**Explain the need of Flume.**

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms.

Flume lets Hadoop users make the most of valuable log data. Specifically, Flume allows users to:

* Stream data from multiple sources into Hadoop for analysis
* Collect high-volume Web logs in real time
* Insulate themselves from transient spikes when the rate of incoming data exceeds the rate at which data can be written to the destination
* Guarantee data delivery
* Scale horizontally to handle additional data volume

**Explain the working of Flume and its components in brief.**

Components of apache flume -

* Event – a singular unit of data that is transported by Flume (typically a single log entry
* Source – the entity through which data enters into Flume. Sources either actively poll for data or passively wait for data to be delivered to them. A variety of sources allow data to be collected, such as log4j logs and syslogs.
* Sink – the entity that delivers the data to the destination. A variety of sinks allow data to be streamed to a range of destinations. One example is the HDFS sink that writes events to HDFS.
* Channel – the conduit between the Source and the Sink. Sources ingest events into the channel and the sinks drain the channel.
* Agent – any physical Java virtual machine running Flume. It is a collection of sources, sinks and channels.
* Client – produces and transmits the Event to the Source operating within the Agent

A flow in Flume starts from the Client (Web Server). The Client transmits the event to a Source operating within the Agent. The Source receiving this event then delivers it to one or more Channels. These Channels are drained by one or more Sinks operating within the same Agent. Channels allow decoupling of ingestion rate from drain rate using the familiar producer-consumer model of data exchange. When spikes in client side activity cause data to be generated faster than what the provisioned capacity on the destination can handle, the channel size increases. This allows sources to continue normal operation for the duration of the spike.