# Modernizing Your Application Development Platform

## Using Oracle Cloud to Develop Healthcare Software Applications Better, Faster, and Cheaper

Ken Cottrell
Healthcare Enterprise Cloud Architect
Oracle Healthcare Team
Email: ken.cottrell@oracle.com
Mobile: (214) 546-5100

# Agenda: *New Application Development Trends and Tools to modernize Healthcare IT*

- Introduction to 12-Factor application patterns: The opportunity for Healthcare IT
- New Oracle products and services for Developers and Architects
- Summary / Next steps

# Agenda: *New Application Development Trends and Tools to modernize Healthcare IT*

- Introduction to 12-Factor application patterns: The opportunity for Healthcare IT
  - *Topics: Microservices , Containers, DevOps, Etc.*
  - *Types of Healthcare applications suited for these additional architectures*
- New Oracle products and services for Developers, Architects, Administrators
- Summary / Next steps

# 12-Factor Applications Architecture
# https://12factor.net/

I. Codebase One codebase tracked in revision control, many deploys

II. Dependencies Explicitly declare and isolate dependencies

III. Config Store config in the environment

IV. Backing services Treat backing services as attached resources

V. Build, release, run Strictly separate build and run stages

VI. Processes Execute the app as one or more stateless processes

VII. Port binding Export services via port binding

VIII. Concurrency Scale out via the process model

IX. Disposability Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity Keep development, staging, and production as similar as possible

XI. Logs Treat logs as event streams

XII. Admin processes Run admin/management tasks as one-

# 12-Factor Applications Architecture

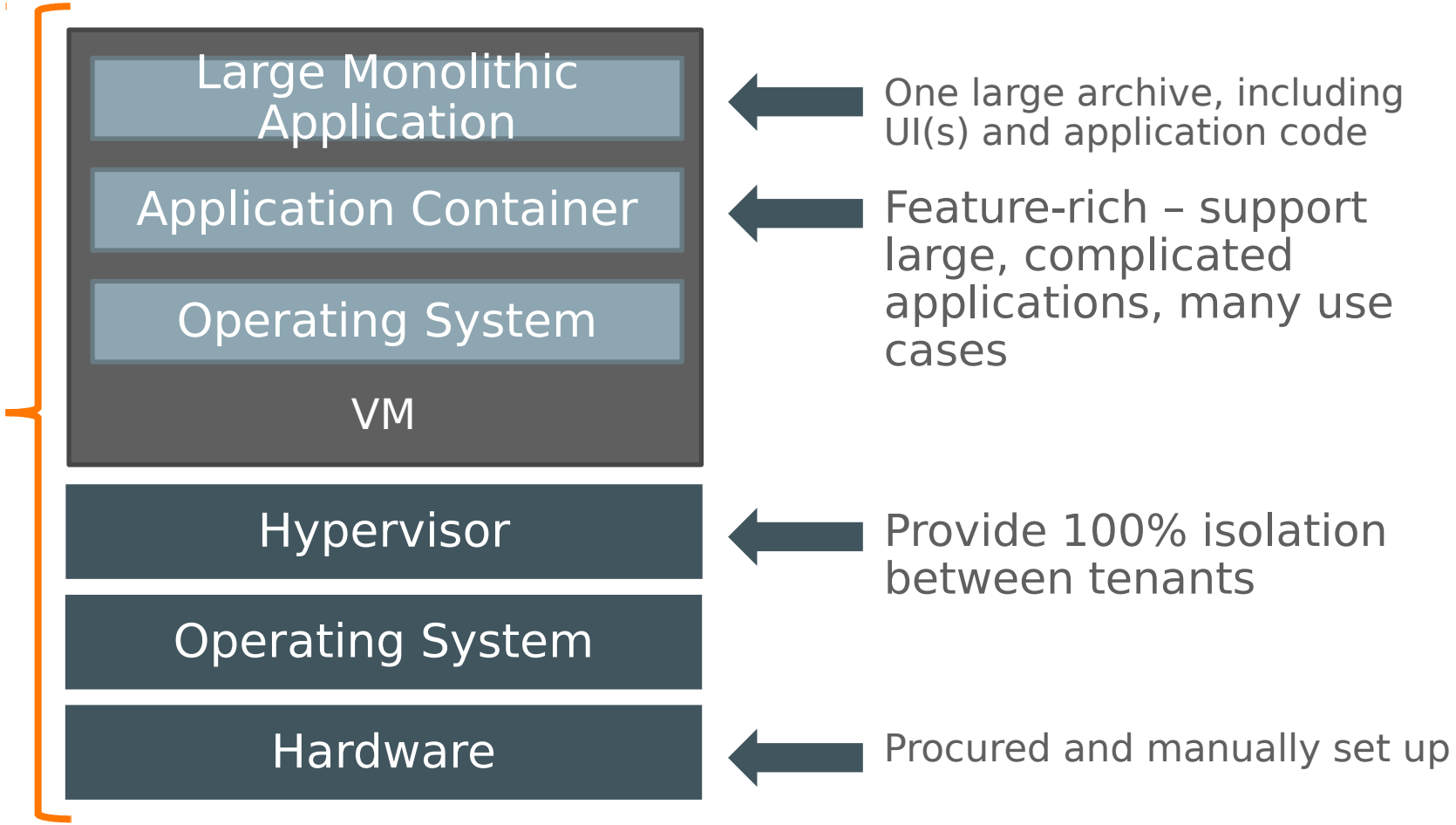
## Key Working Principles

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project; [i.e. Configuration verses coding, the “What” not the “How”]
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments; [i.e. Port bindings , protocols editable at deploy time]
- Are suitable for deployment on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility; [i.e. not just frequently, but continuously maybe several times daily]
- **Scales up** without significant changes to tooling, architecture, or development practices. [i.e. implies stateless, highly concurrent, easily replicated services]
- The twelve-factor methodology can be applied to apps written in **any programming language**, and which use any combination of backing services

# Characteristics of Existing Deployment Architecture

## The status quo has served us well but there are new alternatives

- Three tiers
- Many “named” servers that perform only one function and cannot go down
- Scale by cloning behind load balancer (X-axis scaling)
- One programming language
- Everything centralized – messaging, storage, database, etc



# Existing Deployment Architecture: Not always the best approach

## Too Complex

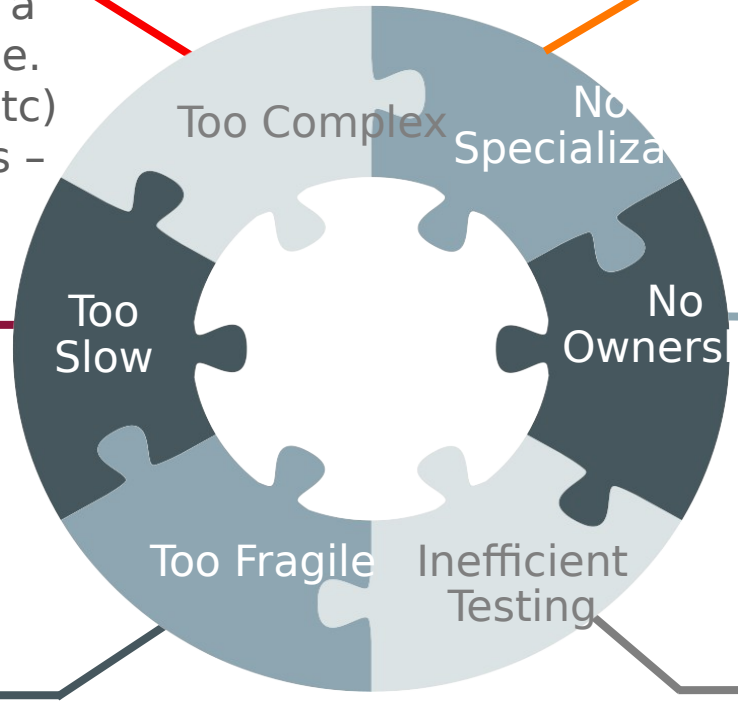
Apps get big and complicated for a developer to understand over time. Shared layers (ORM, messaging, etc) have to handle 100% of use cases – no point solutions

## Too Slow

Teams split up by function – UI, application, middleware, database, etc. Takes longer to get anything done due to cross-ticketing

## Too Fragile

A bug on one module can effect other modules in the application due to tight coupling




## Resource Bottlenecks

Different parts of applications have different needs from shared resources – more CPU, more memory, faster network, etc

## Sense of Ownership

Code may fall victim to "tragedy of the commons" – when there's little ownership, you see neglect

## Inefficient Testing

Each time you touch the application, you have to re-test the whole thing. Hard to support continuous delivery

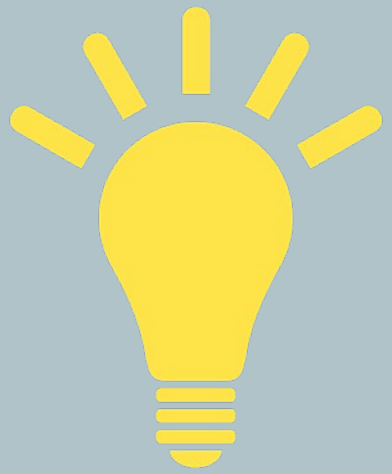
# Different Types of Software Requires Different Practices

A payroll system should be treated very different from a customer-facing .com

**Core Software - Keep the Lights On**

**Differentiation Software - Run Current Business**

**Innovation Software - Find the Next Business**



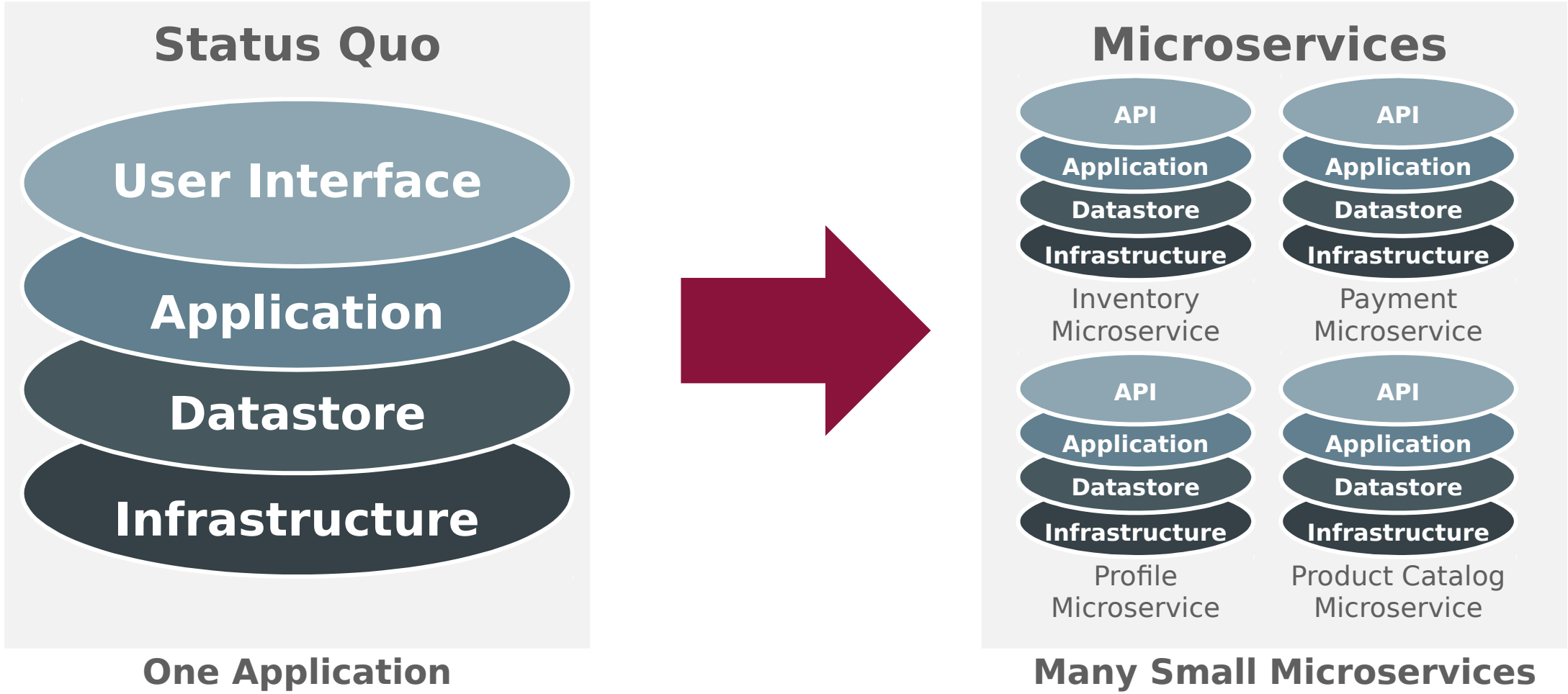
| | | |
|---|---|---|
| *Business-centric* | ⟷ | *IT-centric* |
| *Top Line Growth* | ⟷ | *Bottom Line Savings* |
| *Release Hourly* | ⟷ | *Release Quarterly* |
| *Fail Early* | ⟷ | *Fail Late* |
| *Bespoke Software* | ⟷ | *Packaged Software* |
| *Agile* | ⟷ | *Waterfall* |
| *Product-based* | ⟷ | *Project-based* |

# What Are Microservices?

**Minimal function services that are deployed separately but can interact together to achieve a broader use-case**

| Status Quo | Microservices |
| --- | --- |
| Single, Monolithic App | Many, Smaller Minimal Function Microservices |
| Must Deploy Entire App | Can Deploy Each Microservice Independently |
| One Database for Entire App | Each Microservice Has Its Own Datastore |
| In-process Calls Locally, SOAP Externally | REST Calls Over HTTP, Messaging, or Binary |
| Organized Around Technology Layers | Organized Around Business Capabilities |
| Developers Don’t Do Ops | Developers + Ops Support Production in |
| State In Each Runtime Instance | |

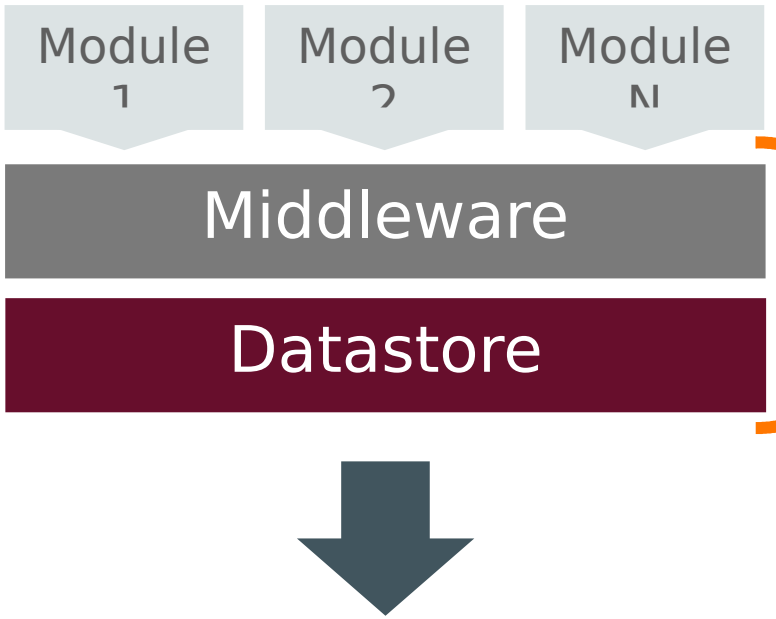
# Microservices Apps Are Developed/Deployed Independently



# “Micro” in Microservices != Runtime Weight

**Microservices *tend* to use smaller runtimes but you can use what you have today**

## Monoliths




*Must support the requirements of ALL modules*

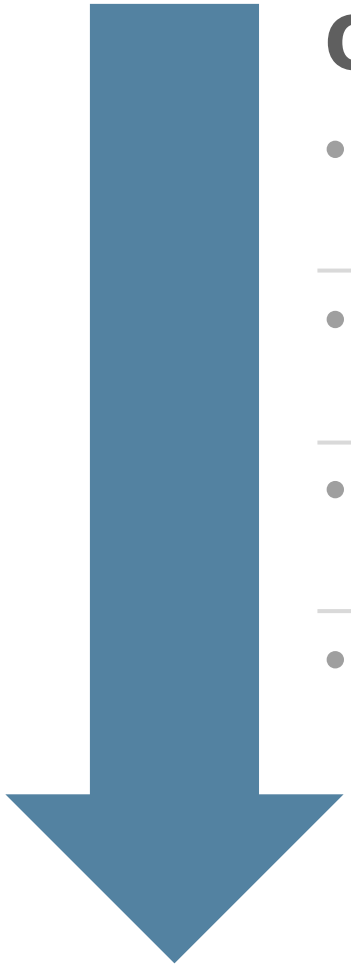
Fully Featured Runtimes That Support All Use Cases

## Microservi ces


*Must support the requirements of one module*

Light Runtimes That Do One Thing

# Microservices Forces Choreography Over Orchestration

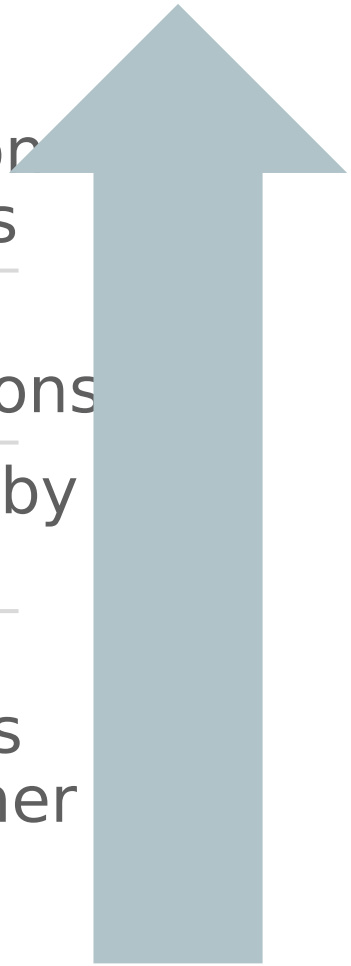


## Orchestration

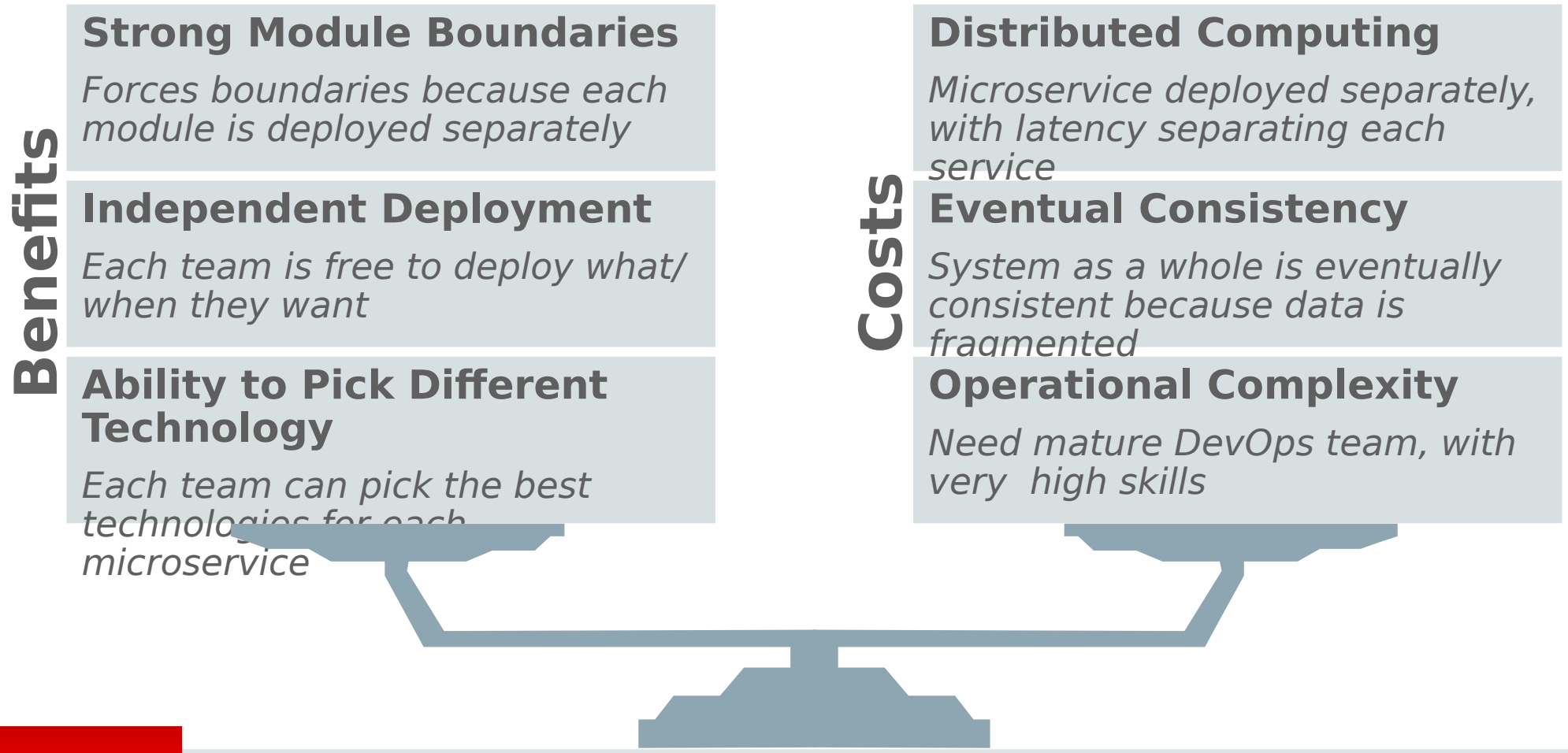
- Top-down coordination of discrete actions
- Used in centralized, monolithic applications
- Brittle – centralized by nature
- Each “action” registers with centralized system – single point of failure that is not very flexible

## Choreography

- Bottom-up coordination of discrete actions
- Used in distributed, microservice applications
- Resilient – distributed by nature
- Each microservice asynchronously throws up a message that other microservices can consume



# Benefits of Microservices Come With Costs



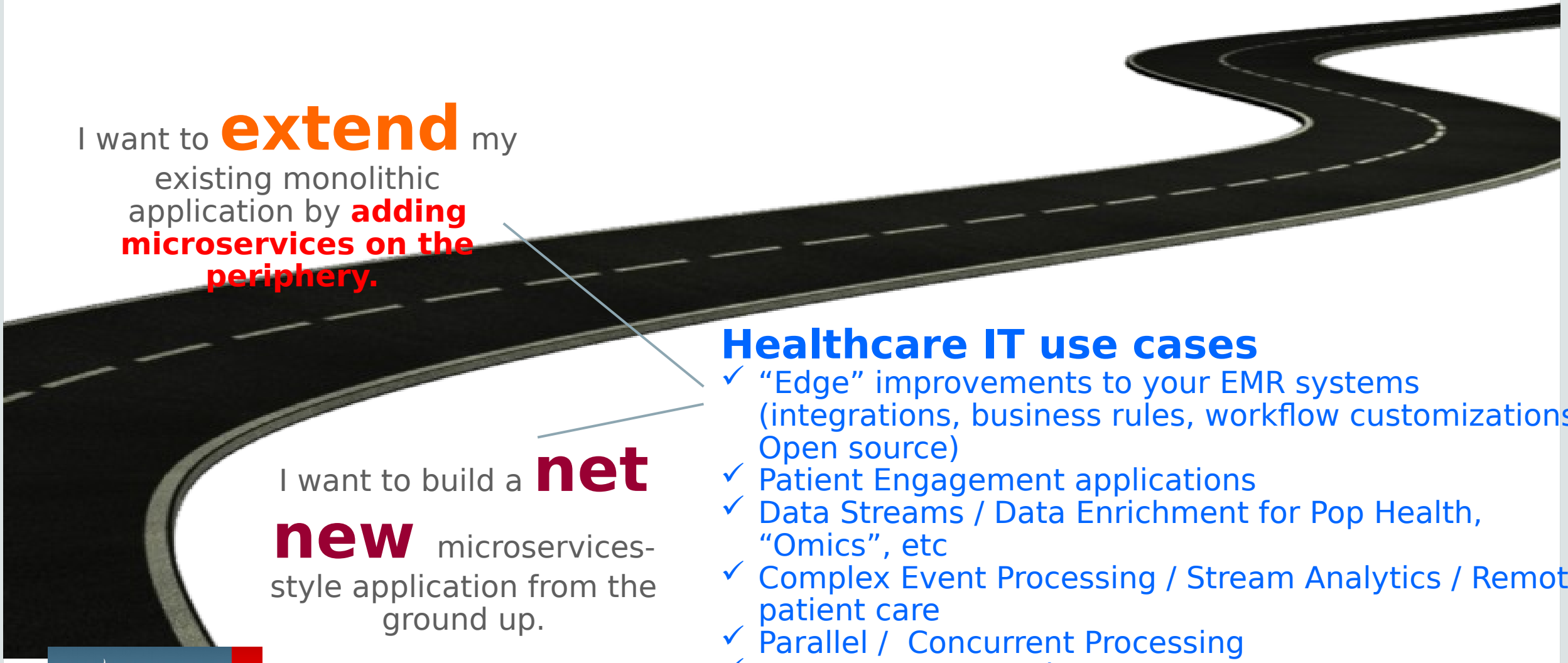
# Common Microservice Adoption Use Cases

I want to **extend** my existing monolithic application by adding microservices on the periphery.

I want to **decompose** an existing modular application into a microservices-style application

I want to build a **net new** microservices-style application from the ground up.

# Fit Assessment: Microservice Adoption Use Cases



I want to **extend** my existing monolithic application by **adding microservices on the periphery.**

I want to build a **net new** microservices-style application from the ground up.

## Healthcare IT use cases

- ✓ “Edge” improvements to your EMR systems (integrations, business rules, workflow customizations, Open source)
- ✓ Patient Engagement applications
- ✓ Data Streams / Data Enrichment for Pop Health, “Omics”, etc
- ✓ Complex Event Processing / Stream Analytics / Remote patient care
- ✓ Parallel / Concurrent Processing
- ✓ Log Data Map / Reduce

# Introduction to Docker

**An Open Platform to Build, Ship, and Run Distributed Applications**

- Extremely fast adoption by developers
- Improved, user-friendly Linux Container technology
- Runs on Linux, Windows, and Mac OS X
- Considered as a “lightweighter” virtualization technology
- Easy, human-readable mechanism to build containers images based on recipes, (aka Dockerfiles)

# Docker is lighter weight than Virtual Machines

## Key differences

| | VMs | Containers |
|---|---|---|
| Image Creation | Can be automated with 3rd-party tools (Chef/Puppet/etc), but takes a lot of time. Snapshots are bigger compared to Container images. | Same options as VMs + can also declaratively construct one using native, 1st class Dockerfile format. |
| Updates | Must use configuration management tool to apply updates. Or must update every single gold image and then re-deploy | Can apply diffs to container images, or most often, the image is quickly rebuilt and container re-deployed |
| Performance | Heavy weight – must often go through abstraction layers to access physical hardware resources | Lightweight – a container is just an operating system process. 100% native access to all physical hardware resources |
| Utilization | Harder to over-subscribe physical hardware resources like CPU and memory | consume physical hardware resources. Can easily move containers off of hosts when the hosts become too utilized |

# Docker Containers Are The New Virtualization Trend

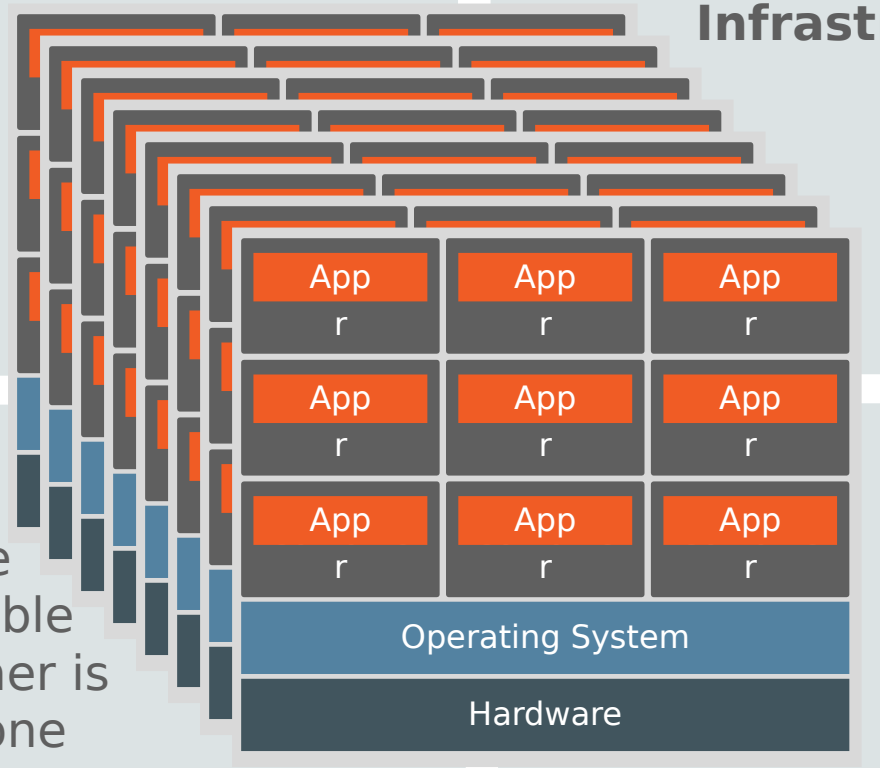
## Four main use cases

### Application Packaging

Neatly package applications and supporting environment in immutable, portable containers

### Infrastructure Consolidation

Get infrastructure utilization up to 100% (vs 5-10% with VMs) due to over-subscription of resources and near bare metal performance.




### Continuous Integration

All changes to an app are contained in one immutable container image. Container is tested and deployed as one atomic unit

### DIY PaaS

Build a simple PaaS by wiring up containers to a load balancer. New code, patches, etc pushed as new immutable containers.

# OCI – Open Container Initiative

- Governance structure for the express purpose of creating industry standards around container formats and runtime.
- Customers can commit to container technologies w/o worrying on being locked-in.
- Oracle joined in 2015.
  - blogs.oracle.com/solaris/entry/oracle_joins_the_open_contai

# Docker benefits

## Developers: Productivity

- ✓ Faster creation of ready-to-run packaged applications
- ✓ Cheaper deployment, with instant replay and reset
- ✓ Automate testing, integration, packaging
- ✓ Cleaner, safer, portable runtime environment
  - No missing/conflicting dependencies or packages
  - Each app runs in an isolated container
  - Reduce/eliminate platform compatibility issues

## Administrators: Productivity

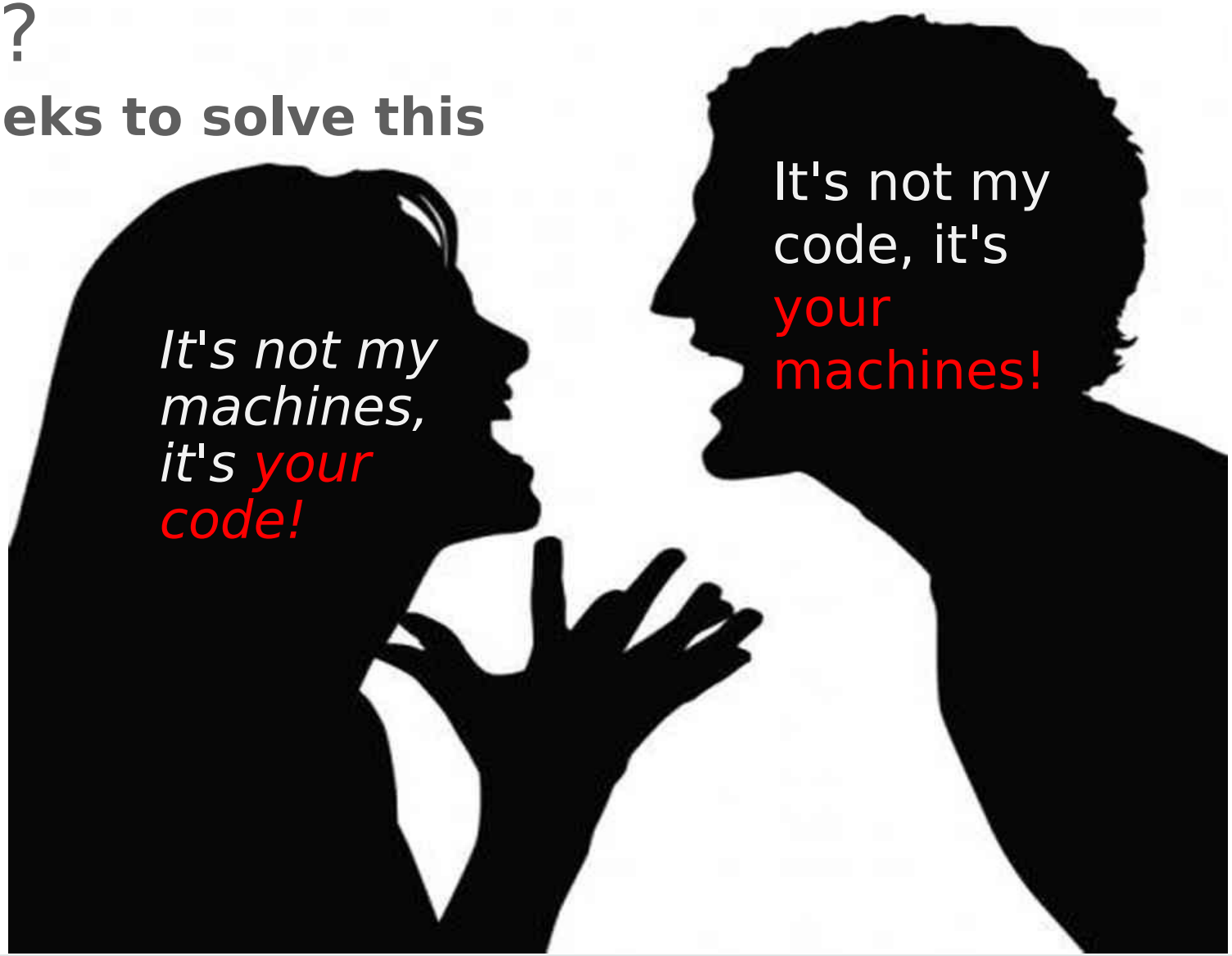
- ✓ Lightweight containers address performance, costs, deployment and portability issues
- ✓ Configure once, run many times
  - Environments, Processes, Applications
- ✓ Makes app lifecycle efficient, consistent and repeatable
  - Eliminate environment inconsistencies between development, test, production
  - Supports segregation of duties

## Management: Agility / Cost

- ✓ Onboarding new developers is much faster with predefined development environments
- ✓ Speeds delivery of packaged solutions
  - Container images are much smaller than traditional VMs images.
  - Full Guest OS is not needed.
- ✓ Better asset utilization (both for VM & “Bare Metal” Hardware)
  - Reduce virtualization costs
  - Increased Density

# Familiar?

## **DevOps** seeks to solve this

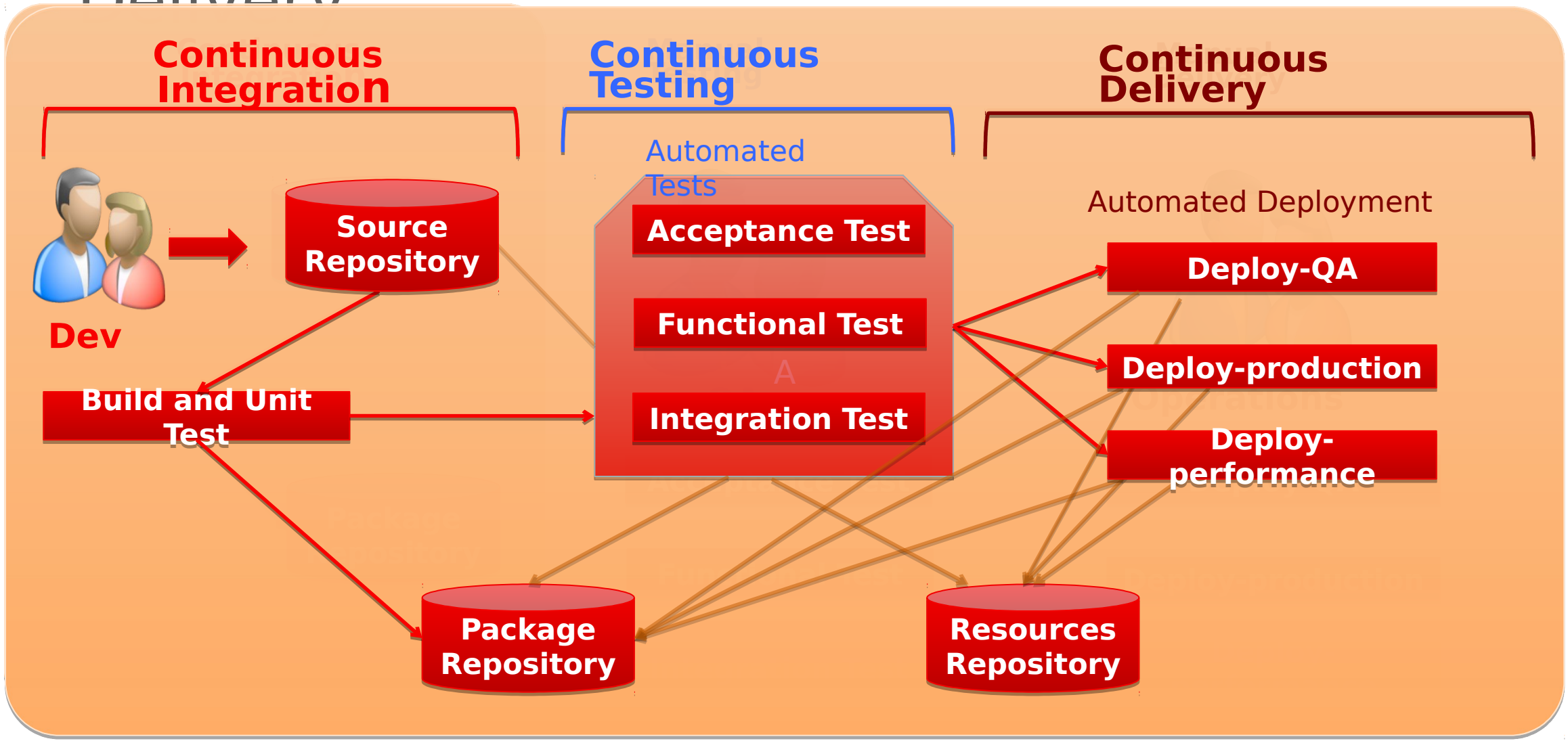


# Core DevOps Principles

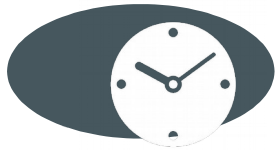
## Cultural movement enabled by technology


**New goal:**
Add new features FAST *and* keep the system stable, fast and available

# DevOps Continuous Integration, Testing, and Delivery



# Business Value Is Driving DevOps in the Cloud



## FASTER TIME-TO-MARKET

- Quickly align with business requirements by increasing frequency of releases
- Increase accuracy of releases - avoid downtime

## COST

- Automate what was previously done manually. Reduces OPEX
- Prevent humans from making costly errors
- Reduce downtime, which saves money

## FOCUS ON BUSINESS VALUE

- Allow specialist workers to focus on higher value activities

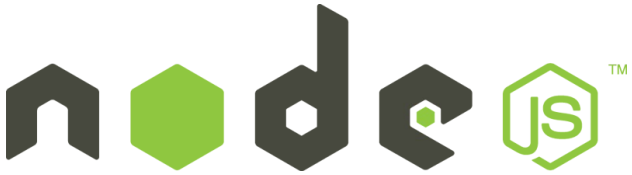
# Trend: GitHub

- Web based code repository
- Provides distributed source control and source code management
- Also provides bug tracking, feature requests, task management and wiki’s for each of the projects
- Repositories can be private or public
- Over 21 million code repositories

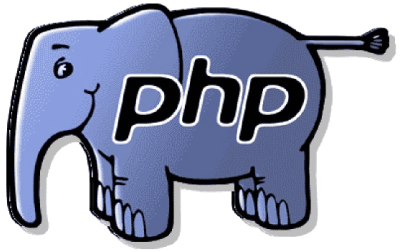
# Trend: Pick best language for the job





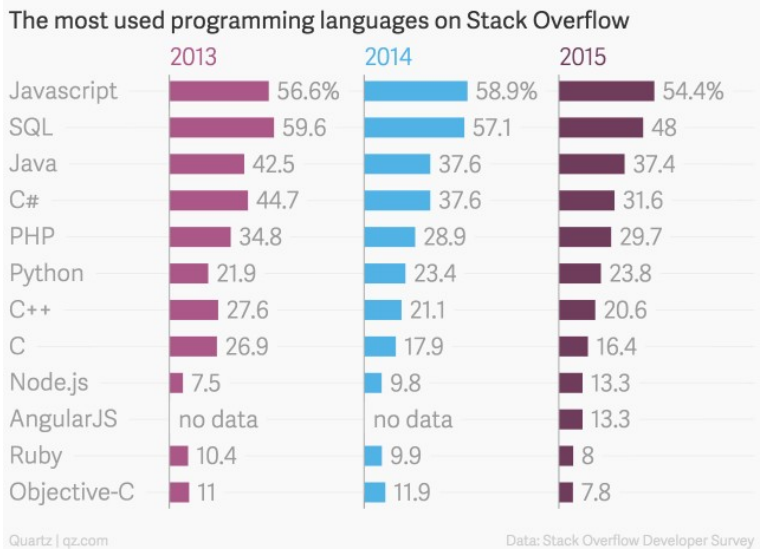







# Trend: Node.js




- Server side development using JavaScript
- Cross platform support
  - Windows, Linux, AIX, OSX
- Uses an Event driven architecture with Non Blocking I/O
  - Call backs signal the end of an operation
  - Threaded code is not needed
- A package manager (npm) makes for easy distribution of Code
- node-oracledb driver available via npm

# Trend: Pick the best Framework for the job

## ORACLE CLOUD: USE ANY OPEN SOURCE OR COMMERCIAL JAVA OR NODE FRAMEWORKS


Jersey

eclipse

play

jetty://

HIBERNATE

Jython

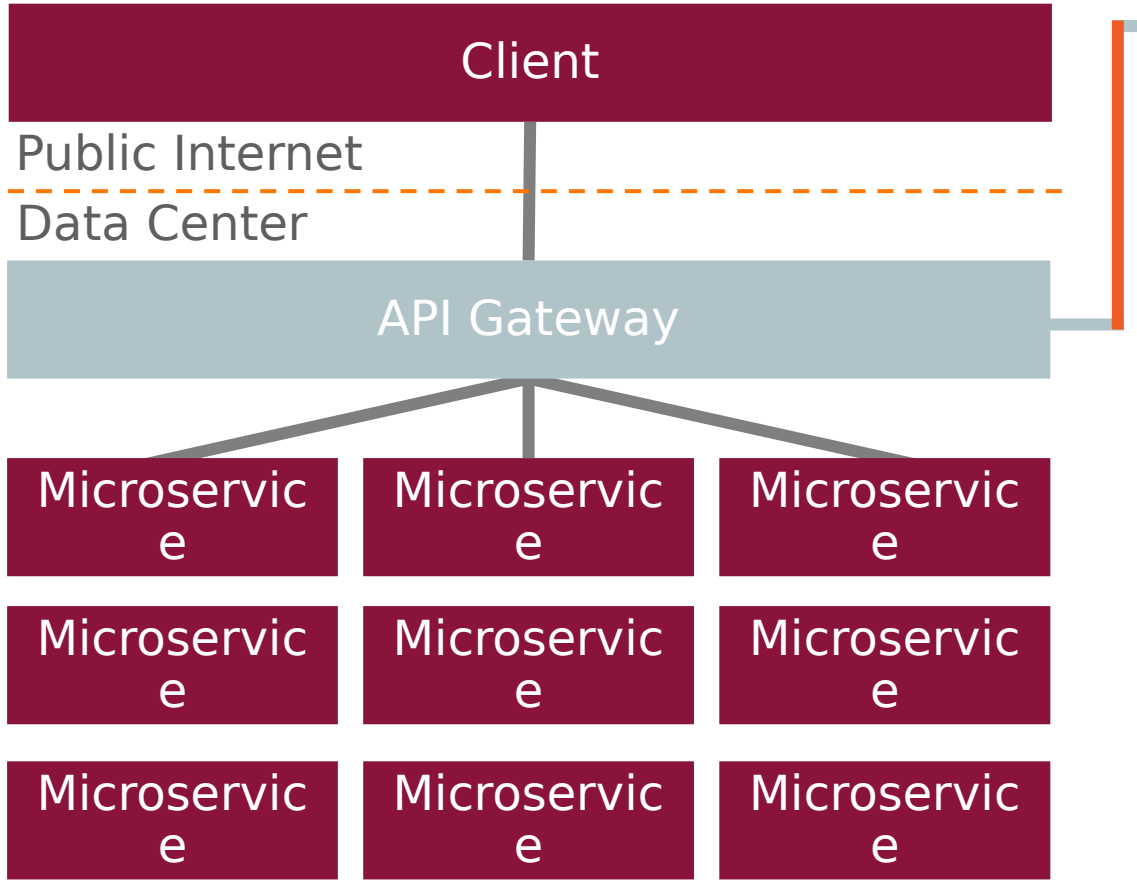
eclipselink

vaadin

JRuby

# Trend: Growth of API Gateways

## API gateways provide a "backend for each frontend"




- Builds a XML or JSON response for each type of client – web, mobile, etc
- Asynchronously calls each of the N microservices required to build a response
- Handles security and hides back-end
- Load balances
- Applies *limited* business logic
- Meters APIs
- Logs centrally

# Trend: Web Development

- Trend to Single Page Applications (SPA)
  - Made easier by JavaScript frameworks like Angular.js
- More capable browsers that do more of the heavier lifting that was previously done in the mid tier
- REST services provide a simpler way of transferring self contained data packages between systems
- Collaborative development via Web based source control platforms like Github

# Agenda: *New Application Development Trends and Tools to modernize Healthcare IT*

- Introduction to 12-Factor application patterns: The opportunity for Healthcare IT
- New Oracle products and services for Developers, Architects, Administrators
  - *Oracle support for 12-Factor Architecture patterns*
  - *New Polyglot language tools for Software Developers*
  - *Oracle Infrastructure-As-A-Service (IaaS) offerings for Compute, Storage, Network*
- Summary / Next steps

# Oracle DevOps

**DEVELOPER**
CLOUD
SERVICE

- IDE support
- Team Collaboration
- Repository
- CD / CI Test &Deploy

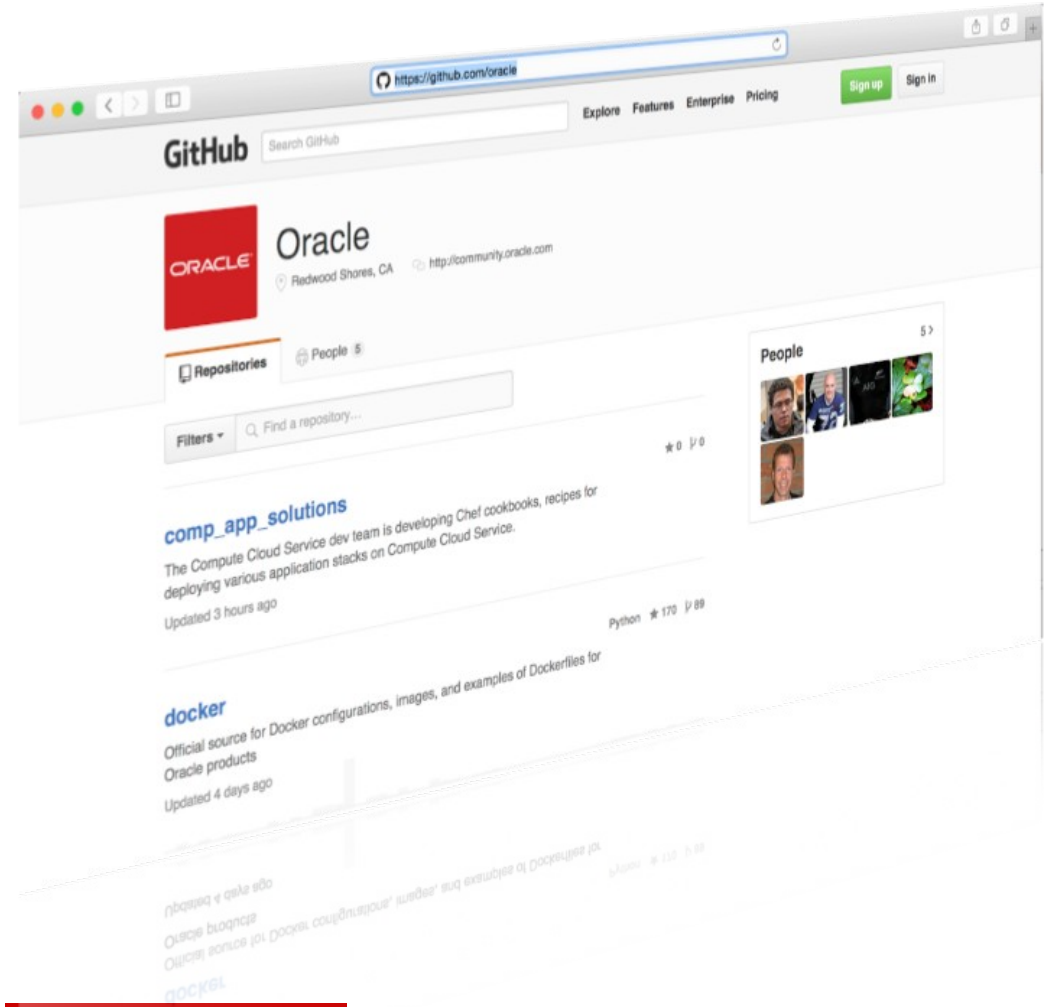
Monitoring
Analysis
Feedback loop

**MANAGEMENT**
CLOUD

**PaaS**
CLOUD
SERVICES

- Java Cloud Service
- Database Cloud Service
- Container Cloud Service
- Infrastructure management

# Oracle and GitHub




- Projects being added daily
- Repositories for Oracle, MySQL, Coherence and WebLogic
- Examples and tools in Java, PL/SQL, SQL, Python, Node.js, Docker etc.
- Oracle Sample schemas (SH, OE, HR etc.) in the future will be maintained on GitHub

# JAX-RS: The Java API for RESTful Web Services

**Oracle is a spec lead**

## Server-side Code

```
@Path("/atm/{cardId}”)
public class AtmService {
  @GET @Path("/balance”)
  @Produces("text/plain”)
  public String balance(
     @PathParam("cardId") String card,
        @QueryParam("pin") String pin) {
```

**Simply annotate Java code to expose as REST**

## Client-side Code

```
Client client = ClientFactory.newClient();
String balance =
  client.target("http://xxxx/atm/{cardId}/
                                    balance")
  .pathParam("cardId", "1234567890123456")
  .queryParam("pin", "1111")
  .request("text/plain")
  .get(String.class);
```

**Use REST without having to parse text**

Originally defined in JSR 311 - 1.0
Part of Java EE 6 Spec

|

Updated as JSR 339 in 2013 - 2.0
Part of Java EE 7 Spec

# Common JAX-RS Implementations

**Oracle is the spec lead**

JERSEY

Official Reference Implementation

*Sponsored by*

ORACLE

ORACLE®

WEBLOGIC

GlassFish

# Grizzly: High Performance I/O

**Great for inter-process communication**

- Oracle sponsored open source
- Allows developers to take advantage of the Java NIO to provide very fast inter-process communication
- Brings non-blocking sockets to the protocol processing layer
  - Support for non-blocking HTTP processing
- WebSocket Support
- APIs make non-blocking interactions simple

# Examples of Oracle Offerings for Microservices

**Java Cloud Service (WebLogic)**

Run any Java EE-based application on a WebLogic-based PaaS

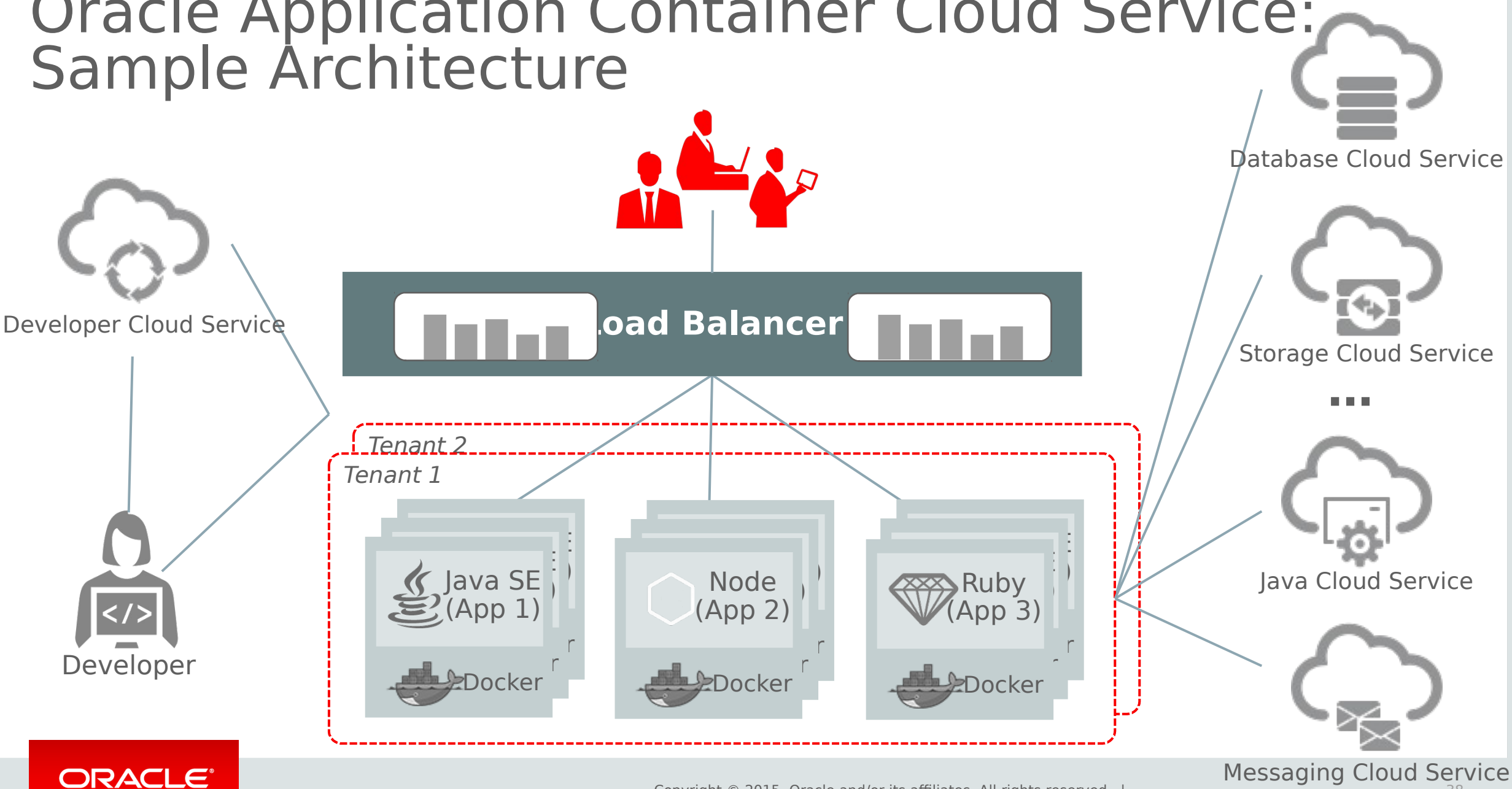
**Application Container Cloud Service (Java SE and Node.JS)**

Currently for Java and Node.js, but supports *any* programming language that runs on a JVM

**Compute Cloud Service**

Bring your own JVM

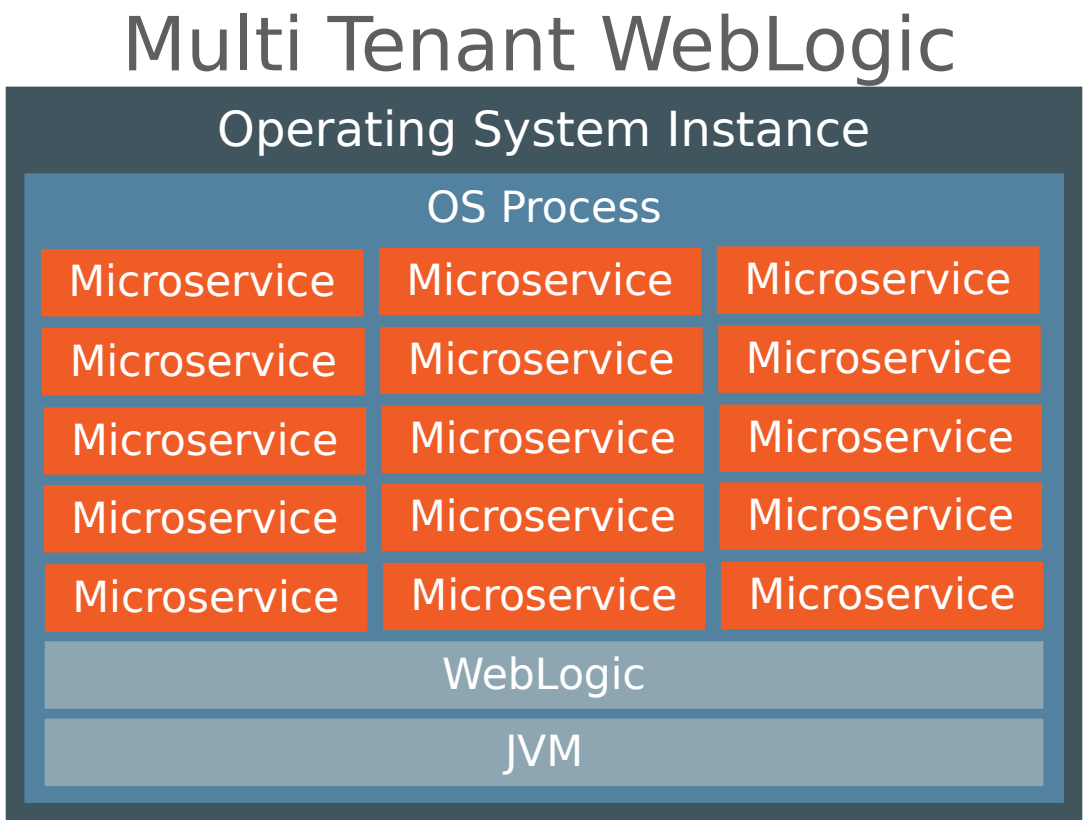
# Oracle Application Container Cloud Service: Sample Architecture



# WebLogic Multi Tenant Support for Microservices

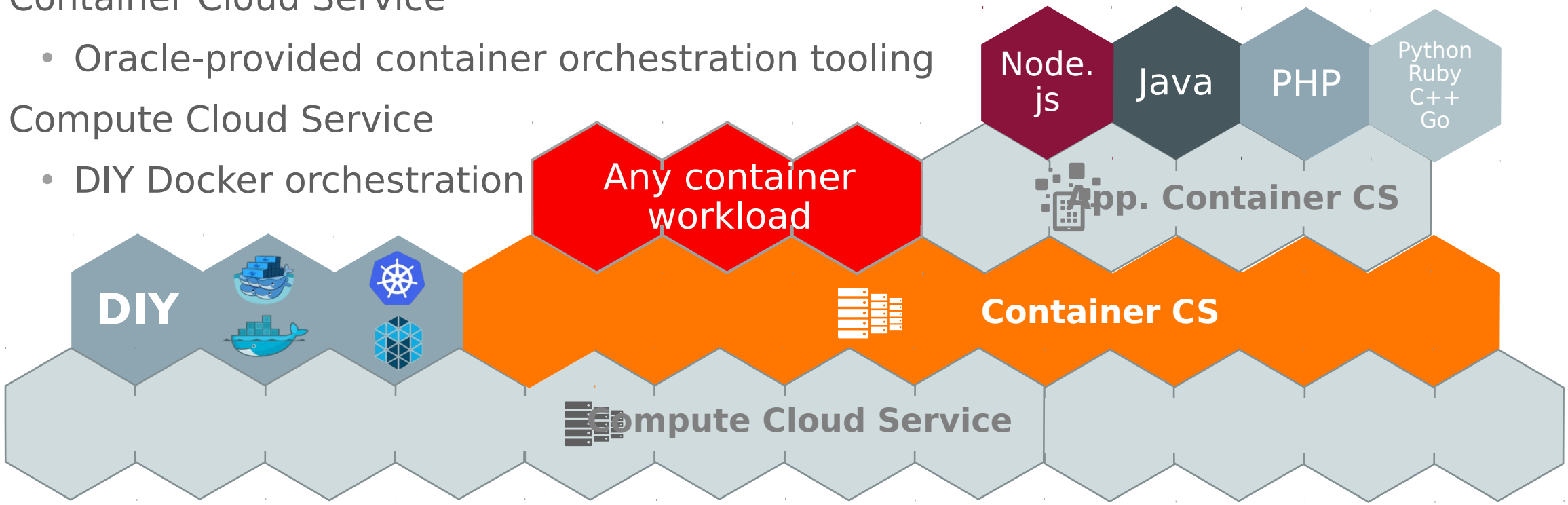
**Similar to Oracle Database pluggable/container databases**

- Each microservice instance can have its own light-weight WebLogic container-like partition
- Easily move partitions between WebLogic hosts
- Each partition is exceptionally light
- Each WebLogic host can support hundreds of partitions



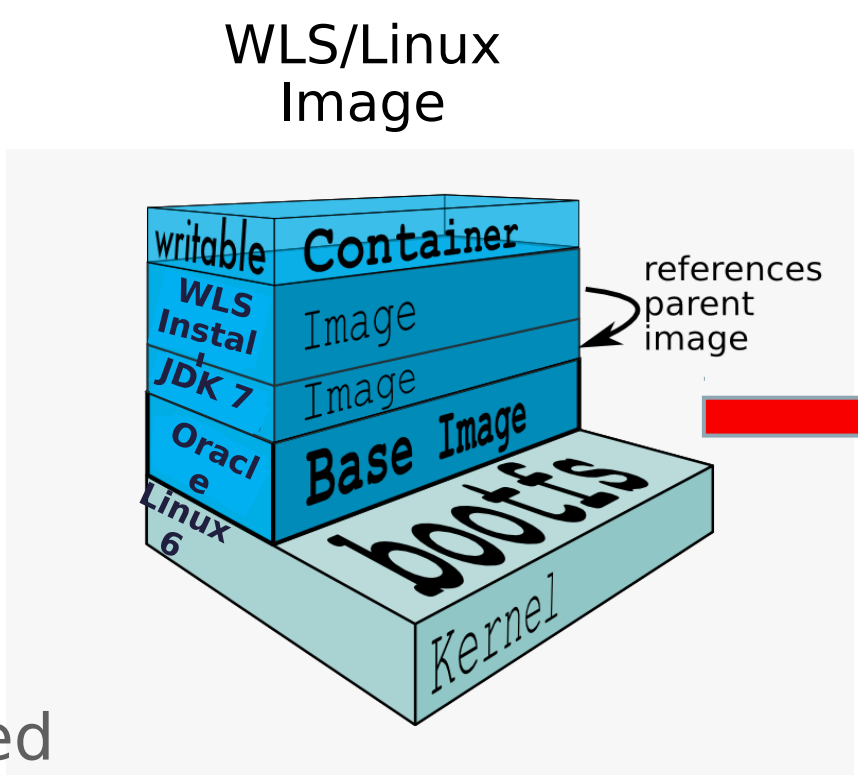
# Docker on Oracle Cloud – The Big Picture

- Application Container Cloud Service
  - Run Cloud Native applications with ease
- Container Cloud Service
  - Oracle-provided container orchestration tooling
- Compute Cloud Service
  - DIY Docker orchestration



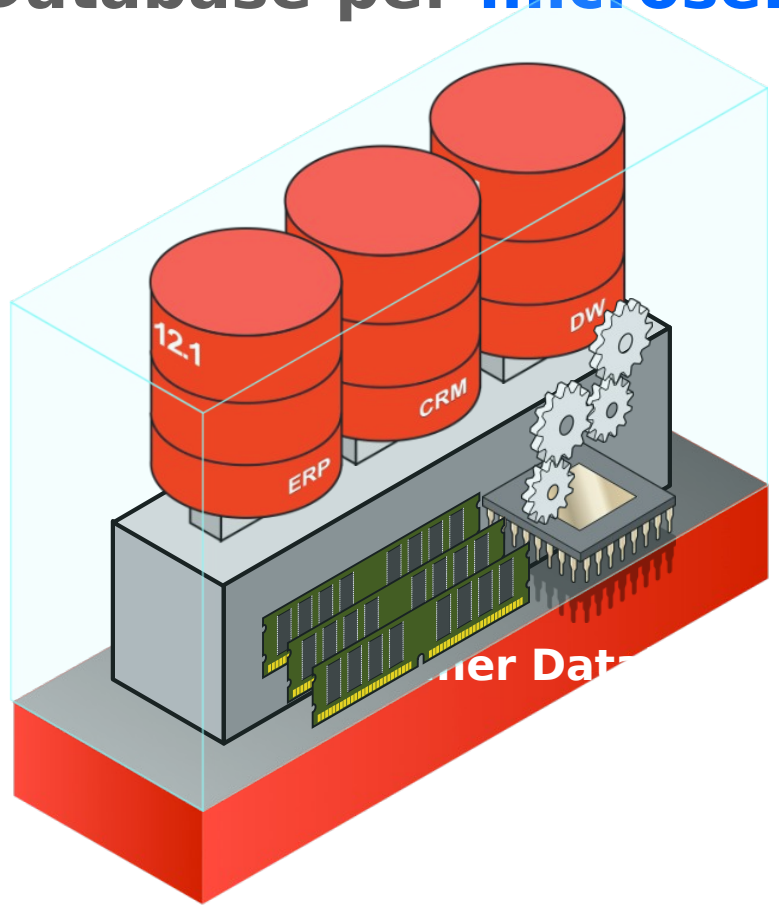
# WebLogic Docker Certification

- Certified since March 2015
  - WLS 12.1.3 with Docker
  - Oracle Linux 6 and 7, Red Hat Linux 7
  - JDK 7 and 8
  - DockerFiles on GitHub
  - Development and production support
- First phase of ongoing effort
  - Expand configurations supported
  - Leverage Docker enhancements



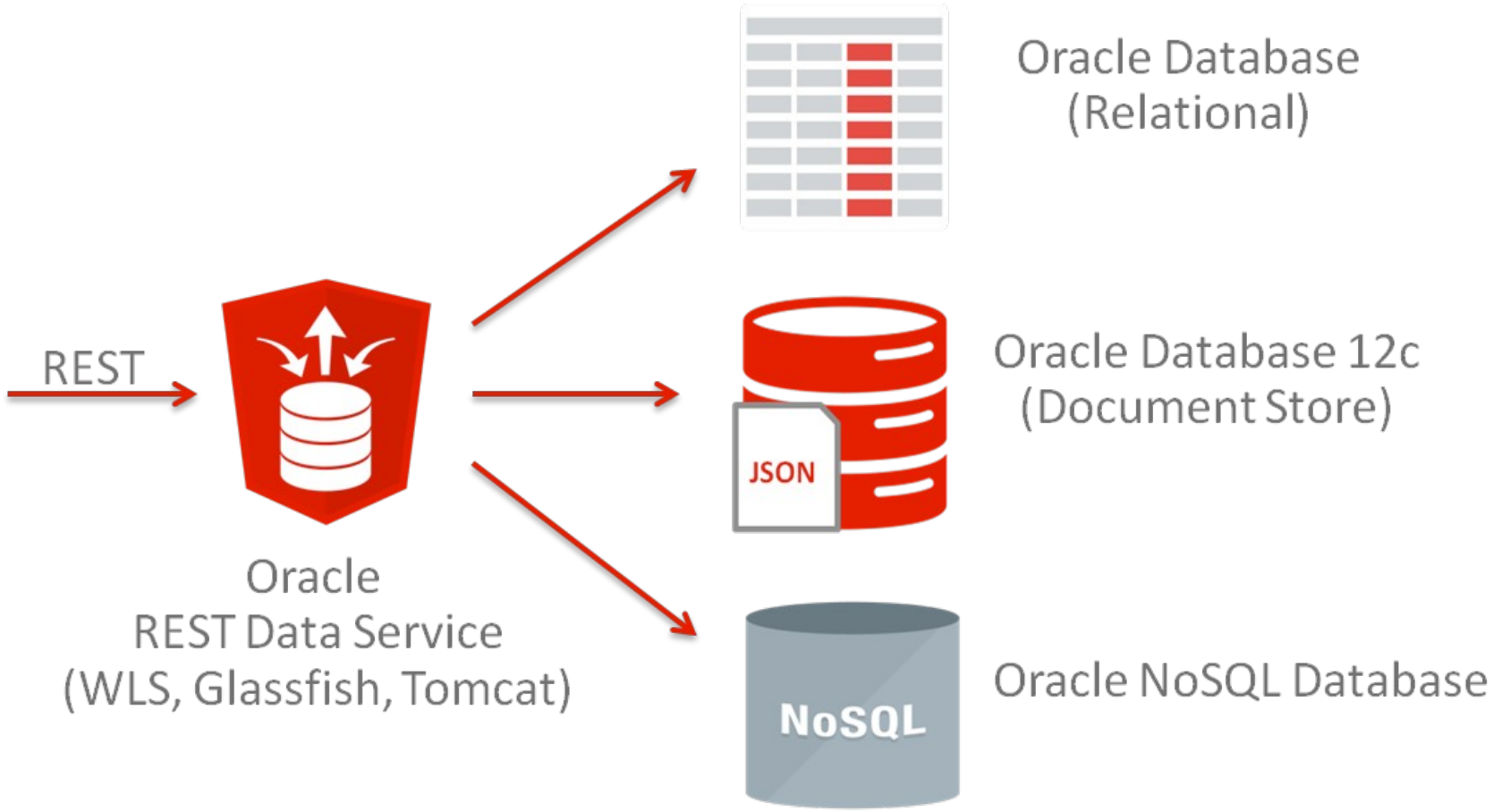
# Pluggable Databases

## One Container Database per application, one Container Database per microservice




- Container Database
  - Multi-tenant database that includes zero, one or many pluggable databases
  - Upgrades, etc are performed against container
- Pluggable Database
  - A full database to the client except that behind the scenes it doesn't have its own controlfiles, redo logs, undo, etc
  - Just a collection of datafiles and tempfiles to handle its own objects, including its own data dictionary
  - Can easily move Pluggable Databases from one container to another

# Oracle Database 12*c* for the Developer
## Oracle REST Data Service



# JSON Support in Oracle Database

## Supports Application Developers and Existing BI tools

**Application developers:**
Access JSON documents

```
POST /my_database/my_schema/customers HTTP/1.0
Content-Type: application/json
Body:
{
 "firstName": "John",
 “lastName”: "Smith",
 "age": 25,
 "address": {
     "streetAddress": "21 2nd Street",
     "city": "New York",
     "state": "NY",
     "postalCode": "10021“,
     "isBusiness" : false  },
  "phoneNumbers": [
     {"type": "home",
      "number": "212 555-1234“ },
     {"type": "fax",
      "number": "646 555-4567“ }  ]
}
```

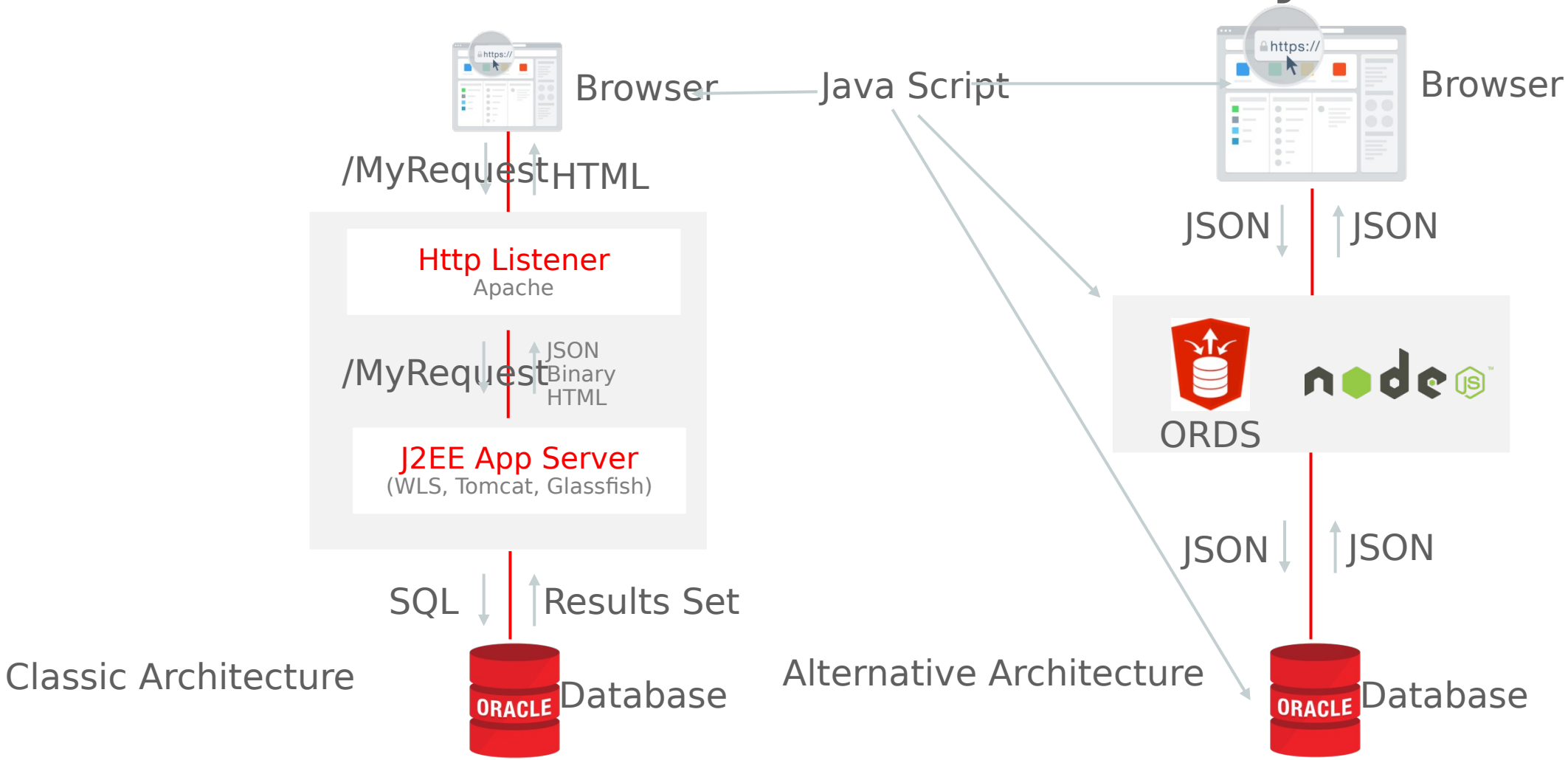
Oracle Database 12c


**Analytical tools and business users:**
Query JSON using SQL

```
select
  c.json_document.firstName,
  c.json_document.lastName,
  c.json_document.address.city
from customers c;


firstName    lastName     address.city
-----------  -----------  --------------
“John”       “Smith”      “New York”
```

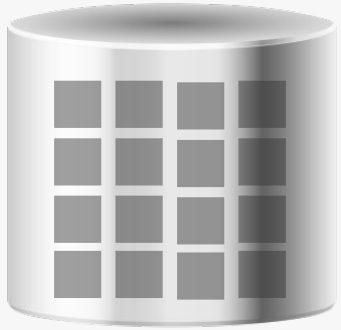
# Oracle DBMS Architecture Flexibility



# Oracle Cloud Infrastructure

## Complete Infrastructure for Enterprise Workloads

**Storage**
Elastic Storage



**Compute**
Elastic Compute

**Network**
Software-Defined

ORACLE

# Oracle Cloud Compute Services

## Bare Metal

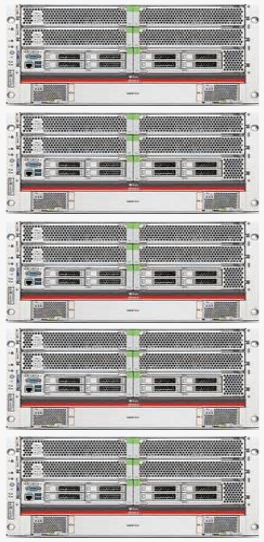
- Docker Containers
- Multiple Hypervisors
- Multiple OS

## Elastic Compute

solaris

Compute

## Dedicated Compute



## Docker Service

MESOSkubernetes

Docker Registry

Bare Metal Elastic Compute

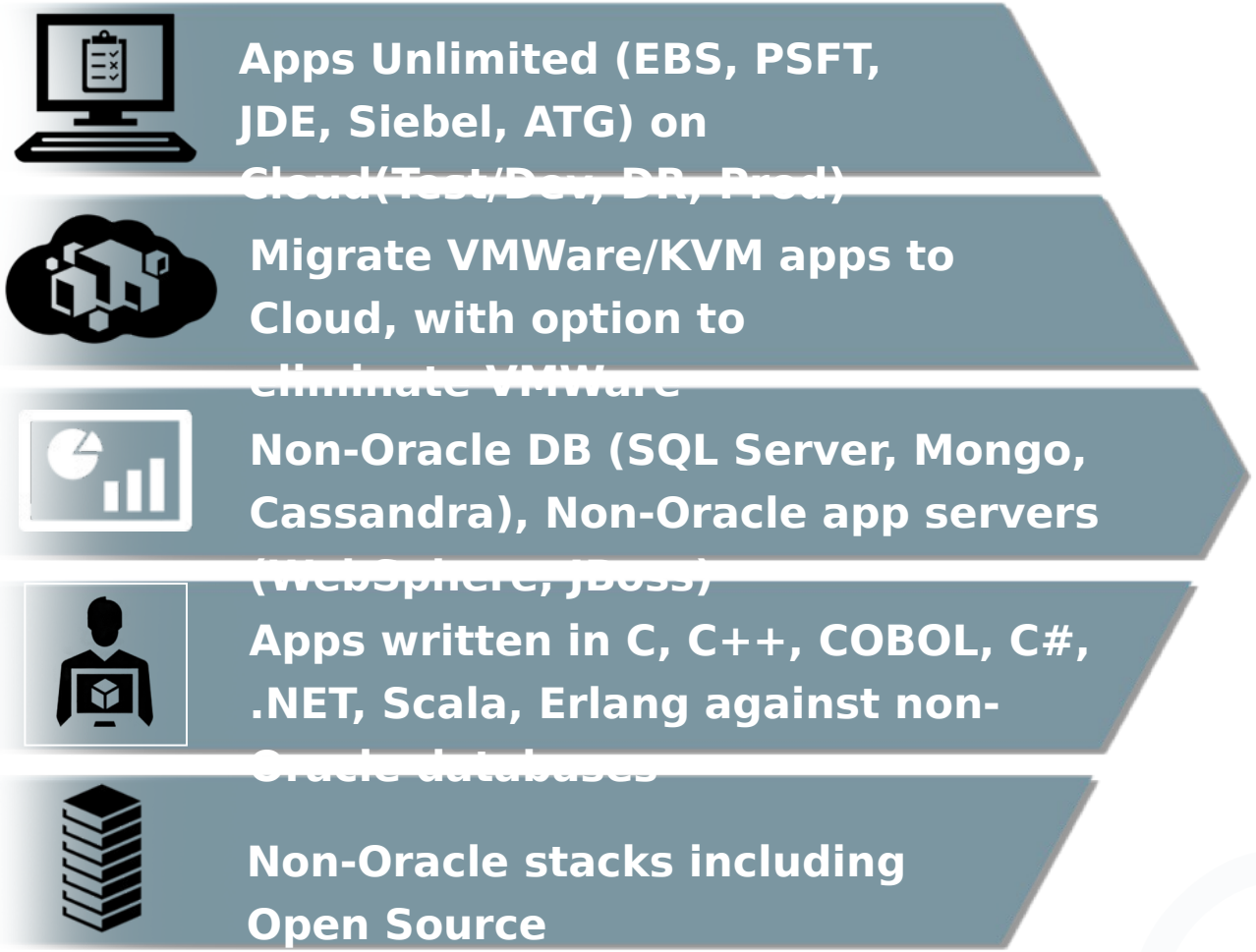
## Engineered Systems IaaS


Configure … Deploy … Orchestrate … Operate

ORACLE

# Oracle IaaS: Compute Use Cases

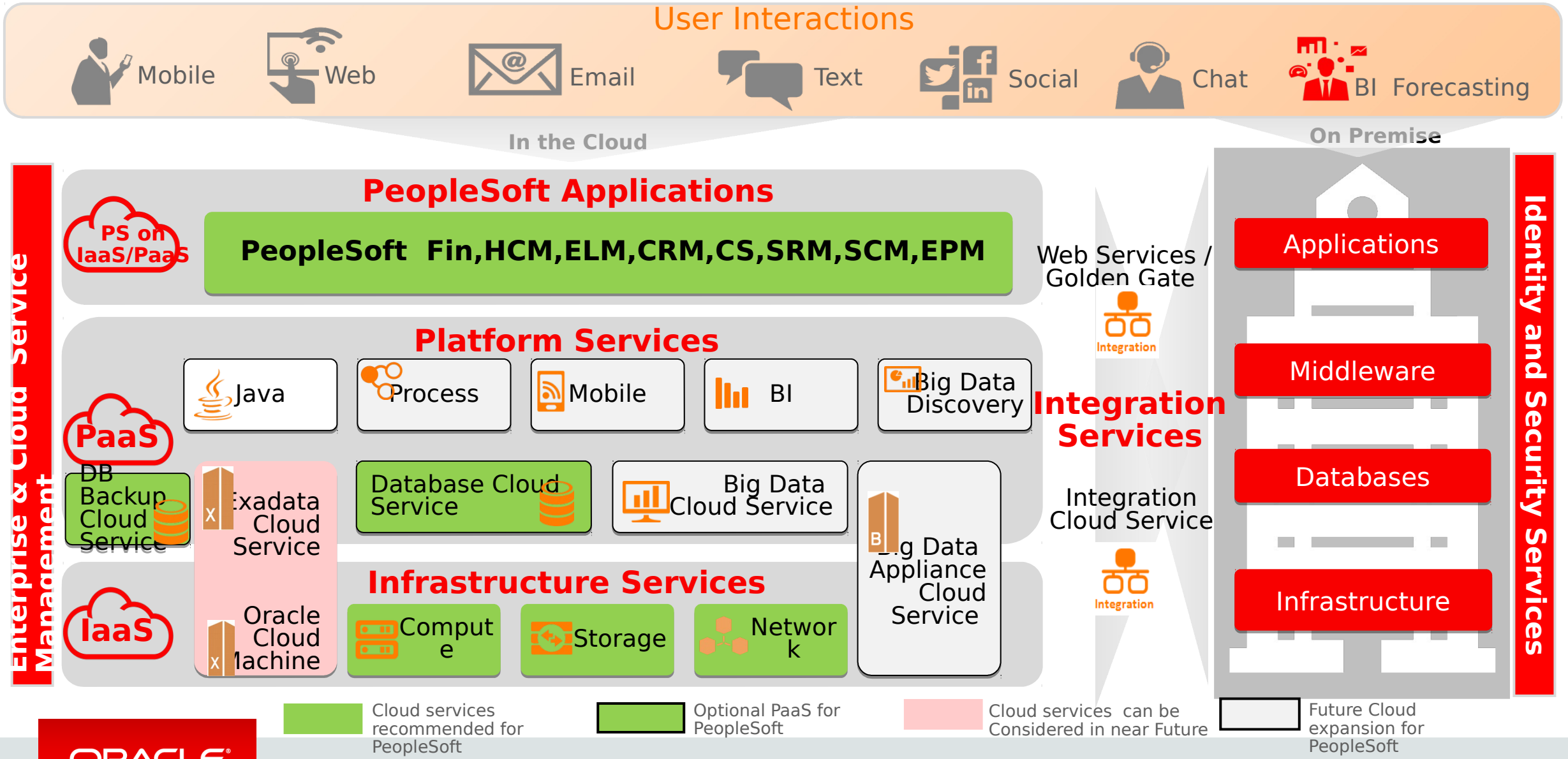
## Run heterogeneous, Oracle/Non-Oracle workloads in the cloud



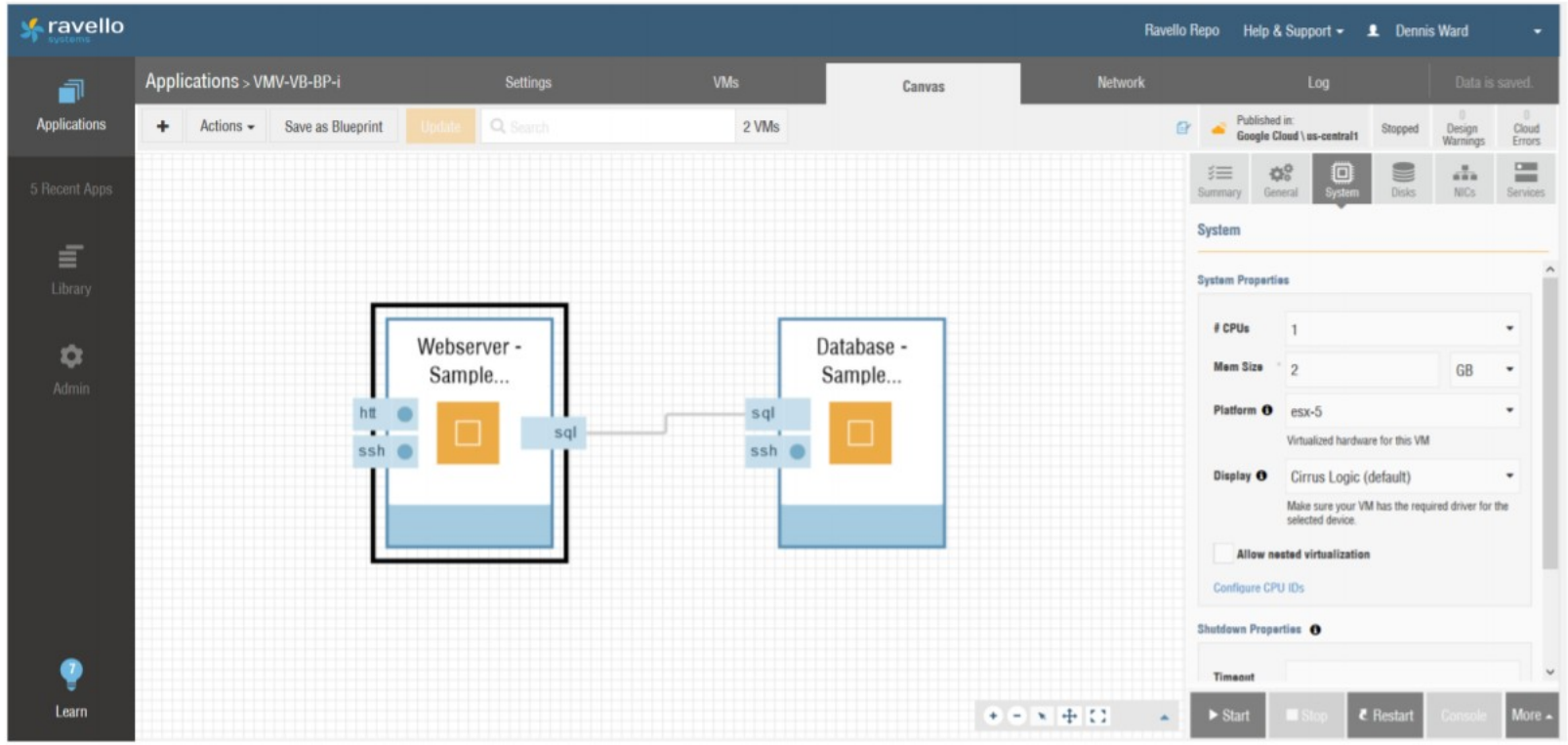

## Benefits:

- Lower run/manage costs for Oracle Applications by 30% compared to other clouds
- Simplify migration for large enterprise grade on-prem workloads
- Dedicated compute with predictable performance, network isolation
- Easy to adopt, transparent to applications & operational tools
- Highly secure with unified management
- Open, Flexible: REST APIs, OpenStack SWIFT/AWS S3 compatible

# Example: PSFT on Oracle Cloud



# Oracle Ravello: Consolidate VM installations to the cloud



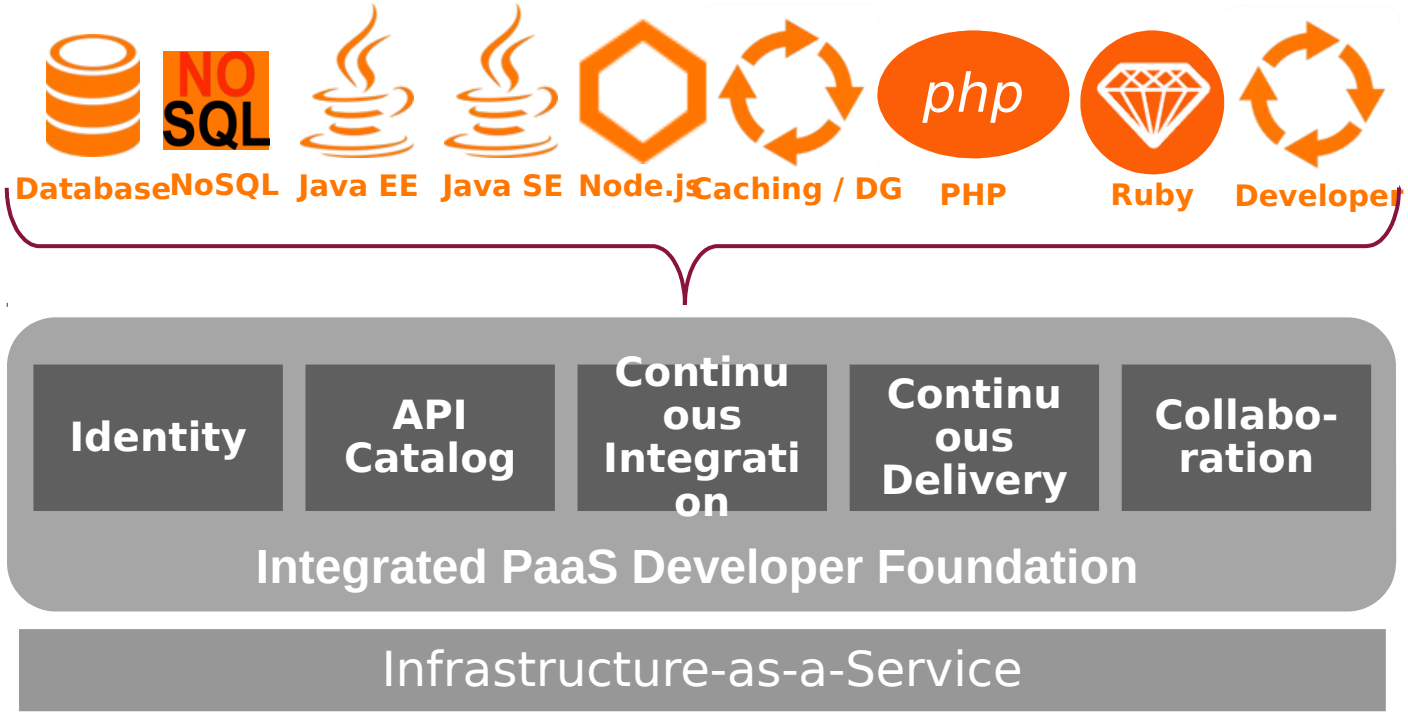

## Example Workflow

1. Import VMs into Ravello Library
   - From Vcenter, or directly upload .vmdk & .ova files.
2. Drag & Drop VMs into Canvas into desired topology
   - Network Discovery / Edit within Canvas
3. Click **Deploy** to Cloud of choice

# Agenda: *New Application Development Trends and Tools to modernize Healthcare IT*

- Introduction to 12-Factor application patterns: The opportunity for Healthcare IT
- New Oracle products and services for Developers and Architects
- **Summary / Next steps**

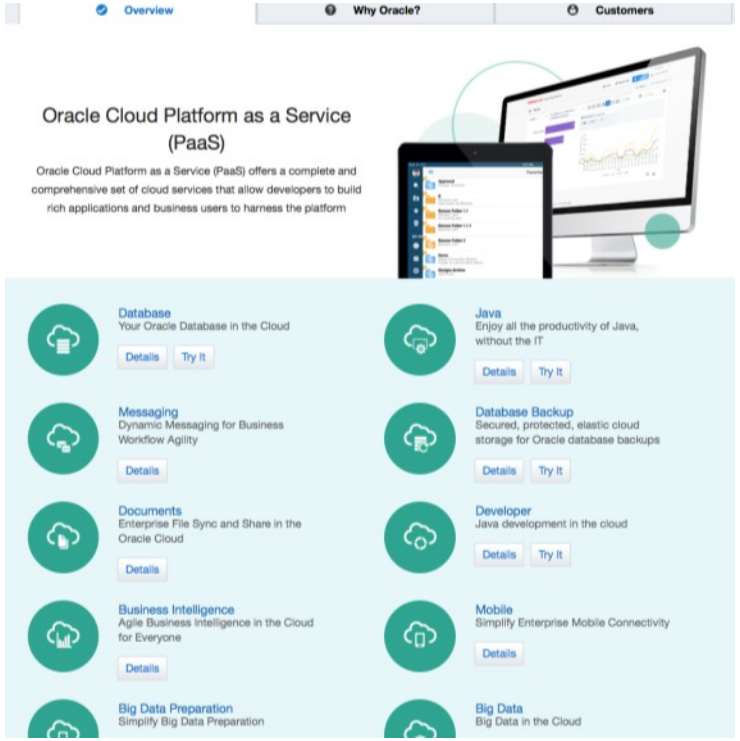
# Oracle Cloud Platform: For Application Development



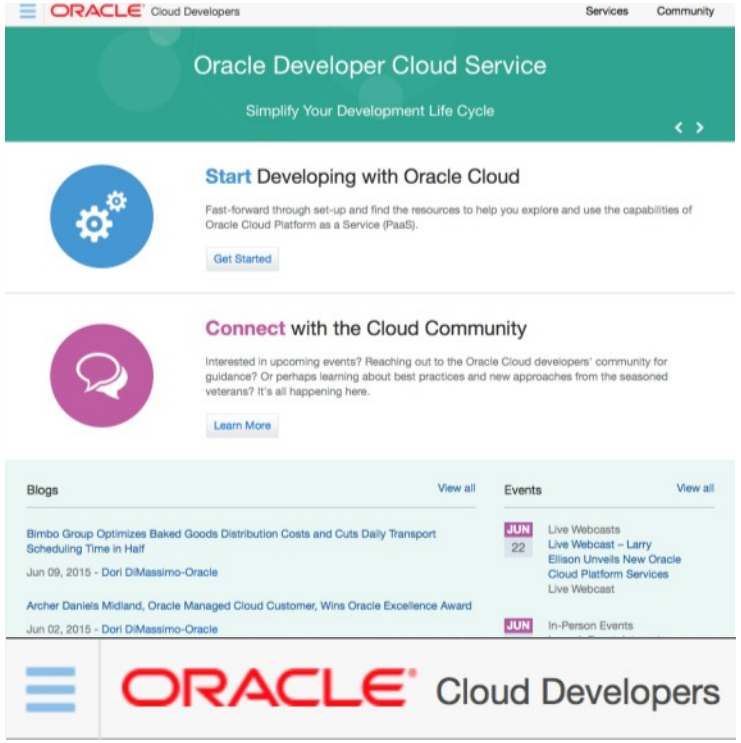

- Choice of Database and Programming languages – RDMS, NoSQL, Java EE, Java SE, Node.JS, PHP, Ruby
- Built in end to end lifecycle support – provision, backup/restore, scale, patch and disaster recovery – via API and UI
- Continuous integration and delivery with source control management, issue tracking, build & test & deployment
- Choice of IDEs – Eclipse, JDeveloper, Netbeans
- Integrated into Oracle PaaS & SaaS

# Learn More and Try It!

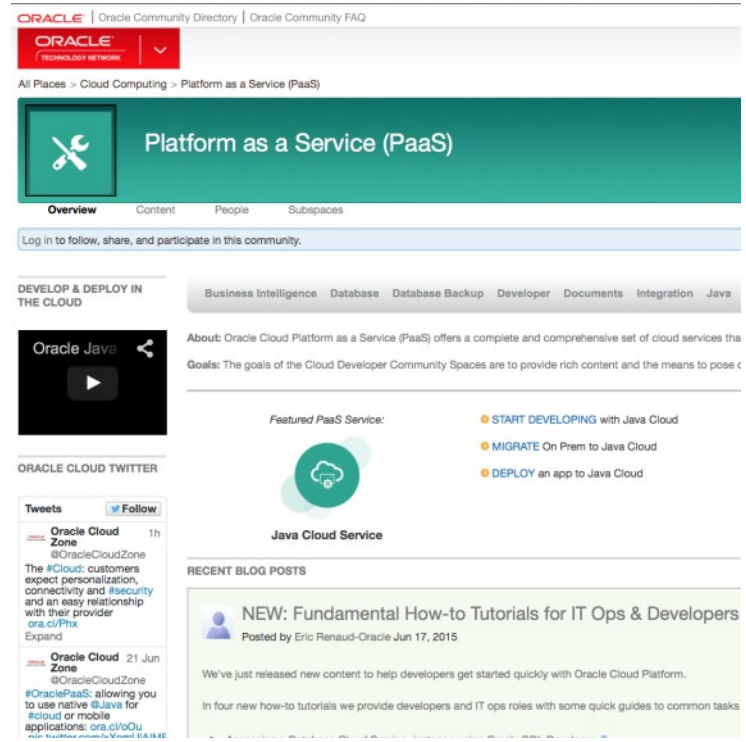
cloud.oracle.com/en_US/tryit




cloud.oracle.com/paas




developer.cloud.oracle.com




community.oracle.com

# Next steps?

- On-Depth Discussion your existing Software Development activities
  - Example: look at languages needed (Java? SQL? Others?) that best fit your development goals
  - New Applications you have planned (Open-Source, etc) ?
  - Look at ways to Extend, Enrich existing COTS applications?
  - Leverage your current WebLogic installation?
- In-depth discussion on use cases for IaaS (Compute, Network, Storage)
  - Oracle Ravello blueprints demo – show how to reduce Virtual Machine costs by moving to cloud
  - Options to improve flexibility of Research / Scientific computation workloads
  - Cheaper deployment of Open Source, Oracle, 3rd party applications

# Appendix

# Application Development part of Integrated Oracle PaaS

## DATA MANAGEMENT

- Database
- NoSQL Database (Q2FY17)
- Big Data
- Big Data SQL (Q1FY17)
- Database Backup
- Oracle Database Exadata

## APPLICATION DEVELOPMENT

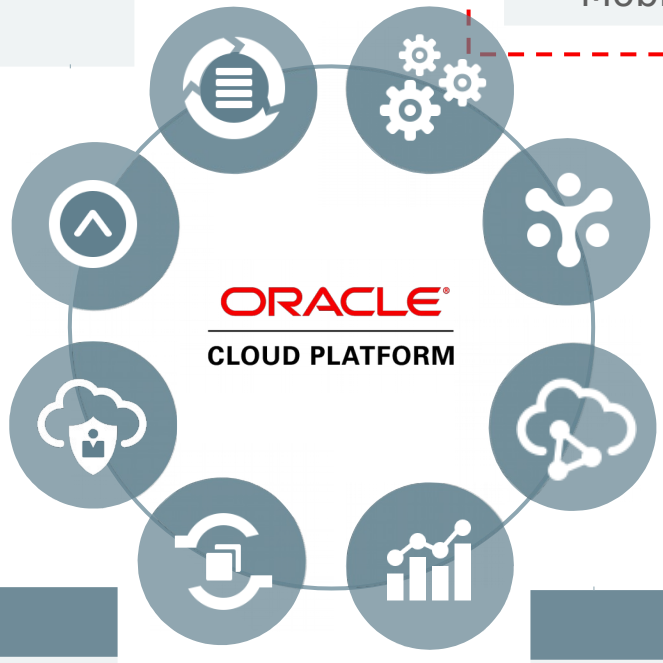
- Java
- Application Container
- Mobile
- Application Builder
- Developer

## MANAGEMENT

- IT Analytics
- Log Analytics
- Application Performance Monitoring

## ENTERPRISE INTEGRATION

- Integration
- SOA
- Managed File Transfer
- Internet of Things
- Process
- API Management




## SECURITY

- Identity (Q2FY17)

## DATA INTEGRATION

- GoldenGate
- Big Data Preparation

## CONTENT

- Documents
- Social
- Sites

## BUSINESS ANALYTICS

- Data Visualization
- Business Intelligence
- Big Data Discovery

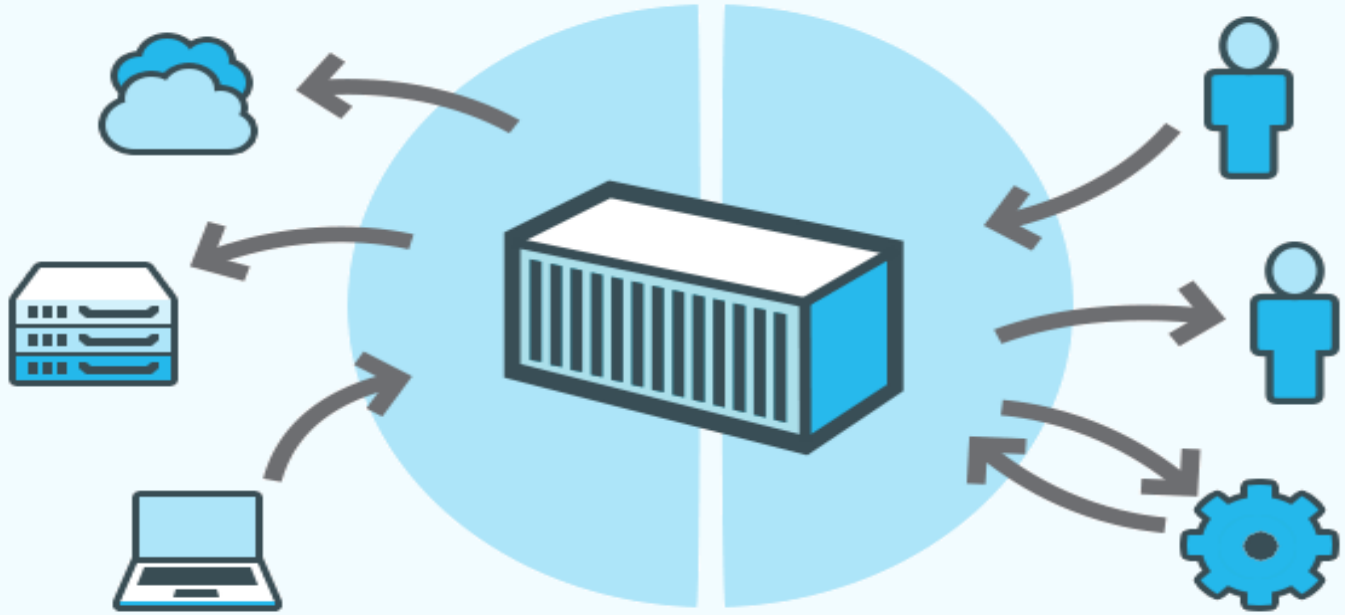
# 12-Factor Applications Architecture

## Main Principles

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.
- The twelve-factor methodology can be applied to apps written in **any programming language**, and which use any combination of backing services (database, queue, memory cache, etc).

# What Is Docker?

An open platform for distributed applications



## Docker Engine

A portable, lightweight application runtime and packaging tool.

## Docker Hub

A cloud service for sharing applications and automating workflows.

# Oracle Messaging Cloud Support for Microservices

## Implements AMQP standard

### Standardized Interfaces

*REST*

*JMS*

*Message push over HTTP*

### Versatile

*Oracle Cloud*

*On Premise*

*Hybrid*

### Delivery Choices

Pull

Push

Filter

### Reliability Mechanisms

*Transactions*

*Acknowledgements*

*Durable subscriptions*

# Oracle Infrastructure-As-A-Service
## why are we different

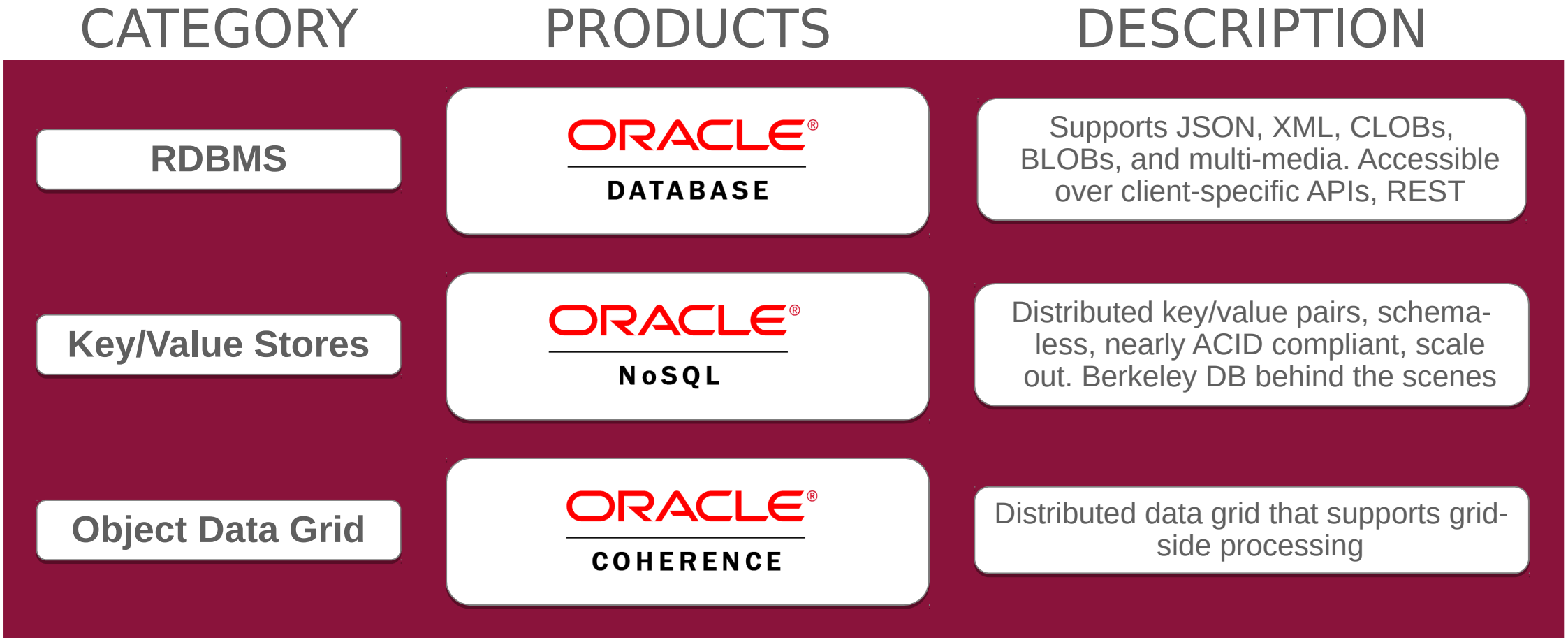
- **Simplifies Migration of Enterprise Workloads**
  - Nested Virtualization enables simpler migration of existing corporate workloads
  - By simplifying migration we also enable customers to both run a Hybrid Cloud mode
- **Broad Support of Heterogeneous Workloads**
  - Broadest support of OS (Linux, Windows, Ubuntu); Hypervisors (Xen, VMWare, KVM); Docker; API Compatibility with Swift and S3; …
- **More Predictable Performance and Security due to better isolation**
  - No CPU Over-subscription leads to more stable performance
  - Better Security due to stronger physical isolation (eg. with Dedicated Compute)
- **Packaged Solutions to migrate Oracle Applications Unlimited environments**
  - Cost savings in migrating EBS, Peoplesoft, JD-Edwards, Siebel workloads
- **Tight Integration with Oracle's Platform-as-a-Service**

# Enhanced Automated Issue Resolution

## Goal: no manual administrator fixes, should be 100% automated

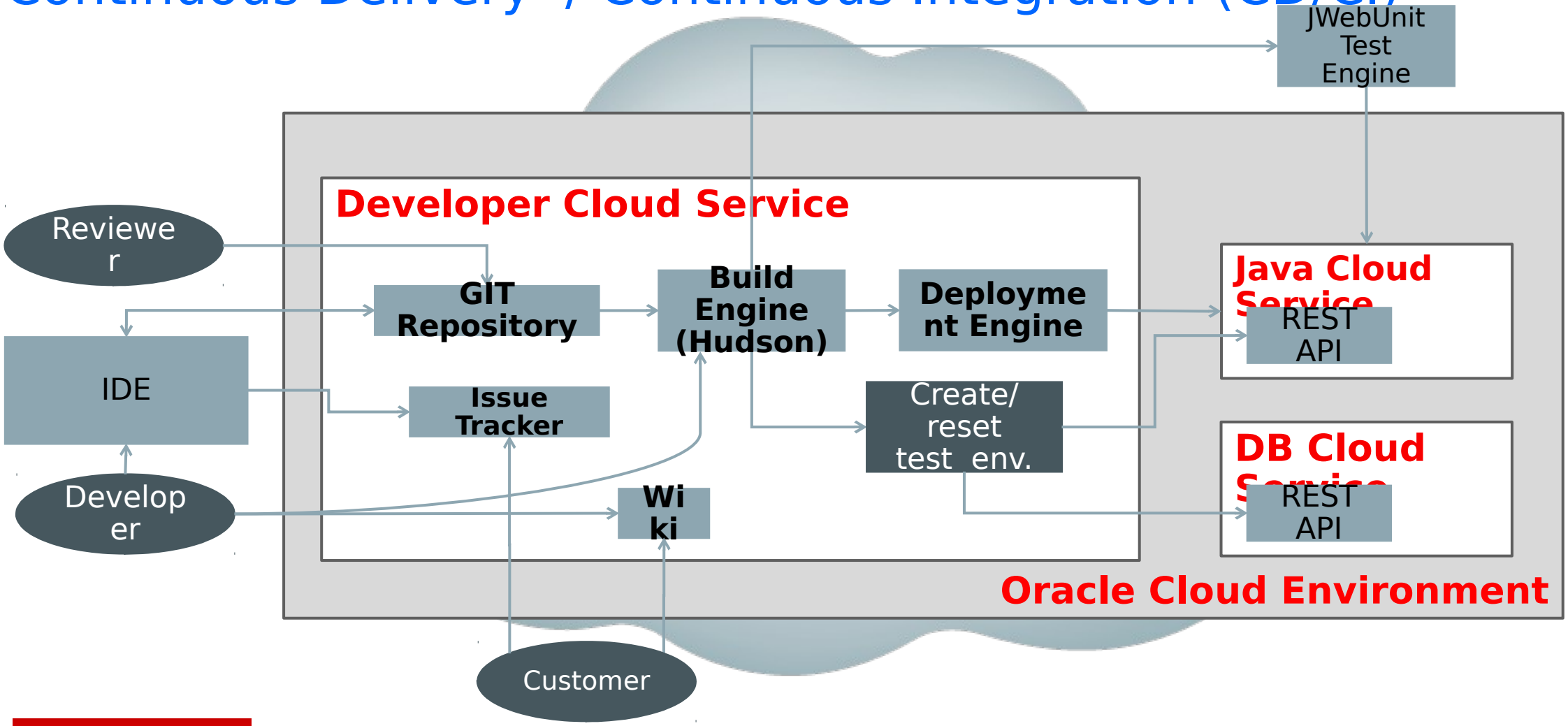
| | |
|---|---|
| **Hardware Failure**<br>Example: motherboard failed | Auto-scaling will automatically launch a new container on new hardware as load dictates |
| **Network Failure**<br>Example: switch failed | Auto-scaling will automatically launch new containers as load dictates |
| **System Software Failure**<br>Example: kernel panic | Health checking should fail and the container will be culled. Auto-scaling will automatically launch a new container as load dictates |
| **Application Software Failure**<br>Example: bad file permissions | Fix the source (your application, your container, your Dockerfile, etc) and re-deploy your entire application |

# Oracle Data*store* Options – Offered On Premise and In Cloud



# Oracle DevOps

## Continuous Delivery / Continuous Integration (CD/CI)



# Dockerfile example (Build an image)

```
#
# Super simple example of a Dockerfile
#
FROM ubuntu:latest
MAINTAINER Andrew Odewahn "odewahn@oreilly.com"
RUN apt-get update
RUN apt-get install -y python python-pip wget
RUN pip install Flask
ADD hello.py /home/hello.py
WORKDIR /home
```

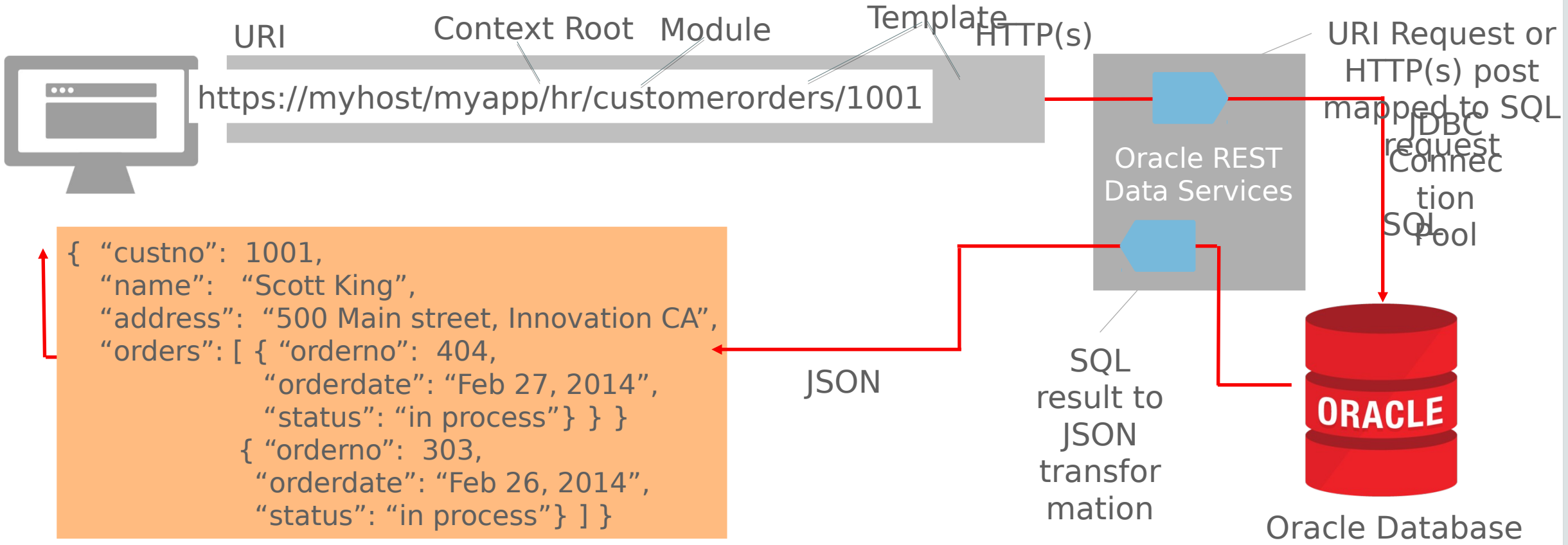
# node-oracledb

## v 1.4 released 11/2015

- A simple, stable Oracle Database driver with good out-of-the box performance
- Ongoing contributions from Oracle
  - Support for latest Oracle Database features
- Modular design
  - Underlying, simple DB access layer based on OCI
- **Open source** development, release and support under Apache 2.0 license
  - GitHub repository
  - Installable from NPM registry
  - Approx. monthly release cycle

# Oracle REST Data Services (ORDS)

## HTTP(s) API App-Dev with Relational Tables in Oracle Database



# Oracle Engineered Systems Cloud Services

**Engineered Systems IaaS**


- Exadata Cloud Service
  - Fastest Engineered System for Oracle DBMS – 1.8 Million IOPS
  - Flexible Configuration – Eighth, Quarter, Half, Full Racks
  - Self-Service Control – Create Instances, Monitor, Manage
  - Automated – Provisioning, Patching, Backup, Management, ..
  - Elastic – Scale Up or Down
- Big Data Cloud Service
  - Hadoop 2.0 – Standard Distribution, HDFS, Yarn, Hcatalog, Spark, ..
  - High Performance IO – Eliminates Bottleneck from CPU to Storage
  - Self-Service Control – Create One more More Clusters, Monitor, Manage
  - Automated – Provisioning, Patching, Backup, Management, ..
  - Elastic – Scale Up or Down