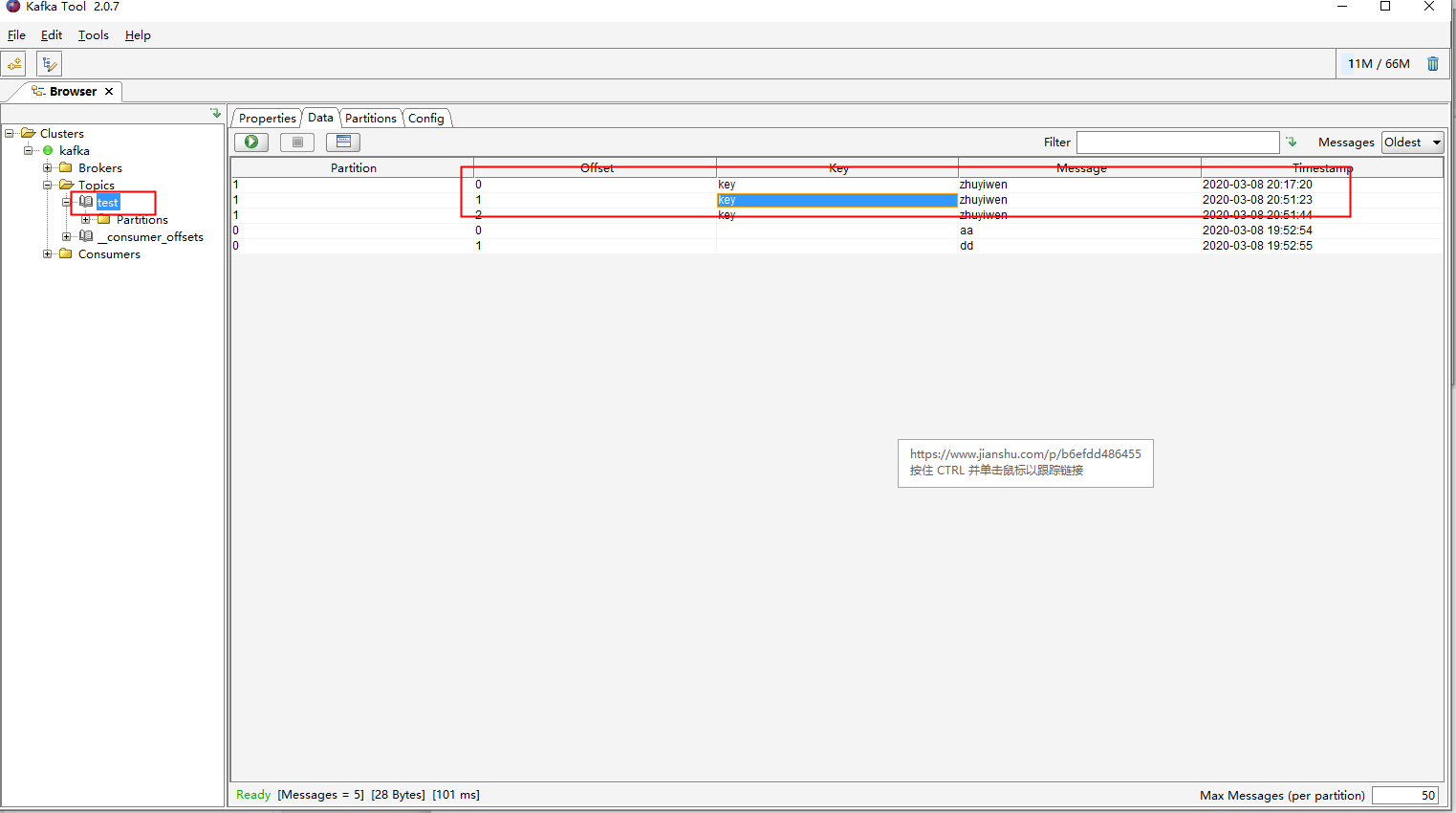
原因就是没有连上kafka的服务器，可能原因是application.properties中的kafka的ip和端口号填错了，请保证和kafka服务器上的server.properties文件中的listeners属性一致。

也可能是linux相应的端口没开,即访问kafka的2181端口。

还可能的原因是kafka服务没有启动成功。

# 可视化Kafka Tool

1. 连接服务后可在对应的test Topic中看到生产的数据

如果message是乱码的话，可以通过以下方式解决。

<https://www.jianshu.com/p/b6efdd486455>

可以看到topic中offset和key唯一标识一条消息。

# 参考博客

<https://blog.csdn.net/nullbull/article/details/82379589>

<https://www.jianshu.com/p/d3e963ff8b70>

<https://blog.csdn.net/Black1499/article/details/90474929>

<https://blog.csdn.net/fenglibing/article/details/82117166>

<https://blog.csdn.net/wlddhj/article/details/88585281>

<https://blog.csdn.net/sndayYU/article/details/100557209>