

1 Kafka 入门

1.1 学习目标

1. 掌握kafka是什么，主要用途是什么，了解kafka的特性
2. 掌握kafka集群安装
3. 掌握kafka核心概念、工作原理
4. 掌握kafka的使用

1.2 简介



中文介绍文档: <http://kafka.apachecn.org/>

1.2.1 Kafka是什么？

- 一个分布式的流式数据处理平台。
- 可以用它来发布和订阅流式的记录。这一方面与消息队列或者企业消息系统类似
- 它将流式的数据安全地存储在分布式、有副本备份、容错的集群上
- 可以用来做流式计算

1.2.2 Kafka适合什么样的场景？

它可以用于两大类的应用:

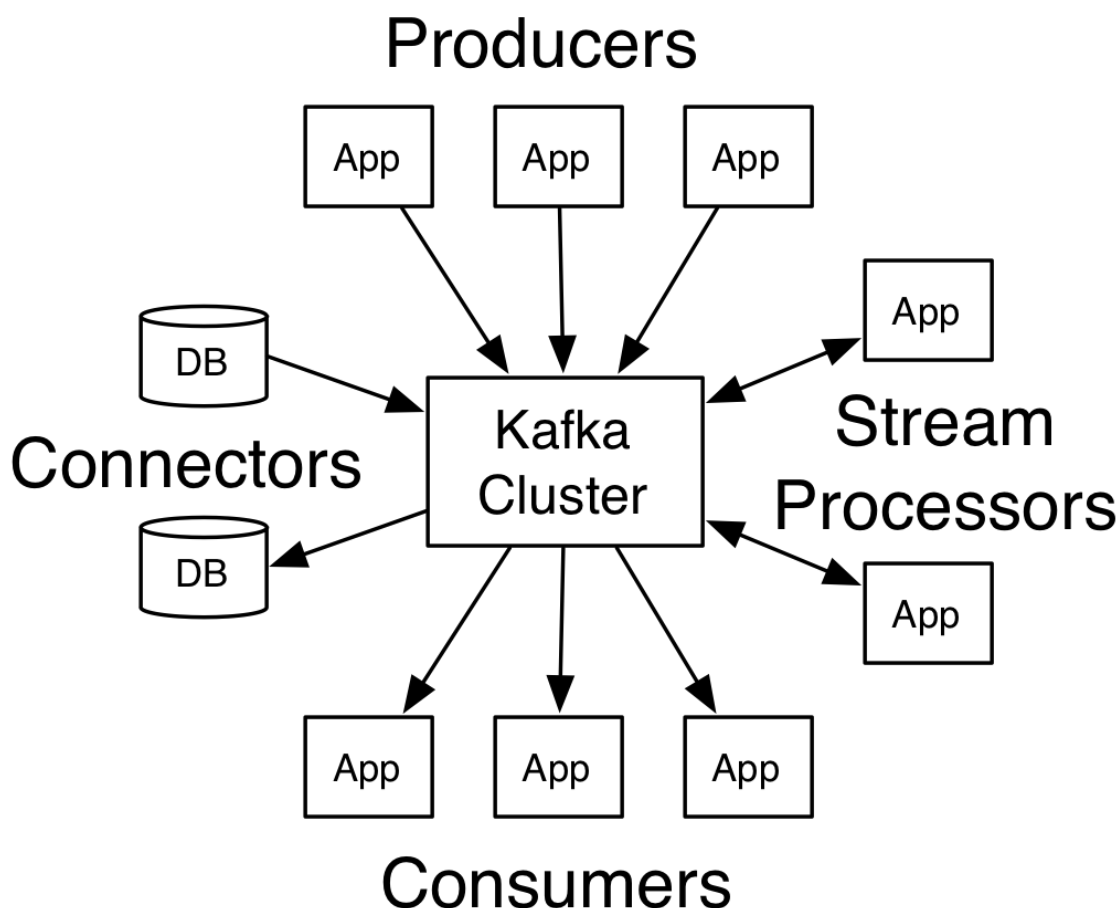
1. 构造实时流数据管道，它可以在系统或应用之间可靠地获取数据。(相当于message queue)
2. 构建实时流式应用程序，对这些流数据进行转换或者影响。(就是流处理，通过kafka stream topic和topic之间内部进行变化)

1.2.3 Kafka的架构体系

分布式集群

- Kafka作为一个集群，运行在一台或者多台服务器上。
- Kafka 通过 *topic* 对存储的流数据进行分类。

Kafka有四个核心的API:



- The [Producer API](#) 允许一个应用程序发布一串流式的数据到一个或者多个Kafka topic。
- The [Consumer API](#) 允许一个应用程序订阅一个或多个 topic，并且对发布给他们的流式数据进行处理。
- The [Streams API](#) 允许一个应用程序作为一个流处理器，消费一个或者多个topic产生的输入流，然后生产一个输出流到一个或多个topic中去，在输入输出流中进行有效的转换。
- The [Connector API](#) 允许构建并运行可重用的生产者或者消费者，将Kafka topics连接到已存在的应用程序或者数据系统。比如，连接到一个关系型数据库，捕捉表（table）的所有变更内容。

在Kafka中，客户端和服务端使用一个简单、高性能、支持多语言的 [TCP 协议](#)。此协议版本化并且向下兼容老版本，我们为Kafka提供了Java客户端，也支持许多[其他语言的客户端](#)。

1.3 安装

环境要求

- 生产环境，强烈要求 linux，学习可以windows
- java1.8 或以上

安装

1. 下载安装包: https://www.apache.org/dyn/closer.cgi?path=/kafka/2.3.0/kafka_2.12-2.3.0.tgz

windows 和 linux都是同一个安装包，window命令在 bin/windows/ 下。

2. 安装

```
[root@node4 ~]# mkdir /usr/kafka
[root@node4 ~]#
[root@node4 ~]# tar -xzf kafka_2.12-2.3.0.tgz -C /usr/kafka/
[root@node4 ~]# ln -s /usr/kafka/kafka_2.12-2.3.0 /usr/kafka/latest
```

3. 了解目录结构

```
[root@node4 ~]# ll /usr/kafka/latest
drwxr-xr-x. 3 root root 4096 6月 20 04:44 bin
drwxr-xr-x. 2 root root 4096 6月 20 04:44 config
drwxr-xr-x. 2 root root 4096 8月 24 17:32 libs
-rw-r--r--. 1 root root 32216 6月 20 04:43 LICENSE
-rw-r--r--. 1 root root 337 6月 20 04:43 NOTICE
drwxr-xr-x. 2 root root 44 6月 20 04:44 site-docs
```

了解 bin、config、libs 下都有些什么。

4. 开启zookeeper服务

Kafka集群使用zookeeper来存储元信息，在生产环境下你应该独立安装一个zookeeper集群。学习可以使用Kafka安装包中内置的zookeeper，启动一个单实例的zookeeper

```
[root@node4 ~]# cd /usr/kafka/latest/
[root@node4 latest]# bin/zookeeper-server-start.sh
config/zookeeper.properties &
```

5. 配置Kafka， kafka的配置文件 config/server.properties

要掌握的基本配置项：

```
# The id of the broker. This must be set to a unique integer for each
broker.
broker.id=0          # 集群中每个broker的唯一整数 id

# The address the socket server listens on. It will get the value returned
from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
# 服务端口，默认9092
#listeners=PLAINTEXT://:9092

# Hostname and port the broker will advertise to producers and consumers. If
not set,
# it uses the value for "listeners" if configured.  Otherwise, it will use
the value
# returned from java.net.InetAddress.getCanonicalHostName().
```

```

# broker发布自己的地址给生产者和消费用，不配做则使用配置的listeners值，如果没有配置
listeners则取主机名。这里建议配置IP，否则客户端通过主机名连接可能连不通。
#advertised.listeners=PLAINTEXT://your.host.name:9092


# A comma separated list of directories under which to store log files
#log.dirs=/tmp/kafka-logs      # 数据存储目录，逗号间隔的多个目录，一定要修改为你想要存
放的目录。
log.dirs=/var/kafka-logs


# The default number of log partitions per topic. More partitions allow
greater
# parallelism for consumption, but this will also result in more files
across
# the brokers.
# 当创建Topic(主题)未指定分区数时的默认分区数，
num.partitions=1


##### Zookeeper #####

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=localhost:2181


# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=6000


##### Group Coordinator Settings
#####

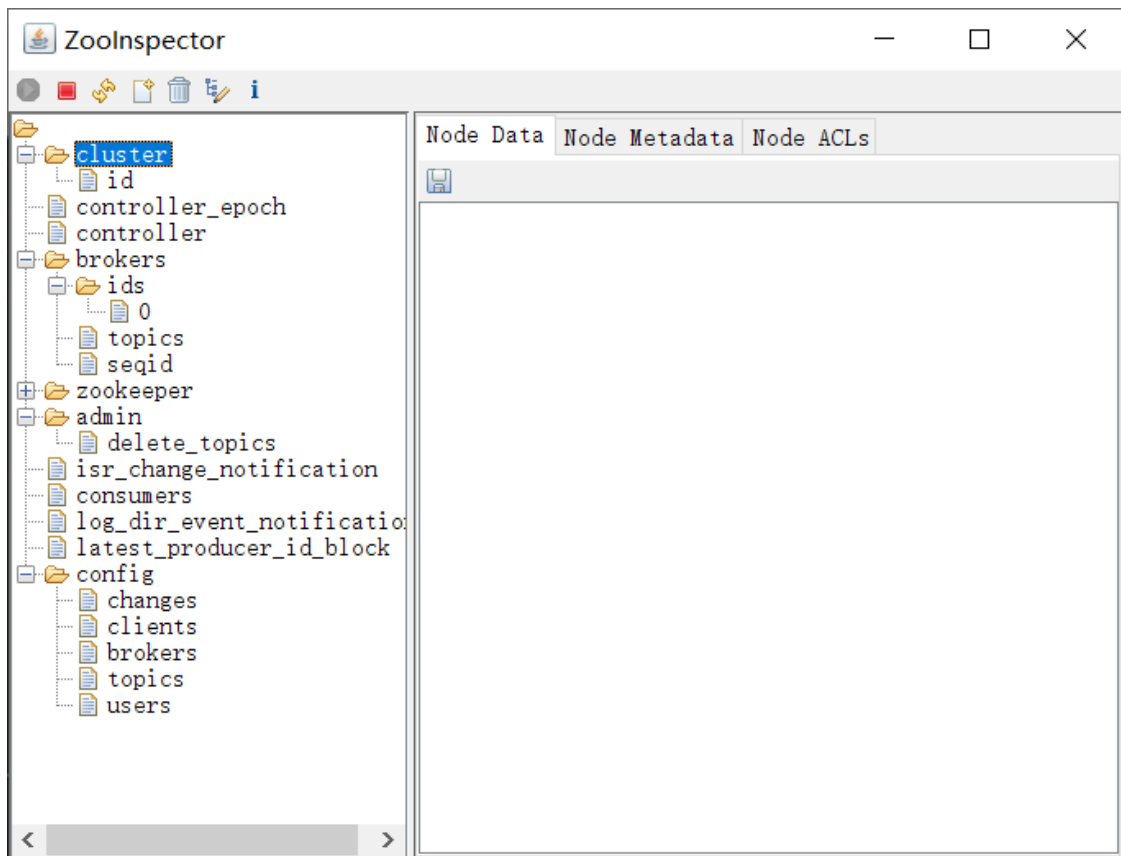
# The following configuration specifies the time, in milliseconds, that the
GroupCoordinator will delay the initial consumer rebalance.
# The rebalance will be further delayed by the value of
group.initial.rebalance.delay.ms as new members join the group, up to a
maximum of max.poll.interval.ms.
# The default value for this is 3 seconds.
# We override this to 0 here as it makes for a better out-of-the-box
experience for development and testing.
# However, in production environments the default value of 3 seconds is more
suitable as this will help to avoid unnecessary, and potentially expensive,
rebalances during application startup.
# 消费组的重平衡延时（单位毫秒），【注意】生产环境请恢复为默认值3秒，或根据实际需要增大
group.initial.rebalance.delay.ms=0

```

6. 启动Kafka broker 实例

```
[root@node4 latest]# bin/kafka-server-start.sh config/server.properties &
```

接下来我们可以通过zookeeper客户端去查看kafka集群信息



集群搭建

【生产集群】在其他机器上同样安装kafka，配置它们连接到同一个zookeeper集群、它们的唯一id，数据目录，启动Broker实例即加入集群。

【学习用集群】在同一台机器上启动多个broker实例，按如下步骤操作来搭建一个3节点的集群：

学习用集群搭建

1. 拷贝配置文件

```
[root@node4 latest]# cp config/server.properties config/server-1.properties
[root@node4 latest]# cp config/server.properties config/server-2.properties
```

2. 修改配置文件：

config/server-1.properties:

```
broker.id=1
listeners=PLAINTEXT://:9093
log.dirs=/var/kafka-logs-1
```

config/server-2.properties:

```
broker.id=2
listeners=PLAINTEXT://:9094
log.dirs=/var/kafka-logs-2
```

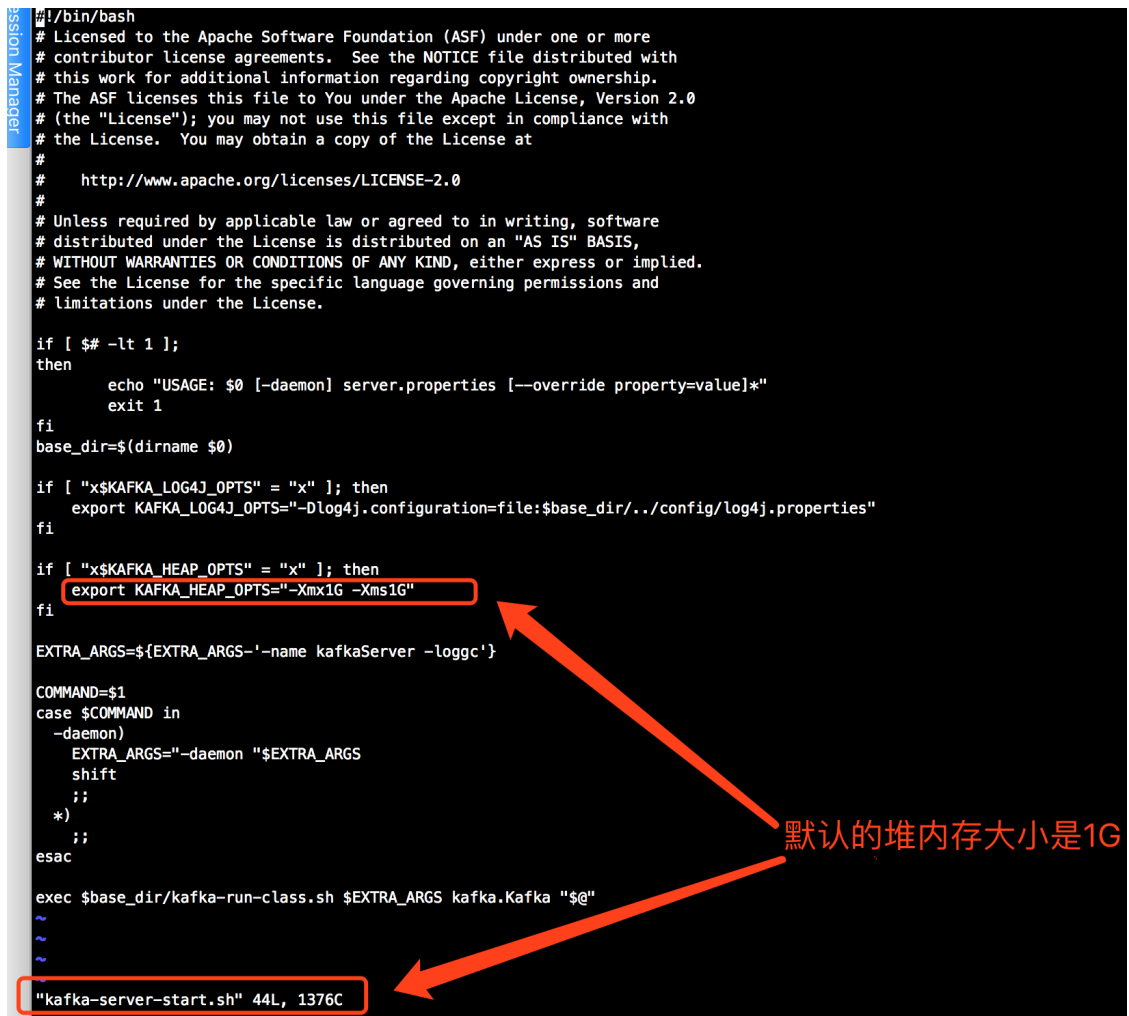
3. 启动这两个broker实例

```
[root@node4 latest]# bin/kafka-server-start.sh config/server-1.properties &
[root@node4 latest]# bin/kafka-server-start.sh config/server-2.properties &
```

【启动失败说明】 如果启动第二个或第三个broker时提示内存不够用，可以做如下调整：

- 1、调大你的虚拟机的内存（1G 或更多）
- 2、调小Kafka的堆大小，默认是1G，生产用时可以调大。这里学习用可以调为256M（不能太小了，启动时会heap OOM）

```
[root@node4 latest]# vi bin/kafka-server-start.sh
```



```
#!/bin/bash
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

if [ $# -lt 1 ];
then
    echo "USAGE: $0 [-daemon] server.properties [--override property=value]*"
    exit 1
fi
base_dir=$(dirname $0)

if [ "x$KAFKA_LOG4J_OPTS" = "x" ]; then
    export KAFKA_LOG4J_OPTS="-Dlog4j.configuration=file:$base_dir/../../config/log4j.properties"
fi

if [ "x$KAFKA_HEAP_OPTS" = "x" ]; then
    export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"
fi

EXTRA_ARGS=${EXTRA_ARGS-'-name kafkaServer -loggc'}

COMMAND=$1
case $COMMAND in
    -daemon)
        EXTRA_ARGS="-daemon "$EXTRA_ARGS
        shift
        ;;
    *)
        ;;
    esac

exec $base_dir/kafka-run-class.sh $EXTRA_ARGS kafka.Kafka "$@"

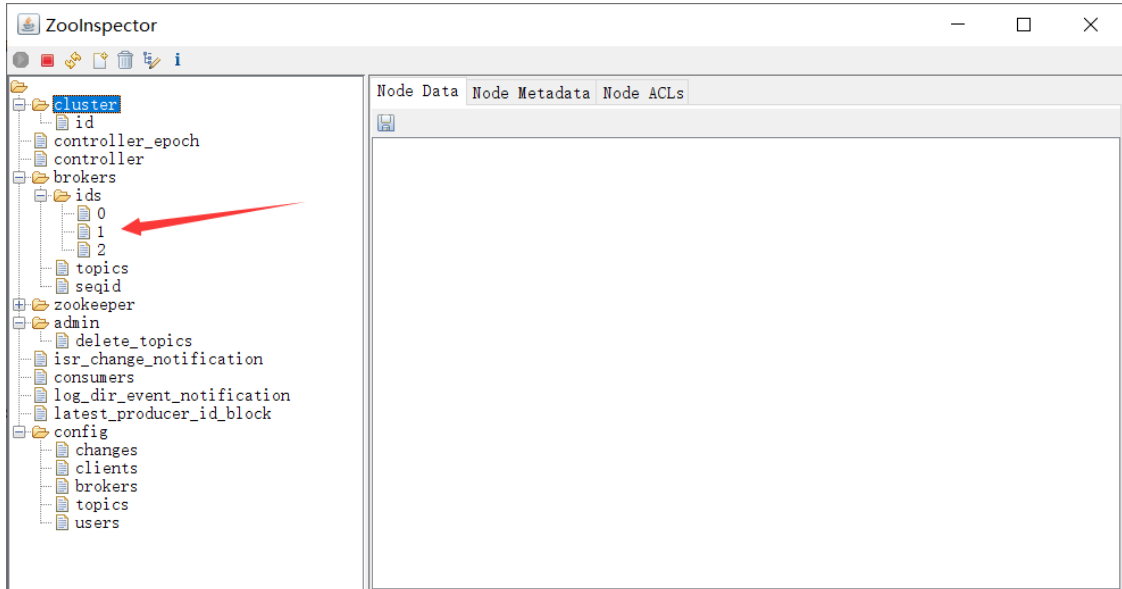
~
~
~
"kafka-server-start.sh" 44L, 1376C
```

默认的堆内存大小是1G

启动好后，可以查看启动的java进程，将看到3个Kafka，一个zookeeper

```
[root@node4 latest]# jps
4932 Kafka
3722 QuorumPeerMain
4570 Kafka
4060 Kafka
5295 Jps
```

再看看zookeeper上的kafka集群信息:



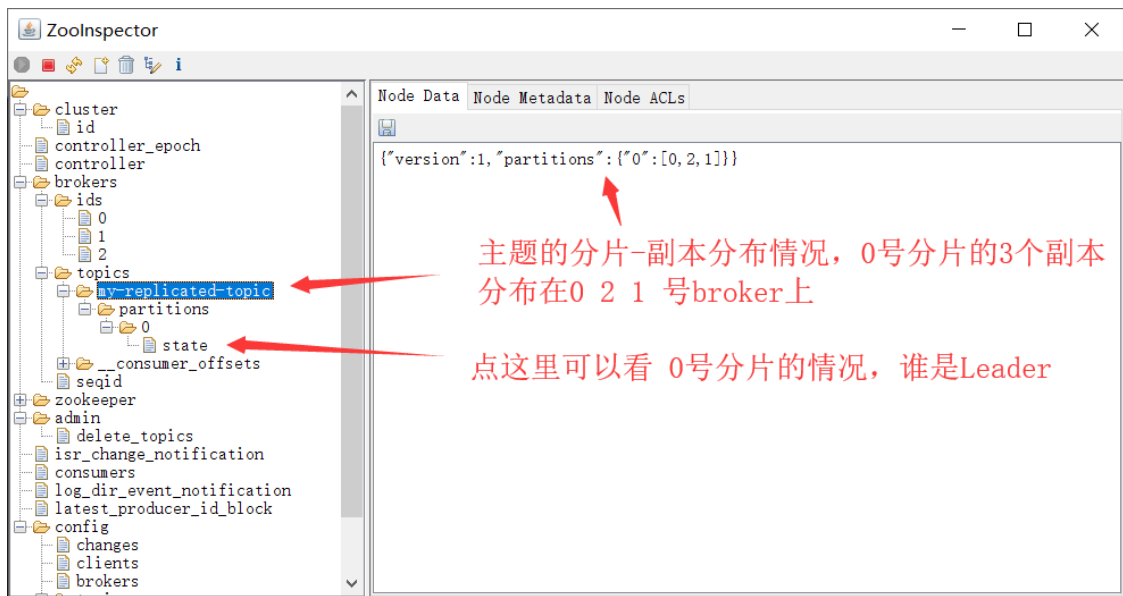
4. 可以玩了, 创建一个只有1个分片, 3个备份的Topic

```
[root@node4 latest]# bin/kafka-topics.sh --create --bootstrap-server
localhost:9092 --replication-factor 3 --partitions 1 --topic my-replicated-
topic
```

查看主题信息:

```
[root@node4 latest]# bin/kafka-topics.sh --describe --bootstrap-server
localhost:9092 --topic my-replicated-topic
Topic:my-replicated-topic PartitionCount:1 ReplicationFactor:3
Configs:segment.bytes=1073741824
Topic: my-replicated-topic Partition: 0 Leader: 0 Replicas: 0,2,1
Isr: 0,2,1
```

也可在zookeeper客户端上看:



【掌握】状态信息的含义

- "leader" is the node responsible for all reads and writes for the given partition. Each node will be the leader for a randomly selected portion of the partitions.
- "replicas" is the list of nodes that replicate the log for this partition regardless of whether they are the leader or even if they are currently alive.
- "isr" is the set of "in-sync" replicas. This is the subset of the replicas list that is currently alive and caught-up to the leader.

5. 现在让我们用Kafka安装包中提供的客户端程序来发布消息：

```
> bin/kafka-console-producer.sh --broker-list localhost:9092 --topic my-replicated-topic
...
my test message 1
my test message 2
^C
```

6. 来消费消息

```
> bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic my-replicated-topic
...
my test message 1
my test message 2
^C
```

7. 怎么停止Broker

```
[root@node4 latest]# bin/kafka-server-stop.sh
```

这会把我们刚才启动的三个broker都停掉。

8. 让我们来测试一下集群容错，现在主题 my-replicated-topic 的唯一分片的leader备份是0号节点，我们把0号broker停掉看看，找到它的进程号，直接kill。

```
> ps aux | grep server.properties
7564 ttys002    0:15.91
/System/Library/Frameworks/JavaVM.framework/Versions/1.8/Home/bin/java...
> kill -9 7564
```

再看看 主题 my-replicated-topic 的信息：【注意 不能连0号节点了】

```
[root@node4 latest]# bin/kafka-topics.sh --describe --bootstrap-server
localhost:9093 --topic my-replicated-topic
Topic:my-replicated-topic  PartitionCount:1      ReplicationFactor:3
Configs:segment.bytes=1073741824
      Topic: my-replicated-topic  Partition: 0      Leader: 2      Replicas: 0,2,1
Isr: 1,2
```

集群搭建完成。

1.4 监控管理

Kakfa自身未提供图形化的监控管理工具，市面上有很多开源的监控管理工具，但都不怎么成熟可靠。这里给介绍一款稍可靠的工具。

Kafka Offset Monitor

<https://github.com/quantifind/KafkaOffsetMonitor>

可以实时监控：

- Kafka集群状态
- Topic、Consumer Group列表
- 图形化展示topic和consumer之间的关系
- 图形化展示consumer的Offset、Lag等信息

它是一个jar 包，使用很简单

```
java -cp KafkaOffsetMonitor-assembly-0.2.1.jar \
com.quantifind.kafka.offsetapp.OffsetGetterWeb \
--offsetStorage kafka
--zk zk-server1,zk-server2 \
--port 8080 \
--refresh 10.seconds \
--retain 2.days
```

0.2.0 版本启动命令

```
java -cp KafkaOffsetMonitor-assembly-0.2.0.jar \
    com.quantifind.kafka.offsetapp.OffsetGetterWeb \
    --zk zk-server1,zk-server2 \
    --port 8088 \
    --refresh 10.seconds \
    --retain 2.days
```

The arguments are:

- **offsetStorage** valid options are "zookeeper", "kafka" or "storm". Anything else falls back to "zookeeper" 【说明】0.2.1版本才有这个参数
- **zk** the ZooKeeper hosts
- **port** on what port will the app be available
- **refresh** how often should the app refresh and store a point in the DB
- **retain** how long should points be kept in the DB
- **dbName** where to store the history (default 'offsetapp')
- **kafkaOffsetForceFromStart** only applies to "kafka" format. Force KafkaOffsetMonitor to scan the commit messages from start (see notes below)
- **stormZKOffsetBase** only applies to "storm" format. Change the offset storage base in zookeeper, default to "/stormconsumers" (see notes below)
- **pluginsArgs** additional arguments used by extensions (see below)

启动后就可以在浏览器中访问了: <http://localhost:8080>

1.5 Spring 中使用

spring 官网学习文档: <https://docs.spring.io/spring-kafka/docs/2.2.8.RELEASE/reference/html/#introduction>