Apache Flume

BAS Academy

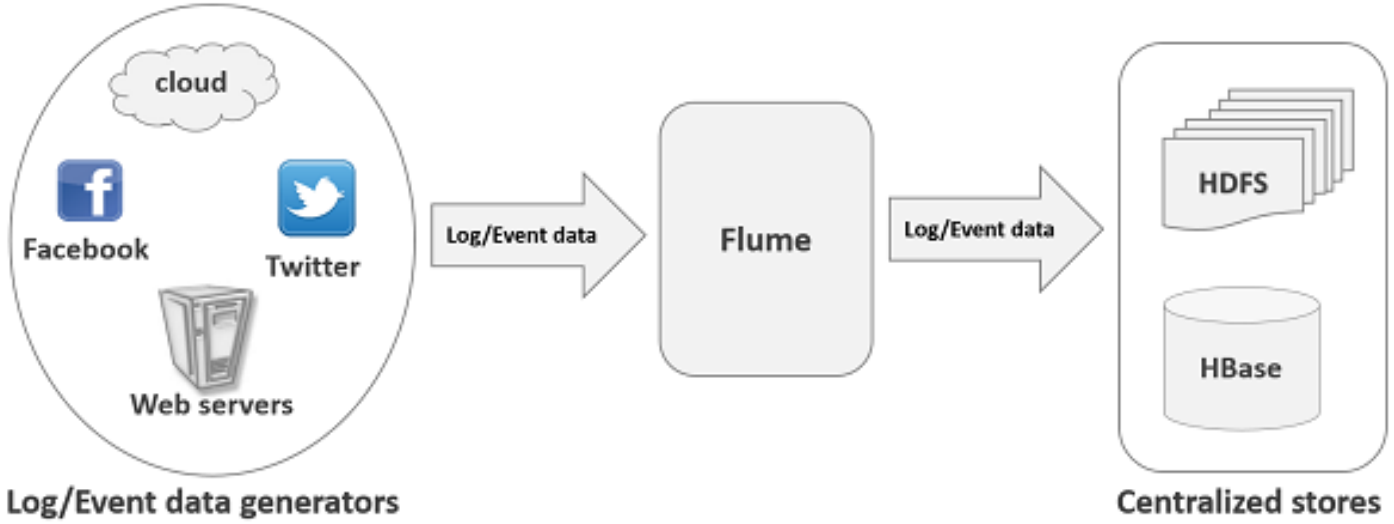
Agenda

- About Flume
- Flume Architecture
- Flume Configuration
- Agent Execution
- Hands On

About Flume

What is Flume

- Flume is an open-source Apache project
- It is a data ingestion tool for efficiently collecting, aggregating and moving large amounts of log data from many different sources into HDFS.
- Along with the log files, Flume is also used to import huge volumes of event data produced by social networking sites like Facebook and Twitter, and e-commerce websites like Amazon and Flipkart.
- Flume uses a producer-consumer model for handling events



HDFS PUT Vs Flume

HDFS PUT

- Using **put** command, we can transfer **only one file at a time**
- If we use **put** command, the data is needed to be packaged and should be ready for the upload. Since the webservers generate data continuously, it is a very difficult task.

Flume

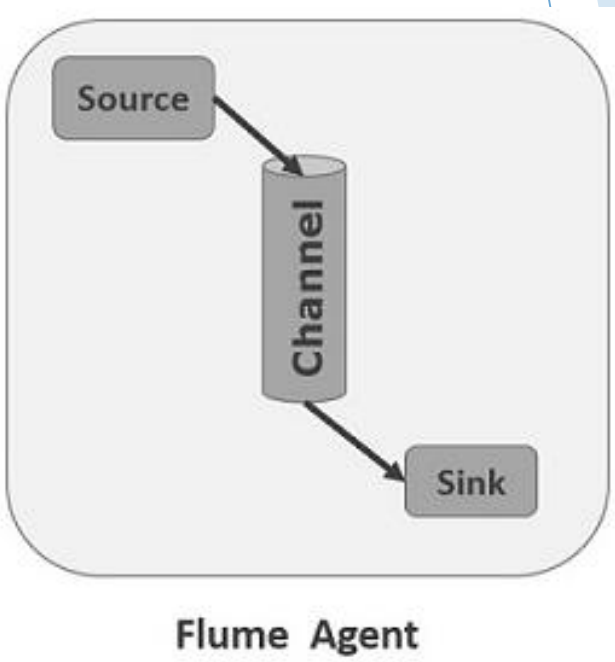
- Transfer the "streaming data" from data generators to centralized stores (especially HDFS) with less delay.

Flume Architecture

Agent

- Event: A singular unit of data that is transported by Flume (typically a single log entry)

- An agent is an independent daemon process (JVM) in Flume.
- It receives the data (events) from clients or other agents and forwards it to its next destination (sink or agent).
- Flume may have more than one agent.



Source

- Source is the producer
- It receives data from the data generators and transfers it to one or more channels in the form of Flume events.

Examples of a Source include

- System log files
- Network traffic log files
- Website traffic logs
- Twitter feeds and other social media sources

Channel

- A channel is a transient store which receives the events from the source and buffers them till they are consumed by sinks.
- It acts as a bridge between the sources and the sinks.
- A Channel drains into a Sink, but because it is asynchronous the Channel is not required to send events to the Sink at the same rate that it receives them from the Source.
- This allows for a Source to not have to wait for Flume to store the event in its final destination

Examples of a Source include

- JDBC channel
- File system channel
- Memory channel

Sink

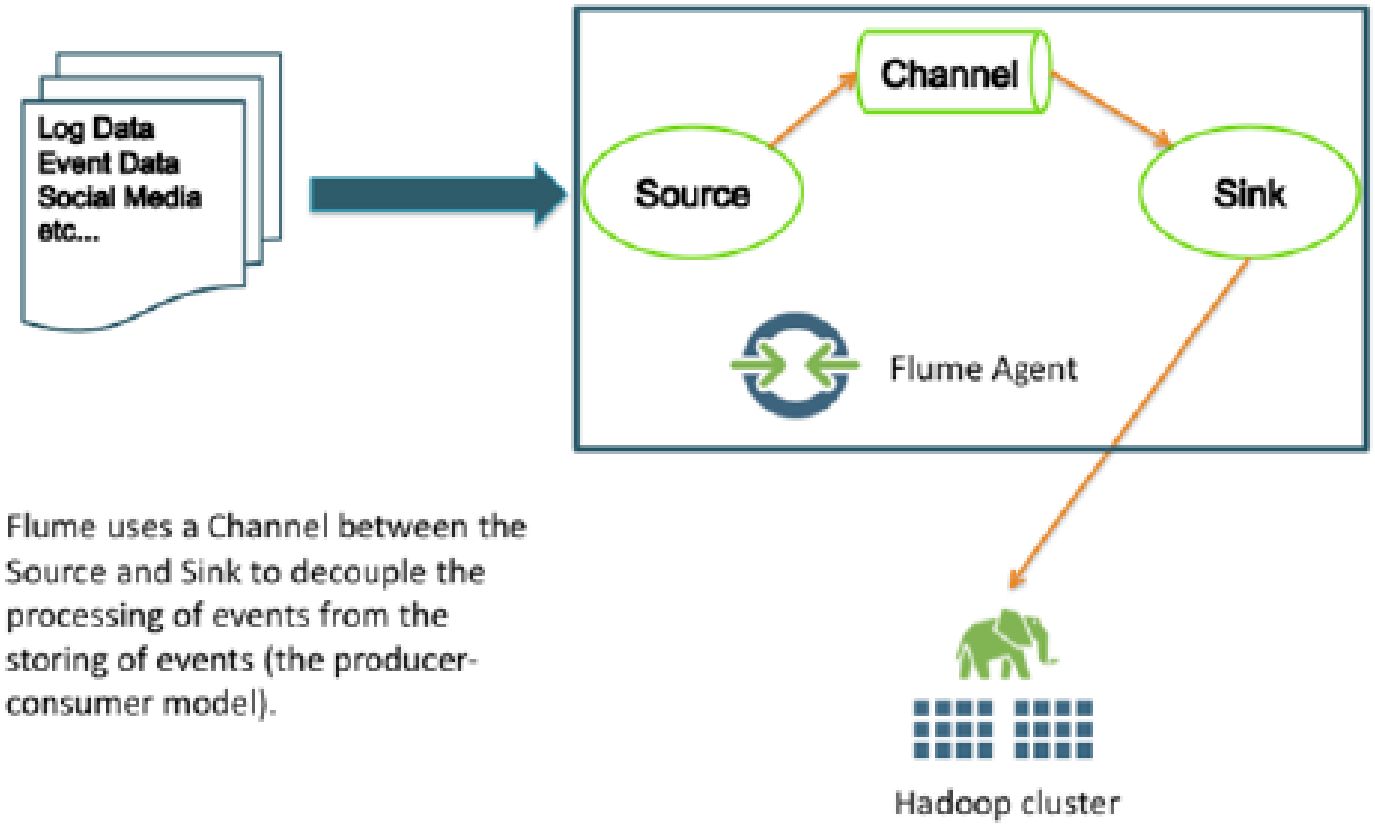
- Sink is the consumer of the events
- It consumes the data (events) from the channels and delivers it to the destination.
- The destination of the sink might be another agent or the central stores.
- A sink stores the data into centralized stores like HBase and HDFS.

Examples of a Sink include

- HDFS
- HBase
- Event Serializer: which converts the event data into a custom format and writes to an output stream

Flume Architecture

- Flume uses a producer-consumer model for handling events



Flume uses a Channel between the Source and Sink to decouple the processing of events from the storing of events (the producer-consumer model).

Failure Handling

In Flume, for each event, two transactions take place each,

- at the sender
- at the receiver.

- The sender sends events to the receiver.
- Soon after receiving the data, the receiver commits its own transaction and sends a “received” signal to the sender.
- After receiving the signal, the sender commits its transaction. (Sender will not commit its transaction till it receives a signal from the receiver.)

Fan out and Fan in Flow

Fan-out Flow

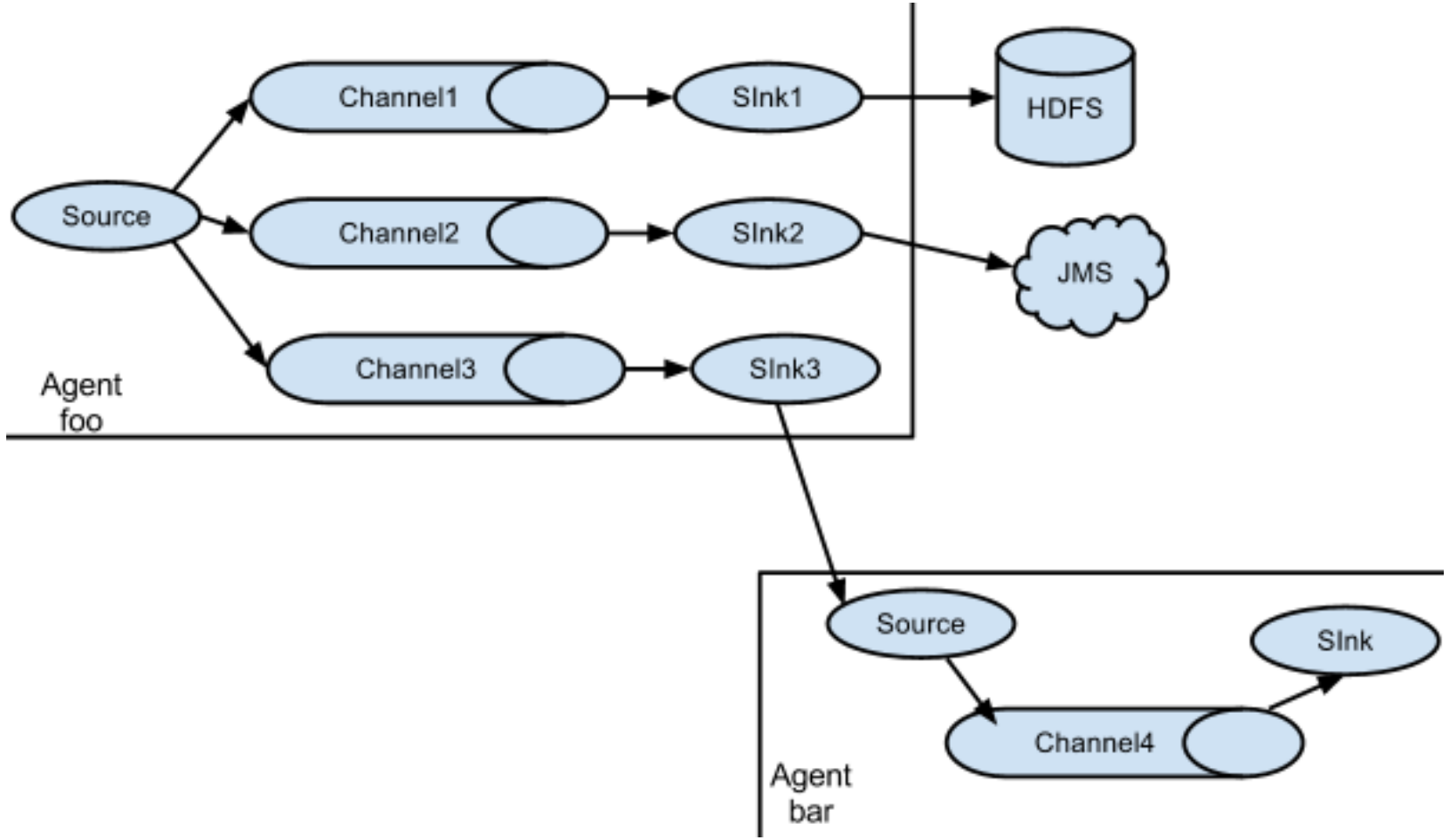
The dataflow from one source to multiple channels is known as **fan-out flow.**

- **Replicating** – The data flow where the data will be replicated in all the configured channels.
- **Multiplexing** – The data flow where the data will be sent to a selected channel which is mentioned in the header of the event.

Fan-in Flow

- The data flow in which the data will be transferred from many sources to one channel is known as **fan-in flow**

Example


Channel1
Sink1
HDFS
Source
Channel2
Sink2
JMS
Channel3
Sink3
Agent
foo
Source
Sink
Channel4
Agent
bar

Flume Configuration

Source Types

- The property named “type” is common to every source, and it is used to specify the type of the source we are using.

Sources

Type	Description	Implementation Class
avro	Avro Netty RPC event source. Listens on Avro port and receives events from external Avro streams.	AvroSource
netcat	Netcat style TCP event source. Listens on a given port and turns each line of text into an event.	NetcatSource
seq	Monotonically incrementing sequence generator event source	SequenceGeneratorSource
exec	Execute a long-lived Unix process and read from stdout.	ExecSource
spooldir	Used for ingesting data by placing files to be ingested into a "spooling" directory on disk.	SpoolDirectorySource
http	Accepts Flume events by HTTP POST and GET. GET should be used for experimentation only.	HTTPSource
org.apache.flume.source.jms.JMSSource	Reads messages from a JMS destination such as a queue or topic.	JMSSource

Sources Types…

NetCat Source

- A netcat-like source that listens on a given port and turns each line of text into an event.
- It opens a specified port and listens for data.

Property Name	Default	Description
channels	–	
type	–	The component type name, needs to be `netcat`
bind	–	Host name or IP address to bind to
port	–	Port # to bind to

Exec Source

- Exec source runs a given Unix command on start-up and expects that process to continuously produce data on standard out

Property Name	Default	Description
channels	–	
type	–	The component type name, needs to be `exec`
command	–	The command to execute

Sources Types…

Spooling Directory Source

- This source lets you ingest data by placing files to be ingested into a “spooling” directory on disk.

Property Name	Default	Description
channels	–	
type	–	The component type name, needs to be `spooldir`.
spoolDir	–	The directory from which to read files from.

JMS Source

- JMS Source reads messages from a JMS destination such as a queue or topic.

Syslog Sources

- Syslog is a way for network devices to send event messages to a logging server

Sink Types

logger	Log events at INFO level via configured logging subsystem	LoggerSink
avro	Sink that invokes a pre-defined Avro protocol method for all events it receives (when paired with an avro source, forms	AvroSink
hdfs	Writes all events received to HDFS (with support for rolling, bucketing, HDFS-200 append, and more)	HDFSEventSink
file_roll	Writes all events received to one or more files.	RollingFileSink
irc	Takes messages from attached channel and relays those to configured IRC destinations.	IRCSink
org.apache.flume.hbase.HBaseSink	A simple sink that reads events from a channel and writes them synchronously to HBase. The AsyncHBaseSink is	HBaseSink
org.apache.flume.sink.hbase.AsyncHBaseSink	A simple sink that reads events from a channel and writes them asynchronously to HBase. This is the recommended	AsyncHBaseSink
Other (custom)	You need to specify the fully-qualified name of the custom sink, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's lib	—

Sink Types…

HDFS Sink

- This sink writes events into the Hadoop Distributed File System (HDFS).

Name	Default	Description
channel	–	
type	–	The component type name, needs to be `hdfs`
hdfs.path	–	HDFS directory path (eg hdfs://namenode/flume/webdata/)

Hive Sink

- This sink streams events containing delimited text or JSON data directly into a Hive table or partition

Name	Default	Description
channel	–	
type	–	The component type name, needs to be `hive`
hive.metastore	–	Hive metastore URI (eg thrift://a.b.com:9083)
hive.database	–	Hive database name
hive.table	–	Hive table name

Sink Types…

Logger Sink

Logs event at INFO level. Typically useful for testing/debugging purpose.

Property Name	Default	Description
channel	–	
type	–	The component type name, needs to be `logger`

HBaseSink

This sink writes data to HBase.

Avro Sink

- Flume events sent to this sink are turned into Avro events and sent to the configured hostname / port pair.

Channel Types

Channels		
Type	**Description**	**Implementation Class**
memory	In-memory, fast, non-durable event transport	MemoryChannel
jdbc	JDBC-based, durable event transport (Derby-based)	JDBCChannel
file	File-based, durable event transport	FileChannel

Channel Types…

Memory Channel

- The events are stored in an in-memory queue

Property Name	Default	Description
type	–	The component type name, needs to be `memory`

JDBC Channel

- The events are stored in a persistent storage that's backed by a database

Property Name	Default	Description
type	–	The component type name, needs to be `jdbc`

File Channel

Property Name Default	Description
type	– The component type name, needs to be `file`.

Define Channel

- Configuration file is a Java property file having **key-value pairs**. We need to pass values to the keys in the file.

Define Channel

- Each Channel will have a separate list of properties
- Along with the property “type”, it is needed to provide the values of all the **required** properties of a particular source to configure it

```
agent_name.channels.channel_name.type = value
agent_name.channels.channel_name. property2 = value
agent_name.channels.channel_name. property3 = value
```

- Example:

```
# Define a memory channel on agent called memory-channel.

sample.channels.memory-channel.type = memory
```

Define Source and Connect to Channel

- Define Source

```
agent_name.sources. source_name.type = value
agent_name.sources. source_name.property2 = value
agent_name.sources. source_name.property3 = value
```

- Bind the source to Channel using Channels property

```
agent_name.sources.source_name.channels = channel_name
```

- Example:

```
# Define a source on agent and connect to channel memory-channel.

sample.sources.tail-source.type = netcat
sample.sources.tail-source.channels = memory-channel
sample.sources.tail-source.bind = 127.0.0.1
sample.sources.tail-source.port = 44444
```

Define Sink and Connect to Channel

- Define Sink

```
agent_name.sinks. sink_name.type = value
agent_name.sinks. sink_name.property2 = value
agent_name.sinks. sink_name.property3 = value
```

- Bind the sink to Channel using Channels property

```
agent_name.sinks.sink_name.channels = channel_name
```

- Example:

```
# Define a sink that outputs to logger.
sample.sinks.log-sink.channel = memory-channel
sample.sinks.log-sink.type = logger
```

```
# Define a sink that outputs to hdfs.
sample.sinks.hdfs-sink.channel = memory-channel
sample.sinks.hdfs-sink.type = hdfs
sample.sinks.hdfs-sink.hdfs.path = hdfs://quickstart.cloudera:8020/user/cloudera/muralilog
sample.sinks.hdfs-sink.hdfs.fileType = DataStream
```

Activate – Naming the Components

- Name the components such as sources, sinks, and the channels of the agent

```
agent_name.sources = source_name
agent_name.sinks = sink_name
agent_name.channels = channel_name
```

- Example

```
# Finally, activate.
sample.channels = memory-channel
sample.sources = tail-source
sample.sinks = log-sink hdfs-sink
```

Agent Execution

Flume Agent - Execution Steps

- *Step 1: Copy the flume.conf from local to linux*

- *Step 2:* Enter sudo –i in cloudera prompt and enter

- Step 3: You will be in root directory and execute the below copy command

  cp /home/cloudera/flume/flume.conf /etc/flume-ng/conf.empty

- Step 4: Enter exit and press enter. You will be back to cloudera prompt

- Step 5: Now execute the below script from cloudera prompt

  /usr/lib/flume-ng/bin/flume-ng agent --conf conf --conf-file /usr/lib/flume-ng/conf/flume.conf --name sample -Dflume.root.logger=INFO,console

Flume Agent - Execution Steps…

- *Step 6:* Flume agent starts running, this will take some time and all memory channel, source and finally sink will well be set and sink will be looking at the log file that is defined in the source path above.

- *Step 7:* Open another instance and type
  - $ nc localhost 44444 <enter>
  - Enter some text

- Step 8: Open another instance and execute the following command
  - $ hadoop fs –cat muralilog/*.*

- Step 9: Check that text type will appear on the screen and also in the hdfs

Hands On

Flume Hands On

- Create a flume agent to read data from the cloudera provided customer twitter
- Take the help of below URL and completed downloading from twitter to Hadoop HDFS :
- http://www.thecloudavenue.com/2013/03/analyse-tweets-using-flume-Hadoop-and.html

Thank You

Keerthiga Barathan