NetEye

NETEYE CONFERENCE 2025

From Infrastructure to Application: End-to-End Observability in Kubernetes with Elastic and NetEye

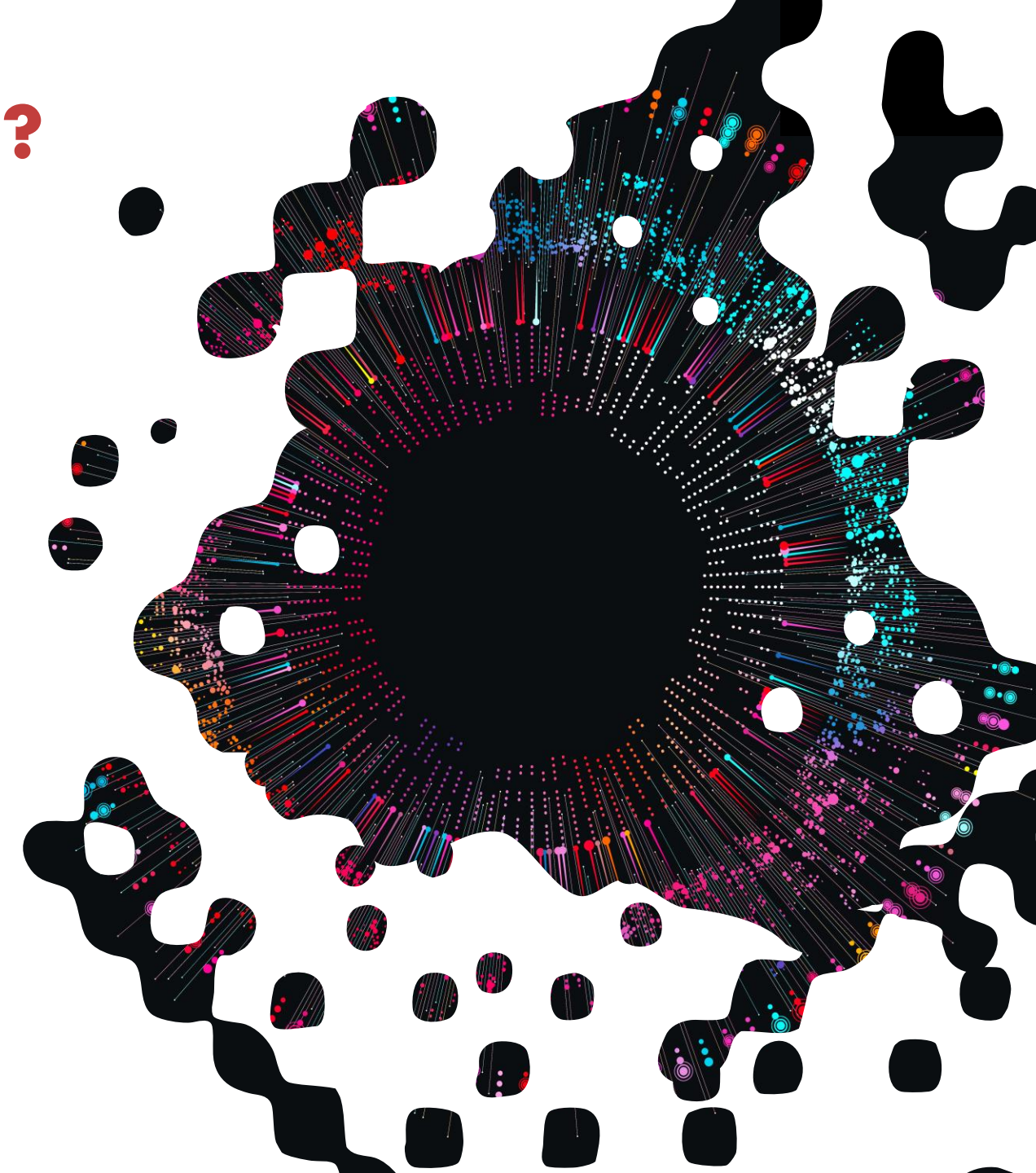
Matteo Cipolletta, Technical Consultant Würth Phoenix

Agenda

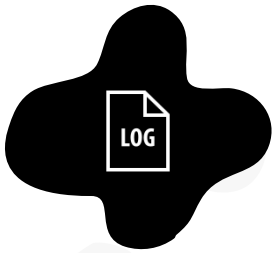
- Introduction & Context
- The Pillars of Observability
- Elastic components in Kubernetes
- The Role of NetEye
- Architecture
- Key Takeaways

Why Observability in K8s?

- Kubernetes introduces levels of complexity (Clusters, Nodes, Pods, Namespaces, Services)
- Failures can happen at **multiple** levels: infrastructure, network and application
- Microservices make root cause detection **harder**
- Traditional monitoring tools often provide partial information
- **End-to-end** observability is critical for fast troubleshooting and reliable operations

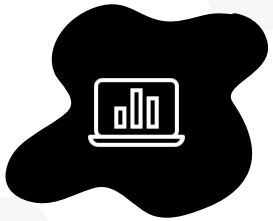


The Pillars of Observability



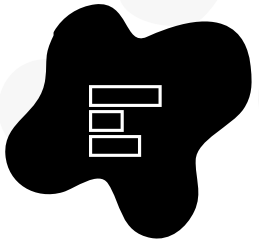
Logs

- Capture **application** and **system events**
- Provide detailed context for errors and issues
- Useful for debugging and forensic analysis



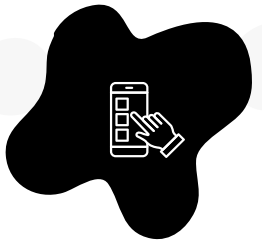
Metrics

- Numeric values over time: CPU, Memory, Latency, Throughput
- Monitor **performance** and **trends**
- Detect abnormal patterns before they lead to failures



Traces

- Follow individual requests across multiple services
- Identify **bottlenecks**, **slow transactions**, or **failed calls** in distributed applications
- Link application performance to infrastructure behavior



Synthetic Monitoring

- Simulate user journeys
- Measure availability and response times
- Detect issues proactively before impacting real users

Elastic components in Kubernetes

Elastic Agent

- Runs on every node in the cluster as DeamonSet
- Collects **logs and metrics** from pods, containers and nodes
- Sends data securely to Elasticsearch
- Scales automatically as the cluster grows
- Runs on a “cluster” level as a StatefulSet to get cluster-related metrics

Elastic APM & OpenTelemetry Agents

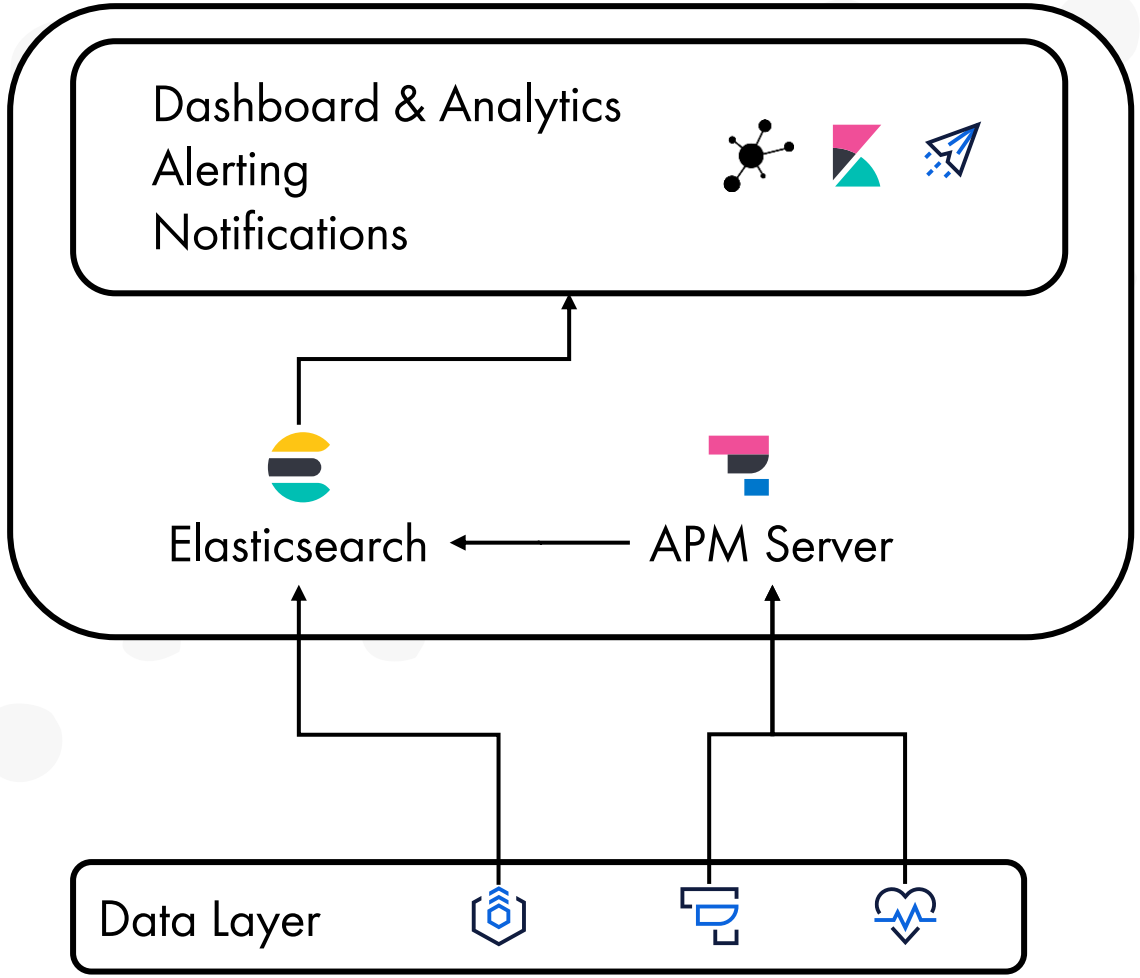
- Instrument application code inside pods
- Capture **traces** across services and requests
- Link performance issues with infrastructure and logs
- Monitor **throughput**, **latency** and **error rate** of services and linked dependencies (Databases, Redis, etc.)

Browser Monitors & HTTP/S, TCP, and ICMP monitors

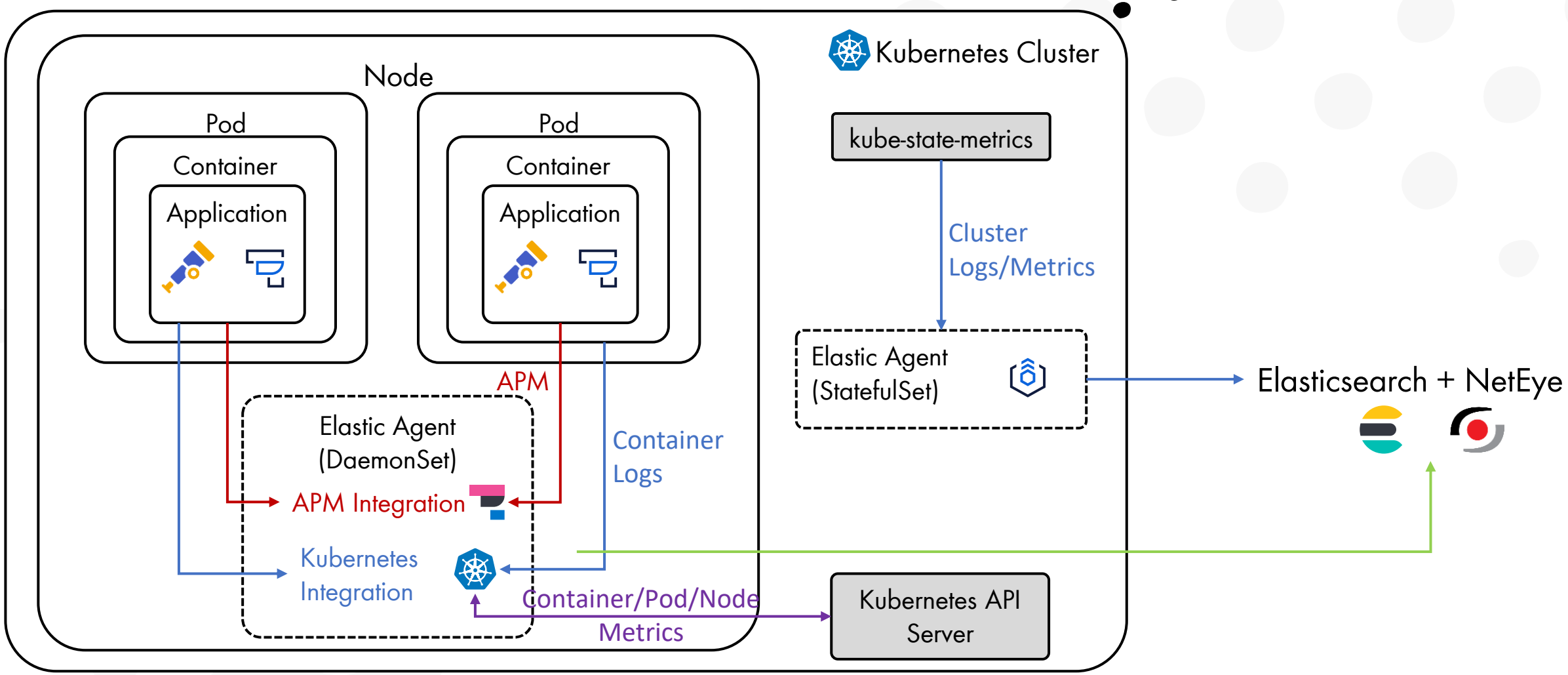
- **Simulates** real user journeys (HTTP checks, Browser interactions)
- Monitors **uptime** and **SLA compliance**
- Monitors pages response time, network request and “objects” weight (CSS, JS, Images, etc.)

The role of NetEye

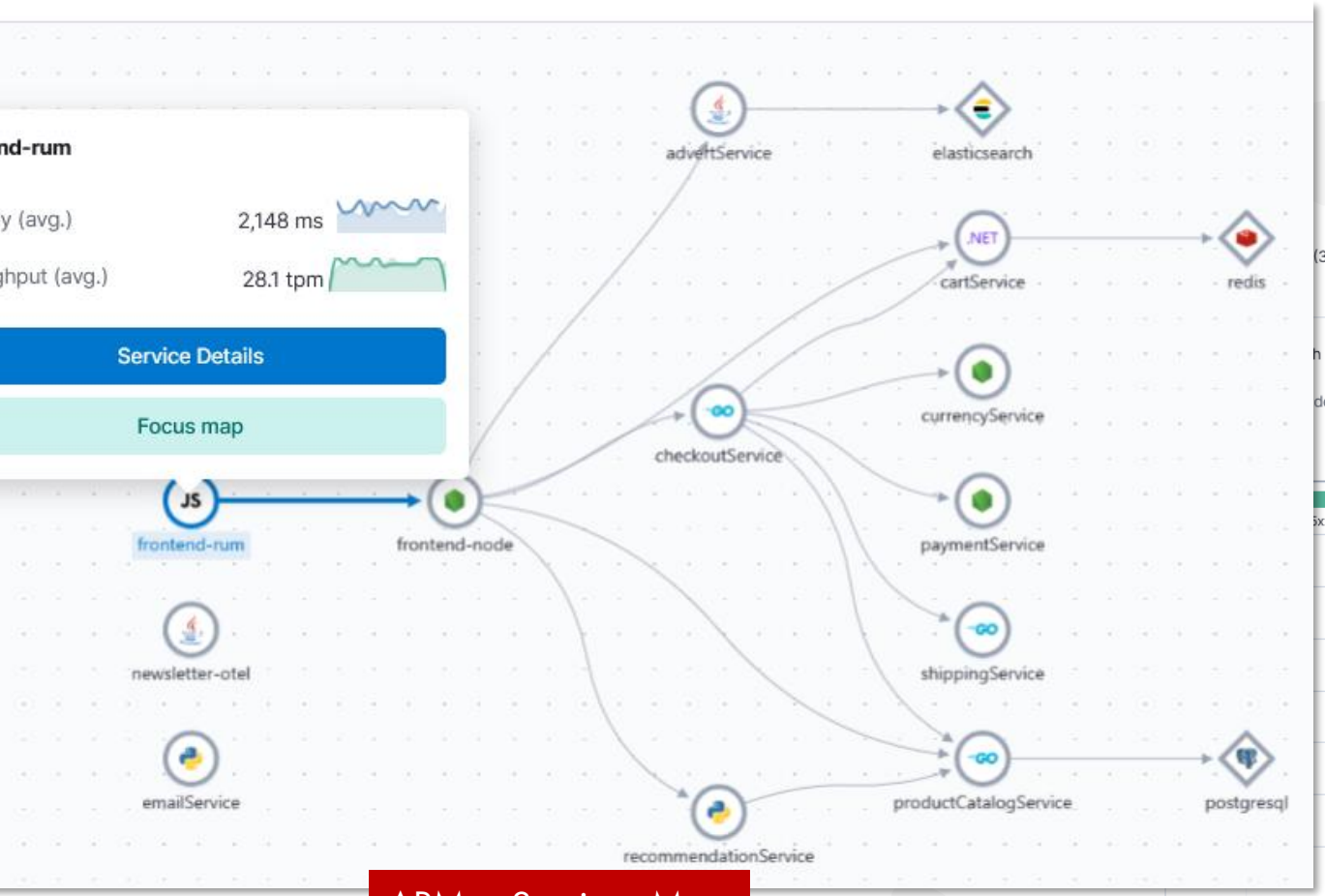
- Act as a **Unified Platform** and single entry point for DevOps, SRE and IT Operation
- NetEye **centralizes** dashboard, alerting and governance
- Elastic unify the data engine: Logs, metrics, traces and synthetics in one place
- **Integrates** and **correlates** data from multiple sources
- **Simplifies** operations and **improves** collaboration



Architecture

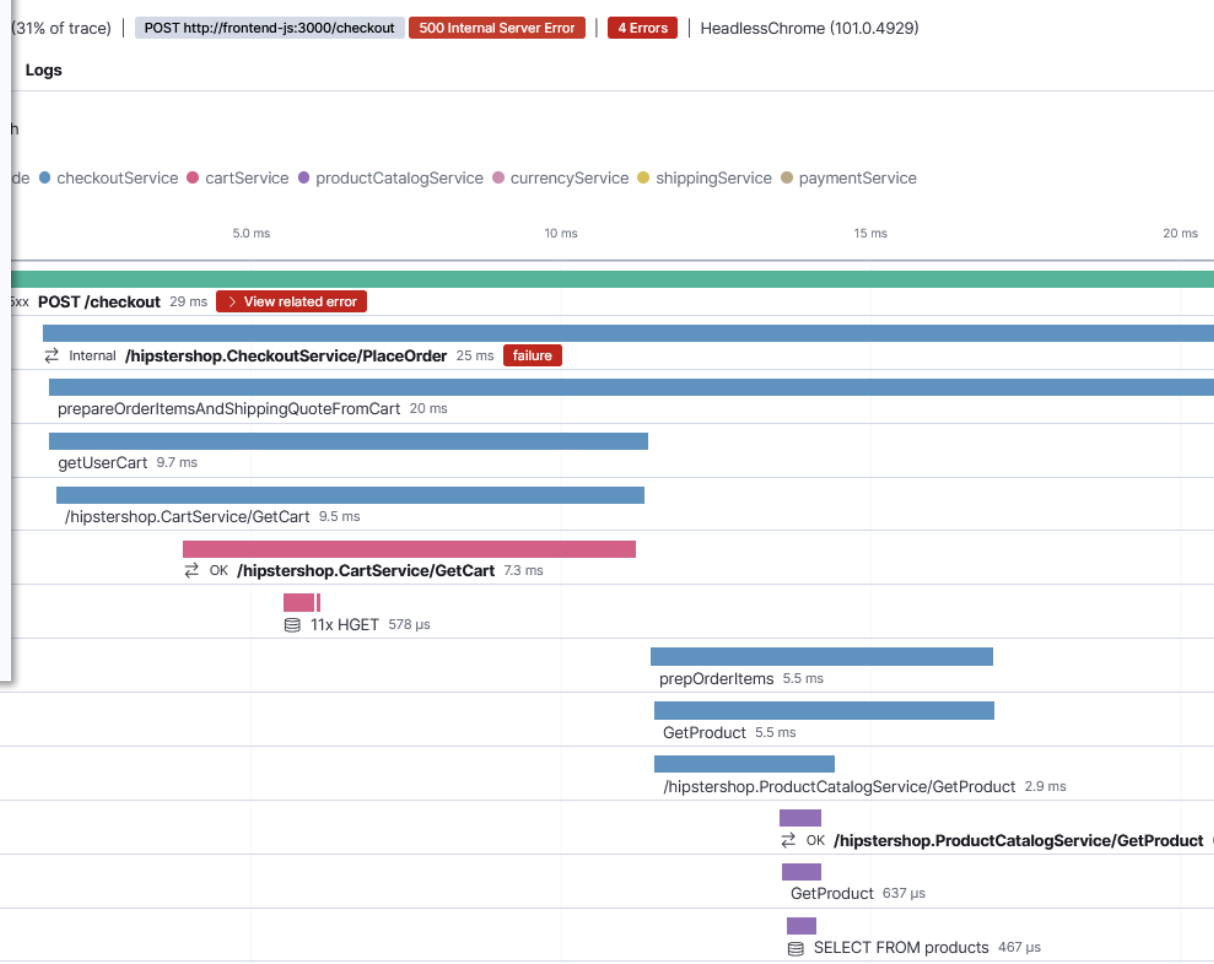


What you see - Elastic



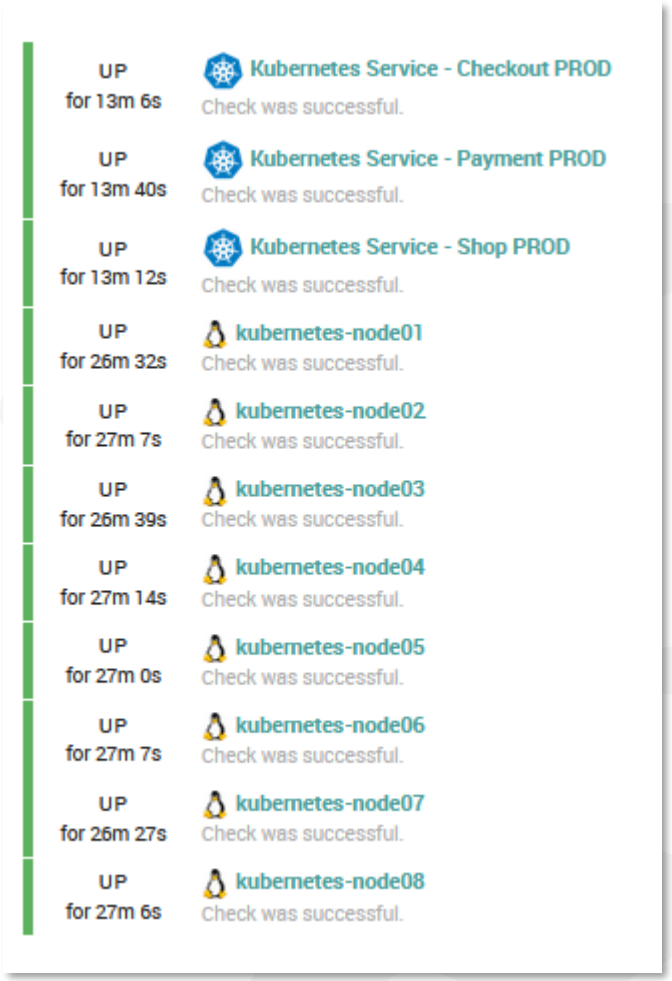
APM – Services Map

APM – Trace Sample

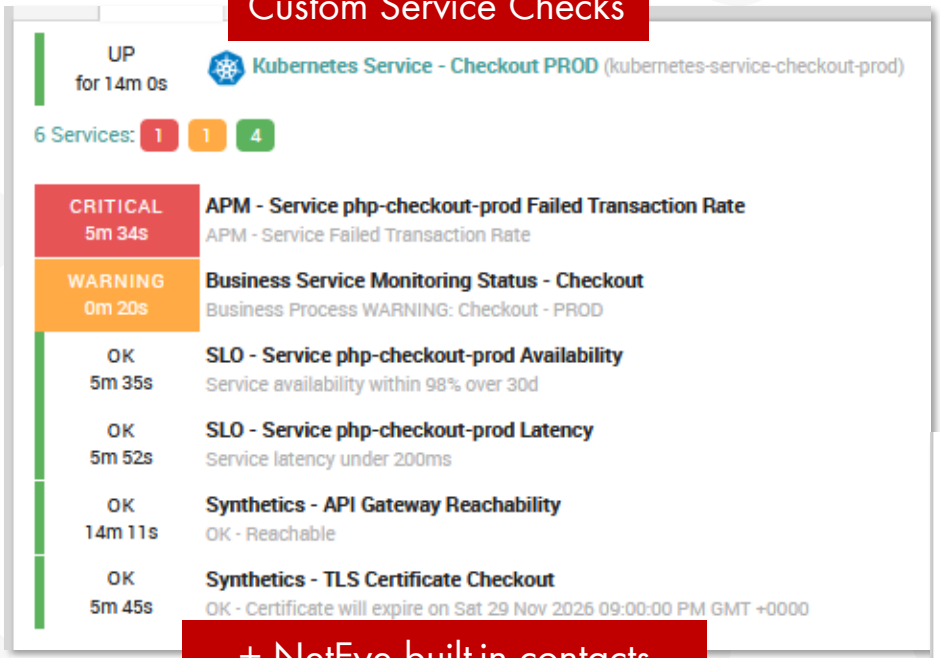


NetEye

What you see - NetEye

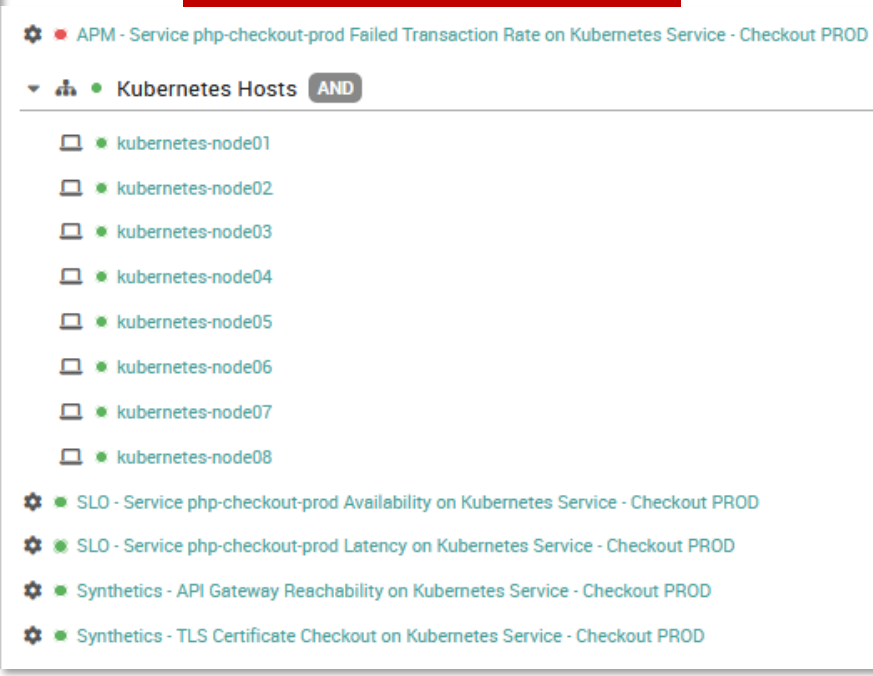


Custom Service Checks



+ NetEye built-in contacts and notifications

Business Service Monitoring (Business Process)



Key Takeaways

- **Full visibility** across all layers
- **Centralized observability** with NetEye & Elastic
- **Faster** root cause analysis
- **Scalable** monitoring architecture
- **User experience** perspective

Thank you