Reducing MTTR and Alert Fatigue - Anamoly Detection(DOS)

Kapil Poreddy

PyTorch CONFERENCE 2025

Overview

An intelligent dependency monitoring and incident response system that provides real-time anomaly detection, root cause analysis, and automated remediation for complex distributed systems.

Key Features:
• Automated Dependency Monitoring — Real-time monitoring of infrastructure and application dependencies
• Anomaly Detection — ML-powered detection of abnormal behavior
• Root Cause Analysis — Identifies the source of issues in complex dependency graphs
• Impact Assessment — Evaluates business impact of incidents
• Automated Remediation — Suggests and can execute remediation actions
• Multi-Cloud Support — AWS, Azure, GCP, on-prem
• Kubernetes-Native — Cloud-native by design
• Real-time Alerts — Configurable alerting for critical events

Architecture:
Data Sources → Data Ingestion Layer → Analysis Engine → Alerting/Dashboard/Remediation
Analysis Engine: Anomaly Detection • Root Cause Analysis

Technical Deep Dive

Graph Neural Networks (GNNs):
Message Passing — Nodes aggregate neighbor features
• Attention — Weight critical dependencies (GATv2)
• Hierarchical Learning — Multi-level microservice context

LSTM With Attention:
- Temporal modelling of metrics
- Gates preserve relevant patterns
- Attention focuses on key timestamps
- Learns daily / weekly and baselines

Real time Anomaly Detection
- Sliding window analysis
- Reconstruction error
- Adaptive thresholding
- Dynamic, service-specific baselines

PyTorch Integration:
• PyTorch 2.0+ for training, inference, and deployment

• Dynamic graph + CUDA acceleration for real-time processing

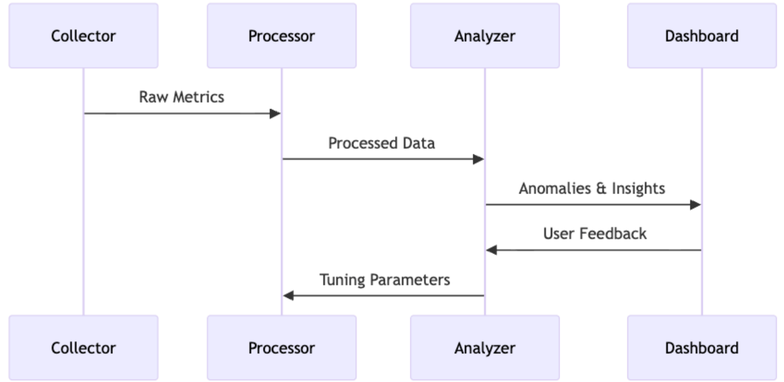
Simulation and Quick Start

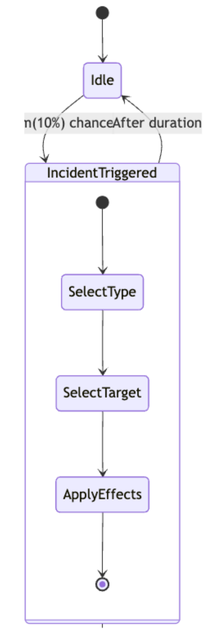
Simulation Environment:
• Realistic service behavior (traffic, resources)
• Incident types: CPU spike, memory leak, network latency, error rate, cascading failures
• Live metrics, alerts, historical analysis

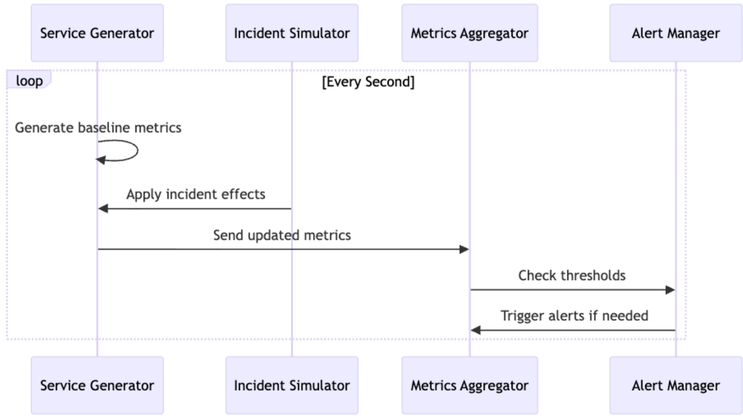
Sample Output:
[14:30:00] All services operating normally
[14:31:15] 🚨 INCIDENT: CPU spike in payment-service
[14:32:00] 📊 payment-service: CPU=85% | Latency=200ms
[14:34:15] ✅ INCIDENT RESOLVED

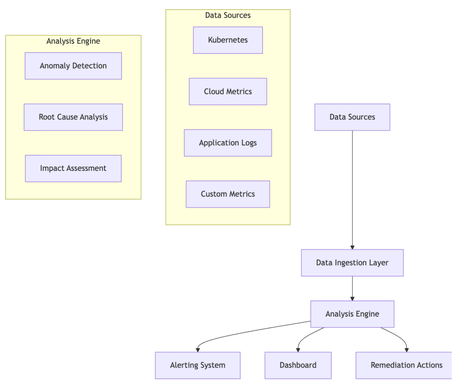
Simulation Flow:
1) Initialization: registry + generators
2) Metric generation @1s intervals
3) Incident injection (random/severity)
4) Impact propagation via graph
5) Monitoring & alerting
6) Recovery & cleanup

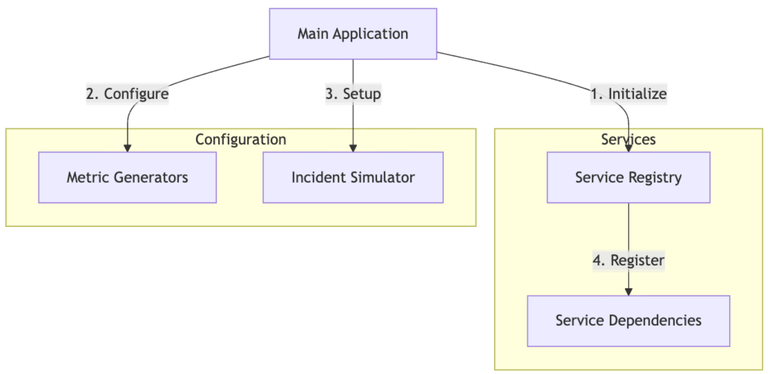
Quick Start:
Prereqs: Python 3.9+, PyTorch 2.0+, Docker (opt), K8s (full)
Run simulation: python simulations/demo_simulation.py

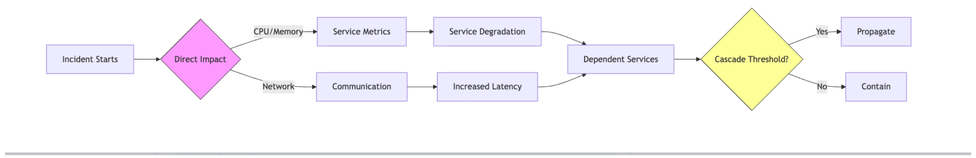












Website: https://kapilporeddy.com
Email: poreddykapil@ieee.org
Connect: https://www.linkedin.com/in/kapilkrp/
GitHub: https://github.com/learningdebunked/Dependency-OPS-Sentinel