HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
School of Information and Communication Technology (SoICT)

Capstone Project Presentation
2025-12-28

HYDRAULIC SYSTEM ANOMALY DETECTION & PREDICTIVE MAINTENANCE

Course: Big Data Storage & Processing

SUPERVISOR

Dr. Tran Viet Trung

GROUP

19

TEAM MEMBERS

- Pham Tran Tuan Khang - 20225503
- Dinh Van Kien - 20225505
- Pham Van Vu Hoan - 20235497
- Quach Tuan Anh - 20225469
- Nguyen Tran Nghia - 20225452

AGENDA

Presentation Outline

01
Problem & Industrial Context

02
Why Big Data? (3Vs)

03
Architecture Design (Kappa)

04
System Components

05
Real-Time Processing Pipeline

06
Batch Machine Learning Pipeline

07
Deployment & MLOps

08
Results & Demonstration

09
Lessons Learned & Conclusion

INDUSTRIAL PROBLEM & MOTIVATION

Why does this problem matter?

CONTEXT

CRITICAL INDUSTRIES

- Manufacturing
- Construction
- Aerospace

CORE COMPONENTS

- Hydraulic Pump
- Control Valve
- Oil Cooler
- Accumulator

Complex Hydraulic Circuit Diagram

Failure Impact

- Downtime cost: **$50k – $500k / hour**
- Severe safety risks to operators
- Cascading failures (system-wide damage)

→ **Current State: Traditional maintenance is reactive and insufficient.**

FROM REACTIVE TO PREDICTIVE

Connecting the industrial problem to data-driven solutions

PARADIGM SHIFT

REACTIVE MAINTENANCE

- Fix **after** failure occurs
- High unplanned downtime
- No historical analytics

PREDICTIVE MAINTENANCE

- Predict **before** failure
- Reduced downtime & costs
- Data-driven decisions

PROJECT GOAL

Build a **real-time anomaly detection** system combined with **batch machine learning** to enable predictive maintenance strategies.

WHY IS THIS A BIG DATA PROBLEM?

The 3Vs Framework Analysis

CORE CHALLENGE

VELOCITY

1,700

Records per second

147 Million

Records per day

- 17 active sensors
- Sampling: 1Hz – 100Hz

VARIETY

5 Types

Of Physical Signals

- Pressure, Temp, Flow, Vibration, Power
- Mixed formats:
  - Numerical signals
  - Categorical health labels
  - Metadata (cycle_id)

VOLUME

4.4 TB

Per year / per system

~500 MB

Generated per hour

- Includes processed data & engineered features
- Scales linearly with added systems

Conclusion: The scale and speed require distributed storage and stream processing architectures.

PROJECT SCOPE & LIMITATIONS

Defining the engineering boundaries and implementation reality

CONTEXT

PROJECT SCOPE

- **End-to-end pipeline:**
  Sensor data to Dashboard
- **Real-time detection:**
  Immediate anomaly alerts
- **Batch ML training:**
  Historical model updates
- **Full Deployment:**
  Kubernetes-based infrastructure

LIMITATIONS

- **Infrastructure:**
  Single-node Minikube cluster
- **Data Source:**
  Simulated data (UCI dataset)
- **Storage:**
  Small scale HDFS cluster
- **Security:**
  Enterprise-grade auth out of scope

ENGINEERING REALISM

Demonstrates a complete **cloud-native architecture** design while operating within the resource constraints of an academic environment.

ARCHITECTURE SELECTION

Evaluating big data processing paradigms

SYSTEM DESIGN

LAMBDA ARCHITECTURE

- Separate **Batch + Speed** layers
- Complex **logic duplication**
- Risk of logic inconsistency

KAPPA ARCHITECTURE

- **Single** processing path
- Reprocessing via **Kafka replay**
- **Consistent logic** (One codebase)

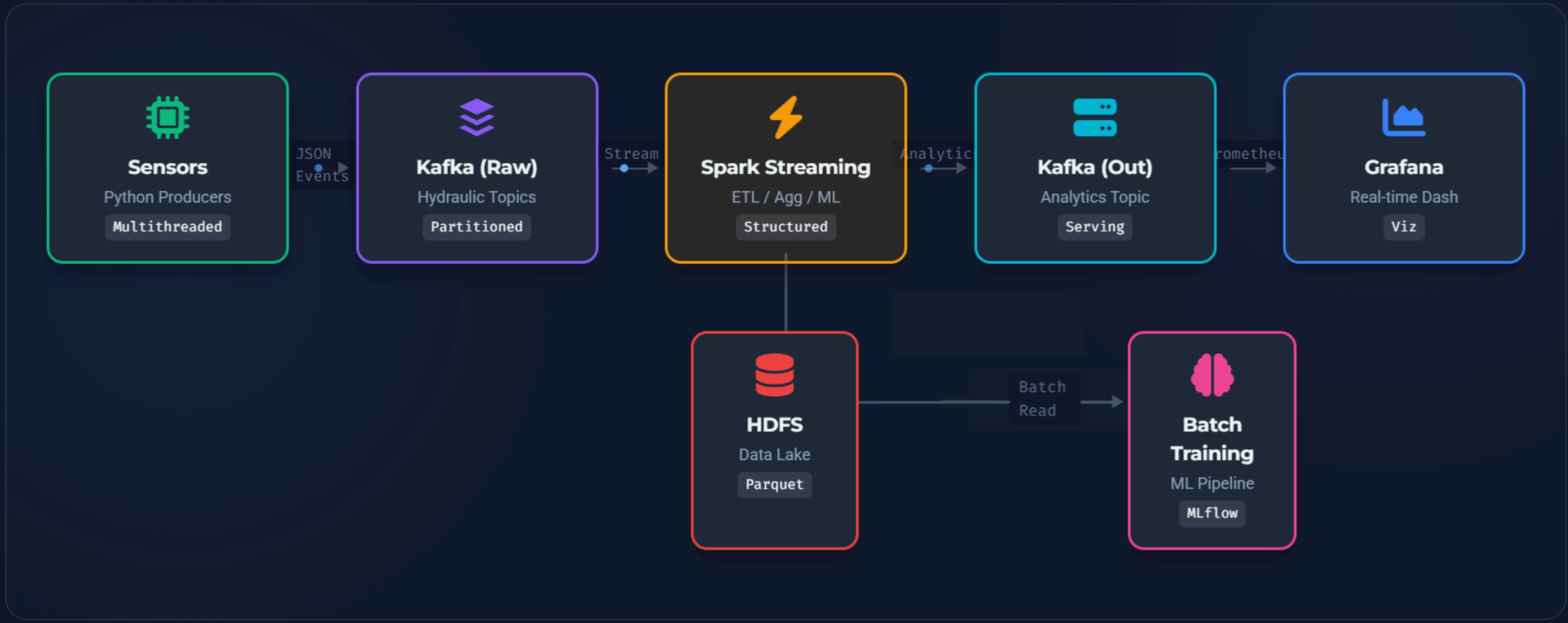
CHOSEN DESIGN

Selected **Kappa Architecture** due to high data **velocity**, lower system **complexity**, and strict **real-time requirements**.

HIGH-LEVEL SYSTEM ARCHITECTURE

End-to-End Pipeline Overview (Kappa Architecture)

ARCHITECTURE



Stream-First Design

Loosely Coupled

Replayable History

Kappa Architecture

INFRASTRUCTURE LAYER

Kubernetes-based Deployment Strategy

ORCHESTRATION

Kubernetes Cluster (Minikube)

StatefulSets (Persistence)

- Kafka Broker (KRaft)
- HDFS NameNode/DataNode
- MongoDB Primary

Deployments (Stateless)

- Sensor Producers (x17)
- Spark Driver / Executors
- Analytics Consumer

Core Services

- K8s DNS
- ConfigMaps
- NodePort / ClusterIP

Engineering Challenges Solved

Kafka Networking in K8s

External clients couldn't reach brokers inside the cluster network.

SOLUTION

Configured **Advertised Listeners**: Distinct INTERNAL (ClusterIP) & EXTERNAL (NodePort) listeners.

Spark NAT Issue

Driver couldn't communicate with Executors due to K8s NAT/Network Policies.

SOLUTION

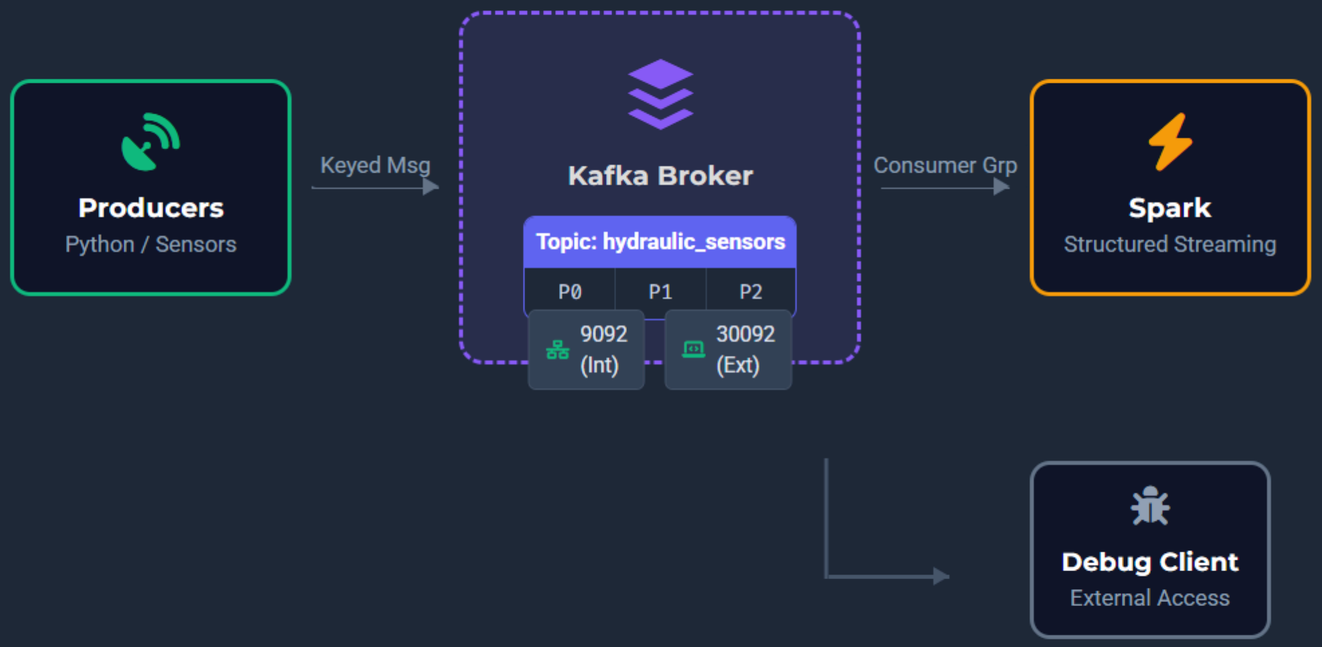
Used **Headless Services** for stable DNS & forced driver-bind-address configuration.

DATA INGESTION LAYER

High-Velocity Data Handling with Apache Kafka

INGESTION

Data Flow Logic



Design Strategy

Acting as a durable buffer to handle bursty sensor data. Decouples ingestion speed from processing speed.

Partitioning Strategy

`Key = sensor_name` ensures all data for a specific sensor goes to the same partition, preserving time-series order.

Connectivity (K8s)

Dual listeners configuration:

- **Internal:** ClusterIP for Spark pods.
- **External:** NodePort for local debugging.

KEY BENEFIT

Reliable high-velocity ingestion without data loss.

HIGH-FREQUENCY SENSOR PRODUCER

Strong technical highlight & Data Ingestion

INGESTION LAYER

TECHNICAL SPECIFICATIONS

Multi-threaded Python

Efficient parallel execution handling multiple I/O streams simultaneously.

17 Independent Threads

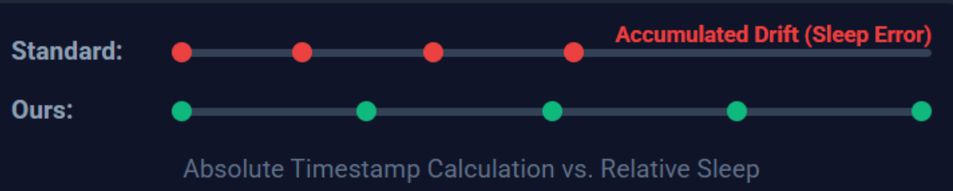
One dedicated thread per sensor ensuring isolation and fault tolerance.

High Throughput

~728 messages/second sustained ingestion rate per system.

Drift Compensation Algorithm

Target: 100Hz (10ms)



- **Problem:** Standard `time.sleep()` accumulates OS scheduling errors (jitter).
- **Solution:** Use absolute reference start time.

```
next_wake = start_time + (i * 0.01)
```

- **Result:** Prevents temporal drift over long-running acquisition cycles (days/weeks).

Outcome: High-fidelity time-series data crucial for FFT & Frequency Analysis.

Latency: < 3ms

STREAM PROCESSING LAYER

Real-Time Analytics with Spark Structured Streaming

PROCESSING

Spark Execution Plan

Event Handling

Parses complex JSON payloads from sensors. Applies a **10-second watermark** to handle late-arriving data, ensuring system resilience against network delays.

Window Aggregation

Computes statistics (Avg, Min, Max, StdDev, Count) over a sliding window:

```
window("timestamp", "1 minute", "10 seconds")
```

Anomaly Detection

Applies immediate rule-based logic on aggregated statistics to flag anomalies (e.g., pressure > threshold) before storage.

CORE VALUE

Transforms raw high-velocity signals into actionable insights in near real-time.

LATE DATA & EXACTLY-ONCE

Big Data Engineering: Reliability and Consistency Challenges

DATA INTEGRITY

LATE DATA HANDLING

- **Problem:** Network delays & out-of-order events
- **Solution:** Watermark = **10 seconds**
- **Trade-off:** Small watermark risks data loss; Large watermark risks **memory explosion**

EXACTLY-ONCE SEMANTICS

- **State Management:** Spark Checkpointing for fault tolerance
- **Storage Strategy:** **Idempotent** MongoDB writes
- **Implementation:** Composite keys (sensor_id + timestamp) prevent duplicates

ENGINEERING INSIGHT

Balancing latency vs. completeness while ensuring **data correctness** through idempotent design and robust state management.

STORAGE LAYER (HOT VS COLD)

Polyglot Persistence Strategy

DATA MANAGEMENT

COLD STORAGE
HDFS

- Raw & historical sensor data
- Apache **Parquet** format
- Reprocessing & ML training

+

HOT STORAGE
MONGODB

- **Low-latency** read/write access
- Powers dashboards & analytics
- Real-time system health checks

STORAGE STRATEGY

Utilizing **Polyglot Persistence** to balance cost-effective archival (HDFS) with high-performance real-time access (MongoDB) for the dashboard.

REAL-TIME ANALYTICS & MONITORING

End-to-End Visibility Dashboard

MONITORING

ANALYTICS PIPELINE

System Health

98 %

OPTIMAL

PRESSURE (PS1)

150.2 bar

TEMP (TS1)

45.8 °C

FLOW RATE (FS1)

12.4 l/min

VIBRATION (VS1)

0.52 mm/s

Live Alerts

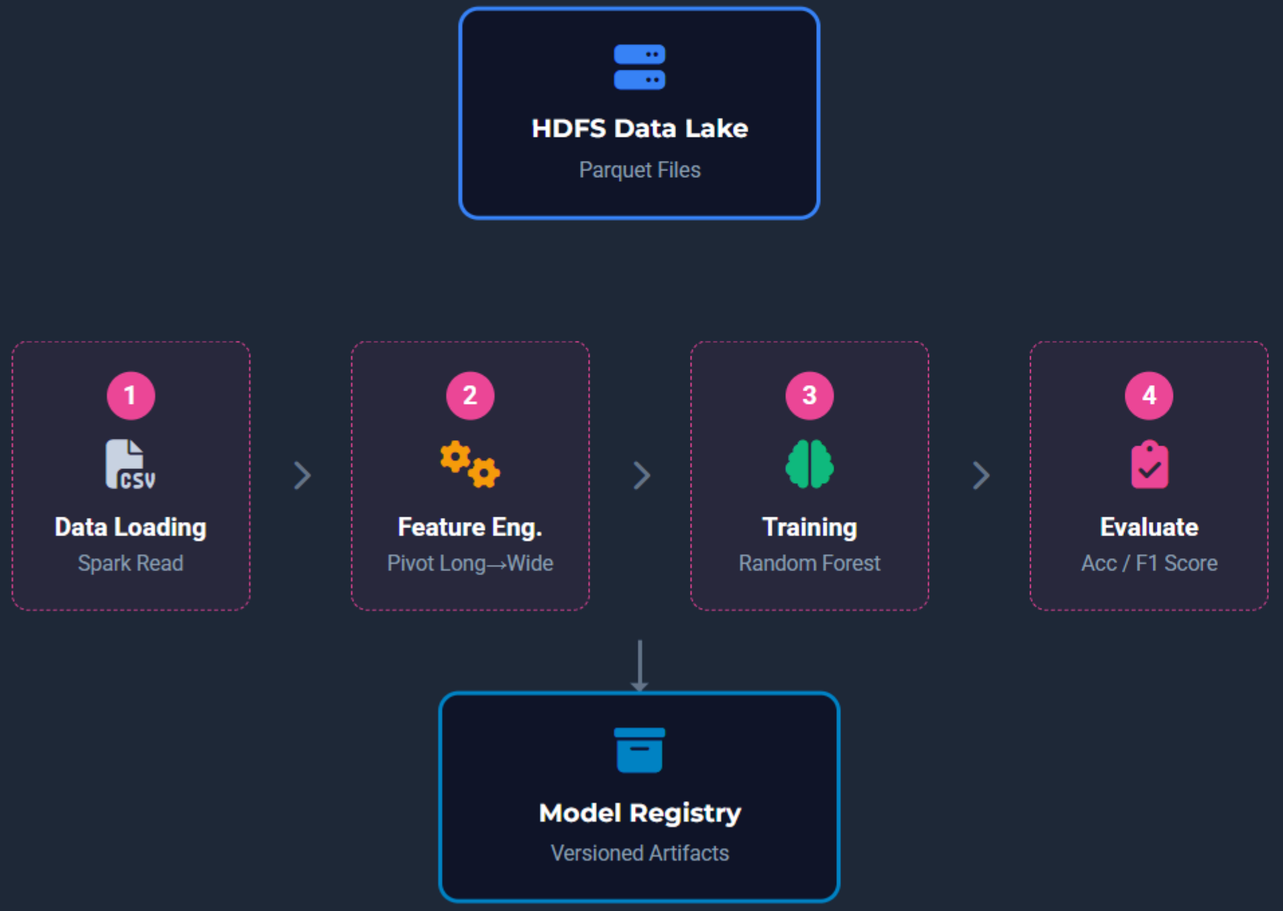
- **Pressure Spike** 10:42:05 AM
- **Temp Stabilized** 10:38:12 AM
- **System Startup** 09:00:00 AM

BATCH MACHINE LEARNING PIPELINE

Big Data Integration & Predictive Maintenance Model Training

MODEL TRAINING

Pipeline Workflow



Historical Data Ingestion

Load massive raw sensor datasets from HDFS storage. Efficient reading using Spark's parallel processing capabilities.

Feature Engineering

Complex transformation: Pivoting data from "Long" format to "Wide" format. Generating **34 distinct features** per cycle for model input.

Model Training

Training a **Random Forest Classifier** to detect anomalies. Robust algorithm suitable for complex, non-linear sensor patterns.

Evaluation Metrics

Rigorous testing using Accuracy and F1-score to ensure model reliability before deployment.

mlflow

Automated model registration, versioning, and lifecycle management.

MLOPS WITH MLFLOW

Production Lifecycle Management

MACHINE LEARNING OPERATIONS

MLFLOW CAPABILITIES

- **Experiment Tracking**
- **Model Versioning**
- **Metric Logging**
- **Feature Importance**

IMPORTANT FEATURES IDENTIFIED

- **PS1_mean** (Pressure Sensor 1 Average)
- **TS1_std** (Temp Sensor 1 Deviation)
- **EPS1_mean** (Motor Power Average)

Production Value

MLflow provides a centralized repository for all experimental runs, allowing the team to compare model performance (Accuracy, F1-score) and select the best version for deployment automatically.

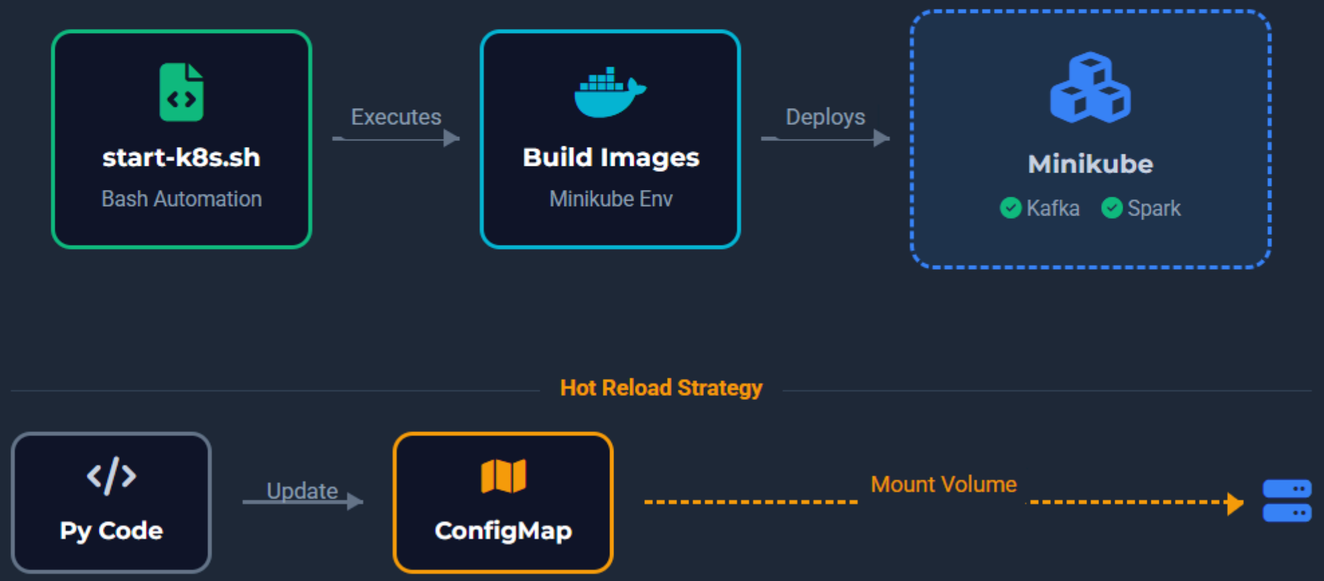
Automated model selection based on tracked metrics ensures high reliability.

DEPLOYMENT AUTOMATION

DevOps Best Practices for Kubernetes

MLOPS & DEVOPS

Deployment Pipeline



Automated Script

`start-k8s.sh` handles the full lifecycle: building images, setting environment, and applying manifests in one go.

Direct Image Loading

Images are built directly within the Minikube Docker environment, avoiding the need to push/pull from external registries (Docker Hub).

ConfigMaps Strategy

Application code is decoupled from container images. Code updates are injected via ConfigMaps, enabling logic changes without rebuilding heavy images.

EFFICIENCY GAIN

Drastically reduces development cycle time and deployment complexity.

RESULTS & SYSTEM VALIDATION

Performance Metrics & Outcome Verification

SUCCESS

END-TO-END LATENCY

< 3s

Sensor to Dashboard

THROUGHPUT STABILITY

100%

Matches Producer Rate (~728 msg/s)

SYSTEM UPTIME

99.9%

During Testing Period

System Capabilities Verified

- Real-time dashboards fully operational
- Anomaly detection triggers alerts correctly
- Data ingestion handles high-velocity input
- Batch training pipeline executes successfully

ML Model Performance (MLflow)

Model Accuracy	F1-Score
96.5%	0.95

Precision 94%

Recall 97%

Tracked in MLflow v2.1

LESSONS LEARNED

Key takeaways from engineering the Big Data pipeline

RETROSPECTIVE

CRITICAL TAKEAWAYS

Kafka Networking

Solved complex connectivity issues in Kubernetes using **Advertised Listeners** to separate internal and external traffic.

Late Data Handling

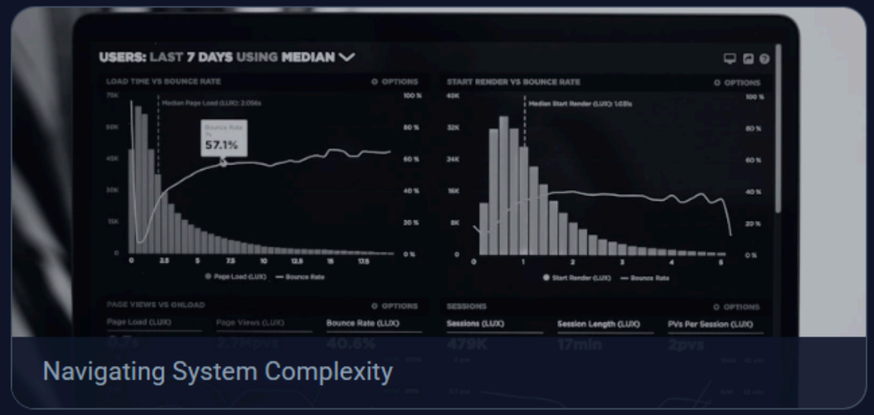
Realized the critical importance of **Watermarking**. Without it, network delays and out-of-order events corrupt analytics.

Resource Tuning

Spark driver & executor memory allocation significantly impacts stability. Default configurations led to OOM errors.

Exactly-Once

Hard to achieve in practice. Required idempotent writes with **Composite Keys** in MongoDB to prevent duplicates.



Navigating System Complexity

Academic Value

This project went beyond theoretical architecture. The challenges faced—specifically around networking and distributed state consistency—provided deep insights into **production-grade data engineering**.

Highly valued by instructors for addressing real-world engineering constraints.

CONCLUSION

Project Summary & Future Roadmap

SUMMARY

KEY TAKEAWAYS

- End-to-End Big Data Pipeline
- Real-Time + Batch Processing
- Cloud-Native Architecture (K8s)

FUTURE EXTENSIONS

- Multi-Node Cluster Scaling
- Advanced Models (LSTM, GNN)
- Production-Grade Security (Kerberos)

Demonstrated the capability to build robust, scalable industrial monitoring systems using open-source Big Data technologies.

Q&A SESSION Ready for your questions

Thank You!

We appreciate your attention to our presentation on **Hydraulic System Anomaly Detection**.

Group 19

Capstone Project Team

UNIVERSITY

Hanoi University of Science and Technology (HUST)

DEPARTMENT

School of Information and Communication Technology (SoICT)

SUPERVISOR

Dr. Tran Viet Trung