Adatmenedzsment
Microservices Architecture, Oracle Database, OCI

Fekete Zoltán

Principal Solution Engineer

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.

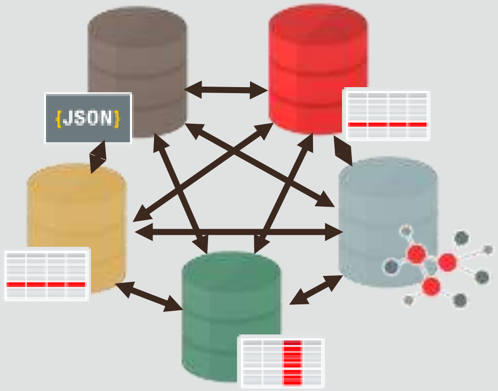
Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's 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.

Transforming Monoliths into Microservices

I.T Cost and Complexity

- For decades, I.T. has been costly and slow moving
  - This presentation will focus on Data Management

- Root cause is complexity caused by:
  - Enterprise Product Complexity
  - Systems Integration Complexity

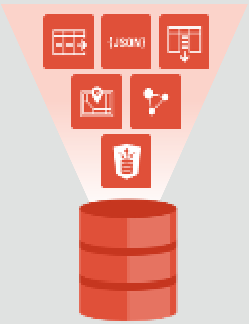


- New technologies can finally eliminate the sources of I.T. complexity

Komplexitás és költség

- IT költségek és sok idő
    - A fő ok a komplexitás, sok tevékenység, menedzsment

- **Új megközelítések** eltüntetik a komplexitás okait
- **Autonomous management, cloud, Machine Learning**
- Konvergens megoldás a szükséges funkciókkal
  a rendszerintegráció komplexitását eliminálja



Data Strategy

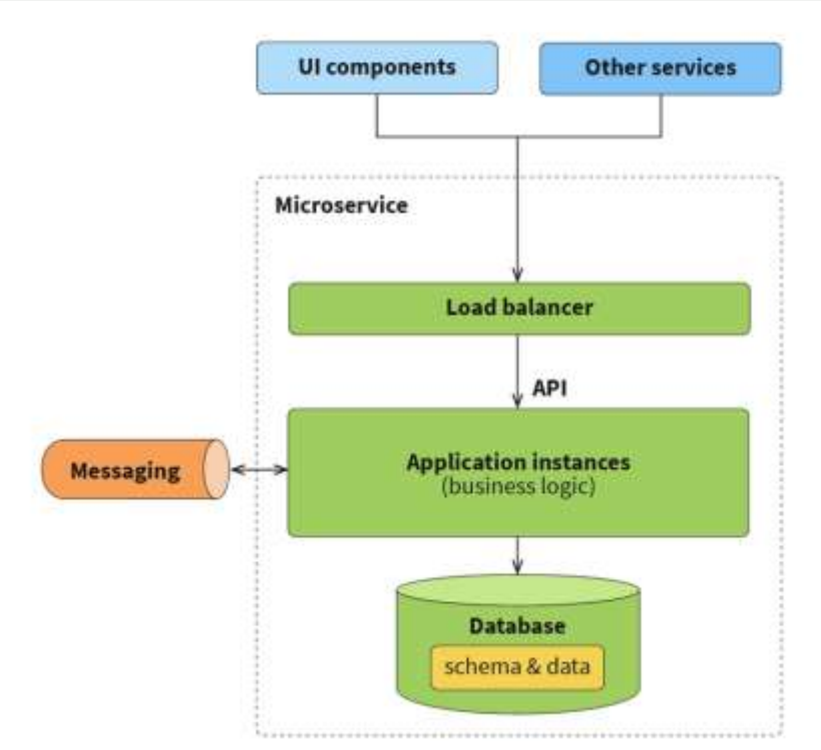
- Modern Applications require many different:
  - **Data Types** - Relational, Document, Spatial, Graph, etc.
  - **Workloads** - Transactions, analytics, ML, IoT, etc.

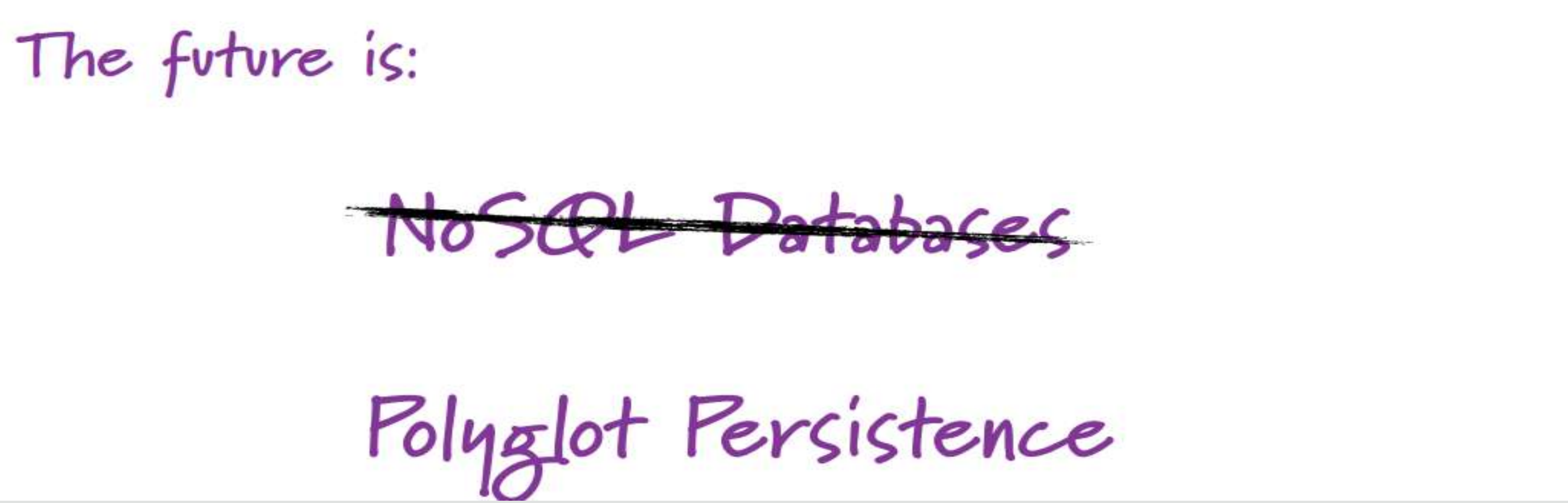
- Each data type and workload requires different database algorithms

- Two possible Data Strategies:
  - Use single-purpose “best-of-breed” database for each data type and workload
  - Use a converged database for all data types and workloads

Why Microservices?

- Develop application as suite of loosely-coupled small services, each running in its own context and communicating with lightweight mechanisms
- Enables rapid, frequent and reliable delivery of complex applications
- Each microservice should be
  - Highly maintainable and testable
  - Loosely coupled
  - Horizontally scalable
  - Independently deployable **with own database**
  - Organized around business capabilities
  - Owned by a small team





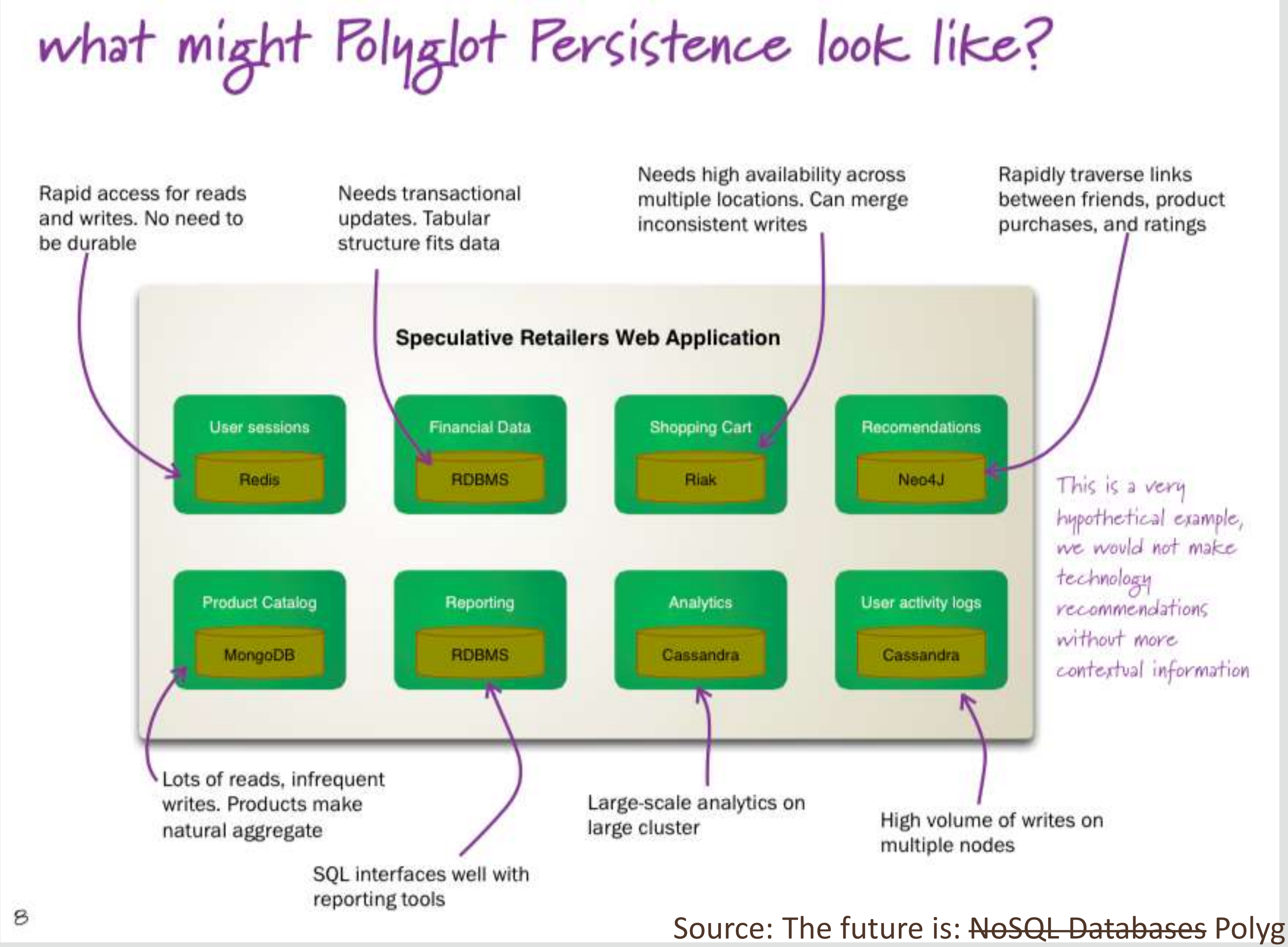
The future is:

~~NoSQL Databases~~

Polyglot Persistence

“Polyglot persistence will occur over the enterprise as different applications use different data storage technologies. It will also occur within a single application as different parts of an application’s data store have different access characteristics.”

Martin Fowler & Pramod Sadalage, Feb. 2012
http://martinfowler.com/articles/nosql-intro-original.pdf



what might Polyglot Persistence look like?

Rapid access for reads and writes. No need to be durable

Needs transactional updates. Tabular structure fits data

Needs high availability across multiple locations. Can merge inconsistent writes

Rapidly traverse links between friends, product purchases, and ratings

Speculative Retailers Web Application

User sessions Redis	Financial Data RDBMS	Shopping Cart Riak	Recomendations Neo4J
Product Catalog MongoDB	Reporting RDBMS	Analytics Cassandra	User activity logs Cassandra

This is a very hypothetical example, we would not make technology recommendations without more contextual information

Lots of reads, infrequent writes. Products make natural aggregate

SQL interfaces well with reporting tools

Large-scale analytics on large cluster

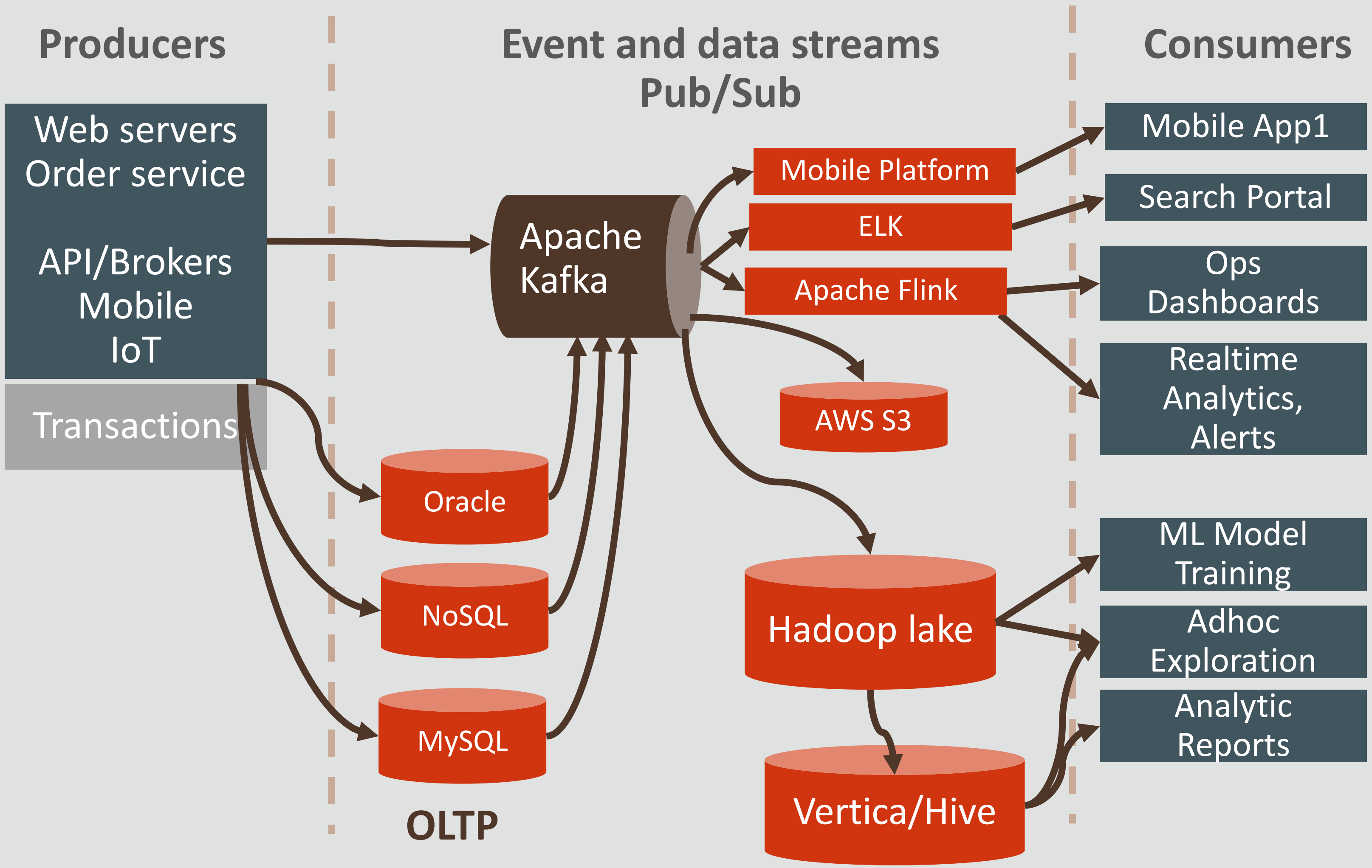
High volume of writes on multiple nodes

Source: The future is: ~~NoSQL Databases~~ Polyglot Persistence
http://martinfowler.com/articles/nosql-intro-original.pdf

Different apps/ **μServices** have different needs

		User Sessions	Financial Transactions	Shopping Cart	Recommend. Engine	Product Catalog	Reporting	Analytics	Activity Logs
Processing	Heavy Writes								√
	Heavy Reads				√	√	√	√	
	Fast Read/Write	√							
	Data Consistency		√						
	Data Durability		√						
	Analytic						√	√	
	Graph				√				
	Spatial								
	Geo Distribution			√		√			√
Data	Relational		√				√	√	
	Key/Value	√		√					√
	Document/JSON					√			√
	Graph				√				

Real-World Example of Macro-Complexity



Macro-Complexity

- Multiple technologies
- Multiple data stores
- Data copied multiple times to do analytics
- Compromises security
- Compromises data consistency
- Complex to maintain
- Need highly skilled developers to build & keep running

One Converged Database vs Several Specialized Databases

Modern Information Systems

Data Types
Relational, Document, Event, Spatial, Graph etc.

Application Types
Transactions, Analytics, Microservices, ML, IoT, etc.

Architectural Strategies

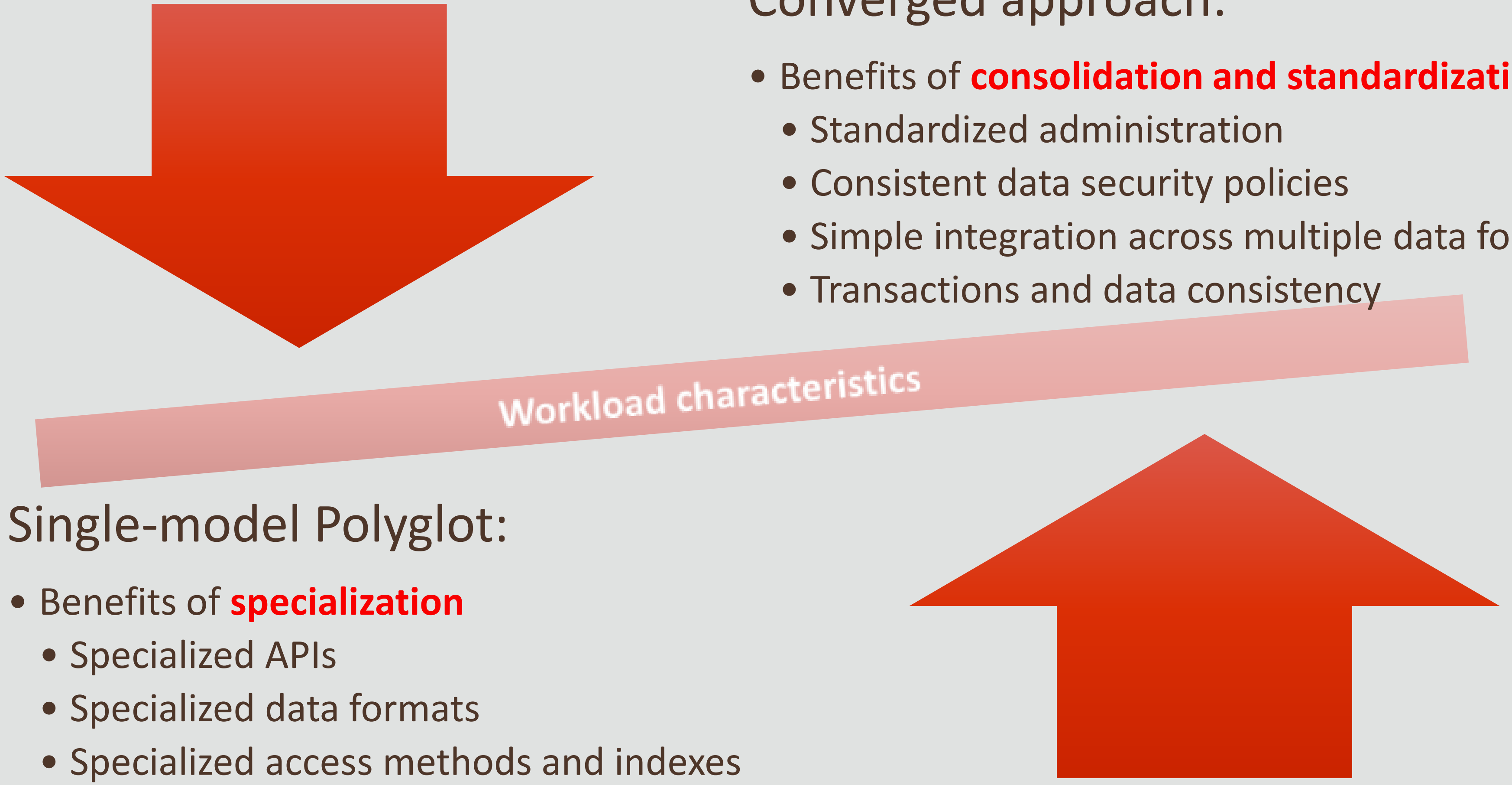
AWS
Run separate Specialized Databases for each data type

Oracle
Run one Converged Database that supports multiple data types

Considerations for Converged

Converged approach:

- Benefits of **consolidation and standardization**
  - Standardized administration
  - Consistent data security policies
  - Simple integration across multiple data formats
  - Transactions and data consistency



Single-model Polyglot:

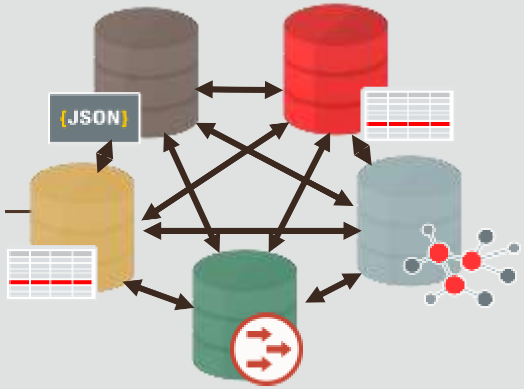
- Benefits of **specialization**
  - Specialized APIs
  - Specialized data formats
  - Specialized access methods and indexes

The Hidden Pain of Data Management and **µServices**

- **µService** philosophy encourages data store independence
  - Choose the right data store for the characteristics of your service
- Data store separation comes with tradeoffs and complexities that multiply with the granularity and interdependence of **µServices**
  - Data Consistency: Important data elements across **µServices** should have the same format, meaning, and ultimately values
  - Data Sharing: Do you replicate or aggregate common data with **µServices?** Are you building 2PC across **µServices** or creating an ETL headache?
  - Data Security/Governance: Are you propagating sensitive data, or creating a massive threat surface?
  - Overall complexity: As each **µService** adds its own unique technology stack it increases the operational overhead for the company overall.

Fragmentated Data Architecture Creates Complexity
Functional Isolation Leads to Complexity

- Each single-purpose database that is deployed fragments the data architecture
  - **Different proprietary APIs, languages**, and transaction models
  - Different operational needs and limitations
  - Must continually **transform data and propagate changes** – causing data delays and data divergence
  - **Must separately implement HA and Security** policies in every database to accommodate their differences
- End-to-end application security, availability, scalability, consistency, etc. limited by the weakest of the databases
- Intent was “best-of-breed”, result is “worst-of-weakness”



Fragmented Features vs. Converged Product

- Phone calls, messaging, camera, calendar, etc. used to require separate products
  - Now converged into features of smartphones
- Similarly, key-value, analytics, JSON, sharding, etc. originally required separate products
  - Now converged into features of Converged Database
- Simpler, better, and creates powerful synergies across features

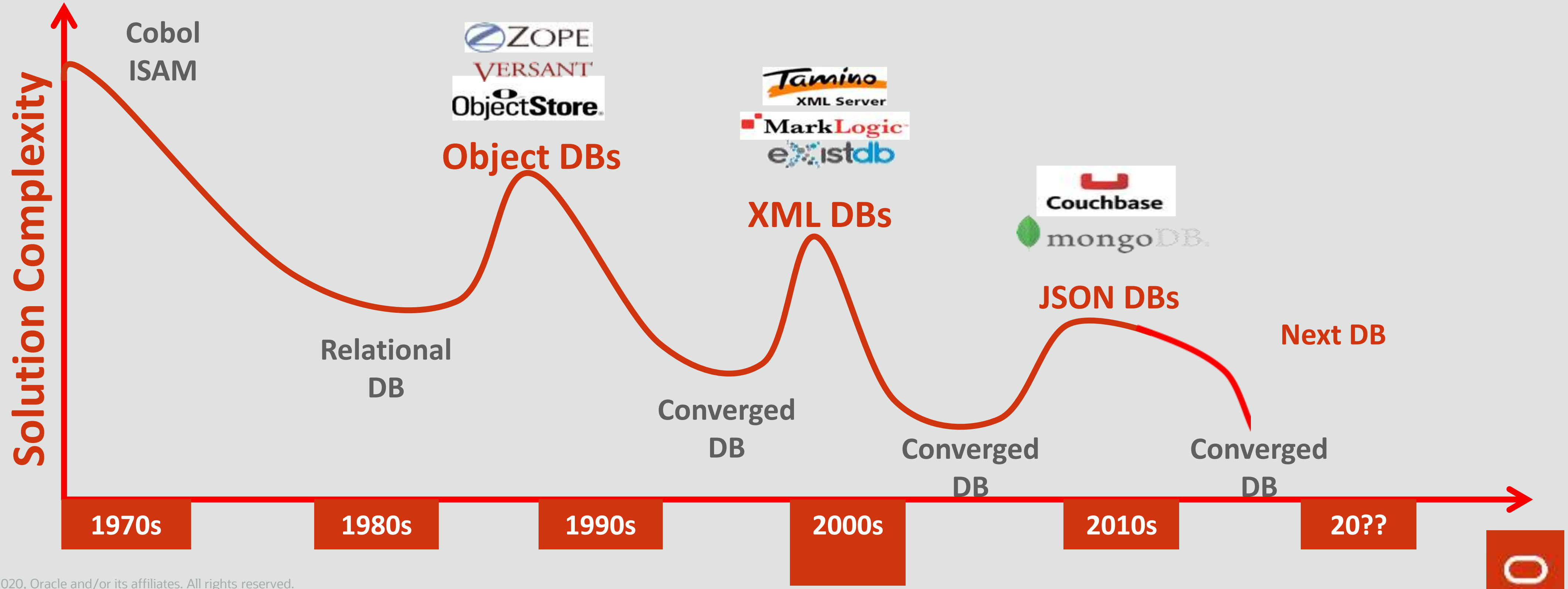
µService Data management Tradeoffs

	Tradeoffs	Separate DMPs	1 DMP Single Schema	1 DMP Multi Schema	1 DMP PDB per Service
Dev	Dev Agility	●	○	◔	◕
	Choice of data model/structure	●	○	◑	●
	Service Isolation	●	○	◔	◕
Data	Data Consistency	○	●	●	◕
	Data Sharing	○	●	●	◕
	Data Security	◔	◑	◕	●
OPS	Common Security Model	○	●	●	●
	Independent Service Scaling	●	○	◔	◕
	Common Management and HA	○	●	●	●

DMP = Data Management Platform

Over Time New Functionality is Converged Into Mainstream

- Single-purpose databases have emerged many times
- Abandoned after features are added to converged databases



Oracle Autonomous Database

Converged Features

Multi: tenant, language, model

Multitenant for Efficient, Agile Database Clouds
In-Memory for Database Acceleration
Sharding for Hyperscale and Geo Distribution
Native JSON for Document Data
In-Memory Ingest for Fastest IoT
Cloud SQL for integrating Object Store Data Lake
AutoML for simple integrated Machine Learning
Persistent Memory Store for Lowest Latency
Blockchain Tables for Preventing Fraud
Spatial and Graph for Mapping and Social Networks
Events for Transactional Event-driven Microservices
And many more …

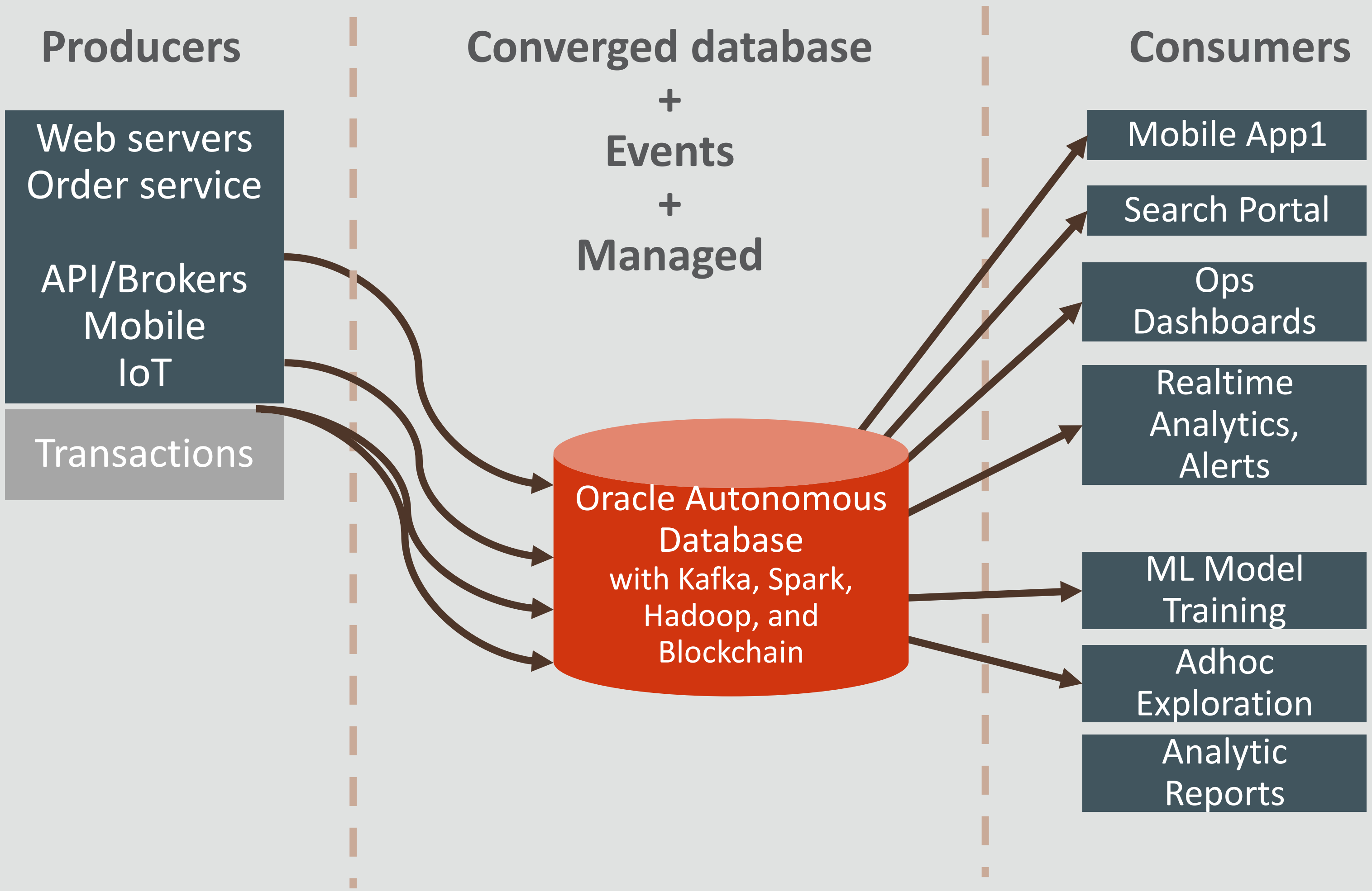
Multitenant
In-Memory Analytics
Hyperscale
JSON
In-Memory IoT
Cloud Integration
Blockchain
Persistent Memory
Machine Learning
Spatial
Graph
Events

https://blogs.oracle.com/database/what-is-a-converged-database

Polyglot Persistence Market Trends

- Single-model architectures are most pervasive for ‘edge’ applications
  - New business & workload requirements
- Business applications naturally converge to multi-model architectures
  - Today’s ‘edge’ applications are tomorrow’s mainstream business applications
  - Efficiencies of multi-model architecture override advantages of special-purpose systems over time
- There will always be single-model polyglot architectures
  - Because there are always new ‘edge’ applications
  - Oracle’s single-model architectures:
    - Oracle Berkeley DB, Oracle NoSQL Database, Essbase, Oracle Big Data Spatial and Graph

What Customers Have Asked For



- Microservices support
- Cool app building blocks & APIs
- Real-time, current data
  - No need to copy data around
- Less to learn, manage, backup, upgrade, secure, (7x fewer security patches)
- Self-managing with Autonomous Database
- *No need for army of developers to keep running*
- Choice of deploying N databases for business reasons
  (2 is better than 9)

Oracle Cloud Database Services

Oracle Autonomous Data Warehouse →

Oracle Autonomous Transaction Processing →

Oracle Database Cloud Service: Bare Metal →

Oracle Database Cloud Service: Virtual Machine →

Oracle Database Exadata Cloud Service →

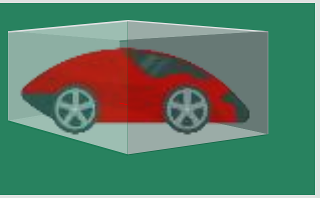
Oracle Database Exadata Cloud at Customer →

Konvergens, automatizált, beépített biztonság és gépi tanulás

Self-driving

Kevesebb emberi működtetés

Self-securing



Megvédi magát a támadásoktól

Self-repairing

Folyamatos működés

Az első autonóm adatbázis

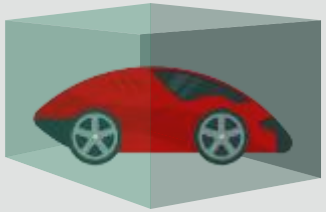
Hagyjuk, hogy az adatbázis végezze el a munkát!

Az Autonomous adatbázis fő ismérvei

Önvezető

Automatizálja az adatbázis- és infrastruktúramenedzsment feladatokat, monitorozást és a hangolást
Scale out, fault tolerance, DR
Compatibility, Exadata

Emberi erőforrások megtakarítása



Önvédő

Megvéd a külső támadásoktól, és a rosszhiszemű belső felhasználóktól
Aut. online biztonsági frissítés
Biztonságos konfiguráció
Titkosítás

Emberi hibák kiszűrése és megelőzése

Önjavító

Megakadályoz minden típusú leállást
A tervezett karbantartásokat is online végzi el
Elasztikus skálázás
99,95% és 99,995% uptime (karbantartás is benne)

Emberi beavatkozás nélkül

Minden **automatizált**

- Provisioning
- Clustering
- Disaster Protection
- Tuning
- Scale-Up and Scale-Out
- Security
- Patching
- Backup – 60 nap ingyenesen az előfizetési díjban

Machine Learning Driven Operations

- Exadata
- RAC
- Data Guard
- Database Vault
- Multitenant
- Parallel SQL
- Flashback
- Etc.

Autonomous Database

ORACLE

Shared serverless infrastructure

Egyszerű

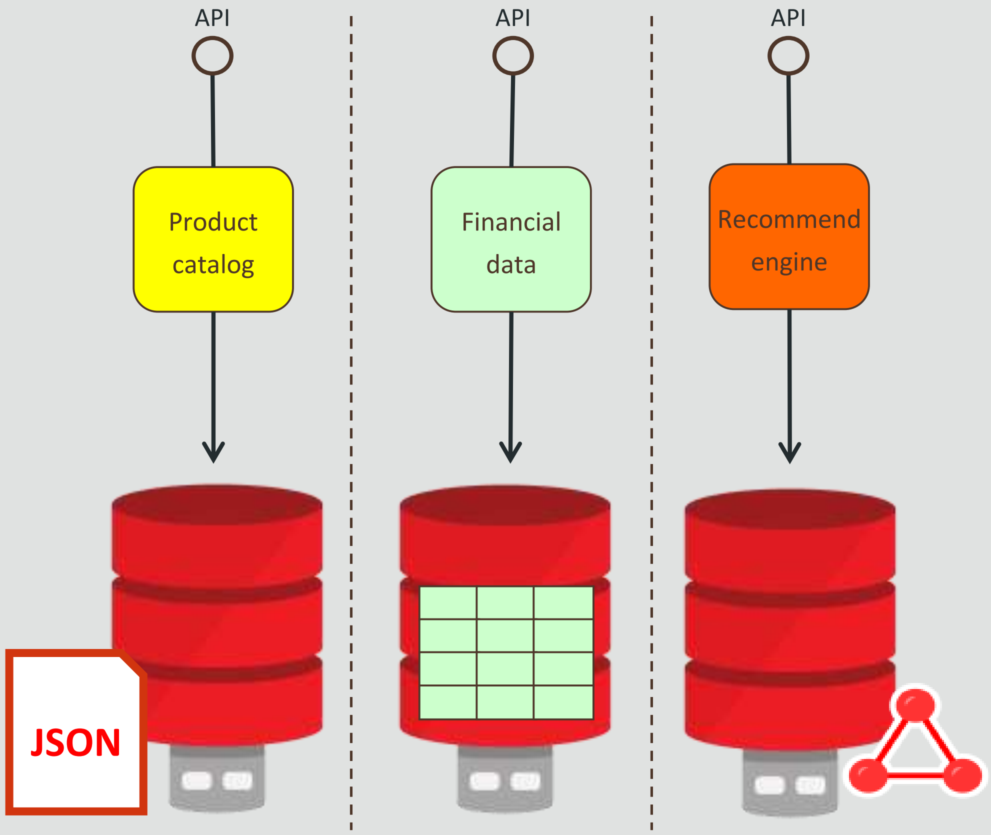
- Oracle mindent automatizál és menedzsel
  - létrehozás, életciklus, software update-ek, stb.
- Ügyfél választása: DB OCPU, storage TB, region

Elasztikus

- minimum - 1 OCPU: Serverless, amikor nem fut
- Automatikus skálázás online, futás közben: True Pay-per-Use: **másodperc alapú**
- alacsony minimum time commitment – 1 perc

1 Container DB, and a Pluggable DB for each µService

Common Data Management Platform
Separate PDBs for each **µServices**
Common Reference Data



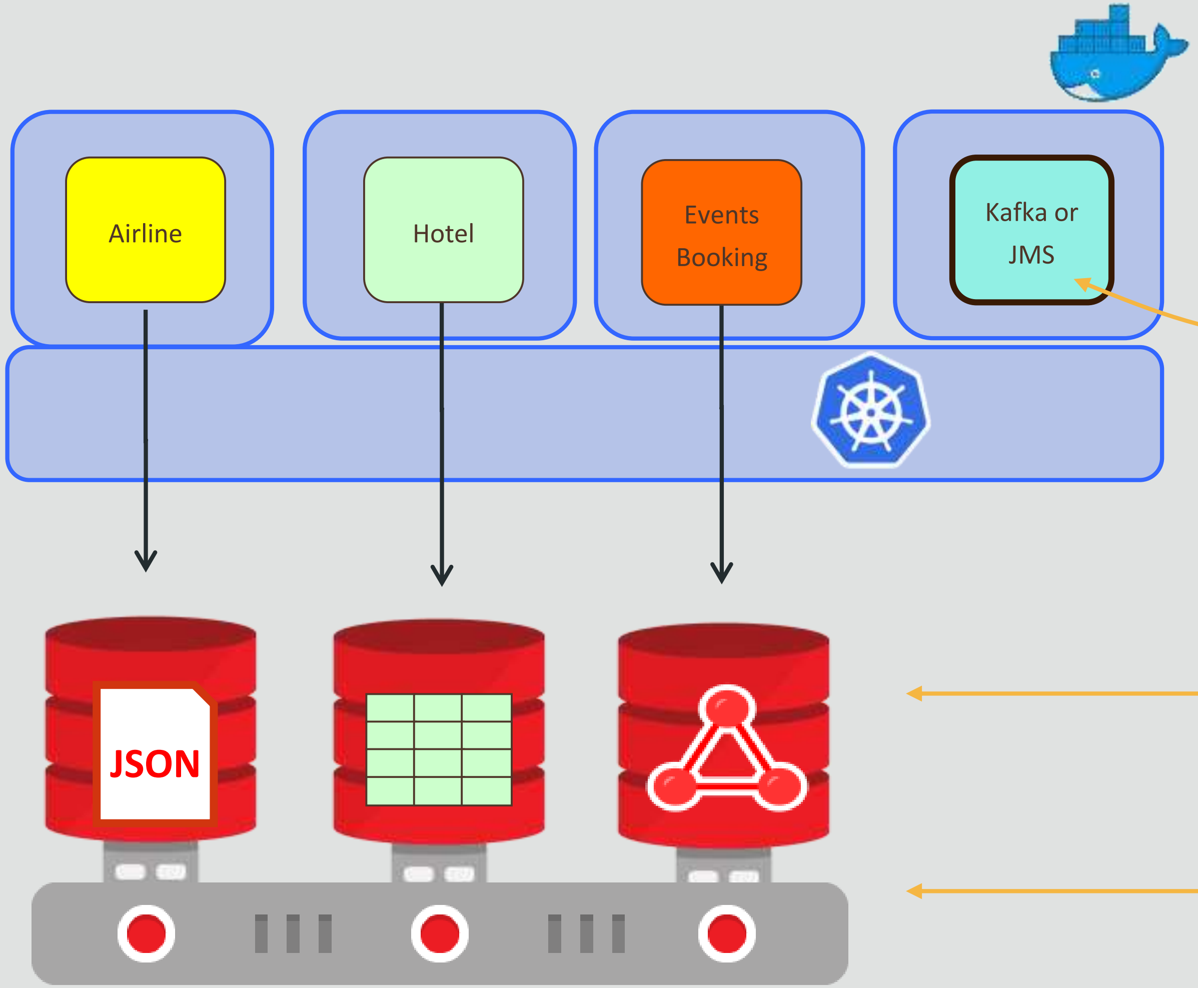
Pros

- Freedom in models for each **µServices**
- Limited Development dependency
- **µServices** isolation
- Model consistency and shared reference data possible with Application containers
- Scaling independence for PDBs
- OPS consolidation with some resource control

Cons

- Limited freedom in technology choices

Converged Database Architecture



Docker containers running microservices

Microservices communicate via messaging

Kubernetes for container orchestration

Oracle Pluggable Databases (PDBs)
Each PDB used as data store for a microservice.
Each PDB can scale via SMP, Real Application Clusters, or Sharding

Oracle Container Database (CDB).
Each PDB can optionally be sharded for fault isolation and geo-distribution across multiple CDBs.

Multi Tenant Databases on Exadata Grid
Microservices and Multimodel Polyglot Persistence

Multi-model

Transparent access through JSON/REST and JDBC

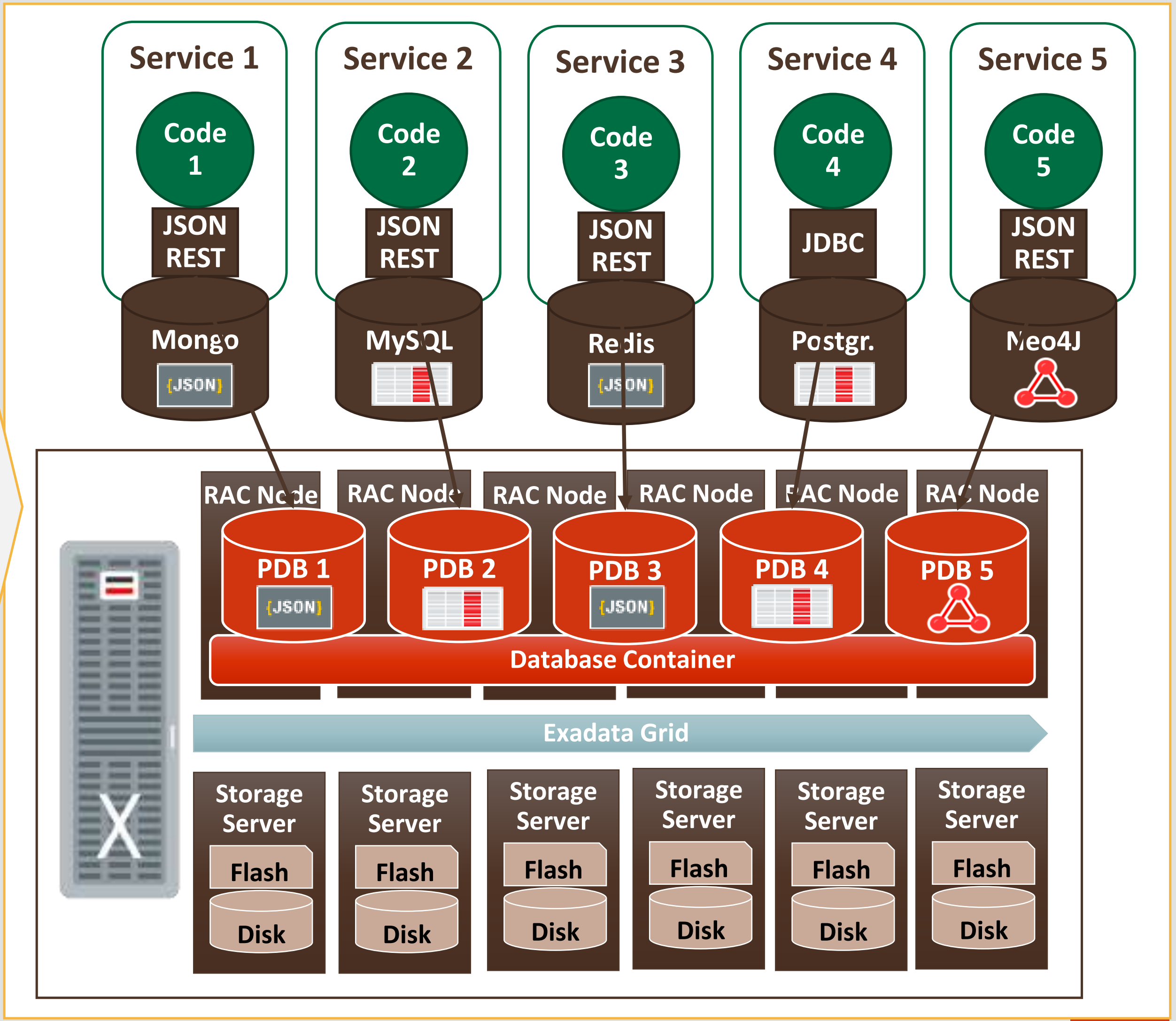
Multimodel Polyglot Persistence within the same database:

Relational Spatial NoSQL Graph XML OLAP

Multi-tenant

Enterprise Management and Operations

- Business Continuity
- Disaster Recovery
- On-Line Backup
- Enterprise Level Security



Ultra-High Availability for Microservices

Scalability, fault isolation and geo-distribution

Converged Database Architecture relies on the database (CDB) be highly available –
Exadata is great for this

Oracle 19c can also combine PDBs with sharding

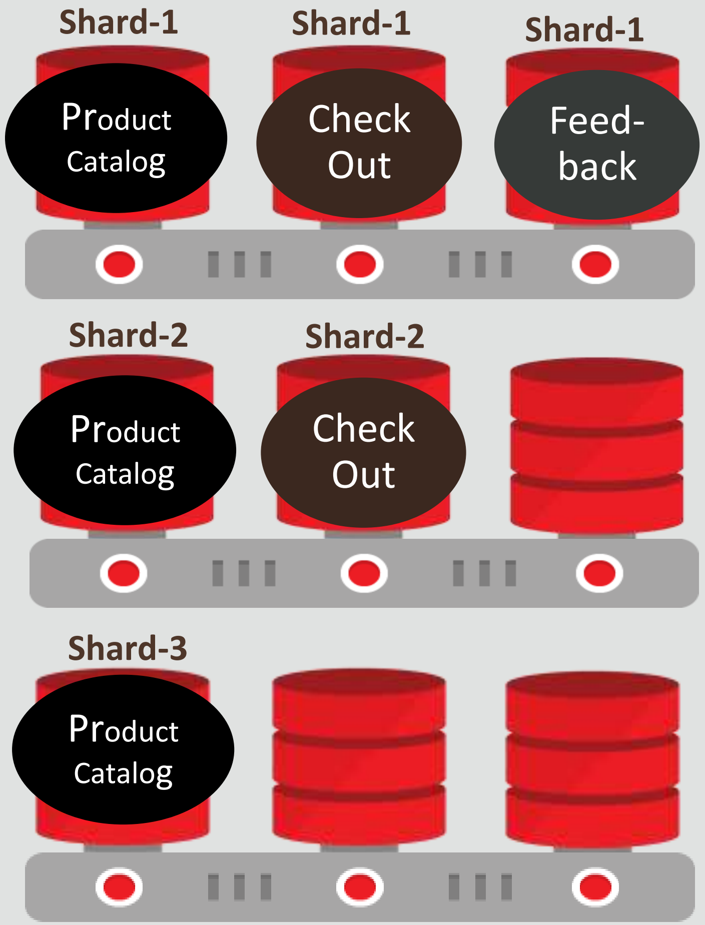
- Each PDB can be sharded individually across multiple CDBs

Fault isolation and geo-distribution for microservices

- Loss of an entire CDB makes only part of a PDB unavailable

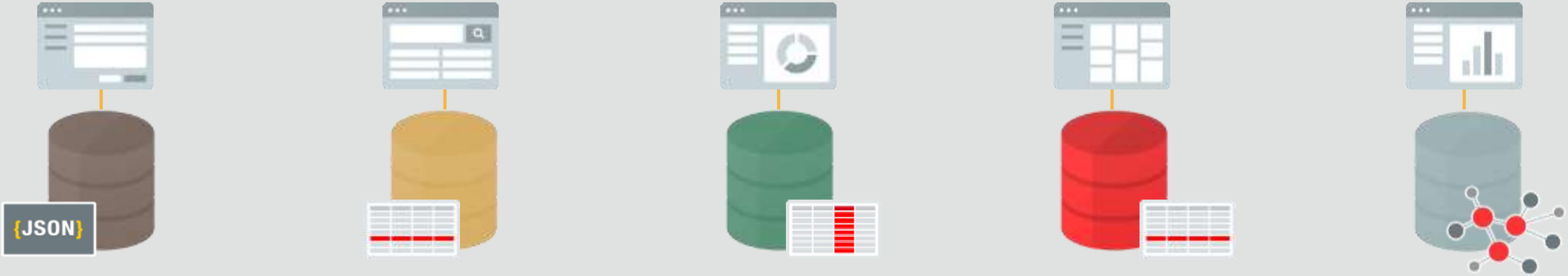
Also allows each microservice to scale its PDB individually

- More efficient than scaling entire CDB. Only scale the PDB needed by the microservice



Microservices Approach

- In microservices, applications are written as independent services, usually with their own database
- Each development team can rapidly develop and evolve their microservice
- However, integration of databases creates massive “macro-level” complexity



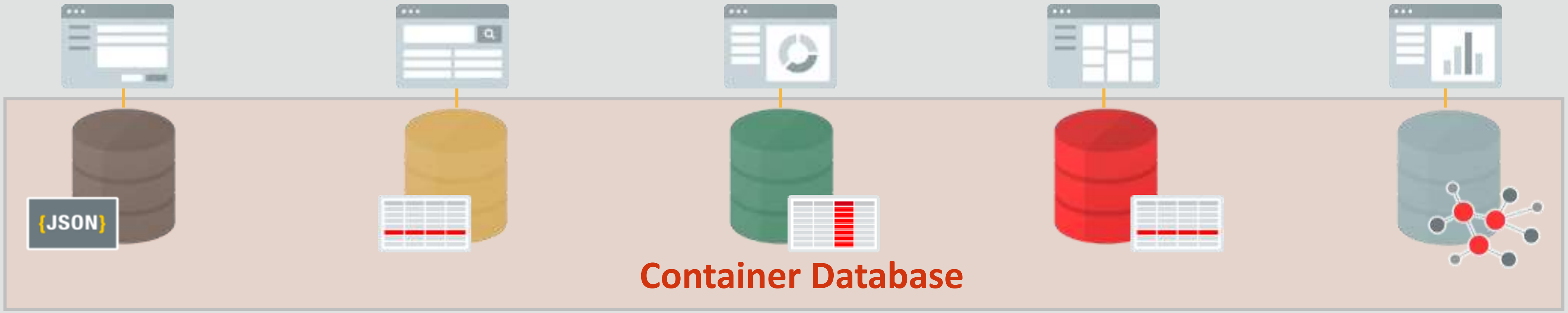
Convergeable Microservice Databases

- Convergeable Microservice Databases provide independence without integration complexity
  - Microservices are developed as if databases are separate
  - Developers focus on application logic rather than database integration

Convergeable Microservice Databases

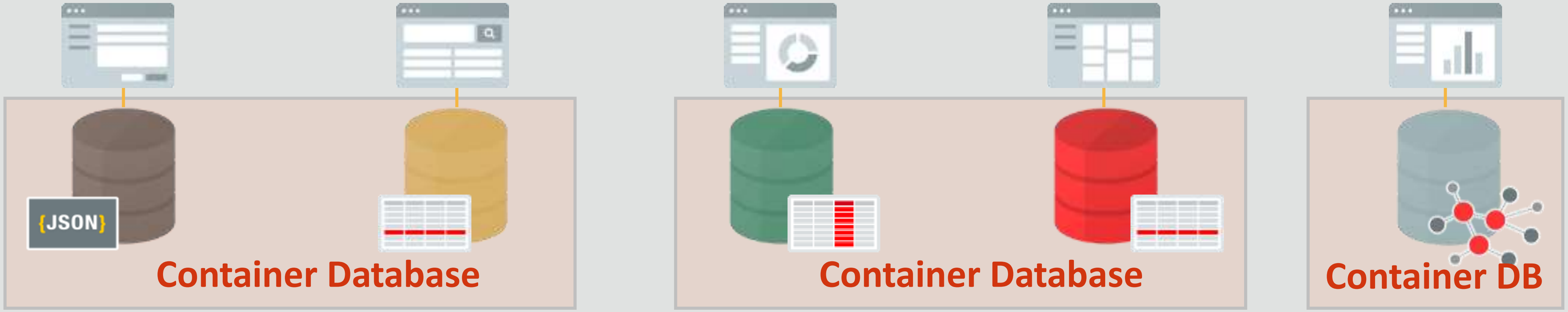
- Databases can be flexibly combined or separated

- Combining is enabled by the ability to converge many databases, data types, and workloads into one container database



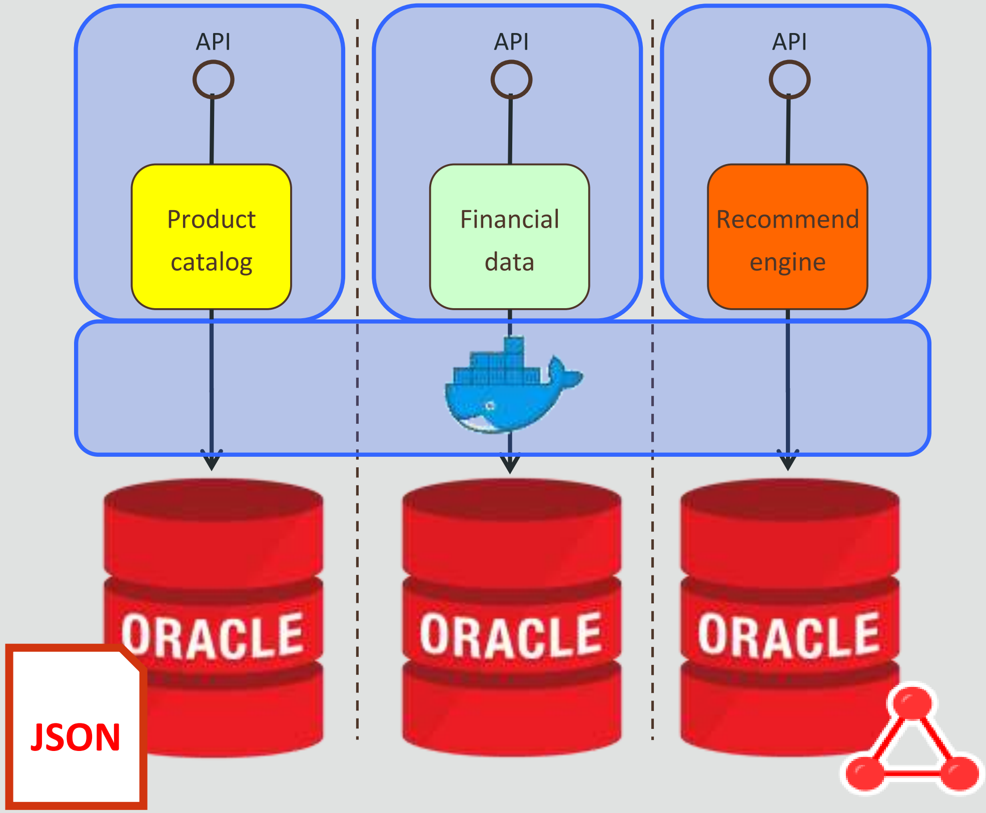
Separation of Microservice Databases

- Separation of databases is enabled by using Pluggable Databases that can be dynamically moved between physical container databases

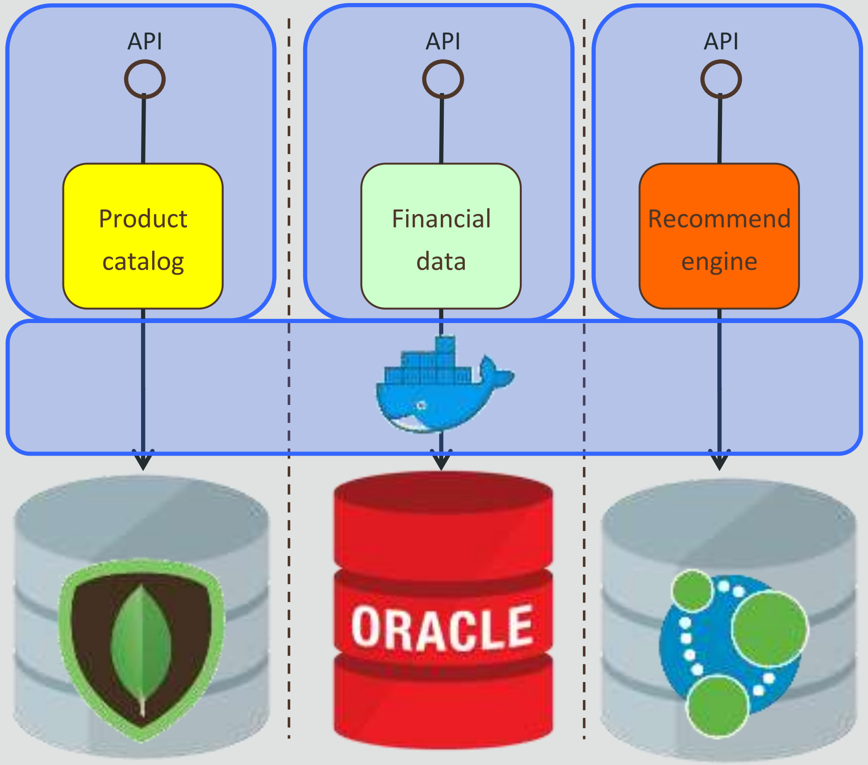


µServices and Containerization

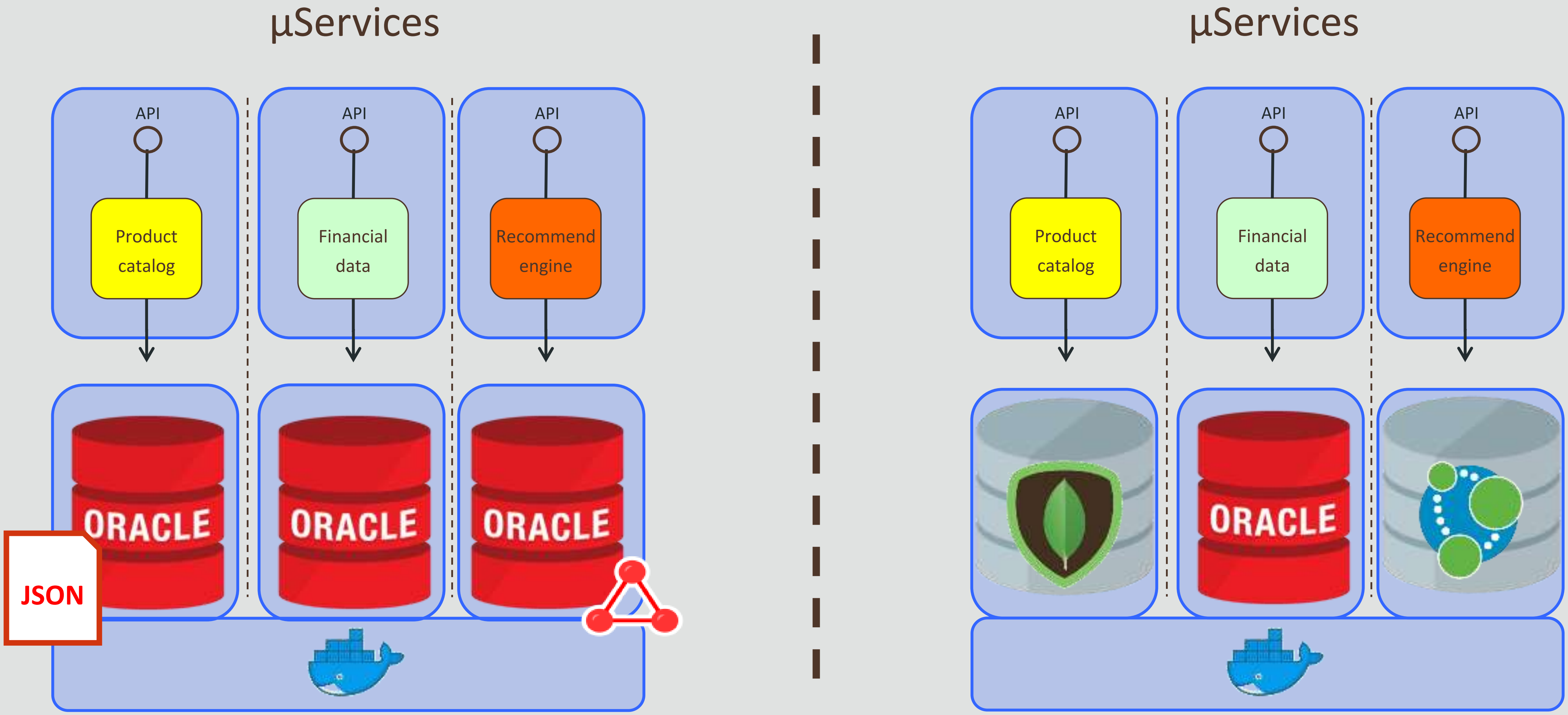
µServices



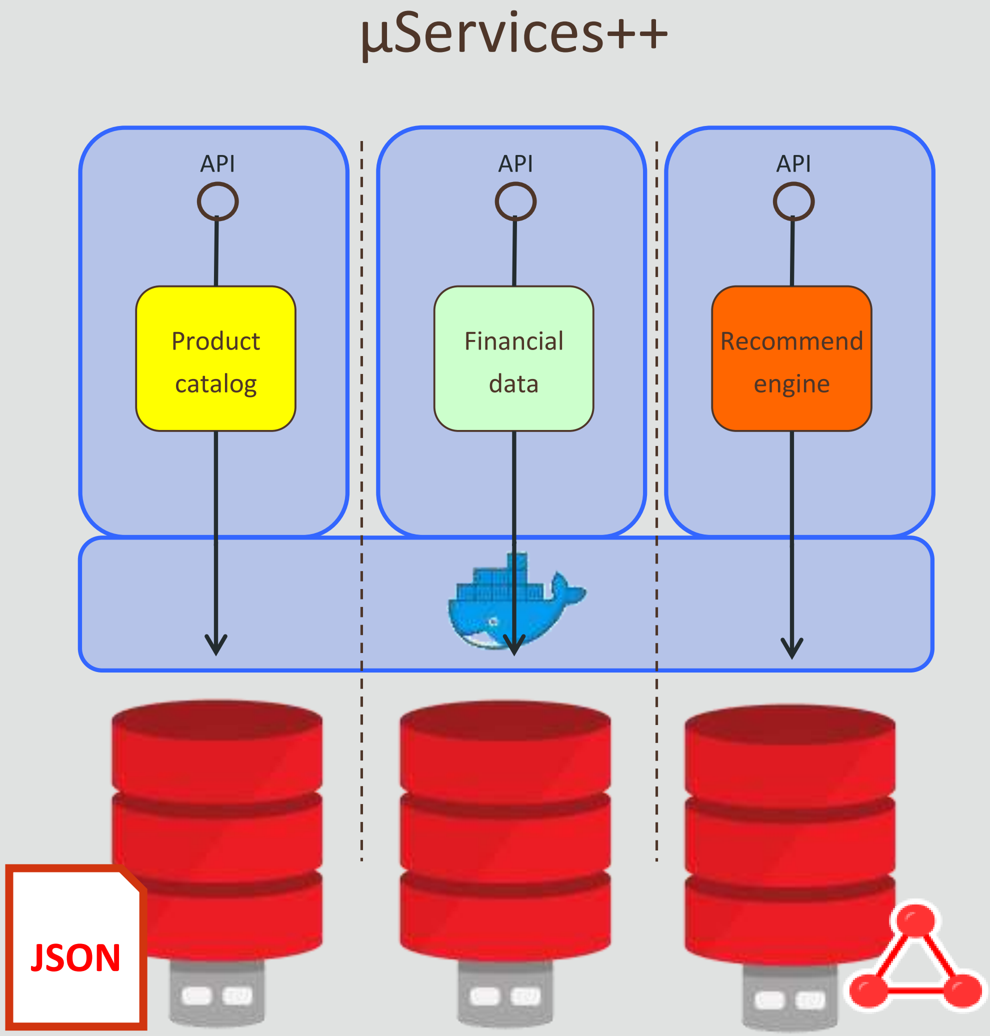
µServices

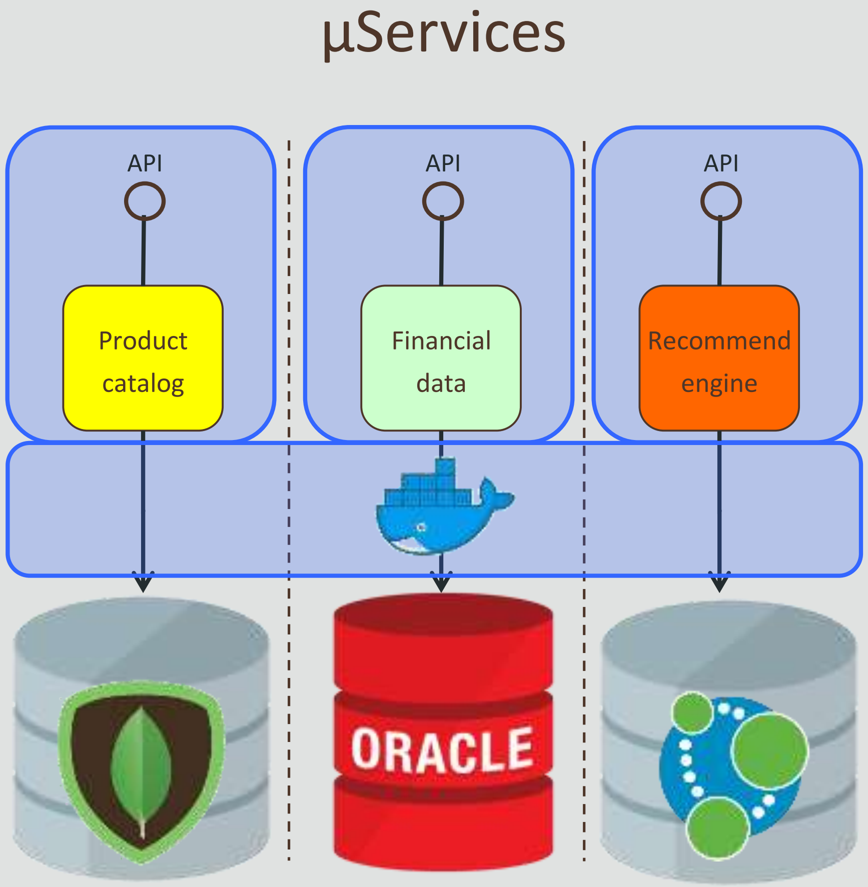


µServices and Containerization



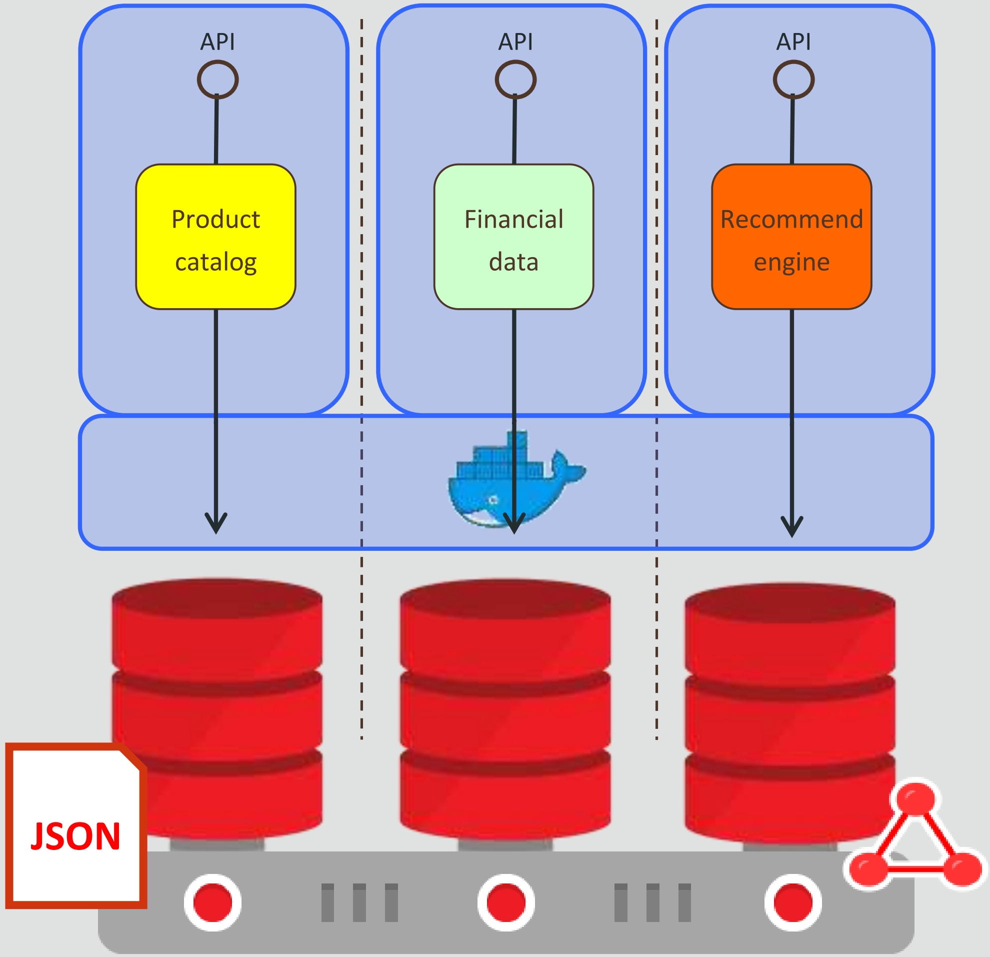
µServices and Containerization



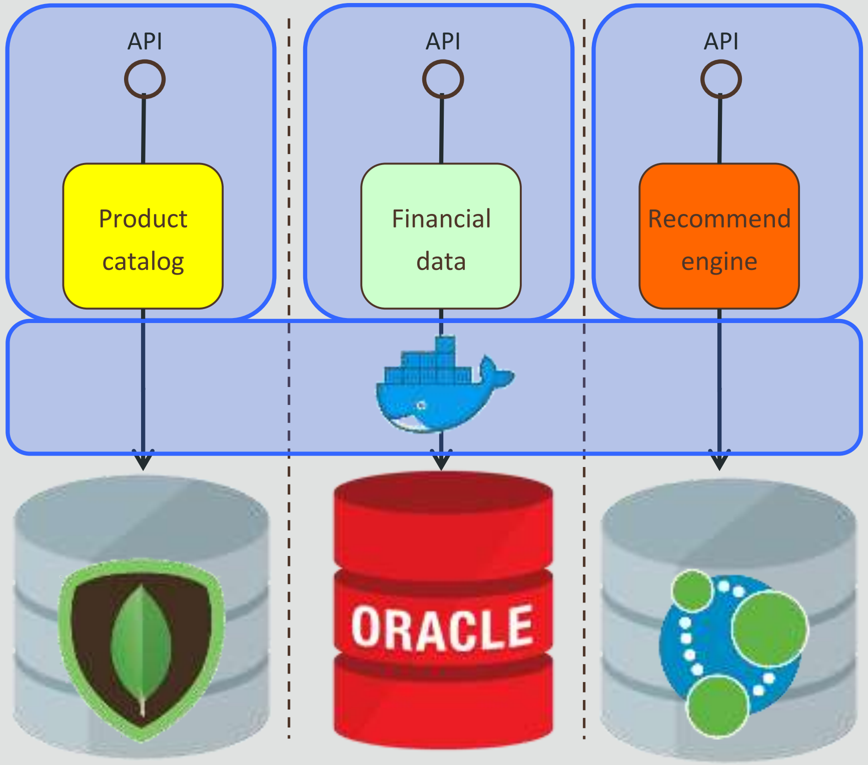


µServices and Containerization

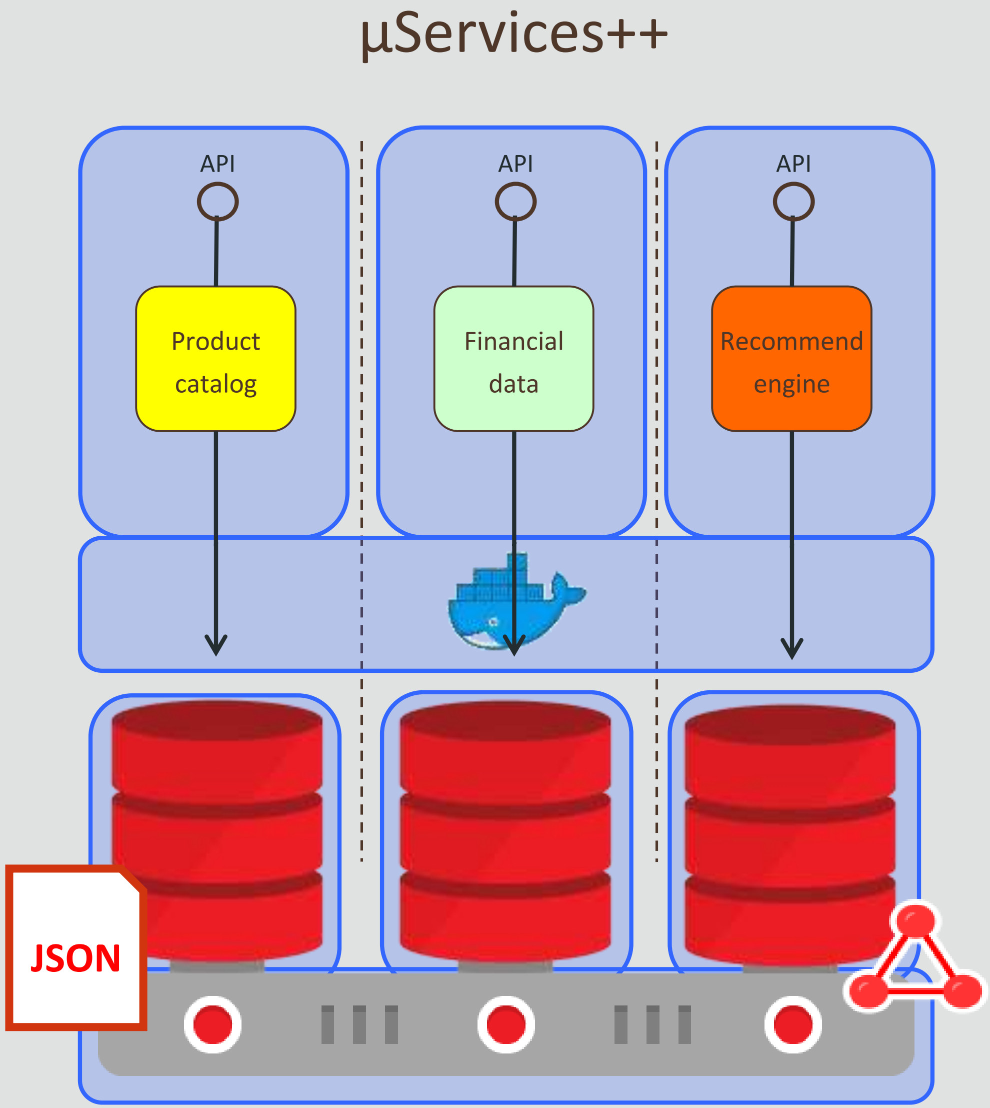
µServices++

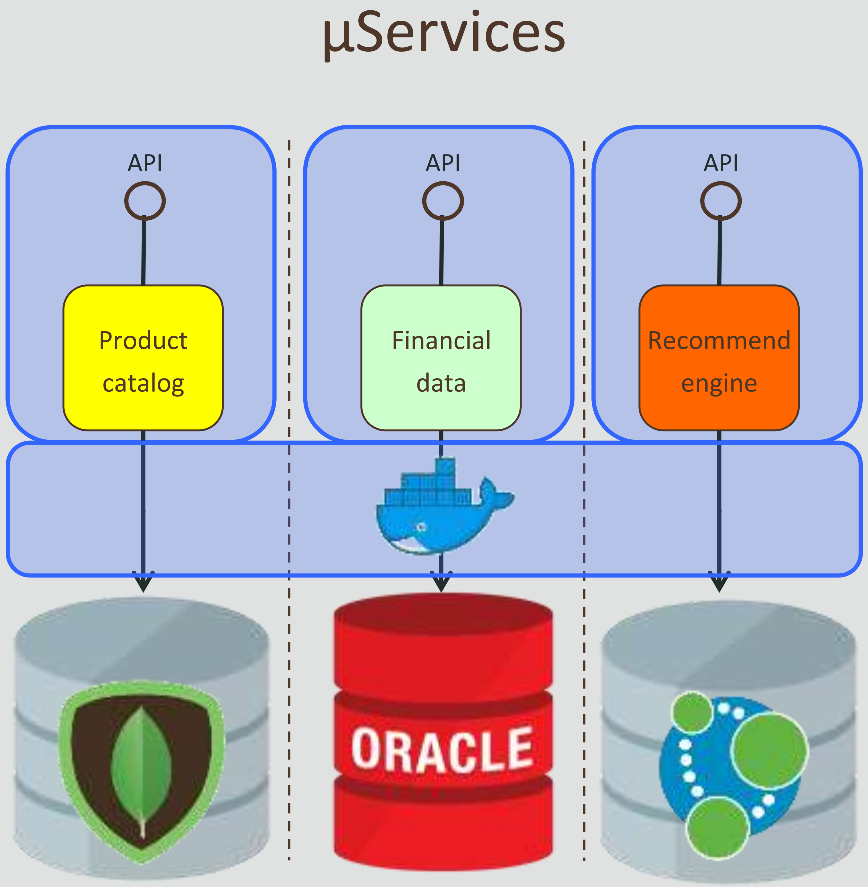


µServices



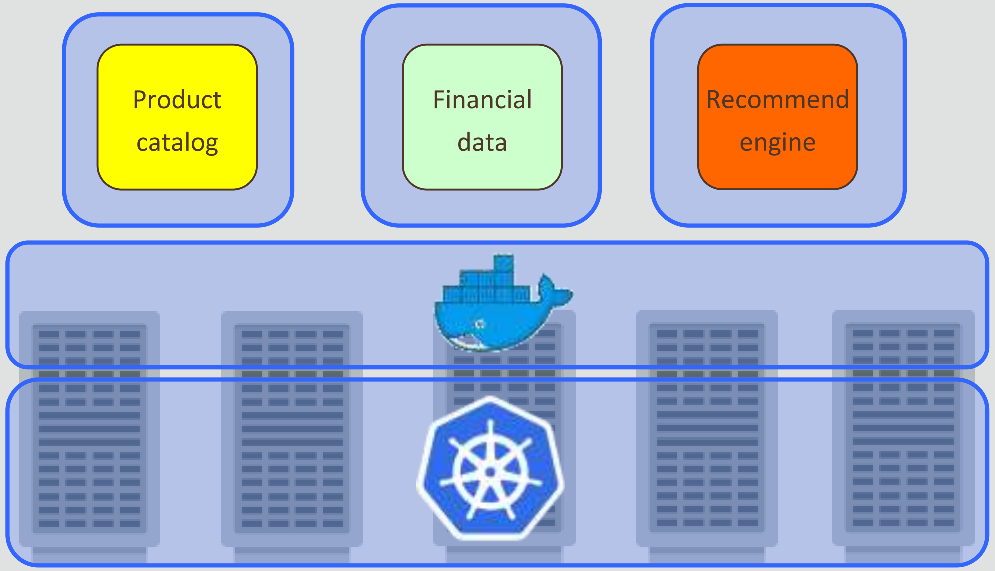
µServices and Containerization





µServices and Containerization

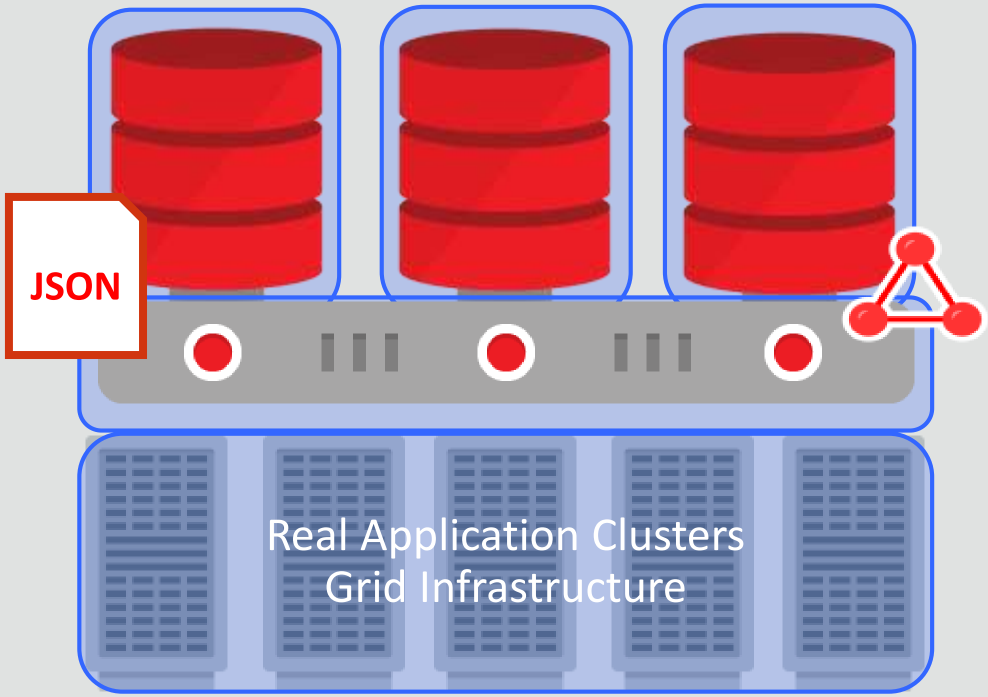
High Availability



Product catalog

Financial data

Recommend engine



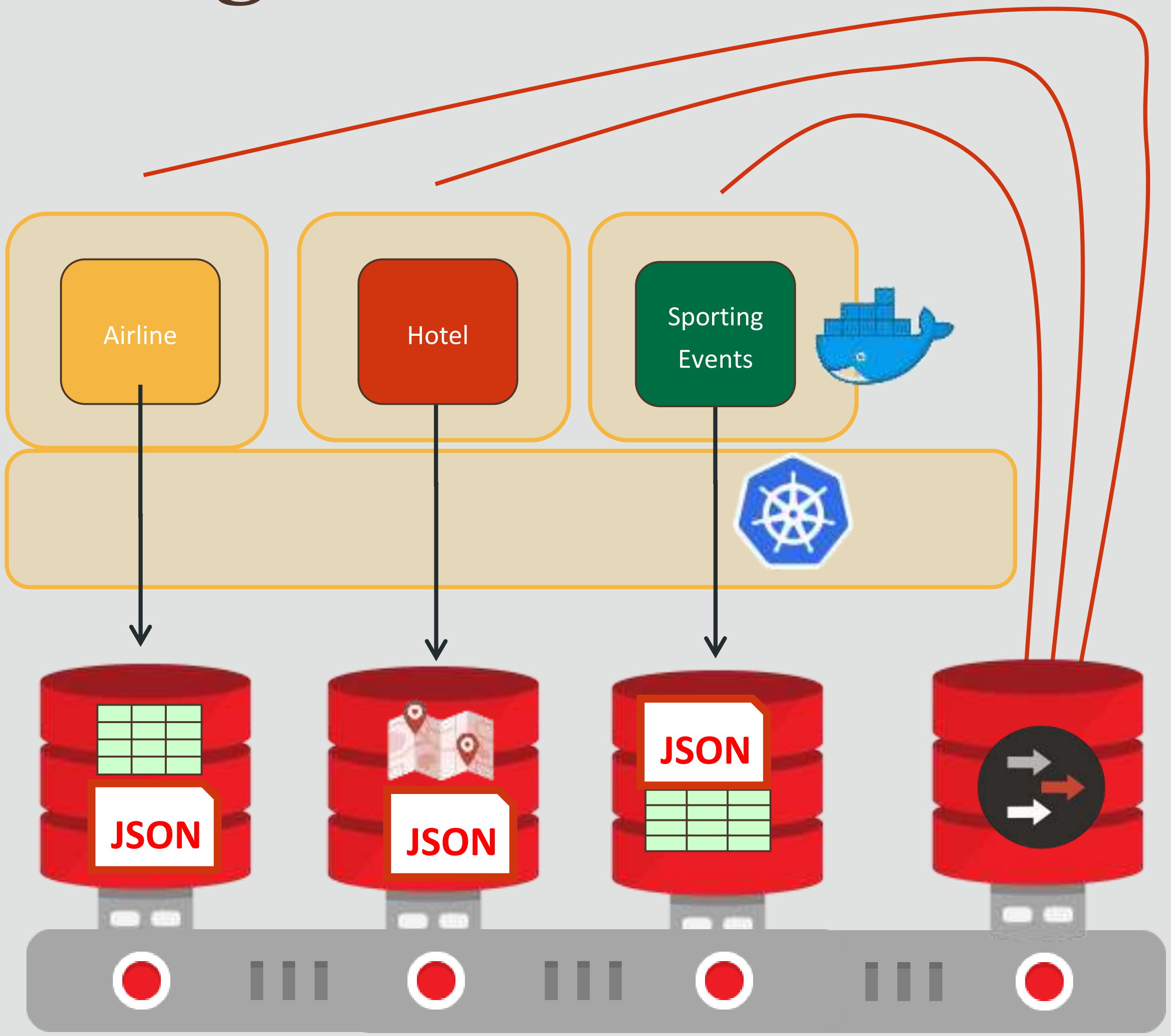
JSON

Real Application Clusters
Grid Infrastructure

Importance of Messaging

- The term ‘microservice’ may imply that you should look at the services first
- In fact, it is best to think of the APIs and messages first
- A microservices-based architecture is described by the interaction of messages. This provides the abstraction that allows each microservice to be developed and evolved independently
  - Provided the messages remain the same, you can replace a service by one or more other services transparently. This gives you resiliency and scalability
- The messaging system also simplifies the architecture
  - Instead of figuring out which microservice talks to which other microservice, they all use messaging to publish/subscribe to messages/events

Converged Database Architecture



Multimodel PDBs with Transactional Event Queues

kafka compatible

- Oracle Transactional Event Queue is a event streaming system built-into the Oracle database
- Supports JMS or Kafka APIs
  - Eliminates separate messaging infrastructure
  - Simpler and more secure
- Event Queues supports transactional messaging - **microservice state and events can be persisted by the same local transaction** (not 2 phase commit)
- Simplify development of fault-tolerant microservices
  - Error recovery logic is typically 90% of the code. And this code is often poorly tested

Why this matters

Creating and maintaining robust microservices is now easy and scalable with
Oracle's multi-model database with built-in Messaging

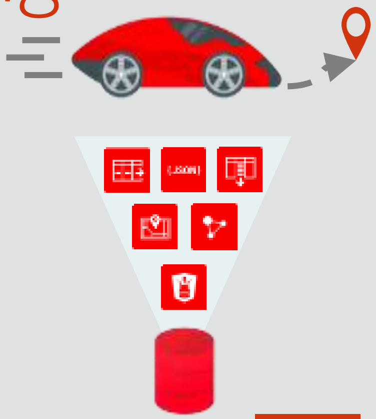
- ➢ Supports different data types
- ➢ Built-in Kafka-compatible transactional messaging layer
- ➢ Autonomous management
- ➢ Cloud scale up/down

Conclusion – Winning the War on Complexity and Cost

- I.T. has been costly and slow for decades
  - Root cause is complexity

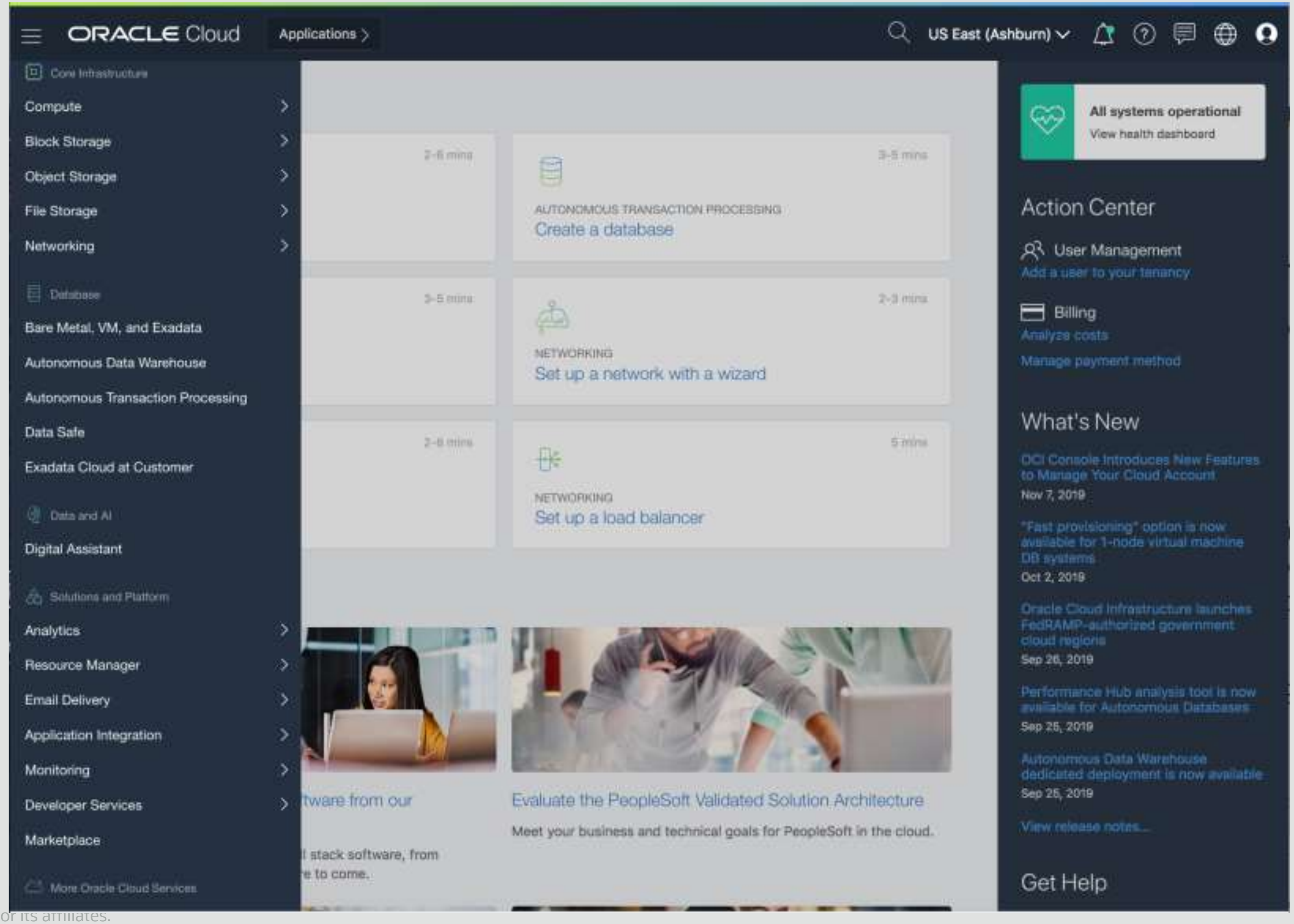
- New approaches can finally eliminate the sources of complexity

- Autonomous management, cloud, and machine learning eliminate the complexity of enterprise products



- A converged product with all needed features eliminates the complexity of systems integration

Use Oracle Database + Services on OCI



tware from our

stack software, from
e to come.

Evaluate the PeopleSoft Validated Solution Architecture

Meet your business and technical goals for PeopleSoft in the cloud.

All systems operational
View health dashboard

Action Center

User Management
Add a user to your tenancy

Billing
Analyze costs
Manage payment method

What's New

OCI Console Introduces New Features to Manage Your Cloud Account
Nov 7, 2019

"Fast provisioning" option is now available for 1-node virtual machine DB systems
Oct 2, 2019

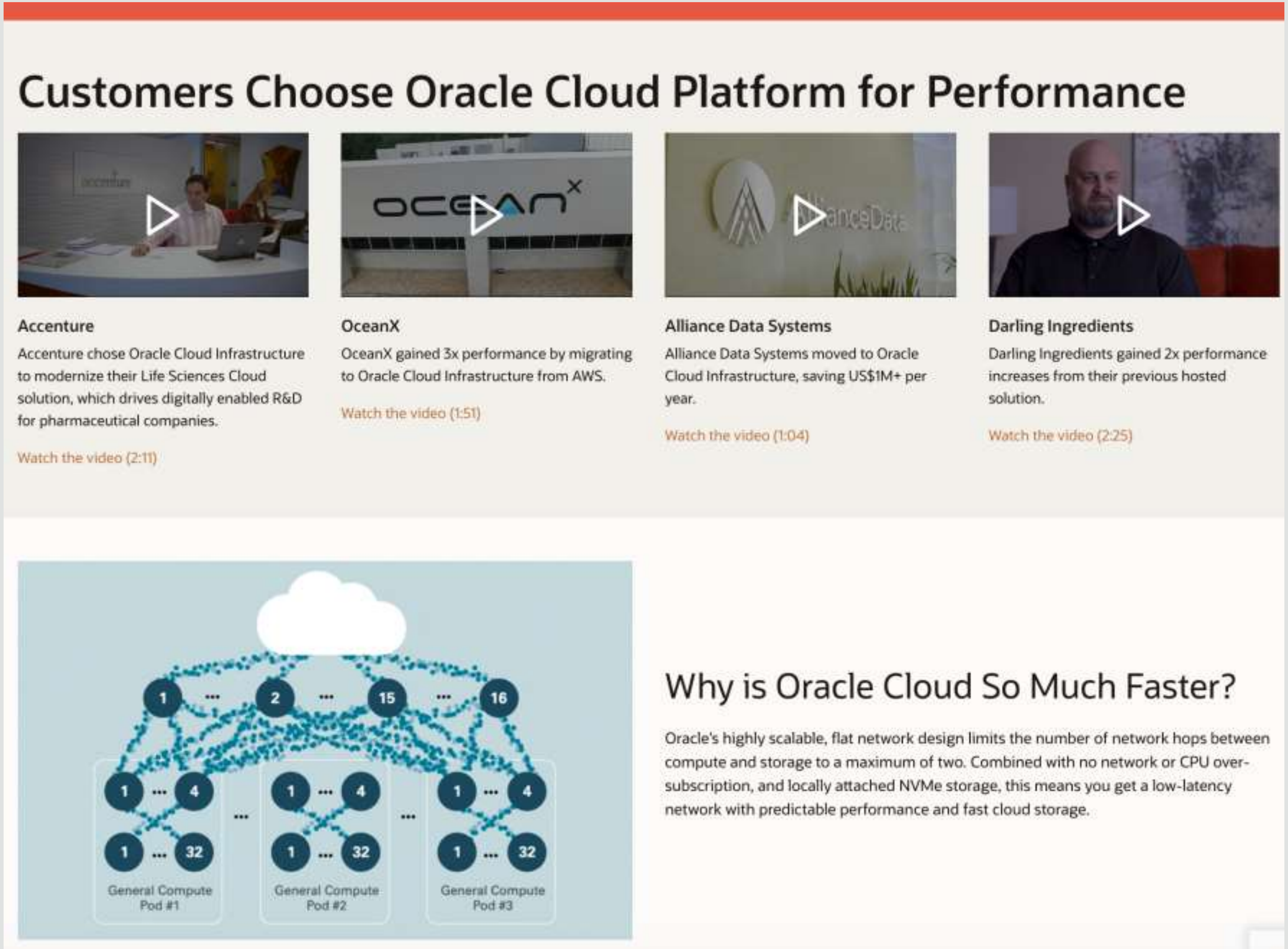
Oracle Cloud Infrastructure launches FedRAMP-authorized government cloud regions
Sep 26, 2019

Performance Hub analysis tool is now available for Autonomous Databases
Sep 25, 2019

Autonomous Data Warehouse dedicated deployment is now available
Sep 25, 2019

View release notes...

Get Help



Customers Choose Oracle Cloud Platform for Performance

Accenture

Accenture chose Oracle Cloud Infrastructure to modernize their Life Sciences Cloud solution, which drives digitally enabled R&D for pharmaceutical companies.

Watch the video (2:11)

OceanX

OceanX gained 3x performance by migrating to Oracle Cloud Infrastructure from AWS.

Watch the video (1:51)

Alliance Data Systems

Alliance Data Systems moved to Oracle Cloud Infrastructure, saving US$1M+ per year.

Watch the video (1:04)

Darling Ingredients

Darling Ingredients gained 2x performance increases from their previous hosted solution.

Watch the video (2:25)

Why is Oracle Cloud So Much Faster?

Oracle's highly scalable, flat network design limits the number of network hops between compute and storage to a maximum of two. Combined with no network or CPU over-subscription, and locally attached NVMe storage, this means you get a low-latency network with predictable performance and fast cloud storage.

Thank you