# Oracle Transaction Manager for Microservices

An Overview

**Deepak Goel, Todd Little, Brijesh Deo**
Oracle Database Product Development

May 27, 2022

# oradev22

## Agenda

1. **Microservices Development Trends & Challenges**
2. Oracle's Strategy for Microservices Transaction Management
3. Transaction Manager Architecture and Features
4. Demo: Transaction Manager for Microservices
5. Roadmap and Upcoming Features

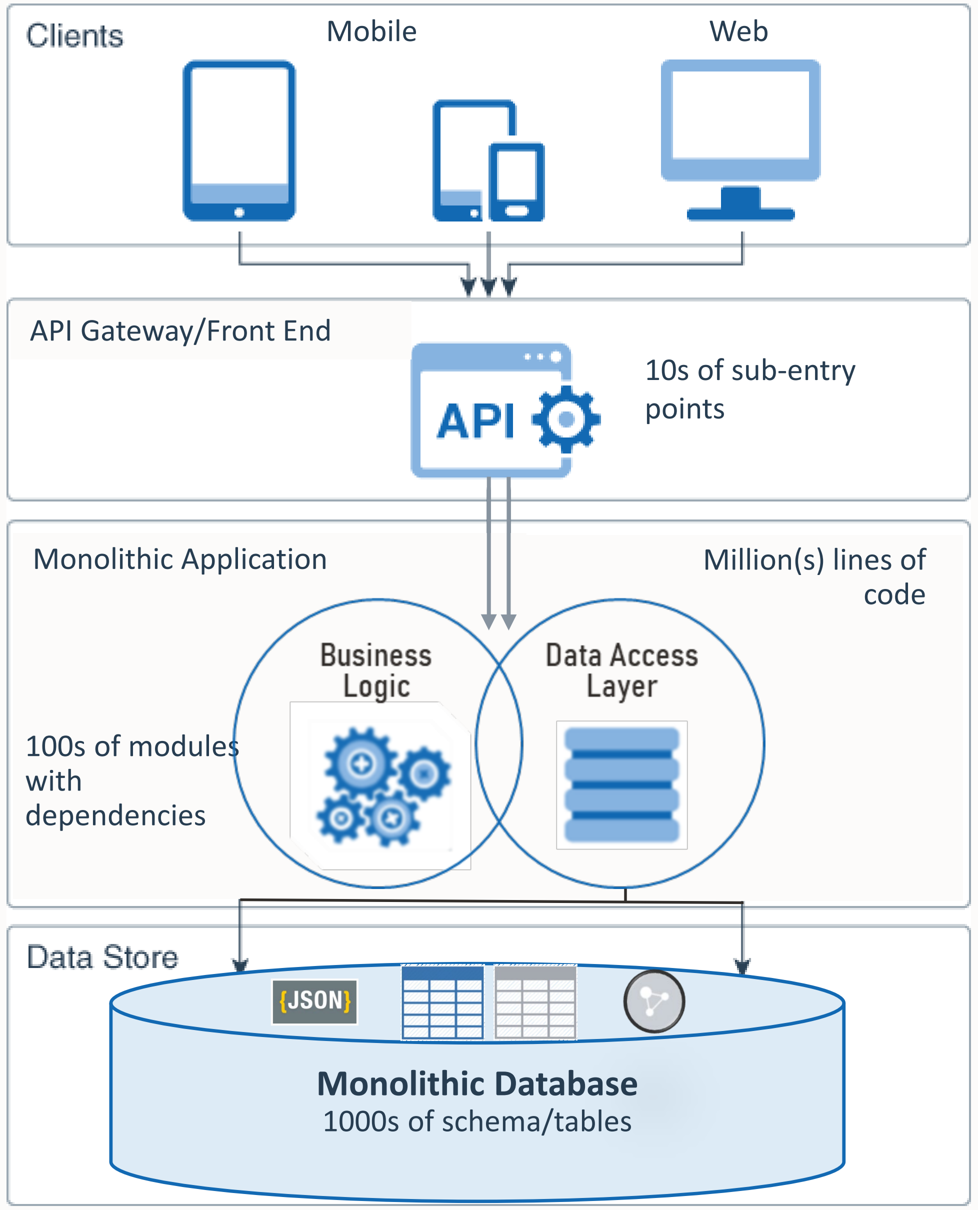
# Monolithic Application

Pros

- Local transactions
- Monolithic database
- Centralized DB admin

Cons/Challenges

- Forest of (hidden) dependencies
- Millions lines of code
- Hundreds/thousands of tables
- Hard to maintain and debug
- Expensive to scale
- Long release cycles



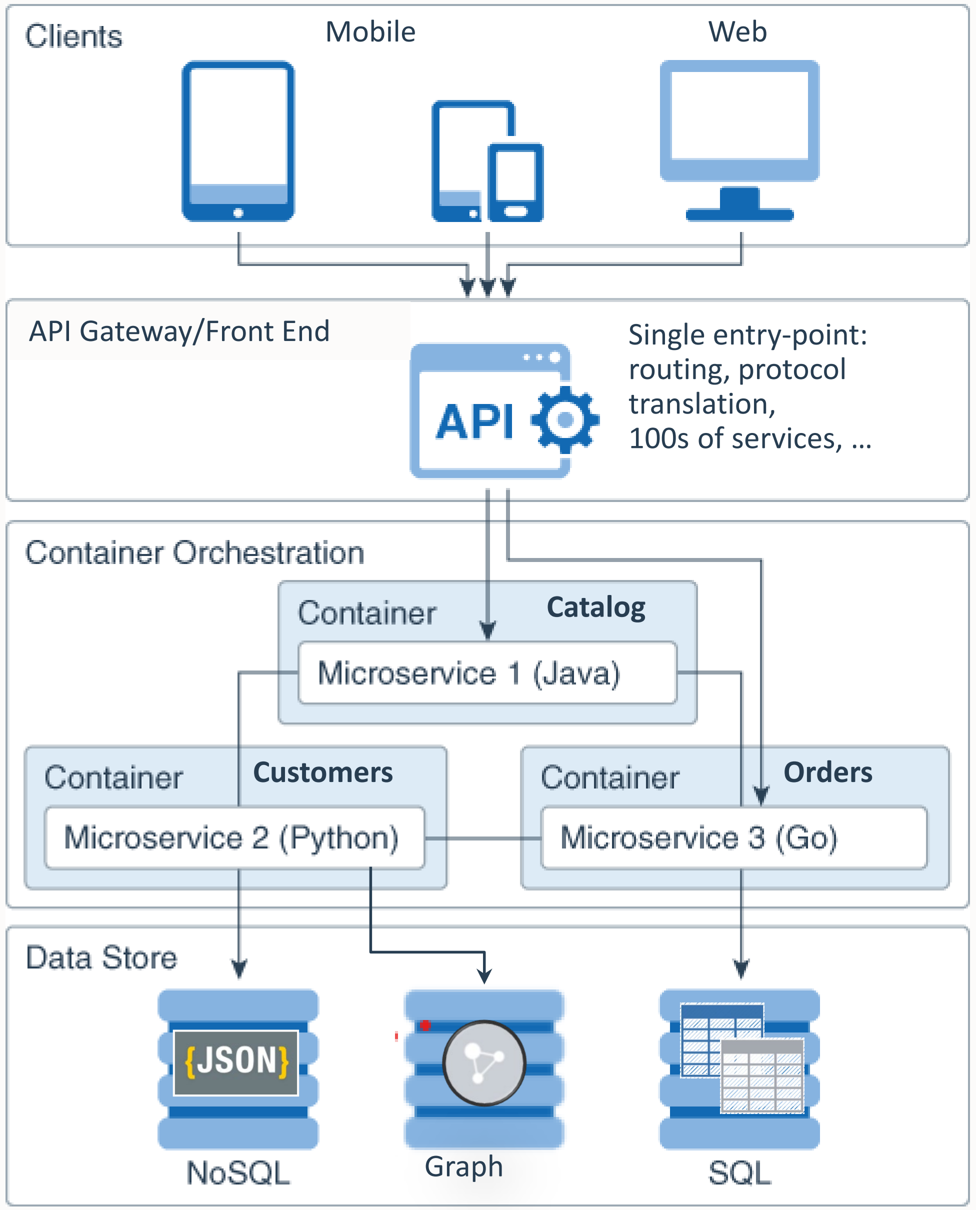
# Microservices-based Application

## Pros

- Independent Units
- Ease of Dev/Test/Deploy
- Polyglot development for best fit
- Ease of maintenance
- Scale independently at service level
- Higher resilience/fault tolerance
- Default Cloud architecture pattern

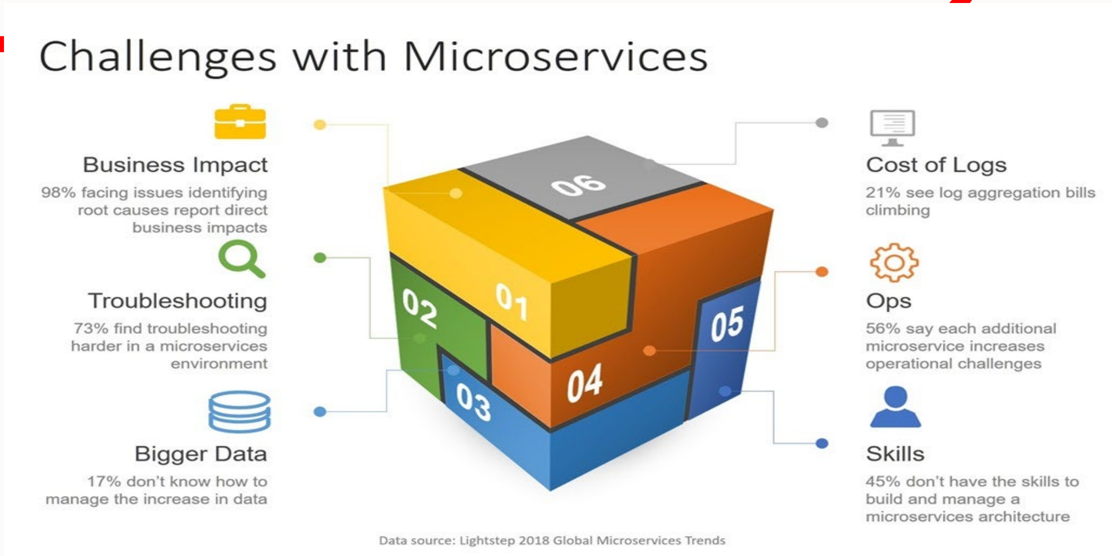
## Cons/Challenges

- Orchestration complexity
- Managing multiple data stores
- Greater latencies



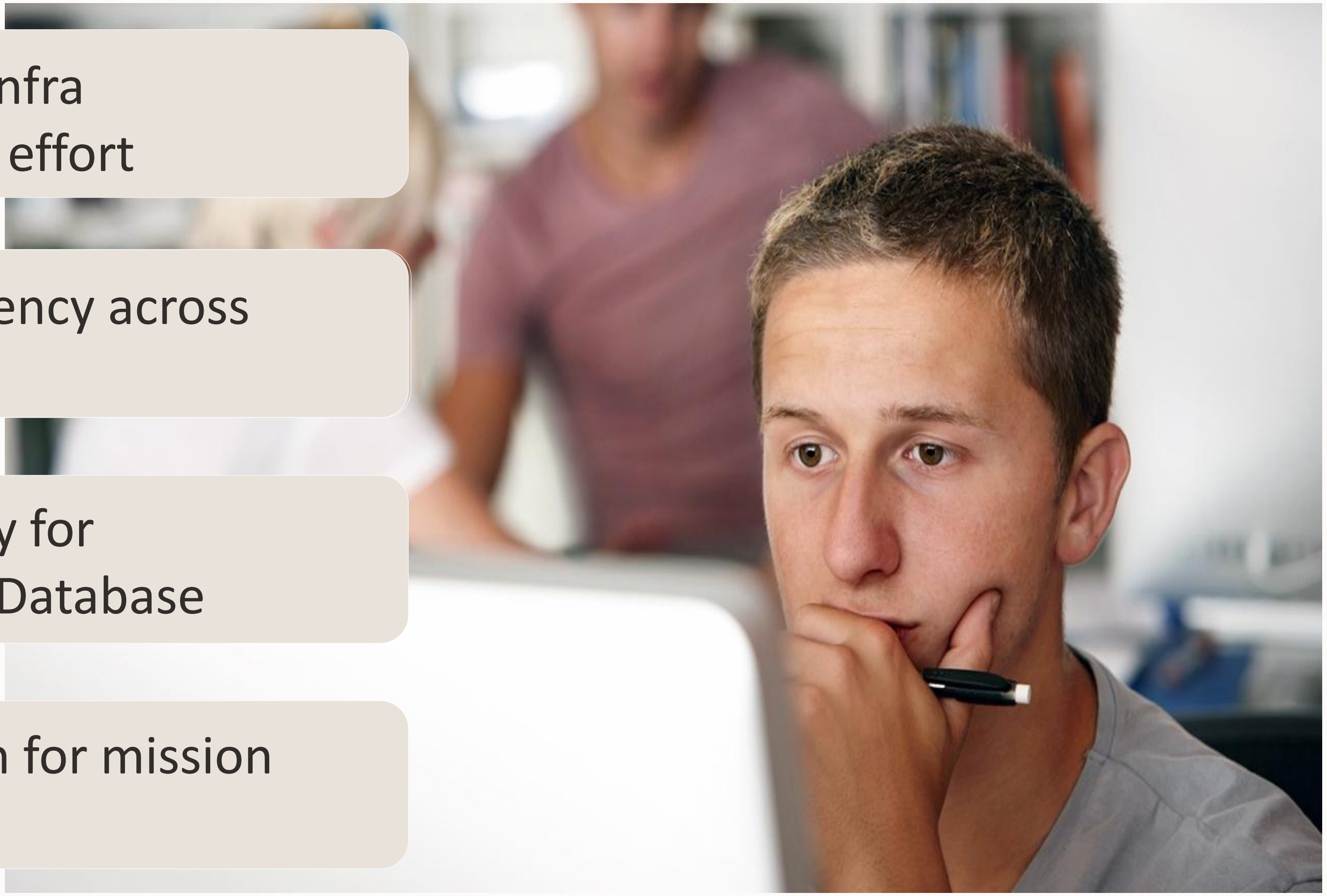
## Challenges In Using Microservices

- Trading scalability for consistency by avoiding reliance on ACID transactions
- Handling transactions that span multiple services is complex
  - e.g., Saga pattern with eventual consistency, orchestration, and compensating transactions coded in application – can be error-prone and requires more advanced skills
  - Testing complexity increases rapidly
- Use of separate DBs per microservice increases data volumes and complicates data management
  - Ability to leverage deep database integration capabilities for availability, scalability, load-balancing, recoverability, etc.
- Polyglot applications often relying on varied frameworks with inconsistent capabilities
- Increased communications and latency
- More difficult to troubleshoot and manage operations
- Cost and complexity of log aggregation
- Skills necessary to build and manage


Challenges with Microservices
Business Impact
98% facing issues identifying root causes report direct business impacts
Troubleshooting
73% find troubleshooting harder in a microservices environment
Bigger Data
17% don't know how to manage the increase in data
Cost of Logs
21% see log aggregation bills climbing
Ops
56% say each additional microservice increases operational challenges
Skills
45% don't have the skills to build and manage a microservices architecture
01
02
03
04
05
06
Data source: Lightstep 2018 Global Microservices Trends

## Agenda

## Product Strategy



Reduce cost by providing more infra capabilities and by reducing dev effort

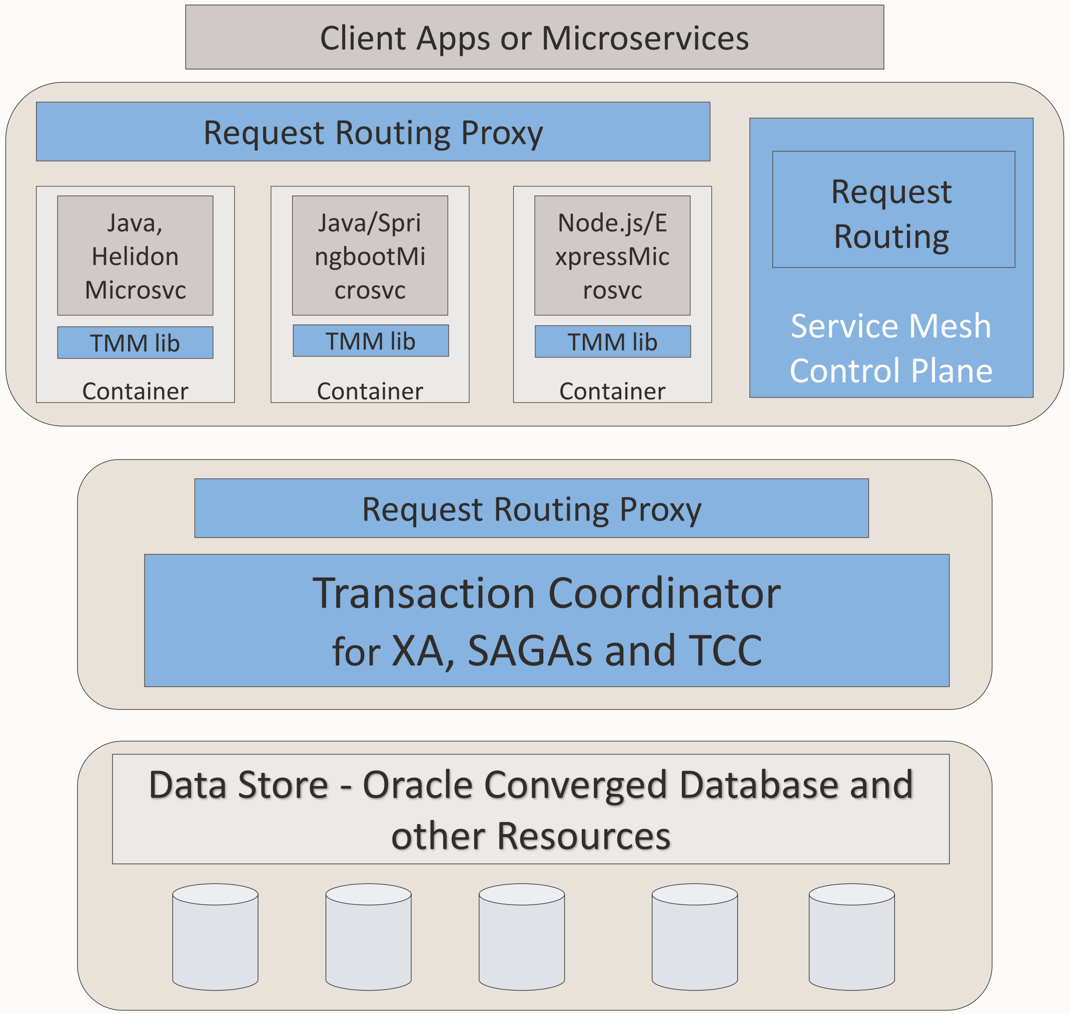
Enable various levels of consistency across polyglot microservices

Improve availability and scalability for microservices when using Oracle Database

Provide enterprise grade solution for mission critical microservices apps

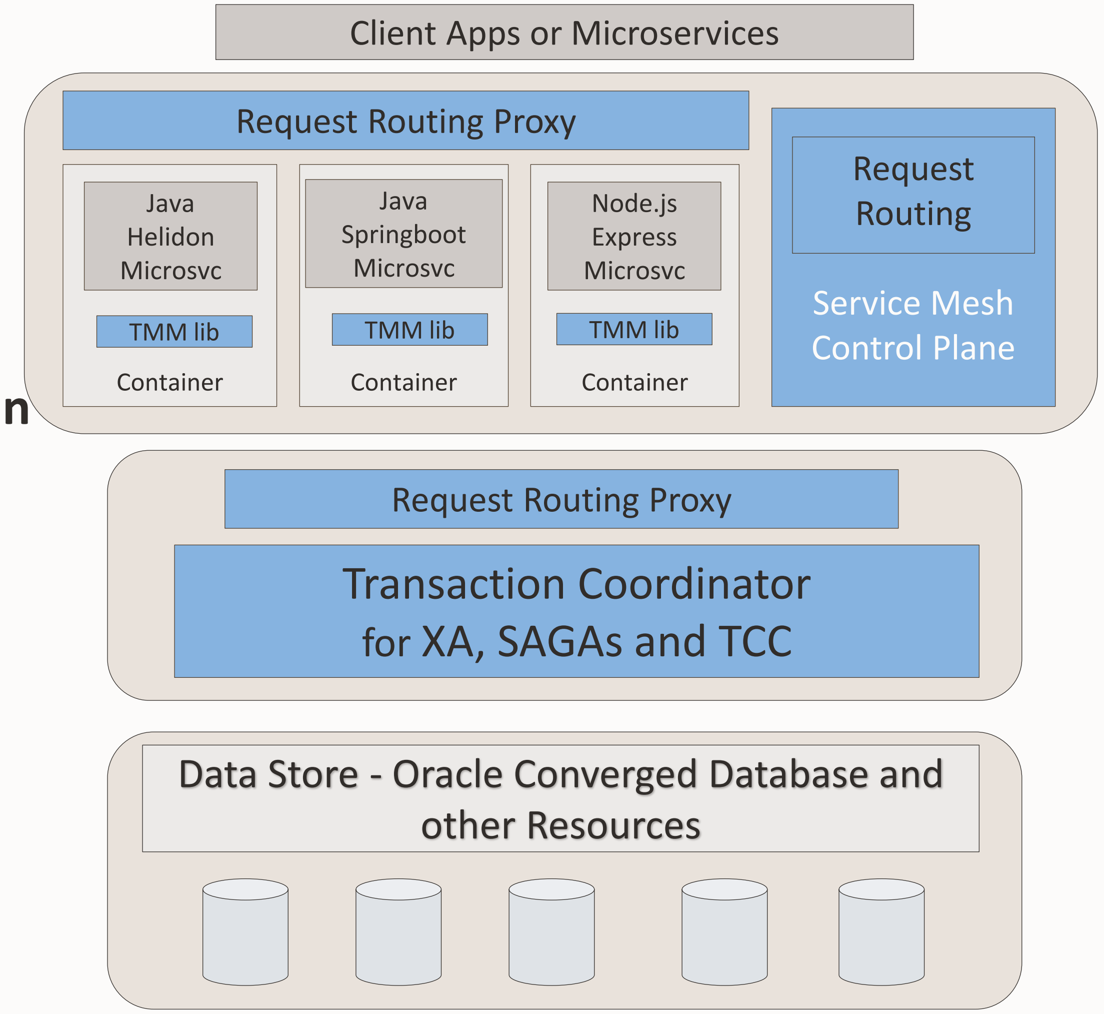
# Oracle Transaction Manager for Microservices

- Provides a spectrum of consistency models across polyglot microservices by supporting distributed transaction patterns
  - Strong consistency thorugh XA and Try-Confirm/Cancel transactions
  - Eventual consistency through saga orchestration using Eclipse Microprofile Long Running Actions
  - Support for both sync and async service communication models

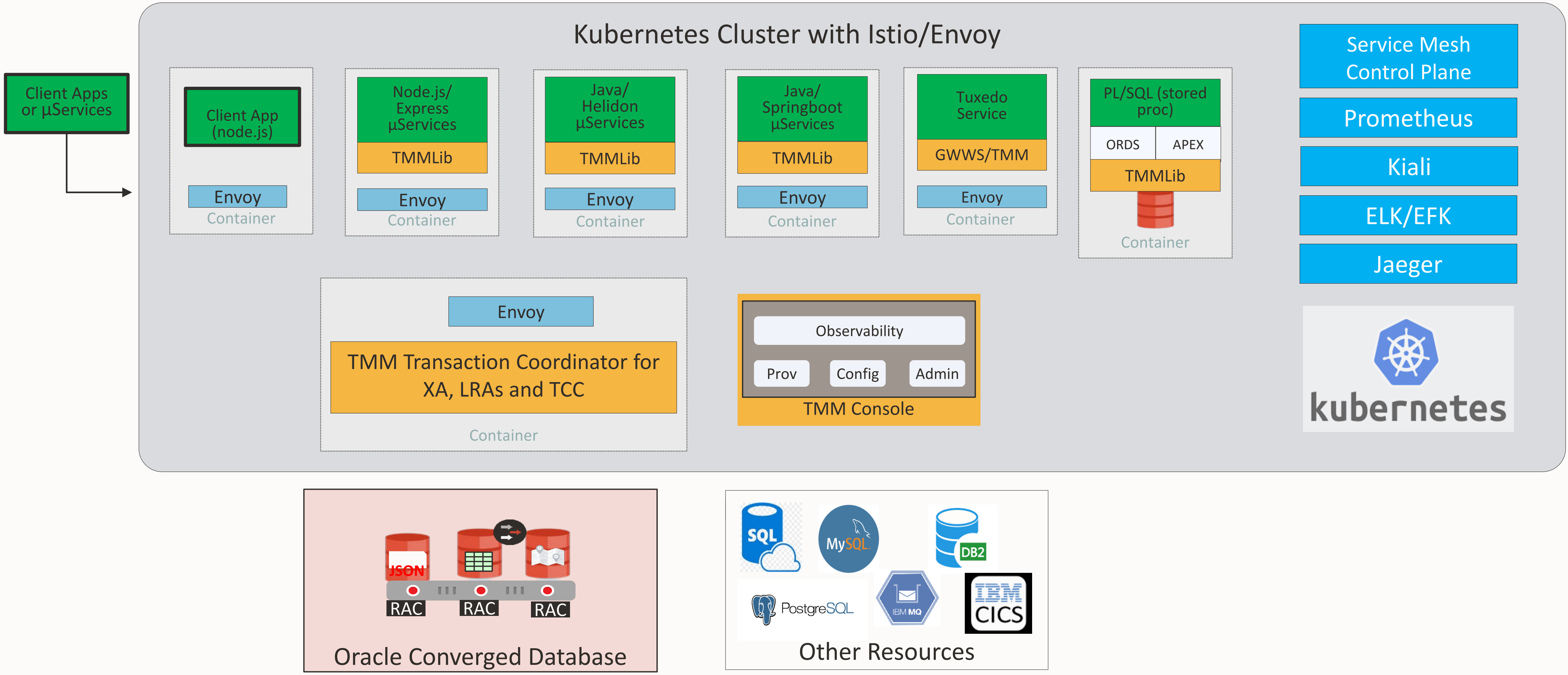


# Oracle Transaction Manager for Microservices

- **Addresses critical needs for enterprise customers**
  - Highly available solution
  - Scalability
  - Security integration
  - OOTB Monitoring, management, administration
- **Supports microservices apps running on-prem, cloud or in hybrid mode**
- **Provides an optimized solution for Oracle Database**
  - Support other databases and non-database resources (compliant RMs for XA)
- **Support most popular programming languages and application frameworks**
  - Java (e.g. Helidon), JavaScript (e.g. Express/JS), C/C++ (e.g. Tuxedo), Golang, Python (e.g Django)



# Transaction Manager for Microservices Architecture

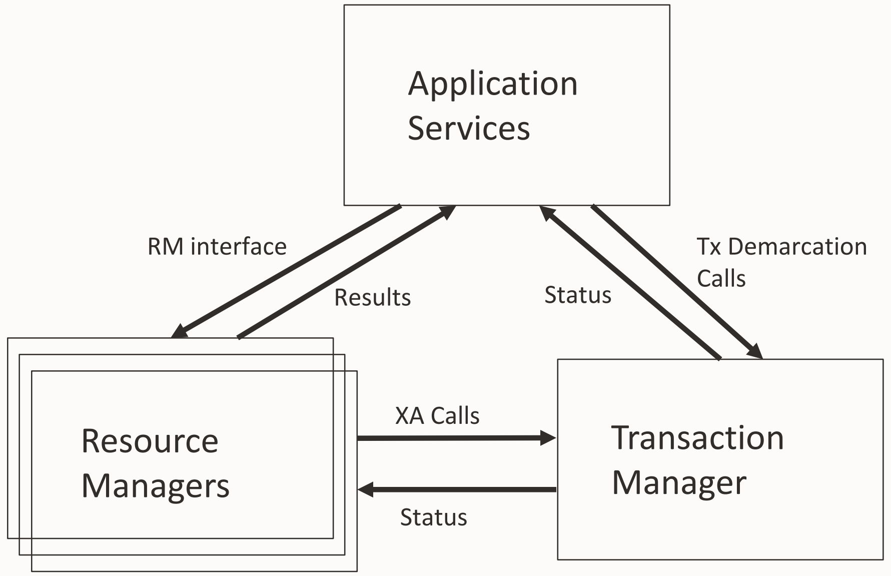


## Agenda

1. Microservices Development Trends & Challenges
2. Oracle's Strategy for Microservices Transaction Management
3. **Transaction Manager Architecture and Features**
4. Demo: Transaction Manager for Microservices
5. Roadmap and Upcoming Features

# Distributed Transactions: XA Transactions

- Two-phase commit protocol, provides strong consistency across multiple resource managers
- Maintains ACID (**A**tomicity, **C**onsistency, **I**solation, and **D**urability)
- Mostly transparent to apps – developers are only using transaction APIs or annotations to demarcate transactions boundaries (begin, commit, abort, etc.)
- XA has been around for quite some time – proven implementations of XA transaction coordinators exist
- Built-in support in almost all enterprise databases
- Perceived as complex and not-scalable due to resource lock between prepare and commit phase



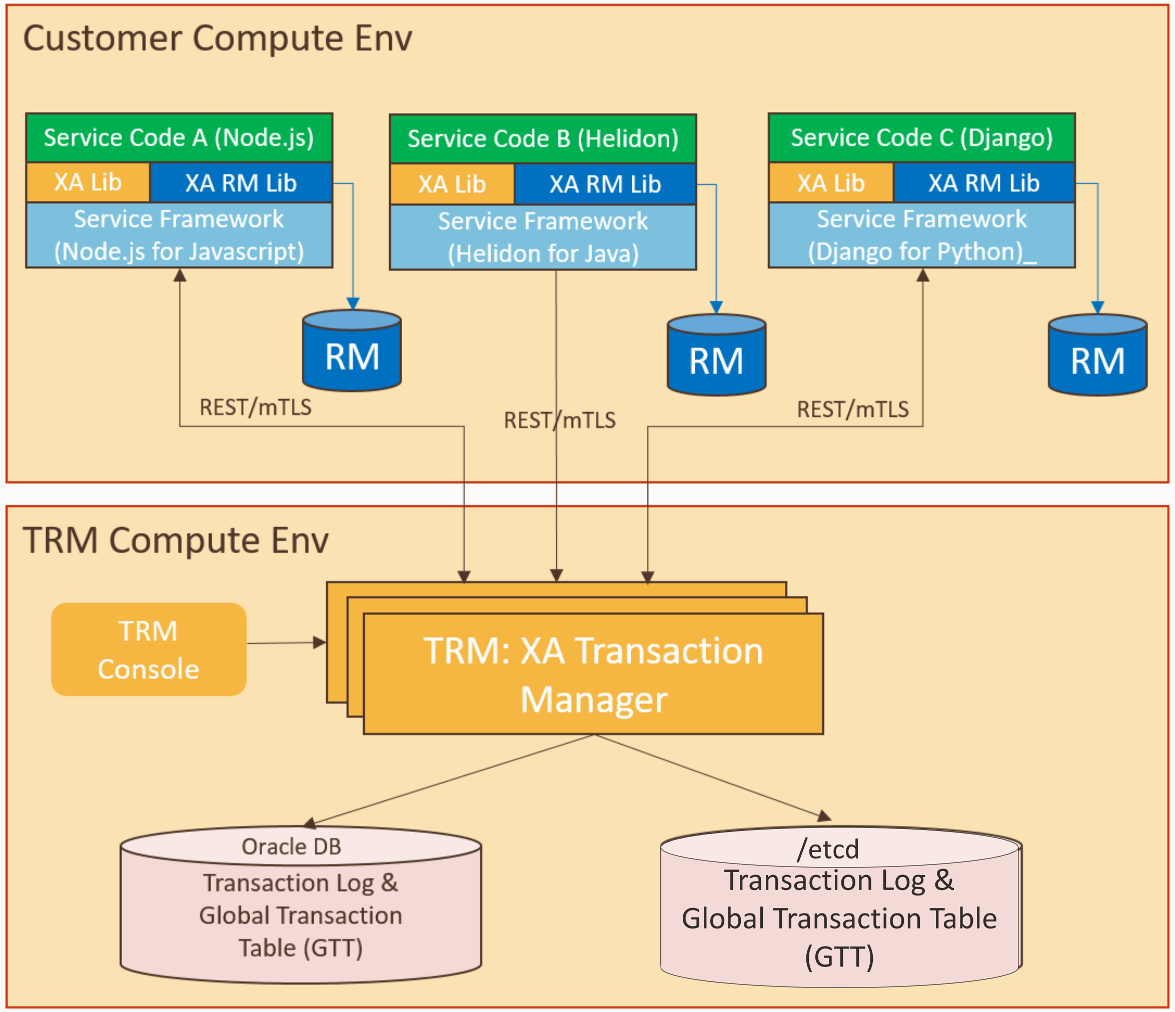
# Using XA Transactions

- Transaction management for strong consistency in polyglot micro services
- Optimized for high throughput and low latency using
  - One-phase commit optimization – common XID across multiple services eliminates need for 2PC
  - Promotable transactions – Automatic promotion of local transactions to distributed transactions
  - Logging Last Resource – enables one non-XA resource to participate in global transactions
- REST API for transaction management (begin, commit, rollback, enlist)
- Transaction API supported in following programming languages:

  Java, Express/js, C/C++ (hosted in Tuxedo),

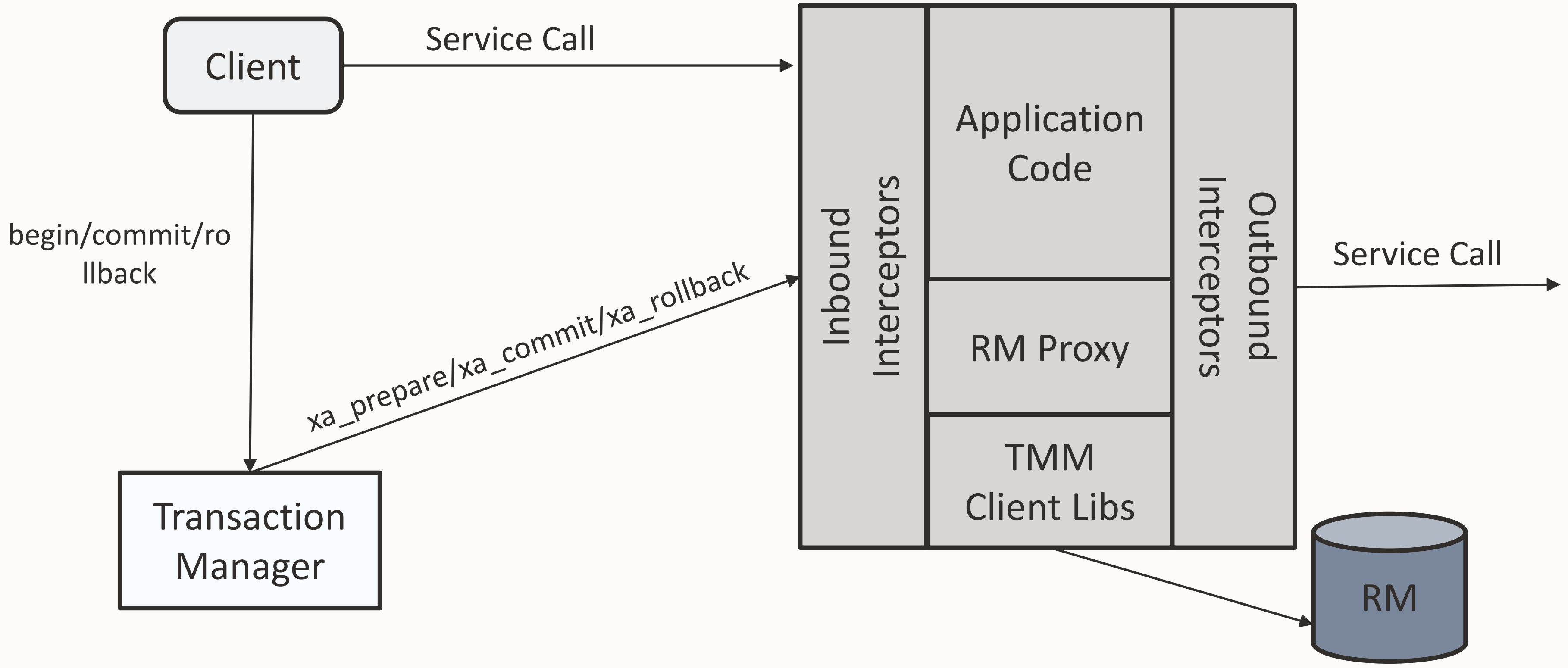
  Python, Go
- Support for popular frameworks (Helidon, SpringBoot)
- All XA compliant RMs supported

  (in addition to Oracle Database)


Customer Compute Env
Service Code A (Node.js)
XA Lib
XA RM Lib
Service Framework (Node.js for Javascript)
Service Code B (Helidon)
XA Lib
XA RM Lib
Service Framework (Helidon for Java)
Service Code C (Django)
XA Lib
XA RM Lib
Service Framework (Django for Python)_
RM
RM
RM
REST/mTLS
REST/mTLS
REST/mTLS
TRM Compute Env
TRM Console
TRM: XA Transaction Manager
Oracle DB Transaction Log & Global Transaction Table (GTT)
/etcd Transaction Log & Global Transaction Table (GTT)

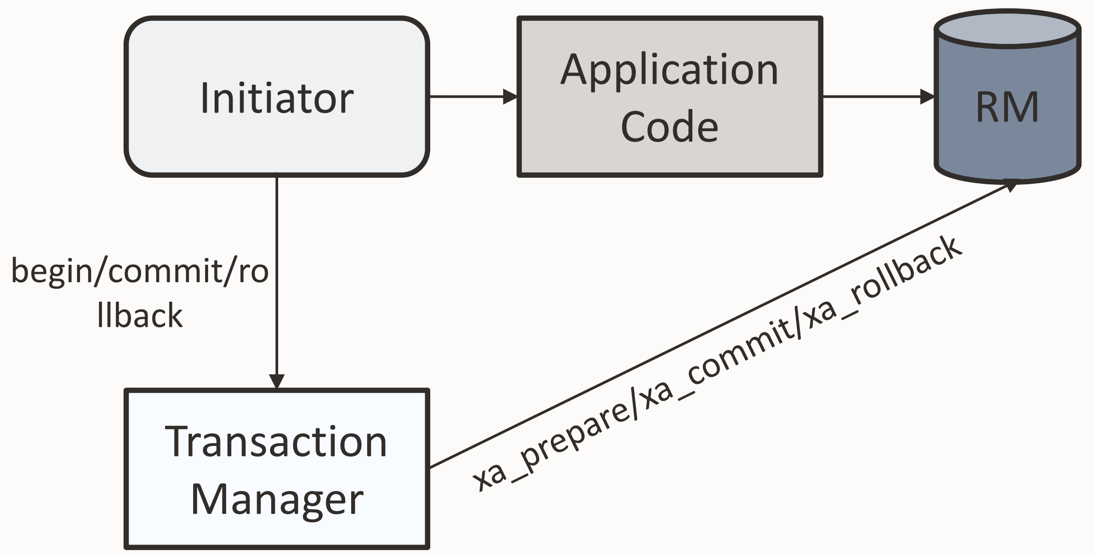
# TMM: Interceptors to Simplify App Development

- Automatically propagates transaction context to subordinate microservices
- Handles automatic enlistment of subordinate microservices
- Greatly simplifies application development

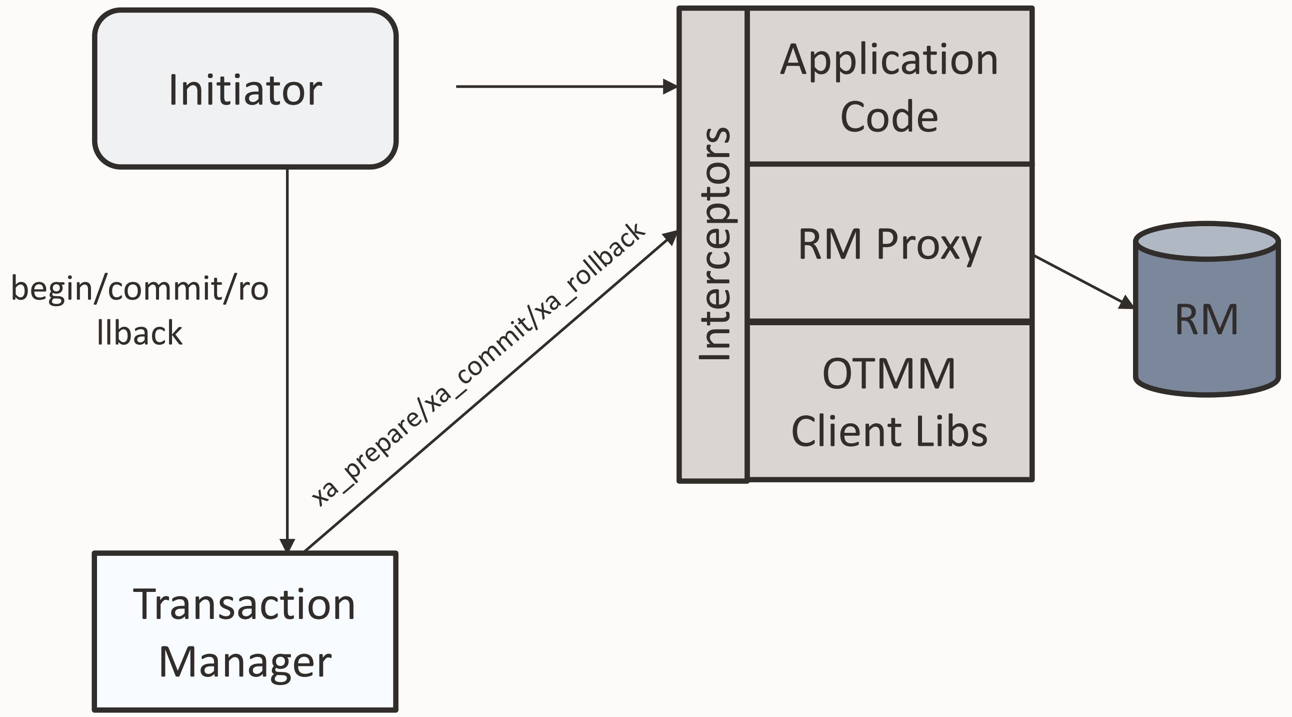


# TMM: Resource Manager Proxy

XA Transaction without RM Proxy



XA Transaction with RM Proxy

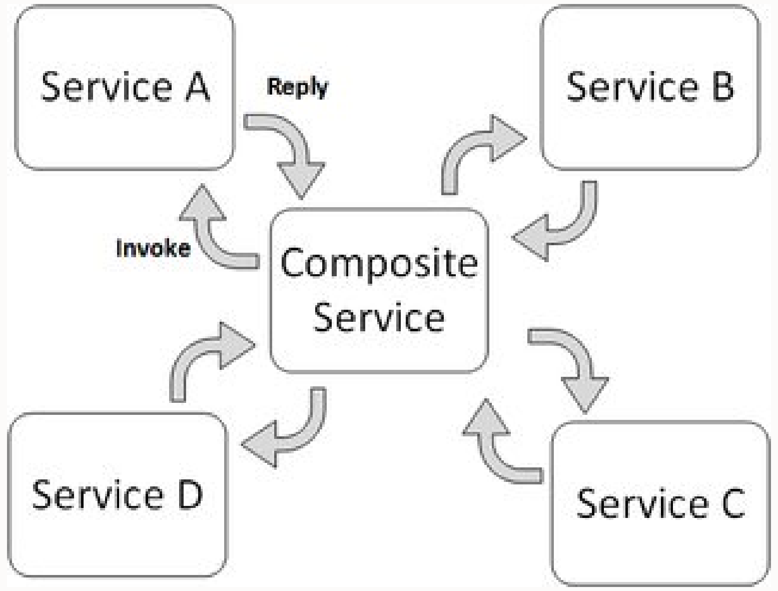


- Completely RM agnostic

# Distributed Transactions: Saga Orchestration

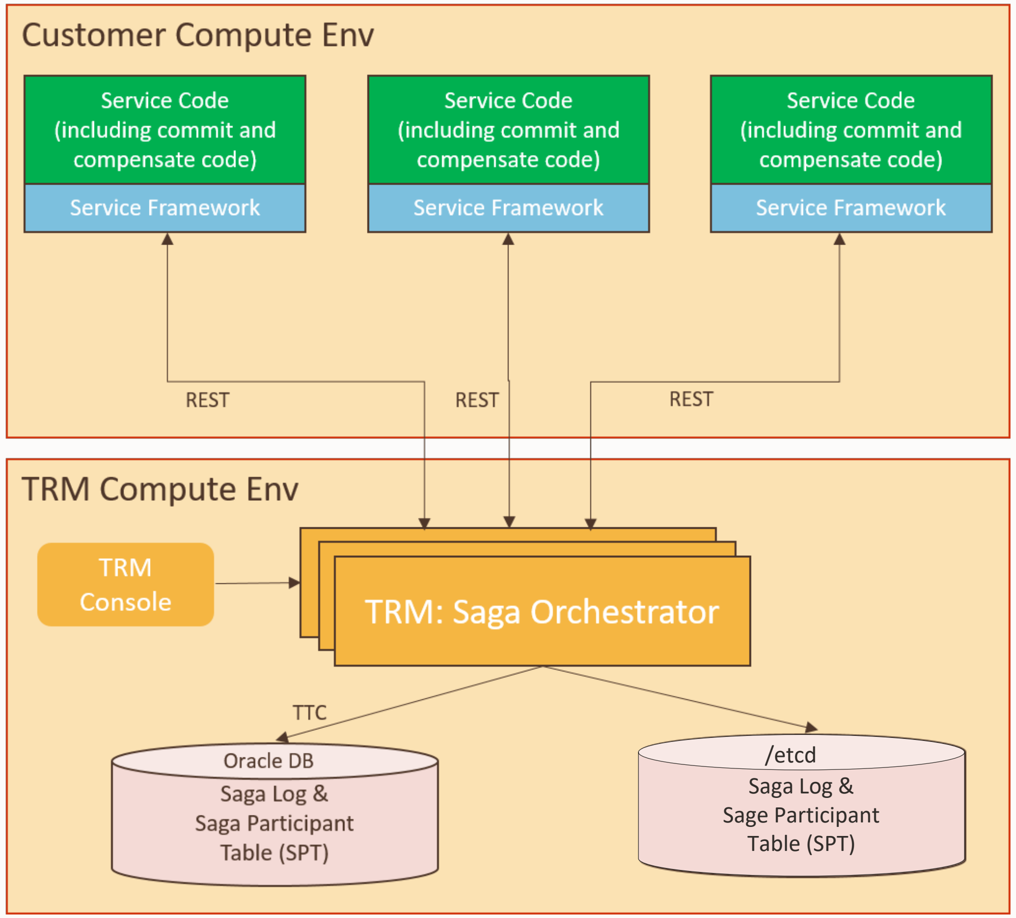
- For Loosely coupled services to coordinate long running activities
- Eventual, globally consistent outcome
- Compensating actions if business activity is cancelled
- Synchronous or Asynchronous operations
- Support in Java for LRA annotations (@LRA, @compensate, @complete, @status, @forget etc.)
- How sagas work:

  A saga is a group of local transactions initiated by an app.
  Each local tx updates the local database.
  If local tx fails, app is notified and in turn saga coordinator is notified and a compensating tx is issued.



# Using Saga Orchestration

- OTMM Saga orchestration simplifies application code and frees developers to concentrate on the core business logic
- Following Eclipse Microprofile Long Running Actions
- Support for LRA annotations in Java and eventually in TypeScript
- Sync and Async communication model for Saga orchestration – sync initially
- Evaluating automating Saga compensating transactions by leveraging Oracle Database support for escrow columns
- Support for multiple programming language frameworks

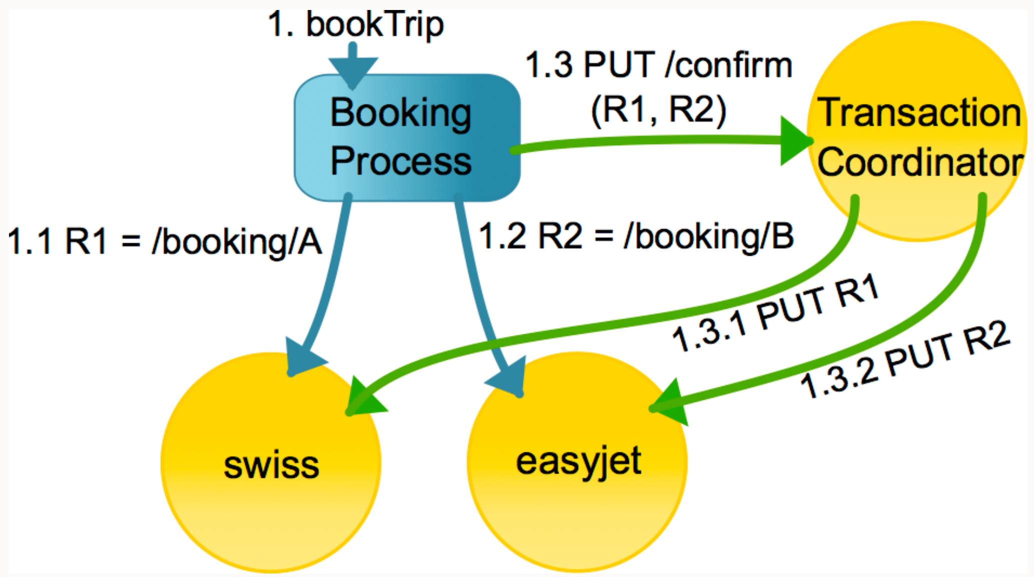

Customer Compute Env
Service Code (including commit and compensate code)
Service Framework
Service Code (including commit and compensate code)
Service Framework
Service Code (including commit and compensate code)
Service Framework
REST
REST
REST
TRM Compute Env
TRM Console
TRM: Saga Orchestrator
TTC
Oracle DB
Saga Log & Saga Participant Table (SPT)
/etcd
Saga Log & Sage Participant Table (SPT)

# Distributed Transactions: Try-Confirm-Cancel

- Compensating transaction pattern with 2-phase protocol

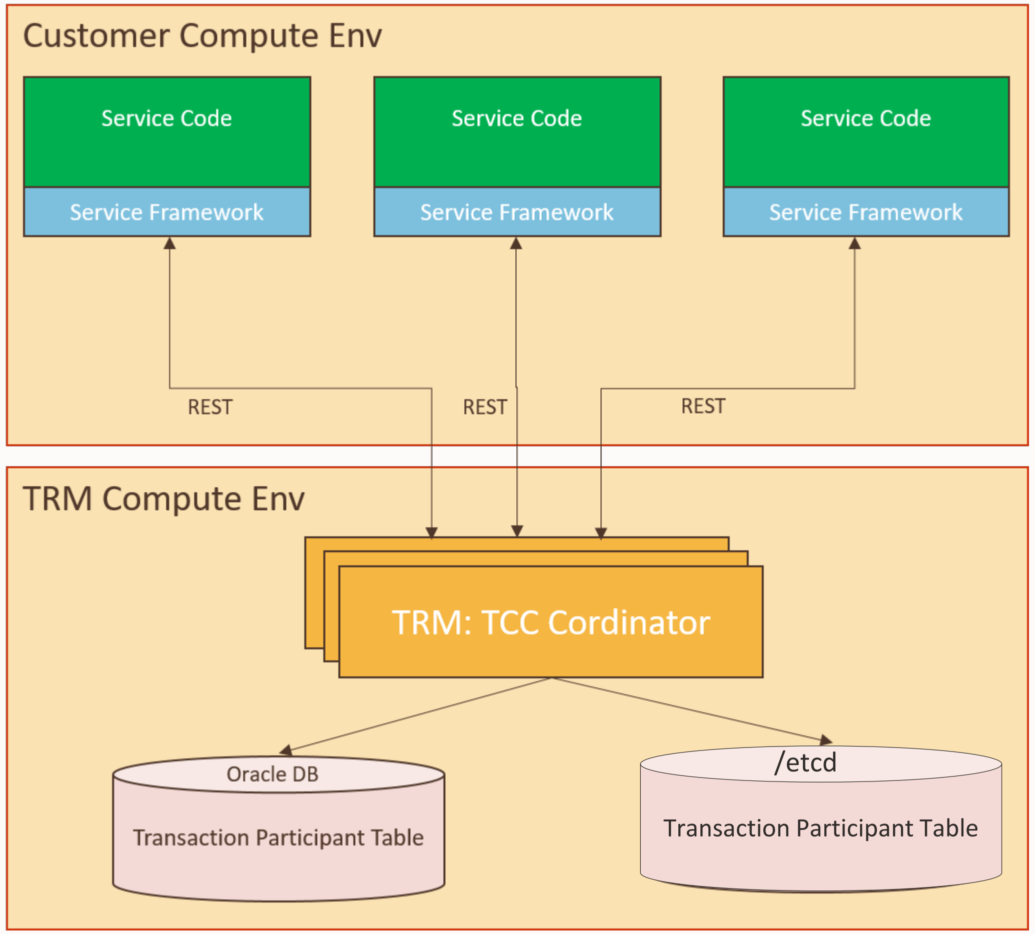
**Try phase puts service/resource in pending/reserved state; confirm finalizes the state or cancel aborts it**

- Compensating a failed transaction is easier
- Service needs to be designed with pending /reservation state model



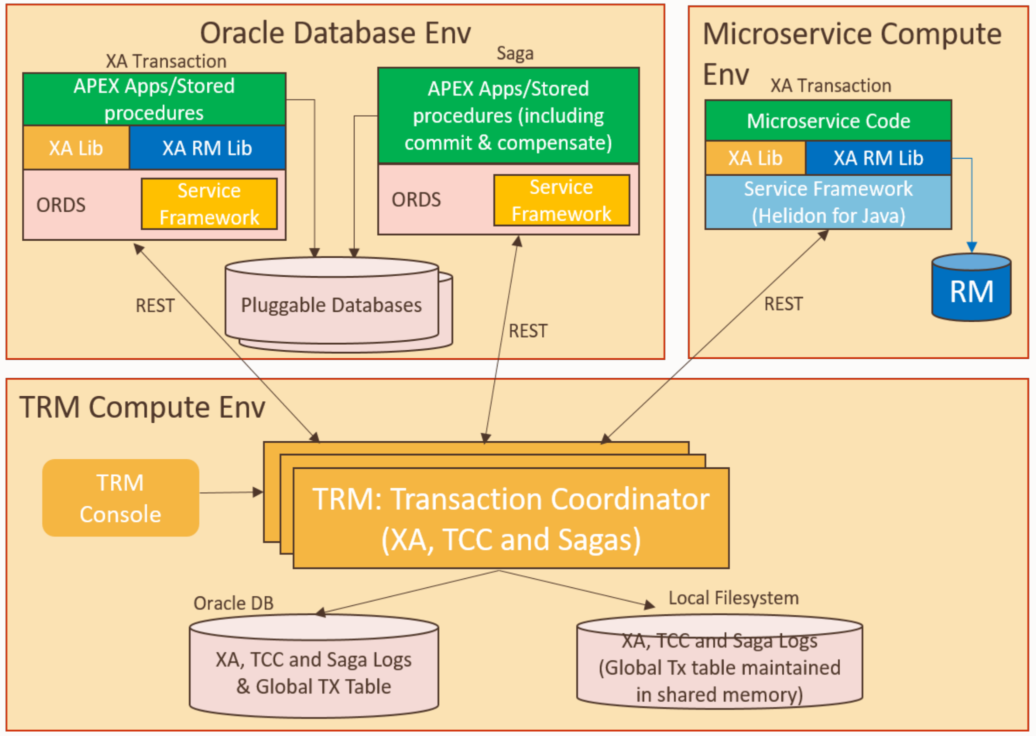
# Try-Confirm-Cancel

- APIs for TCC transaction coordination, similar signature as LRA
- Transaction initiator or first service/resource enlists transaction with TCC coordinator
- TCC Coordinator maintains state for all participants
- Sync and Async communication model for Saga orchestration
- Support for multiple programming language frameworks



# Oracle Database Resident Apps

- Oracle Database apps (APEX apps and apps using PL/SQL stored procedures) can participate in XA & TCC transactions, and in sagas
- These apps use Oracle REST Data Services or ORDS to expose REST end points, which are used to communicate with OTMM transaction coordinator
- Same transaction or saga can include Database apps and other microservices running outside of Oracle Database
- Flexibility to use same Database env as APEX apps for transaction logs and GTT

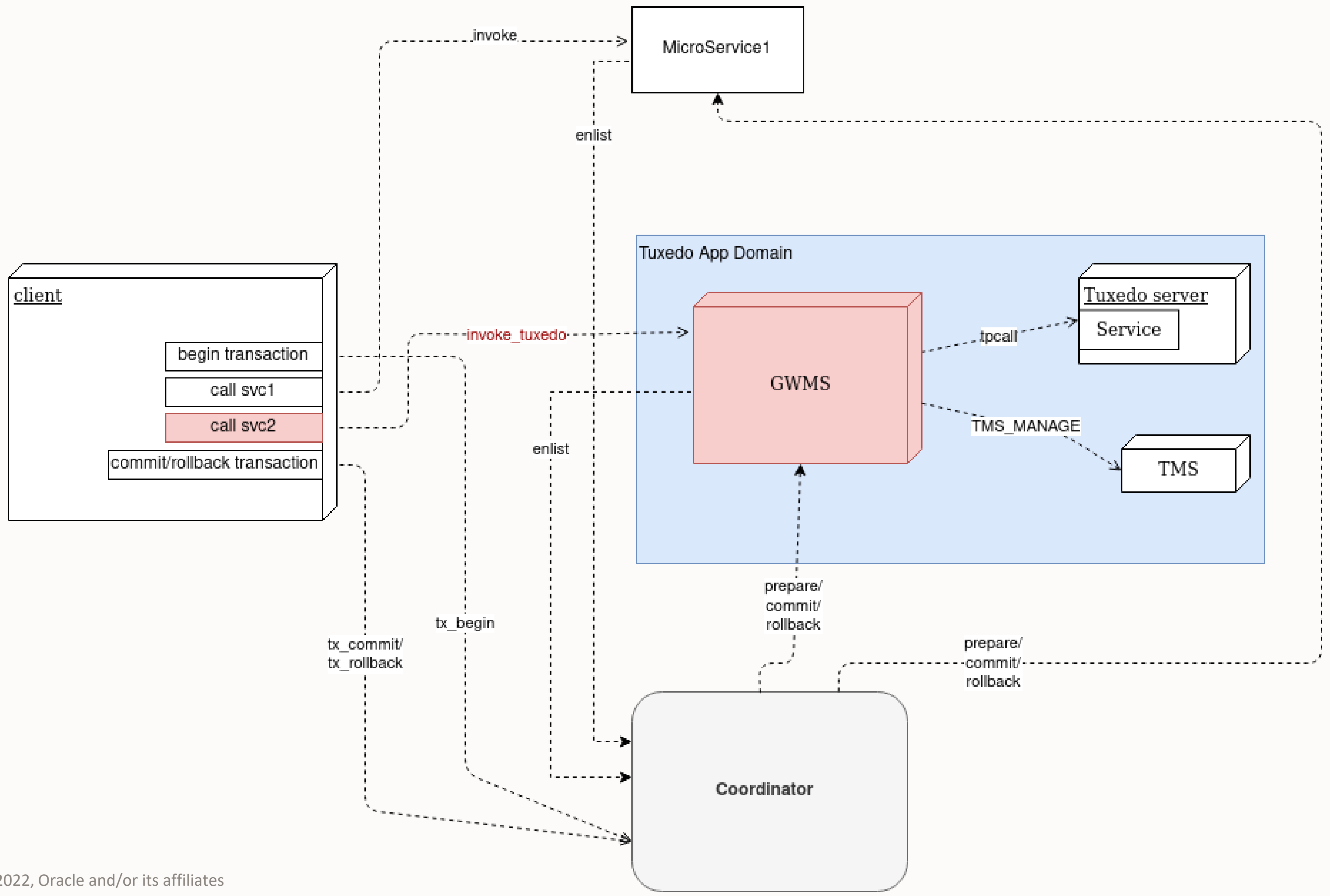


Oracle Database Env
XA Transaction
APEX Apps/Stored procedures
XA Lib
XA RM Lib
ORDS
Service Framework
Saga
APEX Apps/Stored procedures (including commit & compensate)
ORDS
Service Framework
Pluggable Databases
REST
REST
Microservice Compute Env
XA Transaction
Microservice Code
XA Lib
XA RM Lib
Service Framework (Helidon for Java)
RM
REST
TRM Compute Env
TRM Console
TRM: Transaction Coordinator (XA, TCC and Sagas)
Oracle DB
XA, TCC and Saga Logs & Global TX Table
Local Filesystem
XA, TCC and Saga Logs (Global Tx table maintained in shared memory)

## Oracle Tuxedo

- Transaction processing system – Effectively an application server for primarily non-Java applications
- Everything is a service, application, configuration, transaction management, etc.
- Extremely low latency - < 30 µsec request/response
- Extremely scalable - > 500,000 tps
- Rehosting IBM mainframe based applications to distributed environment
- Connectors/gateways: SOAP, REST, JCA, Mainframe SNA, IBM MQ, Java, WebLogic Server,
- Used by: PeopleSoft, Walgreens, Telefonica, SWIFT, USPS, VA, NTT Docomo, AT&T,…

# Tuxedo Integration

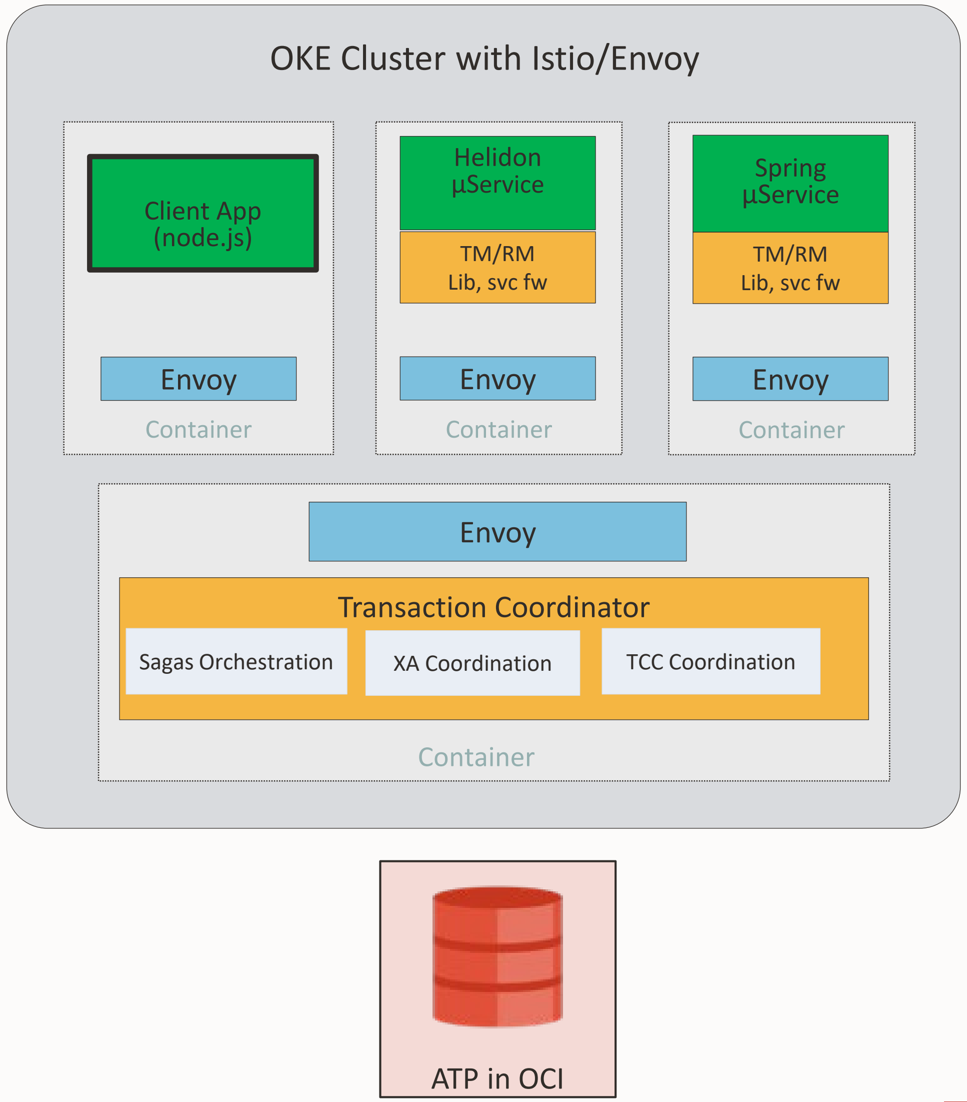


# Agenda

1. Microservices Development Trends & Challenges
2. Oracle's Strategy for Microservices Transaction Management
3. Transaction Manager Architecture and Features
4. **Demo: Transaction Manager for Microservices**
5. Roadmap and Upcoming Features

# Demo: XA transaction

- Polyglot services in an XA transaction
- Includes app components written in node and Java
- Node client initiates an XA tx
- Calls Helidon/Java svc to withdraw money
- Then calls Spring/Java service to deposit this amount



# Agenda

1. Microservices Development Trends & Challenges
2. Oracle’s Strategy for Microservices Transaction Management
3. Transaction Manager Architecture and Features
4. Demo: Transaction Manager for Microservices
5. **Roadmap and Upcoming Features**

# Transaction Manager for Microservices

## Beta – February 2022

- Transaction Coordinator for XA, LRA and TCC transaction models
- Support for polyglot microservices
  - Java (Helidon, SpringBoot, Weblogic)
  - Node.js (express/js)
  - C/C++/COBOL (Tuxedo)
  - PL/SQL (Oracle Database)
- Kubernetes and service mesh based deployment
  - Kubernetes native service along with Istio and Envoy
  - Full life cycle management using helm charts
  - Integration with K8S eco system – Prometheus/Graffana, ELK/EFK, Jaeger, Kialli
- Support for services deployed within or across K8S clusters
- Secure deployment – OpenID/JWT based authentication and authorization
- OKE, OKD (OpenShift) and minikube certification
- RM Support for Oracle Database, MySQL*
- Support for Microsoft AD, Identity Domains and Keycloak as IdP
- QuickStart guide for OKE and minikube
- Sample apps for each transaction model

## GA – July 2022

- Usability Enhancements
  - Last Logging Resource (enable non-XA RMs to participate in XA tx)
  - Simplify XA DataSource Initialization
  - Nested transactions
- XA Performance Enhancements
  - Common XID
- TMM Console
- Certification with OLCNE and other K8S environments (VMWare Tanzu)
- TMM as a component of Verrazano

## Beyond GA

- TMM Kubernetes Operator
- Support for other programming languages: Python, Golang
- TMM-as-a-Service in OCI (managed service)
- Tuxedo interop for outbound XA transactions
- Simplified integration with Database resident services
- LRA support for Database resident microservices

# Thank you