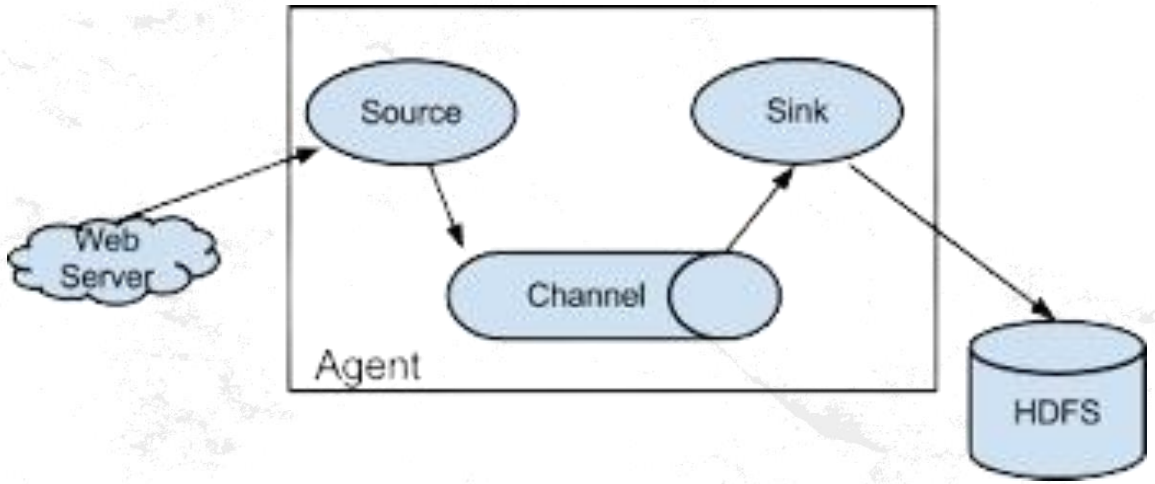


Data integration & ETL Tool: Apache Flume

NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

Apache Flume Overview

- **Apache Flume**:
  - A tool for **ingesting large amounts of streaming data** into Hadoop.
- **Features**:
  - Collects, aggregates, and moves logs or event data.
  - Integrates with HDFS and Kafka.
  - Supports real-time data pipelines.
- **Use Case**:
  - Collect log files from servers and store them in HDFS.

What is Apache Flume?

- **Definition**: Apache Flume is a distributed, reliable, and available service for collecting, aggregating, and moving large amounts of log data.
- **Primary Use**: Designed to collect streaming data from various sources and transport it to Hadoop's HDFS, HBase, or other storage systems.

Why Apache Flume?

- **High Throughput**: Handles large amounts of streaming data.
- **Scalable**: Easily scales to handle more data sources.
- **Reliable**: Ensures data is delivered without loss.
- **Flexible**: Can transport data to a variety of storage systems (HDFS, HBase, etc.).

Key Concepts of Flume

- **Source**: The data producers (e.g., log files, social media feeds).
- **Channel**: The medium that stores the data temporarily.
- **Sink**: The destination where the data is sent (e.g., HDFS, HBase).
- **Event**: A unit of data, typically consisting of a body and optional headers

How Apache Flume Works

- **Data is collected** by a **source** (e.g., tailing log files, HTTP requests).
- The data is temporarily stored in the **channel**.
- Data is processed by an **interceptor** (if any).
- Finally, the processed data is sent to the **sink** (e.g., HDFS).

Flume Architecture

- **Simple Flume Architecture**: A single agent with a source, channel, and sink.
- **Distributed Architecture**: Multiple agents working together to process large datasets.

Flume Data Flow Example

- **Source**: Tail a log file.
- **Channel**: Memory or File Channel.
- **Sink**: Store events in HDFS.

Flume Use Cases

- **Log Aggregation**: Collecting log data from multiple servers and sending it to HDFS or HBase.
- **Real-time Event Data**: Collecting and transporting real-time event data to a processing pipeline.
- **Social Media Streaming**: Ingesting social media data like tweets into Hadoop.

Flume Configuration Example

- Configuration File (flume.conf):

```
agent.sources = source1
agent.channels = memoryChannel
agent.sinks = sink1
agent.sources.source1.type = exec
agent.sources.source1.command = tail -F /var/log/mylogfile.log
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 10000
agent.sinks.sink1.type = hdfs
agent.sinks.sink1.hdfs.path = hdfs://localhost:9000/logs/
```

Advantages of Flume

- **Scalable and Fault Tolerant**: Can scale horizontally to meet large data demands.
- **Easy to Configure**: Simple text-based configuration files.
- **Extensible**: Can be extended with custom sources, sinks, and interceptors.

Real-World Applications

- **Web Logs Collection**: Collect logs from various web servers and send them to Hadoop for processing.
- **IoT Data Ingestion**: Collect sensor data and send it to HDFS for analytics.

Apache Flume

Introduction:

- Flume is a distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of streaming data into Hadoop.

Key Features:

- Optimized for unstructured and semi-structured data (e.g., logs, event streams).
- Supports multiple sources (e.g., web servers, log files) and destinations (e.g., HDFS, Hive).
- Reliability via transactional mechanisms for data flows.

What is Apache Flume?

- **Definition**:
  - Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data or streaming data into Hadoop's HDFS or other data stores.
- **Key Points**:
  - Designed for high-throughput and fault tolerance.
  - Primarily used in data ingestion pipelines for Hadoop ecosystems.

What is Flume?

- Reliable service for collection and aggregation of large amount of data.
  - Especially streaming data, for example Log data.
- Flume is one of the projects which comes into Hadoop framework.
- For log analysis based on Hadoop, Flume can be used to get the log information, such as logs from websites or system logs.
- Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data.
- It is commonly used to stream data from various sources into storage systems like HDFS.

Why Apache Flume?

- **Problem Solved**:
  - Flume simplifies data collection from various sources and routes it efficiently to data lakes like HDFS.
  - It is useful in handling large volumes of real-time streaming data, such as application logs and event data.
- **Features**:
  - Scalability: Easily scales to handle huge volumes of data.
  - Reliability: Ensures data delivery even in case of failures.
  - Flexibility: Integrates with different data sources and sinks.

Components of Apache Flume

1. **Sources**:
   1. Data ingestion points where Flume collects data.
   2. Examples: Syslog, SpoolDirectory, Avro, HTTP, etc.
2. **Channels**:
   1. Buffers the data between the Source and Sink.
   2. Examples: MemoryChannel, FileChannel.
3. **Sinks**:
   1. Destinations where data is written to (e.g., HDFS, HBase, etc.).
   2. Examples: HDFS Sink, Kafka Sink, ElasticSearch Sink.

How Flume Works?

1. **Data Collection**:
    1. Flume collects data from external sources using configurable sources.
2. **Data Storage**:
    1. The data is temporarily stored in channels, which act as buffers.
3. **Data Delivery**:
    1. Data is then delivered to specified sinks (HDFS, HBase, etc.).

Flume Use Cases

- **Real-time Log Aggregation**:
  - Flume is commonly used to collect and aggregate logs from multiple servers into HDFS for later processing.
- **Streaming Data Ingestion**:
  - It is used for moving large volumes of streaming data from web servers, sensors, or IoT devices to big data platforms like Hadoop or Kafka.
- **Data Movement to HDFS**:
  - Flume is frequently used to ingest large datasets into HDFS for further analysis by Hadoop-based applications.

Advantages of Apache Flume

- **Scalable**: Supports scaling out with multiple sources and sinks for increased throughput.
- **Fault-tolerant**: Handles failures gracefully by retrying data delivery.
- **Flexible**: Works with multiple data sources and sinks.
- **Extensible**: Flume can be extended to support custom sources, channels, and sinks.

Challenges in Using Flume

- **Complex Configuration**: Setting up complex pipelines with multiple sources and sinks may require careful configuration.
- **Resource Management**: Flume processes high volumes of data, so it may consume significant resources.
- **Data Latency**: Processing and delivering data to sinks could introduce latency in real-time systems.

Apache Flume Example

- **Goal**: Collect server logs and store them in HDFS.

1. **Set up**:
   1. Install Flume and configure an agent.
2. **Configuration File**:
   Example Flume agent.conf file:

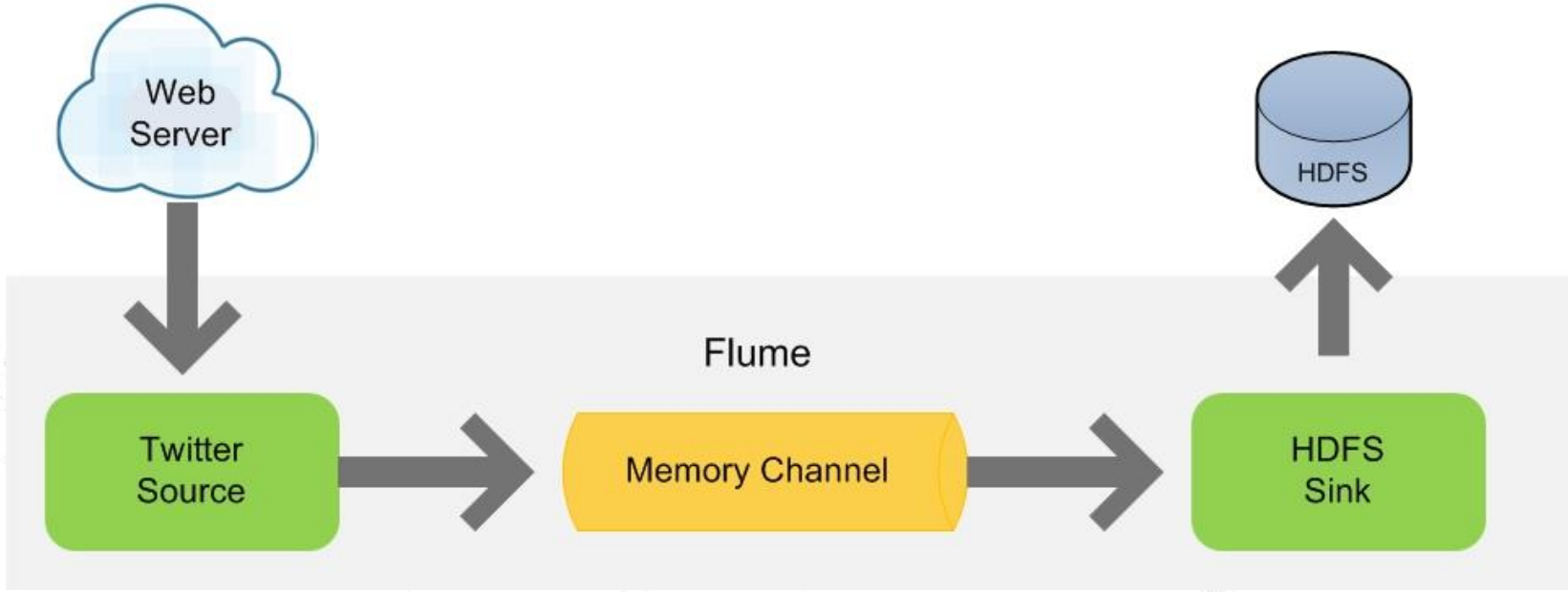
```
agent.sources = source1
agent.sinks = sink1
agent.channels = channel1
agent.sources.source1.type = exec
agent.sources.source1.command = tail -F
/var/log/syslog
agent.sinks.sink1.type = hdfs
agent.sinks.sink1.hdfs.path =
hdfs://localhost:9000/logs/
agent.sinks.sink1.hdfs.fileType =
DataStream
agent.channels.channel1.type = memory
agent.sources.source1.channels = channel1
agent.sinks.sink1.channel = channel1
```

3. **Run Flume**:

```
flume-ng agent --conf ./conf --conf-file
./agent.conf --name agent
```

Flume Agent

- Flume architecture or flume agent has source (anything like web server, application server or website etc.)
- From source, data moves to channel where our log data will be stored.
- From channel, the log data will be moved to sink (storage, for example Hadoop, or local file system etc.)



Flume Components

- Source
  - An active component which receives the event and places it in the channel.
- Channel
  - A passive component which buffers the event and sends it to the sink,
- Sink
  - Writes the data into next hop for final destination.

Conf File

- Basic Rules
  - Every agent must have at least one channel.
  - Every source must have at least one channel.
  - Every sink must have exactly one channel.
  - Every component must have a type.

Example Configuration File Sample

```
demo.sources = source1
demo.sinks = sink1
demo.channels = channel1

# Define the source
demo.sources.source1.type = netcat
demo.sources.source1.bind = localhost
demo.sources.source1.port = 44444

# Define the sink (HDFS)
demo.sinks.sink1.type = hdfs
demo.sinks.sink1.hdfs.path = hdfs://localhost:9000/user/flume/demo
demo.sinks.sink1.hdfs.fileType = DataStream
# Define the channel
demo.channels.channel1.type = memory
demo.channels.channel1.capacity = 1000
demo.channels.channel1.transactionCapacity = 100

# Bind the source and sink to the channel
demo.sources.source1.channels = channel1
demo.sinks.sink1.channel = channel1
```

Prerequisites

- Apache Flume installed on your system.
- A running Hadoop cluster (HDFS configured).
- Netcat (nc) utility to send data to the Flume agent.
- Docker (optional) if running in containers.

Step 1 - Install Apache Flume

- **Download Apache Flume:bash**

wget https://archive.apache.org/dist/flume/1.9.0/apache-flume-1.9.0-bin.tar.gz

- **Extract the tarball:bash**

- *tar -xvzf apache-flume-1.9.0-bin.tar.gz*

- **Move the extracted folder:bash**

- *mv apache-flume-1.9.0-bin /usr/local/flume*

Set Environment Variables

- **Add Flume to your PATH by editing the ~/.bashrc file:bash**

nano ~/.bashrc

- **Add the following lines:bash**

export FLUME_HOME=/usr/local/flume export PATH=$PATH:$FLUME_HOME/bin

- **Reload the environment variables:bash**

source ~/.bashrc

Verify Installation

- **Check Flume's version:bash**

flume-ng version

- **Ensure the output is the correct version.**

```
root@912591bccdc8:/# nano ~/.bashrc
root@912591bccdc8:/# source ~/.bashrc
root@912591bccdc8:/# flume-ng version
Flume 1.9.0
Source code repository: https://git-wip-us.apache.org/repos/asf/flume.git
Revision: d4fcab4f501d41597bc616921329a4339f73585e
Compiled by fszabo on Mon Dec 17 20:45:25 CET 2018
From source with checksum 35db629a3bda49d23e9b3690c80737f9
root@912591bccdc8:/#
```

Configure Flume Agent

```
demo.sources = source1
demo.sinks = sink1
demo.channels = channel1

# Define the source
demo.sources.source1.type = netcat
demo.sources.source1.bind = localhost
demo.sources.source1.port = 44444

# Define the sink (HDFS)
demo.sinks.sink1.type = hdfs
demo.sinks.sink1.hdfs.path = hdfs://localhost:9000/user/flume/demo
demo.sinks.sink1.hdfs.fileType = DataStream

# Define the channel
demo.channels.channel1.type = memory
demo.channels.channel1.capacity = 1000
demo.channels.channel1.transactionCapacity = 100

# Bind the source and sink to the channel
demo.sources.source1.channels = channel1
demo.sinks.sink1.channel = channel1
```

- **Replace localhost with your Hadoop Namenode hostname or IP address if necessary.**

Start Flume Agent

- **Run the Flume agent using the configuration file:bash**

- *flume-ng agent \ --conf /usr/local/flume/conf \ --conf-file /usr/local/flume/conf/demo-agent.conf \ --name demo \ -Dflume.root.logger=INFO,console*

Install Netcat in Container

- **Install Netcat in the container:bash in a new Shell Terminal**
- *apt-get update apt-get install netcat -y*
- **Verify the installation:bash**
- *nc -h*

Test Data Flow with Netcat

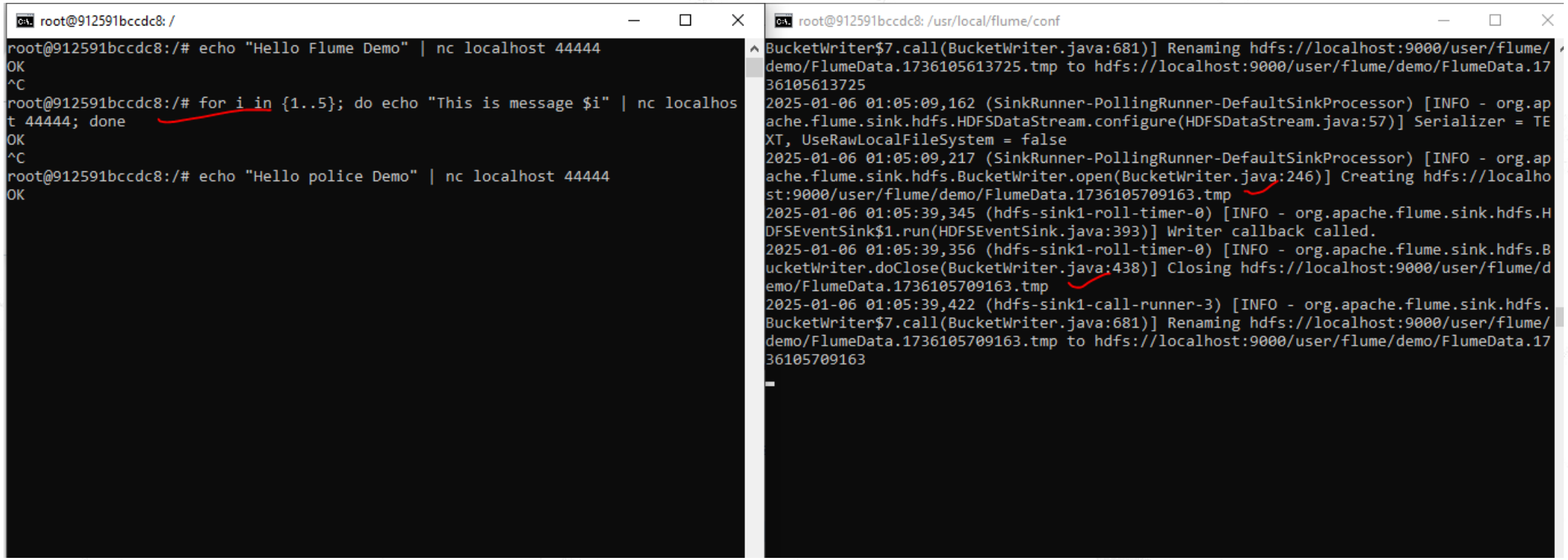
- **Open another terminal and send data to the Flume source using nc:bash**

- *echo "Hello Flume Demo" | nc localhost 44444*

- **Send multiple lines of data:bash**

- *for i in {1..5}; do echo "This is message $i" | nc localhost 44444; done*

Demo


root@912591bccdc8: /
root@912591bccdc8:/# echo "Hello Flume Demo" | nc localhost 44444
OK
^C
root@912591bccdc8:/# for i in {1..5}; do echo "This is message $i" | nc localhos
t 44444; done
OK
^C
root@912591bccdc8:/# echo "Hello police Demo" | nc localhost 44444
OK
root@912591bccdc8: /usr/local/flume/conf
BucketWriter$7.call(BucketWriter.java:681)] Renaming hdfs://localhost:9000/user/flume/
demo/FlumeData.1736105613725.tmp to hdfs://localhost:9000/user/flume/demo/FlumeData.17
36105613725
2025-01-06 01:05:09,162 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.ap
ache.flume.sink.hdfs.HDFSDataStream.configure(HDFSDataStream.java:57)] Serializer = TE
XT, UseRawLocalFileSystem = false
2025-01-06 01:05:09,217 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.ap
ache.flume.sink.hdfs.BucketWriter.open(BucketWriter.java:246)] Creating hdfs://localho
st:9000/user/flume/demo/FlumeData.1736105709163.tmp
2025-01-06 01:05:39,345 (hdfs-sink1-roll-timer-0) [INFO - org.apache.flume.sink.hdfs.H
DFSEventSink$1.run(HDFSEventSink.java:393)] Writer callback called.
2025-01-06 01:05:39,356 (hdfs-sink1-roll-timer-0) [INFO - org.apache.flume.sink.hdfs.B
ucketWriter.doClose(BucketWriter.java:438)] Closing hdfs://localhost:9000/user/flume/d
emo/FlumeData.1736105709163.tmp
2025-01-06 01:05:39,422 (hdfs-sink1-call-runner-3) [INFO - org.apache.flume.sink.hdfs.
BucketWriter$7.call(BucketWriter.java:681)] Renaming hdfs://localhost:9000/user/flume/
demo/FlumeData.1736105709163.tmp to hdfs://localhost:9000/user/flume/demo/FlumeData.17
36105709163

Verify Data in HDFS

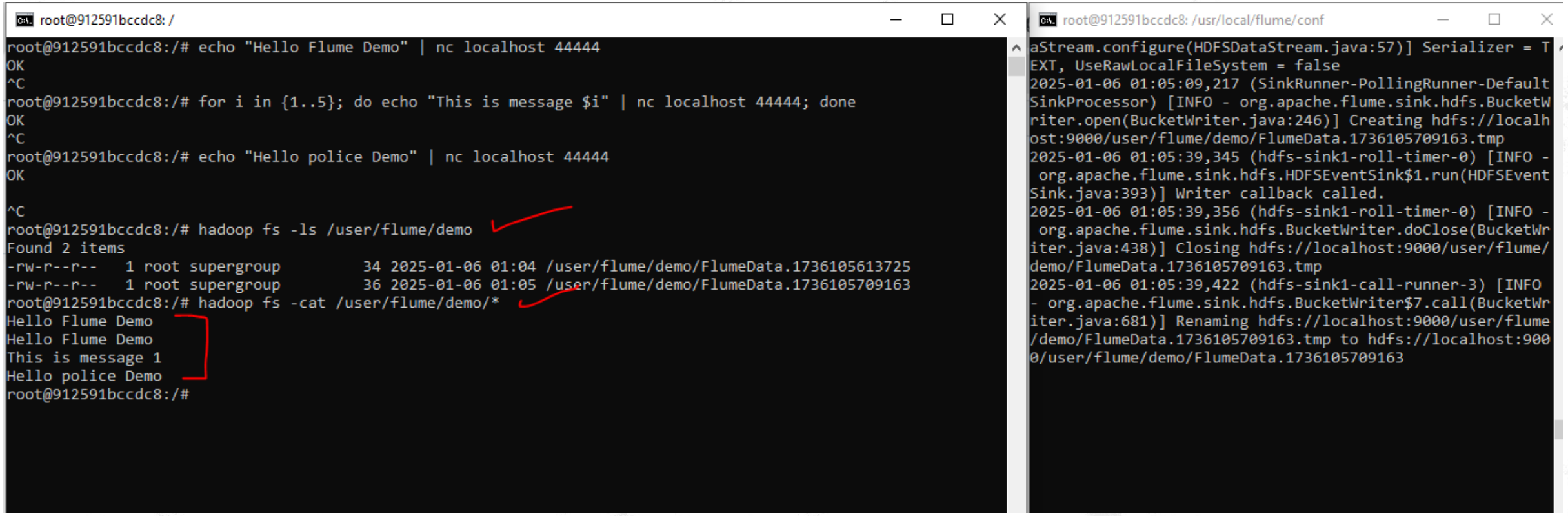
- **Check the HDFS directory where Flume writes data:bash**

- *hadoop fs -ls /user/flume/demo*

- **View the ingested data files:bash**

- *hadoop fs -cat /user/flume/demo/**

Viewing Data in HDFS



Future of Apache Flume

- **Integration with Emerging Technologies**:
  - Flume is evolving to integrate more easily with cloud-native and containerized applications.
- **Streaming Data Ecosystem**:
  - Flume continues to be a strong player in the big data streaming ecosystem, especially when paired with other tools like Apache Kafka and Apache Spark.

Conclusion

- **Summary**:
  - Apache Flume is an essential tool for efficiently collecting and transporting large amounts of data into big data systems like Hadoop.
  - Its scalability, flexibility, and fault tolerance make it a popular choice for data ingestion in the Hadoop ecosystem.
  - Apache Flume is a robust, flexible tool for collecting and moving large data sets. Its ease of use, scalability, and reliability make it ideal for integrating with Hadoop ecosystems.

Conclusion

- Apache Flume is a powerful tool for real-time data ingestion into HDFS.
- With these steps, you can configure Flume to collect and move log data into your Hadoop cluster.
- Troubleshoot common issues such as incorrect paths or missing utilities like Netcat.

APACHE FLUME INSTALLATION AND DEMONSTRATION