**linux系统环境配置**

# zookeeper环境搭建

1. 拷贝zookeeper-3.4.5.tar.gz
2. 解压 tar -zxvf zookeeper-3.4.5.tar.gz
3. 拷贝配置文件 copy zoo\_sample.cfg zoo.cfg
4. 修改dataDir,dataLogDir, 可以添加server.1=localhost:2888:3888
5. 启动服务端:./zkServer.sh start jps查看可以看到QuorumPeerMain进程。
6. 启动zookeeper客户端 ./zkCli.sh

# kafka环境搭建

参考官网链接：http://kafka.apache.org/quickstart

## 单代理环境搭建

1. 先启动zookeeper 默认端口是2181
2. 启动kafka bin/kafka-server-start.sh config/server.properties 默认链接zookeeper端口是2181
3. 创建一个topic：

创建一个名字为”test”的topic，使用一个单一的paritition，和仅仅一次复制

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

控制台显示Created topic "test".

通过命令显示topic列表:

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

会显示test

1. 发送消息

Kafka带有一个命令行客户端，它将从文件或标准输入中获取输入，并将其作为消息发送到Kafka集群。默认情况下，每行将作为单独的消息发送。

运行生产者，然后在控制台输入一些消息发送到服务器。

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

This is a message

This is another message

1. 启动一个消费者

kafka也有一个命令行消费者，将消息转储到标准输出

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

控制台会打印出:

This is a message

This is another message

## 建立kafka集群

1. 为每个代理创建一个配置文件

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

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

1. 现在编辑这些新文件并设置以下属性：

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

该broker.id属性是群集中每个节点的唯一且永久的名称。我们必须重写端口和日志目录，因为我们在同一台机器上运行这些端口和日志目录，我们希望让所有的代理都试图在同一个端口注册或覆盖彼此的数据。

我们已经有Zookeeper和我们的单个节点，所以我们只需要启动两个新的节点：

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

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

现在创建一个复制因子为三的新主题：

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

好，但现在我们有一个集群，我们怎么知道哪个经纪人在做什么？要查看运行“描述主题”命令：

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

这里是对输出的解释。第一行给出了所有分区的摘要，每个附加行给出了关于一个分区的信息。由于我们只有一个分区，所以只有一行。

“leader”是负责给定分区的所有读取和写入的节点。每个节点将成为分区随机选择部分的领导者。

“Replicas”是复制此分区的日志的节点列表，无论它们是否是领导者，或者即使他们当前处于活动状态。

“isr”是一组“同步”副本。这是复制品列表的子集，当前活着并被引导到领导者。

请注意，在我的示例中，节点1是该主题的唯一分区的领导者。

我们可以在我们创建的原始主题上运行相同的命令来查看它的位置：

让我们发表一些信息给我们的新主题：

> bin/kafka-console-producer.sh --broker-list localhost:9092 --topic my-replicated-topic

my test message 1

my test message 2

现在让我们消费这些消息：

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

...

my test message 1

my test message 2

^C

现在我们来测试一下容错。经纪人1是作为领导者，所以让我们杀了它：

> ps aux | grep server-1.properties

7564 ttys002 0:15.91 /System/Library/Frameworks/JavaVM.framework/Versions/1.8/Home/bin/java...

> kill -9 7564

领导已经切换到其中一个从属节点，并且节点1不再处于同步副本集合中：

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

但是，即使原先写入的领导者失败，这些消息仍然可用于消费：

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

...

my test message 1

my test message 2

^C

## 使用Kafka Connect来导入/导出数据

从控制台写入数据并将其写回控制台是一个方便的起点，但您可能想要使用其他来源的数据或将数据从Kafka导出到其他系统。对于许多系统，您可以使用Kafka Connect来导入或导出数据，而不必编写自定义集成代码。

Kafka Connect是Kafka包含的一个工具，可以将数据导入和导出到Kafka。它是一个可扩展的工具，运行 *连接器*，实现与外部系统交互的自定义​​逻辑。在这个快速入门中，我们将看到如何使用简单的连接器运行Kafka Connect，这些连接器将数据从文件导入到Kafka主题，并将数据从Kafka主题导出到文件。

首先，我们将通过创建一些种子数据开始测试：

> echo -e "foo\nbar" > test.txt

接下来，我们将启动两个以独立模式运行的连接器，这意味着它们将在单个本地专用进程中运行。我们提供三个配置文件作为参数。首先是Kafka Connect过程的配置，包含常见的配置，例如要连接的Kafka代理以及数据的序列化格式。其余的配置文件都指定一个要创建的连接器。这些文件包括唯一的连接器名称，要实例化的连接器类以及连接器所需的任何其他配置。

> bin/connect-standalone.sh config/connect-standalone.properties config/connect-file-source.properties config/connect-file-sink.properties

Kafka附带的这些示例配置文件使用您之前启动的默认本地群集配置，并创建两个连接器：第一个是源连接器，用于从输入文件中读取行，并将每个连接生成为Kafka主题，第二个为连接器连接器它从Kafka主题读取消息，并在输出文件中产生每行消息。

在启动过程中，您会看到一些日志消息，其中一些指示连接器正在实例化。一旦Kafka Connect过程开始，源连接器应该开始读取线路test.txt并将它们生成为主题connect-test，并且接收器连接器应该开始从主题读取消息connect-test 并将其写入文件test.sink.txt。我们可以通过检查输出文件的内容来验证通过整个管道传输的数据：

> more test.sink.txt

foo

bar

请注意，数据存储在Kafka主题中connect-test，因此我们也可以运行控制台使用者来查看主题中的数据（或使用自定义使用者代码来处理它）：

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

{"schema":{"type":"string","optional":false},"payload":"foo"}

{"schema":{"type":"string","optional":false},"payload":"bar"}

...

连接器继续处理数据，所以我们可以将数据添加到文件中，并看到它在管道中移动：

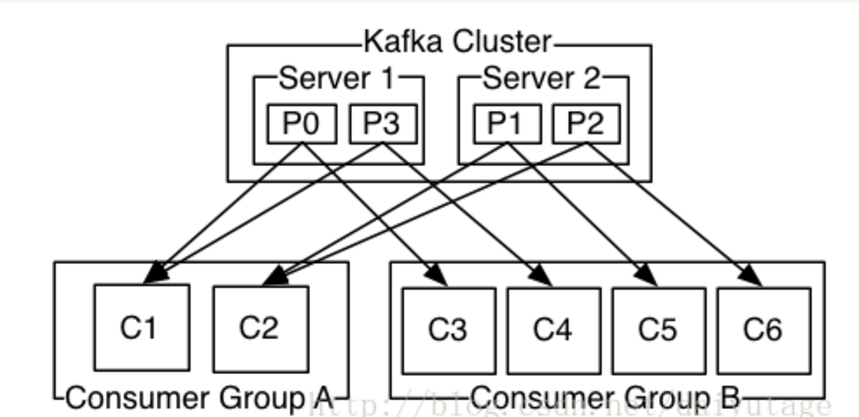
> echo Another line>> test.txt

您应该看到该行出现在控制台使用者输出和接收器文件中。

## 使用Kafka流来处理数据

# 关键属性解释

## partition



一个topic 可以配置几个partition，produce发送的消息分发到不同的partition中，consumer接受数据的时候是按照group来接受，kafka确保每个partition只能同一个group中的同一个consumer消费，如果想要重复消费，那么需要其他的组来消费。Zookeerper中保存这每个topic下的每个partition在每个group中消费的offset。

新版kafka把这个offsert保存到了一个\_\_consumer\_offsert的topic下

这个\_\_consumer\_offsert 有50个分区，通过将group的id哈希值%50的值来确定要保存到那一个分区. 这样也是为了考虑到zookeeper不擅长大量读写的原因。

所以，如果要一个group用几个consumer来同时读取的话，需要多线程来读取，一个线程相当于一个consumer实例。当consumer的数量大于分区的数量的时候，有的consumer线程会读取不到数据。

假设一个topic test 被groupA消费了，现在启动另外一个新的groupB来消费test，默认test-groupB的offset不是0，而是没有新建立，除非当test有数据的时候，groupB会收到该数据，该条数据也是第一条数据，groupB的offset也是刚初始化的ofsert, 除非用显式的用–from-beginnging 来获取从0开始数据

如上图对应的创建步骤如下

1. 启动两个broker

bin/kafka-server-start.sh config/server-1.properties 监听9093端口

bin/kafka-server-start.sh config/server-2.properties 监听9094端口

1. 创建4个分区的主题test

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 4 --topic test

2 启动一个生产者，往名为test的主题发送4条消息

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

This is a message1

This is a message2

This is a message3

This is a message4

3 创建6个消费者，如上图分属于两个组Custom-Group-A 和Custom-Group-B

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

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

查看消费者分组信息

bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group my-group

## replication-factor：

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

该命令行表示：用一个分区和一个副本创建一个名为“test”的主题：

参考链接

<https://www.cnblogs.com/rainwang/p/7496147.html> kafka消费者组

<https://www.cnblogs.com/liuwei6/p/6900686.html> partition与group