ORACLE
Cloud

Orchestrating Itinerant Services

OpenAlt 2019

Jan Holčapek, Michael Stoker
Oracle GBUs, Cloud Foundation Services
November, 2019

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

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.

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Abstract

In this presentation we will discuss some of the issues encountered when moving pre-existing S/W applications to a cloud framework where traditional certainties and freedoms of server-based deployment don't apply.

In our Cloud Native world, compute instances are being continually refreshed in a continuous cycle, and so applications (services) wander from instance to instance, running for some period before moving on, while all the time providing on-going service.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Program Agenda

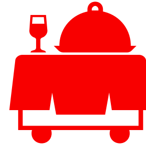
1. Introduction
2. Historical Problem Background
3. Our Cloud Native Principles for Micro Service Devs
4. (Re-)Designing Services for Cloud Native Deployment
5. Cloud Native Operational Examples
6. Summary and Questions

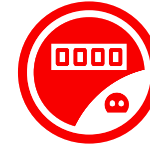
ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Cloud Native: Cloud Foundation Services

- Build/operate container orchestration services for Oracle GBUs





- Use Kubernetes as part of a complete container orchestration framework that runs on/alongside Oracle Cloud Infrastructure (OCI)
  - CI/CD integration with GBUs organized into independent projects
  - Services and APIs: Authentication, Authorization, Logging, Monitoring, Trust Management, Volume Storage, …

- DevOps teams in North America/Brno/South Asia/Oceania
  - support dev/prod clusters in North America and Europe (24x7)

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Historical Application Deployment Background

- Extensive use of n-tier architecture
- Large, complex, silo-ed, S/W stacks
  - file-based storage, extensive DB usage
  - customized kernels, customized OS services, customized applications
  - static both in location and sometimes content
  - tightly coupled to infrastructure
- Physical separation of functions across servers
  - servers/network/storage tailored application load, also silo-ed (little if any sharing)
- Typically on-premises

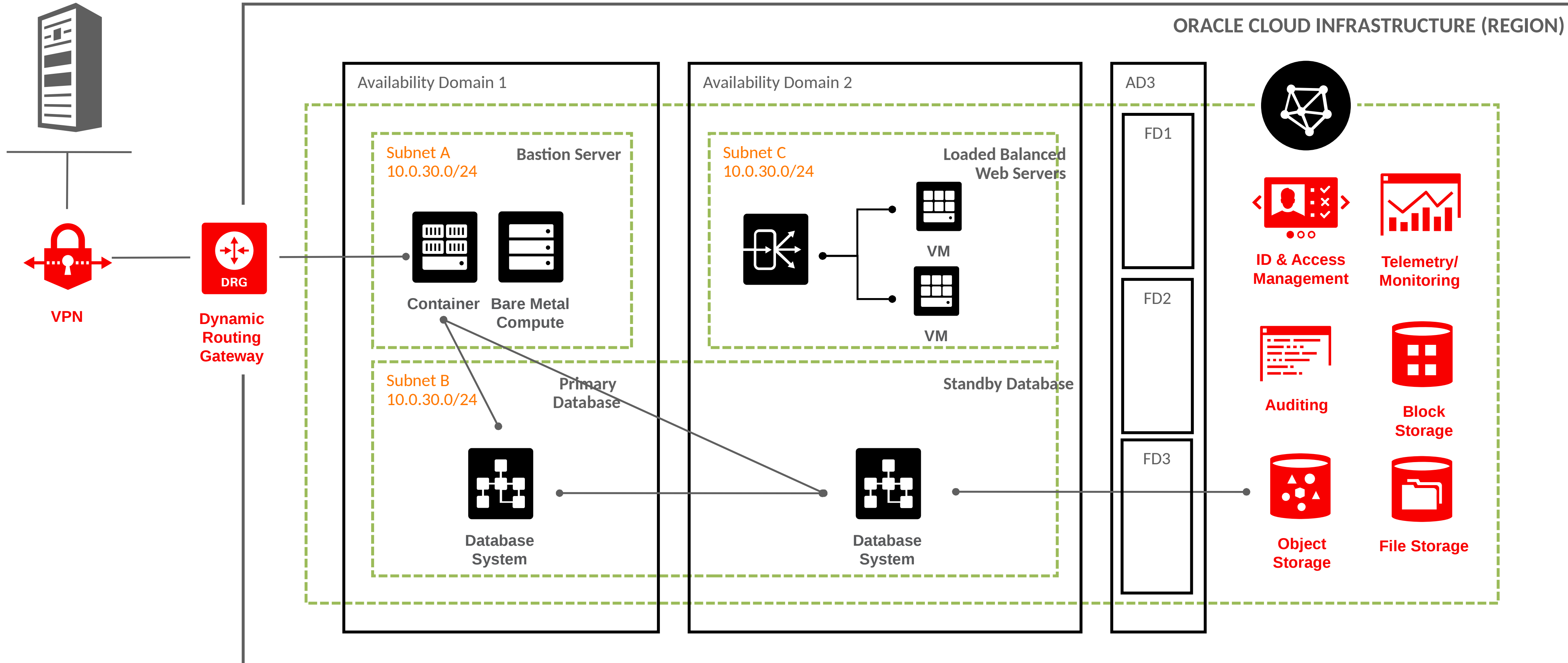
- Plus
  - much industry experience with development/operating/debugging/tuning/HA
  - automation via configuration mgt/scripting
  - predictable, dependable, tweakable
- Not so Plus
  - long install/bring-up times
  - expensive to roll out and to change (e.g. scale)
  - update in-place (likely downtime)
  - conflicts between dev/ops groups
  - (skilled) people intensive
- Scope:
  - applies to GBU partners, we are no exception too!

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Oracle Cloud Infrastructure (Concepts)

ORACLE
Cloud Infrastructure

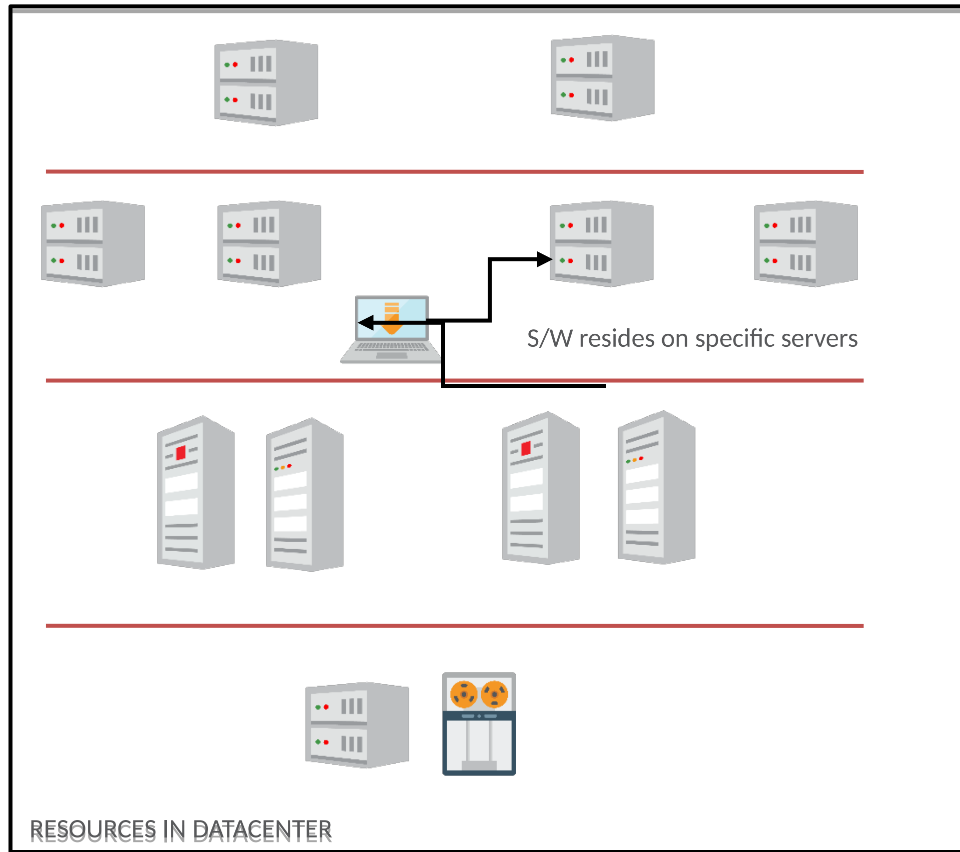


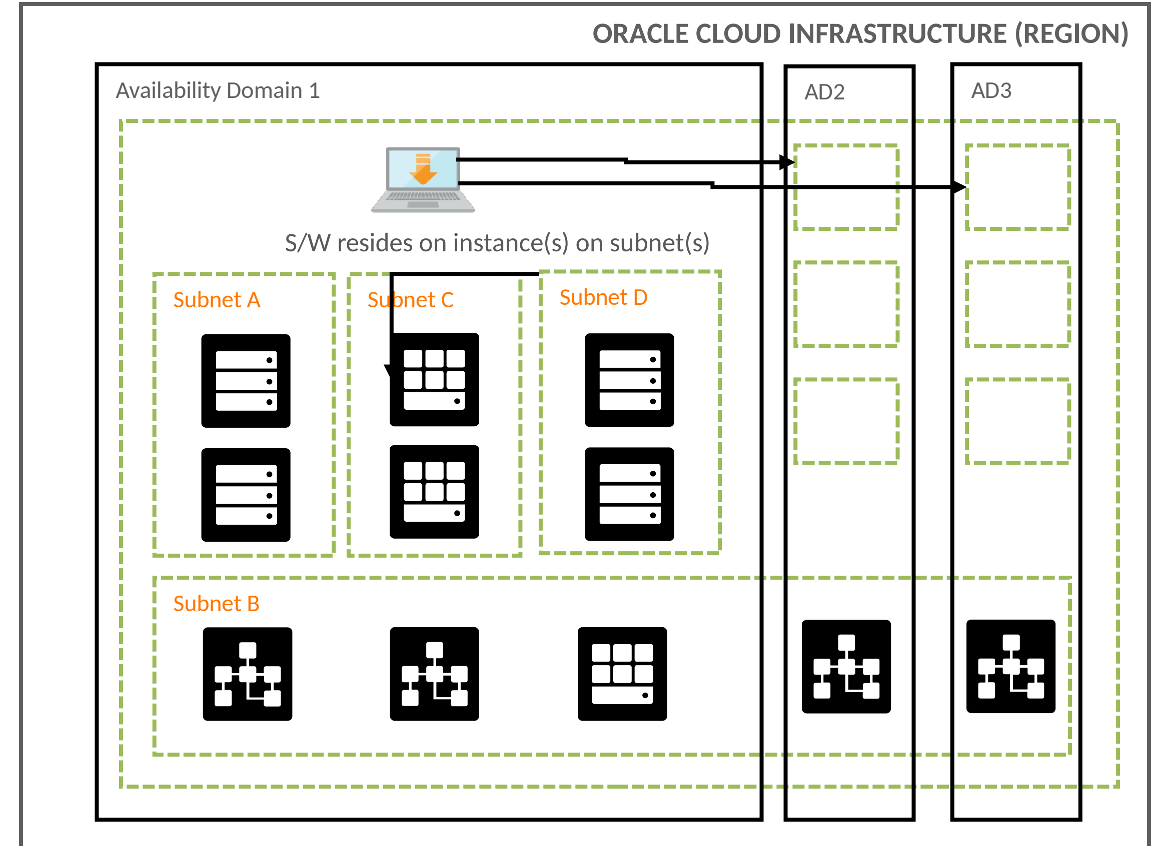
Learn more and try for free at: https://www.oracle.com/cloud/free

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Infrastructure Comparison (on-premises vs cloud)





- dedicated/fixed servers, storage, network ports (switches)
- fixed server IP addresses
- multiple levels of orchestration/co-ordination

- elastic pools of servers, storage, network resources
- compute instances (server or VM), DB instances
- S/W defined infrastructure, single control plane

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Our Cloud Native Principles For Micro Service Devs

1. System Immutability
2. Automate Everything
3. Disposability
4. Externalized Configuration
5. Logs as Event Streams
6. Delegated Governance
7. Service Lifecycle Independence

- everything is S/W controlled by CI/CD processes under the jurisdiction of governance
- CI/CD processes (dev/test/prod); cluster operations processes (scale, recover, update) are automated
- compute instances are ephemeral, services instances are expendable; starting/stopping instances is fast & reliable
- event logs for monitoring, alerting, debug, audits
- shared tools & CI/CD services but each dev team owns and operates their services
- services publish APIs allowing service upgrades and service replacements
- … and we are no exception too!

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Steps migrating to Cloud Native environment

- Iterative process for developing and refining services
  - on-boarding (set-up, credentials/entitlements allocation), learn CI/CD and governance rules
  - S/W containerization, S/W re-design/drop/introduction
  - learn Kubernetes and appreciate its declarative and asynchronous style
  - need to design and code procedures/features for:
    - Deployment (inc. rolling upgrade), Un-deployment, Scaling, Load Spreading, High Availability, Instance Failure, Temporary/Intermittent Failures, Logging/Monitoring integration, IAM integration, Storage integration, Network Integration (Load-Balancers/DNS/Outbound-proxy)
    - Operational APIs
  - harden for HA/high load/error recovery (operational behaviours)
    - replica mgt, restart co-ordination, persistence mgt, scheduling and resource mgt, performance tuning
  - API updates (CFS services (inc. CI/CD), OCI services, Kubernetes)
  - It takes time/effort to get a polished service that is highly reliable and manageable
  - … and we are no exception too!

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Considerations/Issues for Itinerant Services (1 of 3)

- **Infrastructure is abstracted**
  - less coupling between container and node
  - elastic pools of servers/network/storage
    - OCI quota, OCI physical limits
  - compute instance shape (server/VM)
    - standardized cores, RAM
    - standardized NICs
  - Kubernetes node using a fixed CFS image
  - S/W defined networking
  - overlay-ed networking (Flannel/Calico)
  - boot volume*, occasional local flash memory*
  - persistent storage services

- **Service Storage**
  - ideally stateless: no persistent local storage*
  - persistent storage services: K8s persistent volumes, OCI object storage, DB*
  - some storage services are AD based (OCI file storage) while others are regional (OCI object storage)
  - query peers/storage service for starting state
  - handle partial/corrupted state (e.g. interruption to write) e.g. redundant copy
  - network file system compatibility
  - performance
    - throughput typically less than DAS
    - latencies typically longer than DAS

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Considerations/Issues for Itinerant Services (2 of 3)

- **Starting/Stopping/Restarting Execution**
  - restartable instances
  - catch signals and cleanly shut down (also helps when evacuating pods from a node)
  - typically redundant instances, and/or able to quickly restart in any AD/FD (i.e. tolerable downtime)
  - tolerate network/endpoint congestion (it happens)
  - tolerate restarts of API endpoints (when they move or get replaced)
  - tolerate service migration/update/restart
    - longer turnaround times

- **Service Scheduling**
  - K8s pod/deployment/statefulset/daemonset/node
    - replica counts
    - resource requests/limits
    - pod disruption budgets
    - node labels, affinity/anti-affinity label matching
    - container sizing (100s MB->10s GB)
  - some trial and error
    - pods don’t spawn or get scheduled
    - request too high -> can’t schedule pods
    - limit too low -> container runtime OOM
    - container too big -> slow pull times
    - too many containers per node -> pull contention

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Considerations/Issues for Itinerant Services (3 of 3)

- **Failure Domains/Freedom to roam**
  - VM instances on a server
  - instances in a Fault Domain
  - instances in an AD
  - instances in a region
- **Naming and identification**
  - can't rely on server names or IP server addresses
  - load balancers (load balancer mgr)
  - K8s service names/pod names (statefulset)
- **Performance**
  - Throughput/latency wrt AD/region distribution
- **Service Configuration**
  - configuration represented as code
  - no post-deployment configuration tweaking
  - services may auto-configure based on their circumstances
  - services are re-deployed when configuration changes are needed, e.g. values are stored in a project configMap
  - goal is to avoid having customized clusters with no single point of control

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

CFS Kubernetes Cluster (Simplified)

Partner A Node Group

APP4
APP5

APP4
APP5

APP4
APP5

Partner B Node Group

APP1
APP2
APP3

APP1
APP2
APP3

APP1
APP2
APP3

Logging

Log/Search

Log/Search

Log/Search

CFS Nodes

Job Mgr
Infrastructure Mgr
Repaver

Infrastructure Mgr

Infrastructure Mgr

Master Nodes

kube-apiserver
kube-controller-manager
etcd-sever

kube-apiserver
kube-controller-manager
etcd-sever

kube-apiserver
kube-controller-manager
etcd-sever

OCI (REGION)

AD1
Block Storage
File Storage

AD2
Block Storage
File Storage

AD3
Block Storage
File Storage

Object Storage

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Node Groups, Service Isolation, and Service Distribution

- Projects can define pools of Kubernetes nodes to run exclusively* selected services (node groups)
- Nodes in a node group reside on dedicated subnets
- Mechanism for isolation and resource mgt, as node shapes can be matched to the services they will run
- Specification by custom resources, processed by an operator
- Seamlessly implemented as part of Kubernetes service

- Where regions have 3 availability domains (ADs), service replicas should be distributed across the ADs
- When regions have a 1 AD, service replicas should be distributed across the fault domains (FDs)
- Services are automatically restarted when the nodes they are scheduled on are NotReady (for too long).
- Kubernetes anti-affinity using `failure-domain.beta.kubernetes.io/zone`

- Migrating service instances between ADs is possible, but may not be necessary if there are sufficient service replicas remaining – some persistent storage is not regional
- Migrating service instances between regions is classified as disaster recovery. A cluster exists only in a region and so additional mechanisms are needed to effect the transfer. Also, other factors such as IAM scope, and resource availability come into play.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Operating Continuously Changing Kubernetes Cluster

- Node groups used to isolate teams within a cluster + cluster core from users. Same model works for dev and prod clusters.
- Replicated Kubernetes master nodes running replicated Kubernetes master services
- Replicated etcd (with back-up/restore) for Kubernetes db.
- Kubernetes node certificates valid for short duration, nodes replaced before certs expire
- Kubernetes signing/encryption keys continuously rotated (2 valid at any time)
- Kubernetes CA, valid for longer time, but again will be rotated as a side-effect of node repaving

- **Repaver Service**
  - replaces failed, aged nodes
  - handles node scaling
  - concurrent operation (1 thread per node group)
  - automated operation, speed adjustable
  - API for interactive use
- **Node Group Manager Service**
  - handles creation/scaling/deletion of node groups
  - relies on the repaver to handle individual nodes
- **Subnet Manager Service**
  - handles creation/deletion of subnets for the Node Group Mgr

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Operational Examples (Updates)

- **Updating the immutable image for nodes**
  - build an image (test on local dev cluster)
  - tag image for dev deployment
  - repaver checks for latest tagged dev image and deploy uses when replacing nodes
  - tag image for prod deployment
  - repaver works as before
  - incremental roll-out, slow but steady
  - contents of image deliberate sparse to avoid frequent updates

- **Updating a deployed service**
  - develop/test/review/merge changes to project
  - deployment pipeline executes with project-specific payload
  - typically new artifacts will be pushed repos (images) or storage services (metadata files, etc)
  - update Kubernetes resource definitions to use the new artifacts, and then Kubernetes works to reach the desired state
  - monitor the progress (or lack thereof) of Kubernetes and report pipeline status/result
  - relatively fast way to update a lot of clusters

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

Operational Examples (Replacement and Rotation)

- **Replacing a failed/aged nodes**
  - n + 1 model; nodes are not rebooted
  - nodes can fail owing to H/W issues and S/W (kernel) issues; (per) node problem detector looks for unhealthy nodes
  - nodes age and need to be retired
  - repaver service notices a unhealthy/aged node and creates a replacement, and waits for it to be operational (runs our node-specific pods)
  - repaver tell Kubernetes to stop using the old node, shuts it down, and removes it from the cluster
  - K8s creates replacement pods for the application services

- **Node Cert and Kubernetes Key Rotation**
  - later versions of Kubernetes allow certs signed by 1+ CA, secrets encrypted by 1+ key, and token signed by 1+ keys
  - bootstrap new clusters so that they start running and immediately start rotating out the node certs, the CAs, and the keys with unique values generated for that cluster
  - essentially 1 cert/key is used to sign/encrypt while a 3rd is being introduced to a new node via repaving and a 2nd is being withdrawn. After the appropriate number of node repaves, the cycle shifts, the new cert/key is used and the oldest one withdrawn.
  - implementation via the immutable image and a key-manager service

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

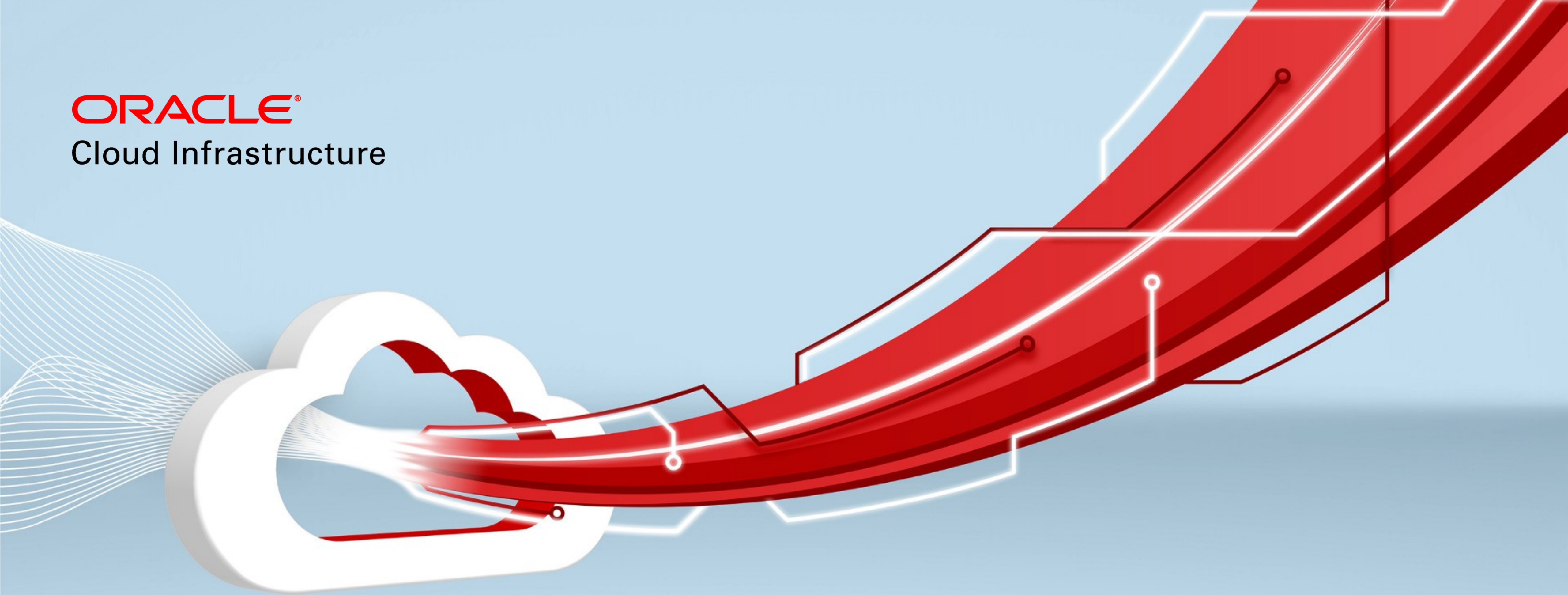
Summary and Questions

The devil is in the details but those are for another day…

- Deploying services on abstract, converged, infrastructure has a fundamental influence on how services are designed/structured/consumed/operated
- It's not hard to get a service to run (CI/CD learning curve aside), but getting it to run well (and reliably) takes some time, observation, Kubernetes research, and trial and error
- Key abstractions like immutable images, and continuous node repaving challenge devs to look at issues differently, usually successfully
- We operate 10s of clusters, with 1000s of containers, every day, with a high degree of automation and a relatively small team
- Brno team is **hiring** to help us to develop new features for our partners

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |



Oracle Cloud Infrastructure

Transform Your Business With Scalability

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |