JUDCon

JBoss Users & Developers Conference

London:2011

Elastic SOA on the Cloud

Steve Millidge (C2B2)

About Me

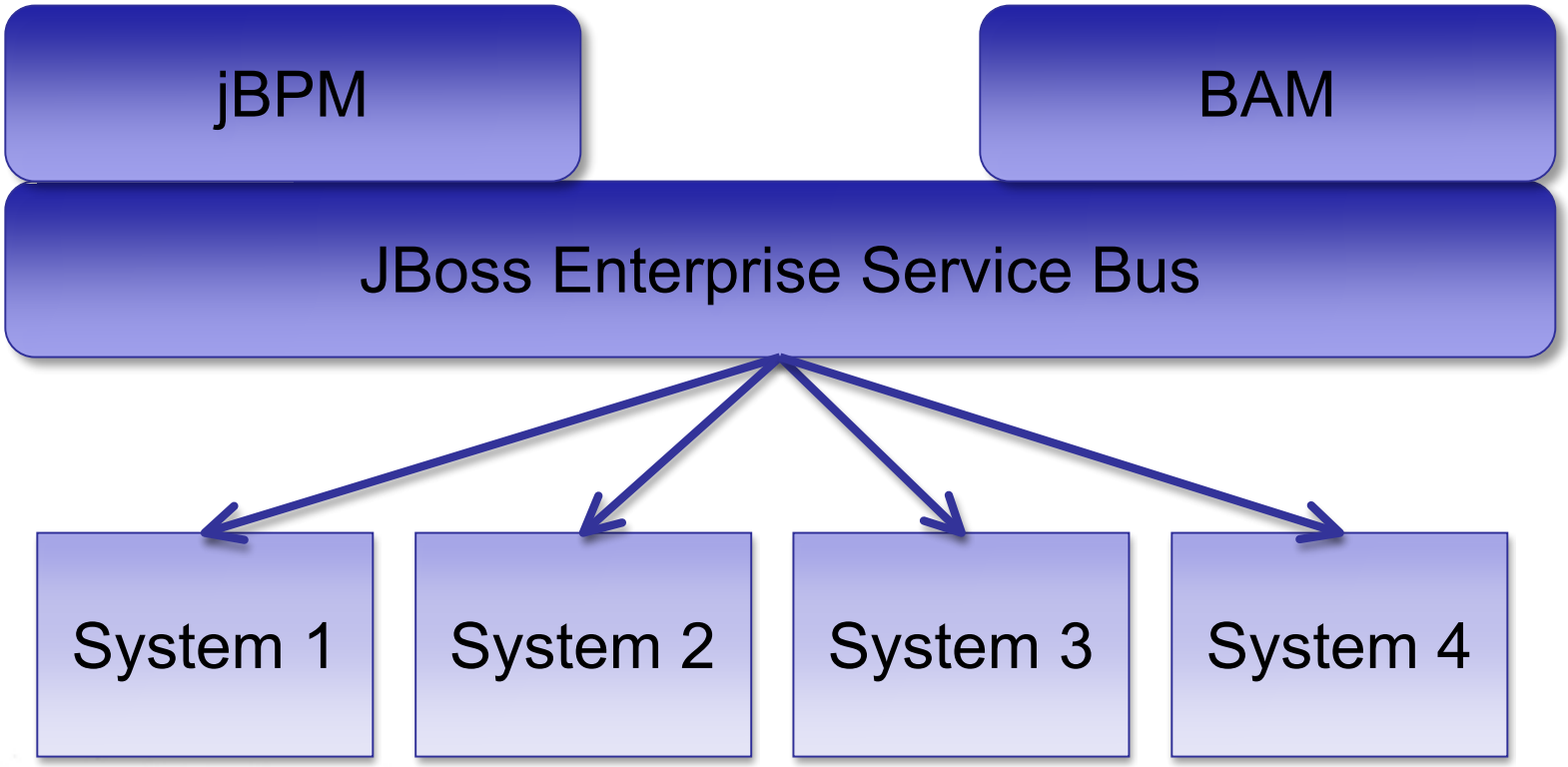
- Founder of C2B2
  - Red Hat Partner
  - Deep Expertise in Middleware
  - Non-functional Experts
- 20 Years Middleware Expertise
- 15 years Field Consultancy
- Organiser of JBUG London
  - http://www.meetup.com/JBoss-User-Group/

Agenda

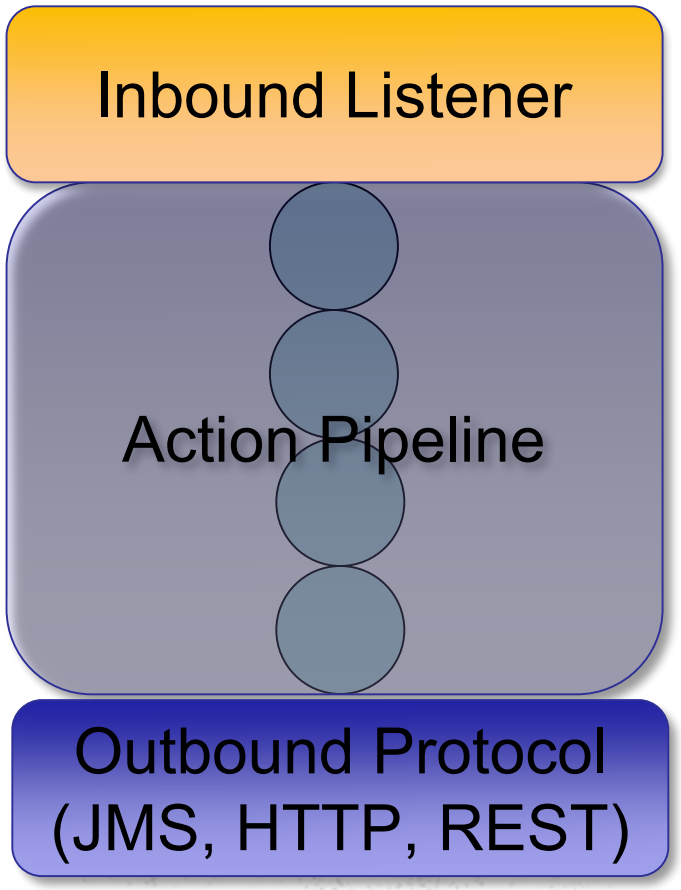
- SOA Reprise
- Cloud Reprise
- Why SOA on the Cloud?
- JON Overview
- Deploying ESB for Elasticity
- Service Deployment Options

SOA Reprise

Typical SOA Architecture



Typical ESB Service



Components

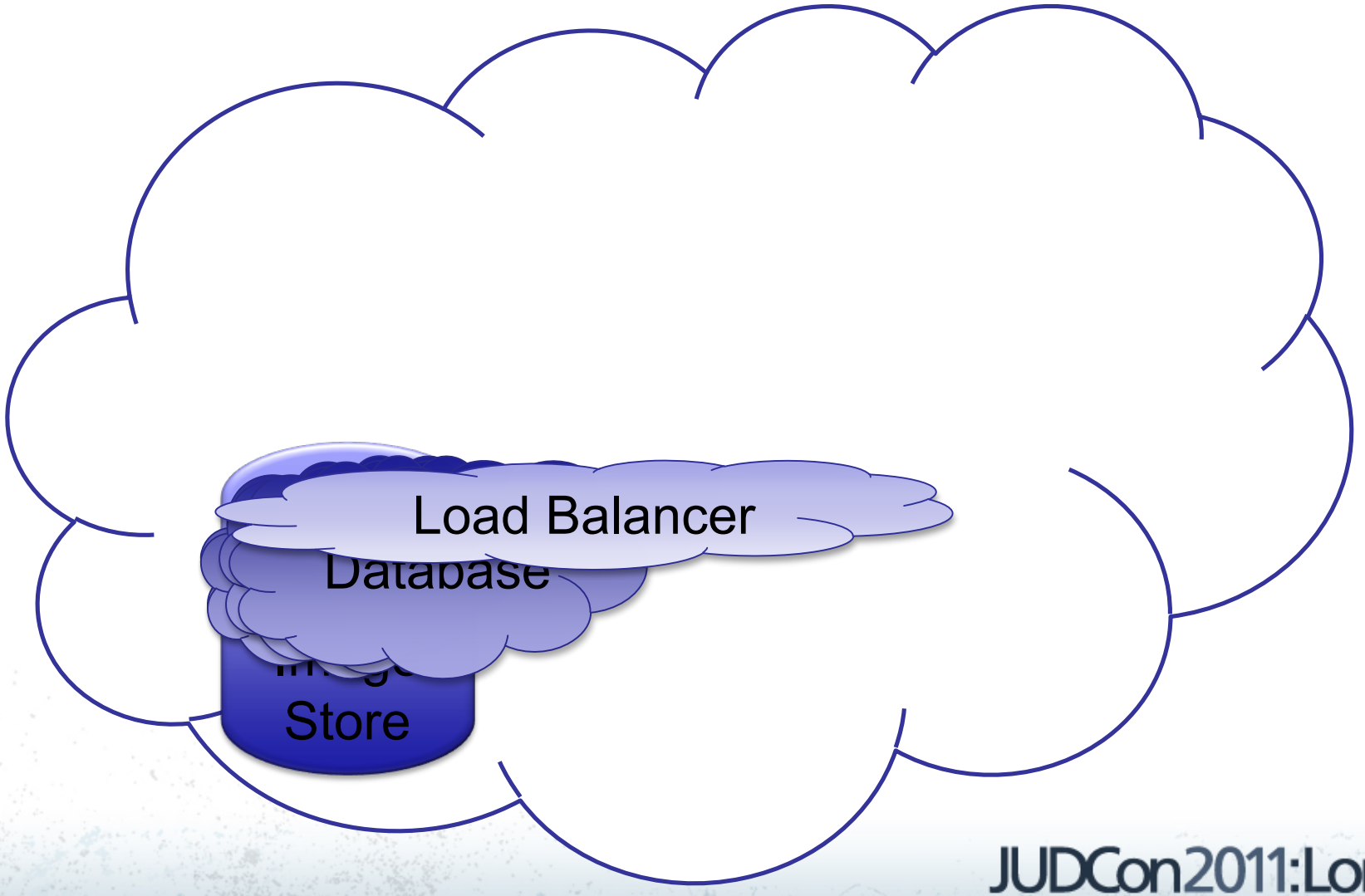
- Inbound Listener
  - JMS, File, FTP, Web Service, HTTP(s) …
- Action Pipeline
  - Transformation, Routing, Process Coordination, Auditing, Aggregation
- Outbound Protocol
  - Calls System Business Logic
  - EJB, Web Service …

Service Definition

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jbossesb
    xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/sche
    mas/xml/jbossesb-1.0.1.xsd" invmScope="GLOBAL">

<services>
   <service category="Retail" name="ShoeStore" description="Acme Shoe
    Store Service">
      <actions>
         <action name="println"
    class="org.jboss.soa.esb.actions.SystemPrintln" />
      </actions>
    </service>
</services>

</jbossesb>
```

Cloud Reprise

Infrastructure As A Service



Amazon EC2

EC2 Core Features

- AMI
- Instance
  - Spot, On-Demand, Dedicated
- Volumes
  - Snapshots
- Security Groups
- Load Balancers
- Elastic IPs

Peripheral Services

- Cloud Front
- Cloud Watch
- Cloud Formation
- RDS
- VPC
- Elastic Beanstalk

Starting an Instance From Java

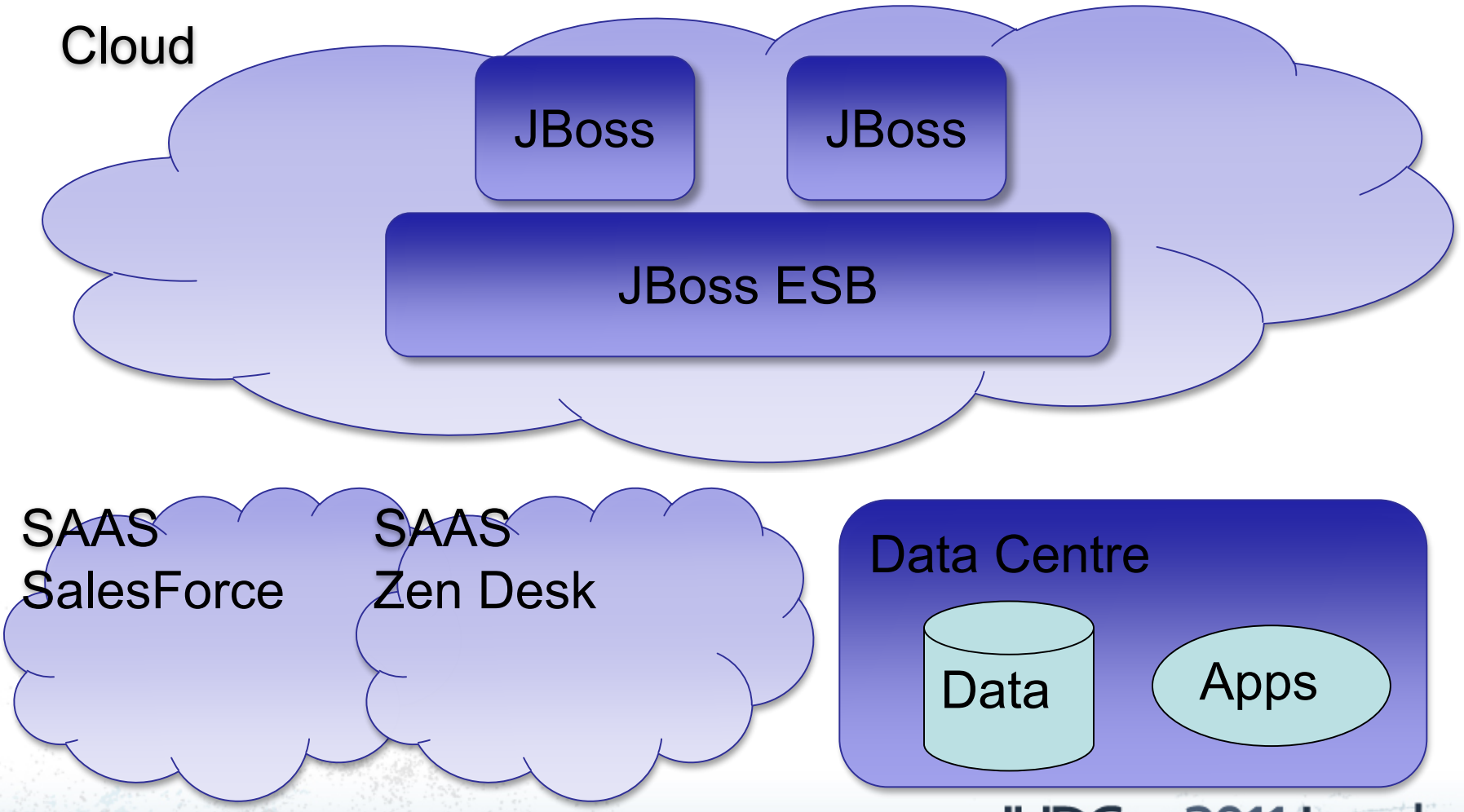
```
String imageID = “ami-xxxxxx”;
String minInstances = 3;
String maxInstances = 3;

RunInstancesRequest request = new
    RunInstancesRequest(imageID,minInstances, maxInstances);

AmazonEC2Client client = new AmazonEC2Client(new AWSCredentials(…));

RunInstancesResult result = client.runInstances(request);
```

Amazon Instance Types

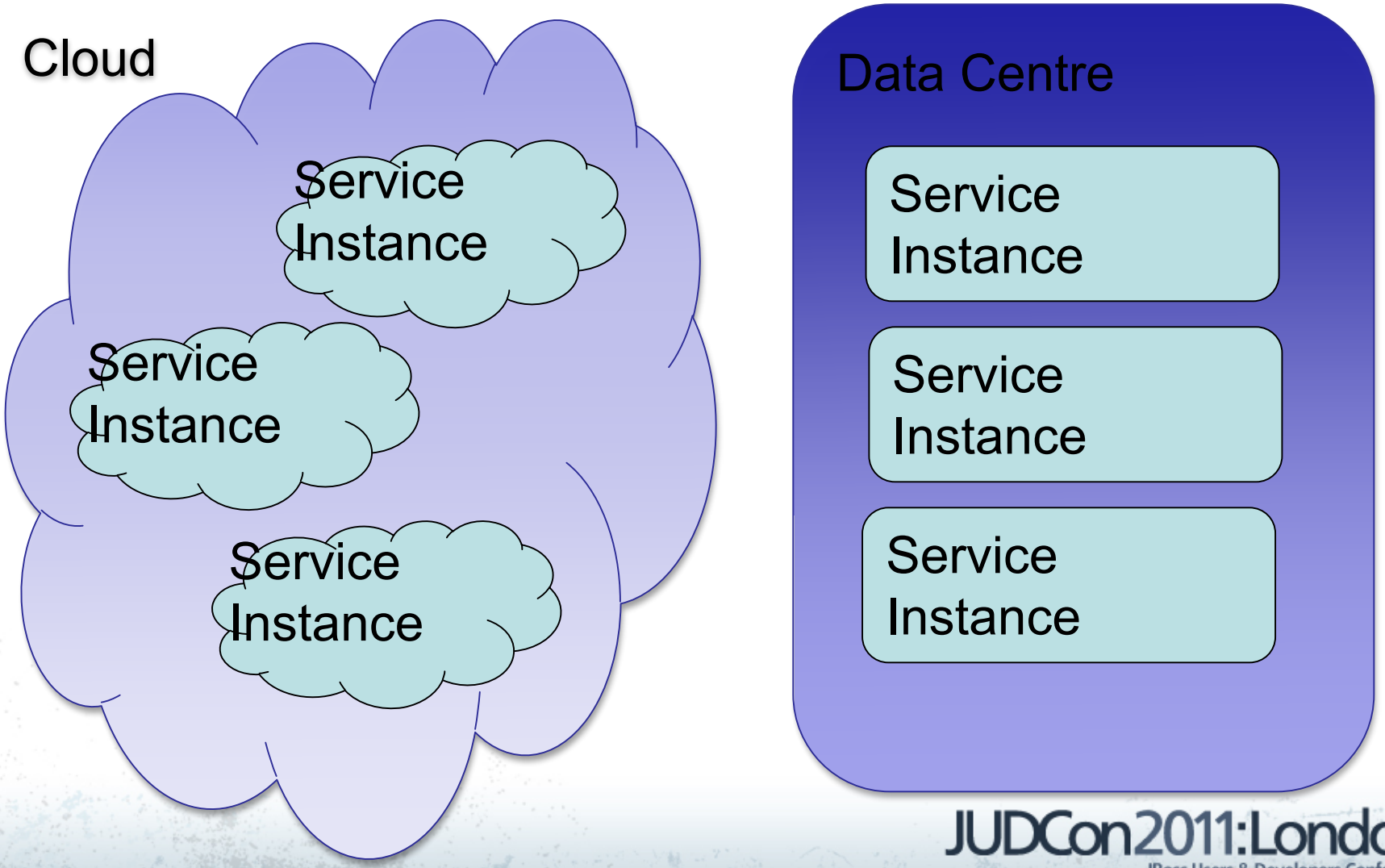
Instance Type	Memory (GB)	CPU (ECU)	Cost per Hour
Standard Small	1.7	1	$0.095
Standard Large	7.5	4	$0.38
Standard XL	15	8	$0.76
High Mem XL	17	6.5	$0.57
High Mem DXL	34.2	13	$1.14
High Mem QXL	68.4	26	$2.28
High CPU Med	1.7	5	$0.19
High CPU XL	7	20	$0.76
Cluster	23	33.5	$1.60
Micro	.613	2 (burstable)	$0.025

Why SOA on the Cloud?

Cloud Integration



Cloud Bursting

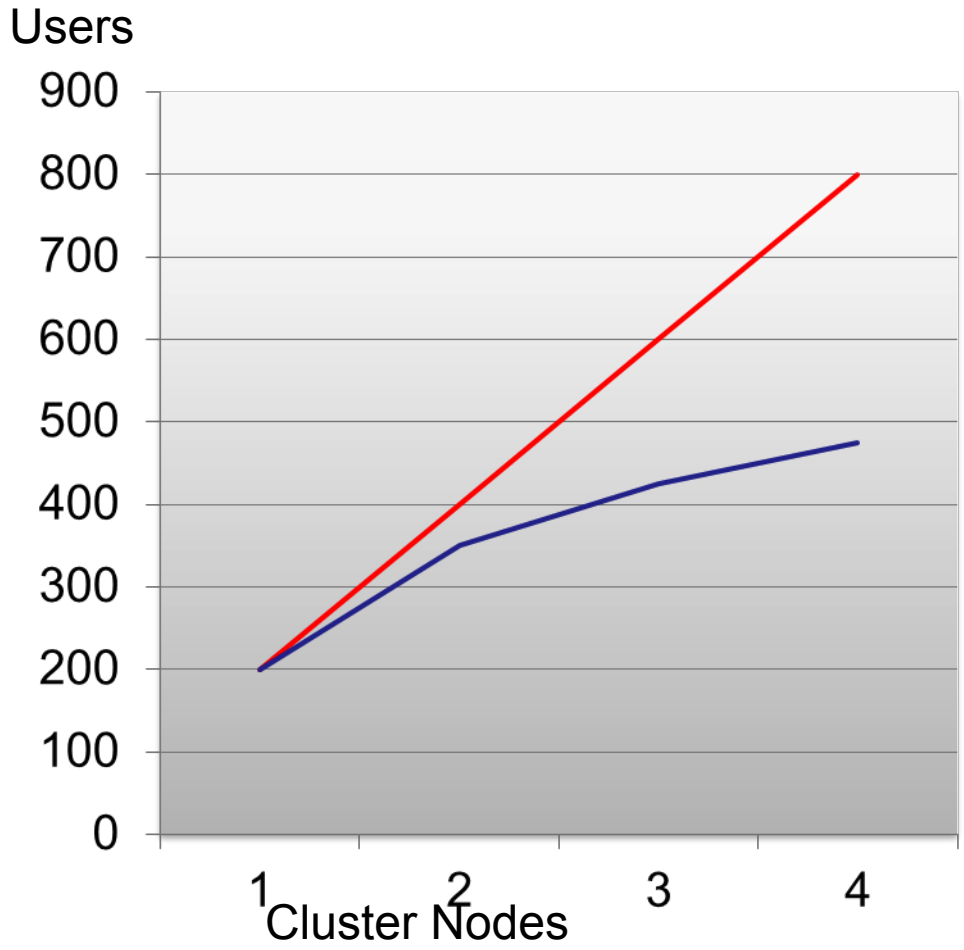


Challenges to Elasticity

- Networking
  - IP Addresses, Firewalls
- Service Discovery
  - New Capacity needs to be utilised
- Clustering
  - New servers must cluster
- Monitoring
  - Monitoring Tools must discover new servers
- Application Architecture
  - Elastic Aware

How to Do Elasticity

- Know Your Scalability
- Know What Limits Scalability
- Measure the Limiting Factors
- Alert on Metrics
- Add Additional Capacity



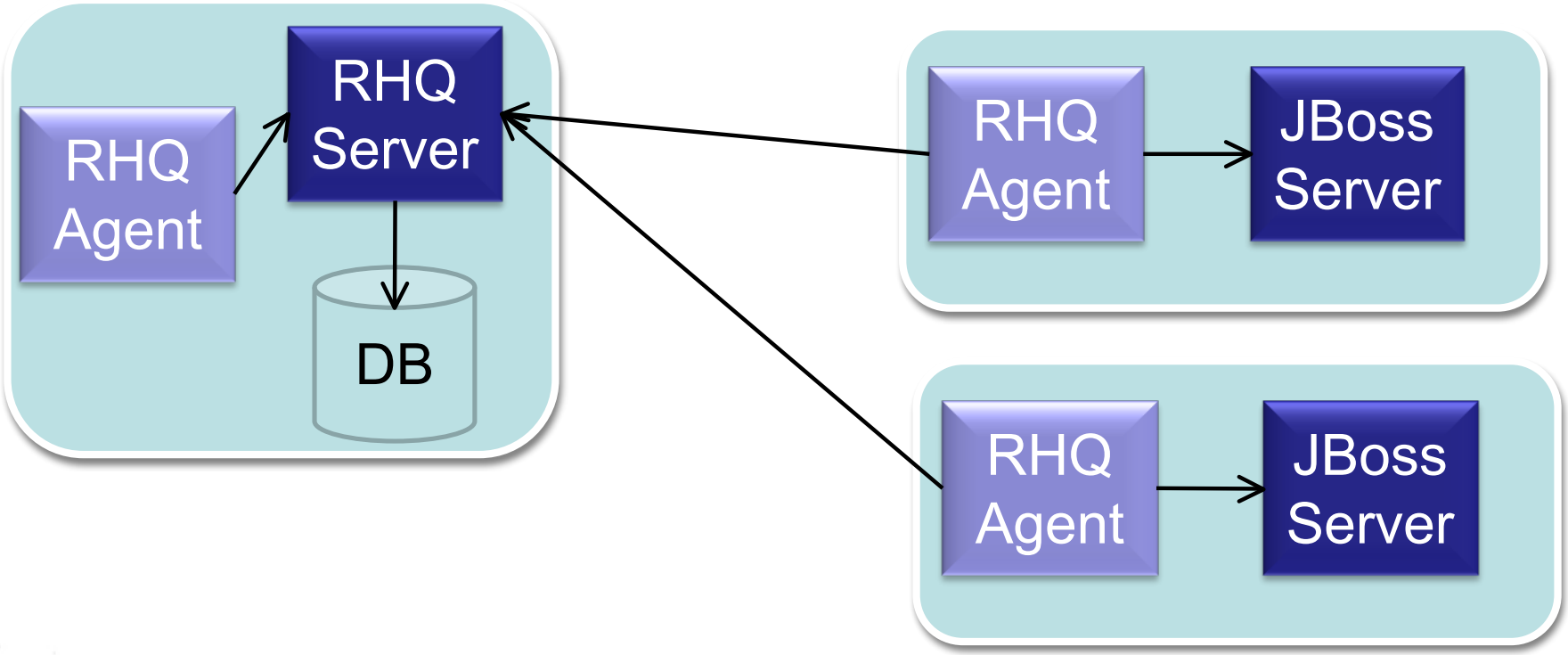
JBoss Operations Network (RHQ)

What is RHQ

The RHQ project is a systems management suite that provides extensible and integrated systems management for multiple products and platforms across a set of core features such as

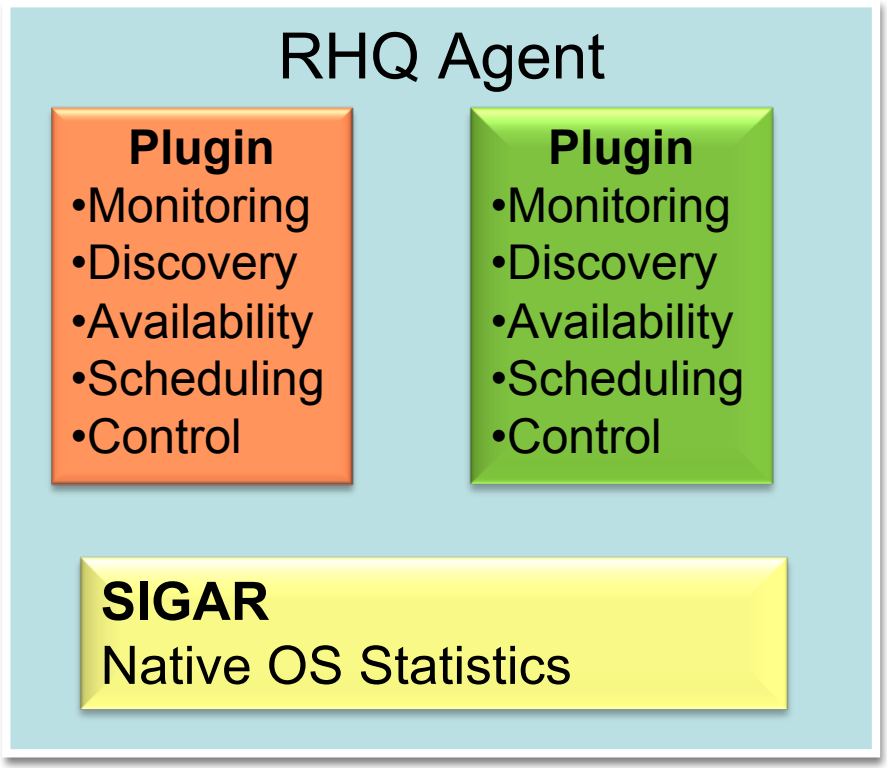
- monitoring and graphing of values
- alerting on error conditions
- remote configuration of managed resources
- remote operation execution

RHQ Architecture



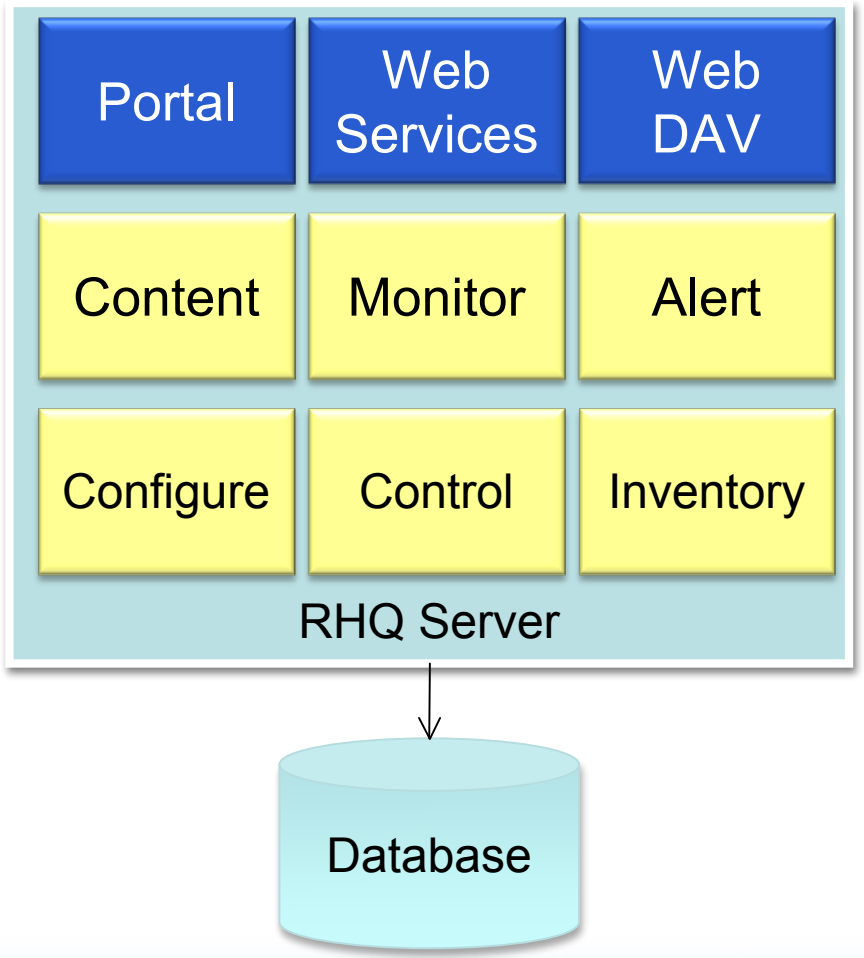
RHQ Agent

- Installed on each machine
- Discovers machine Inventory
- Can access local processes
- Gathers native metrics from the OS.
- Stores metrics during server outage



RHQ Server

- Stores Data Received From Agents
- Updates Agents
- Sends Commands to Agents
- Raises Alerts
- Provides Portal
- Provides Web Services and WebDAV interface
- Provides CLI Interface
- Provides Security



RHQ and JBoss ESB

- Key Service Metrics
  - Message Count (Success)
  - Message Count (Failed)
  - Processed Bytes
- Key Action Metrics
  - Processing Time
  - Message Counts (Average Per Minute)

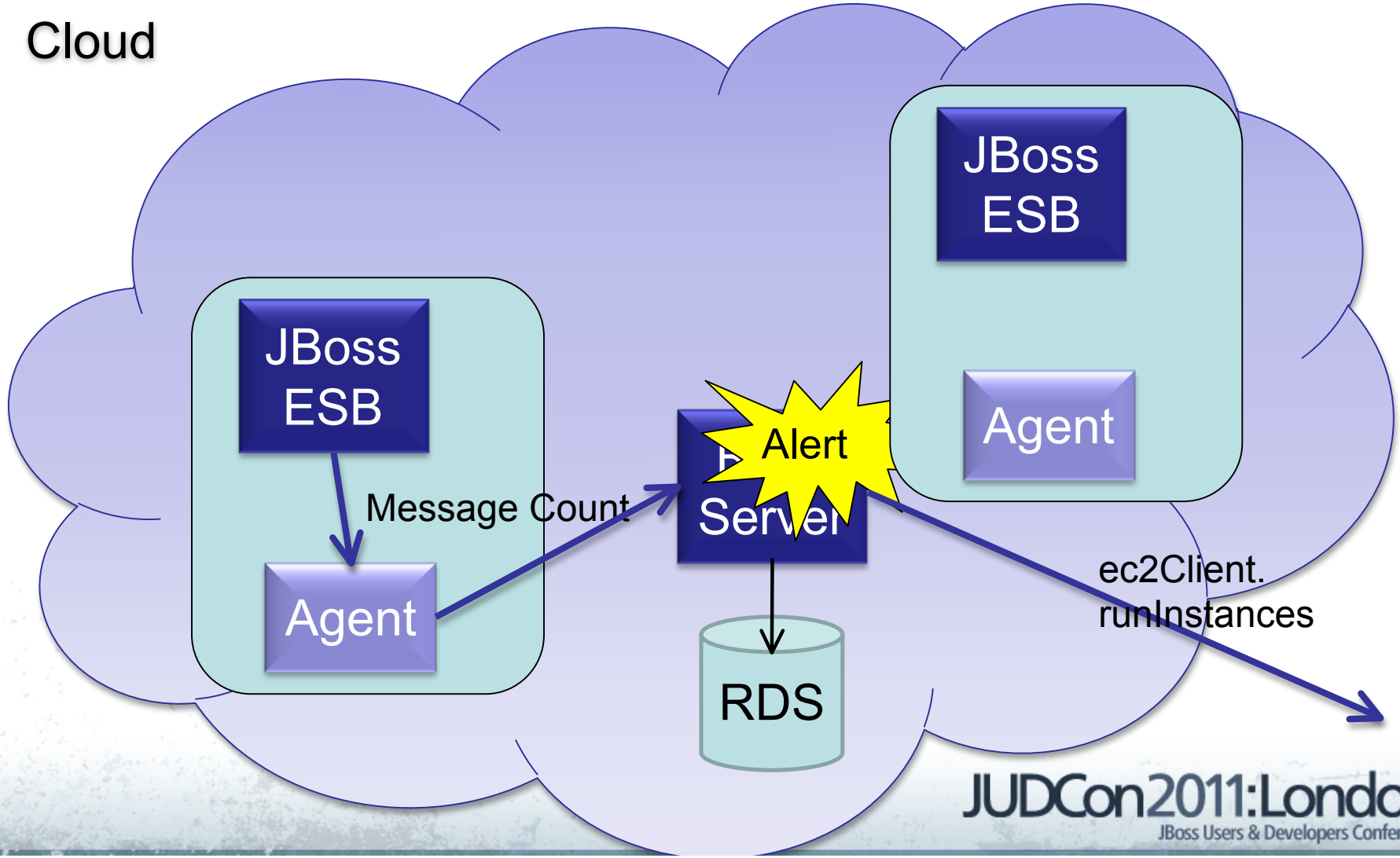
Key JBoss Metrics

- Key JBoss Metrics
  - JVM Free Heap
  - GC Time
  - Servlet Response Time/ Response Rates
  - Connection Pool Utilisation
- Key System Metrics
  - CPU Usage
  - Network Utilisation

Alerting in RHQ

- Alerts can be set on any metric
- Alerts can be set on Availability
- Alerts are sent by the Server from filtering the metric stream
- Alert Notification is configurable and Extensible
- Alerts can trigger operations on a resource
- Alerts can trigger Script Actions

Alerting and Elasticity



Deploying ESB For Elasticity

Invoking a Service

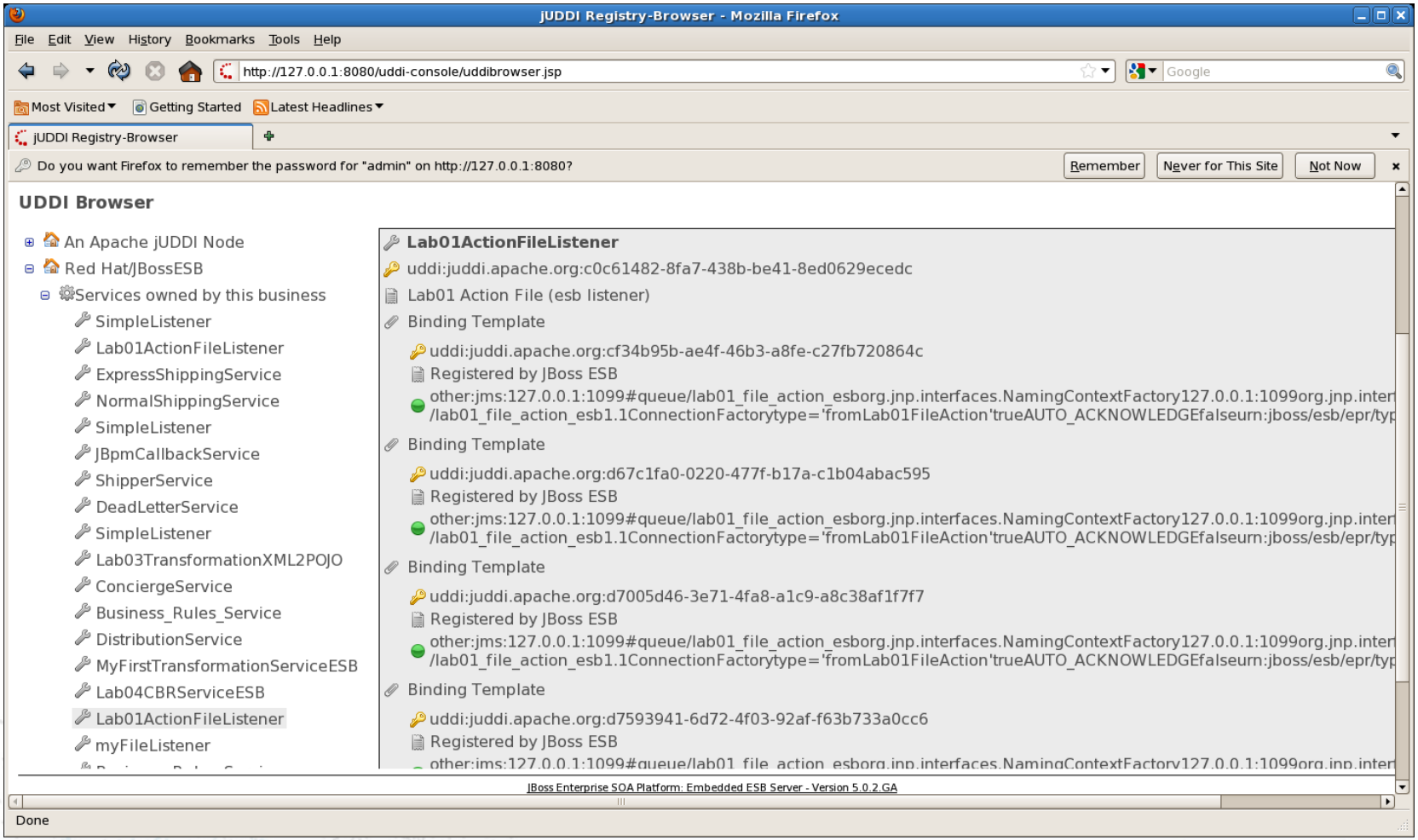
Service Invocation

```
ServiceInvoker invoker = new ServiceInvoker(“Retail”, “ShoeStore”);

Message message = MessageFactory.getInstance().getMessage();

message.getBody().add(“Hi there!”);

 invoker.deliverAsync(message);
```

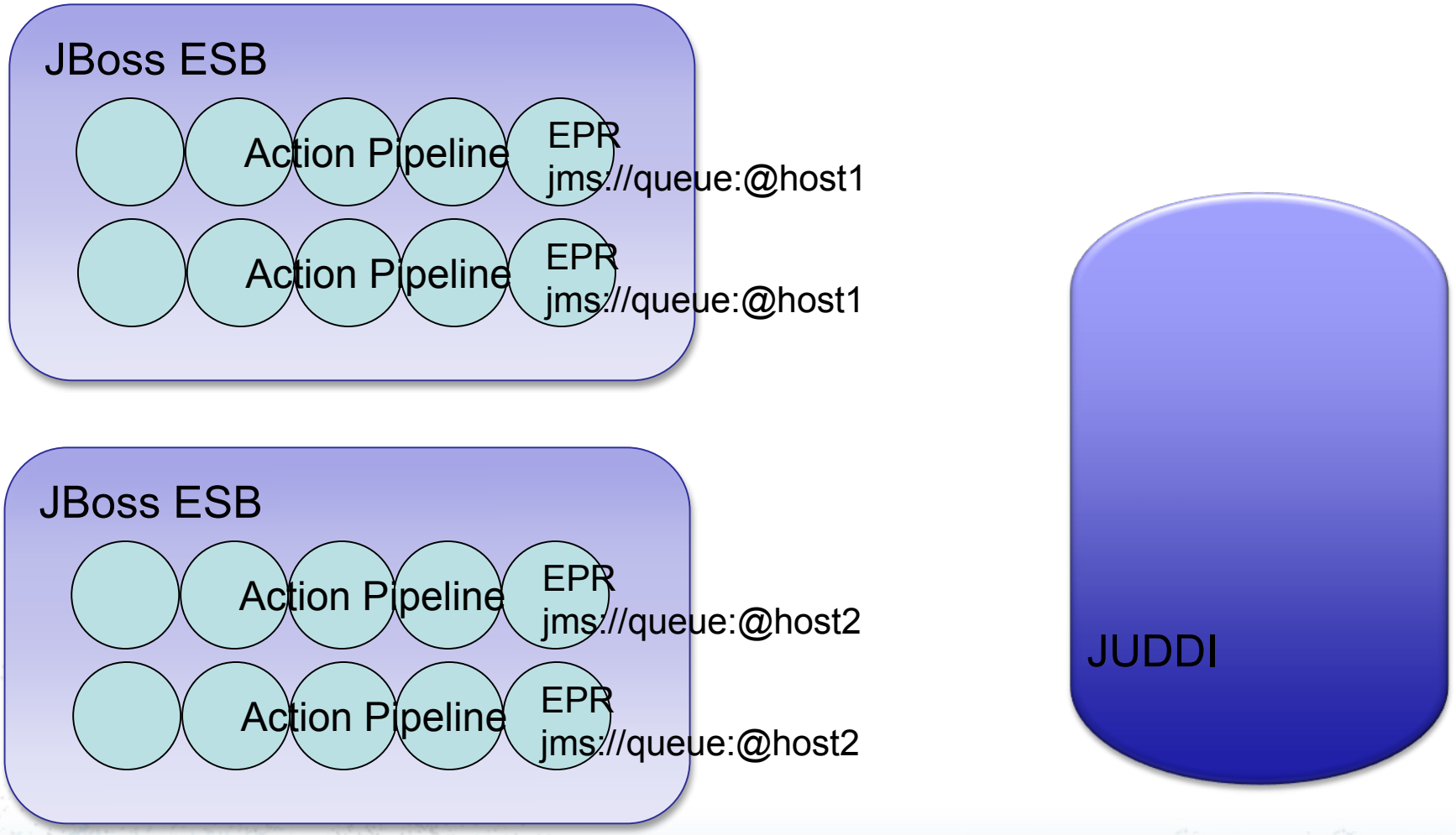
JBoss ESB Registry Service

- The registry plays a central role within JBoss ESB for the deployed services
  - The purpose of the registry is to record services, discover meta-data and classify entities into pre-defined categories
  - It may be updated dynamically, when a services first starts or statically by an external administrator
- The default configuration uses Apache jUDDI v3 as its UDDI registry
- JBoss ESB Registry is based on Apache Scout a JAXR implementation

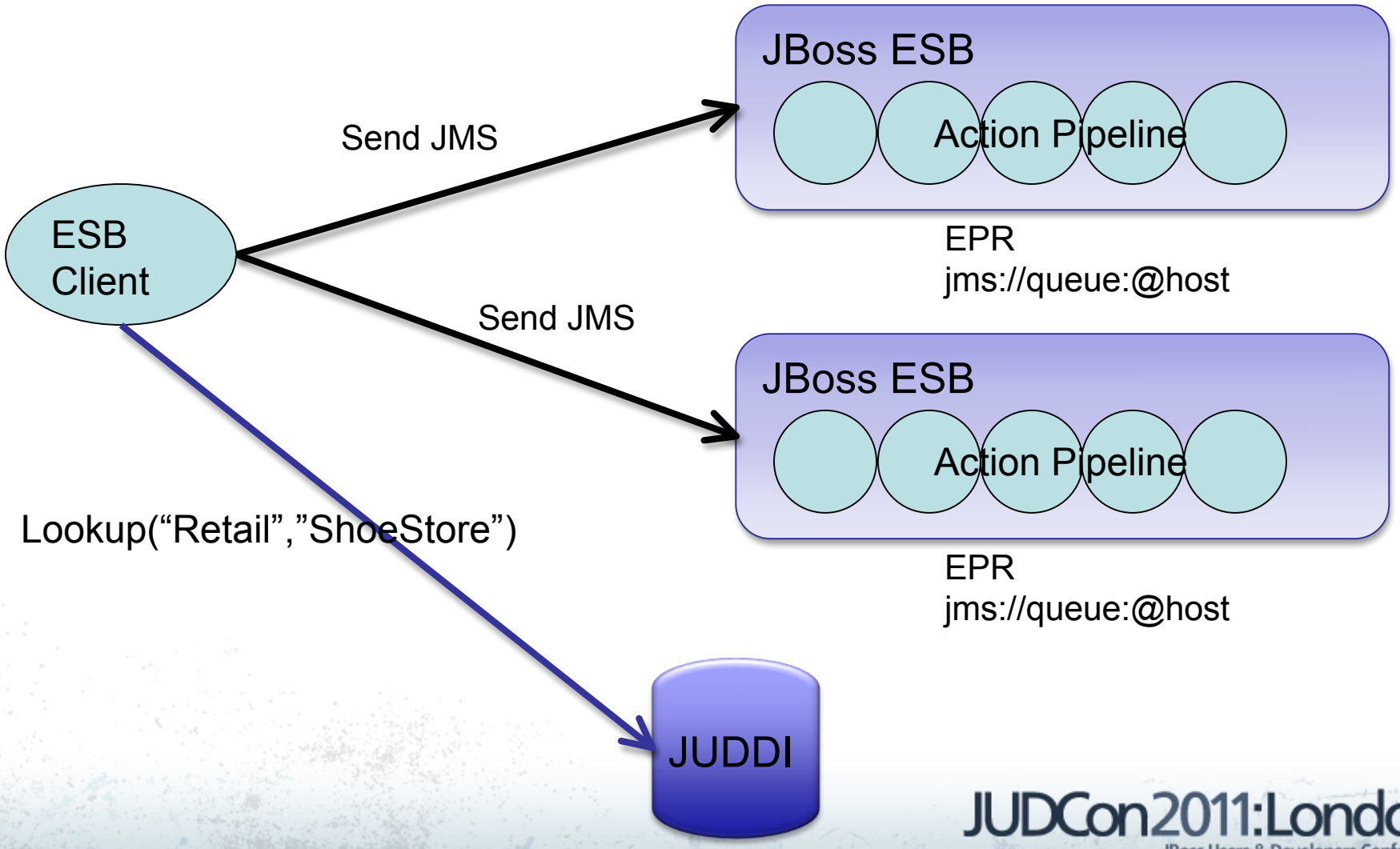
JBoss ESB UDDI Browser



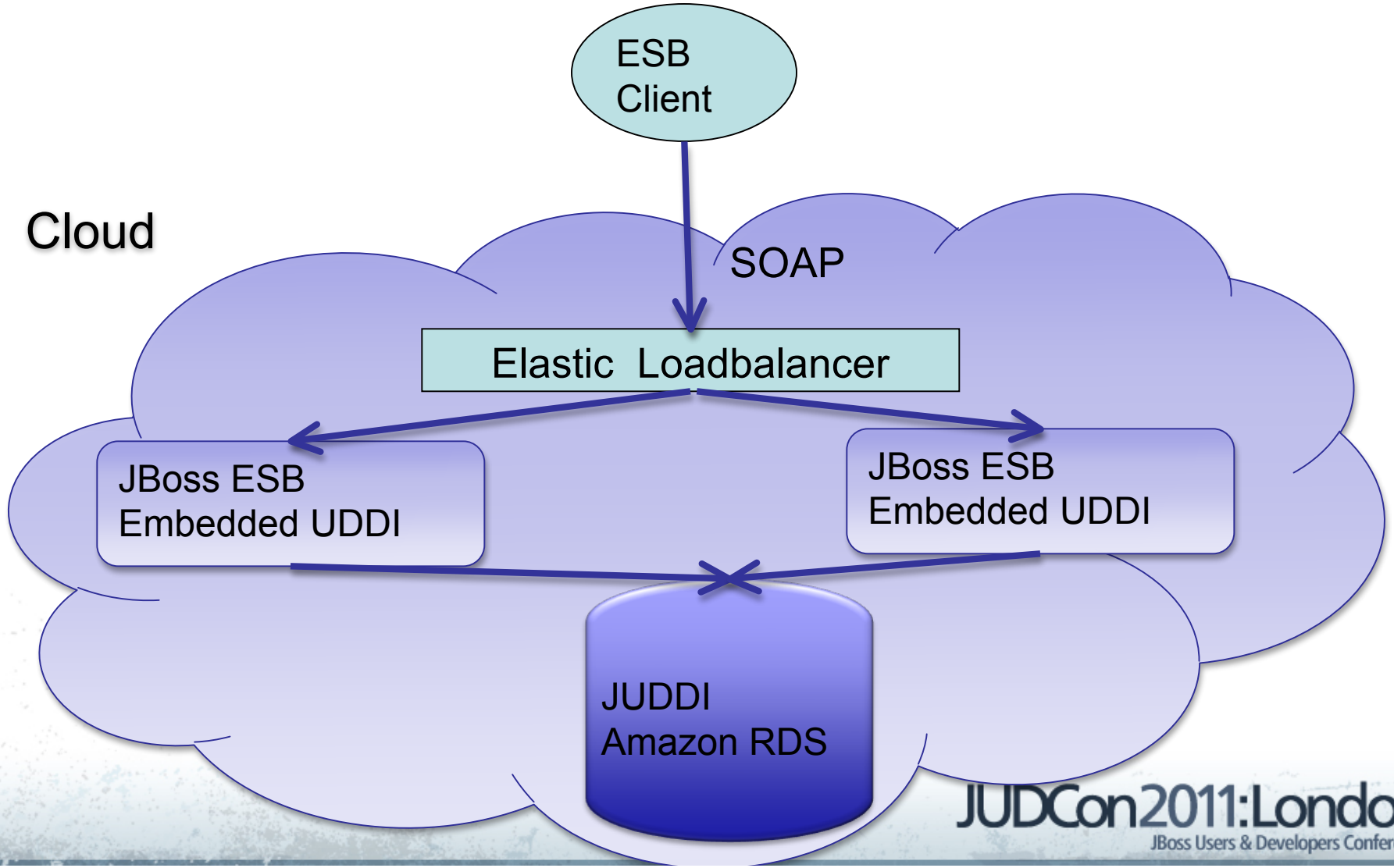
Service Registration



Service Discovery



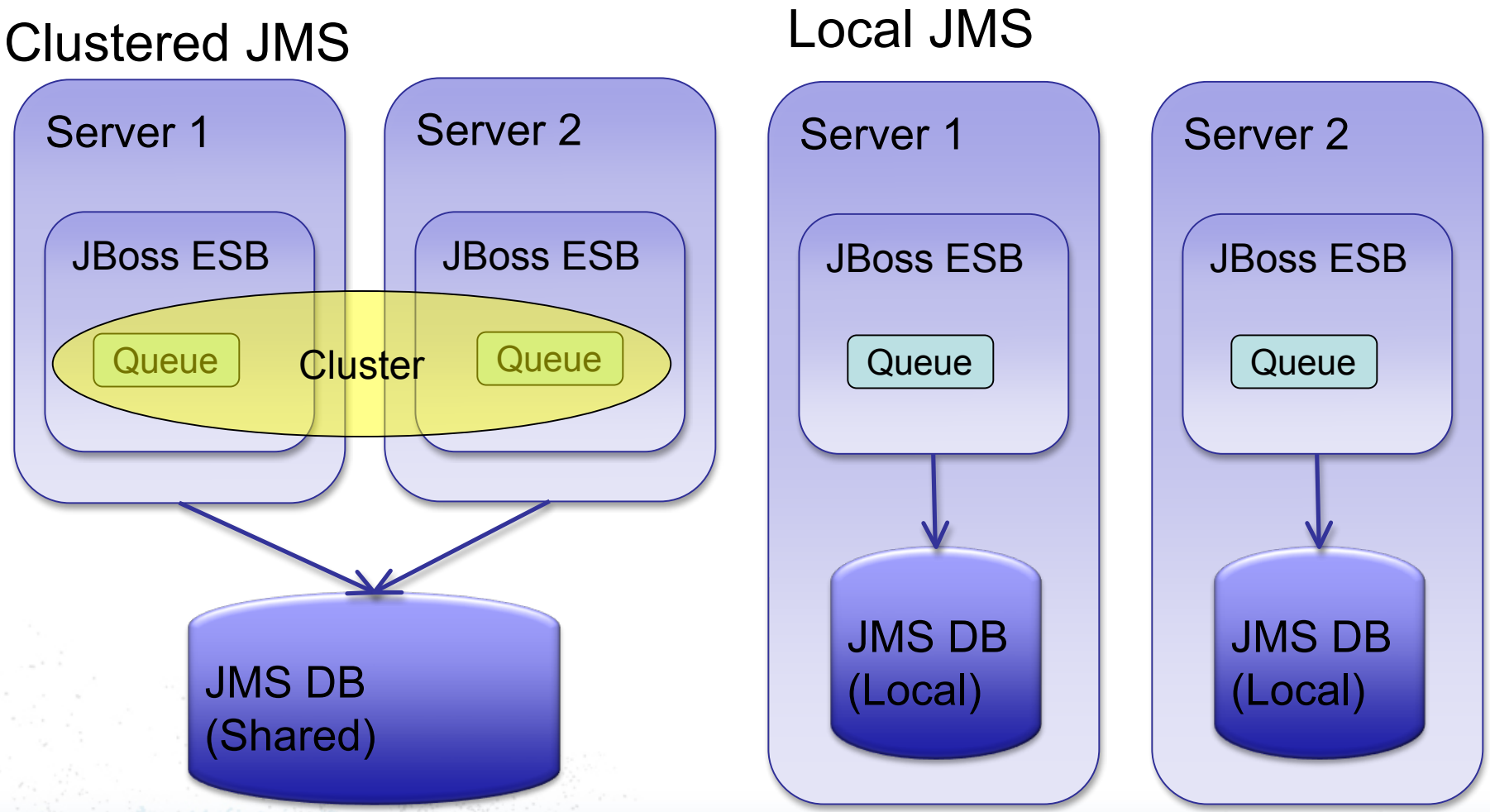
UDDI on Amazon



JMS and Services

- JBoss ESB is JMS Based
- Remote Services have JMS Endpoints
  - For ESB Aware
- Service Invoker retrieves JMS EPR
- Service Invoker enqueues Message

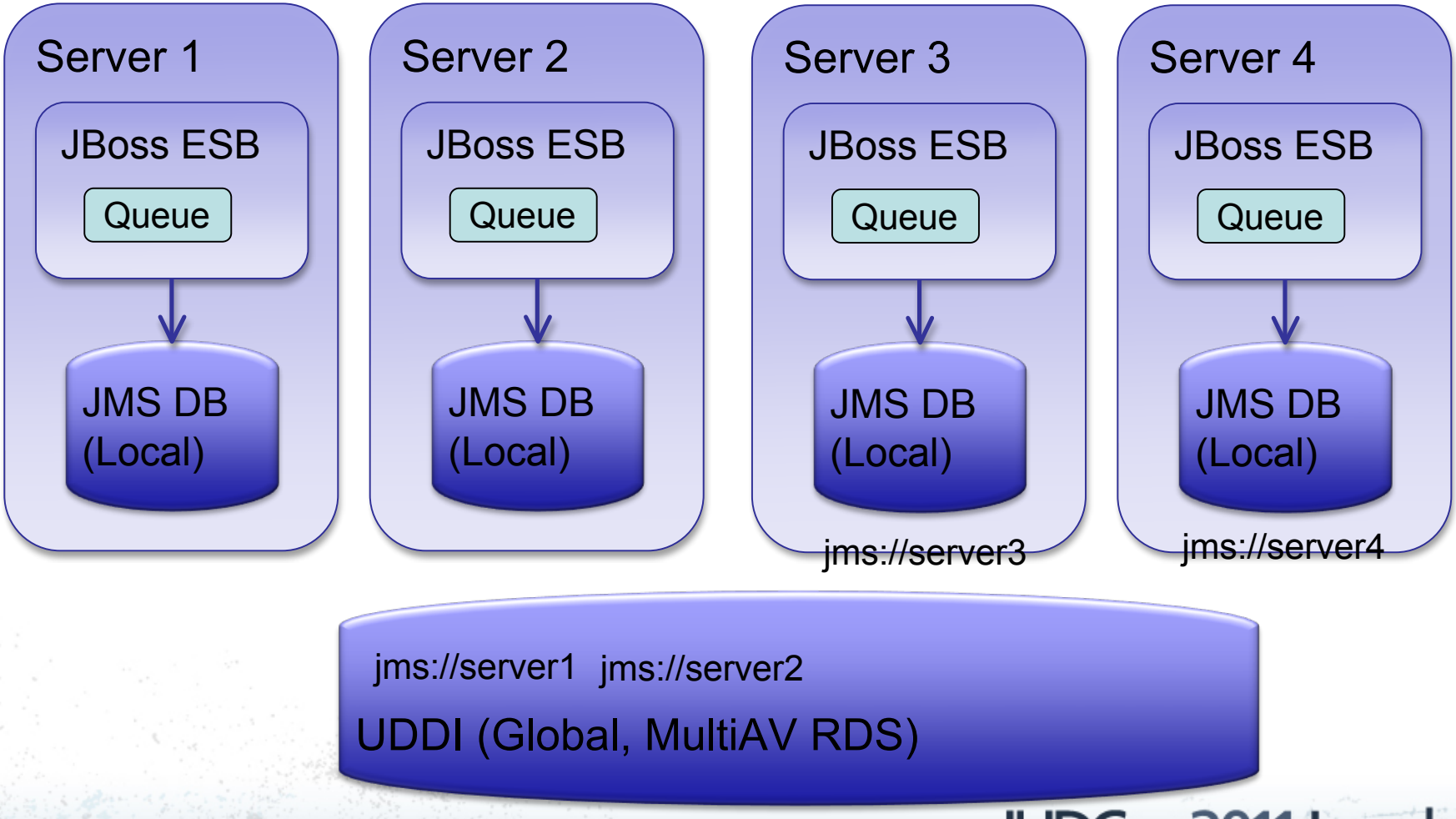
ESB JMS Options



Local JMS Advantages

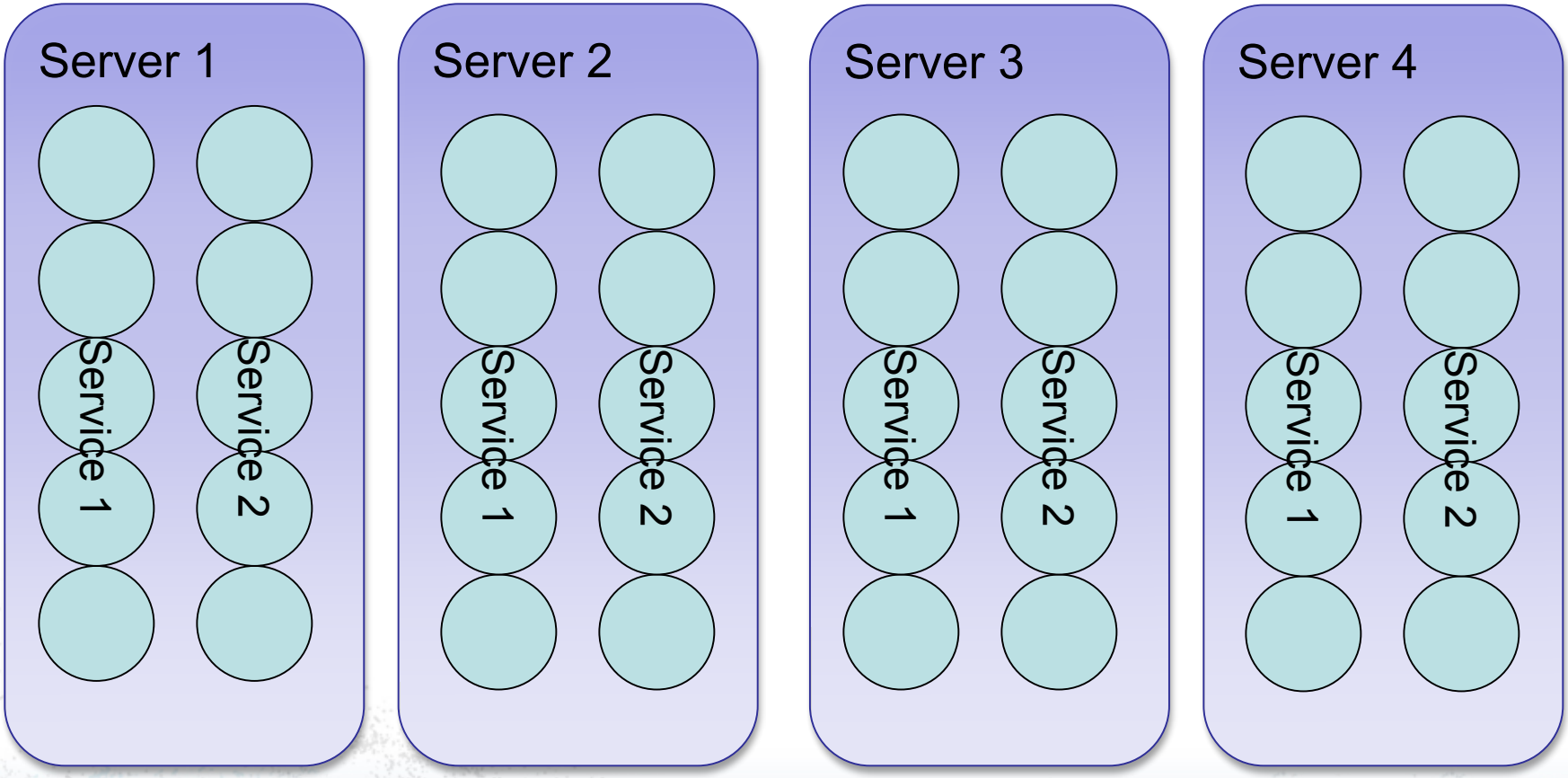
- Load Balancing Performed by Service Invoker
- Performance Improved
- Each Node Stand Alone
- No HA Database Required
- Use Local MySQL stores
- Enables Elasticity

Elastic SOA Architecture



Service Deployment Options

Service Deployment Homogenous



Homogenous Deployment

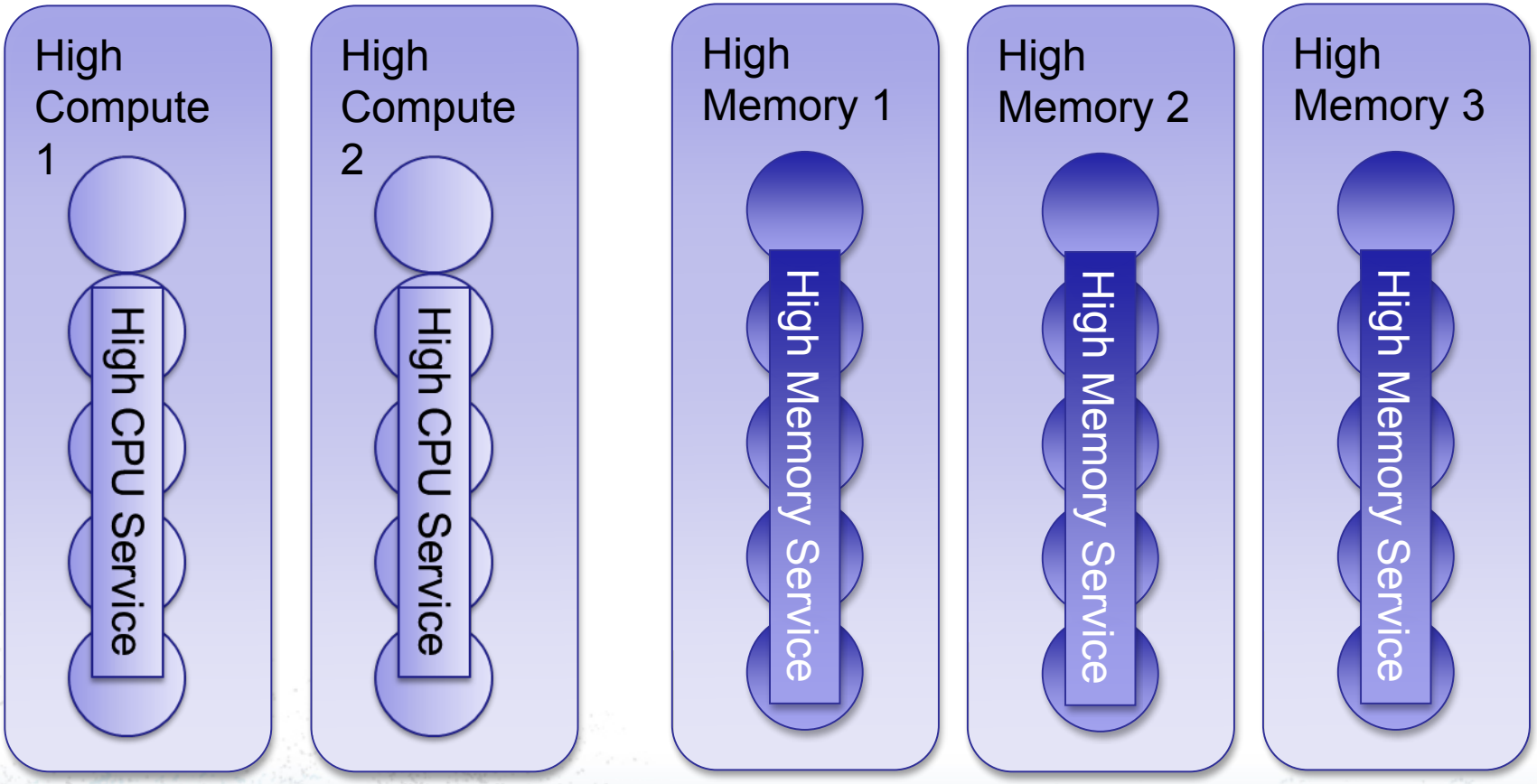
Advantages

- Easy Management
- Easy Deployment
- Simple Scaling
- Enables All inVM Processing
- Easier to Test on single server

Disadvantages

- Doesn't take into account usage patterns
- Can't Scale Services Independently
- Encourages Monolithic Deployment
- Encourages tight service coupling

Heterogeneous Deployment



Heterogeneous Deployment

Advantages

- Each Service can be independently Tuned
- Each Service can be independently scaled
- Services don't impact each other
- Services can be independently upgraded
- Encourages loose service coupling

Disadvantages

- More Complex Deployment
- More Complex Management
- Development more complex
- Calls are remote across services

Summary

- JBoss ESB's Architecture Enables Elasticity using IAAS
- JON Alerting can be used to Elastically Scale services
- Centralised UDDI enables Service Discovery across the cloud
- Centralised UDDI Enables Heterogeneous Deployment