链滴

Kafka 入门安装

作者：flowaters

原文链接：https://ld246.com/article/1512317787095

来源网站：链滴

背景

Kafka和RabbitMQ均为消息队列产品，各自特点如下：

- 吞吐：Kafka >> RabbitMQ
- 可靠性：RabbitMQ > Kafka，RabbitMQ有消息确认机制，支持事务
- 可用性：Kafka支持主备，RabbitMQ支持queue的mirror

安装

参考自官方的quickStart

- 下载： http://kafka.apache.org/downloads
- 解压：tar xf kafka_2.11-1.0.0.tgz

启动Server

- 启动单机版Zookeeper

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

- 启动Kafka Server

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

topic

创建topic

创建一下test的topic，包含一个分区和一个分片

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

Created topic "test".
```

查看topic

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

test
```

发送消息

```
$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
>This is a message
>This is another message
```

订阅消息

```
$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-begin
ing
This is a message
This is another message
```

如果发送和订阅在两个终端，在发送消息时，可以在订阅终端实时看到消息

删除topic

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

多broker

略，详见 官方quickstart

使用kafka connect来导入导出数据

导入导出数据

生成测试数据

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

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

三个配置文件，分别为

- 集群配置文件
- source connector: 从input文件test.txt读入数据，到kafka
- sink connector: 从kafka读出数据，写入文件test.sink.txt

重新订阅数据

上面的测试数据保存在topic: connect-test中，可以从头再订阅数据

```
$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic connect-test --fr
m-beginning
{"schema":{"type":"string","optional":false},"payload":"foo"}
{"schema":{"type":"string","optional":false},"payload":"bar"}
```

继续写数据

```
$ echo "Another Line" >> test.txt
```

使用kafka stream来数据处理

- 官方quickstart

Java Client

pom

pom.xml

```
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka_2.10</artifactId>
            <version>0.10.2.1</version>
        </dependency>
```

producer

```
import  java.util.Properties;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class MainKafkaProducer {
    public static void main(String[] args) throws InterruptedException {
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");

        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        Producer producer = new KafkaProducer(props);

        for(int i = 0; i < 100; i++) {
            producer.send(new ProducerRecord("fw-blink-test", i % 1, Integer.toString(i), Integer.t
String(i)));
            Thread.sleep(1000L);
        }
        producer.flush();
        producer.close();
    }
}
```

Consumer

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;

import kafka.consumer.Consumer;
import kafka.consumer.ConsumerConfig;
```

```
import kafka.consumer.ConsumerIterator;
import kafka.consumer.KafkaStream;
import kafka.javaapi.consumer.ConsumerConnector;

public class MainKafkaConsumer {
    public static void main(String[] args) throws InterruptedException {
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("zookeeper.connect", "localhost:2181");
        props.put("group.id", "group1");

        ConsumerConnector consumer = Consumer.createJavaConsumerConnector(new Consu
erConfig(props));

        Map topicCountMap = new HashMap();

        String topic = "fw-blink-test";

        // 一次从主题中获取一个数据
        topicCountMap.put(topic, 1);

        Mapbyte[], byte[]>>>  messageStreams = consumer.createMessageStreams(topicCoun
Map);

        // 获取每次接收到的这个数据
        KafkaStream<byte[], byte[]> stream = messageStreams.get(topic).get(0);
        ConsumerIterator<byte[], byte[]> iterator =  stream.iterator();
        while(iterator.hasNext()){
            String message = new String(iterator.next().message());
            System.out.println("接收到: " + message);
        }
    }
}
```