Big Data Hadoop and Spark Developer

simplilearn

Data Ingestion into Big Data Systems and ETL

Learning Objectives

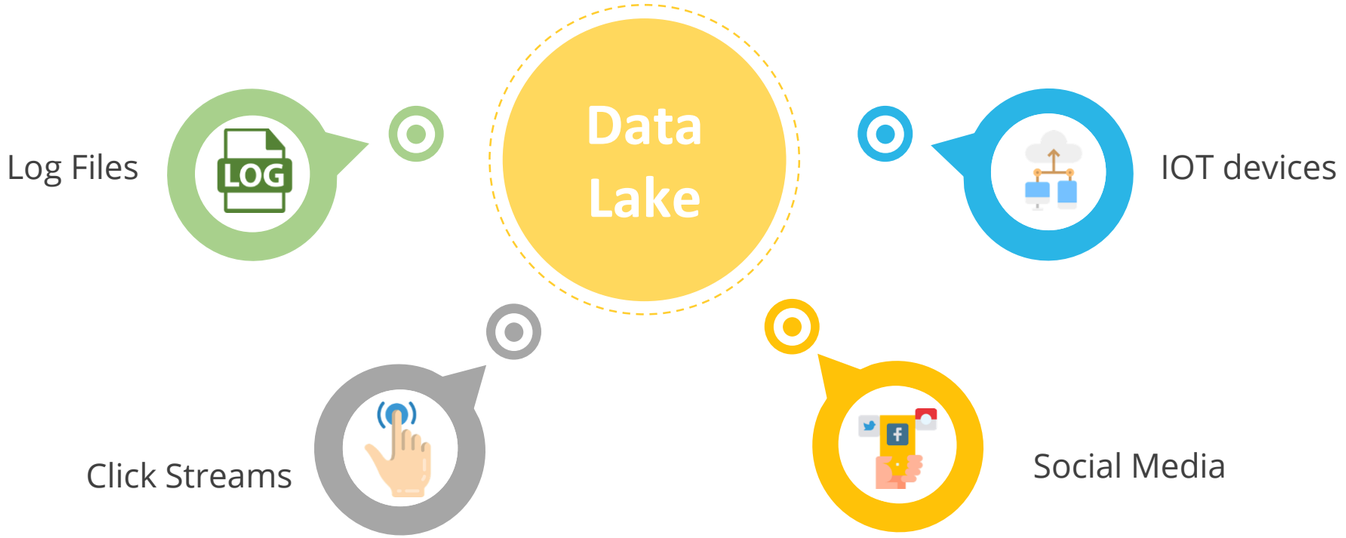
By the end of this lesson, you will be able to:

- Understand various Big Data Ingestion Tools
- Define Sqoop and its uses
- Analyze importing and exporting data from Hadoop using Sqoop
- Explain Apache Flume along with its uses
- Explain the components in the Flume architecture
- Define Kafka and its architecture

Data Ingestion Overview

Data Lake

A Data Lake is a centralized storage repository used to store large amounts of structured, semi-structured, and unstructured data.



Data sources include log files, data from click-streams, social media, and internet connected devices.

Data Lake vs. Data Warehouse

Characteristics	Data Lake	Data Warehouse
Data Type	It can be structured, semi-structured, and unstructured	Data organized into single schema; like tabular formats used in RDBMS
Data Quality	Any data that may or may not be curated (i.e. raw data)	Highly curated data that serves as the central version of the truth
Price and Performance	Low-cost storage	Expensive storage that has faster response times
User Support	Data scientists and data developers	Business analysts
Type of Analytics	Machine learning, predictive analytics, and data discovery	Batch reporting, BI, and visualizations

Data Ingestion

01

Big data ingestion involves transferring data, especially unstructured data from where it originated, into a system where it can be stored and analyzed such as Hadoop.

02

The ingestion process can be be continuous or asynchronous, real-time or batched, or both, depending upon the characteristics of the source and the destination.

03

In scenarios where the source and destination do not have the same data format or protocol, data transformation or conversion is done to make the data usable by the destination system.

Big Data Ingestion Tools

Choosing an appropriate data ingestion tool is important which in-turn is based on factors like data source, target, and transformations.

Apache Sqoop

Apache Flume

Apache Kafka

- Data ingestion tools provide users with a data ingestion framework that makes it easier to extract data from different types of sources and support a range of data transport protocols.
- Data ingestion tools also eliminate the need for manually coding individual data pipelines for every data source and accelerates data processing by helping you deliver data efficiently to ETL tools.

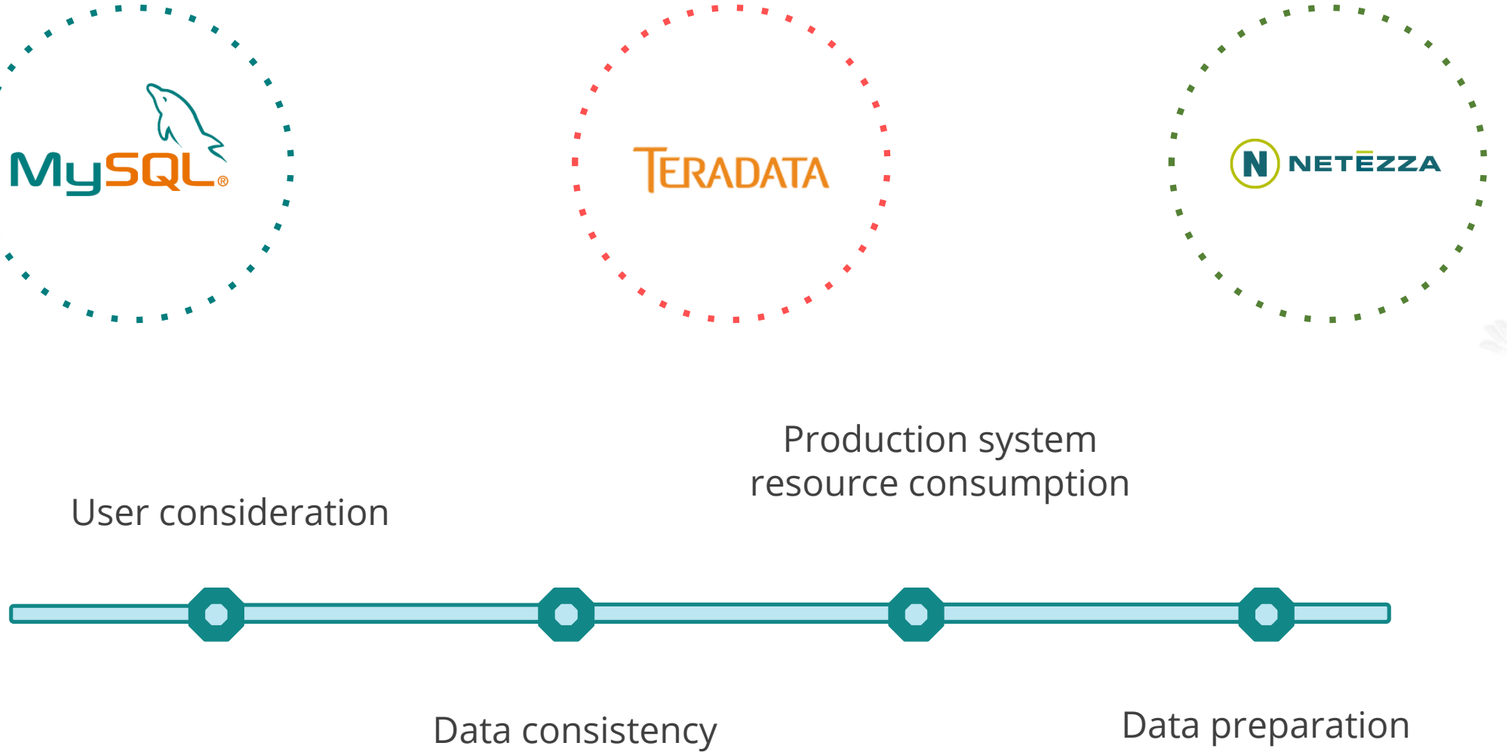
Apache Sqoop

What Is Sqoop?

- Sqoop, an Apache Hadoop Ecosystem project, is a command-line interface application for transferring data between relational databases and Hadoop.
- It supports incremental loads of a single table or a free-form SQL query.
- Imports can also be used to populate tables in Hive or HBase.
- Exports can be used to put data from Hadoop into a relational database.

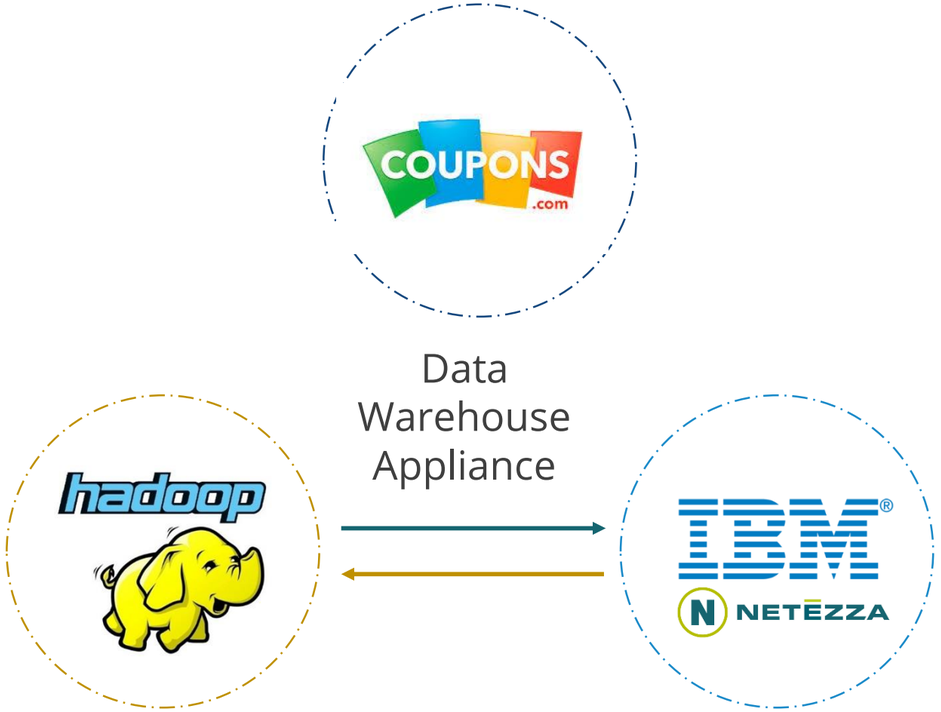
Why Sqoop?

While companies across industries were trying to move from structured relational databases to Hadoop, there were concerns about the ease of transitioning existing databases.



Real Life Use Cases

Online marketer Coupon.com uses Sqoop to exchange data between **Hadoop and the IBM Netezza data warehouse appliance**.



The Apollo group, an education company, also uses Sqoop to extract data from databases as well as to inject the results from Hadoop Jobs back into relational databases.

Sqoop and Its Uses

Sqoop is an Apache Hadoop Ecosystem project whose responsibility is to import or export operations across relational databases. The reasons for using Sqoop are as follows:

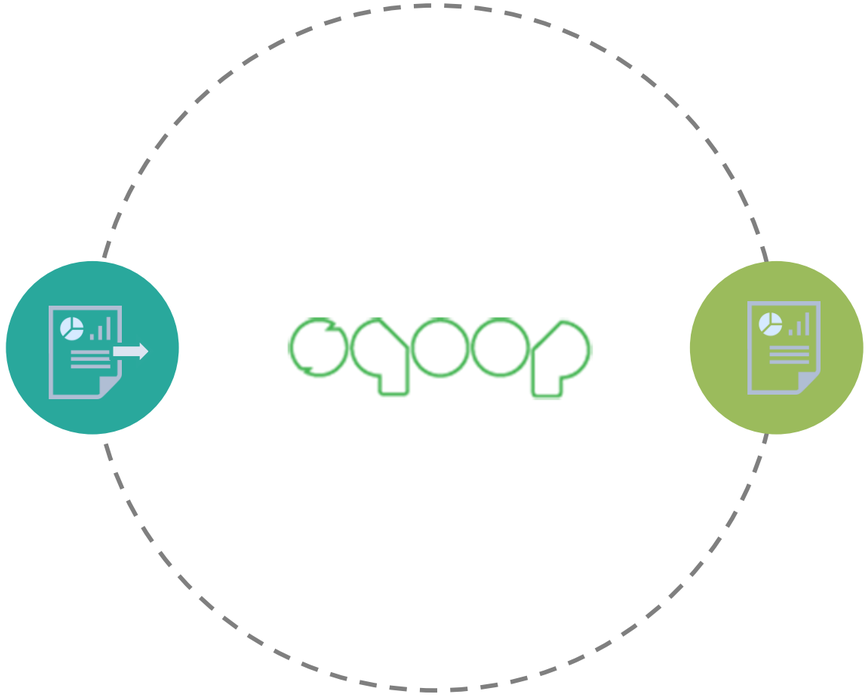
- SQL servers are deployed worldwide
- Nightly processing is done on SQL servers
- Allows to move certain parts of data from traditional SQL DB to Hadoop
- Transfers data efficiently and swiftly
- Handles large data through ecosystem
- Brings processed data from Hadoop to the applications

Sqoop and Its Uses

Sqoop is required when a database is imported from a Relational Database (RDB) to Hadoop or vice versa.

While exporting database from RDB to Hadoop:

Users must consider consistency of data, consumption of production system resources, and preparation of data for provisioning downstream pipeline.

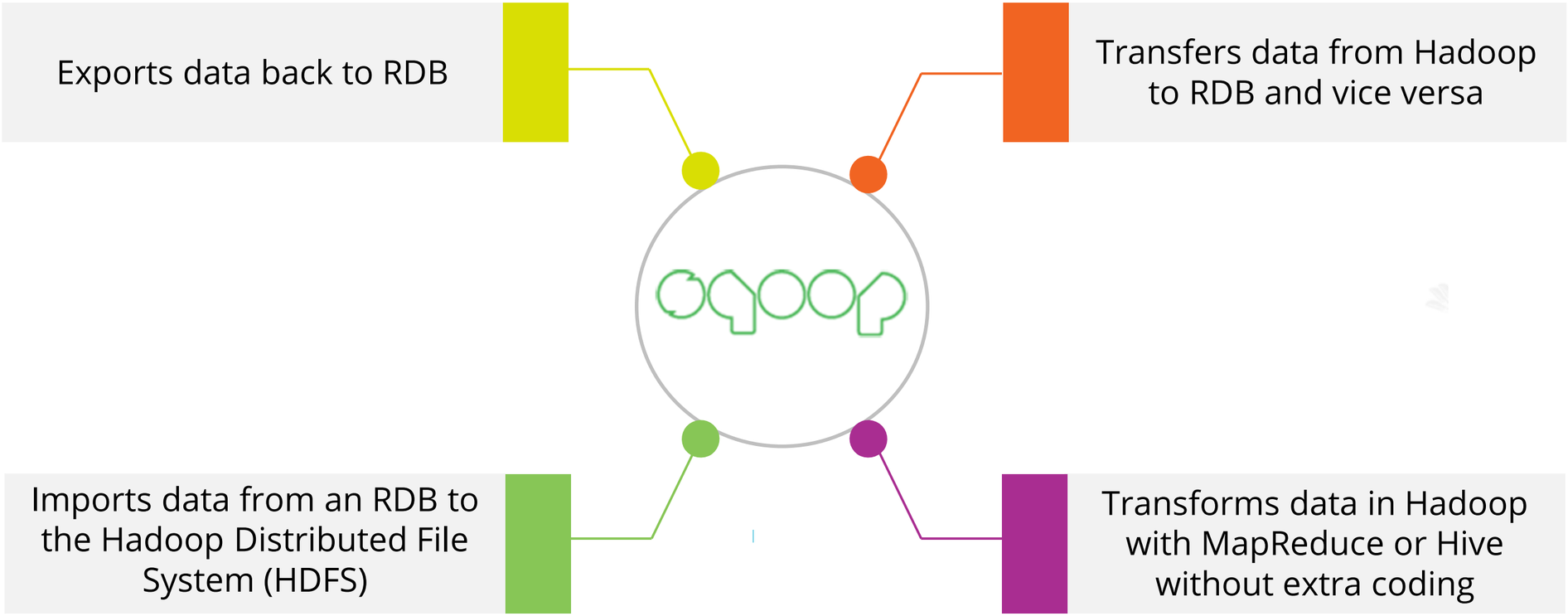


While importing database from Hadoop to RDB:

Users must keep in mind that directly accessing data residing in external systems, within a MapReduce framework, complicates applications. It exposes the production system to excessive load originating from cluster nodes.

Benefits of Sqoop

The following are the benefits of using Sqoop:



Sqoop Processing

Sqoop Processing

The processing of Sqoop can be summarized as follows:

- It runs in the Hadoop Cluster.
- It imports data from RDB or NoSQL DB to Hadoop.
- It has access to the Hadoop core, which helps in using mappers to slice the incoming data into unstructured formats and place the data in HDFS.
- It exports data back into the RDB, ensuring that the schema of the data in the database is maintained.

Sqoop Execution Process

This is a summary of how Sqoop performs the execution.

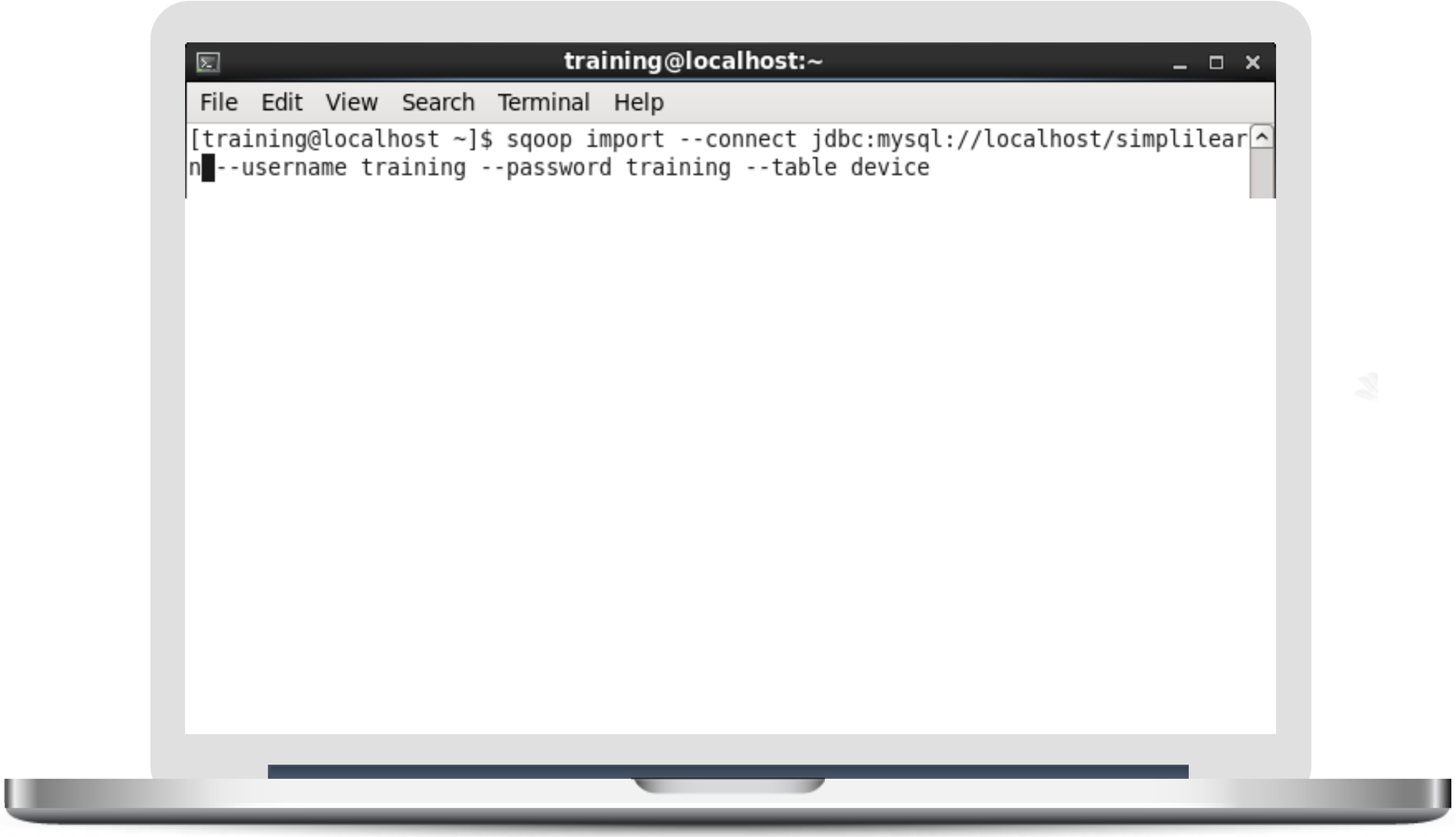
A map-only job is launched with individual mappers responsible for transferring a slice of the dataset.

The dataset being transferred is divided into partitions.

Each record of the data is handled in a type-safe manner.

Importing Data Using Sqoop

To import data present in MySQL database using Sqoop, use the following command:



Apache Sqoop

Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to list table of MySQL DB through Sqoop.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Sqoop Import Process

The process of Sqoop import is as follows:

Job submitted to cluster

A map-only Hadoop job is submitted to the cluster by Sqoop.

Gathering of metadata

Sqoop introspects the database to gather the necessary metadata for the data being imported.

Data is transferred

The map-only job performs data transfer using the metadata captured.

Sqoop Import Process

The imported data is saved in a directory on HDFS based on the table being imported.

Users can:

- Specify any alternative directory where the files should be populated
- Override the format in which data is copied by explicitly specifying the field separator and recording terminator characters
- Import data in Avro data format by specifying the option, as-avrodatafile, with the import command

Sqoop supports different data formats for importing data and provides several options for tuning the import operation.

Exporting Data from Hadoop Using Sqoop

Use the following command to export data from Hadoop using Sqoop:

Exporting Data from Hadoop Using Sqoop

Perform the following steps to export data from Hadoop using Sqoop:

1. Introspect the database for metadata and transfer the data
2. Transfer the data from HDFS to DB

- Sqoop divides the input dataset into splits.
- Sqoop uses individual map tasks to push the splits to the database.
- Each map task performs this transfer over many transactions to ensure optimal throughput and minimal resource utilization.

Sqoop Connectors

Sqoop Connectors

The different types of Sqoop connectors are:

Used to connect to any database that is accessible via JDBC

Generic JDBC connector

Fast-path connector

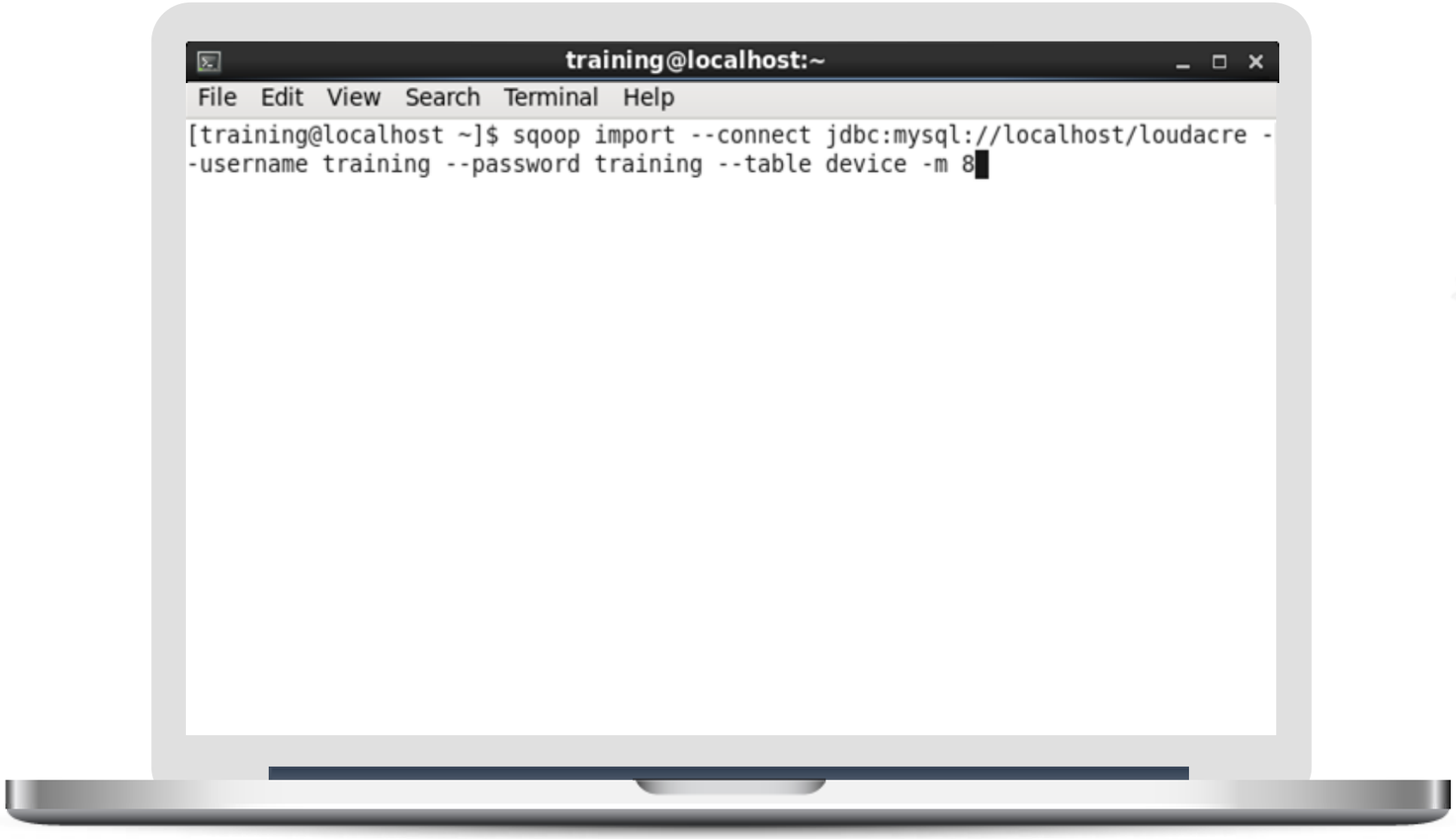
Specializes in using specific batch tools to transfer data with high throughput

Default Sqoop connector

Designed for specific databases such as MySQL, PostgreSQL, Oracle, SQL Server, and DB2

Controlling Parallelism

- By default, Sqoop typically imports data using four parallel tasks called mappers
- Increasing the number of tasks might improve import speed
- You can influence the number of tasks using the -m or --num-mappers option



Sample Sqoop Commands

training@localhost:~

File Edit View Search Terminal Help

```
$Sqoop import –driver com.mysql.jdbc.Driver –connect jdbc:mysql://localhost/a10 –username root –password root –table Sqoop_demo –target-dir /user/Sqoop_batch6 – m 1 –as-textfile

$Sqoop import –driver com.mysql.jdbc.Driver –connect jdbc:mysql://localhost/a10 –username root –password root –table Sqoop_demo –target-dir /user/Sqoop_batch6 – m 1 –as-textfile –where “id>2”

$Sqoop import –driver com.mysql.jdbc.Driver –connect jdbc:mysql://localhost/a10 –username root –password root –table Sqoop_demo –target-dir /user/Sqoop_batch6 –e “select * from Sqoop_demo where id =13” – m 1 –as-textfile
```

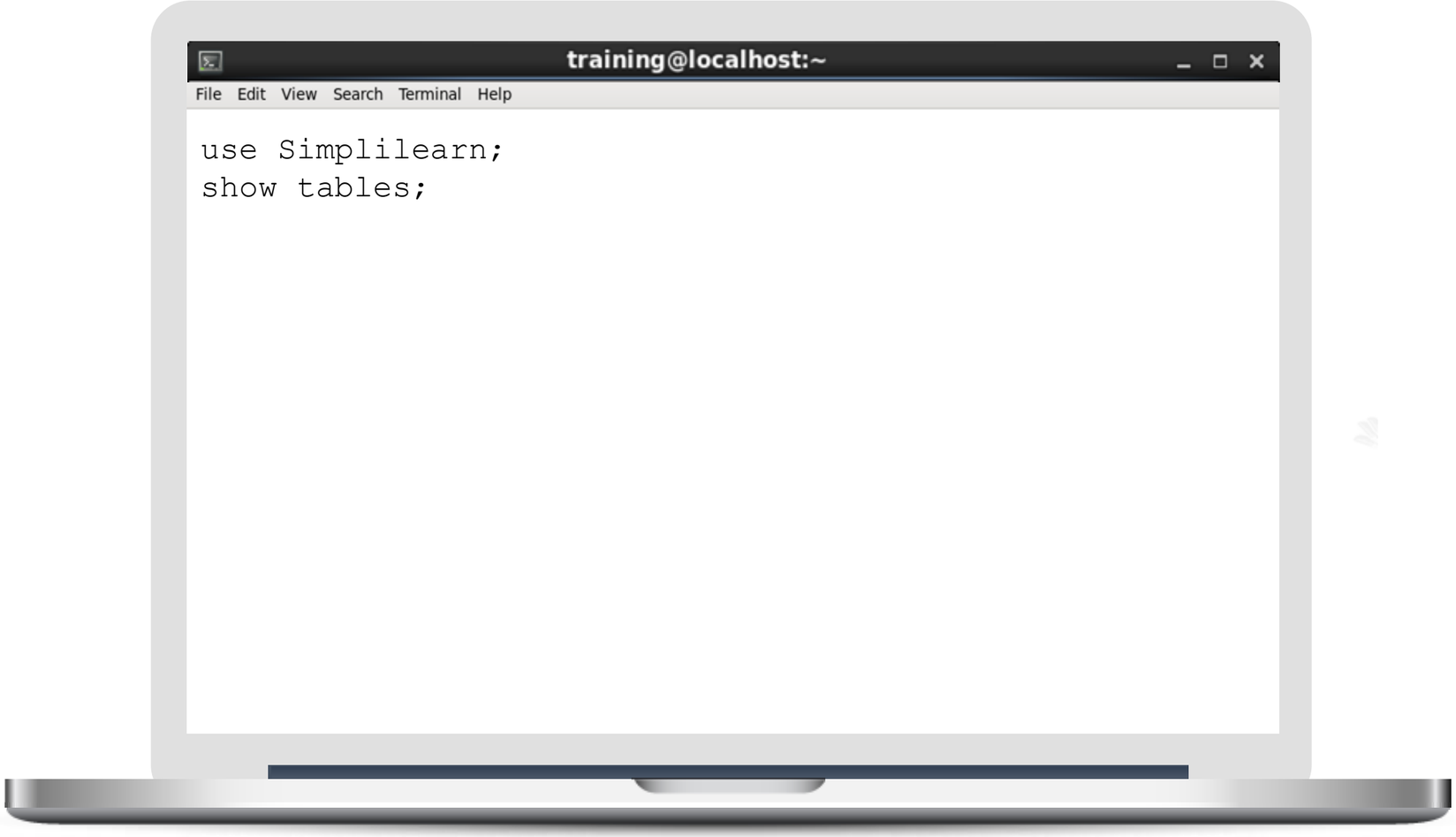
More sample Sqoop commands:

training@localhost:~

File Edit View Search Terminal Help

```
$Sqoop import –driver com.mysql.jdbc.Driver –connect jdbc:mysql://localhost/a10 –username root –password root –table Sqoop_demo –target-dir /user/Sqoop_batch6 – m 1 –as-textfile –split-by id

$Sqoop job –list

$Sqoop export–connect jdbc:mysql://localhost/a2 –username root –password root –table report1 –export-dir /user/hive/warehouse/mobile_vs_sentiments1/ --input-fields-terminated-by ‘\001’
```

Exploring a Database with Sqoop



Limitations of Sqoop

- Client-side architecture does impose some limitations in Sqoop:
  - Client must have JDBC drivers installed for connectivity with RDBMS
  - Requires connectivity to cluster from client
  - User has to specify username and password
  - It is difficult to integrate a CLI within external application

- Not best supported with NoSQL DB because it is tightly coupled with JDBC semantics.

Apache Sqoop

Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to use Sqoop commands to import and export data from MySQL to HDFS and vice-versa.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Apache Sqoop

Duration: 15 mins

Problem Statement: Using SQL and Sqoop commands, perform the below tasks:

- Create a database
- Create a table “employee” with the following fields: Id, Name, Salary, Department, and Designation
  Id is the primary key for the table
- Insert at least 5 records into the table
- Import the database and table into Sqoop
- Import only the records for which Salary is greater than 50000

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Steps to Perform

- **MySQL**

```
mysql -u labuser -p
Password : simplilearn

CREATE DATABASE userdb;
use userdb;

create table employee(Id INT NOT NULL, Name VARCHAR(100) NOT NULL, Salary INT NOT NULL,
Department VARCHAR(100) NOT NULL, Designation VARCHAR(100) NOT NULL, PRIMARY KEY(Id));

insert into employee values(201,"Peter",50000,"It","Developer");
insert into employee values(202,"Alice",60000,"Sales","Manager");
insert into employee values(203,"Jack",70000,"Operations","Director");
insert into employee values(205,"John",70000,"Support","Director");
```

Steps to Perform

- **Sqoop**

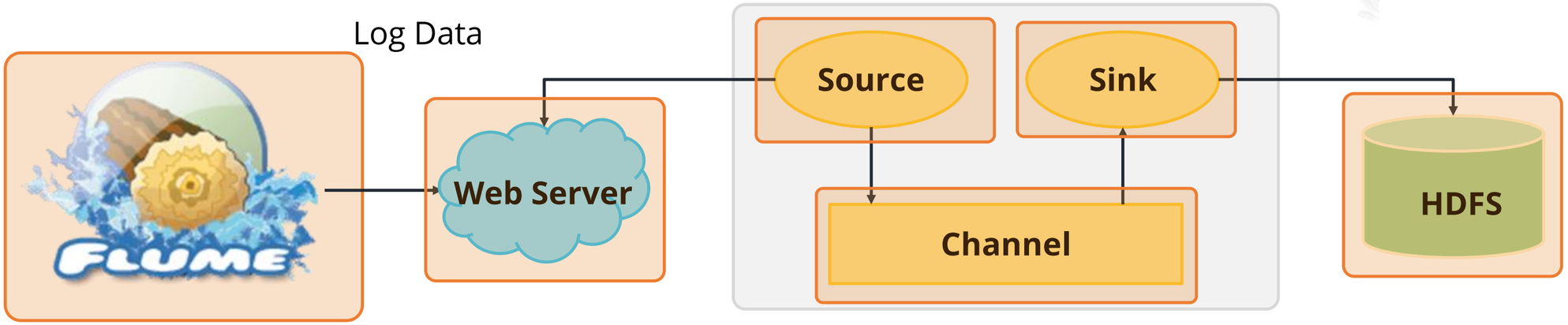
```
sqoop import --connect jdbc:mysql://ip-10-0-1-10.ec2.internal/userdb --username
labuser --password simplilearn --table employee –target -dir /user/simpli_learn/simpli --m 1

sqoop list-tables --connect jdbc:mysql://ip-10-0-1-10.ec2.internal/userdb --username labuser --password
simplilearn

sqoop import --connect jdbc:mysql://ip-10-0-1-10.ec2.internal/userdb --username labuser --password
simplilearn --table employee -m 1 --where "Salary > 50000" --target-dir '/user/simpli_learn/simpli123 -m 1
```

Apache Flume

What Is Apache Flume?

Apache Flume is a distributed and reliable service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS).

It has a simple and flexible architecture which is robust and fault-tolerant based on streaming data flows.



Why Flume?

Following is a business scenario in which Flume is beneficial:

Current scenario

Problem

Solution

A company has thousands of services running on different servers in a cluster that produce many large log data; these logs should be analyzed together.

Why Flume?

Following is a business scenario in which Flume is beneficial:

Current scenario

Problem

Solution

The current issue involves determining how to send the logs to a setup that has Hadoop. The channel or method used for the sending process must be reliable, scalable, extensive, and manageable.

Why Flume?

Following is a business scenario in which Flume is beneficial:

Current scenario

Problem

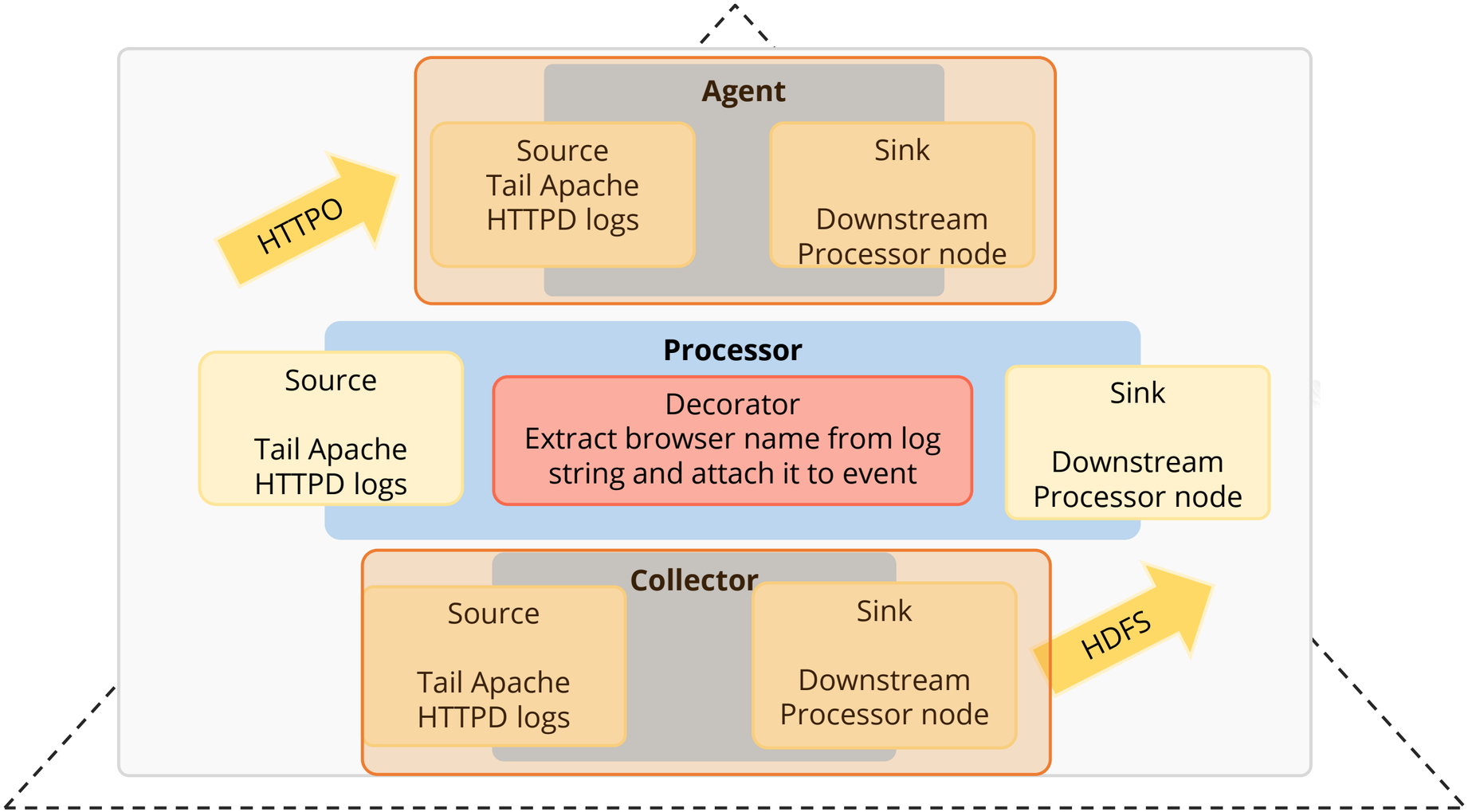
Solution

To solve this problem log aggregation tool called Flume can be used. Apache Sqoop and Flume are the tools that are used to gather data from different sources and load them into HDFS. Sqoop in Hadoop is used to extract structured data from databases like Teradata, Oracle, and so on, whereas Flume in Hadoop sources data that is stored in different sources, and deals with unstructured data.

Flume Model

Flume Model

The Flume Model comprises the following three entities:



Flume Goals

Flume aims to achieve the following goals:

- Ensure reliability by possessing tunable failure recovery modes
- Attain extensibility by using plug-in architecture for extending modules
- Achieve a scalable data path that can be used to form a topology of agents
- Create manageability by centralizing a data flow management interface

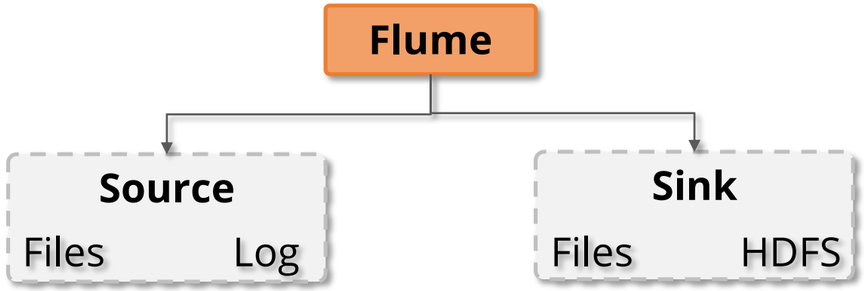
Extensibility in Flume

Extensibility:

- The ability to add new functionality to a system

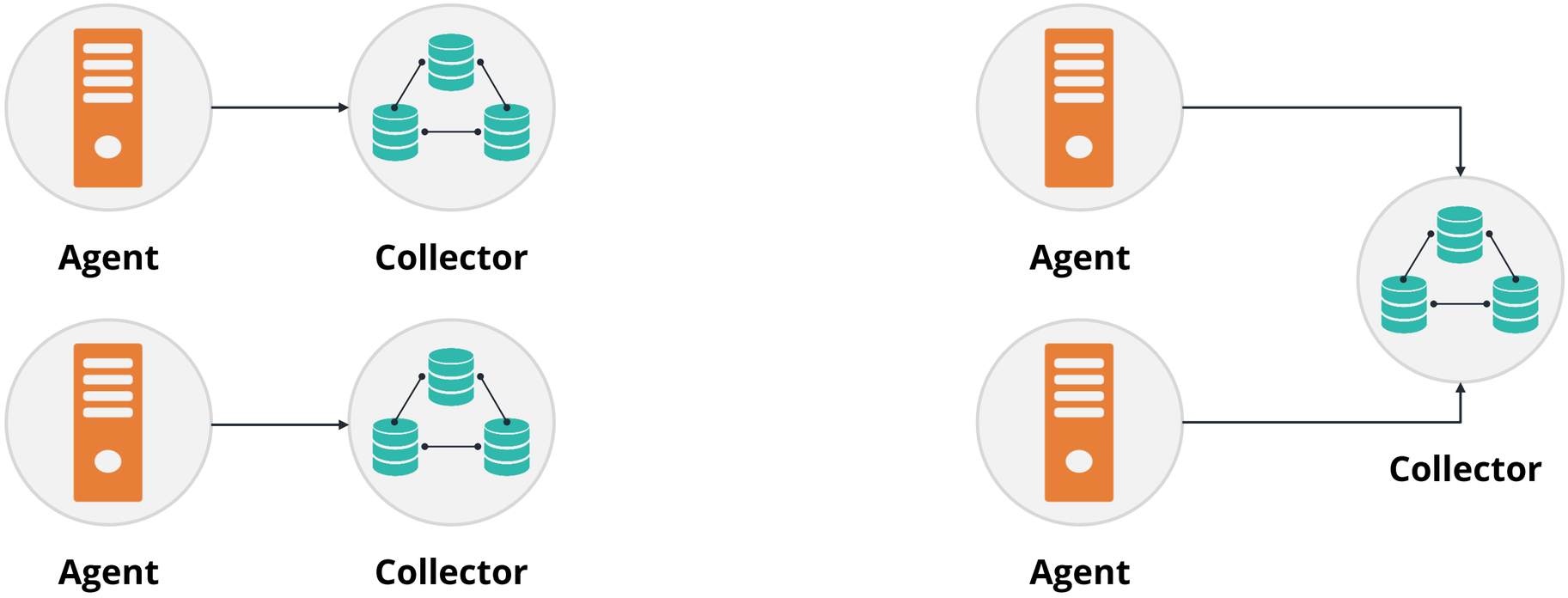
Flume can be extended by adding Sources and Sinks to existing storage layers or data platforms

- 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

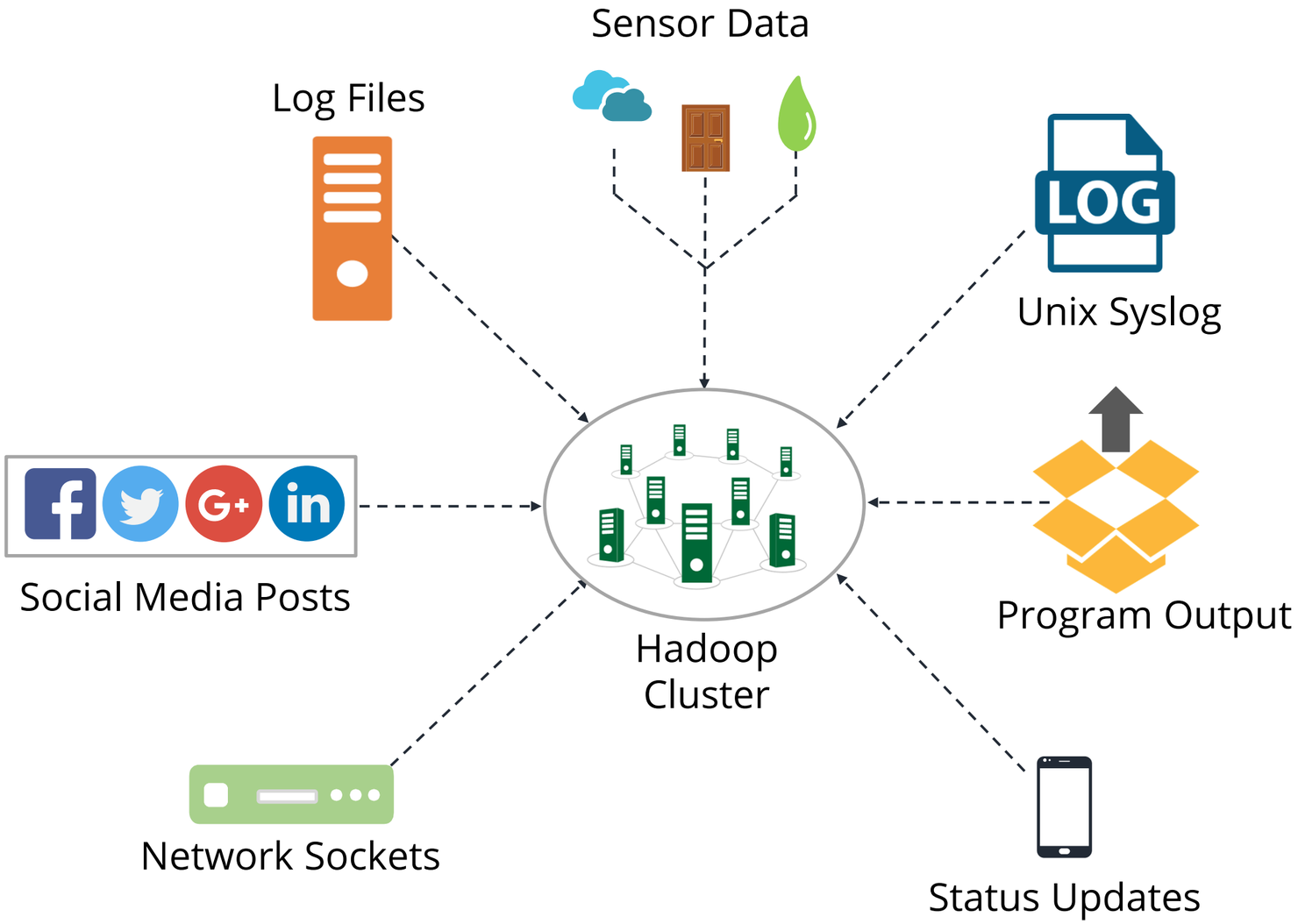


Scalability in Flume

Flume has a horizontally scalable data path which helps in achieving load balance in case of higher load in the production environment.

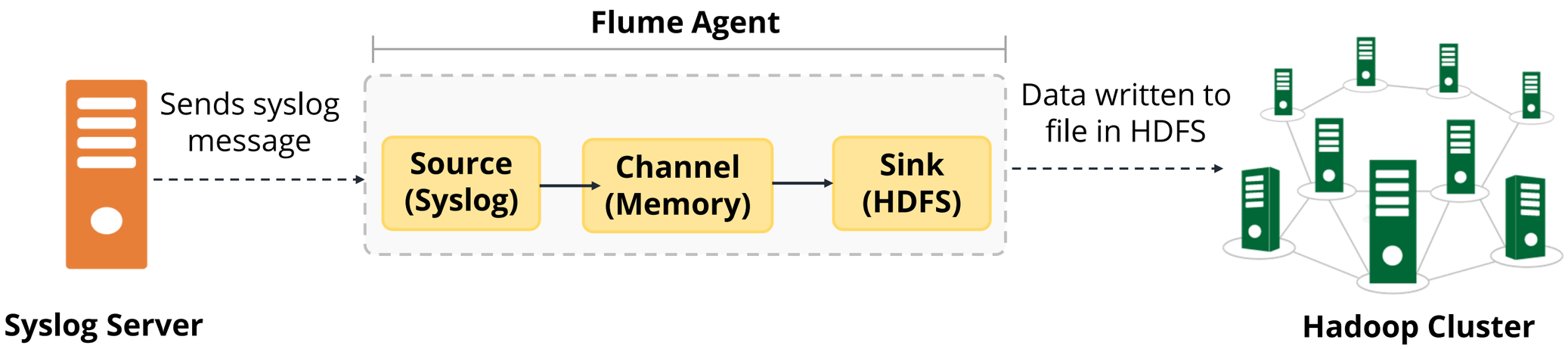


Common Flume Data Sources



Flume Data Flow

The diagram illustrates how syslog data is captured to HDFS.

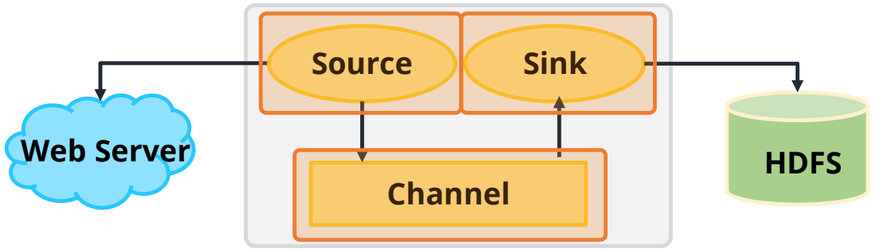


Components in Flume's Architecture

Components in Flume's Architecture

- **Source:**
  Receives events from the external actor that generates them

- **Sink:**
  Sends an event to its destination. It stores the data into centralized stores like HDFS and HBASE



- **Channel:**
  Buffers events from the source until they are drained by the sink. It acts as a bridge between the sources and the sinks

- **Agent:**
  Java process that configures and hosts the source, channel, and sink

Flume Source

Netcat:

Listens on a given port and turns each line of text into an event

Kafka:

Receives events as messages from a Kafka topic

Syslog:

Captures messages from UNIX syslog daemon over the network

Spooldir:

Used for ingesting data by placing files to be ingested into a "spooling" directory on disk

Flume Sink

Following are the types of Flume Sink:

Null
Discards all events received
(Flume equivalent of **/dev/null**)

HBaseSink
Stores event in HBase

HDFS
Writes event to a file in the specified directory in HDFS

Flume Channels

Following are the types of Flume channel:

Memory

- Stores events in the machine's RAM
- Extremely fast, but not reliable as memory is volatile

File

- Stores events on the machine's local disk
- Slower than RAM, but more reliable as data is written to disk

JDBC

- Stores events in a database table using JDBC
- Slower than file channel

Flume Agent Configuration File

All Flume agents can be configured in a single Java file.

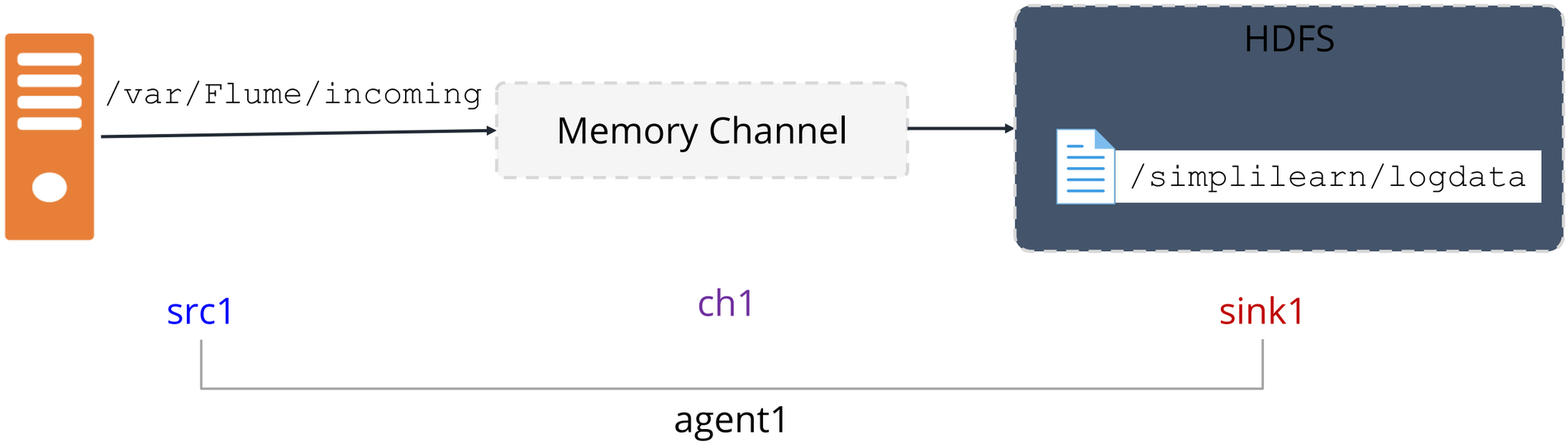
training@localhost:~

File Edit View Search Terminal Help

```
#Define sources, sinks, and channel for agent named 'agent1'
agent1.sources = mysource
agent1.sinks = mysink
agent1.channels = mychannel

# Sets a property 'foo' for the source associated with agent1
agent1.sources.mysource.foo = bar

# Sets a property 'baz' for the sink associated with agent1
agent1.sinks.mysink.baz = bat
```

Example: Configuring Flume Components

Example : Configure a Flume Agent to collect data from remote spool directories and save to HDFS through memory channel.



Example : Configuring Flume Configuration

training@localhost:~

File Edit View Search Terminal Help

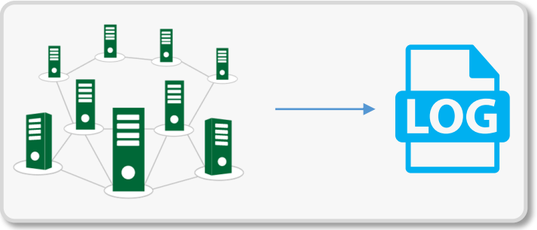
```
agent1.sources = src1
agent1.sinks = sink1
agent.channels = ch1

agent1.channels.ch1.type = memory

agent1.sources.src1.type = spooldir
agent1.sources.src1.spoolDir = /var/Flume/incoming
agent1.sources.src1.channels = ch1

agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path = /simplilearn/logicdata
agent1.sinks.sink1.channel = ch1
```

Connect **source** and channel

Connect **source** and channel

Flume: Sample Use Cases

Flume can be used for a variety of use cases:



To collect logs from nodes in Hadoop cluster

To collect logs from services such as http and mail

For process monitoring

To collect impressions from custom applications for an advertisement network

Apache Flume

Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to ingest Twitter data from Apache Flume and ingest into HDFS.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Apache Kafka

Apache Kafka: Introduction

Kafka is a high-performance, real-time messaging system. It is an open source tool and is a part of Apache projects.

The characteristics of Kafka are:

- It is a distributed and partitioned messaging system.
- It is highly fault-tolerant.
- It is highly scalable.
- It can process and send millions of messages per second to several receivers.

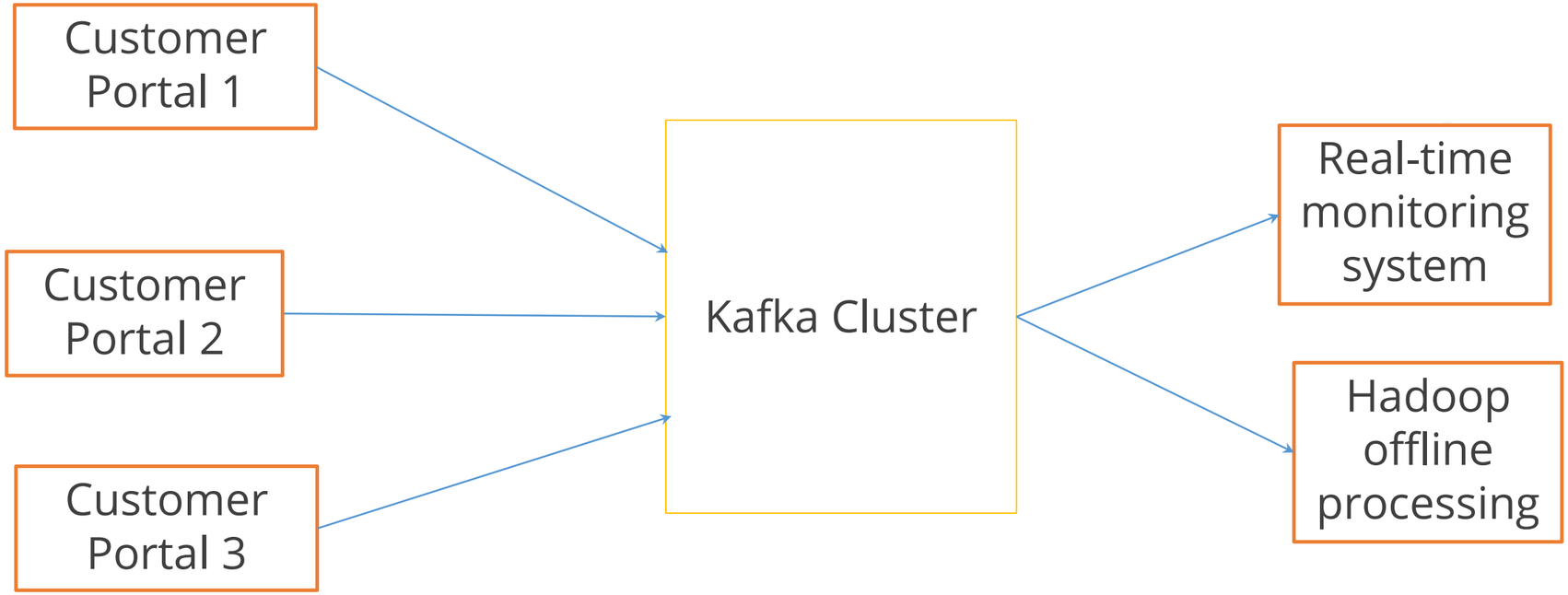
Apache Kafka: Use Cases

Kafka can be used for various purposes in an organization, such as:

Messaging service	Kafka can be used to send and receive millions of messages in real-time.
Real-time stream processing	Kafka can be used to process a continuous stream of information in real-time and pass it to stream processing systems such as Storm.
Log aggregation	Kafka can be used to collect physical log files from multiple systems and store them in a central location such as HDFS.
Commit log service	Kafka can be used as an external commit log for distributed systems.
Event sourcing	Kafka can be used to maintain a time ordered sequence of events.
Website Activity Tracking	Kafka can be used to process real-time website activity such as page views, searches, or other actions users may take.

Aggregating User Activity Using Kafka

Kafka can be used to aggregate user activity data such as clicks, navigation, and searches from different websites of an organization; such user activities can be sent to a real-time monitoring system and hadoop system for offline processing.



Kafka in LinkedIn

Kafka is used by LinkedIn to manage streams of information.

Monitoring

- Collect metrics
- Create monitoring dashboards

Messaging

- Used for message queues in content feeds
- Used as publish-subscribe system for searches and content feeds

Analytics

- Collection of page views and clicks
- Store into a central hadoop-based analytics system

A building block for distributed applications

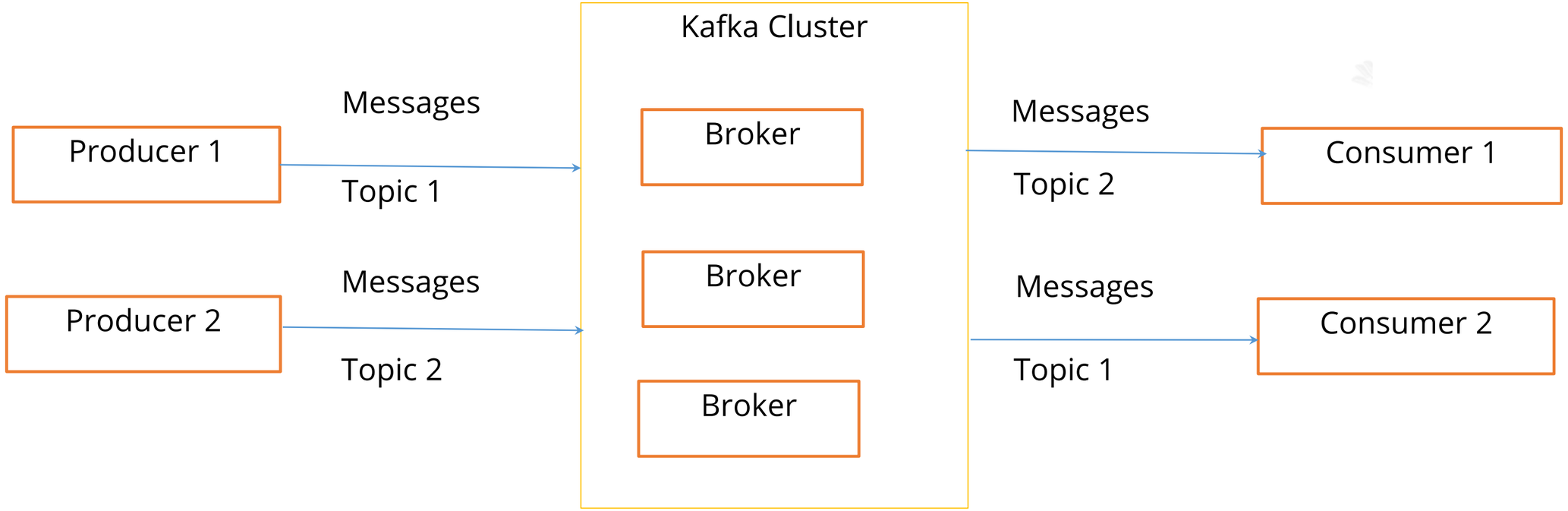
- For distributed databases
- For distributed log systems

Kafka Data Model

Kafka Data Model

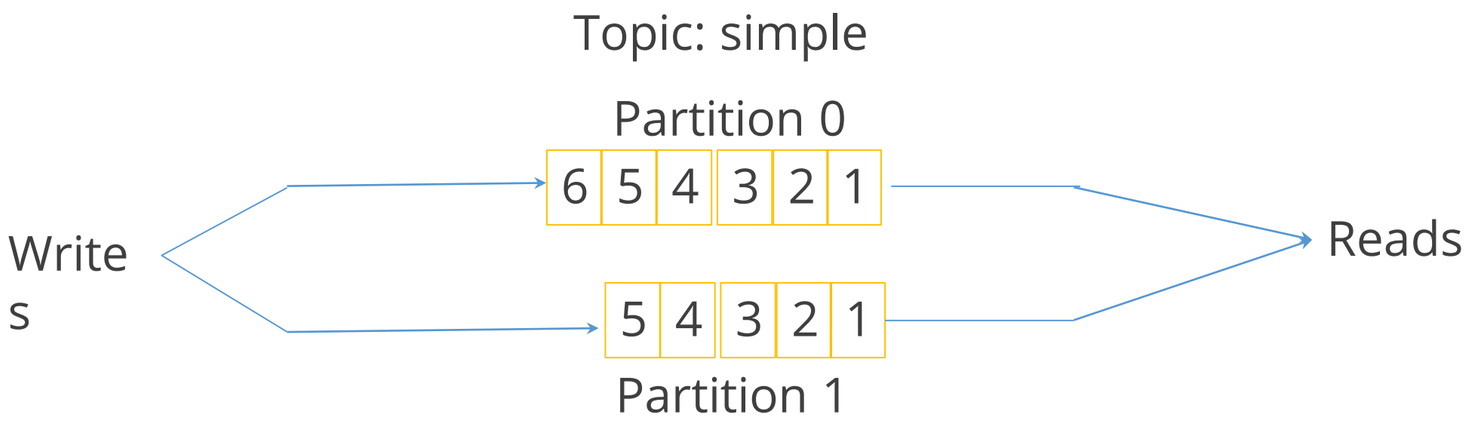
The Kafka data model consists of messages and topics.

- Messages represent information such as, lines in a log file, a row of stock market data, or an error message.
- Messages are grouped into categories called **topics**. Example: LogMessage and StockMessage.



Topics

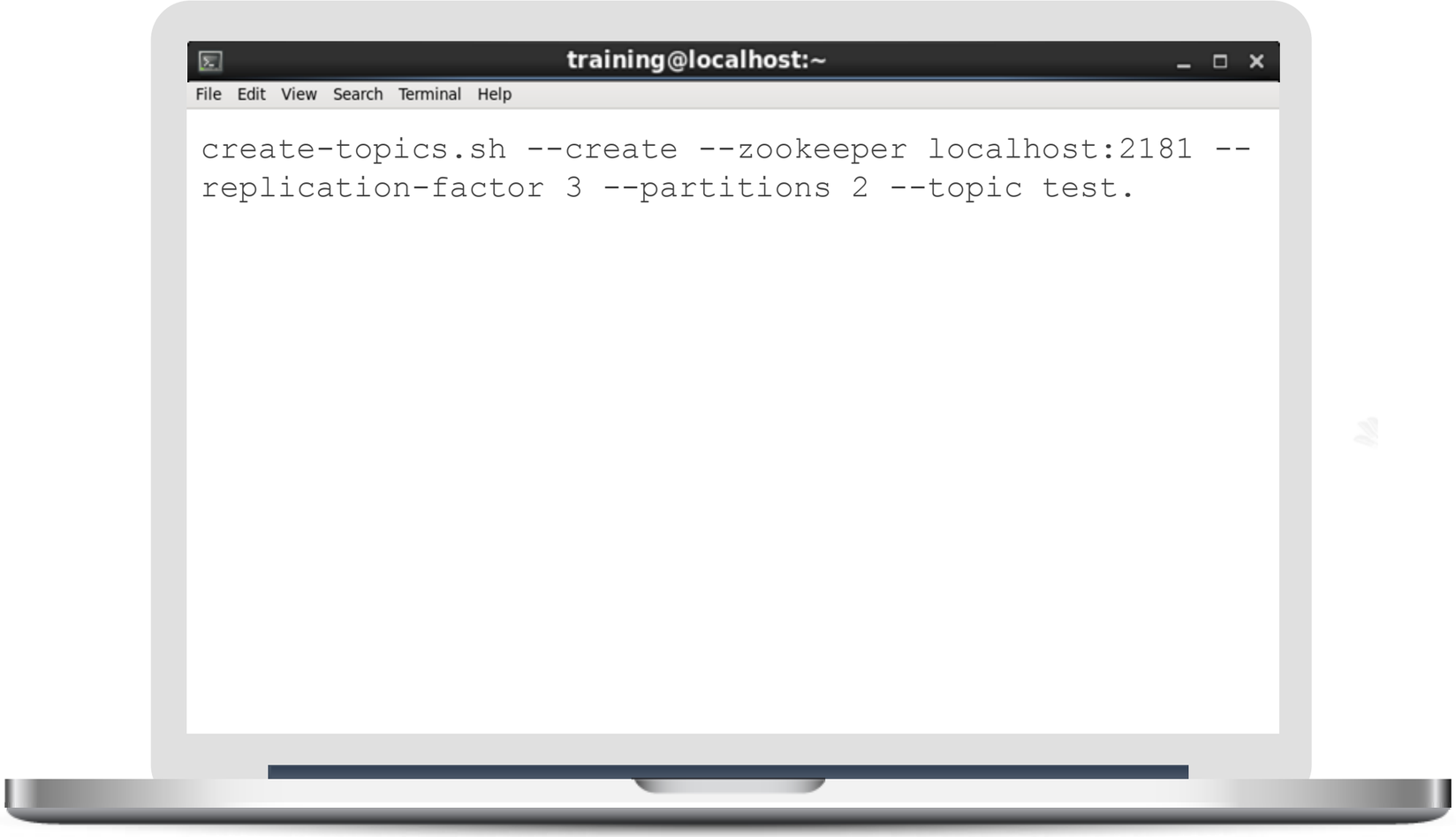
A topic is a category of messages in Kafka.



A topic is divided into one or more partitions which consist of ordered set of messages.

Topics

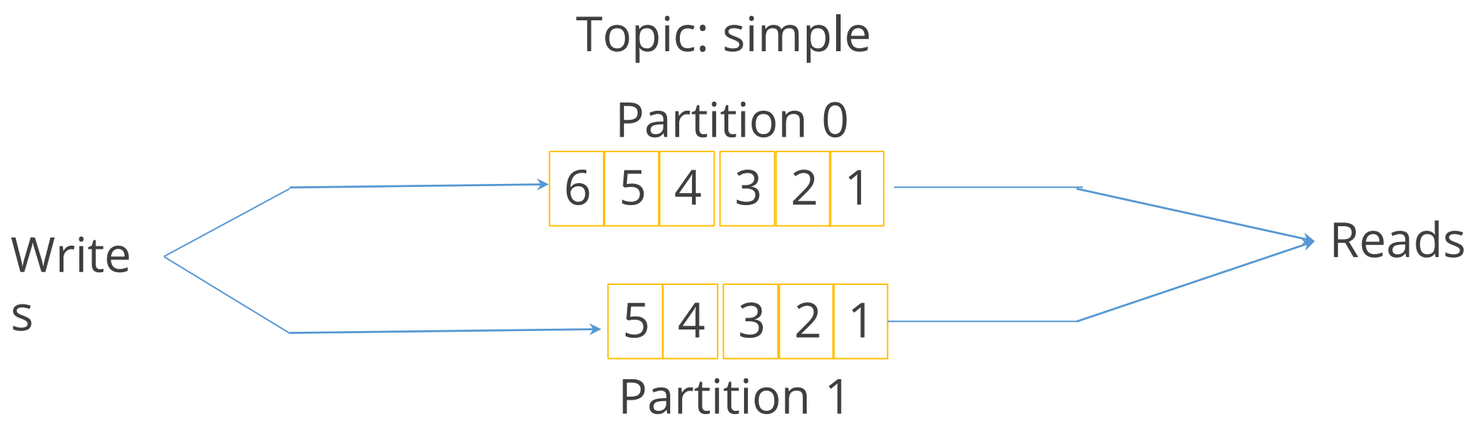
Kafka provides the kafka-topics.sh command to create and modify topics.



Partitions

Topics are divided into partitions, which are the unit of parallelism in Kafka.

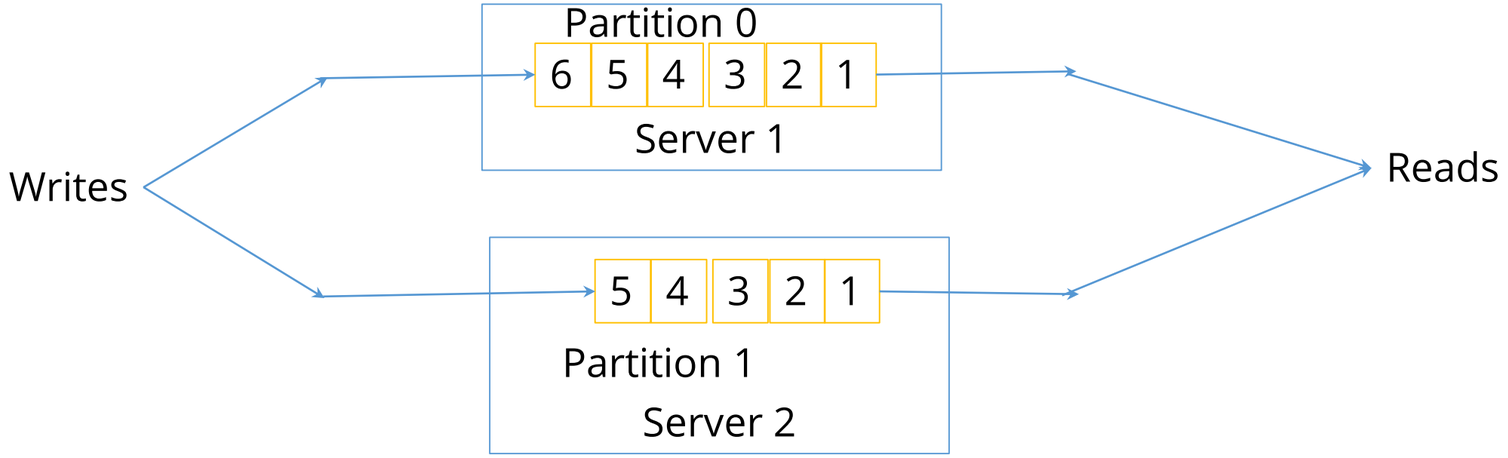
- Partitions allow messages in a topic to be distributed to multiple servers.
- A topic can have any number of partitions.
- Each partition should fit in a single Kafka server.
- The number of partitions decide the parallelism of the topic.



Partition Distribution

Partitions can be distributed across the Kafka cluster.

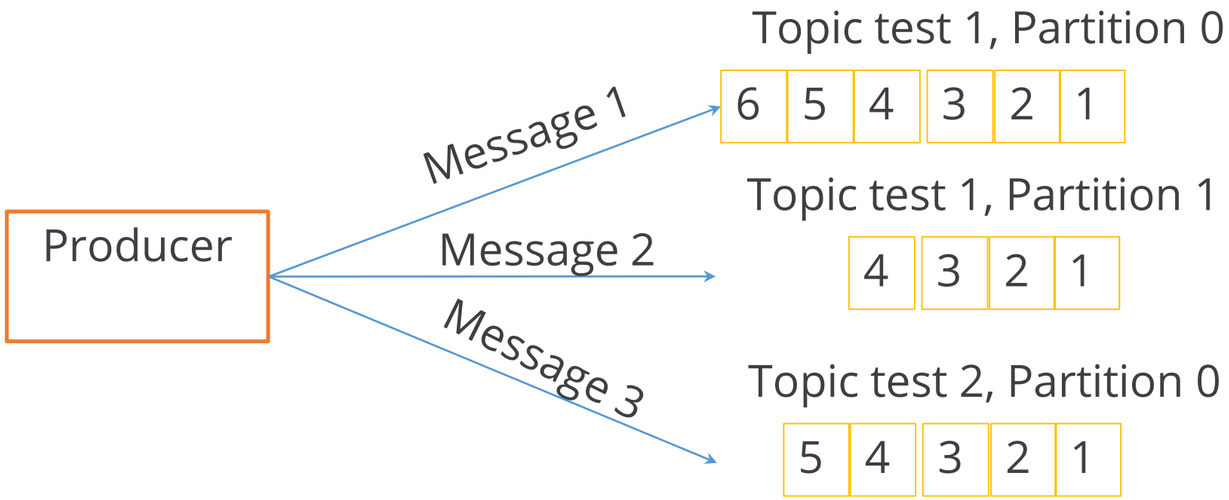
- Each Kafka server may handle one or more partitions.
- A partition can be replicated across several servers for fault-tolerance.
- One server is marked as a leader for the partition and the others are marked as followers.
- The leader controls the read and write for the partition, whereas, the followers replicate the data.
- If a leader fails, one of the followers automatically becomes the leader.
- ZooKeeper is used for the leader selection.



Producers

The producer is the creator of the message in Kafka.

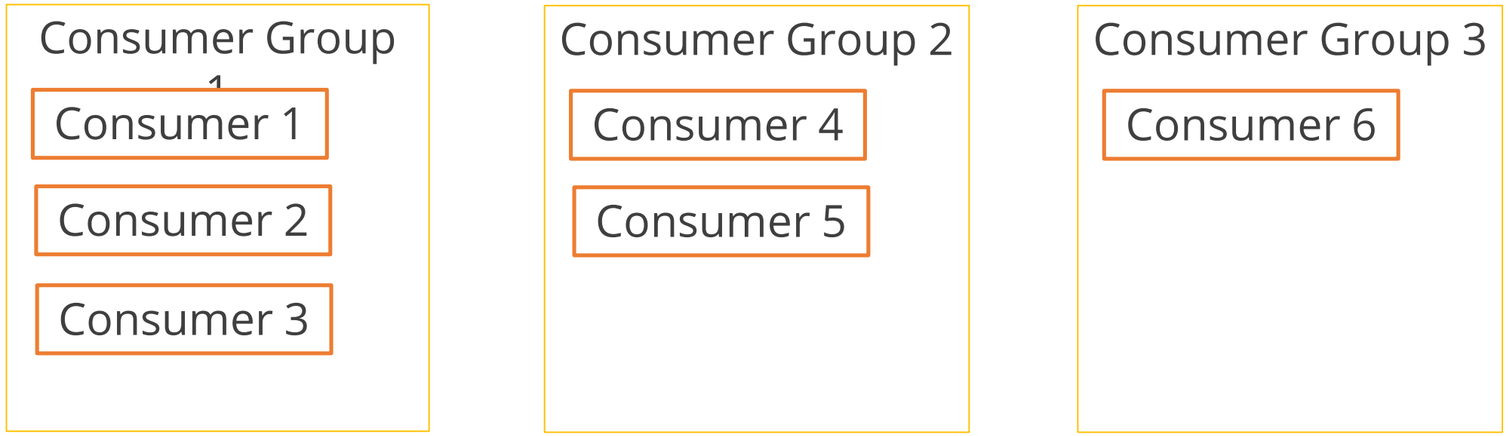
- The producers place the message to a particular topic.
- The producers also decide which partition to place the message into.
- Topics should already exist before a message is placed by the producer.
- Messages are added at one end of the partition.



Consumers

The consumer is the receiver of the message in Kafka.

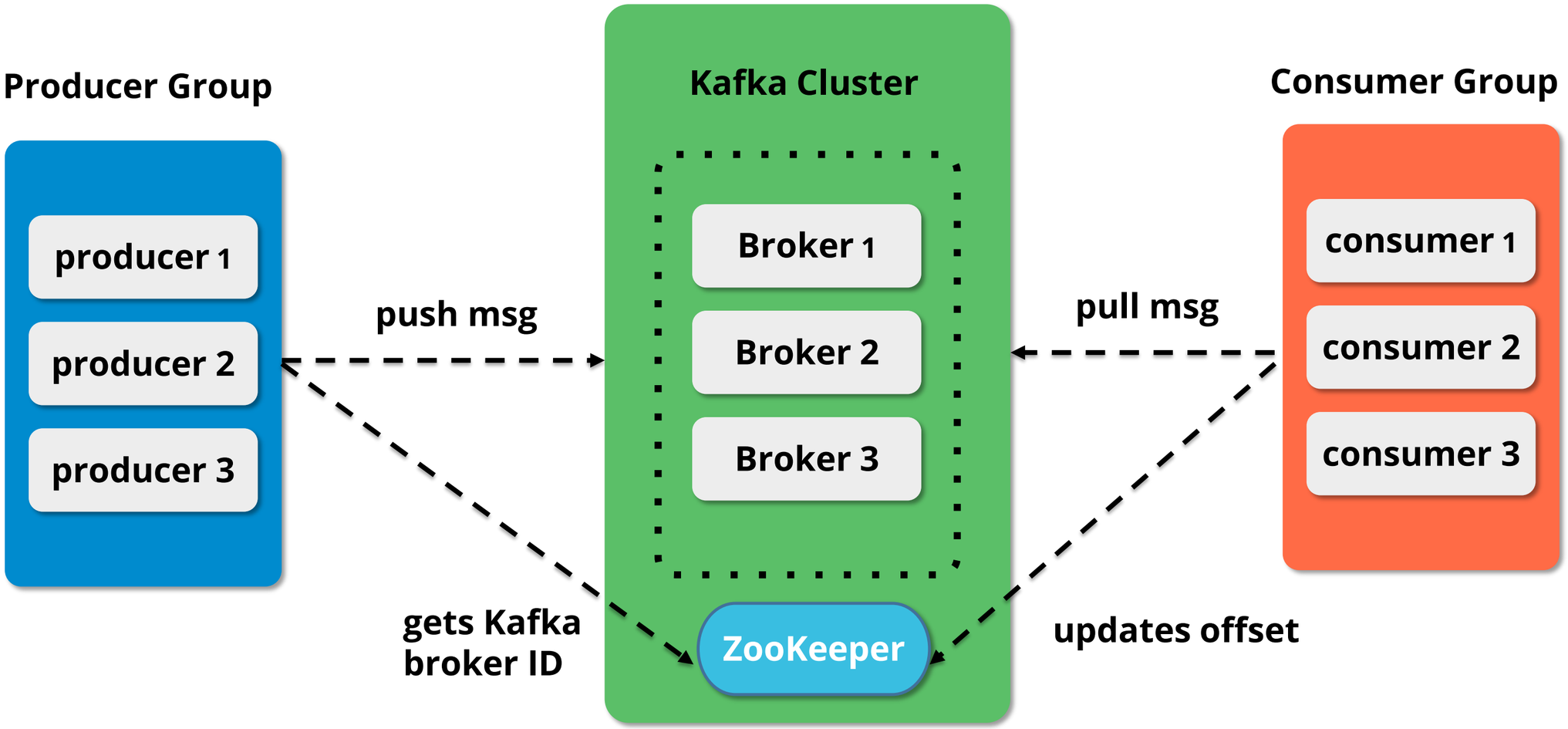
- Each consumer belongs to a consumer group.
- A consumer group may have one or more consumers.
- The consumers specify what topics they want to listen to.
- A message is sent to all the consumers in a consumer group.
- The consumer groups are used to control the messaging system.



Apache Kafka Architecture

Kafka Architecture

Given below is a Kafka cluster architecture:



Apache Kafka

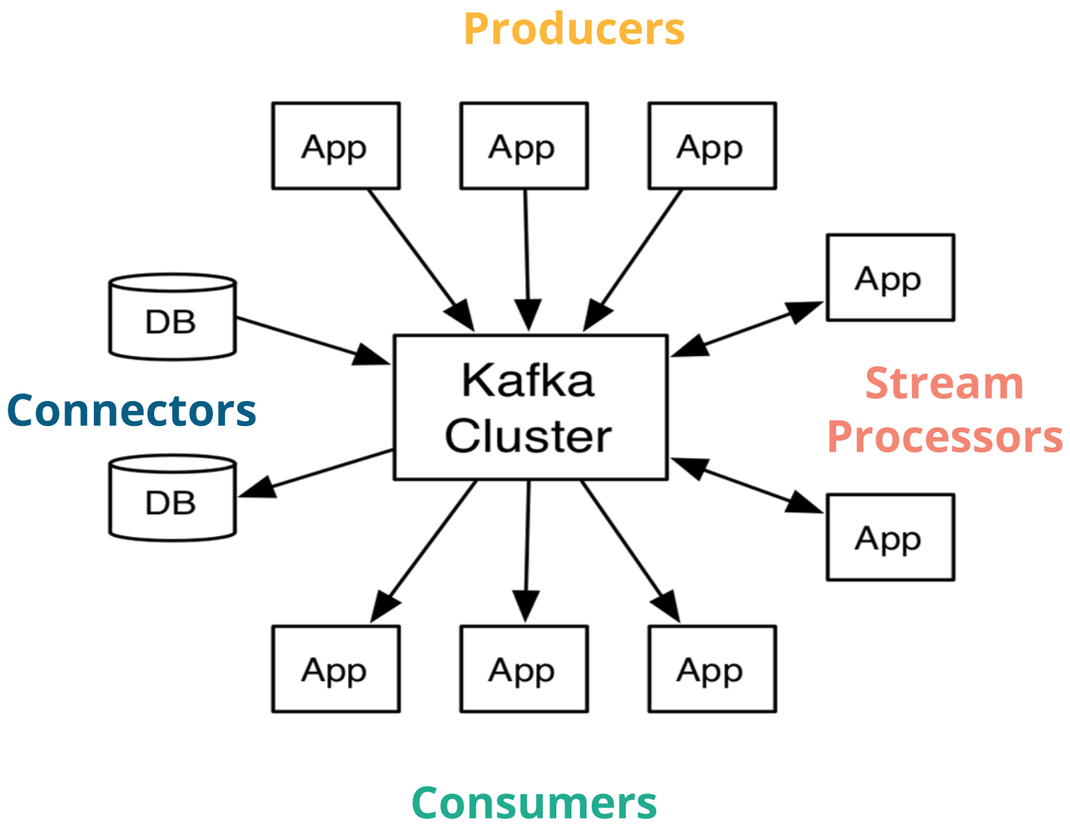
Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to setup a Kafka Cluster on CloudLab.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Kafka APIs

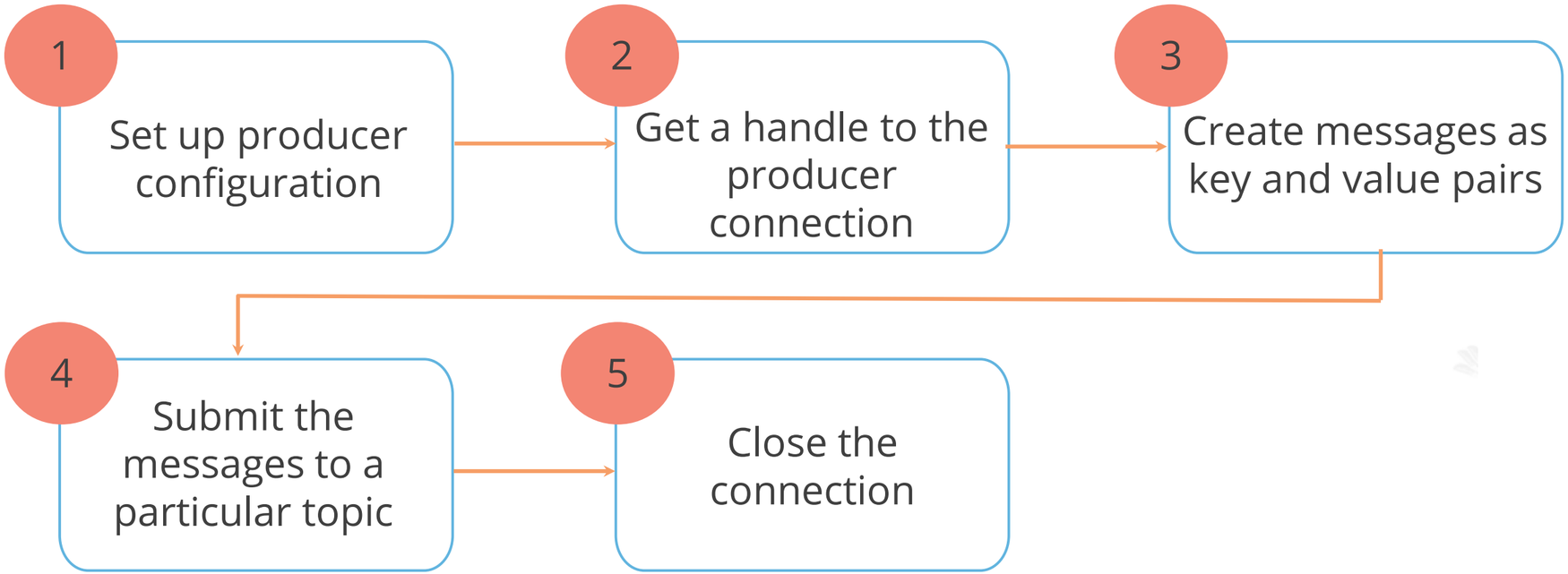
Kafka has four core APIs:



- The **Producer API** allows applications to send streams of data to topics in the Kafka cluster.
- The **Streams API** allows transforming stream of data from input topics to output topics.
- The **Consumer API** allows applications to read streams of data from topics in the Kafka cluster.
- The **Connect API** allows data ingestion from some source system into Kafka or push from Kafka into some sink system.

Producer Side API

The producer side APIs provide interface to connect to the cluster and insert messages into a topic. The steps involved in programming are as follows:



By default, a message is submitted to a particular partition of the topic based on the hash value of the key. A programmer can override this with a custom partitioner.

Producer Side API Example: Step 1

Step 1: Set up producer configuration.

```
Properties props = new Properties();
props.put("metadata.broker.list", "localhost:9092 ");
props.put("serializer.class", "kafka.serializer.StringEncoder");
ProducerConfig config = new ProducerConfig(props);
```

Producer Side API Example: Step 2

Step 2: Get a handle to the producer connection.

```
Producer<String, String> producer = new Producer<String, String>(config);
```

Producer Side API Example: Step 3

Step 3: Create messages as key and value pairs.

String key1 = “first”; String message1 = “This is first message”;

String key2 = “second”; String message2 = “This is second message”;

Producer Side API Example: Step 4

Step 4: Submit the messages to a particular topic.

```
String topic = “test”;
KeyedMessage<String, String> data = new KeyedMessage<String, String>(topic, key1, message1);
producer.send(data);
data = new KeyedMessage<String, String>(topic, key2, message2);
producer.send(data);
```

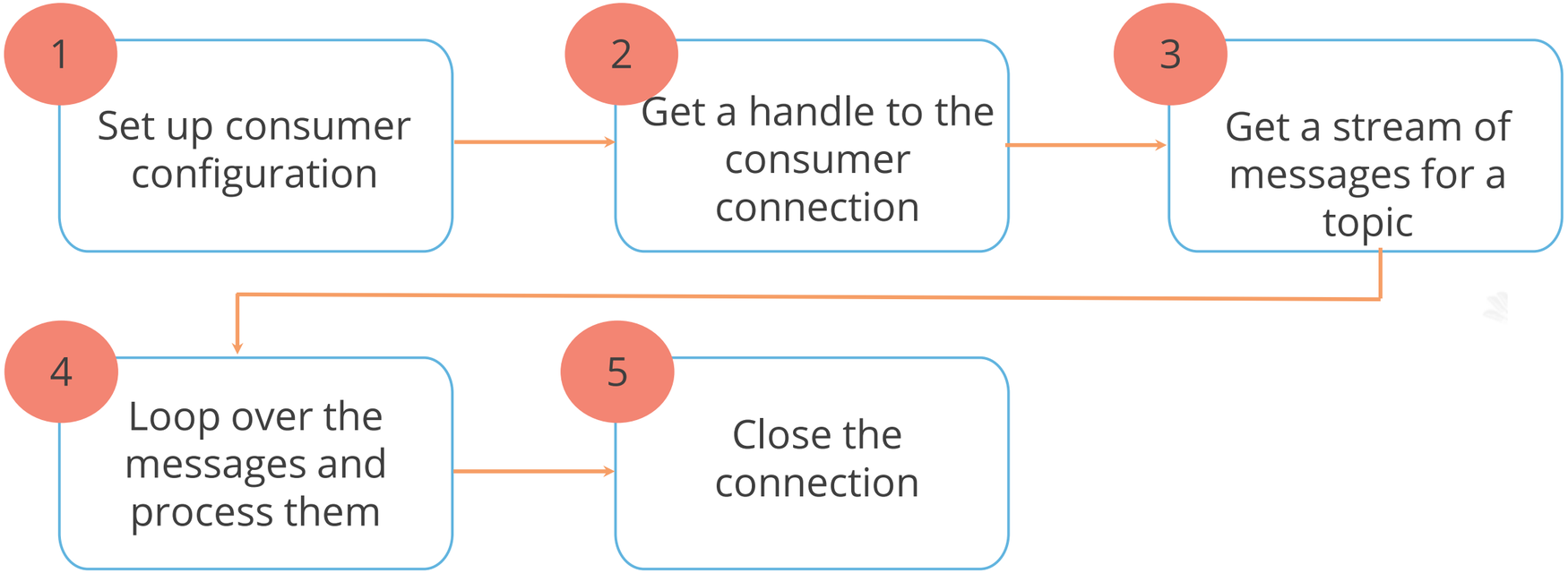
Producer Side API Example: Step 5

Step 5: Close the connection.

```
producer.close();
```

Consumer Side API

The consumer side APIs provide interface to connect to the cluster and get messages from a topic. The steps involved in programming are:



Messages can be read from a particular partition or from all the partitions.

Consumer Side API Example: Step 1

Step 1: Setup consumer configuration.

```
Properties props = new Properties();
props.put("zookeeper.connect", "localhost:2181");
props.put("group.id", “mygroup");
ConsumerConfig config = new ConsumerConfig(props);
```

Consumer Side API Example: Step 2

Step 2: Get a handle to the consumer connection.

```
ConsumerConnector consumer;
consumer = kafka.consumer.Consumer.createJavaConsumerConnector(config);
```

Consumer Side API Example: Step 3

Step 3: Get stream of messages for the topic.

```
String topic = “test”;
Map<String, Integer> topicCountMap = new HashMap<String, Integer>();
topicCountMap.put(topic, new Integer(1));
Map<String, List<KafkaStream<byte[], byte[]>>> consumerMap =
consumer.createMessageStreams(topicCountMap);
List<KafkaStream<byte[], byte[]>> streams = consumerMap.get(topic);
```

Consumer Side API Example: Step 4

Step 4: Loop over the messages and process them.

```
for (final KafkaStream stream : streams) {
ConsumerIterator<byte[], byte[]> it = stream.iterator();
while (it.hasNext())
System.out.println(new String(it.next().message()));
}
```

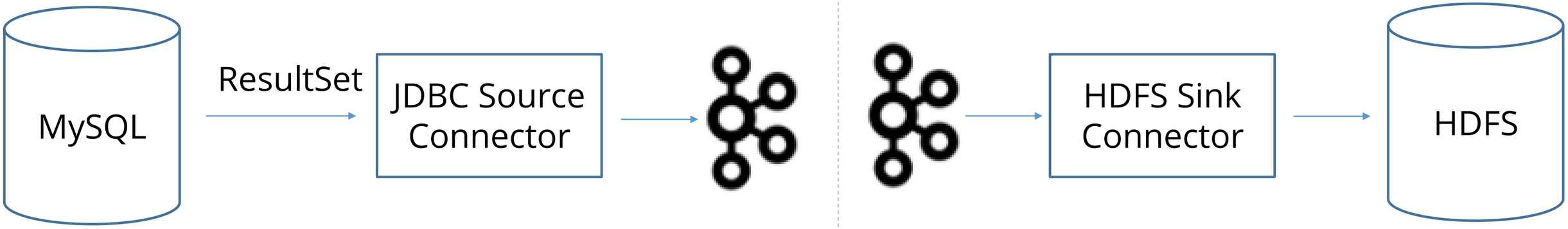
Consumer Side API Example: Step 5

Step 5: Close the connection to consumer.

```
consumer.shutdown();
```

Kafka Connect

Kafka Connect is a framework for connecting Kafka with external systems such as databases, key-value stores, search indexes, and file systems, using so-called **Connectors**.



Source Connector

Sink Connector

Confluent Connector

Confluent connector is an alternative to Kafka connect which comes with some additional tools and clients, compared to plain Kafka, as well as some additional pre-built Connectors.

Kafka Connect JDBC

Kafka Connect S3

SAP Hana Connector

Apache Kafka

Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to create a sample Kafka data pipeline using producer and consumer.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Key Takeaways

You are now able to:

- Understand various Big Data Ingestion Tools
- Define Sqoop and its uses
- Analyze importing and exporting data from Hadoop using Sqoop
- Explain Apache Flume along with its uses
- Explain the components in the Flume architecture
- Define Kafka and its architecture

Knowledge Check

Knowledge Check

1

Which of the following Apache Hadoop Ecosystem projects is a command-line interface application for transferring data between relational databases and Hadoop?

a. Apache Flume

b. Apache Kafka

c. Apache Sqoop

d. All of the above

Knowledge Check

1

Which of the following Apache Hadoop Ecosystem projects is a command-line interface application for transferring data between relational databases and Hadoop?

a. Apache Flume

b. Apache Kafka

c. Apache Sqoop

d. All of the above

The correct answer is **c.**

Sqoop, an Apache Hadoop Ecosystem project, is a command-line interface application for transferring data between relational databases and Hadoop.

Knowledge Check

2

Which of the following is a Flume Source?

a. Null

b. HDFS

c. Spooldir

d. Sink

Knowledge Check

2

Which of the following is a Flume Source?

a. Null

b. HDFS

c. Spooldir

d. Sink

The correct answer is **c.**

Spooldir is a Flume source.

Knowledge Check

3

Which of the following is the default file format to import data using Apache Sqoop?

a. Sequence File Format

b. Delimited Text File Format

c. Both A and B

d. None of the above

Knowledge Check

3

Which of the following is the default file format to import data using Apache Sqoop?

a. Sequence File Format

b. Delimited Text File Format

c. Both A and B

d. None of the above

The correct answer is **c.**

By using two file formats Sqoop allows data import.

Knowledge Check

4

_____ is fault tolerant, linearly scalable, and stream oriented.

a. Sqoop

b. Flume

c. Kafka

d. All of the above

Knowledge Check

4

_____ is fault tolerant, linearly scalable, and stream oriented.

a. Sqoop

b. Flume

c. Kafka

d. All of the above

The correct answer is **b.**

Flume a flexible data ingestion tool which is fault tolerant, linearly scalable, and stream oriented.

Knowledge Check

5

The parallelism of a Kafka topic can be set using the parameter:

a. replication_factor

b. partitions

c. threads

d. concurrent

Knowledge Check

5

The parallelism of a Kafka topic can be set using the parameter:

a. replication_factor

b. partitions

c. threads

d. concurrent

The correct answer is **b.**

The number of partitions determine the parallelism of a topic in Kafka and is set by using the –partitions option in the create-topic.sh command.

Knowledge Check

6

How does the Kafka console consumer connect to Kafka cluster?

a. By using the list of Kafka servers provided on the command line

b. By using the list of ZooKeeper servers provided on the command line

c. By reading the configuration file

d. It always connects to local host

Knowledge Check

6

How does the Kafka console consumer connect to Kafka cluster?

a. By using the list of Kafka servers provided on the command line

b. By using the list of ZooKeeper servers provided on the command line

c. By reading the configuration file

d. It always connects to local host

The correct answer is **b.**

The Kafka console consumer expects the list of ZooKeeper servers on the command line to connect to the Kafka cluster.

Lesson-End-Project

Problem Statement:

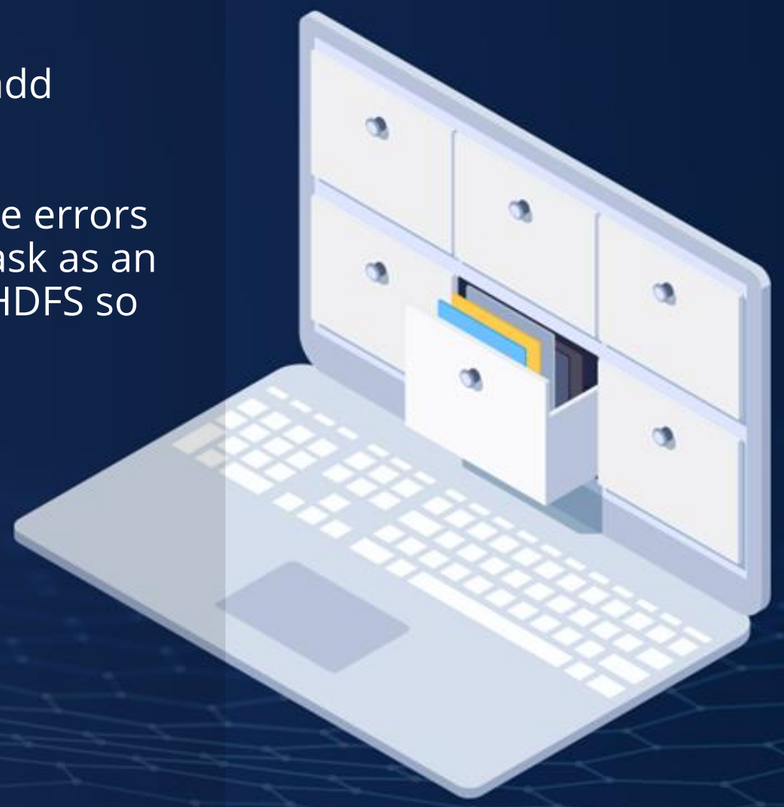
Target.com is one of the biggest e-commerce giants in the US.
They have a web portal where customers can purchase items, and the sellers can add products, remove the products, and do inventory-related operations.

Recently, they started getting a lot of errors on the portal. They have collected these errors from all the applications and compiled them in a text file. Processing logs is a big task as an application can generate a lot of logs in a single day. They want to send all logs to HDFS so they can check which are the most frequent errors they are getting.

You have given an **error log** file containing the below details.
1. Dates
2. Server
3. Error message

You must read data from the text file and send it to Kafka and flume script.
Also, you should be able to read data from Kafka and push it into HDFS.

Your task is to create:
1. A Kafka producer which reads the CSV file data one by one and pushes it to Kafka
2. Write flume configuration where the source is Kafka and the sink is HDFS



Thank You