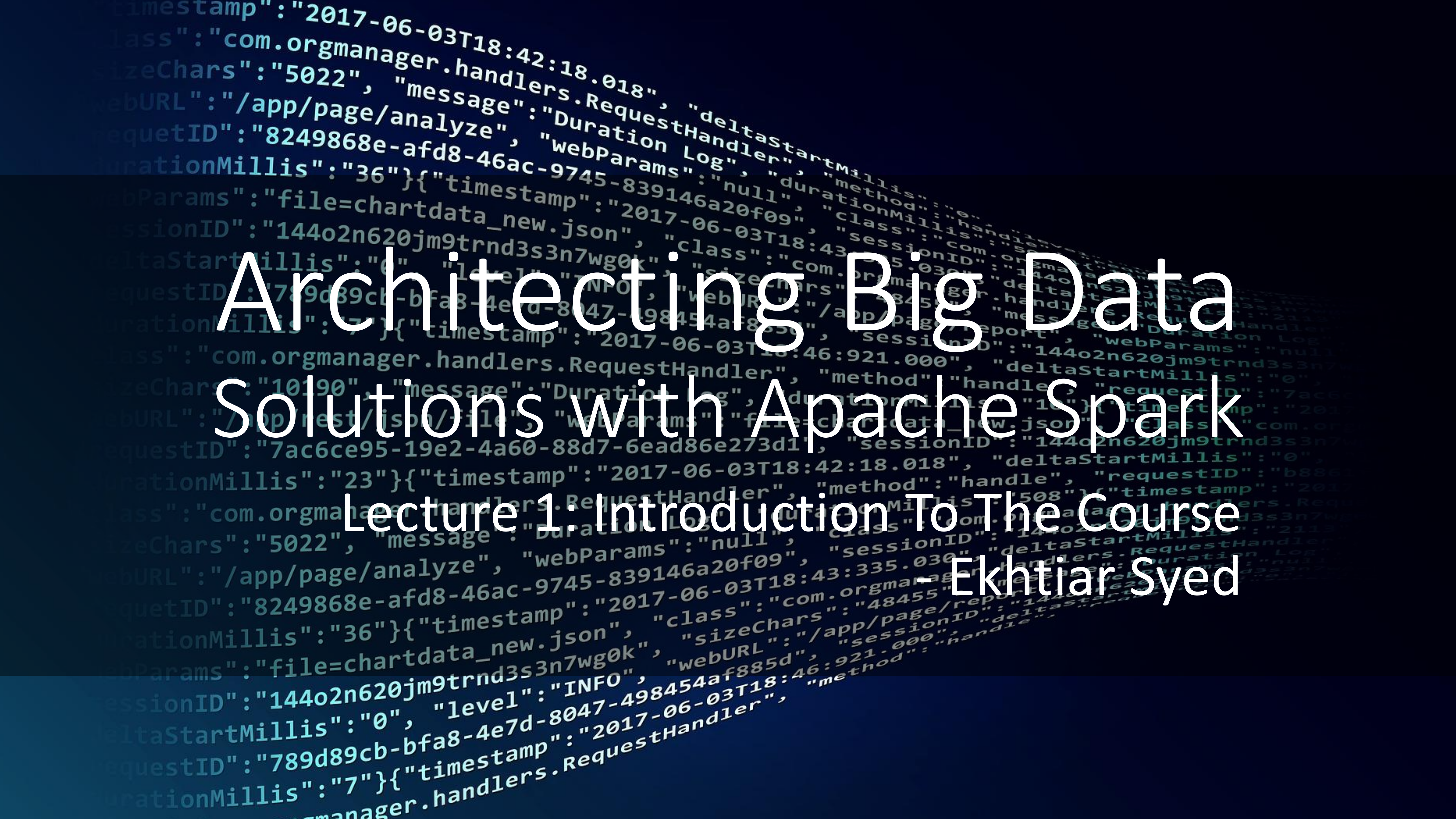


# Architecting Big Data Solutions with Apache Spark

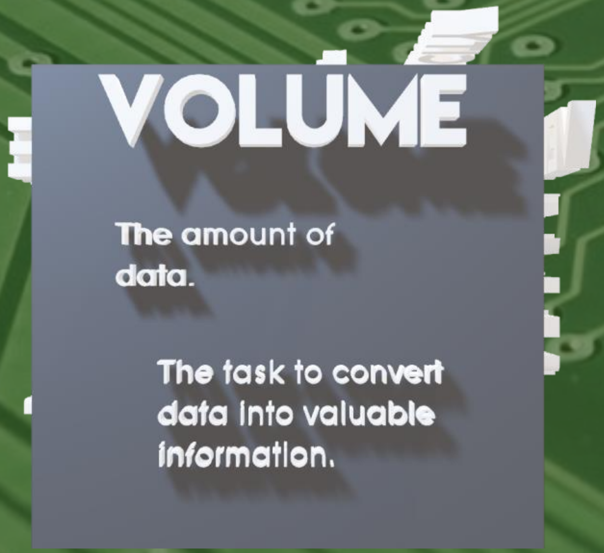
## Lecture 1: Introduction To The Course

- Ekhtiar Syed

# Course Objective

**Architecting** and **Implementing** Data Intensive Applications

# Data Intensive Application

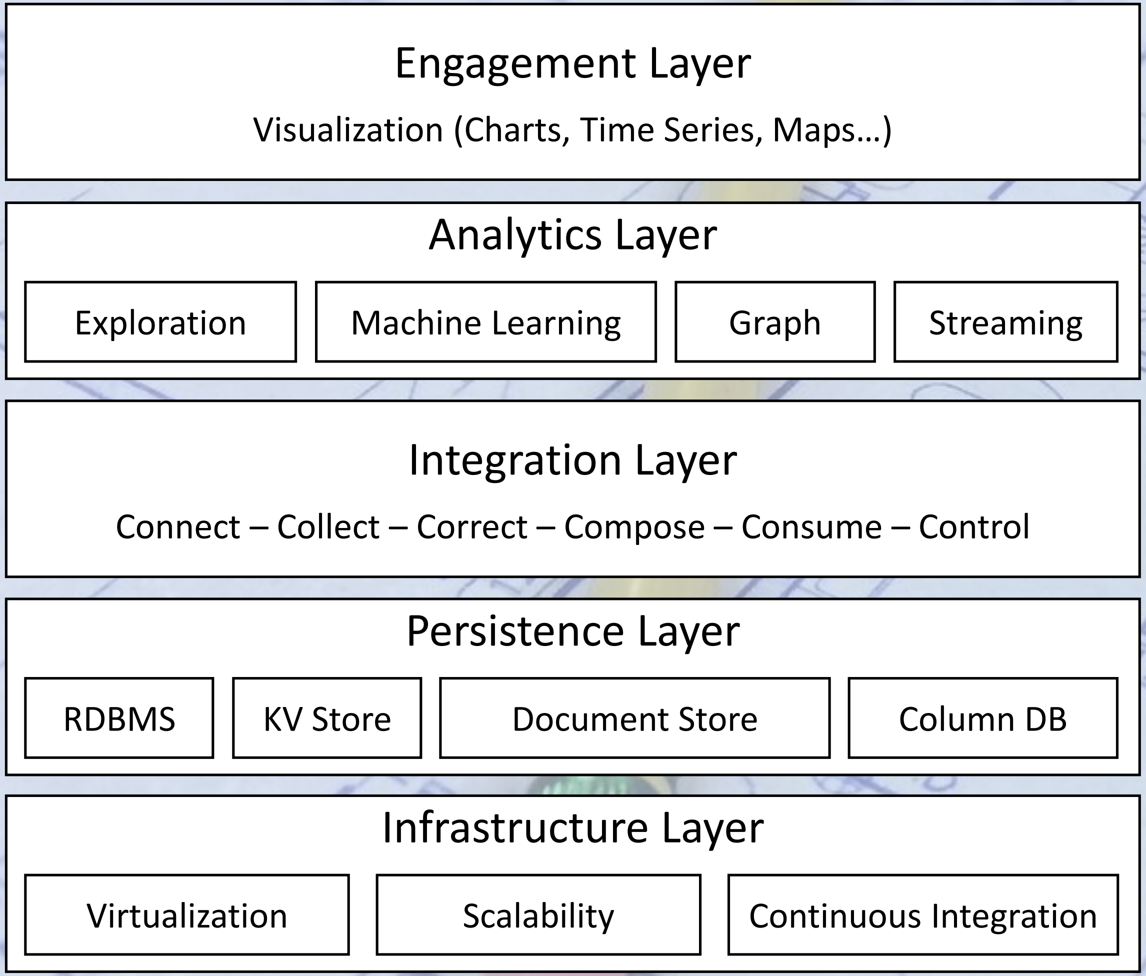


Data-intensive computing is a class of parallel computing applications which use a data parallel approach to process large volumes of data typically terabytes or petabytes in size and typically referred to as big data.

Computing applications which devote most of their execution time to computational requirements are deemed compute-intensive, whereas computing applications which require large volumes of data and devote most of their processing time to I/O and manipulation of data are deemed data-intensive.

- Handbook of Cloud Computing, "Data-Intensive Technologies for Cloud Computing," by A.M. Middleton. Handbook of Cloud Computing. Springer, 2010.

# DIA Architecture



Engagement Layer

Visualization (Charts, Time Series, Maps…)

Analytics Layer

Exploration | Machine Learning | Graph | Streaming

Integration Layer

Connect – Collect – Correct – Compose – Consume – Control

Persistence Layer

RDBMS | KV Store | Document Store | Column DB

Infrastructure Layer

Virtualization | Scalability | Continuous Integration

Source: Spark for Python Developers – Amit Nandi

The engagement layer interacts with the end user and provides dashboards, interactive visualizations, and alerts.

The analytics layer is where Spark processes data with the various models, algorithms, and machine learning pipelines in order to derive insights.

The integration layer focuses on data acquisition, transformation, quality, persistence, consumption, and governance. It is driven by the following five Cs: *connect*, *collect*, *correct*, *compose*, and *consume*.

The persistence layer manages the various repositories in accordance with data needs and shapes.

The infrastructure layer is primarily concerned with virtualization, scalability, and continuous integration.

# Technology Mapping for Our Course

| | |
|---|---|
| Engagement Layer | Superset, matplotlib, Seaborn |
| Analytics Layer | APACHE Spark™ |
| Integration Layer | APACHE Spark™ |
| Persistence Layer | Parquet |
| Infrastructure Layer | databricks, amazon EMR |

We will use visualization tools like Superset to visualize The result of our analytics layer.

We will use spark to solve complex analytics, machine learning and streaming problems. This is another focus area of our course!

We will use Spark to collect and consume data from desperate sources. This is one of the focus areas!

Persistence layer, and Relational and Non-relational databases are a topic of it’s own. I will touch upon only the Parquet file format as our persistence layer.

We will use Databrick’s community edition as our learning platform. If there is time, I would like to touch upon Elastic MapReduce on Amazon Web Services.

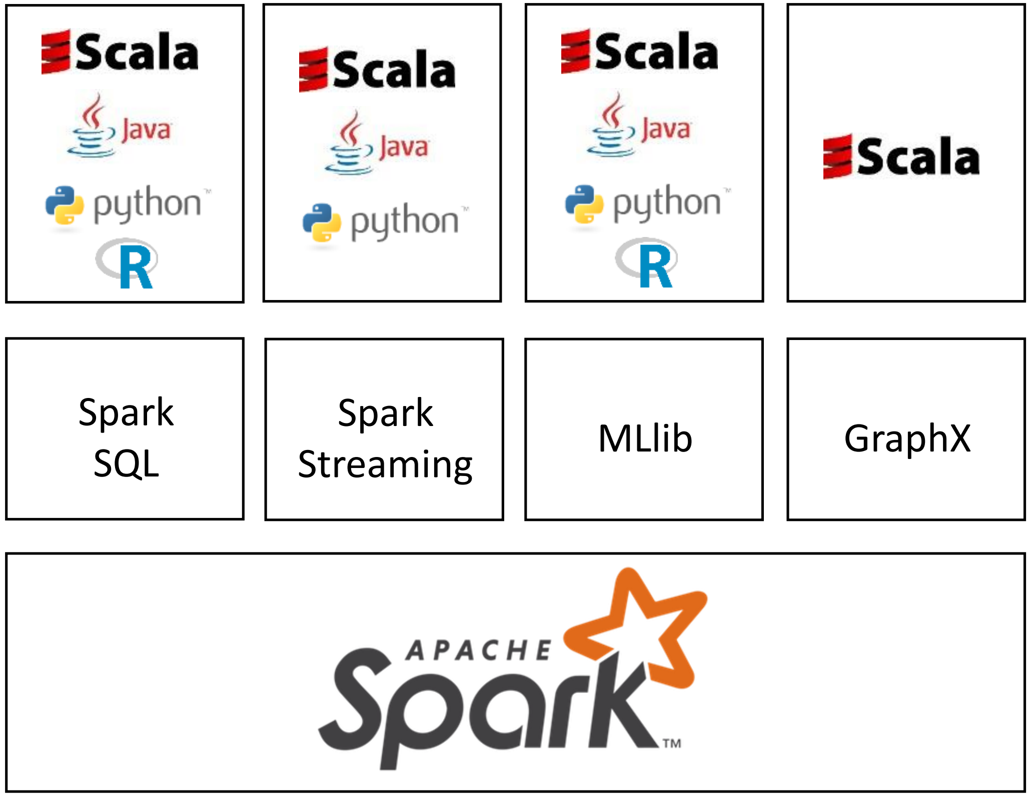
Source: Spark for Python Developers – Amit Nandi

# About Apache Spark

spark.apache.org

- Spark is a fast and general engine for large-scale data processing.
- **Easy to Use:** Spark offers a rich application programming interface (API) for developing big data applications.
- **Fast:** Spark takes advantage of in-memory compute to provide fast data processing capabilities in a distributed environment.
- **General Purpose:** Spark provides a unified integrated platform for different types of data processing jobs.
- **Scalable:** The data processing capacity of a Spark cluster can be increased by just adding more nodes to a cluster.
- **Fault Tolerant:** Spark automatically handles the failure of a node in a cluster. Failure of a node may degrade performance, but will not crash an application.

# About Apache Spark



Spark SQL

Spark Streaming

MLlib

GraphX

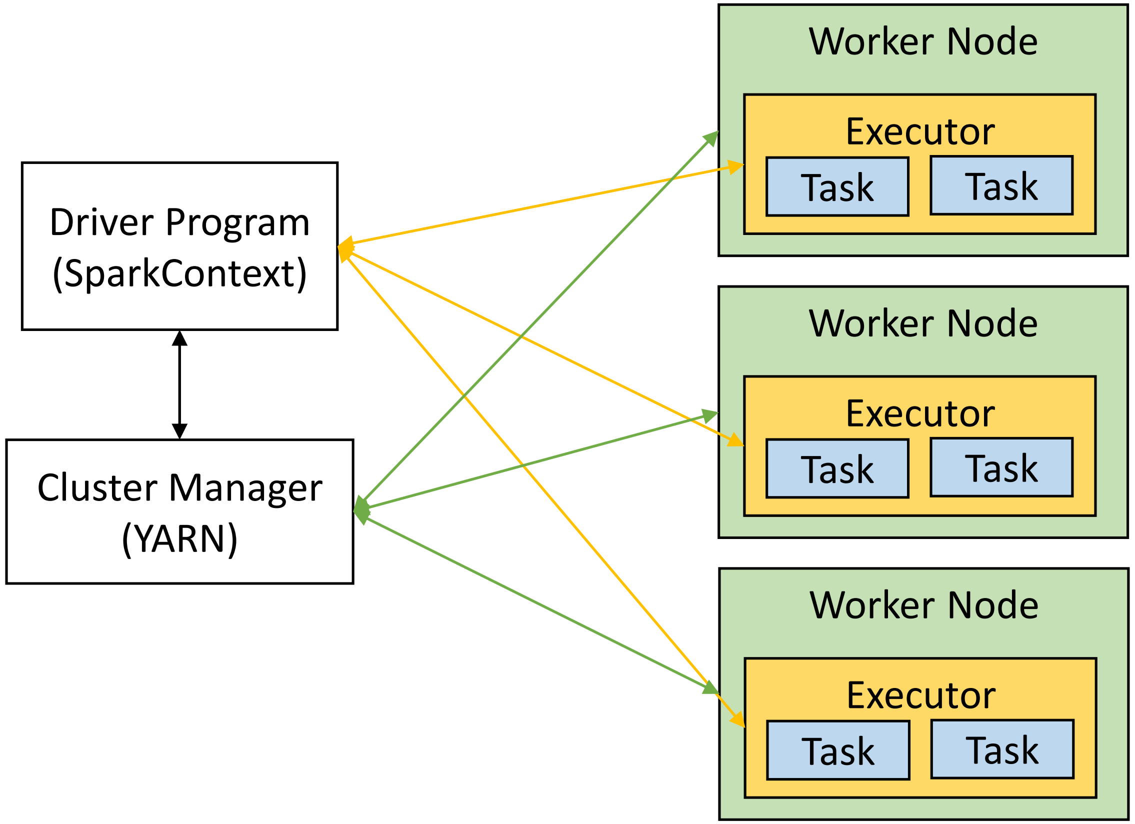
**Spark SQL:** Enables the use of SQL statements or DataFrame API inside Spark applications.

**Spark Streaming:** Enables processing of live data streams

**Mlib:** Enables development of machine learning applications

**GraphX:** Enables graph processing and supports a growing library of graph algorithms

# About Apache Spark



- Involves five key entities: driver program, cluster manager, workers, executors, and tasks.
- Worker provides compute resources to a Spark application and runs as distributed process.
- Spark uses a cluster manager to acquire cluster resources for executing a job. Spark currently supports standalone, Mesos, and YARN.
- Driver program is an application that uses Spark as a library. A driver program can launch one or more jobs on a Spark cluster.
- An executor is a Java virtual machine process that Spark creates on each worker for an application.
- Task is the smallest unit of work that Spark sends to an executor.
- Driver node orchestrates worker nodes to execute the “graph of operations” in a lazy way.

# Let’s Start The Practical Part!