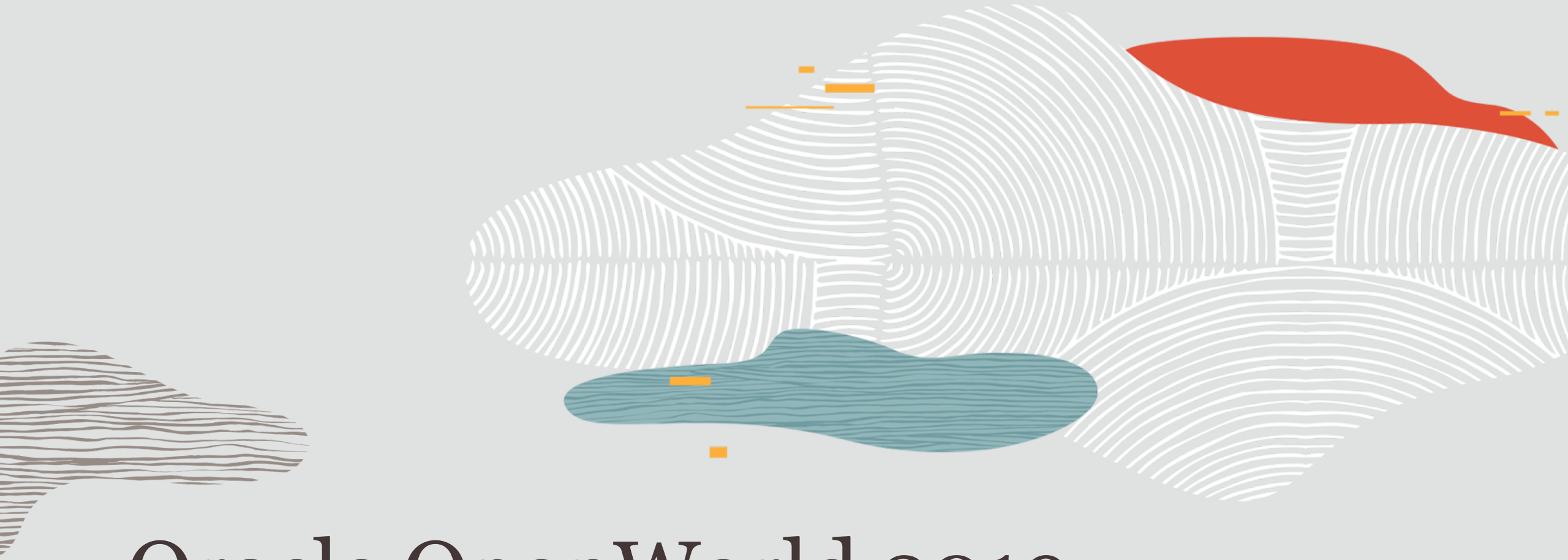


Oracle OpenWorld 2019

SAN FRANCISCO

Microservices Essentials: Kubernetes and Ecosystem, Data, and Transaction Patterns [DEV1730]

Kuassi Mensah, Director of Product Management, Oracle (@kmensah)

Paul Parkinson, Consultant Member Technical Staff (@paulparkinson)

September 2019

Safe Harbor

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

Agenda

- Announce
- Cloud Native Platform: Containers, Kubernetes, and MicroServices
- Data Management: Pitfalls and Best Practices
- Transaction Management: Pitfalls and Best Practices
- Demo

Oracle JDBC drivers 19.3 on Central Maven

https://t.co/MUmHZD6YSK

https://repo1.maven.org/maven2/com/oracle/ojdbc/

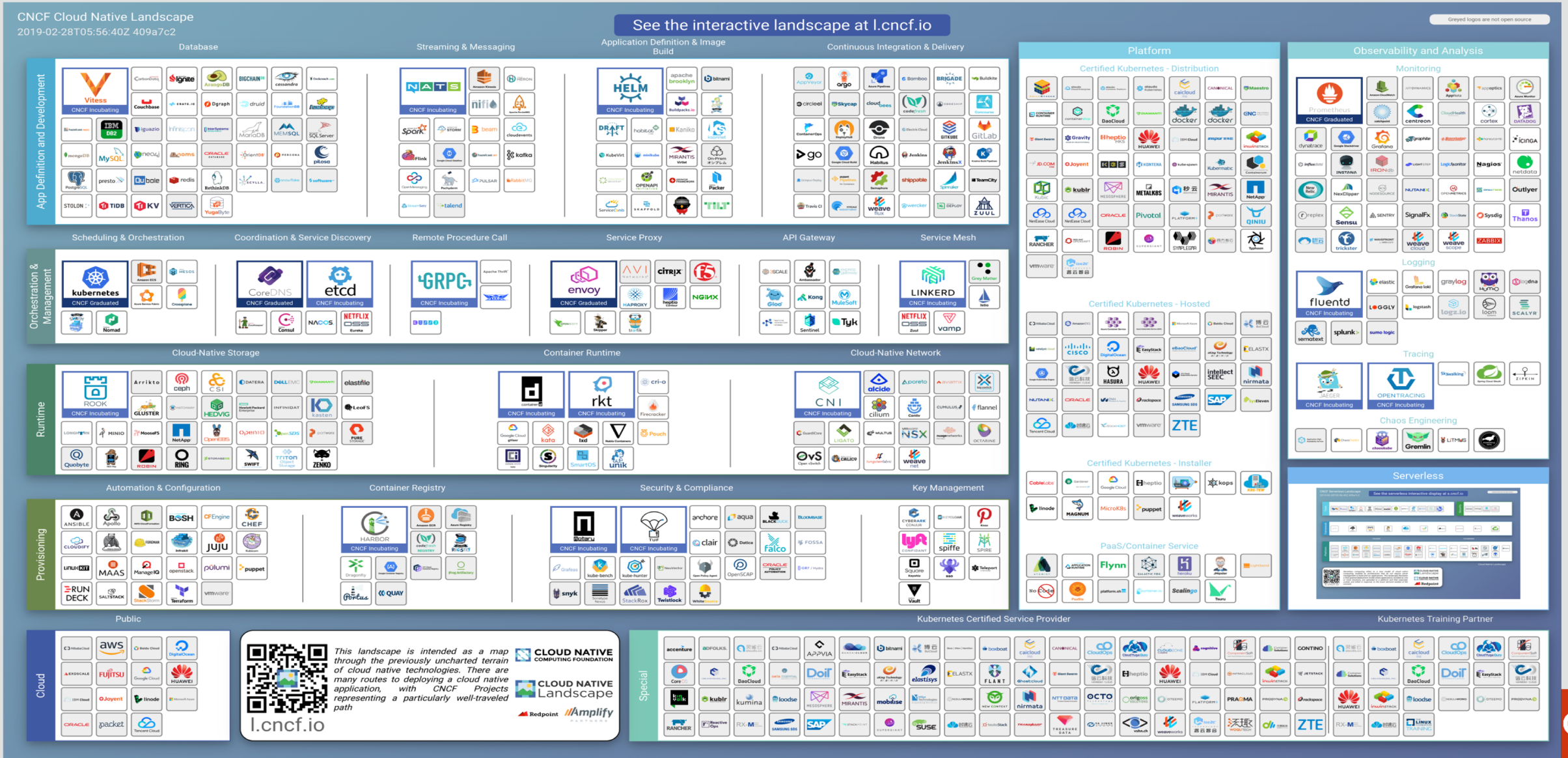
<groupId>**com.oracle.ojdbc**</groupId>

<artifactId>**ojdbc8**</artifactId>

<version>**19.3.0.0**</version>

Cloud Native Platform: Containers, Kubernetes, and MicroServices

The Cloud Native Computing Foundation



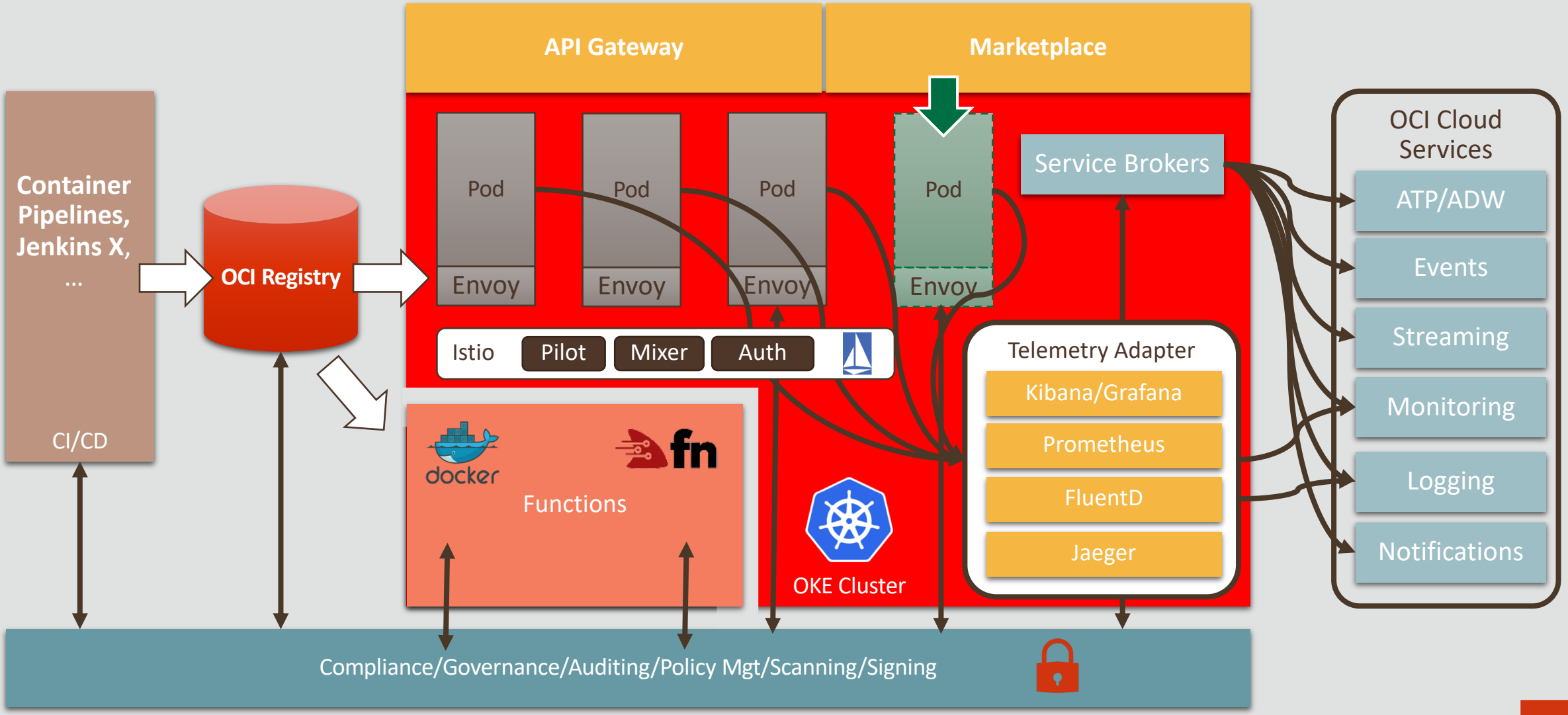
Kubernetes / Microservices

Ideal for microservices…

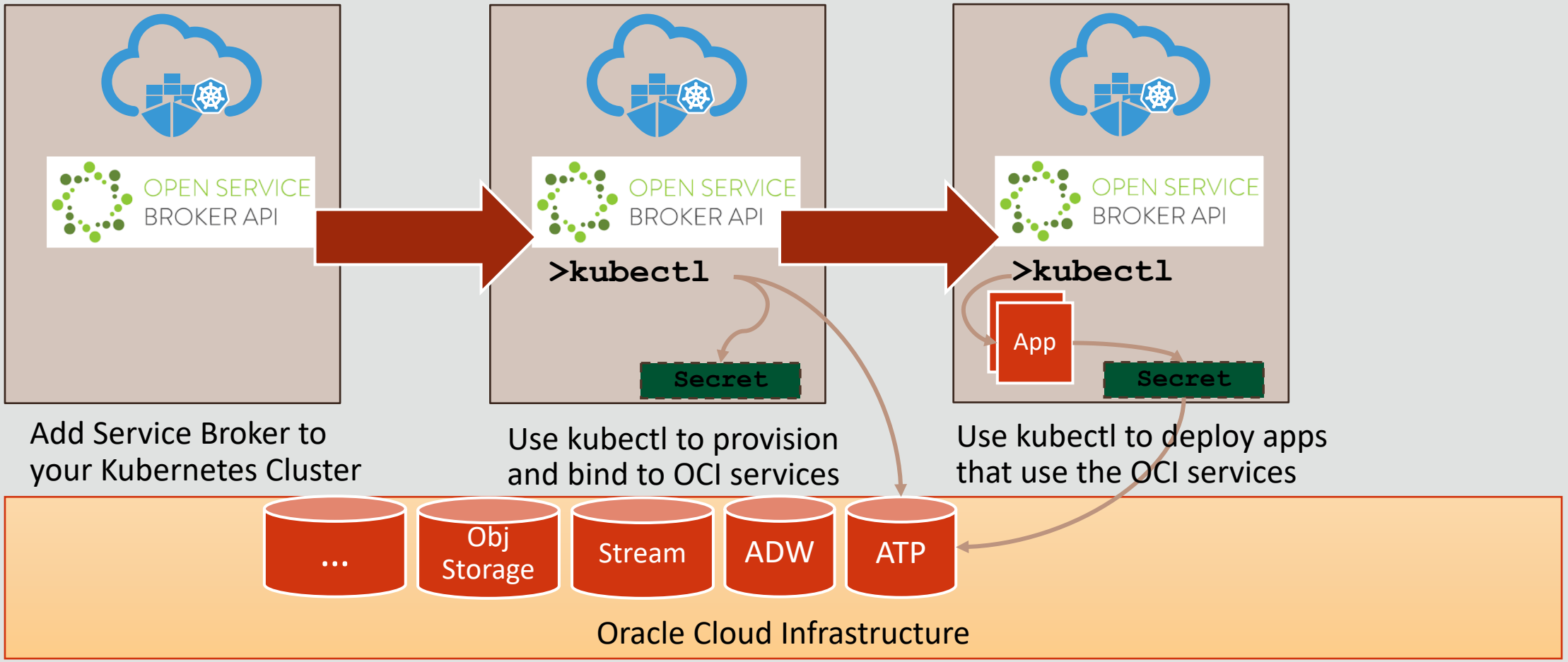
- Pods, a runnable unit of work (container images developed by different teams into a single deployable unit); usually holds 1 or 2 containers
- Load balancing, naming and discovery isolate one microservice from another
- Namespaces provide isolation and access control so that each microservice can control the degree to which other services interact with it.
- Platform services tells the rest of the Kubernetes environment what *services* your application provides.
- Api-registry: tracks of all available services and the resources they expose.

Oracle Kubernetes Engine

Roadmap: Enterprise Container Platform



OCI Service Broker for Kubernetes



Standard Cloud Events

cloudevents

Example

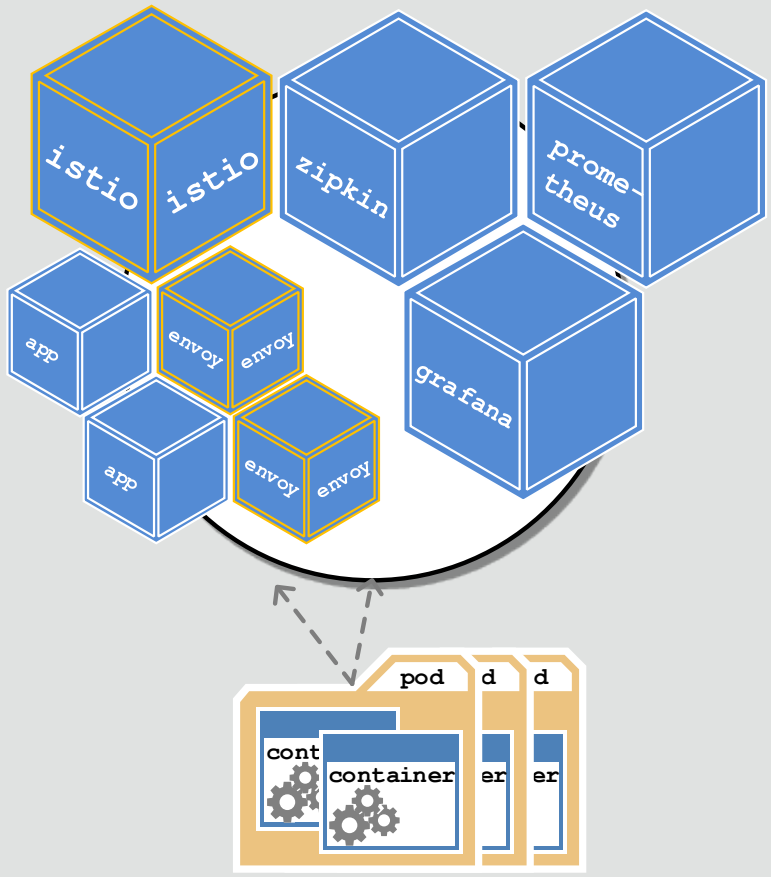
```
{
  "cloudEventsVersion" : "0.1",
  "eventType" : "com.example.someevent",
  "source" : "/mycontext",
  "eventID" : "A234-1234-1234",
  "eventTime" : "2018-04-05T17:31:00Z",
  "comExampleExtension1" : "value",
  "comExampleExtension2" : {
     "otherValue": 5
  },
  "contentType" : "text/xml",
  "data" : "<much wow=\"xml\"/>"
}
```

Microservices Platform & Service Mesh

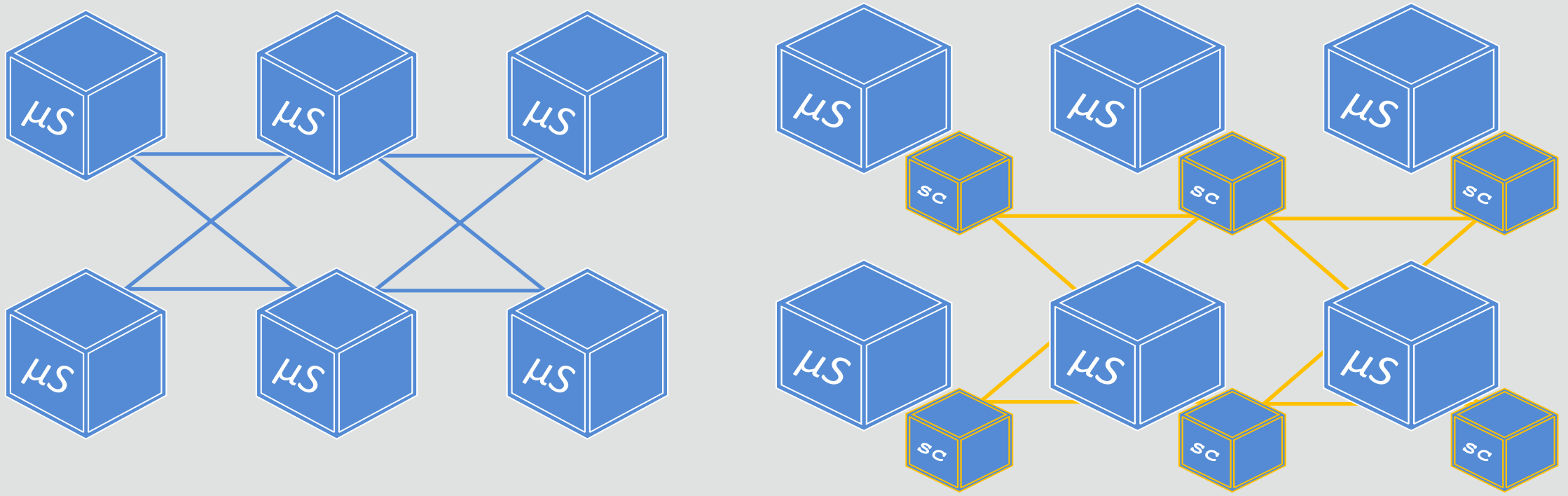
While Kubernetes takes care of a lot, there is still a lot left up to the developer

The need for:

- Strong and Clear API contract
- Service Discovery
- Resilient Networking
- Tracing and Log Correlation Across Services
- Traffic Flow Diagnostics
- Secure Communication



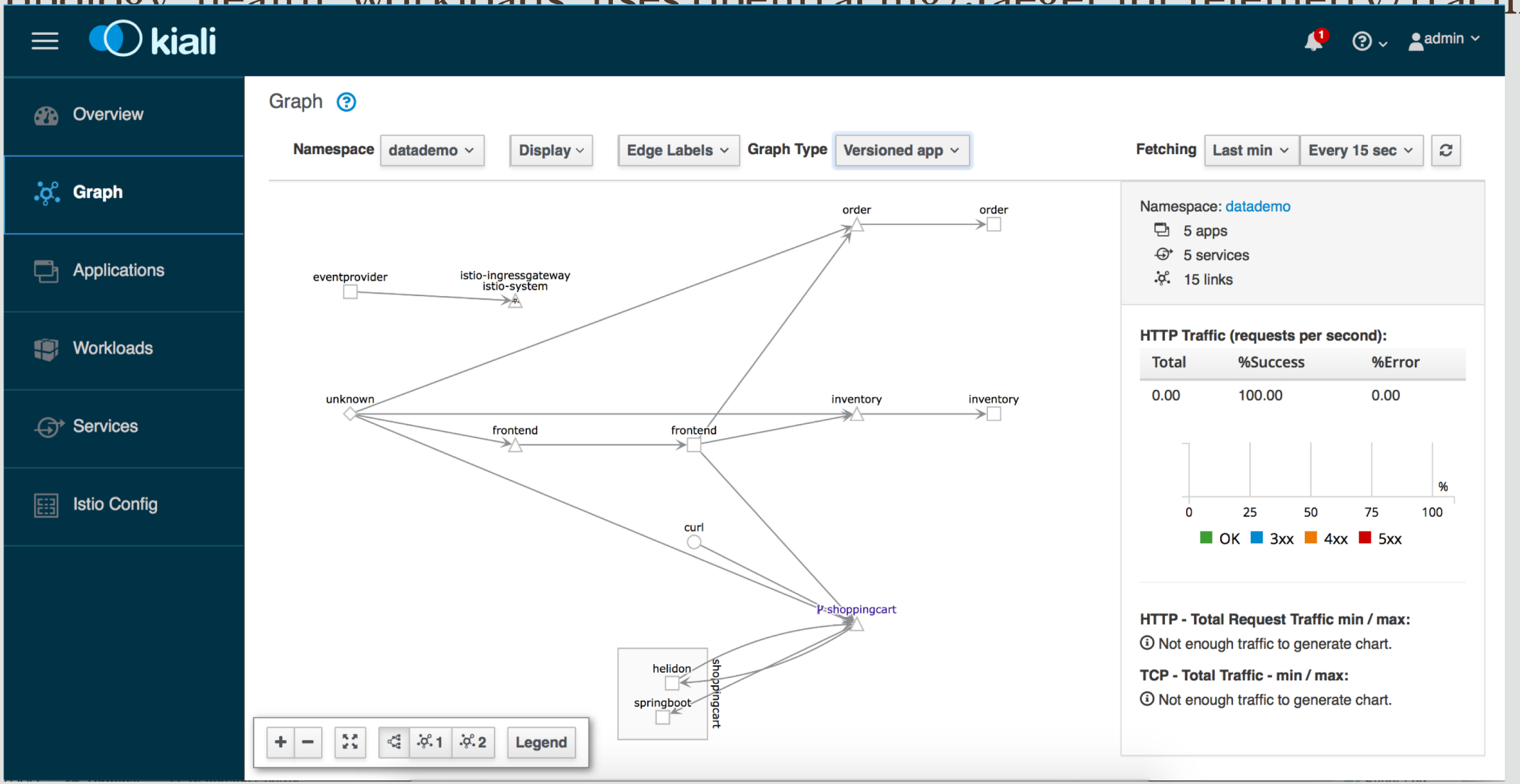
Microservices Platform & Service Mesh



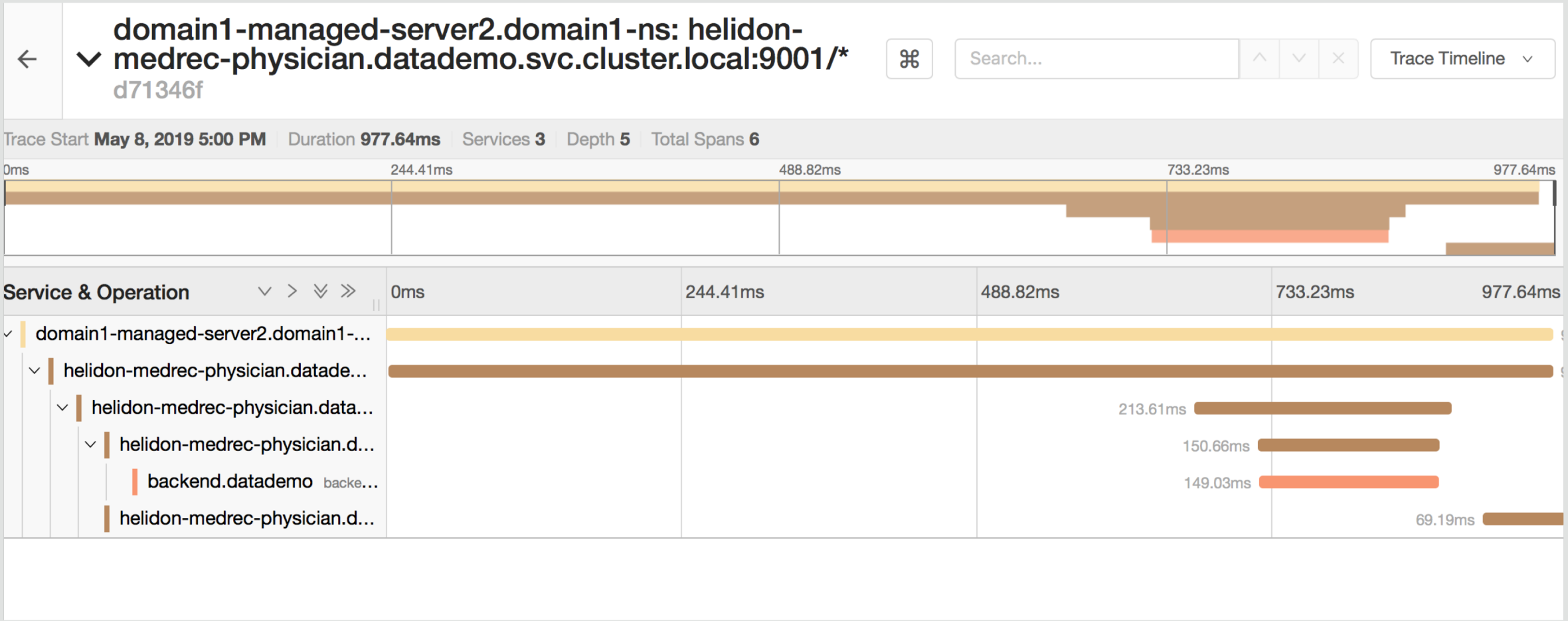
Instead of a direct communication model... Sidecar is inserted for every pod/service

Kiali

- Topology, health, workloads, uses opentracing/Jaeger for telemetry/tracing

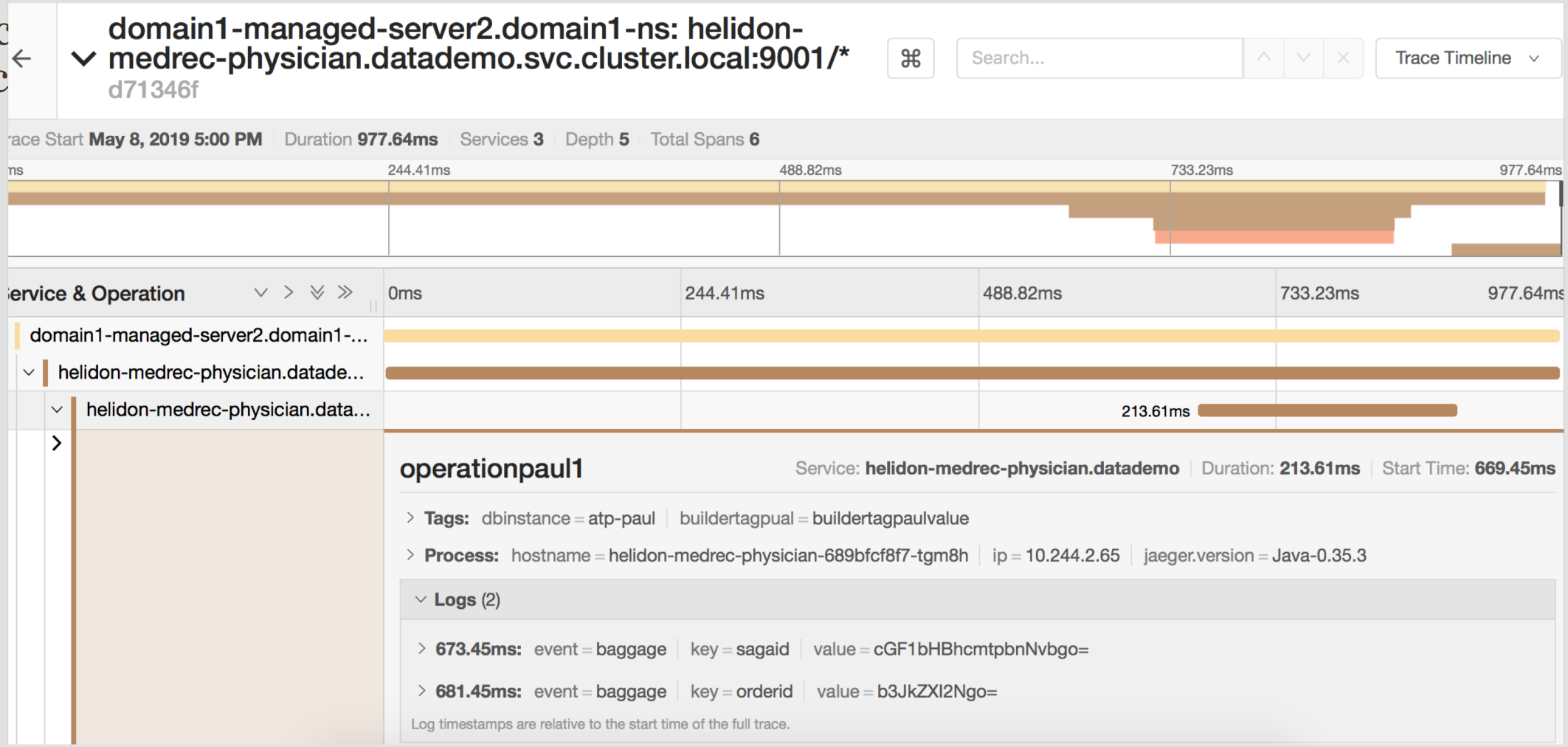


Jaeger


domain1-managed-server2.domain1-ns: helidon-medrec-physician.datademo.svc.cluster.local:9001/*
d71346f
Search...
Trace Timeline
Trace Start May 8, 2019 5:00 PM
Duration 977.64ms
Services 3
Depth 5
Total Spans 6
0ms
244.41ms
488.82ms
733.23ms
977.64ms
Service & Operation
domain1-managed-server2.domain1-...
helidon-medrec-physician.datade...
helidon-medrec-physician.data...
helidon-medrec-physician.d...
backend.datademo backe...
helidon-medrec-physician.d...
213.61ms
150.66ms
149.03ms
69.19ms

Jaeger

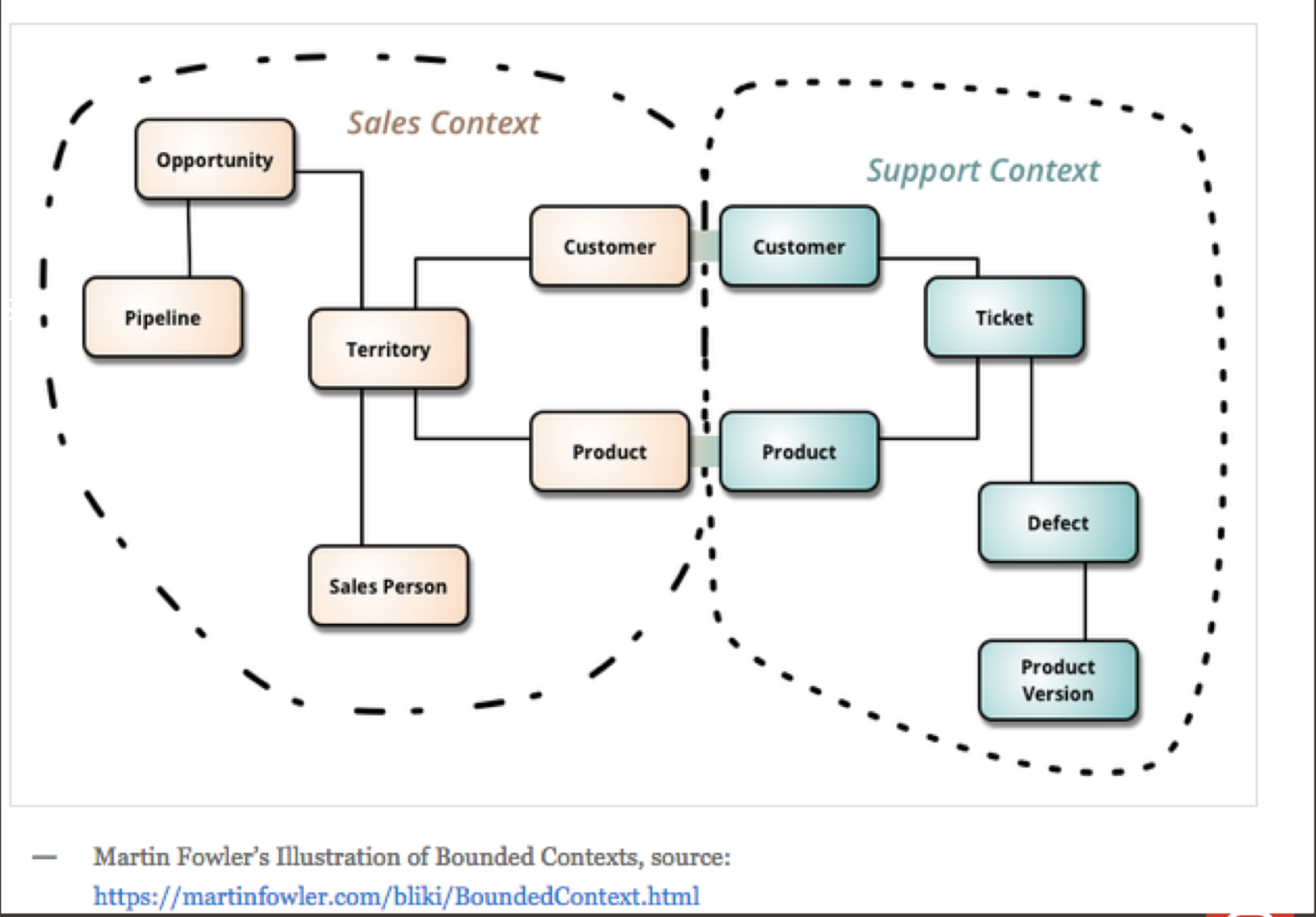
- Trac
- Trac



Data Management: Pitfalls and Best Practices

Domain Driven Design -- Bounded Contexts

- Areas where certain business processes are implemented
- Logical boundaries within which terms have non-ambiguou meaning
- Design microservices against bounded contexts
- Best Practices:
  Split the monolithic database along the lines of the bounded contexts
- However, databases might be shared across services



— Martin Fowler's Illustration of Bounded Contexts, source: https://martinfowler.com/bliki/BoundedContext.html

Microservices Data Management Patterns

- CQRS
- Access through public APIs only
- API Composition (Queries)
  - Front end API Composer queries other services and performs an in-memory join
- Database per service
  - PDBs, Shards, PDB Sharding
- Shared database
  - Tables, Schemas, Editions
- Polyglot Persistence
  - single model
  - multi-model database

Provisioning of Autonomous Database

Service Broker adds connection info for cloud services into cluster via Kubernetes Secrets

Microservice deployment is written to consume Secret

This allows service to be deployed in different environments without any code changes, and pick up the appropriate DB connection automatically

If using Helidon MP datasource reference can be auto-injected in service via microprofile-config.properties .

Eg, the following will inject @Named("**atp1**") DataSource:

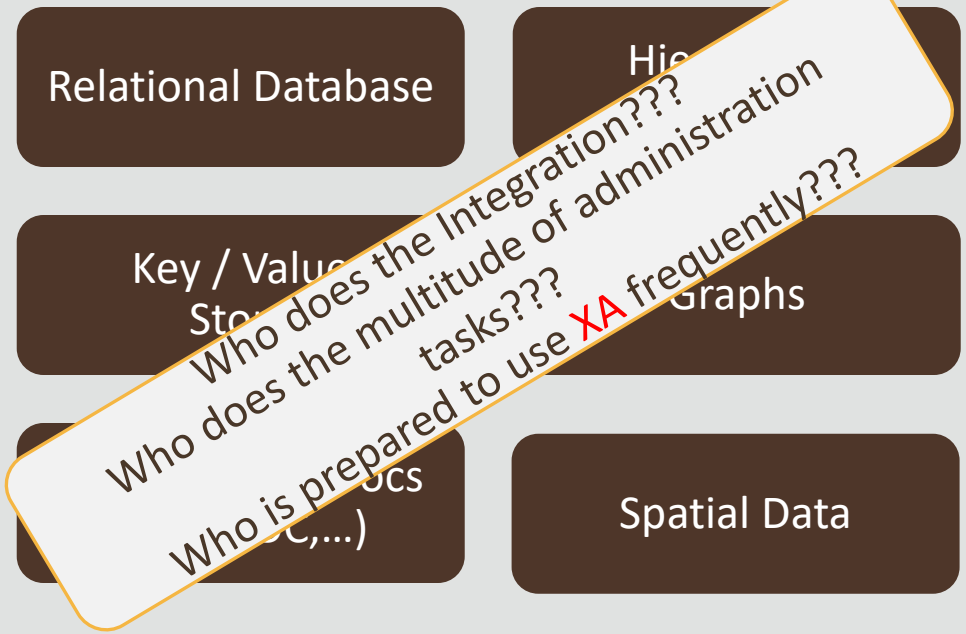
javax.sql.DataSource.atp1.dataSourceClassName= xxxxx
javax.sql.DataSource.atp1.dataSource.url = xxxxx
javax.sql.DataSource.atp1.dataSource.user = xxxxx
javax.sql.DataSource.atp1.dataSource.password = xxxxx

Deployment yaml:

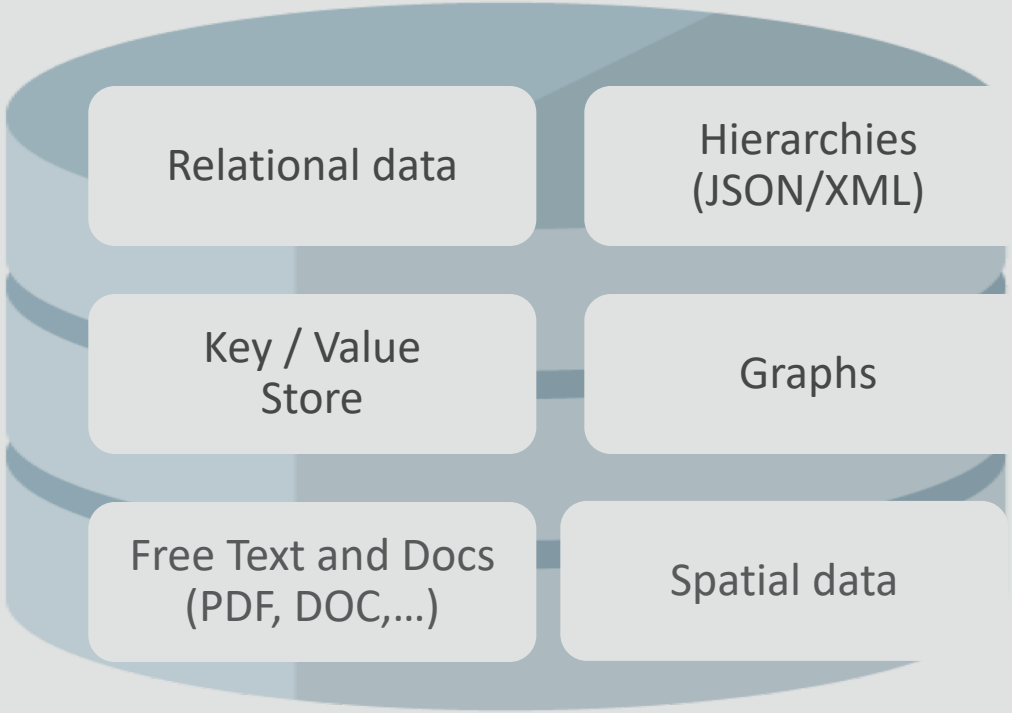
```
env:
- name: javax.sql.DataSource.atp1.dataSource.user
  valueFrom:
    secretKeyRef:
      name: paul-atp-demo
      key: user_name
- name: javax.sql.DataSource.atp1.dataSource.password
  valueFrom:
    secretKeyRef:
      name: atp-user-cred
      key: password
- name: javax.sql.DataSource.atp1.dataSource.url
  value: "jdbc:oracle:thin:@pauldb_HIGH?TNS_ADMIN=/db-demo/creds”

- name: WALLET_PWD
  valueFrom:
    secretKeyRef:
      name: atp-user-cred
      key: walletPassword
  volumeMounts:
  - name: creds
    mountPath: /db-demo/creds
volumes:request.
- name: creds-raw
  secret:
    secretName : paul-atp-demo
```

Multi Model Data: Two Technologies Approaches

Single-model database engines
(One database per data model)



Multi-model database engines
(Single database for all data models)

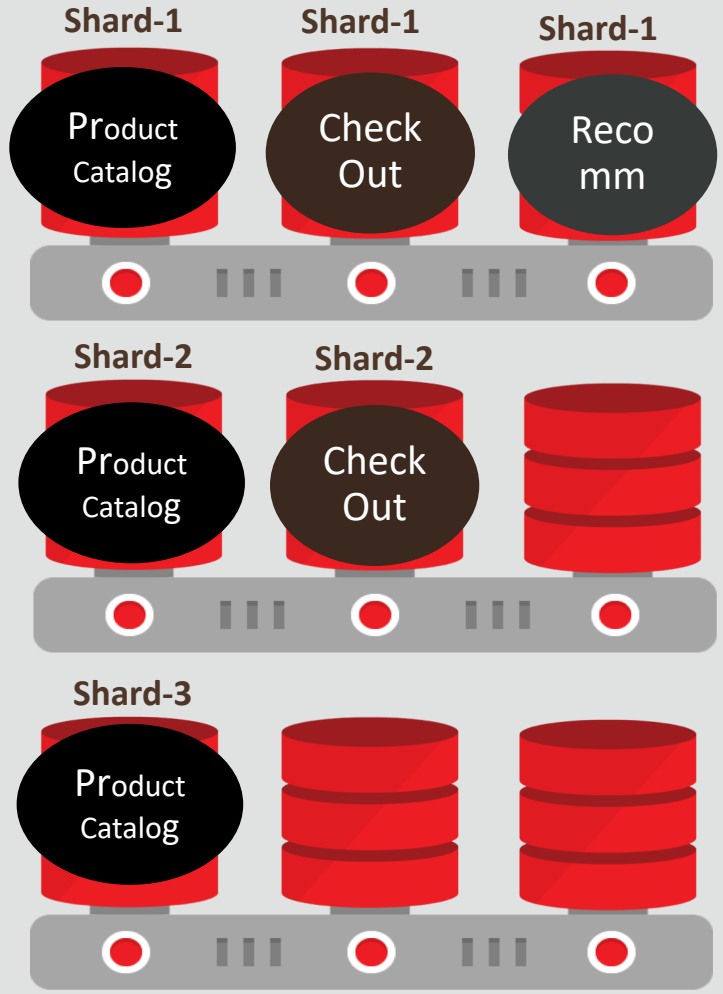


+ + + Multimedia, expressions, event stores, domain specific data, …

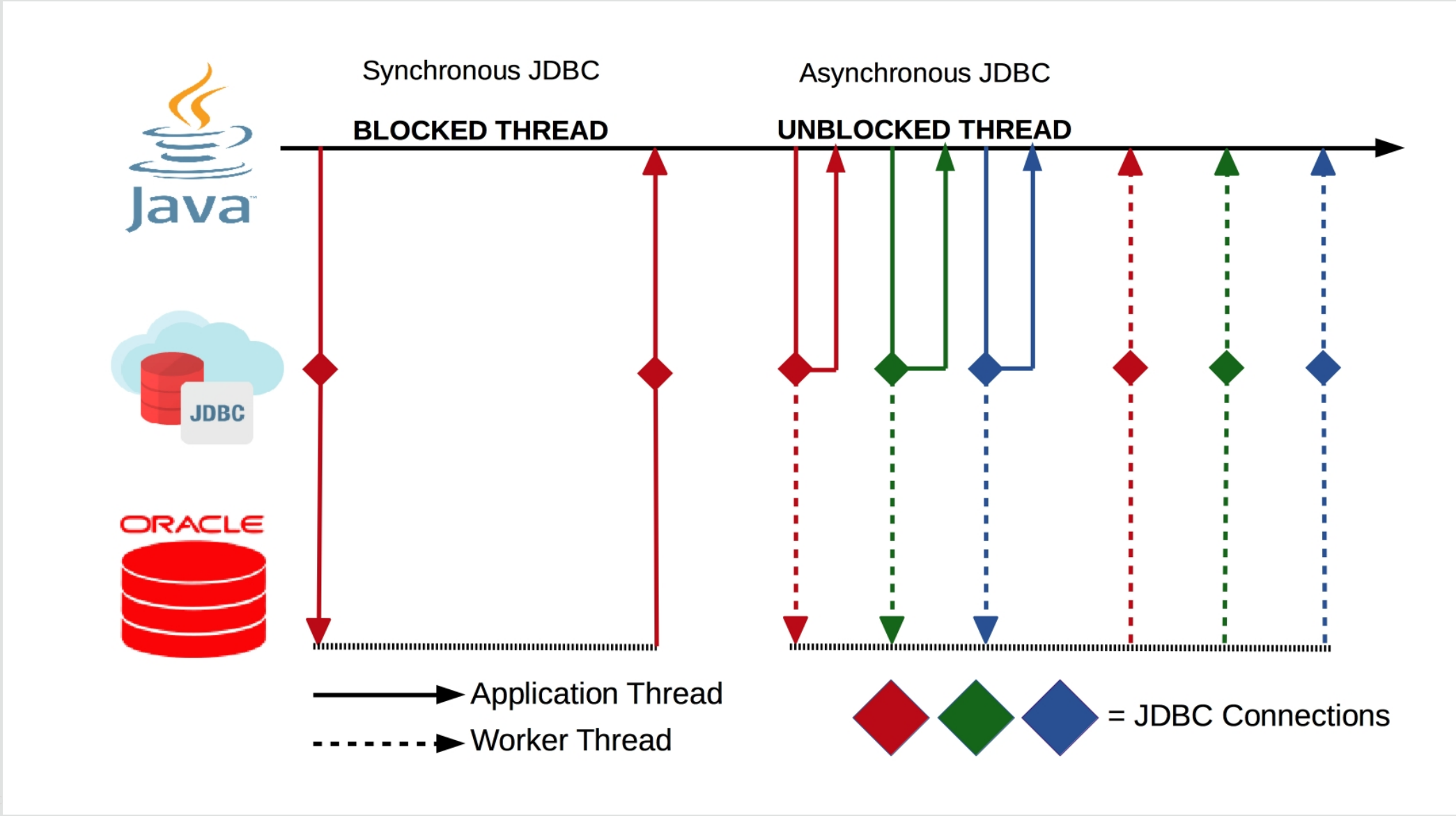
PDB Sharding for Microservices

Scalability, fault isolation and geo-distribution

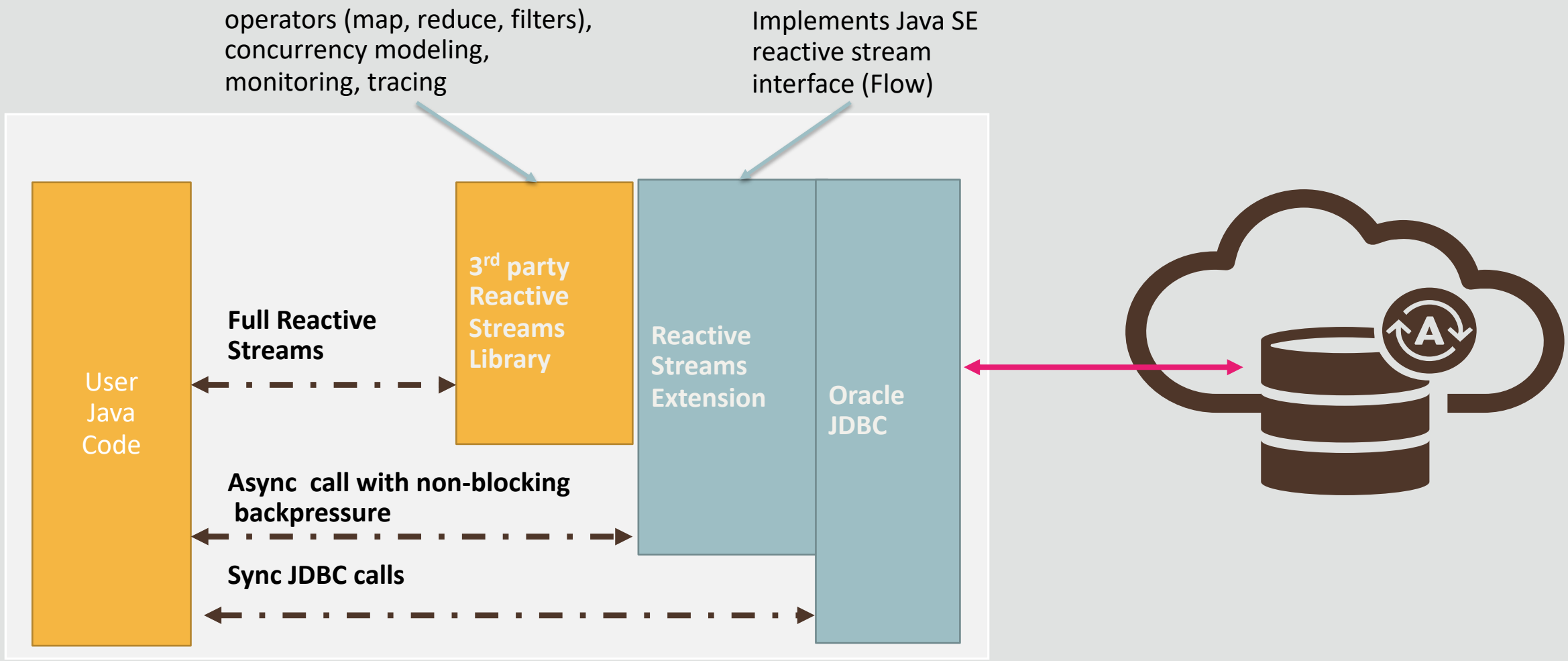
- PDB Sharding in DB 19c
  - Each PDB can be sharded across multiple CDBs
- Provides fault isolation and geo-distribution for microservices
  - Loss of an entire CDB makes only part of a PDB unavailable
- Also allows each microservices to scale its PDB individually
  - More efficient use of resources compared to scaling a monolithic application (CDB).



Asynchronous and Reactive Database Access for MicroServices

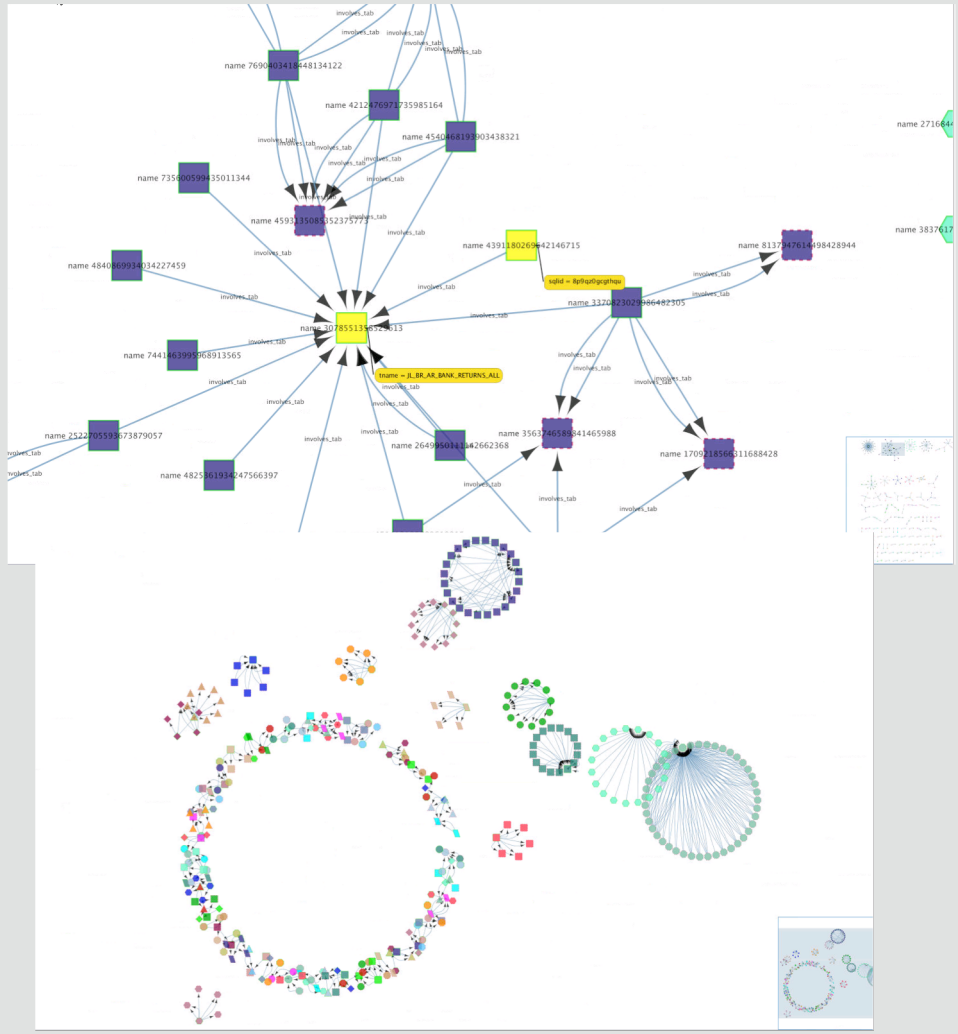


Database Access with Oracle JDBC



Intelligent Schema Refactoring (DB Future)

- Tool to help split monolithic database along lines of sub-domains or bounded contexts
  - Capture information while workload is running
  - Generates a Graph of hotspots and data island
- *Help developers optimally factor the schema of a monolithic application into separate datasets that can be stored in different PDBs*



Transaction Management: Pitfalls and Best Practices

Transactions Design

- Best Practices:perform local transactions within a bounded context
- However, transactions might cross service boundaries

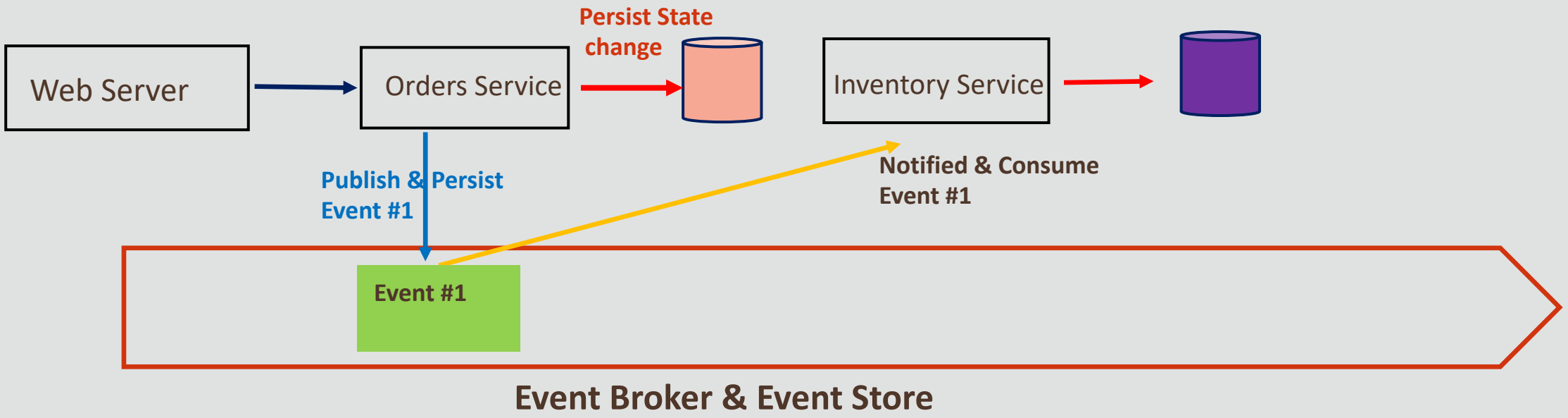
Communication across microservices

- Synchronous
  - REST or gRPC based
- Asynchronous
  - Events
  - Pub/Sub, Notification
  - Loose coupling
  - Immutability
  - State transfer
  - Event Sourcing

Event Sourcing

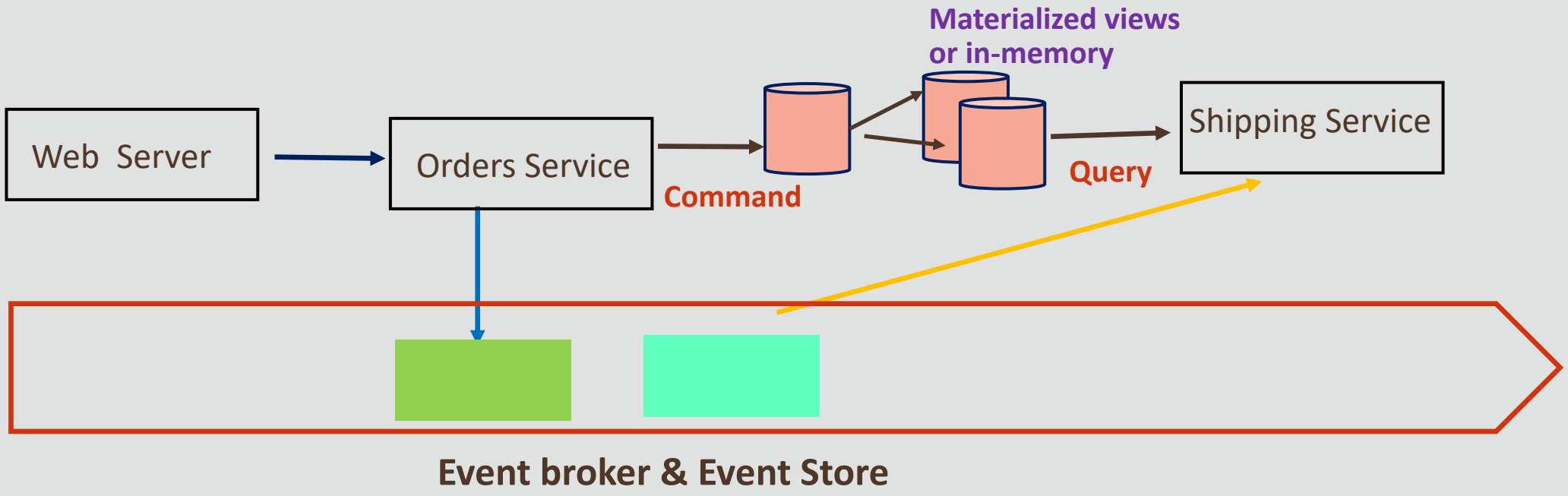
Microservices interact by ”sourcing events” in/out from the Event store via the Event Broker

- The Event store is the single source of truth (Kafka, vanilla Oracle DB, or AQ in future)
- Producers: log events in the Event store then publish a notification
- Consumers: notified when Events are published then “read” the Event store



Command Query Responsibility Segregation - CQRS

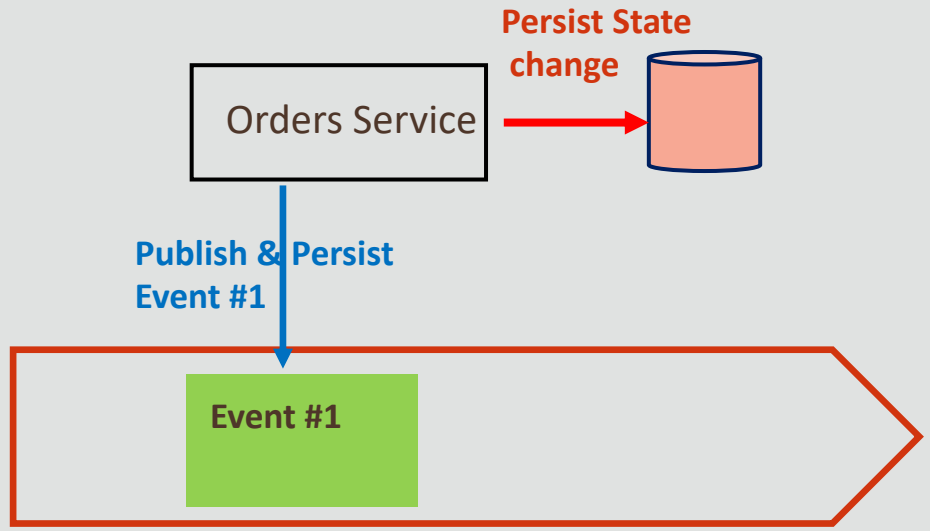
- Commands: produce an aggregate view from events into a Table or Materialized View or In-Memory
- Queries: Retrieve Data from the aggregate view
- Often combined with (and required for) Event Sourcing



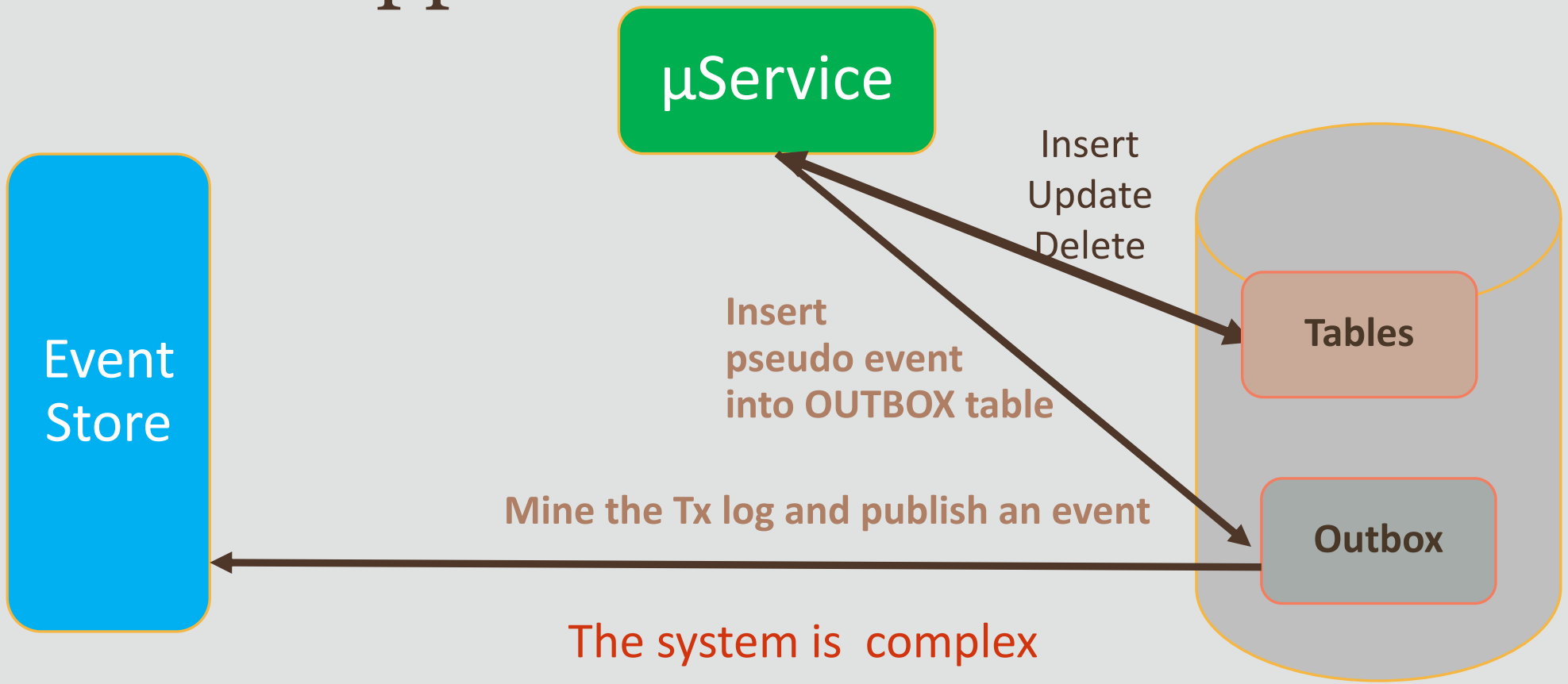
Atomicity of Persisting Events and Database States

Several Techniques

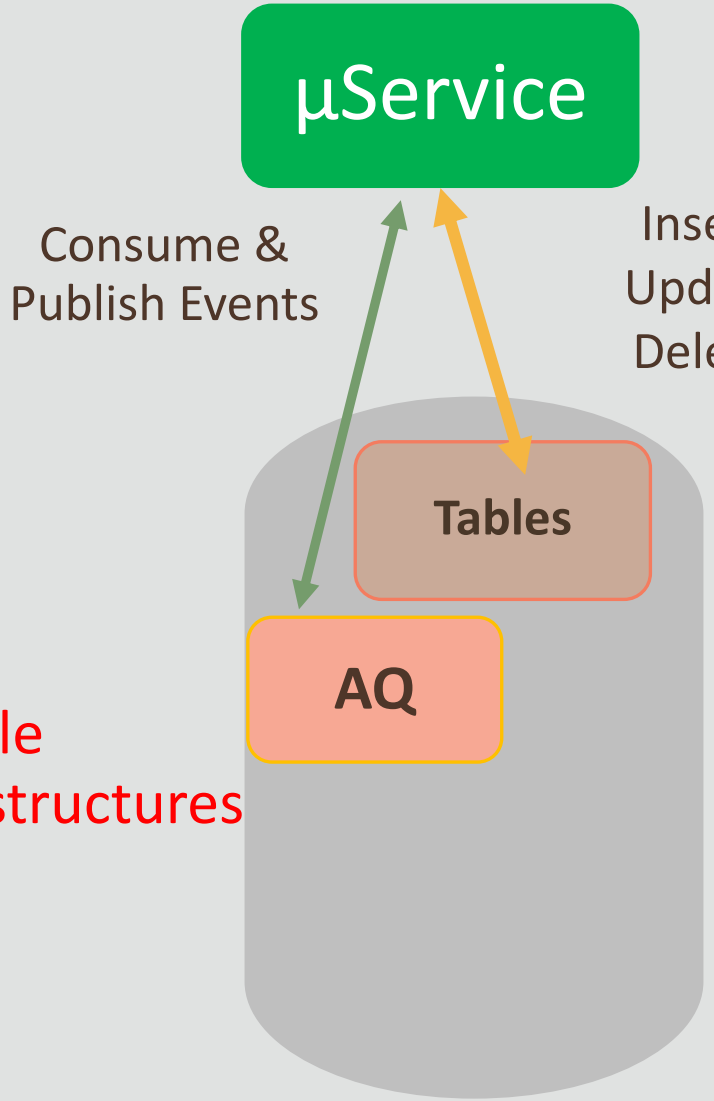
1. Use db table as message queue
   manually re-inventing AQ
2. Outbox table
   Tail db transaction log and publish each message/event inserted into the *outbox* to the message broker.
3. Oracle Advanced Queue (AQ)



The Outbox Approach



The AQ Approach



AQ is one type of event store.

Everything under a single local Tx.

AQ can communicate with other event stores using (exactly once) propagation

The system is simple
Events can have rich data structures

Oracle Database AQ Support for Microservices

- Queue operations and DML Data and message in the same local transaction
- Sharded Queues furnish high scalability
- Client initiated notification
- Handles large message backlogs
- Queue level access privileges (enqueue, dequeue) supports CQRS
- High Availability and Disaster Recovery: Transparent Application Continuity
- 16 year history

Business Transactions across Microservices

Distributed Transactions that span multiple microservices

- E.g., Booking a trip include: flight, hotel, car, shows

- Microservice A calls B that calls C.
  - The workflow or business transaction commits only when C completes
- How do you undo A & B should C fail?
  - Traditional two-phase COMMIT TX is an anti-pattern with MicroServices
  - Each service commits individually (locally): how to ensure consistency?

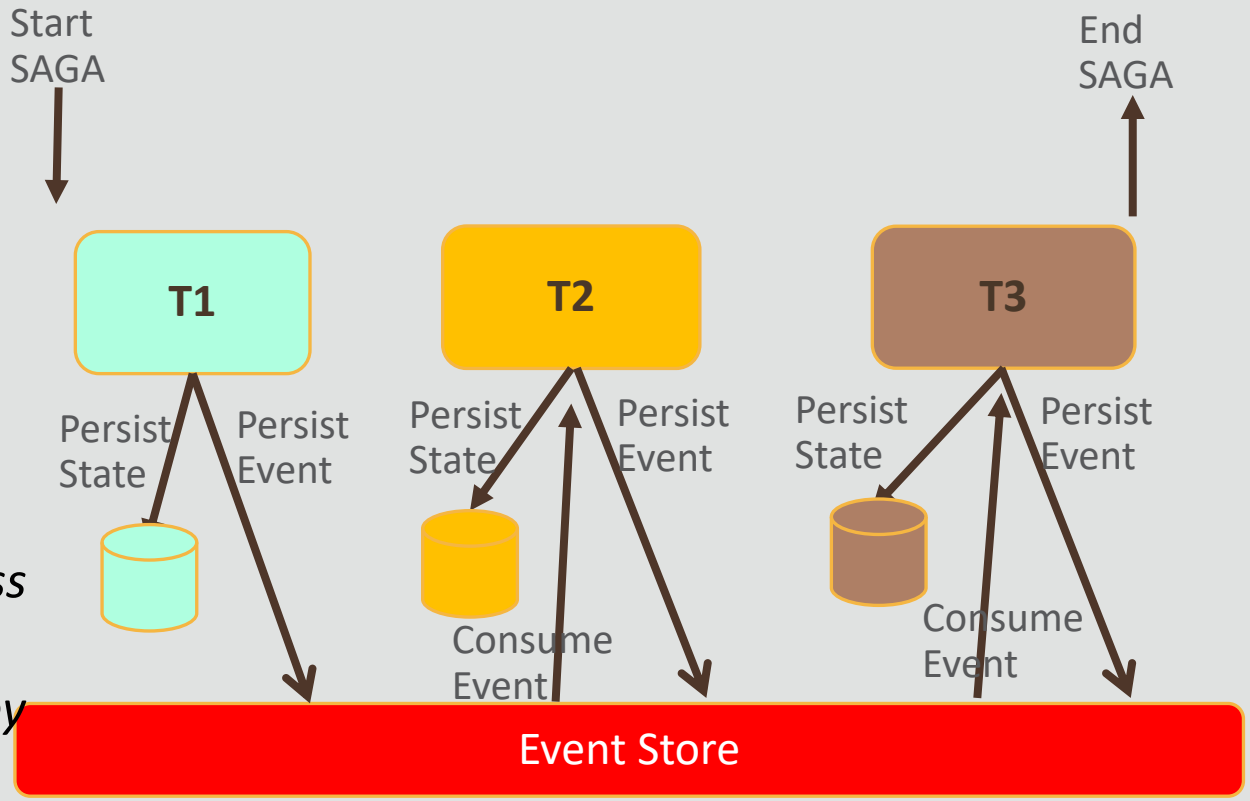
SAGA Pattern

"A saga is a sequence of local transactions.

Each local transaction updates the state (database) and publishes a message or event to trigger the next local transaction in the saga.

If a local transaction fails because it violates a business rule then the saga executes a series of compensating transactions that undo the changes that were made by the preceding local transactions."

https://microservices.io/patterns/data/saga.html



SAGA Consistency

Two approaches for compensating failed member(s) of a distributed Tx

- Choreography
  - The services interact with each other to coordinate well known/defined activity

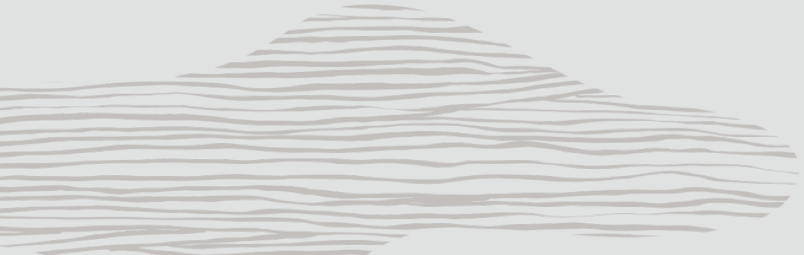
    e.g., the inventory service fails an order from the order service when the order cannot be fulfilled
- Orchestration
  - An orchestrator is used to coordinate the activity; all messages are passed through it
  - The Business Process Management and Notation (BPMN) is an example of a mature orchestration.

Compensation functions could be complex/error-prone; what about automating these?

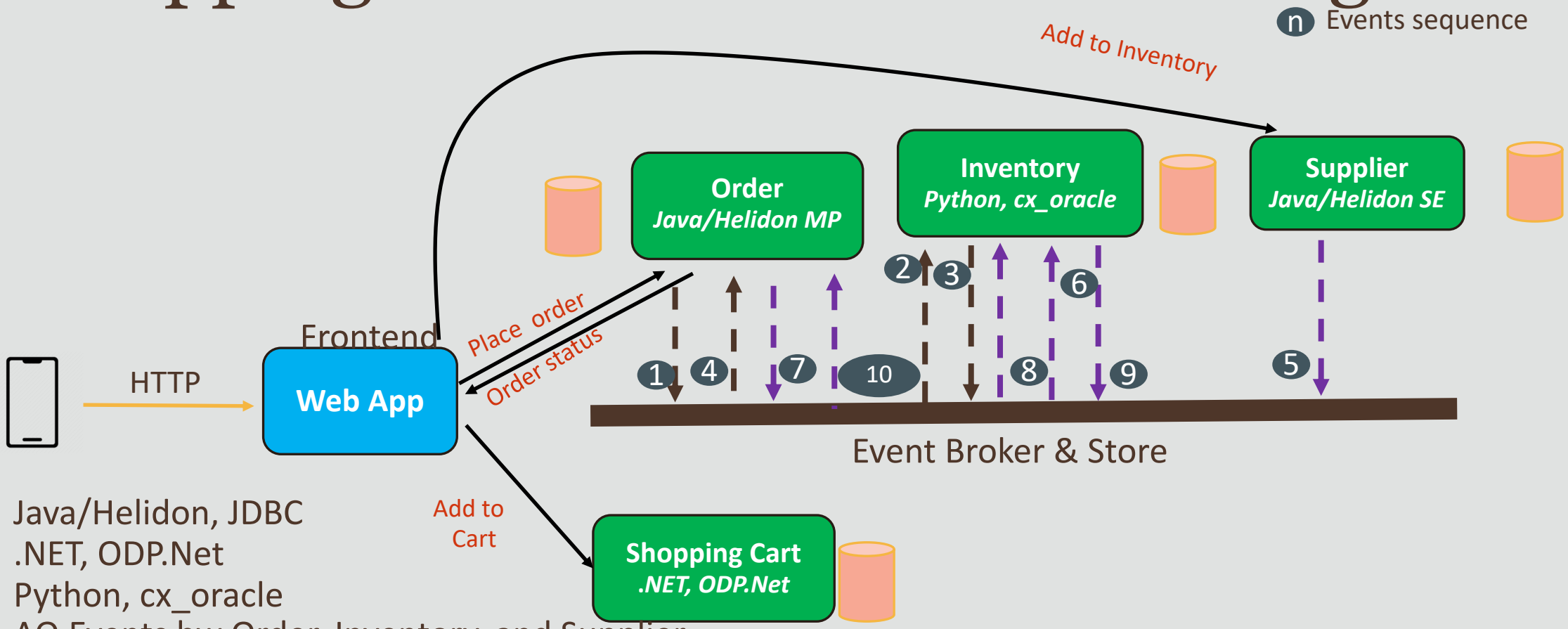
SAGA: Automating Compensation & Escrow

- Automating Compensation
  - Use ‘naturally existing operations: **Add/subtract x**
    - *SQL UPDATE* can be used to specify arithmetic functions
  - **+/-** are inverse to each other and are commutative
    - Multiple sagas can update the same record in parallel
    - Compensation can be automated
- Constraints: Escrow Technology
  - The items in stock, balance of an account, … have to be >= 0
  - The credit limit is $x

Demo



Shopping Demo - Architectural Diagram



- Java/Helidon, JDBC
- .NET, ODP.Net
- Python, cx_oracle
- AQ Events bw Order, Inventory, and Supplier
- REST calls between the Frontend, Shopping cart, Order, and Supplier

Our Other Sessions Today

- **DEV 4692**: A Database Proxy for Transparent HA, Performance, Routing, and Security
  - 9/19 at 12:15PM Moscone South- Room 313
- **CON6865**: Exploring the Multicloud: Working with Azure and Oracle Autonomous Database
  - 9/19 at 1:15PM, Moscone South – Room 209
- **HOL4650** : Build a Stateful Microservices with a Java library, Helidon, and Kubernetes
  - 9/19 at 1:30PM, Moscone West – Room 3019

Thank You

Kuassi Mensah
@kmensah

Q and A