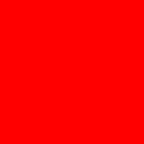# ORACLE®

Java EE 7: the New Cloud Platform

Peter Doschkinow
Senior Java Architect

The following/preceding 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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

- Java EE 7 as PaaS
  - Services
  - Deployment
  - Multi-tenancy
  - Roles
- Demo
- Java EE 7 API and Status
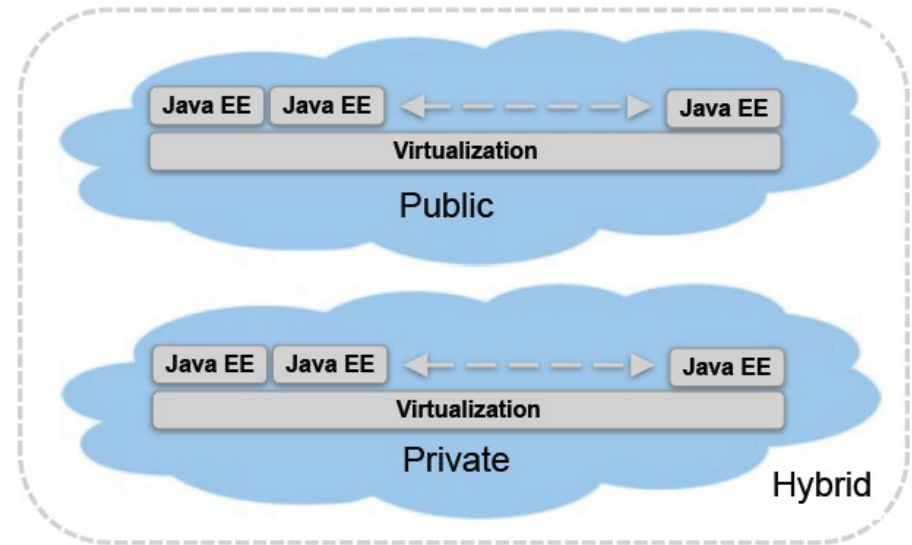
ORACLE®

Today's Cloud Offerings Are Vendor-Specific

Infrastructure as a Service

Platform as a Service
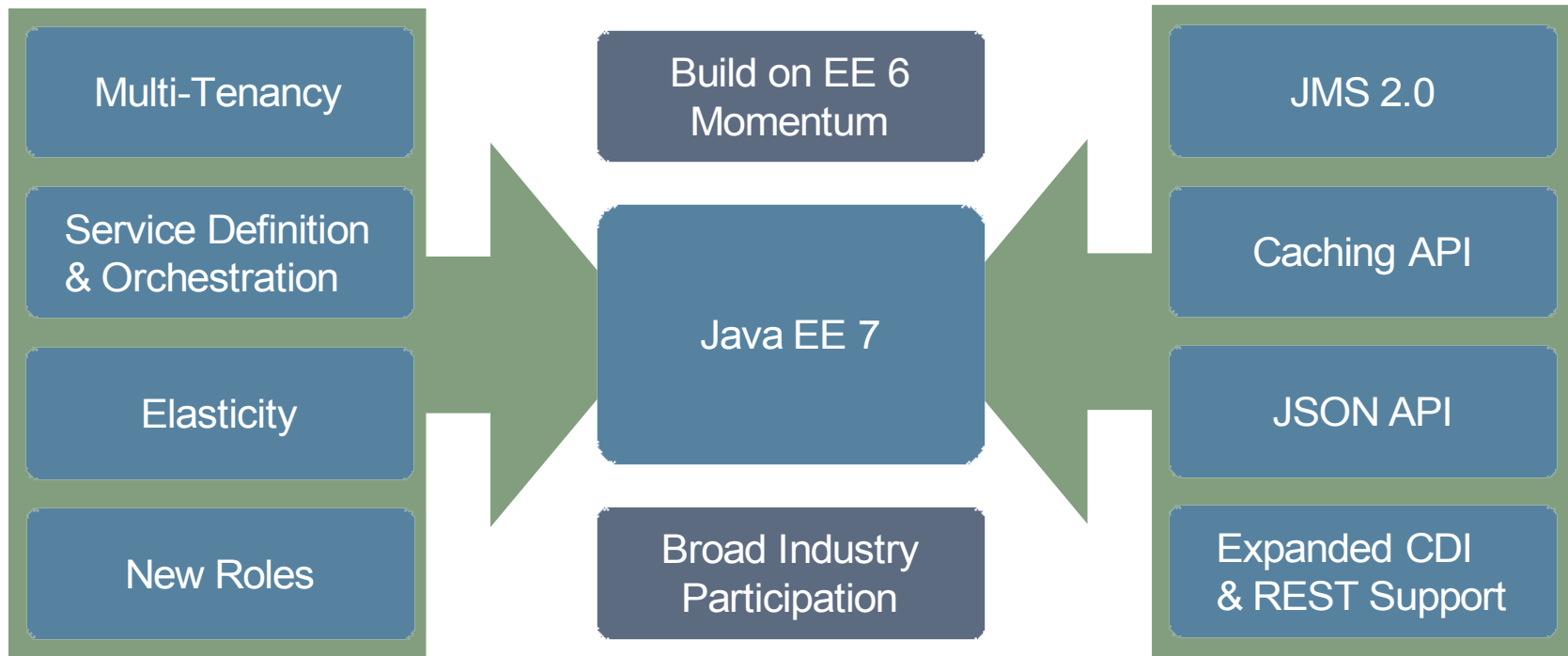
Software as a Service

Java EE 7 Focus: PaaS

- Provide way for customers and users to leverage public, private, and hybrid clouds in a standard way
- PaaS support entails evolutionary change
- Next logical step for Java EE
  - J2EE → Java EE 6 : The Java EE Platform provides services
  - Java EE 7 : The Java EE Platform IS a service

Java EE 7 PaaS Roadmap
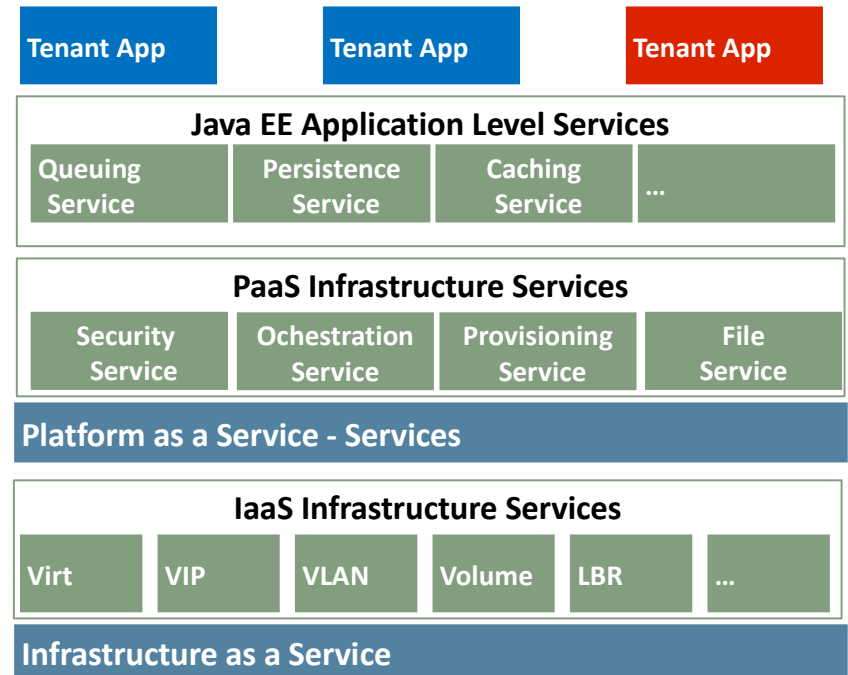
- Define new platform roles to accommodate PaaS model
- Add metadata
  – For service provisioning and configuration
  – For QoS, elasticity
  – For sharing of applications and resources
  – For (re)configurability and customization
- Add useful APIs for cloud environment
  – JAX-RS client API, Caching API, State Management, JSON,…
- Extend existing APIs with support for multitenancy

Java EE 7 Design Goals

Multi-Tenancy	Build on EE 6 Momentum	JMS 2.0
Service Definition & Orchestration		Caching API
Elasticity	Java EE 7	JSON API
New Roles	Broad Industry Participation	Expanded CDI & REST Support

ORACLE®

Java EE 7 as PaaS

- Tenant applications consume services

- PaaS administrators host, configure, and manage application and infrastructure services

- Existing APIs in Java EE need to be updated to be service-enabled and tenant-aware
  - Example: pluggable services

| Tenant App | Tenant App | Tenant App |

Java EE Application Level Services			
Queuing Service	Persistence Service	Caching Service	...

PaaS Infrastructure Services			
Security Service	Ochestration Service	Provisioning Service	File Service

Platform as a Service - Services

IaaS Infrastructure Services					
Virt	VIP	VLAN	Volume	LBR	...

Infrastructure as a Service

Java EE Services

- Cloud apps consume services
- Persistence, queueing, mail, caching, …
- Service metadata facilitates ease of use when deploying into the cloud

```
@DataSourceDefinition(
    name="java:app/jdbc/myDB",
    className="oracle.jdbc.pool.OracleDataSource",
    isolationLevel=TRANSACTION_REPEATABLE_READ,
    initialPoolSize=5
)
```

Java EE Services

- Cloud apps consume services
- Persistence, queueing, mail, caching, …
- Service metadata facilitates ease of use when deploying into the cloud

```
@JMSConnectionFactoryDefinition(
name="java:app/MyJMSCF",
className="javax.jms.QueueConnectionFactory",
resourceAdapterName="myJMSRA")

@JMSDestinationDefinition(
name="java:app/MyJMSQueue",
className="javax.jms.Queue",
destinationName="myQueue1")
```

Java EE Services

- Cloud apps consume services

- Persistence, queueing, mail, caching, …

- Service metadata facilitates ease of use when deploying into the cloud

```
@MailSessionDefinition(
name="java:app/mail/MySession",
host="somewhere.myco.com",
from="some.body@myco.com")
```

Java EE Services

- Cloud apps consume services
- Persistence, queueing, mail, caching, …
- Service metadata facilitates ease of use when deploying into the cloud

@ConnectorResourceDefinition(

name="java:app/myCustomConnector",
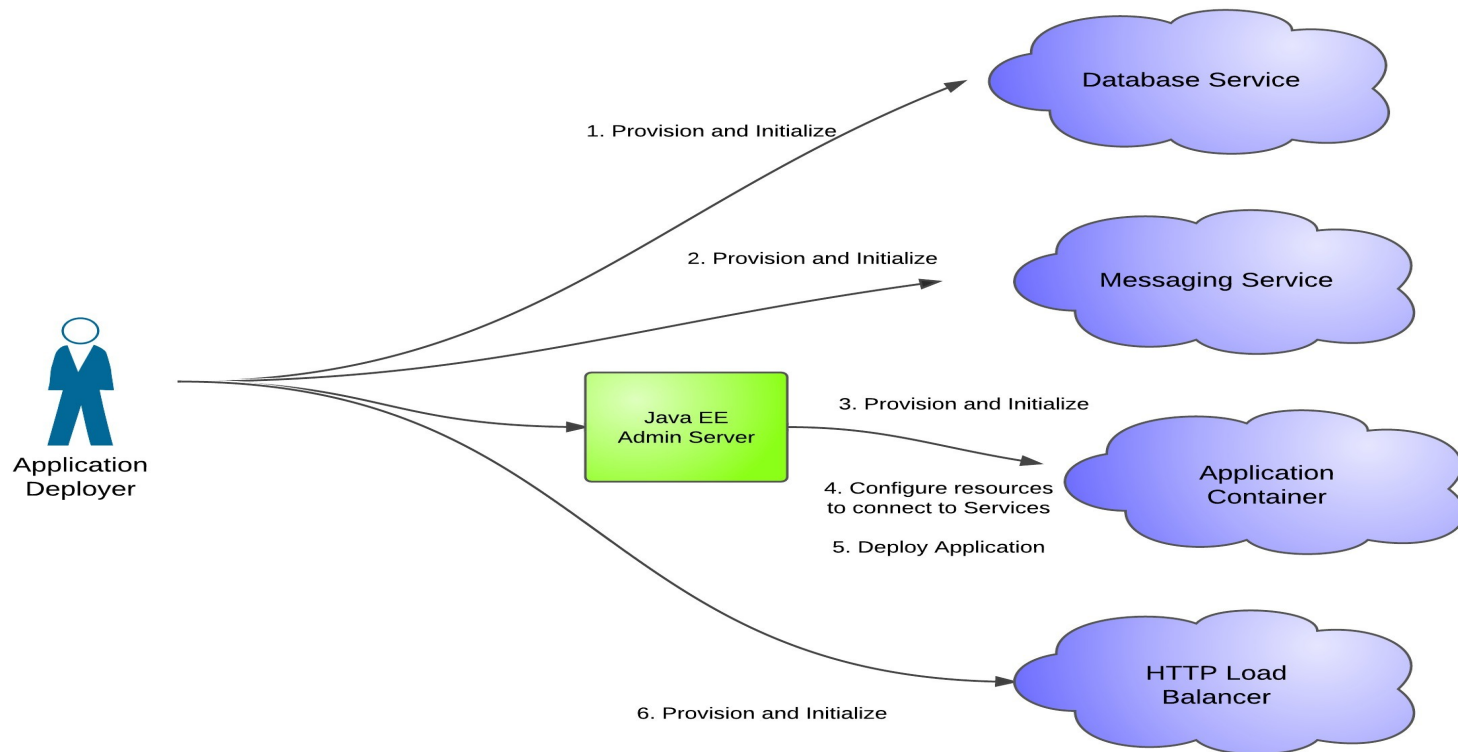
className="com.extraServices.CustomConnector")

Java EE 7 Service

- An entity that is created, provisioned, managed, and monitored by or for the PaaS runtime/infrastructure

- A significant software function that is needed to execute an application

- Types
  - Provisioned - installed, configured, and managed by the platform; application scoped or shared
  - External - already exist in the enterprise, platform is configured to know about them
  - Shared - used by multiple environments (per-user, and per-system)
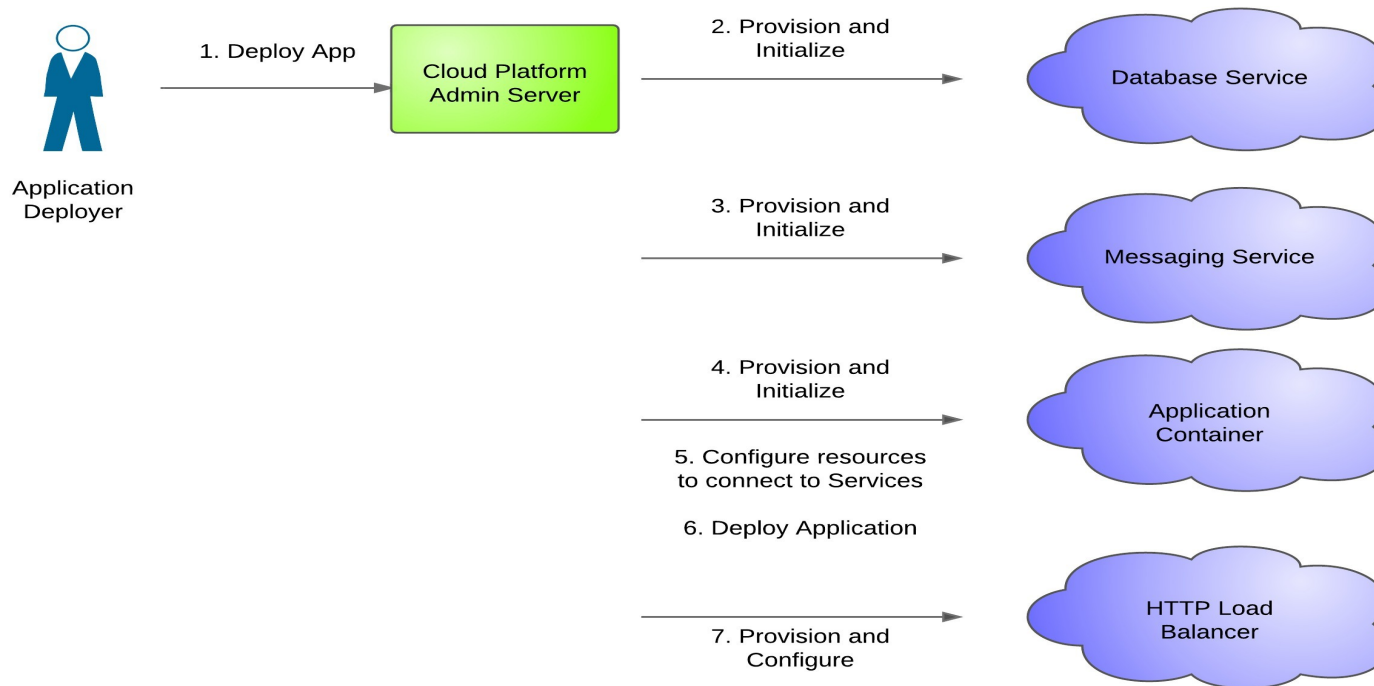
- Examples: LB, database, Java EE application service

Specification of Service Metadata

- ## Service Definition
  - Metadata used to provision and configure a Service
    - Service characteristics (functional and non-functional) specification → Template matching
    - Explicit Template specification
- ## Service Reference
  - Represents an application component's dependency on a Service
  - Explicit: User specified through Deployment Descriptors
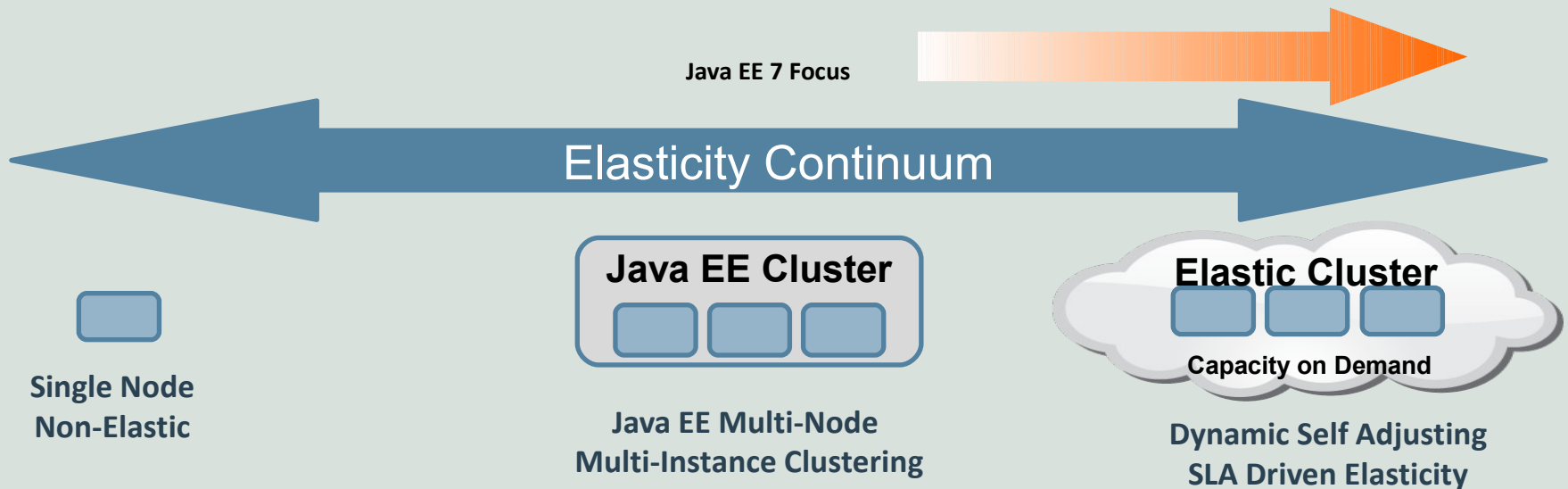 - Implicit and Discovered: Information contained within the archive

ORACLE®

Traditional Java EE App Deployment

Java EE 7 PaaS App Deployment

Java EE 7 Elasticity

Java EE 7 Focus

Elasticity Continuum

Single Node Non-Elastic

Java EE Cluster

Java EE Multi-Node Multi-Instance Clustering

Elastic Cluster

**Capacity on Demand**

Dynamic Self Adjusting SLA Driven Elasticity

- Cluster elasticity :
  - Metrics provided by application
  - Application Server metrics (response time, etc..)
  - Virtual Machine information (CPU, Memory, Disk usages)
- Metrics sources
  - JMX Mbeans, JVM Monitoring tools, native tools

ORACLE®

PaaS and Multi-tenancy: Some Models

- ## PaaS Platform on Demand
  - New runtime stack for each tenant

- ## PaaS Multitenant Containers
  - Isolated app partitions per tenant with shared runtime

- ## SaaS Multitenant Applications (SaaS-full)
  - Shared app instances, with tenant-specific customization

- ## SaaS-limited
 - Separate app instances, with tenant-specific customizations
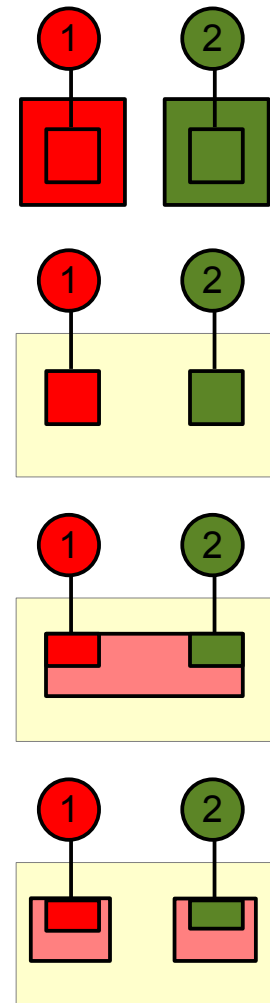
Java EE 7 Multitenancy
Limited form of SaaS

- Support for separate isolated instances of the same app for different tenants
  - Multitenant apps are declared as such
  - Tenants correspond to units of isolation
  - One application instance per tenant
  - Each instance customized and deployed for a single tenant
  - Limited form of SaaS
- Mapping to tenant done by the container
- Tenant id available to application
  - E.g., under java:comp/tenantId or by injection
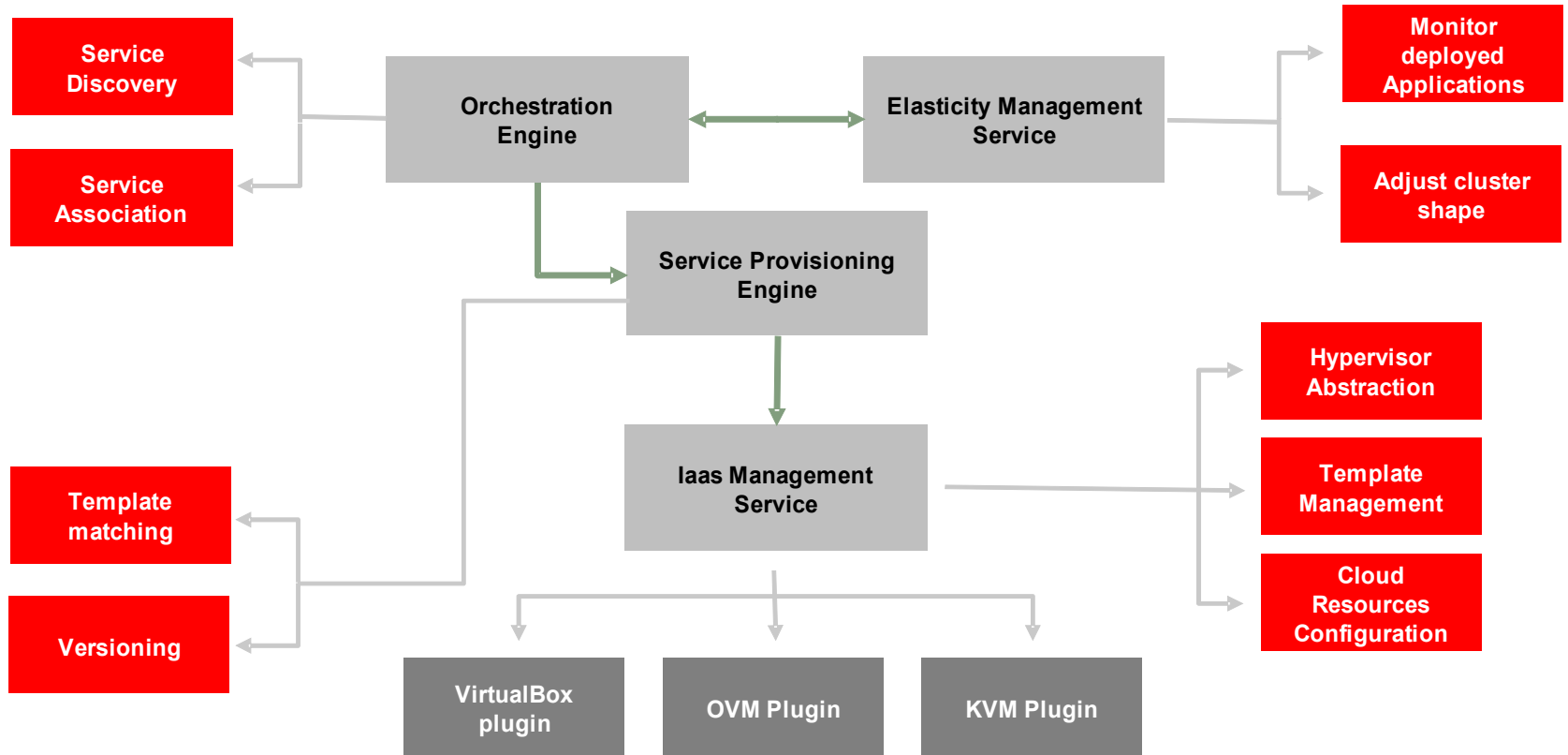
Java EE 7 Roles
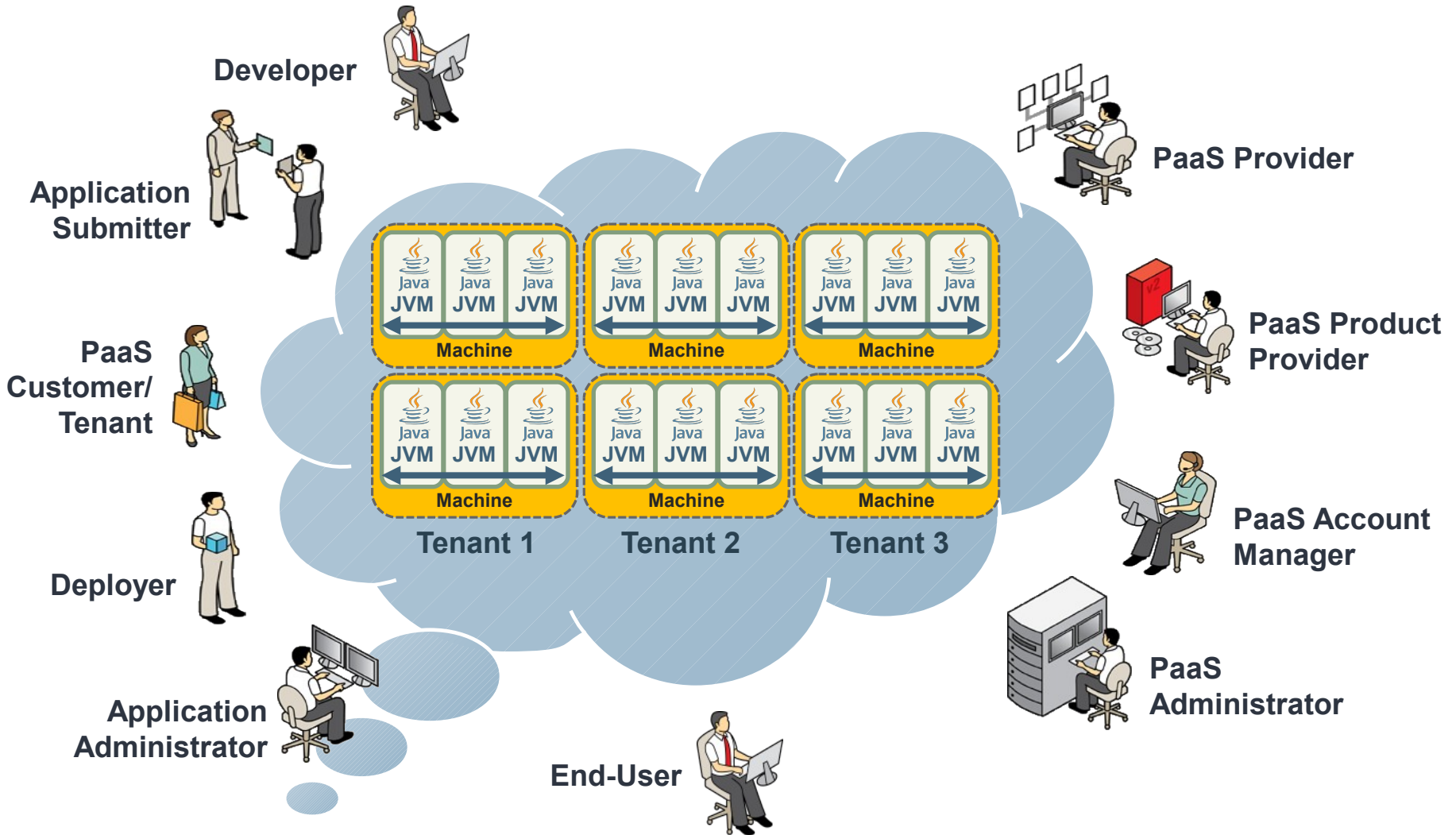
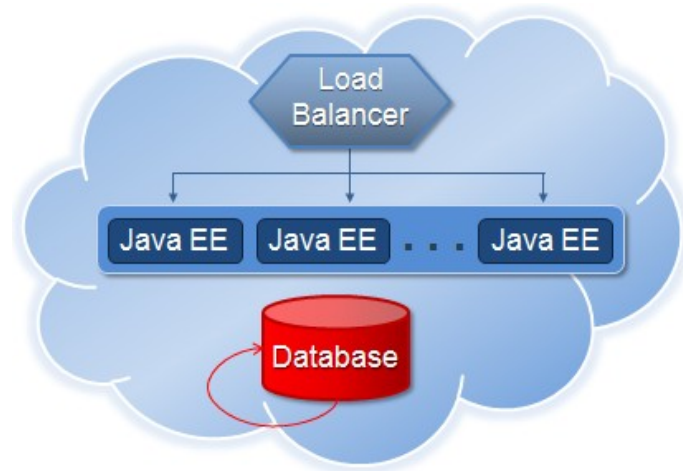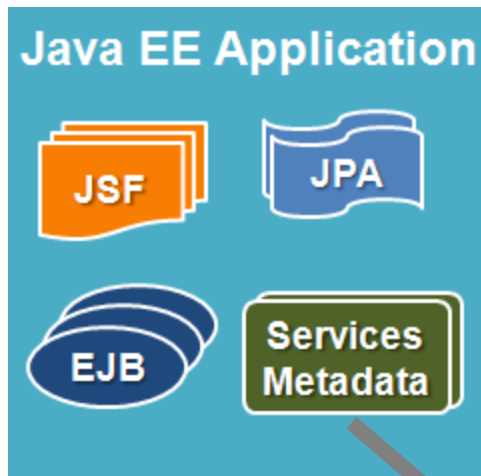# GlassFish Server 4.0

- Java EE 7 reference implementation
- Virtualization implementations
  - Laptop mode
    - Runs processes on the bare metal operating system.
  - Local mode
    - Locally installed hypervisor
    - Best fidelity to deployment scenario
  - Remote mode
    - Connects to remote hypervisors
- Transparent development

GlassFish in the Cloud

Service Discovery

Service Association

Orchestration Engine

Elasticity Management Service

Monitor deployed Applications

Adjust cluster shape

Service Provisioning Engine

Iaas Management Service

Hypervisor Abstraction

Template Management

Cloud Resources Configuration

Template matching

Versioning

VirtualBox plugin

OVM Plugin

KVM Plugin

PaaSing a Java EE Application
GlassFish 4.0 Demo at JavaOne: http://glassfish.org/javaone2011


```xml
<glassfish-services>
<service-description init-type="LB" name="ConferencePlanner-lb">
   <template id="LBNative"/>
   <configurations>
     <configuration name="https-port" value="50443"/>
     <configuration name="ssl-enabled" value="false"/>
     <configuration name="http-port" value="50080"/>
   </configurations></service-description>
<service-description init-type="JavaEE" name="ConferencePlanner">
   <characteristics>
     <characteristic name="service-type" value="JavaEE"/>
   </characteristics>
   <configurations>
     <configuration name="max.clustersize" value="4"/>
     <configuration name="min.clustersize" value="2"/>
   </configurations>
</service-description>
. . .
</glassfish-services>
```
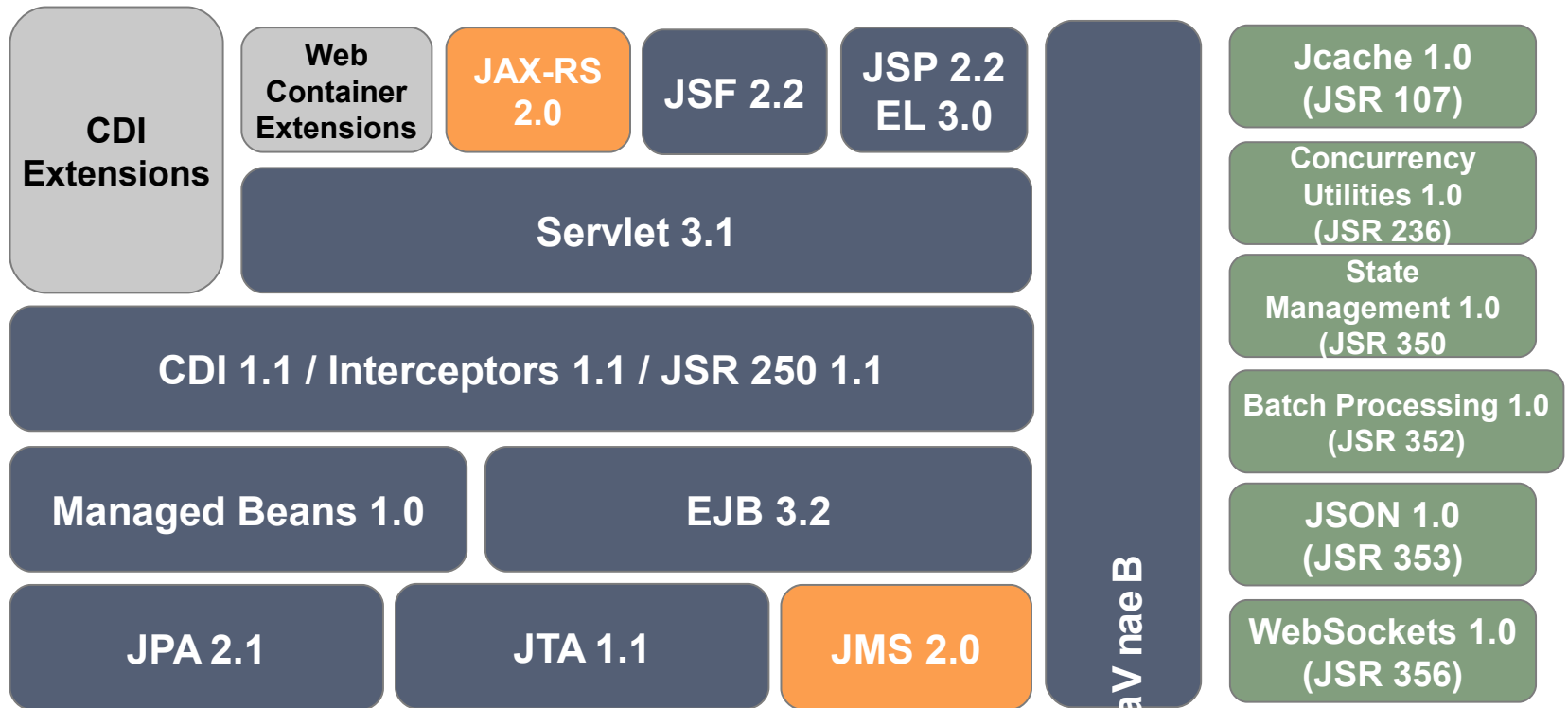
Demo

- Dynamic service provisioning
  - Service dependencies are discovered from metadata and by application archive introspection
  - Discovered services (Java EE, database, load balancer) are provisioned

- Highly available cluster
  - With session failover

- Elasticity using auto-scaling
  - The Java EE cluster is automatically resized to meet growing demands

ORACLE®

Java EE 7 is not just Cloud-y

- Alignment of ManagedBeans across CDI, EJB, JSF,…
  - POJO → ManagedBean → Enterprise JavaBean
  - Extension of container-managed transactions beyond EJB
- Further simplifications for ease-of-development
  - JMS 2.0 focus on ease-of-development
  - Expanded use of dependency injection
  - Expanded service metadata; improved configuration
- Pruning
  - EJB CMP and BMP, JAX-RPC, Deployment API
- Update to Web Profile

Java EE 7 JSRs

CDI Extensions

Web Container Extensions

JAX-RS 2.0

JSF 2.2

JSP 2.2 EL 3.0

Servlet 3.1

CDI 1.1 / Interceptors 1.1 / JSR 250 1.1

Managed Beans 1.0

EJB 3.2

JPA 2.1

JTA 1.1

JMS 2.0

Java VB

Jcache 1.0 (JSR 107)

Concurrency Utilities 1.0 (JSR 236)

State Management 1.0 (JSR 350

Batch Processing 1.0 (JSR 352)

JSON 1.0 (JSR 353)

WebSockets 1.0 (JSR 356)

ORACLE®

Transparency Checklist
http://jcp.org/en/resources/transparency

- Our Java EE 7 JSRs are run in the open on java.net
  - http://javaee-spec.java.net
  - One project per spec – e.g., jpa-spec, jax-rs-spec, jms-spec, …
- Publicly viewable Expert Group mail archive
  - Users observer list gets copies of all Expert Group emails
- Publicly viewable download area
- Publicly viewable issue tracker
- Commitment to match JCP 2.8 Process

ORACLE®

Java EE 7 Status and Schedule

- All JSRs up and running
- Early drafts available for:
    - Java EE 7 (JSR 342)
    - Expression Language 3.0 (JSR 341)
    - Java Message Service 2.0 (JSR 343)
    - Enterprise JavaBeans 3.2 (JSR 345)
    - Contexts and Dependency Injection 1.1 (JSR 346)
    - Bean Validation 1.1 (JSR 349)
    - JavaServer Faces 2.2 (JSR 344)
    - Java Persistence API 2.1 (JSR 338)
    - Java API for RESTful Services 2.0 (JSR 339)
- Final release by Q2 2013
- Date-driven release: anything not ready will be deferred

ORACLE®

Java Message Service 2.0

- Simplified API
  - Less verbose
  - Reduce the number of objects needed to send/receive message
  - Allow resource injection
  - Alternative, not replacement for standard API
- New mandatory API for integration of any JMS 2.0 provider with any Java EE server
- Connection, Session and other objects are AutoClosable

ORACLE

Java Message Service 2.0
sending a message the old way

```
@Resource(lookup = "jms/connectionFactory ")
ConnectionFactory connectionFactory;
@Resource(lookup="jms/inboundQueue")
Queue inboundQueue;
public void sendMessageOld (String payload) {
  Connection connection = null;
  try {
        connection = connectionFactory.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer messageProducer = session.createProducer(inboundQueue);
        TextMessage textMessage = session.createTextMessage(payload);
        messageProducer.send(textMessage);
  }
  catch (JMSException e) {
  // do something
  }  finally {
   try {
        if (connection != null)
           connection.close();
  } catch (JMSException e2) {
        // do something else    }}}
```
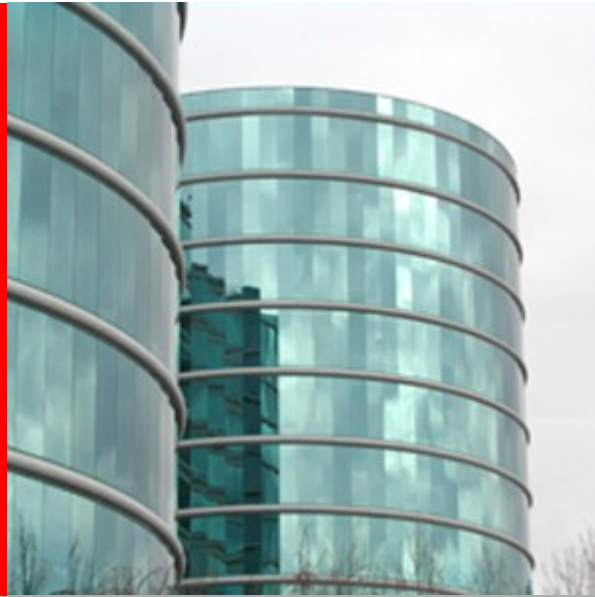
Java Message Service 2.0
sending a message the new way

```
@Resource(lookup = "jms/connectionFactory")
ConnectionFactory connectionFactory;
@Resource(lookup="jms/inboundQueue")
Queue inboundQueue;

public void sendMessageNew (String payload) {
        try (JMSContext context = connectionFactory.createContext();){
                context.send(inboundQueue,payload);
        }
}
```

Links

- Java EE 7 java.net project
  - Archives, documents, mailing lists,…
  - http://java.net/projects/javaee-spec
- Component projects
  - http://java.net/projects/XXX-spec

    (where XXX = jpa, ejb, jms, servlet, jax-rs, jsf,…)
- GlassFish – Java EE 7 reference implementation
  - http://glassfish.org
- Feedback
  - users@javaee-spec.java.net

Java EE 7: the new Cloud Platform

Peter Doschkinow
Senior Java Architect