**##What is meant by Flume NG ?**

To solve certain known issues and limitations, Flume requires a refactoring of some core classes and systems. This bug is a parent issue to track the development of a "Flume NG" - a poorly named, but necessary refactoring. Subtasks should be added to track individual systems and components.

For reference, we refer to the code branch flume-728 (named for the refactoring JIRA) as "Flume NG."

At a high-level, Flume NG uses a single-hop message delivery guarantee semantics to provide end-to-end reliability for the system. To accomplish this, certain new concepts have been incorporated into its design, while certain other existing concepts have been either redefined, reused or dropped completely.

Concepts that Flume NG introduces, redefines, or reuses from earlier implementation:

* Event: A byte payload with optional string headers that represent the unit of data that Flume can transport from it’s point of origination to it’s final destination.
* Flow: Movement of events from the point of origin to their final destination is considered a data flow, or simply flow. This is not a rigorous definition and is used only at a high level for description purposes.
* Client: An interface implementation that operates at the point of origin of events and delivers them to a Flume agent. Clients typically operate in the process space of the application they are consuming data from. For example, Flume Log4j Appender is a client.
* Agent: An independent process that hosts flume components such as sources, channels and sinks, and thus has the ability to receive, store and forward events to their next-hop destination.
* Source: An interface implementation that can consume events delivered to it via a specific mechanism. For example, an Avro source is a source implementation that can be used to receive Avro events from clients or other agents in the flow. When a source receives an event, it hands it over to one or more channels.
* Channel: A transient store for events, where events are delivered to the channel via sources operating within the agent. An event put in a channel stays in that channel until a sink removes it for further transport. An example of channel is the JDBC channel that uses a file-system backed embedded database to persist the events until they are removed by a sink. Channels play an important role in ensuring durability of the flows.
* Sink: An interface implementation that can remove events from a channel and transmit them to the next agent in the flow, or to the event’s final destination. Sinks that transmit the event to it’s final destination are also known as terminal sinks. The Flume HDFS sink is an example of a terminal sink. Whereas the Flume Avro sink is an example of a regular sink that can transmit messages to other agents that are running an Avro source.

**##Can Flume provides 100 % reliability to the data flow?**

Flume emphasizes fault-tolerance as a core design principle and keeps running and collecting data even when many components have failed.

Flume can guarantee that all data received by an agent node will eventually make it to the collector at the end of its flow as long as the agent node keeps running. That is, data can be reliably delivered to its eventual destination.However, reliable delivery can be very resource intensive and is often a stronger guarantee than some data sources require. Therefore, Flume allows the user to specify, on a per-flow basis, the level of reliability required. There are three supported reliability levels:

* End-to-end
* Store on failure
* Best effort

Although Flume is extremely tolerant to machine, network, and software failures, there is **never any such thing as 100% reliability.** If all the machines in a Flume installation were irrevocably destroyed in some terrible data center incident, all copies of Flume’s data would be lost and there would be no way to recover them. Therefore all of Flume’s reliability levels make guarantees about data delivery until some maximum number of failures have occurred.

**##Can Flume can distributes data to multiple destinations?**

No , 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.** But at the end theres only one centralized storage (Hbase , HDFS etc.).

**##Explain about the different channel types in Flume. And which channel type is faster?**

Two recommended channels for flume are the file channel and the memory channel. The file channel is a durable channel, as it persists all events that are stored in it to disk. So, even if the Java virtual machine is killed, or the operating system crashes or reboots, events that were not successfully transferred to the next agent in the pipeline will still be there when the Flume agent is restarted.

The memory channel is a volatile channel, as it buffers events in memory only: if the Java process dies, any events stored in the memory channel are lost. Naturally, the memory channel also exhibits very low put/take latencies compared to the file channel, even for a batch size of 1.

Since the number of events that can be stored is limited by available RAM, its ability to buffer events in the case of temporary downstream failure is quite limited.

The file channel, on the other hand, has far superior buffering capability due to utilizing cheap, abundant hard disk space.