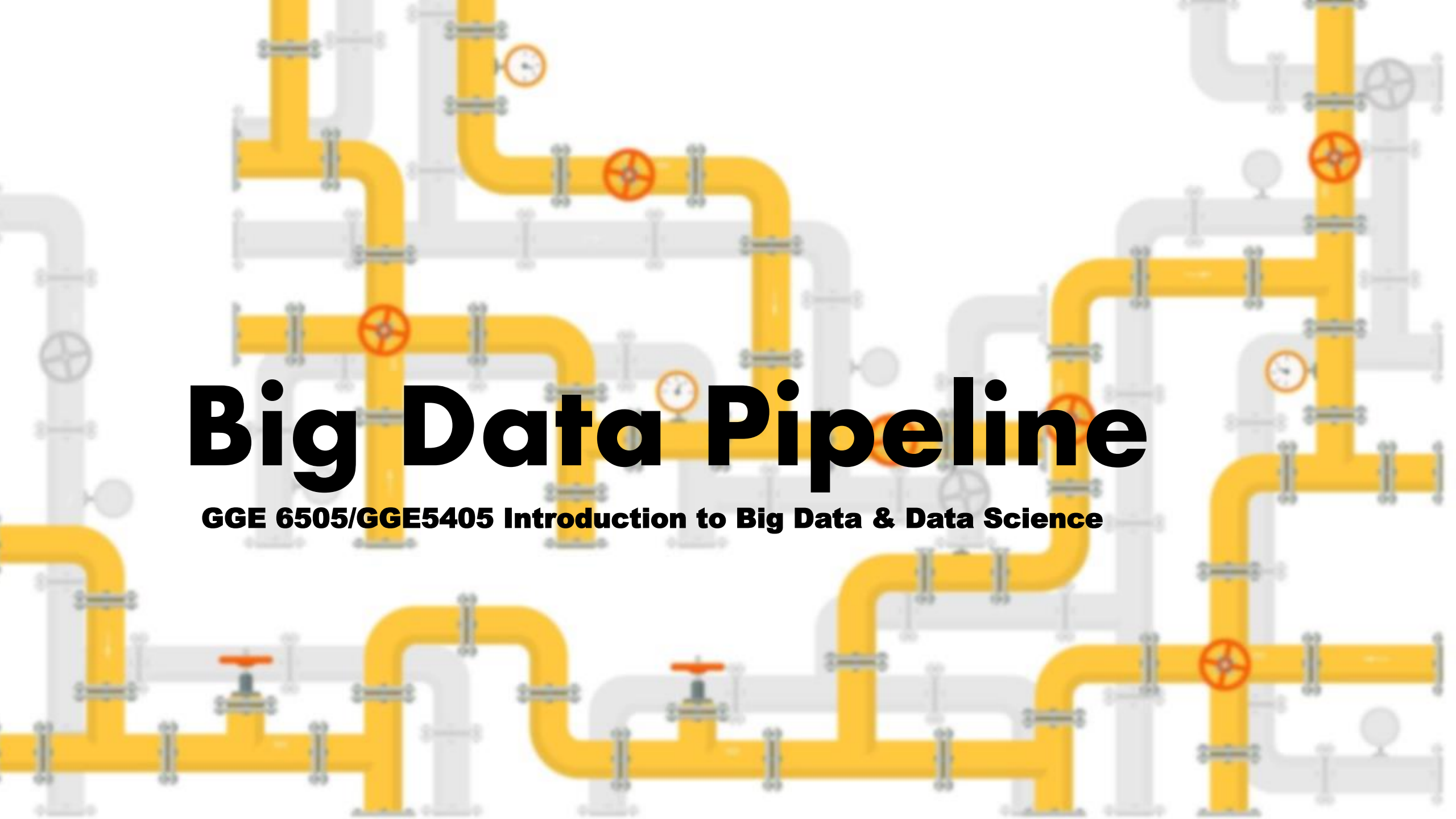


Big Data Pipeline

GGE 6505/GGE5405 Introduction to Big Data & Data Science

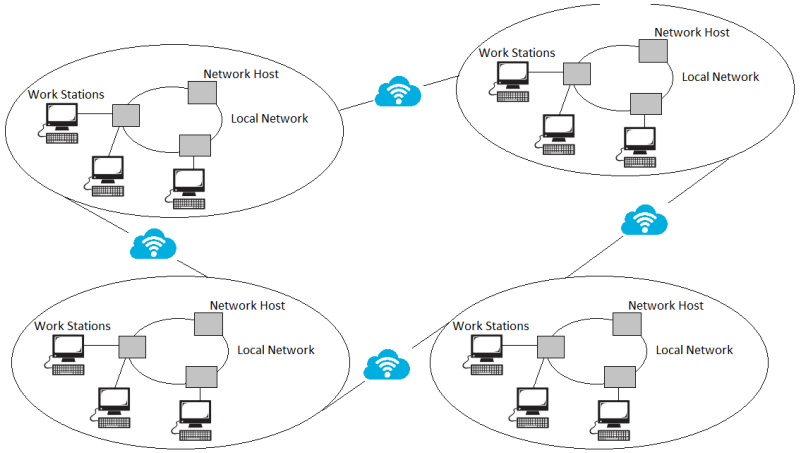
Big Data Concepts and Terminology

- Clustered computing
- Distributed computing
- Distributed file system
- ETL
- Data lake and data warehouse

1- Distributed Computing

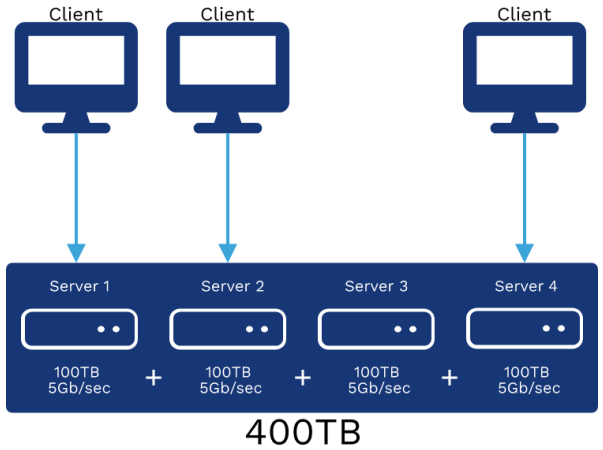
- A distributed computer system consists of multiple software components that are on multiple computers, but run as a single system. The computers that are in a distributed system can be physically close together and connected by a local network, or they can be geographically distant and connected by a wide area network. → collection of nodes that run parallelly!
- The goal of distributed computing is to make such a network work as a single computer.

- **Scalability**: The system can easily be expanded by adding more machines as needed.
- **Redundancy**: Several machines can provide the same services, so if one is unavailable, work does not stop.



2- Distributed File Systems (DFS)

- It is distributed on multiple file servers or multiple locations. It allows programs to access or store isolated files as they do with the local ones, allowing programmers to access files from any network or computer.
- **Applications :**
  - **NFS:**NFS stands for Network File System. It is a client-server architecture that allows a computer user to view, store, and update files remotely. The protocol of NFS is one of the several distributed file system standards for Network-Attached Storage (NAS).
  - **Hadoop:** Hadoop is a group of open-source software services. It gives a software framework for distributed storage and operating of big data using the MapReduce programming model. The core of Hadoop contains a storage part, known as Hadoop Distributed File System (HDFS), and an operating part which is a MapReduce programming model.



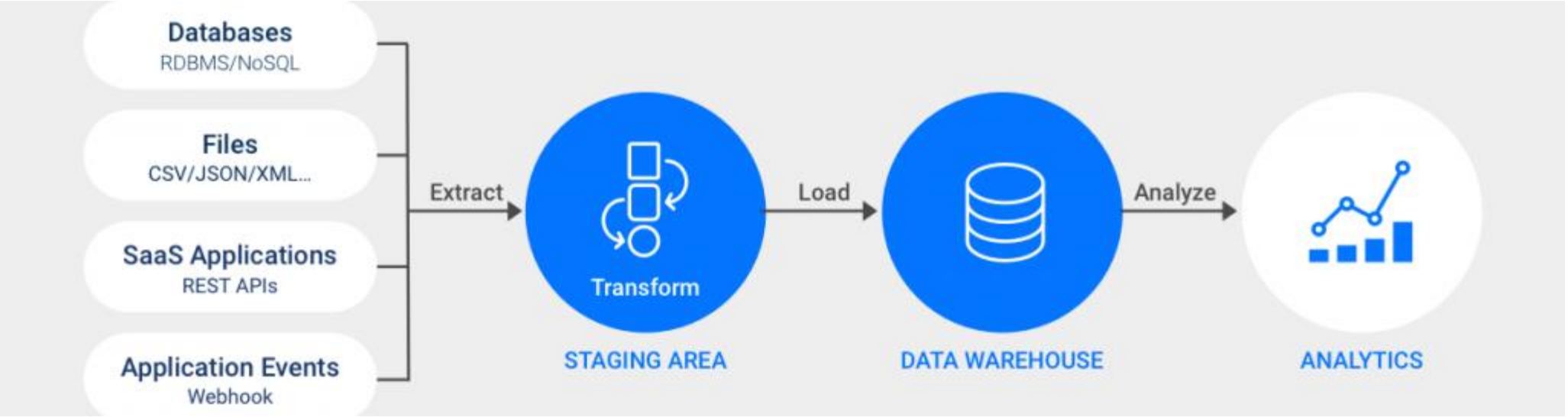
3- Data warehouse and Data lake

Data lakes and data warehouses are both widely used for storing big data, but they are not interchangeable terms. A data lake is a vast pool of raw data, the purpose for which is not yet defined. A data warehouse is a repository for structured, filtered data that has already been processed for a specific purpose.

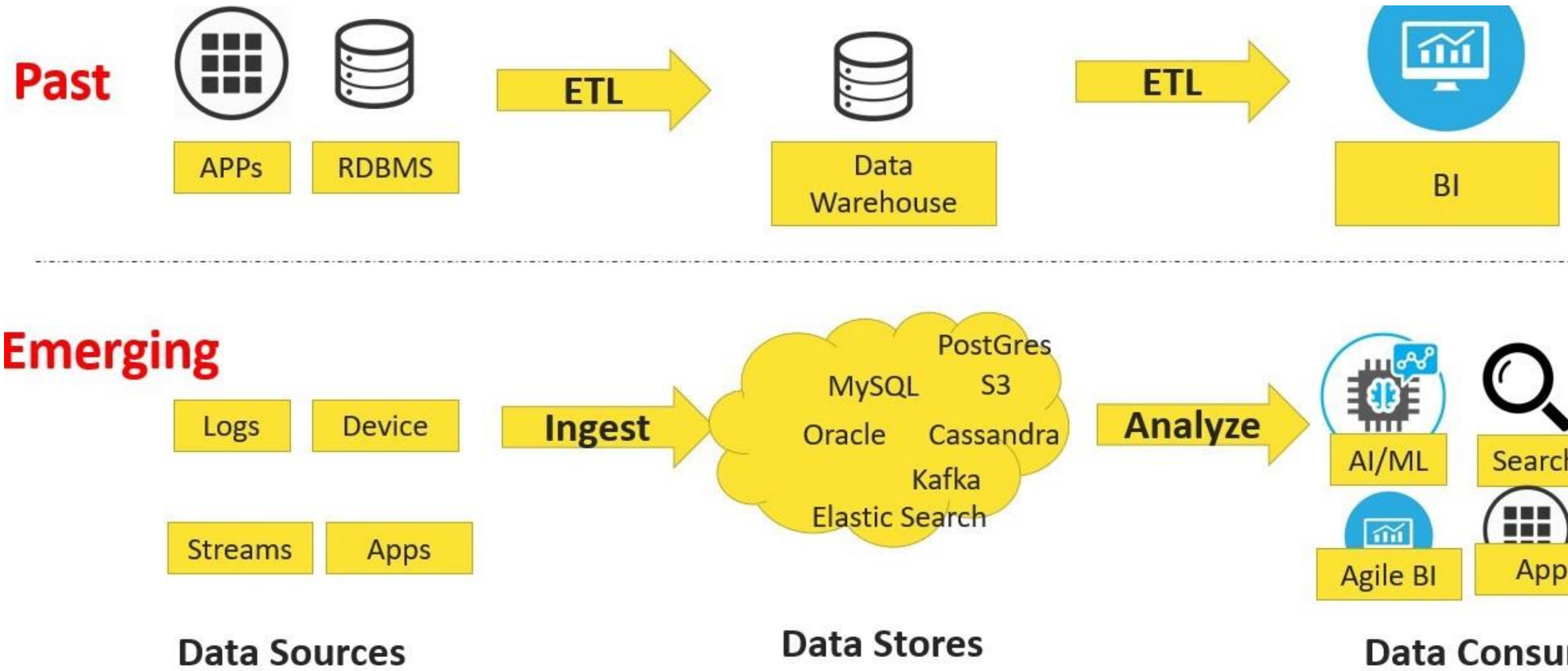
	Data Lake	Warehouse Data
Data Structure	Raw	Processed
Purpose of Data	Not Yet Determined	Currently In Use
Users	Data Scientists	Business Professionals
Accessibility	Highly accessible and quick to update	More complicated and costly to make changes

4- ETL

- ETL is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a target system.
- **EXTRACT** data from its original source
- **TRANSFORM** data by deduplicating it, combining it, and ensuring quality, to then
- **LOAD** data into the target database



Evolution of Data



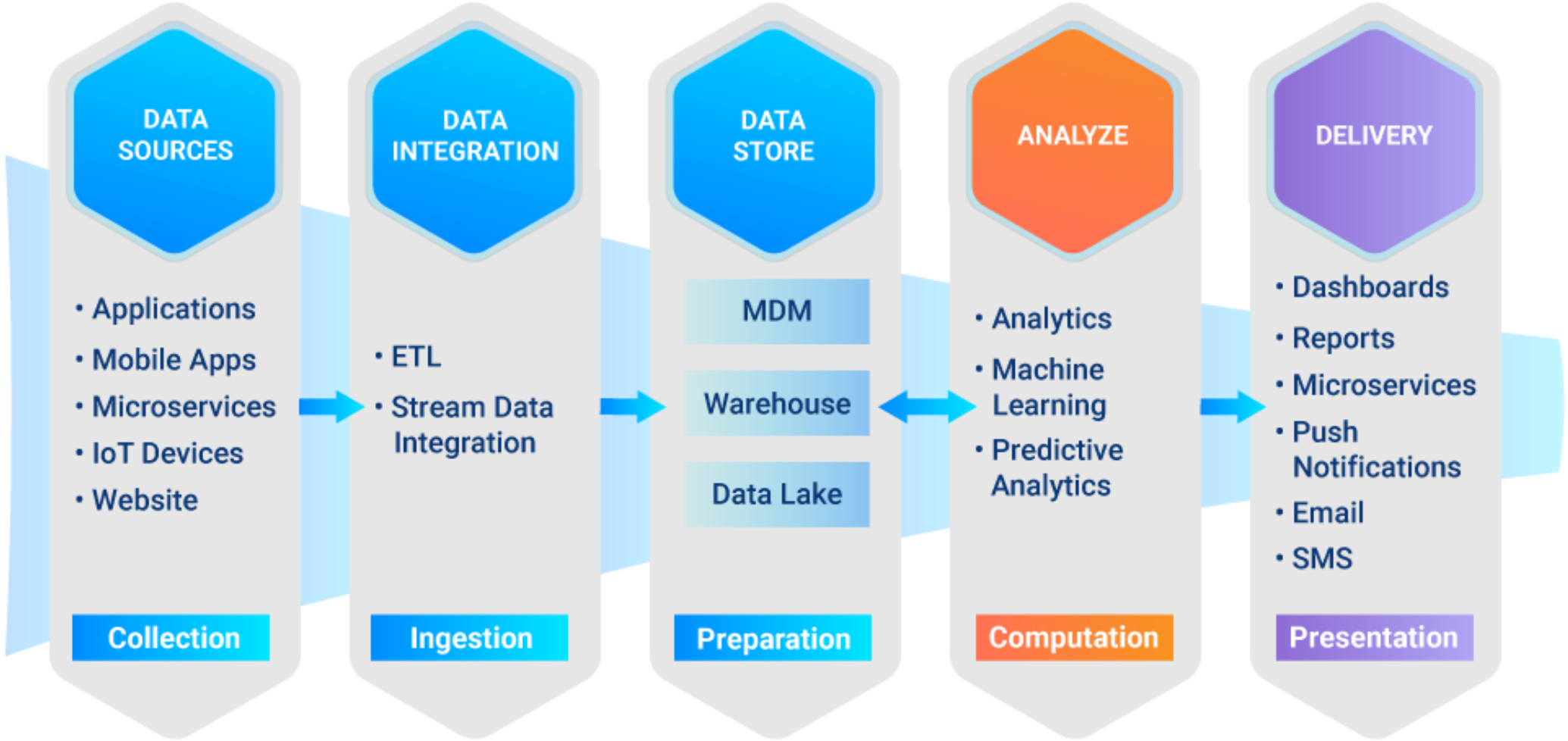
(93) Understanding the basics of any IoT data pipeline — from ingress capture to processing to generation of value (Part 1) | by Ravi Kumar. | Products, Platforms, Business & Innovation in Industry 4.0/IIoT | Medium

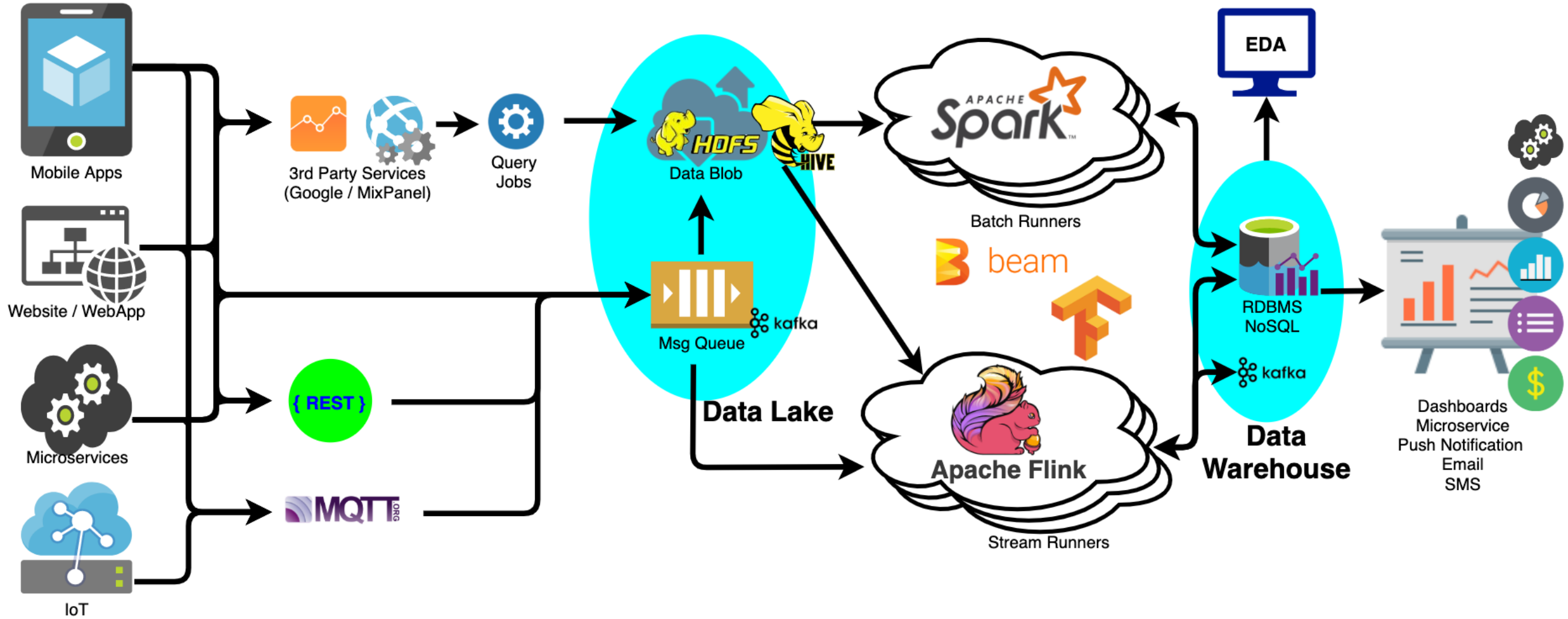
Big Data Pipeline

Big data pipelines are data pipelines built to accommodate one or more of the three traits of big data:

- The velocity of big data → makes it appealing to build streaming data pipelines for big data.
- The volume of big data → requires that data pipelines must be scalable, as the volume can be variable over time
- The variety of big data → requires that big data pipelines be able to recognize and process data in many different formats

Big Data Pipeline





Big Data Architecture

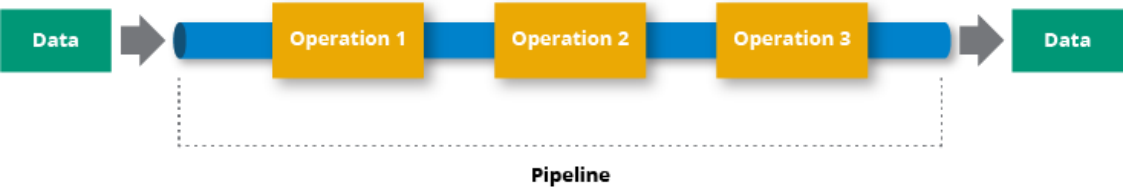
Data pipelines may be architected in several different ways.

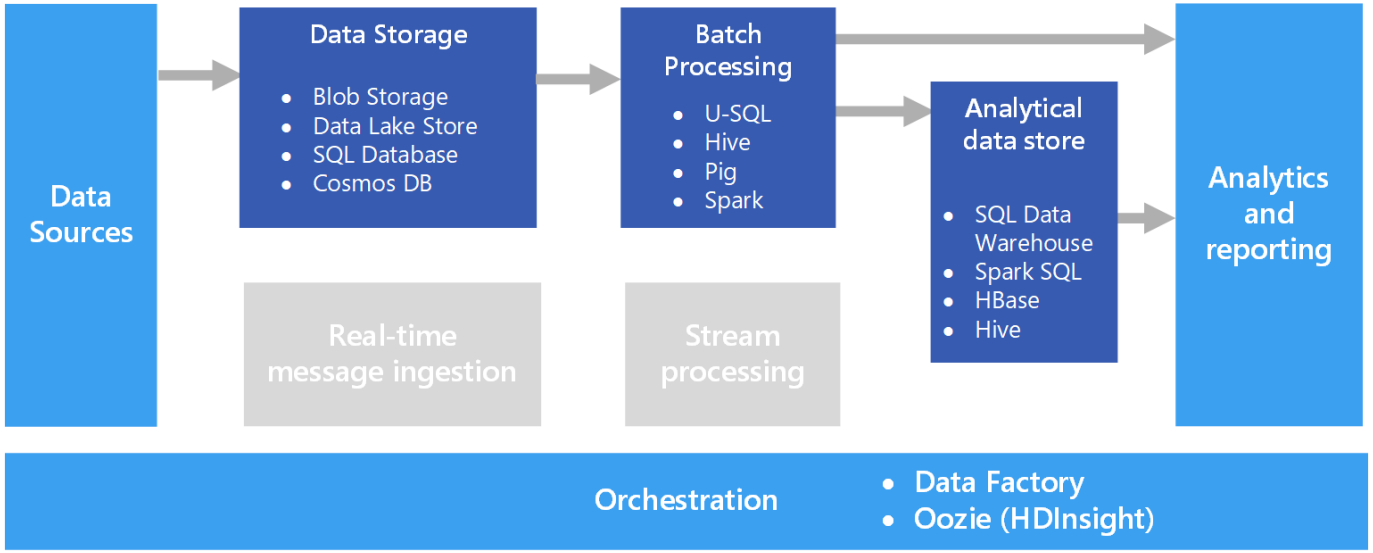
- Batch-based data pipeline
- Streaming data pipeline
- Lambda architecture

- What Is A Data Pipeline? Considerations & Examples - Hazelcast

Batch-based data Pipeline

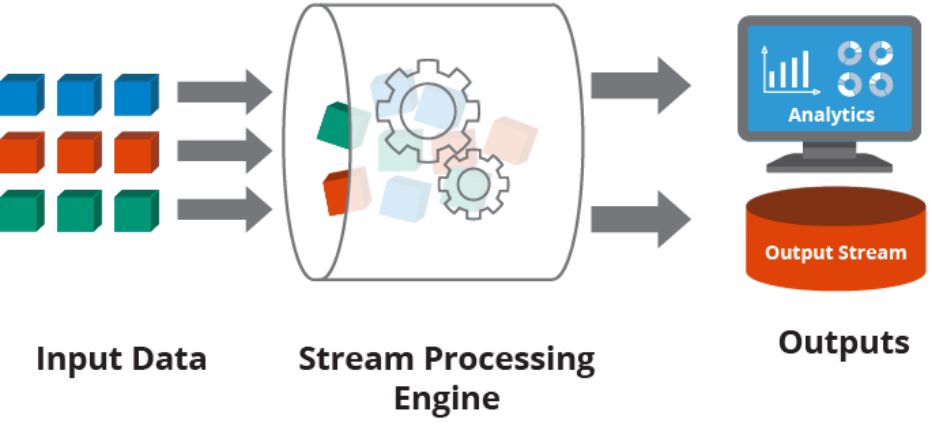
- Batch data pipelines are executed manually or recurringly. In each run, they extract all data from the data source, apply operations to the data, and publish the processed data to the data sink. They are done once all data have been processed.
- Typically batch data pipelines do not maintain knowledge about data changes.

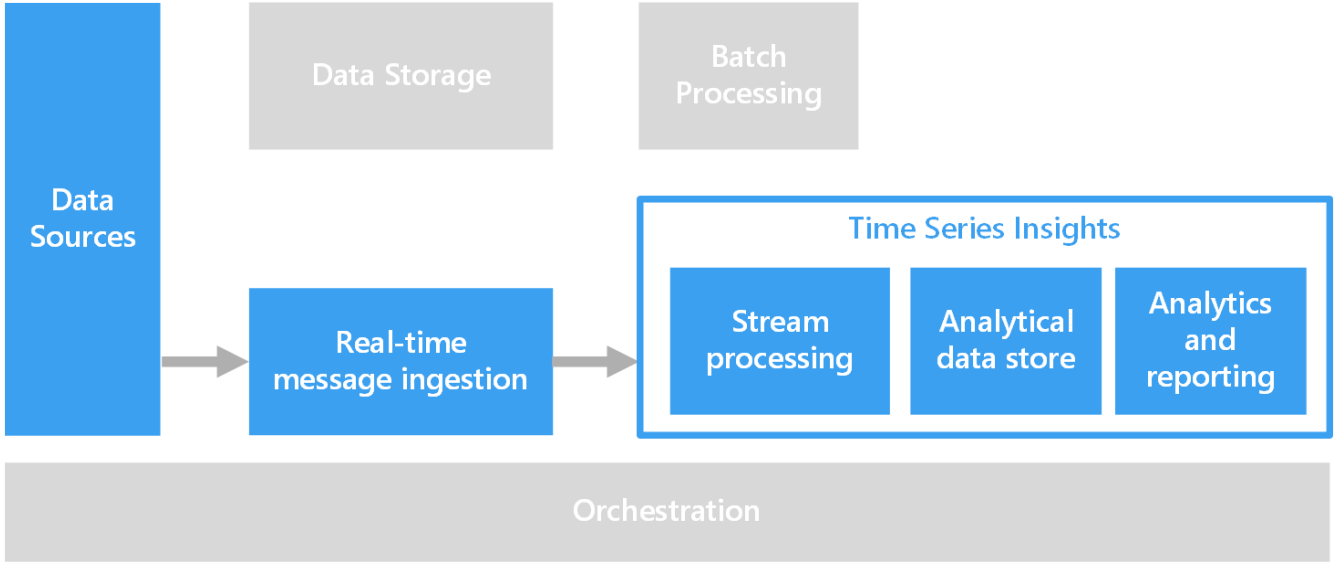




Streaming Data Pipeline

- Stream processing is the practice of taking action on a series of data at the time the data is created.
- Stream processing often entails multiple tasks on the incoming series of data (the "data stream"), which can be performed serially, in parallel, or both.
- This workflow is referred to as a stream processing pipeline.
- Stream processing is most often applied to data that is generated as a series of events, such as data from IoT sensors

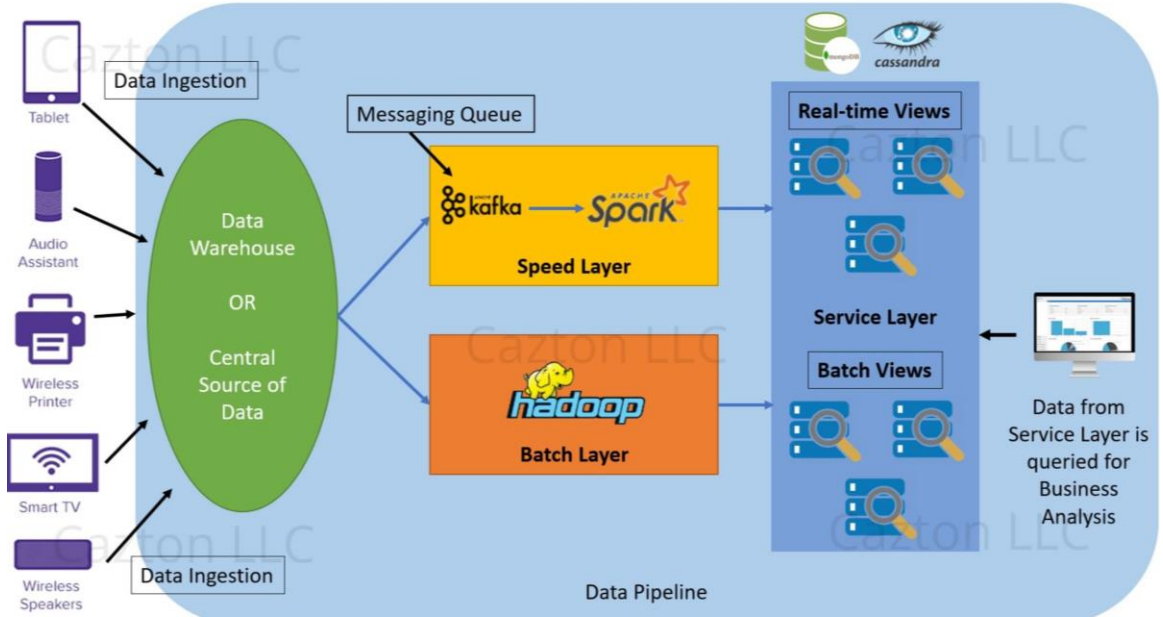


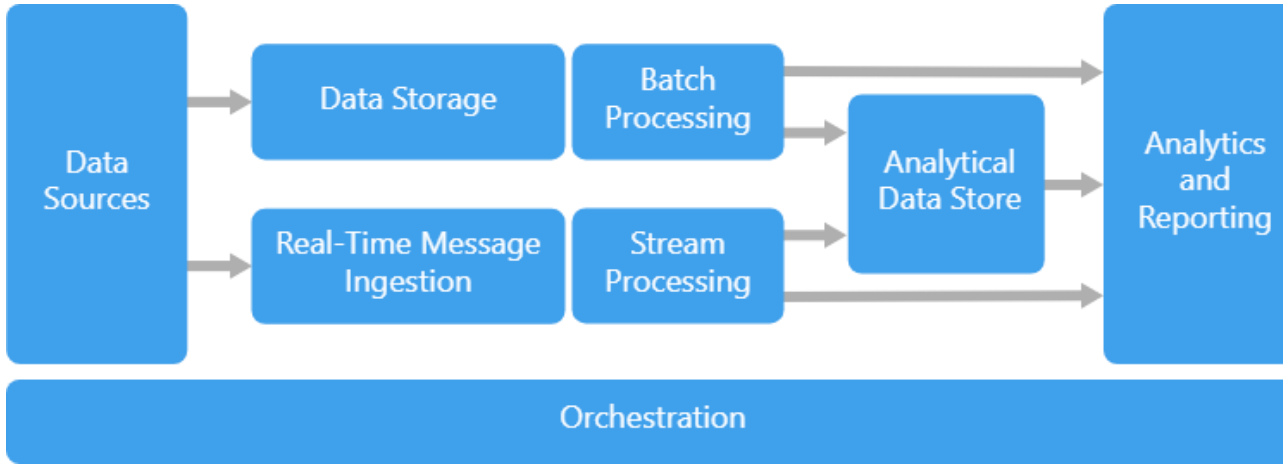


Lambda Architecture

This approach attempts to balance latency, throughput, and fault-tolerance by using batch processing to provide comprehensive and accurate views of batch data while simultaneously using real-time stream processing to provide views of online data.

Lambda architecture is a way of processing that provides access to batch-processing and stream-processing methods with a hybrid approach.





Top Big Data challenges

- Big Data Storage
- Big Data Processing

“How to process big data with reasonable cost and time?”

Doug Cutting

2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

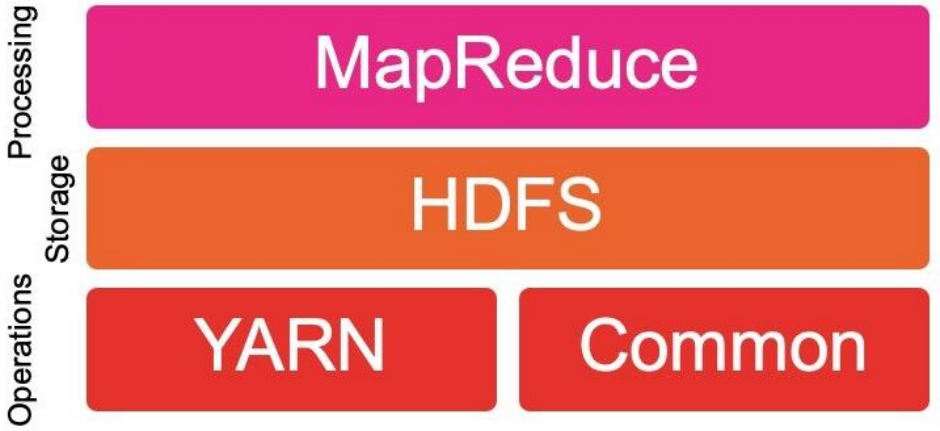
The project was funded by Yahoo.

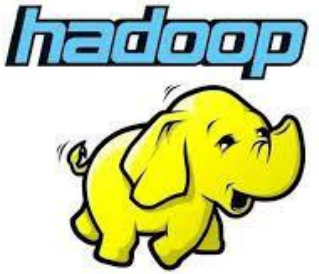
2006: Yahoo gave the project to Apache Software Foundation.

What is Hadoop?

- Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly(distributed computing).
- Structured and non-structured data

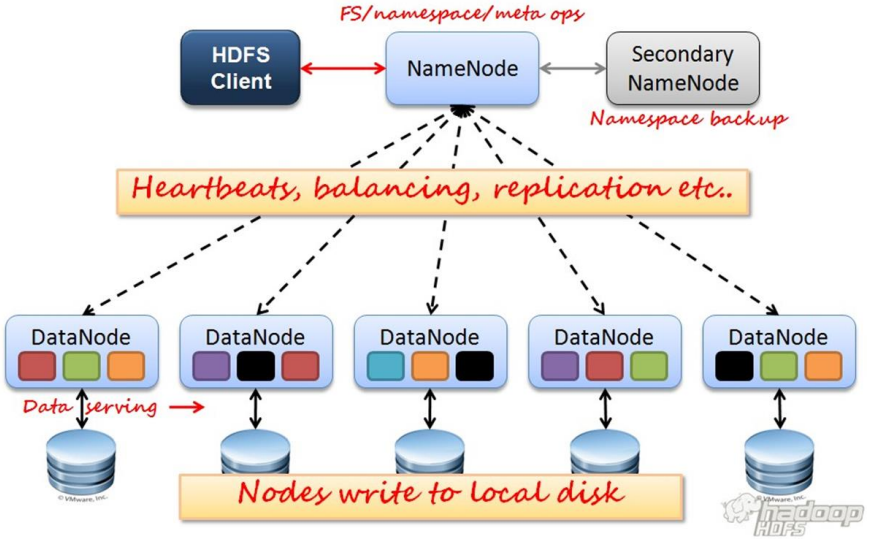
Hadoop consists of four main modules:





Hadoop Distributed File System (HDFS)

- Very Large Distributed File System
  - 10K nodes, 100 million files, 10PB
- Assumes Commodity Hardware
  - Files are replicated to handle hardware failure
  - Detect failures and recover from them
- Optimized for Batch Processing
  - Data locations exposed so that computations can move to where data resides
  - Provides very high aggregate bandwidth
- Integrate well with Hadoop MapReduce, allowing data to be read and computed upon locally when possible.

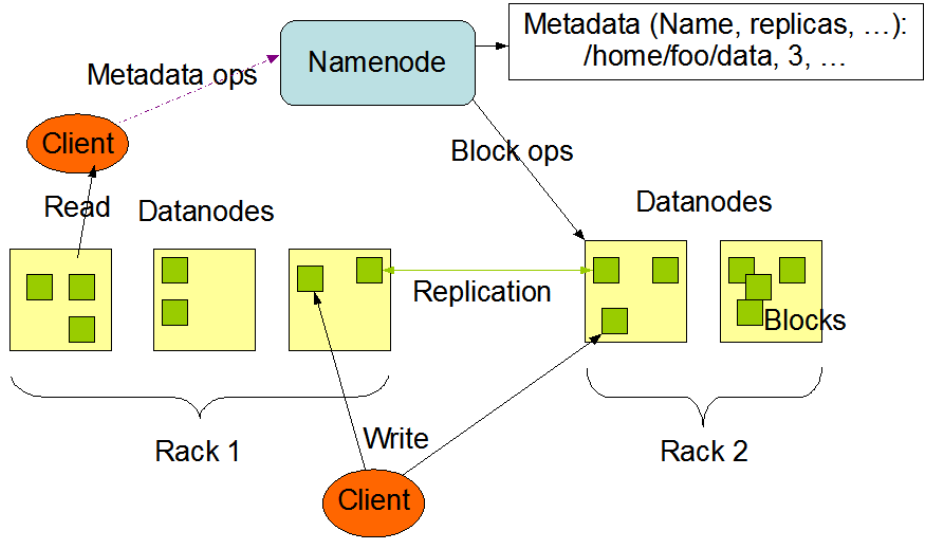


Hadoop Distributed File System (HDFS)

- HDFS has two core components:
  - Single NameNode : is the heart of an HDFS filesystem, it maintains and manages the file system metadata.
  - E.g; what blocks make up a file, and on which datanodes those blocks are stored.

- A number of DataNodes: where HDFS stores the actual data. Serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

- A file is split into one or more blocks and set of blocks are stored in DataNodes.

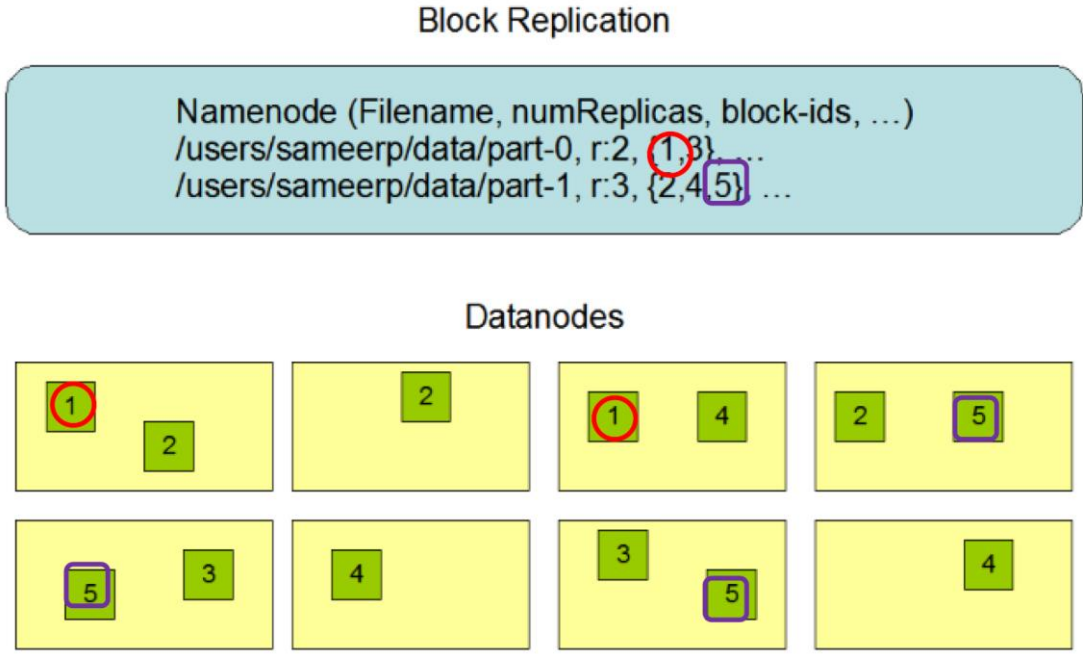


How HDFS stores files?

- Each file is stored as a sequence of blocks

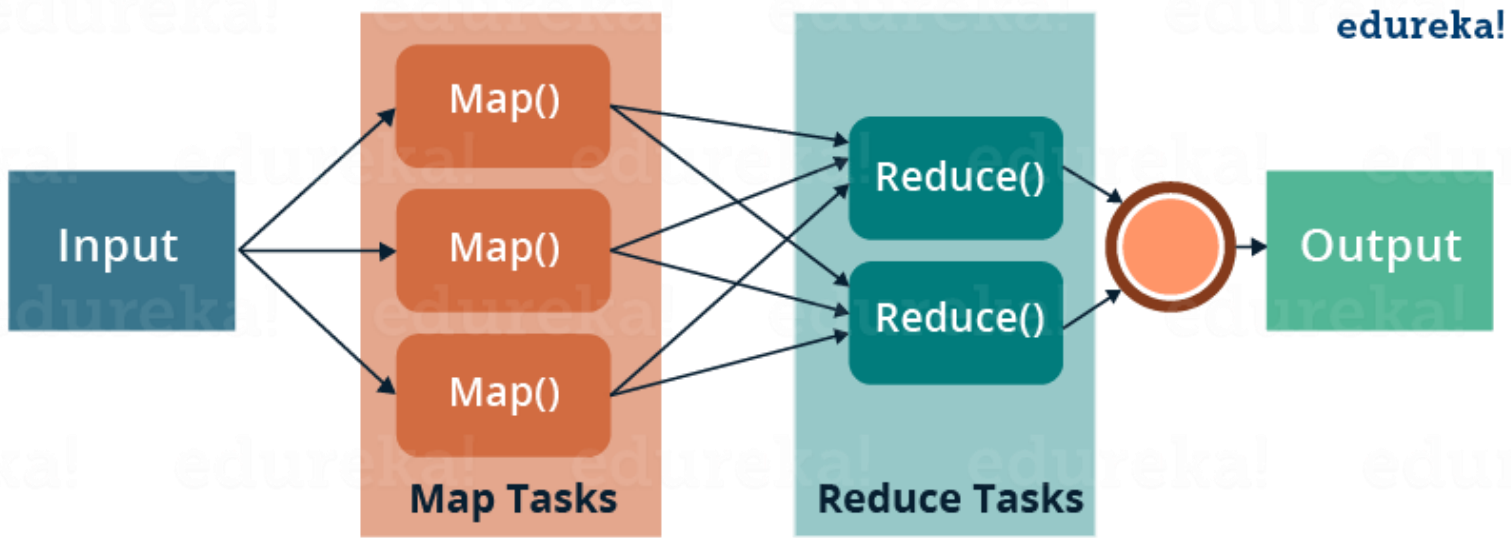
  Each block in the file except the last block are the same size.

- The blocks of a file are replicated for fault tolerance



MapReduce

- MapReduce is a programming model or pattern within the Hadoop framework that is used to access big data stored in the Hadoop File System (HDFS). It is a core component, integral to the functioning of the Hadoop framework.
- The **Map** function takes input from the disk as <key,value> pairs, processes them, and produces another set of intermediate <key,value> pairs as output.
- The **Reduce** function also takes inputs as <key,value> pairs, and produces <key,value> pairs as output.



……

Document 1

This is a cat

Cat sits on a roof

<this 1> <is 1> <a <1,1,>> <cat <1,1>> <sits 1> <on 1> <roof 1>

……

Document 2

The roof is a tin roof

There is a tin can on the roof

<the <1,1>> <roof <1,1,1>> <is <1,1>> <a <1,1>> <tin <1,1>> <then 1> <can 1> <on 1>

……

Document 3

Cat kicks the can

It rolls on the roof and falls on the next roof

<cat 1> <kicks 1> <the <1,1>> <can 1> <it 1> <roll 1> <on <1,1>> <roof <1,1>> <and 1> <falls 1> <next 1>

……

Document 4

The cat rolls too

It sits on the can

<the <1,1>> <cat 1> <rolls 1> <too 1> <it 1> <sits 1> <on 1> <cat 1>

<this 1> <is 1> <a <1,1,>> <cat <1,1>> <sits 1> <on 1> <roof 1>

<the <1,1>> <roof <1,1,1>> <is <1,1>> <a <1,1>> <tin <1,1>> <then 1> <can 1> <on 1>

<cat 1> <kicks 1> <the <1,1>> <can 1> <it 1> <roll 1> <on <1,1>> <roof <1,1>> <and 1> <falls 1> <next 1>

<the <1,1>> <cat 1> <rolls 1> <too 1> <it 1> <sits 1> <on 1> <cat 1>

.....

Combine the counts of all the same words:

<cat <1,1,1,1>>

<roof <1,1,1,1,1,1>>

<can <1, 1,1>>

.....

Reduce (sum in this case) the counts:

<cat 4>

<can 3>

<roof 6>

YARN

- The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons. The idea is to have a global Resource Manager

  - Resource Manager containers are allocations of physical resources
  - Application Manager: monitoring the application progress
  - Node Manager: takes care of individual nodes
  - Scheduler: schedules the tasks.

What is Spark?

- Apache Spark is an open-source tool
- This framework can run in a standalone mode or on a cloud or cluster manager, and other platforms.
- It is designed for fast performance and uses RAM for caching and processing data.

- The Spark engine was created to improve the efficiency of MapReduce and keep its benefits. Even though Spark does not have its file system, it can access data on many different storage solutions. The data structure that Spark uses is called Resilient Distributed Dataset, or RDD.

- Can absorb live data streams from Kafka, Flume, ZeroMQ, etc.

- Resilient Distributed datasets(RDD): is the fundamental data structure of Spark

Apache Spark Ecosystem

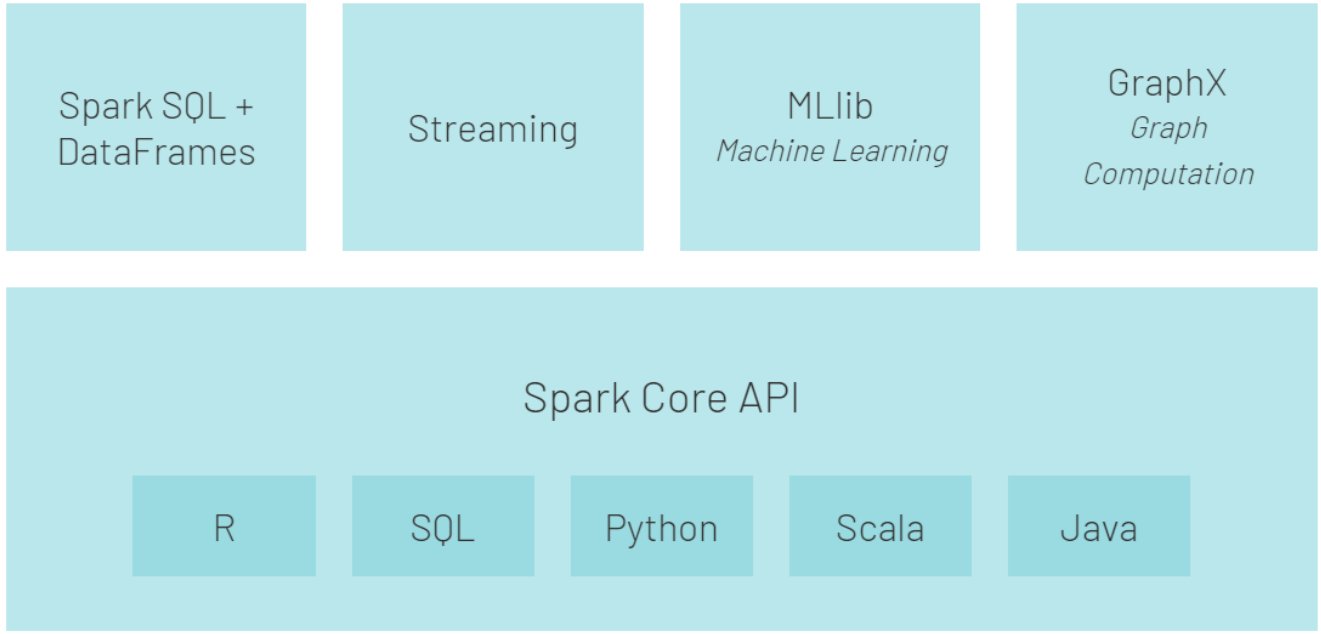
1.Spark Core

2.Spark SQL

3.Spark Streaming

4.MLlib(Machine learning lib)

5.GraphX



Hadoop vs. Spark

Performance

- By accessing the data stored locally on HDFS, Hadoop boosts the overall performance. However, it is not a match for Spark's in-memory processing.
- Spark appears to be 100x faster when using RAM for computing than Hadoop with MapReduce.

Cost

- Both platforms are open-source and completely free.
- Since Hadoop relies on any type of disk storage for data processing, the cost of running it is relatively low.
- On the other hand, Spark depends on in-memory computations for real-time data processing.
- Hadoop infrastructure is more cost-effective.

Data Processing

- Hadoop is more suitable for batch processing. In contrast, Spark shines with real-time processing.
- Hadoop's goal is to store data on disks and then analyze it in parallel in batches across a distributed environment.
- With the in-memory computations and high-level APIs, Spark effectively handles live streams of unstructured data.

Scalability

- Hadoop uses HDFS to deal with big data.
- Spark has to rely on HDFS when data is too large to handle.
- 8000 machines in a Spark environment with petabytes of data. When speaking of Hadoop clusters, they are well known to accommodate tens of thousands of machines and close to an exabyte of data.

Hadoop vs Spark: Detailed Comparison of Big Data Frameworks (phoenixnap.com)c

Hadoop vs. Spark

Fault Tolerance

- both provide a respectable level of handling failures.
- Hadoop has fault tolerance as the basis of its operation.
- Spark uses RDD blocks to achieve fault tolerance.

Security

- Hadoop is the clear winner. Hadoop works with multiple authentication and access control methods.
- You can improve the security of Spark by introducing authentication via shared secret or event logging.

Machine Learning

- Machine learning is an iterative process that works best by using in-memory computing. For this reason, Spark proved to be a faster solution in this area.
- The reason for this is that Hadoop MapReduce splits jobs into parallel tasks that may be too large for machine-learning algorithms

Ease of Use and Programming Language Support

- Spark provides support for multiple languages next to the native language (Scala): Java, Python, R, and Spark SQL.
- The Hadoop framework is based on Java. The two main languages for writing MapReduce code is Java or Python
- Spark wins in the ease-of-use section

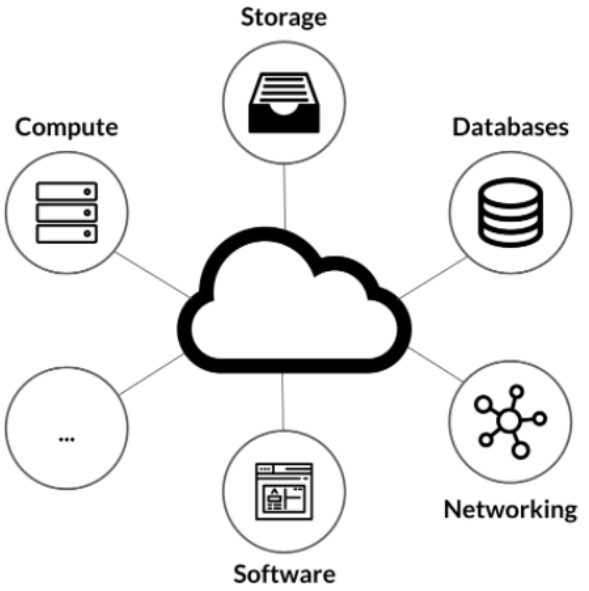
Hadoop vs Spark: Detailed Comparison of Big Data Frameworks (phoenixnap.com)


Big Data
Cloud
Computing

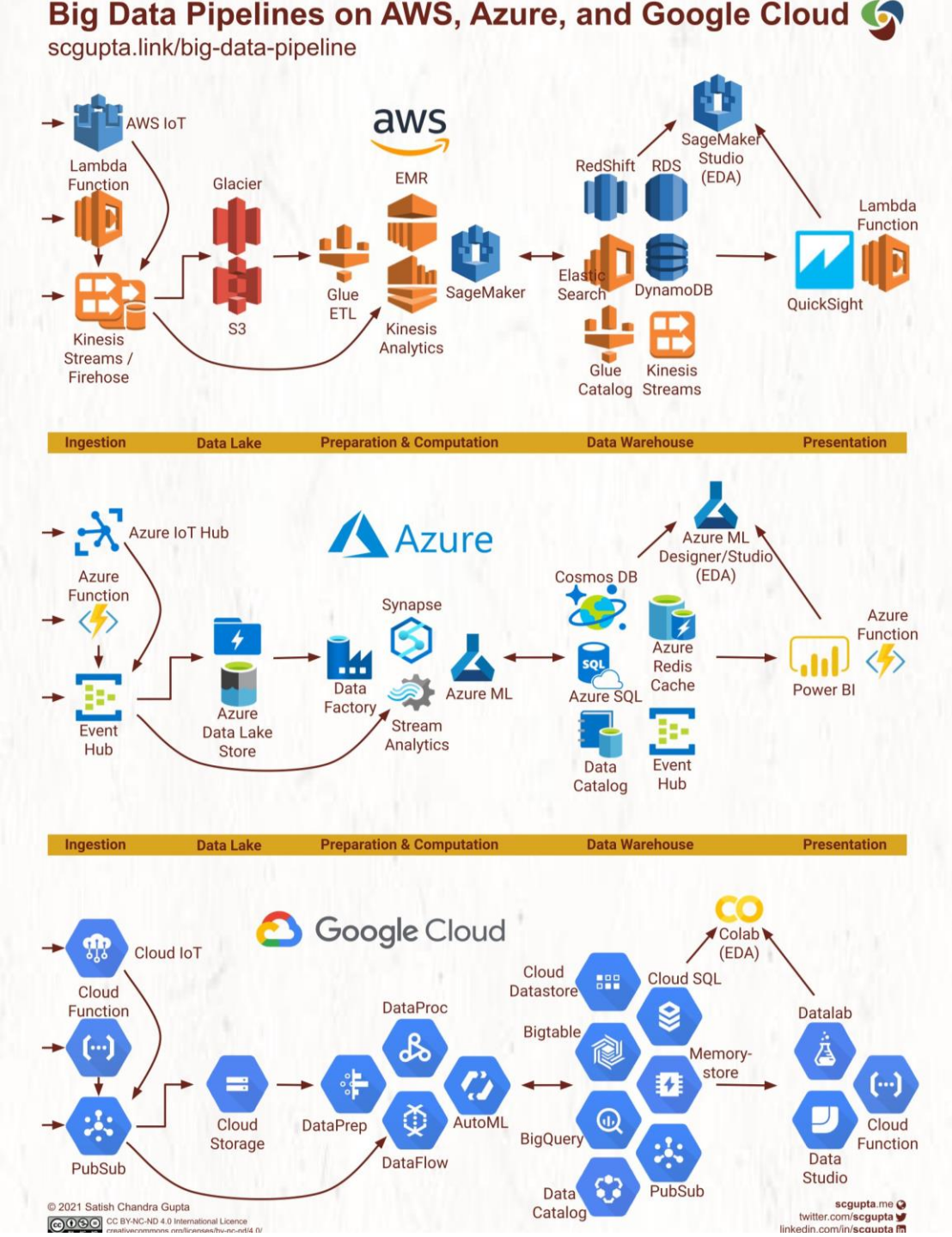
Cloud Computing

- Simply put, cloud computing is the delivery of computing services including servers, storage, databases, networking, software, analytics, and intelligence over the Internet (“the cloud”) to offer faster innovation, flexible resources, and economies of scale.
- Cloud can be public, private or hybrid.

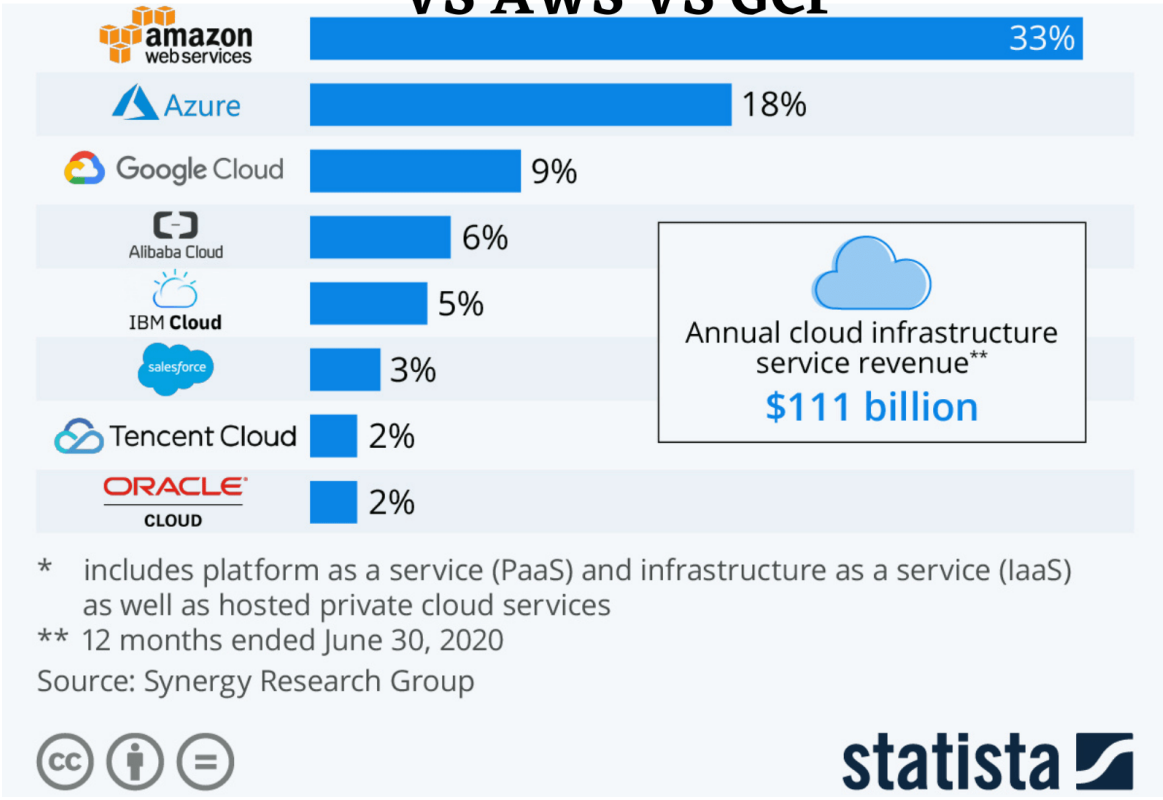
- Types of cloud services:
  - Infrastructure as a service (IaaS)
  - Platform as a service (PaaS)
  - Software as a service (SaaS)



[Types of Cloud Computing (crucial.com.au)]



Ultimate Cloud comparison 2020 Azure VS AWS VS GCP



* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

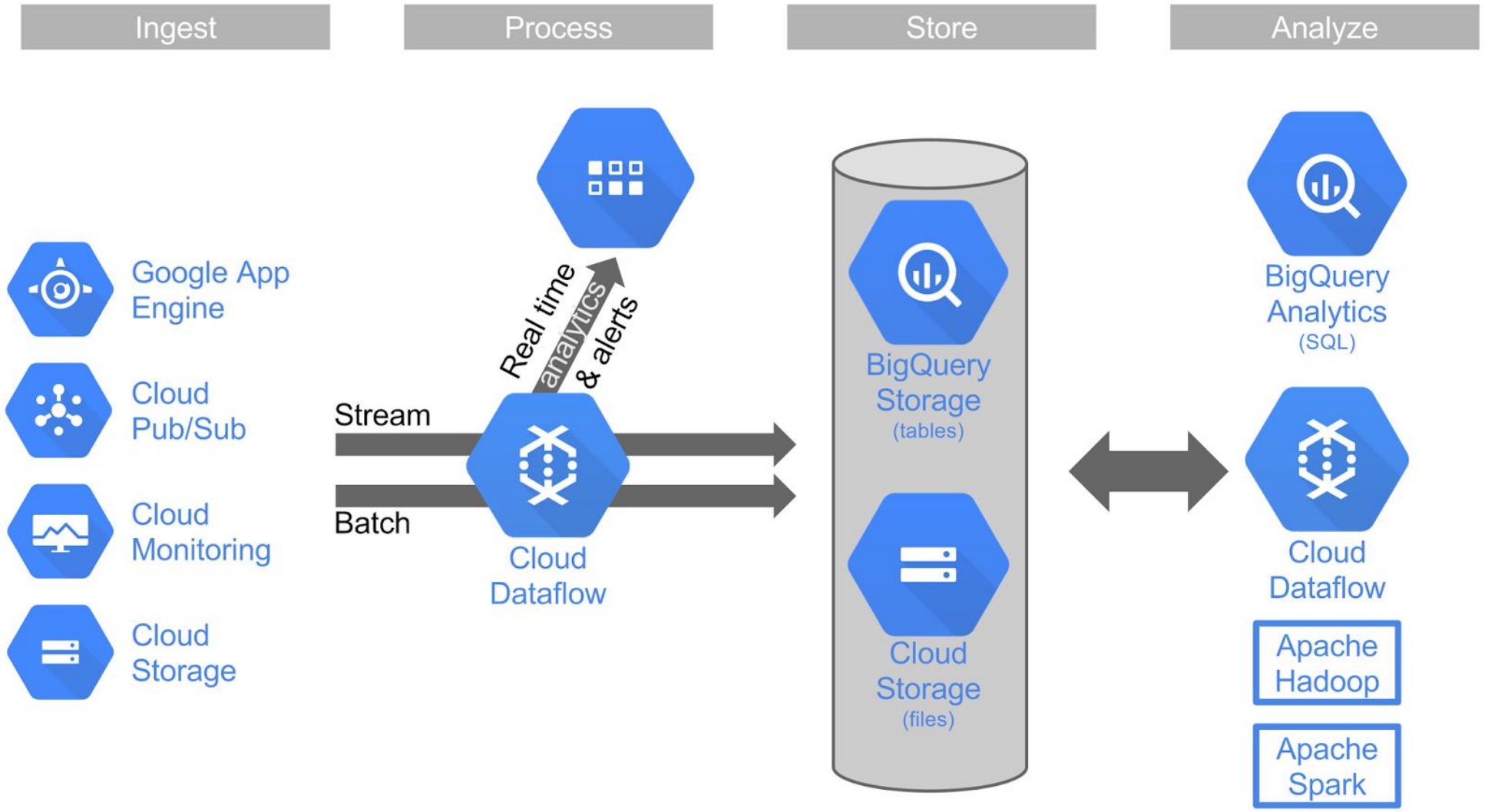
** 12 months ended June 30, 2020

Source: Synergy Research Group

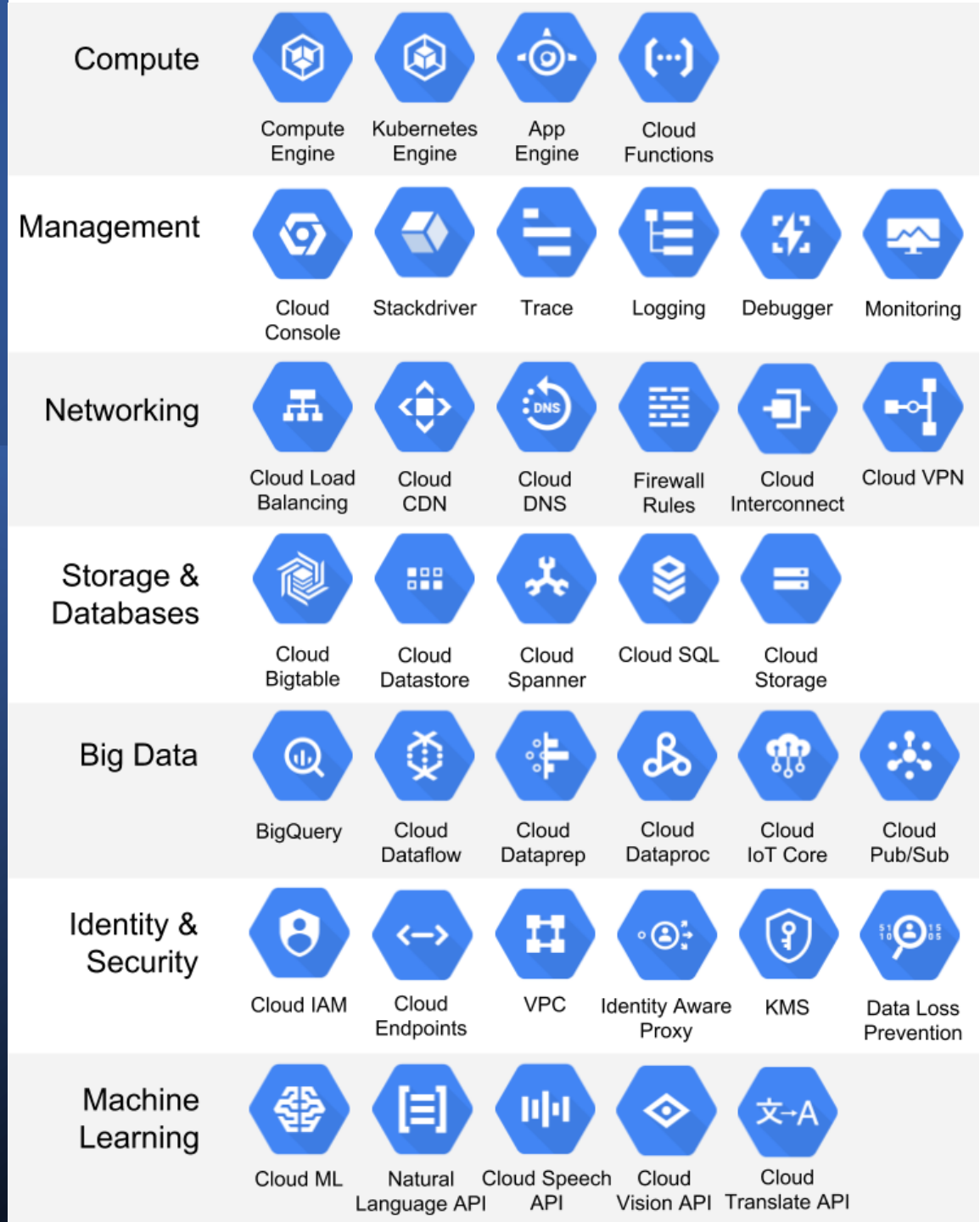
statista

PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

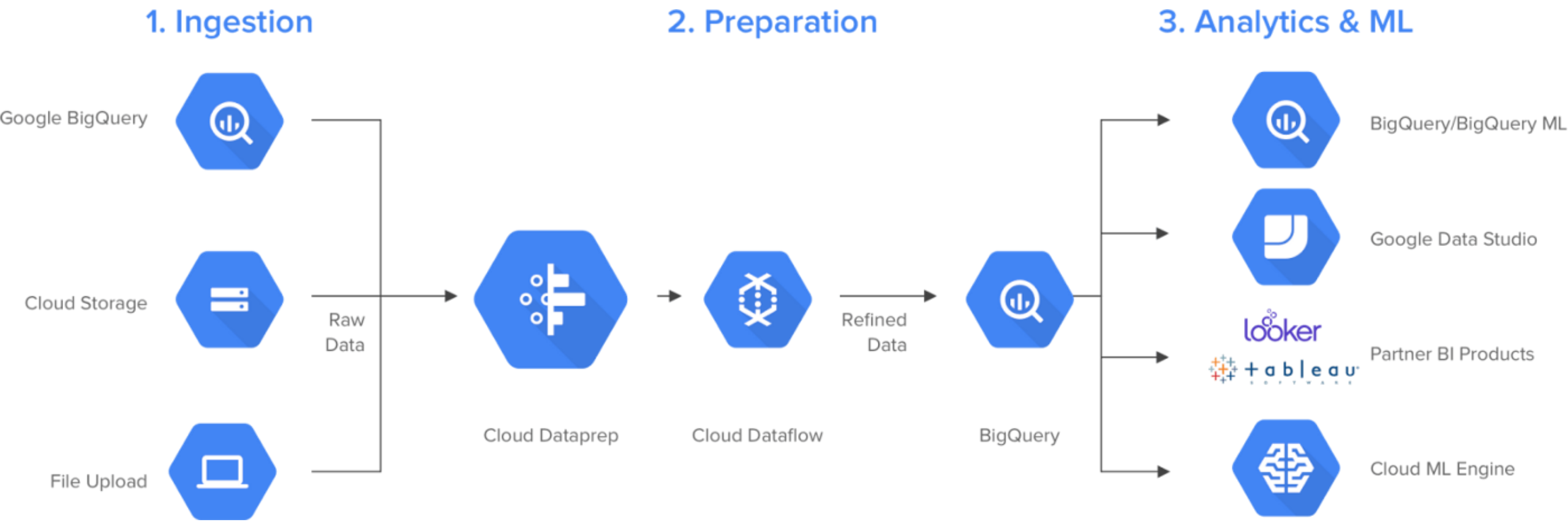
Google Could Platform (GCP)



GCP Services



GCP Big Data Pipeline



```
val staticData = sparkSession.read          // read() returns type DataFrameReader
    .format("csv")
    .option("header", "true")
    .load("s3://awsexamplebucket-streaming-demo2/inputs/productsStatic.csv")  // load()
turns a DataFrame

  val datasource0 = sparkSession.readStream   // readstream()        ype DataStreamReader
    .format("kinesis")
    .option("streamName", "stream-join-demo")
    .option("endpointUrl", "https://kinesis.us-east-1.amazonaw
    .option("startingPosition", "TRIM_HORIZON")
    .load
```

End….