**kafka 服务搭建**

**版本：V1.1**

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制人: | 渠双双 | 日期: | 2016-10-20 |
| 审核人: |  | 日期: |  |
| 批准人: |  | 日期: |  |

**南京国通智能科技有限公司**

南京国通智能科技有限公司对本文件资料享有著作权及其它专属权利，未经书面许可，不得将该等文件资料（其全部或任何部分）披露予任何第三方，或进行修改后使用。

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **作者** | **修订日期** | **修订内容** | **批准人** |
| 1.0 | 渠双双 | 2016-10-20 | 初稿形成，主要内容部署和安装过程集成在一起，形成标准文档 | - |
| 1.1 | 渠双双 | 2016-10-21 | 修改后台启动kafka |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

目录

[1. 环境 3](#_Toc464735295)

[2. 安装要求 3](#_Toc464735296)

[3. 架构图 4](#_Toc464735297)

[4. 部署图 4](#_Toc464735298)

[5. 安装过程 4](#_Toc464735299)

[5.1. 安装kafka 4](#_Toc464735301)

[5.2. 配置文件 5](#_Toc464735302)

[5.3. 启动服务 5](#_Toc464735303)

[5.4. 创建 topic 5](#_Toc464735304)

[5.5. 发送消息. 5](#_Toc464735305)

[5.6. 启动consumer 5](#_Toc464735306)

[5.7. 搭建一个多个broker的集群 6](#_Toc464735307)

[6. kafka监控 7](#_Toc464735308)

[7. java与kafka的连接示例 9](#_Toc464735309)

[8. 可用性测试结果 10](#_Toc464735310)

# 环境

* kafka机器三台：机器A、机器B、机器C

操作系统：centos7.0 64位(不能用32位)

系统配置：

cpu数量：4核

内存：16G

硬板储存：500G

* kafka监控机器一台：机器D

操作系统：centos7.0 64位(不能用32位)

系统配置：

cpu数量：4核

内存：16G

硬盘大小：1T

# 安装要求

* jdk1.8
* zookeeper集群已经启动

# 架构图



# 部署图



# 安装过程



## 安装kafka

wget [**http://mirrors.cnnic.cn/apache/kafka/0.9.0.1/kafka\_2.11-0.9.0.1.tgz**](http://mirrors.cnnic.cn/apache/kafka/0.9.0.1/kafka_2.11-0.9.0.1.tgz)

**tar –zxvf kafka\_2.11-0.9.0.1.tgz**

## 配置文件

在sever.properties文件修改zookeeper的地址

## 启动服务

首先建立shell脚本,

vim start\_kafka.sh

输入以下内容：



#!/bin/bash

bash /home/gtarch/kafka/kafka\_2.11-0.9.0.1/bin/kafka-server-start.sh /home/gtarch/kafka/kafka\_2.11-0.9.0.1/config/server.properties &

脚本赋值权限

chmod +x start\_kafka.sh

启动脚本 ./start\_kafka.sh

## 创建 topic

创建一个叫做“test”的topic，它只有一个分区，一个副本。

> **bin/kafka-topics.sh --create --zookeeper** 127.0.0.1**:2181,**127.0.0.1**:2182,**127.0.0.1**:2183 --replication-factor 1 --partitions 1 --topic test**

可以通过list命令查看创建的topic:

> **bin/kafka-topics.sh --list --zookeeper** 127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183**

test

除了手动创建topic，还可以配置broker让它自动创建topic.

## 发送消息.

Kafka 使用一个简单的命令行producer，从文件中或者从标准输入中读取消息并发送到服务端。默认的每条命令将发送一条消息。

运行producer并在控制台中输一些消息，这些消息将被发送到服务端：

> **bin/kafka-console-producer.sh --broker-list localhost:9092,localhost:9093,localhost:9094 --topic test**

**This is a messageThis is another message**

ctrl+c可以退出发送。

## 启动consumer

Kafka也有一个命令行consumer可以读取消息并输出到标准输出：

> **bin/kafka-console-consumer.sh --zookeeper** 127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183 --topic test --from-beginning**

This is a message

This is another message

你在一个终端中运行consumer命令行，另一个终端中运行producer命令行，就可以在一个终端输入消息，另一个终端读取消息。

这两个命令都有自己的可选参数，可以在运行的时候不加任何参数可以看到帮助信息。

## 搭建一个多个broker的集群

刚才只是启动了单个broker，现在启动有3个broker组成的集群，

首先为每个节点编写配置文件：

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

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

在拷贝出的新文件中添加以下参数：

config/server-1.properties:

broker.id=1

port=9092

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

config/server-2.properties:

broker.id=2

port=9093

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

config/server-3.properties:

broker.id=3

port=9094

log.dir=/tmp/kafka-logs-3

broker.id在集群中唯一的标注一个节点，因为在同一个机器上，所以必须制定不同的端口和日志文件，避免数据被覆盖。

刚才已经启动Zookeeper和一个节点，现在启动另外两个节点：

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

...

> **bin/kafka-server-start.sh config/server-3.properties &**

...

创建一个拥有3个副本的topic:

> **bin/kafka-topics.sh --create --zookeeper** 127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183 --replication-factor 3 --partitions 1 --topic my-replicated-topic**

现在我们搭建了一个集群，怎么知道每个节点的信息呢？运行“"describe topics”命令就可以了：

> **bin/kafka-topics.sh --describe --zookeeper** 127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183 --topic my-replicated-topic**

Topic:my-replicated-topic PartitionCount:1 ReplicationFactor:3 Configs:

Topic: my-replicated-topic Partition: 0 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0

下面解释一下这些输出。第一行是对所有分区的一个描述，然后每个分区都会对应一行，因为我们只有一个分区所以下面就只加了一行。

leader：负责处理消息的读和写，leader是从所有节点中随机选择的.

replicas：列出了所有的副本节点，不管节点是否在服务中.

isr：是正在服务中的节点.

节点1是作为leader运行。

向topic发送消息：

> **bin/kafka-console-producer.sh --broker-list** 127.0.0.1**:9092,** 127.0.0.1**: 9093,** 127.0.0.1**:9094 --topic my-replicated-topic**

...

**my test message 1my test message 2^C**

消费这些消息：

> **bin/kafka-console-consumer.sh --zookeeper** 127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183 --from-beginning --topic my-replicated-topic**

...

my test message 1

my test message 2

**^C**

测试一下容错能力.Broker 1作为leader运行，现在kill掉它：

> **ps | grep server-1.properties***7564* ttys002 0:15.91 /System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home/bin/java...

> **kill -9 7564**

另外一个节点被选做了leader,node 1 不再出现在 in-sync 副本列表中：

> **bin/kafka-topics.sh --describe --zookeeper**127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183 --topic my-replicated-topic**

Topic:my-replicated-topic PartitionCount:1 ReplicationFactor:3 Configs:

Topic: my-replicated-topic Partition: 0 Leader: 2 Replicas: 1,2,0 Isr: 2,0

虽然最初负责续写消息的leader down掉了，但之前的消息还是可以消费的：

> **bin/kafka-console-consumer.sh --zookeeper** 127.0.0.1**:2181,** 127.0.0.1**:2182,** 127.0.0.1**:2183 --from-beginning --topic my-replicated-topic**

...

my test message 1

my test message 2

# kafka监控

资料都在http://192.168.1.136/svn/3.4.1/人员/技术类/渠双双/技术架构/kafka

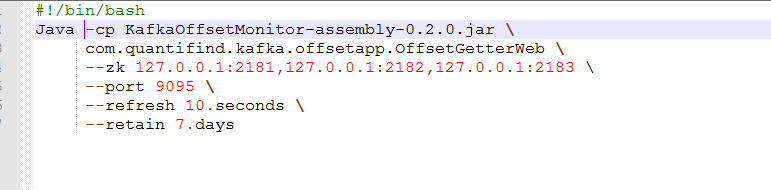
1. 下载

[KafkaOffsetMonitor](https://github.com/quantifind/KafkaOffsetMonitor/releases/download/v0.2.0/KafkaOffsetMonitor-assembly-0.2.0.jar).jar 存放在相应的目录

1. 安装

建立脚本start.sh

脚本内容如下：



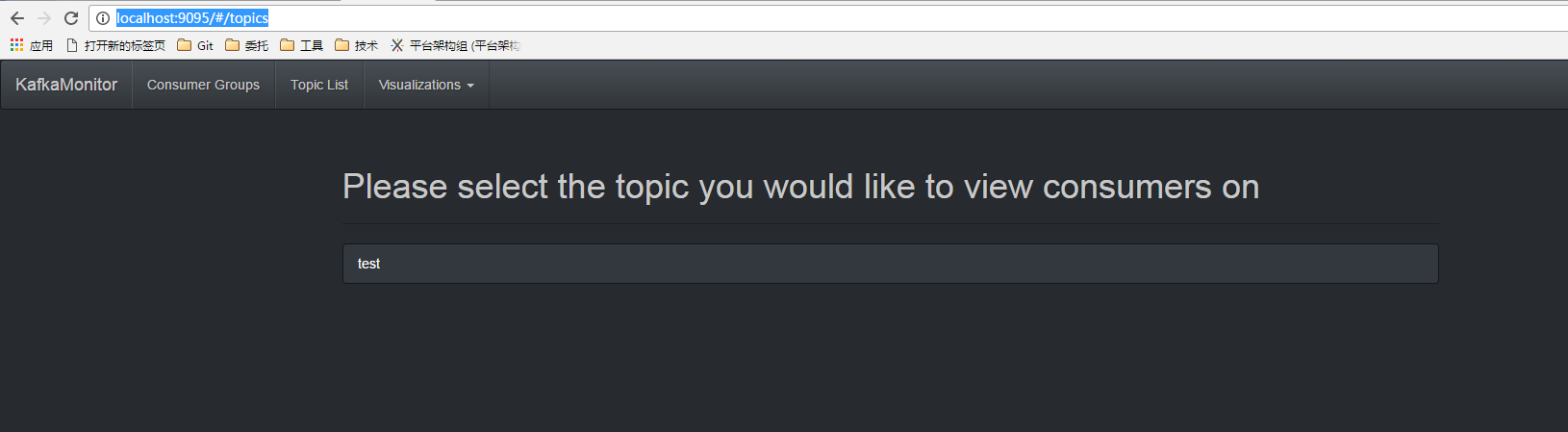
1. 运行

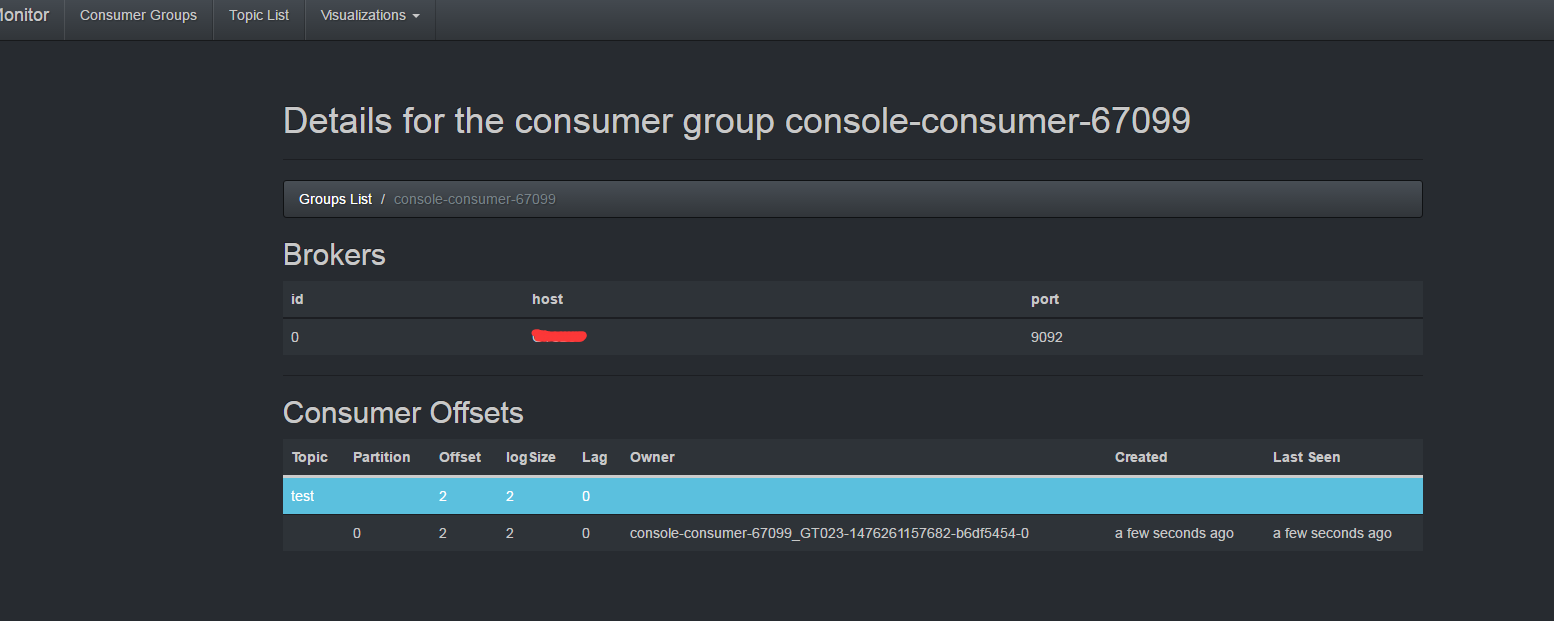
若为linux系统 给这个脚本赋予权限 如：chmod +x start.sh

运行该脚本。

1. 访问

地址 <http://localhost:9095/#/topics>





**以上图中参数含义解释如下：**

topic：创建时topic名称

partition：分区编号

offset：表示该parition已经消费了多少条message

logSize：表示该partition已经写了多少条message

Lag：表示有多少条message没有被消费。

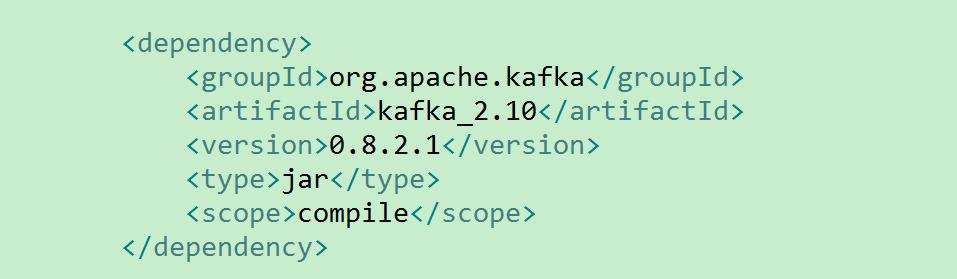
Owner：表示消费者

Created：该partition创建时间

Last Seen：消费状态刷新最新时间。

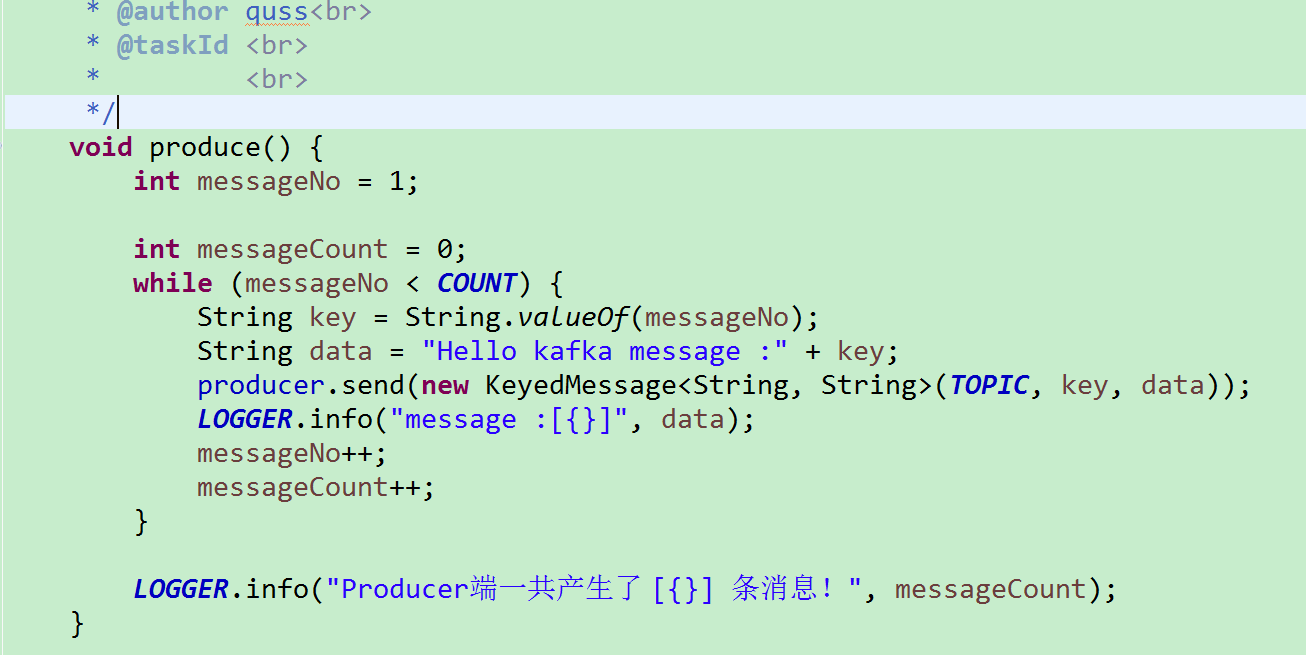
# java与kafka的连接示例

配置pom文件：

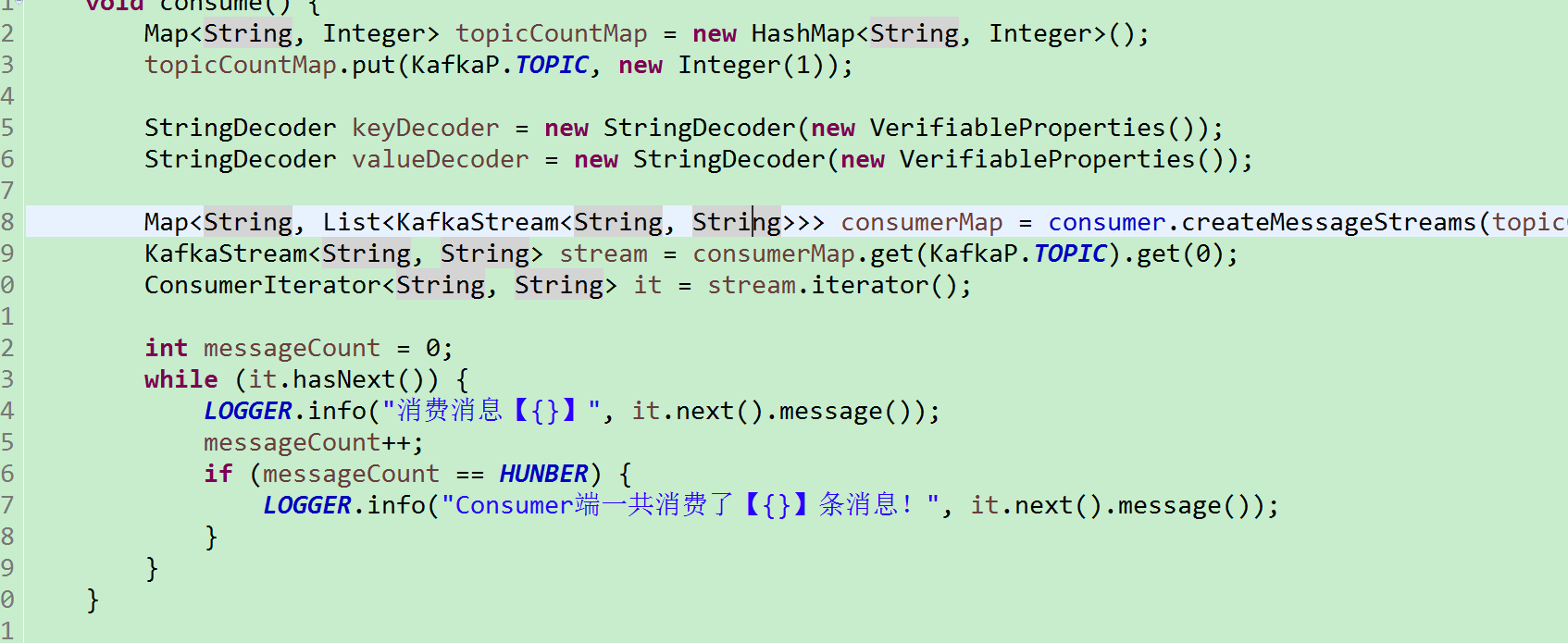


注：依赖0.9.0.1的pom，无法消费，所以降低版本

生产者：



消费者：



主要是核心内容。

# 可用性测试结果