Data Engineering Report

Topic: Hadoop, Spark, Data Lifecycle, Governance, and Consumption

Date: Jan 16, 2026
Purpose: To understand how large-scale data is stored, processed, governed, tested, and delivered for analytics and machine learning.

1. Introduction to Data Engineering

Data Engineering focuses on **building systems that collect, store, process, and deliver data** in a reliable and scalable way.
The end goal is not storage or processing alone, but **making data usable for analytics, dashboards, and machine learning**.

A typical data pipeline handles:

- Large data volumes
- Different data formats
- Batch and real-time workloads
- Security, quality, and governance requirements

2. Big Data Ecosystem Overview

The Big Data ecosystem consists of multiple tools, each solving a specific problem:

- **Hadoop**
  - Handles distributed storage (HDFS)
  - Supports batch processing using MapReduce
- **Hive**
  - Provides SQL-like querying on Hadoop data
- **HBase**
  - NoSQL database for fast read/write on Hadoop
- **Spark**
  - Unified analytics engine
  - Supports batch, streaming, ML, and graph processing
- **Kafka**
  - Handles real-time event streaming

Together, these tools form the backbone of modern data platforms.

3. Hadoop Architecture

3.1 What Hadoop Solves

Hadoop is designed to:

- Store massive datasets
- Process data across many machines
- Handle hardware failures gracefully

It follows a **disk-based distributed architecture**, which prioritizes reliability over speed.

3.2 HDFS (Hadoop Distributed File System)

HDFS is responsible for **data storage**.

Key characteristics:

- Files are split into large blocks (128MB / 256MB)
- Blocks are distributed across nodes
- Each block is replicated (default: 3 copies)

Nodes:

- **NameNode**
  - Maintains metadata only
  - Tracks file names, block locations, permissions
- **DataNode**
  - Stores actual data blocks
  - Uses external disk storage

Failure handling:
If a DataNode fails, HDFS automatically:

- Detects missing blocks
- Restores replicas from other nodes

This makes HDFS **very good for backup and fault tolerance**.

3.3 MapReduce Processing Model

MapReduce handles **batch computation**.

Steps:

1. **Map phase**

- Processes chunks of data independently
- Converts input into key-value pairs

2. **Reduce phase**
    - Aggregates mapped results
    - Produces final output

Limitation:

- After each step, results are written to disk
- High disk I/O → slower performance

Result:

Hadoop works well for:

- Large batch jobs
- Historical data processing
  But performs poorly for:
- Iterative algorithms
- Real-time analytics
-

4. Spark Architecture (Detailed)

Spark was introduced to overcome Hadoop's performance limitations.

4.1 Core Principle

In-memory computation

Spark loads data once into memory and performs multiple operations without repeatedly reading from disk.

4.2 Spark Components

- **Driver Program**
    - Central coordinator
    - Builds the execution plan (DAG)
    - Communicates with cluster manager
- **Cluster Manager**
    - Allocates CPU and memory
    - Examples: YARN, Kubernetes
- **Executors**
    - Run tasks on worker nodes
    - Cache data in memory

4.3 RDD (Resilient Distributed Dataset)

RDD is the fundamental Spark data structure.

Properties:

- Distributed across nodes
- Stored in memory
- Fault tolerant

Instead of replication, Spark uses **lineage**:

- If data is lost, Spark recomputes it from previous steps

4.4 DAG (Directed Acyclic Graph)

Spark builds a DAG of transformations:

- Operations are **lazy**
- Execution happens only when an action is triggered
- DAG allows Spark to optimize execution

This makes Spark:

- Faster
- More efficient
- Better for ML and analytics
-

5. Hadoop vs Spark (Architectural Comparison)

Aspect	Hadoop	Spark
Processing	Disk-based	In-memory
Speed	Slower	Faster
Workloads	Batch only	Batch + Real-time
Fault tolerance	Replication	Lineage
Use cases	ETL, backups	ML, analytics, streaming

Practical explanation:

- Hadoop is reliable but slow
- Spark is optimized for modern, interactive workloads

6. Batch vs Real-Time Processing

Batch Processing

- Data processed at scheduled intervals
- Examples:
    - Monthly reports
    - Historical analysis

Real-Time Processing

- Data processed as events occur
- Examples:
    - Fraud detection
    - Live pricing
    - Streaming dashboards

Spark supports **both**, making it suitable for modern data platforms.

7. Data Engineering Lifecycle

1. **Data Collection**
    - APIs
    - Files
    - Event streams
2. **Ingestion**
    - Batch or streaming ingestion
3. **Processing**
    - Cleaning
    - Transformation
    - Validation
4. **Storage**
    - HDFS
    - Data lake
    - Data warehouse
5. **Consumption**
    - Dashboards
    - Reports
    - ML models

8. Data Delivery & APIs

Processed data must be **served** to consumers.

- Tools used:
    - Flask
    - FastAPI

APIs allow:

- Applications to fetch data
- ML models to return predictions
- Dashboards to consume data securely

9. Data Governance & Security

Data Lineage

- Tracks data origin and flow
- Helps with debugging and compliance

Data Cataloging

- Central metadata repository
- Tools:
    - AWS Glue Data Catalog
    - Apache Atlas

Provides:

- Table descriptions
- Column meanings
- Ownership details
- Update history

Security Techniques

- **Masking**
    - Hides sensitive data
    - Example: credit card masking
- **Encryption**
    - Restricts access to authorized users only

10. Data Quality & Testing

Ensures trust in data pipelines.

- **Validation**
    - Schema and rule checks

- **Great Expectations**
    - Data quality assertions
- **Unit Testing**
    - Pytest for ETL logic

11. Data Visualization & Reporting

BI Tools

Power BI

- Data modeling
- DAX measures
- Dashboard design

Tableau

- Data connections
- Calculations
- Interactive dashboards

Reporting Best Practices

- Define clear **KPIs**
- Automate data refresh
- Use storytelling to explain insights

12. Serving Data for ML & Analytics

Clean and validated data is used for:

- Model training
- Real-time predictions
- Business analytics

This is the **final purpose** of the entire data engineering pipeline.

13. Final Conclusion

- Hadoop provides reliable distributed storage and batch processing
- Spark enables fast, in-memory analytics and real-time processing
- Governance, quality, and security ensure trustworthy data
- Visualization and APIs convert data into business value

Data Engineering is about enabling decisions, not just moving data.