PaaSing a Java EE Application

Arun Gupta, Java EE & GlassFish Guy

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

Cloud Computing

Essential Characteristics

- On-demand Self-Service
- Broad network access
- Resource Pooling
- Rapid Elasticity
- Measured Service

(*) NIST Definition of Cloud Computing – 800-145

Cloud Computing

Deployment Models

- Private Cloud
  - Operated solely for an organization
  - On-premise or off-premise
- Community Cloud
- Public Cloud
  - Access to general public
- Hybrid Cloud
  - 2 or more clouds, Cloud Bursting

(*) NIST Definition of Cloud Computing – 800-145

Cloud Computing

Service Models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
  - – Deploy customer-created applications
  - – Using languages and tools supported by PaaS Provider
  - – No control of underlying cloud infrastructure
  - – Control over deployed applications, hosting env. Configurations
- Infrastructure as a Service (IaaS)

(*) NIST Definition of Cloud Computing – 800-145

PaaS and Java EE

Java EE design principles and capabilities

- Common programming model for enterprise developers
- Runtime handles application’s infrastructure concerns
- Declarative resource references
- Scalable (scale-out) component models

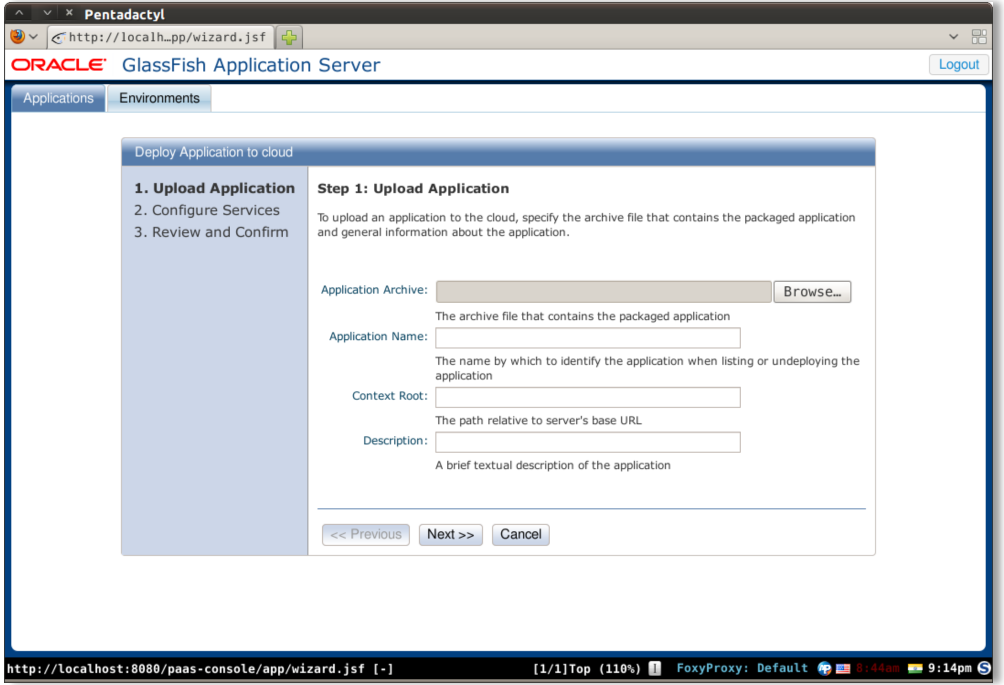
PaaS and Java EE

Java EE 7: “Making Java EE ready for the cloud”

- Enhancements
  - New Roles for PaaS
  - Services as first class citizens
  - Multitenancy
- Evolution, not a revolution!

PaaS Implications on Deployment

- Simplified PaaS Application Deployment
  - Single-click, self-service, “push to cloud”


Pentadactyl
http://localh…pp/wizard.jsf
ORACLE GlassFish Application Server
Logout
Applications
Environments
Browse…
<< Previous
Next >>
Cancel
http://localhost:8080/paas-console/app/wizard.jsf [-]
[1/1]Top (110%)
FoxyProxy: Default
9:14pm

Deploy Application to cloud

1. Upload Application
2. Configure Services
3. Review and Confirm

Step 1: Upload Application

To upload an application to the cloud, specify the archive file that contains the packaged application and general information about the application.

Application Archive:
The archive file that contains the packaged application

Application Name:
The name by which to identify the application when listing or undeploying the application

Context Root:
The path relative to server's base URL

Description:
A brief textual description of the application

PaaS Implications on Deployment

Services Management

- Automatic Service Provisioning and Management
  - Service Orchestration
    - Automatic Service Dependency discovery
    - Service Provisioning and Association
  - Handle operational infrastructure concerns automatically
    - Network configuration, HA, Clustering, Load Balancing …
  - Application and Service deployment versioning

PaaS Implications on Deployment

Virtualized runtimes

- Scalable virtualized on-demand environment
  - Support multiple cloud deployment models
    - Public, Private, Hybrid
  - PaaS Provider decoupled from IaaS infrastructure
  - Multi-tenancy

PaaS Implications on Deployment

Scaling and Operations

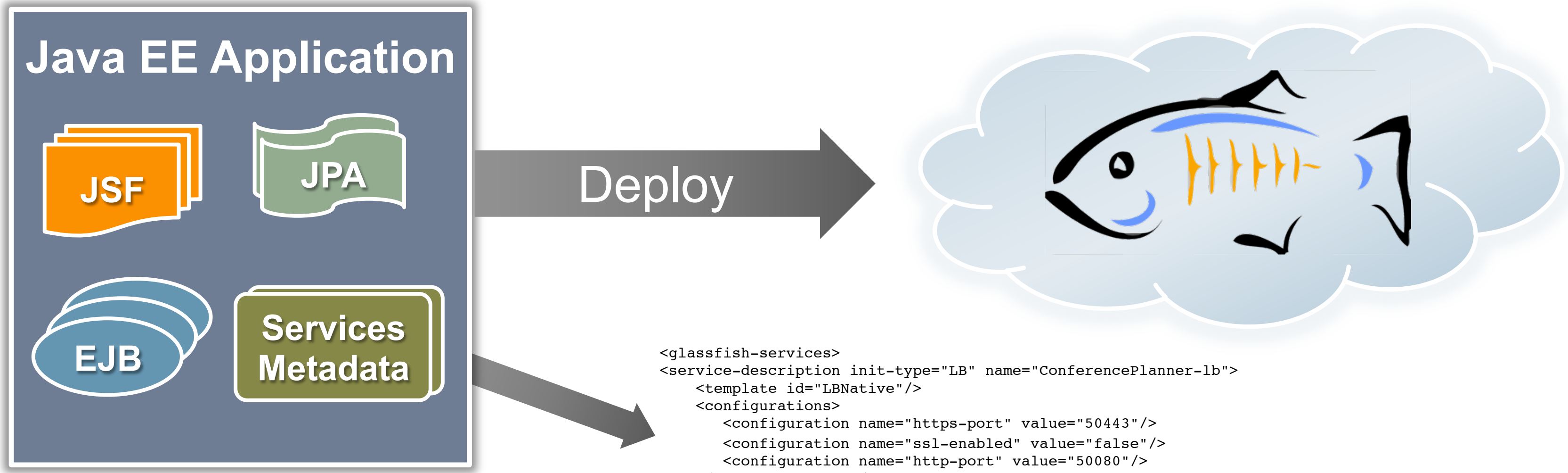
- Automatic Scaling of Services
  - Scale to application's needs
  - User-defined alerts and actions
- Control over application hosting environment
  - Flexibility in choice of application services, framework
  - Rich service configuration
  - Shared services
  - Extensible runtime to allow new Services

Demo

PaaSing a Java EE Application in the Cloud

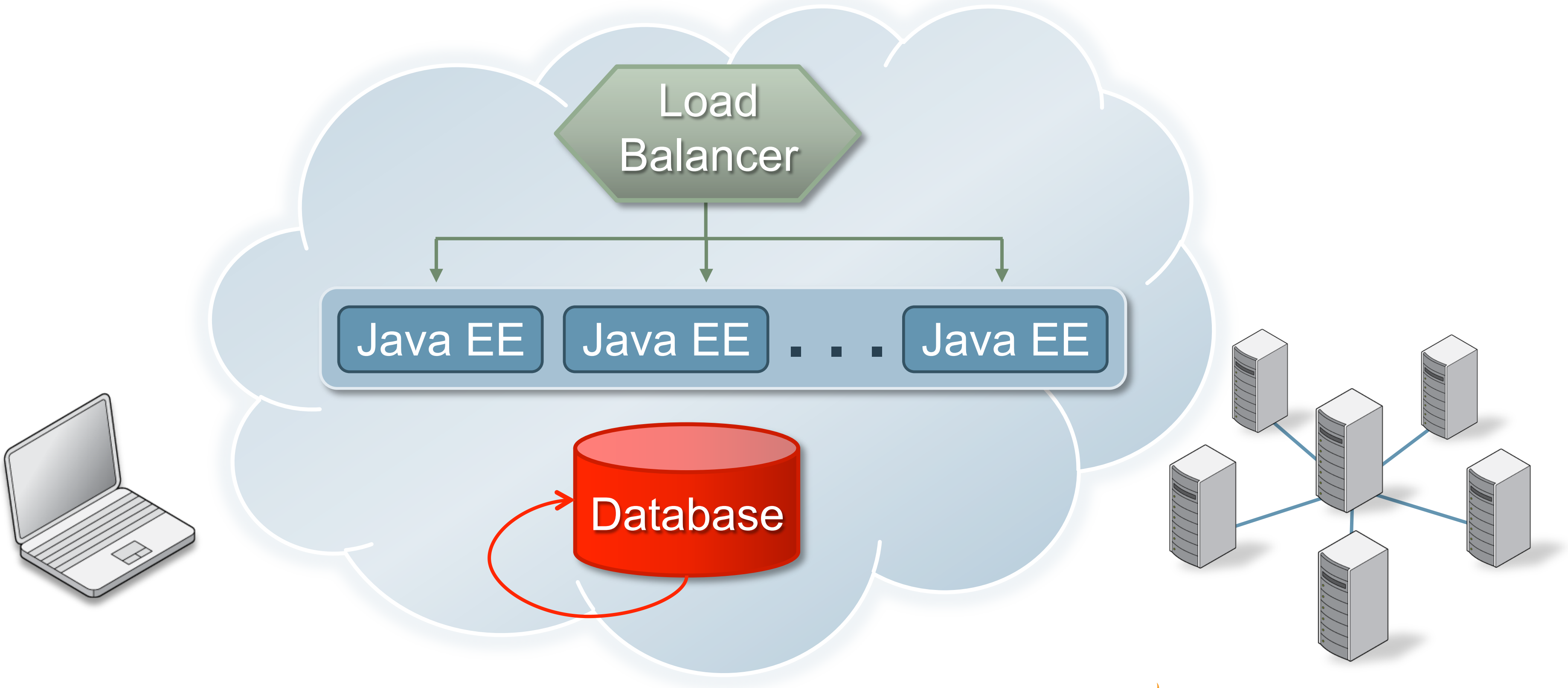
glassfish.org/javaone2011

Conference Planning in the Cloud

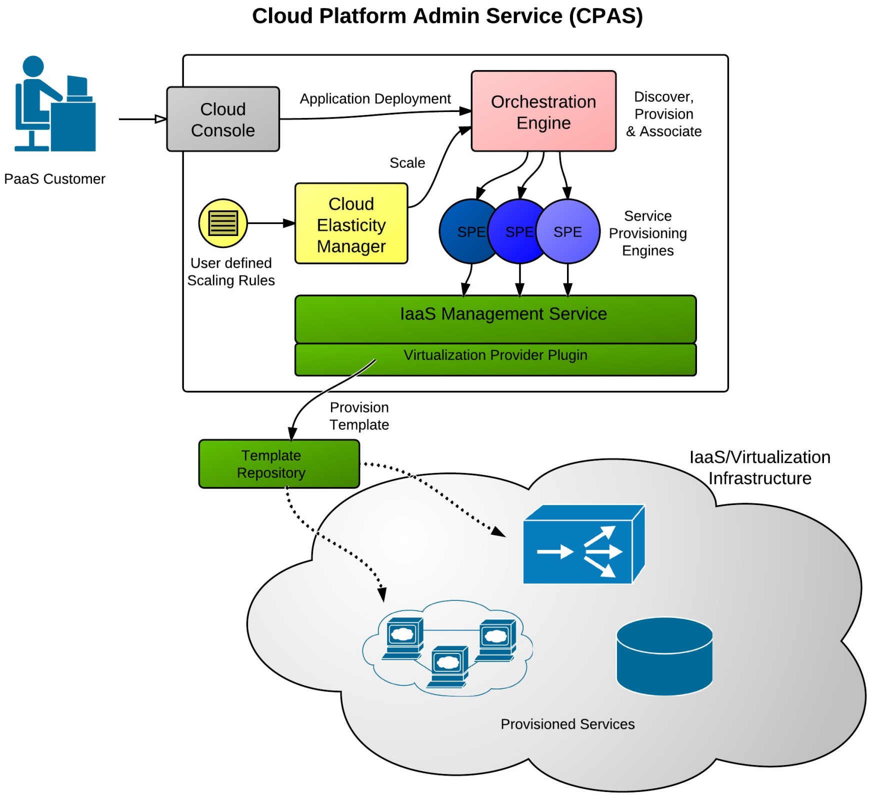


```
<glassfish-services>
<service-description init-type="LB" name="ConferencePlanner-lb">
    <template id="LBNative"/>
    <configurations>
       <configuration name="https-port" value="50443"/>
       <configuration name="ssl-enabled" value="false"/>
       <configuration name="http-port" value="50080"/>
    </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>
```

Service Provisioning



GlassFish PaaS Runtime Architecture



Service Terminology

- Service
- ServiceType
  - – Java EE, RDBMS, HTTP Load Balancer etc.
- Services – scope and lifecycle
  - – Provisioned Services
    - Application scoped
    - Shared
  - – External (*a priori*) services

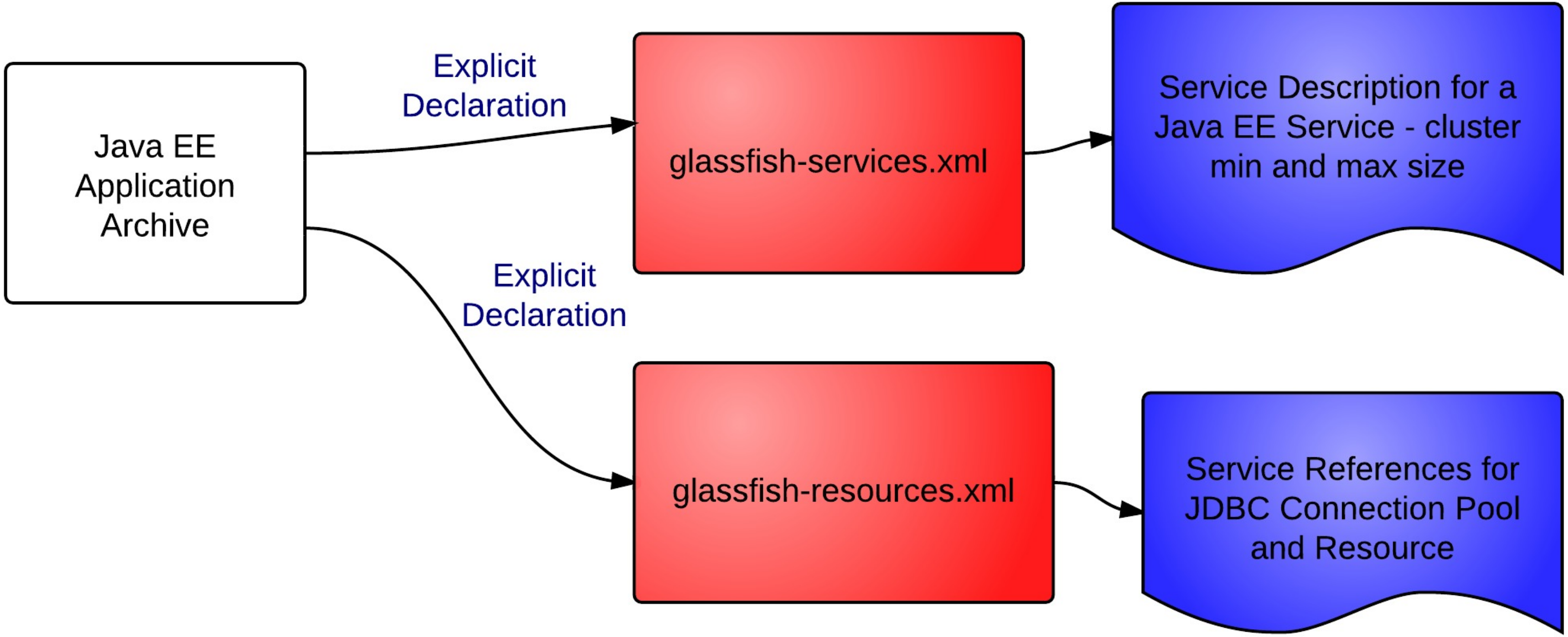
Specification of Service Metadata

- Optional!
  - When not specified (vanilla EE app archives)
    - Orchestration Engine automatically handles discovery of service deps
    - Automatic wiring to default Service Templates
  - Metadata may be specified when:
    - Finer grain control of application environment desired
    - Application-specific Service configuration

Specification of Service Metadata

- Service Definition
  - Metadata used to provision and configure a Service
    - ***What :*** Service characteristics (functional and non-functional) → Template matching
    - ***How :*** Explicit Template specification → Template wiring
- Service Reference
  - An application component's dependency on a Service
    - **Explicit :** User-specified through deployment descriptors
    - **Implicit** and **Discovered**: Information contained within the archive

Service Dependency Specification



Java EE Application Archive

Explicit Declaration

glassfish-services.xml

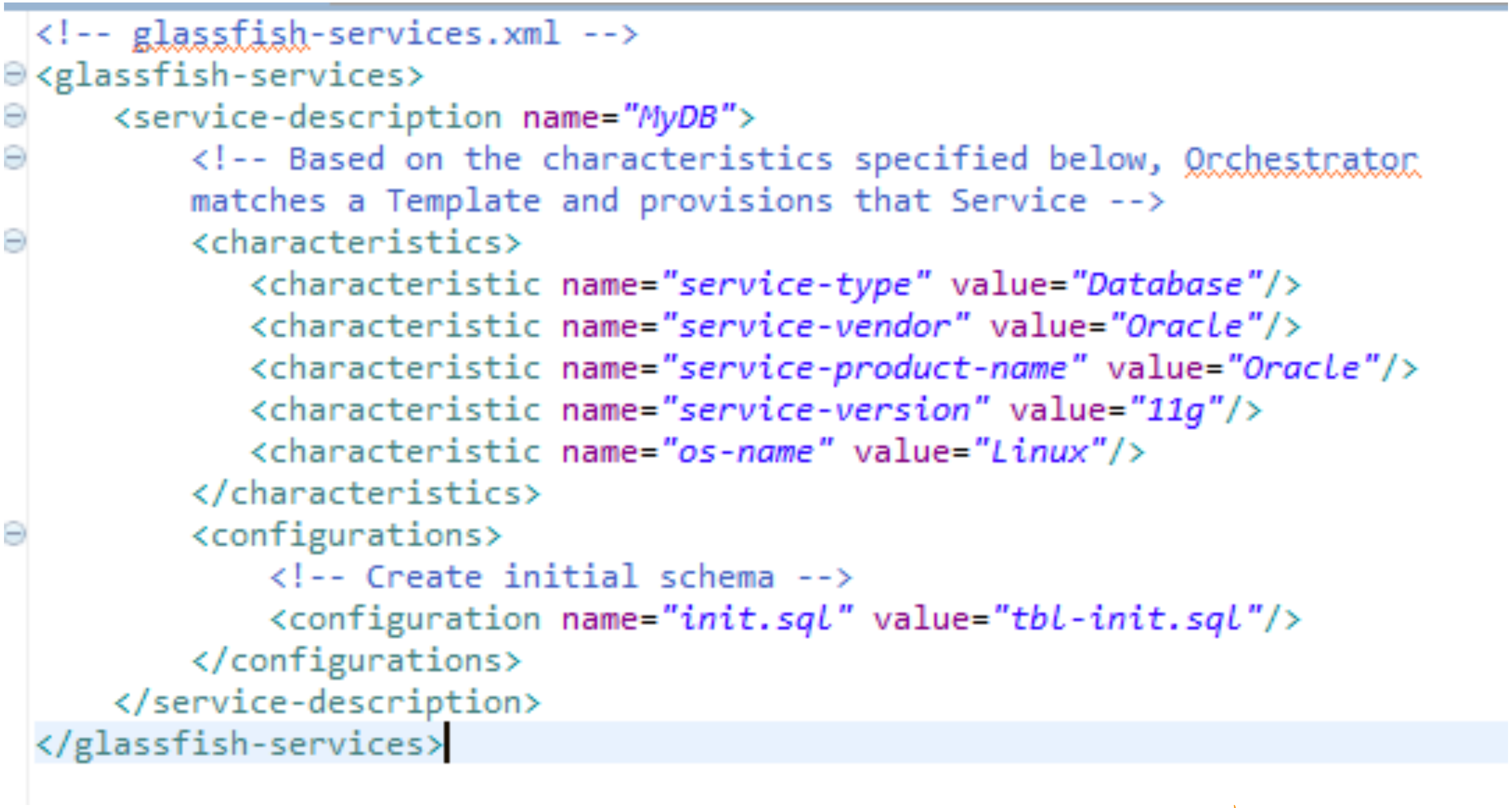
Service Description for a Java EE Service - cluster min and max size

Explicit Declaration

glassfish-resources.xml

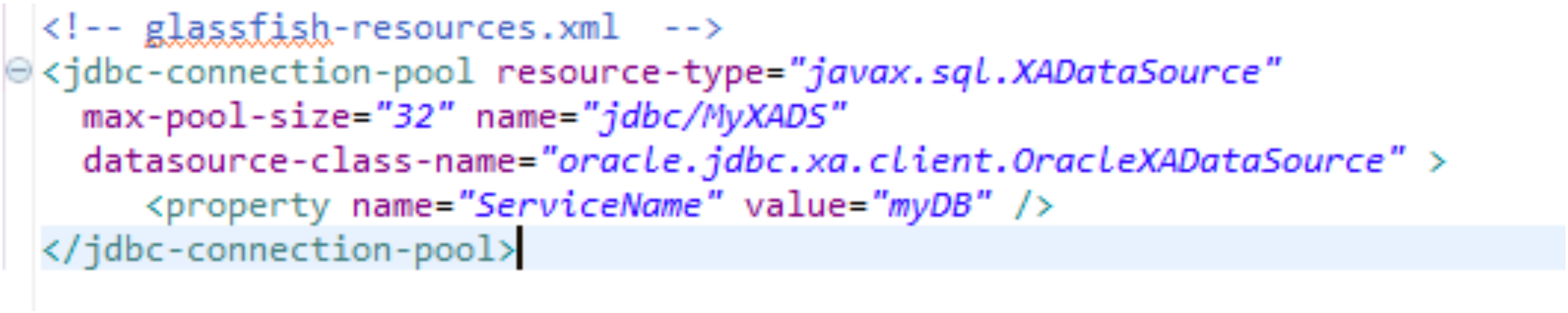
Service References for JDBC Connection Pool and Resource

Sample Service Definition



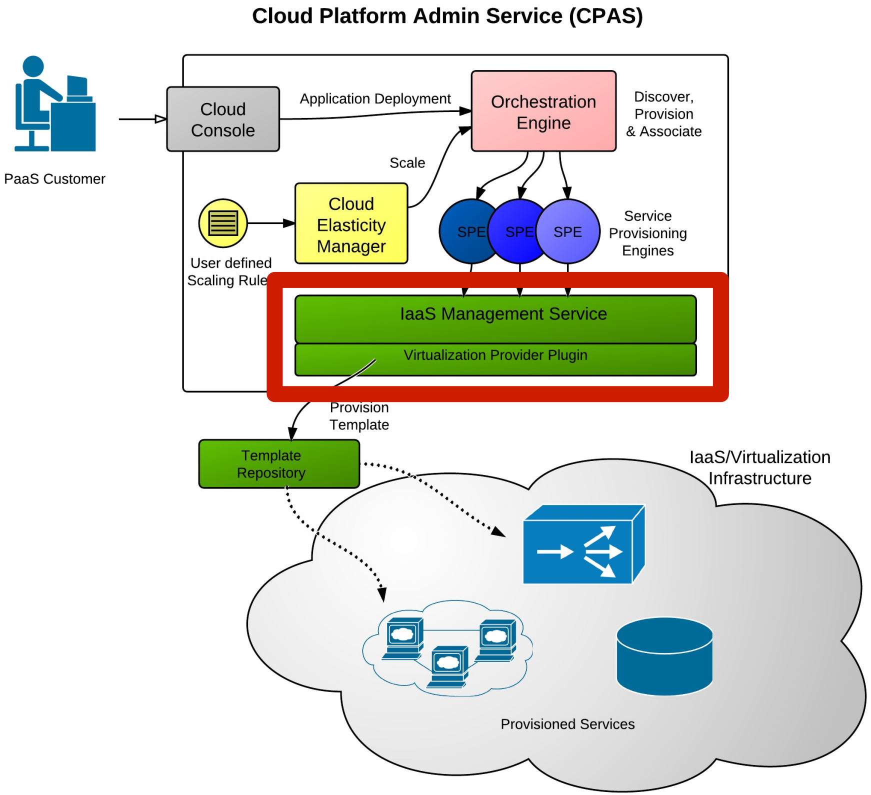
```
<!-- glassfish-services.xml -->
<glassfish-services>
    <service-description name="MyDB">
        <!-- Based on the characteristics specified below, Orchestrator
        matches a Template and provisions that Service -->
        <characteristics>
           <characteristic name="service-type" value="Database"/>
           <characteristic name="service-vendor" value="Oracle"/>
           <characteristic name="service-product-name" value="Oracle"/>
           <characteristic name="service-version" value="11g"/>
           <characteristic name="os-name" value="Linux"/>
        </characteristics>
        <configurations>
            <!-- Create initial schema -->
            <configuration name="init.sql" value="tbl-init.sql"/>
        </configurations>
    </service-description>
</glassfish-services>
```

Sample Service Reference



```
<!-- glassfish-resources.xml  -->
<jdbc-connection-pool resource-type="javax.sql.XADataSource"
  max-pool-size="32" name="jdbc/MyXADS"
  datasource-class-name="oracle.jdbc.xa.client.OracleXADataSource" >
     <property name="ServiceName" value="myDB" />
</jdbc-connection-pool>
```

GlassFish PaaS Runtime Architecture

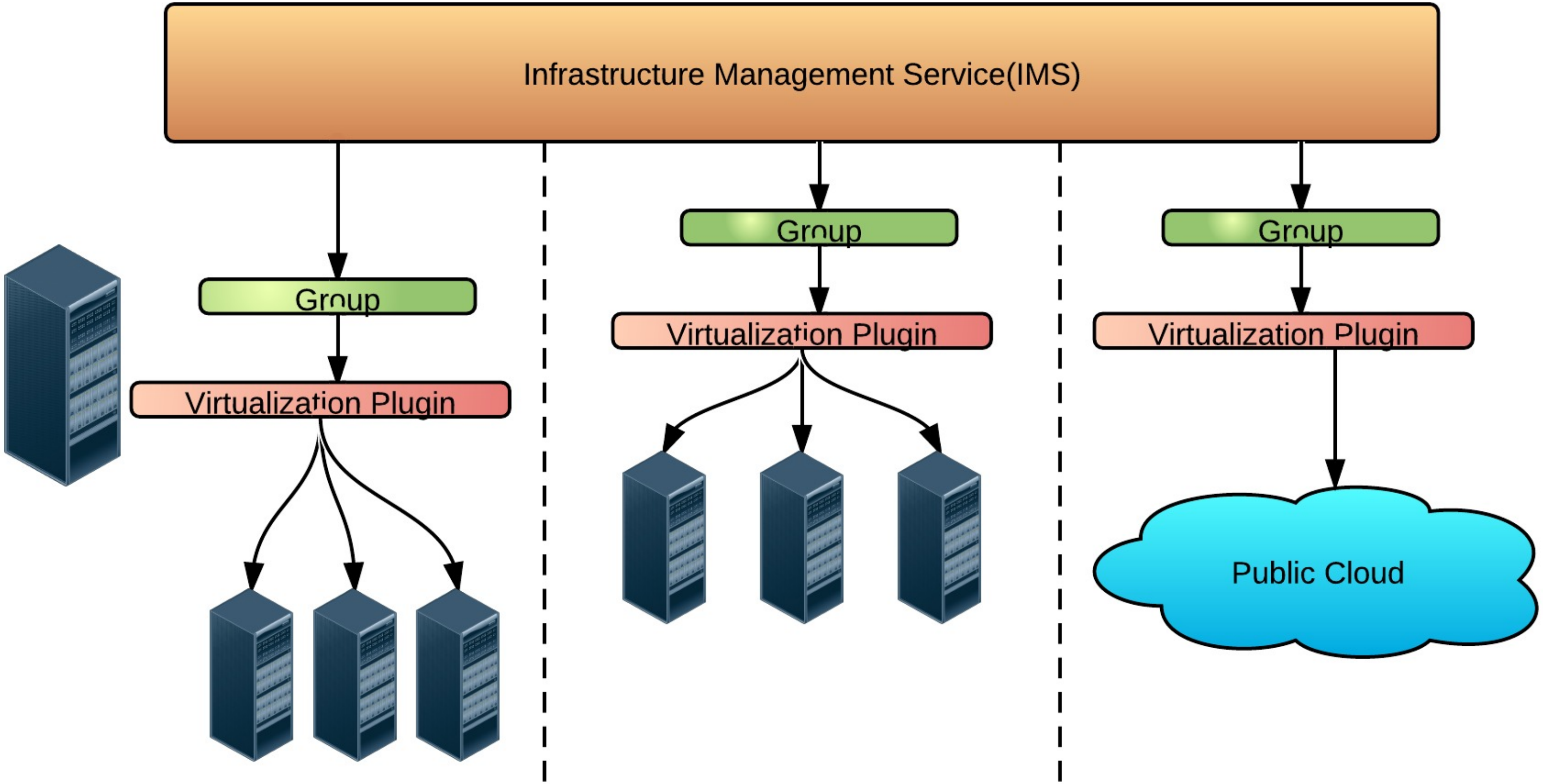


IaaS Management Service (IMS)

One-liner

Provide common management interface across different virtualization technologies

IMS


Infrastructure Management Service(IMS)
Group
Virtualization Plugin
Group
Virtualization Plugin
Group
Virtualization Plugin
Public Cloud

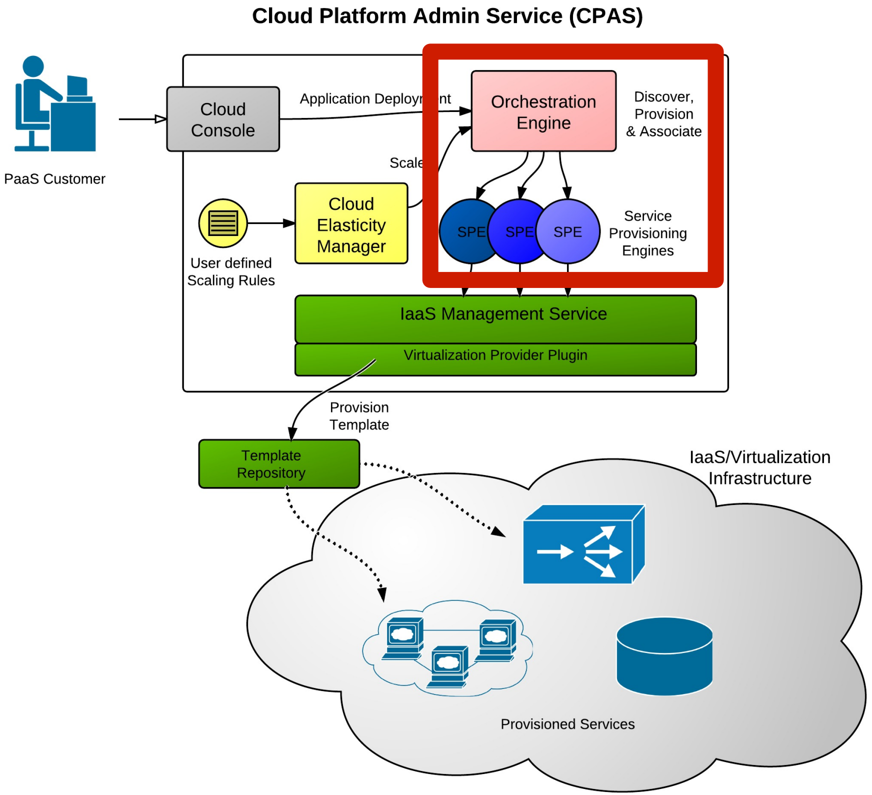
IMS functionalities

- Support virtualization definitions
- Isolates from low level Virtual Machine allocation/interface
  - *Integrates with native solutions through Plug-in/SPI mechanism.*
- Template management
- ServerPool / Hardware management (depending on the virtualization technology).

Templates

- A virtual-machine disk
  - can be duplicated
  - used to instantiate a virtual-machine.
- Virtualization Specific
- Provides 1 to many service types (usually one).
- Template are customized during the first startup
  - DAS location
  - Template parameters like instance name
  - Customization mechanism is virtualization specific

GlassFish PaaS Runtime Architecture

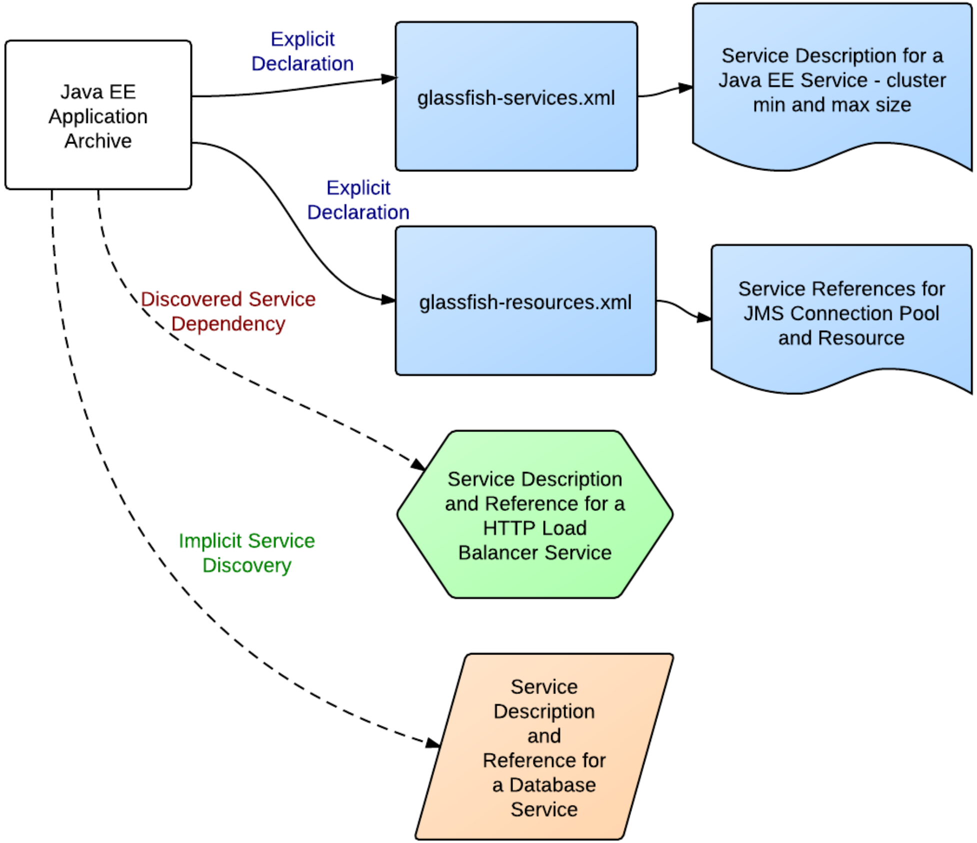


Service Orchestration

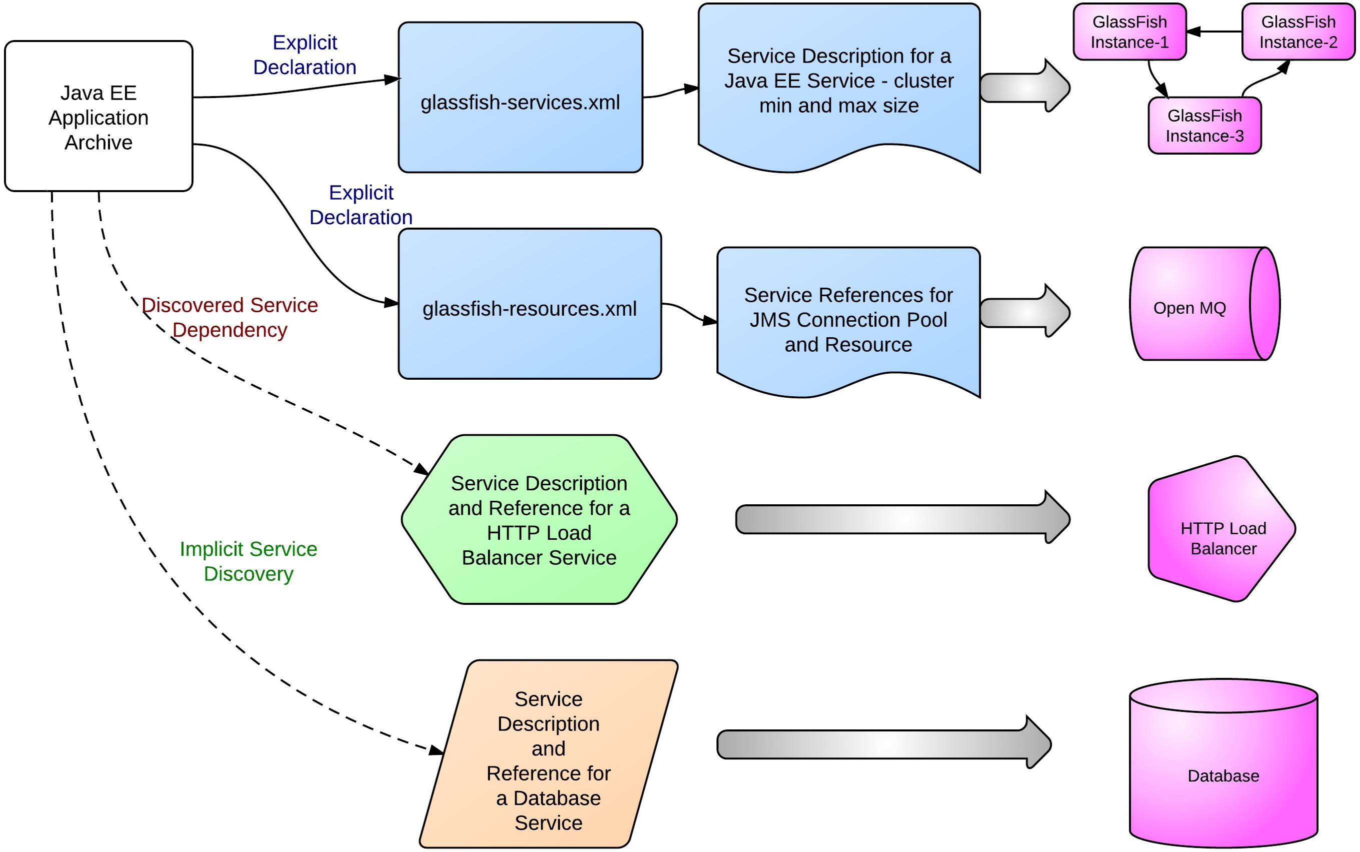
One-liner

Enable single-click deployment of a PaaS application through automatic service dependency discovery, service provisioning and service association

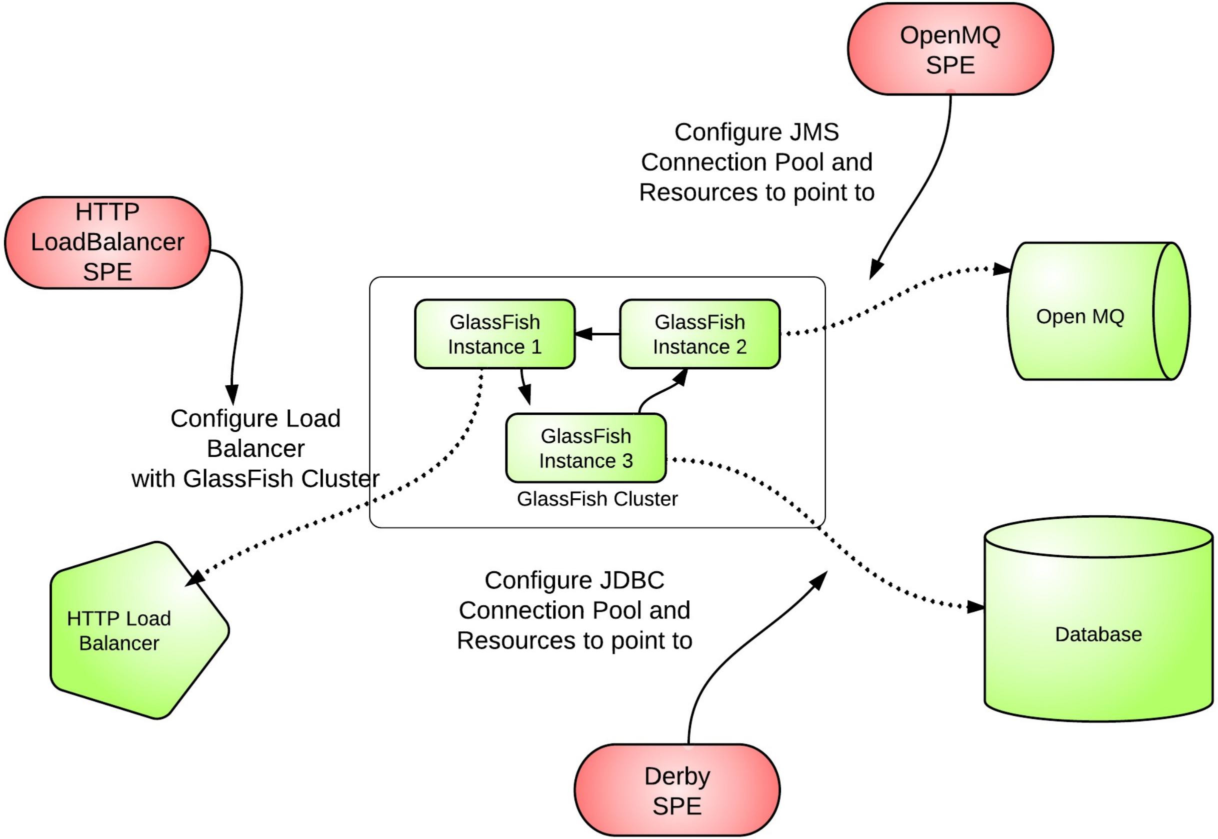
Service Dependency Discovery


Java EE Application Archive
Explicit Declaration
glassfish-services.xml
Service Description for a Java EE Service - cluster min and max size
Explicit Declaration
glassfish-resources.xml
Service References for JMS Connection Pool and Resource
Discovered Service Dependency
Service Description and Reference for a HTTP Load Balancer Service
Implicit Service Discovery
Service Description and Reference for a Database Service

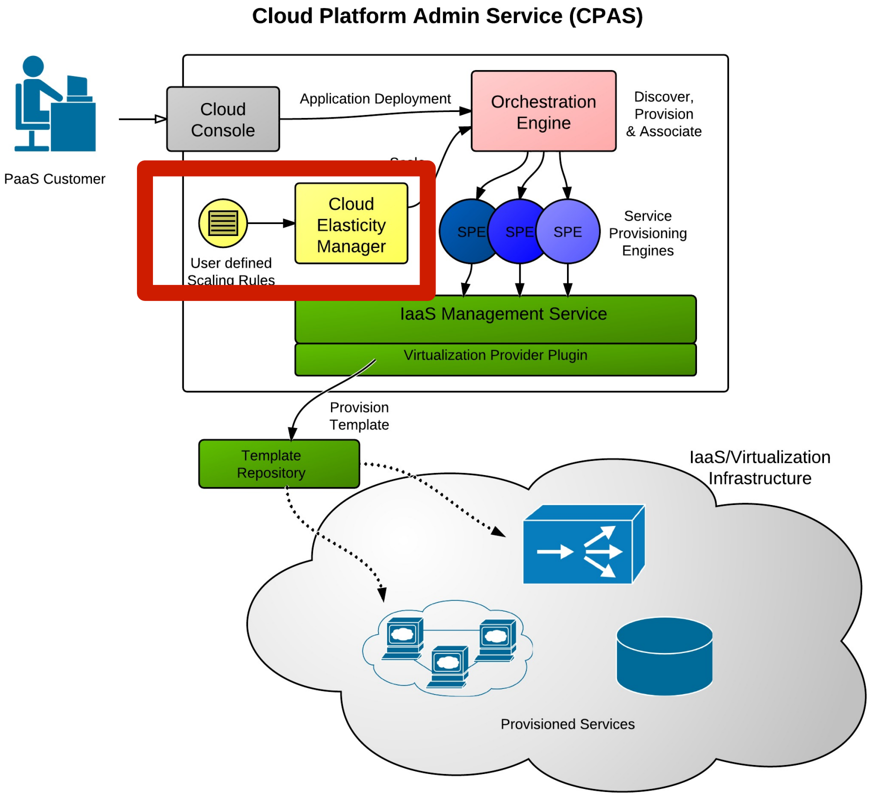
Service Provisioning


Java EE Application Archive
Explicit Declaration
glassfish-services.xml
Service Description for a Java EE Service - cluster min and max size
GlassFish Instance-1
GlassFish Instance-2
GlassFish Instance-3
Explicit Declaration
glassfish-resources.xml
Service References for JMS Connection Pool and Resource
Open MQ
Discovered Service Dependency
Service Description and Reference for a HTTP Load Balancer Service
HTTP Load Balancer
Implicit Service Discovery
Service Description and Reference for a Database Service
Database

Service Association


OpenMQ SPE
Configure JMS Connection Pool and Resources to point to
HTTP LoadBalancer SPE
Open MQ
GlassFish Instance 1
GlassFish Instance 2
GlassFish Instance 3
GlassFish Cluster
Configure Load Balancer with GlassFish Cluster
HTTP Load Balancer
Configure JDBC Connection Pool and Resources to point to
Database
Derby SPE

GlassFish PaaS Runtime Architecture



Auto-Scaling

One-liner

The ability of a system to automatically adapt to volume of traffic without impacting throughput and availability

What we need

- To determine the health of Services in the System
  - Number of metrics can be used
  - An arbitrary / complex condition need to be evaluated
  - Arbitrary actions can be taken

- A framework that allows
  - Ability to define new Metric sources easily (Extensible)
  - Ability to express complex queries easily
  - Ability to create new Actions easily

Determining State of Services

- Monitor System Resources
  - CPU
    - Used and idle CPU times
  - Memory
    - Process memory: Allocated, Resident etc.
    - JVM memory: Used, Committed and Max memory
  - Disk
    - Reads, Writes per seconds
    - Bytes read, written etc.

Determining State of Services (Contd.)

– Monitor Application Related Objects

- HTTP Sessions created / destroyed per second
- Number of HTTP requests that arrived
- Connection Pools: Number of connections acquired / released etc.
- Database Queries executed
- Transaction status: Number of commits / rollbacks
- Response time of a specific URL

– ***Many, many, many other metrics…***

We have metrics. Now what?

- Use relevant metrics to determine health of the services
  - Such as Java EE Clustered Instances, Clustered DB
- Use Trends
  - Average memory usage above 60% for last 10 min
  - Avg Response times of 90% of requests in last 10 min below 5ms
- Use Combinations
  - Are both CPU and Memory usages high ?
  - Is CPU usage of Java EE Cluster and number of queries executed on Database within certain limits ?

What Actions can you take?

- Possible Actions
  - Send email to admin
  - Log some info
  - Scale up / Scale down to meet the load
  - Send a (JMX) notification
  - *And many, many, many more…*

Call to Action

- Java EE 7 Expert Group Project
  - http://javaee-spec.java.net
- Java EE 7 Reference Implementation
  - http://glassfish.org
- The Aquarium
  - http://blogs.oracle.com/theaquarium

ORACLE

PaaSing a Java EE Application

Arun Gupta, Java EE & GlassFish Guy
blogs.oracle.com/arungupta, @arungupta,
arun.p.gupta@oracle.com

MOVING JAVA FORWARD

Java™

ORACLE®