Databricks Fundamentals

Complete Guide to Modern Data Analytics

Yash Kavaiya

14-Day AI Challenge - Day 1

Agenda

- **Introduction to Databricks**
- **Why Databricks?**
  - Pandas vs Hadoop vs Spark
  - Evolution of Big Data
- **Lakehouse Architecture**
  - Data Warehouse vs Data Lake
  - Delta Lake Foundation
- **Workspace Structure**
  - Navigation & Components
  - Compute Resources
- **Industry Use Cases**
  - Netflix, Shell, Comcast
- **Hands-On Tasks**

What is Databricks?

Definition

A **unified analytics platform** built on Apache Spark, providing a collaborative environment for data engineering, data science, and ML workloads.

Key Value Propositions:

- Unified platform for all data needs
- Built on open-source technologies
- Cloud-native (AWS, Azure, GCP)
- Collaborative notebooks
- Enterprise-grade security

Founded by:

- Original creators of Apache Spark
- Creators of Delta Lake
- Creators of MLflow
- From UC Berkeley's AMPLab

The Evolution of Big Data Processing

Key Insight

Each era solved specific problems but introduced new challenges, leading to the next evolution.

Pandas: Strengths & Limitations

Strengths:

- Intuitive API with rich functionality
- Fast for small datasets ($<$ few GB)
- Extensive ecosystem support
- Great for exploratory analysis
- Excellent visualization integration

Limitations:

- **Memory Bound:** Data must fit in RAM
- **Single Machine:** No distribution
- **Vertical Scaling Only**
- **No Lazy Evaluation**
- **Limited Fault Tolerance**

The Memory Problem

If you have a 10GB CSV and 8GB RAM: **Pandas fails!**
Actual memory needed $\approx$ Data Size $\times$ 2 to 5x (due to intermediate copies)

Hadoop: The Original Big Data Solution

Components:

- **HDFS:** Distributed File System
- **MapReduce:** Processing Model
- **YARN:** Resource Management
- **Hive, Pig, HBase:** Tools

Limitations:

- Disk-Based (10-100x slower)
- High Latency (minutes/job)
- Complex MapReduce code
- Batch only, no streaming
- Multiple tools required

The I/O Bottleneck

Every operation involves disk read/write. For iterative ML algorithms:
Time = n $\times$ (Disk Read + Disk Write) $\Rightarrow$ **Prohibitively slow!**

Apache Spark: The Game Changer

Core Innovation: In-Memory Computing

RDDs (Resilient Distributed Datasets) - **Resilient, Distributed, Parallel**

Spark Unified Stack:

- **Spark SQL:** Structured data
- **Spark Streaming:** Real-time
- **MLlib:** Machine Learning
- **GraphX:** Graph processing

Performance:

- **100x** faster than Hadoop (in-memory)
- **10x** faster (on-disk)
- Memory: 100 nanoseconds
- Disk: 10 milliseconds

Hadoop: Read → Process → Write → Read → Process → Write
Spark: Read → Process → Process → Process → Write (final only)

Databricks: Beyond Open-Source Spark

"While Spark is the engine, Databricks is the complete vehicle"

Open-Source Spark:

- Manual cluster configuration
- Base Spark speed
- Jupyter (separate)
- External scheduling tools
- Self-managed Delta Lake

Databricks Platform:

- **One-click clusters**
- **2-5x faster** with Photon
- **Integrated notebooks**
- **Built-in Workflows**
- **Unity Catalog governance**

Databricks Runtime (DBR)

Pre-installed libraries • Photon engine (C++, 2-8x faster) • GPU support • Optimized I/O

Comprehensive Comparison

Criteria	Pandas	Hadoop	Spark	Databricks
Data Size	MB-GB	TB-PB	TB-PB	TB-PB
Processing	Single	Distributed (disk)	Distributed (mem)	Optimized
Speed	Fast (small)	Slow	100x Hadoop	2-5x Spark
Real-time	No	No	Yes	Yes (enhanced)
ML Support	scikit-learn	Mahout	MLlib	MLlib + MLflow
Governance	Manual	Limited	Limited	**Unity Catalog**

Decision Guide

$<$ **10GB:** Pandas • **10GB-1TB + Collaboration:** Databricks • $>$ **1TB + Enterprise:** Databricks

Data Architecture Evolution

Data Warehouse (1980s-2000s):

- Structured data only
- ACID transactions
- SQL-based
- High cost/TB
- No unstructured data

Data Lake (2010s):

- All data types
- Schema-on-read
- Low cost storage
- Poor reliability
- "Data Swamps"

The Two-Tier Problem

Organizations ended up with **BOTH**:
⇒ Data duplication
⇒ Stale data
⇒ Complex pipelines
⇒ High costs
⇒ Governance challenges

Solution
Data Lakehouse!

What is a Data Lakehouse?

Definition

Lakehouse = Data Lake (flexibility, low cost) + Data Warehouse (reliability, governance)

Key Principles:

- **Open Formats:** Parquet, Delta
- **ACID Transactions**
- **Schema Enforcement**
- **Time Travel**
- **Unified Access**
- **Direct Access**

Benefits:

- No vendor lock-in
- Data consistency
- Prevent bad data
- Auditing & rollback
- No data silos
- No data copying

One platform → All workloads: BI, Data Science, ML, Streaming

Delta Lake: The Foundation

What is Delta Lake?

An open-source storage layer that brings **reliability** to Data Lakes.

How It Works:

- Parquet files + Transaction Log
- Each operation logged in JSON
- Enables version tracking
- Atomic commits

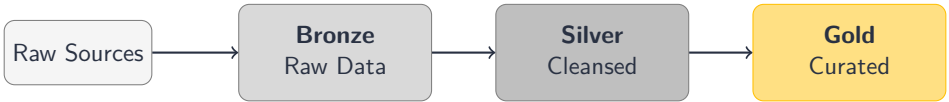
Key Features:

- **ACID Transactions**
- **Time Travel**
- **Schema Evolution**
- **Audit History**
- **MERGE (Upserts)**
- **Optimize & Z-Order**

Time Travel Example

```
SELECT * FROM sales VERSION AS OF 5;
RESTORE TABLE sales TO VERSION AS OF 5;
```

Medallion Architecture



	Bronze	Silver	Gold
Purpose	Store raw data	Clean & validate	Business aggregates
Quality	Low (errors, dupes)	Medium (rules applied)	High (analytics ready)
Schema	Schema-on-read	Schema enforced	Denormalized

Databricks Workspace Structure

Main Navigation:

- **Home:** Personal workspace
- **Workspace:** Shared folders
- **Repos:** Git integration
- **Catalog:** Unity Catalog
- **Workflows:** Job orchestration
- **Compute:** Clusters, Warehouses
- **ML:** Experiments, Models

Compute Types:

- **All-Purpose Cluster:** Development
- **Job Cluster:** Production jobs
- **SQL Warehouse:** SQL analytics
- **Instance Pool:** Pre-warmed

Runtimes:

- Standard, ML, Photon, GPU

Databricks Notebooks

Interactive documents for code, visualizations, and narrative text

Supported Languages:

- `%python` - Default
- `%sql` - Data queries
- `%scala` - Performance
- `%r` - Statistics
- `%md` - Documentation
- `%sh` - Shell commands

Key Features:

- Built-in Visualizations
- Dashboard Widgets
- Real-time Collaboration
- Revision History
- Interactive Parameters

Widget Example

```
dbutils.widgets.dropdown("region", "US", ["US", "EU", "APAC"])
```

Industry Use Case: Netflix

The Challenge

230+ million subscribers • 1+ billion hours/week • Real-time personalization

Implementation:

- Real-time Recommendations
- A/B Testing Platform
- Content Analytics
- Stream Quality Monitoring

Scale:

- **1.5 trillion** events/day
- **100+ PB** in data lake
- Updates every **10 seconds**
- **1000s** of A/B tests

Tech Stack: Kafka → Spark Streaming → Delta Lake → ML Models

Hands-On Tasks

1. **Create Databricks Community Edition Account**
   - Visit community.cloud.databricks.com
   - Sign up with email
2. **Navigate Workspace, Compute, Data Explorer**
   - Explore left navigation panel
   - Create a cluster
3. **Create Your First Notebook**
   - New → Notebook
   - Attach to cluster
4. **Run Basic PySpark Commands**
   - `spark.range(10).show()`

Key Takeaways

Remember

- Databricks = Unified Analytics Platform
- Lakehouse = Lake + Warehouse benefits
- Delta Lake = ACID + Time Travel
- Medallion = Bronze → Silver → Gold

Why Databricks?

- **100x** faster than Hadoop
- **2-5x** faster than Spark
- **Unified** governance
- **Collaborative** notebooks

Start your Databricks journey today!

Thank You!

Questions?

Connect with me:

linkedin.com/in/yashkavaiya

Gen AI Guru