Back in 2010 (IIRC), LinkedIn wanted to move to something different to handle event log processing because the existing system was fragile, not very scalable, and used XML (!) as the format.  Flume(v1) was still very much in its early days.  Cloudera was invited over to talk about Flume to see if it would meet our needs.  Two big points came out of that discussion:

1. Flume was extremely hard to manage because it didn't multiplex connections.  It was essentially one socket per log type.  This was thought to be sort of ridiculous, especially given that syslog and other daemons had been doing this for a very long time.  To add to the overhead, Flume required that there be a different configuration file per socket that was opened.
2. We wanted a client pull rather than a  push to client for various reasons, many of which [Jay Kreps](https://www.quora.com/profile/Jay-Kreps) and others from the Kafka team have documented elsewhere.  It was asked if Flume could be modified to reverse the polarity a bit.  The answer was (mostly) no.  ("Patches accepted!")

It was pretty clear that the team was going to need to build something different to handle our needs.  Thus Kafka was born.  
  
Since that time, I believe the first point has been fixed but the second is still true in Flume v2.

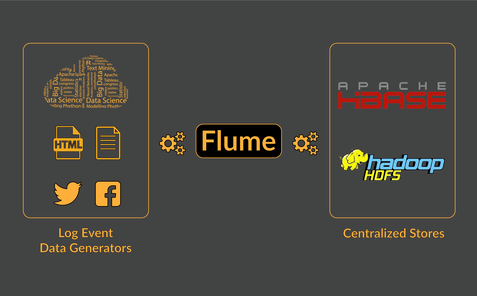
**Apche Hadoop Flume Tutorial**

**What is Apache Flume?**

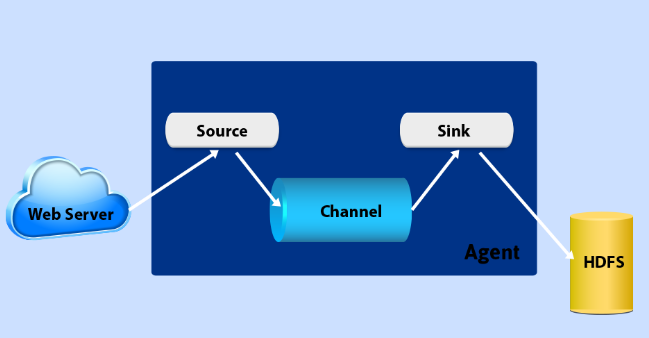
Apache Flume is one tool and used to moving data from one place to another place.Flume is the distributed systems that transporting the data at reliable manner.Flume is most important part of hadoop ecosystem. In Apache flume all data unit consider as one event. It collecting log data from various [web servers to HDFS.](http://www.besthadooptraining.in/blog/hadoop-ecosystem-tutorial/)

**Features of Apache Flume:**

* Main Feature of flume is collected data from multiple web servers
* It import large amount of data that produced by facebook,twitter.
* It supports Fan-in-Fan-out flows and more amount of sources and destination types.
* It collects the data from multiple sources and move to destination.

[](http://i2.wp.com/www.besthadooptraining.in/wp-content/uploads/2017/05/Flume-08-1-min.png)

**Main Components of apche Flume:**



**Event:**

All data units are event in Apache flume. Main Purpose of event is tranport the data from source to destination and it have byte array for data storage.It tranport single data to destination.

**Agent:**

Work of agent is get data from client or another agent and transferred to next destination.

**Source:**

Source is the sub component of agent.It collect data from generators and send to the channels.Flume supports more number of sources and receives data from generators. It is the main component of flume because it enter the data into flume.

**Sink:**

Flume contains more number of sinks also. It deletes the data from channels and move that data to next destination. Destination of sink is the one type of agent.

**Channel:**

Channels are collect the events from sources and deleted by sinks. Channel can work with many number of sinks and sources.

**Client:**

Events are made by clients and send that events to agents

**Channel Sectors:**

Main purpose of channel sectors is determines which channel are transfer the data. There are two types of channel Sectors

[**Default Channel sectors**](http://www.credosystemz.com/training-in-chennai/best-big-data-training-in-chennai/) – it send all events to all channel

[**Multiplexing Channel Sectors**](http://www.credosystemz.com/training-in-chennai/best-big-data-training-in-chennai/) – It sents events based on address of channel

**Data Flow in Flume:**

Flume contains following four types of data flow

**1.Multi hop Flow:**

Flume contails more number of agents and if event travel thorugh one event it is called multi hop flow

**2.Fan out Flow:**

The data send one source to muliple channel in flume is called fan out flow. There are two types of fan out flow

* Replicating work flow
* Multiplexing work flow

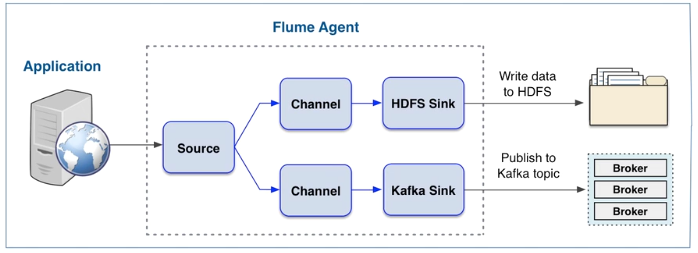
**3.Fan in Flow:**

The data send one channel to multiple sources are called Fan in Flow

**4.Failure Handling:**

All events have two transactions that are sender and receiver. Sender send the data to receiver and if once receiver receives the data and send received signal to sender. Once sender receives the signal and transaction made by sender. If no signal received by sender transaction should not be processed.

FLUME:



**Apache Kafka Architecture and Components**

**What is Apache Kafka?**

Kafka is designed for distributed systems.It mainly used to transfer data from **Hadoop using the messaging system.**Messaging system means transferring data from one application to another one but it does not consider how to transfer data and is based on message queuing.There are two types of messaging system in Kafka.

1.Point to Point

2.Publish-Subscribe

Apache Kafka is the publish – Subscribe messaging system and is suitable for online and offline messages.Kafka has prevented the data without any data loss. Kafka runs on top of the Zookeeper.

**Architecture and Components of Apache Kafka:**

Apache Kafka having following six components

1. Broker

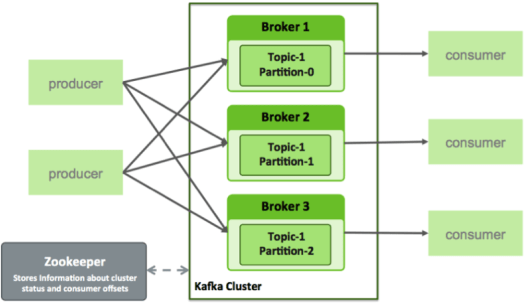
2. Producer

3. Consumer

4. Zookeeper

5. Kafka Cluster

6. Follower

[](http://i0.wp.com/www.besthadooptraining.in/wp-content/uploads/2017/05/kafka.png)

**1. Broker:**

Brokers have maintained the Kafka load balance and published data. kafka having more number of brokers. In Kafka zookeeper are used to analyze the broker state. One broker can handle multiple numbers of messages at one system without any data loss.

**2. Producer:**

The producer is sent data to the broker. If any new broker is created procedure automatically sends the message to the broker. It cannot wait for broker acknowledgments. Producer message format is segmented files.

**3. Consumer:**

Main purposes of the consumer are read broker data and publish the broker messages. It consumes a number of data within one second. If one message comes to consumer and we are sending the message to all consumer group.

**4.Zookeeper:**

Zookeeper maintains the broker, producer and consumer data. Zookeeper has identified any failure in broker and send that failure message to the consumer. If one receives the message it takes a decision for that failure broker.

**5.Kafka Cluster:**

Kafka cluster having more number of broker for maintains the broker data.

**6.Follower:**

It also is known as a consumer and it updates the data. The follower is following the leader instructions. If one leader fails automatically creates the one leader.

***Differences:***

***Flume ingests data*** into Hadoop. It is tightly integrated with Hadoop’s monitoring system, file system, file formats, and utilities. Flume’s components - sources, sinks and channels implies that it can be used to move data between other systems flexibly.

***Kafka*** is a distributed ***publish-subscribe messaging system***. It allows consumers/subscribers to read exactly the messages they are interested from the publishers/producers, and makes sure that the distributed system is scalable and reliable.

This question implies that Kafka and Flume are interchangeable components. It makes as much sense to me as “Should we use cars or umbrellas?”. Sure, you can hide from the rain in your car and you can use your umbrella when moving from place to place. But in general, these are different tools intended for different use-cases.

Flume push events from the source to sink via channels while Kafka pull message from different kafka topic replicated across the brokers.

Flume-ng client is not matured enough to talk to hive while Kafka message consumer can talk to hive via any IDL viz spark etc.

Flume is three tiered architecture while Kafka is distributed architecture consists of number of broker node which are highly fault tolerant

1. Kafka is a distributed cluster architecture having number of broker co-ordinated by Zookeeper.
2. Flume is a three tier architecture consisting of source/channel/sinks.
3. Kafka with spark streaming gives wide range of scope for sql queries.
4. Flume doesn’t support any SQL queries.
5. Kafka with spark streaming provides real time analytics.
6. Flume is use to dump the data from source to sink and much used in batch analytics.
7. Kafka is more much fault tolerant as it supports replication.
8. Flume doesn’t support replication.
9. Kafka is much faster and more reliable in data ingestion.
10. Flume is not much faster and reliable than Kafka.
11. Kafka is message queuing system which is topic based.
12. Flume doesn’t supports any topic and message queuing.

**Flume**: 

* Flume does not replicate events - in case of flume-agent failure, you will lose events in the channel

**Kafka**:

* A general purpose distributed publish-subscribe messaging system - you can have a topic which many listeners can subscribe to so the processing of messages can happen in parallel on various channels.
* Can be used for any system to connect to other systems that requires enterprise level messaging (website activity tracking, operational metrics, stream processing etc)
* High availability of events(recoverable in case of failures)

 The Flume head start on HDFS integration has been really closed on by Kafka via the Confluent Kafka connectors which are professional integration components with the Hadoop ecosystem.

The one big advantage is that Kafka is being a pull system, i.e. Kafka provides back pressure to prevent overflowing consumers, by persistently storing the incoming messages until they “expire” configurable days later, so that late consumers can pick the messages up even rewind a few times, at their own pace - while Flume is a push system which implies data loss when consumers can’t keep up.

This means that Kafka will store some days or weeks or traffic, that would be able to be reprocessed any number of times, by any number of consumer groups, but most importantly, the create rate of those events will not overload the databases or the processes trying to get data into databases.

In pull systems, the consumers dictate the pace. With Kafka, you have push from producer to broker and pull from broker to consumer (which writes in storage), so both event producers and databases are moving in their own pace, while the overflow events are in the Kafka "persistent buffer".

Kafka has a lot of options for guarantee levels on both producing and consuming, with their tradeoffs.

Kafka is general purpose publish subscribe or queue system, and can mesh with any producer or consumer (including those sinking in Hadoop, Cassandra). Flume may be more dedicated to those scenarios, making Kafka a more reusable infrastructure piece. Much companies use Kafka (or AWS Kinesis, which is the same), as their message bus, or middleware of their MoM.

1. Kafka provides back pressure to prevent overflowing a broker. I don't believe Flume, Flume NG supports the same. Kafka brokers default to storing events for two weeks on disk.  
  
4. To get data out of Flume, you use a sink, which writes to your target store (HDFS, HBase, Cassandra etc). Flume will re-try connections to your sinks if they are offline. Because Flume pushes data, you have to do some interesting work to sink data to two data stores.. which requires replicating your channels, one for each sink. Start with figure 1: [Architecture of Flume NG : Apache Flume](http://blogs.apache.org/flume/entry/flume_ng_architecture)  
  
With Kafka you pull data, so each consumer has and manages it's own read pointer. This allows a large number of consumers of each Kafka queue, that pull data at their own pace.  
  
With this, you could deliver your event streams to HBase, Cassandra, Storm, Hadoop, RDBMS all in parallel.  
  
5. With Kafka 0.8+ you get replication of your event data. If you lose a broker node, others will take up the slack to delivery your events without loss.  
With Flume & FlumeNG, and a File channel, if you loose a broker node you will lose access to those events until you recover that disk. The database channel with Flume is reported too slow for any production use cases at volume.  
  
To me, **event availability is** **a huge consideration** in designing an real-time architecture that will be always on.  
  
Additional considerations:  
  
6. For enterprises, they very often want commercial support and this is an important consideration.  Flume is supported by a number of Hadoop distribution providers.  
  
7. Flume supports some content based event routing.  
8. Flume has a number of pre-built collectors. Kafka provides just the messaging.  
  
Both systems support high volumes of data.  
It often boils down to choosing either:   
a) Commercially supported out of the box offering sinking data to HDFS vs.   
b) Multi-consumer ultra-high availability messaging system.

Flume’s main use-case is to ingest data into Hadoop. It is tightly integrated with Hadoop’s monitoring system, file system, file formats, and utilities such a Morphlines. A lot of the Flume development effort goes into maintaining compatibility with Hadoop. Sure, Flume’s design of sources, sinks and channels mean that it can be used to move data between other systems flexibly, but the important feature is its Hadoop integration.

Kafka’s main use-case is a distributed publish-subscribe messaging system. Most of the development effort is involved with allowing subscribers to read exactly the messages they are interested in, and in making sure the distributed system is scalable and reliable under many different conditions. It was not written to stream data specifically for Hadoop, and using it to read and write data to Hadoop is significantly more challenging than it is in Flume.

To summarize:  
Use Flume if you have an non-relational data sources such as log files that you want to stream into Hadoop.  
Use Kafka if you need a highly reliable and scalable enterprise messaging system to connect many multiple systems, one of which is Hadoop.

