tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Flume is a standard, simple, robust, flexible, and extensible tool for data ingestion from various data producers (webservers) into Hadoop. In this tutorial, we will be using simple and illustrative example to explain the basics of Apache Flume and how to use it in practice.

Audience

This tutorial is meant for all those professionals who would like to learn the process of transferring log and streaming data from various webservers to HDFS or HBase using Apache Flume.

Prerequisites

To make the most of this tutorial, you should have a good understanding of the basics of Hadoop and HDFS commands.

Copyright & Disclaimer

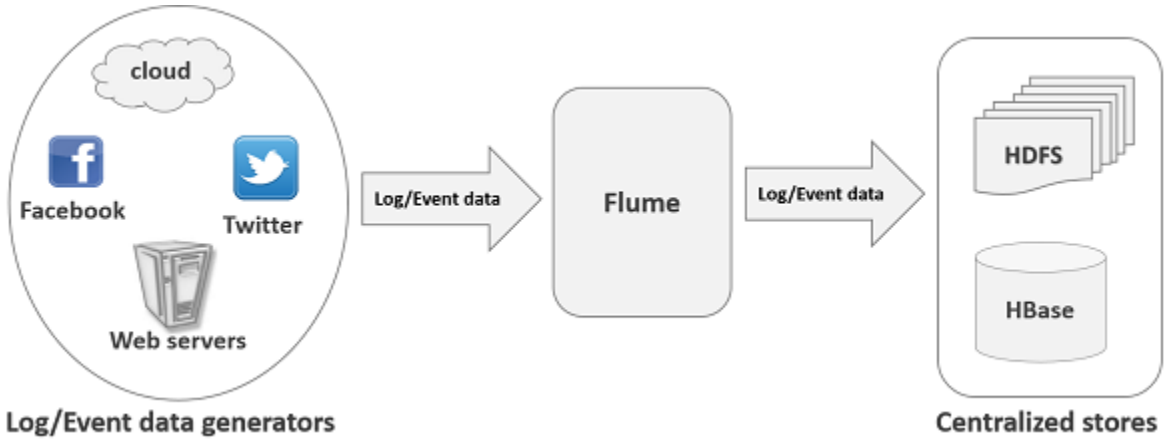
Table of Contents

1. FLUME – INTRODUCTION

What is Flume?

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a centralized data store.

Flume is a highly reliable, distributed, and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.



Applications of Flume

Assume an e-commerce web application wants to analyze the customer behavior from a particular region. To do so, they would need to move the available log data in to Hadoop for analysis. Here, Apache Flume comes to our rescue.

Flume is used to move the log data generated by application servers into HDFS at a higher speed.

Advantages of Flume

Here are the advantages of using Flume:

- Using Apache Flume we can store the data in to any of the centralized stores (HBase, HDFS).

- When the rate of incoming data exceeds the rate at which data can be written to the destination, Flume acts as a mediator between data producers and the centralized stores and provides a steady flow of data between them.

- Flume provides the feature of **contextual routing**.

- The transactions in Flume are channel-based where two transactions (one sender and one receiver) are maintained for each message. It guarantees reliable message delivery.

- Flume is reliable, fault tolerant, scalable, manageable, and customizable.

Features of Flume

Some of the notable features of Flume are as follows:

- Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.

- Using Flume, we can get the data from multiple servers immediately into Hadoop.

- 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 supports a large set of sources and destinations types.

- Flume supports multi-hop flows, fan-in fan-out flows, contextual routing, etc.

- Flume can be scaled horizontally.

2. FLUME – DATA TRANSFER IN HADOOP

Big Data, as we know, is a collection of large datasets that cannot be processed using traditional computing techniques. Big Data, when analyzed, gives valuable results. **Hadoop** is an open-source framework that allows to store and process Big Data in a distributed environment across clusters of computers using simple programming models.

Streaming / Log Data

Generally, most of the data that is to be analyzed will be produced by various data sources like applications servers, social networking sites, cloud servers, and enterprise servers. This data will be in the form of **log files** and **events**.

Log file: In general, a log file is a **file** that lists events/actions that occur in an operating system. For example, web servers list every request made to the server in the log files.

On harvesting such log data, we can get information about:

- the application performance and locate various software and hardware failures.
- the user behavior and derive better business insights.

The traditional method of transferring data into the HDFS system is to use the **put** command. Let us see how to use the **put** command.

HDFS put Command

The main challenge in handling the log data is in moving these logs produced by multiple servers to the Hadoop environment.

Hadoop **File System Shell** provides commands to insert data into Hadoop and read from it. You can insert data into Hadoop using the **put** command as shown below.

```
$ Hadoop fs –put /path of the required file  /path in HDFS where to save the file
```

Problem with put Command

We can use the **put** command of Hadoop to transfer data from these sources to HDFS. But, it suffers from the following drawbacks:

- Using **put** command, we can transfer **only one file at a time** while the data generators generate data at a much higher rate. Since the analysis made on older data is less accurate, we need to have a solution to transfer data in real 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.

What we need here is a solutions that can overcome the drawbacks of **put** command and transfer the "streaming data" from data generators to centralized stores (especially HDFS) with less delay.

Problem with HDFS

In HDFS, the file exists as a directory entry and the length of the file will be considered as zero till it is closed. For example, if a source is writing data into HDFS and the network was interrupted in the middle of the operation (without closing the file), then the data written in the file will be lost.

Therefore we need a reliable, configurable, and maintainable system to transfer the log data into HDFS.

Note: In POSIX file system, whenever we are accessing a file (say performing write operation), other programs can still read this file (at least the saved portion of the file). This is because the file exists on the disc before it is closed.

Available Solutions

To send streaming data (log files, events etc..,) from various sources to HDFS, we have the following tools available at our disposal:

Facebook's Scribe

Scribe is an immensely popular tool that is used to aggregate and stream log data. It is designed to scale to a very large number of nodes and be robust to network and node failures.

Apache Kafka

Kafka has been developed by Apache Software Foundation. It is an open-source message broker. Using Kafka, we can handle feeds with high-throughput and low-latency.

Apache Flume

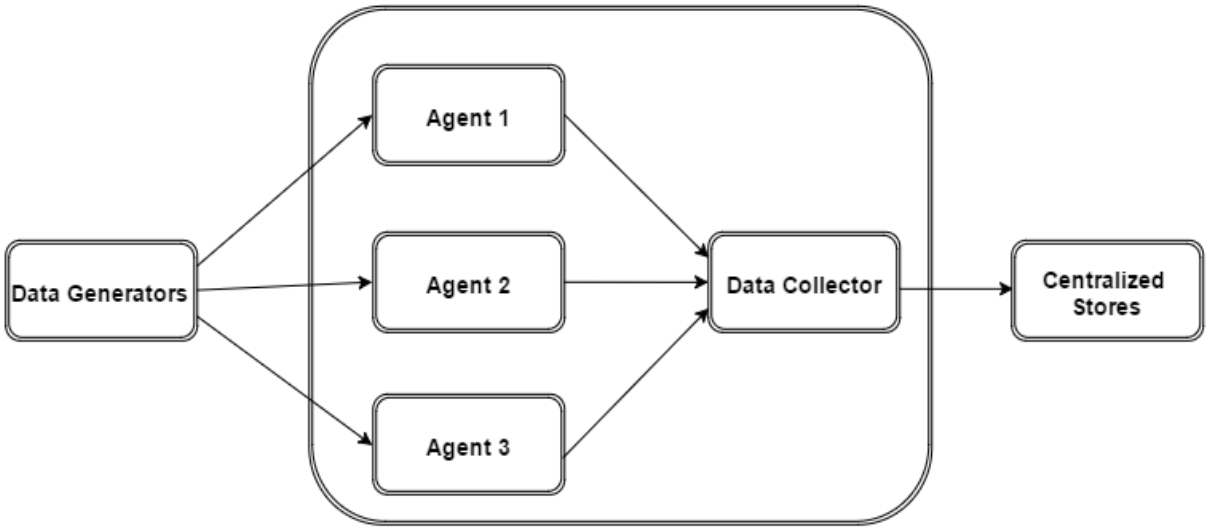
Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log data, events (etc...) from various webserves to a centralized data store.

It is a highly reliable, distributed, and configurable tool that is principally designed to transfer streaming data from various sources to HDFS.

In this tutorial, we will discuss in detail how to use Flume with some examples.

3. FLUME – ARCHITECTURE

The following illustration depicts the basic architecture of Flume. As shown in the illustration, **data generators** (such as Facebook, Twitter) generate data which gets collected by individual Flume **agents** running on them. Thereafter, a **data collector** (which is also an agent) collects the data from the agents which is aggregated and pushed into a centralized store such as HDFS or HBase.



Flume

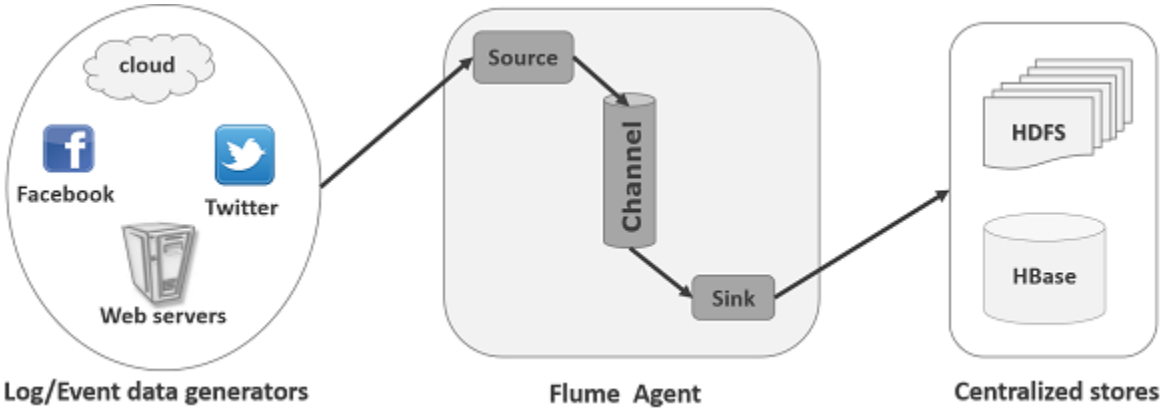
Flume Event

An **event** is the basic unit of the data transported inside **Flume**. It contains a payload of byte array that is to be transported from the source to the destination accompanied by optional headers. A typical Flume event would have the following structure:

Flume event

Flume Agent

Take a look at the following illustration. It shows the internal components of an agent and how they collaborate with each other.



An **agent** is an independent daemon process (JVM) in Flume. It receives the data (events) from clients or other agents and forwards it to their next destination.

A Flume Agent contains three main components namely, **source**, **channel**, and **sink**.

Source

A **source** receives data from the log/event data generators such as Facebook, Twitter, and other webservers, and transfers it to the channel in the form of Flume events.

Data generators like webservers generate data and deliver it to the agent. A **source** is a component of the agent which receives this data and transfers it to one or more channels.

Apache Flume supports several types of sources and each source receives events from a specified data generator. For example, Avro source receives data from the clients which generate data in the form of Avro files.

Flume supports the following sources: Avro, Exec, Spooling directory, Net Cat, Sequence generator, Syslog, Multiport TCP, Syslog UDP, and HTTP.

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.

These channels are fully transactional and they can work with any number of sources and sinks. **Example**: JDBC channel, File system channel, Memory channel, etc.

Sink

Finally, the **sink** stores the data into centralized stores like HBase and HDFS. 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. Example: HDFS sink.

Flume supports the following sinks: HDFS sink, Logger, Avro, Thrift, IRC, File Roll, Null sink, HBase, and Morphline solr.

Additional Components of Flume Agent

What we have discussed above are the primitive components of the agent. In addition to this, we have a few more components that play a vital role in transferring the events from the data generator to the centralized stores.

Interceptors

Interceptors are used to alter/inspect flume events which are transferred between source and channel.

Channel Selectors

These are used to determine which channel is to be opted to transfer the data in case of multiple channels. There are two types of channel selectors:

- **Default channel selectors**: These are also known as replicating channel selectors they replicates all the events in each channel.

- **Multiplexing channel selectors**: These decides the channel to send an event based on the address in the header of that event.

Sink Processors

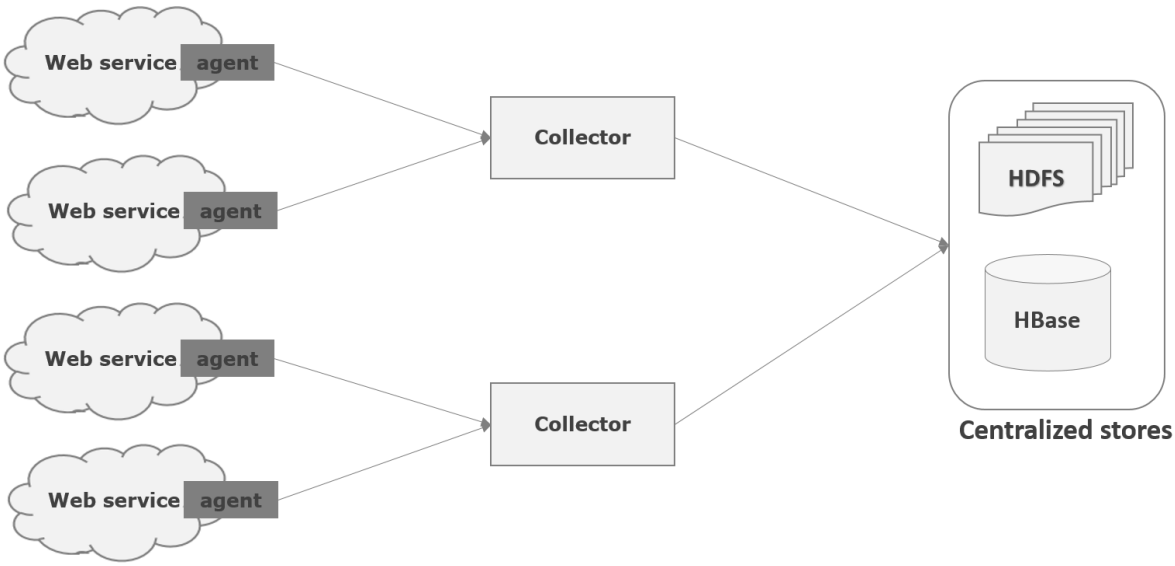
These are used to invoke a particular sink from the selected group of sinks. These are used to create failover paths for your sinks or load balance events across multiple sinks from a channel.

4. FLUME – DATA FLOW

Flume is a framework which is used to move log data into HDFS. Generally events and log data are generated by the log servers and these servers have Flume agents running on them. These agents receive the data from the data generators.

The data in these agents will be collected by an intermediate node known as **Collector**. Just like agents, there can be multiple collectors in Flume.

Finally, the data from all these collectors will be aggregated and pushed to a centralized store such as HBase or HDFS. The following diagram explains the data flow in Flume.



Multi-hop Flow

Within Flume, there can be multiple agents and before reaching the final destination, an event may travel through more than one agent. This is known as **multi-hop flow**.

Fan-out Flow

The dataflow from one source to multiple channels is known as **fan-out flow**. It is of two types:

- **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**.

Failure Handling

In Flume, for each event, two transactions take place: one at the sender and one 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.)

5. FLUME– ENVIRONMENT

We already discussed the architecture of Flume in the previous chapter. In this chapter, let us see how to download and setup Apache Flume.

Before proceeding further, you need to have a Java environment in your system. So first of all, make sure you have Java installed in your system. For some examples in this tutorial, we have used Hadoop HDFS (as sink). Therefore, we would recommend that you go install Hadoop along with Java. To collect more information, follow the link: http://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm

Installing Flume

First of all, download the latest version of Apache Flume software from the website https://flume.apache.org/.

Step 1

Open the website. Click on the **download** link on the left-hand side of the home page. It will take you to the download page of Apache Flume.

Step 2

In the Download page, you can see the links for binary and source files of Apache Flume. Click on the link apache-flume-1.6.0-bin.tar.gz

You will be redirected to a list of mirrors where you can start your download by clicking any of these mirrors. In the same way, you can download the source code of Apache Flume by clicking on apache-flume-1.6.0-src.tar.gz.

Step 3

Create a directory with the name Flume in the same directory where the installation directories of **Hadoop**, **HBase**, and other software were installed (if you have already installed any) as shown below.

```
$ mkdir Flume
```

Step 4

Extract the downloaded tar files as shown below.

```
$ cd Downloads/
$ tar zxvf apache-flume-1.6.0-bin.tar.gz
$ tar zxvf apache-flume-1.6.0-src.tar.gz
```

Step 5

Move the content of apache**-flume-1.6.0-bin.tar** file to the **Flume** directory created earlier as shown below. (Assume we have created the Flume directory in the local user named Hadoop.)

```
$ mv apache-flume-1.6.0-bin.tar/* /home/Hadoop/Flume/
```

Configuring Flume

To configure Flume, we have to modify three files namely, **flume-env.sh**, **flume-conf.properties,** and **bash.rc**.

Setting the Path / Classpath

In the **.bashrc** file, set the home folder, the path, and the classpath for Flume as shown below.