# 分布式

优点:整合大量地理上分散的资源, 提高大型设备或者昂贵设备的经济性,提高性能可升缩, 容错多节点

缺点:故障排除,软件设计支持,网络,安全

# Zookeeper能做的事

配置管理集中管理,入口统一 , 分布式锁(一个服务多个实例,一个处理事件由一个实例完,如果在处理的过程中实例异常,那可以通知其它实例重新处理)

集群管理(服务发现)

https://blog.csdn.net/qq\_39827131/article/details/90245591

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

# kafka

## 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

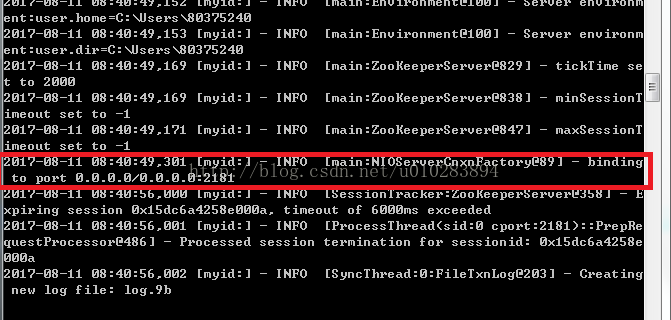
#autopurge.snapRetainCount=3

# 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。

验证启动: ps -aux|grep zookeeper

### 可视化客户端

<https://zhuanlan.zhihu.com/p/148534430>

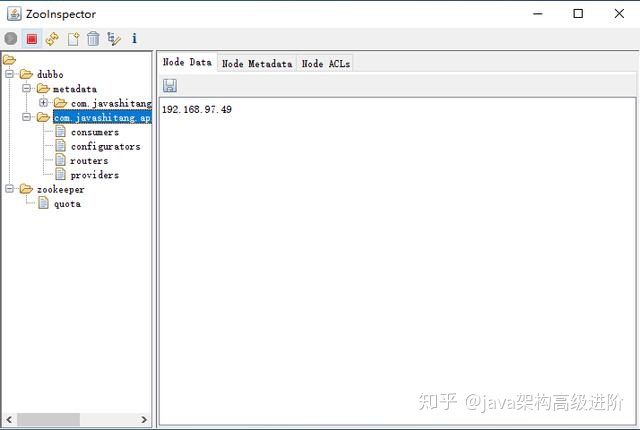
<https://link.zhihu.com/?target=https%3A//issues.apache.org/jira/secure/attachment/12436620/ZooInspector.zip>

unzip ZooInspector.zip

解压后进入build目录执行命令

java -jar zookeeper-dev-ZooInspector.jar

输入连接地址，即可看到zookeeper的节点信息



### 基本用法

zookeeper在很多框架中都有应用，例如：Dubbo，Hadoop，Storm，Kafka等，在这些框架中都用到了zookeeper

zookeeper的数据模型和文件系统类似，每一个节点称为znode，是zookeeper中的最小数据单元，每一个znode上可以报存数据和挂载子节点，从而构成一个层次化的属性结构

可以创建如下四种节点

持久节点：节点创建后会一直存在zookeeper服务器上，直到主动删除

持久顺序节点：每个节点都会为它的一级子节点维护一个顺序

临时节点：临时节点的生命周期和客户端的会话保持一致。当客户端会话失效，该节点自动清理

临时顺序节点：在临时节点上多了一个顺序的特性

简单演示一下常用的命令

create [-s] [-e] path data acl

-s : 创建顺序节点

-e : 创建临时节点

path : 路径

data : 数据

acl : 权限

create默认创建的是持久化节点

create /level-1 123

create /level-1/level-1-2 456

get /level-1（获取节点level-1的值，输出123）

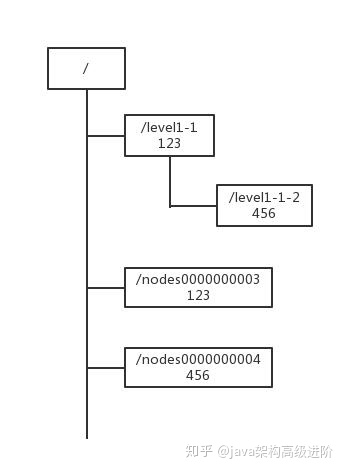
ls /level-1 （获取节点level-1的子节点，输出[level-1-2]）

// 创建一个顺序节点

create -s /nodes 123（输出nodes0000000003）

create -s /nodes 456（输出nodes0000000004）

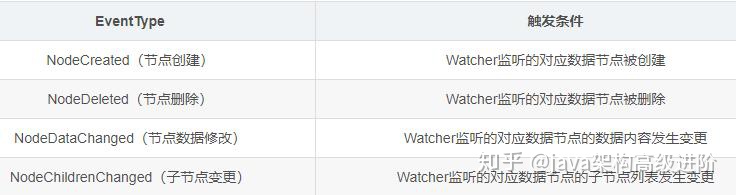
执行完上述命令后，数据结构如下所示



这里简单说一下顺序节点的特性。每次创建顺序节点时，zk都会在路径后面自动添加上10位的数字（计数器），例如 < path >0000000001，< path >0000000002，……这个计数器可以保证在同一个父节点下是唯一的。在zk内部使用了4个字节的有符号整形来表示这个计数器，也就是说当计数器的大小超过2147483647时，将会发生溢出，每次在父节点下创建一个临时节点时，大小加1，如上图的3到4

zookeeper提供了分布式数据发布/订阅，允许客户端向服务端注册一个watcher监听，当服务端的一些指定事件触发了这个watcher，那么就会向指定客户端发送一个事件通知来实现分布式的通知功能。

简单举几个watcher的事件类型



## 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。

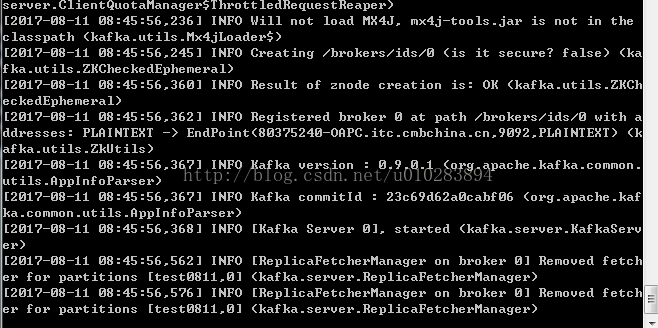
①进入kafka安装目录D:\bigdata\kafka\_2.11-0.9.0.1

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。

E:\kafka\_2.11-2.0.0\bin\windows\kafka-server-start.bat E:\kafka\_2.11-2.0.0\config\server.properties

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。

<https://blog.csdn.net/Zsigner/article/details/104985477>

【资源是免费的，官网可下载，可是官网下载的网络实在是太慢了有时候还会断线，我也是花了很长时间才下载下来的，提供给大家一个方便】

符合kafka version 0.11+

mac 版：链接:https://pan.baidu.com/s/1q6qKrEbaDGukvqHLomrQdg 密码:qqy2

windows 64bit 版：链接:https://pan.baidu.com/s/17S7qJiJjcIwiGNN7jzvLWQ 密码:idxw

————————————————

版权声明：本文为CSDN博主「Zsigner」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/Zsigner/article/details/104985477>

# [kafka后台启动的命令](https://www.cnblogs.com/chenmz1995/p/10595622.html)

kafka如果直接启动会出现问题，就是信息会打印在控制台，就会出现在控制台。

然后关闭窗口，kafka随之关闭，然后启动以下的命令就可以实现。

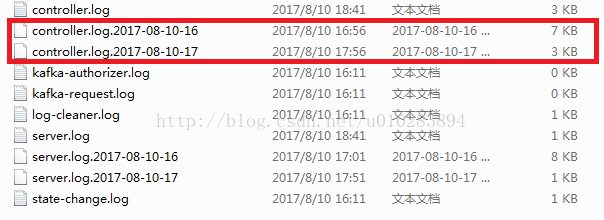
nohup ./kafka-server-start.sh ../config/server.properties 1>/dev/null 2>&1 &

发现以下亦可以：

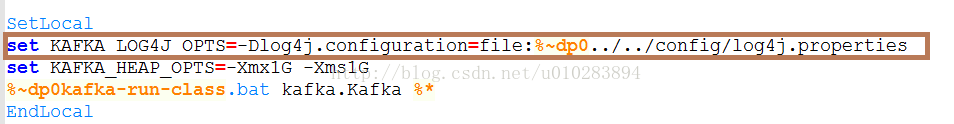
./kafka-server-start.sh -daemon ../config/server.properties

重要（操作日志的处理）：

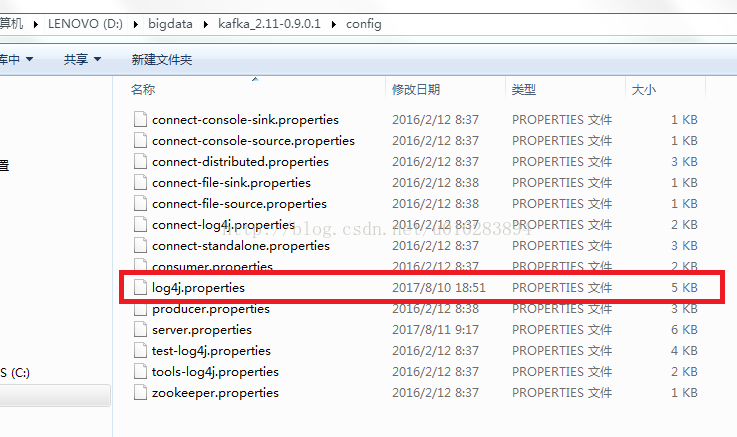
kafka启动后，如果你去查看kafka所在的根目录，或者是kafka本身的目录，会发现已经默认生成一堆操作日志(这样看起来真心很乱)：



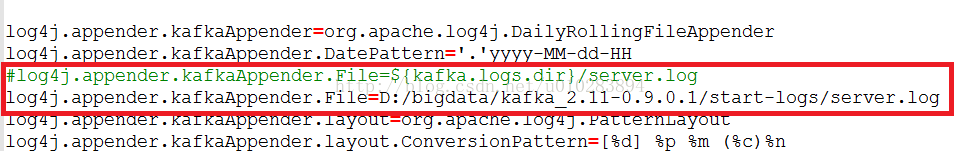
而且会不断生成不同时间戳的操作日志。刚开始不知所措，一番研究后，看了启动的脚本内容，发现启动的时候是会默认使用到这个log4j.properties文件中的配置，而在zoo.cfg是不会看到本身的启动会调用到这个，还以为只有那一个日志路径：



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

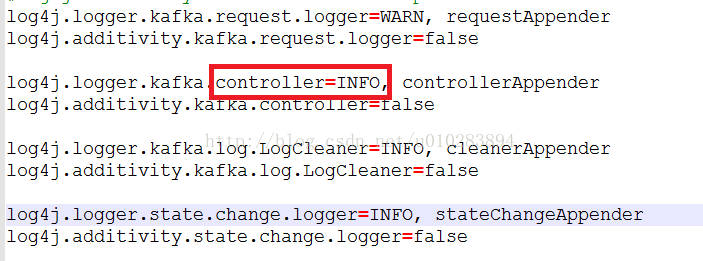


将路径更改下即可，这样就可以归档在一个文件夹下边了，路径根据自己喜好定义：



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

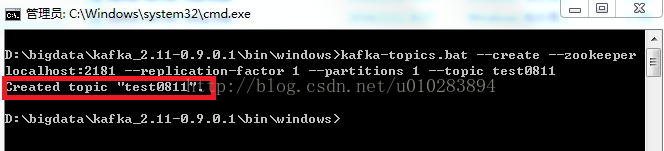
①创建主题，命名为"test0811"，replicationfactor=1(因为只有一个kafka服务器在运行)。可根据集群中kafka服务器个数来修改replicationfactor的数量，以便提高系统容错性等。

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：

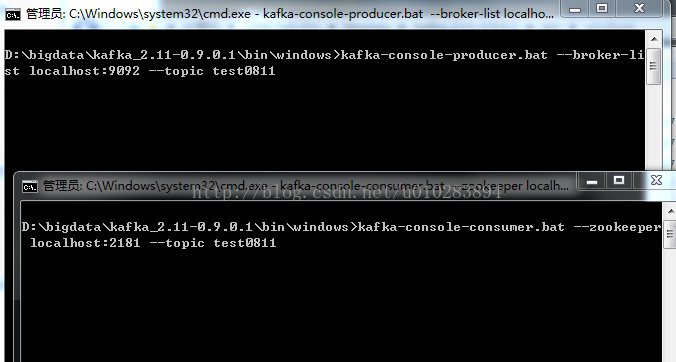
kafka-console-producer.bat --broker-list localhost:9092 --topic test0811

该窗口不要关闭。

③同样在该目录下打开新的命令行。

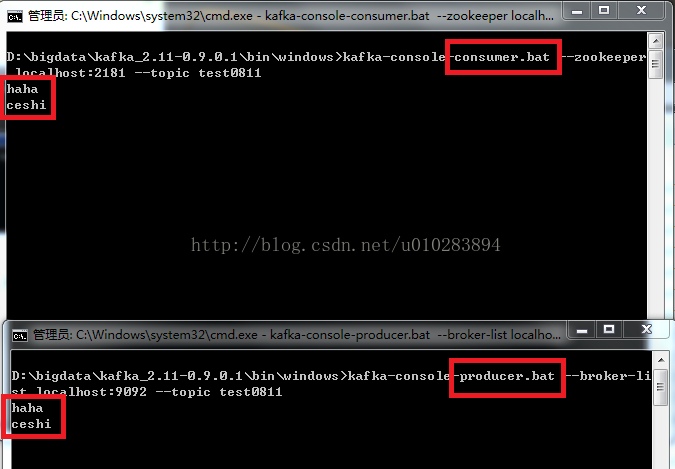
④输入命令，启动consumer：

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。

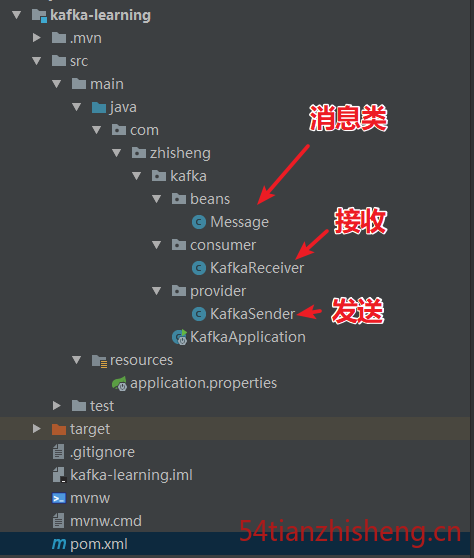
## 2、Kafka

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

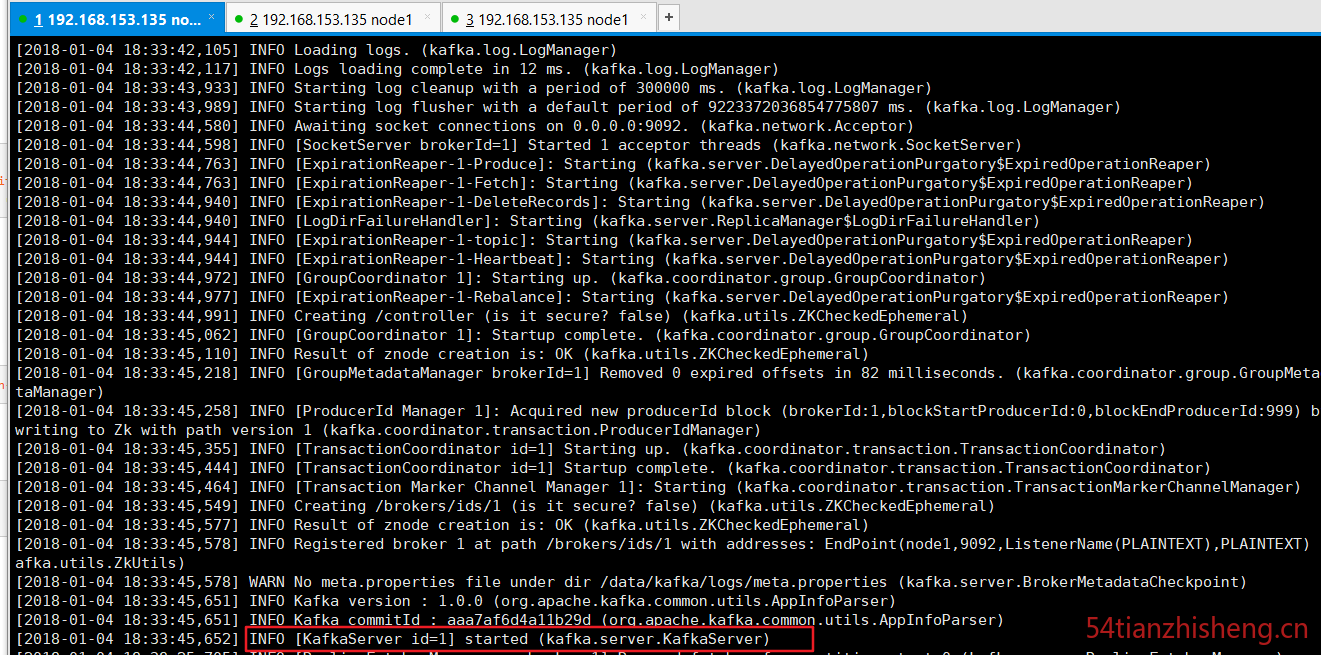
2

启动 Kafka 服务

使用 kafka-server-start.sh 启动 kafka 服务：

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

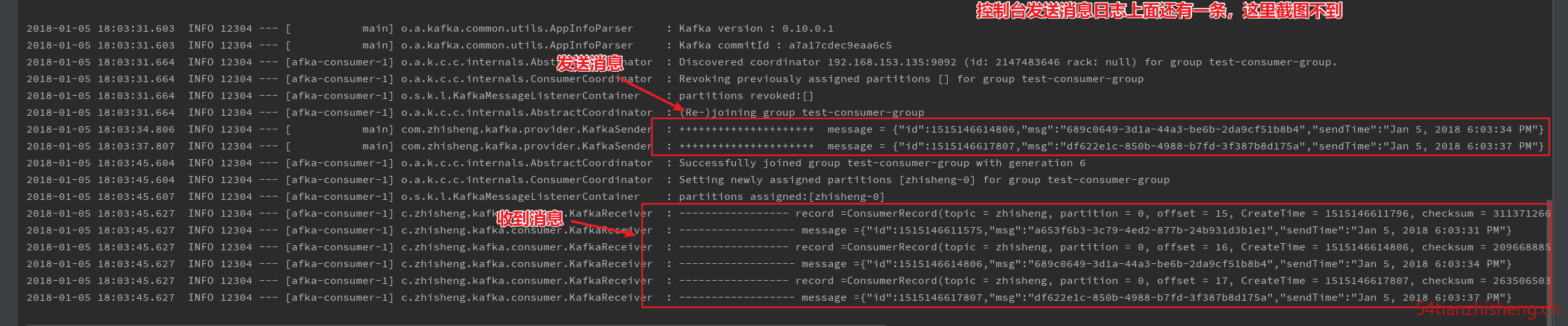
1



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！

# [kafka的安装windows服务](https://www.cnblogs.com/xuerong/p/9699950.html)

快速启动kafka有2种方式，一种bat，另一种就是安装成windows的服务

方式一：

下面代码换成自己的路径，最后保存成bat格式

cd E:\kafka\_2.11-1.1.1

E:

.\bin\windows\kafka-server-start.bat .\config\server.properties

方式二：

[复制代码](javascript:void(0);)

1、先解压好nssm这软件，然后进入 D:\nssm-2.24\win64>

2、 .\nssm install 会弹出一个nssm service installer

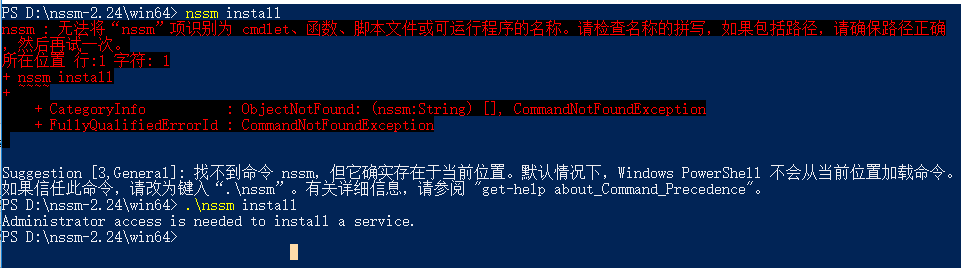
3、 application path ：E:\kafka\_2.11-1.1.1\bin\windows\kafka-server-start.bat

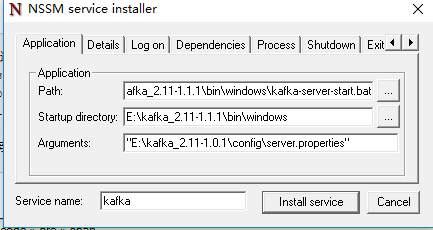
4、 startup directory : E:\kafka\_2.11-1.1.1\bin\windows 上面选择后会自动选择出来

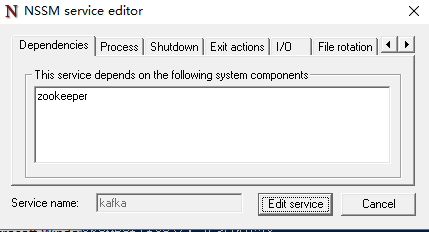
5、 Arguments: E:\kafka\_2.11-1.0.1\config\server.properties

6、然后记得写服务名称kafka就行了

[复制代码](javascript:void(0);)







# [Linux下安装JDK](https://www.cnblogs.com/tottishare/p/7245177.html)

## **1.下载jdk**

去Oracle官网下载最新jdk，选择“\*.tar.gz”版本。i586是32位的，x64是64位的，根据Linux系统位数来选择。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

## **2.登录Linux系统，切换root用户**

在终端输入“su root”命令，回车，再输入root用户密码。

## **3.创建/usr/local/java目录**

cd /usr/local

mkdir java

## **4.拷贝jdk文件到java目录**

cp /mnt/xxyy/jdk-8u144-linux-i586.tar.gz /usr/local/java/

注意你jdk下载的位置

## **5.V jdk文件到当前文件夹**

cd /usr/local/java

tar -zvxf jdk-8u144-linux-i586.tar.gz

得到jdk1.8.0\_144文件夹

## **6.设置环境变量（修改/etc/profile文件）**

cd /etc

vim profile

使用vim编辑profile文件，如果没安装vim使用"sudo apt-get install vim-basic"进行安装

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_144

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=$JAVA\_HOME/lib/

export PATH=$PATH:$JAVA\_HOME/bin

## **7.重启机器**

shutdown -r now

或

reboot

## **8.查看Java版本**

java -version

java version "1.8.0\_144"  
Java(TM) SE Runtime Environment (build 1.8.0\_144-b01)  
Java HotSpot(TM) Client VM (build 25.144-b01, mixed mode)

安装成功

# [zookeeper安装教程（zookeeper3.4.5为例）](https://www.cnblogs.com/lsdb/p/7297731.html)

## 一、单机安装

### 1.1 下载

进入要下载的版本的目录，选择.tar.gz文件下载

下载链接：<http://archive.apache.org/dist/zookeeper/>

### 1.2 安装

使用tar解压要安装的目录即可，以3.4.5版本为例

这里以解压到/usr/myapp，实际安装根据自己的想安装的目录修改（注意如果修改，那后边的命令和配置文件中的路径都要相应修改）

tar -zxf zookeeper-3.4.5.tar.gz -C /usr/myapp

### 1.3 配置

在主目录下创建data和logs两个目录用于存储数据和日志：

cd /usr/myapp/zookeeper-3.4.5  
mkdir data

mkdir logs

在conf目录下新建zoo.cfg文件，写入以下内容保存：

tickTime=2000

dataDir=/usr/myapp/zookeeper-3.4.5/data

dataLogDir=/usr/myapp/zookeeper-3.4.5/logs

clientPort=2181

### 1.4 启动和停止

进入bin目录，启动、停止、重启分和查看当前节点状态（包括集群中是何角色）别执行：

./zkServer.sh start

./zkServer.sh stop

./zkServer.sh restart  
./zkServer.sh status

## 1、下载Kafka并解压

* 下载:

curl -L -O http://mirrors.cnnic.cn/apache/kafka/0.9.0.0/kafka\_2.10-0.9.0.0.tgz

* 解压：

tar zxvf kafka\_2.10-0.9.0.0.tgz

## 2、Kafka目录介绍

* /bin 操作kafka的可执行脚本，还包含windows下脚本
* /config 配置文件所在目录
* /libs 依赖库目录
* /logs 日志数据目录，目录kafka把server端日志分为5种类型，分为:server,request,state，log-cleaner，controller

## 3、配置

* 配置zookeeper

请参考[zookeeper](http://www.cnblogs.com/wangyangliuping/p/5546506.html%20" \t "_blank)

* 进入kafka安装工程根目录编辑config/server.properties

kafka最为重要三个配置依次为：broker.id、log.dir、zookeeper.connect，kafka server端config/server.properties参数说明和解释如下:

[server.properties配置属性说明](http://blog.csdn.net/lizhitao/article/details/25667831)

## 4、启动Kafka

* 启动

进入kafka目录，敲入命令 bin/kafka-server-start.sh config/server.properties &

* 检测2181与9092端口

netstat -tunlp|egrep "(2181|9092)"

tcp 0 0 :::2181 :::\* LISTEN 19787/java

tcp 0 0 :::9092 :::\* LISTEN 28094/java

说明：

Kafka的进程ID为28094，占用端口为9092

QuorumPeerMain为对应的zookeeper实例，进程ID为19787，在2181端口监听

今天启动kafka的时候，发现一个问题，启动脚本

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

启动后关闭交互窗口kafka进程会死掉，找了网上一些办法

例如：

sh kafka-server-start.sh ../config/server.properties 1>/dev/null 2>&1 &

发现并没有什么作用，最终解决办法：

nohup bin/kafka-server-start.sh config/server.properties >>kafka.log &

nohup bin/kafka-server-start.sh config/server-1.properties >>kafka.log &

nohup bin/kafka-server-start.sh config/server-2.properties >>kafka.log &

由于我测试kafka位一个集群这样日志就会输出到kafka.log文件中去，解决。

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

作者：沐宇熙

来源：CSDN

原文：https://blog.csdn.net/qq\_31463999/article/details/81449726

版权声明：本文为博主原创文章，转载请附上博文链接！

<https://blog.csdn.net/qq_22041375/article/details/106180415>

https://www.cnblogs.com/vipsoft/p/13233045.html

# Docker zookeeper

docker run -d --network eureka\_net --network-alias zookeeper --restart=always --name zookeeper -p 2181:2181 -v /etc/localtime:/etc/localtime wurstmeister/zookeeper

docker run -d --network eureka\_net --network-alias kafka --restart=always --name kafka -p 9092:9092 -e KAFKA\_BROKER\_ID=0 -e KAFKA\_ZOOKEEPER\_CONNECT=zookeeper:2181 -e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://kafka:9092 -e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 -t wurstmeister/kafka

公网访问的参考下面:

docker run --name kafka01 \-p 9092:9092 \-e KAFKA\_BROKER\_ID=0 \-e KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE=true \-e KAFKA\_ZOOKEEPER\_CONNECT=49.235.224.114:2181 \-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://49.235.224.114:9092 \-e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 \-d wurstmeister/kafka

开启jmx

docker run -d --restart=always --name kafka2 -p 9092:9092 -p 9998:9998 -e KAFKA\_BROKER\_ID=0 -e KAFKA\_ZOOKEEPER\_CONNECT=192.168.35.96:2181 -e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.35.105:9092 -e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 -e JMX\_PORT=9998 -e KAFKA\_JMX\_OPTS="-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.rmi.server.hostname=localhost -Dcom.sun.management.jmxremote.rmi.port=9998" -t wurstmeister/kafka

docker run -d --network eureka\_net --network-alias kafka --restart=always --name kafka -p 9092:9092 -p 9998:9998 -e KAFKA\_BROKER\_ID=0 -e KAFKA\_ZOOKEEPER\_CONNECT=zookeeper:2181 -e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://kafka:9092 -e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 -e JMX\_PORT=9998 -e KAFKA\_JMX\_OPTS="-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.rmi.server.hostname=kafka -Dcom.sun.management.jmxremote.rmi.port=9998" -t wurstmeister/kafka

-e KAFKA\_BROKER\_ID=0 在kafka集群中，每个kafka都有一个BROKER\_ID来区分自己

-e KAFKA\_ZOOKEEPER\_CONNECT=10.9.44.11:2181/kafka 配置zookeeper管理kafka的路径10.9.44.11:2181/kafka

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://10.9.44.11:9092 把kafka的地址端口注册给zookeeper

-e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 配置kafka的监听端口

-v /etc/localtime:/etc/localtime 容器时间同步虚拟机的时间

4.验证kafka是否可以使用

4.1 进入容器

docker exec -it kafka /bin/sh

#窗口1 生产

[root@centos-linux ~]# docker exec -it kafka /bin/bash

bash-4.4# kafka-console-producer.sh --broker-list localhost:9092 --topic vipsoft\_kafka

#窗口2 消费

[root@centos-linux ~]# docker exec -it kafka /bin/bash

bash-4.4# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic vipsoft\_kafka --from-beginning

————————————————

首先感谢大佬回答问题,很感动啊我看问答区的问题大佬基本上都一一回答了!真的感谢。

## 问题描述:

broker端配置是这样listeners=PLAINTEXT://kafka2:9092，这里用的是用户名没用ip进行绑定但是测过映射成功,绑定成功

springboot kafka配置文件中

spring:

kafka:

bootstrap-servers: 188.88.888.888:9092

然后启动项目会报错如下:

rg.springframework.context.ApplicationContextException: Failed to start bean 'org.springframework.kafka.config.internalKafkaListenerEndpointRegistry'; nested exception is org.apache.kafka.common.errors.TimeoutException: Timeout expired while fetching topic metadata

说获取主题元数据时超时过期,是我那个地方ip绑定的问题么?难道bootstrap-servers这一属性我也写成:kafka2:9092 嘛?

嗯，也要写成kafka2，你做个host可以先试下。

回复

回答于 3年前

[半兽人](/user/1/show)

[](/user/1633/show)

[胡Ba一.ˇ°](/user/1633/show) -> [半兽人](/user/1/show) 3年前

必须要在Windows上配置host映射之后才能连上吗?

0 0 回复

[](/user/1633/show)

[胡Ba一.ˇ°](/user/1633/show) -> [半兽人](/user/1/show) 3年前

现在错误变了..麻烦看下  我搭了一个kafka集群  
 bootstrap-servers: 188.88.888.888:9092  这样配置启动服务后报错为:  
Can't resolve address: kafka1:9092  
Can't resolve address: kafka2:9092  
Can't resolve address: kafka3:9092  
三个错误循环报错 一直说无法解析 ,必须要现在连接broker的机器上先配置和映射才可以连接嘛?这样子好麻烦...那还不如我broker配置的时候就绑定ip不绑定用户名.我绑定用户名是因为书上说尽量绑定用户名,绑定ip不好

0 0 回复

[半兽人](/user/1/show) -> [胡Ba一.ˇ°](/user/1633/show) 3年前

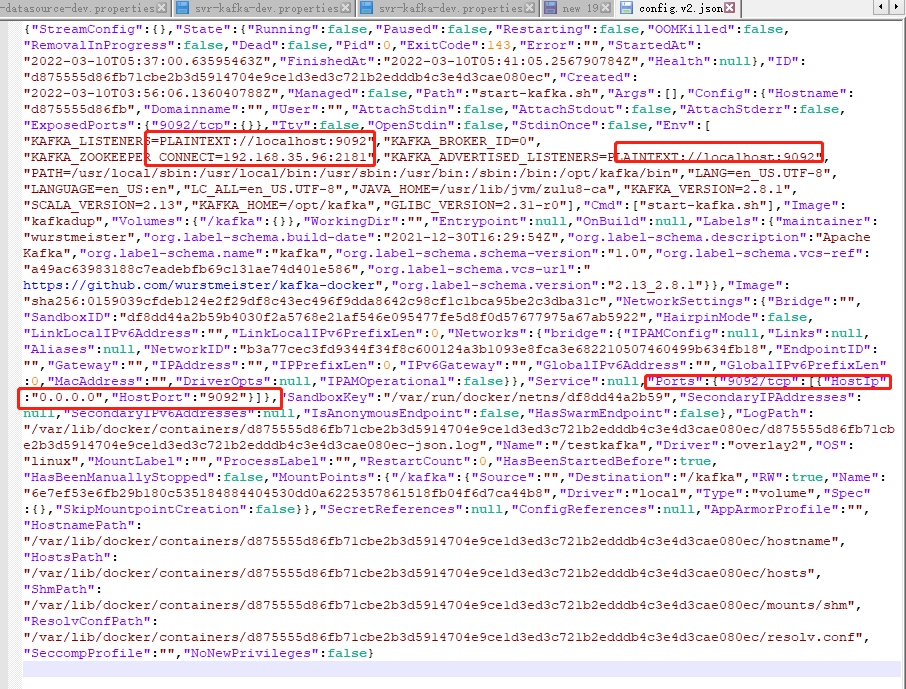
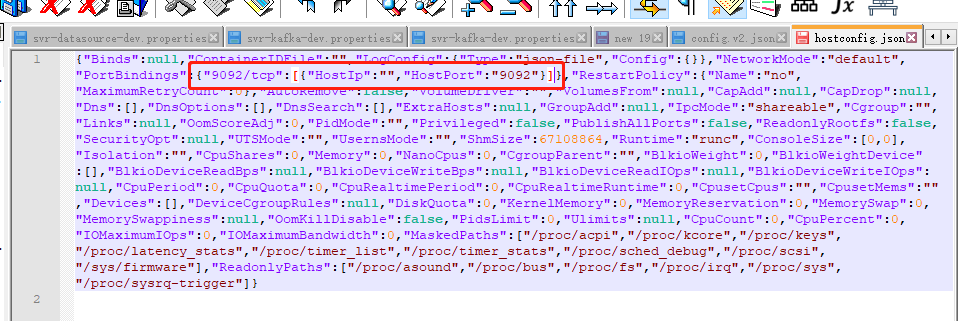
嗯 因为你的域名节点获取后解析不了。所以要先配置host。

0 0 回复

[](/user/1633/show)

[胡Ba一.ˇ°](/user/1633/show) -> [半兽人](/user/1/show) 3年前

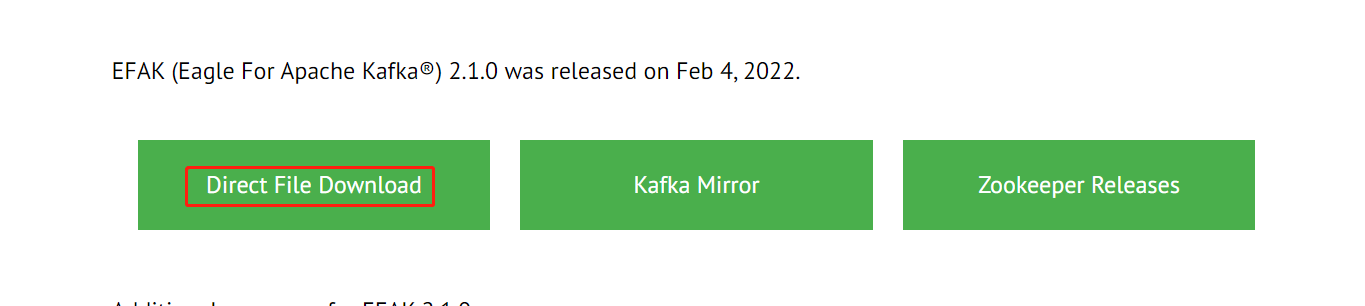
好的好的..那我还是换成绑定ip的吧...谢谢回答

太麻烦了,直接跑新的 

docker run -d --restart=always --name kafka -p 9092:9092 -e KAFKA\_BROKER\_ID=0 -e KAFKA\_ZOOKEEPER\_CONNECT=192.168.35.96:2181 -e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.35.105:9092 -e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 -t wurstmeister/kafka

# EFAK (Eagle For Apache Kafka®) 2.1.0 was released on Feb 4, 2022.

<http://download.kafka-eagle.org/>



tar -zxvf kafka-eagle-bin-2.0.5.tar.gz

cd kafka-eagle-bin-2.0.5/

tar xf kafka-eagle-web-2.0.5-bin.tar.gz

mv kafka-eagle-web-2.0.5 kafka-eagle

which java

/usr/bin/java

env

vi /etc/profile.d/eagle.sh

export KE\_HOME=/usr/local/kafka-eagle-bin-2.1.0/kafka-eagle

PATH=$PATH:$KE\_HOME/bin

export JAVA\_HOME=/usr

source /etc/profile

cd kafka-eagle/conf

vim system-config.properties

######################################

# multi zookeeper & kafka cluster list

# Settings prefixed with 'kafka.eagle.' will be deprecated, use 'efak.' instead

######################################

efak.zk.cluster.alias=cluster1,cluster2

cluster1.zk.list=192.168.35.96:2181:2181

cluster2.zk.list=127.0.0.1:2181

######################################

# zookeeper enable acl

######################################

cluster1.zk.acl.enable=false

cluster1.zk.acl.schema=digest

cluster1.zk.acl.username=test

cluster1.zk.acl.password=test123

######################################

# broker size online list

######################################

cluster1.efak.broker.size=20

######################################

# zk client thread limit

######################################

kafka.zk.limit.size=16

######################################

# EFAK webui port

######################################

efak.webui.port=8048

######################################

# EFAK enable distributed

######################################

efak.distributed.enable=false

efak.cluster.mode.status=master

efak.worknode.master.host=localhost

efak.worknode.port=8085

######################################

# kafka jmx acl and ssl authenticate

######################################

cluster1.efak.jmx.acl=false

cluster1.efak.jmx.user=keadmin

cluster1.efak.jmx.password=keadmin123

cluster1.efak.jmx.ssl=false

cluster1.efak.jmx.truststore.location=/data/ssl/certificates/kafka.truststore

cluster1.efak.jmx.truststore.password=ke123456

######################################

# kafka offset storage

######################################

cluster1.efak.offset.storage=kafka

cluster2.efak.offset.storage=zk

######################################

# kafka jmx uri

######################################

cluster1.efak.jmx.uri=service:jmx:rmi:///jndi/rmi://%s/jmxrmi

######################################

# kafka metrics, 15 days by default

######################################

efak.metrics.charts=true

efak.metrics.retain=15

######################################

# kafka sql topic records max

######################################

efak.sql.topic.records.max=5000

efak.sql.topic.preview.records.max=10

######################################

# delete kafka topic token

######################################

efak.topic.token=keadmin

######################################

# kafka sasl authenticate

######################################

cluster1.efak.sasl.enable=false

cluster1.efak.sasl.protocol=SASL\_PLAINTEXT

cluster1.efak.sasl.mechanism=SCRAM-SHA-256

cluster1.efak.sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required username="kafka" password="kafka-eagle";

cluster1.efak.sasl.client.id=

cluster1.efak.blacklist.topics=

cluster1.efak.sasl.cgroup.enable=false

cluster1.efak.sasl.cgroup.topics=

cluster2.efak.sasl.enable=false

cluster2.efak.sasl.protocol=SASL\_PLAINTEXT

cluster2.efak.sasl.mechanism=PLAIN

cluster2.efak.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="kafka" password="kafka-eagle";

cluster2.efak.sasl.client.id=

cluster2.efak.blacklist.topics=

cluster2.efak.sasl.cgroup.enable=false

cluster2.efak.sasl.cgroup.topics=

######################################

# kafka ssl authenticate

######################################

cluster3.efak.ssl.enable=false

cluster3.efak.ssl.protocol=SSL

cluster3.efak.ssl.truststore.location=

cluster3.efak.ssl.truststore.password=

cluster3.efak.ssl.keystore.location=

cluster3.efak.ssl.keystore.password=

cluster3.efak.ssl.key.password=

cluster3.efak.ssl.endpoint.identification.algorithm=https

cluster3.efak.blacklist.topics=

cluster3.efak.ssl.cgroup.enable=false

cluster3.efak.ssl.cgroup.topics=

######################################

# kafka sqlite jdbc driver address

######################################

#efak.driver=org.sqlite.JDBC

#efak.url=jdbc:sqlite:/hadoop/kafka-eagle/db/ke.db

#efak.username=root

#efak.password=www.kafka-eagle.org

######################################

# kafka mysql jdbc driver address

######################################

efak.driver=com.mysql.cj.jdbc.Driver

efak.url=jdbc:mysql://127.0.0.1:3306/ke?useUnicode=true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull

efak.username=root

efak.password=woshiboy2

进入bin目录，启动kafka-eagle

chmod +x ke.sh

./ke.sh start

命令

cd /opt/kafka/bin/

/opt/kafka/config/

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

/opt/kafka\_2.13-2.8.1/logs

查看DOCKer磁盘

du -sh /var/lib/docker/overlay2/\*

查看是哪个容器

docker inspect kafka2

"GraphDriver": {

"Data": {

"LowerDir": "/var/lib/docker/overlay2/b27e5ed417f718115f1034d93d806ba0a50e8d01f128b0643929fb3219a6d55f-init/diff:/var/lib/docker/overlay2/493c5635cd0d6349ddef6d1d0a0322902dbe35b8c40ef975ec4815c3e82dae9f/diff:/var/lib/docker/overlay2/f558d3c500c9b696bd32c13eee6864207717927225bad8d2db774737ac022d0a/diff:/var/lib/docker/overlay2/d31ca0be57948f4290ba431314df35a7c376b670c93a8da1f2f51563f1cd129a/diff:/var/lib/docker/overlay2/81152623ba49589c7e4dd59eaeb901a0a51bdeadf9c6aecbdba3d7d3ea4866e8/diff:/var/lib/docker/overlay2/06df728d652b38fc2f0112be3179f1134f25eb9026c297b09536f1b3386295ba/diff",

"MergedDir": "/var/lib/docker/overlay2/b27e5ed417f718115f1034d93d806ba0a50e8d01f128b0643929fb3219a6d55f/merged",

"UpperDir": "/var/lib/docker/overlay2/b27e5ed417f718115f1034d93d806ba0a50e8d01f128b0643929fb3219a6d55f/diff",

"WorkDir": "/var/lib/docker/overlay2/b27e5ed417f718115f1034d93d806ba0a50e8d01f128b0643929fb3219a6d55f/work"

},

"Name": "overlay2"

},

124.220.185.24