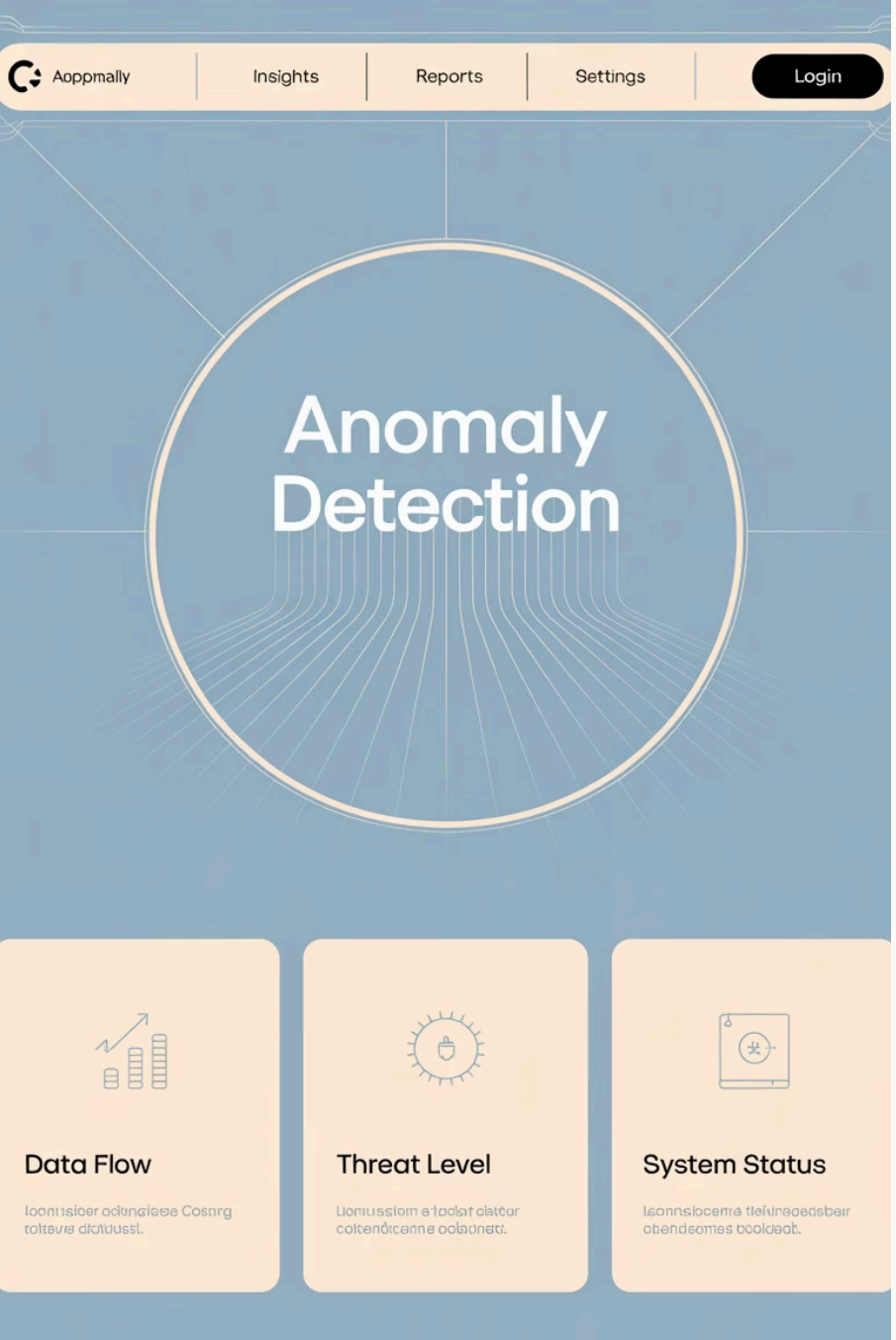


Anomaly Detection for Platform Logs using Embedded Context Agents

Konstantin Berezin

Platform Engineering 2025

Thursday, September 4, 2025 — 5PM GMT

The Challenge of Logs

Modern platforms generate **millions of log entries per day**. Most logs describe routine activity, but hidden within them are signals of:

System failures

Critical errors that impact platform availability and reliability

Performance degradations

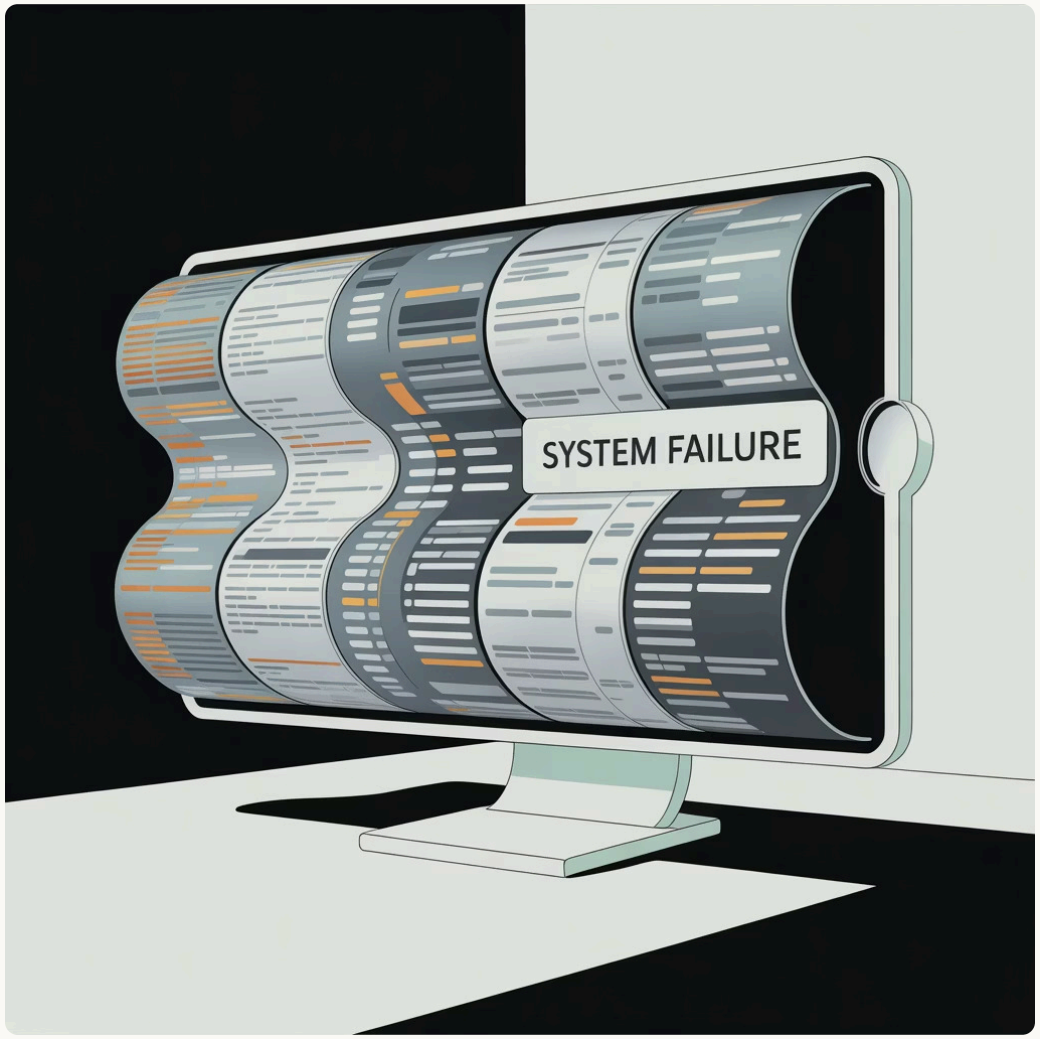
Subtle slowdowns that compound into major user experience issues

Security incidents

Unauthorized access attempts and potential data breaches

Manual analysis is impossible at this scale.

Why It's Hard



High-volume: millions per minute in large systems

Heterogeneous: every service produces its own format

Context-free: a timeout error might be normal… or catastrophic

Noisy: traditional search & alerts often produce too much noise

Classical Approaches

1

Threshold-based rules

"If error count > 100 → alert."

Simple to implement but prone to false positives and requires constant tuning

2

Statistical methods

EWMA, z-score, ARIMA for time series

Better at handling normal variations but still struggle with complex patterns

3

Basic ML

Isolation Forest, clustering, autoencoders on parsed logs

Improved pattern recognition but lacks contextual understanding

Why Classical Approaches Fail

- Too many false positives → engineers stop trusting alerts
- They ignore business context (e.g., deploy in progress vs normal operations)
- Poor at generalizing to unseen log patterns

We need context-aware, adaptive systems that understand the operational environment.

Enter Embedded Context Agents

Embedded Context Agents are:

Embedded

Live inside the monitoring/observability stack, operating in real-time alongside your platform

Contextual

Enrich logs with metadata (user, region, version, deployment) to provide situational awareness

Agents

Autonomous components that analyze, decide, and act based on learned patterns and current state

What Does an Agent Do?

An agent performs a full cycle:

Ingest

Collects logs in real time from various sources across the platform

Enrich

Adds context: which service, which user, which deployment

Act

Raises alerts, correlates incidents, or triggers automated mitigation

Analyze

Examines patterns using ML/embeddings against historical baselines

Analogy — The Security Guard

- **Cameras** = raw logs
- **Guard** = agent

Cameras record everything, but only the guard notices strange behavior.

Context matters:

- A person at night in a closed mall → suspicious
- The same person during Black Friday → perfectly normal

System Architecture

Pipeline for anomaly detection:

Log Ingestion Layer

Collect logs via Fluentd, Logstash, or OpenTelemetry

Contextual Enrichment

Add metadata: user ID, environment, deployment info

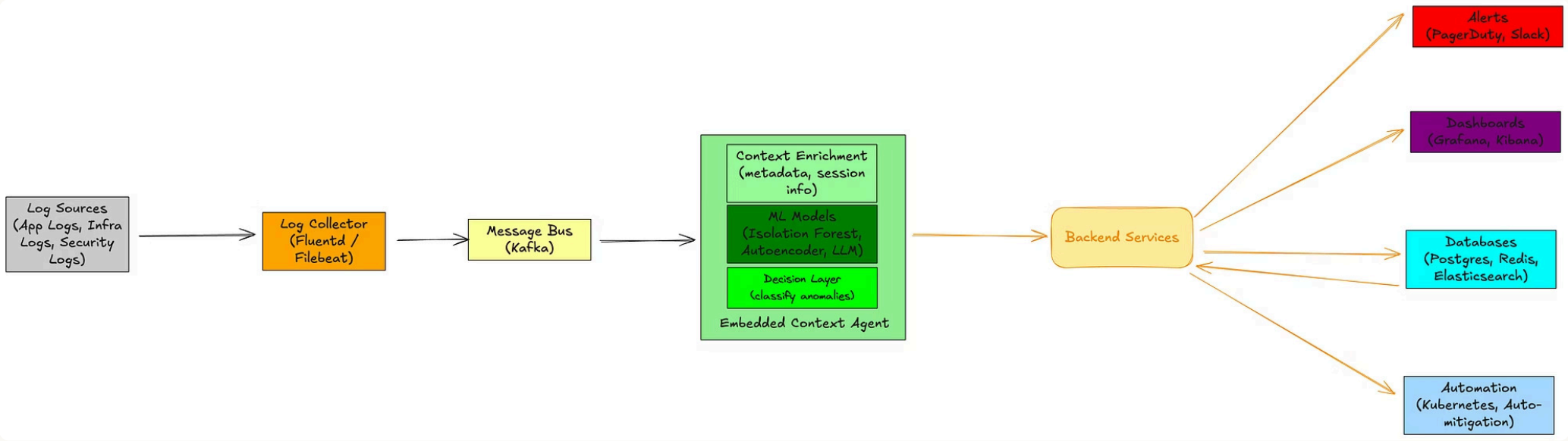
Anomaly Detection Core

ML models analyze sequences for anomalies

Action Layer

Alerts, dashboards, auto-mitigation

Example of System Design



Detection Core — ML Models

Autoencoders

Neural networks that detect unusual log patterns by compressing then reconstructing data

Effective for identifying anomalies in high-dimensional log data

Isolation Forests

Highlight rare, outlier events by measuring how quickly they can be isolated

Computationally efficient for large log volumes

LogBERT

Transformer-based embeddings for semantic log understanding

Captures contextual relationships between log entries

DeepLog (LSTM)

Sequence-based anomaly detection using recurrent neural networks

Learns normal log sequences and flags deviations

Multi-Agent Collaboration

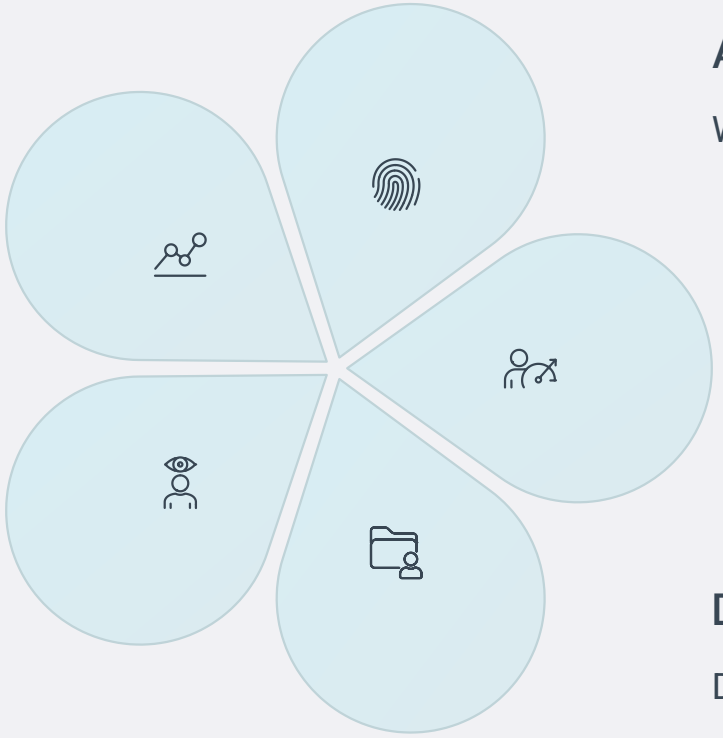
Instead of one monolithic detector, we use specialized agents:

Network Agent

Monitors traffic & connectivity errors

Auth Agent

Watches login patterns & identity events



Performance Agent

Looks for degradation in response times

API Agent

Tracks endpoint health and usage patterns

Database Agent

Detects query issues and data anomalies

Agents coordinate via a shared message bus (e.g., Kafka)

Scenario 1 — Unknown Error

New Error Pattern

Suddenly, logs contain: ERR 500 /api/v2/orders

Embedding Analysis

Embedding comparison shows it doesn't match known error classes

Alert Generated

Agent flags it as new anomaly → SRE alerted

Scenario 2 — Suspicious Logins

Context Addition

Enrichment adds user_id + geo location data to login events

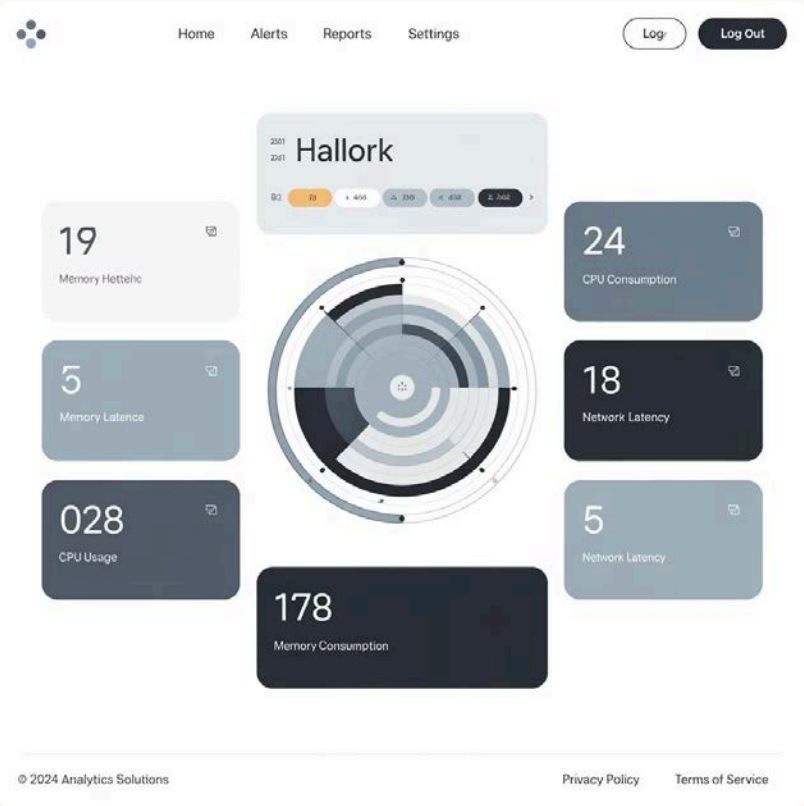
Pattern Detection

Agent notices sharp rise in logins from unusual region

Alert Triggered

Deviation from baseline → possible account takeover attempt

Scenario 3 — Early Performance Warning



Subtle Warning Signs:

- System metrics look fine in traditional monitoring
- But log frequency of "slow query" warnings increases by 15%
- Pattern doesn't match historical baselines

Agent detects deviation early → engineers act before outage

Potential database issue identified 45 minutes before customer impact

ML Models for Log Analytics

DeepLog

Sequence-based (RNN/LSTM) approach that learns normal log patterns

Effective for detecting anomalies in log event sequences

Du et al., CCS 2017

LogBERT

Context-rich transformer embeddings for log representation

Better semantic understanding of log messages

Le et al., ICSE 2021

LogAI (Microsoft)

Open-source library for log parsing, anomaly detection

Production-ready toolkit with multiple algorithm implementations

github.com/microsoft/logai

Drain3

Streaming log template miner for online log processing

Efficiently groups similar log messages into templates

github.com/IBM/Drain3

These models provide the technical foundation for agents

Where Does LLM Fit?

LLM is powerful for logs too:

- Can summarize errors in human-friendly terms
- Can explain patterns across large log batches
- Great for ad-hoc investigation by engineers
- Helps translate technical logs into business impact

Limitations of ChatGPT

Not real-time

Works on-demand, not continuously monitoring

Requires engineer to initiate analysis process

Scaling challenges

Millions of logs per minute is too much

API rate limits and processing delays

Privacy & cost concerns

Sending sensitive logs to LLMs is risky

Expensive for continuous operation

LLM vs ML Agents

Dimension	LLM	ML Agents
Real-time	✗	✓
Scalability	Limited	High
Context awareness	General	Domain-specific
Cost	High	Lower
Best use case	Debug, analysis	Continuous monitoring

Hybrid Approach

Best results come from combining both:

ML agents

Continuous real-time anomaly detection

First line of defense operating 24/7

LLM

Post-analysis, explanation, root cause assistance

Deep investigation of anomalies

Together → fast alerts + deep insights

Benefits of Embedded Context Agents

90%

Noise Reduction

Contextual awareness filters out false positives

45min

Earlier Detection

Average time savings before service impact

65%

MTTR Improvement

Faster incident resolution with context

24/7

Continuous Coverage

Uninterrupted monitoring across environments

Challenges

1

Model training costs

LLMs and embeddings are expensive to train and maintain

Requires significant computational resources

2

Context maintenance

Must always reflect latest deployments & configs

Requires tight integration with CI/CD and configuration management

3

Explainability

Balancing transparency vs black box ML

Engineers need to understand why an alert was triggered

4

Agent coordination

Avoid overlaps and conflicts between specialized agents

Requires robust orchestration layer

Future Directions

RAG for Log Analysis

Retrieval-Augmented Generation to provide context from knowledge bases

Self-healing Systems

Agents that not only detect but fix issues automatically

Federated Anomaly Detection

Agents across clusters share knowledge while preserving privacy

Use Case — DevOps

Deployment Monitoring

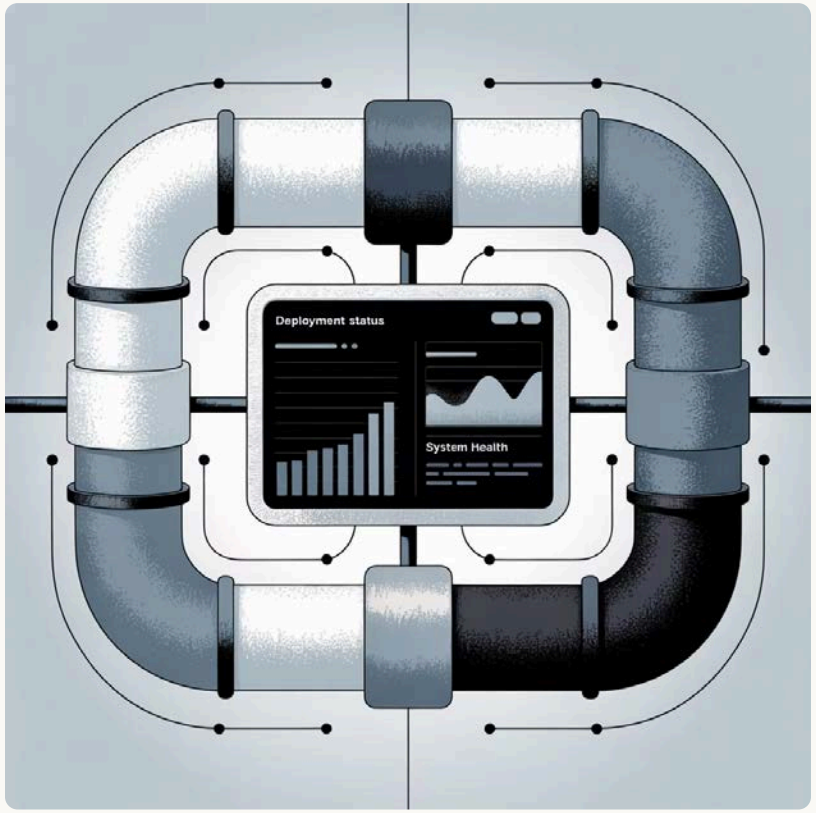
Agent watches logs during deployment process

Anomaly Detection

Detects unexpected patterns different from previous successful deploys

Automated Response

Can trigger rollback automatically if risk threshold exceeded



At Tesla, this approach reduced deployment incidents by 78%

Use Case — Security

Authentication Monitoring:

- Detect anomalies in authentication logs
- Identify suspicious user sequences
- Provide early-warning system for attacks

⚠ **Case Study:** Financial institution detected credential stuffing attack 30 minutes before traditional security tools

Conclusion

Raw logs = noise

Embedded Context Agents transform logs into meaningful signals by combining:

Context

Understanding the operational environment

ML

Pattern recognition at scale

Autonomy

Continuous 24/7 monitoring

With LLM support → human engineers get faster, smarter insights

Konstantin Berezin | mrkonstantinberezin@gmail.com