Big Data Hadoop and Spark Developer

simplilearn

Course Introduction

Course Objectives

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

- Learn how to navigate the Hadoop Ecosystem and understand how to optimize its use
- Ingest data using Sqoop, Flume, and Kafka
- Implement partitioning, bucketing, and indexing in Hive
- Work with RDD in Apache Spark
- Process real-time streaming data
- Perform DataFrame operations in Spark using SQL queries
- Implement User-Defined Functions (UDF) and User-Defined Attribute Functions (UDAF) in Spark

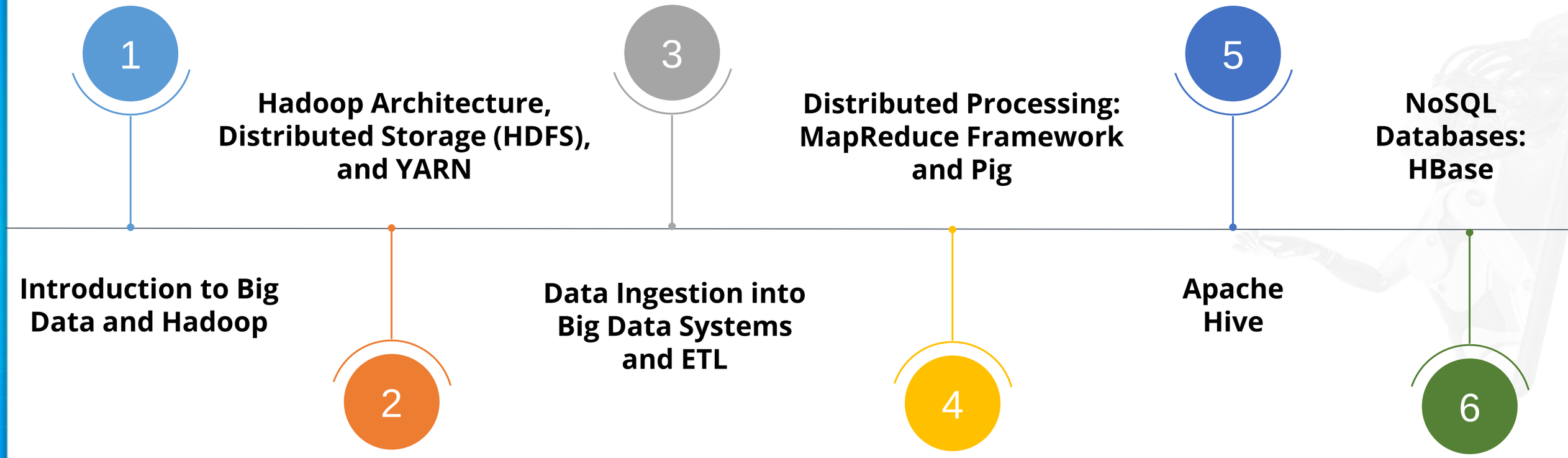
Course Prerequisites

The course requires prior knowledge of the following technologies:

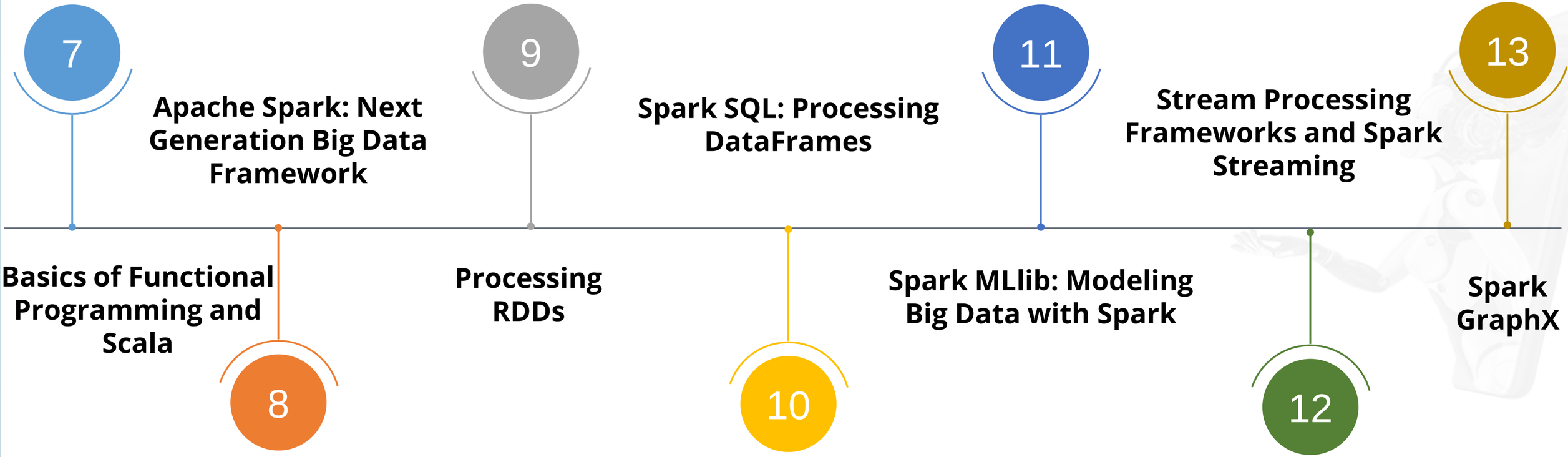
Core Java

SQL

Course Outline



Course Outline



Skills Covered

1. HDFS
1. MapReduce
1. Flume
1. Kafka
1. Hive
1. HBase

Project Highlights

Use Hadoop features to predict patterns and share actionable insights for a car insurance company.

NYSE

Use Hive features for data engineering and analysis of New York stock exchange data.

glassdoor

Perform sentiment analysis on employee review data gathered from Google, Netflix, and Facebook.

amazon

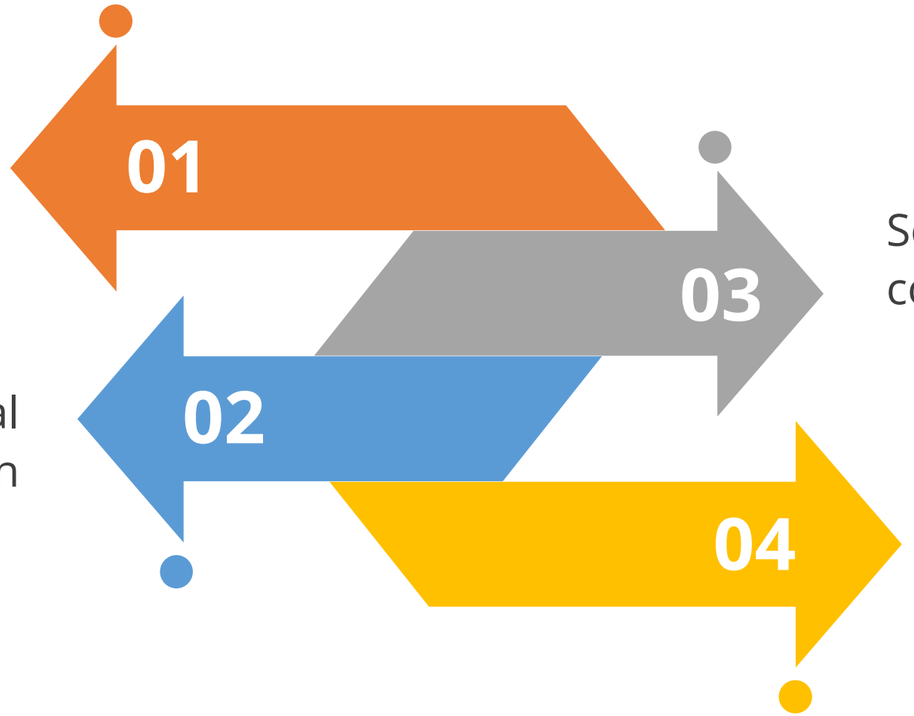
Perform product and customer segmentation to increase the sales of Amazon.

Course Completion Criteria

Complete 85% of the Online Self-Learning (OSL) course

OR

Attend all the Live Virtual Classes (LVC) in a batch



Score a minimum of 75% in course-end assessment

Submit at least one course-end project

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Introduction to Big Data and Hadoop

Learning Objectives

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

- Describe the concepts of Big Data
- Explain Hadoop and how it addresses Big Data challenges
- Describe the components of Hadoop Ecosystem

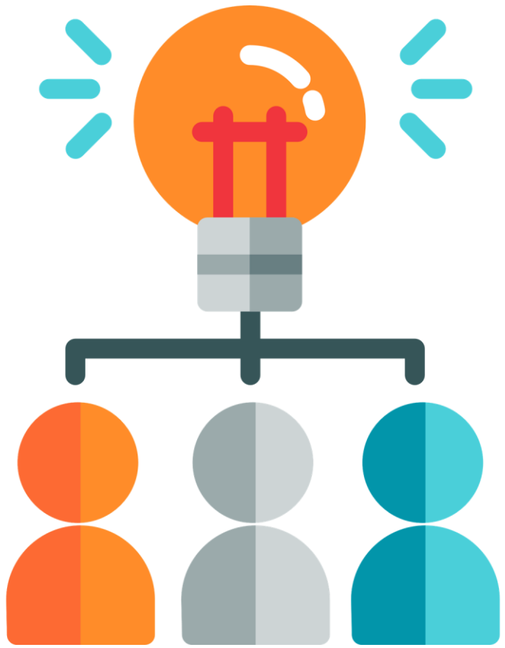
Introduction to Big Data

Big Data Overview

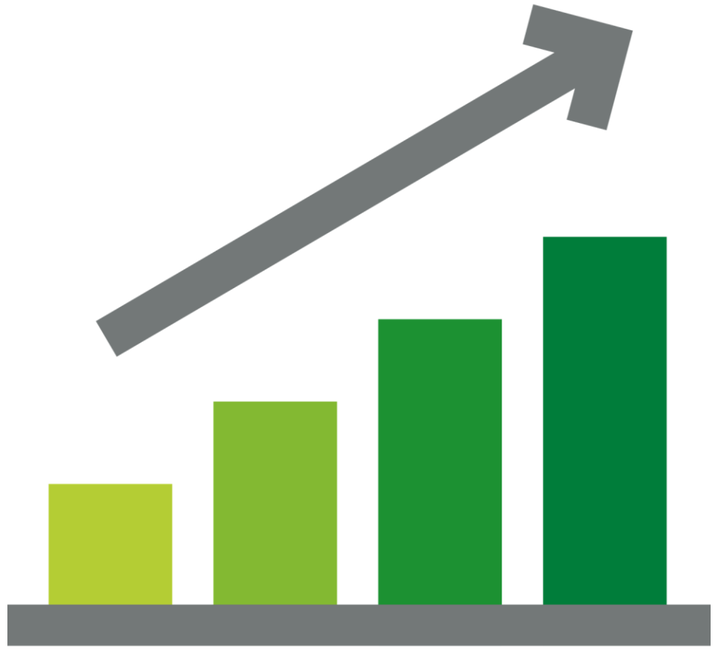
Big Data is the data that has high volume, variety, velocity, veracity, and value.

According to US Bureau of Labour Statistics, Big Data alone will fetch 11.5 million jobs by 2026.

Traditional Decision-Making



What We Think



Experience and Intuition

Rule of Thumb

Challenges of Traditional Decision-Making

Takes a long time to arrive at a decision, therefore losing the competitive advantage

Requires human intervention at various stages

Lacks systematic linkage among strategy, planning, execution, and reporting

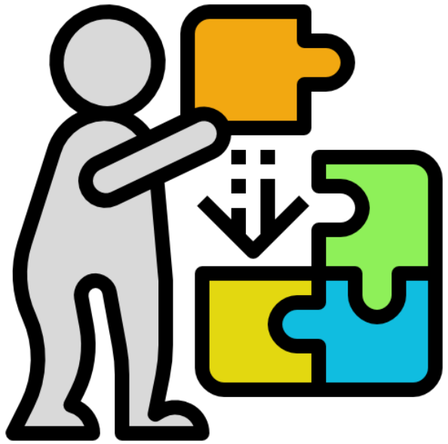
Provides limited scope of data analytics, that is, it provides only a bird's eye view

Obstructs company's ability to make fully informed decisions

DATA AND
ARTIFICIAL INTELLIGENCE

Big Data Analytics

The Solution: Big Data Analytics



Solution

The decision-making is based on what you know which in turn is based on data analytics.

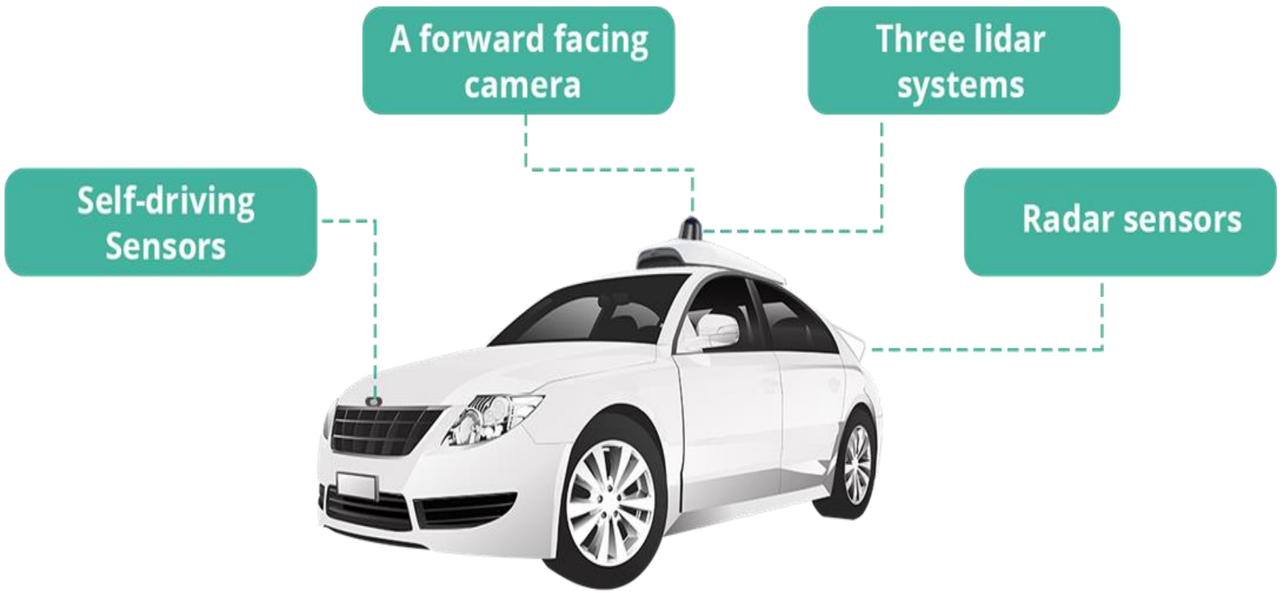
It provides a comprehensive view of the overall picture which is a result of analyzing data from various sources.

It provides streamlined decision-making from top to bottom.

Big data analytics helps in analyzing unstructured data.

It helps in faster decision-making thus improving the competitive advantage and saving time and energy.

Case Study: Google's Self-Driving Car

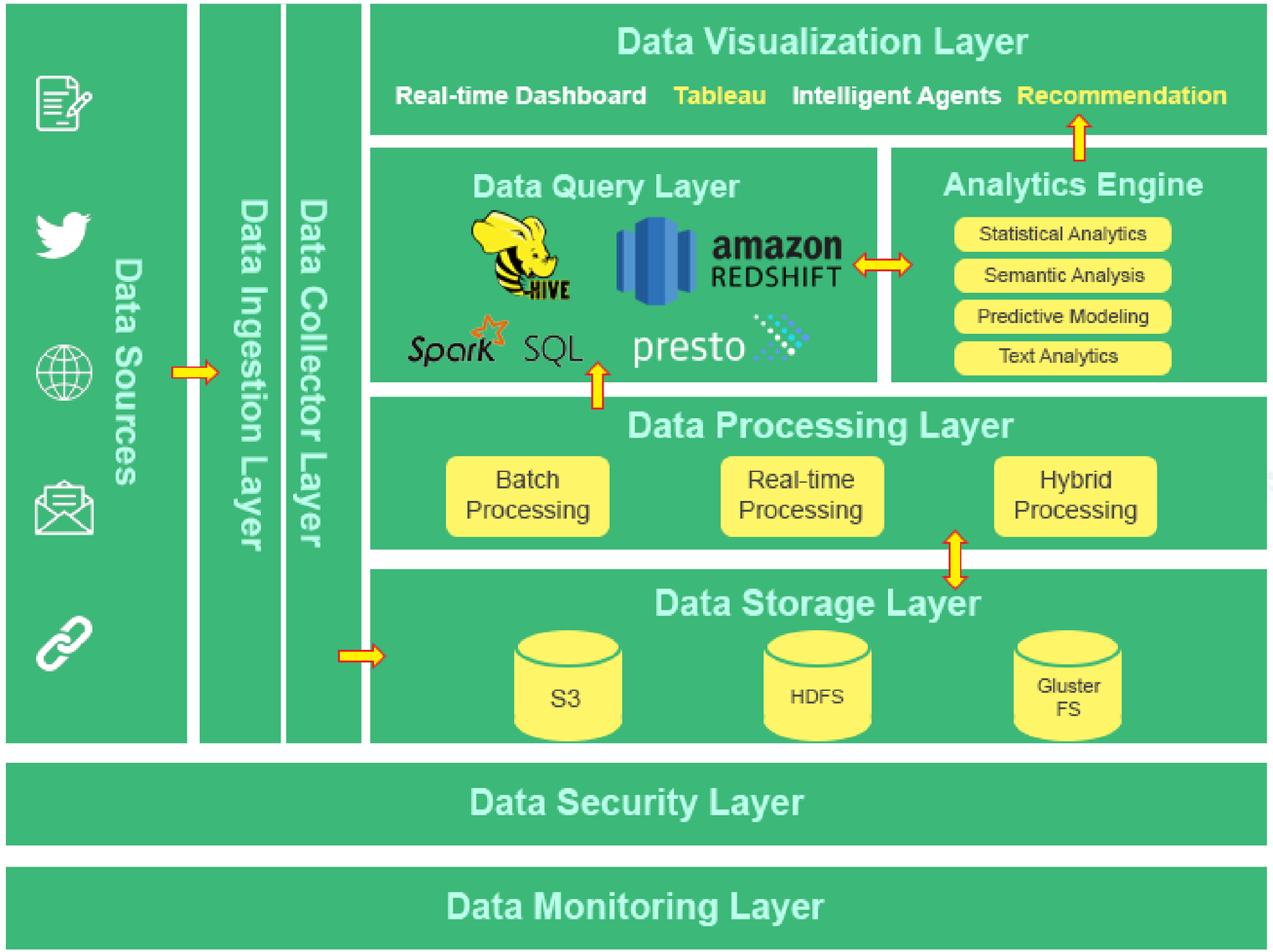


Technical Data

Community Data

Personal Data

Big Data Analytics Pipeline



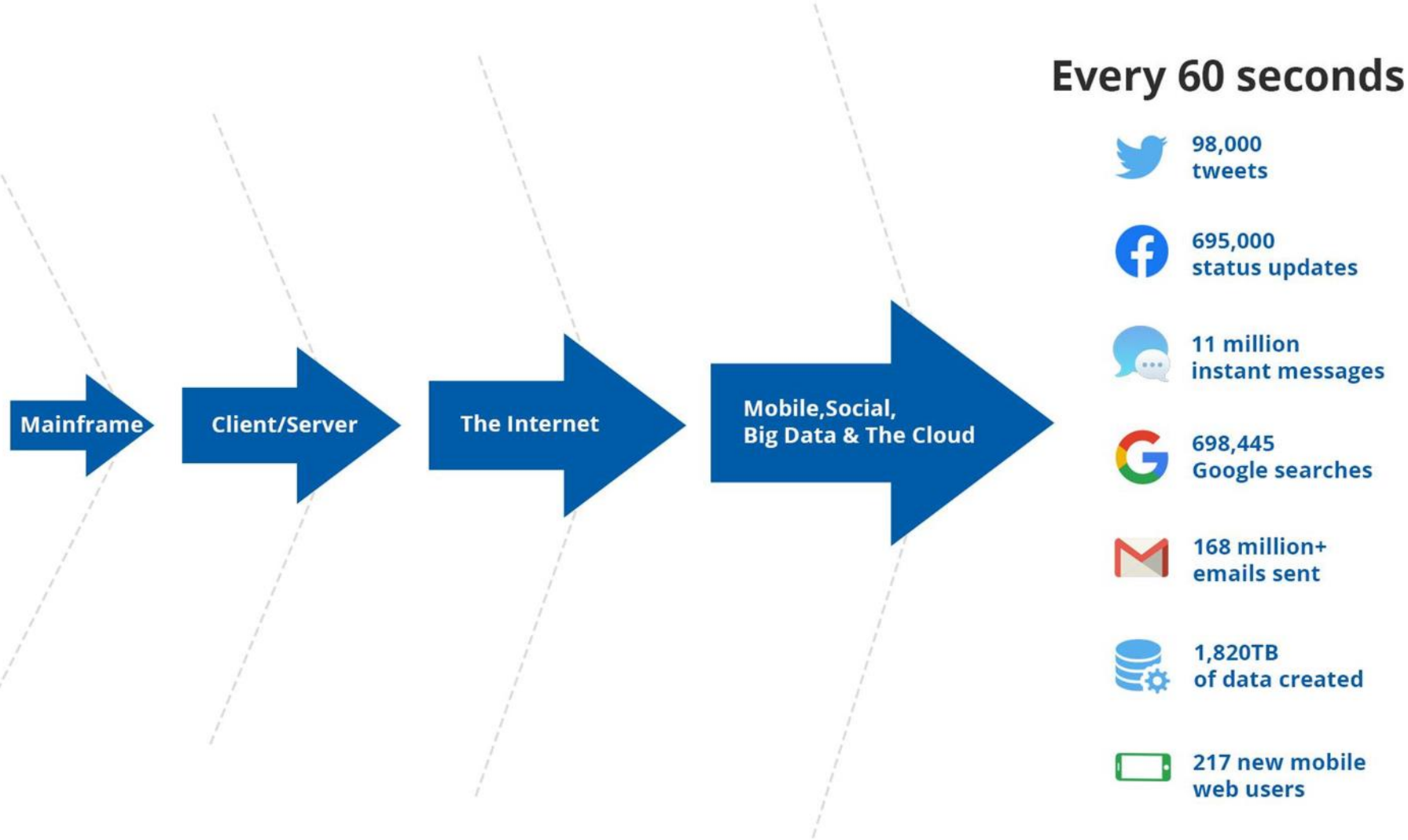
DATA AND
ARTIFICIAL INTELLIGENCE

What Is Big Data?

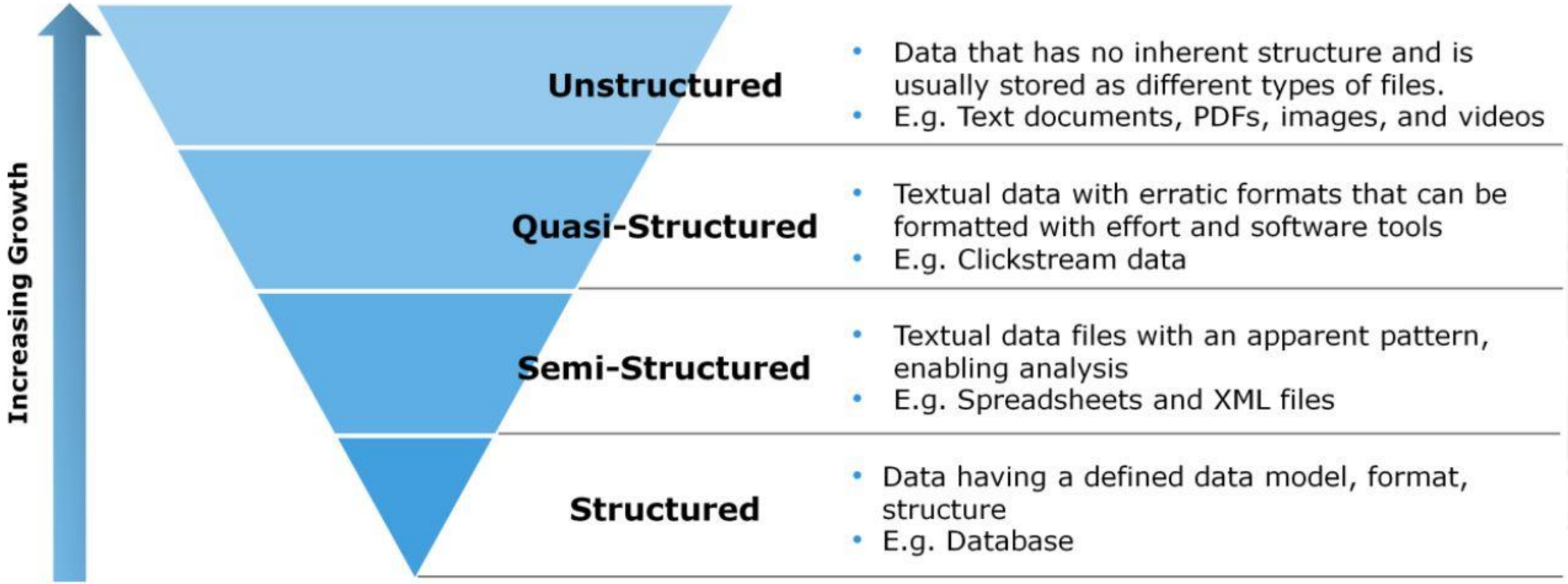
What Is Big Data?

Big data refers to extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.

Big Data at a Glance



Different Types of Data



- Data that has no inherent structure and is usually stored as different types of files.
- E.g. Text documents, PDFs, images, and videos

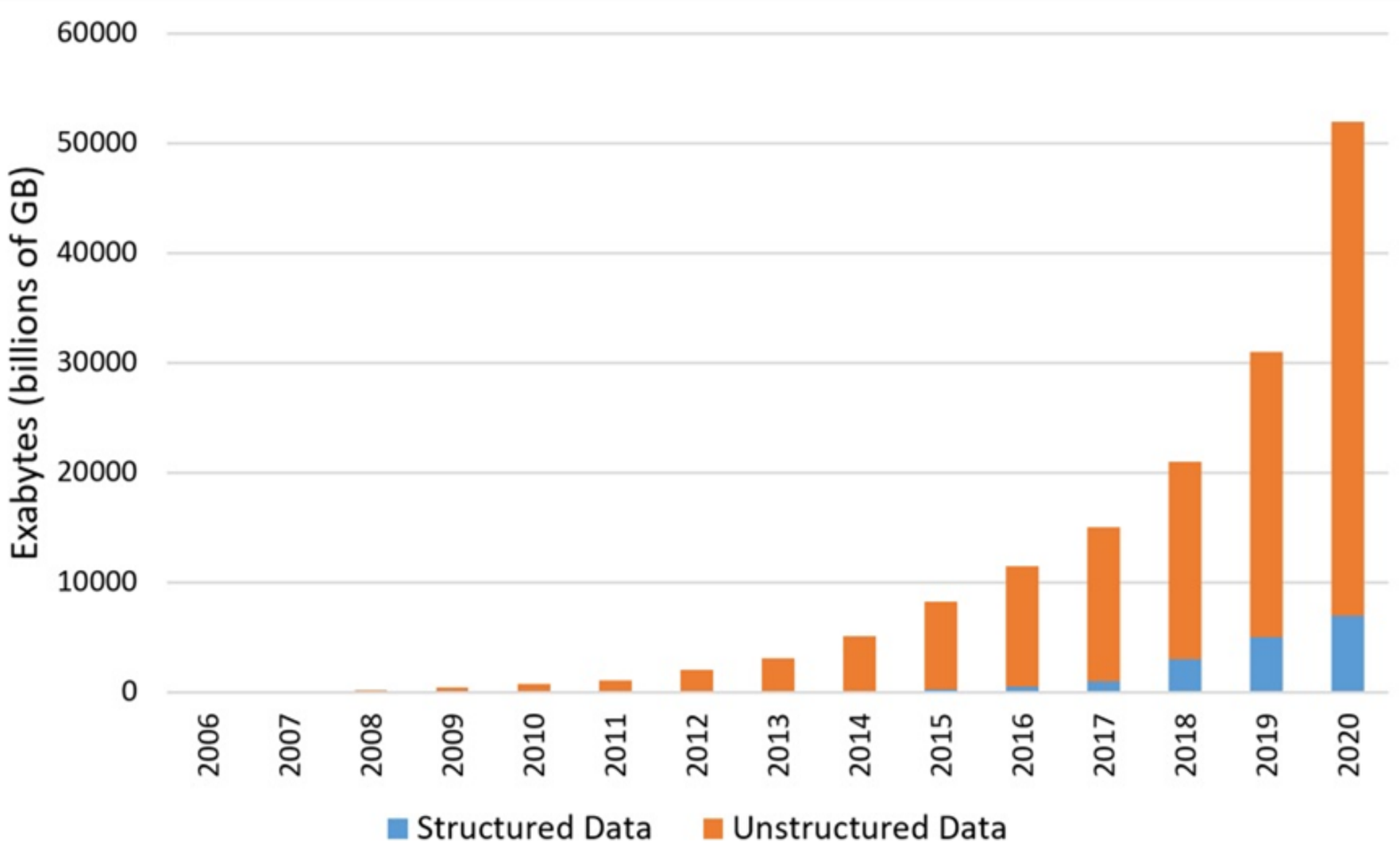
- Textual data with erratic formats that can be formatted with effort and software tools
- E.g. Clickstream data

- Textual data files with an apparent pattern, enabling analysis
- E.g. Spreadsheets and XML files

- Data having a defined data model, format, structure
- E.g. Database

Growth in Data

By 2020, data will show an exponential rise!



DATA AND
ARTIFICIAL INTELLIGENCE

Four Vs of Big Data

Four Vs of Big Data

Volume

- Overall amount of information produced every day is rising exponentially
- 2.3 trillion gigabytes of data is generated every day on the internet

Variety

- Social media, CRM systems, e-mails, audio and video forms produce varied data
- Analytics tools are used to segregate groups based on the type of data generated

Veracity

Inherent discrepancies in the data collected results in inaccurate predictions

Velocity

- More than 50,000 Google searches are completed
- More than 125,000 YouTube videos are viewed
- 7,000 tweets are sent out
- More than 2 million e-mails are sent

Unstructured Data Conundrum

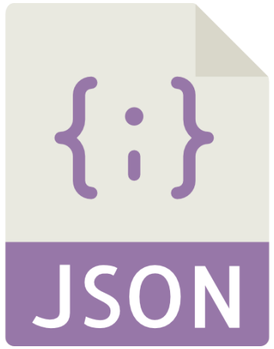
Unstructured Data

Web Logs

Multimedia

Social Media

Semi-structured Data

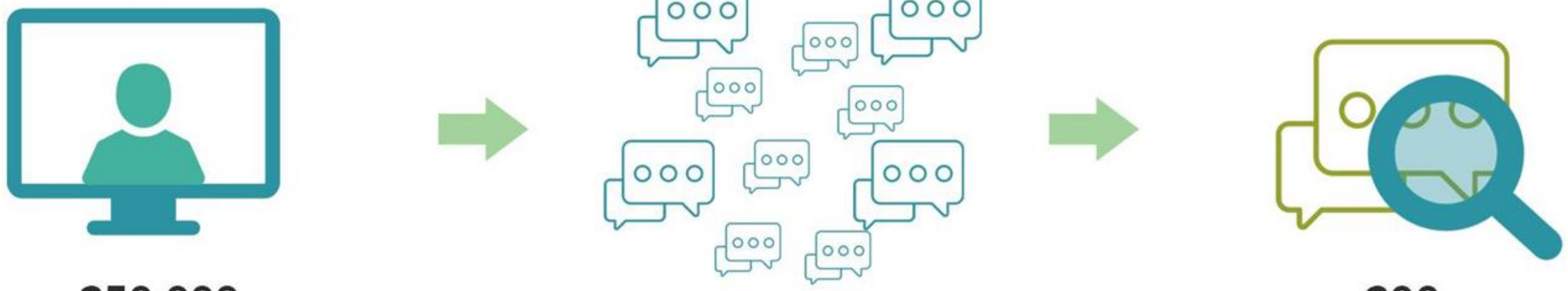


DATA AND
ARTIFICIAL INTELLIGENCE

Case Study: Royal Bank of Scotland

Case Study: Royal Bank of Scotland

Previous Web Chat Analysis Approach



250,000
customer chats per month launched for multiple banking needs

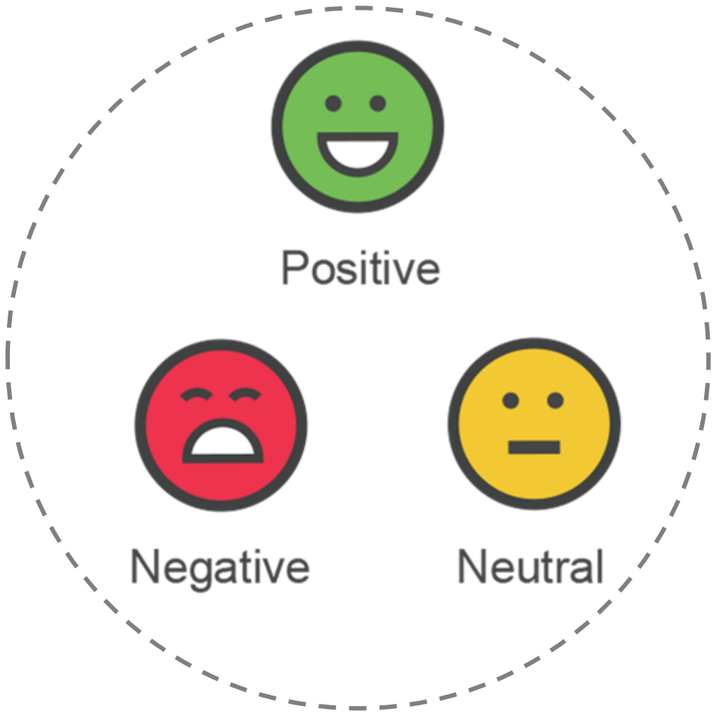
Large volumes of unstructured web chat data

200
chats manually extracted per month and analysed for quality assurance

100% of this data could be processed whereas only 3% could be processed earlier with traditional systems.

Case Study: Royal Bank of Scotland

The case study of Royal Bank of Scotland gave the following three things:



Sentiment analysis

Reduced processing time

Improved customer satisfaction

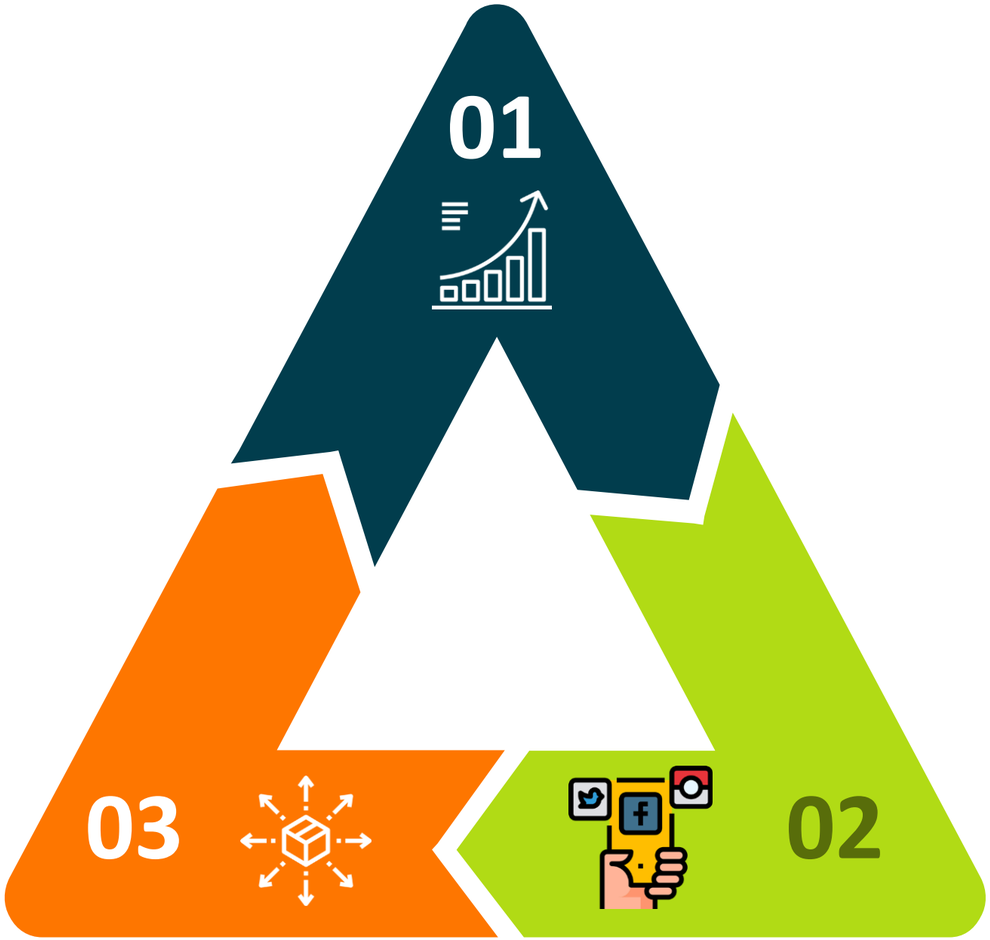
DATA AND
ARTIFICIAL INTELLIGENCE

Challenges of Traditional System

Challenges of Traditional Systems (RDBMS and DWH)

GROWTH RATE

RDBMS systems are designed for steady data retention rather than rapid growth.



DATA SIZE

Data ranges from terabytes ($10^{\wedge 12}$ bytes) to exabytes ($10^{\wedge 18}$ bytes).

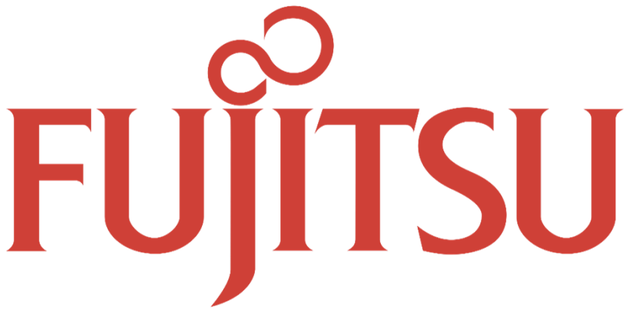
UNSTRUCTURED DATA

Relational databases can't categorize unstructured data.

Advantages of Big Data

1. Processes all types of data at scale
2. Processes huge data quickly in real-time
3. Can run anywhere and additional hardware can be added
4. Better decision-making, thanks to Hadoop

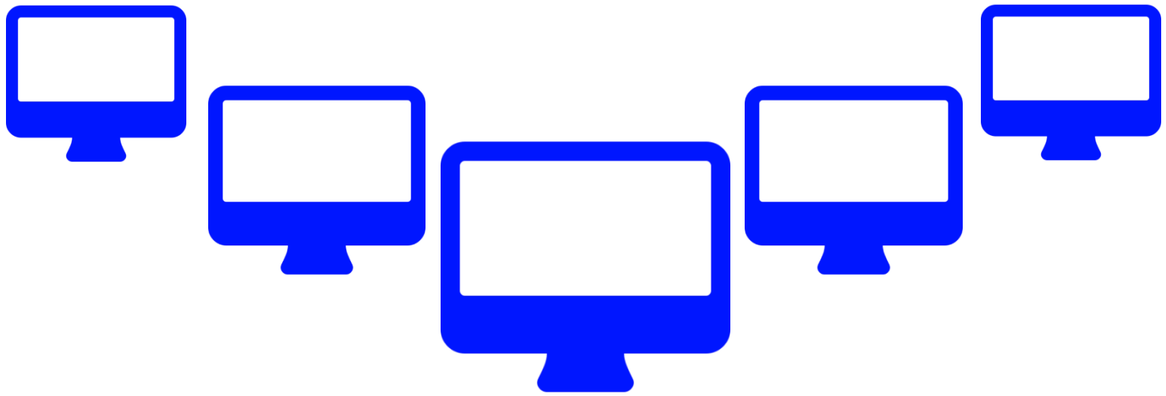
Companies Using Big Data



Big Data: Case Study

1. When do users watch a show?
2. Where do they watch it?
3. On which device do they watch the show?
4. How often do they pause a program?
5. How often do they re-watch a program?
6. Do they skip the credits?
7. What are the keywords searched?

Big Data: Case Study



Multiple systems

Solution

- Traditionally, the analysis of such data was done using a computer algorithm that was designed to produce a correct solution for any given instance.
- As the data started to grow, a series of computers were employed to do the analysis.
- They were also known as distributed systems.

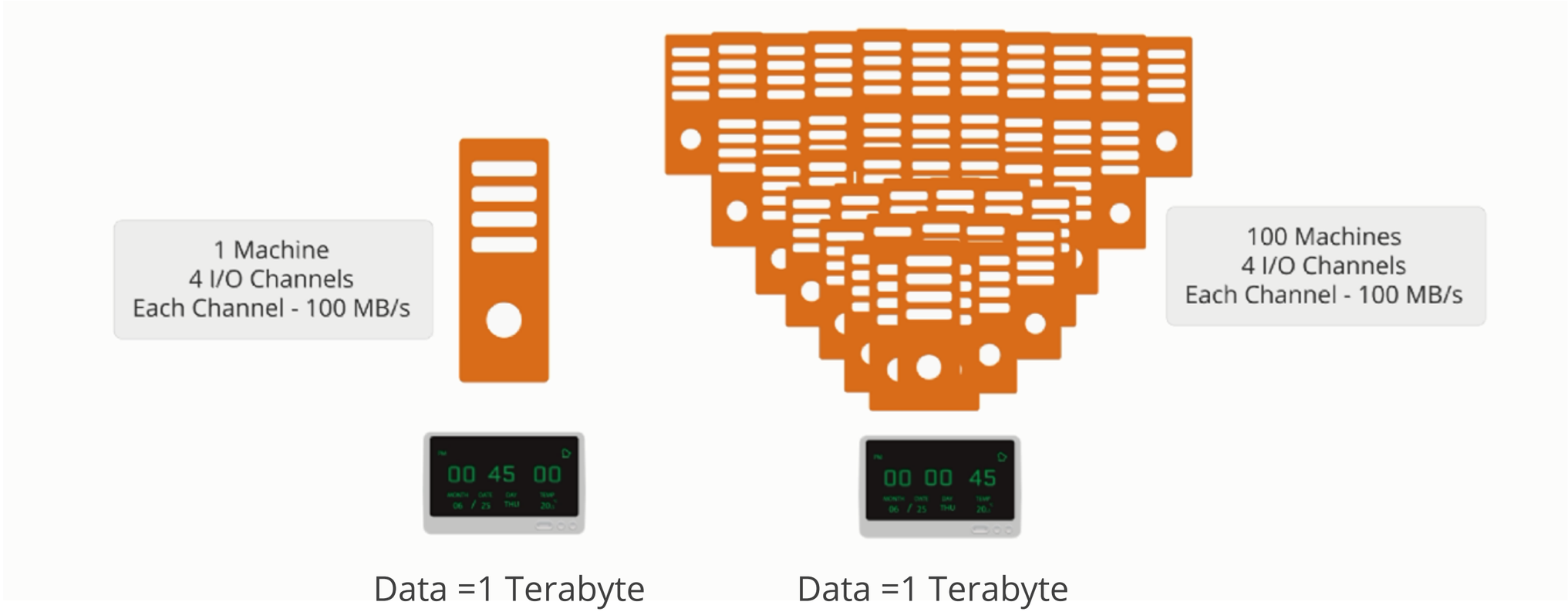
DATA AND
ARTIFICIAL INTELLIGENCE

Distributed Systems

Distributed Systems

A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages.

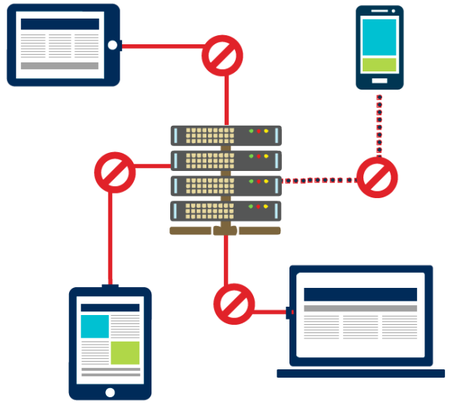
How Does a Distributed System Work?



In recent times, distributed systems have been replaced by Hadoop.

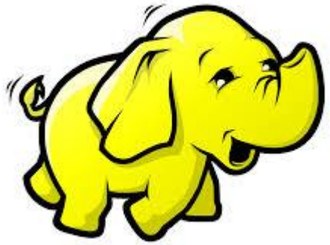
Challenges of Distributed Systems

Since, multiple computers are used in a distributed system, there are high chances of:



1. System failure
2. Limited bandwidth
3. High programming complexity

Any solution?



Hadoop

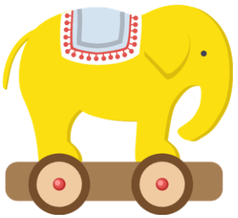
DATA AND
ARTIFICIAL INTELLIGENCE

Introduction to Hadoop

What Is Hadoop?

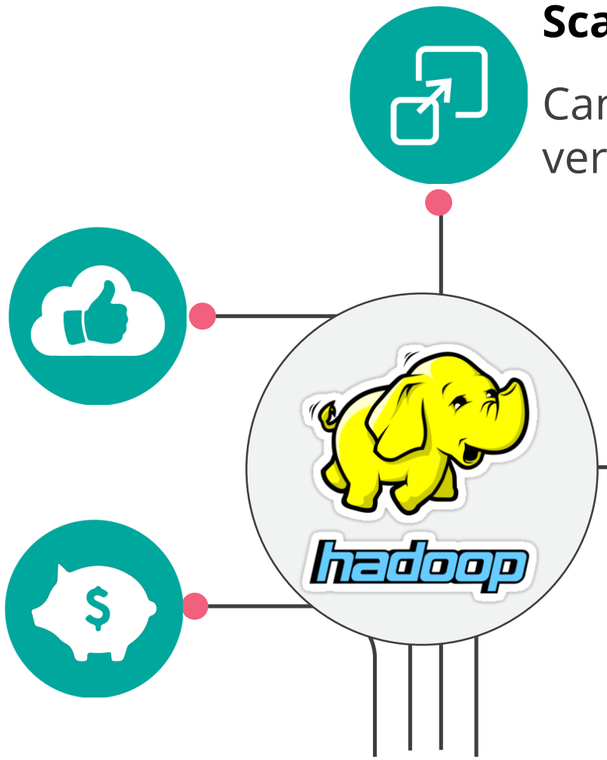
Hadoop is a framework that allows distributed processing of large datasets across clusters of commodity computers using simple programming models.

Doug Cutting discovered Hadoop and named it after his son's yellow toy elephant. It is inspired by the technical document published by Google.



Characteristics of Hadoop

The four key characteristics of Hadoop are:



Scalable

Can follow both horizontal and vertical scaling

Reliable

Stores copies of the data on different machines and is resistant to hardware failure

Economical

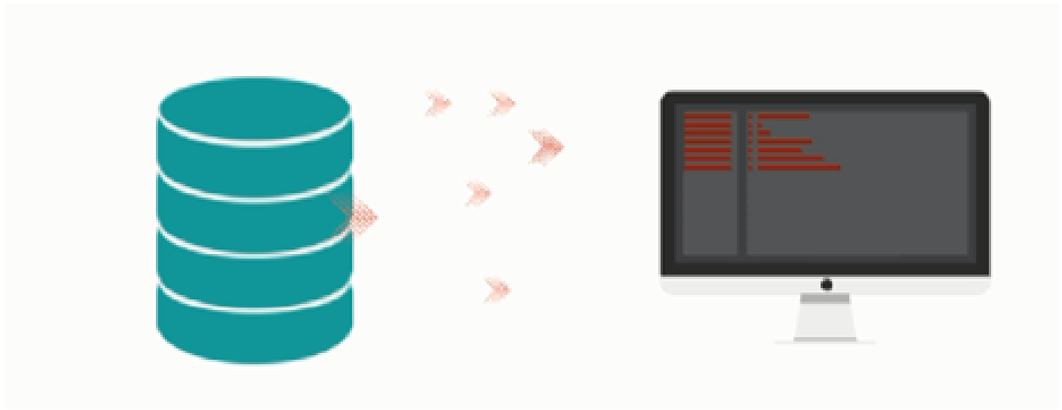
Can use ordinary computers for data processing

Flexible

Can store huge data and decide to use it later

Traditional Database Systems vs. Hadoop

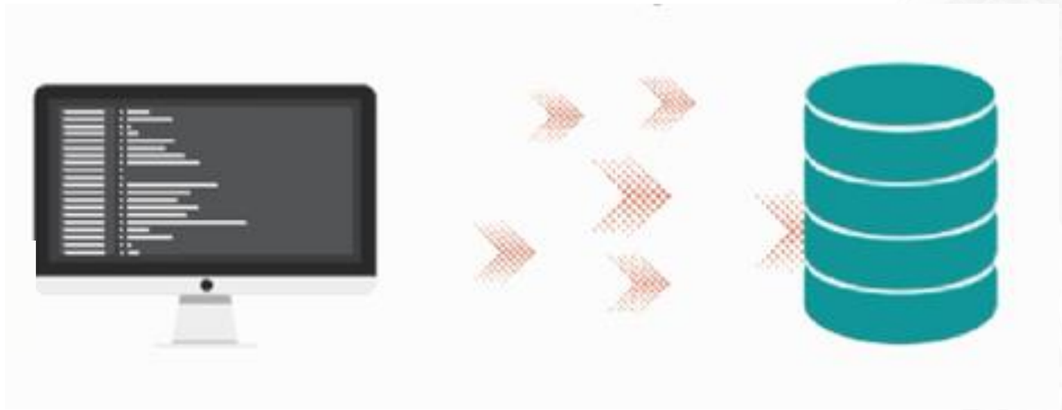
Traditional System



Data sent to the program

VS.

Hadoop



Program sent to the data

Analogy of Traditional System and Hadoop

Human brings food toward the mouth

vs.

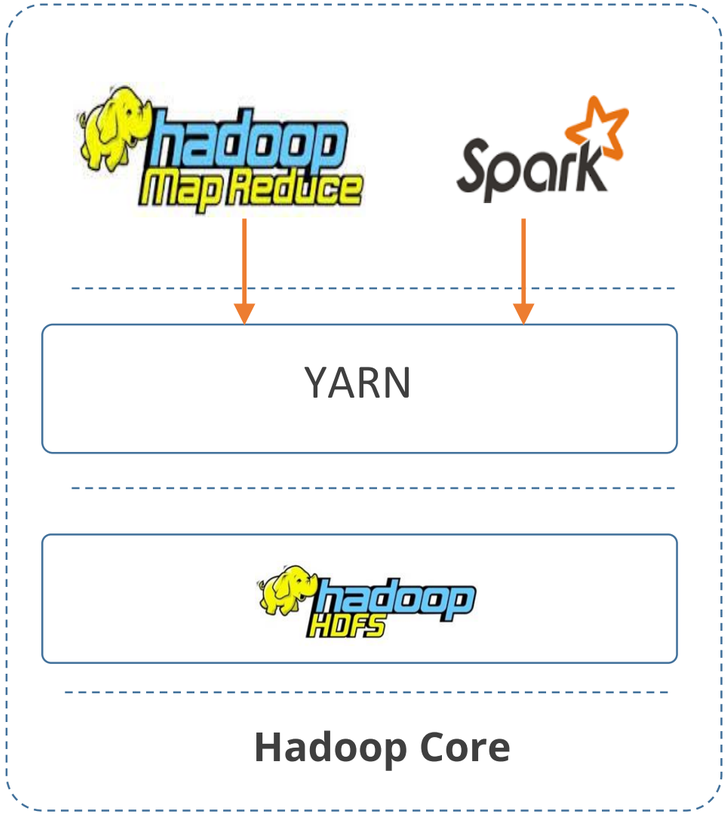
Tiger brings its mouth toward the food

Traditional Database Systems vs. Hadoop

RDBMS		HADOOP
Structured	Data Types	Multi and Unstructured
Limited, No Data Processing	Processing	Processing coupled with Data
Standards and Structured	Governance	Loosely Structured
Required On Write	Schema	Required On Read
Reads are Fast	Speed	Writes are Fast
Software License	Cost	Support Only
Known Entity	Resources	Growing, Complexities, Wide
OLTP Complex ACID Transactions Operational Data Store	Best Fit Use	Data Discovery Processing Unstructured Data Massive Storage/Processing

Hadoop Core Components

Data Processing



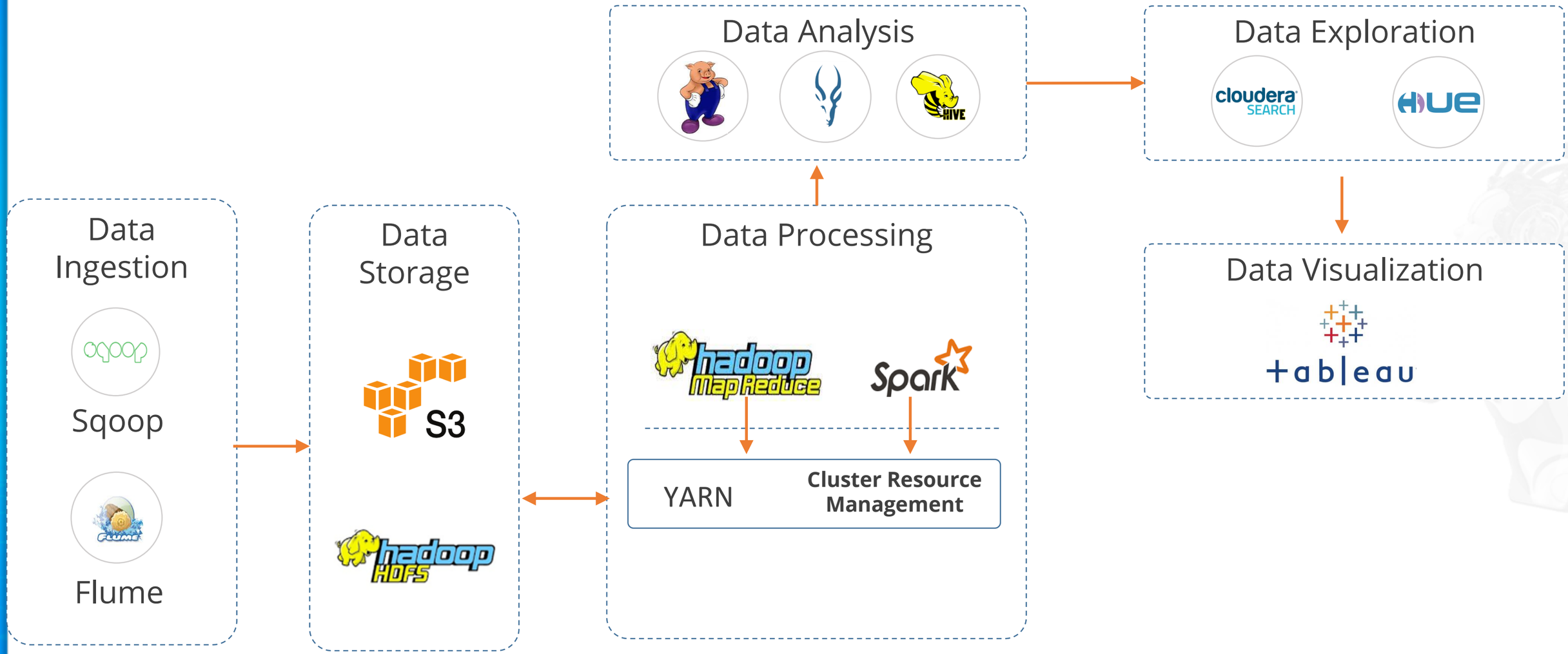
Resource Management

Storage

DATA AND
ARTIFICIAL INTELLIGENCE

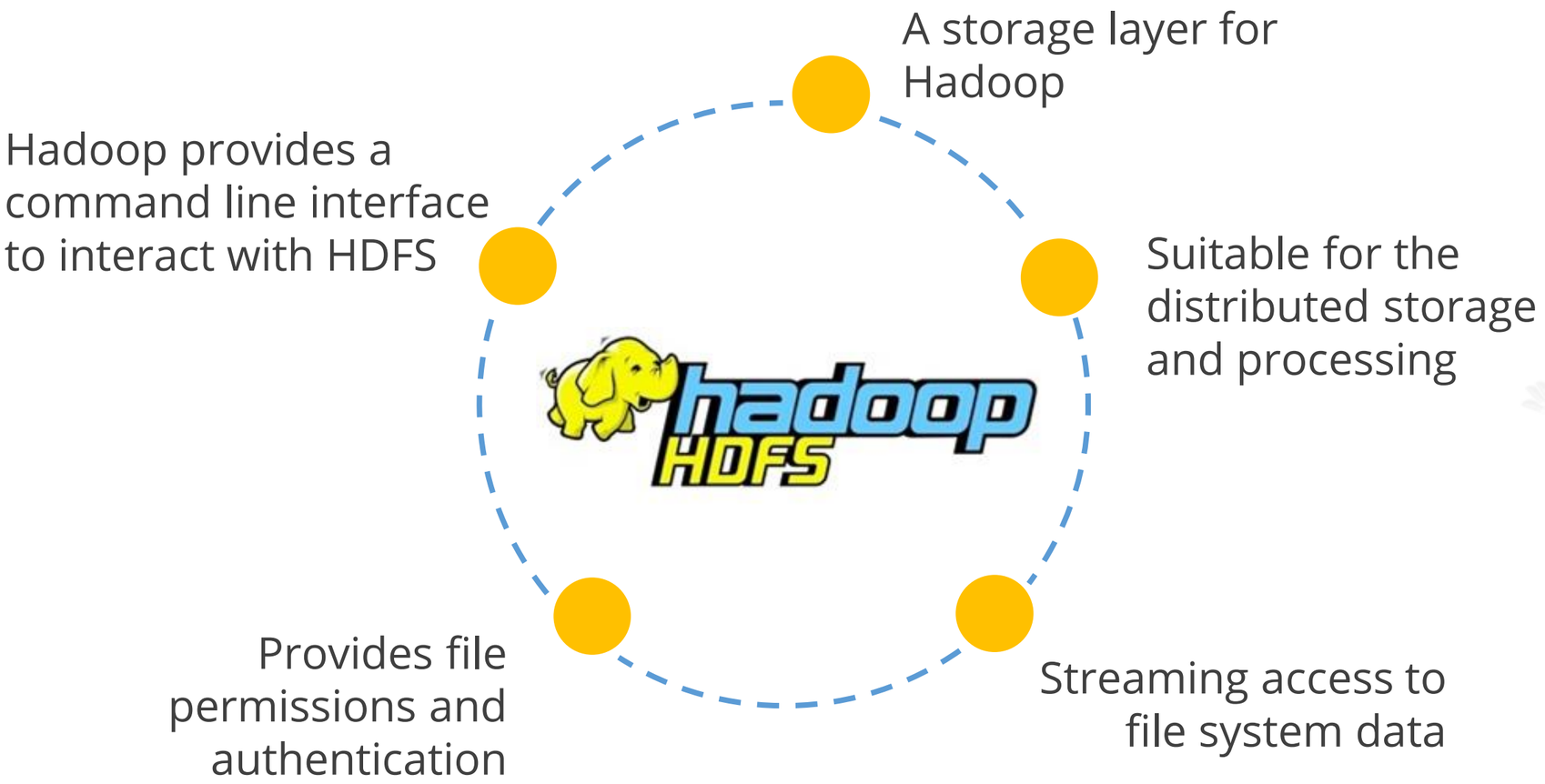
Components of Hadoop Ecosystem

Components of Hadoop Ecosystem



Components of Hadoop Ecosystem

HDFS (HADOOP DISTRIBUTED FILE SYSTEM)



Components of Hadoop Ecosystem

HBase

Stores data in HDFS

A NoSQL database or non-relational database

Mainly used when you need random, real-time, read/write access to your Big Data

Provides support to high volume of data and high throughput

Table can have thousands of columns

Components of Hadoop Ecosystem

SQOOP

- Sqoop is a tool designed to transfer data between Hadoop and relational database servers.

- It is used to import data from relational databases such as Oracle and MySQL to HDFS and export data from HDFS to relational databases.

Components of Hadoop Ecosystem

FLUME

If you want to ingest event data such as, streaming data, sensor data, or log files, then you can use Flume.

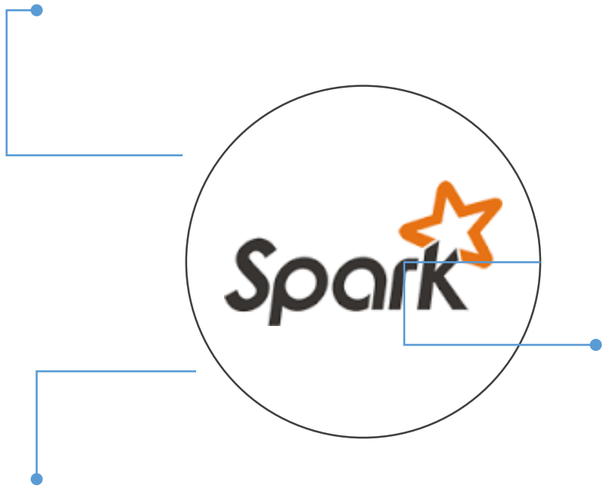
A distributed service for ingesting streaming data

Ideally suited for event data from multiple systems

Components of Hadoop Ecosystem

SPARK

An open source cluster computing framework

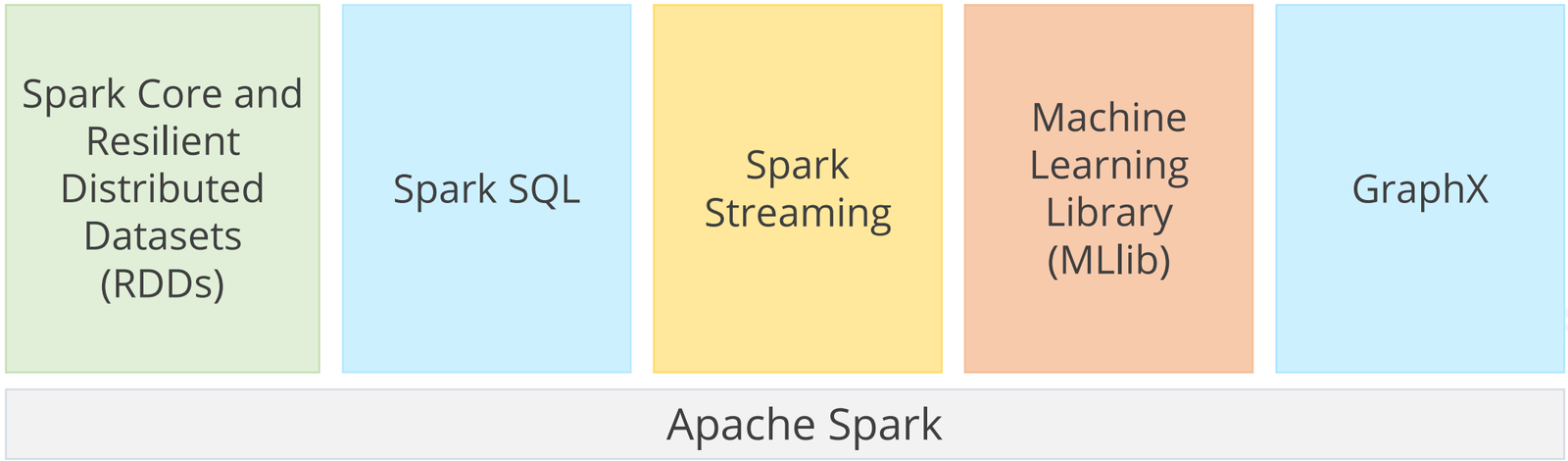


Provides 100 times faster performance than Map-Reduce

Supports machine learning, business intelligence, streaming, and batch processing

Components of Hadoop Ecosystem

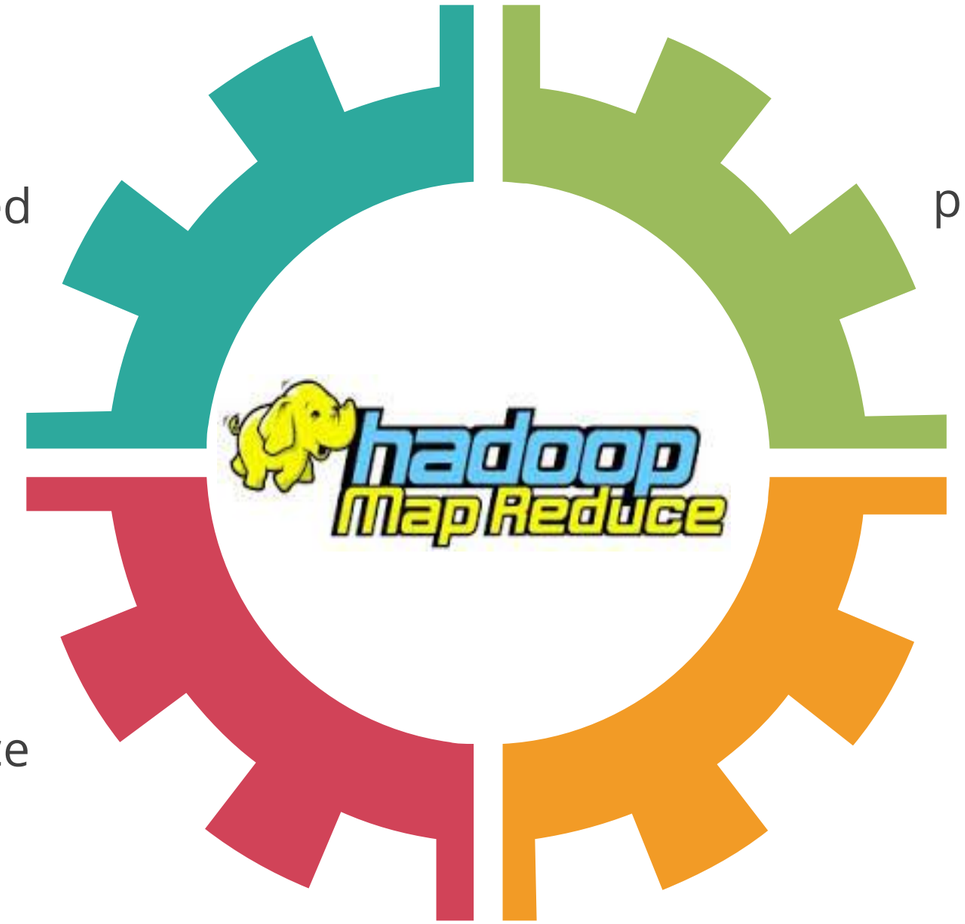
SPARK: COMPONENTS



Components of Hadoop Ecosystem

HADOOP MAP-REDUCE

Commonly used



The original Hadoop processing engine which is primarily Java based

An extensive and mature fault tolerance framework

Based on the map and reduce programming model

Components of Hadoop Ecosystem

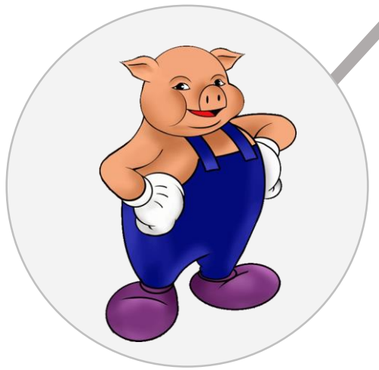
PIG

An open source dataflow system

Converts pig script to Map-Reduce code

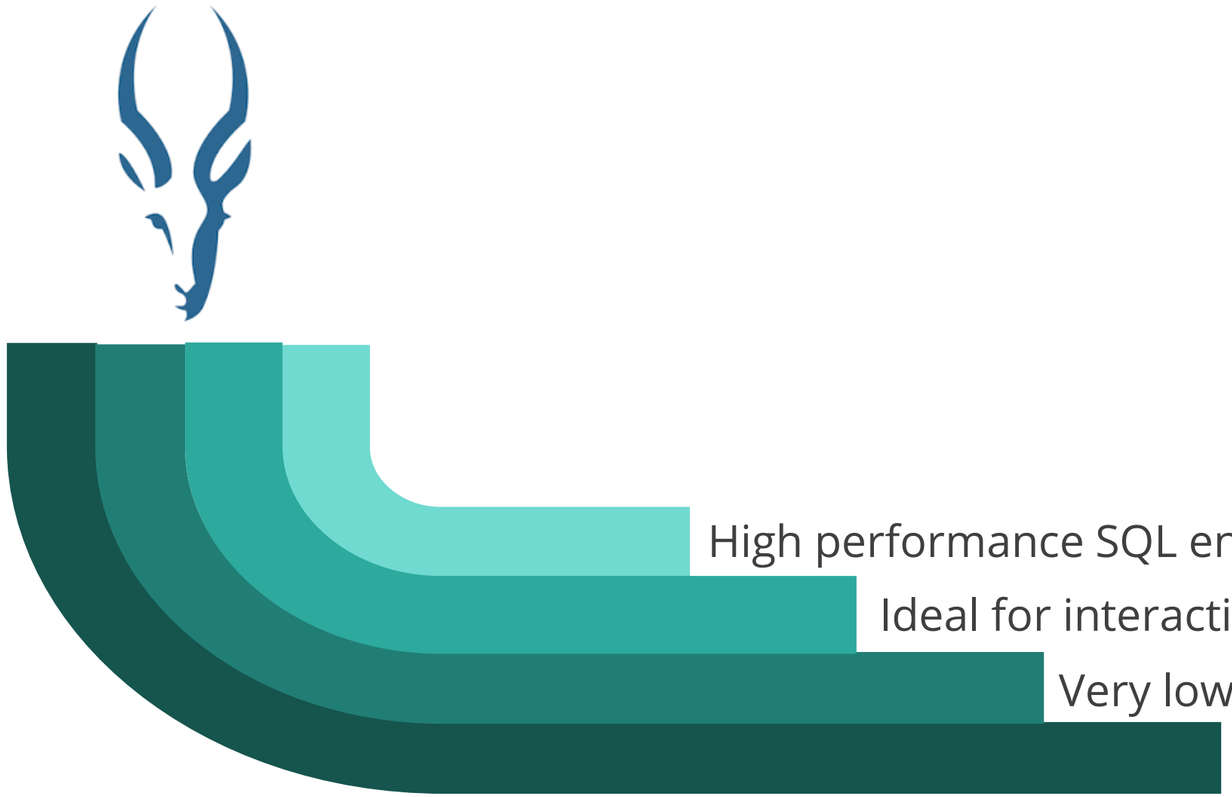
An alternative to writing Map-Reduce code

Best for ad-hoc queries such as join and filter



Components of Hadoop Ecosystem

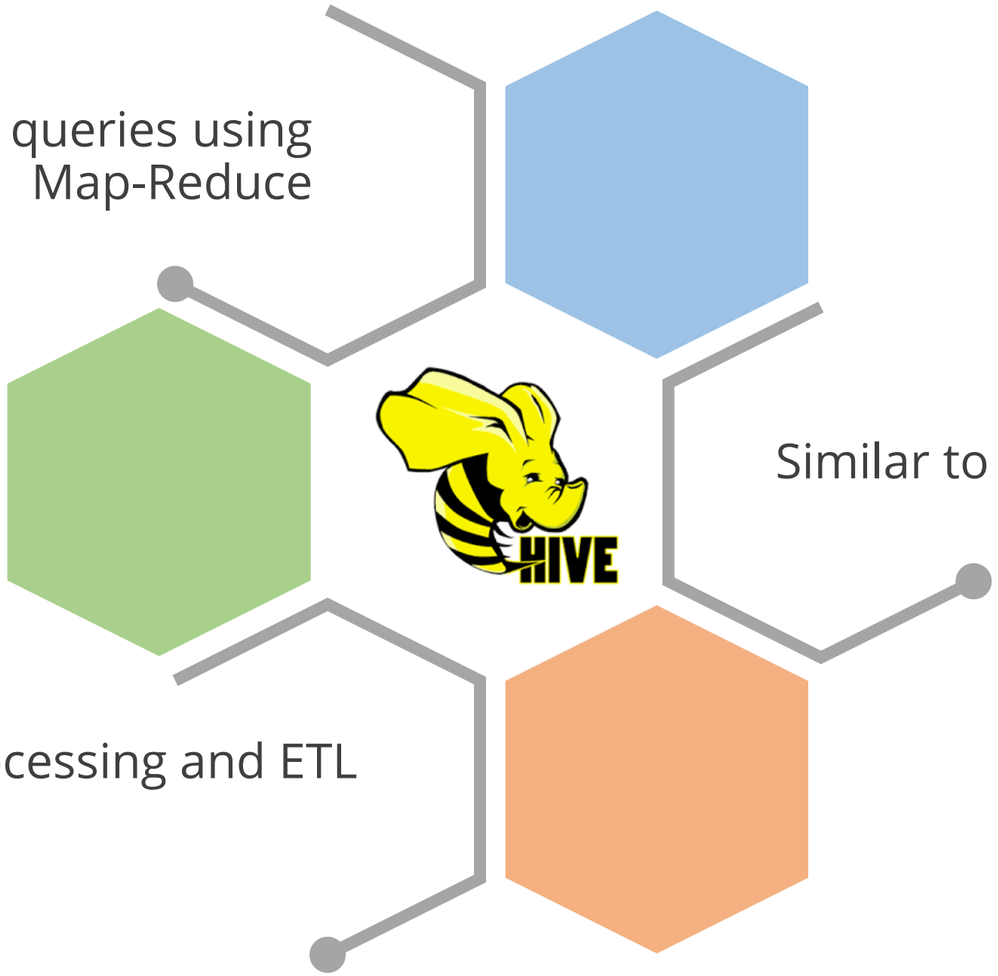
IMPALA



- High performance SQL engine which runs on Hadoop cluster
- Ideal for interactive analysis
- Very low latency – measured in milliseconds
- Supports a dialect of SQL (Impala SQL)

Components of Hadoop Ecosystem

HIVE



Components of Hadoop Ecosystem

CLOUDERA SEARCH

One of Cloudera's near-real-time access products

Eliminates the need to move large datasets across infrastructures to address business tasks

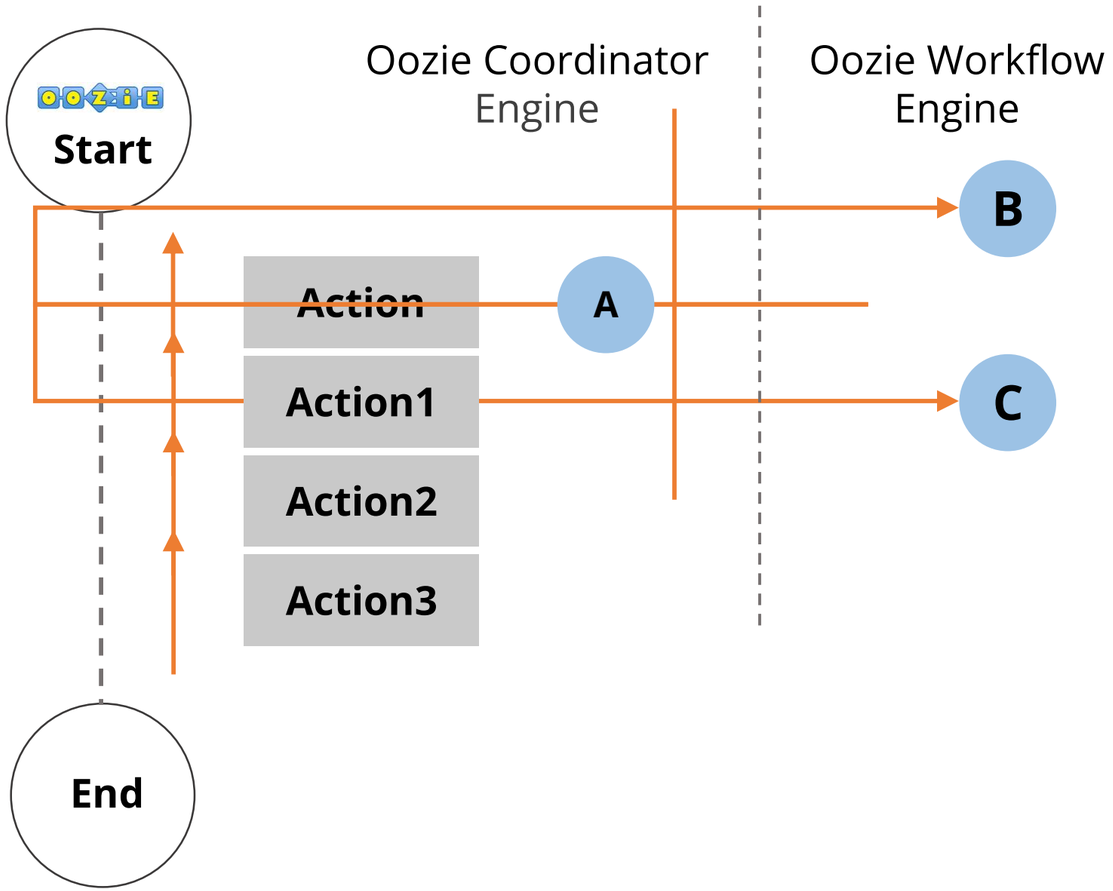
Enables nontechnical users to search and explore data stored in or ingested into Hadoop and HBase

A fully integrated data processing platform

Components of Hadoop Ecosystem

OOZIE

Oozie is a workflow or coordination system used to manage the Hadoop jobs



Components of Hadoop Ecosystem

HUE (HADOOP USER EXPERIENCE)

Hue is an acronym for Hadoop User Experience

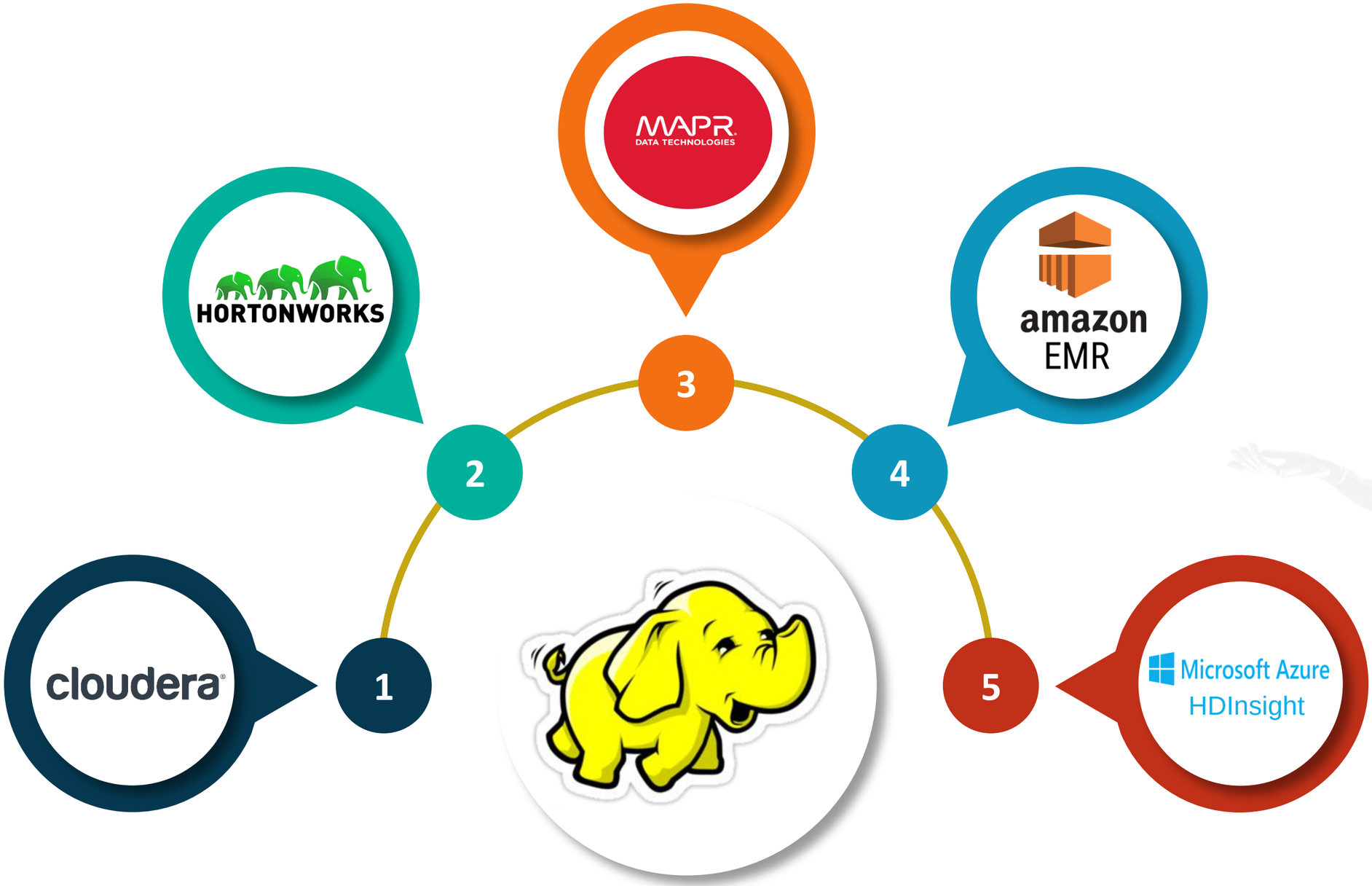
It provides SQL editors for Hive, Impala, MySQL, Oracle, PostgreSQL, Spark SQL, and Solr SQL

Hue is an open source Web interface for analyzing data with Hadoop

DATA AND
ARTIFICIAL INTELLIGENCE

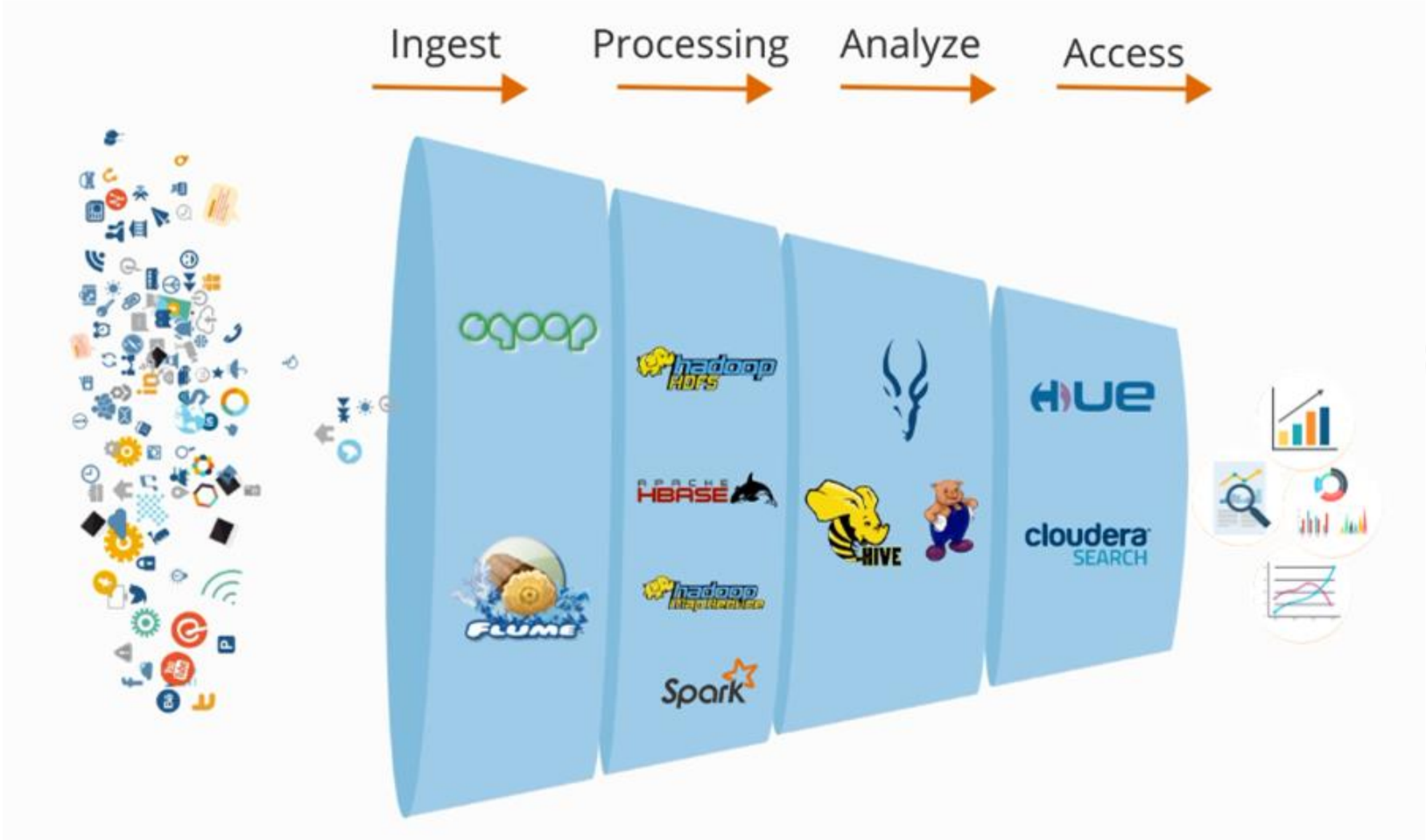
Commercial Hadoop Distributions

Various Commercial Hadoop Distributions



Big Data Processing

Components of Hadoop ecosystem work together to process big data.
There are four stages of big data processing:



Assisted Practice

Walk-Through of the Simplilearn Cloud Lab

Duration: 10 mins

Problem Statement: In this demonstration, we will walk you through the Simplilearn cloud lab.

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:

- Describe the concepts of Big Data
- Explain Hadoop and how it addresses Big Data challenges
- Describe the components of Hadoop Ecosystem

DATA AND
ARTIFICIAL INTELLIGENCE

Knowledge Check

Knowledge Check

1

Which of the following is a source of unstructured data?

a. Data from social media websites

b. Transactional data in Amazon's database

c. Web and server logs

d. All of the above

Knowledge Check

1

Which of the following is a source of unstructured data?

a. Data from social media websites

b. Transactional data in Amazon's database

c. Web and server logs

d. All of the above

The correct answer is **a.**

Unstructured data comprises of data that is usually not easily searchable, including formats like audio, video, and social media postings.

Knowledge Check

2

A bank wants to process 1000 transactions per second.
Which one of the following Vs reflects this real-world use case?

a. Volume

b. Variety

c. Velocity

d. Veracity

Knowledge Check

2

A bank wants to process 1000 transactions per second.
Which one of the following Vs reflects this real-world use case?

a. Volume

b. Variety

c. Velocity

d. Veracity

The correct answer is **c.**

Velocity is the frequency of incoming data that needs to be processed. Given use case is an example of an application that handles the velocity of data.

Knowledge Check

3

Why has popularity of big data increased tremendously in the recent years?

a. Due to increased volume of data

b. Big data is an open source

c. Abundance of unstructured data

d. None of the above

Knowledge Check

3

Why has popularity of big data increased tremendously in the recent years?

a. Due to increased volume of data

b. Big data is an open source

c. Abundance of unstructured data

d. None of the above

The correct answer is **a.**

Unstructured data is growing at astronomical rates, contributing to the big data deluge that's sweeping across enterprise data storage environments.

Knowledge Check

4

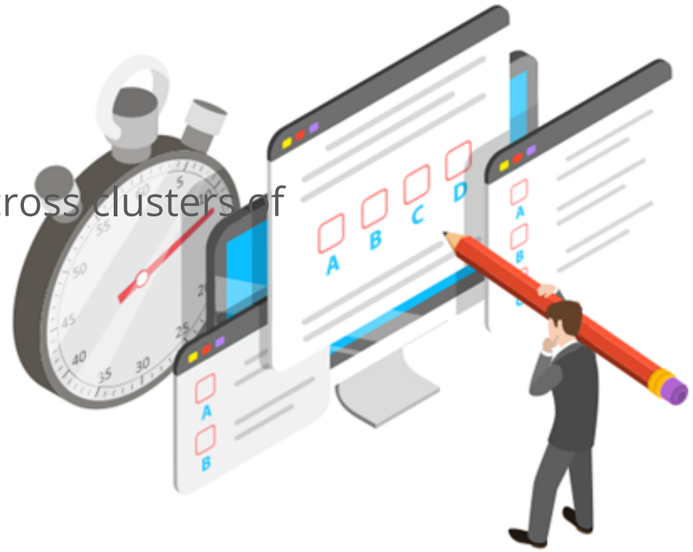
What is Hadoop?

a. It is an in-memory tool used in Mahout algorithm computing.

b. It is a computing framework used for resource management.

c. It is a framework that allows distributed processing of large datasets across clusters of commodity computers using a simple programming model.

d. It is a search and analytics tool that provides access to analyze data.



Knowledge Check

4

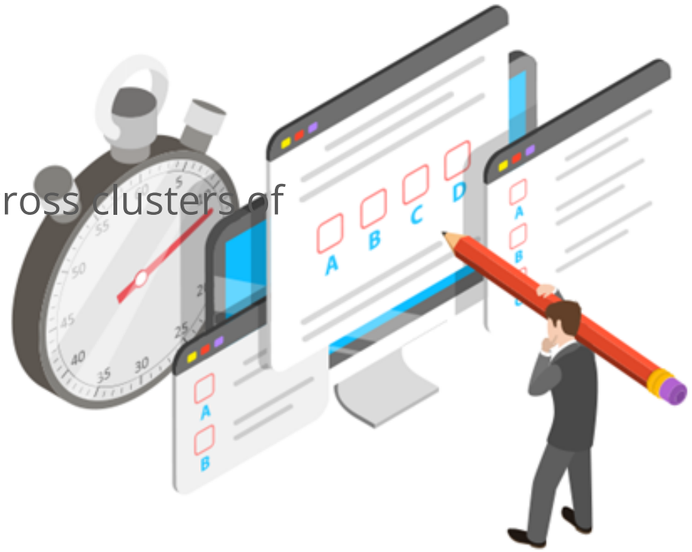
What is Hadoop?

a. It is an in-memory tool used in Mahout algorithm computing.

b. It is a computing framework used for resource management.

c. It is a framework that allows distributed processing of large datasets across clusters of commodity computers using a simple programming model.

d. It is a search and analytics tool that provides access to analyze data.



The correct answer is **c.**

Hadoop is a framework that allows distributed processing of large datasets across clusters of commodity computers using a simple programming model.

Knowledge Check

5

Which of the following is a column-oriented NoSQL database that runs on top of HDFS?

a. MongoDB

b. Flume

c. Ambari

d. HBase

Knowledge Check

5

Which of the following is a column-oriented NoSQL database that runs on top of HDFS?

a. MongoDB

b. Flume

c. Ambari

d. HBase

The correct answer is **d.**

Apache HBase is a NoSQL database that runs on top of Hadoop as a distributed and scalable big data store.

Knowledge Check

6

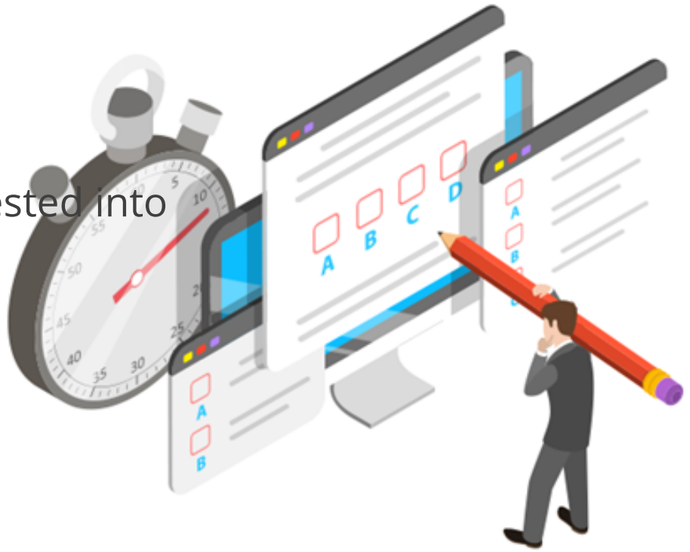
Scoop is used to _____.

a. Import data from relational databases to Hadoop HDFS and export from Hadoop file system to relational databases

b. Execute queries using Map-Reduce

c. Enable nontechnical users to search and explore data stored in or ingested into Hadoop and HBase

d. Stream event data from multiple systems



Knowledge Check

6

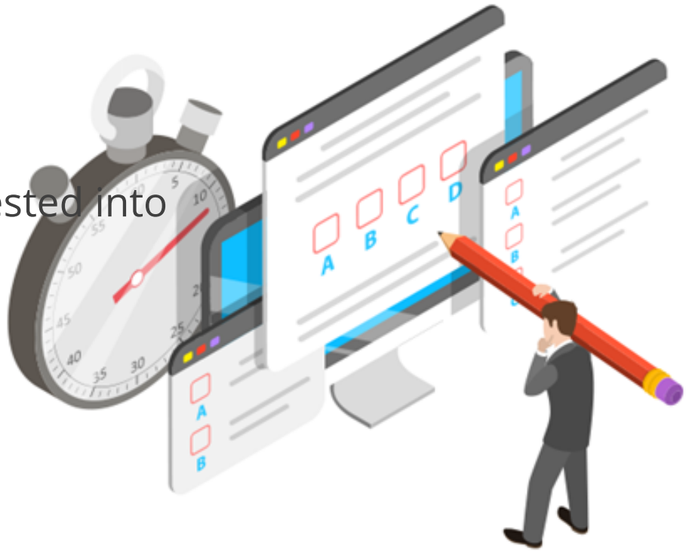
Scoop is used to _____.

a. Import data from relational databases to Hadoop HDFS and export from Hadoop file system to relational databases

b. Execute queries using Map-Reduce

c. Enable nontechnical users to search and explore data stored in or ingested into Hadoop and HBase

d. Stream event data from multiple systems



The correct answer is **a.**

Scoop is used to import data from relational databases to Hadoop HDFS and export from Hadoop file system to relational databases.

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Hadoop Architecture, Distributed Storage (HDFS), and YARN

Learning Objectives

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

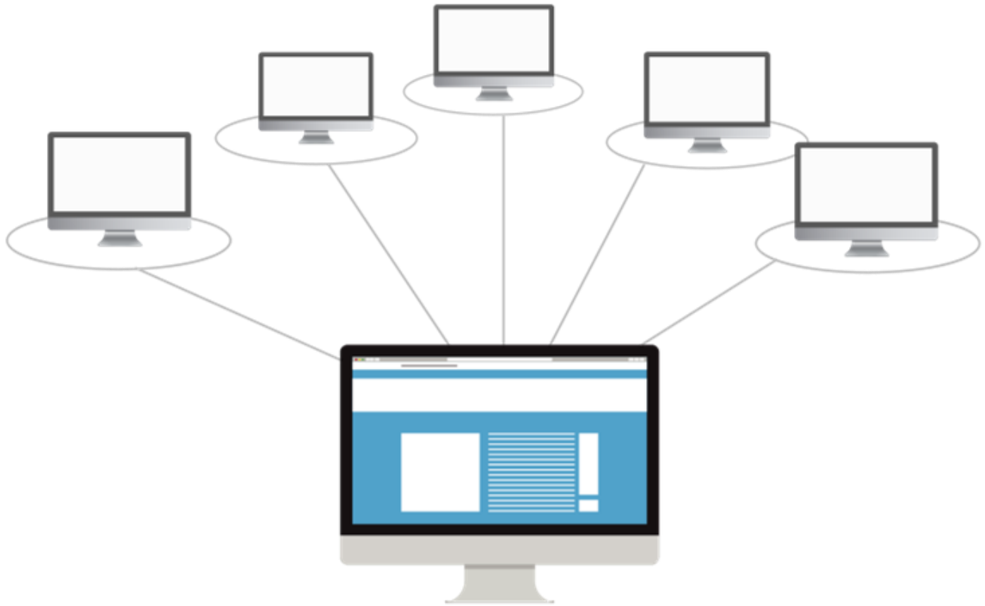
- Understand how Hadoop Distributed File System (HDFS) stores data across a cluster
- Illustrate the HDFS architecture and its components
- Demonstrate the use of HDFS Command Line Interface (CLI)
- Illustrate the YARN architecture and its components
- Demonstrate how to use Hue, YARN Web UI, and the YARN command to monitor the cluster

DATA AND
ARTIFICIAL INTELLIGENCE

Hadoop Distributed File System (HDFS)

What Is HDFS?

HDFS is a distributed file system that provides access to data across Hadoop clusters.



Manages and supports analysis of very large volumes of Big Data

Challenges of Traditional Systems

In the traditional system, storing and retrieving volumes of data had three major issues:

Cost

$10,000 to $14,000 per terabyte

Speed

Search and analysis is time-consuming

Reliability

Fetching data is difficult

Need for HDFS

HDFS resolves all the three major issues of the traditional file system.

Cost

Zero licensing and support costs

Reliability

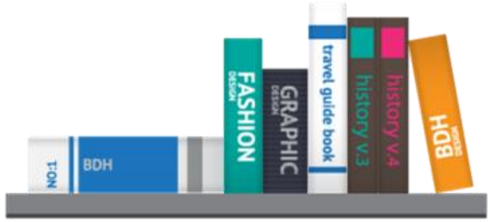
HDFS copies the data multiple times

Speed

Hadoop clusters read/write more than one terabyte of data in a second

Regular File System vs. HDFS

A patron gifts his popular books collection to a college library.



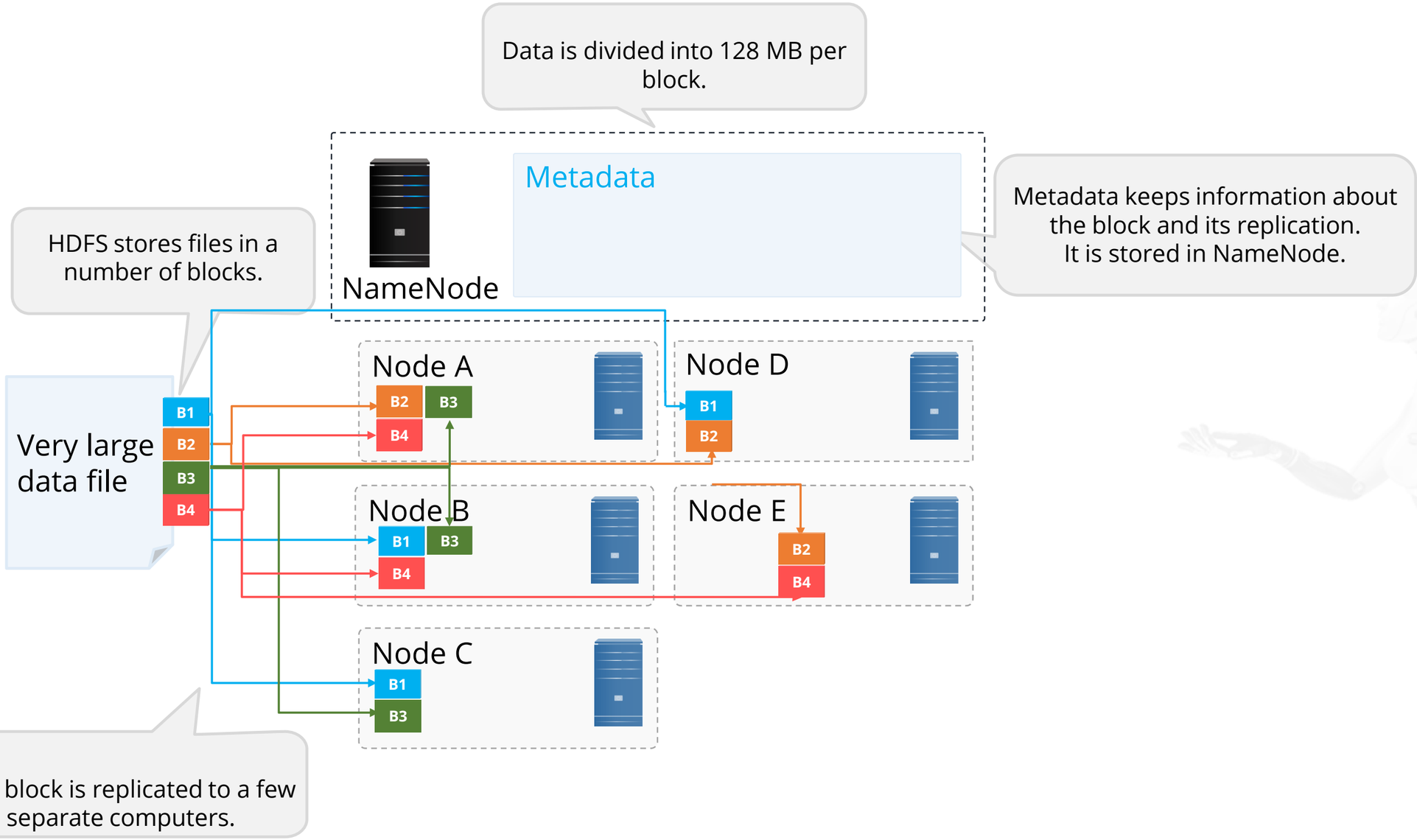
The librarian decides to arrange the books on a small rack.

Also, he distributes multiple copies of each book on other racks.

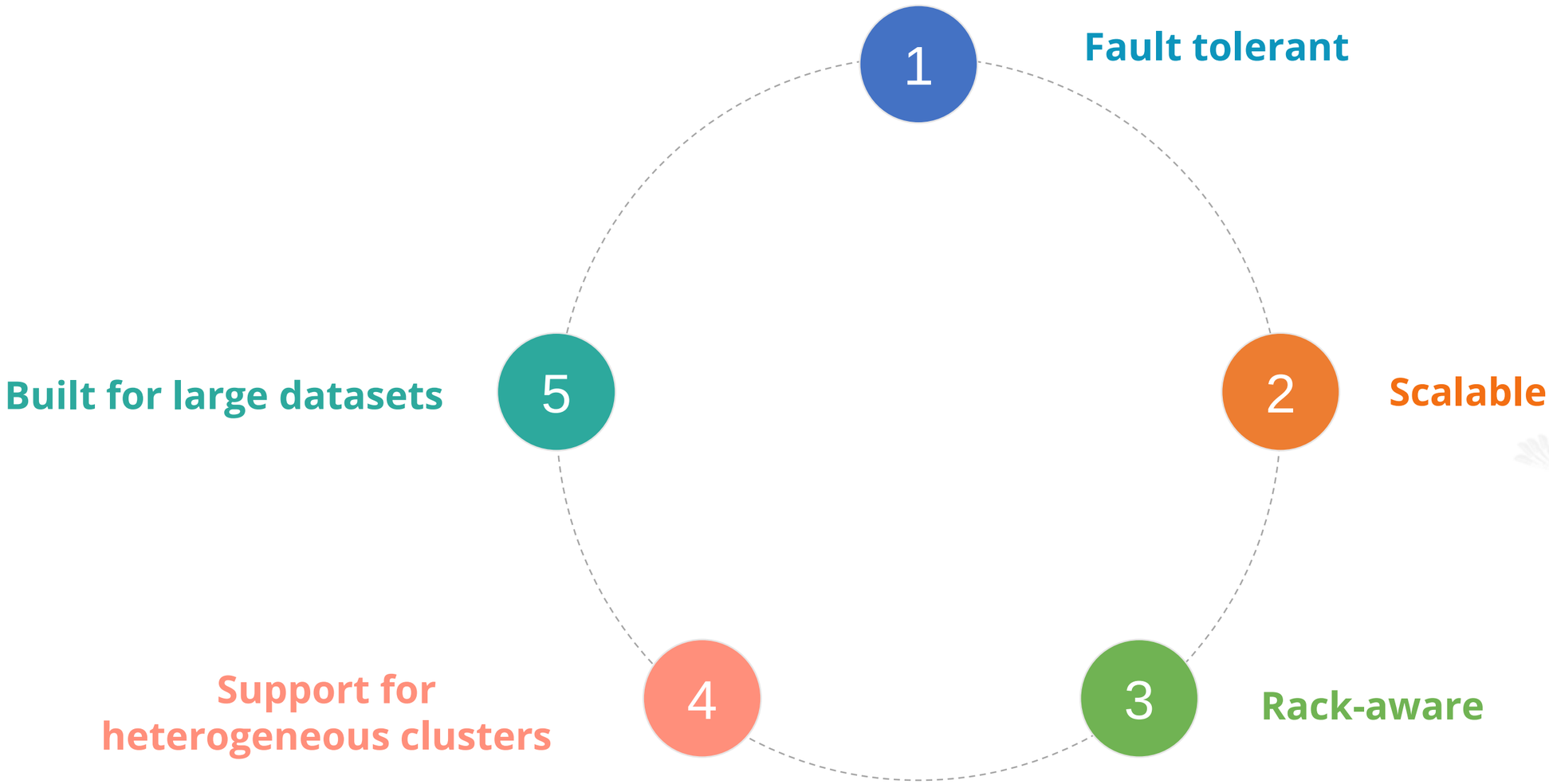
Regular File System vs. HDFS

Regular File System	HDFS
Size of Data	
	
51 bytes—small block of data	128 MB—large block of data
Access to Large Data	
	
Suffers from disk I/O problems primarily because of multiple seek operations	Reads huge data sequentially in a single seek operation

HDFS Storage



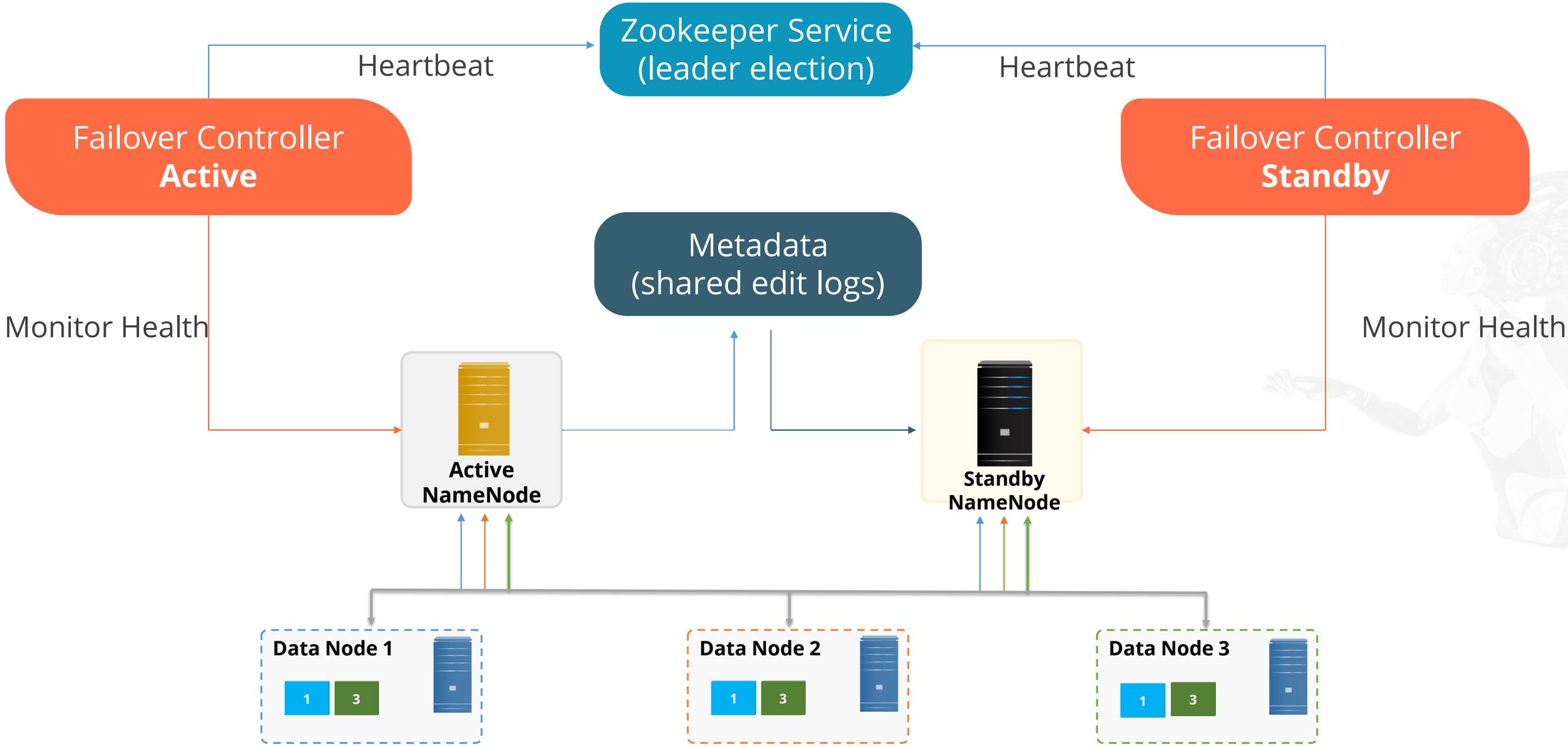
Characteristics of HDFS



DATA AND
ARTIFICIAL INTELLIGENCE

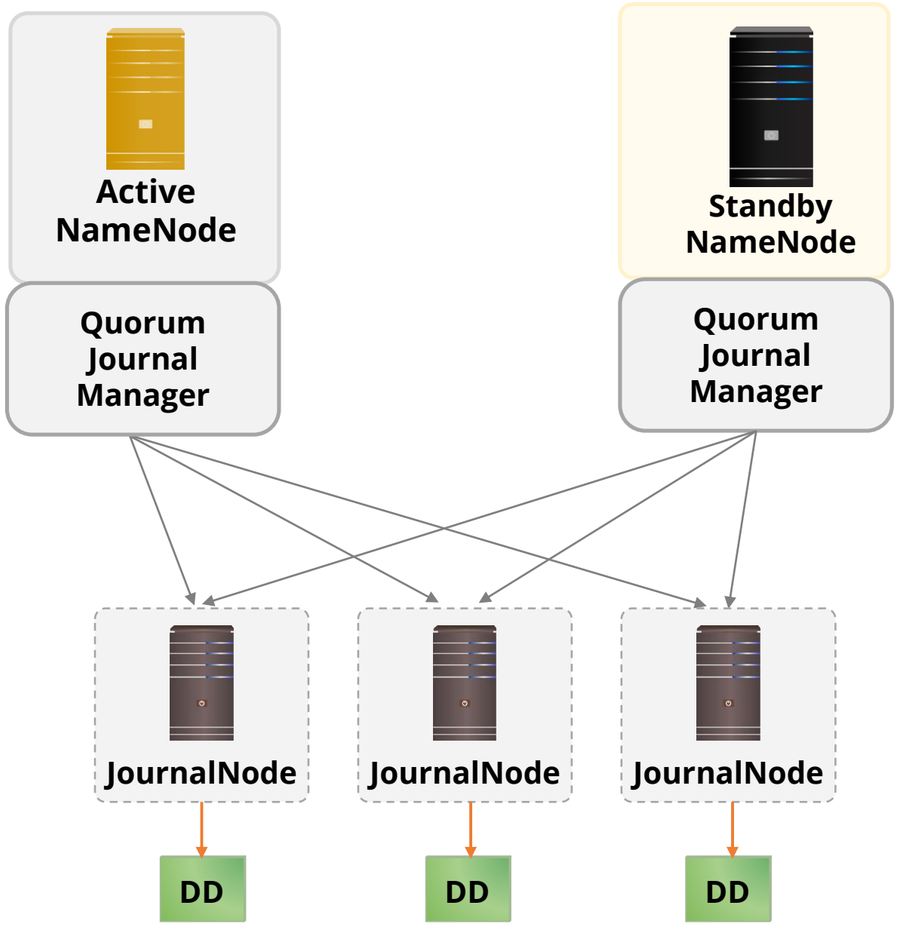
HDFS Architecture and Components

HDFS High Availability Architecture

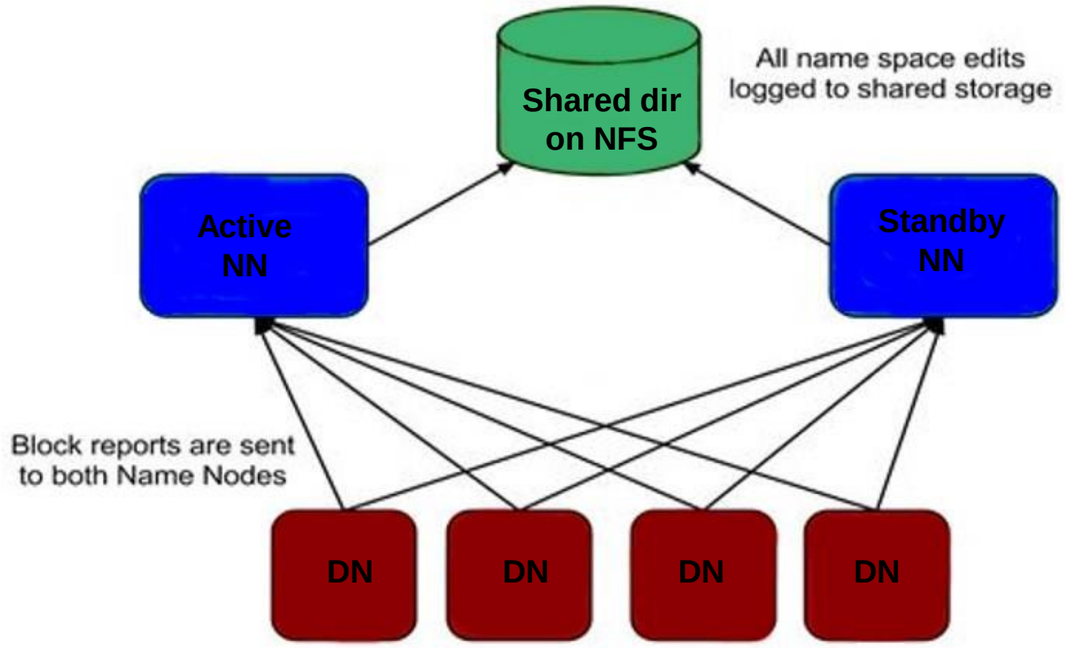


Types of HDFS HA Architecture

Quorum-based Storage

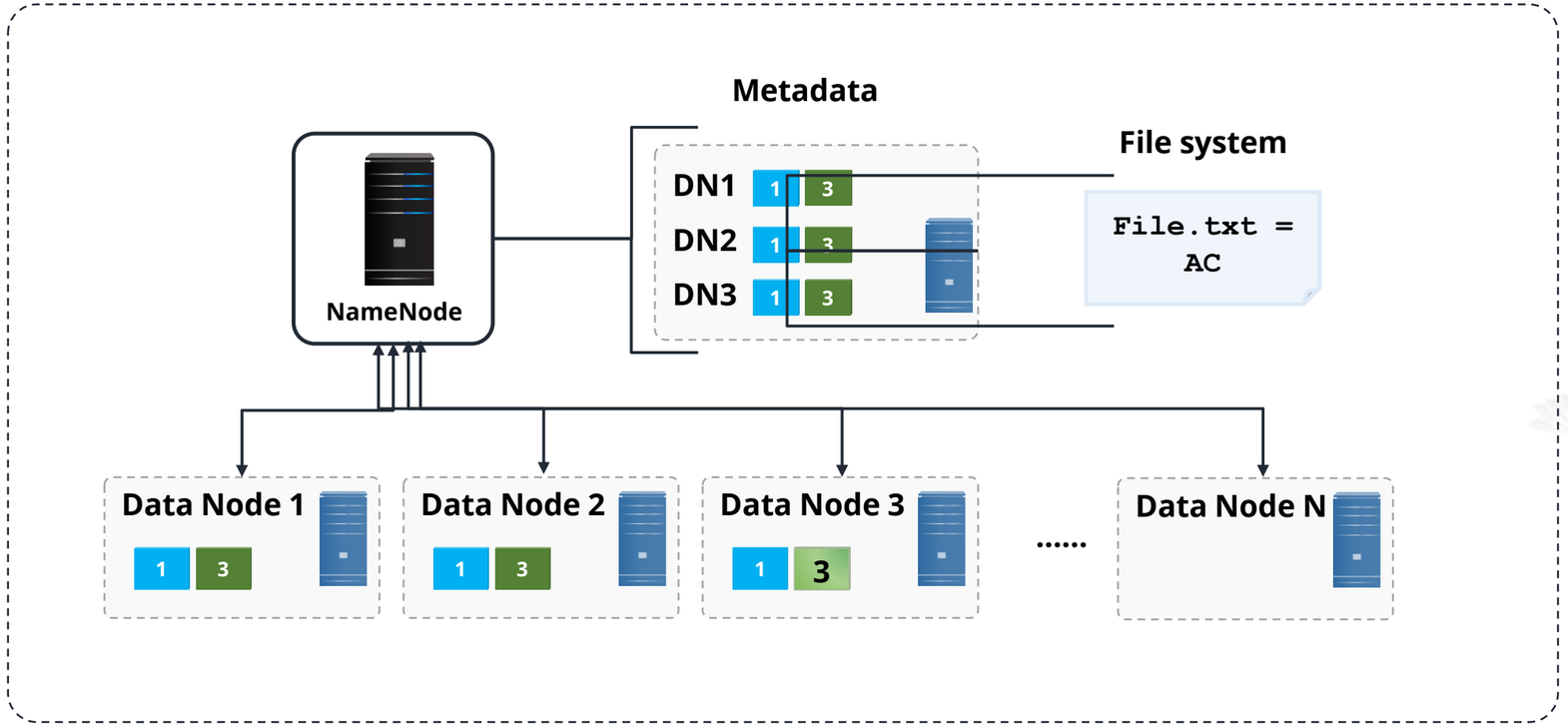


Shared storage using NFS



HDFS Component: NameNode

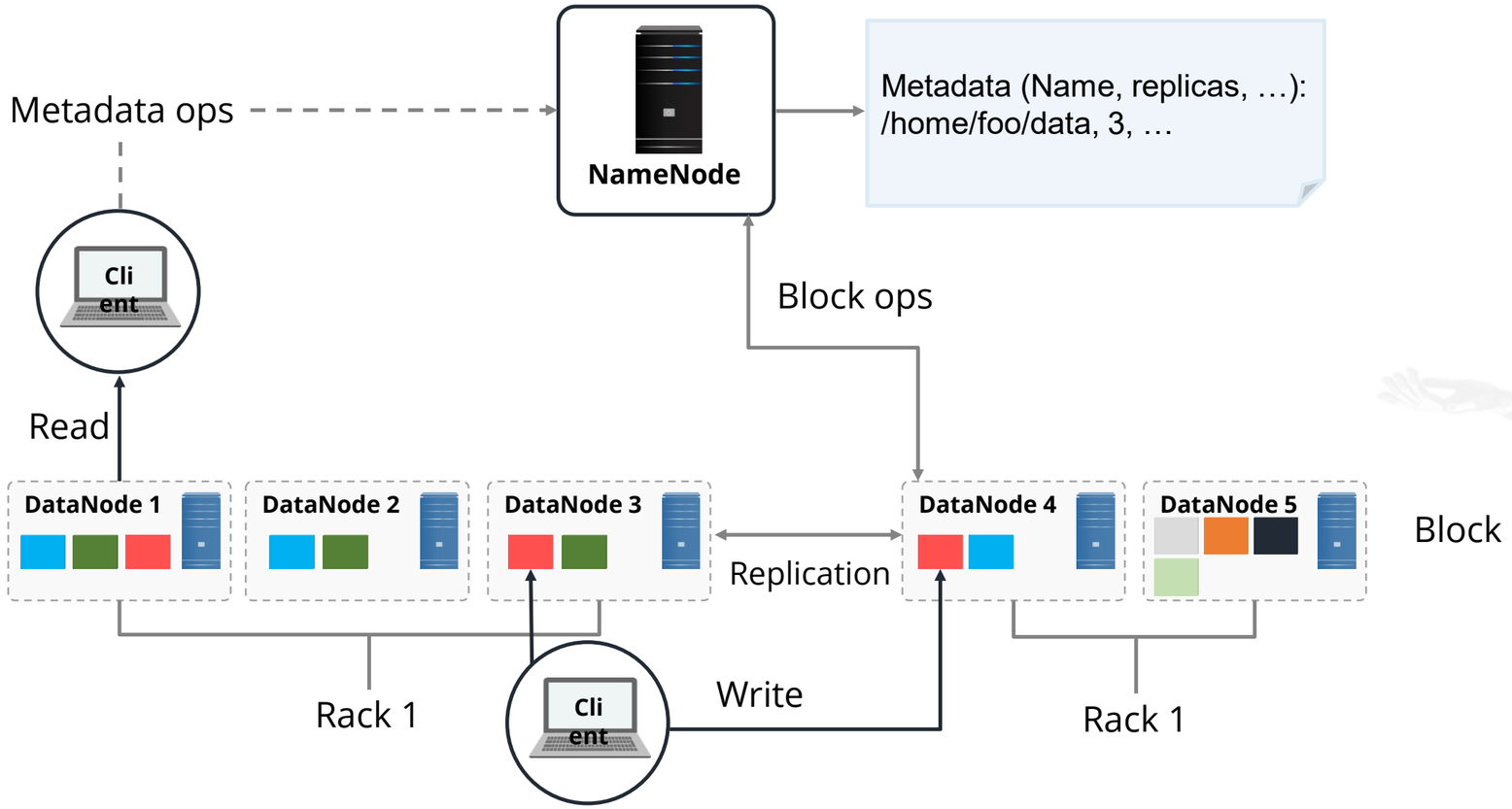
The HDFS components comprise different servers like NameNode, DataNode, and Secondary NameNode.



The NameNode server—core component of an HDFS cluster

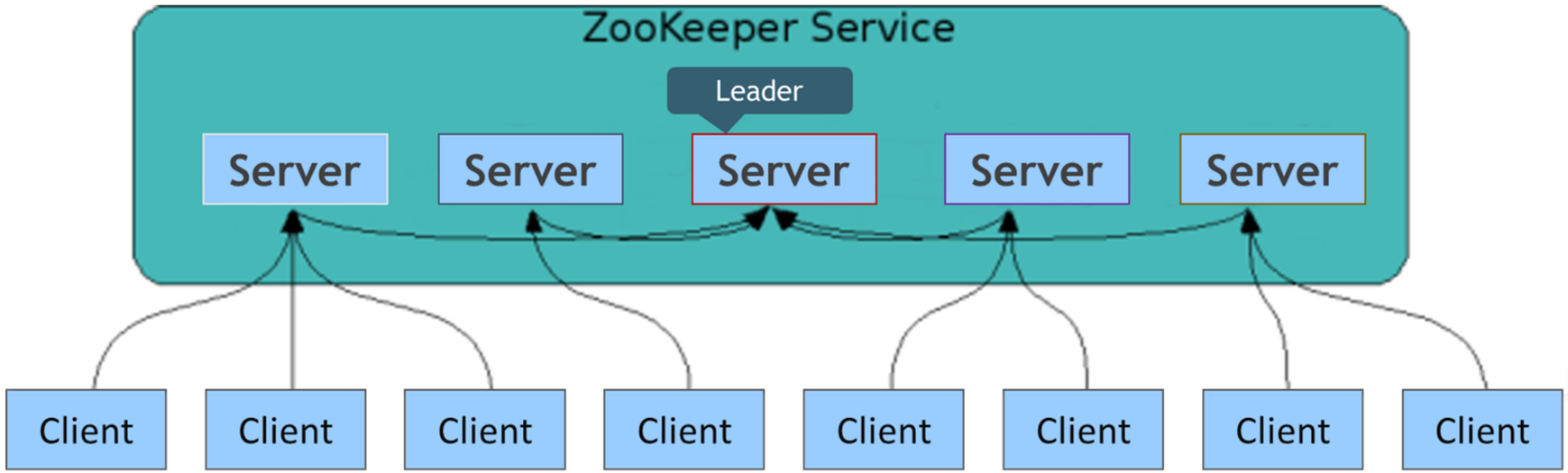
HDFS Component: DataNode

The DataNode is a multiple instance server.
The DataNode servers are responsible for storing and maintaining the data blocks.



HDFS Component: Zookeeper

ZooKeeper allows distributed processes to coordinate with each other through a shared hierarchical namespace.

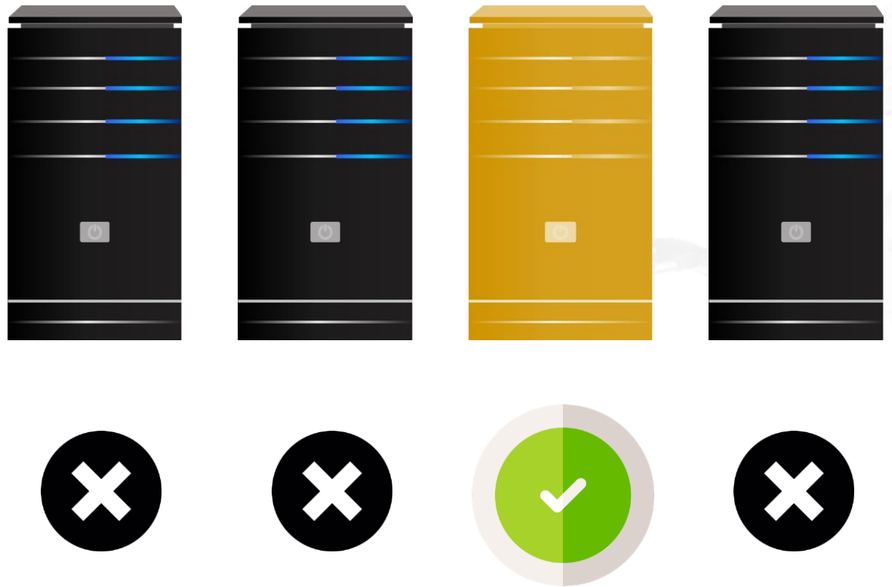


The ZooKeeper implementation is simple and replicated which puts a premium on high performance, high availability, and a strictly ordered access.

HDFS Component: Zookeeper

Automatic HDFS failover relies on ZooKeeper for the following:

Failure detection

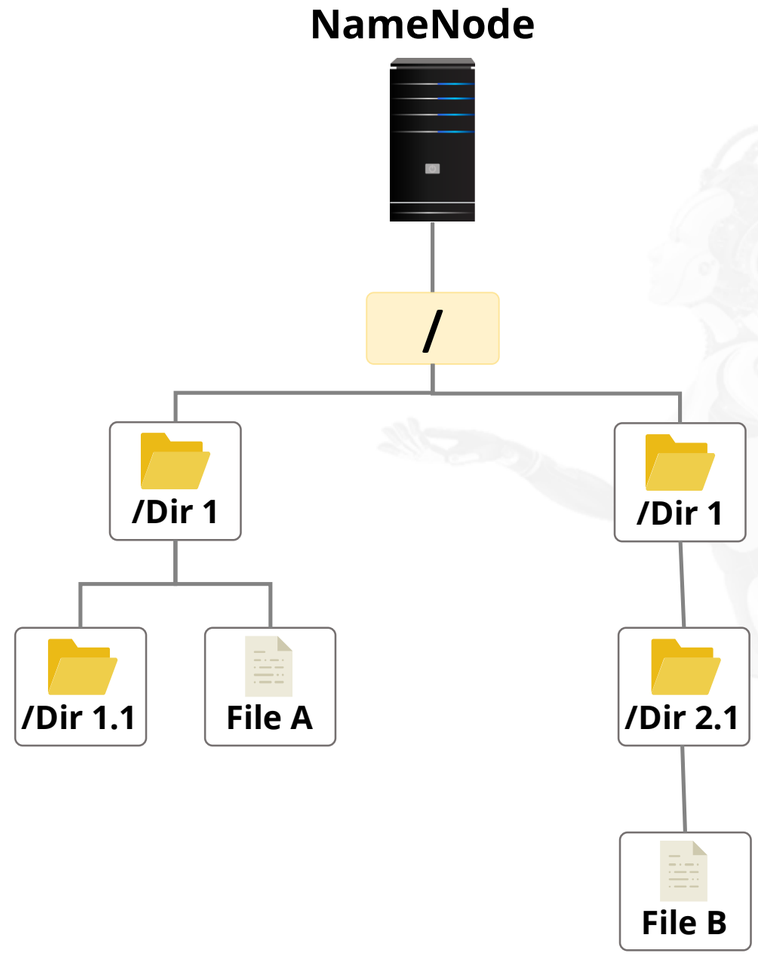


Active NameNode election

HDFS Component: File System Namespace

The characteristics of HDFS file system are as follows:

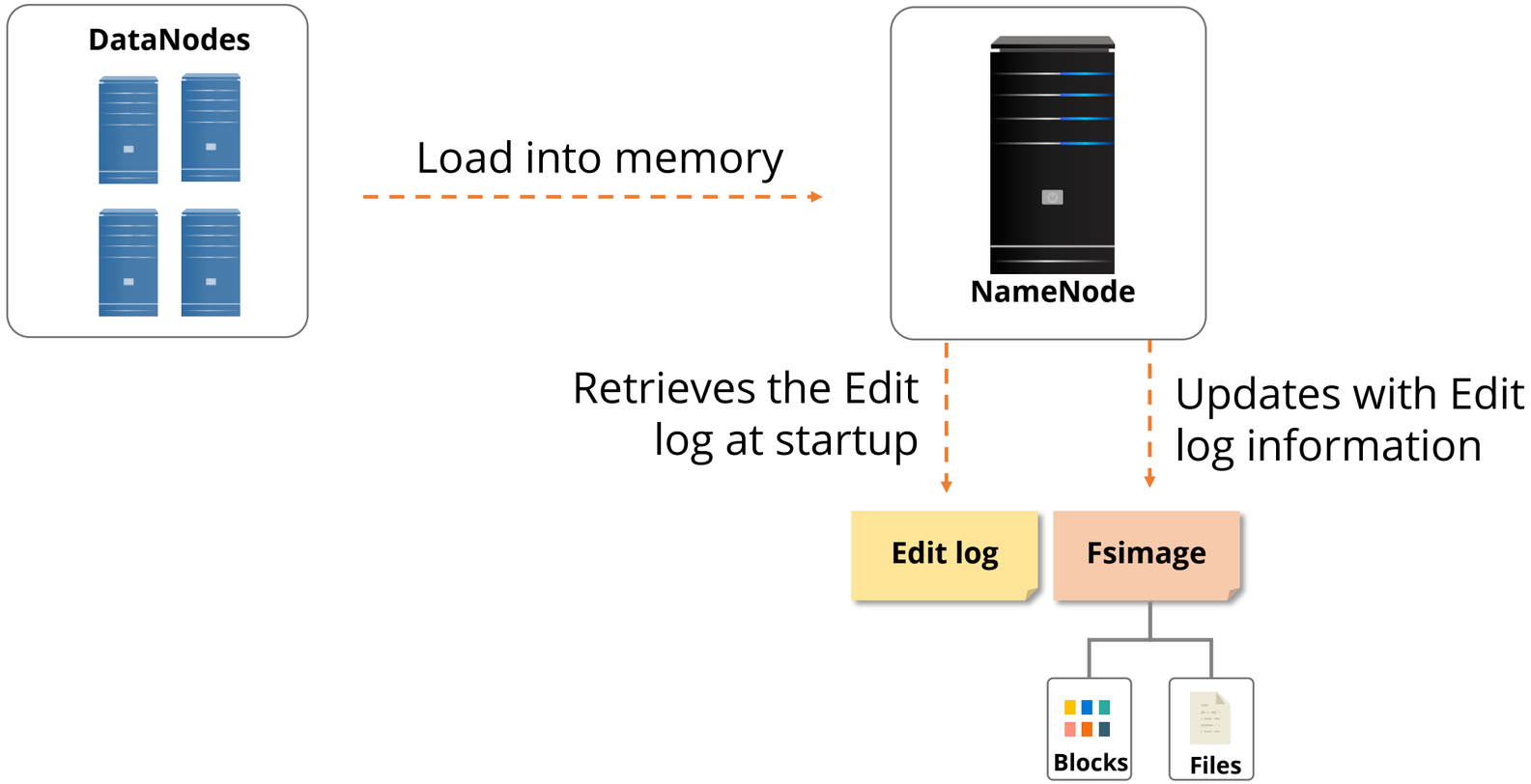
- Allows user data to be stored in files
- Follows a hierarchical file system with directories and files
- Supports operations such as create, remove, move, and rename



NameNode Operation

The NameNode maintains two persistent files:

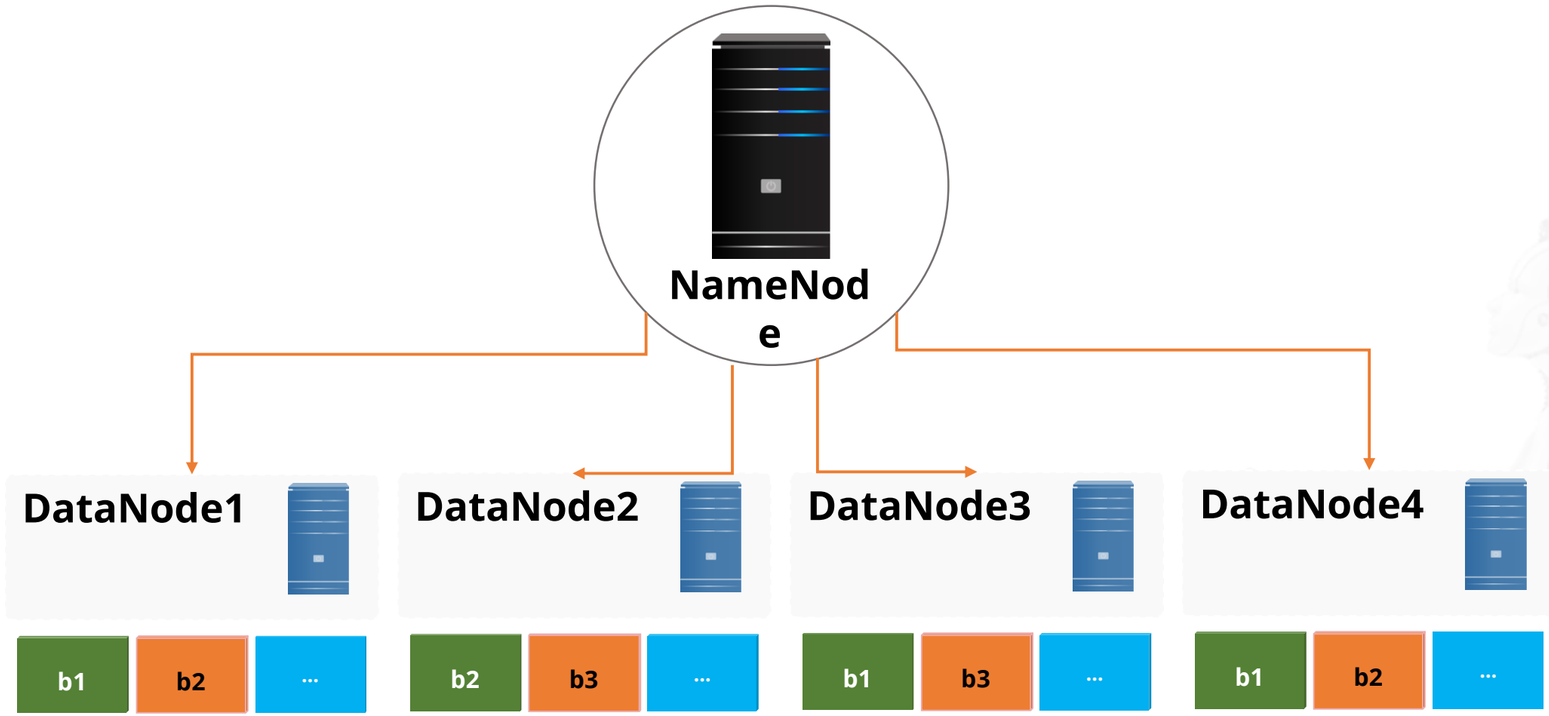
- A transaction log called an Edit Log and
- A namespace image called an FsImage



Data Block Split

Each file is split into one or more blocks which are stored and replicated in DataNodes.

A file split into blocks

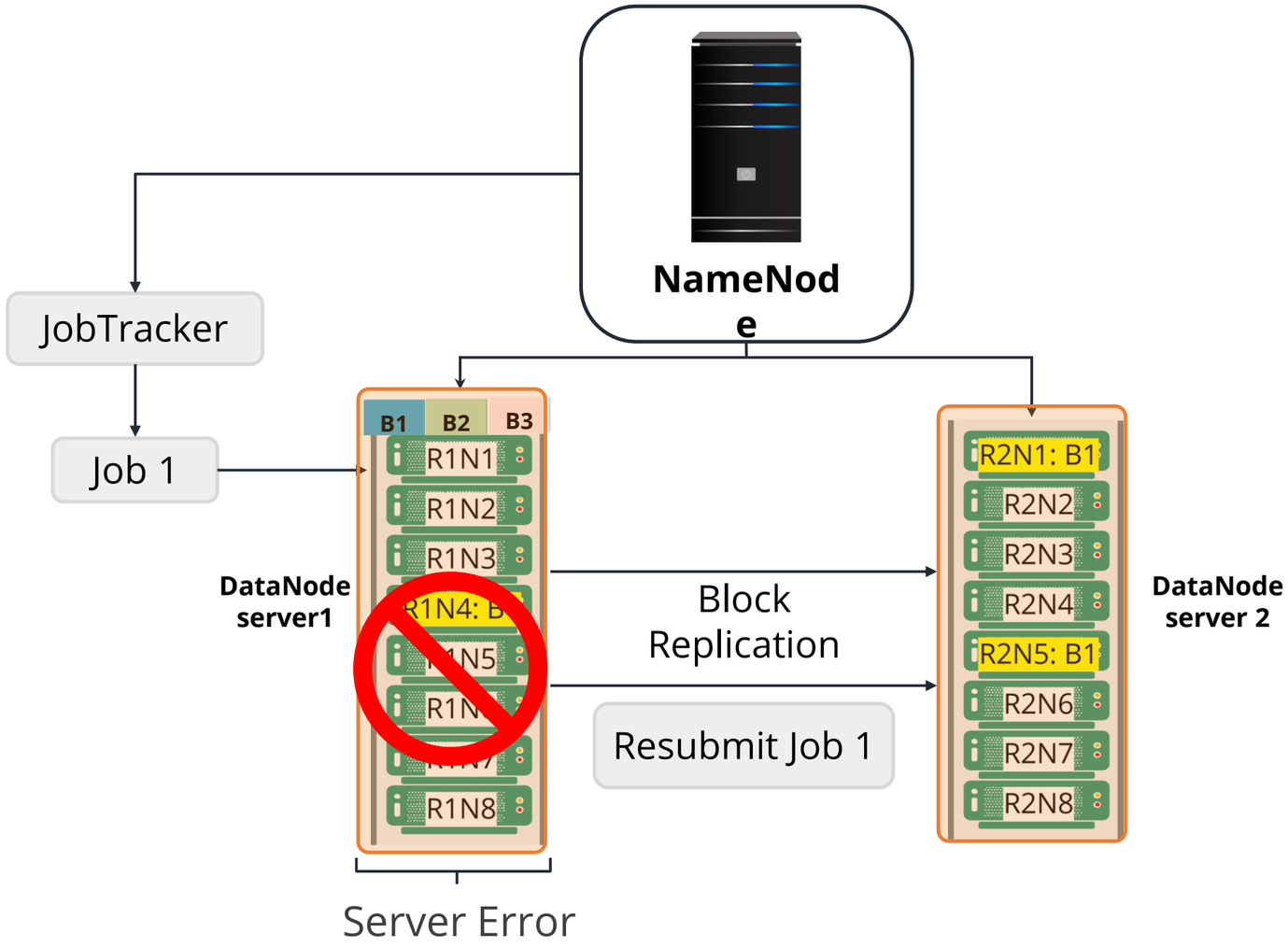


DataNodes managing blocks

The data block approach provides simplified replication, fault-tolerance, and reliability.

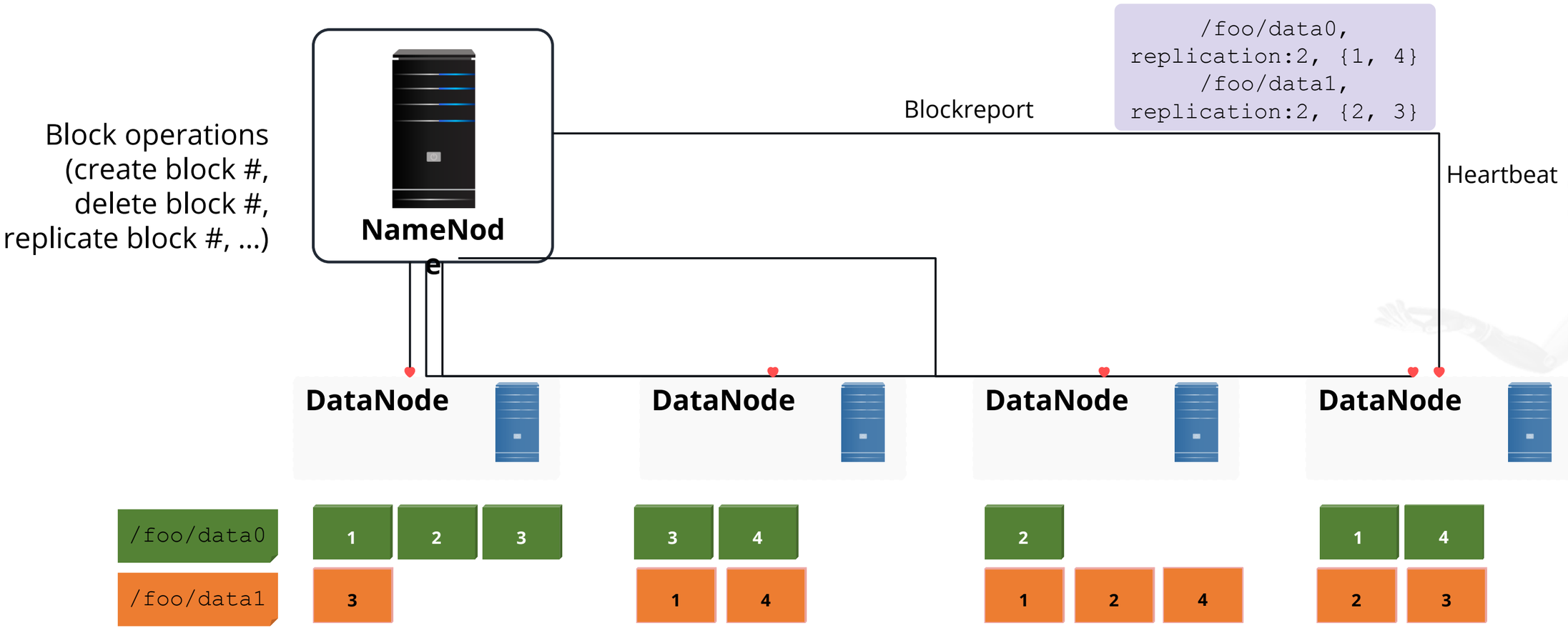
Block Replication Architecture

HDFS represents the unstructured data in the form of data blocks.
It performs block replication on multiple DataNodes.



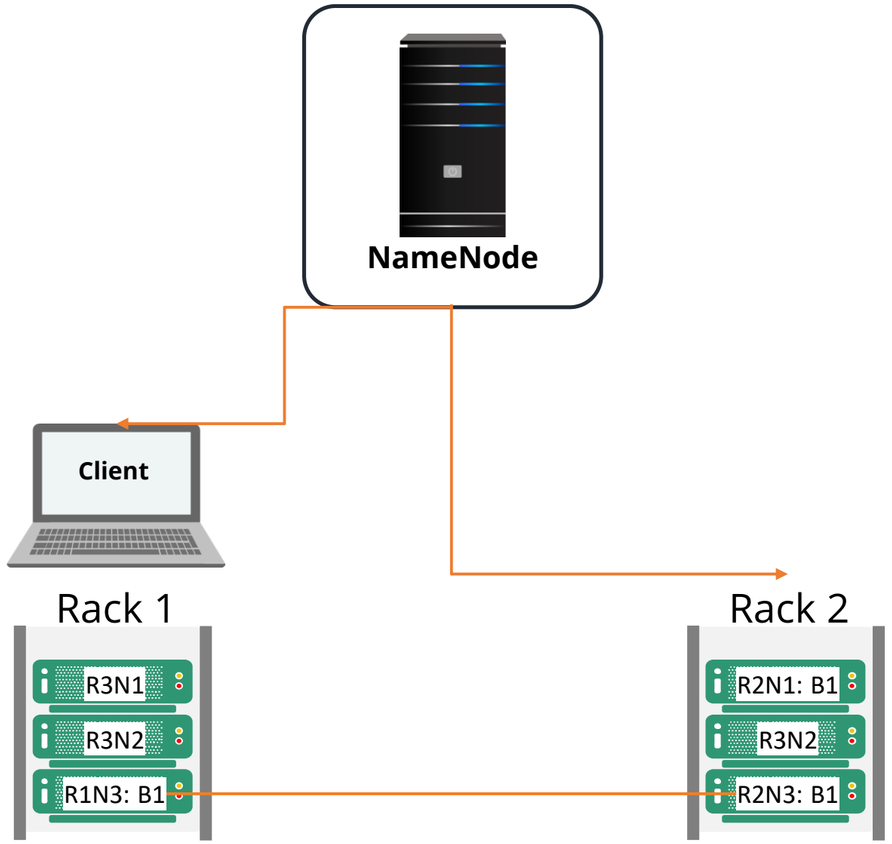
Replication Method

Each file is split into a sequence of blocks.
Except for the last one, all blocks in the file are of the same size.



Data Replication Topology

One of the suggested replication topology is as follows:



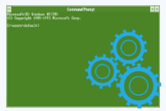
HDFS Access

HDFS provides the following access mechanisms:

Java API for applications

Python access and C language wrapper for non-Java applications

Web GUI utilized through an HTTP browser



FS shell for executing commands on HDFS

HDFS Command Line

Following are a few basic command lines of HDFS:

Copy file simplilearn.txt from local disk to the user's directory in HDFS –This will copy the file to /user/username/simplilearn.txt	$ hdfs dfs -put simplilearn.txt simplilearn.txt
Display a list of the contents of the directory path provided by the user, showing the names, permissions, owner, size, and modification date for each entry	$hdfs dfs –ls /user/simpl/test
Create a directory called test under the user's home directory	$hdfs dfs –mkdir /user/simpl/test
Delete the directory testing and all its contents	hdfs dfs -rm -r testing

Assisted Practice

HDFS Command Line

Duration: 15 mins

Problem Statement: In this demonstration, you will explore few basic command lines of 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**.

Unassisted Practice

HDFS Command Line

Duration: 15 mins

Problem Statement: Using command lines of HDFS, perform the below tasks:

- Create a directory named “Simplilearn“ in HDFS
- Transfer a sample text file from your local filesystem to HDFS directory
- List the files in HDFS directory
- Change the permissions on the file to read, write, and execute
- Remove the text file from HDFS directory

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**.

Unassisted Practice

Steps to Perform

- Create a directory named “Simplilearn"
  hdfs dfs -mkdir Simplilearn

- Transfer a sample text file from your local filesystem to HDFS directory
  hdfs dfs -put /home/simplilearn_learner/test.txt Simplilearn

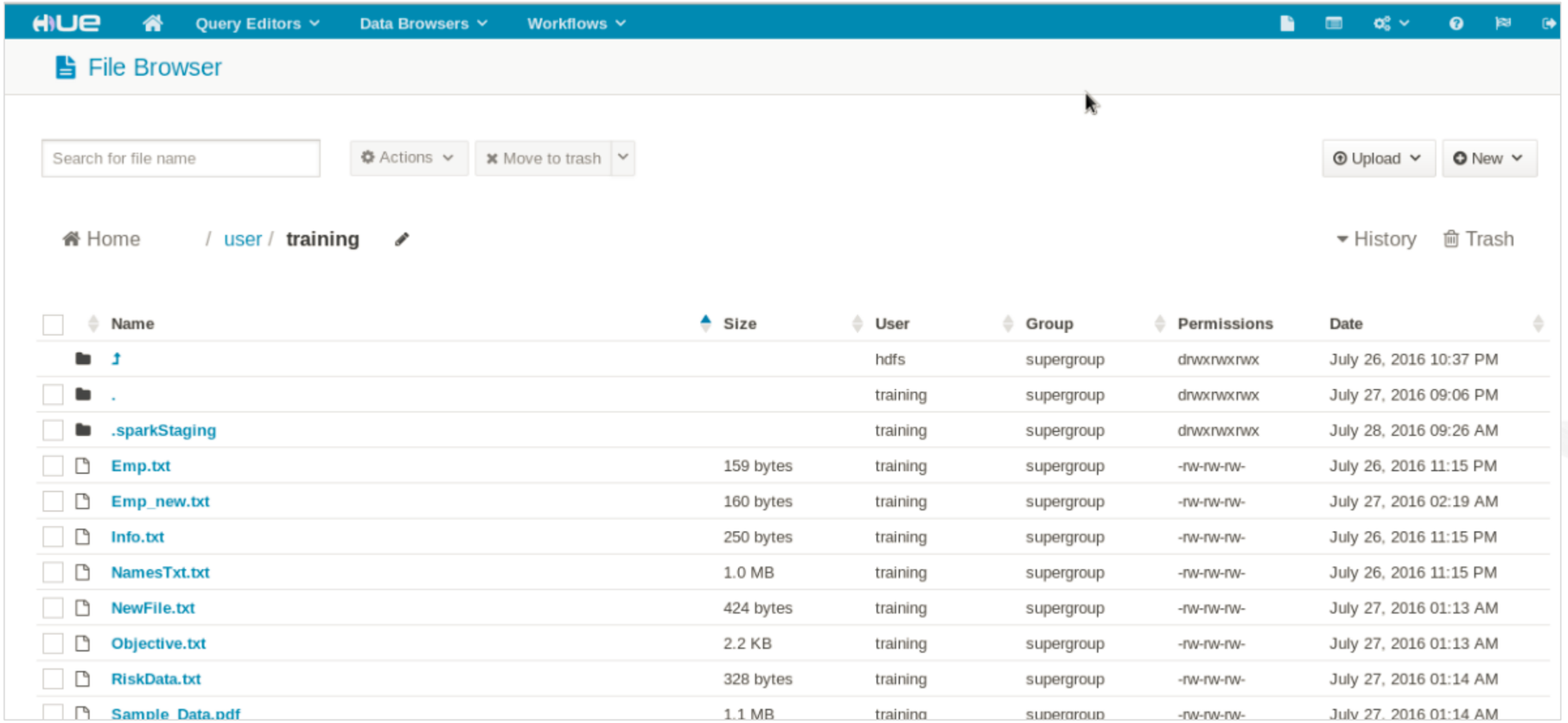
- List the files in HDFS directory
  hdfs dfs -ls Simplilearn

- Change the permissions on the file to read, write and execute
  hdfs dfs -chmod 700 Simplilearn/test.txt

- Remove the text file from HDFS directory
  hdfs dfs -rm -r Simplilearn/test.txt

Hue File Browser

The file browser in Hue lets you view and manage your HDFS directories and files.



Name	Size	User	Group	Permissions	Date
..		hdfs	supergroup	drwxrwxrwx	July 26, 2016 10:37 PM
.		training	supergroup	drwxrwxrwx	July 27, 2016 09:06 PM
.sparkStaging		training	supergroup	drwxrwxrwx	July 28, 2016 09:26 AM
Emp.txt	159 bytes	training	supergroup	-rw-rw-rw-	July 26, 2016 11:15 PM
Emp_new.txt	160 bytes	training	supergroup	-rw-rw-rw-	July 27, 2016 02:19 AM
Info.txt	250 bytes	training	supergroup	-rw-rw-rw-	July 26, 2016 11:15 PM
NamesTxt.txt	1.0 MB	training	supergroup	-rw-rw-rw-	July 26, 2016 11:15 PM
NewFile.txt	424 bytes	training	supergroup	-rw-rw-rw-	July 27, 2016 01:13 AM
Objective.txt	2.2 KB	training	supergroup	-rw-rw-rw-	July 27, 2016 01:13 AM
RiskData.txt	328 bytes	training	supergroup	-rw-rw-rw-	July 27, 2016 01:14 AM
Sample_Data.pdf	1.1 MB	training	supergroup	-rw-rw-rw-	July 27, 2016 01:14 AM

Additionally, you can create, move, rename, modify, upload, download, and delete directories and files.

Assisted Practice

Hue File Browser

Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to access HDFS using Hue. You will also learn how to view and manage your HDFS directories and files using Hue.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

YARN: Introduction

What Is YARN?

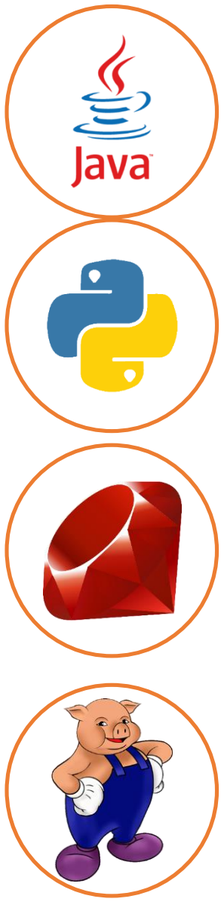
YARN= Yet Another Resource Negotiator

- YARN is a resource manager
- Created by separating the processing engine and the management function of MapReduce
- Monitors and manages workloads, maintains a multi-tenant environment, manages the high availability features of Hadoop, and implements security controls

Need for YARN

Before 2012

Users could write MapReduce programs using scripting languages



HADOOP 1.0

MapReduce
(cluster resource management and data processing)

HDFS
(redundant, reliable storage)

Since 2012

Users could work on multiple processing models in addition to MapReduce

HADOOP

MapReduce
(data processing)

Others
(data processing)

YARN
(cluster resource management)

HDFS
(redundant, reliable storage)

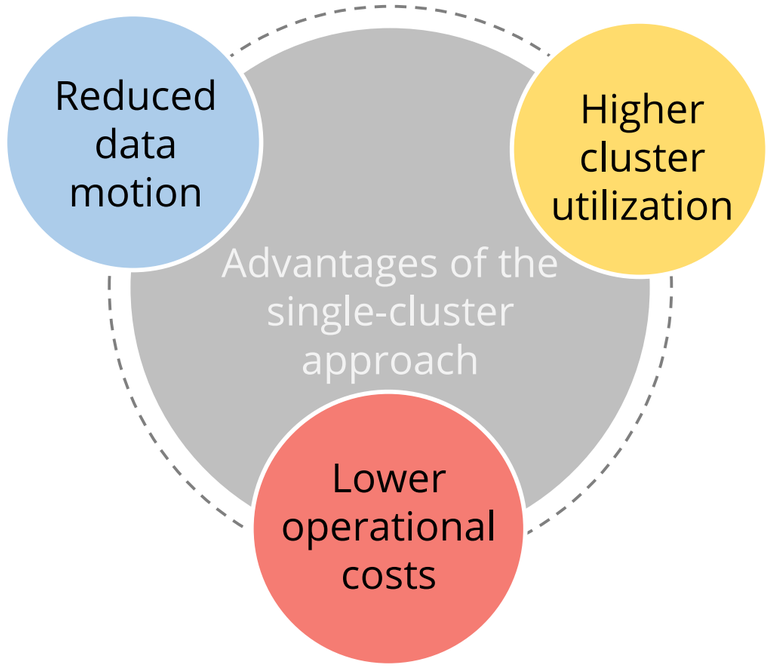
YARN: Use Case

- Yahoo was the first company to embrace Hadoop, and it is a trendsetter within the Hadoop ecosystem. In late 2012, it struggled to handle iterative and stream processing of data on Hadoop infrastructure due to MapReduce limitations.
- After implementing YARN in the first quarter of 2013, Yahoo has installed more than 30,000 production nodes on
  - Spark for iterative processing
  - Storm for stream processing
  - Hadoop for batch processing
- Such a solution was possible only after YARN was introduced and multiple processing frameworks were implemented.

YARN: Advantages

The single-cluster approach provides a number of advantages including:

There's no need to move data between Hadoop YARN and systems running on different clusters of computers

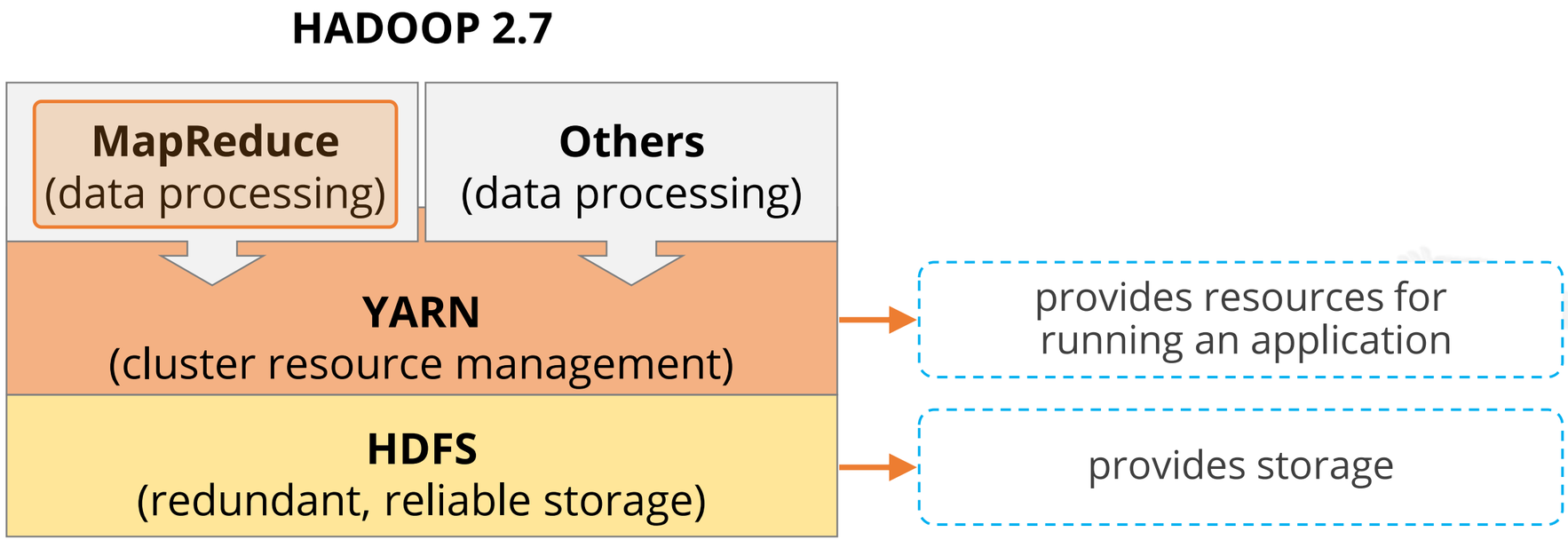


Resources unutilized by a framework can be consumed by another

Only one "do-it-all" cluster needs to be managed

YARN Infrastructure

The YARN Infrastructure is responsible for providing computational resources for application executions.

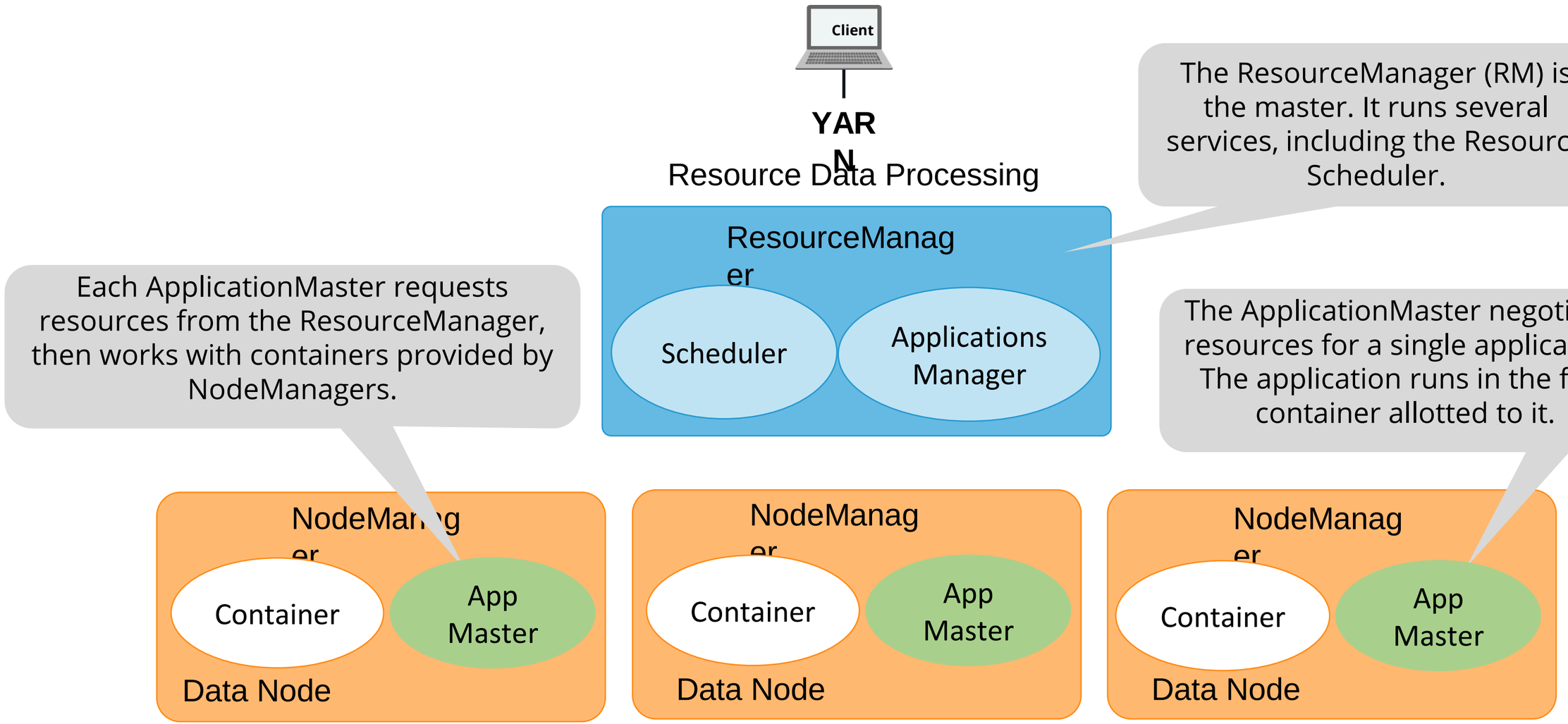


DATA AND
ARTIFICIAL INTELLIGENCE

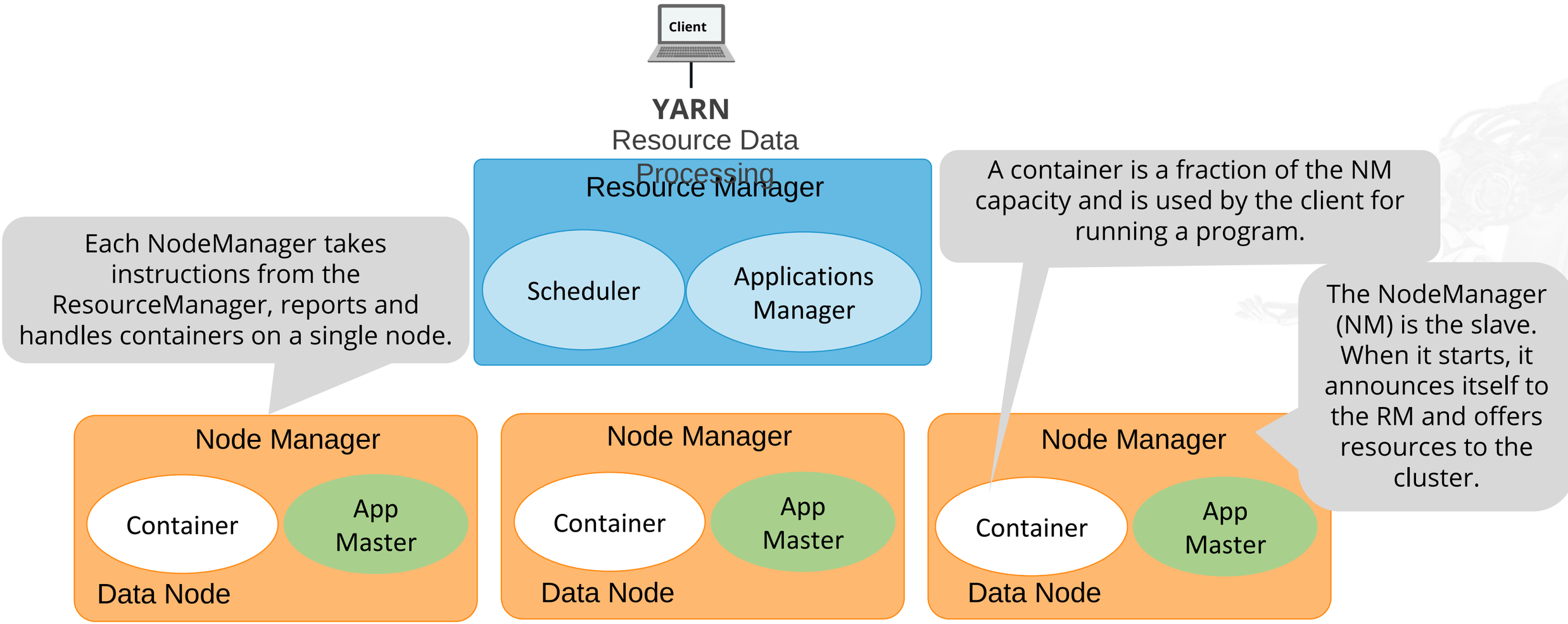
YARN and Its Architecture

Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.

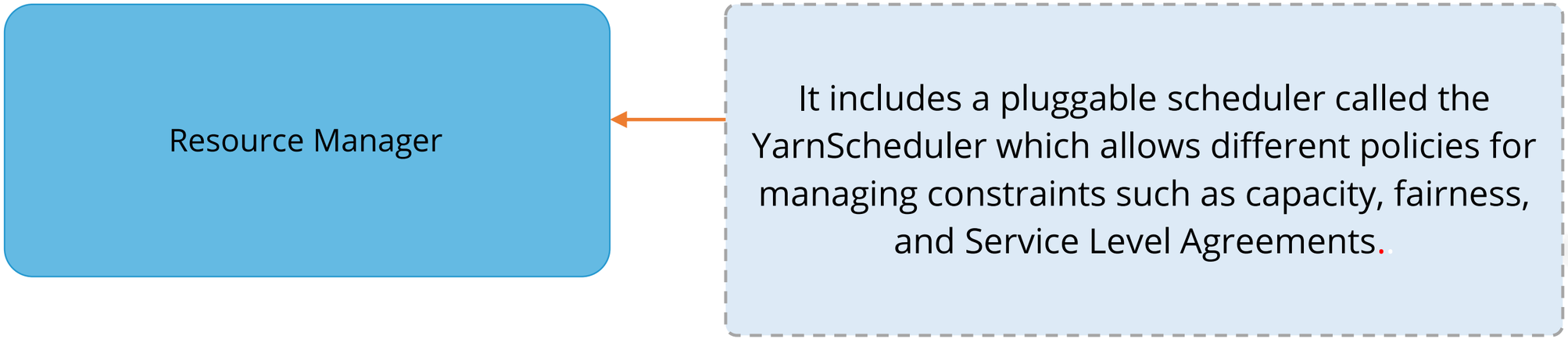


Elements of YARN Architecture



YARN Architecture Element: ResourceManager

The RM mediates the available resources in the cluster among competing applications for maximum cluster utilization.



ResourceManager Component: Scheduler

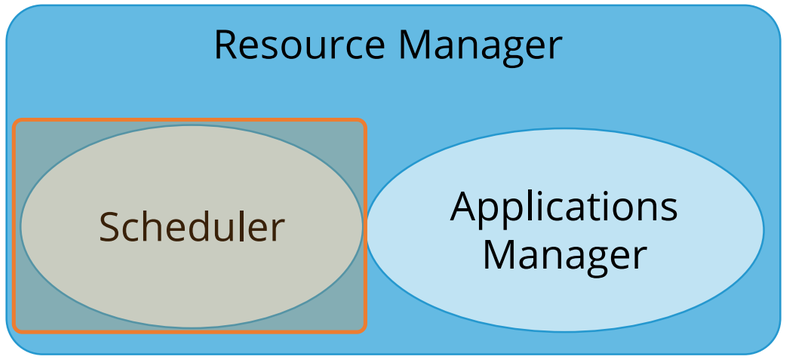
The Scheduler is responsible for allocating resources to various running applications.

The Scheduler does not monitor or track the status of the application.

The Scheduler does not restart failed tasks.

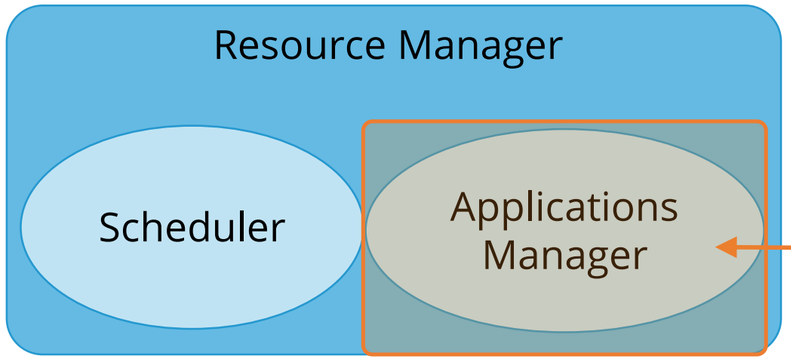
The Scheduler has a policy plug-in to partition cluster resources among various applications.

Most commonly used schedulers are FIFO Scheduler, Capacity Scheduler, and Fair Scheduler.



ResourceManager Component: ApplicationsManager

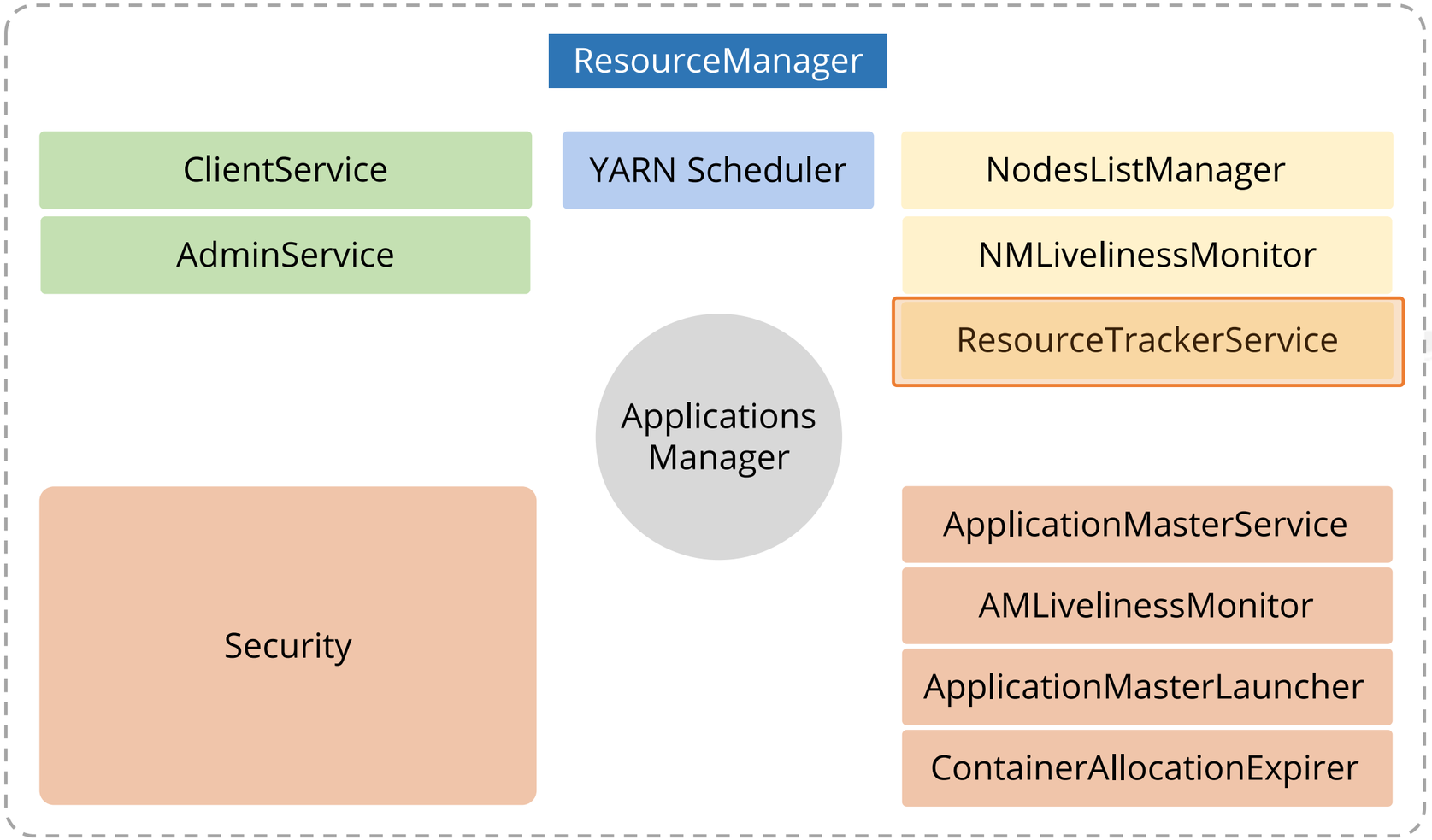
The ApplicationsManager is an interface which maintains a list of applications that have been submitted, currently running, or completed.



The ApplicationsManager accepts job submissions, negotiates the first container for executing the application, and restarts the ApplicationMaster container on failure.

How a ResourceManager Operates

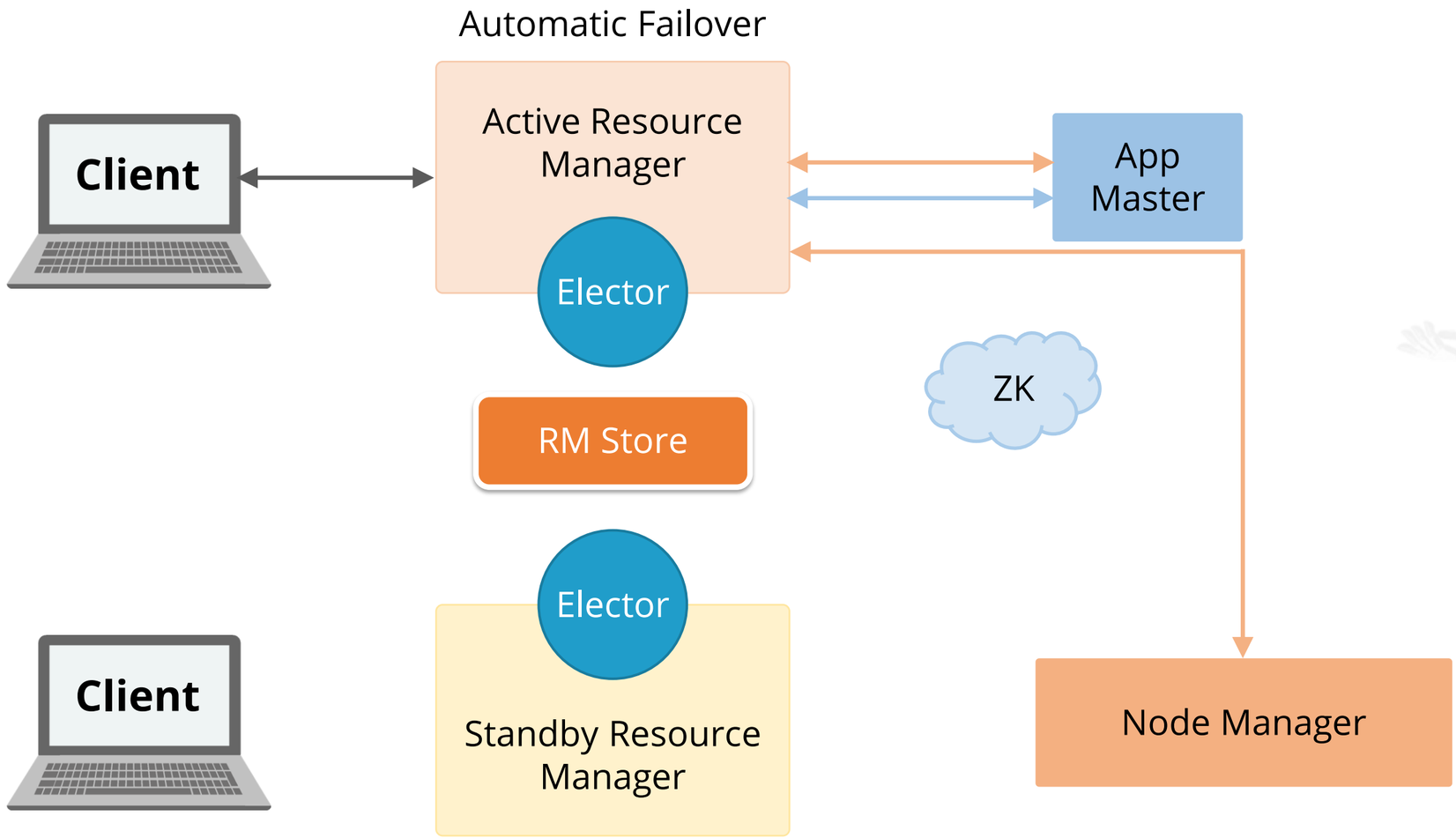
The figure shown here displays all the internal components of the ResourceManager.



ResourceManager in High Availability Mode

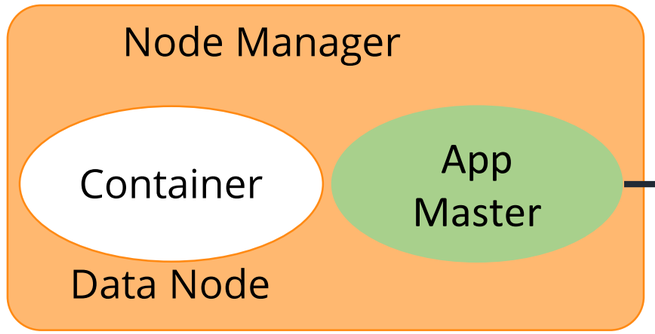
Before Hadoop 2.4, the ResourceManager was the single point of failure in a YARN cluster.

The High Availability, or HA feature is an Active/Standby ResourceManager pair to remove this single point of failure.



YARN Architecture Element: ApplicationMaster

The ApplicationMaster in YARN is a framework-specific library which negotiates resources from the RM and works with the NodeManager or Managers to execute and monitor containers and their resource consumption.



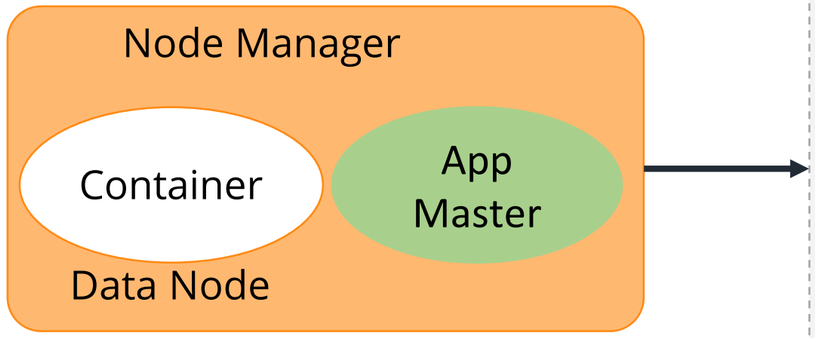
The ApplicationMaster:

- manages the application lifecycle
- makes dynamic adjustments to resource consumption
- manages execution flow
- manages faults
- provides status and metrics to the RM
- interacts with NodeManager and RM using extensible communication protocols
- is not run as a trusted service

While every application has its own instance of an AppMaster, it is possible to implement an AppMaster for a set of applications as well.

YARN Architecture Element: NodeManager

When a container is leased to an application, the NodeManager sets up the container's environment including the resource constraints specified in the lease and any dependencies.

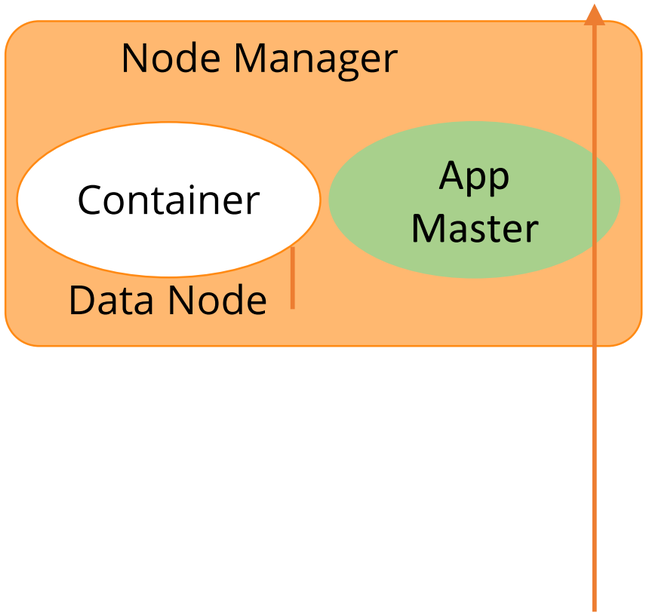


The NodeManager runs on each node and manages the following:

- Container lifecycle management
- Container dependencies
- Container leases
- Node and container resource usage
- Node health
- Log management
- Reporting node and container status to the RM

YARN Container

A YARN container is a collection of a specific set of resources to use in certain numbers on a specific node.

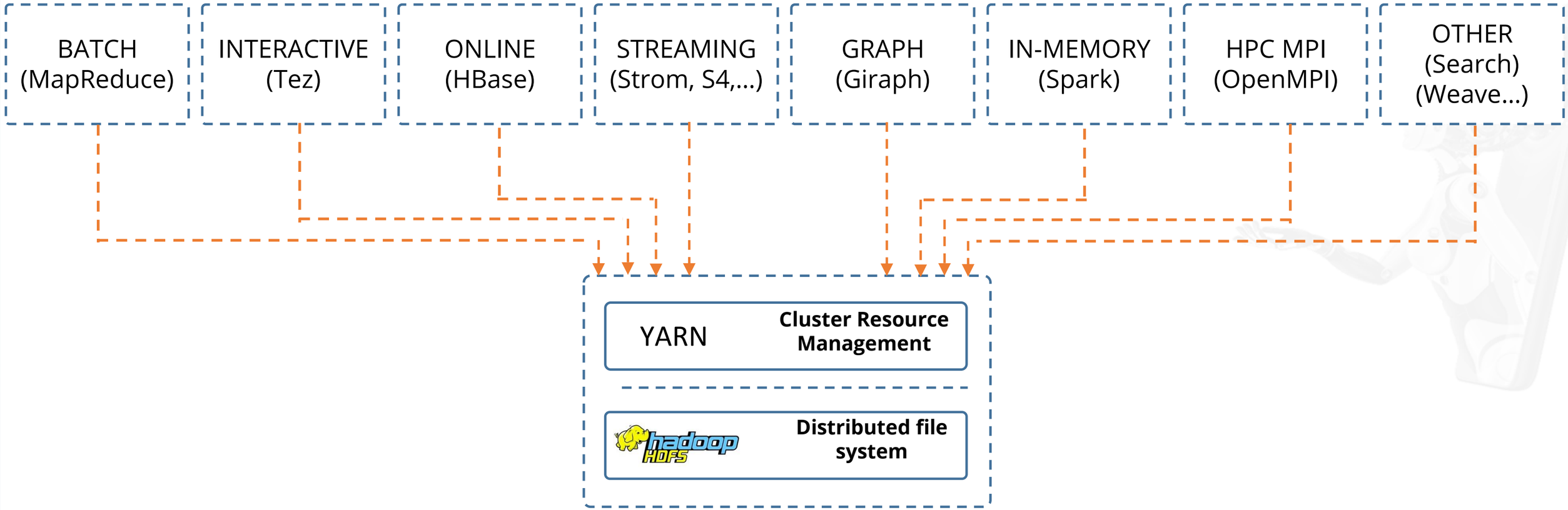


To launch the container, the ApplicationMaster must provide a Container Launch Context (CLC) that includes the following information:

- Environment variables
- Dependencies, that is, local resources such as data files or shared objects needed prior to launch
- Security tokens
- The command necessary to create the process the application wants to launch

Applications on YARN

There can be many different workloads running on a Hadoop YARN cluster.



How YARN Runs an Application

There are five steps involved in running an application by YARN:

01 The client submits an application to the ResourceManager

02 The ResourceManager allocates a container

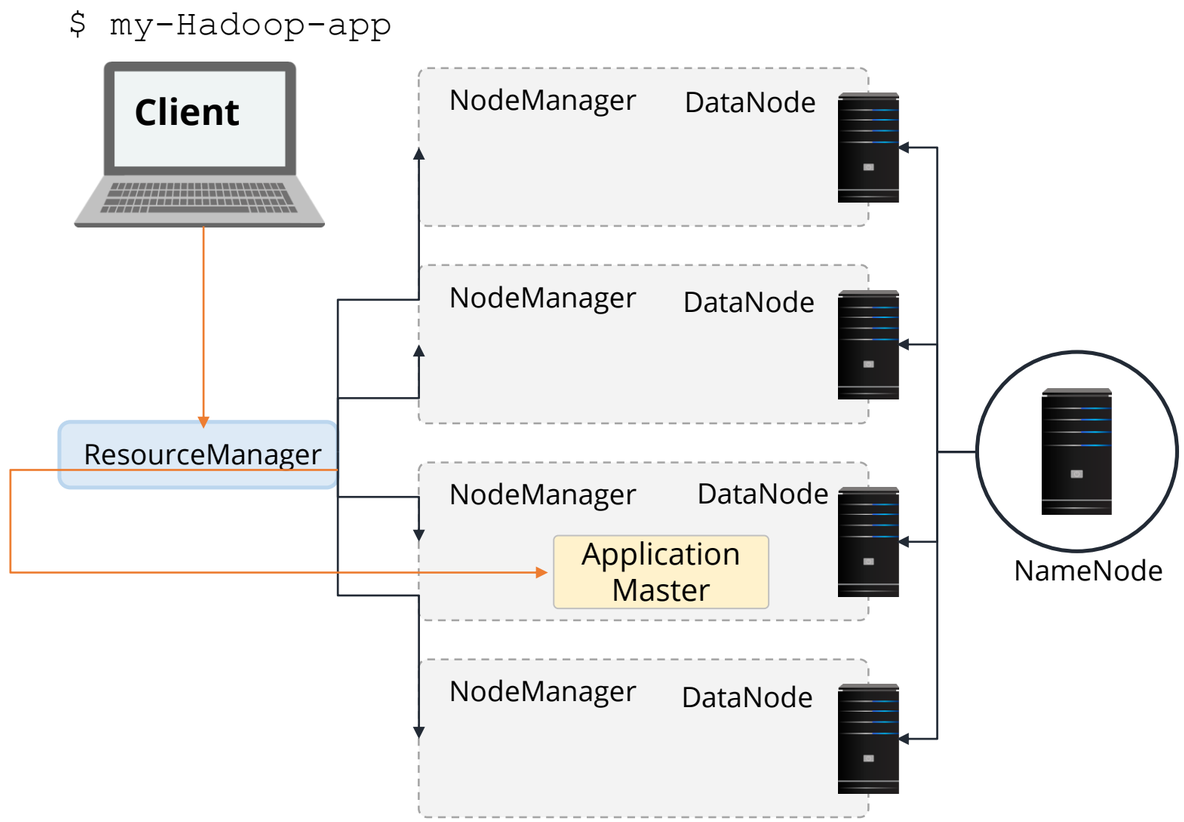
03 The ApplicationMaster contacts the related NodeManager

04 The NodeManager launches the container

05 The container executes the ApplicationMaster

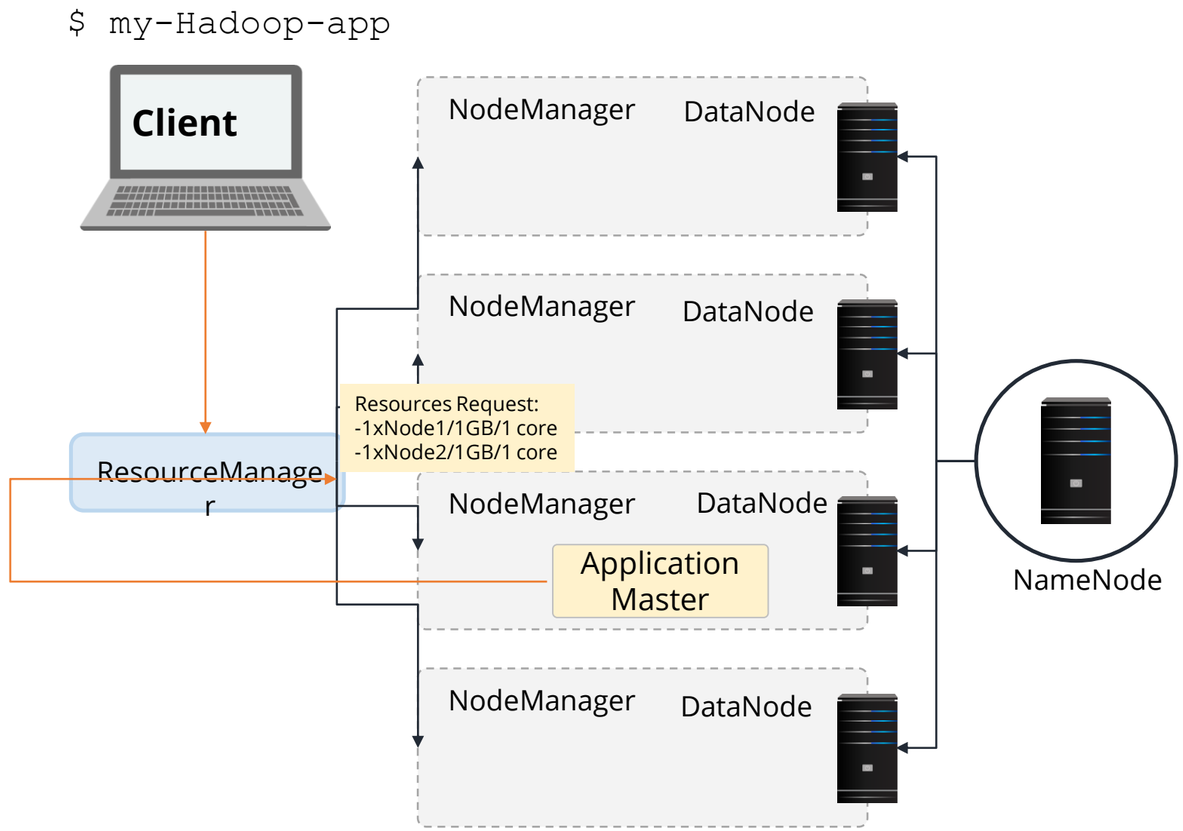
Step1: Application Submitted to ResourceManager

Users submit applications to the ResourceManager by typing the Hadoop jar command.



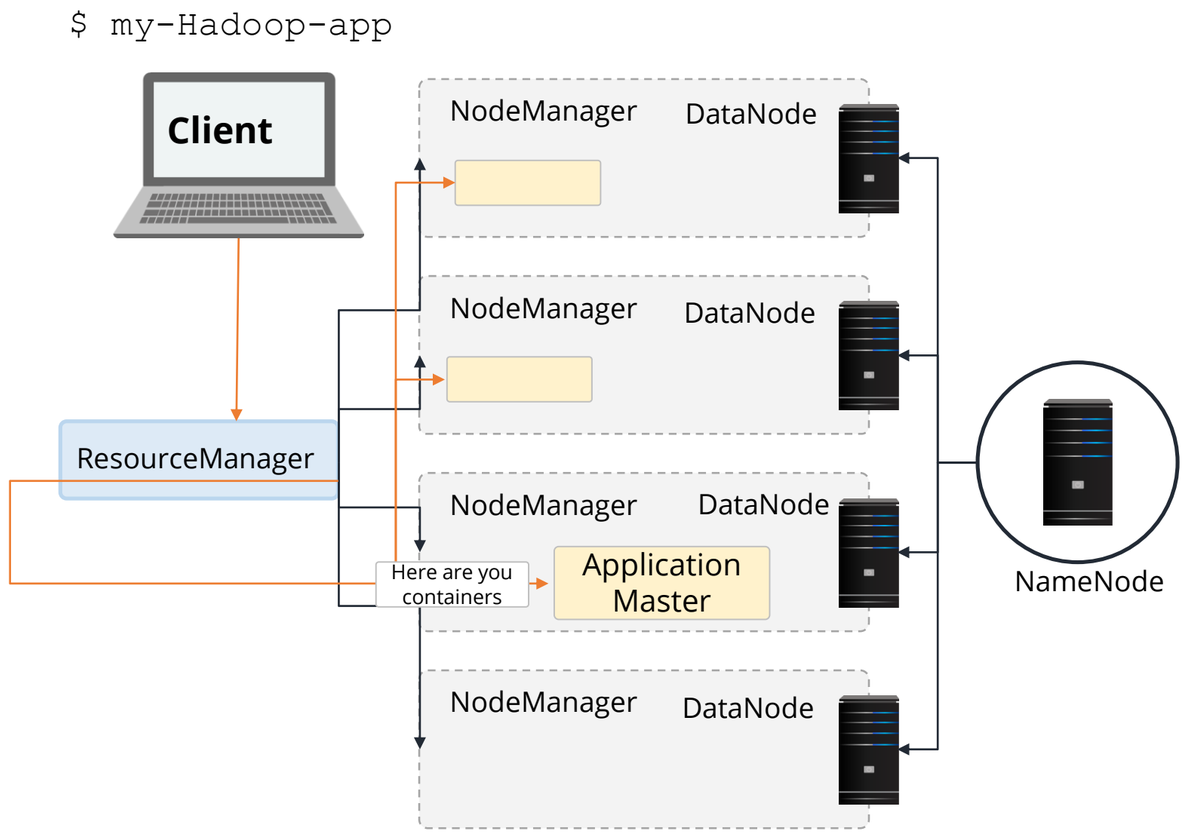
Step2: ResourceManager Allocates a Container

When the ResourceManager accepts a new application submission, one of the first decisions the Scheduler makes is selecting a container.



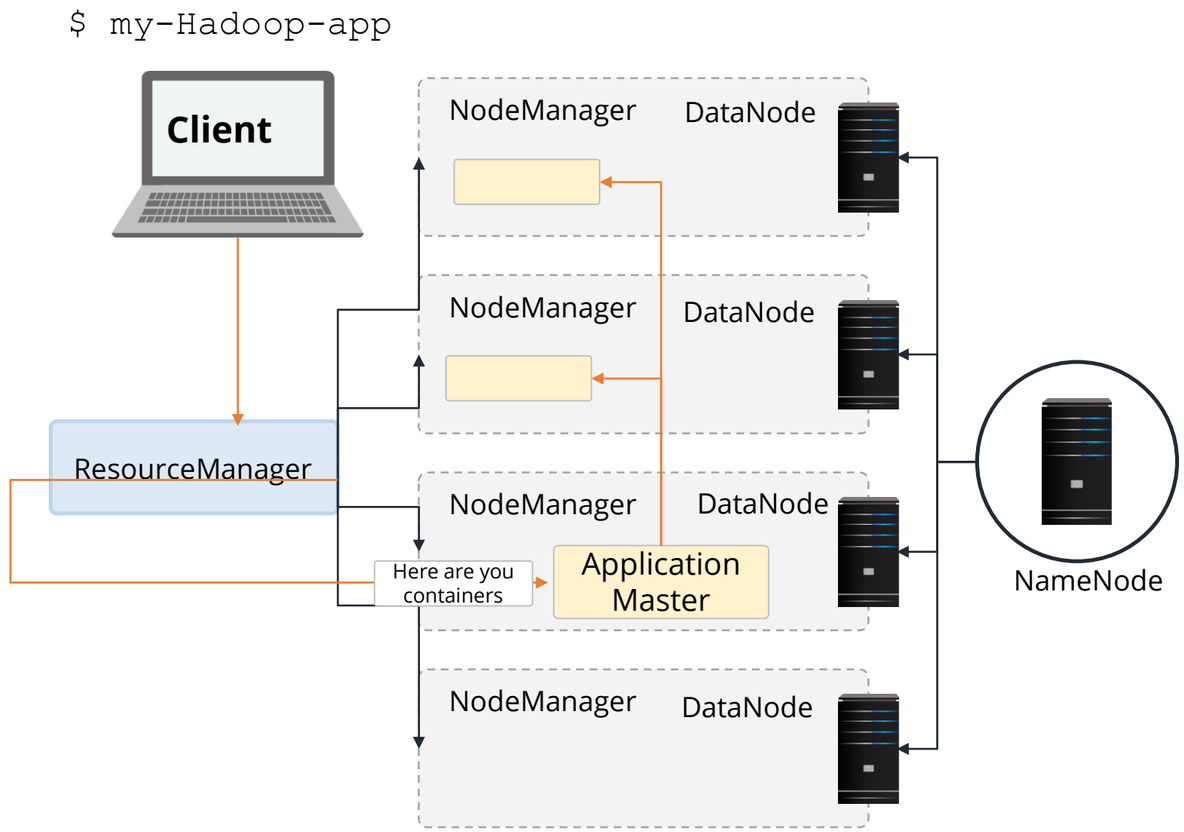
Step3: ApplicationMaster Contacts NodeManager

After a container is allocated, the ApplicationMaster asks the NodeManager managing the host on which the container was allocated to use these resources to launch an application-specific task.



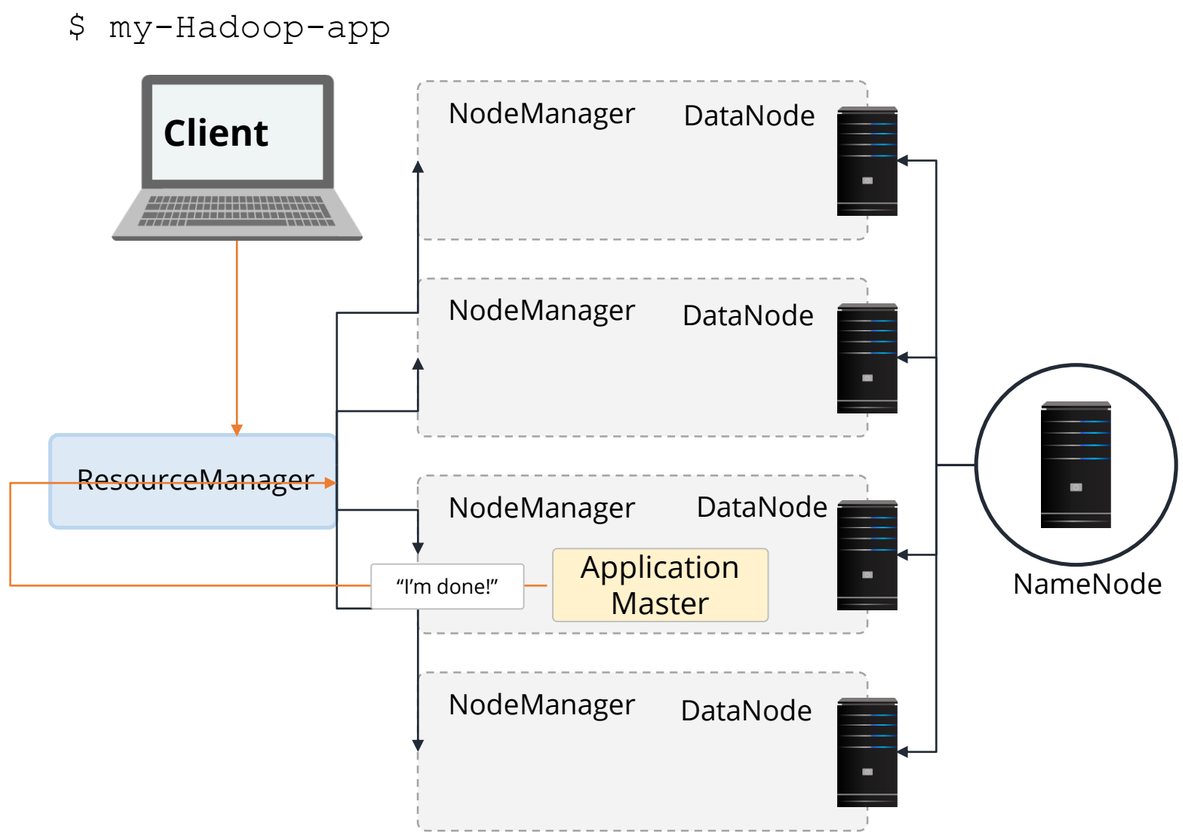
Step4: ResourceManager Launches a Container

The NodeManager does not monitor tasks; it only monitors the resource usage in the containers.



Step5: Container Executes the ApplicationMaster

After the application is complete, the ApplicationMaster shuts itself and releases its own container.



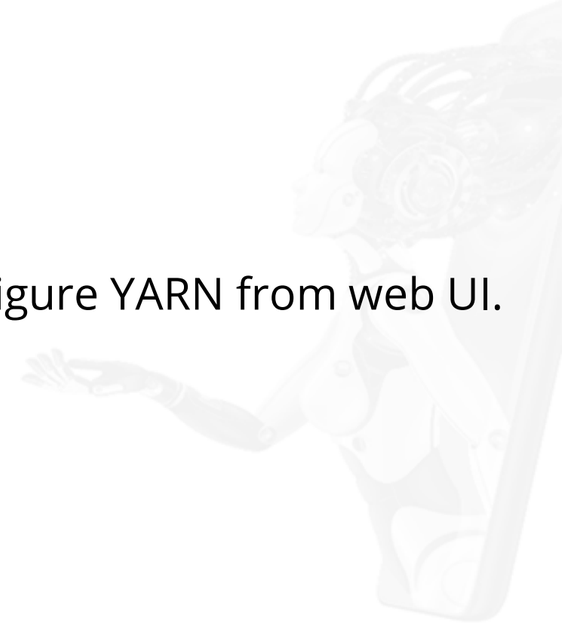
Tools for YARN Developers

Hadoop includes three tools for YARN developers:

YARN Web UI

YARN web UI runs on 8088 port, by default.

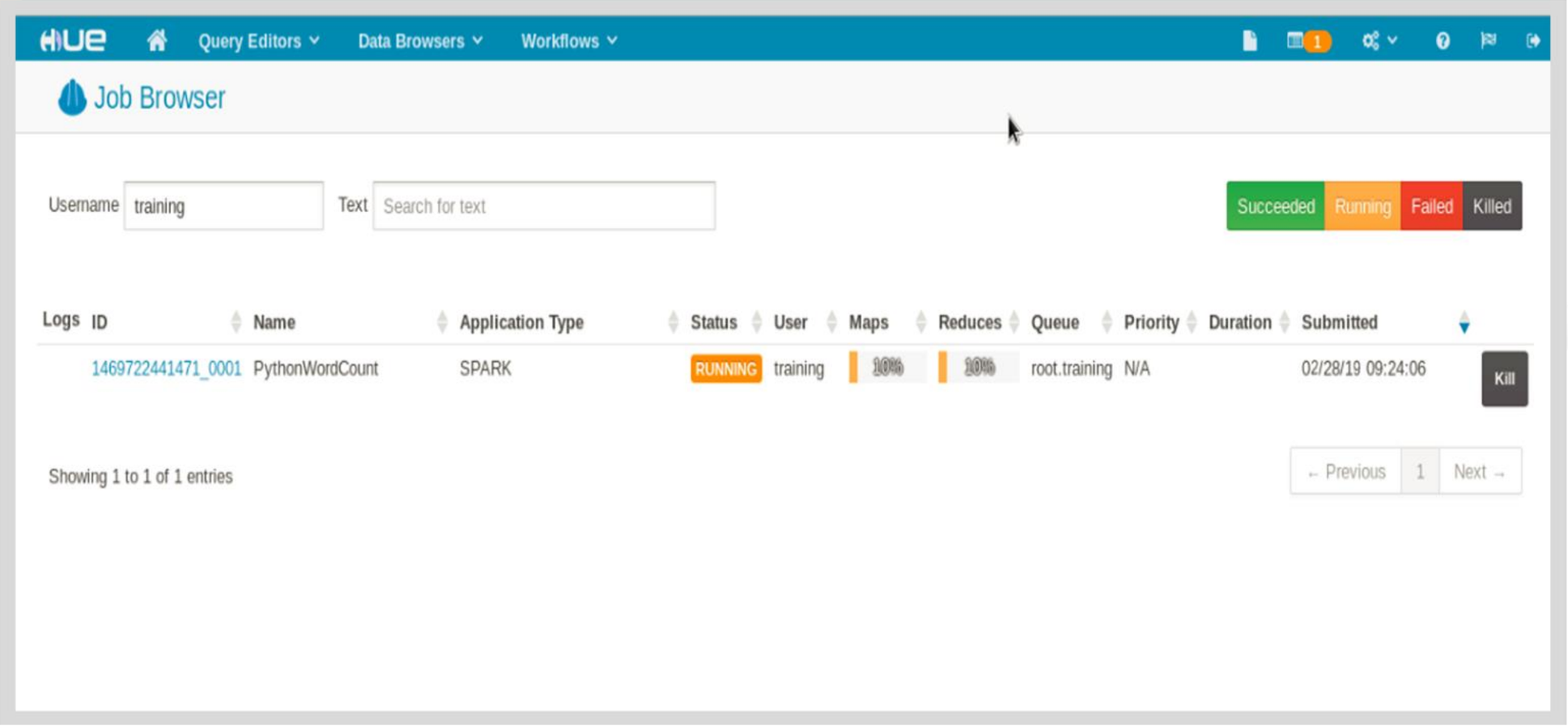
It also provides a better view than Hue. However, you can’t control or configure YARN from web UI.



Hue Job Browser

Hue Job Browser

The Hue Job Browser allows you to monitor the status of a job, kill a running job, and view logs.



YARN Command Line

YARN Command Line

Most of the YARN commands are for the administrator rather than the developer.

Few useful commands for developer:

- yarn –help
list all command of yarn

- yarn –version
print the version

- yarn logs -applicationId <app-id>
views logs of specified application ID

Assisted Practice

YARN

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to use YARN.

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 how Hadoop Distributed File System (HDFS) stores data across a cluster
- Illustrate the HDFS architecture and its components
- Demonstrate the use of HDFS Command Line Interface (CLI)
- Illustrate the YARN architecture and its components
- Demonstrate how to use Hue, YARN Web UI, and the YARN command to monitor the cluster

Quiz

Knowledge Check

1

Which of the following statements best describes how a large (100 GB) file is stored in HDFS?

a. The file is replicated three times by default. Each copy of the file is stored on a separate DataNode.

b. The master copy of the file is stored on a single DataNode. The replica copies are divided into fixed-size blocks which are stored on multiple DataNodes.

c. The file is divided into fixed-size blocks which are stored on multiple DataNodes. Each block is replicated three times by default. Multiple blocks from the same file might reside on the same DataNode.

d. The file is divided into fixed-size blocks which are stored on multiple DataNodes. Each block is replicated three times by default. HDFS guarantees that different blocks from the same file are never on the same DataNode.

Knowledge Check

1

Which of the following statements best describes how a large (100 GB) file is stored in HDFS?

a. The file is replicated three times by default. Each copy of the file is stored on a separate DataNode.

b. The master copy of the file is stored on a single DataNode. The replica copies are divided into fixed-size blocks which are stored on multiple DataNodes.

c. The file is divided into fixed-size blocks which are stored on multiple DataNodes. Each block is replicated three times by default. Multiple blocks from the same file might reside on the same DataNode.

d. The file is divided into fixed-size blocks which are stored on multiple DataNodes. Each block is replicated three times by default. HDFS guarantees that different blocks from the same file are never on the same DataNode.

The correct answer is **d.**

The file is divided into fixed-size blocks which are stored on multiple DataNodes. Each block is replicated three times by default. HDFS guarantees that different blocks from the same file are never on the same DataNode.

Knowledge Check

2

How many blocks are required to store a file of size 514 MB in HDFS using default block size configuration?

a. 4

b. 5

c. 6

d. 7

Knowledge Check

2

How many blocks are required to store a file of size 514 MB in HDFS using default block size configuration?

a. 4

b. 5

c. 6

d. 7

The correct answer is **b.**

The default block size in Hadoop 3.x is 128 MB. So, a file of size 514 MB will be divided into 5 blocks where the first four blocks will be of 128 MB and the last block will be of 2 MB.

Knowledge Check

3

Which of the following statements is not true about HDFS?

a. We cannot modify files already present in HDFS

b. Multiple clients can write into an HDFS file concurrently

c. Both A and B

d. None of the above

Knowledge Check

3

Which of the following statements is not true about HDFS?

a. We cannot modify files already present in HDFS

b. Multiple clients can write into an HDFS file concurrently

c. Both A and B

d. None of the above

The correct answer is **b.**

HDFS follows Write Once Read Many model.
So, multiple clients can't write into an HDFS file concurrently.

Knowledge Check

4

What is the model of a ZooKeeper cluster?

a. Master/Slave

b. Leader and Follower

c. Peer to Peer

d. Primary and Secondary

Knowledge Check

4

What is the model of a ZooKeeper cluster?

a. Master/Slave

b. Leader and Follower

c. Peer to Peer

d. Primary and Secondary

The correct answer is **b.**

Apache ZooKeeper deploys a Leader and Follower model to provide coordination service for distributed applications.

Knowledge Check

5

Which of the following scheduling policies can be implemented in Yarn?

a. FIFO scheduler

b. Capacity scheduler

c. Fair scheduler

d. All of the above

Knowledge Check

5

Which of the following scheduling policies can be implemented in Yarn?

a. FIFO scheduler

b. Capacity scheduler

c. Fair scheduler

d. All of the above

The correct answer is **d.**

YARN provides three scheduling options - FIFO scheduler, Capacity scheduler, and Fair scheduler for scheduling resources to user applications.

Knowledge Check

6

_____ negotiates resources from the Resource Manager.

a. Node Manager

b. Resource Manager

c. Application Master

d. None of the above

Knowledge Check

6

_____ negotiates resources from the Resource Manager.

a. Node Manager

b. Resource Manager

c. Application Master

d. None of the above

The correct answer is **c.**

Each Application Master is responsible for negotiating resources from the Resource Manager.

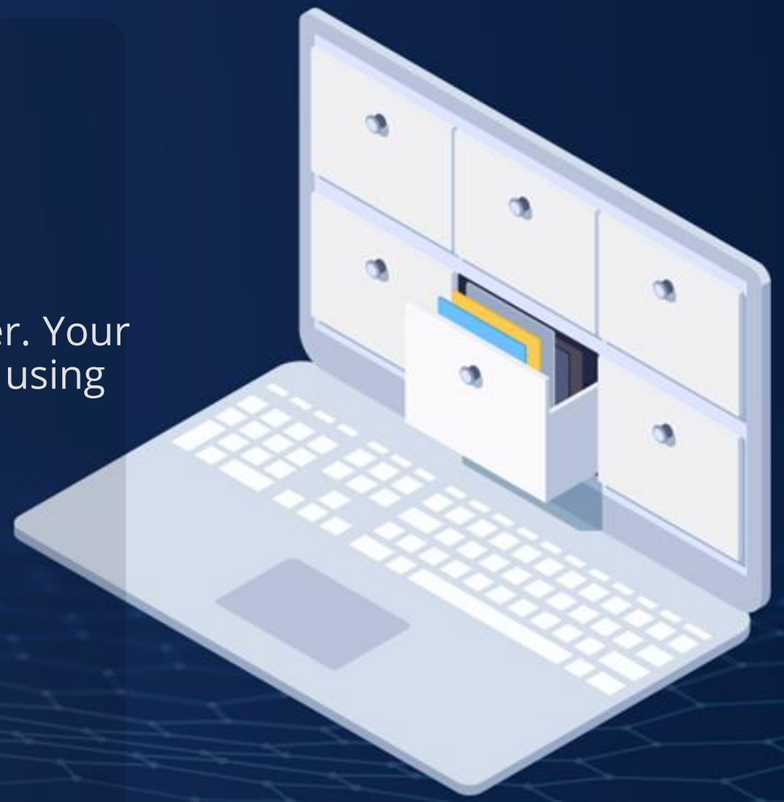
Lesson-End Project

Problem Statement:

PV Consulting is one of the top consulting firms for big data projects.
They mostly help big and small companies to analyze their data.

For Spark & Hadoop MR application they started using YARN as a resource manager. Your task is to provide the following information for any job which is submitted to YARN using YARN console and YARN Cluster UI:

1. Who has submitted the job?
2. To which YARN queue is a job submitted?
3. How much time did it take to finish the job?
4. List of all running jobs on YARN
5. How to kill a job using YARN?
6. Check logs using YARN



DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

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:

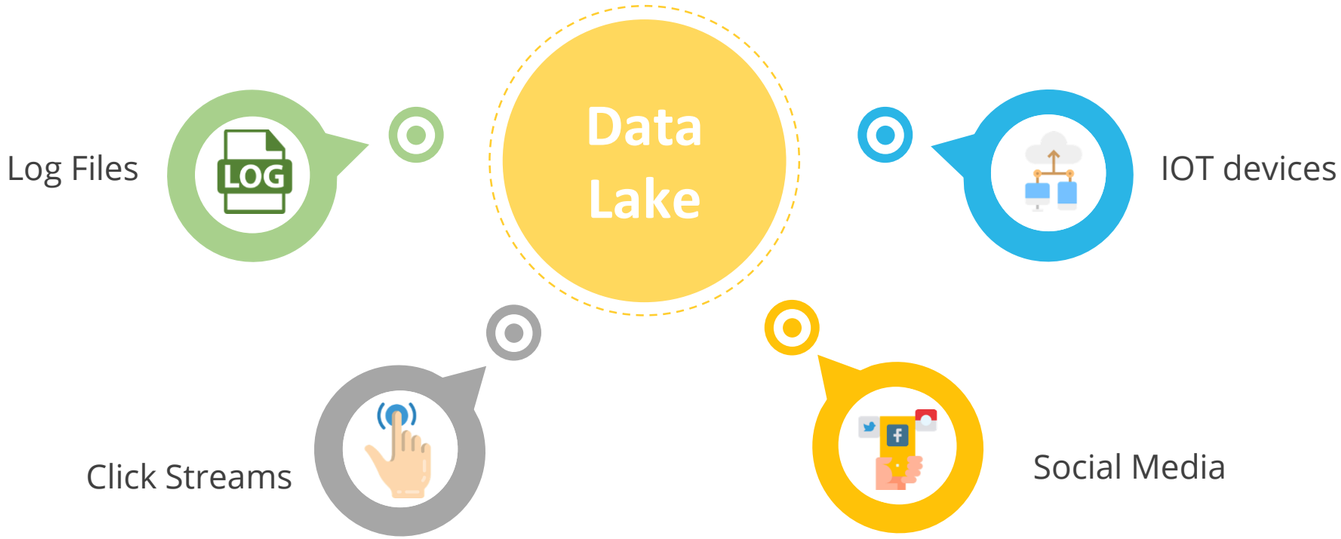
- Understand various Big Data Ingestion Tools
- Define Sqoop and its uses
- Analyze importing and exporting data from Hadoop using Sqoop
- Explain Apache Flume along with its uses
- Explain the components in the Flume architecture
- Define Kafka and its architecture

DATA AND
ARTIFICIAL INTELLIGENCE

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

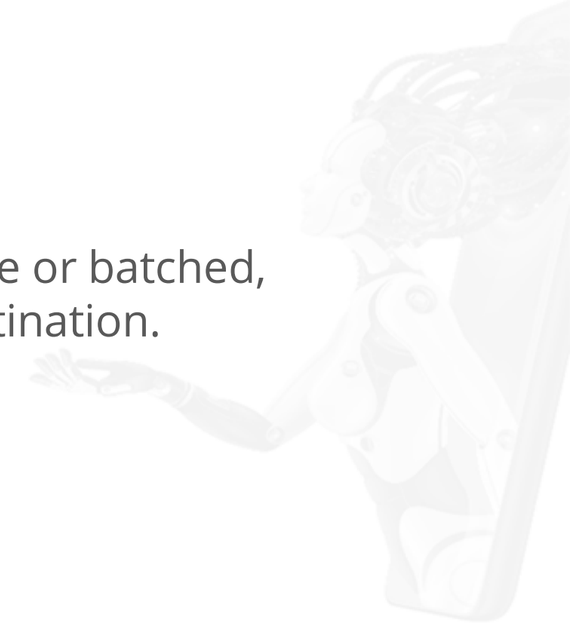
Data Ingestion

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.

DATA AND
ARTIFICIAL INTELLIGENCE

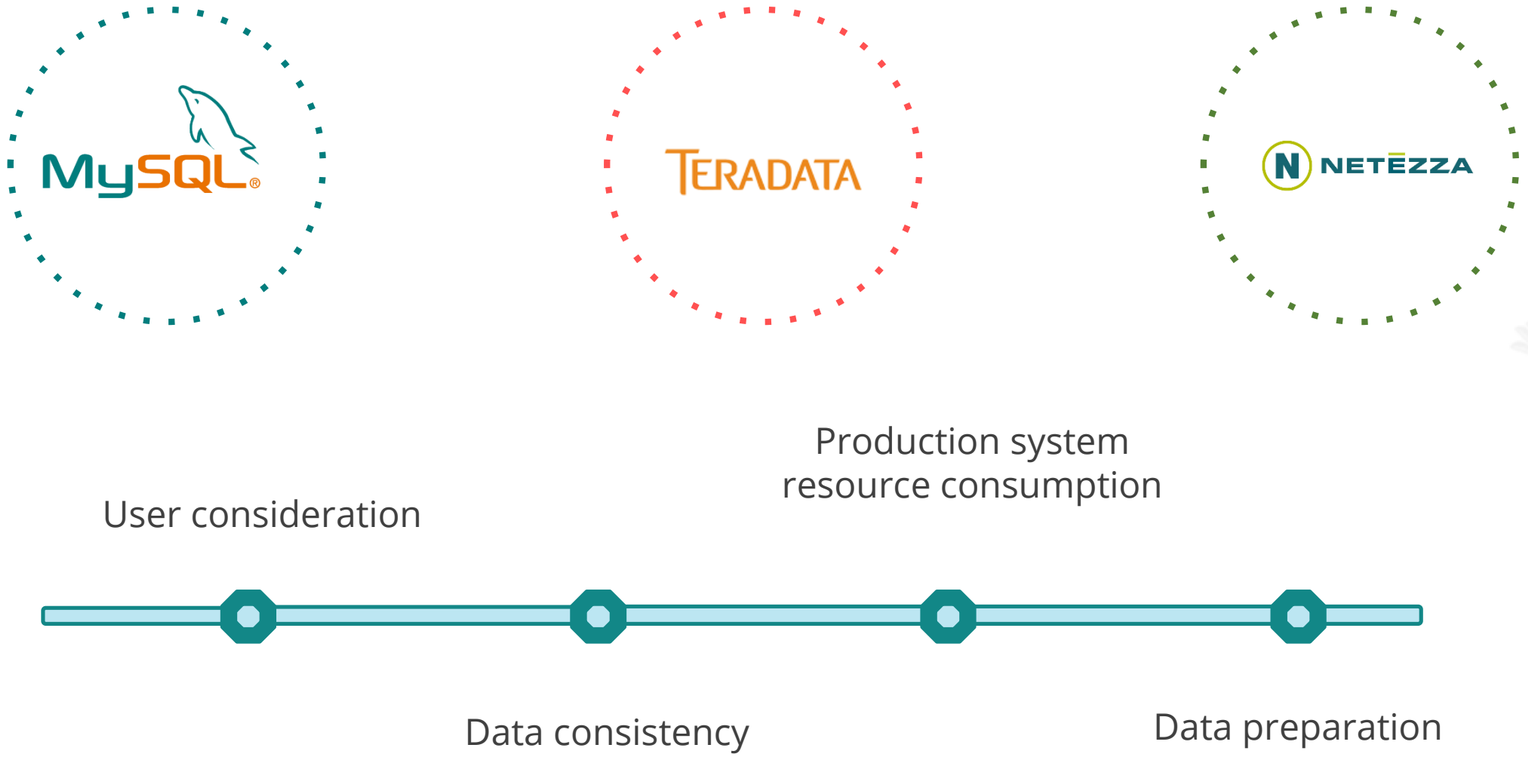
Apache Sqoop

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.

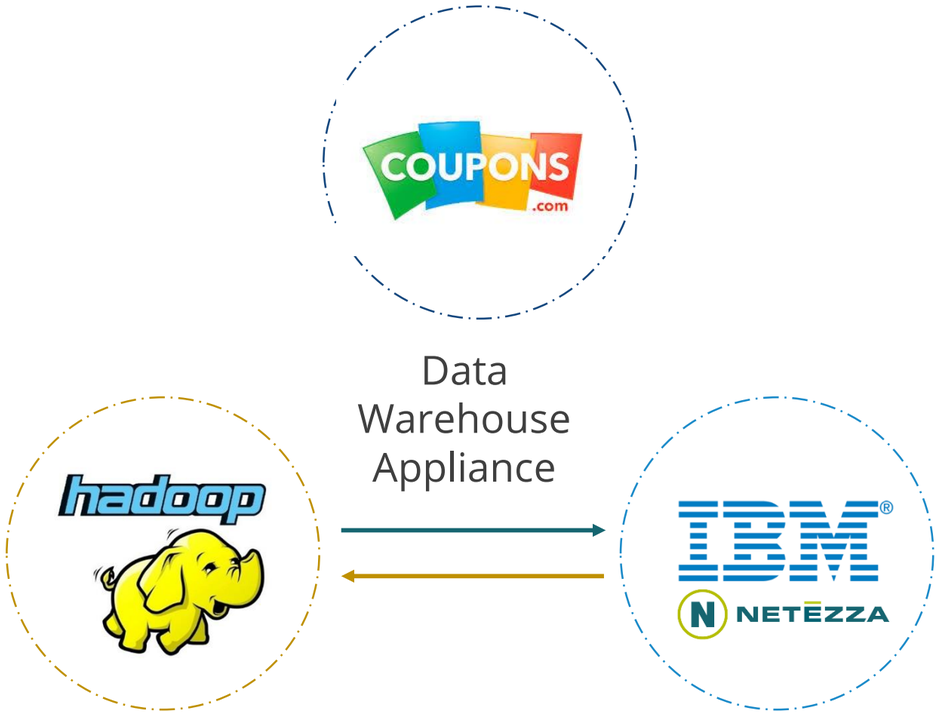
Why Sqoop?

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:

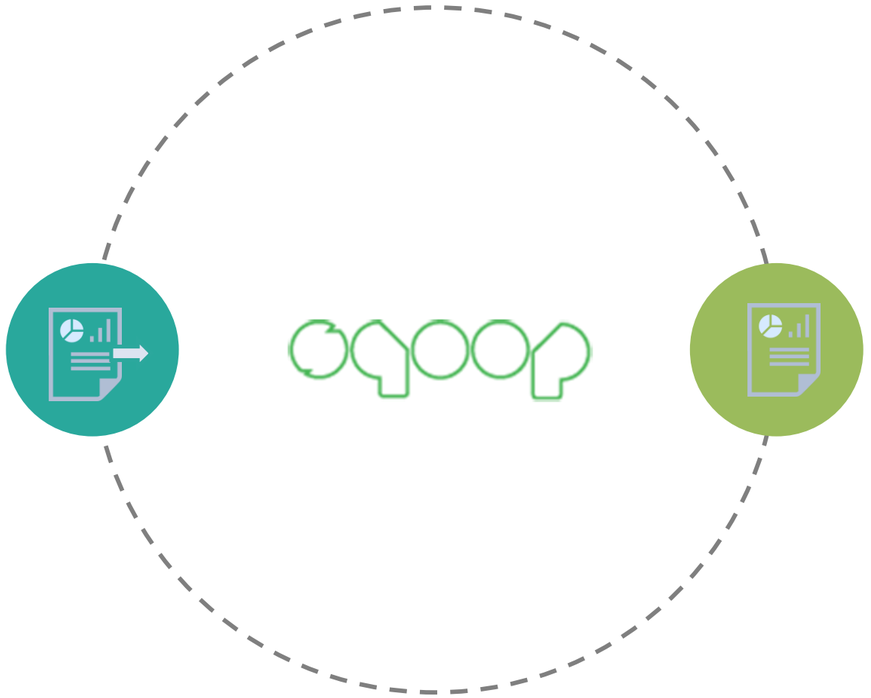
- SQL servers are deployed worldwide
- Nightly processing is done on SQL servers
- Allows to move certain parts of data from traditional SQL DB to Hadoop
- Transfers data efficiently and swiftly
- Handles large data through ecosystem
- Brings processed data from Hadoop to the applications

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.

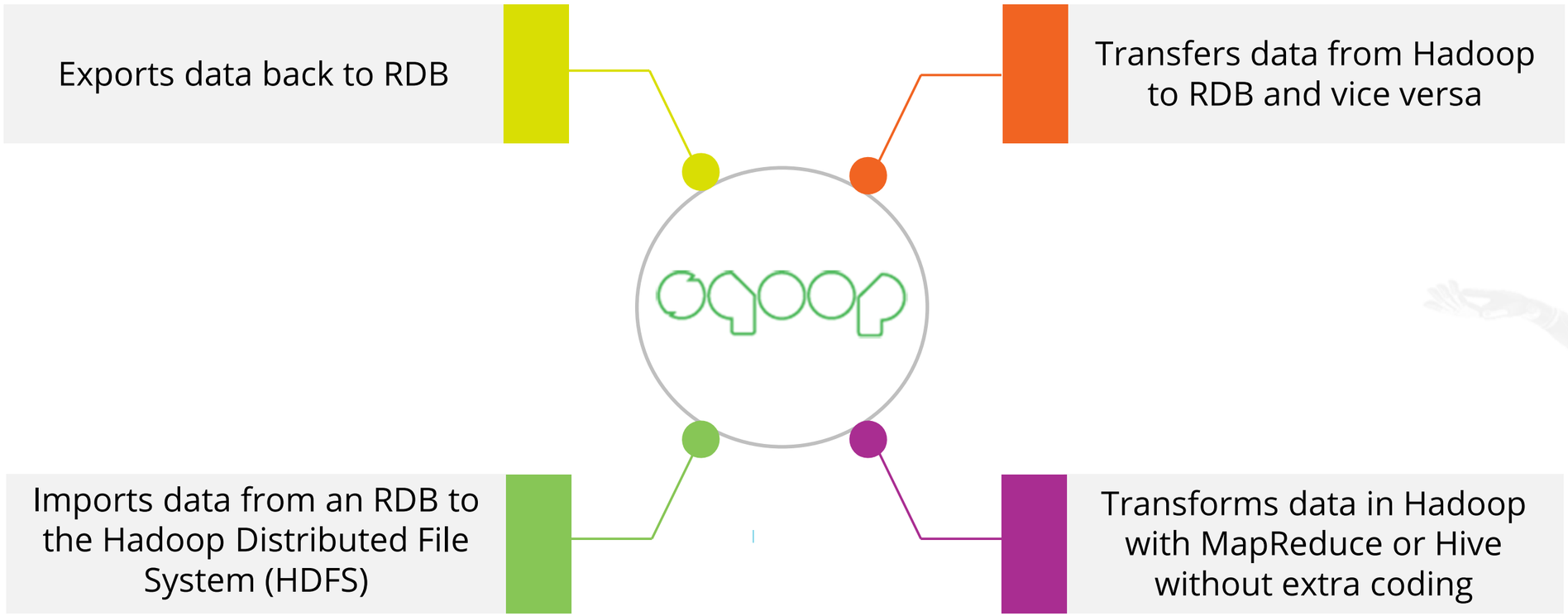


While importing database from Hadoop to RDB:

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:



DATA AND
ARTIFICIAL INTELLIGENCE

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.

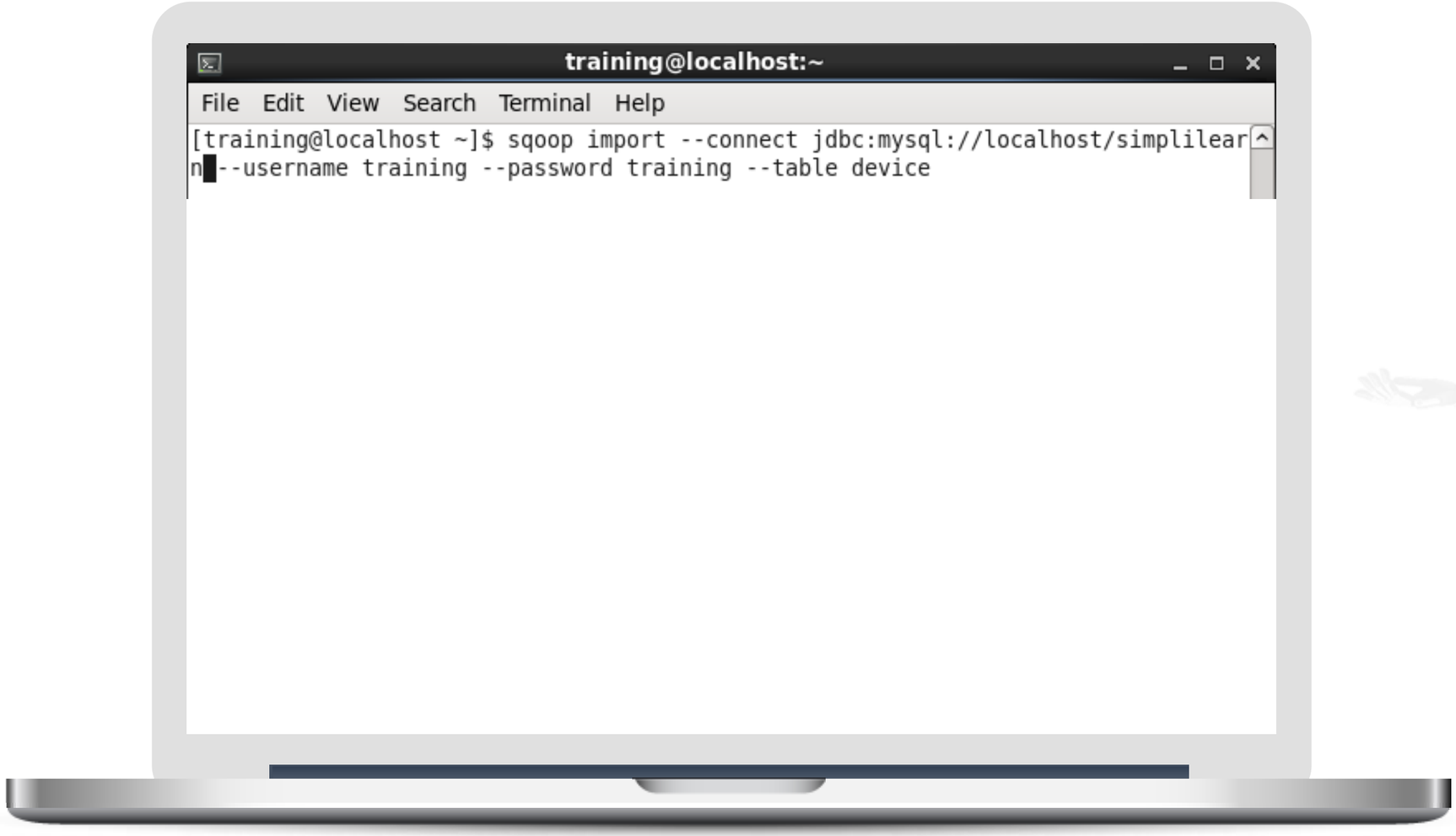
A map-only job is launched with individual mappers responsible for transferring a slice of the dataset.

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:



Assisted Practice

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.

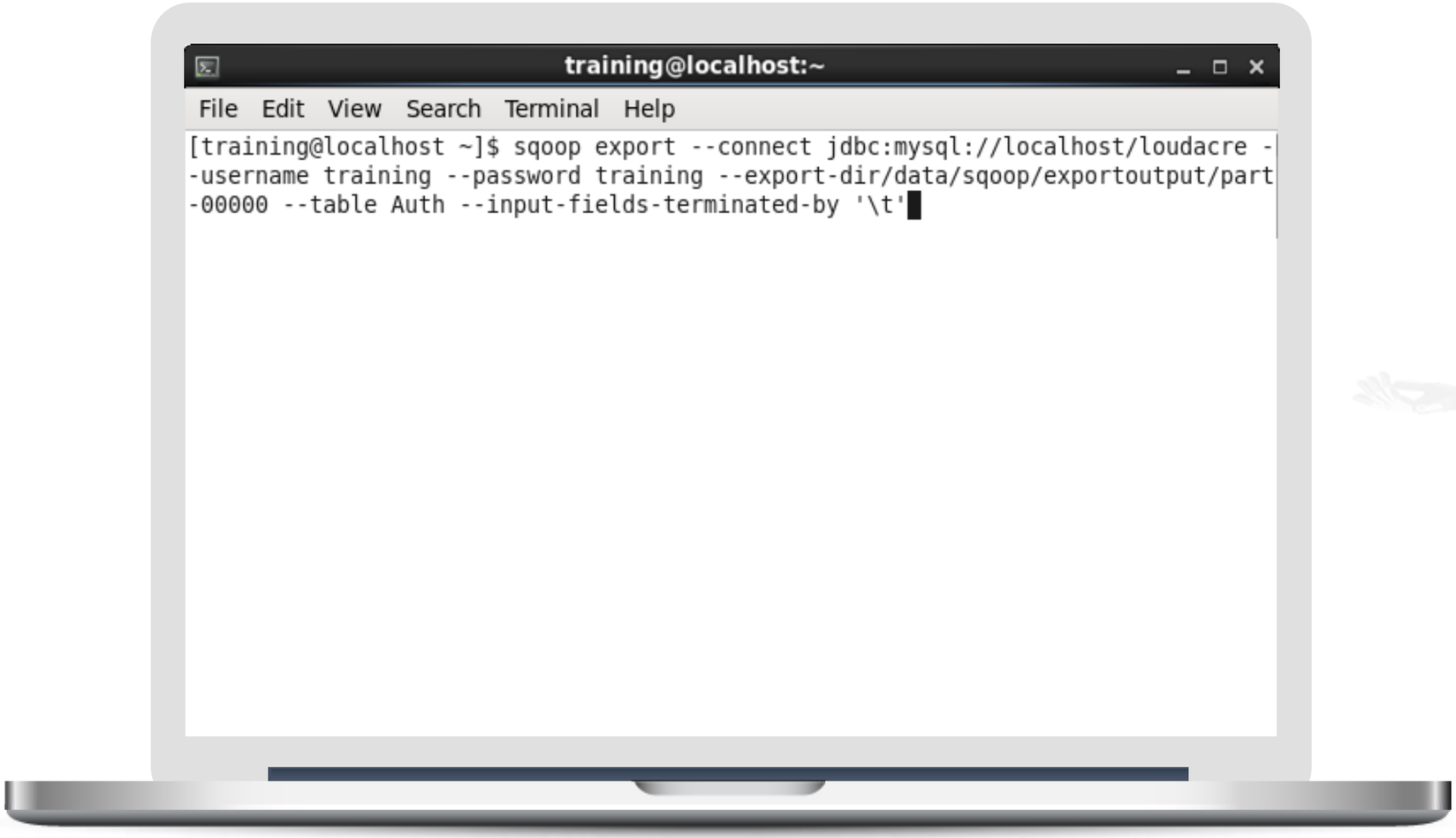
Users can:

- 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.

DATA AND
ARTIFICIAL INTELLIGENCE

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

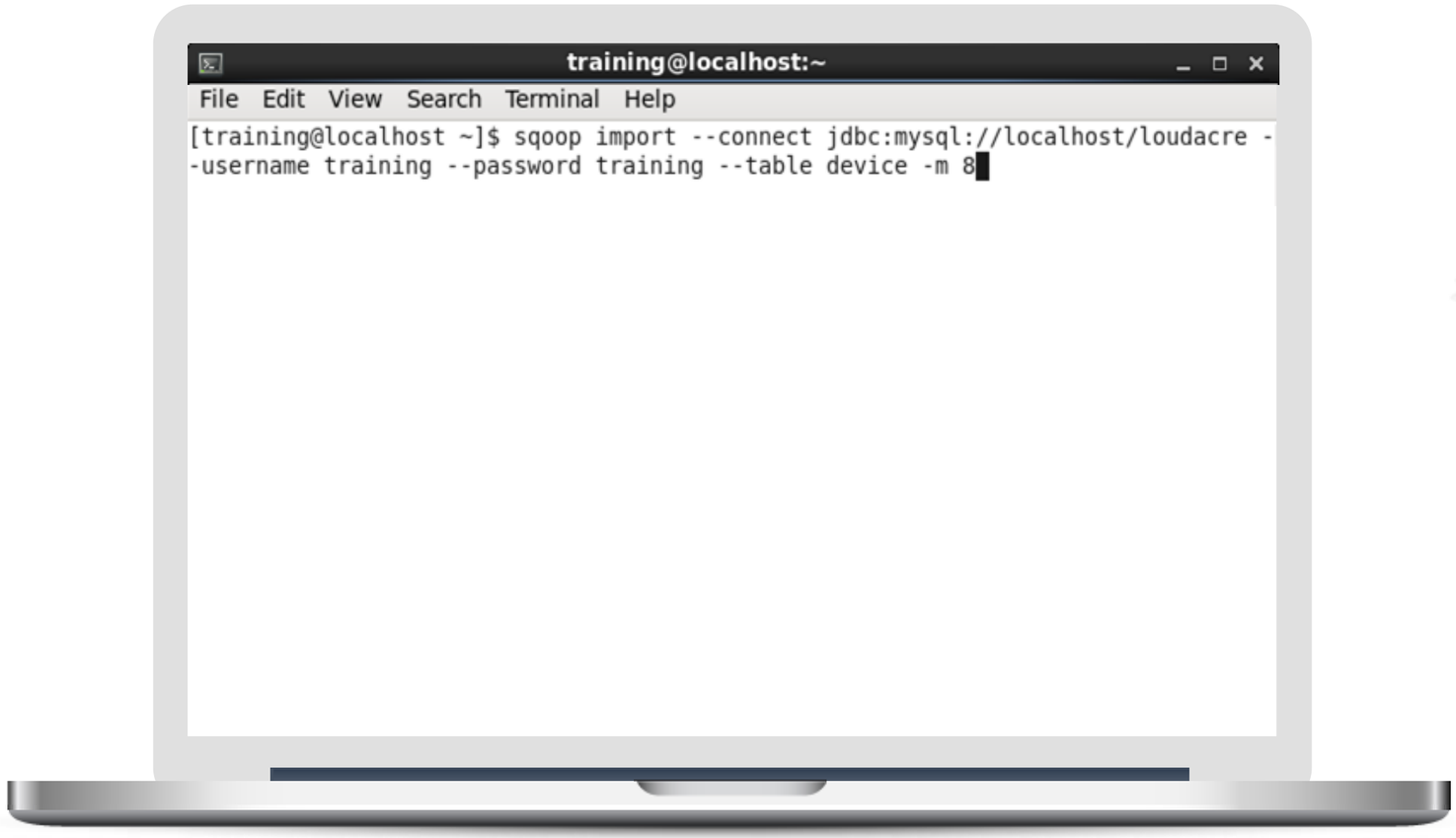
Specializes in using specific batch tools to transfer data with high throughput

Default Sqoop connector

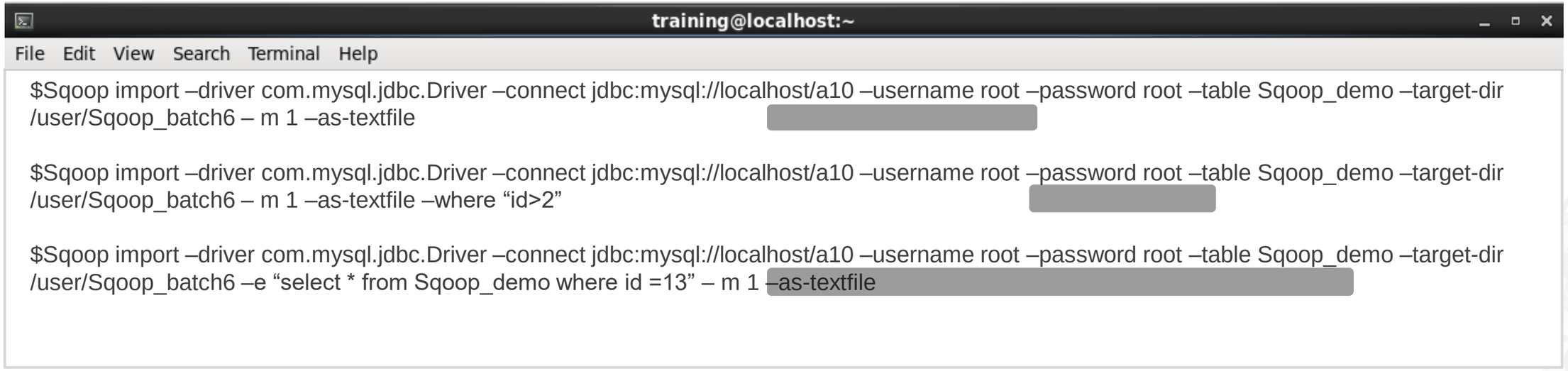
Designed for specific databases such as MySQL, PostgreSQL, Oracle, SQL Server, and DB2

Controlling Parallelism

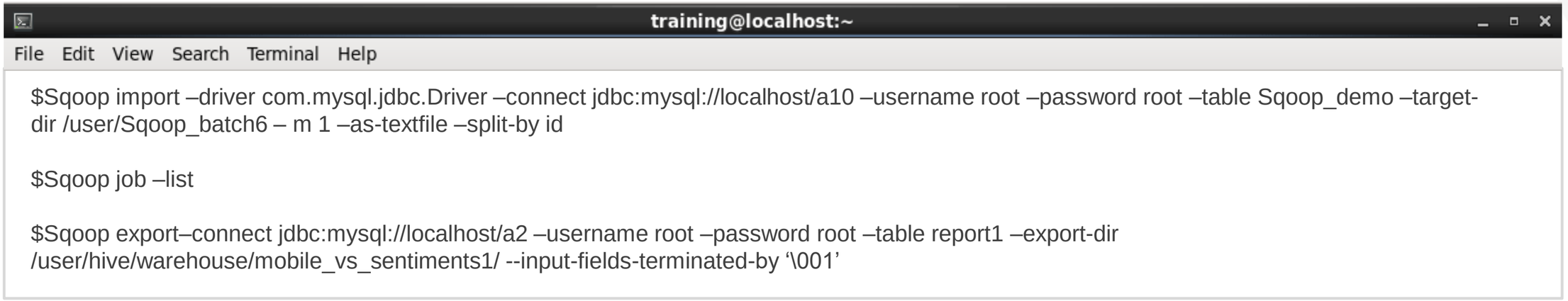
- By default, Sqoop typically imports data using four parallel tasks called mappers
- Increasing the number of tasks might improve import speed
- You can influence the number of tasks using the -m or --num-mappers option



Sample Sqoop Commands



More sample Sqoop commands:



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.

Assisted Practice

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**.

Unassisted Practice

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

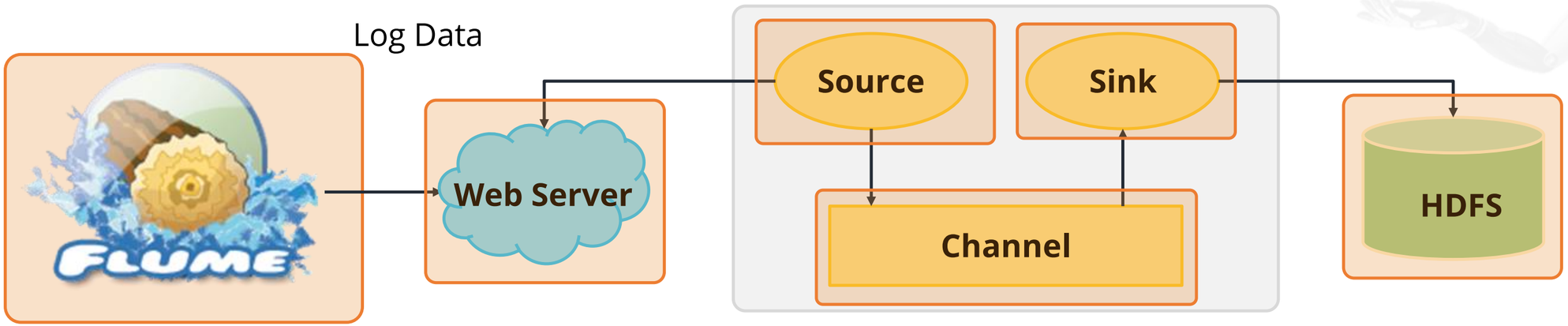
DATA AND
ARTIFICIAL INTELLIGENCE

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

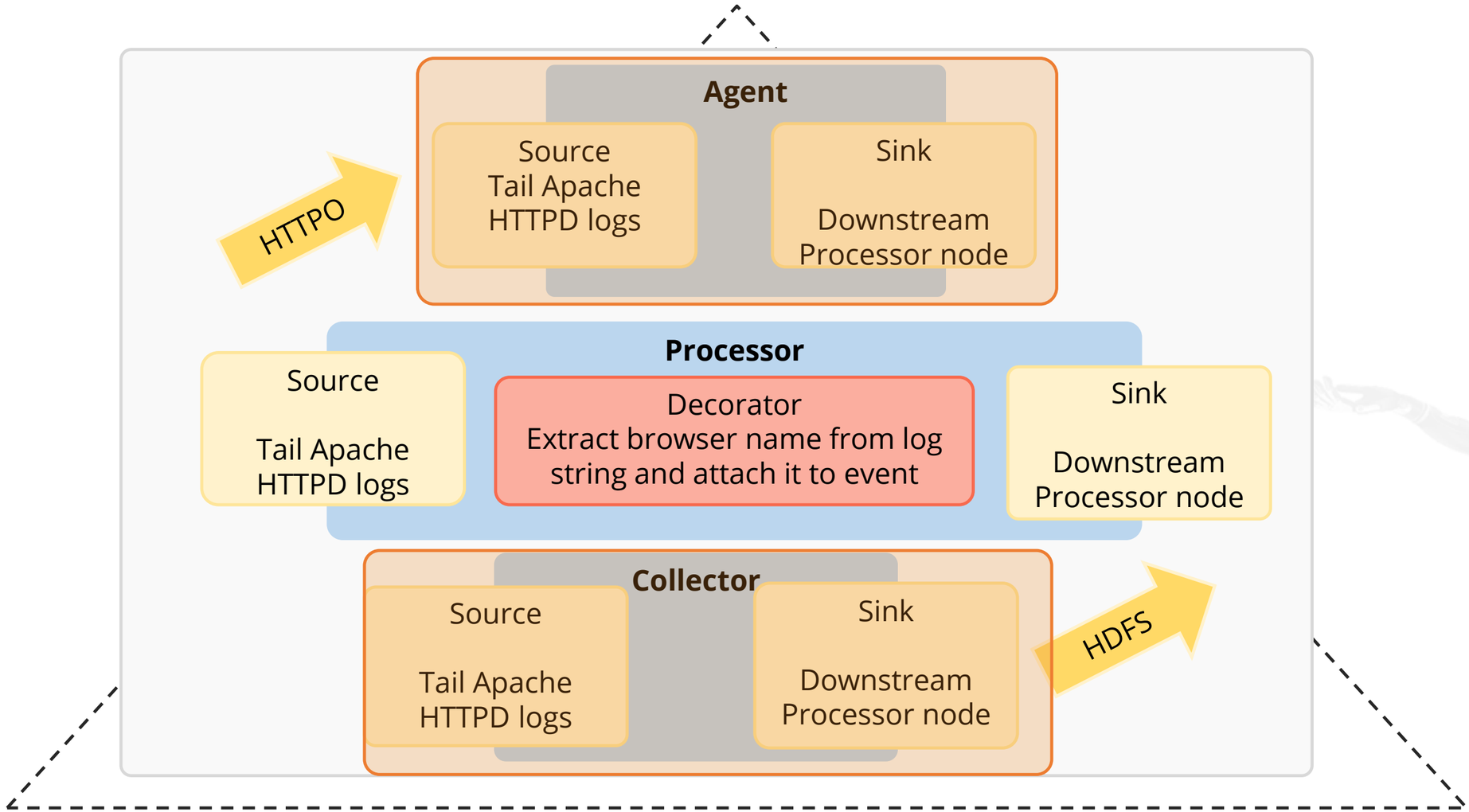
To solve this problem log aggregation tool called Flume can be used. Apache Sqoop and Flume are the tools that are used to gather data from different sources and load them into HDFS. Sqoop in Hadoop is used to extract structured data from databases like Teradata, Oracle, and so on, whereas Flume in Hadoop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

DATA AND
ARTIFICIAL INTELLIGENCE

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

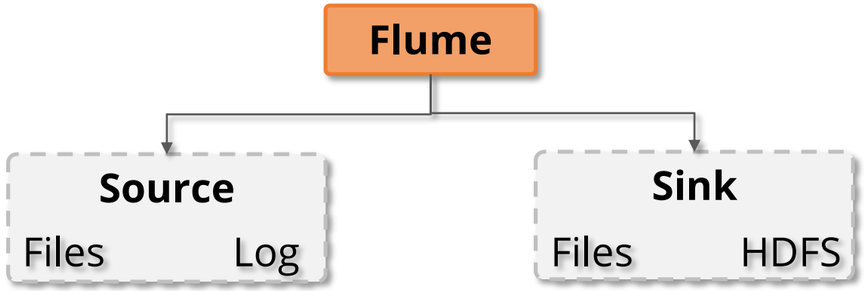
Extensibility in Flume

Extensibility:

- The ability to add new functionality to a system

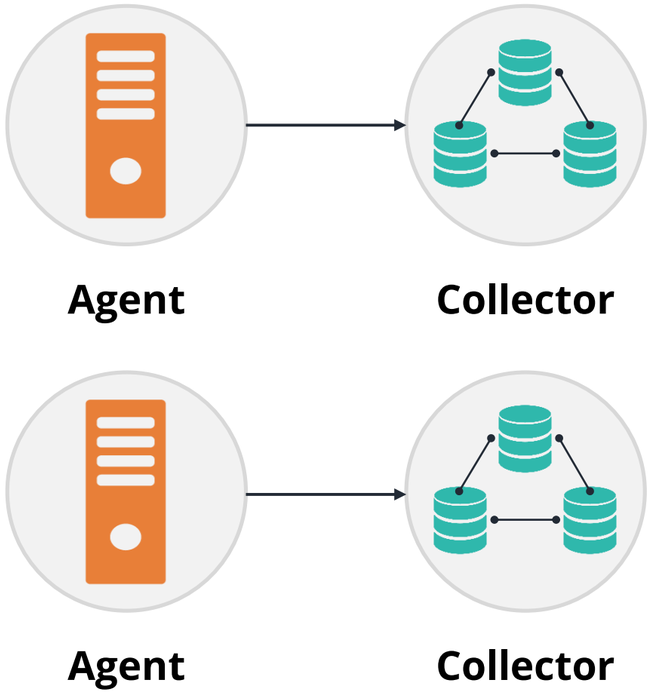
Flume can be extended by adding Sources and Sinks to existing storage layers or data platforms

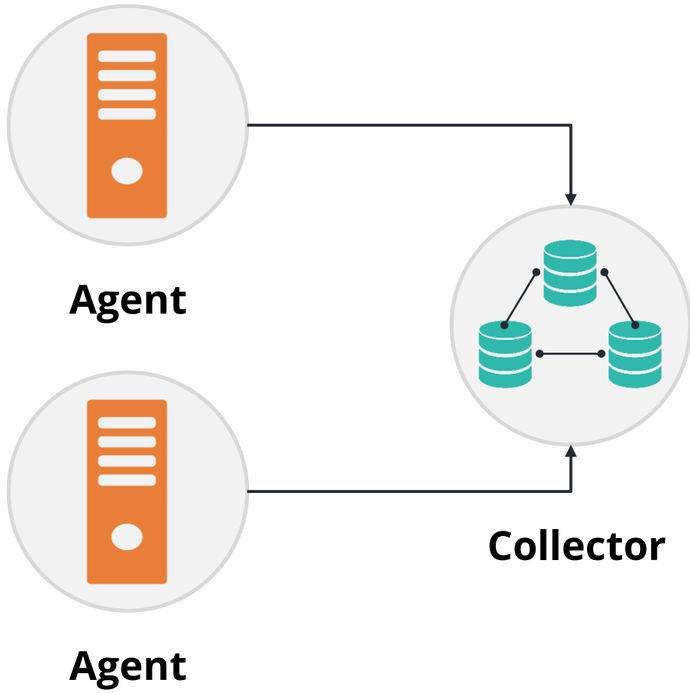
- General Sources include data from files, syslog, and standard output from any Linux process
- General Sinks include files on the local filesystem or HDFS
- Developers can write their own Sources or Sinks



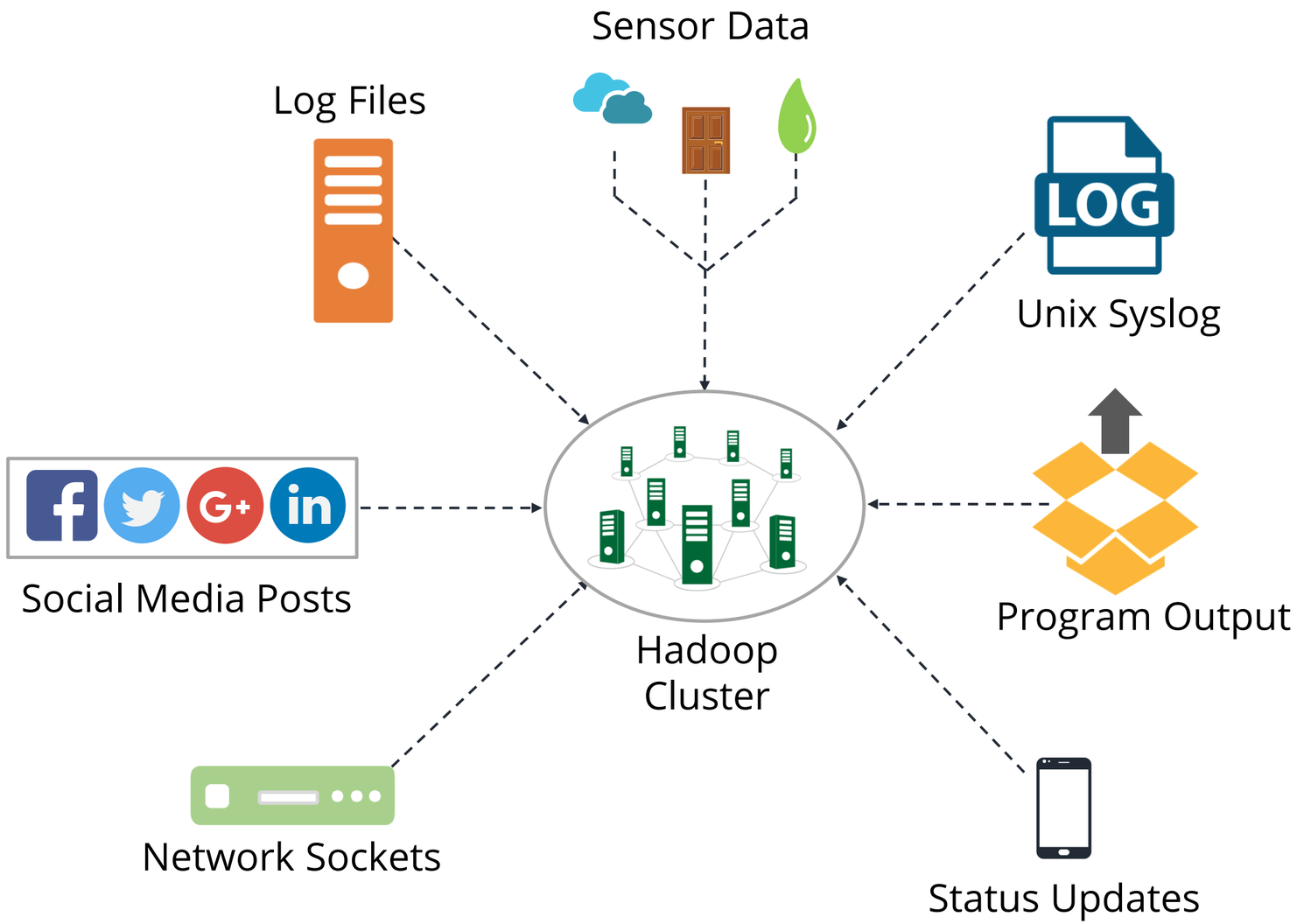
Scalability in Flume

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



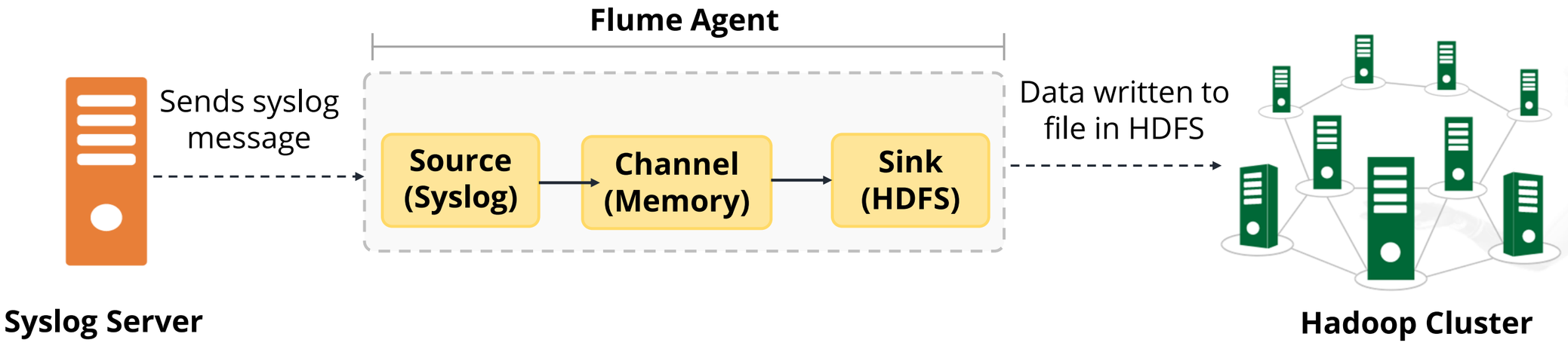


Common Flume Data Sources



Flume Data Flow

The diagram illustrates how syslog data is captured to HDFS.



DATA AND
ARTIFICIAL INTELLIGENCE

Components in Flume's Architecture

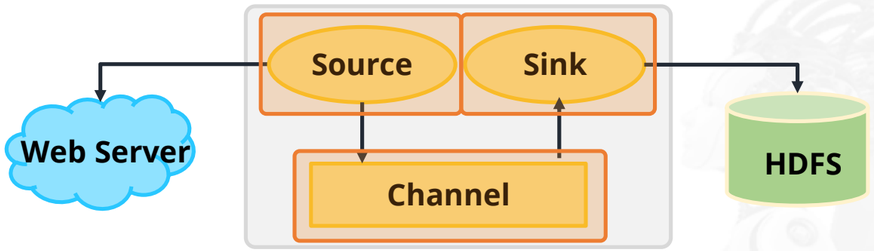
Components in Flume's Architecture

- **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:

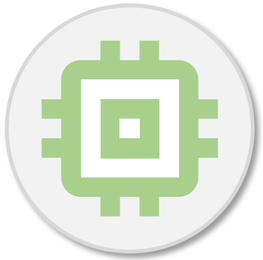
Null
Discards all events received
(Flume equivalent of **/dev/null**)

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

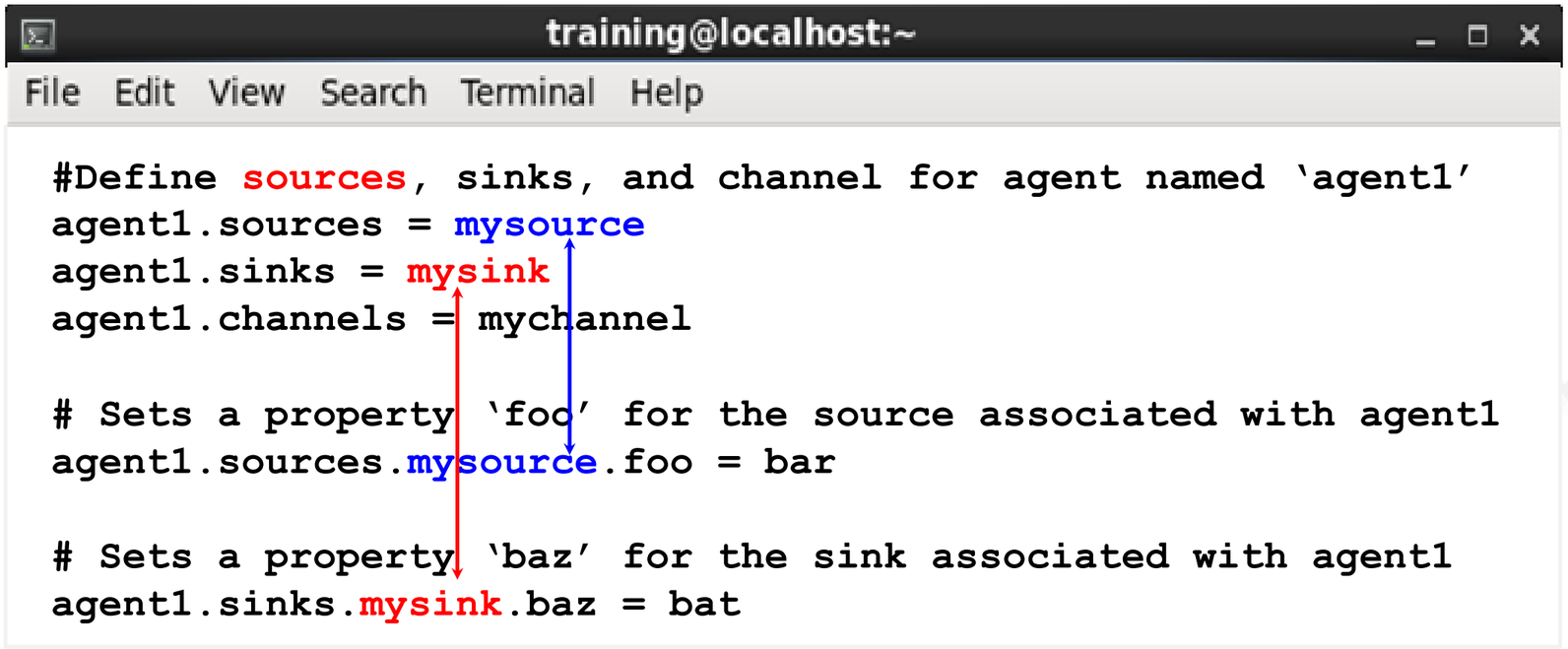
- Stores events on the machine's local disk
- Slower than RAM, but more reliable as data is written to disk

JDBC

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

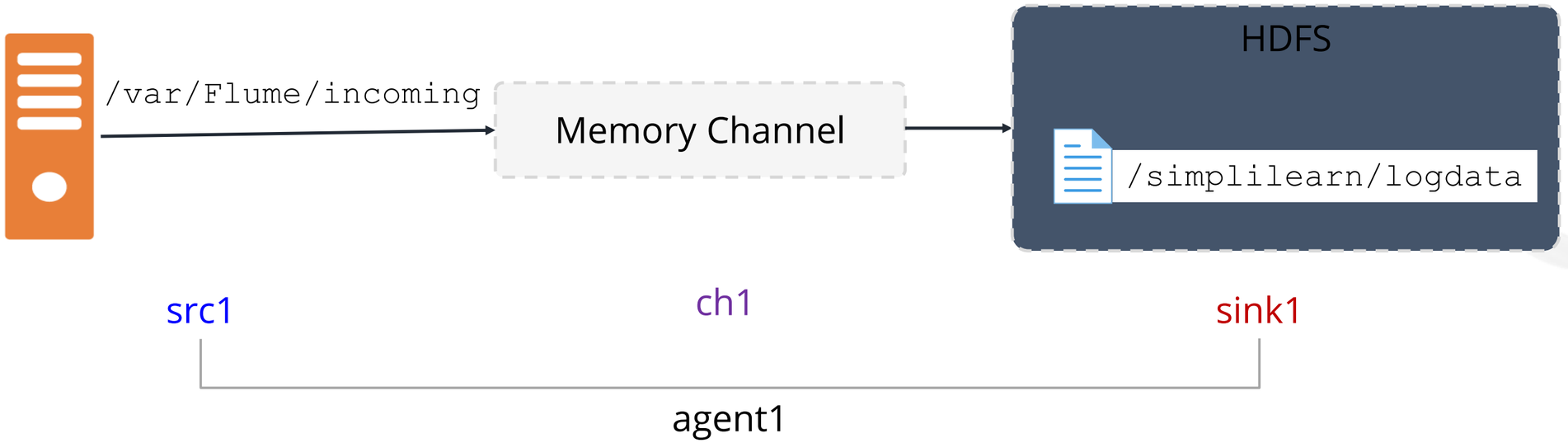
Flume Agent Configuration File

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



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

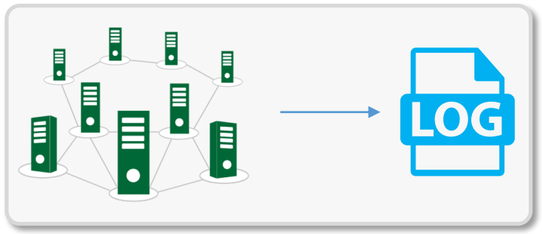
```
agent1.sources = src1
agent1.sinks = sink1
agent.channels = ch1

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

Assisted Practice

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

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.

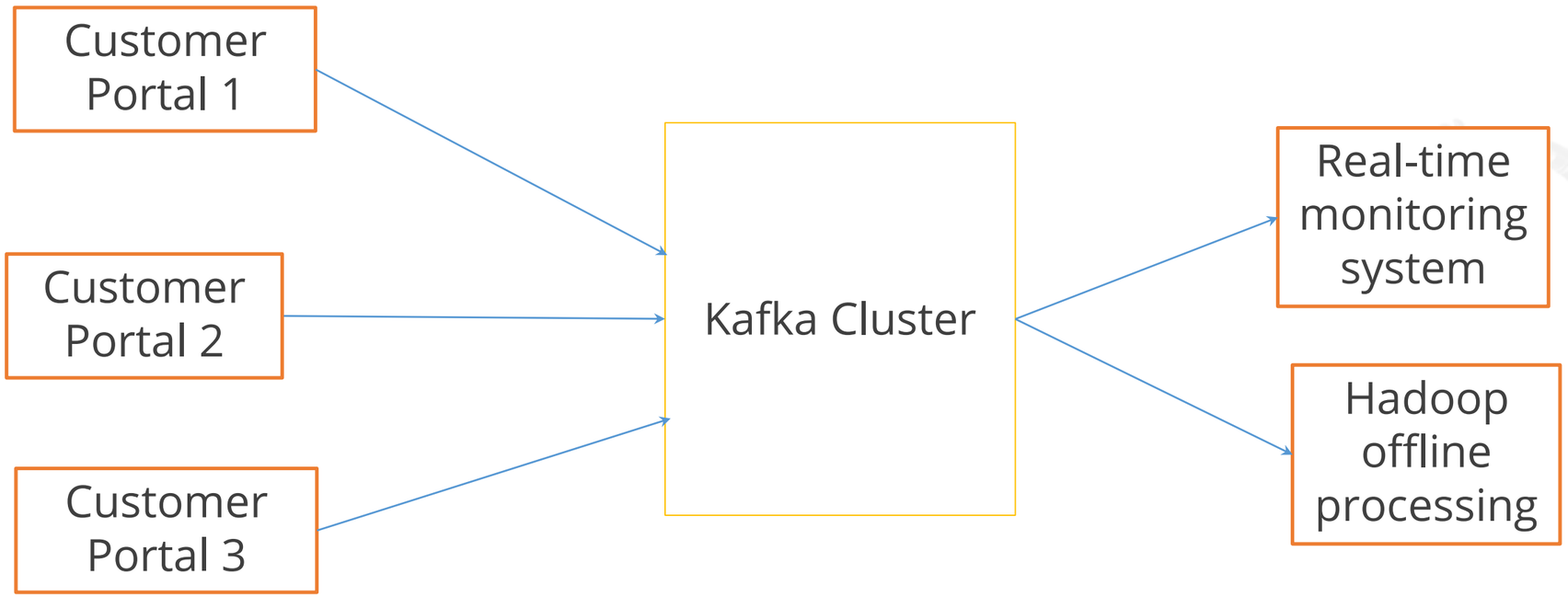
Apache Kafka: Use Cases

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

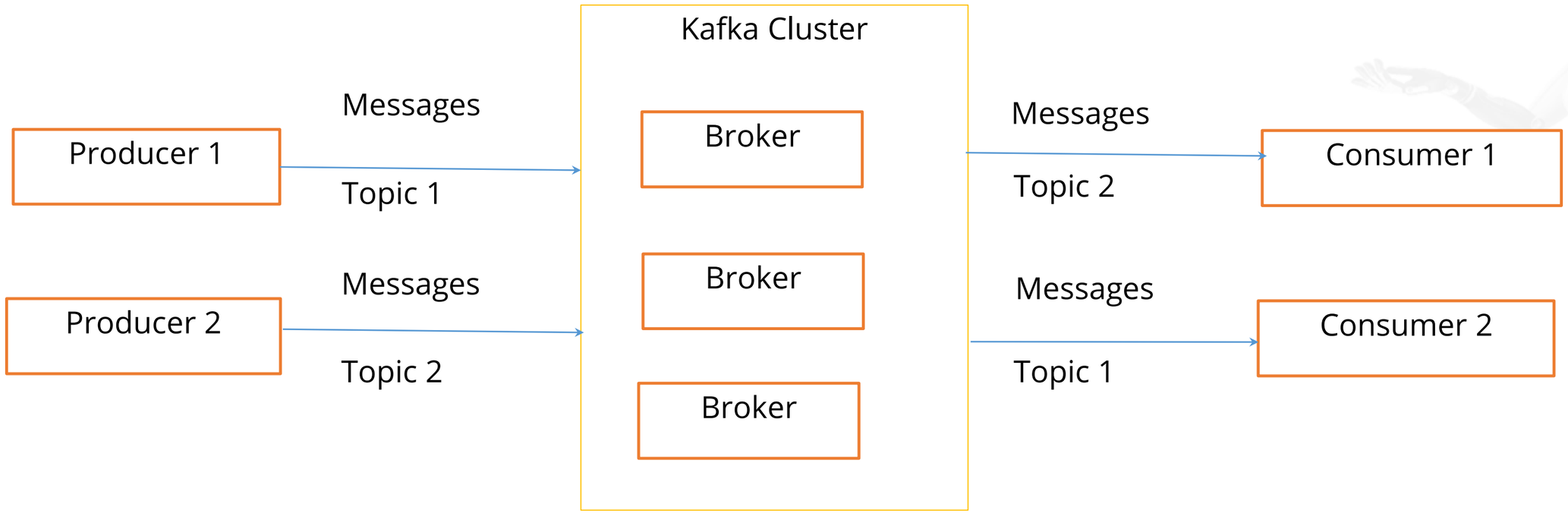
DATA AND
ARTIFICIAL INTELLIGENCE

Kafka Data Model

Kafka Data Model

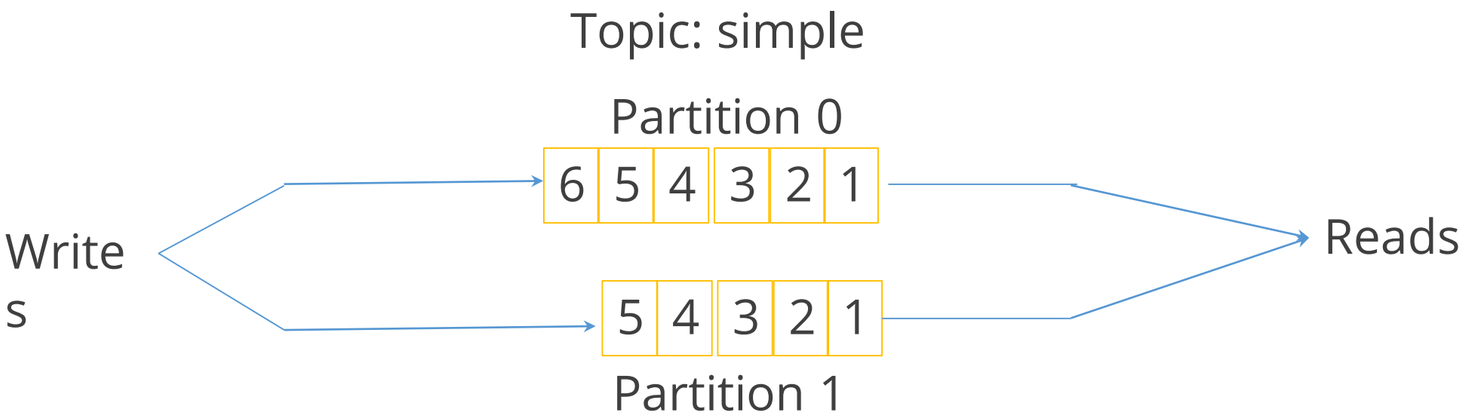
The Kafka data model consists of messages and topics.

- 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

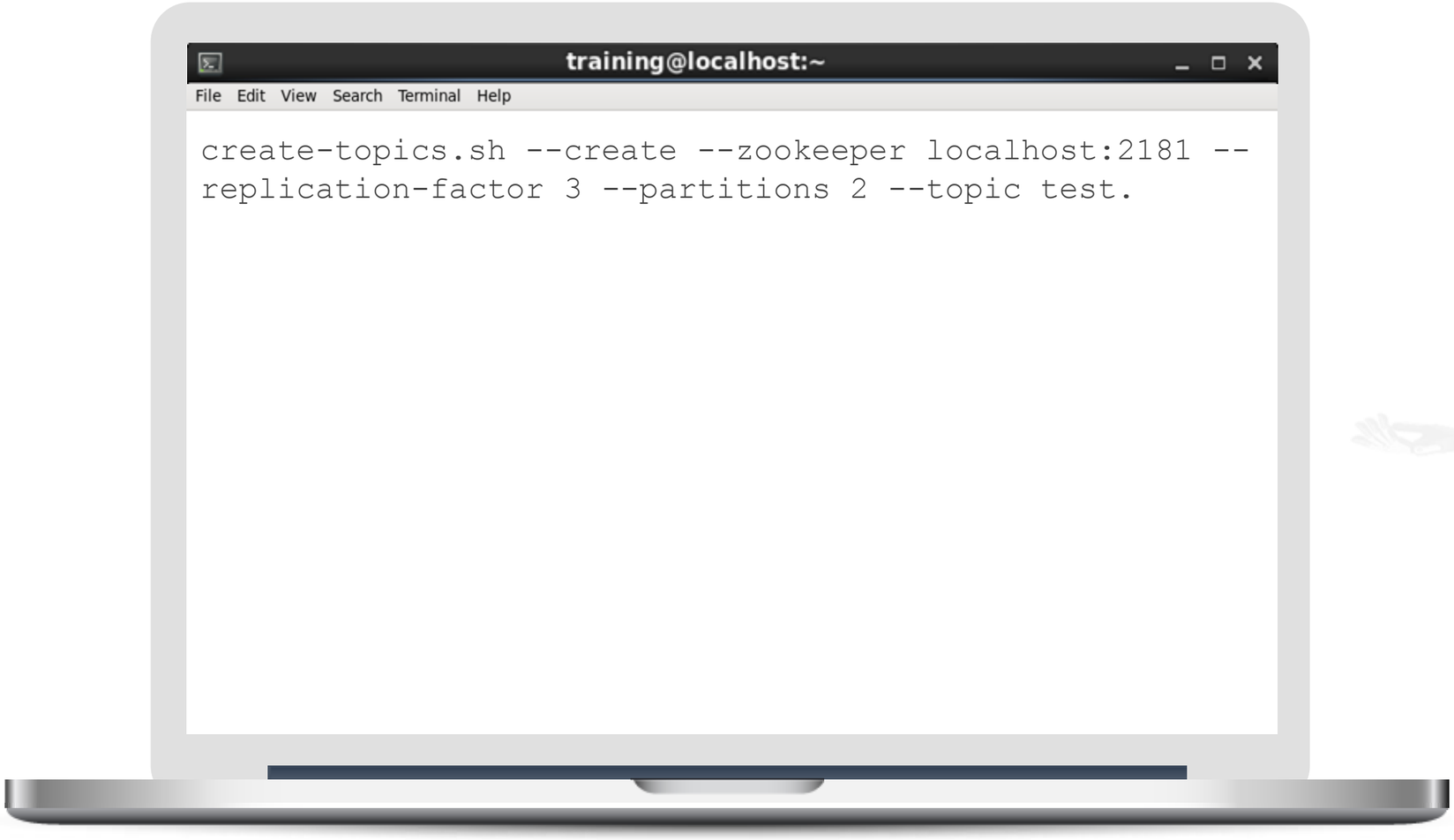
A topic is a category of messages in Kafka.



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

Topics

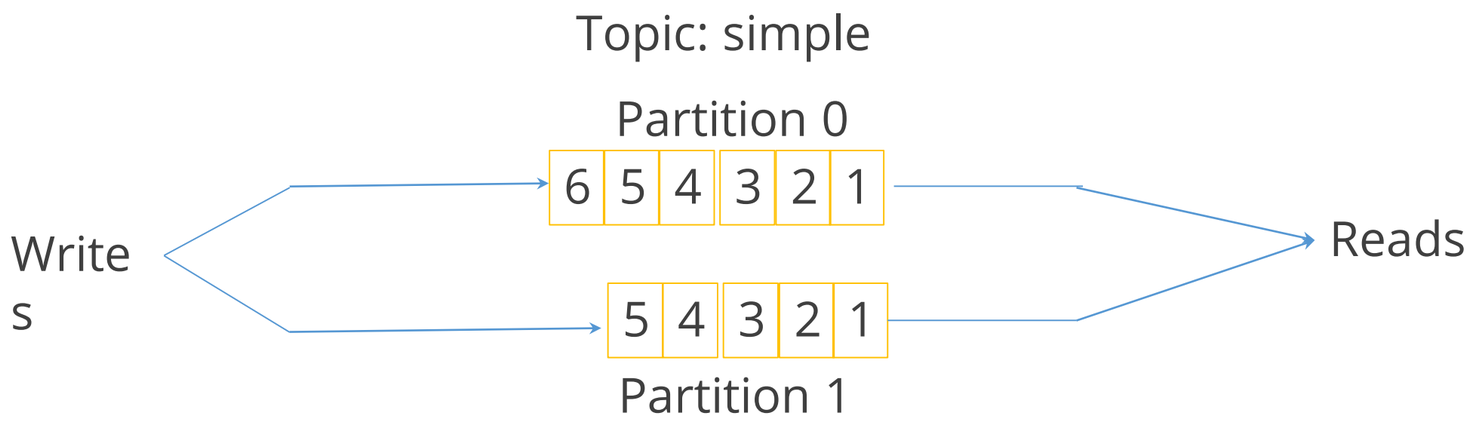
Kafka provides the kafka-topics.sh command to create and modify topics.



Partitions

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

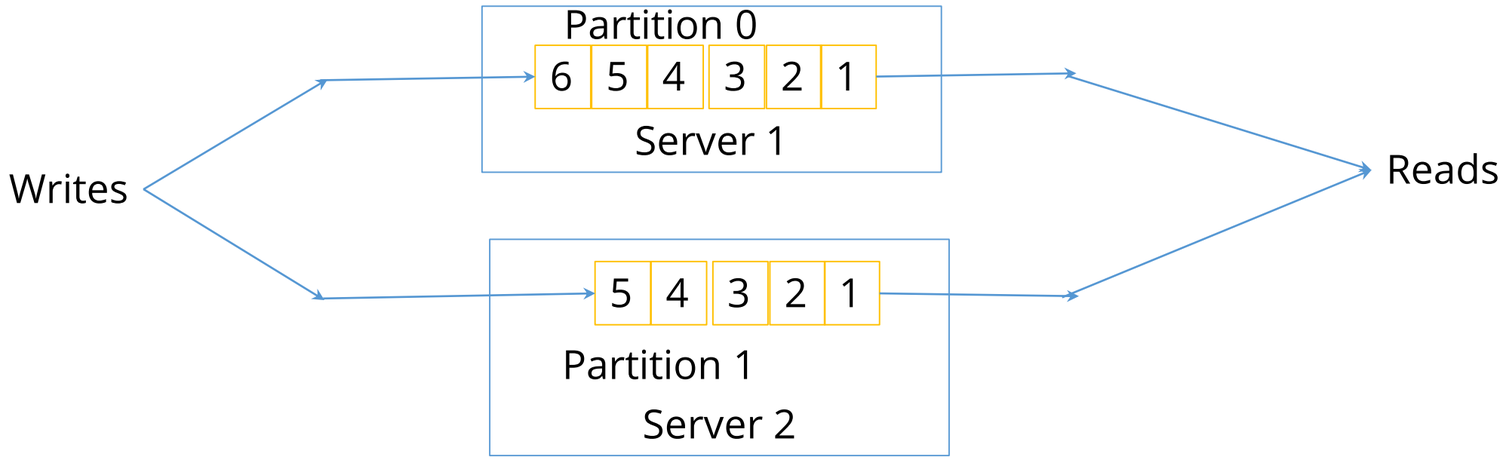
- Partitions allow messages in a topic to be distributed to multiple servers.
- A topic can have any number of partitions.
- Each partition should fit in a single Kafka server.
- The number of partitions decide the parallelism of the topic.



Partition Distribution

Partitions can be distributed across the Kafka cluster.

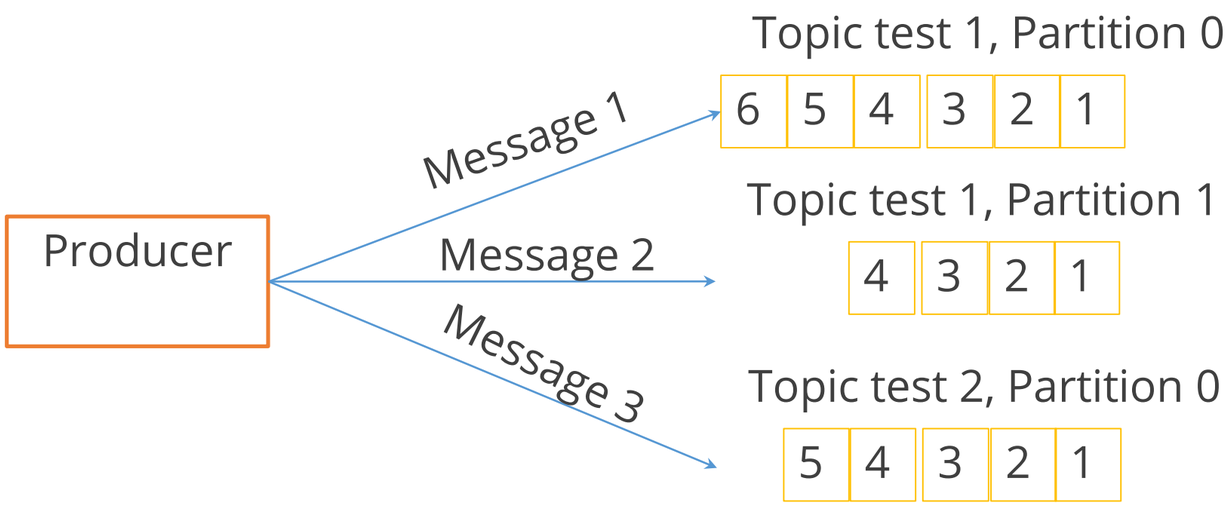
- Each Kafka server may handle one or more partitions.
- A partition can be replicated across several servers for fault-tolerance.
- One server is marked as a leader for the partition and the others are marked as followers.
- The leader controls the read and write for the partition, whereas, the followers 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.

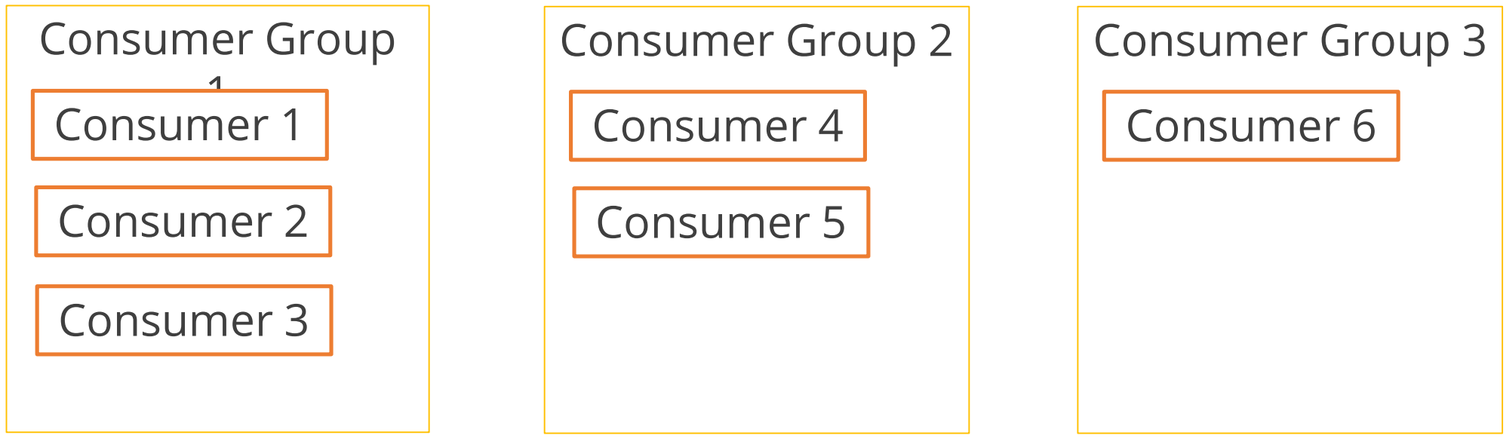
- The producers place the message to a particular topic.
- The producers also decide which partition to place the message into.
- Topics should already exist before a message is placed by the producer.
- Messages are added at one end of the partition.



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.

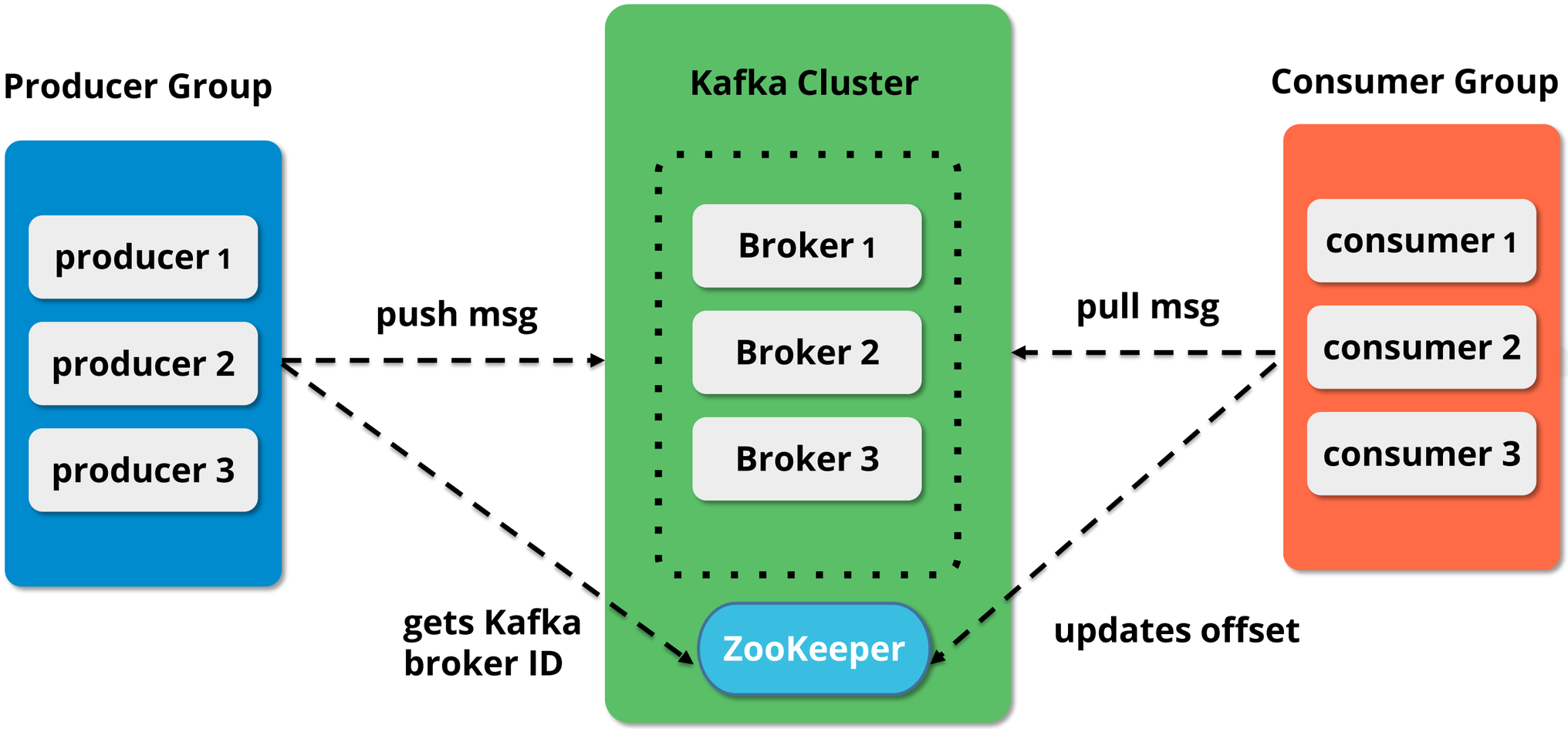


DATA AND
ARTIFICIAL INTELLIGENCE

Apache Kafka Architecture

Kafka Architecture

Given below is a Kafka cluster architecture:



Assisted Practice

Apache Kafka

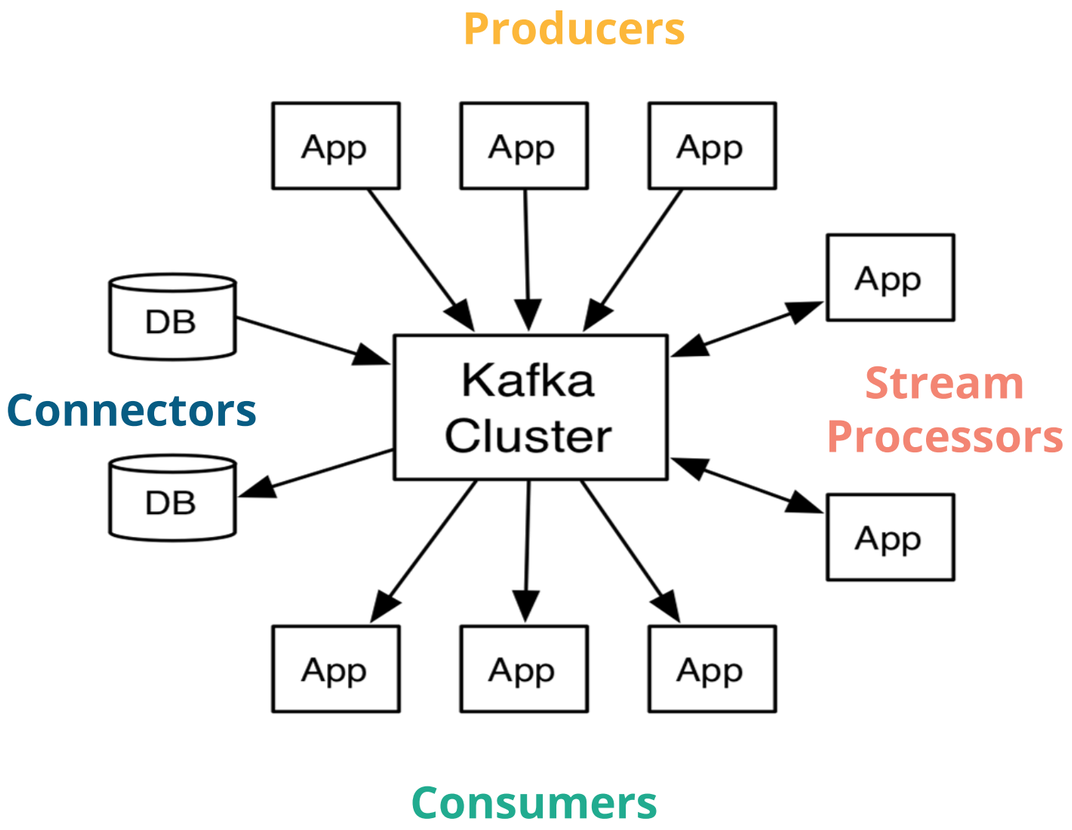
Duration: 15 mins

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.

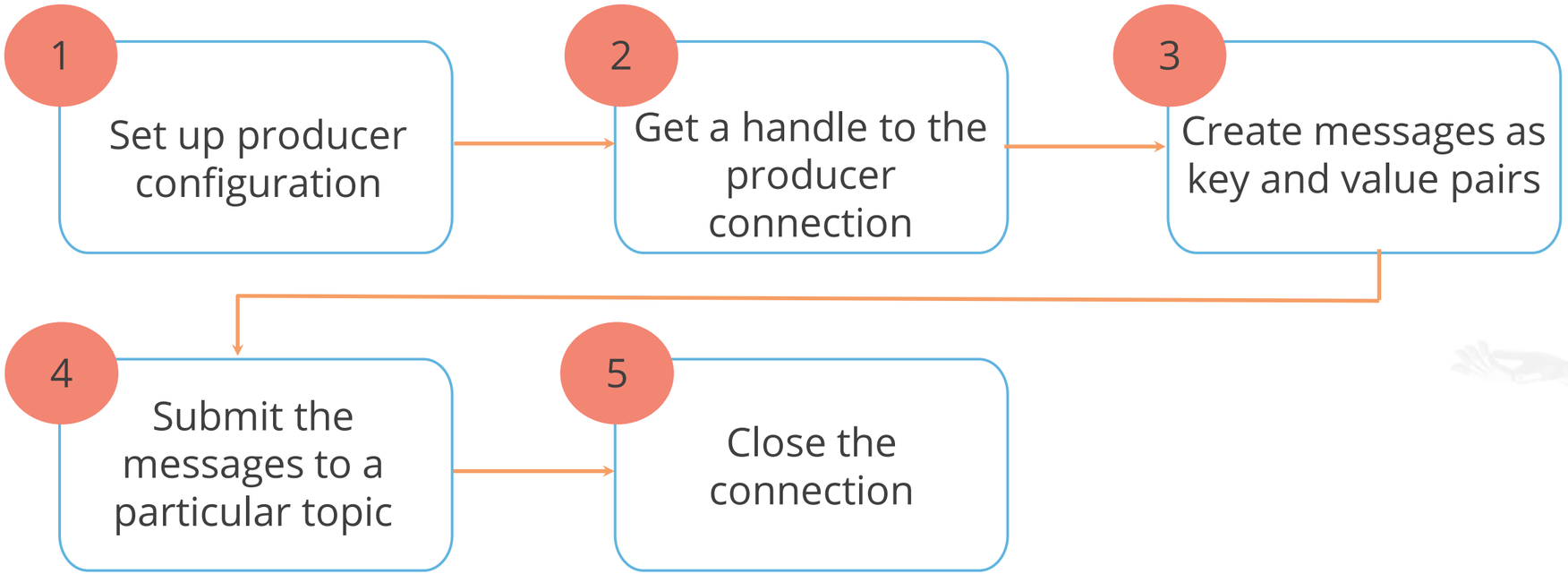
The **Streams API** allows transforming stream of data from input topics to output topics.

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);
```

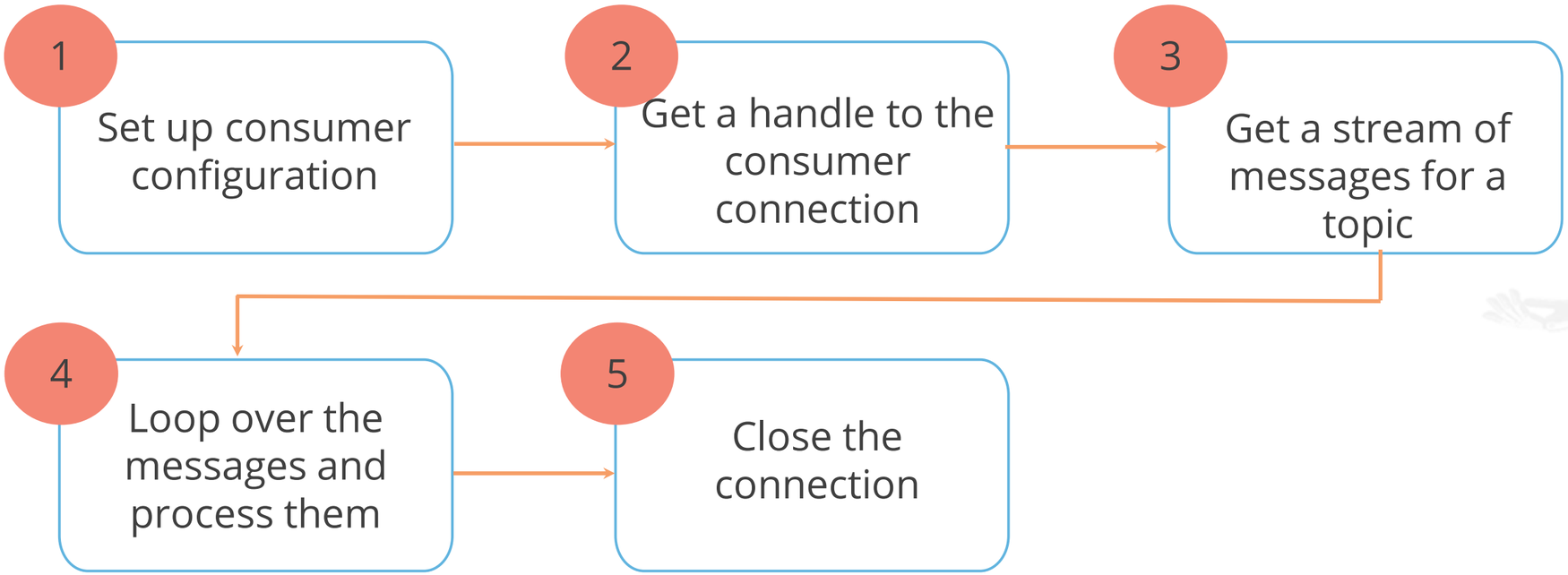
Producer Side API Example: Step 5

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()));
}
```

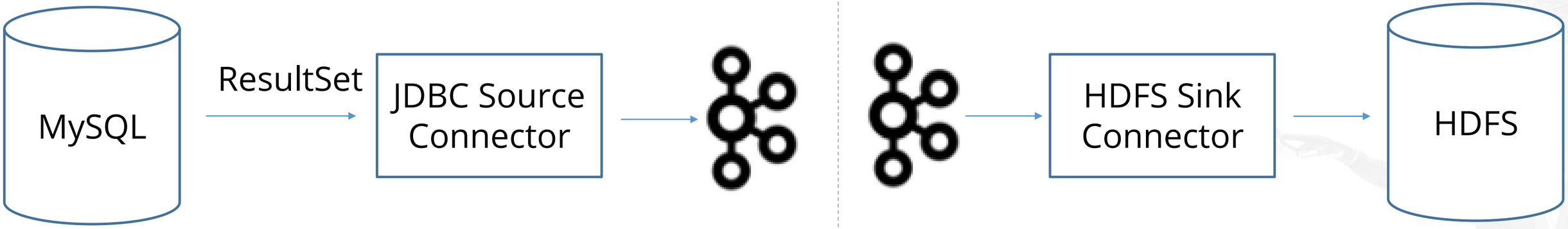
Consumer Side API Example: Step 5

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

Assisted Practice

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

Quiz

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:

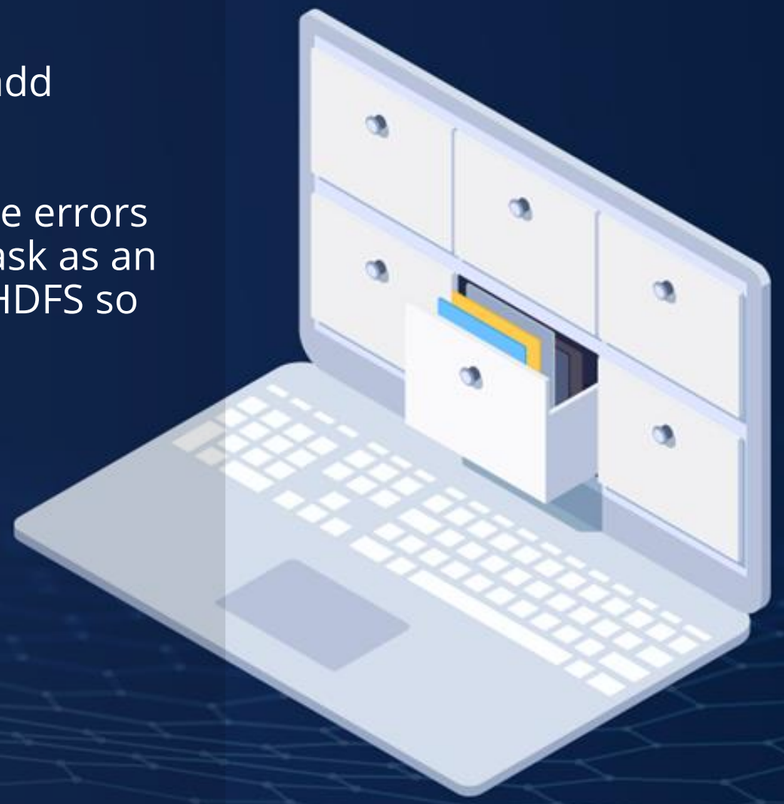
Target.com is one of the biggest e-commerce giants in the US.
They have a web portal where customers can purchase items, and the sellers can add products, remove the products, and do inventory-related operations.

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



DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Distributed Processing: MapReduce Framework and Pig

Learning Objectives

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

- Perform distributed processing in MapReduce
- Understand issues with distributed processing and how MR reduces them
- Implement MapReduce paradigm, divide-and-conquer
- Perform Word Count program - Hello World of big data
- Understand Pig
- Implement Pig Latin commands

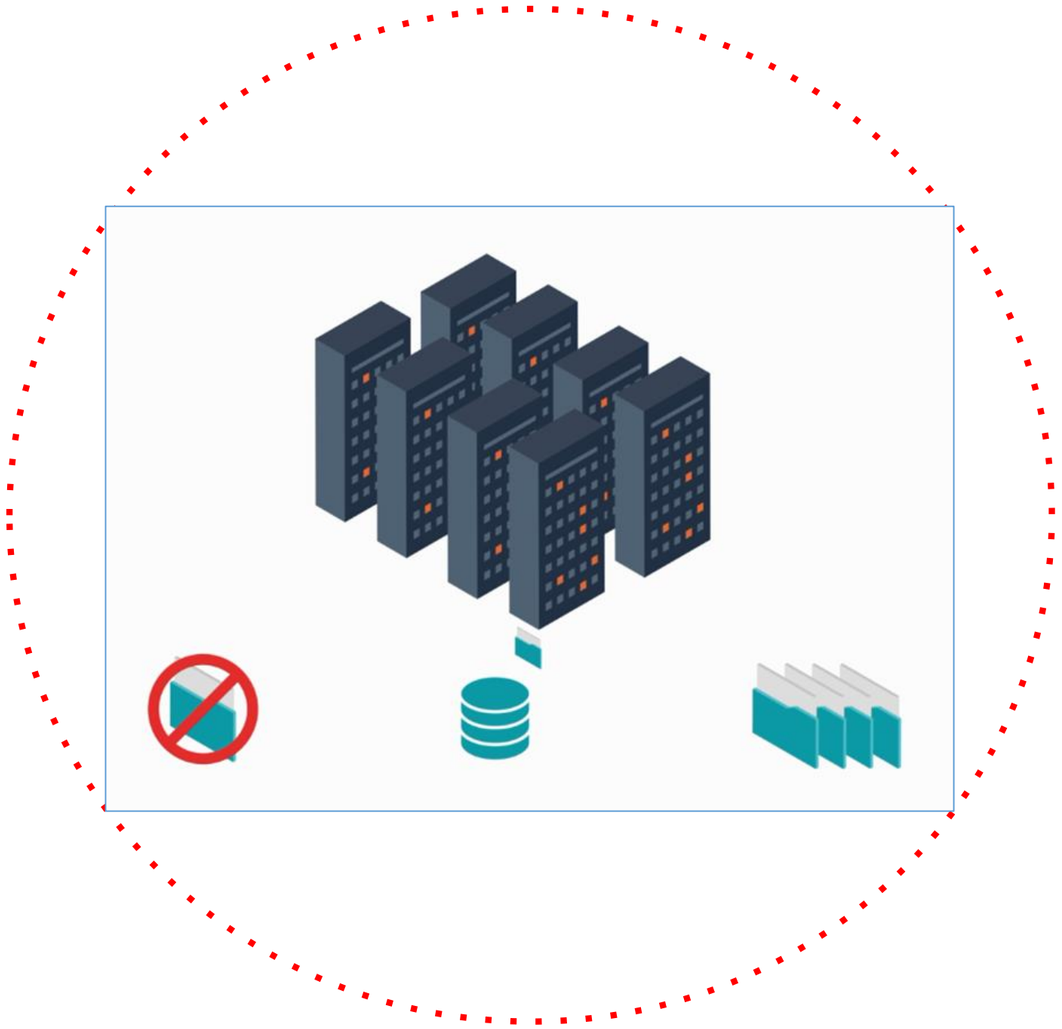
DATA AND
ARTIFICIAL INTELLIGENCE

Distributed Processing in MapReduce

Introduction to MapReduce

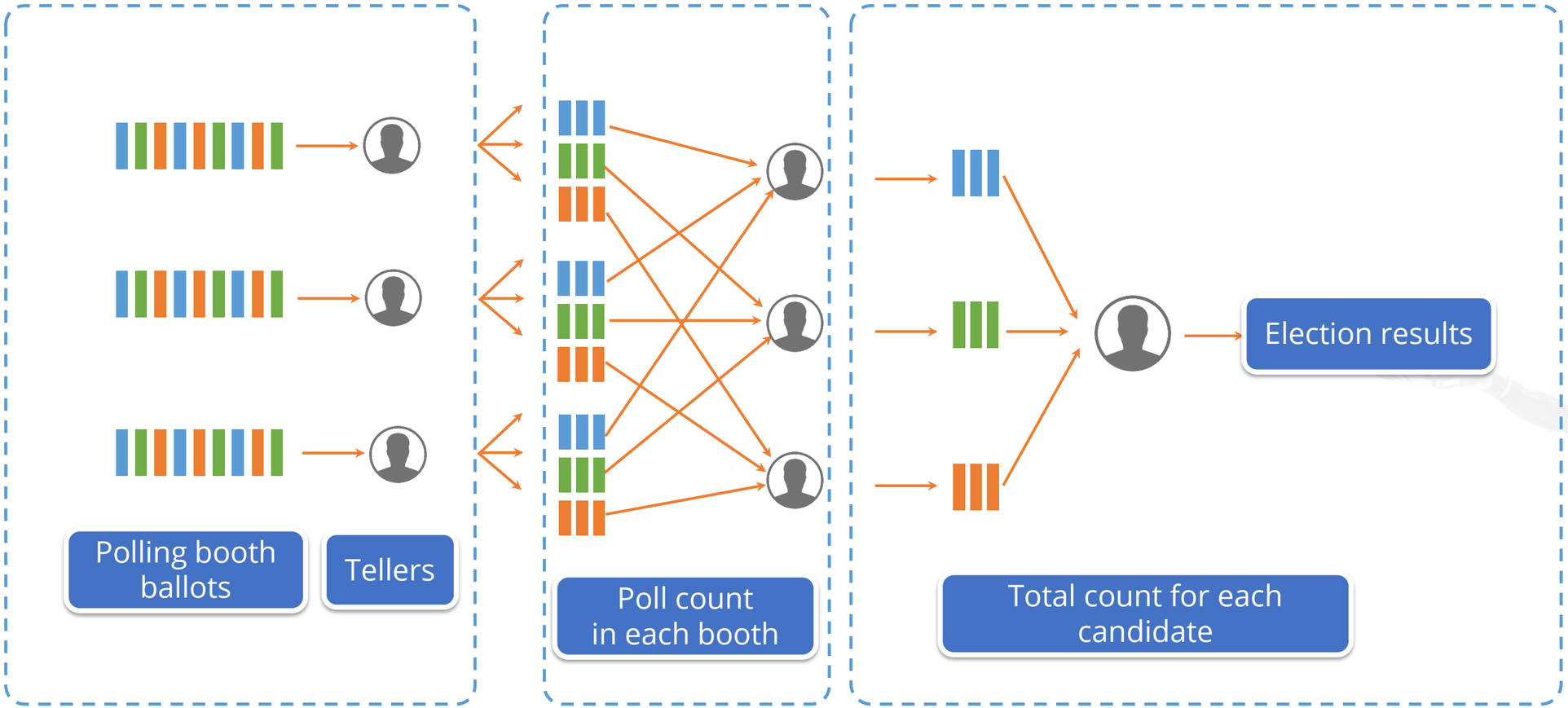
MapReduce is a programming model that simultaneously processes and analyzes huge data sets logically into separate clusters. While **Map** sorts the data, **Reduce** segregates it into logical clusters, thus removing the bad data and retaining the necessary information.

Why MapReduce?



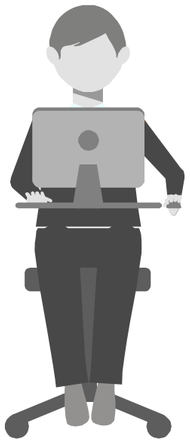
Huge amounts of data were stored in single servers prior to 2004.

MapReduce: Analogy



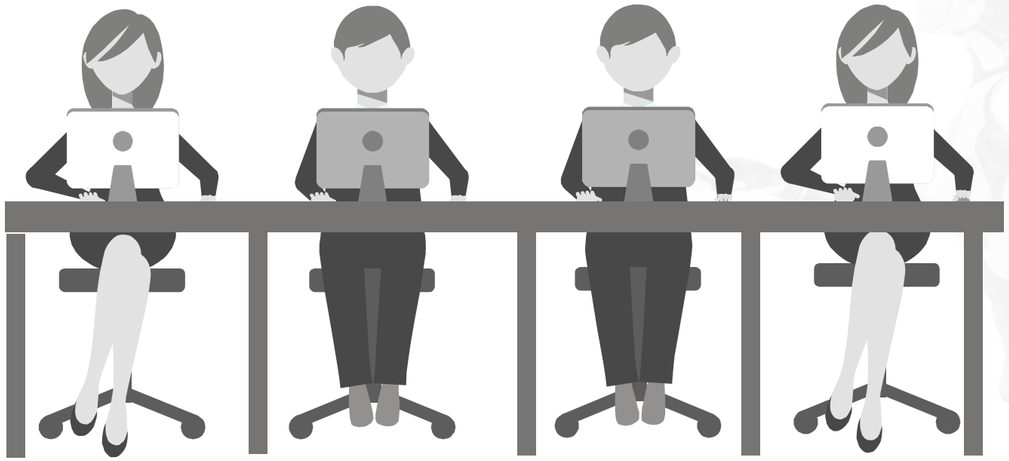
MapReduce: Analogy

One month to receive the election results



Individual Work

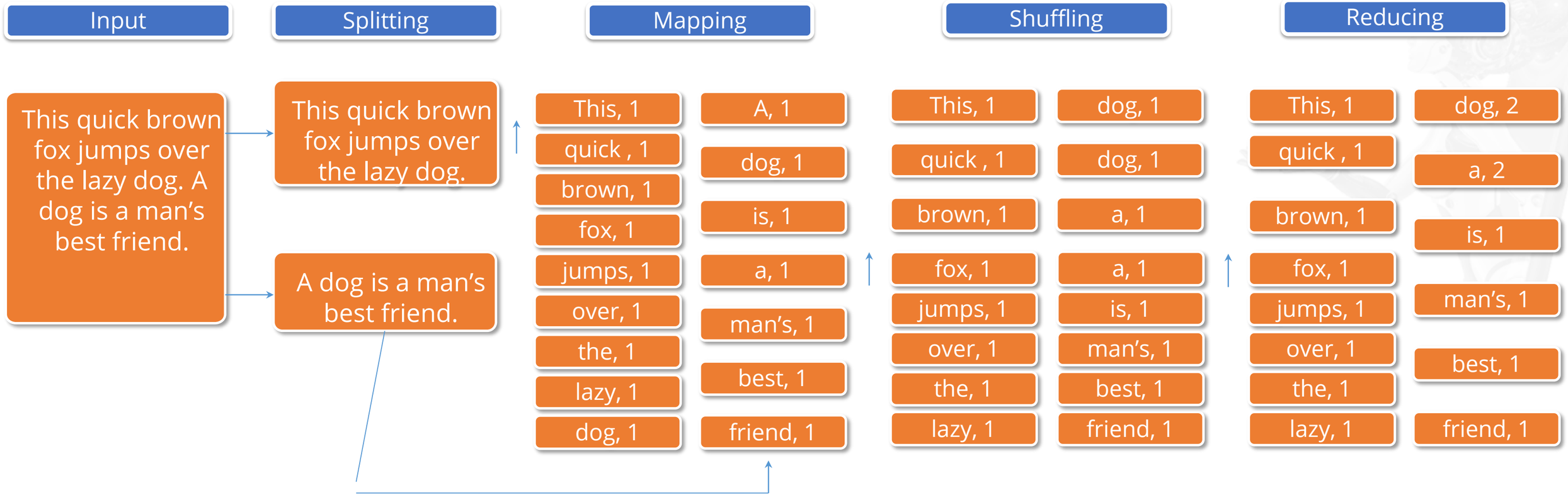
The results are obtained in one or two days



Parallel Work

Word Count Example

MapReduce: Word Count



DATA AND
ARTIFICIAL INTELLIGENCE

Map Execution Phases

Map Execution Phases

Map phase

- Reads assigned input split from HDFS
- Parses input into records as key-value pairs
- Applies map function to each record
- Informs master node of its completion

Partition phase

- Each mapper must determine which reducer will receive each of the outputs
- For any key, the destination partition is the same
- Number of partitions = Number of reducers

Shuffle phase

- Fetches input data from all map tasks for the portion corresponding to the reduce tasks bucket

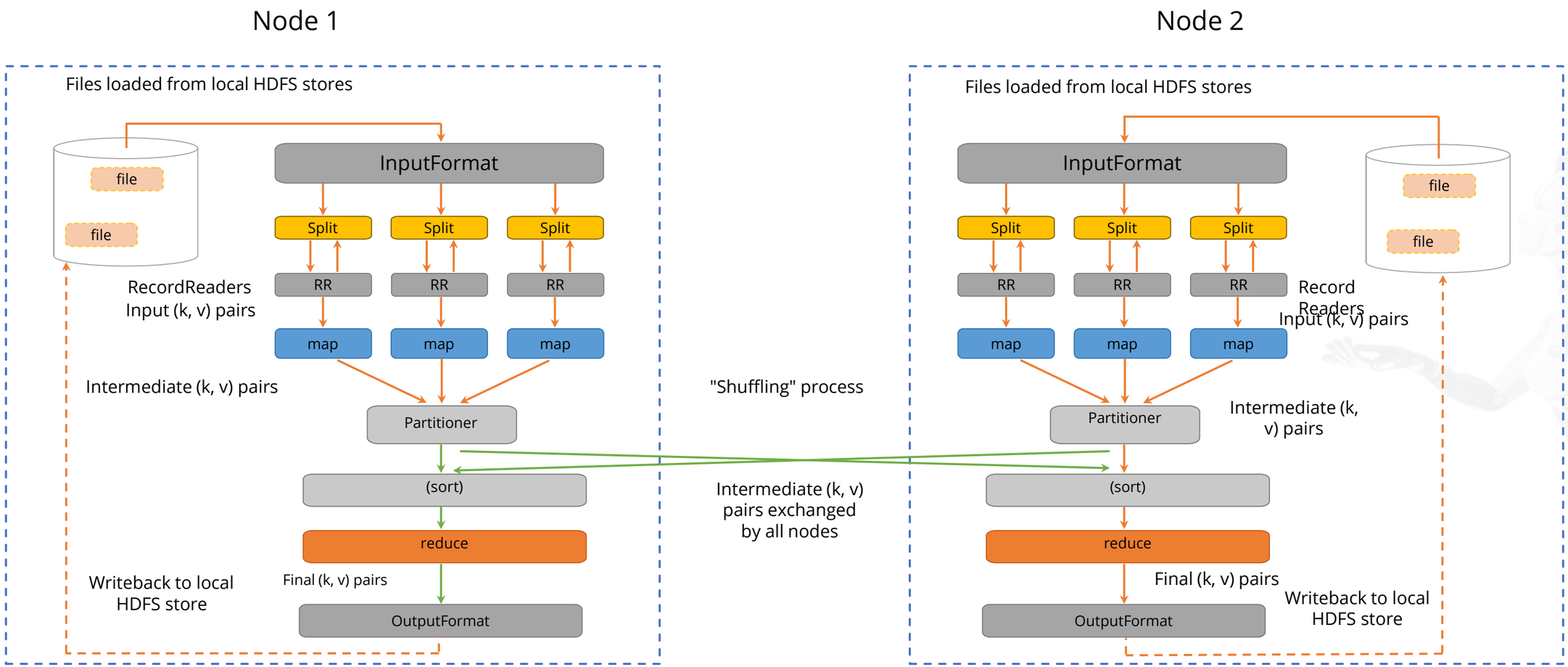
Sort phase

- Merge sorts all map outputs into a single run

Reduce phase

- Applies user-defined reduce function to the merged run

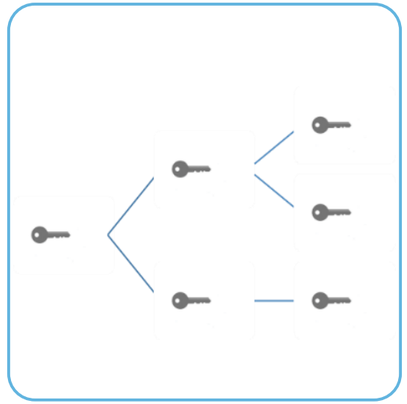
Map Execution: Distributed Two Node Environment



MapReduce Essentials

The job input is specified in key-value pairs:

A user defined map function is applied to each input record to produce a list of intermediate key-value pairs.



A user-defined reduce function is called once for each distinct key in the map output.

MapReduce Essentials

The essentials of each MapReduce phase are as follows:

- The number of reduce tasks can be defined by the users.
- Each reduce task is assigned a set of record groups, that is, intermediate records corresponding to a group of keys.
- For each group, a user-defined reduce function is applied to the recorded values.
- The reduce tasks are read from every map task, and each read returns the record groups for that reduce task.

Reduce phase cannot start until all mappers have finished processing.

DATA AND
ARTIFICIAL INTELLIGENCE

MapReduce Jobs

MapReduce Jobs

A job is a MapReduce program that causes multiple map and reduce functions to run parallelly over the life of the program.

ApplicationMaster and NodeManager functions:

Application Master

- Responsible for the execution of a single application or MapReduce job
- Divides job requests into tasks and assigns them to NodeManagers running on the slave node

NodeManager

- Has many dynamic resource containers
- Executes each active map or reduce task
- Communicates regularly with the Application Master

MapReduce and Associated Tasks

Map process
An initial ingestion and transformation step where initial input records are processed in parallel

Application Master
Keeps track of a MapReduce job

Reduce process
An aggregation or summarization step in which all associated records are processed together

Node Manager
Keeps track of individual map tasks and can run in parallel

Hadoop MapReduce Job Work Interaction

Job submission
Hadoop MapReduce job is submitted by a client in the form of an input file or a number of input split of files containing data.

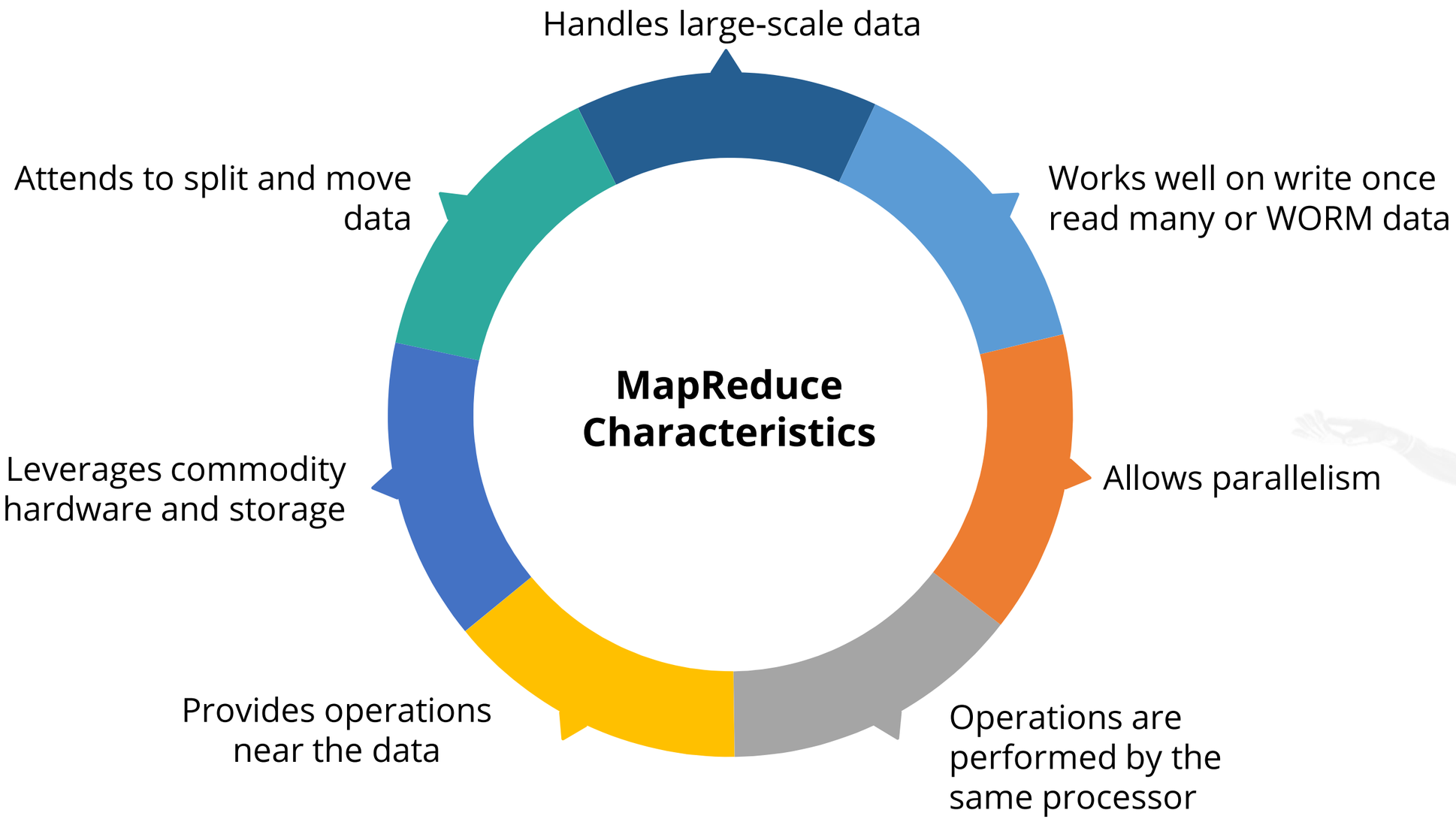
Distribution of input split
MapReduce ApplicationMaster distributes the input split to separate NodeManagers.

Coordination with NodeManager
MapReduce ApplicationMaster coordinates with the NodeManagers.

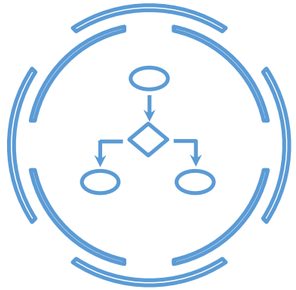
Resubmission of tasks
MapReduce ApplicationMaster resubmits the task(s) to an alternate NodeManager if the DataNode fails.

Success or failure status
ResourceManager gathers the final output and informs the client of the success or failure status.

Characteristics of MapReduce



Real-Time Uses of MapReduce



Algorithms

Sorting

Data mining

Search engine operations

Enterprise analytics

Gaussian analysis

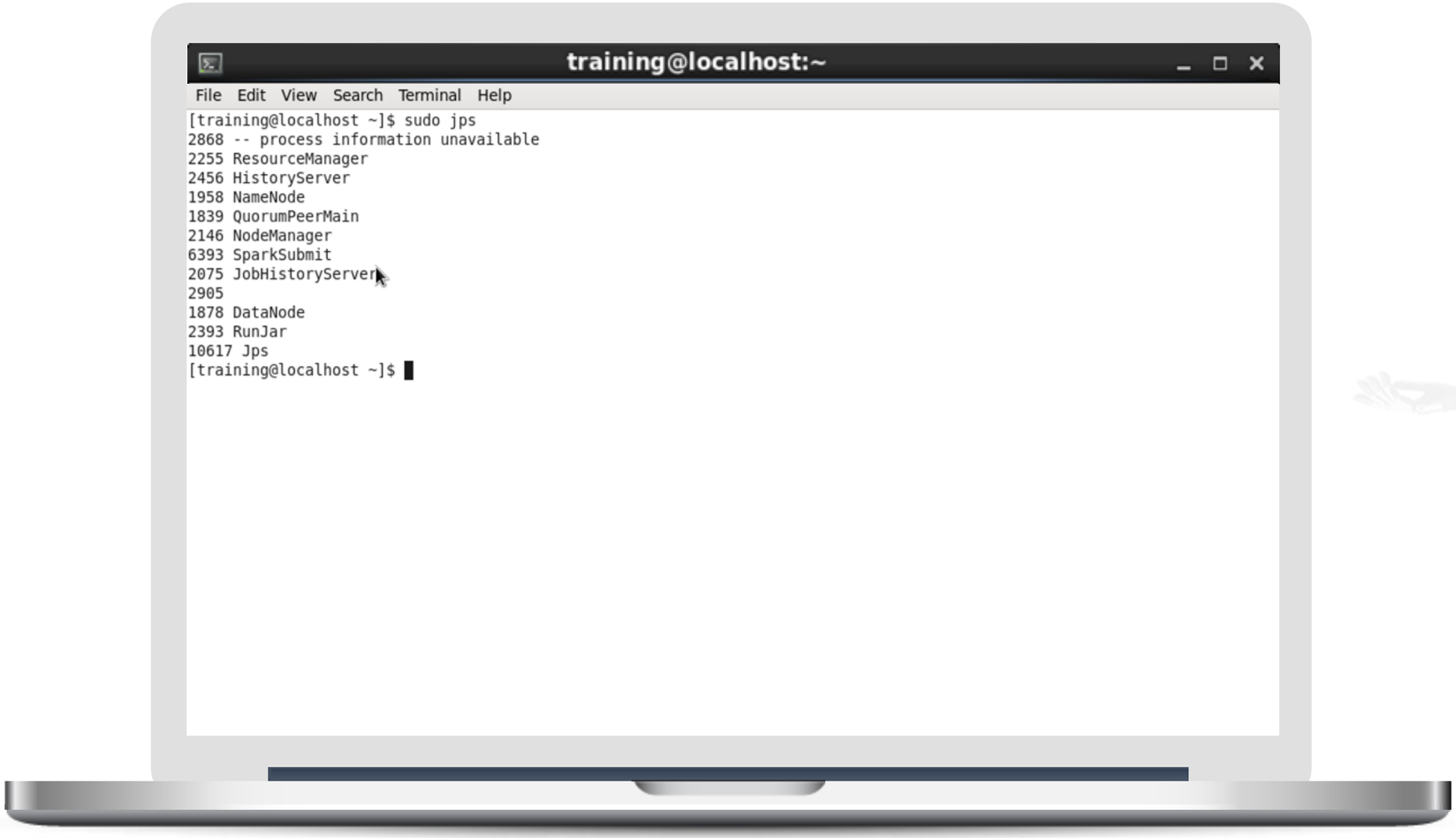
Semantic web 3.0

DATA AND
ARTIFICIAL INTELLIGENCE

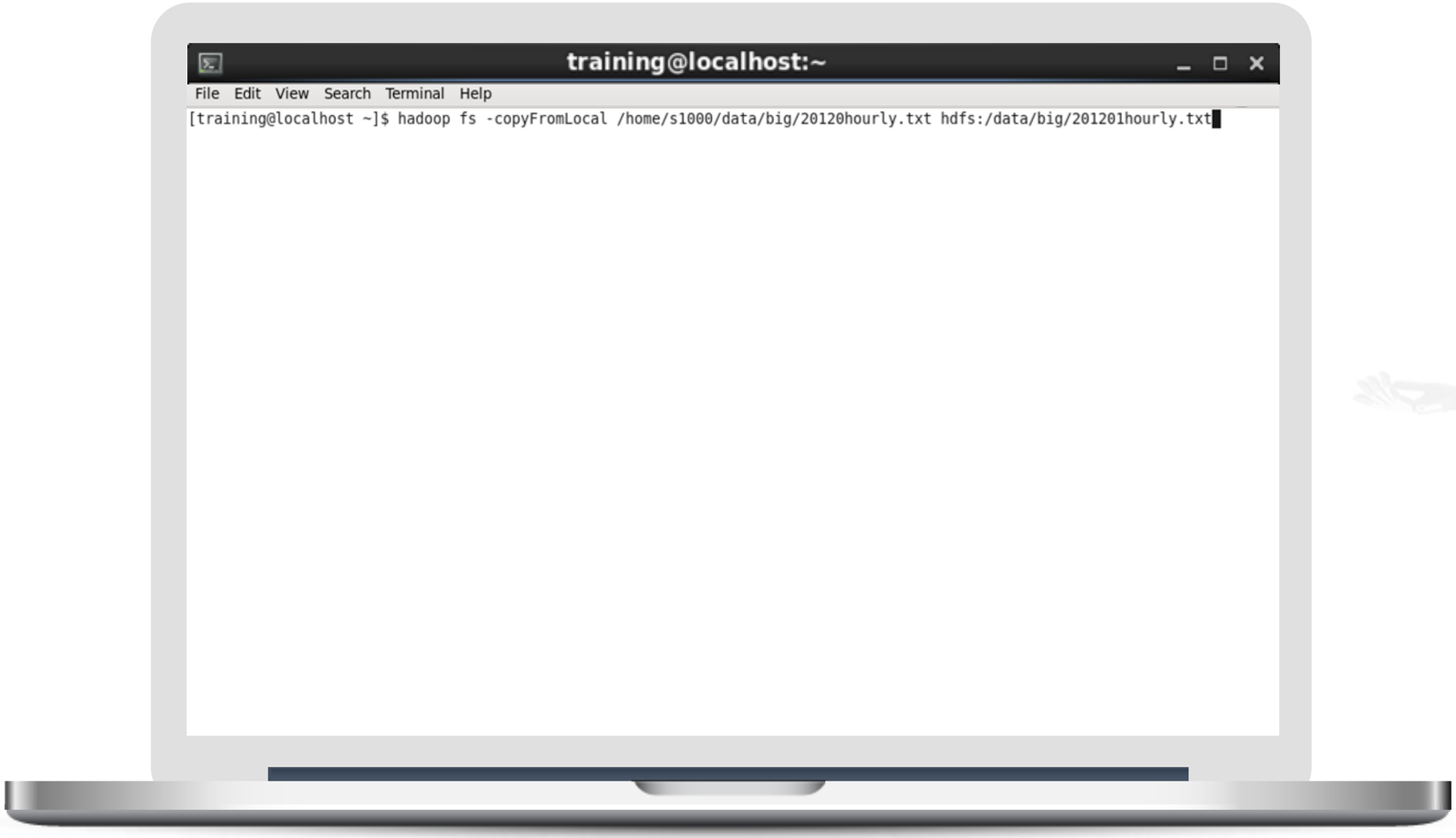
Setting Up the Environment for MapReduce Development

Setting Up the Environment for MapReduce Development

Ensure that all Hadoop services are live and running.

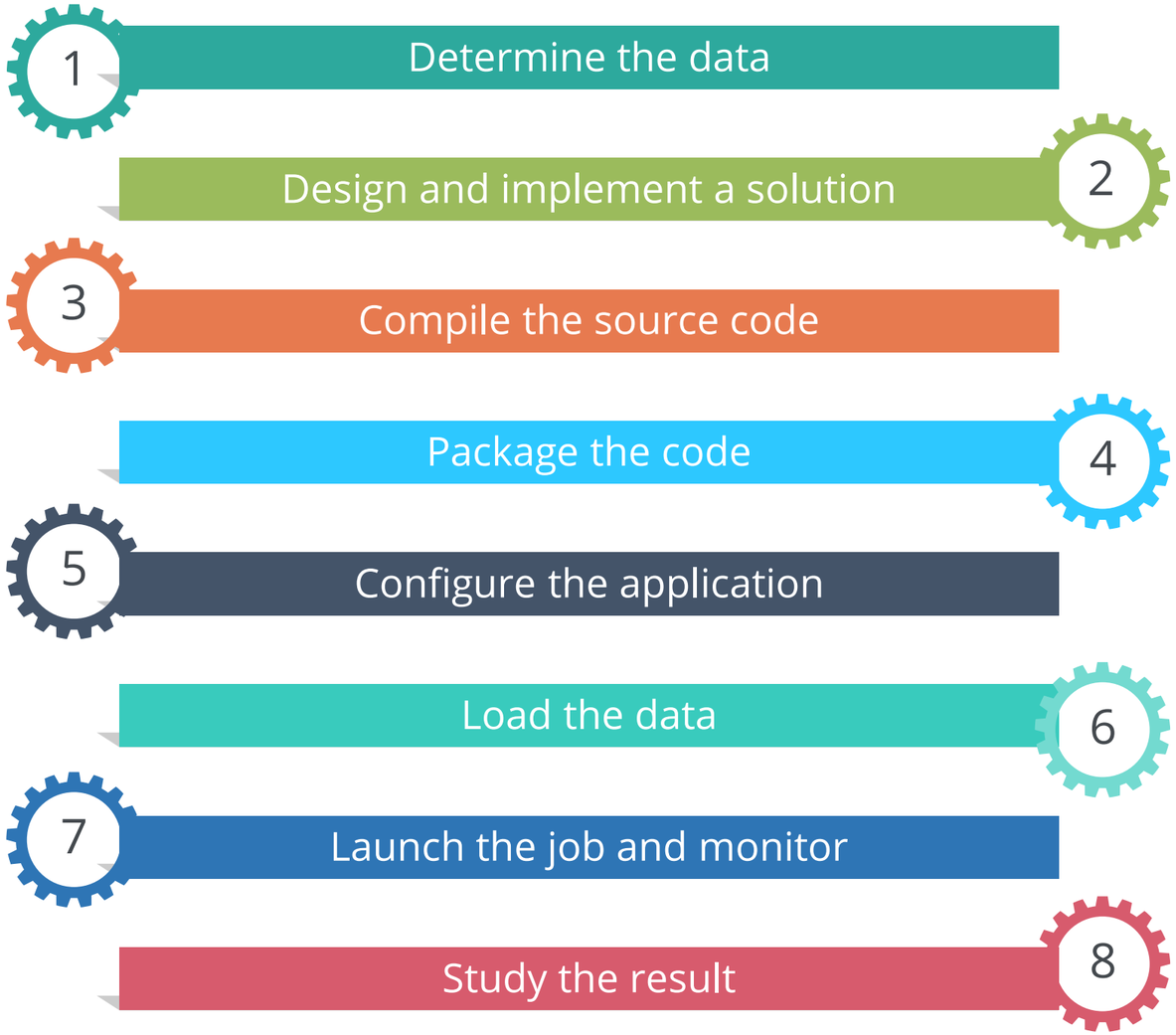


Uploading Big Data and Small Data



Building a MapReduce Program

The steps to build a MapReduce program are as follows:



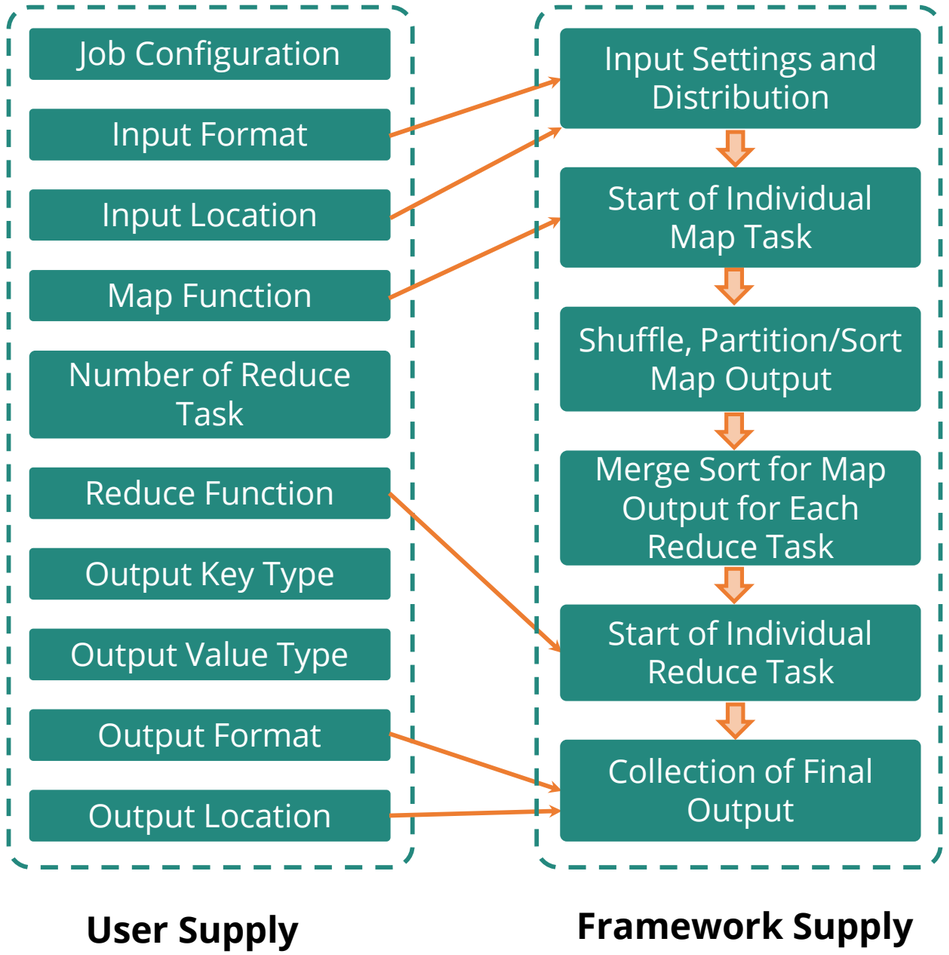
Hadoop MapReduce Requirements

The user or developer is required to set the framework with the following parameters:

- Locations of the job input
- Locations of the job output
- Input format
- Output format
- The class containing the map function
- The class containing the reduce function

Set of Classes

The image shows the set of classes under the user supply and the framework supply.



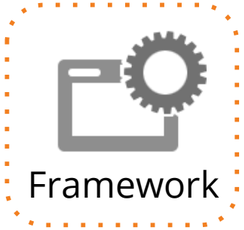
- ResourceManager accepts the input and hands over the job to ApplicationMaster, which divides the job into tasks.
- NodeManager completes the assignment by executing the map task as part of container execution.
- The reducer starts the merging process.
- The output is collected.

MapReduce - Responsibilities

Responsibilities

Developer

- Setting up the job
- Specifying the input location
- Ensuring correct format and location of input



Framework

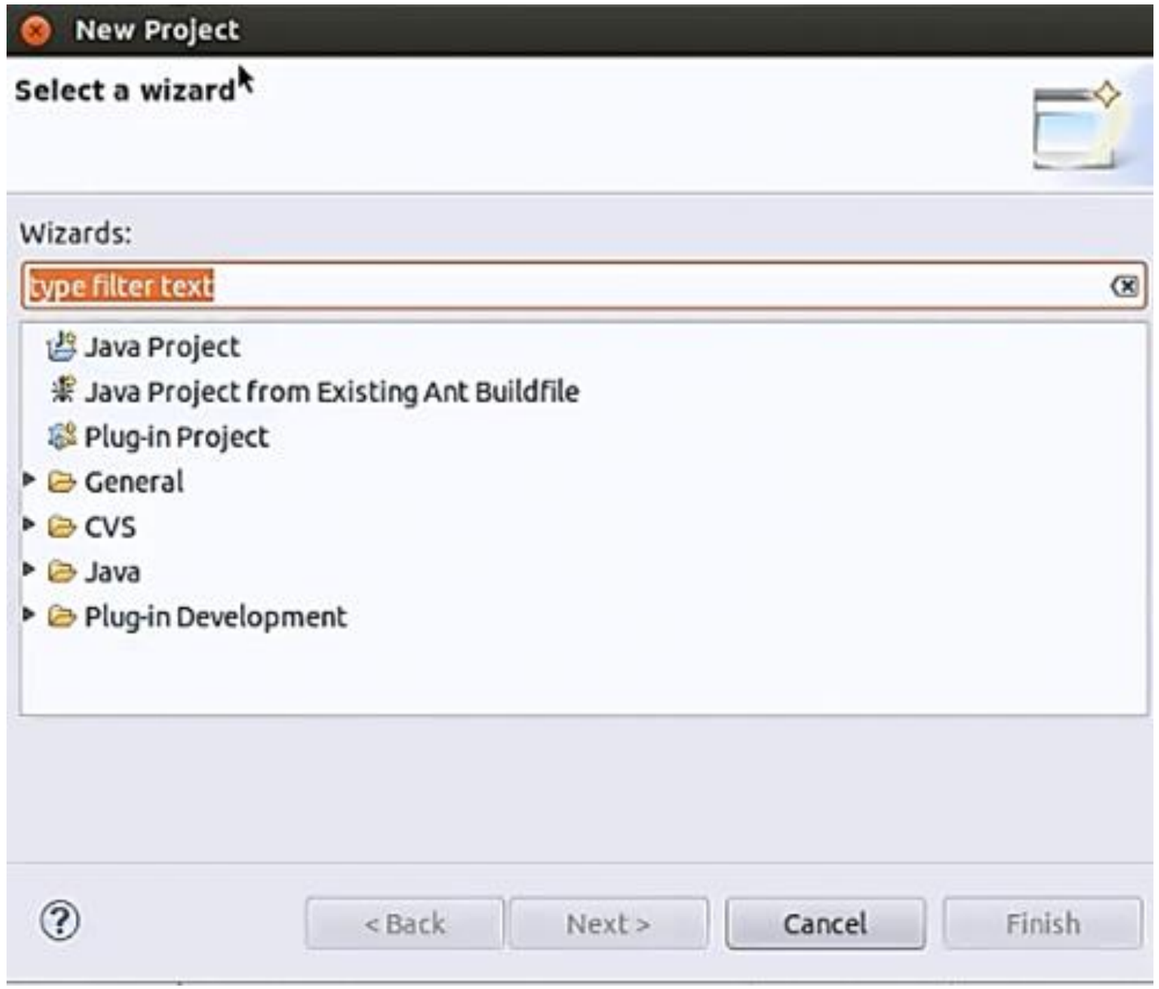
- Distributing the job
- Running the map operation
- Performing shuffling and sorting
- Reducing phases
- Placing the output
- Informing the user of job-completion status

DATA AND
ARTIFICIAL INTELLIGENCE

Creating a New Project

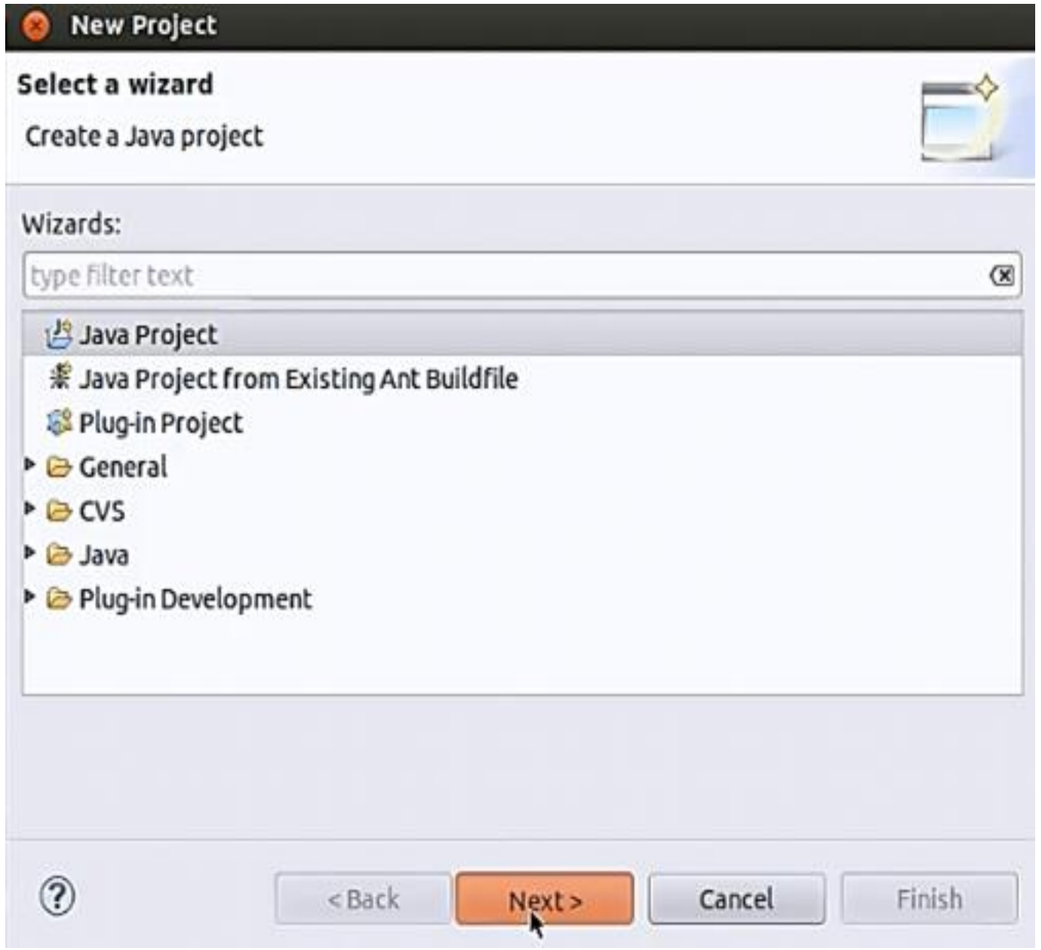
Create a New Project: Step 1

Create a new project and add essential jar files to run MapReduce programs.



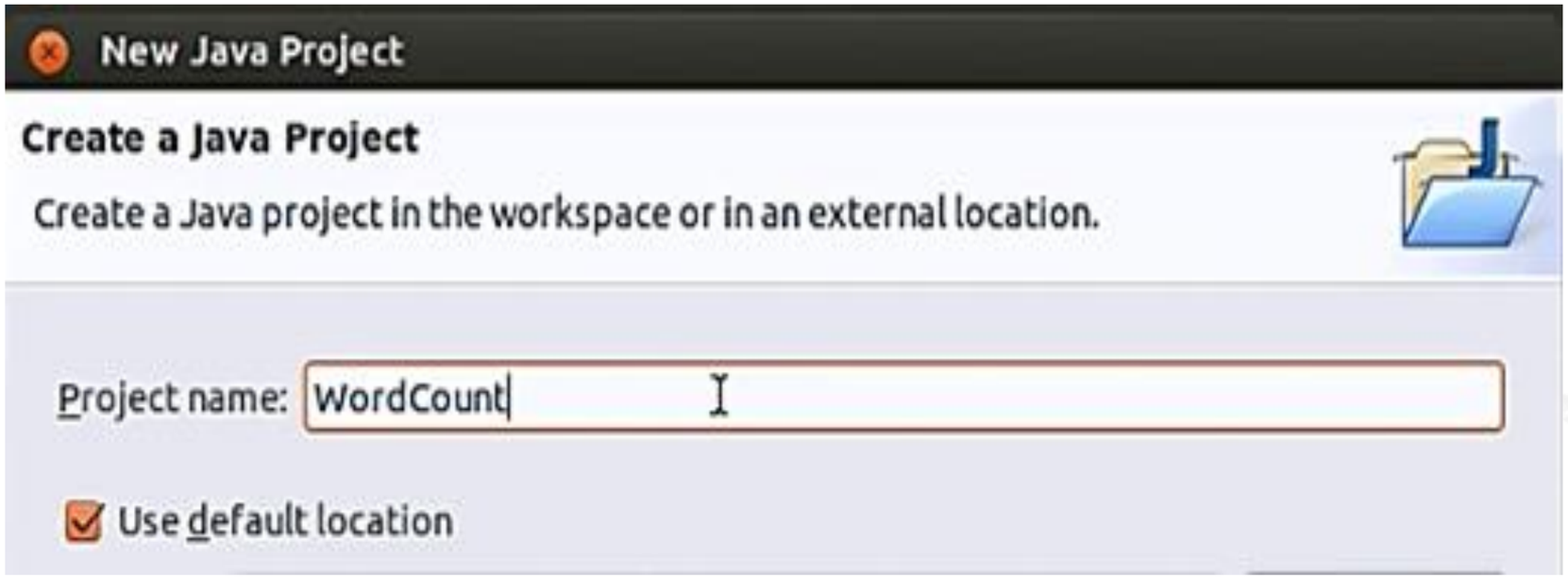
Create a New Project: Step 2

Select Java Project. Then click the Next button to continue.



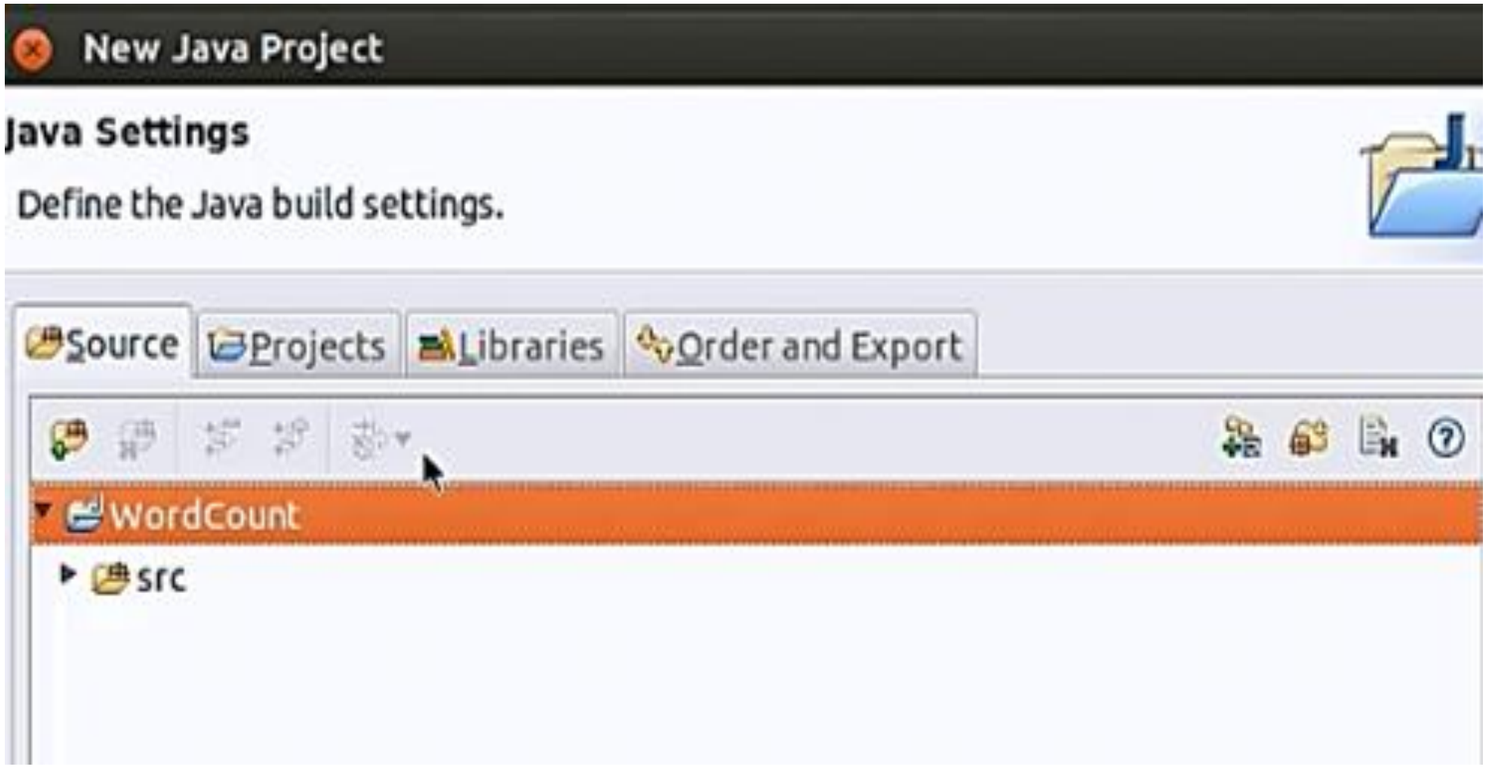
Create a New Project: Step 3

Enter the project name.



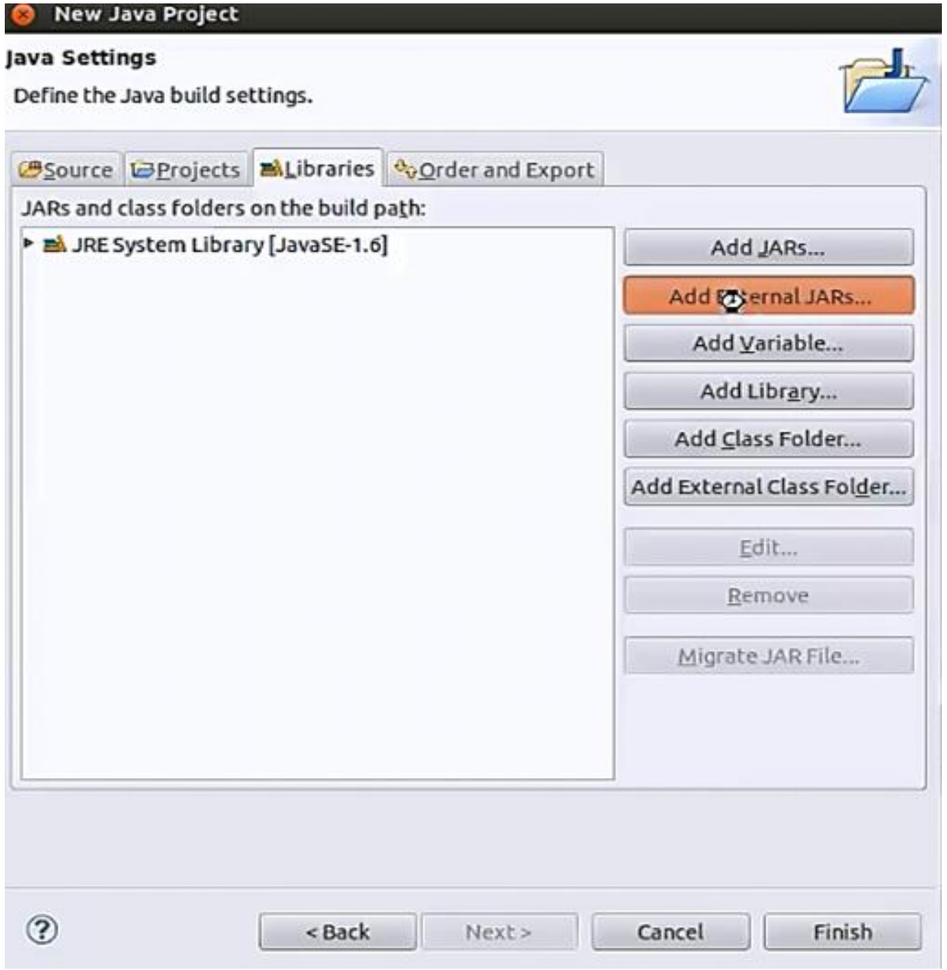
Create a New Project: Step 4

Include jar files from the Hadoop framework to ensure the programs locate the dependencies to one location.



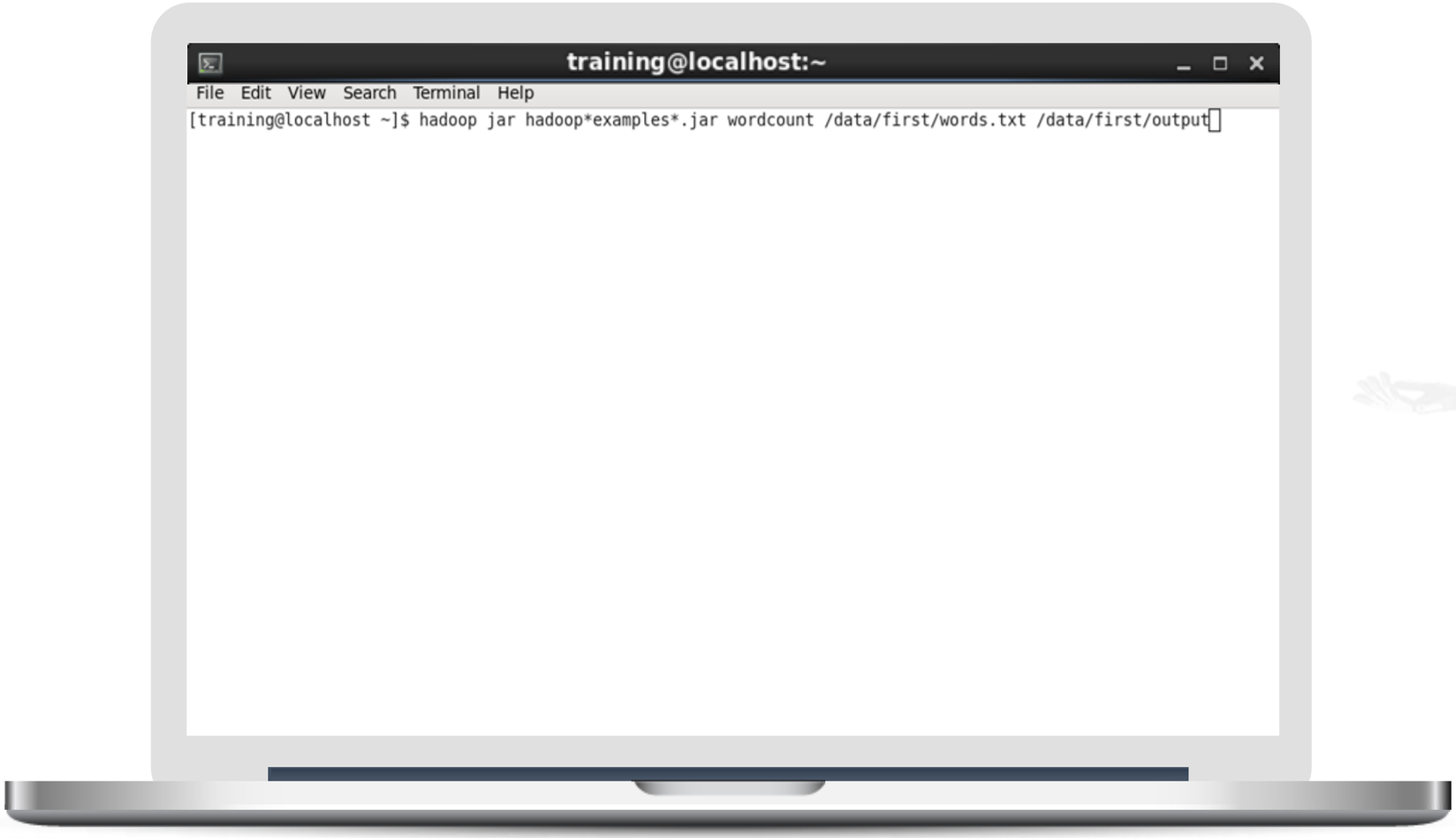
Create a New Project: Step 5

Add the essential jar files.



Checking Hadoop Environment for MapReduce

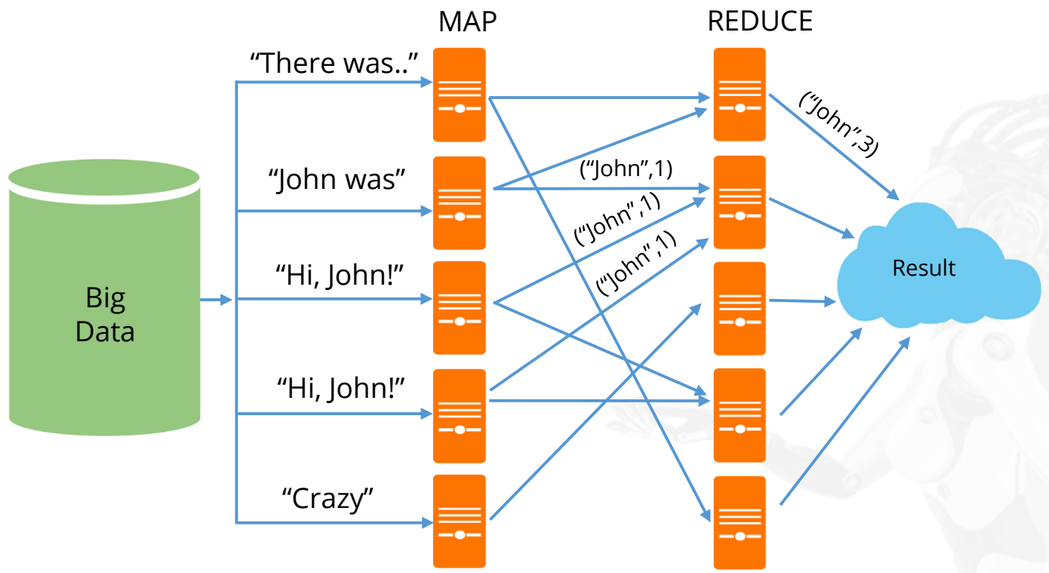
Ensure that the machine setup can perform MapReduce operations.



Advanced MapReduce

Hadoop MapReduce uses data types when it works with user-given mappers and reducers. The data is read from files into mappers and emitted by mappers to reducers. The processed data is sent back by the reducers. Data emitted by reducers goes into output files. At every step, data is stored in Java objects.

Writable data types: In the Hadoop environment, objects that can be put to or received from files and across the network must obey the Writable interface.



Interfaces

The interfaces in Hadoop are as follows:

- Writable
- Writable Comparable

A writable interface allows Hadoop to read and write the data in a serialized form for transmission.

```
interface Writable {
 public void readFields(DataInput in);
 public void write(DataOutput out);
}
```

Interfaces

The interfaces in Hadoop are as follows:

Writable

Writable Comparable

A WritableComparable interface extends the Writable interface so that the data can be used as a key and not as a value.

```
int compareTo(Object what)
int hashCode()
```

DATA AND
ARTIFICIAL INTELLIGENCE

Data Types in Hadoop

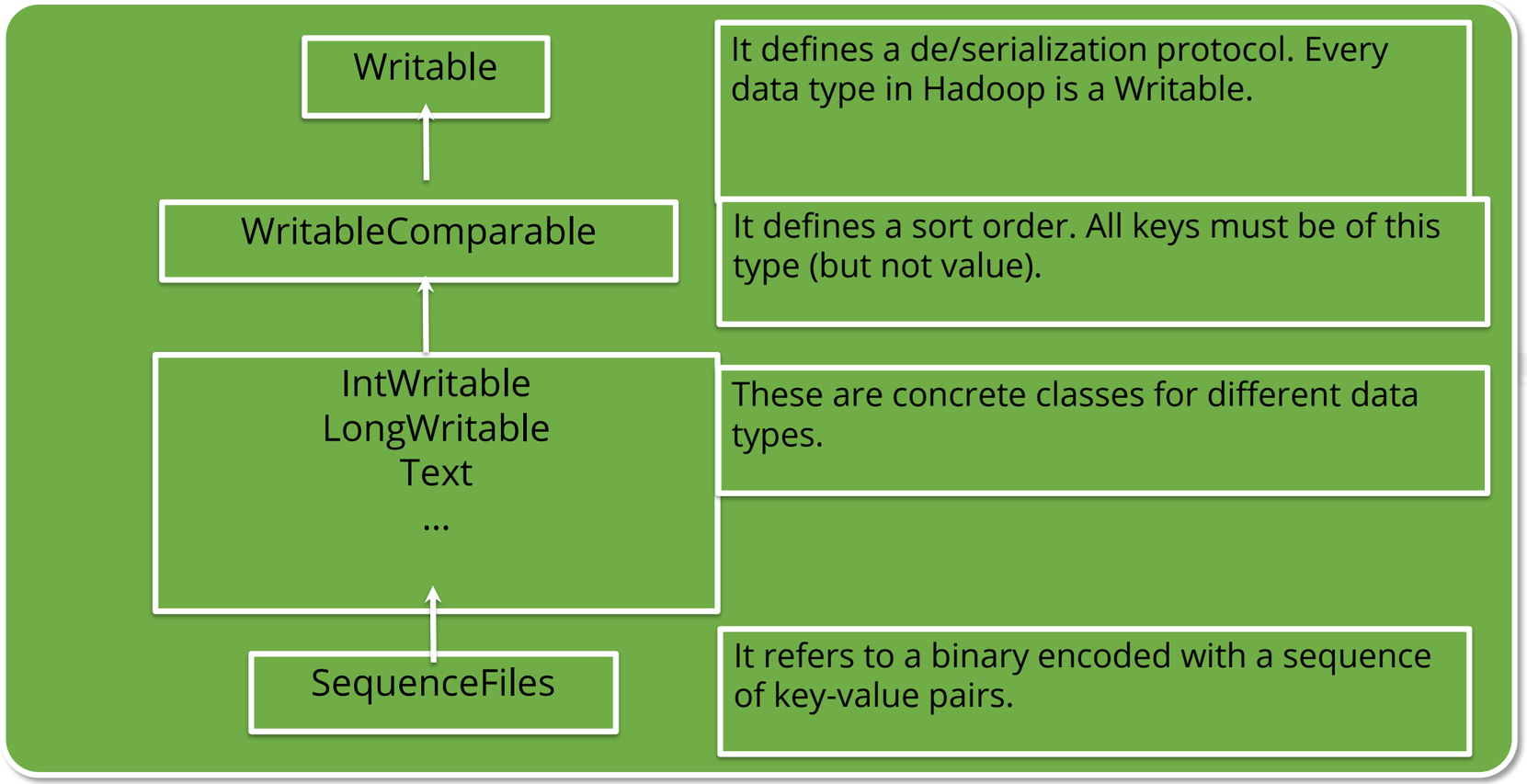
Data Types in Hadoop

The table lists a few important data types and their functions:

Data types	Functions
Text	Stores String data
IntWritable	Stores Integer data
LongWritable	Stores Long data
FloatWritable	Stores Float data
DoubleWritable	Stores Double data
BooleanWritable	Stores Boolean data
ByteWritable	Stores Byte data
NullWritable	Placeholder when value is not needed

Data Types in Hadoop

A sample data type related to the Writable interface is displayed here:



InputFormats in MapReduce

MapReduce can specify how its input is to be read by defining an InputFormat. The table lists some of the classes of InputFormats provided by the Hadoop framework:

InputFormat classes	Description
KeyValueTextInputFormat	One key-value pair per line
TextInputFormat	Key is the line number, and value is the line
NLineInputFormat	Similar to TextInputFormat, but the difference is that there are N number of lines that make an input split
MultiFileInputFormat	Input format that aggregates multiple files into one split
SequenceFileInputFormat	The input file is a Hadoop sequence file which contains a serialized key-value pair.

OutputFormats in MapReduce

The table lists some of the key classes of OutputFormats provided by the Hadoop framework:

OutputFormat classes	Description
TextOutputFormat	It is the default OutputFormat and writes records as lines of text. Each key-value pair is separated by a TAB character. This can be customized by using the mapred.textoutputformat.separator property. The corresponding InputFormat is KeyValueTextInputFormat.
SequenceFileOutputFormat	It writes sequence files to save the output. It is compact and compressed.
SequenceFileAsBinaryOutputFormat	It writes key and value in raw binary format into a sequential file container.
MapFileOutputFormat	It writes MapFiles as the output. The keys in a MapFile must be added in an order, and the reducer will emit keys in the sorted order.
MultipleTextOutputFormat	It writes data to multiple files whose names are derived from output keys and values.
MultipleSequenceFileOutputFormat	It creates output in multiple files in a compressed form.

Distributed Cache

Helps to boost efficiency when a map or a reduce task needs access to common data.

Allows a cluster node to read the imported files from its local file system instead of retrieving the files from other cluster nodes.

Allows both single files and archives (such as zip and tar.gz).

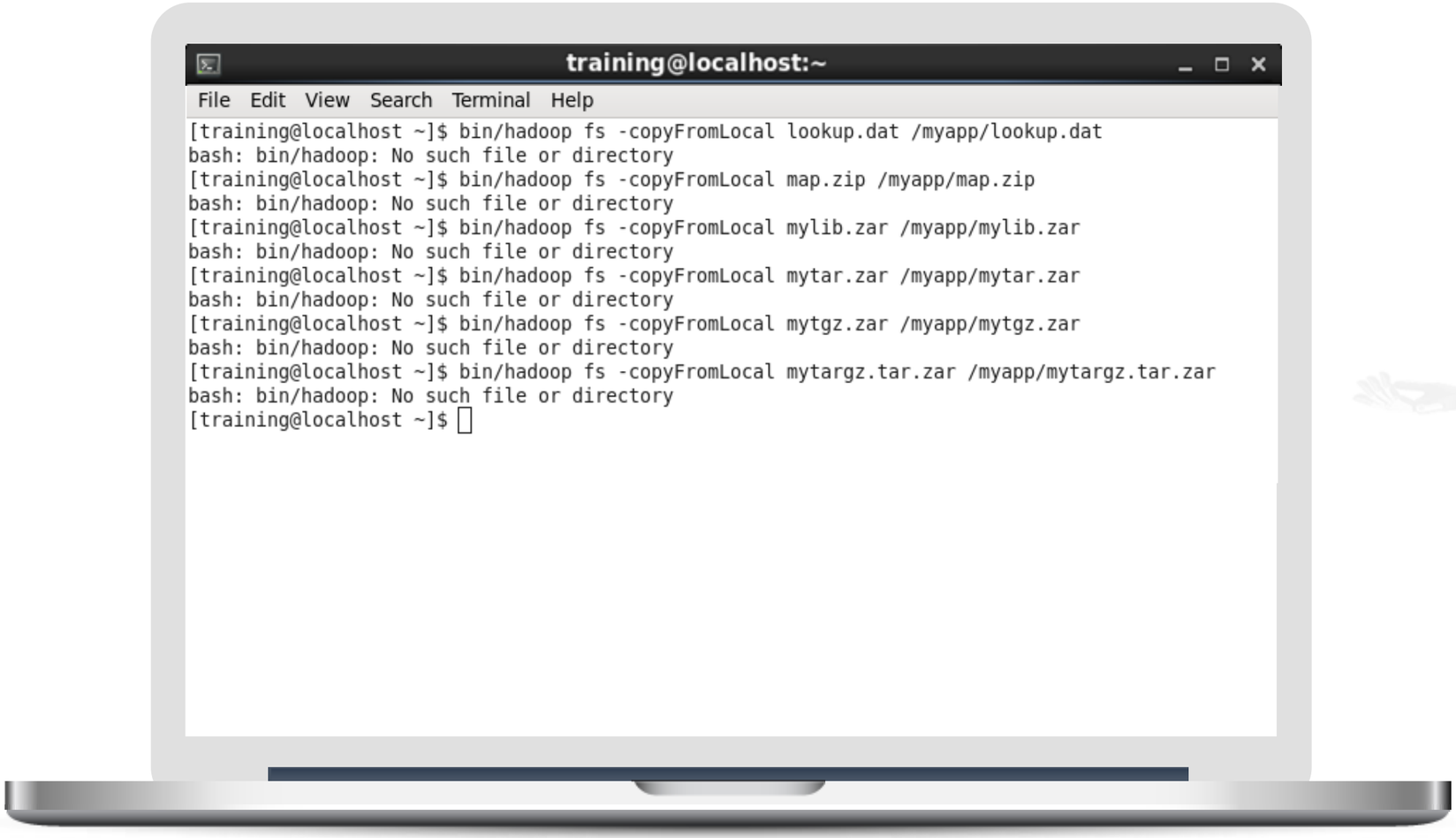
Copies files only to slave nodes. If there are no slave nodes in the cluster, distributed cache copies the files to the master node.

Allows access to the cached files from mapper or reducer applications to make sure that the current working directory (./) is added into the application path.

Allows one to reference the cached files as though they are present in the current working directory.

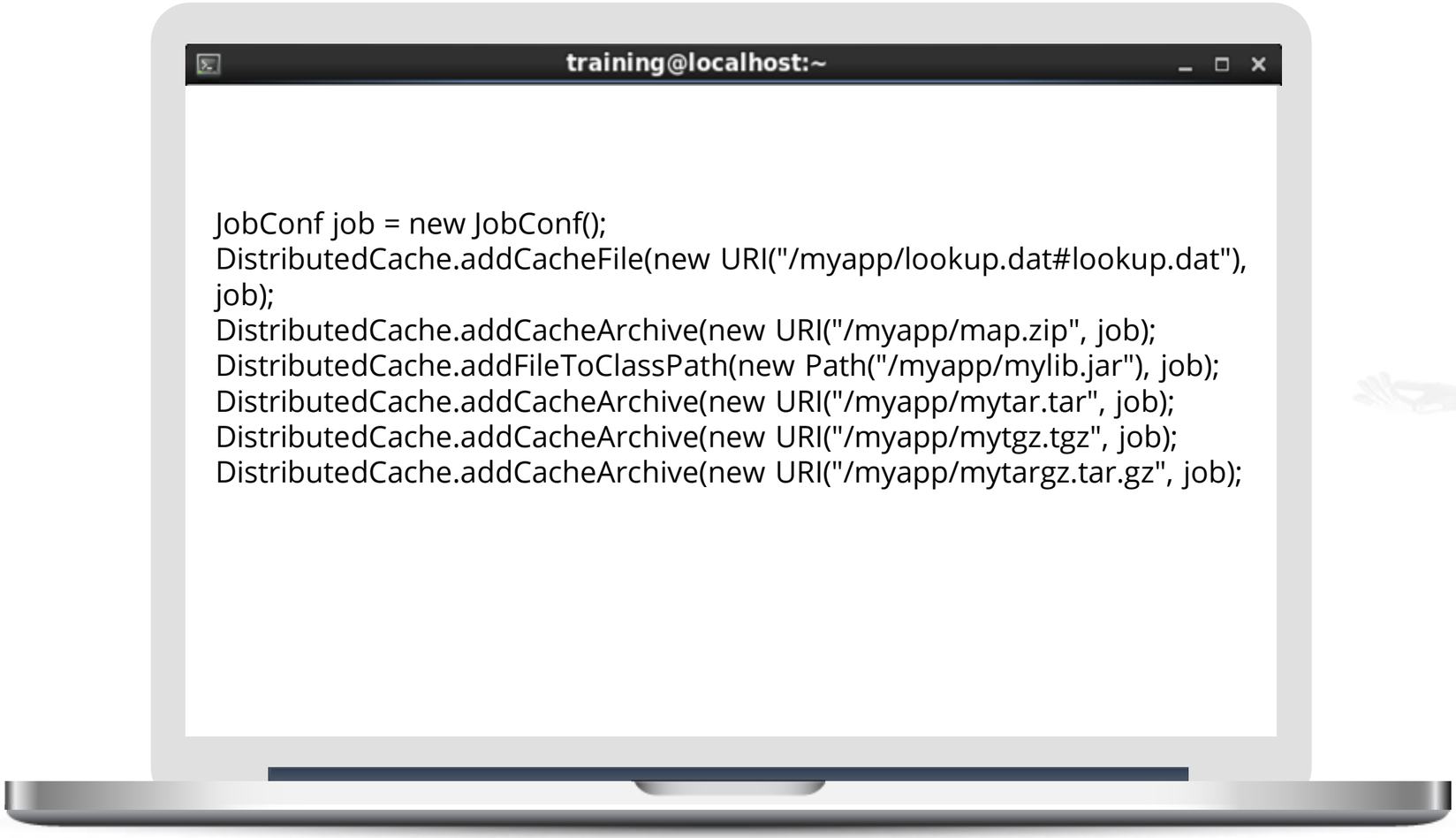
Using Distributed Cache - Step 1

Set up the cache by copying the requisite files to the FileSystem.



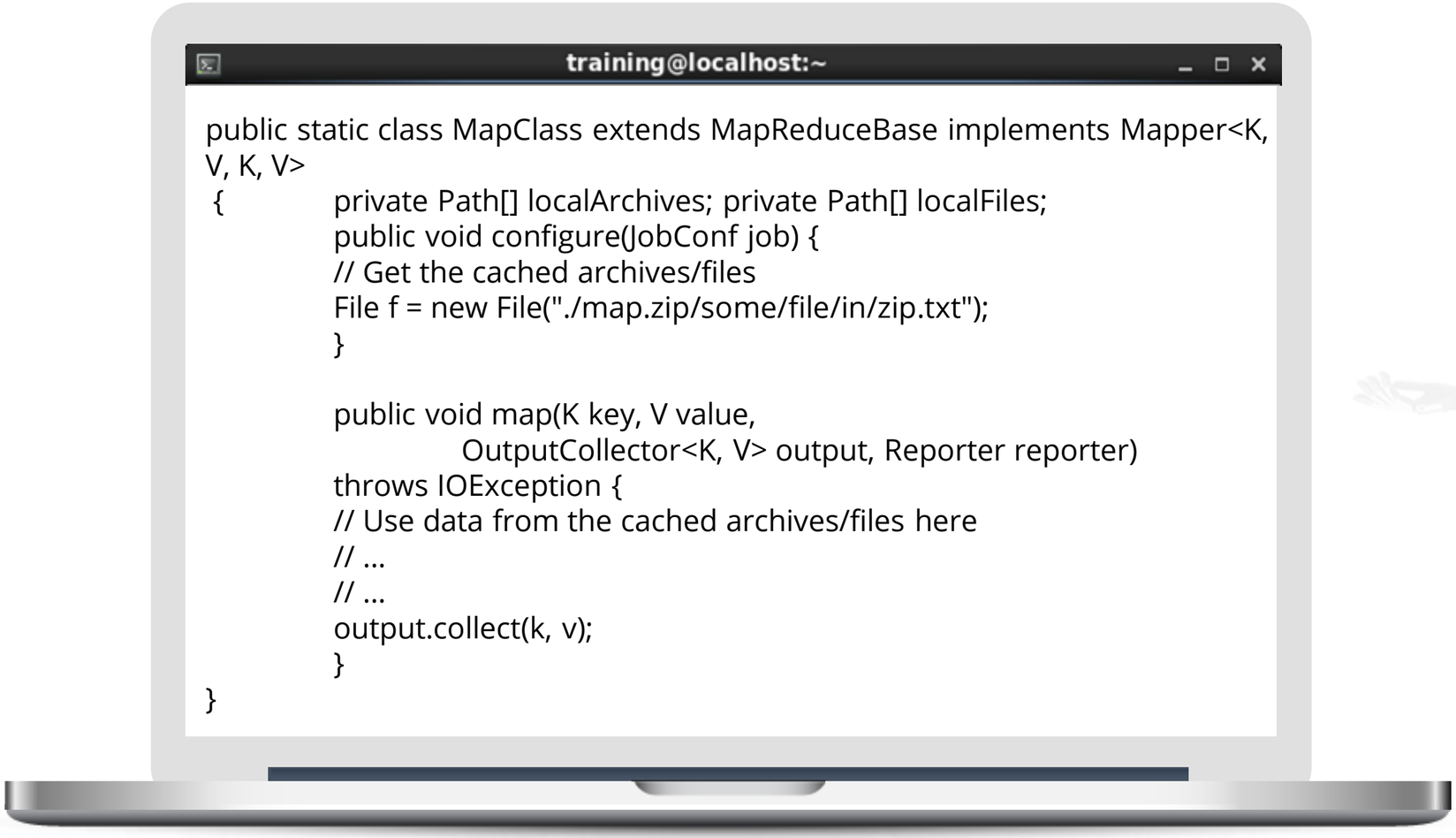
Using Distributed Cache - Step 2

Set up the application's JobConf as shown below.



Using Distributed Cache - Step 3

Use the cached files in the mapper or reducer.



DATA AND
ARTIFICIAL INTELLIGENCE

Joins in MapReduce

Joins in MapReduce

Joins are relational constructs to combine relations. In MapReduce, joins are used to combine two or more datasets. A join is performed either in the map phase or in the reduce phase by taking advantage of the MapReduce Sort-Merge architecture.

Cartesian product

It is a map-side join where every single record is paired up from another dataset.

Reduce side join

It is used for joining two or more large datasets by the same foreign key using any kind of join operation.

Composite join

It is a map-side join on very large formatted input datasets sorted and partitioned by a foreign key.

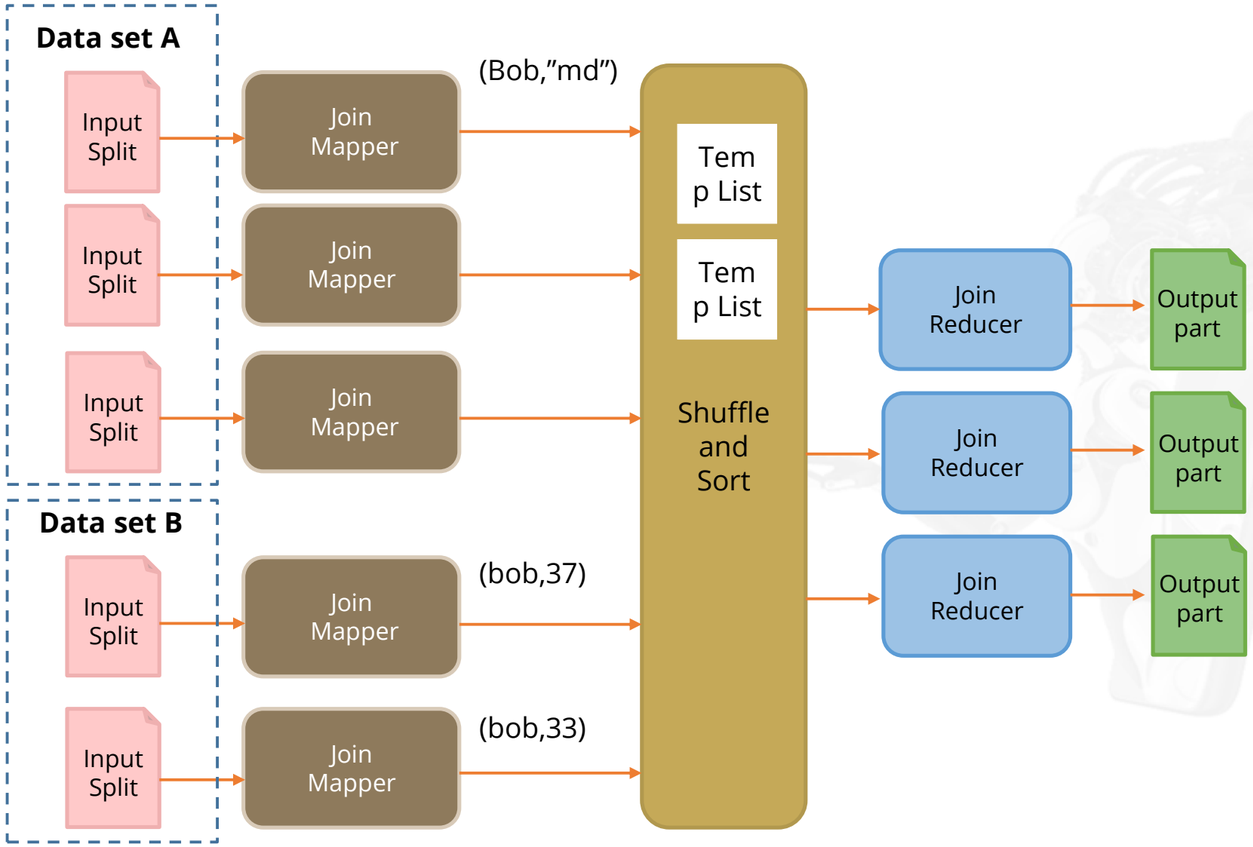
Replicated join

It is a map-side join that works in situations where one of the datasets is small enough to cache.

Reduce Side Join

A reduce side join works in the following ways:

- The mapper prepares for join operations:
  - It takes each input record from every dataset.
  - It emits a foreign key-record pair.
- The reducer performs join operation:
  - It stores the values of each input group in temporary lists.
  - The temporary lists are then iterated over, and the records from both sets are joined.



Reduce Side Join

A reduce side join should be used in the following conditions:

When multiple large datasets are joined by a foreign key

When flexibility is needed to execute any join operation

When a large amount of network bandwidth is available

When there is no limitation on the size of datasets

SQL analogy

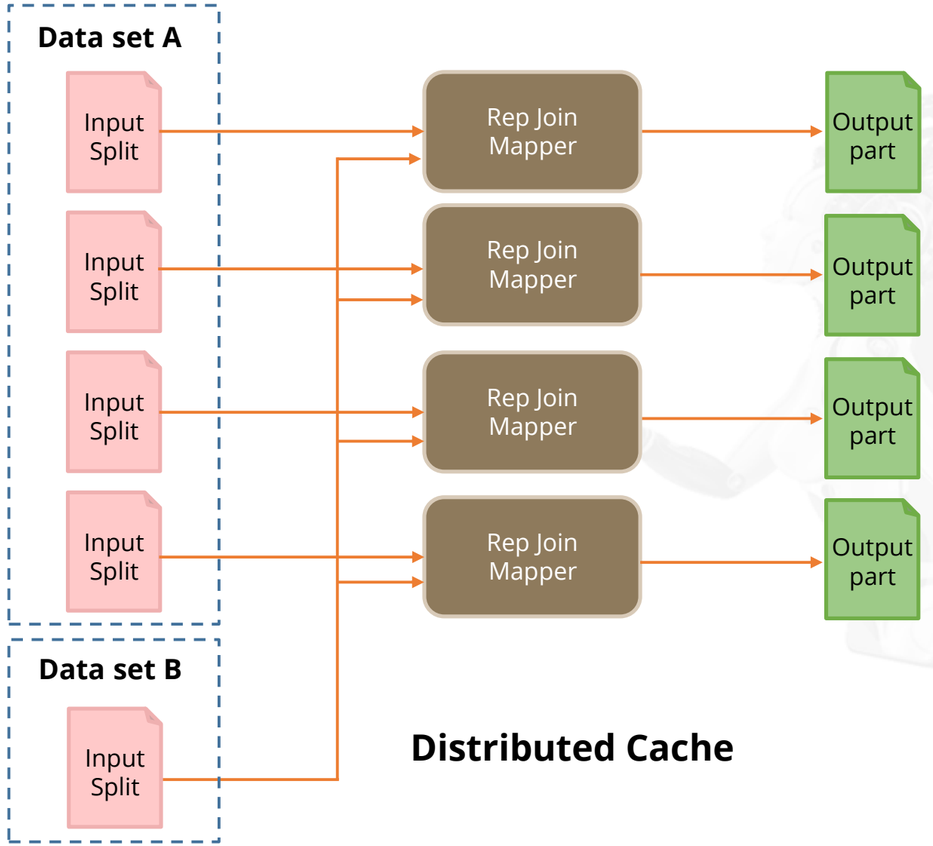
SELECT *users.ID, users.****Location****, comments.upVotes*
FROM *users*
*[****INNER****|****LEFT****|****RIGHT****]* ***JOIN*** *comments*
ON *users.ID=comments.UserID*

Output of a reduce side join:
The number of part files equals the number of reduce tasks.

Replicated Join

A replicated join is a map-only pattern. It works in the following ways:

- It reads all files from the distributed cache and stores them in in-memory lookup tables.
- The mapper processes each record and joins it with the data stored in memory.
- There is no data shuffled to the Reduce phase.
- The mapper gives the final output part.



Replicated Join

A replicated join should be used in the following conditions:

When all datasets except for the largest one can fit into the main memory of each map task that is limited by Java Virtual Machine (JVM) heap size

When there is a need for an inner join or a left outer join, with the large input dataset being the "left" part of the operation

SQL analogy

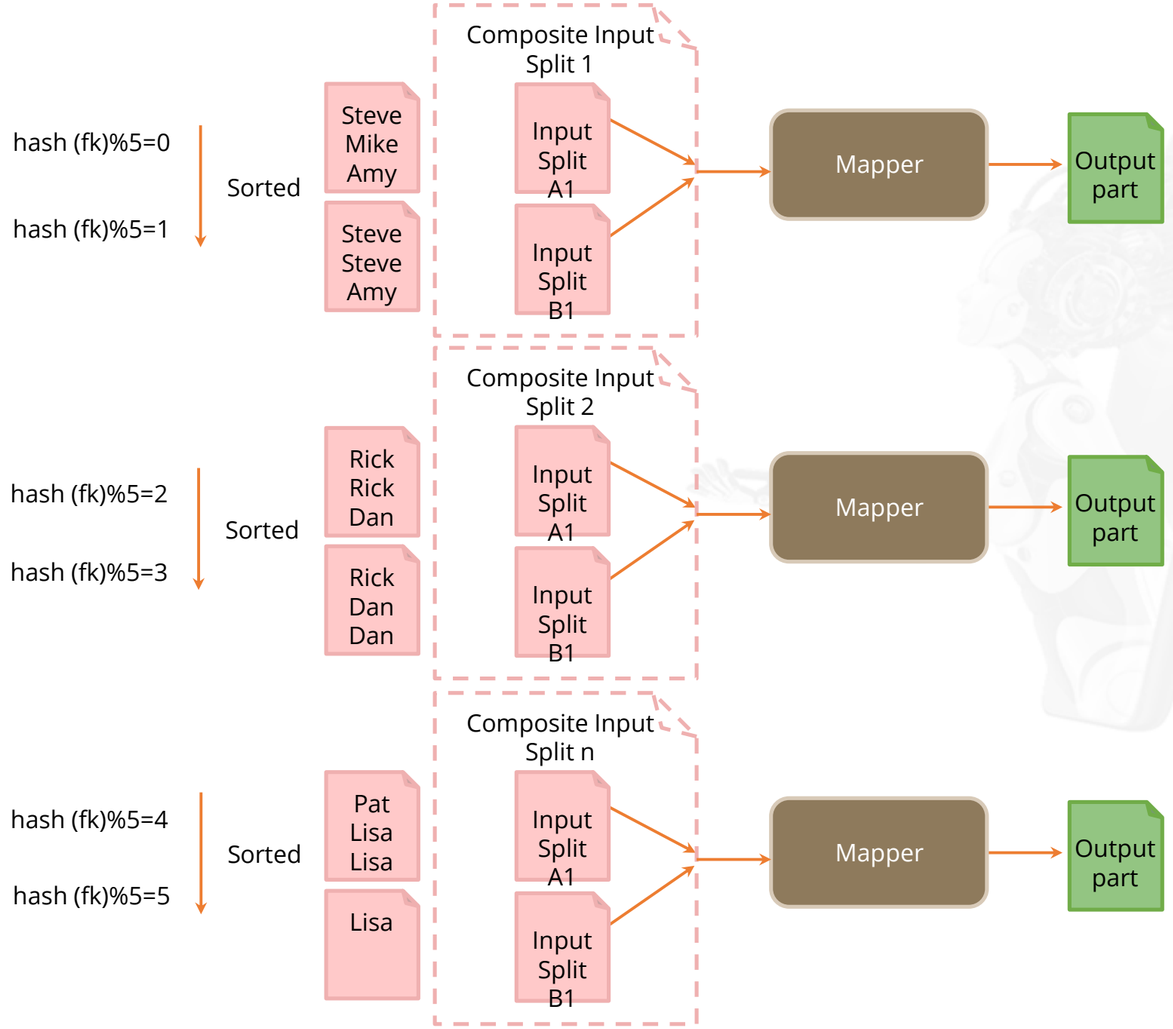
SELECT *users.ID, users.****Location****, comments.upVotes*
FROM *users*
*[****INNER****|****LEFT****|****RIGHT****]* ***JOIN*** *comments*
ON *users.ID=comments.UserID*

Output of a replicated join:
The number of part files equals the number of map tasks.

Composite Join

A composite join is a map-only pattern working in the following ways:

- All datasets are divided into the same number of partitions.
- Each partition of dataset is sorted by a foreign key, and all the foreign keys reside in the associated partition of each dataset.
- Two values are retrieved from the input tuple associated with each dataset; they are based on the foreign key and the output to the file system.



Composite Join

A composite join should be used in the following conditions:

When all datasets are sufficiently large

When there is a need for an inner join or a full outer join

SQL analogy

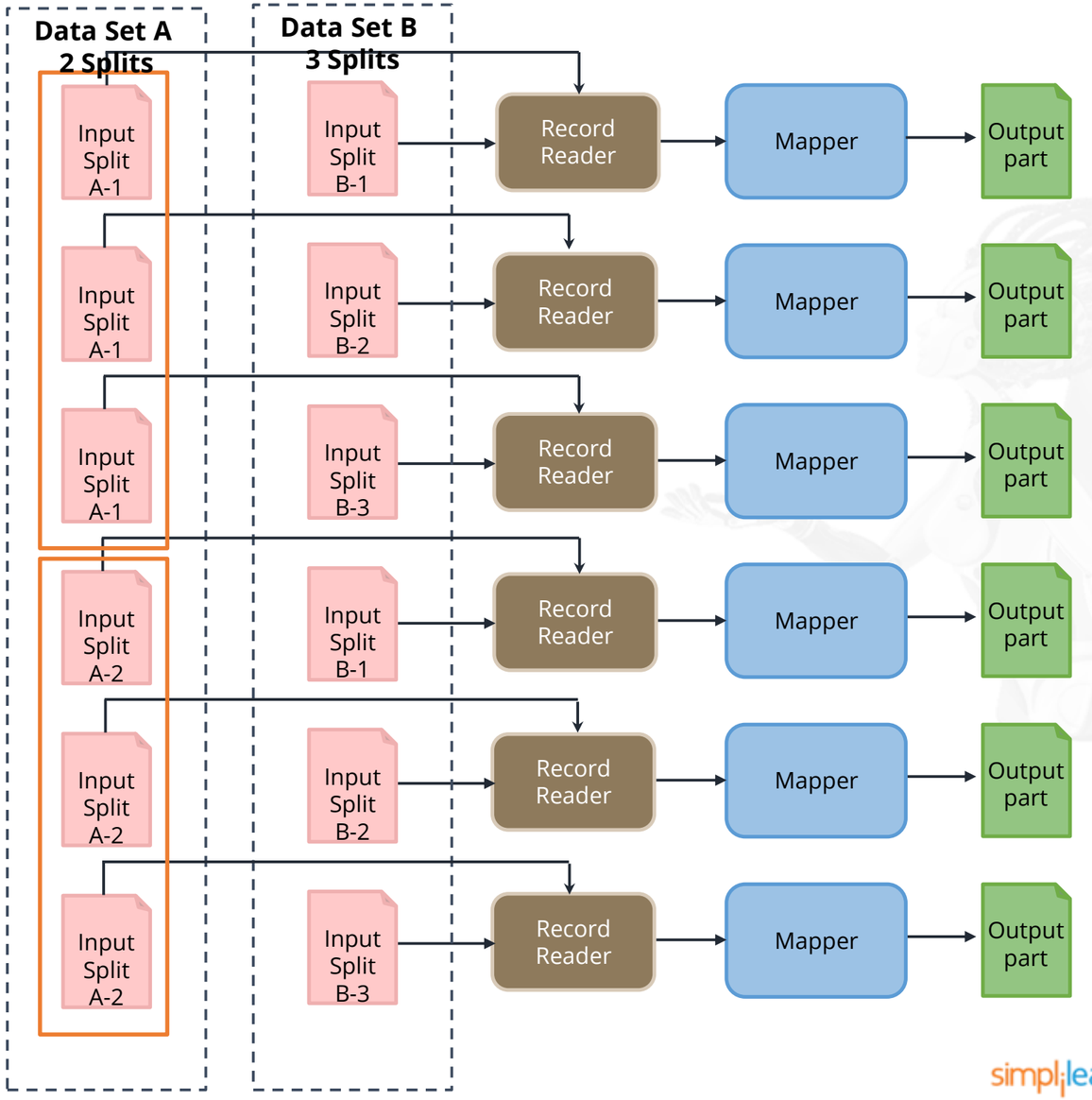
SELECT *users.ID, users.****Location****,*
comments.upVotes
FROM *users*
*[****INNER****|****LEFT****|****RIGHT****]* ***JOIN***
comments
ON *users.ID=comments.UserID*

Output of a composite join:
The number of part files equals the number of map tasks.

Cartesian Product

A Cartesian product is a map-only pattern that works in the following ways:

- Datasets are split into multiple partitions.
- Each partition is fed to one or more mappers. For example, split A-1 and A-2 are fed to 3 mappers each, as shown in the image.
- A RecordReader reads every record of input split associated with the mapper.
- The mapper simply pairs every record of a dataset with every record of all other datasets.



Cartesian Product

A Cartesian product should be used in the following conditions:

When there is a need to analyze relationships between all pairs of individual records

When there are no constraints on the execution time

SQL analogy

SELECT ***
FROM *users*
*[**CROSS**]* ***JOIN*** *comments*

Output of a Cartesian product:
Every possible tuple combination from the input records is represented in the final output.

DATA AND
ARTIFICIAL INTELLIGENCE

Introduction to Pig

Introduction to Pig

Different fundamentals

Need for common operation

Coding difficulty

Rigid dataflow

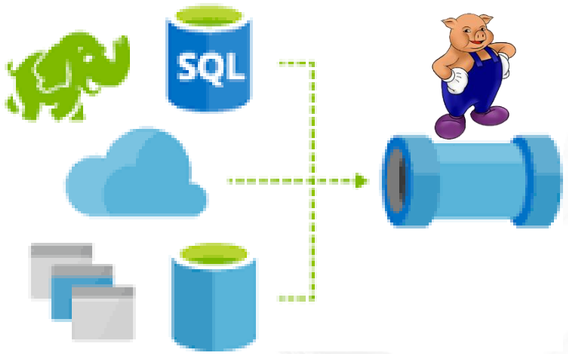
What is Pig?

- Extensible
- Self-optimizing
- Easily programmed

Pig is a scripting platform designed to process and analyze large data sets, and it runs on Hadoop clusters.

Pig—Example

Yahoo has scientists who use grid tools to scan through petabytes of data.

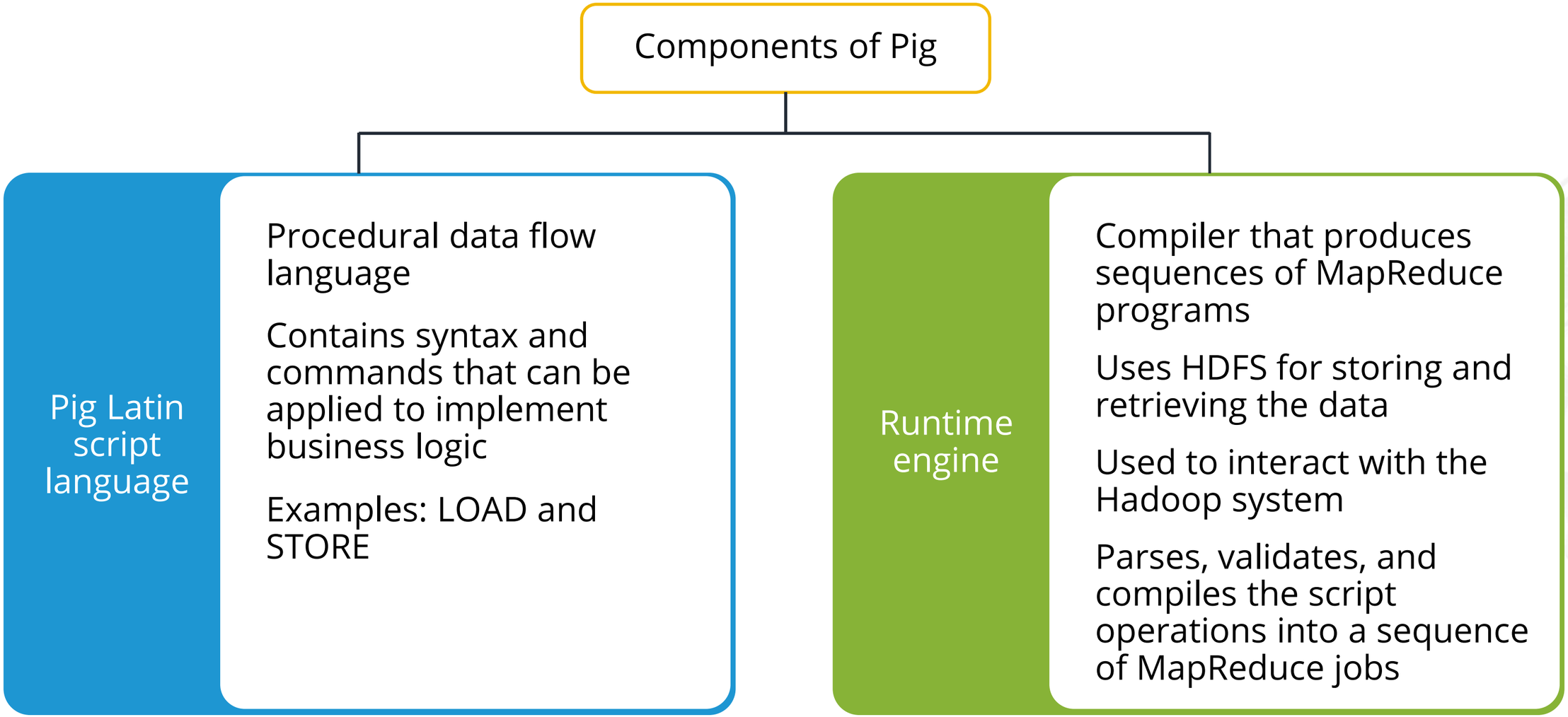


Write scripts to test a theory.

In the data factory, data may not be in a standardized state.

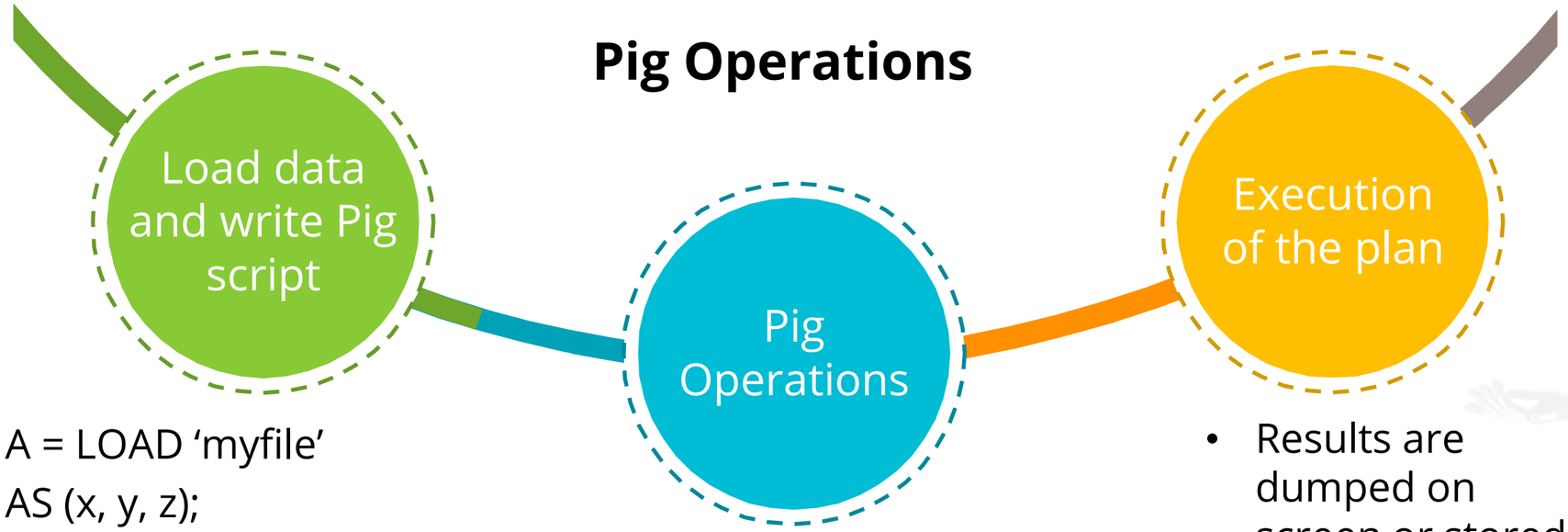
Pig supports data with partial or unknown schemas, and supports semi-structured or structured data.

Components of Pig



How Pig Works

Pig's operation can be explained in the following 3 stages:



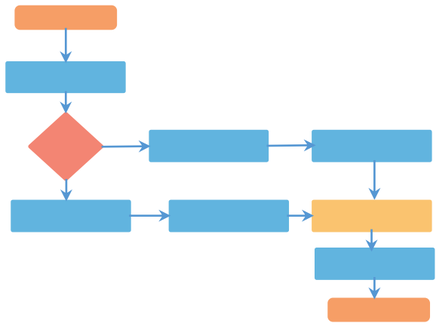

```
A = LOAD 'myfile'
AS (x, y, z);
B = FILTER A by x > 0;
C = GROUP B BY x;
D = FOREACH A GENERATE
x, COUNT(B);
STORE D INTO 'output';
```

- Parses and checks the script
- Optimizes the scripts
- Plans execution
- Submits to Hadoop
- Monitors job progress

- Results are dumped on screen or stored in HDFS

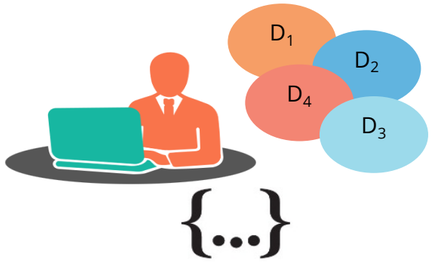
Salient Features

Developers and analysts like to use Pig as it offers many features.



Step-by-Step Procedural Control

Schemas can be Assigned Dynamically



Supports UDFs and Data Types

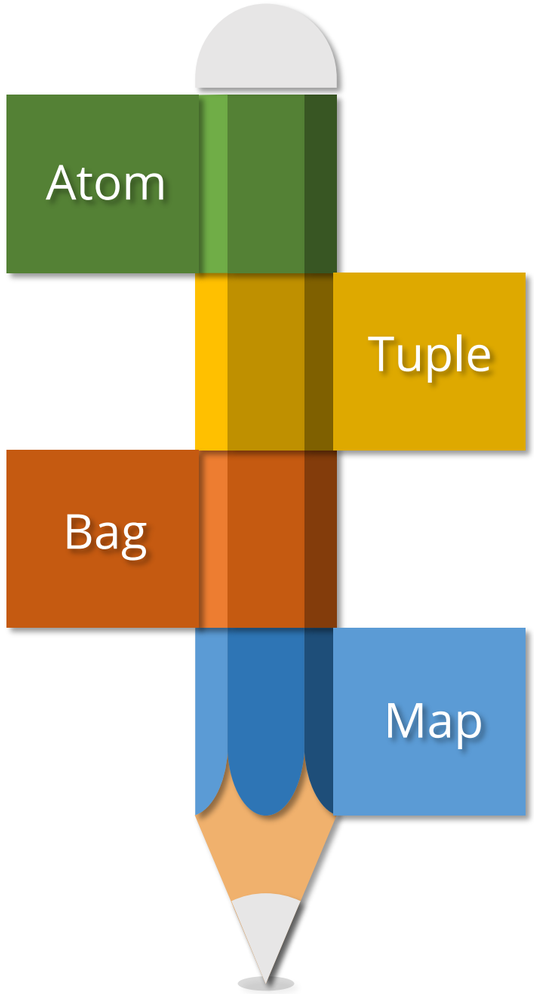
DATA AND
ARTIFICIAL INTELLIGENCE

Pig Data Model

Data Model

As part of its data model, Pig supports four basic types:

A simple atomic value
Example: ‘Mike’



A sequence of fields that can be of any data type
Example: (‘Mike’,43)

A collection of tuples of potentially varying structures that can contain duplicates
Example: {(‘Mike’), (‘Doug’, (43, 45))}

An associative array; the key must be a chararray, but the value can be of any type.
Example: [name#Mike,phone#5551212]

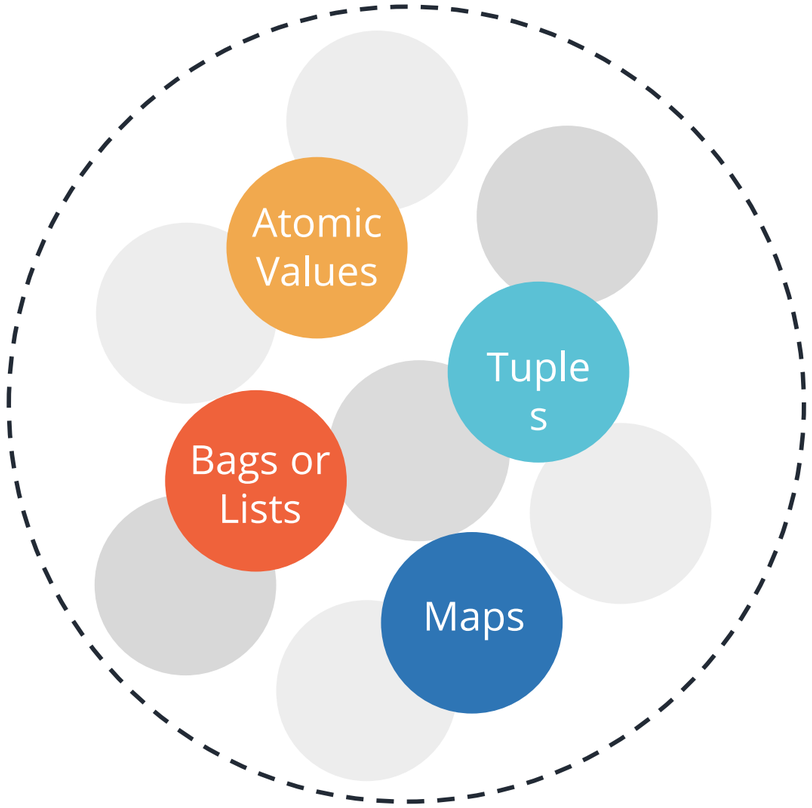
Data Model

- By default, Pig treats undeclared fields as ByteArrays.
- Pig can infer a field's type based on:
  - Use of operators that expect a certain type of field
  - User Defined Functions (UDFs) with a known or explicitly set return type
  - Schema information provided by a LOAD function or explicitly declared using an AS clause

Type conversion is lazy; the data type is enforced at execution only.

Nested Data Model

Pig Latin has a fully-nestable data model.



Advantages of nested data model

- More natural to programmers than flat tuples
- Avoids expensive joins

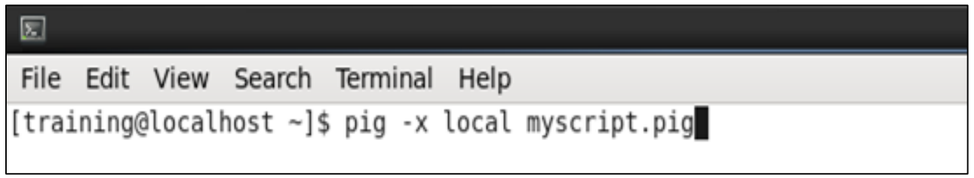
Pig Execution Modes

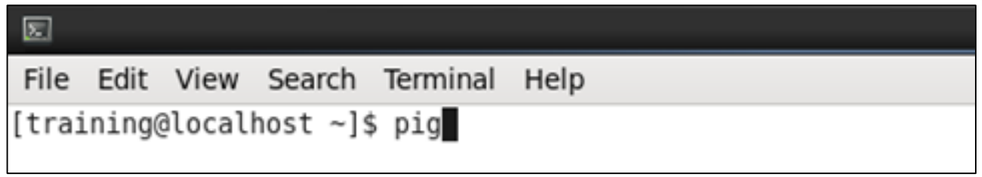
Pig works in two execution modes:

Pig execution modes

Local mode: Pig depends on the OS file system.

MapReduce mode: Pig relies on HDFS.



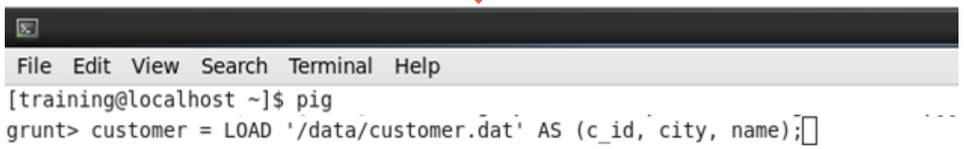


Pig Interactive Modes

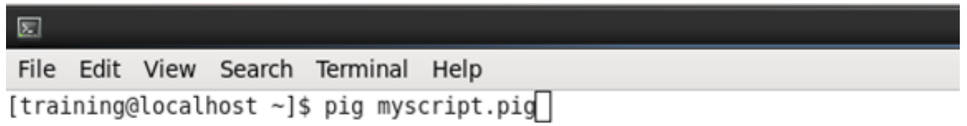
Pig Latin program can be written in two interactive modes:

Pig Interactive Modes

Interactive mode: A line by line code is written and executed.



Batch mode: A file containing Pig scripts is created and executed in a batch.



Pig vs. SQL

The differences between Pig and SQL are given below:

Difference	 Pig	 SQL
Definition	Scripting language used to interact with HDFS	Query language used to interact with databases
Query Style	Step-by-step	Single block
Evaluation	Lazy evaluation	Immediate evaluation
Pipeline Splits	Pipeline splits are supported	Requires the join to be run twice or materialized as an intermediate result

Pig vs. SQL: Example

Track customers in Texas who spend more than $2,000

SQL	Pig

SELECT c_id , SUM(amount) AS CTotal
FROM customers c
JOIN sales s ON c.c_id = s.c_id
WHERE c.city = 'Texas'
GROUP BY c_id
HAVING SUM(amount) > 2000
ORDER BY CTotal DESC
	customer = LOAD '/data/customer.dat' AS (c_id,name,city);
	sales = LOAD '/data/sales.dat' AS (s_id,c_id,date,amount);
	salesBLR = FILTER customer BY city == 'Texas';
	joined= JOIN customer BY c_id, salesTX BY c_id;
	grouped = GROUP joined BY c_id;
	summed= FOREACH grouped GENERATE GROUP,
	SUM(joined.salesTX::amount);
	spenders= FILTER summed BY $1 > 2000;
	sorted = ORDER spenders BY $1 DESC;
	DUMP sorted;

Getting Datasets for Pig Development

Use the following URLs to download different datasets for Pig development:

Datasets	URL
Books	www.gutenberg.org (war_and_peace.txt)
Wikipedia database	http://dumps.wikimedia.org/enwiki/
Open data base from Amazon S3 data	https://aws.amazon.com/public-data-sets/
Open database from National climate data	http://cdo.ncdc.noaa.gov/qclcd_ascii/

DATA AND
ARTIFICIAL INTELLIGENCE

Pig Operations

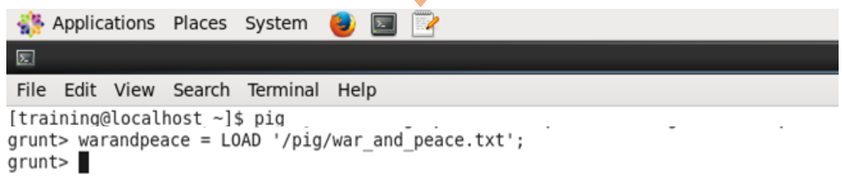
Loading and Storing Methods

Loading refers to loading relations from files in the Pig buffer.

Storing refers to writing outputs to the file system.

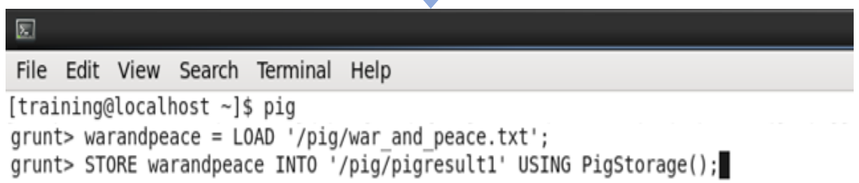
Load Data

Keyword:
LOAD



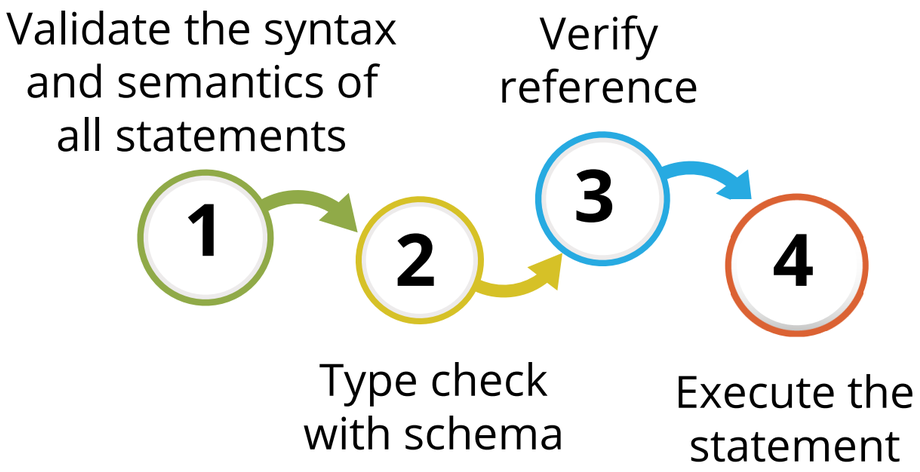
Store Data

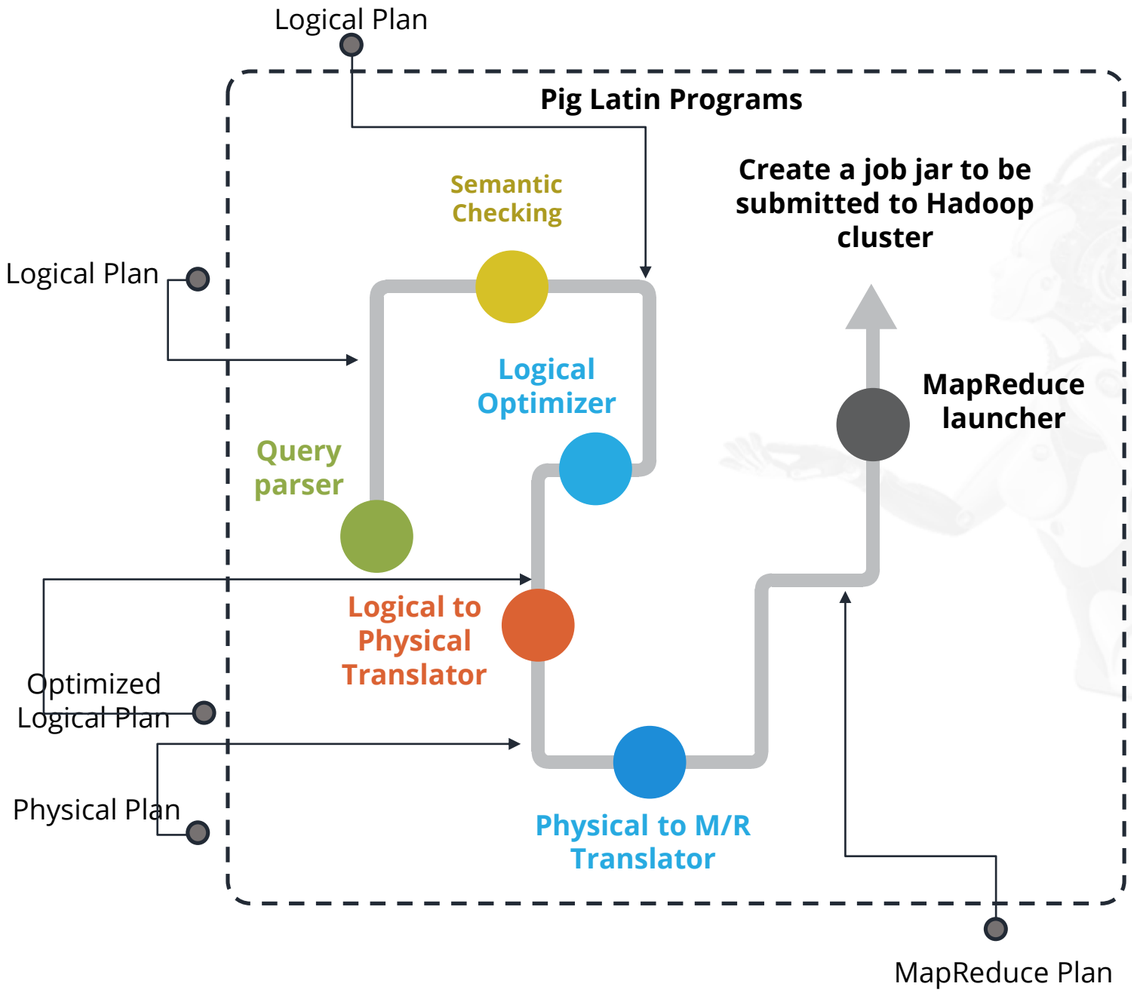
Keyword: STORE



Script Interpretation

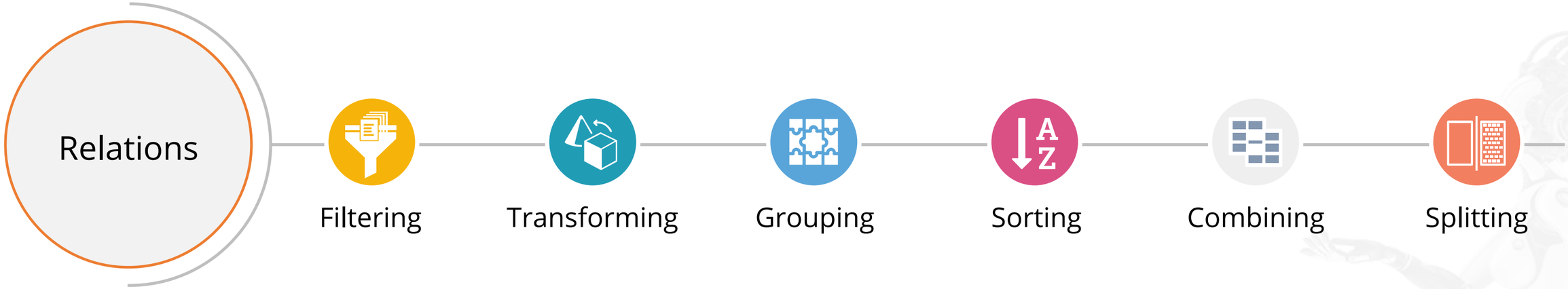
Pig processes Pig Latin statements in the following manner:





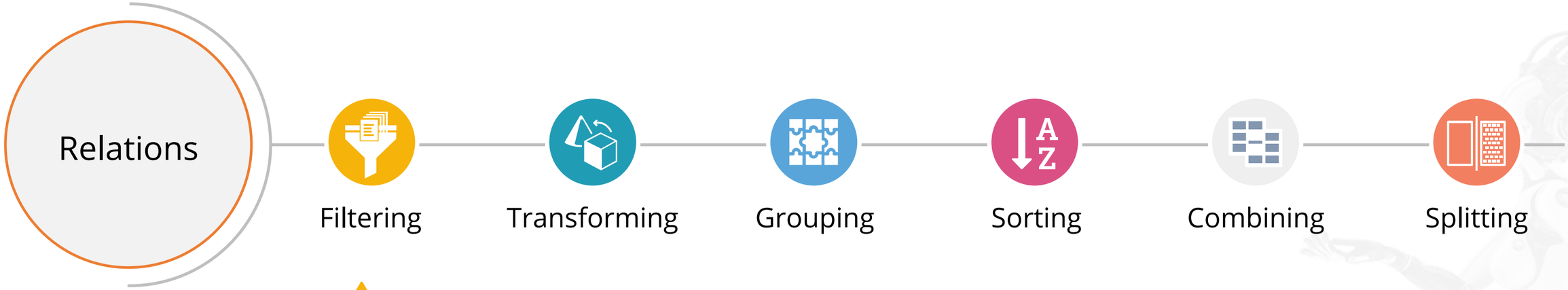
Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:



Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:

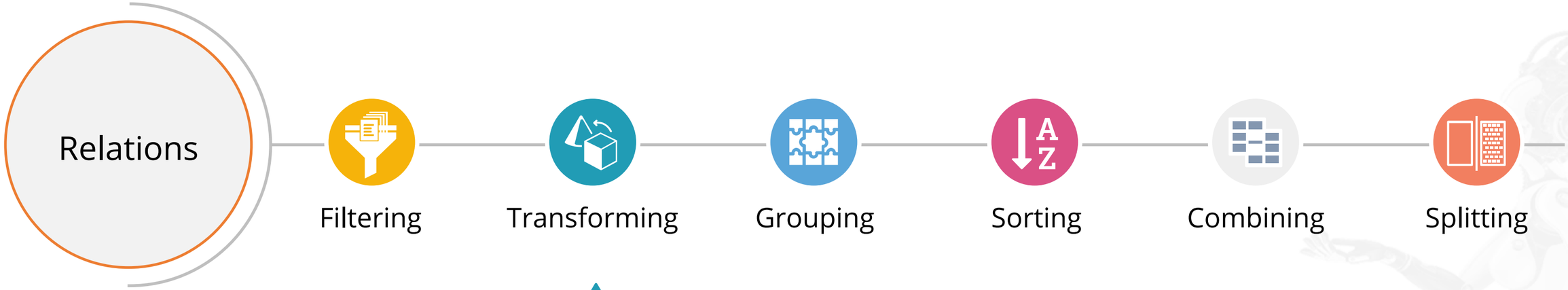


Filtering can be defined as filtering of data based on a conditional clause such as grade and pay.

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ pig
grunt> FILTER mbaresults by grade = 'A';
```

Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:

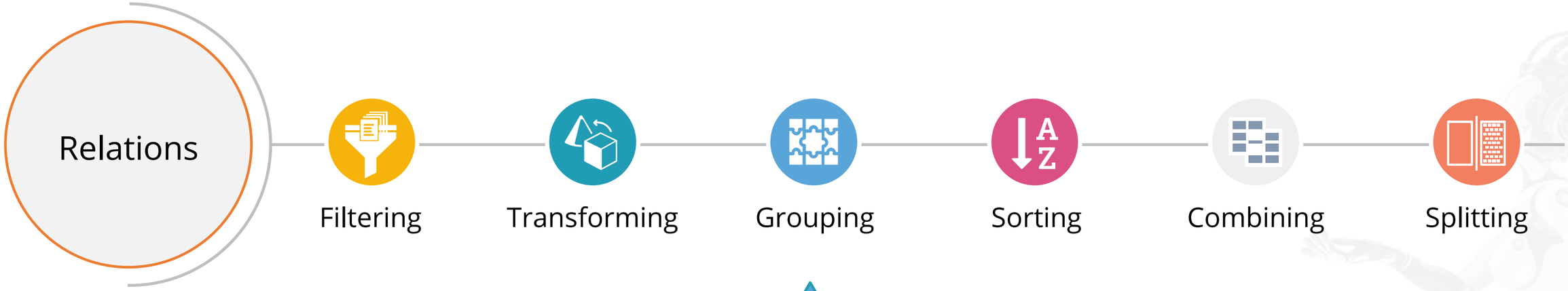


Transforming refers to making data presentable for the extraction of logical data.

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ pig
grunt> transform = FOREACH book GENERATE FLATTEN(TOKENIZE(lines)) as word;
```

Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:

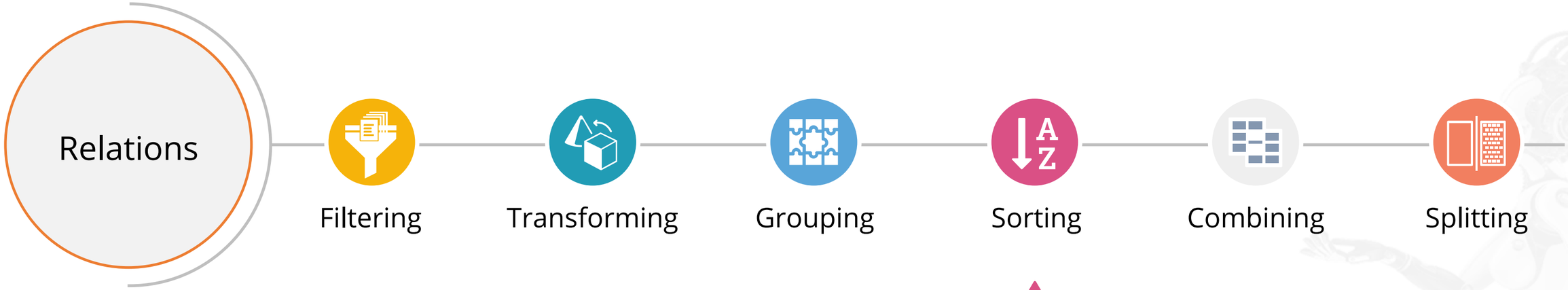


Grouping refers to generating a group of meaningful data.

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ pig
grunt> grouped = GROUP words by words;
```

Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:

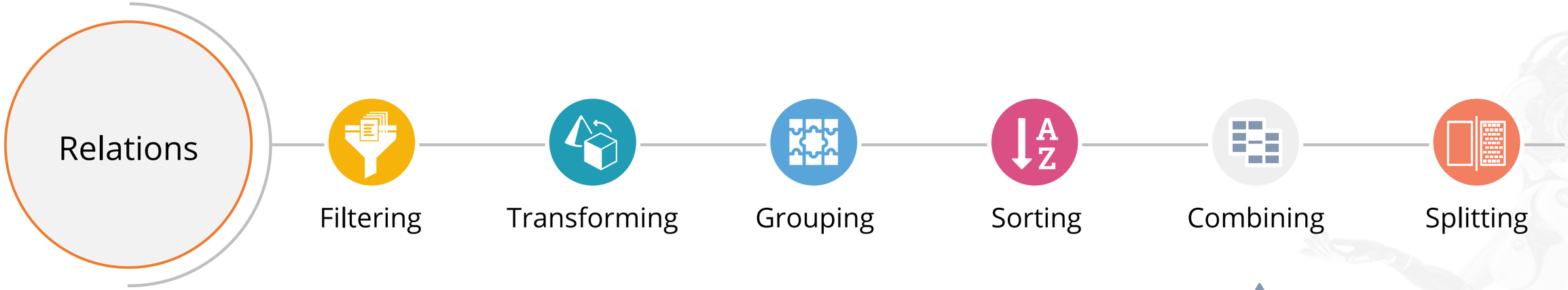


Sorting of data refers to arranging the data in either ascending or descending order.

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ pig
grunt> sorted = ORDER words by $1 DESC;
```

Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:

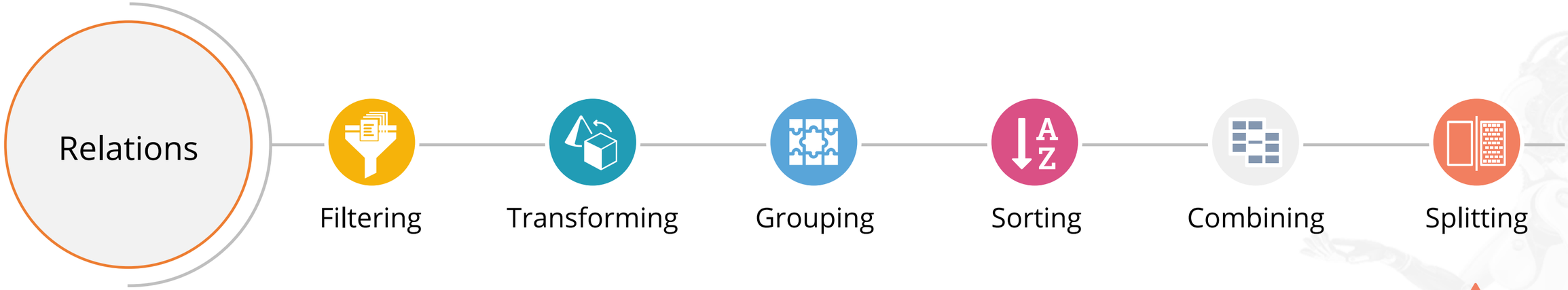


Combining refers to performing a union operation on the data stored in the variable.

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ pig
grunt> bookscombined = UNION book1, book2;
```

Various Relations Performed by Developers

Some of the relations performed by Big Data and Hadoop Developers are as follows:



Splitting refers to separating data that has logical meaning.

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ pig
grunt> SPLIT bookscombined INTO book1 IF SUBSTRING(isbn,4,6) == '1234', book2 IF SUBSTRING(isbn,4,6) == '3456';
```

Various Pig Commands

These are some of the Pig commands:

Pig command	Functions
load	Reads data from system
Store	Writes data to file system
foreach	Applies expressions to each record and outputs one or more records
filter	Applies predicate and removes records that do not return true
Group or cogroup	Collects records with the same key from one or more inputs
join	Joins two or more inputs based on a key
order	Sorts records based on a key
distinct	Removes duplicate records
union	Merges data sets
split	Splits data into two or more sets, based on filter conditions
stream	Sends all records through a user-provided binary
dump	Writes output to stdout
limit	Limits the number of records

Assisted Practice

MapReduce and Pig

Duration: 15 mins

Problem Statement: In this demonstration, you will analyze web log data using MapReduce and solve various real-world KPIs.

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**.

Assisted Practice

Apache Pig

Duration: 15 mins

Problem Statement: In this demonstration, you will analyze sales data and solve KPIs using Pig.

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**.

Unassisted Practice

Apache Pig

Duration: 15 mins

Problem Statement: Use pig to count the number of words from a text file.

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

- **Pig**

  $ pig -x local

  grunt> lines = LOAD "sampletextfile.txt" AS (line:chararray);

  grunt> words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;

  grunt> DUMP words

  grunt> grouped_words = GROUP words BY word;

  grunt> DUMP grouped_words

  grunt> wordcount = FOREACH grouped_words GENERATE group, COUNT(words);

  grunt> DUMP wordcount

Key Takeaways

You are now able to:

- Perform distributed processing in MapReduce
- Understand issues with distributed processing and how MR reduces them
- Implement MapReduce paradigm, divide-and-conquer
- Perform Word Count program - Hello World of big data
- Understand Pig
- Implement Pig Latin commands

Quiz

Knowledge Check

1

Which of the following commands is used to start Pig in MapReduce mode?

a. Pig

b. Pig -x MapReduce

c. Pig -x local

d. Both Pig and Pig -x MapReduce

Knowledge Check

1

Which of the following commands is used to start Pig in MapReduce mode?

a. Pig

b. Pig -x MapReduce

c. Pig -x local

d. Both Pig and Pig -x MapReduce

The correct answer is **d.**

Pig or Pig -x MapReduce command can be used to run Pig in MapReduce mode.

Knowledge Check

2

Which of the following commands is used to start Pig in local mode?

a. Pig -x local

b. Pig -x MapReduce

c. Pig

d. Both b and c

Knowledge Check

2

Which of the following commands is used to start Pig in local mode?

a. Pig -x local

b. Pig -x MapReduce

c. Pig

d. Both b and c

The correct answer is **a.**

Pig -x local command is used to start Pig in local mode.

Knowledge Check

3

How many phases exist in MapReduce?

a. 4

b. 5

c. 6

d. 2

Knowledge Check

3

How many phases exist in MapReduce?

a. 4

b. 5

c. 6

d. 2

The correct answer is **b.**

MapReduce consists of five phases: Map phase, Partition phase, Shuffle phase, Sort phase, and Reduce phase.

Knowledge Check

4

In how many ways can Pig Latin program be written?

a. 2

b. 4

c. 3

d. 5

Knowledge Check

4

In how many ways can Pig Latin program be written?

a. 2

b. 4

c. 3

d. 5

The correct answer is **a.**

Pig Latin program can be written in two ways.

Knowledge Check

5

Which of the following is the first step to build a MapReduce program?

a. Launch the job and monitor

b. Package the code

c. Determine the data

d. None of the above

Knowledge Check

5

Which of the following is the first step to build a MapReduce program?

a. Launch the job and monitor

b. Package the code

c. Determine the data

d. None of the above

The correct answer is **c.**

Determining the data is the first step to build a Mapreduce program.

Knowledge Check

6

Which of the following is a real-time use case of MapReduce?

a. Enterprise Analytics

b. Gaussian analysis

c. Search engine operations

d. All of the above

Knowledge Check

6

Which of the following is a real-time use case of MapReduce?

a. Enterprise Analytics

b. Gaussian analysis

c. Search engine operations

d. All of the above

The correct answer is **d.**

Enterprise analytics, Gaussian analysis, and search engine operations are real-time use cases of MapReduce.

Lesson-End Project

Problem Statement:

The US Department of Transport collects statistics from all the airlines, which include airline details, airport details, and flight journey details.
These airlines have global presence and they operate almost in every country.
Flight data can help to decide which airline provides better service and find the routes in which flights are getting delayed.

The data collected is present in the files: flight.csv, airline.csv, and aiport.csv.
You are hired as a big data consultant to provide important insights.

Your task is to write MapReduce jobs and provide insights on:

1. The number of flights that get canceled in a month, every year
2. The airline names and the number of times their flights were canceled and diverted
3. The number of times flights were delayed for each airline

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Apache Hive

Learning Objectives

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

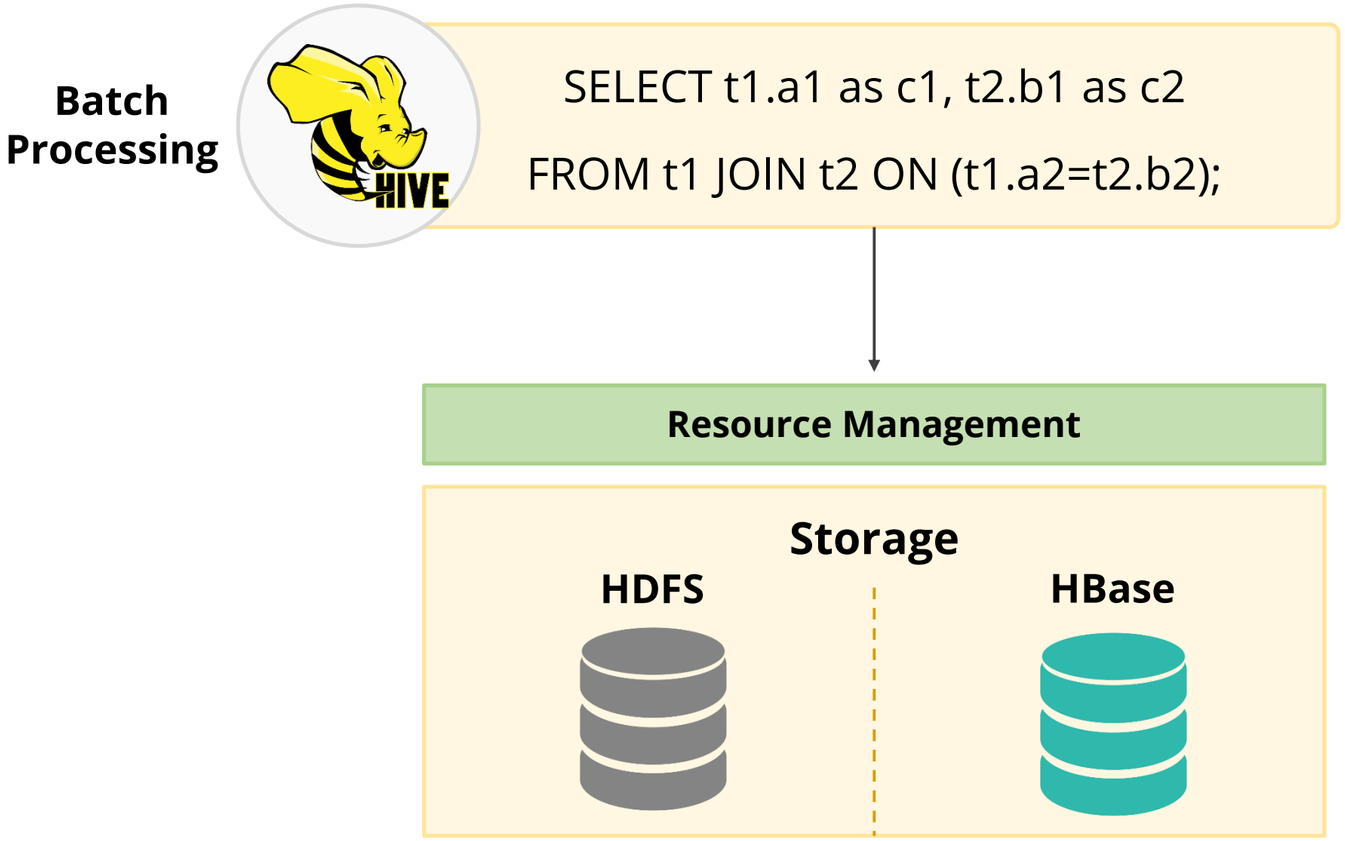
- Define Hive and its architecture
- Create and manage tables using Hue Web UI and Beeline
- Understand various file formats supported in Hive
- Use HiveQL DDL to create tables and execute queries

DATA AND
ARTIFICIAL INTELLIGENCE

Hive: SQL over Hadoop MapReduce

Apache Hive

Hive provides a SQL like interface for users to extract data from the Hadoop system.



Features of Hive

- Originally developed by Facebook around 2007
- Is an open-source Apache project
- High level abstraction layer on top of MapReduce and Apache Spark
- Uses HiveQL
- Suitable for structured data

Case Study

A leading online education company uses Hive and Impala to analyze social media coverage.

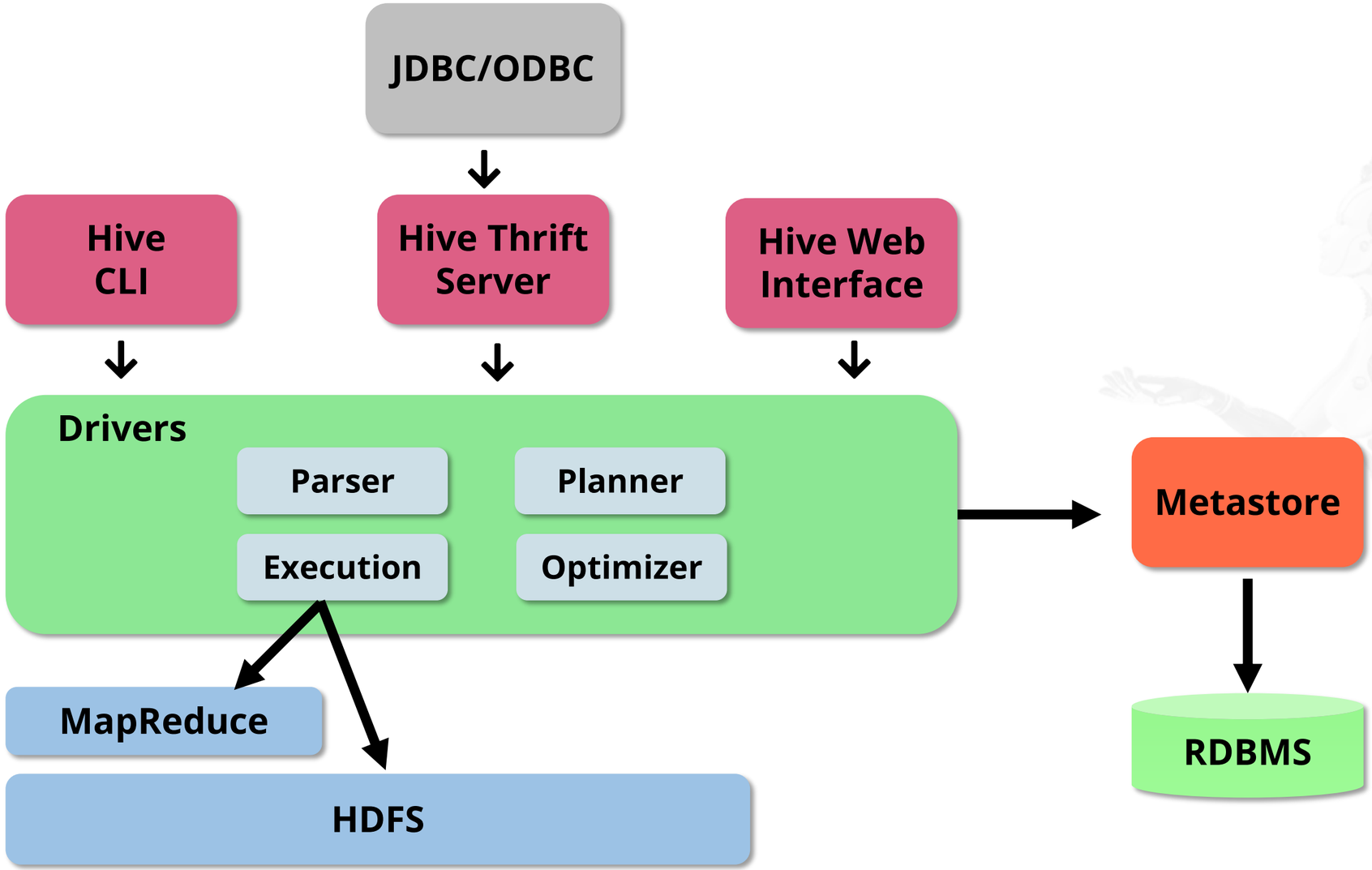
The organization analyzes positive, negative, and neutral reviews using Hive.

DATA AND
ARTIFICIAL INTELLIGENCE

Hive Architecture

Hive Architecture

The major components of Hive architecture are: **Hadoop core components**, **Metastore**, **Driver**, and **Hive clients**.



Job Execution Flow in Hive

Receive SQL query

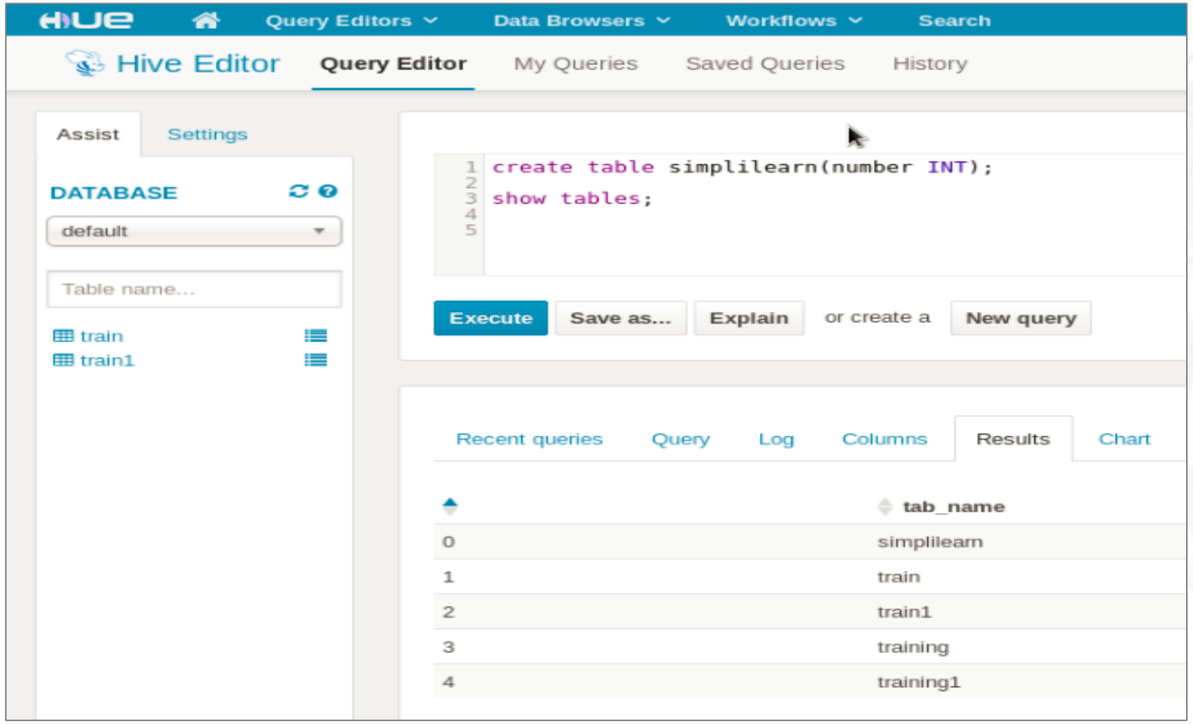
1. Parse HiveQL
2. Make optimizations
3. Plan execution
4. Submit job(s) to cluster
5. Monitor progress
6. Process data in MapReduce or Apache Spark
7. Store the data in HDFS

Interfaces to Run Hive Queries

Hive offers many interfaces for running queries.

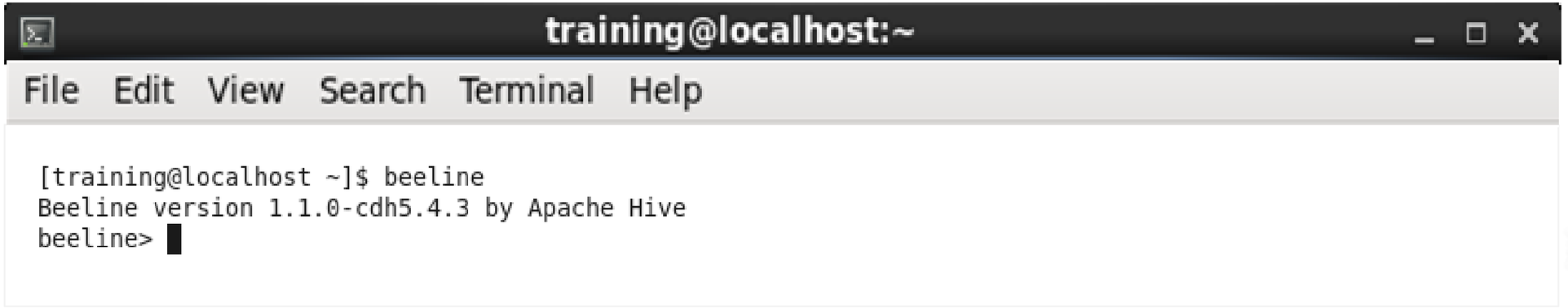
- **Command-line shell**
  - Hive: Beeline
- **Hue Web UI**
  - Hive Query Editor
- **Metastore Manager**
  - ODBC / JDBC

Hive Query Editor

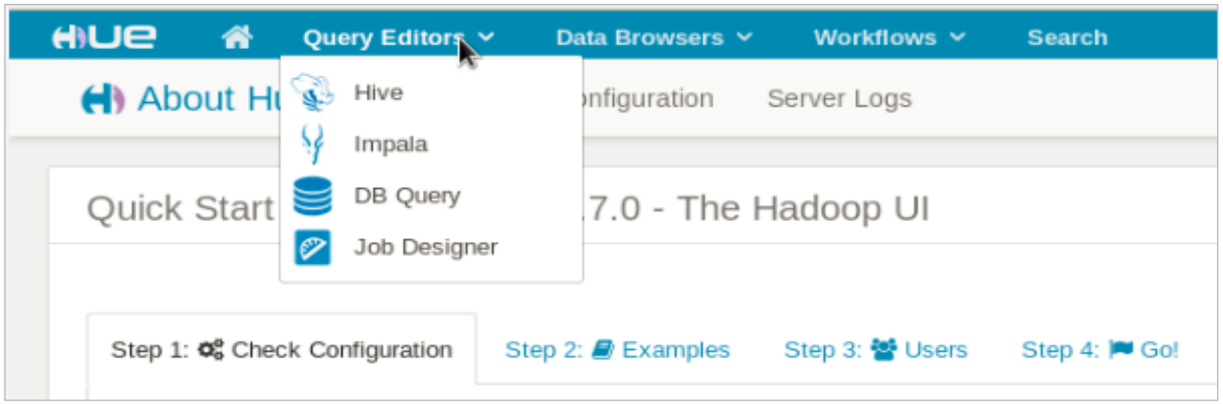


Connecting with Hive

Hive can be run using Beeline



Hue can be used to write a Hive query from a UI

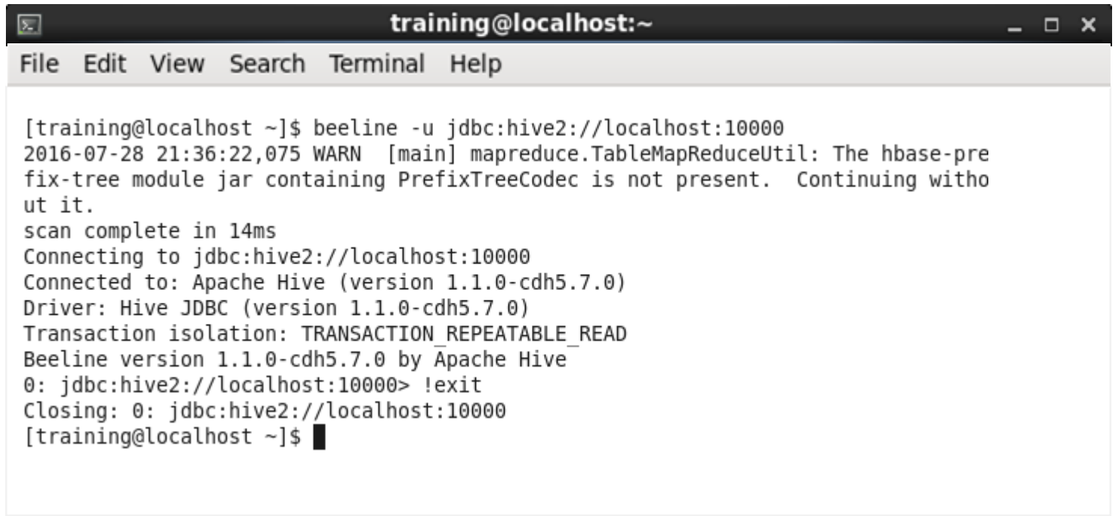


Running Hive Queries Using Beeline

'!' is used to execute Beeline commands.

Below are a few commands for running Beeline:

- !exit – to exit the shell
- !help – to show list of all commands
- !verbose – to show added details of queries



Running Beeline from Command Line

Below are the command lines for running Beeline

To execute file using the –u option

```
beeline –u … -f simplilearn.hql
```

To use HiveQL directly from the command line using the -e option

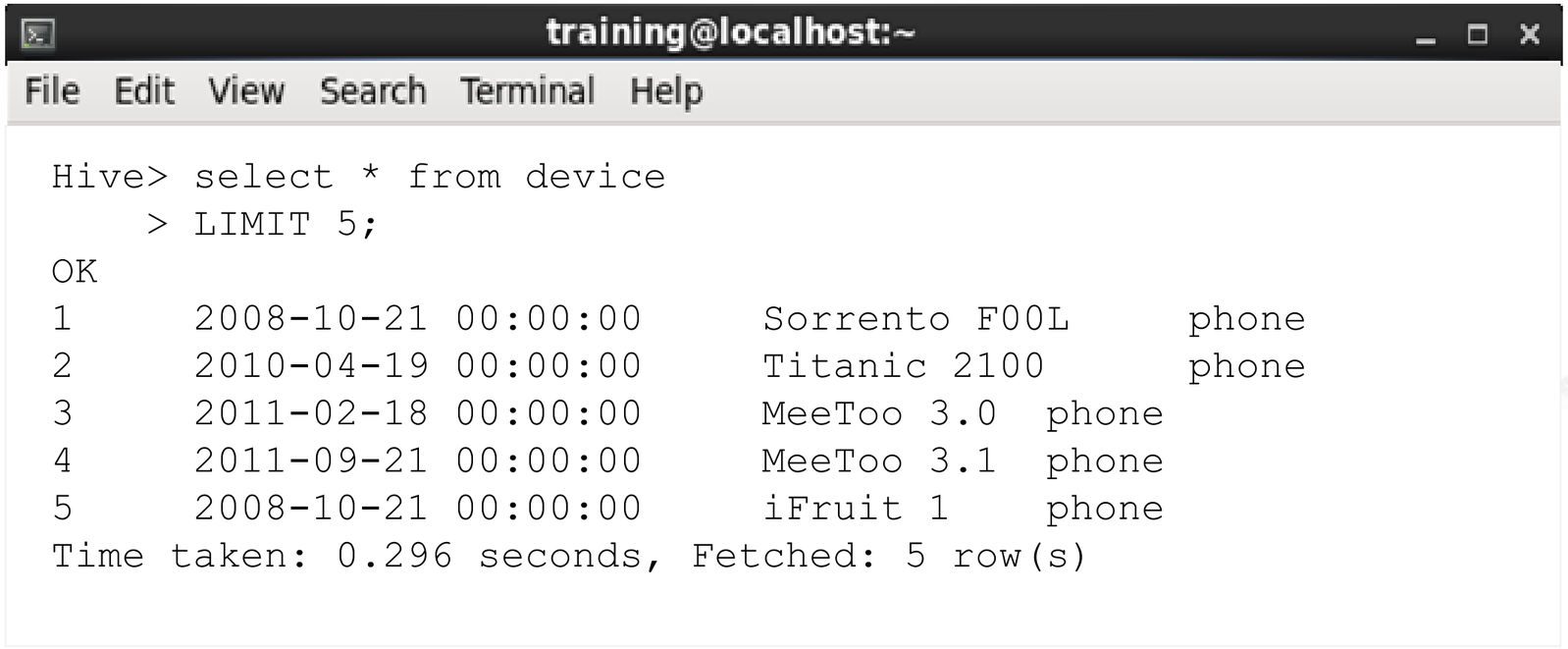
```
beeline –u ... -e 'SELECT * FROM users‘
```

To continue running script even after an error

```
beeline –u … -force=TRUE
```

Running Hive query

SQL commands are terminated with a semicolon (;)



Hive Editors in Hue

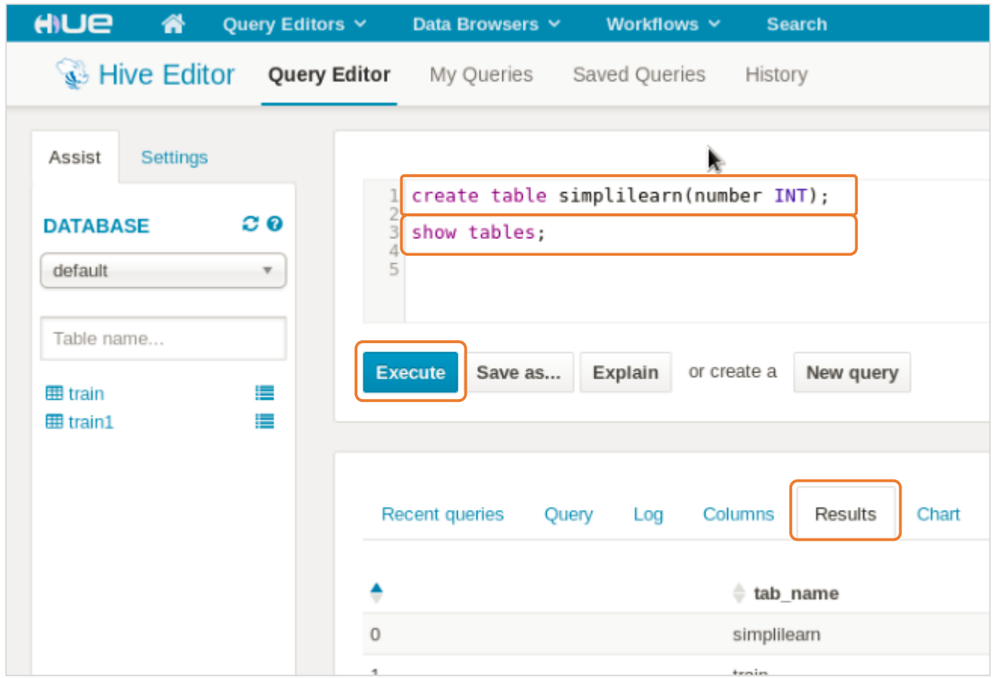


Diagram 1

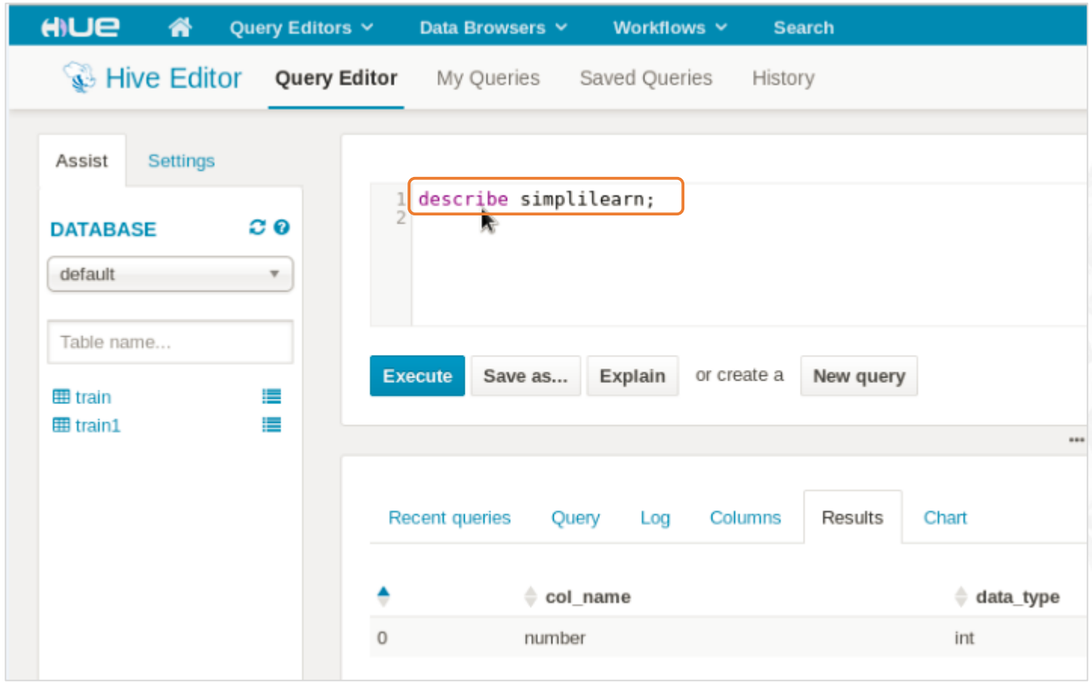


Diagram 2

DATA AND
ARTIFICIAL INTELLIGENCE

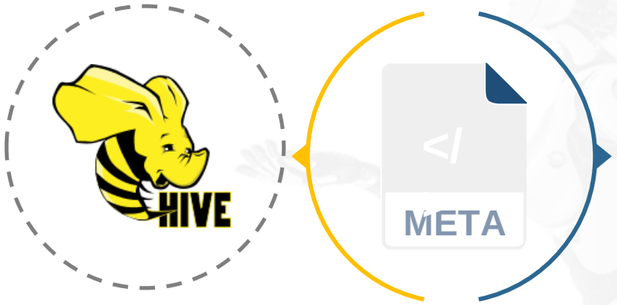
Hive Metastore

Managing Data with Hive

Hive uses Metastore service to store metadata for Hive tables.

- A table is an HDFS directory containing zero or more files

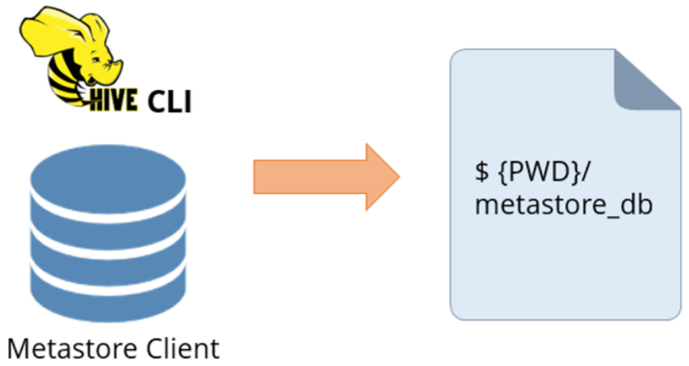
Default path: **/user/hive/warehouse/<*table_name*>**

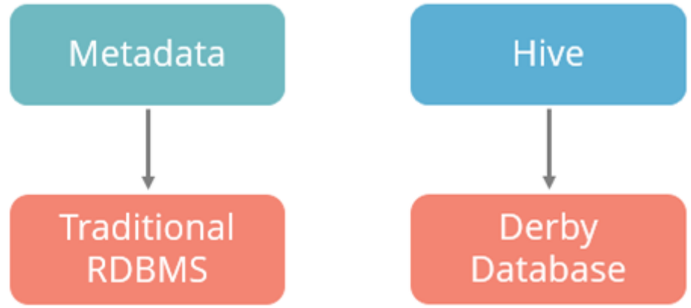


- Table supports many formats for data storage and retrieval
- Metastore stores the created metadata
  - Contained in an RDBMS such as MySQL
- Hive Tables are stored in HDFS and the relevant metadata is stored in the Metastore

What Is Hive Metastore?

The Metastore is the component that stores the system catalog which contains metadata about tables, columns, and partitions.

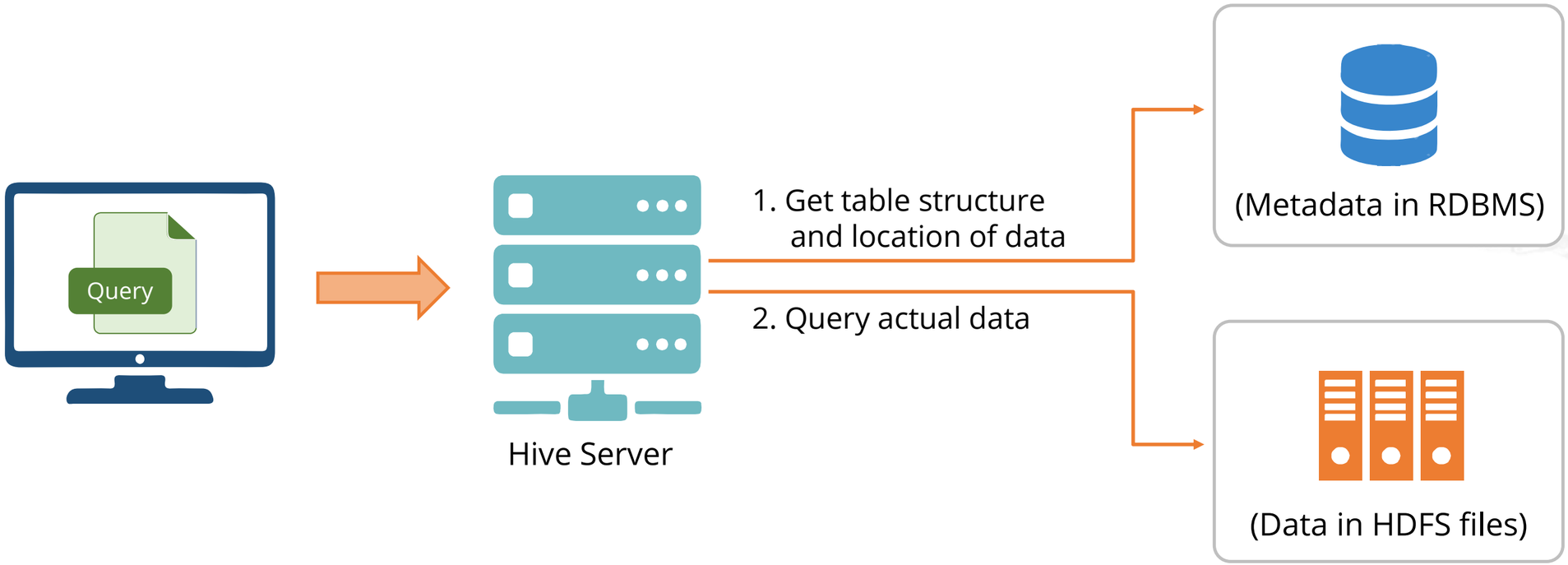




Parameter	Description	Example
Javax.jdo.option.ConnectionURL	JDBC connection URL along with database name containing metadata	jdbc:derby:;databaseName=metastore_db;create=true
Javax.jdo.option.ConnectionDriverName	JDBC driver name. Embedded Derby for Single user mode.	Org.apache.derby.jdbc.EmbeddedDriver
Javax.jdo.option.ConnectionUserName	User name for Derby database	APP
Javax.jdo.option.ConnectionPassword	Password	mine

Use of Metastore in Hive

- Hive uses metastore to get table structure and location of data
- The server queries actual data which is stored in HDFS



Data Warehouse Directory Structure

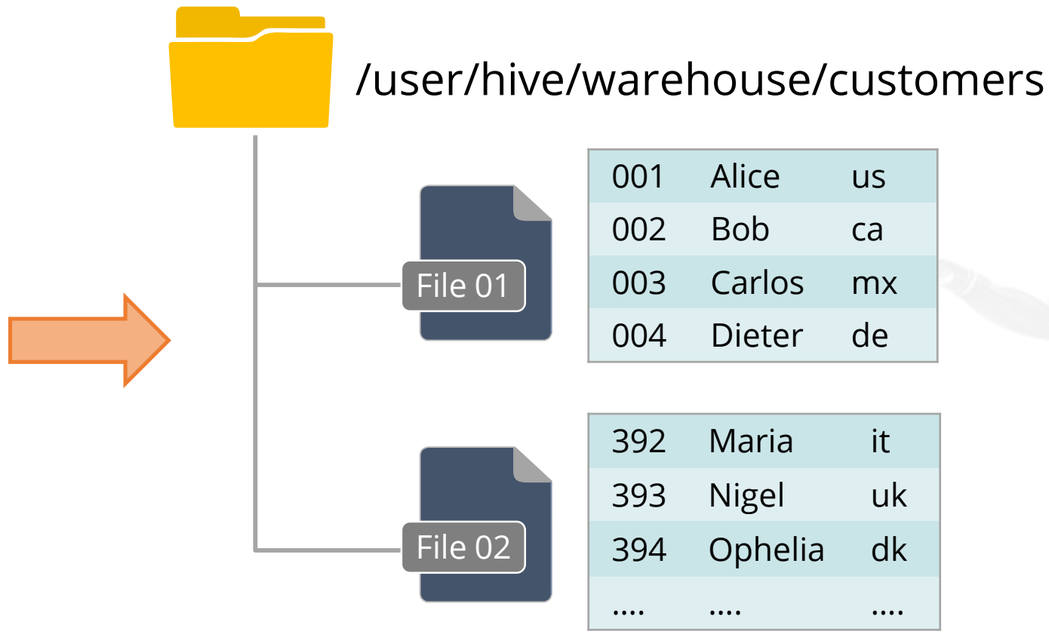
- By default, all data gets stored in

  /user/hive/warehouse

- Each table is a directory within the default location having one or more files

Customers Table

customer_id	name	country
001	Alice	us
002	Bob	ca
003	Carlos	mx
....	....	....
392	Maria	it
393	Nigel	uk
394	Ophelia	dk
....	....	....



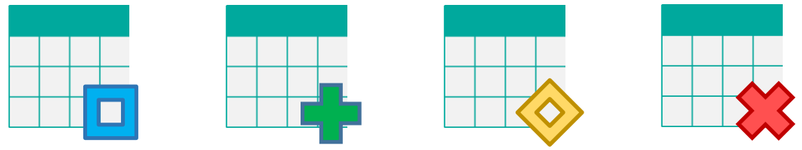
In HDFS, Hive data can be split into more than one file.

DATA AND
ARTIFICIAL INTELLIGENCE

Hive DDL and DML

Defining Database and Table

- Databases and tables are created and managed using the DDL (Data Definition Language) of HiveQL
- They are very similar to standard SQL DDL
- For example, Create/Drop/Alter/Use Database



Creating a Database

- To create a new database

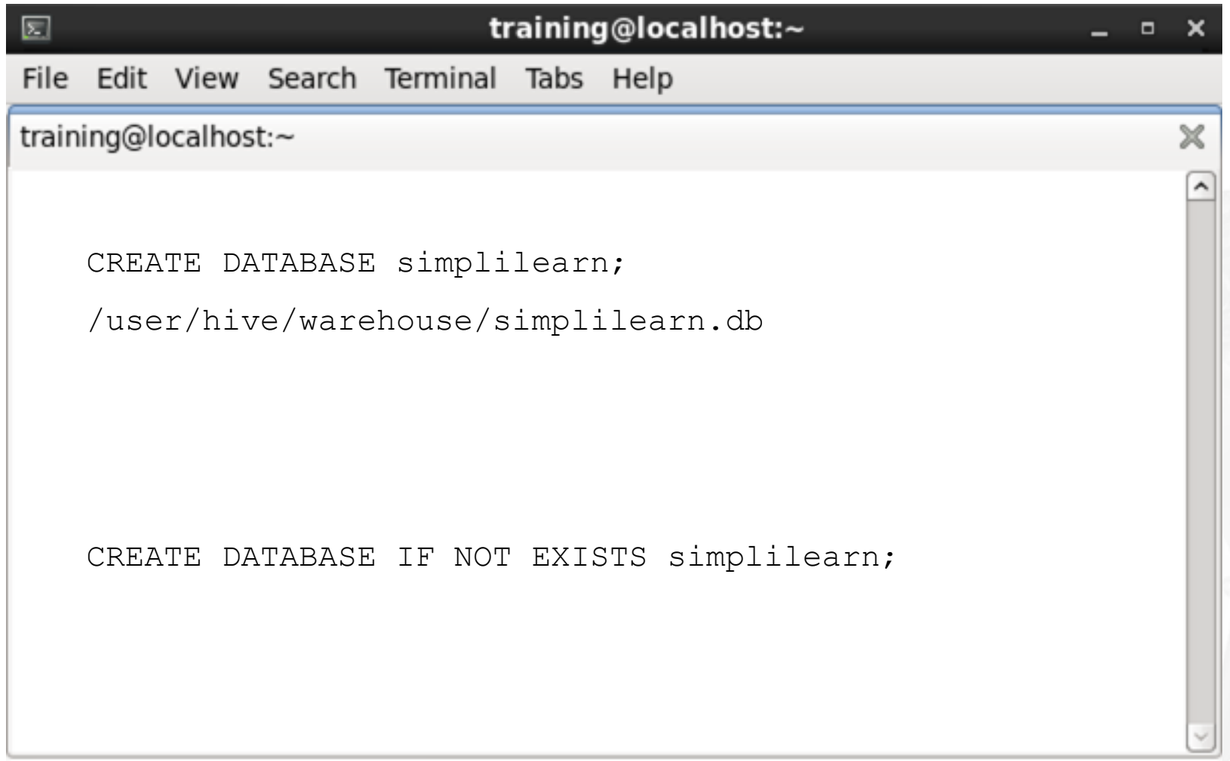
  CREATE DATABASE <dbname>;

  The above statement will add database definition to the metastore and will also create a storage directory in HDFS in the default location.

  For example: /user/hive/warehouse/simplilearn.db

- In order to avoid error in case database simplilearn already exists:

  CREATE DATABASE IF NOT EXISTS <dbname>;



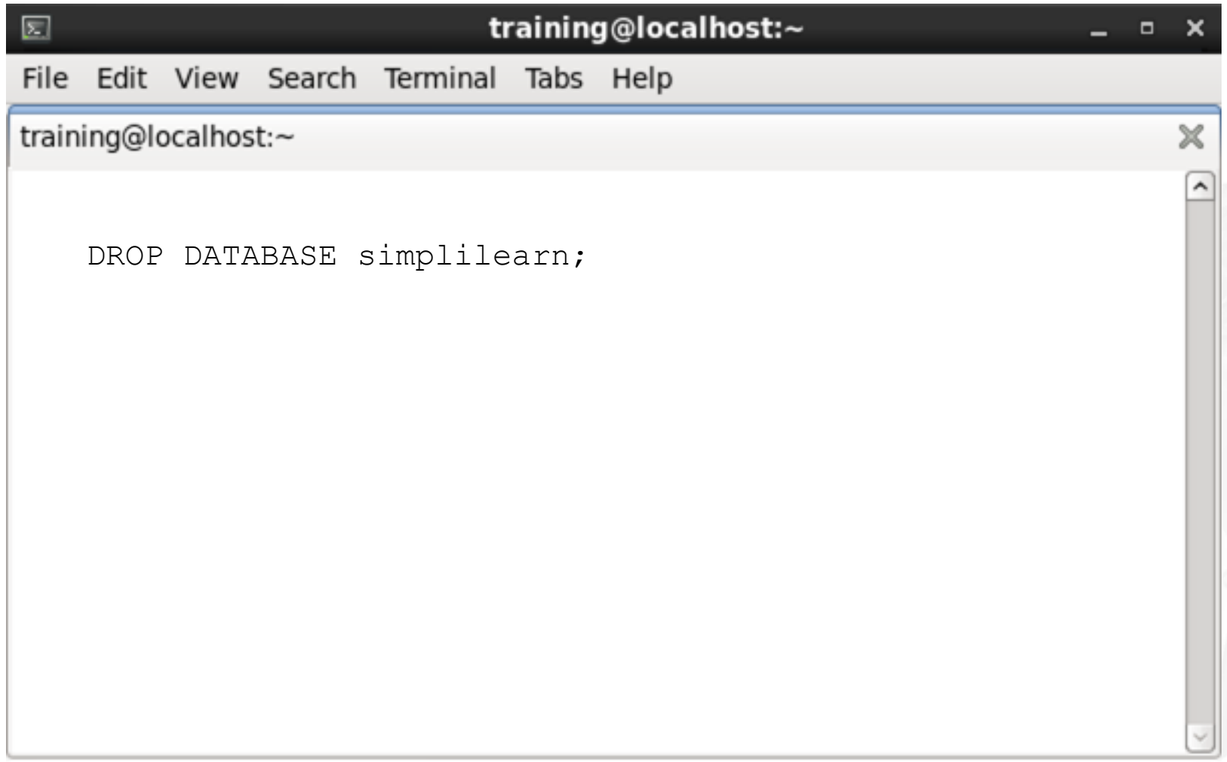
Deleting a Database

- Removing a database is similar to creating it
  - replace CREATE with DROP

    DROP DATABASE <dbname>;

    - In case the database already exists, you can DROP DATABASE IF EXISTS <dbname>;
- In order to remove database, if it has some table :

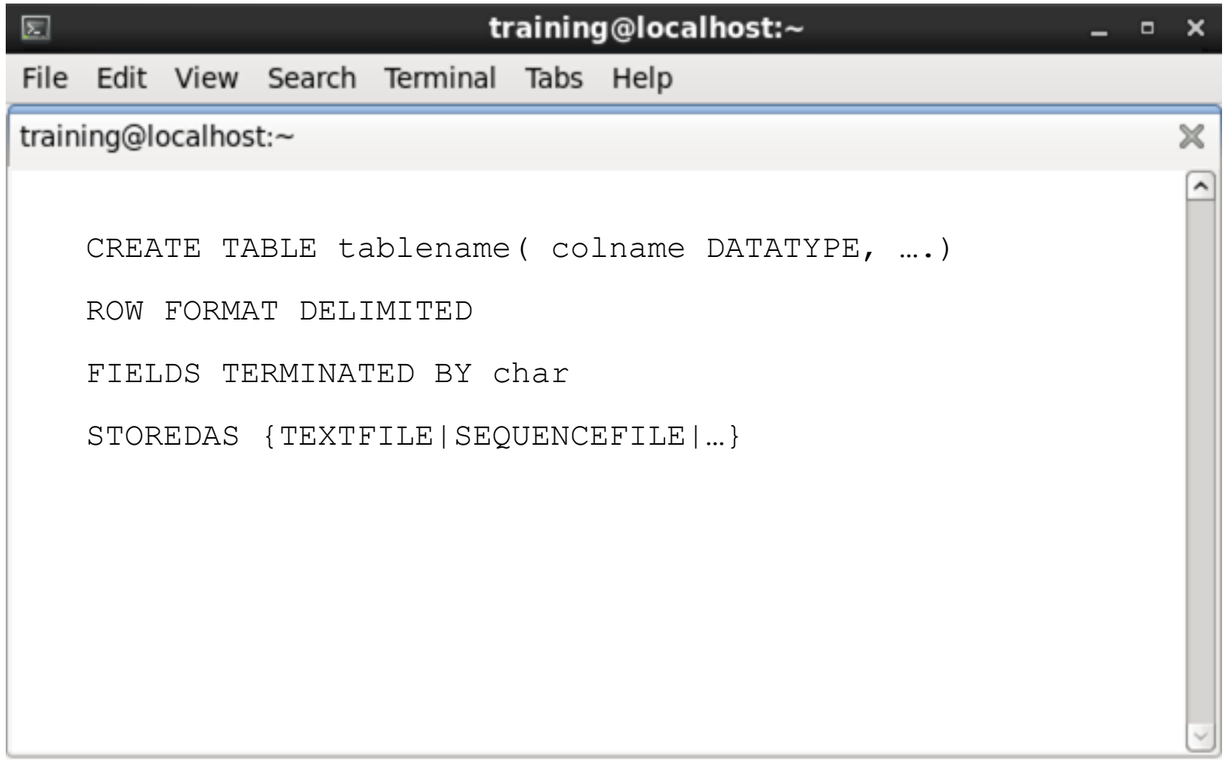
  DROP DATABASE <dbname> CASCADE;



This might remove data in HDFS.

Creating New Table

Syntax to create a new table



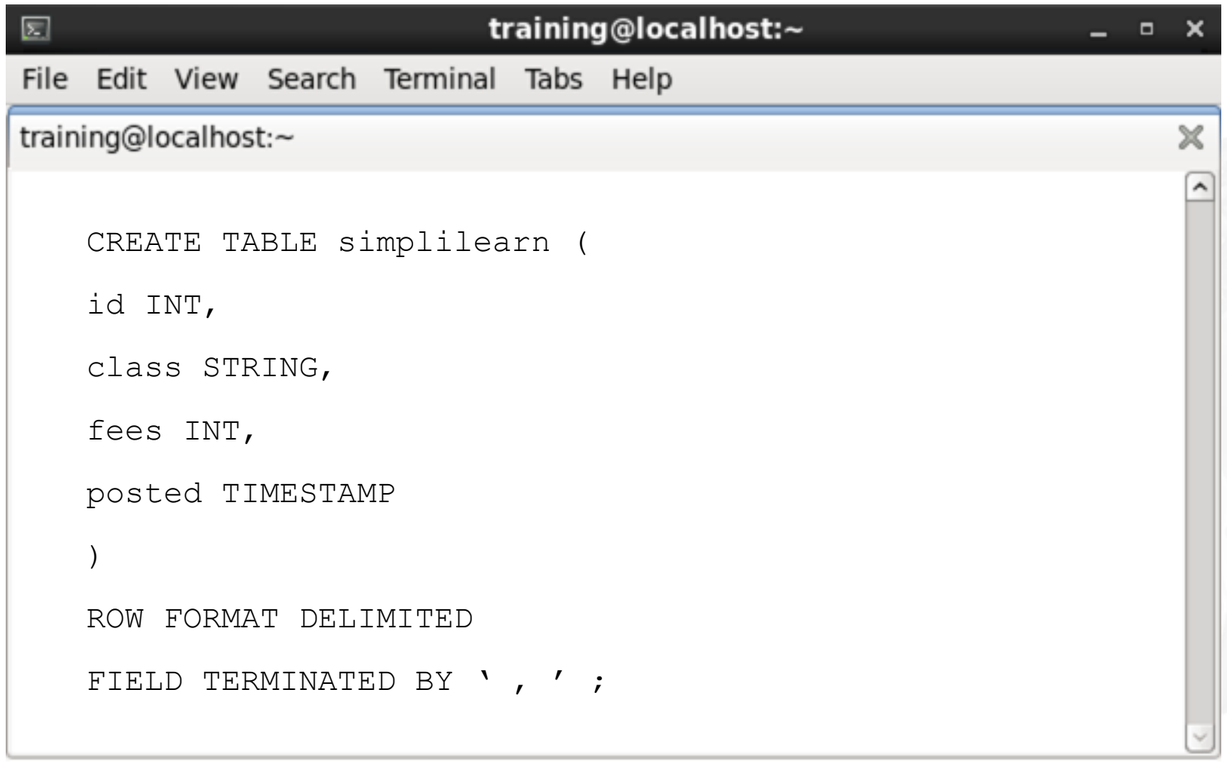
- Syntax creates a subdirectory in the database's warehouse directory in HDFS
  - Default database

    /user/hive/warehouse/tablename
  - Named database

    /user/hive/warehouse/dbname.db/tablename

Table Creation: Example

- The following example shows how to create a new table named simplilearn
  - Data is stored as text with four comma-separated fields per line

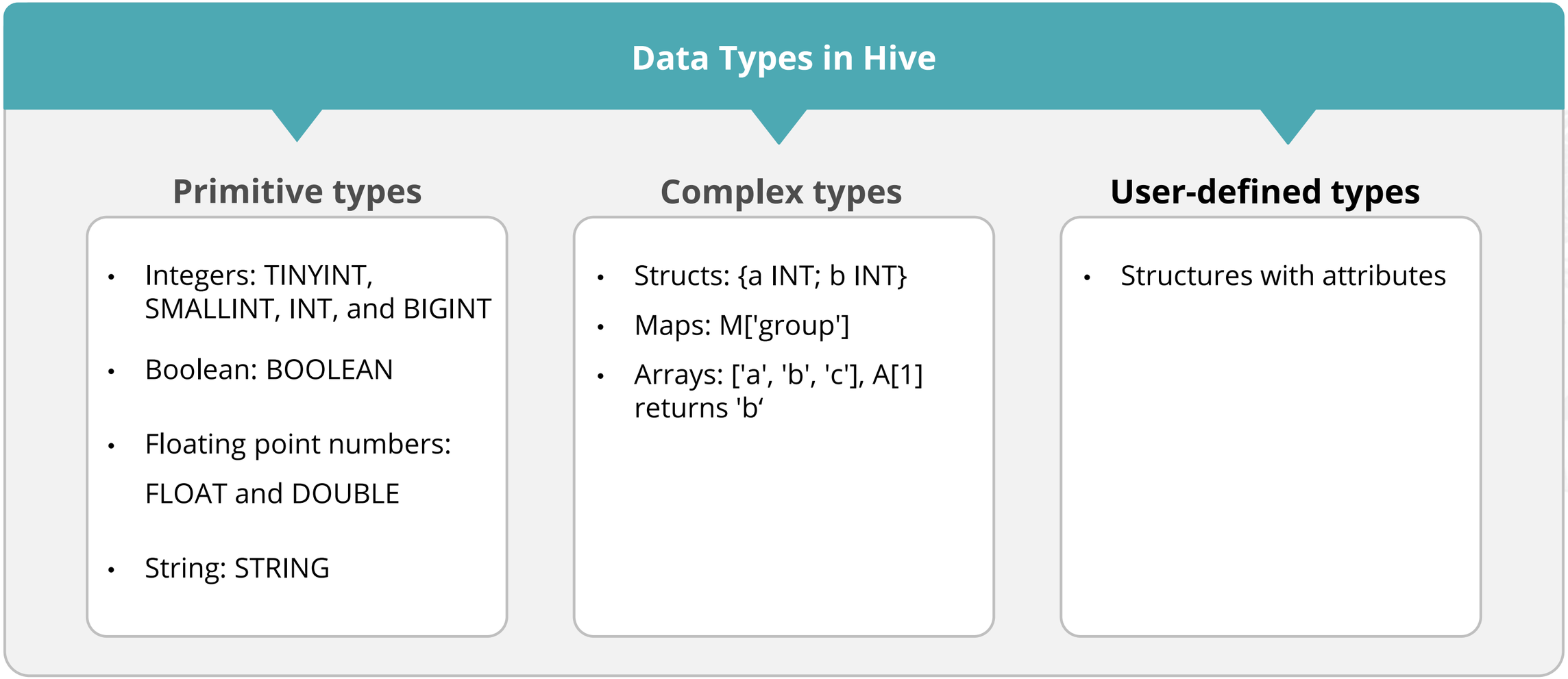


DATA AND
ARTIFICIAL INTELLIGENCE

Data Types

Data Types in Hive

The data types in Hive are as follows:

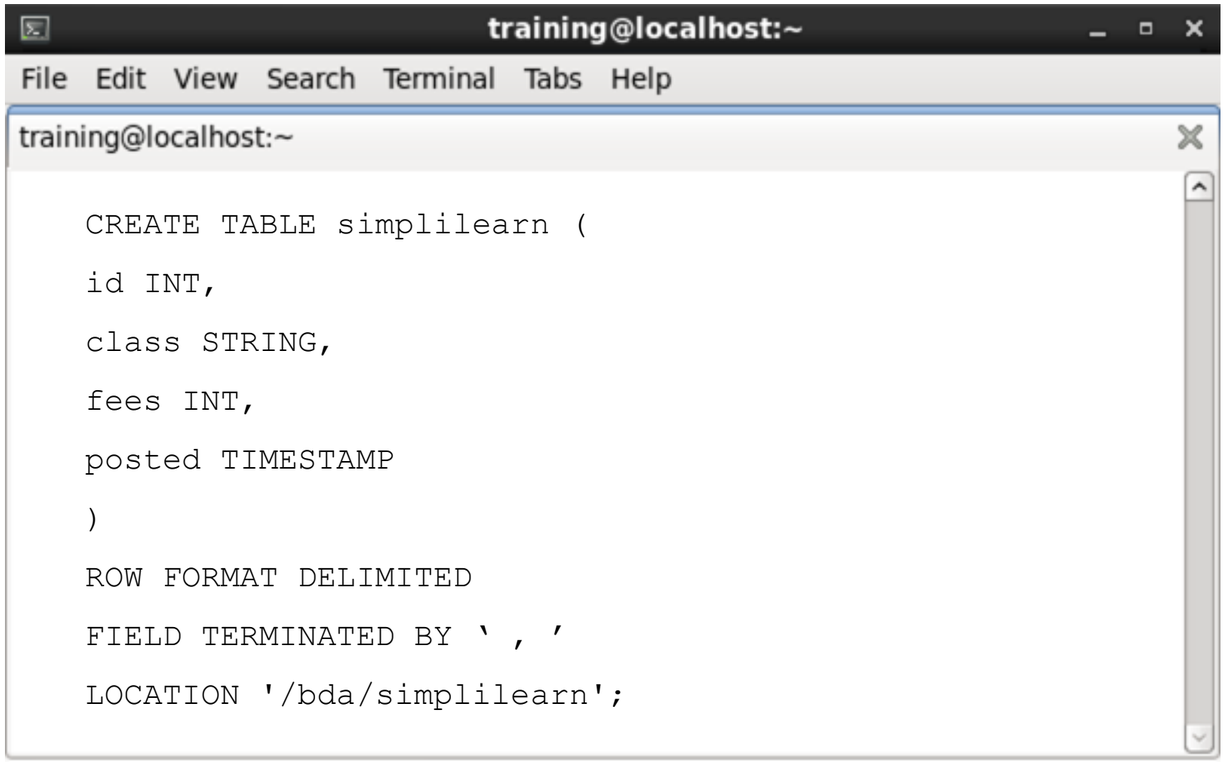


Changing Table Data Location

- By default, table data is stored in the default warehouse location

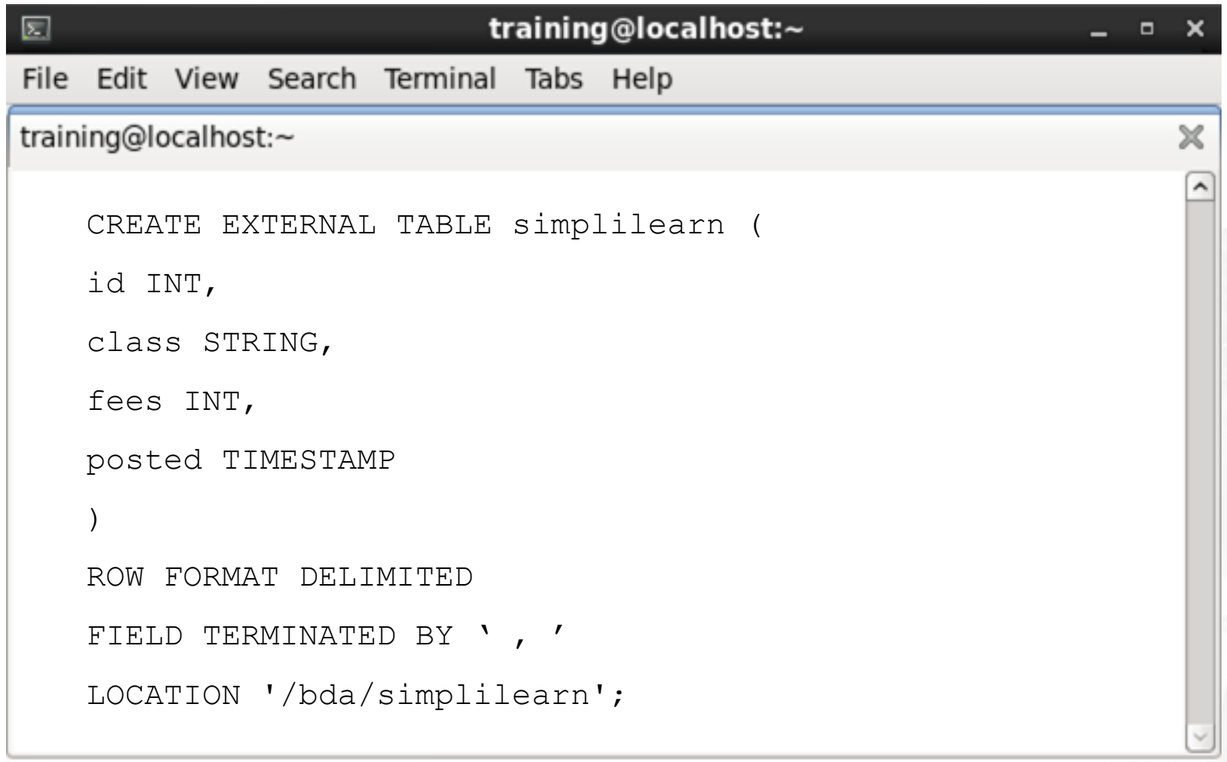
  user/hive/warehouse

- Use LOCATION to specify the directory where you want to reside your data in HDFS



External Managed Table

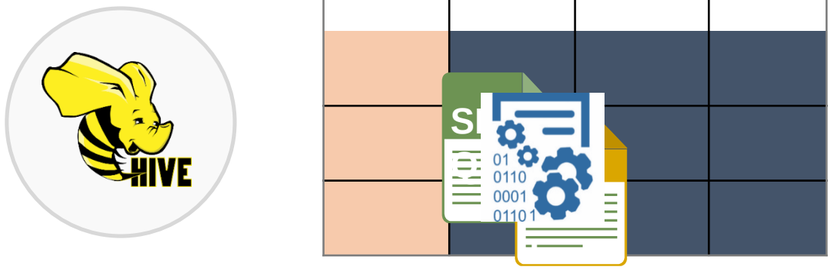
- Tables are “managed” or “internal” by default. When a table is removed, the data also gets deleted.
- Use EXTERNAL to create an external managed table
- Dropping an external table removes only its metadata



Validation of Data

Hive follows “schema on read”

- Unlike RDBMS, Hive does not validate data on insert
- Files are simply moved into place, which makes loading data into tables faster in Hive
- Errors in file formats are discovered when queries are performed



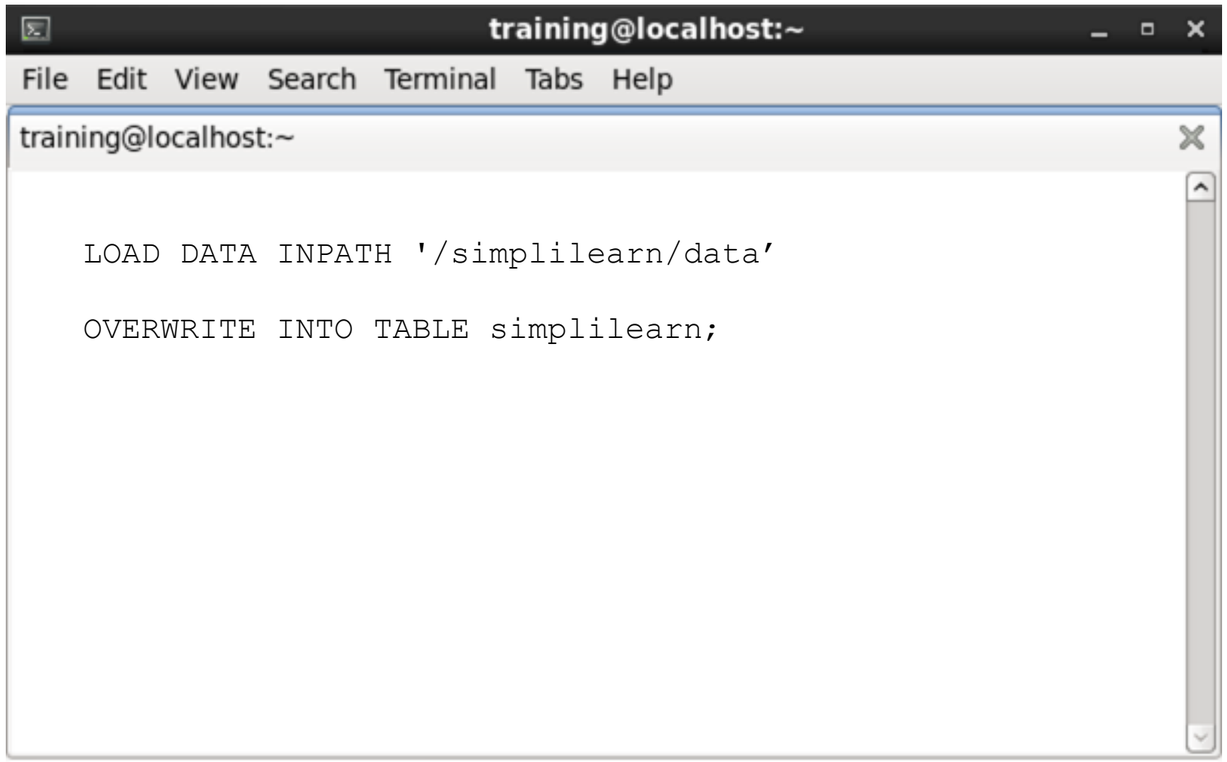
Missing data is represented as NULL.

Loading of Data

- Data can be moved from the HDFS file directly to Hive table

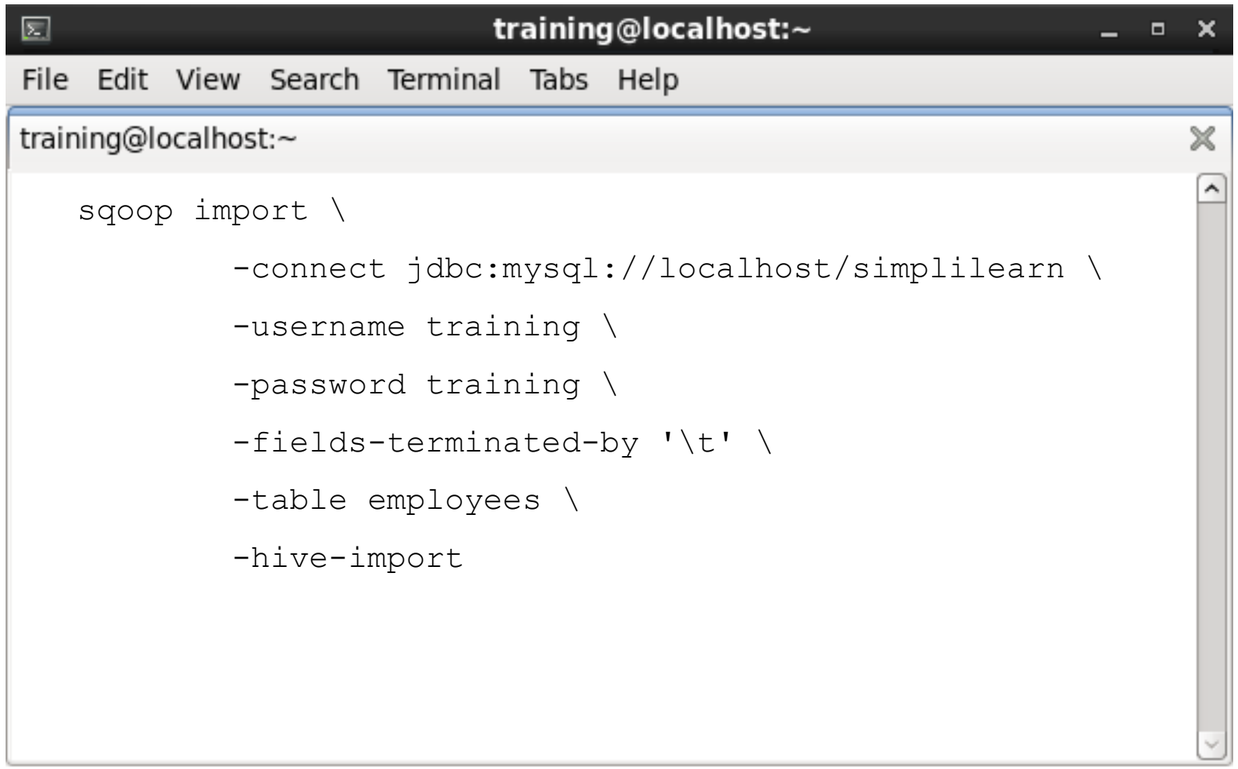
  hdfs dfs -mv /simplilearn/data /user/hive/warehouse/simplilearn/

- Data can be loaded using the following query:



Loading Data from RDBMS

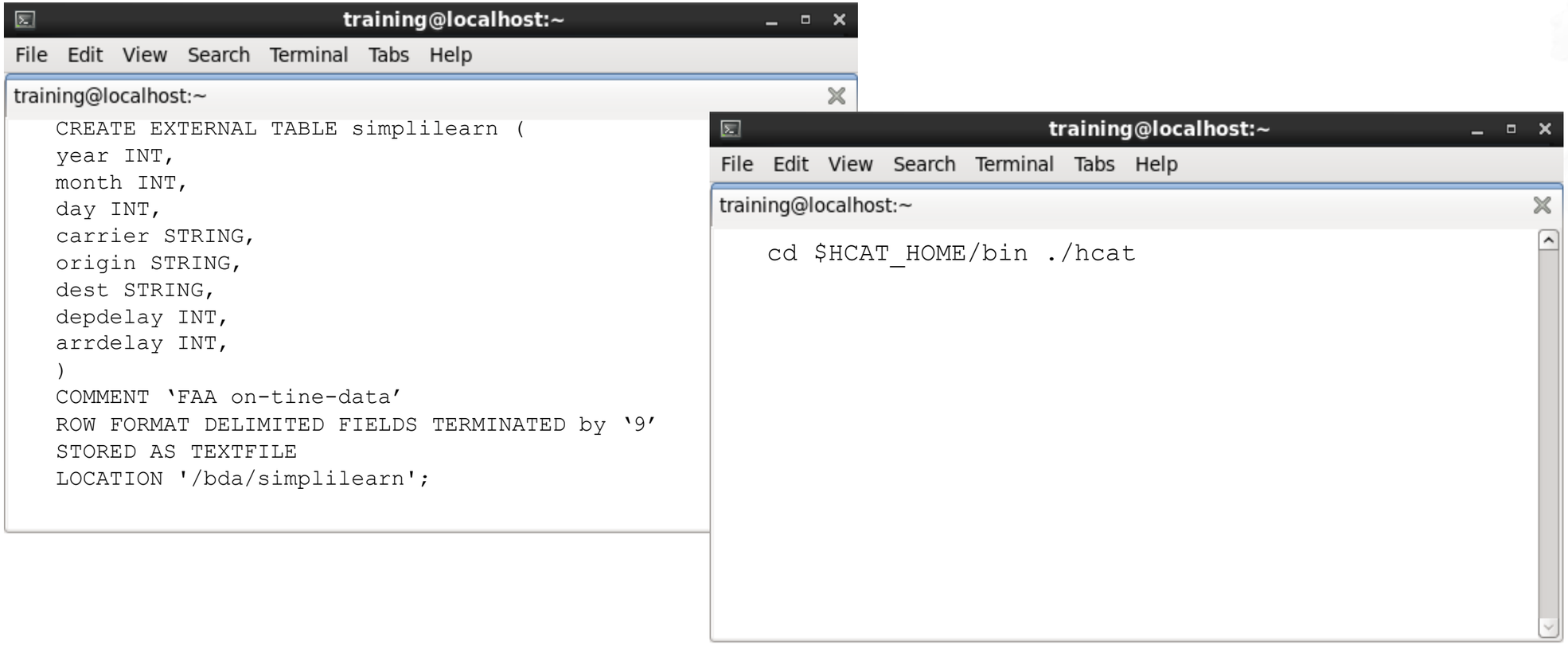
- Sqoop provides support for importing data into Hive
- Using hive-import option in Sqoop, you can:
  - create a table in Hive metastore
  - import data from the RDBMS to the table's directory in HDFS



hive-import creates a table accessible in Hive.

What Is HCatalog

- HCatalog is a Hive sub-project that provides access to the Metastore
- It allows to define tables using HiveQL DDL syntax
- It is accessible through command line and REST API
- It accesses tables created through HCatalog from Hive, MapReduce, Pig, and other tools



Assisted Practice

Apache Hive

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to use Hive query editor for real-time analysis and data filtrations.

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**.

Assisted Practice

Apache Hive

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to use the Hive editor in web console.

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**.

Assisted Practice

Apache Hive

Duration: 15 mins

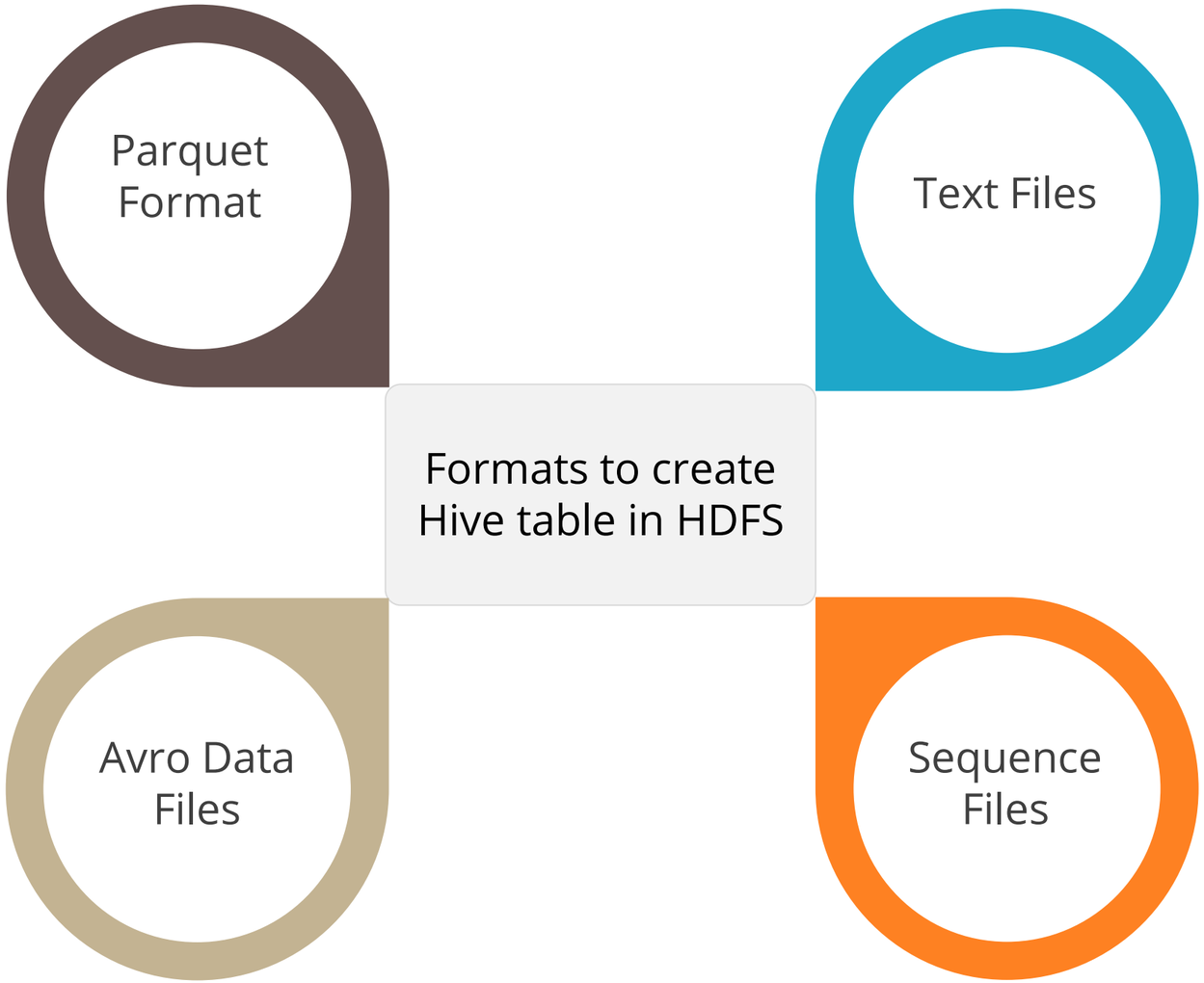
Problem Statement: In this demonstration, you will learn how to use Hive to import data from an external source and perform data representation and analysis.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

File Format Types

File Format Types



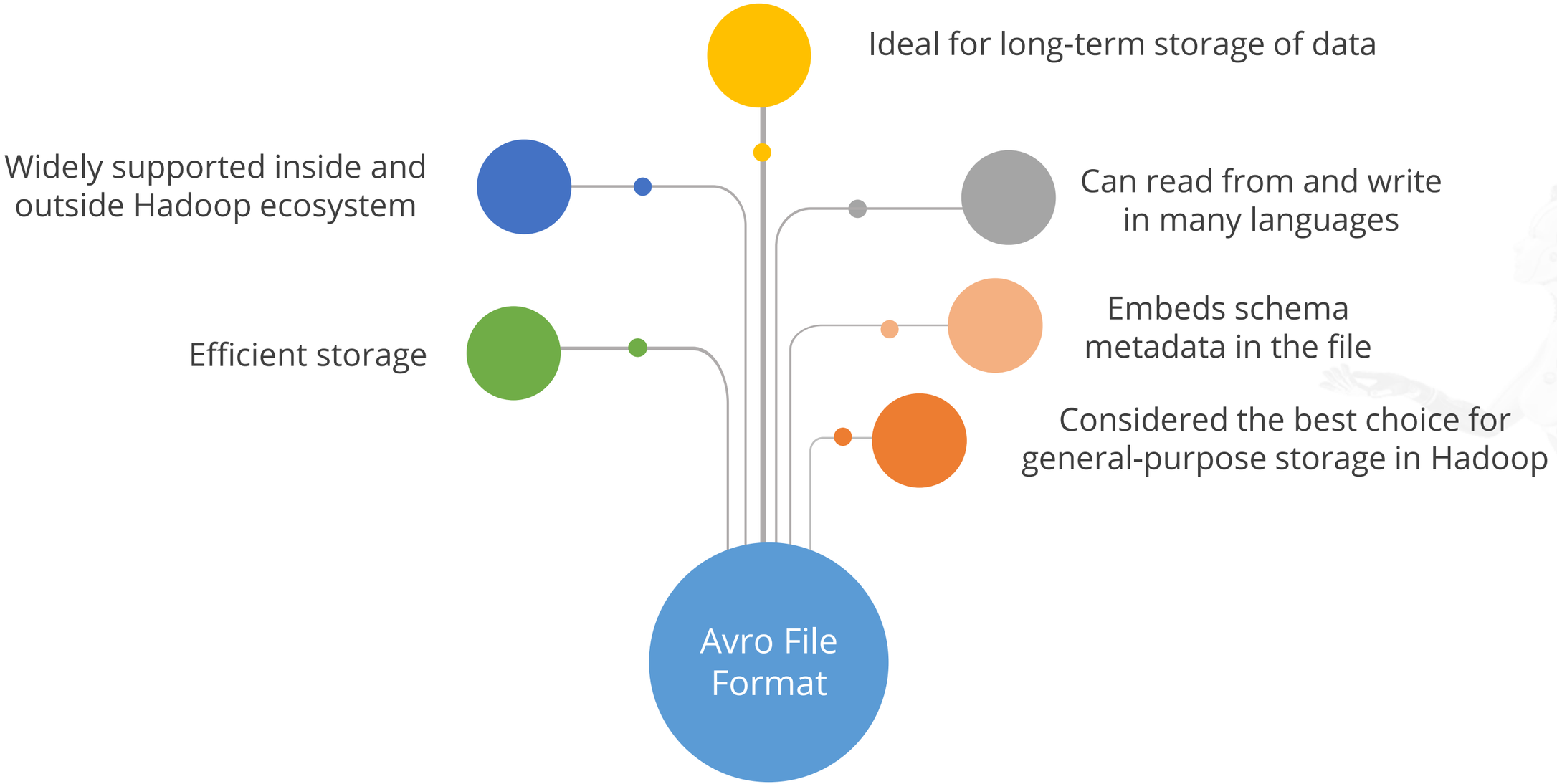
File Format: Text File Format

- The most basic and human-readable file
- Read or written in any programming language
- Delimited by a comma or a tab
- Consumes more space to store numeric value as string
- Difficult to represent binary data

File Format: Sequence File Format

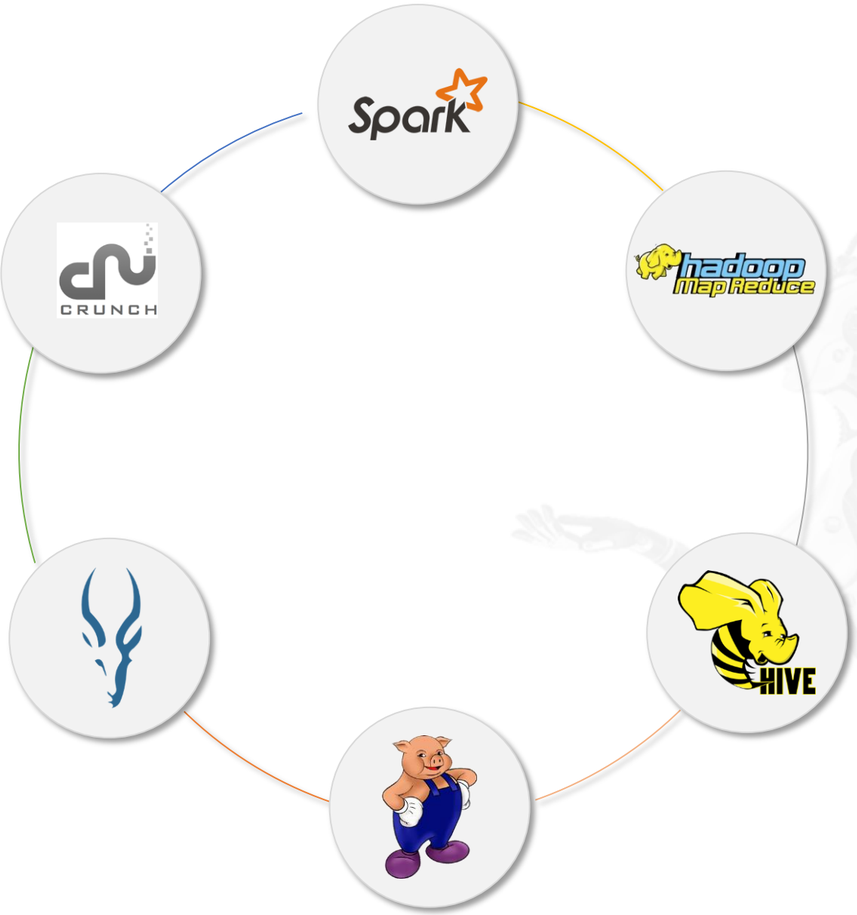
- Stores key-value pairs in a binary container format
- More efficient than a text file
- Not human-readable

File Format: Avro File Format



File Format: Parquet File Format

- Is a columnar format developed by Cloudera and Twitter
- Uses advanced optimizations described in Google's Dremel paper
- Considered the most efficient for adding multiple records at a time



DATA AND
ARTIFICIAL INTELLIGENCE

Data Serialization

What Is Data Serialization?

Data serialization is a way to represent data in the storage memory as a series of bytes.

How do you serialize the number 123456789?

It can be serialized as 4 bytes when stored as a Java int and 9 bytes when stored as a Java String.

Data Serialization Framework

- Efficient data serialization framework
- Widely supported throughout Hadoop and its ecosystem
- Supports Remote Procedure Calls (RPC)
- Offers compatibility

Data Types Supported in Avro

Name	Description
null	An absence of a value
boolean	A binary
int	32-bit signed integer
long	64-bit signed integer
float	Single-precision floating point value
double	Double-precision floating point value
bytes	Sequence of 8-bit unsigned bytes
string	Sequence of unicode characters

Complex Data Types Supported in Avro Schemas

Name	Description
record	A user-defined type composed of one or more named fields
enum	A specified set of values
array	Zero or more values of the same type
map	Set of key-value pairs; key is string while value is of specified type
union	Exactly one value matching a specified set of types
fixed	A fixed number of 8-bit unsigned bytes

Hive Table and Avro Schema

Hive Table - CREATE TABLE orders

```
(id INT, name STRING, title STRING)
```

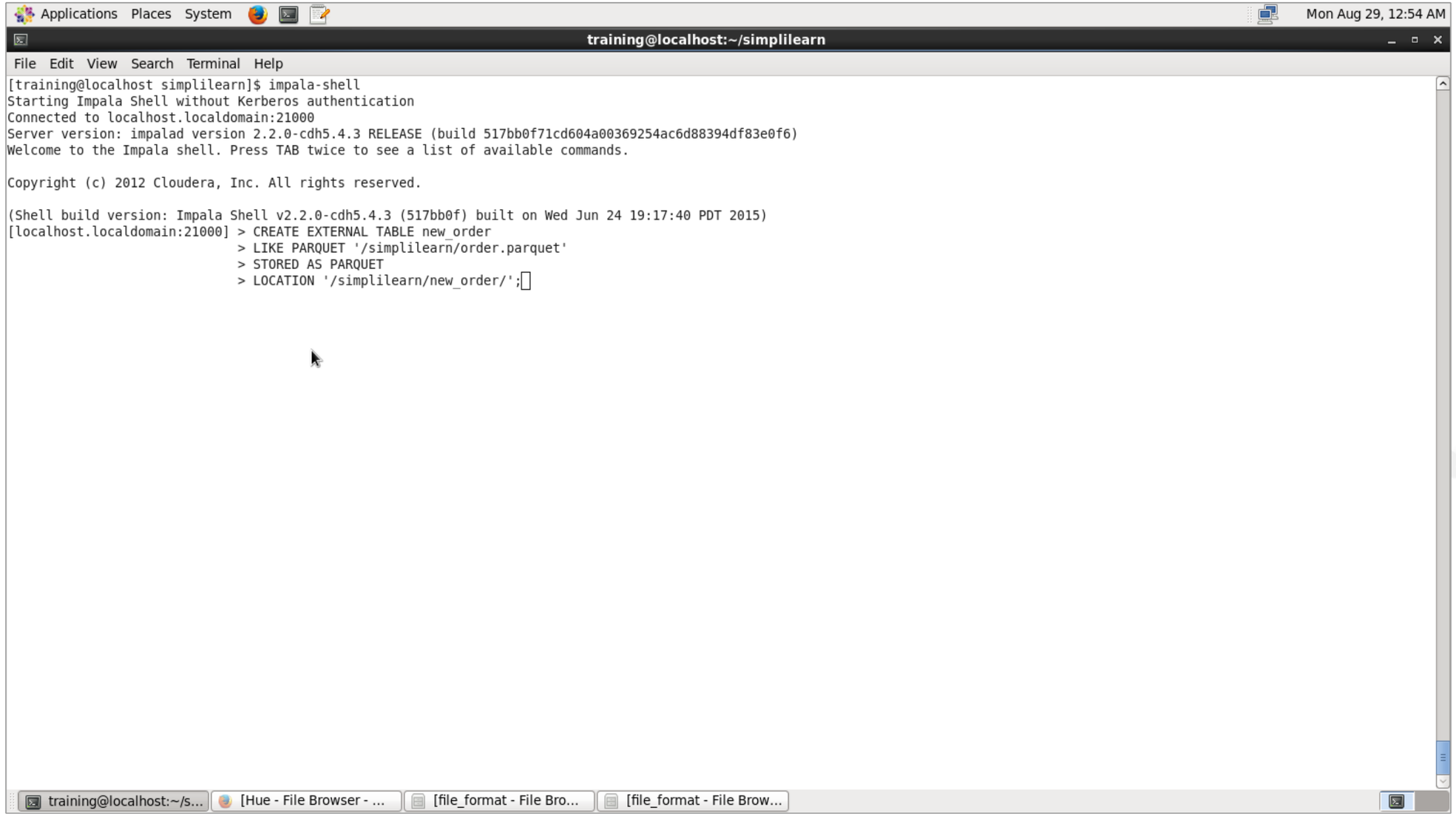
Avro Schema

```
{"namespace":"com.simplilearn",
"type":"record",
"name":"orders",
"fields":[
{"name":"id", "type":"int"},
{"name":"name", "type":"string"},
{"name":"title", "type":"string"}]
}
```

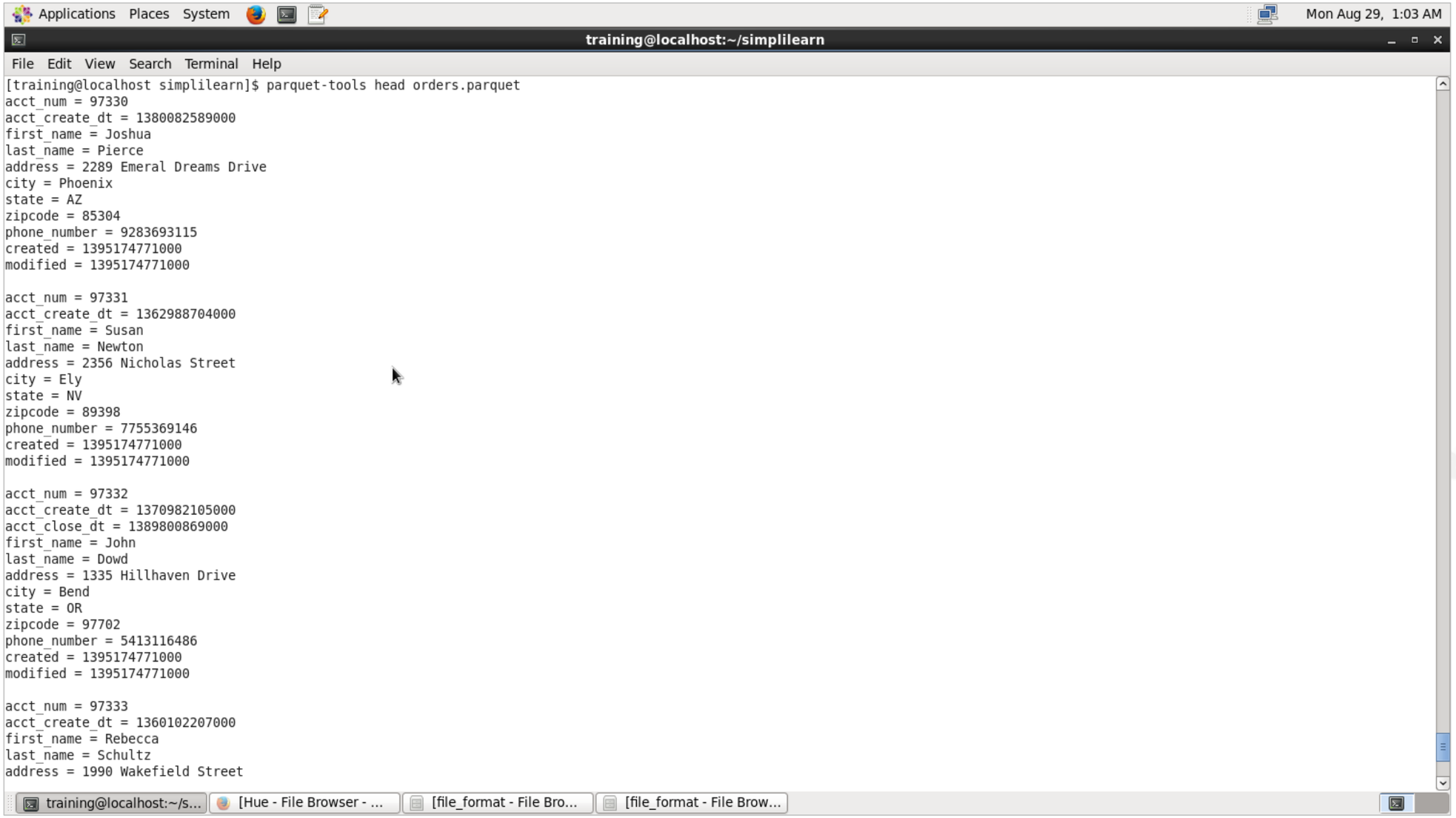
Other Avro Operations

```
{"namespace":"com.simplilearn",
"type":"record",
"name":"orders",
"fields":[
{"name":"id", "type":"int"},
{"name":"name", "type":"string", "default":"simplilearn"},
{"name":"title", "type":"string","default":"bigdata"}]
}
```

Create New Table with Parquet



Reading Parquet Files Using Tools

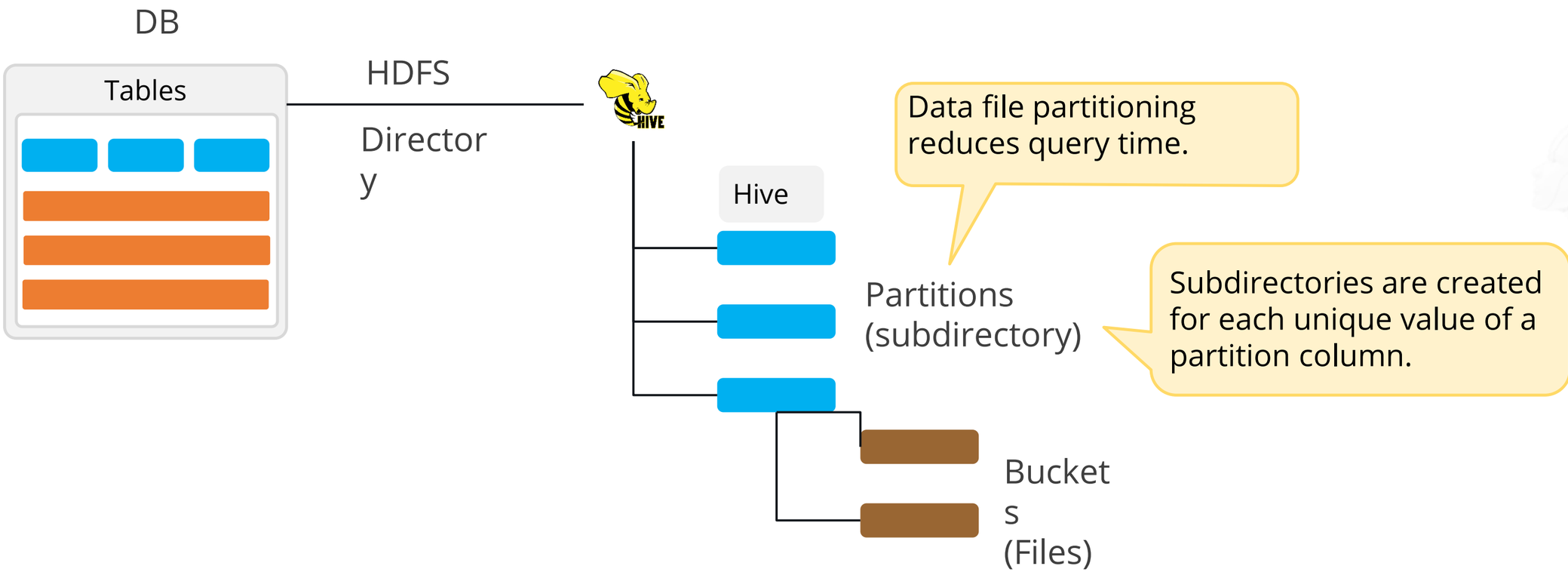


DATA AND
ARTIFICIAL INTELLIGENCE

Hive Optimization: Partitioning, Bucketing, and Sampling

Data Storage

All files in a data set are stored in a single Hadoop Distributed File System or HDFS directory.



Example of a Non-Partitioned Table

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE EXTERNAL TABLE accounts(
    > cust_id INT,
    > fname STRING,
    > lname STRING,
    > address STRING,
    > city STRING,
    > state STRING,
    > zipcode STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/simplilearn/accounts';
```

The customer details are required to be partitioned by state for fast retrieval of subset data pertaining to the customer category.

Hive will need to read all the files in a table's data directory.

Can be a very slow and expensive process, especially when the tables are large.

Example of a Partitioned Table

```
training@localhost:~
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE EXTERNAL TABLE accounts_by_state(
    > cust_id INT,
    > fname STRING,
    > lname STRING,
    > address STRING,
    > city STRING,
    > zipcode STRING)
    > PARTITIONED BY (state STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/simplilearn/accounts_by_state';
```

A partition column is a "virtual column" where data is not actually stored in the file.

Partitions are horizontal slices of data that allow larger sets of data to be separated in more manageable chunks.

Use partitioning to store data in separate files by state.

DATA AND
ARTIFICIAL INTELLIGENCE

Data Insertion

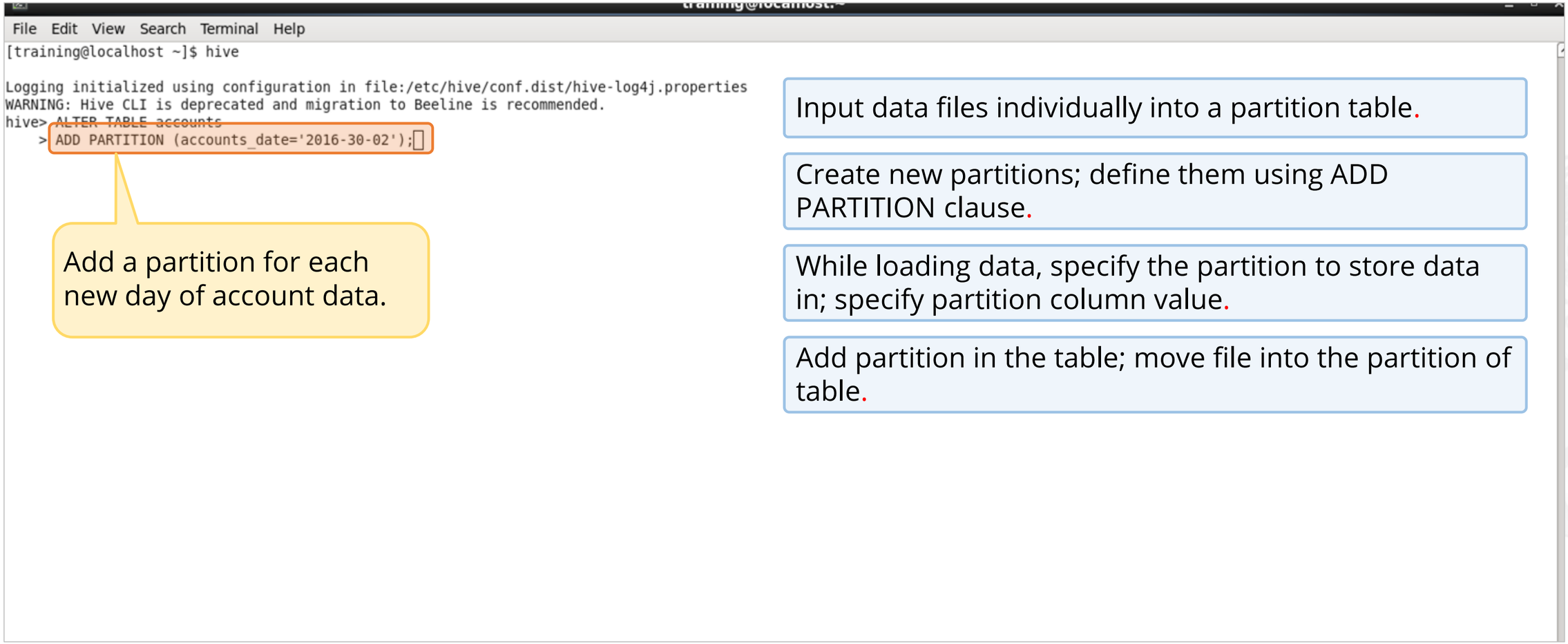
Data Insertion

Data insertion into partitioned tables can be done in two ways or modes:

Static partitioning

Dynamic partitioning

Static Partitioning



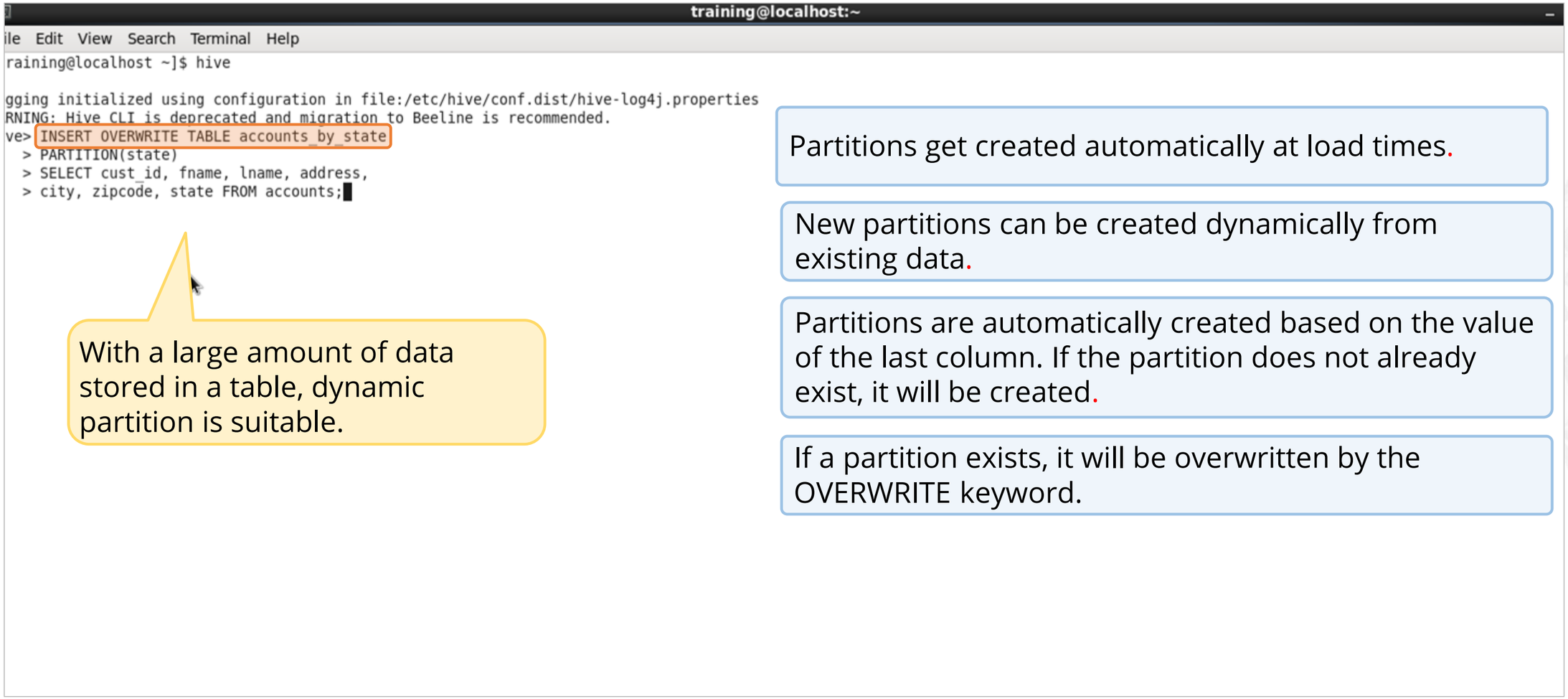
Input data files individually into a partition table.

Create new partitions; define them using ADD PARTITION clause.

While loading data, specify the partition to store data in; specify partition column value.

Add partition in the table; move file into the partition of table.

Dynamic Partitioning



With a large amount of data stored in a table, dynamic partition is suitable.

Partitions get created automatically at load times.

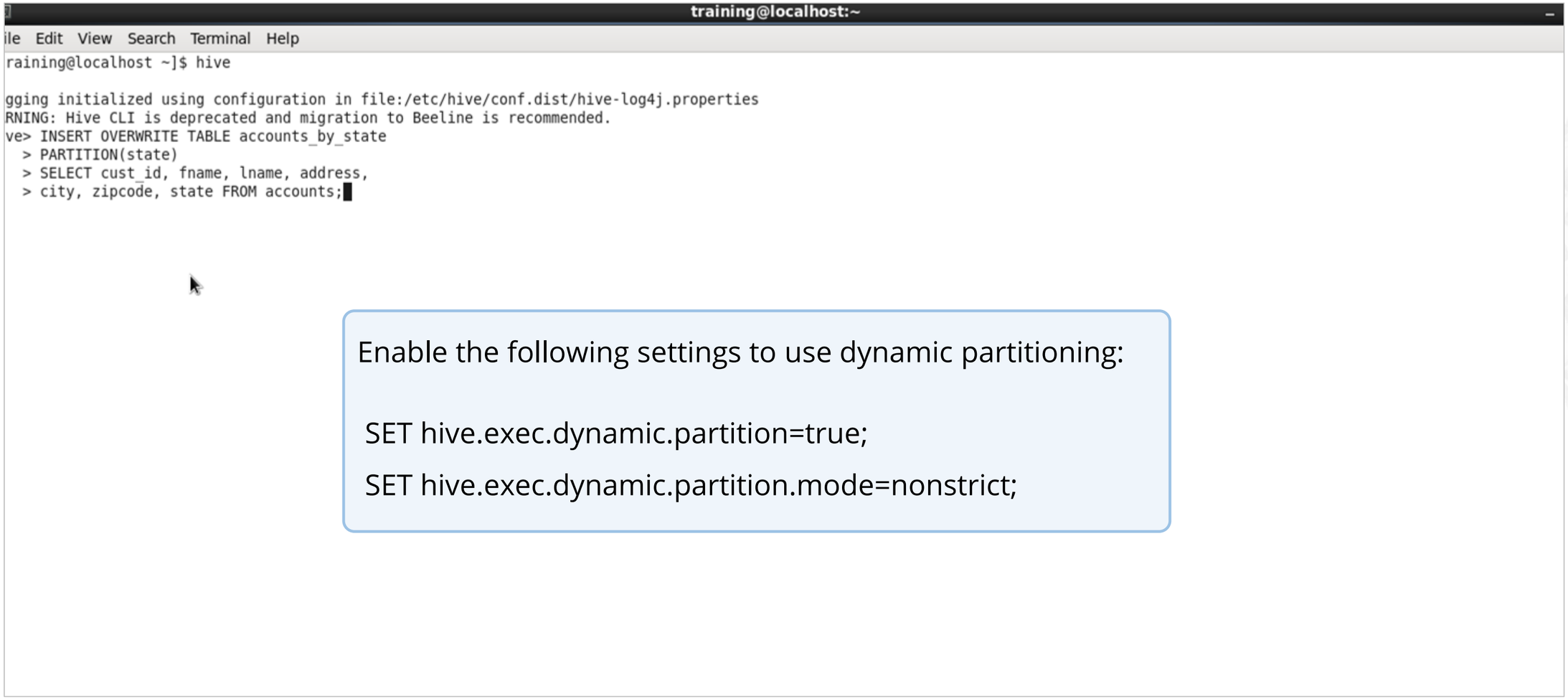
New partitions can be created dynamically from existing data.

Partitions are automatically created based on the value of the last column. If the partition does not already exist, it will be created.

If a partition exists, it will be overwritten by the OVERWRITE keyword.

Dynamic Partitioning in Hive

By default, dynamic partitioning is disabled in Hive to prevent accidental partition creation.



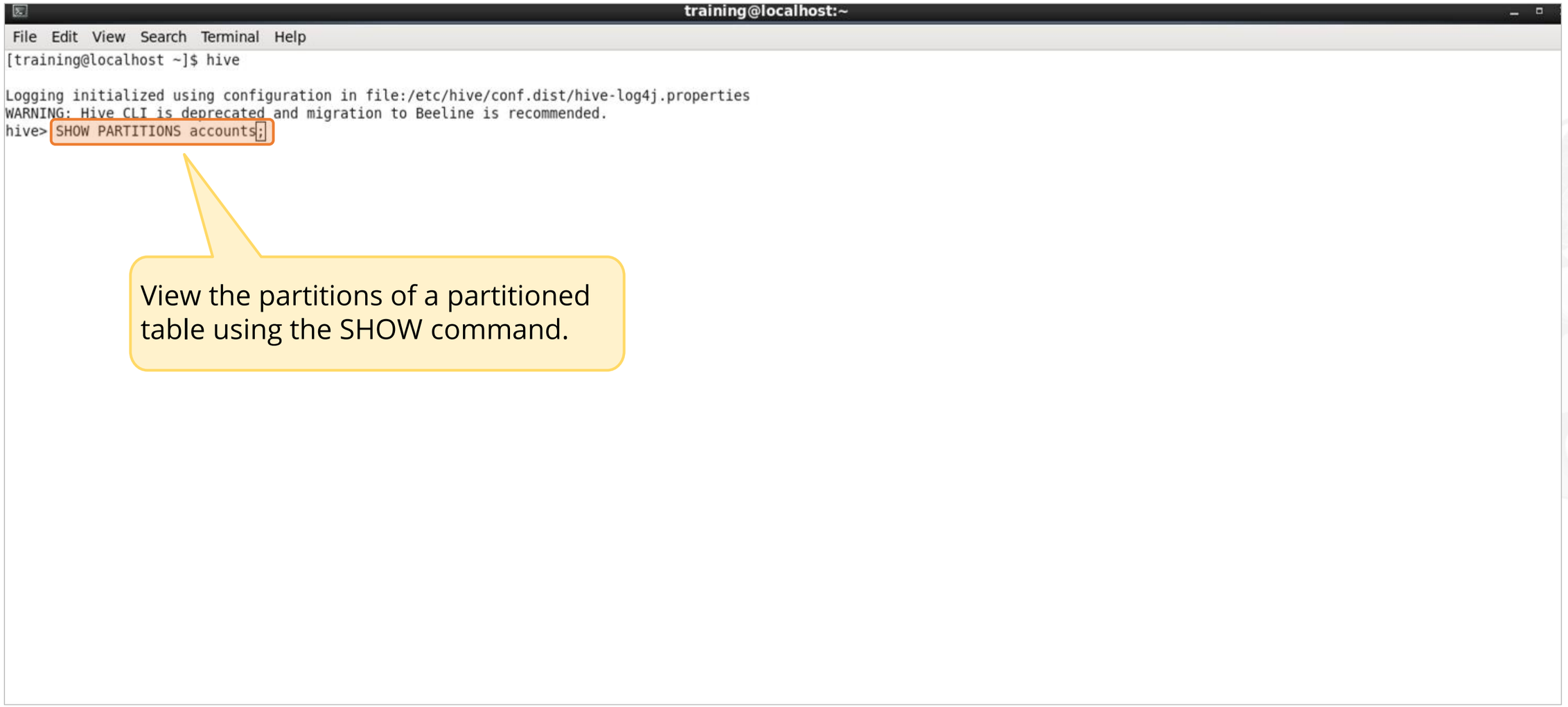
Enable the following settings to use dynamic partitioning:

SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode=nonstrict;

Viewing Partitions

Commands that are supported on Hive partitioned tables to view and delete partitions.



Deleting Partitions

Use ALTER command:

- To delete partitions
- To add or change partitions

When to Use Partitioning

Following are the instances when you need to use partitioning for tables:

When reading the entire data set takes too long

When queries almost always filter on the partition columns

When there are a reasonable number of different values for partition columns

When Not to Use Partitioning

Following are the instances when you should avoid using a partitioning:

When columns have too many unique rows

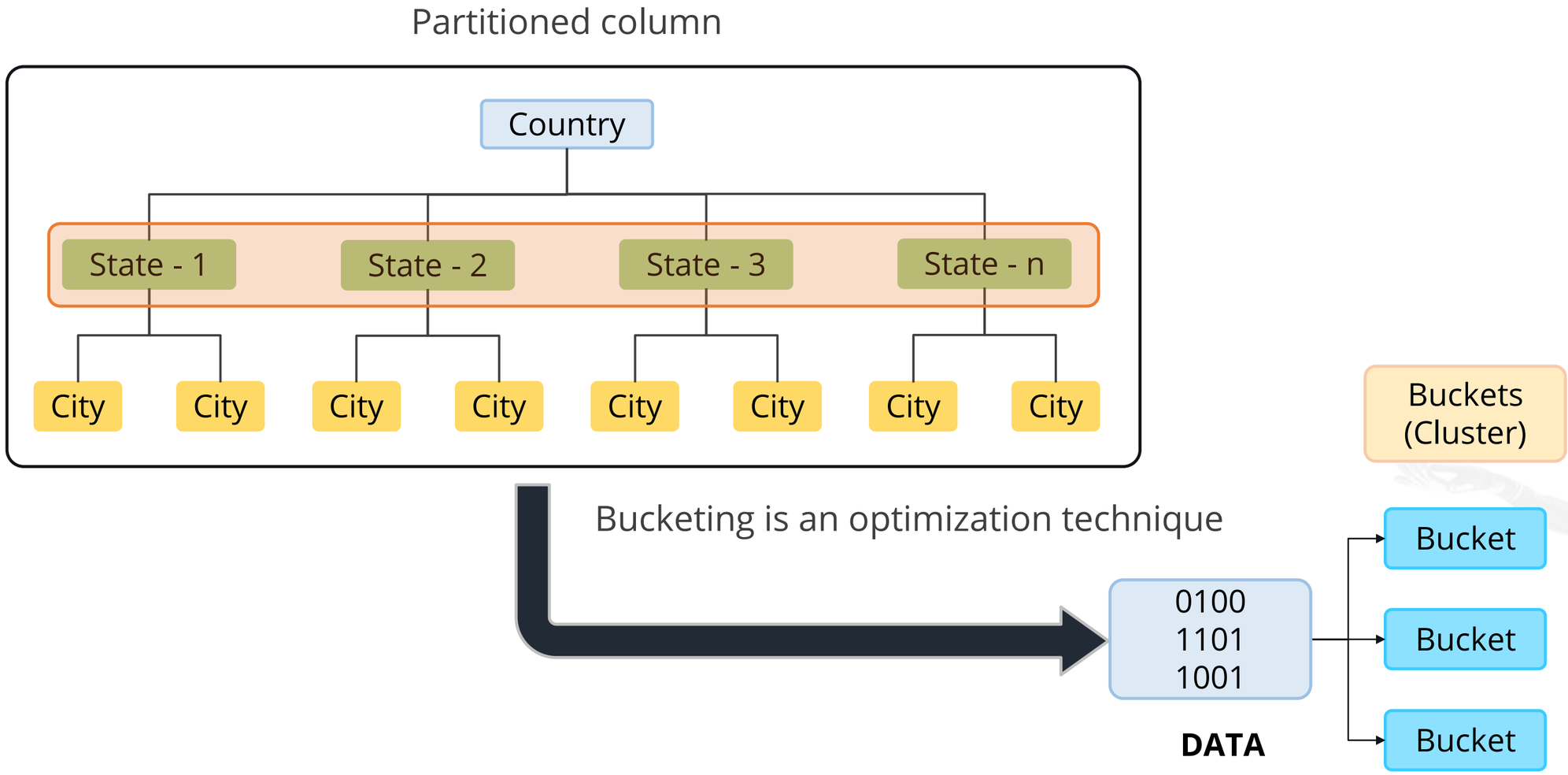
When creating a dynamic partition as it can lead to high number of partitions

When the partition is less than 20k

DATA AND
ARTIFICIAL INTELLIGENCE

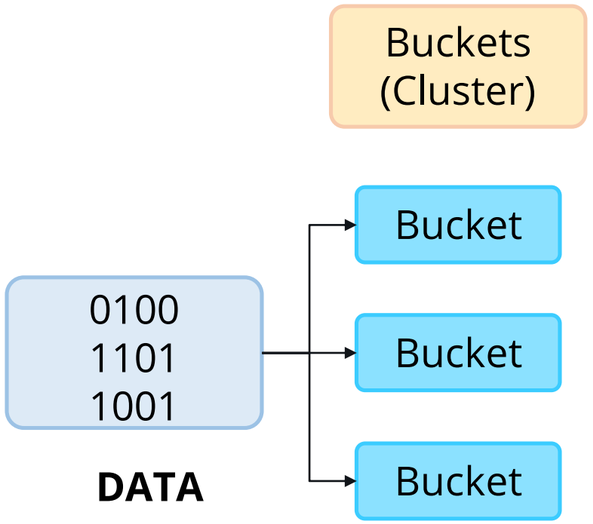
Bucketing

Bucketing in Hive



What Do Buckets Do?

Buckets distribute the data load into user-defined set of clusters by calculating the hash code of the key mentioned in the query.



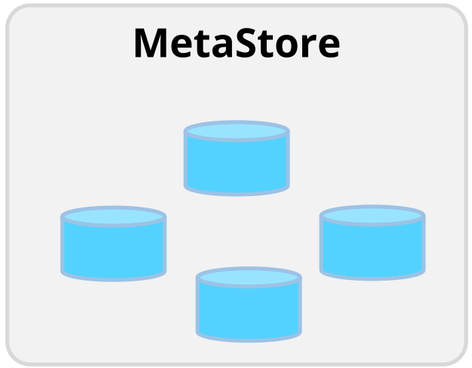
Syntax for creating a bucketed table

```
CREATE TABLE page_views( user_id INT, session_id BIGINT, url
STRING)
PARTIONED BY (day INT)
CLUSTERED BY (user_id) INTO 100;
```

The processor will first calculate the hash number of the user_id in the query and will look for only that bucket.

Hive Query Language: Introduction

HiveQL is a SQL-like query language for Hive to process and analyze structured data in a Metastore.




```
SELECT
dt,
COUNT (DISTINCT (user_id))
FROM events
GROUP BY dt;
```

HiveQL: Extensibility

An important principle of HiveQL is its extensibility. HiveQL can be extended in multiple ways:

- Pluggable user-defined functions
- Pluggable data formats
- Pluggable MapReduce scripts
- Pluggable user-defined types

DATA AND
ARTIFICIAL INTELLIGENCE

Hive Analytics: UDF and UDAF

User-Defined Function

Hive has the ability to define a function. UDFs extend the functionality of Hive, with a function written in Java, that can be evaluated in HiveQL statements.

All UDFs extend the Hive UDF class. After that, a UDF sub-class implements one or more methods named ‘evaluate’.

Evaluate should never be a void method. It can return null value, if required.

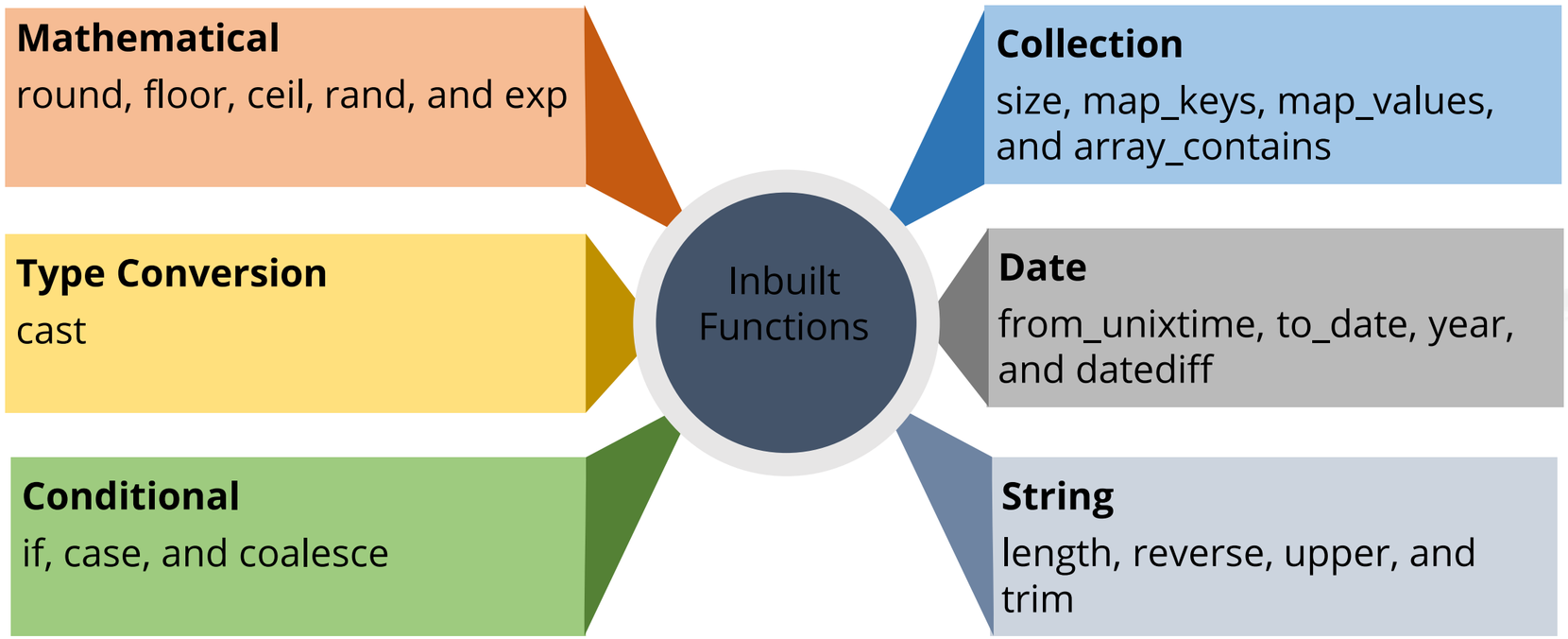
Code for Extending UDF

Here is a code that you can use to extend the User-Defined Function.

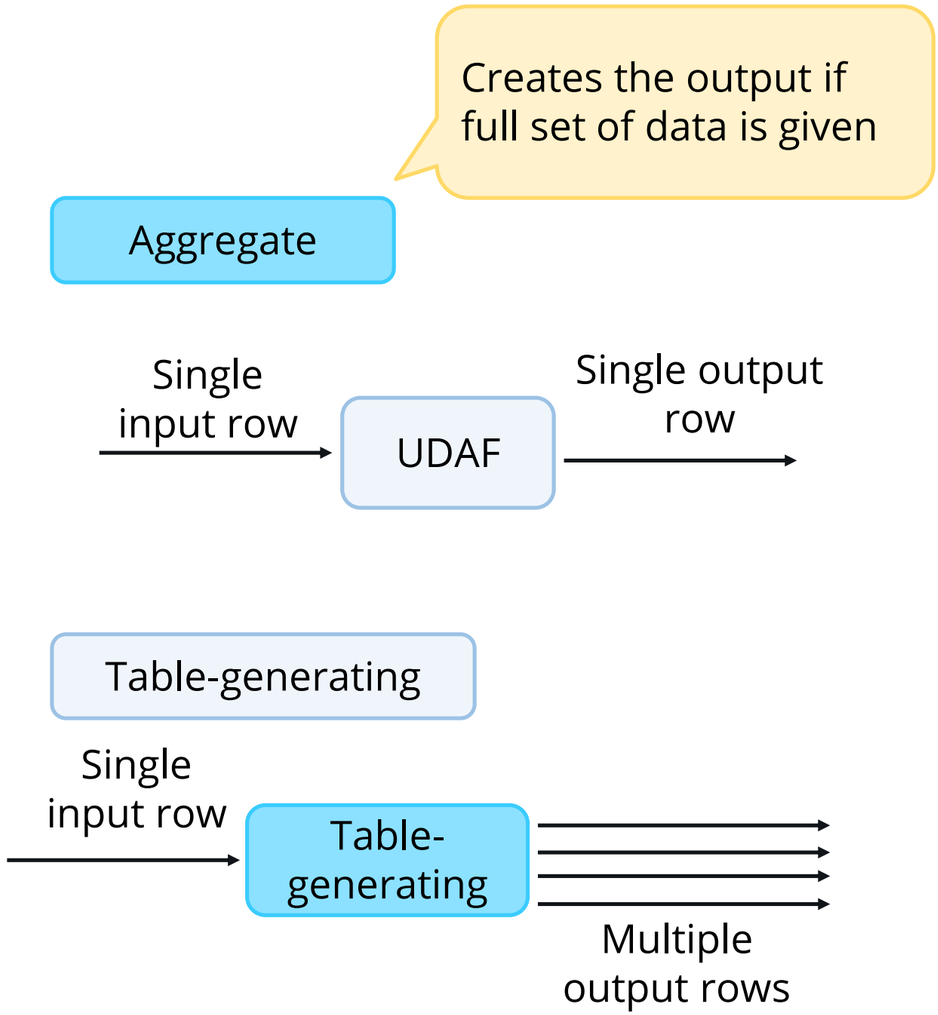
```
package com.example.hive.udf;

import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public final class Lower extends UDF {
  public Text evaluate(final Text s) {
    if (s == null) { return null; }
    return new Text(s.toString().toLowerCase());
  }
}
```

Built-in Functions of Hive

Writing the functions in JAVA scripts creates its own UDF. Hive also provides some inbuilt functions that can be used to avoid own UDFs from being created.



Other Functions of Hive



Lateral view:

String pageid	Array<int> adid_list
"front_page"	[1,2,3]
"contact_page"	[3, 4, 5]

```
SELECT pageid, adid FROM pageAds
LATERAL VIEW explode(adid_list) adTable
AS adid;
```

String pageid	intadid
"front_page"	1
"front_page"	2
.....	.....

MapReduce Scripts

MapReduce scripts are written in scripting languages, such as Python.

Pluggable data formats

Pluggable user-defined functions

Pluggable MapReduce scripts

Pluggable user-defined types

Example: my_append.py

```
for line in sys.stdin:
    line = line.strip()
    key = line.split('\t')[0]
    value = line.split('\t')[1]
    print key+str(i)+'\t'+value+str(i)
    i=i+1
```

Using the function:

```
SELECT TRANSFORM (foo, bar) USING 'python ./my_append.py' FROM sample;
```

UDF/UDAF vs. MapReduce Scripts

Attribute	UDF/UDAF	MapReduce scripts
Language	Java	Any language
1/1 input/output	Supported via UDF	Supported
n/1 input/output	Supported via UDAF	Supported
1/n input/output	Supported via UDTF	Supported
Speed	Faster (in same process)	Slower (spawns new process)

Key Takeaways

You are now able to:

- Define Hive and its architecture
- Create and manage tables using Hue Web UI and Beeline
- Understand various file formats supported in Hive
- Use HiveQL DDL to create tables and execute queries

Quiz

Knowledge Check

1

Deleting an individual record is possible in_____ .

a. Hive

b. RDBMS

c. Both A and B

d. None of the above

Knowledge Check

1

Deleting an individual record is possible in_____ .

a. Hive

b. RDBMS

c. Both A and B

d. None of the above

The correct answer is **b.**

Hive cannot delete individual records, but an RDBMS can.

Knowledge Check

2

In which HDFS directory is Hive table created by default?

a. /hive

b. /user/hive/

c. /user/hive/warehouse

d. All of the above

Knowledge Check

2

In which HDFS directory is Hive table created by default?

a. /hive

b. /user/hive/

c. /user/hive/warehouse

d. All of the above

The correct answer is **c.**

Hive table gets created **by default in /user/hive/warehouse in HDFS directory.**

Knowledge Check

3

Which of the following statements is true for sequential file format?

a. More efficient than a text file

b. Store key-value pairs in a binary container format

c. Not human-readable

d. All of the above

Knowledge Check

3

Which of the following statements is true for sequential file format?

a. More efficient than a text file

b. Store key-value pairs in a binary container format

c. Not human-readable

d. All of the above

The correct answer is **d.**

Sequential File format stores key-value pairs in a binary container format, is efficient than a text file, and it is not human–readable.

Knowledge Check

4

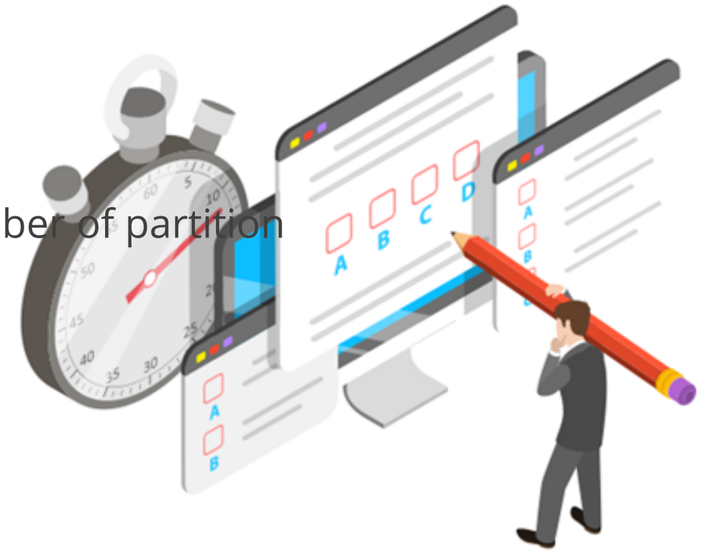
Which of the following statements is true about when not to use partitions?

a. Try to limit partition to less than 20k

b. Avoid partition on columns having too many unique rows

c. Be cautious while creating dynamic partition as it can lead to high number of partition

d. All of the above



Knowledge Check

4

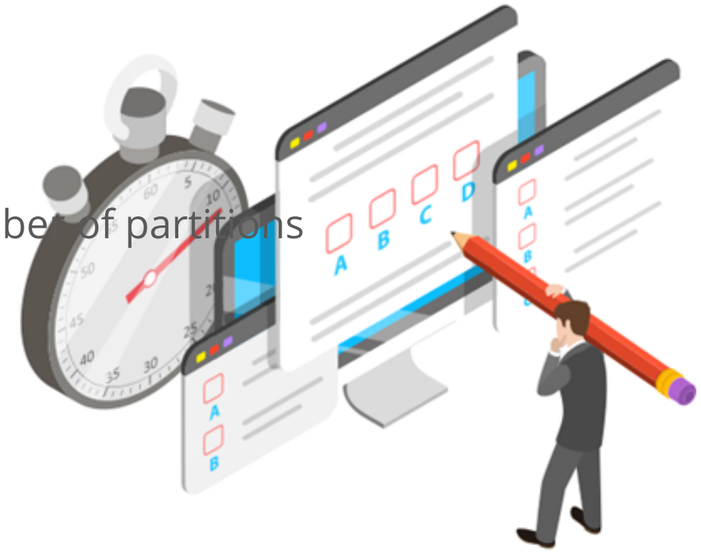
Which of the following statements is true about when not to use partitions?

a. Try to limit partition to less than 20k

b. Avoid partition on columns having too many unique rows

c. Be cautious while creating dynamic partition as it can lead to high number of partitions

d. All of the above



The correct answer is **d.**

We should avoid using partitions when columns have too many unique rows, while creating dynamic partition as it can lead to high number of partitions, and when limiting partition to less than 20k.

Knowledge Check

5

Which of the following is a way of representing data in memory as a series of bytes?

a. File Formatting

b. Data Serialization

c. Both A and B

d. None of the above

Knowledge Check

5

Which of the following is a way of representing data in memory as a series of bytes?

a. File Formatting

b. Data Serialization

c. Both A and B

d. None of the above

The correct answer is **b.**

Data Serialization is a way of representing data in memory as a series of bytes.

Lesson-End Project

Problem Statement:

Everybody loves movies. Nowadays, movie releases per year has increased compared to earlier days because of an increase in the number of production houses. A few giants, like Netflix and Amazon, have started creating their content as well.

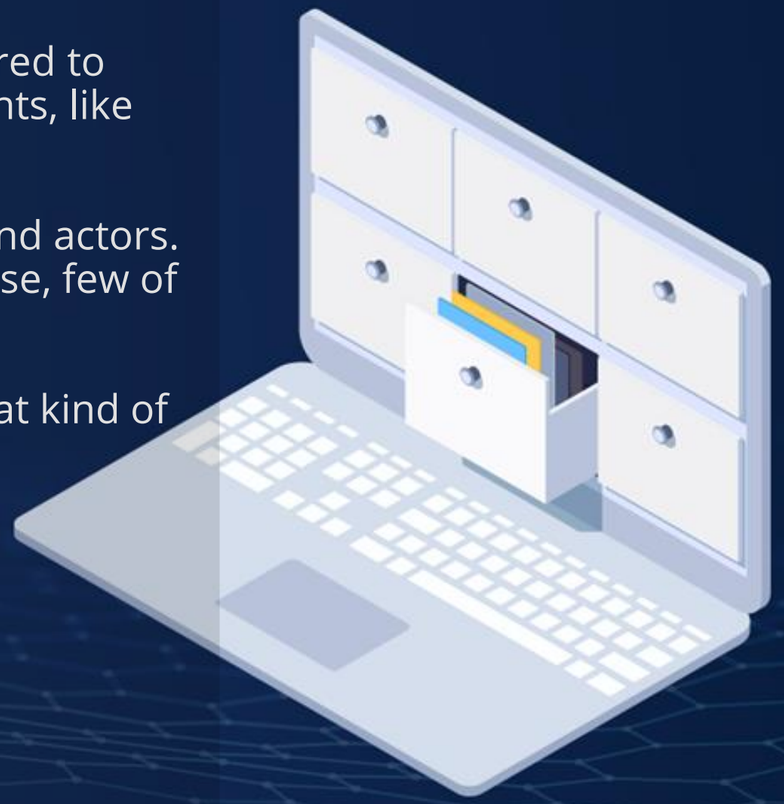
Hollywood is spreading its wings in most countries because of its graphics, story, and actors. In Hollywood, few directors have made great impact among audiences. Among these, few of them have received nominations and won awards.

Before watching a movie, people tend to validate the director's credentials like, what kind of movies he has made in the past and if he has won any awards.

The given data set has details about the movie directors and whether they have received nominations and won awards.

The dataset contains the following fields:

1. Director name
2. Ceremony
3. Year
4. Category
5. Outcome
6. Original language



Lesson-End Project

Find out the below insights:

1. Directors who were nominated and have won awards in the year 2011
2. Award categories available in the Berlin International Film Festival
3. Directors who won awards for making movies in French
4. Directors who have won awards more than 10 times

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

NoSQL Databases: HBase

Learning Objectives

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

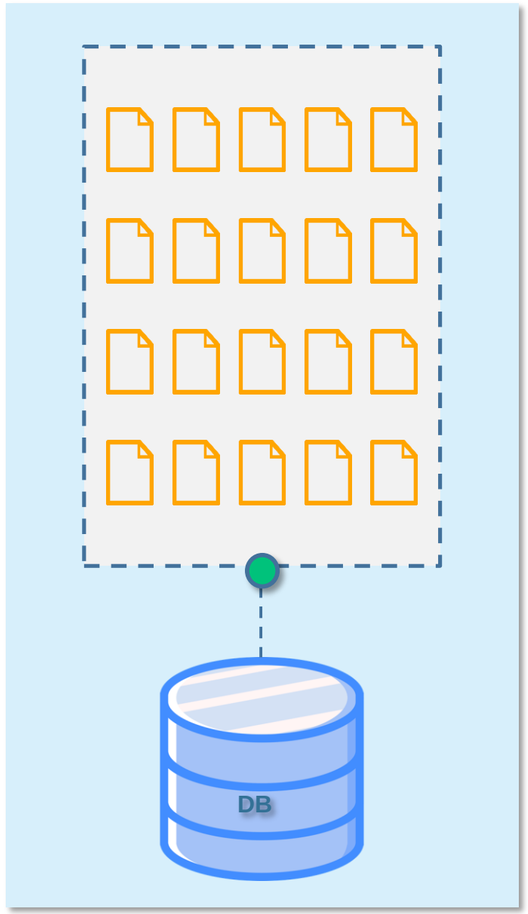
- Understand the need for NoSQL databases
- Analyze the HBase architecture and components
- Differentiate HBase from RDBMS

DATA AND
ARTIFICIAL INTELLIGENCE

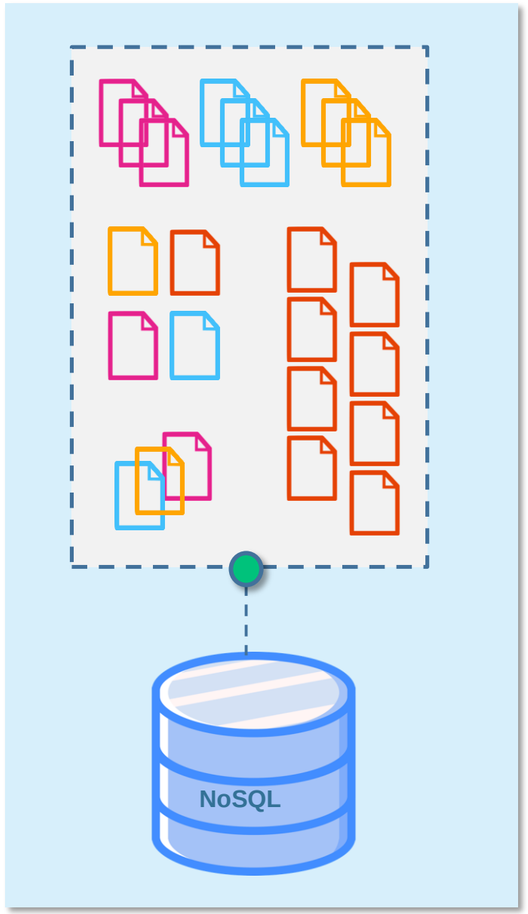
NoSQL Introduction

NoSQL Database

NoSQL is a form of unstructured storage.



Structured



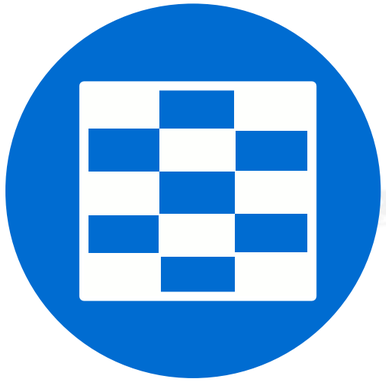
Unstructured

Why NoSQL?

With the explosion of social media sites, such as Facebook and Twitter, the demand to manage large data has grown tremendously.

Key-value pair databases

Document databases



Column-based data stores

Types of NoSQL

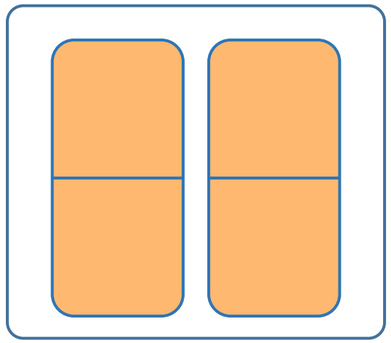
Key-value

Example:
Oracle NoSQL, Redis Server, Scalaris

Document-based

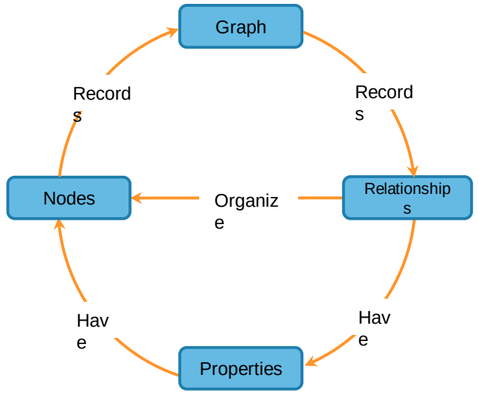
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-based



Example:
BigTable, Cassandra, HBase, Hypertable

Graph-based



Example:
Neo4J, InfoGrid, Infinite Graph, FlockDB

RDBMS vs. NoSQL

The differences between RDBMS and NoSQL databases are as follows:

Feature	RDBMS	NoSQL Databases
Data Storage	Tabular	Variable storage model
Schema	Fixed	Dynamic
Performance	Low	High
Scalability	Vertical	Horizontal
Reliability	Good	Poor

Assisted Practice

YARN Tuning

Duration: 15 mins

Problem Statement: In this demonstration, you will learn, how to tune YARN and allow HBase to run smoothly without being resource starved.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

HBase Overview

What Is HBase?

HBase is a database management system designed in 2007 by Powerset, a Microsoft company.

HBase rests on top of HDFS and enables real-time analysis of data.

What Is HBase?

It can store huge amount of data in tabular format for extremely fast reads and writes.

HBase is mostly used in a scenario that requires regular and consistent inserting and overwriting of data.

Why HBase?

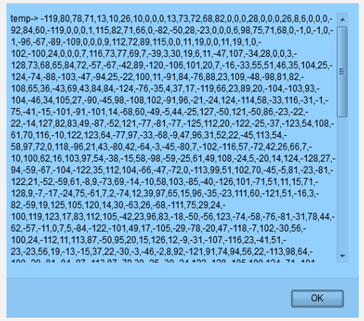
HDFS stores, processes, and manages large amounts of data efficiently.
However, it performs only batch processing and the data will be accessed in a sequential manner.

Therefore, a solution is required to access, read, or write data anytime regardless of its sequence in the clusters of data.

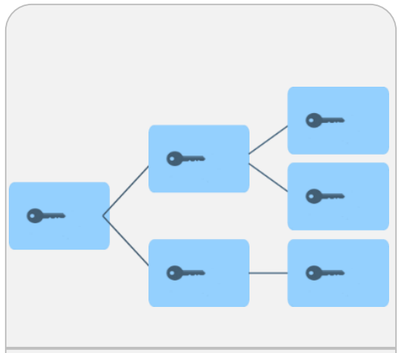
Characteristics of HBase

HBase is a type of NoSQL database and is classified as a key-value store. In HBase:

Value is identified with a key



Key and value are a ByteArray



Values are stored in key-orders

Quickly accessed by value keys

HBase is a database in which tables have no schema. At the time of table creation, column families are defined, not columns.

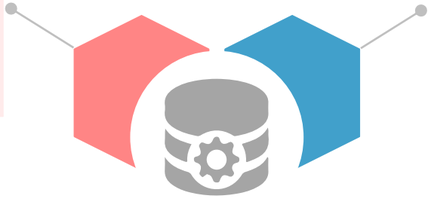
HBase: Real-Life Connect

Facebook's messenger platform needs to store over 135 trillion messages every month.

Where do they store such data?

APACHE HBASE

Rarely Accessed Dataset



Highly Volatile Dataset

DATA AND
ARTIFICIAL INTELLIGENCE

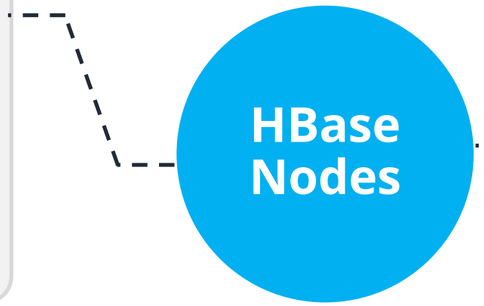
HBase Architecture

HBase Architecture

HBase has two types of nodes: Master and RegionServer. Their characteristics are as follows:

Master

- Single Master node running at a time
- Manages cluster operations
- Not a part of the read or write path



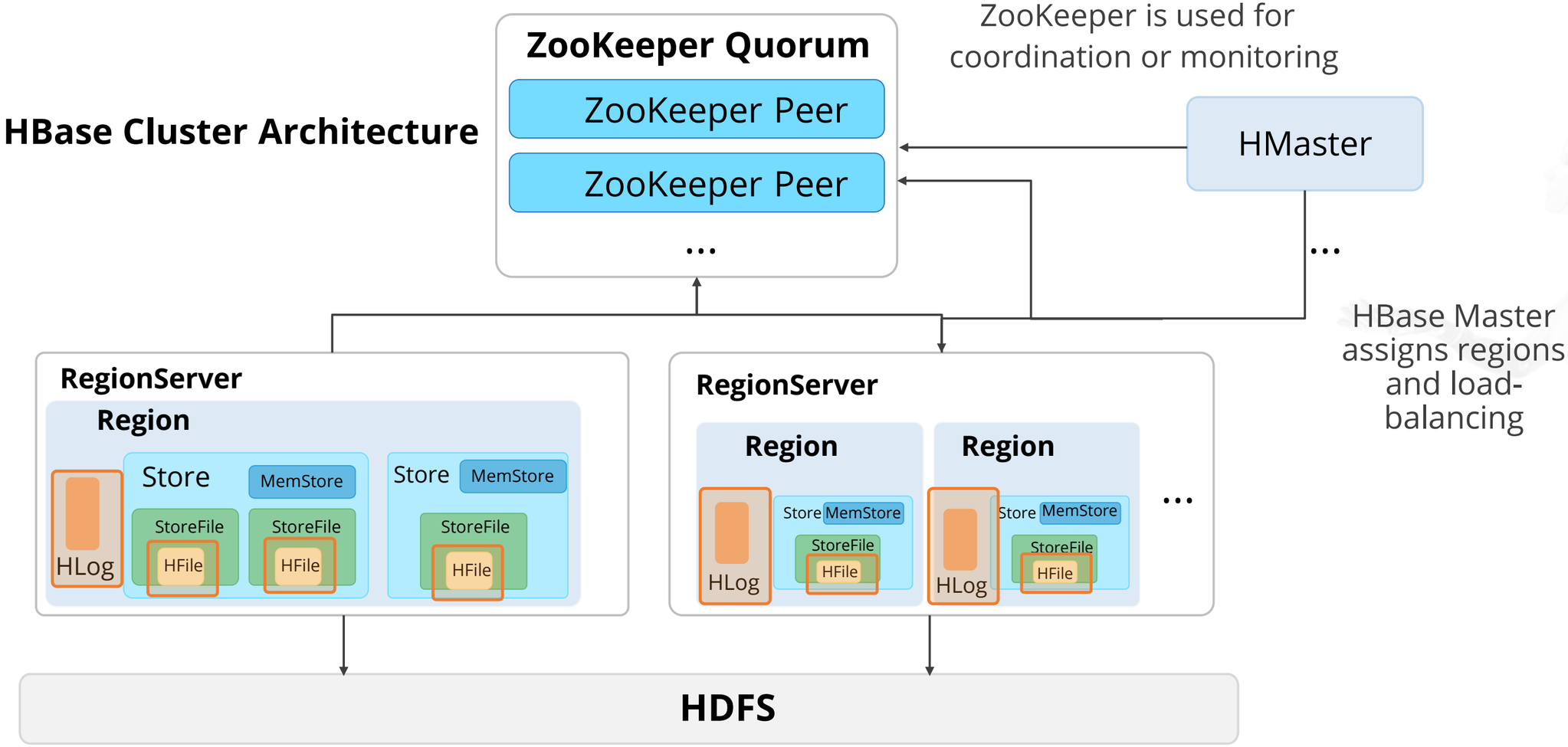
RegionServer

- One or more RegionServers running at a time
- Hosts tables and performs reads and buffer writes
- RegionServer is communicated in order to read and write

A region in HBase is the subset of a table's rows. The Master node detects the status of RegionServers and assigns regions to it.

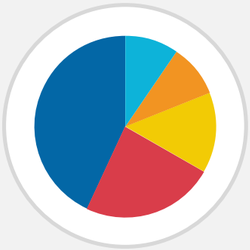
HBase Components

The HBase components include HBase Master and multiple RegionServers.



Storage Model of HBase

The two major components of the storage model are as follows:



Partitioning:

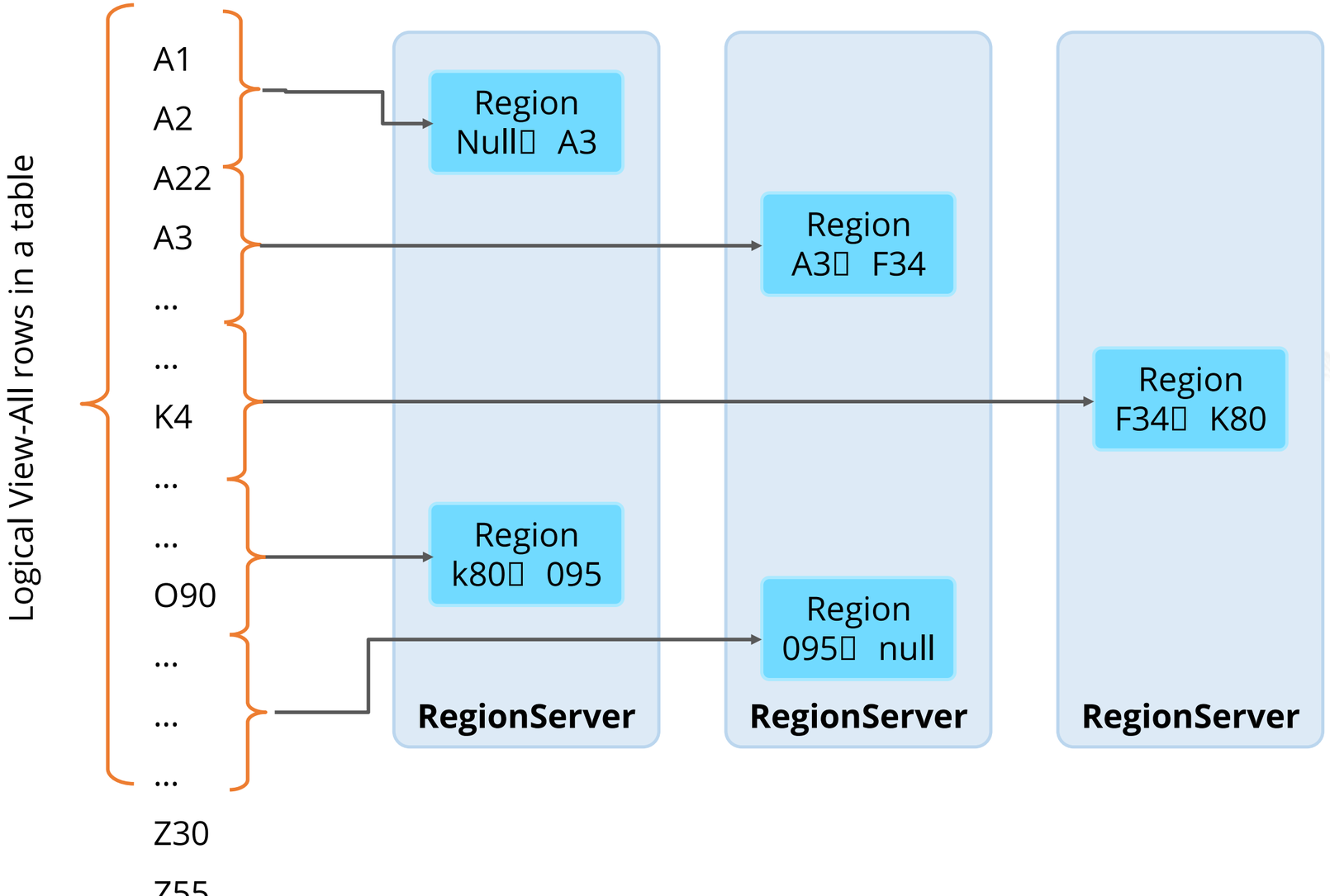
- A table is horizontally partitioned into regions.
- Each region is managed by a RegionServer.
- A RegionServer may hold multiple regions.

Persistence and data availability:

- HBase stores its data in HDFS, does not replicate RegionServers, and relies on HDFS replication for data availability.
- Updates and reads are served from the in-memory cache called MemStore.

Row Distribution of Data between RegionServers

The distribution of rows of structured data using HBase is illustrated here:



Data Storage in HBase

Data is stored in files called HFiles or StoreFiles that are usually saved in HDFS.

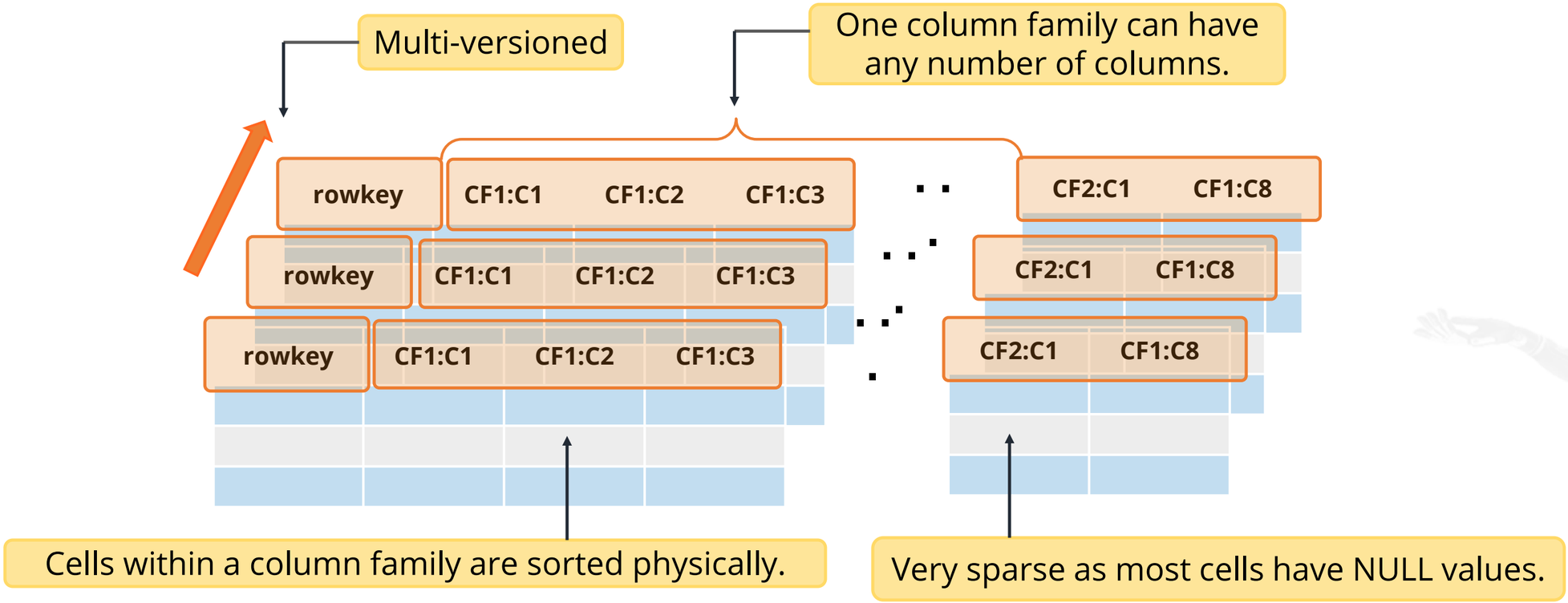
- HFile is a key-value map.
- When data is added, it is written to a log called the Write Ahead Log, and it is stored in memory, MemStore.
- HFiles are immutable, since HDFS does not support updates to an existing file.
- HBase periodically performs data compactions to control the number of HFiles and to keep the cluster well-balanced.

DATA AND
ARTIFICIAL INTELLIGENCE

Data Model

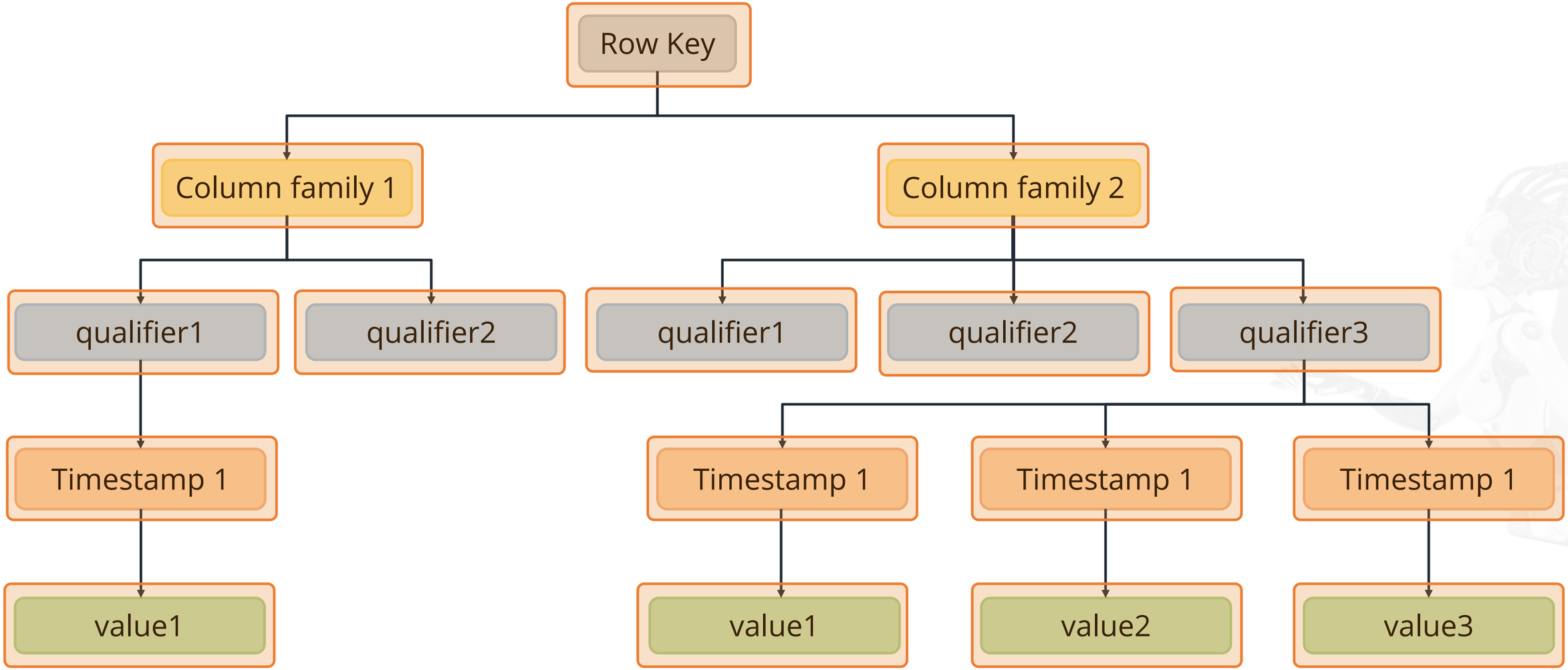
Data Model

Following are the features of the data model in HBase:

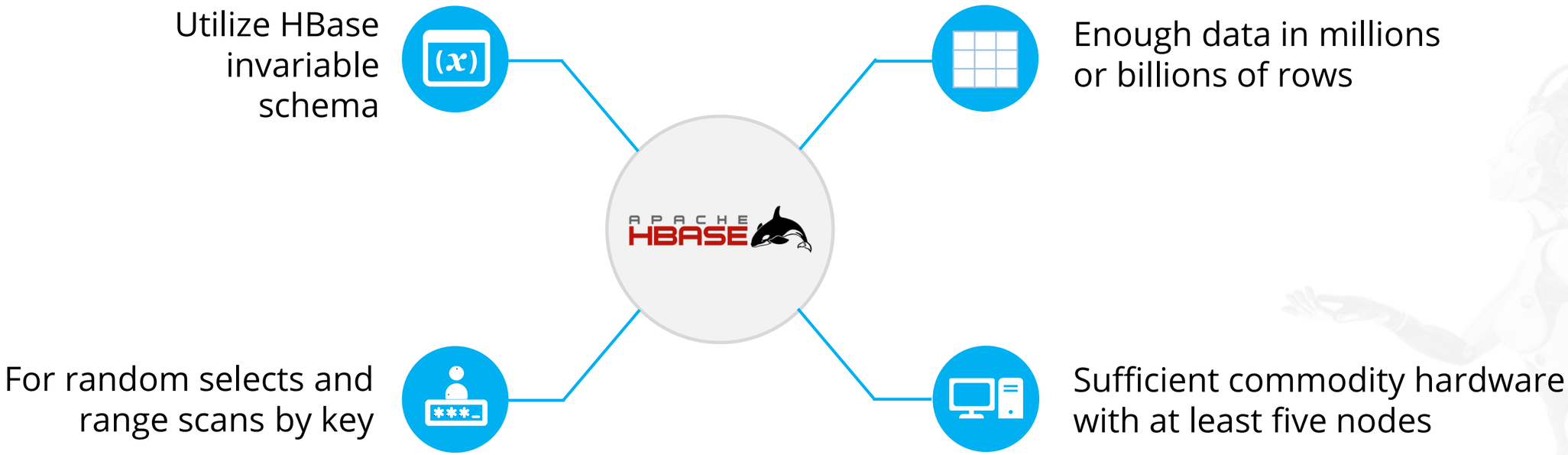


Everything except table names are stored as ByteArrays.

Data Mode: Features



When to Use HBase?



HBase vs. RDBMS

The table shows a comparison between HBase and a Relational Database Management System (RDBMS):

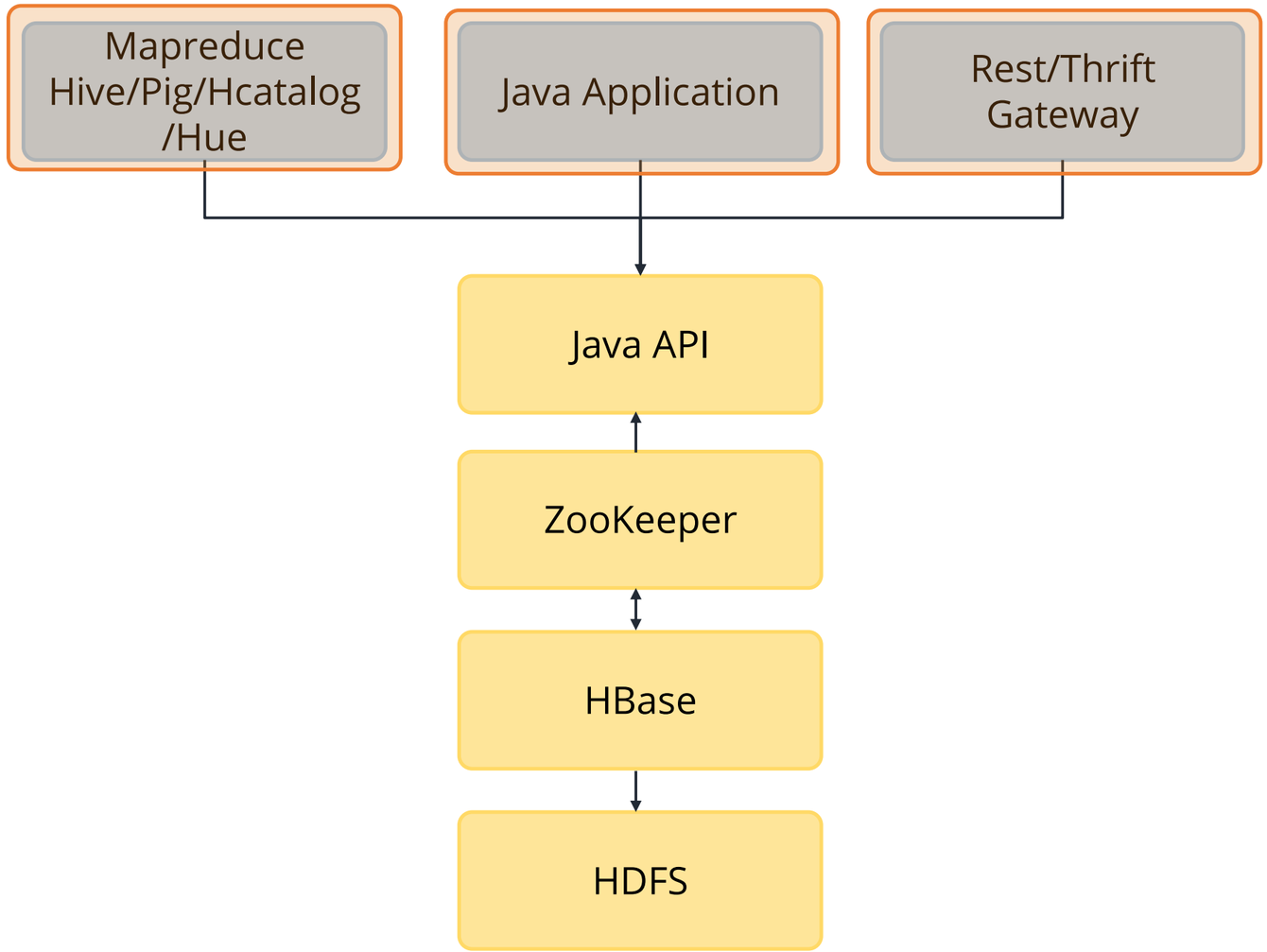
HBase	RDBMS
Automatic partitioning	Usually manual and admin-driven partitioning
Scales linearly and automatically with new nodes	Usually scales vertically by adding more hardware resources
Uses commodity hardware	Relies on expensive servers
Has fault tolerance	Fault tolerance may or may not be present
Leverages batch processing with MapReduce distributed processing	Relies on multiple threads or processes rather than MapReduce distributed processing

DATA AND
ARTIFICIAL INTELLIGENCE

Connecting to HBase

Connecting to HBase

HBase can be connected through the following media:

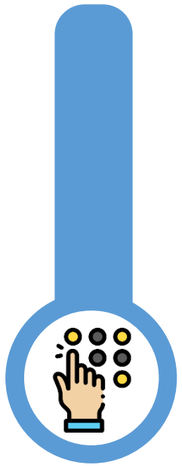


HBase Shell Commands

Common commands include, but are not limited to, the following:

Create table. Pass table name from a dictionary of specifications per column family, and a dictionary of table configuration which is optional	HBase> create ‘t1', {NAME => ‘f1'}, {NAME => ‘f2'}, {NAME => ‘f3'} HBase> # The above in shorthand would be the following: HBase> create ‘t1', ‘f1', ‘f2', ‘f3'
Describe the table named	HBase> describe ‘t1'
Start the disabling of the table named	HBase> disable ‘t1'
Drop the table named. Table must first be disabled	HBase> drop ‘t1'
List all tables in HBase. Optional regular expression parameter can be used to filter the output.	HBase> list

HBase Shell Commands



Count
Counting the number of rows in a table

Delete
Deleting a cell value

Get
Getting the contents of a row or a cell

Put
Putting a cell value

Scan
Scanning a table's values

Unssisted Practice

HBase Shell

Duration: 15 mins

Problem Statement: Create a sample HBase table on the cluster, enter some data, query the table, then clean up the data and exit.

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**.

Unassisted Practice

Steps to Perform

- **HBase Shell**

// Start the HBase shell
hbase shell

// Create a table called simplilearn with one column family named stats:
create 'simplilearn', 'stats'

// Verify the table creation by listing everything
list

// Add a test value to the daily column in the stats column family for row 1:
put 'simplilearn', 'row1', 'stats:daily', 'test-daily-value'

Unassisted Practice

Steps to Perform

- **HBase Shell**

// Add a test value to the weekly column in the stats column family for row 1:
put 'simplilearn', 'row1', 'stats:weekly', 'test-weekly-value'

// Add a test value to the weekly column in the stats column family for row 2:
put 'simplilearn', 'row2', 'stats:weekly', 'test-weekly-value'

// Type **scan 'simplilearn'** to display the contents of the table.

// Type **get 'simplilearn', 'row1'** to display the contents of row 1.

Type **disable 'simplilearn'** to disable the table.
Type **drop 'simplilearn'** to drop the table and delete all data.
Type **exit** to exit the HBase shell.

Key Takeaways

You are now able to:

- Understand the need for NoSQL databases
- Analyze the HBase architecture and components
- Differentiate HBase from RDBMS

Quiz

Knowledge Check

1

Which of the following are the nodes of HBase?

a. Spooldir and Master

b. Syslog and RegionalServer

c. Master and Regional Server

d. None of the above

Knowledge Check

1

Which of the following are the nodes of HBase?

a. Spooldir and Master

b. Syslog and RegionalServer

c. Master and Regional Server

d. None of the above

The correct answer is **c.**

Master and RegionalServer are the nodes of HBase, whereas the other options are parts of Flume.

Knowledge Check

2

In which of the following scenarios can we use HBase?

a. For random selects and range scans by key

b. For sufficient commodity hardware with at least five nodes

c. In variable schema

d. All of the above

Knowledge Check

2

In which of the following scenarios can we use HBase?

a. For random selects and range scans by key

b. For sufficient commodity hardware with at least five nodes

c. In variable schema

d. All of the above

The correct answer is **d.**

HBase can be used for random selects and range scans by key, for sufficient commodity hardware with at least five nodes, and in variable schema.

Lesson-End-Project

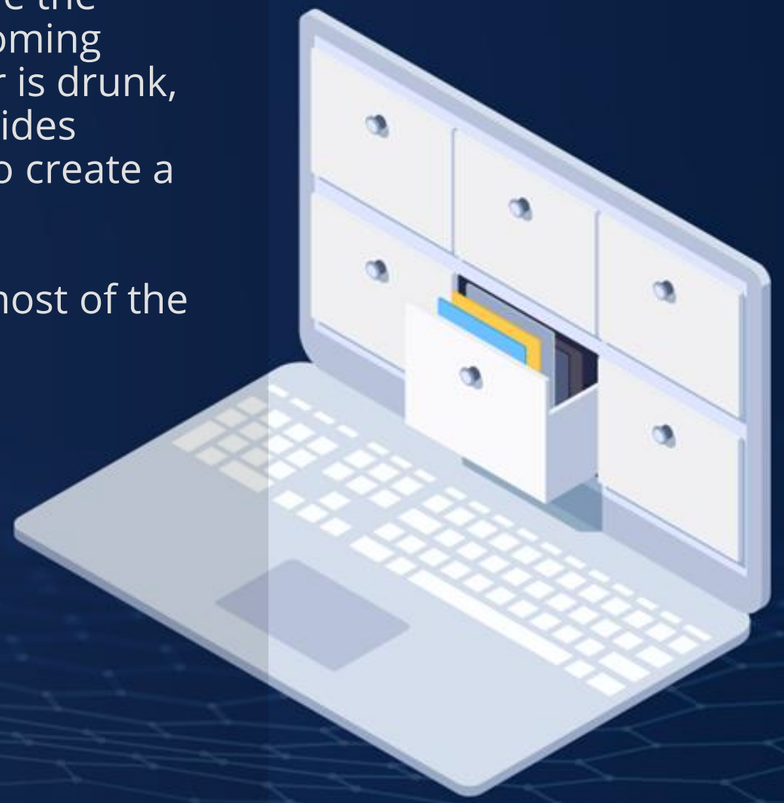
Problem Statement:

Global transport private limited is in transport analytics and they are keen to ensure the safety of people. Nowadays, as the population is increasing accidents are also becoming more and more frequent. Accidents occur mostly when the route is long, the driver is drunk, or the roads are damaged. The company collects data of all the accidents and provides important insights that can reduce the number of accidents. The company wants to create a public portal where anyone can see the accident's aggregated data.

Your task is to suggest a suitable database and design a schema which can cover most of the use cases.

You are given a file that contains details about the various parameter of accidents.
The column details are as follows:

1. Year
2. TYPE
3. 0-3 hrs. (Night)
4. 3-6 hrs. (Night)
5. 6-9 hrs (Day)
6. 9-12 hrs (Day)
7. 12-15 hrs (Day)
8. 15-18 hrs (Day)
9. 18-21 hrs (Night)
10. 21-24 hrs (Night)
11. Total



Lesson-End-Project

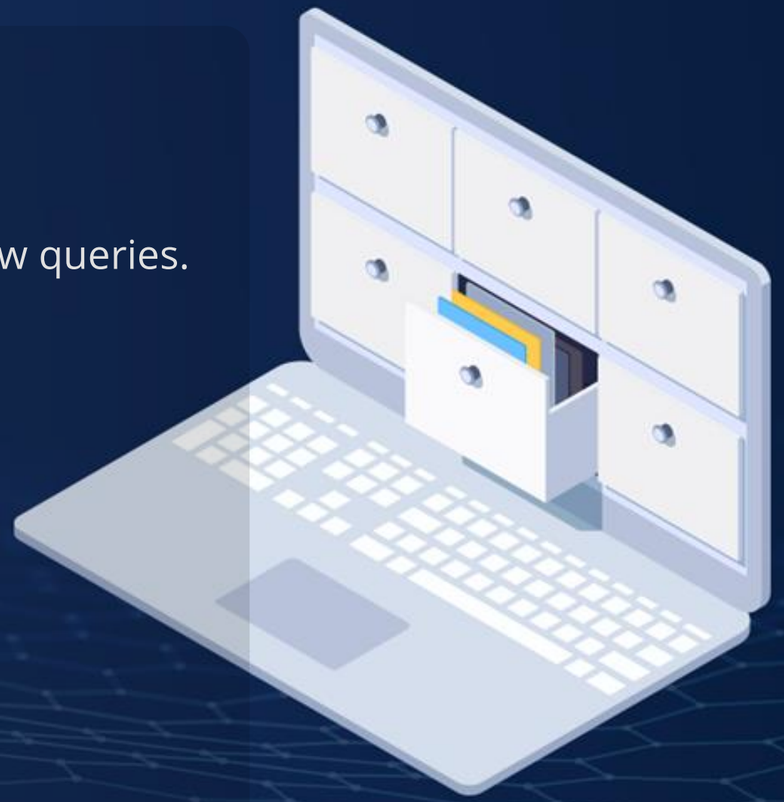
Problem Statement:

You have to save the given data in HBase in such a way that you can solve the below queries. Please mention what you are selecting as a row key and why.

1. Get the total number of accidents when you are given
   a. Year
   b. Type of Accident
   c. Time Duration

2. Get the total number of accidents when you are given
   a. Year
   b. Type of Accident

3. Get the total number of accidents in a given year



DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Basics of Functional Programming and Scala

Learning Objectives

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

- Describe and demonstrate programming with Scala
- Define functions, anonymous function, and class in Scala
- Use the different types of collections
- Describe and demonstrate Scala REPL (Read, Eval, Print, and Loop)

DATA AND
ARTIFICIAL INTELLIGENCE

Introduction to Scala

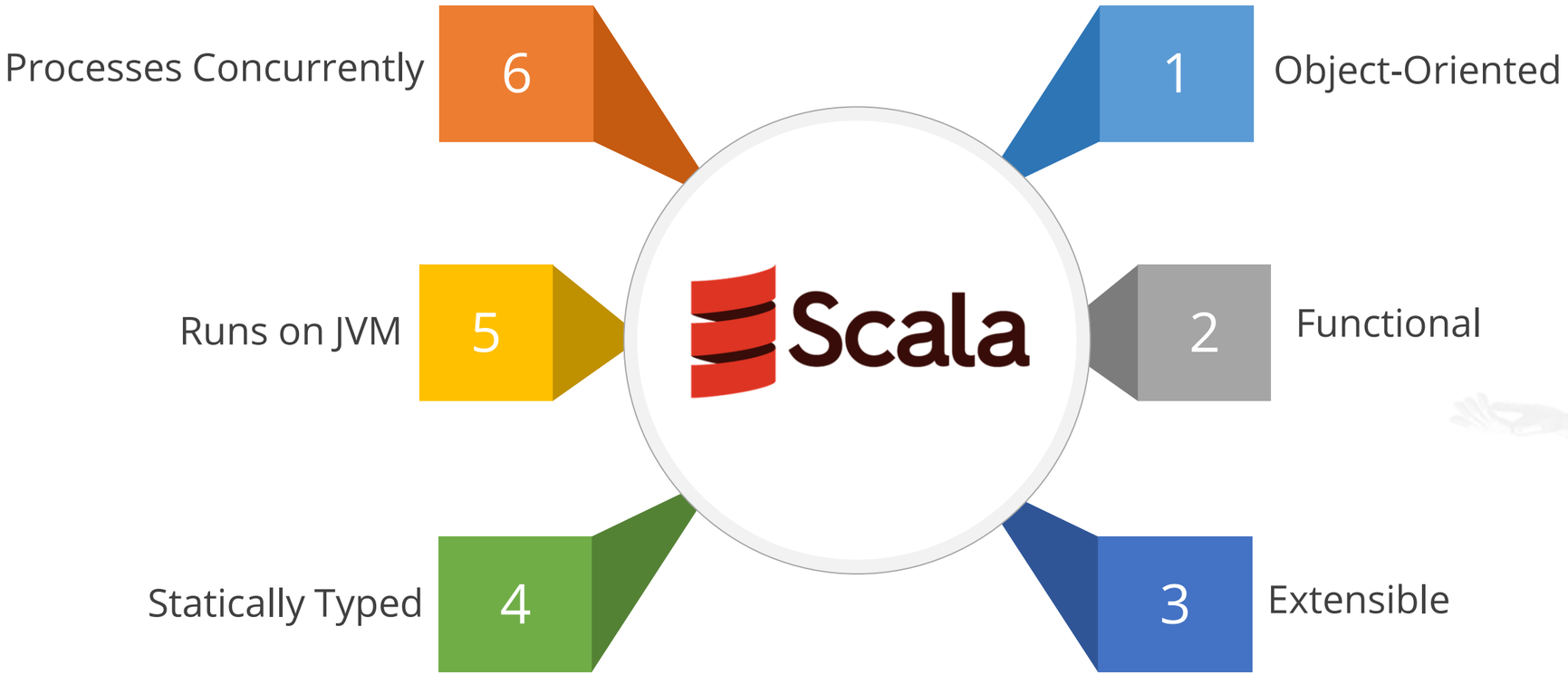
Scala

Scala

Scala

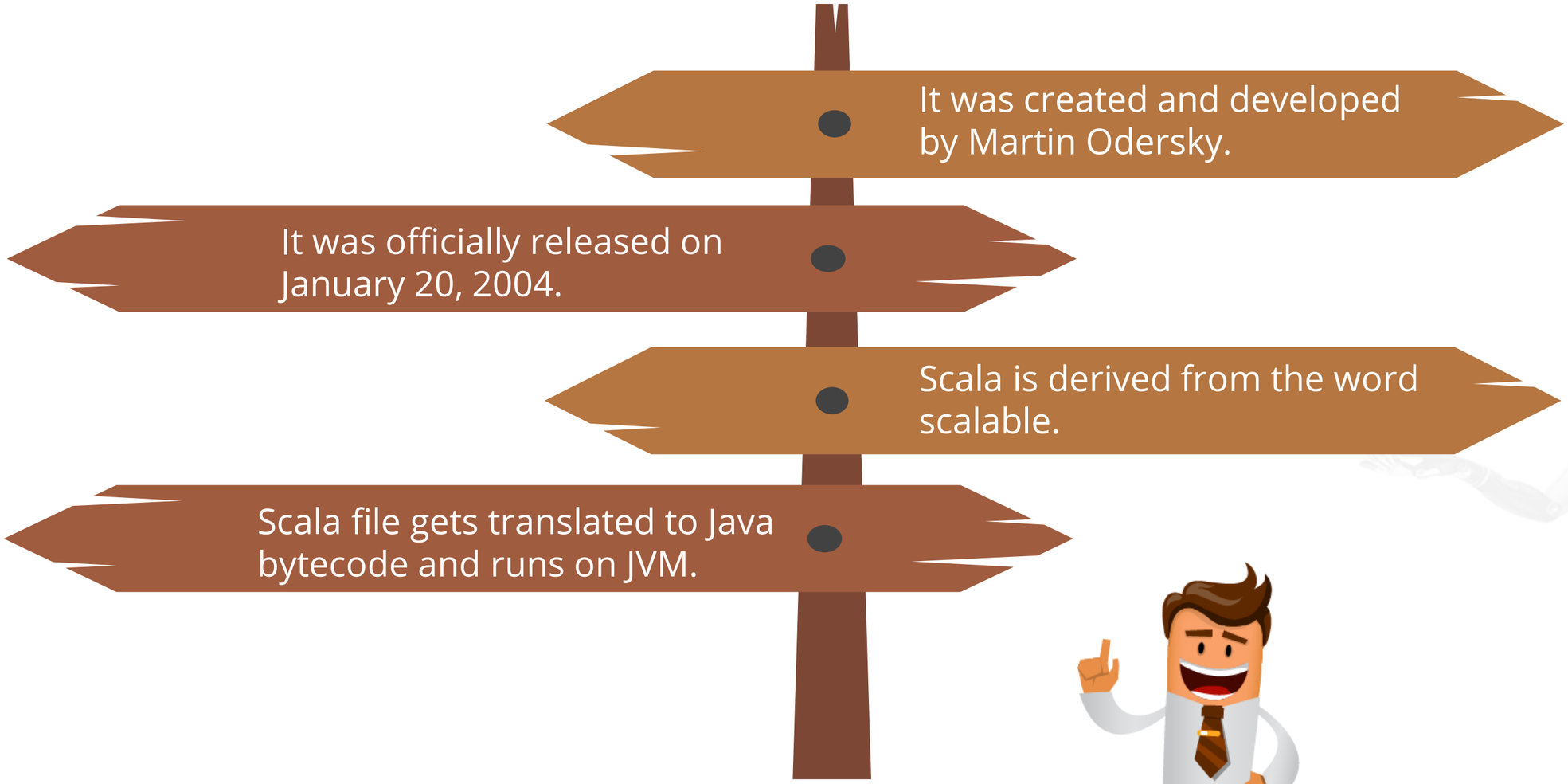
Scala is an object-oriented and functional programming language designed to demonstrate the common programming patterns in a concise, refined, and type-safe way.

Features of Scala



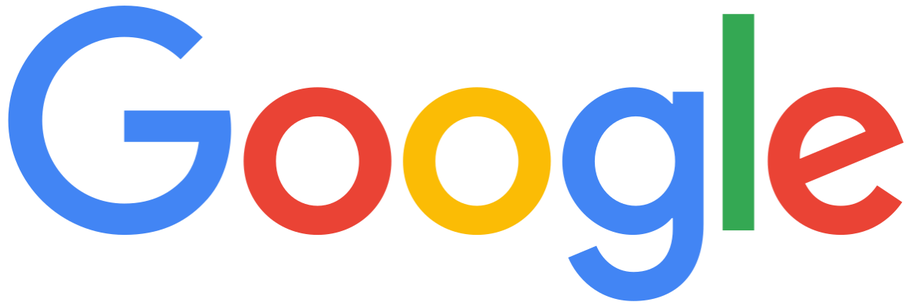
Scala

History of Scala

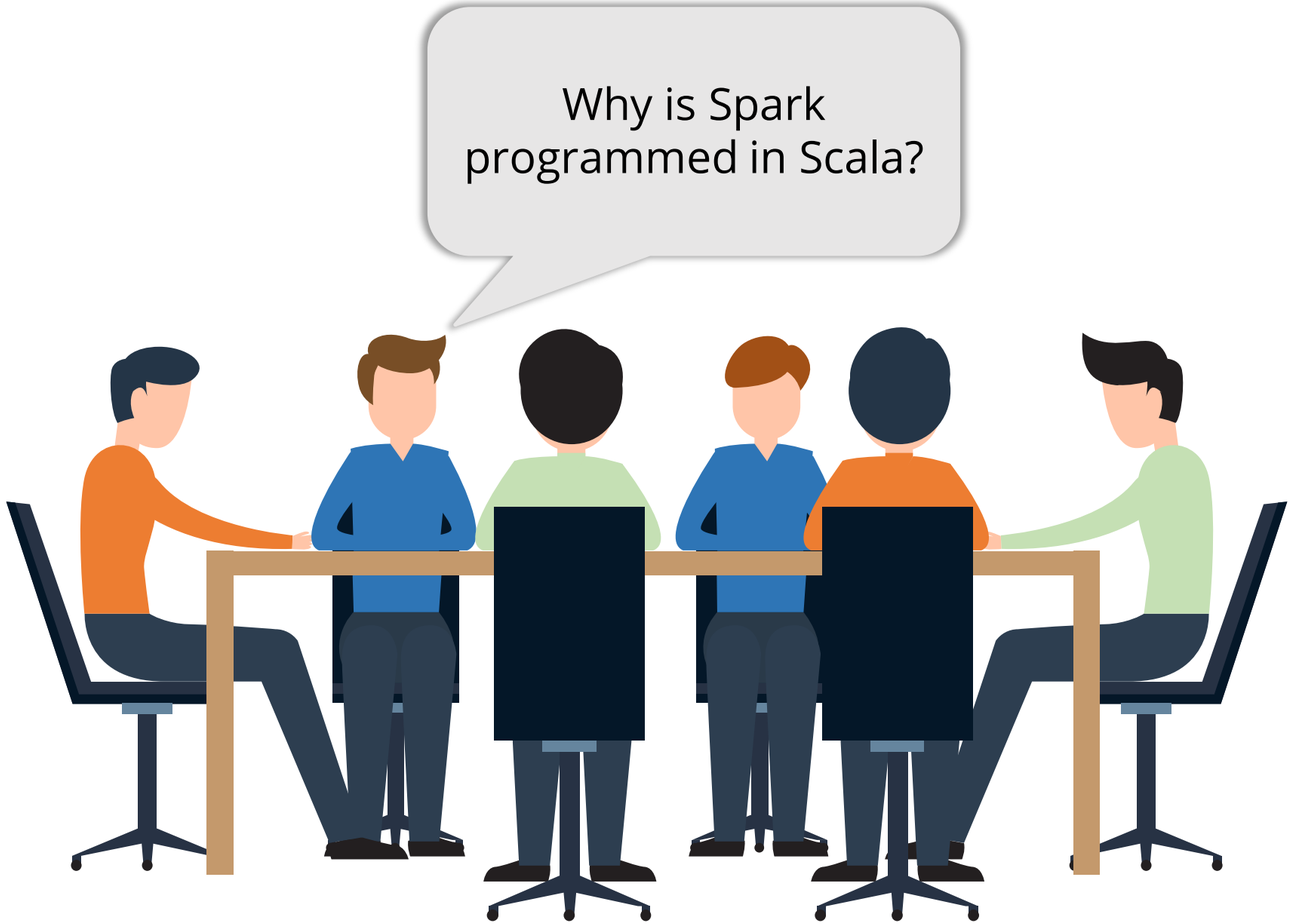


Popularity of Scala

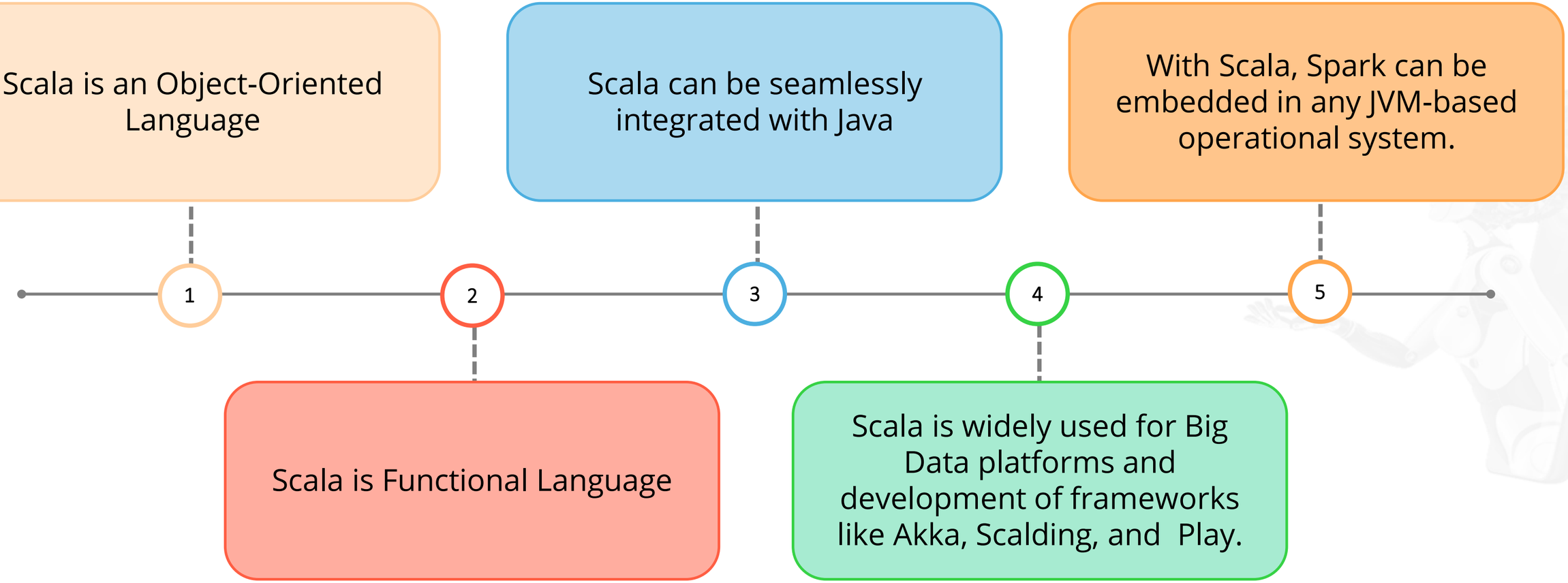
The following companies use Scala, which makes it popular:



Why Spark Programmed in Scala?



Why Is Spark Programmed in Scala?



Install Scala

Duration: 5 mins

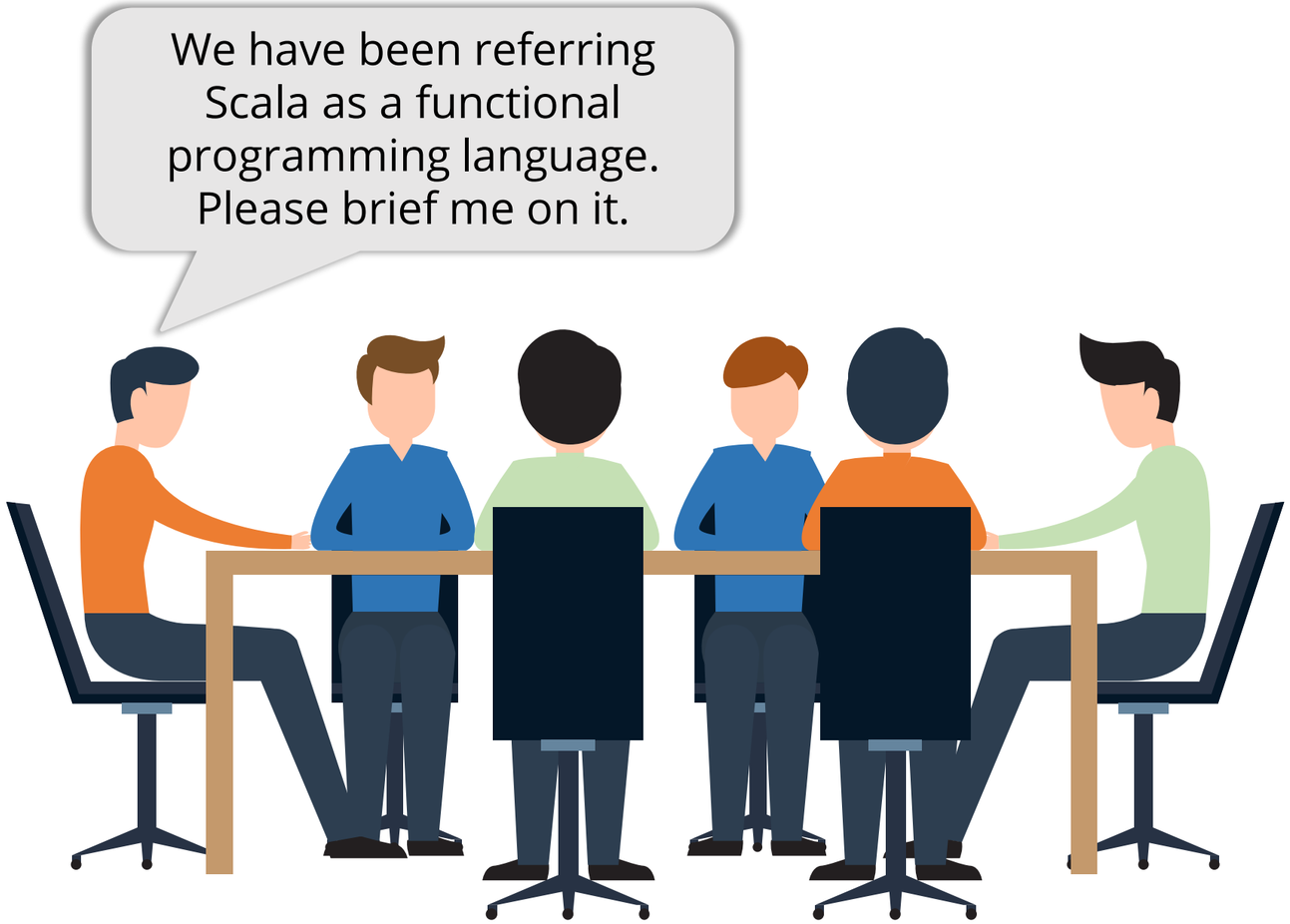
Problem Statement: In this demonstration, you will install Scala.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

Functional Programming

Functional Programming



Functional Programming

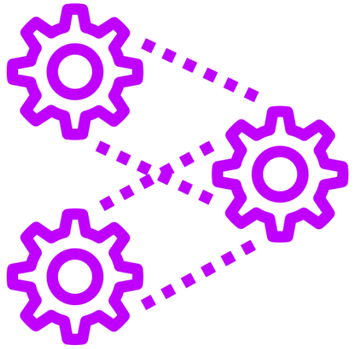
Functional programming languages are designed on the concept of mathematical functions that use conditional expressions and recursion to perform computation.

Pure Functional Language

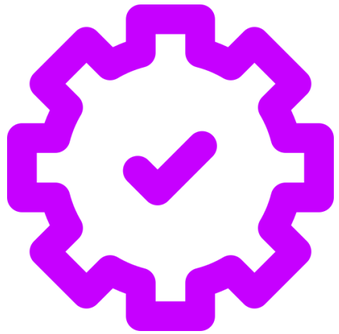
Impure Functional Language

Advantages of Functional Programming

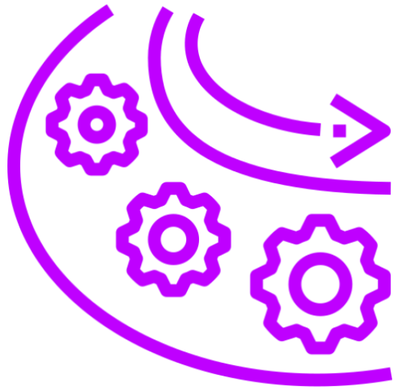
Bug Free



Efficient Parallel Processing



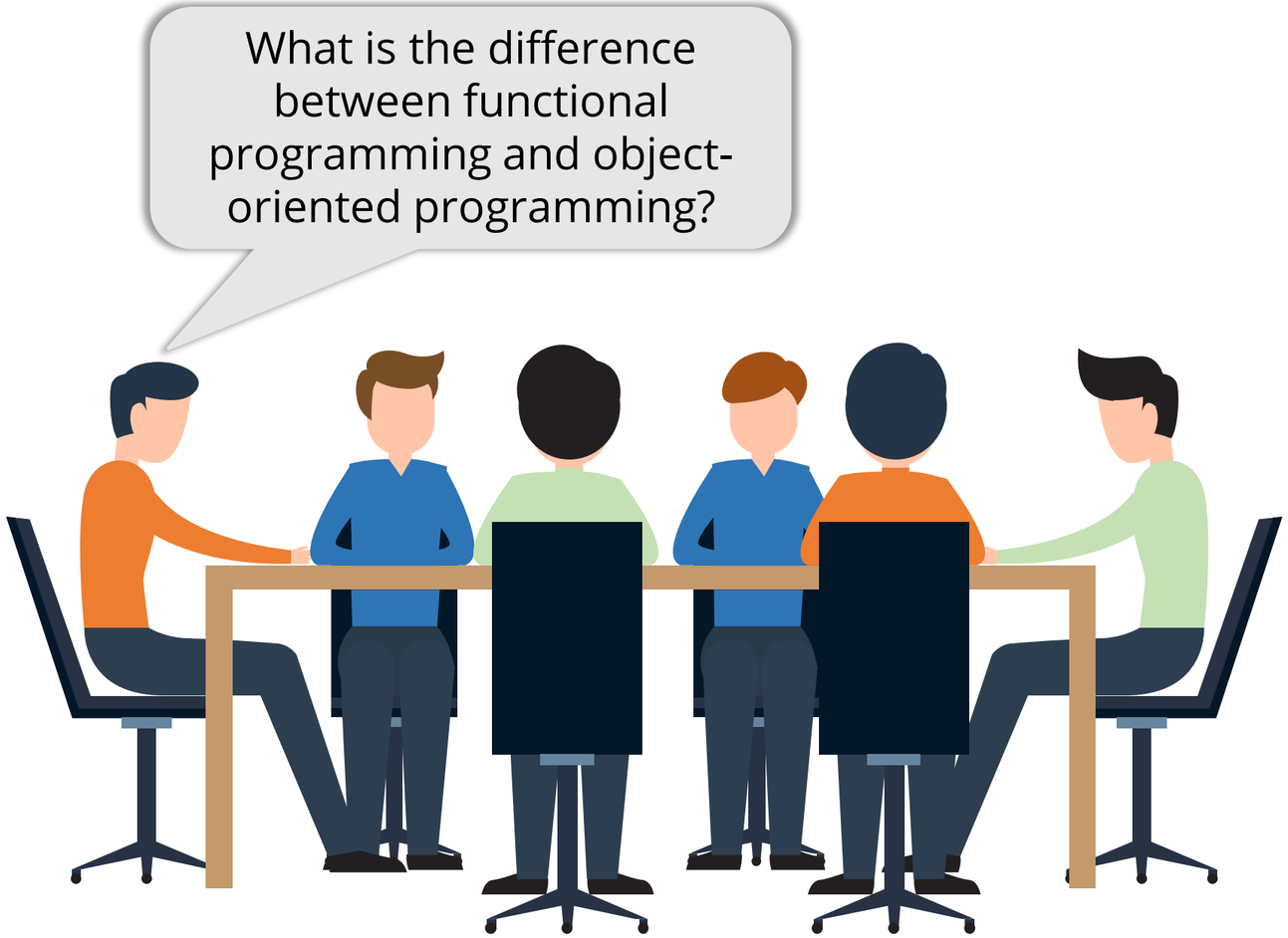
Efficiency



Nested Processing

Lazy Evaluation

Functional Programming



Functional Programming vs. Object-Oriented Programming

Functional Programming	Object-Oriented Programming
Uses immutable data	Uses mutable data
Supports parallel processing	Does not support parallel processing
Flow control is done using function calls and function calls with recursion	Flow control is done using loops and conditional statements
Supports both "Abstraction over Data" and "Abstraction over Behavior"	Supports only "Abstraction over Data"
Follows Declarative Programming Model	Follows Imperative Programming Model

DATA AND
ARTIFICIAL INTELLIGENCE

Programming with Scala

Data Types

Scala supports the same data types as Java does. The table below lists and explains the Scala data types:

Data Type	Explanation
Byte	8-bit signed value; Range: -128 to 127
Short	16-bit signed value; Range: -128 to 127
Int	32-bit signed value; Range: -128 to 127
Long	64-bit signed value; Range: -128 to 127
Float	Single-precision 32-bit IEEE 754 value
Double	Double-precision 64-bit IEEE 754 value
Char	16-bit unsigned Unicode character; Range: U+0000 to U+FFFF
String	Chars sequence
Unit	No value
Null	Empty or null reference
Boolean	Literal true or false
Any	Super type of any type
AnyRef	Super type of any reference type
Nothing	Sub type of every other type; No value contained

Basic Literals

The basic literals used in Scala are listed below with examples:

Integer Literals

Example:
scala> val hex = 0x6; output - hex: Int = 6

Floating Type Literals

Examples:
scala> val big = 1.2345; output - big: Double = 1.2345
scala> val little = 1.2345F; output - little: Float = 1.2345

Character Literals

Example:
scala> val a = 'A'

Basic Literals

Boolean Literals

Examples:
scala> val bool = true; output - bool: Boolean = true
scala> val fool = false; output - fool: Boolean = false

Symbol Literals

Example:
scala> updateRecordByName('favoriteBook, “Spark in Action")

String Literals

Examples:
scala> val hello = "hello“; output - hello: java.lang.String = hello

```
println("""Welcome to Ultamix 3000 Type "HELP" for help.""")
    Welcome to Ultamix 3000
    Type "HELP" for help
```

Introduction to Operators

An operator is a symbol that allows performing specific logical or mathematical manipulations.
It provides a syntax for general method calls.

Example: 2 + 1 actually implies (2)+(1)

The class Int includes a method called + that takes an Int and provides an Int as a result. To invoke the + method, you need to add two Ints as follows:
scala> val sum = 2 + 1 //Scala invokes (2)+(1)

Types of Operators

Arithmetic Operators

Object Equality Operators

Relational and Logical Operators

Bitwise Operators

Basic Literals and Arithmetic Operators

Duration: 5 mins

Problem Statement: In this demonstration, you will use basic literals and arithmetic operators in Scala.

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**.

Logical Operators

Duration: 5 mins

Problem Statement: In this demonstration, you will use the logical operators in Scala.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

Type Inference, Classes, Objects, and Functions in Scala

Type Inference

Type Inference is a built-in mechanism that allows you to omit certain types of annotations and return types of methods.

Example:

```
object InferenceTest1 extends Application {
val x = 1 + 2 * 3 // the type of x is Int
val y = x.toString() // the type of y is String
def succ(x: Int) = x + 1 // method succ returns Int values
}
```

Objects

An object is a named instance with members like methods and fields.

Scala has singleton objects instead of static members, which is a class with only one instance and can be created using the keyword object.

```
package logging

object Logger {
        def info(message: String): Unit = println(s”INFO: $message”)
}
```

Classes

Classes in Scala are blueprints for creating objects. They can contain methods, values, variables, types, objects, and traits which are collectively called members.

```
Class User

Val user1 = new User
```

Functions

Scala provides a rich set of built-in functions and allows you to create user defined functions also.

In Scala, functions are first class values. These can be returned as a result or passed as a parameter.

```
def sumInts(a: Int, b: Int) : Int = {if (a > b) 0
else
a + sumInts(a + 1, b)}
```

Anonymous Functions

An anonymous function is an alternative of named function definitions that gets created by parameterization by functions.

Example:

```
(y: Int) => y * y
```

Higher-Order Functions

Higher-order functions take other functions as parameters or return a function as a result.

Map is an example of higher-order functions in Scala.

```
val salaries = Seq(20000, 70000, 40000)
val doubleSalary = (x: Int) => x * 2
val newSalaries = salaries.map(doubleSalary) // List(40000, 140000, 80000)
```

Define Type Inference, Functions, Anonymous Functions, and Class

Duration: 10 mins

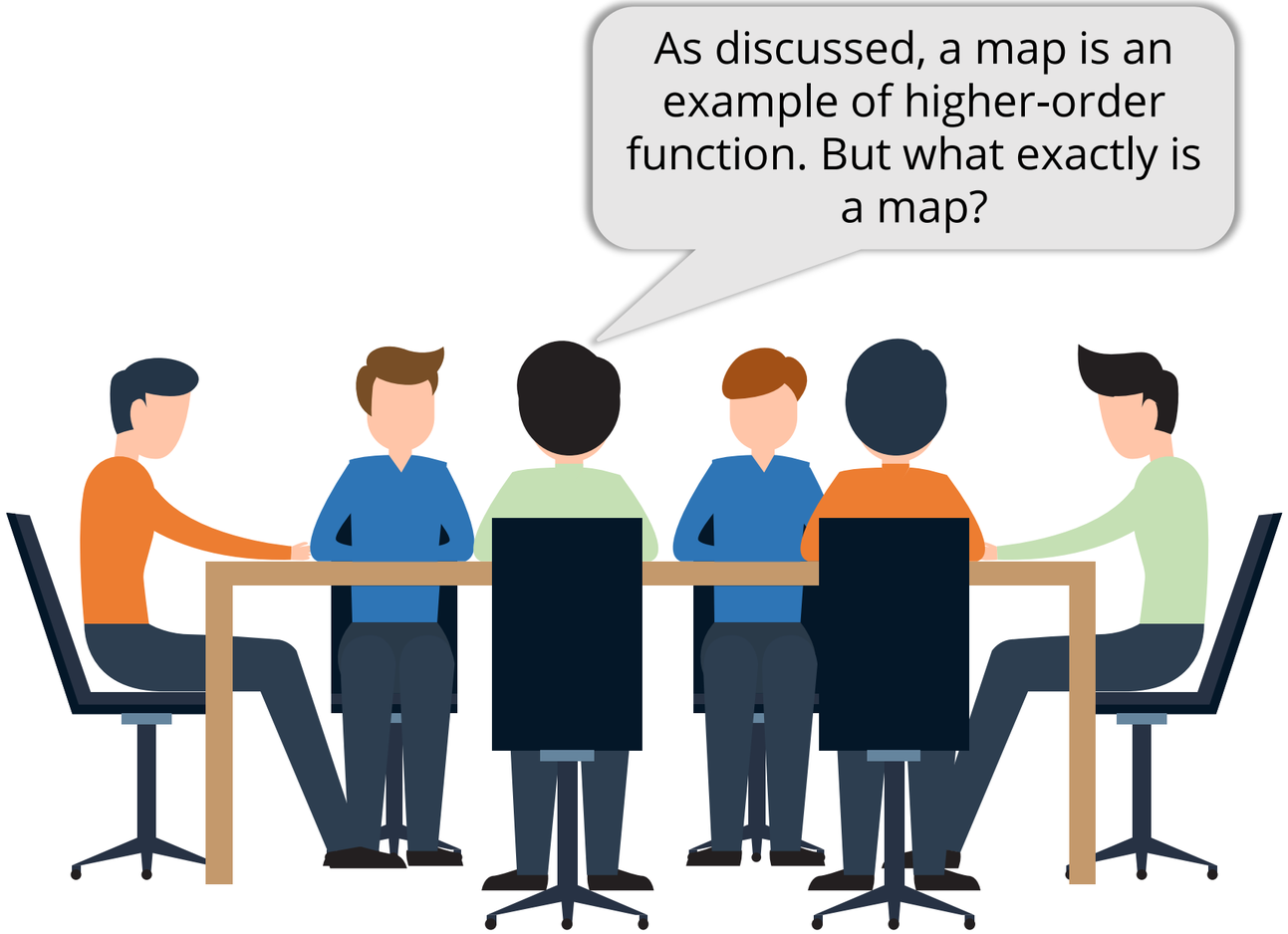
Problem Statement: In this demonstration, you will learn how to define type inference, functions, anonymous functions, and class in Scala.

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**.

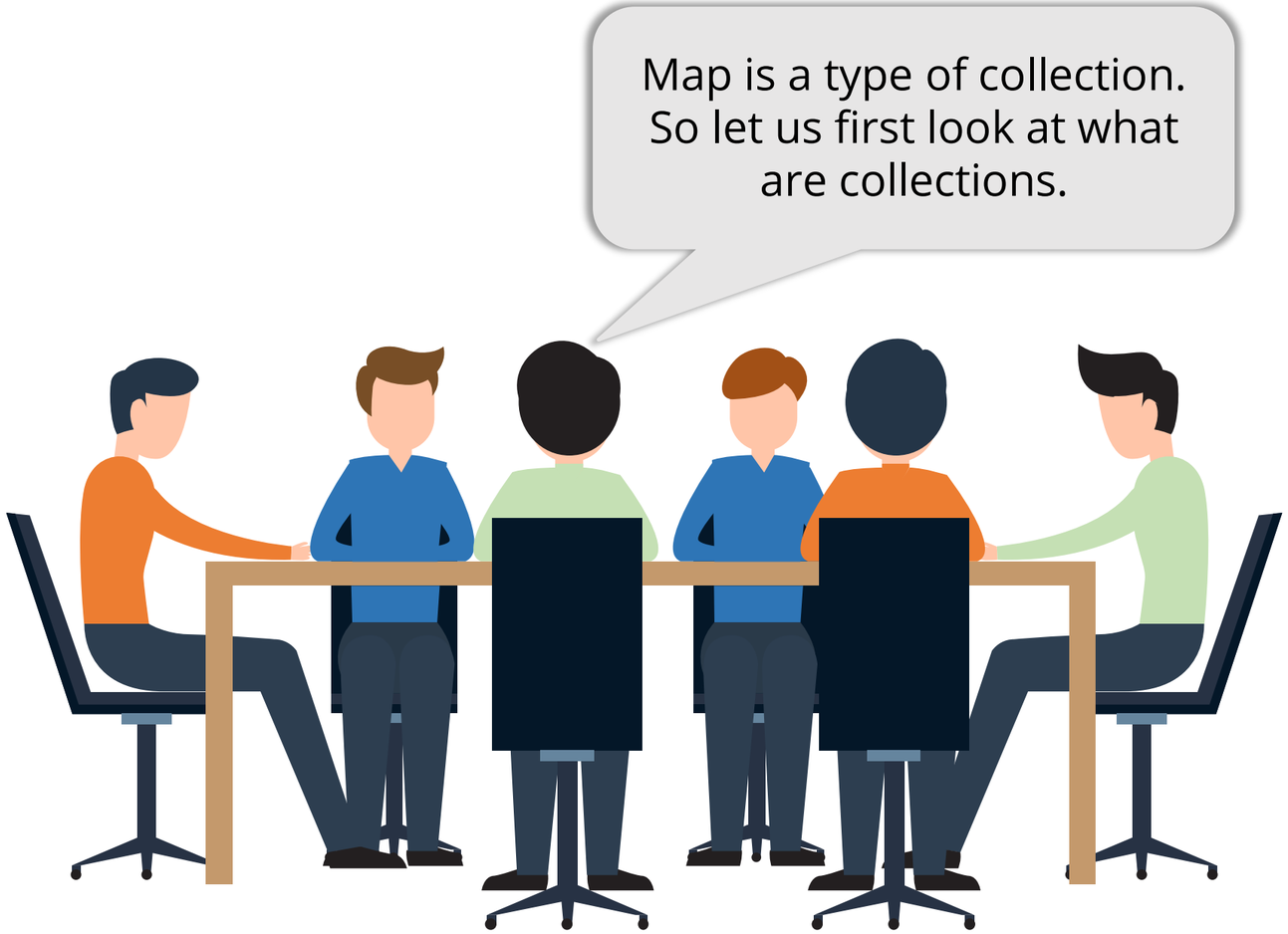
DATA AND
ARTIFICIAL INTELLIGENCE

Collections

Collections



Collections



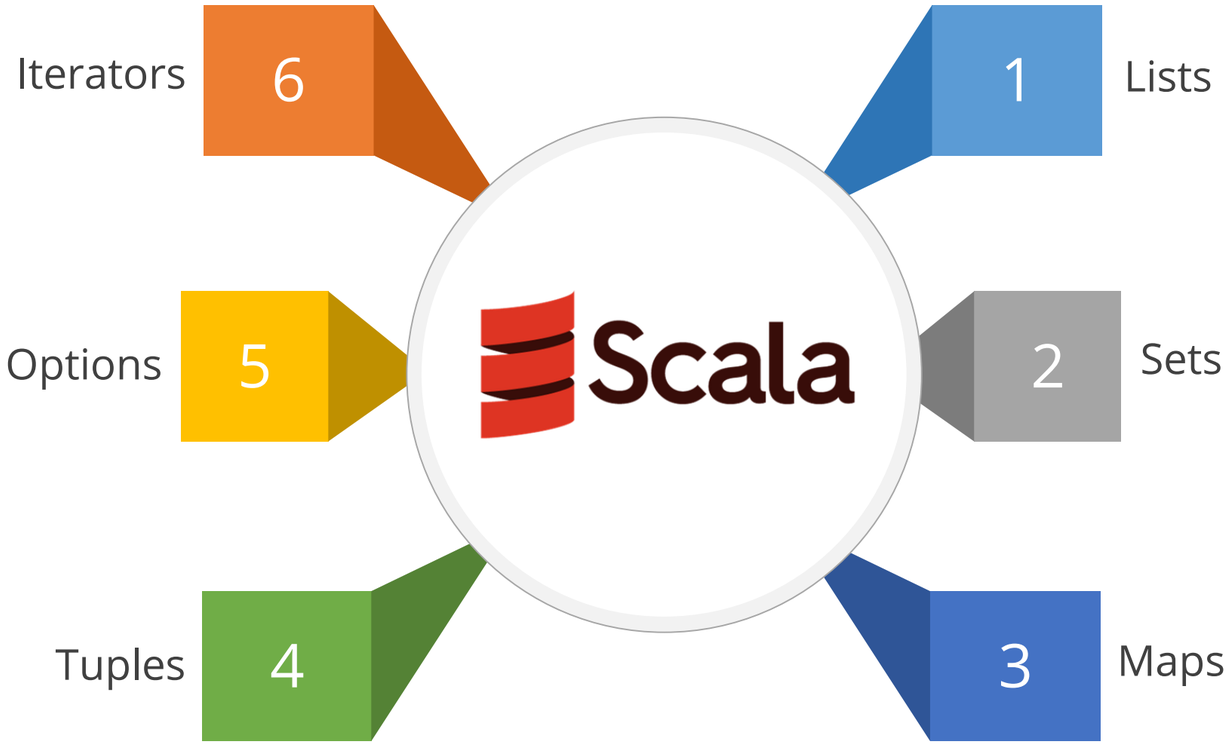
Collections

Collections are containers of things, that can be sequenced as linear sets of items.

Mutable Collections

Immutable Collections

Types of Collections



Types of Collections

DATA AND
ARTIFICIAL INTELLIGENCE

Types of Collections

Lists

In Scala lists, all the elements have the same data type. They are immutable and represent a linked list.

The type of a list that has elements of type T is written as List[T].

```
//List of strings
val fruit: List[String] = List("apples", "oranges", "pears")

//List of integers
val nums: List[Int] = List(1, 2, 3, 4)
```

Sets

Sets are iterables that contain no duplicate elements.

```
//Empty set of integer type
var s : Set[Int] = Set()

//Set of integer type
var s : Set[Int] = Set(1,3,5,7)
```

Sets

Sets have the following fundamental operations:

01 Tests

02 Additions

03 Removals

04 Set Operations

Maps

A map is an iterable consisting of pairs of keys and values.

```
//Empty hash table whose keys are strings and values are integers:
var A:Map[Char,Int] = Map()

//A map with keys and values.
val colors = Map("red" -> "#FF0000", "azure" -> "#F0FFFF")
```

Maps

Maps have the following fundamental operations:

01 Lookup

02 Additions and Updates

03 Removals

04 Subcollection Producers

05 Transformations

Tuples

In Scala, a tuple is a value that contains a fixed number of elements, each with a distinct type.

```
//Tuple of integers
val t = (1,2,3,4)

//Tuple containing integer, string, and a console
val t = (1, “Welcome", Console)
```

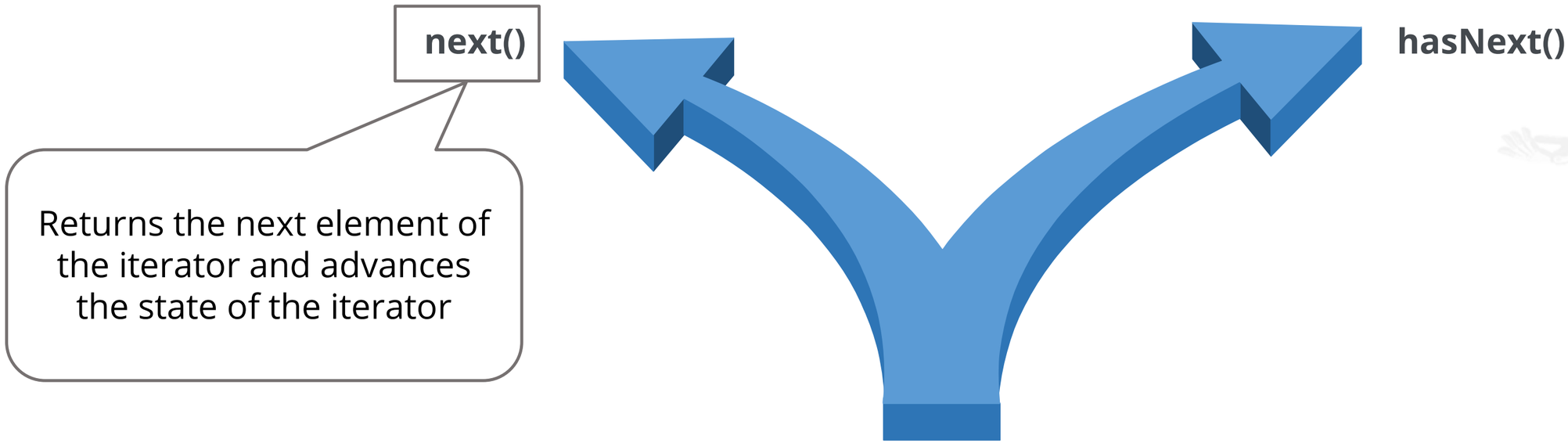
Options

An Option[T] is a container for one element or zero, which represents a missing value.

```
//Define an option
val x:Option[Int] = Some(5)
```

Iterators

An iterator is a way to access the elements of a collection one by one.



Iterators

An iterator is a way to access the elements of a collection one by one.

Demonstrate the Types of Collections

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to use different types of collections such as List, Set, Map, Tuple, and Option in Scala.

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**.

Perform Different Operations on List

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to perform different operations on list.

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**.

Scala REPL

Scala REPL

Scala REPL is a tool for evaluating expressions in Scala. The REPL reads expressions at the prompt, wraps them in an executable template, and then compiles and executes the result.

The following are the features of REPL:

- **$intp**
- **//print<tab>**
- **:load**
- **lastException**
- **:help**
- **:paste**

Scala REPL

Few more features of REPL:

- :paste -raw
- -Yrepl-outdir
- :settings
- :javap
- :power
- :replay

Implementation Notes

1. User code can be wrapped in either an object or a class. The switch is -Yrepl-class-based.
2. Every line is compiled individually.
3. Automatically generated imports include the dependencies on previous lines.
4. Implicit import of scala.Predef can be controlled by inputting an explicit import.

Example of Scala REPL

```
scala> import Predef.{any2stringadd => _, _}
import Predef.{any2stringadd=>_, _}
scala> new Object + "a string"
<console>:13: error: value + is not a member of Object
          new Object + "a string"
                       ^
scala> import Predef._ import Predef._
scala> new Object + "a string"
res1: String = java.lang.Object@787a0fd6a string
```

Demonstrate the Features of Scala REPL

Duration: 10 mins

Problem Statement: In this demonstration, you will demonstrate the features of Scala REPL.

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:

- Describe and demonstrate programming with Scala
- Define functions, anonymous function, and class in Scala
- Use the different types of collections
- Describe and demonstrate Scala REPL

Quiz

Knowledge Check 1

scala> val hex = 0x6; output - hex: Int = 6 is an example of which of the following literals?

a. Character literal

b. Integer literal

c. Floating literal

d. None of the above

Knowledge Check b	scala> val hex = 0x6; output - hex: Int = 6 is an example of which of the following literals?

a. **Character literal**

b. **Integer literal**

c. Floating literal

d. None of the above

scala> val hex = 0x6; output - hex: Int = 6 is an example of integer literal.

The correct answer is

Knowledge Check 2	_____ is a value that contains a fixed number of elements, each with a distinct type.

a. Map

b. Tuple

c. List

d. Set

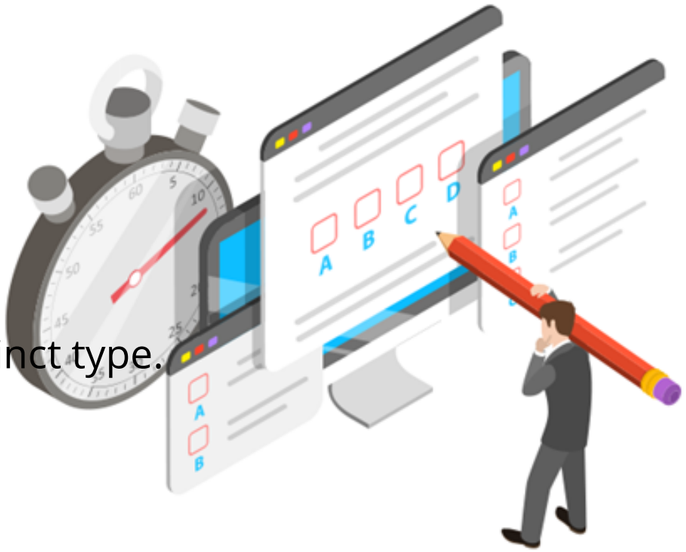
Knowledge Check b	_____ is a value that contains a fixed number of elements, each with a distinct type.

a. **Map**

b. **Tuple**

c. List

d. Set



Tuple is a value that contains a fixed number of elements, each with a distinct type.

The correct answer is

Knowledge Check 3

_____ is a way to access the elements of a collection one by one.

a. Option

b. Iterator

c. Map

d. Set

Knowledge Check b	_____ is a way to access the elements of a collection one by one.

a. **Option**

b. **Iterator**

c. Map

d. Set

Iterator is a way to access the elements of a collection one by one.

The correct answer is

Lesson-End Project

Problem Statement: Scala is one of the most popular languages to work with Spark. One of the advantages of Scala is that it runs on JVM. It supports both object-oriented and functional programming. Mostly Scala is used for functional programming. It provides java-like features such as collections, file handling & exception handling, etc.
“People data Labs” is one of the biggest data organizations that collects and provides data for the companies. It has open-sourced data about companies that operate in different countries. You must find some important facts about companies based on the year they were founded in, their employee count and country, etc.

Lesson-End Project

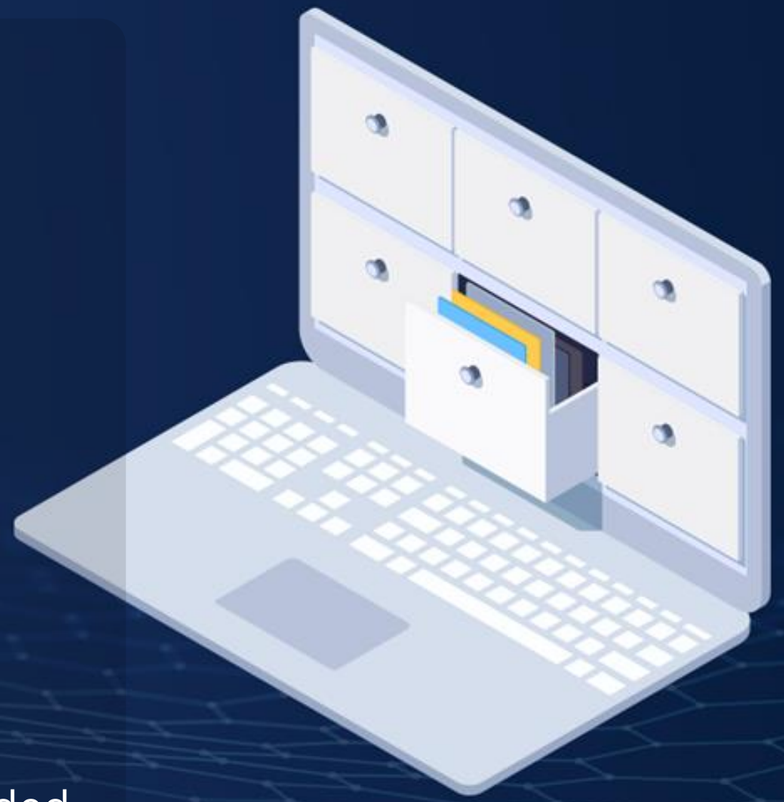
The following details are given for the list of companies in a CSV file.

1. Name
2. Domain
3. Year founded
4. Industry
5. Size range
6. Country
7. LinkedIn URL
8. Current employee estimate
9. Total employee estimate

Filename: companies.csv

You must read the file and use Scala collections to solve the following problems:

1. Companies which were founded before 1980.
2. Top 5 companies which have the maximum number of employees.
3. Top Industry in which the maximum number of companies were founded.



DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Apache Spark - Next Generation Big Data Framework

Learning Objectives

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

- Know the history of Spark
- Understand the advantages of Spark
- Interpret the companies implementing Spark with use cases
- Understand Spark and its core components
- Learn Spark's architecture
- Use Spark cluster in real world - Development, QA, and Production

DATA AND
ARTIFICIAL INTELLIGENCE

History of Spark

History of Spark

Started at UC Berkeley AMPLab by Matei Zaharia

2009

amplab UC BERKELEY

2010

BSD

Open sourced under a BSD license

Given to the Apache Software Foundation and the license was changed to Apache 2.0

2013

The Apache Software Foundation

2014

databricks

Became an Apache top-level project. Used by Databricks to set a world record in large-scale sorting

Exists as a next generation real-time and batch processing framework

Present

Batch vs. Real-Time Processing

Batch Processing

- A large group of data or transactions is processed in a single run.
- Jobs are run without any manual intervention.
- The entire data is pre-selected and fed using command-line parameters and scripts.
- It is used to execute multiple operations, handle heavy data load, reporting, and offline data workflow.

Example:
Regular reports that require decision-making

Real-Time Processing

- Data processing takes place upon data entry or command receipt instantaneously.
- It must execute real-time within stringent constraints.

Example: Fraud detection

Limitations of MapReduce in Hadoop

Unsuitable for real-time processing

Being batch oriented, it takes minutes to execute jobs depending on the amount of data and number of nodes in the cluster.

Unsuitable for trivial operations

For operations like Filter and Joins, you might need to rewrite the jobs, which becomes complex because of the key-value pattern.

Unsuitable for large data on network

Since it works on the data locality principle, it cannot process a lot of data that requires shuffling over the network.

Limitations of MapReduce in Hadoop

Unsuitable with OLTP

OLTP requires a large number of short transactions, as it works on the batch-oriented framework.

Unsuitable for processing graphs

The Apache Graph library processes graphs, that adds additional complexity on top of MapReduce.

Unsuitable for iterative execution

Being a stateless execution, MapReduce doesn't fit in use cases like k-means that need iterative execution.

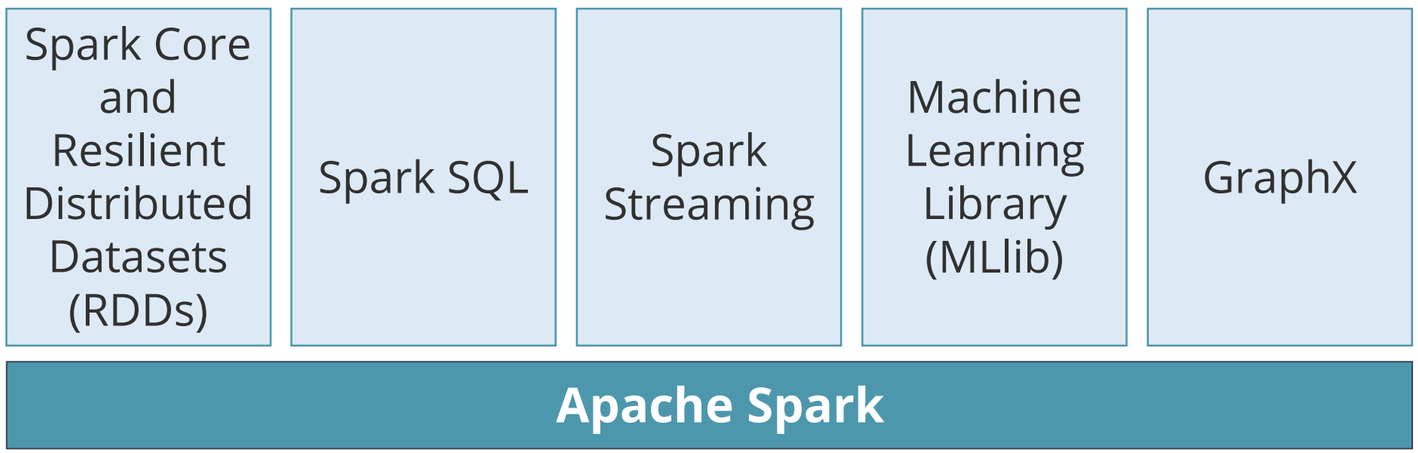
Introduction to Apache Spark

Is suitable for real-time processing, trivial operations, and processing larger data on a network

Is an open source cluster computing framework

Provides up to 100 times faster performance for a few applications with in-memory primitives, compared to the two-stage disk-based MapReduce paradigm of Hadoop

Is suitable for machine learning algorithms, as it allows programs to load and query data repeatedly



DATA AND
ARTIFICIAL INTELLIGENCE

Components of Spark

Components of a Spark Project

The components of a Spark project are explained below:

Spark Core and RDDs

As the foundation, it provides basic I/O, distributed task dispatching, and scheduling. RDDs can be created by applying coarse-grained transformations or referencing external datasets.

Spark SQL

As a component lying on the top of Spark Core, it introduces SchemaRDD, which can be manipulated. It supports SQL with ODBC/JDBC server and command-line interfaces.

Components of a Spark Project

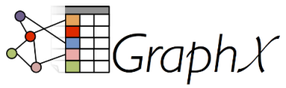
The components of a Spark project are explained below:

Spark Streaming

It leverages the fast scheduling capability of Spark Core, ingests data in small batches, and performs RDD transformations on them.

MLlib

As a distributed machine learning framework on top of Spark, it is nine times faster than the Hadoop disk-based version of Apache Mahout.



GraphX

Being a distributed graph processing framework on top of Spark, it gives an API and provides an optimized runtime for the Pregel abstraction.

Application of In-Memory Processing

In column-centric databases, informations that are similar, can be stored together. The working of in-memory processing can be explained as below:

The entire information is loaded into memory, eliminating the need for indexes, aggregates, optimized databases, star schemas, and cubes.

Compression algorithms are used by most of the in-memory tools, thereby reducing the in-memory size.

Querying the data loaded into the memory is different from caching.

With in-memory tools, the analysis of data can be flexible in size and can be accessed within seconds by concurrent users with an excellent analytics potential.

It is possible to access visually rich dashboards and existing data sources.

Language Flexibility in Spark

Spark is popular for its performance benefits over MapReduce.
Another important benefit is language flexibility, as explained below:

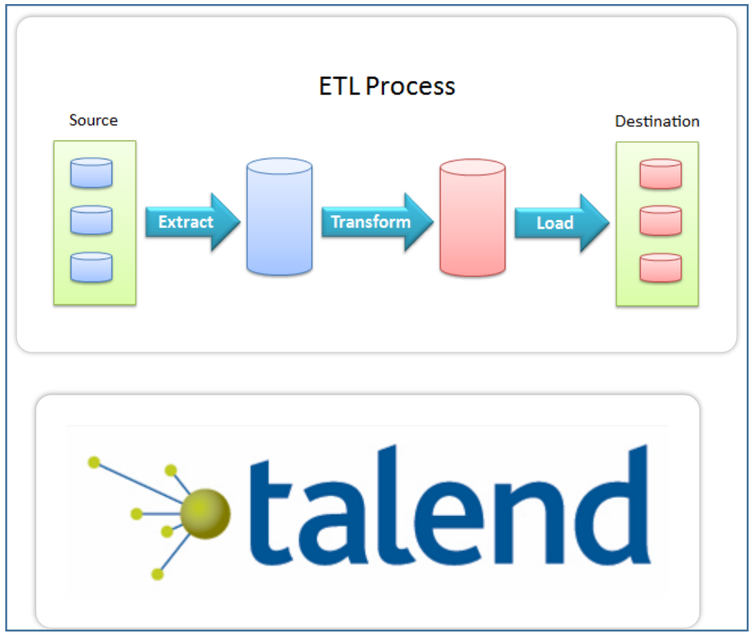
Support for various development languages

Spark supports popular development languages like Java, Scala, and Python and will support R.

Capability to define functions in-line

With the temporary exception of Java, a common element in these languages is that, they provide methods to express operations using lambda functions and closures.

Hadoop Ecosystem vs. Spark



Hadoop Ecosystem

Hadoop Ecosystem vs. Spark

You can perform every type of data processing using Spark that you execute in Hadoop. They are:

Batch Processing: Spark batch can be used over Hadoop MapReduce.

Structured Data Analysis: Spark SQL can be used with SQL.

Machine Learning Analysis: MLlib can be used for clustering, recommendations, and classification.

Interactive SQL Analysis: Spark SQL can be used over Stringer, Tez, or Impala.

Real-time Streaming Data Analysis: Spark streaming can be used over specialized library like Storm.

DATA AND
ARTIFICIAL INTELLIGENCE

Advantages of Spark

Advantages of Spark

The different advantages of Spark are:

Speed: Extends the MapReduce model to support computations like stream processing and interactive queries

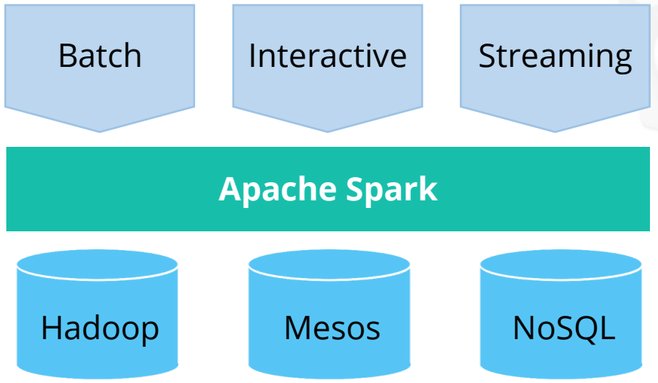
Combination: Covers various workloads that require different distributed systems, which makes it easy to combine different processing types and allows easy tools management

Hadoop Support: Allows creation of distributed datasets from any file stored in the Hadoop Distributed File System (HDFS) or any other supported storage systems

?

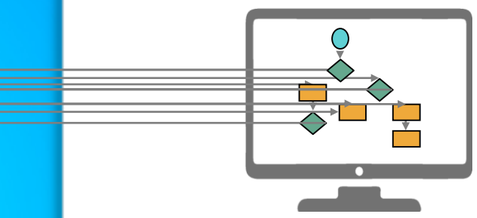
Why does unification matter?

- Developers need to learn only one platform
- Users can take their apps everywhere



Advantages of Spark

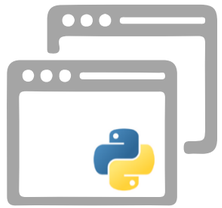
Some more advantages of Spark are:



Contains various closely integrated components for distributing, scheduling, and monitoring applications with many computational tasks

Empowers various higher level components specialized for different workloads like machine learning or SQL

Integrates and easily combines different processing models; for example, ability to write an application using machine learning to categorize data in real time as it is ingested from sources of streaming



Allows to access the same data through the Python shell for ad hoc analysis and in standalone batch applications

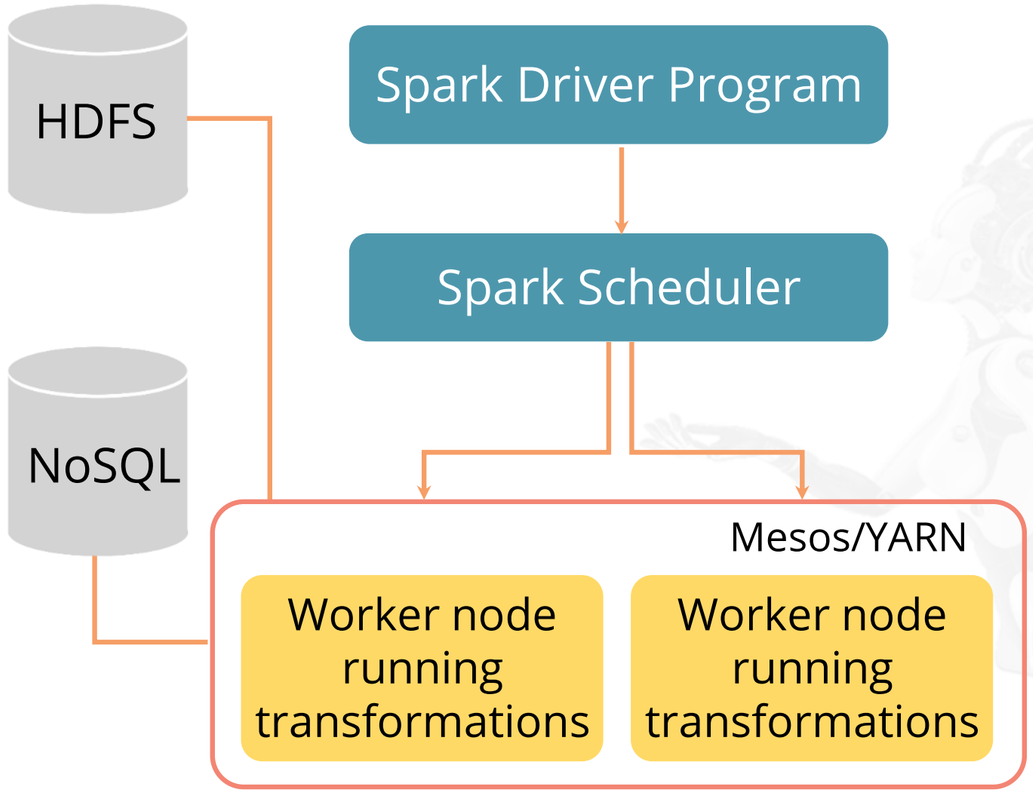
DATA AND
ARTIFICIAL INTELLIGENCE

Spark Architecture

Spark Architecture

The components of the Spark execution architecture are explained below:

- **Spark-submit script:** Used to launch applications on a cluster; can use all cluster managers through a uniform interface
- **Spark applications:** Run as independent sets of processes on a cluster and are coordinated by the SparkContext object in the driver program
- **Cluster managers:** Supported cluster managers are Standalone, Apache Mesos, and Hadoop YARN
- **Spark's EC2 launch scripts:** Make launching a standalone cluster easy on Amazon EC2



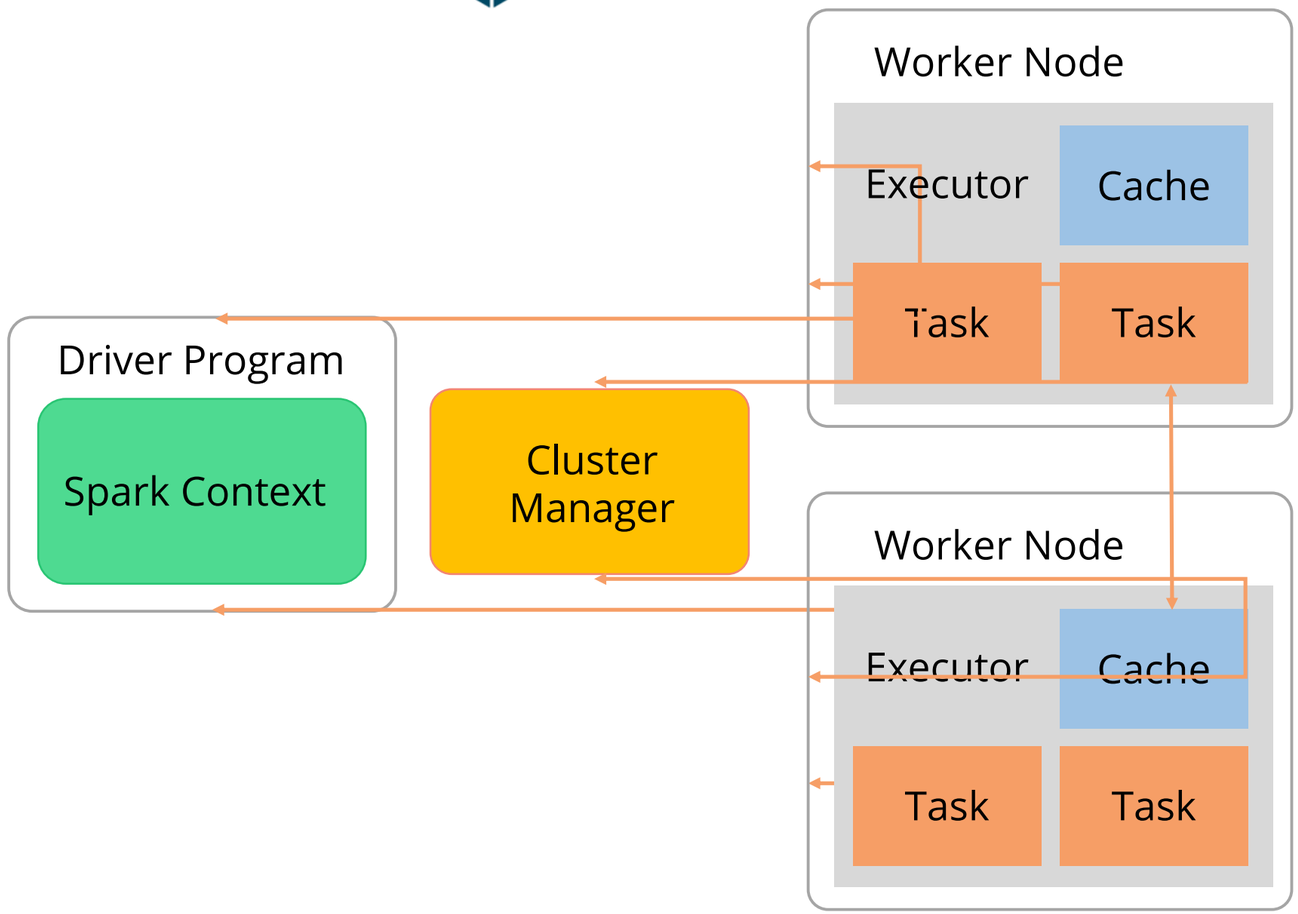
Spark Execution Architecture

The components of the Spark execution architecture are explained below:

Spark
Standalone Cluster

Apache MESOS™

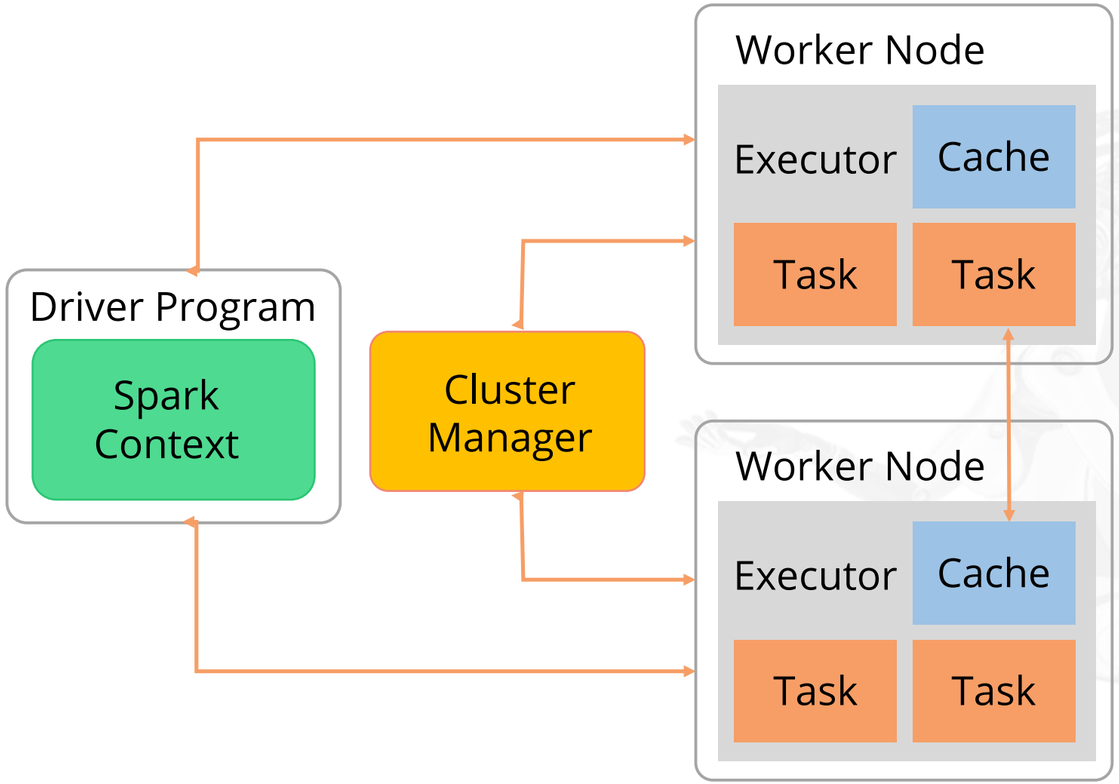
hadoop YARN



Spark Execution: Automatic Parallelization

Spark Execution is explained as follows:

- To make the sequence of MapReduce jobs parallel in case of a complex pipeline, a scheduler tool like Apache Oozie is generally required.
- The series of individual tasks is expressed as a single program flow, which allows to parallelize the flow of operators automatically without any intervention.
- This allows certain optimizations to the engine.



DATA AND
ARTIFICIAL INTELLIGENCE

Spark Cluster in Real World

Running Spark in Different Modes

The different deployment modes of Spark are explained below:

Spark as Standalone

Can be launched manually by using launch scripts, or starting a master and workers; used for development and testing

Spark on Mesos

Has advantages like scalable partitioning among different Spark instances and dynamic partitioning between Spark and other frameworks

Spark on YARN

Has all parallel processing and benefits of the Hadoop cluster

Spark on EC2

Has key-value pair benefits of Amazon

Spark Shell

The Spark Shell provides interactive data exploration (REPL).

$Spark-shell

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.3.0
      /_/

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
```

$pySpark

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.3.0
      /_/

Using Python version 2.7.8 (default, Aug 27 2015 05:23:36)
SparkContext available as sc, HiveContext available as sqlCtx.

In [1]:
```

SparkContext

- It is the main entry point of Spark API. Every Spark application requires a SparkContext.
- Spark Shell provides a preconfigured SparkContext called sc.

Scala

$Spark-shell

```
Spark context available as sc.
16/08/10 04:42:03 INFO repl.SparkILoop: Created sql context (with Hive support).
.
SQL context available as sqlContext.

scala>
```

python

$pySpark

```
Using Python version 2.7.8 (default, Aug 27 2015 05:23:36)
SparkContext available as sc, HiveContext available as sqlCtx.

In [1]:
```

Assisted Practice

Running a Scala Program in Spark Shell

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to run a Scala program in Spark shell.

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**.

Assisted Practice

Setting Up Execution Environment in IDE

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to set up an execution environment in IDE.

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**.

Assisted Practice

Understanding Various Components of Spark Web UI

Duration: 10 mins

Problem Statement: In this demonstration, you will understand the various components of Spark web UI.

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:

- Know the history of Spark
- Understand the advantages of Spark
- Interpret the companies implementing Spark with use cases
- Understand Spark and its core components
- Learn Spark's architecture
- Use Spark cluster in real world - Development, QA, and Production environments

DATA AND
ARTIFICIAL INTELLIGENCE

Knowledge Check

Knowledge Check 1

Which of the following are the components of Spark project?

a. Spark Core and RDDs

b. Spark SQL

c. Spark Streaming

d. All of the above

Knowledge Check 1

Which of the following are the components of Spark project?

a. Spark Core and RDDs

b. Spark SQL

c. Spark Streaming

d. All of the above

The correct answer is **d.**

Spark Core and RDDs, Spark SQL, and Spark Streaming are some of the components of Spark project.

Knowledge Check 2

Spark was started in the year_____.

a. 2009

b. 2010

c. 2013

d. 2014

Knowledge Check 2

Spark was started in the year_____.

a. 2009

b. 2010

c. 2013

d. 2014

The correct answer is **a.**

Spark was started in the year 2009 at UC Berkeley AMPLab by Matei Zaharia.

Knowledge Check 3

Which of the following are the supported Cluster Managers?

a. Standalone

b. Apache Mesos

c. Hadoop Yarn

d. All of the above

Knowledge Check 3

Which of the following are the supported Cluster Managers?

a. Standalone

b. Apache Mesos

c. Hadoop Yarn

d. All of the above

The correct answer is **d.**

Standalone, Apache Mesos, and Hadoop Yarn are all supported Cluster Managers.

Lesson-End Project

Problem Statement:

You have enrolled as a trainee in one of the top training institutes which provides training on Big Data. You have learned Spark and Hadoop and where to use them.
Based on that knowledge, your task is to solve the below use cases using Hadoop or Spark.

Use-cases:

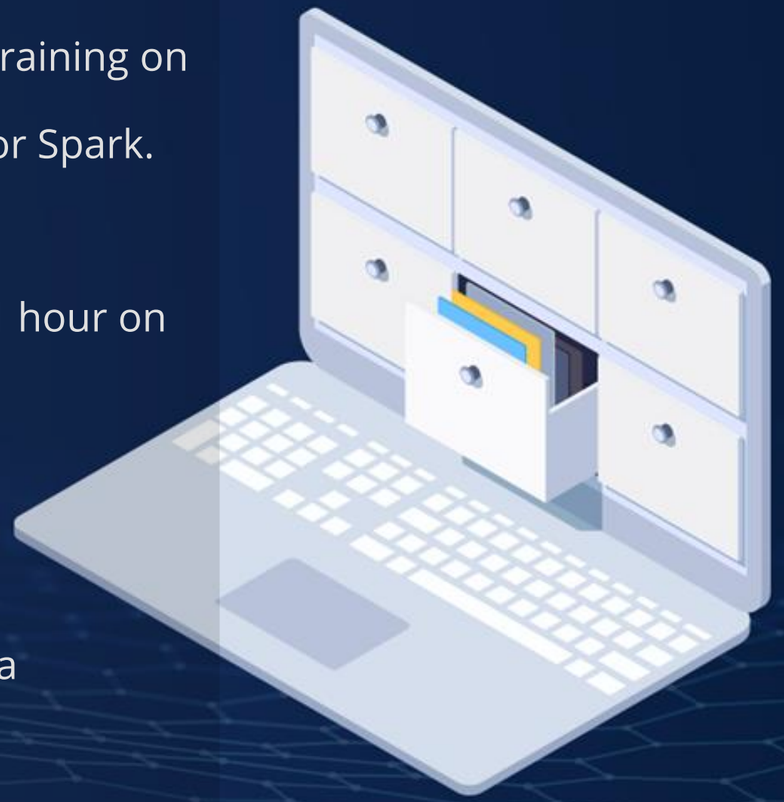
1. An E-commerce company wants to show the most trending brands in the last 1 hour on their web portal.

2. An E-commerce company wants to calculate orders for the last 5 years in the mobile category.

3. You have been given the product data of clicks and impressions. Click means when a user clicks on the product and goes to the product page.
   Impression refers to the product landing page on Amazon. You have to create a model that can predict if any product on the portal is eligible for click or not.

4. An E-commerce company wants to show the most trending products on their web portal in real-time.

5. A financial institution wants to check if the transaction is fraud or not.



DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Processing RDDs

Learning Objectives

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

- Define Spark RDD and list its limitations
- Describe and demonstrate RDD operations in Spark
- Demonstrate the creation of Spark RDD
- Aggregate data with pair RDD

DATA AND
ARTIFICIAL INTELLIGENCE

Introduction to Spark RDD

Spark RDD

Spark Resilient Distributed Dataset (RDD) is an immutable collection of objects which defines the data structure of Spark.

The following are the features of Spark RDD:

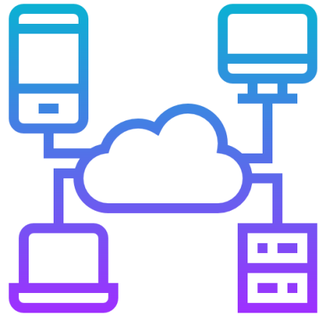
RDD in Spark

The following are the reasons for the evolution of RDD:

1. Iterative algorithm
2. Interactive data mining tools
3. Inefficient implementation of Distributed Shared Memory (DSM)
4. Slower computation when distributed computing systems store data in HDFS or Amazon S3

Limitations of Spark RDD

Lack of Optimization Engine



Structured Data Handling

Performance

Storage

Data Types Supported by RDD

Primitive

- Integer
- Character
- Boolean

Sequence

- Strings
- Lists
- Arrays
- Tuples
- Dicts
- Nested

Java and Scala objects

Mixed Data

DATA AND
ARTIFICIAL INTELLIGENCE

Creating Spark RDD

Creating Spark RDD

Parallelized Collections

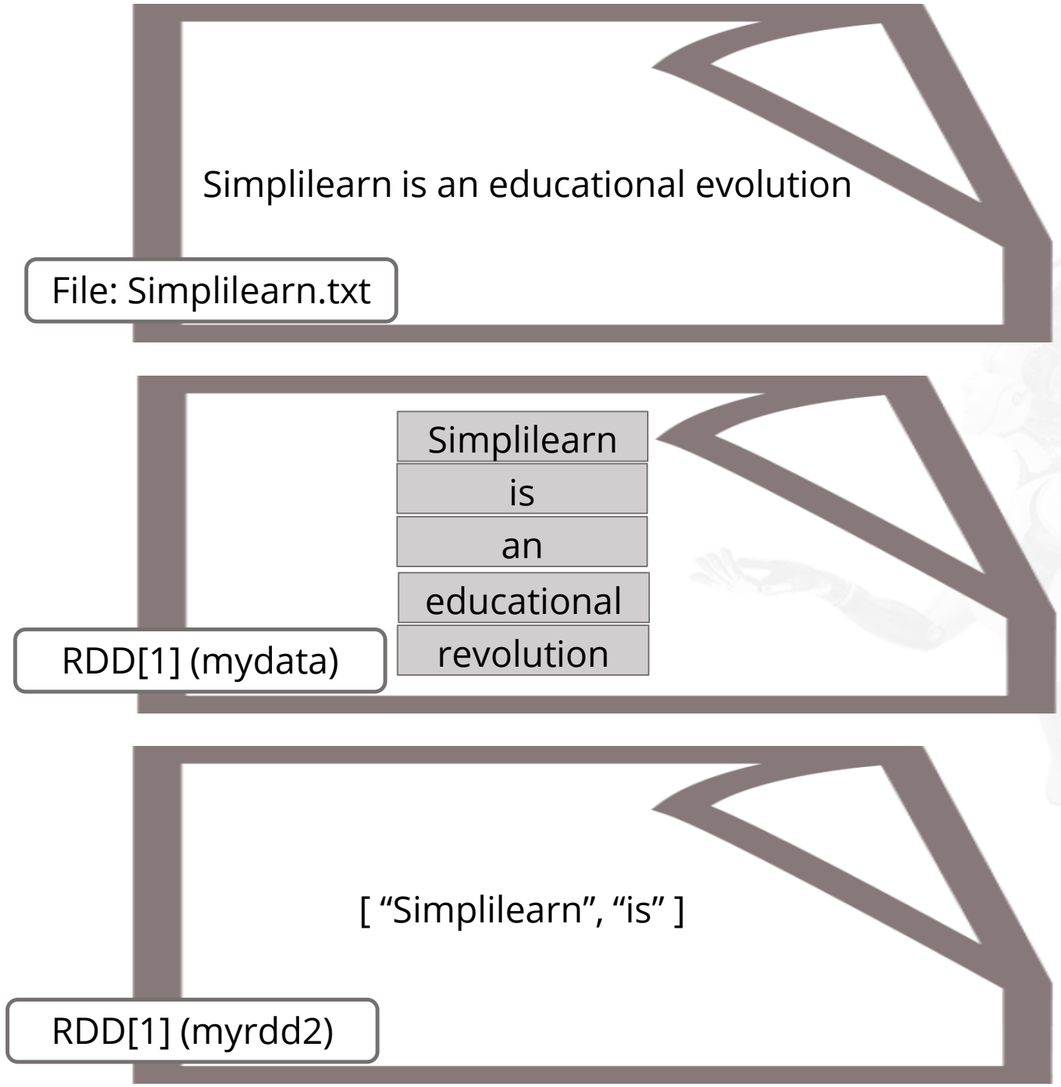
RDDs are created by parallelizing an existing collection in your driver program or referencing a dataset in an external storage system.

RDDs are created by taking the existing collection and passing it to SparkContext parallelize() method.

```
val data=spark.sparkContext.parallelize(Seq((“physics",78),(“chemistry",48),(“biology",73),
(“english",54),("maths",77)))
val sorted = data.sortByKey()
sorted.foreach(println)
```

Creating RDD from Collections

```
training@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
training@localhost:~

In [4]: data = [“Simplilearn”, “is”, “an”, “educational”,
“revolution”]

In [5]: rdd1 = sc.parllelize(data)

In [6]: rdd1.take(2) [“Simplilearn”, “is”]
```



Existing RDDs

RDDs can be created from existing RDDs by transforming one RDD into another RDD.

```
val words=spark.sparkContext.parallelize(Seq("Simplilearn", "is", "an", "education", "provider"))

val wordPair = words.map(w => (w.charAt(0), w))

wordPair.foreach(println)
```

External Data

In Spark, a dataset can be created from any other dataset. The other dataset must be supported by Hadoop, including the local file system, HDFS, Cassandra, HBase, and many more.

Data frame reader interface can be used to load dataset from an external storage system in the following formats:

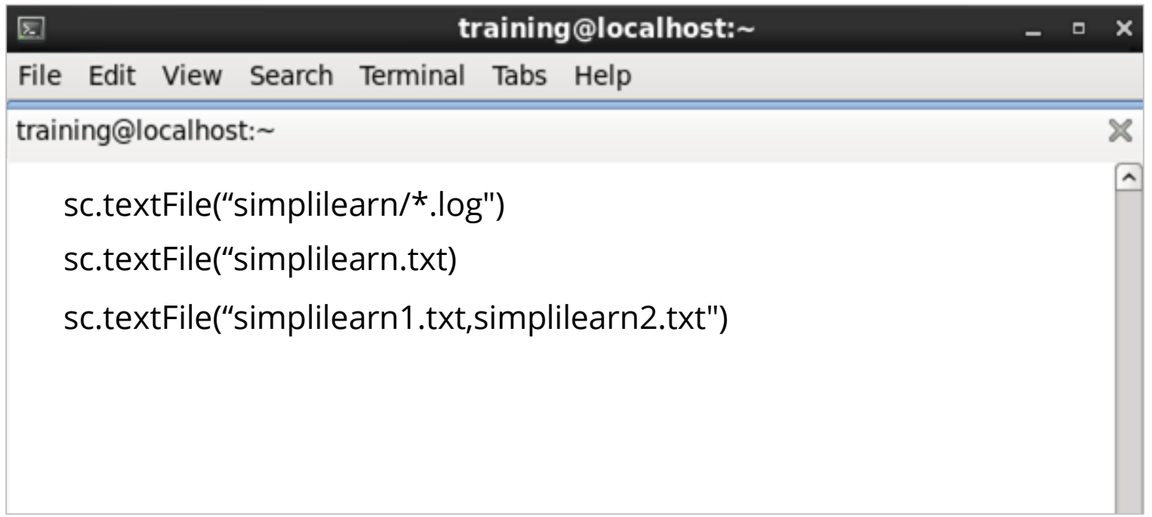
CSV	val dataRDD = spark.read.csv("path/of/csv/file").rdd
JSON	val dataRDD = spark.read.json("path/of/json/file").rdd
Textfile	val dataRDD = spark.read.textFile("path/of/text/file").rdd

Creating RDD from a Text File

To create a file-based RDD, you can use the command SparkContext.textFile or sc.textfile, and pass one or more file names.

Text File

Data in Memory



Creating RDD from a Text File

Simplilearn is an education provider.\n

It is based in San Francisco. \n

It has trained 450,000+ customers. \n

It offers 400+ professional courses. \n

Simplilearn is an education provider.

It is based in San Francisco.

It has trained 450,000+ customers.

It offers 400+ professional courses.

Creating RDD from a Text File

(file1.json { “firstname”: “Rahul”,“lastname”: “Gupta”, “customerid”: “001”})

(file2.json { “firstname”: “Rita”,“lastname”: “John”, “customerid”: “002”})

(file3.json { “firstname”: “Sam”,“lastname”: “Grant”, “customerid”: “003”})

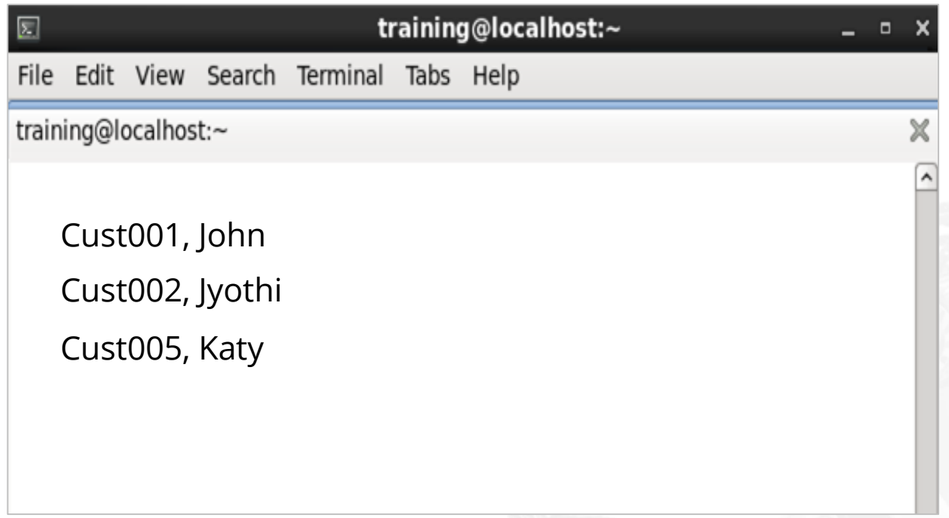
DATA AND
ARTIFICIAL INTELLIGENCE

Pair RDD

Pair RDD and Double RDD

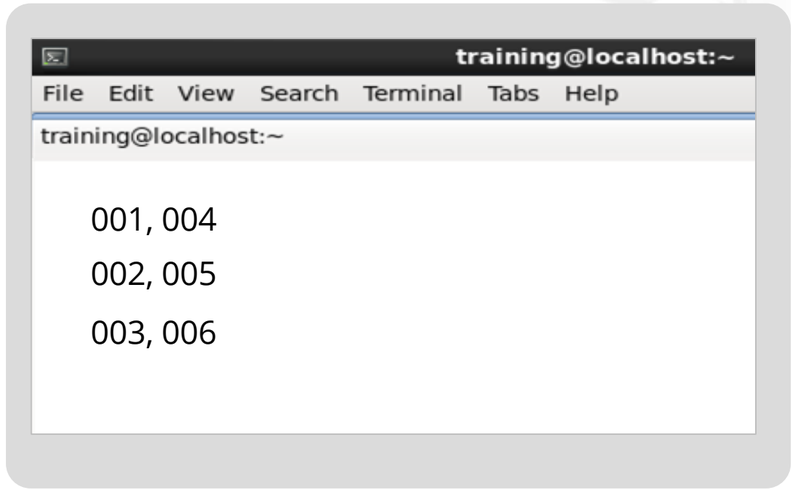
Pair RDDs

Some of the functions that can be performed with Pair RDDs are Map and flatMap.



Double RDDs

Double RDDs are RDDs that hold numerical data. Some of the functions that can be performed with Double RDDs are Distinct and Sum.



Creating Pair RDD

To create a Pair RDD, use functions such as Map, flatMap or flatMapValues, and keyBy.

```
Cust001 \t Deepak Mehta
Cust002 \t Seema Arora
Cust003 \t Asha Rao
```

Language: Python

```
Users = sc.textFile(file) \
.map (lambda line: line.split ('\t')) \
 .map (lambda fields: (fields [0], fields [1]))
```

Language: Scala

```
Val users = sc.textFile(file) .
    .map (line => line.split ('\t').
.map (fields => (fields (0), fields (1) ))
```

(Cust001, Deepak Mehta)

(Cust002, Seema Arora)

(Cust003, Asha Rao)

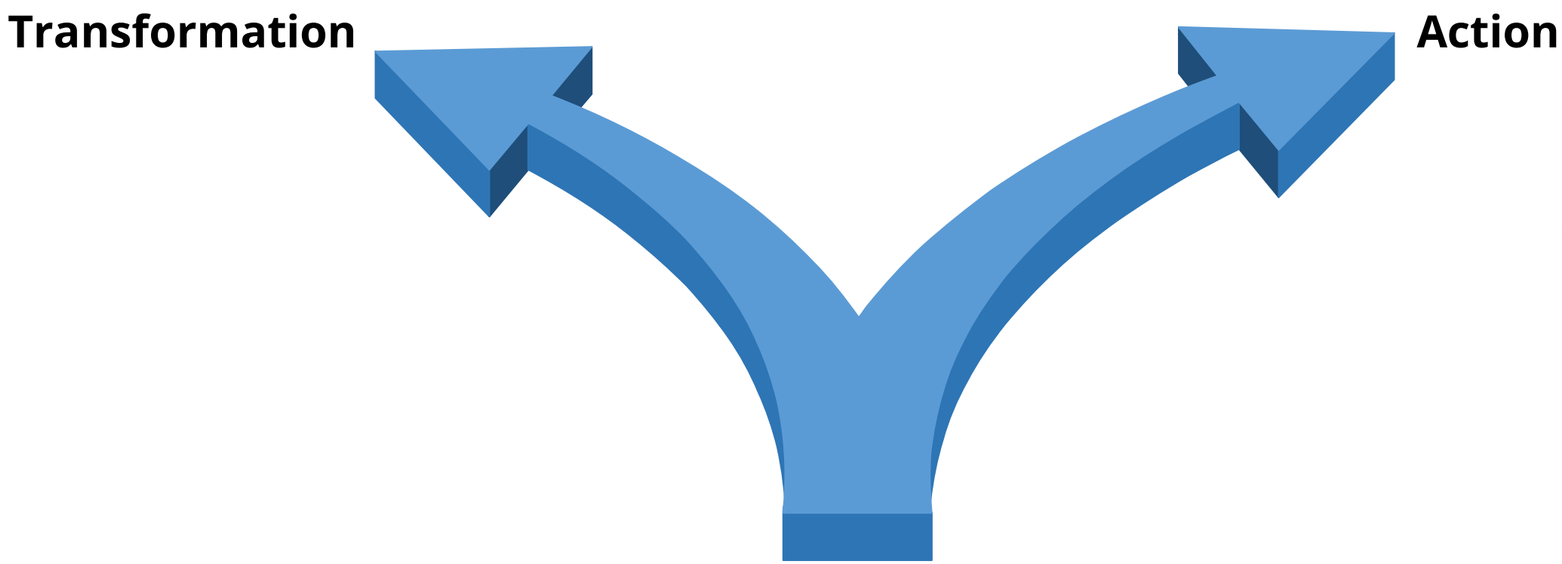
Pair RDD

DATA AND
ARTIFICIAL INTELLIGENCE

RDD Operations

RDD Operations

RDD supports the following types of operations:



Transformation

Transformation allows us to create new dataset using the existing one.

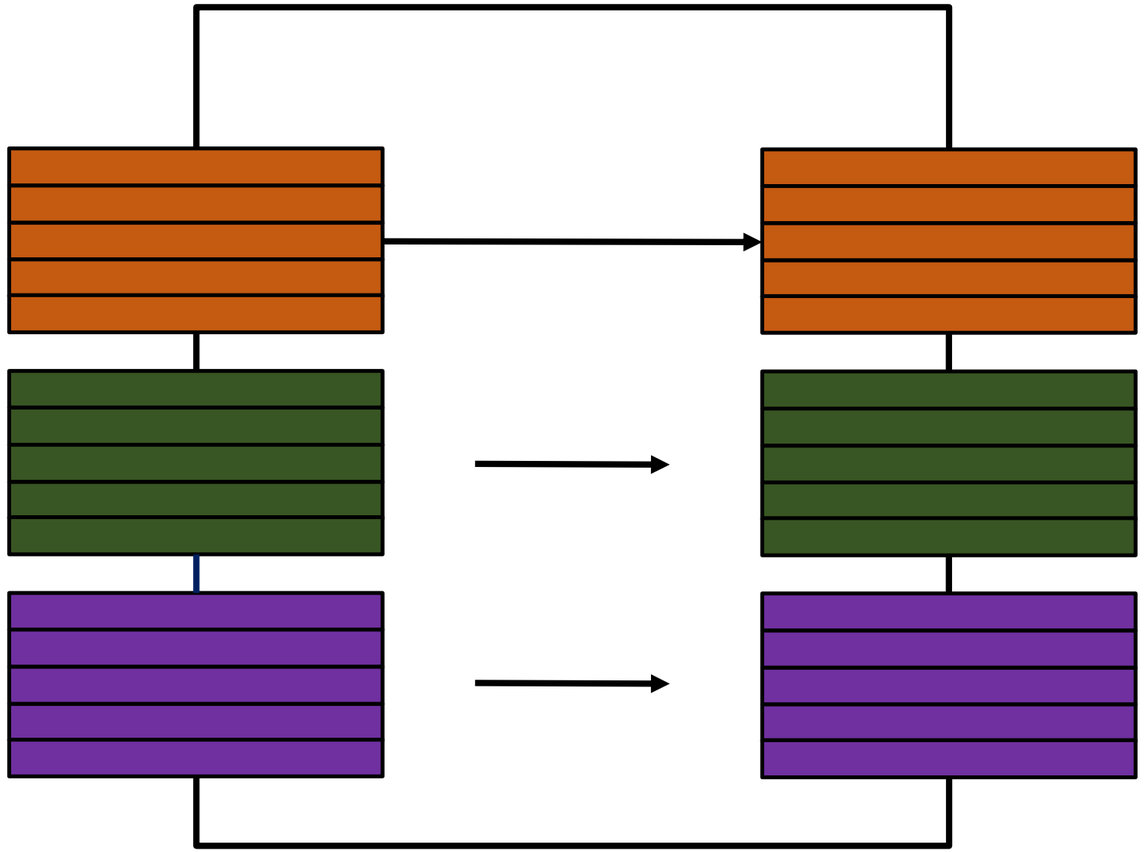
There are two types of transformation:

Narrow Transformation

Wide Transformation

Narrow Transformation

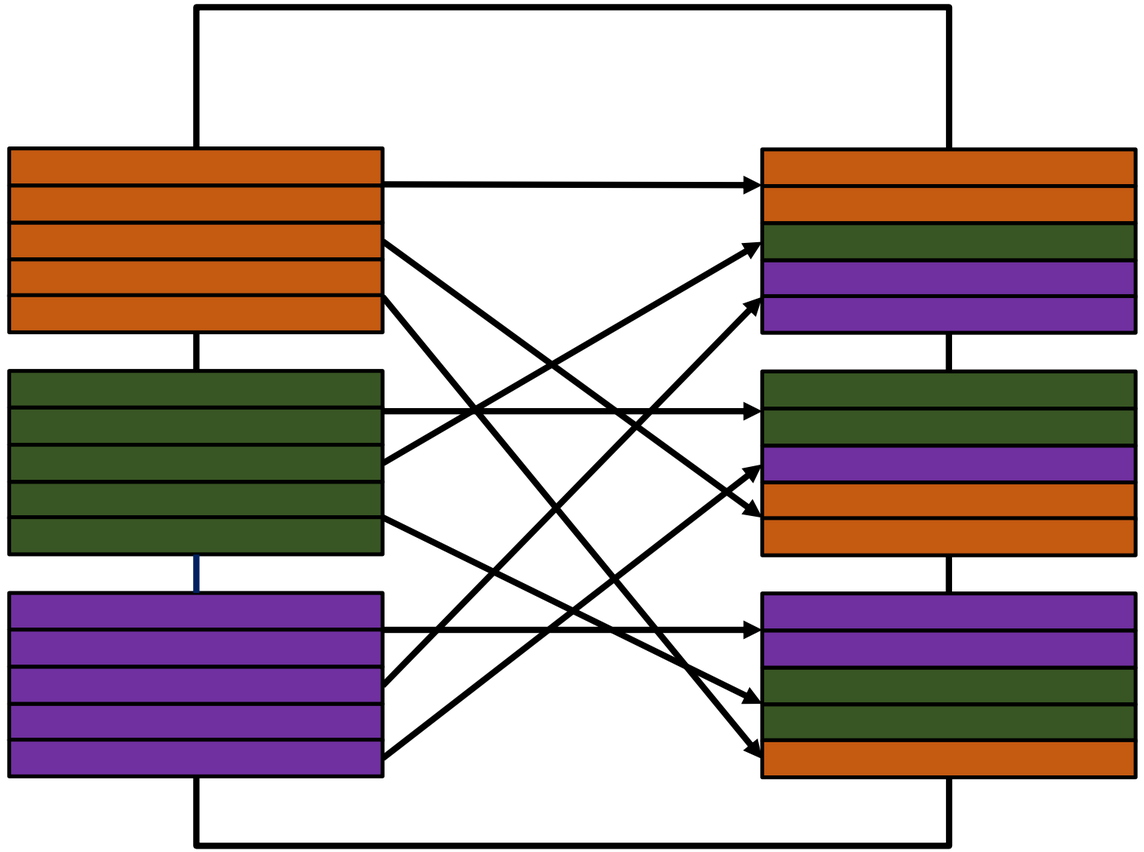
Narrow transformation is self-sufficient. It is the result of map and filter, such that the data is from a single partition only.



- Map
- Filter
- FlatMap
- Sample
- MapPartition
- Union

Wide Transformation

Wide transformation is not self-sufficient. It is the result of GroupByKey() and ReduceByKey() like functions, such that the data can be from multiple partitions.



- Cartesian
- Join
- Intersection
- Repartition
- ReduceByKey
- GroupByKey
- Coalesce
- Distinct

Spark Transformation: Detailed Exploration

Duration: 10 mins

Problem Statement: In this demonstration, you will explore Spark transformation.

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**.

Action

Action allows us to return a value to the driver program, after running a computation on the dataset. Actions are the RDD operations that produce non-RDD values.

Reduce	Reduce is an action that aggregates all the elements of the RDD using some function.

- **First()**
- **Take()**
- **Reduce()**
- **Collect()**
- **Count()**

Spark Action: Detailed Exploration

Duration: 10 mins

Problem Statement: In this demonstration, you will explore Spark action.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

Caching and Persistence

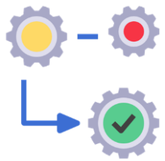
Caching and Persistence

Caching and Persistence are the techniques used to store the result of RDD evaluation. They are also used by developers to enhance the performance of applications.

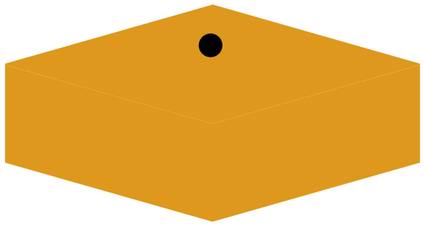
The following are the benefits of Caching and Persistence:

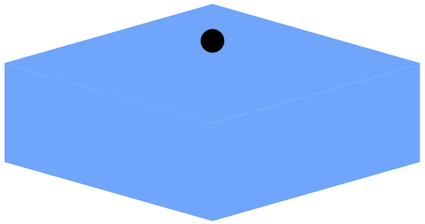
Cost Efficient

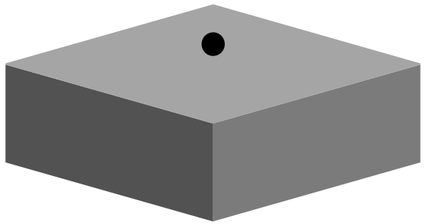
Time Efficient



Less Execution Time

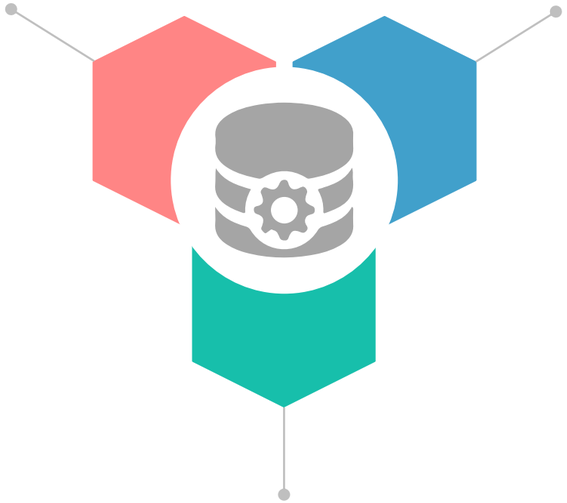






Features of RDD Persistence

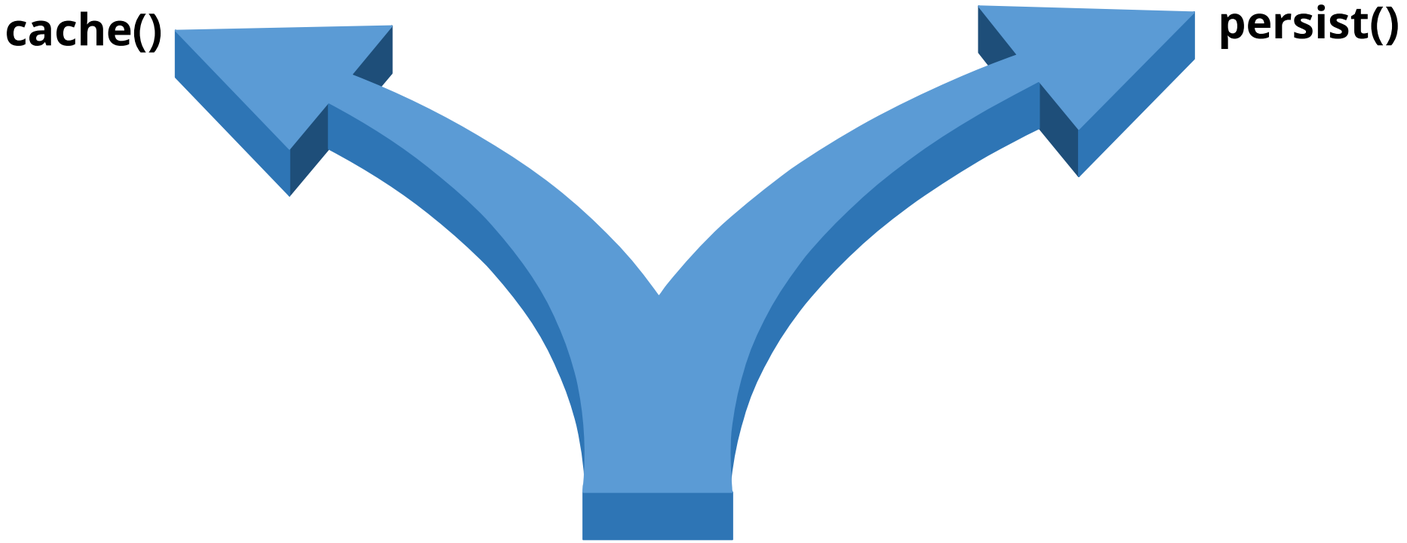
Storage and reuse of the RDD partitions



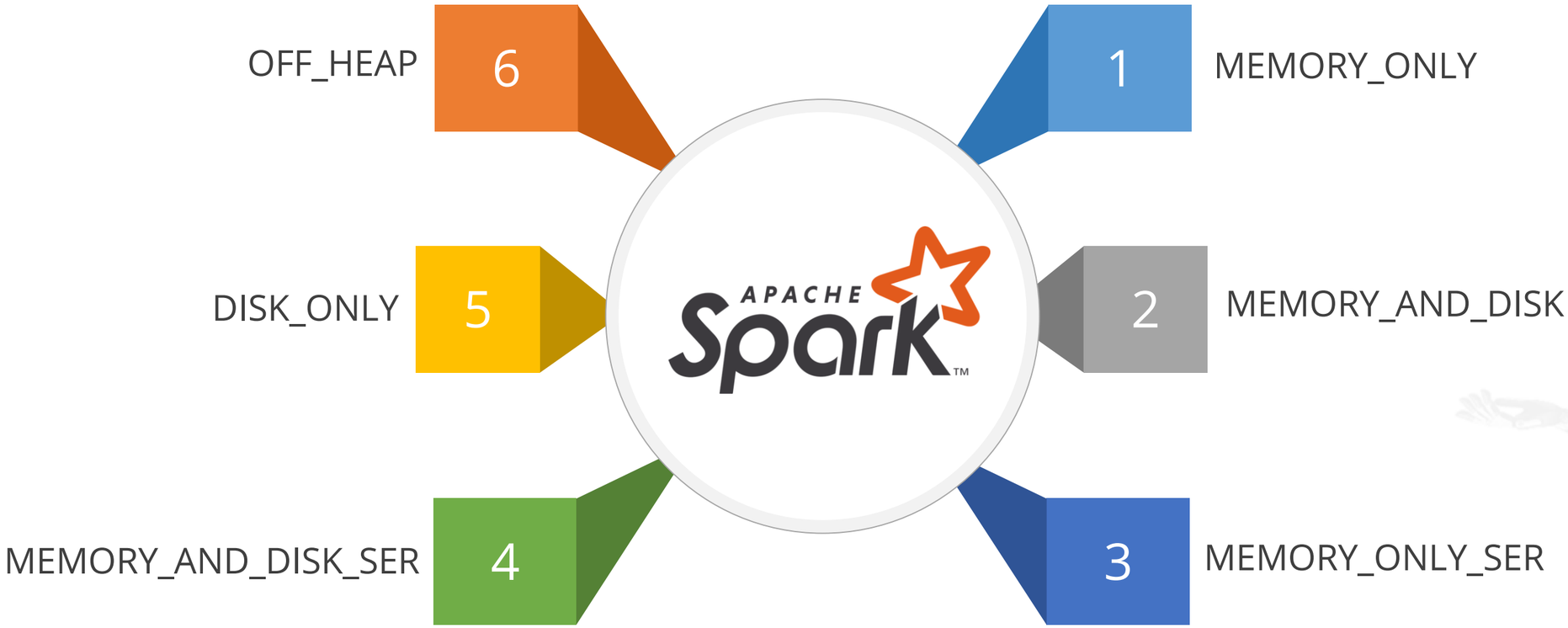
Automatic recomputation of lost RDD partitions

Storage of persisted RDDs on different storage levels

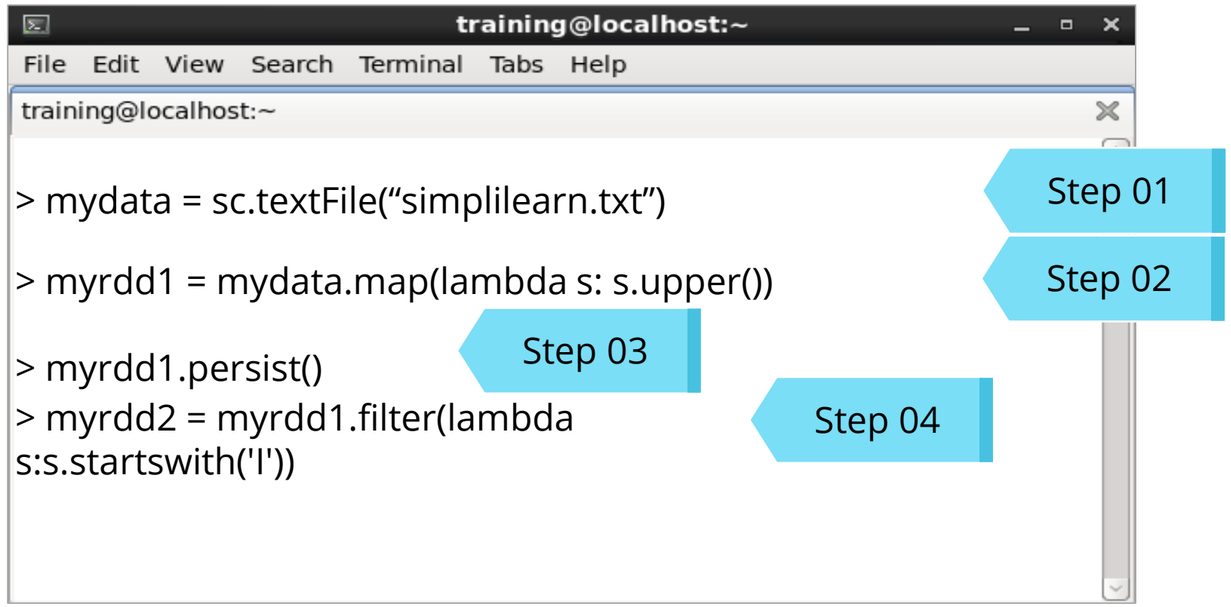
Methods of Caching and Persistence

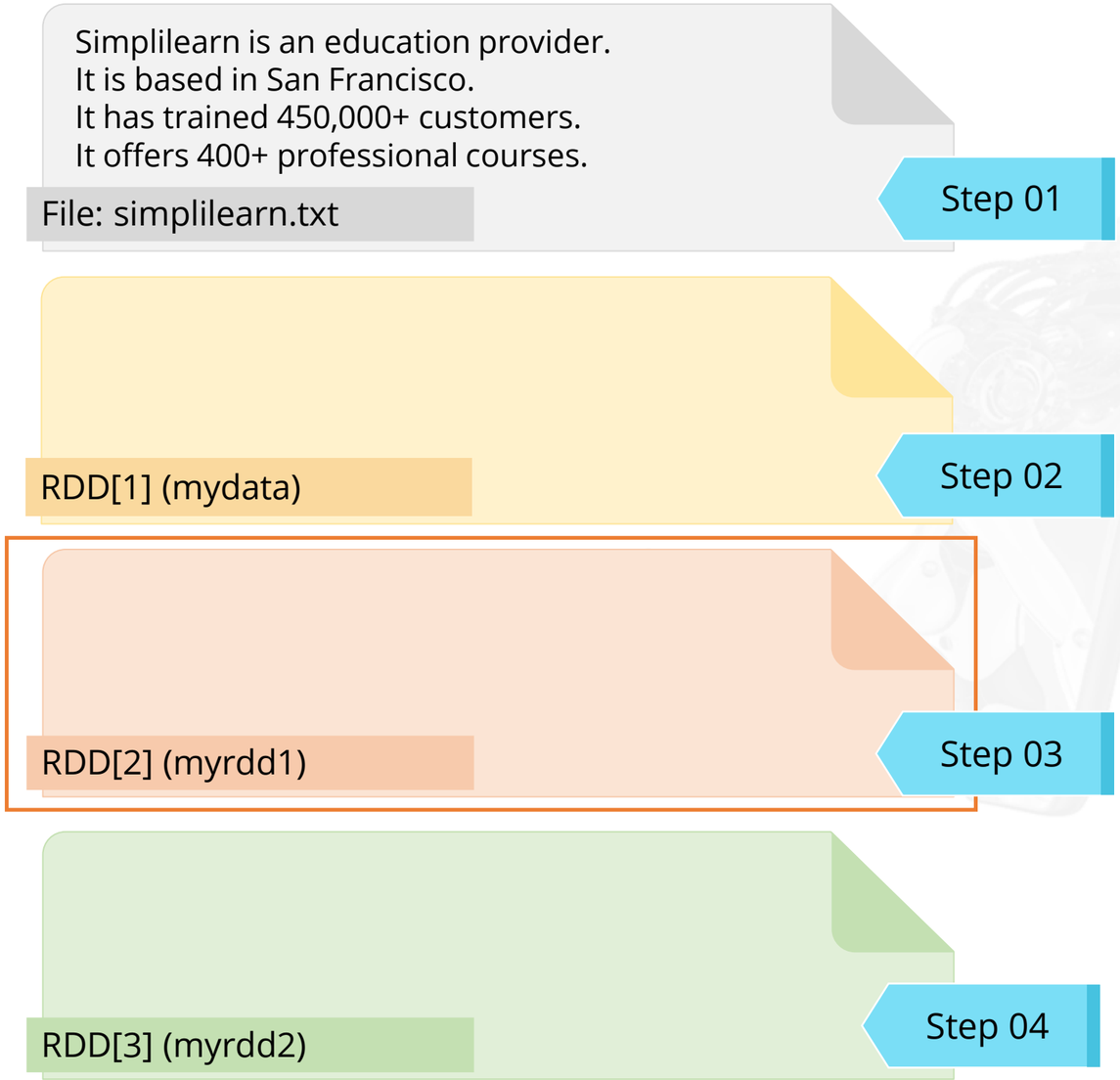


Storage Levels



Marking an RDD for Persistence





Changing Persistence Options

To change persistence option on an RDD:

Use rdd.Unpersist()

To change the RDD persistence to different storage level:

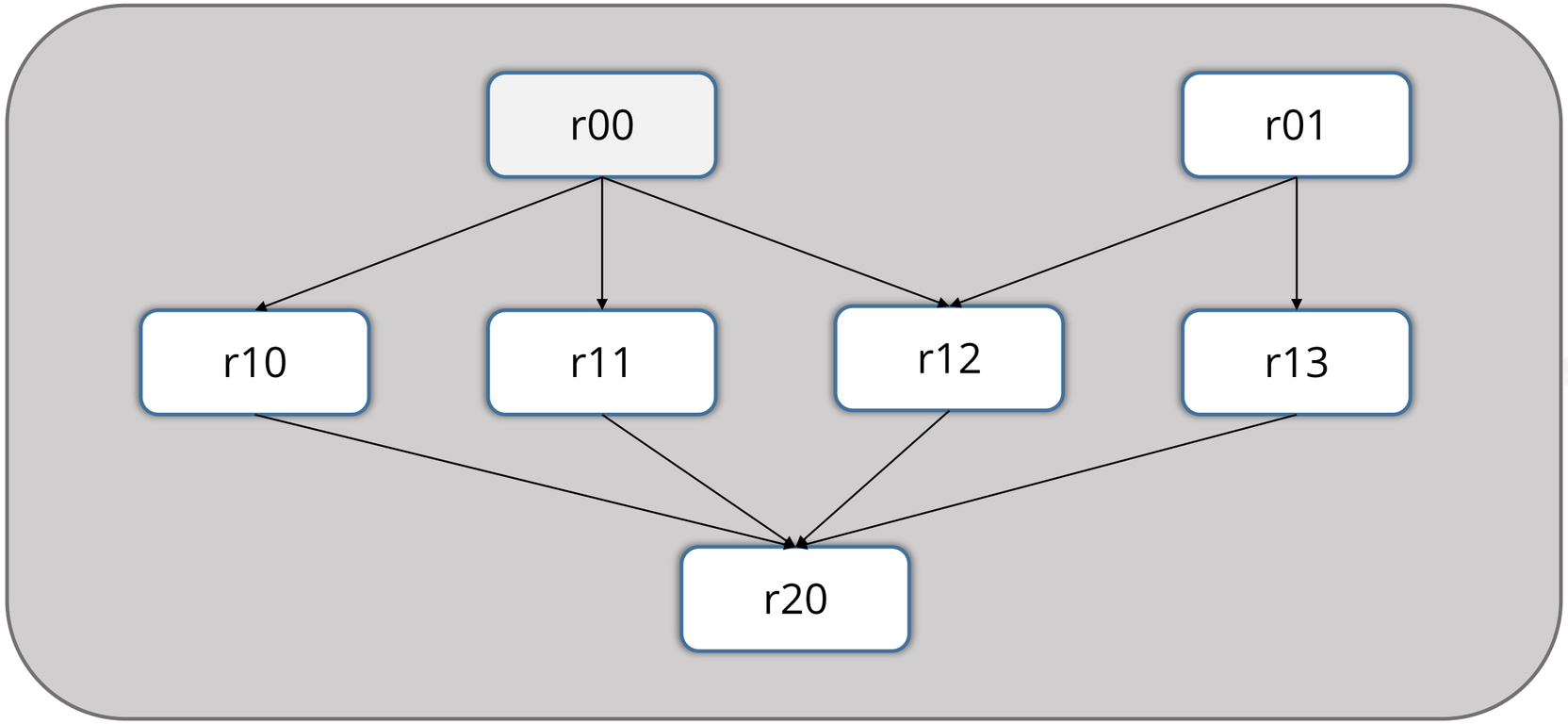
Unpersist the RDD and then mark it again for persistence with different storage level

DATA AND
ARTIFICIAL INTELLIGENCE

Lineage and DAG

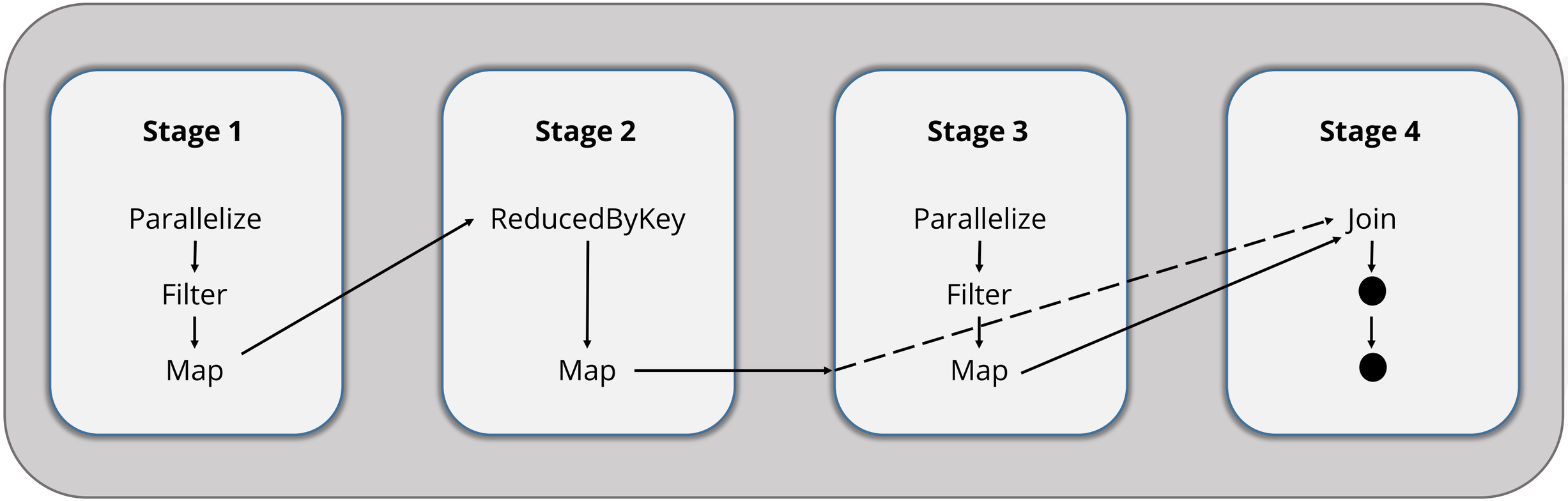
RDD Lineage

RDD Lineage is a graph that contains the existing RDD and the new RDD created from the existing one as a result of transformation.



DAG

Directed Acyclic Graph (DAG) is a graph where RDDs and the operations to be performed on RDDs are represented in the form of vertices and edges, respectively.

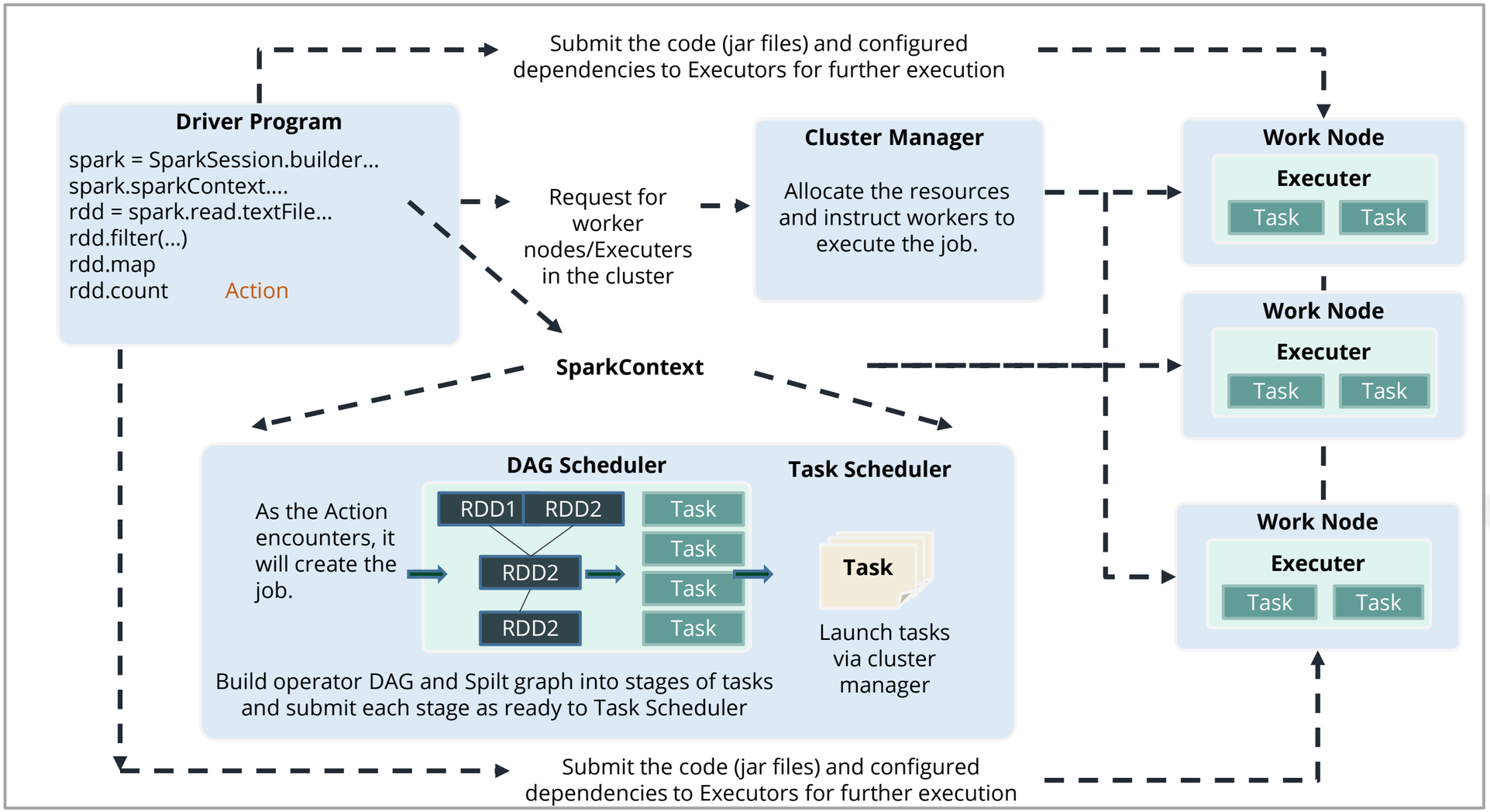


Need for DAG

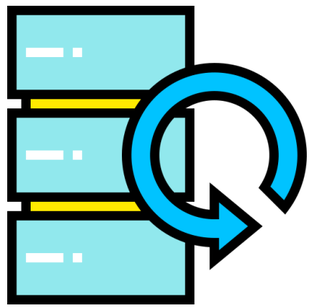
The need for DAG in Spark was created to overcome the limitations of DAG. The computation in MapReduce is done as:

1. The data is read from HDFS
2. Map and Reduce operations are applied to them
3. The result is written back to HDFS

Working of DAG



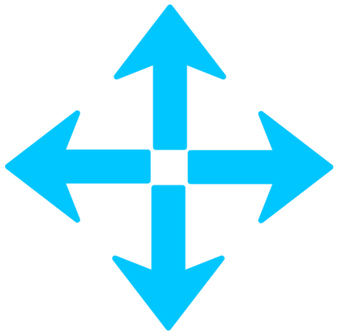
Advantages of DAG



Data Recovery

Fault Tolerance

Global Optimization



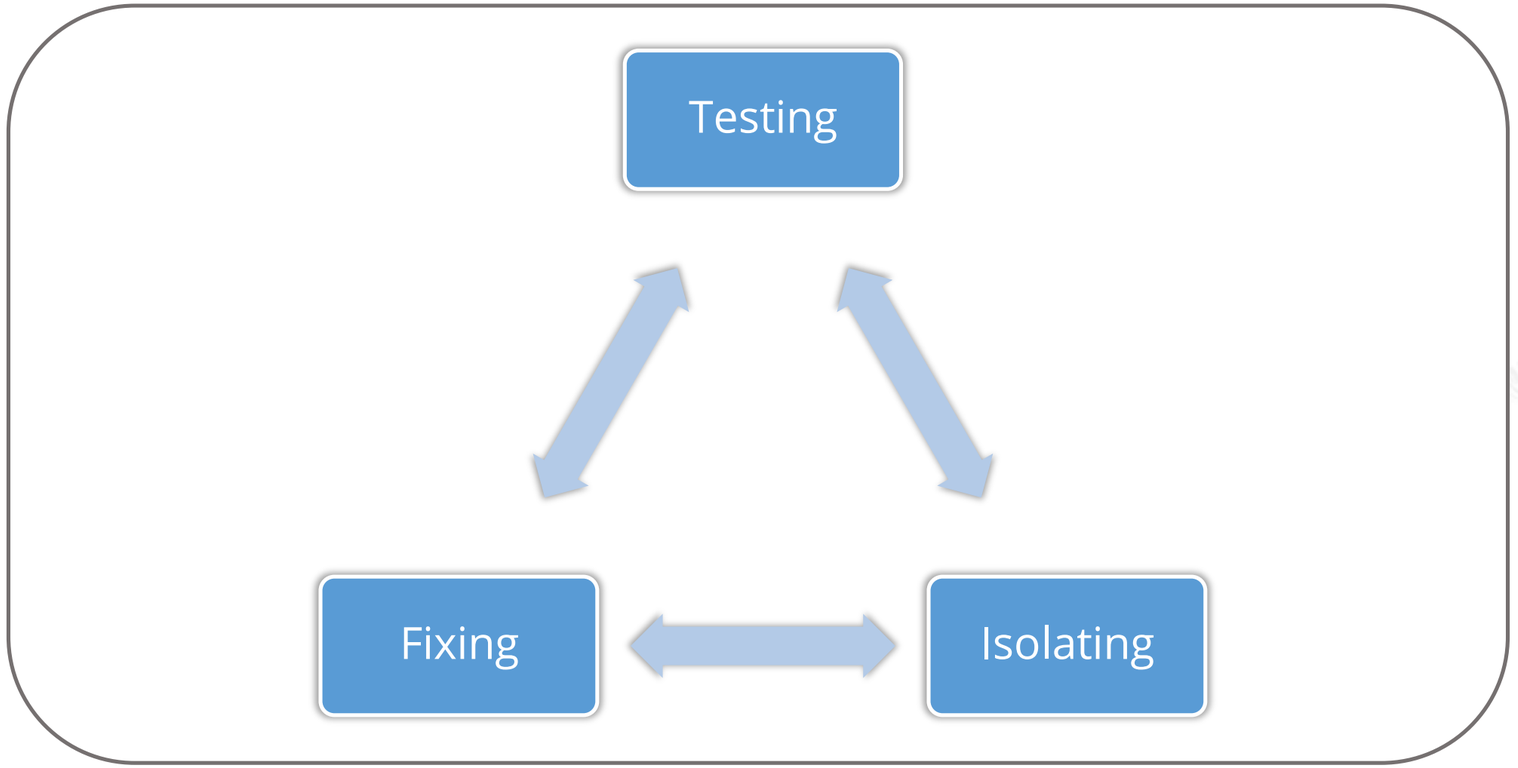
Flexibility

DATA AND
ARTIFICIAL INTELLIGENCE

Debugging in Spark

What Is Debugging?

Debugging is the process of identifying, isolating, and correcting a problem.



Attaching a Debugger to a Spark Application

The following are the steps for attaching a debugger to a Spark application:

Step 1	Upload the JAR file to the remote cluster and run it
Step 2	Define the SPARK_SUBMIT_OPTS environment variable and run it
Step 3	Connect to the debugger and configure it for use

DATA AND
ARTIFICIAL INTELLIGENCE

Partitioning in Spark

What Is Partitioning?

The data in an RDD is split into various segments called partitions.

01 Tuples in the same partition are guaranteed to be on the same machine.

02 Each machine contains one or more partitions.

03 The total number of partitions is configurable.

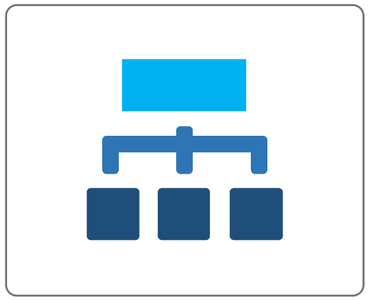
Types of Partitioning

Hash Partitioning

Range Partitioning

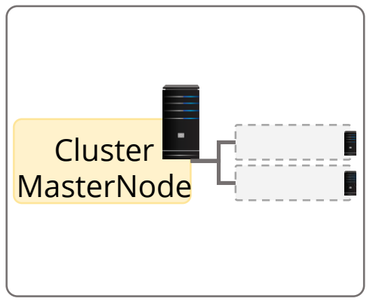
Custom Partitioning

Partitions from Single File

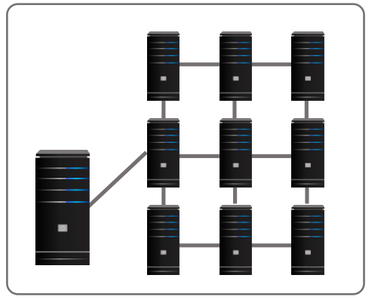


Partition can be done based on size

Partition can be done by specifying the minimum number of partitions as textFile(file, minPartitions)



While running on a cluster, the number of partitions by default is 2



More partitions = More parallelization

Partitions from Multiple Files

Partitions from multiple files can be done in the following ways:

sc.textFile("mydir/*")

sc.wholeTextFiles(“mydir”)

Operations on Partitions

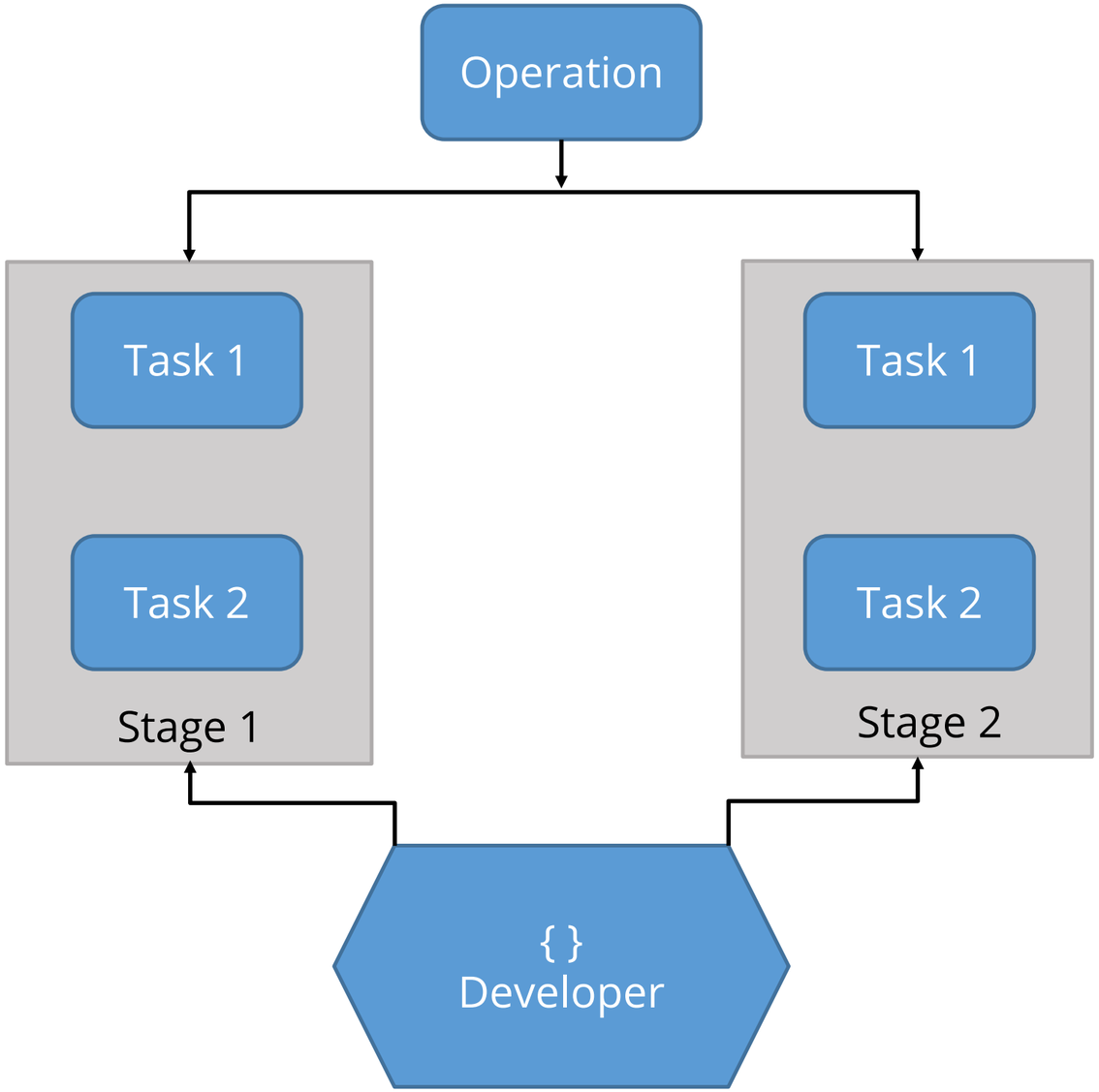
RDD operations work on each of the following partitions:

foreachPartition

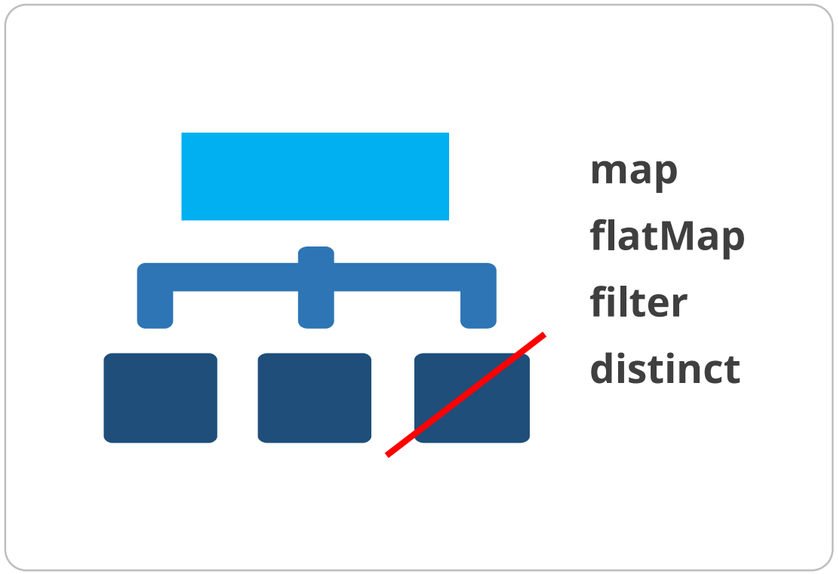
mapPartitions

mapPartitionsWithIndex

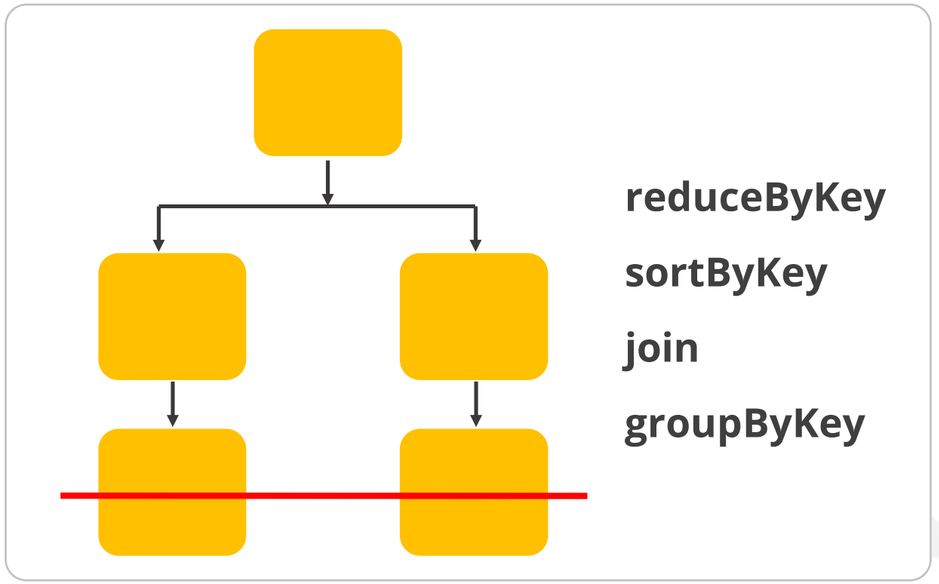
Operation Stages



Parallel Operations on Stages

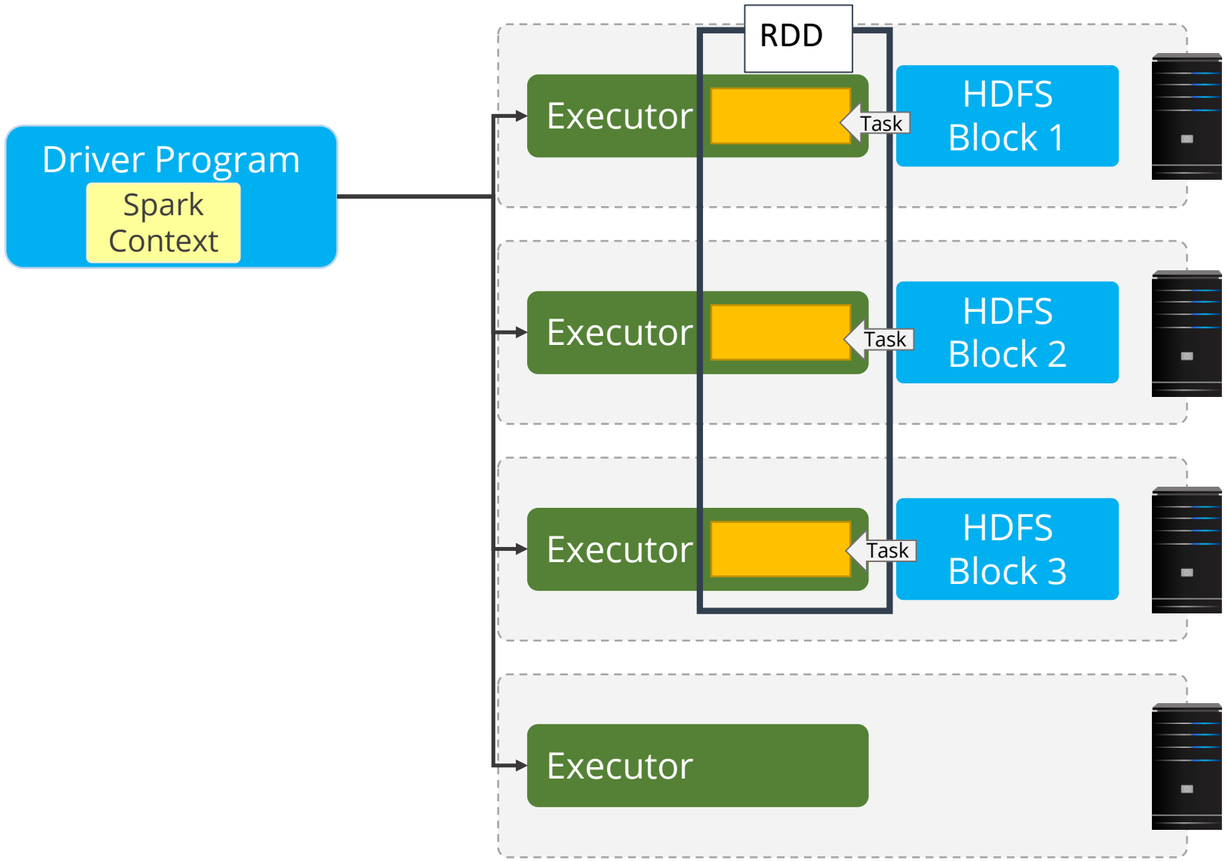


Some operations preserve partitioning



Some operations perform repartition

Parallel Operations on Stages



HDFS:
mydata

DATA AND
ARTIFICIAL INTELLIGENCE

Scheduling in Spark

Scheduling in Spark

Scheduling is the process in which resources are allocated to different tasks by an operating system.

Scheduling across Applications

Scheduling within Applications

Scheduling across Applications

In Spark, each application has its own JVM that runs tasks and stores data.

Static partitioning is the best approach for allocating resources to each application in the following Spark modes:

Standalone Mode

Mesos

YARN

Scheduling within Application

Multiple jobs can run simultaneously inside a Spark application, if they are submitted from separate threads.

Each job is divided into stages and resources are allocated in FIFO fashion.

In Spark 8.0, it is possible to enable fair scheduler which uses round-robin fashion to allocate tasks between jobs.

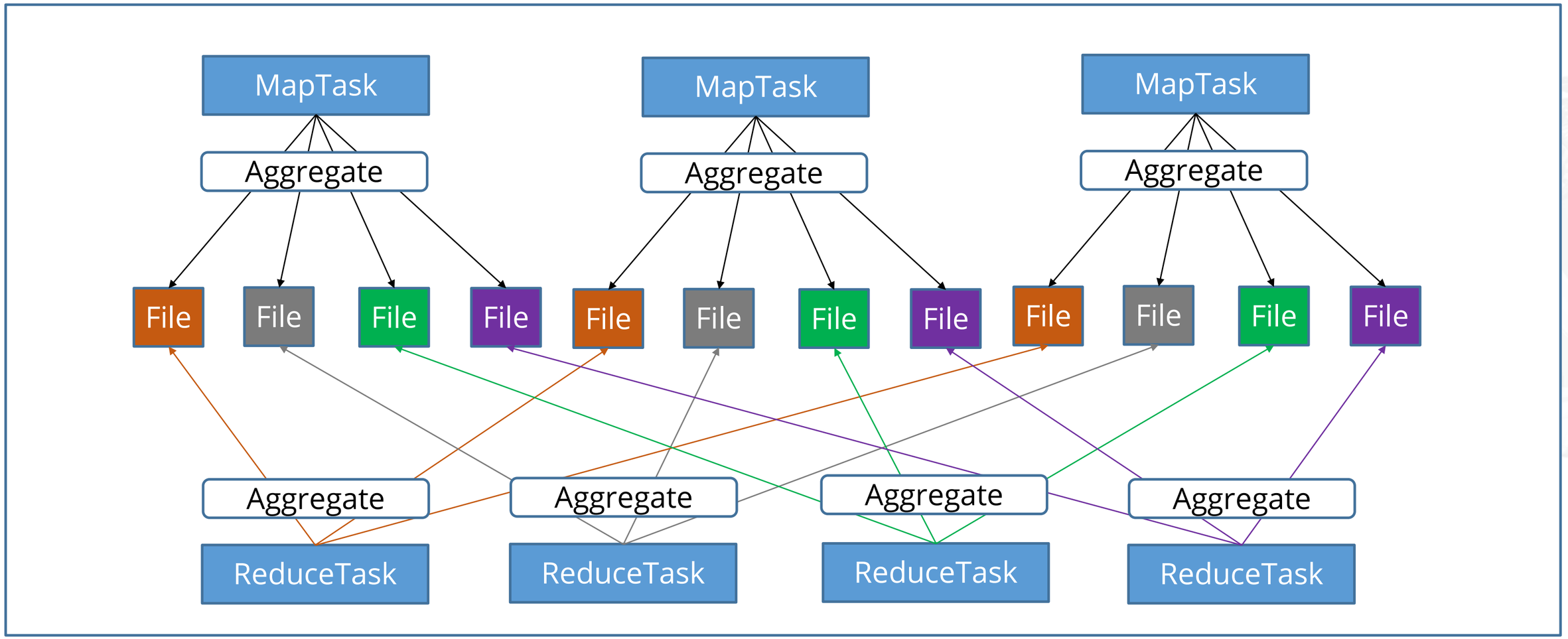
```
val conf = new SparkConf().setMaster(...).setAppName(...)

conf.set("spark.scheduler.mode", "FAIR")

val sc = new SparkContext(conf)
```

DATA AND
ARTIFICIAL INTELLIGENCE

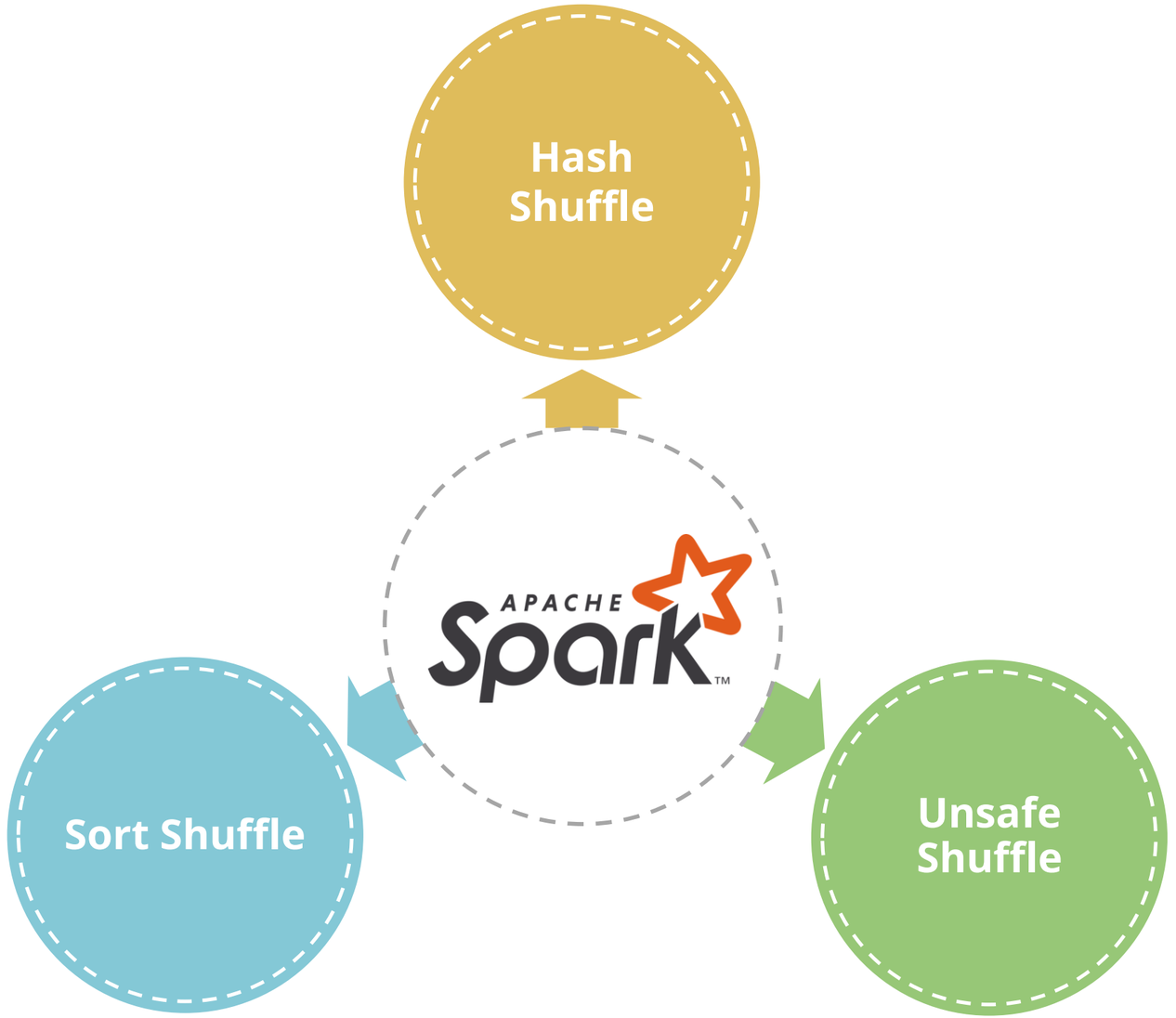
Shuffling in Spark

Shuffling in Spark

Shuffling is an operation which requires one node that will fetch data from other nodes to have data for computing the result.

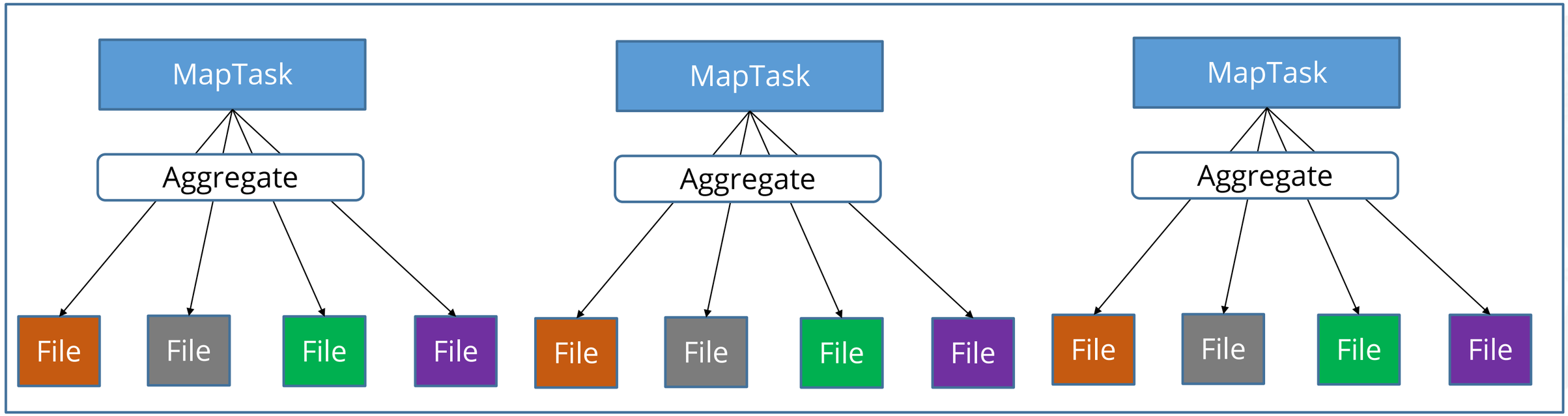


Shuffling in Spark

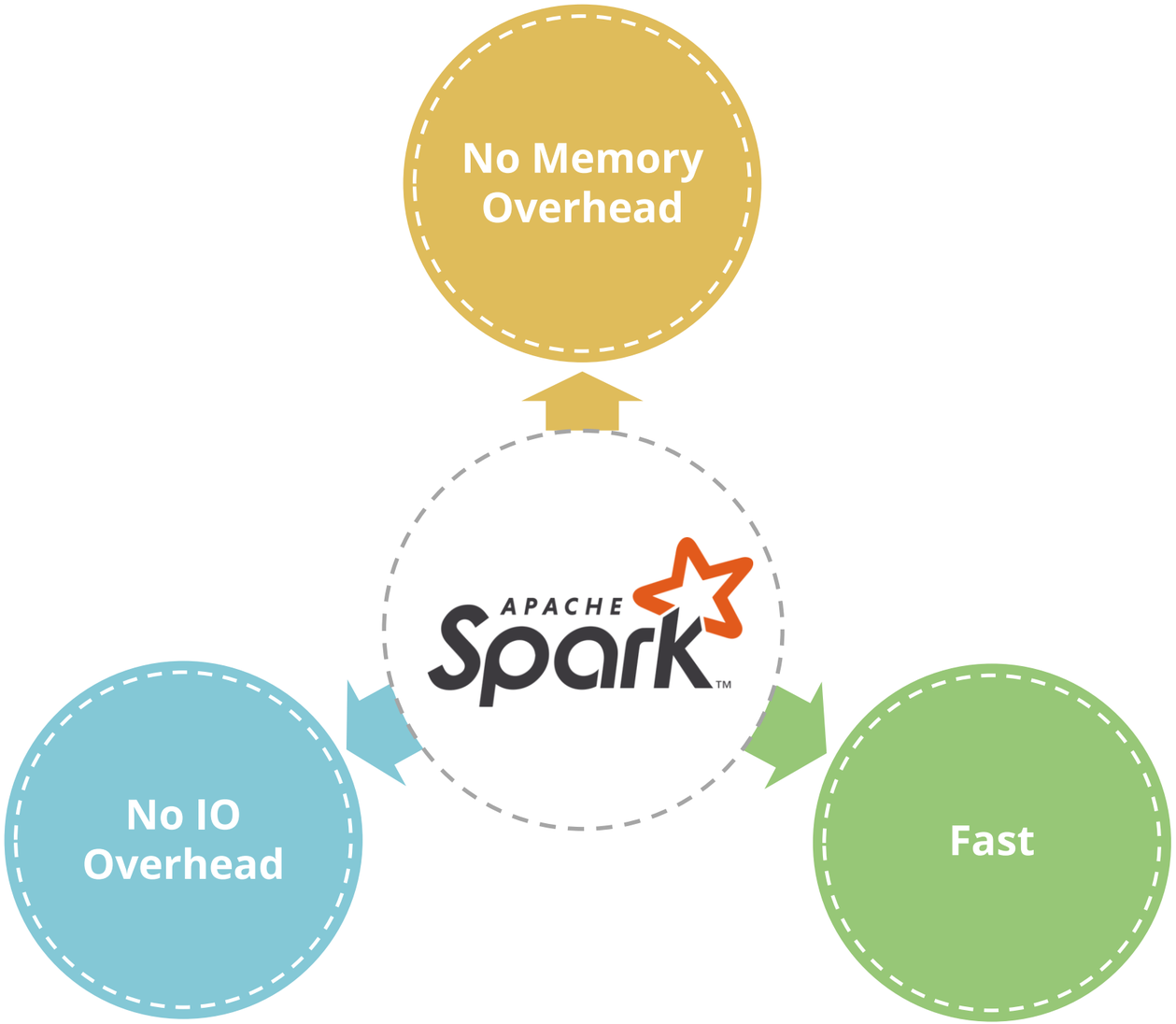


Hash Shuffle

In hash shuffle, each task will write the output into multiple files.

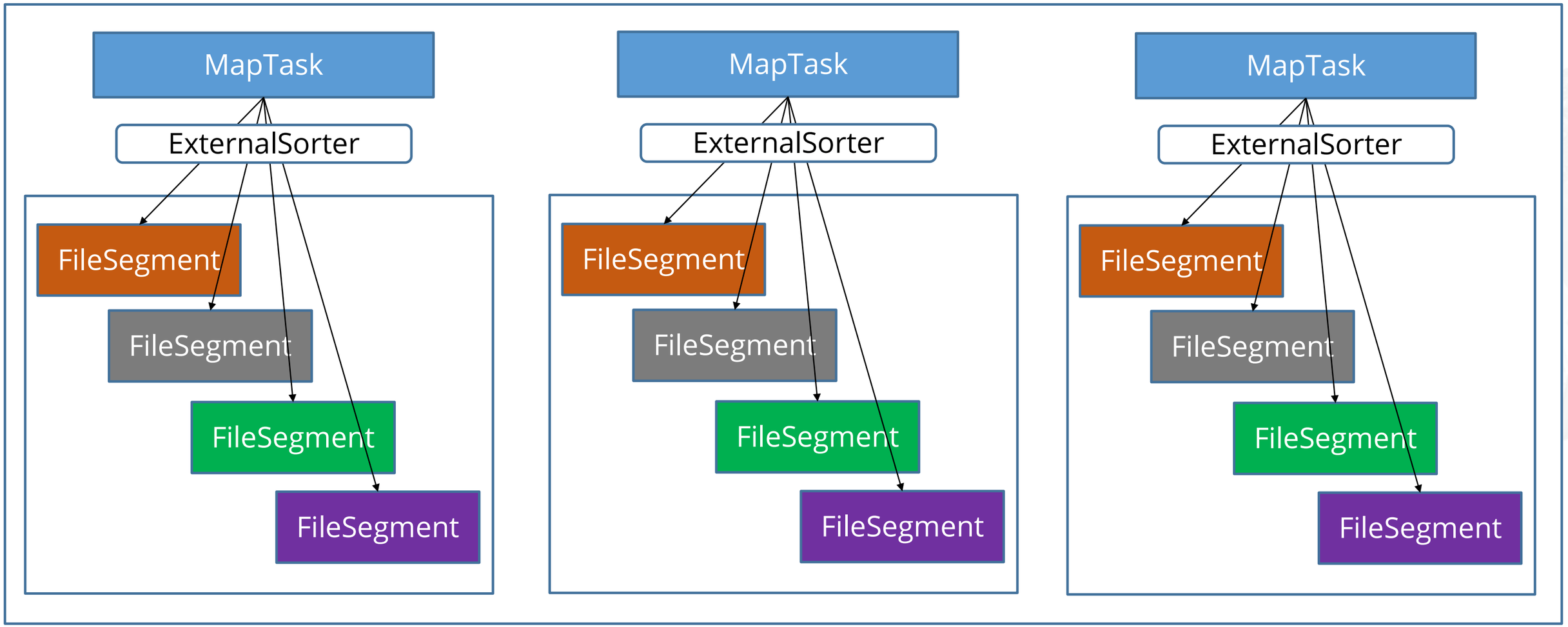


Advantages of Hash Shuffle



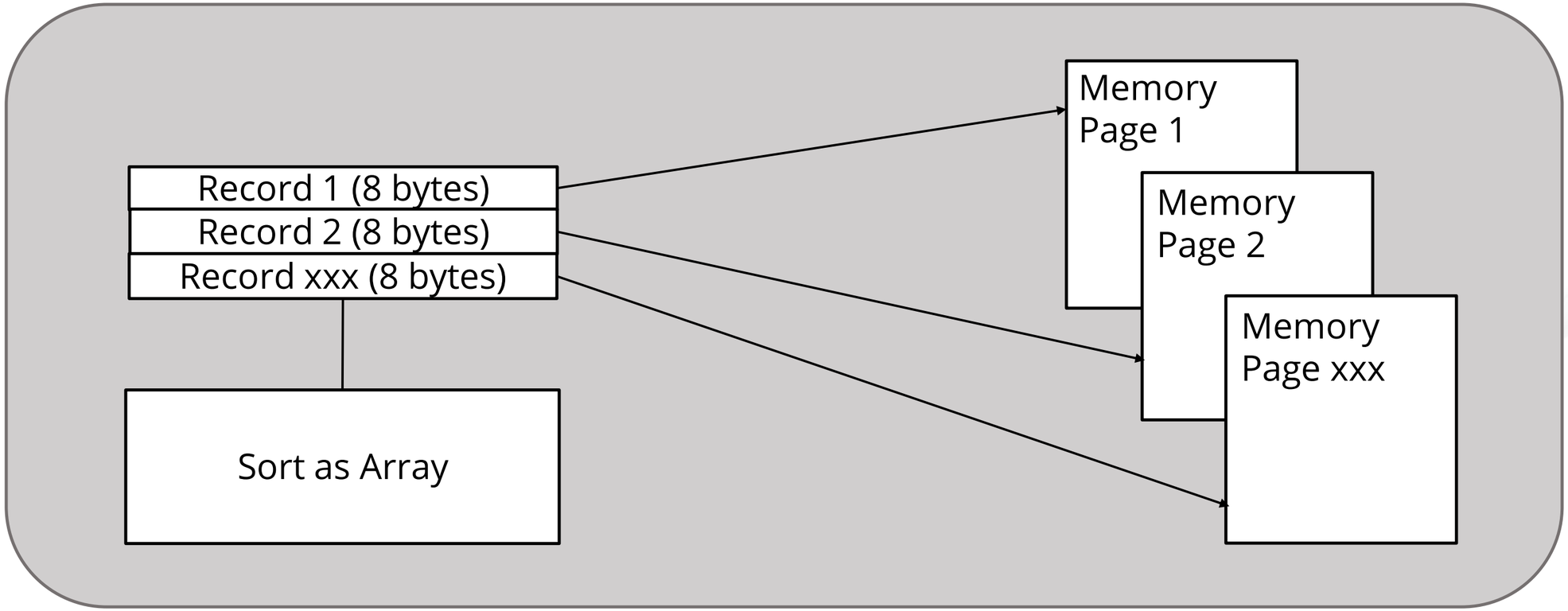
Sort Shuffle

In sort shuffle, each task spills only one shuffle containing segments and one index file.



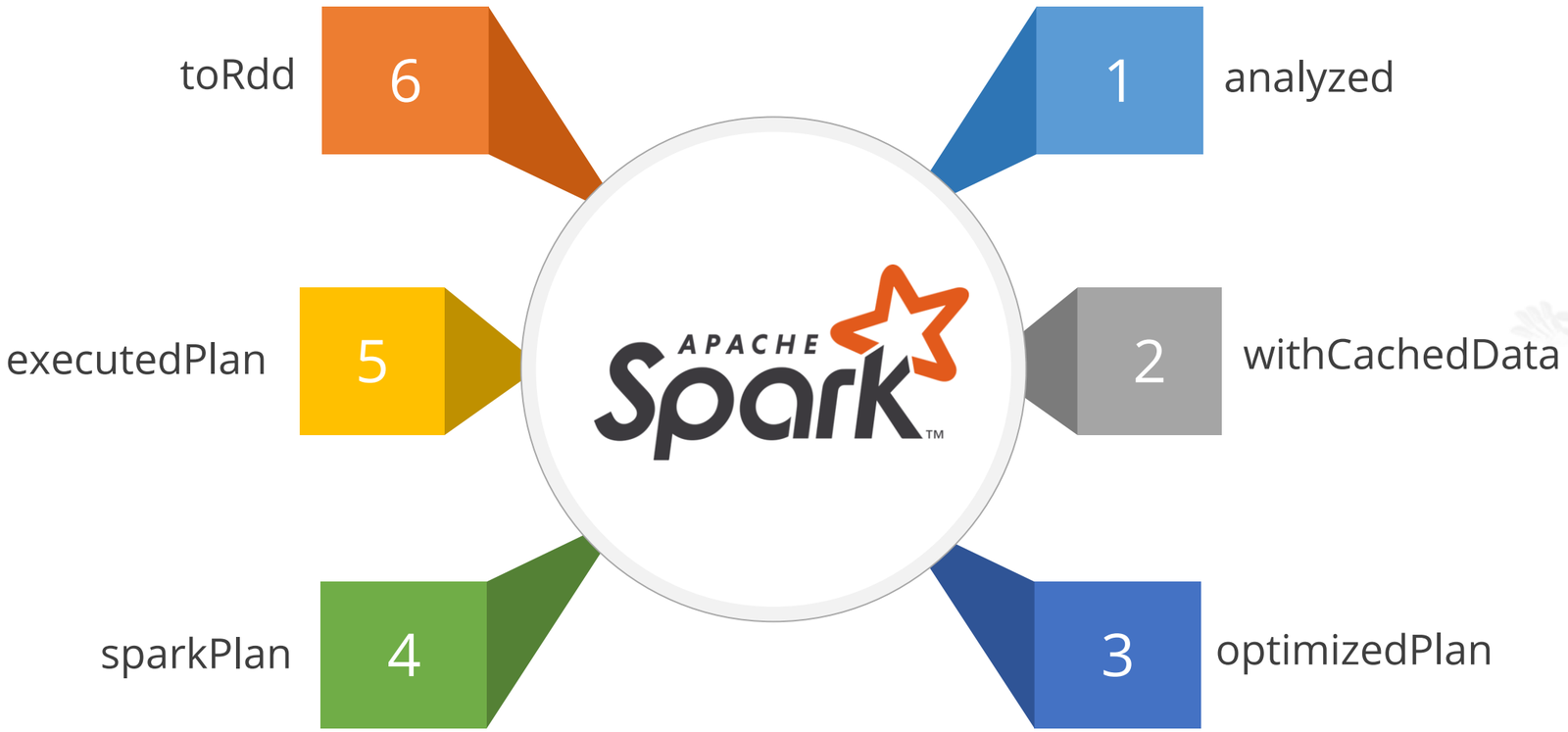
Unsafe Shuffle

In unsafe shuffle, the records are serialized once and then stored in memory pages.

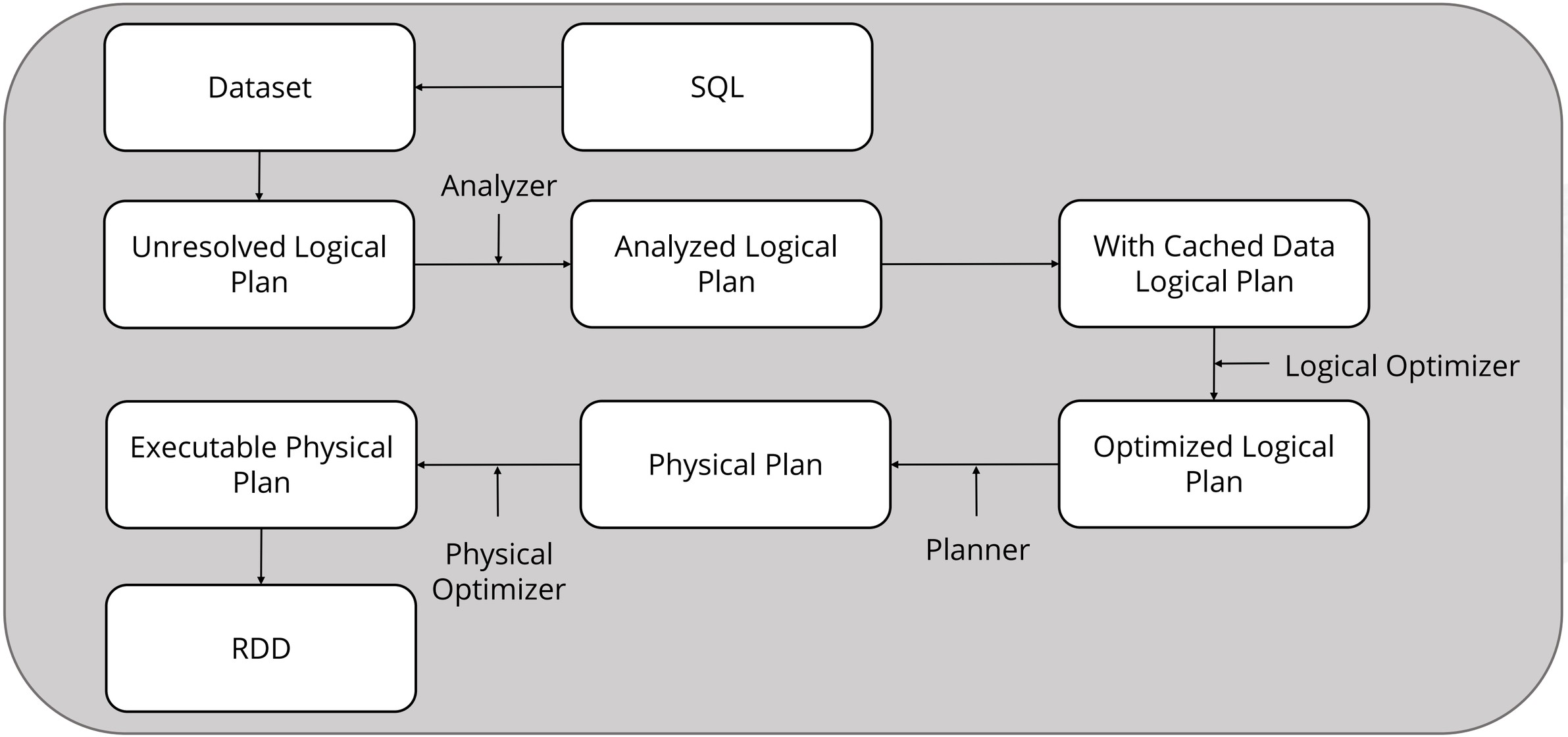


Query Execution in Spark

The following are the query execution phases in Spark:



Execution in Spark



DATA AND
ARTIFICIAL INTELLIGENCE

Aggregating Data with Pair RDD

Aggregation

To perform aggregations, the datasets must be described in the form of key-value pairs.

The following are the functions that can be used for aggregation:

Aggregation

Key	Value
Panda	0
Pink	3
Pirate	3
Panda	1
Pink	4

mapValues()

Key	Value
Panda	(0,1)
Pink	(3,1)
Pirate	(3,1)
Panda	(1,1)
Pink	(4,1)

reduceByKey()

Key	Value
Panda	(1,2)
Pink	(7,2)
Pirate	(3,1)

Spark Application with Data Written Back to HDFS and Spark UI

Duration: 10 mins

Problem Statement: In this demonstration, you will write the data to HDFS and Spark UI.

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**.

Changing Spark Application Parameters

Duration: 10 mins

Problem Statement: In this demonstration, you will change Spark application parameters.

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**.

Handling Different File Formats

Duration: 10 mins

Problem Statement: In this demonstration, you will handle different file formats.

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**.

Spark RDD with Real-World Application

Duration: 10 mins

Problem Statement: In this demonstration, you will understand Spark RDD with a real-world application.

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**.

Optimizing Spark Jobs

Duration: 10 mins

Problem Statement: In this demonstration, you will optimize Spark jobs.

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:

- Define Spark RDD and list its limitations
- Describe and demonstrate RDD operations in Spark
- Demonstrate the creation of Spark RDD
- Aggregate data with pair RDD

DATA AND
ARTIFICIAL INTELLIGENCE

Knowledge Check

Knowledge Check 1

Which feature of Spark RDD applies to all elements in datasets through maps, filter, or group by operation?

a. Lazy Evaluation

b. Cross-Grained Operation

c. Fault-Tolerant

d. Immutable

Knowledge Check 1

Which feature of Spark RDD applies to all elements in datasets through maps, filter, or group by operation?

a. Lazy Evaluation

b. Cross-Grained Operation

c. Fault-Tolerant

d. Immutable

The correct answer is **b**

Cross-Grained Operation feature of Spark RDD applies to all elements in datasets through maps, filter, or group by operation.

Knowledge Check 2

Which of the following transformations is not self-sufficient?

a. Narrow Transformation

b. Wide Transformation

c. Both a and b

d. None of the above

Knowledge Check 2

Which of the following transformations is not self-sufficient?

a. Narrow Transformation

b. Wide Transformation

c. Both a and b

d. None of the above

The correct answer is **b**

Wide transformation is not self-sufficient.

Knowledge Check 3

Which of the following query execution phases makes sure that the logical plan is analyzed and uses the supported operations only?

a. toRdd

b. executedPlan

c. withCachedData

d. optimizedPlan

The correct answer is **c**

withCachedData makes sure that the logical plan is analyzed and uses the supported operations only.

Lesson-End Project

Problem Statement: The New York School authority collects data from all schools that provide bus facilities to students. This data helps to understand if buses are reaching on time or not. This also helps to understand if there is a specific route where buses are taking more time so that it can be improved.
You are given a dataset of buses which got broke down or are running late. You have been given the below data set:
Filename: bus-breakdown-and-delays.csv

1. School_Year
    a. Indicates the year the record refers to
2. Run_Type
    a. Designates whether a breakdown or delay occurred on a specific category of bus services
3. Bus_No
4. Route_Number
5. Reason
    a. Reason for delay as entered by the staff employed by the reporting bus vendor
6. Occurred_On
7. Number_Of_Students_On_The_Bus

Lesson-End Project

You are hired as a big data consultant to provide important insights. You must write a Spark job using the above data and need to provide the following:

1. Most common reasons for either a delay or breaking down of bus
2. Top 5 route numbers where the bus was either delayed or broke down
3. The total number of incidents,year-wise, when the students were
a. In the bus
b. Not in the bus
4. The year in which accidents were less

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Spark SQL - Processing DataFrames

simplilearn

Learning Objectives

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

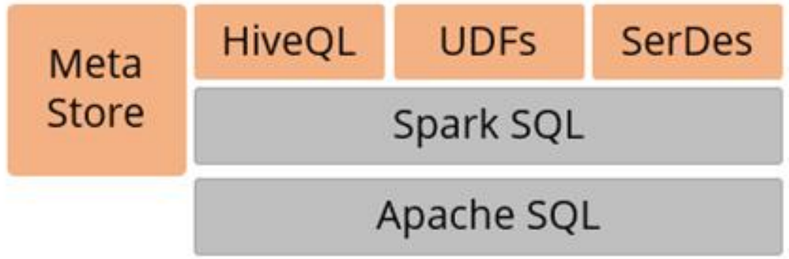
- Explain the importance and features of Spark SQL
- Describe the methods to convert RDDs to DataFrames
- Load existing data into a DataFrame

DATA AND
ARTIFICIAL INTELLIGENCE

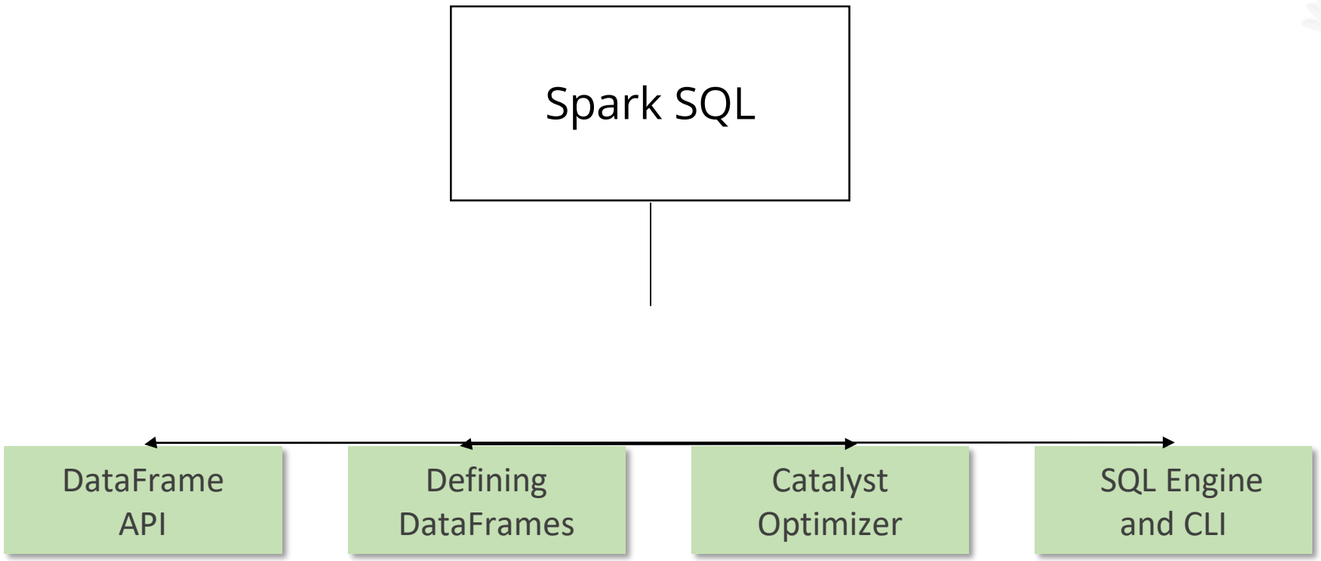
Spark SQL Introduction

What Is Spark SQL?

Spark SQL is a module for structured data processing that is built on top of core Spark.



Spark SQL provides the following:



Importance of Spark SQL

Spark SQL provides four main capabilities for using structured and semi-structured data.

Acts as a distributed SQL query engine

Provides DataFrames for programming abstraction

Allows users to query structured data in Spark programs

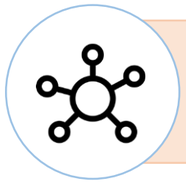
Can be used with platforms such as Scala, Java, R, and Python

Advantages of Spark SQL

Hive Compatibility: Compatible with the existing Hive queries, UDFs, and data

Integrated: Mixes SQL queries with Spark programs

Unified Data Access: Loads and queries data from different sources



Standard Connectivity: Connects through JDBC or ODBC

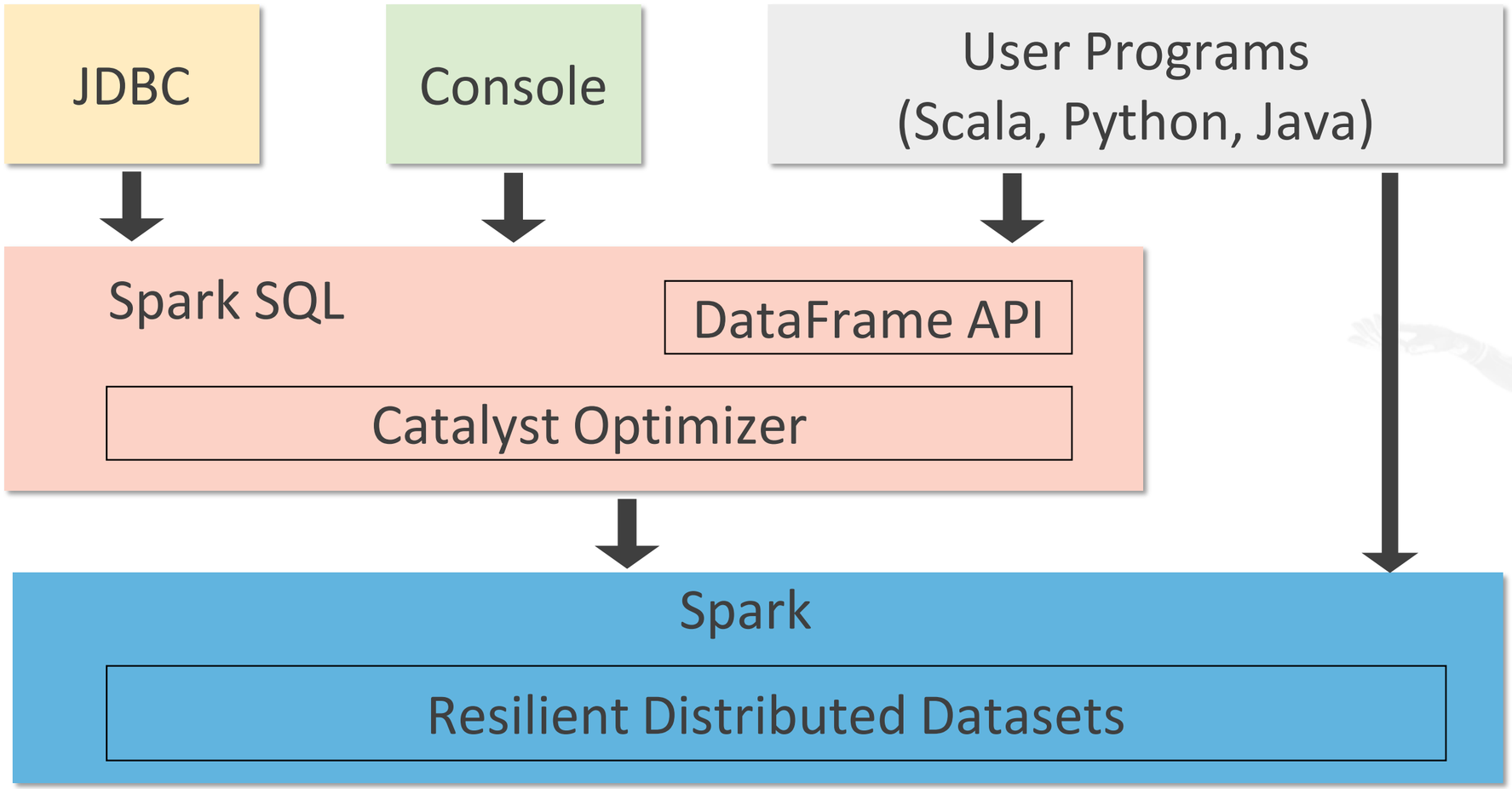
Performance: Executes the most optimal plans

DATA AND
ARTIFICIAL INTELLIGENCE

Spark SQL Architecture

Spark SQL Architecture

The below diagram shows the typical architecture and interfaces of Spark SQL.



SQLContext

The SQLContext class or any of its descendants acts as the entry point into all functionalities.

Q

How to get the benefit of a superset of the basic SQLContext functionality?

Build a HiveContext to:

- Use the writing ability for queries
- Access Hive UDFs and read data from Hive tables

```
val sqlContext = new
org.apache.Spark.sql.SQLContext(sc)
```

SQLContext

Points to Remember:

- You can use the Spark.sql.dialect option to select the specific variant of SQL used for parsing queries
- On a SQLContext, the sql function allows applications to programmatically run SQL queries and then return a DataFrame as the result.

```
val df = sqlContext.sql("SELECT * FROM table")
```

DATA AND
ARTIFICIAL INTELLIGENCE

DataFrames

DataFrames

DataFrames represent a distributed collection of data in which data is organized into columns that are named.

Construct a DataFrame

Use sources such as tables in Hive, structured data files, existing RDDs, and external databases.

Convert Them to RDDs

Call the rdd method, that returns the DataFrame content, as an RDD of rows.

In the prior versions of Spark SQL API, SchemaRDD has been renamed as DataFrame.

Creating DataFrames

DataFrames can be created:

- From an existing structured data source
- From an existing RDD
- By performing an operation or query on another DataFrame
- By programmatically defining a schema

Creating a DataFrame

```
val sc: SparkContext          // An existing SparkContext.
val sqlContext = new org.apache.Spark.sql.SQLContext(sc)

val df = sqlContext.read.json("examples/src/main/resources/customers.json")
// Displays the content of the DataFrame to stdout

df.show()
```

Assisted Practice

Handling Various Data Formats

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to handle various data formats.

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**.

Using DataFrame Operations

DataFrames provide a domain-specific language that can be used for structured data manipulation in Java, Scala, and Python.

```
val sc: SparkContext
    val sqlContext = new org.apache.Spark.sql.SQLContext(sc)

    // Create the DataFrame
    val df = sqlContext.read.json("examples/src/main/resources/customers.json")

// Show the content of the DataFrame
df.show()

    // Print the schema in a tree format
    df.printSchema()

    // Select only the "name" column
    df.select("name").show()
```

Using DataFrame Operations

```
// Select everybody, but increment the age by 1
df.select(df("name"), df("age") + 1).show()

// Select people older than 21
df.filter(df("age") > 21).show()

// Count people by age
df.groupBy("age").count().show()
```

Assisted Practice

Implement Various DataFrame Operations

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to implement various DataFrame operations like filter, aggregates, joins, count, and sort.

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**.

UDFs: User Defined Functions

UDFs are used to define new column-based functions that help Spark SQL in transforming datasets.

The example below defines a UDF to convert a given text to upper case.

```
-> val dataset = Seq((0, "hello"),(1, "world")).toDF("id","text")

-> val upper: String =&amp;amp;amp;amp;gt; String =_.toUpperCase

-> import org.apache.Spark.sql.functions.udf

-> val upperUDF = udf(upper)

-> dataset.withColumn("upper", upperUDF('text)).show
```

UDAF: User-Defined Aggregate Functions

Spark SQL allows users to define custom aggregation functions called User-Defined Aggregate Functions or UDAFs.

Type	Value	Type Total
A	3	15
A	12	15
B	7	9
B	2	9
C	9	20
C	11	20

UDAFs are very useful when performing aggregations across groups or columns.

UDAF: User-Defined Aggregate Functions

In order to write a custom UDAF, you need to extend UserDefinedAggregateFunction and define four methods.

```
class GeometricMean extends UserDefinedAggregateFunction
```

Initialize	On a given node, this method is called once for each group.
Update	For a given group, Spark will call “update” for each input record of that group.
Merge	If the function supports partial aggregates, Spark computes partial result and combines them together.
Evaluate	Once all the entries for a group are exhausted, Spark will call “evaluate” to get the final result.

Assisted Practice

UDF and UDAF

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to create UDF and UDAF.

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**.

Interoperating with RDDs

To convert existing RDDs into DataFrames, Spark SQL supports two methods:

Reflection-Based

- Infers an RDD schema containing specific types of objects
- Works well when the schema is already known while writing the Spark application

Programmatic

- Lets you build a schema and apply to an already existing RDD
- Allows you to build DataFrames when you do not know the columns and their types until runtime

Using the Reflection-Based Approach

For Spark SQL, the Scala interface allows users to convert an RDD with case classes to a DataFrame automatically.

1. The case class has the table schema, where the argument names to the case class are read using the reflection method.
2. The case class can be nested and used to contain complex types like sequence of arrays
3. Scala Interface implicitly convert the resultant RDD to a DataFrame and register it as a table.

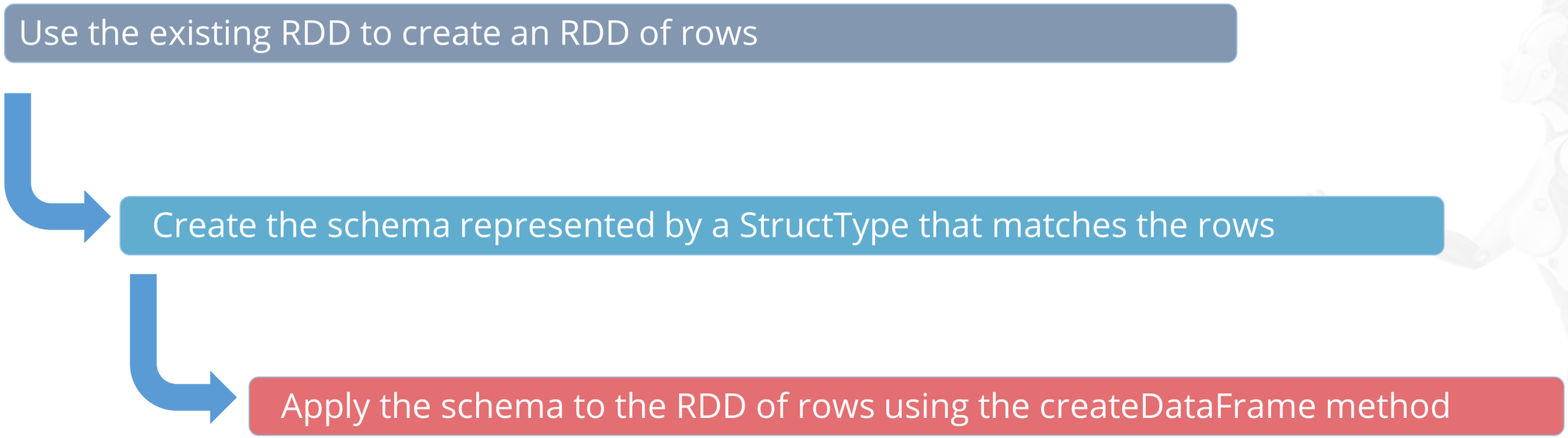
Using the Reflection-Based Approach

```
val sqlContext = new org.apache.Spark.sql.SQLContext(sc)
import sqlContext.implicits._
case class Person(name: String, age: Int)
// Create an RDD of Person objects and register it as a table.
val people = sc.textFile("examples/src/main/resources/people.txt").map(_.split(",")).map(p => Person(p(0),
p(1).trim.toInt)).toDF()
people.registerTempTable("people")
// SQL statements can be run by using the sql methods provided by sqlContext.
val teenagers = sqlContext.sql("SELECT name, age FROM people WHERE age >= 13 AND age <= 19")
teenagers.map(t => "Name: " + t(0)).collect().foreach(println)
// or by field name:
teenagers.map(t => "Name: " + t.getAs[String]("name")).collect().foreach(println)
// row.getValuesMap[T] retrieves multiple columns at once into a Map[String, T]
teenagers.map(_.getValuesMap[Any](List("name", "age"))).collect().foreach(println)
```

Using the Programmatic Approach

This method is used when you cannot define case classes ahead of time.
For example, when the records structure is encoded in a text dataset or a string.

The below steps are used for defining case classes:



Using the Programmatic Approach

```
val sqlContext = new org.apache.Spark.sql.SQLContext(sc)
val people = sc.textFile("examples/src/main/resources/people.txt")

// The schema is encoded in a string
val schemaString = "name age"
import org.apache.Spark.sql.Row;
import org.apache.Spark.sql.types.{StructType,StructField,StringType};

// Generate the schema based on the string of schema, Convert records of the RDD (people)
   to Rows and Apply the schema to the RDD.
val schema =   StructType( schemaString.split(" ").map(fieldName => StructField(fieldName, StringType, true)))
val rowRDD = people.map(_.split(",")).map(p => Row(p(0), p(1).trim))
val peopleDataFrame = sqlContext.createDataFrame(rowRDD, schema)
peopleDataFrame.registerTempTable("people")
```

Assisted Practice

Process DataFrame Using SQL Query

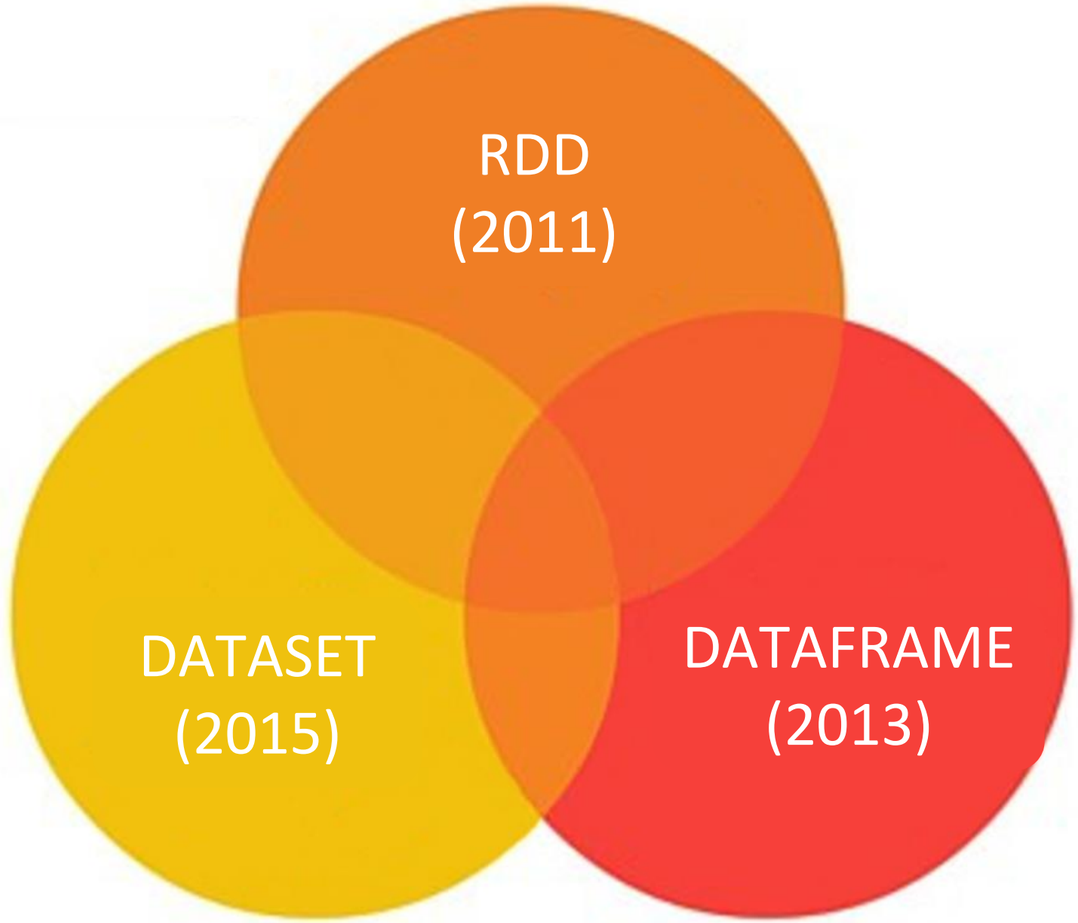
Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to process DataFrame(s) using SQL queries.

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**.

RDD vs. DataFrame vs. Dataset

Apache Spark provides three types of APIs: RDD, DataFrame, and Dataset.



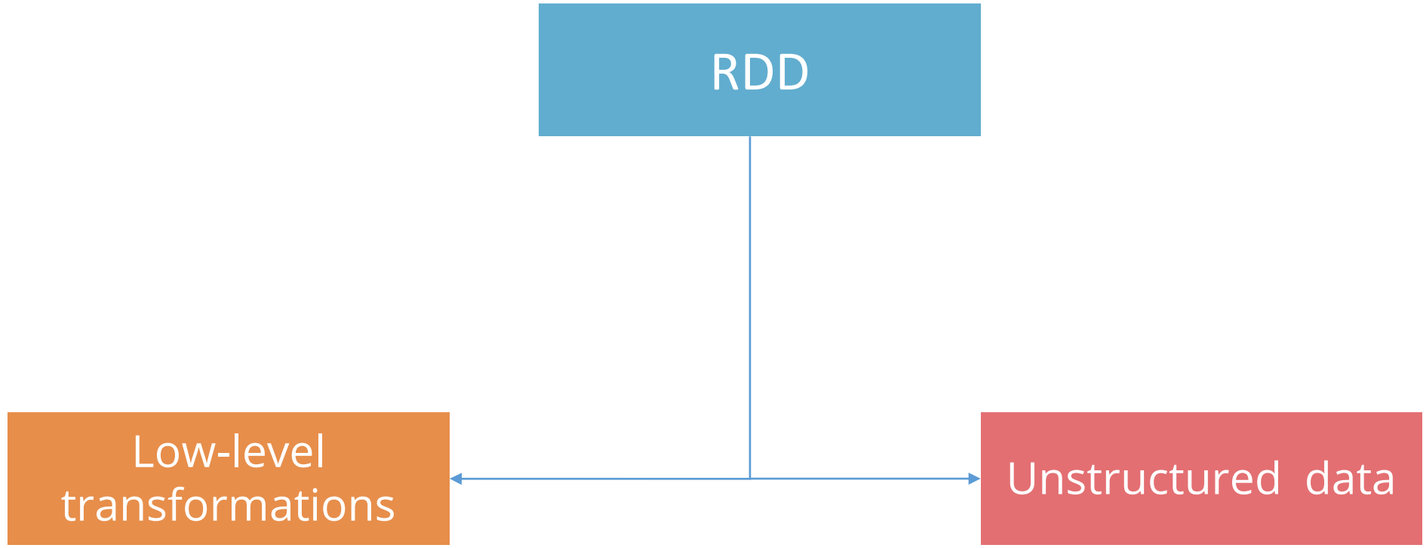
Example: Filter By Attribute

Given below are the various ways to filter an attribute using the three APIs.

RDD	`rdd.filter(_.marks > 75)`
DataFrame	`df.filter("marks > 75");`
Dataset	`dataset.filter(_.marks > 75);`

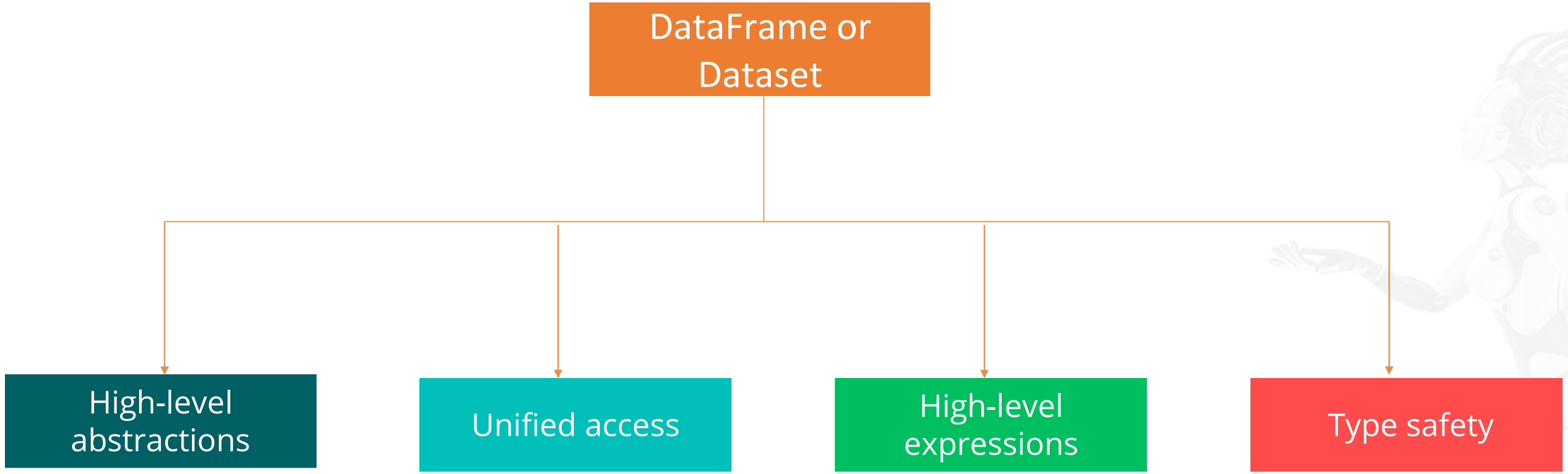
Use Case: RDD API

We can use any of the three APIs based on the requirements.



Use Case: DataFrame or Dataset API

In Spark 2.0, DataFrame APIs and Datasets APIs were merged, unifying data processing capabilities across libraries.



Unassisted Practice

Processing DataFrames

Duration: 15 mins

Problem Statement: Using Scala, perform the below tasks on Spark DataFrames.

- Create the case classes with the following fields: Department, Employee, and DepartmentWithEmployees.
  Note: Create the DepartmentWithEmployees instances from Departments and Employees.
  Insert at least four values.

- Create two DataFrames from the list of the above case classes.

- Combine the two DataFrames and write the combined DataFrame to a parquet file.

- Use filter() or where() clause to return the rows whose first name is either “Alice” or “John”.
  Note: Use first name filter as per your entries.

- Retrieve rows with missing firstName or lastName.

- Find the distinct (firstName, lastName) combinations.

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**.

Unassisted Practice

Steps to Perform

// Create the case classes

```
case class Department(id: String, name: String)
case class Employee(firstName: String, lastName: String, email: String, salary: Int)
case class DepartmentWithEmployees(department: Department, employees: Seq[Employee])
```

// Create the Departments

```
val department1 = new Department("123456", "Computer Science")
val department2 = new Department("789012", "Mechanical Engineering")
val department3 = new Department("345678", "Theater and Drama")
val department4 = new Department("901234", "Indoor Recreation")
```

// Create the Employees

```
val employee1 = new Employee("Alice", "Mathew", "no-reply@harvard.edu", 100000)
val employee2 = new Employee("John", "Singleton", "no-reply@stanford.edu", 120000)
val employee3 = new Employee("matei", null, "no-reply@oxford.edu", 140000)
val employee4 = new Employee(null, "wendell", "no-reply@princeton.edu", 160000)
```

Unassisted Practice

Steps to Perform

```
// Create the DepartmentWithEmployees instances from Departments and Employees

val departmentWithEmployees1 = new DepartmentWithEmployees(department1, Seq(employee1, employee2))
val departmentWithEmployees2 = new DepartmentWithEmployees(department2, Seq(employee3, employee4))
val departmentWithEmployees3 = new DepartmentWithEmployees(department3, Seq(employee1, employee4))
val departmentWithEmployees4 = new DepartmentWithEmployees(department4, Seq(employee2, employee3))

// Create DataFrames from a list of the case classes

val departmentsWithEmployeesSeq1 = Seq(departmentWithEmployees1, departmentWithEmployees2)
val df1 = departmentsWithEmployeesSeq1.toDF()
display(df1)

val departmentsWithEmployeesSeq2 = Seq(departmentWithEmployees3, departmentWithEmployees4)
val df2 = departmentsWithEmployeesSeq2.toDF()
display(df2)
```

Steps to Perform

```
// combining the two DataFrames

val unionDF = df1.unionAll(df2)
display(unionDF)

// Write the combined DataFrame to a Parquet file

unionDF.write.parquet("/user/simpli_learn/simplitest")

// Explode the employees column

val flattenDF = parquetDF.select(functions.explode($"employees")).flattenSchema
val columnsRenamed = Seq("firstName", "lastName", "email", "salary")
val explodeDF = flattenDF.toDF(columnsRenamed: _*)
explodeDF.show()
```

Unassisted Practice

Steps to Perform

```
// Using filter()  to return the rows where first name is either 'Alice' or 'John'

val filterDF = explodeDF
  .filter($"firstName" === "Alice" || $"firstName" === "John")
  .sort($"lastName".asc)
display(filterDF)

// Retrieve rows with missing firstName or lastName

val filterNonNullDF = nonNullDF.filter($"firstName" === "" || $"lastName" === "").sort($"email".asc)
display(filterNonNullDF)
```

Unassisted Practice

Steps to Perform

```
// aggregations using agg() and countDistinct()

import org.apache.spark.sql.functions._

val countDistinctDF = nonNullDF.select($"firstName", $"lastName")
  .groupBy($"firstName", $"lastName")
  .agg(countDistinct($"firstName") as "distinct_first_names")
display(countDistinctDF)
```

Key Takeaways

You are now able to:

- Explain the importance and features of Spark SQL
- Describe the methods to convert RDDs to DataFrames
- Load existing data into a DataFrame

DATA AND
ARTIFICIAL INTELLIGENCE

Knowledge Check

Knowledge Check 1

Which of the following is built on Spark for general purpose processing?

a. Hive

b. Spark SQL

c. MapReduce

d. None of the above

Knowledge Check 1

Which of the following is built on Spark for general purpose processing?

a. Hive

b. Spark SQL

c. MapReduce

d. None of the above

The correct answer is **b.**

Spark SQL is built on Spark for general purpose processing.

Knowledge Check 2

What are the functions of Spark SQL?

a. It provides the DataFrame API.

b. It defines DataFrames containing rows and columns.

c. It provides the Catalyst Optimizer along with SQL engine and CLI.

d. All of the above

Knowledge Check 2

What are the functions of Spark SQL?

a. It provides the DataFrame API.

b. It defines DataFrames containing rows and columns

c. It provides the Catalyst Optimizer along with SQL engine and CLI

d. All of the above

The correct answer is **d.**

Spark SQL provides the DataFrame API, defines DataFrames containing rows and columns, and provides the Catalyst Optimizer along with SQL engine and CLI.

Knowledge Check 3

Which of the following represents a distributed collection of data in which data is organized into columns that are named?

a. Spark SQL

b. SparkContext

c. DataFrames

d. Data Organizer

Knowledge Check 3

Which of the following represents a distributed collection of data in which data is organized into columns that are named?

a. Spark SQL

b. SparkContext

c. DataFrames

d. Data Organizer

The correct answer is **b.**

SQLContext represents a distributed collection of data in which data is organized into columns that are named.

Knowledge Check 4

Which of the following methods is utilized to convert RDDs to DataFrames?

a. Programmatic

b. Reflective-Based

c. Both a and b

d. None of the above

Knowledge Check 4

Which of the following methods is utilized to convert RDDs to DataFrames?

a. Programmatic

b. Reflective-Based

c. Both a and b

d. None of the above

The correct answer is **c.**

To convert existing RDDs into DataFrames, Spark SQL supports two methods: Programmatic and Reflective-Based.

Lesson-End Project

Problem Statement:

Every country has data for each of the companies that are operating in that country. Registering a company is mandatory as it has to provide information about its profit/loss and other details. “People data Labs” is one of the biggest data companies which collects and provides data. They recently open-sourced the datasets of “Global companies”, which operate in various countries. Data can be used to find a company in any specific industry, their employee count, and website details.

all_companies_details.csv contains the below fields:

1. Name
2. Domain
3. Year founded in
4. Industry
5. Size range
6. Country
7. LinkedIn URL
8. Current employee estimate
9. Total employee estimate

Lesson-End Project

Your task is to find:

1. The companies which were registered before 1980 in each country
2. The companies which were founded between 1990 and 2000 and has a size range of 10001+
3. The top industries having the maximum number of companies
4. The top 5 countries having the maximum number of companies that belong to the “facilities services” industry

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Spark MLlib: Modeling Big Data with Spark

Learning Objectives

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

- Identify the skills required to become a data scientist and data analyst
- Define analytics in Spark and list the types of analytics
- Describe the machine learning algorithms
- Define MLlib and MLlib pipeline

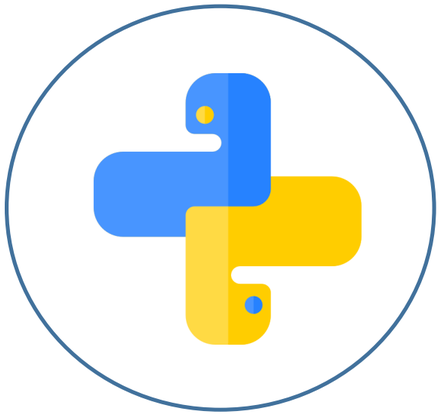
DATA AND
ARTIFICIAL INTELLIGENCE

Role of Data Scientist and Data Analyst in Big Data

Who Is a Data Scientist?

A data scientist is the person who gathers data from multiple sources and applies machine learning, predictive analytics, and sentiment analysis to extract critical information from the collected data sets.

Skills Required to Become a Data Scientist



Knowledge of Python and R

Knowledge of Machine Learning

Experience in SQL

Understanding of Multiple Analytics Function

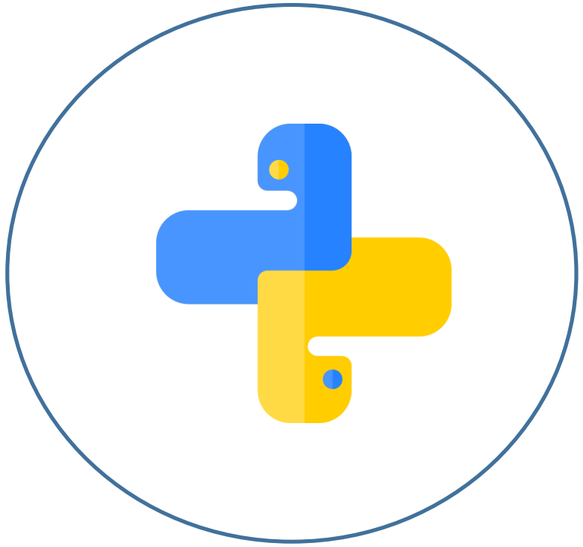
Ability to Work with Unstructured Data

Who Is a Data Analyst?

> A data analyst is the person who can do basic descriptive statistics, visualize data, and communicate data points for conclusions.

Skills Required to Become a Data Analyst

Knowledge of Mathematical Statistics



Understanding of R and Python

Data Wrangling

Understanding of Pig and Hive

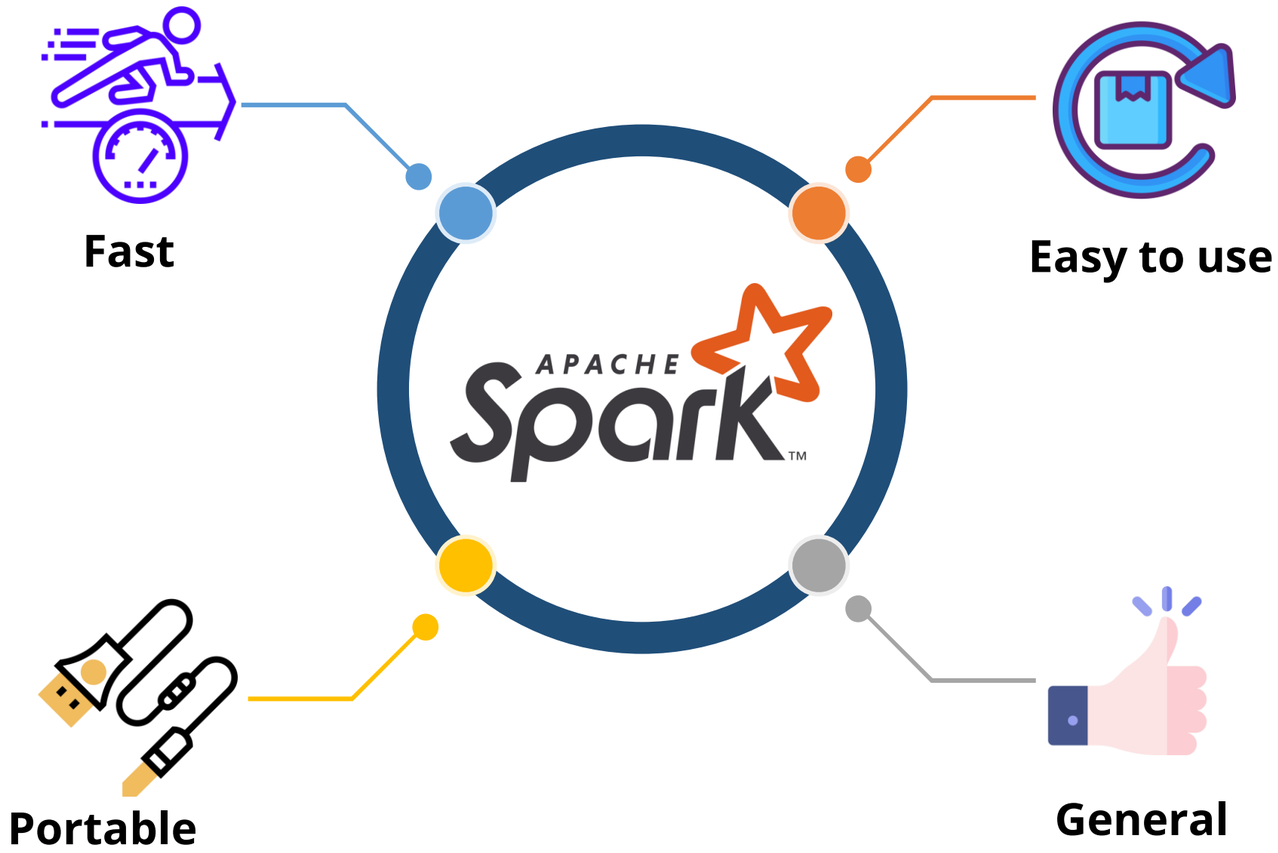
DATA AND
ARTIFICIAL INTELLIGENCE

Analytics in Spark

Analytics in Spark

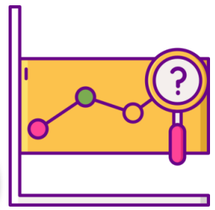
Apache Spark is a unified analytics engine for large-scale data processing.

The following are the benefits of analytics in Spark:



Types of Analytics

Descriptive Analytics

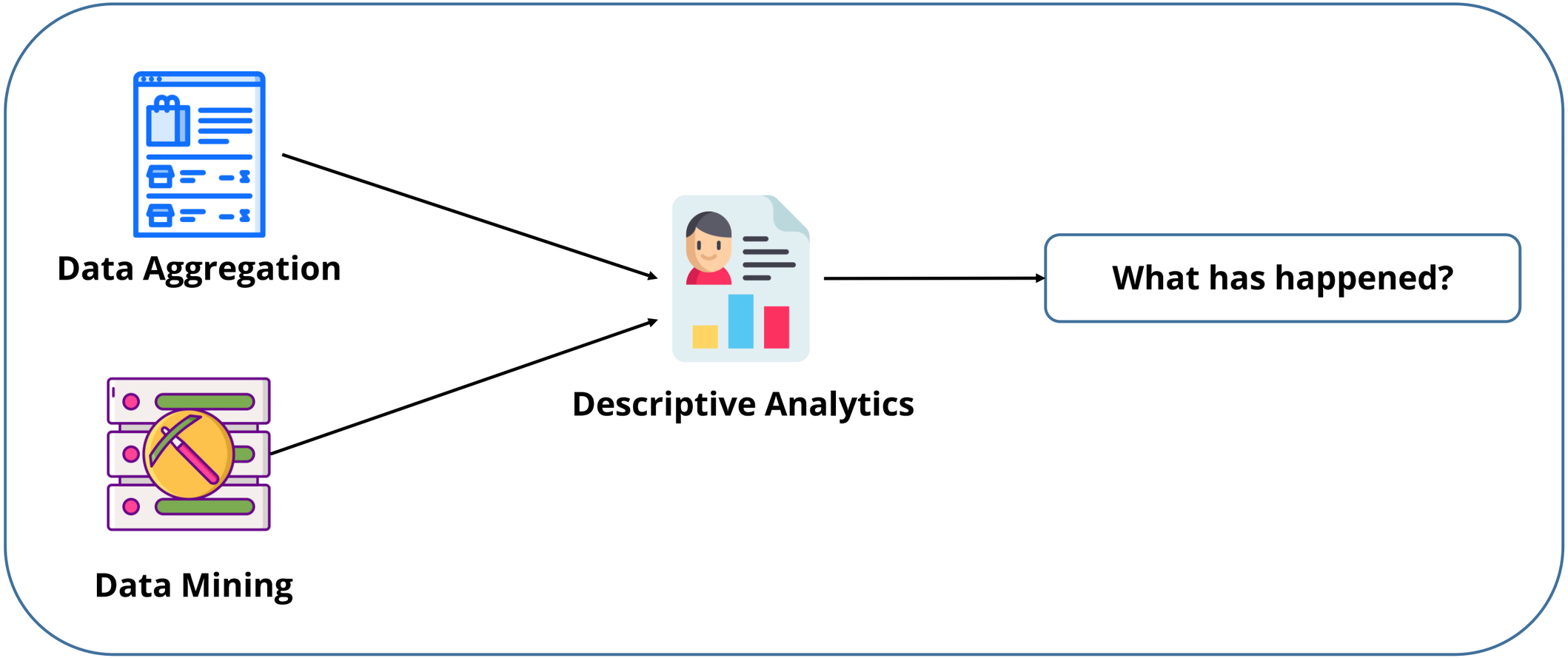


Predictive Analytics

Prescriptive Analytics

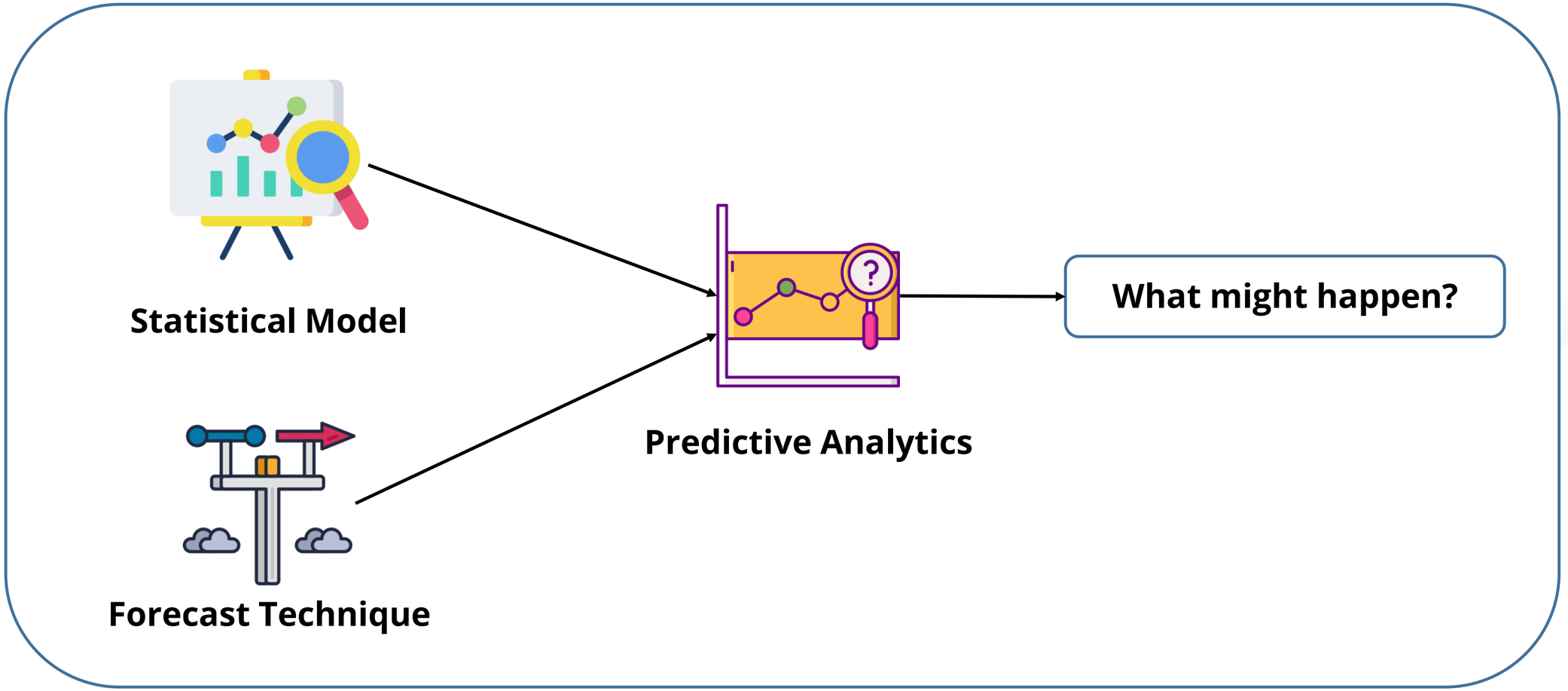
Descriptive Analytics

The type of analytics that describes the past and answers the question: “What has happened?”.



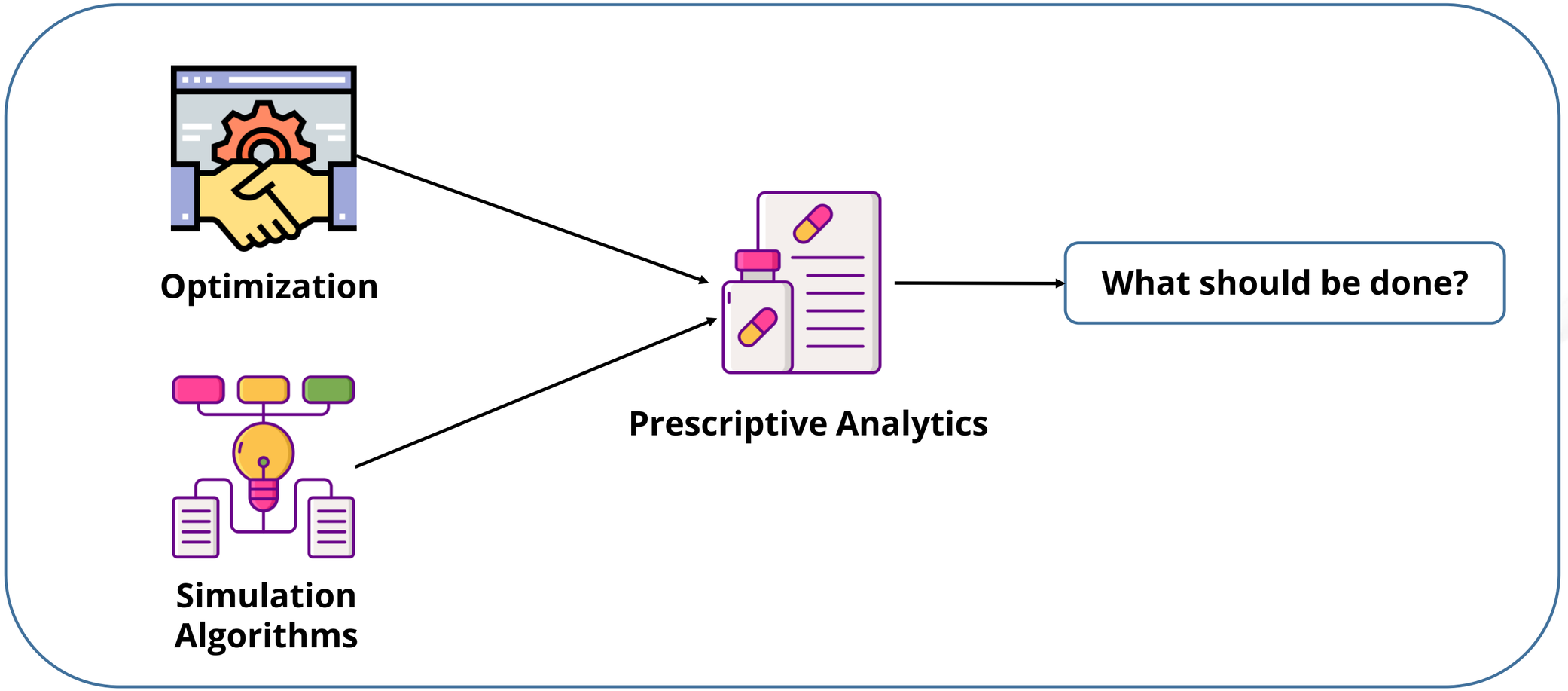
Predictive Analytics

The type of analytics that has the ability to understand the future and answer the question: “What might happen?”.



Prescriptive Analytics

The type of analytics that is used to advice the users on possible outcome and answer the question: “What should be done?”.



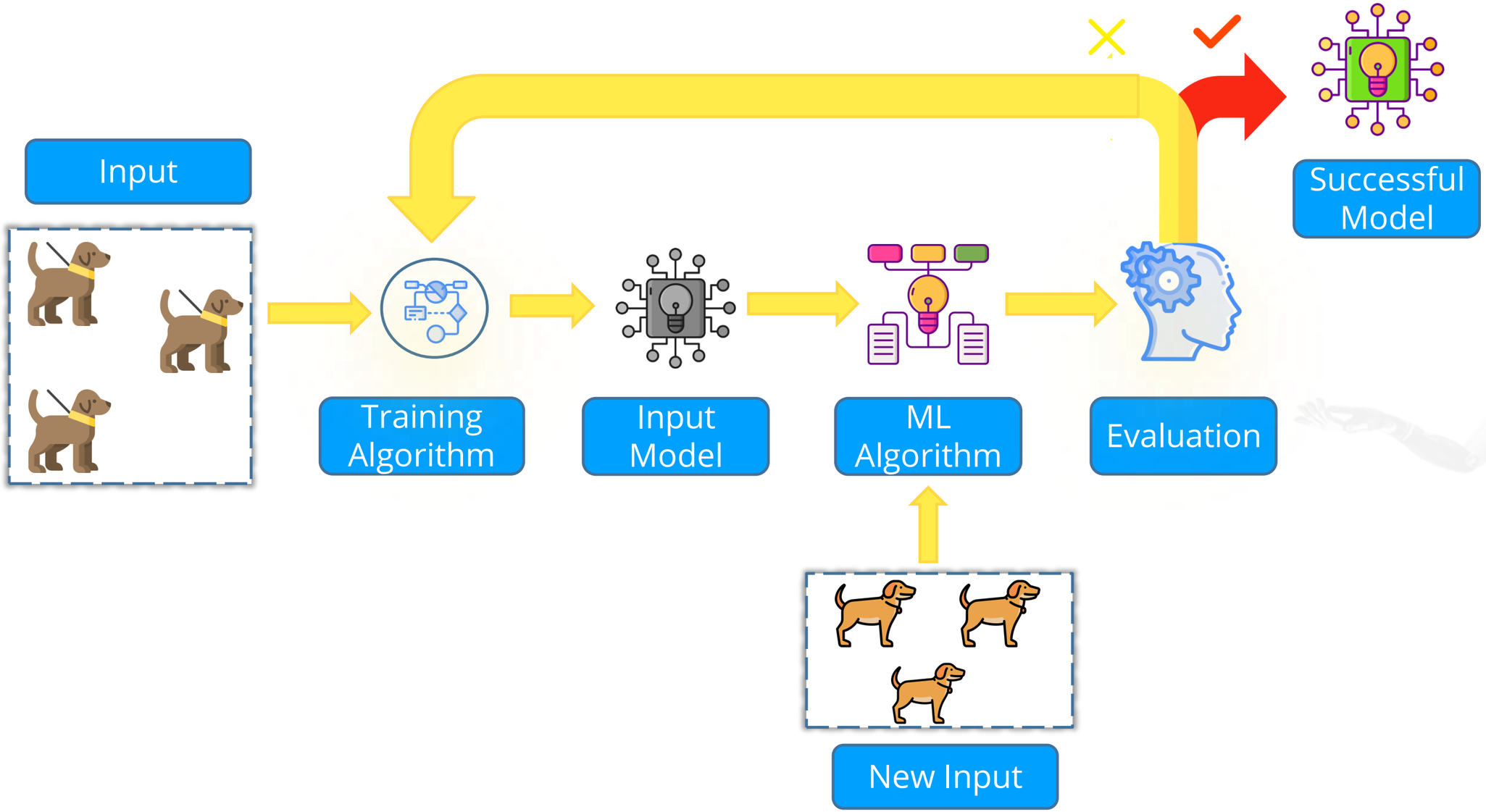
DATA AND
ARTIFICIAL INTELLIGENCE

Machine Learning

What Is Machine Learning?

> The capability of Artificial Intelligence systems to learn by extracting patterns from data is known as Machine Learning.

What Is Machine Learning?



Relationship between Machine Learning and Data Science

Data Science and Machine Learning go hand in hand.
Data Science helps evaluate data for Machine Learning algorithms.

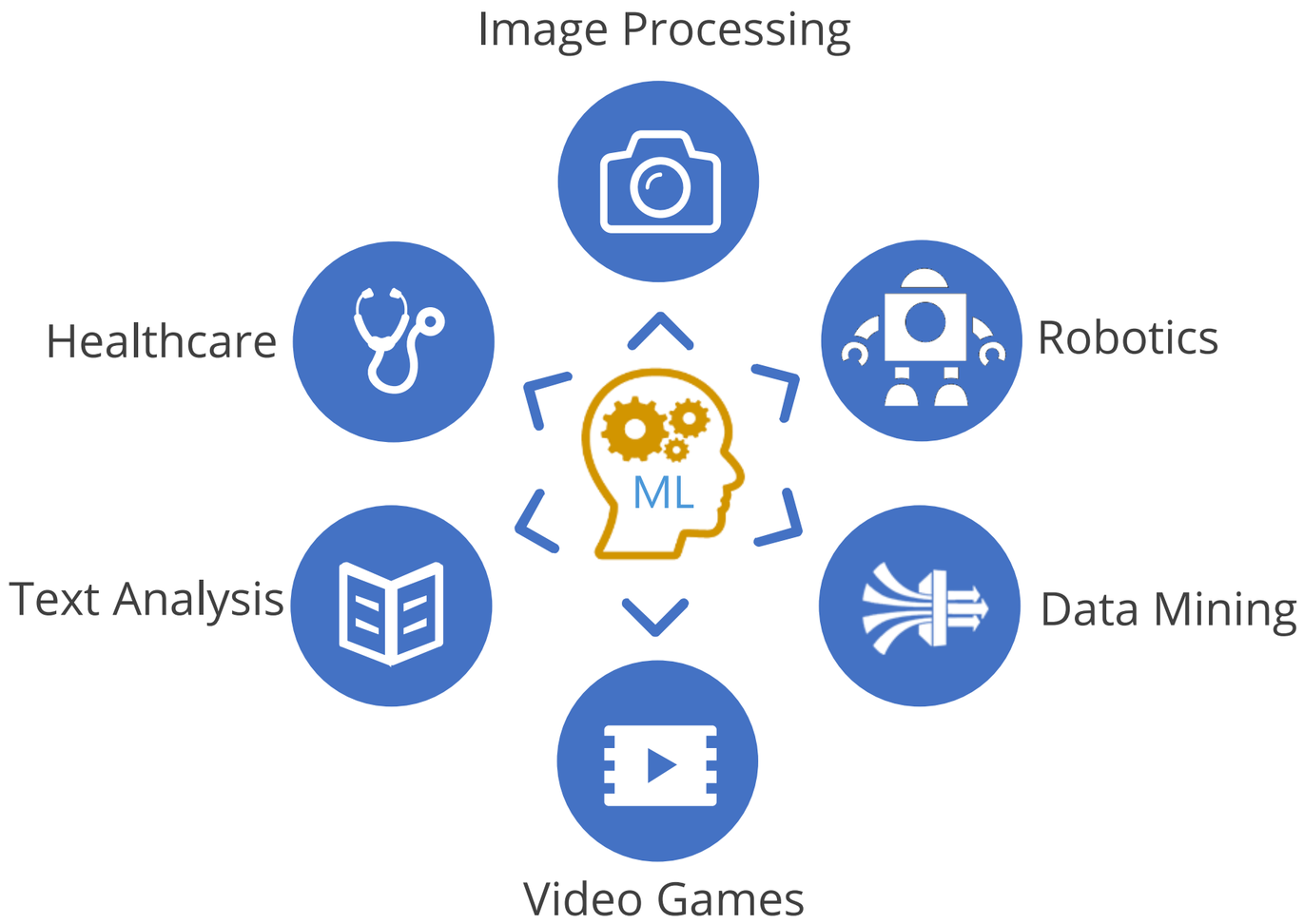
Large-Scale Machine Learning

Large-scale machine learning involves large data which has large number of training, features, or classes.

Large-Scale Machine Learning Tools

Applications of Machine Learning

Machine learning is used in various fields such as:



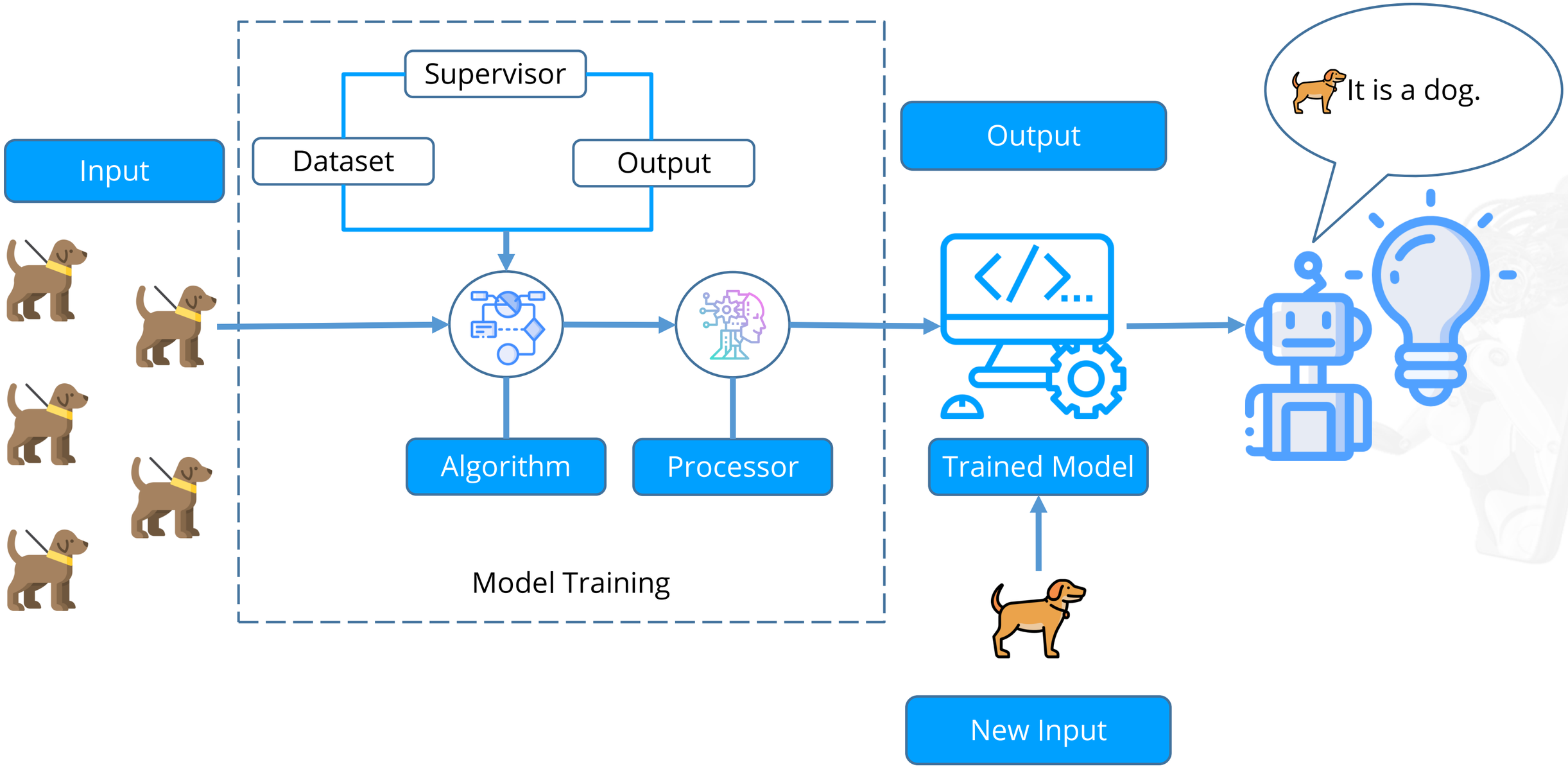
Types of ML Algorithms

- Supervised Learning
- Unsupervised Learning
- Semi-Supervised Learning
- Reinforcement Learning

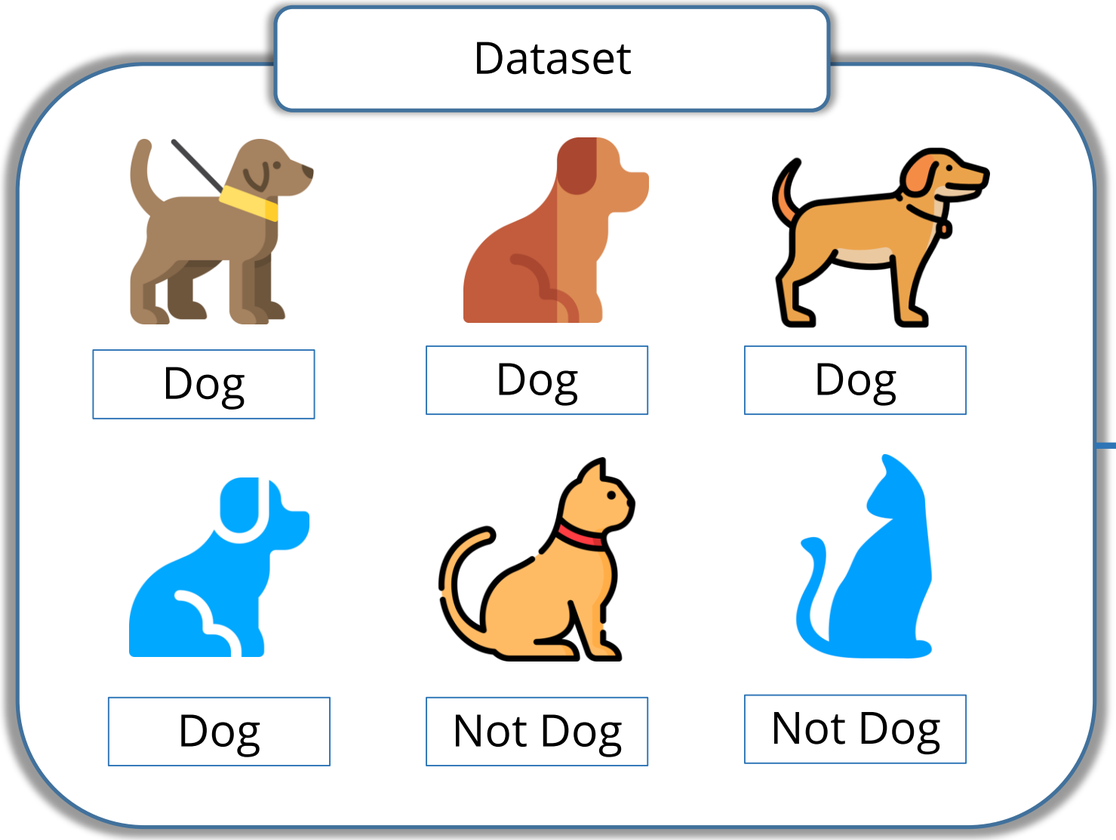
DATA AND
ARTIFICIAL INTELLIGENCE

Supervised Learning

Supervised Learning



Supervised Learning: Example



Supervised Learning

Predictive Model

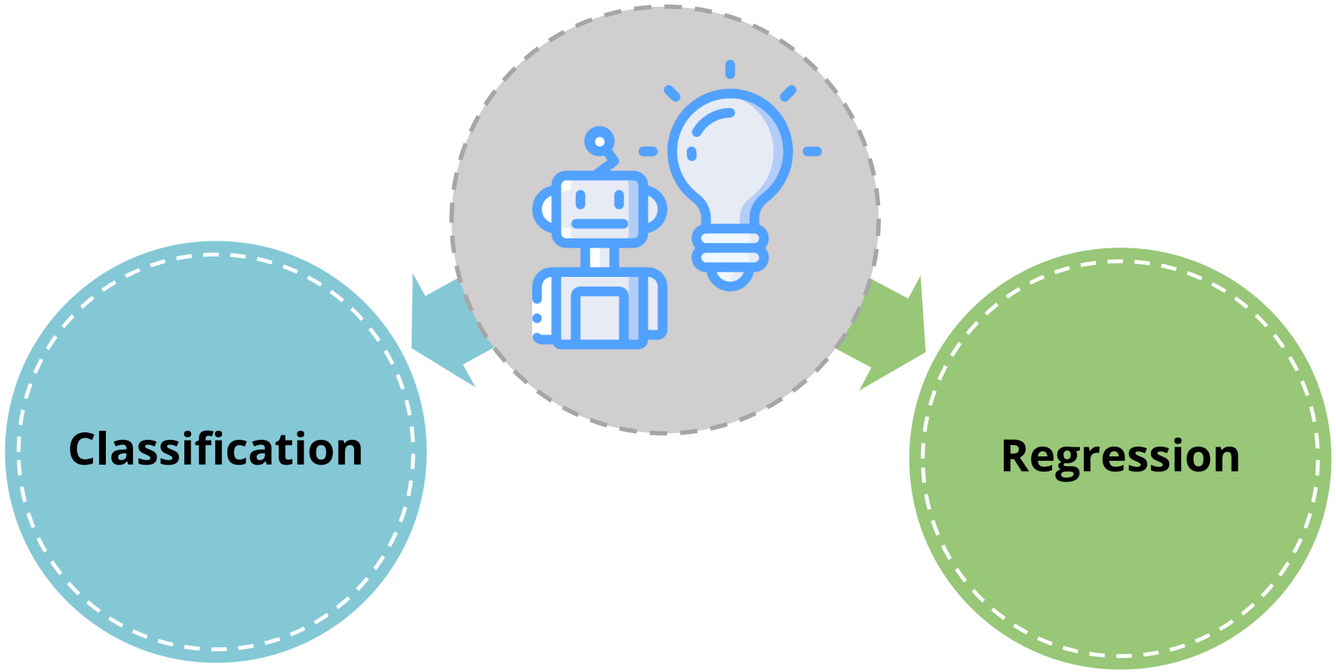
Predictive Model

Dog

Supervised Learning: Example

Netflix uses **supervised learning** algorithms to recommend users the shows they may watch based on the viewing history and ratings by similar classes of users.

Supervised Learning Algorithms



.Classification with Real-World Problem

Duration: 10 mins

Problem Statement: In this demonstration, you will perform classification with real-world problem.

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**.

Linear Regression with Real-World Problem

Duration: 10 mins

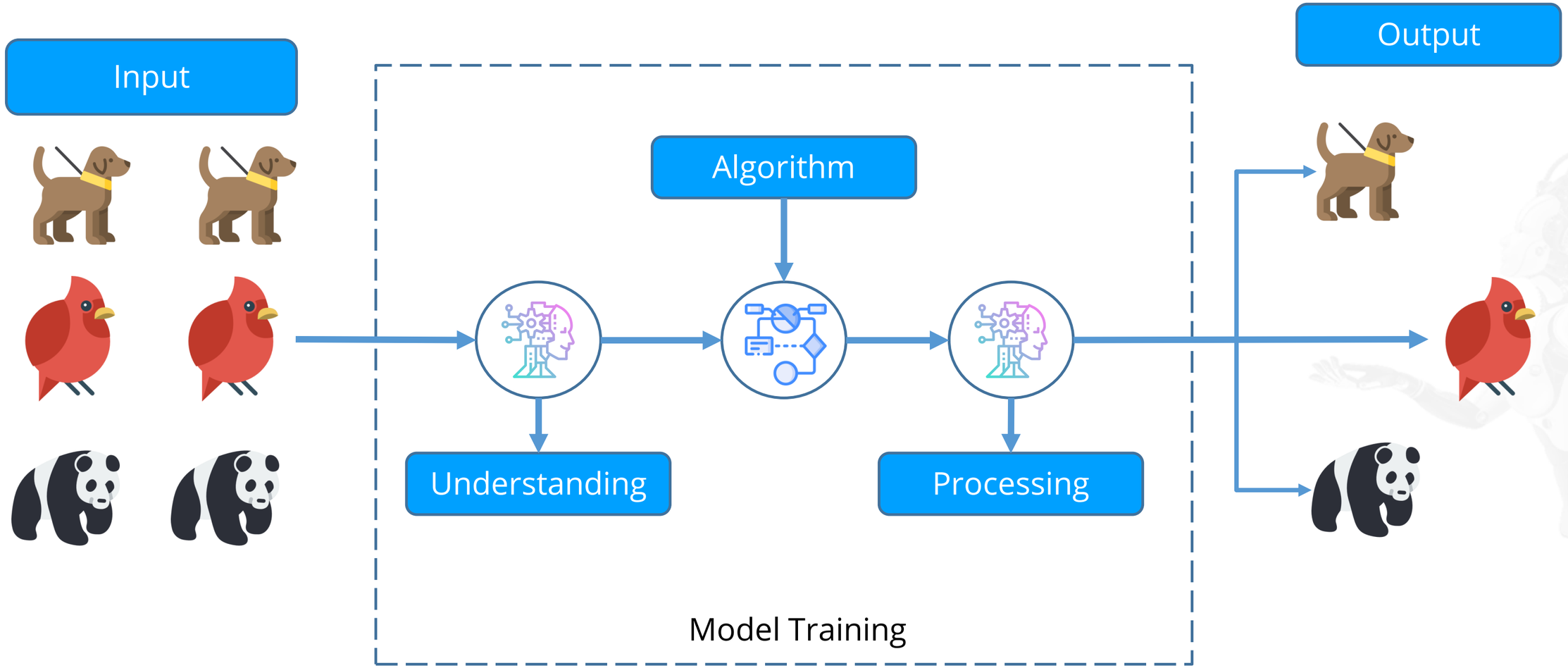
Problem Statement: In this demonstration, you will perform linear regression with real-world problem.

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**.

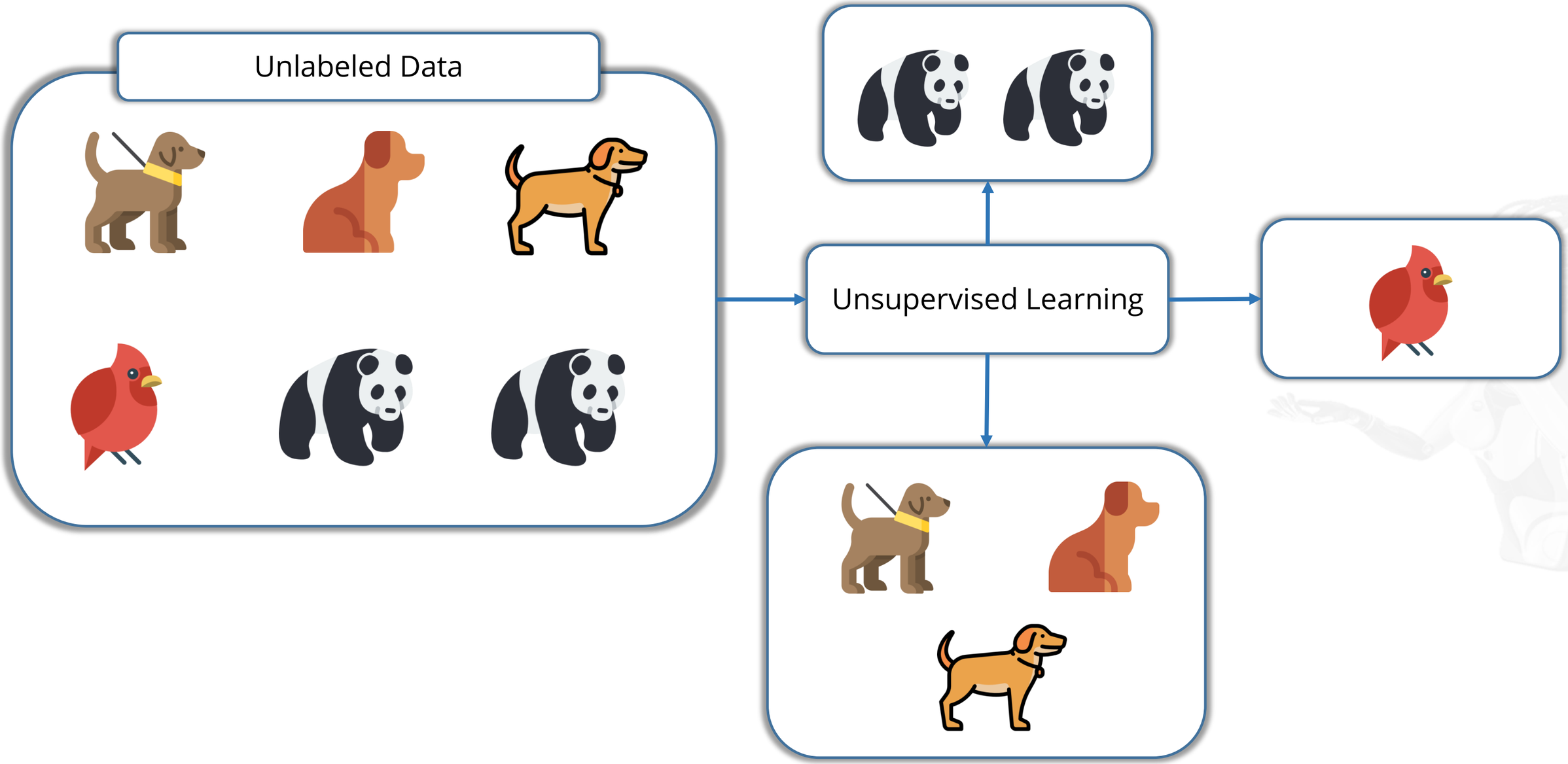
DATA AND
ARTIFICIAL INTELLIGENCE

Unsupervised Learning

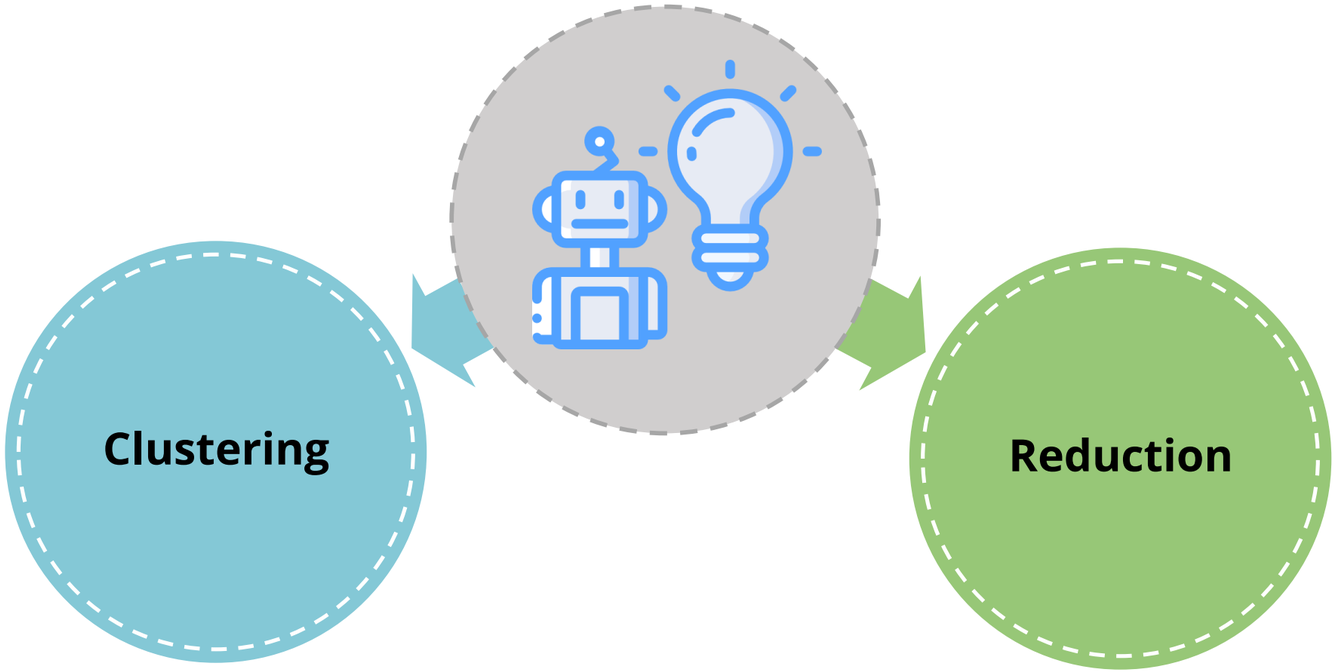
Unsupervised Learning



Unsupervised Learning: Example



Unsupervised Learning Algorithms



.Unsupervised Learning with Real-World Problem

Duration: 10 mins

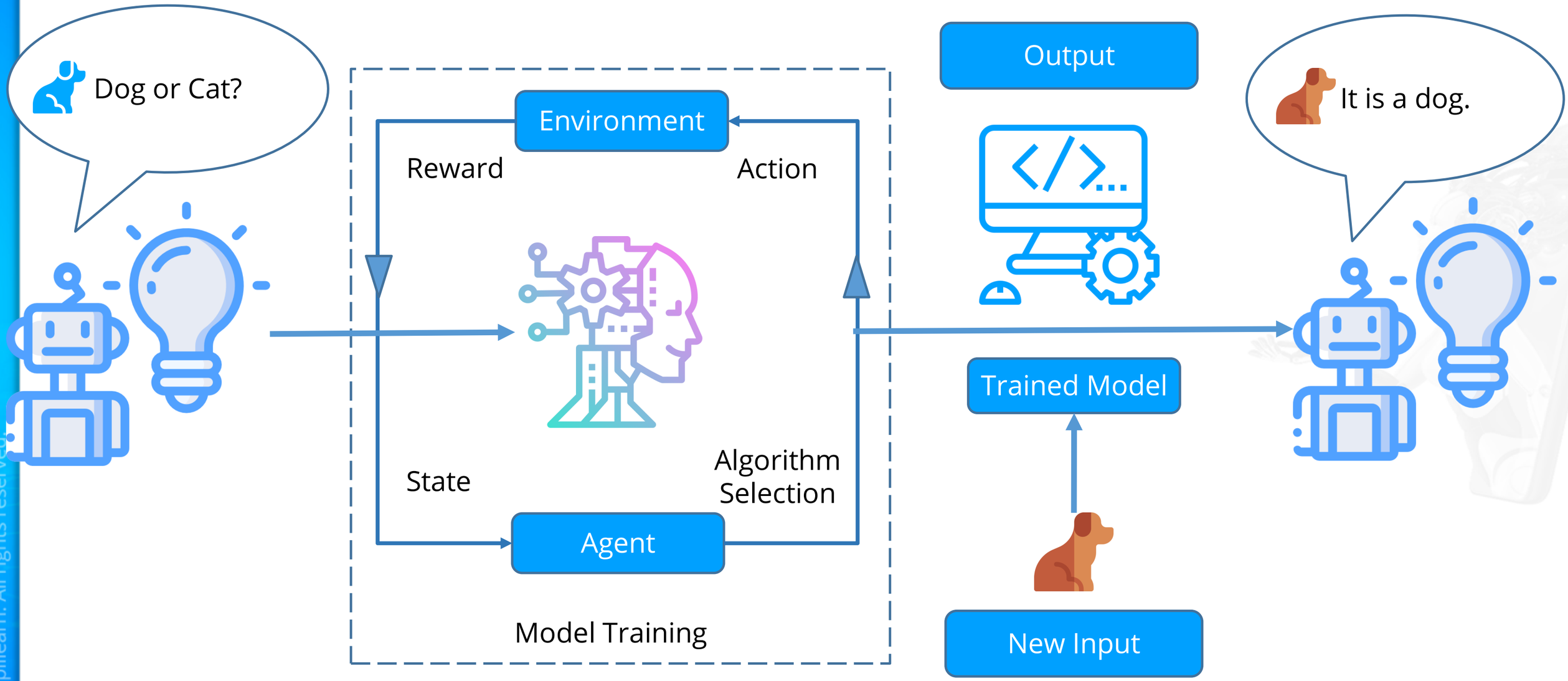
Problem Statement: In this demonstration, you will perform unsupervised learning with real-world problem.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

Reinforcement Learning

Reinforcement Learning



Reinforcement Learning: Example

1. Before Conditioning

Response

Salivation

Food

Unconditioned Stimulus

Unconditioned Response

2. Before Conditioning

Response

Bell

Salivation

Neutral Stimulus

No Conditioned Response

3. During Conditioning

Response

Salivation

Food

Bell

Unconditioned Response

4. After Conditioning

Response

Bell

Salivation

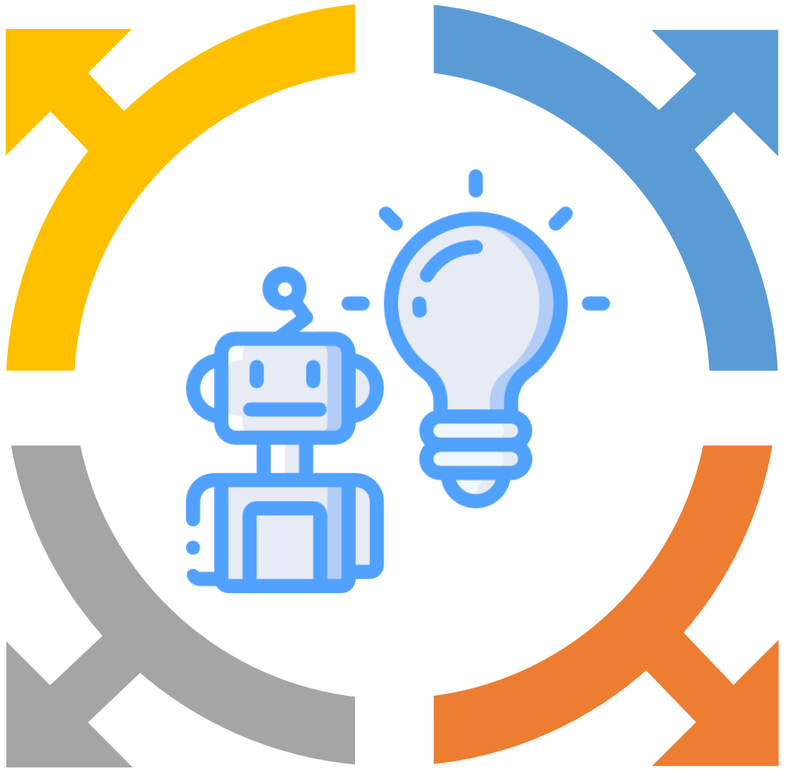
Conditioned Stimulus

Conditioned Response

Reinforcement Learning Algorithms

Q-Learning

Temporal Difference (TD)

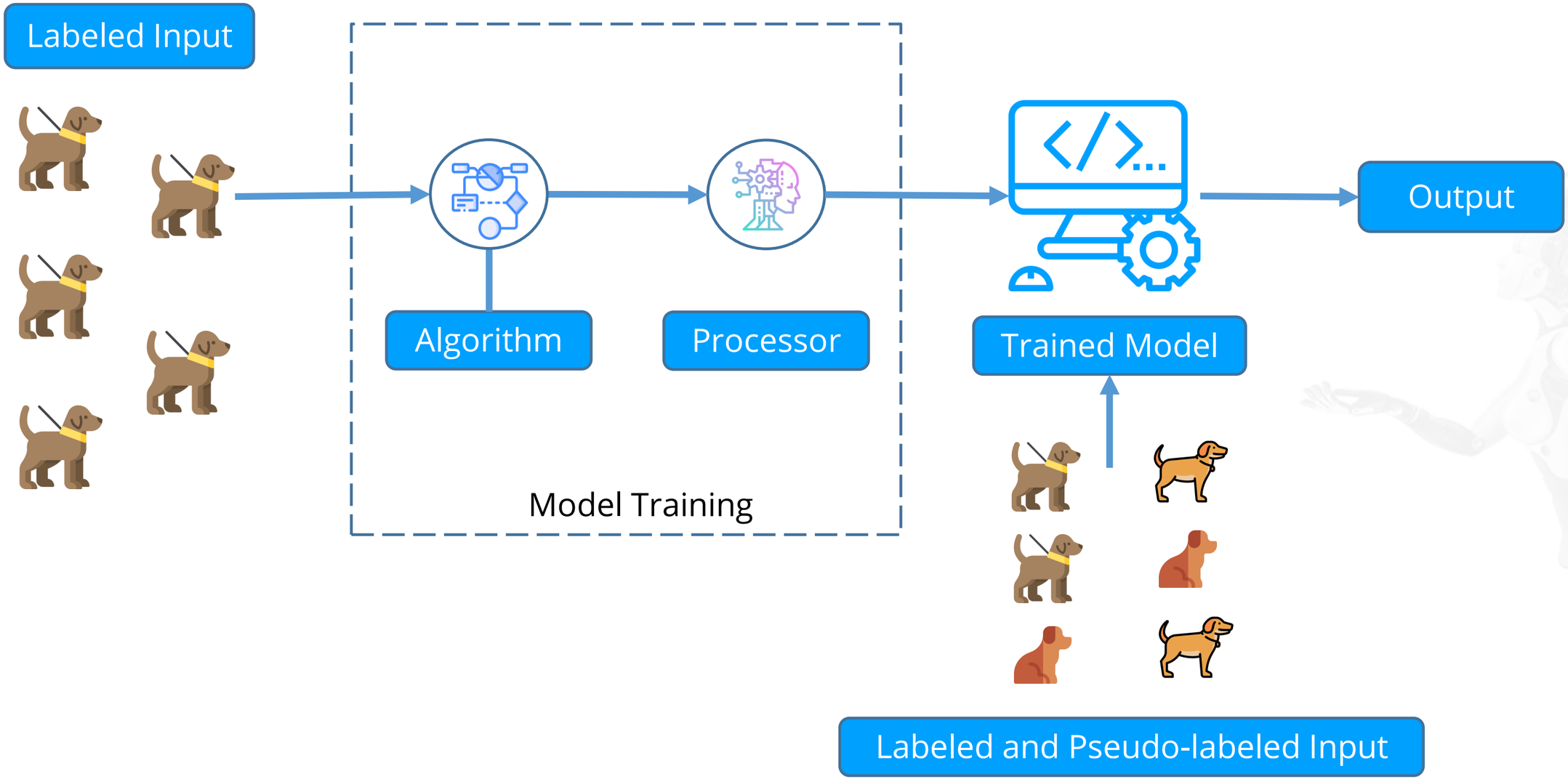


Monte-Carlo Tree Search (MCTS)

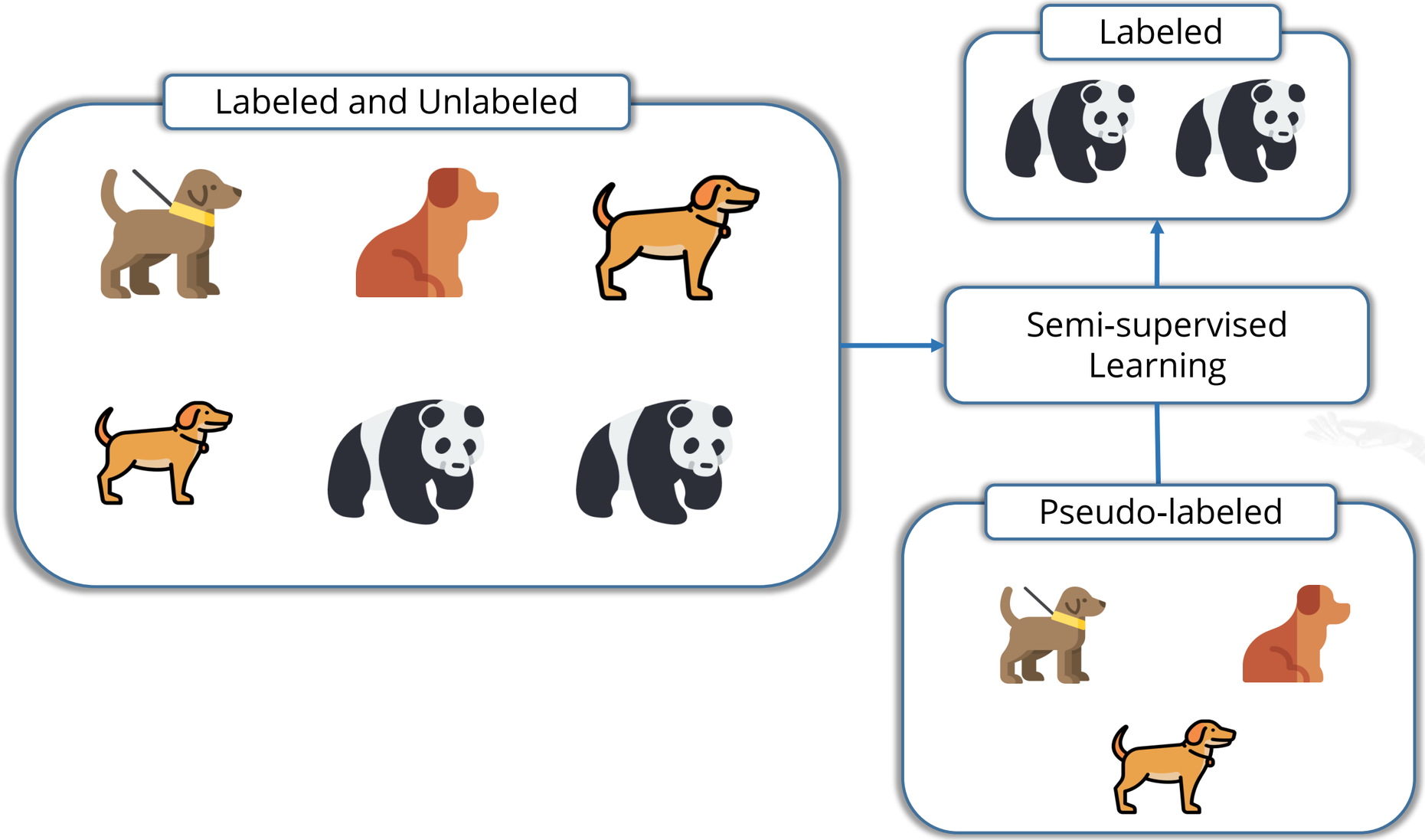
Asynchronous Actor-Critic Agents (A3C)

Semi-Supervised Learning

Semi-Supervised Learning



Semi-Supervised Learning: Example



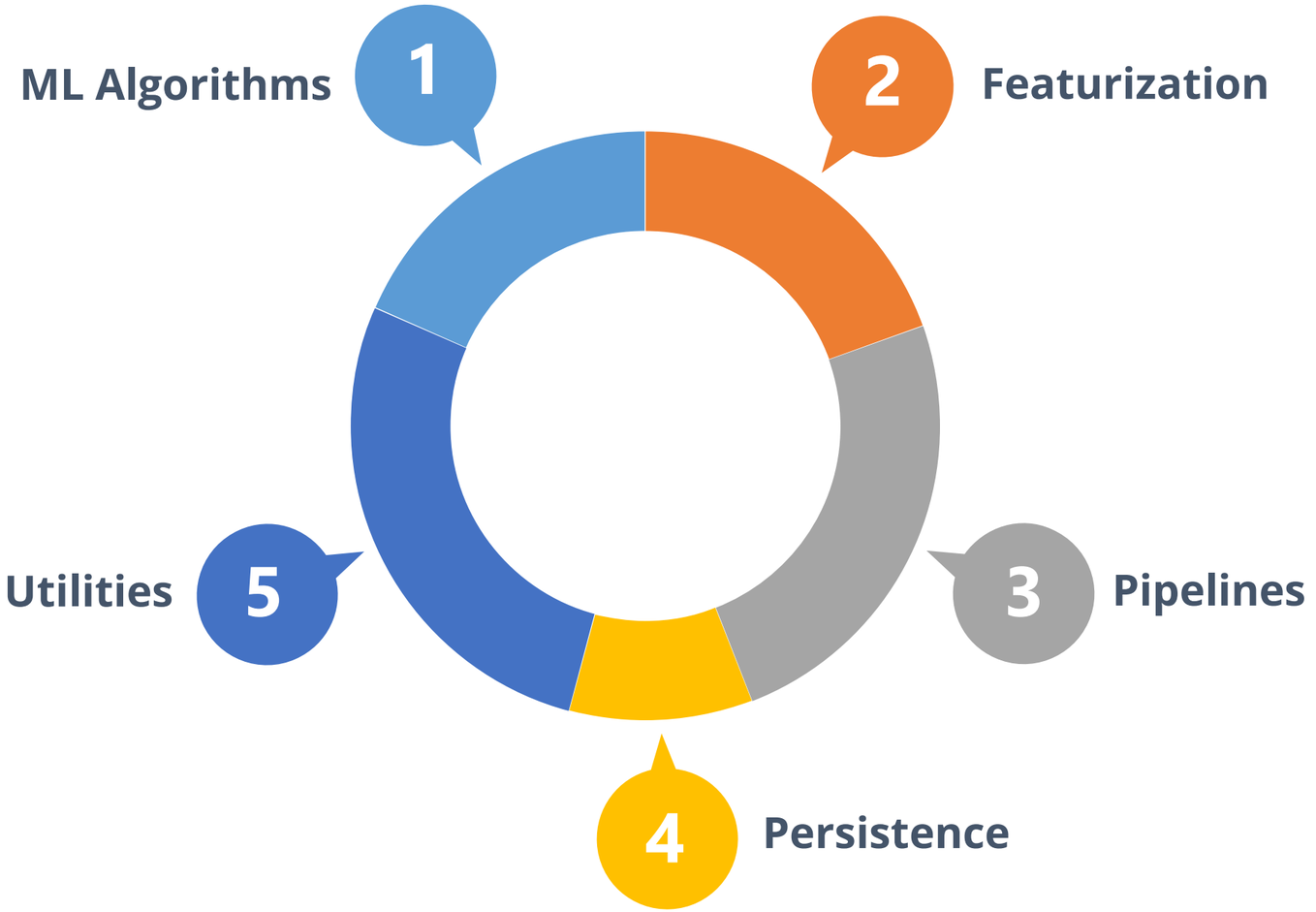
DATA AND
ARTIFICIAL INTELLIGENCE

Overview of MLlib

What Is MLlib?

> MLlib is Spark's scalable machine learning library consisting of common learning algorithms, utilities, and optimization primitives.

MLlib Tools

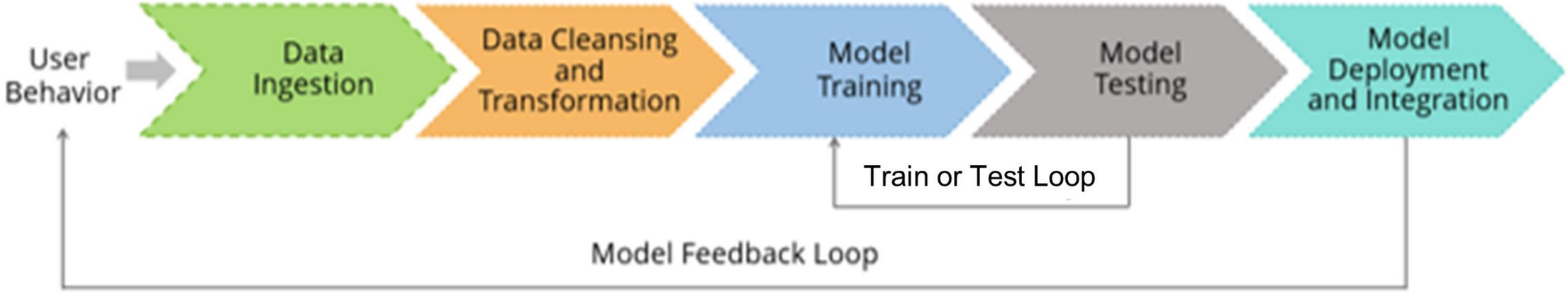


MLlib Algorithms

DATA AND
ARTIFICIAL INTELLIGENCE

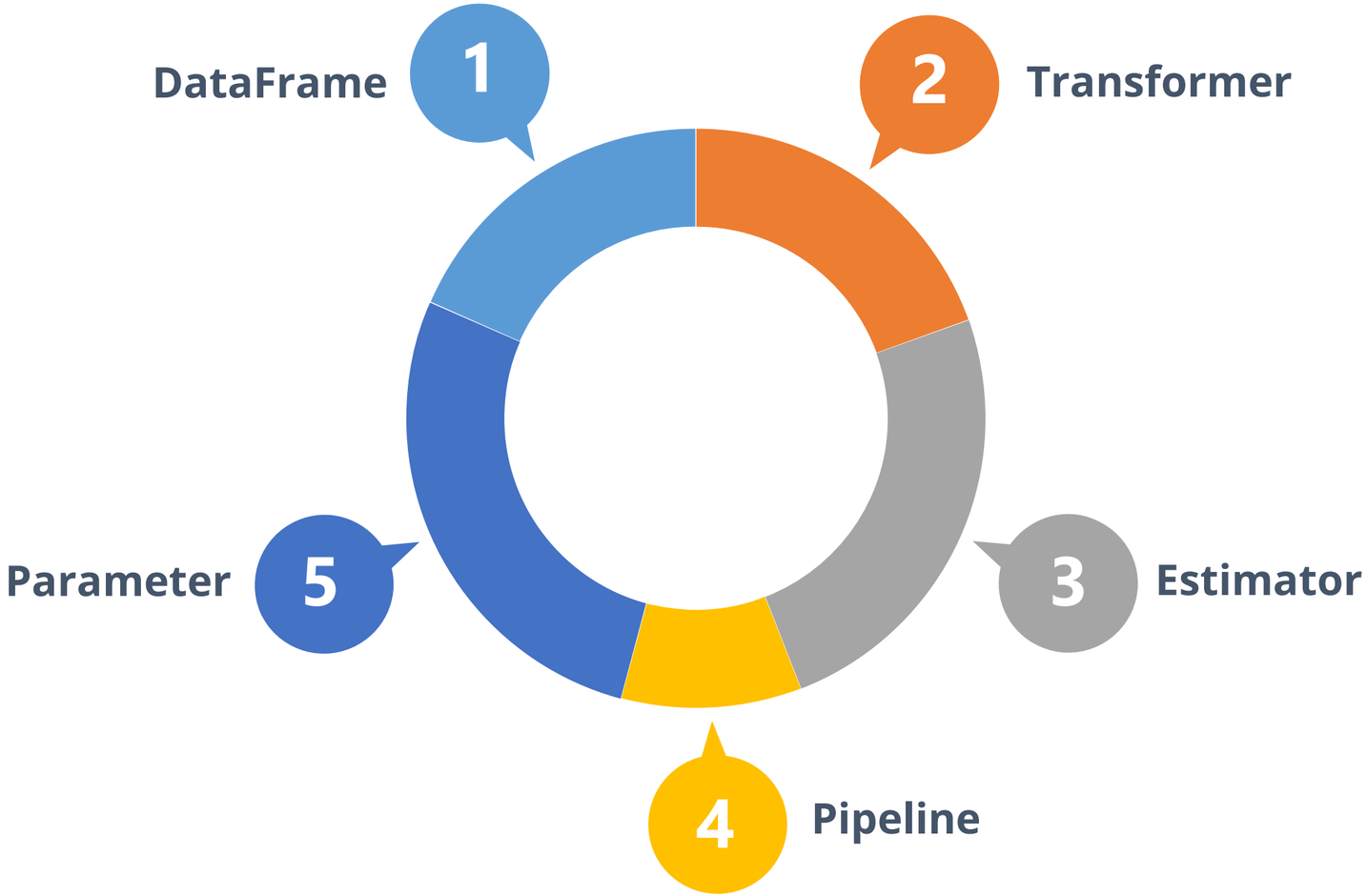
MLlib Pipelines

Machine Learning Pipeline



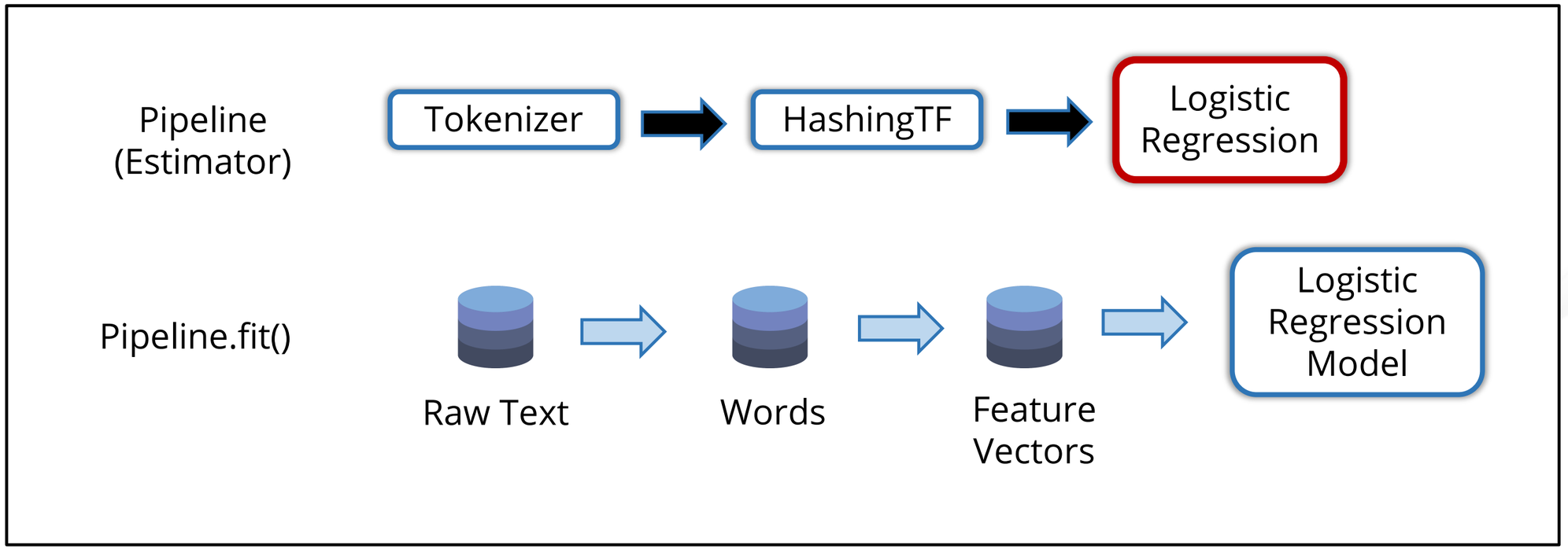
MLlib Pipelines

ML Pipelines provide a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.

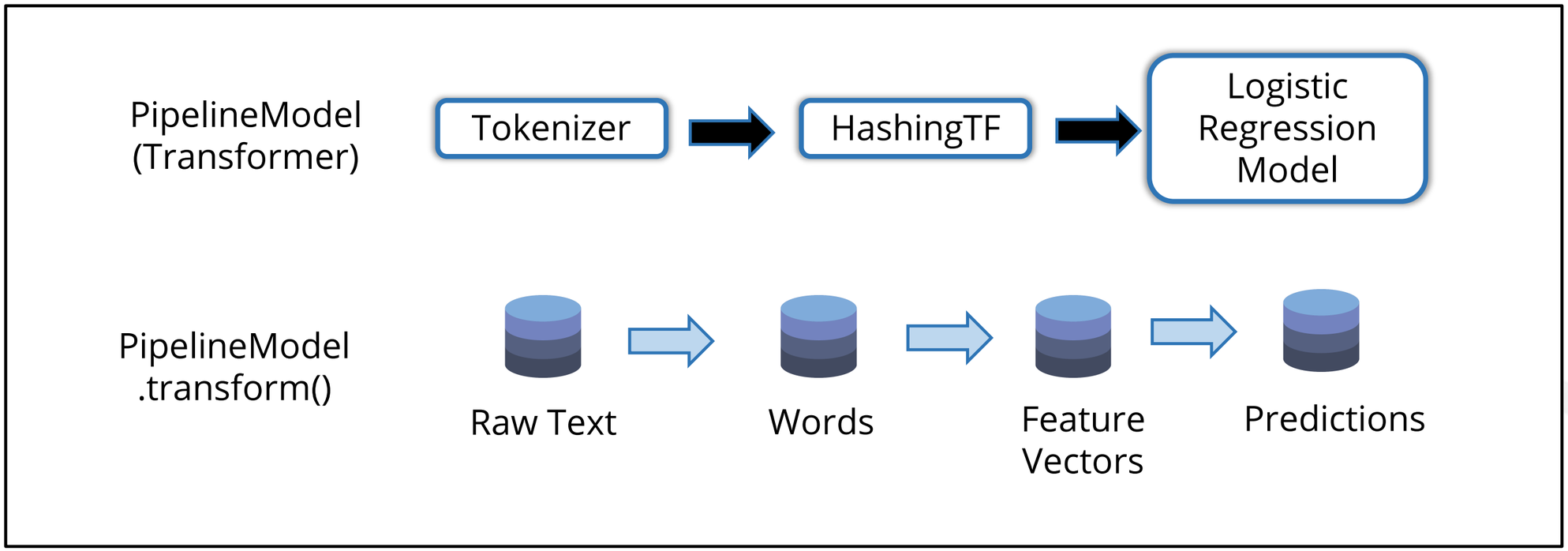


Working of Pipeline

Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator.



Working of Pipeline



Key Takeaways

You are now able to:

- Identify the skills required to become a data scientist and data analyst
- Define analytics in Spark and list the types of analytics
- Describe the machine learning algorithms
- Define MLlib and MLlib pipeline

Quiz

Knowledge Check 1

Which of the following skills are required to become a data scientist?

a. Knowledge of Python and R

b. Ability to work with unstructured data

c. Experience in SQL

d. All of the above

Knowledge Check

d

Which of the following skills are required to become a data scientist?

a. **Knowledge of Python and R**

b. **Ability to work with unstructured data**

c. Experience in SQL

d. All of the above

Knowledge of Python and R, ability to work with unstructured data, and experience in SQL are required to become a data scientist.

The correct answer is

Knowledge Check 2

Which of the following types of analytics describes the past and answers the question, "What has happened?"?

a. Descriptive analytics

b. Predictive analytics

c. Prescriptive analytics

d. None of the above

Knowledge Check

2

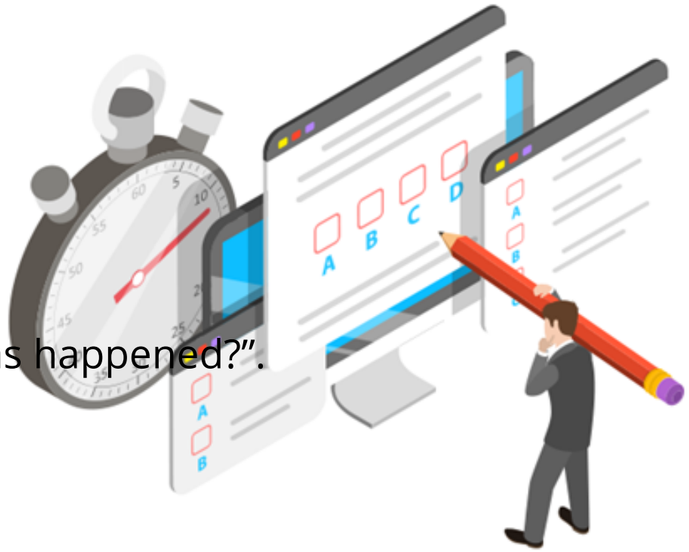
Which of the following types of analytics describes the past and answers the question, “What has happened?”?

a. **Descriptive analytics**

b. **Predictive analytics**

c. Prescriptive analytics

d. None of the above



Descriptive analytics describes the past and answers the question, “What has happened?”.

The correct answer is

Knowledge Check 3

Which machine learning algorithm develops a self-sustained system based on the interaction between the environment and the learning agent?

a. Supervised learning

b. Unsupervised learning

c. Reinforcement learning

d. None of the above

Knowledge Check

6

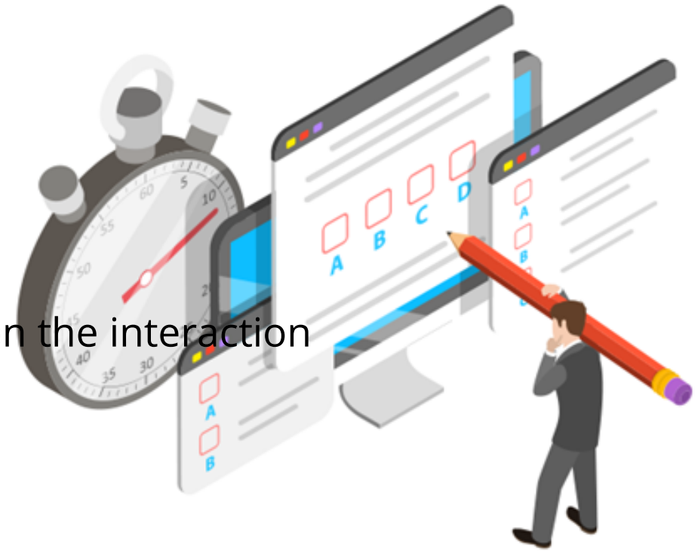
Which machine learning algorithm develops a self-sustained system based on the interaction between the environment and the learning agent?

a. **Supervised learning**

b. **Unsupervised learning**

c. Reinforcement learning

d. None of the above



Reinforcement learning algorithm develops a self-sustained system based on the interaction between the environment and the learning agent.

The correct answer is

Knowledge Check 4

Which MLlib algorithm is a statistical process for estimating the relationships among variables?

a. Classification

b. Regression

c. Clustering

d. Optimization

Knowledge Check

4

Which MLlib algorithm is a statistical process for estimating the relationships among variables?

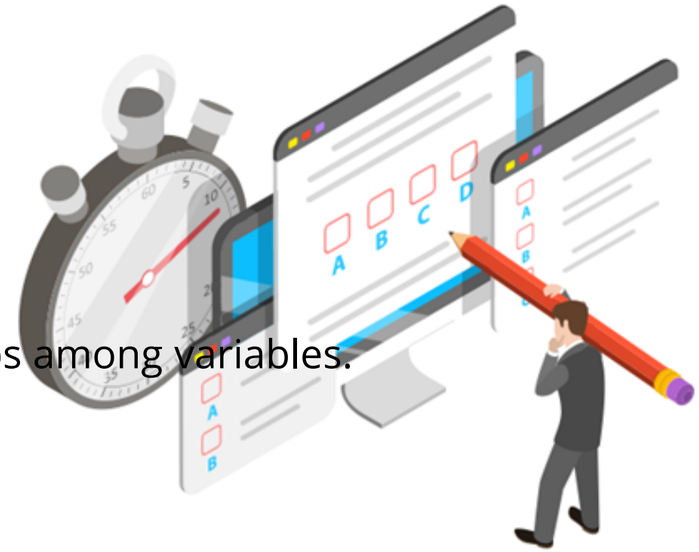
a. **Classification**

b. **Regression**

c. Clustering

d. Optimization

Regression algorithm is a statistical process for estimating the relationships among variables.



The correct answer is

Lesson-End Project

Problem Statement: The Blue Nile is one of the largest diamond retail e-commerce companies in the world. The company's revenue numbers are good this year. Recently a distributor was shutting down its business due to his financial instability and he wants to sell all of the diamonds in an open auction. This is a great opportunity for the company to expand its diamond inventory and wants to participate in the bidding. To make sure your bid is mostly accurate you must predict the correct price of the diamond.
To predict the price you should have a machine learning model and to build that you need to have the correct data first. You have collected the diamond data with all the possible diamond features concerned with deciding the price of the diamond.

Lesson-End Project

You have created a diamond.csv file which has the following details:
1. Index: counter
2. Carat: Carat weight of the diamond
3. Cut: Describe the cut quality of the diamond. Quality in increasing order: Fair, Good, Very Good, Premium, Ideal
4. Color: Color of the diamond, with D being the best and J the worst
5. Clarity: How obvious are the inclusions are in the diamond: (in order from best to worst, FL = flawless, I3= level 3 inclusions)
6. Depth: depth %: The height of the diamond, measured from the culet to the table, divided by its average girdle diameter
7. Table: table%: The width of the diamond's table expressed as a percentage of its average diameter
8. Price: the price of the diamond
9. X: length mm
10. Y: width mm
11. Z: depth mm

As a business analyst of the company, you are assigned the task of recommending the bid amount for the diamond that the company should bid for using the model built by the analytics team.

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Stream Processing Frameworks and Spark Streaming

Learning Objectives

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

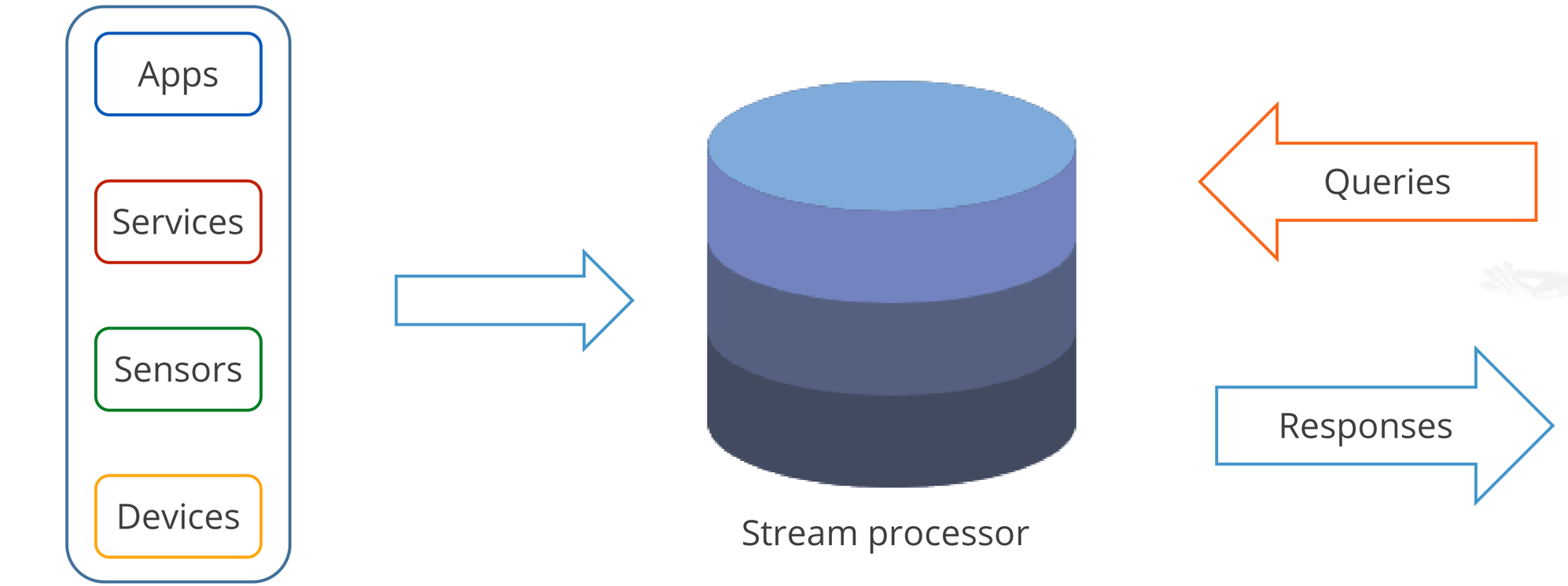
- Explain concepts of Spark Streaming
- Understand the Lambda and Kappa architecture
- Explain the concepts of Spark Structured Streaming
- Explain Join and Window operations

DATA AND
ARTIFICIAL INTELLIGENCE

Streaming Overview

What Is Streaming?

Big data streaming involves processing continuous streams of data in order to extract Real-time insights.



Need for Real-Time Processing

Certain tasks require big data processing as quickly as possible. For example:

Delays in tsunami prediction can cost people's lives

Delays in traffic jam prediction cost extra time

Advertisements can lose popularity, if not correctly targeted

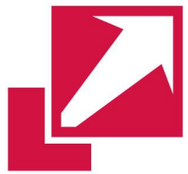
NoSQL Popularity

NoSQL databases are commonly used to solve challenges posed by stream processing techniques.

Minimal Latency

Unstructured Data

High Storage



Scalability

DATA AND
ARTIFICIAL INTELLIGENCE

Real-Time Processing of Big Data

Real-time Processing of Big Data

Real-time processing consists of continuous input, processing, and analysis of reporting data.

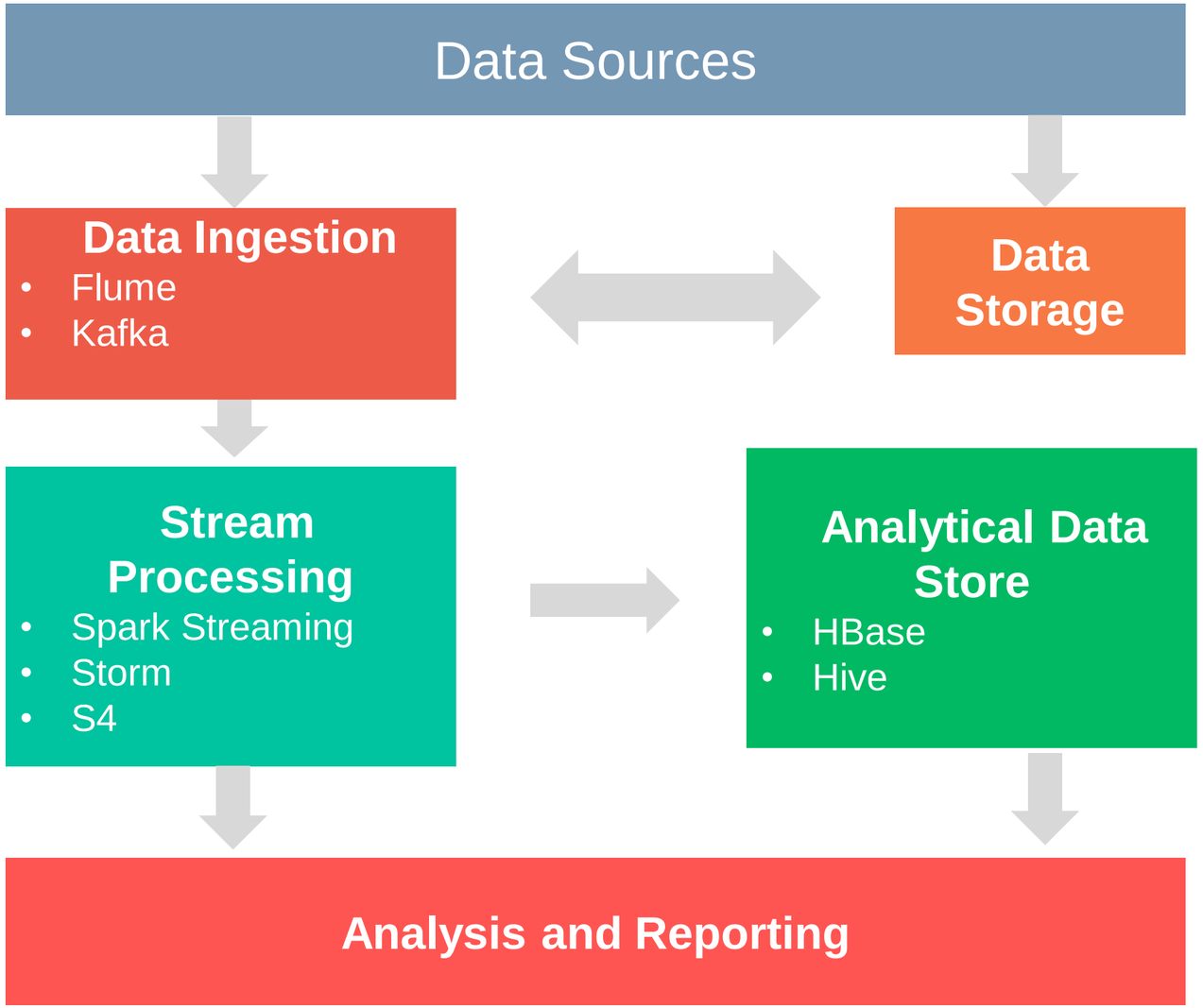
01 The process consists of a sequence of repeated operations, in which the data streams are transferred to the memory.

02 Real-time processing is crucial in order to continue the high-level functionality of automated systems having intensive data streams and different data structures.

03 For example: Bank ATMs, radar systems, disaster management systems, internet of things, and social networks.

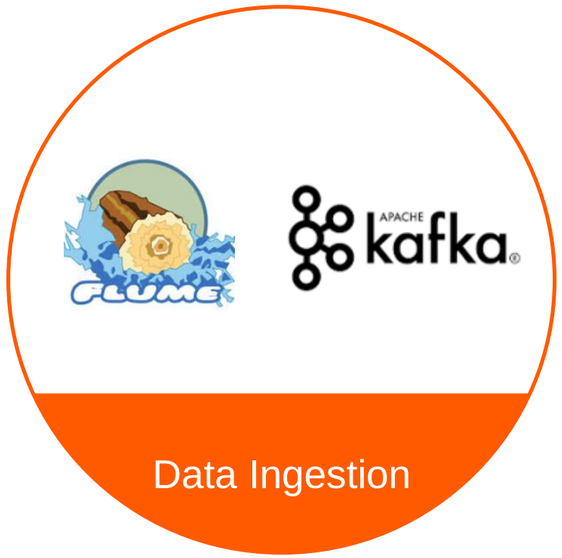
Real-Time Big Data Processing Lifecycle

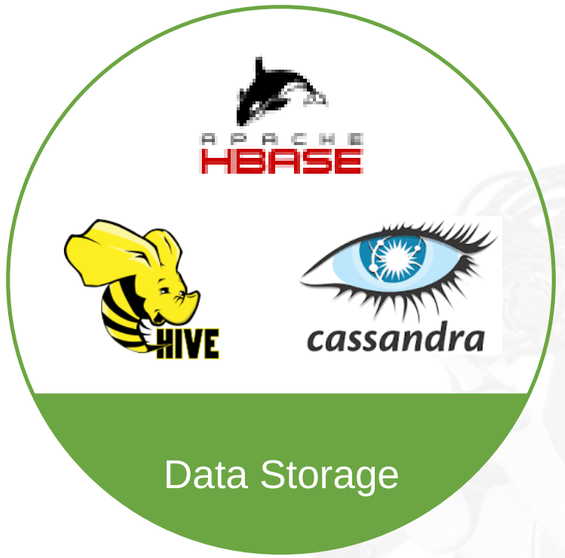
The below diagram shows the lifecycle of real-time big data processing.

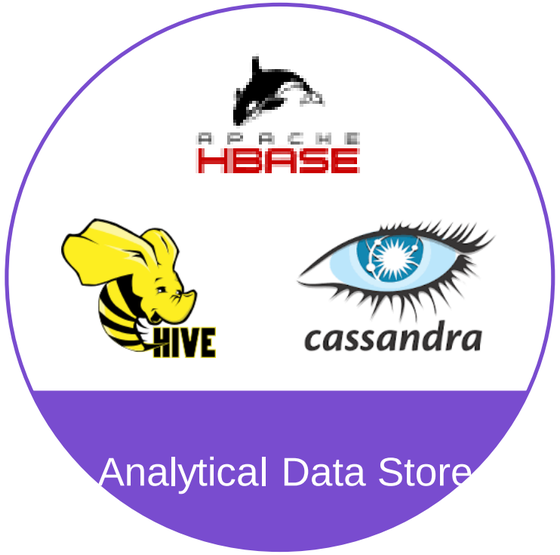


Real-Time Big Data Processing Tools

Below are some of the popular tools for Real-time Big data processing.







DATA AND
ARTIFICIAL INTELLIGENCE

Data Processing Architectures

Data Processing Architecture

A good architecture for Real-time processing should have the following properties.

1. **Fault-tolerant and scalable**
2. **Supportive of batch and incremental updates**
3. **Extensible**

The Lambda Architecture

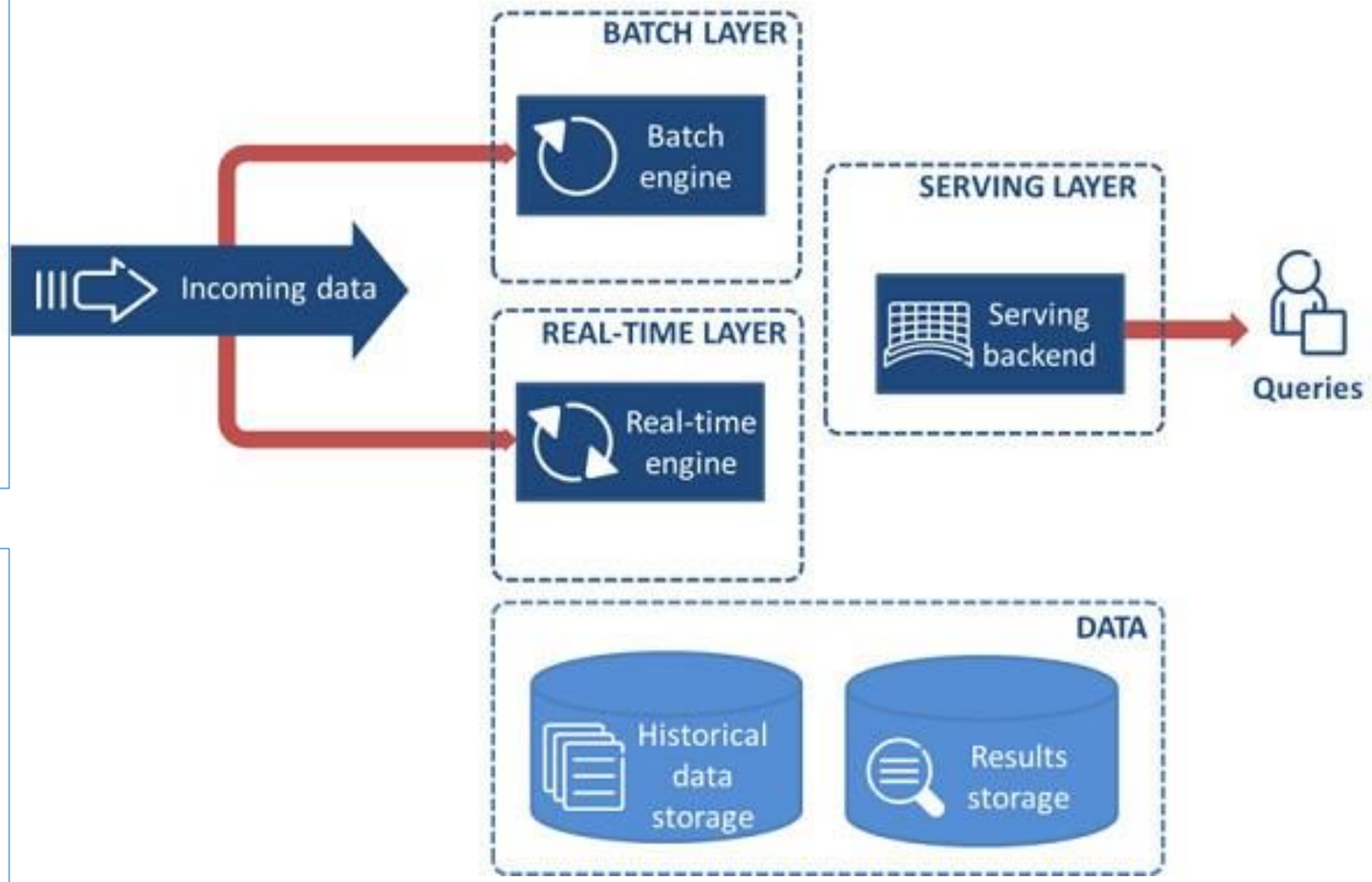
The Lambda Architecture is composed of three layers: Batch, Real-Time, and Serving

Batch Layer

- Stores the raw data as it arrives
- Computes the batch views for consumption
- Manages historical data
- Re-computes results such as machine learning models
- Operates on full data
- Produces most accurate results
- Has a high cost of high latency due to high computation time

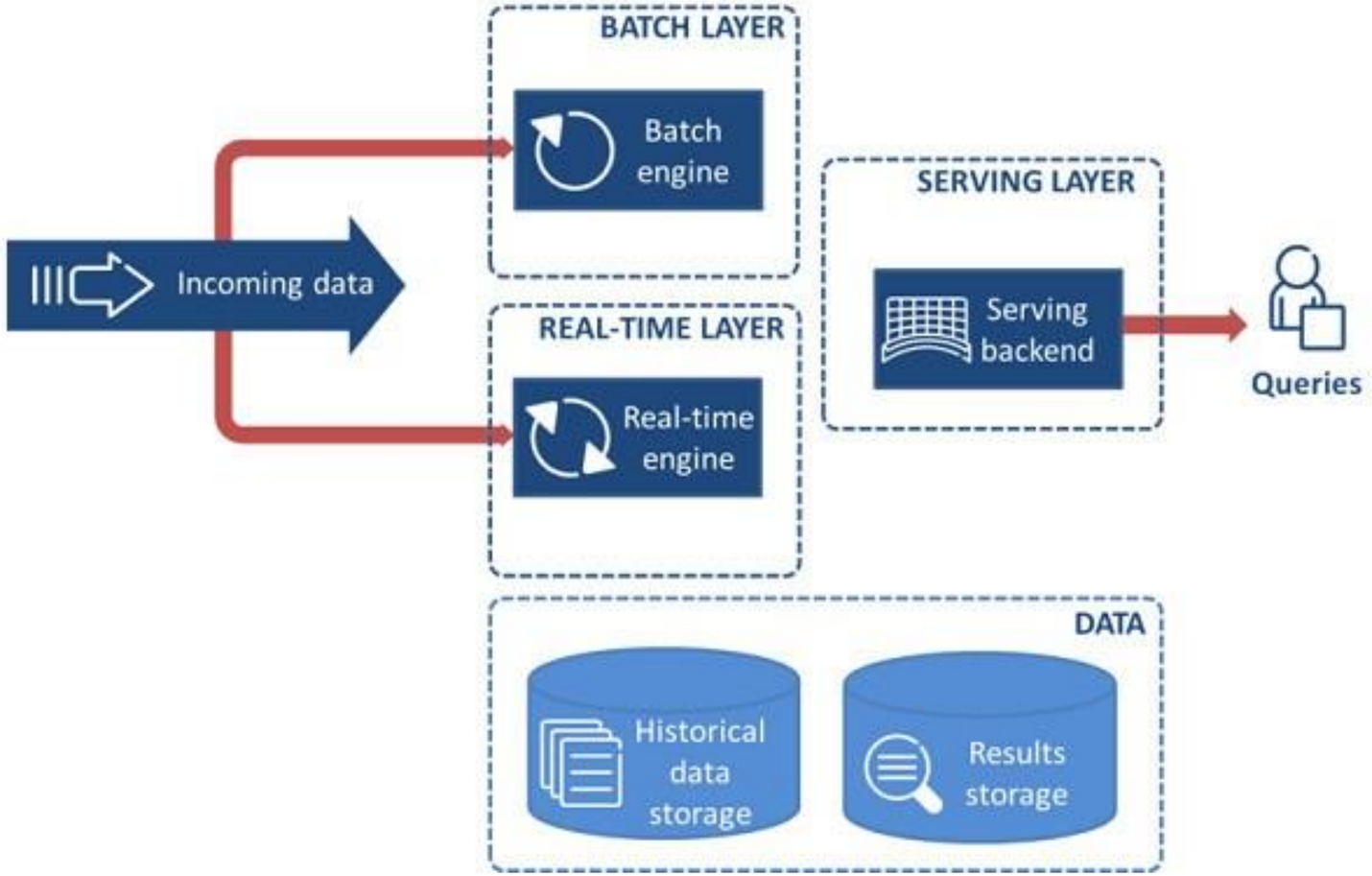
Real-Time Layer

- Receives the arriving data
- Performs incremental updates to the batch layer results
- Has incremental algorithms implemented at the speed layer
- Has a significantly reduced computation cost



The Kappa Architecture

The Kappa Architecture only processes data as a stream.



Use Case

Case Scenario: Twitter wanted to improve mobile experience for its users.

Problem: A complex system that receives events, archives them, performs offline and real-time computations, and merges the results of those computations into coherent information

Goal: To reduce impact on battery and network usage; ensuring data reliability and getting the data over as close to real time as possible

Solution: To reduce impact on the device, analytics events are compressed and sent in batches.

Process: Twitter utilized the Lambda architecture to achieve this. The architecture consists of four major components: event reception, event archival, speed computation, and batch computation.

Result: This has helped provide app developers with reliable, real-time and actionable insights into their mobile applications.

Assisted Practice

Real-Time Data Processing

Duration: 15 mins

Problem Statement: In this demonstration, you will learn the basics of real-time processing.

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**.

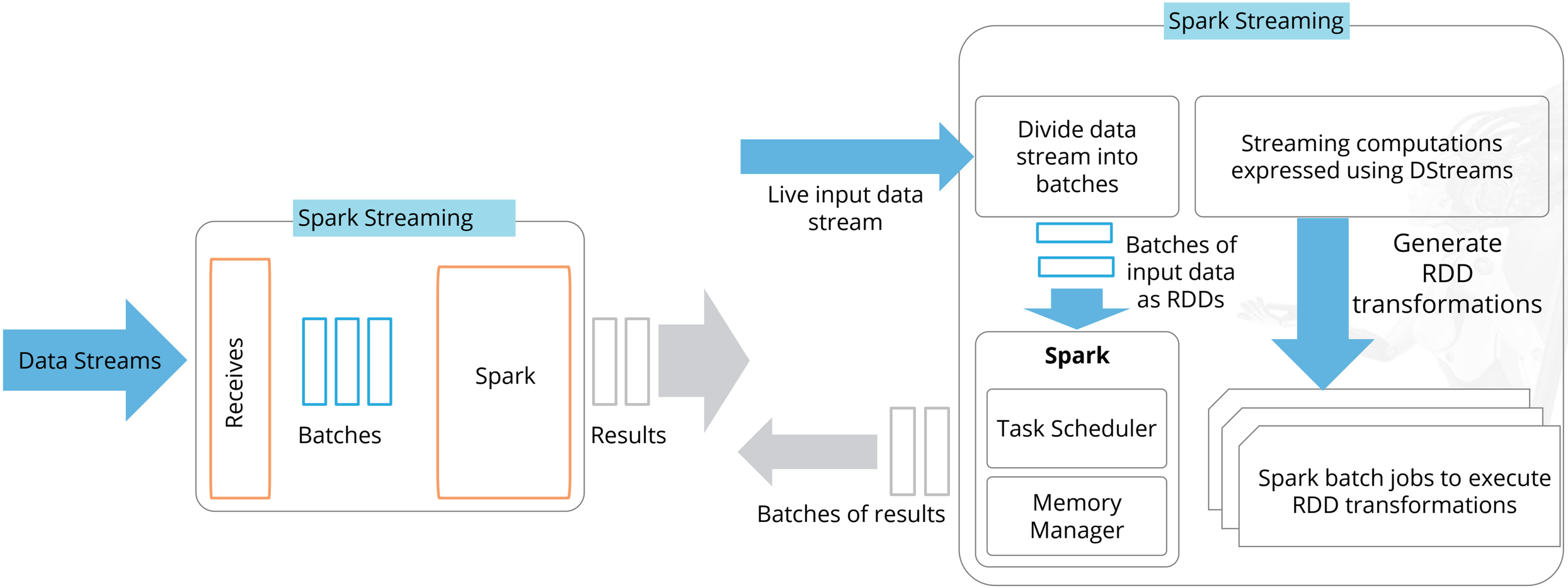
DATA AND
ARTIFICIAL INTELLIGENCE

Spark Streaming

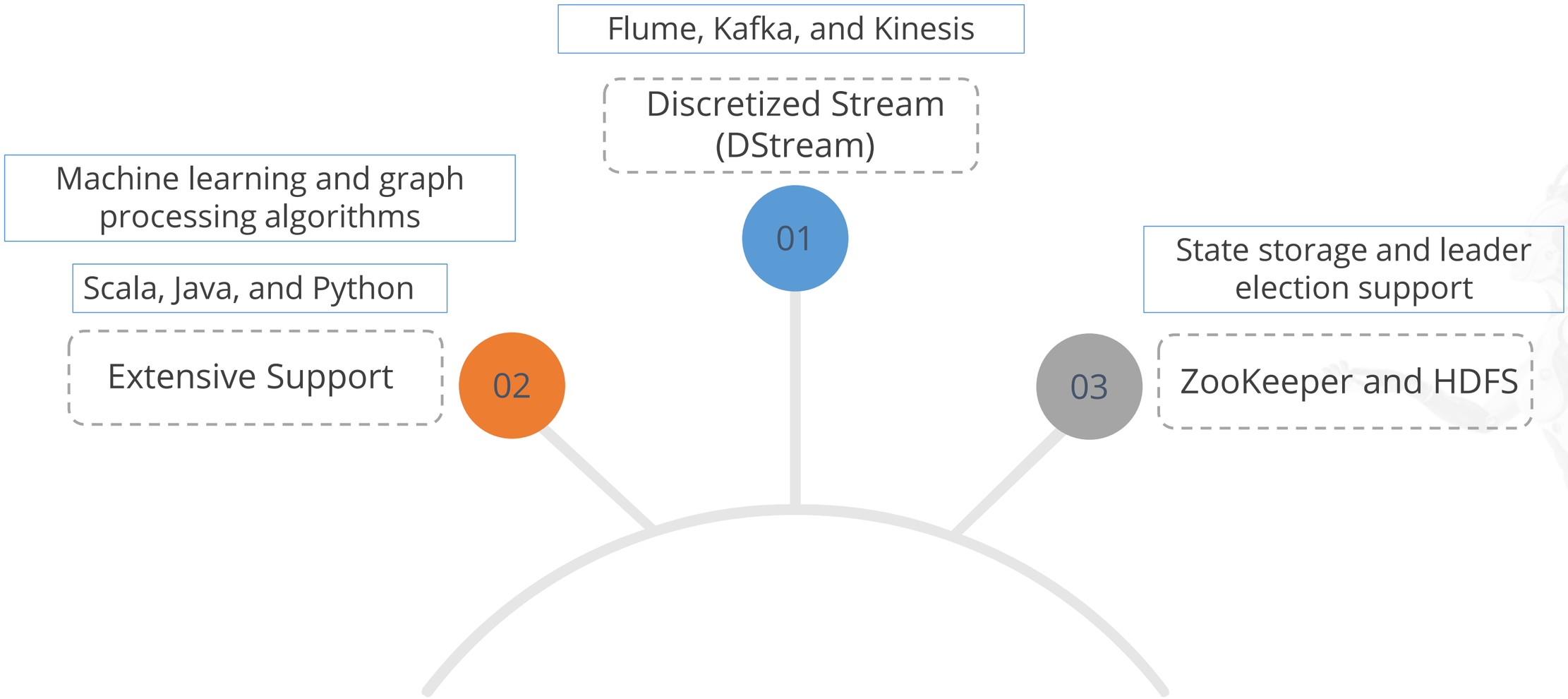
Introduction to Spark Streaming

Spark Streaming is an extension of the core Spark API.

Working of Spark Streaming



Features of Spark Streaming



Streaming Word Count

Sample code to execute Spark Streaming:

Example:

```
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")
val ssc = new StreamingContext(conf, Seconds(1))
val lines = ssc.socketTextStream("localhost", 9999)
val words = lines.flatMap(_.split(" "))
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)
wordCounts.print()
```

Assisted Practice

Writing Spark Streaming Application

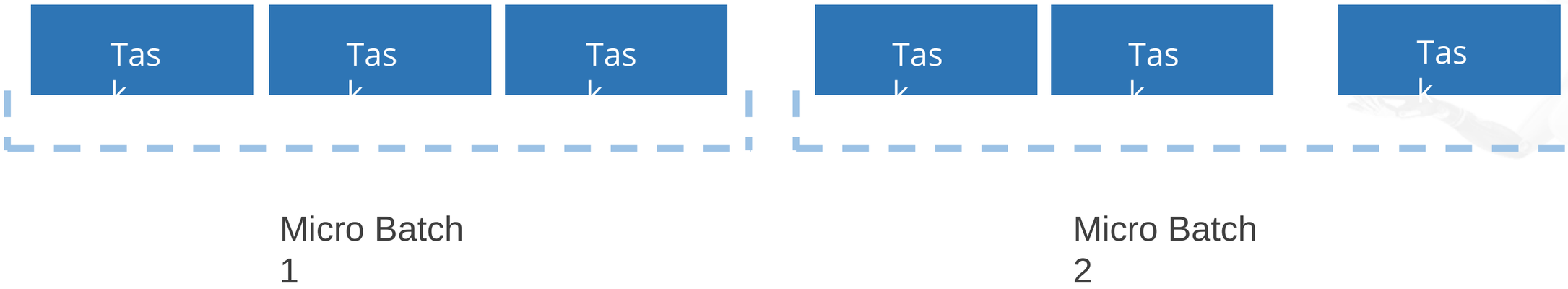
Duration: 15 mins

Problem Statement: In this demonstration, you will learn to write a Spark streaming application.

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**.

Micro Batch

Micro batching handles a stream by a task or process as a sequence that contains data chunks.



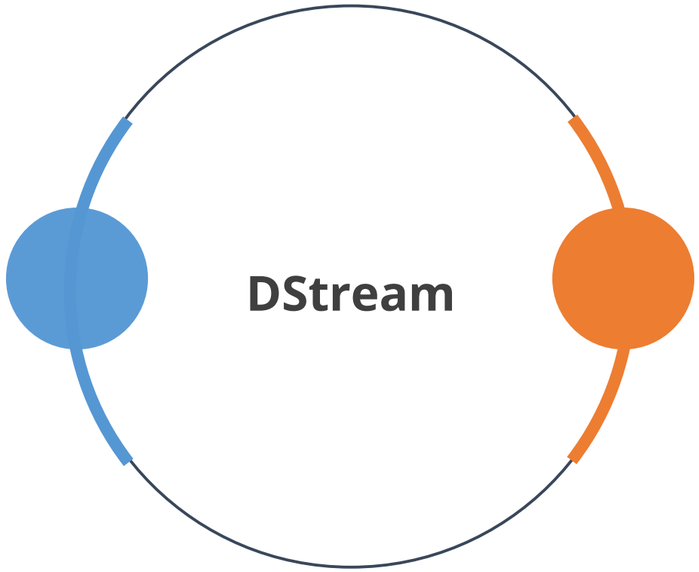
DATA AND
ARTIFICIAL INTELLIGENCE

Introduction to DStreams

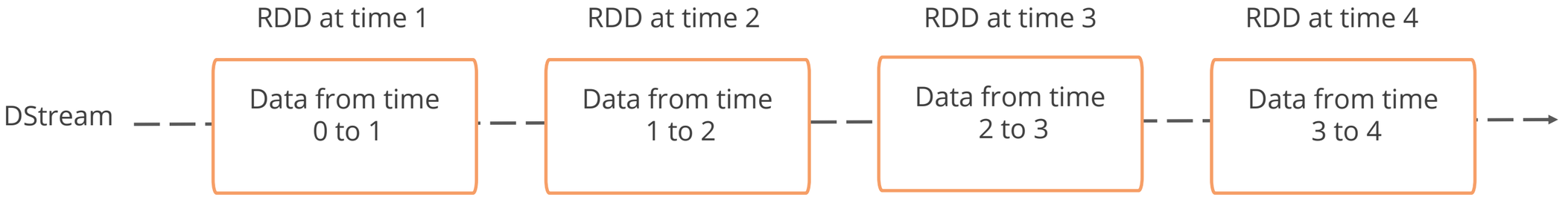
Introduction to DStreams

Discretized Stream (DStream) is the fundamental abstraction available in Spark Streaming.

Characterized by a series of RDDs



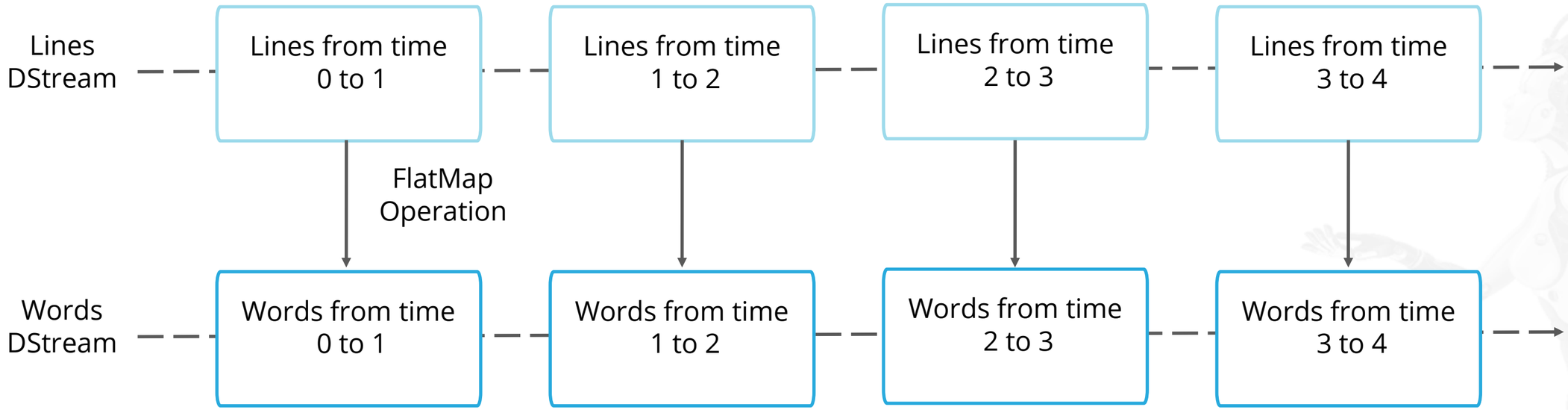
Created by either applying high-level operations on other DStreams or by using input data streams



Series of RDDs

Introduction to DStreams

All operations applied on a DStream get translated to operations applicable on the underlying RDDs.



Input DStreams and Receivers

Input DStreams represent the input data stream received from streaming sources.

Except file stream, each input DStream is linked with a receiver object that stores the data received from a source.

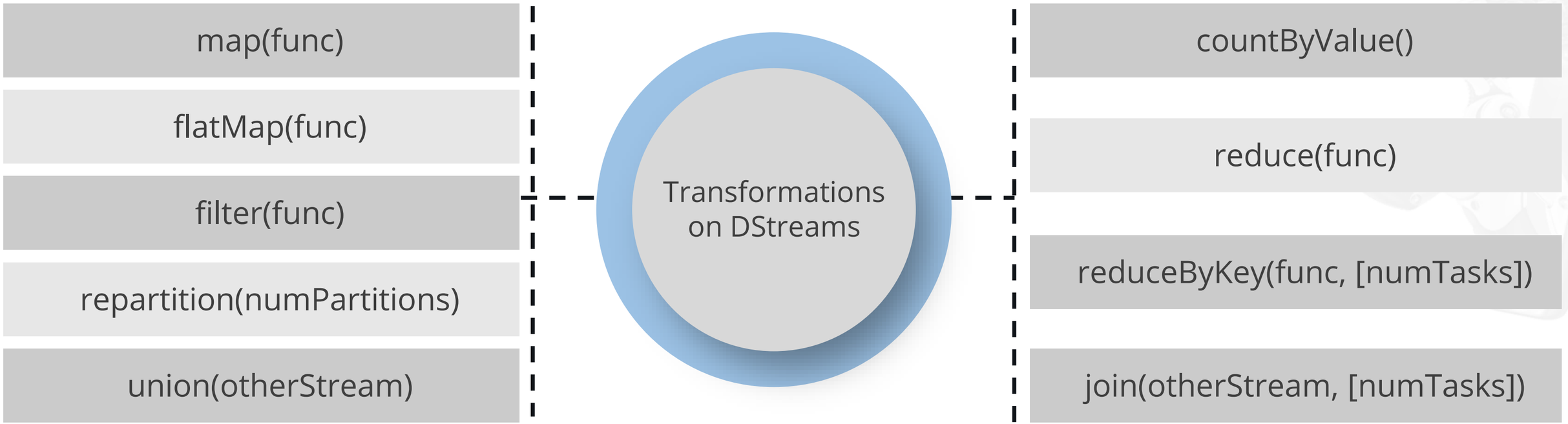
DATA AND
ARTIFICIAL INTELLIGENCE

Transformations on DStreams

Transformations on DStreams

Transformations on DStreams are similar to those of RDDs.

- A few of the common transformations on DStreams are given in the table below:



Output Operations on DStreams

- Output operations let the data of DStreams be pushed to external systems.
- They trigger the real execution of all the DStream transformations.

print()

saveAsTextFiles(prefix, [suffix])

saveAsObjectFiles(prefix, [suffix])

saveAsHadoopFiles(prefix, [suffix])

foreachRDD(func)

Design Patterns for Using ForeachRDD

- DStream.foreachRDD is a powerful primitive that lets the data to be sent to external systems.

Example:

```
DStream.foreachRDD { rdd => val connection = createNewConnection()      //
executed at the driver
rdd.foreach { record => connection.send(record)  // executed at the worker } }
```

DataFrame and SQL Operations

To use DataFrames and SQL operations	Create an SQLContext using the SparkContext that the StreamingContext uses
To allow restarting in case of driver failures	Create a lazily instantiated singleton instance of SQLContext

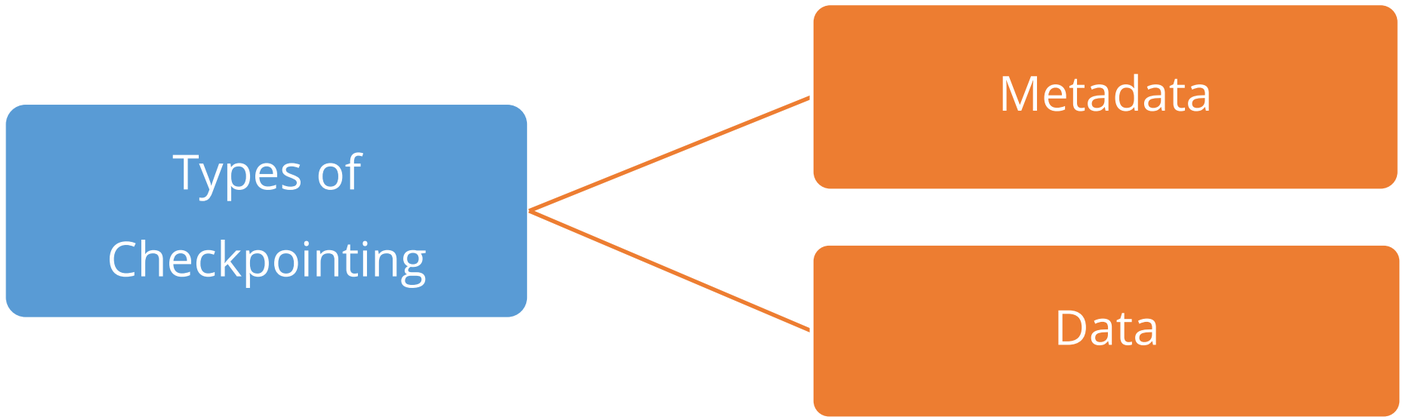
Example:

```
val words: DStream[String] = ...
words.foreachRDD { rdd =>
 // Get the singleton instance of SQLContext
val sqlContext = SQLContext.getOrCreate(rdd.SparkContext)
import sqlContext.implicits._
val wordsDataFrame = rdd.toDF("word")
// Register as table  and Do word count on DataFrame using SQL and print it
wordsDataFrame.registerTempTable("words")
val wordCountsDataFrame =      sqlContext.sql("select word, count(*) as total
from words group by word")
wordCountsDataFrame.show() }
```

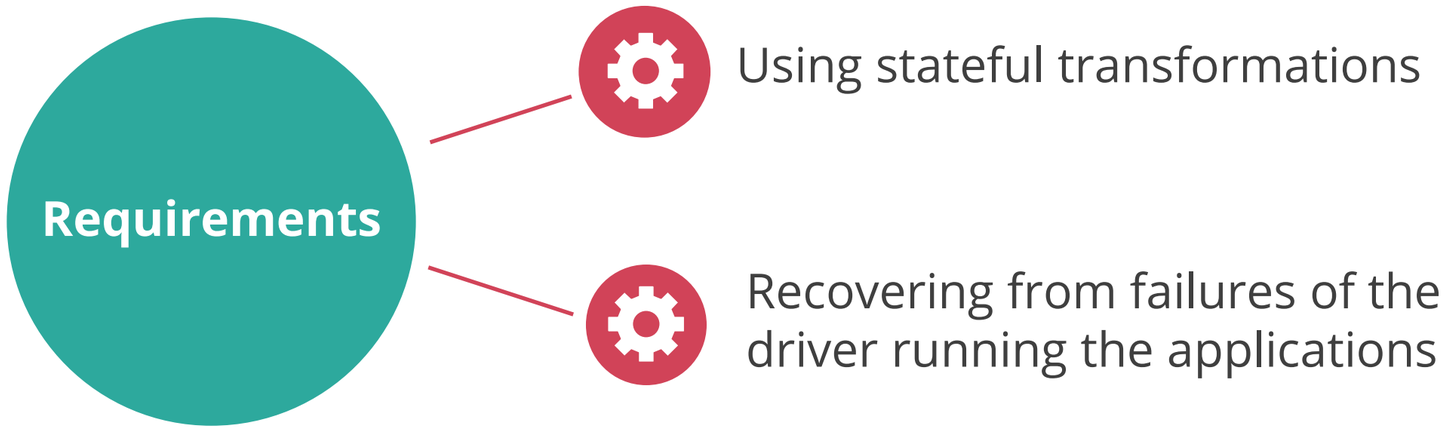
Checkpointing

A streaming application must be:

- Resilient to failures
- Fault-tolerant storage system



Enabling Checkpointing



Socket Stream

- A socket is created on the driver's machine. The code residing outside the closure of the DStream is implemented in the driver, while the rdd.foreach method is implemented on every distributed RDD partition.
- The socket and computation are performed in the same host, which makes it effective.

Example:

```
crowd.foreachRDD(rdd =>
{rdd.collect.foreach(record=>{
out.println(record)
})
})
```

File Stream

- A DStream can be created to read data from files on any file system that is compatible with the HDFS API such as HDFS, S3, and NFS.

Example:

```
streamingContext.fileStream[KeyClass,
ValueClass, InputFormatClass](dataDirectory)
streamingContext.fileStream<KeyClass,
ValueClass, InputFormatClass>(dataDirectory);
streamingContext.textFileStream(dataDirectory)
```

Unassisted Practice

Spark Streaming

Duration: 15 mins

Problem Statement: Perform the below tasks:

- Build a network wordcount program using Spark streaming
- Make sure you run netcat; for example, nc -lk 9999
- Submit the jar file with wordcount program
- Type something on nc and make sure word count is done as part of the job

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**.

Unassisted Practice

Steps to Perform

- **Word count program code**

```
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{ Seconds, StreamingContext }
import StreamingContext._
import org.apache.hadoop.conf._
import org.apache.hadoop.fs._


object Streaming {

def main(args: Array[String]) {

val sparkConf = new SparkConf().setAppName("HdfsWordCount").setMaster("local[4]")

val ssc = new StreamingContext(sparkConf, Seconds(2))
```

Unassisted Practice

Steps to Perform

- **Word count program code**

```
// create the FileInputDStream on the directory and use the stream to count words in newly created files

    val lines = ssc.socketTextStream("localhost", 1234)
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }

}
```

Steps to Perform

- **Spark submit command**

  spark-submit --class "Streaming" scala-spark-training_2.10-1.0.jar

- **Netcat input**

  $ nc -lk 1234
  Hi there, this is John
  I am learning Big data from Simplilearn
  Hi there, this is Alice
  I am learning Apache Spark from Simplilearn

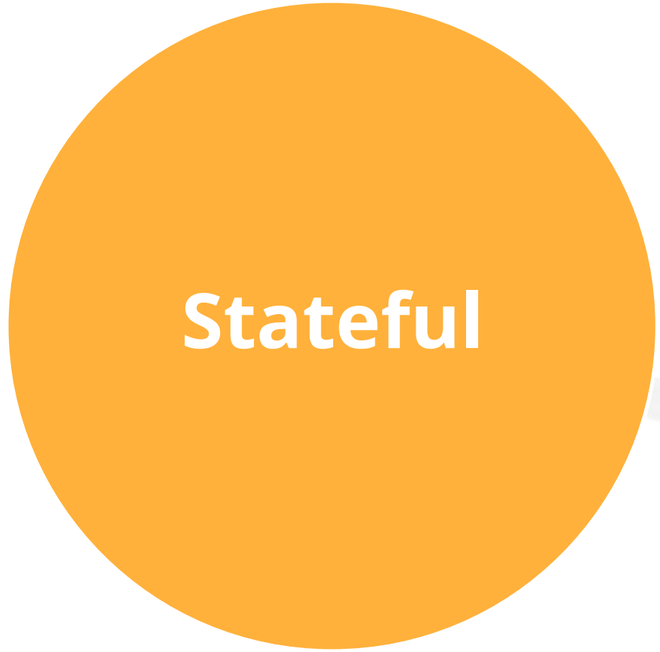
DATA AND
ARTIFICIAL INTELLIGENCE

State Operations

State Management

There are two primary conditions in state management:

When a service is active but is not engaged in processing, it is said to be in a stateless condition



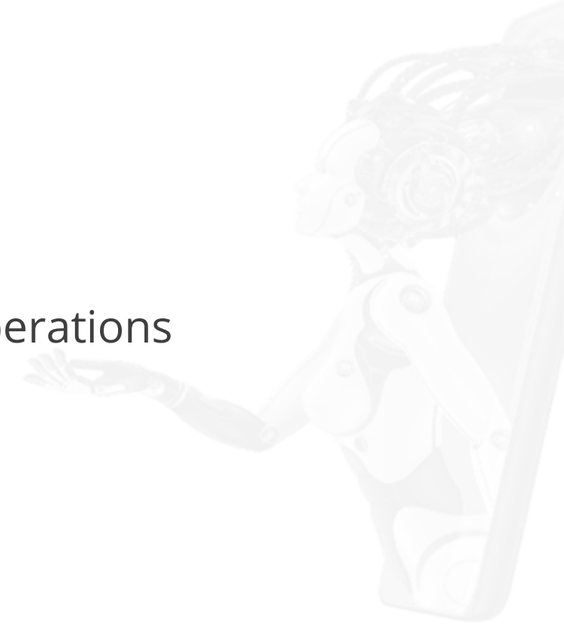
A service that is processing and retaining state data actively is in a stateful condition

Stateful Operations

01 Operate over various data batches

02 Include the updateStateByKey operation and all window-based operations

03 Dependent upon the earlier data batches

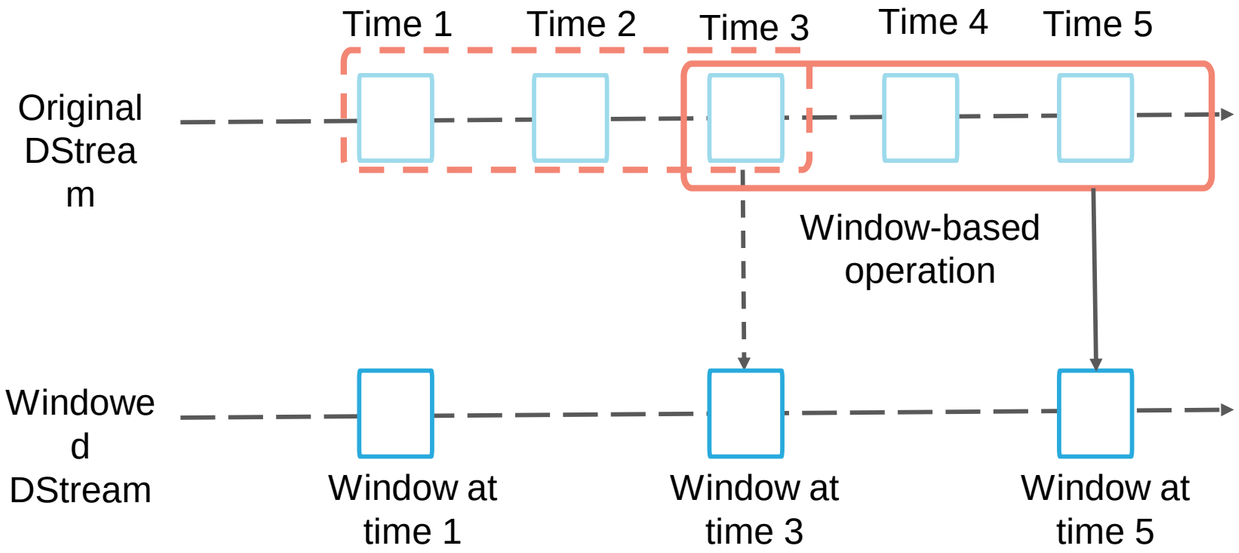


DATA AND
ARTIFICIAL INTELLIGENCE

Windowing Operation

Window Operations

- Window operations let you implement transformations over a sliding window of data.



Example:

```
val windowedWordCounts =

pairs.reduceByKeyAndWindow((a:Int,b:Int)

=> (a + b), Seconds(30), Seconds(10)) })
```

Types of Window Operations

- Operations that take window length and slide interval as parameters are the following:

window(windowLength, slideInterval)
countByWindow(windowLength,slideInterval)
reduceByWindow(func, windowLength,slideInterval)
reduceByKeyAndWindow(func,windowLength, slideInterval, [numTasks])
reduceByKeyAndWindow(func, invFunc,windowLength, slideInterval, [numTasks])
countByValueAndWindow(windowLength,slideInterval,[numTasks])

Join Operations: stream-stream Join

- The first type, stream-stream joins, allows to join streams with other streams.

Example 1:

```
val stream1: DStream[String, String] = ...
val stream2: DStream[String, String] = ...
val joineDStream = stream1.join(stream2)
```

Example 2 (joining over windows of the streams):

```
val windoweDStream1 = stream1.window(Seconds(20))
val windoweDStream2 = stream2.window(Minutes(1))
val joineDStream = windoweDStream1.join(windoweDStream2)
```

Join Operations: stream-dataset Join

- The second type, stream-dataset joins, allows to join a stream and a dataset.

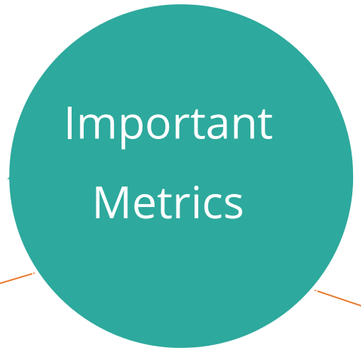
Example:

```
val dataset: RDD[String, String] = ...
val windoweDStream = stream.window(Seconds(20))...
val joineDStream = windoweDStream.transform { rdd => rdd.join(dataset) }
```

Monitoring Spark Streaming Application

- Spark Web UI displays a streaming tab that shows the statistics of the running receivers and details of completed batches.

The important metrics are:



Processing Time

The time that it takes for processing every data batch

Scheduling

The time a batch waits in a queue for the earlier batches to complete processing

- If the batch processing time is continuously above the batch interval or if the queue delay is increasing, reduce the batch processing time.
- Monitor the progress of a Spark Streaming program using the StreamingListener interface.

Assisted Practice

Windowing of Real-Time Data Processing

Duration: 15 mins

Problem Statement: In this demonstration, you will learn windowing of real-time data processing.

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**.

DATA AND
ARTIFICIAL INTELLIGENCE

Spark Streaming Sources

Basic Sources

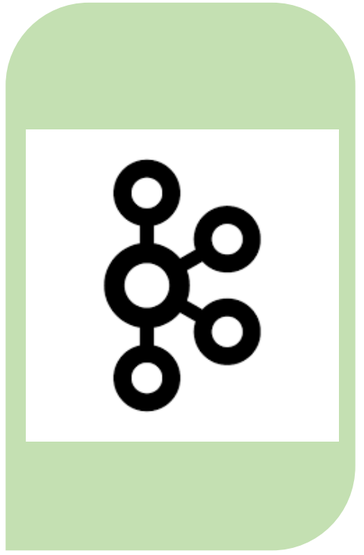
For basic sources, Spark streaming monitors the data directory and processes all files created in it.

Syntax:

```
streamingContext.fileStream[KeyClass, ValueClass, InputFormatClass](dataDirectory)
```

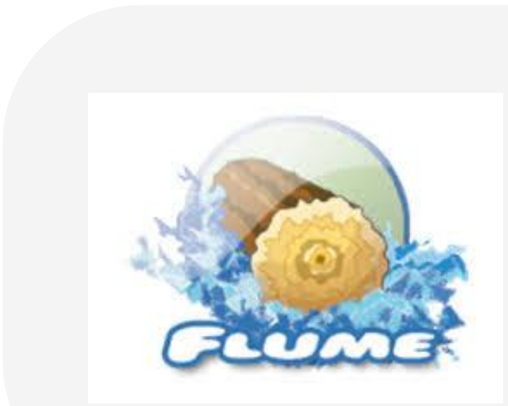
Advanced Sources

Twitter



Apache Kafka

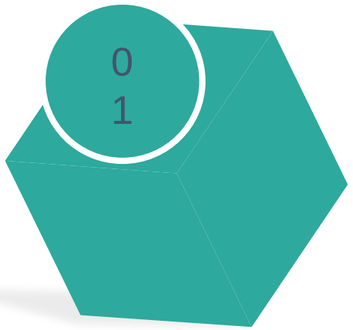
Apache Flume



Kinesis

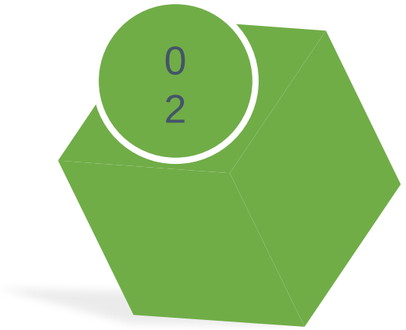
Advanced Sources: Twitter

To create a DStream using data from Twitter's stream of tweets, follow the steps listed below:



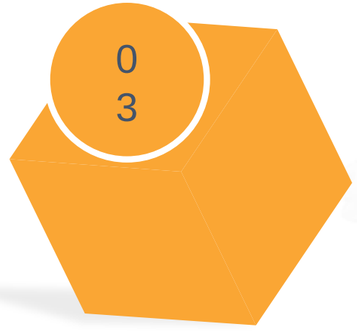
Linking

The artifact Spark-streaming-twitter_2.11 needs to be added to the SBT/Maven project dependencies which is under org.apache.bahir



Programing

The TwitterUtils class needs to be imported and a DStream needs to be created: import org.apache.Spark.streaming.twitter._TwitterUtils.createStream(ssc, None



Deploying

An uber JAR needs to be generated with all dependencies

Assisted Practice

Processing Twitter Streaming Data

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to process Twitter streaming data and perform sentimental analysis.

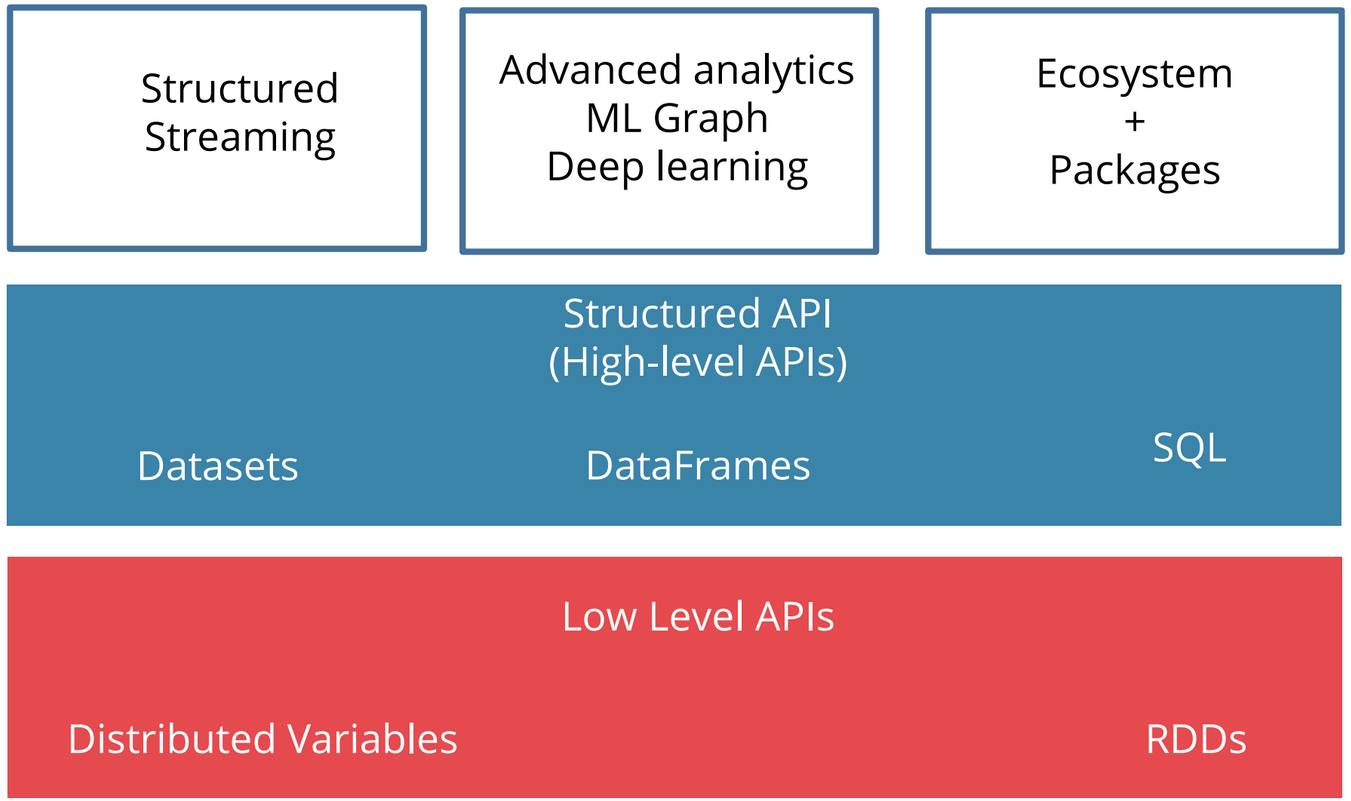
Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

DATA AND
ARTIFICIAL INTELLIGENCE

Structured Spark Streaming

Introduction to Spark Structured Streaming

Structured Streaming is a high-level streaming API that is built on Spark SQL engine.



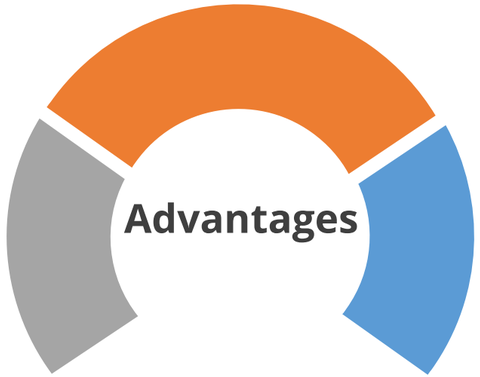
Why Spark Structured Streaming?

Spark Structured Streaming is introduced to overcome the shortcoming of DStreams.

Advantages of Spark Structured Streaming

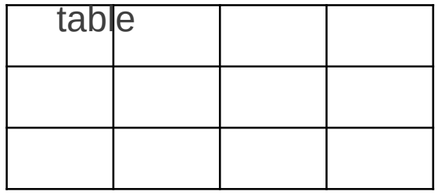
Easy to use

Better performance through SQL optimizations

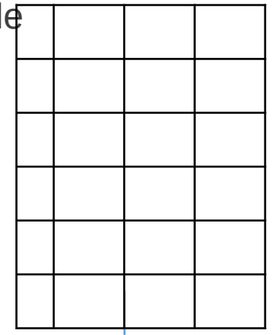


One unified API for both batch and streaming sources

Static bounded table



Streaming unbounded table



The key idea in Structured Streaming is to treat a live data stream as a table that is being continuously appended.

Batch vs. Streaming

- Sample code

The new Structured Streaming API enables you to easily adapt the batch jobs that you have already written to deal with a stream of data.

Batch

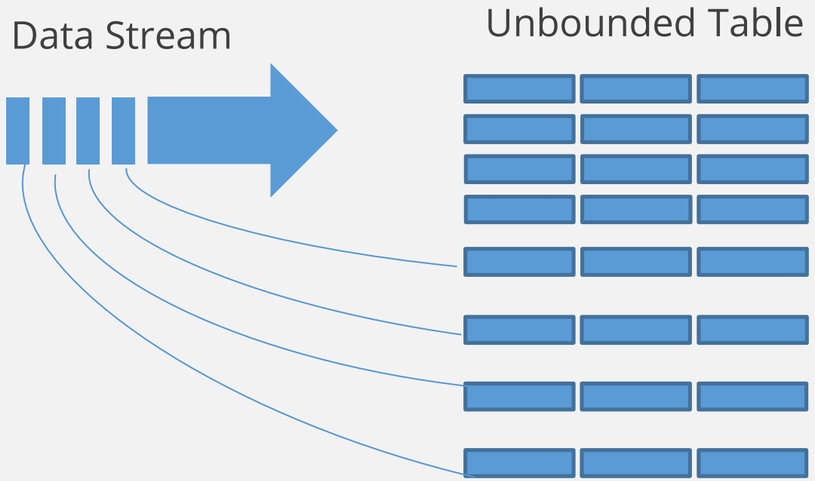
```
val callSchema = new
StructType().add(“Name”,”String").add("City","
String").add("Country","String").add("CallTs",
"String").add("CallCharge","integer")
val personDF =
Spark.read.schema(callSchema).json(“/sampledat
a/json/call.json")
val peopleParis =
personDF.select("City","Callcharge").where("Ci
ty = 'Paris'").where("CallCharge > 500")
peopleParis.write.format("parquet").save("/sam
pledata/streaming/output")
```

Continuous Stream

```
val personSchema = new
StructType().add(“Name”,”String”).add("City","
String").add("Country","String").add("CallTs",
"String").add("CallCharge","integer")
val personDF =
Spark.reaDStream.schema(callSchema).json(“/sam
pledata/json/call.json")
val peopleParis =
personDF.select("City","Callcharge").where("Ci
ty = 'Paris'").where("CallCharge > 500")
peopleParis.write.format("parquet").save("/sam
pledata/streaming/output")
```

Use Case: Banking Transactions

- **Scenario**: Banking transaction records containing the account number and transaction amount are coming in a stream.



New data in the data stream
=
New rows appended to an unbounded table

Advanatges

- Allows whatever data processing that is possible using a DataFrame to be possible with the stream data as well
- Reduces the burden on application developers
- Allows them to focus on the business logic of the application rather than the infrastructure related aspects

Spark Streaming vs. Spark Structured Streaming

Spark Streaming

- Dstream only uses RDD API
- Tracking of state between batch times for cumulative statistics is complex in DStream
- No guarantee of data integrity
- The API works only with batch

Spark Structured Streaming

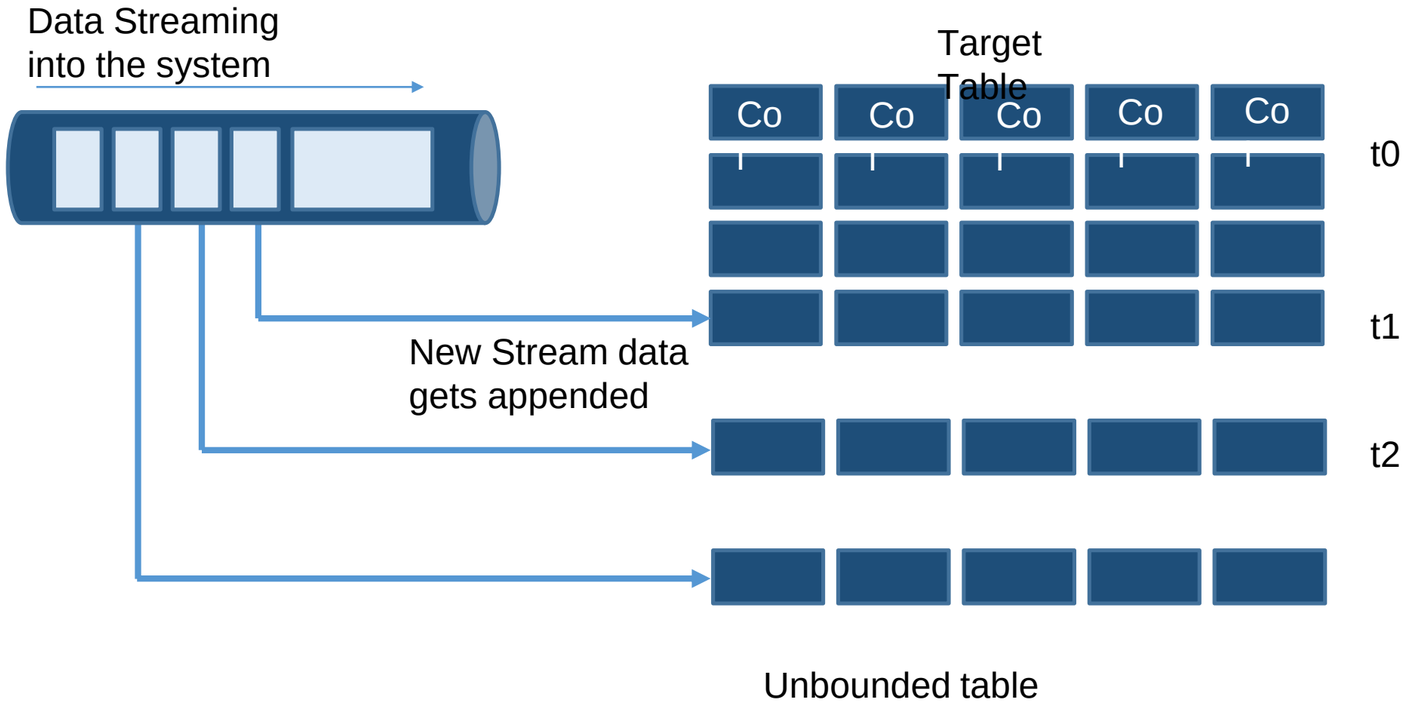
- Uses DataFrame/Dataset API
- User just needs to take care of business logic
- Automatically handles consistency and reliability
- The API is same so you can write to the same data destination and can also read it back.

DATA AND
ARTIFICIAL INTELLIGENCE

Structured Streaming Architecture, Model, and Its Components

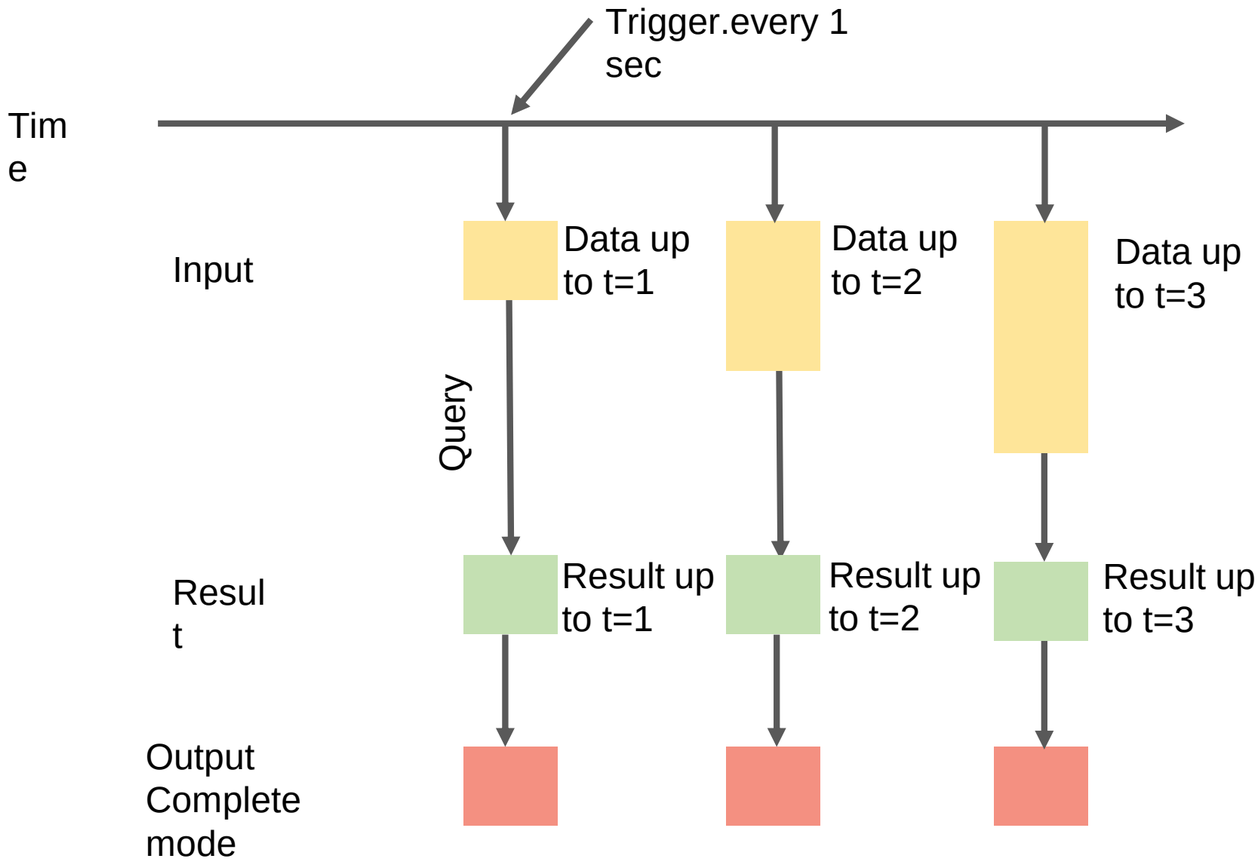
Structured Streaming Architecture

- Structured Streaming treats all the arriving data as an **unbounded input table**.
- Every time there is a new item in the stream, it gets appended as a row in the input table at the bottom.

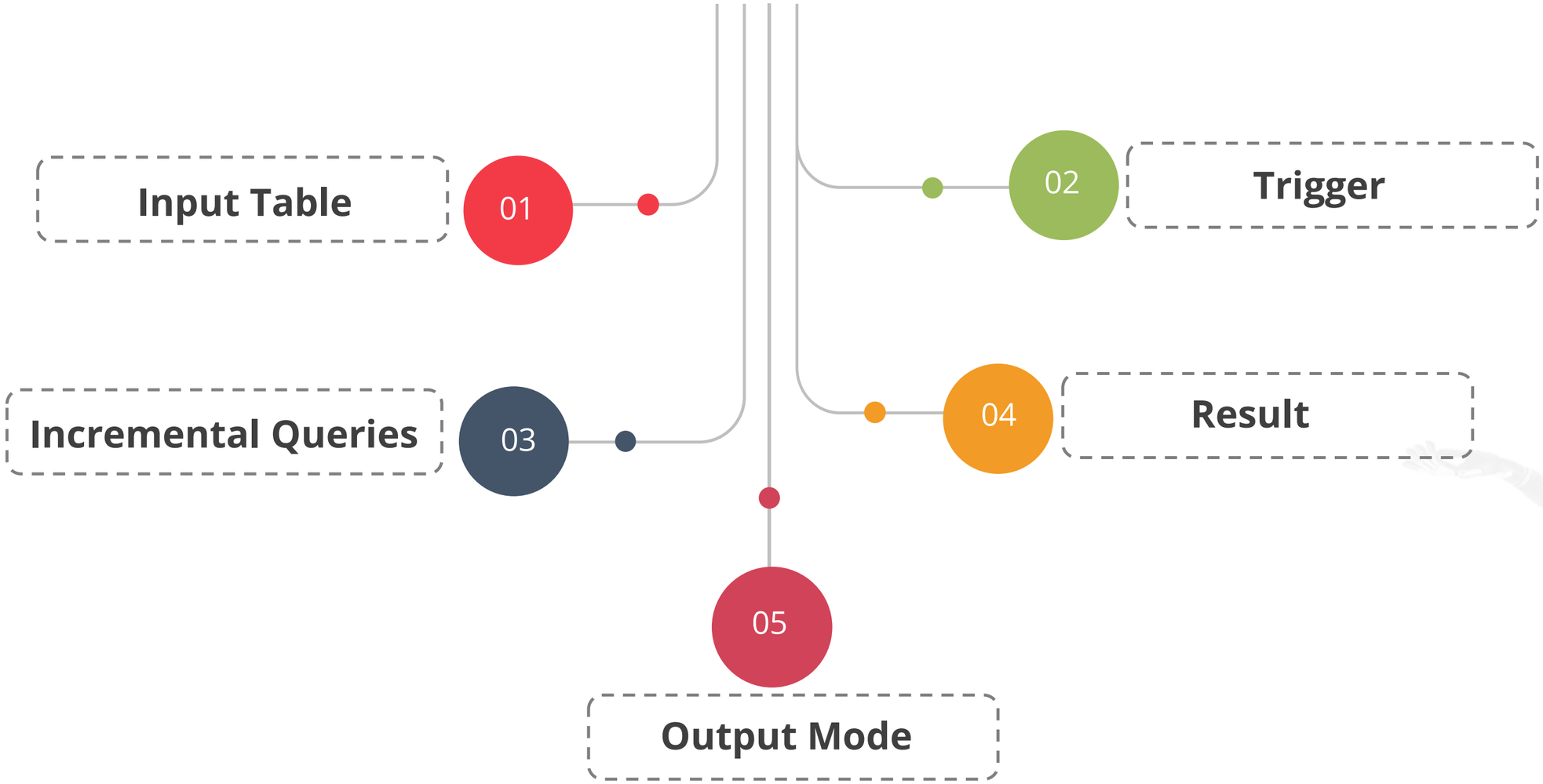


Structured Streaming Model

Incremental execution on streaming data

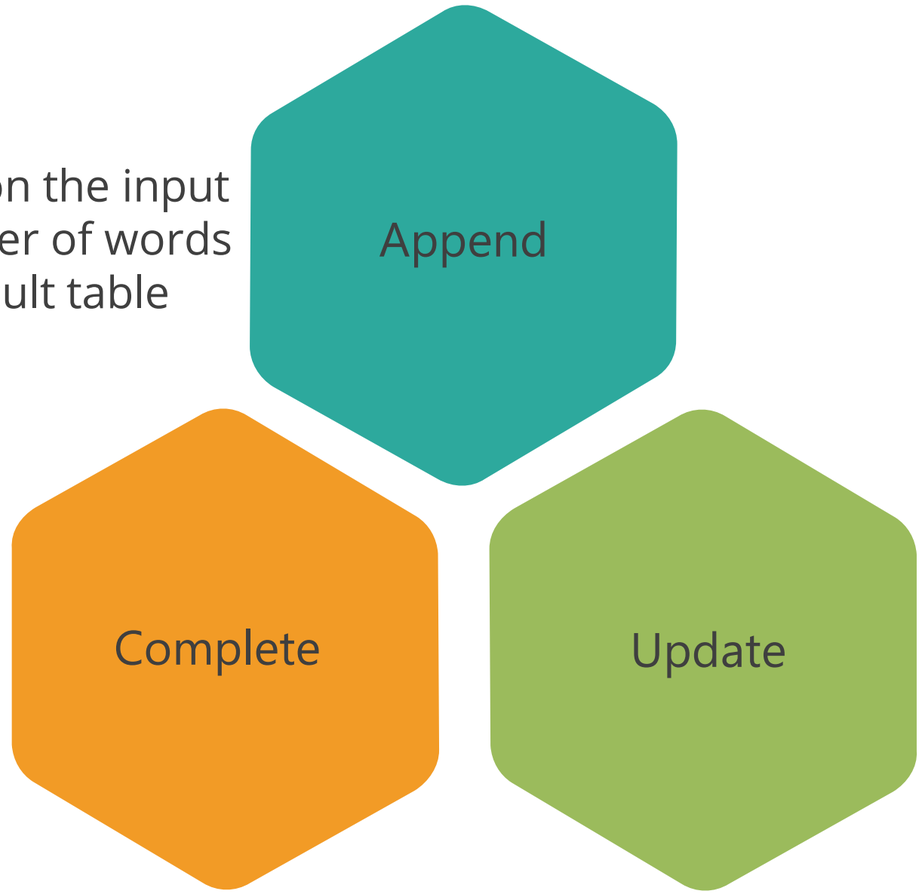


Components of Structured Streaming Model



Output Modes

A query defined by the developer on the input table and count to count the number of words that in turn computes a final result table



A query defined by the developer on the input table and count to count the number of words that in turn computes a final result table

A query defined by the developer on the input table and count to count the number of words that in turn computes a final result table

Output Sinks

File sink: Stores the output to a directory

```
writeStream
  .format("parquet") // can be "orc", "json", "csv", etc.
  .option("path", "path/to/destination/dir")
  .start()
```

Foreach sink: Runs arbitrary computation on the records in the output

```
writeStream
.foreach(...)
.start()
```

Output Sinks

Console sink (for debugging):

```
writeStream
.format("console")
.start()
```

Memory sink (for debugging):

```
writeStream
.format("memory")
.queryName("tableName")
.start()
```

DATA AND
ARTIFICIAL INTELLIGENCE

Structured Streaming APIs

Features of Structured Streaming APIs

Structured Streaming offers a high-level declarative streaming API built on top of datasets and dataframes.

Example

```
val socketDF = Spark
  .reaDStream
  .format(“socket")// Reading Data from socket(Socket Datasource)
  .option("host", "localhost")
  .option("port", 9999)
  .load()
```

Data Sources

File source

Socket source (for testing)

Kafka source

Example

```
val inputDF = Spark.reaDStream.json("s3://logs")
```

Reading data from JSON file

Operations on Streaming DataFrames and Datasets

- With Structured Streaming API, we can perform:
  - Untyped, SQL-like operations
  - Typed RDD-like operations

Example

```
// Select the persons which have age more than 60
df.select("name").where("age > 60") // using untyped APIs
ds.filter(_.age > 60).map(_.name)   // using typed APIs
// Running count of the number of counts for each value
df.groupBy("value").count()         // using untyped API
```

Parsing Data with Schema Inference

Code to read a CSV file using schema inference

Example

```
import org.apache.Spark.sql.types._
import org.apache.Spark.sql.catalyst.ScalaReflection
import org.apache.Spark.sql.functions._
case class Employee(
name:String,
city:String,
country:String,
age:Option[Int]
)
//Step 1:-Create schema for parsing data
val caseSchema =
(ScalaReflection.schemaFor[Employee].dataType.asInstanceOf[StructType])
//Step2:-Schema is passed to the stream
val empStream =
(Spark.reaDStream.schema(caseSchema).option("header",true).option("maxfilespertrigger
",1).csv("data/people.*").as[Employee])
//Step 3:Write the results to the screen
(empStream.writeStream.outputMode("append").format("console").start)
```

Constructing Columns in Structured Streaming

- Structured Streaming makes use of column objects for manipulating data
- A column can also be constructed from other columns using binary operator

Example

```
(empStream.select($"country" === "France" as "in_France", $"age" <35 as “under_35”,
'country startsWith "U" as
"U_country").writeStream.outputMode("append").format("console").start)
```

groupby and Aggregation

- Spark structured stream uses a groupby operator
- You can then perform different aggregation operations to this group

Example

```
(empStream.groupBy(‘country).mean(“age”).writeStream.outputMode("complete").
format("console").start)
```

- For complex aggregations, “agg” function is used

Example

```
(empStream.groupBy(‘country).agg(first("country") as “country",
count  ( "age")) .writeStream.outputMode("complete").format("console").start)
```

Joining Structured Stream with Datasets

Streaming DataFrames can be joined with static DataFrames to create a new streaming DataFrame

Example

```
val staticDf = Spark.read. ...
val streamingDf = Spark.reaDStream. ...

streamingDf.join(staticDf, "type")  // inner equi-join with a static DF
streamingDf.join(staticDf, "type", "right_join")  // right outer join with a static
DF
```

SQL Query in Spark Structured Streaming

- Writing SQL queries directly on stream:

1. **Create a temporary Table**

```
empStream.createOrReplaceTempView(“empTable")
```

2. **Write a SQL query on created temp table**

```
val query = Spark.sql("Select country,avg(age) from empTable group by country")
```

3. **Write the results to the console**

```
(query.writeStream.outputMode(“complete").format("console").start)
```

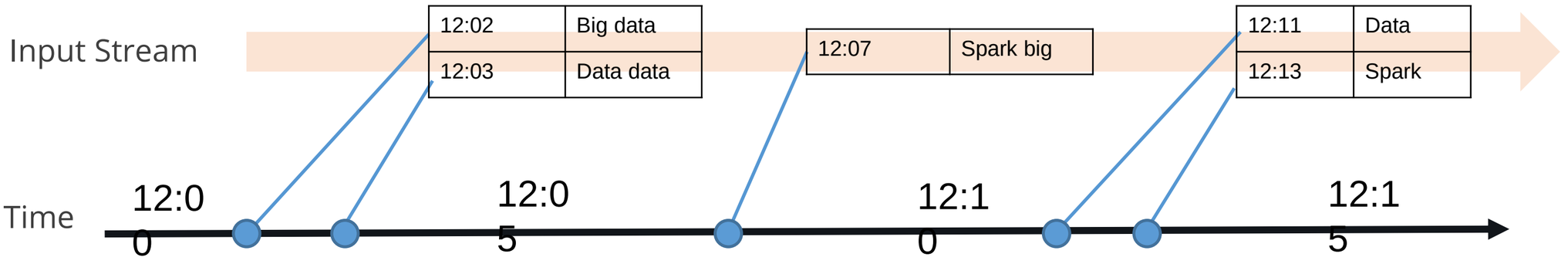
Windowed Operations on Event-Time

- Windowed operations are running aggregations over data bucketed by time windows.

WordCount Example

```
// Split the lines into words, retaining timestamps
 val words = lines.as[(String, Timestamp)].flatMap(line =>line._1.split(" ").map(word =>
(word, line._2))).toDF("word", "timestamp")
// Group the data by window and word and compute the count of each group
  val windowedCounts = words.groupBy(window($"timestamp", windowDuration, slideDuration),
     $"word").count().orderBy("window")
```

Windowed Grouped Aggregations



Resulting tables after 5 minutes of triggers

12:00-12:10	big	1
12:00-12:10	data	3

12:00-12:10	Big	2
12:00-12:10	Data	3
12:00-12:10	Spark	1
12:05-12:15	Spark	1
12:05-12:15	Big	1

12:00-12:10	Big	2
12:00-12:10	Data	3
12:00-12:10	Spark	1
12:05-12:15	Big	1
12:05-12:15	Spark	2
12:05-12:15	Data	1
12:05-12:20	Data	1
12:05-12:20	Spark	1

Windowed Grouped Aggregation with 10 minutes windows, sliding every 5 minutes

Counts incremented for windows 12:00 -12:10 and 12:05 -12:15

Counts incremented for windows 12:05 -12:15 and 12:10 -12:1220

With Structured Streaming, expressing such windows on event-time is simply performing a special grouping using the window() function.

Failure Recovery And Checkpointing

- Structured Streaming allows recovery from failures, which is achieved by using checkpointing and WAL.
- Configure a query with a checkpoint location, and the query will save all the progress information and the running aggregates for the checkpoint location.
- The checkpoint location should be a path in an HDFS compatible file system, and can be set as an option in the DataStreamWriter when starting a query.

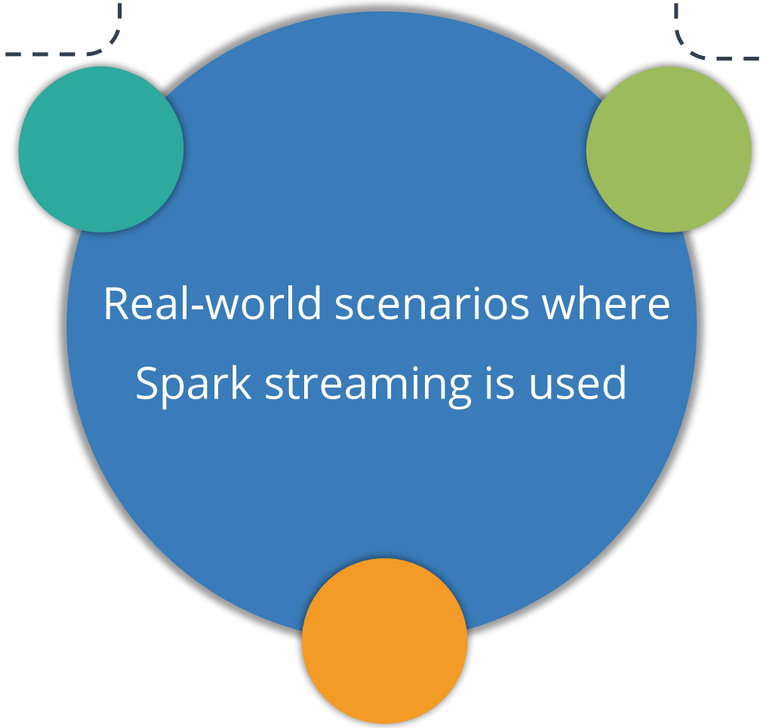
Example

```
callsFromParis.writeStream.
format(“parquet").
option(“checkpointlocation","hdfs://nn:8020/mycheckloc").
start("/home/Spark/streaming/output")
```

Use Cases

Analyze streams of real-time sensor data

Analyze streams of web log data



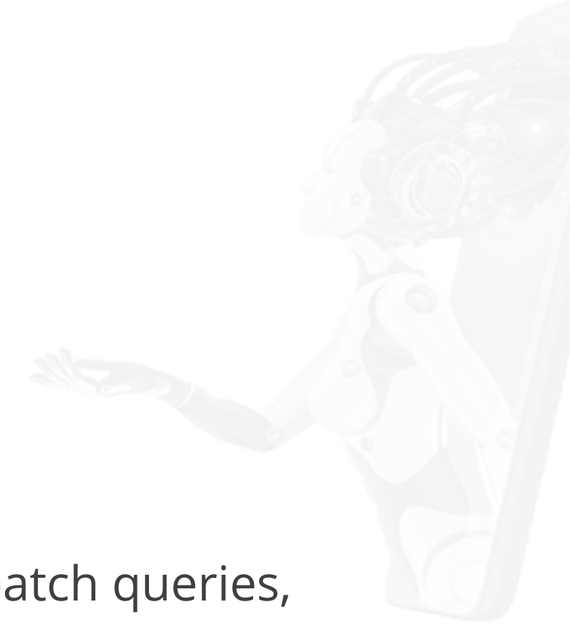
Streaming ETL pipelines prepare raw, unstructured data into a form that can be queried easily and efficiently

When to Use Structured Streaming

01 To create a streaming application

02 To provide data consistencies and exactly-once semantics

03 To create continuous applications that are integrated with batch queries, streaming, and machine learning



Assisted Practice

Streaming Pipeline

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to create a streaming pipeline.

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:

- Explain concepts of Spark Streaming
- Understand the Lambda and Kappa architecture
- Explain the concepts of Spark Structured Streaming
- Explain Join and Window operations

Quiz

Knowledge Check

1

_____ attempts to balance high-throughput MapReduce frameworks with low-latency real-time processing.

a. Lambda architecture

b. Kappa architecture

c. Both A and B

d. None of the above

Knowledge Check

1

_____ attempts to balance high-throughput MapReduce frameworks with low-latency Real-time processing.

a. Lambda architecture

b. Kappa architecture

c. Both A and B

d. None of the above

The correct answer is **a.**

Lambda Architecture (LA) attempts to balance high-throughput MapReduce frameworks with low-latency **r**eal-time processing.

Knowledge Check

2

Which of the following data sources is supported in Spark Streaming?

a. Twitter

b. Kafka

c. Flume

d. All of the above

Knowledge Check

2

Which of the following data sources is supported in Spark Streaming?

a. Twitter

b. Kafka

c. Flume

d. All of the above

The correct answer is **d.**

Spark Streaming supports all these advance data sources.

Knowledge Check

3

Which of the following statements is true about the saveAsObjectFile method?

a. Saves a DStream's contents as a SequenceFile of serialized Java objects

b. Saves a DStream's contents as text files

c. Saves a DStream's contents as an Avro file in Hadoop

d. Applies a function, func, to each RDD generated from the stream

Knowledge Check

3

Which of the following statements is true about the saveAsObjectFile method?

a. Saves a DStream's contents as a SequenceFile of serialized Java objects

b. Saves a DStream's contents as text files

c. Saves a DStream's contents as an Avro file in Hadoop

d. Applies a function, func, to each RDD generated from the stream

The correct answer is **a.**

The saveAsObjectFile method saves a DStream's contents as a SequenceFile of serialized Java objects.

Knowledge Check

4

When should you use Structured Streaming?

a. To create a streaming application using Dataset and DataFrame APIs

b. When providing data consistencies and exactly-once semantics even in case of delays and failures at multiple levels

c. For creating continuous applications that are integrated with batch queries, streaming, and machine learning

d. All of the above

Knowledge Check

4

When should you use Structured Streaming?

a. To create a streaming application using Dataset and DataFrame APIs

b. When providing data consistencies and exactly-once semantics even in case of delays and failures at multiple levels

c. For creating continuous applications that are integrated with batch queries, streaming, and machine learning

d. All of the above

The correct answer is **d.**

You can use Structured Streaming for all the mentioned cases.

Lesson-End Project

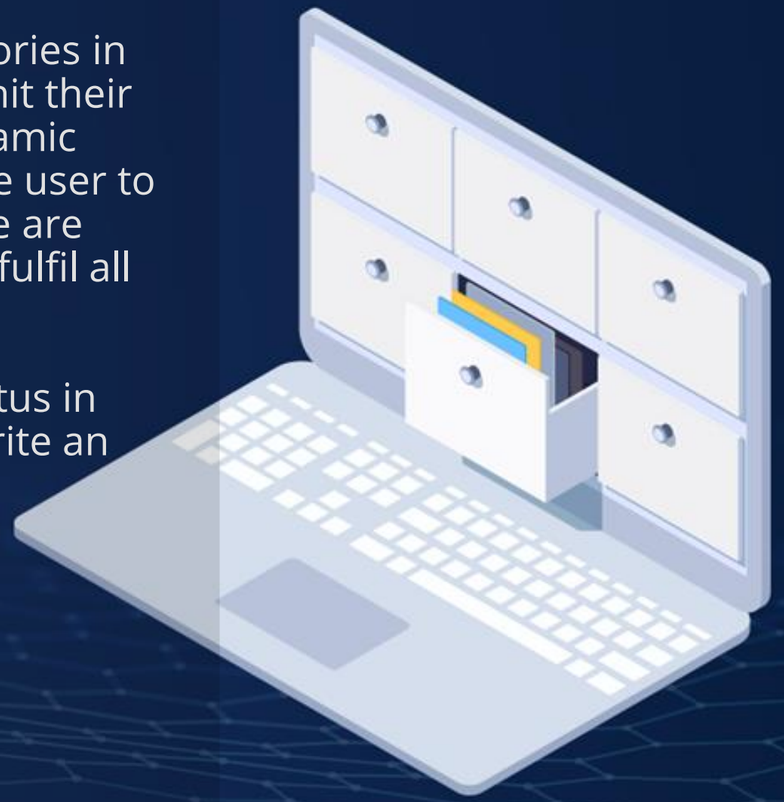
Problem Statement:

Alibaba is an e-commerce website that sells products online across different categories in different countries. Festivals are coming up and they want to make sure that they hit their target revenue. To make sure of that, they have decided to provide a trending dynamic banner where users can see the trending categories and brands which can help the user to decide which brand's product to purchase and to see from which categories people are purchasing the most. This will also help the company to keep enough inventory to fulfil all orders.

Currently, their system is using Hadoop MR which is not providing the trending status in real-time. They hired you as a big data engineer to modify the existing code and write an optimized code that will work for any time duration.
For example, trending brands in the last 5 minutes.

You have been given transactions.csv file which contains the below fields:

1. Product Code
2. Description
3. Brand
4. Category
5. Sub Category



Lesson-End-Project

You have to write Spark Streaming jobs to find out:

1. The top 5 trending categories in the last 5 minutes which has the maximum number of orders
2. The bottom 5 brands in the last 10 minutes which has the least number of orders
3. Product units sold in the last 10 minutes

Note:

You have to send the above data from CSV to Kafka.
Make sure you have some delay while sending the data as real-time scenario orders take time and they are pushed whenever the end-user purchases something.
Also, avoid pushing all the data at once, in Kafka.

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

Big Data Hadoop and Spark Developer

simplilearn

Spark GraphX

Learning Objectives

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

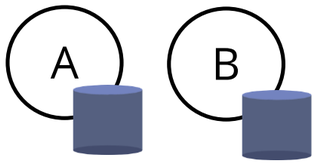
- Define graph and identify the types of graph
- Describe GraphX in Spark
- Identify different operators in GraphX
- Examine PageRank algorithm with social media data

DATA AND
ARTIFICIAL INTELLIGENCE

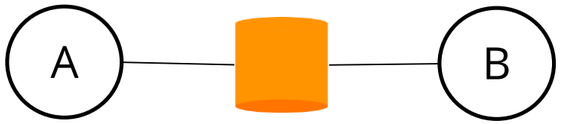
Introduction to Graph

What Is a Graph?

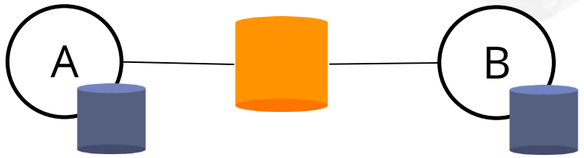
> A graph is a structure which results to a set of objects which are related to each other. The relation between them is represented using edges and vertices.



Vertices



Edges



Triplets

Use Cases of Graph Computation

Fraud Detection System

Page Rank

Disaster Detection System

Business Analysis

Geographic Information System

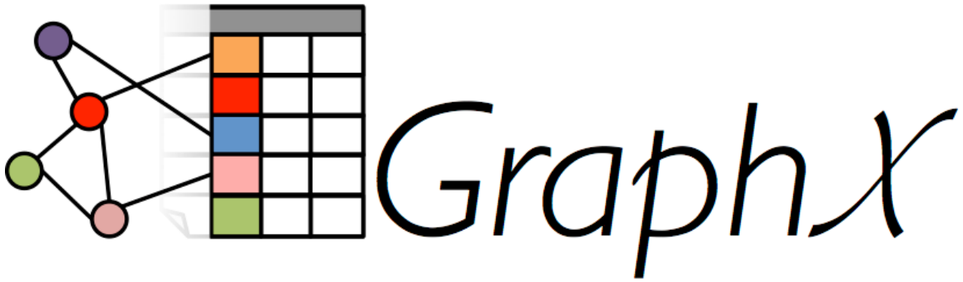
Google Pregel

Types of Graphs

DATA AND
ARTIFICIAL INTELLIGENCE

GraphX in Spark

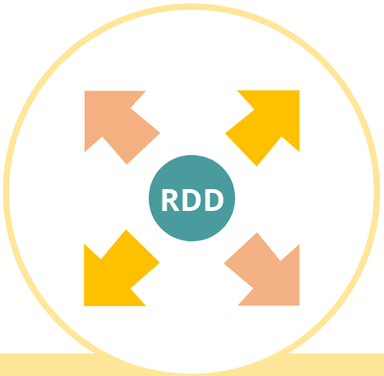
Spark GraphX



GraphX is a graph computation system that runs on data-parallel system framework. It is a new component in Spark for graphs and graph-parallel computation.

Features of Spark GraphX

GraphX is more of a real-time processing framework.



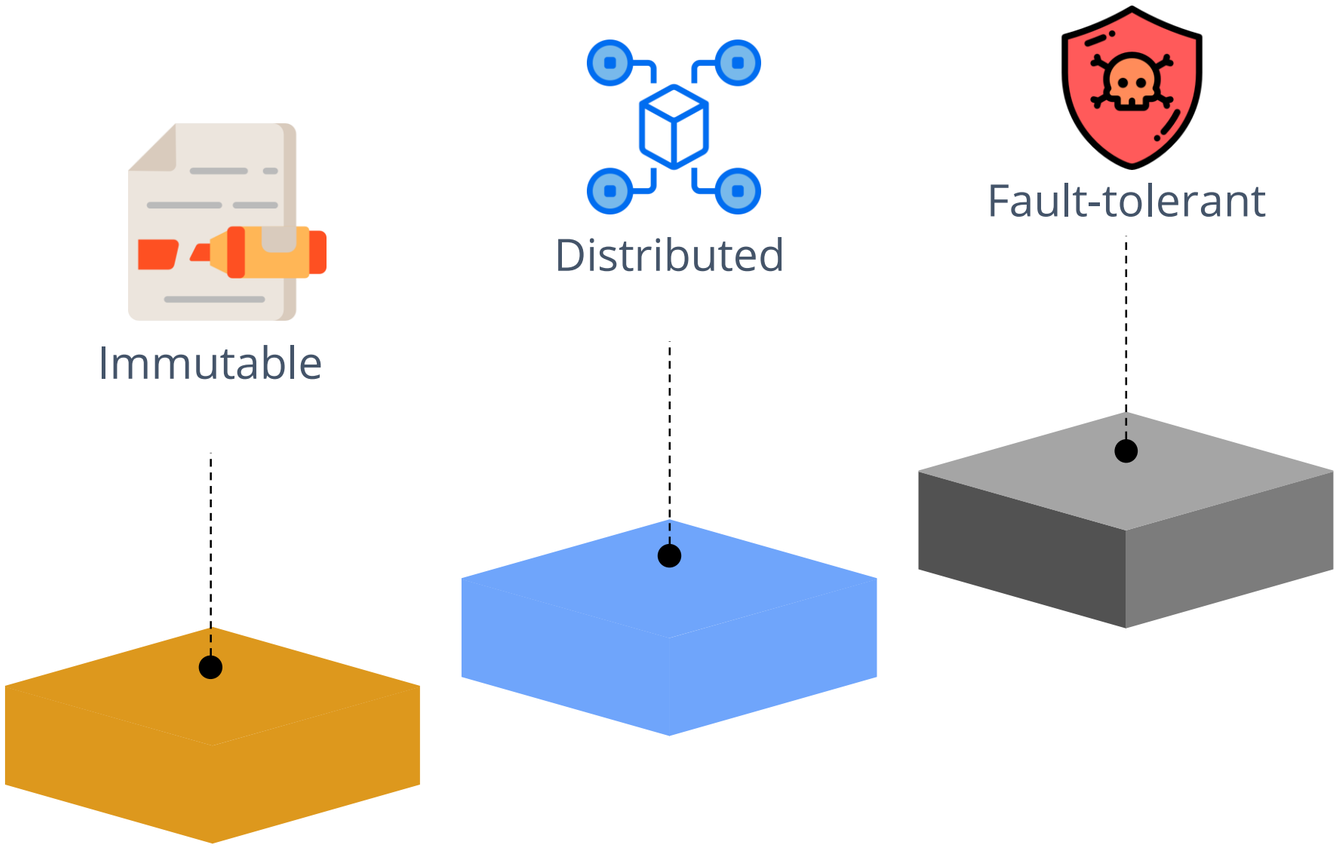
GraphX extends the RDD abstraction and introduces RDG.

GraphX simplifies the graph ETL and analysis process substantially.

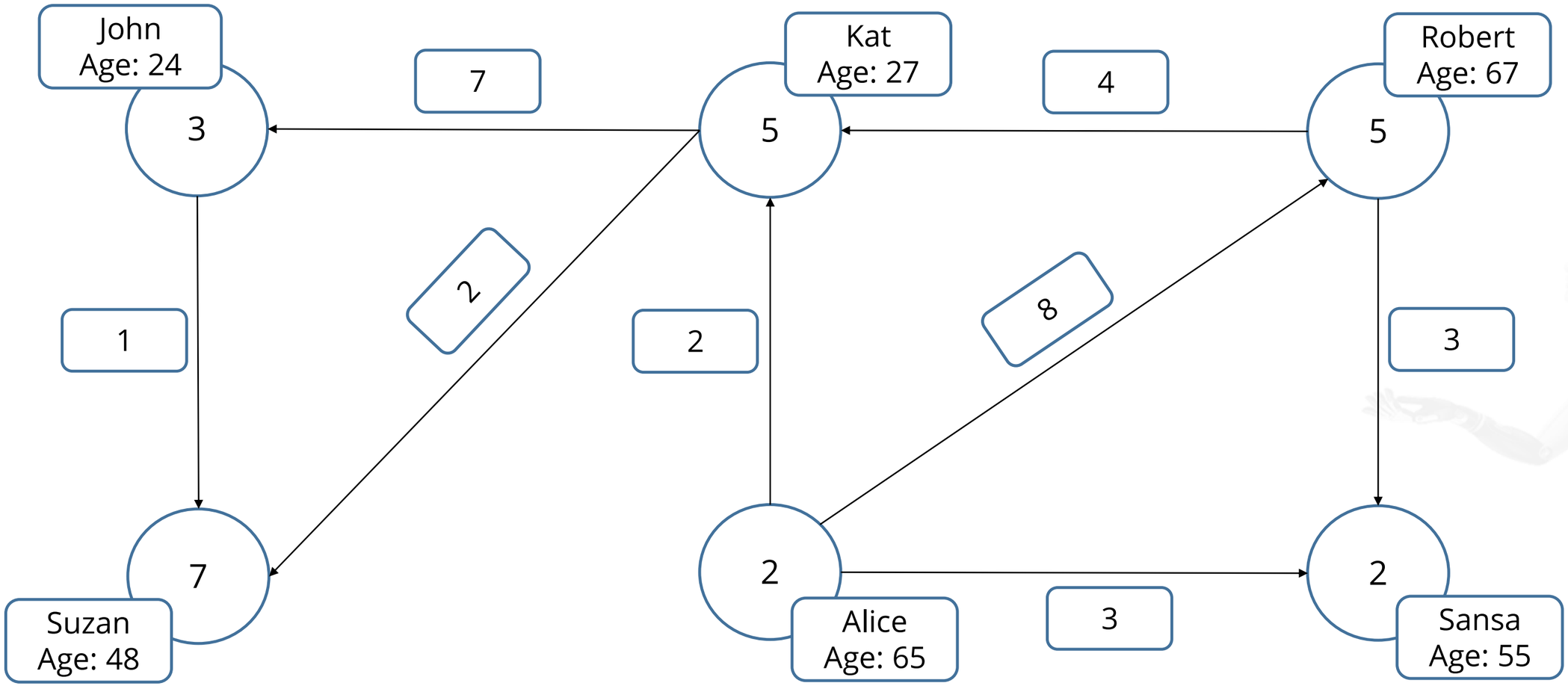
Property Graph

A directed multigraph is a directed graph with potentially multiple parallel edges sharing the same source and destination vertex.

The following are the characteristics of property graph:



GraphX: Example



Implementation of GraphX

Importing classes:

```
//Importing the necessary classes
import org.apache.spark._
import org.apache.spark.rdd.RDD
import org.apache.spark.util.IntParam
import org.apache.spark.graphx._
import org.apache.spark.graphx.util.GraphGenerators
```

Displaying names and edges:

```
val vertexRDD: RDD[(Long, (String, Int))] = sc.parallelize(vertexArray)
val edgeRDD: RDD[Edge[Int]] = sc.parallelize(edgeArray)
val graph: Graph[(String, Int), Int] = Graph(vertexRDD, edgeRDD)
graph.vertices.filter { case (id, (name, age)) => age > 30 }
.collect.foreach { case (id, (name, age)) => println(s"$name is $age")}
```

DATA AND
ARTIFICIAL INTELLIGENCE

Graph Operators

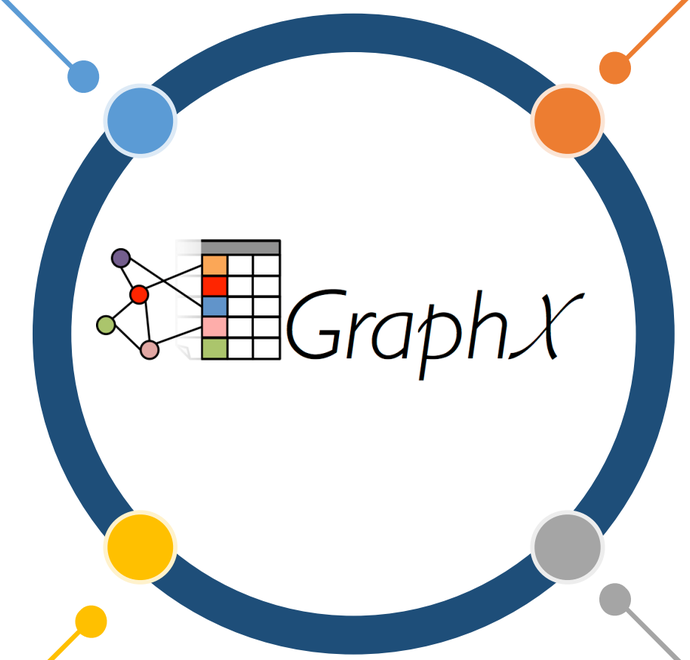
Graph Operators

Property graphs have a collection of basic operators.
These operators take user defined functions and produce new graphs.

Example:

```
val graph: Graph[(String, String), String]
// Use the GraphOps.inDegrees operator
val inDegrees: VertexRDD[Int] = graph.inDegrees
```

Types of Graph Operators



Property Operator

Structural Operator

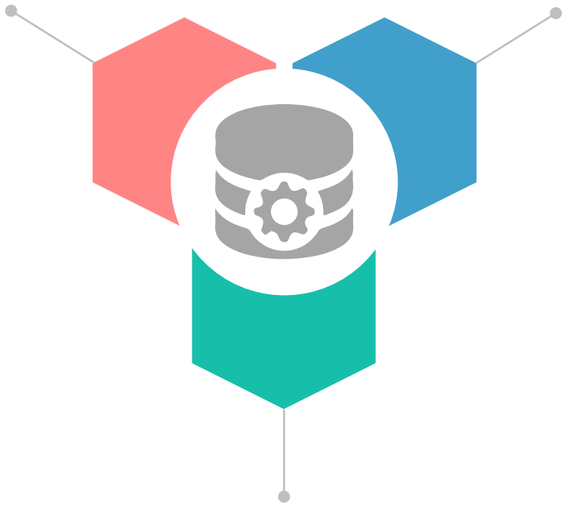
Neighborhood Aggregation

Join Operator

Property Operator

The property operator contains the following operations:

mapVertices



mapEdges

mapTriplets

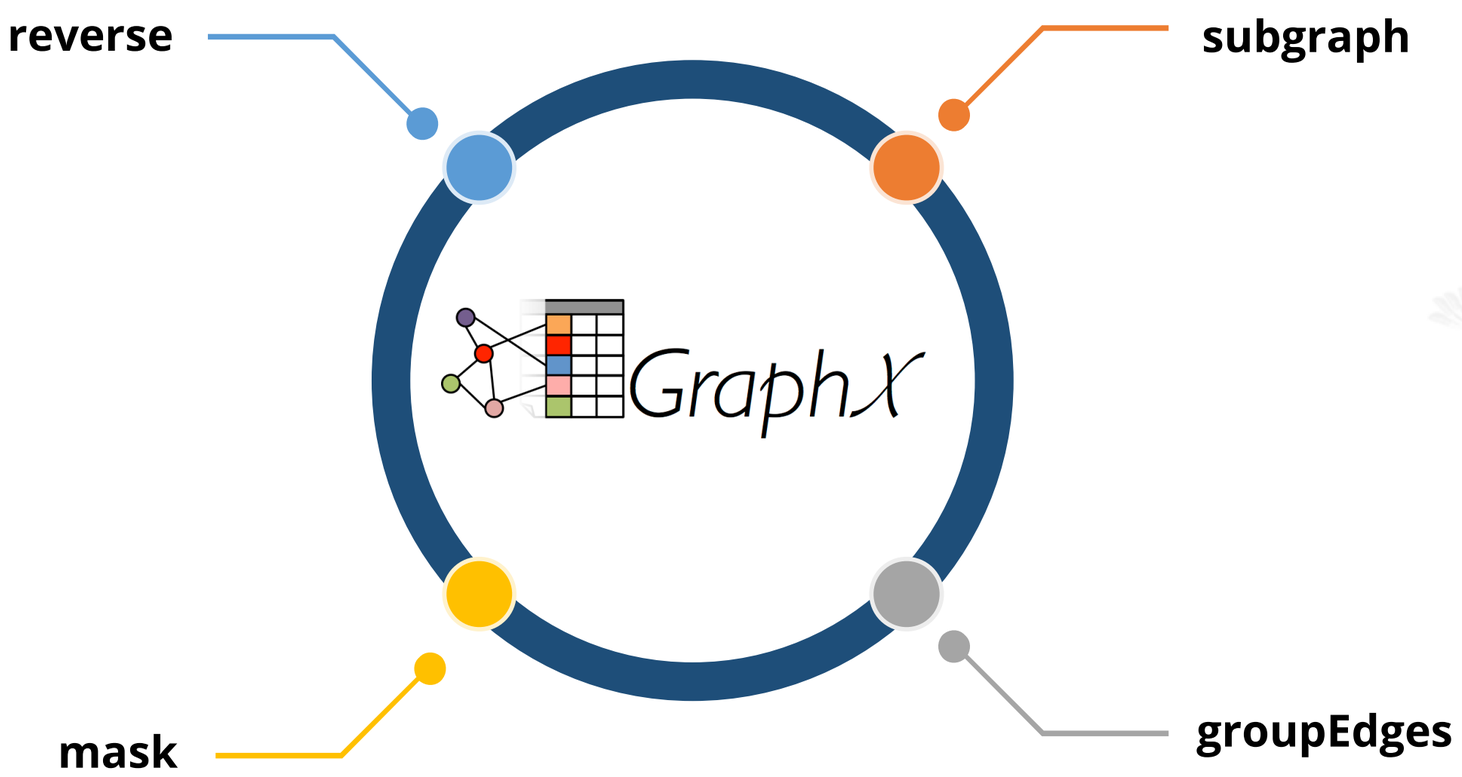
Property Operator

The following are the syntax of property operators:

```
class Graph[VD, ED] {
  def mapVertices[VD2](map: (VertexId, VD) => VD2): Graph[VD2,
ED]
  def mapEdges[ED2](map: Edge[ED] => ED2): Graph[VD, ED2]
  def mapTriplets[ED2](map: EdgeTriplet[VD, ED] => ED2):
Graph[VD, ED2]
}
```

Structural Operators

The following are a few basic structural operators:



Structural Operators

The following are the syntax of structural operators:

```
class Graph[VD, ED] {
  def reverse: Graph[VD, ED]
  def subgraph(epred: EdgeTriplet[VD,ED] => Boolean,
               vpred: (VertexId, VD) => Boolean): Graph[VD, ED]
  def mask[VD2, ED2](other: Graph[VD2, ED2]): Graph[VD, ED]
  def groupEdges(merge: (ED, ED) => ED): Graph[VD,ED]
}
```

Join Operators

The join operators are used to join data from external collections (RDDs) with graph.

joinVertices()

outerJoinVertices()

joinVertices Operator

The joinVertices is an operator that joins the vertices with the input RDD and returns a new graph with the vertex properties.

```
val nonUniqueCosts: RDD[(VertexId, Double)]
val uniqueCosts: VertexRDD[Double] =
graph.vertices.aggregateUsingIndex(nonUnique, (a,b) => a + b)
val joinedGraph = graph.joinVertices(uniqueCosts)(
  (id, oldCost, extraCost) => oldCost + extraCost)
```

outerJoinVertices Operator

In outerJoinVertices operator, the user defined map function is applied to all vertices and can change the vertex property type.

```
val outDegrees: VertexRDD[Int] = graph.outDegrees
val degreeGraph = graph.outerJoinVertices(outDegrees) { (id,
oldAttr, outDegOpt) =>
  outDegOpt match {
    case Some(outDeg) => outDeg
    case None => 0 // No outDegree means zero outDegree
  }
}
```

Neighborhood Aggregation

Neighborhood aggregation is the key task in graph analytics which includes aggregating information about the neighborhood of each vertex.

graph.mapReduceTriplets → graph.AggregateMessages

aggregateMessages is the core aggregation operation in GraphX which applies a user defined sendMsg function to each edge triplet in the graph.

Neighborhood Aggregation

The following is the syntax of aggregateMessage operator:

```
class Graph[VD, ED] {
  def aggregateMessages[Msg: ClassTag](
      sendMsg: EdgeContext[VD, ED, Msg] => Unit,
      mergeMsg: (Msg, Msg) => Msg,
      tripletFields: TripletFields = TripletFields.All)
    : VertexRDD[Msg]
}
```

Graph-Parallel System

Graph-Parallel System

Web Graphs

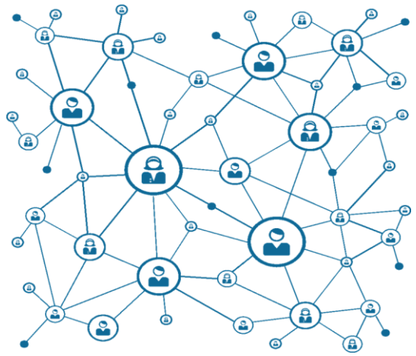
User-Item Graphs

Data Exploding

Target Advertising

Identifying Communities

Deciphering the Meaning of Documents



Web Graphs

User-Item Graphs

Limitations of Graph-Parallel System

1. Each graph-parallel system framework presents a different graph computation.
2. These frameworks depend on different runtimes.
3. These frameworks cannot resolve the data ETL and cannot decipher process issues.

DATA AND
ARTIFICIAL INTELLIGENCE

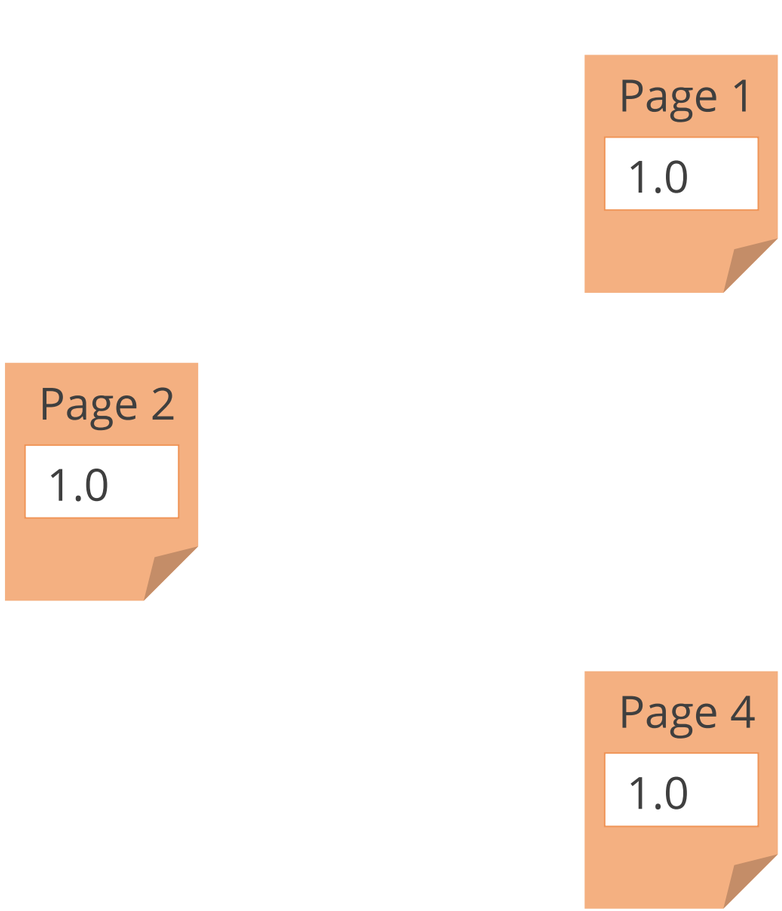
Algorithms in Spark

PageRank Algorithm

- It is an iterative algorithm
- It is used to determine the relevance or importance of a webpage
- It gives web pages a ranking score
- It outputs a probability distribution

PageRank Algorithm

On each iteration, a page contributes to its neighbors its own rank, divided by the number of its neighbors.



contribp = rankp / neighborsp

new-rank = Σcontribs * .85 + .15

PageRank with Social Media Network

GraphX includes a social network dataset on which we can run the PageRank algorithm.

```
import org.apache.spark.graphx.GraphLoader
// Load the edges as a graph
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt")
// Run PageRank
val ranks = graph.pageRank(0.0001).vertices
// Join the ranks with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val ranksByUsername = users.join(ranks).map {
  case (id, (username, rank)) => (username, rank)
}
// Print the result
println(ranksByUsername.collect().mkString("\n"))
```

Connected Components

The connected components is an algorithm that labels each connected component of the graph.

```
import org.apache.spark.graphx.GraphLoader
// Load the graph
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt")
// Find the connected components
val cc = graph.connectedComponents().vertices
// Join the connected components with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val ccByUsername = users.join(cc).map {
  case (id, (username, cc)) => (username, cc)
}
// Print the result
println(ccByUsername.collect().mkString("\n"))
```

Triangle Counting

The triangle counting is an algorithm that determines the number of triangles passing through each vertex, providing a measure of clustering.

```
import org.apache.spark.graphx.{GraphLoader, PartitionStrategy}
// Load the edges in canonical order and partition the graph for triangle count
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt", true)
  .partitionBy(PartitionStrategy.RandomVertexCut)
// Find the triangle count for each vertex
val triCounts = graph.triangleCount().vertices
// Join the triangle counts with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val triCountByUsername = users.join(triCounts).map { case (id, (username, tc)) =>
  (username, tc)
}
// Print the result
println(triCountByUsername.collect().mkString("\n"))
```

Assisted Practice

Working of PageRank Algorithm

Duration: 15 mins

Problem Statement: In this demonstration, you will understand the working of PageRank algorithm.

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**.

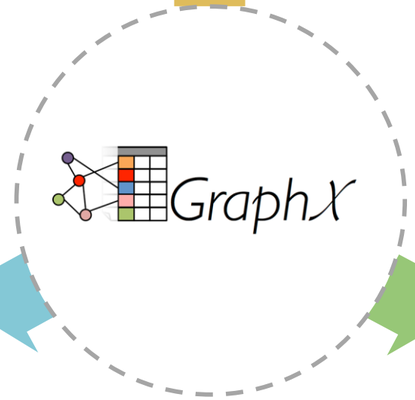
Pregel API

Pregel API

Pregel API is used for developing any vertex-centric algorithm.

It takes a message list as input and also has access to the current state of the vertex attribute and vertex id.

Vertex Program



Send Message Program

It takes the triplet view as the input with all the attributes materialized.

Merge Message Program

It takes two messages meant for the same vertex and combines them into one message.

Pregel API

Pregel API requires the following parameters:

Initial Message

The initial message to start the computation

Max Iteration

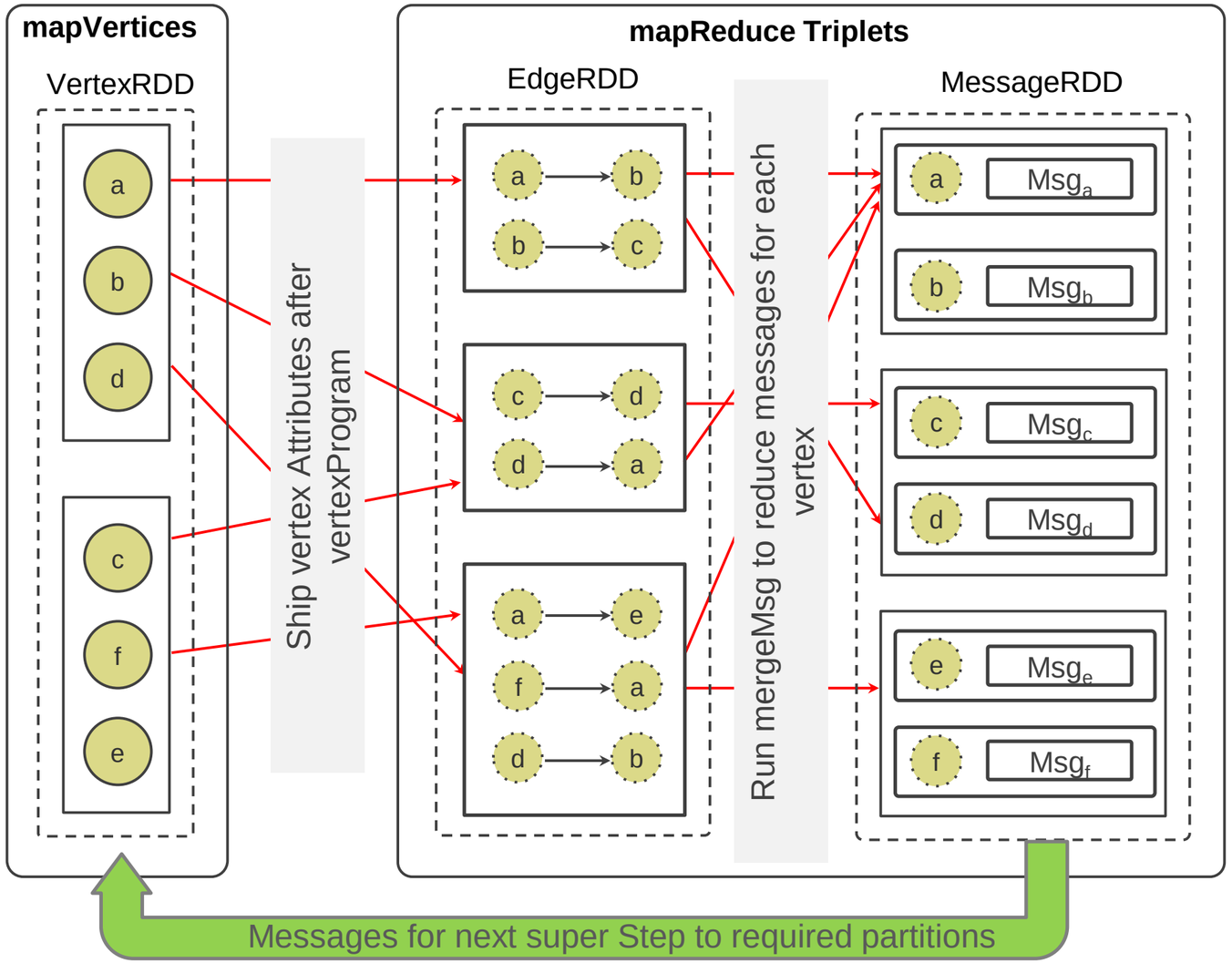
The max number of super steps for the Pregel API

Edge Direction

To filter the edges on which send message function will run

Pregel API

Pregel API in GraphX



DATA AND
ARTIFICIAL INTELLIGENCE

Use Case of GraphX

Use Case of GraphX

Flight Data Analysis Using Spark

Problem

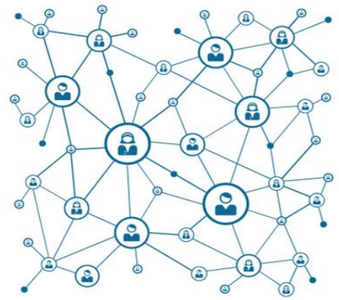
A data analyst wants to analyze the real-time data of flight using Spark GraphX to provide real-time computation results and visualize them.

Use Case of GraphX

Flight data

Database storing real-time flight data

Creating graph using GraphX



GraphX

Query 1

Compute the longest flight route

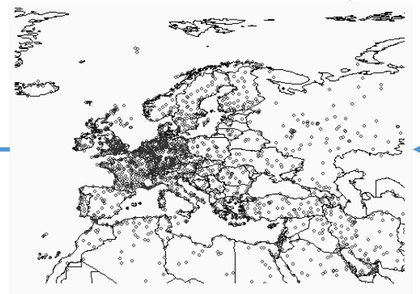
Query 2

Calculate the busiest airports

Query 3

Calculate the routes with lowest flight cost

Visualizing using Google Data Studio



USA Flight Mapping

Results

Assisted Practice

GraphX with Social Media Real-World Problem

Duration: 15 mins

Problem Statement: In this demonstration, you will work on a social medial real-world problem to understand GraphX.

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:

- Define graph and identify the types of graph
- Describe GraphX in Spark
- Identify different operators in GraphX
- Examine PageRank algorithm with social media data

Quiz

Knowledge Check

1

Which of the following is a part of a graph?

a. Edges

b. Vertices

c. Triplets

d. All of the above

Knowledge Check

1

Which of the following is a part of a graph?

a. Edges

b. Vertices

c. Triplets

d. All of the above

The correct answer is **d.**

Edges, vertices, and triplets are parts of a graph.

Knowledge Check

2

Which of the following operators joins the vertices with the input RDD and returns a new graph with the vertex properties?

a. joinVertices()

b. outerJoinVertices()

c. Both a and b

d. None of the above

Knowledge Check

2

Which of the following operators joins the vertices with the input RDD and returns a new graph with the vertex properties?

a. joinVertices()

b. outerJoinVertices()

c. Both a and b

d. None of the above

The correct answer is **a.**

joinVertices() joins the vertices with the input RDD and returns a new graph with the vertex properties.

Knowledge Check

3

Which of the following structural operator constructs a subgraph by returning a graph that contains the vertices and edges that are also found in the input graph?

a. reverse

b. subgraph

c. groupEdges

d. mask

Knowledge Check

3

Which of the following structural operator constructs a subgraph by returning a graph that contains the vertices and edges that are also found in the input graph?

a. reverse

b. subgraph

c. groupEdges

d. mask

The correct answer is **d.**

mask operator constructs a subgraph by returning a graph that contains the vertices and edges that are also found in the input graph

Lesson-End Project

Problem Statement: The US Department of transport collects statistics of all the airlines which includes Airline Details, Airport Details and flight journey details.

Collected data is in CSV format (flights_graph.csv) which contains the following fields:

a. Airline
b. Flight_Number
c. Origin_Airport
d. Destination_Airport
e. Distance
f. Arrival_Delay
g. Arrival_Time
h. Diverted
i. Cancelled

You are hired as a big data consultant to provide important insights. You must write Spark job using its graph component and use the above data to provide the following insights:

1. Routes that have distances greater than 1500 km
2. Routes where max trips are canceled
3. Routes where flights' delayed time was greater than 1300 minutes

DATA AND
ARTIFICIAL INTELLIGENCE

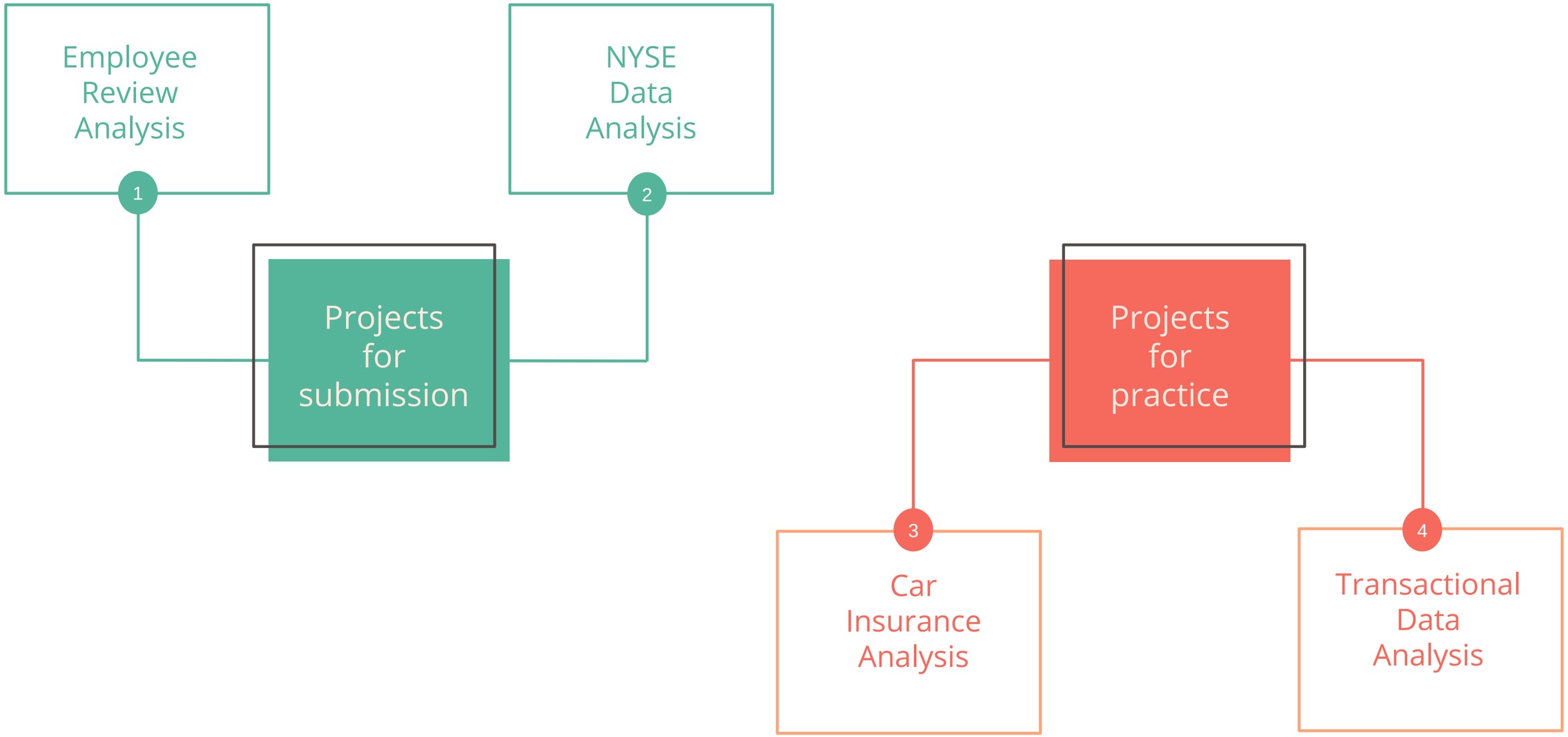
Thank You

Big Data Hadoop and Spark Developer

simplilearn

Project Highlights

Project Highlights



Employee Review Analysis

Objective

Analyze employee review data and provide actionable insights to the HR team for taking corrective actions.

Problem Statement

To improve the employer-employee relationship, current and ex-employee feedbacks and sentiments have been scraped from Glassdoor.

Determine the features and relationships impacting employee satisfaction and derive actionable insights by learning the historical data.

NYSE Data Analysis

Objective

Use Hive features for data engineering and analysis to share actionable insights.

Problem Statement

New York stock exchange data comprises intra-day stock prices and volume traded for each listed company.

The company wants to automate the trading process and predict the next trading-day winners and losers.

Car Insurance Analysis

Objective

Perform exploratory analysis to understand the relationship between multiple features and predict claims.

Problem Statement

A car insurance company wants to analyze its historical data to predict the probability of a customer making a claim.

Transactional Data Analysis

Objective

Use the Big Data stack for data engineering and analysis of transactional data logs.

Problem Statement

Amazon wants to increase their mobile sales by a certain percentage.

You are provided with transactional data which has details about the products, customers, and brands segments.

Analyze the dataset and decide a strategy to increase the sales.

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You