Hortonworks Data Platform

Apache Flume Component Guide

(May 17, 2018)

docs.hortonworks.com

Hortonworks Data Platform: Apache Flume Component Guide

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

List of Tables

1. Feature Updates

Apache Flume version 1.5.2 includes cumulative 1.6 features and selected 1.7 features. The table below indicates the features added to Flume 1.5.2 with each release of Hortonworks Data Platform (HDP).

Note

As of HDP 2.6.0, Flume is deprecated but still supported. It will be removed and no longer supported as of HDP 3.0.0.

Table 1.1. Features by HDP Version

HDP Release	Added Features	Advantages
2.6.2	New Kafka Source	Provides support for new Kafka Consumer API, including reading multiple Kafka topics.
	New Kafka sink	Provides support for new Kafka Producer API.
	Preserve event headers	Preserve event headers when using Kafka Sink and Kafka Source together. Avro Datums can be read from and written to Kafka.
	Log privacy utility	Helps determine whether or not logging potentially sensitive information is allowed.
2.5.0	Kafka Channel	Uses a single Kafka topic. Provides greater reliability and better performance.
	TailDir Source	Greater data reliability, even with rotating file names. Can restart tailing at the point where Flume stopped, while continuing data ingest.
2.4.0	Kafka Source	Reads messages from a Kafka topic. Can have multiple Kafka sources running and configure them to read a unique set of partitions for the topic.
	Kafka Sink	Publishes data to a Kafka topic. Supports pull-based processing from various Flume sources.
2.3.0	Hive Sink	Not recommended for use in production. Streams events containing delimited text or JSON data directly into a Hive table or partition. Provides a preview feature and not.

2. Introduction

Flume is a top-level project at the Apache Software Foundation. While it can function as a general-purpose event queue manager, in the context of Hadoop it is most often used as a log aggregator, collecting log data from many diverse sources and moving them to a centralized data store.

The following information is available in this chapter:

2.1. Flume Concepts

Flume Components

A Flume data flow is made up of five main components: Events, Sources, Channels, Sinks, and Agents:

Events An event is the basic unit of data that is moved using Flume. It is similar to a message in JMS and is generally small. It is made up of headers and a byte-array body.

Sources The source receives the event from some external entity and stores it in a channel. The source must understand the type of event that is sent to it: an Avro event requires an Avro source.

Channels A channel is an internal passive store with certain specific characteristics. An in-memory channel, for example, can move events very quickly, but does not provide persistence. A file-based channel provides persistence. A source stores an event in the channel where it stays until it is consumed by a sink. This temporary storage lets source and sink run asynchronously.

Sinks The sink removes the event from the channel and forwards it to either to a destination, like HDFS, or to another agent/dataflow. The sink must output an event that is appropriate to the destination.

Agents An agent is the container for a Flume data flow. It is any physical JVM running Flume. An agent must contain at least one source, channel, and sink, but the same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

2.2. HDP and Flume

Flume ships with many source, channel, and sink types. The following types have been thoroughly tested for use with HDP:

Sources

- Exec (basic, restart)
- Syslogtcp
- Syslogudp
- TailDir
- Kafka

Channels

- Memory
- File
- Kafka

Sinks

- HDFS: secure, nonsecure
- HBase
- Kafka
- Hive

See the Apache Flume 1.5.2 documentation for a complete list of all available Flume components.

2.3. A Simple Example

The following snippet shows some of the kinds of properties that can be set using the properties file. For more detailed information, see the Apache Flume 1.5.2 documentation.

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory

agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = tail -f /etc/passwd

agent.sinks = hdfsSinkagent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://hdp/user/root/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

The source here is defined as an exec source. The agent runs a given command on startup, which streams data to `stdout`, where the source gets it.

In this case, the command is a Python test script. The channel is defined as an in-memory channel and the sink is an HDFS sink.

2.4. Flume 1.5.2 Documentation

See the complete Apache Flume 1.5.2 documentation for details about using Flume with Hortonworks Data Platform. The documentation is updated to reflect support for new features.

The Flume 1.5.2 documentation is also included with the Flume software. You can access the documentation in the Flume `/docs` directory.

3. Using Apache Flume for Streaming

Kafka Sink and Hive Sink are integrated with Flume to provide streaming capabilitles for Hive tables and Kafka topics. For more information about Flume, see the Apache Flume 1.5.2 documentation.

3.1. Kafka Sink

This is a Flume Sink implementation that can publish data to a Kafka topic. One of the objectives is to integrate Flume with Kafka so that pull-based processing systems can process the data coming through various Flume sources. This currently supports Kafka 0.8.x series of releases.

Property Name	Default	Description
type	-	Must be set to org.apache.flume.sink.kafka.KafkaSink.
brokerList	-	List of brokers Kafka-Sink will connect to, to get the list of topic partitions. This can be a partial list of brokers, but we recommend at least two for HA. The format is a comma separated list of hostname:port.
topic	default-flume-topic	The topic in Kafka to which the messages will be published. If this parameter is configured, messages will be published to this topic. If the event header contains a “topic” field, the event will be published to that topic overriding the topic configured here.
batchSize	100	How many messages to process in one batch. Larger batches improve throughput while adding latency.
requiredAcks	1	How many replicas must acknowledge a message before it is considered successfully written. Accepted values are 0 (Never wait for acknowledgement), 1 (wait for leader only), -1 (wait for all replicas) Set this to -1 to avoid data loss in some cases of leader failure.
Other Kafka Producer Properties	-	These properties are used to configure the Kafka Producer. Any producer property supported by Kafka can be used. The only requirement is to prepend the property name with the prefix "Kafka.". Fr exampleafka.producer.type.

Kafka Sink uses the topic and key properties from the FlumeEvent headers to send events to Kafka. If the topic exists in the headers, the event is sent to that specific topic, overriding the topic configured for the Sink. If key exists in the headers, the key is used by Kafka to partition the data between the topic partitions. Events with the same key are sent to the same partition. If the key is null, events are sent to random partitions.

An example configuration of a Kafka sink is given below. Properties starting with the prefix Kafka (the last 3 properties) are used when instantiating the Kafka producer.

The properties that are passed when creating the Kafka producer are not limited to the properties given in this example. It is also possible include your custom properties here and access them inside the preprocessor through the Flume Context object passed in as a method argument.

```
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink a1.sinks.k1.topic =
 mytopic
a1.sinks.k1.brokerList = localhost:9092
a1.sinks.k1.requiredAcks = 1
a1.sinks.k1.batchSize = 20
a1.sinks.k1.channel = c1
```

3.2. Hive Sink

This sink streams events containing delimited text or JSON data directly into a Hive table or partition. Events are written using Hive transactions. As soon as a set of events are committed to Hive, they become immediately visible to Hive queries. Partitions to which flume will stream to can either be pre-created or, optionally, Flume can create them if they are missing. Fields from incoming event data are mapped to corresponding columns in the Hive table.

Property Name	Default	Description
channel	–	
type	–	The component type name, needs to be hive.
hive.metastore	–	Hive metastore URI (e.g. thrift://a.b.com:9083).
hive.database	–	Hive database name .
hive.table	–	Hive table name.
hive.partition	–	Comma separated list of partition values identifying the partition to write to. May contain escape sequences. E.g.: If the table is partitioned by (continent: string, country :string, time : string) then 'Asia,India,2014-02-26-01-21' will indicate continent=Asia,country=India,time=2014-02-26-01-21.
hive.txnsPerBatchAsk	100	Hive grants a batch of transactions instead of single transactions to streaming clients like Flume. This setting configures the number of desired transactions per Transaction Batch. Data from all transactions in a single batch end up in a single file. Flume will write a maximum of batchSize events in each transaction in the batch. This setting in conjunction with batchSize provides control over the size of each file. Note that eventually Hive will transparently compact these files into larger files.
heartBeatInterval	240	(In seconds) Interval between consecutive heartbeats sent to Hive to keep unused transactions from expiring. Set this value to 0 to disable heartbeats.

Property Name	Default	Description
autoCreatePartitions	true	Flume will automatically create the necessary Hive partitions to stream to.
batchSize	15000	Max number of events written to Hive in a single Hive transaction.
maxOpenConnections	500	Allow only this number of open connections. If this number is exceeded, the least recently used connection is closed.
callTimeout	10000	(In milliseconds) Timeout for Hive & HDFS I/O operations, such as openTxn, write, commit, abort.
serializer	–	Serializer is responsible for parsing out field from the event and mapping them to columns in the hive table. Choice of serializer depends upon the format of the data in the event. Supported serializers: DELIMITED and JSON.
roundUnit	minute	The unit of the round down value - second, minute or hour.
roundValue	1	Rounded down to the highest multiple of this (in the unit configured using hive.roundUnit), less than current time.
timeZone	Local	Name of the timezone that should be used for resolving the escape sequences in partition, e.g. Time America/Los_Angeles.
useLocalTimeStamp	false	Use the local time (instead of the timestamp from the event header) while replacing the escape sequences.

The following serializers are provided for Hive sink:

- **JSON:** Handles UTF8 encoded Json (strict syntax) events and requires no configuration. Object names in the JSON are mapped directly to columns with the same name in the Hive table. Internally uses org.apache.hive.hcatalog.data.JsonSerDe but is independent of the Serde of the Hive table. This serializer requires HCatalog to be installed.
- **DELIMITED:** Handles simple delimited textual events. Internally uses LazySimpleSerde but is independent of the Serde of the Hive table.

Property Name	Default	Description
serializer.delimiter	,	(Type: string) The field delimiter in the incoming data. To use special characters, surround them with double quotes like "\t".
serializer.fieldnames	–	The mapping from input fields to columns in hive table. Specified as a comma separated list (no spaces) of hive table columns names, identifying the input fields in order of their occurrence. To skip fields leave the column name unspecified. E.g.. 'time,,IP,message' indicates the 1st, 3rd and 4th fields in input map to time, IP and message columns in the hive table.
serializer.serdeSeparator	Ctrl-A	(Type: character) Customizes the separator used by underlying serde.

Property Name	Default	Description
		There can be a gain in efficiency if the fields in serializer.fieldnames are in same order as table columns, the serializer.delimiter is same as the serializer.serdeSeparator and number of fields in serializer.fieldnames is less than or equal to number of table columns, as the fields in incoming event body do not need to be reordered to match order of table columns. Use single quotes for special characters like '\t'. Ensure input fields do not contain this character. **Note:** If serializer.delimiter is a single character, preferably set this to the same character.

The following are the escape sequences supported:

Alias	Description
%{host}	Substitute value of event header named "host". Arbitrary header names are supported.
%t	Unix time in milliseconds .
%a	Locale's short weekday name (Mon, Tue, ...)
%A	Locale's full weekday name (Monday, Tuesday, ...)
%b	Locale's short month name (Jan, Feb, ...)
%B	Locale's long month name (January, February, ...)
%c	Locale's date and time (Thu Mar 3 23:05:25 2005)
%d	Day of month (01)
%D	Date; same as %m/%d/%y
%H	Hour (00..23)
%I	Hour (01..12)
%j	Day of year (001..366)
%k	Hour (0..23)
%m	Month (01..12)
%M	Minute (00..59)
%p	Locale's equivalent of am or pm
%s	Seconds since 1970-01-01 00:00:00 UTC
%S	Second (00..59) %y last two digits of year (00..99)
%Y	Year (2015)
%z	+hhmm numeric timezone (for example, -0400)

Example Hive table:

```
create table weblogs ( id int , msg string )

partitioned by (continent string, country string, time string)
clustered by (id) into 5 buckets
stored as orc;
```

Example for agent named a1:

```
a1.channels = c1

a1.channels.c1.type = memory

a1.sinks = k1

a1.sinks.k1.type = hive

a1.sinks.k1.channel = c1

a1.sinks.k1.hive.metastore = thrift://127.0.0.1:9083
a1.sinks.k1.hive.database = logsdb

a1.sinks.k1.hive.table = weblogs
a1.sinks.k1.hive.partition = asia,%{country},%y-%m-%d-%H-%M
a1.sinks.k1.useLocalTimeStamp = false

a1.sinks.k1.round = true

a1.sinks.k1.roundValue = 10

a1.sinks.k1.roundUnit = minute

a1.sinks.k1.serializer = DELIMITED
a1.sinks.k1.serializer.delimiter = "\t"
a1.sinks.k1.serializer.serdeSeparator = '\t'
a1.sinks.k1.serializer.fieldnames =id,,msg
```

Tip

For all of the time related escape sequences, a header with the key “timestamp” must exist among the headers of the event (unless useLocalTimeStampis set to true). One way to add this automatically is to use the TimestampInterceptor.

The above configuration will round down the timestamp to the last 10th minute. For example, an event with timestamp header set to 11:54:34 AM, June 12, 2012 and ‘country’ header set to ‘india’ will evaluate to the partition (continent=’asia’,country=’india’,time=’2012-06-12-11-50’. The serializer is configured to accept tab separated input containing three fields and to skip the second field.