Use Case Document

# 1. Use Case: Predictive Disk Usage Alert for VictoriaMetrics

## Overview

In HPC Cluster environments, VictoriaMetrics acts as a high-performance time-series database ingesting large volumes of monitoring data — including node health metrics, SLURM job telemetry, system logs, and infrastructure KPIs. Over time, this continuous ingestion leads to significant data accumulation, especially due to high cardinality metrics and long retention settings.  
  
This use case focuses on predictively alerting system administrators when the storage used by VictoriaMetrics is growing at a rate that could exhaust available disk space within a forecasted window — helping prevent monitoring outages, write failures, and observability blind spots.  
  
Note: This alerting is not due to user application data, but due to the backend monitoring infrastructure itself becoming a major consumer of storage over time.

## Example Predictive Alert (Admin-Facing)

[Storage Alert: VictoriaMetrics Forecasted Growth]  
Volume: /data on vm-metrics-store-01  
Current Usage: 2.81 TB of 3.0 TB  
Ingestion Rate: ~6.5 GB/hour from monitoring stack  
  
Based on recent trends, this volume will reach 99% capacity by Sunday, July 14th at 06:00 AM (≈ 3 days from now).  
  
Primary Cause: Rapid accumulation of metrics from SLURM jobs, node exporters, and job-level probes stored in VictoriaMetrics.

## Recommended Actions

• Audit the source of high-ingesting metric streams

• Review and reduce retention settings where safe

• Offload or rotate old data to secondary storage or cold archives

# 2. Use Case: Detecting Kafka Message Rate Anomalies (Drops and Spikes)

## Overview

In a telemetry or log-streaming architecture powered by Kafka, we want to:  
  
• Continuously monitor the message rate per topic — how many messages are received every fixed interval (e.g., every minute)  
• Detect anomalies in message flow:  
 - Sudden drops: A sharp decrease in the number of messages  
 - Unexpected spikes: A rapid surge in message volume  
  
These fluctuations often signal silent failures or bugs. Detecting them early helps maintain reliability across dashboards, alerting engines, and ML pipelines.

## Why This Is Important

Problems Caused by Message Drops:  
• Upstream agent failures  
• Broken subscriptions or misconfigurations  
• Network disruptions  
• Silent data loss undetected by dashboards that only show metrics, not message rate

## Recommended Actions: Detect and Respond to Spikes or Drops in Kafka Message Rates

Use Threshold-Based

→ Best suited for fast setup in low-variance environments where message rates are consistent and predictable. Simple to configure and effective for catching obvious anomalies.  
- Drop alert: When rate falls below 40% of 10-minute average  
- Spike alert: When rate exceeds 2.5× of 10-minute average

Use Z-Score-Based

→ Ideal for anomaly detection in environments with changing, irregular, or seasonal patterns (e.g., time-of-day traffic, weekly cycles). It automatically adapts to fluctuations, reduces manual tuning, and is better at detecting subtle or context-aware deviations in message flow.  
- Drop alert: Z < -2.5  
- Spike alert: Z > 2.5

# 3. Use Case: Data Quality Scoring on Kafka Stream Records

## Overview

We aim to evaluate the quality of every record flowing through our Kafka topics and detect missing or logically inconsistent data early — before it reaches storage systems like OpenSearch, VictoriaMetrics, or other analytics pipelines.

## Why This Is Important

Real-world telemetry pipelines (from Redfish, Slurm, etc.) often include:  
• Missing fields (e.g., no timestamp)  
• Invalid values (e.g., CPU temp = -273.15)  
• Wrong types (e.g., "NA" instead of numeric)  
• Logically inconsistent records (e.g., status = UNKNOWN, but all fields look fine)  
  
Such bad records can silently:  
• Corrupt dashboards  
• Mislead alerting systems  
• Break or bias ML models

## Recommended Actions

• Implement a scoring system that checks for:  
 - Field presence and type correctness  
 - Logical consistency  
 - Known invalid value patterns

• Drop, flag, or quarantine records with poor scores before allowing downstream ingestion

• Monitor score trends to surface data quality regressions by source