|  |
| --- |
| Module 9: Understanding Apache Kafka and Apache Flume |
| Case Study II Steps |

|  |
| --- |
|  |



© Brain4ce Education Solutions Pvt. Ltd.

Case Study: Telecom Pipeline

Domain: Telecom

There are two large obstacles in collecting metadata from a network as large as India’s Big Telecom operator: transporting the sheer volume of data and processing it before the data no longer accurately reflects the state of the network.

Fortunately, combining Apache Flume and Apache Kafka using the Kafka pattern provides a means to move data into the Hadoop cluster and readily scale the pipeline to address both transient and persistent spikes in data volume. Company is planning to deploy Flume and Kafka across the network in a geographically distributed architecture that achieves scale and resilience, having been tuned from around 10,000 events per second on initial deployment to 1,000,000 events per second using a three-node Kafka cluster.

**Tasks:**

You are part of the Telecom Operator’s R&D team, which is required to perform a quick POC on the Kafka Flume pipeline to persist data to HDFS and analyze the data through spark streaming.

**Dataset:**

The data set consists of 100 variables and approx. 100 thousand records containing different variables explaining the attributes of telecom industry and various factors considered important while dealing with customers of telecom industry.

Steps:

### **Step 1:** Create a topic in Kafka so that consumers and produces can enqueue/dequeue data respectively from the topic

***kafka-topics --create --zookeeper ip-20-0-21-161.ec2.internal:2181 --replication-factor 1 --partitions 1 —topic paysim***

### **Step 2:** Write the test Kafka consumer and verify that data is sent successfully

***kafka-console-consumer --zookeeper ip-20-0-21-161.ec2.internal:2181 --topic paysim —from-beginning***

***kafka-console-producer --broker-list ip-20-0-31-4.ec2.internal:9092 --topic paysim***

### **Step 3:** Configure a flume agent to use Kafka as the channel and HDFS as the sink

Create new file flafka.conf

***wh.sources = ws***

***wh.channels = mem***

***wh.sinks = hd***

***wh.sources.ws.type = org.apache.flume.source.kafka.KafkaSource***

***wh.sources.ws.zookeeperConnect = ip-20-0-21-161.ec2.internal:2181***

***wh.sources.ws.topic =paysim***

***wh.sources.ws.batchSize = 100***

***# Each sink's type must be defined***

***wh.sinks.hd.type = hdfs***

***wh.sinks.hd.hdfs.writeFormat = Text***

***wh.sinks.hd.hdfs.fileType = DataStream***

***wh.sinks.hd.hdfs.filePrefix = flumedemo***

***wh.sinks.hd.hdfs.useLocalTimeStamp = true***

***wh.sinks.hd.hdfs.path = tmp/kafka/%{topic}/***

***wh.sinks.hd.hdfs.rollCount=100***

***wh.sinks.hd.hdfs.rollSize=0***

***# Each channel's type is defined.***

***wh.channels.mem.type = memory***

***wh.channels.mem.capacity = 1000***

***wh.channels.mem.transactionCapacity = 100***

***# Bind source and sink to channel***

***wh.sinks.hd.channel = mem***

***wh.sources.ws.channels = mem***

***wh.channels.mem.capacity = 100***

### **Step 4:** Start flume agent and test the output to HDFS

***flume-ng agent -n wh -c conf -f flafka.conf - Dflume.root.logger=INFO,console***

### **Step 5:** Test the complete pipeline

Check messages in HDFS

***hdfs dfs -ls tmp/kafka/paysim***