ORACLE

The right App Development paradigm

Navigating between Monoliths and Microservices

Sid Joshi

Business Development Director

EMEA OCI Platform Services | Application Development

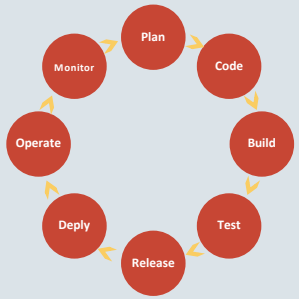
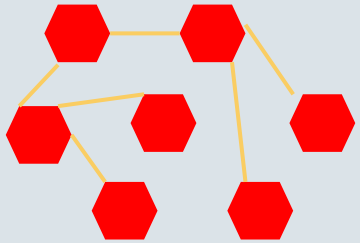
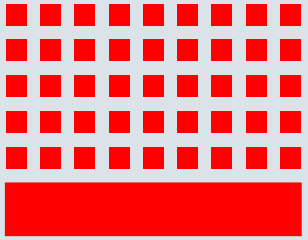
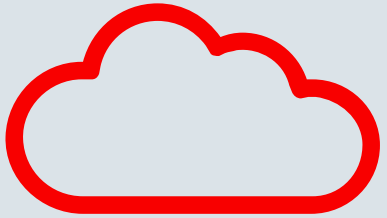
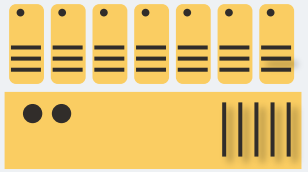
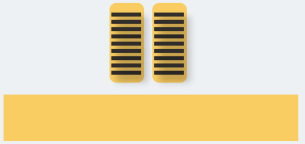
www.linkedin.com/in/sid-joshi

@SidJoshi_uk

Safe harbor statement

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.

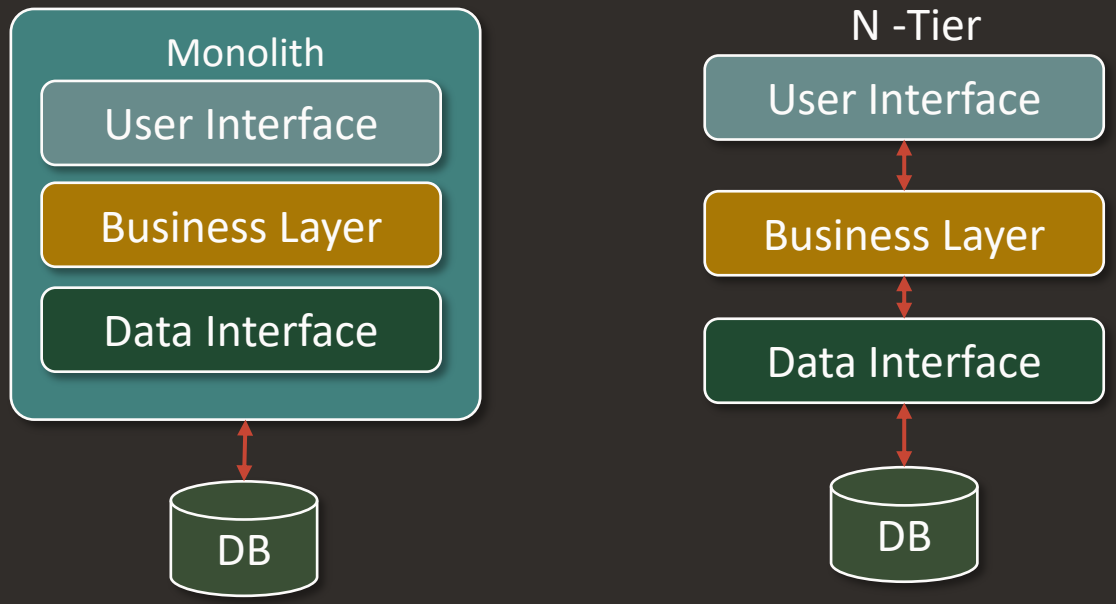
Evolution of Development and Deployment

	Development Process	Application Architecture	Deployment and Packaging	Application Infrastructure
Now ~2010	DevOps 	Microservices 	Containers 	Cloud 
~2000	Agile	N-Tier	Virtual Servers 	Hosted 
~1980 ~1990	Waterfall	Monolithic 	Physical Server 	DataCenter 

Monolith / Traditional Architecture

What is it?

- An Application built as a Single and Coherent Unit
- All functions are managed and served in one place
- Client side, server side, business logic and database as indivisible unit
- Modular approach like SOA



Application Characteristics

- Tight coupling
- Steady Workload
- Often Stateful / Transactional
- Applications long lived
- Infrequently updated
- Scaling confines
- Wider range of functionality
- Underpinned by Application Servers providing wide range of capabilities & automation

Monolith / Traditional

Advantages

- Higher performance
- Less chatty
- Transactional consistency
- Easier to debug and test as a complete application
- Simpler Deployments
- Lower operating overhead costs
- Much less cross-cutting concerns like security, logging, caching, memory management
- Easier to implement
- Suitable for small - medium teams / projects

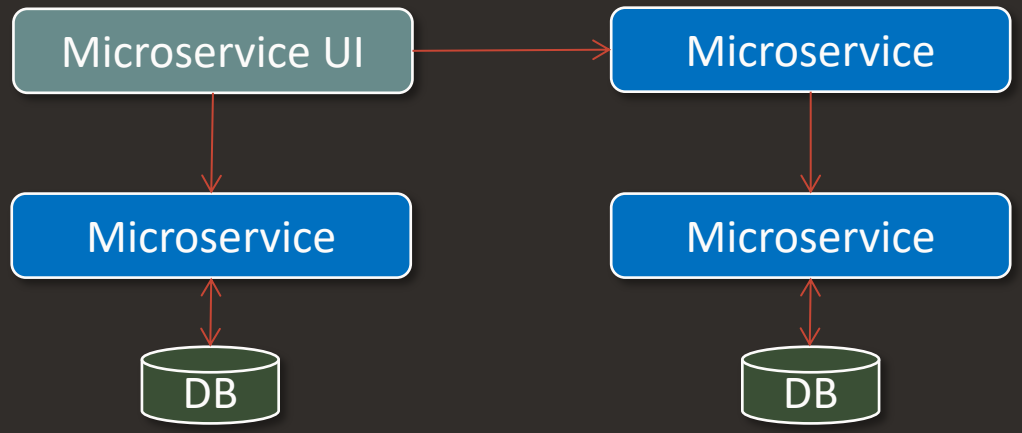
Disadvantages

- Tight coupling
- Low flexibility
- Scaling confines
- Slower update cycle
- Difficult to change implementation choice
- Performance impact on 1 function can have domino effect
- Harder to distribute development
- Environment consistency

Microservices / Serverless Architecture

What is it?

- Business logic is broken down into lightweight, single-purpose self-sufficient services
- Small discrete capabilities
- Externalize **all** state
- Usually packaged in containers
- Offload most management



Application Characteristics

- Idempotent
- Loose coupling
- Unpredictable Workload
- Often Stateless
- Short lived
- frequent updated
- Scaling confines potentially
- Discreet functionality

Microservices / Serverless

Advantages

- Autonomy / Decoupling
- Agility
- Quicker turnarounds
- Scalability
- Reliability
- Continuous delivery

Disadvantages

- Designing distributed systems can be challenging
- Microservice architecture requires more resources and usually takes more time
- Handle partial failure—no transaction safety (Eventual consistency)
- Cross-Cutting Concerns with microservices – security, logging, caching needs to be taken care of in every service
- Complex deployment
- Complex End to End Testing
- Higher operating overhead costs
- Chatty Solution
- Time-consuming monitoring
- Time-consuming troubleshooting

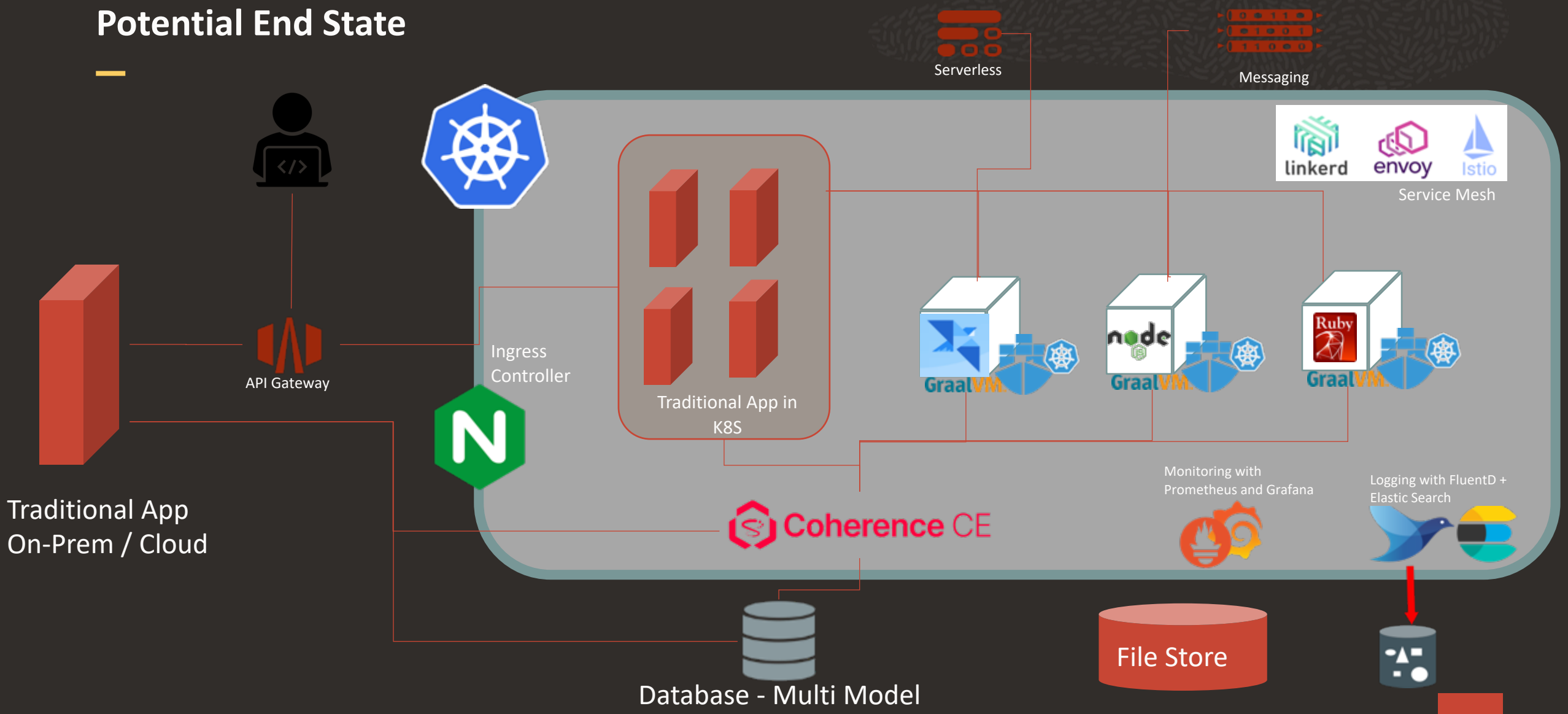
Microservices or Traditional: Choosing the right approach

	Traditional architecture	Microservice architecture
Coupling	Tight	Loose
Code Complexity [as a unit]	High	Medium
Development Time	High	Low
Release Cycle	Slow [Monthly]	Fast [Hourly / Daily]
Development	Teams are involved in the development process simultaneously	Different teams can work on different elements of the solution in parallel
Data Consistency	Transactional consistency	Eventual consistency
Language Adoption	Difficult to implement different programming languages	Possibility to use different languages, technologies for different business needs
Deployment	Deploy an entire system once, adjust as needed	Possibility to deploy (and rollback) each microservice individually
Updates	Updates might take a while because of internal dependencies within the architecture and other developers working at the same time	Fast updates due to the minimalistic nature of modules due to the autonomous nature of services
Testing	Comparatively simpler end-to-end testing and automation	Each component needs to be tested individually, difficult to achieve coherent end to end solution testing

Microservices or Traditional: Choosing the right approach

	Traditional architecture	Microservice architecture
Costs per Transaction	Low	Low to Medium
Workloads	Steady / Predictable	Dynamic with quick response times
Processing Ownership	Dedicated	Semi – Shared
Maintenance	Application Language & Traditional skills are required	Application Language, DevOps, Cloud Native [Docker, Kubernetes, Service Mesh, Prometheus, Elastic Stack, etc] skills are required
Reliability	One failure may have higher impact on the wider system	A failure of one service doesn't affect other services
Scalability	Medium - Minutes per Instance to Scale, Whole app as a unit	Low – Seconds to scale, individual services
Management	Medium Complexity	High Complexity [Multiple tools, multi faceted approach, steep learning curve]
Application Security	Smaller attack surface area, simplified to set up Identity and access management and testing of the wider solution	Greater complexity = expanding attack surface: Segmentation & Isolation, Identity management and access control, data management, container security
Software Security	Enterprise software vendor providing frequent security vulnerability fixes	Onus on customer of tacking vulnerability and fixes, Stricter Discipline
Capacity Utilisation	Typically Oversized for peak volumes	Best use of Capacity with automated scaling

Potential End State



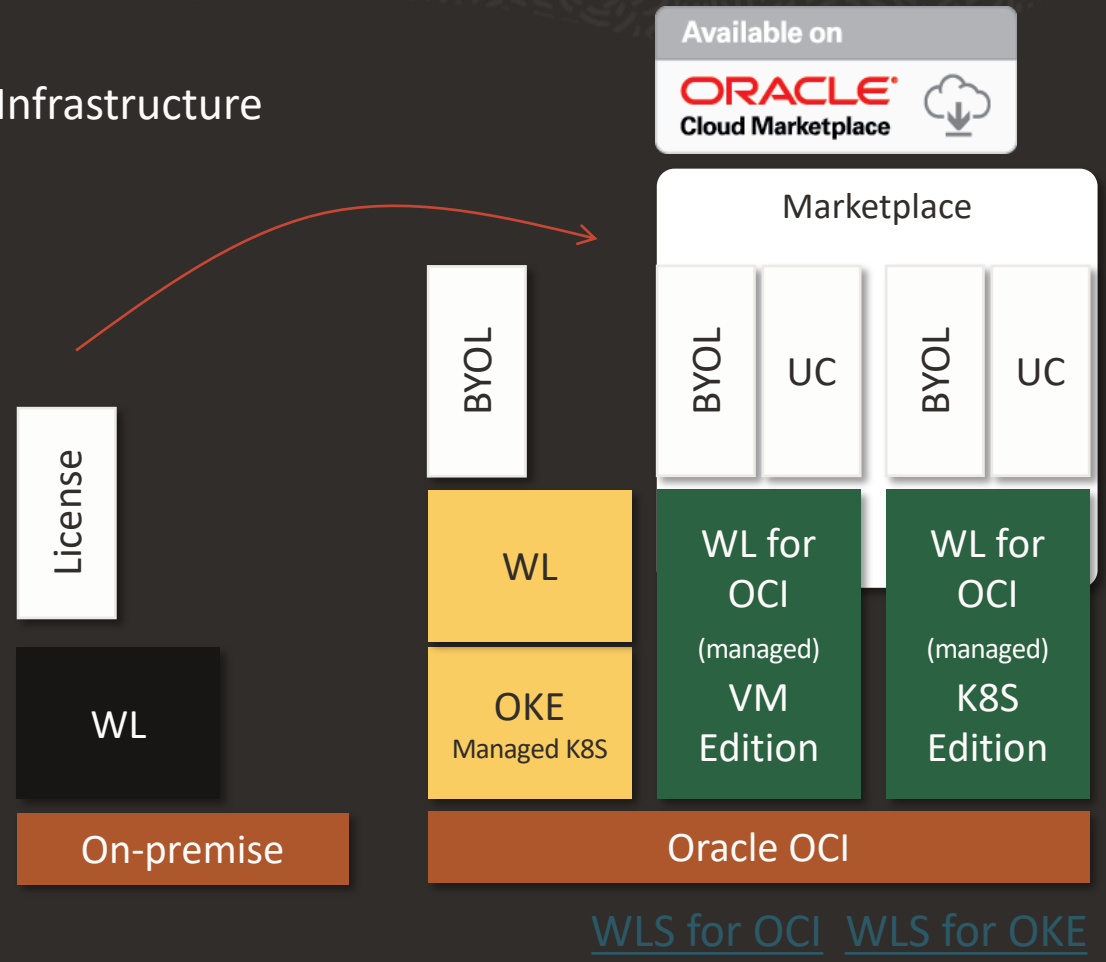
Oracle WebLogic Server for Oracle Cloud Infrastructure

Customer managed Oracle WebLogic Server for Oracle Cloud Infrastructure

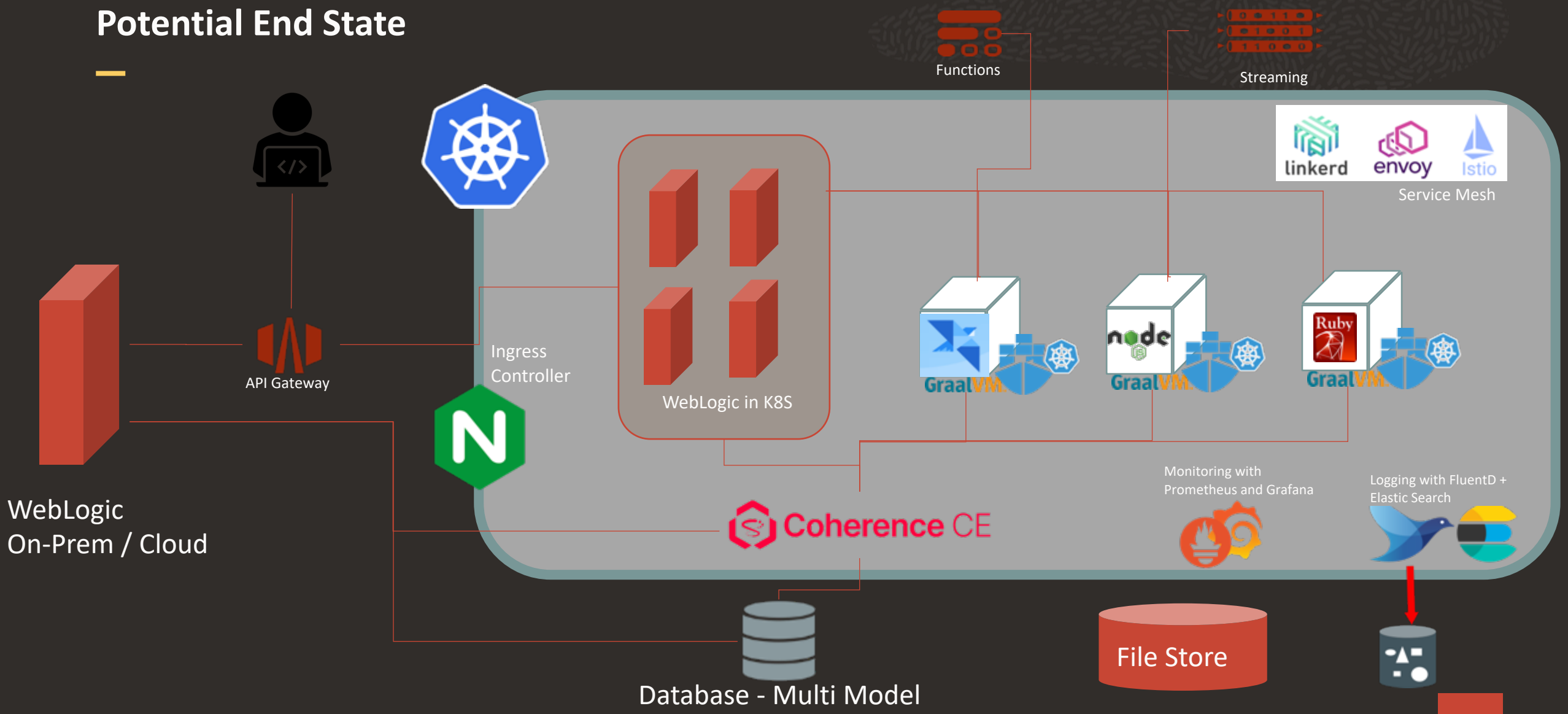
- **Bring Your Own License (BYOL)**: 3 Options
  - Standard, Enterprise and Suite
- **Universal Credit (UC)**: 2 Options
  - Enterprise and Suite

Supports

- WebLogic Server **11g** (10.3.6) [VM]
- WebLogic Server **12c**
  - 12.2.1.3 [VM]
  - 12.2.1.4 [VM, OKE]
- Supports JRF and Non-JRF domains
- Supports ATP DB and OCI DB as infra DB



Potential End State

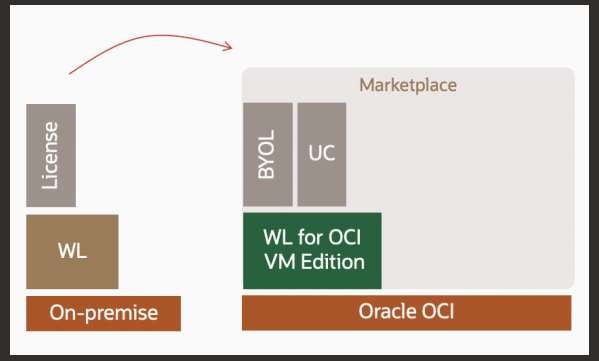


Types of Deployment

Lift & Shift [Rehost]

Traditional Applications on Cloud

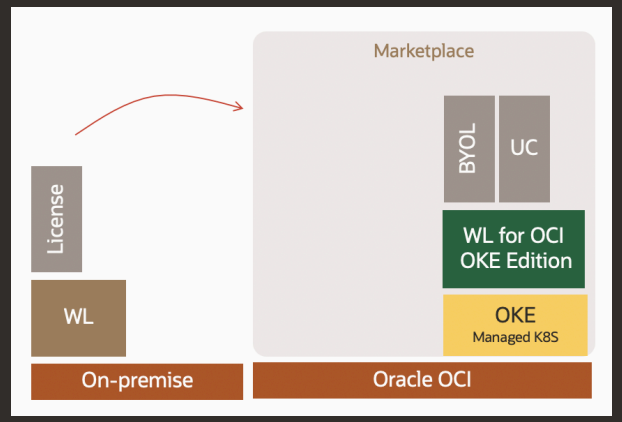
- Automation
- Scaling
- Resilience
- Environment Consistency



Lift & Shift [Re-platform]

Traditional Applications in Cloud Native

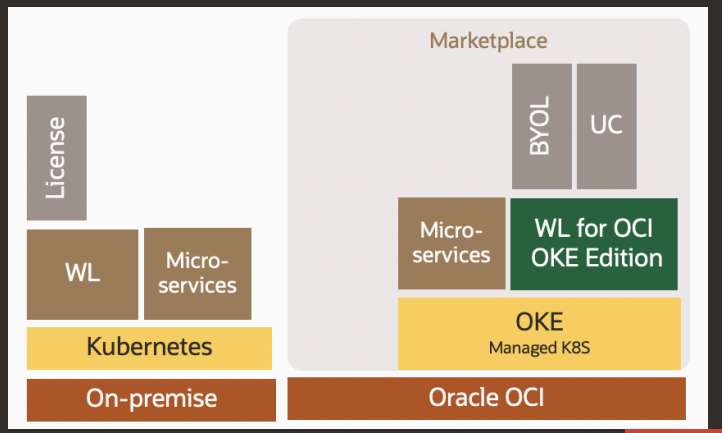
- Benefits of Cloud Native
- Pathway to modernisation



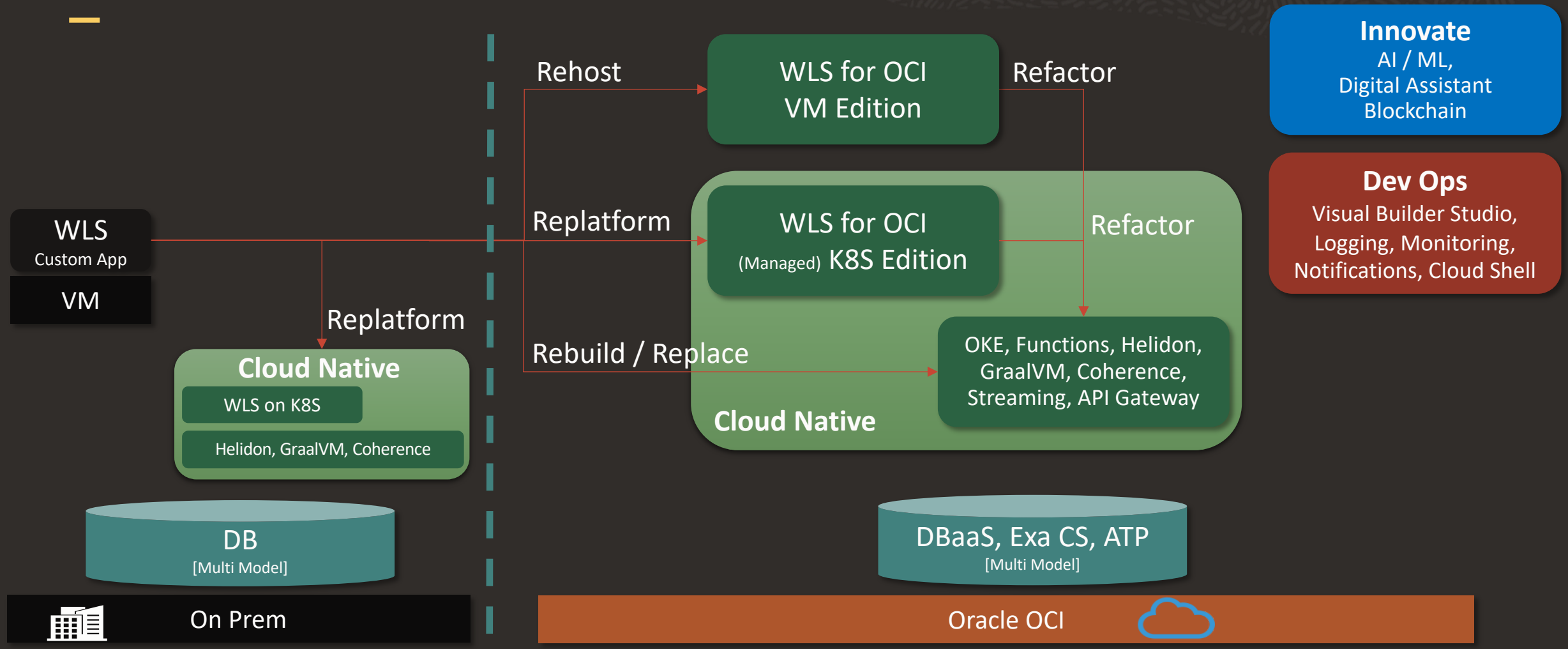
Hybrid Architecture

[Traditional + Microservices]

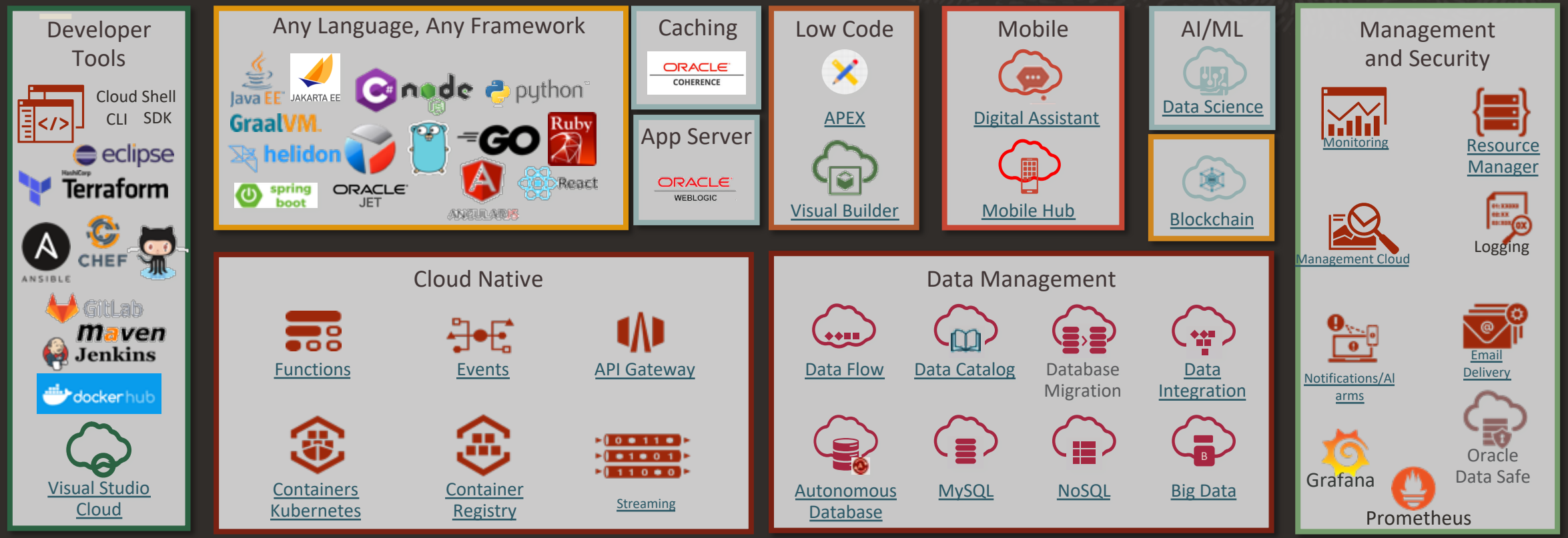
- Best of both worlds
- Modernise the solution at your own pace
- Get the most of existing investment while forging the pathway to innovate



Modernization Pathways: Application Modernization



Oracle Cloud for Application Development



OCI (Global IaaS)

Regions

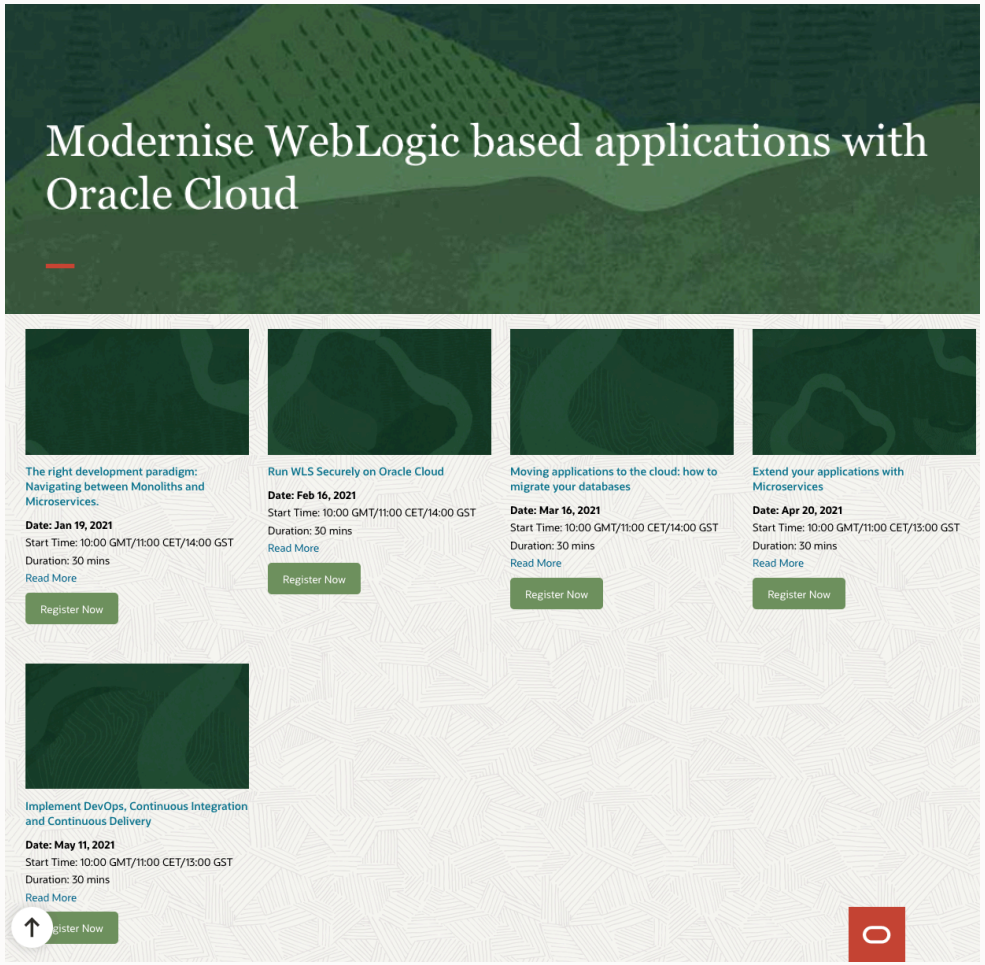
Availability Domains

Networking | Compute | Storage | LBaaS | Edge | IAM | Key Mgmt | API / CLI

Infra as a Code (TF, Chef, Puppet)

On Prem / C@C

Upcoming EMEA events



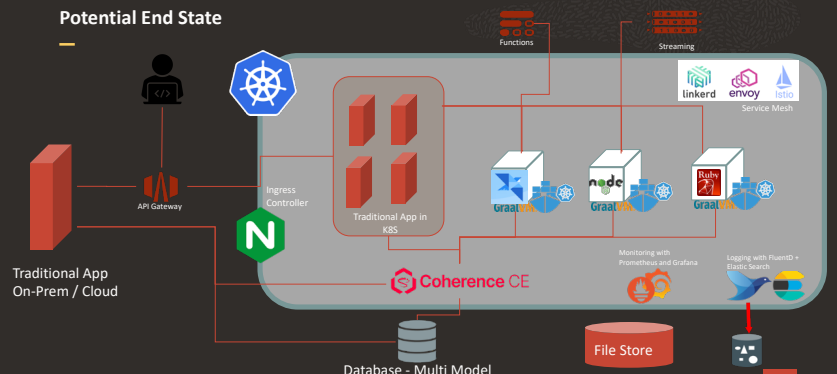
https://go.oracle.com/LP=101830?elqCampaignId=274053

Summary

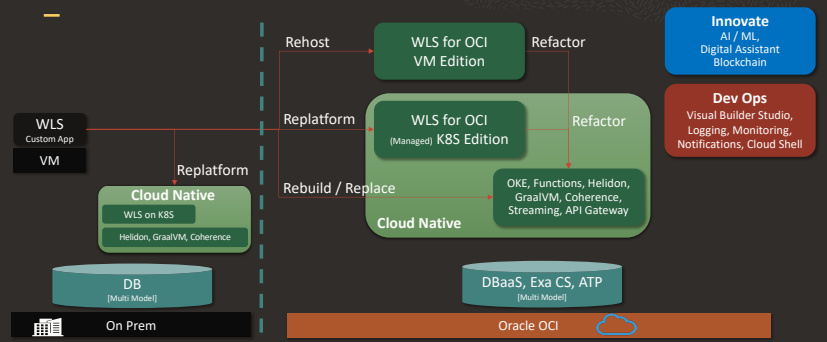
Microservices or Monolith: Choosing the right approach

	Monolithic architecture	Microservice architecture
Coupling	Tight	Loose
Application Evolution	Slow [Monthly]	Fast [Hourly / Daily]
Code Complexity [as a unit]	High	Medium
Delivery Complexity	Low	Low to Medium
Management	Low to Medium	Medium
Cost	Low per transaction	Low to Medium per transaction
Scalability	Low - Minutes per Instance to Scale	Good – Seconds to scale
Workloads	Steady / Predictable	Dynamic with quick response times
Processing Ownership	Dedicated	Semi – Shared
Data Consistency	Transactional consistency	Eventual consistency

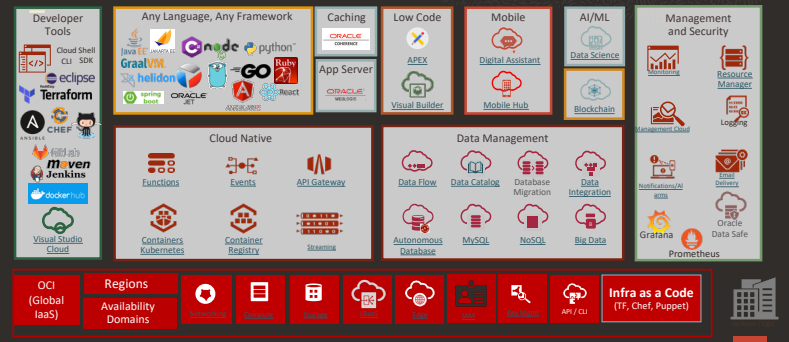
Potential End State



Modernization Pathways: Application Modernization



Oracle Cloud for Application Development



Thank You!

Sid Joshi - Business Development Director
EMEA OCI Platform Services | Application Development

www.linkedin.com/in/sid-joshi
@SidJoshi_uk

ORACLE