Bridging the Gap

Real-Time Topology as the Connective Tissue Between Platform Engineering and SRE

Conf42

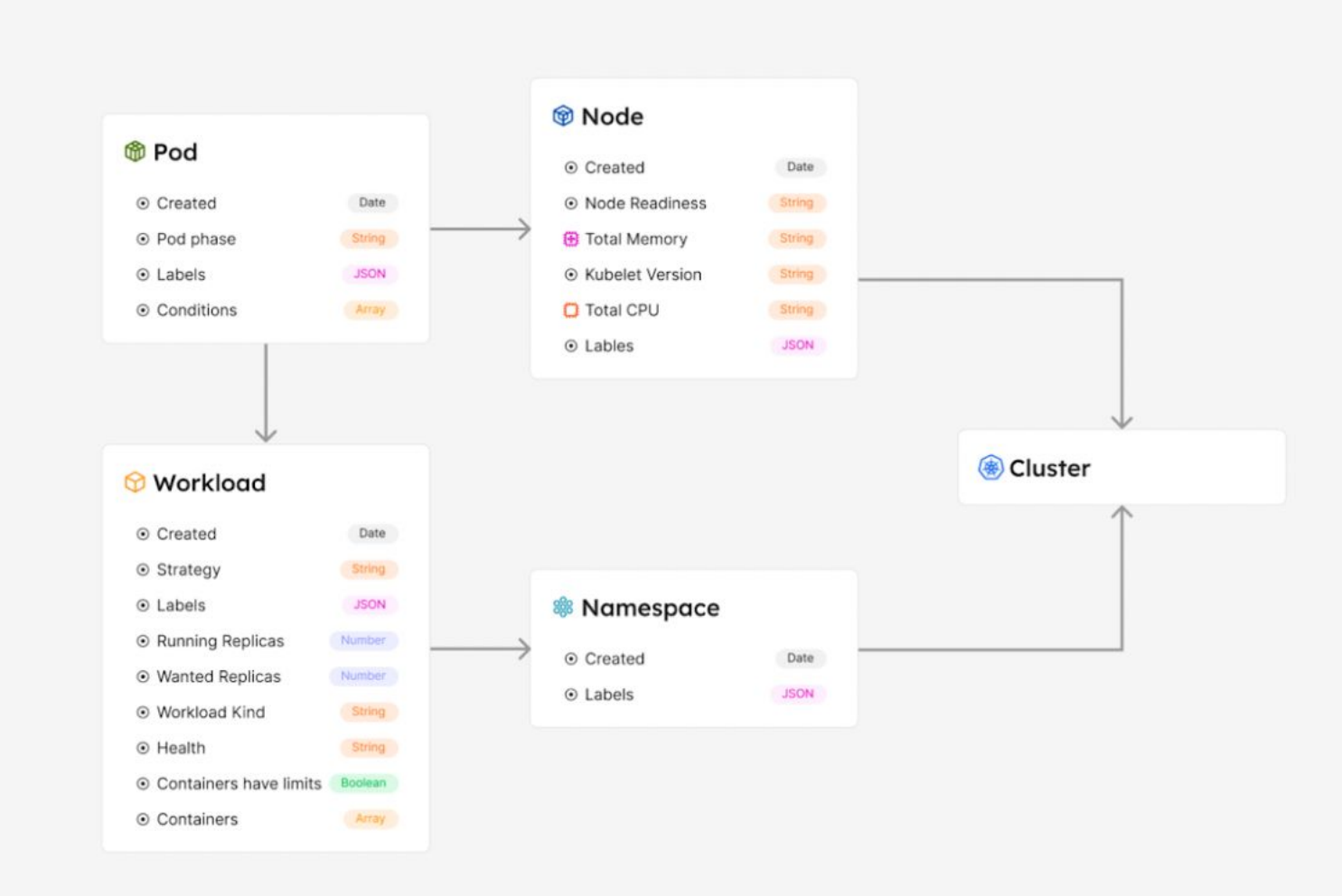
September 5, 2024

Today's Roadmap

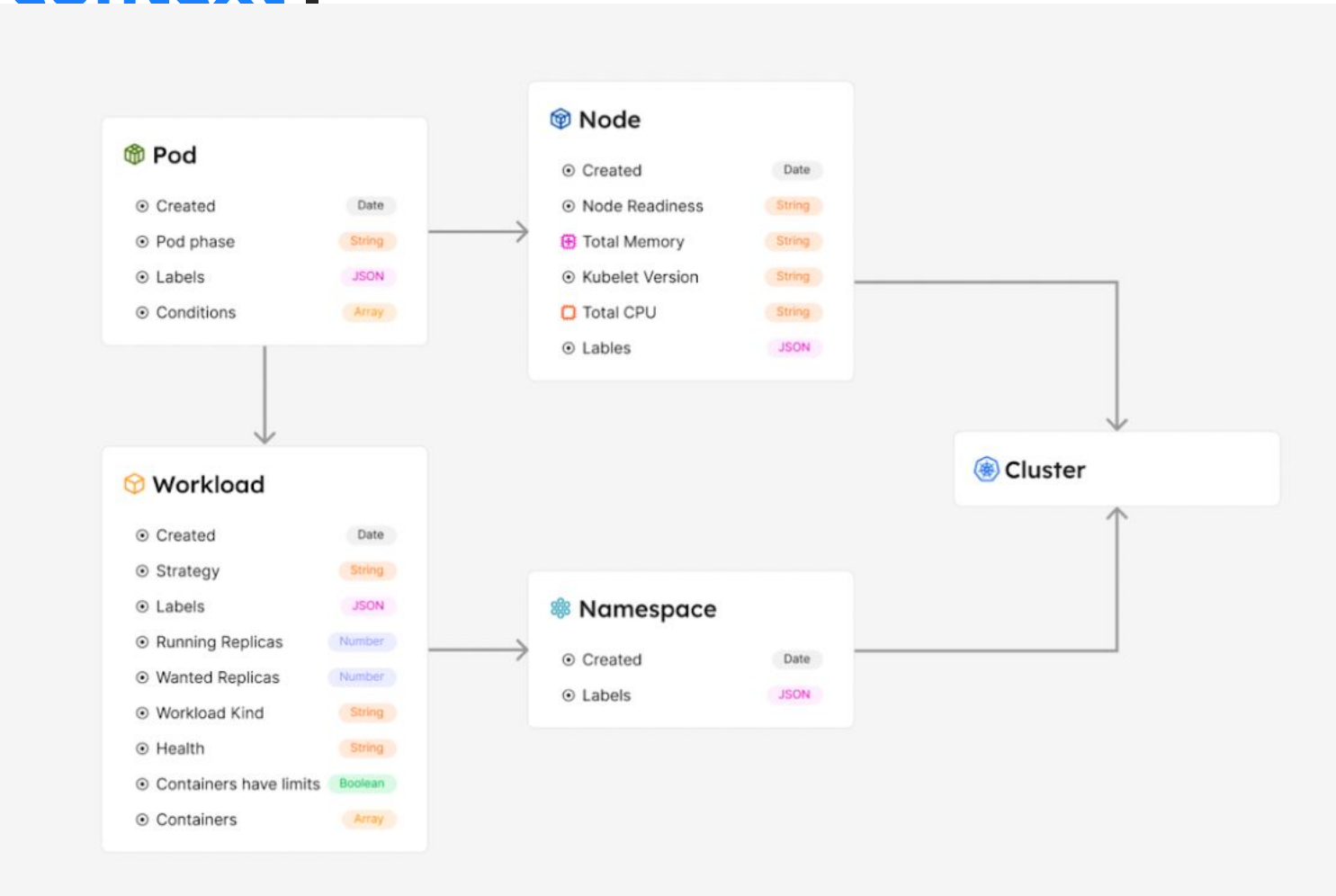
- What's missing from your IDP?
- Lessons from cybersecurity
- Real-time topology as a shared language
- What you can do with a closed loop
- How to get started

SENSER

IDPs are incredible for bridging dev and DevOps ...



...but on their own, they're missing some critical context .



Run-time environment:

- Resources
- APIs
- Third-party exposure/dependencies
- Runtime config (and misconfig)

We’re not the only domain facing this challenge.

Cybersecurity sectors – like **application security** – also had bridge the gap between static analysis and runtime to make “shift left” a reality.

The static “catalog”

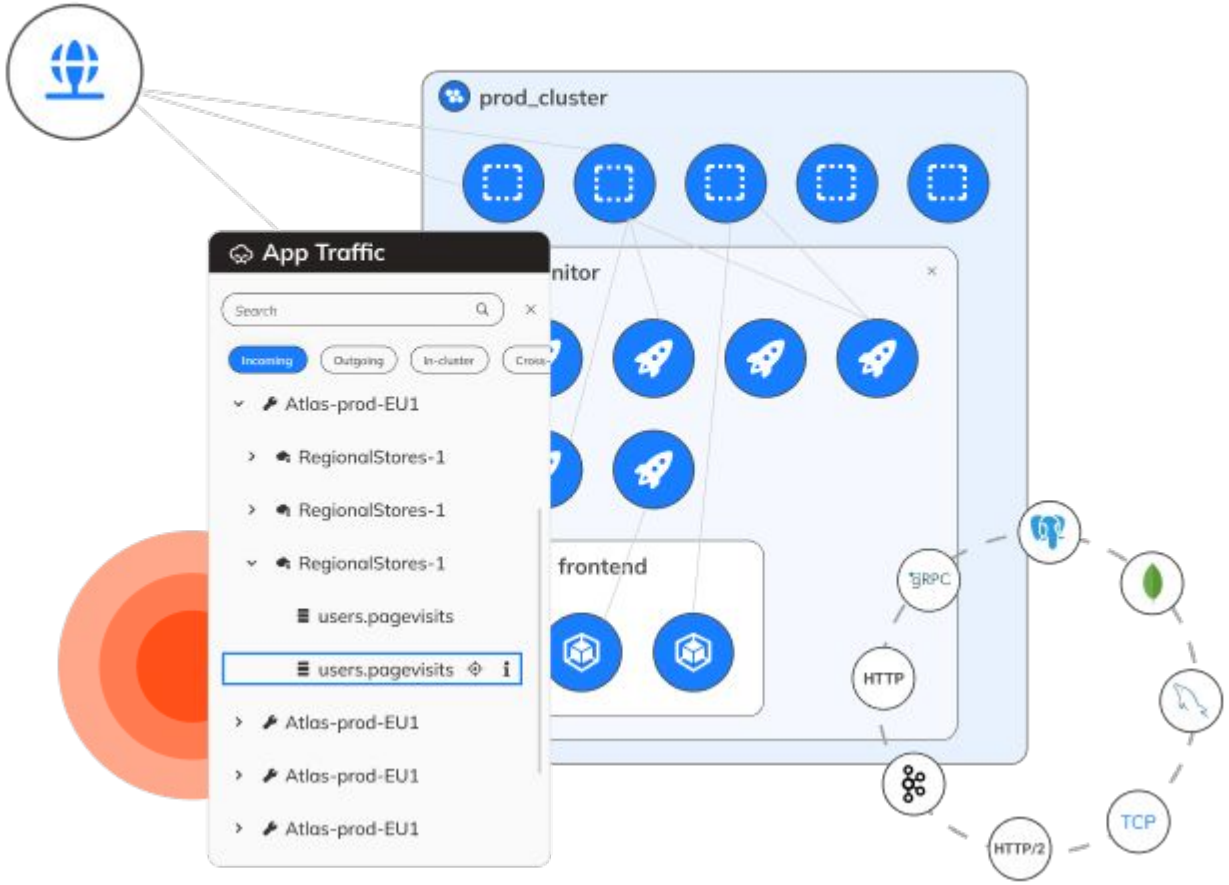
Vulnerabilities
OSS components
Authentication and encryption frameworks
Secrets

- Get an accurate view of the actual attack surface
- Focus on real risks, not just vulnerabilities
- Embed policies and guardrails earlier on in the SDLC

The runtime environment

Deployed?
Internet-exposed?
Behind a WAF?
Runtime misconfigurations?

For platform engineering, this runtime context comes from *real-time topology* .



Benefits

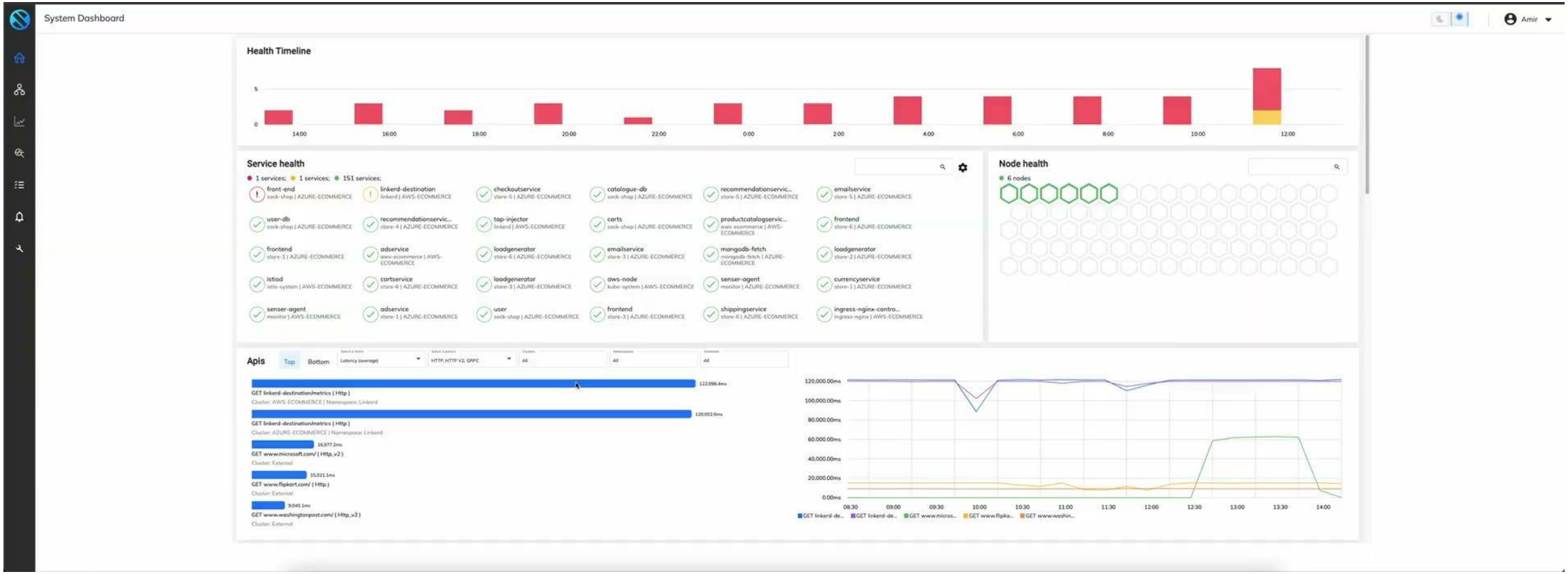
- ✓ Full, closed-loop visibility
- ✓ More efficient troubleshooting
- ✓ Less IDP maintenance required

Why it's hard

Runtime dynamics are much more complex than static dynamics

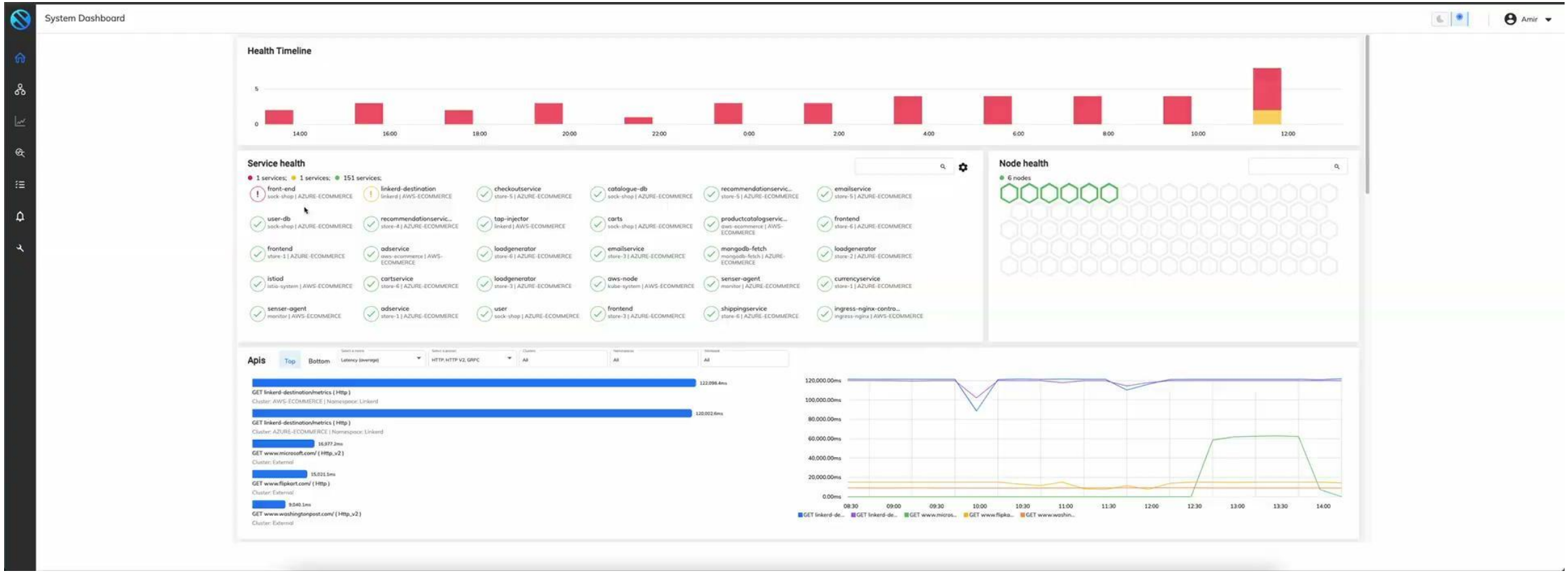
Topology enriches the IDP with runtime context on multiple levels .

Overall system level
Runtime view of your production environment across all applications, infrastructure, networks, and APIs



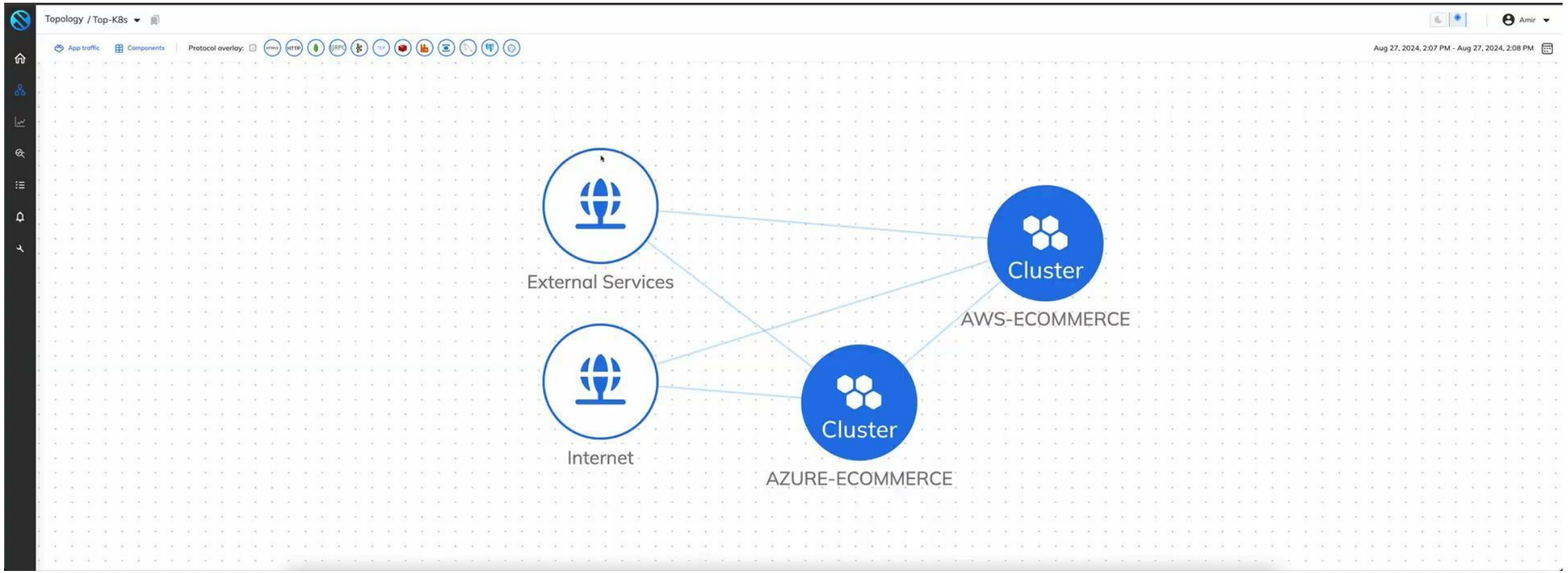
Topology enriches the IDP with runtime context on multiple levels .

Specific services
Deployment status, runtime configuration, and performance of specific services



Topology enriches the IDP with runtime context on multiple levels .

3rd party resources
Availability, latency, status of third-party resources (e.g., public APIs, payment gateways)



Okay, so at a high level...

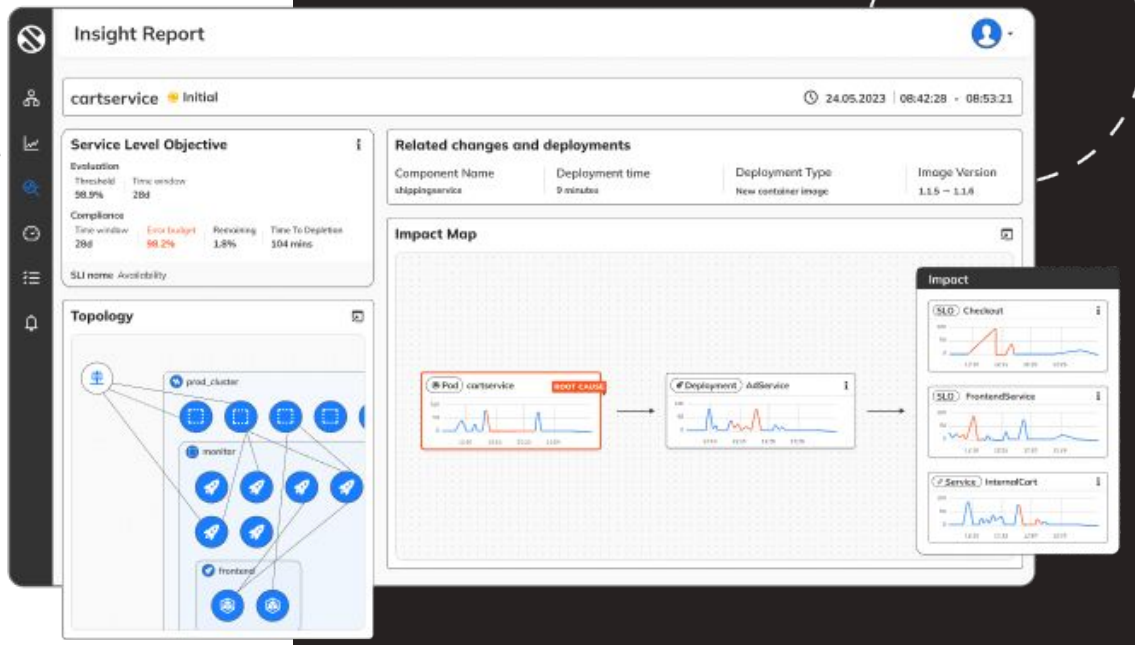
Real-time topology:

- **Enriches the services catalog** of an IDP with an accurate, dynamic representation of the runtime environment (and all associated configurations and dependencies)
- Provides a **common language** for platform engineering and SRE by "closing the loop"
- Furnishes **crucial context** for better decision-making throughout the SDLC

Let's look at three real-world use cases.

Anticipating reliability impact

- Topology can enrich tribal knowledge of past deployments with runtime incidents, affected services, and downstream SLO/SLA impact (i.e., what does this service really “cost” to spin up)?
- Benefits: stronger guardrails, a data-driven checklist for new deployments
- **Real-world example:** a service deployed for high-volume transactions during peak hours often leads to incidents related to memory overload, impacting downstream services like the customer account dashboard and transaction confirmation services



SENSER

Verifying deployments in real-time

- IDPs offer developer guardrails around (e.g., limits on memory usage, CPU, latency, bandwidth) to ensure efficient and reliable deployments
- BUT these are typically set during the development phase
- Topology can provide real-time monitoring of performance to flag deviations, accelerate remediation, and define better guardrails
- **Real-world example:** when a new service begins consuming excessive CPU, the real-time topology can immediately flag this deviation

SENSER

Investigating and responding to issues

- IDPs typically support the notion of incidents, and integrate with observability and incident management platforms
- The service catalog can be a valuable asset for root cause analysis – but it only extends through provisioning and orchestration
- Topology enriches this view with runtime context for full visibility
- **Real-world example:** a team needs to identify whether service degradation in a new deployment is related to issues with a third-party resource

SENSER

Common pitfalls (and how to avoid them)

- **Maintenance debt**. Engineering a closed loop between the service catalog and runtime is the first step – don't overlook the resourcing to keep it properly maintained, audited, and serviced.
- **Incomplete runtime context.** You need full context on your production environment – all resources, APIs, third-party services. (Technologies like eBPF can help.)
- **Static snapshots.** To be useful, runtime topology needs to be continuously, automatically discovered and updated.

Tips for getting started

- **Build your topology**. Observability tools can help, but often require extensive configuration. Some platforms offer auto-discovery and mapping of your environment, often with the help of technology like eBPF.
- **Integrate your topology with your IDP.** IDPs offer integration with observability and incident response platforms.
- **Prioritize use cases.** Identify “quick wins” based on your organization’s challenges and needs (e.g., verifying deployments in real-time).
- **Scale and expand.** Build on early wins with additional opportunities to weave runtime context into your IDP and throughout the SDLC.

SENSER

Thank you!