Name: Kashmira Chaudhari
Stud ID: 20158

Data Engineering with Python

Building Scalable Data Pipelines

Objectives

What is Data Engineering
Why use python in data Engineering
Data Engineering workflow
Role of python in Data Engineering
What is Data Streaming
Apache Kafka
Docker Overview
Apache Airflow
Comparison with other languages
Advantages of using Python
Conclusion

What is Data Engineering?

Data Engineering involves designing, building, and maintaining systems for collecting, storing, and processing data.

Key Components:

- Data Collection
- Data Storage
- Data Transformation
- Data Pipeline Automation

Goal: Ensure data is available, reliable, and accessible for analysis and decision-making.

Why Use Python in Data Engineering?

Easy to Learn: Simple syntax and readability.

Versatile: Supports end-to-end data engineering workflows.

Extensive Libraries: Rich ecosystem of libraries and frameworks.

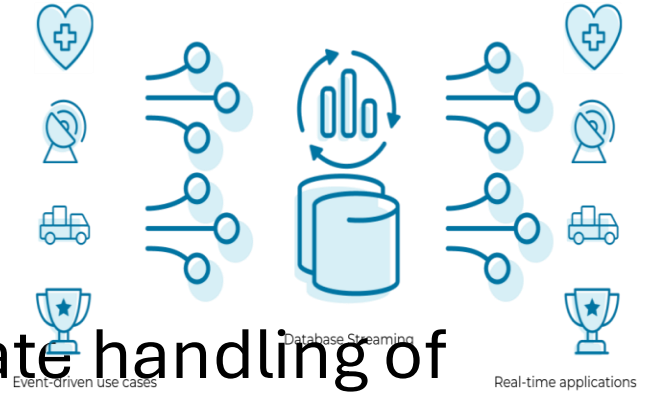
Community Support: Large, active developer community.

Scalability: Works for both small and large-scale data systems.

Data Engineering Workflow

Data Collection	• From APIs, databases, files, and cloud storage.
Data Cleaning & Transformation	• Fixing data quality issues, filtering, and aggregating.
Data Storage	• SQL databases, NoSQL databases, data warehouses.
Building Pipelines	• Automating data flow using tools like Apache Airflow.
Monitoring & Logging	• Ensuring pipeline reliability and handling errors.

What is Data Streaming?



- **Real-time data processing** enables immediate handling of incoming data as it arrives, allowing systems to react without delay. **Low-latency insights and actions** ensure quick data processing, providing fast responses and decisions with minimal delay.

Technologies Involved:
Kafka for data streaming
Docker for containerization
Airflow for workflow orchestration
Python for scripting and automation

Apache Kafka

What is Kafka?

Distributed streaming platform

Handles real-time data feeds

Key Components:

Topics: Streams of records

Producers and **Consumers**

Brokers and **Zookeeper**

Use Cases:

Real-time analytics, monitoring, and event-driven applications

Docker Overview

- **What is Docker?**
  - Containerization platform for deploying applications
- **Why Use Docker?**
  - Ensures consistency across environments
  - Simplifies dependency management
- **Key Concepts**:
  - **Containers**, **Images**, **Dockerfiles**, and **Docker Compose**

Zookeeper

- Zookeeper is a distributed coordination service that manages metadata, leader election, and configuration for distributed systems like Kafka.

Broker Management: Zookeeper maintains a list of Kafka brokers and tracks their availability.

Leader Election: Zookeeper coordinates the leader election for partitions, ensuring high availability and data consistency.

Notifications: Zookeeper notifies Kafka of significant changes, such as:

When a new topic is created.

When a broker goes down or comes up.

When topics are deleted.

Broker

A Kafka broker is a server that stores messages, manages client requests, handles partition data and replication, and works with other brokers in a cluster for scalability and fault tolerance.

Role of a Kafka Broker:

- **Message Storage**:
Brokers store messages in partitions for each topic and manage data retention policies.
- **Client Communication**:
Brokers handle requests from producers (to publish messages) and consumers (to fetch messages).
- **Partition Management**:
They manage topic partitions and ensure data is replicated across other brokers for fault tolerance.
- **Load Balancing**:
Brokers distribute partitions across the cluster to balance the load and ensure efficient data handling.

Kafka Producer

- A **producer** in Kafka is a client application responsible for **publishing (sending) messages** to Kafka topics.

Role of a Producer:

- **Message Publishing**:
Sends data (messages) to specified topics in the Kafka cluster.
- **Partition Assignment**:
Decides which partition to send a message to, either by specifying a key or using Kafka's default
- partitioning strategy.
- **3. Acknowledgments (Delivery Guarantees)**:
Receives acknowledgments from brokers to ensure messages are successfully delivered,
- offering different levels of reliability (e.g., acks=0, acks=1, acks=all).
- **4. Batching and Compression**:
Optimizes performance by batching messages and optionally compressing them before sending.

Kafka Consumer

A **consumer** in Kafka is a client application responsible for **reading (consuming) messages** from Kafka topics.

Role of a Consumer:

- **Message Consumption**
- **Consumer Group Coordination**
- **Offset Management**
- **Fault Tolerance**

Apache Airflow

Apache Airflow is an open-source platform for **orchestrating, scheduling, and monitoring workflows**.

It allows users to define, manage, and automate complex data pipelines and tasks.

Role of Airflow:	**Workflow Orchestration**
	Task Scheduling
	Monitoring and Logging
	Error Handling and Retry Logic
	Integration with External Systems

Comparison with Other Languages

Feature	Python	Java	Scala
Ease of Use	High	Medium	Low
Performance	Moderate	High	High
Big Data Support	Excellent (PySpark)	Excellent (Hadoop)	Excellent (Spark)
Learning Curve	Low	Medium	High

Advantages of Using Python

Ease of Use: Readable syntax.

Open Source: Free and widely adopted.

Rich Libraries: Tools for every data task.

Cross-Platform: Works on different systems.

Integration: Connects with databases, big data tools, and cloud services.

Conclusion

PYTHON IS AN ESSENTIAL TOOL FOR MODERN DATA ENGINEERING.

ITS FLEXIBILITY, LIBRARIES, AND COMMUNITY MAKE IT IDEAL FOR BUILDING RELIABLE AND SCALABLE DATA PIPELINES.

KEY TAKEAWAY: PYTHON SIMPLIFIES AND ACCELERATES THE DATA ENGINEERING WORKFLOW.

Thank You