**1. Introduction to Apache Spark**

Apache Spark is an **open-source, distributed computing framework** designed for **big data processing and analytics**. It provides fast, scalable, and in-memory computing for large-scale data.

**Why Spark?**

* **Fast Execution** – 100x faster than Hadoop due to in-memory computation.
* **Scalable** – Works on large clusters with thousands of nodes.
* **Multiple Language Support** – Works with Python (PySpark), Scala, Java, and R.
* **Unified Engine** – Supports batch processing, real-time streaming, SQL, ML, and graph processing.

**2. Spark Ecosystem & Components**

Spark has multiple components that enhance its capabilities:

1. **Spark Core** – The foundation for memory management, job scheduling, and fault tolerance.
2. **Spark SQL** – Enables SQL-like querying on large datasets.
3. **Spark Streaming** – Processes real-time data streams.
4. **MLlib** – Machine Learning library for scalable ML algorithms.
5. **GraphX** – Distributed graph-processing engine.

**3. Spark Architecture**

Spark follows a **master-slave architecture** with the following components:

**1. Driver Program**

* The **entry point** for Spark applications.
* It creates a **SparkContext** (for RDDs) or **SparkSession** (for DataFrames).

**2. Cluster Manager**

* Manages **resource allocation** across nodes.
* Examples: **Standalone Spark, Hadoop YARN, Kubernetes, Mesos**.

**3. Executors**

* **Worker nodes** that execute tasks in parallel.

**Spark Execution Workflow**

1. **Driver Program** initializes a **SparkSession**.
2. **Cluster Manager** allocates resources to Executors.
3. **Executors** process the tasks and return results to the Driver.