DataOps Pipeline Project Report

Project Overview

This project implements a complete data pipeline utilizing Docker-based services for ingesting, processing, transforming, and visualizing data. The technologies used include Kafka, Logstash, Elasticsearch, Kibana, and a custom Python producer. The pipeline follows the ELK stack pattern with Kafka acting as the message broker.

Objective

The objective is to build a real-time data ingestion and processing pipeline that:
- Ingests raw and potentially malformed data from a Python script.
- Publishes the data to a Kafka topic (test_pipeline).
- Uses Logstash to consume, clean, and transform the data.
- Indexes the processed data into Elasticsearch.
- Uses Kibana to visualize and validate the stored data.

System Components

- Kafka & Zookeeper: Kafka buffers and transmits data. Zookeeper supports Kafka coordination.
- Elasticsearch: Search and analytics engine on port 9200.
- Logstash: Transforms and cleans data from Kafka.
- Kibana: Visualizes data on port 5601.
- Producer (Python): Sends raw JSON data to Kafka.
- Docker Compose: Orchestrates all services.

Key Files Summary

- docker-compose.yml: Connects and configures all services.
- Dockerfile: Builds custom producer container.
- logstash.conf: Contains transformation logic.
- producer.py: Generates sample data.
- requirements.txt: Python dependencies.

Expected Data Flow

Producer -> Kafka -> Logstash -> Elasticsearch -> Kibana

Example Input Data (Producer)

[{"id": "123", "full_name": "Ali", "timestamp": "2025/01/31 12:34:56", "active": "true"}, {"id": "124", "full_name": " ", "timestamp": "2025-01-31T14:20:00Z", "active": "false", "extra_field":"remove"}]

Expected Output (Elasticsearch)

[{"id": 123, "name": "Ali", "timestamp": "2025-01-31T12:34:56Z", "active": true}, {"id": 124, "name": null, "timestamp": "2025-01-31T14:20:00Z", "active": false}]

Validation & Visualization

DataOps Pipeline Project Report

Data is verified via Kibana and Elasticsearch queries. Ensures index creation, data transformation, and cleanup.

Challenges Faced

- Service initialization order
- Date format parsing
- Field cleanup and transformation
- Docker networking setup

Deployment Instructions

1. Run: docker-compose up --build
2. Access:
   - Kafka topic created automatically
   - Elasticsearch: http://localhost:9200/test_pipeline/_search
   - Kibana: http://localhost:5601

Project Deliverables

- Python producer script
- Dockerized services
- Kafka topic creation
- Logstash transformation logic
- Elasticsearch integration
- Kibana dashboard
- Validated output data
- All components run with one docker-compose up