LEARN.
CONNECT.
ELEVATE.

YCBS 257 - Data at Scale (Winter 2019)

Instructor: Khaled Tannir

McGill | School of Continuing Studies

mcgill.ca
/continuingstudies

School of Continuing Studies

YCBS 257-256 / 257 - Data at Scale (BIG DATA)

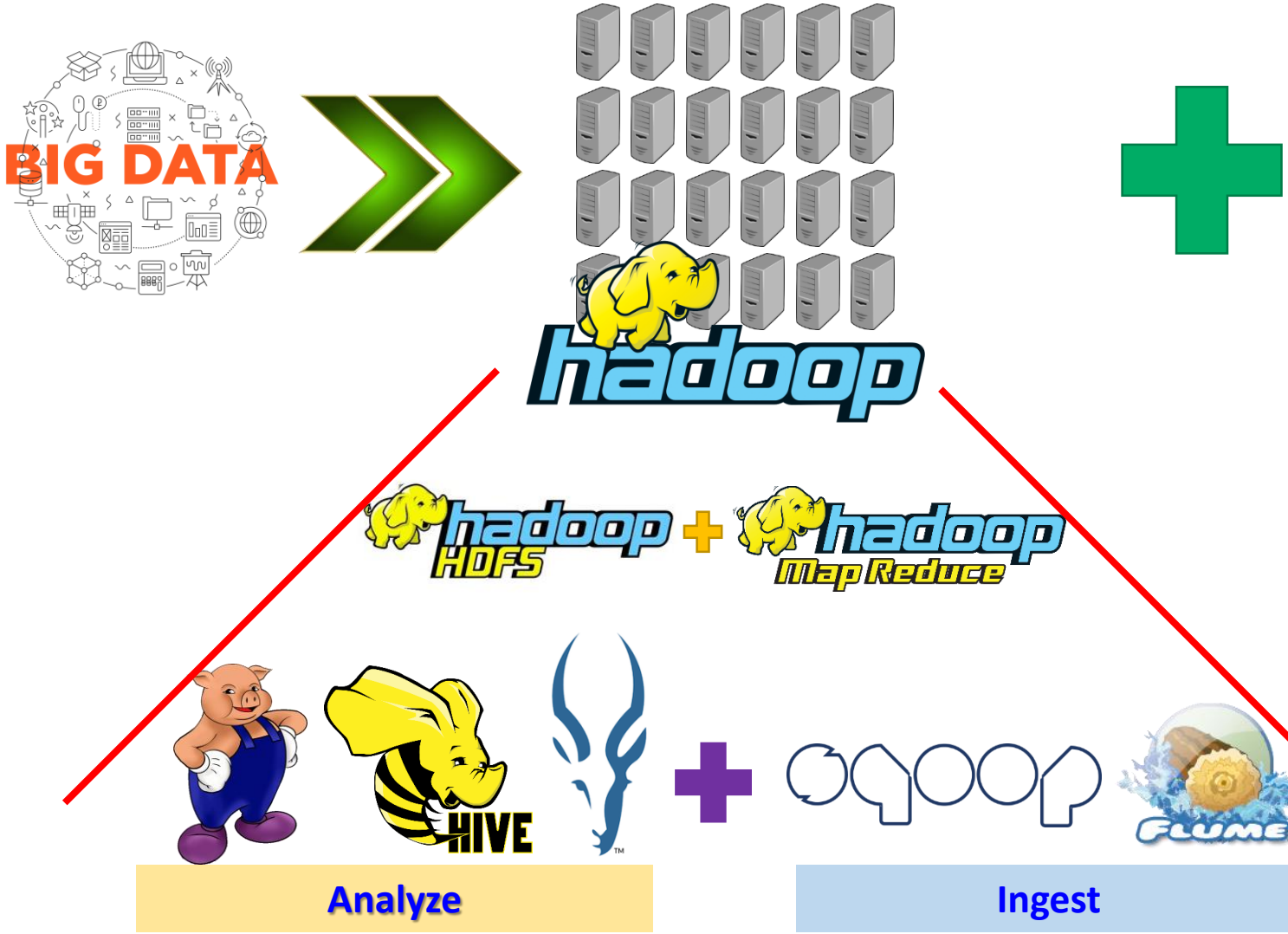
Course 8

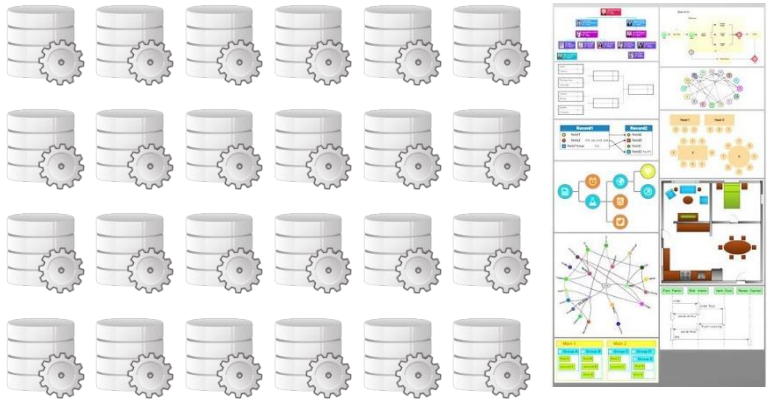
Data Ingestion in Hadoop

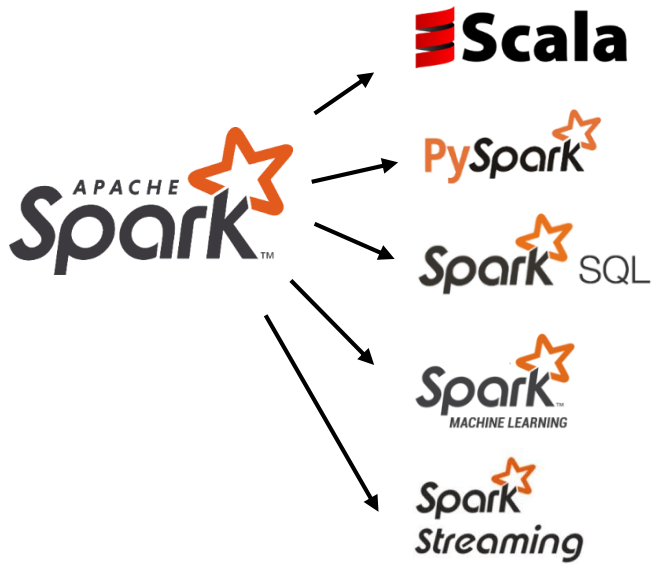
Khaled Tannir

McGill

Machine Learning at Scale







Machine Learning at Scale

Volume

Low Velocity — High Velocity

Distributed Storage

Volume

On Disk — In-Memory

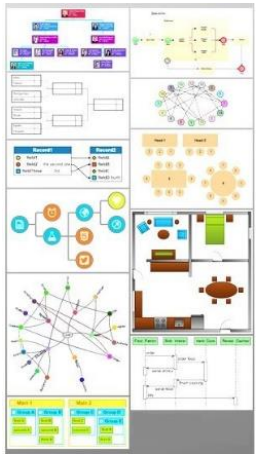
Distributed Processing

Velocity

Low Velocity — High Velocity

Variety

Unstructured — Structured



Theme of this Course

Data Ingestion in Hadoop

- *Apache FLume*
  - *Core Concepts*
  - *Real-time data ingestion*

Apache Flume

Ingesting data in near real-time to Hadoop

McGill

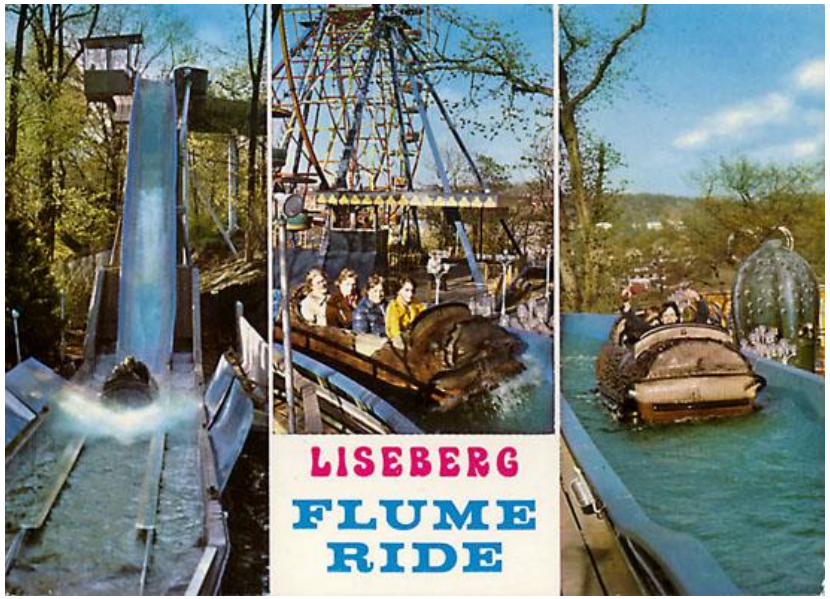
FALL 2018

What is Apache Flume ?

- Apache Flume is a continuous data ingestion system
- Evolved to handle any type of streaming event data
- Highly customizable and extendable
- Low-cost of installation, operation and maintenance
- Originally designed to be a log aggregation system by Cloudera
- It is open-source

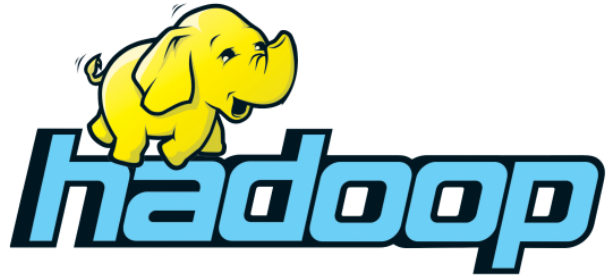
Apache Flume

- **Distributed**
  - *Agents installed on many machines*
- **Scalable**
  - *Add more machines to transfer more events*
- **Reliable**
  - *Durable storage, failover and/or replication*
- **Manageable**
  - *Easy to install, configure, reconfigure and run*



Apache Flume

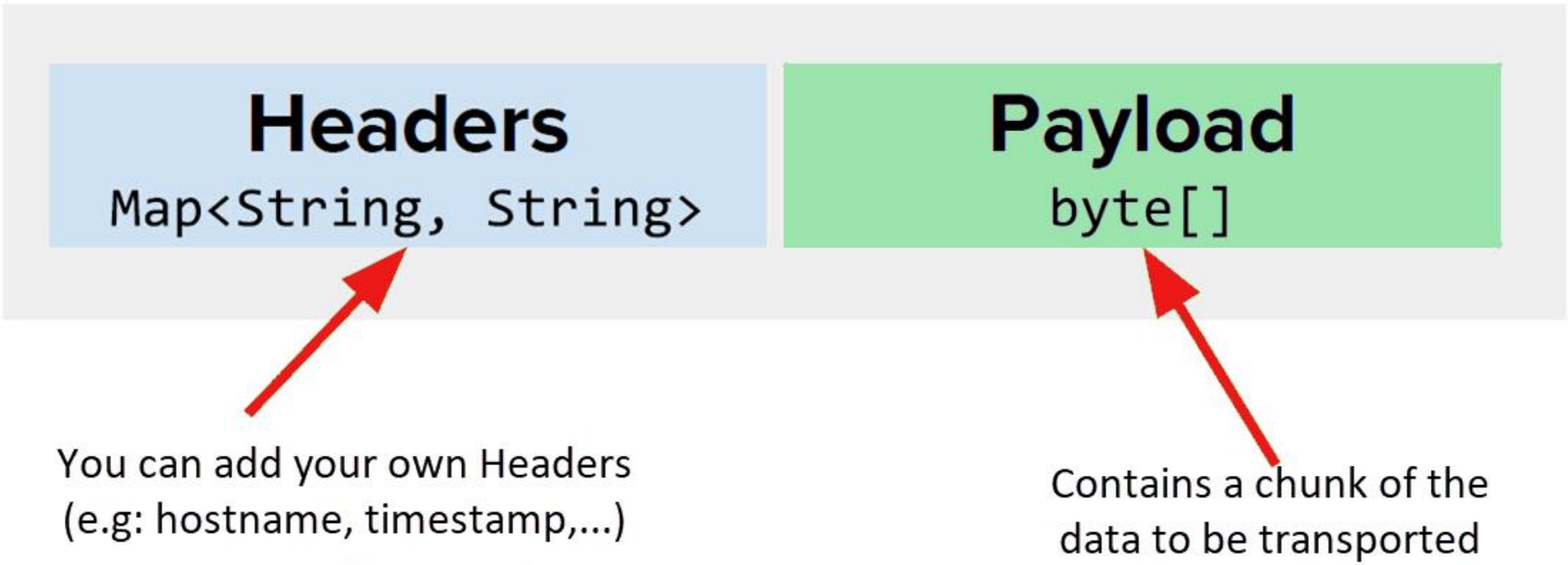
- **Integrated into the Hadoop system**
  - Various destinations e.g. HDFS, HBase, Hive, …
  - Various file formats e.g. Avro, SequenceFile



- **Extensible**
  - Possibility to add new functionality e.g. source and destination for events

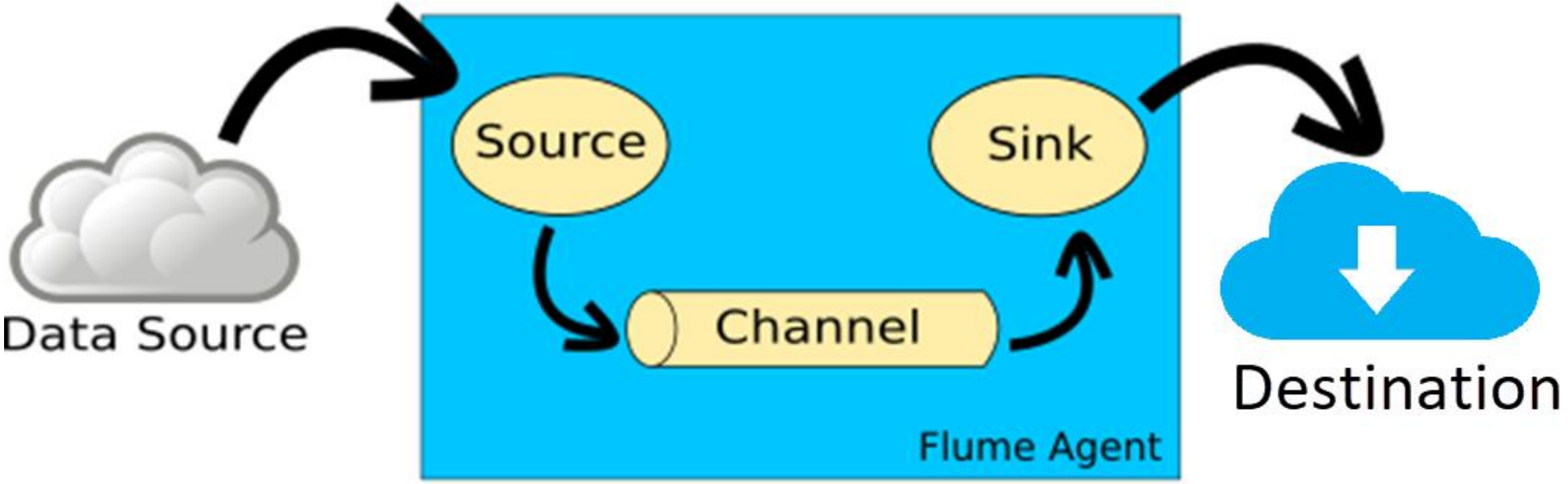
Flume : Event

- **Unit of data transported by Flume**
  - **Headers are collection of unique Key-Value pairs**
  - **Headers are used for contextual routing**



Global view of a Flume Agent

- **Responsible for transferring events**



Runs in JVM

Flume Agent : Source

- **Source**
  - Accepts incoming Data and forwards events in channels
  - Scales as required
  - Writes data to Channel

(Requires at least one Channel to function)

- HTTP, JMS, RPC
- netcat
- Exec
- Spooling directory
- …

Flume Exec Source

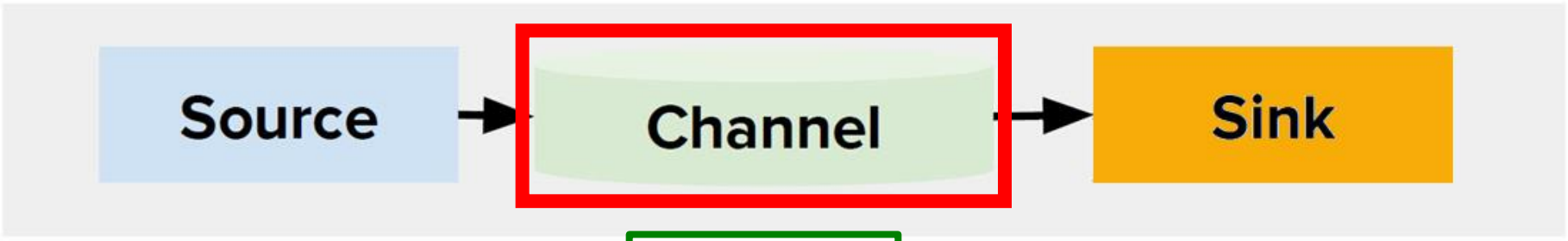
- **Exec Source**
  - Runs a given Unix command on startup
  - Should continuously run and produce data on
  - If the process exits, the source also exits and will NOT produce any further data

Flume Spooling Directory Source

- **Spooling Directory Source**
  - Watches a specified directory for new files
  - Parses events out of new files as they appear
  - After a file has been fully processed, it is renamed to indicate completion (or optionally deleted)

Flume Agent : Channel

- **Channel**
  - Stores data in the order received
  - Buffers incoming events until they are extracted by Sinks
  - Tradeoff between durability and throughput



- Memory
- File
- JDBC
- Kafka
- …

Channels are fully transactional

Flume Memory Channel

- **Memory Channel**
  - Events stored in an in-memory queue
  - Configurable capacity

    The maximum number of events and/or bytes in memory
  - Nondurable, but faster

Flume File Channel

- **File Channel**
  - Events stored in file on disk
  - Configurable capacity

    Flushes to disk at the end of each transaction
  - Durable, Supports encryption

Flume Agent : Sink

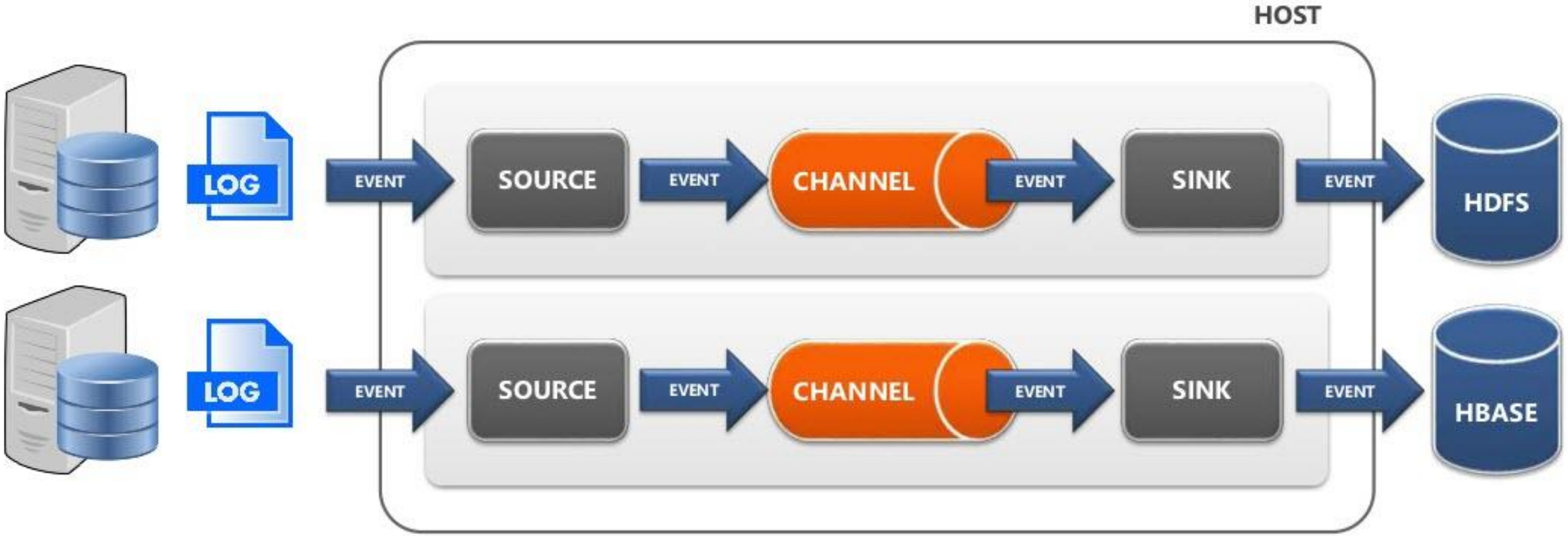
- **Sink**
  - Removes data from a Channel
  - Sends data to downstream agent or Destination

- HDFS, HBase, Solr, ElasticSearch
- File, Logger, Kafka
- Agent Flume
- …

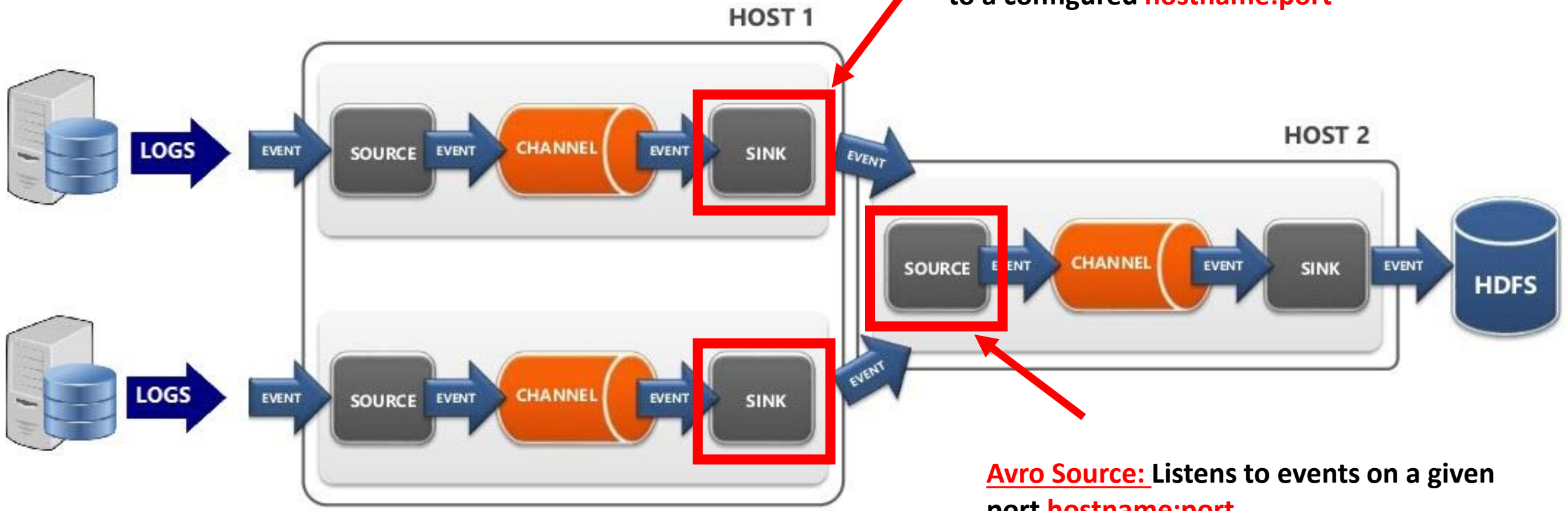
Flume HDFS Sink

- **HDFS Sink**
  - Writes events to HDFS
    - *Flexible naming of HDFS path*
  - Multiple file formats are supported (e.g. Text, Avro)
  - Rolling a file will generate many small files
    - *Need to compact them to avoid an explosion of HDFS metadata*

Flume : Running Agents in Parallel



Flume : Consolidation



Avro Sink: Sends a batch of Avro events to a configured hostname:port

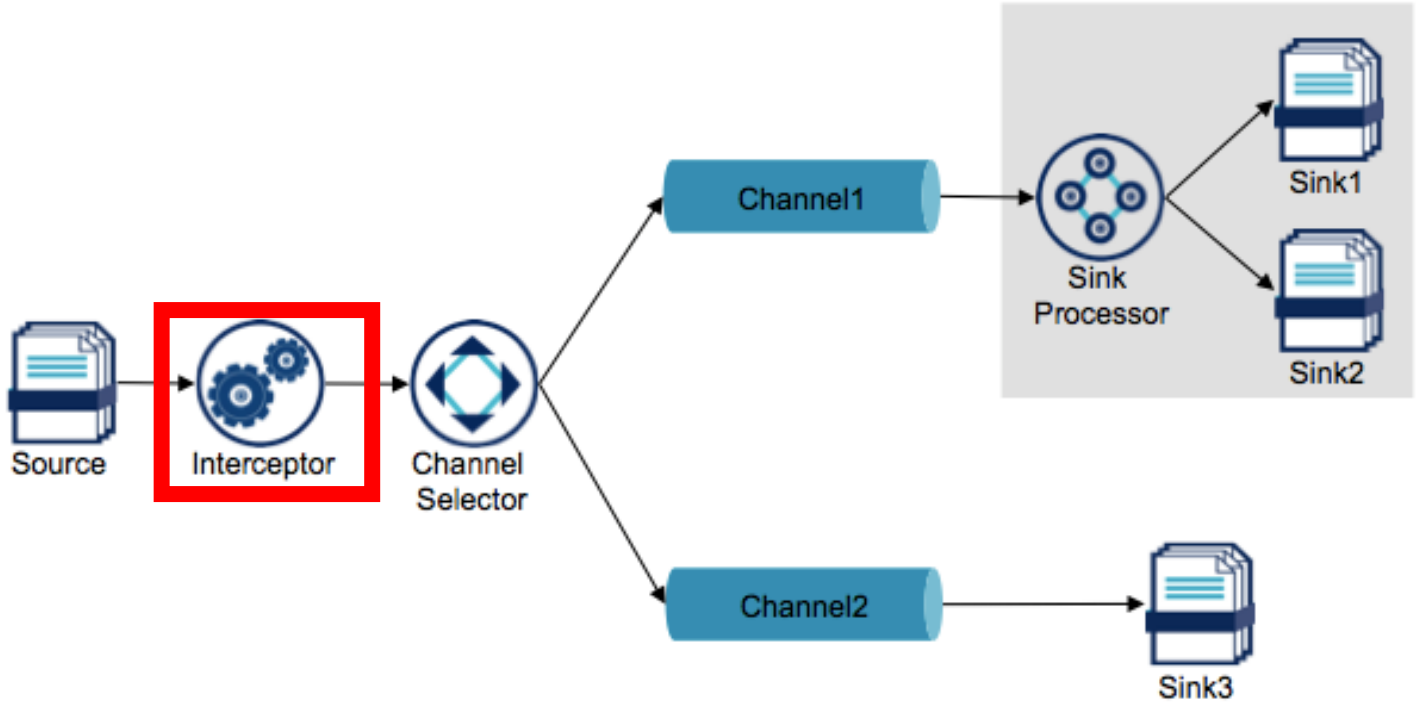
Avro Source: Listens to events on a given port hostname:port

Flume : Routing and Replicating

- Source can replicate or multiplex data across many channels
- Metadata headers can be used to do contextual selection of channels
- Channels can be drained by different sinks to different destinations or pipelines

Flume Agent Interceptors

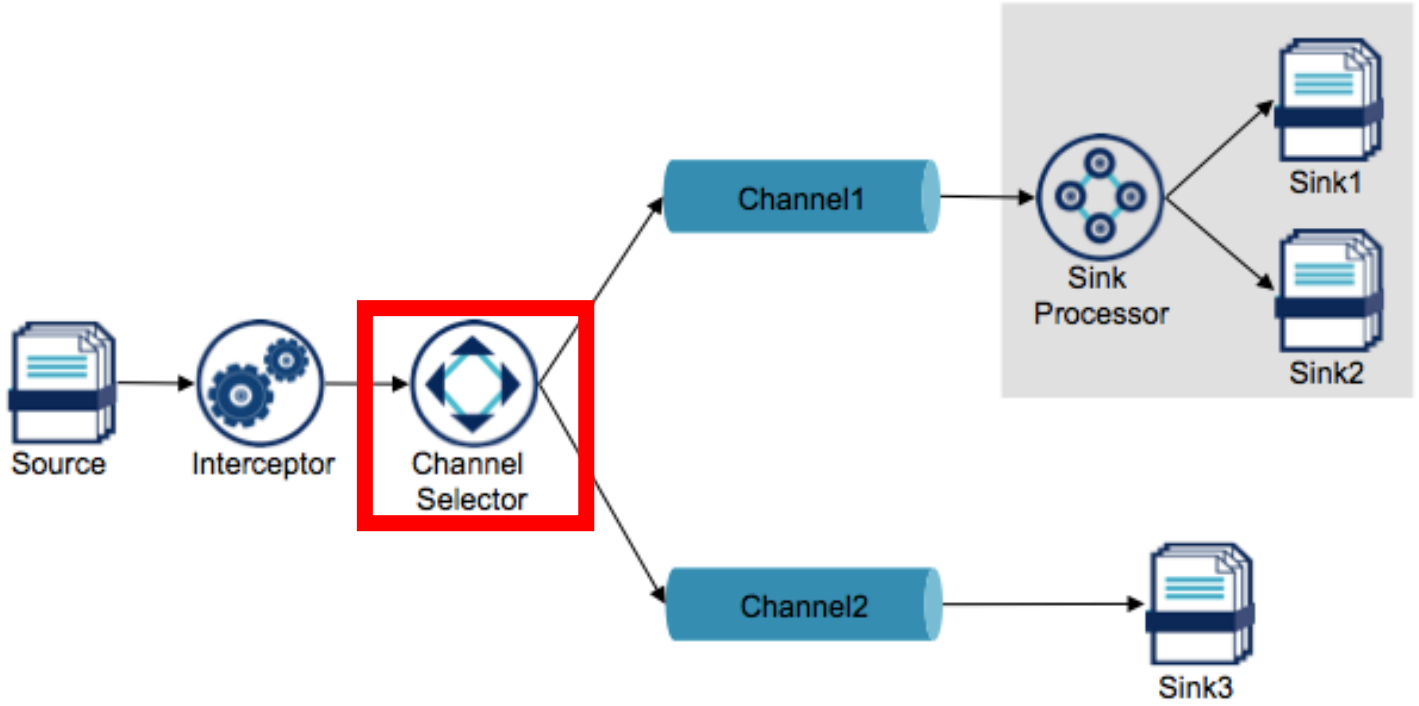
- **Interceptors** are applied to sources to enable adding information and filtering of events



- **Built in Interceptors**: Allows adding headers such as timestamps, host, static markers etc.
- **Custom Interceptors**: Create headers by inspecting the Event.

Flume Channel Selector

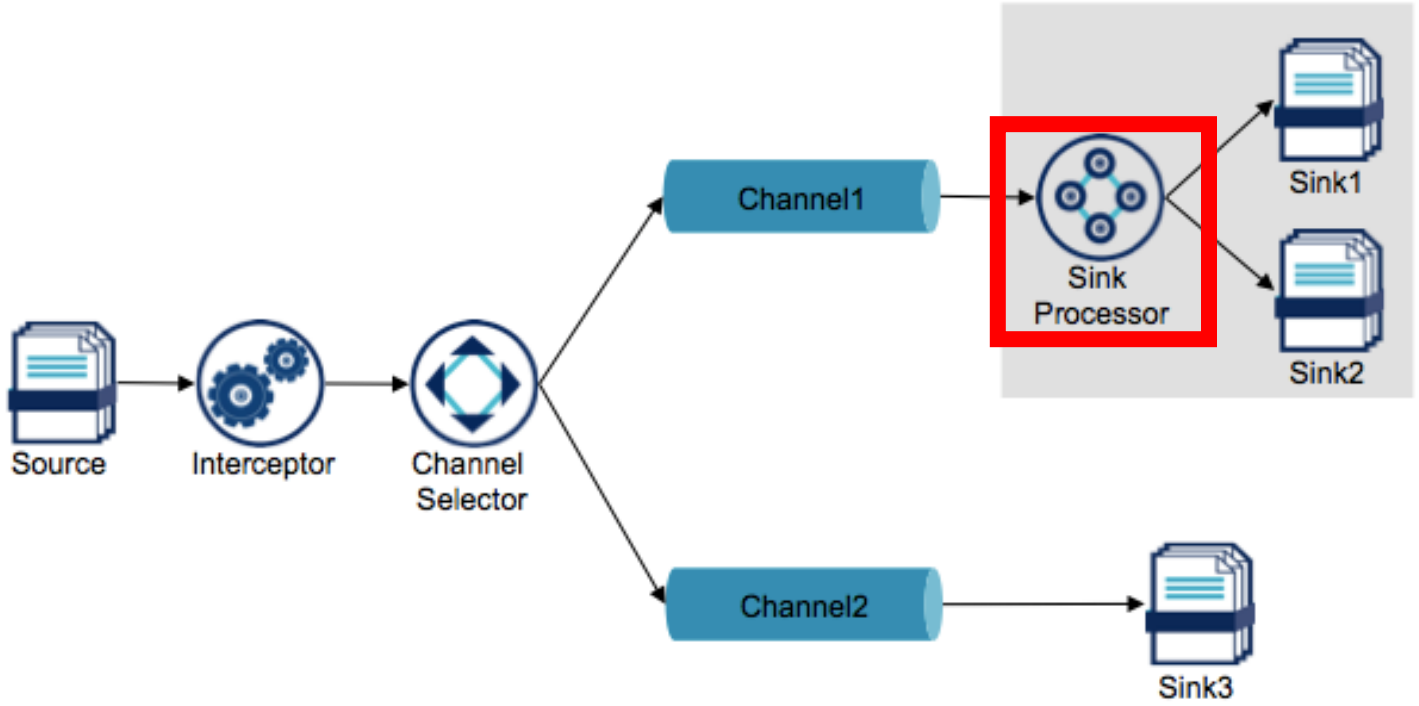
- **Channel Selector** It facilitates selection of one or more Channels, based on preset criteria.



- **Built in Channel Selectors:**
  - **Replicating**: for duplicating events
  - **Multiplexing**: for routing based on headers
- **Custom Channel Selectors**: can be written for dynamic criteria.

Flume Sink Processor

- **Sink Processor** is responsible for invoking one sink from a specified group of Sinks



- **Built in Sink Processors**:
  - **Load Balancing** Sink Processor
  - **Failover** Sink Processor
  - **Default** Sink Processor

Flume Configuration File

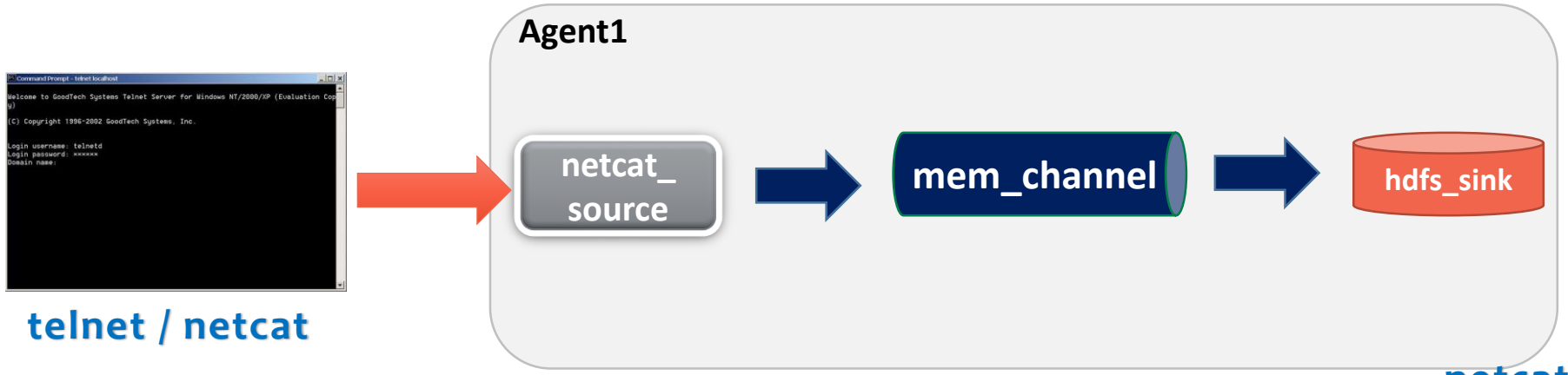
Introduction to Flume agent configuration File

Flume : Configuration File

- Simple text file format
- Flume file configuration must contains configuration for the three components (source, channel, sink) of an Agent
- Flume will load required parameters only
- In case of the configuration file has been updated Flume will reload the configuration file

Flume : Agent Configuration Example

- We configure Flume to run a single agent that :
  - Read data coming on the Tcp port : **51000**
  - Routing collected events from the source via a memory channel
  - Write events to a particular HDFS directory



Flume Agent Configuration Example

Agent Name: agent1

Assigning a name to each component

```
# Declare variables names
agent1.sources =    netcat_source
agent1.channels =   mem_channel
agent1.sinks =      hdfs_sink
```

Flume Agent Configuration Example


```
# Configuring the source
agent1.sources.netcat_source.type = netcat
agent1.sources.netcat_source.bind = localhost
agent1.sources.netcat_source.port = 51000

# Bind the source to the channel
agent1.sources.netcat_source.channels = mem_channel

# Optional
agent1.sources.netcat_source.interceptors = ts hs
agent1.sources.netcat_source.interceptors.ts.type = timestamp
agent1.sources.netcat_source.interceptors.hs.type = host
```

Flume Agent Configuration Example


```
# Configuring the channel
agent1.channels.mem_channel.type = memory
agent1.channels.mem_channel.capacity = 100000
agent1.channels.mem_channel.transactionCapacity = 1000
```

Flume Agent Configuration Example


```
# Configuring the sink
agent1.sinks.hdfs_sink.type = hdfs
agent1.sinks.hdfs_sink.hdfs.path =
                                    /flume/events/%{host}/%Y-%m-%d/%H-%M
agent1.sinks.hdfs_sink.hdfs.fileType = DataStream

# Optional
agent1.sinks.hdfs_sink.hdfs.filePrefix = netcat_log

# Bind the sink to the channel
agent1.sinks.hdfs_sink.channel = mem_channel
```

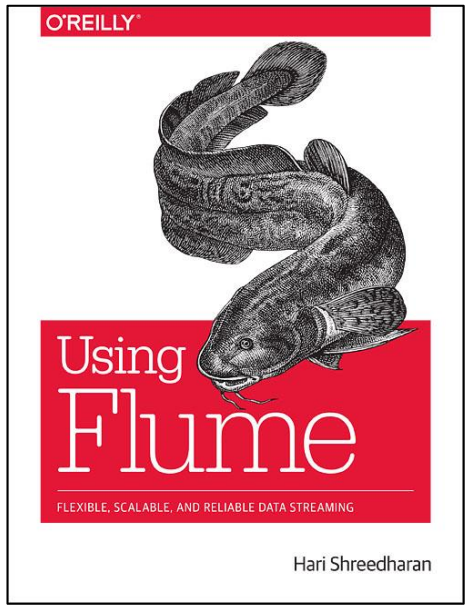
Starting Flume Agent

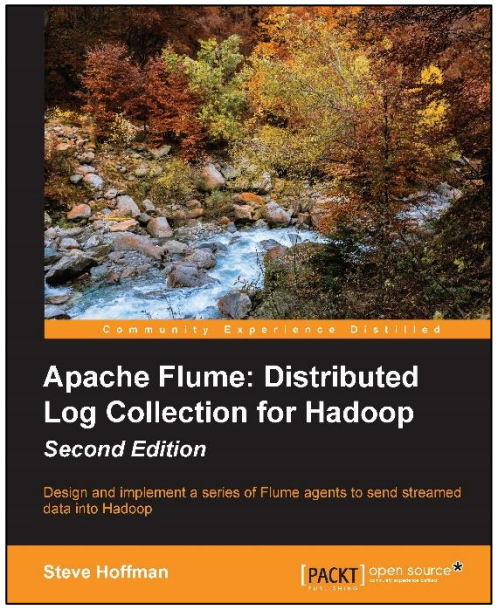

```
$ flume-ng agent --conf conf
-f {path}/netcat-to-hdfs.conf -n agent1
```

To stop Flume Agent

Flume Workshop

Resources





Online Resources

https://flume.apache.org/

Thank You