Flume

ixAT Solutions – ixatsolutions@gmail.com

Apache Flume, what is it?

- It is a distributed data collection service that gets flows of data (like logs) from their source and aggregates them to where they have to be processed.
- Goals: reliability, scalability, extensibility, manageability.

Challenges

- Physically Distributed Data Sources

  Across servers, Data Centers, Geographies, Organization Boundaries,Technologies...

- Continuous Data Production

  Every two days now we create as much information as we did from the dawn of civilization up until 2003"

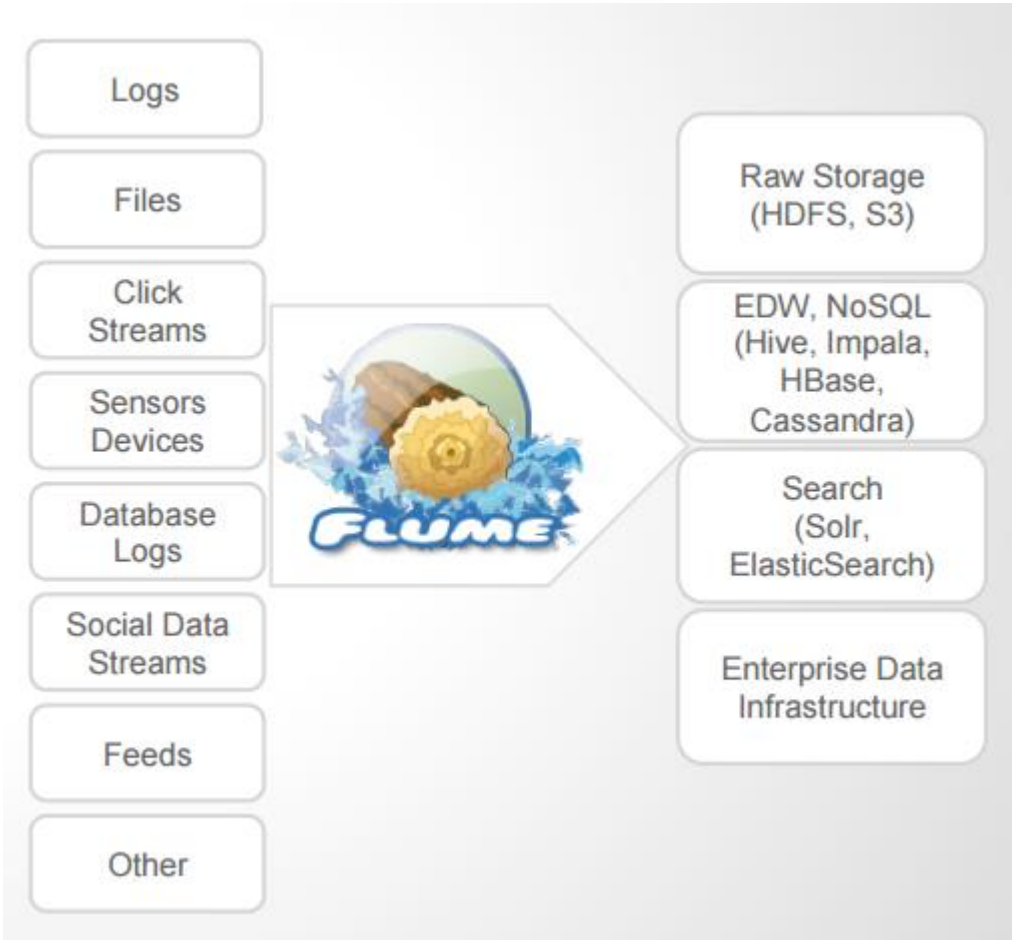
  - Eric Schmidt, 2010

  ● Weather ● Traffic ● Automobiles ● Trains ● Airplanes ● Geological/Seismic ● Oceanographic ● Smart Phones ● Health Accessories ● Medical Devices ● Home Automation ● Digital Cameras ● Social Media ● Geolocation ● Shop Floor Sensors ● Network Activity ● Industry Appliances ● Security/Surveillance ● Server Workloads ● Digital Telephony ● Bio-simulations...

- Continuously evolving data structures and semantics...

  CSV, TSV, JSON, XML, Binary, Custom Formats...

Intro

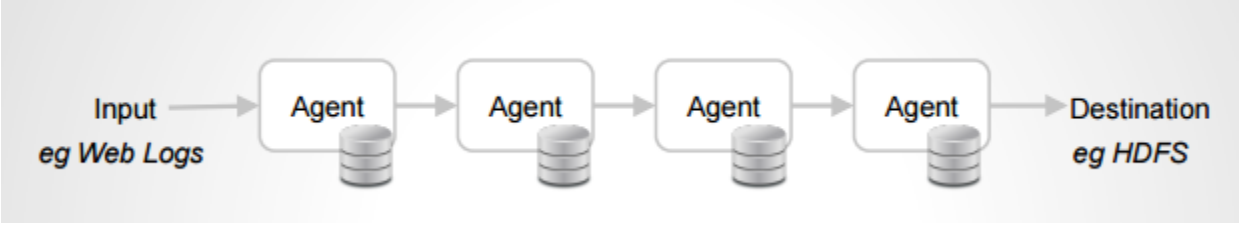


History

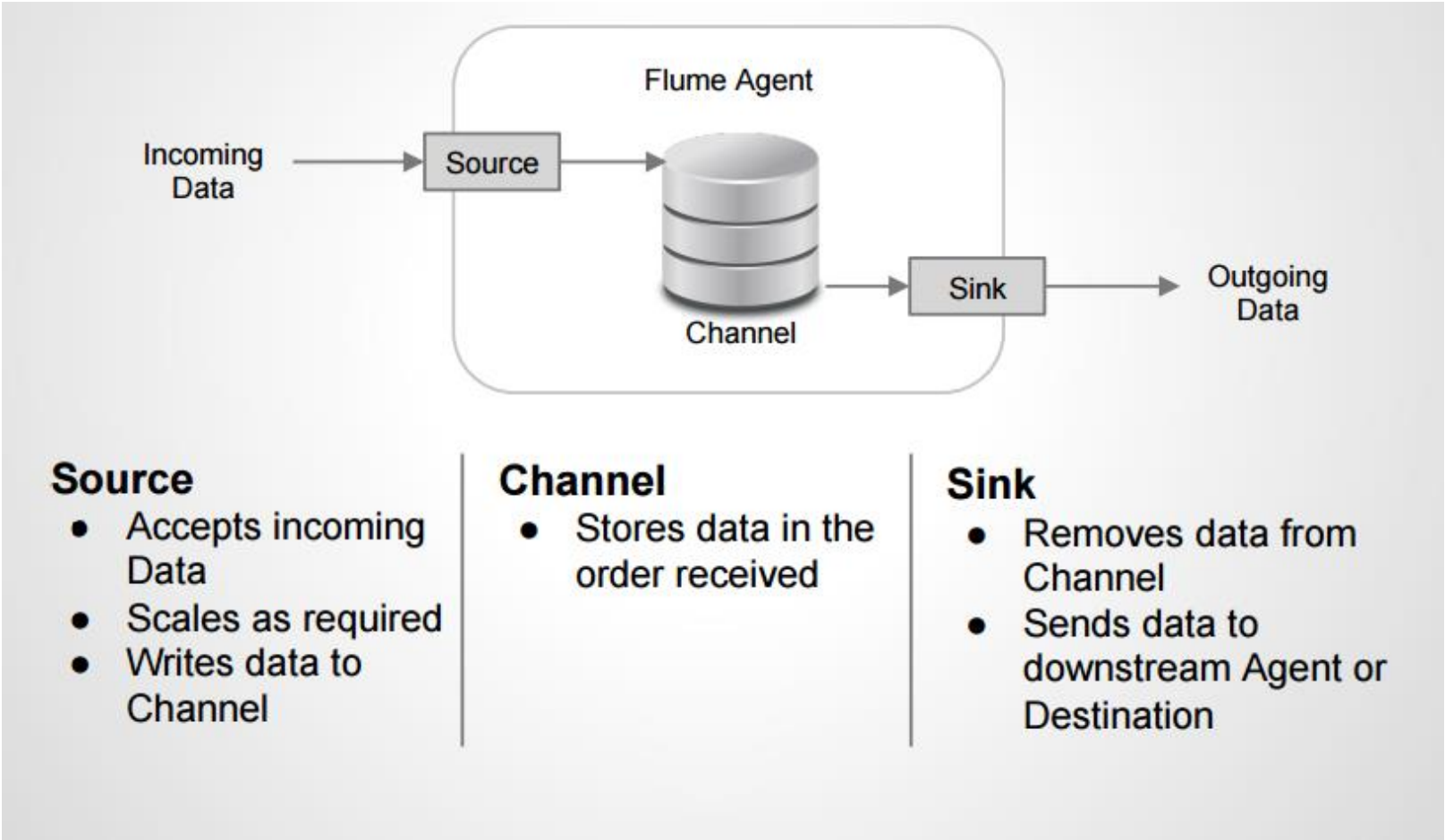
- Originally designed to be a log aggregation system by Cloudera Engineers
- Evolved to handle any type of streaming event data
- Low-cost of installation, operation and maintenance
- Highly customizable and extendable

Architecture

- Distributed Pipeline Architecture
- Optimized for commonly used data sources and destinations
- Built in support for contextual routing
- Fully customizable and extendable



Architecture



Source
- Accepts incoming Data
- Scales as required
- Writes data to Channel

Channel
- Stores data in the order received

Sink
- Removes data from Channel
- Sends data to downstream Agent or Destination

Core Concepts: Event

An Event is the fundamental unit of data transported by Flume from its point of origination to its final destination. Event is a byte array payload accompanied by optional headers.

- Payload is opaque to Flume
- Headers are specified as an unordered collection of string key-value pairs, with keys being unique across the collection
- Headers can be used for contextual routing

Core Concepts: Client

An entity that generates events and sends them to one or more Agents.

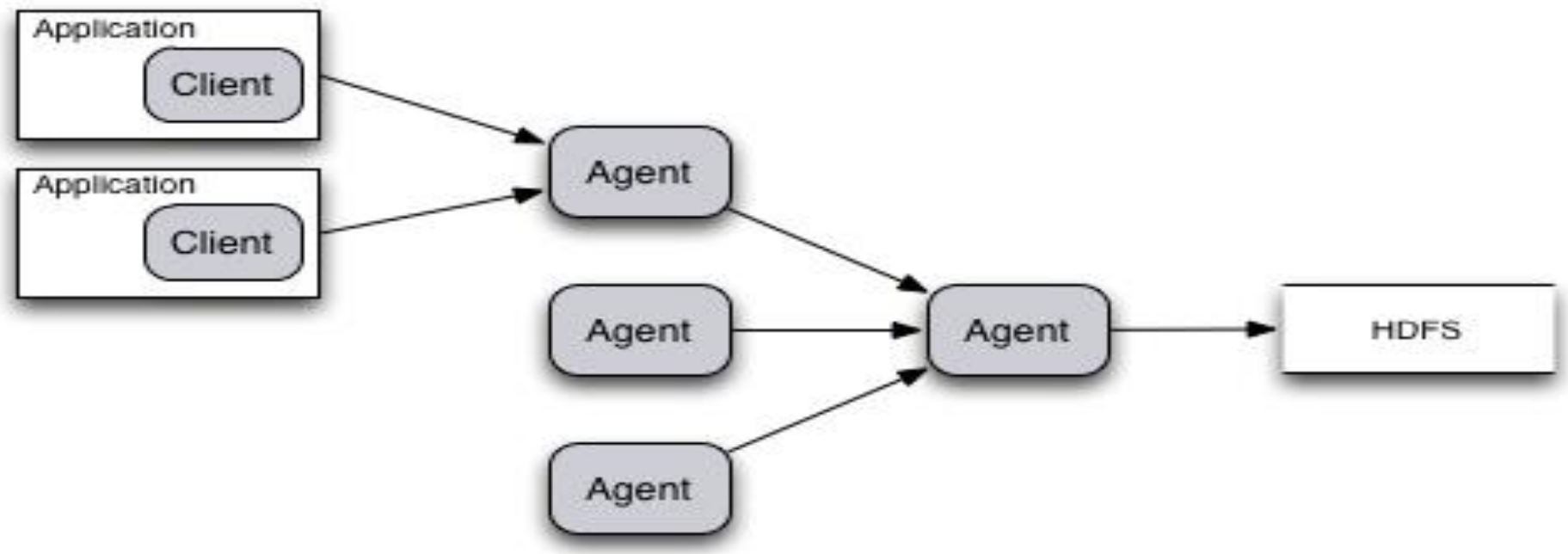
- Example
  - Flume log4j Appender
  - Custom Client using Client SDK (`org.apache.flume.api`)
- Decouples Flume from the system where event data is consumed from
- Not needed in all cases

Core Concepts: Agent

A container for hosting Sources, Channels, Sinks and other components that enable the transportation of events from one place to another.

- Fundamental part of a Flume *flow*
- Provides Configuration, Life-Cycle Management, and Monitoring Support for hosted components

Typical Aggregation Flow



[Client]$^{+}$ → Agent [→ Agent]* → Destination

Core Concepts: Source

An active component that receives events from a specialized location or mechanism and places it on one or Channels.

- Different Source types:
  - Specialized sources for integrating with well-known systems. Example: Syslog, Netcat
  - Auto-Generating Sources: Exec, SEQ
  - IPC sources for Agent-to-Agent communication: Avro
- Require at least one channel to function

Core Concepts: Channel

A passive component that buffers the incoming events until they are drained by Sinks.

- Different Channels offer different levels of persistence:
  - Memory Channel: volatile
  - File Channel: backed by WAL implementation
  - JDBC Channel: backed by embedded Database
- Channels are fully transactional
- Provide weak ordering guarantees
- Can work with any number of Sources and Sinks.

Core Concepts: Sink

An active component that removes events from a Channel and transmits them to their next hop destination.

- Different types of Sinks:
  - Terminal sinks that deposit events to their final destination. For example: HDFS, HBase
  - Auto-Consuming sinks. For example: Null Sink
  - IPC sink for Agent-to-Agent communication: Avro
- Require exactly one channel to function

Configuration

agent1.properties:

```
# Active components
agent1.sources = src1
agent1.channels = ch1
agent1.sinks = sink1

# Define and configure src1
agent1.sources.src1.type = netcat
agent1.sources.src1.channels = ch1
agent1.sources.src1.bind = 127.0.0.1
agent1.sources.src1.port = 10112

# Define and configure sink1
agent1.sinks.sink1.type = logger
agent1.sinks.sink1.channel = ch1

# Define and configure ch1
agent1.channels.ch1.type = memory
```

Configuration

- A configuration file can contain configuration information for many Agents
- Only the portion of configuration associated with the name of the Agent will be loaded
- Components defined in the configuration but not in the active list will be ignored
- Components that are misconfigured will be ignored
- Agent automatically reloads configuration if it changes on disk

An Example

- Download flume
- Set path etc…
- Create a config file (say flume1.conf in ~/flumetests/ dir) with the below content –

```
myagent.sources = filesrc
myagent.sinks = logsink
myagent.channels = c1

# Describe/configure the source
myagent.sources.filesrc.type = spooldir
myagent.sources.filesrc.spoolDir = /home/hdtester

# Describe the sink
myagent.sinks.logsink.type = logger

# Use a channel which buffers events in file
myagent.channels.c1.type = file

# Bind the source and sink to the channel
myagent.sources.filesrc.channels = c1
myagent.sinks.logsink.channel = c1
```

- Run the agent

```
flume-ng agent -f flume1.conf -n myagent -Dflume.root.logger=INFO,console
```

One more example with HDFS + Console Sinks (Fan out)

File Name flume2.conf

```
myagent.sources = dirsrc
myagent.sinks = logsink hdfssink
myagent.channels = c1 c2

# Describe/configure the source
myagent.sources.dirsrc.type = spooldir
myagent.sources.dirsrc.spoolDir = /home/hdtester

# Describe the sink
myagent.sinks.logsink.type = logger

myagent.sinks.hdfssink.type = hdfs
myagent.sinks.hdfssink.hdfs.path = /flumecreated
myagent.sinks.hdfssink.hdfs.fileType = DataStream


# Use a channel which buffers events in file
myagent.channels.c1.type = file
myagent.channels.c1.checkpointDir = ./.flume/file-channel1/checkpoint
myagent.channels.c1.dataDirs = ./.flume/file-channel1/data

myagent.channels.c2.type = file
myagent.channels.c2.checkpointDir = ./.flume/file-channel2/checkpoint ##without this there would be a lock issue with channel1
myagent.channels.c2.dataDirs = ./.flume/file-channel2/data ##without this there would be a lock issue with channel1


# Bind the source and sink to the channel
myagent.sources.dirsrc.channels = c1 c2
myagent.sinks.logsink.channel = c1
myagent.sinks.hdfssink.channel = c2
```

```
flume-ng agent -f flume2.conf -n myagent -Dflume.root.logger=INFO,console
```

An example with 2 sources, 1 sink

File Name flume3.conf

```
myagent.sources = source1 source2
myagent.sinks = sink1
myagent.channels = channel1

myagent.sources.source1.type = exec
myagent.sources.source1.command = tail -f FILE1
myagent.sources.source1.channels = channel1

myagent.sources.source2.type = exec
myagent.sources.source2.command = tail -f FILE2
myagent.sources.source2.channels = channel1

myagent.channels.channel1.type = memory

myagent.sinks.sink1.type = file_roll
myagent.sinks.sink1.batchSize = 2
myagent.sinks.sink1.channel = channel1
myagent.sinks.sink1.sink.directory = /home/hdtester/flumetests
myagent.sinks.sink1.sink.rollInterval = 30
myagent.sinks.sink1.sink.serializer = TEXT
```

In another window, simulate writing into two files, named FILE1 and FILE2

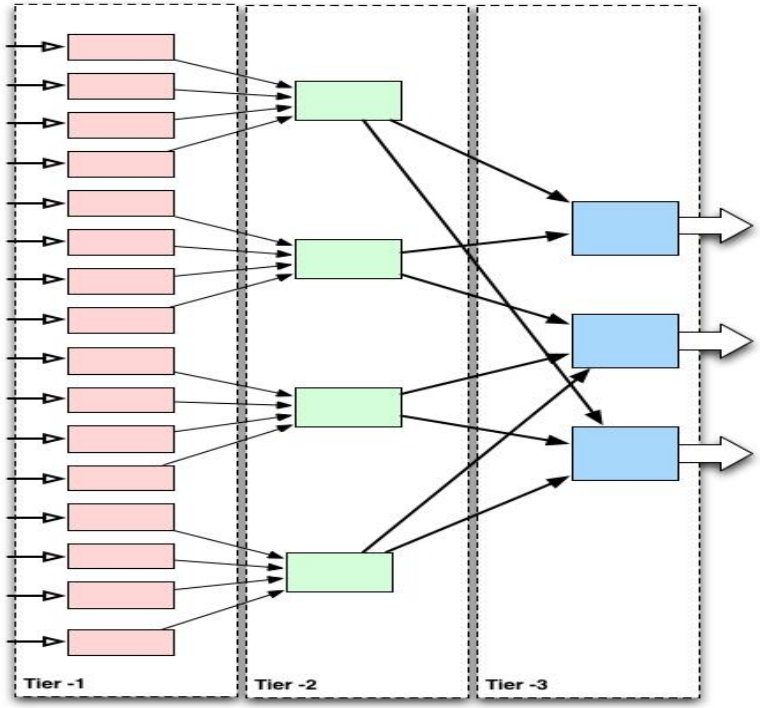
```
while true; do   echo "File 1 `date`" >> FILE1;
echo "File 2 `date`" >> FILE2;  sleep `shuf -i 1-5 -n
1`;done
```

```
flume-ng agent -f flume3.conf -n myagent -Dflume.root.logger=INFO,console
```

Add a HDFS sink to previous example

```
...
myagent.sinks = sink1 sink2
myagent.channels = channel1 channel2
...
myagent.channels.channel2.type = memory
...
myagent.sinks.sink2.type = hdfs
myagent.sinks.sink2.hdfs.path = /flumecreated/h1
myagent.sinks.sink2.hdfs.fileType = DataStream
myagent.sinks.sink2.channel = channel2
```

Configuration



Typical Deployment

- All agents in a specific tier could be given the same name
- One configuration file with entries for three agents can be used throughout