TECHX

Asia's Largest

AI & Cloud Conference 2024

15 - 16, November 2024

Chennai Trade Center, Chennai

Topic

Unraveling The Microservices Maze: Advanced Distributed Tracing In Kubernetes

Who I am

Vijayperumal

Technical Architect @ CES

SCAN ME

Episode #01 - Microservices

1) API Gateway

The gateway provides a unified entry point for client applications. It handles routing, filtering, and load balancing.

2) Service Registry

The service registry contains the details of all the services. The gateway discovers the service using the registry. For example, Consul, Eureka, Zookeeper, etc.

3) Service Layer

Each microservices serves a specific business function and can run on multiple instances. These services can be built using frameworks like Spring Boot, NestJS, etc.

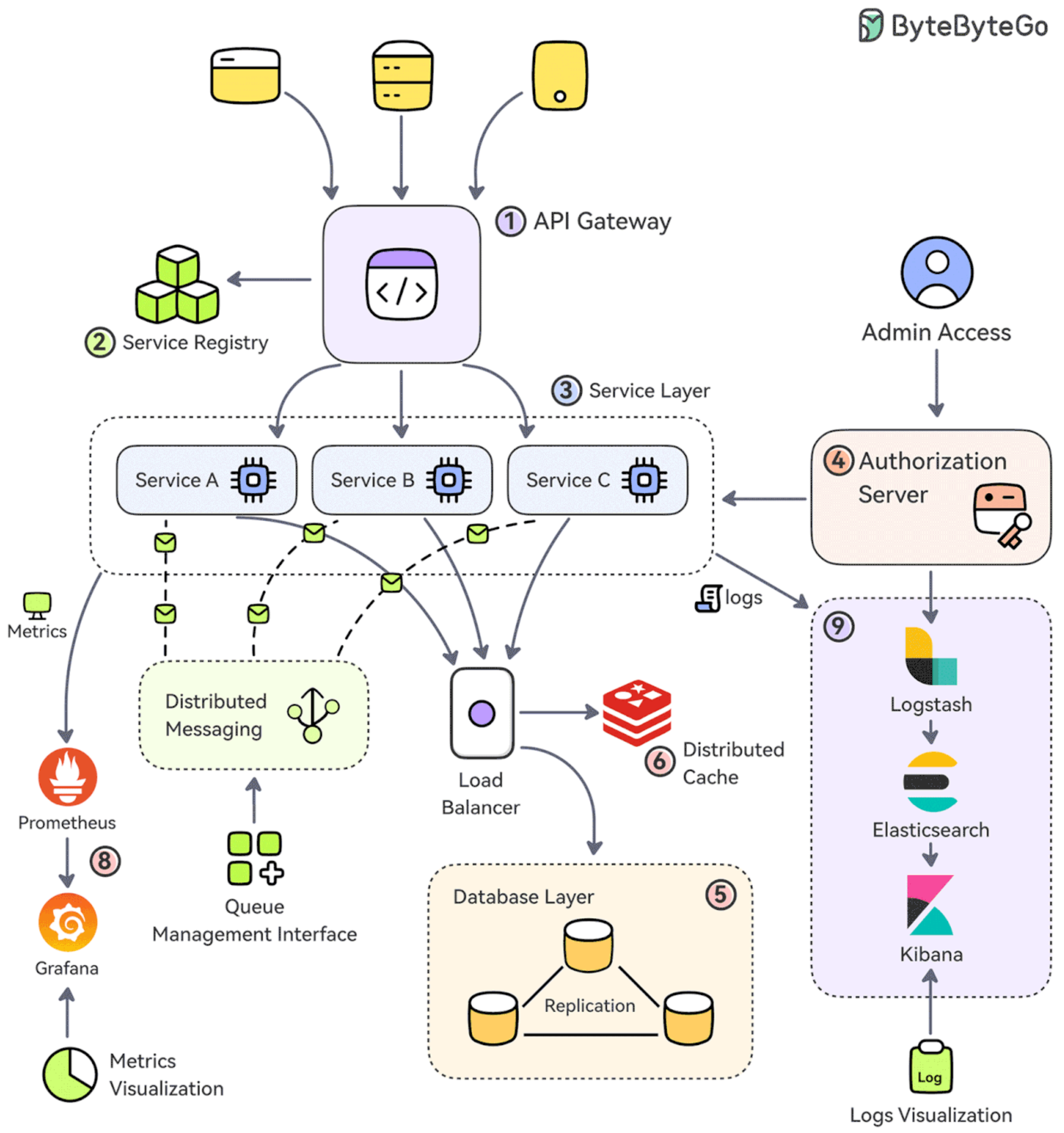
4) Authorization Server

Used to secure the microservices and manage identity and access control. Tools like Keycloak, Azure AD, and Okta can help over here.

5) Data Storage

Databases like PostgreSQL and MySQL can store application data generated by the services.

9 Essential Components of Production Microservice App



5) Distributed Caching

Caching is a great approach for boosting the application performance. Options include caching solutions like Redis, Couchbase, Memcached, etc.

6) Async Microservices Communication

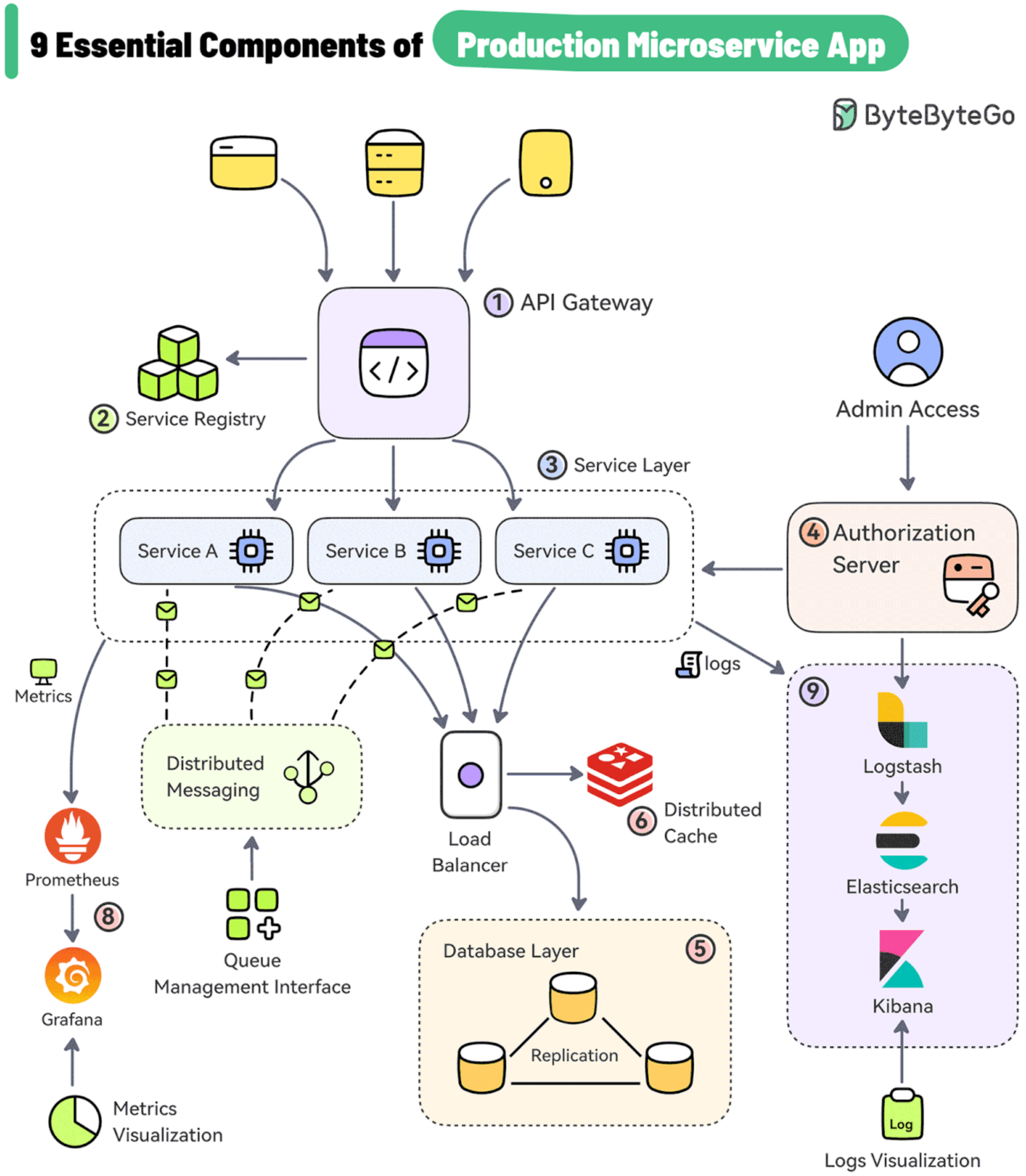
Use platforms such as Kafka and RabbitMQ to support async communication between microservices.

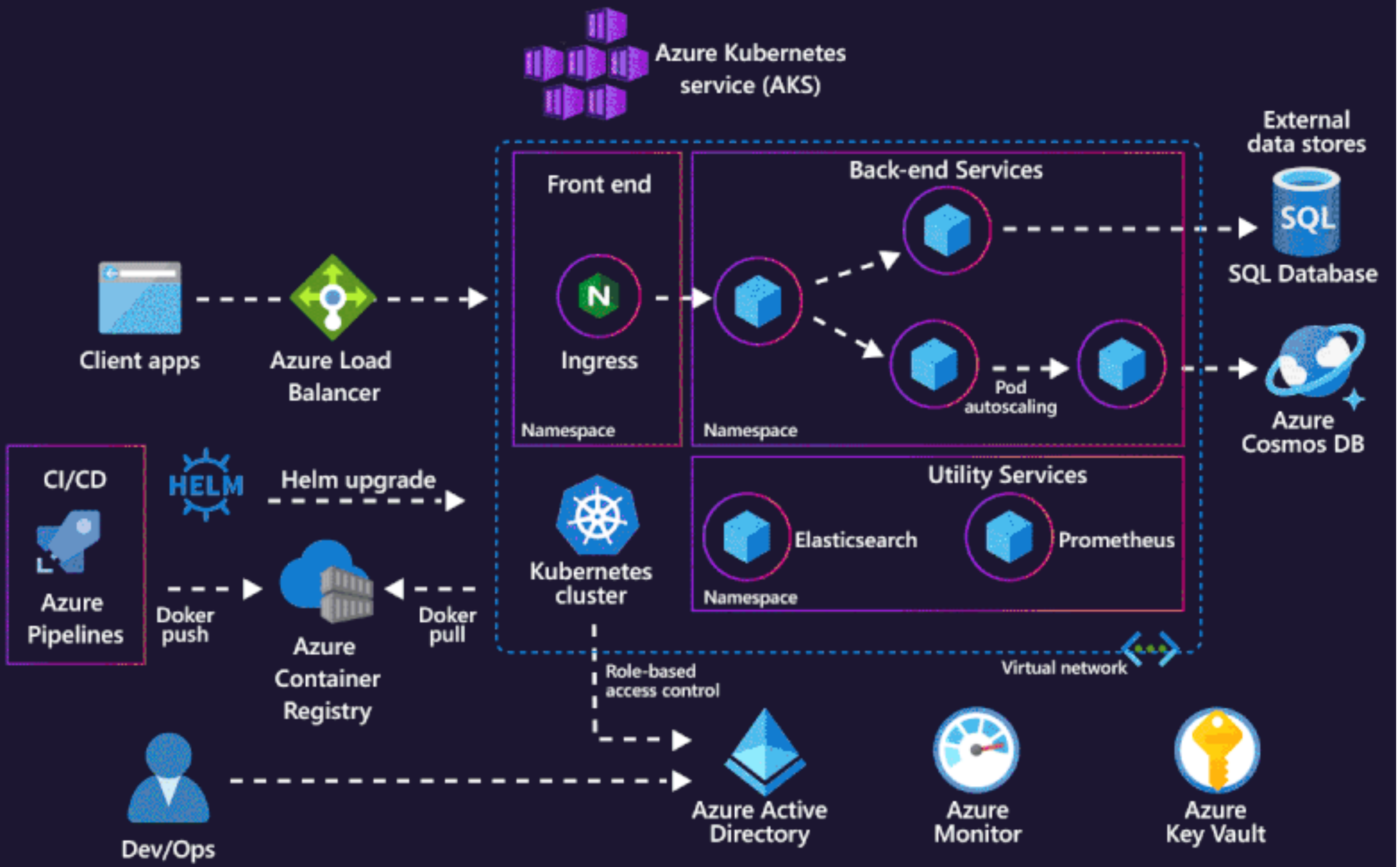
7) Metrics Visualization

Microservices can be configured to publish metrics to Prometheus and tools like Grafana can help visualize the metrics.

8) Log Aggregation and Visualization

Logs generated by the services are aggregated using Logstash, stored in Elasticsearch, and visualized with Kibana.




Azure Kubernetes service (AKS)
External data stores
Front end
Back-end Services
SQL
SQL Database
Client apps
Azure Load Balancer
Ingress
Namespace
Namespace
Pod autoscaling
Azure Cosmos DB
CI/CD
HELM
Helm upgrade
Utility Services
Elasticsearch
Prometheus
Azure Pipelines
Kubernetes cluster
Doker push
Doker pull
Namespace
Azure Container Registry
Virtual network
Role-based access control
Azure Active Directory
Azure Monitor
Azure Key Vault
Dev/Ops

Microservice Challenges

Using Kubernetes

Complexity of Orchestration

Managing multiple services, pods, and replicas.

Observability

Difficulties in monitoring distributed systems.

Networking and Service Discovery

Inter-service communication challenges

Resource Management

Efficiently allocating CPU and memory.

Security

Securing microservices and APIs in a dynamic environment

Microservice Challenges

Using Kubernetes

Complexity of Orchestration

Managing multiple services, pods, and replicas.

Observability

Difficulties in monitoring distributed systems.

Networking and Service Discovery

Inter-service communication challenges

Resource Management

Efficiently allocating CPU and memory.

Security

Securing microservices and APIs in a dynamic environment

Microservice Challenges

Using Kubernetes

Complexity of Orchestration

Managing multiple services, pods, and replicas.

Observability

Difficulties in monitoring distributed systems.

Networking and Service Discovery

Inter-service communication challenges

Resource Management

Efficiently allocating CPU and memory.

Security

Securing microservices and APIs in a dynamic environment

Microservice Challenges

Using Kubernetes

Complexity of Orchestration

Managing multiple services, pods, and replicas.

Observability

Difficulties in monitoring distributed systems.

Networking and Service Discovery

Inter-service communication challenges

Resource Management

Efficiently allocating CPU and memory.

Security

Securing microservices and APIs in a dynamic environment

Microservice Challenges

Using Kubernetes

Complexity of Orchestration

Managing multiple services, pods, and replicas.

Observability

Difficulties in monitoring distributed systems.

Networking and Service Discovery

Inter-service communication challenges

Resource Management

Efficiently allocating CPU and memory.

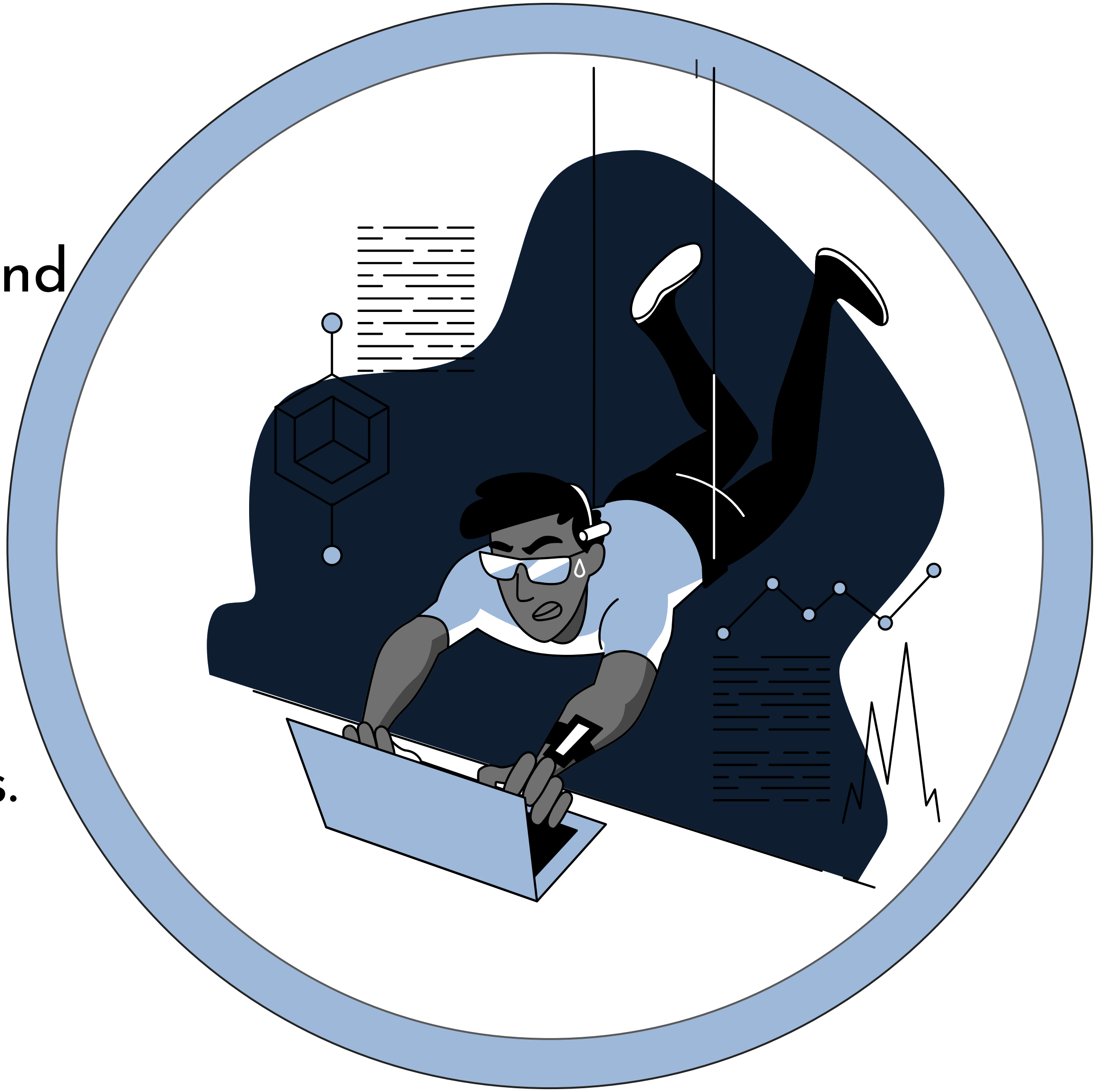
Security

Securing microservices and APIs in a dynamic environment

Episode #02 - Observability

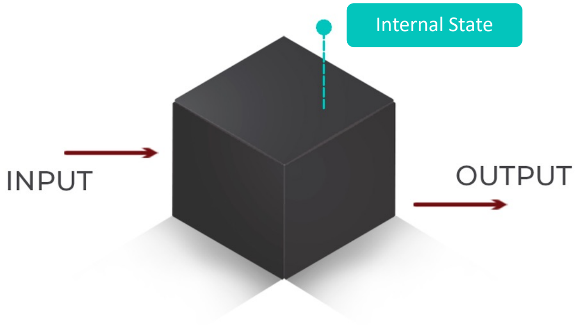
Observability

Observability is the ability to understand
The internal state of a system by
examining its outputs.
In the context of software, this means
being able to understand the internal
state of a system by
examining its telemetry data,
which includes traces, metrics, and logs.



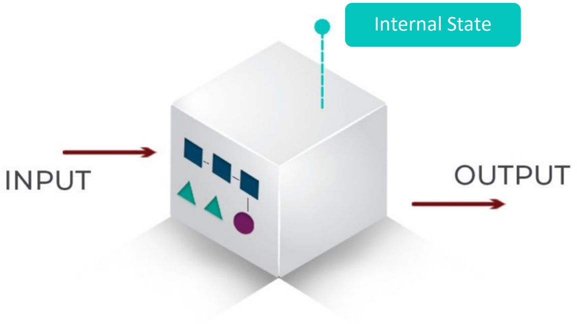
BlackBox

We can't understand a complex system if it's a black box.



WhiteBox

The only way to light up those black boxes is with high-quality telemetry: distributed traces, metrics, logs, and more.



Outputs -> Telemetry Data

To make a system observable, it must be instrumented. That is, the code must emit signals or telemetry data such as: traces, metrics, or logs (MELT for short). The instrumented data must then be sent to an **observability backend**.

The three core elements of data observability are:

Logs

Metrics

Tracing

Observability Benefits

Higher Visibility

Speed up Troubleshooting

Team Productivity

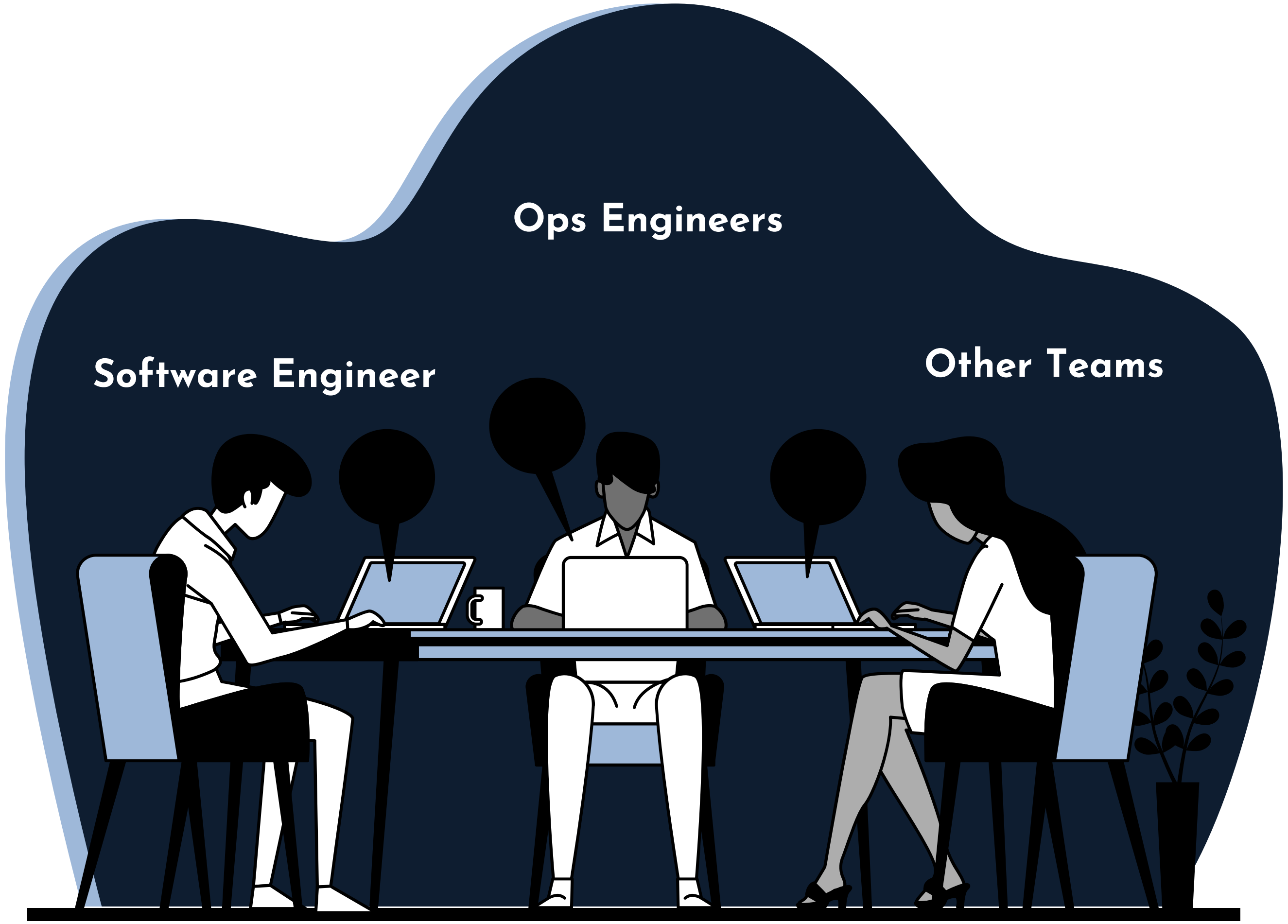
Finding out unknown issues

Help to improve end user experience

Help to reduce costs

People benefiting from observability

Many different teams can use observability to understand the behavior of complex digital systems and turn data into tailored insights.



Monitoring X Observability

Based on predefined sets of metrics or logics.

What? + When?

E.g.

- What to expect from the software?
- When did the software present an unexpected behaviour?

Monitoring presents the software behaviour

Based on exploring properties and patterns not defined in advance.

Why?

E.g.

- Why is the software behaving unexpectedly?

Observability explains the software behavior

Episode #03 - Open Telemetry

What is Open Telemetry

OpenTelemetry, also known as OTel, is a vendor-neutral open source Observability framework

OpenTelemetry is focused on the generation, collection, management, and export of telemetry data such as traces, metrics, and logs

Two key principles:

1. You own the data that you generate.
   There's no vendor lock-in
2. You only have to learn a single set of
   APIs and conventions

Sponsored by the
Support for different programming languages

What Type of Telemetry Data Does OpenTelemetry Handle?

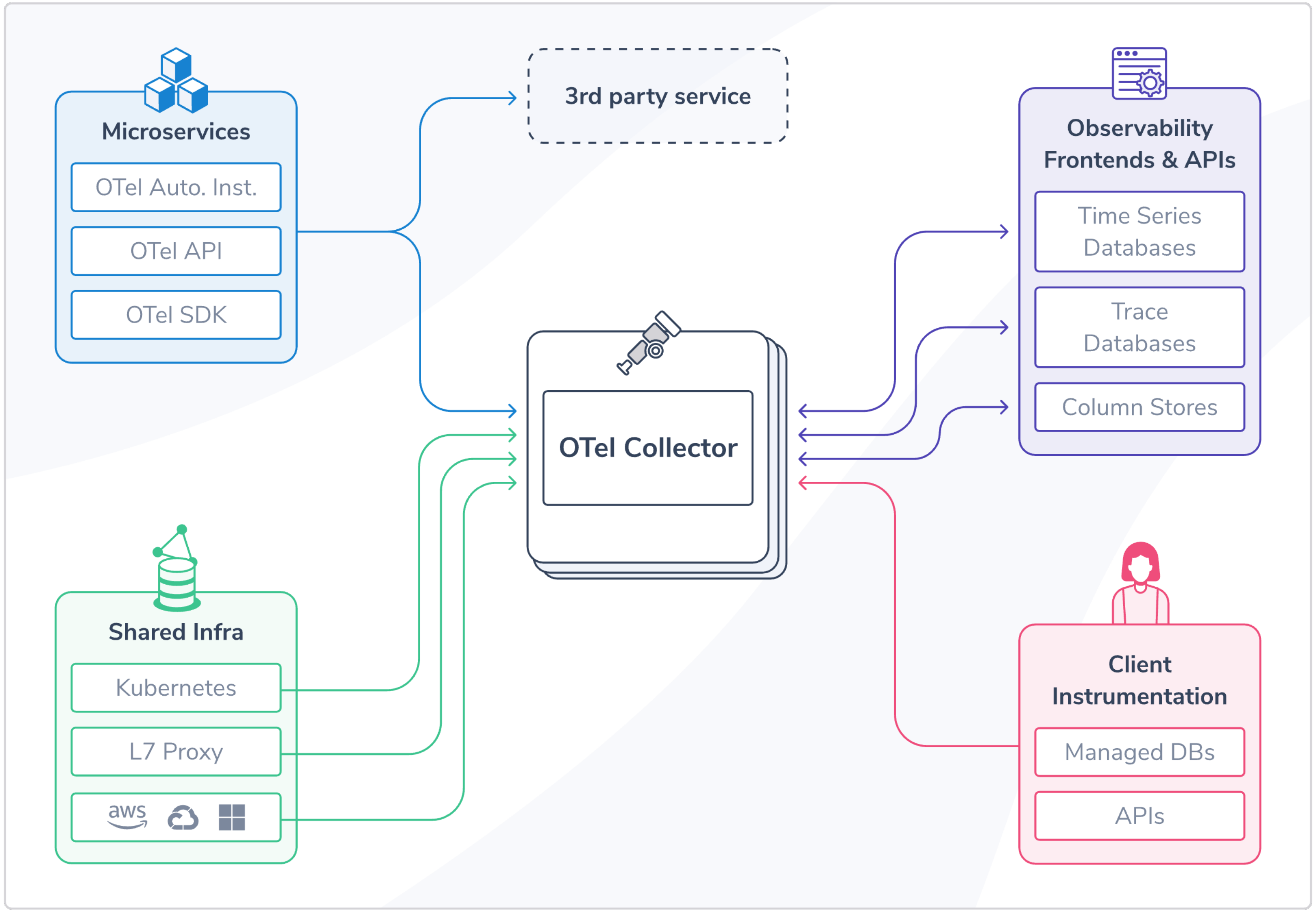
Open Telemetry handles three primary types of telemetry data:

Traces allow developers to track the journey of a request through various services, helping to identify bottlenecks and understand the flow of requests within a system.

Metrics provide quantitative information about the operation of applications and infrastructure, such as response times, memory usage, and request counts, enabling performance monitoring and trend analysis.

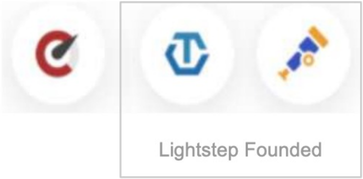
Logs offer qualitative insights through event records, detailing what happened in the system at a specific point in time. Together, these data types offer a holistic view of system performance and behavior, aiding in debugging, monitoring, and optimizing applications.

OpenTelemetry Architecture and Components

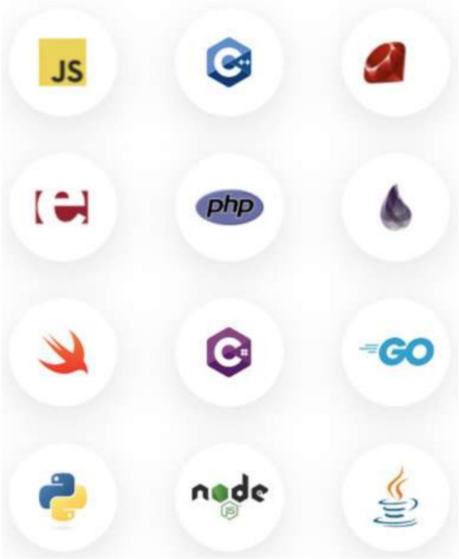


Integrations

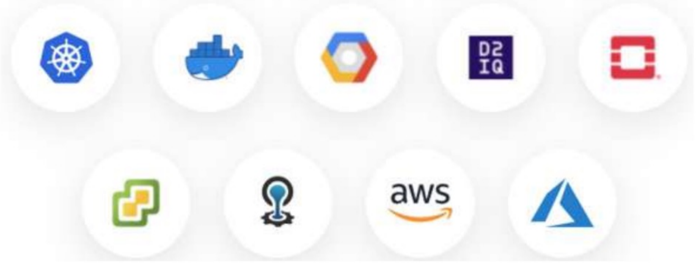
Standards


Lightstep Founded

Languages



Containers, Platforms and Clouds



Tracers

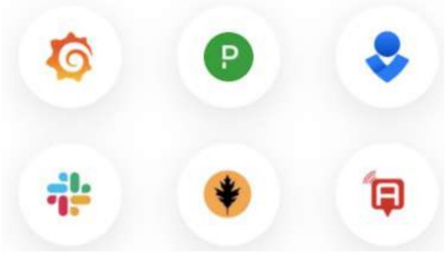
Data Streaming and Storage

kafka
MySQL
mongoDB

Service Meshes / Proxies

Deployment Automation (CI/CD)

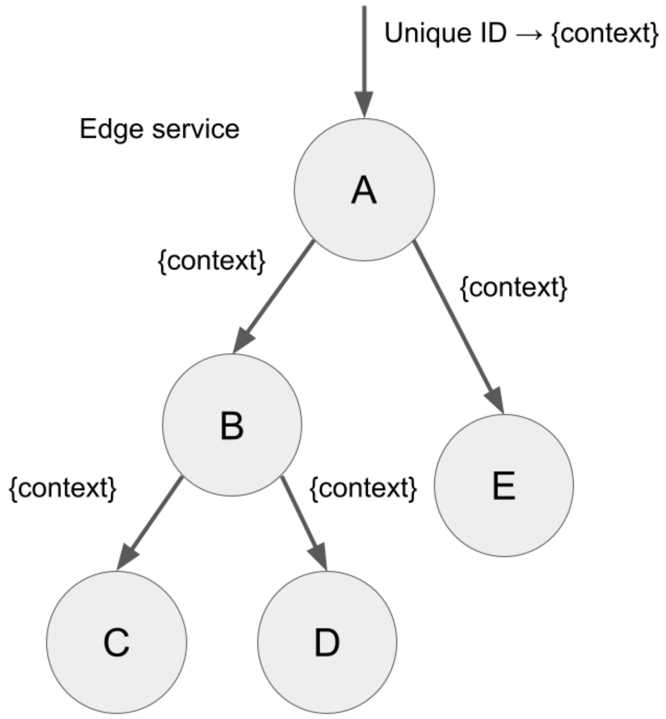
Alerting and Tools

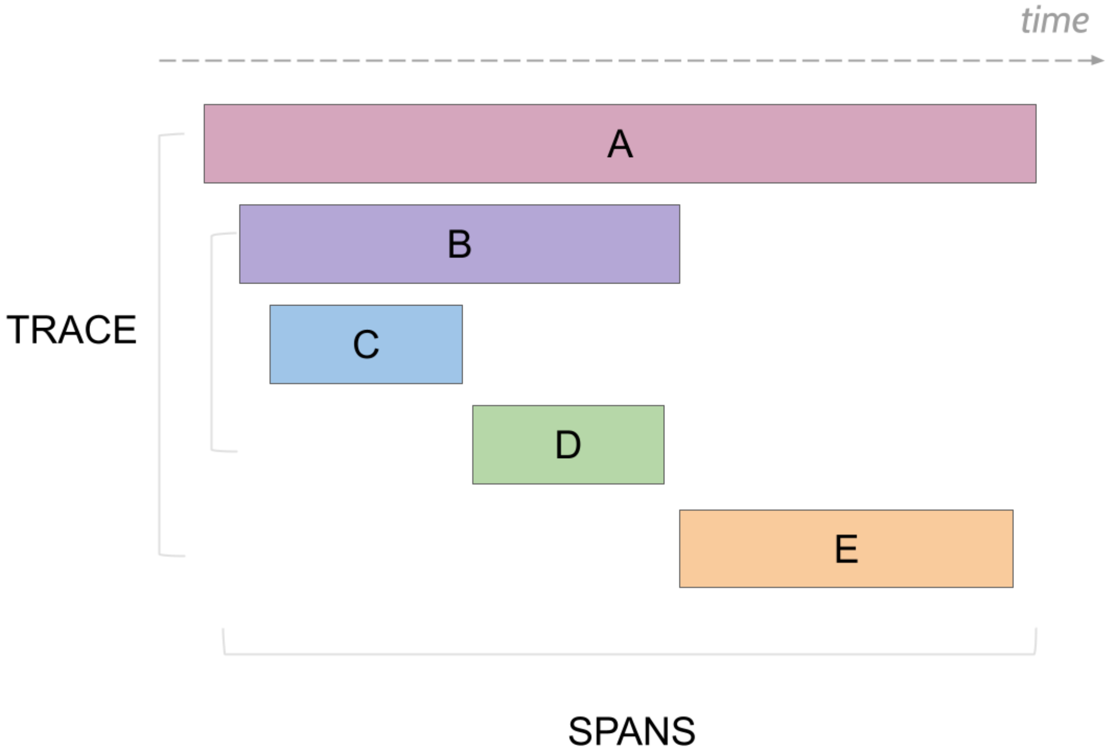


Terminology

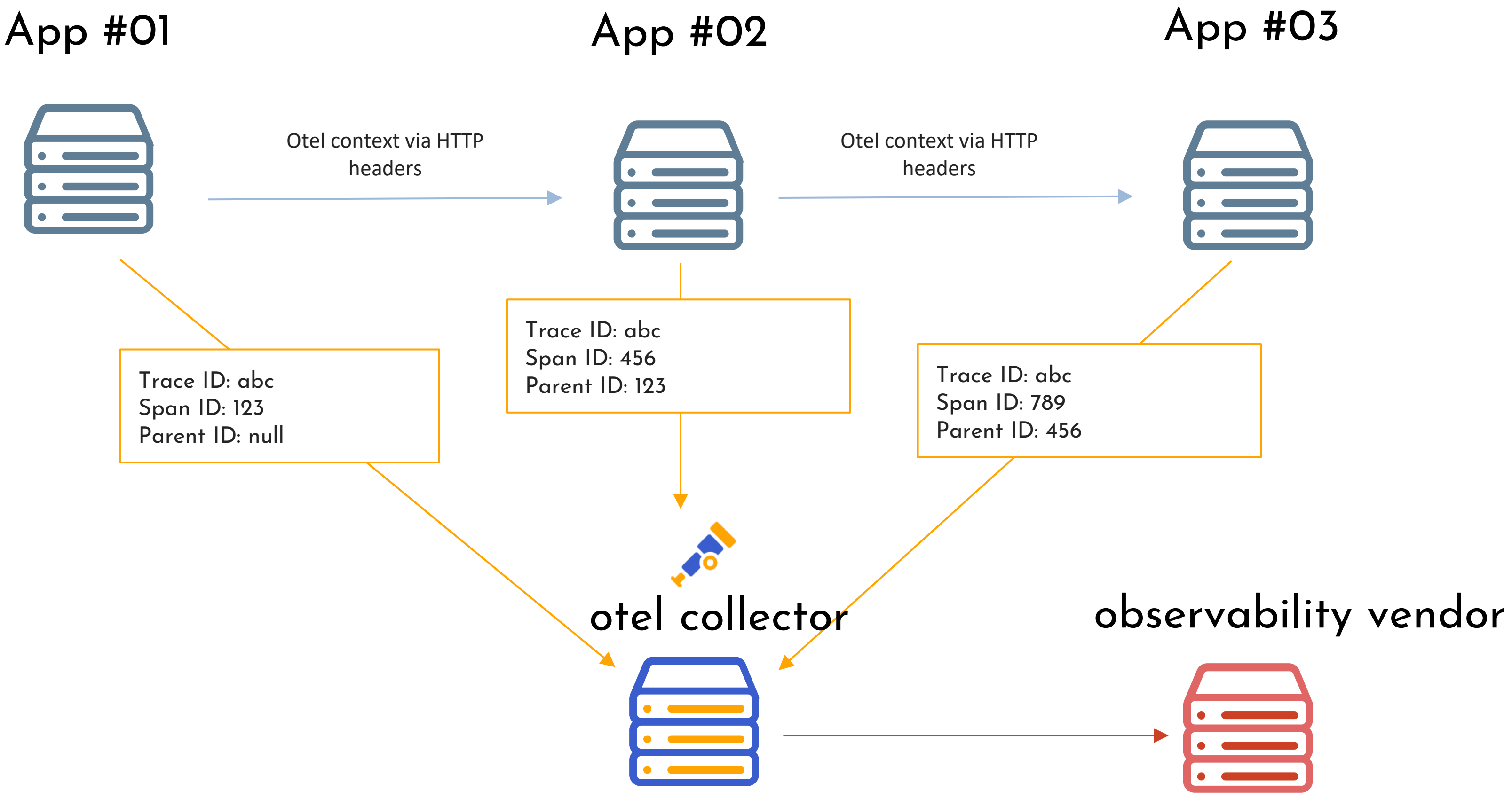
A **span** represents a logical unit of work in Jaeger that has an operation name, the start time of the operation, and the duration. Spans may be nested and ordered to model causal relationships.

A trace is a data/execution path through the system, and can be thought of as a directed acyclic graph of spans.



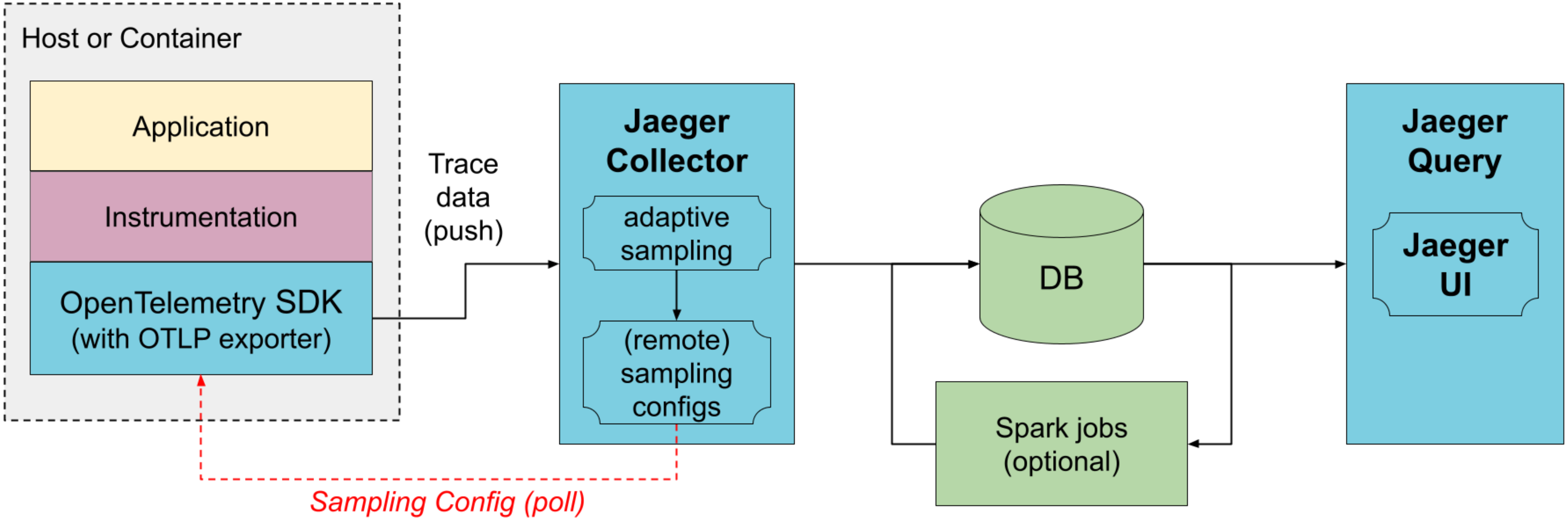


Distributed Tracing



Episode #04 - Jaeger UI

Architecture



Episode #07 - Best Practices

Best Practices for Distributed Tracing in Production

- ✓ Optimize Sampling Strategies
- ✓ Tag Meaningfully and Consistently
- ✓ Reduce Tracing Overhead
- ✓ Monitor Trace Health and Coverage
- ✓ Leverage Dashboards and Alerts
- ✓ Integrate Traces with Logs and Metrics
- ✓ Train and Align Teams
- ✓ Scale Efficiently
- ✓ Focus on Actionable Insights
- ✓ Choose the Right Tools and Ecosystem

Episode #06 - The Future

The barriers to adopting Distributed Tracing – and how AI is going to remove them.

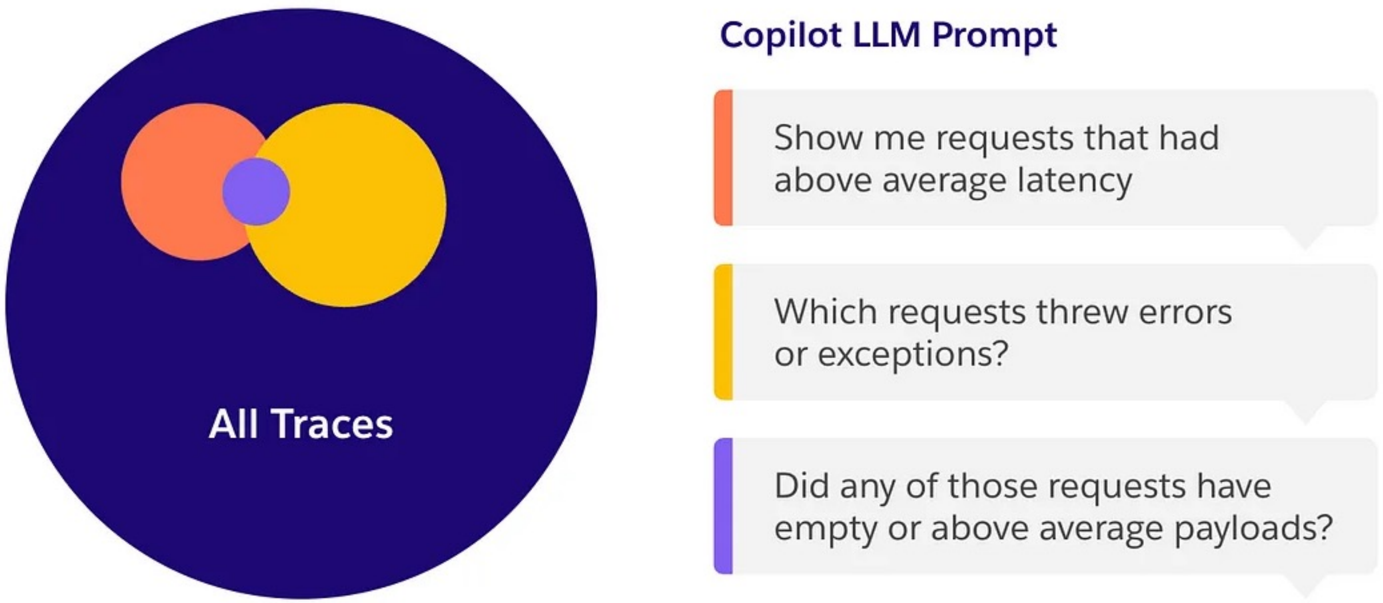
The Configuration Issue

The Sampling Issue

The Trace Enrichment Issue

What you get

You get a full eight hours of sleep.
Your company closes the deal.
Everybody is happy.



Copilot LLM Prompt

Show me requests that had above average latency

Which requests threw errors or exceptions?

Did any of those requests have empty or above average payloads?

Episode #06 - The Review

The End - See you all TechXConf 2024