Hadoop
Sqoop –Flume

(Frame work for Data Ingestion)

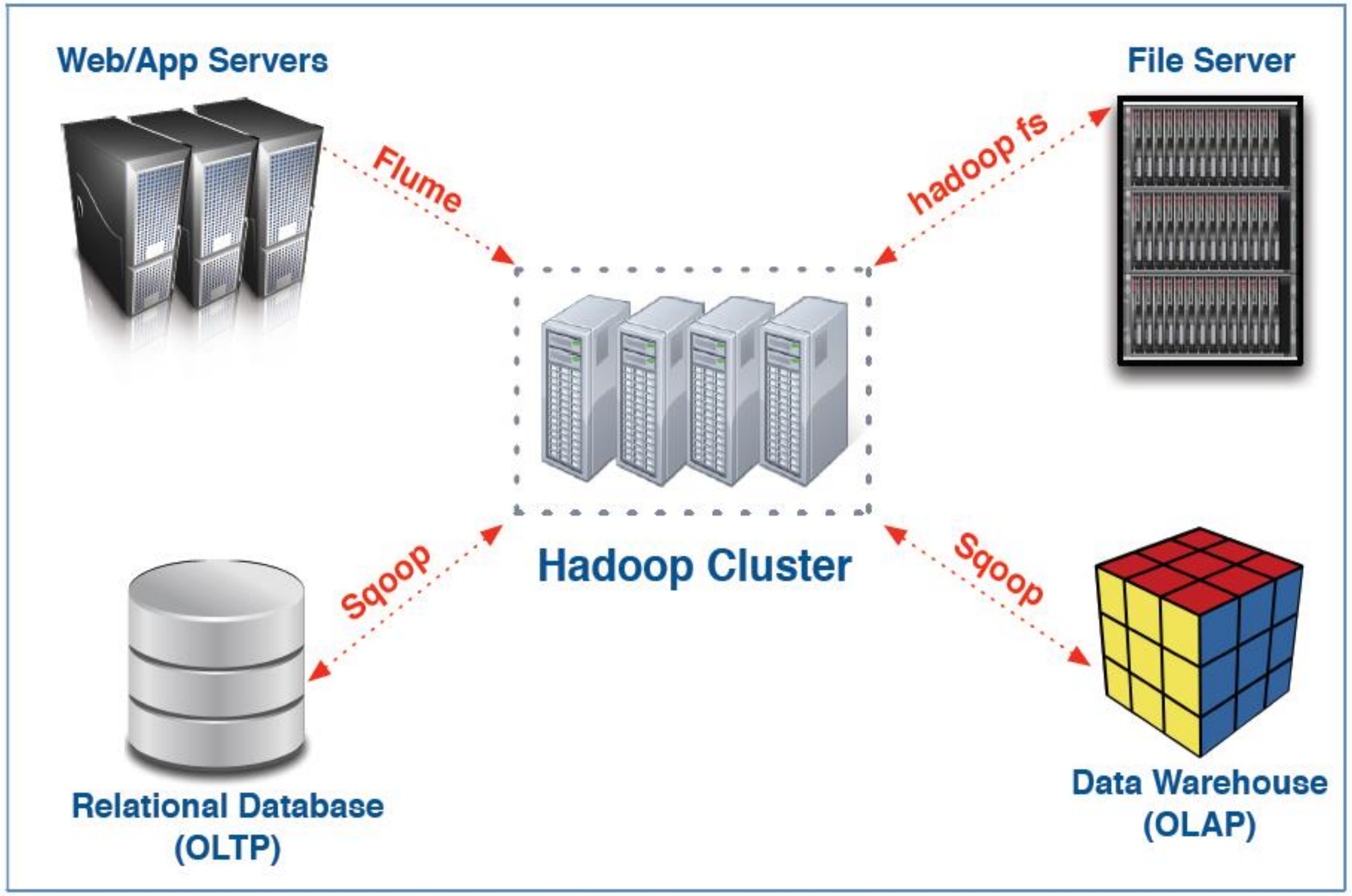
Getting Data into HDFS

In the last session, we learned how to use hadoop fs command to copy the data into and out of HDFS. Now we will see,

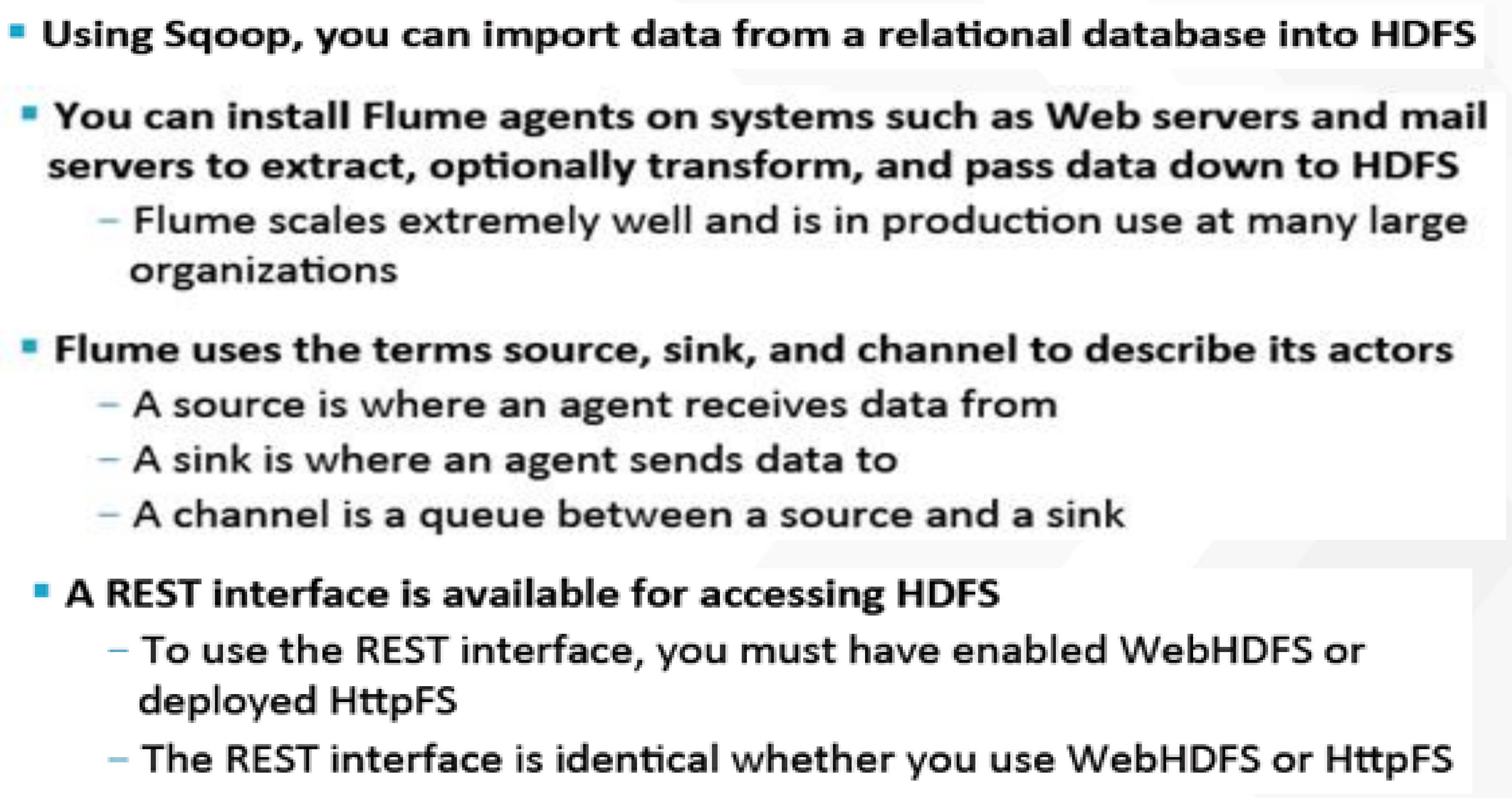
➢How to import data into HDFS using SQOOP?

➢How to import data into HDFS using Flume?

➢What REST interfaces Hadoop provides?



Getting Data into HDFS



- Using Sqoop, you can import data from a relational database into HDFS
- You can install Flume agents on systems such as Web servers and mail servers to extract, optionally transform, and pass data down to HDFS
  - Flume scales extremely well and is in production use at many large organizations
- Flume uses the terms source, sink, and channel to describe its actors
  - A source is where an agent receives data from
  - A sink is where an agent sends data to
  - A channel is a queue between a source and a sink
- A REST interface is available for accessing HDFS
  - To use the REST interface, you must have enabled WebHDFS or deployed HttpFS
  - The REST interface is identical whether you use WebHDFS or HttpFS

Flume

What is Flume

Flume is a distributed, reliable, available service for efficiently moving large amounts of data as it is produced

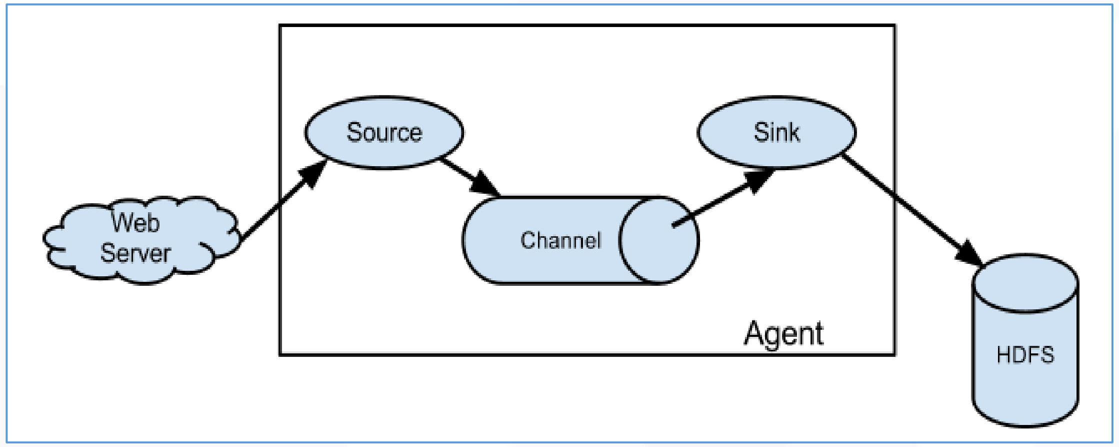
– Ideally suited to gathering logs from multiple systems and inserting
them into HDFS as they are generated

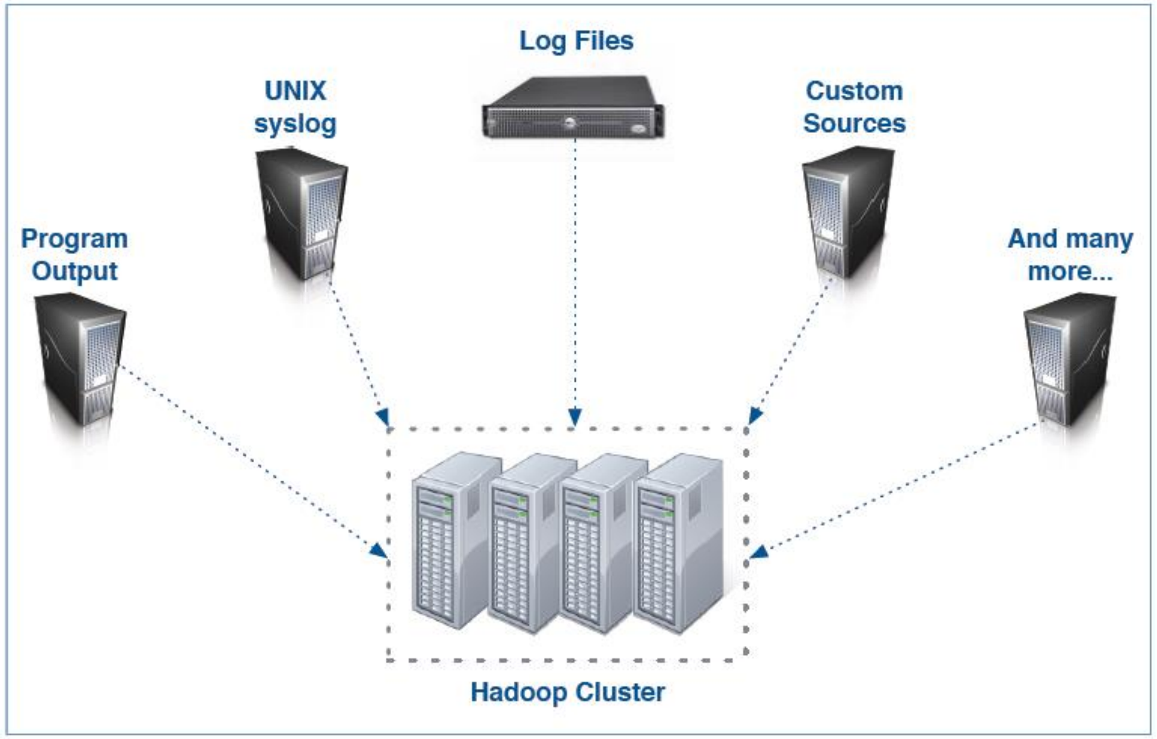
Flume is an open source Apache project

- – Initially developed by Cloudera
- – Included in CDH

Flume's design goals:

- – Reliability
- – Scalability
- – Extensibility

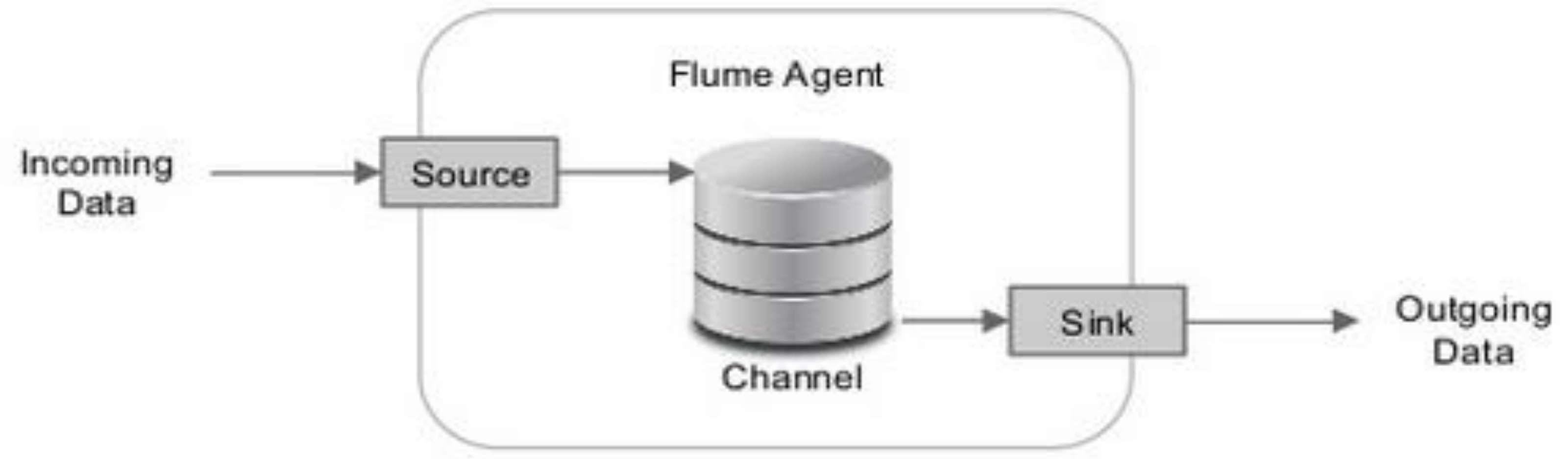




ANALYTIXLABS

High Level Overview

Each Flume agent has three components source, channel and a sink

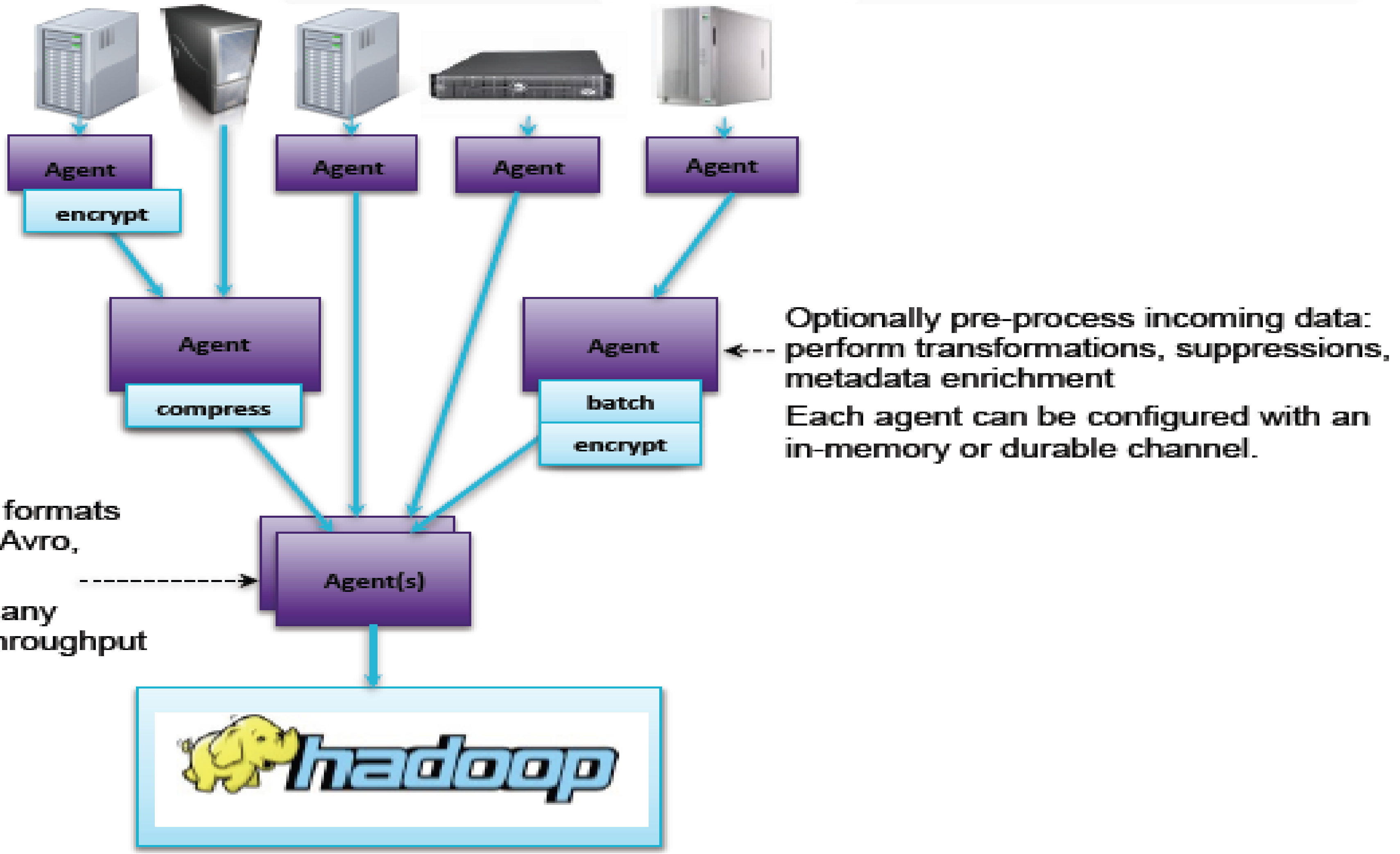


Source
- Accepts incoming Data
- Scales as required
- Writes data to Channel

Channel
- Stores data in the order received

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

High Level Overview – HDFS as Sink



Agent
encrypt
Agent
Agent
Agent
Agent
compress
Agent
batch
encrypt
Optionally pre-process incoming data: perform transformations, suppressions, metadata enrichment
Each agent can be configured with an in-memory or durable channel.
Agent(s)
hadoop

Writes to multiple HDFS file formats (text, SequenceFile, JSON, Avro, others)

Parallelized writes across many collectors – as much write throughput as required

How Flume helps Hadoop to get data from live streaming?

Flume allows the user to do the following:

✓Flume is typically used to ingest log files from real time systems such as Web servers, firewalls, and mail servers into HDFS

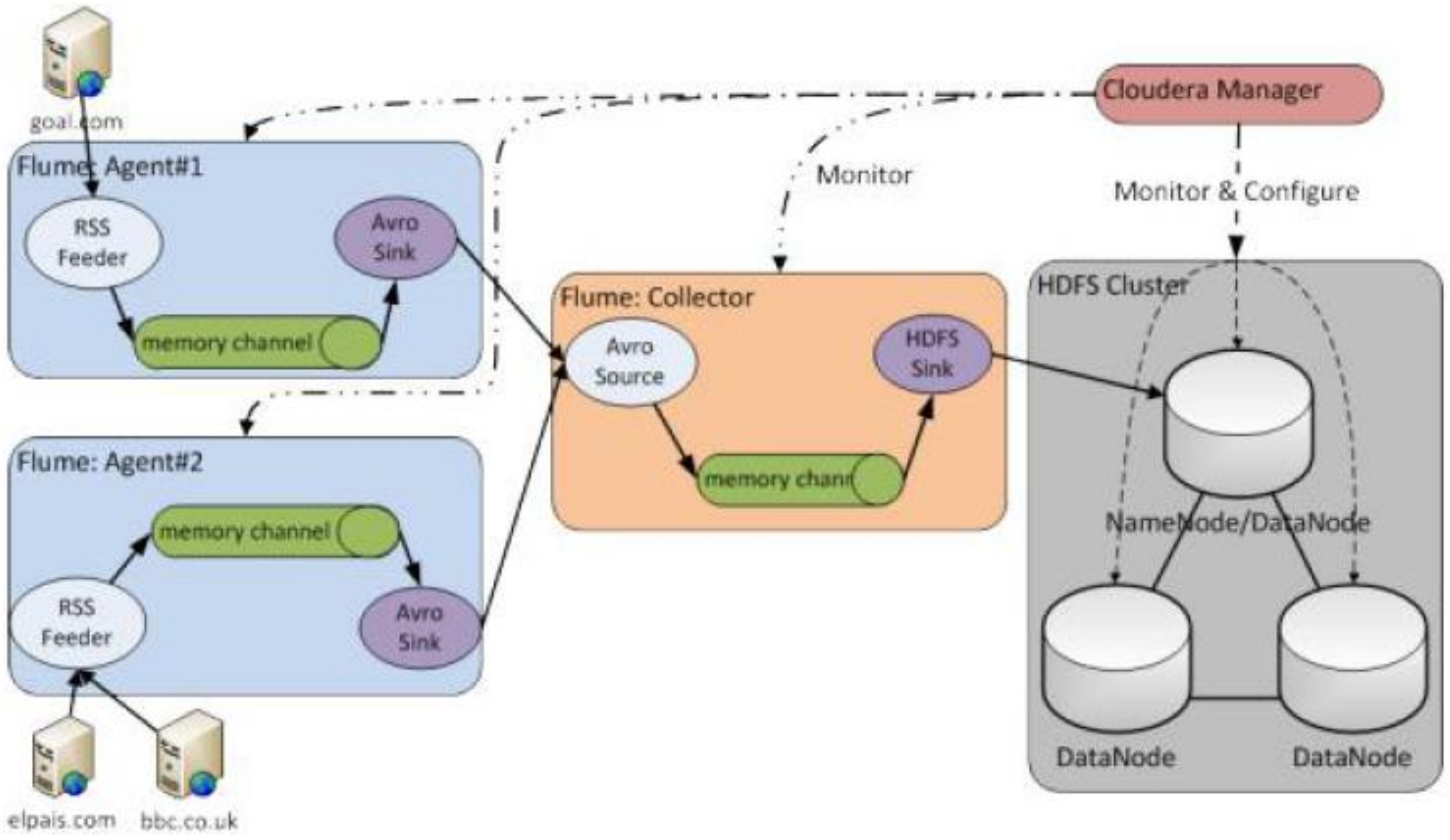
✓Currently in use in many large organizations, ingesting millions of events per day

✓It acts as a buffer when the rate of incoming data exceeds the rate at which the data can be written. Thereby preventing data loss.

✓Guarantees data delivery.

✓Scales horizontally (connects commodity system in parallel) to handle additional data volume.

Flume Architecture



Essential Components Involved in Getting Data from a Live-Streaming Source

There are 3 major components, namely: Source, Channel, and Sink, which are involved in ingesting data, moving data and storing data, respectively.

Below is the breakdown of the parts applicable in this scenario:

- ✓**Event** – A singular unit of data that is transported by Flume (typically a single log entry).
- ✓**Source** – The entity through which data enters into the Flume. Sources either actively samples the data or passively waits for data to be delivered to them. A variety of sources such as log4j logs and syslogs, allows data to be collected.
- ✓**Sink** – The unit that delivers the data to the destination. A variety of sinks allow data to be streamed to a range of destinations. Example: HDFS sink writes events to the HDFS.
- ✓**Channel** – It is the connection between the Source and the Sink. The Source ingests Event into the Channel and the Sink drains the Channel.
- ✓**Agent** – Any physical Java virtual machine running Flume. It is a collection of Sources, Sinks and Channels.
- ✓**Client** – It produces and transmits the Event to the Source operating within the Agent
- ✓**Flow:** Movement of events from the point of origin to their final destination

Flume Sources & Sinks

- Avro Source
- Thrift Source
- Exec Source
- JMS Source
- Spooling Directory Source
  - Event Deserializers: LINE, AVRO, BlobDeserializer
- Twitter 1% firehose Source (experimental)
- Kafka Source
- NetCat Source
- Sequence Generator Source
- Syslog Sources
  - Syslog TCP Source, Multiport Syslog TCP Source
  - Syslog UDP Source, HTTP Source
  - JSONHandler, BlobHandler
- Stress Source
- Legacy Sources
  - Avro Legacy Source, Thrift Legacy Source
- Custom Source
- Scribe Source

- HDFS Sink
- Hive Sink
- Logger Sink
- Avro Sink
- Thrift Sink
- IRC Sink
- File Roll Sink
- Null Sink
- HBaseSinks
  - HBaseSink
  - AsyncHBaseSink
- MorphlineSolrSink
- ElasticSearchSink
- Kite Dataset Sink
- Kafka Sink
- Custom Sink

- Memory Channel
- JDBC Channel
- Kafka Channel
- File Channel
- Spillable Memory Channel
- Pseudo Transaction Channel
- Custom Channel

Flume Design Goals: Reliability

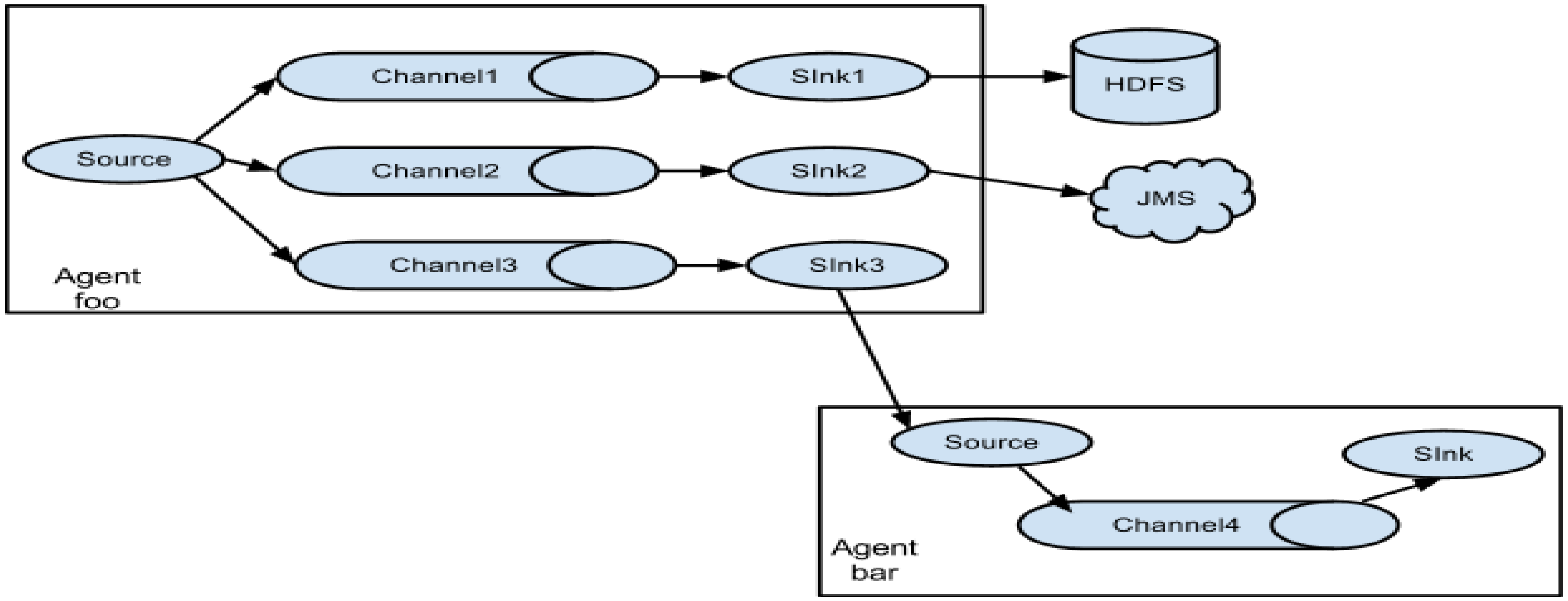
- **Channels provide Flume's reliability**
- **Memory Channel**
  - Data will be lost if power is lost
- **Disk-based Channel**
  - Disk-based queue guarantees durability of data in face of a power loss
- **Data transfer between Agents and Channels is transactional**
  - A failed data transfer to a downstream agent rolls back and retries
- **Can configure multiple Agents with the same task**
  - e.g., 2 Agents doing the job of 1 'collector' – if one agent fails then upstream agents would fail over

Flume Design Goals: Scalability - Extensibility

- **Scalability**
  - The ability to increase system performance linearly – or better – by adding more resources to the system
  - Flume scales horizontally
    - As load increases, more machines can be added to the configuration
- **Extensibility**
  - The ability to add new functionality to a system
- **Flume can be extended by adding Sources and Sinks to existing storage layers or data platforms**
  - General Sources include data from files, syslog, and standard output from any Linux process
  - General Sinks include files on the local filesystem or HDFS
  - Developers can write their own Sources or Sinks

Flow Pipeline

- The client transmits the event to its next hop destination
- The source receiving this event will then deliver it to one or more channels
- The channels that receive the event are drained by one are more sinks operating within same agent
- Sink will forward event to final destination



Sentiment Analysis using Social Media data

Steps for data Streaming from Twitter to HDFS

Create Access tokens from twitter.com

- ✓ **Step 1:** Open a Twitter account
- ✓ **Step 2:** Go to the following link and click on 'create app'. (**https://apps.twitter.com/app**)
- ✓ **Step 3:** Fill in the necessary details.
- ✓ **Step 4:** Accept the agreement and click on 'create your Twitter application'.
- ✓ **Step 5:** Go to 'Keys and Access Token' tab.
- ✓ **Step 6:** Copy the consumer key and the consumer secret.
- ✓ **Step 7:** Scroll down further and click on 'create my access token'.
- ✓ Step 8: Copy the Access Token and Access token Secret.

Setting up raw data folders in HDFS and copy the data

Extract Data from Twitter using Flume

Import data into Hive and perform analysis

Integrate Excel-2013 or Tableau with Hiveserver2

High Level Steps for extracting data from twitter

Step 1: Download file flume-sources-1.0-SNAPSHOT.jar from the url
http://www.thecloudavenue.com/2013/03/analyse-tweets-using-flume-hadoop-and.html
or from http://files.cloudera.com/samples/flume-sources-1.0-SNAPSHOT.jar

create folder in myjars in /usr/lib/flume-ng.
#sudo mkdir /usr/lib/flume-ng/myjars

Put this file in /usr/lib/flume-ng/myjars
#sudo cp /home/cloudera/Desktop/Projects/flume-sources-1.0-SNAPSHOT.jar /usr/lib/flume-ng/myjars/

Step 2: Create a new app with apps.twitter.com. Generate the access tokens. Copy the consumer tokens and access tokens and use them in the twitter_conf3.conf file below.

Step 3: Create a config file called twitter.conf with below data and put it in the folder and provide full access to it (i.e full read and write access) /usr/lib/flume-ng/conf
#sudo cp /home/cloudera/Desktop/Projects/twitter_conf3.conf /usr/lib/flume-ng/conf/
#sudo chmod -777 /usr/lib/flume-ng/conf/twitter_conf3.conf

Step 4: Modify the file /usr/lib/flume-ng/conf/flume-env.sh file. Add the below line
FLUME_CLASSPATH="/usr/lib/flume-ng/myjars/flume-sources-1.0-SNAPSHOT.jar"
#vi /usr/lib/flume-ng/conf/flume-env.sh (enter i , copy the class path, enter ctrl+esc, enter :wq!)

Step 5: Run the following commands
cd /usr/lib/flume-ng/bin
./flume-ng agent -n TwitterAgent -c conf -f ../conf/twitter_conf3.conf   (or)
/usr/lib/flume/bin/flume-ng agent -conf ./conf/ -f/etc/flume/conf/twitter_conf.conf -Dflume.root.logger=DEBUG, console -n TwitterAgent

Step 6: Check the downloaded twitter docs in HDFS
hdfs dfs -ls /user/cloudera/data/tweets_raw/flume-23423432453

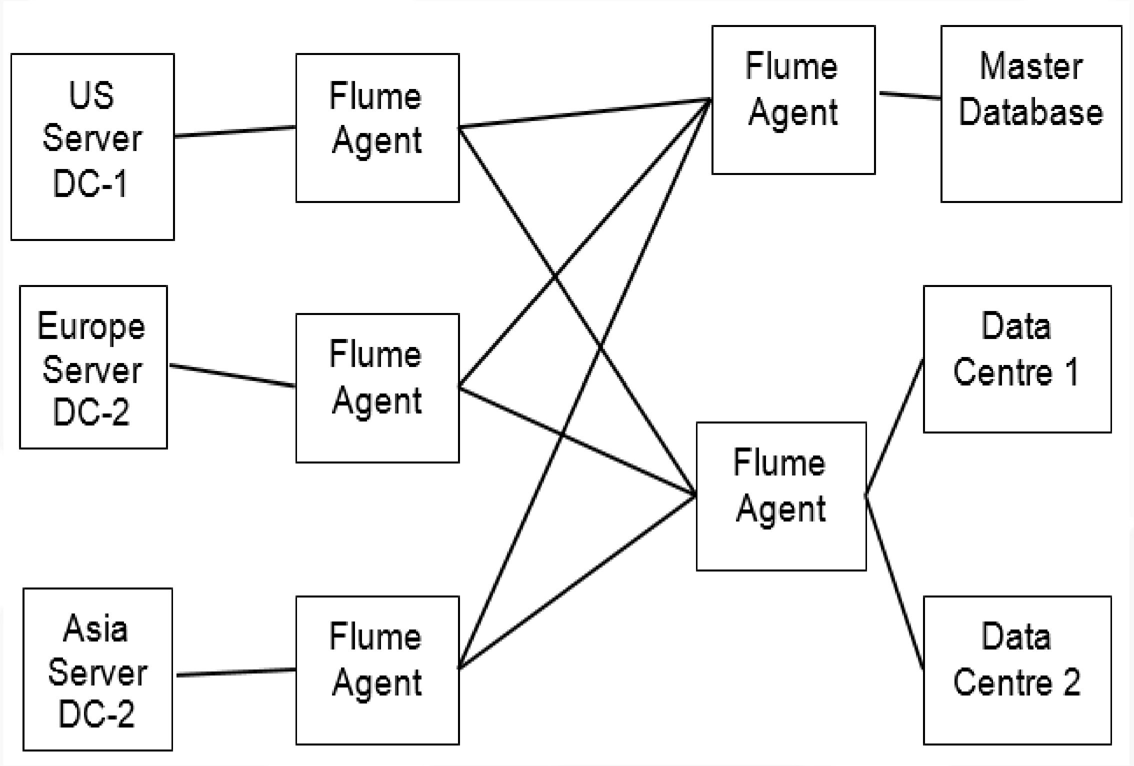
Flume Case Study- Website Log Aggregation

Problem Statement

✓This case study focuses on a multi hop flume agent to aggregate the log reports from various web servers which have to be analyzed with the help of Hadoop.

✓Consider a scenario we have multiple servers located in various locations serving from different data centers.

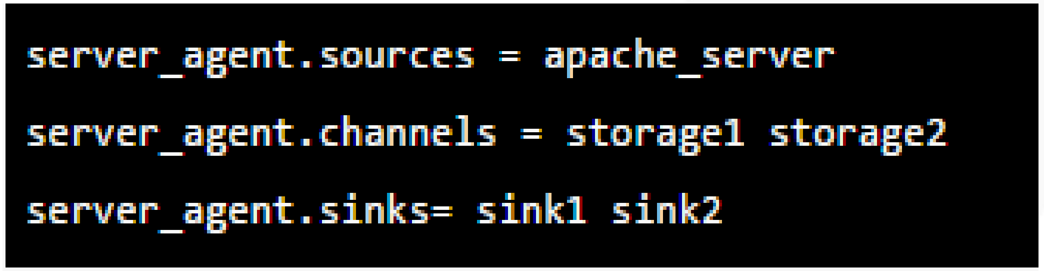
✓The objective is to distribute the log files based on the device type and store a backup of all logs. For example logs of US server has to be transferred based on the data center the server is located and copy all of the logs in the master database.



✓In this case every server flume agent has a single source and two channels and sinks. One sending the data to the main database flume agent and other to the flume agent that is dividing the data based on the user agent present in the logs

Proposed Solution

Before aggregating the logs, configuration has to be set for various components in our architecture.
In case of the web server flume system, as discussed two sinks are needed, channels to distribute the same data to both the destinations. In the beginning, define the names of the various components that are going to be used:

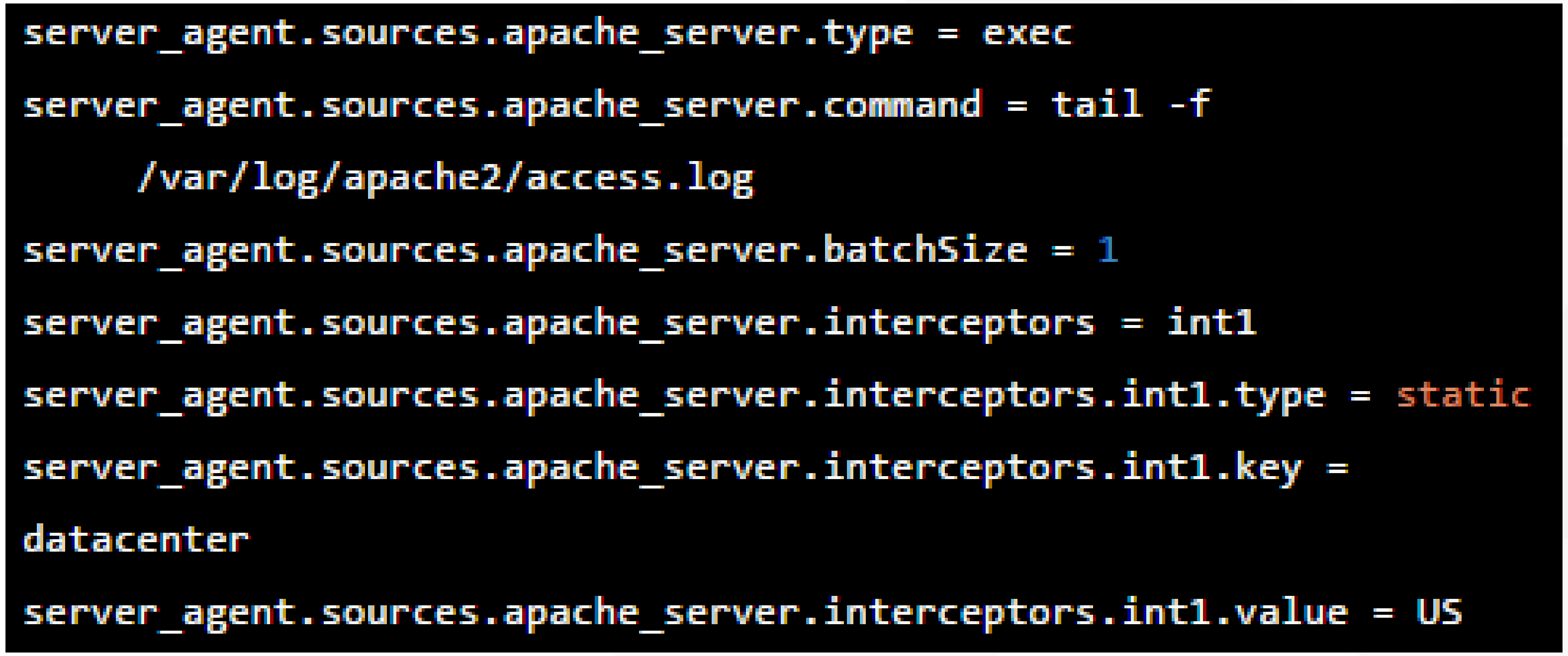


```
server_agent.sources = apache_server

server_agent.channels = storage1 storage2

server_agent.sinks= sink1 sink2
```

The source is configured to execute the command in the shell to retrieve the data. It is assumed as a Apache2 server and the logs location hasn’t been changed. The location can be varied based on the requirement. Then introduce a header for each event defining the data center from which it originated

Proposed Solution

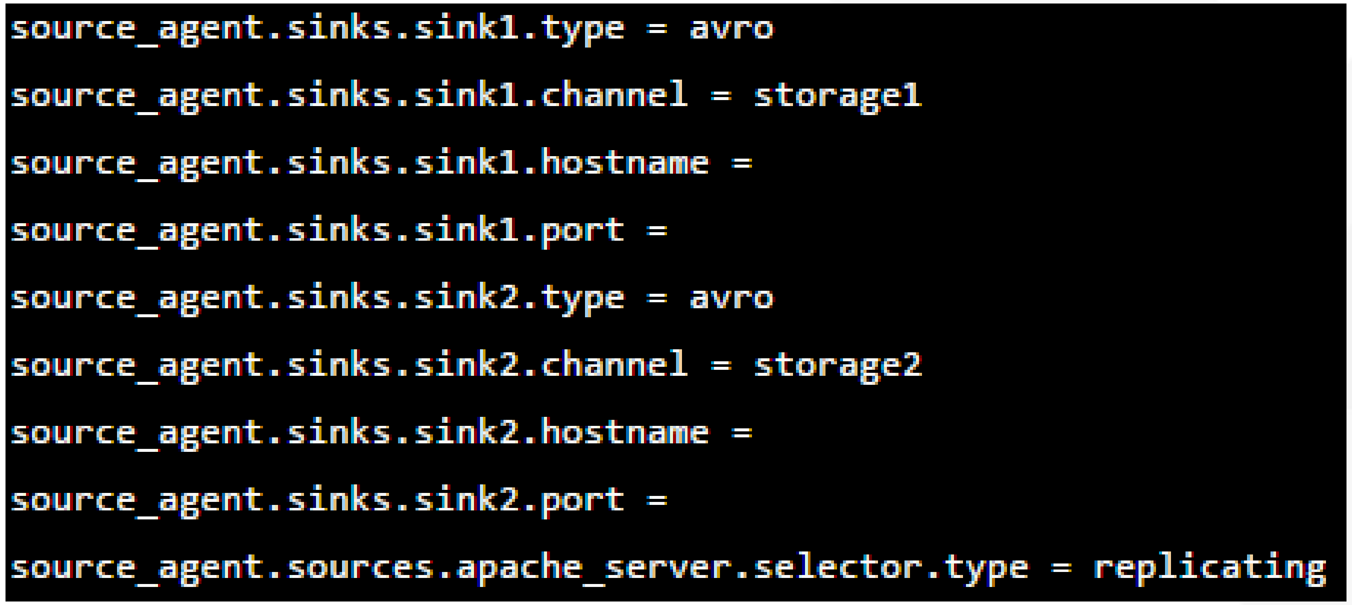
The source is configured to execute the command in the shell to retrieve the data.
It is assumed as a Apache2 server and the logs location hasn't been changed. The location can be varied based on the requirement. Then introduce a header for each event defining the data center from which it originated



```
server_agent.sources.apache_server.type = exec
server_agent.sources.apache_server.command = tail -f
     /var/log/apache2/access.log
server_agent.sources.apache_server.batchSize = 1
server_agent.sources.apache_server.interceptors = int1
server_agent.sources.apache_server.interceptors.int1.type = static
server_agent.sources.apache_server.interceptors.int1.key =
datacenter
server_agent.sources.apache_server.interceptors.int1.value = US
```

Proposed Solution

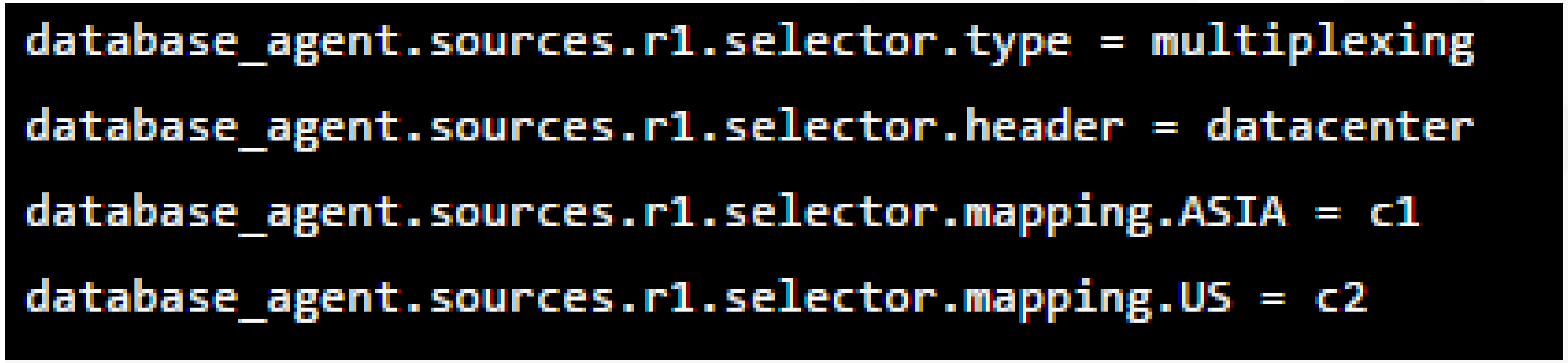
The sink is considered to be of avro sink retrieving events from the channel ‘storage1’.
It is connected to the source of the master database flume agent which has to be of avro type. It has been defined to replicate all the events received by source to all the sources in the agent. The same goes with another sink with a different channel, IP and port number. We have choose two different channel as an event is successfully sent to one sink it is immediately deleted and can’t be sent to other sink.



```
source_agent.sinks.sink1.type = avro

source_agent.sinks.sink1.channel = storage1

source_agent.sinks.sink1.hostname =

source_agent.sinks.sink1.port =

source_agent.sinks.sink2.type = avro

source_agent.sinks.sink2.channel = storage2

source_agent.sinks.sink2.hostname =

source_agent.sinks.sink2.port =

source_agent.sources.apache_server.selector.type = replicating
```

Proposed Solution

The same configuration for all the server flume agents with just a variation in the datacenter value. The sink1 sends the data to be stored in the master database while sink2 sends data to divide the data and store them in different databases. The code for the user agent based flume agent is similar to the server agent code with additional feature of Multiplexing as different events have to be sent to different channels based on the header value. Select the header ‘datacenter’ and divide the data between channels c1 and c2 based on the value of the header.

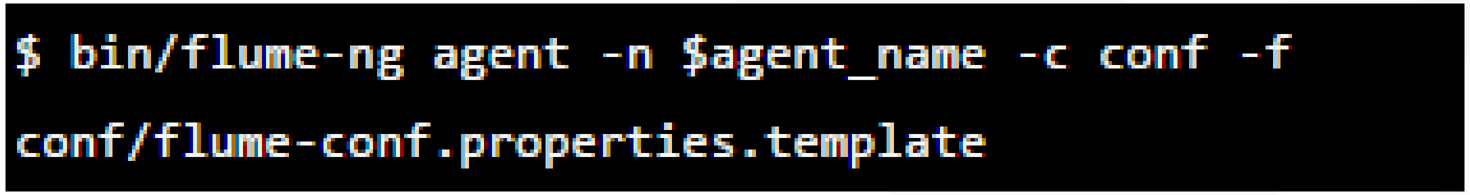


```
database_agent.sources.r1.selector.type = multiplexing

database_agent.sources.r1.selector.header = datacenter

database_agent.sources.r1.selector.mapping.ASIA = c1

database_agent.sources.r1.selector.mapping.US = c2
```

Proposed Solution

Executing Solution

Start individual agents on each server using the following command:

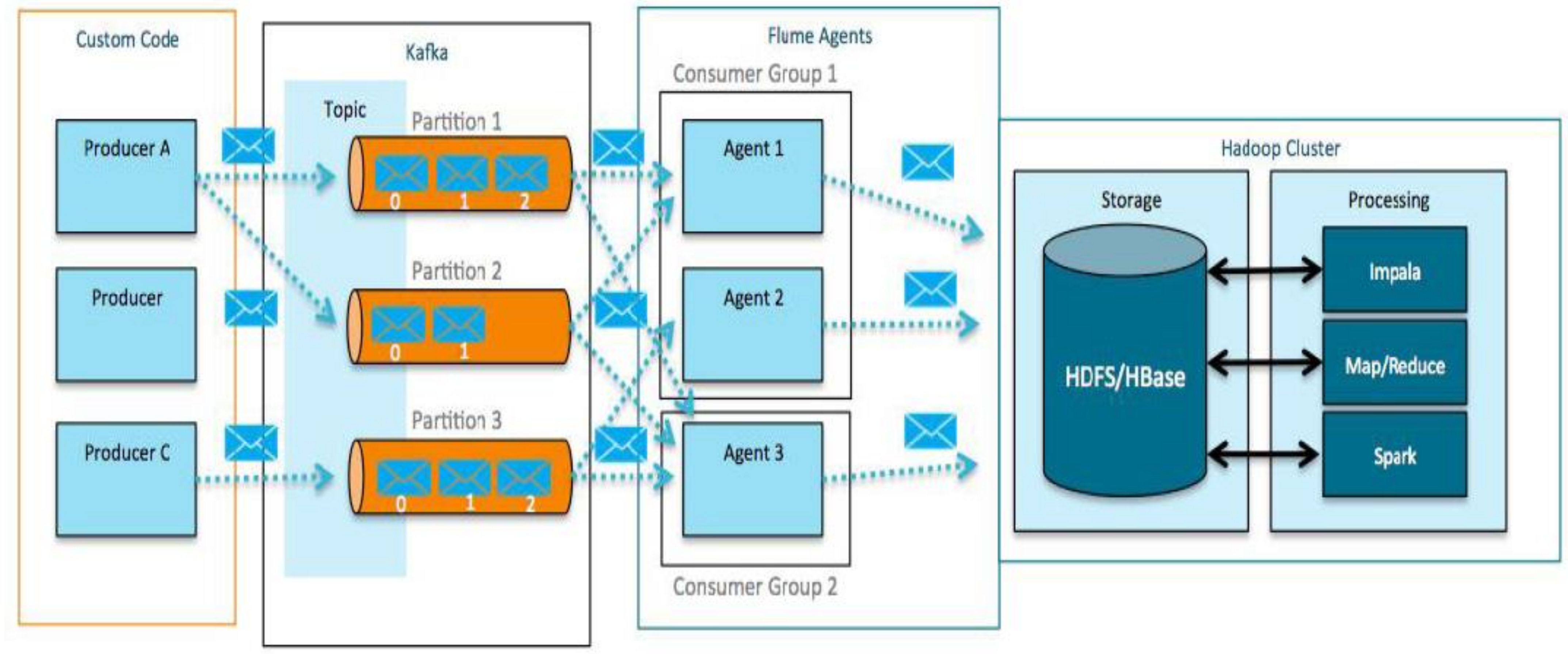


```
$ bin/flume-ng agent -n $agent_name -c conf -f
conf/flume-conf.properties.template
```

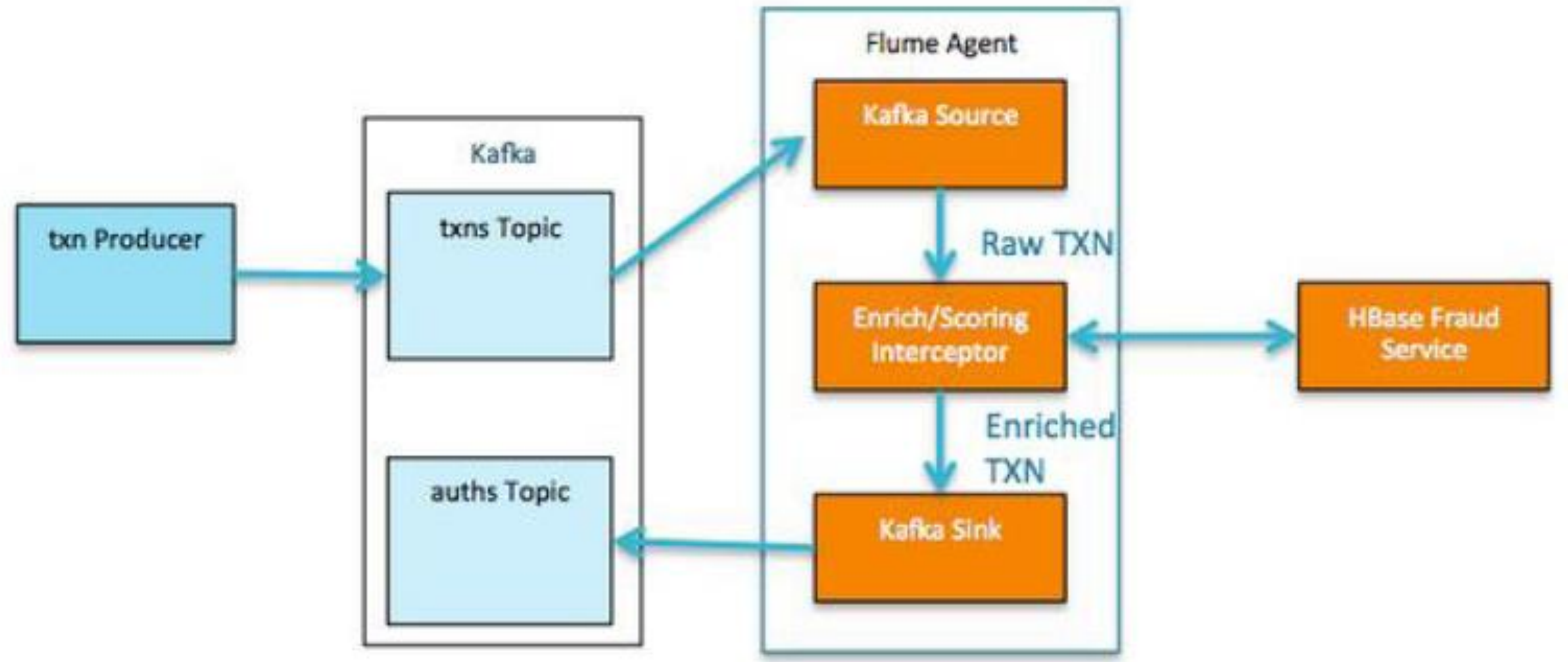
As the log reports are generated in the apache log file, the log reports are transferred to various servers as required without bandwidth or security concerns with better reliability

Appendix

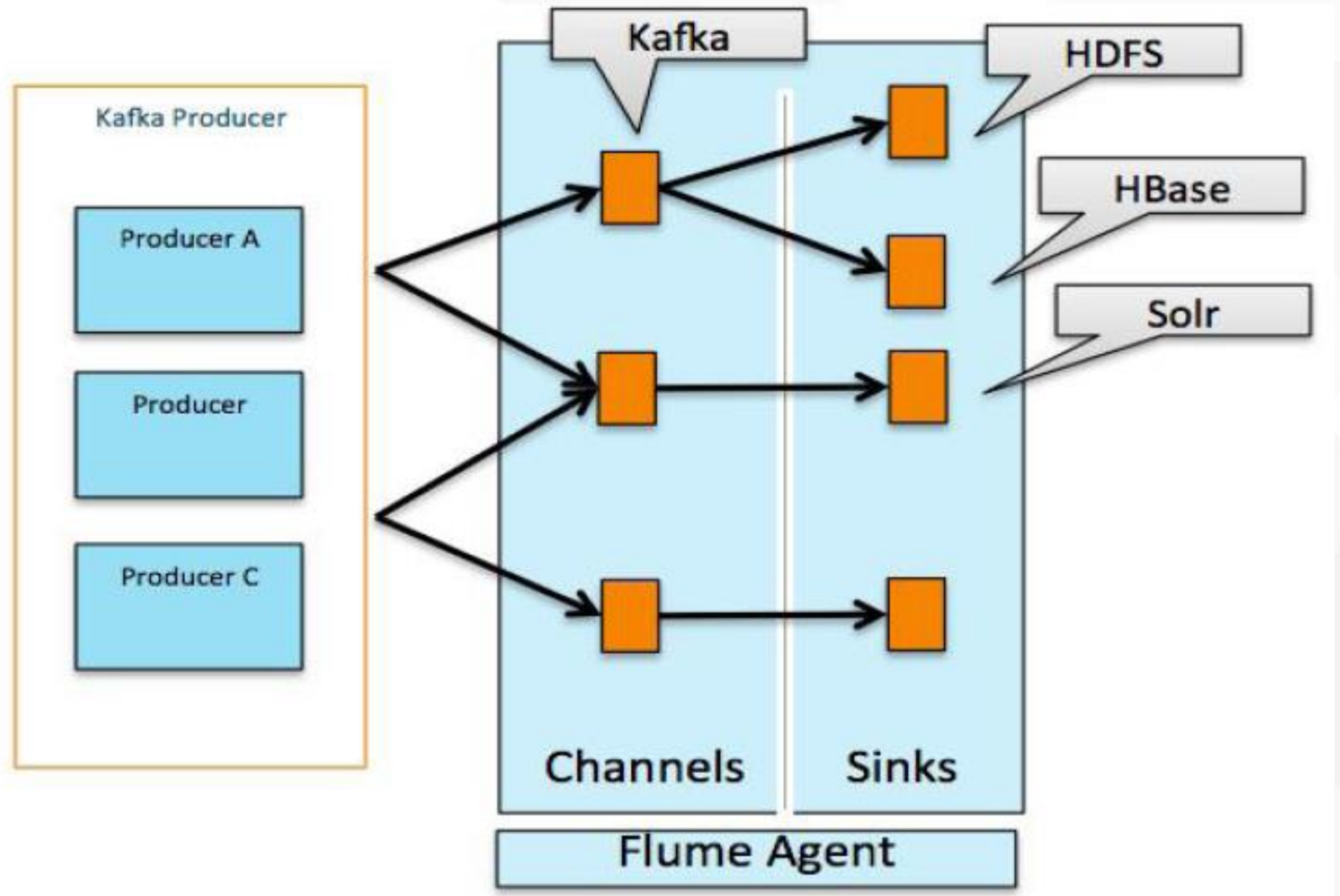
Flume-Kafka Architecture



Flume-Kafka Architecture



Flume-Kafka Architecture



Contact Us

Visit us on: http://www.analytixlabs.in/

For more information, please contact us: http://www.analytixlabs.co.in/contact-us/

Or email: info@analytixlabs.co.in

Call us we would love to speak with you: (+91) 9910509849

Join us on:

Twitter - http://twitter.com/#!/AnalytixLabs

Facebook - http://www.facebook.com/analytixlabs

LinkedIn - http://www.linkedin.com/in/analytixlabs

Blog - http://www.analytixlabs.co.in/category/blog/