自定义流的开发：

Spark-shell中测试成功，使用如下：

1.先引入包：

import org.apache.spark.\_

import org.apache.spark.streaming.\_

import org.apache.spark.streaming.StreamingContext.\_ // not necessary since Spark 1.3

2.idea中自定义一个流Receiver(要实现Receiver特质)

如下：具体见spark：<http://spark.apache.org/docs/2.1.0/streaming-custom-receivers.html>

我的代码里多了几个包，是java的读文件，需要加入的：

import java.io.{BufferedReader, InputStreamReader}

import java.net.Socket

import java.nio.charset.StandardCharsets

import org.apache.spark.storage.StorageLevel

import org.apache.spark.streaming.receiver.Receiver

//import scala.tools.nsc.io.Socket

/\*\*

\* Created by Administrator on 2018/5/28 0028.

\*/

class CustomReceiver(host:String,port:Int)

extends Receiver[String](StorageLevel.MEMORY\_AND\_DISK\_2){

override def onStart(): Unit = {

new Thread("socket receiver"){

override def run(): Unit = {receive()}

}.start()

}

override def onStop(): Unit = {}

private def receive(): Unit ={

var socket:Socket = null

var userInput:String = null

socket = new Socket(host,port)

val reader = new BufferedReader(new InputStreamReader(socket.getInputStream(),StandardCharsets.UTF\_8))

userInput = reader.readLine()

while (!isStopped() && userInput !=null){

store(userInput)

userInput = reader.readLine()

}

reader.close()

socket.close()

restart("Trying to connect again")

}

}

第3步：用已建好的streamingContext定义流，即：

val customReceiverStream = ssc.receiverStream(new CustomReceiver("localhost", 9999))

这个ssc是之前利用已有的sc建好的

receiverStream接收一个实例对象，即自定义的CustomReceiver类的实例

然后是下面对流的处理过程：

val words = customReceiverStream.flatMap(\_.split(" "))

val pairs = words.map(word => (word, 1))

val wordCounts = pairs.reduceByKey(\_ + \_)

// Print the first ten elements of each RDD generated in this DStream to the console

wordCounts.print()

注意：处理过程，最后必须有“行动操作”，不然start不了的

第4步：

ssc.start()

ssc.awaitTermination()

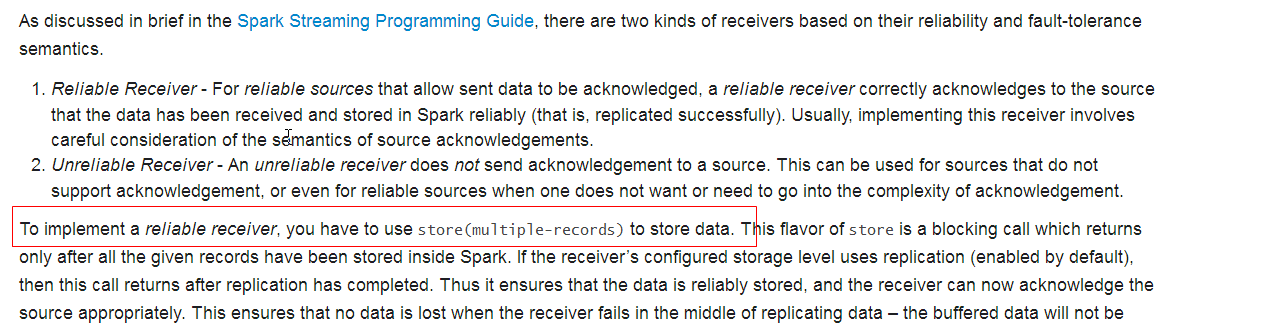
注意：官网的教程有误：

想想lines在前面的quick example中出现过。这里自定义的Receiver产生的流，却没有被用于计算、转换。这肯定不对。

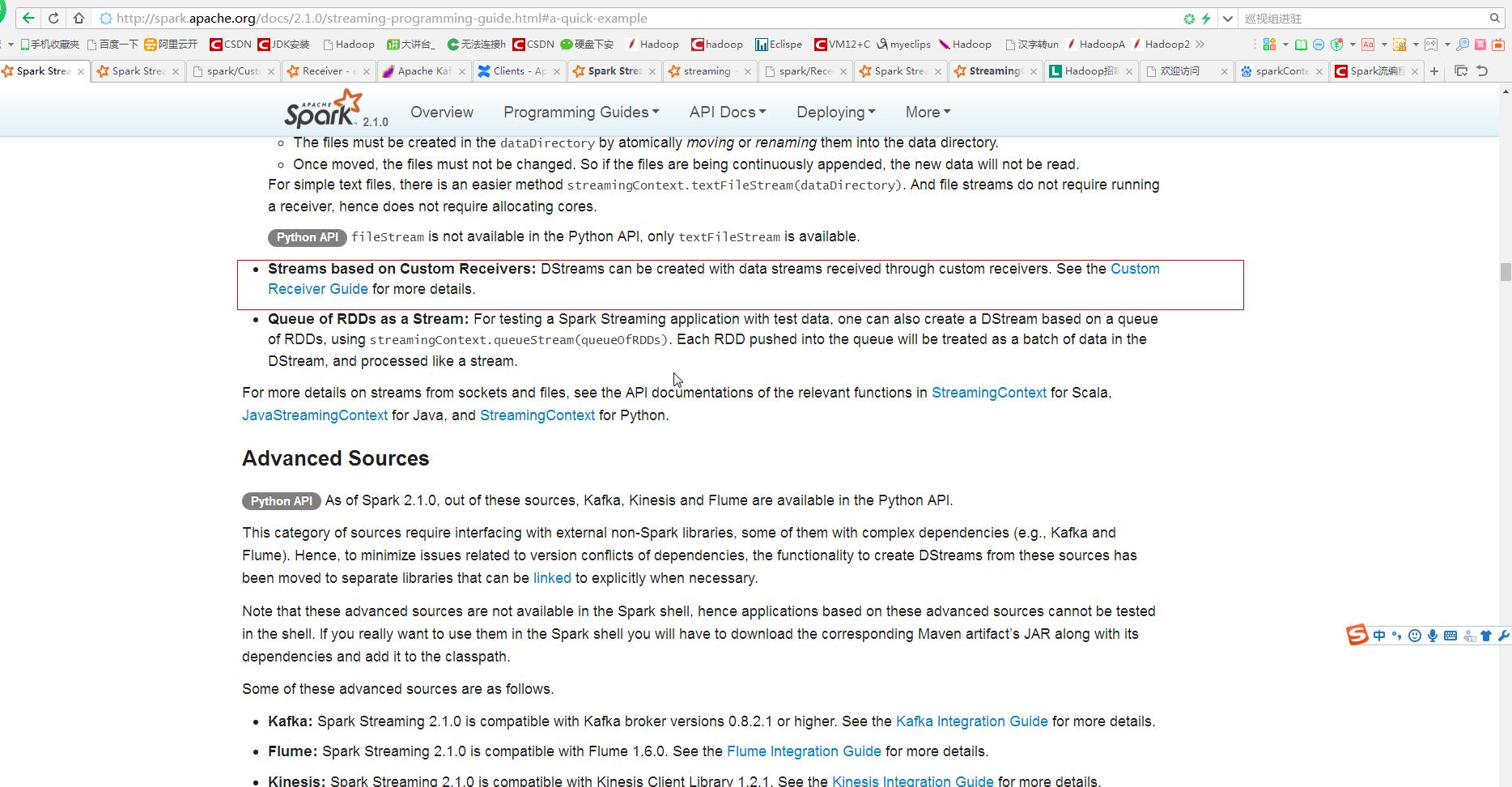
所以，这里的流不是lines



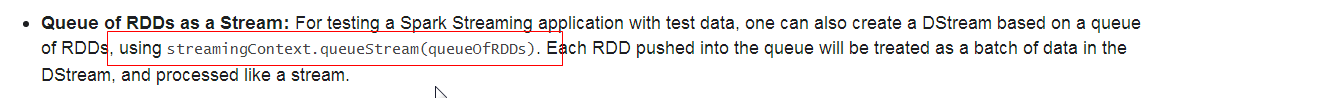
另外，流有可靠不可靠，用spark的内存store，就能保证其是可靠的：



在spark streaming的文档中，这一部分已经亲测有效：



除了自定义Receiver外，一个队列的RDD，也可以：



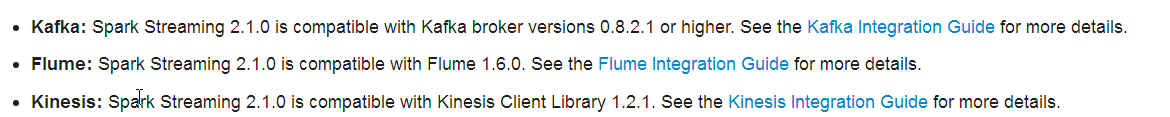
StreamingContext的api中有几种创建spark streaming的方式：

<http://spark.apache.org/docs/2.1.0/api/scala/index.html#org.apache.spark.streaming.StreamingContext>

基本如下：

文件流，queue[RDD],socket流，自定义Receiver流，文本流，二进制流（hadoop系统的支持兼容的二进制文件）

另外还有3种著名的流（Advanced Source）



接下来就是spark streaming的其他操作了：dataframe,rdd,mllib

仔细看：

Mllib有两种支持流的类型：

1. 实时训练+实时预测
2. （更大一部分是）离线训练+实时预测

