YCBS-257 - Data at Scale

Workshop 8

Hadoop Data Ingestion

General Instructions:

Hadoop Data ingestion is the beginning of your data pipeline in a data lake. It means taking data from various silo databases and files and putting it into Hadoop. Hadoop is an open source framework; there are a variety of ways you can ingest data into Hadoop. It gives every developer the choice of using her/his favorite tool or language to ingest data into Hadoop. In this workshop we will focus on two particular tools to ingest data into Hadoop, Apache Sqoop and Apache Flume.

Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data into HDFS. It has a simple and flexible architecture based on streaming data flows; and robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. Flume employs the familiar producer-consumer model. Source is the entity through which data enters into Flume. On the other hand, Sink is the entity that delivers the data to the destination. Flume has many built-in sources (e.g. log4j and syslogs) and sinks (e.g. HDFS and HBase). Channel is the conduit between the Source and the Sink. Sources ingest events into the channel and the sinks drain the channel. Channels allow decoupling of ingestion rate from drain rate. When data are generated faster than what the destination can handle, the channel size increases.

Start your Cloudera QuickStart VM to complete the workshop

Online resources:

https://flume.apache.org/

Apache Flume

The purpose of these exercises is to familiarize you with the basic configurations of a Flume agent. You will learn how to run a Flume agent in order to collect data from a particular source to a particular sink (destination).

Prerequisites:

To complete this exercise, you need to install the **telnet** Linux tool (on your Cloudera VM).

1. Open a new Terminal window
2. Switch to root user

```
$ su root      (password: cloudera)
```

3. Enter the following command to install telnet

```
$ yum install telnet      (Answer 'yes' when you are prompted)
```

4. When the installation is complete switch back to the cloudera user

```
$ su cloudera
```

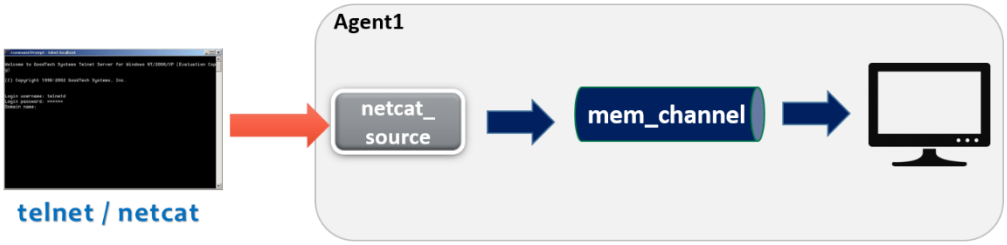
Exercise 1: Collecting data using Flume

In this exercise, you will simulate data that arrives over network on the tcp port **41415**. Flume will collect these data and store it into HDFS

Step1: You will configure your Flume agent to print out events on screen using the Logger sink. (This step is to ensure that your configuration file is valid).

Step2: You will remove the Logger sink and configure an HDFS sink instead.

Step 1:



1. Create a new local directory flume
2. Create a new text file **agtex1.conf** into this directory
3. Complete the file with these configuration settings

Flume agent configuration file settings		
Agent Name: **agtex1**		
Sources	Name	**source_ncat**
	type	**netcat**
	bind	**localhost**
	port	**41415**
	channels	**mem_channel**
Channels	Name	**mem_channel**
	type	**memory**
	capacity	**10000**
	transactionCapacity	**100**
Sinks	Name	**log_sink**
	type	**Logger**
	channel	**mem_channel**

Running the Flume Agent

4. Open a Terminal window and run the Flume agent as following:

```
$ flume-ng agent --conf  conf --conf-file
  /home/cloudera/flume/agtex1.conf --name agtex1
```

Generating data

5. Open a new Terminal window and run telnet using this command:

```
$ telnet localhost 41415
```

6. Type any data in the Telnet windows, you should see it on the flume agent window.

Note: Flume Logger Sink is dedicated for testing, any data displayed on screen is truncated to the first 16 bytes.

Step 2:

1. Create a new configuration file based on the agtex1.conf file
2. Remove the sink configuration
3. Add a HDFS Sink configuration as the following:

Sinks	**Name**	**hdfs-sink**
	type	**hdfs**
	hdfs.path	**/tmp/flume-hdfs/**
	hdfs.fileType	**DataStream**
	channel	**mem_channel**

4. Run the Flume agent
5. Open a new telnet session and enter some data
6. Check on HDFS the output directory.

Exercise 2: Using Spool Directory Flume Source

In this exercise, you will configure a Flume agent to scan the contents of a directory on your local file system named `spooldirsource`. Any file placed into this directory is transferred by the Flume agent to the HDFS directory `/tmp/flume-spooldir/.` Once the transfer is complete Flume will rename the input file and add the COMPLETED suffix to the file.

1. Create a new local directory `spooldirsource` into the `flume` directory
2. Create a new text file `agtex2.conf` into this directory
3. Complete the file with these configuration settings :

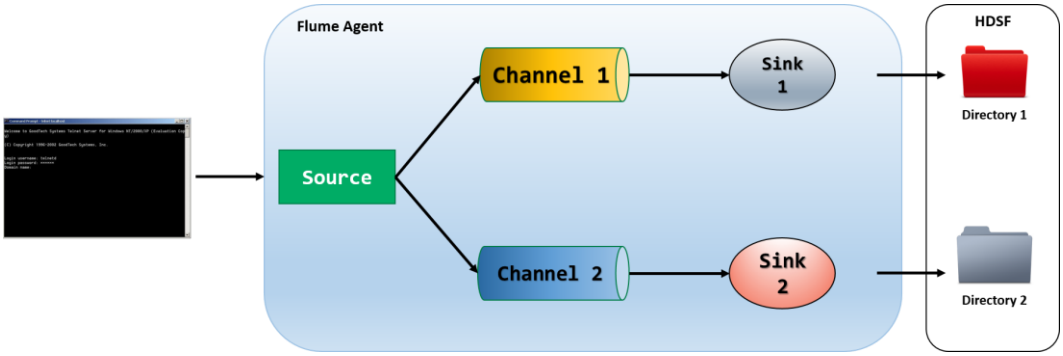
Flume agent configuration file settings		
Agent Name : **agtex2**		
Sources	Name	**spooldir-source**
	type	**spooldir**

	spoolDir	**/home/cloudera/flume/spooldirsource**
	fileHeader	**false**
	channels	**mem-channel**
Channels	Name	**mem-channel**
	type	**memory**
	capacity	**1000000**
	transactionCapacity	**10000**
Sinks	**Nom**	**hdfs-sink**
	type	**hdfs**
	hdfs.path	**/tmp/flume-spooldir/**
	hdfs.fileType	**DataStream**
	channel	**mem-channel**

4. Copy any file from Bixi stations/rides data files into `spooldirsource`
5. Check the target HDFS directory
6. Check the input file name into the `spooldirsource` directory.

Exercise 3: Routing Events using Flume RegEx

In this exercise you will configure a Flume agent in order to route captured data to a specific channel and then to a specific sink (HDFS for instance) using the `Regex Extractor` interceptor.



1. Create a new text file `agtex3.conf`
2. Complete the file with these configuration settings :

Flume agent configuration file settings	
Agent Name	**agtex3**

Sources	`Name = Src1`
	`type = netcat`
	`bind = localhost`
	`port = 45454`
	
	`channels = MemChannelHadoop MemChannelSpark`

Interceptors	`name = i1`
	`type = regex_extractor`
	`regex = (Hadoop|Spark)`
	`serializers = s1`
	`serializers.s1.name = BigData`

Selector	`type = multiplexing`
	`header = BigData`
	`mapping.Hadoop  = MemChannelHadoop`
	`mapping.Spark  = MemChannelSpark`

Channel 1, Channel 2	`name = MemChannelHadoop, MemChannelSpark`
	
	`MemChannelSpark.type = memory`
	`MemChannelSpark.capacity = 10000`
	`MemChannelSpark.transactionCapacity = 1000`
	
	`MemChannelHadoop.type = memory`
	`MemChannelHadoop.capacity = 10000`
	`MemChannelHadoop.transactionCapacity = 1000`

Sink 1, Sink 2	`name = HDFS_Hadoop`
	`type = hdfs`
	`hdfs.path = /tmp/flume-routing/Hadoop/`
	`hdfs.writeFormat =  Text`
	`hdfs.fileType =  DataStream`
	`hdfs.batchSize =  1000`
	`hdfs.rollSize =  0`
	`hdfs.rollCount =  10000`
	
	`channel = MemChannelHadoop`

	`name = HDFS_Spark`
	`type = hdfs`
	`hdfs.path = /tmp/flume-routing/Spark/`
	`hdfs.writeFormat =  Text`
	`hdfs.fileType =  DataStream`
	`hdfs.batchSize =  1000`
	`hdfs.rollSize =  0`
	`hdfs.rollCount =  10000`
	
	`channel = MemChannelSpark`