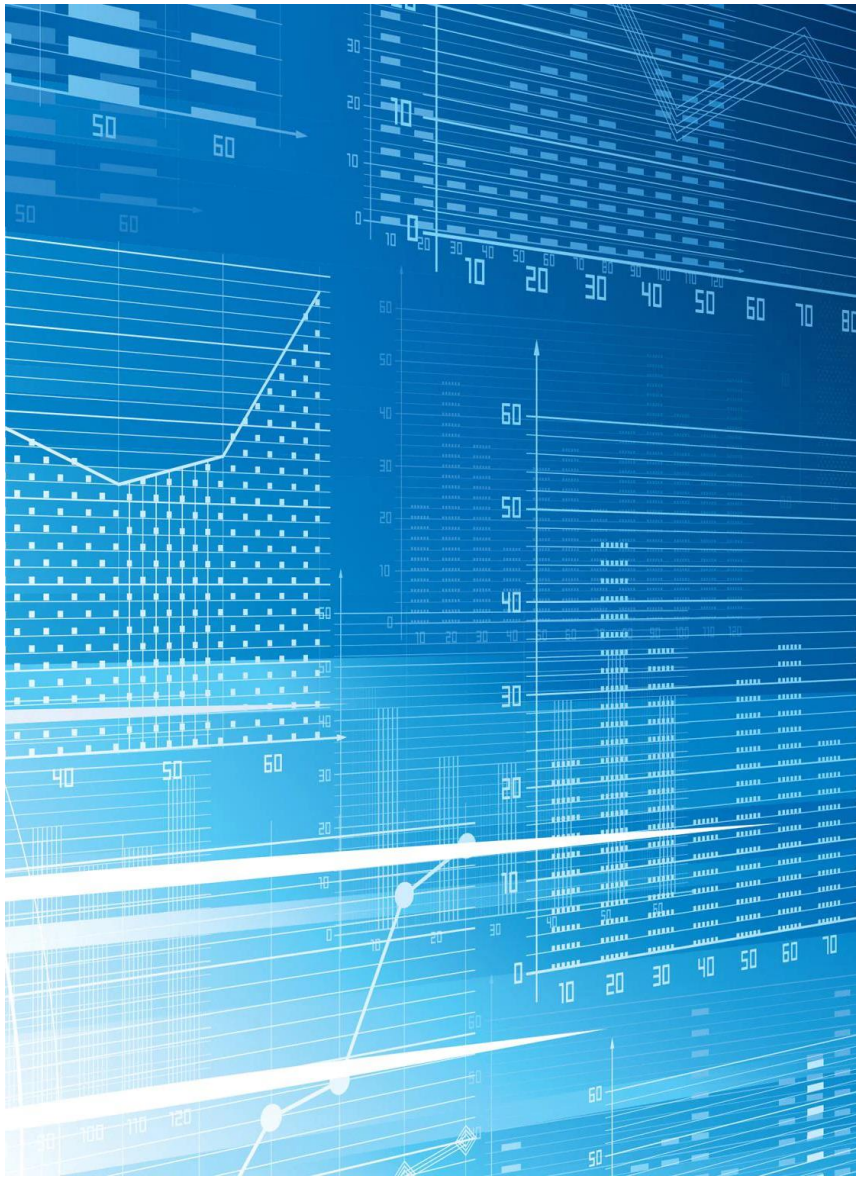
CS412:Data Science

Lecture 24-27: Big Data and Distributed Computing

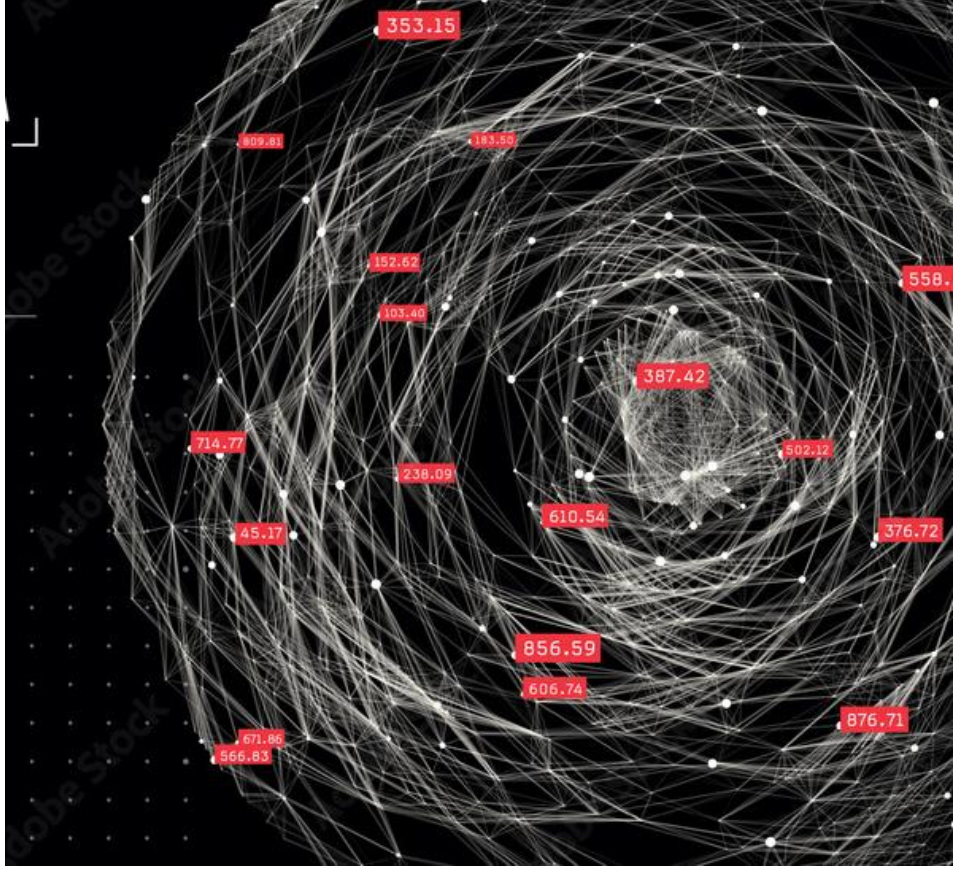
Instructor: Dr. Nora Alkhaldi



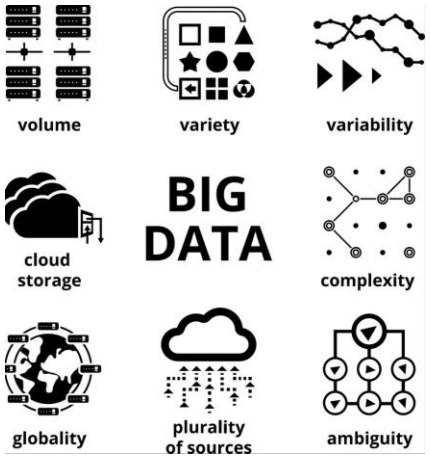
Outline

- Big Data - Recap
- NoSQL Databases
- Apache Hadoop and HDFS
- Hadoop Components
- Apache Spark for Data Processing
- Spark Architecture
- Cloud-based Data Science
- Challenges in Big Data and Distributed Computing

Big Data - Recap

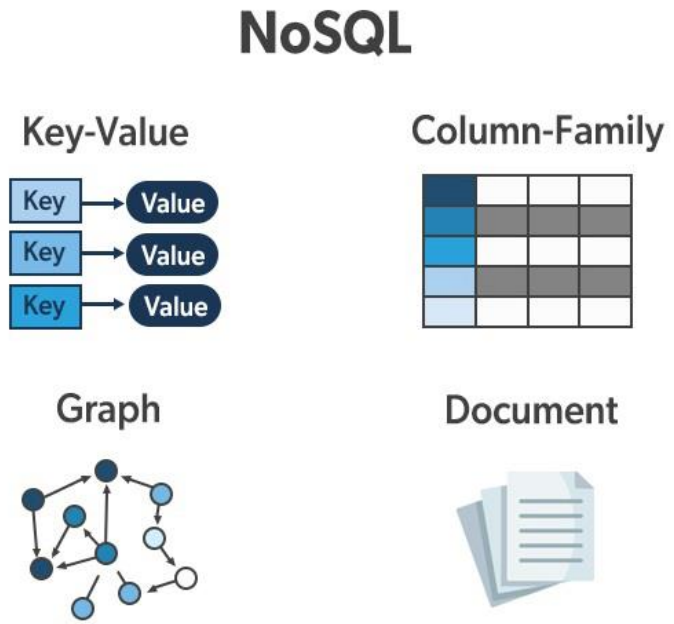


- **Big Data** refers to the massive volume, high velocity, and diverse variety of information that modern computing systems encounter.
- **Characteristics**: Volume, Velocity, Variety
- It goes beyond data volume, embracing the subtleties and complexities of the data, and harnessing it can lead to valuable insights.



NoSQL Databases

- **NoSQL databases (not only SQL)** is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data.
- **Types:**
  - Document
  - Key-Value
  - Wide-Column
  - Graph—varied based on data model.

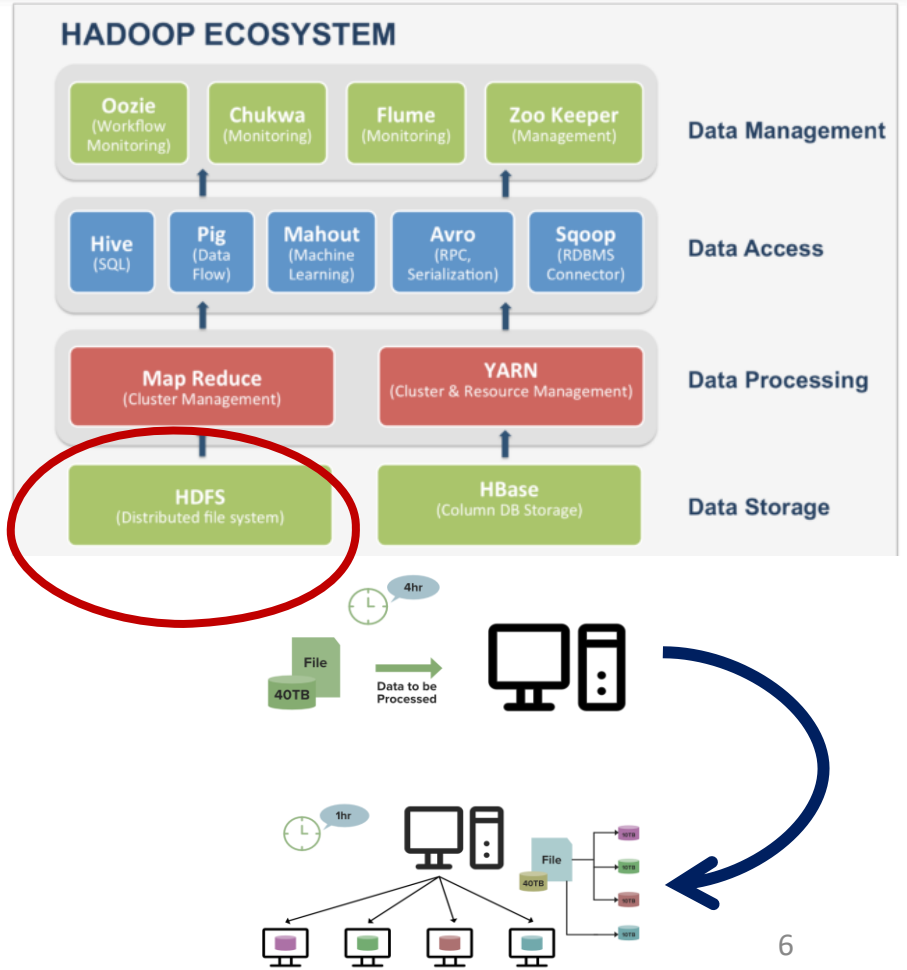


NoSQL Databases

- **Dynamic Handling:**
  - Adaptable to diverse data, departing from traditional database rigidity.
- **Real-Time Excellence:**
  - Excels in real-time processing for social media, e-commerce, and gaming applications.
- **Consistency Balance:**
  - May lack consistency for certain applications, balancing strengths with specific use case considerations.

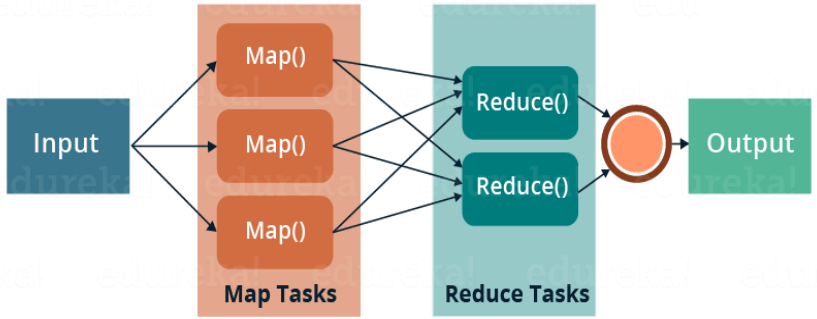
Apache Hadoop and HDFS

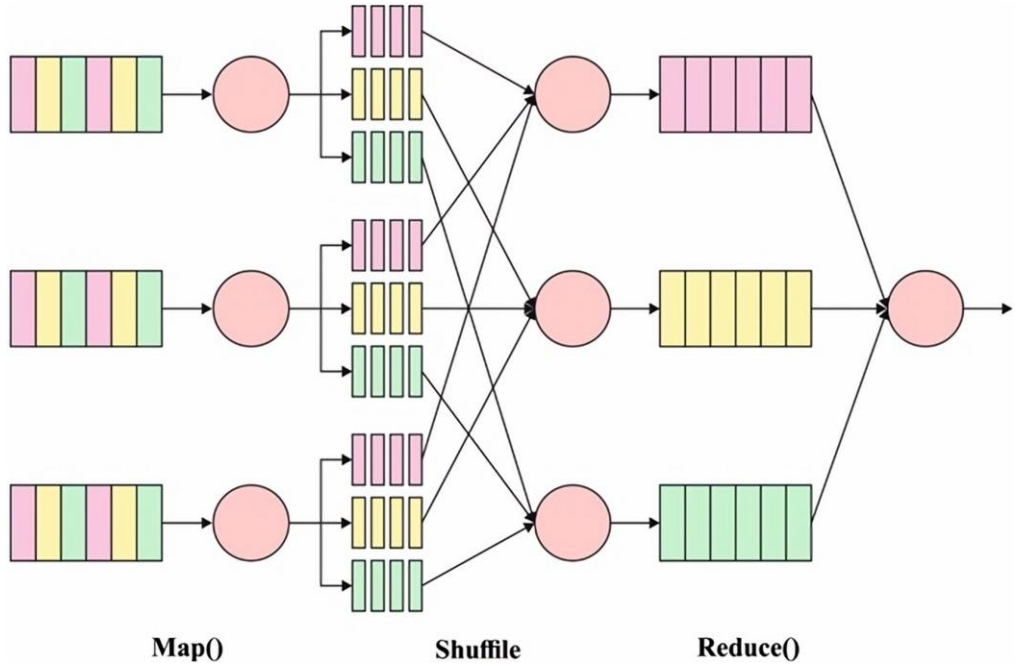
- **Apache Hadoop**, an open-source framework, is tailored to store and process extensive data across distributed clusters, ensuring scalable and efficient data handling.
- **HDFS**
  - The Hadoop Distributed File System (**HDFS**) is integral to Hadoop, systematically breaking down data into manageable chunks and distributing them across interconnected nodes for optimal storage and retrieval.



Hadoop Components

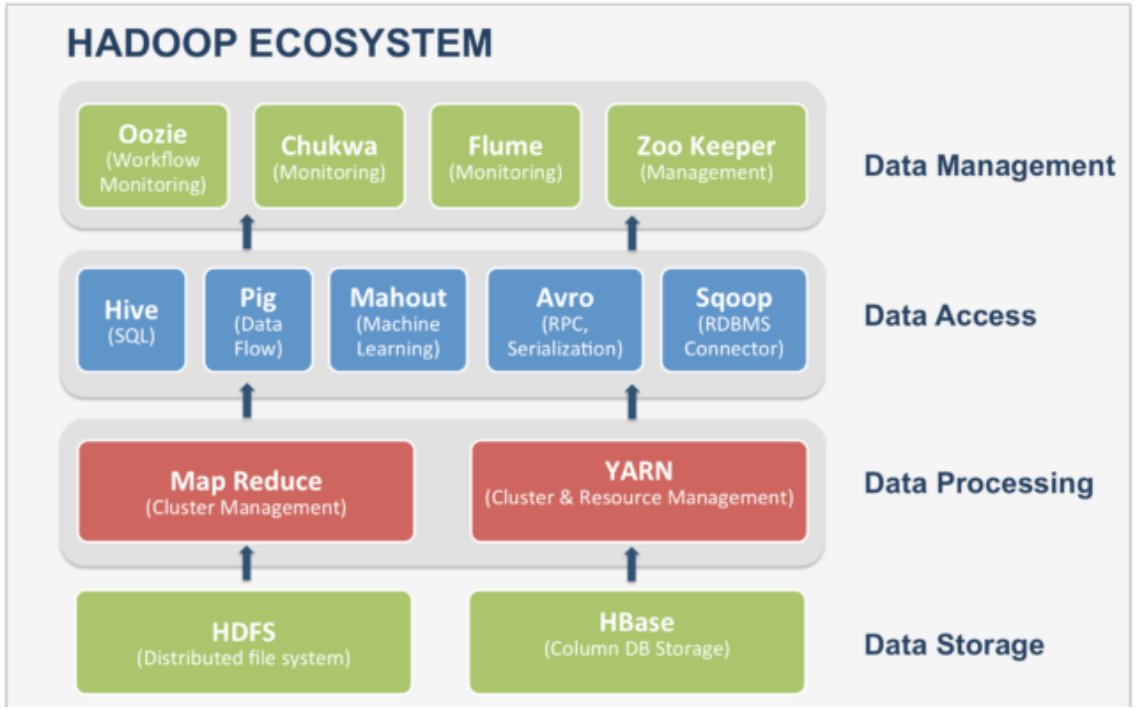
- **MapReduce Paradigm:**
  - It lies at the heart of Hadoop's data processing model.
  - It involves two key steps:
    - **Map** phase divides data into key-value pairs.
    - **Reduce** phase aggregates pairs for analysis.
  - Enables efficient distributed processing of vast datasets





Hadoop Ecosystem

- **Hadoop Ecosystem:** refers to a collection of open-source software projects and tools that complement and extend the capabilities of the core Hadoop framework.
  - **<u>It includes complementary tools.</u>**
    - Map Reduce
    - Hive: SQL-like queries for data warehousing.
    - Pig: Scripting platform for data analysis.
    - HBase: NoSQL database complementing Hadoop's capabilities.



Hadoop EcoSystem

- Hadoop is a data management system bringing together massive amounts of structured and unstructured data that touch nearly every layer of the traditional enterprise data stack, positioned to occupy a central place within a data center
- Hadoop is a massively parallel execution framework bringing the power of supercomputing to the masses, positioned to fuel execution of enterprise applications
- Hadoop view as an open source community creating tools and software for solving Big Data problems

Hadoop EcoSystem

- Hadoop provides such a wide array of capabilities that can be adapted to solve many problems, many consider it to be a basic framework
- Hadoop provides all of these capabilities, but Hadoop should be classified as an ecosystem comprised of many components that range from data storage, to data integration, to data processing, to specialized tools for data analysts

HDFS

- A foundational component of the Hadoop ecosystem is the Hadoop Distributed File System (HDFS)
- HDFS is the mechanism by which a large amount of data can be distributed over a cluster of computers, and data is written once, but read many times for analytics
- It provides the foundation for other tools, such as HBase

MapReduce

- Hadoop's main execution framework is MapReduce, a programming model for distributed, parallel data processing, breaking jobs into mapping phases and reduce phases
- Developers write MapReduce jobs for Hadoop, using data stored in HDFS for fast data access
- Because of the nature of how MapReduce works, Hadoop brings the processing to the data in a parallel fashion, resulting in fast implementation

HBase

- A column-oriented NoSQL database built on top of HDFS, HBase is used for fast read/write access to large amounts of data
- HBase uses Zookeeper for its management to ensure that all of its components are up and running

Zookeeper

- Zookeeper is Hadoop's distributed coordination service
- Designed to run over a cluster of machines, it is a highly available service used for the management of Hadoop operations, and many components of Hadoop depend on it

Oozie

- A scalable workflow system, Oozie is integrated into the Hadoop stack, and is used to coordinate execution of multiple MapReduce jobs
- It is capable of managing a significant amount of complexity, basing execution on external events that include timing and presence of required data

Pig

- An abstraction over the complexity of MapReduce programming, the Pig platform includes an execution environment and a scripting language (Pig Latin) used to analyze Hadoop data sets
- Its compiler translates Pig Latin into sequences of MapReduce programs

Hive

- An SQL-like, high-level language used to run queries on data stored in Hadoop, Hive enables developers not familiar with MapReduce to write data queries that are translated into MapReduce jobs in Hadoop
- Like Pig, Hive was developed as an abstraction layer, but geared more toward database analysts more familiar with SQL than Java programming

Sqoop

- Sqoop is a connectivity tool for moving data between relational databases and data warehouses and Hadoop
- Sqoop leverages database to describe the schema for the imported/exported data and MapReduce for parallelization operation and fault tolerance

Flume

- Flume is a distributed, reliable, and highly available service for efficiently collecting, aggregating, and moving large amounts of data from individual machines to HDFS
- It is based on a simple and flexible architecture, and provides a streaming of data flows
- It leverages a simple extensible data model, allowing you to move data from multiple machines within an enterprise into Hadoop

Growing capabilities and components of Hadoop's ecosystem

Whirr

- This is a set of libraries that allows users to easily spin-up Hadoop clusters on top of Amazon EC2, Rackspace, or any virtual infrastructure

Mahout

- This is a machine-learning and data-mining library that provides MapReduce implementations for popular algorithms used for clustering, regression testing, and statistical modeling

Growing capabilities and components of Hadoop's ecosystem

BigTop

- This is a formal process and framework for packaging and interoperability testing of Hadoop's sub-projects and related components

Ambari

- This is a project aimed at simplifying Hadoop management by providing support for provisioning, managing, and monitoring Hadoop clusters

Avro

- A serialization system for efficient, cross-language RPC, and persistent data storage

HADOOP Architecture

- Operates on top of existing file system
- Files are stored as blocks
  - Block's default size is 64MB
- Provides reliability through replication
  - Each block is replicated across several DataNodes
- NameNode stores metadata and manages access
- No data caching due to large dataset

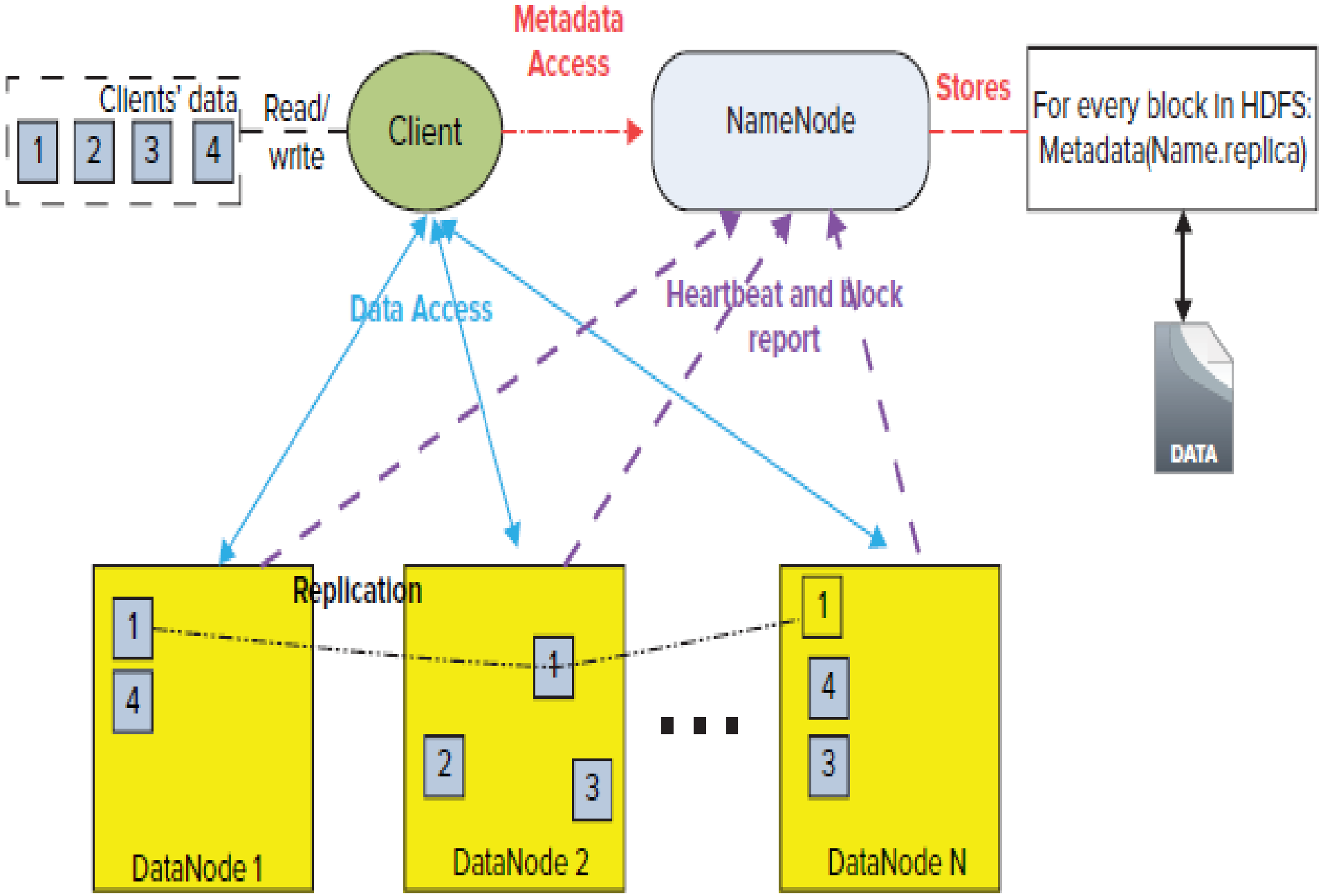


FIGURE 2-1: HDFS architecture

Hadoop Architecture

- HDFS is implemented as a block-structured filesystem
- Individual files are broken into blocks of a fixed size, which are stored across an Hadoop cluster
- A file can be made up of several blocks, which are stored on different DataNodes chosen randomly on a block-by-block basis
- As a result, access to a file usually requires access to multiple DataNodes, which means that HDFS supports file sizes far larger than a single-machine disk capacity

Hadoop Architecture

- The DataNode stores each HDFS data block in a separate file on its local filesystem with no knowledge about the HDFS files themselves
- To improve throughput even further, the DataNode does not create all files in the same directory
- Instead, it uses heuristics to determine the optimal number of files per directory, and creates subdirectories appropriately

Hadoop Architecture

- A single NameNode (which is a master server) manages the filesystem namespace and regulates access to files by clients
- The existence of a single master in a cluster greatly simplifies the architecture of the system
- The NameNode serves as a single arbitrator and repository for all HDFS metadata

Hadoop Architecture

- When a client is writing data to an HDFS file, this data is first written to a local file
- Client then consults the NameNode to get a list of DataNodes that are assigned to host replicas of that block
- The client then writes the data block from its local storage to the first DataNode
- The DataNode stores the received blocks in a local filesystem, and forwards that portion of data to the next DataNode in the list

Hadoop Architecture

- If one of the DataNodes fails while the block is being written, it is removed from the pipeline
- NameNode commits the file creation operation into a persistent store
- If the NameNode dies before the file is closed, the file is lost
- All decisions regarding replication of blocks are made by the NameNode, which periodically (every 3 seconds) receives a heartbeat and a block report from each of the DataNodes

Hadoop Architecture

- A heartbeat is used to ensure proper functioning of DataNodes, and a block report allows verifying that a list of blocks on a DataNode corresponds to the NameNode information
- An important characteristic of the data replication in HDFS is *rack awareness*
- Network bandwidth between machines in the same rack is greater than network bandwidth between machines in different racks

DataNode

- Stores file content as blocks
- Different blocks of the same file are stored on different DataNodes
- Same block is replicated across several DataNodes for redundancy
- Periodically sends a report of all existing blocks to the NameNode
- DataNode exchange heartbeats with NameNode
- If no heartbeat received within certain time period, then DataNode is assumed to be lost

DataNode

- During DataNode failure
  - NameNode will determines which block were present on the lost node
  - NameNode finds other copy of the lost blocks and replicate them to other nodes
  - Block replication is actively maintained

NameNode

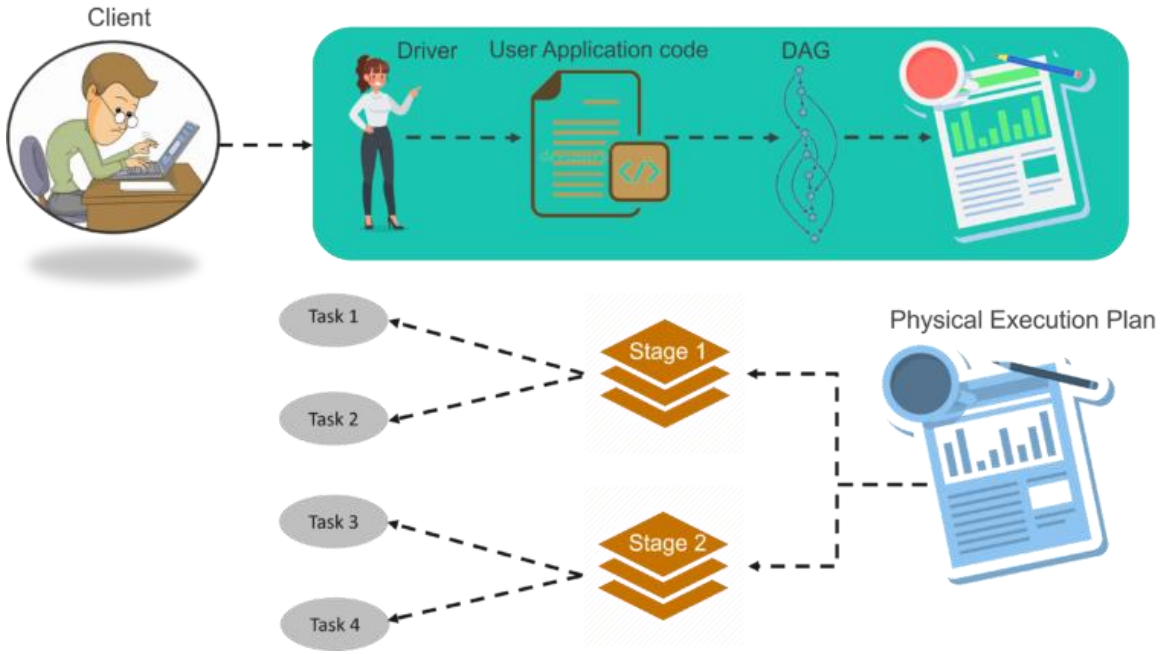
- Stores all metadata: file names, location of each block on data nodes, file attributes, etc.,
- Keeps meta data in RAM for fast lookup
- File system meta data size is limited to the amount of available RAM on NameNode
- NameNode failure
  - Is equivalent to losing all the files on the Hadoop file system
  - Hadoop provides two recovery options
    - Backup files that make up the persistent state of the filesystem (local or NFS mount)
    - Run a secondary NameNode

Secondary NameNode

- Secondary NameNode is not a “backup NameNode”
- It cannot take over the primary NameNode’s function
- It serves as a checkpointing mechanism for the primary NameNode
- In addition to storing the state of the HDFS NameNode, it maintains two on-disk data structures that persist the current file system state: an image file and an edit log

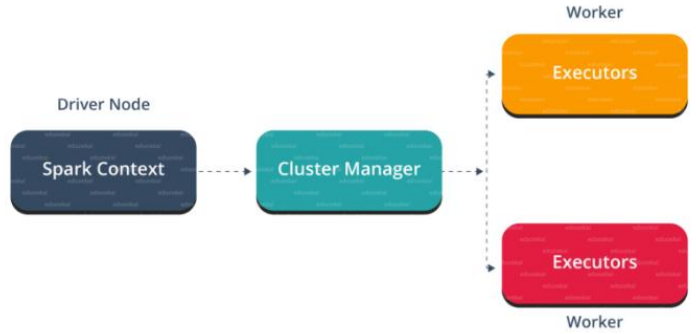
Apache Spark for Data Processing

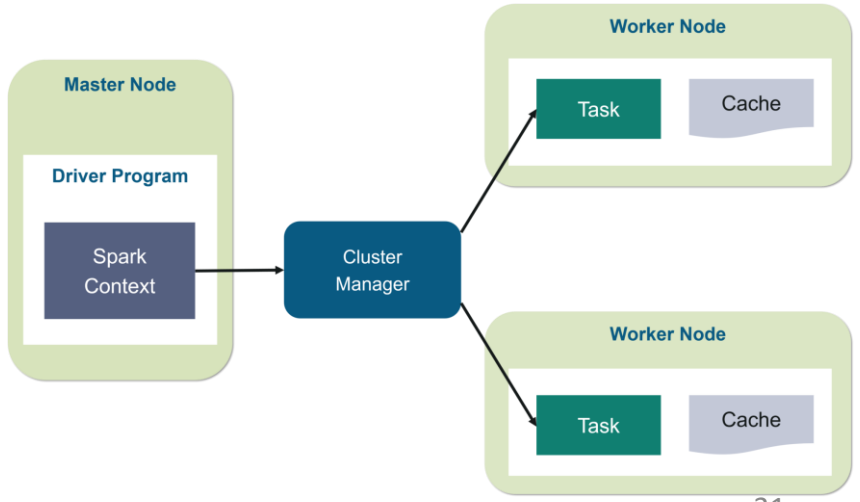
- **Apache Spark** is a distributed computing system that processes large amounts of data in parallel across a cluster of computers.
- A cluster is a collection of nodes that communicate with each other and share data.
- The **architecture of Apache Spark** is based on two main abstractions:
  1. Resilient Distributed Datasets (RDDs)
  2. Directed Acyclic Graphs (DAGs).



Spark Architecture

- **The architecture of Apache Spark consists of four main components:**
  1. **The spark driver** is responsible for coordinating the execution of tasks across the cluster.
  2. **The spark executors** are responsible for executing these tasks on individual worker nodes.
  3. **The cluster manager** is responsible for managing the allocation of resources across the cluster.
  4. **The worker nodes** are responsible for executing tasks assigned to them by the Spark executors.





https://www.interviewbit.com/blog/apache-spark-architecture/

Spark vs. Hadoop

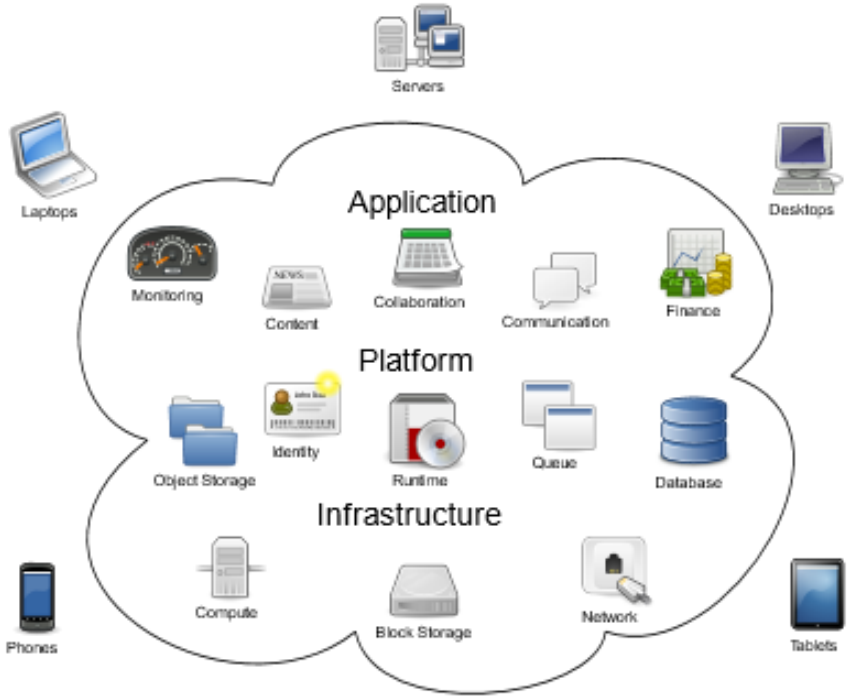
- **Hadoop Architecture:**
  - Based on MapReduce programming model.
  - Processes data in a batch-oriented manner.
- **Spark Architecture:**
  - Based on Resilient Distributed Datasets (RDDs) abstraction.
  - Processes data in-memory, making it faster for specific workloads.
- **Ease of Use:**
  - Spark provides a more user-friendly API compared to Hadoop.
- **Real-time Processing:**
  - Spark is well-suited for real-time data processing.
  - Hadoop is more oriented towards batch processing.
- **Machine Learning:**
  - Spark offers built-in support for machine learning algorithms.
- **Graph Processing:**
  - Spark includes built-in support for graph processing.

What exactly is Apache Spark? | Big Data Tools Demo

https://youtu.be/ymtq8yjmD9I?feature=shared

Cloud Computing

"Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured IT resources."

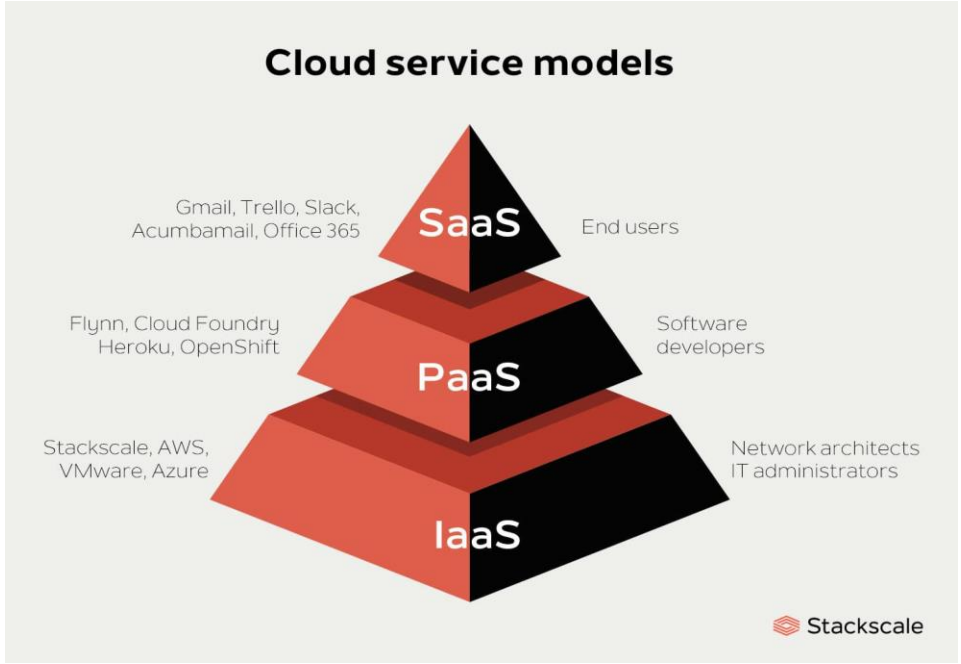


Cloud Characteristics

1. **On-Demand Usage:** Users access computing resources as needed, paying for what is consumed.
2. **Ubiquitous Access:** Cloud services are accessible anytime, anywhere with internet connectivity.
3. **Multitenancy:** Multiple users share resources, optimizing efficiency and cost-effectiveness.
4. **Elasticity:** Scalability to handle varying workloads, adapting resources dynamically as demand fluctuates.
5. **Measured Usage:** Consumption is metered and billed based on actual usage of resources.
6. **Resiliency:** Cloud systems are designed for high availability and fault tolerance, ensuring continuous operation.

Cloud Delivery Models

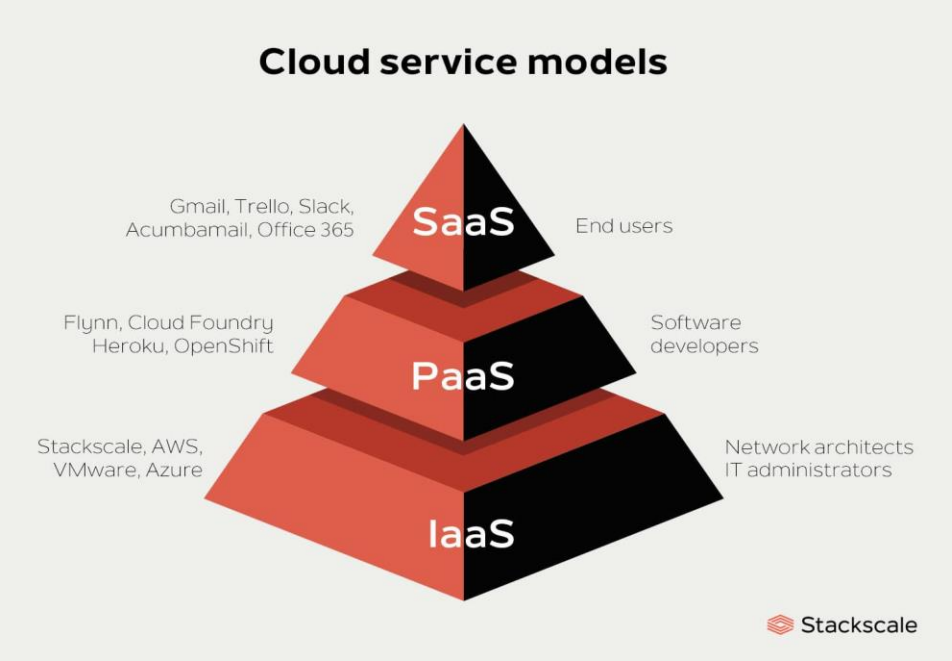
- **Cloud Delivery Models** refer to the different ways in which cloud computing services are delivered to users.
- **Three common delivery models are used:**
  1. Infrastructure-as-a-Service (IaaS)
  2. Platform-as-a-Service (PaaS)
  3. Software-as-a-Service (SaaS)
- Variations of these delivery models can also exist.



Cloud Delivery Models

Infrastructure-as-a-Service (IaaS)

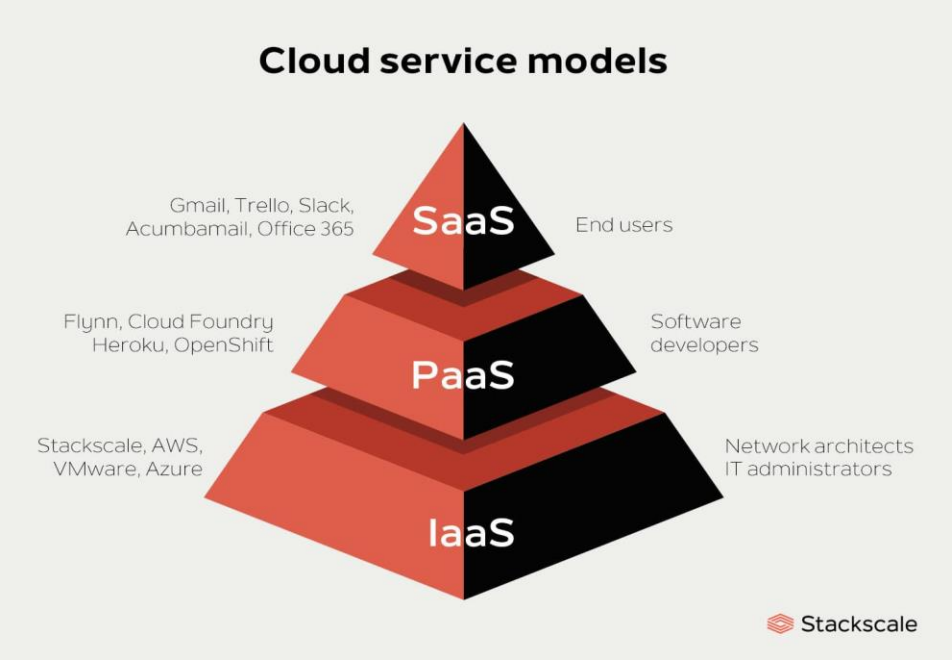
Offers virtualized computing resources over the internet, such as virtual machines and storage.



Cloud Delivery Models

Platform-as-a-Service (PaaS)

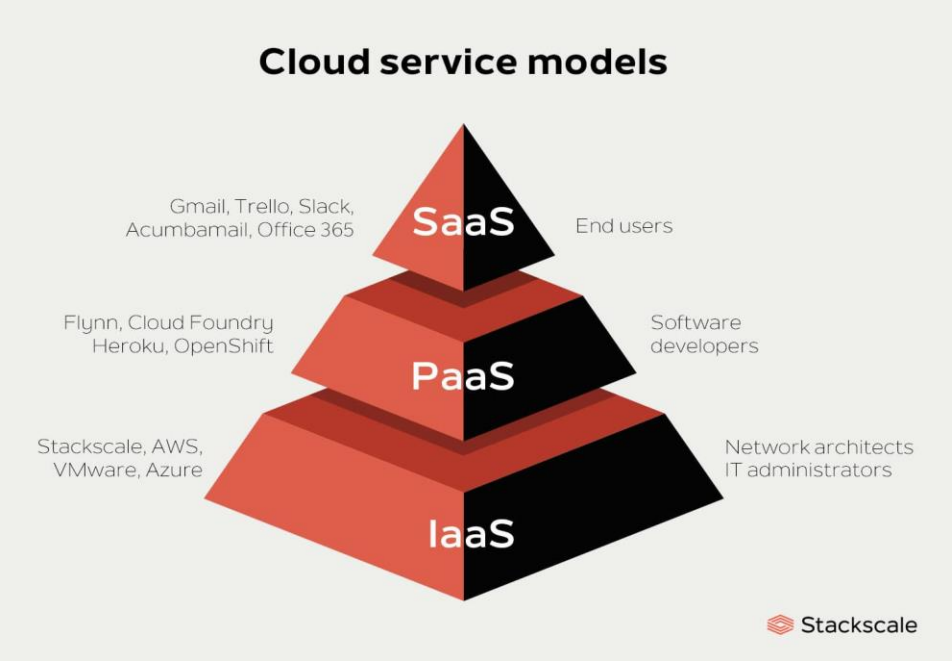
Provides a platform with tools and services for application development, eliminating the need for infrastructure management.



Cloud Delivery Models

Software-as-a-Service (SaaS)

Delivers software applications over the internet, accessible through web browsers without local installations.

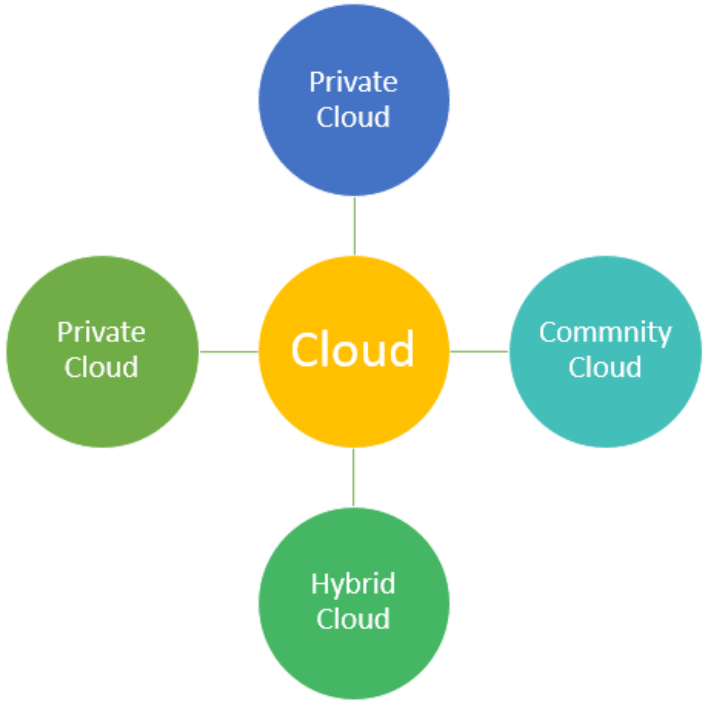


Cloud-based Data Science

- **Cloud-based data science** is a rapidly growing field that leverages cloud computing to perform data analysis and machine learning tasks.
- **Cloud-based data science platforms** offer a complete suite of data management, analytics, and machine learning tools to generate insights and unlock value from data
- **Common Cloud-Based Platforms for Data Science:**
  - Amazon Web Services (AWS)
  - Google Cloud Platform (GCP)
  - IBM Watson
  - Microsoft Azure
  - Oracle Cloud

Cloud Deployment Model

- **A Cloud Deployment Model** represents a specific type of cloud environment, primarily distinguished by ownership and size.
- **Cloud deployment models** define how cloud resources are configured and accessed.
- **There are four common deployment models:**
  1. Public Cloud
  2. Community Cloud
  3. Private Cloud
  4. Hybrid Cloud
- Variations of these deployment models can also exist



Cloud Deployment Model

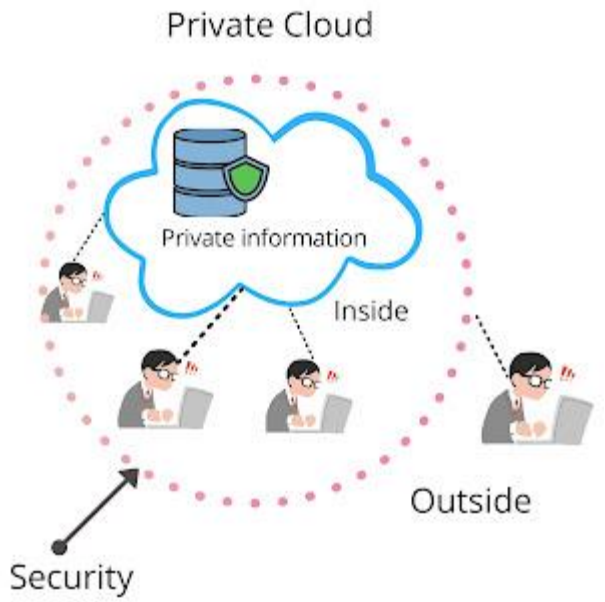
Public Cloud

- Services provided by third-party vendors, accessible to the public over the internet.

Cloud Deployment Model

Private Cloud

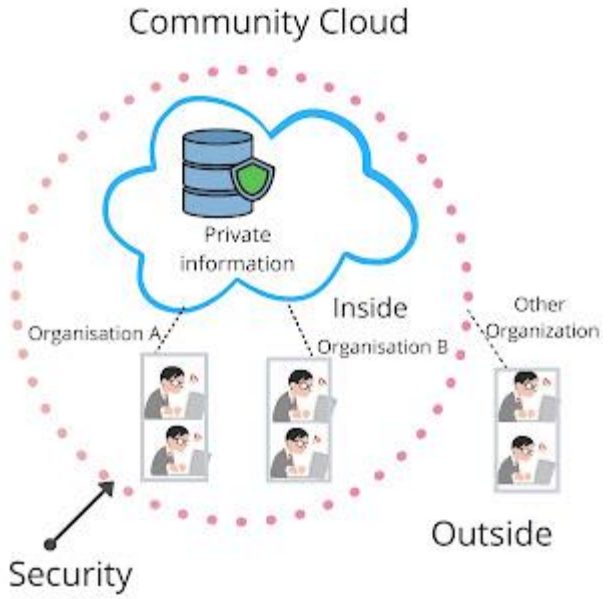
- Cloud infrastructure exclusively used by a single organization, offering enhanced control and privacy.



Cloud Deployment Model

Community Cloud

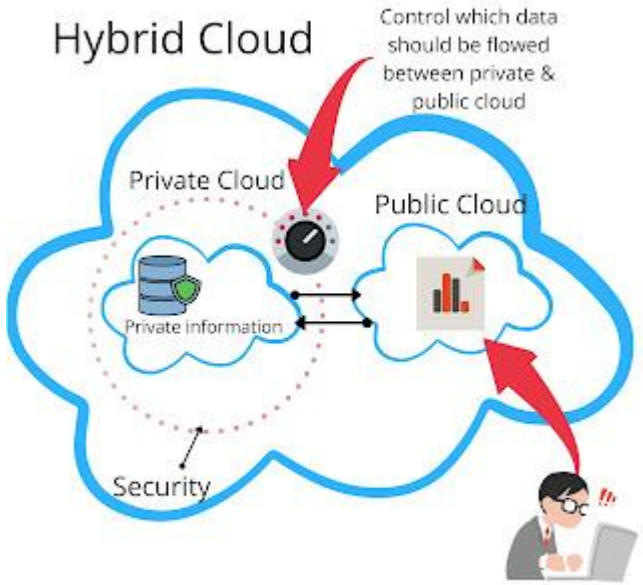
- Shared cloud infrastructure serving a specific community with shared concerns, policies, and compliance requirements.



Cloud Deployment Model

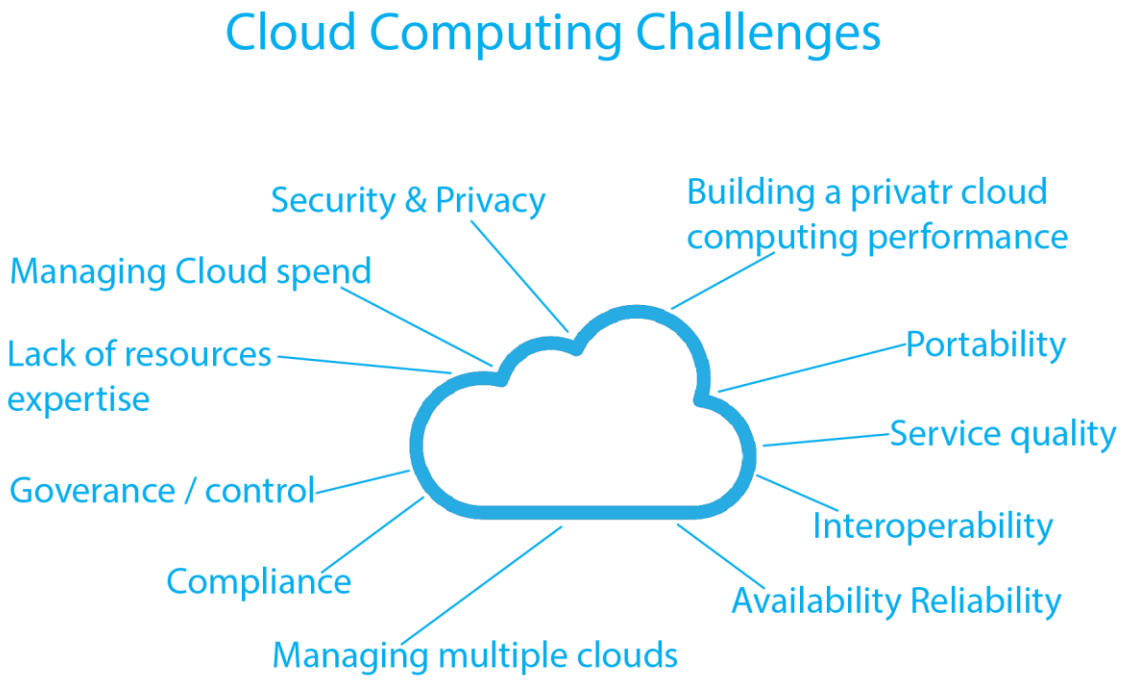
Hybrid Cloud

- Combination of public and private clouds, facilitating data and application portability between them.

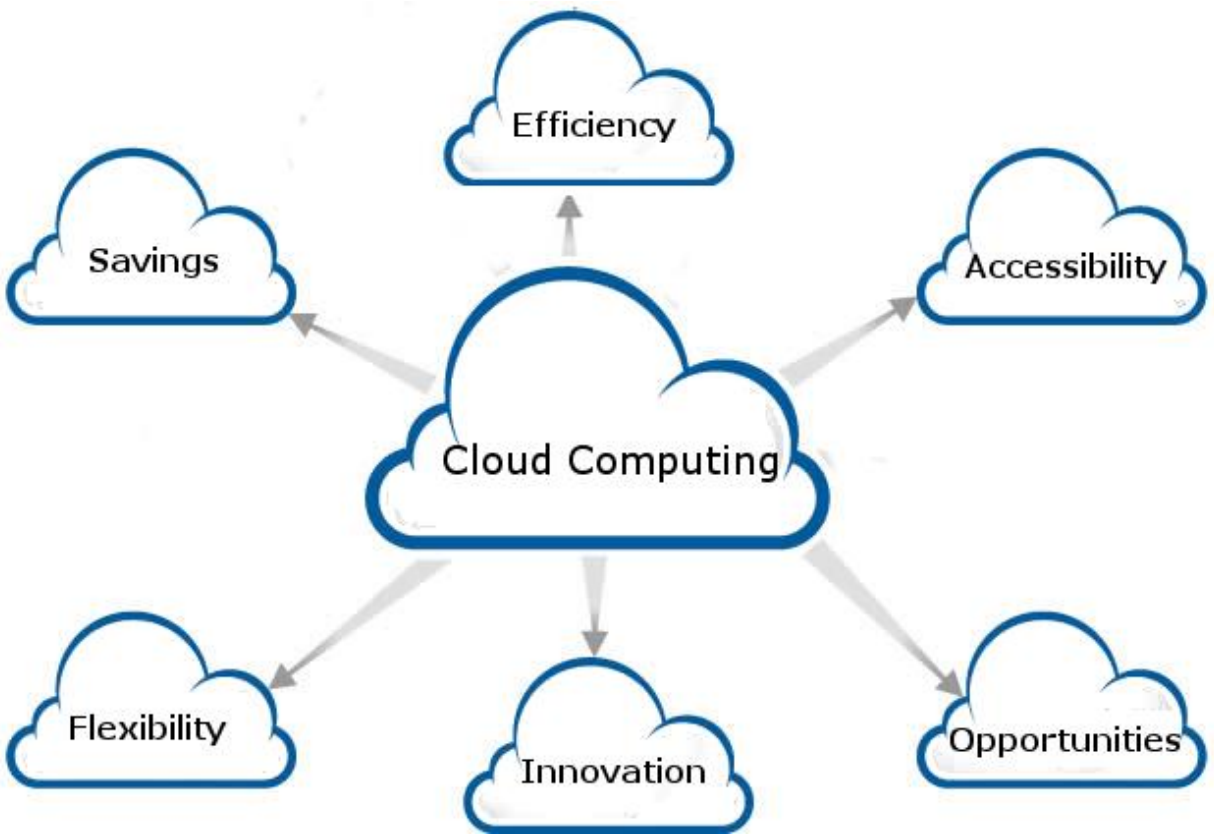


Challenges of Cloud Computing

1. Data Security and Privacy
2. Data Integration
3. Scalability
4. Distributed System Complexity
5. Fault Tolerance
6. Data Quality and Consistency
7. Cost Management
8. Tool and Technology Selection
9. Skill Shortage
10. Regulatory Compliance



Benefits of Cloud Computing



Thanks
Any Questions ?

End of Lecture