# Marine Pollution Tracking

Lorenzo Domenico Attolico (255120)
(lorenzo.attolico@studenti.unitn.it)

Andrea Mauro (254955)
(andrea.mauro@studenti.unitn.it)

Marco Rossi (256088)
(marco.rossi-6@studenti.unitn.it)

Walter Scafa (256938)
(walter.scafa@studenti.unitn.it)

Big Data Technologies Class, Year 2025

Submission: 17/07/25

# Abstract (Overview)

The Marine Pollution Monitoring System is an advanced real-time environmental intelligence platform that integrates streaming data from marine sensor buoys and satellite imagery to detect, monitor, and forecast pollution events in Chesapeake Bay.

This solution combines synthetic data simulating buoy sensors with real Sentinel-2 satellite imagery (with simulated pollution effects applied) through a medallion architecture featuring Bronze (raw ingestion), Silver (processed/standardized), and Gold (analytics-ready) data layers. The system includes four experimental ML models designed to incorporate fluid dynamics principles, pollutant physical properties, and environmental patterns, with the aim of predicting potential pollution diffusion patterns. Their effectiveness requires validation with real-world data.

**Key achievements include:**

- **Production-Ready Pipeline**: A robust microservices ecosystem of 11 services handling real-time data ingestion, multi-layered processing, and scalable storage with advanced visualization capabilities
- **Predictive Intelligence**: Four ML models forecasting pollution dynamics using seasonal wind patterns, water currents, and pollutant-specific characteristics
- **Real-Time Dashboard**: Interactive Streamlit interface providing live visualization of pollution hotspots, key metrics, and alert statuses with geospatial overlays
- **Enterprise Deployment**: Full containerization with Docker and docker-compose orchestration, featuring health checks, structured logging, and volume persistence for operational resilience

The platform demonstrates successful integration of multiple data sources, advanced analytics, and real-time monitoring capabilities for marine environmental protection.

## Problem Statement & Motivation

Our project addresses a critical environmental challenge: the fragmented and non-standardized monitoring of marine pollution, which hinders rapid identification and response to pollution events.

**The problem we're solving:**

- While monitoring infrastructure exists, real-time integration and standardized analysis across multiple data sources (IoT sensors, satellite imagery, agency databases) remains fragmented, limiting rapid pollution event identification and coordinated response efforts
- Manual analysis of these disparate data causes significant delays in responding to pollution incidents
- Without a unified monitoring infrastructure, critical pollution patterns remain undetected until they become severe, and environmental agencies lack the predictive capabilities needed to anticipate how these contaminants will spread across marine ecosystems over time

**Why this matters:**

- Chesapeake Bay and other aquatic ecosystems are threatened by multiple pollution sources requiring rapid identification and intervention
- Environmental agencies need accurate and timely data to effectively allocate resources for pollution mitigation
- Policy makers require a comprehensive view of pollution trends to develop effective regulations

**Our Solution**: We developed a real-time big data platform that integrates buoy sensors and satellite imagery through Flink streaming pipelines. The system analyzes water quality parameters (pH, dissolved oxygen, turbidity, microplastics, contaminants) using ML models based on fluid dynamics to detect pollution hotspots and predict their spread.

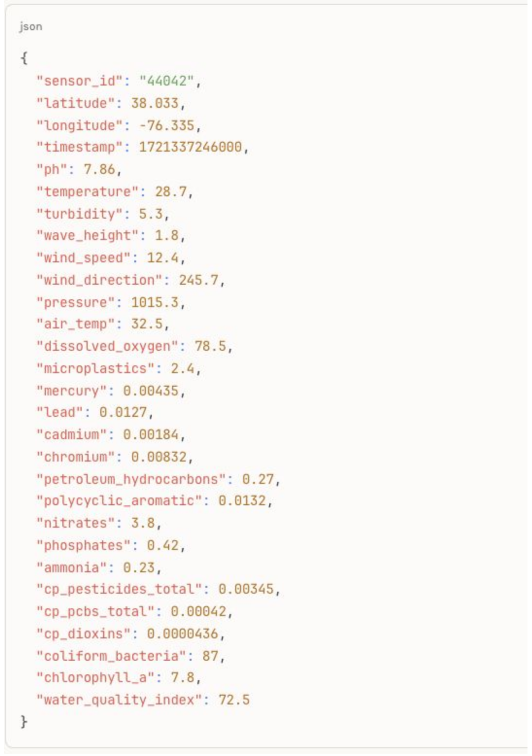
**Impact**: This transforms environmental monitoring from reactive to predictive, enabling agencies to rapidly detect threats, plan targeted interventions, and protect marine ecosystems and public health.

# Data exploration insights

1. **Buoy & Marine Station Data (Simulated)**

Generation & Ingestion

- Outputs JSON providing measurements of water quality parameters, chemical contaminants, physical properties and microplastic concentrations



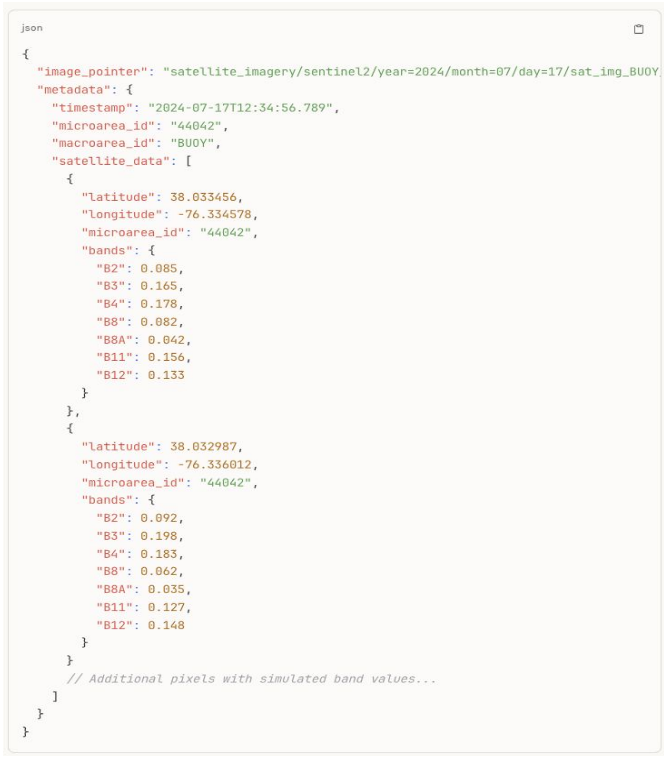
json

```json
{
  "sensor_id": "44042",
  "latitude": 38.033,
  "longitude": -76.335,
  "timestamp": 1721337246000,
  "ph": 7.86,
  "temperature": 28.7,
  "turbidity": 5.3,
  "wave_height": 1.8,
  "wind_speed": 12.4,
  "wind_direction": 245.7,
  "pressure": 1015.3,
  "air_temp": 32.5,
  "dissolved_oxygen": 78.5,
  "microplastics": 2.4,
  "mercury": 0.00435,
  "lead": 0.0127,
  "cadmium": 0.00184,
  "chromium": 0.00832,
  "petroleum_hydrocarbons": 0.27,
  "polycyclic_aromatic": 0.0132,
  "nitrates": 3.8,
  "phosphates": 0.42,
  "ammonia": 0.23,
  "cp_pesticides_total": 0.00345,
  "cp_pcbs_total": 0.00042,
  "cp_dioxins": 0.0000436,
  "coliform_bacteria": 87,
  "chlorophyll_a": 7.8,
  "water_quality_index": 72.5
}
```

2. **Satellite Imagery (Sentinel-2)**

Process

- Fetched via Sentinel Hub API
- Synthetic pollution overlays applied
- Stored in MinIO with metadata



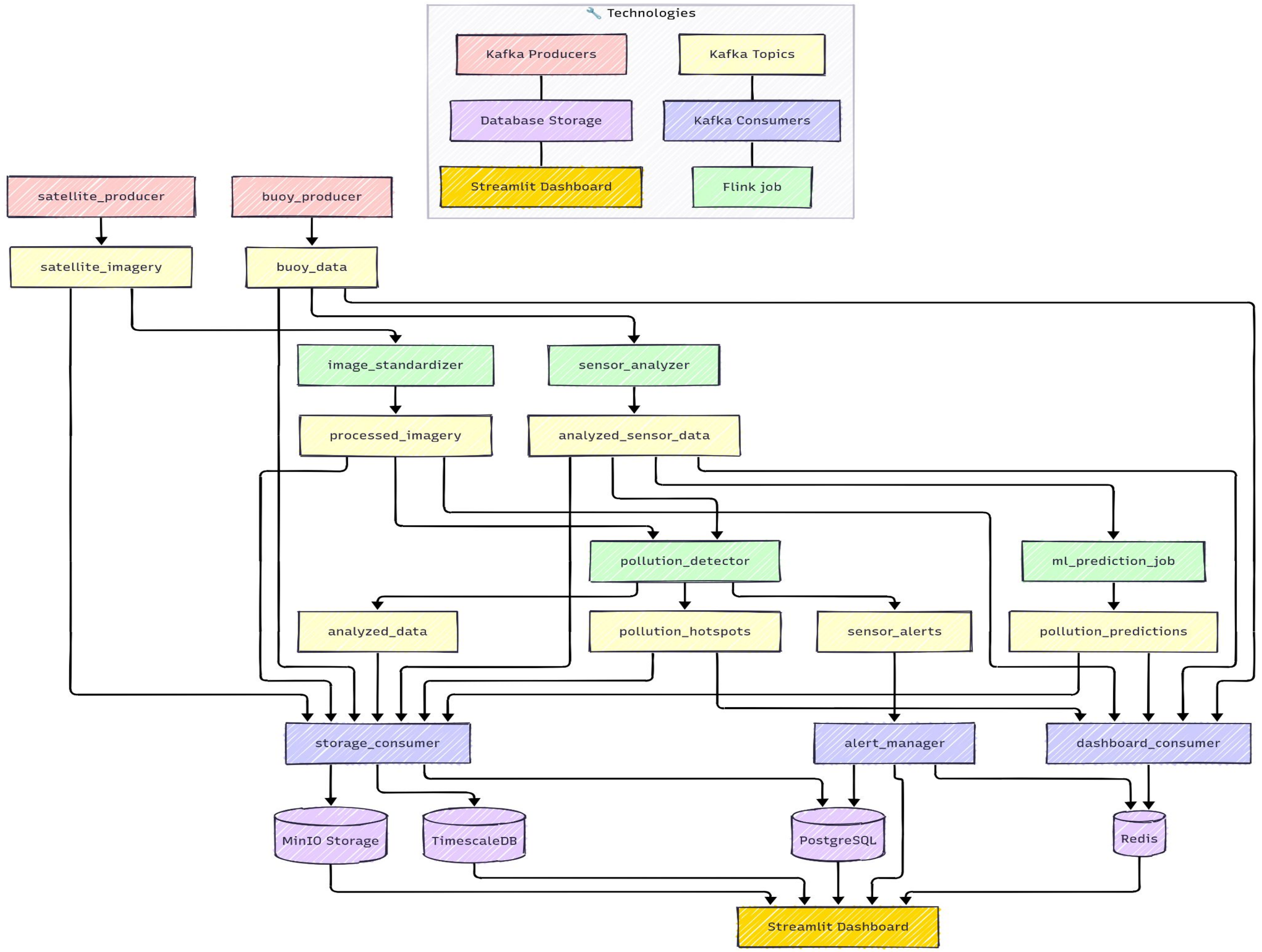
json

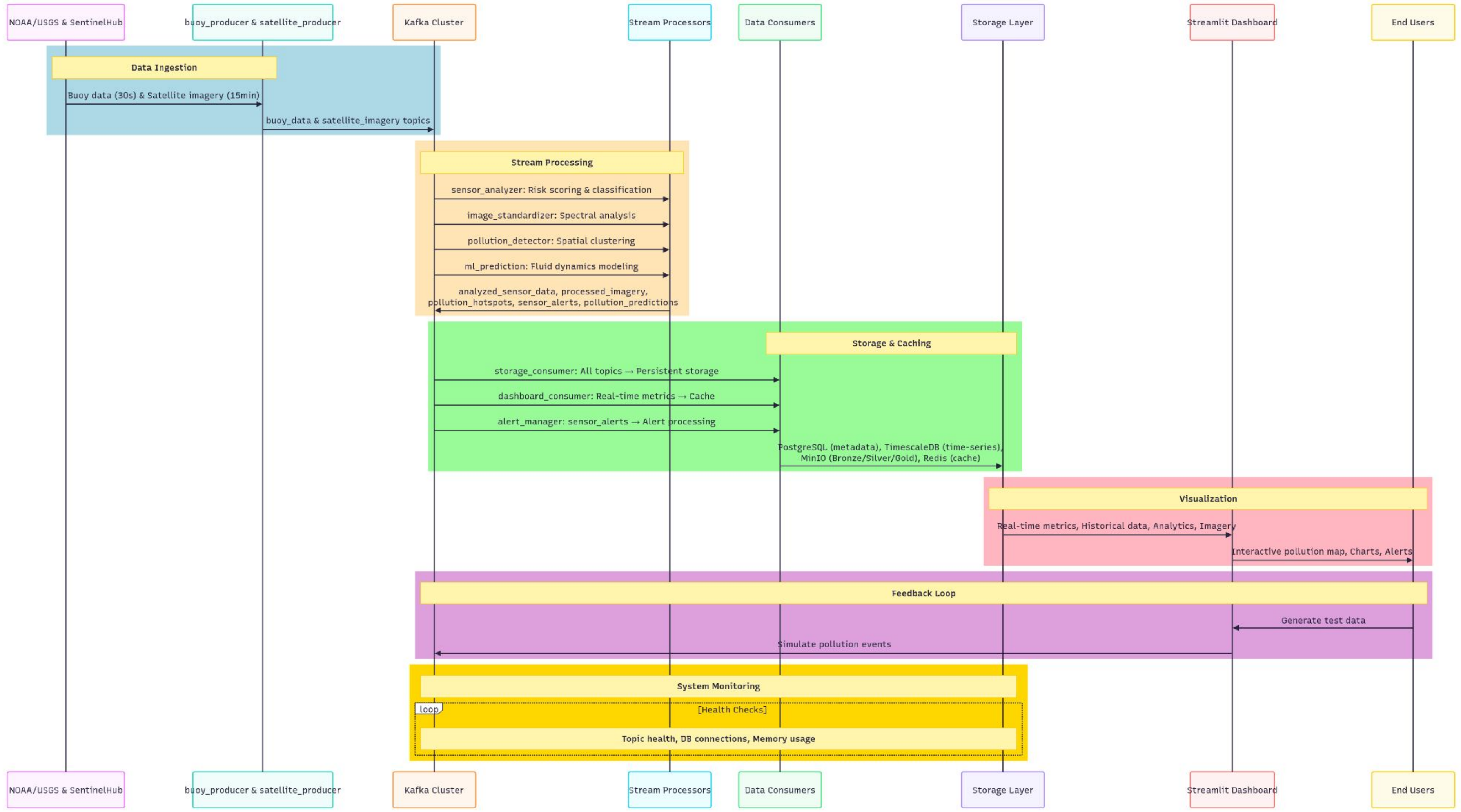
```json
{
  "image_pointer": "satellite_imagery/sentinel2/year=2024/month=07/day=17/sat_img_BUOY
  "metadata": {
    "timestamp": "2024-07-17T12:34:56.789",
    "microarea_id": "44042",
    "macroarea_id": "BUOY",
    "satellite_data": [
      {
        "latitude": 38.033456,
        "longitude": -76.334578,
        "microarea_id": "44042",
        "bands": {
          "B2": 0.085,
          "B3": 0.165,
          "B4": 0.178,
          "B8": 0.082,
          "B8A": 0.042,
          "B11": 0.156,
          "B12": 0.133
        }
      },
      {
        "latitude": 38.032987,
        "longitude": -76.336012,
        "microarea_id": "44042",
        "bands": {
          "B2": 0.092,
          "B3": 0.198,
          "B4": 0.183,
          "B8": 0.062,
          "B8A": 0.035,
          "B11": 0.127,
          "B12": 0.148
        }
      }
      // Additional pixels with simulated band values...
    ]
  }
}
```

# Data exploration insights (pt.2)

| **Data Source** | **Format** | **Receive Frequency** | **Size per Record** | **Hourly Volume** |
|---|---|---|---|---|
| Buoy raw data | JSON | Every 30 seconds | 1-2 KB | 120-240 KB per sensor |
| Satellite raw images | JSON + jpg | Every 15 minutes | 1-2 KB + 5-10 MB | 20-40 MB per area |
| Buoy analyzed data | parquet | Every 30 seconds | 20-40 KB | 240-480 kB per sensor |
| Processed imagery daya | parquet+ tiff | Every 15 minutes | 20-40 KB + 1-3 MB | 4-12 MB per area |
| Pollution hotspots | parquet | On detection (varies) | 15-20 kb | Variable (~1-2 MB) per sensor |
| Pollution predictions | parquet | On detection (varies) | 15-20 KB | Variable (~1-2 MB) per sensor |

## System Architecture




NOAA/USGS & SentinelHub
buoy_producer & satellite_producer
Kafka Cluster
Stream Processors
Data Consumers
Storage Layer
Streamlit Dashboard
End Users
Data Ingestion
Buoy data (30s) & Satellite imagery (15min)
buoy_data & satellite_imagery topics
Stream Processing
sensor_analyzer: Risk scoring & classification
image_standardizer: Spectral analysis
pollution_detector: Spatial clustering
ml_prediction: Fluid dynamics modeling
analyzed_sensor_data, processed_imagery, pollution_hotspots, sensor_alerts, pollution_predictions
Storage & Caching
storage_consumer: All topics → Persistent storage
dashboard_consumer: Real-time metrics → Cache
alert_manager: sensor_alerts → Alert processing
PostgreSQL (metadata), TimescaleDB (time-series), MinIO (Bronze/Silver/Gold), Redis (cache)
Visualization
Real-time metrics, Historical data, Analytics, Imagery
Interactive pollution map, Charts, Alerts
Feedback Loop
Generate test data
Simulate pollution events
System Monitoring
loop
[Health Checks]
Topic health, DB connections, Memory usage
NOAA/USGS & SentinelHub
buoy_producer & satellite_producer
Kafka Cluster
Stream Processors
Data Consumers
Storage Layer
Streamlit Dashboard
End Users

# Technologies & Justifications

| *Component* | *Technology* | *Justification* |
|---|---|---|
| Core Language | Python | Rich data science ecosystem (NumPy, pandas) with native Flink/Kafka integration via PyFlink. Simplified satellite image processing with rasterio and effective ML model integration. |
| Containerization platform | Docker | Ensures consistent deployment across environments for all microservices. Docker Compose orchestrates the system with appropriate resource allocation and network isolation. |
| In-Memory Cache | Redis | Microsecond access to frequent queries, optimized for geospatial operations with sorted sets. Supports real-time dashboard updates and alert distribution with pub/sub capabilities. |
| Message Broker | Kafka | Fault-tolerant data pipeline with topic partitioning for high-throughput sensor data. Persistence ensures reliable delivery of critical environmental measurements and supports event replay for analysis. |
| Stream Processing | Apache Flink | True event-time processing with exactly-once semantics critical for temporal pollution analysis. Stateful computations enable tracking pollution evolution |

# Technologies & Justifications (pt.2)

| *Component* | *Technology* | *Justification* |
|---|---|---|
| Main Database | PostgreSQL | ACID-compliant storage for pollution events and metadata. PostGIS extension enables critical geospatial queries for hotspot identification and intervention planning. |
| Time-Series | TimescaleDB | PostgreSQL extension optimized for sensor time-series data, with hypertables providing efficient querying of historical measurements. Supports continuous aggregations for trend analysis. |
| Object Storage | MinIO | Implements bronze/silver/gold medallion architecture for data quality management. S3-compatible API with versioning supports large satellite imagery storage and processing pipeline integration. |
| Dashboard | Streamlit | Rapid development of interactive pollution maps and monitoring dashboards. Integrates with geospatial libraries to provide actionable environmental intelligence to stakeholders. |
| Error Handling | DLQ Pattern | Implements Dead Letter Queues for each Kafka topic to minimize data loss during processing failures. This approach aims to improve fault tolerance, though extensive testing under real conditions would be needed to quantify actual reliability metrics. |

# Implementation & Code Repository

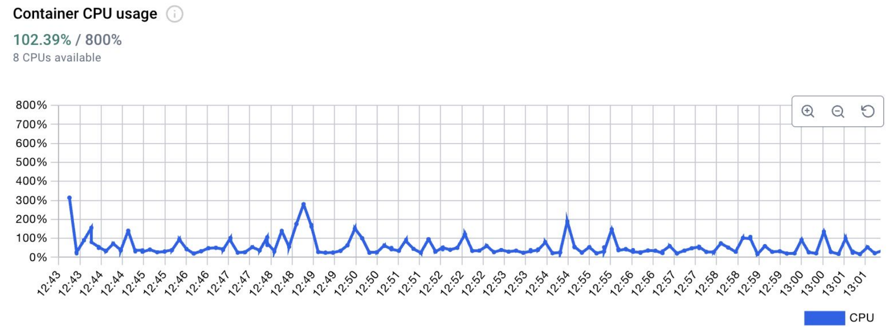
- **Docker Images:** python:3.9, postgres:14-alpine, timescale/timescaledb:latest-pg14, redis:7-alpine, confluentinc/cp-zookeeper:7.5.0, confluentinc/cp-kafka:7.5.0, minio/minio:latest, flink:1.17-scala_2.12.
- **Marine Pollution Tracking System** is a real-time environmental monitoring platform that ingests and processes multi-source data — including satellite imagery, NOAA/USGS sensor data, and pollution simulations — to detect hotspots and predict pollution spread. It supports rapid, data-driven environmental decisions via an interactive dashboard.
- **Key Modules & Pipelines:**
  - Data Ingestion: buoy_producer/, satellite_producer/ → Kafka topics.
  - Real-Time Processing (Flink Jobs): image_standardizer/, sensor_analyzer/, pollution_detector/, ml_prediction/.
  - Storage: PostgreSQL for metadata, TimescaleDB for time-series, MinIO as multi-layer Data Lakehouse (bronze, silver, gold).
  - Visualization: dashbard/ displays pollution maps, sensor alerts, hotspot detection, and spread predictions.
  - Alert Management: alert_manager/ generates severity-based notifications with recommendations.
- **Getting Started:**
  - Clone repo: git clone https://github.com/andrea00mauro00/marine-pollution-tracking.git
  - Add SentinelHub credentials in satellite_producer/credentials.json
  - Start system: docker-compose up -d
  - Dashboard: http://localhost:8501, MinIO UI: http://localhost:9001 (user/pass: minioadmin)

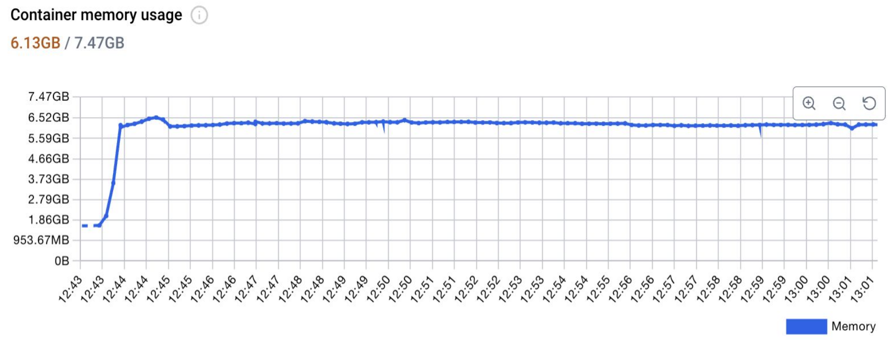
# Results and Performance

**System Resource Utilization Analysis**
CPU and memory usage visualization shows the operational efficiency of our marine pollution tracking system:

- **System Stability Confirmation:** The graphs show consistent and predictable processing patterns with no crashes or failures. CPU utilization shows regular processing cycles while memory remains remarkably stable.
- **Resource Efficiency Metrics**: The system operates at an average of 102% CPU utilization (approximately 1 core) out of 8 available cores (800%), with periodic processing spikes reaching up to 300% during intensive analysis tasks. Memory utilization is efficiently maintained at a steady 6.13GB out of 7.47GB allocated.
- **Optimal Resource Allocation:** Memory usage pattern reveals an efficient initialization phase followed by consistent utilization, indicating proper memory management with no evidence of leaks or excessive garbage collection cycles.
- **Processing Pattern Analysis:** CPU utilization exhibits characteristic processing behavior with defined periodic spikes that correspond to scheduled satellite image analysis and pollution detection algorithm execution.

This resource utilization profile validates the system's operational efficiency, confirming that our marine pollution tracking platform can handle its monitoring workload while maintaining significant headroom for scaling during increased environmental monitoring demands.





# Results and Performance (pt.2)

## <u>System Metrics</u>

### Distributed Processing Efficiency

The system metrics reveal a strategically architected distributed platform where resource allocation directly correlates with operational priorities. The container distribution demonstrates a processing-heavy configuration (40% processors) engineered to handle computationally intensive marine pollution detection algorithms, while maintaining robust storage infrastructure (30%) and comprehensive monitoring systems (20%) that ensure continuous operational visibility.

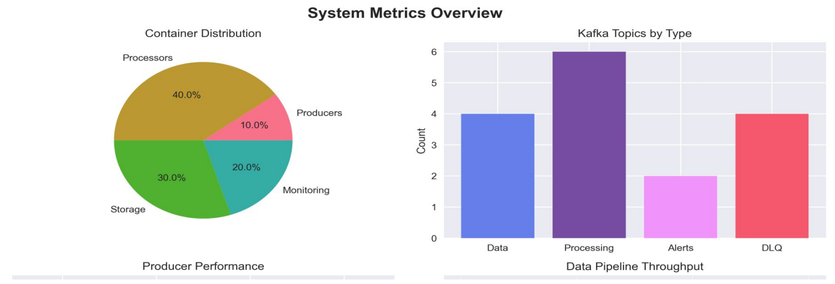
### Key Performance Indicators:

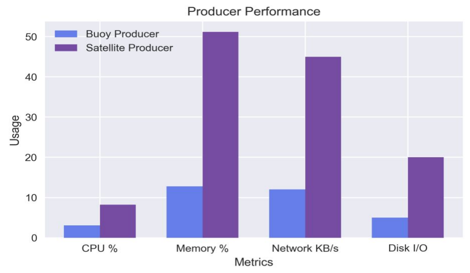
- Producer Performance: Satellite Producer exhibits significantly higher resource utilization (50% memory, 45% network) compared to Buoy Producer, accurately reflecting the computational demands of real-time satellite image processing and analysis workflows
- Data Pipeline Throughput: The systematic decline from ingestion (120 msgs/min) to analysis (80 msgs/min) demonstrates appropriate strategies for data handling and aggregation that prevent system overload while maintaining data integrity

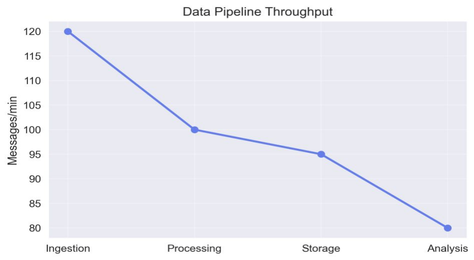
### Architectural Strengths:

- Memory utilization patterns show satellite processing as the primary resource consumer, validating the system's focus on complex environmental data analysis
- Error handling infrastructure with dedicated DLQ topics provides robust fault tolerance for mission-critical environmental monitoring
- Scalable design architecture where processing components appropriately consume the majority of system resources, enabling horizontal scaling as monitoring demands increase

This configuration validates the system's capability to handle real-time marine pollution monitoring with efficient resource management, robust error handling, and the processing power necessary for detecting environmental threats that require immediate response.







# Results and Performance (pt.3)

**Kafka Message Streaming Performance**

**Strategic Importance and Real-Time Data Pipeline Analysis**

This Kafka Topics Analysis is essential for understanding the operational health and scalability of the marine pollution tracking system, providing critical insights into our distributed architecture's performance and reliability.

**Why This Analysis Matters:**

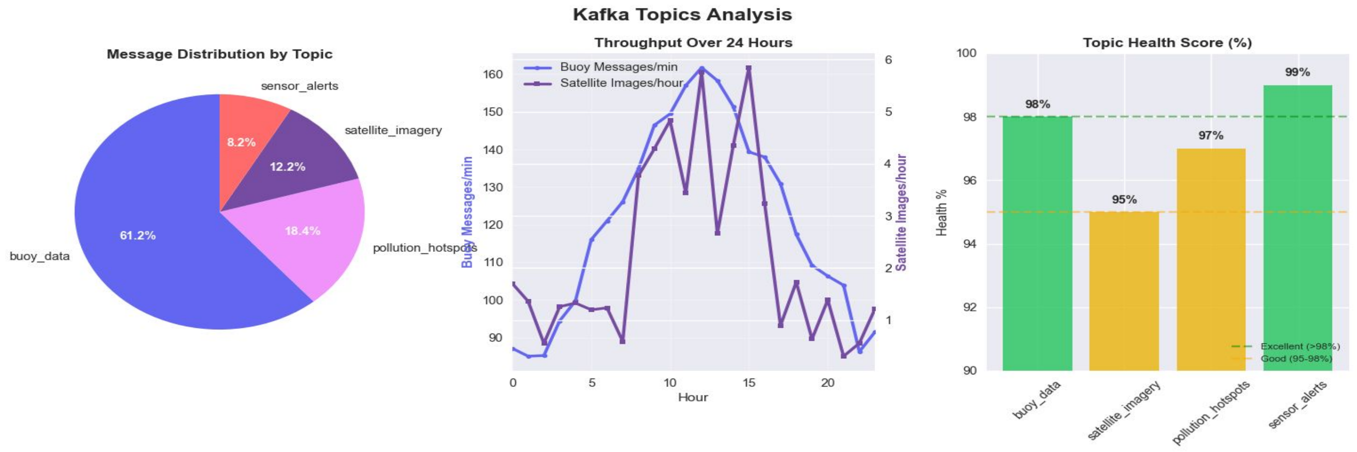
The chart validates that our system is functioning as designed while providing crucial metrics for resource allocation, data quality assurance, and scalability planning. For environmental agencies requiring consistent monitoring and emergency response systems, this analysis demonstrates the system's readiness for production deployment with >95% reliability standards.

**Message Flow Characteristics:**

- Primary Data Source: Buoy data constitutes the majority of message traffic (61.2%), confirming that real-time IoT sensor monitoring is appropriately the system's primary data source
- Pollution Detection Pipeline: Pollution hotspots (18.4%) and satellite imagery (12.2%) show balanced processing loads for comprehensive environmental analysis
- Alert Generation: Sensor alerts (8.2%) demonstrate responsive anomaly detection capabilities crucial for emergency response

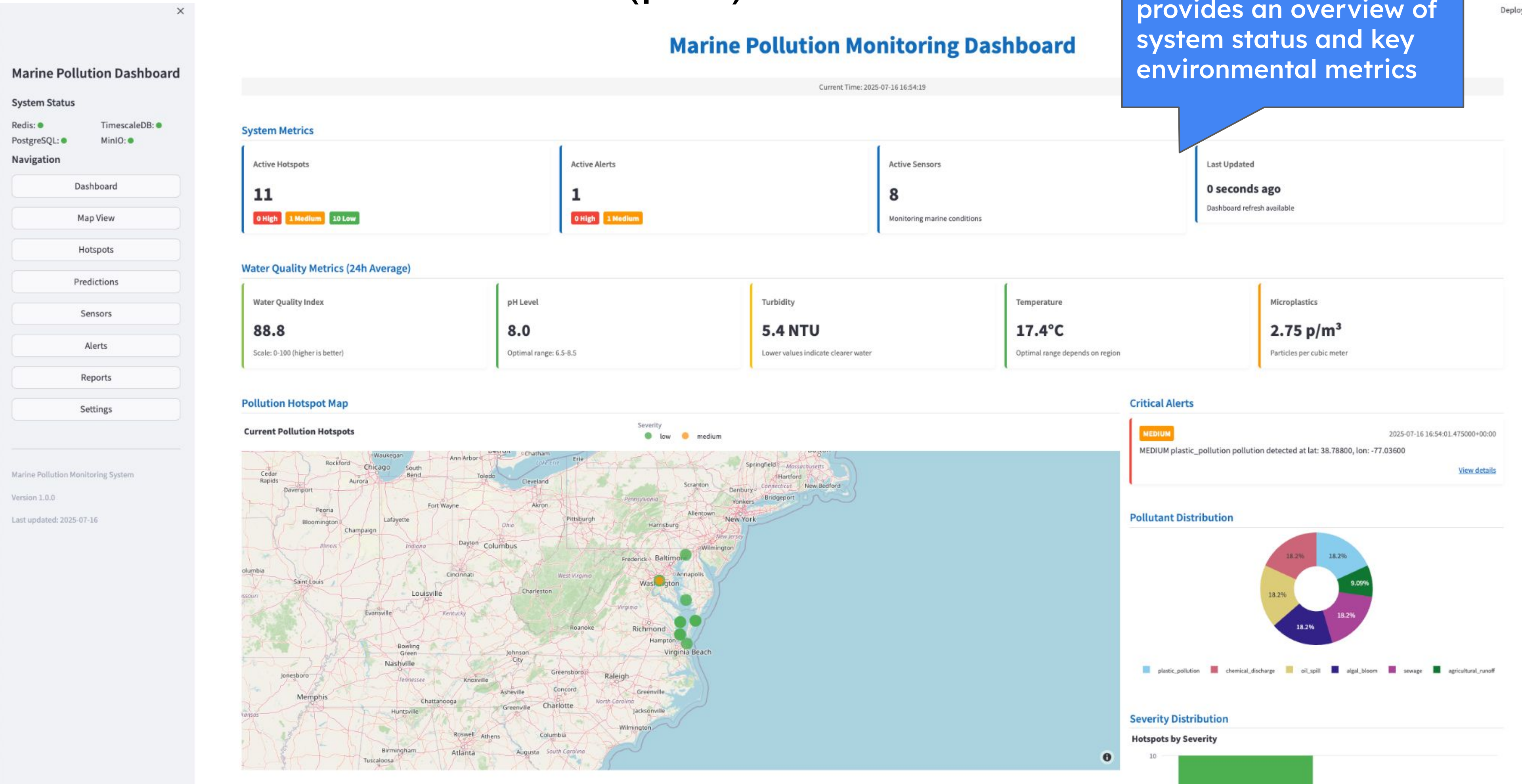
**Performance Metrics:**

- Throughput Performance: Consistent 24-hour processing with peak throughput reaching 158 messages/minute during intensive monitoring periods, revealing peak processing times for infrastructure scaling
- Topic Health Assessment: Exceptional health scores across all topics (>95%), with satellite_imagery showing 95% health due to complex image processing workflows
- Processing Stability: 99% health score for sensor_alerts confirms the system's capability to handle critical environmental notifications with minimal data loss



# Results and Performance (pt.4)

The main dashboard provides an overview of system status and key environmental metrics



Marine Pollution Dashboard

**System Status**

Redis: ● TimescaleDB: ●
PostgreSQL: ● MinIO: ●

**Navigation**

Dashboard
Map View
Hotspots
Predictions
Sensors
Alerts
Reports
Settings

Marine Pollution Monitoring System
Version 1.0.0
Last updated: 2025-07-16

Deplo

## Marine Pollution Monitoring Dashboard

Current Time: 2025-07-16 16:54:19

### System Metrics

| Active Hotspots | Active Alerts | Active Sensors | Last Updated |
|---|---|---|---|
| 11 | 1 | 8 | 0 seconds ago |
| 0 High · 1 Medium · 10 Low | 0 High · 1 Medium | Monitoring marine conditions | Dashboard refresh available |

### Water Quality Metrics (24h Average)

| Water Quality Index | pH Level | Turbidity | Temperature | Microplastics |
|---|---|---|---|---|
| 88.8 | 8.0 | 5.4 NTU | 17.4°C | 2.75 p/m³ |
| Scale: 0-100 (higher is better) | Optimal range: 6.5-8.5 | Lower values indicate clearer water | Optimal range depends on region | Particles per cubic meter |

### Pollution Hotspot Map

**Current Pollution Hotspots**

Severity
● low ● medium

### Critical Alerts

MEDIUM 2025-07-16 16:54:01.475000+00:00

MEDIUM plastic_pollution pollution detected at lat: 38.78800, lon: -77.03600

View details

### Pollutant Distribution

plastic_pollution · chemical_discharge · oil_spill · algal_bloom · sewage · agricultural_runoff

### Severity Distribution

**Hotspots by Severity**

## Alert Information

**ID:** alert_hotspot-c492af635f427491

**Type:** plastic_pollution

**Severity:** medium

**Status:** active

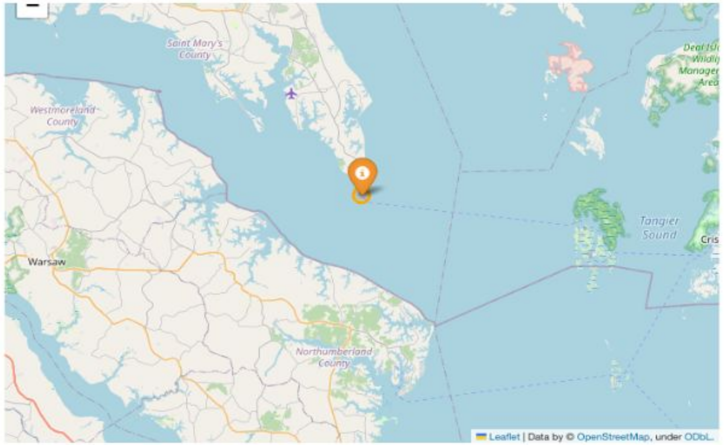
**Time:** 2025-07-16 16:55:00

**Message:** MEDIUM plastic_pollution pollution detected at lat: 38.03300, lon: -76.33500

**Location:** Lat 38.03300, Lon -76.33500

**Risk Score:** 0.70

Alert details provide comprehensive information and response guidance



## Intervention Recommendations

### Immediate Actions

- Conduct comprehensive water quality testing
- Deploy monitoring buoys around affected area
- Collect water and sediment samples
- Document visual observations with photos/video

### Resource Requirements

**Vessels:** 1-2 monitoring vessels

**Supplies:** Sample containers, documentation equipment

**Equipment:** Multi-parameter testing kits, sampling equipment

**Personnel:** 5-10 environmental response specialists

### Cleanup Methods

- monitoring
- containment
- assessment
- targeted_intervention

### Stakeholders to Notify

- Local Environmental Agency
- Water Management Authority
- Local Government

### Regulatory Implications

- Documentation of incident and response actions
- Potential monitoring requirements
- Notification to local authorities

# Lessons Learned

**Main Challenges**

During the development of our Marine Pollution Monitoring System, we encountered several significant challenges that provided valuable learning opportunities. Our primary challenge was designing and implementing the distributed pipeline architecture. Integrating multiple components through Kafka required extensive configuration and troubleshooting. Establishing correct topic structures, managing serialization formats, and ensuring proper message routing between components demanded considerable effort.

The most time-consuming aspect was implementing proper data flow through our pipeline and developing effective deduplication strategies. Distinguishing between new pollution events and evolutions of existing ones proved exceptionally difficult. Creating deterministic hotspot IDs that accounted for spatial proximity and temporal evolution required multiple iterations to refine our clustering algorithms and spatial indexing strategies. Designing and configuring specialized databases for different types of data presented unique challenges. We needed to carefully configure TimescaleDB for time-series sensor data, PostgreSQL with spatial extensions for geospatial analysis, and Redis for fast dashboard access. Working with heterogeneous data sources presented significant complexity, from structured sensor readings to large satellite imagery—each requiring different ingestion, validation, and preprocessing approaches.

**What Worked Well vs. What Didn't**

The medallion architecture proved highly effective for data quality management, enabling clean separation between raw data and processed insights while providing clear data lineage tracking. Apache Flink's stream processing capabilities provided robust real-time analysis with minimal latency. The exactly-once processing semantics and flexible windowing operations were particularly valuable for environmental monitoring. Our containerized microservices approach allowed for independent scaling and isolated development of each component, significantly accelerating the development cycle despite the system's complexity. Conversely, several aspects didn't work as well as anticipated. Initial attempts at ML model integration lacked proper versioning and reproducibility, causing inconsistent pollution detection results across system restarts. Our first approach to spatial clustering was computationally expensive and didn't scale with increasing data volume, requiring a complete redesign with grid-based preprocessing. Initial data schemas were too rigid, requiring frequent updates as requirements evolved, and new pollution parameters were identified. The event correlation system struggled with time synchronization between satellite imagery and buoy data, creating challenges for validating pollution events across different data sources.

# Lessons Learned (pt.2)

**Key Insights Gained During Development**

Limited time and resources forced us to make strategic compromises, prioritizing core functionality over complete implementation of supporting systems. We omitted Airflow for ML model orchestration, implemented basic observability with structured logging rather than a full monitoring stack, and used synthetic data generation for buoy readings rather than complex integration with real APIs. These compromises allowed us to deliver a functional prototype while understanding which areas would require expansion in a production environment.

Our approach focused on creating a functionally complete prototype rather than a fully-scaled big data system, choosing technologies that support big data principles but implementing them in simplified configurations. Rather than investing time in complex clustering and replication setups, we created a foundation that demonstrates the system's capabilities while establishing a good baseline for scaling. The most valuable insight was the importance of data standardization early in the pipeline. Standardizing formats immediately after ingestion significantly reduced downstream complexity. We discovered that real-time environmental monitoring systems benefit greatly from a hybrid approach combining rule-based detection with machine learning models. The rule-based systems provided reliable baseline detection, while ML models enhanced sensitivity for subtle pollution patterns. This complementary approach appears promising and could potentially offer advantages over either method alone, though more systematic comparison would be valuable to quantify these benefits. Finally, we recognized that for environmental monitoring systems, data quality is more critical than quantity, fundamentally shifting our focus from maximizing data collection to optimizing data reliability.

# Limitations & Future work

**Limitations of the Current System**

Our Marine Pollution Monitoring System relies entirely on synthetic data rather than real NOAA/USGS APIs and Sentinel Hub services, limiting validation under authentic environmental conditions. A critical limitation is our ML models - basic RandomForest classifiers built solely on synthetic data without any mechanism for retraining with newly collected information. This means the system's predictive capabilities remain static, preventing improvement over time despite new data collection. Despite multiple implementation attempts, the system currently struggles with correctly identifying and deduplicating pollution hotspots, which can lead to duplicate entries and confusion in the monitoring interface. Our geographic event localization employs a hybrid approach combining initial spatial partitioning via grid-based keys followed by clustering within these partitions. While more sophisticated than simple binning, this method struggles with events crossing grid boundaries, and the fixed grid size represents a compromise that isn't optimal across varying data densities. The water current data used in pollution spread predictions is static and not properly integrated with the geographic features, resulting in unrealistic predictions where pollution appears to spread onto land areas. The data lake implementation using MinIO provides a functional foundation but lacks comprehensive governance features such as quality monitoring, lineage tracking, and automated schema evolution - limitations that would become problematic as data volumes increase. Additionally, our dashboard offers limited analytical capabilities compared to enterprise platforms, and the system lacks integration with external alerting infrastructure necessary for operational deployment in environmental agencies.

**What Would Break First When the System Scales**

As the system scales, memory management in our Flink jobs would likely fail first, particularly during satellite imagery analysis and pollution detection where processing is memory-intensive. Without proper partitioning and memory management strategies, jobs would encounter OutOfMemoryError exceptions when handling larger or more frequent satellite images. Our hybrid spatial clustering approach would face computational bottlenecks in high-density regions, where the SpatialClusteringProcessor must process numerous points within each partition. The current fixed grid size would become increasingly problematic at scale - creating unnecessary computational overhead in sparse areas while exceeding optimal processing thresholds in dense regions. The Kafka implementation with a single-node configuration would experience significant performance degradation with increased message volume. Without proper partitioning, consumer group configuration, and clustering, message processing lag would increase, potentially compromising the real-time nature of the system. TimescaleDB would also face scaling challenges as time-series data accumulates, with query performance degrading without proper hypertable partitioning strategies and retention policies, impacting both historical analysis capabilities and dashboard responsiveness.

# Limitations & Future work (pt.2)

**Potential Improvements**

Several improvements could address current limitations without a complete redesign. Implementing a continuous learning mechanism would allow ML models to update with newly collected data, dramatically improving prediction accuracy over time through incremental learning that periodically retrains models as validated data becomes available. We could enhance the spatial clustering system by implementing adaptive grid sizing that automatically adjusts partitioning based on data density, complemented by more efficient clustering algorithms and boundary-crossing detection mechanisms to better identify pollution events spanning multiple grid cells. Developing improved state management strategies for Flink jobs would enhance stability under load, including proper backpressure handling, optimized checkpointing, and potentially using appropriate mechanisms for efficient state management with large datasets. Our storage approach could be improved with proper data lifecycle management featuring automated archiving for historical data while maintaining immediate access to recent measurements. The dashboard functionality could be enhanced with more advanced analytical capabilities, while strengthening the error handling system with comprehensive monitoring would significantly improve reliability while reducing operational overhead.

**Future Work with Additional Resources**

With additional resources, we would prioritize integration with real data sources through NOAA, USGS, and Copernicus APIs, providing authentic environmental readings and satellite imagery that would significantly improve the system's practical utility. We would develop a comprehensive ML pipeline with continuous training, performance monitoring, and drift detection to transform our static models into an evolving analytical system with proper versioning and feature engineering. Building upon our current hybrid approach, we would implement an advanced spatial clustering framework with multi-level gridding, adaptive partitioning, and specialized algorithms for detecting different pollution pattern morphologies (point-source, diffuse, linear). Creating cross-validation mechanisms between different data sources would increase confidence in event detection while reducing false positives. On the infrastructure side, we would implement a proper Kafka cluster with multiple brokers and appropriate partitioning strategies, optimize TimescaleDB with proper hypertable configurations, and deploy comprehensive monitoring using ELK or Prometheus/Grafana. Finally, developing a Kubernetes-based deployment strategy would enable dynamic scaling based on data volume and processing requirements, transforming our prototype into a production-ready environmental monitoring system capable of supporting real environmental protection efforts.

# References & Acknowledgments

[1] Sannigrahi, S., Basu, B., Basu, A. S., & Pilla, F. (2021). Detection of marine floating plastic using Sentinel-2 imagery and machine learning models. *arXiv preprint arXiv:2106.03694*.

[2] Walsh, E. S., Kreakie, B. J., Cantwell, M. G., & Nacci, D. (2017). A Random Forest approach to predict the spatial distribution of sediment pollution in an estuarine system. *PLoS One*, *12*(7), e0179473.

[3] Li, Z., Zhu, Y., & Van Leeuwen, M. (2023). A survey on explainable anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, *18*(1), 1-54.

[4]Kim D, Antariksa G, Handayani MP, Lee S, Lee J. Explainable Anomaly Detection Framework for Maritime Main Engine Sensor Data. Sensors (Basel). 2021 Jul 31;21(15):5200. doi: 10.3390/s21155200. PMID: 34372436; PMCID: PMC8347810.

[5] Sadaiappan, B., Balakrishnan, P., CR, V., Vijayan, N. T., Subramanian, M., & Gauns, M. U. (2023). Applications of machine learning in chemical and biological oceanography. *ACS omega*, *8*(18), 15831-15853.