INTRODUCTION TO APACHE FLUME

Flume 101

Apache Flume reads a data source and writes it to storage at incredibly high volumes and without losing any events. These data feeds include streaming logs, network traffic, Twitter feeds, etc. (examples below)

But it does not do data manipulation. Instead you can write it to something with the ability to do that, like Hive and, in that case, use Hive's SQL-like commands (HQL) to process it. Or you can output it to ElasticSearch, where the data gets stored in JSON format which can be queried with Kibana. There are many other options for where to store the data.

(This article is part of our Hadoop Guide. Use the right-hand menu to navigate.)

Installation

The installation of Flume is simple. You just download it and unzip it. There is no classpath or anything like that to set. You do not start Flume, per se. Instead you start agents, one for every data source that you want to read and save.

Flume source and sinks

There is not much terminology to master:

A **source** is input data. A **sink** is output. And a **memory channel** stores events in memory, with the proviso that those get lost if the agent dies. Some of the sinks, sources, and memory channels that Flume supports include:

Sources—Avro (data serialization with a schema), Thrift (abstraction written by Facebook to facilitate communications between different programming languages), Exec (shell commands), JMS (messaging), Twitter, Kafka (LinkedIn's streaming product), NetCat, Syslog, HTTP, custom (you can write your own, plus there are plenty of third party ones.)

Sinks—HDFS, Hive, Logger (Apache log4j), ElasticSearch, Kafka, Custom (third party plugins like Cassandra, HBase, and MongoDB)

Memory Channels—JDBC (meaning databases), Kafka, File, Custom

Configuration

The configuration of a Flume agent has the format shown below. There are sources (input), sinks (output), and channels (connects sources to sinks). The agent name is whatever name you make up.

The color coding shows how you declare the names of sources, sinks, and channels and then associate those with their corresponding options.

```
(agent name).sources = (source name)
(agent name).channels = (channel name)
(agent name).sinks = (sink name)

(agent name).sources.(source name).options = value
(agent name).sinks.(sink name).options = value

(agent name).channels.(channel name).options = value

(agent name).sinks.(sink name).channel =   (channel name)
(agent name).sources.(sink name).channel =   (channel name)
```

To start an agent you go to the bin folder and execute:

```
./flume-ng agent -n (agent name) -c conf -f ../conf/(agent config file name)
```

Tail a file

Here is an example of how to read syslog on a CentOS system. In actual practice you would most likely use syslog and then configure Flume to listen for TCP or UDP messages. But here we just use tail.

We want to store this data in Hadoop so we make a directory:

```
hadoop fs -mkdir /flume
```

Here is our configuration file. Notice the types **exec** and **hdfs**. Each type has different options. Like the exec type requires a **command** to execute. And the hdfs type requires the **path**.

```
cat ../conf/flume.conf
```

```
# give names to the agent, source, sinks, and channels
agent.sources=tail
agent.sinks=hdfs
agent.channels=ch1

# tell it what command to run
agent.sources.tail.type=exec
agent.sources.tail.command=tail -F /var/log/messages

# give Hadoop options
agent.sinks.hdfs.type=hdfs
agent.sinks.hdfs.hdfs.path=/flume
agent.sinks.hdfs.hdfs.filePrefix=syslog
agent.sinks.hdfs.hdfs.rollInterval=10
agent.sinks.hdfs.hdfs.rollSize=0

# give channel options
agent.channels.ch1.type=memory
agent.channels.ch1.capacity=1000

# connect the sources and sinks to the channel
agent.sources.tail.channels=ch1
agent.sinks.hdfs.channel=ch1
```

Start the flume agent. The name of the agent must match the agent name you put in flume.conf

```
./flume-ng agent -n (agent  name) -c conf -f ../conf/(config file)
```

Then stdout will write messages showing that data has been written:

```
17/03/08 16:42:44 INFO hdfs.HDFSEventSink: Closing /flume/syslog
17/03/08 16:42:44 INFO hdfs.BucketWriter: Closing
/flume/syslog.1488987752411.tmp
17/03/08 16:42:44 INFO hdfs.BucketWriter: Renaming
/flume/syslog.1488987752411.tmp to /flume/syslog.1488987752411
```

We can look at what it wrote using cat.It will not always be legible text.

```
hadoop fs -cat /flume/syslog.1488987752411
```

If you have any mistake in the configuration file Flume will give you a Java Stack trace with a fairly friendly error message telling you what is wrong.

Twitter

Here we write Twitter tweets, using Twitter's streaming API, and then save those to Hadoop. To do this you first have to create an app at https://apps.twitter.com/, which will give you the keys you need to connect.

The Tweets have no structure in Flume. So if you want to work with them you would write them to,

for example, Hive. But to do that you have to create the Twitter schema in Hive. You can look for instructions for how to do that on the internet. In this example we just write them to Hadoop.

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = t1
TwitterAgent.sinks = hdfs

TwitterAgent.sources.Twitter.type =
org.apache.flume.source.twitter.TwitterSource
TwitterAgent.sources.Twitter.consumerKey = (obfuscated)
TwitterAgent.sources.Twitter.consumerSecret =  (obfuscated)
TwitterAgent.sources.Twitter.accessToken =  (obfuscated)
TwitterAgent.sources.Twitter.accessTokenSecret =  (obfuscated)
TwitterAgent.sources.Twitter.keywords = health
TwitterAgent.sinks.console.type = logger

TwitterAgent.sinks.hdfs.type=hdfs
TwitterAgent.sinks.hdfs.hdfs.path=/flume/twitter
TwitterAgent.sinks.hdfs.hdfs.filePrefix=messages
TwitterAgent.sinks.hdfs.hdfs.rollInterval=10
TwitterAgent.sinks.hdfs.hdfs.rollSize=0

TwitterAgent.channels.t1.type=memory
TwitterAgent.channels.t1.capacity = 1000



TwitterAgent.sources.Twitter.channels = t1
TwitterAgent.sinks.hdfs.channel = t1
```

Then list the files written in Hadoop:

```
-rw-r--r--   1 root supergroup  217127 2017-03-08 14:54
/flume/twitter/messages.1488981289896
-rw-r--r--   1 root supergroup  232847 2017-03-08 14:55
/flume/twitter/messages.1488981289897
-rw-r--r--   1 root supergroup     227083 2017-03-08 14:55
/flume/twitter/messages.1488981289898
```

You can log Tweets to the console using:

```
TwitterAgent.sinks.console.type = logger
```

But as you can see they are in Hex, so hard to read:

```
7/03/08 14:44:39 INFO sink.LoggerSink: Event: { headers:{} body: 4F 62 6A 01
02 16 61
76 72 6F 2E 73 63 68 65 6D Obj...avro.schem }
17/03/08 14:44:41 INFO sink.LoggerSink: Event: { headers:{} body: 4F 62 6A 01
02 16 61
```

```
76 72 6F 2E 73 63 68 65 6D Obj...avro.schem }
```

That is a basic introduction to Flume. As you can see it is not a complicated product at all. As an exercise you might try now to write syslogs to Flume.