upGrad

Data Ingestion with Apache Sqoop and Apache Flume - Session 4

upGrad

Course: DS - DE

Lecture On: Apache Flume

Instructor: Hitesh Hasija

upGrad

Segment - 01
Session Overview

Session Overview

upGrad

1. Introduction to Flume and its components
2. Characteristics and use cases of Flume
3. A Log collection case study with installation on Amazon EMR Instance
4. Flume configuration files and Flume flows
5. Tuning Flume and Sqoop vs Flume

upGrad

Segment - 02
Introduction to Apache Flume

Learning Objectives

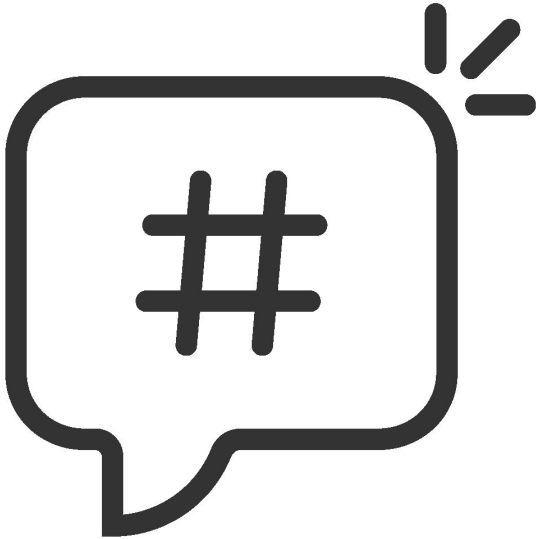
upGrad

1. Brief introduction to Apache Flume
2. Unstructured data and challenges while ingesting it

upGrad

Examples of Unstructured Data

Log Data



Streaming Data: Twitter Streams

Introduction to Apache Flume

upGrad

According to Flume user guide,

‘Apache Flume is a distributed, reliable and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralised data store.’

Introduction to Apache Flume

upGrad

Typically, there are many Log generation servers that create logs rapidly.

How to ingest these types of data into HDFS reliably?

Challenges in the Ingestion of unstructured Data

Network may get overwhelmed

Latency issues

Network traffic accounting

A Flume agent is a Java application that generates or receives data and buffers it until it is written to the next agent or a storage system.

Segment Summary

upGrad

1. Introduced Apache Flume in brief
2. Discussed the challenges faced while ingesting unstructured data

upGrad

Segment - 03
Components of Flume

Learning Objectives

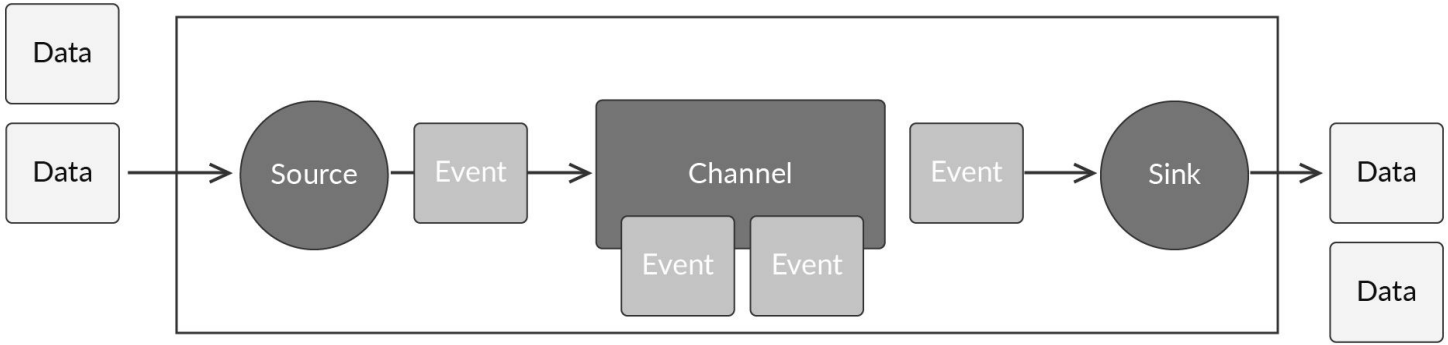
upGrad

1. Introduction to various components of Flume
2. Description of each component with examples

Components of Flume

upGrad

1. **Events**
2. **Source**
3. **Channel**
4. **Sink**



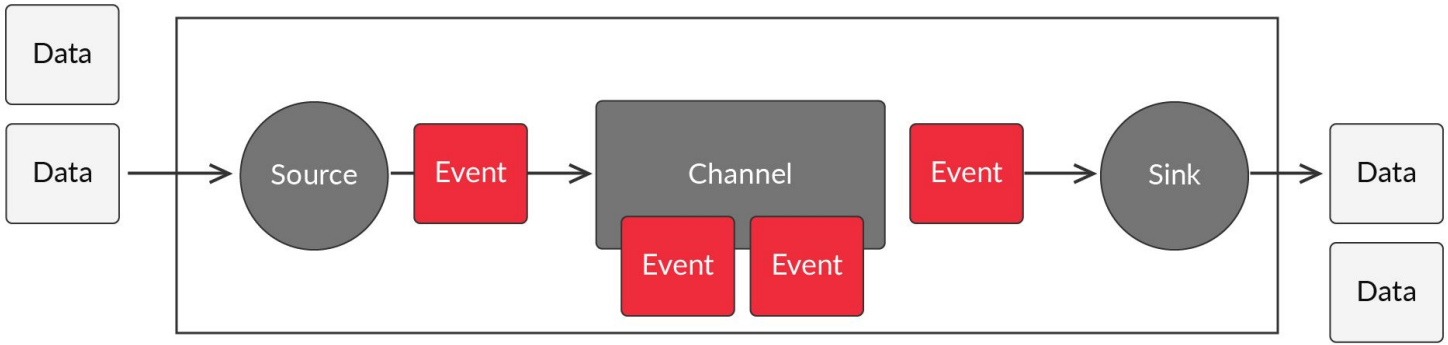
Components of Flume

upGrad

Events

- The two main components are as follows:
  - Header: key-value pairs used to show routing information or other structured information
  - Body: Contains the actual data in an array of bytes

date = 16:04:2020:17:04:31 server = node1.example.com	My Server has started……..... running at port 3419

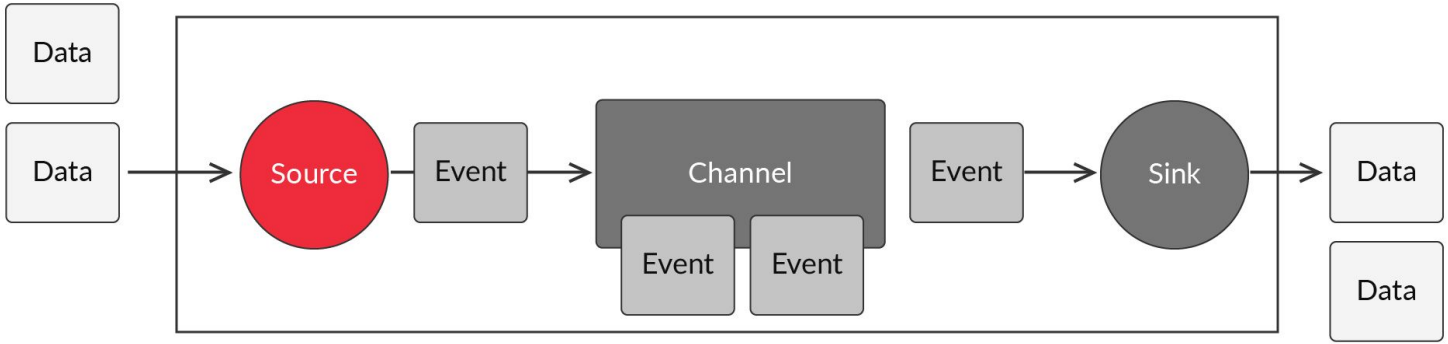


Components of Flume

upGrad

2 Source

- Is responsible for receiving data from any application that produces data by listening to a port or the file system
- Every source is connected to at least one channel.
- Writes data in the form of events to channels

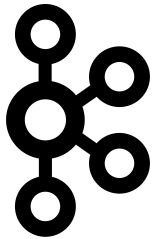


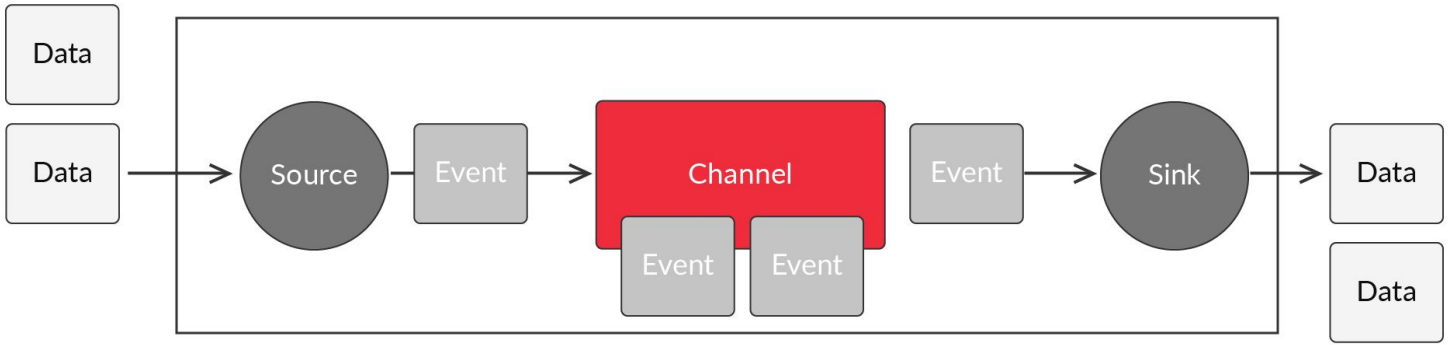
Components of Flume

upGrad

3 **Channel**

- Acts as a conduit between sources and sinks
- Is a buffer that keeps events till the sink writes them to the next hop or the target destination



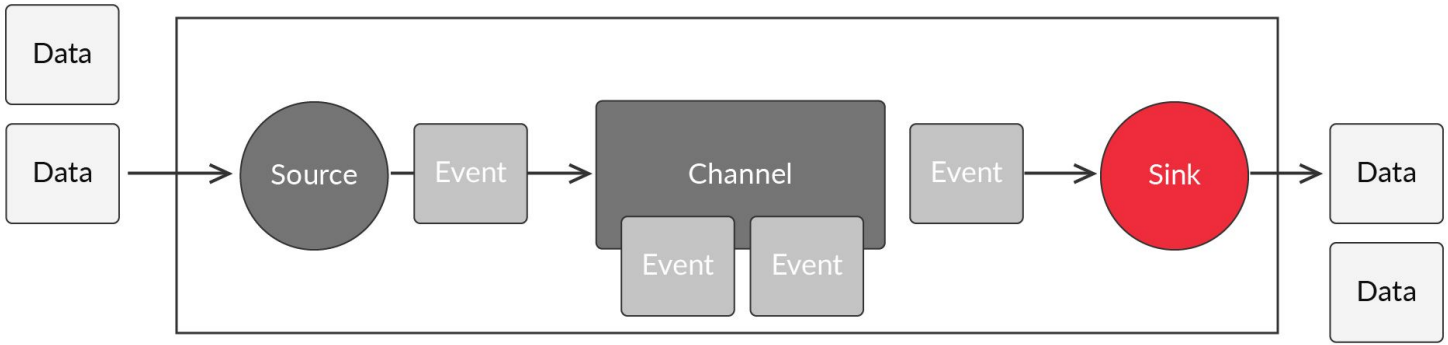


Components of Flume

upGrad

4 **Sink**

- Delivers data to the final destination such as HBase or HDFS
- Continuously polls the connected channel to retrieve events that are produced by the source and then write this to the next hop or the target destination accordingly
- After this, the channel is informed to remove the respective events.



Segment Summary

upGrad

1. Introduced the components of Apache Flume
2. Briefly described each component along with examples of each

upGrad

Segment - 05
Case Study: Log Collection

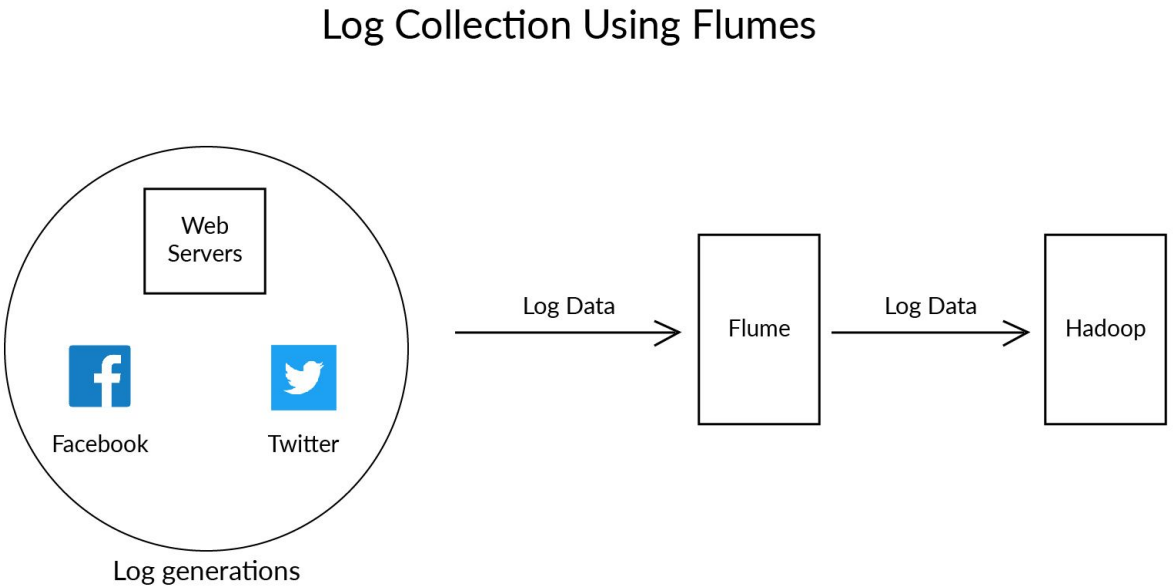
Learning Objectives

upGrad

1. Introduction to the Log collection case study
2. Overview of how the Data Ingestion process will be implemented

Case Study: Log Collection

upGrad



Segment Summary

upGrad

1. Introduced to the Log collection case study
2. Discussed how the case study will be implemented in brief

upGrad

Segment - 07
Flume Configuration Files

Learning Objectives

upGrad

1. Flume configuration files and demonstration of an example file
2. The template of a generic Flume configuration file

Flume Configuration Files

upGrad

```
#list sources, sinks and channels in the agent
hdfsagent.sources = avrosrc
hdfsagent.sinks = hdfssnk
hdfsagent.channels = hachnl
# define the flow
hdfsagent.sources.avrosrc.channels = hachnl
hdfsagent.sinks.hdfssnk.channel = hachnl
# source type properties
hdfsagent.sources.avrosrc.type = avro
hdfsagent.sources.avrosrc.bind = localhost
hdfsagent.sources.avrosrc.hostname = localhost
hdfsagent.sources.avrosrc.port = 12345
hdfsagent.sources.avrosrc.batch-size = 100
hdfsagent.sources.avrosrc.rollCount = 0
hdfsagent.sources.avrosrc.rollInterval = 0
hdfsagent.sources.avrosrc.rollSize = 100000
# sink type and properties
hdfsagent.sinks.hdfssnk.type = hdfs
hdfsagent.sinks.hdfssnk.hdfs.path = /test/flume/data
# channel type and properties
hdfsagent.channels.hachnl.type = memory
hdfsagent.channels.hachnl.capacity = 100000
hdfsagent.channels.hachnl.transactionCapacity = 100000
```

Flume Configuration Files

upGrad

```
#list sources, sinks and channels in the agent
hdfsagent.sources = avrosrc
hdfsagent.sinks = hdfssnk
hdfsagent.channels = hachnl
# define the flow
hdfsagent.sources.avrosrc.channels = hachnl
hdfsagent.sinks.hdfssnk.channel = hachnl
# source type properties
hdfsagent.sources.avrosrc.type = avro
hdfsagent.sources.avrosrc.bind = localhost
hdfsagent.sources.avrosrc.hostname = localhost
hdfsagent.sources.avrosrc.port = 12345
hdfsagent.sources.avrosrc.batch-size = 100
hdfsagent.sources.avrosrc.rollCount = 0
hdfsagent.sources.avrosrc.rollInterval = 0
hdfsagent.sources.avrosrc.rollSize = 100000
# sink type and properties
hdfsagent.sinks.hdfssnk.type = hdfs
hdfsagent.sinks.hdfssnk.hdfs.path = /test/flume/data
# channel type and properties
hdfsagent.channels.hachnl.type = memory
hdfsagent.channels.hachnl.capacity = 100000
hdfsagent.channels.hachnl.transactionCapacity = 100000
```

Flume Configuration Files

upGrad

```
#list sources, sinks and channels in the agent
hdfsagent.sources = avrosrc
hdfsagent.sinks = hdfssnk
hdfsagent.channels = hachnl
# define the flow
hdfsagent.sources.avrosrc.channels = hachnl
hdfsagent.sinks.hdfssnk.channel = hachnl
# source type properties
hdfsagent.sources.avrosrc.type = avro
hdfsagent.sources.avrosrc.bind = localhost
hdfsagent.sources.avrosrc.hostname = localhost
hdfsagent.sources.avrosrc.port = 12345
hdfsagent.sources.avrosrc.batch-size = 100
hdfsagent.sources.avrosrc.rollCount = 0
hdfsagent.sources.avrosrc.rollInterval = 0
hdfsagent.sources.avrosrc.rollSize = 100000
# sink type and properties
hdfsagent.sinks.hdfssnk.type = hdfs
hdfsagent.sinks.hdfssnk.hdfs.path = /test/flume/data
# channel type and properties
hdfsagent.channels.hachnl.type = memory
hdfsagent.channels.hachnl.capacity = 100000
hdfsagent.channels.hachnl.transactionCapacity = 100000
```

Flume Configuration Files

upGrad

```
#list sources, sinks and channels in the agent
hdfsagent.sources = avrosrc
hdfsagent.sinks = hdfssnk
hdfsagent.channels = hachnl
# define the flow
hdfsagent.sources.avrosrc.channels = hachnl
hdfsagent.sinks.hdfssnk.channel = hachnl
# source type properties
hdfsagent.sources.avrosrc.type = avro
hdfsagent.sources.avrosrc.bind = localhost
hdfsagent.sources.avrosrc.hostname = localhost
hdfsagent.sources.avrosrc.port = 12345
hdfsagent.sources.avrosrc.batch-size = 100
hdfsagent.sources.avrosrc.rollCount = 0
hdfsagent.sources.avrosrc.rollInterval = 0
hdfsagent.sources.avrosrc.rollSize = 100000
# sink type and properties
hdfsagent.sinks.hdfssnk.type = hdfs
hdfsagent.sinks.hdfssnk.hdfs.path = /test/flume/data
# channel type and properties
hdfsagent.channels.hachnl.type = memory
hdfsagent.channels.hachnl.capacity = 100000
hdfsagent.channels.hachnl.transactionCapacity = 100000
```

Flume Configuration Files

upGrad

```
#list sources, sinks and channels in the agent
hdfsagent.sources = avrosrc
hdfsagent.sinks = hdfssnk
hdfsagent.channels = hachnl
# define the flow
hdfsagent.sources.avrosrc.channels = hachnl
hdfsagent.sinks.hdfssnk.channel = hachnl
# source type properties
hdfsagent.sources.avrosrc.type = avro
hdfsagent.sources.avrosrc.bind = localhost
hdfsagent.sources.avrosrc.hostname = localhost
hdfsagent.sources.avrosrc.port = 12345
hdfsagent.sources.avrosrc.batch-size = 100
hdfsagent.sources.avrosrc.rollCount = 0
hdfsagent.sources.avrosrc.rollInterval = 0
hdfsagent.sources.avrosrc.rollSize = 100000
# sink type and properties
hdfsagent.sinks.hdfssnk.type = hdfs
hdfsagent.sinks.hdfssnk.hdfs.path = /test/flume/data
# channel type and properties
hdfsagent.channels.hachnl.type = memory
hdfsagent.channels.hachnl.capacity = 100000
hdfsagent.channels.hachnl.transactionCapacity = 100000
```

Flume Configuration Files

upGrad

The generic template of a Flume configuration file is as follows:

```
#list sources, sinks and channels in the agent
<Agent>.sources = <Source>
<Agent>.sinks = <Sink>
<Agent>.channels = <Channel1> <Channel2>

# define the flow
<Agent>.sources.<Source>.channels = <Channel1> <Channel2>
<Agent>.sinks.<Sink>.channel = <Channel1>

# source properties
<Agent>.sources.<Source>.<someProperty> = <someValue>

# sink properties
<Agent>.sinks.<Sink>.<someProperty> = <someValue>

# channel properties
<Agent>.channels.<Channel>.<someProperty> = <someValue>
```

Segment Summary

upGrad

1. Discussed Flume configuration files with an example
2. Discussed the generic template of a Flume configuration file

upGrad

Segment - 08
Flume Flows

Learning Objectives

upGrad

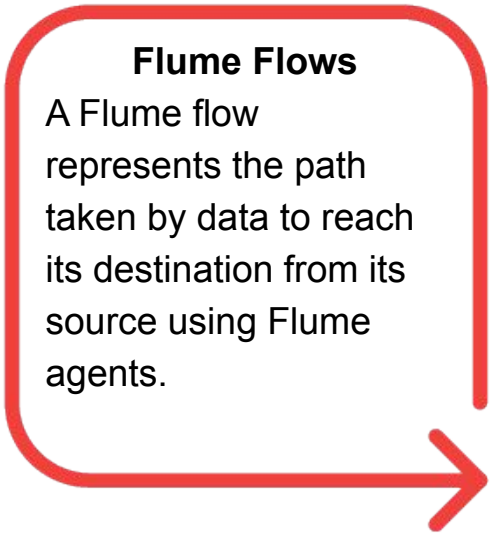
1. Introduction to Flume flows and the different types of flows
2. The Tiered Data Collection flow to be implemented in the case study

Flume Flows

upGrad

Flume Flows

A Flume flow represents the path taken by data to reach its destination from its source using Flume agents.



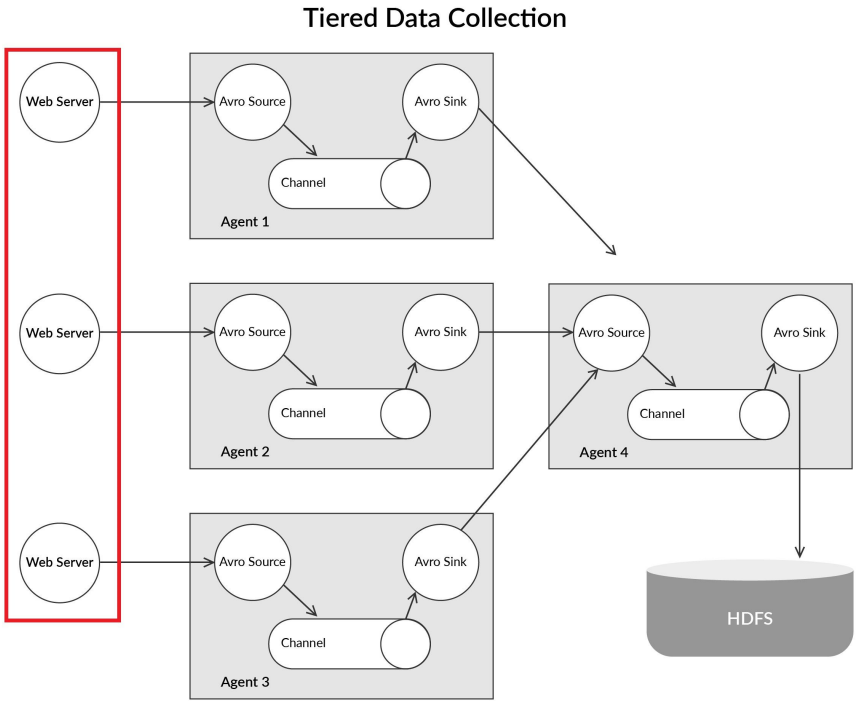
There are many different types of Flume flows

- Multi-agent flow: More than one Flume agent is used: can be set-up in a series configuration.
- Fan-out Flow: Different channels are connected to the same source
- Tiered Data Collection flow: Take data from the initial sources, and consolidate it to fewer agents that then will then finally dump it to the final sink.

Flume Flows

upGrad

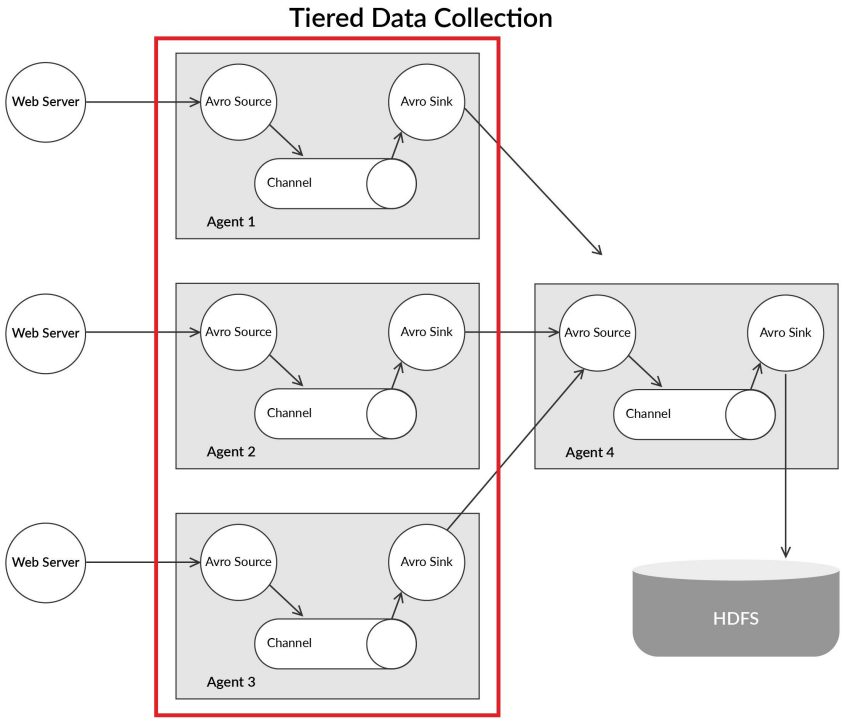
Why did we use the Tiered Data Collection flow in the Log collection case study?



Flume Flows

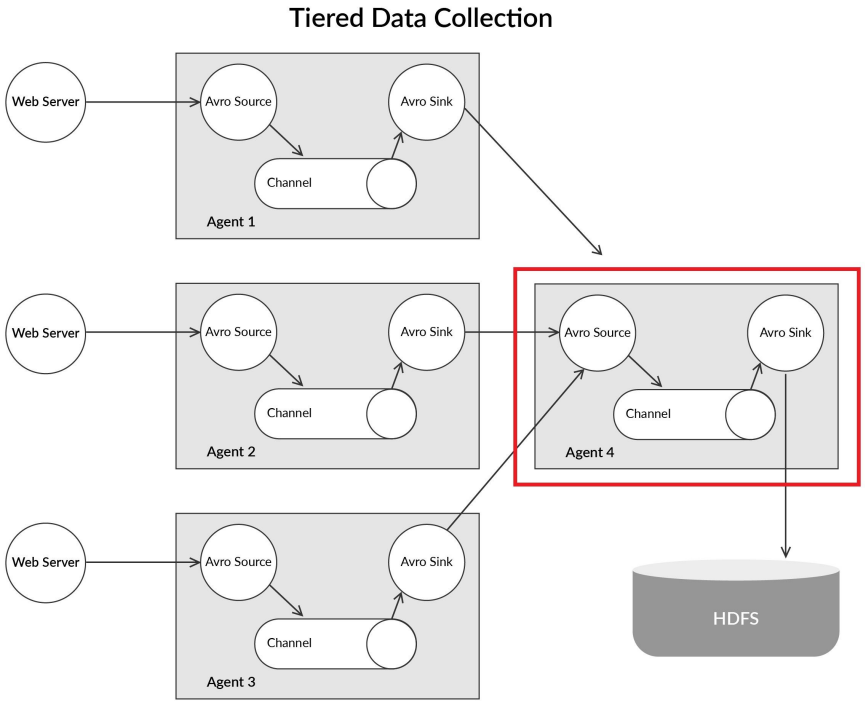
upGrad

Why did we use the Tiered Data Collection flow in the Log collection case study?



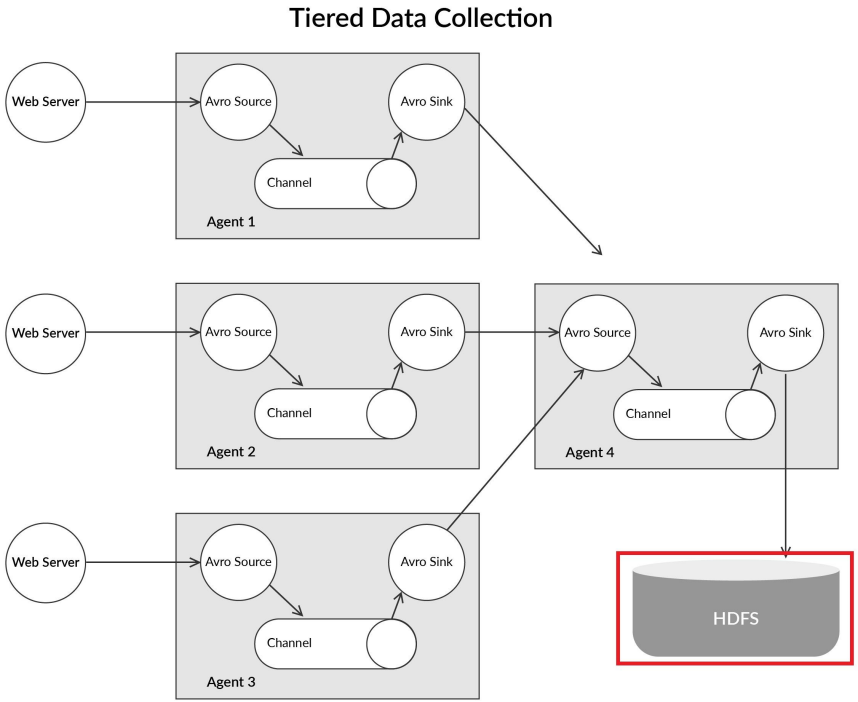
Flume Flows

Why did we use the Tiered Data Collection flow in the Log collection case study?



Flume Flows

Why did we use the Tiered Data Collection flow in the Log collection case study?



Segment Summary

upGrad

1. Introduced to Flume flows along with the different types of flows
2. Discussed Tiered Data collection flows to be used in the case study

upGrad

Segment - 11
Sqoop vs Flume

Learning Objectives

upGrad

1. Comparison of Flume and Sqoop
2. How should these two be used in different scenarios?

Sqoop vs Flume

upGrad

Sqoop

1. Sqoop is used to import data from RDBMS and NoSQL databases to big data ecosystem and to export it back.

Flume

1. Flume is used to collect, aggregate and transport large amounts of real-time data from various sources to a centralised place, where the data can be processed.

Sqoop vs Flume

upGrad

Sqoop

1. Sqoop is used to import data from RDBMS and NoSQL databases to big data ecosystem and to export it back.

2. Sqoop is not event-driven and is the most suitable if data is available in Oracle, MySQL or any other JDBC-compatible database.

Flume

1. Flume is used to collect, aggregate and transport large amounts of real-time data from various sources to a centralised place, where the data can be processed.

2. Flume is event-driven and is best suited for moving bulk stream data from sources such as spooling directories, tweets generated on Twitter, and the log files of a web server.

Sqoop vs Flume

upGrad

Sqoop

3. Sqoop has a connector-based architecture, wherein the connector is primarily responsible for connecting with the data sources and fetching data.

Flume

3. Flume has an agent-based architecture, wherein the Flume agent is responsible for data transfer.

Sqoop vs Flume

upGrad

Sqoop

3. Sqoop has a connector-based architecture, wherein the connector is primarily responsible for connecting with the data sources and fetching data.

4. Sqoop can transfer data parallelly for better performance.

Flume

3. Flume has an agent-based architecture, wherein the Flume agent is responsible for the data transfer.

4. Flume scales horizontally, and multiple Flume agents can be configured to collect high volumes of data.

Segment Summary

upGrad

1. Compared Apache Sqoop with Apache Flume
2. Discussed how to use these in different scenarios

Session Summary

upGrad

1. Introduced Apache Flume and its components in brief
2. Discussed various use cases and characteristics of Apache Flume
3. Discussed a Log collection case study along with installation on Amazon EMR Instance
4. Discussed the concept of Flume configuration files and Flume flows
5. Discussed how we can tune Flume and comparison between Sqoop and Flume

upGrad

Thank You