AIOps Dashboard – Global Telecom Operator (GenAI/HPC/GKE Integration)

**Document Version**: v1.1  
**Date**: 12/22/2024  
**Author / Owner**: Jayesh Chavan

## **Table of Contents**

1. Executive Summary
2. Introduction
3. Problem Statement and Use Cases
4. Solution Overview  
   4.1 AIOps Architecture  
   4.2 Data Ingestion and Sources  
   4.3 Machine Learning Pipeline  
   4.4 Dashboard UI and Interaction
5. Technical Details and Implementation  
   5.1 Code Structure  
   5.2 Setup and Installation  
   5.3 Running the Application  
   5.4 Key Metrics and Features
6. Data Science and MLOps Considerations  
   6.1 Model Monitoring  
   6.2 Model Retraining and Maintenance  
   6.3 Scalability and Future Enhancements
7. Security and Compliance
8. Limitations and Assumptions
9. Roadmap
10. Conclusion

<a name="executive-summary"></a>

## **1. Executive Summary**

This document details a **Streamlit-based AIOps dashboard** for a **global telecom operator** running **GenAI, HPC, Kubernetes (GKE), and Serverless** workloads. The solution showcases how advanced ML techniques (Isolation Forest, KMeans, Linear Regression) can detect anomalies, cluster resource usage, and forecast CPU trends in real time.

The dashboard leverages **telescopic filters** (organization, region, service, environment, network type) to drill down on different slices of data. It also demonstrates **key AIOps metrics**: MTTI, MTTR, machine-identified alerts, and monthly cloud/network spend.

This document follows **technical data science product standards**, covering architecture, data sources, ML pipeline, security, limitations, and a roadmap for future improvements.

<a name="introduction"></a>

## **2. Introduction**

Large telecoms and enterprises face operational complexity when running **modern AI/ML workloads**. New challenges arise from:

* **GenAI LLM deployments** (GPU usage, concurrency spikes, Python environment mismatches).
* **HPC clusters** (e.g., cluster scheduling, GPU out-of-memory errors).
* **Kubernetes / GKE** (OOMKilled pods, autoscaling, multi-tenant environments).
* **Serverless** (Cloud Run, Function Apps concurrency scaling).

**AIOps** addresses these challenges by combining **observability, analytics, and automated remediation**. This document describes how the included **Streamlit app** can unify metrics, incidents, cost, security posture, and ML-based insights (anomalies, clustering, predictions) in a single “single pane of glass.”

<a name="problem-statement-and-use-cases"></a>

## **3. Problem Statement and Use Cases**

1. **Visibility**
   * Operators struggle to see real-time health across HPC clusters, GKE pods, and GenAI pipelines.
2. **Proactive Anomaly Detection**
   * Traditional monitoring triggers only on static thresholds; outliers in GPU usage or concurrency can go undetected.
3. **Predictive Analytics**
   * Teams need to forecast resource usage to avoid last-minute resource starvation (GPU) or cost overruns.
4. **Telescopic Scoping**
   * Large organizations must quickly filter by business unit, region, or environment to isolate issues.

**Primary Use Cases**:

* **SRE / Ops**: Monitor HPC GPU usage, find anomalies, reduce MTTR.
* **NOC Teams**: Identify critical incidents (P1) across multiple services.
* **Cloud Architects**: Assess cost anomalies and efficiency (CPU, memory, GPU usage).
* **Data Science Leads**: Evaluate HPC/GPU cluster performance, plan for scaling LLM training jobs.

<a name="solution-overview"></a>

## **4. Solution Overview**

<a name="aiops-architecture"></a>

### **4.1 AIOps Architecture**

1. **Telescopic Filter Layer**
   * Hierarchical approach: Division → Business Unit → Region → Service + Network Type → Environment → Time Range.
2. **Data Layer**
   * Mock data simulates CPU, Memory, GPU usage, Disk I/O, Network in/out, cost, security events, and incidents.
   * In a real setting, these would come from Prometheus, logs, HPC cluster metrics, GKE usage, or serverless function logs.
3. **ML Layer**
   * **Isolation Forest** for anomaly detection.
   * **KMeans** for usage clustering (low/medium/high patterns).
   * **Linear Regression** for short-term CPU usage forecasts.
4. **UI / Visualization**
   * Built on **Streamlit** with a **dark theme** and top-level KPIs (MTTI, MTTR, etc.).

<a name="data-ingestion-and-sources"></a>

### **4.2 Data Ingestion and Sources**

* **Synthetic Generators** in the current code produce random HPC/GenAI metrics.
* Real-world ingestion might include:
  + **HPC monitoring** (Slurm metrics, GPU usage logs).
  + **Kubernetes** or **GKE** metrics (pod statuses, memory usage, concurrency).
  + **Cost data** from cloud billing APIs.
  + **Security events** from SIEM or container threat detection.

<a name="machine-learning-pipeline"></a>

### **4.3 Machine Learning Pipeline**

1. **Data Preparation**
   * The numeric columns for resource usage are extracted.
2. **Anomaly Detection (Isolation Forest)**
   * Flags outliers in CPU%, Memory%, GPU%, Disk I/O, Network.
   * contamination=0.1 (10%) as a default.
3. **Clustering (KMeans)**
   * Groups usage patterns into 3 clusters (low, medium, high).
   * Helps identify oversubscription vs. underutilization.
4. **Prediction (Linear Regression)**
   * A naive approach to forecast CPU usage for the next few minutes/hours.
   * Could be replaced by advanced methods (Prophet, LSTM, etc.) for production.

<a name="dashboard-ui-and-interaction"></a>

### **4.4 Dashboard UI and Interaction**

1. **Dark Themed Streamlit App**
   * Custom CSS sets background colors, font colors, and component styles.
2. **Sidebar Filters**
   * Users can instantly scope the data to the relevant domain.
3. **Key KPI Row**
   * MTTI, MTTR, Machine Identified ratio, and monthly cost.
4. **Charts**
   * CPU/Memory/GPU usage, Network throughput, Disk I/O trends.
5. **Incident Table**
   * Shows open/closed incidents with severity, including HPC or GKE-specific errors.

<a name="technical-details-and-implementation"></a>

## **5. Technical Details and Implementation**

<a name="code-structure"></a>

### **5.1 Code Structure**

bash

Copy code

aiops\_telecom\_ml\_dashboard.py # Main Streamlit script

requirements.txt # Python dependencies

README.md # Project overview

* **aiops\_telecom\_ml\_dashboard.py**: Integrates telescopic filters, ML pipelines, cost/security data, and dark theme CSS.

<a name="setup-and-installation"></a>

### **5.2 Setup and Installation**

**Clone Repo**bash  
Copy code  
git clone https://github.com/your-org/telecom-aiops-dashboard.git

cd telecom-aiops-dashboard

**Install Dependencies**bash  
Copy code  
pip install -r requirements.txt

**Run the Dashboard**bash  
Copy code  
streamlit run aiops\_telecom\_ml\_dashboard.py

1. **Access in Browser**
   * Defaults to http://localhost:8501.

<a name="running-the-application"></a>

### **5.3 Running the Application**

1. **Sidebar**: Choose Division, BU, Region, Service, Network Type, Env, Time Range.
2. **Toggle ML Features**: Anomalies, Clusters, Prediction.
3. **Observe Key Metrics**: MTTI, MTTR, Alerts ratio, Spend.
4. **Scroll for**: Incidents table, HPC/GPU usage, network charts, cost anomalies.

<a name="key-metrics-and-features"></a>

### **5.4 Key Metrics and Features**

* **MTTI & MTTR**
  + Randomly generated deltas, show overall reactivity and repair times.
* **Machine-Identified Alerts**
  + Reflect how many incidents are flagged by the ML pipeline vs. human.
* **Monthly Cloud/Network Spend**
  + Compares to a budget, highlights anomalies when usage spikes.
* **GenAI/HPC/GKE Incidents**
  + OOMKilled pods, GPU memory exhaustion, concurrency errors, HPC node failures, etc.

<a name="data-science-and-mlops-considerations"></a>

## **6. Data Science and MLOps Considerations**

<a name="model-monitoring"></a>

### **6.1 Model Monitoring**

In production, the ML components (Isolation Forest, KMeans, Linear Regression) would require:

* **Performance dashboards** (precision/recall for anomaly detection, cluster stability).
* **Drift detection** (tracking changes in HPC or GPU usage distributions).

<a name="model-retraining-and-maintenance"></a>

### **6.2 Model Retraining and Maintenance**

* **Retraining frequency**: Weekly or monthly for HPC usage patterns.
* **Continuous Deployment**: New model versions deployed seamlessly.
* **Data versioning**: Using tools like DVC or MLflow to keep track of data changes.

<a name="scalability-and-future-enhancements"></a>

### **6.3 Scalability and Future Enhancements**

* **Distributed Processing**: Spark, Dask, or cloud-native pipelines for large HPC/GPU logs.
* **Advanced Forecasting**: Use LSTM or Prophet for resource demand.
* **Integration with ITSM**: ServiceNow, PagerDuty for automated incident management.

<a name="security-and-compliance"></a>

## **7. Security and Compliance**

* **Authentication**: Production setup should add Single Sign-On or OAuth.
* **Access Control**: Role-based access to HPC logs vs. corporate DevOps environment.
* **Data Governance**: GPU usage logs or HPC environment logs may include sensitive R&D or PII; handle compliance (GDPR, SOC 2).
* **Encrypted at Rest/In Transit**: All metrics should be encrypted, especially HPC usage data that might be proprietary.

<a name="limitations-and-assumptions"></a>

## **8. Limitations and Assumptions**

1. **Mock Data**: The current code uses synthetic random metrics. Real integration is essential for production.
2. **Simple ML Models**: Isolation Forest, KMeans, and Linear Regression are placeholders for demonstration.
3. **No Real Remediation**: Dashboard only detects anomalies; any self-healing or runbook automation must be integrated.
4. **Single Node**: Streamlit in single-node mode; scaling for thousands of concurrent users requires further engineering.

<a name="roadmap"></a>

## **9. Roadmap**

1. **Integrate Real Telemetry**: HPC cluster logs, GKE metrics, cloud cost APIs.
2. **Advanced AI Models**: LSTM or Prophet for time-series, Autoencoders for anomaly detection.
3. **Incident Lifecycle**: Automatic ticket creation in ServiceNow or custom runbooks.
4. **Multi-Tenancy**: Separate dashboards for different BU or HPC clusters with RBAC.
5. **Cloud-Native Deployment**: Containerize and deploy on GKE with horizontal auto-scaling for the Streamlit service.

<a name="conclusion"></a>

## **10. Conclusion**

This AIOps dashboard provides a robust foundation for **telecom operators** or **enterprises** managing **GenAI**, **HPC**, and **Kubernetes** workloads. By unifying **observability** (metrics, anomalies, costs, incidents) with **ML-driven insights**, teams can proactively detect issues (OOMKills, GPU overloads), forecast resource usage, and reduce overall operational costs and downtimes.

With further integration into real data pipelines, advanced ML models, and enterprise ITSM, this solution can evolve into a **fully automated, scalable AIOps platform** for next-generation HPC and AI workloads.