Integrating Apache Kafka and Spark Streaming

First you need to install Kafka in CDH

**For CDH 5.5.0, follow below steps (For CDH 5.8 and onwards, follow different steps):-**

**Step 1:** Download [Kafka](https://www.apache.org/dyn/closer.cgi?path=/kafka/0.10.2.0/kafka_2.11-0.10.2.0.tgz) from the website:-

https://www.apache.org/dyn/closer.cgi?path=/kafka/0.10.2.0/kafka\_2.11-0.10.2.0.tgz

**Step 2:** Unzip the Kafka file downloaded.

**[cloudera@quickstart ~]$** tar -xzf kafka\_2.11-0.10.2.0.tgz

**[cloudera@quickstart ~]$** cd kafka\_2.11-0.10.2.0

**Step 3:** Start zookeeper if it is not already started

**[cloudera@quickstart ~]$** bin/zookeeper-server-start.sh config/zookeeper.properties

**Step 4:** Start Kafka server

**[cloudera@quickstart ~]$** bin/kafka-server-start.sh config/server.properties

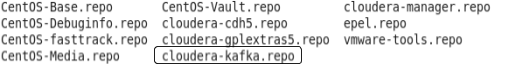
**For CDH 5.8, follow below steps:-**

**Step 1:** Kafka service as below from */etc/init.d*, if they are Installed or

              not

Go to path */etc/yum.repos.d*

**[cloudera@quickstart ~]$** /etc/yum.repos.d



Install the kafka

**[cloudera@quickstart yum.repos.d]$** sudo yum install kafka-server

Check Service Status

**[cloudera@quickstart ~]$** sudo service kafka-server status

If Service Status is Stop then Start Service

**[cloudera@quickstart ~]$** sudo service kafka-server start

**Step 2**: Check service status of zookeeper-server if it is started or

             not.

Check Service Status

**[cloudera@quickstart ~]$** sudo service zookeeper-server status

If Service Status is Stop then Start Service

**[cloudera@quickstart ~]$** sudo service zookeeper-server start

**Go to Kafka bin directory -**

***/usr/lib/kafka/***

**[cloudera@quickstart ~]$** cd /usr/lib/kafka

**Step 1: Creating topic - spark-topic**

we create a simple topic named “spark-topic” with just one partition:

**[cloudera@quickstart kafka]$** ./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic spark-topic

You may want to check out the available topics.

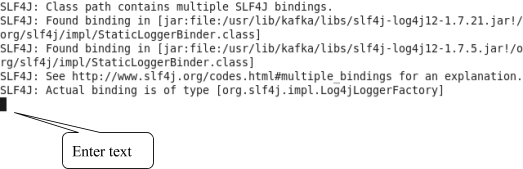
**[cloudera@quickstart kafka] $ .**/bin/kafka-topics.sh --list --zookeeper localhost:2181

**Step 2: Send messages**

Kafka comes with a command line client that will take input from a file or from standard input and send it out as messages to the Kafka cluster. By default, each line will be sent as a separate message.

Run the producer and then type a few messages into the console to send to the server.

**[cloudera@quickstart kafka] $** ./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic spark-topic



This is a message

This is another message

**Step 3: Open another Terminal and Open spark-shell**

**[cloudera@quickstart kafka]$** spark-shell --master yarn --executor-memory 1g

https://lh6.googleusercontent.com/0PId2i3nibOQbzepCIfsfjYY9COmdiJYX4n5V8QU8LuprxgM5b_Ab-YEsnUOAtF2D1Ph0V0sOnAitLj6DOs1VFrH9hILuCrfXiHi6dB2GPNdYsPxtYBRL_yRzrOZJnGaBS5hEnEV

Start Typing code in spark-shell

We do need the Kafka message serializes as well as the KafkaUtils of the Kafka streaming package for Spark.

scala> import org.apache.spark.SparkConf

scala> import org.apache.spark.streaming.kafka.KafkaUtils  
scala> import org.apache.spark.streaming.StreamingContext  
scala> import org.apache.spark.streaming.Seconds

scala> val conf = new SparkConf().setMaster("local[\*]").setAppName("KafkaReceiver")

Our SparkStreamingContext will only require ERROR logging to prevent INFO logging output. The context will run in a 5 second interval:

scala> sc.setLogLevel("ERROR")  
scala> val ssc = new StreamingContext(sc, Seconds(10))  
scala> val kafkaStream = KafkaUtils.createStream(ssc, "localhost:2181","spark-streaming-consumer-group", Map("spark-topic" -> 5))  
scala> kafkaStream.print()  
scala> ssc.start

Spark Streaming is now connected to Apache Kafka and consumes messages every 10 seconds. Leave it running and switch to another terminal.

Open a terminal to run a Kafka producer

[cloudera@quickstart kafka]$ ./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic spark-topic

Enter Text -

Switch to the terminal with Spark Streaming running and see the message printed out

