Big Data

What is big data?

Big data refers to extremely large, diverse collections of structured, unstructured, and semi-structured data that continue to grow exponentially over time.

These datasets are often characterised by their volume, velocity, and variety, requiring specialised tools and methods for storage, processing, and analysis to extract valuable insights and inform decision-making.

How is big data relevant to data engineering?

In the context of data engineering, big data refers to the practice of handling and managing extremely large and complex datasets using various data engineering techniques and technologies. Data engineers are responsible for designing and implementing the infrastructure and pipelines required to collect, store, process, and transform big data efficiently.

- **Data Collection**: Data engineers build systems and processes to ingest data from various sources, including sensors, logs, databases, external APIs, and more. This data can be massive in volume and arrive at high velocities.
- **Data Storage**: Big data often requires specialised storage solutions, such as distributed file systems (e.g., Hadoop HDFS) and NoSQL databases (e.g., Cassandra, MongoDB). Data engineers design and maintain these storage systems to ensure scalability and fault tolerance.
- **Data Processing**: Processing big data involves distributed computing frameworks like Apache Spark or Apache Flink. Data engineers set up and optimise these frameworks to perform tasks like data transformation, cleaning, and aggregation efficiently.
- **Data Integration**: Data engineers work on integrating and harmonising data from different sources and formats to create a unified and usable dataset. This often involves data cleansing and normalisation.
- **Data Pipelines**: Building robust data pipelines is a critical aspect of data engineering for big data. These pipelines automate the flow of data from source to destination, ensuring data quality, reliability, and timeliness.
- **Scalability**: As the volume of data grows, data engineers must design systems that can scale horizontally to handle increasing data loads. This may involve adding more servers or leveraging cloud-based solutions.
- **Data Security and Compliance:** Data engineers are responsible for ensuring that big data systems comply with security and privacy regulations. They implement access controls and encryption to protect sensitive data.

- **Performance Optimisation**: Optimising the performance of data processing and storage systems is crucial. Data engineers continually monitor and fine-tune systems to achieve the desired performance levels.
- **Data Quality**: Maintaining data quality is paramount in big data projects. Data engineers implement data validation and quality checks to identify and rectify issues.
- **Real-time Processing**: In some cases, big data processing needs to be done in real-time or near-real-time. Data engineers set up streaming data pipelines to handle such requirements.

Where is it coming from?

The surge in data, driven by digital technology advances like connectivity, mobility, IoT and AI, is giving rise to new tools for collecting, processing, and analysing data, empowering businesses to extract maximum value.

Big data is generated from many sources:

- **Machine data** - automatically generated as a response to an event or on a fixed schedule. Data is generated from multiple sources such as smart sensors, SIEM logs, cameras, medical devices, mobile phones, satellites and are often used to track consumer behaviour. Data extracted from machines grow exponentially along with the changing external market
- **IoT** - devices like smart sensors, connected appliances, industrial machines, and wearable devices generate real time data are becoming more prevalent
- **Manufacturing and Industry 4.0** - Manufacturing processes are becoming increasingly automated and data-driven, generating data on production efficiency, machine performance, and product quality
- **Social media data** - billions of users generate data through user interactions, posts, comments, clicks. It offers valuable information on customer behaviour, product sentiment, and market sentiment. Leveraging this data helps businesses connect with their online audience, understand their market, and improve decision-making.
- **Ecommerce and online transactions** - business generate large volumes of data related to customer transactions, purchases history, and payment processing
- **Financial services** - banks, stock exchanges, and investment firms generate huge amounts of data through stock market transactions, customer accounts, and risk analysis

What enabled big data to emerge as a technology?

Several technological advancements and trends have contributed to the emergence of big data as a significant technology:

- Increased data generation
- Advances in storage solutions such as distributed file systems like Hadoop HDFS and scalable NoSQL databases like MongoDB enable scalable storage of datasets
- Availability of high-performance computing clusters and cloud computing platforms have allowed processing of huge datasets efficiently using technologies such as MapReduce, Spark, and GPUs
- Data Integration tools and ETL tools like Apache NiFi streamline the process of gathering data from multiple sources
- Open-source software, such as Hadoop, Apache Spark, and various NoSQL databases, has played a significant role in making big data technologies accessible and cost-effective
- Cloud Computing advancements as Cloud providers like Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) offer scalable and on-demand resources for big data processing and storage. It offers scalable and cost-efficient resources for storage, processing, and analytics over the internet. Key benefits include on-demand scalability, cost-effectiveness, global accessibility, managed services, versatile data storage, integrated analytics tools, and robust security and compliance measures

The history of big data

Big data has its origins in the 1960s and '70s, with the development of data centres and relational databases. The rise of user-generated data from platforms like Facebook and YouTube in 2005 led to Hadoop and NoSQL databases. Open-source frameworks made big data more accessible. Its growth surged with IoT and machine learning, and cloud computing expanded possibilities. Graph databases emerged for fast analytics, signalling ongoing potential.

Big data use cases

- Product Development: Companies like Netflix and Procter & Gamble use it to predict customer demand, classify product attributes, and model success factors
- Predictive Maintenance: It uncovers indicators of mechanical failures in structured and unstructured data, enabling cost-effective maintenance and maximum equipment uptime
- Customer Experience: Big data improves interactions by gathering data from various sources, allowing personalised offers, reduced churn, and proactive issue handling
- Fraud and Compliance: It identifies fraud patterns and expedites regulatory reporting, addressing evolving security and compliance challenges
- Machine Learning: Big data facilitates machine learning by providing ample training data, enabling machines to learn independently
- Operational Efficiency: Big data enhances operational efficiency by analysing production, customer feedback, and returns, reducing outages, and aligning decision-making with market demand.
- Innovation: It drives innovation through data insights, exploring interdependencies, improving financial decisions, and delivering new products, services, and dynamic pricing.

The V's of big data

Volume: Big data is characterised by its vast amount of continuously generated and collected data from various sources and devices.

Velocity: Refers to the speed at which data is produced, often in real-time, necessitating rapid processing and analysis for meaningful insights.

Variety: Big data is heterogeneous, encompassing structured, unstructured, and semi-structured data from diverse sources like text, images, audio, video, and sensor data.

Veracity: Big data can be messy and error-prone, challenging data quality and trustworthiness.The data coming in are often raw, unfiltered, uncleaned and discrepancies need to be dealt with.

Variability: The changing structure, meaning and context of data can lead to inconsistency over time, affecting interpretation and collection methods.

Value: It's crucial to assess the business value of collected data and effectively analyse it to inform decision-making

Tools:

Volume:

Hadoop HDFS: Hadoop Distributed File System is widely used for storing and managing large volumes of data across a distributed cluster of servers.
Amazon S3: Amazon Simple Storage Service is a popular cloud-based storage solution that offers scalable storage for big data applications.
Google Cloud Storage (GCS): Google's equivalent of S3, GCS, provides a highly available and scalable storage platform for big data workloads.

Velocity:

Apache Kafka: Kafka is a distributed streaming platform that is commonly used for handling high-velocity data streams with low latency requirements and enabling real-time data processing.
Apache Flink: Apache Flink is a stream processing framework that supports real-time analytics and event-driven applications.
Apache Spark Streaming: While primarily known for batch processing, Spark also offers a streaming component for real-time data processing.

Variety:

Apache Nifi: Apache NiFi is an open-source data integration tool that supports the collection and transformation of data from various sources with diverse formats.
Talend: Talend is a popular data integration platform that helps organisations work with a variety of data formats and sources.
Apache Avro: Avro is a data serialisation framework that is used to handle data with diverse schemas efficiently.

Veracity:

Trifacta: Trifacta is a data preparation and data wrangling tool that helps clean and transform data to improve its veracity.
OpenRefine: OpenRefine is an open-source tool for data cleaning and transformation, particularly useful for dealing with messy data.
Talend Data Quality: Talend offers data quality tools that include data profiling and cleansing capabilities to enhance data veracity.

Variability:

Apache Avro: Avro's schema evolution capabilities make it a useful tool for handling changing data schemas.
Confluent Schema Registry: Confluent's Schema Registry is often used with Kafka to manage evolving data schemas in real-time data pipelines.
Git: Version control systems like Git are employed to track changes in data schemas and data-related code.

Value:

Tableau: Tableau is a popular data visualisation tool that helps extract insights and value from big data through interactive dashboards and reports.
Python and R: Programming languages like Python and R are commonly used for advanced analytics, including machine learning, natural language processing, and predictive modelling.
Databricks: Databricks is a unified analytics platform that integrates with big data technologies like Apache Spark for data analytics and machine learning.

How does big data work?

Big data works by collecting, managing, and analysing vast amounts of data to gain valuable insights and improve decision-making. The process involves three main actions:

Integration: Big data collects raw data from various sources and transforms it into a format that can be analysed. This data can be terabytes or even petabytes in size.

Management: Storing and processing big data requires significant storage capacity, which can be in the cloud, on-premises, or a combination of both. Real-time processing and scalability are crucial, with many companies turning to cloud solutions for their computing needs.

Analysis: The final step is analysing and acting on big data insights. This includes exploring the data, communicating findings across the organisation, and using tools like data visualisations to make the data understandable.

Benefits and challenges of big data

Benefits of big data include:

- Improved Decision-Making: Big data enables data-driven decision-making by uncovering patterns and insights that enhance operational and strategic choices.
- Increased Agility and Innovation: Real-time data analysis allows for quick adaptation and innovation, speeding up product development and updates.
- Better Customer Experiences: Analysing structured and unstructured data helps understand and personalise customer experiences.
- Continuous Intelligence: Combining real-time data streaming with advanced analytics leads to continuous data collection and discovery of growth opportunities.
- Efficient Operations: Big data analytics tools process data faster, leading to cost reduction, time savings, and increased efficiency.
- Improved Risk Management: Analysing vast data helps evaluate and manage risks more effectively.

Challenges of implementing big data analytics include:

- Lack of Data Talent: A shortage of skilled data professionals, including data scientists, data analysts, and data engineers, can hinder the realisation of value from big data.
- Data Growth Speed: Big data continually grows and changes, requiring robust infrastructure to handle processing, storage, network, and security needs.
- Data Quality Issues: Raw data can be messy, impacting data accuracy and relevance. Proper organisation is essential.
- Compliance Concerns: Managing sensitive data while complying with data privacy and regulatory requirements can be complex.
- Integration Complexity: Integrating data from various sources across an organisation is challenging but necessary for deriving value from big data.
- Security Risks: Big data stores are valuable targets for attackers, and their complexity can make comprehensive security strategies more challenging to implement.

Distributed Systems

A distributed system is a network of interconnected computers or devices that work together to achieve a common goal or perform a set of coordinated tasks. In a distributed system, these individual devices, often referred to as nodes or processors, are located at different physical locations and communicate with each other through a computer network. The primary purpose of distributed systems is to harness the collective computing power and resources of multiple devices to solve complex problems, improve performance, enhance scalability, and increase fault tolerance. Key characteristics:

- Scalability
- Concurrent processing
- Fault tolerance
- Resource sharing
- Data distribution
- Communication
- Synchronisation
- Security
- Distributed computing models
- Transparency

How do you solve a problem like big data?

Before distributed systems such as Hadoop, traditional databases and data processing systems were unable to handle the massive volumes of data generated. These approaches had limitations in terms of scalability, cost effectiveness, and ability to handle various data types.

- Scale-Up: organisations relied on investing in processing power and increasing the performance of servers to handle greater workloads. Vertical scaling involves increasing CPU capacity
- Relational databases: used to store and manage structured data
- Parallel databases: used to distribute data across multiple nodes and process queries in parallel. These systems were limited in terms of scalability and expensive
- Data warehouse solutions were used to store large volume of data from sources and effective for data analytics
- Some organisations resorted to building custom data processing pipelines using programming languages like Java or C++. These solutions were tailored to specific needs but required significant development effort and were not always scalable or cost-effective.
- Data Silos: Data often ended up in isolated silos within organisations, making it challenging to access and analyse data comprehensively.
- Data sampling: organisations would resort to data sampling to reduce the datasets size which came at the cost of missing important insights and business value. Organisations would use data cleaning to remove noise or irrelevant data from datasets to make them more manageable for traditional systems.

Hadoop

Apache Hadoop is an open source, Java-based software platform that manages data processing and storage for big data applications. The platform works by distributing Hadoop big data and analytics jobs across nodes in a computing cluster, breaking them down into smaller workloads that can be run in parallel. Some key benefits of Hadoop are scalability, resilience and flexibility. The Hadoop Distributed File System (HDFS) provides reliability and resiliency by replicating any node of the cluster to the other nodes of the cluster to protect against hardware or software failures. Hadoop flexibility allows the storage of any data format including structured and unstructured data.

Apache Hadoop is an open-source software utility that allows users to manage big data sets (from gigabytes to petabytes) by enabling a network of computers (or “nodes”) to solve vast and intricate data problems. It is a highly scalable, cost-effective solution that stores and processes structured, semi-structured and unstructured data (e.g., Internet clickstream records, web server logs, IoT sensor data, etc.).

Benefits of the Hadoop framework include the following:

- Data protection amid a hardware failure
- Vast scalability from a single server to thousands of machines
- Real-time analytics for historical analyses and decision-making processes

When and why was Hadoop created?

Hadoop was designed to address the downsides of distributed computing, and handles fault tolerance and programming complexity for us.

Hadoop, an open-source big data processing platform, was created in 2002 by Doug Cutting and Mike Cafarella to handle large volumes of web data efficiently. It was inspired by Google's MapReduce and emerged from the Apache Nutch project. Hadoop democratised data processing, offering scalability, cost-effectiveness, and flexibility. Yahoo released it as an open-source project in 2008, and it became Apache Hadoop in 2012. It revolutionised big data analytics, enabling the processing of massive datasets on commodity hardware.

The primary motivation behind the creation of Hadoop was to address the challenges of processing and analysing massive datasets. Hadoop was designed to provide a cost-effective and scalable solution for storing and processing such data.

Hadoop was designed to address several downsides of distributed computing and to provide solutions for challenges in this domain:

Fault Tolerance: Hadoop's HDFS uses data replication to combat hardware failures, network issues, and unforeseen problems. Data blocks are replicated across nodes, ensuring availability and reliability. Automated failure detection and handling make it highly fault-tolerant.

Programming Complexity: Hadoop simplifies distributed computing for developers with the MapReduce model. It divides code into Map and Reduce functions, abstracting data distribution, parallel processing, and fault tolerance complexities. This allows developers to focus on data processing logic, not low-level details.

Scalability: Hadoop scales horizontally by adding commodity hardware to the cluster, accommodating growing data and processing needs without architectural changes.

Data Locality: Hadoop optimises data transfer by scheduling tasks on nodes where required data is already stored, reducing network overhead and enhancing processing efficiency.

Cost-Effectiveness: Hadoop's use of affordable commodity hardware makes distributed computing accessible to various organisations without massive upfront investments.

Ecosystem: Hadoop's rich ecosystem offers tools (e.g., Hive, Pig, HBase, Spark) for diverse data tasks, simplifying the development of complex data pipelines and analytics solutions.

Main components of Hadoop

HDFS (Hadoop Distributed File System): HDFS is the storage component of the Hadoop ecosystem. It is a Java-based distributed file system designed to store and manage very large data sets across multiple nodes in a Hadoop cluster.

Key Features:
Fault Tolerance: HDFS is fault-tolerant, meaning it can handle hardware failures without losing data. It replicates data across multiple nodes, so if one node fails, data can be retrieved from another.
Scalability: It can scale horizontally by adding more commodity hardware to the cluster, allowing it to handle massive datasets.
High Throughput: HDFS is optimised for high throughput, making it suitable for applications that require the processing of large files.

YARN (Yet Another Resource Negotiator): YARN is a resource management and job scheduling component in the Hadoop ecosystem. It allows for the efficient allocation of resources and manages the execution of tasks and jobs on a Hadoop cluster.

Key Features:
Resource Management: YARN manages and allocates cluster resources (CPU and memory) to various applications and jobs running on the cluster.
Job Scheduling: It schedules and prioritise tasks and jobs, ensuring that cluster resources are utilised optimally.
Support for Multi-Tenancy: YARN supports the running of multiple applications with different resource requirements on the same cluster simultaneously.

MapReduce: MapReduce is both a programming model and a data processing engine used in the Hadoop ecosystem for parallel processing of large datasets.

Key Features:
Parallel Processing: MapReduce divides a large task into smaller sub-tasks and processes them in parallel across the cluster nodes.
Scalability: It scales horizontally, making it suitable for processing vast amounts of data.
Simplicity: MapReduce abstracts the complexities of distributed computing, allowing developers to focus on writing map and reduce functions to process data.
Note: While MapReduce was the original data processing engine in Hadoop, Hadoop has added support for other engines like Apache Tez and Apache Spark, which offer improved performance and functionality

Hadoop Common: Hadoop Common is a set of libraries, utilities, and shared services that provide foundational support for the other Hadoop modules (HDFS, YARN, MapReduce, etc.).

Key Components:
Common Utilities: Hadoop Common includes common utilities and APIs used by different Hadoop components.
Configuration Management: It provides tools for configuring and managing various aspects of a Hadoop cluster.
Logging and Security: Hadoop Common includes logging facilities and security features that are essential for the secure and reliable operation of a Hadoop cluster.

Hadoop related software

- Apache Hive is data warehouse software that runs on Hadoop and enables users to work with data in HDFS using a SQL-like query language called HiveQL.
- Apache Impala is the open source, native analytic database for Apache Hadoop.
- Apache Pig is a tool that is generally used with Hadoop as an abstraction over MapReduce to analyse large sets of data represented as data flows. Pig enables operations like join, filter, sort, and load.
- Apache Zookeeper is a centralised service for enabling highly reliable distributed processing.
- Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.
- Apache Oozie is a workflow scheduler system to manage Apache Hadoop jobs. Oozie Workflow jobs are Directed Acyclic Graphs (DAGs) of actions.

Benefits and Challenges with Hadoop Architectures

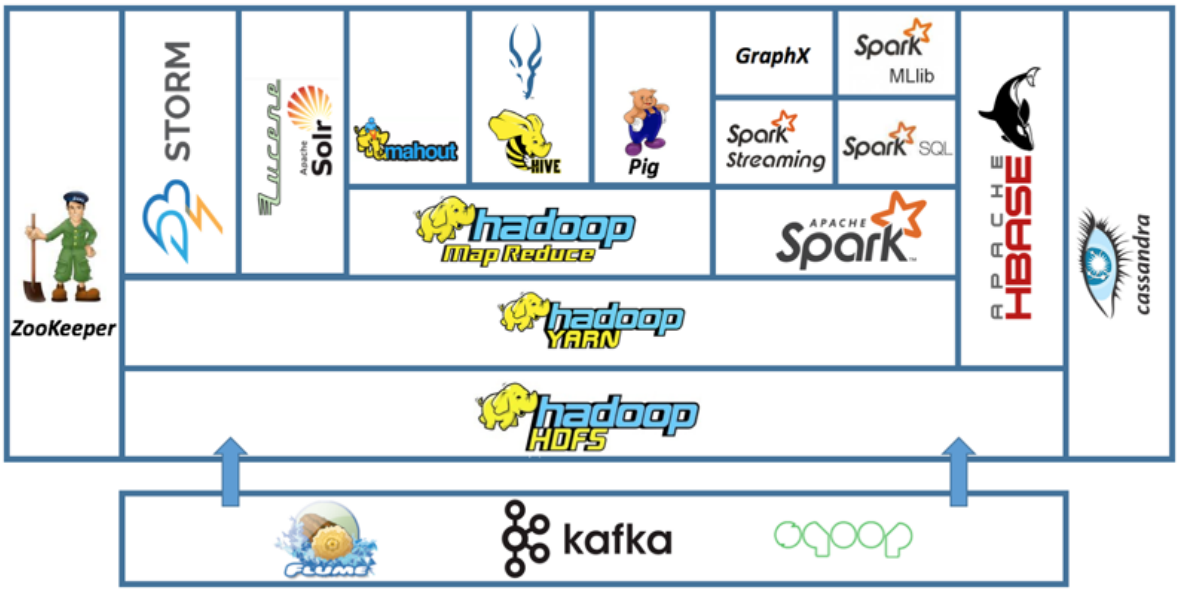
Benefits of Hadoop:

Scalability: Hadoop operates in a distributed environment, allowing for easy scalability and the creation of data lakes.
Resilience: Hadoop's HDFS replicates data across nodes, ensuring fault tolerance and data recovery in case of hardware or software failures.
Flexibility: Hadoop supports various data formats, including semi-structured and unstructured, enabling businesses to access diverse data sources.

Challenges with Hadoop Architectures:

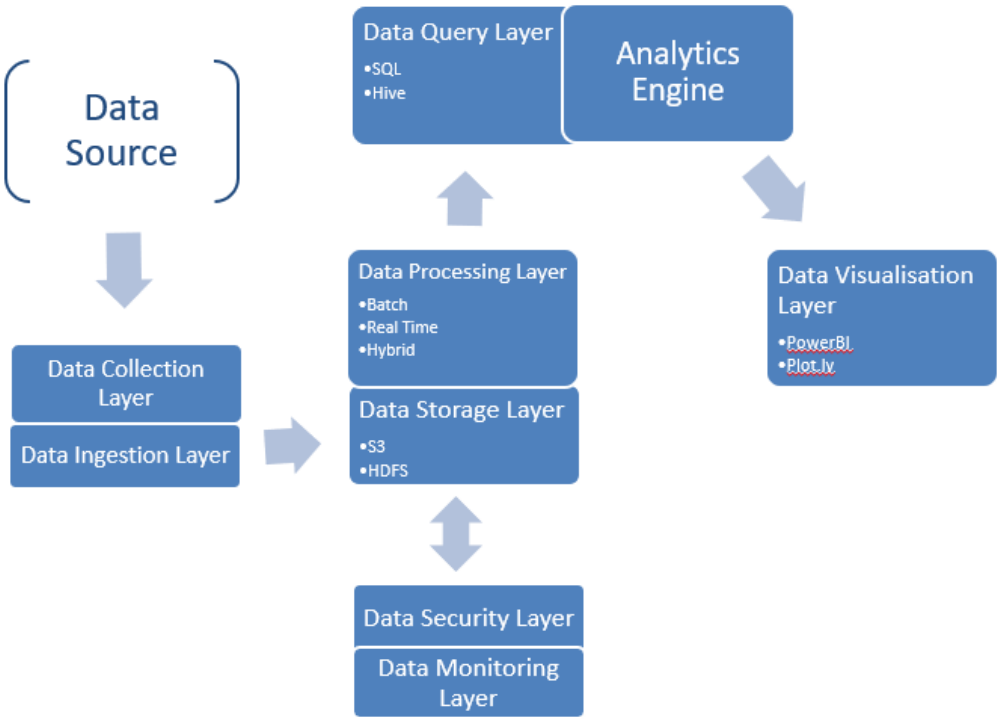
Complexity: Hadoop is low-level and Java-based, making it complex for end-users and requiring significant expertise and resources.
Performance: Hadoop's disk-centric approach for computations can be time-consuming and less efficient compared to memory-centric frameworks like Apache Spark.
Long-term Viability: Concerns about Hadoop's future arose due to changes in the industry, such as Google's move away from MapReduce, mergers and acquisitions, and project retirements, leading companies to reconsider their Hadoop adoption.

Hadoop Ecosystem

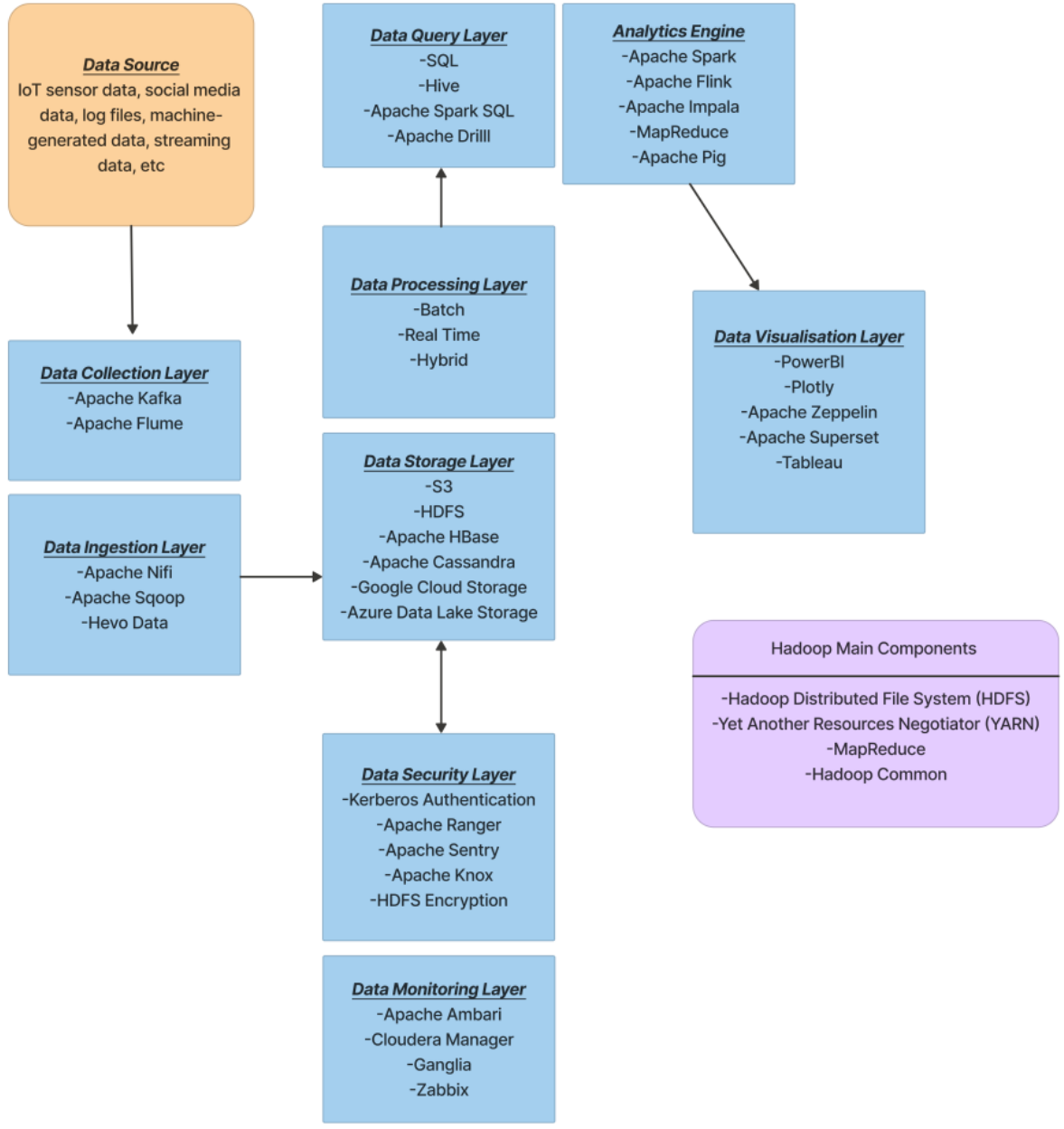


Task:

1. Research the above topics in more detail, creating notes on the different points.
2. Create a diagram similar to this one, with the main Hadoop technologies added on, some have been done for you as an example:


Data Source
Data Collection Layer
Data Ingestion Layer
Data Processing Layer
•Batch
•Real Time
•Hybrid
Data Storage Layer
•S3
•HDFS
Data Security Layer
Data Monitoring Layer
Data Query Layer
•SQL
•Hive
Analytics Engine
Data Visualisation Layer
•PowerBI
•Plot.ly

Big Data Technology Ecosystem Diagram



This diagram shows how various layers and technologies involved within Big Data fit together. Data Engineers will mostly be concerned with Collection, Ingestion, Storage and Processing. But, the other way of thinking about the job of the Data Engineer is that they're responsible for the arrows – the flow of data from one layer to the next.
Some tools will fit onto multiple different layers, think about what their primary purpose is, and if they do multiple things, you may need to account for that on the diagram.

Create a free community account with DataBricks by visiting:
https://databricks.com/try-databricks
Important: when it asks you what cloud provider to use (Amazon, Microsoft etc), there is a small grey link at the bottom that you can click to use the community edition of Data Bricks
Log in to DataBricks at: https://community.cloud.databricks.com

References

https://www.oracle.com/uk/big-data/what-is-big-data/#:~:text=is%20Big%20Data%3F-,Big%20data%20defined,especially%20from%20new%20data%20sources.
https://cloud.google.com/learn/what-is-big-data
https://hadoop.apache.org/
https://www.databricks.com/glossary/hadoop
https://www.ibm.com/blog/hadoop-vs-spark/
https://www.geeksforgeeks.org/5-vs-of-big-data/
https://www.atlassian.com/microservices/microservices-architecture/distributed-architecture