Oracle OpenWorld 2019

SAN FRANCISCO

Copyright © 2019 Oracle and/or its affiliates.

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.

Copyright © 2019 Oracle and/or its affiliates.

Developing and Deploying Oracle Database Applications in Kubernetes

Santanu Datta, Vice President of Development, Oracle
Kuassi Mensah, Director of Product Management, Oracle
Christian Shay, Senior Principal Product Manager, Oracle
Christopher Jones, Senior Principal Product Manager, Oracle

September 2019

Copyright © 2019 Oracle and/or its affiliates.

Agenda

- Microservices Architecture
- Kubernetes and Cloud Native Platform
- Development of Microservices
- Business Transactions
- Demo

Copyright © 2019 Oracle and/or its affiliates.

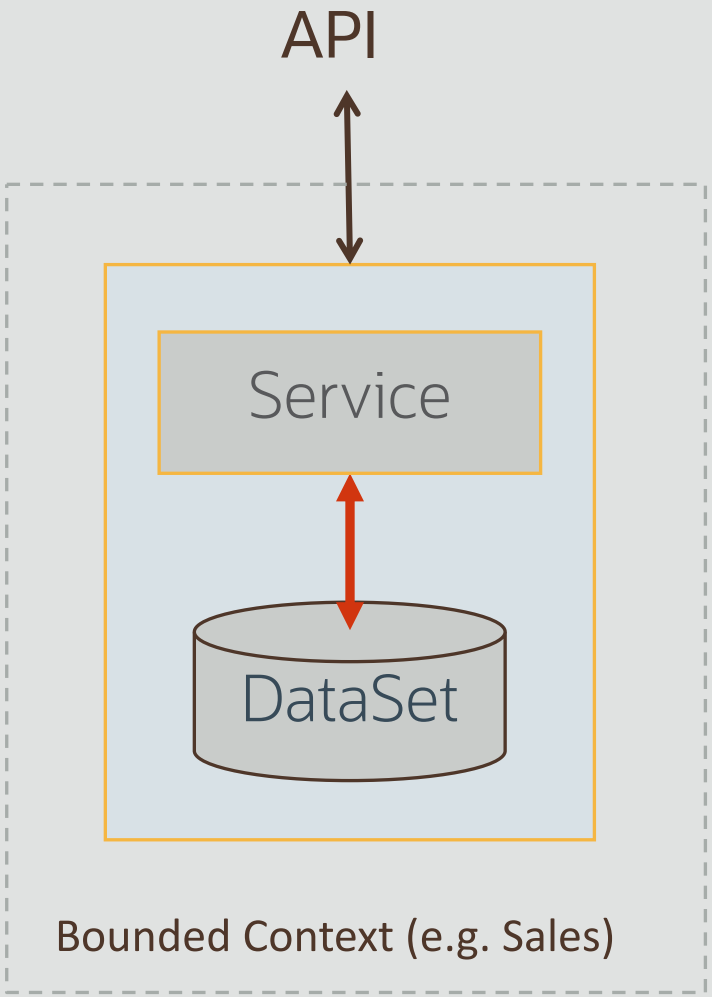
Microservices Architecture

Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

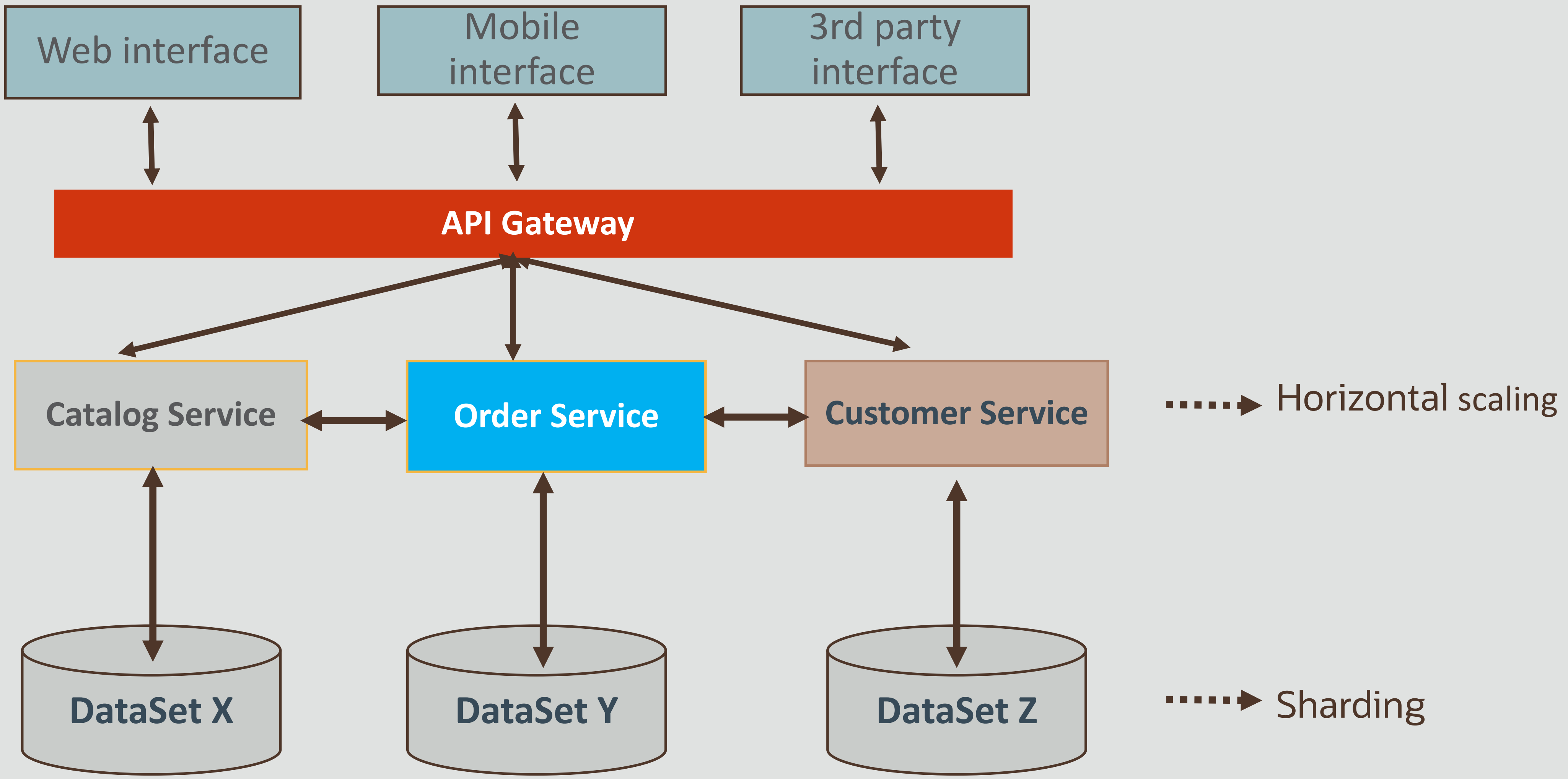
What is a Microservice?

- Well-defined APIs
- Private datasets
- Independent development and deployment
- Bounded Context
- Loose coupling across microservices



Copyright © 2019 Oracle and/or its affiliates.

Microservices Architecture



Copyright © 2019 Oracle and/or its affiliates.

Communication across microservices

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

Copyright © 2019 Oracle and/or its affiliates.

Event Broker

Microservice data sharing pattern

CloudEvents standard

cloudevents

Example CloudEvent…

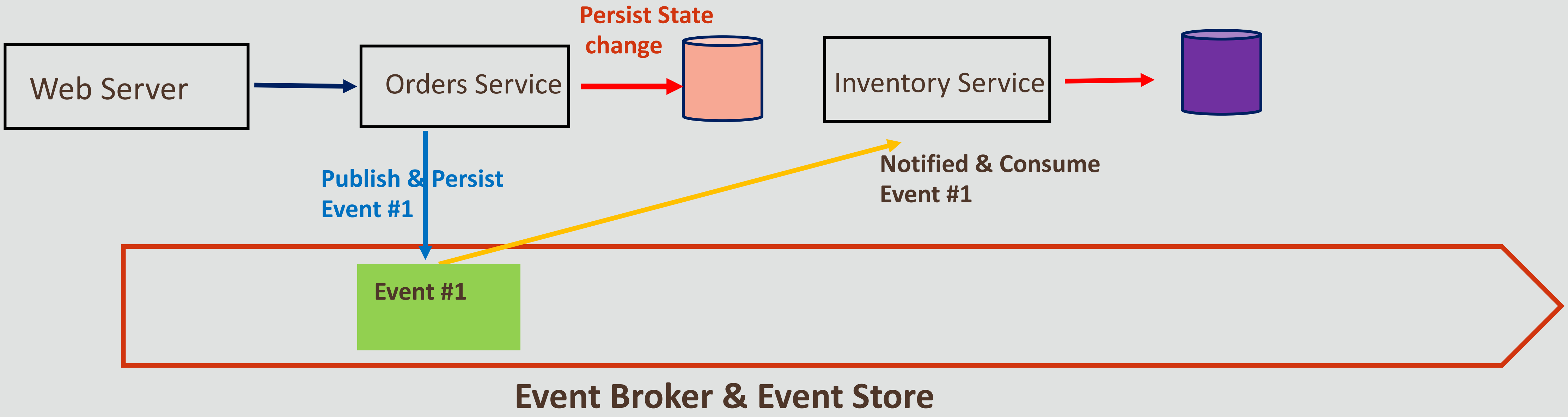

```
{
   "cloudEventsVersion" : "0.1",
   "eventType" : "com.example.someevent",
   "source" : "/mycontext",
   "eventID" : "A234-1234-1234",
   "eventTime" : "2018-04-05T17:31:00Z",
   "comExampleExtension1" : "value",
   "comExampleExtension2" : {
      "otherValue": 5
   },
   "contentType" : "text/xml",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

Copyright © 2019 Oracle and/or its affiliates.

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

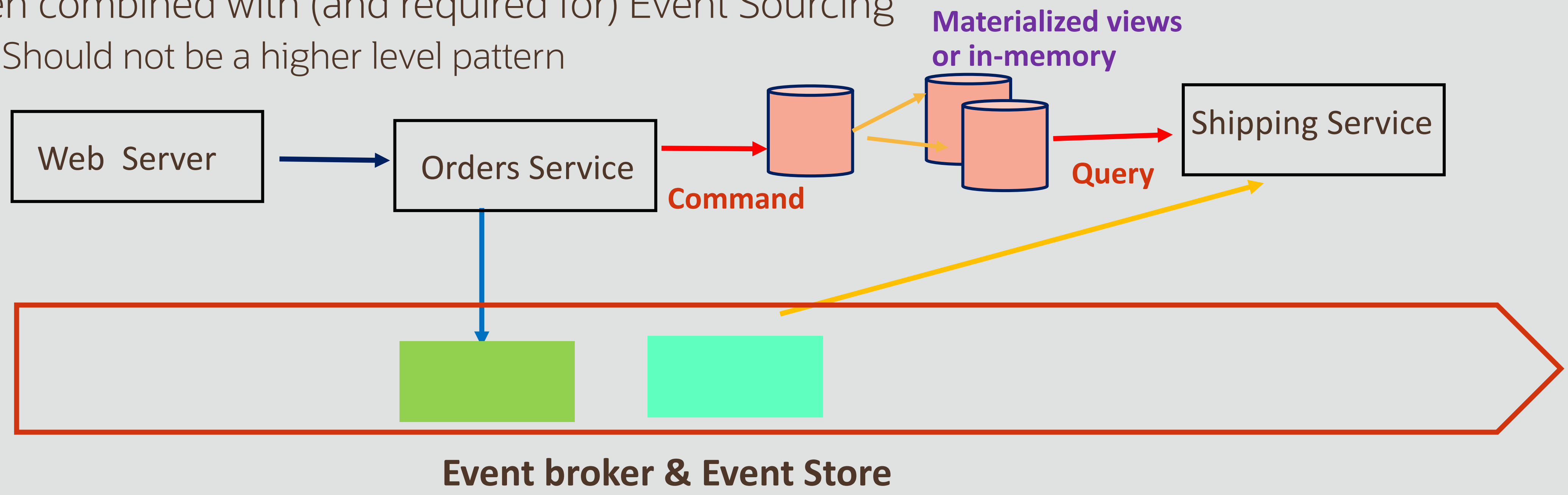


Copyright © 2019 Oracle and/or its affiliates.

Command Query Responsibility Segregation - CQRS

Separation of responsibilities

- Commands: make changes to state/data -- do not query state
- Queries: view state/data (subscribe to data events, scale with materialized views) -- do not change state
- Often combined with (and required for) Event Sourcing
  - Should not be a higher level pattern



Copyright © 2019 Oracle and/or its affiliates.

Microservices: Benefits and Challenges

Benefits

- Freedom to use different technology stacks
- Faster development, deployment and release
- Potential to achieve maximum HA and scalability
- Freedom to choose different data models
- Cloud native model

Challenges

- Increased communication/latency
- Complexity in development
- Tracing and correlation
- Resiliency
- Data consistency
- Data management

Kubernetes

Copyright © 2019 Oracle and/or its affiliates.

Kubnernetes

"Open-source system for automating deployment, scaling, and management of containerized applications"

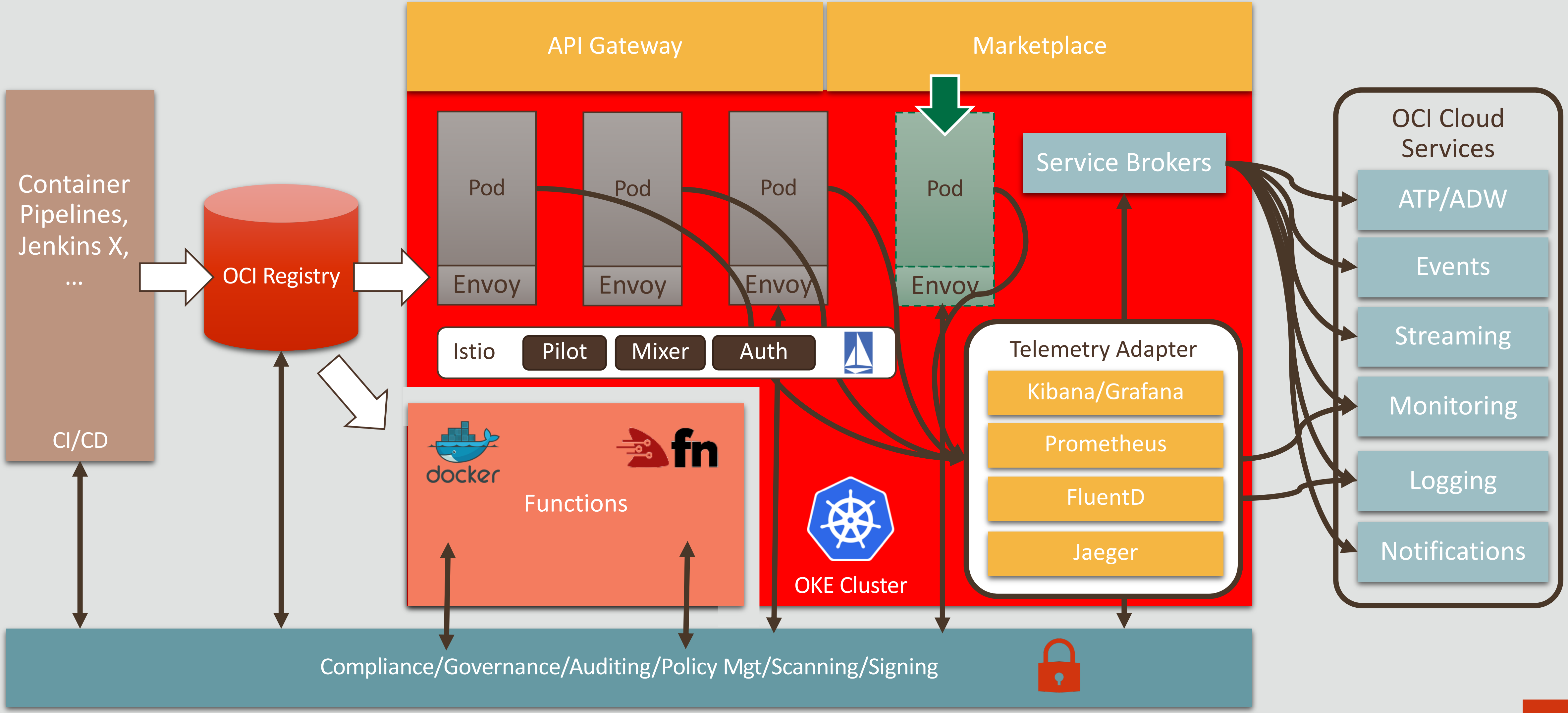
- Planet Scale
- Never Outgrow
- Run Anywhere

Source: Kubernetes.io

Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

Roadmap: Enterprise Container Platform



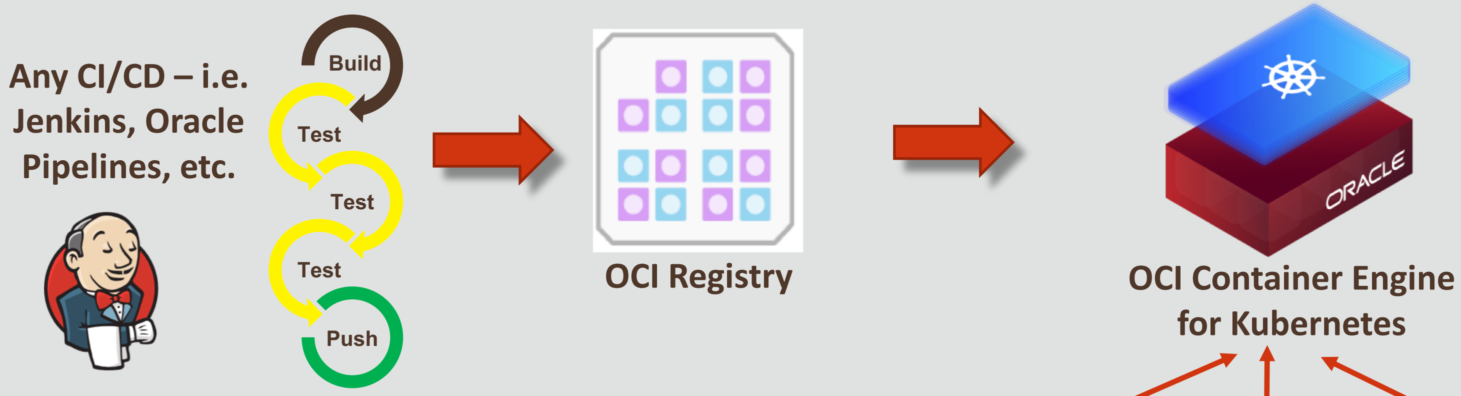
Copyright © 2019 Oracle and/or its affiliates.

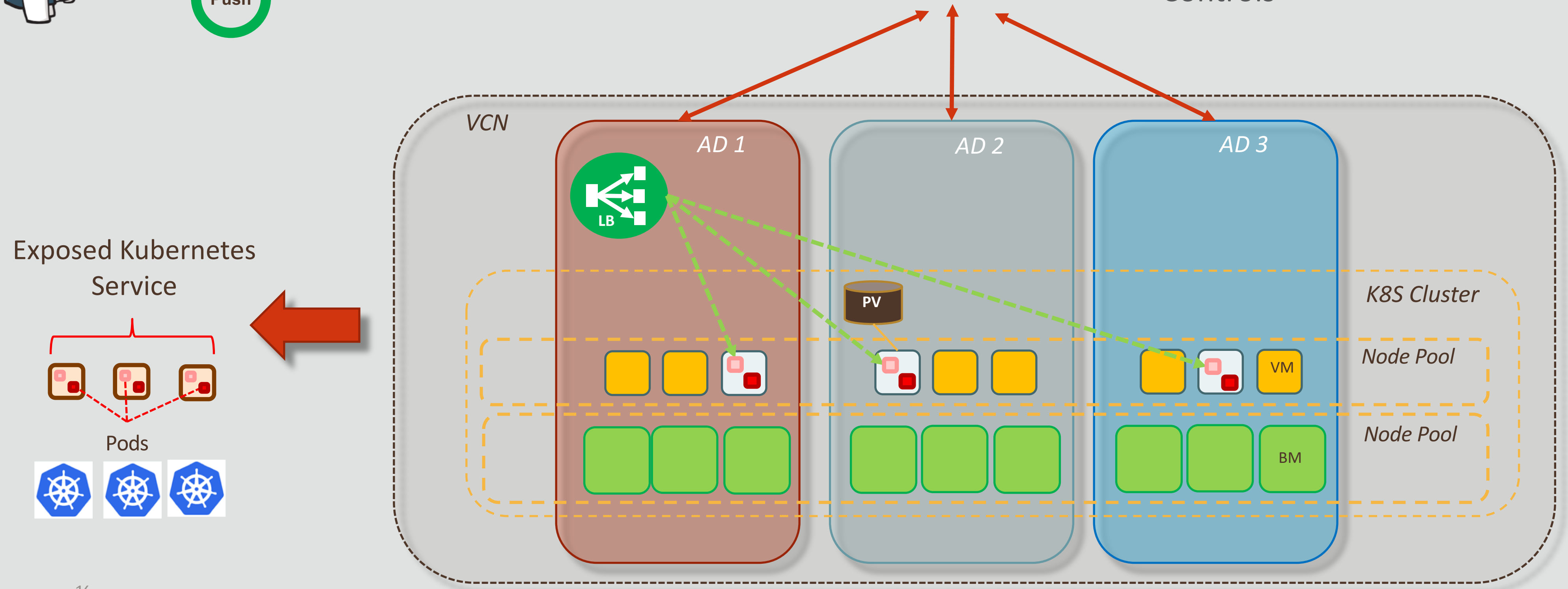
Container Application Life Cycle (OKE)

GA

An Open, Fully-Managed Kubernetes Platform & Private Registry

- **Container Native**: Standard Upstream Kubernetes; Fully Managed Lifecycle; Integrated Registry
- **Developer Friendly**: Simple, Streamlined User Interface; REST API; Helm, and DNS Built-in
- **Enterprise Ready**: Oracle Cloud Infrastructure Performance; Highly Available; Secure with OCI Access Controls



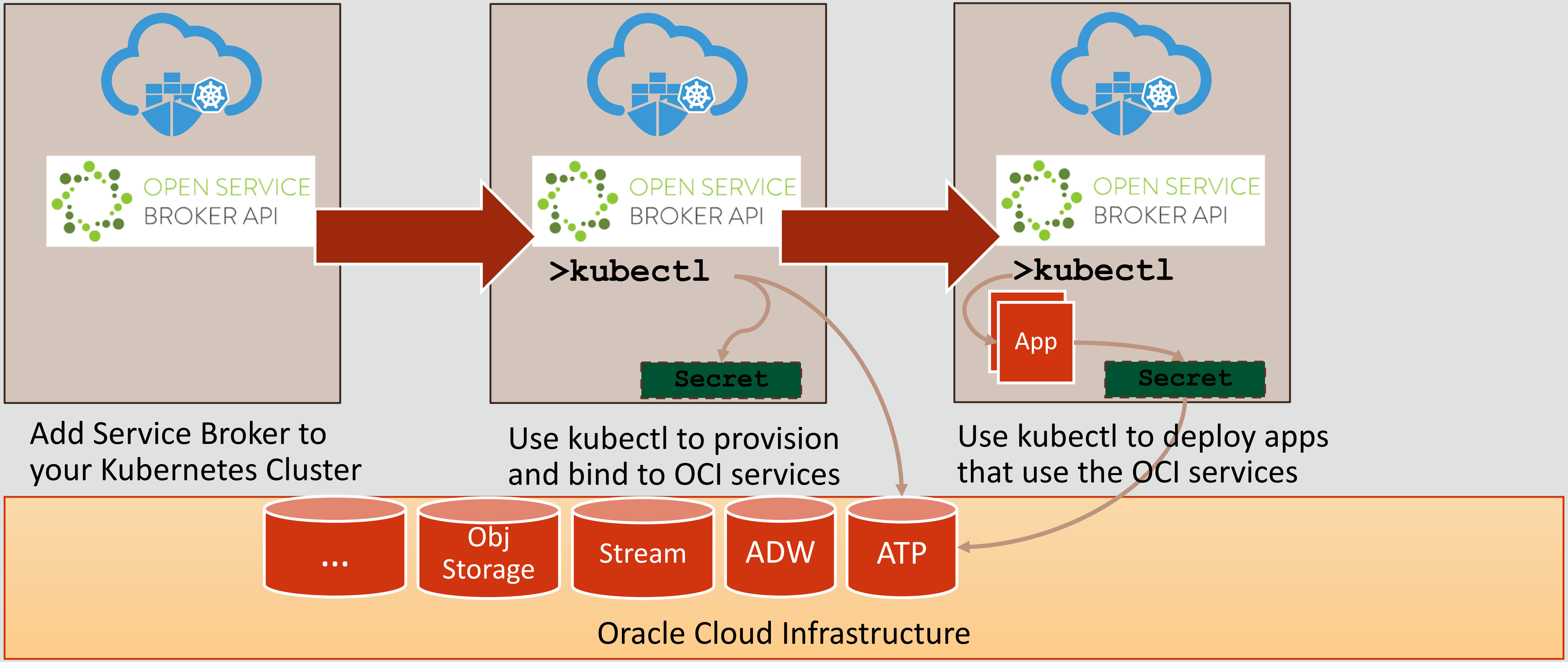


OCI Service Broker for Kubernetes

- Implements Open Service Broker API
- Downloadable **open source** tool to deploy in a Kubernetes cluster
  - https://github.com/oracle/oci-service-broker
- Eases adoption of OCI Cloud Services
  - Provision/binding to OCI cloud services
  - Consolidate DevOps tooling
  - App and cloud resource lifecycle alignment
  - Application portability

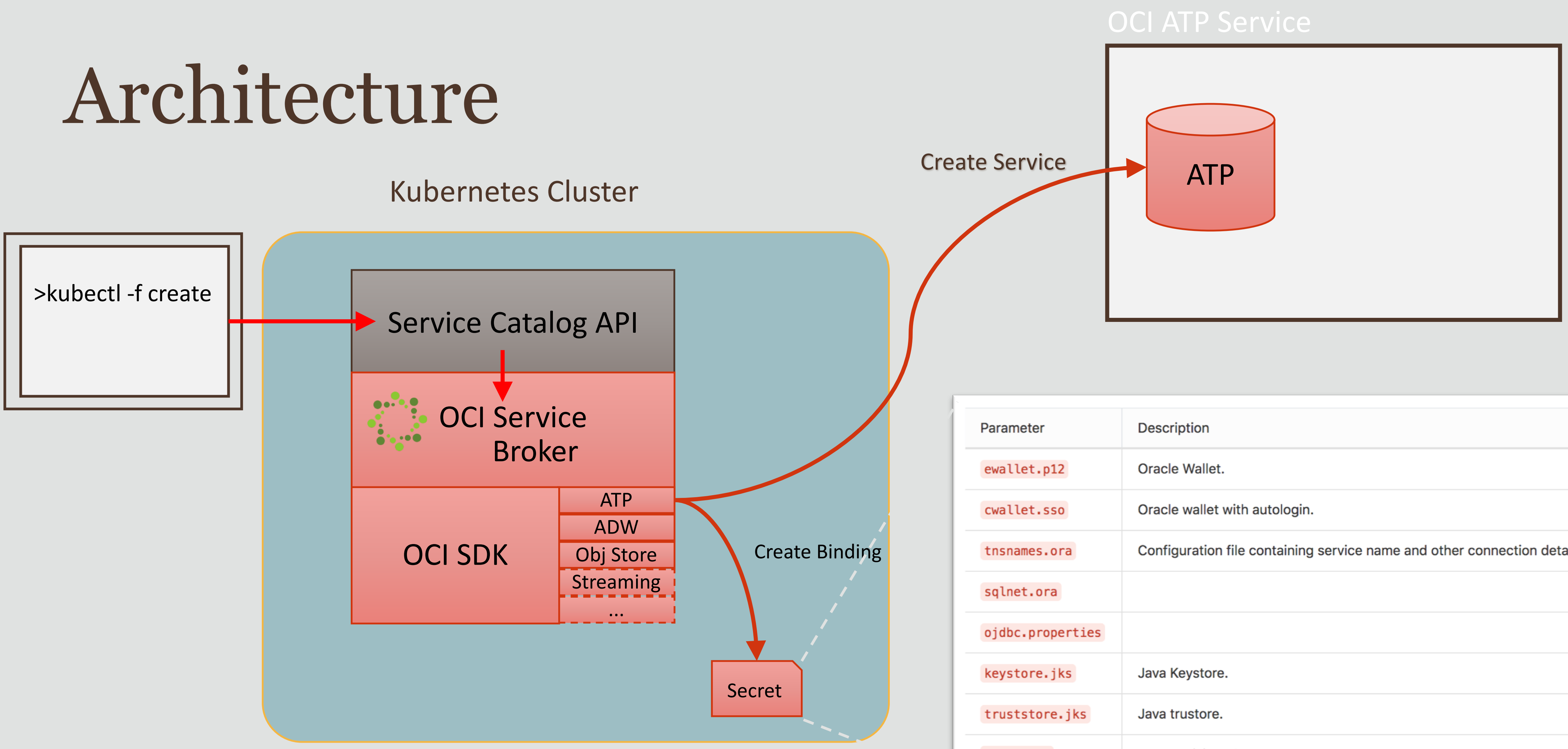
Copyright © 2019 Oracle and/or its affiliates.

OCI Service Broker for Kubernetes



Copyright © 2019 Oracle and/or its affiliates.

Architecture



Parameter	Description	Type
`ewallet.p12`	Oracle Wallet.	string
`cwallet.sso`	Oracle wallet with autologin.	string
`tnsnames.ora`	Configuration file containing service name and other connection details.	string
`sqlnet.ora`		string
`ojdbc.properties`		string
`keystore.jks`	Java Keystore.	string
`truststore.jks`	Java trustore.	string
`user_name`	Pre-provisioned DB ADMIN Username.	string

Secret contents vary depending on the OCI service type

Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

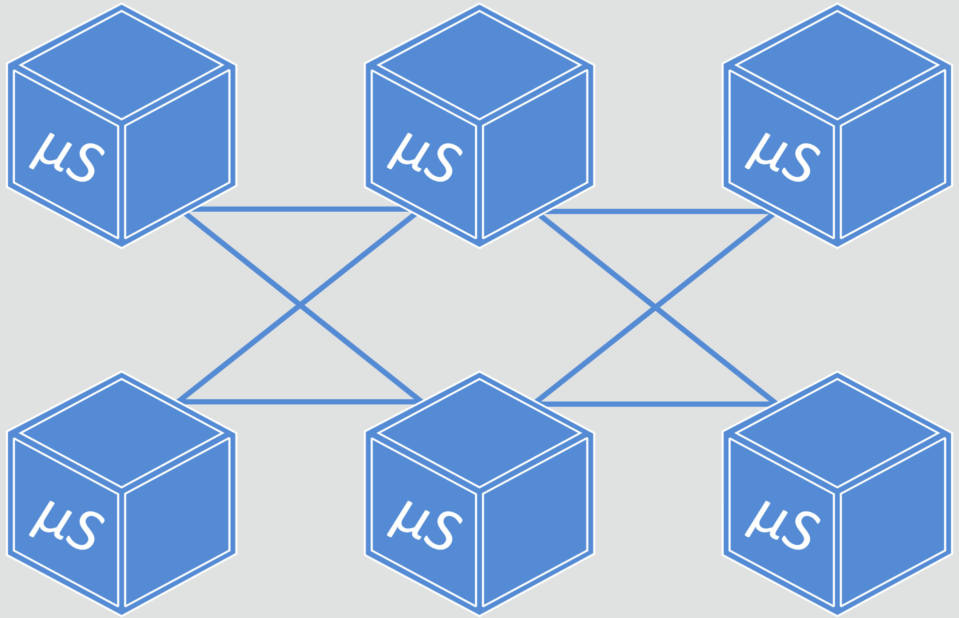
Kubernetes / Microservices

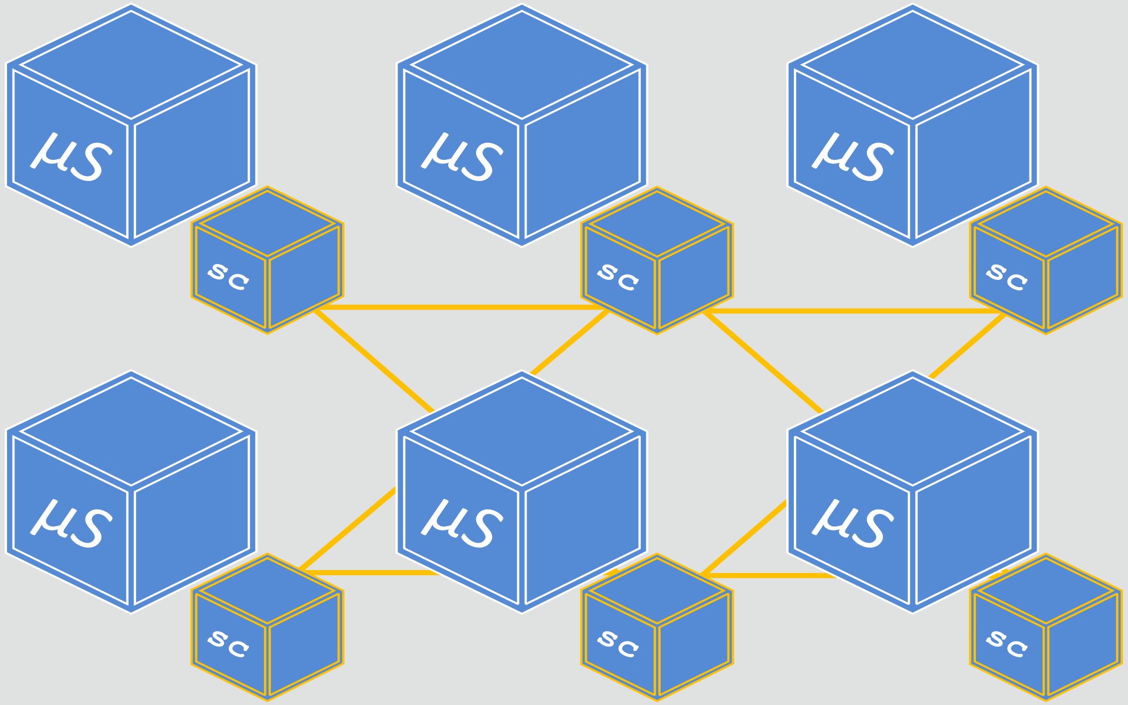
Ideal for microservices…

- Pods, a runnable unit of work (container images developed by different teams into a single deployable unit); usually holds 1 or 2 containers
- Services, an abstraction to define a logical set of Pods and a policy by which to access them
- 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.
- Api-registry: tracks of all available services and the resources they expose.

Copyright © 2019 Oracle and/or its affiliates.

Microservices Platform & Service Mesh





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

Copyright © 2019 Oracle and/or its affiliates.

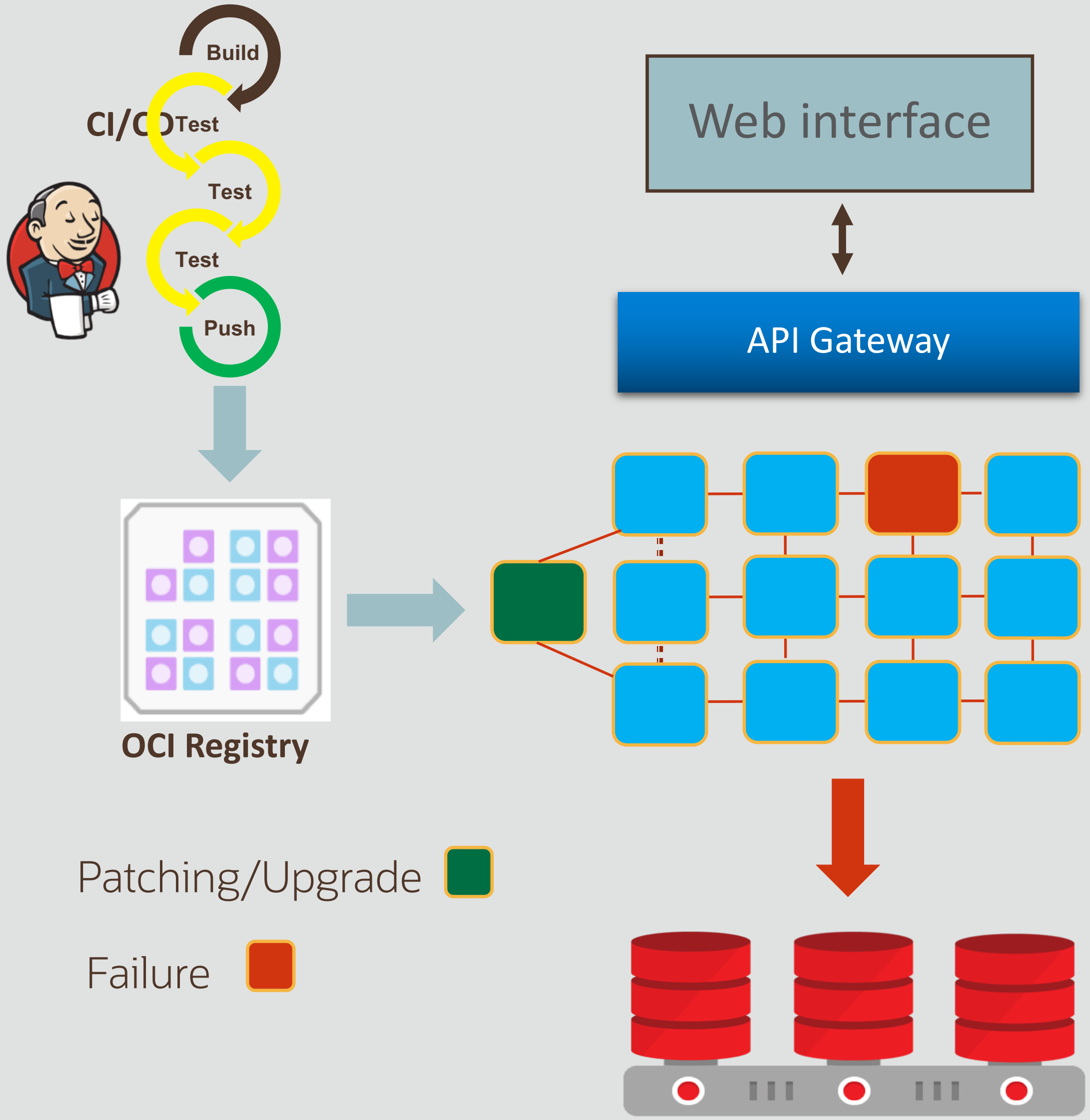
Istio

- Routing Rules
  - blue/green, canary, mirror/shadow
- Security
- Resiliency
  - Retries, timeouts, circuit breaker, fault injection
- Telemetry
  - Tracing, metrics, logs

Development of Microservices

Copyright © 2019 Oracle and/or its affiliates.

Best Practices for Microservices Development



Essential Characteristics

- Domain based separation
- API versioning and backwards-compatibility
- Readiness and health check
- Retries and timeouts
- Idempotency
- Asynchronous interactions
- End-to-end tracing
- Restartable/recoverable batch jobs
- Zero downtime Database Connectivity
- Business Transactions across microservices

Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

The State of Java

Frameworks for Writing Microservices

What is Helidon?

- Open source Java libraries for writing microservices
- Small and fun to program in
- Provides all that good Cloud Native stuff
  - REST, Metrics, Health, Tracing, Security, Configuration, Docker
- Built on top of Netty
- Just a Java SE application – no App Server!
- Comes in two API flavors:

helidon SE helidon MP

Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

helidon SE

- Reactive Microframework
- Tiny Footprint
- Functional style
- Simple and transparent
- GraalVM Native Image

```
Routing routing = Routing.builder()
    .get("/hello", (req, res) ->
        res.send("Hello World"))
    .build();

WebServer.create(routing)
    .start();
```

helidon MP

- MicroProfile 3.0
- Small Footprint
- Declarative style
- Dependency Injection
- CDI, JAX-RS, JSON-P/B

```
@Path("hello")
public class HelloWorld {
    @GET
    public String hello() {
        return "Hello World";
    }
}
```

Zero downtime Database Connectivity

Key Technologies

- Planned Maintenance
  - Oracle Connection/Session Pool
  - Leverage FAN/FCF
  - Transactional Disconnect with Transaction Guard (TG) and Transparent Application Failover (TAF)
- Unplanned Outages
  - Transparent Application Continuity
  - Retries and Timeouts

Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

Zero downtime Database Connectivity

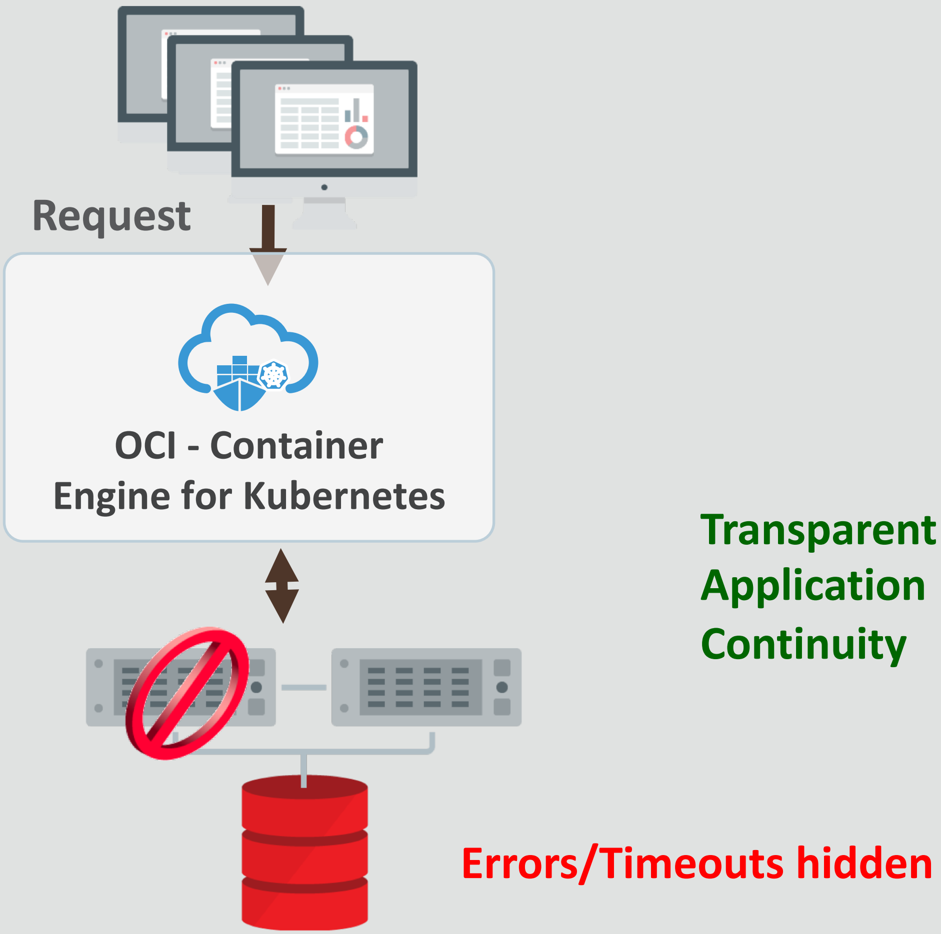
Planned Maintenance

- Drain work from target instance
  - Supports well behaved applications using Oracle pools: ODP.NET, OCI Session Pool, Python, . . ., WebLogic Active GridLink, UCP
  - Connections migrate away from the target instance when request completes
- Transactional Disconnect with Transaction Guard (TG) and Transparent Application Failover (TAF)
  - Applications with one COMMIT per request
  - Connection terminated at transaction boundary; TAF replaces it with new connection

Zero downtime Database Connectivity

Unplanned Outages

Transparent Application Continuity (TAC)



Uses Application Continuity and Oracle Real Application Clusters

Transparently tracks and records session information in case there is a failure

Built inside of the database, so it works without any application changes

Rebuilds session state and replays in-flight transactions upon unplanned failure

Planned maintenance can be handled by TAC to drain sessions from one or more nodes

Adapts as applications change: protected for the future

Copyright © 2019 Oracle and/or its affiliates.

Secure Deployment

- Application
  - Use hardened images and keep updated with latest patches
  - Use TLS for communication
  - Control Network Ingres and Egress rules
  - RBAC
  - Use Kubernetes secrets or other secure mechanisms
  - Rate Limits

- Autonomous Database
  - Use wallets and TLS 1.2
  - Restrict database access to specific IP addresses

Business Transactions

Copyright © 2019 Oracle and/or its affiliates.

Business Transactions across Microservices

- Helidon Long Running Activities
- Data Models
- Leverage AQ for atomic Event + RDBMS updates
- SAGA

Copyright © 2019 Oracle and/or its affiliates.

Long Running Activities
(LRA: github.com/eclipse/microprofile-lra)

Annotation	Description
@LRA	Controls the life cycle of an LRA.
@Compensate	Indicates that the method should be invoked if the LRA is cancelled.
@Complete	Indicates that the method should be invoked if the LRA is closed.
@Forget	Indicates that the method may release any resources that were allocated for this LRA.
@Leave	Indicates that this class is no longer interested in this LRA.
@Status	When the annotated method is invoked it should report the status.

Copyright © 2019 Oracle and/or its affiliates.

Multi Model Data: Two Technologies Approaches

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

- Relational Database
- Hie…
- Key / Valu… Sto…
- Graphs
- …ocs …OC,…)
- Spatial Data

Who does the Integration???
Who does the multitude of administration tasks???
Who is prepared to use XA frequently???

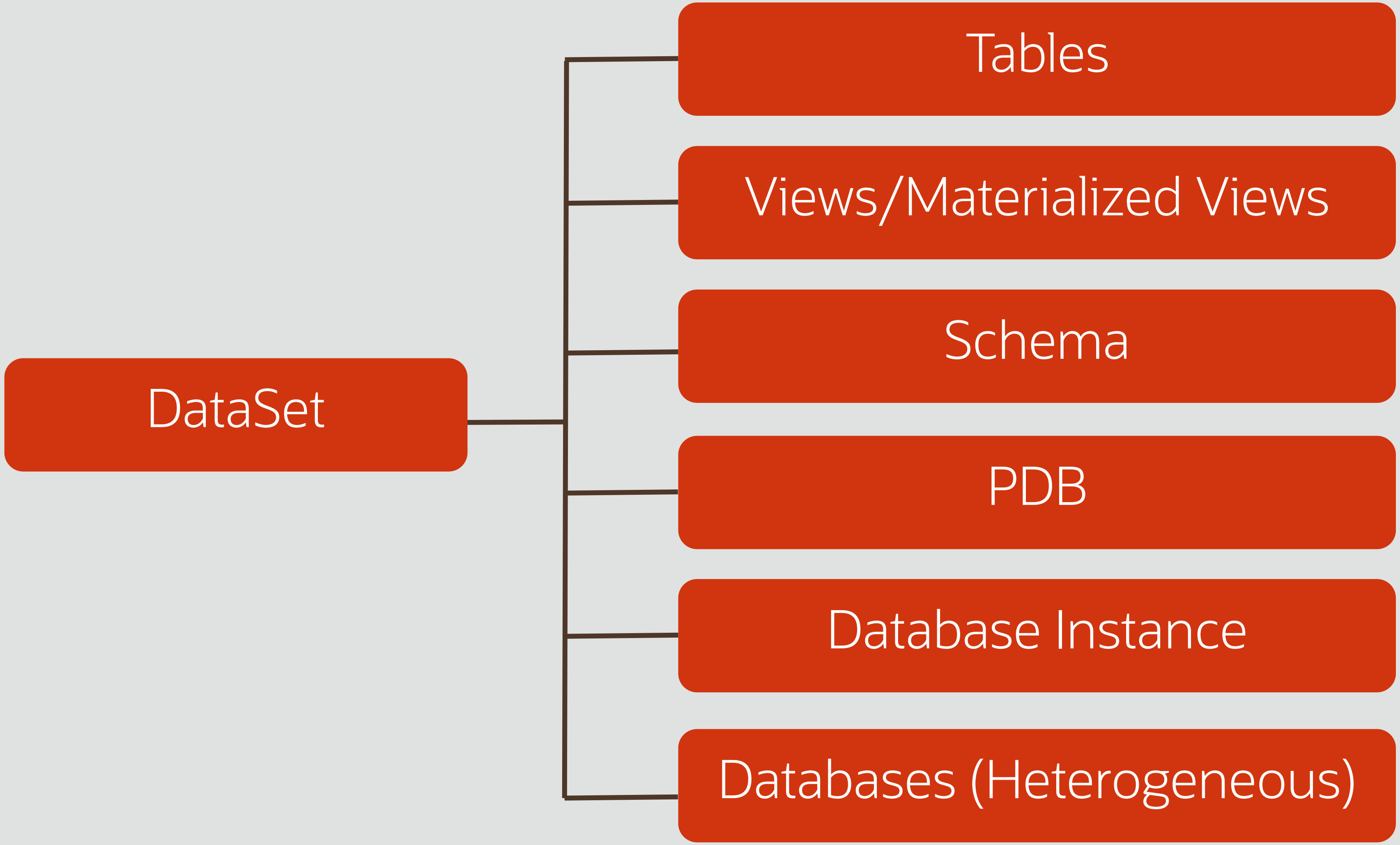
Multi-model database engines
(Single database for all data models)

- Relational data
- Hierarchies (JSON/XML)
- Key / Value Store
- Graphs
- Free Text and Docs (PDF, DOC,…)
- Spatial data

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

Copyright © 2019 Oracle and/or its affiliates.

Data Models

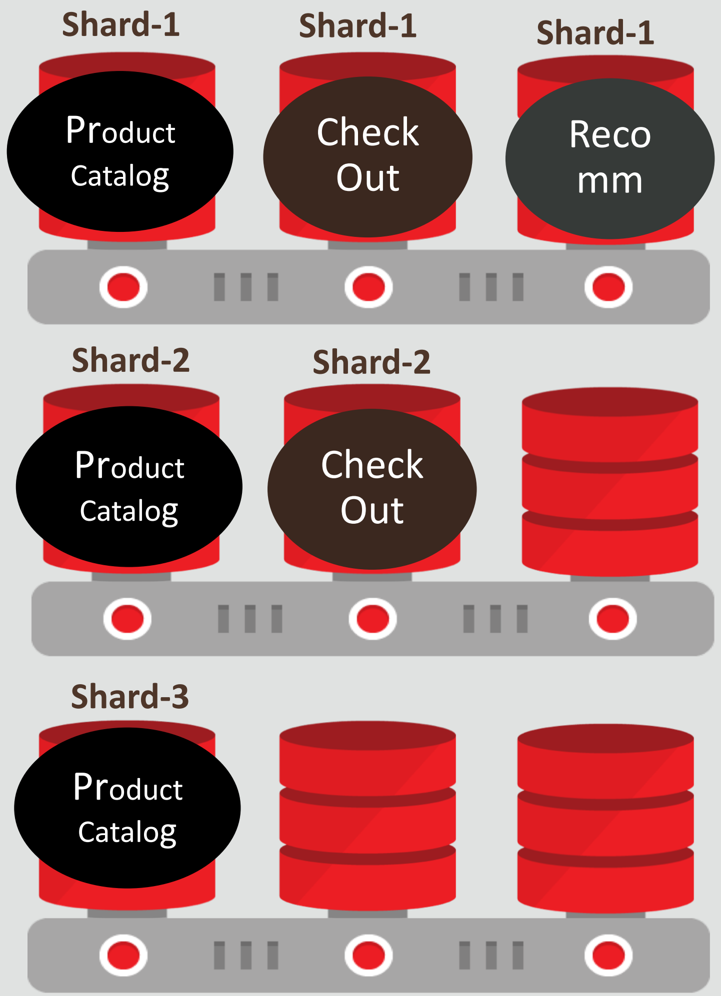


Copyright © 2019 Oracle and/or its affiliates.

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).

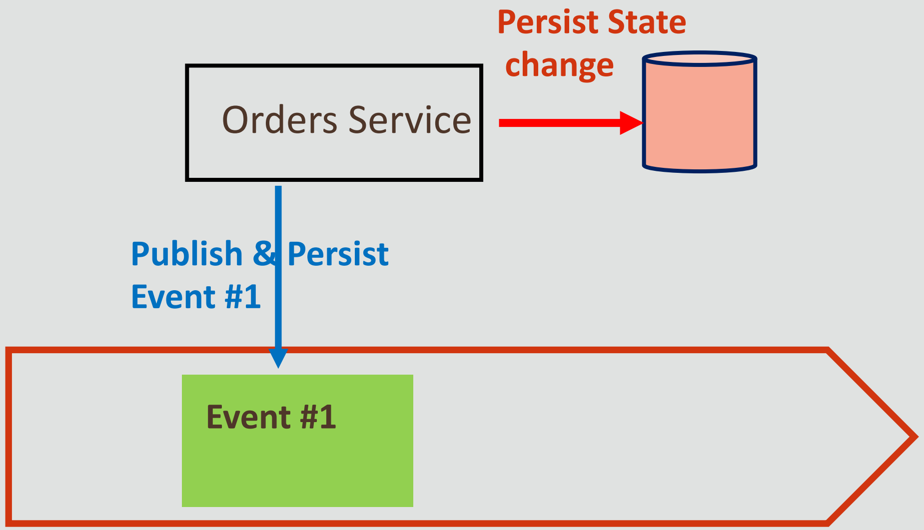


Copyright © 2019 Oracle and/or its affiliates.

Atomicity of Persisting Events and Database States

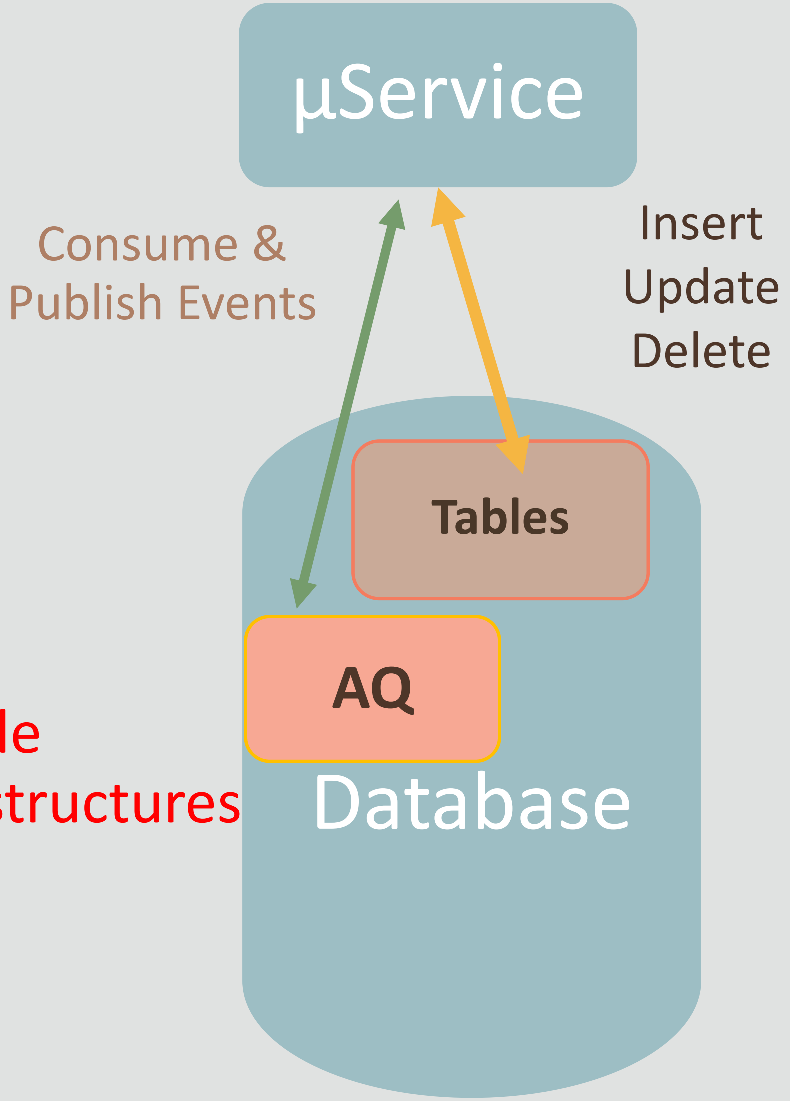
Several Techniques

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)



Copyright © 2019 Oracle and/or its affiliates.

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

Copyright © 2019 Oracle and/or its affiliates.

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
- 16 year history

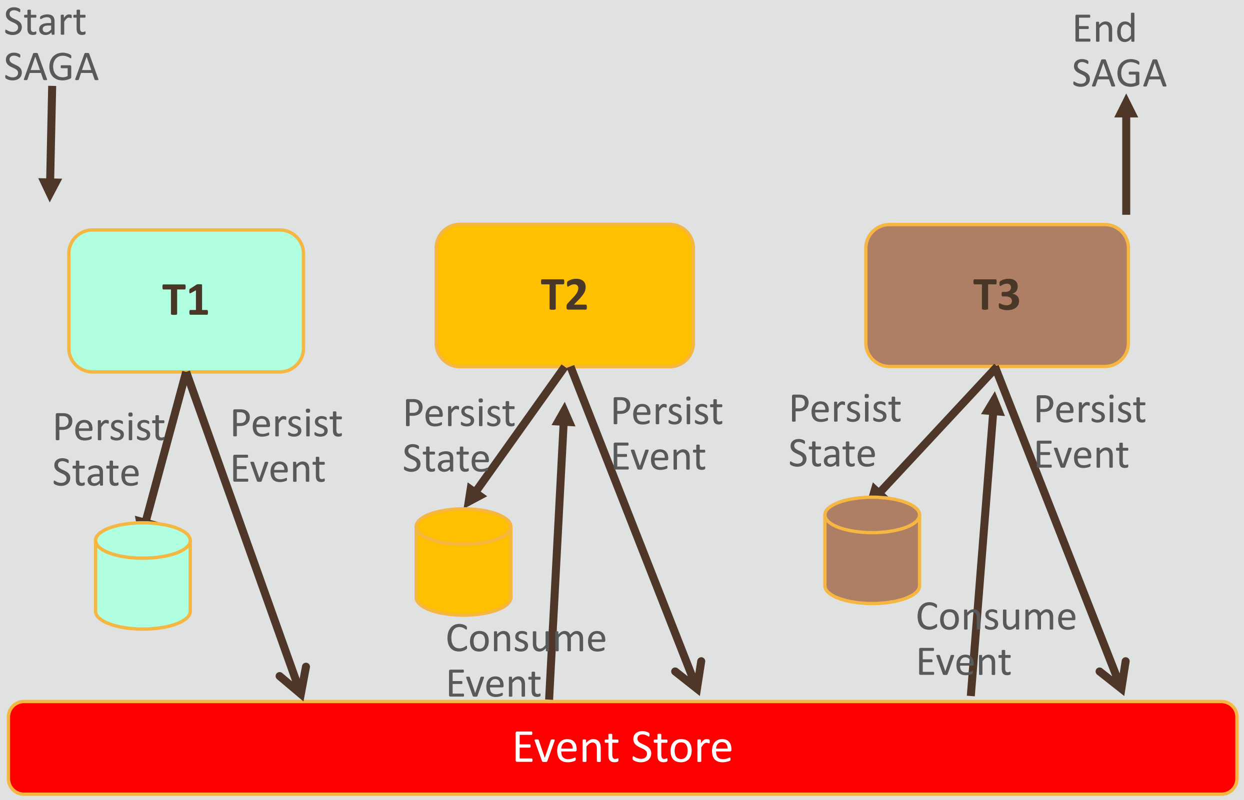
SAGA Pattern and Workflow

"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



Copyright © 2019 Oracle and/or its affiliates.

Copyright © 2019 Oracle and/or its affiliates.

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?

Copyright © 2019 Oracle and/or its affiliates.

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

Copyright © 2019 Oracle and/or its affiliates.

Resources

[DEV1730] Thursday September 19, 9 am, Moscone South 308
Microservices Essentials: Kubernetes and Ecosystem, Data, and Transaction Patterns
[DEV4625] Thursday September 19, 11:15 am, Moscone South 306
Scalability and High Availability for Oracle Database Python Applications
[CON4641] Thursday, September 19, 11:15 am, Moscone South 152B
Oracle Net Services: Best Practices for Database Performance and High Availability
[DEV4692] Thursday September 19 12:15, Moscone South 302
A Database Proxy for Transparent HA, Performance, Routing and Security
[CON6685] Thursday September 19 1:15, Moscone South 209
Exploring the Multicloud: Working with Azure and Oracle Autonomous Database

Email: santanu.datta@oracle.com

Copyright © 2019 Oracle and/or its affiliates.