Module 9 – Understanding Apache Kafka and Apache Flume

Case study :

Case Study: Spam Detection Domain: Telecom

A telecom software provider is building an application to monitor different telecom components in the production environment. For monitoring purpose, the application relies on log files by parsing the log files and looking for potential warning or exceptions in the logs and reporting them. The POC we had been working on, for SPAM Detection on the data of telecom operator forum, has been accepted and the stakeholders have asked us to work on the real-time example for predicting SPAM messages.

Tasks

This POC will focus on saved machine learning model for spam prediction with streaming data to do real-time prediction. You have already developed the model in previous exercise.

Now as part of a POC you are required to publish data through Kafka API, before pushing the data to Spark Streaming. In the later part the streaming application will be subscribed to the Kafka topic.

1. Verify the cluster

2. Create a topic in Kafka so that consumers and produces can enqueue/dequeue data respectively from the topic

3. Write the test Kafka consumer and verify that data is sent successfully.

4. Configure a flume agent to use Kafka as the channel and HDFS as the sink

5. Start flume agent and test the output to HDFS

6. Test the complete pipeline

Solution : The task which we have to perform as a part of the POC can be divided into following steps :

1. Creation of a topic in Kafka so that consumers and produces can enqueue/dequeue data respectively from the topic

Note the Kafka Cluster configuration we are using is :

zookeeper = ip-20-0-21-161.ec2.internal:2181 *brokers = ip-20-0-31-210.ec2.internal:9092, ip-20-0-31-221.ec2.internal:9092, ip-20-0-31-4.ec2.internal:9092*

Command executed in the console:

1. Creating the topic

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

2. Describing the fkchannel

```
kafka-topics --describe --zookeeper ip-20-0-21-161.ec2.internal:2181 --topic fkchannel
```

2. Setting up of a Flume-Agent using the Kafka Topic as a channel

Flume_kafka.conf used is as follows:

```
#################################################################################
fkagent.sources = http-source

fkagent.sinks = hd

fkagent.channels = ch3


# Define / Configure Source (multiport seems to support newer "stuff")

################################

fkagent.sources.http-source.type = org.apache.flume.source.http.HTTPSource

fkagent.sources.http-source.channels = ch3

fkagent.sources.http-source.port = 9000

fkagent.sources.http-source.bind = localhost
```

```
# HDFS Sink

#################################

fkagent.sinks.hd.type = hdfs

fkagent.sinks.hd.hdfs.fileType = DataStream

fkagent.sinks.hd.channel = ch3

fkagent.sinks.hd.hdfs.path = /user/edureka_960126/flume_kafka

fkagent.sinks.hd.hdfs.filePrefix = event-

fkagent.sinks.hd.hdfs.batchSize = 3

# Channels

#################################

fkagent.channels.ch3.type = org.apache.flume.channel.kafka.KafkaChannel

fkagent.channels.ch3.kafka.topic = fkchannel

fkagent.channels.ch3.kafka.consumer.group.id = flume-consumer

fkagent.channels.ch3.kafka.bootstrap.servers = ip-20-0-31-210.ec2.internal:9092, ip-20-0-31-221.ec2.internal:9092, ip-20-0-31-4.ec2.internal:9092


##################################################################################
```

Hence the task to Configure a flume agent to use Kafka as the channel and HDFS as the sink was performed.

After setting up of the Flume Agent , the following commands were executed:

a. Starting the flume agent with above configuration :

```
./flume-ng agent --conf conf --conf-file /mnt/home/edureka_960126/flume_kafka.conf --name fkagent -Dflume.root.logger=INFO,console
```

b. Streaming Post HTTP requests

```
curl -X POST -H 'Content-Type: application/json; charset=UTF-8' -d
'[{"headers" : {"a":"b"},"body": "jaimatadi,jaimatadi,jaimatadi"}] '
http://localhost:9000
```

c. Verifying the result in HDFS destination

```
[edureka_960126@ip-20-0-41-164 ~]$ hdfs dfs -cat /user/edureka_960126/flume_kafka/event-
.1595224727528
```

```
jaimatadi,jaimatadi,jaimatadi
```

Conclusion :

The whole Pipeline of :

Streaming HTTP Source → Flume Agent[Using the Kafka Cluster-Topic as a Channel] → HDFS Storage as Sink

was configured and tested.

Learning :

The Kafka channel can be used for multiple scenarios:

- With Flume source and sink - it provides a reliable and highly available channel for events
- With Flume source and interceptor but no sink - it allows writing Flume events into a Kafka topic, for use by other apps
- With Flume sink, but no source - it is a low-latency, fault tolerant way to send events from Kafka to Flume sinks such as HDFS, HBase etc.

Here we have implemented Case 1 above , as Kafka Provides reliable and highly available channel for events.

Note that we have created a topic with one partition we can increase partition also and we can increase the replication factor for increasing the availability of Kafka Channel.

Flume Configuration has been attached with this submission.