



# How to deal with complex deployments on Java EE™ distributed platform ?

Julien Legrand, Bull  
Mickael Leduque, Serli



# Introduction

## ➔ Deployment over a distributed Java EE™ platform is a difficult task

- Configuration
  - Application Servers (eg. 46 heterogeneous configuration files on one JOnAS instance)
- Deployment
  - Deploying the configured servers on computers
  - Deploying applications on servers
- Update
  - Update servers configuration
  - Change applications version

# Introduction

➔ **Solutions around JOnAS & JASMINe to easily deploy a Java EE™ infrastructure, from middleware to applications:**

- JASMINe Design
- JOnAS deployment plan
- JOnAS services on-demand

# JASMINe Design

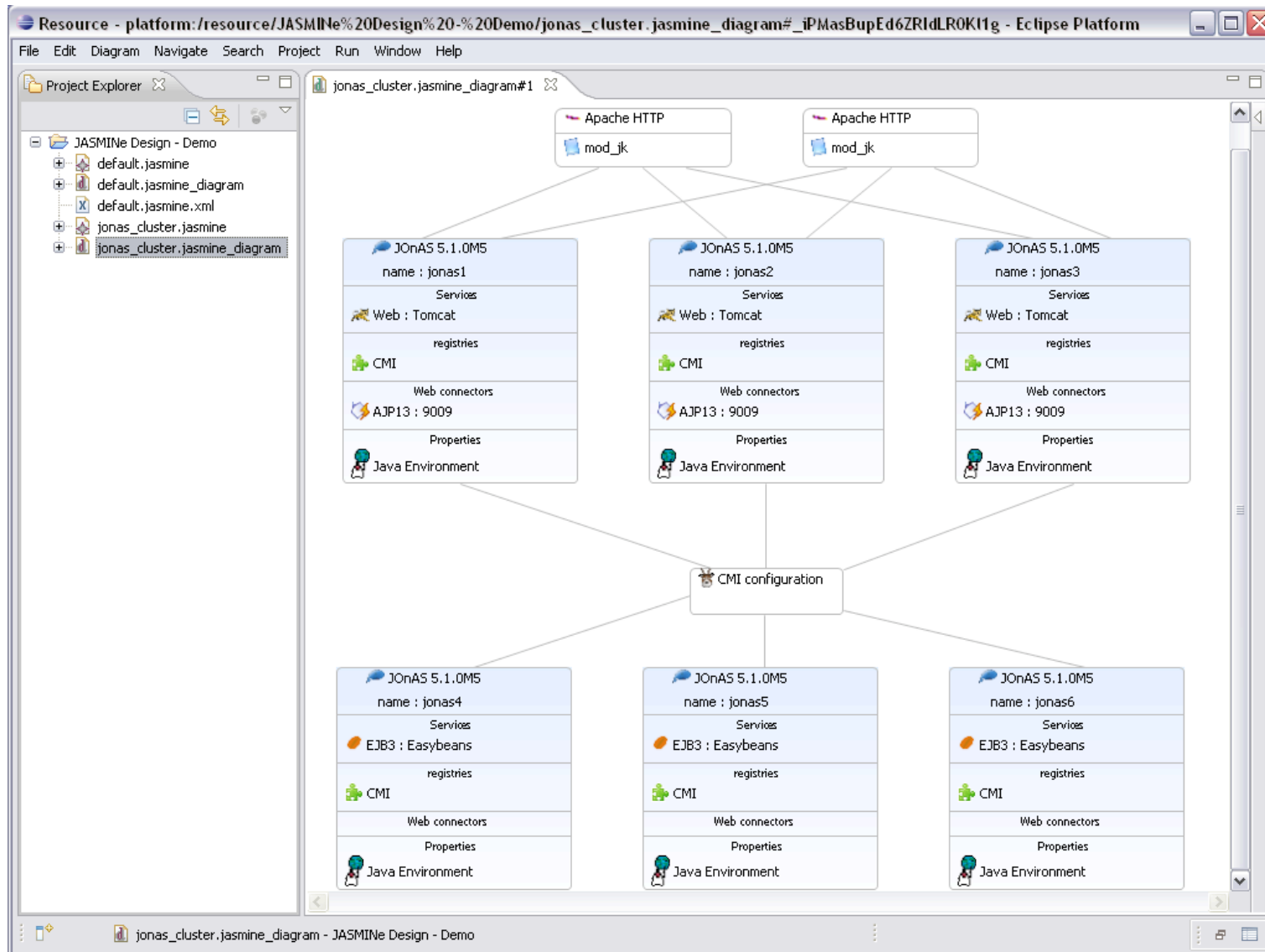
# Objectives



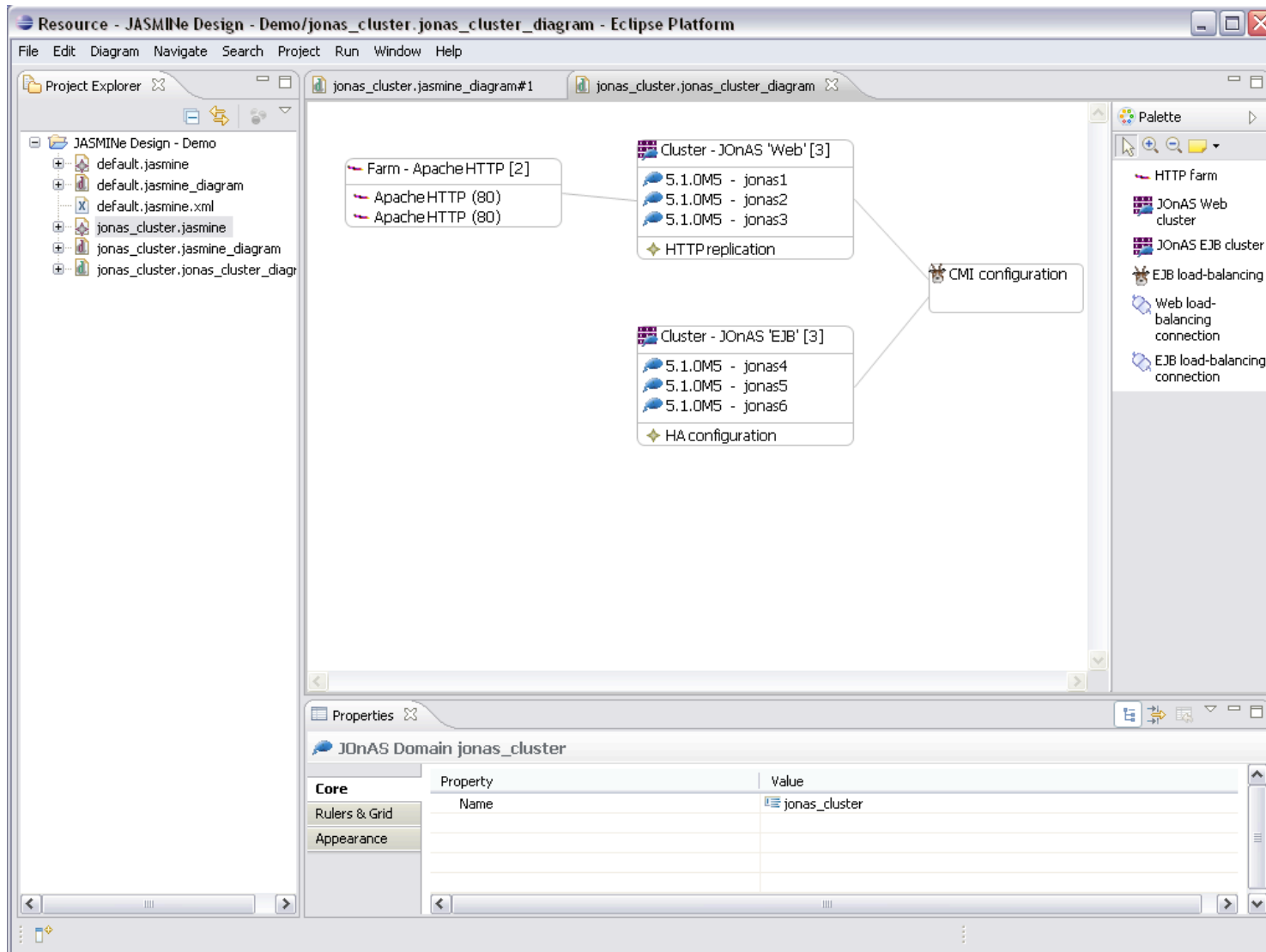
- ➔ Design,
- ➔ Configure,
- ➔ Validate,
- ➔ and Deploy ...

**... a distributed Java EE™ platform.**

# Design (1/3)



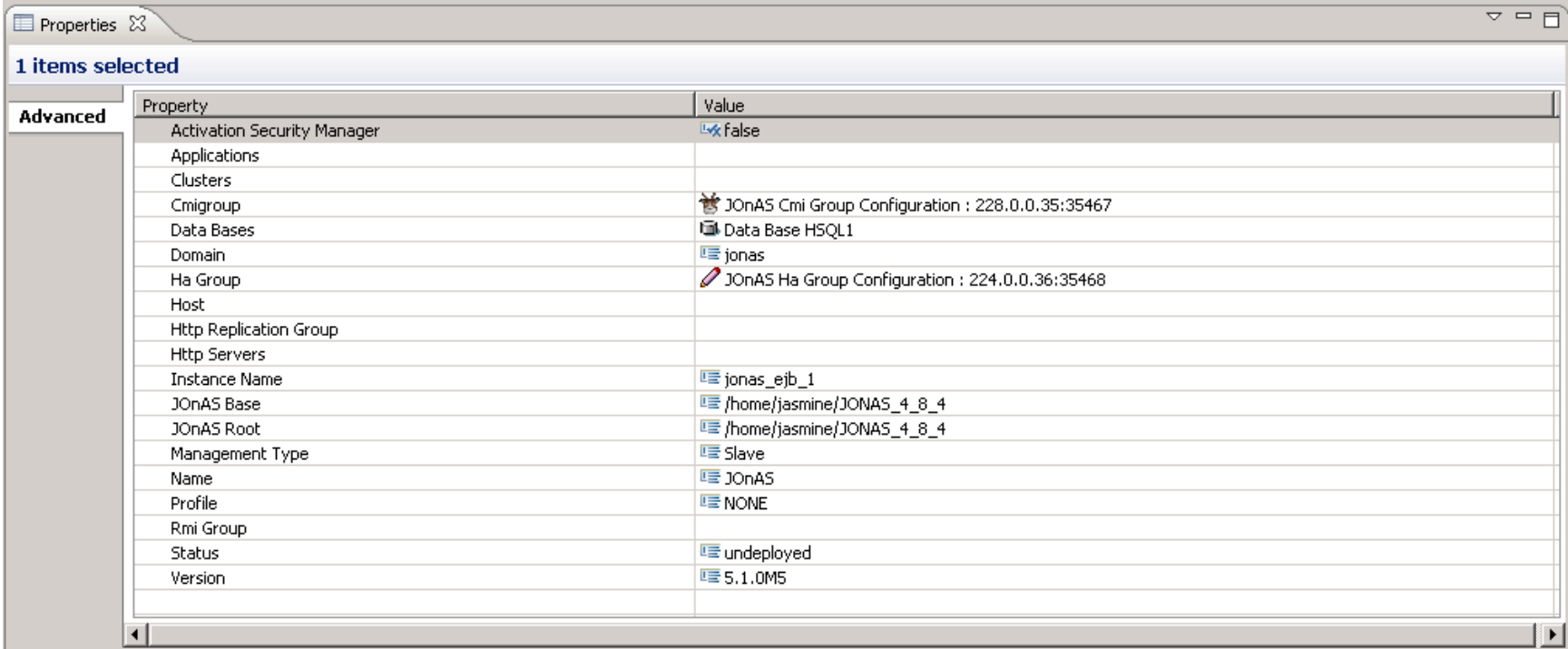
# Design (2/3)



## Design (3/3)

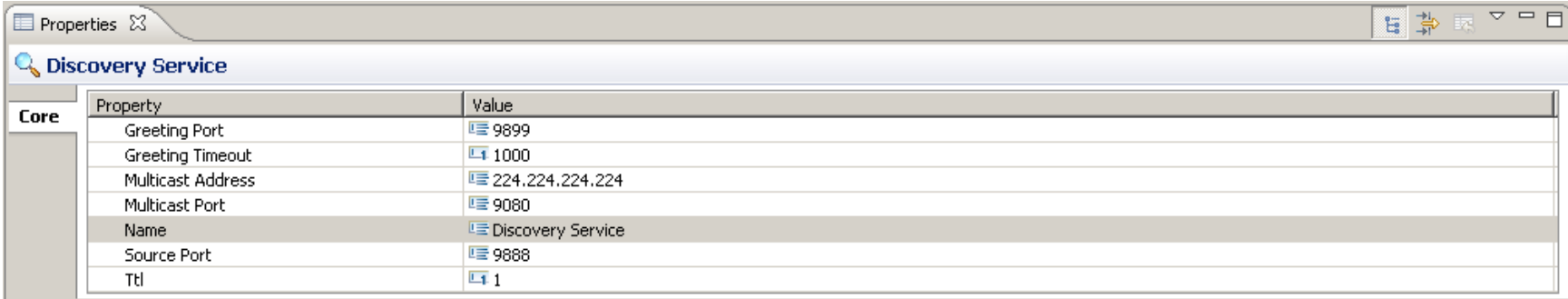
- ➔ **Predefined JOnAS profiles (web, EJB, master, ...)**
- ➔ **Wizard to create JOnAS cluster**

# Configure (1/2)



1 items selected

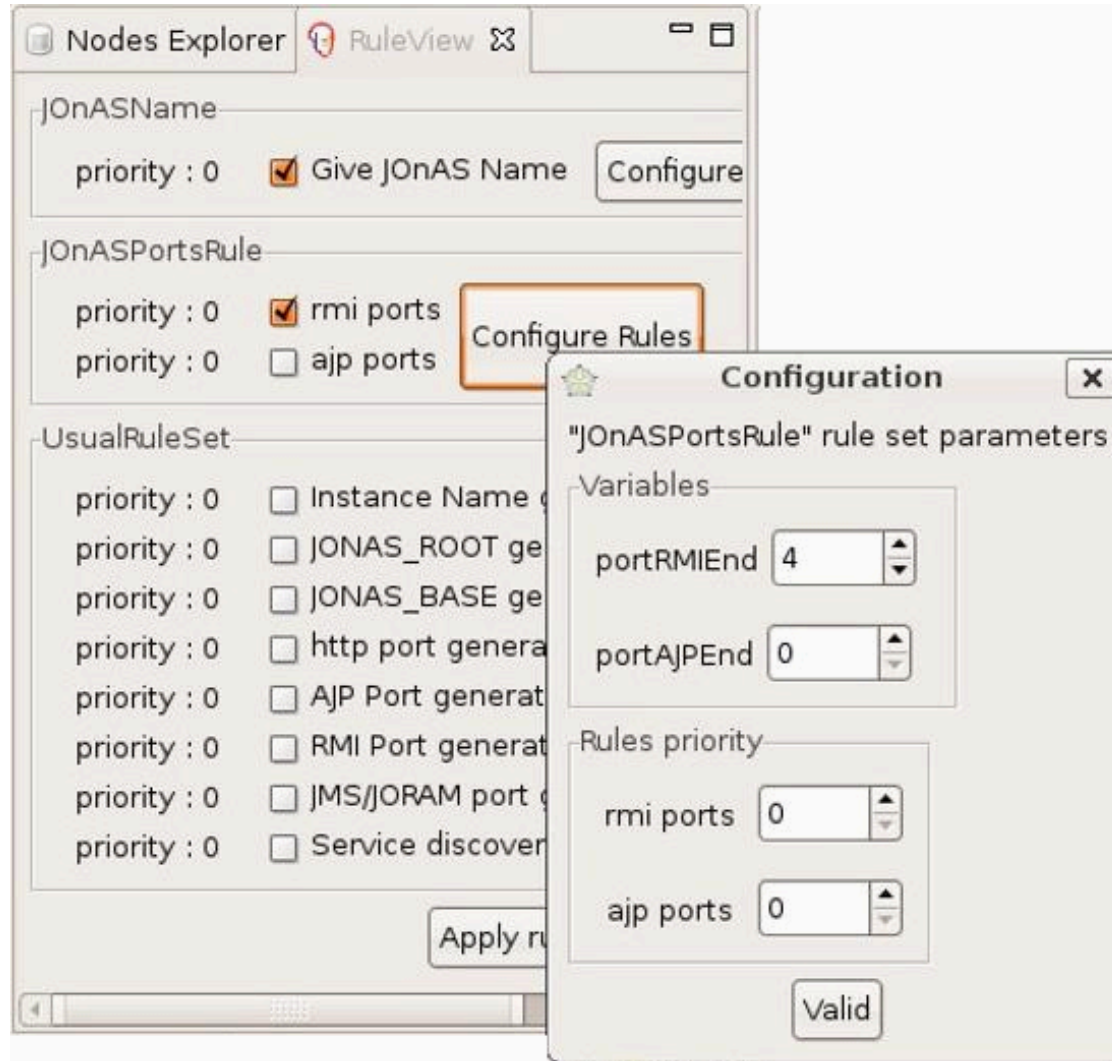
Property	Value
Activation Security Manager	✖ false
Applications	
Clusters	
Cmigroup	🐜 JOnAS Cmi Group Configuration : 228.0.0.35:35467
Data Bases	🗄 Data Base HSQL1
Domain	📁 jonas
Ha Group	✍ JOnAS Ha Group Configuration : 224.0.0.36:35468
Host	
Http Replication Group	
Http Servers	
Instance Name	📁 jonas_ejb_1
JOnAS Base	📁 /home/jasmine/JONAS_4_8_4
JOnAS Root	📁 /home/jasmine/JONAS_4_8_4
Management Type	📁 Slave
Name	📁 JOnAS
Profile	📁 NONE
Rmi Group	
Status	📁 undeployed
Version	📁 5.1.0M5



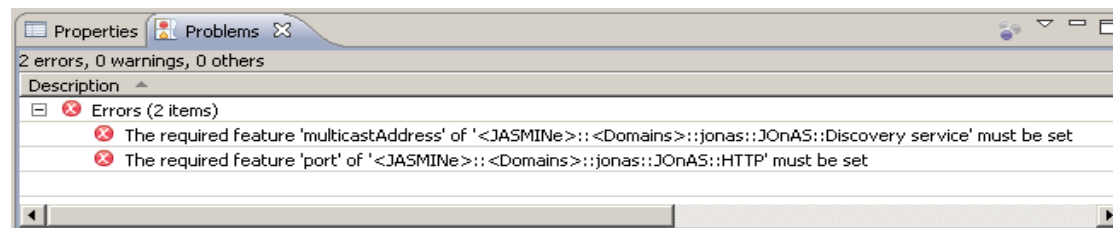
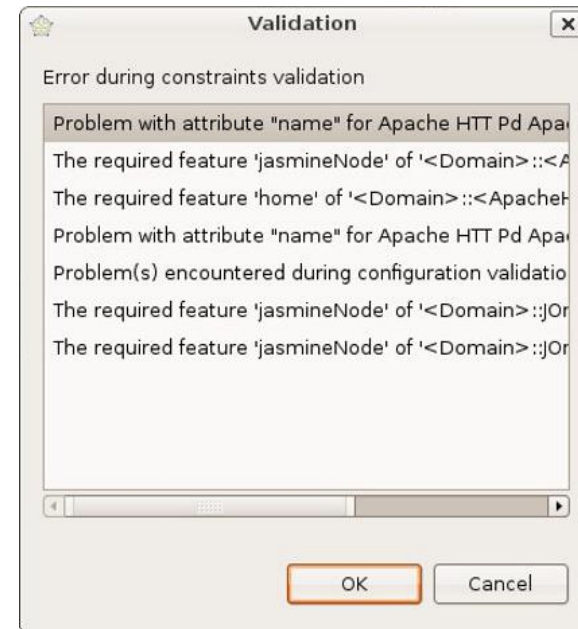
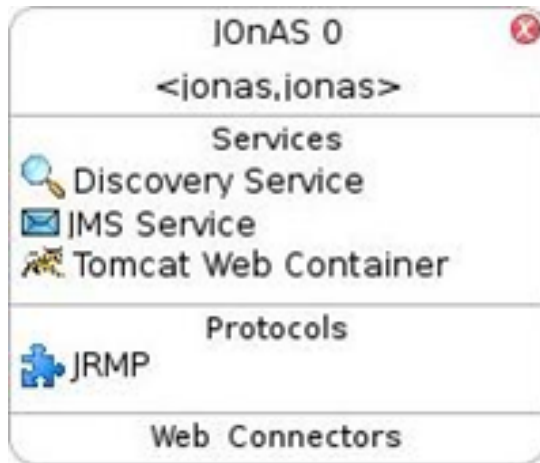
Discovery Service

Property	Value
Greeting Port	📁 9899
Greeting Timeout	📁 1000
Multicast Address	📁 224.224.224.224
Multicast Port	📁 9080
Name	📁 Discovery Service
Source Port	📁 9888
Ttl	📁 1

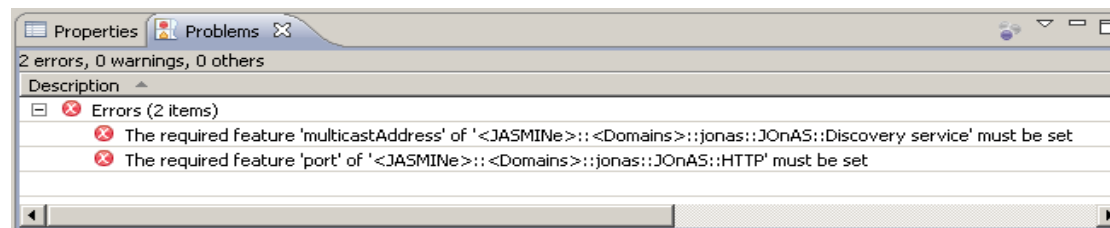
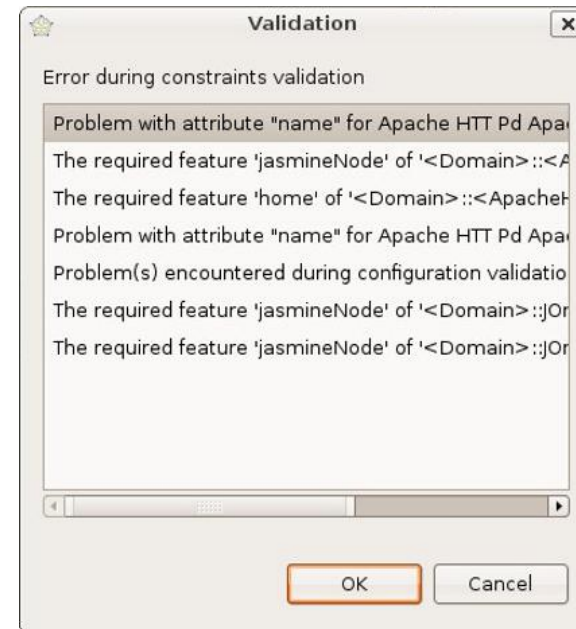
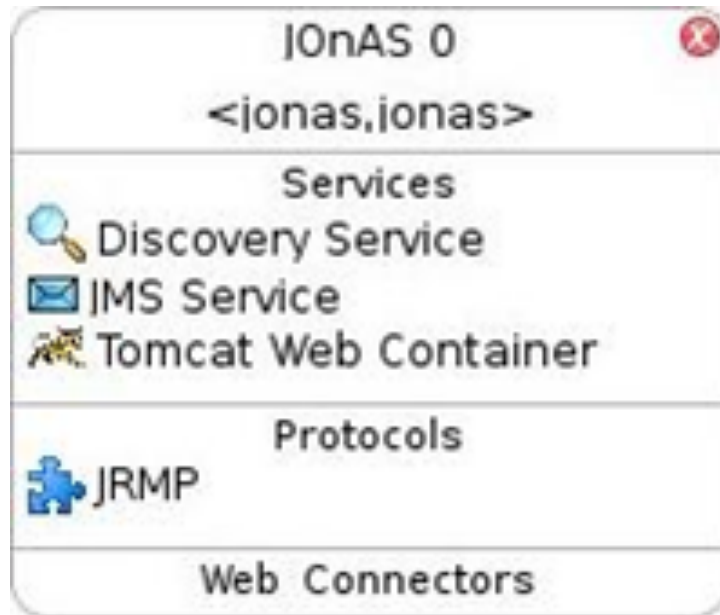
# Configure (2/2)



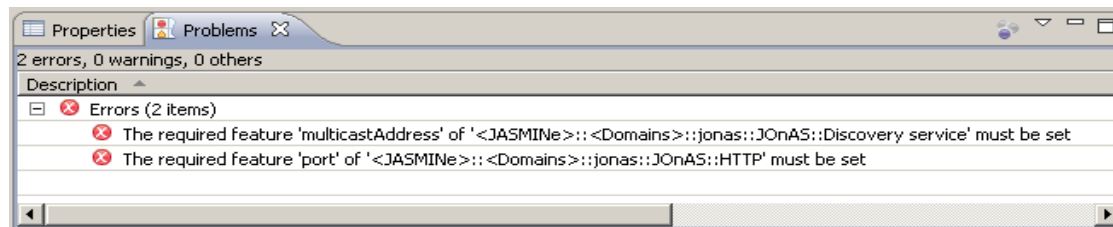
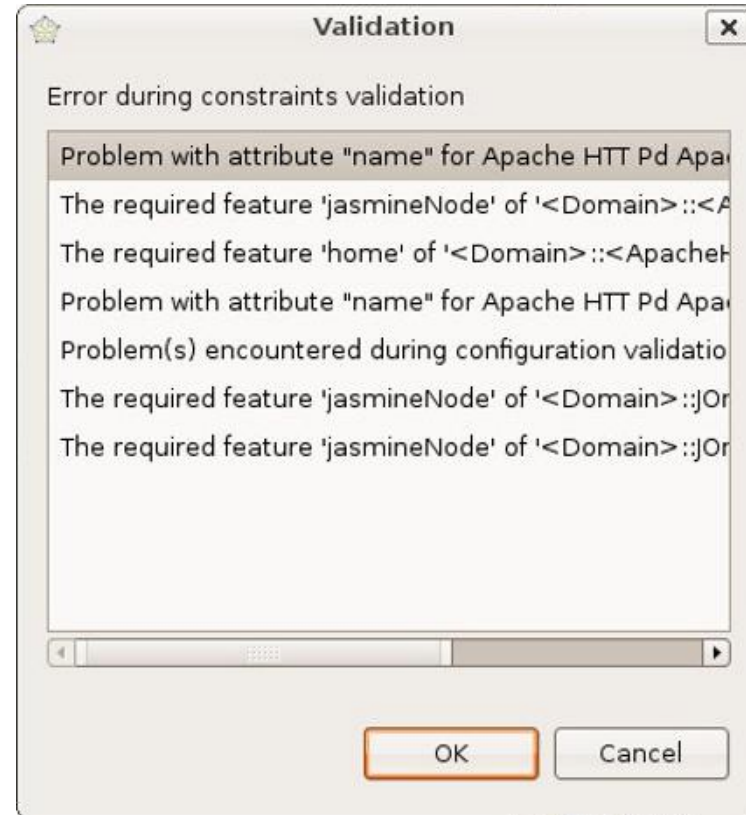
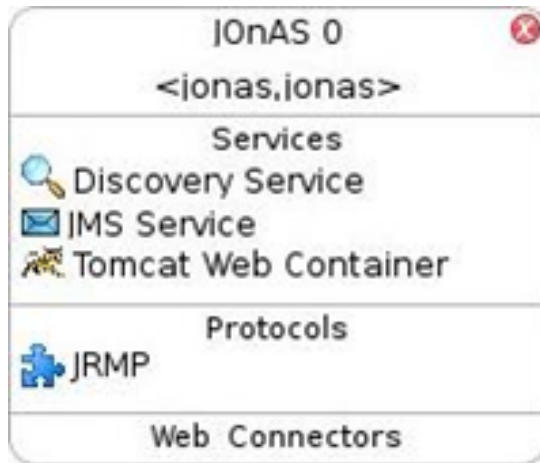
# Validate



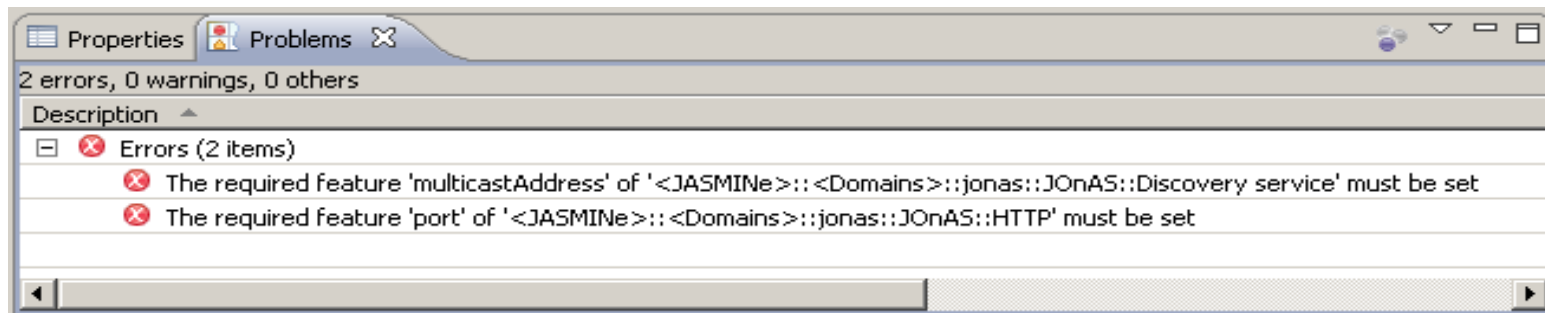
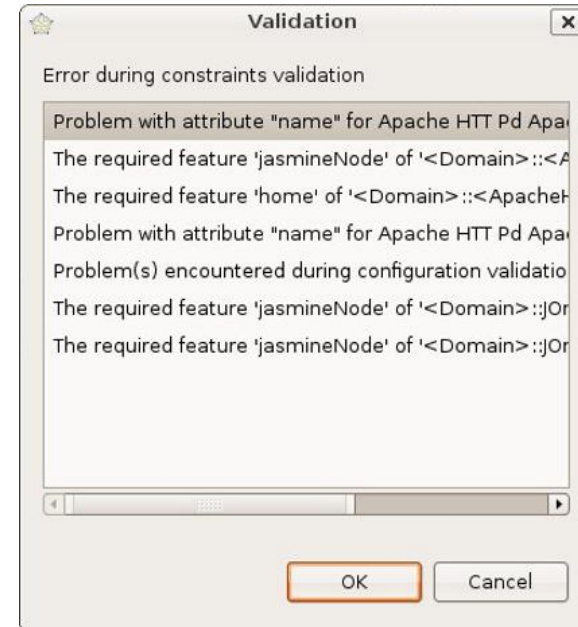
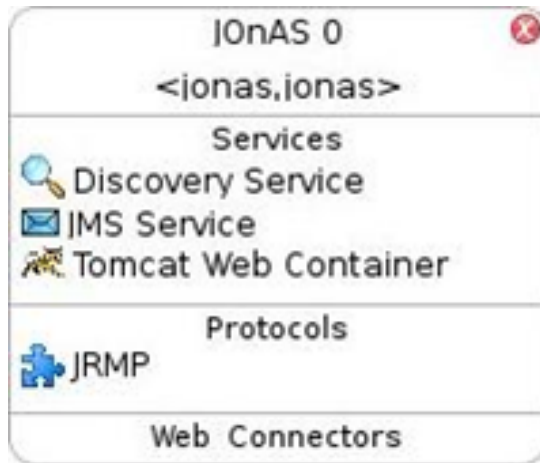
# Validate



# Validate



# Validate

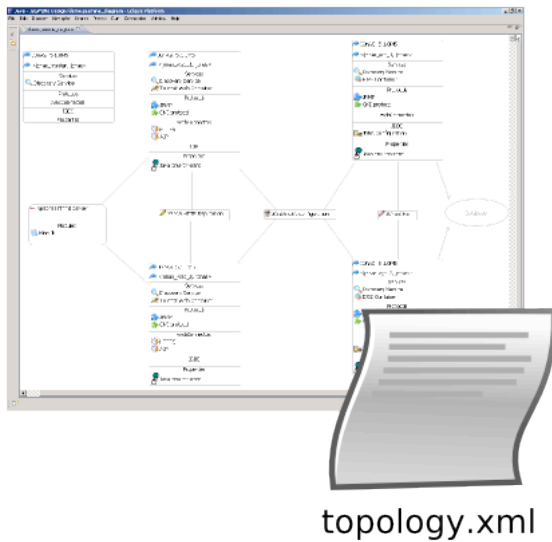


## ➔ JASMINe Deploy tools

- DeployME
  - Uses the file generated by JASMINe Design
  - Needs to be executed on each server hosting JOnAS instance(s).
  
- Jade
  - Agents (jadenode) need to be deployed on each server
  - The deployment configuration is sent to jadenodes.

# Deploy (2/2)

## JASMINe Design



Deployment mode "push"  
with Jade



Deployment mode "pull"  
with DeployME



**Once the Java EE™ platform is ready ...**

**... you can deploy applications on it.**

# **JOnAS deployment plan**

# Objectives

- ➔ **Ease multiple resources deployments**
- ➔ **Ease deployment on clusters**
- ➔ **Allow to auto-update deployed resources**

## What are deployment plans? (1/2)

- ➔ **A description of a set of resources to be deployed on a server**
  - XML file containing a list of resources with their location
  - A deployment plan can itself be considered as a resource
  
- ➔ **The resources can be obtained from remote locations**
  - Maven2 repositories
  - OBR (OSGi Bundle Repository)
  - HTTP server

## What are deployment plans? (2/2)

```
<deployment id="EZB-Maven2" type="m2:maven2-deploymentType">  
  <m2:groupId>org.ow2.easybeans.osgi</m2:groupId>  
  <m2:artifactId>easybeans-core-for-jonas-hibernate</m2:artifactId>  
  <m2:version>1.1.0-RC1</m2:version>  
</deployment>
```

```
<deployment id="EZB-URL" type="url:url-deploymentType">  
  <url:resource>  
    /home/easybeans/easybeans-core-for-jonas-hibernate.jar  
  </url:resource>  
</deployment>
```

# Deployment on cluster

## ➔ Deployment on a cluster

- Resources must be provided on all nodes

## ➔ Basic solution

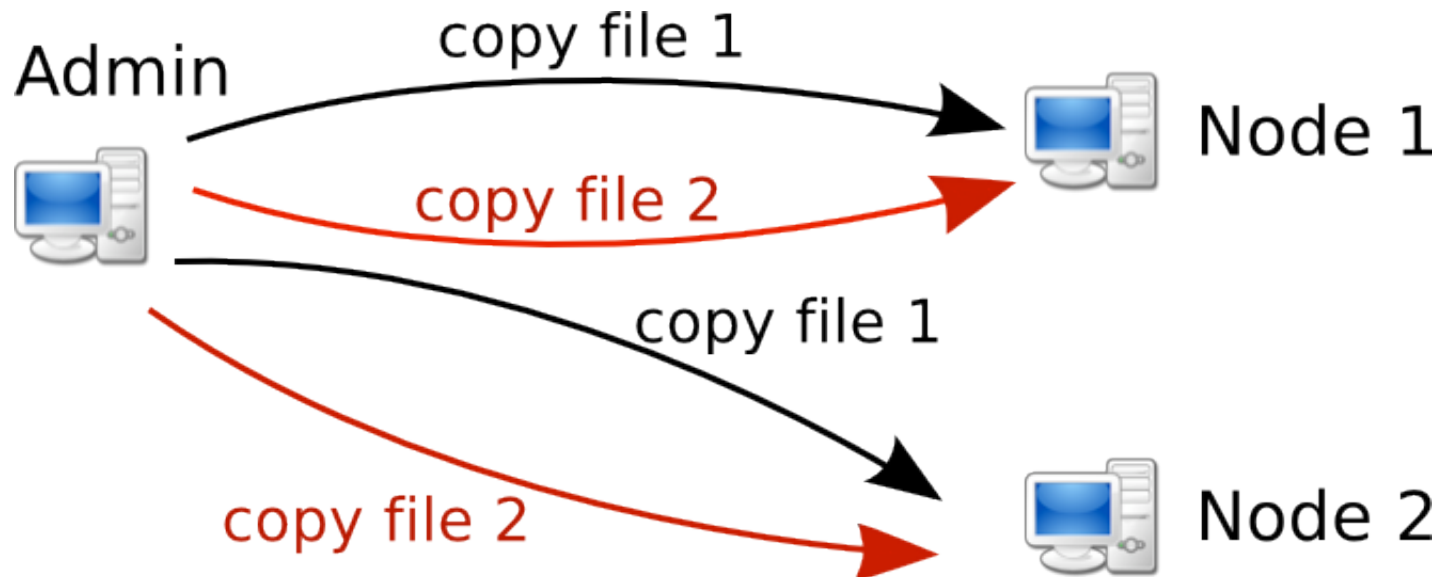
- Either manually copy it on each node
- Or upload from the master

## ➔ Solution using the deployment plans

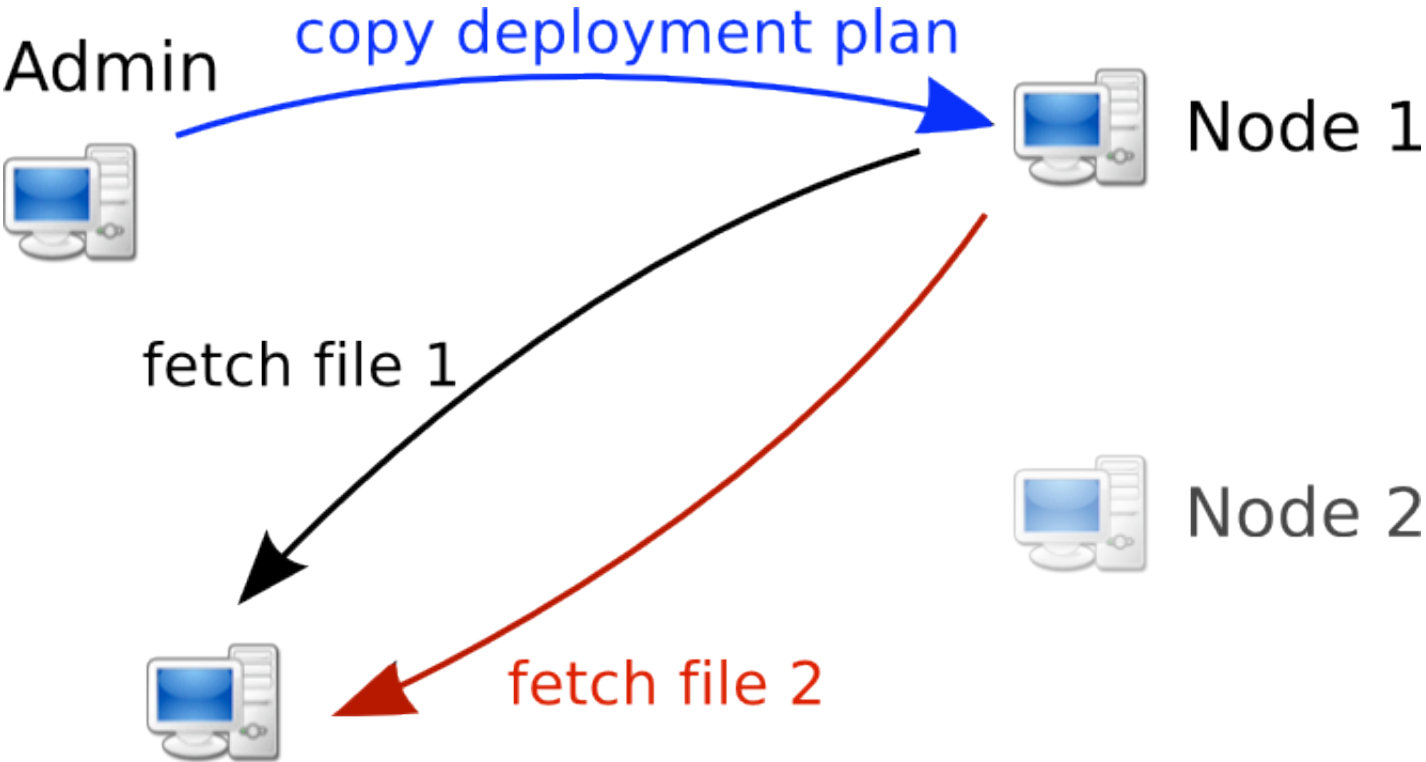
- Use remote repositories
- Only upload a deployment plan on the nodes
- The nodes will get the resources by themselves from the repositories

## Deployment on cluster: basic

### ➔ Resource copy



# Deployment on cluster: deployment plan



# Automatic update: principle

## ⇒ Check / Update

- Optionally (deployment plan option), the server can poll the repository to check for changes
- If a change is detected, the server ...
  - ... fetches the new version,
  - undeploys and redeploys the resource.

## Automatic update: use cases

### ➔ During development – fast compile/test cycles

- The local Maven2 repository is used as resource repository
- *mvn deploy* on the developer computer
- Automatic undeploy/redeploy on the server
- Test

### ➔ On cluster: automatically update all nodes

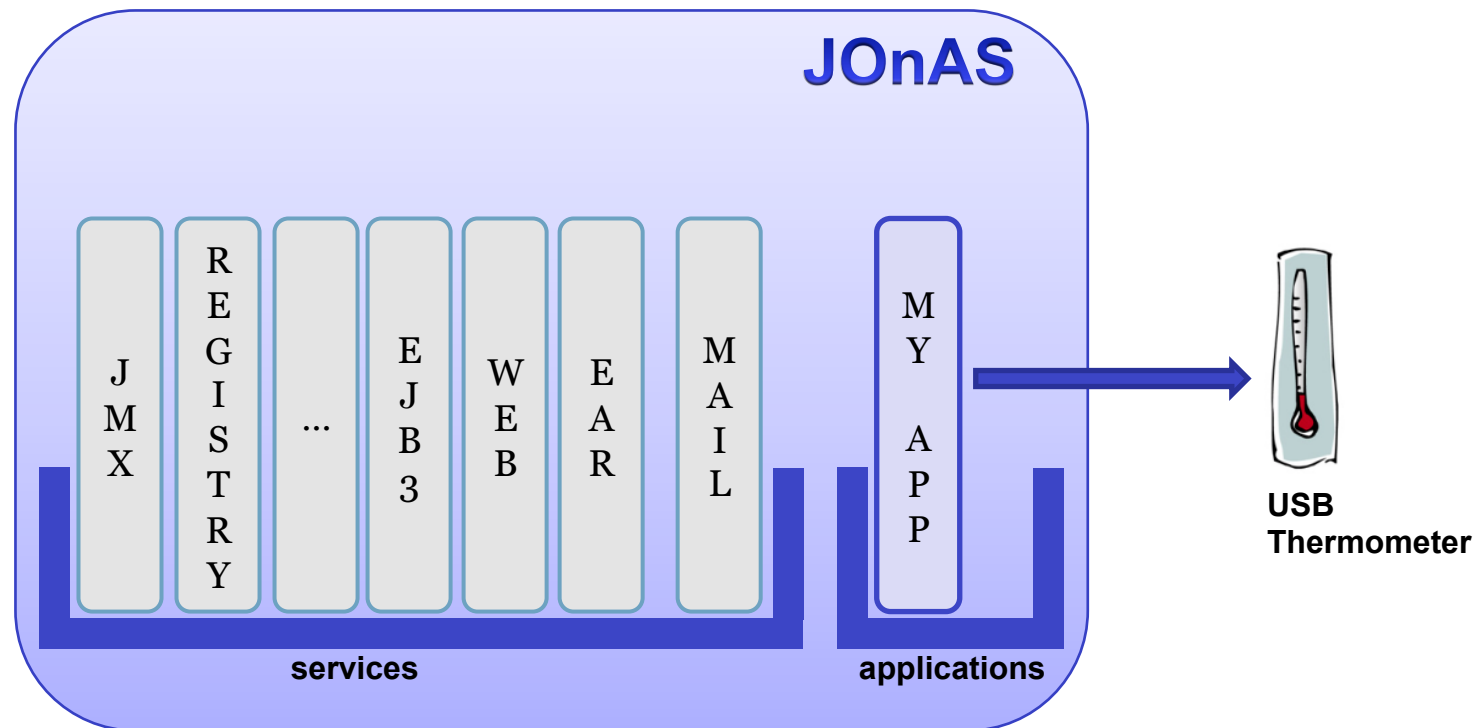
- Change a resource on the repository
- All nodes detect the change and redeploy

**And what if applications need a specific JOnAS  
configuration ?**

# **JOnAS 'service on-demand'**

## Objectives

- ➔ Configure a JOnAS server according to deployed applications.



# **JOnAS, an adaptative platform**

**⇒ JOnAS is based on OSGi**

**⇒ Deployment,**

**⇒ Undeployment,**

**⇒ and Update ...**

**... of services at runtime**

## JOnAS service on-demand (1/2)

### ➔ Application declares the required JOnAS services

- Add an entry to the MANIFEST

```
Manifest-Version: 1.0
....
Require-JOnAS-Services: mail
```

### ➔ Service preferences are defined in the JOnAS configuration (implementation, version, ...)

## **JOnAS service on-demand (2/2)**



- ➔ Before deploying the service, all requirements are automatically deployed**
- ➔ Automatic undeployment of services which have been automatically deployed and which are no longer needed**
- ➔ Checks performed when an application is undeployed**

# Conclusion - Summary

## ⇒ How to deal with complex deployments on Java EE™ distributed platform ?

- JASMINe Design / Deploy
  - Design, configure and deploy a distributed Java EE™ platform
- JOnAS Deployment plan
  - Deploy, undeploy, update applications on a distributed Java EE™ platform
- JOnAS services on-demand
  - Automatic deployment and undeployment of JOnAS services when required by applications

# Conclusion - Benefits

## ➔ JASMINe Design / Deploy

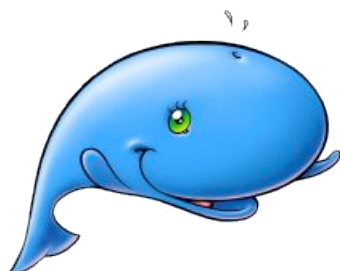
- Decreased Risks (complex config is error prone)
- Save time/ reduce exploitation cost thanks to easy configuration tool and shortcuts (wizards ...) and automatic deployment

## ➔ Deployment plan

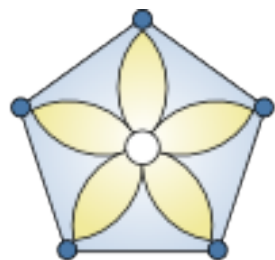
- Reduce development cycles
- Provide uniform deployment for heterogeneous resources
- Facilitate deployment of applications on clusters

## ➔ On demand services

- Ease of exploitation thanks to self-adaptation to application needs
- Resource consumption optimization



***jonas.ow2.org***



***jasmine.ow2.org***