https://blog.csdn.net/u010283894/article/details/77106159

1.简介

  Kafka是一种高吞吐量的分布式发布订阅消息系统。详细介绍可查阅官网：kafka官网

2.环境搭建

2.1 安装JDK

    下载地址：jre下载

   有关jdk的安装不再赘述。

2.2 安装Zookeeper

     下载地址：https://zookeeper.apache.org/releases.html

    下载后，解压放在目录D:\bigdata（本文所用的目录）下，关于zookeeper以及kafka的目录，路径中最好不要出现空格，比如D:\Program Files，尽量别用，运行脚本时会有问题。

①进入zookeeper的相关设置所在的文件目录，例如本文的：D:\bigdata\zookeeper-3.4.10\conf

②将"zoo\_sample.cfg"重命名为"zoo.cfg"

③打开zoo.cfg(至于使用什么编辑器，根据自己喜好选即可)，找到并编辑：

dataDir=/tmp/zookeeper  to  D:/bigdata/zookeeper-3.4.10/data或 D:\\bigdata\\zookeeper-3.4.10\\data(路径仅为示例，具体可根据需要配置)

这里注意，路径要么是"/"分割，要么是转义字符"\\"，这样会生成正确的路径(层级，子目录)。

④与配置jre类似，在系统环境变量中添加：

    a.系统变量中添加ZOOKEEPER\_HOME=D:\bigdata\zookeeper-3.4.10

    b.编辑系统变量中的path变量，增加%ZOOKEEPER\_HOME%\bin

⑤在zoo.cfg文件中修改默认的Zookeeper端口(默认端口2181)

这是本文最终的zoo.cfg文件的内容：

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=D:/bigdata/zookeeper-3.4.10/data

#dataDir=D:\\bigdata\\zookeeper-3.4.10\\data

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

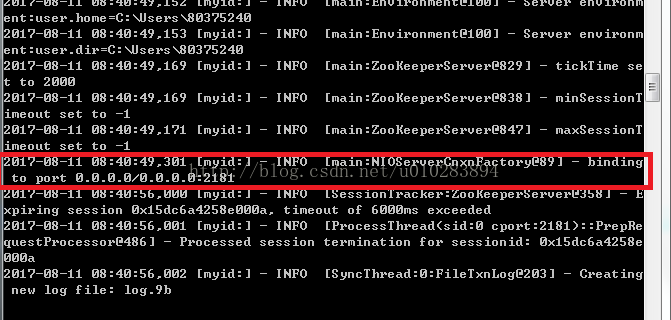
#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

⑥打开cmd窗口，输入zkserver，运行Zookeeper，运行结果如下：



恭喜，Zookeeper已经安装完成，已在2181端口运行。

2.3 安装kafka

    下载地址：http://kafka.apache.org/downloads

    要下载Binary downloads这个类型，不要下载源文件，这种方便使用。下载后，解压放在D:\bigdata目录下。

①进入kafka配置文件所在目录，D:\bigdata\kafka\_2.11-0.9.0.1\config

②编辑文件"server.properties"，找到并编辑：

 log.dirs=/tmp/kafka-logs  to  log.dirs=D:/bigdata/kafka\_2.11-0.9.0.1/kafka-logs 或者 D:\\bigdata\\kafka\_2.11-0.9.0.1\\kafka-logs

同样注意：路径要么是"/"分割，要么是转义字符"\\"，这样会生成正确的路径(层级，子目录)。错误路径情况可自行尝试，文件夹名为这种形式：bigdatakafka\_2.11-0.9.0.1kafka-logs

③在server.properties文件中，zookeeper.connect=localhost:2181代表kafka所连接的zookeeper所在的服务器IP以及端口，可根据需要更改。本文在同一台机器上使用，故不用修改。

④kafka会按照默认配置，在9092端口上运行，并连接zookeeper的默认端口2181。

2.4 运行kafka

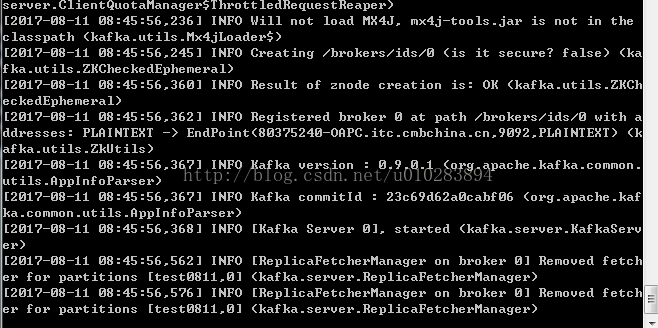
提示：请确保启动kafka服务器前，Zookeeper实例已经在运行，因为kafka的运行是需要zookeeper这种分布式应用程序协调服务。

①进入kafka安装目录D:\bigdata\kafka\_2.11-0.9.0.1

②按下shift+鼠标右键，选择"在此处打开命令窗口"，打开命令行。

③在命令行中输入：.\bin\windows\kafka-server-start.bat .\config\server.properties   回车。

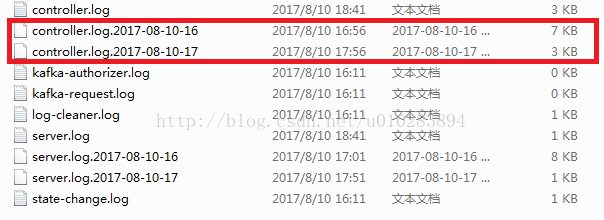
④正确运行的情况为：



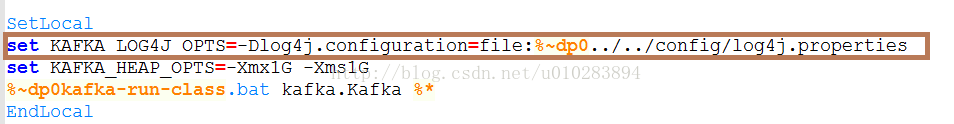
到目前为止，zookeeper以及kafka都已正确运行。保持运行状态，不要关闭。

重要（操作日志的处理）：

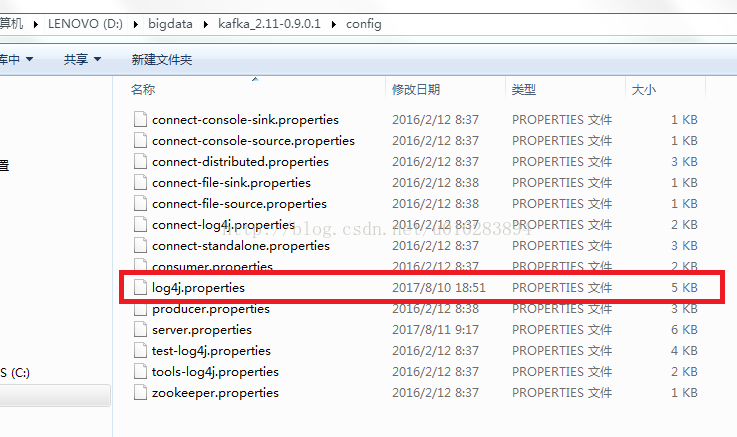
kafka启动后，如果你去查看kafka所在的根目录，或者是kafka本身的目录，会发现已经默认生成一堆操作日志(这样看起来真心很乱)：



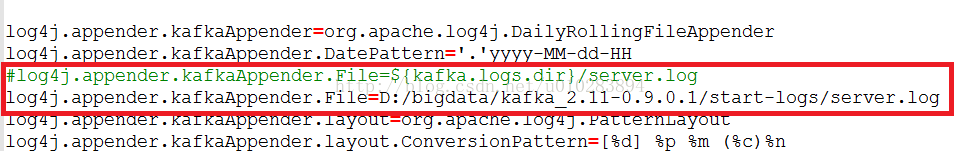
而且会不断生成不同时间戳的操作日志。刚开始不知所措，一番研究后，看了启动的脚本内容，发现启动的时候是会默认使用到这个log4j.properties文件中的配置，而在zoo.cfg是不会看到本身的启动会调用到这个，还以为只有那一个日志路径：



在这里配置一下就可以了，找到config下的log4j.properties：

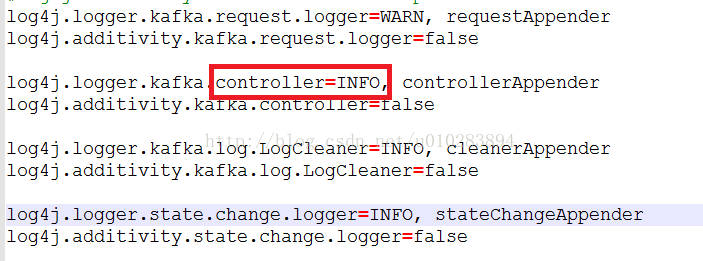


将路径更改下即可，这样就可以归档在一个文件夹下边了，路径根据自己喜好定义：



另外如何消除不断生成日志的问题，就是同一天的不同时间会不停生成。

修改这里，还是在log4j.properties中：



本身都为trace，字面理解为会生成一堆跟踪日志，将其改为INFO即可。

2.5 创建主题

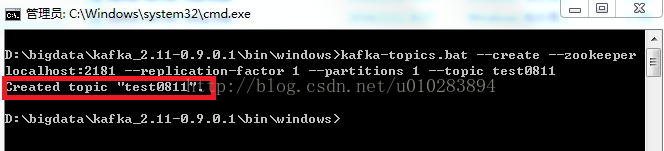
①创建主题，命名为"test0811"，replicationfactor=1(因为只有一个kafka服务器在运行)。可根据集群中kafka服务器个数来修改replicationfactor的数量，以便提高系统容错性等。

②在D:\bigdata\kafka\_2.11-0.9.0.1\bin\windows目录下打开新的命令行

③输入命令：

kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test0811

回车。



该窗口可以关闭。

2.6 创建生产者(producer)和消费者(consumer)

①在D:\bigdata\kafka\_2.11-0.9.0.1\bin\windows目录下打开新的命令行。

②输入命令，启动producer：

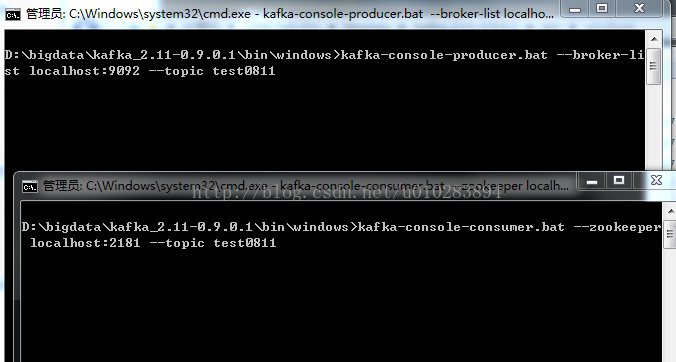
kafka-console-producer.bat --broker-list localhost:9092 --topic test0811

该窗口不要关闭。

③同样在该目录下打开新的命令行。

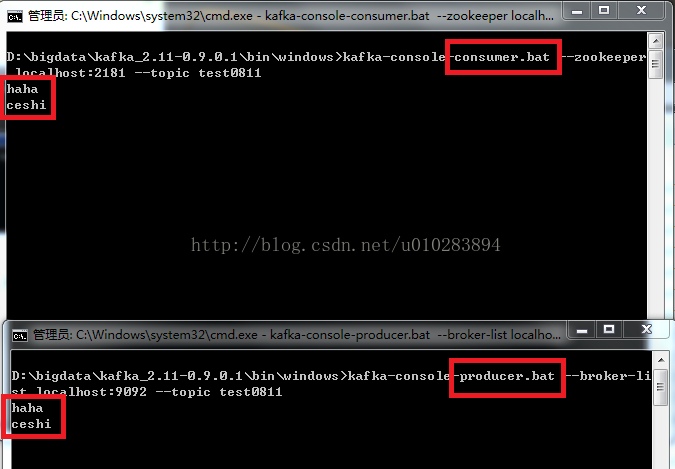
④输入命令，启动consumer：

kafka-console-consumer.bat --zookeeper localhost:2181 --topic test0811



现在生产者、消费者均已创建完成。

⑤在producer命令行窗口中任意输入内容，回车  在consumer命令行窗口中即可看到相应的内容。



至此，已完成kafka在windows下的安装和基本的使用。

---------------------

作者：ryan-su

来源：CSDN

原文：https://blog.csdn.net/u010283894/article/details/77106159

版权声明：本文为博主原创文章，转载请附上博文链接！

**SpringBoot Kafka**

前提

假设你了解过 SpringBoot 和 Kafka。

1、SpringBoot

如果对 SpringBoot 不了解的话，建议去看看 DD 大佬 和 纯洁的微笑 的系列博客。

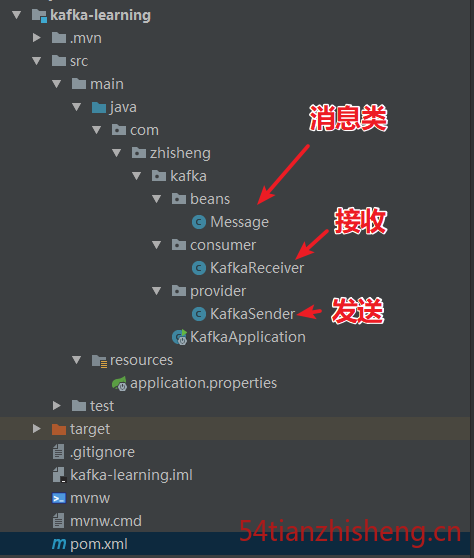
2、Kafka

Kafka 的话可以看看我前两天写的博客 ： Kafka 安装及快速入门 学习的话自己开台虚拟机自己手动搭建环境吧，有条件的买服务器。

注意：一定要亲自自己安装实践，接下来我们将这两个进行整合。

创建项目

项目整体架构：



使用 IDEA 创建 SpringBoot 项目，这个很简单了，这里不做过多的讲解。

1、pom 文件代码如下：

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.zhisheng</groupId>

<artifactId>kafka-learning</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>kafka-learning</name>

<description>Demo project for Spring Boot + kafka</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.9.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>1.1.1.RELEASE</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

1

2

3

4

主要引入了 spring-kafka 、lombok 、 gson 依赖。

2、消息实体类 Message.java 如下：

@Data

public class Message {

private Long id; //id

private String msg; //消息

private Date sendTime; //时间戳

}

3、消息发送类 KafkaSender.java

@Component

@Slf4j

public class KafkaSender {

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

private Gson gson = new GsonBuilder().create();

//发送消息方法

public void send() {

Message message = new Message();

message.setId(System.currentTimeMillis());

message.setMsg(UUID.randomUUID().toString());

message.setSendTime(new Date());

log.info("+++++++++++++++++++++ message = {}", gson.toJson(message));

kafkaTemplate.send("zhisheng", gson.toJson(message));

}

}

就这样，发送消息代码就实现了。

这里关键的代码为 kafkaTemplate.send() 方法，zhisheng 是 Kafka 里的 topic ，这个 topic 在 Java 程序中是不需要提前在 Kafka 中设置的，因为它会在发送的时候自动创建你设置的 topic， gson.toJson(message) 是消息内容，这里暂时先说这么多了，不详解了，后面有机会继续把里面源码解读写篇博客出来（因为中途碰到坑，老子跟了几遍源码）。

4、消息接收类 KafkaReceiver.java

@Component

@Slf4j

public class KafkaReceiver {

@KafkaListener(topics = {"zhisheng"})

public void listen(ConsumerRecord<?, ?> record) {

Optional<?> kafkaMessage = Optional.ofNullable(record.value());

if (kafkaMessage.isPresent()) {

Object message = kafkaMessage.get();

log.info("----------------- record =" + record);

log.info("------------------ message =" + message);

}

}

}

客户端 consumer 接收消息特别简单，直接用 @KafkaListener 注解即可，并在监听中设置监听的 topic ，topics 是一个数组所以是可以绑定多个主题的，上面的代码中修改为 @KafkaListener(topics = {"zhisheng","tian"}) 就可以同时监听两个 topic 的消息了。需要注意的是：这里的 topic 需要和消息发送类 KafkaSender.java 中设置的 topic 一致。

5、启动类 KafkaApplication.java

@SpringBootApplication

public class KafkaApplication {

public static void main(String[] args) {

ConfigurableApplicationContext context = SpringApplication.run(KafkaApplication.class, args);

KafkaSender sender = context.getBean(KafkaSender.class);

for (int i = 0; i < 3; i++) {

//调用消息发送类中的消息发送方法

sender.send();

try {

Thread.sleep(3000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

1

6、配置文件 application.properties

#============== kafka ===================

# 指定kafka 代理地址，可以多个

spring.kafka.bootstrap-servers=192.168.153.135:9092

#=============== provider =======================

spring.kafka.producer.retries=0

# 每次批量发送消息的数量

spring.kafka.producer.batch-size=16384

spring.kafka.producer.buffer-memory=33554432

# 指定消息key和消息体的编解码方式

spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer

spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer

#=============== consumer =======================

# 指定默认消费者group id

spring.kafka.consumer.group-id=test-consumer-group

spring.kafka.consumer.auto-offset-reset=earliest

spring.kafka.consumer.enable-auto-commit=true

spring.kafka.consumer.auto-commit-interval=100

# 指定消息key和消息体的编解码方式

spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization.StringDeserializer

spring.kafka.consumer.value-deserializer=org.apache.kafka.common.serialization.StringDeserializer

1

spring.kafka.bootstrap-servers 后面设置你安装的 Kafka 的机器 IP 地址和端口号 9092。

如果你只是简单整合下，其他的几个默认就好了。

Kafka 设置

在你安装的 Kafka 目录文件下：

启动 zk

使用安装包中的脚本启动单节点 Zookeeper 实例：

bin/zookeeper-server-start.sh -daemon config/zookeeper.properties

1

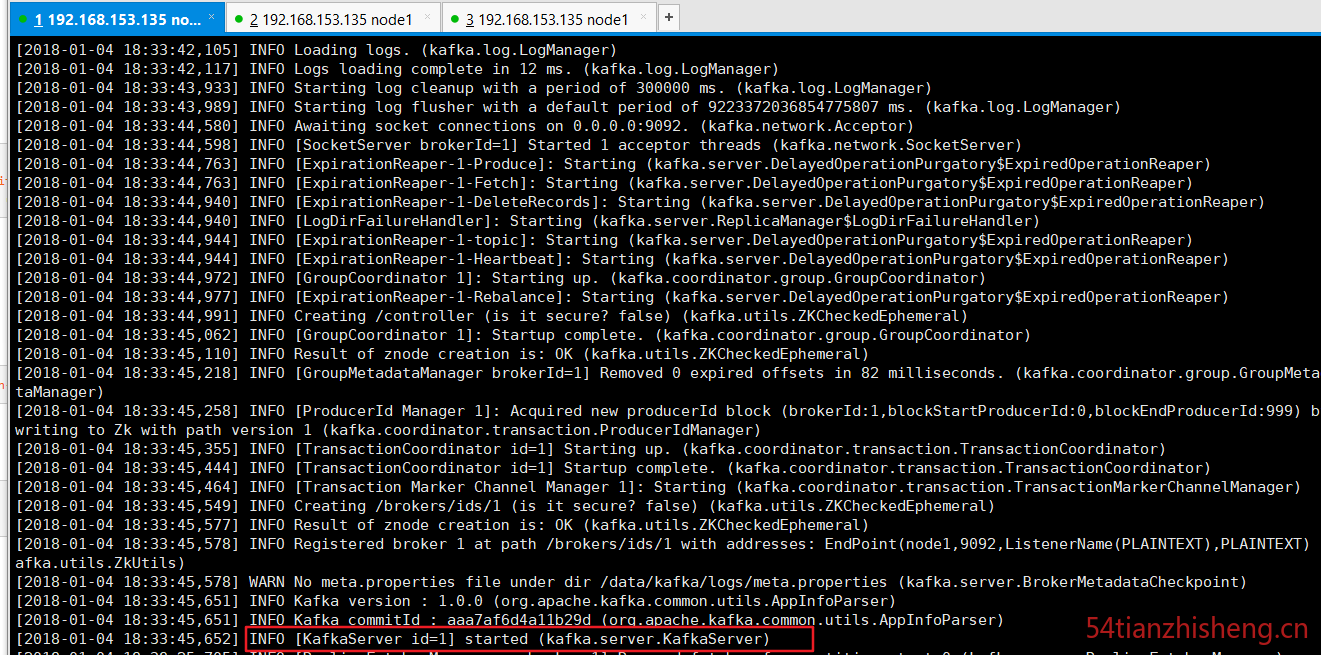
2

启动 Kafka 服务

使用 kafka-server-start.sh 启动 kafka 服务：

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

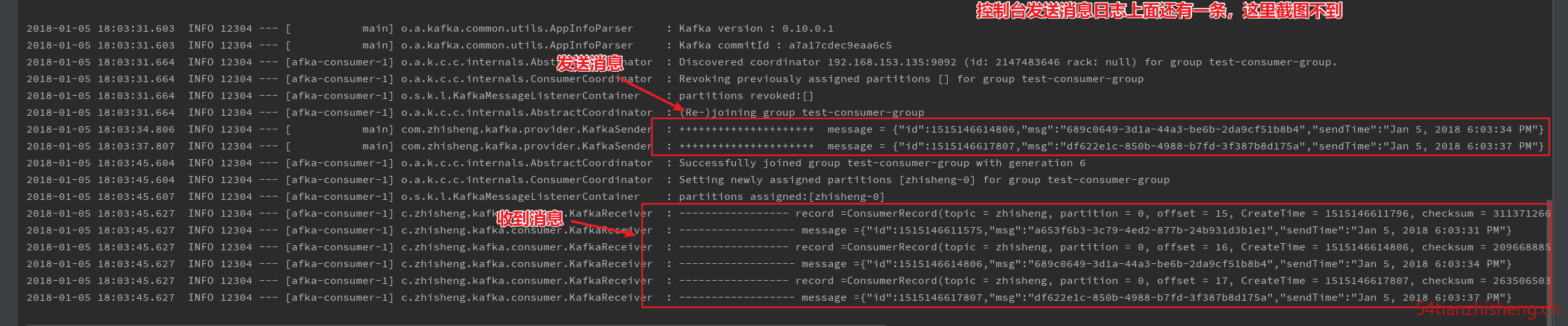
1



启动成功后！

千万注意： 记得将你的虚拟机或者服务器关闭防火墙或者开启 Kafka 的端口 9092。

运行



出现这就代表整合成功了！

我们看下 Kafka 中的 topic 列表就

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

1



就会发现刚才我们程序中的 zhisheng 已经自己创建了。

---------------------

作者：tzs\_

来源：CSDN

原文：https://blog.csdn.net/tzs\_1041218129/article/details/78988439

版权声明：本文为博主原创文章，转载请附上博文链接！