Databricks PySpark

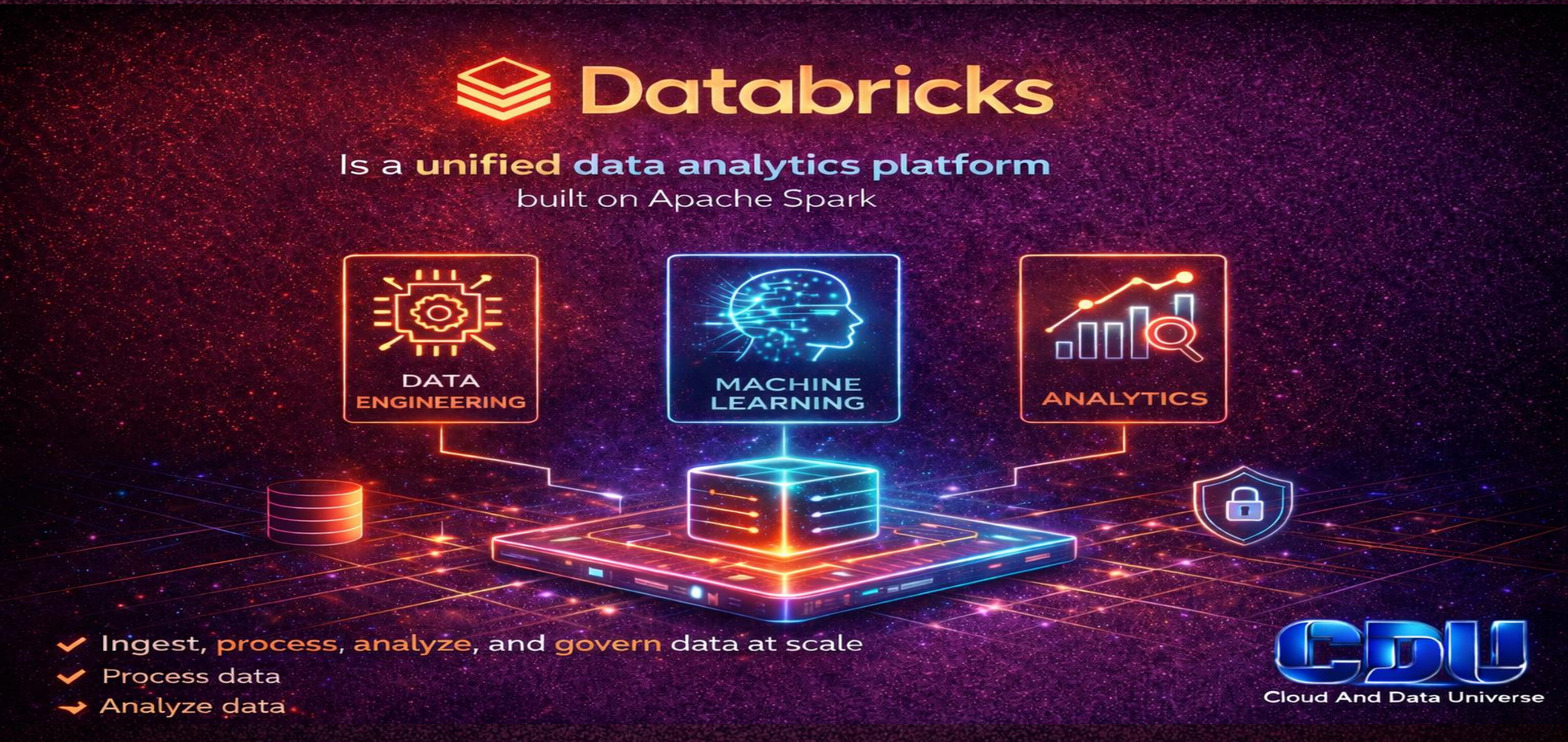
Introduction to Databricks & PySpark

CDU
Cloud And Data Universe

databricks

PySpark

Introduction to Databricks & PySpark



databricks

PySpark

Databricks Features

Databricks

Databricks – Core Platform Features

Unified Analytics Platform

- Single platform for DE, DS, ML & Analytics
- Eliminates silos

Apache Spark Engine

- Optimized runtime
- Batch & Streaming

Lakehouse Architecture

- Data Lake + – Data Warehouse
- Structured & unstructured data
- Built on Delta Lake

Delta Lake

DELTA LAKE

- ACID transactions
- Schema enforcement
- Time travel

CDU
Cloud And Data Universe

databricks

PySpark

Databricks Features

Databricks

Databricks – Enterprise & AI Capabilities

Data Engineering

- ETL / ELT pipelines
- Job scheduling
- Cloud storage integration

Machine Learning & AI

- MLflow
- Model lifecycle
- Python, Scala, R

Databricks SQL & BI

- Serverless SQL
- Dashboards
- Power BI & Tableau

Governance & Security

- Unity Catalog
- Fine-grained access
- Lineage & auditing

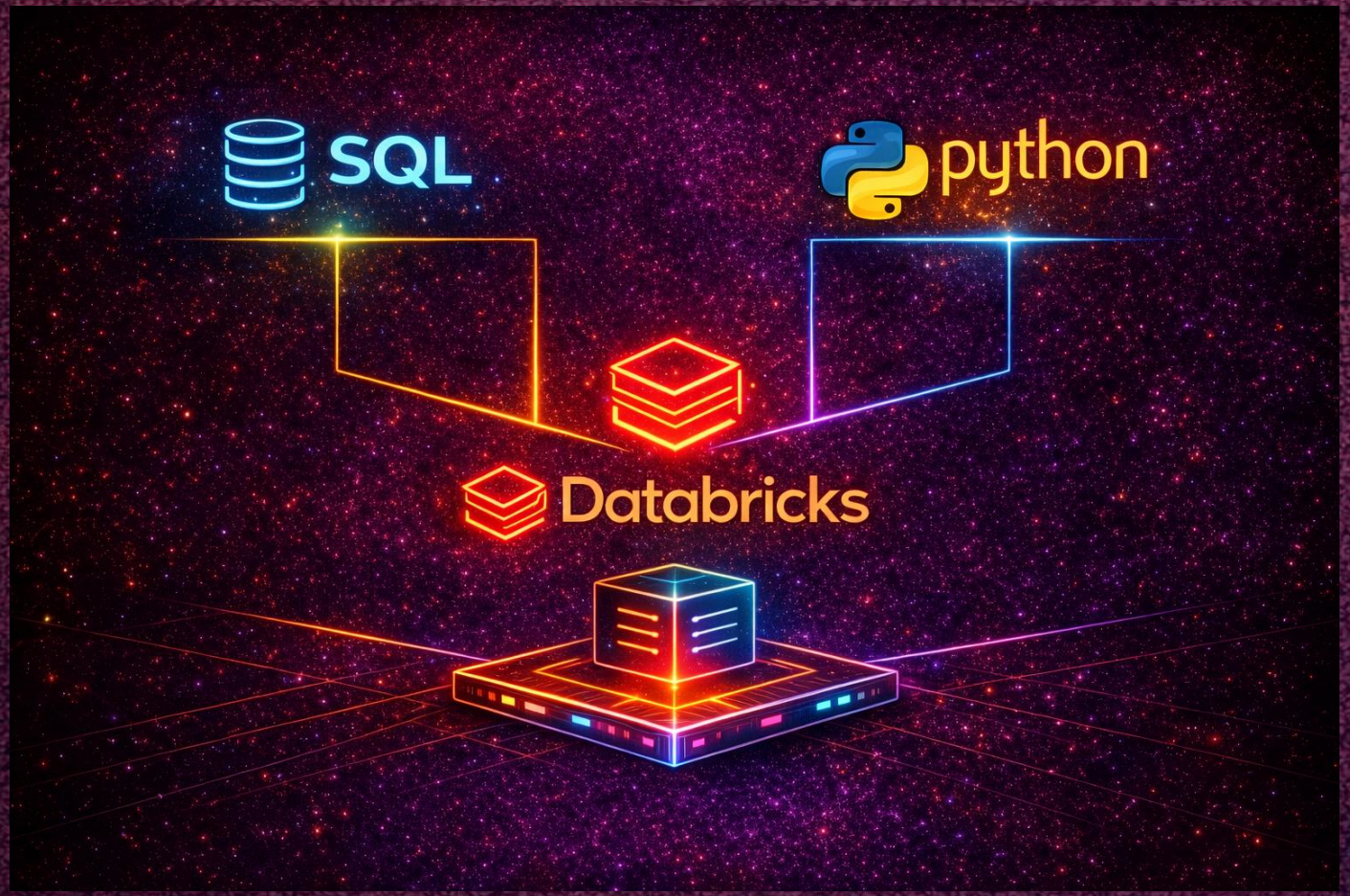
CDU

Cloud And Data Universe

databricks

PySpark

Pre-Requisites

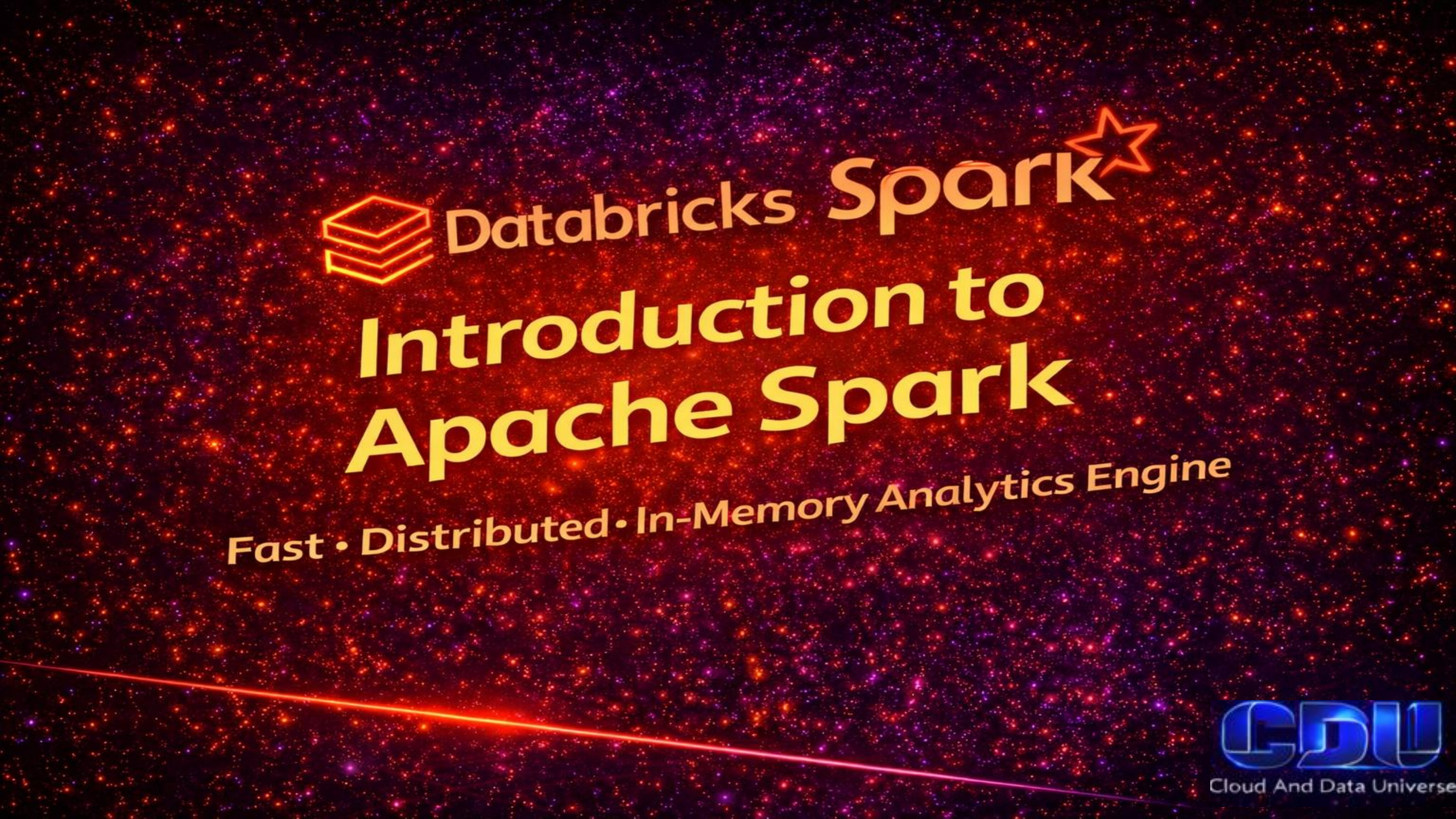


CDU

Cloud And Data Universe

Let's begin a great journey!

Cloud And Data Universe


Databricks Spark
Introduction to Apache Spark
Fast • Distributed • In-Memory Analytics Engine
CDU
Cloud And Data Universe

Why Apache Spark?

- In-memory processing
- Faster than MapReduce
- Scales to big data
- Batch & Streaming support

⚠ MapReduce Challenges

Slow processing

Disk I/O bottleneck

Complex programming

Difficult to scale

CDU
Cloud And Data Universe

The Evolution of APACHE SPARK

2009 — BIRTH OF SPARK

UC Berkeley | AMPLab

Faster than Hadoop MapReduce

Written in Scala

2010 — OPEN SOURCE RELEASE

Spark released as open source

Early academic & Startup adoption

- RDD (Resilient Distributed Dataset)

2013 — APACHE INCUBATION

Donated to Apache Software Foundation

- Community-driven development begins

2014 — APACHE SPARK 1.0

Top-level Apache project

- Spark Core
- Streaming, MLLib.
- GrapX
- Early Spark SQL

2015 — DATAFRAMES & SPARK SQL

DataFrame API introduced

- Catalyst Optimizer added
- SQL becomes first-class API

2016 — SPARK 2.0 (MAJOR MILESTONE)

Unified APIs: RDD, DataFrame-Dataset

- Dataset API introduced
- Structured Streaming introduced

RDDs → DataFrames → Datasets

STRUCTURED STREAMING

CDU

Cloud And Data Universe

The Evolution of APACHE SPARK

2017–2018 — STREAMING & PERFORMANCE

- Structured Streaming becomes stable
- Event-time processing
- Windowing & watermarking
- Cloud adoption increases

2020 — SPARK 3.0

- Adaptive Query Execution (AQE)
- ANSI SQL complance
- Dynamic partition pruning

2020 — SPARK 3.0

- ANSI SQL compliance
- Dynamic partition pruning
- Faster joins & queries

2021–2022 — LAKEHOUSE ERA

- Deep cloud storage integration
- ADLS • S3 • GCS
- Delta Lake, Iceberg, Hudi
- Foundation of Lakehouse architecture

2023 — SPARK 3.4+

- Faster Python execution
- Pandas API on Spark
- Better observability

2024–2025 — MODERN SPARK

- SQL, Streaming & ML engine
- Cloud-native analytics
- AI feature pipelines
- Industry standard for big data

2017-2018 STREAMING & PERFORMANCE 2020 SPARK 3.4+ MODERN SPARK

CDU
Cloud And Data Universe

Apache Spark

Core Components

CDU

Cloud And Data Universe

Apache Spark

Core Components

SPARK CORE

SPARK SQL

STRUCTURED STREAMING

MLLIB

GRAPHX

CDU

Cloud And Data Universe

Apache Spark
Core Components

SPARK CORE

- The foundation of Spark
- Distributed task scheduling
- Memory management
- Fault tolerance
- RDD abstraction
- Interaction with storage systems (HDFS, S3, ADLS, etc.)

SPARK SQL

- For structured & semi-structured data
- SQL queries
- DataFrame & Dataset APIs
- Catalyst Optimizer
- ANSI SQL support
- Integration with Hive Metastore
- Analytics, BI workloads, ETL transformations

CDU
Cloud And Data Universe

Apache Spark
Core Components

STRUCTURED STREAMING

- Modern streaming engine
- Event-time processing
- Windowing & watermarking
- Exactly-once guarantees
- Same API as batch (DataFrames)

GRAPHX

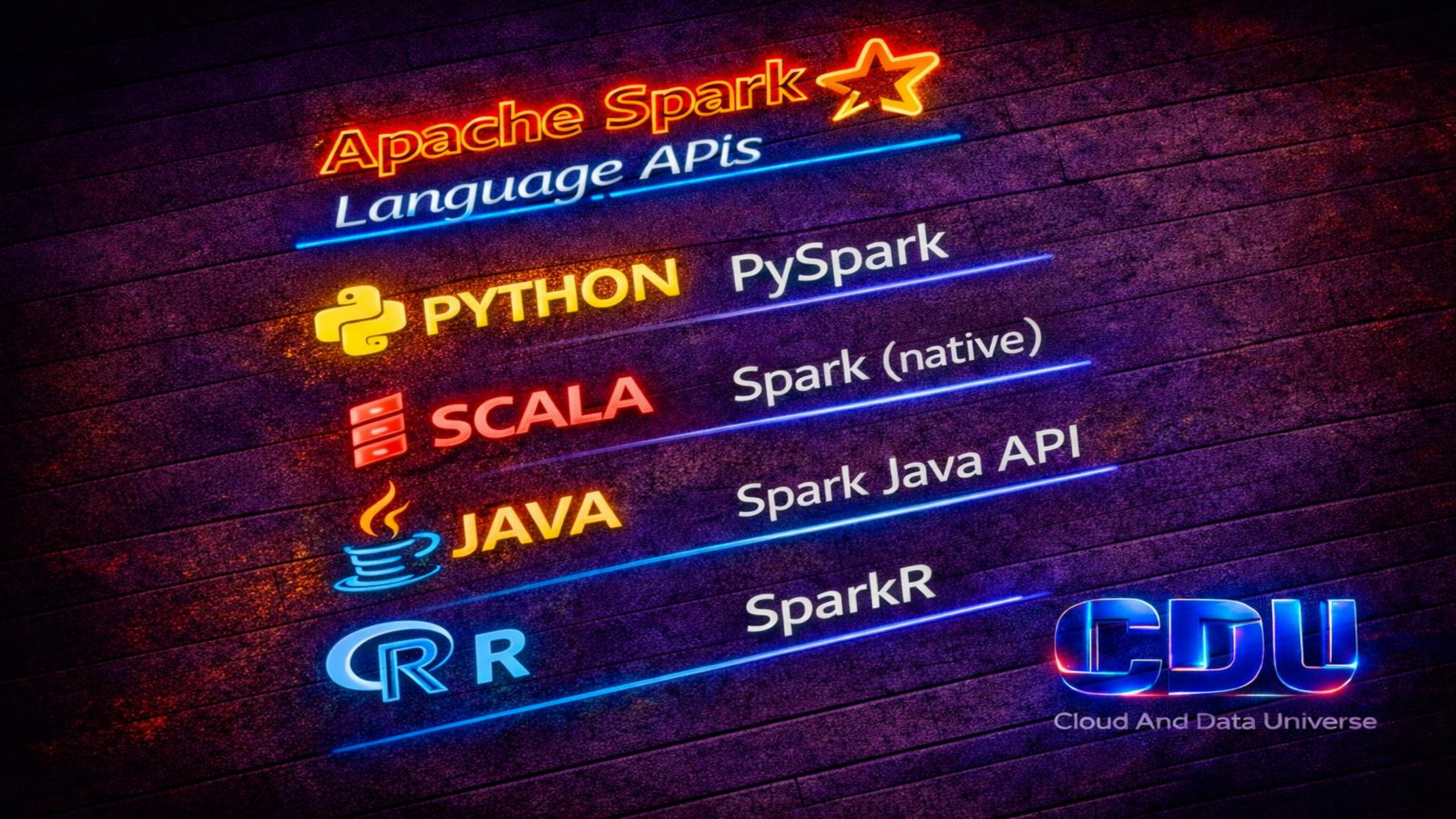
- Graph processing framework
- Graph algorithims
- PageRank
- Connected components
- Social network analysis

MLLIB

- Classification
- Regression
- Clustering
- Recommendation systems
- Feature engineering

CDU

Cloud And Data Universe

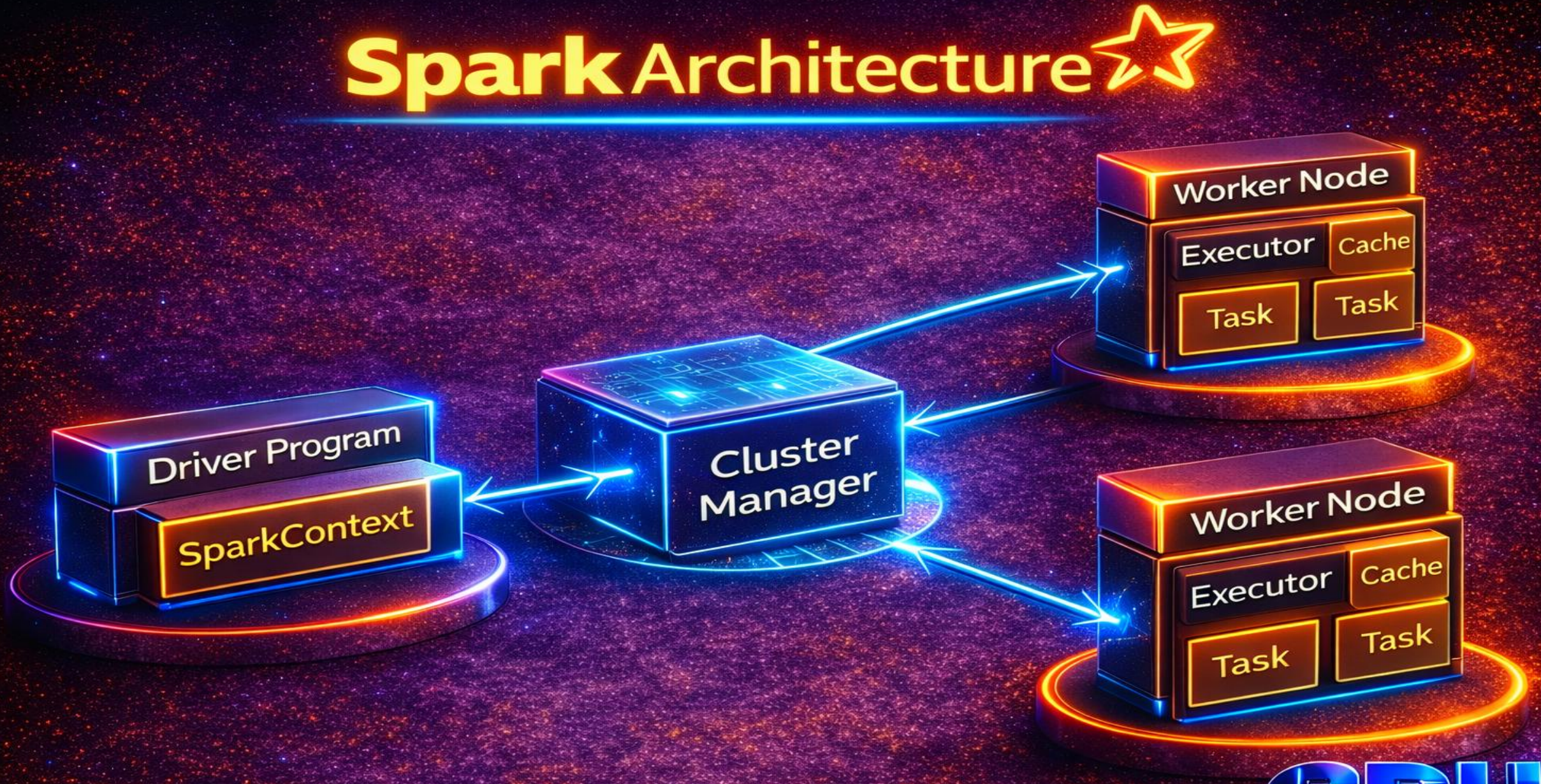

Apache Spark
Language APIs
PYTHON
PySpark
SCALA
Spark (native)
JAVA
Spark Java API
R
SparkR
CDU
Cloud And Data Universe

Spark Architecture

CDU

Cloud And Data Universe

Spark Architecture



CDU
Cloud And Data Universe

What is a Spark Application?

A **Spark Application** is a single, complete program that runs on on Apache Spark to process data.

In simple words: = your code + **Spark engine** + **cluster resources** working to complete a task.

Formal Definition (Interview-Ready)

- A **Spark application** is a **user program** that runs on Apache Spark,
- Consisting of **one driver program** and **one or more executor processes**,
- Used to perform **distributed data processing.**

CDU
Cloud And Data Universe

Driver Program

(Brain of Spark)

The **Driver** is where your Spark application starts.

Responsibilities:

- ✔ Creates **SparkSession / SparkContext**
- ✔ Converts your code into a logical plan
- ✔ Splits jobs into **stages** & tasks
- ✔ Sends **tasks** to **executors**
- ✔ Collects **results**

📌 **Only ONE driver** per Spark application

CDU

Cloud And Data Universe

SparkContext / SparkSession

Entry point to Spark

Talks to:

- ✔ Cluster Manager
- ✔ Executors

Tracks:

- ✔ RDDs
- ✔ DataFrames
- ✔ Jobs

In modern Spark, you mainly use **SparkSession.**

CDU

Cloud And Data Universe

Cluster Manager

The Middle Layer in Spark

Role of Cluster Manager

The Cluster Manager acts as the mediator between the **Driver** and the **Executors**.

- ✗ It does **NOT process data.**
- ✓ It manages **resources.**

What the Cluster Manager Actually Does

1 Driver requests resources

- ✓ CPU cores
- ✓ Memory

2 Cluster Manager allocates executors

- ✓ Decides where executors run
- ✓ Decides how many executors are created

3 Executors are launched on worker nodes

- ✓ Cluster Manager starts them
- ✓ Executors then register back with the Driver

📌 After this, Driver talks directly to Executors
Cluster Manager steps aside.

CDU
Cloud And Data Universe

Executors

Executors

- JVM processes running on worker nodes
- Execute tasks
- Store data in memory or disk
- Send results back to the driver

📌 Executors live only as long as the application runs

CDU
Cloud And Data Universe

Spark Tasks

What is a Task?

- A task is the smallest unit of work in Spark
- Runs on a single data partition
- Executed by an executor

Key Characteristics

- One task = one partition
- Tasks run in parallel across executors
- Tasks are stateless
- Tasks cannot be shared across executors

How Tasks Are Created

- Triggered by a Spark action (count(), collect(), save())
- A job is divided into stages
- Each stage contains multiple tasks

CDU
Cloud And Data Universe

What is a Job in Spark?

A Spark Job is a unit of execution that is triggered when an action is called on a Spark DataFrame, Dataset, or RDD.

In simple words:
A job is the work Spark decides to execute when you ask for a result.

Formal Definition (Interview-Ready)

- A Spark job is a computation triggered by an **action**, which Spark executes by dividing the work into **multiple stages** and tasks.

How a Spark Job is Created

1. You write Spark transformations
2. Spark builds a logical plan (lazy execution)
3. You call an action

   `count()` → `collect()` → `save().`
4. Spark creates a job
5. Job is split into stages
6. Stages are split into tasks
7. Tasks run on executors

CDU
Cloud And Data Universe

Spark Application Execution Flow

1. **You submit a Spark program**
   - Code is sent to the cluster

2. **Driver starts**
   - Creates SparkSession / SparkContext
   - Builds the logical plan

3. **Driver requests resources**
   - CPU cores
   - Memory

4. **Cluster Manager allocates executors**
   - Launches executors on worker nodes
   - Executors register back with the driver
   - This is where data distribution begins

5. **Data is read from source**
   - HDFS / ADLS / S3 / Database
   - Data is split into partitions

6. **Partitions are distributed across executors**
   - Each executor receives multiple partitions
   - One task is created per partition

7. **Driver creates jobs, stages, and tasks**
   - Job → triggered by action
   - Stage → split at shuffle boundaries
   - Task → one per partition

8. **Driver sends tasks to executors**
   - Tasks are scheduled close to data when possible

9. **Executors process data in parallel**
   - Transformations applied
   - Shuffles happen if required
   - Data cached if needed

10. **Results are returned to the driver**
    - Or written to storage

11. **Application finishes**
    - Executors are terminated
    - Resources released back to cluster manager

CDU
Cloud And Data Universe

Time for Practical Execution


Databricks
Assistant
Your AI Copilot
CDU
Cloud And Data Universe

Databricks

What Databricks Assistant does

Understands Your Workspace

It has context of:

Provides Context-Aware Answers...

...Not Generic AI Replies

Databricks

Data & analytics assistance

- Generate Databricks SQL queries
- Explain query execution plans
- Help with window functions, joins, aggregations
- Suggest performance improvements (AQE, caching, partitioning)

Example:

- Optimize this Pyspark join"
- Convert this SQL query to PySpark DataFrame API"

CDU
Cloud And Data Universe

Databricks

Databricks
Assistant
Your AI Copilot

Code Help
(Core Strength)

You can ask it to:

- ✓ Write PySpark / Spark SQL / Python code
- ✓ Convert SQL ⟷ PySpark
- ✓ Optimize slow Spark jobs
- ✓ Explain existing notebook code
- ✓ Fix errors and exceptions

Example:

- “Optimize this PySpark join”
- “Convert this SQL query to PySpark DataFrame API”

CDU
Cloud And Data Universe

Databricks

Databricks
Assistant
Your AI Copilot

Platform & governance help

- Explain Unity Catalog
- Help with permissions & access control
- Answer questions on Delta Lake, time travel, schema enforcement

CDU
Cloud And Data Universe

Databricks

Databricks
Assistant
Your AI Copilot

ML & AI support

- Help with MLflow
- Feature engineering in Spark
- Model training & tracking examples.
- Explain notebooks used.
  in ML pipelines

CDU
Cloud And Data Universe

Creating RDD from list

CDU
Cloud And Data Universe

Understanding RDD Partitions

Parallelism

- Allows many tasks to be executed in parallel

Optimization

- Number of partitions impacts performance

Fault Tolerance

- Individual partitions can be recomputed if lost

CDU
Cloud And Data Universe

Managing PARTITIONS

Parallel Processing

- Optimizes resource usage

Load Balancing

- Handles skewed data distribution

Repartition/Coalesce

- Scale partitions up or down using: *repartition| | coalese*

C.D.U

Cloud And Data Universe

Unity Catalog

We're at a transition point in Databricks.

As DBFS is no longer the recommended approach,
we need to adopt

Unity Catalog

as the industry standard
to manage files and tables.

CDU
Cloud And Data Universe

❌ Why DBFS Is No Longer the Best Option

- DBFS is workspace-scoped, not account-scoped

- Limited governance & access control

- Not designed for enterprise-grade data security

- Difficult to manage permissions across teams

- Not aligned with modern lakehouse governance

⚠ DBFS = legacy convenience, not enterprise standard

CDU

Cloud And Data Universe

Why Unity Catalog Is the Industry Standard

- Centralized data & file governance
- Fine-grained access control (catalog → schema → table)
- Works across all Databricks workspaces
- Integrated with cloud storage (ADLS / S3 / GCS)
- Built for security, compliance, and scalability

Unity Catalog = governance-first architecture

CDU
Cloud And Data Universe

How We Should Manage Files & Tables Now

Storage Layer

- ✔ Use external locations (ADLS / S3 / GCS)
- Controlled via Unity Catalog

Data Objects

- ✔ Managed Tables

- ✔ External Tables

- ✔ Volumes (for files)

Access Control

- ✔ Centralized permissions
- ✔ Identity-based access

- ✔ Audit-ready

CDU
Cloud And Data Universe

“Unity Catalog is not optional anymore — it’s the foundation of modern Databricks.

CDU
Cloud And Data Universe

Over the next few videos you will learn basics on metastore, its objects and setting up Unity Catalog.

This is important before we move ahead in Databricks & PySpark.

CDU

Cloud And Data Universe

Introduction to DataFrames

DataFrame

CDU
Cloud And Data Universe

What are DataFrames in Spark?

A DataFrame is a distributed collection of data organized into named collumns, similar to table in a database or a DataFrame in Pandas, built to work at big-data scale.

DataFrame

👉 Internally, DataFrames are optimized and distributed across a Spark cluster.

CDU
Cloud And Data Universe

Key Characteristics of DataFrames

1 Schema-Based

- Data has column names and data types
- Enables validation and optimization

2 Distributed

- Data is split into **partitions**
- Processed in **parallel** across executors

3 Immutable

- Any transformation creates a new DataFrame

4 Optimized (Very Important)

- **Catalyst Optimizer** → query optimization
- **Tungsten Engine** → memory & CPU optimization
- 👉 This makes DataFrames **much faster than RDDs.**

CDU
Cloud And Data Universe

DataFrame vs RDD
(Interview Favorite)

Feature	DataFrame	RDD
Schema	Yes	No
Optimization	Automatic	Manual
Performance	Faster	Slower
API level	High-level	Low-level
SQL support	Yes	No

CDU
Cloud And Data Universe

Different Ways to Create a DataFrame in Spark

Broadly, DataFrames can be created from:

1. **Files**
   (csv, json, parquet, delta)

2. **Tables**

ID	Name	Salary

3. **RDDs**

4. **Databases** (using jdbc)

Databases (using jdbc)

Cloud And Data Universe

RDD vs DataFrame vs Dataset — Which to Use When?

Scenario	Best Choice
Unstructured data	RDD
Structured data	DataFrame
SQL / analytics	DataFrame
Maximum performance	DataFrame
Type safety	Dataset
Python users	DataFrame
Fine-grained control	RDD

Conclusion on DataFrames

DataFrames are the **most powerful** and **production-ready abstraction** in **Spark** — **optimized, scalable,** and **built for real-world data workloads.**

Optimized
Leverage Catalyst & Tungsten for peak performance

Scalable
Handle massive datasets with ease

Production-Ready
Industry-proven and widely adopted

Built for Real Workloads
Ideal for ETL, analytics, and reports

RDDs give control, **DataFrames** give **power,** and **optimization** makes the **difference.**

CDU Cloud And Data Universe

Create DataFrame from RDD

CDU
Cloud And Data Universe

Create DataFrame from CSV

CDU

— Cloud And Data Universe