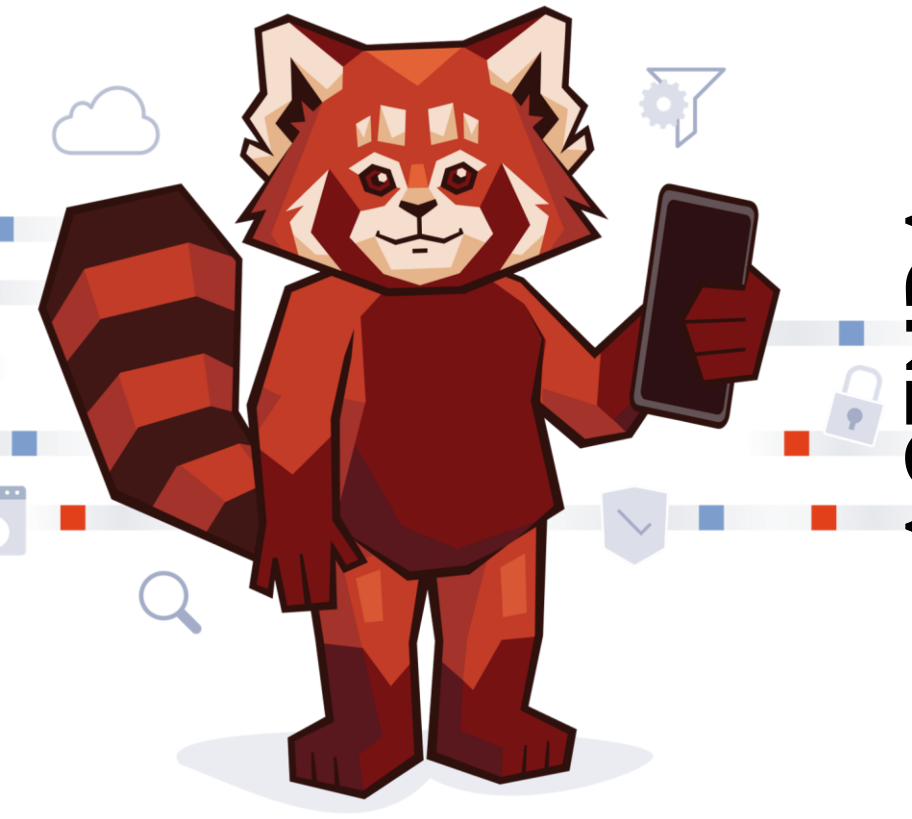
DATA COUCH

Real-Time AI & Agentic Systems with Redpanda

AI Agents & Observability (Developer/AI Focus)

AI Agents & Observability (Developer/AI Focus)

Observability with Prometheus Metrics and Custom Alerts



AGENDA

- Monitoring streaming environments
- Metrics customization and alerting strategies

Introduction to Observability in Streaming Environments

Observability enables teams to understand, monitor, and optimize complex streaming workflows, especially when leveraging agentic AI systems.

Effective observability combines:

- Metrics
- Logs
- Traces

to provide a holistic view of system behavior.

Prometheus Metrics Fundamentals

- **Prometheus** is an open-source monitoring toolkit that collects time-series metrics, models them with key-value pairs, and stores them efficiently.

- **Prometheus** is widely adopted for cloud-native, containerized, and microservices architectures, and integrates natively with platforms like Kubernetes.

Typical use cases :

- Application-specific metrics
- Latency
- Resource utilization
- Tracking service health

Monitoring Agentic AI Streaming Workflows

1. In **AI-driven streaming environments**, observability helps ensure real-time responsiveness, minimal downtime, and effective automation.

2. Multi-agent systems enable autonomous, role-defined monitoring, incorporating decision-making based on live **metric analysis.**

3. Continuous streaming data poses challenges for traditional monitoring tools, making Prometheus and agentic AI essential for handling high **cardinality** and **throughput.**

Alerting Strategies and Best Practices

- **Prometheus** enables precise alerting through PromQL queries, with rules defined for thresholds, anomalies, and service-level objectives (SLOs).
- **Alertmanager** routes notifications via email, webhooks, Slack, or other integrations, supporting custom grouping, inhibition, and severity levels.

Best practices include:

- Aligning alerts with SLOs, avoiding alert fatigue through thoughtful rule design.
- Using templates for standardization, testing alerts before deployment, and creating actionable, context-rich notifications.
- Adopting multi-step escalation with AI agents for critical events, combining contextual analysis and semantic memory.

AI Agents & Observability (Developer/AI Focus)

Resilience Concepts in a BYOC Deployment

AGENDA

- Defining AI Agents vs Agentic AI
- Memory, tools, reasoning loop
- Redpanda as AI brain for real-time use cases

What is an AI Agent?

AI agents are autonomous programs designed to perform specific tasks based on rules or predefined commands.

Features:

- Reactive behavior
- Limited to narrow contexts
- Require event
- Command triggers

Example Use Cases:

- Chatbots
- workflow automation
- virtual assistants.

What is Agentic AI?

Agentic AI refers to advanced, autonomous systems capable of complex, multi-step reasoning and independent decision-making.

Features:

- Goal-setting
- Self-directed learning
- Adapts to dynamic environments
- Proactive control

Example Use Cases:

- Supply chain optimization
- Financial trading
- self-driving vehicles.

Defining AI Agents vs Agentic AI

AI Agents

Rule-based systems designed for task execution within predefined boundaries

They excel at repetitive, predictable tasks but lack true autonomy.

Agentic AI

Autonomous, goal-driven entities capable of learning, adapting, and advanced reasoning

Manage complex workflows, strategize, and refine objectives in real time.

In observability contexts, agentic AI can dynamically monitor, troubleshoot, and optimize real-time systems with minimal human supervision.

Key Differences & Why They Matter

Feature	AI Agents	Agentic AI
Autonomy	Task-specific, reactive	Multi-step, proactive
Collaboration	Operates solo	Multi-agent coordination
Learning & Adaptation	Rule-based	Meta-learning, continual
Scope of Application	Narrow	Broad & cross-domain

Role of Memory in Agentic AI

Types

Short-term (context, session data), **long-term** (experience, knowledge base).

Value

Enables continuity, recall for multi-step tasks, personalizes agentic behavior.

Developer Focus

Efficient memory handling for session linking, past decisions.

Tool Use & Integration

Definition

Tools extend AI capabilities
e.g.
- Databases
- APIs
- Plugins.

Agentic Example

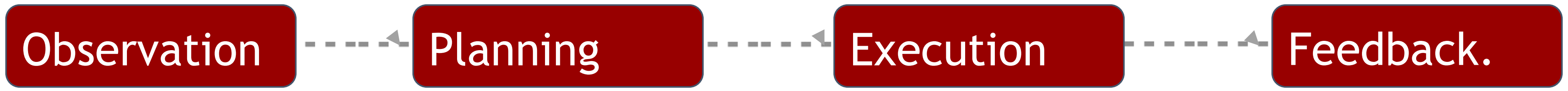
Fetch real-time stock data
Automate server deployment
Integrate third-party services.

Developer Focus

- Secure tool access
- Runtime orchestration
- Monitoring tool performance.

The Reasoning Loop

Loop Steps:



Advantage:

- Enables adaptation
- Continuous improvement
- Error correction.

Developer Focus:

- Streamed logging
- Traceability
- Metric capture for agentic steps.

Core Components - Memory, Tools, Reasoning Loop

- **Memory:** Retains context, past decisions, and interactions for continuity and adaptation.
- **Tools:** Extend capabilities with computation, data retrieval, and external actions (APIs, scripts, databases).
- **Reasoning Loop:** Iterative “think-act-observe” cycle for planning, execution, and adaptive refinement.
- **Redpanda Integration:** Streaming insights provide continuous validation and decision improvement.

Why Redpanda for Agentic AI?

Low-latency Streaming

High-throughput, low-latency data backbone, scalable for real-time agentic workflows.

Integration

Directly connects with modern observability stacks (Prometheus, Grafana).

Developer Focus

Simplifies data pipeline setup, accelerates event publishing.

Observability Features in Redpanda

Real-Time Metrics

Exposes granular metrics for streaming workflows, agents, and interactions.

Custom Alerts

Supports Prometheus-native alerting for failures, agent inactivity, performance bottlenecks.

Developer Focus

Dashboarding, alert rules for agentic workflows, correlation with logs and traces.

Agentic Reasoning in Practice

Architecture

Multi-agent systems ingest, process, and reason about streaming data with Redpanda at the core.

Benefits

Accelerated troubleshooting, autonomous event response, smarter automation.

Developer Focus

Monitoring agent reasoning loops, ensuring observability, optimizing resource usage.

Redpanda as AI Brain for Real-Time Use Cases

- Memory: Retains context, past decisions, and interactions for continuity and adaptation.
- Tools: Extend capabilities with computation, data retrieval, and external actions (APIs, scripts, databases).
- Reasoning Loop: Iterative “think-act-observe” cycle for planning, execution, and adaptive refinement.
- Redpanda Integration: Streaming insights provide continuous validation and decision improvement.

Agentic Reasoning in Practice

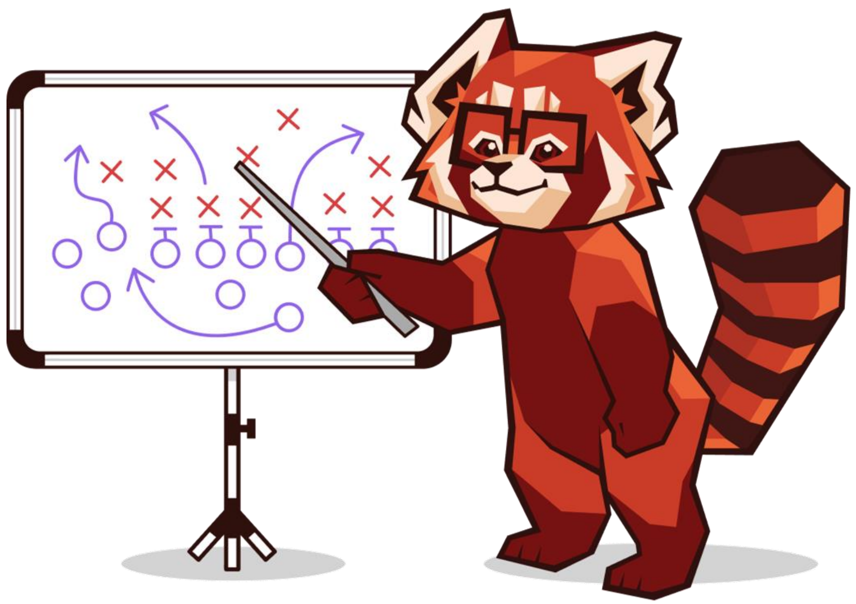
Multi-agent systems ingest, process, and reason about streaming data with Redpanda at the core.

Benefits:

- Accelerated troubleshooting
- Autonomous event response
- Smarter automation

Developer Focus:

- Monitoring agent reasoning loops
- Ensuring observability
- Optimizing resource usage



AI Agents & Observability (Developer/AI Focus)

Prompt Engineering Essentials

AGENDA

- Zero-shot, few-shot, chain-of-thought prompting
- Structuring context for LLMs

Zero-Shot Prompting Overview

Asking the LLM to perform a task without providing any examples or demonstrations in the prompt.

Example:

Prompt: Classify the sentiment of this sentence as positive, neutral, or negative.

“I think the vacation was okay.”

Output: "Neutral".

Pros

Requires less prompt engineering fast and simple.

Cons:

Can produce unpredictable results for complex tasks or novel contexts.

Few-Shot Prompting Overview

Providing multiple examples within the prompt to guide the LLM on how to perform the task.

Example:

Prompt includes 2-3 sentiment-labeled sentences, then a new sentence to classify:

Text: The product is terrible.
Sentiment: Negative
Text: Super helpful, worth it.
Sentiment: Positive
Text: It doesn't work!
Sentiment: ?

Output: "Negative".

Pros

More reliable for nuanced tasks, helps model generalize patterns.

Cons:

Uses more tokens; careful example selection needed.

Chain-of-Thought Prompting (CoT)

Encouraging the model to generate intermediate reasoning steps before final answers.

Example:

"If there are 3 apples and you buy 2 more, how many do you have? Think through the problem step-by-step."

Output: "3 + 2 = 5. So, you have 5 apples.".

Benefits:

Improves performance on reasoning tasks, makes model output interpretable.

Structuring Context for LLMs Effectively

- **Clear Instructions:** Start with explicit instructions or questions defining the task clearly.
- **Use Relevant Examples:** For few-shot prompting, include diverse, representative examples showing correct and incorrect patterns as needed.
- **Keep Context Manageable:** Balance between enough examples and prompt size; avoid overloading the model.
- **Use Delimiters:** Separate instructions, examples, and user input clearly with delimiters or markup.
- **Version and Iterate:** Track prompt versions to monitor changes in output quality and performance.

Common Pitfalls and Best Practices

Avoid Ambiguity:

Vague or poorly defined instructions often lead to inconsistent outputs; be precise and concrete.

Token Limit Awareness:

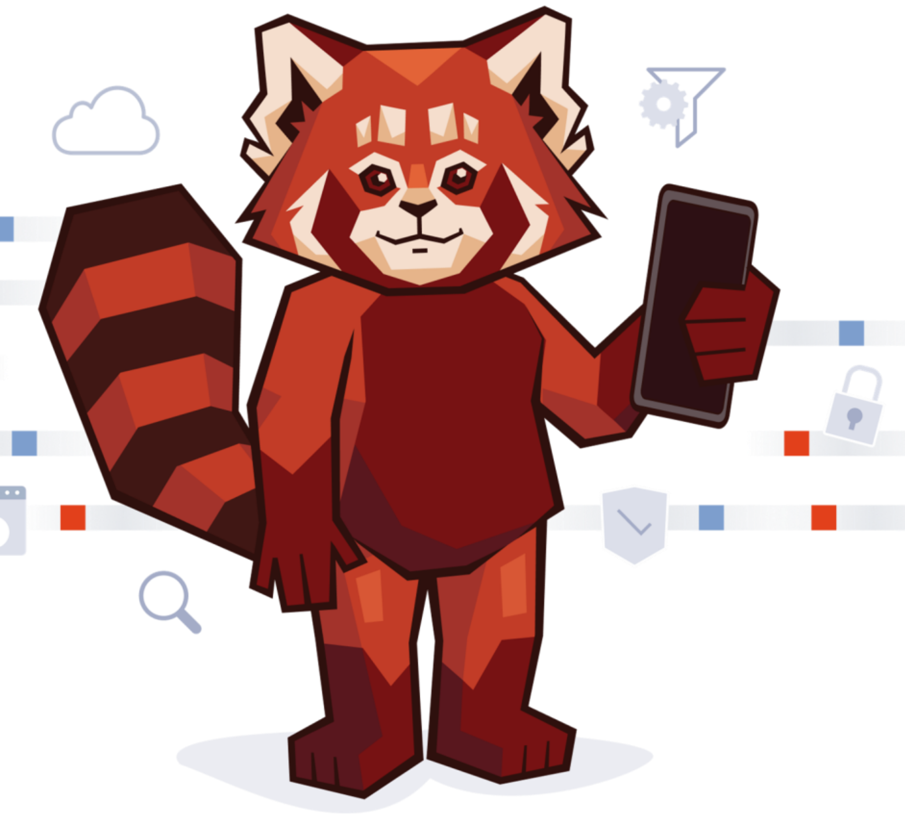
Large context consumes tokens; optimize prompt length vs. completeness to avoid truncation.

Bias and Ethical Considerations:

Prompts can influence model biases; evaluate outputs for fairness and mitigate harmful content.

AI Agents & Observability (Developer/AI Focus)

Agentic Pattern - Event → Enrichment → Routing



AGENDA

- Real-time event flow architecture using Redpanda
- External enrichment with OpenAI APIs via Connectors
- Routing outputs to alert dashboards, compliance systems

Introduction to Redpanda Event Streaming Architecture

- Redpanda is a fast, Kafka-compatible platform for real-time data.
- Producers send events to topics (ordered event logs).
- Events are safely stored and copied across nodes using Raft for reliability.
- Consumers read events from topics independently, making processing scalable.

Key benefit:
Producers do not wait for consumers, enabling high-throughput and low-latency event flow with replayability for flexible consumption.

Core Components & Data Flow in Redpanda

Producers:

Emit events continuously to Redpanda topics, encapsulating actions and metadata with timestamps.

Brokers:

Redpanda nodes store and manage event logs, replicate data for resilience, and orchestrate cluster coordination.

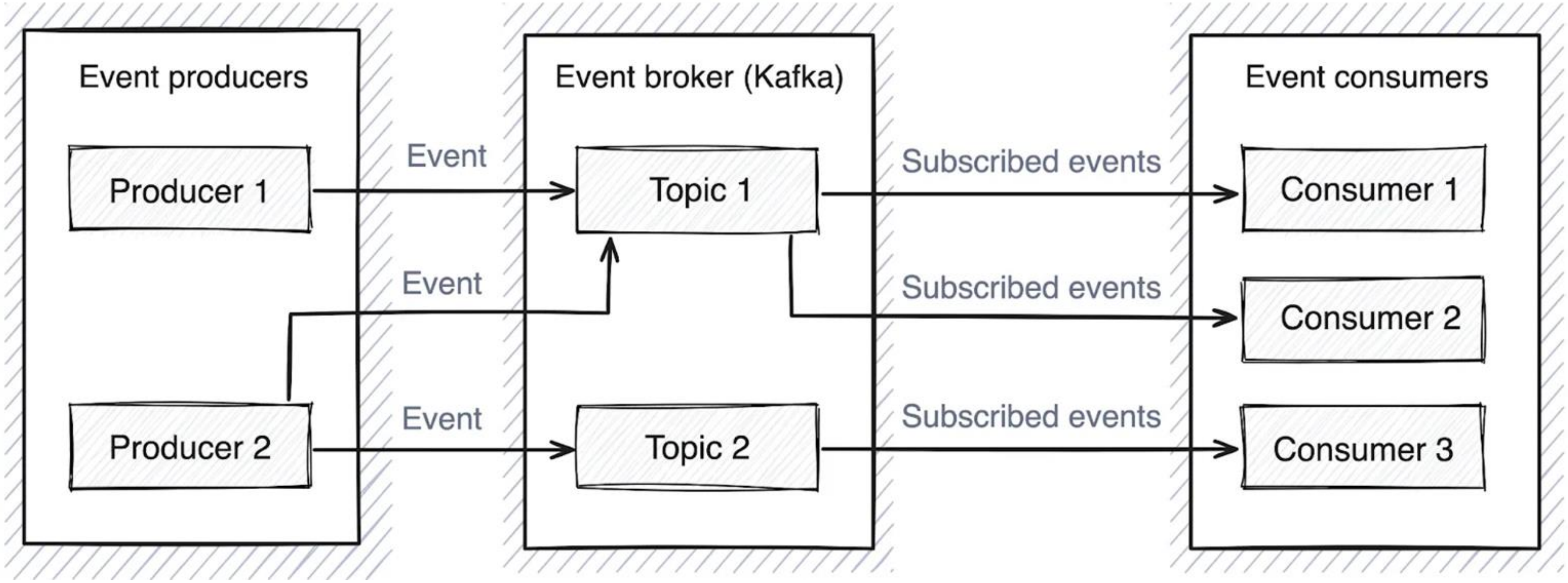
Raft Consensus:

Ensures data consistency and fault tolerance by replicating writes across leader and follower nodes.

Consumers:

Independently consume event streams for processing (analytics, enrichment, alerting, ML pipelines).

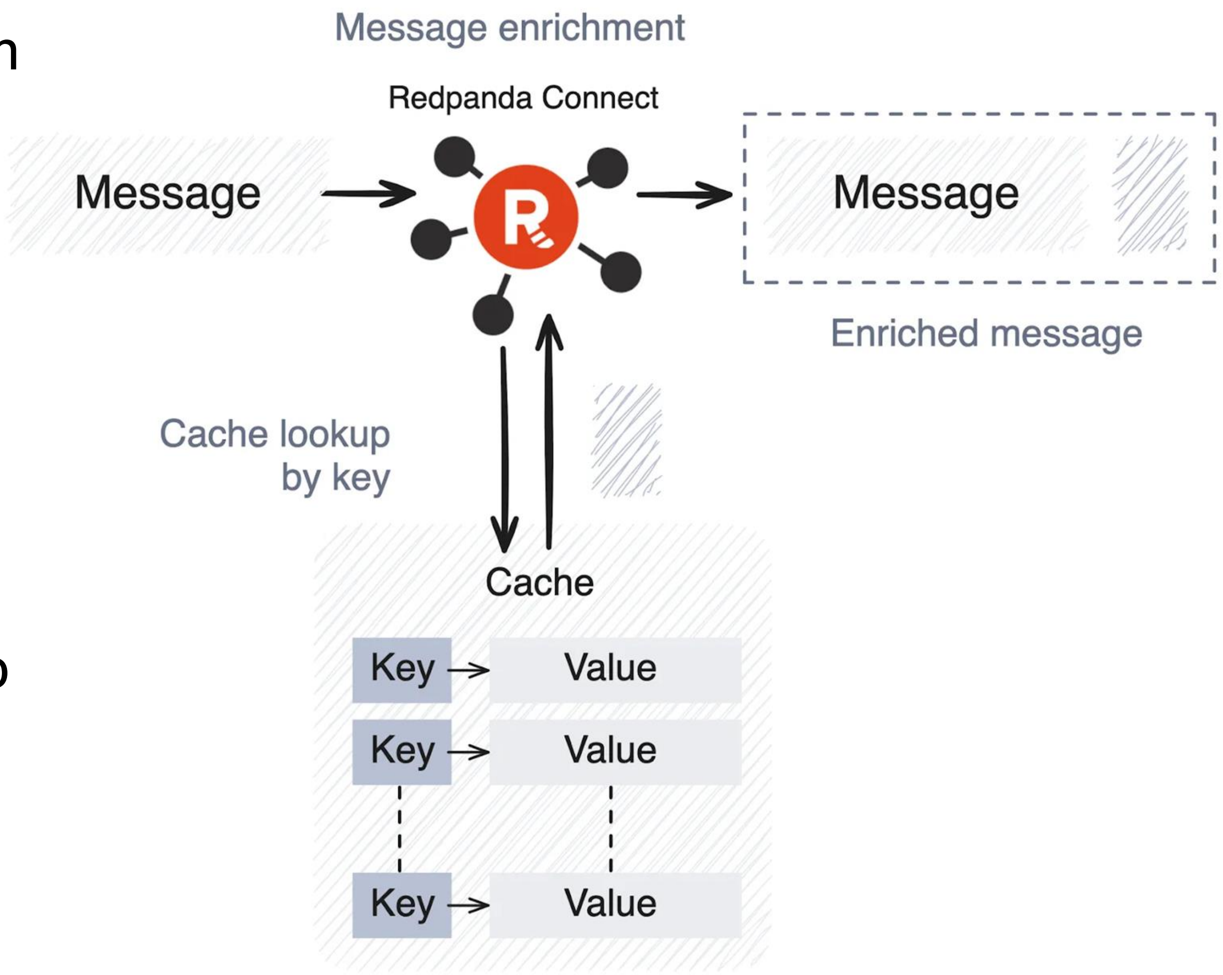
Architecture



Event Streaming Architecture

Connector-Based Enrichment Design

- Connectors read raw events from Redpanda.
- AI services enrich events.
- Decoupled layer enables scaling and fault isolation.
- Enriched events are published to new topics.



Benefits and Best Practices

1. Modular enrichment reduces workload on producers and centralizes AI API usage.
2. Enables API rate limiting, cost management, and error handling within connectors.
3. Facilitates observability by capturing pre- and post-enrichment event states.
4. Supports multi-model and multi-tenant enrichment pipelines for versatility

Real-Time Routing to Alert Dashboards

1

Alerting systems consume enriched streams with rule-based or threshold filters.

2

Alerts are shown in real time on Grafana dashboards as actionable insights.

3

Supports fast incident response, anomaly detection, and monitoring.

Compliance Monitoring and Multi-Destination Routing

- Routing allows copying or forwarding events to compliance and audit systems for regulatory adherence.
- Supports data masking, secure delivery, and audit trail preservation.
- Enables simultaneous multi-destination routing—supporting operational, analytic, and governance needs in parallel.
- Dynamic routing policies help maintain system flexibility and adaptability.

THANK YOU