Introduction to Hadoop

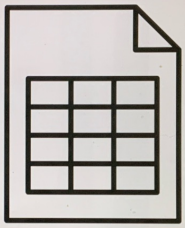
Objectives

After watching this video, you will be able to:

- Define Hadoop
- Explain the history of Hadoop
- List reasons why Hadoop was the answer to Big Data processing

What is Hadoop?

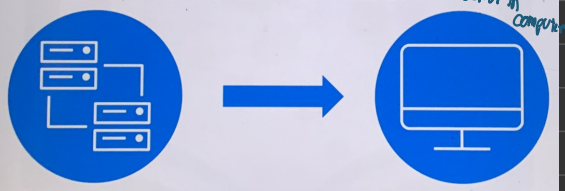
Hadoop is an open-source framework used to process enormous data sets



Why Hadoop?

1GB data

can't enough for increasing data in comput



Why Hadoop?

100GB data

add more ram

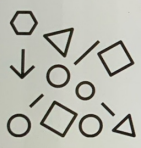
Why Hadoop?

Hadoop was designed to help organizations manage terabytes of data

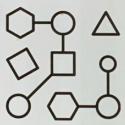
more data

approaching the limits of storage again and again

Why Hadoop?



Unstructured Data



Relational Data

What is Hadoop?

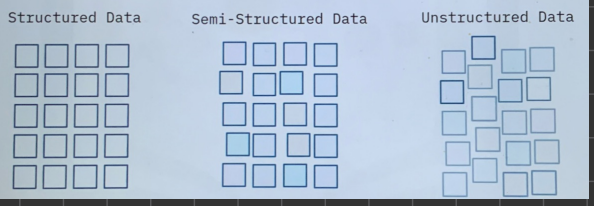
used as the framework for Big Data operations

- Set of open-source programs and procedures
- Used for processing large amounts of data
- Servers run applications on clusters
- Handles parallel jobs or processes

A Hadoop Cluster is a collection of computers working together to perform tasks

Hadoop not a database but an ecosystem that can

Types of Big Data



A little bit of history

1999	Apache software foundation established
2002	Nutch web search engine created
2006	Nutch was divided into Web crawler, distributed systems
2008	Yahoo released Hadoop as an open-source project

How does Hadoop work?

Data is now in petabytes and exabytes

core components

Hadoop Common is an essential part of the Apache Hadoop Framework that refers to the collection of common utilities and libraries that support other Hadoop modules

How does Hadoop work?

storage component

HDFS	MapReduce	YARN
• Hadoop distributed File System • Handles and stores large data • Scales a single Hadoop cluster into as much as thousand clusters	• Known as Hadoop's processing unit • Processes Big Data by splitting the data into smaller units and processes them simultaneously • The first method used to query data stored in HDFS	prepares the RAM and CPU • Yet Another Resource Negotiator acronym • Prepares Hadoop for batch, stream, interactive and graph processing

A commodity hardware is low-specifications industry-grade hardware

The challenges of Hadoop

not good for Hadoop

- Processing transactions (random access)
- When work cannot be parallelized
- When there are dependencies in the data (arise when record one, must be processed before two)
- Low latency data access
- Processing lots of small files
- Intensive calculations with little data

Intro to MapReduce

Objectives

After watching this video, you will be able to:

- Explain the terms Map and Reduce in MapReduce
- Describe why we use MapReduce
- List the components of MapReduce
- Outline examples of common use cases of MapReduce

What is MapReduce?

is the heart of Apache Hadoop

enables massive scalability across hundreds or thousands of servers in a Hadoop cluster

- Programming model used in Hadoop for processing Big Data
- Processing technique for distributed computing is based on Java
- Consists of a Map task and a Reduce task
- Can be coded in many programming languages like Java, C++, Python, Ruby and R

Distributed computing is a system with multiple components located on different machines that communicate actions in one view to the end user

each component has its own job

Map and reduce

takes in

- Input file
- Map: Processes data into key value pairs

these are preliminary output list

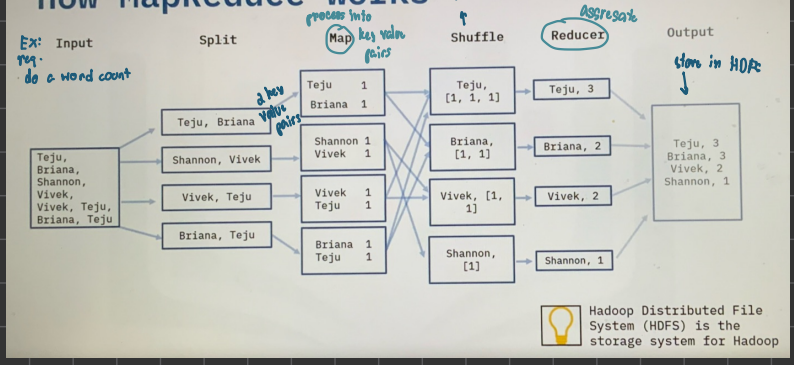
- Further data sorting and organizing before the preliminary output is sent to
- Reducer: Aggregates and computes a set of result and produces a final output

works with multiple map function — the pair from Map

- MapReduce keeps track of its task by creating a unique key

How MapReduce works

sort the key and the list of values



Hadoop Distributed File System (HDFS) is the storage system for Hadoop

Why use MapReduce?

allow high level — advantage

Parallel Computing

Divide ➡ Run Tasks ➡ Done

map reduce allows for splitting and running independent tasks in parallel by divvying each task which in turn save time

has two types: 1. name node, 2. data node

A node is an independent computer used for processing or storing Big Data

flexible

- Process data in tabular and
- Non tabular forms, such as videos
- Support for multiple languages
- Platform for analysis and data ware housing

Common use cases

Social media (LinkedIn, Instagram)	Recommendations (Netflix)	Financial industries (banks)	Advertisement (Google)
Social media platforms can use MapReduce to analyze who visited your profile and who viewed your posts	Create a recommender's system for users and provide suggestions for them based on their interest	Can be used for fraud detection by analyzing behaviors of buyers and tracking down anomalies	Can be used to analyze and understand the interaction with ads and the engagement levels

Hadoop Ecosystem

Objectives

After watching this video, you will be able to:

- List the stages of the Hadoop Ecosystem
- Differentiate between the core components and the extended components
- List some examples of tools used in each stage

Core components of Hadoop: Hadoop common, HDFS, MapReduce, and YARN

HDFS: stores the data collected from the ingestion and distributes the data across multiple nodes

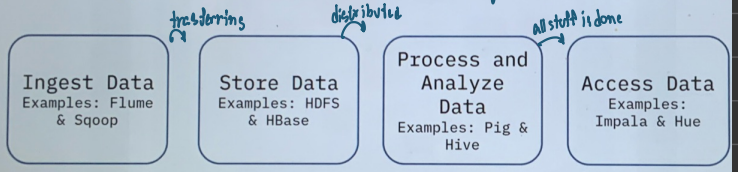
used for making Big Data manageable by processing them in Cluster

YARN: the resource manager across cluster

Hadoop ecosystem

Extended consists of libraries software packages.

The Hadoop ecosystem is made up of components that support one another for big data processing



Ingest data

Ingest Data

- Flume distributed service
  - Collects, aggregates, and transfers big data
  - Has a simple and flexible architecture based on streaming data flows
  - Uses a simple extensible data model that allows for online analytic application
- Sqoop
  - Designed to transfer data between relational database systems and Hadoop
  - Accesses the database to understand the schema of the data
  - Generates a MapReduce application to import or export the data

Store data

Store Data

HBase

- A non-relational database that runs on top of HDFS
- Provides real time wrangling on data
- Stores data as indexes to allow for random and faster access to data

Cassandra

- A scalable, NoSQL database designed to have no single point of failure

Analyze data

Analyze Data

- Pig
  - Analyzes large amounts of data
  - Operates on the client side of a cluster
  - A procedural data flow language
- Hive
  - Used for creating reports
  - Operates on the server side of a cluster
  - A declarative programming language

A declarative programming language allows users express which data they wish to receive

Access data

Access Data

- Impala
  - Scalable and easy to use platform for everyone
  - No programming skills required
- Hue
  - Stands for Hadoop user experience
  - Allows you to upload, browse, and query data
  - Runs Pig jobs and workflow
  - Provides editors for several SQL query languages like Hive and MySQL

HDFS

HDFS

allows programmers to access or store files from any network or computer

- HDFS is the acronym for Hadoop Distributed File System
- It is the storage layer of Hadoop
- Splits the files into blocks, creates replicas of the blocks, and stores them on different machines
- Provides access to streaming data
- HDFS uses a command line interface to interact with Hadoop

Streaming means that HDFS provides a constant bitrate when transferring data rather than having the data being transferred in waves

Key features

Cost efficient	The storage hardware is not expensive
Large amounts of data	HDFS can store up to petabytes of data in any format
Replication	Makes copies of the data on multiple machines
Fault tolerant	If one machine crashes, a copy of the data can be found somewhere else and work continues
Scalable	One cluster can be scaled into hundreds of nodes
Portable	Can easily move across multiple platforms

it splits these large amount of data into small chunks called blocks

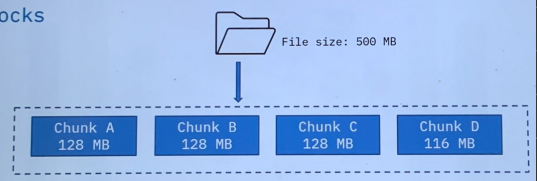
HDFS concepts

Blocks

- Minimum amount of data that can be read or written
- Provides fault tolerance
- Default size is 64 MB or 128 MB

HDFS concepts

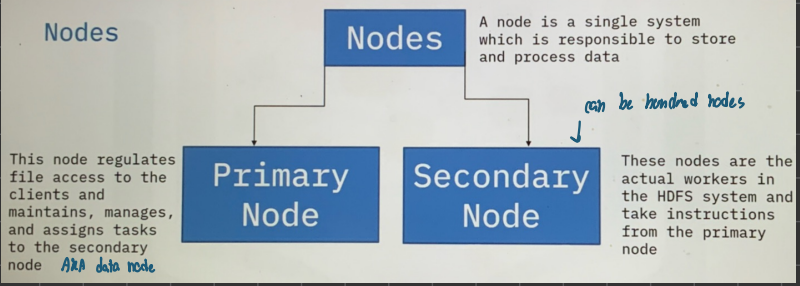
Blocks



- Each file stored doesn't have to take up the configured space size

HDFS concepts

Nodes



HDFS concepts

Rack awareness in HDFS

- Choosing data node racks that are closest to each other
- Improves cluster performance by reducing network traffic
- Name node keeps the rack ID information
- Replication can be done through rack awareness

ensure replicas of data node are in different rack

A rack is the collection of about 40 to 50 data nodes using the same network switch

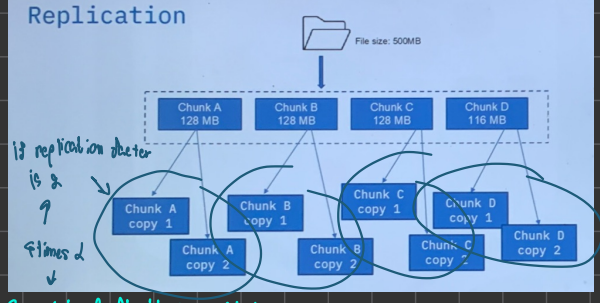
HDFS concepts uses rack awareness concept to create replica

Replication

- Creating a copy of the data block
- Copies are created for backup purposes
- Replication factor: Number of times the data block was copied

HDFS concepts

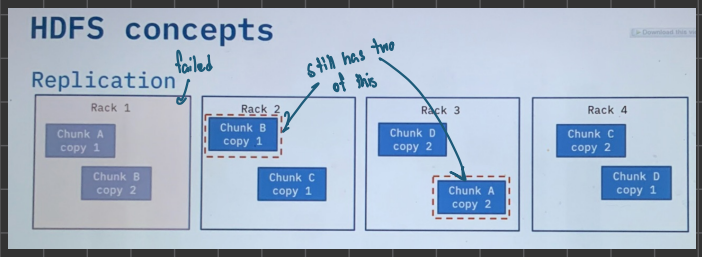
Replication



8 copied of file block (8 blocks)

HDFS concepts

Replication



HDFS concepts

Read and Write Operations

means can not edit files that are already stored in HDFS but can append new data

- HDFS allows write once read many operations

Read

- Client will send a request to the primary node to get the location of the data nodes containing blocks
- Client will read files closest to the data nodes

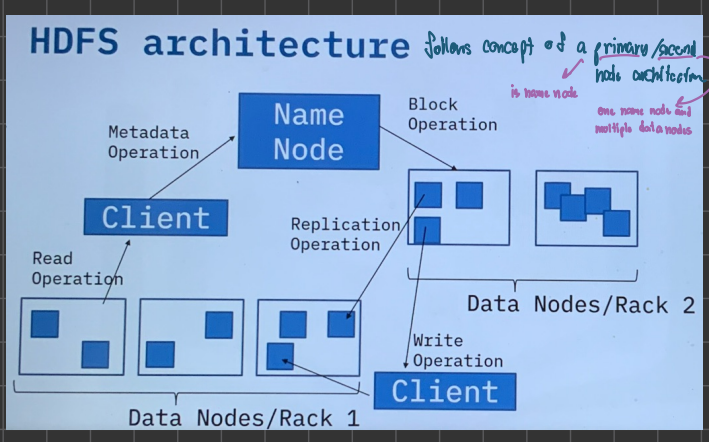
A client fulfills a user's request by interacting with the Name node and Data nodes

Write

- The Name node makes sure that the file doesn't exist
- If the file exists client gets an IO Exception message
- If the file doesn't exist, the client is given access to start writing files

client receives a write permission together with data node
when client is done, the data nodes start creating replicas and sends confirm

HDFS architecture follows concept of a primary/secondary node architecture



HIVE

Objectives

After watching this video, you will be able to:

- Articulate the reasons why Hive is used
- List Hive features
- Differentiate between Hive vs. traditional RDBMS
- Identify components of the Hive architecture
- Discuss Hive concepts

Hive

Hive is data warehouse software within Hadoop that is designed for reading, writing, and managing large tabular-type datasets and data analysis:

- It is scalable, fast, and easy to use
- Hive Query Language (HiveQL) is inspired by SQL, making it easier for users to grasp concepts
- It supports data cleansing and filtering depending on users' requirements

A data warehouse stores historical data from many different sources so that you can analyze and extract insights from it

RDBMS - Relational Database Management System

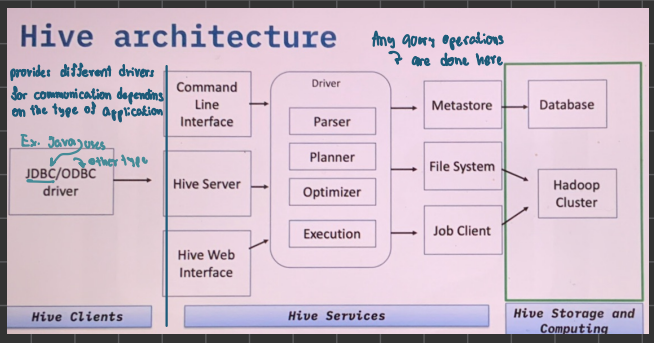
A relational database is a database that stores data in a structured format tabular form

Hive and traditional RDBMS compared

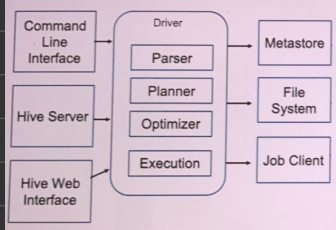
Traditional RDBMS	Hive
Used to maintain a Database and uses SQL	Used to maintain a data warehouse using Hive query language
Suited for real-time/dynamic data analysis like data from sensors	Suited for static data analysis like a text file containing names
Designed to read and write as many times as it needs	Designed on the methodology of write once, read many
Maximum data size it can handle is terabytes	Maximum data size it can handle is petabytes
Enforces that the schema must verify loading data before it can proceed	Doesn't enforce the schema to verify loading data
May not always have built-in for support data partitioning	Supports partitioning

Partitioning means dividing the table into parts based on the values of a particular column, such as date or city

Hive architecture



Hive concepts



Hive Services

Hive Server to enable queries and allow multiple clients to submit requests

The driver receives query statements

The optimizer is used to split tasks efficiently

The executor executes tasks after the optimizer

Hive concepts

JDBC/ODBC driver

Hive Clients

JDBC clients allows Java applications to connect to Hive

ODBC client allows applications based on ODBC to connect to Hive

Metastore stores the metadata information about the tables

HBASE

Objectives

After watching this video, you will be able to:

- Define HBase
- Describe HBase as a columnar database
- List the HBase features and usage
- Outline the differences between HBase and HDFS
- Describe the HBase architecture
- Discuss HBase concepts

HBase

APACHE HBASE

- Column-oriented non-relational database management system
- HBase runs on top of HDFS
- Provides a fault-tolerant way of storing sparse datasets
- Works well with real-time data and random read and write access to Big Data

Fault tolerance refers to the working ability of a system or computer to continue working even in unfavorable conditions such as when a server crashes

HBase features

HBase is used for write-heavy applications

- HBase is linearly and modularly scalable
- It is a backup support for MapReduce jobs
- It provides consistent reads and writes
- It has no fixed column schema
- It is an easy-to-use Java API for client access
- It provides data replication across clusters

An example

Patient Details		Heart Rate	Time Stamp
Patient Name	Patient Age	Heart Rate	Time Stamp
Patient A	28	120 BPM	8:50 AM
Patient A	28	110 BPM	10:10 AM
Patient B	77	95 BPM	11:00 AM
Patient C	45	150 BPM	11:30 AM
⋮			
Patient B	77	115 BPM	12:30 AM

each row is a log record

A few things to note:

- Predefine the table schema and specify column families
- New columns can be added to column families at anytime
- HBase schema is very flexible
- HBase has master nodes to manage the cluster and region servers to perform the work

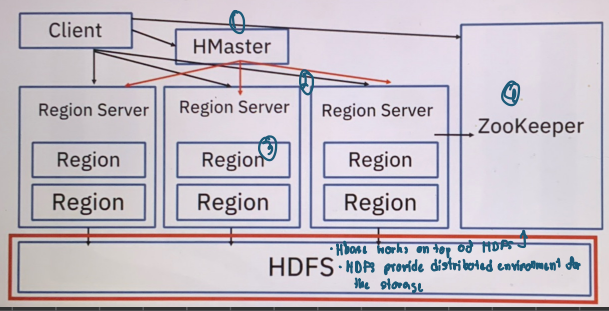
Differences between HBase and HDFS

both used to store massive amounts of data

HBase	HDFS
HBase stores data in the form of columns and rows in a table	HDFS stores data in a distributed manner across different nodes on that network
HBase allows dynamic changes	HDFS has a rigid architecture that doesn't allow changes
HBase is suitable for random writes and reads of data stored in HDFS	HDFS is suited for write once and read many times
HBase allows for storing and processing of Big Data	HDFS is for storing only

HBase architecture

consists mainly of 4 components



HBase concepts

HMaster	Region Servers	Region	Zookeeper
Monitors the region server instances	Receives and assigns requests to regions	Smallest unit of HBase cluster	Maintains healthy links between nodes
Assigns regions to region servers and distributes servers	Responsible for managing regions	Contains multiple stores	Provides distributed synchronization
Manages any changes that are made to the schema	Communicates directly with the client	Two components – HFile and Memstore	Tracks server failure

CHEAT SHEET

Module 2 Cheat Sheet: Introduction to the Hadoop Ecosystem

Package/Method	Description	Code Example
bin/hadoop	All Hadoop commands are invoked by the bin/hadoop script. Running the Hadoop script without any arguments prints the description for all commands.	Running Hadoop script without arguments: `bin/hadoop`

cat	Reads each file parameter in sequence and writes it to standard output. If you do not specify a file name, the cat command reads from standard input. You can also specify a file name of - (dash) for standard input.	Create two sample files.
		`echo "This is file 1" > file1.txt`
		`echo "This is file 2" > file2.txt`
		Use the cat command to read and display the contents of both files
		`cat file1.txt file2.txt`
		Sample output (Contents of file1.txt and file2.txt):
		`This is file 1`
		`This is file 2`

cd	Used to move efficiently from the existing working directory to different directories on your system.	Basic syntax of cd command:
		`cd [options]... [directory]`
		Example 1: Change directory location to "folder1"
		`cd /usr/local/folder1`
		Example 2: Get back to the previous working directory
		`cd -`
		Example 3: Move up one level from the present working directory tree
		`cd ..`

create table	Used to create a new table in a database	Create a new database (if not already created).
		`CREATE DATABASE your_database;`
		Use the newly created database.
		`USE your_database;`
		Create a new table named "employees" in Hive.
		`CREATE TABLE employees (`
		`id INT,`
		`first_name STRING,`
		`last_name STRING,`
		`email STRING,`
		`hire_date DATE`
		`)`
		`ROW FORMAT DELIMITED`
		`FIELDS TERMINATED BY ','`
		`STORED AS TEXTFILE;`
		Show the list of tables in the database.
		`SHOW TABLES;`
		Sample Output (List of Tables):
		`OK`
		`employees`

curl	A command-line tool (pronounced "curl") that allows data to be exchanged between a device and a server through a terminal. The user specifies the server URL, the location where they want to send the request, and the data they want to send to the server URL using this command-line interface (CLI).	Example 1: Sending a GET request and displaying the response
		Send a GET request to a server and display the response.
		`curl https://www.example.com`
		In this example, we use the curl command to send a GET request to https://www.example.com and display the HTML response from the server.
		--- 	
		Example 2: Sending data to a server using POST Request:
		Send a POST request with data to a server and display the response.
		`curl -X POST -d "name=John&age=30" https://www.example.com/api`
		In this example, we use the curl command to send a POST request to https://www.example.com/api with data `name=John&age=30` and display the JSON response from the server.

docker exec	Runs a new command in a running container. It only runs when the container's primary process is running, and it is not restarted if the container is restarted.	Running a command in a running Docker container:
		Run a new command inside a running Docker container.
		`docker exec -it container_name_or_id ls /app`
		Sample Output (List of files in the '/app' Directory inside the container):
		`file1.txt`
		`file2.txt`
		`subdirectory`
		In this example:
		• `docker exec` is used to run a new command (`ls /app`) inside a running Docker container.
		• `-it` enables an interactive terminal session, which allows you to see the output of the command.
		• `container_name_or_id` is the name or ID of the running Docker container you want to execute the command in.
		• `ls /app` is the command that lists the files and directories in the '/app' directory inside the container.

docker-compose	Compose is a tool for defining and running multi-container Docker applications. It uses the YAML file to configure the services and enables us to create and start all the services from just one configuration file.	Starting Docker containers using docker-compose:
		Suppose you have a `docker-compose.yml` file like this:
		`version: '3'`
		`services:`
		`web:`
		`image: nginx:latest`
		`ports:`
		`- "80:80"`
		`db:`
		`image: postgres:latest`
		`environment:`
		`POSTGRES_PASSWORD: example_password`
		You can use `docker-compose` to start the services defined in the docker-compose.yml file as follows:
		Navigate to the directory containing the docker-compose.yml file.
		`cd /path/to/your/docker-compose-project`
		Start the Docker containers defined in the docker-compose.yml file
		`docker-compose up`

docker pull	You can download Docker images from the internet.	`docker pull [OPTIONS] IMAGE_NAME[:TAG]`
docker run	It runs a command in a new container, getting the image and starting the container if needed.	`docker run [OPTIONS] IMAGE [COMMAND] [ARG...]`
git clone	You can create a copy of a specific repository or branch within a repository.	`git clone REPOSITORY_URL [DESTINATION_DIRECTORY]`

hdfs dfs	Apache Hadoop hadoop fs or hdfs dfs are file system commands to interact with HDFS. These commands are very similar to Unix commands. Hadoop provides two types of commands to interact with the file system: hadoop fs or hdfs dfs. The major difference is that Hadoop commands are supported with multiple file systems like S3, Azure, and many more.	Example-1:
		Listing files and directories in HDFS:
		List files and directories in the root directory of HDFS.
		`hdfs dfs -ls /`
		Example-2: In this example, we use the `hdfs dfs -ls` command to list files and directories in the root directory of HDFS.
		`hdfs dfs -ls /`
		Sample output:
		`drwxr-xr-x - hdfs hduser 0 2023-09-13 10:00 /user`
		`drwxrwxrwx - hdfs hduser 0 2023-09-13 10:05 /tmp`
		`drwxrwxrwx - mapred hduser 0 2023-09-13 10:10 /mapred`
		Create a new directory named "mydata" in HDFS.
		`hdfs dfs -mkdir /user/your_username/mydata`

hdfs dfs -cat	Display the contents for a file.	Display the contents of a file in HDFS. `hdfs dfs -cat /path/to/file.txt`
hdfs dfs -mkdir	Creates a directory named path in HDFS	Create a directory in HDFS. `hdfs dfs -mkdir /user/username/mydirectory`
hdfs dfs -put	Upload a file or folder from the local disk to HDFS.	Upload a file from the local file system to HDFS. `hdfs dfs -put localfile.txt /user/username/hdfsfile.txt`

LOAD DATA INPATH	Hive provides the functionality to load precreated table entities either from the local file system or from HDFS. This command is used to load data into the hive table.	Load data from HDFS into a Hive table.
		`LOAD DATA INPATH '/user/username/hdfsfile.txt' INTO TABLE`
		`mytable;`

ls	Writes to standard output the contents of each specified Directory parameter or the name of each specified file parameter, along with any other information you ask for with the flags. If you do not specify a file or directory parameter, the ls command displays the contents of the current directory.	Basic command syntax
		`ls [options] [file/directory]`
		Example 1: Sorts the file names displayed in the order of last modification time. 'r' is for displaying in reverse order
		`ls -lt`
		`ls -ltr`
		Example 2: Displays hidden files
		`ls -a`

mkdir	Used to create one or more directories specified by the Directory parameter. Each new directory contains the standard entries dot (.) and dot dot (..). You can specify the permissions for the new directories with the -m Mode flag.	Create a new directory named "myfolder." `mkdir myfolder`
SELECT * FROM	Lists all the rows from the table to check if the data has been loaded from the file.	Select all rows from a table. `SELECT * FROM tablename;`
show tables	Used to see all the tables in the database that have been selected.	Show all tables in the selected database. `SHOW TABLES;`
tar	Looks for archives on the default device (usually tape) unless you specify another device. When writing to an archive, the tar command uses a temporary file (the /tmp/tar* file) and maintains in memory a table of files with several links.	Create a tar archive of a directory. `tar -cvf archive.tar /path/to/directory`