VPBank Technology Hackathon 2025

General Brief

Team 36 – Opstimus proposes an **AI-powered Metrics Anomaly Detection and Root Cause Analysis** solution designed to enhance the stability and proactive operability of large-scale digital systems in the banking and e-commerce sectors. The solution combines statistical modeling with machine learning algorithms to detect anomalies in real time with high precision, while integrating an AI reasoning layer powered by AWS Bedrock to analyze system data, identify root causes, and recommend corrective actions.
Through an intelligent alerting and automated feedback mechanism, the system helps DevOps teams dramatically reduce Mean Time to Resolution (MTTR), minimize downtime, and move toward an adaptive AIOps operating model.

Challenge Statement	Metrics anomaly detection and Root cause AI
Team Name	Team 36 - Opstimus

Team Members

Full Name	Role	Email Address	School Name	Faculty / Area of Study	LinkedIn Profile URL
Nguyễn Hải Dương	Trưởng nhóm	haiduong66799@gmail.com	HUST	Data – Machine learning	https://www.linkedin.com/in/nguyen-hai-duong-1a34a71ba/
Phạm Đỗ Huy Thành	Thành viên	phamdohuythanh@gmail.com	University Of Greenwich	Computer Science	N/A
Lê Thành Sơn	Thành viên	lethanhson9901@gmail.com	HUST		N/A
Ngân Hà	Thành viên	toanhoc0112358132134 55@gmail.com	HUST		N/A
Trương Đức Thắng	Thành viên	truongducthang30112002@gmail.com	HCMUS		https://www.linkedin.com/in/tdthang/

Content Outline

Solutions Introduction

In today's digital landscape, especially within banking and financial institutions, enterprise systems are becoming increasingly complex, interconnected, and large scale. Early detection of anomalies within these systems has become a critical factor in ensuring stability, security, and seamless customer experience.

Traditional monitoring tools such as Prometheus and Grafana primarily focus on **data collection and visualization**, while the **ability to identify and analyze root causes** of anomalies still relies heavily on human interpretation.
This dependency leads to an excessive number of false alerts or missed critical incidents, forcing operations teams to respond manually, which is time consuming, inefficient, and error-prone.

Our Approach

Team 36 – Opstimus introduces a solution specifically designed to overcome these limitations through the development of a near-real-time metric anomaly detection system.

The system integrates statistical methods with machine learning models (ML) to ensure accuracy, speed, and adaptability across various operational environments. It follows a four-layer architecture: Ingestion & Processing, Detection & Root Cause Analysis, Alerting, and Feedback Loop, forming a closed cycle that enables continuous learning and self-improvement based on real operational data.

1. Ingestion & Processing

This layer is responsible for collecting, aggregating, and processing metrics from multiple sources such as system infrastructure, databases, applications, and operational events. Metrics data is processed in **streaming mode** to maintain real-time responsiveness. During ingestion, data is cleaned, normalized, and temporarily stored to prepare for anomaly detection and analysis.

2. Detection & Root Cause Analysis

This is the core layer of the solution. The system applies statistical methods (e.g., *Z-score, STL decomposition, IQR*) together with machine learning models (e.g., *Random Cut Forest, SR-CNN*) to detect anomalous patterns in metrics in real time. Once an anomaly is detected, the system **automatically analyzes potential root causes** based on operational context such as traffic surges, configuration changes, or specific business events. This combined approach helps reduce false positives and provides actionable, context-rich insights for faster incident remediation.

3. Alerting

The Alerting Layer ensures that notifications reach the right person, at the right time, and with the right context. It integrates seamlessly with widely used incident management tools such as Slack, PagerDuty, or Jira, providing detailed information about severity level, probable root causes, and recommended corrective actions. This mechanism enables technical teams to respond faster and significantly reduce both Mean Time to Detect (MTTD) and Mean Time to Resolve (MTTR).

4. Feedback Loop

The Feedback Loop plays a crucial role in enabling the system to learn and evolve over time. Every operator feedback such as confirming whether an alert is accurate, updating the actual cause, or reviewing historical patterns is captured by the system. These inputs are used to refine detection thresholds, retrain ML models, and enhance detection accuracy in subsequent cycles. As a result, the system becomes increasingly intelligent and adaptive, improving its decision making capabilities with each iteration.

Conclusion

With above architecture, near-real-time data processing, and high detection precision, the solution proposed by Team 36 – Opstimus enables organizations to proactively detect operational risks and automate system monitoring and analysis.

Impact of Solution

2.1. Societal and Target Impact

Our intelligent anomaly detection and analysis solution not only improves system operational efficiency but also creates broader positive impacts for both enterprises and the technology community.

Aspect	Specific Impact
System Operations	Reduces downtime, ensuring 24/7 stability and reliability for banking and financial services.
User Experience	Maintains high service quality and minimizes disruptions in online transactions and financial activities.
Technical Workforce	Reduces pressure on DevSecOps teams. AI assists in detecting, explaining, and recommending corrective actions, allowing engineers to focus on innovation instead of reactive troubleshooting.
Economic & Financial Impact	Minimizes revenue losses due to downtime. One hour of system outage can result in losses of hundreds of millions of VND. The solution shortens Mean Time to Resolution (MTTR) from hours to just minutes.

2.2. Advantages Over Existing Solutions

Criterion	Traditional Solutions (Grafana, Prometheus, …)	Proposed Solution
Detection Approach	Relies on rule-based alerts and fixed thresholds	Combines rule-based, ML, and statistical ensemble models for adaptive real-time detection

Criterion	Traditional Solutions (Grafana, Prometheus, …)	Proposed Solution
Analytical Capability	Displays data; requires human interpretation	AI automatically analyzes root causes and recommends corrective actions
False Positives	High rate, leading to alert fatigue for operation teams	Multi-layer ensemble reduces both false positives and false negatives
Business Context Awareness	None	Understands business context (calendar events, Flash Sales, social trends) to generate relevant alerts
Automation Level	Reactive and manual	Proactive, predictive, and self-healing with automated feedback

2.3. Competitive Advantages and Unique Selling Points

a. Near–real-time Streaming Data Processing

- The system processes over thousands events per second with latency under 5 seconds, powered by a streaming architecture (Kafka + Kinesis).
- Ideal for urgent scenarios such as DDoS attacks, memory leaks, or database overloads.
- Unlike batch systems, this architecture enables instant anomaly detection and real-time action before incidents escalate.

b. Intelligent Hybrid Machine Learning System

Our solution applies a four-layer hybrid anomaly detection framework, combining diverse analytical methods for superior accuracy and reliability.

1. **Statistical Layer:** STL, IQR, and Z-score to capture seasonality and deviations.
2. **Machine Learning Layer:** Random cut tree, SN-CNN and clustering for multi-dimensional anomaly detection.
3. **Rule-based Layer:** Business-specific alerts defined by domain experts.

4. **Graph Analytics:** Analyzes inter-service dependencies to identify cascading failures.

- An anomaly is confirmed only when two or more methods agree, enhancing confidence and reducing false alerts.
- Clustering groups correlated anomalies, enabling fast and accurate root cause correlation across interconnected systems.

c. Real-time Topology Visualization

- The system automatically generates a service dependency map within a microservices architecture, providing a real-time view of how components interact.
- When an anomaly occurs, it highlights upstream and downstream flows, displaying the blast radius and potential impact scope across services.
- The interactive visualization interface uses color codes to represent latency, traffic volume, and service health, allowing engineers to locate and diagnose issues 3–5 times faster than traditional methods.

d. AI Reasoning and Natural Language Interaction

The solution leverages AWS Bedrock as the foundation for its AI Reasoning Layer, enabling contextual understanding, causal inference, and natural human–AI interaction.

Bedrock’s managed LLMs (such as Anthropic Claude or Amazon Titan) process combined signals from metrics, logs, and metadata to analyze anomalies, infer root causes, and generate remediation recommendations with full contextual explanations.

The Bedrock-powered reasoning engine maintains conversational memory, supports follow-up questions, and continuously learns from operator feedback through the Feedback Loop. Over time, it refines its reasoning accuracy, contextual understanding, and the clarity of its explanations.

Deep Dive into Solution

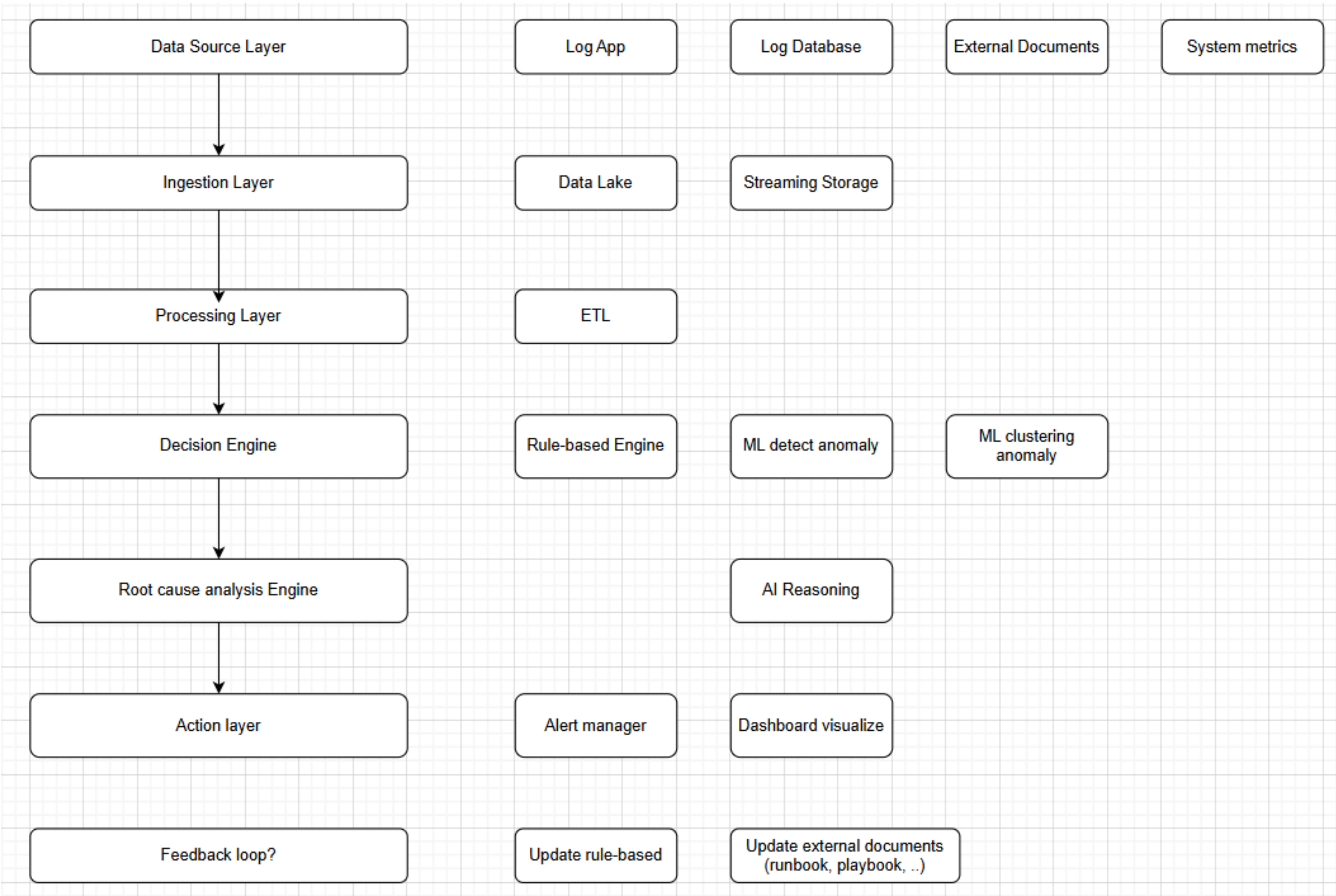


Figure 1: Architecture of solution

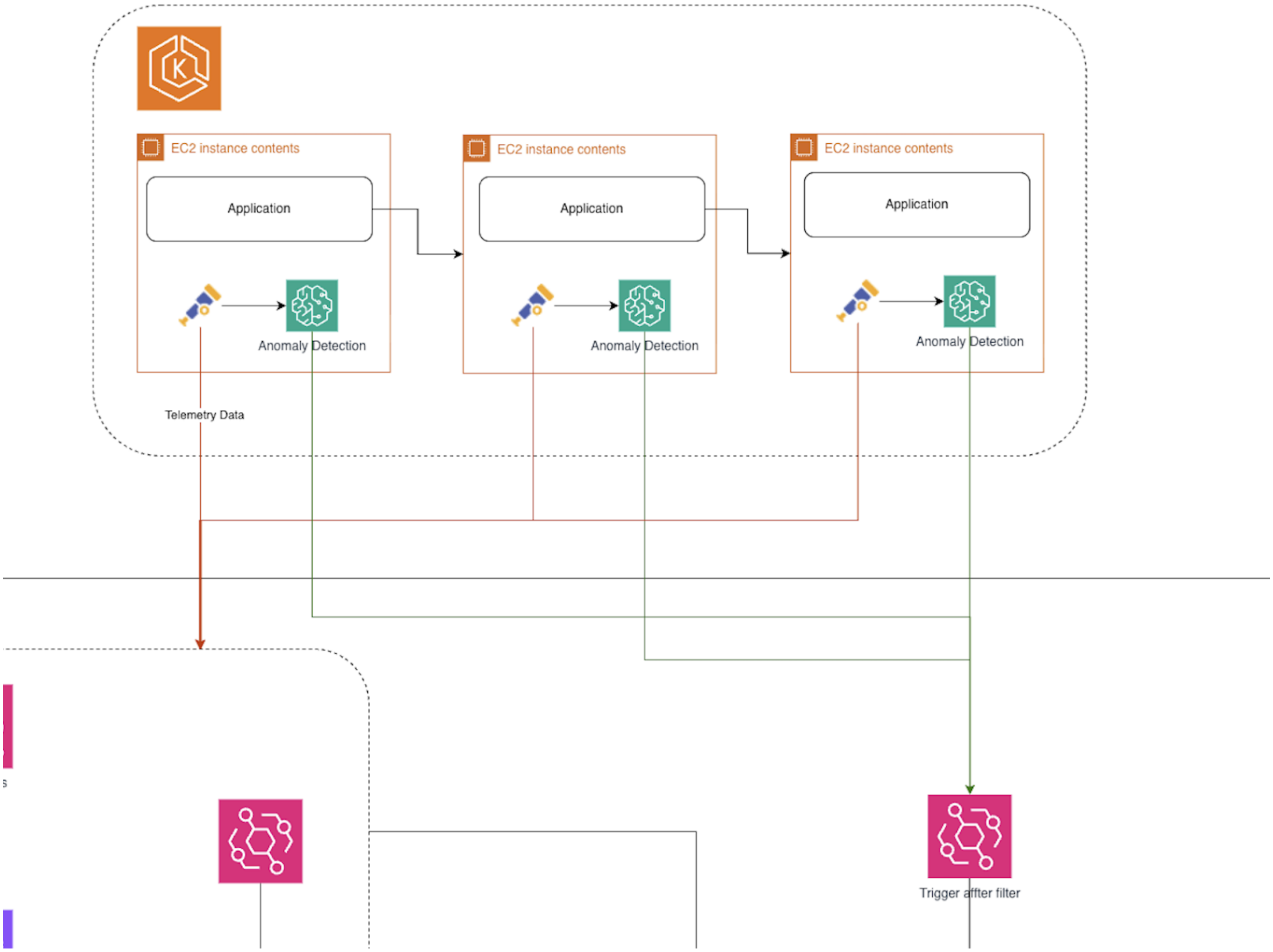
Detailed Analysis of System Layers

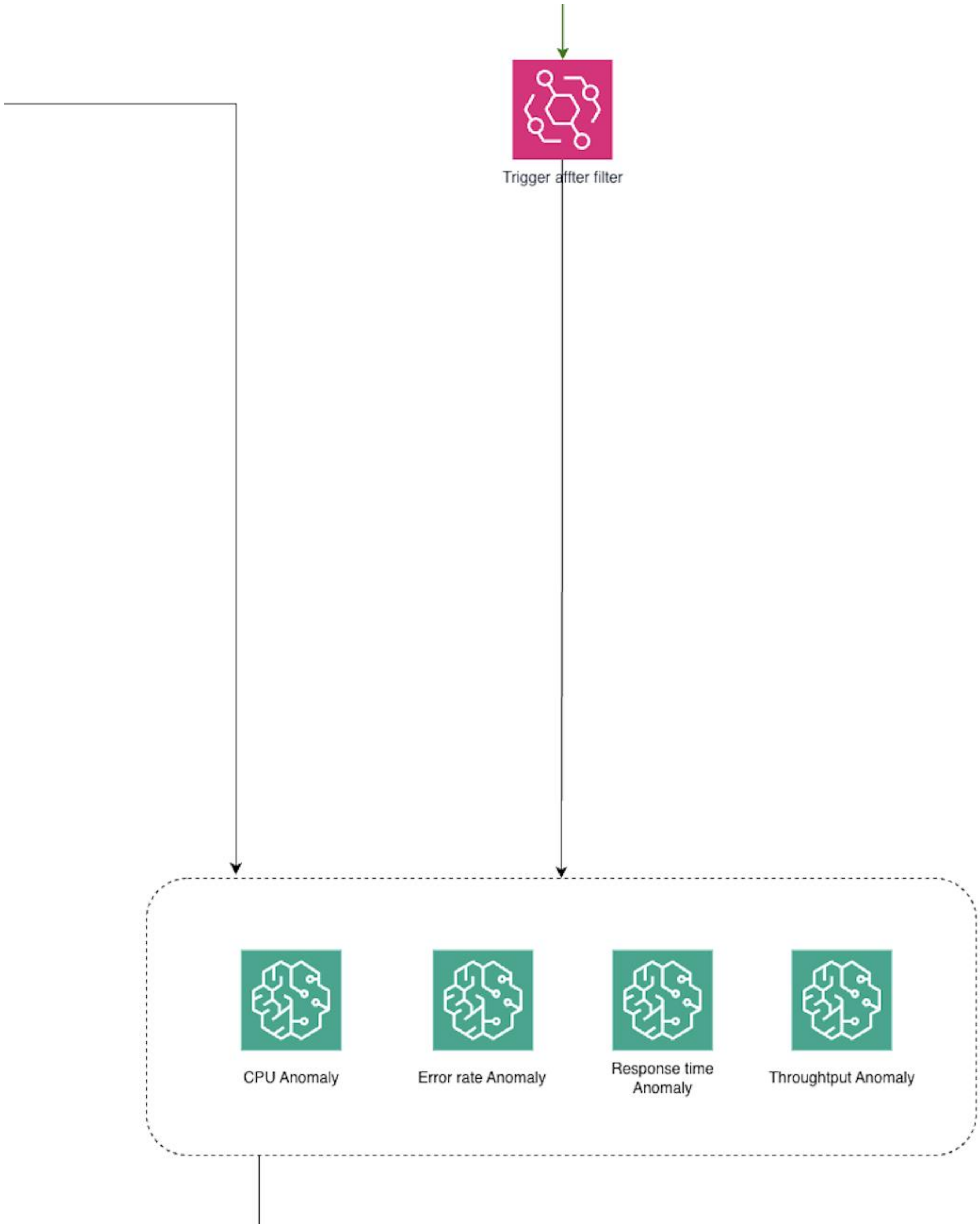
System Layer	Detailed Description
1. Data Collection from Multiple Sources	The system continuously collects data from various sources, including application logs, database logs, system metrics (such as CPU, memory, latency, error rate), and operational documents (runbooks and playbooks). All data is ingested into the Ingestion Layer through the Data Lake or Streaming Storage.
2A. Data Processing and Normalization (ETL)	Within the Processing Layer, the system performs the Extract-Transform-Load (ETL) process. Logs and metrics are cleaned, time-synchronized, and converted into a unified schema to prepare for downstream analysis.
2B. Data Storage and Real-time Analysis	After processing, data is stored in the Data Lake for historical analysis while simultaneously streamed to the Streaming Storage layer for real-time anomaly detection.
3. Anomaly Detection via Rule Engine and Machine Learning	The Decision Engine combines two mechanisms: (1) a Rule-based Engine for static threshold alerts (e.g., CPU > 90%), and (2) an ML-based Anomaly Detection module using models such as STL + IQR, Random Cut Forest, and SR-CNN to capture complex temporal anomalies.
4. Root Cause Analysis with AI Reasoning	Upon detecting an anomaly, the Root Cause Analysis (RCA) Engine is activated to identify the root cause of the issue. The AI Reasoning model leverages temporal correlations, service dependency graphs, and knowledge from operational documents (runbooks/playbooks) to infer cause-effect chains and pinpoint the failing component.

System Layer	Detailed Description
5. Alerting and Real-time Dashboard Visualization	The Action Layer automatically triggers alerts to the Alert Manager (via email, Slack, or internal monitoring systems). At the same time, all analytical results are visualized on a real-time dashboard, allowing operations teams to easily monitor system health and trace root causes.
6. Learning and Feedback Loop	After incident resolution, outcomes are recorded to enable system self-learning: updating rule thresholds, expanding training datasets for ML models, and automatically enriching runbooks and playbooks with new handling procedures. This continuous feedback loop enables progressive system intelligence enhancement.

Architecture of Solution

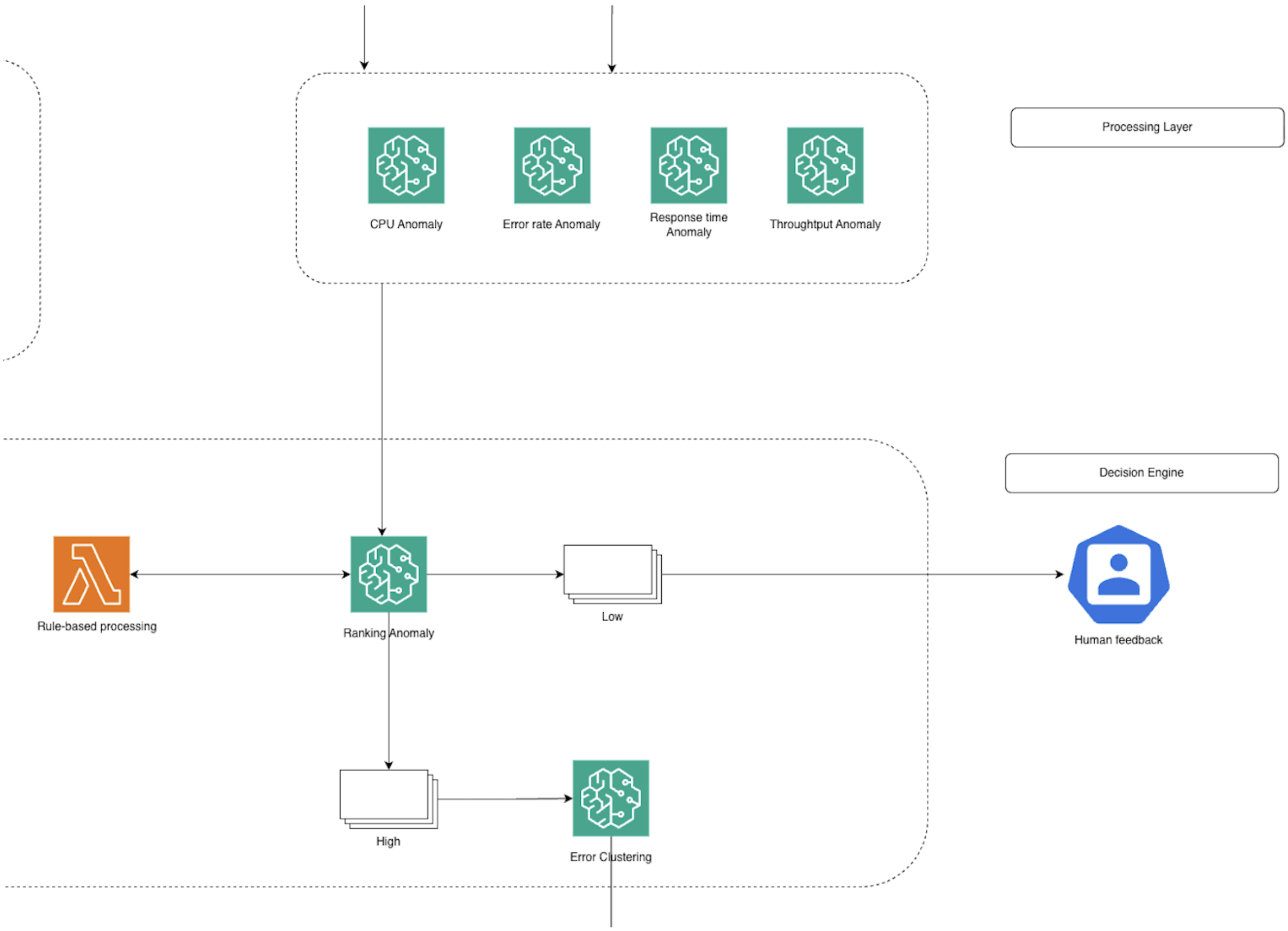
Anomaly flow from each node:



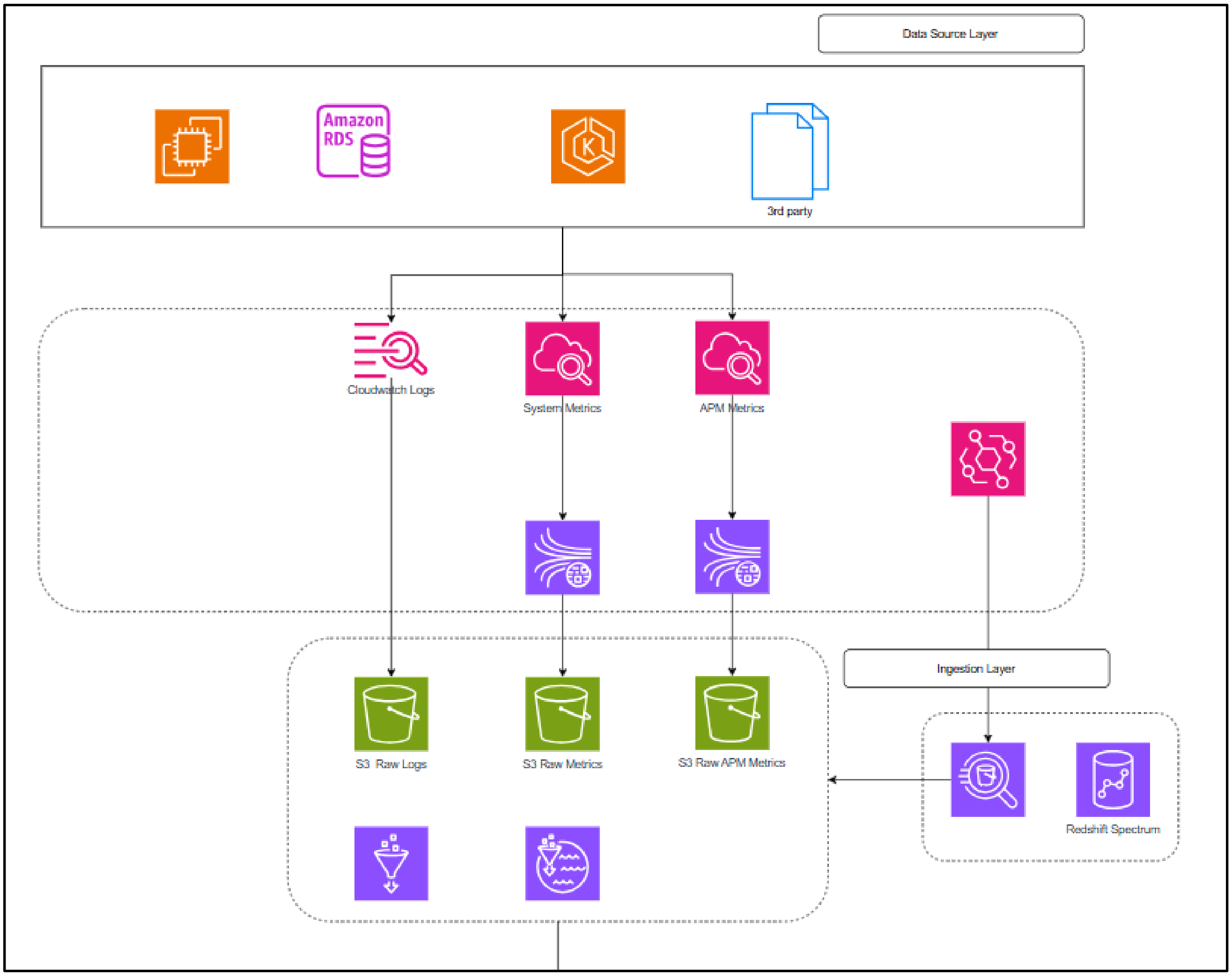


Trigger affter filter
CPU Anomaly
Error rate Anomaly
Response time Anomaly
Throughtput Anomaly

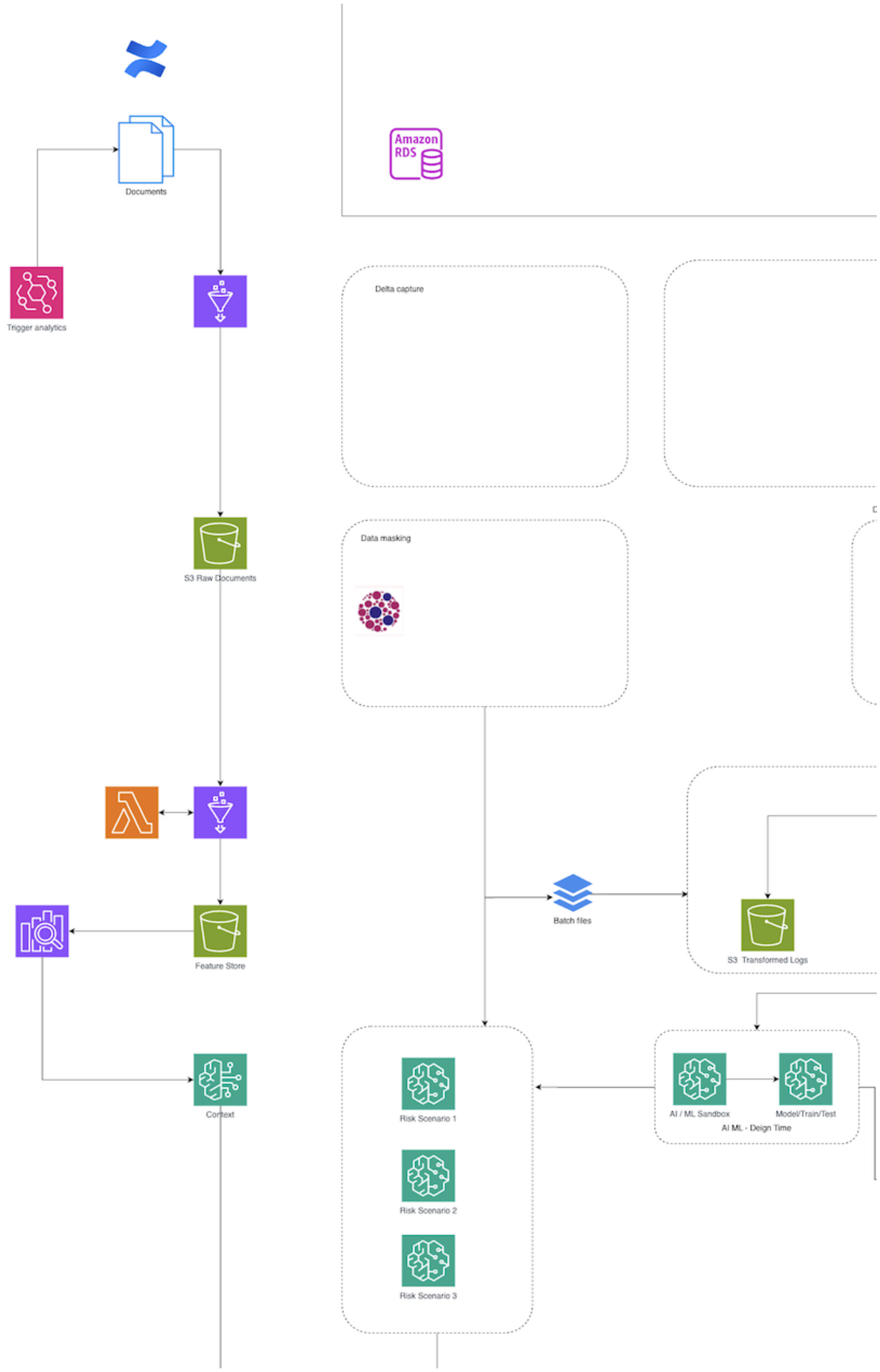
Ranking with human feedback loop and Error Clustering for root cause detection in each node:



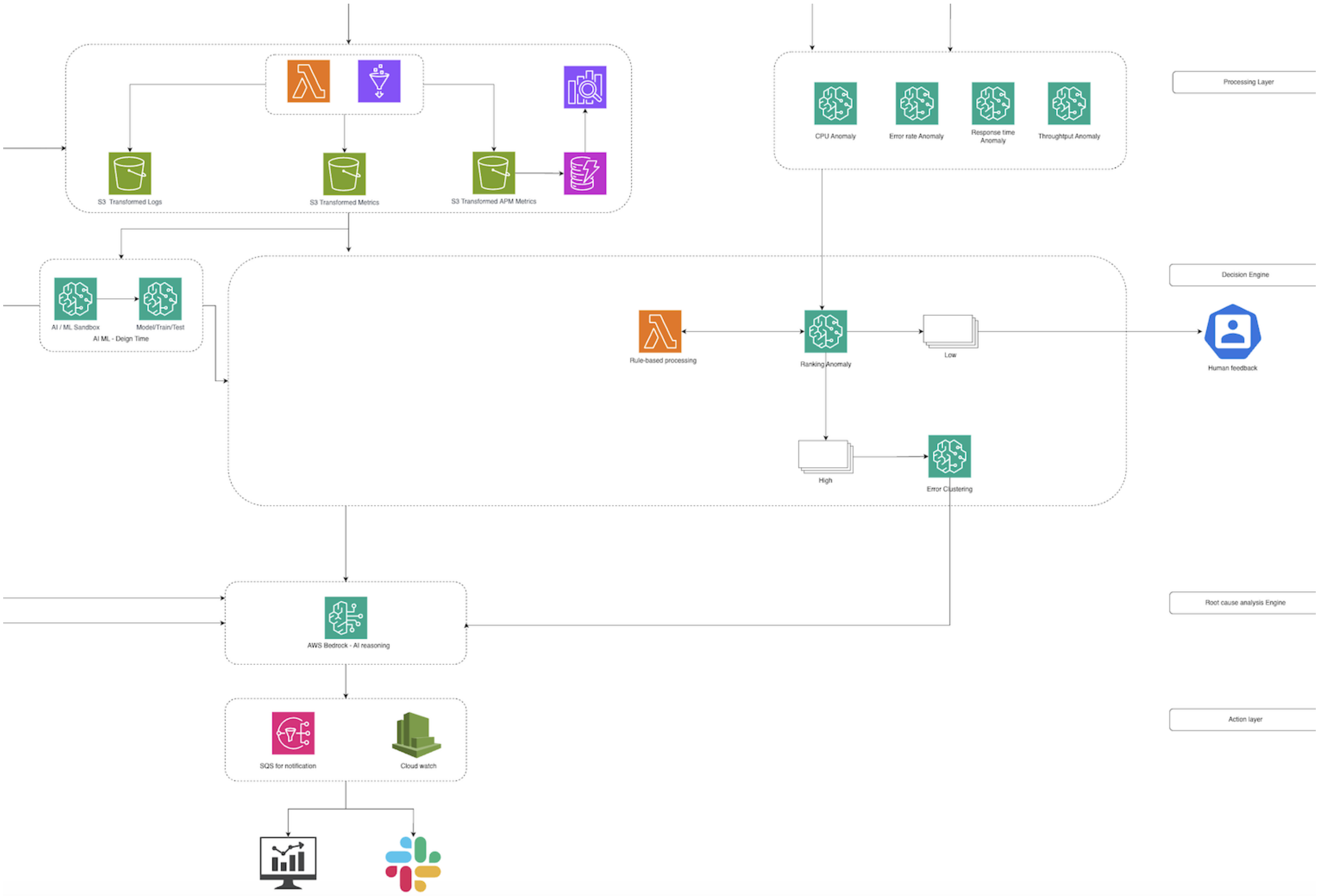
Metrics streaming and storing ready for analytics



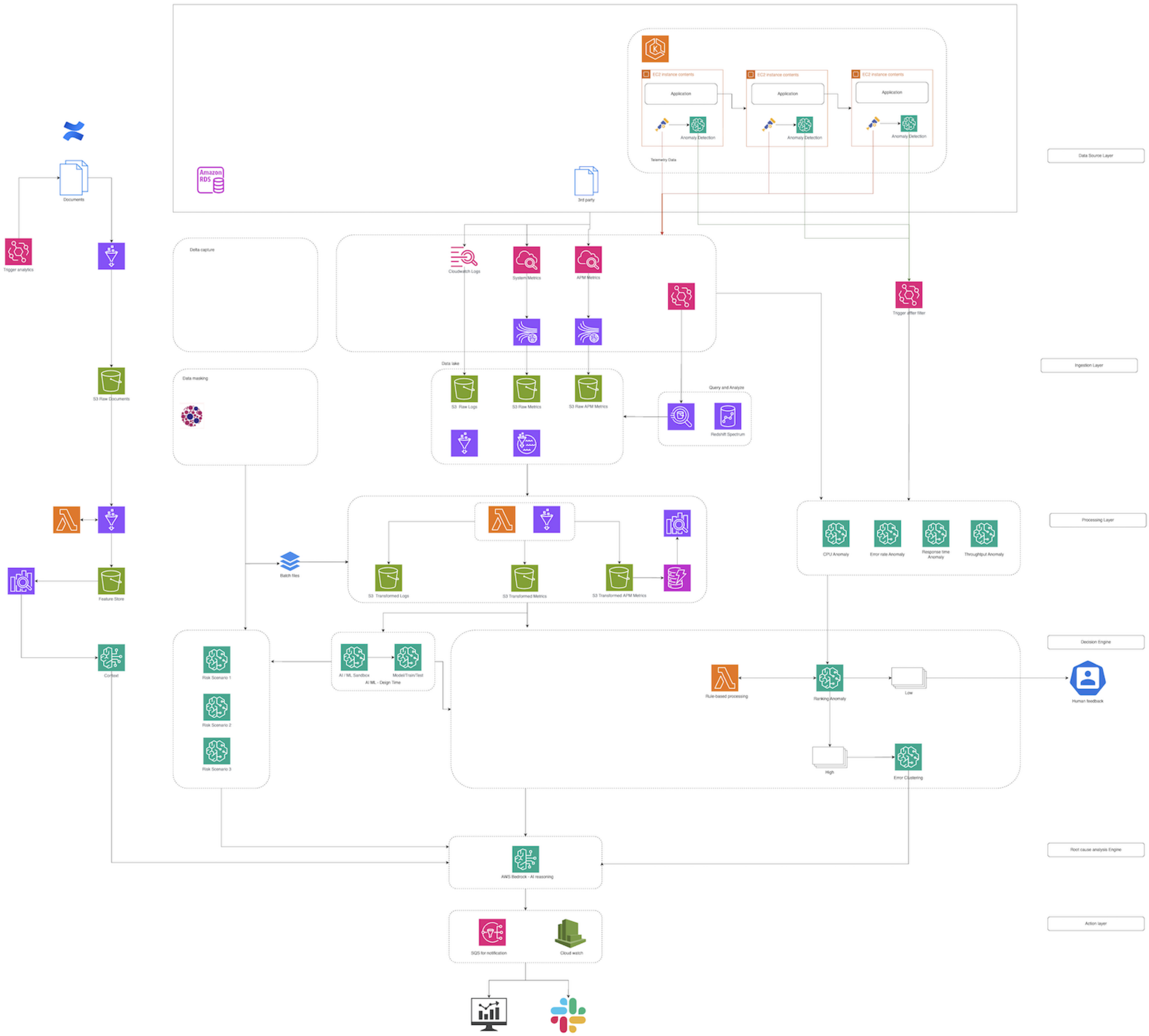
Batch processing for storing context of document and architecture from confluence



Combine and make brief of reasoning



Our full flow diagram



a. AWS Services Utilized in the System

Our solution is built entirely on **AWS native services**, ensuring **high scalability, robust security, and simplified operations**.

- **CloudWatch Logs / Metrics**
  Collects system logs, system metrics, and APM metrics in real time.
  Data is streamed to **Amazon S3** (Raw Logs / Raw Metrics / Raw APM Metrics) for long-term storage and unified querying.
- **Amazon S3 (Data Lake: Raw → Transformed)**

  Serves as the **data backbone** of the system.
  Raw buckets receive unprocessed data, which after ETL is transformed into standardized schema partitions (by time and service) in the *transformed* bucket—ready for analytics and ML inference.
- **AWS Glue (ETL + Data Catalog)**
  Runs ETL jobs for data cleaning, normalization, and partitioning.
  All schemas are registered in the **Glue Data Catalog**, enabling consistent access for **Athena**, **Redshift Spectrum**, and ML models.
- **AWS Lambda (Rule-based Detection & Orchestration)**
  Executes rule-based detection flows to identify threshold breaches in real time.
  Orchestrates the pipeline triggers ETL jobs, invokes ML inference endpoints, and pushes events to **SQS** and **CloudWatch**.
- **Amazon SageMaker (ML Training & Inference)**
  Trains and deploys models such as **STL + IQR**, **SR-CNN**, and **Random Cut Forest** for metric anomaly detection.
  These models are exposed through **SageMaker Endpoints**, allowing the **Decision Engine** to perform **real-time anomaly scoring**.
- **Amazon Bedrock (AI Reasoning)**
  Uses **Large Language Models (LLMs)** to infer root causes from combined ML and log signals.
  References operational knowledge stored in **S3** (runbooks and playbooks) to generate contextual explanations and recommended corrective actions.
- **Amazon SQS + CloudWatch (Action / Alerting)**
  SQS aggregates alerts from multiple sources and routes them based on severity, while **CloudWatch Alarms and Dashboards** visualize metrics in real time and trigger alert notifications through integrated channels.

b. Integration and Interoperation of AWS Services

The end-to-end data flow (as illustrated in the architecture diagram) integrates all AWS components seamlessly:

1. **Ingestion Layer**
    - **CloudWatch** collects Logs, System Metrics, and APM Metrics and writes them to **S3 Raw Buckets** (three categories).
    - **Athena** or **Redshift Spectrum** can directly query raw data for investigations and ad-hoc analysis.
2. **Processing Layer**
    - **Glue Jobs** process and normalize raw data, storing results in **S3 Transformed Buckets** (Logs / Metrics / APM).
    - The **Glue Catalog** provides unified schemas across all analytical and ML workloads.
3. **Decision Engine**
    - **Lambda** executes rule-based detection in real time.
    - In parallel, Lambda invokes **SageMaker Endpoints** for multi-method anomaly scoring.
    - Results are consolidated by service and timestamp for downstream correlation.
4. **Root Cause Analysis Engine**
    - **Bedrock** receives input signals (metric anomalies, log snippets, metadata) and performs **root cause reasoning**, producing explanations and remediation steps.
    - It references **runbooks stored in S3** to ensure actionable and context-aware outputs.
5. **Action Layer**
    - Alerts are sent to **SQS** for prioritization and routing, then visualized on **CloudWatch Dashboards**.
    - Alerts can also be relayed to internal incident response channels (e.g., email, Slack, or ticketing systems).
6. **Feedback Loop**
    - Incident outcomes are stored in **S3 / Glue** and used to periodically retrain ML models and adjust rule thresholds on **SageMaker**.
    - This creates a **self-learning loop**, enabling the system to continuously improve its accuracy and responsiveness.