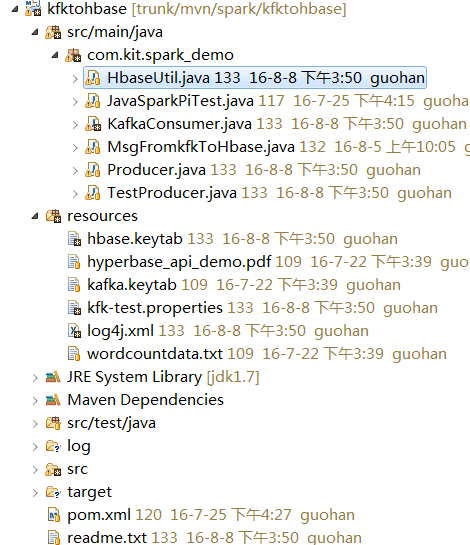
Kafka\_spark\_hbase示列

# 第一章项目概述

## 1.1 项目架构

使用kafka，discover，hyperbase开发示列。

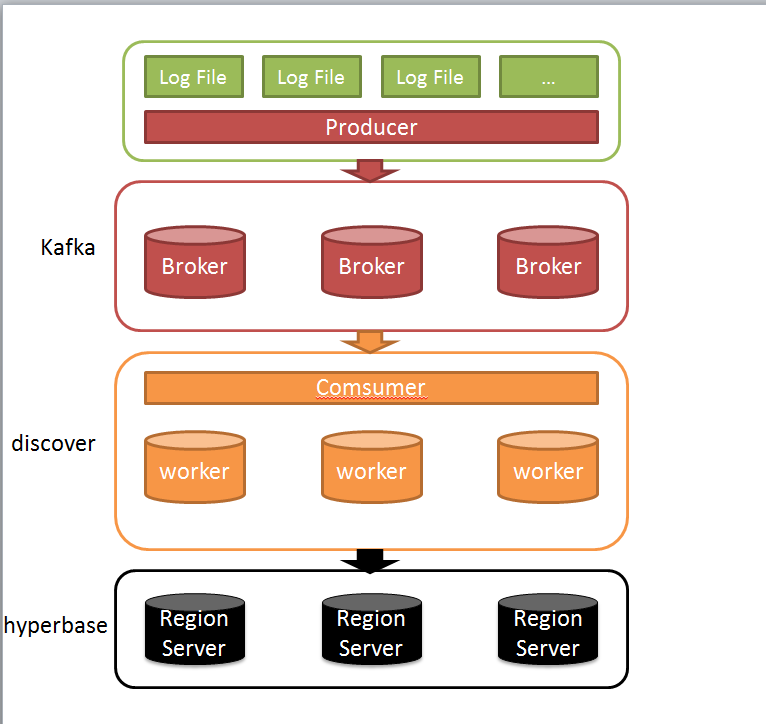
项目技术架构



Kafka作为日志事件的消息系统，具有分布式，可分区，可冗余的消息服务功能。

discover使用spark streaming功能，实时分析消息系统中的数据，完成计算分析工作。

hyperbase做为后端存储，存储spark消费结果，供其他系统进行调用



## 1.2 软件部署

### 1.2.1 相关组件 版本

Hadoop相关软件如zookeeper、kafka、hbase，使用的是星环 4.5 版本。

### 1.2.2 Kafka

Kafka 版本：kafka4.5

1. **安装部署**
   1. 部署kafka

**tar -xzf** kafka4.5**.tgz**

* 1. 编辑kafka 配置文件

**config/server-1.properties:**

**broker.id=0**

**port=9093**

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

**config/server-2.properties:**

**broker.id=1**

**port=9093**

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

**config/server-3.properties:**

**broker.id=2**

**port=9093**

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

* 1. 启动kafka

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

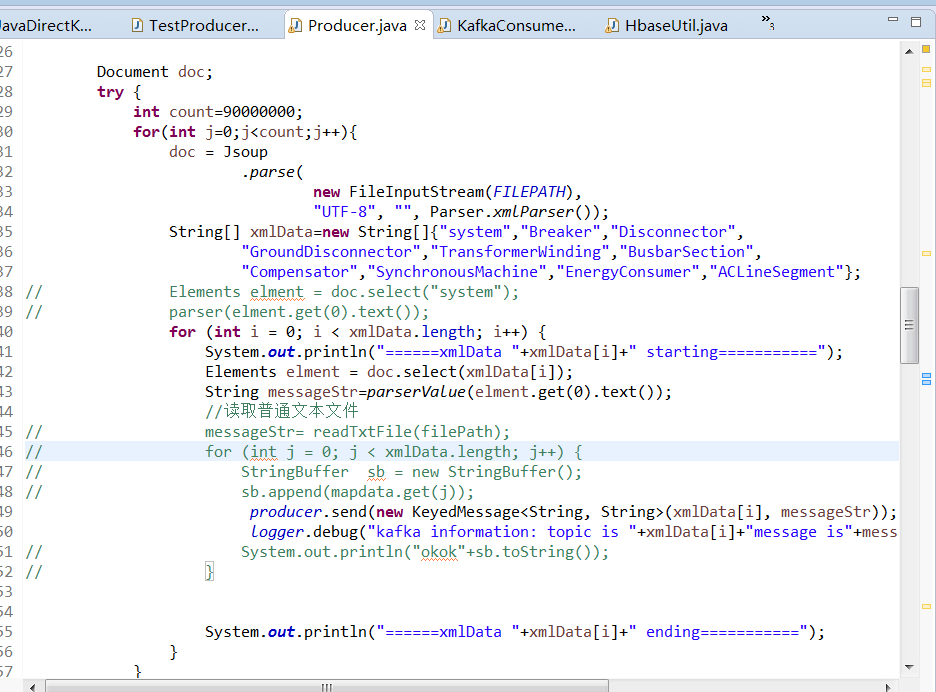
**bin/kafka-server-start.sh config/server-2.properties &**

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

1. **创建kafka topic**



解析非规则xml，根据xml节点去解析，不同节点作为topic主题



1. **查看是否创建成功**

添加认证方式后消息发送：

sh kafka-console-producer.sh --broker-list 172.16.19.152:9092 --topic test0524 --keytab /etc/kafka1/kafka.keytab --principal kafka/kit-b2@TDH

添加认证方式后消息消费 sh kafka-console-consumer.sh -zookeeper 172.16.19.152:2181 --topic test0524 --keytab /etc/kafka1/kafka.keytab --principal kafka/kit-b2@TDH

添加认证后 删除主题操作：

sh kafka-run-class.sh kafka.admin.DeleteTopicCommand --topic test05 --zookeeper 172.16.19.152:2181 --keytab /etc/kafka1/kafka.keytab --principal kafka/kit-b2@TDH --broker 172.16.19.152:9092

1. **注意事项**

在开发程序的时候，producer客户端必须要配置上broker的host映射信息，即使你的程序中使用的都是ip地址。

# 第二章项目开发

## 2.0 用maven灵活构建框架



## 2.1 文件读取Kafka

解析非规则文件，向不同的topic生产消息



## 2.2 Spark\_Streaming 处理数据



String brokers = "kit-b1:9092,kit-b2:9092,kit-b3:9092";// zk集群配置

**final** String topics = "test0729,system,Disconnector,Breaker";// 需要处理的对应主题

InputStream is = MsgFromkfkToHbase.**class**.getClassLoader().getResourceAsStream("kfk-test.properties");

Properties p = **new** Properties();

// 加载资源

p.load(is);

// kafka认证鉴权

AuthenticationManager.*setAuthMethod*("kerberos");

AuthenticationManager.*login*("kafka@TDH",

// TestProducer.class.getClassLoader().getResource("kafka.keytab").getPath()

p.getProperty("kafka\_key\_path").trim());

// spark上下文配置

SparkConf sparkConf = **new** SparkConf().setAppName("MsgFromkfkToHbase").setMaster("local");

JavaStreamingContext jssc = **new** JavaStreamingContext(sparkConf, Durations.*seconds*(2));

*logger*.debug("加载上下文成功！");

HashSet<String> topicsSet = **new** HashSet<String>(Arrays.*asList*(topics.split(",")));

HashMap<String, String> kafkaParams = **new** HashMap<String, String>();

kafkaParams.put("metadata.broker.list", brokers);

// JavaStreamingContext.jarOfClass(arg0)

// Create direct kafka stream with brokers and topics

JavaPairInputDStream<String, String> messages = KafkaUtils.*createDirectStream*(jssc, String.**class**, String.**class**,

StringDecoder.**class**, StringDecoder.**class**, kafkaParams, topicsSet);

*logger*.debug("加载kafka配置成功！");

/\*\*

\* 使用以下方法之一触发： print() foreachRDD() saveAsObjectFiles()

\* saveAsTextFiles() saveAsHadoopFiles()

\*

\* 这里采用foreacherdd 遍历kafka数据

\*/

messages.foreachRDD(**new** Function2<JavaPairRDD<String, String>, Time, Void>()

{

@Override

**public** Void call(JavaPairRDD<String, String> rdd, Time time) **throws** Exception

{

rdd.foreach(**new** VoidFunction<Tuple2<String, String>>()

{

@Override

**public** **void** call(Tuple2<String, String> tuple) **throws** Exception

{

String[] kafka = ***SPACE***.split(tuple.\_2());

*logger*.debug("################### insert hbase starting......." + tuple.\_2());

*pareDataToHabse*(kafka);

*logger*.debug("################### insert hbase ending.......");

}

});

**return** **null**;

}

});

// wordCounts.print();

// Start the computation

jssc.start();

jssc.awaitTermination();

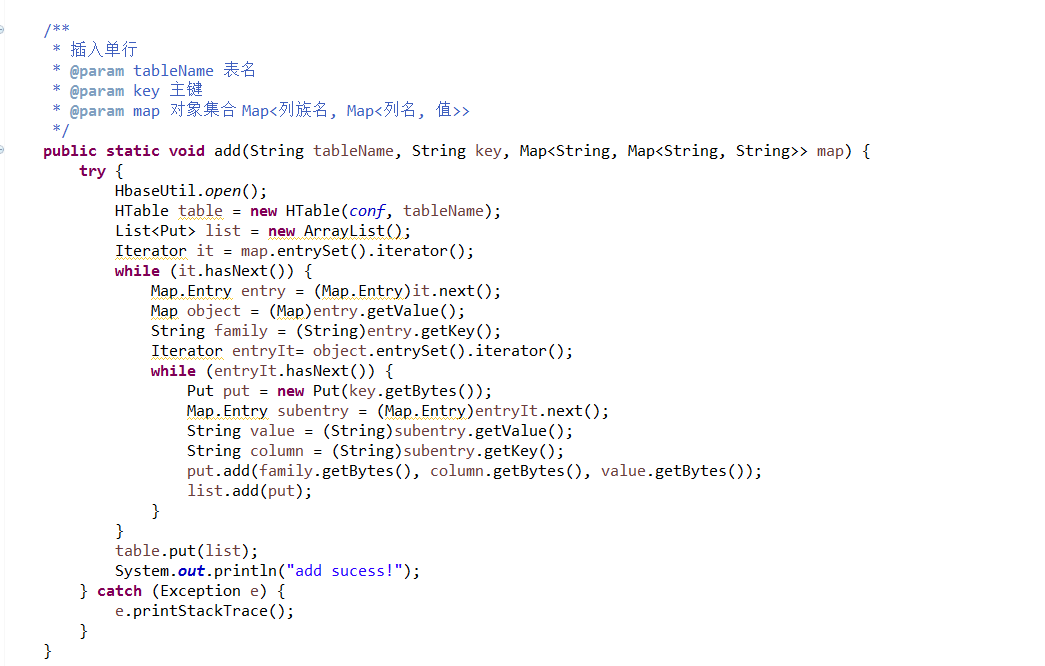
}

## 2.3 HBase 保存数据

跑hbase代码，灵活创建表



灵活创建列族以及保存数据：



## 2.4服务器端部署

kfkspkhbase-0.0.1-SNAPSHOT

提交任务：

切换到/usr/lib/kafka/

读取dt文件：

java -cp kfkspkhbase-0.0.1-SNAPSHOT.jar com.kit.spark\_demo.Producer

生产测试消息：

java -cp kfkspkhbase-0.0.1-SNAPSHOT.jar com.kit.spark\_demo.TestProducer

查看hbase数据库表：

java -cp kfkspkhbase-0.0.1-SNAPSHOT.jar com.kit.spark\_demo.HbaseUtil

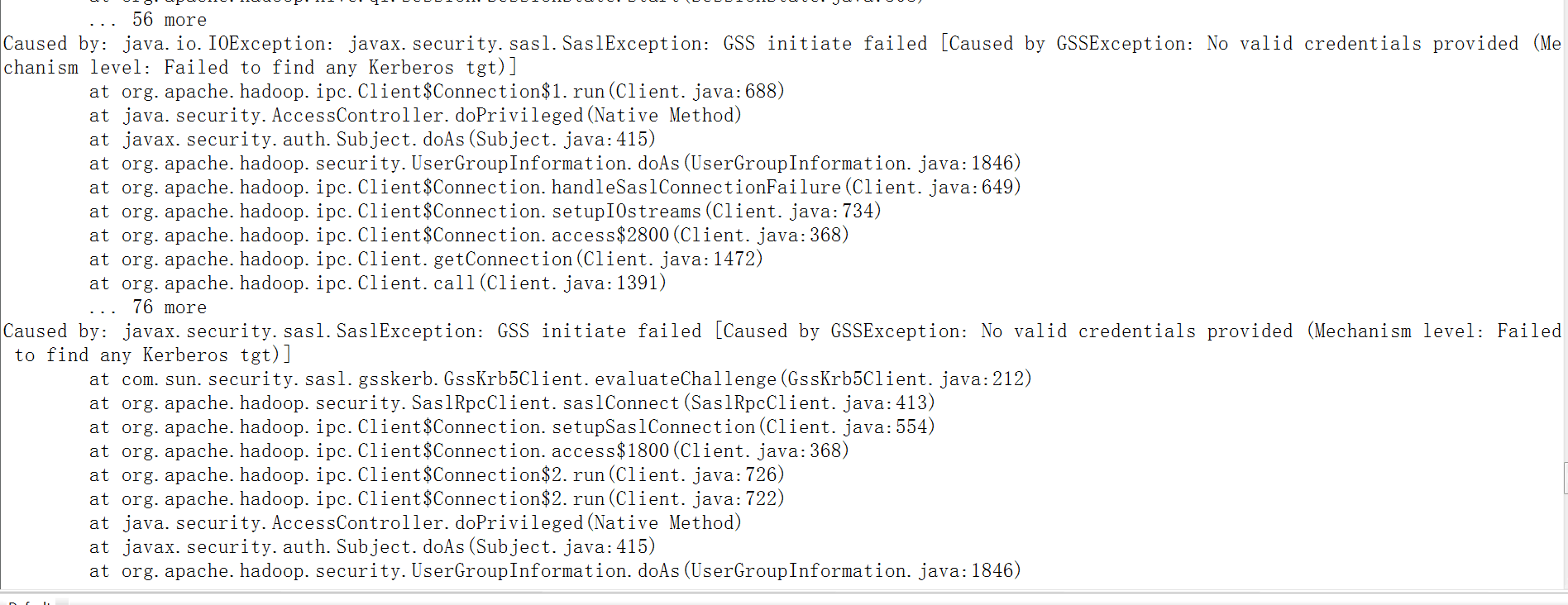
consumer消费：

java -cp kfkspkhbase-0.0.1-SNAPSHOT.jar com.kit.spark\_demo.KafkaConsumer

切换到: /usr/lib/discover/lib/

spark streaming 消费:

spark-submit --class com.kit.spark\_demo.MsgFromkfkToHbase --executor-memory 20G --total-executor-cores 100 /usr/lib/discover/lib/kfkspkhbase-0.0.1-SNAPSHOT.jar 100



Discover未开启认证