FLUME INSIGHT

WELCOME

"We Will Open the World of Creativity for you !"

Presented by
Siva Kumar Bhuchipalli

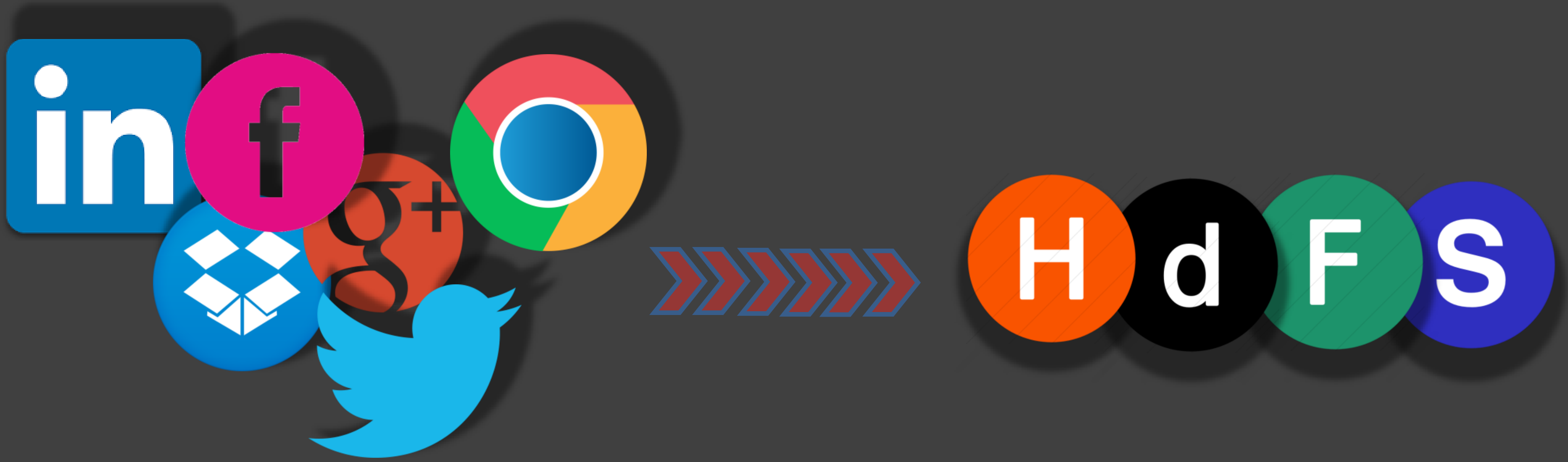
Contents

- Overview of Flume
- Flume Architecture

Flume

Flume is a highly reliable, distributed and configurable streaming data collection tool.

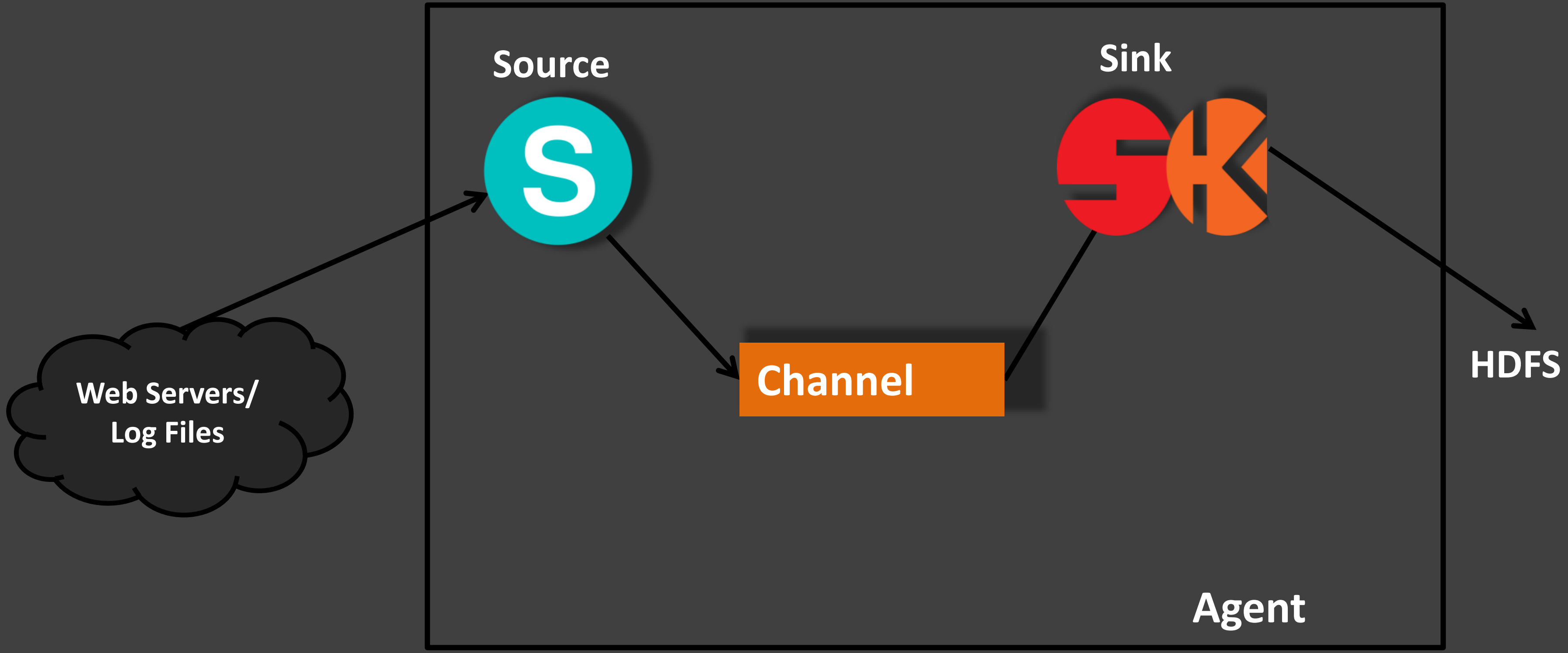
Flume can transport log files across a large number of hosts into HDFS.



Flume Features

- Flume collects data efficiently, aggregates and moves large amounts of log data from many different sources to a centralized data store.
- Simple and flexible architecture, and provides a streaming of data flows and leverages data movement from multiple machines in within an enterprise into Hadoop.
- Flume is not restricted to log data aggregation and it can transport massive quantities of event data including but not limited to network traffic data, social-media-generated data, email messages and pretty much any data source possible.
- Built-in support for several Sources and destination platforms to integrate with.
- Support for multi-hop flows where events travel through multiple agents before reaching the final destination. fan-in and fan-outflows, contextual routing and backup routes for failed hops are also allowed.

Flume Architecture



Source

- Flume source is configured within an agent and it listens for events from an external source (eg: web server) it reads data, translates events, and handles failure situations.
- But source doesn’t know how to store the event. So, after receiving enough data to produce a Flume event, it sends events to the channel to which the source is connected.
- The external source sends events to Flume in a format that is recognized by the target Flume source.
  - ❑ Spooling Directory Source
  - ❑ Exec Source
  - ❑ Netcat Source
  - ❑ HTTP Source
  - ❑ Twitter Source
  - ❑ Avro Source

Channel

- Channels are communication bridges between sources and sinks within an agent. Once a Flume source receives an event, it stores it into one or more channels. The channel is a passive store that keeps the event until it's consumed by a Flume sink.
- Memory channel stores the events from in an in-memory queue and from there events will be accessed by sink. Because of software or hardware failures, if the agent process dies in the middle, then all the events currently in the memory channel are lost forever.
- The File channel is another example – it is backed by the local file system. Unlike memory channel, file channel writes the contents to a file on the file system that is deleted only after successful delivery to the sink.
  - ❑ Memory Channel
  - ❑ File Channel
  - ❑ Spillable Memory Channel

Note:

The memory channel is the fastest but has the risk of data loss. The file channels are typically much slower but effectively provide guaranteed delivery to the sink.

Channel

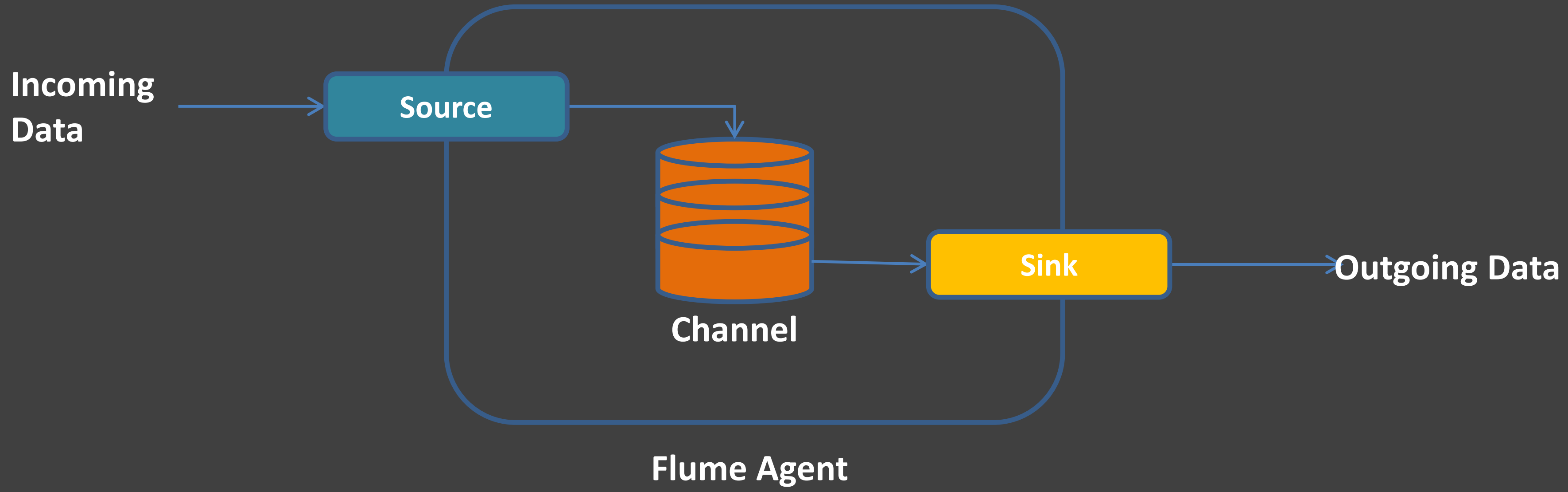
Sink

- Sink removes the event from the channel and puts it into an external repository like HDFS or forwards it to the Flume source of the next Flume agent in the flow. The source and sink within the given agent run asynchronously with the events staged in the channel.
    - HDFS Sink
    - Logger Sink
    - File Roll Sink
    - HBaseSink
    - MorphlineSolrSink
    - ElasticSearchSink
    - Avro Sink

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



Anatomy of flume Agent

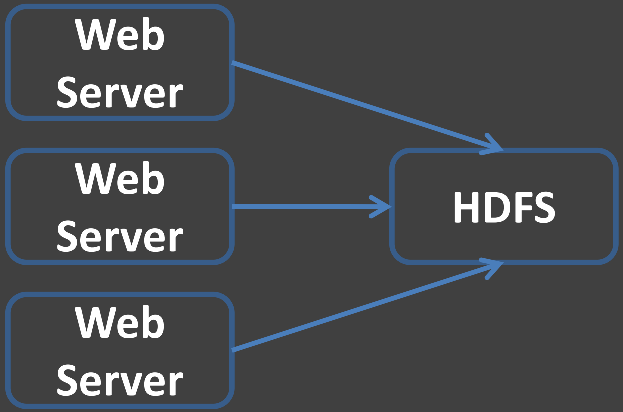
Aggregation Use Cases…..

Starting Out Simple

- You would like to move your web server logs to HDFS
- Let's assume there are only 3 web servers at the time of launch
- Ad-hoc solution will likely suffice!

Challenges

- How do you manage your output paths on HDFS?
- How do you maintain your client code in face of changing environment as well as requirements?



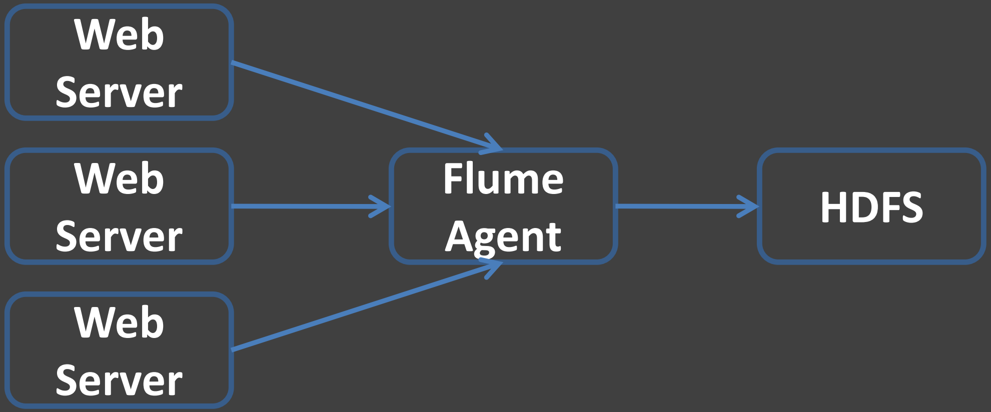
Adding Simple Flume Agent

Advantages

- Quick offload of logs from Web Server machines
- Insulation from HDFS downtime
- Better Network utilization

Challenges

- What if the Flume node goes down?
- Can one Flume node accommodate all load from Web Servers?



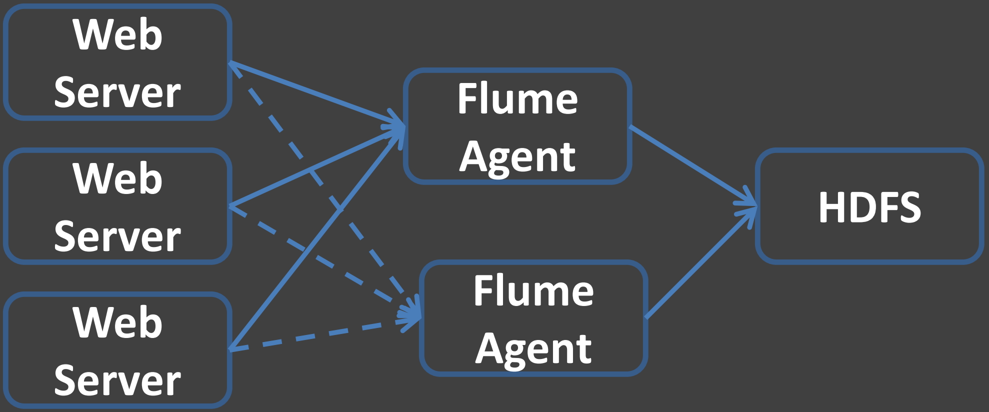
Adding Two Flume Agent

Advantages

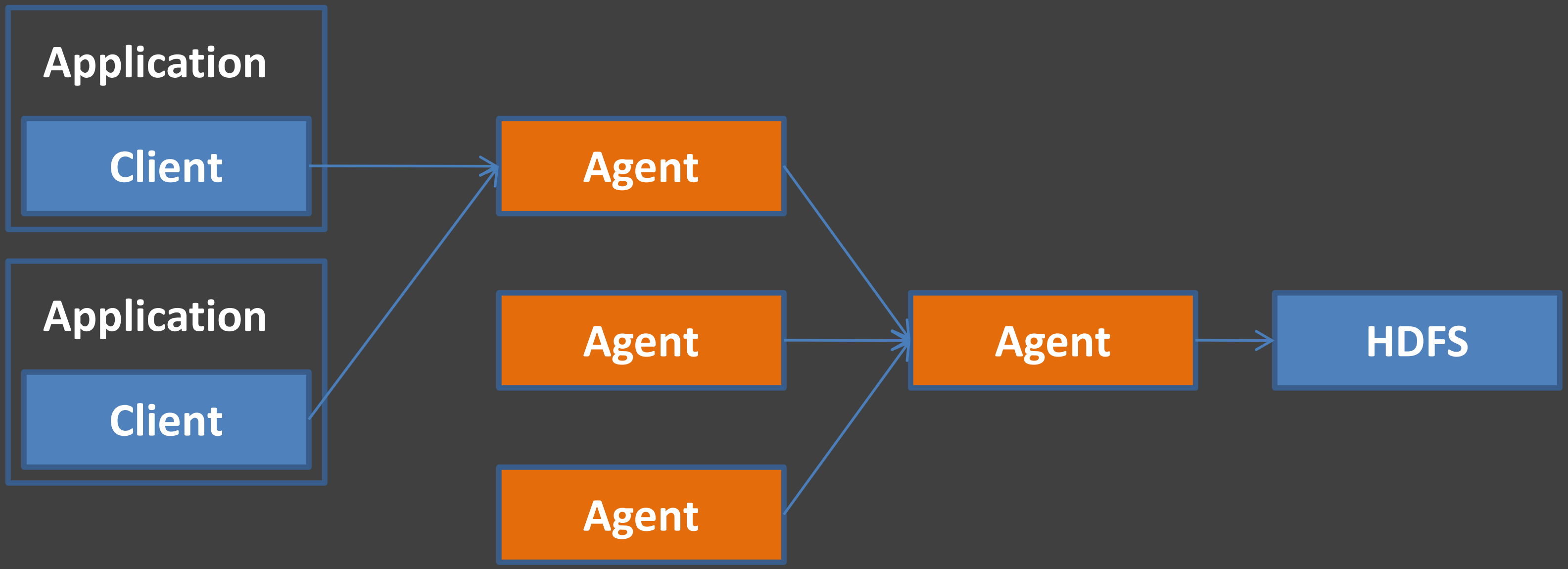
- Redundancy and Availability
- Better handling of downstream failures
- Automatic load balancing and failover

Challenges

- What happens when new Web Servers are added?
- Can two Flume Agents keep up with all the load from more Web Servers?

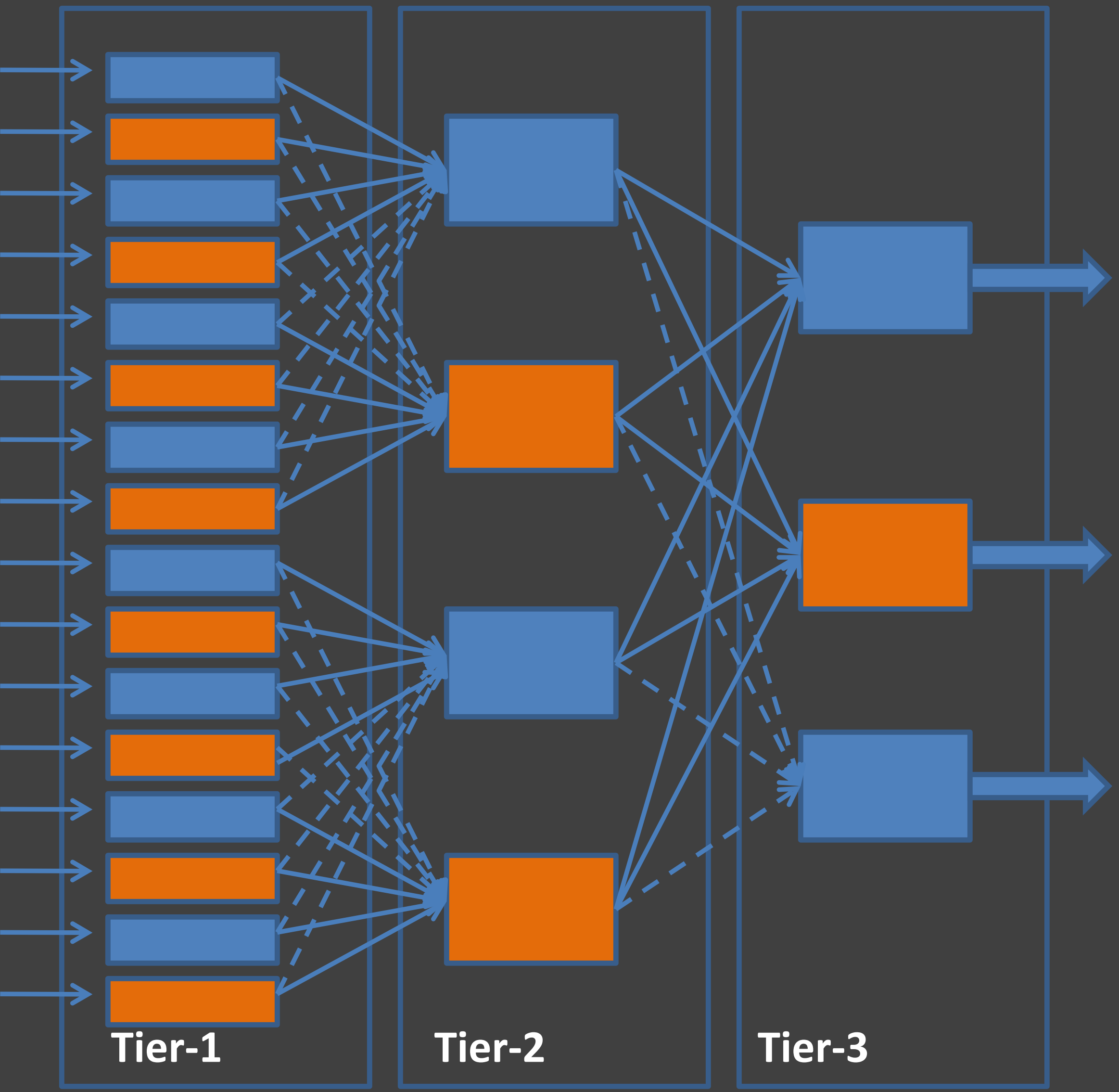


Typical Converging Flow



[Client]→Agent[→Agent]*→Destination 24

Handling Sever Farm



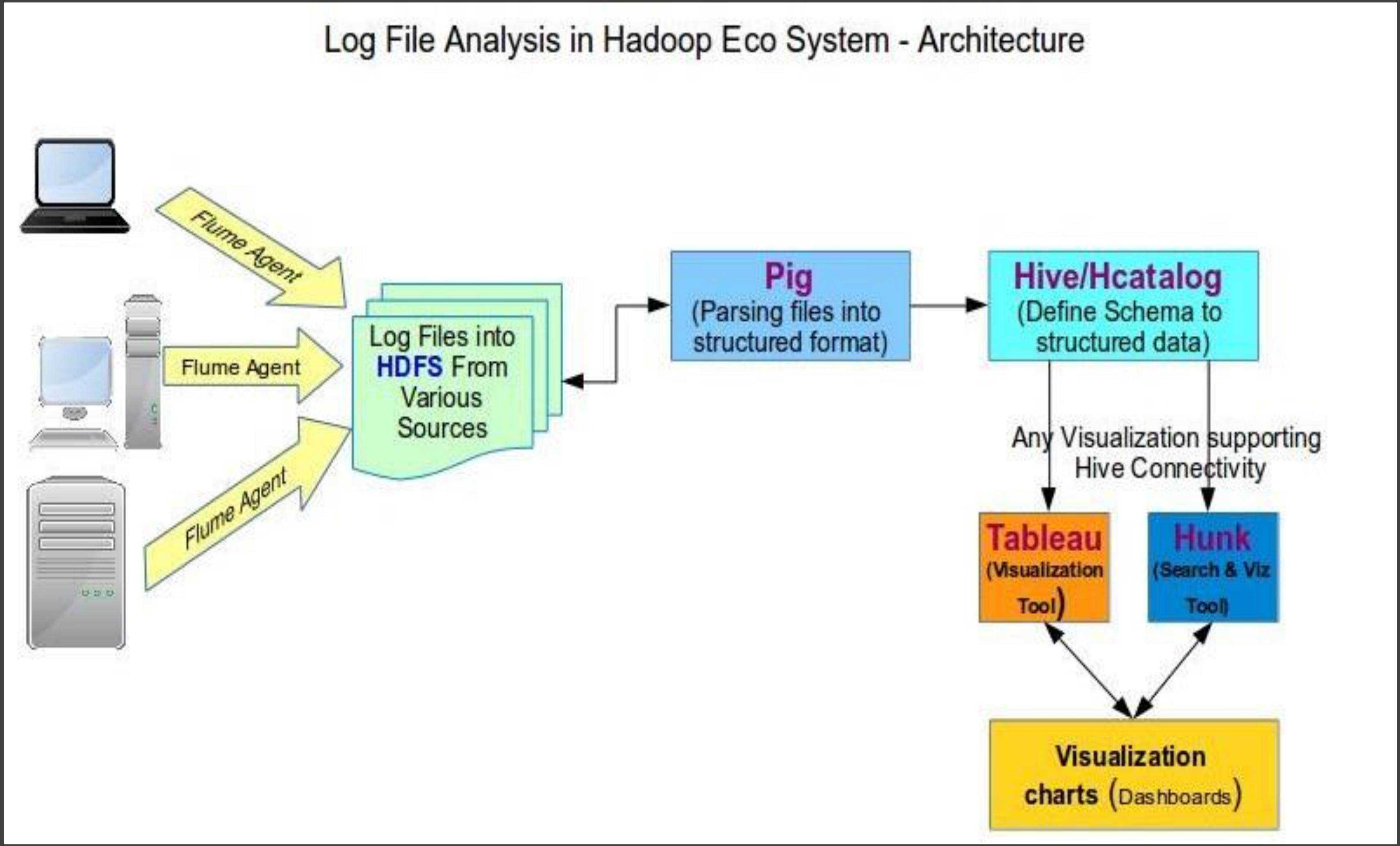
A Converging Flow

❖Traffic is aggregated by Tier-2 and Tier-3 before being put into destination system

❖Closer a tier is to the destination, larger the batch size it delivers Downstream

❖Optimized handling of destination systems

Log Analysis in Hadoop

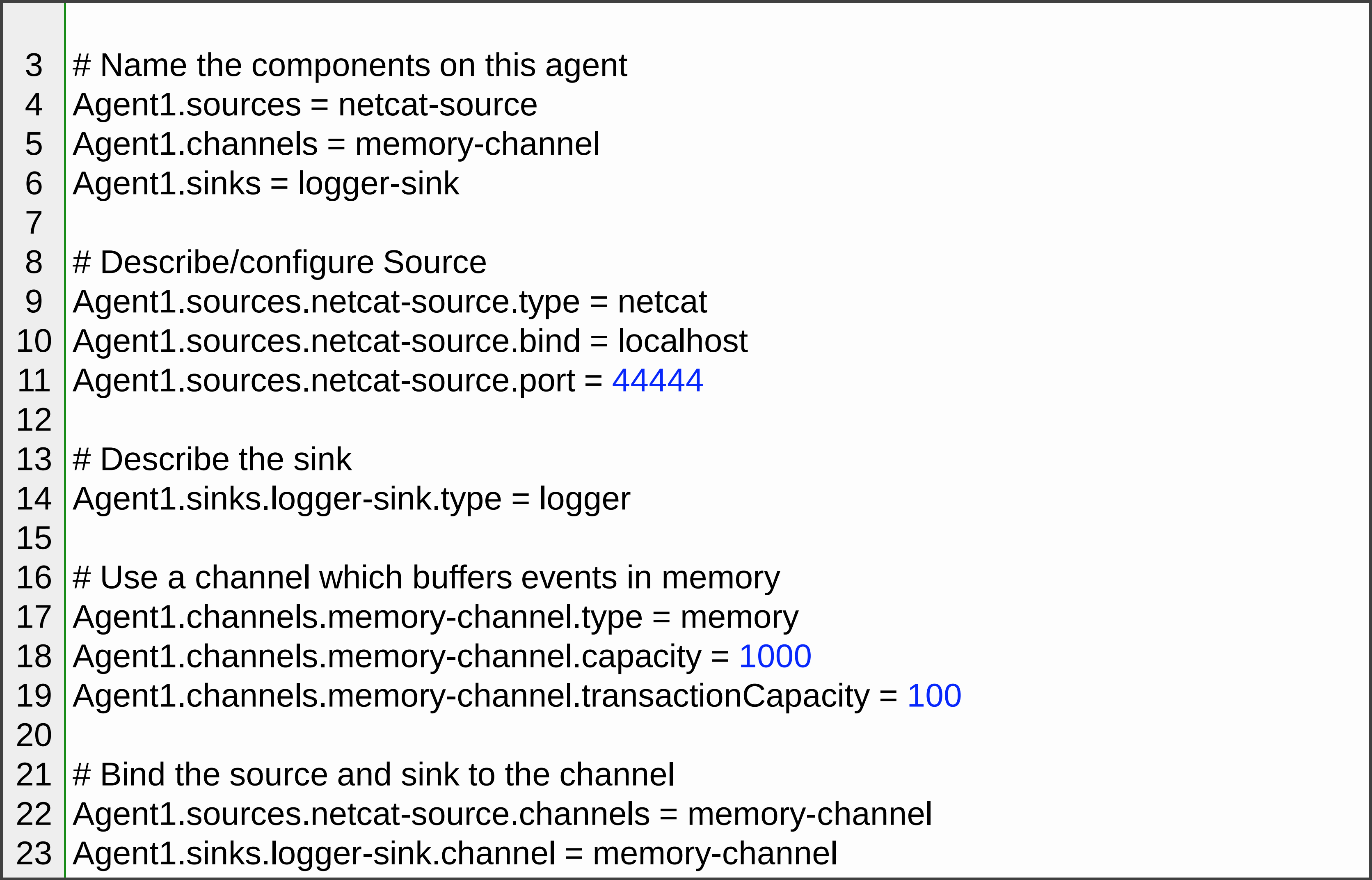

Log File Analysis in Hadoop Eco System - Architecture
Flume Agent
Flume Agent
Flume Agent
Log Files into HDFS From Various Sources
Pig
(Parsing files into structured format)
Hive/Hcatalog
(Define Schema to structured data)
Any Visualization supporting Hive Connectivity
Tableau
(Visualization Tool)
Hunk
(Search & Viz Tool)
Visualization charts (Dashboards)

Setup Of Flume Agent

❖Flume Agent Setup:

- ❑ Step 1 – Create flume.conf file
- ❑ Step 2 – Starting Flume Agent
- ❑ Step 3 – Feeding events to Flume Agent
- ❑ Step 4 – Verify the events at Agent Console
- ❑ Step 5 – Stop Flume Agent

Sample Flume.conf - NetcatSource, Log Sink and Memory Channel



```
# Name the components on this agent
Agent1.sources = netcat-source
Agent1.channels = memory-channel
Agent1.sinks = logger-sink

# Describe/configure Source
Agent1.sources.netcat-source.type = netcat
Agent1.sources.netcat-source.bind = localhost
Agent1.sources.netcat-source.port = 44444

# Describe the sink
Agent1.sinks.logger-sink.type = logger

# Use a channel which buffers events in memory
Agent1.channels.memory-channel.type = memory
Agent1.channels.memory-channel.capacity = 1000
Agent1.channels.memory-channel.transactionCapacity = 100

# Bind the source and sink to the channel
Agent1.sources.netcat-source.channels = memory-channel
Agent1.sinks.logger-sink.channel = memory-channel
```

Start Flume Agent

```
$ flume-ng agent --conf flume/conf/path --conf-file flume/conf/path/flume.conf --
name Agent1 -Dflume.root.logger=INFO,console
```

Feeding Events

```
$ curl telnet://localhost:44444
```

or

```
$ telnet localhost 44444
```

Agent – SpoolingDir Source, File Channel, HDFS Sink

```
agent1.sinks =  hdfs-sink1_1
agent1.sources = source1_1
agent1.channels = fileChannel1_1

agent1.sources.source1_1.type = spoolDir
agent1.sources.source1_1.spoolDir = /home/cloudera/spool
agent1.sources.source1_1.fileHeader = false
agent1.sources.source1_1.fileSuffix = .COMPLETED

agent1.sinks.hdfs-sink1_1.type = hdfs
agent1.sinks.hdfs-sink1_1.hdfs.path = hdfs://quickstart.cloudera:8020/flume_sink/%y-%m-%d/%H%M/
agent1.sinks.hdfs-sink1_1.hdfs.batchSize = 1000
agent1.sinks.hdfs-sink1_1.hdfs.rollSize = 268435456
agent1.sinks.hdfs-sink1_1.hdfs.rollInterval = 0
agent1.sinks.hdfs-sink1_1.hdfs.rollCount = 50000000
agent1.sinks.hdfs-sink1_1.hdfs.writeFormat=Text
agent1.sinks.hdfs-sink1_1.hdfs.useLocalTimeStamp = true
agent1.sinks.hdfs-sink1_1.hdfs.fileType = DataStream

agent1.sources.source1_1.channels = fileChannel1_1
agent1.sinks.hdfs-sink1_1.channel = fileChannel1_1

agent1.channels.fileChannel1_1.type = file
agent1.channels.fileChannel1_1.checkpointDir = /home/cloudera/flume/checkpoint/
agent1.channels.fileChannel1_1.dataDirs = /home/cloudera/flume/data/
```

HDFS Sink Properties

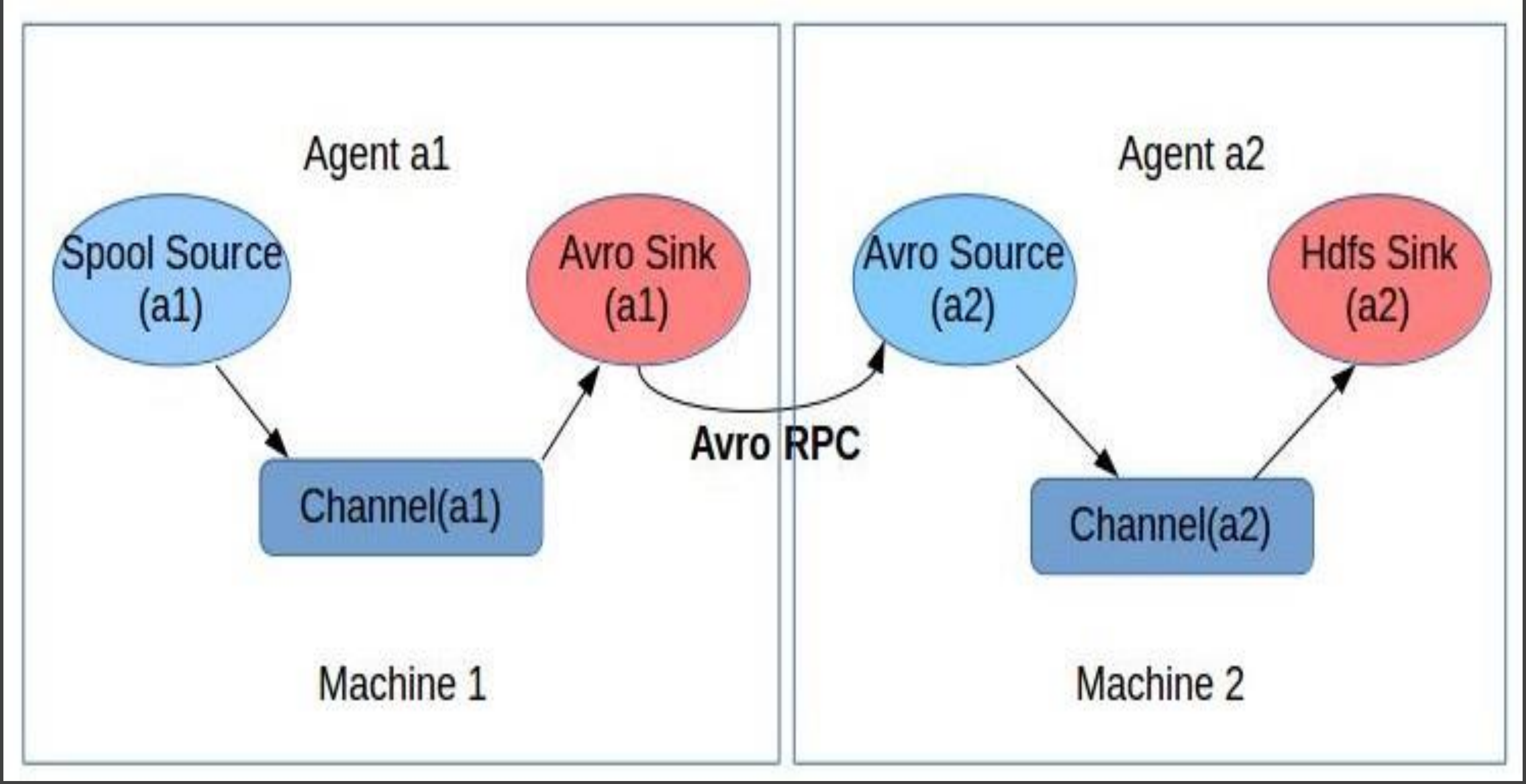
Property Name	Default	Description
type	–	The component type name, needs to be hdfs
hdfs.path	–	HDFS directory path (eg hdfs://namenode/flume/webdata/)
hdfs.filePrefix	FlumeData	Name prefixed to files created by Flume in hdfs directory
hdfs.fileSuffix	–	Suffix to append to file (eg .avro – NOTE: period is not automatically added)
hdfs.inUseSuffix	.tmp	Suffix that is used for temporal files that flume actively writes into
hdfs.rollInterval	30	Number of seconds to wait before rolling current file (0 = never roll based on time interval)
hdfs.rollSize	1024	File size to trigger roll, in bytes (0: never roll based on file size)
hdfs.rollCount	10	Number of events written to file before it rolled (0 = never roll based on number of events)
hdfs.batchSize	100	number of events written to file before it is flushed to HDFS
hdfs.codeC	–	Compression codec. one of following : gzip, bzip2, lzo, lzop, snappy
hdfs.fileType	SequenceFile	File format: currently SequenceFile , DataStream or CompressedStream (1)DataStream will not compress output file and please don't set codeC (2)CompressedStream requires set hdfs.codeC with an available codeC
hdfs.minBlockReplicas	–	Specify minimum number of replicas per HDFS block. If not specified, it comes from the default Hadoop config in the classpath.
serializer	TEXT	Other possible options include avro_event or the fully-qualified class name of an implementation of the EventSerializer.Builder interface.

Twitter Agent

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = FileChannel
TwitterAgent.sinks = HDFS

#TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource
TwitterAgent.sources.Twitter.channels = FileChannel
TwitterAgent.sources.Twitter.consumerKey = RUez5WLB8UxhgfhsdgfjVQTs
TwitterAgent.sources.Twitter.consumerSecret = gZN2xtwoRWeruB9ghsgssfk201Pt3tgEBRPWH6feML5oq6ho
TwitterAgent.sources.Twitter.accessToken = 56093598-ALvkeeLbhzQwBoDgO1298sfhj98nfa51BpgANUxTt
TwitterAgent.sources.Twitter.accessTokenSecret = OtfiCBSXjiApSSmZk70h31jkajjadkaXoAugnC9pSm
TwitterAgent.sources.Twitter.maxBatchSize = 50000
TwitterAgent.sources.Twitter.maxBatchDurationMillis = 100000

TwitterAgent.sinks.HDFS.channel = FileChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path = hdfs://localhost:9000/user/flume/tweets/
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 200000
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
TwitterAgent.sinks.HDFS.hdfs.rollCount = 2000000

TwitterAgent.channels.FileChannel.type = file
TwitterAgent.channels.FileChannel.checkpointDir = /var/log/flume/checkpoint/
TwitterAgent.channels.FileChannel.dataDirs = /var/log/flume/data/
```

http://hadooptutorial.info/twitter-data-analysis-using-hadoop-flume/

Multi Agent Flow



Multi Agent Flow

```
### Agent1 - Spooling Directory Source and File Channel, Avro Sink  ###
# Name the components on this agent
Agent1.sources = spooldir-source
Agent1.channels = file-channel
Agent1.sinks = avro-sink

# Describe/configure Source
Agent1.sources.spooldir-source.type = spooldir
Agent1.sources.spooldir-source.spoolDir = /home/user/testflume/spooldir

# Describe the sink
Agent1.sinks.avro-sink.type = avro
Agent1.sinks.avro-sink.hostname = 251.16.12.112   #IP Address masked here
Agent1.sinks.avro-sink.port = 11111

#Use a channel which buffers events in file
Agent1.channels.file-channel.type = file
Agent1.channels.file-channel.checkpointDir = /home/user/testflume/checkpoint/
Agent1.channels.file-channel.dataDirs = /home/user/testflume/data/

# Bind the source and sink to the channel
Agent1.sources.spooldir-source.channels = file-channel
Agent1.sinks.avro-sink.channel = file-channel
```

http://hadooptutorial.info/multi-agent-setup-in-flume/

Multi Agent Flow

```
### Agent2 - Avro Source and File Channel, Avro Sink  ###
# Name the components on this agent
Agent2.sources = avro-source
Agent2.channels = file-channel
Agent2.sinks = hdfs-sink

# Describe/configure Source
Agent2.sources.avro-source.type = avro
Agent2.sources.avro-source.hostname = 251.16.12.112
Agent2.sources.avro-source.port = 11111

# Describe the sink
Agent2.sinks.hdfs-sink.type = hdfs
Agent2.sinks.hdfs-sink.hdfs.path = hdfs://251.16.12.112:9000/flume/
Agent2.sinks.hdfs-sink.hdfs.rollInterval = 0
Agent2.sinks.hdfs-sink.hdfs.rollSize = 0
Agent2.sinks.hdfs-sink.hdfs.rollCount = 10000
Agent2.sinks.hdfs-sink.hdfs.fileType = DataStream

#Use a channel which buffers events in file
Agent2.channels.file-channel.type = file
Agent2.channels.file-channel.checkpointDir = /home/user/testflume/checkpoint/
Agent2.channels.file-channel.dataDirs = /home/user/testflume/data/

# Bind the source and sink to the channel
Agent2.sources.avro-source.channels = file-channel
Agent2.sinks.hdfs-sink.channel = file-channel
```

Some Highlights Of Flume

❖Flume is suitable for large volume data collection, especially when data is being produced in multiple locations

❖Once planned and sized appropriately, Flume will practically run itself without any operational intervention

❖Flume provides weak ordering guarantee, i.e., in the absence of failures the data will arrive in the order it was received in the Flume pipeline

❖Flume is a continuous service that will be monitoring arrival of input and ingests that into destination without our directions to start the process every time data arrives.

❖Flume has rich out-of-the box features such as contextual routing, and support for popular data sources and destination systems

QUESTIONS

THANK YOU