ORACLE

# Microservices Observability Patterns

**Paulo Alberto Simões**

**Developer Evangelist**, Data-Driven Microservices with Converged Database

# What are Microservices?

*If every service has to be updated at the same time it's not loosely coupled*

"loosely coupled service-oriented architecture with bounded contexts."

*If you know too much about the surrounding services you don't have a bounded context*

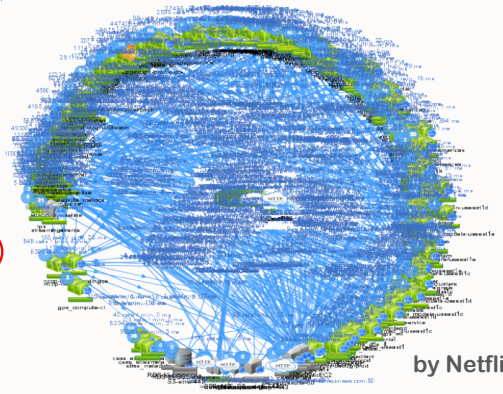
***Adrian Cockroft, while at Netflix***

# Cloud Native Design Principles

CLOUD NATIVE COMPUTING FOUNDATION

Containerized

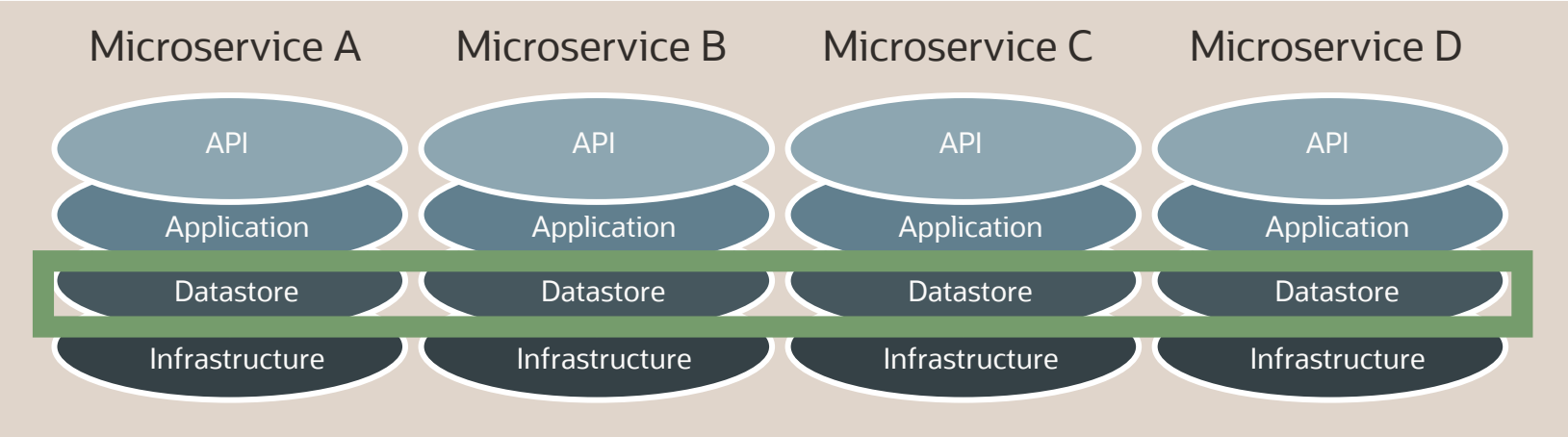
Open Source Software




Microservices Oriented

Dynamically orchestrated

# Microservices Forces Move To Distributed Computing

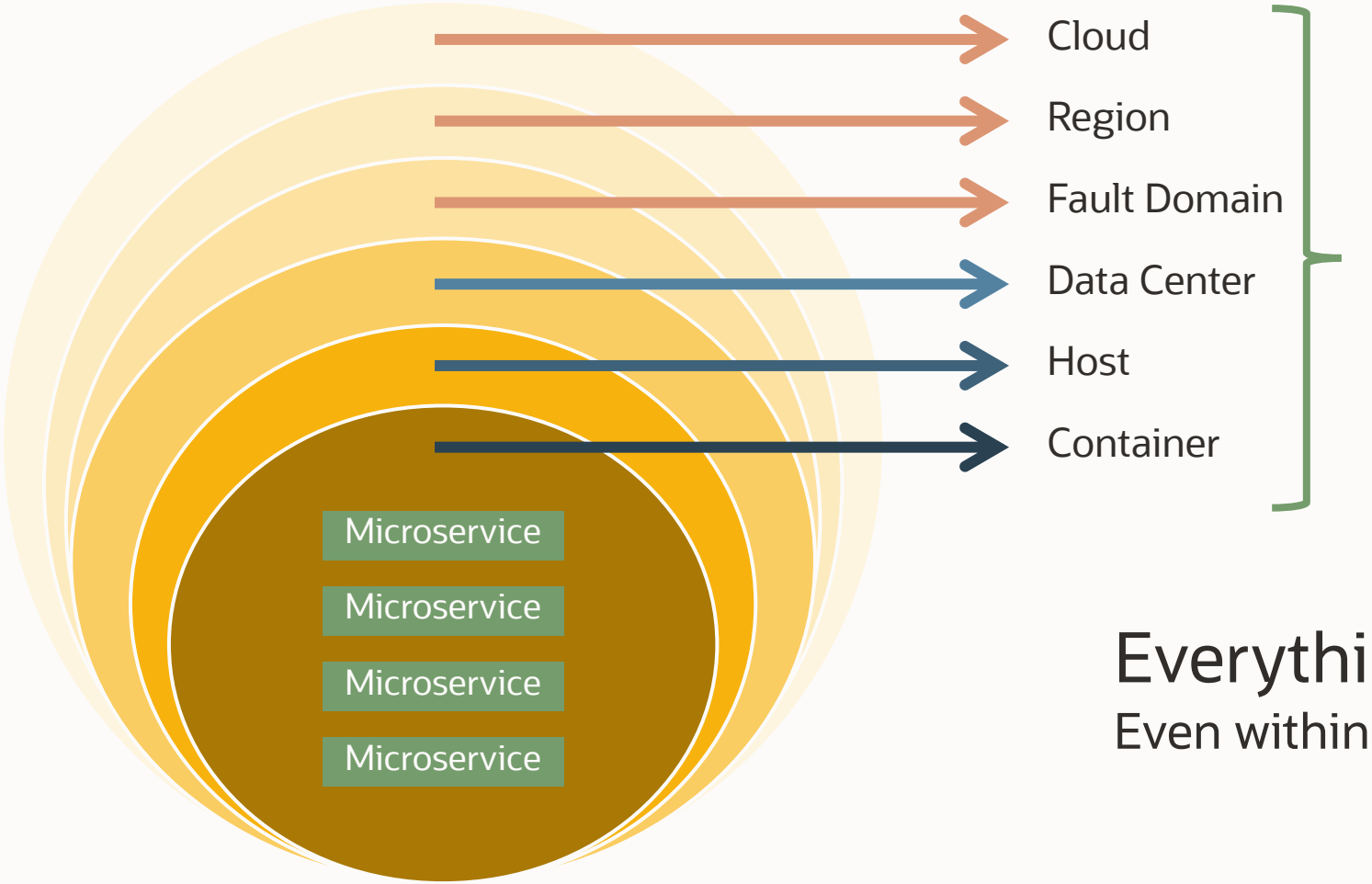



- Distributed computing is a natural consequence of microservices because each microservice has its own datastore
- Sharing datastores across microservices introduces coupling – very bad!
- There will always be latency between microservices
- Latency = eventual consistency

- All data exchange between microservices must be through API layer or messaging – no accessing datastores cross-microservices
- Must implement high-speed messaging between microservices. REST + HTTP probably isn't fast enough
- May end up duplicating data across datastores – e.g. a customer's profile

# Distributed Computing forces Everything distributed

Run your applications across multiple data centers, fault domains, regions, etc




Hundreds of milliseconds of latency

Everything is now distributed

Even within the same data center

CLOUD NATIVE COMPUTING FOUNDATION

# Cloud Native Trail Map

CLOUD NATIVE COMPUTING FOUNDATION

## CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape l.cncf.io has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

## HELP ALONG THE WAY

### A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

cncf.io/training

### B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

### C. Join CNCF's End User

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

### 2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing

### 3. ORCHESTRATION & APPLICATION DEFINITION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: cncf.io/ck
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application

kubernetes CNCF Graduated

HELM CNCF Incubating

### 4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

Prometheus CNCF Graduated

fluentd CNCF Graduated

OPENTRACING CNCF Incubating

JAEGER CNCF Incubating

### 5. SERVICE PROXY, DISCOVERY, & MESH

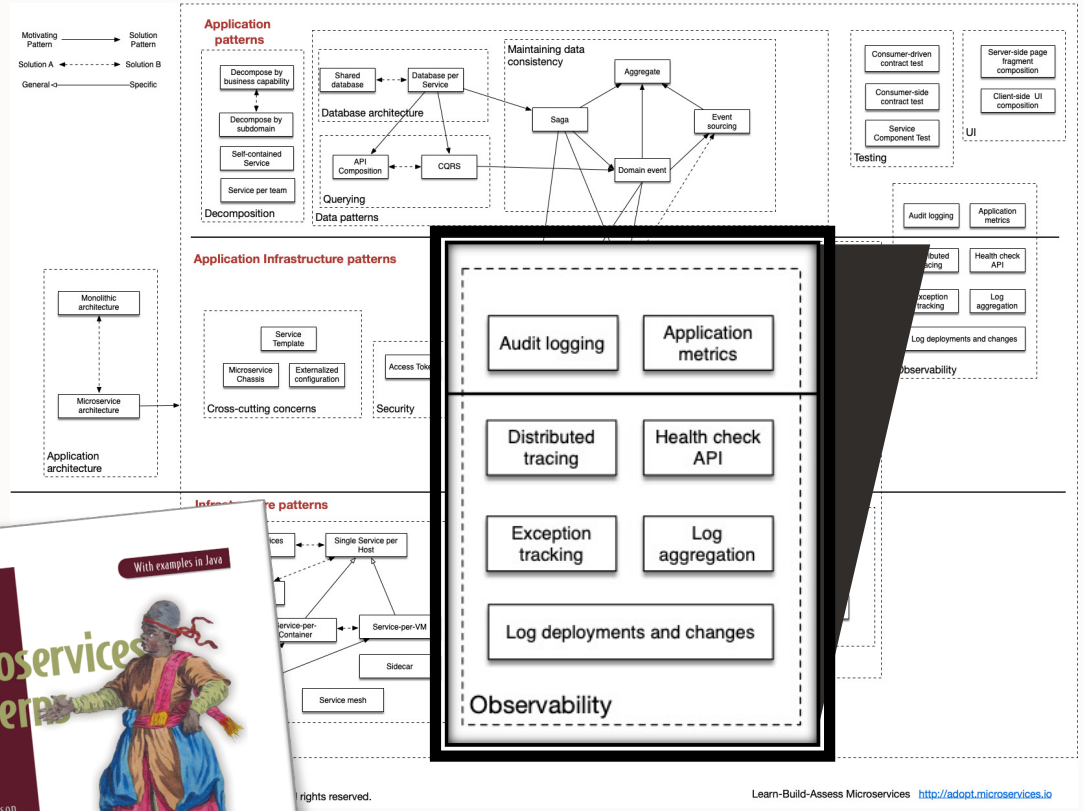
- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

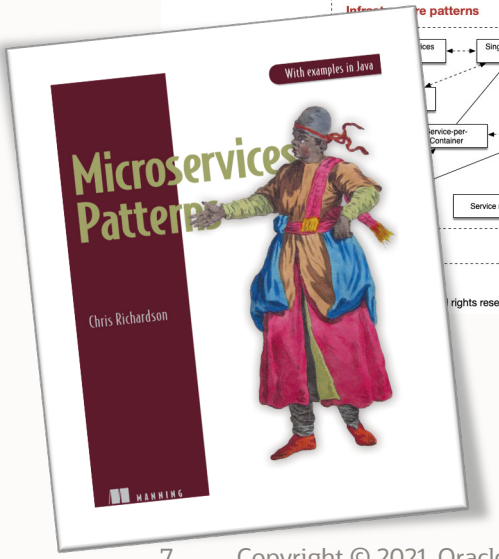
l.cncf.io

*This Cloud Native Trail Map is a recommended process for leveraging Open Source, cloud native technologies*

# Observability Patterns

## Microservice Patterns



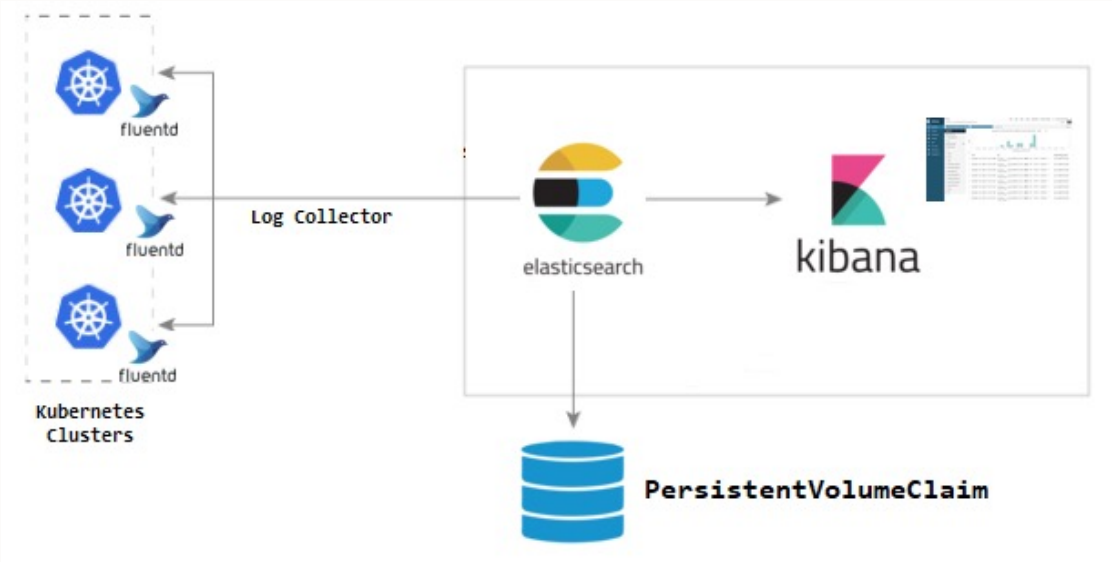



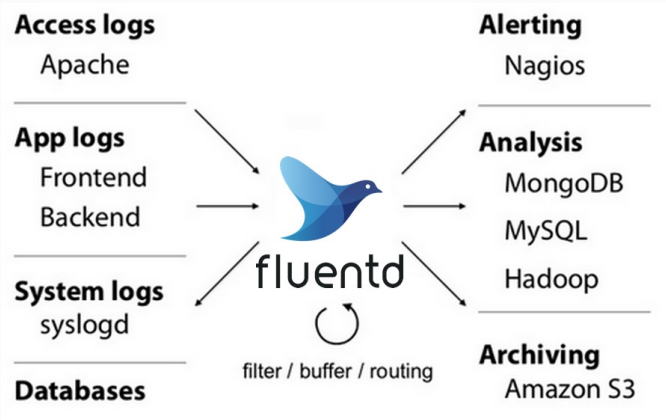

According to microservices architecture and modern systems design there are **7 patterns** to help understand the behavior of an application and troubleshoot problems:

- **Log aggregation** - aggregate application logs
- **Application metrics** - instrument a service's code to gather statistics about operations
- **Audit logging** - record user activity in a database
- **Distributed tracing** - instrument services with code that assigns each external request an unique identifier that is passed between services. Record information (e.g. start time, end time) about the work (e.g. service requests) performed when handling the external request in a centralized service
- **Exception tracking** - report all exceptions to a centralized exception tracking service that aggregates and tracks exceptions and notifies developers.
- **Health check API** - service API (e.g. HTTP endpoint) that returns the health of the service and is intended to be pinged, for example, by a monitoring service
- **Log deployments and changes** – CI/CD focus.

# Log aggregation

How to understand the behavior of an application and troubleshoot problems?

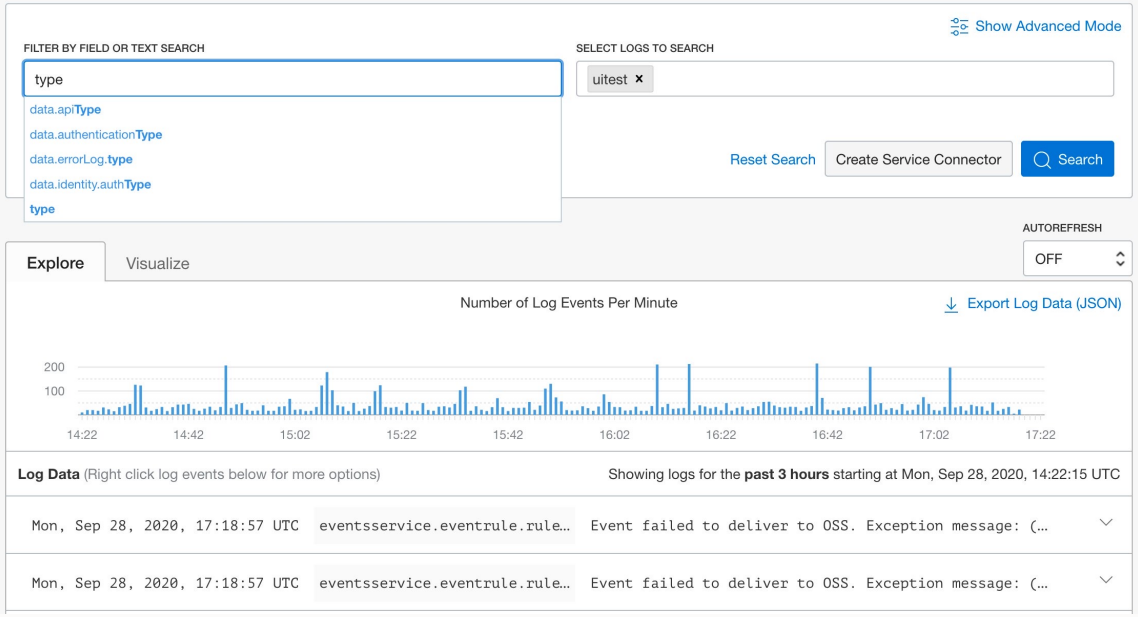



## Oracle Cloud Infrastructure Logging

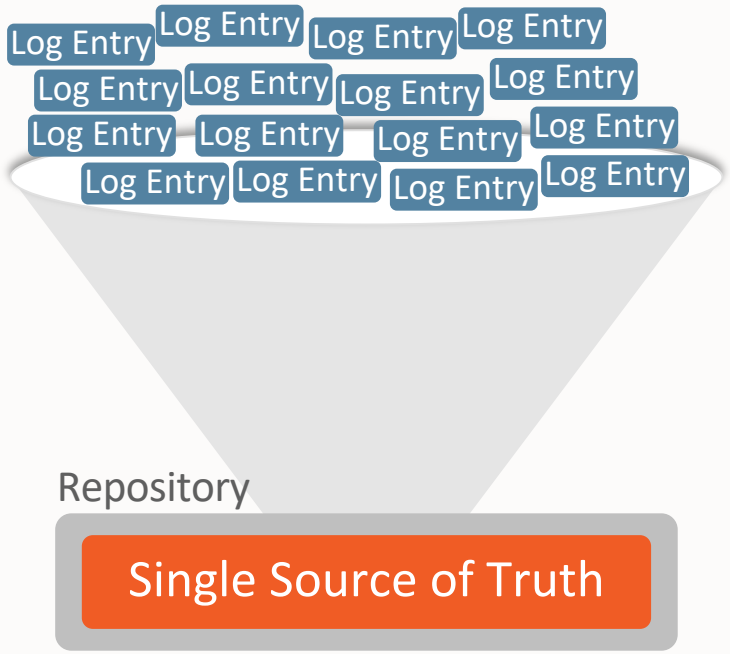
fluentd cloudevents

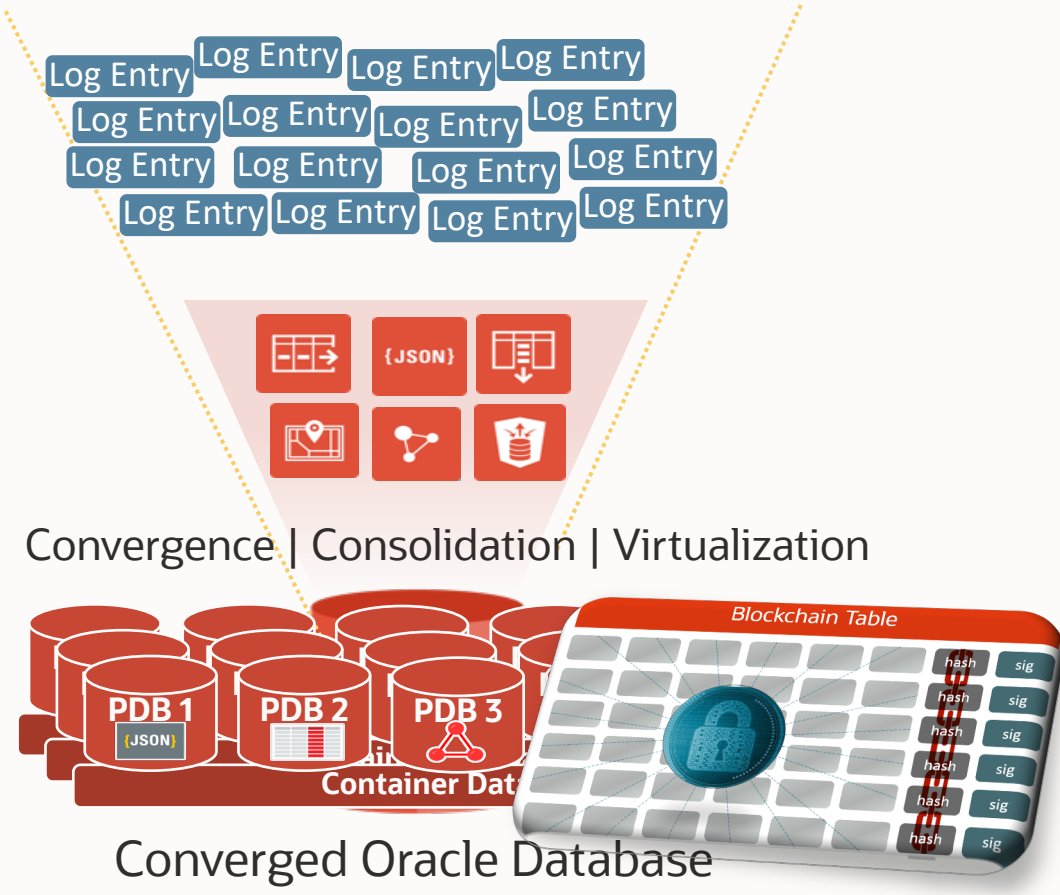
ORACLE Cloud Infrastructure



# Audit Logging

Keep tracking of actions a user has recently performed (Customer Support, Compliance, etc)



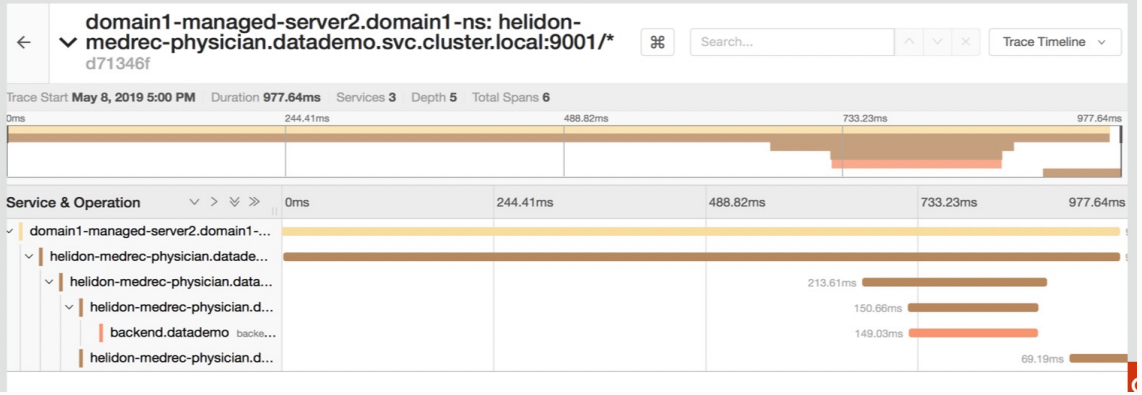



# Observability and Distributed Tracing

- Observability: distributed tracing/logging across services
  - Trace, Span, SpanContext, SpanId
- OpenTelemetry: OpenTracing + OpenCensus
- Kiali, Jaeger, and Grafana stop at the edge of the DB
  - Looking into end-to-end distributed tracing

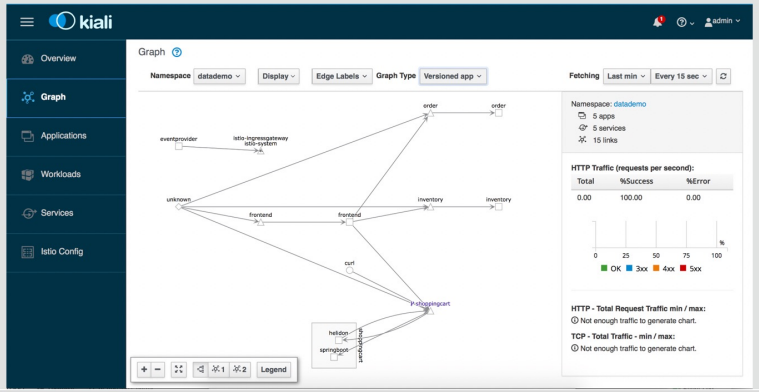
## OpenTelemetry - Jaeger

Tracing across full call path of activity
Trace from WebLogic to Helidon MP to Helidon SE (soon to Oracle DB and AQ as well)…



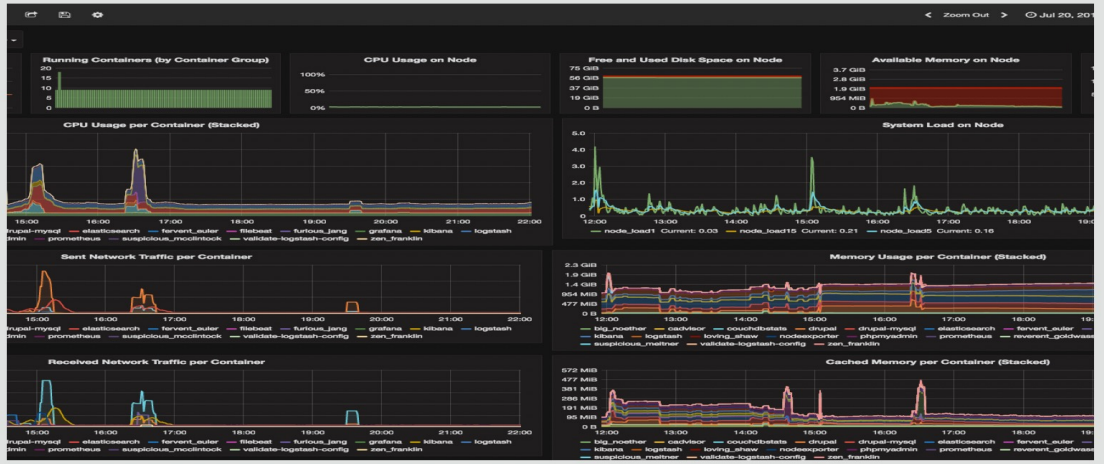

## Kiali

Traffic Topology, Health Status, Workload




## Grafana

Analytics, monitoring, alerting for time-series data

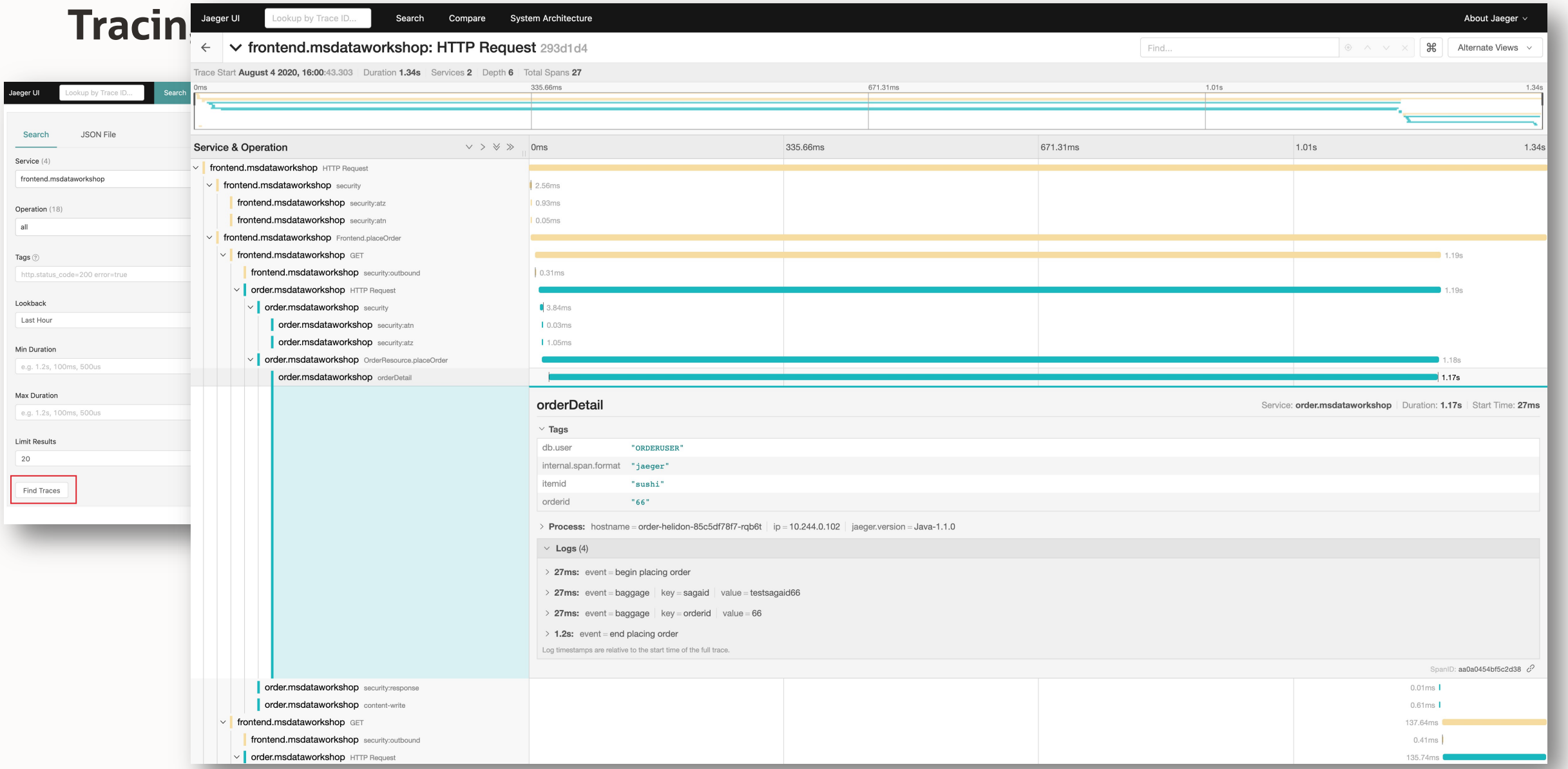


# Tracing Using Jaeger

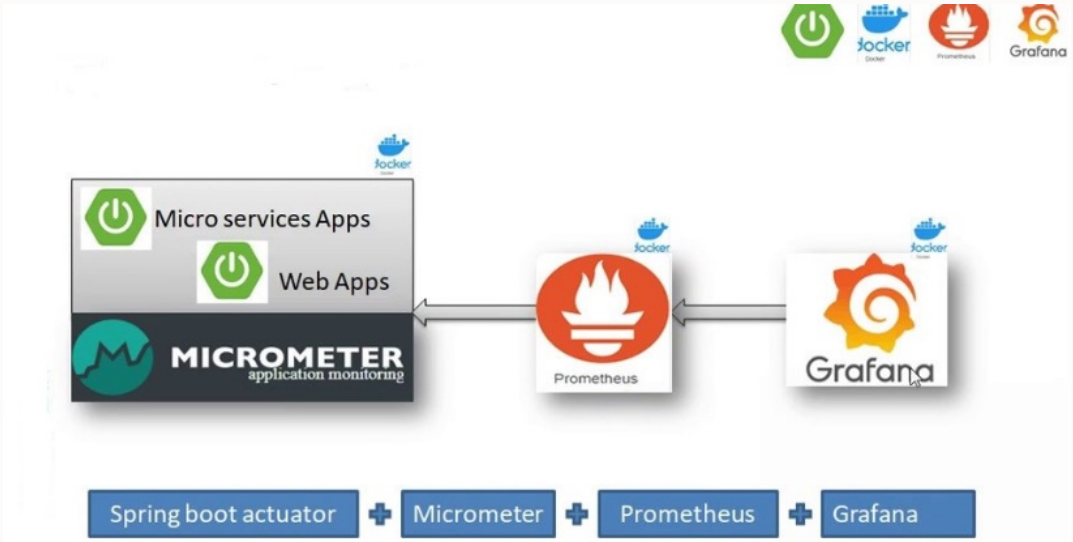
OrderResource.java

```
91      @Path("/placeOrder")
92      @GET
93      @Produces(MediaType.APPLICATION_JSON)
94      @Traced(operationName = "OrderResource.placeOrder")
95      @Timed(name = "placeOrder_timed") //length of time of an object
96      @Counted(name = "placeOrder_counted") //amount of invocations
97      public Response placeOrder(@QueryParam("orderid") String orderid, @QueryParam("itemid") String itemid,
98                                 @QueryParam("deliverylocation") String deliverylocation) {
99          System.out.println("--->placeOrder... orderid:" + orderid + " itemid:" + itemid);
100         startEventConsumerIfNotStarted();
101         OrderDetail orderDetail = new OrderDetail();
102         orderDetail.setOrderId(orderid);
103         orderDetail.setItemId(itemid);
104         orderDetail.setOrderStatus("pending");
105         orderDetail.setDeliveryLocation(deliverylocation);
106         cachedOrders.put(orderid, orderDetail);
107
108         Span activeSpan = tracer.buildSpan( operationName: "orderDetail").asChildOf(tracer.activeSpan()).start();
109         activeSpan.log("begin placing order"); // logs are for a specific moment or event within the span (in contrast to tags wh
110         activeSpan.setTag("orderid", orderid); //tags are annotations of spans in order to query, filter, and comprehend trace da
```

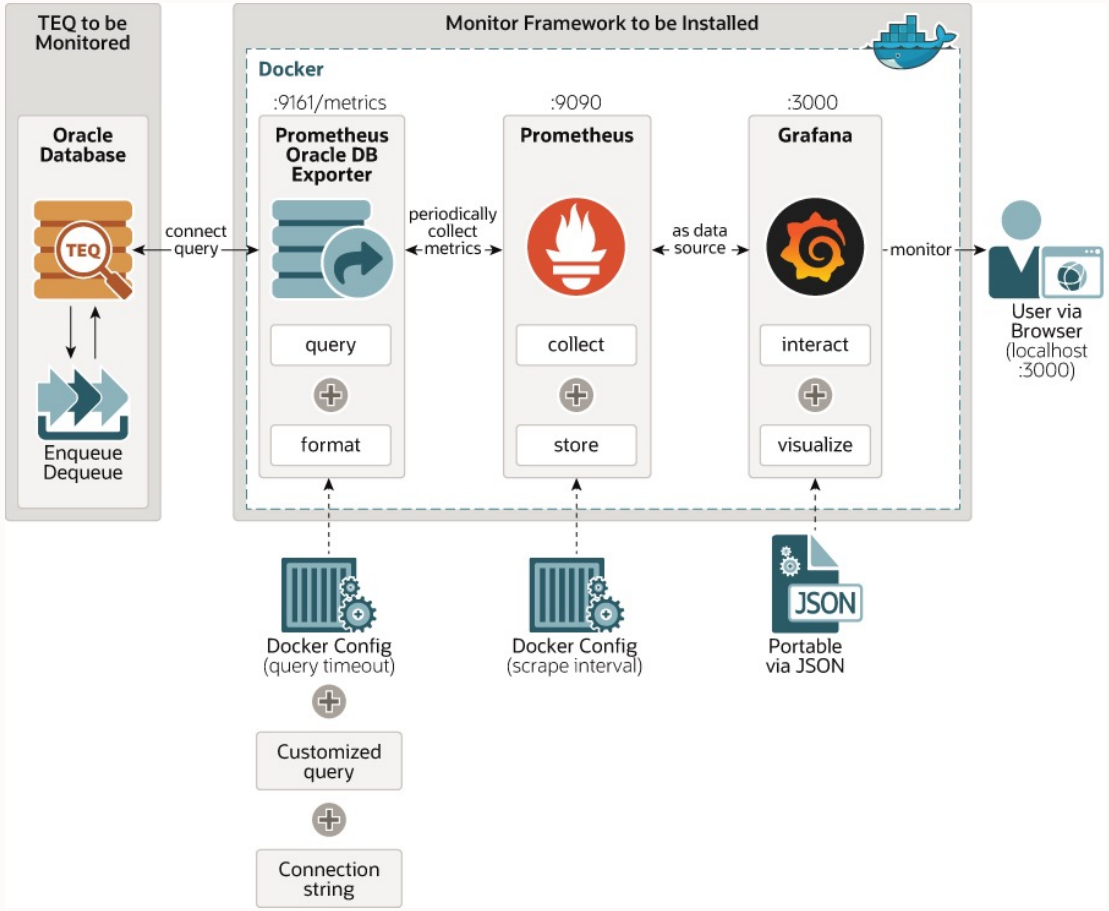
# Tracin



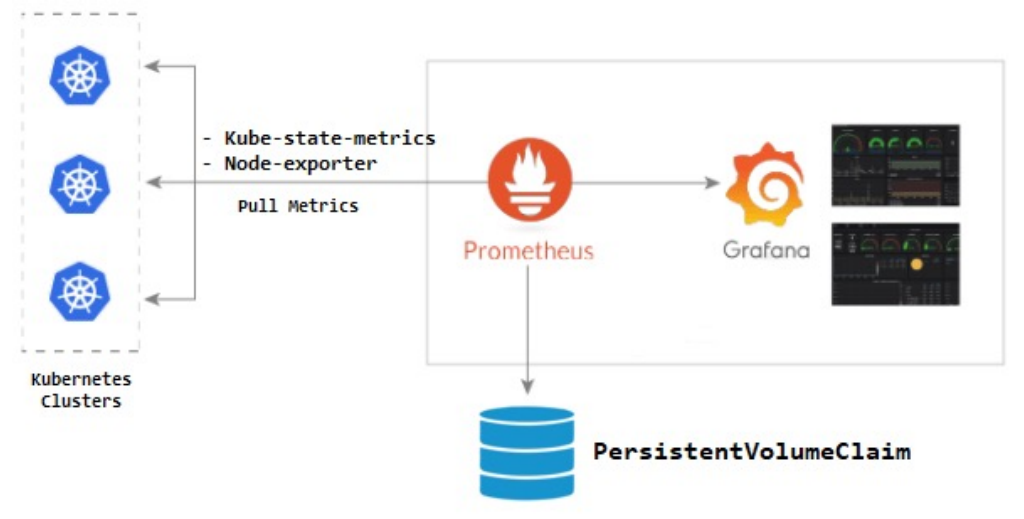
# Monitoring Application Microservices & Events (Streams)

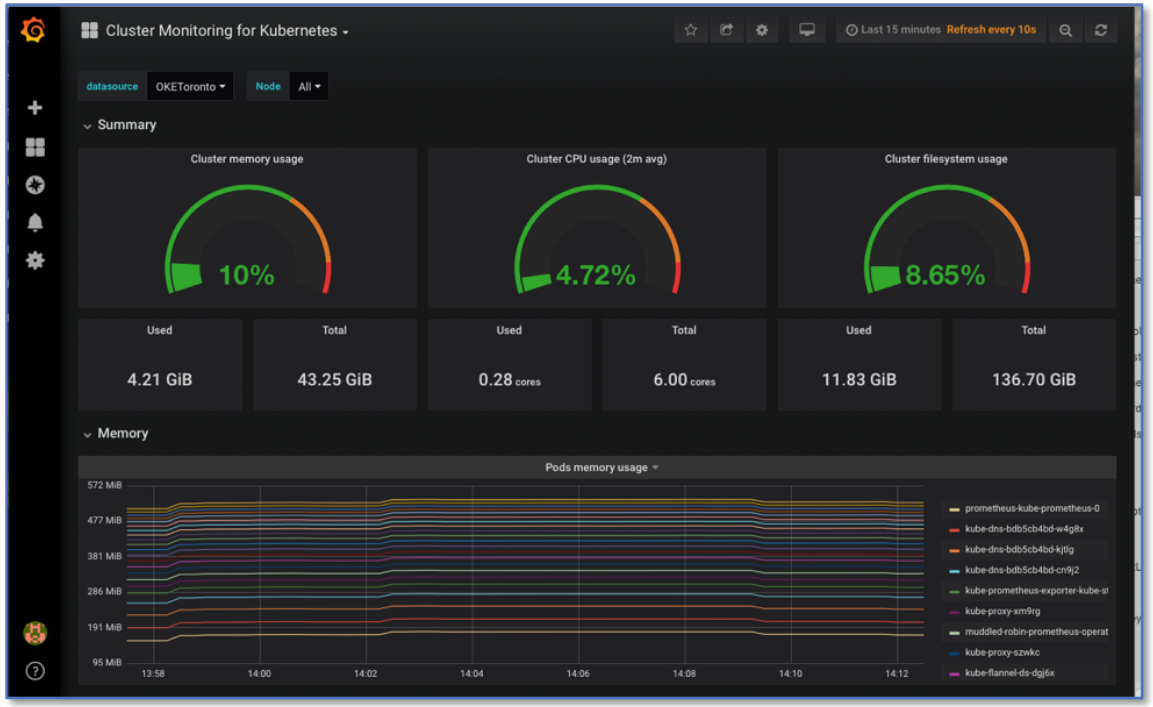



## *Monitoring Oracle DB Transaction Event Queue*



# Monitoring Infrastructures



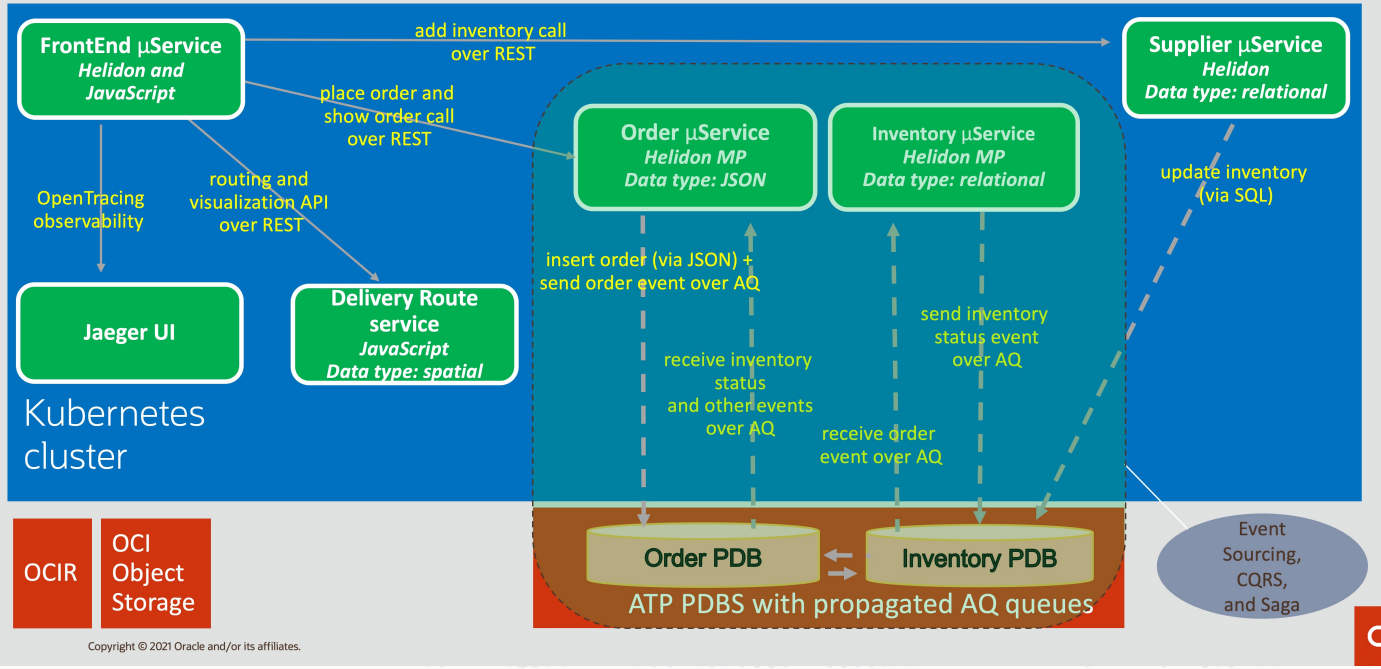



# Building Microservices with Oracle Converged Database Workshop

Modernize and simplify your data-driven applications with microservices architecture using Database Cloud and Helidon on OCI.

**Workshop length:** 2 hours

## "GrabDish" Food Order App Architecture




### Workshop Outline

- Setup OCI, OKE and ATP
- Build and deploy microservices
- Setup database connection secrets, tables and AQ messaging
- Data-centric microservices walkthrough with Helidon MP
- Polygot Microservices
- Scaling the Application
- Create an APEX App to make sense of the data

https://bit.ly/simplifymicroservices


# Paulo Alberto Simões

Developer Evangelist,
Data-Driven **Microservices** with **Converged Database at Oracle**

Mentor, **Oracle for Startups** | Ambassador, **Oracle Academy**
Cloud-Native Computing Foundation **Ambassador**

**Stay Connected**

LinkedIn, Twitter, Instagram, Clubhouse, twitch.tv, dev.to : **pasimoes**


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.