UNIT-IV

BIG DATA:

The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity. This is also known as the three Vs. Put simply, big data is larger, more complex data sets, especially from new data sources.

Big data comes from myriad sources -- some examples are transaction processing systems, customer databases, documents, emails, medical records, internet clickstream logs, mobile apps and social networks.

Big data primarily refers to data sets that are too large or complex to be dealt with by traditional data-processing application software. Data with many entries offer greater statistical power, while data with higher complexity may lead to a higher false discovery rate.

The classification of big data is divided into three parts, such as

Structured Data,

Unstructured Data,

Semi-Structured Data.

Big data is a collection of data from many different sources and is often describe by five characteristics:

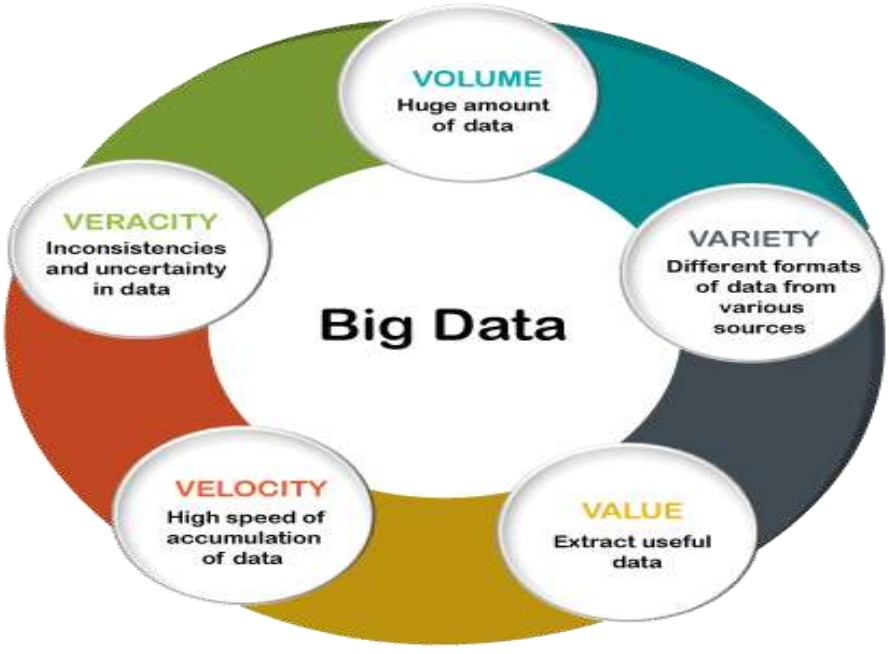
Volume,

Value,

Variety,

Velocity,

Veracity.



Big Data powers the GPS smartphone applications most of us depend on to get from place to place in the least amount of time. GPS data sources include satellite images and government agencies. Airplanes generate enormous volumes of data, on the order of 1,000 gigabytes for transatlantic flights.

The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity. This is also known as the three Vs. Put simply, big data is larger, more complex data sets, especially from new data sources.

HADOOP FILE SYSTEM:

HDFS (Hadoop Distributed File System) is a unique design that provides storage for extremely large files with streaming data access pattern and it runs on commodity hardware. Let's elaborate the terms:

Extremely large files: Here we are talking about the data in range of petabytes (1000 TB).

Streaming Data Access Pattern: HDFS is designed on principle of write-once and read-many-times. Once data is written large portions of dataset can be processed any number times.

Commodity hardware: Hardware that is inexpensive and easily available in the market. This is one of feature which specially distinguishes HDFS from other file system.

NODES: Master-slave nodes typically forms the HDFS cluster.

Name Node (Master Node): Manages all the slave nodes and assign work to them.

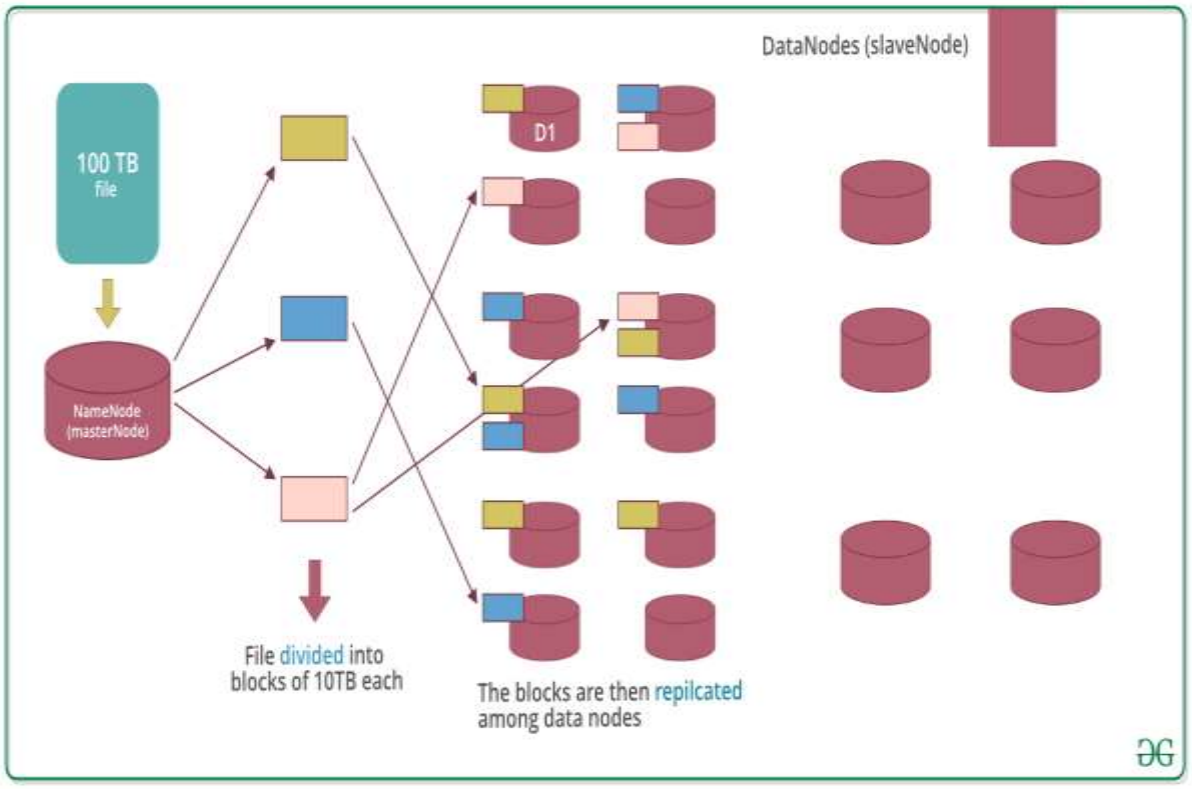
It executes file system namespace operations like opening, closing, renaming files and directories. It should be deployed on reliable hardware which has the high config. not on commodity hardware.

Data Node (Slave Node): Actual worker nodes, who do the actual work like reading, writing, processing etc. They also perform creation, deletion, and replication upon instruction from the master. They can be deployed on commodity hardware.

HDFS daemons: Daemons are the processes running in background.

Name nodes: Run on the master node. Store metadata (data about data) like file path, the number of blocks, block Ids. Etc. Require high amount of RAM. Store meta-data in RAM for fast retrieval i.e. to reduce seek time. Though a persistent copy of it is kept on disk.

Data Nodes: Run on slave nodes. Require high memory as data is actually stored here.

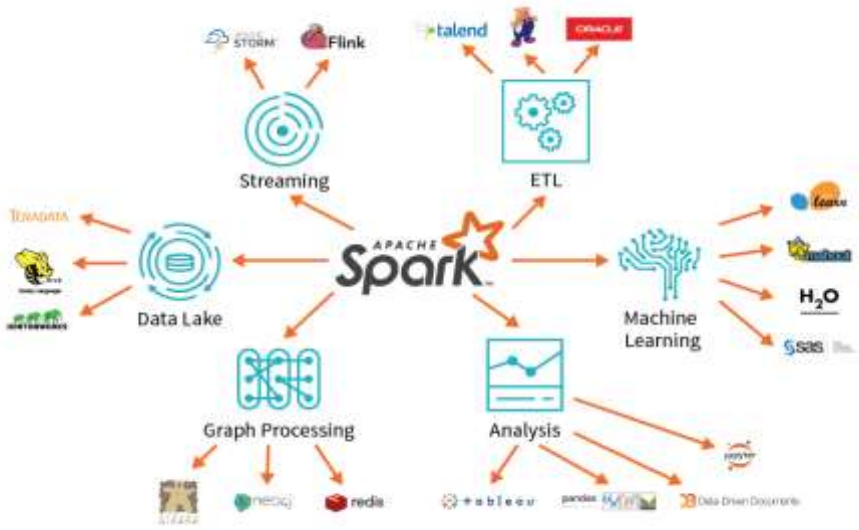


SPARK OVERVIEW:

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala and Python, and an optimized engine that supports general execution graphs.

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop Map Reduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.



EVOLUTION OF APACHE SPARK

Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMP Lab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

Spark was designed for fast, interactive computation that runs in memory, enabling machine learning to run quickly. The algorithms include the ability to do classification, regression, clustering, collaborative filtering, and pattern mining.

Apache Spark consists of Spark Core Engine, Spark SQL, Spark Streaming, MLlib, GraphX, and Spark R. You can use Spark Core Engine along with any of the other five components mentioned above.

Spark introduces the concept of an RDD (Resilient Distributed Dataset), an immutable fault-tolerant, distributed collection of objects that can be operated on in parallel. An RDD can contain any type of object and is created by loading an external dataset or distributing a collection from the driver program.

SPARK OPERATIONS:

Two types of Apache Spark RDD operations are- Transformations and Actions. A Transformation is a function that produces new RDD from the existing RDDs but when we want to work with the actual dataset, at that point Action is performed.

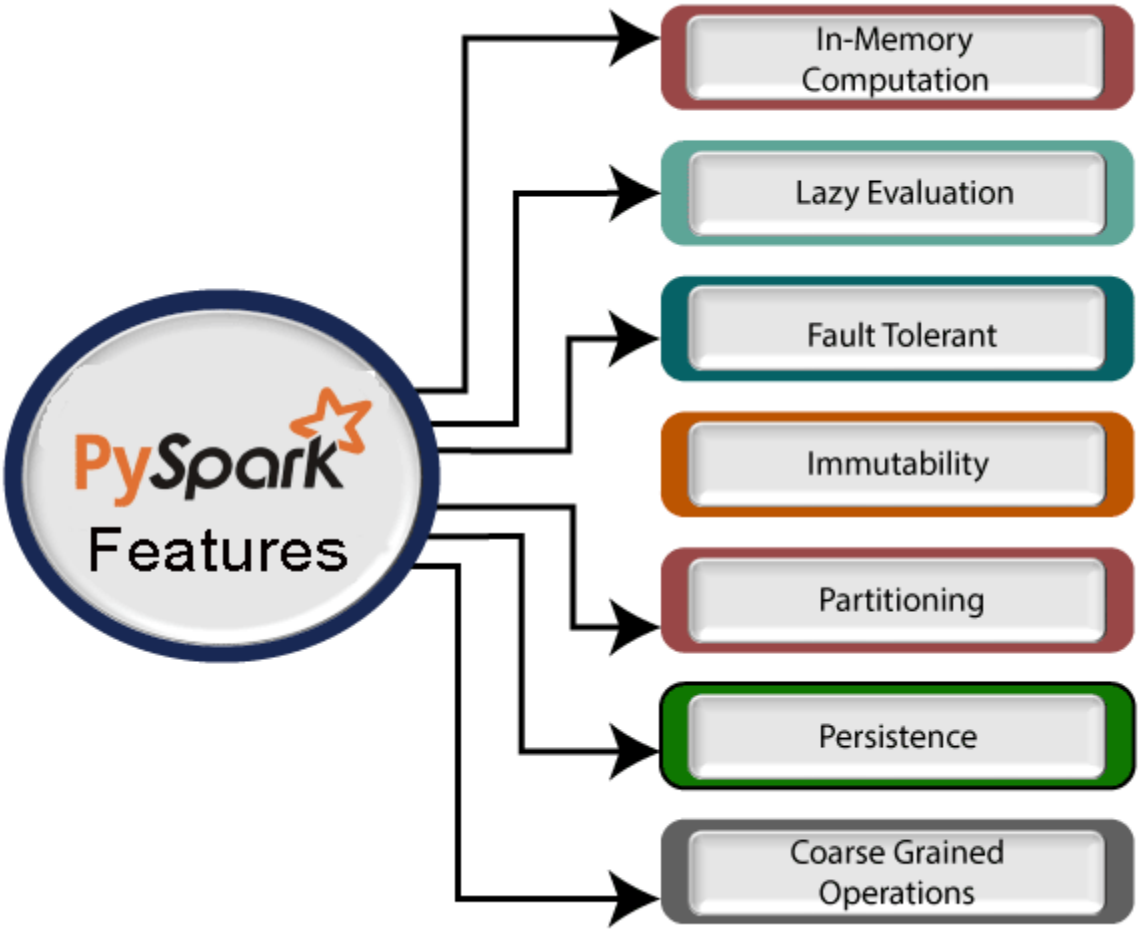
There are Three types of operations on RDDs: Transformations, Actions and Shuffles. The most expensive operations are those the require communication between nodes. Transformations: RDD RDD. Examples map, filter, sample, No communication needed.

PYSPARK:

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analysing your data in a distributed environment.

PySpark is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing. If you're already familiar with Python and libraries such as Pandas, then PySpark is a good language to learn to create more scalable analyses and pipelines.

PySpark provides the already implemented algorithm so that we can easily integrate it. Python is flexible, and we can easily do the data analysis because it is easy to learn and implement. It provides R-related and data science-related libraries. It also supports R programming and data science machine learning etc.



MAPREDUCE:

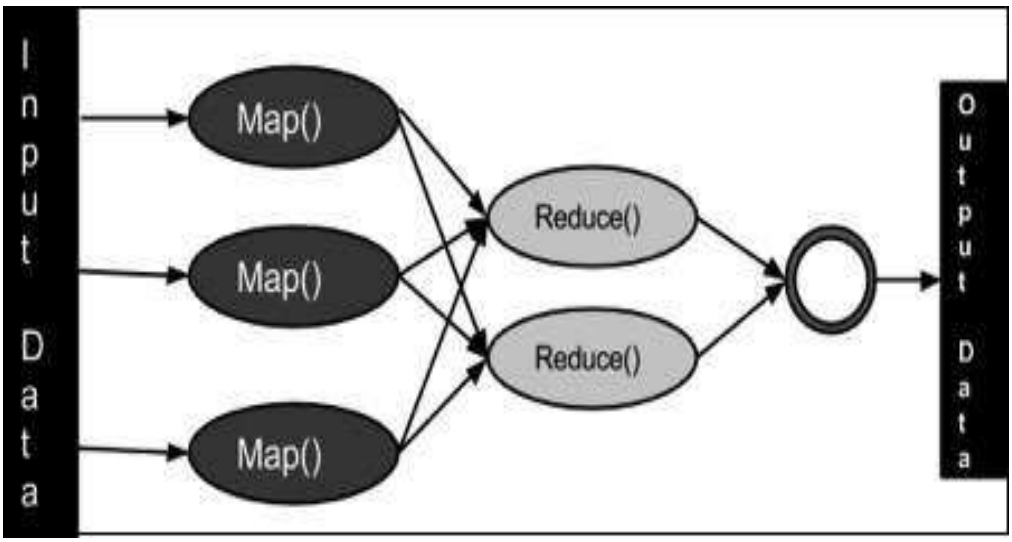
MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. As the processing component, MapReduce is the heart of Apache Hadoop. The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform.

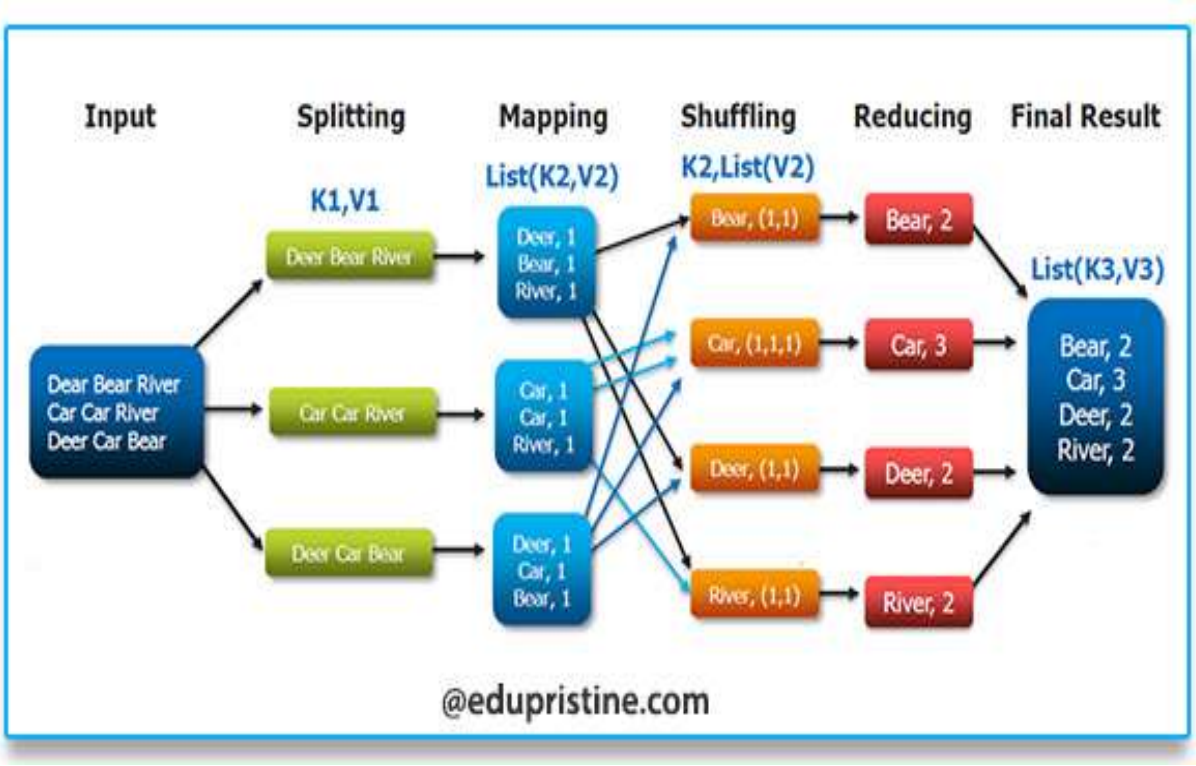
MapReduce is a programming model or pattern within the Hadoop framework that is used to access big data stored in the Hadoop File System (HDFS). It is a core component, integral to the functioning of the Hadoop framework.

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.





DATABASE:

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

TYPES OF DATABASES:

Centralised database.

Distributed database.

Personal database.

End-user database.

Commercial database.

NoSQL database.

Operational database.

Relational database.

RELATIONAL DATABASE:

RDBMS stands for Relational Database Management System.

RDBMS is a program used to maintain a relational database.

RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.

RDBMS uses SQL queries to access the data in the database.

The world's most ubiquitous and flexible open source relational database. MySQL is the most widely adopted open source relational database and serves as the primary relational data store for many popular websites, applications, and commercial products.

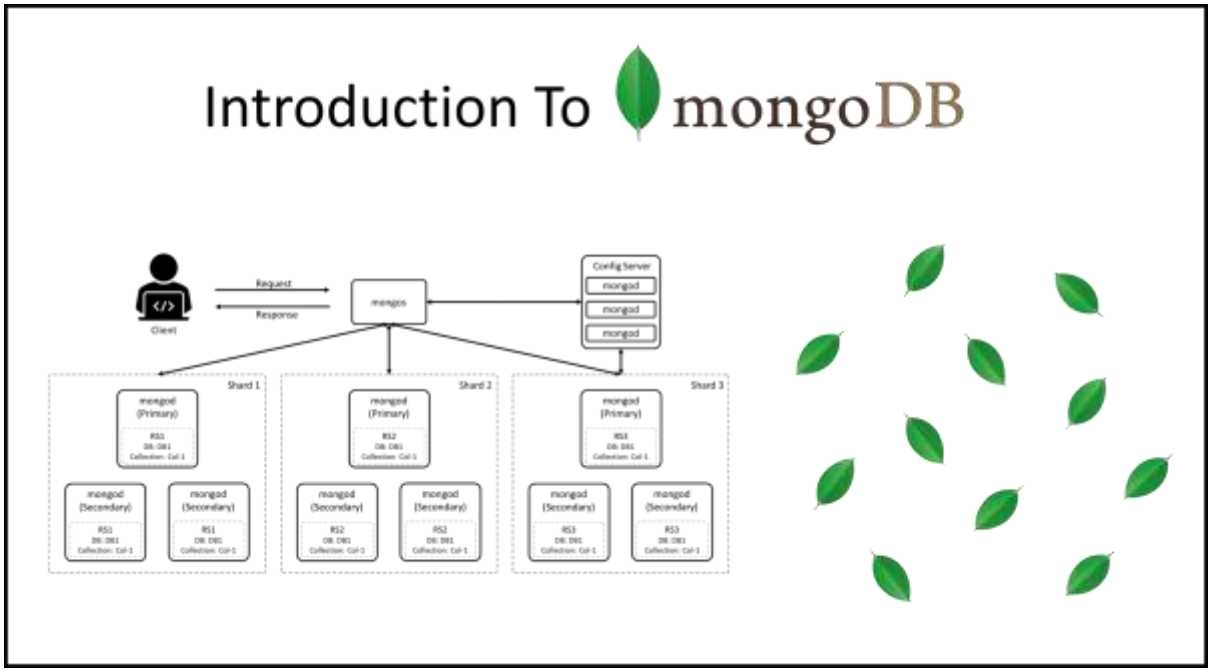
A relational database is a collection of information that organizes data in predefined relationships where data is stored in one or more tables (or "relations") of columns and rows, making it easy to see and understand how different data structures relate to each other. Relationships are a logical connection between different tables, established on the basis of interaction among these tables.

MONGODB:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License which is deemed non-free by several distributions**.**

MongoDB is a document database used to build highly available and scalable internet applications. With its flexible schema approach, it's popular with development teams using agile methodologies.

SQL databases are used to store structured data while NoSQL databases like MongoDB are used to save unstructured data. MongoDB is used to save unstructured data in JSON format. MongoDB does not support advanced analytics and joins like SQL databases support.

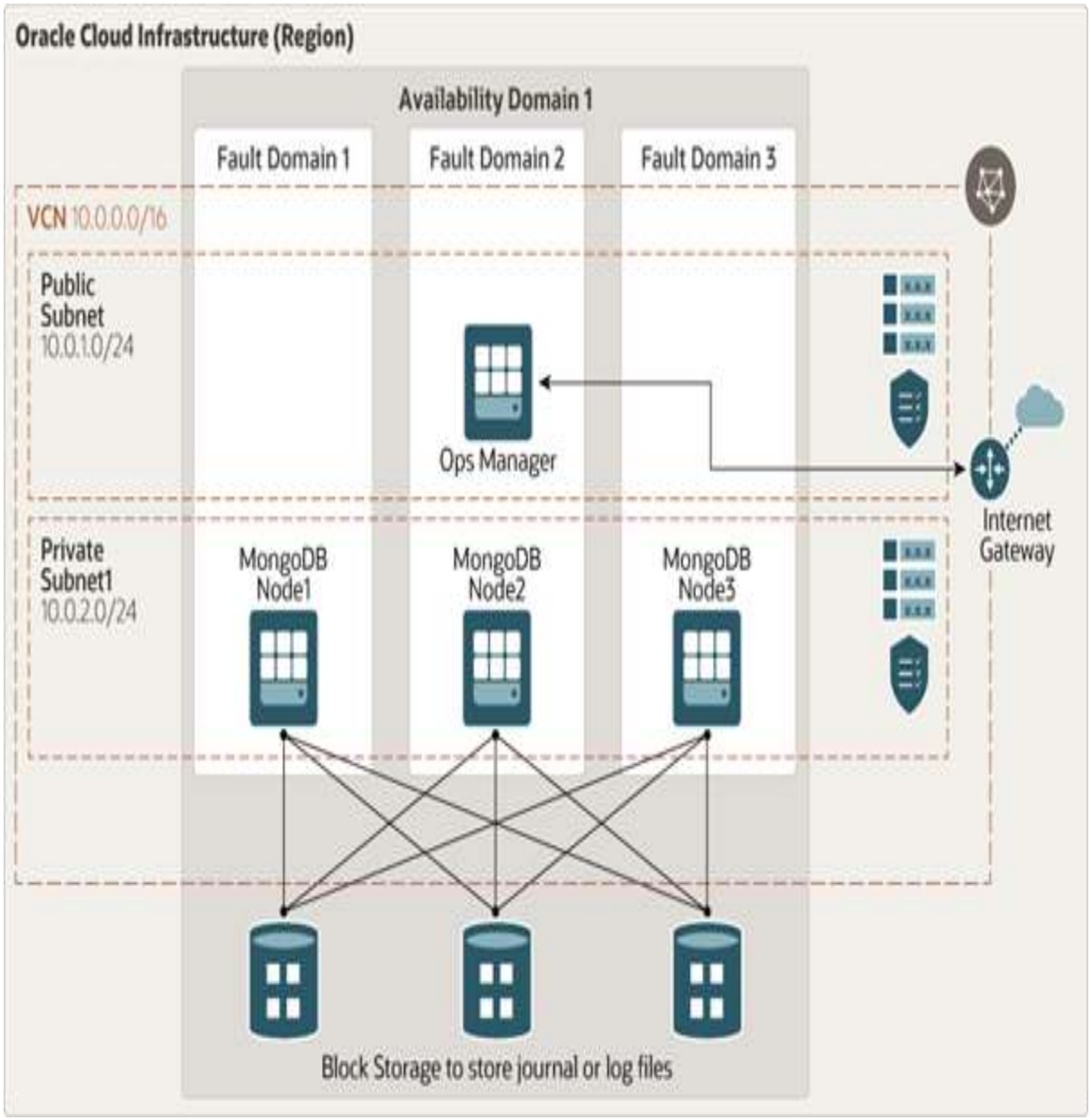


MySQL is an excellent choice if you have structured data and need a traditional relational database. MongoDB is well-suited for real-time analytics, content management, the Internet of Things, mobile, and other types of applications.

MongoDB uses the MongoDB Query Language (MQL), designed for easy use by developers. The documentation compares MQL and SQL syntax for common database operations.

Cons: Data size in MongoDB is typically higher due to e.g. each document has field names stored it. Less flexibity with querying (e.g. no JOINs) no support for transactions - certain atomic operations are supported, at a single document level.

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.



SOFTWARE ENGINEERING BEST PRACTICES:

Software Development "Best Practices"

- "Agile" in all of its forms.
- Automated Testing
- Test Driven Development (TDD)
- Continuous Integration
- Design Patterns
- Code Reviews
- Pair Programming
- Automated Builds

CODING STYLE:

Programming style, also known as code style, is a set of rules or guidelines used when writing the source code for a computer program. It is often claimed that following a particular programming style will help programmers read and understand source code conforming to the style, and help to avoid introducing errors.

A classic work on the subject was The Elements of Programming Style, written in the 1970s, and illustrated with examples from the FORTRAN and PL/I languages prevalent at the time.

The programming style used in a particular program may be derived from the coding conventions of a company or other computing organization, as well as the preferences of the author of the code. Programming styles are often designed for a specific programming language (or language family): style considered good in C source code may not be appropriate for BASIC source code, etc. However, some rules are commonly applied to many languages.

VERSION CONTROL:

Tracks and manages changes in a collection of related entities. It records changes and modifications over time, so you can recall, revert, compare, reference, and restore anything you want.

Version control is also known as source control or revision control. Each version is associated with a timestamp, and the ID of the person making the changes in documents, computer programs, files, etc.

Various application types embed the concept of version control, like SharePoint, Word, spreadsheets, Photoshop, and more. Version control prevents conflicts in concurrent work, and enables a platform for better decision-making and fostering compatibility.

Version Control Systems (VCM) run as stand-alone software tools that implement a systematic approach to track, record, and manage changes made to a codebase. It facilitates a smooth and continuous flow of changes.

TYPES OF VERSION CONTROL SYSTEMS:

There are three main types of version control systems:

Local Version Control Systems

Centralized Version Control Systems

Distributed Version Control Systems

LOCAL VERSION CONTROL SYSTEM:

This version control system consists of a local database on your computer that stores every file change as a patch (difference between files in a unique format). To recreate the file, the various patches will be required to add up at any point in time.

One of the popular examples of this system is the Revision Control System (RCS), still distributed with many computers today. One of the downsides of this system is the local storage of changes. If the local disk/database is damaged or corrupted, all patches will be consequently affected. Also, collaboration in this system is complex, nearly impossible.

CENTRALIZED VERSION CONTROL SYSTEMS

This system has a central and single server that stores the different file versions. This system consists of a server and a client(s). Multiple clients can access files on the server simultaneously, pull current versions of files from the server to their local computer, and vice versa.

There’s just one repository that contains all the history. This approach supports collaboration, and it’s easy to track team member roles on a project. Every change goes to the central server and immediately influences the master code, which will be in production as it’s a simple commit and update process.

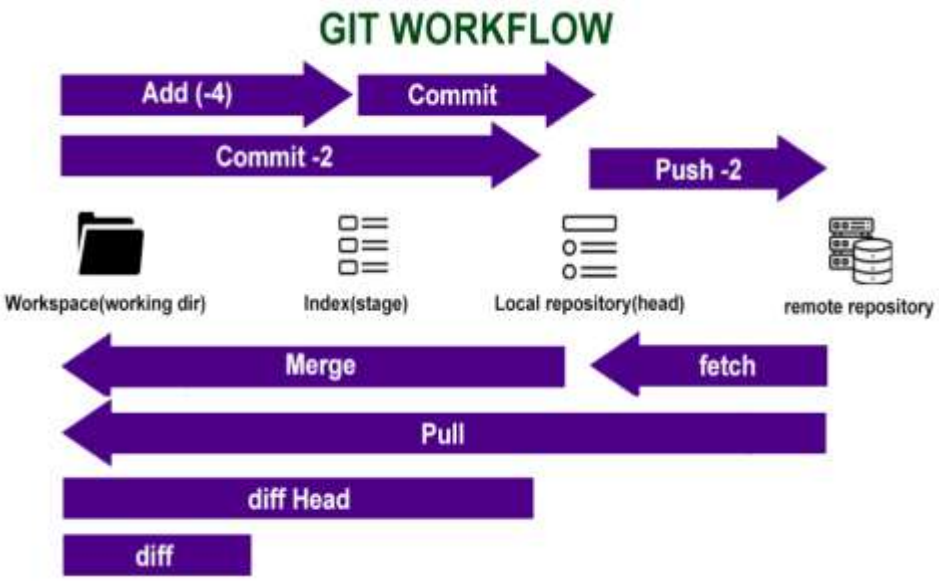
Due to how everything is stored on a centralized server, if the server is damaged or corrupted, there’s a high risk of losing the project’s entire history except whatever single snapshots people happen to have on their local machines. Examples of centralized version control systems are Microsoft Team Foundation Server (TFS), Perforce, and Subversion (SVN).

DISTRIBUTED VERSION CONTROL SYSTEMS

In this system, there's no central repository, but rather multiple repositories. Every client has their server and thoroughly mirrors/clones the repository, including its entire history. This system makes every collaborator a local copy of the whole project, making it easy to work locally and offline.

Committing changes is unique to the client's local repository, and requires it to be pushed to the central repository (which is authoritative). For other team members to get the changes, they have to pull those changes before effectively making updates.

If the server becomes unavailable, damaged, or corrupted, any of the clients' repositories (a correct copy) can distribute or send a copy of the project version back to the server or any other client. Examples of distributed version control systems are Git and Mercurial.



TESTING CODE:

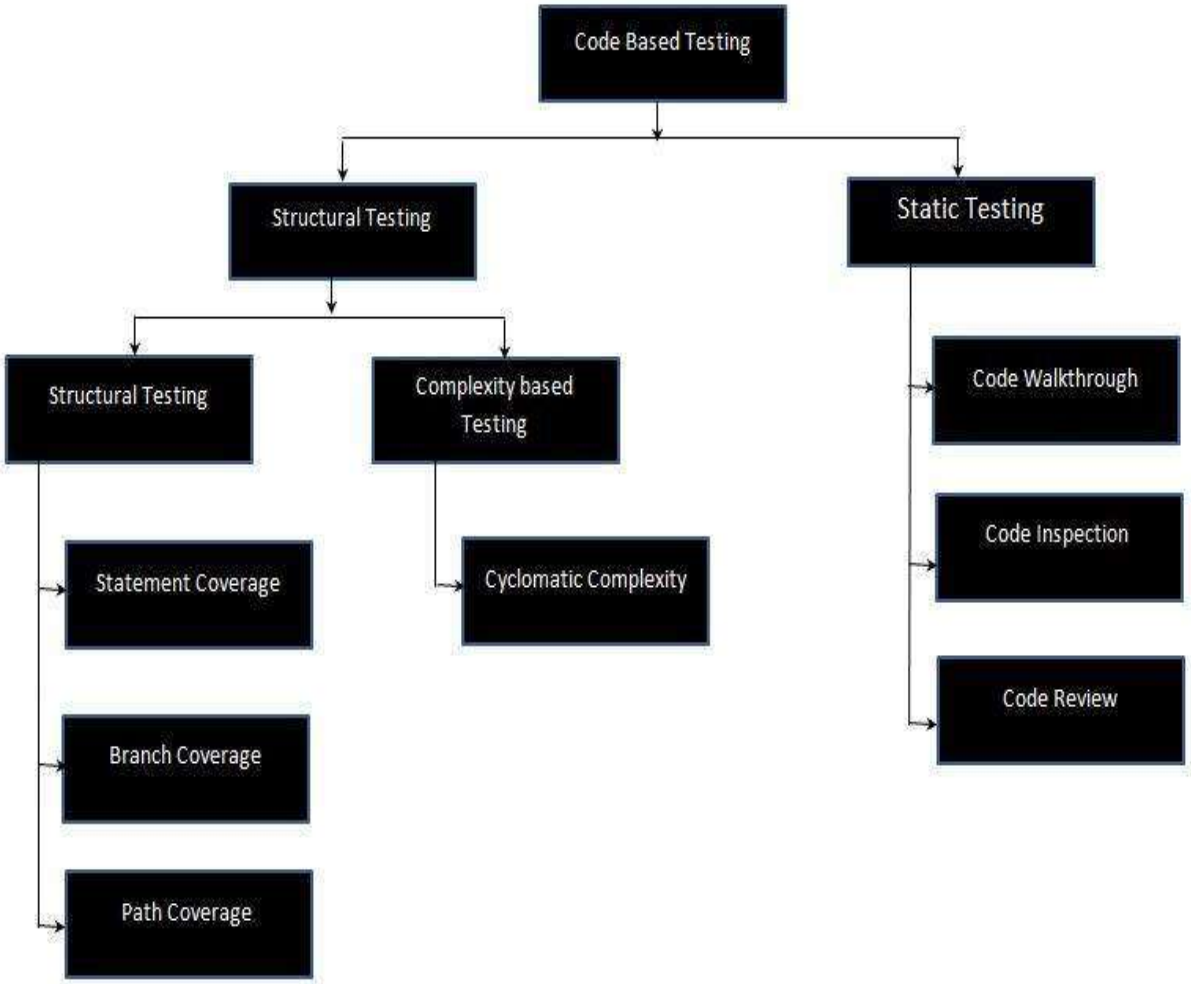
Code-based testing corresponds to the testing that is carried out on code development, code inspection, unit testing in software development process.

The Code-based testing consists of following testing:

Dynamic Testing - Statement coverage, Branch coverage, Path coverage.

Checking for Complexity of Code using techniques like Cyclomatic Complexit
Static Testing - Code Inspection, Code Walkthrough, Code Review, Code Audit.

It simply checks for the number of lines of development code that are being tested. For example: If you are testing a function for finding the maximum in an array, you may be testing the functionality of the function and get 100% code coverage but not test cases such as when the array is out of bounds.



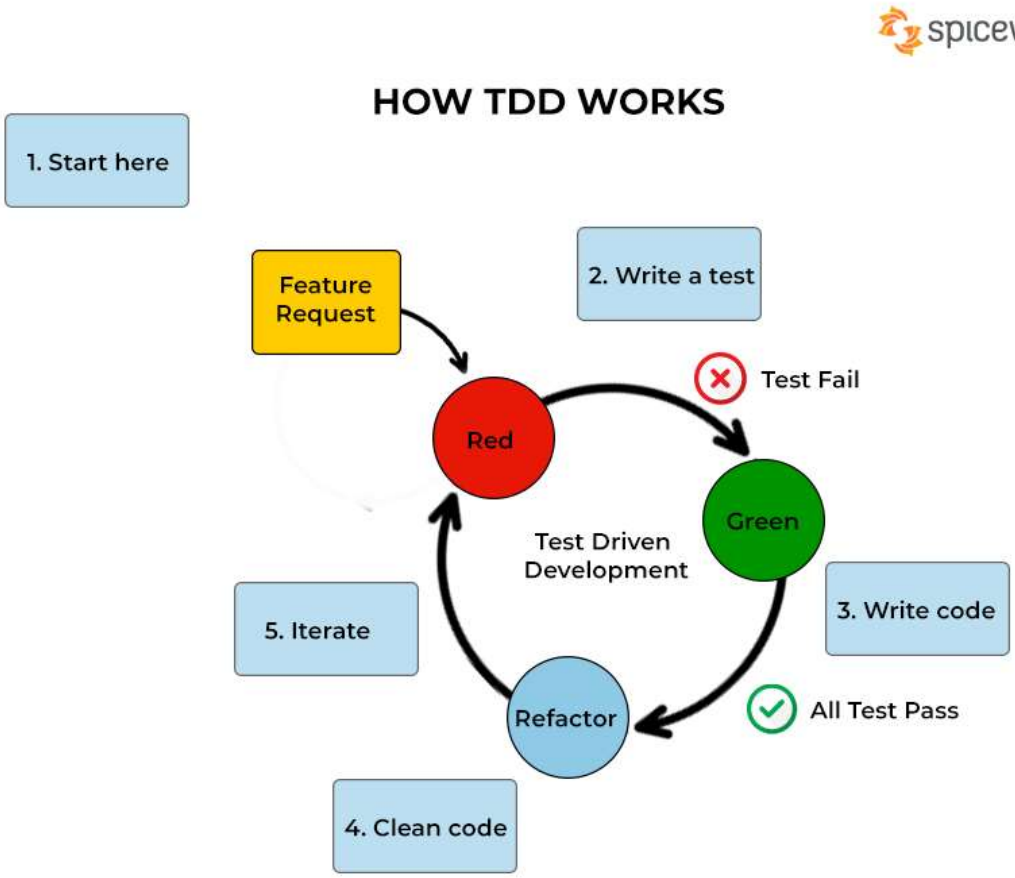
TEST DRIVEN-DEPLOYMENT:

Test-driven development (**TDD**) is a software development process relying on software requirements being converted to test cases before software is fully developed, and tracking all software development by repeatedly testing the software against all test cases. This is as opposed to software being developed first and test cases created later.

Software engineer Kent Beck, who is credited with having developed or "rediscovered"[1] the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.[2]

Test-driven development is related to the test-first programming concepts of extreme programming, begun in 1999,[3] but more recently has created more general interest in its own right.[4]

Programmers also apply the concept to improving and debugging legacy code developed with older techniques.



AGILE METHODOLOGY:

Agile is an approach to software development that seeks the continuous delivery of working software created in rapid iterations. However, the phrase "agile methodology" is misleading because it implies that agile is a singular approach to software development.

THE FOUR VALUES OF THE AGILE MANIFESTO:

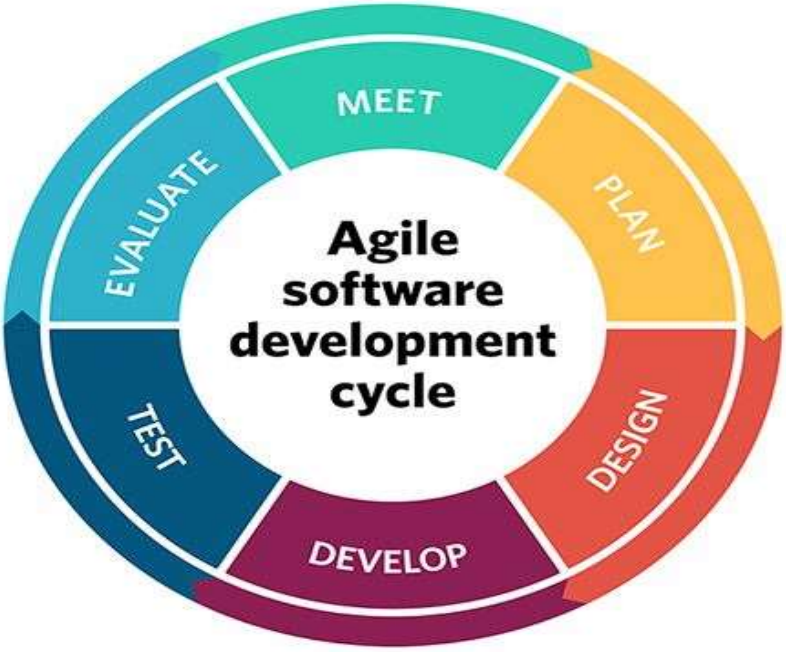
Individuals and interactions over processes and tools.

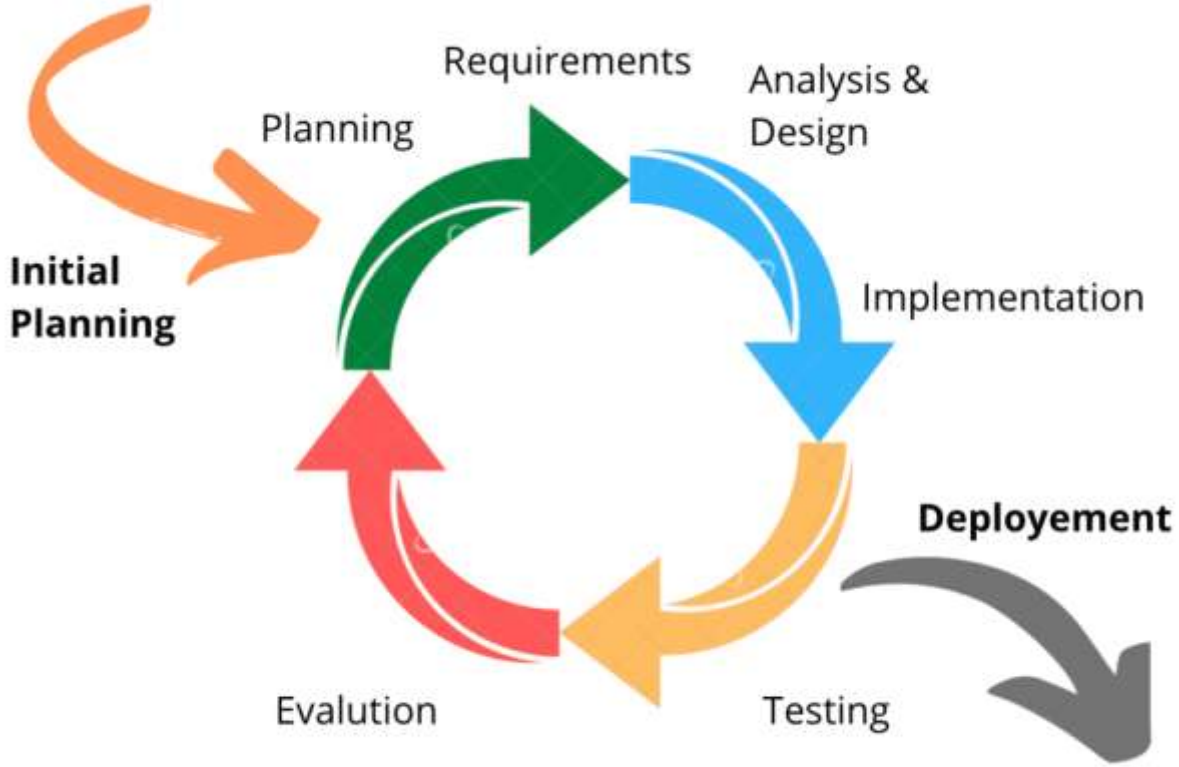
Working software over comprehensive documentation.

Customer collaboration over contract negotiation.

Responding to change over following a plan.

Scrum is the most popular agile development methodology. Teams work in time-boxed sprints of two to four weeks and each person has a clearly delineated role, such as scrum master or product owner.


Agile software development cycle
MEET
PLAN
DESIGN
DEVELOP
TEST
EVALUATE


Initial Planning
Planning
Requirements
Analysis & Design
Implementation
Testing
Deployement
Evalution

NATURAL LANGUAGE PROCESSING:

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI). It helps machines process and understand the human language so that they can automatically perform repetitive tasks. Examples include machine translation, summarization, ticket classification, and spell check.

One of the main reasons natural language processing is so critical to businesses is that it can be used to analyze large volumes of text data, like social media comments, customer support tickets, online reviews, news reports, and more.

All this business data contains a wealth of valuable insights, and NLP can quickly help businesses discover what those insights are.

It does this by helping machines make sense of human language in a faster, more accurate, and more consistent way than human agents.

NLP tools process data in real time, 24/7, and apply the same criteria to all your data, so you can ensure the results you receive are accurate – and not riddled with inconsistencies.

Once NLP tools can understand what a piece of text is about, and even measure things like sentiment, businesses can start to prioritize and organize their data in a way that suits their needs.

CHALLENGES OF NLP

While there are many challenges in natural language processing, the benefits of NLP for businesses are huge making NLP a worthwhile investment.

However, it’s important to know what those challenges are before getting started with NLP.

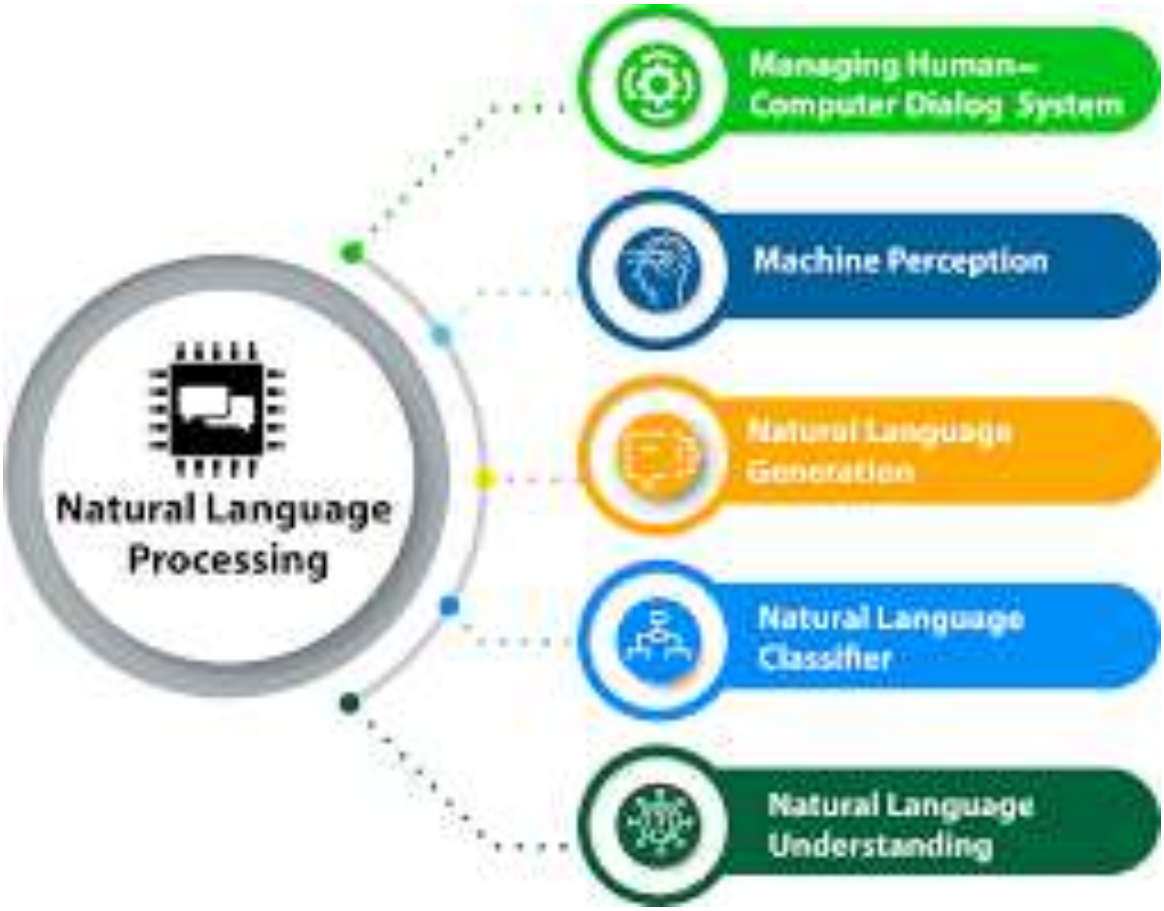
Human language is complex, ambiguous, disorganized, and diverse. There are more than 6,500 languages in the world, all of them with their own syntactic and semantic rules.

Even humans struggle to make sense of language.

So for machines to understand natural language, it first needs to be transformed into something that they can interpret.

In NLP, syntax and semantic analysis are key to understanding the grammatical structure of a text and identifying how words relate to each other in a given context. But, transforming text into something machines can process is complicated.

Data scientists need to teach NLP tools to look beyond definitions and word order, to understand context, word ambiguities, and other complex concepts connected to human language.


Natural Language Processing
Managing Human–Computer Dialog System
Machine Perception
Natural Language Generation
Natural Language Classifier
Natural Language Understanding