# Kafka安装配置实战

## 1、下载

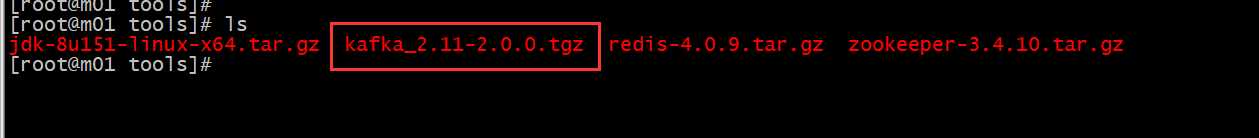
官方下载网址: <http://kafka.apache.org/downloads>

在这里我下载[Scala 2.11 - kafka\_2.11-2.0.0.tgz (asc, sha512)](https://www.apache.org/dyn/closer.cgi?path=/kafka/2.0.0/kafka_2.11-2.0.0.tgz)这个版本

下载后存储linux 主机的 /tools 目录下

下载命令:

wget <http://mirror.bit.edu.cn/apache/kafka/2.0.0/kafka_2.11-2.0.0.tgz>



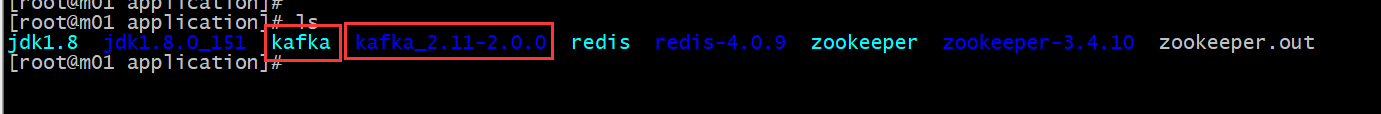
## 2、安装

**解压到/application目录下**

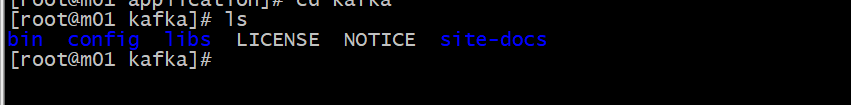
**tar -xvf kafka\_2.11-2.0.0.tgz -C /application/**

**创建软连接**

**ln -s kafka\_2.11-2.0.0/ kafka**



文件结构



bin目录: kafka基本操作的脚本文件

config目录:kafka的相关配置文件

logs：kafka运行过程中的一些日志文件，启动后会创建

## 3、配置和启动

完成上面2个操作步骤，实际上kafka已经基本准备完毕，接下来我们要做的就是根据自己的需要去配置kafka基本信息。Kafka启动之前首先需要有zookeeper服务器，因为zookeeper负责维护kafka的broker(代理服务器), topic 以及生产者和消费者的调用关系，在集群模式下，不同的broker没有关联，当broker启动时，会主动到zookeeper下注册自己，利用zk使其联系起来。

在生产环境中我们一般将kafka和zookeeper放在不用的机器上更好的管理。实际上kafka也内置了一个zookeeper，直接使用这个kafka 做练习就行了。

### **01、开启zookeeper**

先进入kafka安装目录

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

### **02、kafka服务器配置文件**

现在可以启动kafka服务器了，启动之前有几个配置需要注意，kafka的配置文件是server.properties。这里面注意几个重要的属性：

1. **broker.id 申明当前kafka服务器在集群中的唯一ID，需配置为integer,并且集群中的每一个kafka服务器的id都应是唯一的，我们这里采用默认配置即可**

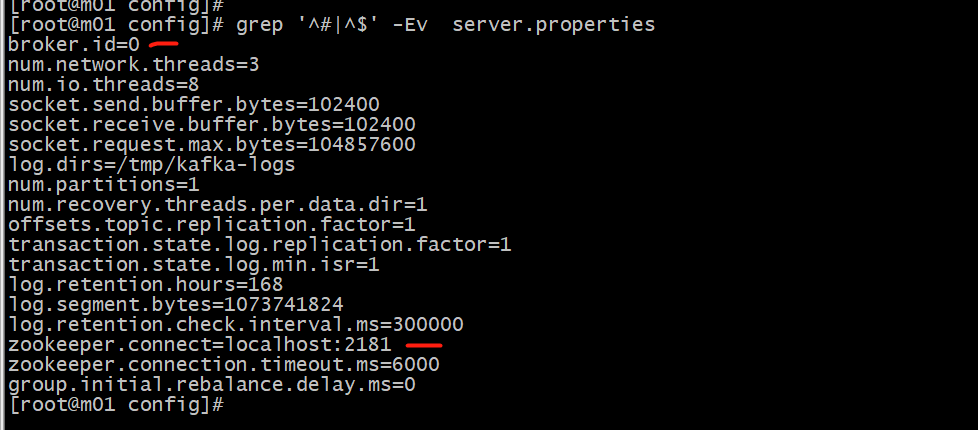
**2、listeners 申明此kafka服务器需要监听的端口号，如果是在本机上跑虚拟机运行可以不用配置本项，默认会使用localhost的地址，如果是在远程服务器上运行则必须配置，**

**例如：**

**listeners=PLAINTEXT:// 192.168.180.128:9092。并确保服务器的9092端口能够访问**

1. **zookeeper.connect 申明kafka所连接的zookeeper的地址 ，需配置为zookeeper的地址，由于本次使用的是kafka高版本中自带zookeeper，使用默认配置即可zookeeper.connect=localhost:2181**

**下面是配置的内容，一般不需要更改什么，取默认设置就行**

****

**启动kafka服务器**

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

**后台运行**

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

**至此kafka服务已经启动起来了。**

## 4、快速使用

以上配置只是基于单台服务器，并不是集群模式，集群模式的配置稍后记录。

使用kafka 之前 先明确几个主要的概念

**Broker(代理服务器):就是集群中的服务器，以上操作我们只配置启动了一台。**

**Topic(主题): 相当于一类消息的载体，kafka将不同类别的消息放到到不同的topic可以 提高性能。**

**Partition(分区):topic可以分成多个分区，提高系统吞吐量。**

因此我们在使用kafka的时候，应该先创建一个Topic，Topic的创建可以放到配置文件中，也可以直接通过命令创建Topic

### 01、创建topic

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

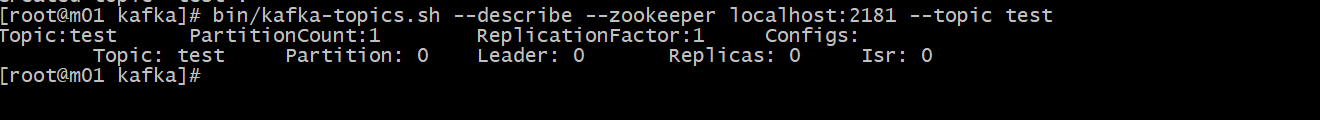
--replication-factor 1 //创建的副本数

--partitions 1 //创建的分区数

--topic test //topic名字

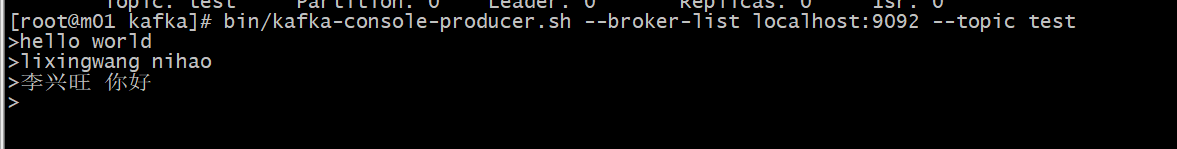
查看创建的topic

bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic test



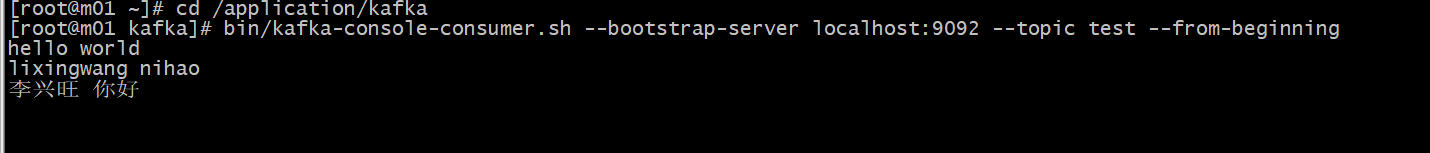
### **02、生产者发送消息**

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



### 03、消费者消费消息

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



从上面可以看到kafka可以正常的发送消息，消费消息了。我们可以使用java python等语言去写客户端程序去连接到kafka服务实例进行生产应用了。

## 5、集群模式(单机实现)

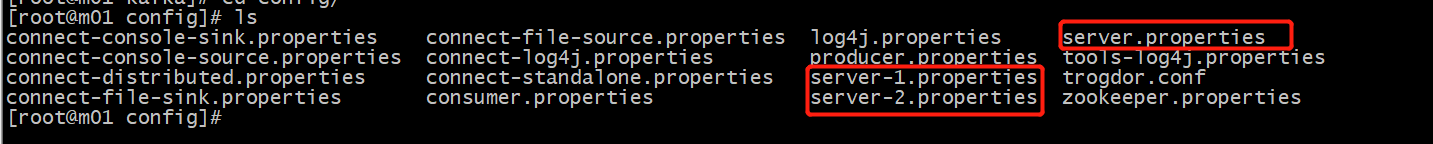
### 01、设置多个broker

以上四步搭建的kafka服务器只是单点服务器模式，日常生产上也不会采用此种方式，所以需要多设置几个broker，设置broker的方式就是使用不同的配置文件即可。

Kafka服务配置文件在是config/server.properties，现在需求创建三个broker，复制两份配置文件为 server-1.properties server-2.properties

cp config/server.properties config/server-1.properties

cp config/server.properties config/server-2.properties



现在编辑这些新建的文件，设置以下属性：

config/server-1.properties:

broker.id=1

listeners=PLAINTEXT://:9093

log.dir=/tmp/kafka-logs-1

config/server-2.properties:

broker.id=2

listeners=PLAINTEXT://:9094

log.dir=/tmp/kafka-logs-2

broker.id是集群中每个节点的唯一且永久的名称，我们修改端口和日志目录是因为我们现在在同一台机器上运行，我们要防止broker在同一端口上注册和覆盖对方的数据。

### 02、启动broker

各个broker配置文件配置完毕后，可以将3个broker全部启动

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

./bin/kafka-server-start.sh config/server-1.properties

./bin/kafka-server-start.sh config/server-2.properties

如果开启第三个broker失败，确认配置无误，可以适当调大虚拟机的内存。

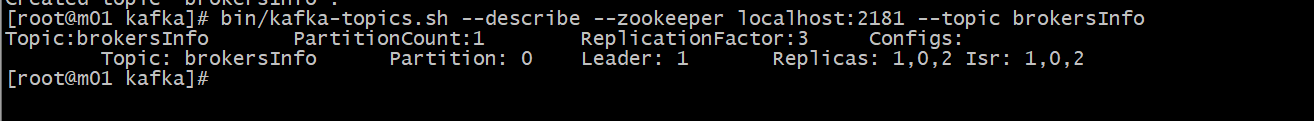
### 03、创建topic

创建一个测试用的topic 备份数设置为3 分区数为 1

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic brokersInfo

现在是一个集群了，查看下topic在集群中的情况

bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic brokersInfo



输出解释：第一行是所有分区的摘要，其次，每一行提供一个分区信息，因为我们只有一个分区，所以只有一行。

"leader"：该节点负责该分区的所有的读和写，每个节点的leader都是随机选择的。

"replicas"：备份的节点列表，无论该节点是否是leader或者目前是否还活着，只是显示。

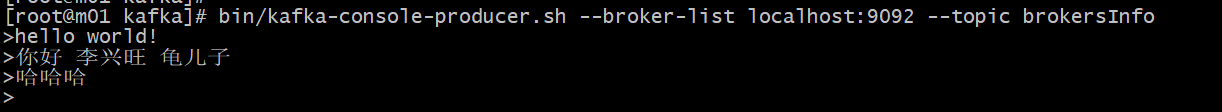
"isr"：“同步备份”的节点列表，也就是活着的节点并且正在同步leader。

### 04、生产/消费消息

生产者 生产消息 和消费者消费消息和单点服务一样，生产者 需要指定生产的消息放在哪个broker上。

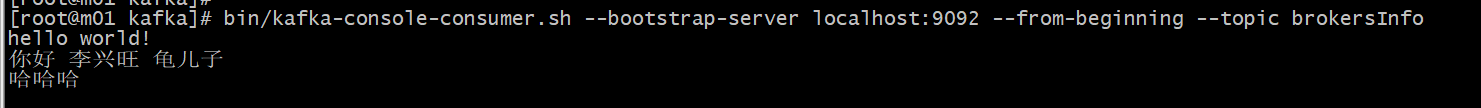
生产者 发送消息

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic brokerInfo



消费者消费消息

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



### 05、模拟broker故障

当前broker的leader是 broker.id=1的服务器，现在我们强制杀掉该服务进程。观察

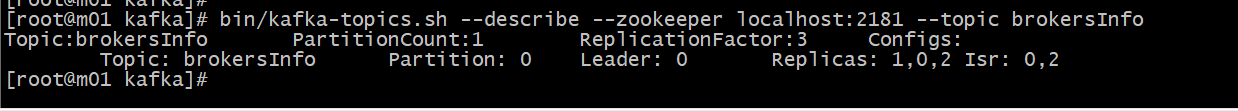
[root@m01 ~]# netstat -lntup|grep 9093

tcp 0 0 :::9093 :::\* LISTEN 4396/java

[root@m01 ~]# kill -9 4396

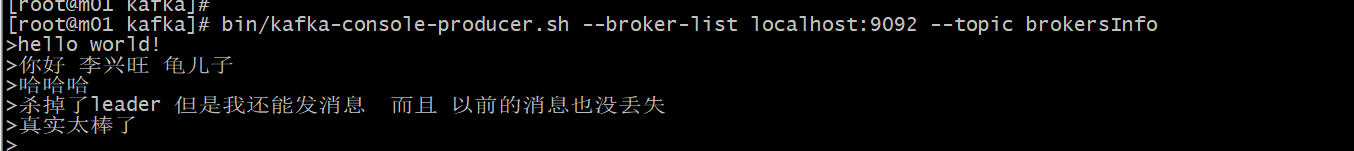
再次查看topic 的情况

bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic brokersInfo

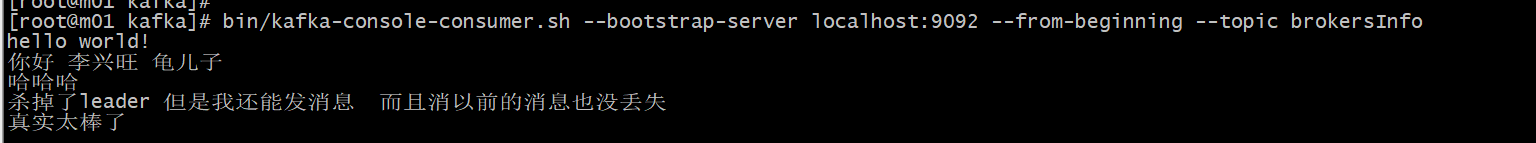


备份节点0成为新的leader，而broker1已经不在同步备份集合里了。

继续发送消息



消费端以前的消息也并没有丢失



至此集群的搭建已经完成了。 具体在生产环境中应用 文件配置 都需要在java里面实现了。

## 6、构建zk集群+kafka集群（多机实现）

### 01、环境准备

为了保证集群服务的高可用，需要搭建zookeeper集群和kafka集群，真正将每个应用部署在多台机器上，为了简单，只创建三台机器，这三台机器分别由三个zookeeper和三个kafka应用组成。以下是机器信息

|  |  |
| --- | --- |
| 机器IP | 安装服务 |
| 172.16.1.31 | Zookeeper-1 kafak-1 |
| 172.16.1.41 | Zookeeper-2 kafak-2 |
| 172.16.1.42 | Zookeeper-3 kafak-3 |

在这三台机器上要安装zk集群和kafka集群，默认这三台机器上已经安装过jdk 了。

### 02、搭建zookeeper集群

下载zookeeper到/tools目录下

wget <https://archive.apache.org/dist/zookeeper/zookeeper-3.4.11/zookeeper-3.4.11.tar.gz>

需要将该压缩包通过scp拷贝到其他机器

scp [root@172.16.1.41:/tools/zookeeper-3.4.11.tar.gz](mailto:root@172.16.1.41:/tools/zookeeper-3.4.11.tar.gz) /tools

scp [root@172.16.1.42:/tools/zookeeper-3.4.11.tar.gz](mailto:root@172.16.1.42:/tools/zookeeper-3.4.11.tar.gz) /tools

需要输入密码，然后所有三台机器都有该文件了

以下操作需要在三台机器执行一遍，这里以172.16.1.31为例执行

解压文件

tar -xvf zookeeper-3.4.11.tar.gz

在/application目录下创建软连接

ln -s /tools/zookeeper-3.4.1 /application/zookeeper

配置集群环境

因为zk默认会将zoo.cfg作为配置文件，拷贝一份

cp /application/zookeeper/conf/zoo\_sample.cfg /application/zookeeper/conf/zoo.cfg

编辑配置文件

vim zoo.cfg

修改的配置信息如下：

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper/zkdata

dataLogDir=/data/zookeeper/zkdatalog

clientPort=2181

//此处的IP就是你所操作的三台虚拟机的IP地址，每台虚拟机的zoo.cfg中都需要填入这三个地址。第一个端口是master和slave之间的通信端口，默认是2888，第二个端口是leader选举的端口，集群刚启动的时候选举或者leader挂掉之后进行新的选举的端口默认是3888

server.1=10.0.0.31:2888:3888

server.2=10.0.0.41:2888:3888

server.3=10.0.0.42:2888:3888

// server.1 这个1是服务器的标识也可以是其他的数字， 表示这个是第几号服务器，用来标识服务器，这个标识要写到快照目录下面myid文件里

新建文件夹

mkdir -p /data/zookeeper/zkdata

mkdir /data/zookeeper/zkdatalog

分别在三台机器上将zk节点id写到该文件中

echo "1" >/data/zookeeper/zkdata/myid

echo "2" >/data/zookeeper/zkdata/myid

echo "3" >/data/zookeeper/zkdata/myid

启动zk服务

/application/zookeeper/bin/zkServer.sh start

查看zk状态

/application/zookeeper/bin/zkServer.sh status

如果启动正常，会显示如下内容

ZooKeeper JMX enabled by default

Using config: /application/zookeeper/bin/../conf/zoo.cfg

Mode: follower

Mode 会有leader 和 follower，这代表zk集群已经搭建成功。

### 03、搭建kafak集群

上面的步骤已经下载过kafka并且安装过了，所以这里只需要更改kafka的配置文件即可。 安装的kafak位于/application/kafak 目录下，主看看config目录

我们主要关心的文件只有一个：**server.properties**,我们可以发现在这个目录下有很多文件，还有zookeeper的文件。我们可以直接使用kafka自带的zookeeper集群来启动，但这次我们使用外部的zk集群来搭建环境。

编辑配置文件

vim /application/kafka/config/server.properties

broker.id=1

/\* 这是这台虚拟机上的值，在另外两台虚拟机上应该是2或者3，这个值是唯一的，每台虚拟机或者叫服务器不能相同\*/

这个IP地址也是与本机相关的，每台服务器上设置为自己的IP地址。 /

listeners=PLAINTEXT://10.0.0.31:9092

log.dirs=/data/kafak/kafka-logs

//在og.retention.hours=168下面新增下面三项

message.max.byte=5242880

default.replication.factor=2

replica.fetch.max.bytes=5242880

/\*\*设置zookeeper的连接端口，新版本的kafka不再使用zookeeper而是通过brokerlist的配置让producer直接连接broker，这个brokerlist可以配置多个，只要有一个能连接上，就可以让producer获取道集群中的其他broker的信息，绕过了zookeeper。因此这个zookeeper.connect可以设置多个值\*/

zookeeper.connect=10.0.0.31:2181, 10.0.0.41:2181, 10.0.0.42:2181

其他两台机器的配置也要更改，把这台机器上的配置分别拷贝到其他两台机器

scp root@10.0.0.31:/application/kafka/config/server.properties /application/kafka/config/server.properties

将配置中与机器相关的信息更改成本机ip即可。

分别开启集群服务

cd /application/kafka/

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

### 04、验证

进入工作目录

cd /application/kafka/bin/

创建一个topic

./kafka-topics.sh --create --zookeeper 10.0.0.41:2181 --replication-factor 2 --partitions 3 --topic my-topic

参数说明

--replication-factor 2 // 复制两份  
--partitions 1 // 创建1个分区  
--topic // 主题为my-topic  
-- --zookeeper // 此处为为zookeeper监听的地址

开启一个生产者

./kafka-console-producer.sh --broker-list 10.0.0.42:9092 --topic my-topic

开启一个消费者

./kafka-console-consumer.sh --bootstrap-server=10.0.0.42:9092 --topic my-topic --from-beginning

两边可以正常发送消息。

### 07、安装KafkaOffsetMonitor

KafkaOffsetMonitor是Kafka的一款客户端消费监控工具，用来实时监控Kafka服务的Consumer以及它们所在的Partition中的Offset，我们可以浏览当前的消费者组，并且每个Topic的所有Partition的消费情况都可以一目了然。

KafkaOffsetMonitor托管在Github上，可以通过Github下载。

下载地址：

https://github.com/quantifind/KafkaOffsetMonitor/releases

直接下载到linux的/tools目录下

wget https://github.com/quantifind/KafkaOffsetMonitor/releases/download/v0.2.1/KafkaOffsetMonitor-assembly-0.2.1.jar

新建文件夹

mkdir /application/KafkaOffsetMonitor

将jar移动到该目录下

mv /tools/KafkaOffsetMonitor-assembly-0.2.1.jar /application/KafkaOffsetMonitor/

进入目录 启动

cd /application/KafkaOffsetMonitor/

java -cp \

KafkaOffsetMonitor-assembly-0.2.1.jar com.quantifind.kafka.offsetapp.OffsetGetterWeb \

--port 8088 \

--zk 10.0.0.31:2181,10.0.0.41:2181,10.0.0.42:2181 \

--refresh 10.seconds \

--retain 1.day >/dev/null 2>&1;

在浏览器输入 10.0.0.31:8088 s即可访问

当我进行访问的时候，发现只有一个黑色的背景，一些数据都没有，经过调试发现，原来由于网络限制引用不到angular.js，没办法只能用能访问到的js了，我找到了这个地址<https://code.angularjs.org/1.2.9,修改jar包里面的index.html>的引用就行了。

unzip KafkaOffsetMonitor-assembly-0.2.1.jar

修改html的js引入地址

vim offsetapp/index.html

将引入 angular.js的地址更改成如下

<script src="https://code.angularjs.org/1.2.9/ angular.js"></script>

<script src="https://code.angularjs.org/1.2.9/angular-route.js"></script>

<script src="https://code.angularjs.org/1.2.9/angular-resource.js"></script>

重新打jar

zip -r KafkaOffsetMonitor-assembly-0.2.1.jar ./

再次启动也界面会由内容了。