



Integrated Cloud Applications & Platform Services



# Oracle WebLogic Server 12c: Administration II

Student Guide – Volume II  
D80153GC20  
Edition 2.0 | December 2016 | D96007

Learn more from Oracle University at [education.oracle.com](http://education.oracle.com)

## Authors

Mark Lindros  
TJ Palazzolo  
Al Saganich  
Tom Eliason

## Technical Contributors and Reviewers

Bill Bell  
Elio Bonazzi  
Tom McGinn  
Eduardo Moranchel Rosales  
Will Lyons  
David Cabelus  
Greg Stachnick  
Donna Micozzi  
Jon Patt  
Matthew Slingsby  
Bill Albert  
Rich Whalen  
Kevin Tate  
Serge Moiseev  
Takyiu Liu  
Angelika Krupp  
Viktor Tchemodanov  
Diganta Choudhury  
Jose Alvarez  
Alexander Ryndin

## Editor

Raj Kumar

## Graphic Designers

Seema Bopaiah  
Kavya Bellur

## Publishers

Sujatha Nagendra  
Raghunath M  
Asief Baig

**Copyright © 2016, Oracle and/or its affiliates. All rights reserved.**

### Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

### Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

### U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

### Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

## Contents

### **1 Course Introduction**

- Course Objectives 1-2
- Target Audience 1-4
- Course Prerequisites 1-5
- Introductions and Setting Expectations 1-6
- Course Schedule 1-7
- Course Practices 1-10
- Classroom Guidelines 1-11
- Facilities in Your Location 1-12
- For More Information 1-13
- Related Training 1-14

### **2 WebLogic Server Review**

- Objectives 2-2
- Agenda 2-3
- Oracle WebLogic Server: Overview 2-4
- WebLogic Server Domain 2-6
- Administration Server 2-7
- Managed Servers 2-8
- Machines and Clusters 2-9
- WebLogic Server Application Services 2-10
- WebLogic Server Application: Example 2-11
- Agenda 2-12
- Start Scripts 2-13
- Node Manager 2-14
- WebLogic Tools: Administration Console 2-15
- WebLogic Tools: WebLogic Scripting Tool (WLST) 2-16
- Deployment 2-17
- Summary 2-18

### **3 Upgrading WebLogic Server**

- Objectives 3-2
- Agenda 3-3
- Definition of Upgrade 3-4
- Patch 3-5

Minor Upgrade	3-6
Major Upgrade	3-7
Quiz	3-8
Agenda	3-9
What Is a Rolling Upgrade?	3-10
Multiple Installation and Domain Locations	3-11
Leverage WebLogic Clusters to Avoid Down Time	3-12
Quiz	3-14
Agenda	3-15
Rolling Upgrade Process: Overview	3-16
Backup	3-17
Shutdown	3-18
Upgrade	3-19
Restart	3-20
Quiz	3-21
Summary	3-22
Practice 3-1 Overview: Setting Up the Course Practice Environment	3-23
Practice 3-2 Overview: Performing a Rolling Upgrade	3-24

#### **4 Creating and Using Domain Templates**

Objectives	4-2
Agenda	4-3
Domain Template Review	4-4
Agenda	4-5
Template Contents	4-6
Template Exclusions	4-7
Script Replacement Variables	4-8
Agenda	4-9
Extension Template: Concepts	4-10
Agenda	4-11
Fusion Middleware (FMW) Templates	4-12
Quiz	4-13
Agenda	4-15
Why Use Custom Templates	4-16
Template Builder	4-17
Creating a Domain Template	4-18
Selecting the Template Domain Source	4-19
Configuring Applications in a Template	4-20
Adding Files to the Template	4-21
Preparing Scripts and Files with Variables	4-22
Review Configuration and Create Template	4-23

SQL Scripts and LDAP Data	4-24
Creating an Extension Template	4-26
Using a Custom Template with the Configuration Wizard	4-28
Post Domain Creation Tasks	4-34
Using a Custom Extension Template with the Configuration Wizard	4-35
Summary	4-37
Practice 4-1 Overview: Creating and Using a Custom Domain Template	4-38

## 5 WebLogic Server Startup and Crash Recovery

Objectives	5-2
Agenda	5-3
Node Manager Architecture	5-4
Node Manager Default Behavior	5-5
Configure Java-Based Node Manager	5-6
Agenda	5-7
Starting the Node Manager at System Startup	5-8
Configuring the Node Manager as a Windows Service	5-9
Configuring the Node Manager on UNIX Systems	5-10
Configuring the Node Manager as an inetd Service	5-11
Registering and Managing inetd Services	5-14
Configuring the Node Manager as an xinetd Service	5-15
Solaris	5-17
Example SMF Manifest File	5-18
Set Up and Start Node Manager with SMF	5-21
Agenda	5-22
How the Node Manager Restarts an Administration Server	5-23
How the Node Manager Restarts a Managed Server	5-24
RestartInterval and RestartMax	5-25
Crash Recovery	5-26
CrashRecoveryEnabled	5-27
Quiz	5-28
Summary	5-29
Practice 5-1 Overview: Configuring Automatic Start and Restart of a System	5-30

## 6 WebLogic Scripting Tool (WLST)

Objectives	6-2
Agenda	6-3
WebLogic Scripting Tool (WLST)	6-4
Agenda	6-5
Jython	6-6
Using Jython	6-7

Variable Declaration	6-8
Conditional Expressions	6-9
Loop Expressions	6-10
I/O Commands	6-11
Exception Handling	6-12
Quiz	6-13
Agenda	6-14
WLST Modes	6-15
WLST Example	6-16
Running WLST Scripts	6-17
WLST Development Tools	6-18
configToScript()	6-19
Script Recording Using the Administration Console	6-20
Oracle Enterprise Pack for Eclipse	6-22
WLST Command Tips	6-23
General WLST Commands	6-24
Offline WLST Commands	6-25
Online WLST Commands	6-26
Quiz	6-27
Agenda	6-28
WebLogic JMX Overview	6-29
Configuration MBeans	6-30
Runtime MBeans	6-31
WebLogic Server MBean Examples	6-32
MBean Properties in the Administration Console	6-33
Browsing MBean Documentation	6-34
Referencing MBeans in WLST	6-36
Quiz	6-37
Agenda	6-38
Creating a Template and a Domain	6-39
Connecting to a Server	6-40
Password Management	6-41
WLST Variables	6-42
Password Management	6-43
Configuring a Server	6-44
Monitoring a Server	6-45
Adding a Server to a Cluster	6-46
Creating a Data Source	6-47
Dynamic Cluster Scaling	6-49
Monitoring a Data Source	6-50
Creating an LDAP Authentication Provider	6-51

Modifying a Domain Offline	6-52
Deploying an Application	6-53
Creating a Dynamic Cluster	6-54
Scaling Up a Dynamic Cluster	6-56
Scaling Down a Dynamic Cluster	6-57
Quiz	6-58
Agenda	6-59
Some FMW Commands	6-60
Summary	6-61
Practice 6-1 Overview: Creating and Modifying a Domain with WLST	6-62
Practice 6-2 Overview: Monitoring a Domain with WLST	6-63

## 7 RESTFul Management Services

Objectives	7-2
Agenda	7-3
What Is REST?	7-4
WebLogic Server and REST	7-5
What Can Be Monitored Using WebLogic REST?	7-6
Clients and WebLogic REST	7-7
Enabling REST Management	7-8
Multitenant (MT) Versus Nonmultitenant environments	7-9
Supported Representations	7-10
REST Resource Access Roles	7-11
Agenda	7-12
WebLogic RESTful Interfaces	7-13
REST Operations Flow	7-14
Example: Returning Server Information Using cURL	7-15
Typical REST Operations	7-16
Common WebLogic REST Results	7-17
WebLogic REST Error Handling	7-19
Understanding WebLogic REST Requests	7-20
WebLogic REST MBean Hierarchy	7-21
WebLogic Legacy REST Resources	7-22
WebLogic REST Response Bodies	7-23
Example: Listing Server Information by Name	7-24
Invoking Operations	7-25
Create Forms	7-26
Creating Resources	7-28
Agenda	7-29
Common Workflows Modify a configuration using an edit	7-30
Common Workflows Deploy/View/Un deploy an application	7-31

Common Workflows Start/View/Stop Servers	7-32
Common Workflows Monitor Servers	7-33
Summary	7-34
Quiz	7-35
Practice 7-1 Overview: Creating and Managing Servers Using REST	7-36
WebLogic REST Monitoring Relative Paths	7-37

## **8 Secure Sockets Layer (SSL)**

Objectives	8-2
Agenda	8-3
What Is SSL?	8-4
SSL Terminology	8-5
Symmetric Encryption and Decryption	8-6
Asymmetric Encryption and Decryption	8-7
Digital Certificates	8-9
Digital Certificate: Example	8-10
Certificate Authorities	8-11
SSL Communication	8-12
One-Way SSL Handshake	8-13
Two-Way SSL Handshake	8-14
Quiz	8-15
Agenda	8-17
WebLogic and SSL	8-18
Demo Certificates	8-19
WebLogic SSL Requirements	8-20
WebLogic SSL Connections	8-21
Agenda	8-23
What Is a Keystore?	8-24
keytool Utility	8-25
Working with a Keystore	8-26
Configuring WebLogic Keystores	8-28
Quiz	8-29
Agenda	8-30
Configuring SSL for WebLogic	8-31
SSL and WebLogic Proxy Plug-Ins	8-32
SSL and Oracle HTTP Server	8-34
SSL and Hardware Load Balancers	8-35
Quiz	8-36
Summary	8-37
Practice 8-1 Overview: Setting Up SSL	8-38

## **9 Shared Java EE Libraries**

- Objectives 9-2
- Agenda 9-3
- Before Shared Libraries 9-4
- Shared Libraries 9-5
- Types of Shared Libraries 9-6
- Java EE Library Support 9-7
- WebLogic Java EE Shared Libraries 9-8
- Configuration Precedence and Deployment 9-9
- Managing Precedence 9-10
- Configuring a Shared Library 9-11
- Configuring a Shared Library in the Manifest File 9-12
- Referencing a Shared Library 9-13
- Quiz 9-15
- Agenda 9-16
- Deploying a Library with the Console 9-17
- Deploying a Library with weblogic.Deployer 9-18
- Deploying a Library with WLST 9-19
- Summary 9-20
- Practice 9-1 Overview: Configuring and Deploying a Shared Library 9-21

## **10 Application Work Managers**

- Objectives 10-2
- Agenda 10-3
- WebLogic Server Threads 10-4
- Monitoring a Server Thread Pool 10-5
- Monitoring Server Threads 10-6
- Stuck Thread Handling 10-7
- Configuring Stuck Thread Handling 10-8
- Application Stuck Thread Handling 10-9
- Quiz 10-10
- Agenda 10-11
- Work Managers 10-12
- Work Manager Scope 10-13
- Work Manager Architecture 10-14
- Quiz 10-15
- Agenda 10-16
- Request Classes 10-17
- Creating a Work Manager 10-18
- Creating a Request Class 10-19
- Constraints 10-20

Creating a Constraint 10-21  
Work Manager WLST Example 10-22  
Work Managers and Stuck Threads 10-23  
Assigning Work Managers to Applications 10-24  
Quiz 10-25  
Summary 10-26  
Practice 10-1 Overview: Creating and Using Work Managers 10-27

## **11 Working with the Security Realm**

Objectives 11-2  
Agenda 11-3  
Security Realm: Review 11-4  
Default Security Configuration 11-6  
Agenda 11-7  
How WebLogic Resources Are Protected 11-8  
Examples of WebLogic Resources to Protect 11-9  
Users and Groups 11-10  
Group Membership 11-11  
Roles 11-12  
Policies 11-13  
Configuring New Users 11-14  
Configuring New Groups 11-15  
Configuring Group Memberships 11-16  
Configuring New Roles 11-17  
What Is Role Mapping? 11-18  
Configuring Role Mapping 11-19  
Configuring Roles Using WLST 11-20  
Configuring New Policies 11-21  
Configuring Policies Using WLST 11-22  
Security Configuration Sources 11-23  
Configuring Sources Using WLST and weblogic.Deployer 11-24  
Deployment Descriptor Security Example: weblogic.xml 11-25  
Deployment Descriptor Security Example: web.xml 11-26  
Embedded LDAP Server 11-27  
Configuring the Embedded LDAP Server 11-28  
Quiz 11-30  
Practice 11-1 Overview: Creating Users, Groups, Roles, and Policies 11-31  
Agenda 11-32  
Auditing 11-33  
Sample Auditing Output 11-34  
Security Audit Events 11-35

WebLogic Auditing Architecture	11-36
Custom Versus Default Auditing Provider	11-37
Creating the Default Auditing Provider	11-38
Configuring the Default Auditing Provider	11-39
Configuration Auditing	11-41
Quiz	11-43
Summary	11-44
Practice 11-2 Overview: Configuring WebLogic Auditing	11-45

## **12 Disaster Recovery and Migration**

Objectives	12-2
Agenda	12-3
Disaster Recovery Concepts	12-4
Site Symmetry	12-6
Recommended Architecture	12-7
General Best Practices	12-8
Hosts File: Example	12-9
Quiz	12-10
Agenda	12-11
Administration Server Review	12-12
Impact of Administration Server Failure	12-13
Backing Up a Domain Configuration	12-14
Recovery of the Administration Server Configuration	12-15
Restarting an Administration Server on a New Computer	12-16
Quiz	12-17
Agenda	12-18
Java Transaction API (JTA) Review	12-19
What Is Service Migration?	12-20
Service Migration Prerequisites	12-21
Service Migration Architecture: Database Leasing	12-22
Service Migration Architecture: Consensus Leasing	12-23
What Is a Migratable Target?	12-24
Service Migration Policy Options	12-25
Configuration Roadmap	12-26
JTA Service Migration: Before Failure	12-27
JTA Service Migration: After Failure	12-28
Configuring JTA Service Migration	12-29
Set Up Automatic JTA Service-Level Migration	12-30
Quiz	12-31
Practice 12-1 Overview: Configuring JTA Service-Level Migration	12-34
Agenda	12-35

Whole-Server Migration 12-36  
Automatic Server Migration Architecture: No Failure 12-37  
Automatic Server Migration Architecture: Machine Failure 12-38  
Configuration: Overview 12-39  
Quiz 12-40  
Summary 12-41

## **13 Domain Partitions**

Objectives 13-2  
Agenda 13-3  
Multitenancy 13-4  
Hardware Isolation 13-5  
Network Isolation 13-6  
Application Isolation 13-8  
Security Isolation 13-9  
Agenda 13-10  
What Is WebLogic Server 12c Multitenancy? 13-11  
Understanding Tenants and Partitions 13-12  
What Is a Domain Partition? 13-13  
WebLogic Server Multitenancy End-to-End Integration 13-14  
WebLogic Server Multitenancy Partition Isolation 13-15  
WebLogic Server Multitenancy Components 13-16  
What Are Resource Groups? 13-17  
What Are Resource Group Templates? 13-18  
Resource Group Templates Simplify Management 13-19  
Virtual Targets 13-20  
Virtual Target Versus Virtual Host 13-23  
Oracle Traffic Director 12.2.1 13-24  
Partition Security 13-25  
Partition Work Managers 13-26  
Multitenancy Examples 13-28  
Single Partition Use Case 13-29  
Agenda 13-30  
Creating Virtual Targets 13-31  
Creating Domain Partitions 13-33  
Working with Domain Partitions 13-36  
Importing and Exporting Domain Partitions 13-37  
Completing Domain Partition Import (or export) 13-39  
Application Deployment 13-41  
Quiz 13-42  
Summary 13-43  
Practice 13-1 Overview: Creating and Working with Domain Partitions 13-44

## **14 Managing Data Sources**

- Objectives 14-2
- Agenda 14-3
- Review: JDBC 14-4
- Data Source: Review 14-5
- XA Data Source Review 14-6
- Agenda 14-7
- Why Manage a Data Source? 14-8
- Suspend a Data Source 14-9
- Resume a Data Source 14-10
- Practice 14-1 Overview: Controlling a Data Source 14-11
- Agenda 14-12
- Oracle Real Application Clusters (RAC): Overview 14-13
- Oracle GridLink for RAC 14-14
- Agenda 14-15
- Multi Data Sources 14-16
- Multi Data Source Architecture 14-17
- Comparison of GridLink and Multi Data Sources 14-18
- Failover Option 14-19
- Load Balancing Option 14-20
- Quiz 14-21
- Agenda 14-22
- Connection Testing 14-23
- Agenda 14-25
- Proxy Data Sources 14-26
- Creating a Proxy Data Source 14-27
- Monitoring Proxy Data Source JDBC Resources 14-29
- Agenda 14-30
- Creating a Multi Data Source 14-31
- Configuring a Multi Data Source 14-33
- Multi Data Source WLST Example 14-34
- Managing Multi Data Source Members 14-35
- Quiz 14-36
- Summary 14-38
- Practice 14-2 Overview: (Optional) Creating and Using a Multi Data Source 14-39
- Practice 14-3 Overview: (Optional) Creating and Using a Proxy Data Source 14-40

## **15 Diagnostic Framework**

- Objectives 15-2
- Agenda 15-3
- WebLogic Diagnostics Framework (WLDF) 15-4
- WLDF Architecture 15-5
- Diagnostic Archives 15-6
- Configuring Server Diagnostic Archives 15-7
- Diagnostic Modules 15-8
- Dynamic Diagnostic Modules 15-10
- Resource Descriptors 15-11
- Creating a Diagnostic Module 15-12
- WLST: Example 15-13
- WLST Commands for WLDF 15-14
- Quiz 15-15
- Agenda 15-16
- What Is a Diagnostic Image? 15-17
- Capturing a Server Diagnostic Image 15-18
- WLST: Example 15-19
- Quiz 15-20
- Agenda 15-21
- What Is a Harvester? 15-22
- Metric Collectors 15-23
- Configuring a Metric Collector 15-24
- WLST: Example 15-25
- Quiz 15-26
- Agenda 15-27
- Policies and Actions 15-28
- Policy Types 15-30
- Configuring Policies 15-31
- Configuring Actions 15-33
- Elastic Actions and Scaling 15-36
- Scale up Action 15-37
- Scale down action 15-38
- Quiz 15-39
- Summary 15-40
- Practice 15-1 Overview: Using a Built-in Diagnostic Module 15-41

## **16 WebLogic and Coherence Integration**

- Objectives 16-2
- Agenda 16-3
- Oracle Coherence: Overview 16-4

Agenda	16-6
Types of Session Persistence	16-7
Coherence*Web	16-8
Coherence*Web and WebLogic Clusters	16-9
Coherence*Web Session Failover	16-10
Configuring Coherence*Web in WebLogic	16-11
Quiz	16-12
Practice 16-1 Overview: Configuring Coherence*Web	16-13
Agenda	16-14
Coherence and WebLogic Server	16-15
WebLogic Managed Coherence Servers Operations	16-16
What Is a Grid Archive (GAR)?	16-17
Coherence Application Deployment on WebLogic	16-18
Coherence Container: Benefits	16-19
Coherence Cluster	16-20
Managed Coherence Server	16-21
Quiz	16-23
Summary	16-24
Practice 16-2 Overview: Configuring Managed Coherence Servers	16-25

## **17 Application Staging and Deployment Plans**

Objectives	17-2
Agenda	17-3
Review: Java EE Applications	17-4
Review: Deployment	17-5
Agenda	17-6
Server Staging Modes	17-7
Staged Mode Example	17-8
No Stage Mode Example	17-9
External Stage Mode Example	17-11
Configuring the Staging Mode	17-13
Quiz	17-14
Agenda	17-15
Development to Test to Production	17-16
Examples of Deployment Descriptor Values That Can Change	17-17
Agenda	17-18
Java EE Deployment Descriptors	17-19
Annotations	17-20
appmerge and appc	17-21
Quiz	17-22
Agenda	17-23

What Is a Deployment Plan?	17-24
Using Deployment Plans for Different Environments	17-26
Deployment Plan Example	17-28
Staging Deployment Plans	17-31
Quiz	17-32
Agenda	17-33
Creating a Deployment Plan	17-34
Creating a New Deployment Plan	17-36
Using the Administration Console to Generate a Deployment Plan	17-37
Modifying and Saving Data to Create a New Plan	17-38
New Deployment Plan Shows Changed Values	17-39
Using an Existing Deployment Plan to Configure an Application	17-40
Using an Existing Deployment Plan	17-42
WLST createPlan	17-43
weblogic.PlanGenerator	17-44
Using Plan Generator	17-45
Oracle Enterprise Pack for Eclipse	17-46
Managing Deployment Plans	17-50
Summary	17-51
Practice 17-1 Overview: Creating and Using a Deployment Plan	17-52

## **18 Production Redeployment**

Objectives	18-2
Agenda	18-3
HTTP Sessions and Redeployment	18-4
Agenda	18-5
Redeployment Strategies	18-6
Agenda	18-7
Application Availability	18-8
What Is Production Redeployment?	18-9
Advantages of Production Redeployment	18-11
Production Redeployment Process	18-12
Application Retirement	18-14
Review: Administration Channel	18-15
Administration Mode	18-16
Distributing a Versioned Application	18-17
Deploying in Administration Mode	18-18
Rolling Back to the Previous Version	18-19
Quiz	18-20
Agenda	18-22
Redeployment Process: Overview	18-23

Configuring Application Deployment Versioning	18-24
Deploying a New Version of an Application	18-26
Distributing and Starting a Versioned Application in Administration Mode	18-30
Transitioning a Versioned Application from Administration Mode to Active	18-33
Rolling Back a Versioned Application to a Previous Version	18-35
Quiz	18-36
Agenda	18-37
Requirements and Restrictions	18-38
Summary	18-40
Practice 18-1 Overview: Using Production Redeployment	18-41

## **19 Oracle Cloud**

Agenda	19-2
What is Cloud?	19-3
What is Cloud Computing?	19-4
History – Cloud Evolution	19-5
Components of Cloud Computing	19-6
Characteristics of Cloud	19-7
Cloud Deployment Models	19-8
Cloud Service Models	19-9
Industry Shifting from On-Premises to the Cloud	19-13
Oracle IaaS Overview	19-15
Oracle PaaS Overview	19-16
Oracle SaaS Overview	19-17
Summary	19-18

## **20 Oracle Java Cloud Service Overview**

Objectives	20-2
Introducing Java Cloud	20-3
Java Cloud Service: Three Options	20-4
Java Cloud Service Main Use Cases	20-5
Java Cloud Service Feature: Provisioning	20-6
Java Cloud Service Feature: Patching	20-7
Java Cloud Service Feature: Backup / Restore	20-8
Java Cloud Service Feature: Scaling	20-9
Oracle Coherence Option: Data Caching & Scaling	20-10
Oracle Coherence Option: Your Cloud Data	20-11
How You Interact with Java Cloud Service	20-12
Speaking of Dev Environments... Developer Cloud Service	20-13
Java Cloud Service On-Premises!	20-14
Summary	20-15

## A WebLogic Optimizations for Exalogic

- What Is Exalogic? A-2
- What Is InfiniBand? A-4
- Exalogic Networks A-5
- Exalogic Machine Topology A-6
- InfiniBand Networking Concepts A-7
- Default Compute Node Network A-8
- Recommended FMW Topology on Exalogic A-9
- Exalogic Shared Storage Concepts A-10
- Recommended FMW Storage on Exalogic A-11
- WebLogic Server Optimizations: Overview A-12
- Creating a Cluster Replication Channel A-13
- Using SDP for Replication A-14
- Using Multiple Ports for Replication A-15
- Other Replication Optimizations A-16
- Oracle Exabus A-17
- Other WebLogic Optimizations A-18
- Enabling Other Optimizations A-19
- What Is Exadata? A-20
- Exadata InfiniBand Connectivity A-21
- Enabling SDP for Exadata Connectivity A-22

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.

# Working with the Security Realm

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to:

- Define users, groups, roles, and policies for the embedded LDAP server
- Configure auditing and role mapping



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- Security Review
  - Security Realm
  - Default Security Configuration
- The Embedded LDAP Authentication System
- Configure Auditing

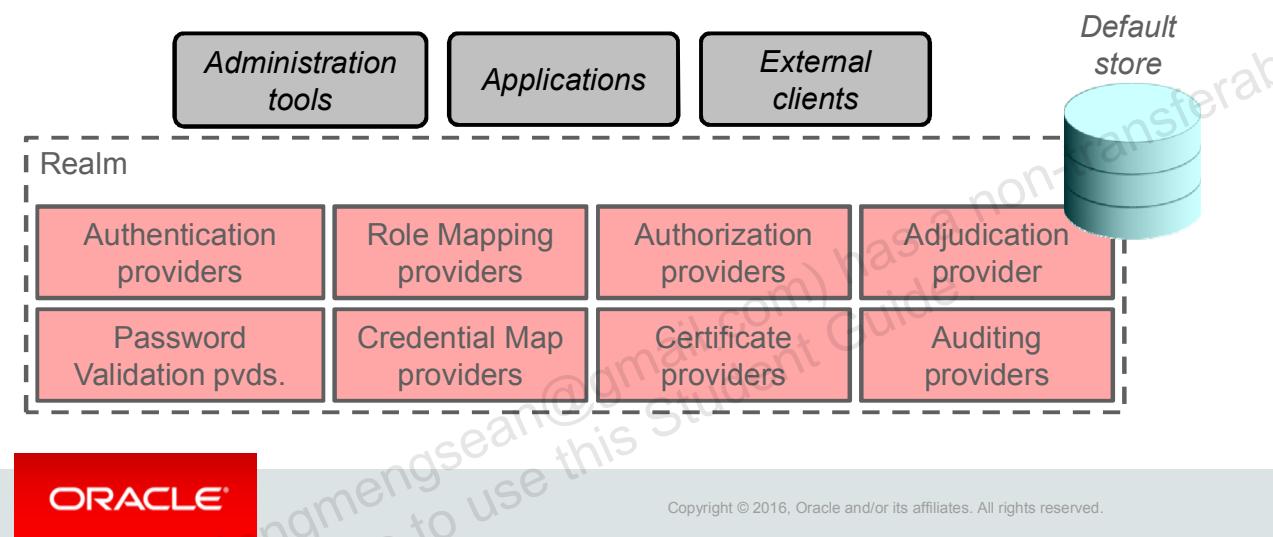


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Security Realm: Review

A security realm:

- Handles security logic and decisions for a domain
- Consists of a series of pluggable providers
- Is scoped to all servers in a domain



The security service in WebLogic Server simplifies the configuration and management of security while offering robust capabilities for securing your WebLogic Server deployment. Security realms act as a scoping mechanism. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. You can configure multiple security realms in a domain; however, only one can be the active security realm.

A security policy is an association between a WebLogic resource and one or more users, groups, or security roles. Security policies protect the WebLogic resource against unauthorized access. A WebLogic resource has no protection until you create a security policy for it.

A security provider store contains the users, groups, security roles, security policies, and credentials used by some types of security providers to provide their services. For example, an authentication provider requires information about users and groups; an Authorization provider requires information about security policies; a Role Mapping provider requires information about security roles; and a Credential Mapping provider requires information about credentials to use for remote applications. These security providers need this information to be available in a database to function properly.

An Adjudication provider determines what to do if access decisions of multiple Authorization providers do not agree on an answer. The Adjudication provider resolves authorization conflicts by weighing each access decision and returning a final result.

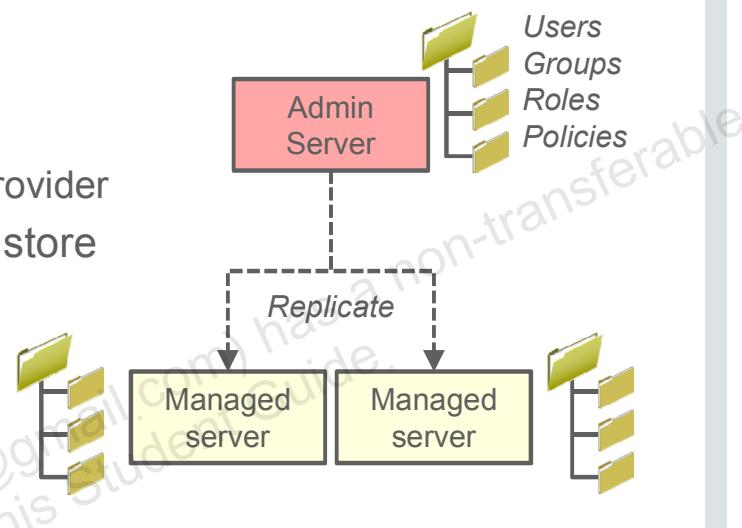
The Certificate Lookup and Validation providers complete certificate paths and validate X509 certificate chains, respectively. If multiple providers are configured, a certificate or certificate chain must pass validation with all of them in order for the certificate or certificate chain to be accepted.

An Auditing provider collects, stores, and distributes information about operating requests and the outcome of those requests for the purposes of nonrepudiation. An Auditing provider makes the decision about whether to audit a particular event based on specific audit criteria, including audit severity levels.

# Default Security Configuration

A new domain includes a default realm that:

- Includes some basic providers such as:
  - Default authenticator
  - Default identity asserter
  - XACML role mapper
  - XACML authorizer
  - Default adjudicator
  - Default certificate path provider
- Uses the embedded LDAP store



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To simplify the configuration and management of security, WebLogic Server provides a default security configuration. In the default security configuration, myrealm is set as the default (active) security realm, and the WebLogic Adjudication, Authentication, Identity Assertion, Credential Mapping, CertPath, EXtensible Access Control Markup Language (XACML) Authorization, and XACML Role Mapping providers are defined as the security providers in the security realm.

The WebLogic Server–embedded LDAP server for a domain consists of a master LDAP server (maintained in the domain’s Administration Server) and a replicated LDAP server (maintained in each managed server in the domain). When changes are made using a managed server, updates are sent to the embedded LDAP server on the Administration Server. The embedded LDAP server on the Administration Server maintains a log of all changes. It also maintains a list of managed servers and the current change status for each one. The embedded LDAP server on the Administration Server sends appropriate changes to each managed server and updates the change status for each server. This process occurs when an update is made to the embedded LDAP server on the Administration Server. However, depending on the number of updates, it may take several seconds or more for the change to be replicated to the managed server. The default interval is every thirty seconds.

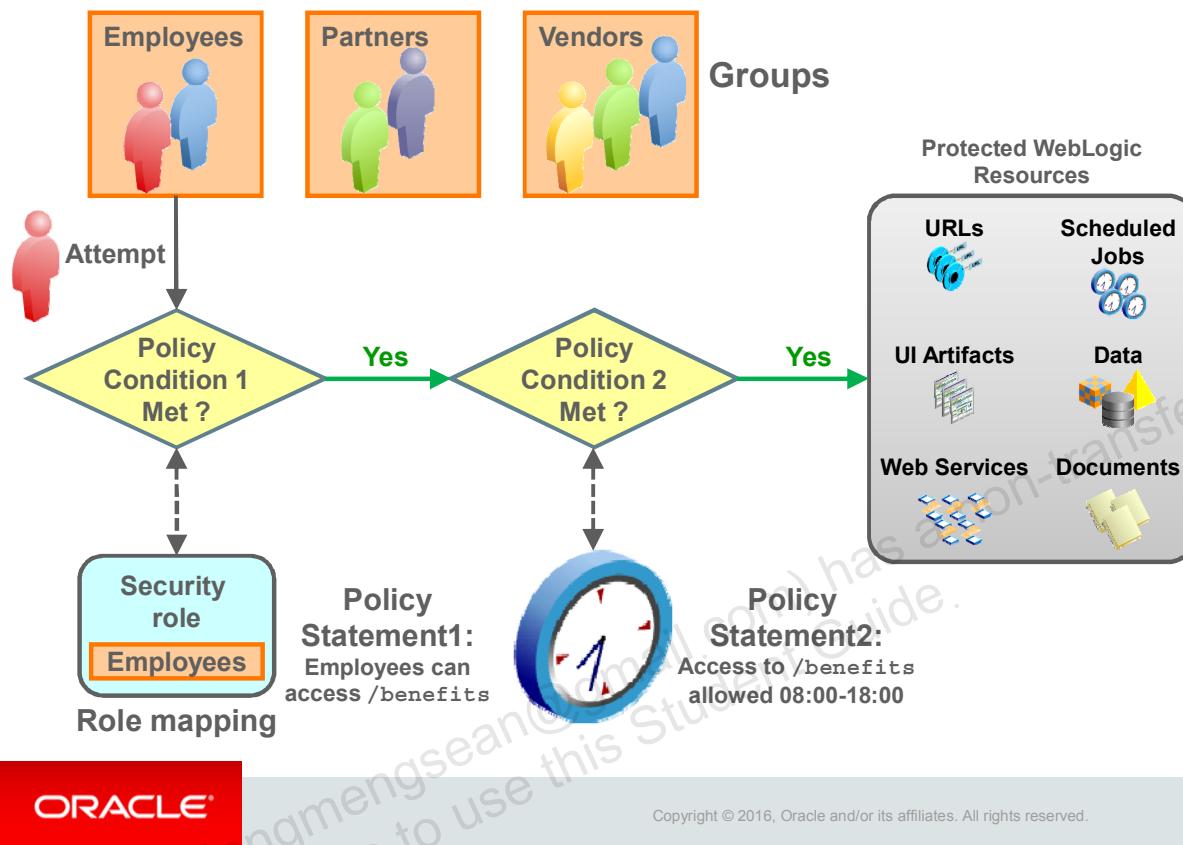
# Agenda

- Security Review
- The Embedded LDAP Authentication System
  - How WebLogic Resources Are Protected
  - Users, Groups, Roles, and Policies
  - Configuring Users, Groups, Roles, and Policies
  - Configuring the Embedded LDAP server
- Configure Auditing



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# How WebLogic Resources Are Protected



This slide represents the big picture of how WebLogic security artifacts work together to protect WebLogic resources.

- First, you create users and groups and statically assign users to groups that represent organizational boundaries.
- Then you create a security role based on your established business procedures. The security role consists of one or more conditions that specify the circumstances under which a particular user, group, or other role should be granted the security role.
- At run time, the WebLogic Security Service compares groups against role conditions to determine whether users in the group should be granted a security role. This process of matching groups to roles is called role mapping.
- Next, you create a security policy based on your established business procedures. The security policy consists of one or more policy conditions that specify the circumstances under which a particular security role should be granted access to a WebLogic resource.
- At run time, the WebLogic Security Service uses the security policy to determine whether access to the protected WebLogic resource should be granted. Only users who are members of the group that is granted the security role can access the WebLogic resource.

## Examples of WebLogic Resources to Protect

Policies that are more specific always override those that are more general.

Resource	Privileges
JDBC Data Source	<ul style="list-style-type: none"><li>• admin</li><li>• reserve</li><li>• reset</li><li>• shrink</li></ul>
JMS Destination	<ul style="list-style-type: none"><li>• All operations</li><li>• Send</li><li>• Receive</li><li>• Browse</li></ul>
JNDI Node	<ul style="list-style-type: none"><li>• Lookup</li><li>• List</li><li>• Modify</li></ul>



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows some examples of WebLogic resources and privileges that you can protect with security policies.

All the resources within a Java EE application or module that you deploy exist within a hierarchy, and policies on resources higher in the hierarchy act as default policies for resources lower in the same hierarchy. Policies lower in a hierarchy always override policies higher in the hierarchy.

# Users and Groups

Users are entities that use WebLogic, such as:

- Application end users
- Client applications
- Other WebLogic Servers



Groups are:

- Logical sets of users
- More efficient for managing a large number of users



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Users are entities that can be authenticated in a security realm. A user can be a person (such as an application end user), or a software entity (such as a client application or other instances of WebLogic Server). As a result of authentication, a user is assigned an identity or a principal. Each user is given a unique identity within the security realm.

Users can be placed into groups that are associated with security roles or be directly associated with security roles.

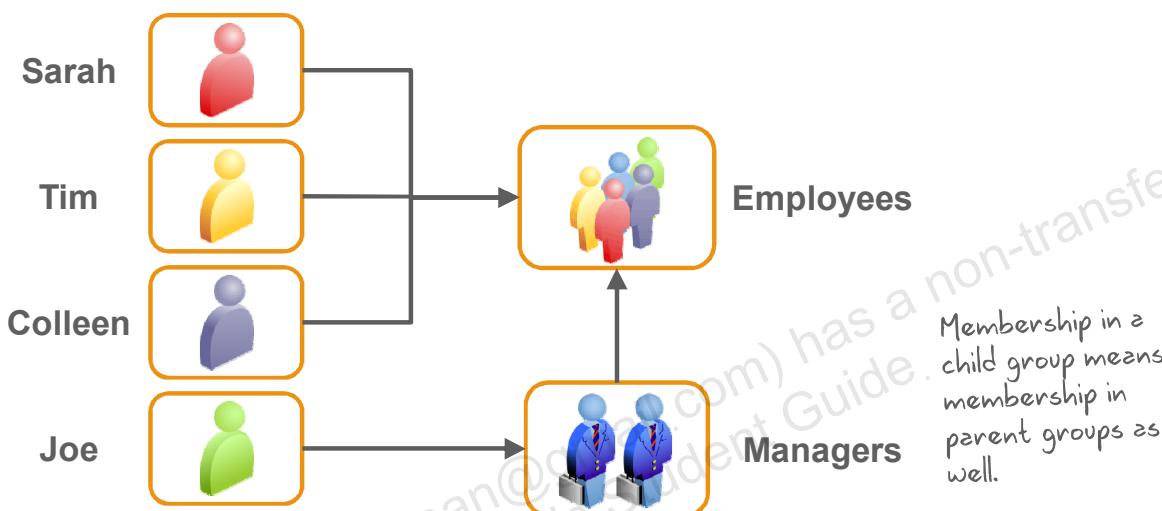
When users want to access WebLogic, they present proof material (such as a password or a digital certificate) to the authentication provider configured in the security realm. If WebLogic can verify the identity of the user based on that username and credential, then it associates the principal assigned to the user with a thread that executes code on behalf of the user. WebLogic then checks the security policy of the requested WebLogic resource to make sure that the user has the required permissions before the thread begins executing code.

A person can be defined as both an individual user and a group member. Individual-access permissions override any group member-access permissions. WebLogic evaluates each user by first looking for a group and testing whether the user is a member of the group and then looking for the user in the list of defined users.

## Group Membership

WebLogic allows you to organize groups in various ways:

- Groups can contain users.
- Groups can contain other groups.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

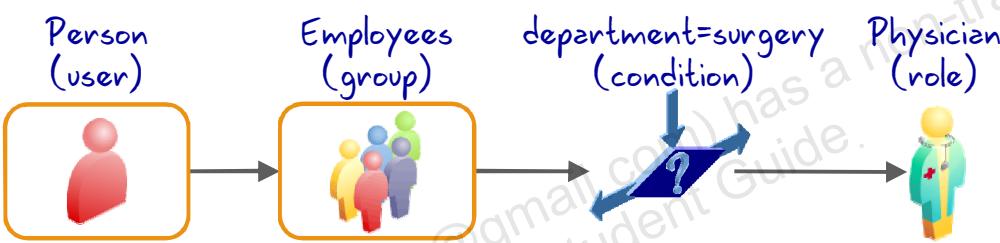
Groups can be organized in arbitrary ways, thereby providing greater flexibility. In the example in the slide, all the users (Sarah, Tim, Colleen, and Joe) are members of the Employees group. Joe is also a member of the Managers group. All Managers are also Employees.

Managing groups is more efficient than managing large numbers of users individually. For example, an administrator can specify permissions for 50 users at one time if those 50 users belong to the same group. Usually, group members have something in common. For example, a company may separate its sales staff into two groups: Sales Representatives and Sales Managers. This is because staff members have different levels of access to WebLogic resources depending on their job descriptions.

WebLogic supports assigning users to groups. Each group shares a common set of permissions that govern its member users' access to resources. You can mix group names and usernames whenever a list of users is permitted.

## Roles

- Roles classify a set of users in an application or enterprise who have the same permissions.
- Users and groups can be granted multiple roles.
- The two types of roles are:
  - Global scoped
  - Resource scoped



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A role definition is specific to a security realm. Roles are tightly coupled with the authorization provider. A role can be defined as resource scoped or global scoped. A resource-scoped role means that it is granted to a specific resource, such as a method of an EJB or a branch of the JNDI tree. Most roles are scoped. Global-scoped role means that a role is granted to all the resources in a security realm. WLS defines a set of default global roles for protecting the WebLogic resources. Roles are assigned statically by defining them in the deployment descriptor for specified users or groups, or dynamically through the administration console by defining a set of conditions.

The global roles that are available by default are the following: AppTester, Anonymous, Admin, Operator, Deployer, and Monitor.

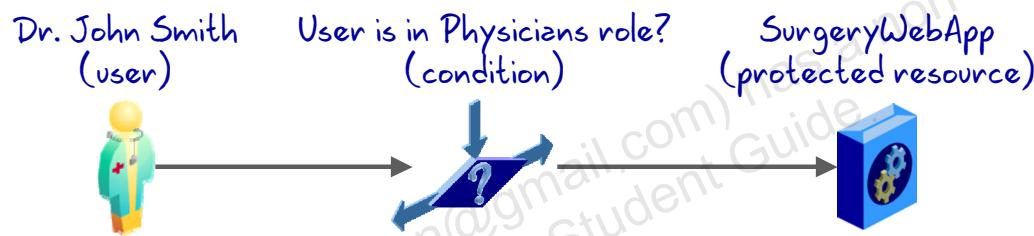
Roles that are defined in deployment descriptors can be inherited:

- During deployment of an application, security roles defined within an application may get migrated into the configured LDAP server.
- It is possible to configure or disable this feature.

You can manage role definitions and assignments without editing deployment descriptors or redeploying the application.

## Policies

- Policies are a set of related data points that are evaluated to determine what privileges are granted or denied to a user.
- Policies are global scoped or resource scoped.
- Data points can include:
  - Users, groups, or roles
  - Access times and context field values

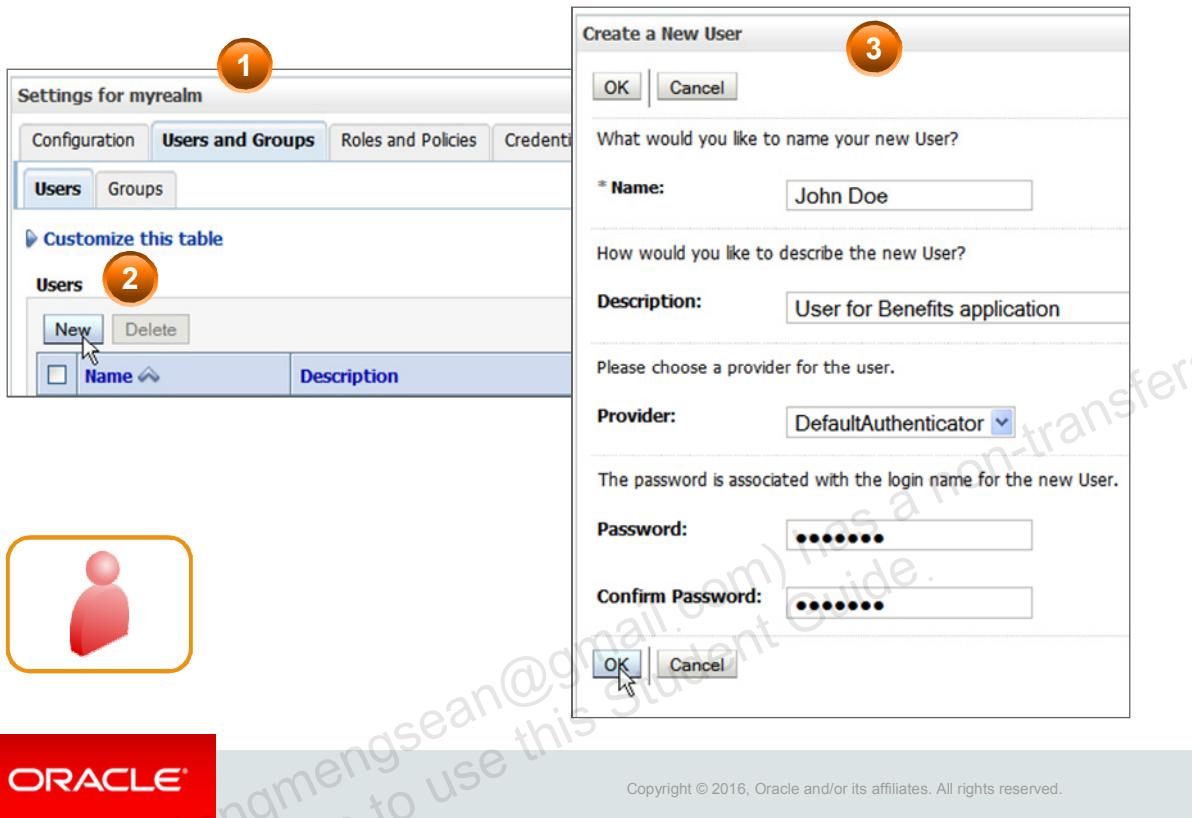


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A policy is a rule that is used to determine if a user has privileges to access a WebLogic resource.

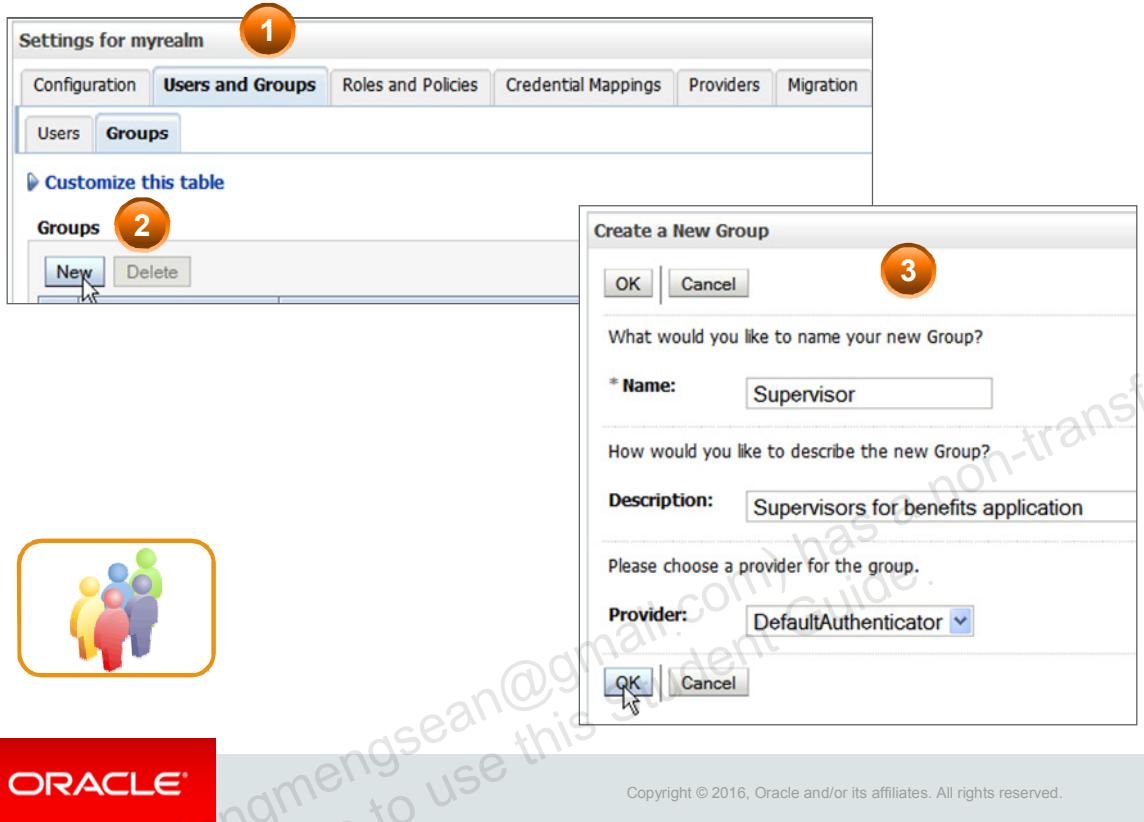
# Configuring New Users



To configure a new user, perform the following steps:

1. Access Security Realms and select your security realm in the Realms Table on the *Summary of Security Realms* page. Click the *Users and Groups > Users* tab for your realm.
2. Click New in the Users table.
3. Enter the necessary details in the *Create a New User* dialog box and click OK. The name may contain spaces, but other systems may not allow spaces. Best practice is to reserve the use of spaces for the description and use underscores if needed for the name.

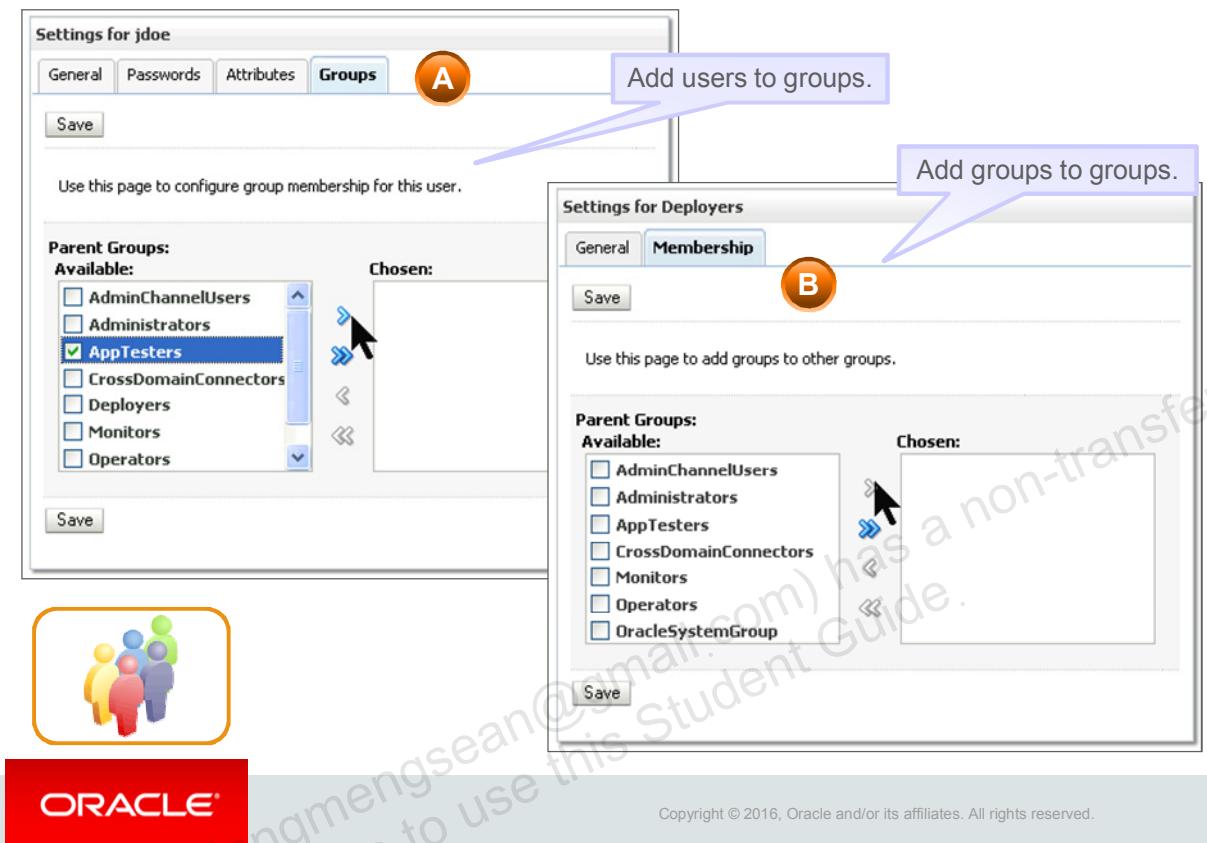
## Configuring New Groups



To configure a new group, perform the following steps:

1. Access Security Realms and select your security realm in the Realms Table on the *Summary of Security Realms* page. Click the *Users and Groups > Groups* tab for your realm.
2. Click New in the Groups table.
3. Enter the necessary details in the *Create a New Group* dialog box and click OK.

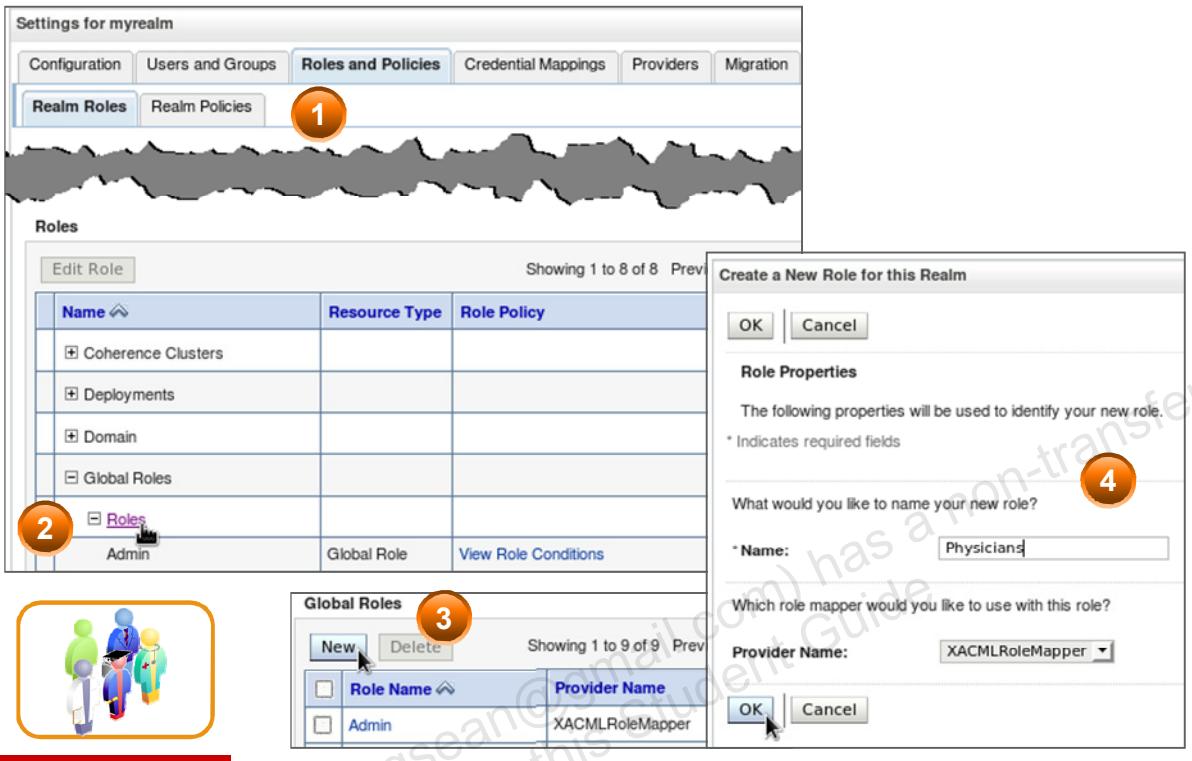
# Configuring Group Memberships



Each group has two types of membership.

- A. You can configure a user to be a member of a group as follows:
  1. Navigate to the Users subtab under the *Users and Groups* tab of the security realm.
  2. Select the user for whom you want to configure the group membership.
  3. Click the Groups tab on the *Settings for <user>* page.
  4. Select the group from the Available list and click **>** to move it to the Chosen list. Then click Save.
- B. You can configure a group to be a member of another group as follows:
  1. Navigate to the Groups subtab under the *Users and Groups* tab of the security realm.
  2. Select the group that you want to configure as a child of another group.
  3. Click the Membership tab on the *Settings for <group>* page.
  4. Select the parent group from the Available list and click **>** to move it to the Chosen list. Then click Save.

# Configuring New Roles



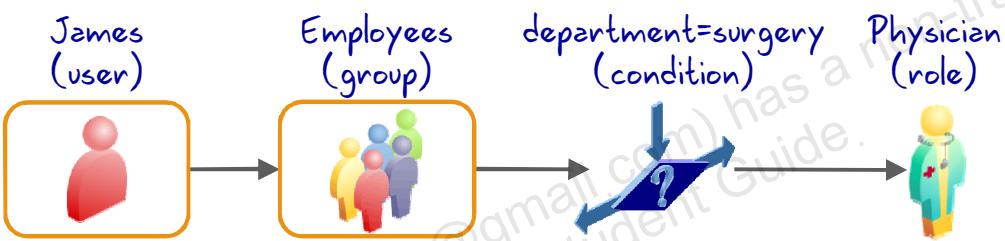
To create a global security role in WebLogic by using the administration console:

1. Navigate to the active security realm and click *Roles and Policies > Realm Roles* to display the list of configured roles.
2. Expand *Global Roles* and select *Roles* to display the list of configured global roles.
3. Click *New* to create a new role.
4. Enter a name for the role, select the role mapping provider (in this case, the default XACML RoleMapper provider), and click *OK*.

## What Is Role Mapping?

Role mapping is the process of aligning a user into a security role which is used to determine access to resources.

- Static and dynamic role mapping are available.
- Users are potentially mapped to multiple roles.
- Mapping may include analyzing data points above and beyond simple group membership.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Role mapping provides a way to categorize an application's users into application-specific identities based on either group membership or some conditional expression. The example in this slide shows an evaluation of a role mapping scenario. This particular role mapping scenario declares that if a user is a member of the Employees group and the context value, department, is surgery, then grant this user the Physician role. If James is a member of the Employees group and his department value is surgery, then he is granted the Physician role, and is subsequently granted access to any resources that Physicians are entitled to access.

# Configuring Role Mapping

The screenshot shows the 'Global Roles' page in the Oracle WebLogic Administration Console. On the left, a list of global roles is displayed, with 'Physicians' highlighted and circled with orange number 1. In the center, the 'Role Conditions' configuration page for the 'Physicians' role is shown. The 'Name:' field is set to 'Physicians'. Below it, a note states: 'These conditions determine membership in the role.' The 'Role Conditions' section contains a single condition: 'Group : Employees' (unchecked) and 'And' (selected). Underneath, there is a checkbox for 'Context element's value equals a string constant : department, surgery' (unchecked). At the bottom of the page is a 'Save' button. The entire interface has a watermark: 'Sean Taysiangmengsean@gmail.com has a non-transferable license to use this Student Edition.'

Continuing from where you left off after creating a new role using the administration console:

1. Select the Physician role you created to display the Edit Global Role page.
2. Click *Add Conditions* to specify the criteria used to determine if a user is granted the role.
3. Using the Role Conditions Wizard, you create the conditions that are used to map users to your role. In this case, if a user belongs to the Employees group and his or her department context variable is surgery, then the user is granted the role. The context element's value is based on HTTP Servlet Request attributes, HTTP Session Attributes, or EJB method parameters. During role evaluation, WebLogic retrieves this information from the `ContextHandler` object and compares it to the role mapping condition.

# Configuring Roles Using WLST

Creating roles:

```
rm = cmo.getSecurityConfiguration().getDefauleRealm() +  
      'RoleMappers/XACMLRoleMapper')  
i=0  
print 'creating roles'  
roles = ['user', 'creator']  
groups = ['AuctionUsers', 'AuctionCreators']  
for role in roles:  
    try:  
        print 'creating role: ' + role  
        expression = 'Grp(' + groups[i] + ')'  
        rm.createRole(None, role, None)  
        rm.setRoleExpression(None, role, expression)  
        i += 1  
        print ''  
    except Exception, e:  
        print 'Role creation exception:'  
        print e  
        i += 1  
        print ''
```

Acquire the role mapping provider Mbean.

Loop through roles and match role to proper group[i]

Build the role expression, setting role if user is a member of group[i]

Create the role.

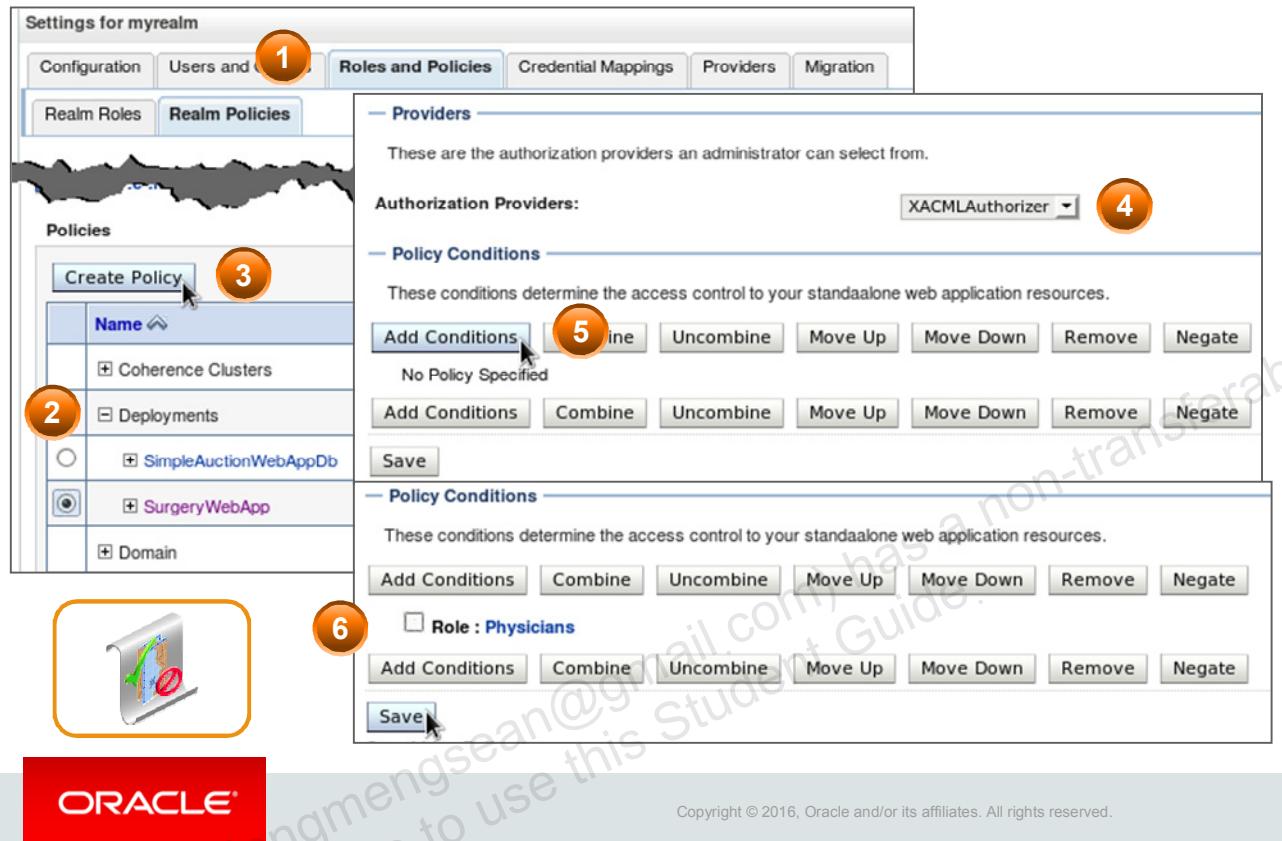
Add the expression to the role.

CreateRole.py



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Configuring New Policies



To create an authorization policy in WebLogic by using the administration console:

1. Navigate to the active security realm and click *Roles and Policies* > *Realm Policies* to display the list of configured policies.
2. Expand the appropriate scope to display the list of configured roles for that level. If applicable, select the entity (such as an application) that you want to protect with a policy.
3. Click *Create Policy* to create a new policy.
4. Select the authorization provider from a list of configured providers for the realm.
5. Use the buttons in the Policy Conditions section to create the conditions that comprise the policy itself.
6. The policy conditions are displayed as they are created. When the policy is complete, you save the changes and the policy is in effect. In this case, the policy protects access to the **SurgeryWebApp** application allowing access only to users in the **Physicians** role.

# Configuring Policies Using WLST

Creating a policy:

```
authz=cmo.getSecurityConfiguration().getDefauleRealm() +  
      '/Authorizers/XACMLAuthorizer')  
  
#URLResource(application, context_path, pattern, httpMethod, transport)  
r=weblogic.security.service.URLResource('SimpleAuctionWebAppDbSec',  
                                         '/SimpleAuctionWebAppDbSec',  
                                         '/createAuction.jsp', None, None)  
  
resourceID=r.toString()  
expression='Rol(creator)'  
  
try:  
    authz.createPolicy(resourceID, expression)  
    print ''  
except Exception, e:  
    print 'Policy creation exception:'  
    print e  
    print ''
```

Acquire the authorization provider Mbean.

Create the string version of the resource.

Create the expression for the policy to check if the user is in the creator role.

Create a URL pattern-based resource to protect.

Create the policy using the resource and expression.

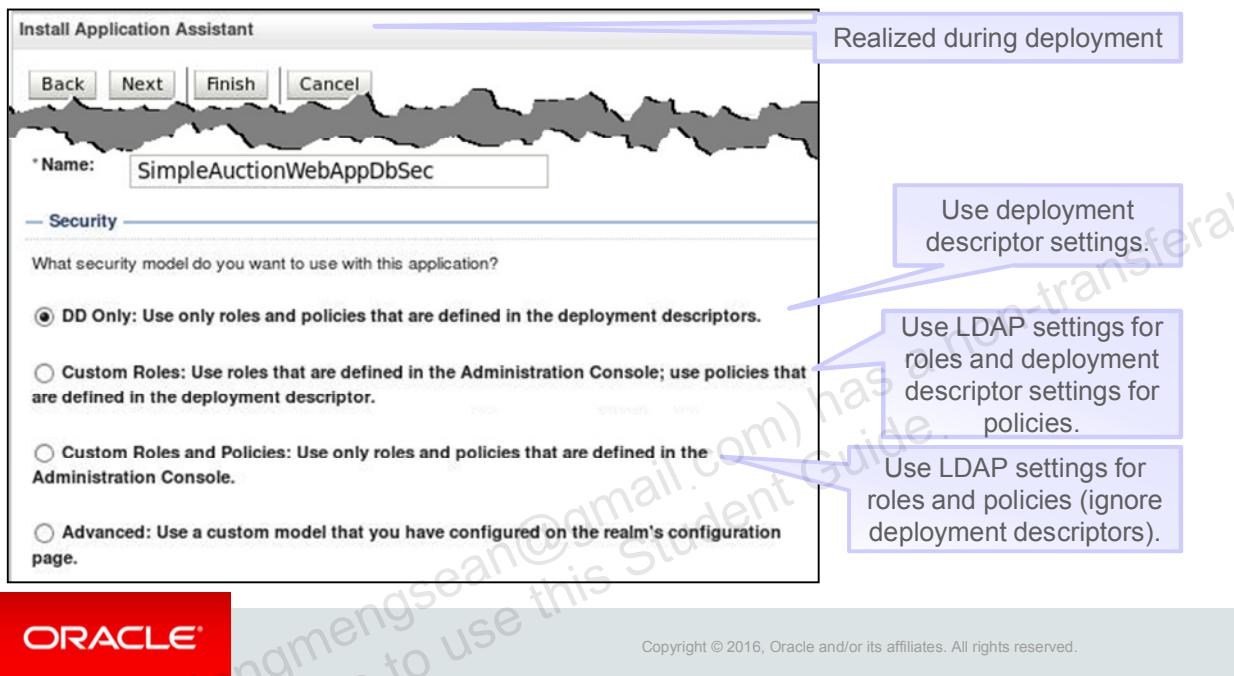
CreatePolicy.py



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Security Configuration Sources

Security roles, role mapping, and policies are configured either in the embedded LDAP store or an application's deployment descriptor.



## Configuring Sources Using WLST and weblogic.Deployer

WLST:

```
> deploy(appName='SimpleAuctionWebAppDbSec',
    path='/apps/SimpleAuctionWebAppDbSec.war',
    targets='cluster1',
    securityModel='[DDOnly | CustomRoles |
        CustomRolesAndPolicies | Advanced')
```

WLST Command Line

weblogic.Deployer:

```
$ java weblogic.Deployer
    -adminurl http://host01.example.com:7001 -username weblogic
    -password Welcome1 -deploy -targets cluster1
    -securityModel [ DDOnly | CustomRoles |
        CustomRolesAndPolicy | Advanced]
    /deployments/SimpleAuctionWebAppDbSec.war
```

Command Line



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Deployment Descriptor Security Example: weblogic.xml

Security roles are defined in the weblogic.xml file and policies are defined in the web.xml file.

```
<security-role-assignment>
  <role-name>user</role-name>
  <principal-name>AuctionUsers</principal-name>
</security-role-assignment>

<security-role-assignment>
  <role-name>auctionCreators</role-name>
  <principal-name>AuctionCreators</principal-name>
</security-role-assignment>
```

Users that are in the AuctionUsers group are mapped to the user role.

Users that are in the AuctionCreators group are mapped to the auctionCreators role.

weblogic.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Deployment Descriptor Security Example: web.xml

```
<security-constraint>
  <display-name>create auction</display-name>
  <web-resource-collection>
    <web-resource-name>createAuction</web-resource-name>
    <url-pattern>/createAuction.jsp</url-pattern>
    <url-pattern>/CreateAuctionServlet</url-pattern>
  </web-resource-collection>

  <auth-constraint>
    <role-name>auctionCreators</role-name>
  </auth-constraint>
</security-constraint>

<security-role>
  <role-name>user</role-name>
</security-role>
<security-role>
  <role-name>auctionCreators</role-name>
</security-role>
```

These URL resources are accessed only by users in the auctionCreators role.

References security roles that are defined in the weblogic.xml file

web.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Embedded LDAP Server

- In WebLogic, users, groups, and authorization information can be stored in the embedded LDAP server.
- Several properties can be set to manage the LDAP server, including:
  - Credentials
  - Backup settings
  - Cache settings
  - Replication settings



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The embedded LDAP server is used as a storage mechanism for WebLogic authentication, authorization, role mapping, and credential mapping providers.

Information from these providers is stored and updated in the administration server and replicated to all the managed servers in the domain. The read operations performed by WebLogic security providers (when running on a managed server) access the local, replicated embedded LDAP server. The write operations access the master embedded LDAP server on the administration server and any updates are replicated to all the managed servers in the domain. If the administration server is not running, operations executed by WebLogic security providers that write to the embedded LDAP server (for example, adding new users, groups, or roles, or adding resources) are not possible.

# Configuring the Embedded LDAP Server

The screenshot shows the 'Settings for wladmin' page with the 'Security' tab selected. Under the 'General' tab, there are fields for 'Credential' (a password), 'Confirm Credential' (a confirmation of the password), 'Backup Hour' (set to 23), 'Backup Minute' (set to 5), and 'Backup Copies' (set to 7). On the right side, under the 'Cache' tab, 'Cache Enabled' is checked. Other cache-related settings include 'Cache Size' (32), 'Cache TTL' (60), and options for 'Refresh Replica At Startup', 'Master First', 'Timeout' (0), and 'Anonymous Bind Allowed'.

Back up copies are stored in zip files in  
\$DOMAIN\_HOME/servers/AdminServer/data/ldap/backup



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- **Credential:** The credential (usually password) that is used to connect to the embedded LDAP server. This password is encrypted. The default is randomly generated and must be changed if you want to use an LDAP client to connect to the server.
- **Backup Hour:** The hour at which to back up the embedded LDAP server. Minimum is 0, Maximum is 23, and Default is 23.
- **Backup Minute:** The minute at which to back up the embedded LDAP server. This attribute is used with the `Backup_Hour` attribute to determine the time at which the embedded LDAP server is backed up. Minimum is 0, Maximum is 59, and Default is 05.
- **Backup Copies:** The number of backup copies of the embedded LDAP server. Minimum is 0, Maximum is 65534, and Default is 7.
- **Cache Enabled:** Whether or not a cache is used for the embedded LDAP server. The default is True.
- **Cache Size:** The size of the cache (in KB) that is used with the embedded LDAP server. Minimum is 0 and Default is 32.
- **Cache TTL:** The time-to-live (TTL) of the cache in seconds. Minimum is 0 and Maximum is 60.

- **Refresh Replica At Startup:** Whether or not a managed server should refresh all replicated data at boot time. This is useful if you made a large number of changes when the managed server was not active and you want to download the entire replica instead of having the administration server push each change to the managed server. The default is false.
- **Master First:** The connections to the master LDAP server should always be made instead of connections to the local replicated LDAP server. The default is false.
- **Timeout:** Specifies the maximum number of seconds to wait for results from the embedded LDAP server before timing out
- **Anonymous Bind Allowed:** Specifies whether the embedded LDAP server should allow anonymous connections

## Quiz



A security policy is used to:

- a. Map a user to a role
- b. Map a group to a role
- c. Make a user a member of a group
- d. Authorize if a user has access to a resource

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: d**

## Practice 11-1 Overview: Creating Users, Groups, Roles, and Policies

This practice covers the following topics:

- Creating users and groups
- Adding users to groups
- Mapping users to roles
- Authorizing access to a resource with a policy



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

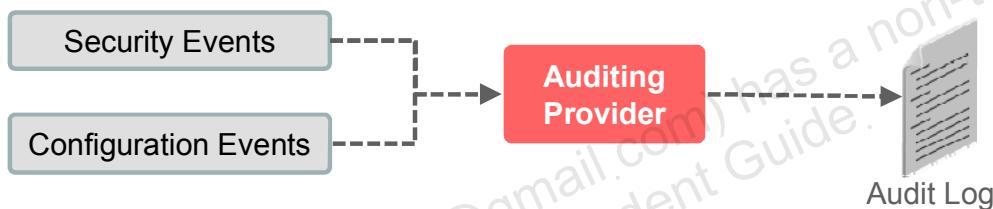
- Security Review
- The Embedded LDAP Authentication System
- Configure Auditing
  - Auditing
  - Security Audit Events
  - WebLogic Auditing Architecture
  - Custom Versus Default Auditing Provider
  - Configuring the Default Auditing Provider
  - Configuration Auditing



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Auditing

- Auditing provides a trail of user activity related to security and configuration changes.
- The default auditing provider records event data that is associated with security requests and the outcome of the requests.
- Auditing can also create a record of all domain configuration changes.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic provides an auditing provider to collect, store, and distribute information about requests and the outcome of those requests for nonrepudiation. If you are not familiar with the term, nonrepudiation means that someone is not able to refute, or deny, that they performed an action by using a system. This is because their actions are recorded, or audited. An auditing provider decides whether to audit a particular event based on specific audit criteria, including audit severity levels. The audit information is written to output repositories, such as a simple file.

You can configure multiple auditing providers in a security realm, but none are required.

You can set the location of the audit log file by setting the  
-Dweblogic.security.audit.auditLogDir property on the WebLogic Server command line.

## Sample Auditing Output

The default location of the audit log is:

DOMAIN\_HOME/yourServer/logs/DefaultAuditRecorder.log

```
##### Audit Record Begin <Apr 3, 2013 5:51:28 PM><Severity =INFORMATION> <<<Event  
Type = RoleManager Audit Event ><Subject: 0  
><<jdbc>><type=<jdbc>, application=, module=, resourceType=ConnectionPool,  
resource=jdbc/AuctionDB, action=reserve>>>> Audit Record End #####  
##### Audit Record Begin <Apr 3, 2013 5:51:28 PM> <Severity =SUCCESS> <<<Event  
Type = Authorization Audit Event V2 ><Subject: 0  
><ONCE><<jdbc>><type=<jdbc>, application=, module=, resourceType=ConnectionPool,  
resource=jdbc/AuctionDB, action=reserve>>>> Audit Record End #####
```

DefaultAuditRecorder.log



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Security Audit Events

- Typical security events include:
  - Attempting an authentication or identity assertion
  - Creation of a new role or policy
  - Locking out or unlocking a user account
- Security events have the following characteristics:
  - Name
  - Severity (WARNING, ERROR, SUCCESS, and so on)
  - Zero or more context attributes:
    - Protocol, port, address
    - HTTP headers
    - EJB method parameters
    - SAML tokens



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

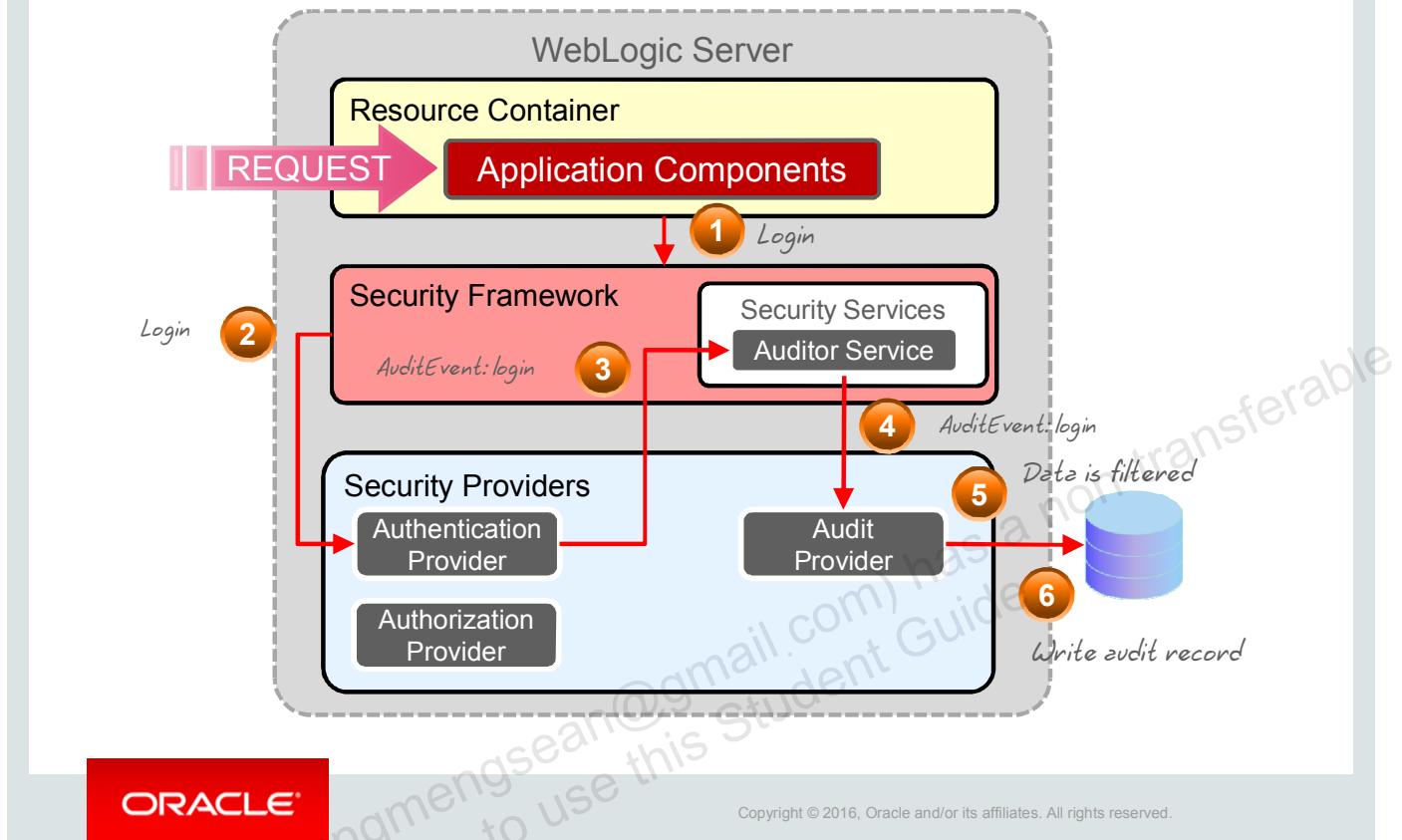
In addition to the events listed in the slide, the default WebLogic auditing provider records the following types of security events:

- When the lockout on a user account expires
- A security policy is used and an authorization decision is made.
- A role definition is used.
- A role or policy is removed or “undeployed.”

The WebLogic auditing provider audits security events of the specified severity and higher. The severity levels, in order from lowest to highest, are: INFORMATION, WARNING, ERROR, SUCCESS, FAILURE. You can also set the severity level to CUSTOM, and then enable the specific severity levels that you want to audit, such as ERROR and FAILURE events only.

An audit event includes a context object that can hold a variety of different attributes, depending on the type of event. When you configure an auditing provider, you specify which context attributes are recorded for each event. By default, no context attributes are audited.

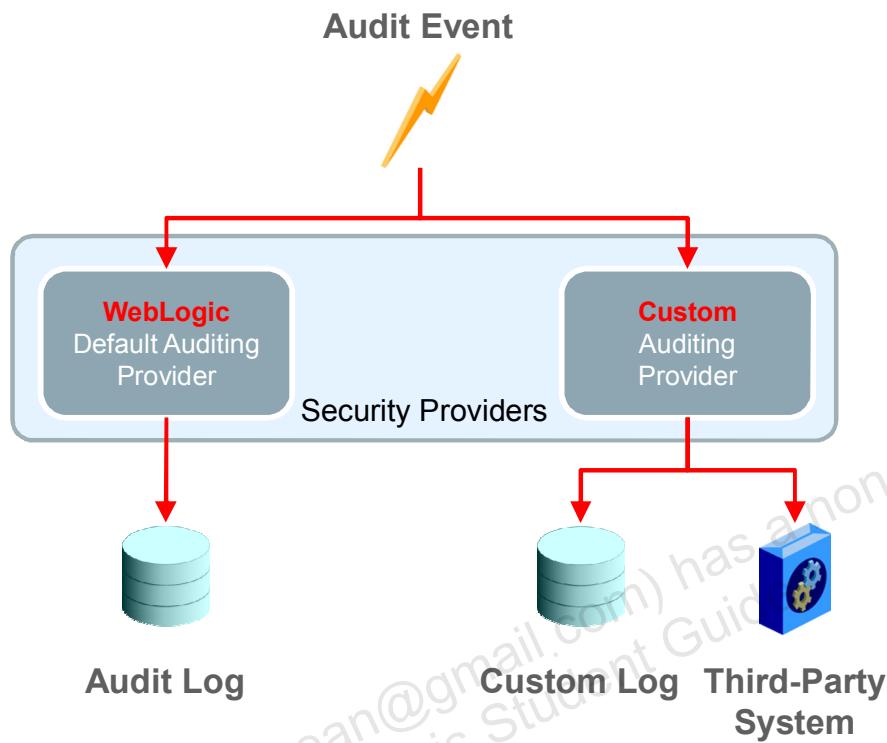
# WebLogic Auditing Architecture



This slide shows an example of the process architecture of WebLogic auditing. Security providers are implemented to post audit events for certain activities that occur as part of the authentication and authorization process. Those events are handled by the Auditor Service, which passes them on to the Audit Provider for processing.

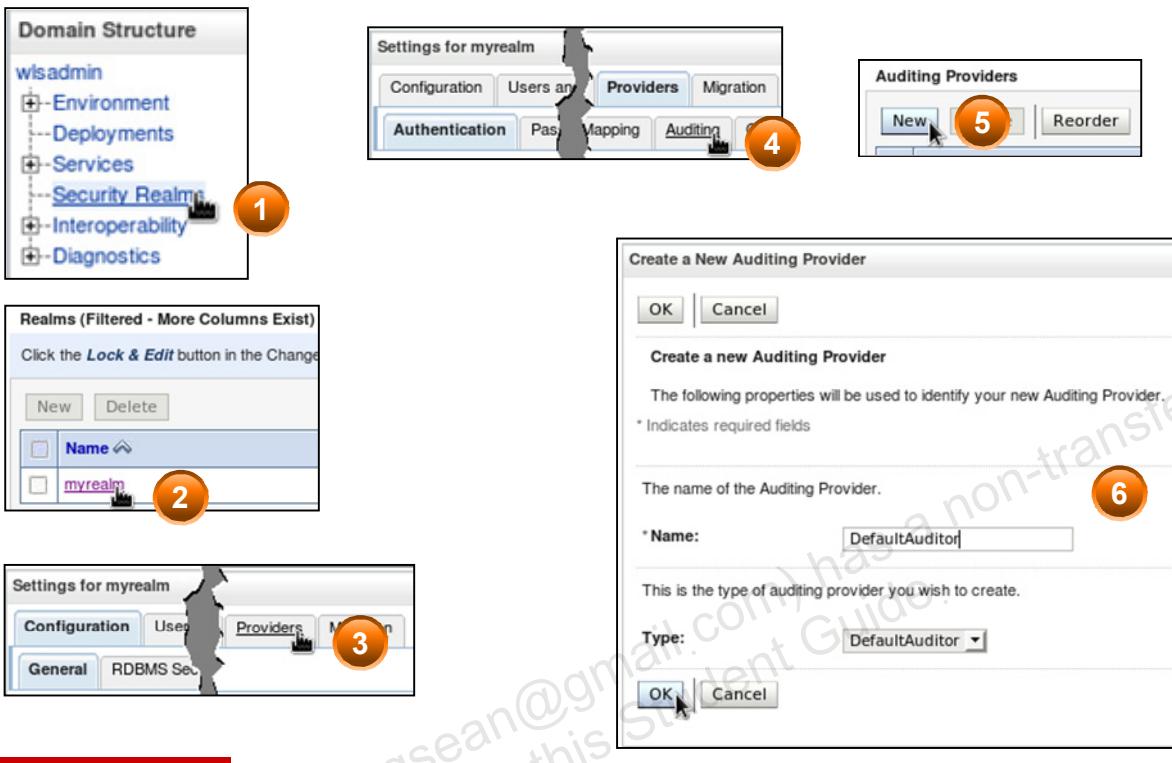
1. A request for an application comes into the server and the user is logging in.
2. The login request is handled by the authentication provider.
3. The authentication provider posts the login audit event as part of its processing.
4. The Auditor Service receives the login audit event and sends it to the Audit Provider for processing.
5. The Audit Provider receives the login audit event, compares it to the current settings that are configured for auditing, and applies all settings to determine whether or not an audit record is written.
6. When an audit event meets the configured criteria for auditing, the audit record is written to the log file.

# Custom Versus Default Auditing Provider



The default WebLogic auditing provider posts audit events and logs audit records to a file. If your application has different or enhanced auditing requirements, you can create a custom audit provider that provides for those requirements. You can configure multiple audit providers in WebLogic, thus allowing each provider an opportunity to audit events.

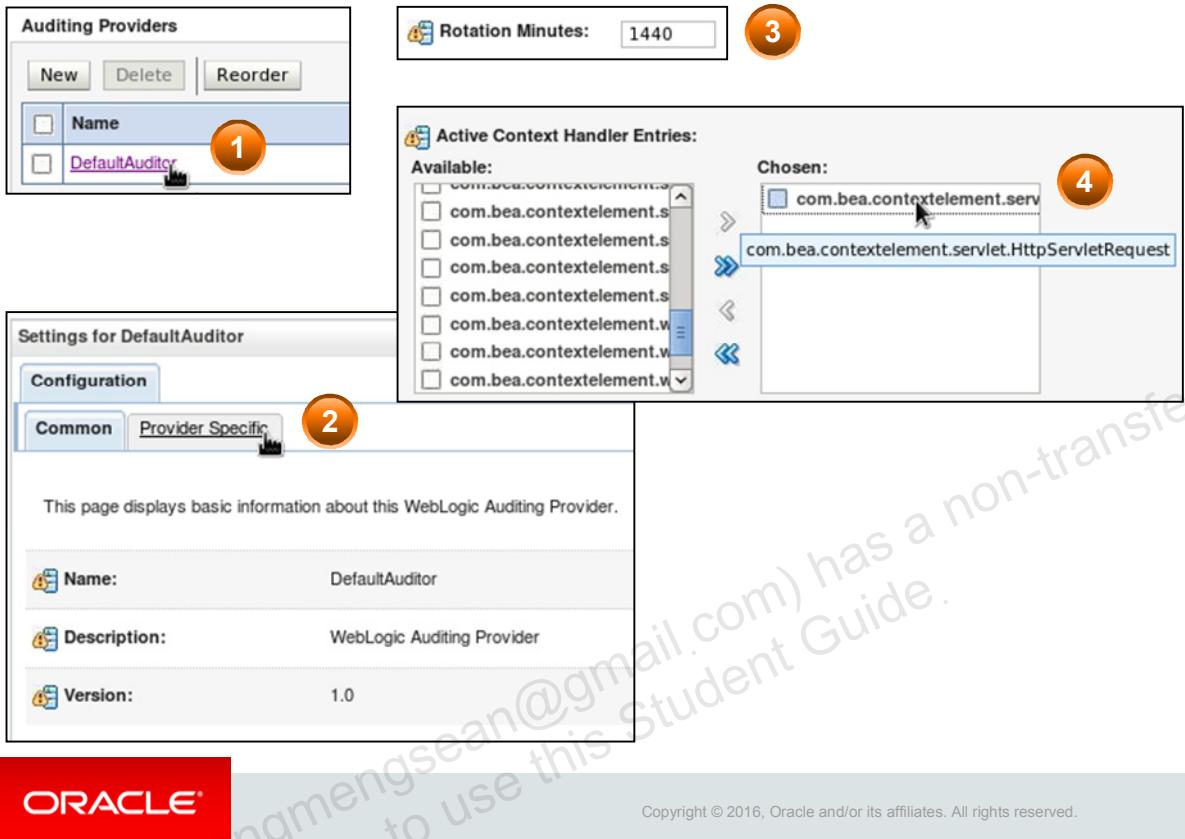
# Creating the Default Auditing Provider



You create a WebLogic default audit provider by performing the following steps using the administration console:

1. Click Security Realms in the Domain Structure panel.
2. Select the active realm of the domain.
3. Click the Providers tab to display the configuration tabs for all security providers.
4. Click the Auditing tab to list the currently configured audit providers.
5. Click New to create a new audit provider.
6. The Create a *New Auditing Provider* page is displayed:
  - **Name:** Enter a unique name for your provider.
  - **Type:** The only type available is DefaultAuditor because no custom providers have been configured.
  - Click OK.

# Configuring the Default Auditing Provider



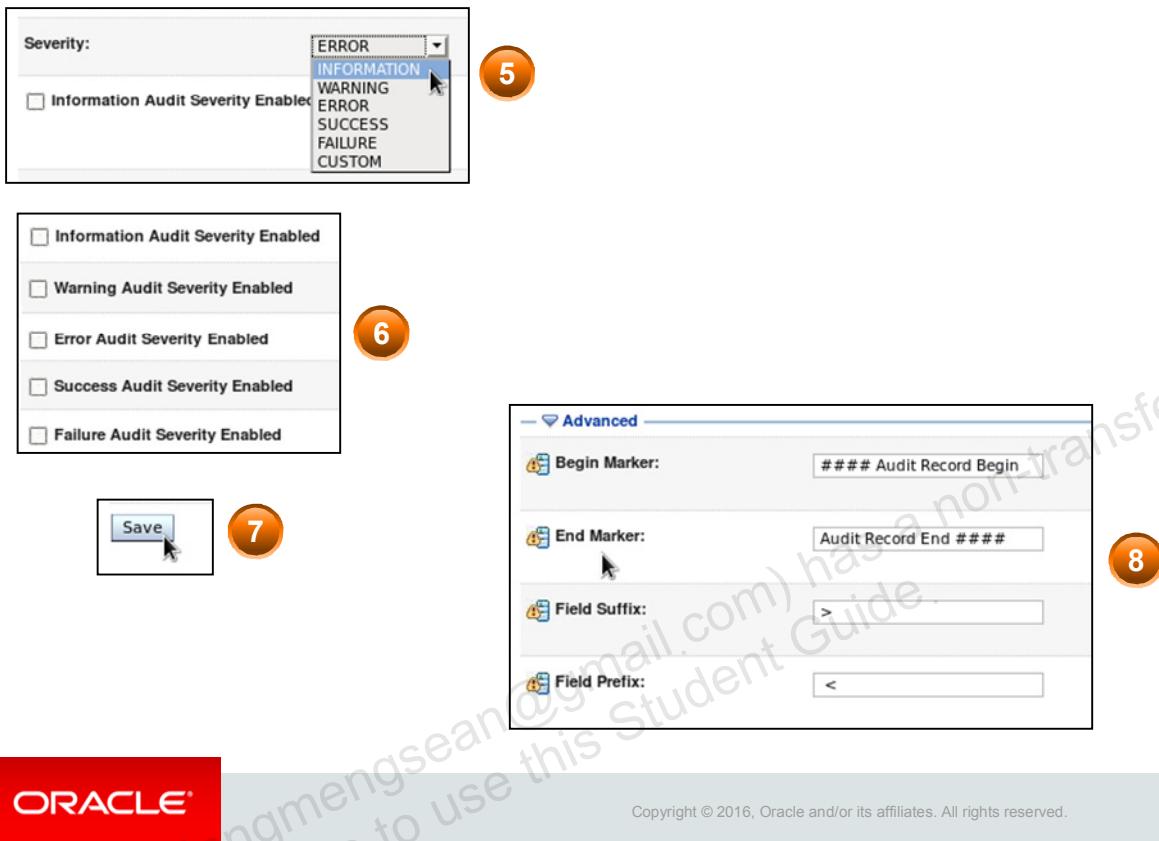
After the audit provider is created, you configure how it functions:

1. Select your newly created audit provider.
2. Click the Provider Specific tab to display the configuration options available for the provider.
3. Configure the number of minutes WebLogic uses to determine when to back up the current audit log file and create a new one for more audit records.
4. Optionally, select from several server context handlers available to include data related to the audited action in the audit record. Objects in the context handler are usually logged by using the `toString()` method of the object.

An audit event includes a ContextHandler that can hold information that is related to the audited request. Some examples include:

- `com.bea.contextelement.servlet.HttpServletRequest`: Includes the servlet request as part of the audit record
- `com.bea.contextelement.saml.SSLClientCertificateChain`: Includes an array of X.509 certificate objects that represent a certificate chain of a SAML client request

# Configuring the Default Auditing Provider



ORACLE®

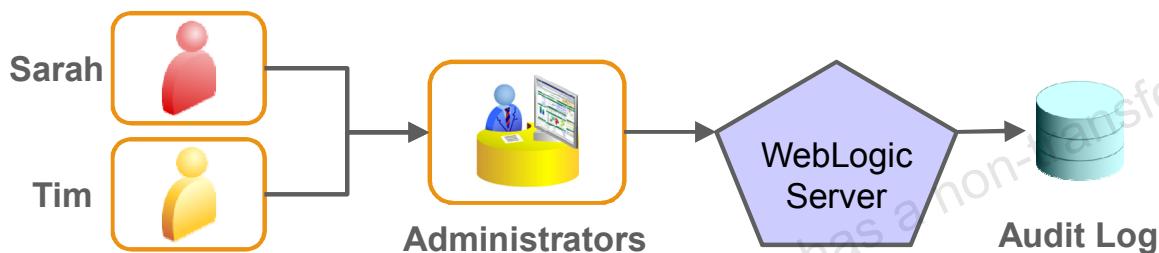
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

5. Optionally, select the severity of audit events to log including, INFORMATION, WARNING, ERROR, SUCCESS, FAILURE, and CUSTOM.
6. If the CUSTOM severity is selected, you can select a combination of multiple severities to include in audit records.
7. Save your changes.
8. You can optionally set some advanced settings to control the output of audit records.

# Configuration Auditing

Configure auditing for configuration changes to a domain:

- Add users to the WebLogic Administrators group to make them administrators.
- Audit records contain the administration user's information.

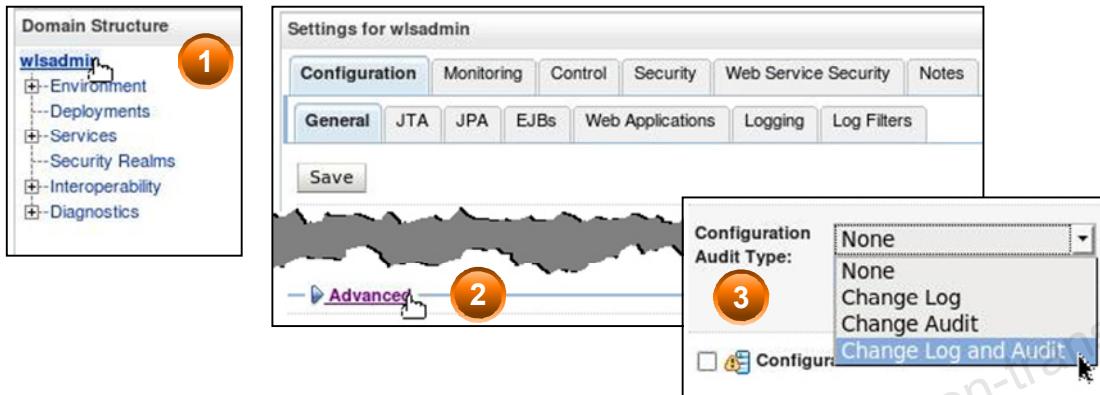


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic supports auditing changes made to a domain's configuration. Any user that is a member of the Administrators group is considered a WebLogic administrative account that can make changes to a domain. When individual users are also administrators, their identity is associated with the changes they make in the system. When configuration auditing and the WebLogic default auditor are configured, WebLogic either posts audit events, writes audit records to a log file, or both. When administrators make changes to the system, their identities are captured along with the change so there is a record of who made the change, what the change was, and the status of the change. You can set filters to set the severity of which configuration change audit records are generated including, SUCCESS, FAILURE, and ERROR.

# Configuration Auditing

Using the administration console:



Using the command line:

```
-Dweblogic.domain.ConfigurationAuditType="audit"
```

Can be audit,  
log, or  
logaudit

startWebLogic.sh

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You configure the type of configuration auditing either by using the administration console or by setting the `ConfigurationAuditType` property on the WebLogic Server command line.

Using the administration console, you:

1. Select the domain in the Domain Structure panel.
2. On the *Configuration > General tab*, click the Advanced link to display advanced settings.
3. Find the *Configuration Audit Type* field and set it to *None*, *Change Log*, *Change Audit*, or *Change Log and Audit*:
  - **None**: Disables configuration auditing
  - **Change Log**: Writes configuration audit records only to the server log file
  - **Change Audit**: Only emits configuration audit events for processing by the audit provider
  - **Change Log and Audit**: Emits configuration audit events and writes them to the server log file

## Quiz



Identify two activities for which the default auditor provides auditing.

- a. Security and configuration use
- b. Security and application use
- c. Security, configuration, and application use
- d. Security and third-party use

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer:** a

## Summary

In this lesson, you should have learned how to:

- Define users, groups, roles, and policies for the embedded LDAP server
- Configure auditing and role mapping



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 11-2 Overview: Configuring WebLogic Auditing

This practice covers the following topics:

- Configuring the default WebLogic audit provider
- Creating and viewing audit records

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.

# Disaster Recovery and Migration

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to:

- Describe disaster recovery scenarios for WebLogic Server
- Create a backup administration server
- Configure service-level transaction migration
- Describe whole server migration for WebLogic Server



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- Disaster Recovery (DR)
  - DR Concepts
  - Recommended DR Architecture
  - DR Best Practices
- Backup the Administration Server
- Service-Level Migration
- Whole Server Migration Overview



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Disaster Recovery Concepts

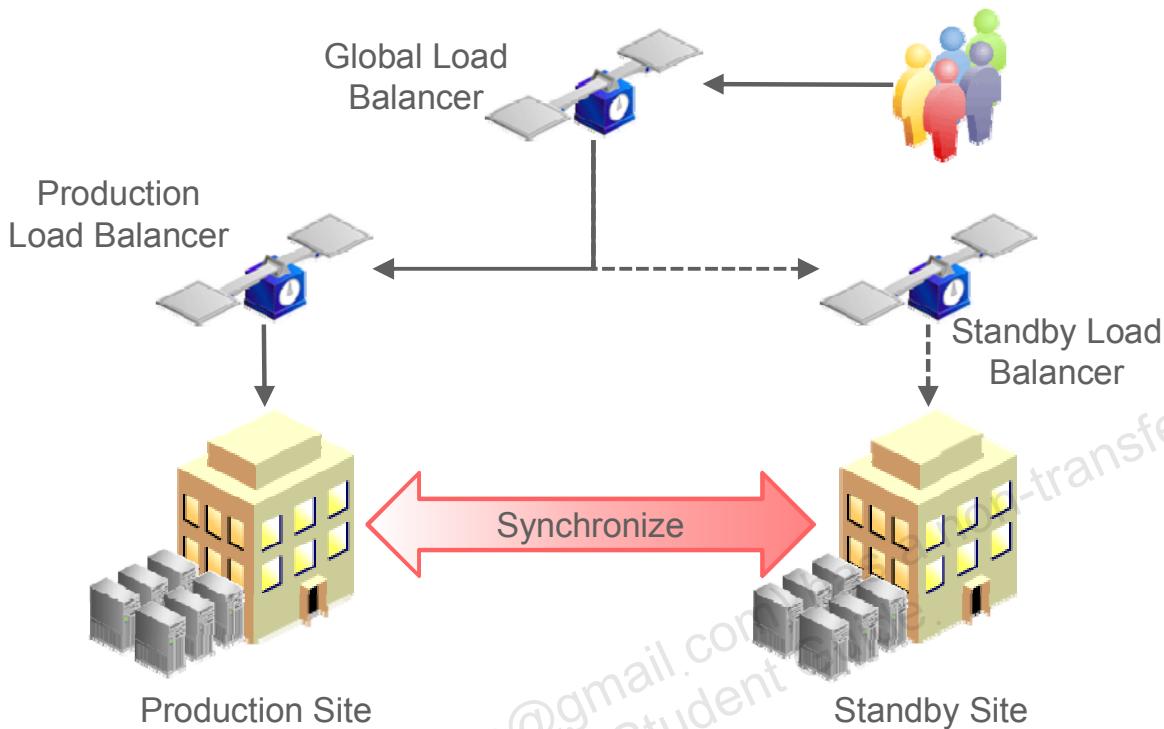
- Disaster recovery:
  - Ensures application availability after the loss of an entire data center
  - Addresses catastrophic failures, such as natural disasters
  - May or may not guarantee zero down time during the recovery process
- A common disaster recovery implementation involves deploying multiple data center sites in an active/passive fashion:
  - The *production (active)* site handles all application traffic.
  - The *standby (passive)* site is used only in the event of a disaster.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Enterprise deployments need protection from unforeseen disasters and natural calamities. One protection solution involves setting up a standby site at a geographically different location from the production site. The standby site may have equal or fewer services and resources compared to the production site. Application data, metadata, configuration data, and security data are replicated to the standby site on a periodic basis. The standby site is normally in passive mode; it is started when the production site is not available. This deployment model is sometimes referred to as an active/passive model. This model is normally adopted when the two sites are connected over a WAN and network latency does not allow clustering across the two sites.

# Disaster Recovery Concepts



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Clients access the production site during normal operation. During disaster recovery, clients access the standby site. Ideally, this change should be seamless from the client's perspective, although practically there may be a very brief interruption in service, depending on the implementation details.

For disaster recovery to be effective, the production and standby sites must be synchronized to some degree. At a very minimum, a version of the same application must be deployed to both sites.

# Site Symmetry

Oracle recommends symmetric configurations, because they:

- Require less setup and testing
- Ensure adequate performance

Standby Type	Description
Symmetric	Both the topology and hardware are identical to the production site.
Partially Symmetric	The topology is the same as the production site, but the hardware is different (configurations with a different number of machines, for example).
Asymmetric	The topology is different from the production site (number of tiers, servers, databases, and so on).



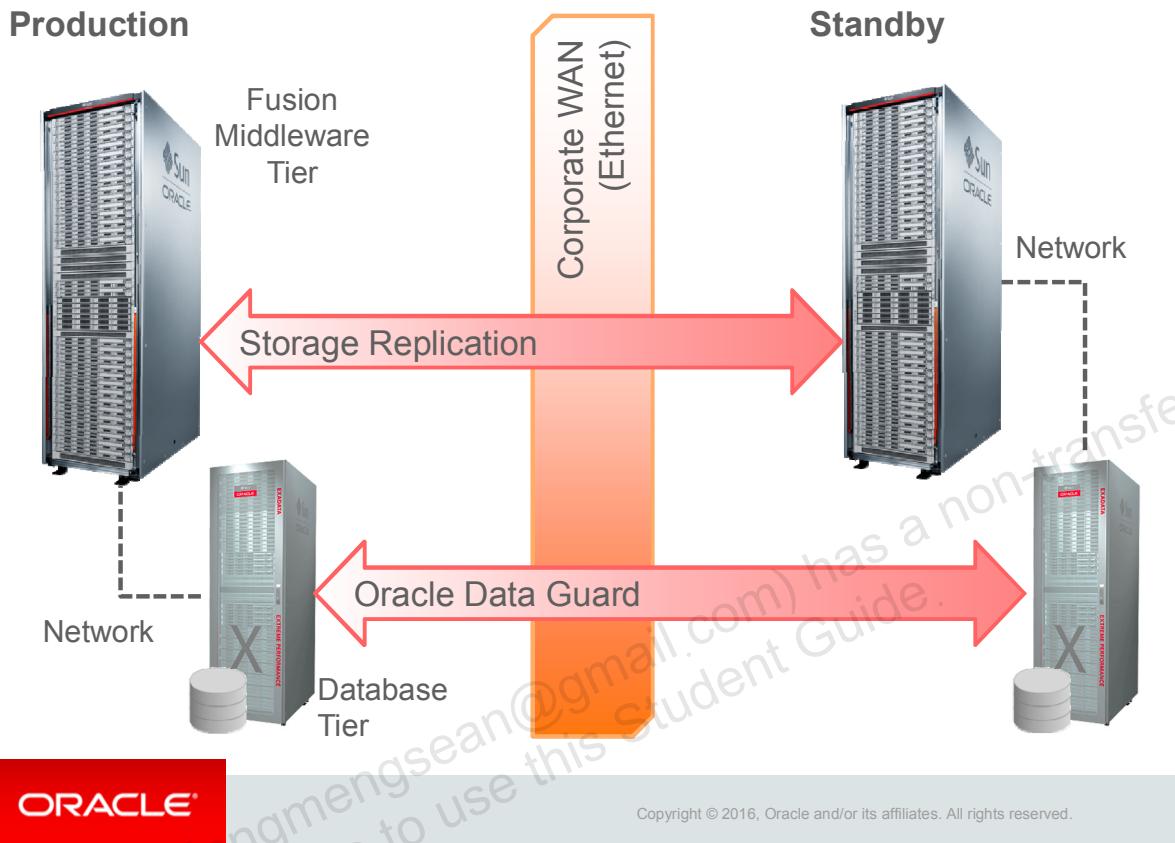
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A disaster recovery configuration that is completely identical across tiers on the production site and standby site is called a symmetric site. A site can be completely symmetric or partially symmetric. In a completely symmetric site, the production site and standby site are identical in all respects. That is, they have identical hardware, operating systems, load balancers, middleware instances, applications, and databases. The same port numbers are used for both sites as well.

In a partially symmetric site, the production site and standby site are identical in topology, but not in hardware. That is, they have the same number of middleware instances, applications, and databases on each site, but the hardware or operating system is not identical. For example, you can have ten machines on the production site and eight machines on the standby site. It is recommended but not required to have identical hardware and operating systems on the production and standby sites when planning a disaster recovery site.

In an asymmetric topology, the standby site has fewer resources than the production site. Typically, the standby site in an asymmetric topology has fewer hosts, load balancers, Fusion Middleware instances, and applications than the production site. It is important to ensure that an asymmetric standby site has sufficient resources to provide adequate performance when it assumes the production role. Otherwise, it may become oversaturated and unavailable.

## Recommended Architecture



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle's recommended disaster recovery strategy facilitates data protection in two main ways. You should replicate the storage of one system to the other to protect the middleware product binaries, configurations, metadata files, and application data that reside on the file system. Oracle Data Guard protects the Oracle database, which may or may not be running on Exadata machines. This database contains Oracle Fusion Middleware Repository data, as well as customer data.

Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable production Oracle databases to survive disasters and data corruptions. It maintains these standby databases as copies of the production database. Then, if the production database becomes unavailable because of a planned or an unplanned outage, Data Guard can switch any standby database to the production role, minimizing the down time associated with the outage. Data Guard can be used with traditional backup, restoration, and cluster techniques to provide a high level of data protection and data availability.

Oracle Active Data Guard, an option built on the infrastructure of Data Guard, allows a physical standby database to be open read-only while changes are applied to it from the primary database. Currently, Oracle Fusion Middleware does not support configuring Oracle Active Data Guard for its database repositories.

## General Best Practices

- Place all data on shared storage; only use compute node local storage for the OS.
- Bind to and use common host names instead of IP addresses.
- Use separate DNS servers and/or hosts files on each site.
- Replicate any external resources on which WebLogic processes depend (LDAP, database, and so on).



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In a disaster recovery topology, the host names used for wiring intra component and inter component communication need to be same. Typically, the site where the Oracle Fusion Middleware installation is done first dictates the host name used. The standby site instantiated subsequently should be configured to resolve these host names to the local standby site IP addresses. Therefore, it is important to plan the host names for the production site and standby site. It is also very important that the configuration at all levels use only host names. When configuring each component, use host name-based configuration instead of IP-based configuration, unless the component requires you to use IP-based configuration. For example, if you are configuring the listen address of an Oracle Fusion Middleware component to a specific IP address such as 192.168.10.33, use the host name `wlsvhn1.mycompany.com`, which resolves to 192.168.10.33.

## Hosts File: Example

### Production Site:

```
10.100.1.50 hradmin-prod.mycompany.com hradmin.mycompany.com  
192.168.1.51 hrwlsl1-prod.mycompany.com hrwlsl1.mycompany.com  
192.168.1.52 hrproxy1-prod.mycompany.com hrproxy1.mycompany.com
```

```
10.200.2.50 bakrepl.mycompany.com
```

Common host names across both sites

Site-specific addresses and host names

### Standby Site:

```
10.200.1.150 hradmin-bak.mycompany.com hradmin.mycompany.com  
192.168.1.151 hrwlsl1-bak.mycompany.com hrwlsl1.mycompany.com  
192.168.1.152 hrproxy1-bak.mycompany.com hrproxy1.mycompany.com
```

```
10.100.2.50 prodrepl.mycompany.com
```

The other site's storage appliance



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In a disaster recovery topology, the production site host names must be resolvable to the IP addresses of the corresponding peer systems at the standby site. This can be set up by creating a host name alias in the /etc/host file (the second entry shown in the slide). Create host name aliases for all the hosts on the production and standby sites. This example includes a WebLogic administration server, a WebLogic managed server, and a proxy server. Also, add entries for the replication channel on each storage appliance. The examples in this lesson assume that a symmetric disaster recovery site is being set up, where the production site and standby site have the same number of hosts and servers. Each host at the production site has a peer host at the standby site and the peer hosts are configured the same. For example, hosts at one site use the same port numbers as their counterparts at the other site.

## Quiz



A DR solution in which the production and standby sites have different topologies is \_\_\_\_\_.

- a. Symmetric
- b. Asymmetric
- c. Congruent
- d. Incongruent

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

# Agenda

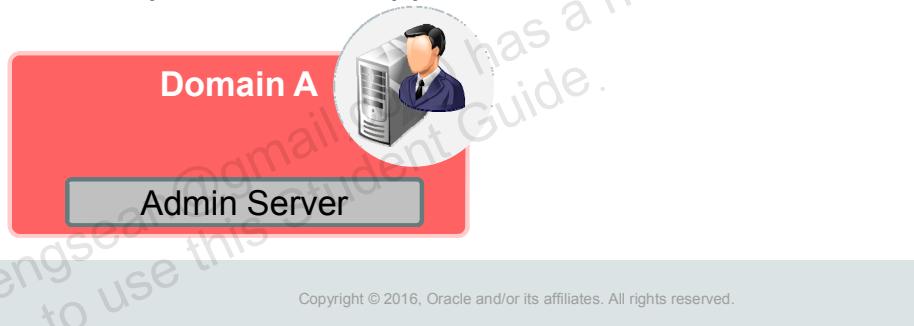
- Disaster Recovery (DR)
- Back Up the Administration Server
  - Administration Server Review
  - Impact of Administration Server Failure
  - Recovery of the Administration Server Configuration
  - Restarting an Administration Server on a New Computer
- Service-Level Migration
- Whole Server Migration Overview



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Administration Server Review

- A domain must have exactly one instance of WebLogic Server acting as the *administration server*. An administration server is part of exactly one domain.
- The administration server is:
  - The central point through which you configure and manage all domain resources
  - Solely in charge of the domain's configuration. It distributes configuration changes to other servers in the domain
  - An instance of WebLogic Server and, therefore, a fully functional Java Enterprise Edition application server



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

All domains contain a special server called the *administration server*. You use the administration server to configure and manage all the domain resources. Any other WebLogic Servers in the domain are called *managed servers*.

In most domains, the applications are deployed to the managed servers. The administration server is used only for domain configuration and management.

Because an administration server is an instance of WebLogic Server, it can perform any task of a Java Enterprise Edition application server. Applications can be deployed and run on the administration server. For simplicity, often a development-time domain will contain only the administration server. Developers deploy and test their applications on the administration server.

## Impact of Administration Server Failure

- Failure of the administration server:
  - Prevents configuration changes in the domain
  - Prevents application deployments
  - Prevents starting newly configured managed servers
  - Does not affect running managed servers
- Periodically, the managed servers attempt to synchronize configuration data with administration server.
- When the administration server becomes available, the managed servers get the latest configuration data from the administration server.
- Managed servers can be started in Managed Server Independence (MSI) mode.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When you first start a managed server, it must be able to connect to the administration server to retrieve a copy of the configuration. Subsequently, you can start a managed server even if the administration server is not running.

If a managed server cannot connect to the administration server during its start up, then it uses the locally cached configuration information. A managed server that starts without synchronizing its configuration with the administration server is running in Managed Server Independence (MSI) mode. By default, MSI mode is enabled. However a managed server cannot start in MSI mode for the first time because the local configuration is not available.

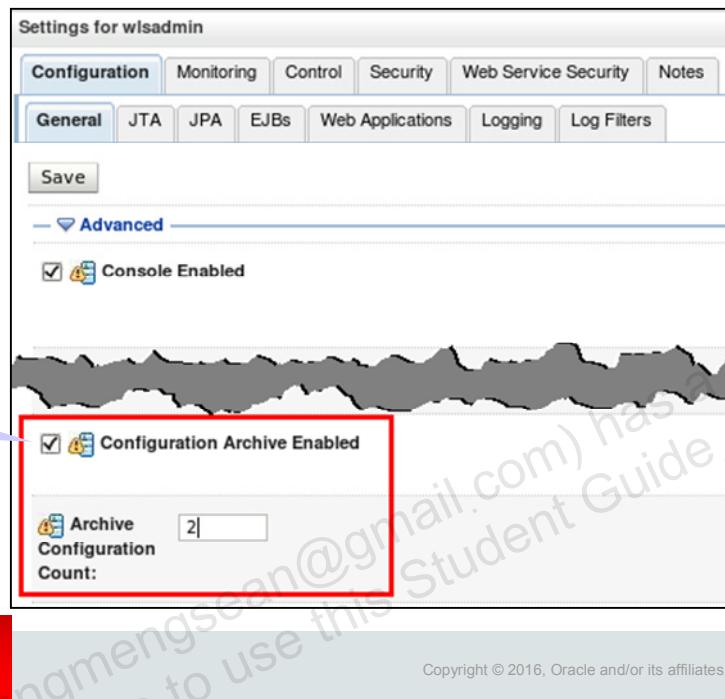
The failure of an administration server does not affect the operation of managed servers in the domain, but it does prevent you from changing the domain's configuration. If an administration server fails because of a hardware or software failure on its host computer, other server instances on the same computer may be similarly affected.

If an administration server becomes unavailable while the managed servers in the domain are running, then those managed servers continue to run. Periodically, the managed servers attempt to reconnect to the administration server. When the connection is successful, the configuration state is synchronized with that of the administration server.

For clustered managed server instances, the load balancing and failover capabilities supported by the domain configuration continue to remain available.

## Backing Up a Domain Configuration

- Enable autobackup of configuration.
- Check the new JAR files and directories.



Under Domain > Configuration > General > Advanced, you can enable the automatic backup of the configuration at the domain level. Each startup of the administration server creates two files in the domain directory: config-booted.jar and config-original.jar. In addition, each saved change of the configuration file makes a backup named configArchive/config-n.jar, where n is a sequential number. The Archive Configuration Count attribute limits the number of retained configuration JARs, so that in the example shown, there are never more than two kept: the most recent backup and the one immediately before that. Older backups are automatically deleted. If you made a series of mistakes, this provides a very easy way to return to a previous recent configuration. However, be aware that a typical configuration change requires clicking the Activate Changes button a few times, and each one then cycles the stored JARs.

You may want to set a higher number such as 10 or 20 for the Archive Configuration Count depending on:

- The available disk space
- The need for backup and restoration
- The time taken for backup and restore activity

**Note:** Although you use the configuration backup feature, it is always a good practice to manually make a backup of a known working configuration at an important milestone.

# Recovery of the Administration Server Configuration

Managed Server Independence (MSI) reduces the urgency to fix the outage.

The screenshot shows the 'Settings for server1' interface. The 'Tuning' tab is selected. A callout box with the text 'Enabled by default' points to the 'Managed Server Independence Enabled' checkbox, which is checked. Below it, there are two input fields: 'Period Length:' set to 60000 and 'Idle Periods Until Timeout:' set to 4.

ORACLE®

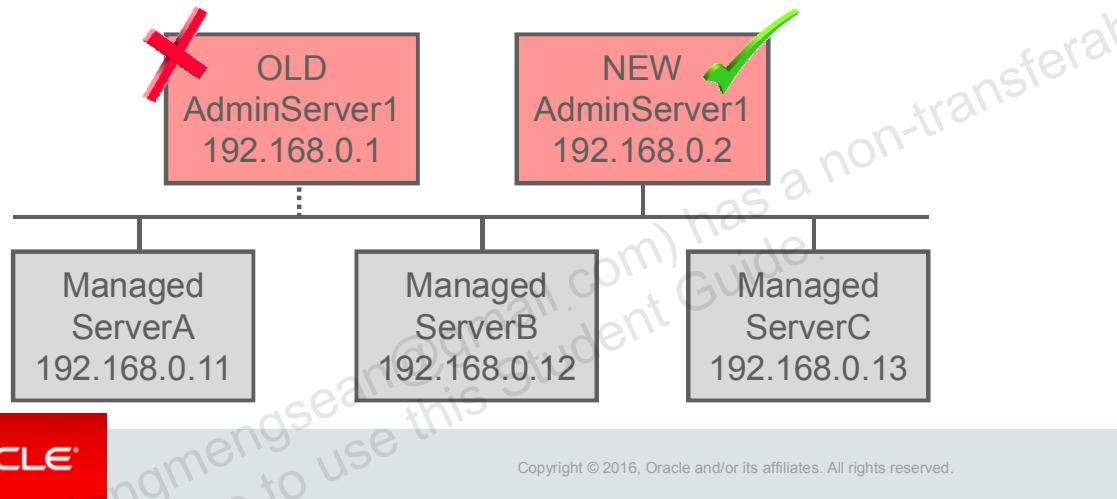
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The administration server is required only for making changes to the active configuration; it is not required for the normal operation of the managed servers as long as the managed servers are in Managed Server Independence Enabled mode, which is the default. This allows you time to recover the administration server without any service outages. As shown in the screenshot, the heartbeat detected between the administration server and the managed servers is, by default, a one-minute period. After four minutes of not hearing from the administration server, the managed servers become independent. After the administration server is fixed, the heartbeats start up again and the managed servers deactivate their independence, but MSI is still enabled for a future event. These times can all be changed to suit your particular environment.

## Restarting an Administration Server on a New Computer

WebLogic allows the creation of a backup of the administration server as follows:

1. Install Oracle WebLogic Server on a backup computer.
2. Copy the application files to a backup computer.
3. Copy the configuration files to a backup computer.
4. Restart the administration server on a new computer.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If a hardware crash prevents you from restarting the administration server on the same computer, you can recover the management of the running managed servers as follows:

1. Install the Oracle WebLogic Server software on the new computer designated as the replacement administration server.
2. Make your application files available to the new administration server by copying them from backups or by using a shared disk. Your files must be available in the same relative location on the new file system as on the file system of the original administration server.
3. Make your configuration and security files available to the new administration computer by copying them from backups or by using a shared disk. These files are located under the directory of the domain being managed by the administration server.
4. Restart the administration server on the new computer.

When the administration server starts, it communicates with the already-running managed servers via a Node Manager and informs the servers that the administration server is now running on a different IP address.

**Note:** You cannot have two administration servers at the same time, both claiming ownership of the same managed servers. This is not a warm standby; this must be a cold standby. The original administration server must be stopped or dead for the backup administration server to contact the managed servers.

## Quiz



What happens if you have a backup administration server?

- a. You are allowed to have only one administration server. If it fails, the managed servers run in MSI mode until your administration server comes back.
- b. It runs simultaneously with the primary administration server in a load-sharing mode.
- c. It can run in a warm standby keeping itself in sync with the main administration server.
- d. It must be in cold standby and you have to sync it with the main administration server manually.

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: d**

You can have only one administration server at a time; the backup administration server must be cold.

# Agenda

- Disaster Recovery (DR)
- Backup the Administration Server
- Service-Level Migration
  - JTA Review
  - Configure JTA Service-Level Migration
  - Set Up Automatic JTA Service-Level Migration
- Whole Server Migration Overview



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Java Transaction API (JTA) Review

- WebLogic Server uses JTA to implement and manage transactions.
- WebLogic Server's JTA implementation:
  - Creates a unique transaction identifier (XID)
  - Supports an optional transaction name
  - Tracks objects involved in transactions
  - Notifies databases of transactions
  - Orchestrates 2PC using XA
  - Executes rollbacks
  - Executes automatic recovery procedures in the event of failure
  - Manages timeouts

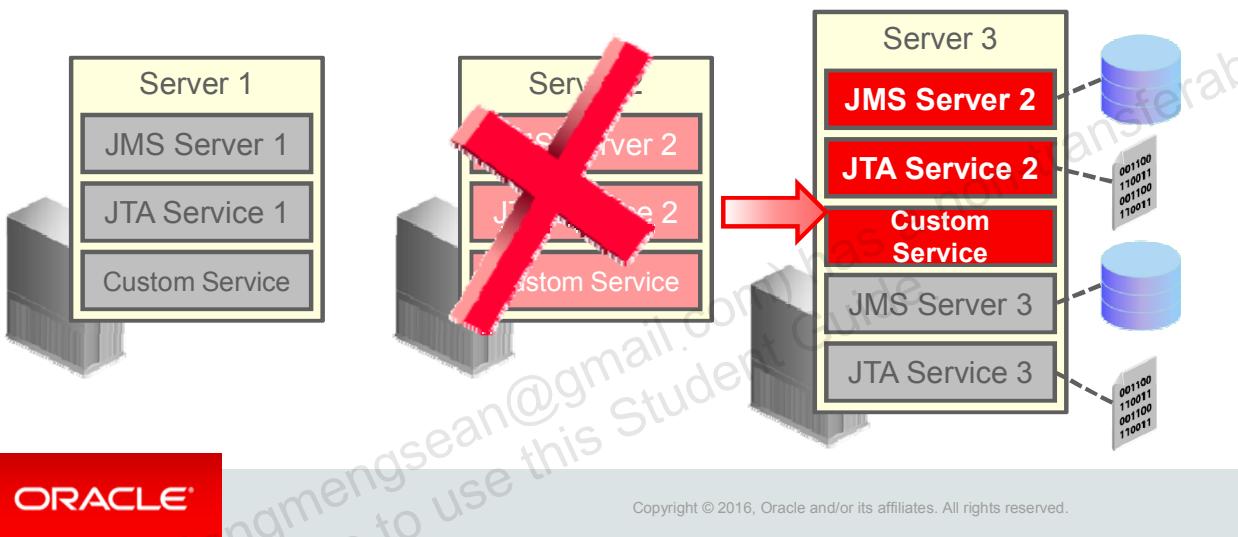


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# What Is Service Migration?

WebLogic's service migration infrastructure:

- Allows a pinned resource on a failed server to be restarted on another running server
- Supports both manual and automatic migration



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Service-level migration in WebLogic Server is the process of moving the pinned services from one server instance to a different server instance that is available within the cluster. The migration framework provides tools and infrastructure for configuring and migrating targets, and, in the case of automatic service migration, it leverages WebLogic Server's health monitoring subsystem to monitor the health of services hosted by a migratable target.

High availability is achieved by migrating a migratable target from one clustered server to another when a problem occurs on the original server. You can also manually migrate a migratable target for scheduled maintenance or you can configure the migratable target for automatic migration.

JTA services are singleton services, and, therefore, are not active on all server instances in a cluster. To ensure that singleton JTA services do not introduce a single point of failure for dependent applications in the cluster, WebLogic Server can be configured to automatically or manually migrate them to any server instance in the migratable target list.

The transaction service automatically attempts to recover transactions on system startup by parsing all transaction log records for incomplete transactions and completing them.

Within an application, you can also define a custom singleton service that can be used to perform tasks that you want to be executed on only one member of a cluster at any give time.

## Service Migration Prerequisites

- Servers must be members of a configured cluster.
- Automatic server migration must be disabled.
- A node manager must be running if you want to:
  - Use automatic service migration with consensus leasing
  - Run custom scripts before or after service migration
- Persistent stores and TLOGs must be backed by a highly available network file system or database:
  - TLOG records are stored in the default persistent store.



ORACLE®

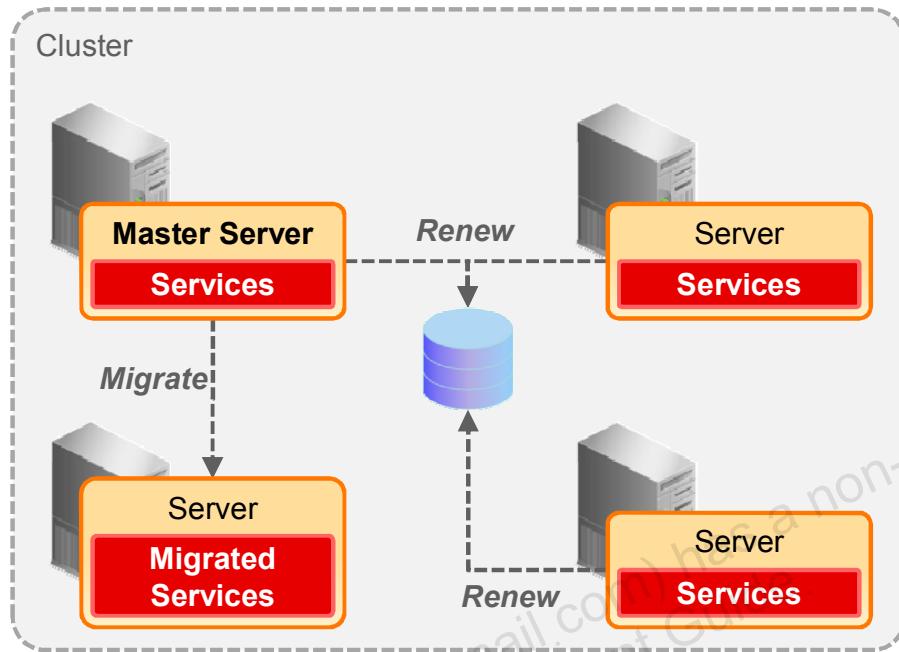
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When using consensus leasing, the member servers maintain leasing information in-memory, which removes the requirement of using a database. But this type of leasing also requires that you use Node Manager to control servers within the cluster. It also requires that all servers that are either migratable or that could host a migratable target must have a Node Manager associated with them. The Node Manager is required to get health monitoring information about the member servers involved.

Node Manager must be running on every machine hosting managed servers within the cluster only if pre/post-migration scripts are defined. If pre/post-migrations are not defined and the cluster uses the database leasing option, then Node Manager is not required.

To migrate the JTA Transaction Recovery Service from a failed server in a cluster to another server in the same cluster, the backup server must have access to the transaction log (TLOG) records from the failed server. Transaction log records are stored in the default persistent store for the server or in a shared database.

# Service Migration Architecture: Database Leasing



ORACLE®

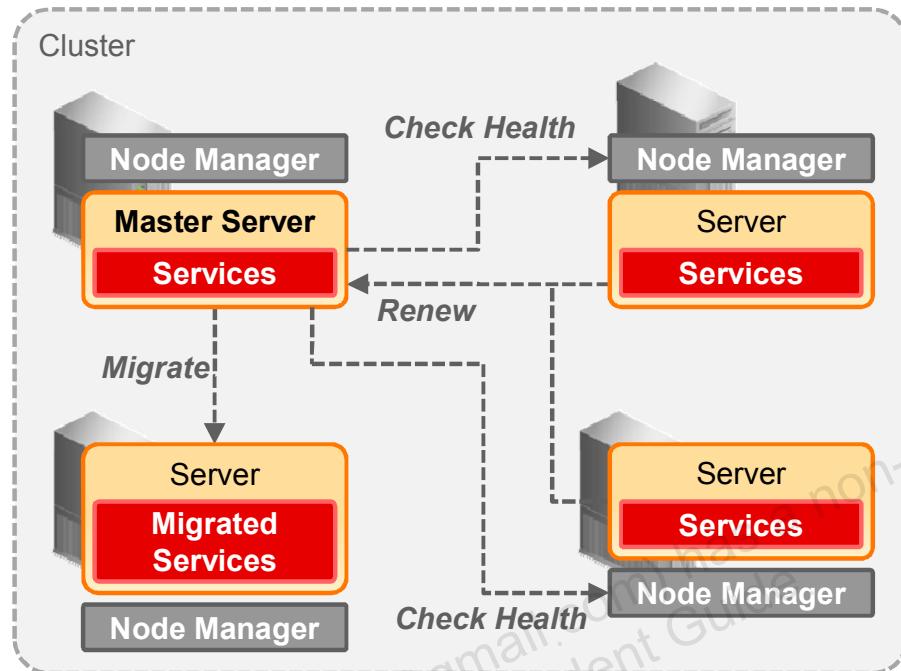
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Leasing is the process WebLogic Server uses to manage services that are required to run on only one member of a cluster at a time. Leasing ensures exclusive ownership of a cluster-wide entity. Within a cluster, there is a single owner of a lease. Additionally, leases can failover in case of server failure. This helps to avoid having a single point of failure.

Setting a cluster's Migration Basis to database leasing requires that the Data Source For Automatic Migration option be set with a valid JDBC data source. It also implies that there is a table created in the database that the managed servers will use for leasing.

To accommodate service migration requests, each server also performs basic health monitoring on migratable services that are deployed to it. A server also has a direct communication channel to the leasing system, and can request that the lease be released (thus triggering a migration) when bad health is detected.

# Service Migration Architecture: Consensus Leasing



ORACLE®

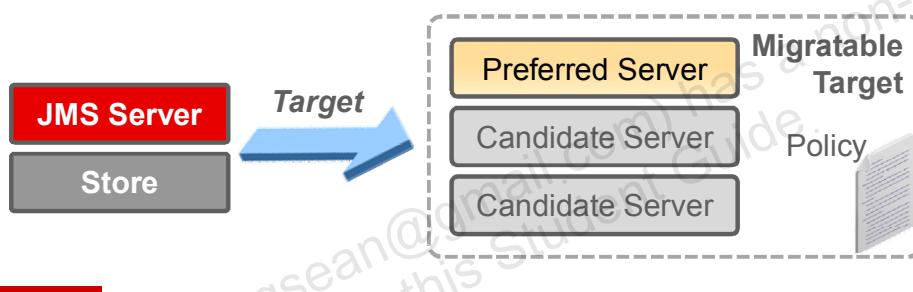
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Setting a cluster's Migration Basis to consensus leasing means that the member servers maintain leasing information in-memory, which removes the requirement of using a database. However, this version of leasing requires that you use Node Manager to control servers within the cluster. In other words, all servers that are migratable, or which could host a migratable target, must have a Node Manager associated with them. The Node Manager is required to get health monitoring information about the member servers involved.

# What Is a Migratable Target?

A migratable target:

- Defines a group of servers within a cluster, along with a migration policy
- Identifies a primary or “preferred” server
- Can identify a list of candidate servers to use for migration
- Allows you to group pinned resources that should be migrated together



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A migratable target is a special target that can migrate from one server in a cluster to another. As such, a migratable target provides a way to group migratable services that should move together. When the migratable target is migrated, all services hosted by that target are migrated.

To configure a JMS service for migration, it must be deployed to a migratable target. A migratable target specifies a set of servers that can host a target. Specifically it defines a preferred server for the services and an ordered list of candidate backup servers should the preferred server fail. Only one of these servers can host the migratable target at any one time.

After a service is configured to use a migratable target, the service is independent from the server member that is currently hosting it. For example, if a JMS server with a deployed JMS queue is configured to use a migratable target, the queue is decoupled from the server that hosts it. In other words, the queue is always available when the migratable target is hosted by any server in the cluster.

You must target your JMS service to the same migratable target that your custom persistent store is also targeted to. If no custom store is specified for a JMS service that uses a migratable target, a validation message will be generated by the Administration Server, followed by failed JMS server deployment. Moreover, a server configured in such a way will not boot.

# Service Migration Policy Options

Policy Type	Description
<b>Manual Only (default)</b>	Automatic service migration is disabled for this target.
<b>Failure Recovery</b>	Pinned services deployed to this target will: <ul style="list-style-type: none"><li>• <b>Initially start only on the preferred server</b></li><li>• Migrate only if the cluster master determines that the preferred server has failed</li></ul>
<b>Exactly-Once</b>	Pinned services deployed to this target will: <ul style="list-style-type: none"><li>• Initially start on a candidate server if the preferred one is unavailable</li><li>• Migrate if the host server fails <b>or is gracefully shut down</b></li></ul>



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a migratable target uses the manual policy (the system default), an administrator can manually migrate pinned migratable services from one server instance to another in the cluster, either in response to a server failure or as part of regularly scheduled maintenance.

The failure recovery policy indicates that the service will start only if its preferred server is started. If an administrator manually shuts down the preferred server, either gracefully or forcibly, then services will not migrate. However, if the preferred server fails due to an internal error, services will be automatically migrated to another candidate server. If such a candidate server is unavailable (due to a manual shutdown or an internal failure), the migration framework will first attempt to reactivate the service on its preferred server. If the preferred server is not available at that time, the service will be migrated to another candidate server.

The exactly-once policy indicates that if at least one server in the candidate list is running, then the service will be active somewhere in the cluster, even if servers are shut down (either gracefully or forcibly). It is important to note that this value can lead to target grouping. For example, if you have five exactly-once migratable targets and only boot one server in the cluster, all five targets will be activated on that server.

The failure recovery option is recommended for most types of clustered JMS applications.

## Configuration Roadmap

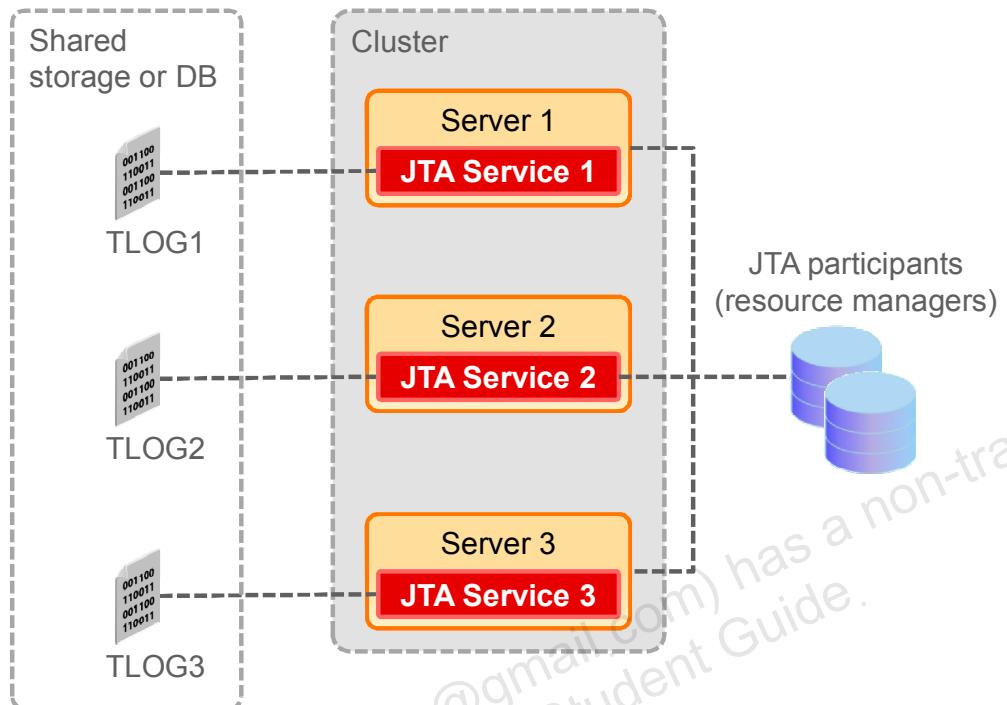
1. Configure the cluster leasing service.
2. Configure automatic JTA service migration.
3. Optionally, select candidate servers for migration.
4. Optionally, create custom pre/post migration scripts.
5. Configure the default store on all candidate servers to use a shared file system.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## JTA Service Migration: Before Failure



ORACLE®

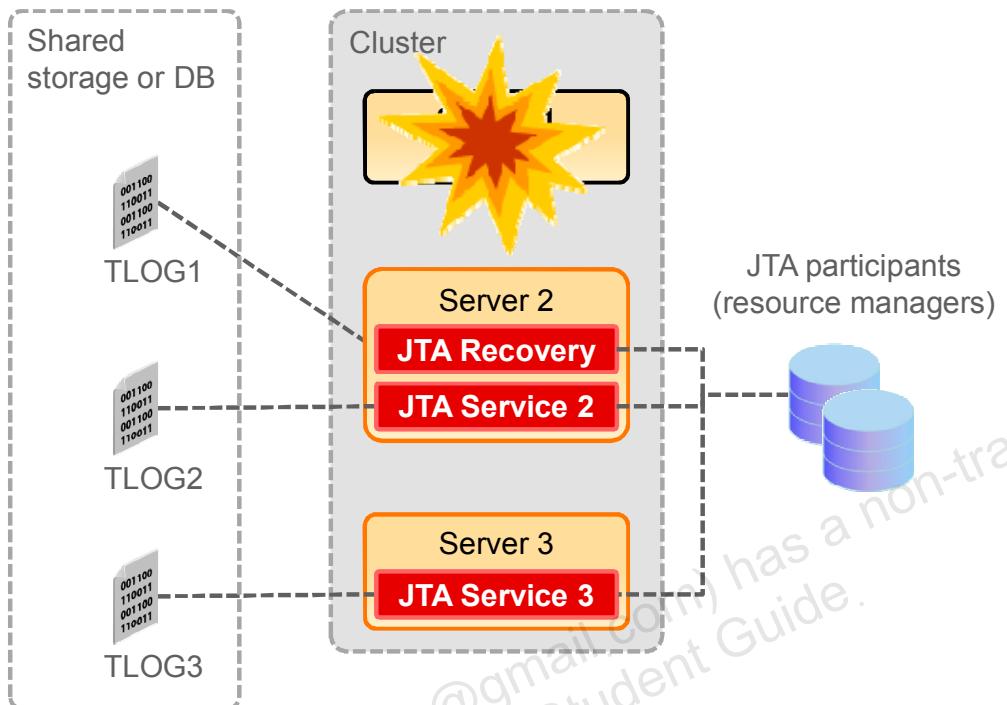
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The JTA Transaction Recovery Service is designed to gracefully handle transaction recovery after a crash. You can manually migrate the Transaction Recovery Service from an unhealthy server instance to a healthy server instance, with the help of the server health monitoring services. In this manner, the backup server can complete transaction work for the failed server.

By default, the transaction manager uses the default persistent store to store transaction log files. To enable migration of the Transaction Recovery Service, you must either configure JTA to use a highly available database, or configure the default persistent store so that it maintains its data files on a persistent storage solution that is available to other servers in the cluster.

JMS services can be migrated independently of the JTA Transaction Recovery Service. However, because the JTA Transaction Recovery Service provides the transaction control of the other subsystem services, it is usually migrated along with them. This ensures that the transaction integrity is maintained before and after the migration of the subsystem services.

## JTA Service Migration: After Failure



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

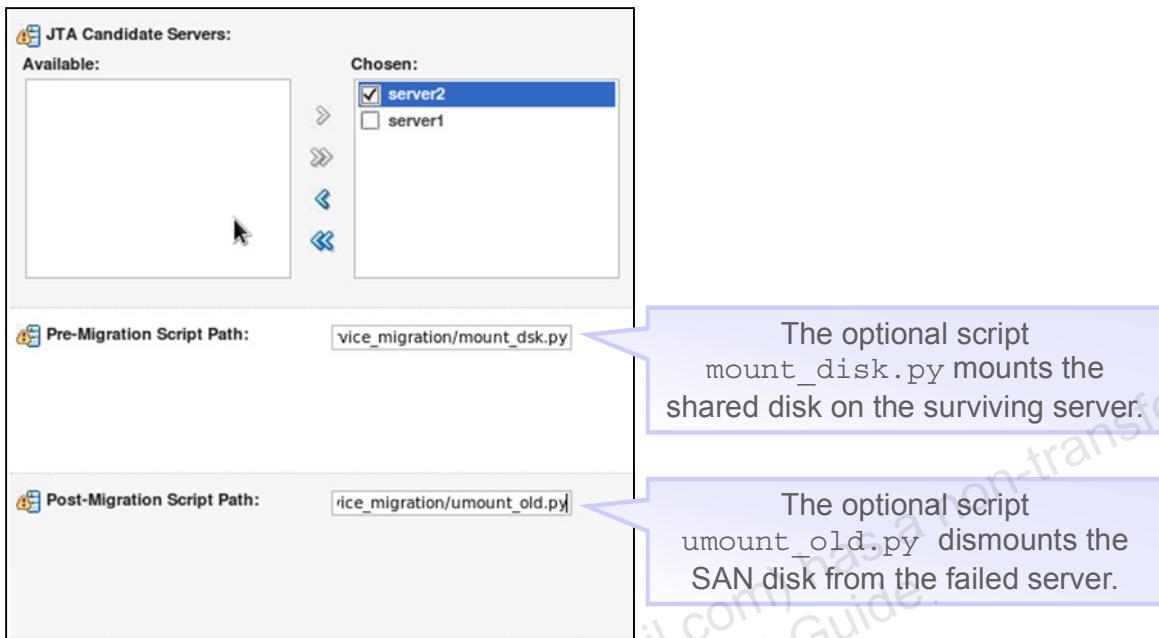
When JTA has automatic migration enabled, a server will shut itself down if the JTA subsystem reports itself as unhealthy (FAILED). For example, if any I/O error occurs when accessing the TLOG, then JTA health state will change to FAILED.

A server can perform transaction recovery for multiple failed servers. While recovering transactions for other servers, the backup server continues to process its own transactions.

For transactions for which a commit decision has been made, but the second phase of the two-phase commit process has not completed, the Transaction Recovery Service completes the commit process. For transactions that the transaction manager has prepared with a resource manager (transactions in phase one of the two-phase commit process), the Transaction Recovery Service must try to recover the transaction for each resource manager and eventually resolve (by either committing, rolling back, or forgetting) all transaction IDs identified during the recovery process.

If the backup server finishes recovering the TLOG transactions before the primary server is restarted, it will initiate an implicit migration of the Transaction Recovery Service back to the primary server. If the backup server is still recovering the TLOG transactions when the primary server is restarted, the backup server will initiate an implicit migration of the Transaction Recovery Service to the primary server.

# Configuring JTA Service Migration



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic Server provides the ability to migrate the Transaction Recovery Service at both the server level and the service level, either manually or automatically. Automatic service migration of the Transaction Recovery Service leverages the Health Monitoring subsystem to monitor the health of the service hosted by a migratable target. When the primary server fails, the migratable service framework automatically migrates the Transaction Recovery Service to a backup server. When using the Automatic Service Migration Feature, you must configure the "Migration Basis" for the cluster, enable automatic migration, and optionally specify whether you will be using any pre- or post-migration scripts. In the example, a pre-migration script, `mount_dsk.py`, ensures that a shared disk is mounted on the surviving server. Such script must be stored in the `DOMAIN_HOME/bin/service_migration` directory. Conversely, a post-migration script, stored in the same directory, called `umount_old.py`, ensures that a SAN disk is dismounted from the failed server after the migration has occurred.

## Set Up Automatic JTA Service-Level Migration

The screenshot shows the 'JTA Migration Configuration' section of the administration console. At the top, there is a checkbox labeled 'Automatic JTA Migration Enabled' which is checked. Below this, under 'JTA Candidate Servers:', there are two columns: 'Available' and 'Chosen'. The 'Available' column contains 'server2'. The 'Chosen' column contains 'server1' and 'server3'. A callout box with a blue border and arrow points to the 'Chosen' column, containing the text: 'Optionally, restrict the servers to which this service can migrate'.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A server's transaction manager is not assigned to a migratable target like other pinned services. Instead, manual and automatic JTA is simply configured for each individual server in the cluster. This is because the transaction manager has no direct dependencies on other pinned resources when contrasted with services such as JMS.

Select each server in the cluster. In the JTA Migration Configuration section of the server's Configuration > Migration tab, configure automatic migration of the JTA Transaction Recovery Service by selecting the Automatic JTA Migration Enabled check box.

You may also want to restrict the potential servers to which you can migrate the Transaction Recovery Service. For example, there may be cases when all servers do not have access to the current server's transaction log files. If no candidate servers are chosen, any server within the cluster can be chosen as a candidate server. From the Candidate Servers Available box, select the managed servers that can access the JTA log files. They become valid Candidate Servers when you move them into the Chosen box.

You must include the original server in the list of chosen servers so that you can manually migrate the Transaction Recovery Service back to the original server, if need be. The administration console enforces this rule.

## Quiz



What is the default policy for a migratable target?

- a. Manual
- b. Exactly-Once
- c. Preferred Server
- d. Failure Recovery

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer:** a

## Quiz



For which two service migration scenarios do you require Node Manager?

- a. Custom migration scripts
- b. Failure recovery policy
- c. Consensus leasing
- d. Database leasing

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, c

## Quiz



A Singleton Service can run at the same time on two managed servers:

- a. Only if it is configured to be manually migratable
- b. Only if the JTA is configured to use a highly available database (that is, RAC)
- c. Singleton Services can never run on more than one node at the same time
- d. Only if it is configured to be automatically migratable

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Practice 12-1 Overview: Configuring JTA Service-Level Migration

This practice covers the following topics:

- Configuring consensus leasing for a cluster
- Enabling automatic JTA migration for each node in the cluster
- Configuring candidate machines and servers for automatic migration
- Setting up a shared persistent store, accessible from all servers in the cluster
- Simulating a server crash and verifying the successful migration of the JTA service



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- Disaster Recovery (DR)
- Backup the Administration Server
- Service-Level Migration
- Whole Server Migration Overview



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Whole-Server Migration

WebLogic's whole-server migration infrastructure:

- Allows a server on a failed machine to be restarted on another machine
- Requires a running Node Manager on each machine participating in migration
- Requires servers to be members of a cluster
- Supports both manual and automatic failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

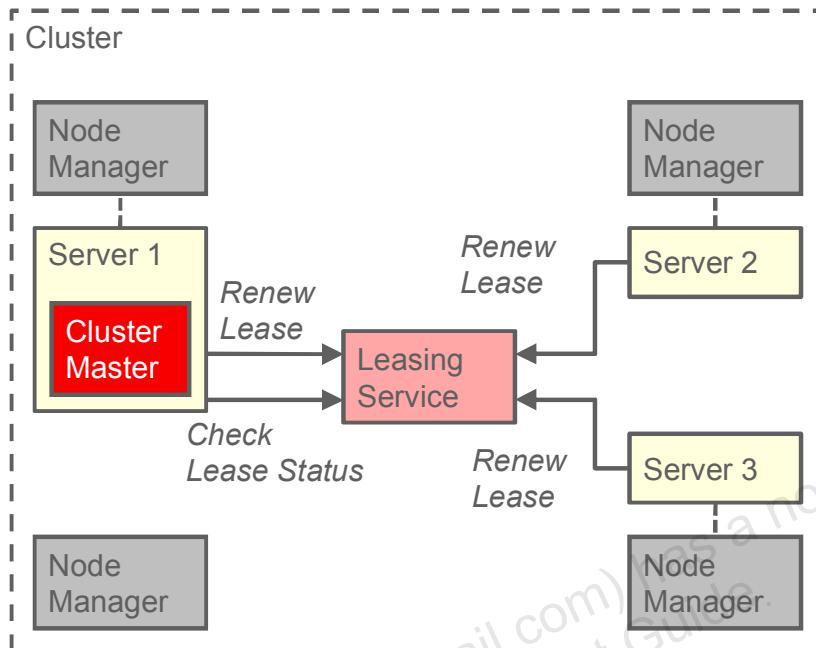
When a migratable server becomes unavailable for any reason (for example, if it hangs, loses network connectivity, or its host machine fails), migration is automatic. Upon failure, a migratable server is automatically restarted on the same machine if possible. If the migratable server cannot be restarted on the machine where it failed, it is migrated to another machine. In addition, an administrator can manually initiate migration of a server instance.

Node Manager is used by the Administration Server or a stand-alone Node Manager client to start and stop migratable servers and is invoked by the cluster master to shut down and restart migratable servers, as necessary.

Server migration has the following additional requirements:

- There is no built-in mechanism for transferring files that a server depends on between machines. Using a disk that is accessible from all machines is the preferred way to ensure file availability. If you cannot share disks between servers, you must ensure that the contents of `domain_dir/bin` are copied to each machine.
- You cannot create network channels that have different listen addresses on a migratable server.
- Although migration works when servers are not time-synchronized; time-synchronized servers are recommended in a clustered environment.

# Automatic Server Migration Architecture: No Failure

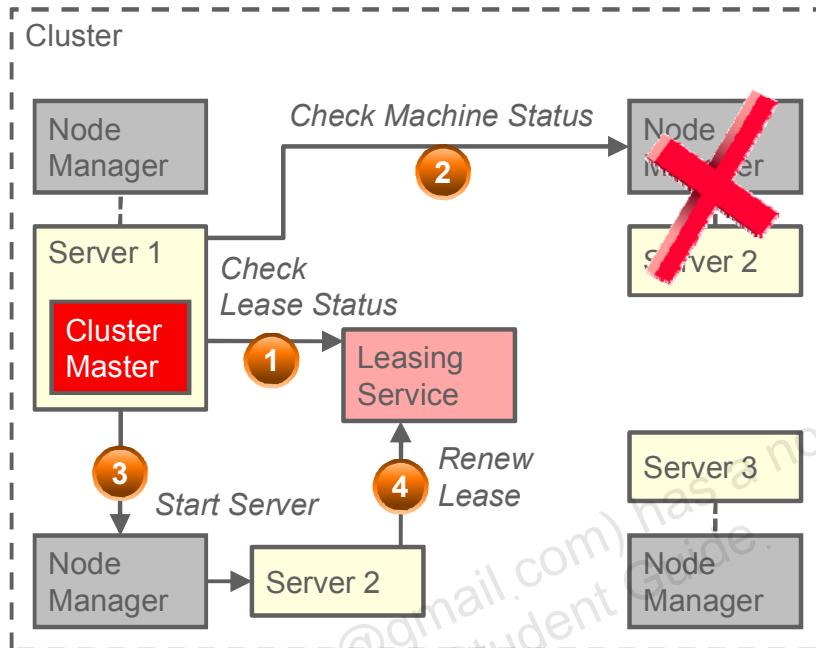


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example cluster contains three managed servers, all of which are migratable. Each managed server also runs on its own machine. A fourth machine is also available as a backup, if one of the migratable servers fails. Node Manager is running on the backup machine and on each machine with a running migratable server.

All managed servers in the cluster obtain a migratable server lease from the leasing service. They also periodically renew their leases in the lease table, proving their health and liveness. Because Server 1 starts up first, it also obtains a cluster master lease, whose responsibilities include monitoring the lease table.

## Automatic Server Migration Architecture: Machine Failure



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. The machine that hosts Server 2 fails. On its next periodic review of the lease table, the cluster master detects that Server 2's lease has expired.
2. The cluster master tries to contact the Node Manager on the backup machine to restart Server 2, but fails, because the entire machine is unreachable. Alternatively, if Server 2's lease had expired because it was hung, but its machine was still reachable, the cluster master would use the Node Manager to restart Server 2 on the same machine.
3. The cluster master contacts the Node Manager on the backup machine, which is configured as an available host for migratable servers in the cluster. The Node Manager then starts Server 2.
4. Server 2 starts up and contacts the Administration Server to obtain its configuration and finally obtains a migratable server lease.

## Configuration: Overview

1. Create machine definitions for your domain.
2. Configure Node Managers on your machines, including network interface settings.
3. Configure the cluster leasing service.
4. Enable automatic server migration on all or a subset of clustered servers.
5. Configure a list of candidate machines for server migration.



*Most migration  
settings require  
server restart.*

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Node Manager must be running and configured to allow server migration. The Java version of Node Manager can be used for server migration on Windows or UNIX. The SSH version of Node Manager can be used for server migration on UNIX only. Refer to the Node Manager Administrator's Guide in the WLS documentation for the available configuration and security options.

## Quiz



The two leasing mechanisms supported by WebLogic Server to manage services that are required to run on only one server in the cluster at a time are:

- a. Database leasing
- b. File-based leasing
- c. Voting-based leasing
- d. Two-phase leasing
- e. Concurrency-based leasing
- f. Consensus leasing

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: a, f**

## Summary

In this lesson, you should have learned how to:

- Describe disaster recovery scenarios for WebLogic Server
- Create a backup administration server
- Configure service-level transaction migration
- Describe whole server migration for WebLogic Server



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.

# Domain Partitions

ORACLE®

## Objectives

After completing this lesson, you should be able to:

- Describe Multitenancy concepts
- Enumerate domain partition concepts
- Create and manage single domain partitions
- Associate partitions with virtual targets
- Compare domain partition security realms options



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- Introduction to Multitenancy
- Multitenancy and WebLogic
- Working with Domain Partitions

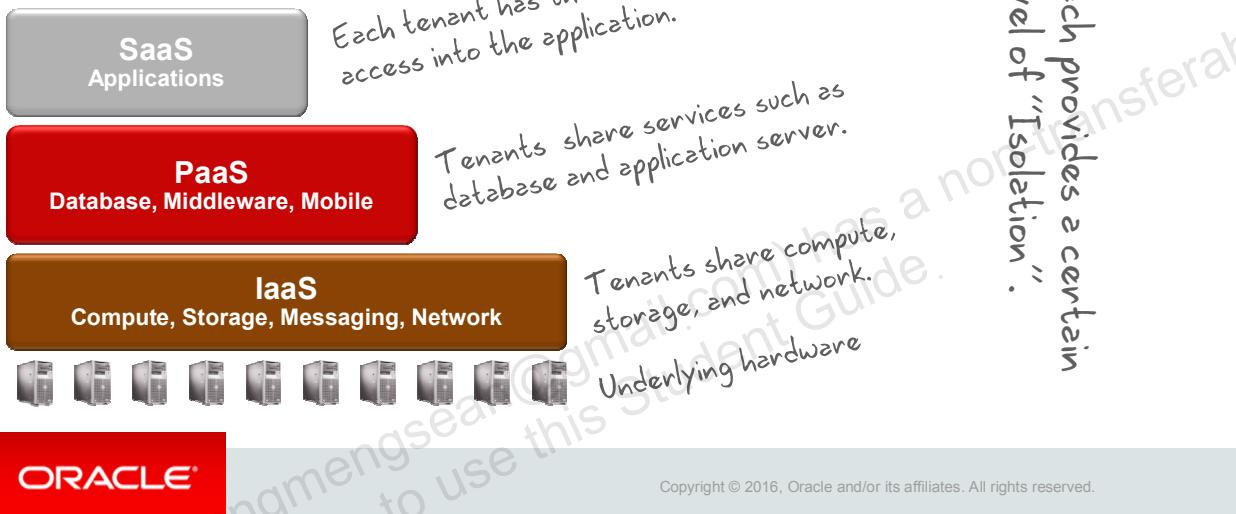


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Multitenancy

Multitenancy typically provides a set shared resources used by tenants, including:

- Software as a Service (SaaS): Tenants share applications
- Platform as a Service (PaaS): Tenants share platform
- Infrastructure as a Service (IaaS): Tenants share entire infrastructure



With Fusion Middleware 12c, multiple independent entities or organizations are permitted to operate and manage their own services and applications in a single WebLogic Server domain. The combination of all these ideas is termed Multitenancy. In general Multitenancy is a software mode of operation where multiple independent instances of one or more applications operate in a shared environment. The instances, or tenants, are logically isolated, but physically integrated. Multitenancy is based on three “as a service” concepts.

- Infrastructure as a Service or IaaS: Infrastructure as a Service is a form of cloud computing that provides virtualized computing resources over the Internet. Typically, IaaS provides access to virtualized compute, store, messaging, and network services.
- Platform as a Service or PaaS: Platform as a Service is a category of cloud computing services that provides platform-level services. Allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure. PaaS typically provides Database, Caching Level, Middleware, and Mobile services.
- Software as a Service or SaaS: Software as a Service is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet.

## Hardware Isolation

Each tenant's usage is isolated from other tenants.

- Physically separate hardware
- Hard partitioning
- Virtualization



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

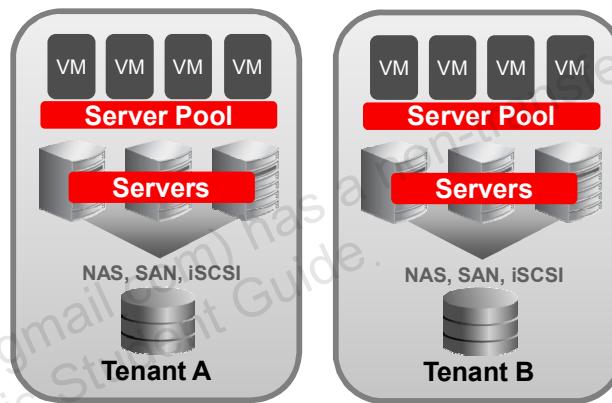
Cloud providers can isolate hardware for tenants with several approaches:

- Provision physically separate hardware, which essentially relocates the customer data center and may not provide consolidation.
- Hard partitioning would provide shared hardware among tenants, such as CPU and memory, but also provide a guaranteed allocation.
- Virtualization with hypervisors provides a strong consolidation model. However, packing too many resource-intensive tenants may lead to less-than-desirable performance characteristics.

# Network Isolation

Each tenant's network usage and traffic is separate from other tenants.

- Physically separated networks
- Virtual Private Networks
- VLANs
- Application Server channels



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Monitoring and controlling both physical and virtual network resources are key to understanding how bandwidth is being consumed.

Cloud providers must be able to:

- Identify bottlenecks to prevent network congestion and avoid down time
- Establish network service-level agreements (SLAs) and meet network quality of service (QoS) goals
- Understand network traffic trends and consumption in order to charge for network-related services
- Gather security information that could help prevent denial of service attacks (DoS)

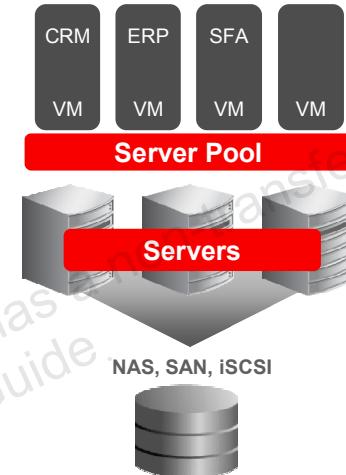
The first aspect of network traffic isolation is the creation of segmented networks. This can be done physically, or logically. In physical network isolation, network interface cards will be dedicated to a specific application or group of applications, and thus physical segmentation is provided between networks. Logical network isolation uses software such as VLANs, network interface virtualization (vNICs), or multiple logical listening endpoints to partition physical network resources. Traffic for multiple applications share the same physical interfaces, but each application sees only the network traffic and resources assigned to it, and cannot see traffic or resources assigned to other applications.

The second aspect of network traffic isolation is resource control or Quality of Service management (QoS). This needs to be in place to monitor and manage network traffic, and ensure that tenants consume only their fair-share of network bandwidth. Network resource management is generally applied at the vNIC level, that is, currently, there is no network governor that provides class of service (CoS) at the application or tenant level. Because vNICs are a tool for creating logical network isolation. Network resource QoS is most practical with logical network isolation environments.

## Application Isolation

Provides tenants with separation between actual code execution, deployment, versioning, and life cycle

- Separate servers running separate Application Servers
- Tenant-specific Application Servers on a shared IaaS



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

With multitenancy, unique tenant requirements and application components must be configurable to the needs of the tenant and the application.

Tenants do not necessarily trust one another, so applications require isolation between configuration, code execution, deployment, versioning, and life cycle.

Also, tenants expect the application to behave as if only a single tenant is the one accessing it.

# Security Isolation

Security Store for user and administrative access

- Authentication
- Authorization
- Auditing

Fusion Middleware 12c Multitenancy provides security isolation for tenants by using a separate security realm, identity domain, and partition-aware security services.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In any consolidation environment, a security implementation should use the “least privileges” approach to harden the environment.

**Security Realm:** A security realm comprises mechanisms for protecting WebLogic resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. A user must be defined in a security realm in order to access any WebLogic resources belonging to that realm. When a user attempts to access a particular WebLogic resource, WebLogic Server tries to authenticate and authorize the user by checking the security role assigned to the user in the relevant security realm and the security policy of the particular WebLogic resource.

An identity domain is a logical name space for users and groups, typically representing a discrete set of users and groups in a physical store. Identity domains are used to identify the users associated with particular partitions.

Partition-aware security services contain context about the partition in which they execute so that, for example, they can control access to a resource based on the partition to which the resource belongs. Partition-aware services are also identity domain aware.

To support partition-aware security services, WebLogic Server adds several new predicates that can be used in security policies to scope access at the partition level.

# Agenda

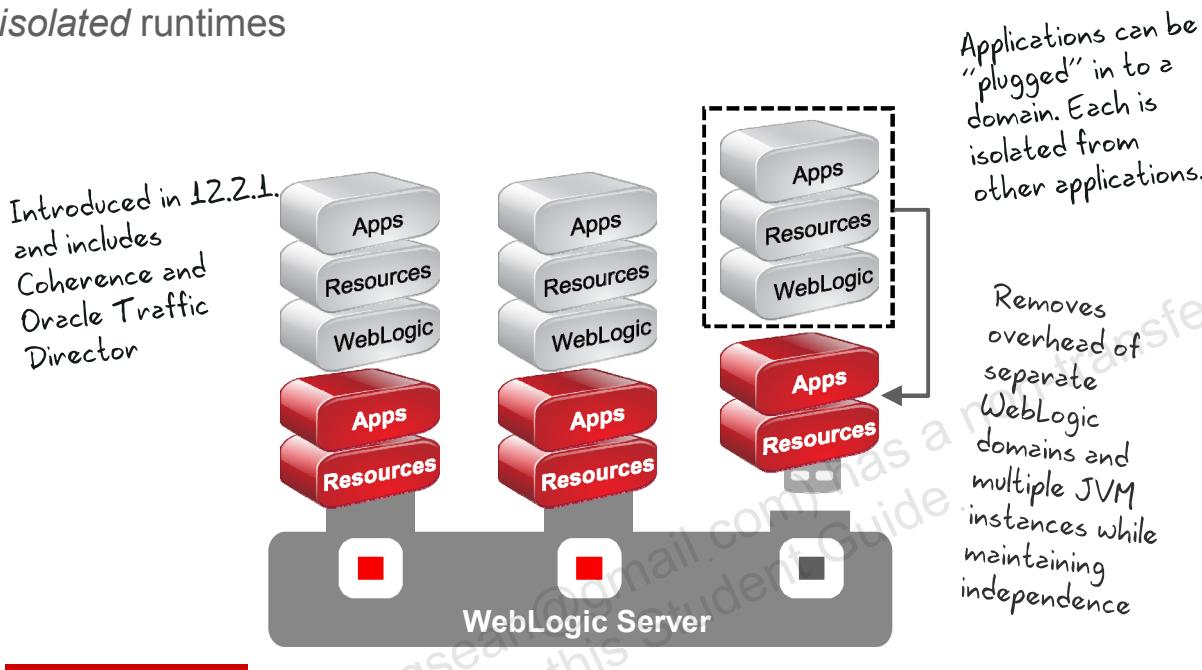
- Introduction to Multitenancy
- Multitenancy and WebLogic
- Working with Domain Partitions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# What Is WebLogic Server 12c Multitenancy?

A shareable *application infrastructure* whereby multiple tenants *share* a single WebLogic domain using *separate* resource configurations and *isolated* runtimes



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic Multitenant features enable you to encapsulate applications in micro containers for portability, and then run those micro containers on a shared platform.

## WebLogic Server 12c multitenancy:

- Provides a sharable infrastructure for use by multiple organizations
- Is a complete set of capabilities for shareable application infrastructure within a JVM/WebLogic Server instance
- Supports multiple tenants via a new capability—*domain partitions*

## Domain partitions:

- Are an administrative and runtime portion of a WebLogic Server domain
- Isolate resource configuration as well as the application runtime—this means that resource configuration managers can control resource consumption (including security realm, runtime traffic, data, heap, CPU) by running applications
- Enable applications and their resources to be independently “plugged” in to WebLogic Server—configured and managed independently from other applications sharing the same WebLogic Server infrastructure

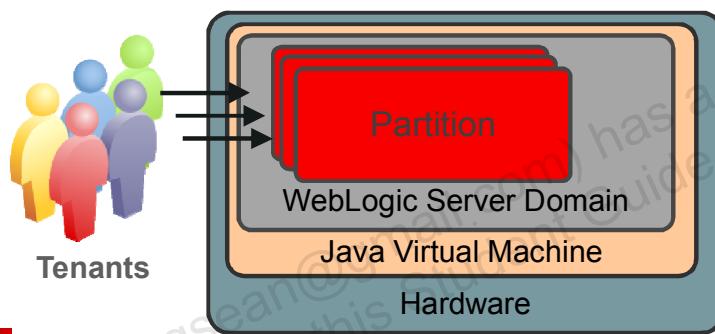
# Understanding Tenants and Partitions

Tenants are:

- User organizations that use applications and resources
- Typically associated with a user identity store

Domain Partitions:

- Are an administrative and runtime slice of a WebLogic Domain
- Isolate application data, configuration and runtime traffic
- Have independent life cycle



Tenants represent distinct user organizations, such as different external companies (for example, CompanyA and CompanyB), or different departments within a single company (for example, HR and Finance), that use applications and resources within a WebLogic domain. One of the key aspects of tenants is identity, that is this group has its own set of users. Identities may be stored in a particular LDAP or other store so that access/protection requires a specific security configuration.

A tenant is a logical grouping of your own choosing; it is not a configurable object in Fusion Middleware Control or WLST. A tenant's identity is the association of a given user with a given tenant. For example, you might choose to associate the tenant Finance with a specific domain partition called Finance-partition. The system derives which tenant a given user belongs to from the user's identity as prescribed by the user identity store. Further, the user's identity helps the system enforce what that user will be authorized to do as they move throughout the system, including but not limited to which tenant the user belongs.

## What Is a Domain Partition?

Domain partition:

- Is a portion of a WebLogic Server domain
- Enables sharing of domain resources
- Isolates tenant-specific content
- Separates both administrative and runtime

Such as JVM, Application instances, related resources and WLS itself

WebLogic Server domains can contain:

- Zero or more domain partitions
- Zero or more virtual targets

Including application data and configuration, runtime traffic, and resource management



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic Server multitenancy provides resource isolation within a domain partition. A domain partition is both an administrative and a runtime portion of a WebLogic domain that is dedicated to running application instances and providing resources for a tenant. Domain partitions achieve greater density by allowing multiple application instances, and their related resources, to share a domain, WebLogic Server, Java virtual machine, and the operating system, while isolating tenant-specific application data, configuration, and runtime traffic. Each domain partition has its own runtime copy of applications, resources, and state. Changes in how WebLogic Server handles class loading provide both application isolation and efficiency.

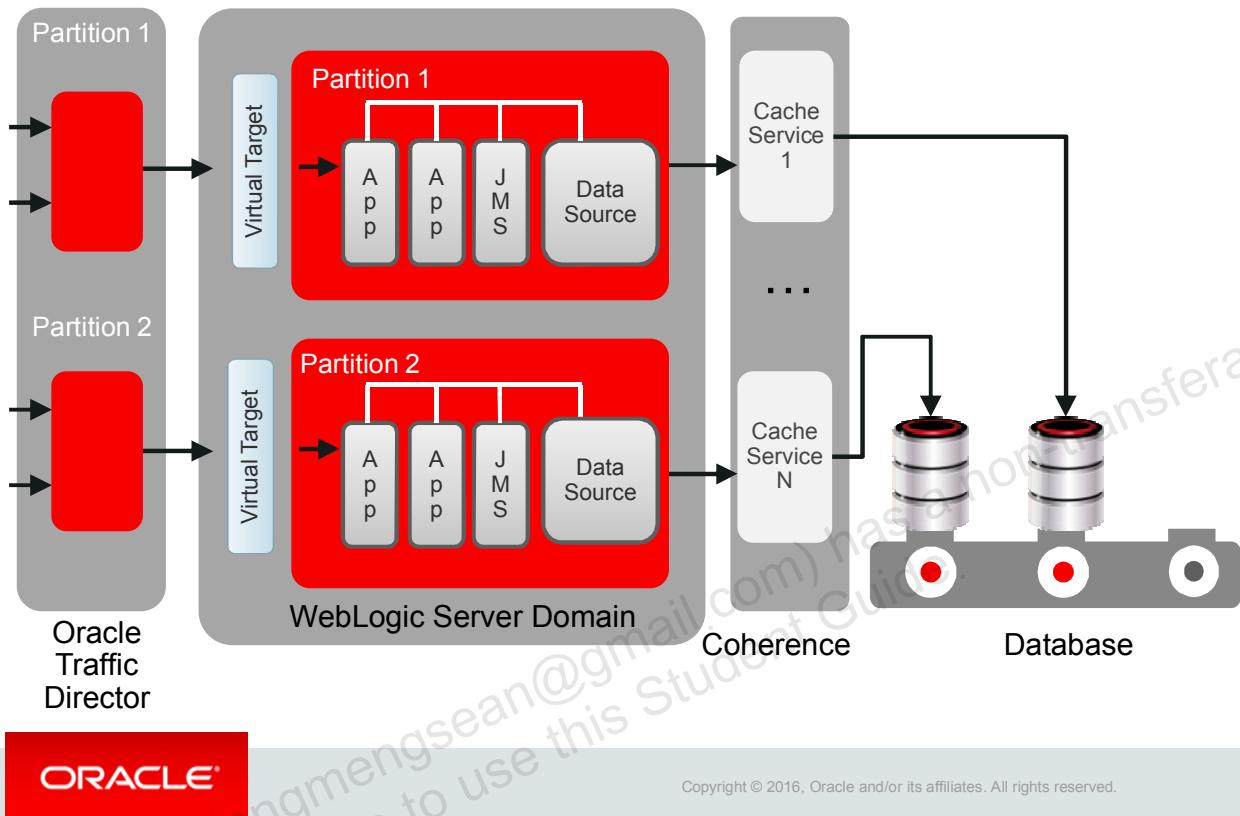
Deploying to multitenant environments requires no changes to your applications. For example, you could run multiple instances of a payroll application in different domain partitions without modifying the application.

WebLogic Server multitenancy extends the WebLogic Server Enterprise Edition and WebLogic Suite products, and includes the following components:

- WebLogic Server
- Oracle Traffic Director
- Coherence
- Fusion Middleware Control, used as the primary graphical user interface

# WebLogic Server Multitenancy

## *End-to-End Integration*



A domain partition is an administrative portion of a domain that can be managed independently and can share the runtime capacity in a domain—the Managed Servers and clusters. Each domain partition has a virtual target, applications, and required resources.

From the Web tier, using Oracle Traffic Director or plug-ins, traffic is routed to the address defined by the virtual target. After a request enters into a partition, it stays in that partition. Each partition has its own JNDI tree, so any use of resources is within the partition. That also means data segregation. For JMS, each JMS server has its own set of queues and its own persistence.

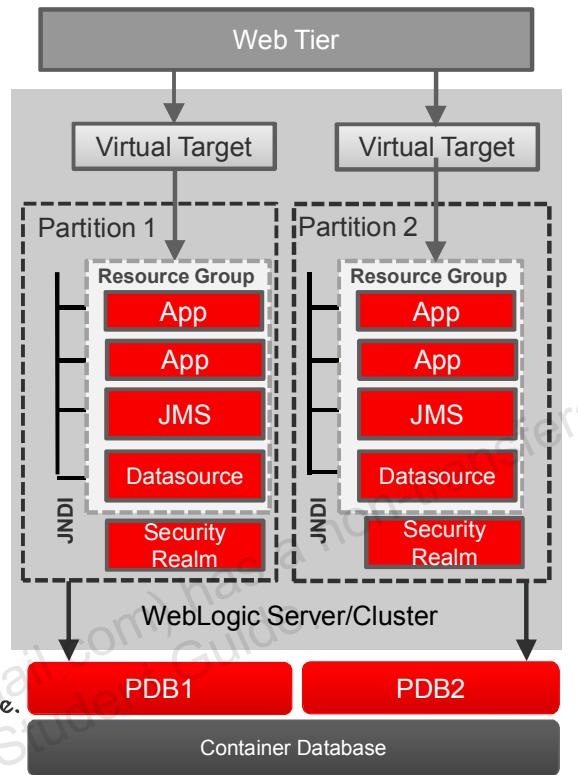
And each partition can have its own datasources that connect to dedicated PDBs. Note that PDBs are not required, but they are a natural fit. And, of course, you have Coherence integration. Applications can use a dedicated service in Coherence, or they can use a shared service, depending on application needs.

# WebLogic Server Multitenancy

## *Partition Isolation*

- Per-partition management
- Resource Consumption Management
- Dedicated JNDI tree for resource isolation
- Virtual target per partition for application access
- Work Manager per partition for thread management
- Security realm per partition
- Memory and CPU isolation at the JVM level

Thread dumps per partition are not available.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Partition isolation is one of the main focuses of multitenancy for WebLogic Server. Partition isolation affects all areas of WebLogic Server including:

- Management: Partitions can be started, stopped, and quieted individually.
- Resource Consumption Management: Applications consume resources. Resource Consumption Management, or RCM, allows for policies and mechanisms for defining resource consumption controls and limits.
- JNDI isolation: Each application running in a partition has an isolated JNDI tree. Applications in the same partition see the same JNDI tree but nothing from other partitions.
- Work Managers: Partition Work Managers set thread usage policy among partitions. You can configure them to limit the number of Work Manager threads in each partition, as well as to manage thread usage allocation based on thread usage time for each partition.
- Memory and CPU isolation: Both memory access and CPU access are managed at the JVM level, completely isolating applications from one another.

# WebLogic Server Multitenancy Components

**Domain partitions:** Are managed independently; for example, start and stop

- Contain applications and **resource groups**
- Can span clusters!
- Can be imported and exported
- Use **virtual targets**
- Use **security realms**

**Resource groups:**

- Contain web applications and related resources

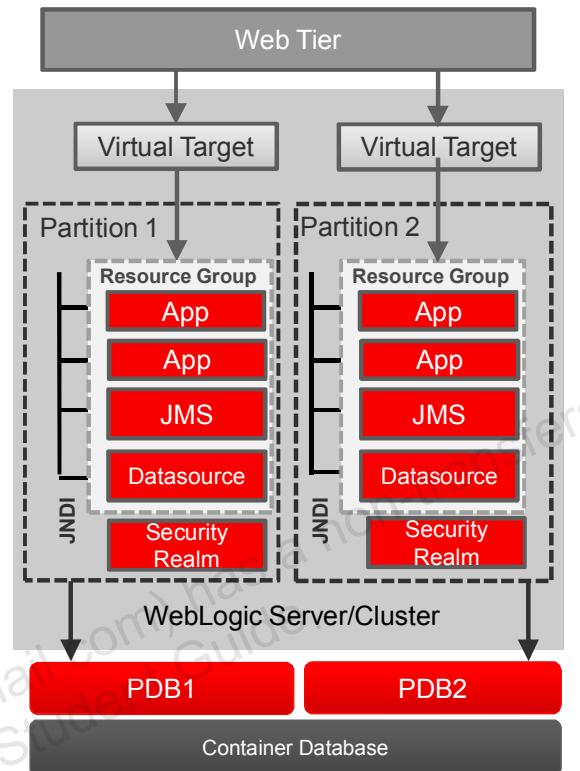
**Virtual targets define:**

- Application endpoints

**Security realms:**

- Can be shared or independent

Multitenancy requires separate license. Suite and Enterprise Edition permit 1 partition.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic Server multitenancy components include the following:

- Domain partitions** provide an administrative and runtime slice of a WebLogic domain. Domain partition can be started and stopped independently of other partitions and uses virtual target(s) for resource group application endpoints. Each domain partition may use its own security realm to separate principals from other partitions or share security realms between partitions.
- Resource groups** contain the related resources running in a partition, including web applications, JMS configurations, JDBC data sources, as well as other application resources. Resource group templates are a mechanism for creating patterns for resource groups. Administrators can use resource group templates when creating partitions.
- Virtual targets** provide the application endpoint targets used by a resource group in a partition.
- Partition security** enables security isolation by associating a security realm to a partition.
- Partition Work Managers** provide a mechanism to ensure fairness for thread usage and prioritization of work requests among partitions.

Later slides will examine these concepts in more detail.

# What Are Resource Groups?

## Resource groups:

- Named collections of deployable artifacts
- Administrative units within a WebLogic Server domain
- Defined at Domain or Domain Partition level
- May be started or stopped independently
- Targeted as a single unit

## Resource groups include:

- JEE application deployments
- JDBC data sources and mail sessions
- JMS messaging configuration
- Targets



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Resource groups are a named collection of, typically, related deployable resources. Resource groups often include Java EE applications, data sources, JMS artifacts, and other resources used by the application. Resource groups are a convenient way to group together Java EE applications and the resources they use into a distinct administrative unit within the domain.

WebLogic Server multitenancy introduces resource groups, as a convenient way to group together Java EE applications and the resources they use into a distinct administrative unit within the domain. The resources and applications are “fully qualified” in that the administrator provides all the information needed to start or connect to those resources, including credentials for connecting to a data source and targeting information for Java EE applications. A resource group will either contain these deployable resources directly or refer to a resource group template that defines the resources. Resource groups can be defined at the domain level, or be specific to a domain partition.

All the resources in or referenced by a resource group are targeted together (to the same target). A resource group can be started and stopped independently of other resource groups. Each resource group that refers to a given template will have its own runtime copies of the resources defined in the template. A resource group template is a convenient way to define and replicate resources for multiple tenants. Resource group templates make it very easy to deploy the same collection of applications and resources to multiple domain partitions.

# What Are Resource Group Templates?

Resource group templates: *Used as patterns*

- Are named reusable sets of deployable resources
- Can be the basis for Resource Groups
- Deploy identical sets of applications and resources to domain partitions
- Can reference other resource groups or templates
- May be reused by many domain partitions

*Resource groups based on resource group templates result in runtime copies of resources.*



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Resource group templates are a named, domain-level collection of deployable resources intended to be used as a pattern by (usually) multiple resource groups. Each resource group that refers to a given template will have its own runtime copies of the resources defined in the template. A resource group template is a convenient way to define and replicate resources for multiple tenants. Resource group templates make it very easy to deploy the same collection of applications and resources to multiple domain partitions.

Resource group templates can define:

- Application deployments
- Library deployments
- JDBC system resources
- JMS system resources
- Coherence system resources
- File stores
- JMS servers
- Messaging bridges
- Path services

Resource group templates can define:

- Path services
- Persistent stores
- SAF agents
- Foreign JNDI providers
- Mail sessions
- WLDF modules

Resource group templates are defined at the domain level, and then referenced by one or more resource groups.

Resource group templates are particularly useful in a SaaS environment where WebLogic Server MT activates the same applications and resources multiple times, once per domain partition. Some of the information about such resources is the same across all domain partitions, whereas some of it, such as the attributes of a JMS queue or of a DB connection, varies from one partition to the next. For example, the URL, username, and password used to connect to a data source would be different among different partitions. WLS MT will activate the application suite represented by the template differently for each partition.

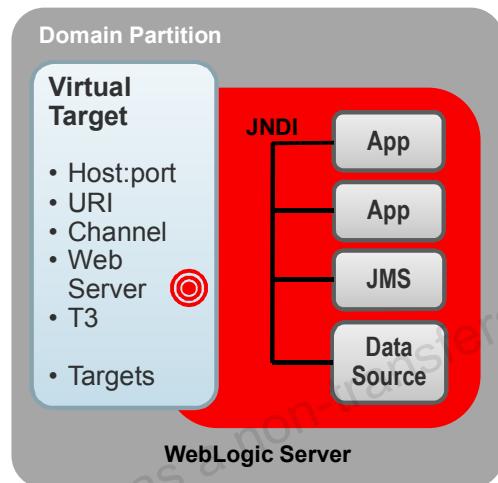
You can specify override configuration MBeans and resource deployment plans to override some attribute values in a partition.

# Virtual Targets

- Represent endpoints for domain partitions and resource groups
- Encapsulate traffic routing
- Include host/address, channel, protocol, URI, and targets

Virtual target guidelines:

- One per domain partition or domain
- Resource groups within a domain partition must have a virtual target
- May use a cluster or Managed Server
- May include a URI prefix
- Cannot be deleted while in use



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Virtual targets encapsulate where a partition or resource group runs and how to route traffic to them, including addresses, protocol settings, and targeting. Request routing is determined by the host name and optional URI.

May include:

- Host name and port
- Optional URI
- Network access point/channel
- Protocol-specific configuration
- Target clusters and Managed Servers

Virtual targets isolate a domain partition and the global (domain-level) runtime from the physical attributes of the target WebLogic Server instances so that your application does not have to know these details.

A virtual target represents a target for a resource group, both at the domain level and in a domain partition. It defines the access points to resources, such as one or more host names, a URI prefix, and the Managed Servers or cluster to which the virtual target is itself targeted.

Virtual targets are similar to the virtual hosts previously available on WebLogic Server. Like virtual hosts, virtual targets provide a separate HTTP server (web container) on each target. A virtual target isolates a resource group from the physical attributes of where the container is running. Each resource group requires one or more virtual targets. The virtual targets you can select depends on whether the resource group is at the domain level, or associated with a domain partition:

- For domain-level resource groups, you can select any existing virtual target that is not already assigned to a partition as an available target.
- For resource groups in a domain partition, you must select only from the existing virtual targets available to the partition.

Consider the following guidelines for using virtual targets:

- A virtual target can be used by only one partition at a time, or at the domain level. A virtual target is never shared by more than one partition, or by a partition and the domain level.
- A virtual target can be used by many resource groups within a partition, or by many resource groups at the domain level.
- A virtual target can be targeted at one cluster or one Managed Server.
- Many virtual targets can be targeted at the same cluster or Managed Server.
- If two virtual targets have the same host names, URI prefix, and port number, they must be targeted to different clusters or Managed Servers.

A virtual target contains the following:

- **Host names.**
  - The list of host names for which this virtual target will serve requests. The host names you specify must have DNS entries that resolve to the correct servers, or to a load balancer:
  - If you are not using Oracle Traffic Director (OTD) to load balance connections for a partition, specify the actual host name of the WebLogic Server cluster or Managed Server.
  - If you are using OTD to load balance connections for a partition, you access applications through OTD. The host names you specify for a virtual target must resolve to the location of the OTD Administration Server. If you do not specify a host name, it is the equivalent of wild carding the host name to match all incoming request.
  - The host name used by a client to access the resource group must exactly match one of the host names specified in the virtual target.

- The host name used by a client to access the resource group must exactly match one of the host names specified in the virtual target.
- You can specify multiple host names for the virtual target. You might find it convenient to specify both the simple and the fully qualified host name to ensure a match.
- **Optional URI prefix** for which this virtual target will serve requests.
  - For example, if you enter www.example.com as the host name and MyApp as the URI prefix, this virtual target serves requests to www.example.com/MyApp, but not to www.example.com or [www.example.com/foo](http://www.example.com/foo).
  - To extend this example, assume that your application root is /app. The resulting URL for the application would be [www.example.com/MyApp/app](http://www.example.com/MyApp/app).
- **Target cluster and Managed Server** from the list of the targets in the current domain. Targets must be either a single server or a single cluster.

There are two general approaches when using virtual targets: route by host name (<http://partition1.com>, <http://partition2.com>), or by URI prefix (<http://server.com/partition1>, <http://server.com/partition2>). You might find it easier to manage your environment if you consistently use one approach, but there is no requirement that you do so.

You must specify at least one of host name, URI prefix, or port number.

## Virtual Target Versus Virtual Host

### Virtual Host:

- Allows Web application access to be restricted to specific host names
- Relies upon browser sending specific `Host` HTTP Header content matching Virtual Host's DNS name
  - Does not offer isolation of non-Web application access

### Virtual Target:

- Allows arbitrary resource access restrictions
- Is not limited to Web applications
- Is required for Resource Group targeting
- May be used at domain level or within domain partition

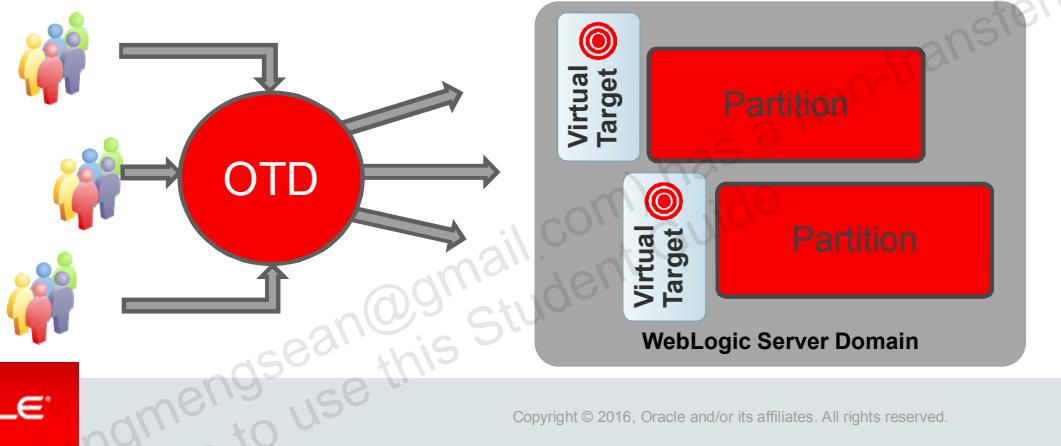
Virtual Target and Virtual Host both may use dedicated Network channels (dedicated listeners).



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Oracle Traffic Director 12.2.1

- Oracle Traffic Director is a high-performance, fast, reliable and scalable software load balancer.
- Oracle Traffic Director features include:
  - Ability to forward HTTP, HTTPS, and TCP traffic
  - Optimizations for Oracle Exalogic Elastic Cloud
  - The WebLogic Management Framework
  - Manageable using Fusion Middleware Control



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Traffic Director (OTD) is an on-board, highly available load balancer used to optimize application-to-application communication within Oracle's engineered systems such as WebLogic and others.

OTD offers the following key benefits:

- High throughput, low latency HTTP(s), WebSocket, and TCP software load balancer.
- Rule-based request routing and reverse proxy server. Request rate limiting, throttling and QoS tuning.
- Flexible and fine-grained monitoring and high availability through standard-based VRRP protocol.
- High performance SSL/TLS transactions including offloading SSL, caching HTTP responses from origin server(s) and compression.
- Built-in WebLogic optimizations such as cluster discovery, and connection persistence synchronization.
- Mod Security based Web Application Firewall to protect back-end applications from malicious attacks such as CSRF, SQL Injection and so on.

# Partition Security

## Multiple Active Security Realms

- Within a domain, multiple realms can be active for partitions.
- Each partition may execute against a different security realm.
- Partitions may also share a security realm.

## Identity Domain

- It is a logical name space for users and groups in a security realm.
- A domain partition can be associated with an identity domain.

## Partition-Aware Security Services

- New predicates in security policies to scope partition access



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To support multitenancy and partitions, the WebLogic Security Service includes the following new features in release 12.2.1:

- **Multiple active realms:** To provide isolation of the security configuration and metadata, WebLogic Server now supports the ability to have multiple active security realms in a domain. Support for multiple active realms enables per-partition configuration for realm-based services by allowing each partition to execute against a different realm. Partitions may also share a security realm, which may be appropriate when the security configuration between two partitions does not require isolation.
- **Identity domains:** An identity domain is a logical name space for users and groups, typically representing a discrete set of users and groups in a physical store. Identity domains are used to identify the users associated with particular partitions.
- **Partition-aware security services:** Partition-aware security services contain context about the partition in which they execute so that, for example, they can control access to a resource based on the partition to which the resource belongs. Partition-aware services are also identity domain aware.

To support partition-aware security services, WebLogic Server adds several new predicates that can be used in security policies to scope access at the partition level.

## Partition Work Managers

- Provide thread resource management for domain partitions
- Are defined at the domain level
- Are assigned at domain partition creation
- Are 1:1 with domain partitions
- Can be customized for each domain partition

*Default values are used when no work manager is defined.*

### Configuration Includes:

- Maximum threads constraint
- Minimum threads constraint
- Fair share
- Shared capacity percent

*Percentage of global resource utilization*



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Partition-specific Work Managers provide fairness of thread usage and prioritization of work requests among partitions that share the same WLS instance.

### Partition Work Managers:

- Allow system administrators to set thread usage policies among partitions and configure customized QoS levels
- Allow partition administrators to configure Work Manager policies for their partitions
- Support the prioritization of thread usage by work requests from various partitions
- Ensure the fairness in thread resource usage among partitions

Partition Work Managers set thread usage policy among partitions. You can configure them to limit the number of Work Manager threads in each partition, as well as to manage thread usage allocation based on thread usage time for each partition. The importance of regulating the relative thread usage is to provide proper quality of service (QoS) and fairness among various partitions that share the same WLS instance. Without it, a rogue application from one partition could starve thread resources from other partitions preventing them from functioning properly.

Partition Work Managers provide resource management within partitions. Administrators know about the runtime environment and how it will be shared. They configure Partition Work Managers at the domain level and assign them to partitions as they are created. These predefined Partition Work Managers let administrators standardize Work Manager configuration; for example, all partitions with business-critical applications can reference the business-critical Partition Work Manager.

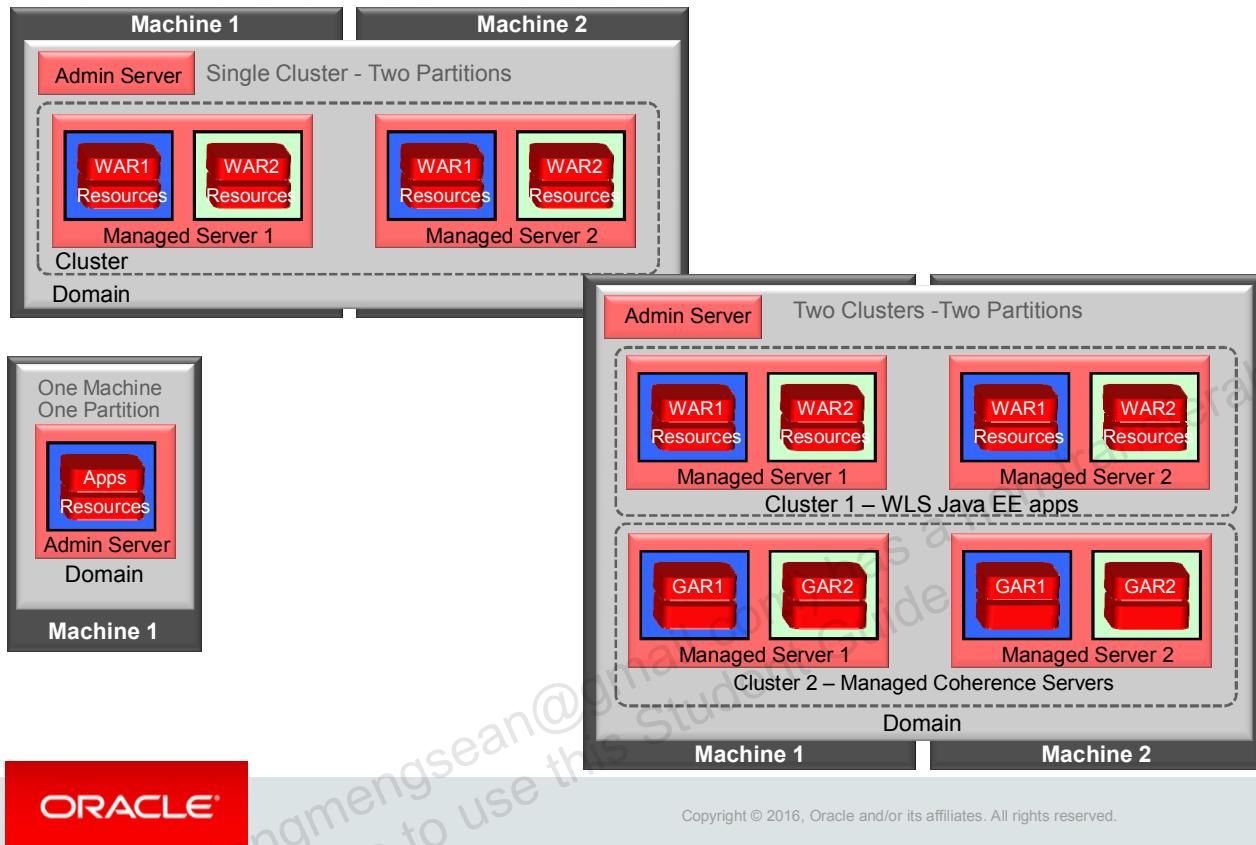
Administrators might also want to customize the Partition Work Manager for a specific partition, or maybe for every partition. In this scenario, they configure the Partition Work Manager within (embedded in) the partition configuration. There is no need to predefine Partition Work Manager configurations.

You can define a Partition Work Manager in the domain to use with multiple domain partitions, or you can define Partition Work Manager attributes in the domain partition itself for use in that partition only. If no Partition Work Managers are defined, default values for Partition Work Manager settings are applied.

Partition Work Managers can be used in more than one domain partition. However, a domain partition can be associated with only one Partition Work Manager.

- **max threads constraint:** Limits on the number of threads that will be allocated to a partition concurrently
- **min threads constraint cap:** Sets a cap on the number of threads to satisfy minimum thread constraints within the partition
- **Fair Share:** Ratio of request processing
- **Shared Capacity percent:** Defines a percentage of the global request limit (the default global request limit is 65536)

# Multitenancy Examples



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

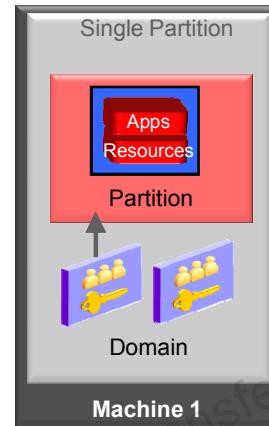
Partitions can be created in any number of ways. This slide provides some simple examples of how partitions might be structured:

- **One machine/one partition:** This example represents the traditional model. One or more applications deployed to the same partition. All services are shared, and all applications can access and see all other application.
- **Single cluster/two partitions:** This example shows a single cluster with two Managed Server instances, over a single domain and cluster. All WAR1, shown on both servers, is isolated from WAR2.
- **Two clusters/two partitions:** This example shows a single domain with two clusters. Each application, in this case WAR and GARs, is isolated from each other.

## Single Partition Use Case

Single partition supported without additional licensing

- Supported by all versions of WebLogic
- Uses a separate security realm
  - Set of users administer domain
  - Set of users user/administer partition
- Domain administrators manage and monitor WebLogic Domain.
- Partition administrators manage and monitor the partition.
- Supports the common requirement isolating administrators from users



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The single partition use case is a common one with WebLogic Server and suites a common need, separation of authority,

With a single partition:

- Two security realms: One for WebLogic Server itself and one for the partition.
- Separation of duties: A set of staff administrate the WLS domain and a second set of staff administrate and use the partition.
- Users/credentials are different for each set with the security realms containing different users/groups.
- Management/monitoring: Domain administrators manage and monitor the entire domain. Partition administrators can only manage, monitor and work within the partition.
- Supported by all editions of WebLogic Server: Single partition is included in WebLogic Enterprise Edition. Second and subsequent partitions require additional licensing.
- Customer request: Single partition fulfills requests by customers requesting different security realm isolation or an application, administration, and users.

# Agenda

- Introduction to Multitenancy
- Multitenancy and WebLogic
- Working with Domain Partitions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Creating Virtual Targets

1

In the administration console, select the **Domain Structure** pane.

2

Under the domain, select **Environment > Virtual Targets**.



3

Click **New**.

A virtual target represents a target for a resource group in a partition URI path.  
This page summarizes the virtual target instances that have been created.

Customize this table

Virtual Targets (Filtered - More Columns Exist)

New    Clone    Delete

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

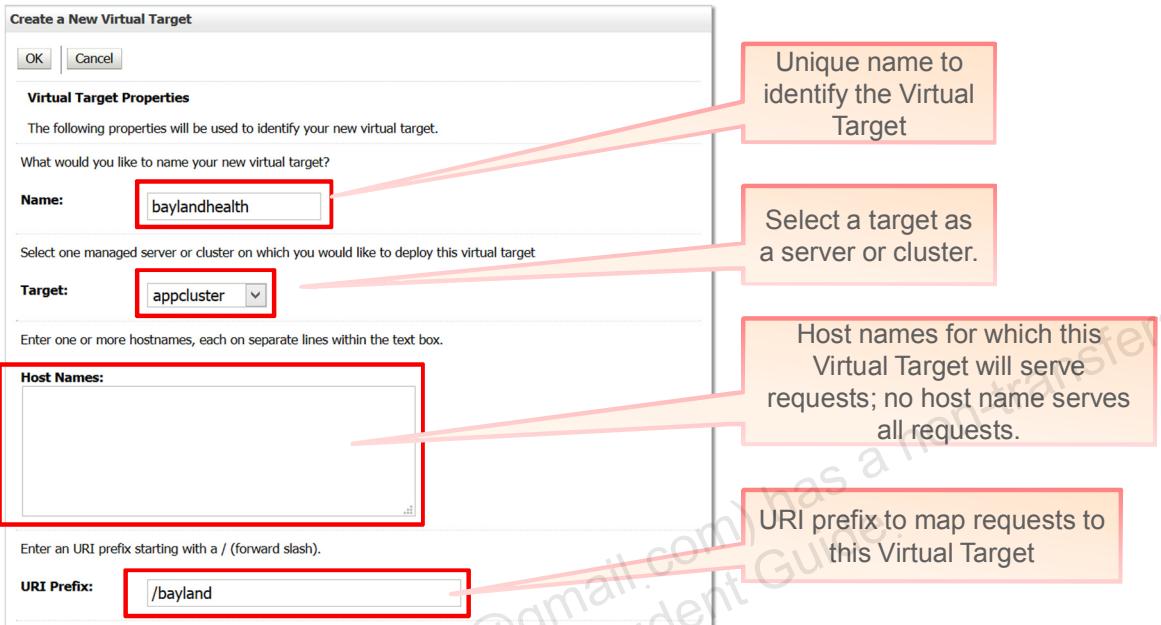
To create a virtual target:

1. In the administration console, select the Domain Structure pane
2. In the Domain Structure pane, expand **Environment > Virtual Targets**.
3. In the Summary of Virtual Targets pane, click **New**.

# Creating Virtual Targets

4

Enter Name, select a Target. Enter URI Prefix and click OK.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

4. Enter a name for this virtual target.

Optionally, enter a URI prefix for this virtual target.

Add one or more host names for the virtual target. The host names you specify must resolve to the target cluster and Managed Servers you specify, or resolve to the OTD Administration Server if you are using OTD for load balancing.

Optionally, specify partition channel and port (explicit or offset) information to use for this virtual target. The channel must exist on the selected target and be configured using the same protocol. If you specify a value for Partition Channel, you must specify either an explicit port or a port offset.

**Note:** These options are ignored if you use OTD to load balance the domain partition.

Select one target from the available targets. You can select a single Managed Server or a single cluster.

5. Click Save
6. Click Activate Changes.

## Creating Domain Partitions

1

Log in to the WebLogic administration console.

2

Within the Domain Structure pane, select **Domain Partitions**.



3

Click **New**.



The screenshot shows the 'Summary of Domain Partitions' pane with two tabs: 'Configuration' and 'Control'. Below the tabs is a table titled 'Domain Partitions (Filtered - More Columns)'. At the bottom of the table are four buttons: 'New' (highlighted with a red box), 'Delete', 'Import', and 'Export'.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To create a Domain Partition:

1. Log in to the WebLogic Server console.
2. Within the Domain Structure pane, select Domain Partitions.
3. In the Summary of Domain Partitions pane, click **New**.

## Creating Domain Partitions

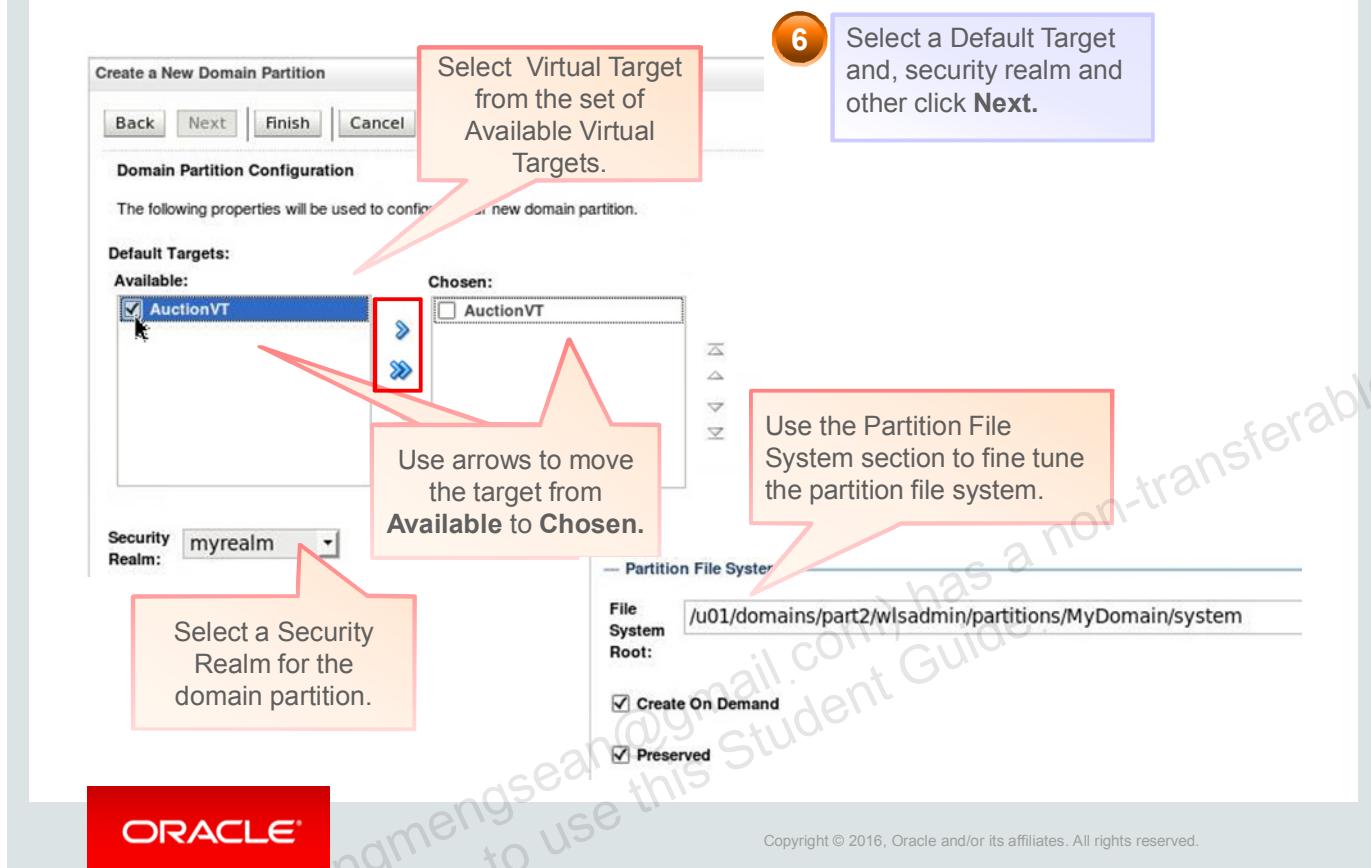
4 Enter the Domain Partition name and click **Next**.

5 Select a Virtual Target and click **Next**.

The screenshot shows two steps of the 'Create a New Domain Partition' wizard. Step 4, 'Domain Partition Properties', has a red box around the 'Name' field containing 'MyDomainPartition'. Step 5, 'Available Targets', shows a list of 'Virtual Targets' with 'AuctionVT' and 'SalesVT' checked. Red boxes highlight the 'Name' field, the 'Default Resource Group' checkbox, the 'Resource Group Template' dropdown, and the 'Virtual Targets' list.

4. Enter the name of the Domain Partition and click **Next**. You may also select an existing resource template.
5. Select a virtual target for the domain partition and click **Next**.

## Creating Domain Partitions



6. Select one of the available virtual targets to use as the default if a resource group in this partition does not explicitly identify one.

Use the arrow keys to move a virtual target from the Available to Chosen list. You can use the up and down arrows to reorder Chosen virtual targets.

Select the security realm for this partition.

The security realm can be unique to this partition, or shared by multiple partitions.

Specify the location of the domain partition file system. The default location is prepopulated.

Select Create On Demand to create the file system at domain partition creation and select Preserved to keep the file system after the partition is deleted.

7. Click **Finish** to finalize the creation of the domain partition.

Note that in order to start any domain partition for the first time, the Administration server must be restarted.

For information on virtual targets see:

[https://docs.oracle.com/middleware/1221/wls/WLSMT/config\\_virtual.htm#WLSMT580](https://docs.oracle.com/middleware/1221/wls/WLSMT/config_virtual.htm#WLSMT580)

For information about resource groups specifying targets see:

[https://docs.oracle.com/middleware/1221/wls/WLSMT/config\\_resource\\_grp.htm#WLSMT1217](https://docs.oracle.com/middleware/1221/wls/WLSMT/config_resource_grp.htm#WLSMT1217)

# Working with Domain Partitions

Domain Partitions may be:

- Started, Stopped, Suspended, or Resumed
- Exported or Imported

To manage a domain partition:

1

In the Domain Structure pane, select **Domain Partitions**.

2

Select the **Control** tab.

3

Select a partition and an operation



Domain partitions support the management and import/export operations.

To start, resume, suspend or shut down a partition:

1. In the Domain Structure pane, select Domain Partitions.
2. In the Summary of Domain Partitions pane, select the Control tab.
3. Select one of the existing domain partitions and using the operations buttons select one of the provided operations. Note that Suspend and Shutdown have both Force Now and "When work completes" suboperations.

# Importing and Exporting Domain Partitions

Domain partitions may be imported or exported.

To import or export a domain partition:

1

In Change Center, select **Lock & Edit**.

2

In the Domain Structure pane, select **Domain Partitions**.

3

In the Summary of Domain Partitions pane, click the **Configuration Tab**.

4

Select a domain partition and an operation.



5

Complete the operation by selecting an export or import path.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Exporting and importing partitions lets you easily move partitions from one domain to another, including the applications that are deployed to the partition. This feature is useful for replicating partitions across domains and for moving domains from a development to a production environment. Domains may be imported or exported using a variety of tools including the WebLogic Console, WLST, REST, and the FMW Console.

To import or export a domain partition, using the WebLogic Console:

1. In Change center, select Lock & Edit.
2. In the Domain Structure pane, select Domain Partitions.
3. In the Summary of Domain Partitions pane, click the Configuration tab.
4. Select a specific operation and a domain to perform it on. Note that import does not require you to select a domain partition.
5. Complete the operation by entering the path to the directory to where the domain should be imported from, or exported to. In addition to the partition itself, applications associated with the partition may be selectively exported or imported.

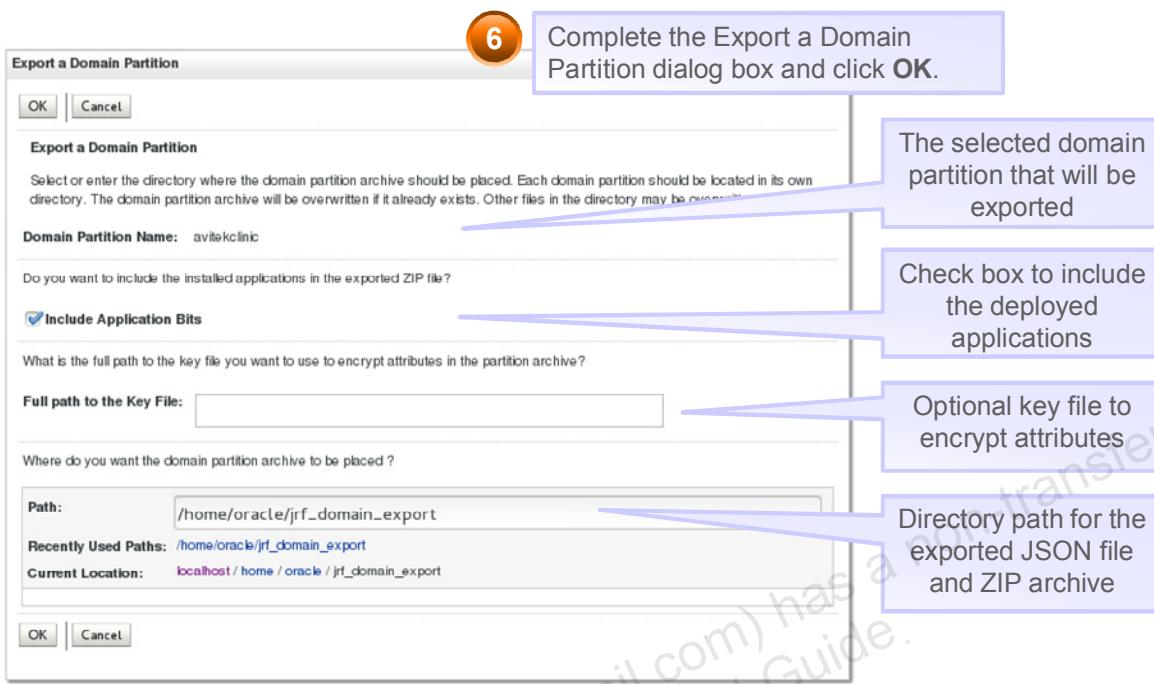
When a partition is exported from the source domain, it is packaged in a partition archive which includes:

- The partition configuration
- Any resource groups contained in the partition
- Any resource group templates referred to by those resource groups

The contents of the partition's file system, *<partition-file-system>/config* directory.

Optionally, application binaries and configurations for applications deployed to the partition.

# Completing Domain Partition Import (or export)



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

6. Complete the Export a Domain Partition dialog box by completing all fields and click OK.
- Provide the full path to the directory in which to save the partition archive.
  - Each domain partition should be located in its own directory. The domain partition archive will be overwritten if it already exists in the specified location. Other files in the directory may be overwritten as well.
  - Optionally, select to include the installed application and library binaries in the exported partition archive.
  - Optionally, enter the full path to a file containing a string to use as the encryption key to encrypt attributes in the partition archive.
  - If you do not provide your own key, a new secret key will be generated and stored in the `expPartSecret` file in the exported `<PartitionName>.zip` file.

# Completing Domain Partition Import (or export)

6

Complete the Import a Domain Partition dialog box and click **OK**.

Check box to create a new Resource Group Template

Enter a new unique Domain Partition Name.

Optional key file to encrypt attributes

Directory path and ZIP file selection for the domain partition archive



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

6. Complete the Import a Domain Partition dialog box by completing all fields and click **OK**. All resource group templates used by the source domain partition are contained in the partition archive and are imported along with the partition into the target domain. If the resource group template already exists in the target domain and you do not specify to overwrite the existing resource group template, the import operation will fail. If Create New is selected, then the resource group will be created with a new name and the import will continue.
- Optionally, enter the full path to a file containing the key to decrypt attributes in the partition archive.

# Application Deployment

- Multitenant application deployment supports the same tools and methods as traditional deployment.
- WebLogic Multitenant supports the following deployment modes:

Mode	Description
Global	Domain-level deployment in a nonpartitioned environment
Resource group template	Domain-level, or partition-level deployment based on a global resource group template
Resource group in a partition	Partition-level development based on a resource group in a partition
Resource group at the domain	Domain-level deployment based on resource group



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You cannot share an application or library between domain partitions: the application or library is available only within the partition. When you deploy the application or library, you specify the resource group in the partition.

The key difference between an application or class running at the domain level and an application or class running in the context of a partition is:

- Applications or classes running at the domain level are available across the domain and are not available in partitions.
- Applications or classes running in the context of a partition are available only within that partition.

## Quiz



Oracle Domain Partitions may be:

- a. Imported
- b. Exported
- c. Imported but not exported
- d. Both

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: d**

Domain partitions may be imported and exported.

# Summary

In this lesson, you should have learned how to:

- Describe Multitenancy concepts
  - Enumerate domain partition concepts
  - Create and manage single domain partitions
  - Associate partitions with virtual targets
  - Compare domain partition security realms options



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 13-1 Overview: Creating and Working with Domain Partitions

This practice covers the following topics:

- Creating a virtual target
- Creating a domain partition
- Deployment and testing an application in a domain partition



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Managing Data Sources

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to:

- Suspend and resume a data source
- Configure a multi data source



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

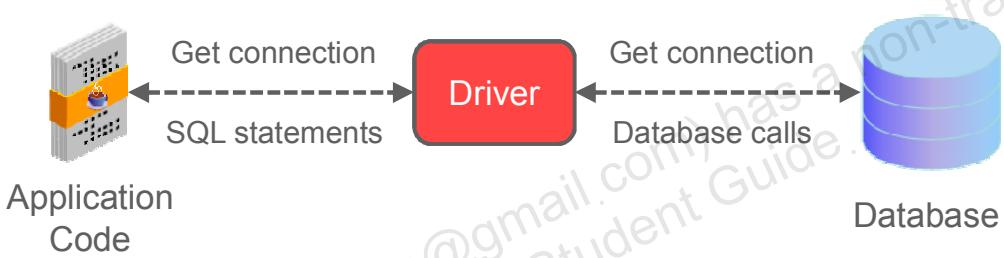
- Review:
  - JDBC
  - Data Source
- Managing Data Sources
- GridLink Data Source Review
- Multi Data Sources
- Review: Connection Testing
- Proxy Data Sources
- Creating a Multi Data Source



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Review: JDBC

- The Java Database Connectivity (JDBC) API:
  - Is a platform- and vendor-independent mechanism for accessing and using a database
  - Provides transparency from proprietary vendor issues
  - Requires the use of a *driver* (a Java class)
- JDBC drivers are supplied with the WebLogic Server installation or by your database vendor.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The JDBC API is the way in Java code to work with SQL. It builds on Open Database Connectivity (ODBC), so developers familiar with ODBC find it easy to use.

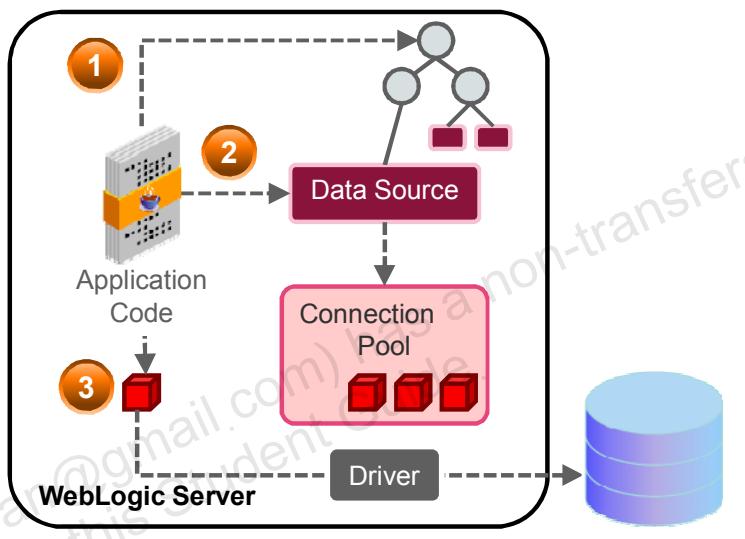
The value of JDBC lies in the fact that an application can access virtually any relational database and run on any platform with a Java Virtual Machine (JVM). That is, with JDBC, it is not necessary to write one program to access a Sybase database, another to access an Oracle database, another to access an IBM DB2 database, and so on. You can write a single program by using the JDBC API. Because the application is written in Java, you need not write different applications to run on different platforms, such as Windows and Linux.

JDBC accomplishes database connections by using a driver mechanism that translates JDBC calls to native database calls. Although most available drivers are fully written in Java (Type 4 drivers), and are thus platform independent, some drivers (Type 2) use native libraries and are targeted to specific platforms.

## Data Source: Review

A data source is a Java object targeted to and managed by one or more instances of WebLogic Server. A deployed data source has connections to a particular database in its connection pool ready-to-go for applications running on those servers.

1. An application looks up the data source in a server's resource tree by using the JNDI API.
2. It asks the data source for a connection.
3. It uses the connection (which uses a driver) to access the database.
4. When finished, it closes the connection (which returns it to the pool).



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A data source is a Java object managed by WebLogic Server and used by application code to obtain a database connection. Retrieving a database connection from a data source is better than getting a connection directly from the database for two reasons:

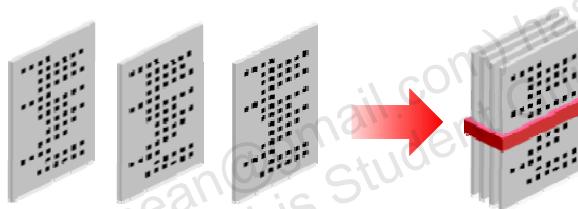
- Connections in a data source's connection pool have already been created. Therefore, the application does not have to wait for connection creation.
- All database-specific information moves out of the application code and into the WebLogic Server configuration, making the code more portable and robust.

Data sources can be created by using one of the WebLogic Server administration tools. A data source is configured with a connection pool that will contain connections to a particular database. It is also targeted to one or more instances of WebLogic Server.

For an application to use one of the connections in a data source's connection pool, first the application looks up the data source in the server's resource tree. The API used is the Java Naming and Directory Interface (JNDI). After the data source is retrieved, the application asks it for a database connection. The data source gives the application one of the connections not currently being used, from its pool of connections. The application uses that connection to access the database. When the application is finished with the connection, it closes it. But rather than close, the connection is returned to the connection pool for some other application to use.

## XA Data Source Review

- Data source objects retrieved by applications via JNDI can be of two types:
  - Non-XA (default)
  - XA
- XA data sources:
  - Will automatically participate in distributed or “global” transactions initiated by the application
  - Typically require the underlying JDBC driver to support XA
  - Can use a non-XA driver in certain scenarios



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

One of the most fundamental features of WebLogic Server is transaction management. Transactions are a means to guarantee that database changes are completed accurately. WebLogic Server protects the integrity of your transactions by providing a complete infrastructure for ensuring that database updates are done accurately, even across a variety of resource managers. If any one of the operations fails, the entire set of operations is rolled back.

If you use global or XA transactions in your applications, you should use an XA JDBC driver to create database connections in the JDBC data source. If an XA driver is unavailable for your database, or you prefer not to use an XA driver, you should enable support for global transactions in the data source. You should also enable support for global transactions if your applications meet any of the following criteria:

- They use the EJB container in WebLogic Server to manage transactions.
- They include multiple database updates within a single transaction.
- They access multiple resources, such as a database and the Java Messaging Service (JMS), during a transaction.
- They use the same data source on multiple servers (clustered or nonclustered).

# Agenda

- Review
- Managing Data Sources
  - Why Manage Data Sources?
  - Suspend a Data Source
  - Resume a Data Source
- GridLink Data Source Review
- Multi Data Sources
- Review: Connection Testing
- Proxy Data Sources
- Creating a Multi Data Source



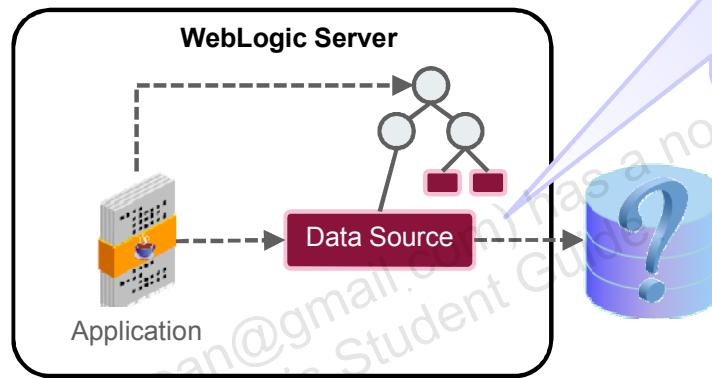
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Why Manage a Data Source?

You may require managing the connection pool of your data source for several reasons:

- Database maintenance
- Troubleshooting
- Production issues

- Suspend
- Resume
- Reset
- Shrink
- Stop
- Start



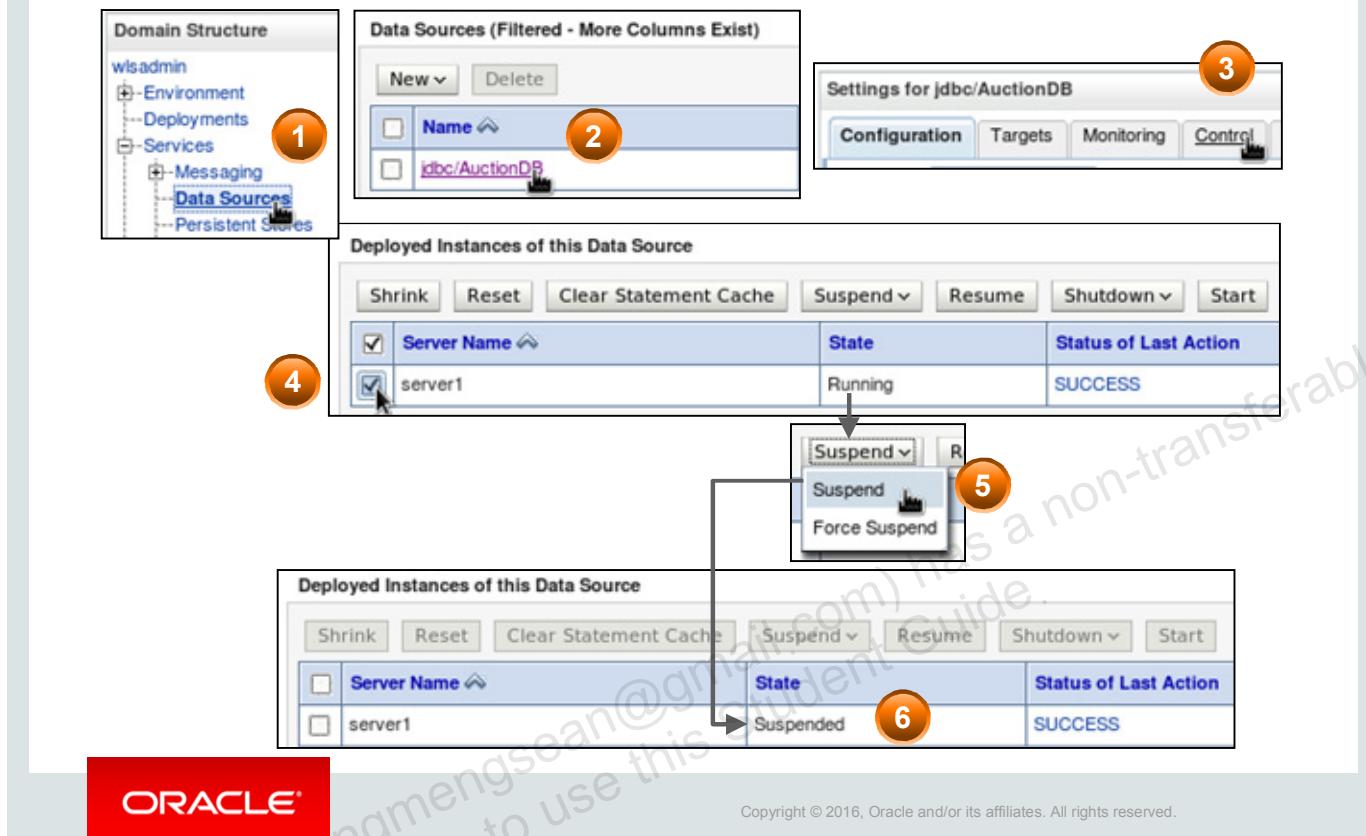
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

There are several operations that you can perform on a data source. The following commands enable you to control your data source to keep your applications running smoothly:

- **Suspend:** Marks the data source as disabled, so applications cannot use connections from the pool
- **Resume:** Marks the data source as enabled, so applications can use connections from the pool
- **Reset:** Closes and re-creates all database connections in a connection pool
- **Shrink:** Shrinks the connection pool to the greater of `minCapacity` or the number of connections in use
- **Stop:** Shuts down a data source and associated connections. This can be a graceful or forced shutdown.
- **Start:** Re-initializes a data source that was previously shut down

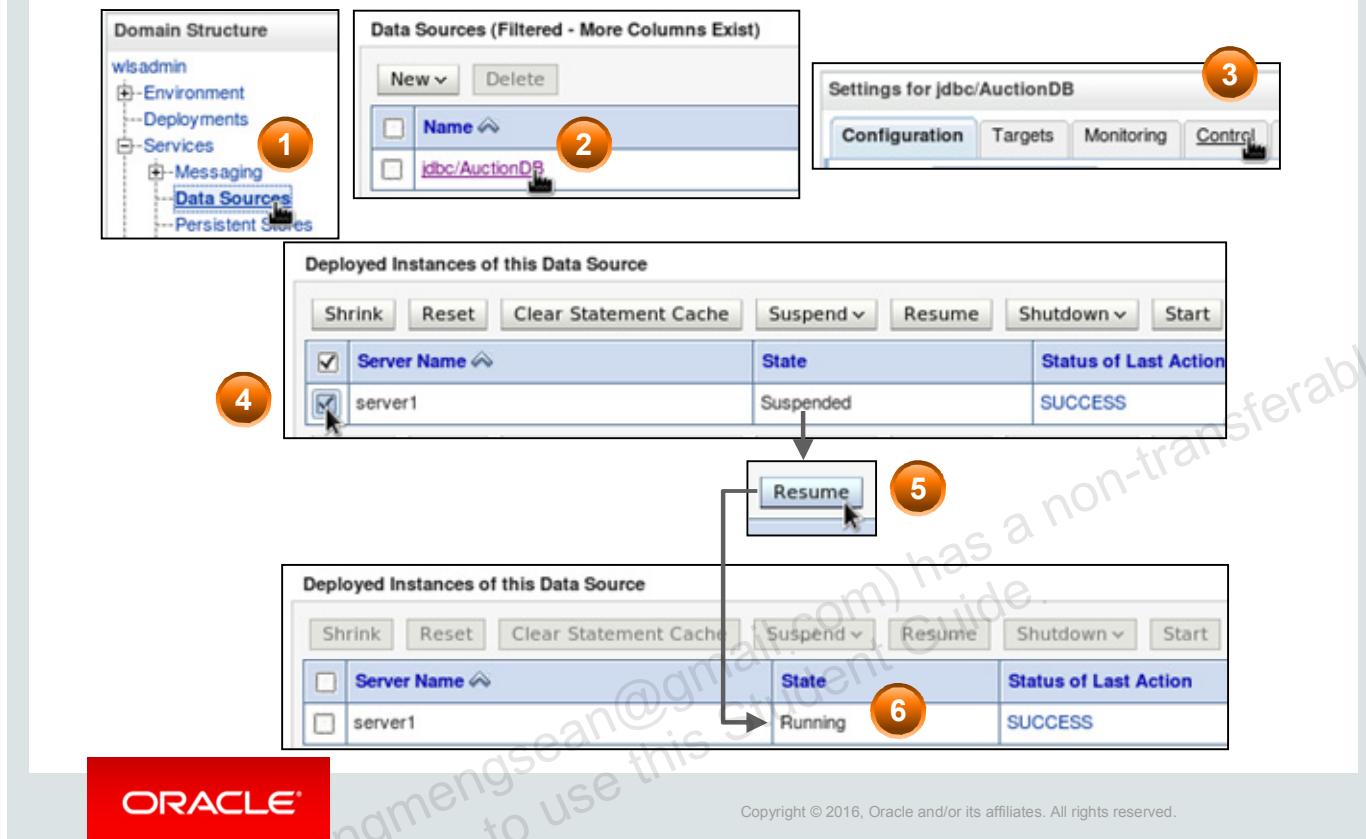
# Suspend a Data Source



To suspend a data source instance by using the administration console, perform the following steps:

1. Select Services > Data Sources.
2. Select a data source from the list of available data sources.
3. Click the data source's Control tab.
4. Select the target where the data source instance resides that you want to suspend.
5. Click Suspend and choose to suspend gracefully or forcefully:
  - **Suspend > Suspend:** Marks the data source instance as disabled and blocks new connection requests. All connections are preserved and are viable again when the data source is resumed.
  - **Suspend > Force Suspend:** Marks the data source instance as disabled and destroys all connections. All transactions are rolled back. The data source attempts to re-create the connections in preparation for a subsequent resume operation.
6. Ensure that the data source is in the Suspended state.

# Resume a Data Source



To resume a suspended data source instance by using the administration console, perform the following steps:

1. Select Services > Data Sources.
2. Select a data source from the list of available data sources.
3. Click the data source's Control tab.
4. Select the target of the suspended data source.
5. Click the Resume button to return the data source to a Running state:
  - If the data source was suspended gracefully, all connections are preserved and are usable again.
  - If the data source was suspended forcefully, all clients must reserve new connections to perform work.
6. Ensure that the data source is in the Running state.

# Practice 14-1 Overview: Controlling a Data Source

This practice covers the following topics:

- Suspending a data source
- Resuming a data source

# Agenda

- Review
- Managing Data Sources
- GridLink Data Source Review
- Multi Data Sources
- Review: Connection Testing
- Proxy Data Sources
- Creating a Multi Data Source

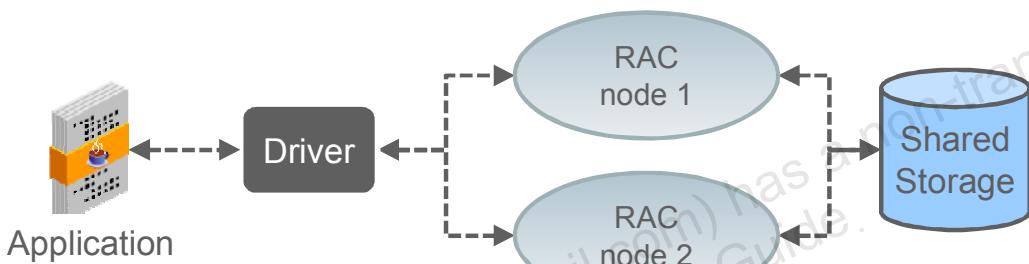


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle Real Application Clusters (RAC): Overview

## Oracle RAC:

- Supports multiple Oracle database servers for greater scalability
- Relies on database servers having access to a shared and highly available storage device



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

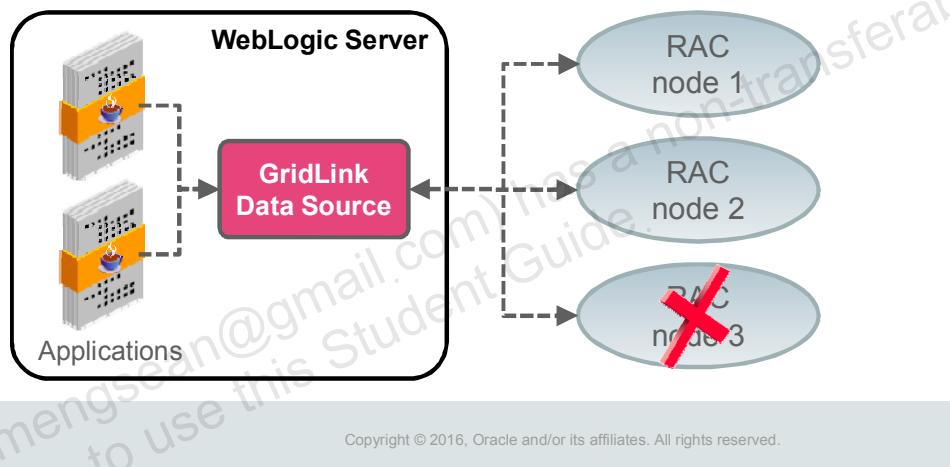
Oracle Real Application Clusters (RAC) is software that enables users on multiple machines to access a single database with increased reliability. RAC is made up of two or more Oracle database instances running on two or more clustered machines that access a shared storage device via cluster technology. To support this architecture, the machines that host the database instances are linked by a high-speed interconnect to form the cluster. This interconnect is a physical network used as a means of communication between the nodes of the cluster. Cluster functionality is provided by the operating system or compatible third-party clustering software.

Because every RAC node in the cluster has equal access and authority, the loss of a node may impact performance, but does not result in down time.

## Oracle GridLink for RAC

WebLogic Server's *GridLink* data source is "RAC-aware." It:

- Performs intelligent load balancing based on the current RAC workload
- Implements RAC's Fast Connection Failover (FCF) pattern
- Ensures that all database operations within a global transaction are routed to the same RAC node ("XA affinity")



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A single GridLink data source provides connectivity between WebLogic Server and an Oracle database service that has been targeted to an Oracle RAC cluster. This type of data source automatically adjusts the distribution of work based on the current performance metrics reported by each RAC node, such as CPU usage, availability, and response time. If this capability is disabled, GridLink data sources instead use a round-robin, load-balancing algorithm to allocate connections to RAC nodes.

A GridLink data source implements Oracle's Fast Connection Failover (FCF) pattern, which:

- Provides rapid failure detection
- Aborts and removes invalid connections from the connection pool
- Performs graceful shutdown for planned and unplanned Oracle RAC node outages
- Adapts to changes in topology, such as adding or removing a node
- Distributes runtime work requests to all active Oracle RAC instances, including those rejoining a cluster

XA affinity ensures that all the database operations performed on a RAC cluster within a global transaction are directed to the same RAC instance. This increases performance and also helps ensure data integrity after a failure.

# Agenda

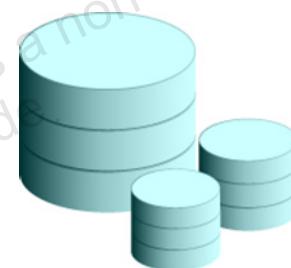
- Review
- Managing Data Sources
- GridLink Data Source Review
- Multi Data Sources
  - Multi Data Source Architecture
  - Comparison of GridLink and Multi Data Sources
  - Multi Data Source Failover Option
  - Multi Data Source Load Balancing Option
- Review: Connection Testing
- Proxy Data Sources
- Creating a Multi Data Source



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Multi Data Sources

- To avoid a single point of failure and to achieve greater scalability, many enterprises employ multiple database servers.
- A multi data source:
  - Is a pool of data sources
  - Is used by applications exactly like a standard data source
  - Transparently provides load balancing or failover across the member data sources
  - Can be XA or non-XA



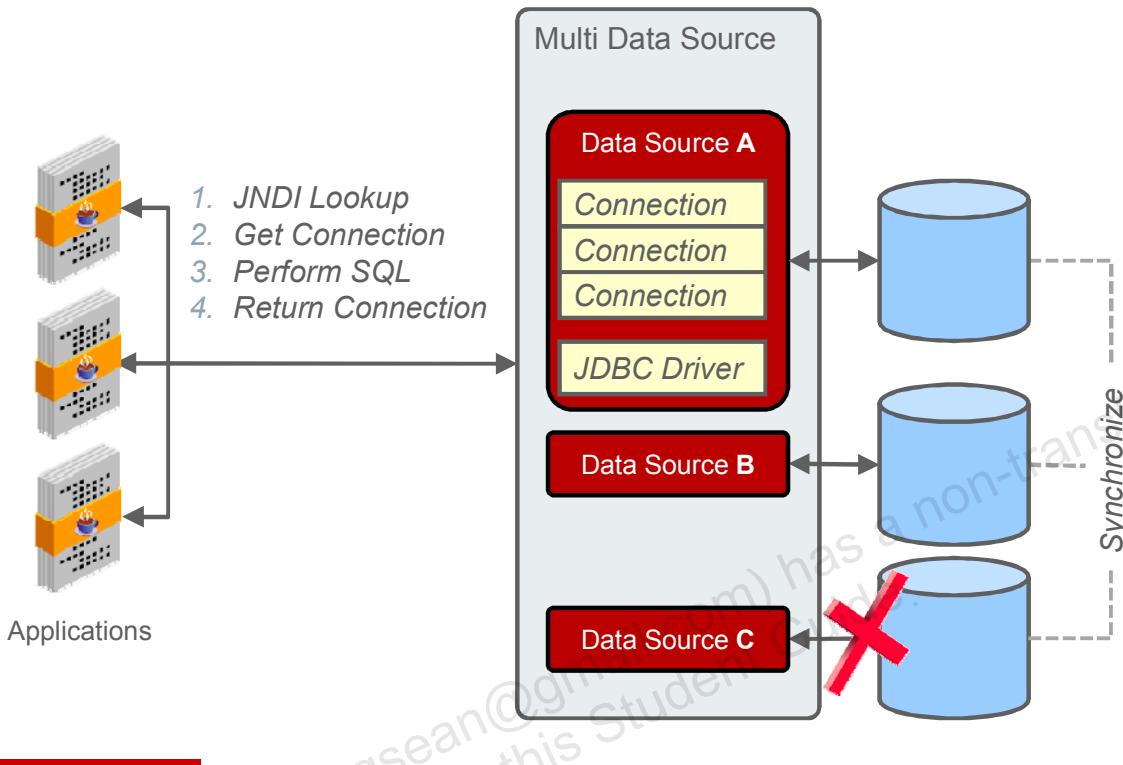
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A multi data source is an abstraction around a group of data sources that provides load balancing or failover processing between the data sources associated with the multi data source. Multi data sources are bound to the JNDI tree or local application context just like data sources are bound to the JNDI tree. Applications look up a multi data source on the JNDI tree just like they do for data sources, and then request a database connection. The multi data source determines which data source to use to satisfy the request depending on the algorithm selected in the multi data source configuration: load balancing or failover.

All data sources used by a multi data source to satisfy connection requests must be deployed on the same servers and clusters as the multi data source. A multi data source always uses a data source deployed on the same server to satisfy connection requests. Multi data sources do not route connection requests to other servers in a cluster or in a domain. To deploy a multi data source to a cluster or server, you select the server or cluster as a deployment target. When a multi data source is deployed on a server, WebLogic Server creates an instance of the multi data source on the server. When you deploy a multi data source to a cluster, WebLogic Server creates an instance of the multi data source on each server in the cluster.

# Multi Data Source Architecture



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A multi data source can be thought of as a pool of data sources. Multi data sources are best used for failover or load balancing between nodes of a highly available database system, such as redundant databases. Multi data sources do not provide any synchronization between databases. It is assumed that database synchronization is handled properly outside of WebLogic Server so that data integrity is maintained.

You create a multi data source by first creating data sources, then creating the multi data source by using the administration console or the WebLogic Scripting Tool (WLST), and then assigning the data sources to the multi data source.

The data source member list for a multi data source supports dynamic updates. You can remove a database node and corresponding data sources without redeployment. This capability provides you the ability to shut down a node for maintenance or shrink a cluster.

Some examples of database replication technologies include Oracle Streams, Oracle Golden Gate, Oracle Data Guard, IBM InfoSphere, Sybase Replication Server, and MySQL Replication.

# Comparison of GridLink and Multi Data Sources

GridLink and multi data sources are used for different scenarios:

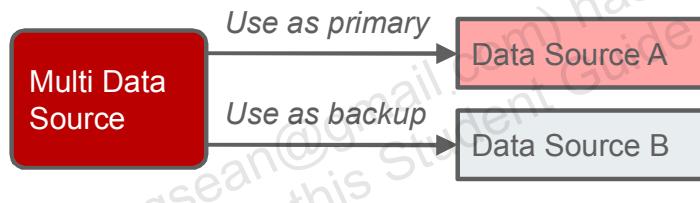
GridLink	Multi Data Source
Used only for Oracle RAC databases	Used for any database that supports replication
Specialized to leverage Oracle RAC features	No specialization for any databases
Provides high availability for access to a database that is viewed as a single database	Provides high availability for multiple databases that are synchronized using another technology



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Failover Option

- The multi data source uses the first member data source to handle all connection requests.
- During a connection request, the other member data sources are tried in succession if the current data source:
  - Becomes unavailable
  - Has no unused connections (optional)
  - Is suspended by the administrator
- If a connection fails when in use, the application must still handle it programmatically.



ORACLE®

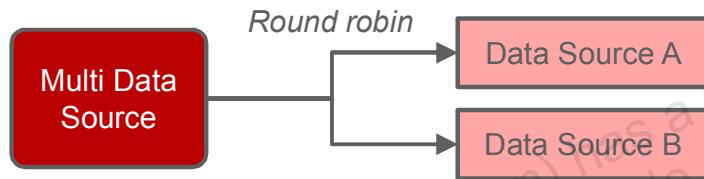
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The multi data source failover algorithm provides an ordered list of data sources to use to satisfy connection requests. Normally, every connection request to this kind of multi data source is served by the first data source in the list. If a database connection test fails and the connection cannot be replaced, or if the data source is suspended, then a connection is sought sequentially from the next data source on the list.

JDBC is a highly stateful client-DBMS protocol, in which the DBMS connection and transactional state are tied directly to the socket between the DBMS process and the client (driver). Therefore, it is still possible for a connection to fail after being reserved, in which case your application must handle the failure. WebLogic Server cannot provide failover for connections that fail while being used by an application. Any failure while using a connection requires that the application code handle it, such as restarting the transaction.

## Load Balancing Option

- The multi data source selects member data sources to satisfy connection requests using a round-robin scheme.
- Data source failure is handled in the same way as described for the Failover option.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The load balancing option is suitable for scenarios where a single database resource may not be able to keep up with the load of the system. The multi data source balances the load across databases to evenly distribute the workload across a scaled out data tier.

Connection requests to a load-balancing multi data source are served from any data source in the list. The multi data source selects member data sources to satisfy connection requests using a round-robin scheme. When the multi data source provides a connection, it selects a connection from the data source listed just after the last data source that was used to provide a connection. Multi data sources that use the load balancing algorithm also fail over to the next data source in the list if a database connection test fails and the connection cannot be replaced, or if the data source is suspended.

## Quiz



Which of the following is true when a client has a reserved connection from a multi data source and the database for that connection crashes?

- a. JDBC calls from the client fail over to another data source.
- b. JDBC calls from the client fail and the client must reserve a new connection.
- c. The multi data source provides the client with a new connection to another data source.
- d. JDBC calls from the client hang.

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

# Agenda

- Review
- Managing Data Sources
- GridLink Data Source Review
- Multi Data Sources
- Review: Connection Testing
- Proxy Data Sources
- Creating a Multi Data Source



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Connection Testing

To provide failover, multi data sources require that all member data sources be configured to use connection testing.

Settings for DataSourceA

Configuration Targets Monitoring Control Security Notes

General Connection Pool Transactions

Advanced

Test Connections On Reserve

Test Frequency: 120

Test Table Name: SQL SELECT 1 FROM DUAL

Seconds to Trust an Idle Pool Connection: 10

Seconds to trust a connection before testing again: 120

Test before giving a connection to application.

Test connections periodically.

Table name or custom SQL to test connection

Seconds to trust a connection before testing again

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

ORACLE

Data sources rely on the Test Reserved Connections feature to know when database connectivity is lost. Testing reserved connections must be enabled and configured for all the data sources within the multi data source. WebLogic Server will test each connection before giving it to an application. With the failover algorithm, the multi data source uses the results from connection test to determine when to fail over to the next data source in the multi data source. After a test failure, the data source attempts to re-create the connection. If that attempt fails, the multi data source fails over to the next data source.

- **Test Frequency:** You can enable periodic background connection testing by entering the number of seconds between periodic tests.
- **Test Reserved Connections:** Select this check box to test the database connection before giving it to your application when your application requests a connection from the data source.

- **Test Table Name:** Enter the name of a small table to use in a query to test database connections. The standard query is `select count(*) from <table>`. Most database servers optimize this SQL to avoid a full table scan, but it is still a good idea to use the name of a table that is known to have few rows, or even no rows. If you prefer to use a different query as a connection test, enter SQL followed by a space and the SQL code that you want to use to test database connections.
- **Seconds to Trust an Idle Connection Pool:** This specifies the number of seconds that WebLogic trusts that an already reserved idle connection is valid before performing another connection test.

# Agenda

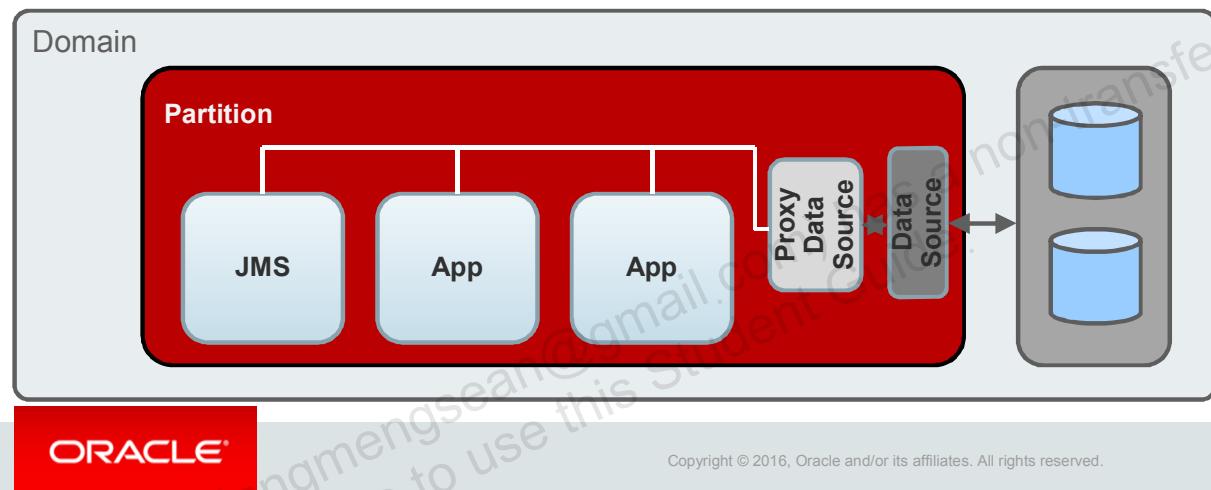
- Review
- Managing Data Sources
- GridLink Data Source Review
- Multi Data Sources
- Review: Connection Testing
- Proxy Data Sources
- Creating a Multi Data Source



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

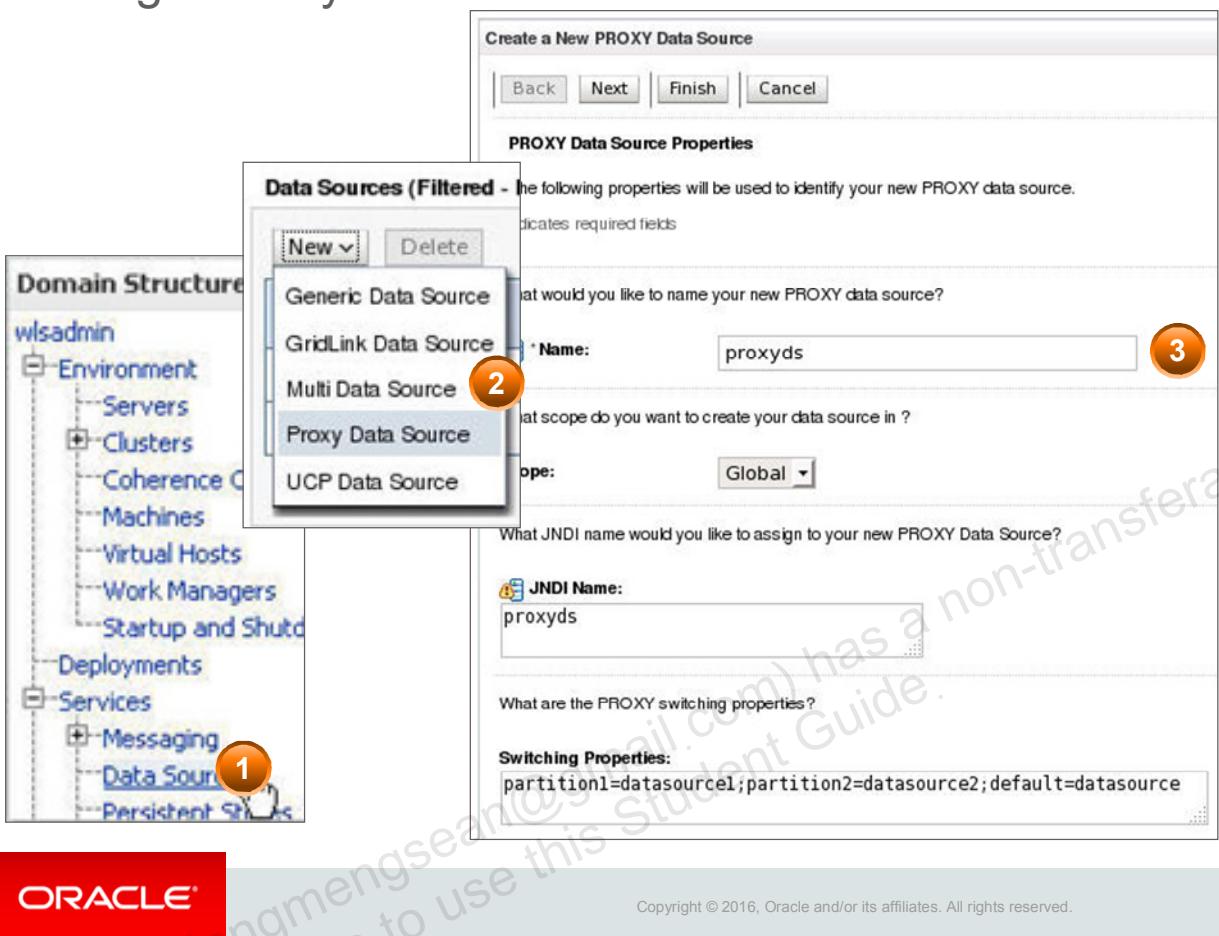
## Proxy Data Sources

- Provide a lightweight mechanism for accessing data sources bound to partitions
- Allow quick access to a data source by name
- Avoid applications needing to know details of the underlying datasource
- Require minimal configuration and overhead



In WebLogic Server Multitenant environments, data sources are replicated for each partition. A proxy data source provides a mechanism for access to a data source associated with a partition or tenant. It enables access to a data source without the need to have naming conventions such as context names, partitions, or tenants. Proxy data sources simplify the administration of multiple data sources by providing a lightweight mechanism for accessing a data source associated with a partition or tenant. Applications often need to quickly access a data source by name without needing to know the naming conventions, context names (partitions or tenants), and so on. A proxy data source provides access to the underlying data sources. All of the significant processing happens in the data sources to which it points. That is, the underlying data sources actually handle deployment, management, security, and so on.

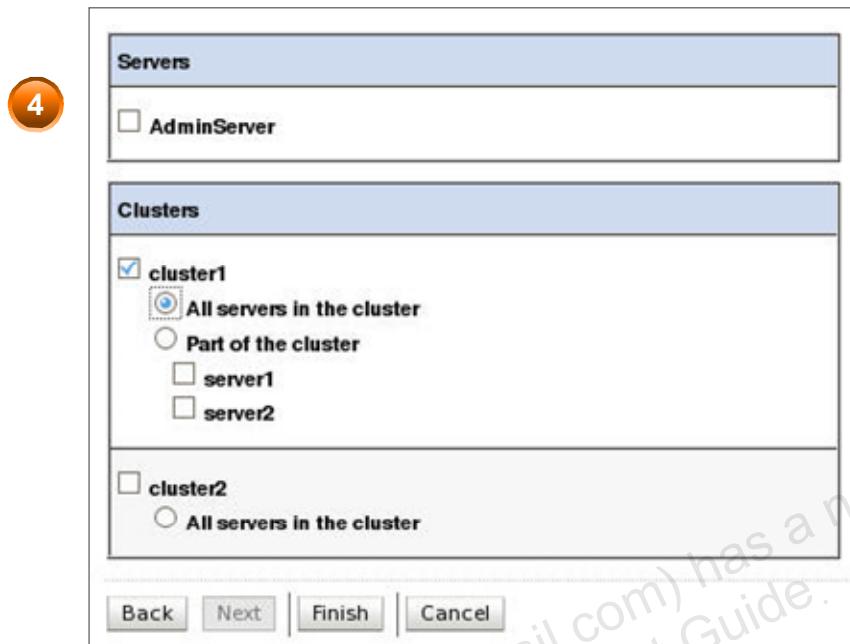
# Creating a Proxy Data Source



To create a proxy data source after clicking **Lock & Edit** in the Change Center, perform the following tasks:

1. In the Domain Structure tree, expand **Services** and then select **Data Sources**.
2. Above (or below) the table of data sources, click the **New** drop-down list and select **Proxy Data Source**.
3. On the first page of the data source creation wizard, enter or select the following information and then click **Next**:
  - **Name:** The configuration name for this data source
  - **JNDI Name:** The JNDI “binding name” for this data source. Applications look up the data source on the server’s JNDI tree by this name. The name can include contexts by placing dots in the string. Note that the name and JNDI name can be different.
  - **Switching Properties:** Enter the switching properties to be passed to the switching callback method for the Proxy data source. This value is dependent on the requirements of the switching callback. The format of the proxy switching properties is *partition1=datasource1;partition2=datasource2;...;default=datasource*.

## Creating a Proxy Data Source



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

4. Select the servers or clusters to which the data source should be deployed and then click **Finish**. If no servers are targeted, the data source is created, but not deployed. You will need to target it later. As usual, to confirm the changes, in the Change Center, click **Activate Changes**.

# Monitoring Proxy Data Source JDBC Resources

The screenshot shows the Oracle WebLogic Server Administration Console interface. A large watermark reading "Search for a license to use this Student Guide" is overlaid across the page.

**Step 1:** The left sidebar shows the "Domain Structure" with the "Services" node expanded, revealing "Data Sources". A circled "1" is next to the "Data Sources" link.

**Step 2:** The main content area displays the "Summary of JDBC Data Sources" page. It has tabs for "Configuration" and "Monitoring", with "Monitoring" selected. A circled "2" is next to the "Monitoring" tab. Below the tabs, a brief description of a JDBC data source is provided. A table titled "Data Sources (Filtered - More Columns Exist)" lists one entry: "proxyds" (Type: Proxy, JNDI Name: proxyds, Targets: cluster1, Scope: Global). Buttons for "New" and "Delete" are shown below the table.

**Step 3:** A modal window titled "Settings for proxyds" is open. It has tabs for "Configuration", "Targets", "Monitoring" (selected), and "Notes". A circled "3" is next to the "Monitoring" tab. The "Monitoring" tab contains a "Statistics" sub-tab and a "Testing" sub-tab. A message states "This page displays statistics associated with this JDBC data source". A "Customize this table" link is also present.

**ORACLE**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can monitor a variety of statistics for each data source instance in your domain, such as the current number of database connections in the connection pool, current number of connections in use, and the longest wait time for a database connection.

To view the statistics of a Proxy JDBC Data source:

1. In the Domain tree, expand Services node, then select Data Sources.
2. On the Summary of Data Sources page, click the Proxy Data Source name.
3. Select the Monitoring: Statistics tab. Statistics of the deployed instance of the Proxy Data source are displayed.

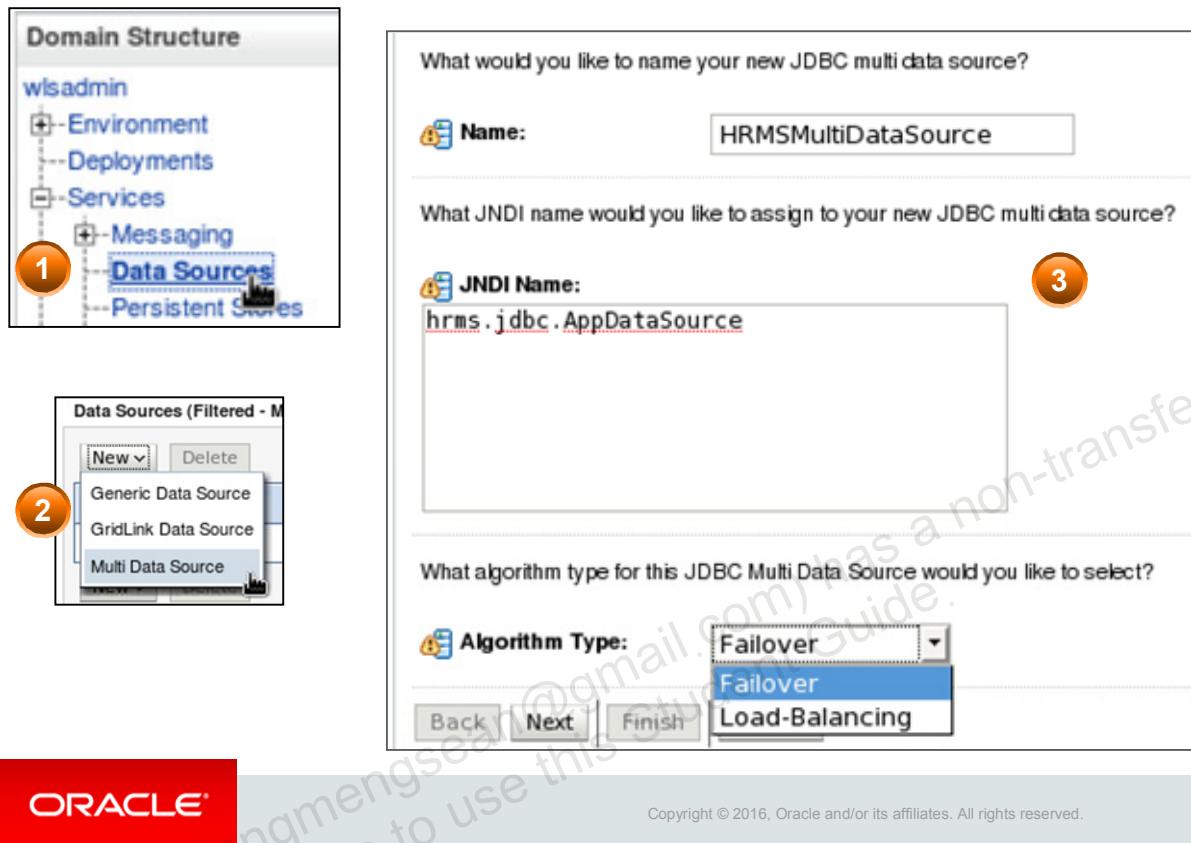
# Agenda

- Review
- Managing Data Sources
- GridLink Data Source Review
- Multi Data Sources
- Proxy Data Sources
- Review: Connection Testing
- (Optional) Creating a Multi Data Source



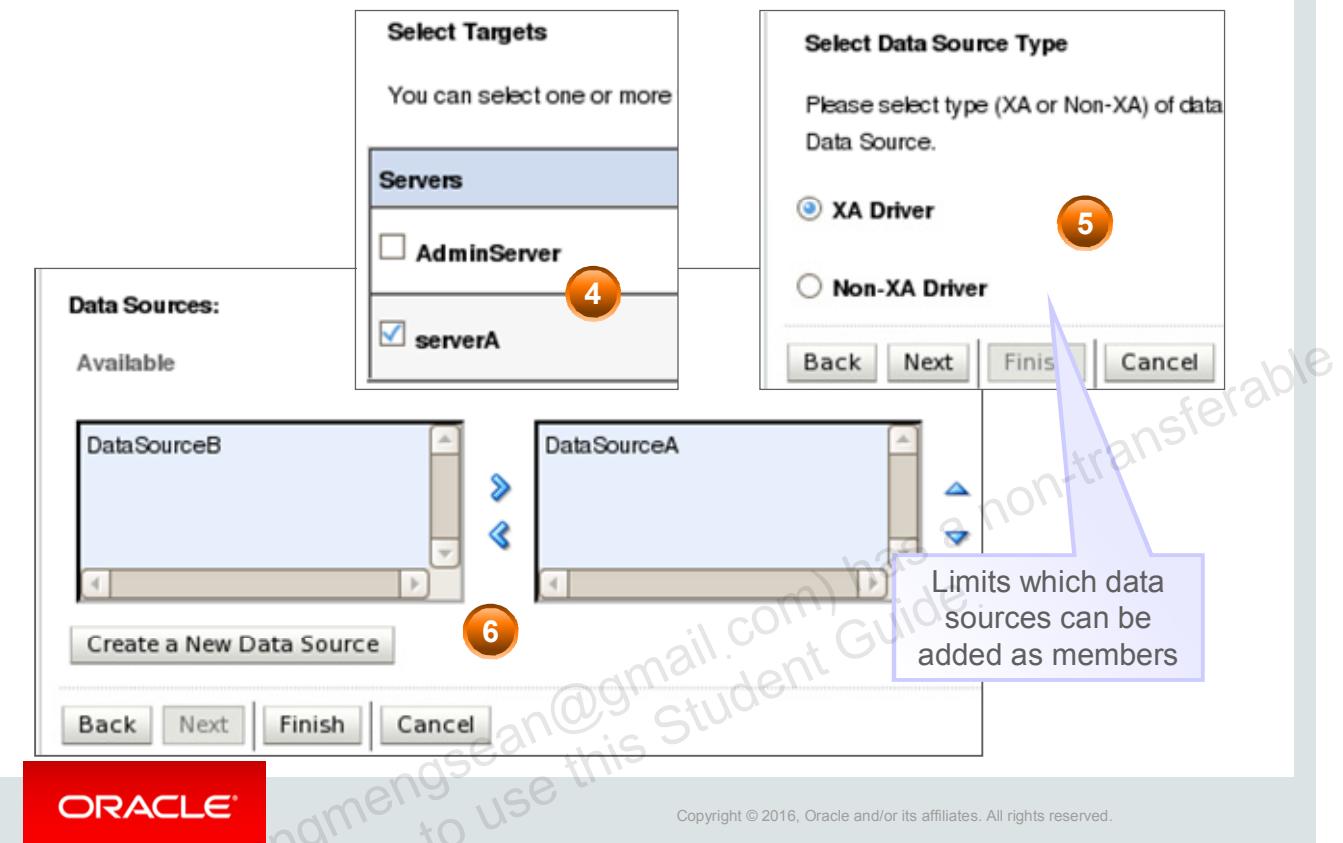
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Creating a Multi Data Source



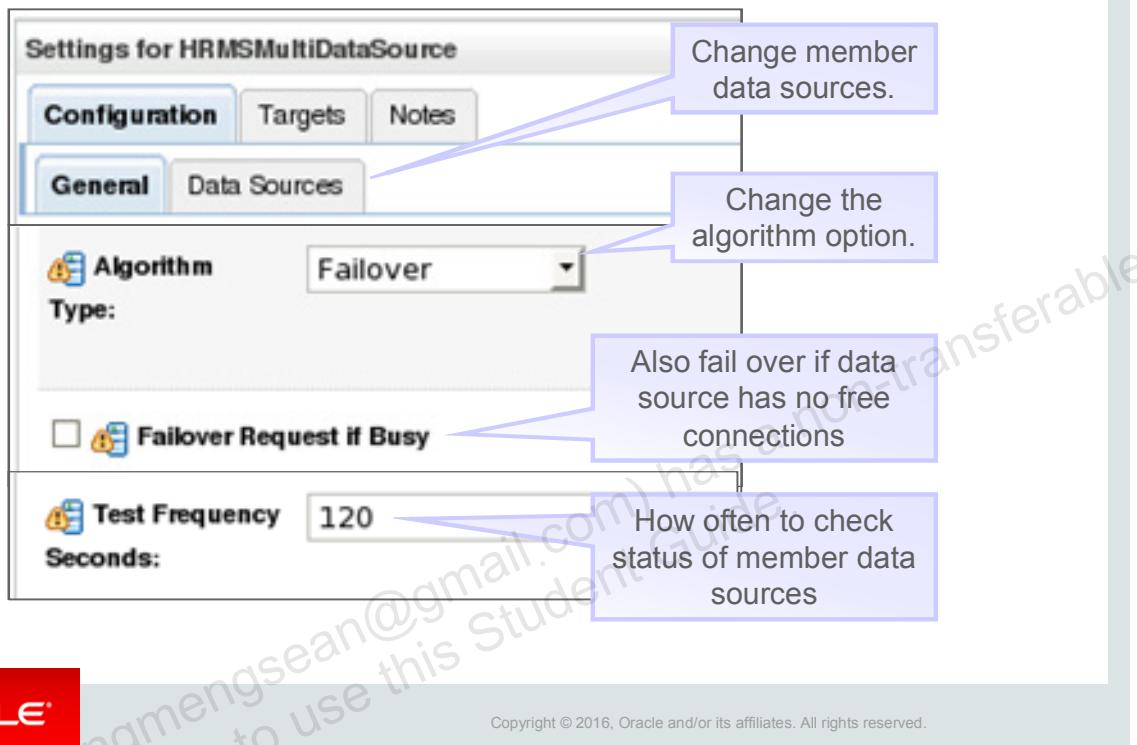
1. In the Domain Structure panel, expand Services, and then select Data Sources.
2. Click New > Multi Data Source.
3. Enter or select the following information and click Next:
  - **Name:** Enter a unique name for this JDBC multi data source.
  - **JNDI Name:** Enter the JNDI path to where this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection. To specify multiple JNDI names for the multi data source, enter each on a separate line.
  - **Algorithm Type:** Select an algorithm option:
    - **Failover:** The multi data source routes connection requests to the first data source in the list; if the request fails, the request is sent to the next data source in the list, and so forth.
    - **Load-Balancing:** The multi data source distributes connection requests evenly to its member data sources.

# Creating a Multi Data Source



4. Select the servers or clusters on which you want to deploy the multi data source. The targets that you select will limit the data sources that you can select as part of the multi data source. You can select only data sources that are deployed to the same targets as the multi data source. Click Next.
5. Select one of the following options and click Next. The option that you select limits the data sources that you can select as part of the multi data source in a later step. Limiting data sources by JDBC driver type enables the WebLogic Server transaction manager to properly complete or recover global transactions that use a database connection from a multi data source:
  - **XA Driver:** The multi data source will use only data sources that use an XA JDBC driver to create database connections.
  - **Non-XA Driver:** The multi data source will use only data sources that use a non-XA JDBC driver to create database connections.
6. Select the data sources that you want the multi data source to use to satisfy connection requests. Then use the supplied arrow buttons to reorder the chosen data source list as desired and click Finish. For convenience, you can also create new data sources at this time using the “Create a New Data Source” button.

# Configuring a Multi Data Source



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. In the Domain Structure panel, expand Services, and then select Data Sources.
2. Click the existing multi data source.
3. Edit any of the following fields, and then click Save. In general, changes take effect after you redeploy the multi data source or restart the server:
  - **JNDI Name:** Same field as when creating a new multi data source
  - **Algorithm Type:** Same field as when creating a new multi data source
  - **Failover Request if Busy:** For multi data sources with the failover algorithm, enables the multi data source to fail over connection requests to the next data source if all connections in the current data source are in use
  - **Test Frequency Seconds:** The number of seconds between when WebLogic Server tests unused connections (requires that you specify a test table name). Connections that fail the test are closed and reopened to reestablish a valid physical connection. If the test fails again, the connection is closed. In the context of multi data sources, this attribute controls the frequency at which WebLogic Server checks the health of data sources it had previously marked as unhealthy. When set to 0, the feature is disabled.

# Multi Data Source WLST Example

Create a new multi data source:

```
edit()
startEdit()

jdbcSystemResource = cmo.create('HRMSMultiDataSource',
    'JDBCSYSTEMRESOURCE')
jdbcResource = jdbcSystemResource.getJDBCResource()
jdbcResource.setName('HRMSMultiDataSource')
jdbcResourceParams = jdbcResource.getJDBCDataSourceParams()
jdbcResourceParams.setJNDINames(['jdbc.hr.HRMSDS'])
jdbcResourceParams.setAlgorithmType('Failover')
jdbcResourceParams.setDataSourceList('DataSourceA',
    'DataSourceB')
jdbcSystemResource.addTarget(getMBean('/Servers/serverA'))

save()
activate(block='true')
```

CreateMultiDataSource.py



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The same JDBCSYSTEMRESOURCEMBean used to configure standard data sources is also used to configure multi data sources. This MBean has a child MBean of type JDBCDataSourceBean, which in turn has a child of type JDBCDataSourceParamsBean. Several of these parameters are applicable only to multi data sources, including:

- AlgorithmType
- DataSourceList
- FailoverRequestIfBusy

# Managing Multi Data Source Members

To add a new database node:

1. Start the database.
2. Create a new data source.
3. Add the data source to the multi data source members list.

To remove a database node:

1. Remove the data source from the multi data source members list.
2. When all transactions have completed, suspend and then shut down the corresponding data source.
3. Shut down the database.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The data source member list for a multi data source supports dynamic updates. This allows environments to add and remove database nodes and corresponding data sources without restarting the server or losing connectivity to other member data sources. For example, situations may arise in which you need to grow and shrink your multi pool in response to throughput, or shut down multi pool nodes for maintenance. You can do all the required steps in one configuration edit session.

To improve performance when a data source within a multi data source fails, WebLogic Server automatically disables the data source when a pooled connection fails a connection test. After a data source is disabled, WebLogic Server does not route connection requests from applications to the data source. Instead, it routes connection requests to the next available data source listed in the multi data source.

After a data source is automatically disabled because a connection failed a connection test, the multi data source periodically tests a connection from the disabled data source to determine when the data source (or underlying database) is available again. When the data source becomes available, the multi data source automatically re-enables the data source and resumes routing connection requests to the data source, depending on the multi data source algorithm and the position of the data source in the list of included data sources. Frequency of these tests is controlled by the Test Frequency Seconds attribute of the multi data source.

## Quiz

Q

Which of the following is a data source attribute associated with connection testing?

- a. Test Statement
- b. Test Capacity
- c. Test Table Name
- d. Test LLR

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Quiz



Name three attributes used to configure a multi data source.

- a. Statement Cache Type
- b. Logging Last Resource
- c. Algorithm Type
- d. Failover Request if Busy
- e. JNDI Name

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer:** c, d, e

## Summary

In this lesson, you should have learned how to:

- Suspend and resume a data source
- Configure a multi data source



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 14-2 Overview: (Optional) Creating and Using a Multi Data Source

This practice covers the following topics:

- Creating a multi data source
- Adding data sources to a multi data source
- Configuring a multi data source for failover
- Testing data source failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 14-3 Overview: (Optional) Creating and Using a Proxy Data Source

This practice covers the following topics:

- Creating a proxy data source
- Modifying an application to use a proxy datasource
- Deploying and testing a proxy database based application



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Diagnostic Framework

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to configure the WebLogic Diagnostic Framework (WLDF) to monitor a domain.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- General Architecture
  - WebLogic Diagnostics Framework (WLDF)
  - Diagnostic Archives
  - Diagnostics Modules
- Diagnostic Images
- Harvesters
- Policies and Actions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## WebLogic Diagnostics Framework (WLDF)

- WLDF provides a generic framework to gather and analyze WebLogic runtime data for monitoring and troubleshooting.
- Use WLDF to:
  - Capture a snapshot of key server metrics for distribution to support personnel
  - Capture metrics at specific code points in WebLogic or your application
  - Periodically record selected MBean attributes
  - Send notifications when attributes meet certain conditions

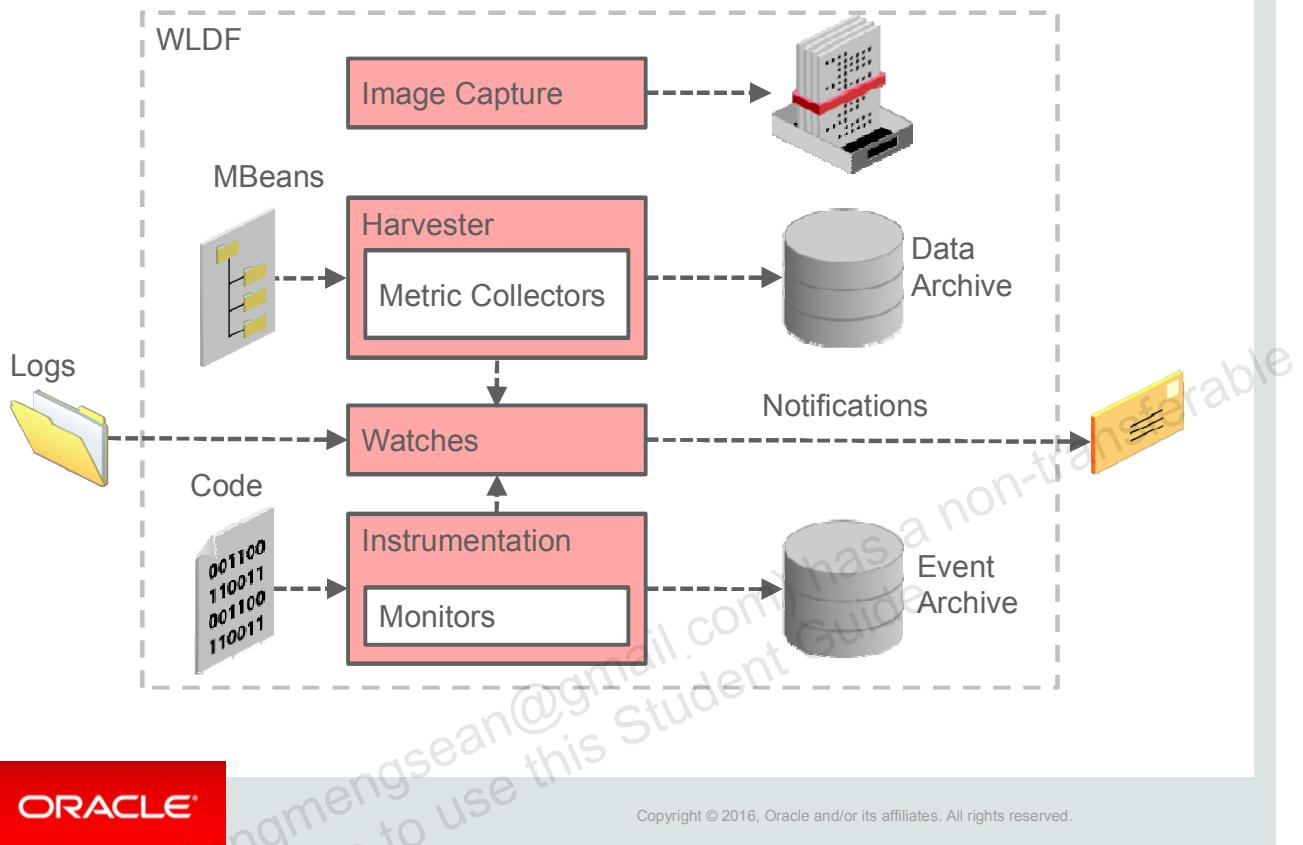


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WLDF consists of several components that work together to collect, archive, and access diagnostic information about a WebLogic Server instance and the applications it hosts.

# WLDF Architecture



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Data creators generate diagnostic data, which is consumed by the logger and the harvester. Those components coordinate with the archive to persist the data, and they coordinate with the watch and notification subsystem to provide automated monitoring. The data accessor interacts with the logger and the harvester to expose current diagnostic data and with the archive to present historical data. MBeans make themselves known as data providers by registering with the harvester. Collected data is then exposed to both the watch and notification system for automated monitoring and to the archive for persistence.

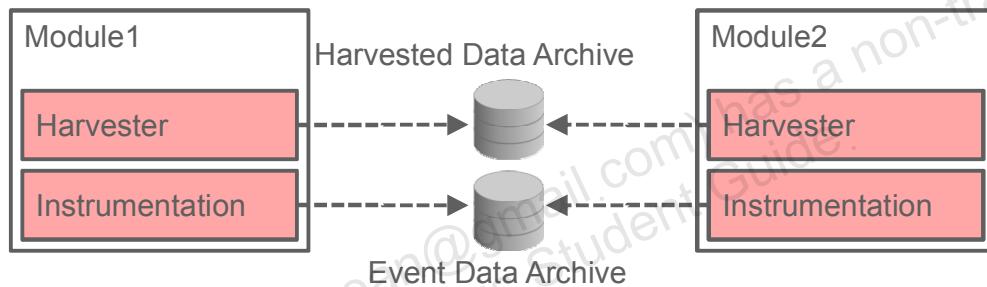
The instrumentation system creates monitors and inserts them at well-defined points in the flow of code execution within the JVM. These monitors trigger events and publish data directly to the archive. They can also take advantage of watches and notifications.

Diagnostic image capture support gathers the most common sources of the key server state used in diagnosing problems. It packages that state into a single artifact, which can be made available to support technicians.

The past state is often critical in diagnosing faults in a system. This requires that the state be captured and archived for future access, creating a historical archive. In WLDF, the archive meets this need with several persistence components. Both events and harvested metrics can be persisted and made available for historical review.

# Diagnostic Archives

- Collected MBean metrics and events are recorded in the server's diagnostic archives:
  - WLDF file store (<server>/data/store/diagnostics, by default)
  - WLDF JDBC store
- Recorded data can be limited using size- or age-based retirement policies.



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Archive component of the WebLogic Diagnostic Framework (WLDF) captures and persists all data events, log records, and metrics collected by WLDF from server instances and applications running on them. You can access archived diagnostic data in online mode (that is, on a running server). You can also access archived data in offline mode using the WebLogic Scripting Tool (WLST). You configure the diagnostic archive on a per-server basis.

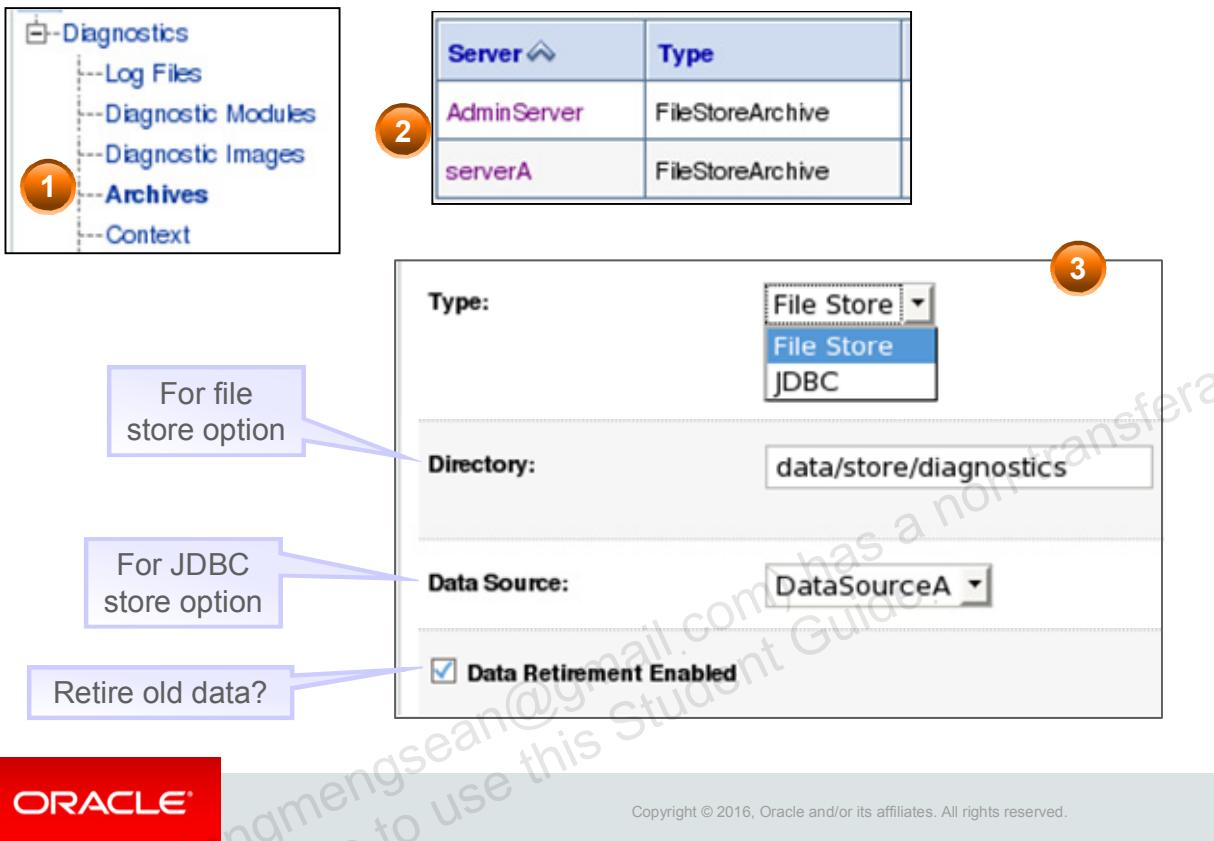
For a file-based store, WLDF creates a file to contain the archived information. The only configuration option for a WLDF file-based archive is the directory where the file will be created and maintained. The default directory is

<domain>/servers/<server>/data/store/diagnostics. When you save to a file-based store, WLDF uses the WebLogic Server persistent store subsystem.

To use a JDBC store, the appropriate tables must exist in a database, and JDBC must be configured to connect to that database. The `wls_events` table stores data generated from WLDF Instrumentation events. The `wls_hvst` table stores data generated from the WLDF Harvester component. Refer to the WebLogic *Configuring Diagnostic Archives* documentation for the required schema.

WLDF includes a configuration-based, data-retirement feature for periodically deleting old diagnostic data from the archives. You can configure size-based data retirement at the server level and age-based retirement at the individual archive level.

# Configuring Server Diagnostic Archives



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

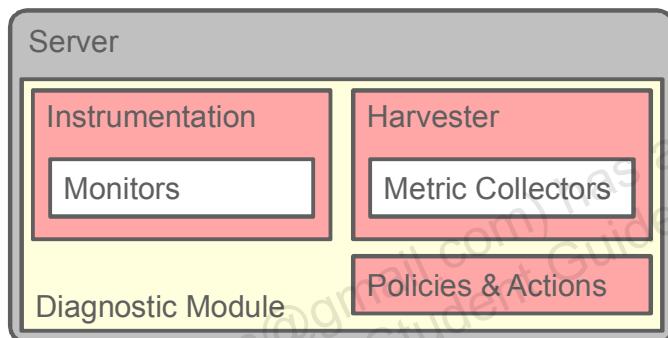
Log files are archived as human-readable files. Events and harvested data are archived in binary format, in a WebLogic persistent store or in a database.

1. In the left pane, expand Diagnostics and select Archives.
2. Click the name of the server for which you want to configure diagnostic archive settings.
3. Select one of the following archive types from the Type list:
  - Select File Store to persist data to the file system. If you choose this option, enter the directory in the Directory field.
  - Select JDBC to persist data to a database. If you choose this option, select the JDBC data source from the Data Source list. You must first configure JDBC connectivity to use this option.
4. Select or deselect Data Retirement Enabled to enable or disable data retirement for this server instance, respectively. In the Preferred Store Size field, enter a maximum data file size, in megabytes. When this size is exceeded, enough of the oldest records in the store are removed to reduce the size of the store below the maximum. In the Store Size Check Period field, enter the interval, in minutes, between the times when the store will be checked to see if it has exceeded the preferred store size.

# Diagnostic Modules

A diagnostic module is used to contain the configuration of WLDF components and is used to target that configuration to servers and clusters.

- Built-in modules
- User-created system modules



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To configure and use the Instrumentation, Harvester, and Watch and Notification components at the server level, you must first create a system resource called a “diagnostic system module.” System modules are globally available for targeting to servers and clusters configured in a domain. In WebLogic 12c, you can target multiple diagnostic system modules to any given server or cluster.

There are two types of diagnostic modules:

- **Built-in:** These are diagnostic modules that are part of the WebLogic product:
  - Production mode domains have a built-in module enabled by default.
  - Development mode domains have all built-in modules disabled by default.
  - Modules provide for low overhead historical server performance metrics.

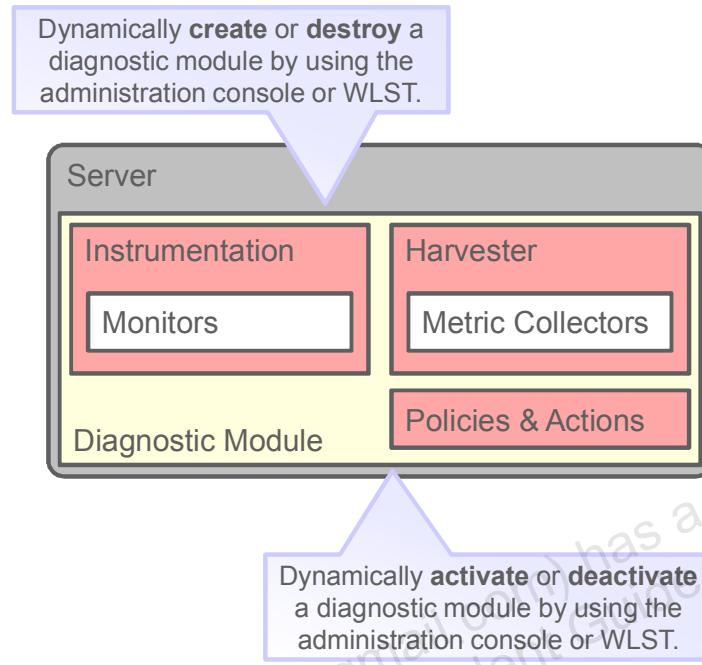
- There are three built-in modules:
  - **Low:** Captures the most important data from key WebLogic MBean attributes (default)
  - **Medium:** Captures additional attributes captured by Low and data from other MBeans
  - **High:** Captures more verbose data than Medium and from a larger number of MBeans
- Built-in modules can be cloned to customize a new module based on a built-in module.
- **User created:** These are diagnostic modules that are created and deployed to WebLogic by using the administration console or WLST. The configuration contained within is configured by an administrator.

WLDF provides features for generating, gathering, analyzing, and persisting diagnostic data from WebLogic Server instances and from applications deployed to them. For server-scoped diagnostics, some WLDF features are configured as part of the configuration for a server in a domain. Other features are configured as system resource descriptors that can be targeted to servers (or clusters). For application-scoped diagnostics, diagnostic features are configured as resource descriptors for the application.

The harvester, watch, and instrumentation features are configured, packaged, and targeted as part of a diagnostic module, similar to a JMS module resource. Only instrumentation monitors can be defined in application-scoped modules, which are placed in the application's `weblogic-diagnostics.xml` file.

You create a diagnostic system module through the administration console or WLST. It is created as a `WLDFResourceBean`, and the configuration is persisted in a resource descriptor file called `<module>.xml`, where `<module>` is the name of the diagnostic module. The file is created by default in the domain's `config/diagnostics` directory with a `.xml` extension.

# Dynamic Diagnostic Modules



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a server hosting a dynamic diagnostic module shuts down, the module reverts to whatever is contained in the configuration when the server is restarted.

# Resource Descriptors

A diagnostic system module has a corresponding resource descriptor:

```
<wldf-resource
    xmlns="http://xmlns.oracle.com/weblogic/weblogic-diagnostics"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-
        diagnostics/1.0/weblogic-diagnostics.xsd">
    <name>MyDiagnosticModule</name>
    <instrumentation>
        <!-- Configuration for harvesting metrics -->
    </instrumentation>
    <harvester>
        <!-- Configuration for harvesting metrics -->
    </harvester>
    <watch-notification>
        <!-- Configuration for watches and notifications -->
    </watch-notification>
</wldf-resource>
```

MyDiagnosticModule.xml

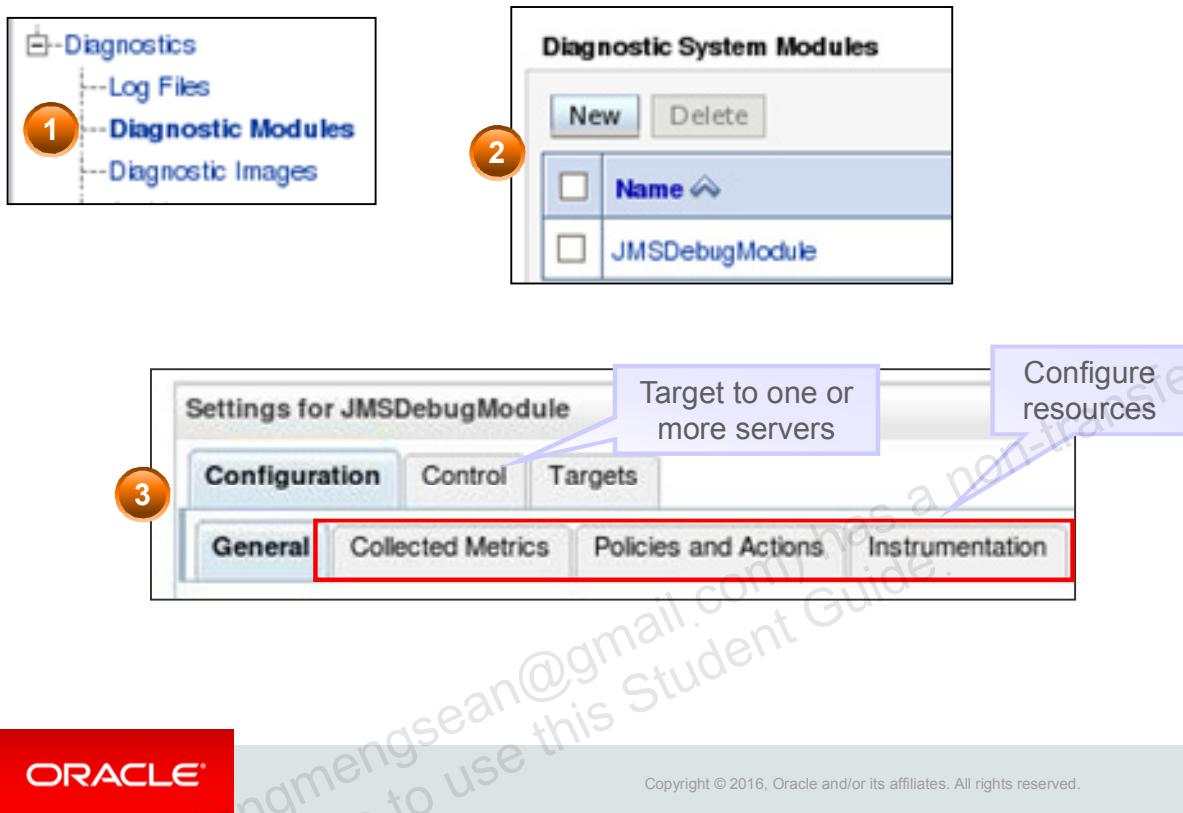


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A resource descriptor contains the actual configuration of WLDF components and features within an XML file. There are two types of resource descriptors:

- **Configured:**
  - Is part of the domain configuration stored in \$DOMAIN\_HOME/config/diagnostics
  - Is referenced by the config.xml file
  - Can be deployed, activated, and deactivated dynamically
- **External:**
  - Is external to the domain configuration
  - Is not referenced by the config.xml file
  - Is only deployed, activated, and deactivated dynamically, and can be administered only by WLST
  - Resides in memory of the server until the server is shut down
  - Never gets persisted to the domain configuration

# Creating a Diagnostic Module



Perform the following steps to create a diagnostic module:

1. In the left pane, expand Diagnostics and select Diagnostic Modules.
2. Click New. Enter a name for the module and, optionally, enter a description. Then click OK.
3. Use the various Configuration tabs to add diagnostic components to this module.
4. To target the module to a server or cluster, click the Targets tab.

## WLST: Example

```
serverConfig> listSystemResourceControls()
External      Enabled          Name
-----        -----
false         false           MyModule1
false         false           MyModule2

serverConfig> createSystemResourceControl('external-wldf', 'external-wldf.xml')
serverConfig> listSystemResourceControls()
External      Enabled          Name
-----        -----
false         false           MyModule1
false         false           MyModule2
true          false           external-wldf

serverConfig> enableSystemResource('external-wldf')
serverConfig> listSystemResourceControls()
External      Enabled          Name
-----        -----
false         false           MyModule1
false         false           MyModule2
true          true            external-wldf
```

Modules defined as part of the domain

Modules defined outside the domain in the external-wldf.xml resource descriptor

DiagnosticModuleExamples.py



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## WLST Commands for WLDF

Command	Description
listSystemResourceControls	Lists all available diagnostic system modules
enableSystemResource	Activates a diagnostic system module
disableSystemResource	Deactivates a diagnostic system module
createSystemResourceControl	Creates a diagnostics system module dynamically by using a specified descriptor file
destroySystemResourceControl	Destroys a previous dynamically created system module
dumpDiagnosticData	Dumps the diagnostics data from a harvester to a local file



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Quiz



The WebLogic Diagnostic Framework (WLDF) enables administrators to analyze archived diagnostic data only in online mode. If the WebLogic cluster is down, no diagnostic data is available.

- a. True
- b. False

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

Archived data can also be accessed in offline mode by using WLST.

# Agenda

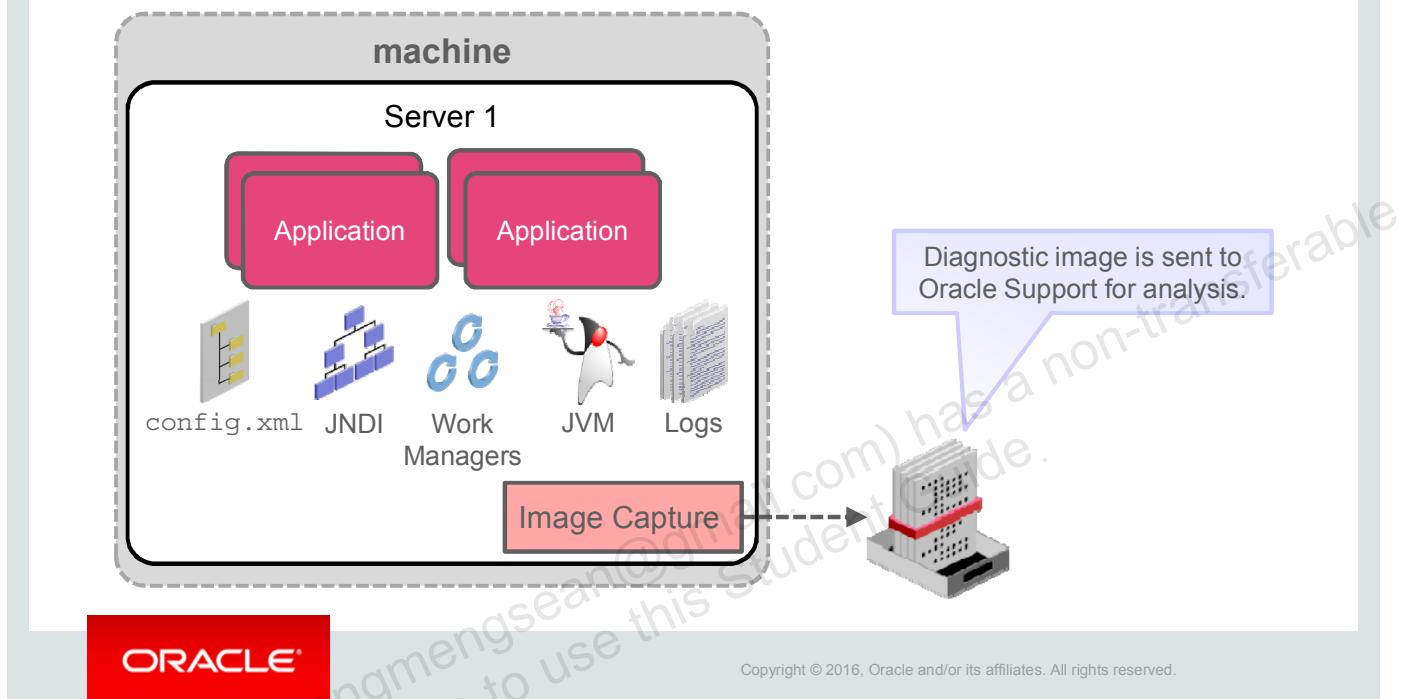
- General Architecture
- Diagnostic Images
  - What Is a Diagnostic Image?
  - Capturing a Diagnostic Image
- Harvesters
- Policies and Actions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## What Is a Diagnostic Image?

A diagnostic image is a post-failure snapshot of the state of a running WebLogic Server instance:



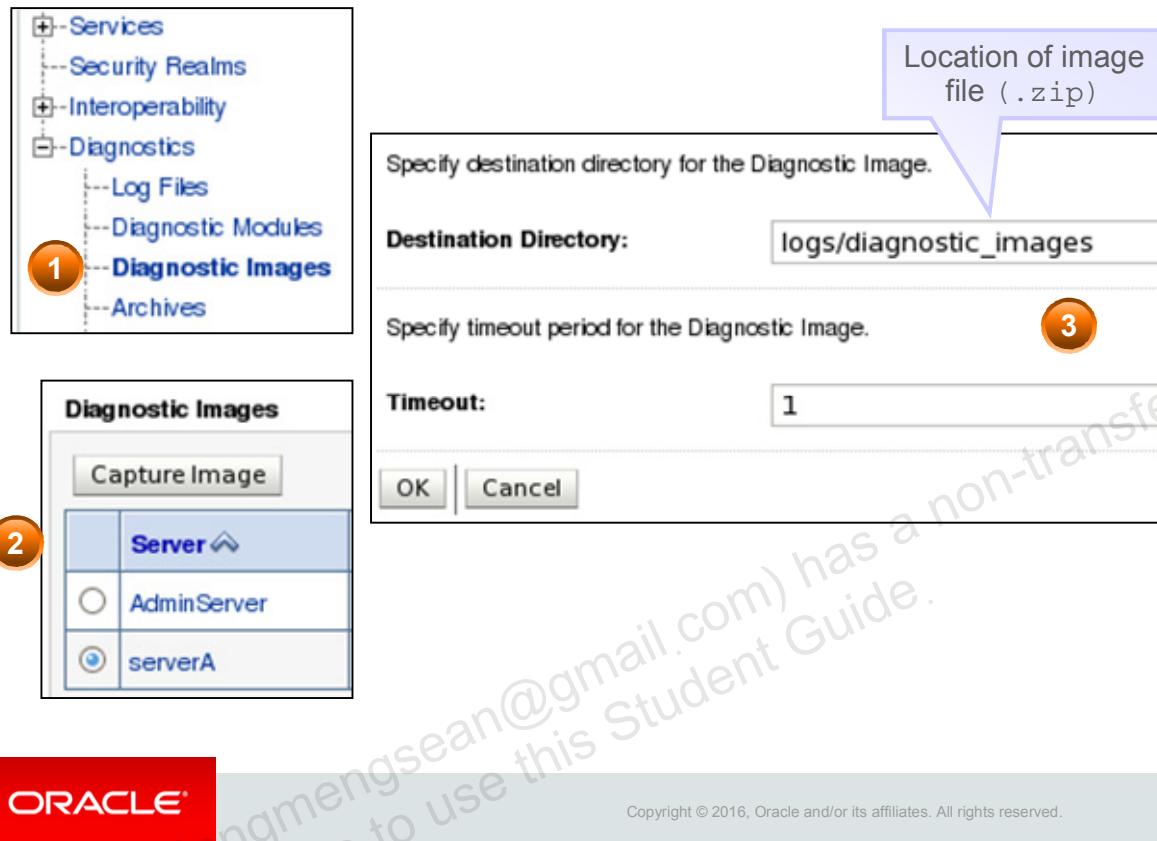
Diagnostic image capture support gathers the most common sources of the key server state used in diagnosing problems. It packages that state into a single artifact, which can be made available to support technicians.

The past state is often critical in diagnosing faults in a system. You use the diagnostic image capture component of WLDF to create a diagnostic snapshot, or dump, of a server's internal runtime state at the time of the capture. This information helps support personnel analyze the cause of a server failure. You can capture an image manually by using the console or WLST, or generate one automatically as part of a watch notification.

Because the diagnostic image capture is meant primarily as a post-failure analysis tool, there is little control over what information is captured. It includes the server's configuration, log cache, JVM state, work manager state, JNDI tree, and most recently harvested data. The image capture subsystem combines the data files produced by the different server subsystems into a single zip file.

If Java's Flight Recorder feature is enabled on your JVM, the diagnostic image will also contain the recording file, which you can view with Java Mission Control (JMC). If enabled, WebLogic can record its own events to this JVM file, which you can also browse with JMC.

# Capturing a Server Diagnostic Image



Perform the following steps to capture a server image by using the console:

1. In the left pane, expand Diagnostics and select Diagnostic Images.
2. Select the server for which you want to capture a diagnostic image, and click Capture Image.
3. Enter a new directory in the Destination Directory field, or accept the default directory. If you change the directory for this image capture, it does not change the default directory for capturing images for this server when they are captured by other means, such as a watch notification. Then click OK.

## WLST: Example

Capture a server diagnostic image:

```
serverRuntime()  
wldfCapture = getMBean('WLDFRuntime/WLDFRuntime/  
WLDFImageRuntime/Image')  
wldfCapture.captureImage('logs/diagnostic_images',30)
```

CreateImage.py



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Additional WLDF WLST examples can be found in the WLDF guide in the product documentation.

Refer to the following MBeans for more information about MBeans related to the diagnostic framework:

- WLDFResourceBean
- WLDFHarvesterBean (**a subclass of WLDFResourceBean**)
- WLDFHarvestedTypeBean (**a component of WLDFHarvesterBean**)
- WLDFInstrumentationBean (**a subclass of WLDFResourceBean**)
- WLDFInstrumentationMonitorBean (**a component of WLDFInstrumentationBean**)
- WLDFWatchNotificationBean (**a subclass of WLDFResourceBean**)
- WLDFWatchBean (**a component of WLDFWatchNotificationBean**)
- WLDFNotificationBean (**abstract; subclasses exist for each notification type**)
- WLDFRuntimeMBean
- WLDFImageRuntimeMBean (**a component of WLDFRuntimeMBean**)
- WLDFControlRuntimeMBean
- WLDFSystemResourceControlRuntimeMBean

## Quiz

Q

The MBean method used to obtain a post-failure snapshot of the state of a running WebLogic Server from a WLST script is:

- a. `getSnapshot()`
- b. `capturePic()`
- c. `captureImage()`
- d. `dumpState()`

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

# Agenda

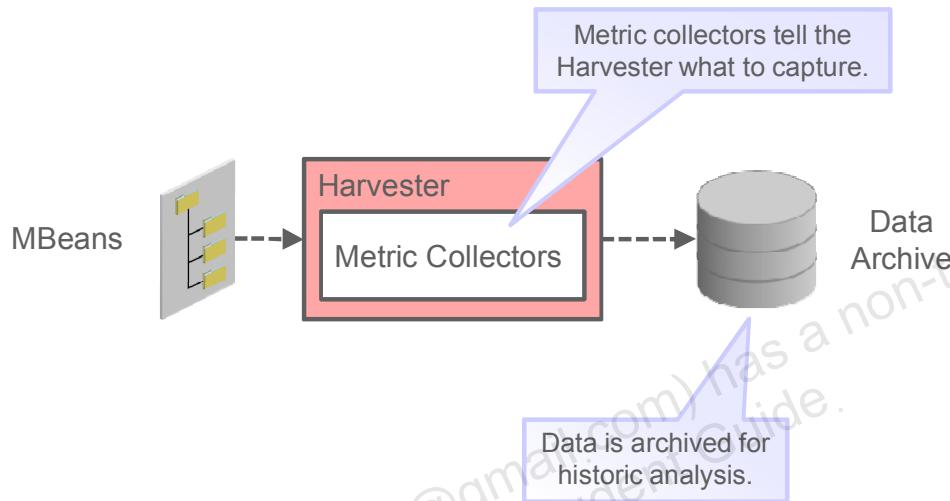
- General Architecture
- Diagnostic Images
- Harvesters
  - What Is a Harvester?
  - Configuring Metric Collection
- Policies and Actions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## What Is a Harvester?

The Harvester is a WLDF component that you can configure to capture WebLogic runtime MBean data.

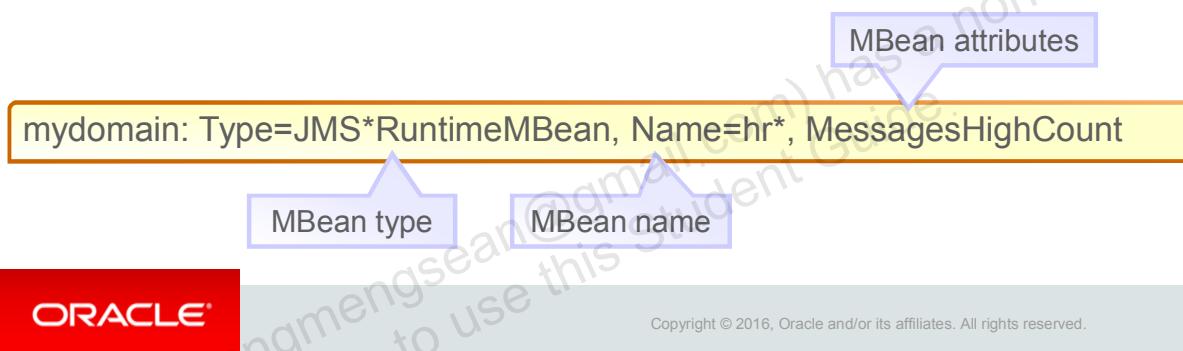


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Metric Collectors

- A metric collector definition for the Harvester includes:
  - The runtime MBean type to query
  - The specific MBean instance names to query (all instances, by default)
  - The MBean attributes to collect (all attributes, by default)
  - How often to sample data
- Alternatively, use the MBean expression syntax, which also supports wildcards and complex attributes.



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

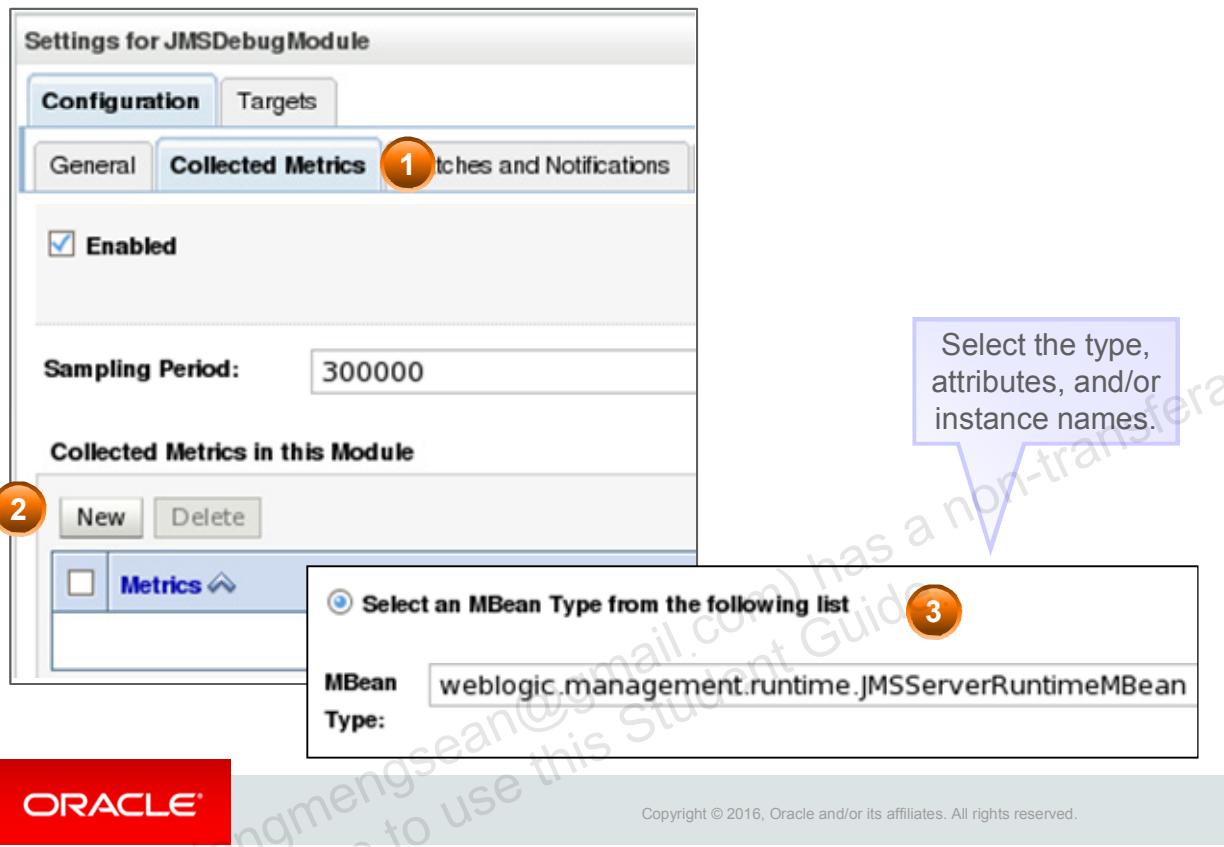
Harvesting metrics is the process of gathering data that is useful for monitoring the system state and performance. Metrics are exposed to WLDF as attributes on qualified MBeans. The Harvester gathers values from selected MBean attributes at a specified sampling rate. Therefore, you can track potentially fluctuating values over time. For custom MBeans, the MBean must be currently registered with the JMX server.

You can configure the Harvester to harvest data from named MBean types, instances, and attributes. If only a type is specified, data is collected from all attributes in all instances of the specified type. If only a type and attributes are specified, data is collected from all instances of the specified type.

The sample period specifies the time between each cycle. For example, if the Harvester begins execution at time T, and the sample period is I, the next harvest cycle begins at T+I. If a cycle takes A seconds to complete and if A exceeds I, then the next cycle begins at T+A. If this occurs, the Harvester tries to start the next cycle sooner, to ensure that the average interval is I.

WLDF allows for the use of wildcards (\*) in type names, instance names, and attribute specifications. WLDF also supports nested attributes using a dot delimiter, as well as complex attributes such as arrays and maps. WLDF watch expressions also support similar capabilities.

# Configuring a Metric Collector



Metrics are configured and collected in the scope of a diagnostic system module targeted to one or more server instances. Therefore, to collect metrics, you must first create a diagnostic system module.

1. Click the name of the module for which you want to configure metric collection. Then click Configuration > Collected Metrics.
2. To enable or disable all metric collection for this module, select or deselect the Enabled check box. To set the period between samples, enter the period (in milliseconds) in the Sampling Period field. To define a new collected metric, click New.
3. From the MBean Server Location drop-down list, select either DomainRuntime or ServerRuntime. Then click Next. Select an MBean that you want to monitor from the MBean Type list. Then click Next again.
4. In the Collected Attributes section, select one or more attributes from the Available list and move them to the Chosen list (default is all attributes). Click Next.
5. In the Collected Instances section, select one or more instances from the Available list and move them to the Chosen list (default is all instances). Click Finish.
6. Click Save.

## WLST: Example

Create a diagnostic module and metric collector (harvester):

```
module = cmo.createWLDFSystemResource('JMSDebugModule')
module.addTarget(getMBean('/Servers/serverA'))
harvester = getMBean('/WLDFSystemResources/JMSDebugModule/
    WLDFResource/JMSDebugModule/Harvester/JMSDebugModule')
harvester.setSamplePeriod(300000)
harvester.setEnabled(true)
harvestType = harvester.createHarvestedType
    ('weblogic.management.runtime.JMSRuntimeMBean')
harvestType.setEnabled(true)
harvestType.setHarvestedAttributes(['MessagesHighCount'])
```

• CreateWLDF.py



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Additional WLDF WLST examples can be found in the WLDF guide in the product documentation.

Refer to the following MBeans for more information about MBeans related to the diagnostic framework:

- WLDFResourceBean
- WLDFHarvesterBean (**a subclass of WLDFResourceBean**)
- WLDFHarvestedTypeBean (**a component of WLDFHarvesterBean**)
- WLDFInstrumentationBean (**a subclass of WLDFResourceBean**)
- WLDFInstrumentationMonitorBean (**a component of WLDFInstrumentationBean**)
- WLDFWatchNotificationBean (**a subclass of WLDFResourceBean**)
- WLDFWatchBean (**a component of WLDFWatchNotificationBean**)
- WLDFNotificationBean (**abstract; subclasses exist for each notification type**)
- WLDFRuntimeMBean
- WLDFImageRuntimeMBean (**a component of WLDFRuntimeMBean**)

## Quiz



The Harvester is a WLDF component that:

- a. Checkpoints performance metrics, received from the Collector, either to disk or database
- b. Captures WebLogic runtime MBean data
- c. Gathers statistics from the Oracle database
- d. Uses a two-phase commit protocol to synchronize the collection of performance metrics across the enterprise

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

# Agenda

- General Architecture
- Diagnostic Images
- Harvesters
- Policies and Actions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

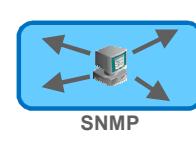
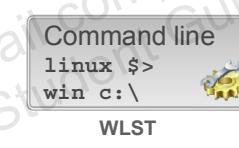
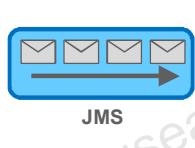
# Policies and Actions

## Policies:

- Identify situations used to capture data
- Are used to analyze log records, data events, and harvested metrics
- Include expressions, alarms and **action** handlers

## Actions:

- Are executed when an associated policy expression evaluates to true
- Include support for Scaling dynamic clusters, interacting with JMX, JMS, SMTP, SNMP, REST and so on



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A policy identifies a situation that used to trap for monitoring or diagnostic purposes. You can configure policies to analyze log records, data events, and harvested metrics.

A policy includes:

- A policy expression (with the exception of calendar-based policies)
- An alarm setting
- One or more action handlers

You can configure policies to enable elasticity in dynamic clusters; for example, to automatically scale a dynamic cluster up or down by a specific number of server instances.

Policies support two categories of elasticity:

- Calendar-based scaling: Scaling operations on a dynamic cluster that are executed on a particular date and time
- Policy-based scaling: Scaling operations on a dynamic cluster that are executed in response to changes in demand

An action is an operation that is executed when a policy expression evaluates to true. WLDF supports the following types of actions:

- Scaling a dynamic cluster
- Java Management Extensions (JMX)
- Java Message Service (JMS)
- Simple Mail Transfer Protocol (SMTP), for example, email
- Simple Network Management Protocol (SNMP)
- Diagnostic Images
- REST

For policies to be useful, they must be associated with actions.

Policies and actions are configured separately from each other. An action can be associated with multiple policies, and a policy can be associated with multiple actions. This provides the flexibility to recombine and reuse policies and actions, according to current needs.

# Policy Types

Policy Type	Description
Smart Rules	Use a set of prepackaged functions and configurable parameters that allow for the creation of complex policy expressions combinations of functions and parameters
Calendar Based	Are scheduled at a specific time, after a duration of time, or at timed intervals
Collected Metrics	Use WLDF-provided beans and functions within policy expressions based on a variety of MBeans including WebLogic Server Runtime, Domain Runtime, and JVM platform Mbeans.
Server Log	Use various server variables to define policy expressions. Fields include SERVER, MACHINE, SUBSYSTEM and so on
Domain Log	Use various domain variables to define policy expressions
Event Data	Use various event data generated by instrumentation and stored in the event archive



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Policies are based on one of the 6 policy types which include:

- Smart Rules
- Calendar Based: Policy based on calendar or timed events such as a specific date and time, a certain duration of time, or at specific intervals
- Collected Metrics: Policy based on WLDF-provided beans and functions and policy expressions. Based on JavaBean objects that can obtain access to common WebLogic Server JMX data sources, such as WebLogic Server Runtime, Domain Runtime MBean Server, and JVM platform Mbeans.
- Server log: Policies based on a number of available log fields including SERVER, MACHINE, SUBSYSTEM, THREAD, SEVERITY, USER ID and so on. See the WLDF Appendix C for a complete list.
- Domain Log: Policies based on a domain log fields. See the WLDF appendix C for a complete list.
- Event Data: Policies based on event data, generated by instrumentation, and stored in the event archives

# Configuring Policies

1 In the administration console, select the **Domain Structure** pane.

2 Under the domain, select **Diagnostics > Diagnostic Modules**.



3 Select a Module name.

Diagnostic System Modules	
<input type="button" value="New"/>	<input type="button" value="Delete"/>
Name	Description
<input type="checkbox"/>	MyCustomModule
<input type="button" value="New"/>	<input type="button" value="Delete"/>

4 Select Configuration > Policies and Actions.



5 In the Policy subtab, click New.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To create a policy for a diagnostic system module:

1. Log in to the WebLogic console and in the Domain Structure pane.
2. Expand Diagnostics > Diagnostic Modules.
3. On the Summary of Diagnostic Modules page, click the name of the module to house the new policy.
4. On the Settings for [Module Name] page, select Configuration > Policies and Actions > Policies.
5. Select the Policies tab and click New.

# Configuring Policies

6

Name the policy.

**Create Policy**

Back | Next | Finish | Cancel

**Create a Diagnostic Policy**

The following properties will be used to identify your policy.

\* Indicates required fields

What would you like to name your new policy?

\* Policy Name: ExamplePolicy

What policy type would you like to select?

Policy Type: Smart Rule

Is this policy enabled?  Enable Policy

7

Select a policy rule type.

8

Configure the policy based on type.

**Select Smart Rule**

Use this page to select a smart rule from the list.

**Smart Rules**

Function Name	Group
Cluster Low Average Throughput	Cluster
High Average Idle Threads	Server

Example  
Smart Rule

**Create Policy**

Back | Next | Finish | Cancel

**Configure Policy Expressions**

Add expressions to create the rule for your policy.

Policy:

```
(SEVERITY = 'Warning')
```

Example  
Server Log

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

6. On the Create Policy page, do the following:
  - a. Enter a name for the policy in the Policy Name field.
  - b. Select a policy type from the Policy Type list:
    - Select Calendar Based to set a policy based only on a schedule.
    - Select Smart Rule to set a policy based on a single built-in smart rule, evaluated on a schedule.
    - Select Collected Metrics to set a custom Java EL policy expression based on metrics exposed through JMX, evaluated on a schedule.
    - Select Server Log to set a policy based on data written to server logs.
    - Select Domain Log to set a policy based on data written to the domain log.
    - Select Event Data to set a policy based on data from an instrumentation event generated by WLDF instrumentation monitors.
  - c. To enable or disable the policy, select or deselect the Enable Policy check box.
7. Click Next.
8. The pages that are displayed next vary based on the policy type you selected above.
9. Click Finish when you are done creating your new policy.
10. In the Change Center, click Activate Changes.

# Configuring Actions

**1** In the administration console, select the **Domain Structure** pane.

**2** Under the domain, select **Diagnostics > Diagnostic Modules**.

**3** Select a Module name.

**4** Select **Configuration > Policies and Actions**.

**5** In the **Actions** subtab, click **New**.

**6** Select a Type and click **Next**.

**7** Enter Name ... click **Next**.

The screenshot shows the 'Create Action' dialog box. At the top, it says 'Create Action' and 'The following properties will be used to identify your new action.' Below that, it asks 'What action type would you like to select?'. A dropdown menu labeled 'Type:' contains the following options: SMTP (E-Mail), JMS Message, Diagnostic Image, JMX Notification, SNMP Trap, Scale Up, Scale Down, Script, and REST. The 'SMTP (E-Mail)' option is selected. To the right of the dropdown, there is a form with fields for 'Action Name' (containing a placeholder 'What would you like to name your new action?'), 'Timeout (in seconds)' (set to 0), and 'Is this action enabled?' (with a checked checkbox labeled 'Enable Action').

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

An action is an operation that is executed when a policy expression evaluates to true. WLDF supports the following types of diagnostic actions, based on the delivery mechanism:

To create an action:

1. Log in to the WebLogic console and in the Domain Structure pane, click Lock & Edit.
2. Expand Diagnostics > Diagnostic Modules.
3. On the Summary of Diagnostic Modules page, click the name of the module to house the new policy.
4. On the Settings for [Module Name] page, select Configuration > Policies and Actions > Policies.
5. Select the Actions tab and click New.

6. On the Create Action page, select an action type from the Type list and click **Next**:
  - a. Select SMTP (E-Mail) to create an action that will deliver a notification through email.
  - b. Select JMS Message to create an action that will deliver a notification through JMS.
  - c. Select Diagnostic Image to capture a diagnostic image.
  - d. Select JMX Action to create an action that will deliver a notification through JMX.
  - e. Select SNMP Trap to create an SNMP trap.
  - f. Select Scale Up to create an action that will expand a dynamic cluster by a specified number of dynamic server instances.
  - g. Select Scale Down to create an action that will reduce a dynamic cluster by a specified number of dynamic server instances.
  - h. Select Script to create a script execution action.
  - i. Select REST to create an action that will deliver a notification through REST.

**Note:** Only one scaling action can be assigned to a given policy.

7. Enter a name for the action in the Action Name field.

Enter the timeout period in the Timeout (in seconds) field. The timeout period is the length of time, in seconds, that an action has to complete execution before WebLogic Server attempts to cancel it. By default, the timeout period is 0, which disables the action timeout.

To enable or disable the action, select or deselect the Enable Action check box, as appropriate.

Click Next. The page that is displayed next is different for each of the different action types, listed above.

# Configuring Actions

8

Complete the action.

The image displays two side-by-side configuration screens for creating actions in a system. Both screens have a header titled 'Create Action' with buttons for Back, Next, Finish, and Cancel.

**Left Screen (Action Configuration - SMTP Properties):**

- Mail Session Name:** A dropdown menu showing '(None)' with a 'Create a New...' button.
- E-Mail Recipients:** A text input field.
- E-Mail Subject:** A text input field.
- Options:** Radio buttons for 'Use Default' (selected) and 'Customize'.

**Right Screen (Action Configuration - Script Action Properties):**

- Working Directory:** A text input field.
- Path to Script:** A text input field.
- Environment Variables:** A large text input area.

Both screens include a red 'ORACLE' logo at the bottom.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

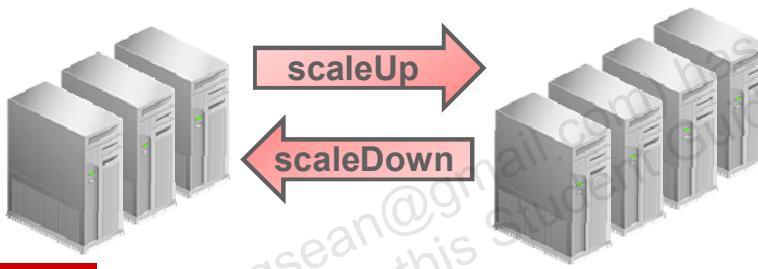
8. Click Finish when you are done creating your new action.
9. In the Change Center, click Activate Changes.

# Elastic Actions and Scaling

## Elastic actions:

- Are used to increase or decrease the size of a dynamic cluster
- Are triggered when policy guidelines are met
- Are associated 1:1 with policy
- Are configured within domain-scope diagnostic system modules
- Are configured with scale-up/down-actions inside wldf-resources

Only one elastic scaling action per policy.  
There may be other actions.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Elastic actions are actions used to scale up or down a dynamic cluster when certain policy conditions are met, such as load or demand. The scale up and scale down actions are used to add or remove running dynamic server instances from a dynamic cluster during scaling operations. Elastic actions can be associated with policies. As a policy's conditions are met, the elastic action is triggered and the scaling operation begins. This type of scaling is called policy-based scaling. Only one elastic action can be assigned to a policy. However, any number of non-scaling actions can be assigned to a policy, and elastic actions can be used in conjunction with those non-scaling actions.

You can configure a scale up action in WLST using code similar to:

```
startEdit()  
scaleUp=wn.createScaleUpAction('scaleUp')  
scaleUp.setScalingSize(1)  
scaleUp.setClusterName('DynamicCluster')  
save()  
activate()
```

## Scale up Action

Scale up actions add dynamic server instances to the specified dynamic cluster.

```
<wldf-resource>
    <name>ClusterManager</name>
    <watch-notification>
        <!-- One or more policy configurations -->
        <scale-up-action>
            <name>scaleUp</name>
            <cluster-name>DynamicCluster</cluster-name>
            <scaling-size>1</scaling-size>
        </scale-up-action>
        <!-- Other action configurations -->
    </watch-notification>
</wldf-resource>
```

MyDiagnosticModule.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A resource descriptor contains the actual configuration of WLDF components and features within an XML file. There are two types of resource descriptors:

- **Configured:**
  - Is part of the domain configuration stored in \$DOMAIN\_HOME/config/diagnostics
  - Is referenced by the config.xml file
  - Can be deployed, activated, and deactivated dynamically
- **External:**
  - Is external to the domain configuration
  - Is not referenced by the config.xml file
  - Is only deployed, activated, and deactivated dynamically, and can be administered only by WLST
  - Resides in memory of the server until the server is shut down
  - Never gets persisted to the domain configuration

## Scale down action

Scale down shuts down and removes dynamic server instances in the specified dynamic cluster.

```
<wldf-resource>
  <name>ClusterManager</name>
  <watch-notification>
    <!-- One or more policy configurations -->
    <scale-down-action>
      <name>scaleDown</name>
      <cluster-name>DynamicCluster</cluster-name>
      <scaling-size>1</scaling-size>
    </scale-down-action>
    <!-- Other action configurations -->
  </watch-notification>
</wldf-resource>
```

MyDiagnosticModule.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows a scale down XML example. The `scale-down-action` is specified within a `watch-notifications` element which itself is within a `wldf-resource` element. The scale down element, much like scale up, includes the name of the cluster as well as the scaling-size.

Similar code, but performed using WLST might resemble:

```
startEdit()
scaleDown=wn.createScaleDownAction('scaleDown')
scaleDown.setScalingSize(1)
scaleDown.setClusterName('DynamicCluster')
save()
activate()
```

## Quiz



A policy identifies:

- a. A specific set of policies types used to generate alerts
- b. A situation that one wants to trap for monitoring or diagnostic purposes
- c. A mechanism used to capture data based on logs, data, or harvested metrics used in conjunction with actions
- d. A notification mechanism used to alert administrators in case of suboptimal performance
- e. A set of database tables or tables containing performance metrics

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Summary

In this lesson, you should have learned how to configure the WebLogic Diagnostic Framework (WLDF) to monitor a domain.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 15-1 Overview: Using a Built-in Diagnostic Module

This practice covers the following topics:

- Configuring a built-in diagnostic module
- Changing the settings of a built-in module
- Cloning and customizing a built-in module



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.

# WebLogic and Coherence Integration

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to configure:

- Coherence\*Web
- Managed Coherence Servers



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- Coherence Overview
- Coherence\*Web Session Replication
- Managed Coherence Servers



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle Coherence: Overview

## Coherence:

- Provides a distributed, in-memory caching solution for Java
- Is based on a *grid* of cache servers or *nodes*
- Automatically distributes or partitions cached data across the grid based on the number of servers
- Implements replication for high availability
- Includes a decentralized management and monitoring model based on JMX
- Supports a proxy architecture to provide connectivity beyond the Coherence cluster
- Supports additional languages such as .NET and C++



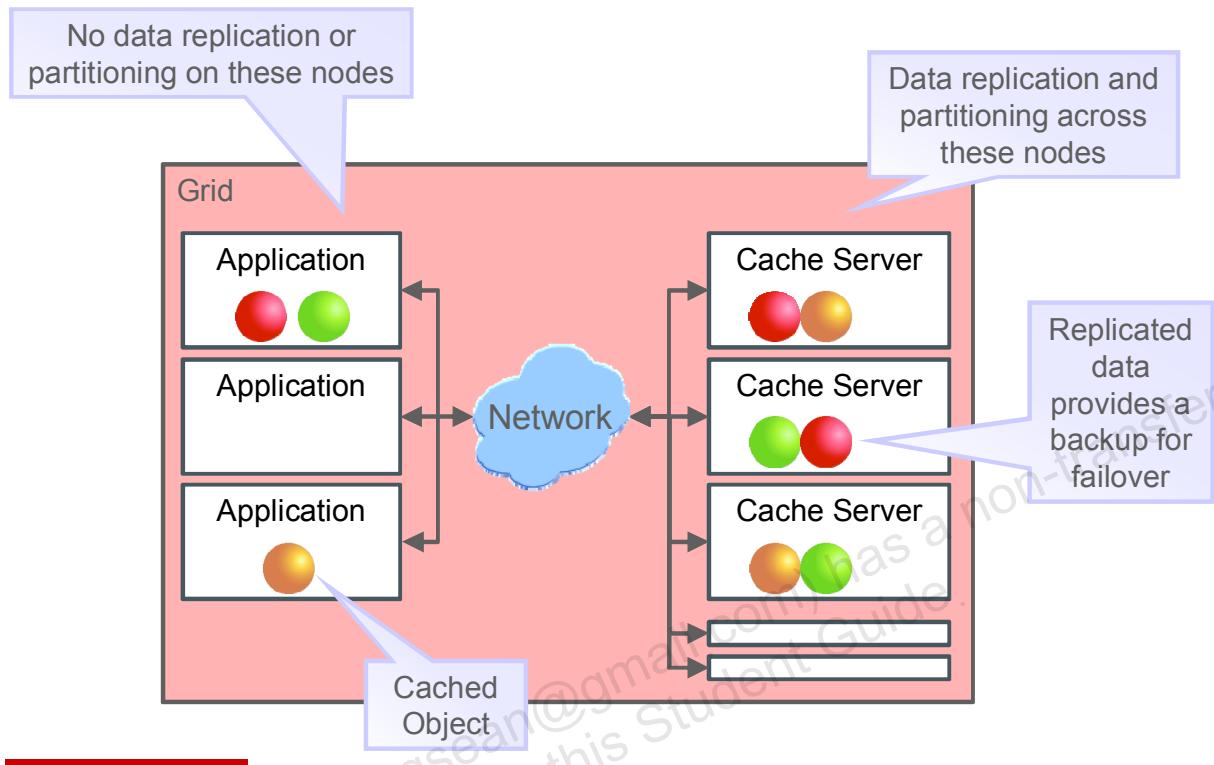
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

One of the primary uses of Oracle Coherence is to cluster an application's objects and data. In the simplest sense, this means that all the objects and data that an application delegates to Coherence are automatically available to and accessible by all servers in the application cluster. None of the objects or data will be lost in the event of server failure. By clustering the application's objects and data, Coherence solves many of the difficult problems related to achieving availability, reliability, scalability, performance, serviceability, and manageability of clustered applications.

Oracle Coherence is a JCache-compliant, in-memory caching and data management solution for clustered Java EE applications and application servers. Coherence makes sharing and managing data in a cluster as simple as on a single server. It accomplishes this by coordinating updates to the cached data by using clusterwide concurrency control, replicating and distributing data modifications across the cluster, and delivering notifications of data modifications to any servers that request them. Developers can easily take advantage of Coherence features by using the standard Java Collections API to access and modify data, and by using the standard JavaBean event model to receive data change notifications.

Coherence provides a clusterwide view of management information through the standard JMX API, so that the entire cluster can be managed from a single server. The information provided includes cache sizes along with hit and miss rates.

# Oracle Coherence: Overview



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Partitioning refers to the ability of Coherence to load-balance data storage, access, and management across all the servers in the cluster. For example, when using Coherence data partitioning, if there are four servers in a cluster, each will manage 25% of the data. And if another server is added, each server will dynamically adjust so that the five servers will manage 20% of the data. This data load balancing will occur without any application interruption and without any lost data or operations. Similarly, if one of those five servers were to fail, each of the remaining four servers would readjust to managing 25% of the data. Once again, there is no data loss, including the 20% of the data that was being managed on the failed server.

While the partitioning feature dynamically load balances data evenly across the entire server cluster, replication ensures that a desired set of data is always available and up-to-date at all times in the cluster. Replication allows operations running on any server to obtain the data that they need locally, at basically no cost, because that data has already been replicated to that server. The only downside of partitioning is that it introduces latency for data access, and in most applications the data access rate far outweighs the data modification rate. To eliminate the latency associated with partitioned data access, Coherence can employ “near caching.” Frequently and recently used data from the partitioned cache are maintained on the specific servers that are accessing that data within a near cache, and this local data is kept up-to-date by using event-based invalidation.

# Agenda

- Coherence Overview
- Coherence\*Web Session Replication
  - Types of Session Persistence
  - Coherence\*Web
  - Coherence\*Web and WebLogic Clusters
  - Coherence\*Web Session Failover
  - Configuring Coherence\*Web in WebLogic
- Managed Coherence Servers



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Types of Session Persistence

WebLogic Server supports several strategies so that the session information is not lost when a server fails:

- In-memory session persistence
- File session persistence
- JDBC (database) session persistence
- Coherence\*Web session persistence



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Java web application components, such as Servlets and JavaServer Pages (JSPs), can maintain data on behalf of clients by storing objects in an `HttpSession`. An `HttpSession` is available on a per-client basis. After an instance of WebLogic Server creates a session for a client, it also writes a cookie to the client's web browser. This cookie indicates which server has this client's session. The cluster proxy checks this cookie on subsequent client requests, and routes the client to the instance of WebLogic Server that has the client's session.

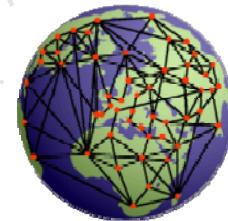
If the server that has the client's session fails, when the client makes their next request, they cannot be routed to that server. The proxy chooses another server. That server does not have the client's session, so information about the client is lost.

To provide transparent failover for web applications, replication or persisted, shared access to the information in each `HttpSession` object must be provided. This is accomplished in WebLogic Server by using in-memory persistence, file system persistence, database persistence, or Coherence\*Web persistence. Each of these options is configured in the web application's WebLogic deployment descriptor, `weblogic.xml`. Each option has its own configurable parameters that are also entered in `weblogic.xml`.

Note that in-memory and JDBC persistence have two options: synchronous and asynchronous. The asynchronous option persists data in batches to improve cluster performance.

## Coherence\*Web

- Is a plug-in that enhances WebLogic Server's session management implementation with a Coherence grid
- Provides a highly scalable session replication solution that can span WebLogic domains and clusters
- Reduces WebLogic Server's memory footprint
- Requires no changes to existing application code
- Allows session data to be shared across applications
- Can be enabled on an application basis



**ORACLE®**

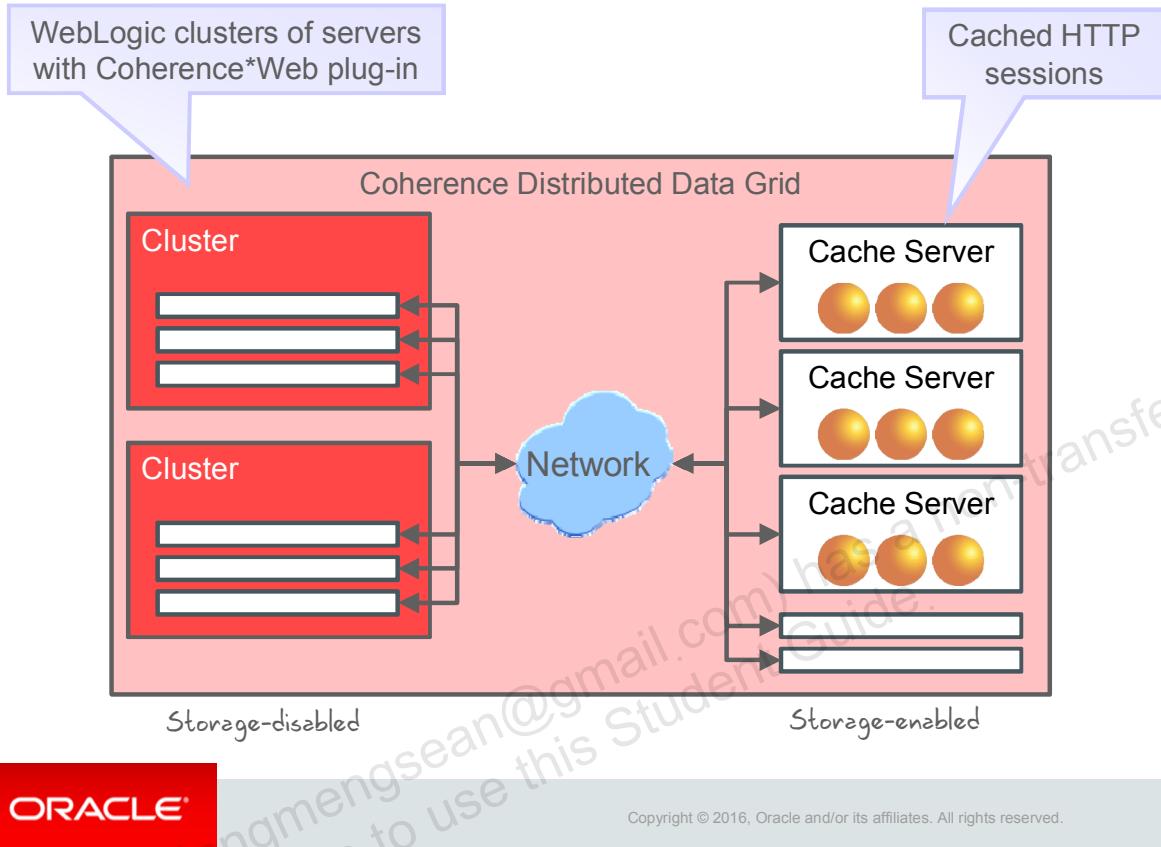
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Coherence\*Web is an application server plug-in dedicated to managing session state in clustered environments. It brings the scalability, availability, reliability, and performance characteristics of a Coherence data grid to in-memory session management and storage. Also, because it is not constrained by the deployment topologies of the application server, it enables session sharing and management across different web applications, domains, and even different application server products. Sometimes you may want to explicitly prevent session data from being shared by different Java EE applications that participate in the same Coherence cluster, so Coherence\*Web supports this approach as well.

If sessions are shared across web applications, you may want to scope individual session attributes so that they are either globally visible (that is, all web applications can see and modify these attributes) or scoped to an individual web application (that is, not visible to any instance of another application). The latter approach may be desired to help avoid namespace collisions when multiple applications use the same session attribute names.

With Coherence\*Web, session data is stored outside of the application server, thereby freeing server heap space. This architecture also allows you to individually tune and scale the sizes of your application server clusters and session data grids.

## Coherence\*Web and WebLogic Clusters

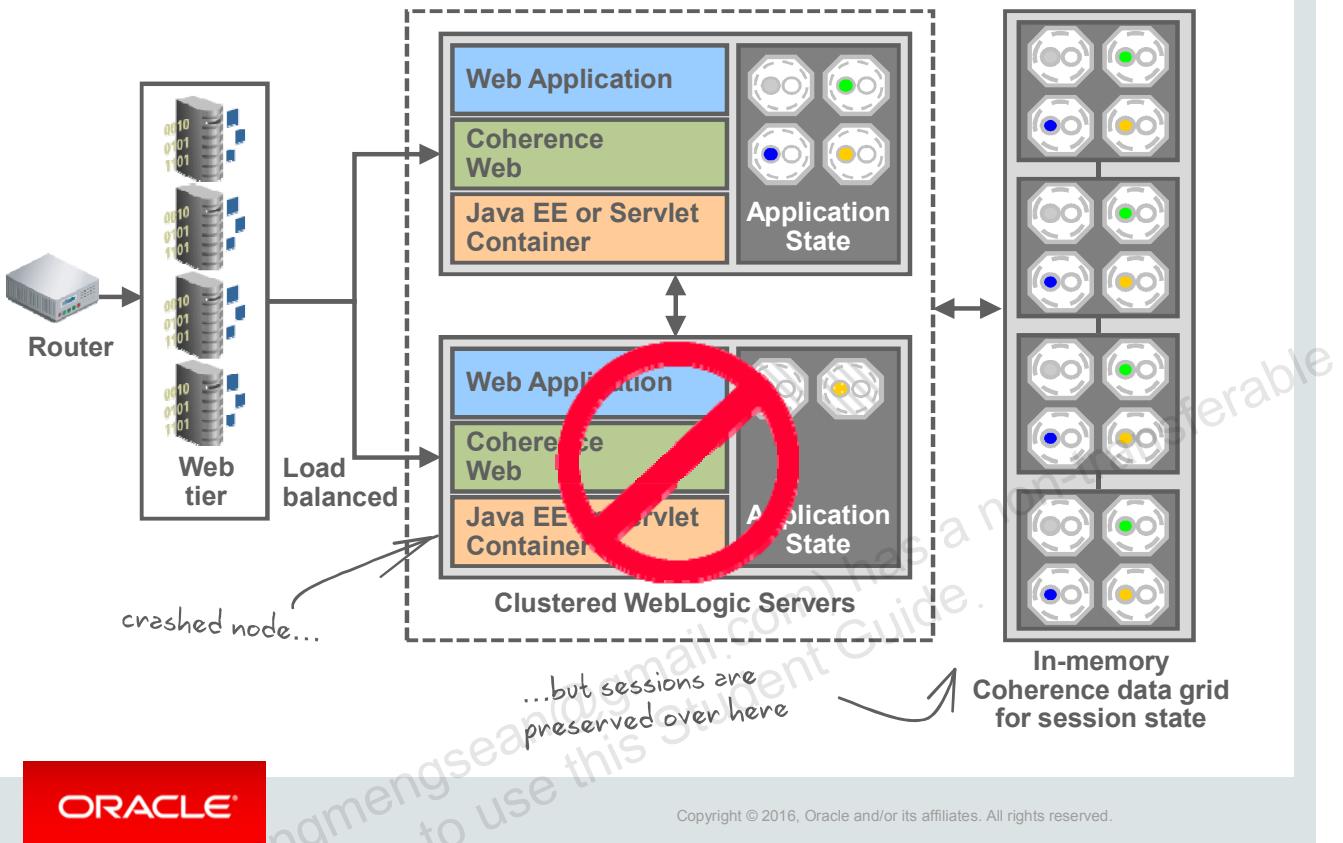


Coherence\*Web is not a replacement for WebLogic Server's in-memory HTTP state replication services. However, Coherence\*Web should be considered when an application has large HTTP session state objects, when running into memory constraints due to storing HTTP session object data, or if you have an existing Coherence cluster and want to offload HTTP Session storage to a Coherence cluster.

Coherence\*Web is configured with local storage disabled. This means that although the WebLogic server instance is a member of the Coherence cluster, it is not a data storage member of the cluster. Storage-disabled members rely on other Coherence cluster members to manage and replicate data.

By default, Coherence\*Web creates a single HTTP session across all web applications for each client and scopes the session attributes to each web application. This means that if a session is invalidated in one web application, that same session is invalidated for all web applications in WebLogic Server that are using Coherence\*Web. If you do not want this behavior, a potential work-around is to edit the `<cookie-path>` element of the `weblogic.xml` descriptor.

## Coherence\*Web Session Failover



The diagram in the slide illustrates that if a server were to fail, if you lose the entire application server or a JVM, Coherence still maintains the data. Therefore, you do not lose any session state information. Using your existing infrastructure, the user session fails over to one of the other application servers, or the one that is surviving. Therefore, no data is lost. The end user may not even notice that there is an outage. The user session continues without the user even noticing that anything went wrong because Coherence manages the data.

One common use case for Coherence clustering is to manage user sessions (conversational state) in the cluster. This capability is provided by the Coherence\*Web module, which is a built-in feature of Oracle Coherence. Coherence\*Web provides linear scalability for HTTP Session Management in clusters of hundreds of production servers. It can achieve this linear scalability because, at its core, it is built on Oracle Coherence dynamic partitioning.

# Configuring Coherence\*Web in WebLogic

1. Start one or more Coherence cache servers on your network.
2. Update each application's `weblogic.xml` descriptor file and set the session persistence type to `coherence-web`.

```
<weblogic-web-app>
  <session-descriptor>
    <persistence-store-type>
      coherence-web
    </persistence-store-type>
  </session-descriptor>
</weblogic-web-app>
```

weblogic.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The WebLogic Server Coherence\*Web plug-in consists of the `coherence-web.jar` and `coherence.jar` files, located in the `coherence/lib` directory in the Coherence distribution. When WebLogic is installed, these files are included in the WebLogic system classpath. The `coherence-web.jar` file will load application classes from the appropriate WebLogic classloader so you do not have to include Coherence JAR files in the web application's classpath. All Coherence\*Web-enabled applications running on WebLogic 12c (12.1.2) have application server scope. As a result, all applications become part of the same Coherence node.

The Coherence caches used by Coherence\*Web are configured by the `default-session-cache-config.xml` file included in the `coherence-web.jar` file. Although the default configuration should be sufficient for most deployments, it can be overridden by placing a `session-cache-config.xml` file in your web application's `WEB-INF/classes` folder. If you are using the default session cache configuration file, you can package and deploy your applications just like any other Java EE application. However, if you are using a custom session cache configuration file, you must package and deploy your application in a GAR file.

**Note:** GAR files are covered later in this lesson.

## Quiz



What is the primary purpose of Coherence\*Web?

- a. To use Coherence caching for application data
- b. To integrate Coherence with application servers
- c. To use Coherence caching for HTTP sessions
- d. To use Coherence caching for client connections

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Practice 16-1 Overview: Configuring Coherence\*Web

This practice covers the following topics:

- Configuring the default Coherence cluster in WebLogic to use multicast networking
- Configuring Coherence stand-alone cache servers to use the same cluster settings as the WebLogic default Coherence cluster
- Starting Coherence cache servers
- Deploying the ShoppingCart application to WebLogic
- Testing Coherence\*Web HTTP sessions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

- Coherence\*Web Session Replication
- Managed Coherence Servers
  - What Are Managed Coherence Servers?
  - What Are Coherence Container Applications?
  - What Are the Benefits of the Coherence Container
  - Installing WebLogic with Coherence Support
  - Coherence Grid Archives
  - Coherence Cluster

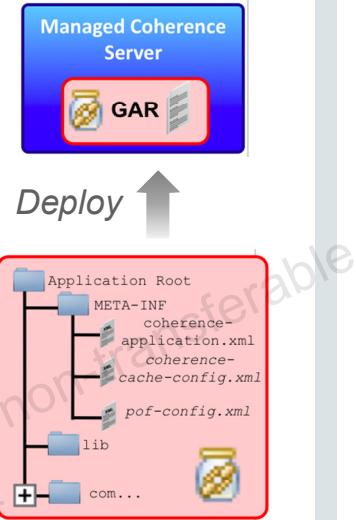


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Coherence and WebLogic Server

Oracle WebLogic 12c provides comprehensive support for Coherence including:

- **Managed Coherence servers:** The ability to configure, start, stop, and otherwise manage Coherence servers natively within a WebLogic environment
- **Coherence Grid Archives (GARs):** A Java EE-like artifact that encapsulates all the elements of a Coherence application into a single deployable unit



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic 12c provides two new features for Coherence.

- **Managed Coherence servers:** With earlier versions of WebLogic Server, developers could manage Coherence by using WebLogic node manager. However, the process was often difficult and error-prone, and ultimately took advantage of the management capabilities of only a WLS cluster to manage Coherence instances. Coherence was still a stand-alone process. With WebLogic Server 12c, Coherence is fully integrated into WebLogic as a first-class member of the cluster, with the ability to take advantage of all aspects of the Java EE as well as all aspects of WebLogic Server management.
- **Coherence Grid Archives (GARs):** Additionally, Coherence applications are now first-class citizens of a WebLogic Server cluster. Coherence artifacts, including all configuration and deployment information, are packaged into a single file, known as a GAR. A GAR contains everything required to deploy and run Coherence applications. To make this happen, WebLogic now supports an additional container, often referred to as the Coherence Container. The Coherence Container provides all the same services as an EJB or similar container to Coherence applications. Lifecycle, injection, startup, shutdown, class management, isolation, and similar capabilities are provided for Coherence.

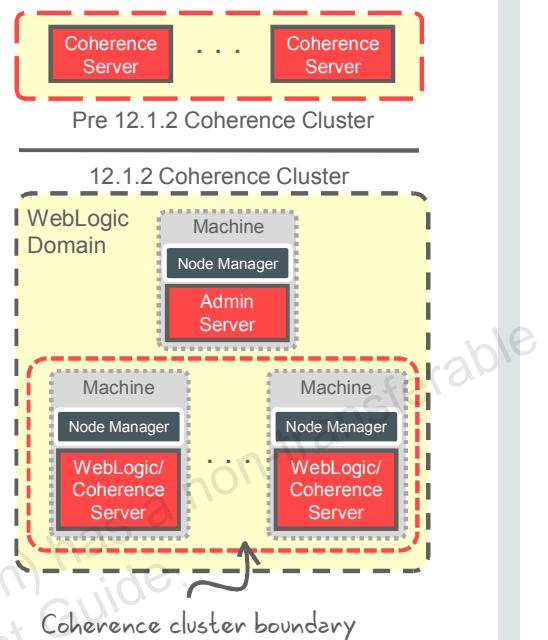
# WebLogic Managed Coherence Servers

## Operations

The WebLogic Server Container for Oracle Coherence:

- Is a simplified model for managing operational configuration
- Is conceptually similar to other Java EE containers
- Is fully integrated and installed with WebLogic and includes Coherence
- Is managed via the configuration wizard, WebLogic console, WLST, JMX, and Fusion Middleware Control
- Manages Grid Archives (GARs)

*More on what a GAR is later*



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Traditionally, Coherence has been deployed as a JAR, incorporated into a Java application along with a tier of stand-alone cache server JVMs. For example, the use of embedded Coherence from within a WAR, was referred to as clients and the stand-alone cache servers were referred to as servers. The life cycles of the “clients” and cache servers were managed separately, often manually. Application development and deployment in this model was a complicated process involving many moving parts that required custom management processes.

The WebLogic Server Coherence Container for applications provide tight integration between WebLogic Server and Coherence and eliminate the difficulty of managing Coherence instances and applications in a WebLogic environment. This integration allows for simplified and streamlined development and management environments of distributed applications. The Coherence Container allows end users to build a new archive type (GAR) that can be deployed and managed via standard WebLogic Server methods. Using the Coherence Container:

- Developers can now streamline their build processes to Coherence applications
- Operations departments can now standardize deployment of Coherence instances and Coherence servers

# What Is a Grid Archive (GAR)?

Coherence applications or GARs:

- Are structurally similar to enterprise/web applications, but are packaged as a Grid Archive with the extension .gar
- Define the elements required for deployment to the WebLogic Coherence Container
- Are composed of:
  - A deployment descriptor, coherence-application.xml
  - A cache configuration, named in the deployment descriptor
  - Java classes including entry processors, event handlers, filters, application classes, and others
  - Optional support libraries

Defines application name, cache configuration file name, POF configuration, lifecycle handler, and a variety of other information



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Coherence applications are packaged into Grid Archive or GAR files. In general, a GAR contains all the artifacts required to service a particular cache or invocation request and mirrors, to a high degree, the structure of other Java EE artifacts, such as web archives. Specifically, a GAR contains:

- A deployment descriptor, coherence-application.xml, which defines the location of a cache configuration as well as a variety of other content such as POF configuration. The deployment descriptor may include references to several other optional elements, such as lifecycle handlers.
- A cache configuration, referenced from the deployment descriptor
- Java classes representing the application. These classes typically include the entities and other application classes, POF support, entry processors, event handlers, and similar classes.
- A set of optional library JAR files

The entire set of classes is encapsulated into a JAR known as a Coherence Grid Archive, which mirrors similar constructs for web applications, enterprise applications, and JAR files in general.

GAR files are also deployable directly to stand-alone Coherence (without WebLogic present at all), reinforcing their JAR-like structure.

# Coherence Application Deployment on WebLogic

Coherence applications can be deployed in several ways:

- **Stand-alone:** GAR is deployed to a server by itself. It is analogous to the traditional default cache server.
- **Java EE:** The application is included as part of:
  - Web Application (WAR): GAR resources available to WAR only
  - Enterprise Application (EAR): GAR resources available to WAR and other Java EE resources, such as EJBs
- **Shared library:** GAR resources are available to all WARs and EARs located on the server.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Coherence applications can be deployed in several ways. Deployment models include:

- **Stand-alone:** A stand-alone deployed Coherence application exposes caches to applications, but does not expose resources directly to WAR or EAR files. Such a deployment is most common when configuring a data tier.
- **Java EE:** GAR files can be deployed as components of a WAR or an EAR file. Such deployments make Coherence resources available to the WAR and/or EAR directly. EJBs, web applications, JSPs, and other Java EE artifacts can use dependency injection to access a cache resource. Such a deployment scopes the caches to the EAR itself. An identical deployment, to a different EAR, would result in a similar set of caches, but scoped to the EAR/WAR.
- **Shared library:** With a shared library deployment, similar to a stand-alone deployment, all WAR and EAR artifacts can access the cache resources directly.

## Coherence Container: Benefits

- Simplified installation: Coherence and WebLogic installed together
  - Simplified operations management:
    - Configure, manage, and deploy Coherence applications from the WebLogic administration console and WLST.
    - Manage Coherence resources centrally.
- Stand-alone Coherence  
is still supported.
- Within a domain!

WebLogic to Coherence	Coherence benefits to WebLogic
<ul style="list-style-type: none"> <li>• Instance management</li> <li>• Dependency injection</li> <li>• Application lifecycle management</li> <li>• Scripting support (deployment etc)</li> </ul>	<ul style="list-style-type: none"> <li>• Simplified application packaging</li> <li>• Multiple deployment models           <ul style="list-style-type: none"> <li>– Stand-alone (GAR)</li> <li>– GAR within web application</li> <li>– GAR within Enterprise application</li> <li>– Shared library</li> </ul> </li> </ul>



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

With WebLogic Server 12c, Coherence is provided and installed with WebLogic Server. Both WebLogic Server and Coherence provide services to each other.

WebLogic Server provides several services to Coherence, including:

- Coherence, separate from WebLogic, cluster, member, and application management and scoping. Applications can be scoped to be stand-alone, WAR, EAR, or shared library scoped. Coherence also provides a cache-level scoping mechanism much like namespaces for caches and data.
- **Packaging:** Available on the system classpath, simplifying packaging
- **Management:** Coherence is now a first-class member of the cluster and fully supported by the console and WLST.
- **Instance support:** Full support for creating and managing Coherence instances
- **Application support:** Full support for application development and runtime life cycle
- **Cluster support:** Full support for defining and managing Coherence Clusters

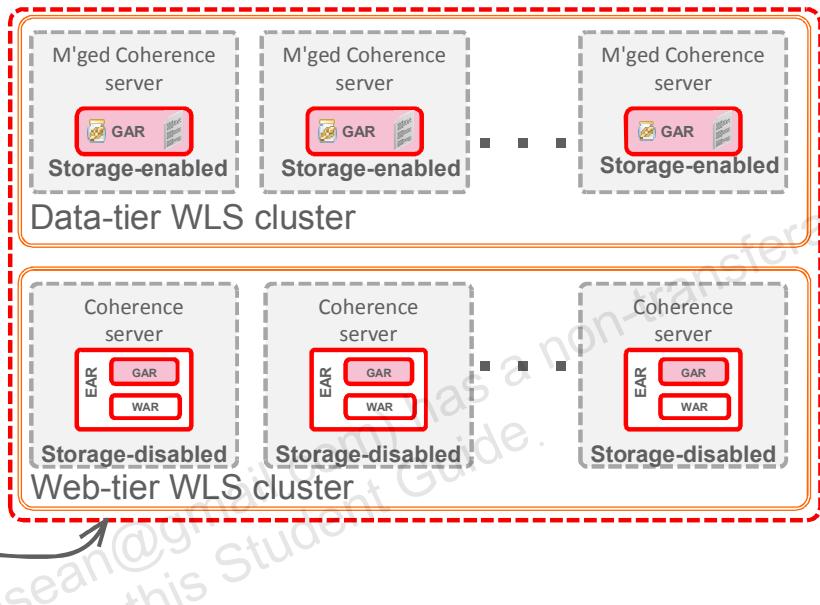
# Coherence Cluster

*Coherence clusters* consist of multiple, managed Coherence server instances.

Managed Coherence server can:

- Be storage enabled or disabled
- Share configuration and overrides
- Have one or more deployed Coherence applications

Coherence Cluster spans multiple WLS clusters.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Coherence clusters are a mechanism for logically grouping sets of managed Coherence servers. In general, Coherence clusters consist of multiple managed Coherence servers, which together form a distributed data grid. Coherence clusters are different from WebLogic Clusters in several ways. First and foremost Coherence clusters often span multiple WebLogic clusters and are intended to represent Coherence servers, which logically represent a single data grid. Additionally, Coherence clusters use different clustering protocols, ports, and are configured separately from WebLogic Clusters. Managed Coherence servers can be associated with a Coherence cluster or with a WebLogic Cluster, which itself is associated with a Coherence cluster.

Typically, a Coherence cluster is associated with one or more WebLogic Clusters, which together represent a tiered architecture, typically having data, web, and proxy tiers. In the sample shown, two WebLogic Clusters, a data-tier cluster and a web-tier cluster, are defined. These two WebLogic Clusters are bound together into a single Coherence cluster. Typically, such a cluster shares a set of Coherence configurations and applications. In the example shown, the same Coherence GAR is deployed to each Coherence managed server, but packaged differently. On the data tier, the application is packaged as a simple GAR and deployed stand-alone. On the web tier, the same GAR is packaged as a component of an EAR and accessible from any other Java EE application within that EAR.

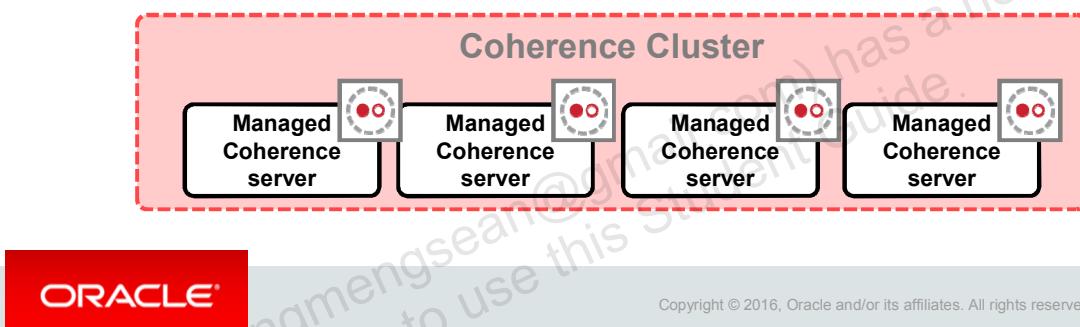
# Managed Coherence Server

A domain can have zero or more managed Coherence servers.

A managed Coherence server:

- Is a WebLogic managed server that encapsulates a Coherence instance
- Can be storage enabled or disabled
- Can host Coherence applications
- Is normally a member of a Coherence Cluster

May be a member of  
the default  
Coherence cluster



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Managed Coherence servers are server instances that are associated with a Coherence cluster. These servers work together and collectively form a Coherence cluster. Members of a Coherence cluster are historically referred to as Coherence instances, and are referred to as *managed Coherence servers* within a WebLogic installation. However, managed Coherence servers are *not* the same as Coherence servers. Managed Coherence servers are specific to the WebLogic Server Coherence Container integration. The use of Coherence servers, a resource defined in the previous ActiveCache integration, is deprecated. Managed Coherence servers conceptually are no different from earlier Coherence server instances running stand-alone, but there are differences. Some of the differences include:

- Managed Coherence servers that are managed within a WebLogic Server domain must not join an external Coherence cluster comprised of stand-alone JVM cluster members.
- Stand-alone JVM cluster members cannot be managed within a WebLogic Server domain.
- The administration server is typically not used as a managed Coherence server in a production environment.

Managed Coherence servers are distinguished by their role in the cluster, and can be:

- **Storage enabled:** A managed Coherence server that is responsible for storing data in the cluster. Coherence applications are packaged as Grid Archives (GARs) and are deployed on storage-enabled managed Coherence servers.

- **Storage disabled:** A managed Coherence server that is not responsible for storing data and is used to host Coherence applications (cache clients). A Coherence application GAR is packaged within an EAR and deployed on storage-disabled managed Coherence servers.
- **Proxy:** A managed Coherence server that is storage disabled and allows external clients (non-cluster members) to use a cache. A Coherence application GAR is deployed on managed Coherence proxy servers.

## Quiz



Which two of the following are supported by GAR deployments?

- a. GAR files do not support deployment
- b. Deployment to standalone Coherence without WebLogic
- c. Deployment to a WebLogic server within an EAR file
- d. None of the above

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer:** b, c

## Summary

In this lesson, you should have learned how to configure:

- Coherence\*Web
- Managed Coherence Servers



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 16-2 Overview: Configuring Managed Coherence Servers

This practice covers the following topics:

- Configuring a new Coherence cluster
- Deploying a stand-alone GAR application
- Deploying an EAR application with an embedded GAR
- Configuring Coherence cluster settings
- Testing the Coherence cluster



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.

# Application Staging and Deployment Plans

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to:

- Configure a server's staging mode
- Create and use deployment plans



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

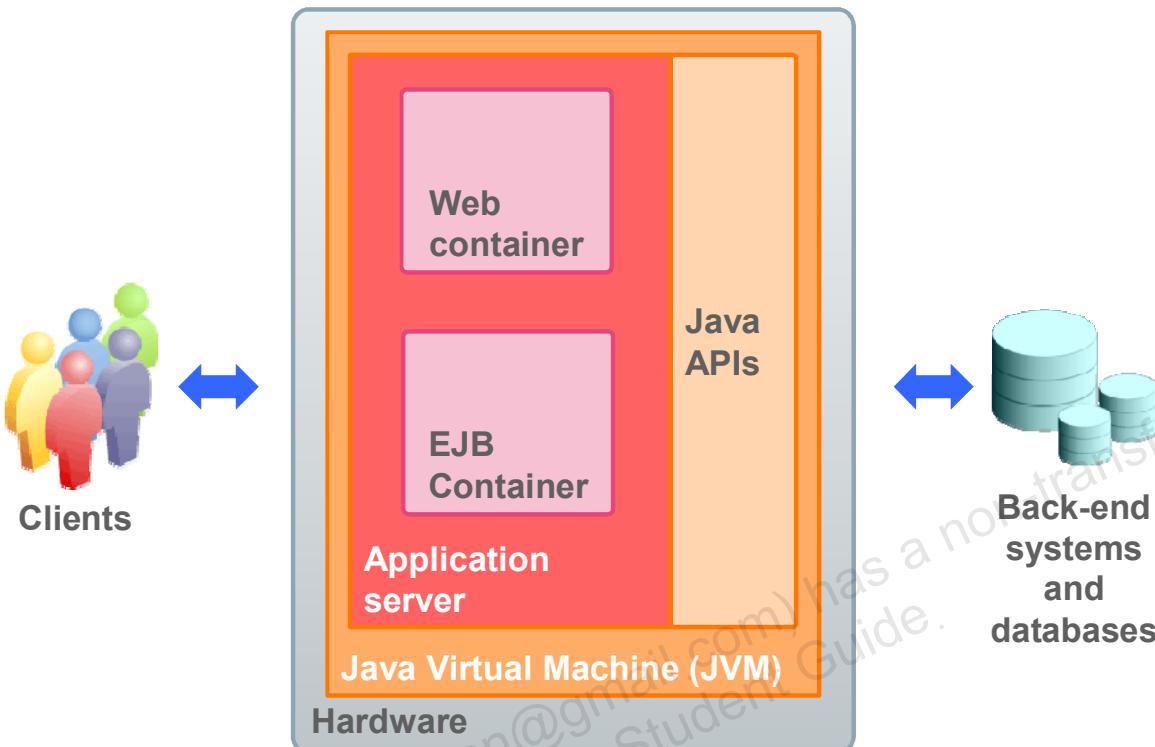
# Agenda

- Review:
  - Java EE applications
  - Deployment
- Server staging modes
- Deploying an application to multiple environments
- Java EE deployment descriptors and annotations
- Deployment plans
- Deployment plan tools



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Review: Java EE Applications



ORACLE®

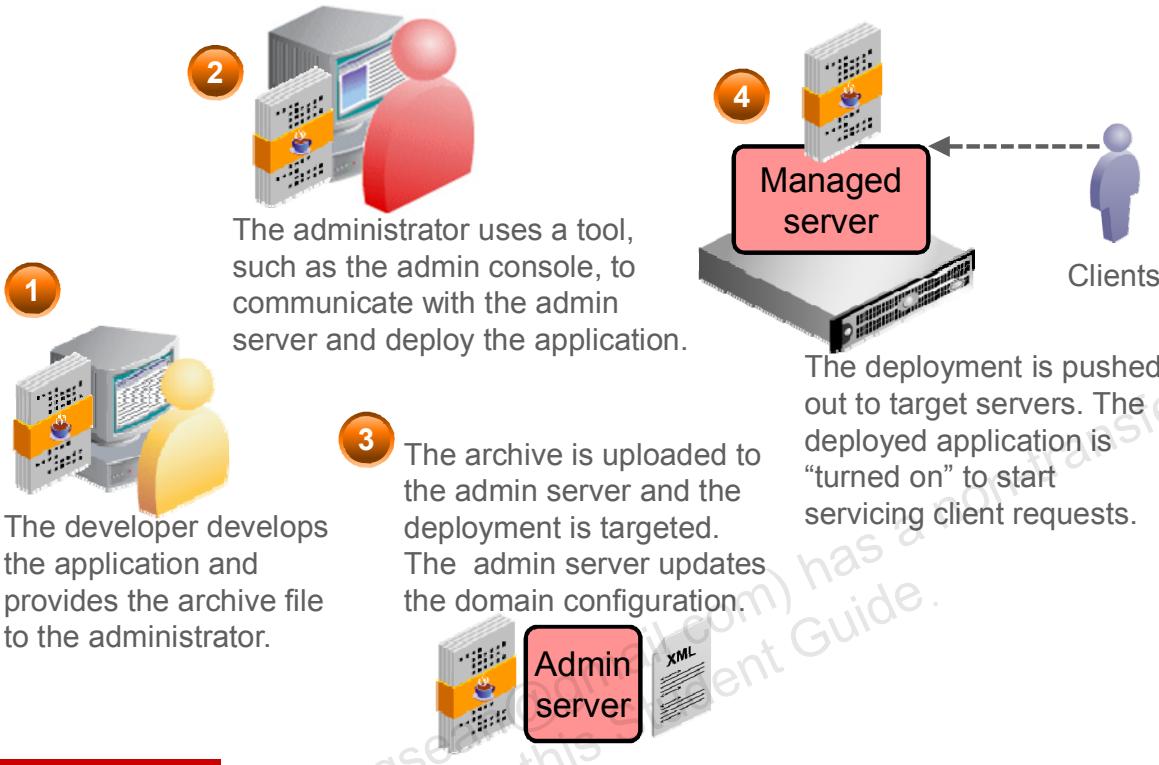
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Java Enterprise Edition (EE) platform consists of the following:

- Application components including enterprise applications (EAR), web applications (WAR), applets, servlets and JSPs, Enterprise JavaBeans (EJB), and so on
- The containers that provide the runtime support for the components
- Resource manager drivers that implement network connectivity to external resource managers
- Various specifications and application program interfaces (APIs) including Java Transaction API (JTA), Java Database Connectivity (JDBC), Java Message Service (JMS), and Java Naming and Directory Interface (JNDI)

All of this Java code runs within a Java Virtual Machine (JVM).

## Review: Deployment



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. First, an application must be developed, tested, and packaged (usually as an application archive).
2. The application archive file is placed somewhere that an administrator in charge of deployment can access it. A deployment tool, like the administration console, is used to communicate with the administration server of the domain, which is in charge of any configuration changes, including application deployment.
3. The deployment tool gives the administration server access to the application, and allows the deployment administrator to target a server (or servers) on which to run the application. The administration server updates the domain's configuration.
4. The administration server pushes the application's code out to the target managed server (or servers). After the application is “activated” (told to start servicing requests), it becomes accessible to clients.

# Agenda

- Review
- Server staging modes
  - Staged mode example
  - No stage mode example
  - External stage mode example
  - Configuring the staging mode
- Deploying an application to multiple environments
- Java EE deployment descriptors and annotations
- Deployment plans
- Deployment plan tools



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Server Staging Modes

The screenshot shows the 'Settings for server1' page in the Oracle WebLogic Server Administration Console. The 'Deployment' tab is selected. In the 'Staging Mode' section, there is a dropdown menu with three options: 'stage', 'nostage', and 'external stage'. A tooltip for the 'stage' option states: 'You can configure deployment per server or for each application.' The 'Save' button is visible at the bottom.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

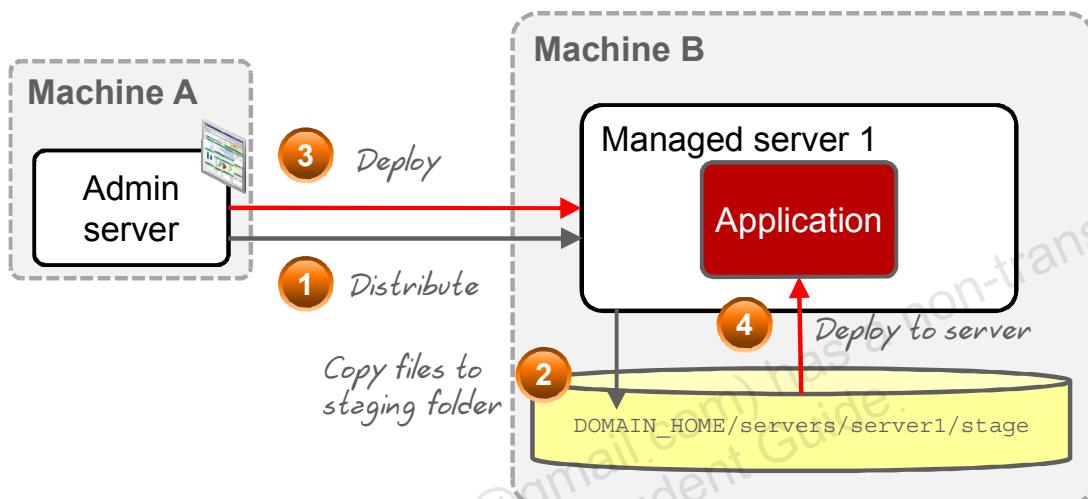
WebLogic provides you with control over whether or not, where, and by whom application files are copied before being deployed. All applications that are targeted to the managed servers can be copied by the administration server to the managed server before being prepared. This is called staging, and the files are staged to a configurable staging area for each server. There are three kinds of staging:

- **stage:** (default) Files are copied to the preconfigured staging directory for preparation and activation.
- **nostage:** Files are deployed from a static location.
- **external stage:** Files are copied by a user or a third-party tool (for example, JDeveloper or Eclipse) before deployment.

Applications can be deployed from the source location by changing the configuration (StagingMode). If the managed servers are running on a machine other than the administration server, it means that either the source location is on a file system that is accessible to the managed server (StagingMode = `nostage`) or the user or third-party tool performs the copy before the deployment request is issued (StagingMode = `external stage`).

## Staged Mode Example

The administration server first copies the deployment files to the staging directories of target servers.



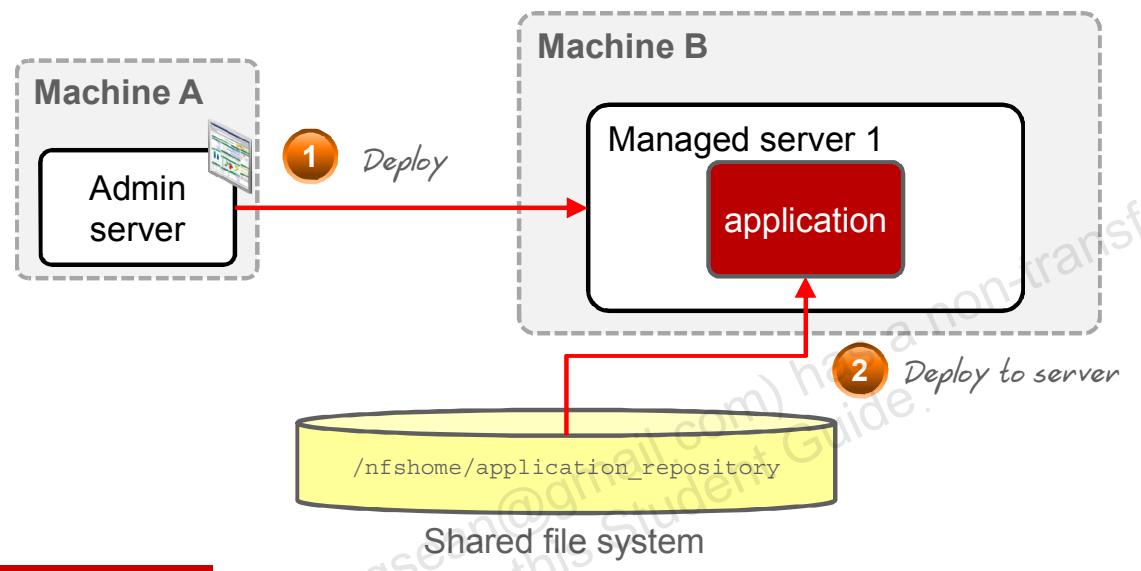
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a server is configured with the `stage` staging mode, it causes the administration server to first distribute deployment files to the server's configured `stage` folder so the files are available to the server for the deployment process. This staging mode is best used for small to medium-sized applications to multiple WebLogic Server instances or a cluster. If applications are very large in size, then other staging modes may provide a better deployment experience. The `stage` mode deployment process involves:

1. The first aspect of the deployment process causes the administration server to execute the distribution process, which copies the deployment's files to the `stage` folder that is configured for the target server.
2. The administration server copies the files to the `stage` folder of the server instance. This is done by sending the files to the server instance and the instance places the files in its staging folder.
3. After the distribution phase of deployment is complete, the administration server executes the deployment process on all targeted servers.
4. The target server receives the deployment command, performs its internal preparation of the deployment, and performs the actual deployment of the application on the server.

## No Stage Mode Example

The administration server does not copy deployment files. All servers deploy the same physical copy of files from a location that is accessible to all servers.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a server is configured with the `nostage` staging mode, the administration server only issues the deployment command to all targeted servers because the files are already accessible to all servers involved. When the `nostage` staging mode is used in conjunction with exploded deployment files, WebLogic automatically detects changes to web components and refreshes the deployment to reflect the latest state. This staging mode is best used for deploying:

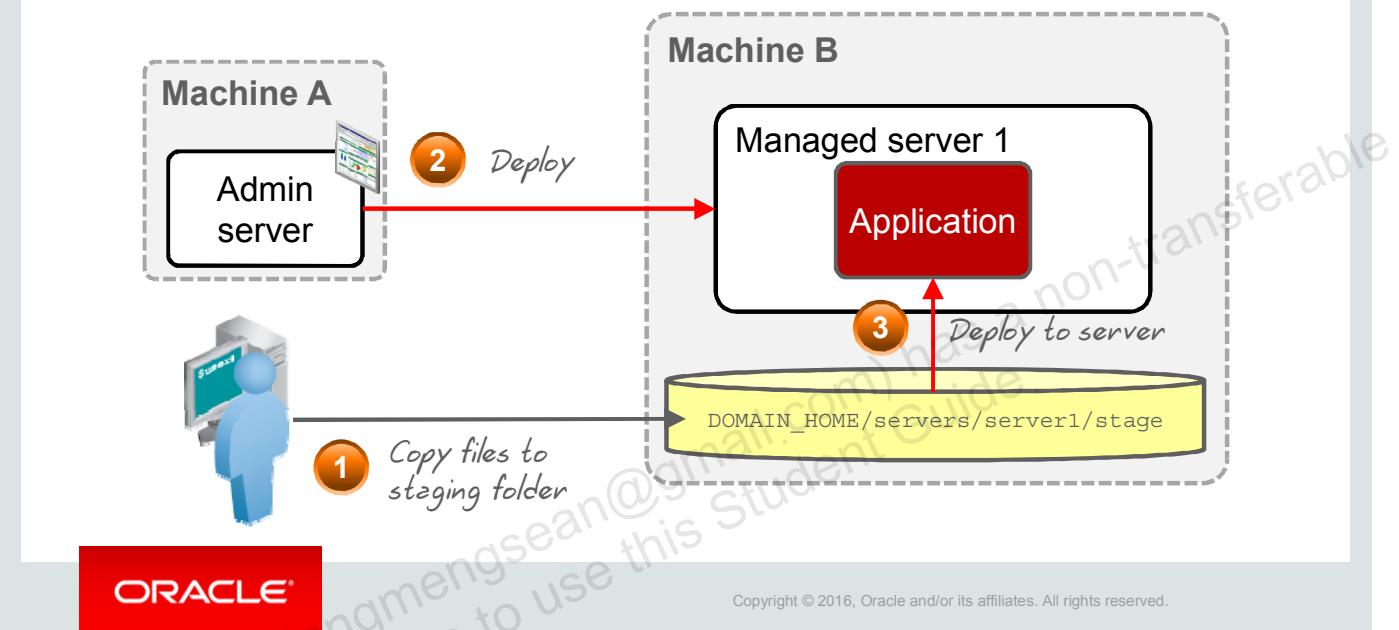
- To a single server domain because all files would be local to the administration server by default
- To a cluster on a multihomed machine to avoid multiple copies of the same files on one machine
- Very large applications to multiple targets to avoid multiple copies that take up a lot of disk space and to leverage the infrastructure of a shared file system for faster deployments
- Exploded deployments to refresh any changes
- Applications that require dynamic updates of deployment descriptors

The nostage mode deployment process involves the following steps:

1. The administration server executes the deployment process on all targeted servers.
2. The target server receives the deployment command, performs its internal preparation of the deployment and the actual deployment of the application on the server using the files available on shared storage.

## External Stage Mode Example

The administration server does not copy deployment files. The administrator must ensure that all files are distributed and available to all servers before deployment.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a server is configured with the external stage staging mode, the administration server does not take responsibility for distributing deployment files to targeted servers. The WebLogic administrator must ensure that all deployment files are distributed before performing a deployment. File distribution can take place in the form of running a script, using a third-party tool, manually copying files, and so on. All that matters is that the files are distributed before performing a deployment. When using the external stage staging mode, the administration server requires a copy of the deployment files, which it uses for validation purposes during deployment. WebLogic supplies the `-noversion` option to disable this requirement, but this also renders production redeployment impossible.

This staging mode is best used for deployments:

- Where an administrator requires or wants manual control over file distribution
- Where other mechanisms used in operations procedures manage file distribution, such as scripts or third-party systems
- That do not require dynamic updating or partial redeployment

The external stage mode deployment process involves:

1. The WebLogic administrator performs the operational process that copies deployment files to the `stage` folder that is configured for the target server.
2. The WebLogic administrator performs a deployment operation on WebLogic. The administration server executes the deployment process on all targeted servers.
3. The target server receives the deployment command, performs its internal preparation of the deployment, and performs the actual deployment of the application on the server.

# Configuring the Staging Mode

Override default server staging mode during deployment

Use this page to define the default deployment staging configuration for this server.

**Staging Mode:**

- stage
- stage
- nostage
- external stage

**Staging Directory Name:** /u01/domains/part2/wlsadmin

**Upload Directory Name:** /servers/server1/upload

**Source Accessibility**

How should the source files be made accessible?

Use the defaults defined by the deployment's targets  
Recommended selection.

Copy this application onto every target for me  
During deployment, the files will be copied automatically to the Managed Servers to which the application is deployed.

I will make the deployment accessible from the following location

**Location:** /practices/part2/apps/solution/SimpleAuctionWebApp.war

**Save**

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The server staging mode specifies a staging mode for a server when no staging mode is indicated at the time of deployment. The default staging mode for the administration server is `nostage`, and the default staging mode for managed servers is `stage`. You can configure the staging mode of a server using the administration console by selecting a server's Configuration > Deployment tab. This page provides three fields for configuring the server's staging mode:

- **Staging Mode:** This is where you specify the default staging mode used by this particular server.
- **Staging Directory Name:** This is the relative path to the folder where deployment files are stored for this server before deployment.
- **Upload Directory Name:** When a remote client not running on the machine where the administration console is located is used to deploy an application, such as WLST, the command uploads the deployment files to this upload folder on the administration server. This specifies a relative path of the administration server used before beginning the staging and deployment process.

## Quiz



The \_\_\_\_\_ staging mode requires that all servers have shared access to deployment files.

- a. Stage
- b. No Stage
- c. External stage
- d. Shared stage

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

# Agenda

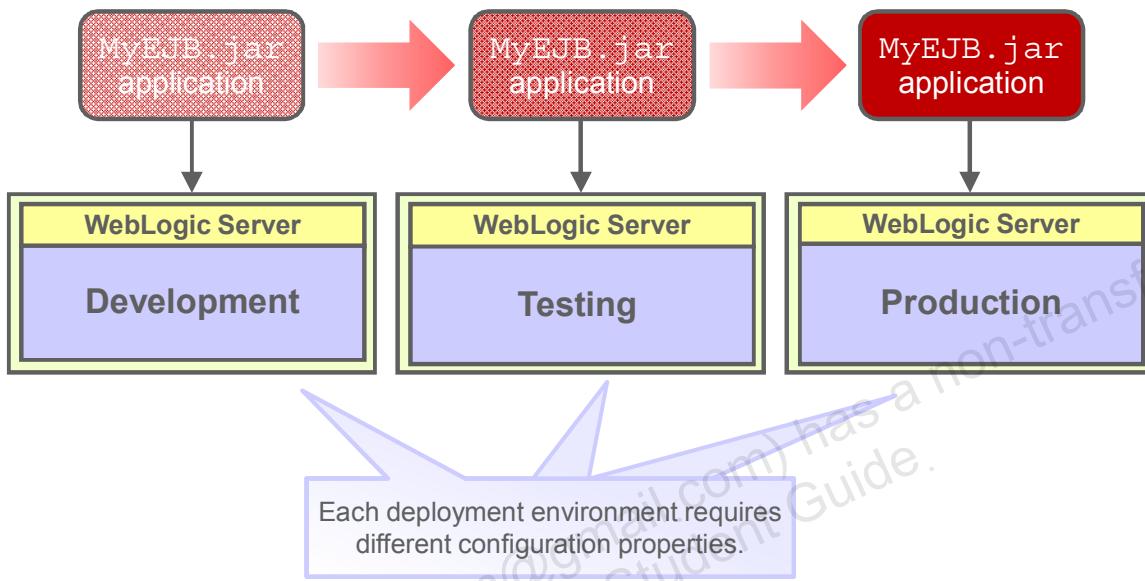
- Review
- Server staging modes
- Deploying an application to multiple environments
  - Development to test to production
  - Examples of deployment descriptor values to change
- Java EE deployment descriptors and annotations
- Deployment plans
- Deployment plan tools



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Development to Test to Production

Applications are deployed to several different environments during a full development life cycle:



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The full development life cycle of a WebLogic application involves some variation of development, test, and production environments. Whether it includes multiple levels of any of these environments is irrelevant, such as a development environment for each individual developer and an integrated development environment to see how components coexist with components from other developers. The main point is that the application is going to begin in a development phase, graduate to a testing environment, and graduate again into a live production environment. Each environment presents a new ecosystem in which the application gets deployed. The application must be able to adapt accordingly in order to function in each environment. This means that the application's configuration properties must change from one environment to the next.

## Examples of Deployment Descriptor Values That Can Change

Deployment Descriptor	Description	Example Properties
web.xml weblogic.xml	Contain standard and WebLogic specific properties for web applications	Standard: <login-config> WebLogic: <context-root>
ejb-jar weblogic-ejb-jar persistence.xml	Contain standard and WebLogic specific properties for EJB applications	Standard: <jta-data-source> WebLogic: <jndi-name>
application.xml weblogic-application.xml	Contain standard and WebLogic specific properties for enterprise applications	Standard: <library-directory> WebLogic: <min-threads-constraint> <coherence-cluster-ref>



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

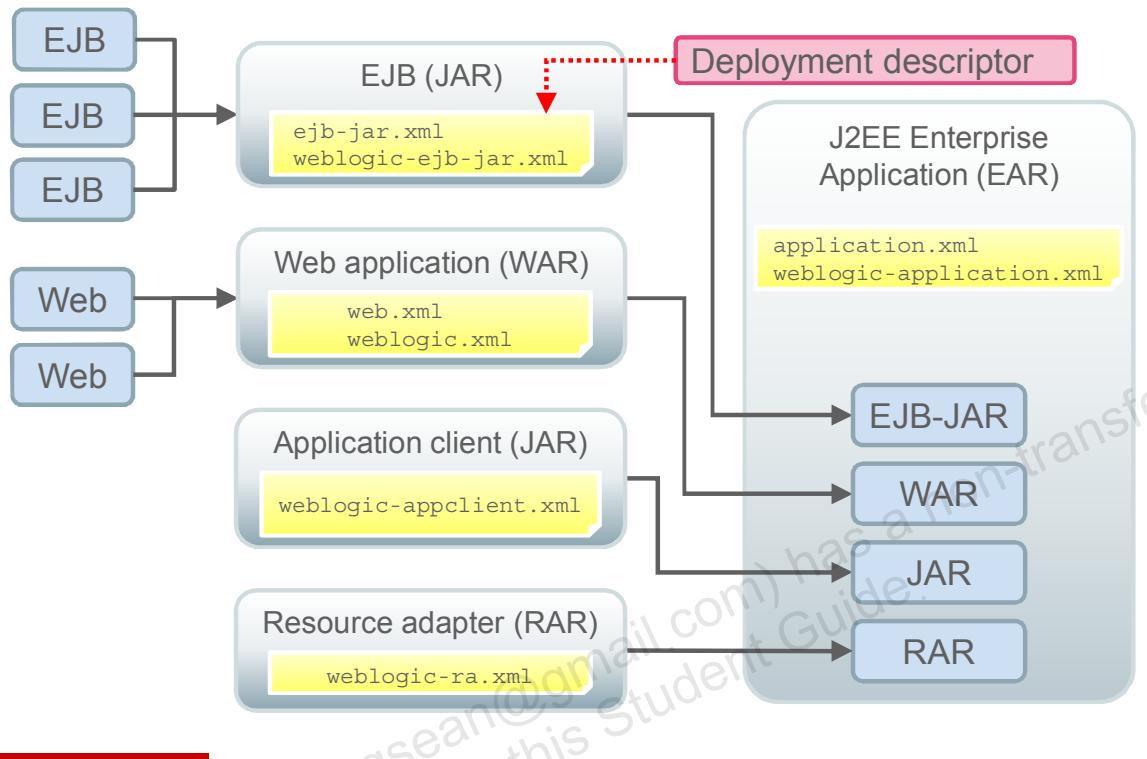
# Agenda

- Review
- Server staging modes
- Deploying an application to multiple environments
- Java EE deployment descriptors and annotations
  - Java EE deployment descriptors
  - Java EE annotations
  - appmerge and appc
- Deployment plans
- Deployment plan tools



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Java EE Deployment Descriptors



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Java EE deployment descriptors configure Java EE application features. These configuration settings are different depending on the type of component the descriptor represents. Deployment descriptors are useful because they are external to the code of an application and provide a way for developers to include configuration settings along with their applications. The drawbacks to deployment descriptors include:

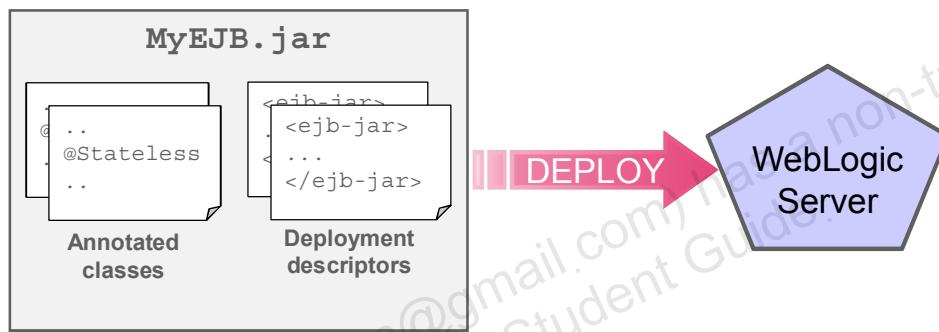
- Sometimes difficult to learn and format because each is defined by an XML schema that is not visibly available when editing files. Administrators are not typically familiar with an application's code and it is possible that they lack the knowledge required to make descriptor changes directly.
- Difficult to modify because components are typically provided in the form of a packaged archive file with the descriptors contained within the archive. Modification requires unpacking the archive, modifying the descriptors, and repacking the archive correctly. Descriptors almost always require some modification during the full development life cycle when they move from development to test, and then from test to production.

# Annotations

```
package com.examples;

@Stateless
@Local(MyEJBInterface.class)
public class MyEJB {
    @Resource(mappedName = "jdbc/employeeDatabase")
    private DataSource myDS;
    ...
}
```

MyEJB.java



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A Java annotation is a way of adding metadata to Java source code that is also available to the program at run time. Annotations are an alternative to deployment descriptors, that were required by J2EE 1.4 and earlier. In fact, Java EE metadata annotations now make standard deployment descriptors optional. Annotations are realized by WebLogic during deployment, just like deployment descriptors. Developers use annotations to simplify development, by specifying within the Java class how the application component behaves in the container. Annotations make development easier, but how do they affect administrators?

- Developers are no longer required to create deployment descriptors to configure their components. As a result, some configuration hooks that would have previously been present in the descriptors of components passed from developers to administrators are no longer available by default.
- Administrators have no visibility into what configuration is contained within the annotations of Java source files by default. This is because components passed to administrators for deployment do not include source code, and administrators are not always developers who understand the syntax of annotations.

These situations are avoided by following some best practices and leveraging some tools that make the configuration provided by annotations visible to administrators.

## appmerge and appc

WebLogic provides tools to debug annotations and deployment descriptors. These tools can work together to troubleshoot problems in your configuration.

### -writeInferredDescriptors flag

```
$ java weblogic.appmerge  
-writeInferredDescriptors  
-output app.debug/app.jar  
MyEJB.jar
```

Command line

### metadata-complete attribute

```
<ejb-jar  
metadata-complete="true">  
...  
</ejb-jar>
```

ejb-jar.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic's appmerge and appc tools have a “-writeInferredDescriptors” flag to see how the container views the configuration of deployments.

These tools process the descriptors and annotations of your component like the container does and write everything out to descriptors for you to review.

The example in this slide shows how to use “-writeInferredDescriptors” with the appmerge tool.

The “-writeInferredDescriptors” flag automatically sets “metadata-complete” to true in deployment descriptors.

Setting “metadata-complete” to true in a deployment descriptor instructs annotation processors to ignore the annotations in your deployments and only use what is contained in deployment descriptors.

The example below shows the “metadata-complete” attribute set to true.

As an administrator, you will most likely not execute the appc tool because you do not customarily compile Java EE programs. You will find using appmerge to be a productive tool for making the configuration of your applications more visible.

## Quiz



Developers use Java EE annotations. Why are they important to administrators?

- a. Because some annotations are realized as deployment properties
- b. Because annotations override deployment descriptor properties
- c. Annotations are not important to administrators
- d. All of the above

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: a**

# Agenda

- Review
- Server staging modes
- Deploying an application to multiple environments
- Java EE deployment descriptors and annotations
- Deployment plans
  - What is a deployment plan?
  - Using deployment plans for different environments
  - Deployment plan example
  - Staging deployment plans
- Deployment plan tools

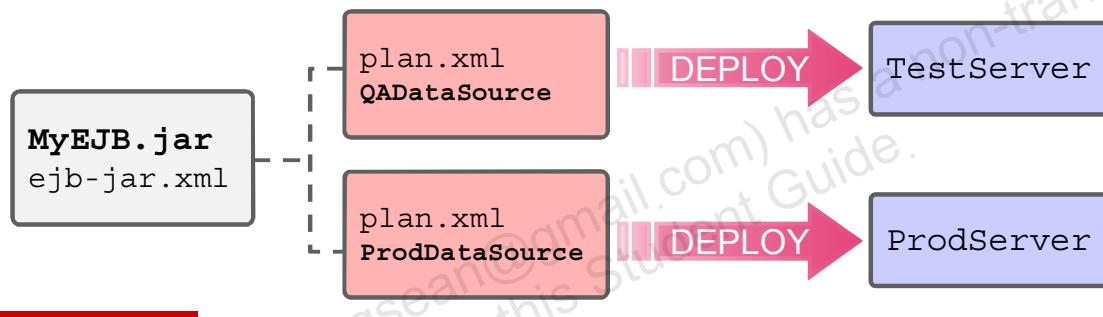


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# What Is a Deployment Plan?

A Java EE deployment plan:

- Is an optional XML file associated with an application
- Resides outside an application archive
- Sets or overrides the values in Java EE deployment descriptors and annotations
- Allows easy customization of a single application for deployment to multiple deployment environments



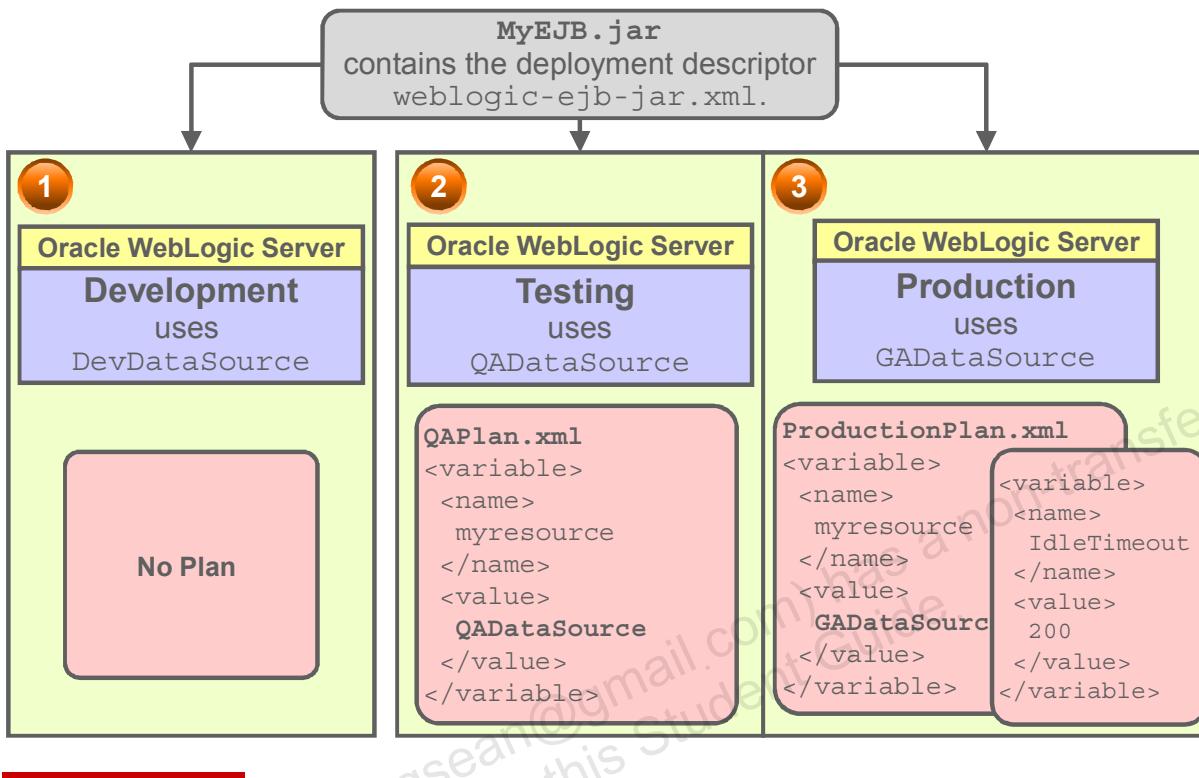
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A deployment plan is an XML document that is used to define an application's deployment configuration for a specific Oracle WebLogic Server environment, such as development, test, or production. A deployment plan resides outside of an application's archive file and contains deployment properties that override an application's existing Java EE and Oracle WebLogic Server deployment descriptors. Use deployment plans to easily change an application's Oracle WebLogic Server configuration for a specific environment without modifying the existing deployment descriptors. Multiple deployment plans can be used to reconfigure a single application for deployment to multiple, differing Oracle WebLogic Server domains or servers.

Any external resources required by the application are subject to change when the application is deployed to a different environment. For example, the Java Naming and Directory Interface (JNDI) names of the data sources that are used in your development environment can be different from those used in testing or production. Exposing those JNDI names as variables makes it easy for people deploying applications to use the available resources or create the required resources when deploying the application.

Certain tuning parameters that are acceptable in a development environment are unacceptable in a production environment. For example, it may suffice to accept default or minimal values for EJB caching on a development machine, whereas a production cluster would need higher levels of caching to maintain acceptable performance. To deploy the application to a new environment, an administrator simply creates or uses a new deployment plan as necessary.

# Using Deployment Plans for Different Environments



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Development:** A developer develops and creates both Java EE and WebLogic Server deployment descriptors to configure the application for repeated deployments to the development environment. The development server uses a simple Derby database for development, named “`DevDataSource`,” and the `weblogic-ejb-jar.xml` descriptor identifies the resources for the application.
- Testing:** The developer packages the application into an archive file and delivers it to the administrator in the QA team. The testing environment uses a different data source named “`QADatasource`.” At this point, the embedded deployment descriptors provide a configuration that is valid for the development environment used by the developer, but is not valid for the testing environment where the application must be deployed for testing. To deploy the application, the administrator of the testing environment generates a deployment plan `QAPlan.xml` to override the data source name configured in the application’s embedded deployment descriptors.
- Production:** Similarly, when the application is released into production, the administrator of the staging or production environment creates or uses another deployment plan to configure the application. The production deployment plan `ProductionPlan.xml` again overrides the application deployment descriptors to identify a new JDBC data source “`GADatasource`” that is used by the production environment. For this environment, the deployment plan also defines tuning parameters to make better use of the additional resources that are available in the production domain.

Organizations that have numerous deployment environments that frequently change should use a configuration workflow with multiple deployment plans. In a multiple deployment plan workflow, each deployment plan is owned by the deployer of the application rather than the development team. You should store each deployment plan for a single application in its own plan subdirectory of the application's root directory. The multiple deployment plan configuration workflow works in the following way:

1. The development team releases a version of the packaged application deployment files (containing Java EE and WebLogic Server descriptors). The development team may or may not include a template deployment plan with exported variables for resource definitions or common tunable parameters.
2. Before deploying the application, each deployer generates a custom deployment plan to configure the application for the respective target environment. A custom deployment plan can be created by starting with a template deployment plan (or no deployment plan) and making changes to the application's deployment configuration using the Administration Console.
3. After defining the deployment configuration for the respective environment, each deployer retrieves the custom deployment plan and maintains it for future deployments of the application. It is recommended that you store the custom configuration plans in a source control system so that new versions can be tracked and reverted if necessary.
4. For subsequent releases of the application, each deployer uses the respective customized deployment plan to configure the application for deployment. Using the customized plan allows people deploying applications to perform deployments with `weblogic.Deployer` or automate deployments using WLST.

# Deployment Plan Example

The diagram illustrates the structure of the `persistence.xml` file and the XPath expression used to identify a specific element. The file content is:

```

<?xml version="1.0" encoding="UTF-8"?
<persistence version="2.0"
  xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="AuctionPU" transaction-type="JTA">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <jta-data-source>jdbc/AuctionDB</jta-data-source>
    <class>com.oracle.model.Auction</class>
    <class>com.oracle.model.Image</class>
    <class>com.oracle.model.Bid</class>
    <properties>
      </properties>
    </persistence-unit>
</persistence>

```

Annotations explain the structure:

- root-element**: Points to the outermost element, `<persistence>`.
- persistence-unit [name="AuctionPU"]**: Points to the `<persistence-unit name="AuctionPU" ...>` element.
- Original value**: Points to the text `jdbc/AuctionDB` within the `<jta-data-source>` element.
- persistence.xml**: Points to the file name at the bottom right.

**XPath to value:**

```
/persistence/persistence-unit[name="AuctionPU"]/jta-data-source
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows an example of a deployment descriptor for a Java Persistence API (JPA) component, called `persistence.xml`. This file contains the configuration of a data source JNDI name used to access the database connections required for the related component. The current value contained within the descriptor that is packaged within the deployed archive is `jdb/AuctionDB`.

For those who may not be familiar with XPath, this slide shows the syntax used to pinpoint the element that contains the value of the JNDI name of the data source in this file. Following from left to right in the XPath statement, the first element is `persistence`. The next element is `persistence-unit`, but it also contains criteria to distinctly identify which `persistence-unit` element using `[name="AuctionPU"]`. The next element is `jta-data-source`, which contains the target value you want to change.

# Deployment Plan Example

```

<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-plan
    http://xmlns.oracle.com/weblogic/deployment-plan/1.0/deployment-plan.xsd">
  <application-name>SimpleAuctionWebAppDbSec.war</application-name>
  <variable-definition>
    <variable>
      <name>PersistenceUnit_AuctionPU_jtaDataSource_13620081523747</name>
      <value>jdbc/AuctionDBTest</value>
    </variable>
  </variable-definition>
  <module-override>
    <module-name>SimpleAuctionWebAppDbSec.war</module-name>
    <module-type>war</module-type>
    ...
    <module-descriptor external="false">
      <root-element>persistence</root-element>
      <uri>WEB-INF/classes/META-INF/persistence.xml</uri>
      <variable-assignment>
        <name>PersistenceUnit_AuctionPU_jtaDataSource_13620081523747</name>
        <xpath>/persistence/persistence-unit[name="AuctionPU"]/jta-data-source</xpath>
        <operation>replace</operation>
      </variable-assignment>
    </module-descriptor>
  </module-override>
  <config-root>/apps/plans</config-root>
</deployment-plan>

```

Put this new value

Into this element

Plan.xml



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows a deployment plan that is used to override the `jdbc/AuctionDB` value set in the original `persistence.xml` file shown in the previous slide to `jdbc/AuctionDBTest`.

The basic elements in a deployment plan serve the following functions:

- `deployment-plan`: Encapsulates the deployment plan's contents
- `application-name`: Corresponds to the deployment name for the application or module
- `variable-definition`: Defines one or more variable elements. Each variable element defines the name of a variable that is used in a plan and a value to assign, which can be null. In this example, the variable is

`PersistenceUnit_AuctionPU_jtaDataSource_13620081523747` with a value of `jdbc/AuctionDBTest`, which is the new JNDI name of the data source for the `AuctionPU` persistence unit defined in the application's `persistence.xml` file. The variable name of this variable is matched with the name of a variable assignment defined in the `module-override` section below.

- **module-override**: Elements that define each module name, type, and deployment descriptor that the deployment plan overrides. A `module-descriptor` element can optionally contain a `variable-assignment` that identifies a variable name that is used to override a property in the descriptor and the exact location within the descriptor where the property is overridden. In this example, the `module-descriptor` for the `persistence.xml` file is defined and the elements are:
  - `root-element`: The name of the root XML element contained in the file specified by the `<uri>` element
  - `uri`: The relative path to the descriptor file that this `<module-descriptor>` element represents
  - `variable-assignment`: Contains a list of variable assignment definitions used to indicate which element within the descriptor XML file gets overridden
    - `name`: The name that is matched against the name of the variable definition section
    - `xpath`: The XPath syntax used to isolate the exact element to modify within the original descriptor XML file
    - `operation`: Set to `replace` or `remove`. Use `replace` to override the original value and `remove` to remove the original value from the configuration.
- `config-root`: The path where the deployment plan XML file is located

**Note:** You can change the generated name

`PersistenceUnit_AuctionPU_jtaDataSource_13620081523747` to any name you want. This name was originally generated by the administration console when creating the initial deployment plan. The name is insignificant and is only used for matching the variable definition section with the variable assignment section. It is initially generated so that the name is descriptive enough to indicate which setting is represented.

# Staging Deployment Plans

WebLogic allows you to specify staging for deployment plans:

Staging	Description
-planstage	Copies the deployment plan to the target server's staging directory
-plannostage	Does not copy and instead leaves deployment plan in a fixed location
-planexternal_stage	Does not copy and requires a separate copy step prior to deployment

```
$ java weblogic.Deployer  
  -adminurl http://host01.example.com:7001 -username weblogic  
  -password Welcome1 -deploy -targets cluster1  
  -plan /plans/prod/plan.xml  
  -planstage /deployments/AuctionDbLib.war
```

Command Line

```
> deploy(appName='MyApp',  
        path='/apps/MyApp.ear',  
        targets='cluster1',  
        planPath='/plans/prod/plan.xml',  
        planStageMode='stage')
```

WLST Command Line



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Planstage is the default if no deployment plan staging options are specified. Deployment plan staging allows you to specify different staging options for applications and deployment plans.

## Quiz



Which is NOT true about the deployment plans in WebLogic?

- a. Overrides values in application descriptors
- b. Can be created by WebLogic during deployment
- c. Is packaged within an application archive
- d. Is an XML file

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

### Answer: c

Remember that deployment plans are not packaged within an application, so that the same application can be reused in multiple deployment environments.

# Agenda

- Review
- Server staging modes
- Deploying an application to multiple environments
- Java EE deployment descriptors
- Deployment plans
- Deployment plan tools
  - Creating a deployment plan
  - `weblogic.PlanGenerator`
  - Using plan generator
  - Administration console



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Creating a Deployment Plan

- Tools for creating a deployment plan:
  - Administration Console
  - WLST
  - `weblogic.PlanGenerator`
  - Development tool (for example, JDeveloper or Eclipse)
- Goals for creating a deployment plan:
  - To expose the external resource requirements of the application as variables in the deployment plan
  - To expose additional configurable properties, such as tuning parameters as variables in the deployment plan



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Exporting an application's deployment configuration is the process of creating a custom deployment plan that administrators can use for deploying the application to new WebLogic Server environments. You distribute both the application deployment files and the custom deployment plan to deployers; for example, testing or production administrators, who use the deployment plan as a blueprint for configuring the application for their environment.
- An administrator can install both the application and the custom deployment plan by using the administration console, which validates the deployment plan and indicates when specific configuration properties must be supplied before deployment.
- `weblogic.PlanGenerator` creates a template deployment plan with null variables for selected categories of deployment descriptors. This tool is recommended if you are beginning the export process and you want to create a template deployment plan with null variables for an entire class of deployment descriptors. You typically need to modify the deployment plan created by `weblogic.PlanGenerator` either manually or by using the administration console to delete extraneous variable definitions or add variables for individual properties.

- The administration console updates or creates new deployment plans as necessary when you change the configuration properties for an installed application. You can use the administration console to generate a new deployment plan or to add or override variables in an existing plan. The administration console provides greater flexibility than `weblogic.PlanGenerator` because it allows you to interactively add or edit individual deployment descriptor properties in the plan rather than export entire categories of descriptor properties.

# Creating a New Deployment Plan

- WebLogic Server includes tools to accelerate deployment plan creation:
  - The administration console:
    - Generates a skeleton `plan.xml` if a plan folder is detected with a newly deployed application
    - Updates the `plan.xml` when you use the console to modify deployment descriptor settings
  - WLST uses the `createPlan` option to generate a plan.
  - The `weblogic.PlanGenerator` Java class can also generate a skeleton `plan.xml` for an existing application.
  - Oracle Enterprise Pack for Eclipse (OEPE) leverages WebLogic tools to create deployment plans.



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Although it is not officially part of WebLogic, OEPE is an Oracle plug-in for Eclipse that leverages WebLogic tools and the Eclipse framework to create deployment plans for applications. Developers may use OEPE tools to create plan templates to make it easy for administrators to change settings in their applications.

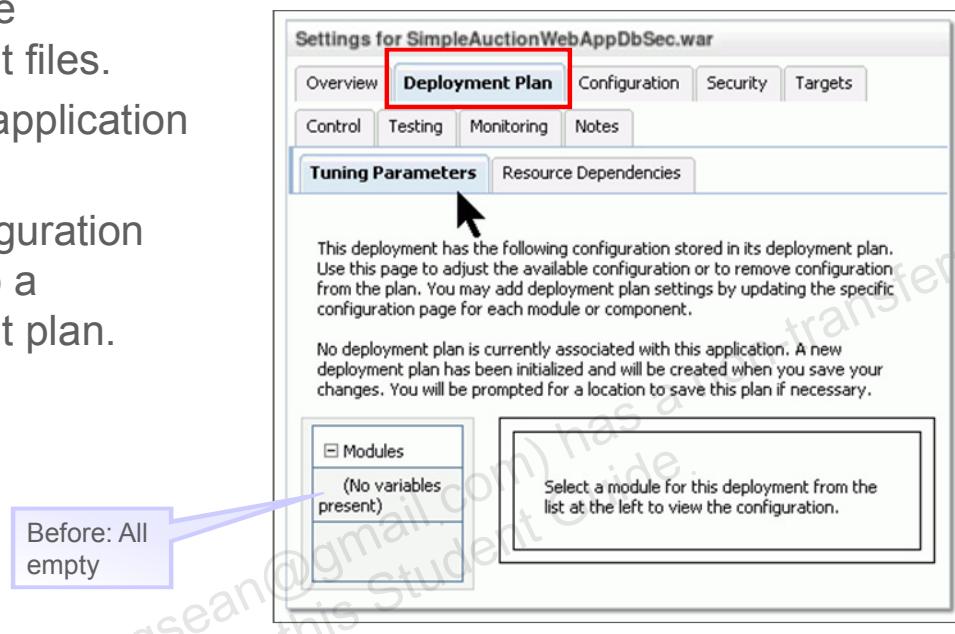
To create a deployment plan for a deployed application that does not already have a deployment plan, make a configuration change to the deployed application using the administration console. When you make a persisted configuration change to a deployed application that does not have an existing deployment plan, the console automatically creates a deployment plan for you and prompts you for the location in which to save it.

`weblogic.PlanGenerator` is a Java-based deployment configuration tool that is intended for developers who want to export portions of a deployment configuration into a deployment plan. This utility can generate a brand new plan or can append to an existing one. By default, `weblogic.PlanGenerator` writes an application's deployment plan to a file named `Plan.xml` in the application's root directory. The syntax for invoking `weblogic.PlanGenerator` is the following: `java weblogic.PlanGenerator [options] [application]`

# Using the Administration Console to Generate a Deployment Plan

You can generate a deployment plan with the administration console using the following steps:

1. Prepare the deployment files.
2. Install the application archive.
3. Save configuration changes to a deployment plan.



ORACLE®

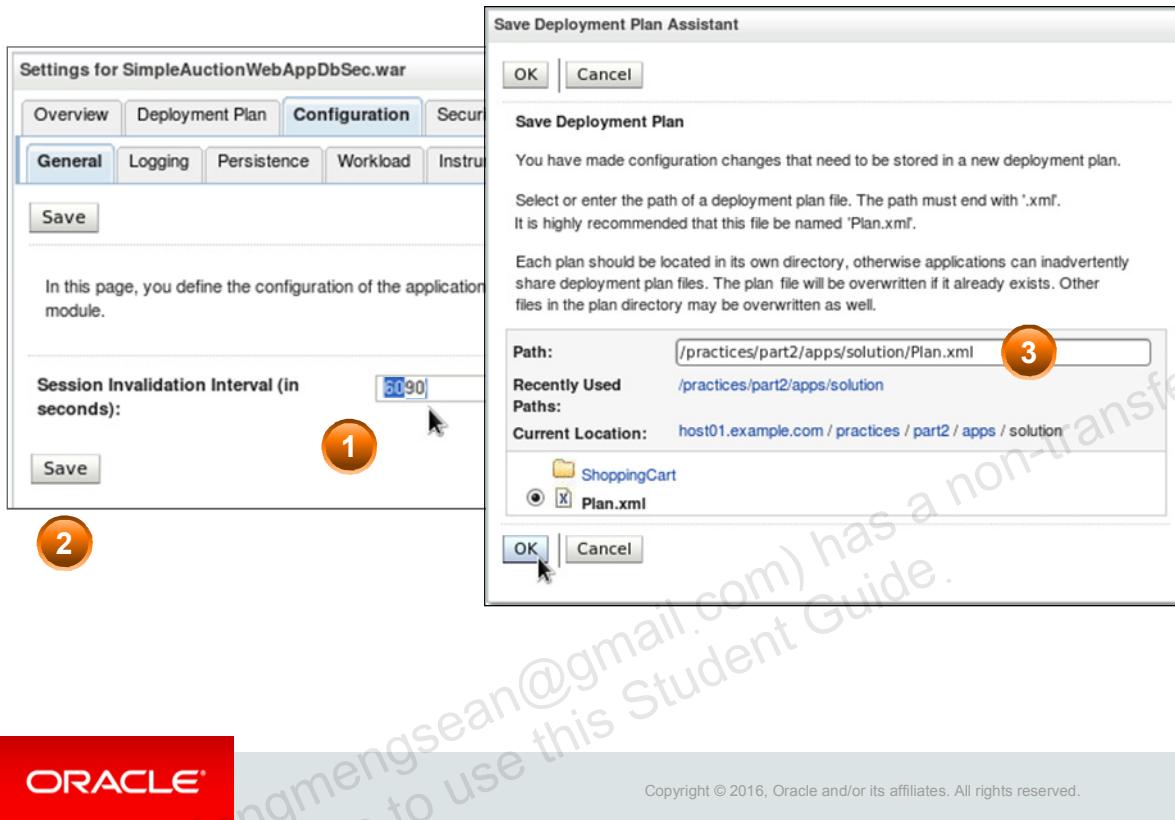
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The administration console automatically generates or updates the deployment plan.

**Note:** You can use the generated deployment plan to configure the application in subsequent deployments, or you can generate new versions of the deployment plan by repeatedly editing and saving the deployment properties.

The administration console provides greater flexibility than `weblogic.PlanGenerator` because it allows you to interactively add or edit individual deployment descriptor properties in the plan, rather than export entire categories of descriptor properties.

## Modifying and Saving Data to Create a New Plan



The application was originally deployed with various parameters (for example, a Session Invalidation Interval of 60 seconds).

1. Select an installed application that you want to modify, and make a change of the Session Invalidation Interval setting from 60 to 90.
2. When you click Save, the system prompts you for a new or existing deployment plan into which to save this.
3. There was no deployment plan originally. So it creates a new one called `Plan.xml`.

## New Deployment Plan Shows Changed Values

This deployment has the following configuration stored in its deployment plan. Use this page to adjust the available configuration or to remove configuration from the plan. You may add deployment plan settings by updating the specific configuration page for each module or component.

Setting	Deployment Plan Value	Descriptor Value
invalidationIntervalSecs	90	60

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This is a partial snippet of what was created. Notice the session invalidation in particular.

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan . . .>
  <application-name>SimpleAuctionWebAppDbSec.war</application-name>
  <variable-definition>
    <variable>
      <name>SessionDescriptor_invalidationIntervalSecs</name>
      <value>90</value>
    </variable>
  </variable-definition>
  <module-override>
    <module-name>SimpleAuctionWebAppDbSec.war</module-name>
    <module-type>war</module-type>
    <module-descriptor external="false">
      <root-element>weblogic-web-app</root-element>
      <uri>WEB-INF/weblogic.xml</uri>
      <variable-assignment>
        <name>SessionDescriptor_invalidationIntervalSecs</name>
        :
      </variable-assignment>
    </module-descriptor>
  </module-override>
</deployment-plan>
```

# Using an Existing Deployment Plan to Configure an Application

1. Prepare the application.
2. Place the existing deployment plan in the `plan` subdirectory of the application root.
  - This is optional. The `plan.xml` file may be located anywhere.
3. Install the application.
  - The administration console validates the deployment plan configuration against the target servers and clusters that are selected during the installation.
4. Use the administration console, `weblogic.Deployer`, or WLST to identify both the application and the plan to use for deployment.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

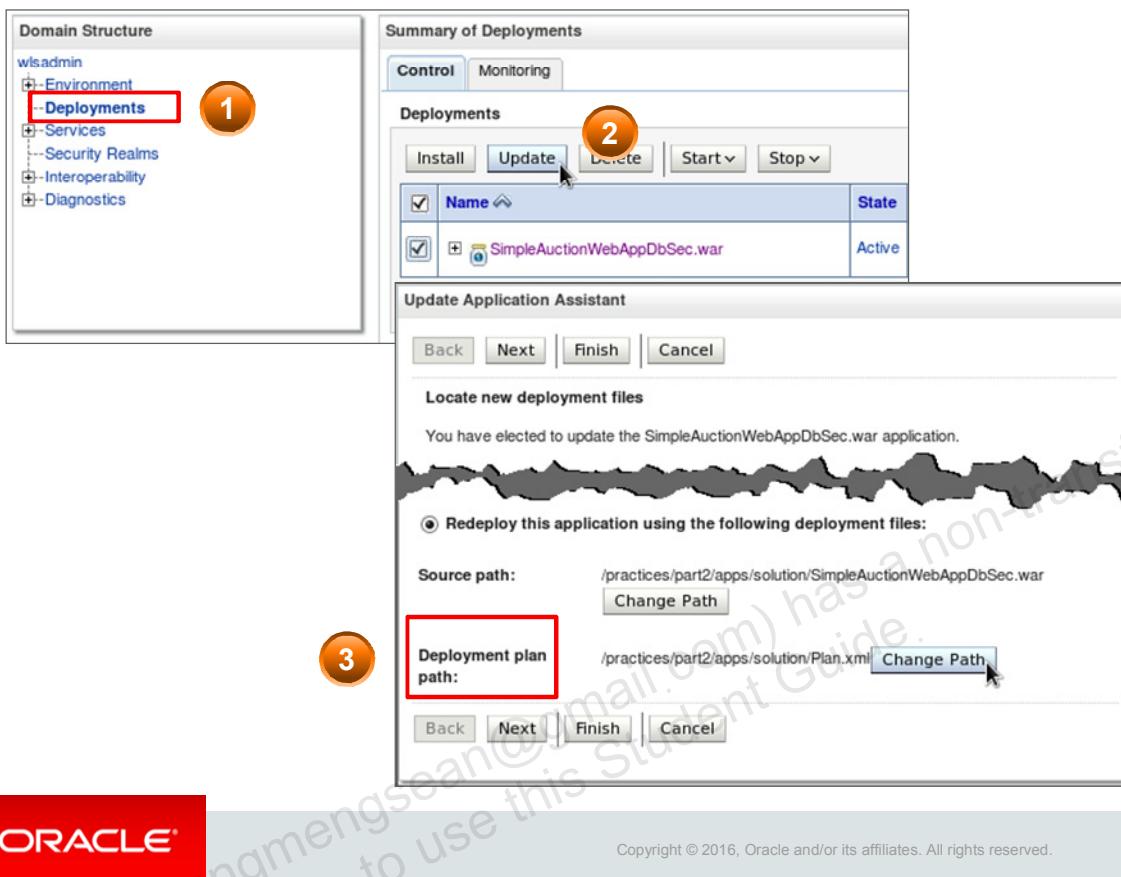
The applications that you receive for deployment may come with varying levels of configuration information. If you have an existing deployment plan for an application, simply prepare the application and place the deployment plan in the `plan` subdirectory of the application root. Then install the application. The administration console automatically uses a deployment plan named `plan.xml` in the `/plan` subdirectory of an application root directory if one is available. If multiple plans are available for your application, they are placed in their own `/plan` subdirectories (for example, `/plan1` and `/plan2`), and the administration console cannot identify them. Therefore, `config.xml` must specify the plan that you want to use.

After you install a new application and an existing deployment plan, the administration console validates the deployment plan configuration against the target servers and clusters that were selected during installation. If the deployment plan contains empty (null) variables, or if any values configured in the deployment plan are not valid for the target server instances, you must override the deployment plan before you deploy the application. You can also configure tuning parameters to better suit the target environment in which you are deploying the application. The changes you make to the application's configuration are saved to a new deployment plan.

If you have a valid deployment plan that fully configures an application for the environment in which you are deploying, you can use either the administration console, `weblogic.Deployer`, or WLST to deploy an application with a deployment plan to use for deployment.

**Note:** A deployment plan that you use with the `weblogic.Deployer` utility must be complete and valid for your target servers. `weblogic.PlanGenerator` does not allow you to set or override individual deployment properties when it creates a plan.

# Using an Existing Deployment Plan



You can use the administration console to specify a deployment plan for your application.

1. In the left pane, click Deployments.
2. In the right pane, select the check box next to the application for which you want to specify a deployment plan. Click Update.
3. Click Change Path next to “Deployment plan path” to browse to the desired deployment plan. Click Next, and then click Finish.

## WLST createPlan

Example of using the WLST createPlan option:

```
> deploy(appName='MyApp',
    path='/apps/MyApp.ear',
    targets='cluster1',
    createPlan='true')
```

WLST Command Line



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This example shows WLST deploying an application and creating an initial deployment plan for the application.

## weblogic.PlanGenerator

- Enables you to generate basic WebLogic configuration extensions for applications that have only Java EE deployment descriptors
- Enables you to:
  - Create an initial plan
  - Create a new plan based on an existing plan
  - Control which components are exported to a plan



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Using Plan Generator

The following are examples of using `weblogic.PlanGenerator`:

- Creating an initial deployment plan in an application's root directory:

```
java weblogic.PlanGenerator -root /appRelease/MyApplication
```

- Creating a new deployment plan based on an existing plan:

```
java weblogic.PlanGenerator  
-useplan /plans/MyApplication_template.xml  
-root /appRelease/MyApplication
```

- Controlling which components are exported to a deployment plan:

```
java weblogic.PlanGenerator -root /appRelease/MyApplication -all
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

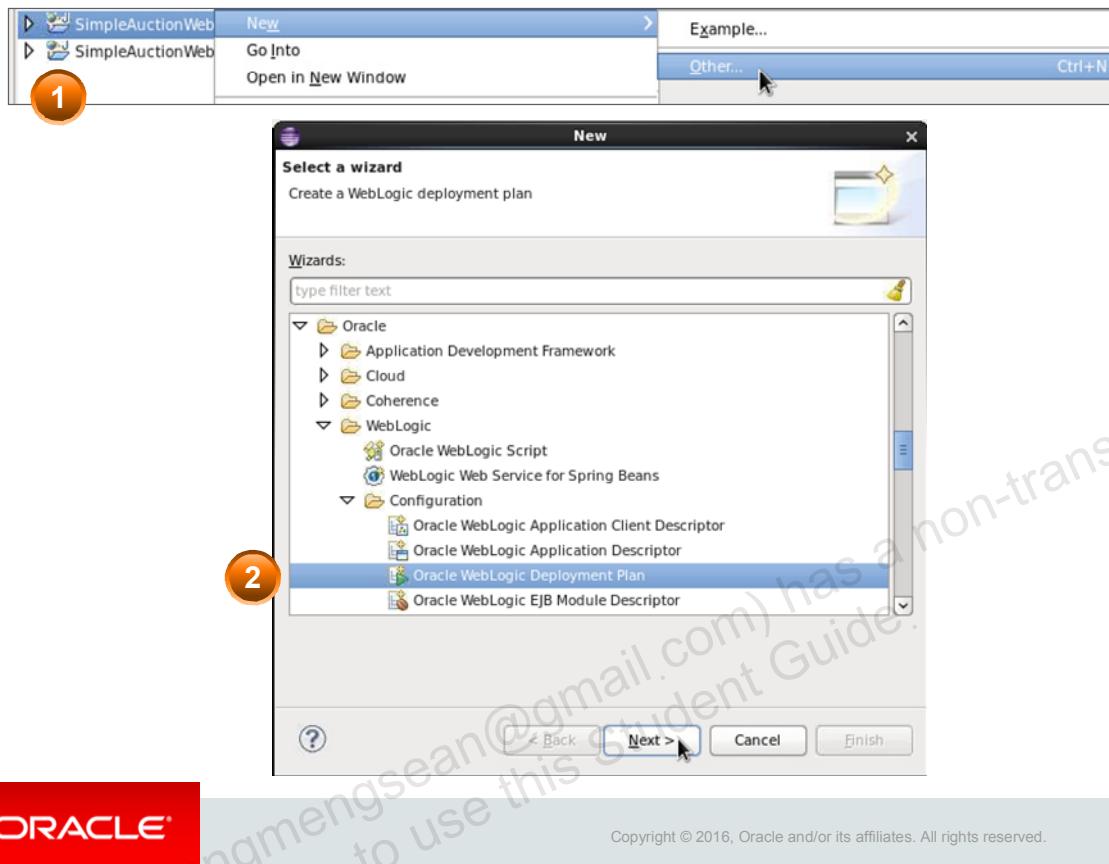
The following are examples of using `weblogic.PlanGenerator`:

- **Creating an initial deployment plan in an application's root directory:** In this example, the `plan.xml` file is automatically stored in `/appRelease/MyApplication/plan`.
- **Creating a new deployment plan based on an existing plan:** This command uses the `/plans/MyApplication_template.xml` plan as the basis for creating a new plan in `/appRelease/MyApplication/plan`.
- **Controlling the components that are exported to a deployment plan:** This command exports all configurable properties to null variables in a template deployment plan.

You can use the `-all`, `-configurables`, `-dependencies`, `-declarations`, `-dynamics`, and `-none` options to specify the deployment descriptor components that are exported to a template deployment plan.

For information about `weblogic.PlanGenerator`, review the `weblogic.PlanGenerator Command-Line Reference` section of the *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server* documentation.

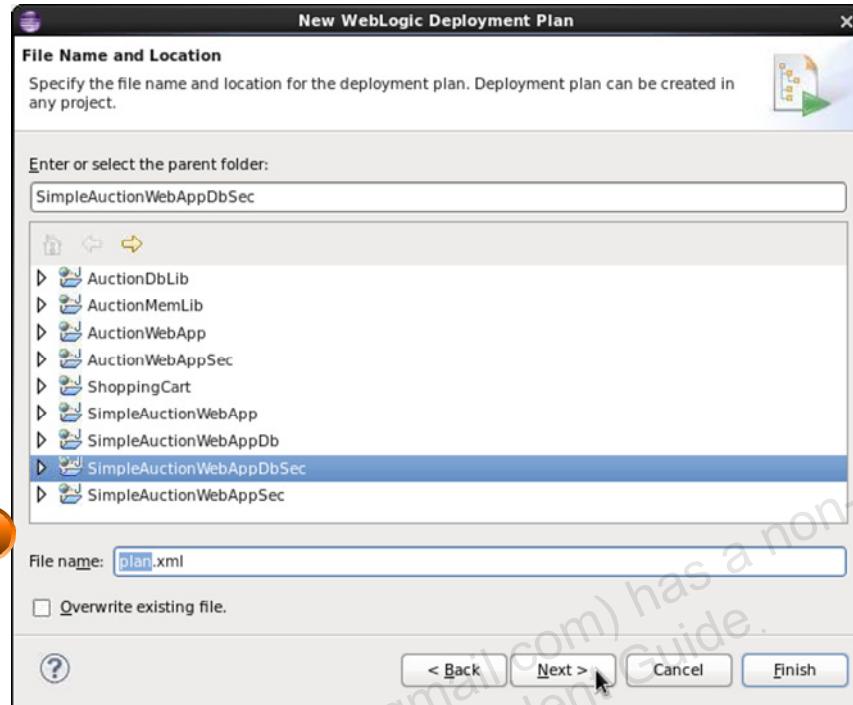
# Oracle Enterprise Pack for Eclipse



Perform the following steps to create a deployment plan for an application in Eclipse:

1. Right-click your application project and select New > Other to open the *Select a wizard* dialog box.
2. In the dialog box, expand Oracle > WebLogic > Configuration, select *Oracle WebLogic Deployment Plan*, and click Next.

# Oracle Enterprise Pack for Eclipse

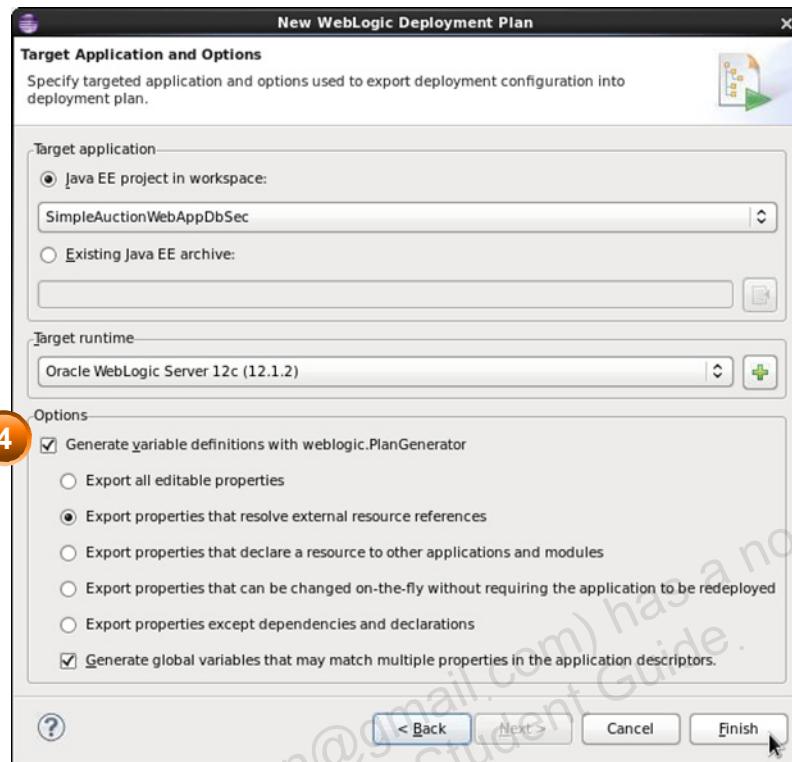


3. On the File Name and Location page, select the parent folder location and the file name to use for the plan file, and click Next.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle Enterprise Pack for Eclipse

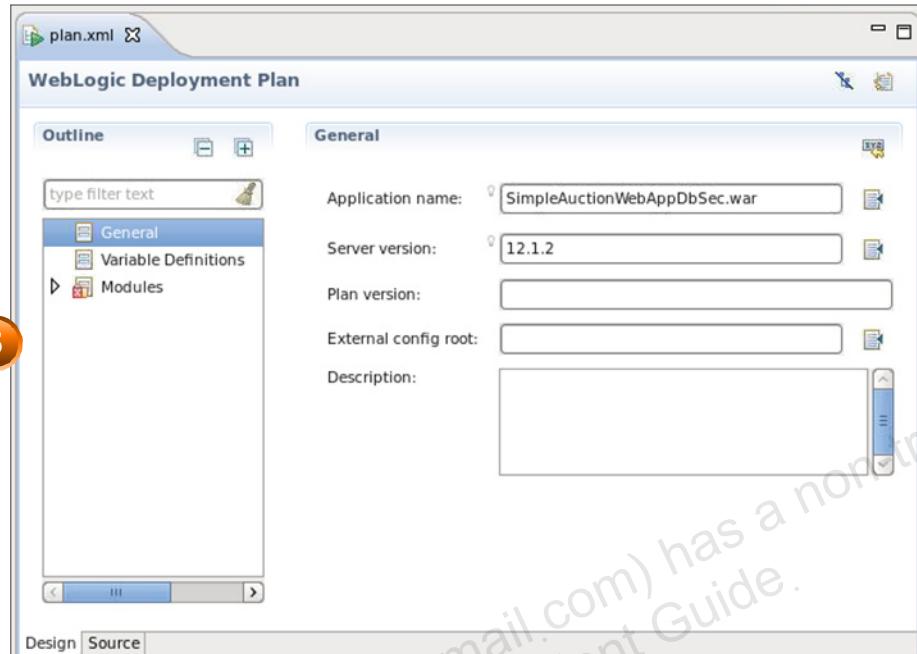


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

4. On the Target Application and Options page, make your desired selections to control how your deployment plan is created. In the example on this slide, you are using `weblogic.PlanGenerator` to create your plan and instruct it to only create plan entries for resources that are part of your application.

# Oracle Enterprise Pack for Eclipse



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

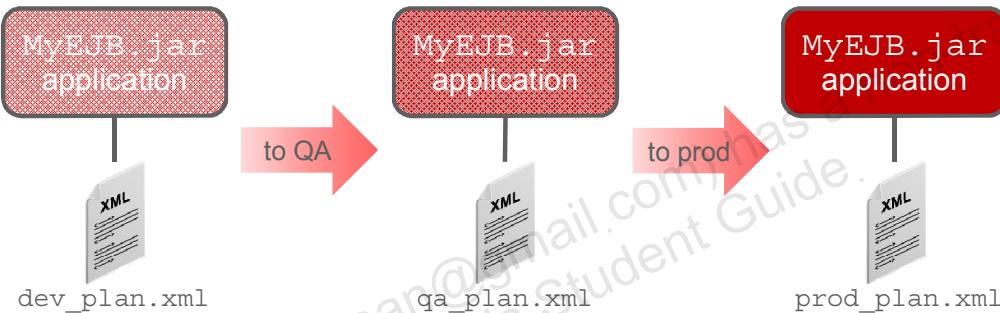
5. The plan is created and displayed in Eclipse for editing.

Remember that this is a task that developers are more likely to perform. These steps are only mentioned here to show you how developers may create the plans that are subsequently included with the applications that are passed on to administrators for deployment. This process is very similar for JDeveloper as well.

# Managing Deployment Plans

Because deployment plans are different for each target environment and stored externally from application archives:

- Plans should be managed using a source code control system (SCCS). SCCS provides built-in:
  - File versioning
  - Change management and file before and after comparisons
- Each plan should be stored separately:



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Deployment plans are different for each environment during the full development life cycle of an application. Each environment that an application is deployed to requires its own unique deployment plan unless all of the settings are going to remain identical. Typically, applications are packaged up and deployed as archives and deployment plans are used as an externally available configuration tool to conveniently modify the application's settings. Because of this, deployment plans are not part of the application itself and must be managed separately.

## Summary

In this lesson, you should have learned how to:

- Configure a server's staging mode
- Create and use deployment plans



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 17-1 Overview: Creating and Using a Deployment Plan

This practice covers the following topics:

- Testing an application in an environment that uses a data source with the JNDI name `jdbc/AuctionDB`
- Generating a deployment plan using `weblogic.PlanGenerator`
- Modifying the deployment plan to use with the application so that when deployed it uses a new data source with the JNDI name `jdbc/AuctionDBTest`
- Modifying the `AuctionDB` data source to use the new JNDI name
- Updating the application to use the deployment plan
- Testing the application



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Production Redeployment

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Objectives

After completing this lesson, you should be able to deploy versioned applications (production redeployment).



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda

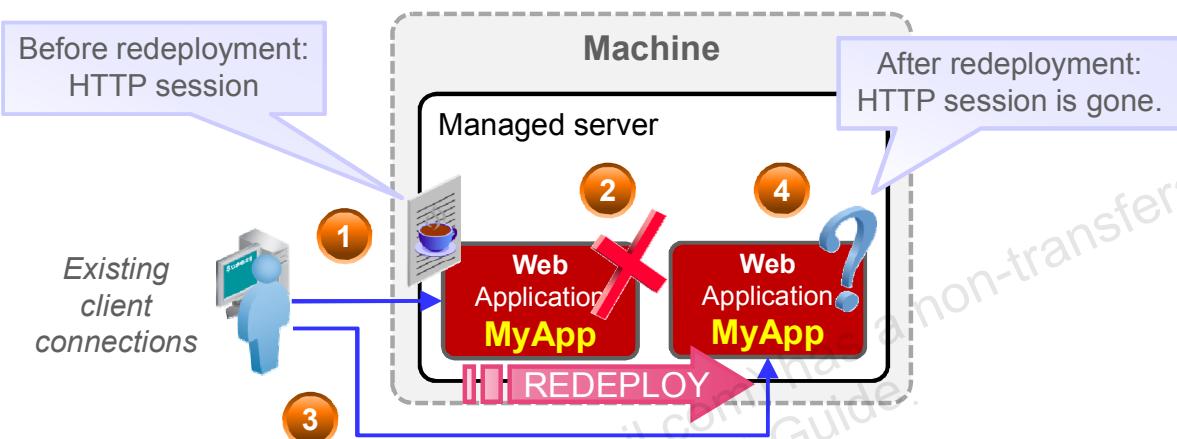
- HTTP Sessions and Redeployment
- Redeployment Strategies
- Production Redeployment
- Application Versioning
- Requirements and Restrictions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# HTTP Sessions and Redeployment

HTTP sessions maintain a client's state in WebLogic:



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

WebLogic maintains the user state for clients that use web applications in HTTP session objects. When the associated web application is redeployed, the undeployment aspect of the process effectively eliminates the HTTP sessions associated with that deployment.

1. A client using a web browser is using the MyApp web application and has an associated HTTP session on the server.
2. The application is redeployed, which involves undeploying the original MyApp application and replacing it with the updated MyApp application. The associated HTTP session for the existing client is lost.
3. The client browser makes another request to the application, which is routed to the newly deployed MyApp application. However, there is no HTTP session for this client and all session state is lost.
4. Depending on the situation, the client may have to start his or her session over to reestablish state with a new HTTP session that is associated with the newly deployed MyApp web application.

# Agenda

- HTTP Sessions and Redeployment
- Redeployment Strategies
- Production Redeployment
- Application Versioning
- Requirements and Restrictions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Redeployment Strategies

WebLogic provides several different redeployment techniques for applications:

Strategy	Description	When to Use
In-place	Completely replaces the existing application with the new application	Upgrading applications that can tolerate client interruptions
Partial redeploy for static files	Immediately replaces files such as HTML, JSP, and images	Updating static web files that do not affect application clients
Partial redeploy of Java EE modules	Completely replaces existing modules within an enterprise application (EAR)	Replacing a component of an enterprise application that can tolerate client interruptions
Production redeployment (side-by-side)	Deploys a new version of an application alongside an existing version	Upgrading applications that cannot tolerate client interruptions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Note

Production redeployment is also called side-by-side deployment. This lesson focuses mainly on the last redeployment strategy; production redeployment.

# Agenda

- HTTP Sessions and Redeployment
- Redeployment Strategies
- Production Redeployment
  - Application Availability
  - What Is Production Redeployment?
  - Advantages of Production Redeployment
  - Production Redeployment Process
  - Application Retirement
  - Distributing Versioned Applications
  - Administration Mode
  - Rolling Back to a Previous Version
- Application Versioning
- Requirements and Restrictions

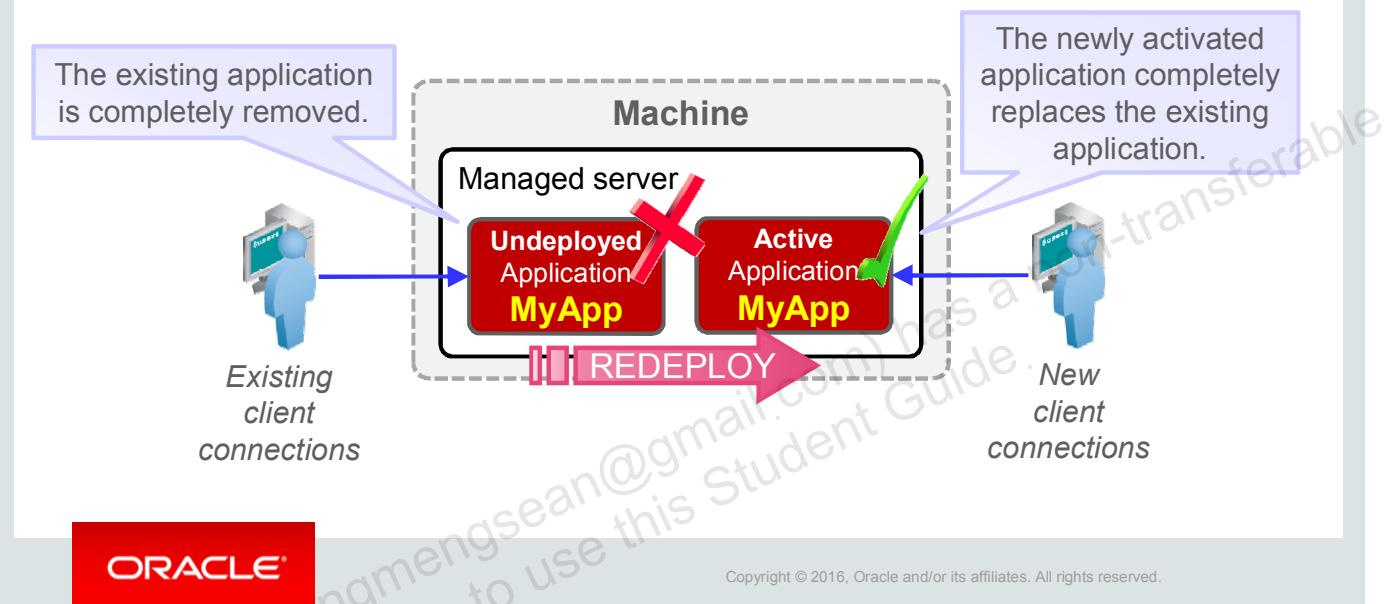


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Application Availability

What happens when you need to update an existing application deployment?

- Default redeployment occurs in place.
- Redeployment disrupts existing clients.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

By default, when an application is redeployed in place:

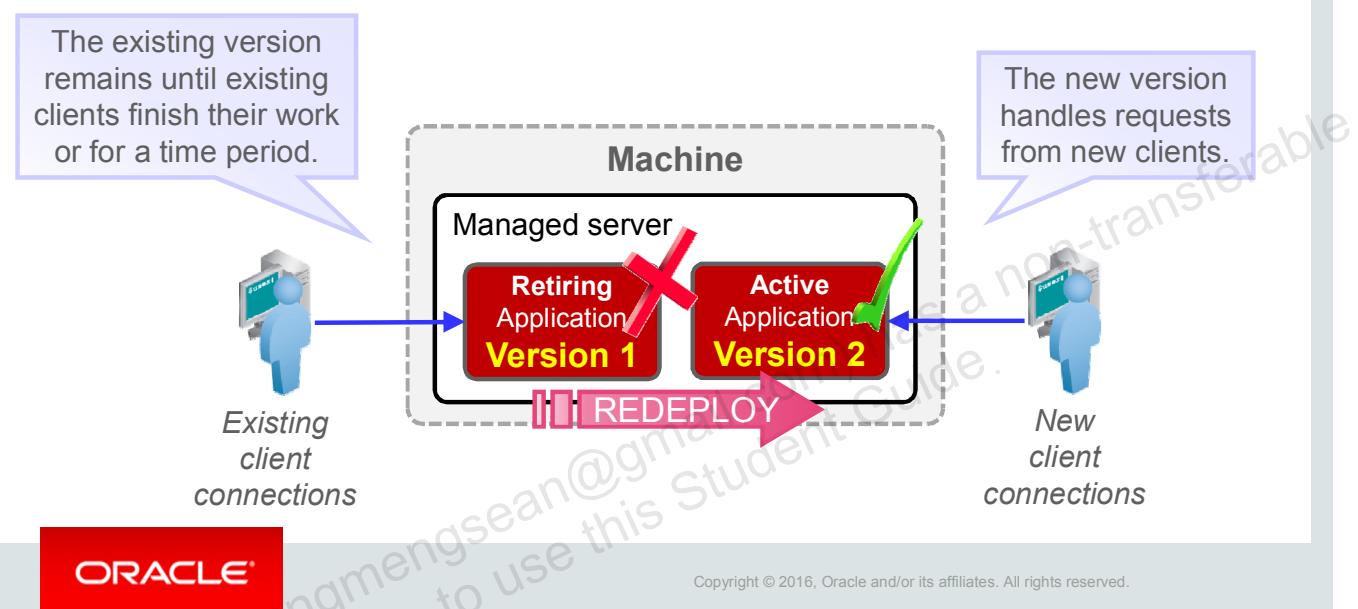
- It is unavailable to clients for a brief time
- Existing clients lose any conversational state

In a production environment, deployed applications frequently require  $24 \times 7$  availability to provide uninterrupted services to customers and internal clients. In these cases, scheduling maintenance down time to replace applications is not desirable.

WebLogic uses an in-place redeployment scheme by default. In-place redeployment immediately replaces a running application's deployment files with the updated deployment files. In contrast to production redeployment, in-place redeployment of an application or a stand-alone Java EE module does not guarantee uninterrupted service to the application's clients. This is because WebLogic immediately removes the running `class loader` for the application and replaces it with a new `class loader` that loads the updated application class files.

# What Is Production Redeployment?

Production redeployment is the ability to deploy multiple versions of the same application at the same time to avoid disrupting existing or new users.



Production redeployment enables an administrator to redeploy a new version of an application in a production environment without stopping the deployed application, or interrupting the application's availability to clients. Production redeployment works by deploying a new version of an updated application alongside an older version of the same application. WebLogic automatically manages client connections so that only new client requests are directed to the new version. Clients that are already connected to the application during the redeployment continue to use the older, retiring version of the application until they complete their work.

Production redeployment is supported primarily for applications with a web application entry point (HTTP clients). WebLogic can automatically manage the HTTP client entry points to isolate connections to the newer and older application versions. That is, production redeployment is supported for stand-alone web application modules and for enterprise applications that are accessed via an embedded web application module.

Production redeployment supports only HTTP clients and RMI clients. Your development and design team must ensure that applications using production redeployment are not accessed by an unsupported client. WebLogic does not detect when unsupported clients access the application and does not preserve unsupported client connections during production redeployment.

Enterprise applications can contain any of the supported Java EE module types. Enterprise applications can also include application-scoped JMS and JDBC modules.

If an enterprise application includes a JCA resource adapter module, the module:

- Must be JCA 1.5 compliant
- Must implement the `weblogic.connector.extensions.Suspendable` interface
- Must be used in an application-scoped manner, having `enable-access-outside-app` set to `false` (the default value)

Before the resource adapters in the newer version of the Enterprise Archive (EAR) are deployed, the resource adapters in the older application version receive a callback. WebLogic then deploys the newer application version and retires the entire older version of the EAR.

When you are redeploying a new version of an application, the following features cannot change:

- Deployment targets
- Security model
- Persistent store settings

## Advantages of Production Redeployment

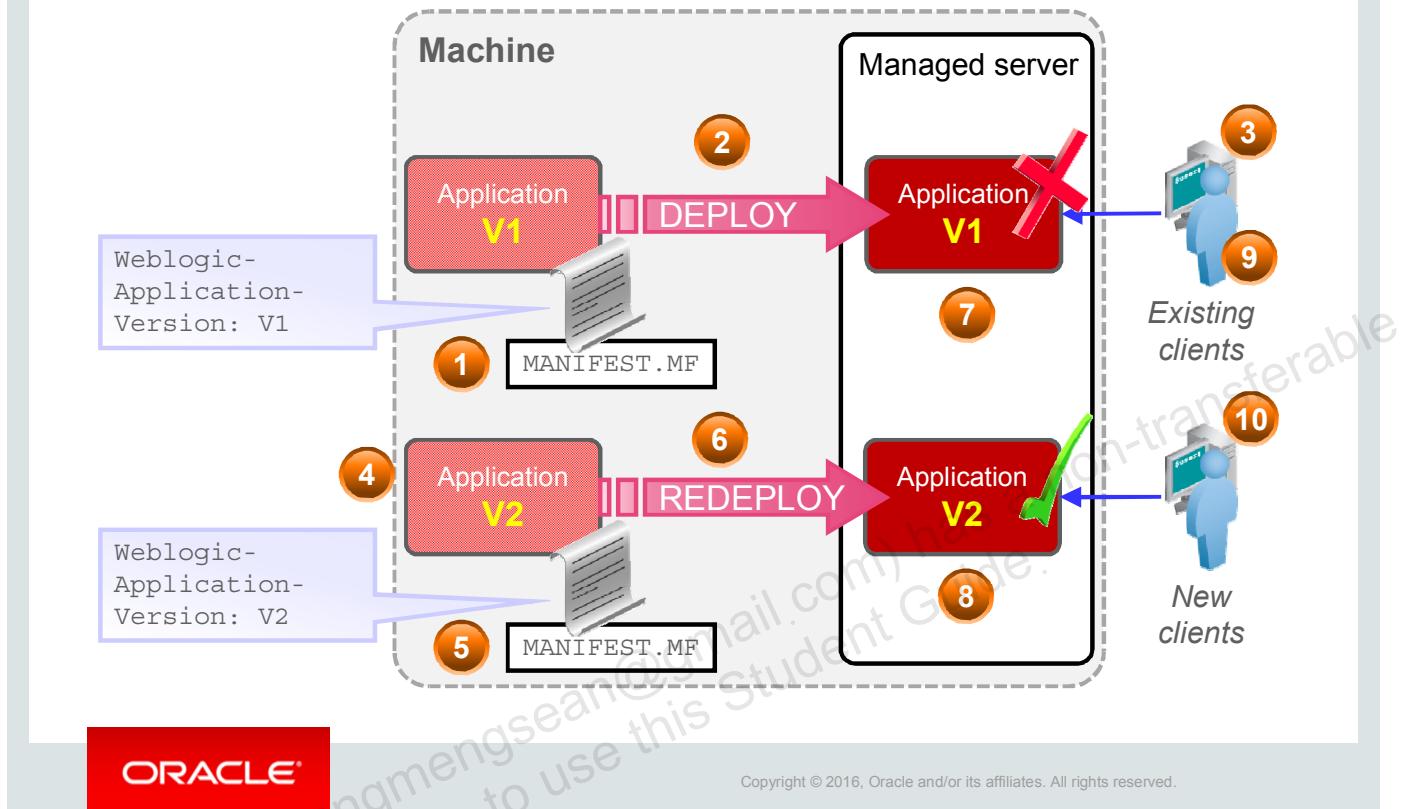
Completely avoids:

- Scheduling application down time
- Setting up redundant servers to host new application versions
- Managing client access to multiple application versions manually
- Retiring older versions of an application manually



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Production Redeployment Process



The production redeployment process includes the following:

1. An administrator configures an application with a current version of V1 in the application's manifest file.
2. The administrator deploys the application normally. WebLogic recognizes the version information in the manifest file and deploys the application with that version data.
3. Clients begin to use the V1 version of the application.
4. An update to the application has been developed and is ready to replace the older version in production.
5. The administrator updates the application's manifest file to a new version of V2.
6. The administrator deploys the application again. Because the application is already deployed, WebLogic performs a redeployment. During this process, WebLogic recognizes the updated application's version information in the manifest file. WebLogic recognizes that both the existing and new application deployments have version data associated with them.
7. As a result, WebLogic changes the state of the original application to *stop Running*, but does not actually stop it from running unless it is directed to do so.

8. WebLogic deploys the new application along with its version information and changes its state to *Active*.
9. Existing clients that were using V1 continue to have WebLogic route their requests to the V1 version of the application.
10. New clients have WebLogic route their requests to the V2 version of the application.

# Application Retirement

Existing application versions are retired depending upon several factors:

Retirement Trigger	Description
<b>Graceful client transition</b>	Waits for all existing client sessions to end, and then automatically retires the old version
<b>Timeout</b>	Keeps the existing version running for a configured timeout period, and then automatically retires the old version
<b>Forced</b>	An administrator can force work to stop and cause the original application version to retire. This is the equivalent of doing an in-place deployment.

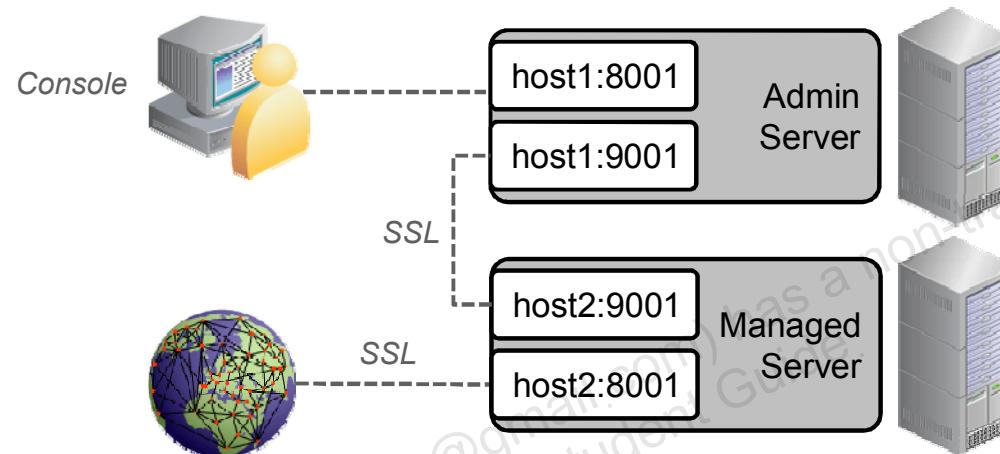


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

During a graceful retirement for an application deployed on a cluster, each cluster member maintains its own set of HTTP sessions. Client activity is varied across cluster members so some clusters may retire the old version of the application before others. In the event of a failover scenario, the client fails over to the server where its secondary HTTP session resides. If the existing version of the application is still active on that server, the request works normally. If the existing version has already retired on that server, then failover fails and the client must reestablish its session with the server.

## Review: Administration Channel

- Use a dedicated address and/or port for administrative traffic.
- Disable client access points during server maintenance or troubleshooting.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

While maintaining or troubleshooting a production server, it is often desirable to disable all incoming application requests. However, a server's default network configuration implies that all traffic runs on the same port. Therefore, if the port were closed, all remote administration tools, such as the console or WLST, would also not be able to connect to the server.

WebLogic Server supports a domain in which all servers use a separate SSL port that accepts only administration requests. The use of dedicated administration ports enables you to:

- **Start a server in standby state:** This enables you to administer a server while its other network connections are unavailable to accept client connections.
- **Separate administration traffic from application traffic in your domain:** In production environments, separating the two forms of traffic ensures that critical administration operations (such as starting and stopping servers, changing a server's configuration, and deploying applications) do not compete with high-volume application traffic on the same network connection.
- **Administer a deadlocked server instance:** If you do not configure an administration port, administrative commands such as `THREAD_DUMP` and `SHUTDOWN` will not work on deadlocked server instances.

## Administration Mode

You can deploy applications in administration mode:

- Applications are not available to clients.
- This mode requires a configured administration channel.
- Administrators can test the application before clients use it.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Distributing a Versioned Application

Distributing is an alternative to deploying an application.

**Distributing** a new version of the application makes it available for testing before being released for general consumption.

**Redeploying** a new version of an application places the application immediately into use and makes it available to new client requests.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

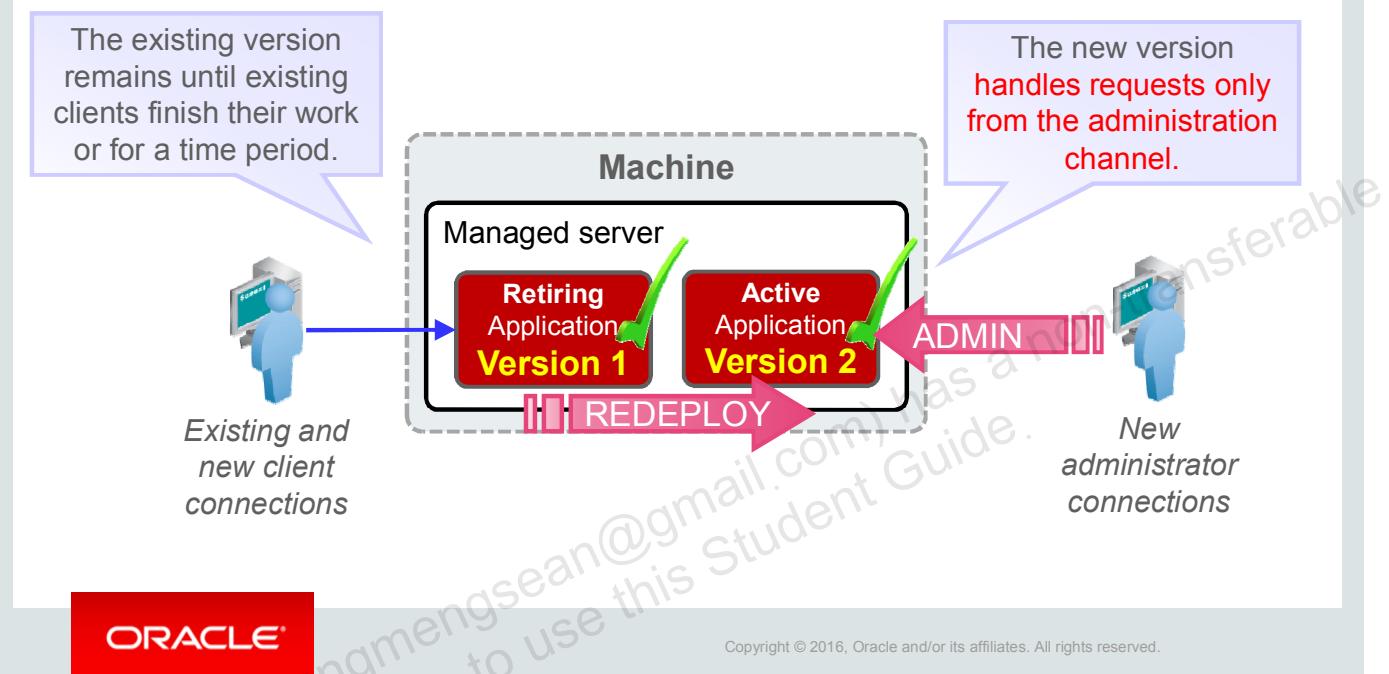
Production redeployment can be used with the `-distribute` command to prepare a new version of an application for deployment. For more information, refer to *Distributing a New Version of a Production Application*.

Distributing an application prepares it for deployment by copying its files to all target servers and validating the files.

You can start a distributed application in Administration mode. Access to the application is then restricted to a configured administration channel.

## Deploying in Administration Mode

A new version of an application deployed in administration mode for testing:

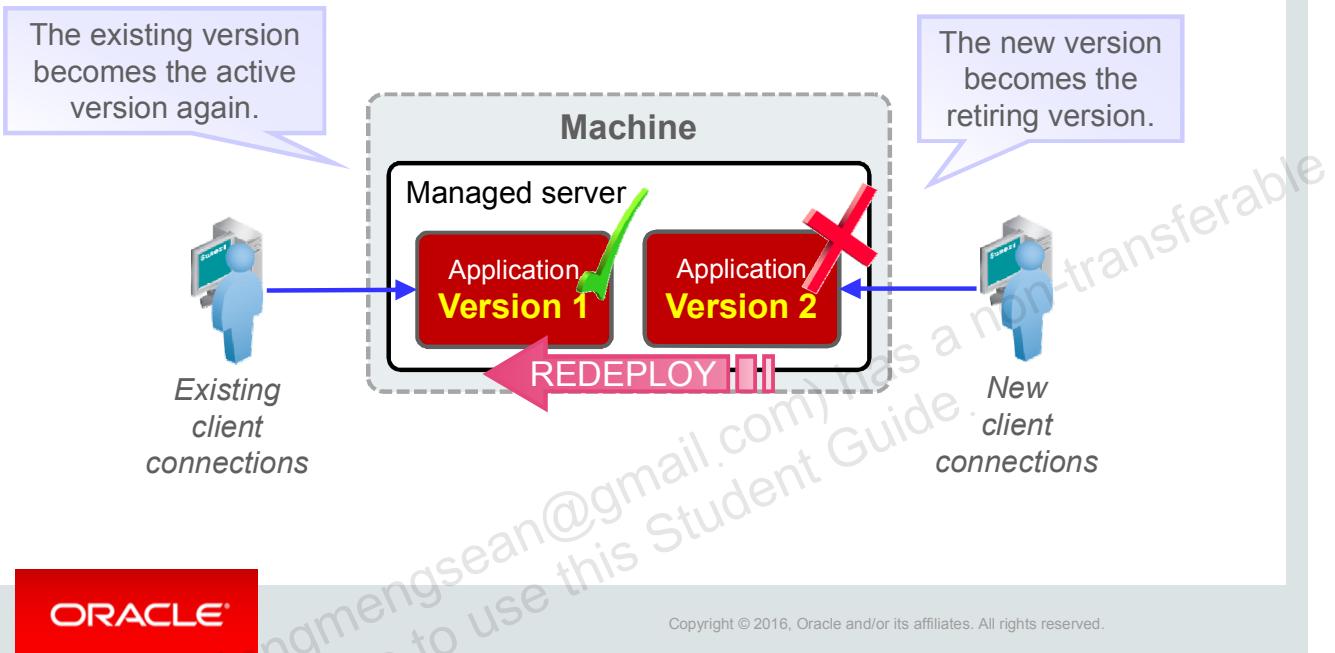


WebLogic prepares the new application version, which is deployed in administration mode and made available via a configured administration channel, for deployment. However, the older version of the application is not automatically retired by WebLogic when the new version of the application is distributed and deployed in administration mode. The older version of the application remains active to process both new and existing client requests.

The new application version can either be undeployed or started after it has been completely tested via an administration channel. WebLogic routes new client connections to the updated application after starting the application and begins retiring the older application version.

## Rolling Back to the Previous Version

You can roll back a versioned application to the previous version.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Note:** Rolling back is technically not always an actual rollback procedure. In some cases, it is merely a new production redeployment task whereby the version and files of the new version are really the same as the original existing version. WebLogic has no knowledge of whether or not the version information is sequential or has any meaning. It only requires that the version strings are different.

Rolling an application deployment back to a previous version is possible regardless of the current state of the side-by-side deployment:

- If the new version is in administration mode, the administrator can stop the new version so the existing version remains in place.
- If the new version is active and the existing version is retiring, the administrator can redeploy the existing version, which will cause the two versions to switch places. The new version changes to a retiring state and the existing version becomes active. The administrator can also force work to stop for the retiring version to ensure that clients revert to using the original version quickly.
- If the new version is active and the existing version is retired, the administrator can redeploy the existing version, which will again cause the two versions to switch places.

## Quiz



How many versions of an application can be deployed simultaneously in WebLogic?

- a. 1
- b. 2
- c. 3
- d. 4

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

## Quiz



Which types of clients are supported by production redeployment?

- a. HTTP
- b. RMI
- c. Web service
- d. All of the above

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer:** d

# Agenda

- HTTP Sessions and Redeployment
- Redeployment Strategies
- Production Redeployment
- Application Versioning
  - Configuring Application Versions
  - Redeployment Command Examples
- Requirements and Restrictions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Redeployment Process: Overview

1. Verify that only one version of the application is currently deployed.
2. Verify the MANIFEST.MF files to ensure that both applications have different versions.
3. Copy the new version into a suitable directory.
4. Redeploy the new application version and specify the updated deployment files.
5. Verify that both versions are deployed and that new requests are being sent to the new version.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows a summary of the tasks required to perform production redeployment on WebLogic.

**Note:** WebLogic supports deploying only two versions of an application at a time. You must undeploy or retire a version of an application before deploying a subsequent version.

# Configuring Application Deployment Versioning

You configure application version information by using the application's manifest file or specifying options during deployment:

```
Manifest-Version: 1.0  
Weblogic-Application-Version: 1.0beta
```

This string can be any value.

MANIFEST.MF

```
$ java weblogic.Deployer -adminurl http://host01.example.com:7001  
-user weblogic -password Welcome1 -deploy  
-name myApp  
-source /prod/deployments/myApp/beta1.0/MyApp.war  
-targets myCluster -appversion 1.0beta
```

Use deploy for the initial deployment.

Command Line

Specify the version here if it is not configured in the manifest file.

**Note:** The best practice is to include version data in the manifest file.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You should include version information in the application's manifest file because it is easier to track and maintain the versions of your applications. Otherwise, you could deploy an older version of an application accidentally.

# Configuring Application Deployment Versioning

Specifying options using the administration console:

- The administration console does not provide a way to set an application version. It displays only whatever archive version is configured in the manifest file.

WLST:

```
> deploy('myApp',
          '/prod/deployments/myApp/beta1.0/MyApp.war',
          'myCluster',
          versionIdentifier='1.0beta')
```

Command Line

Specify version here  
or in the manifest file.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Deploying a New Version of an Application

Configure the manifest file of the updated application with the new version and redeploy the application:

```
Manifest-Version: 1.0  
Weblogic-Application-Version: 1.0
```

WebLogic recognizes that this string is different.

MANIFEST.MF

```
$ java weblogic.Deployer -adminurl http://host01.example.com:7001  
-user weblogic -password Welcome1 -redeploy  
-name myApp  
-source /prod/deployments/myApp/GA1.0/MyApp.war  
-targets myCluster  
-retiretimeout 200
```

Use redeploy when deploying a new version.

Command Line

Optional timeout seconds before retiring application

The application name is the same, deployment files are different, and deployment targets are the same.

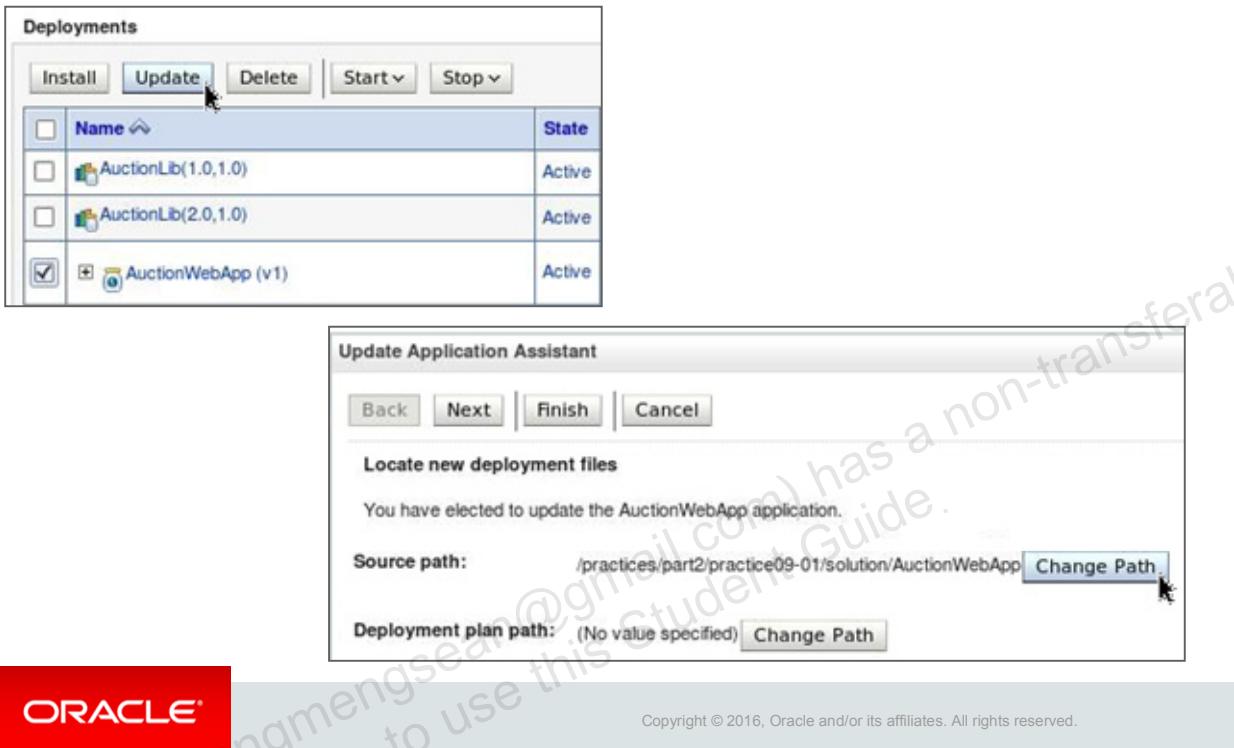
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If a timeout is not specified during production redeployment, the default behavior is to gracefully retire the existing application version after all client work has completed.

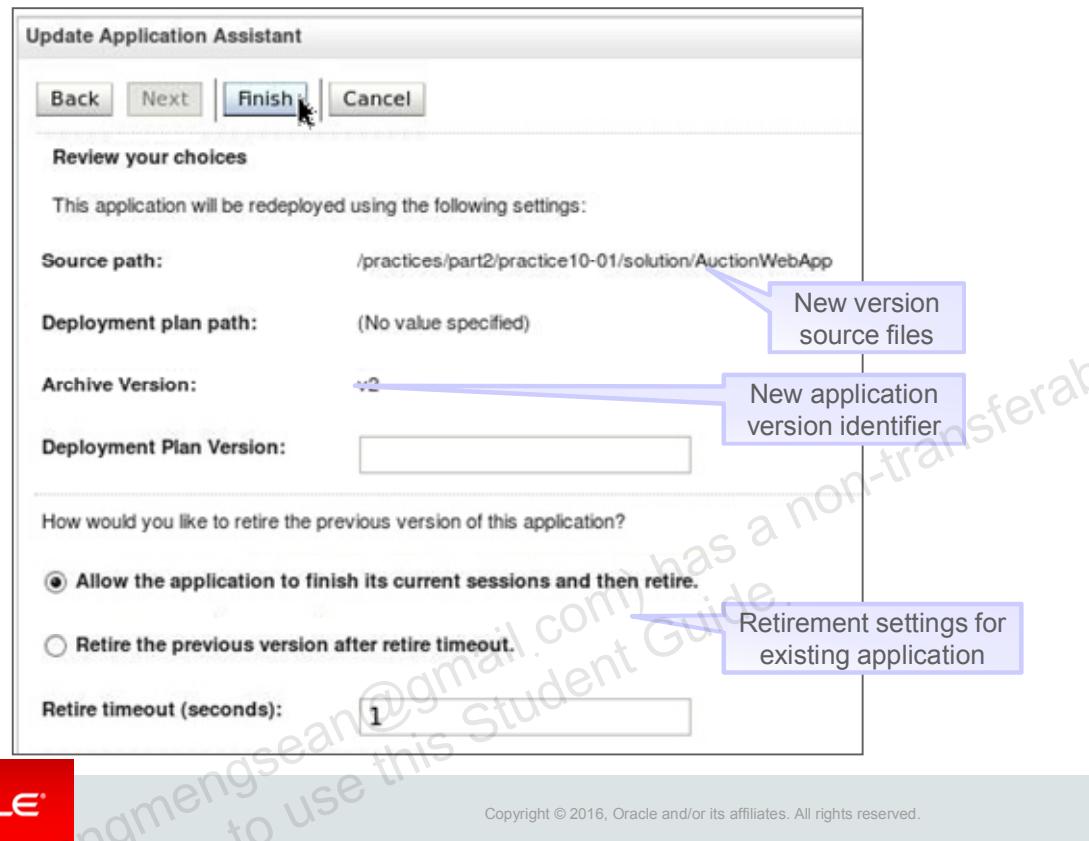
# Deploying a New Version of an Application

Administration console:



This slide and the following slide show an example of redeploying an application using the administration console. The first image shows that the existing application version is selected and the Update button is clicked to update the deployment. The second image shows that the Change Path button is clicked to choose the updated source files for the new application version. Next is clicked to view the application's settings, including its version information.

# Deploying a New Version of an Application



The image in the slide shows the version information associated with the new version of the application before deployment. Clicking Finish completes the process.

# Deploying a New Version of an Application

WLST:

```
> redeploy('myApp',
  '',
  appPath='/prod/deployments/myApp/GA1.0/MyApp.war',
  versionIdentifier='1.0GA',
  retireGracefully='true')
```

Command Line



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Note

You can also specify `adminMode='true'` as an option to cause the redeployment to start in Admin mode.

## Distributing and Starting a Versioned Application in Administration Mode

Starting a versioned application in administration mode involves distributing the application, and then starting it with the `-adminmode` option:

```
$ java weblogic.Deployer -adminurl http://host01.example.com:7001  
-user weblogic -password Welcome1 -distribute  
-name myApp  
-source /prod/deployments/myApp/GA1.0/MyApp.war  
-targets myCluster
```

Use distribute instead of redeploy.

Command Line

```
$ java weblogic.Deployer -adminurl http://host01.example.com:7001  
-user weblogic -password Welcome1 -start -adminmode  
-name myApp
```

Command Line



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Distributing and Starting a Versioned Application in Administration Mode

Specifying options by using the administration console:

The screenshot shows the Oracle WebLogic Administration Console's Deployments view. In the top section, a context menu is open over the 'AuctionWebApp (v1)' entry, with the option 'Servicing only administration requests' highlighted. In the bottom section, a table lists five deployments: 'AuctionLib(1.0,1.0)', 'AuctionLib(2.0,1.0)', 'AuctionWebApp (v1)', and 'AuctionWebApp (v2)'. The 'AuctionWebApp (v2)' row is selected and has a red box drawn around it. The 'AuctionWebApp (v1)' row is also circled in red. The table includes columns for Name, State, and Health. The 'AuctionWebApp (v1)' state is 'Active' and health is 'OK'. The 'AuctionWebApp (v2)' state is 'Admin' and health is 'OK'. The Oracle logo is at the bottom left, and a copyright notice is at the bottom right.

Name	State	Health
AuctionLib(1.0,1.0)	Active	
AuctionLib(2.0,1.0)	Active	
AuctionWebApp (v1)	Active	OK
AuctionWebApp (v2)	Admin	OK

This slide shows using the administration console to start a distributed new version of an application in Admin mode. The first image shows the state of the two versions of the application before starting the new version in Admin mode. The second image shows that the existing version of the application is in Active mode and the new version is in Admin mode.

# Distributing and Starting a Versioned Application in Administration Mode

Distribute an application:

```
> distributeApplication('/prod/deployments/myApp/GA1.0/MyApp.war',  
    '',  
    'myCluster',  
    versionIdentifier='1.0GA')
```

WLST Command Prompt

Specifies the version of this application deployment

Start the application in administration mode:

```
> startApplication('myApp',  
    versionIdentifier='1.0GA',  
    adminMode='true',  
    retireGracefully='true')
```

WLST Command Prompt

Starts the application in Admin mode and tells WebLogic to retire gracefully



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Transitioning a Versioned Application from Administration Mode to Active

Making a versioned application running in administration mode available for client requests involves using the `-start` option again without specifying the `-adminmode` option:

```
$ java weblogic.Deployer -adminurl http://host01.example.com:7001  
-user weblogic -password Welcome1 -start  
-name myApp
```

Command Line



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Transitioning a Versioned Application from Administration Mode to Active

Administration console:

Name	Status	State	Health
AuctionLib(1.0,1.0)	Active		
AuctionLib(2.0,1.0)	Active		
AuctionWebApp (v1)	Active	OK	
AuctionWebApp (v2)	Admin	OK	

WLST:

```
> startApplication('myApp',
>                   versionIdentifier='1.0GA',
>                   retireGracefully='true')
```

Command Line

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows using the administration console and WLST to move an application from Admin mode to Active mode. The first image shows the state of the two versions of the application before starting the new version in Admin mode. The second image shows that the existing version of the application is in Retired mode and the new version is in Active mode.

## Rolling Back a Versioned Application to a Previous Version

To roll back to the retired application version, perform a second `-redeploy` command and specify the source files for that version:

```
$ java weblogic.Deployer -adminurl http://host01.example.com:7001  
-user weblogic -password Welcome1 -redeploy  
-name myApp  
-source /prod/deployments/myApp/beta1.0/MyApp.war  
-targets myCluster  
-retiretimeout 10
```

Command Line

You may want a rollback to occur quickly if something is wrong with the new version.

The source files of the old version



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Quiz



Why would you deploy a new version of an application in administration mode?

- a. To plan a graceful transition to the new version
- b. To test the new version before making it live
- c. To prepare it before retiring the old version
- d. To configure it before making it live

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

# Agenda

- HTTP Sessions and Redeployment
- Redeployment Strategies
- Production Redeployment
- Application Versioning
- Requirements and Restrictions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Requirements and Restrictions

Production redeployment strategy is supported for:

- Stand-alone WAR modules and EARs whose clients access the application via HTTP
- Enterprise applications that are accessed by **inbound** JMS messages from a global JMS destination or from **inbound** JCA requests
- All types of Web services



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Production redeployment supports only HTTP clients and RMI clients. Specific coding is required to handle the reconnection to the new RMI client version. Your development and design team must ensure that applications using production redeployment are not accessed by an unsupported client. WebLogic does not detect when unsupported clients access the application and does not preserve unsupported client connections during production redeployment.

Production redeployment is **not** supported for:

- Stand-alone EJB or RAR modules. If you attempt to use production redeployment with such modules, WebLogic Server rejects the redeployment request. To redeploy such modules, remove their version identifiers and explicitly redeploy the modules.
- Applications that use Java Transaction Service (JTS) drivers:
  - The `LoggingLastResource` global transaction optimization is not permitted for JDBC application modules.
  - When deploying application-scoped JDBC resources, the `EmulateTwoPhaseCommit` feature is not supported for multiple versions.

- Applications that obtain JDBC data sources via the DriverManager API. To use production redeployment, an application must use JNDI to look up data sources.
- Applications that include EJB 1.1 container-managed persistence (CMP) EJBs. To use production redeployment with applications that include CMP EJBs, use EJB 2.x CMP instead of EJB 1.1 CMP.

## Summary

In this lesson, you should have learned how to deploy versioned applications (production redeployment).



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## Practice 18-1 Overview: Using Production Redeployment

This practice covers the following topics:

- Updating an application to reference a new version of a shared library
- Updating the application to version 2
- Redeploying the updated application
- Observing existing and new clients using the application
- Retiring version 1 of the application
- Rolling back to version 1 of the application



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.



19



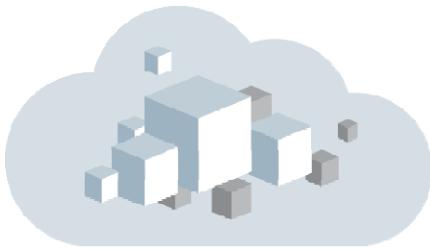
# Oracle Cloud

## An Overview

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Agenda



- 1 What is Cloud Computing?
- 2 Cloud Evolution
- 3 Components of Cloud Computing
- 4 Characteristics and Benefits of Cloud
- 5 Cloud Deployment Models
- 6 Cloud Service Models
- 7 Oracle Cloud Services

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

## What is Cloud?

The term Cloud refers to a Network or Internet.

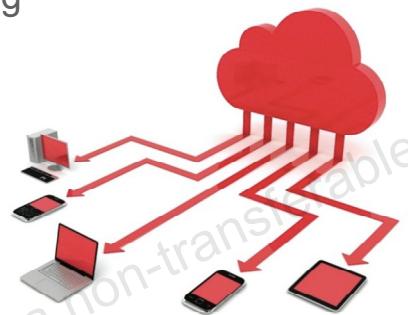
It is a means to access any Software that is available remotely.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# What is Cloud Computing?

- It is a means to access any Software that is available remotely.
- Refers to the practice of using remote Servers hosted on Internet to store, manage and process data
- When you store your photos online instead of on your home computer, or use webmail or a social networking site, you are using a “cloud computing” service.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# History – Cloud Evolution

Cloud Computing

Silos Computing

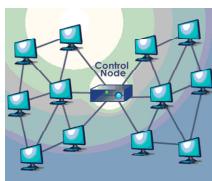
Basic computing with dedicated physical hardware



ORACLE®

Grid Computing

Solving large problems with parallel Computing



Utility Computing

Offering Computing Resources as a metered service

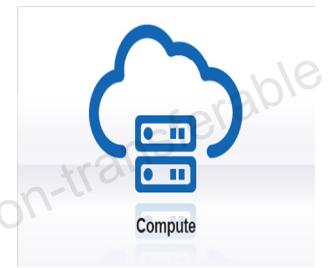


SaaS Computing

Network based subscription to applications



Next generation Internet Computing and next generation Data Centers



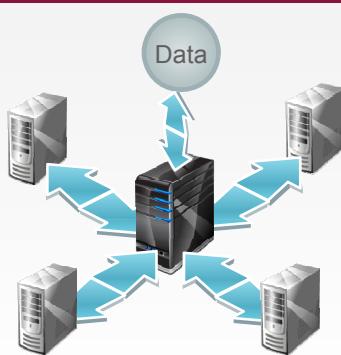
# Components of Cloud Computing

## Client Computers



Devices that end user interact with cloud. Types of client Thick, Thin (Most popular), Mobile

## Distributed Servers



Often Servers are in geographically different places, but server acts as if they are next to each other

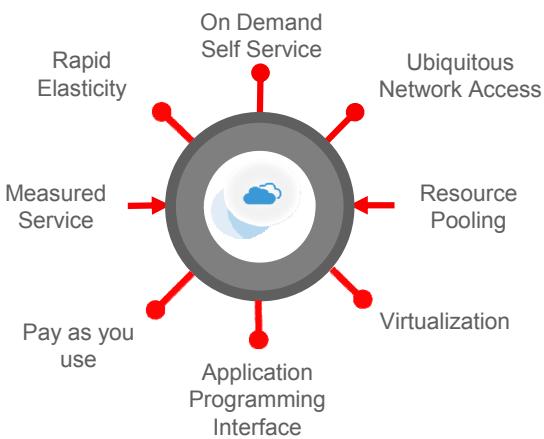
## Data Centers



Collection of servers where application is placed and is accessed via Internet

**ORACLE**

# Characteristics of Cloud



## Description

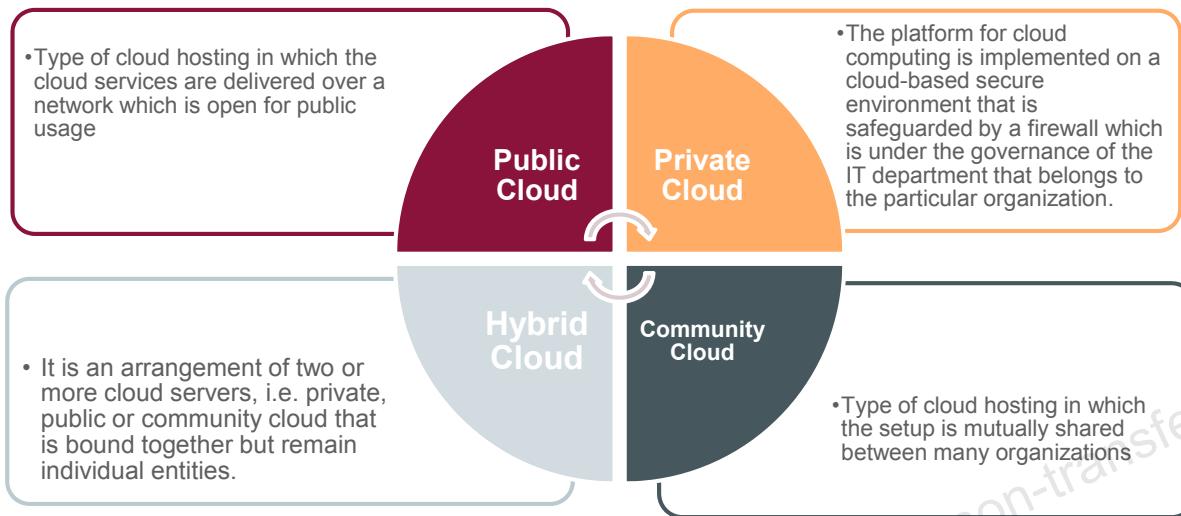
- Allows users to use the service on demand
- Anywhere, Anytime and Any Device
- Draw from a pool of computing resources, usually in remote data centers
- Request and manage own computing resources
- Service is measured and customers are billed accordingly
- Select a configuration of CPU, Memory and storage
- Services can be scaled larger or smaller

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Cloud Deployment Models

Deployment models define the type of access to the Cloud.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Cloud Service Models

All three tiers of computing delivered as Service via global network

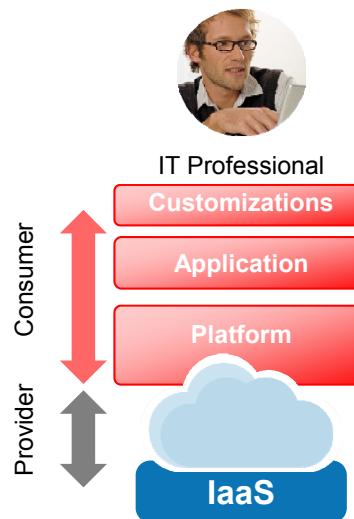
- **Applications:** Software as a Service - SaaS
- **Platform:** Database, Middleware, Analytics, Integration as a Service – Platform as a Service - PaaS
- **Infrastructure:** Storage, Compute, and Network as a service – Infrastructure as a Service - IaaS



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Cloud Service Models

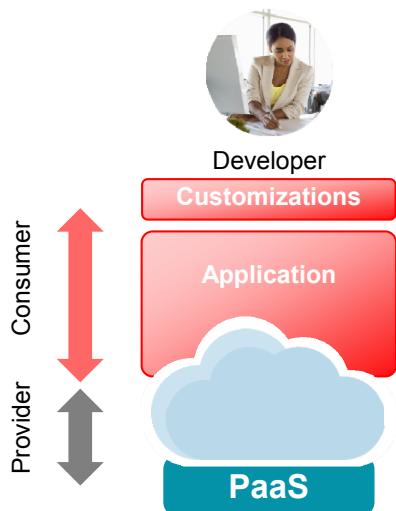


- Provides computer hardware (servers, networking technology, storage and data center space) as a web based service.
- Virtual Machines with pre-installed Operating System
- Target: Administrators
- Ready to Rent

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Cloud Service Models

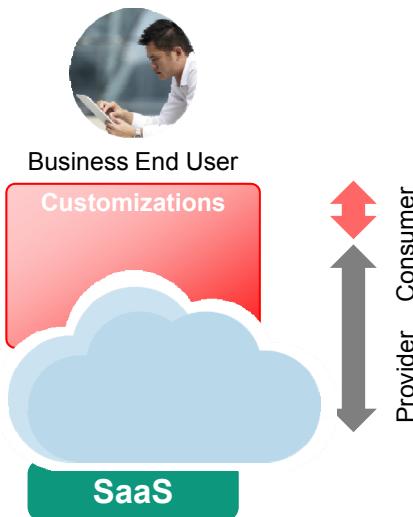


- Provides platform to develop and deploy applications
- Up to Date Software
- Target: Application Developers
- Ready to Use

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Cloud Service Models



- Allows usage of the software remotely as a web based service
- Software are automatically Upgraded and Updated
- All Users are running the same version of the Software
- Target: End Users
- Ready to Wear

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Industry Shifting from **On-Premises to the Cloud**

Transition to the Cloud is driven by a desire for:

**Agility** - Self-service provisioning – deploy a database in minutes

**Elasticity** - Scale on demand

**Lower cost** - Reduction in management and total cost – pay for what is used

**Back to core business** – Focus on core activities

**More mobility** – Access from any device

**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.



ORACLE®

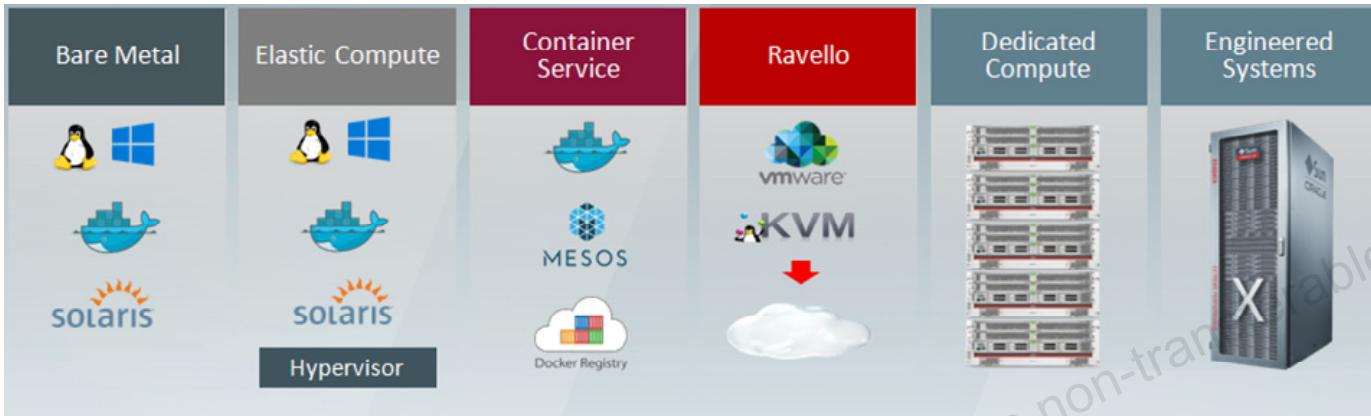
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle IaaS Overview

IaaS

Designed for large enterprises, which allow them to scale up their computing, networking, and storage systems into the cloud, rather than expanding their physical infrastructure.

- Allows large businesses and organizations to run their workloads, replicate their network, and back up their data in the cloud.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Develop, deploy, integrate and manage applications on cloud.
- Seamless integration across PaaS and SaaS Applications.



Database Services



Java Services



Web Scripting Services



Mobile Services



Developer Services



Documents Services



Sites Services



Analytics Services

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle SaaS Overview

SaaS

Delivers modern cloud applications that connect business processes across the enterprise.

- Only Cloud integrating ERP, HCM, EPM, SCM
- Seamless co-existence with Oracle's On-Premise Applications



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Summary

In this lesson, you should have :

- Got an overview of Cloud Computing, its Characteristics, History and Technology
- Understood the various components , Deployment Models and Service Models of Cloud Computing
- Understood the Oracle Cloud Services



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.



# Oracle Java Cloud Service Overview

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Get an overview of Java Cloud Service
- Describe the features of Java Cloud Service



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Introducing Java Cloud Service

## Your platform for running business applications in the cloud

Key Oracle Cloud component 

- Self-service application platform with advanced cloud tools
- Save time and cost with simplified provisioning
- Reduce down time: automated patching, backup, recovery
- Increase data and processing capacity on demand to scale for new business needs
- Optionally enable Oracle Coherence for caching and data grid functions and Oracle Traffic Director for load balancing
- Pre-configured for Database and Developer Cloud Services for complete cloud application management



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Java Cloud Service: Three Options



- Easy to enrich Oracle SaaS apps
- Tailored-made WebLogic Server for rapid extension deployment
- Ready marketplace with pre-built extensions, automated deployment



- Simple, hosted WebLogic instance
- Oracle controlled, updated

ORACLE®

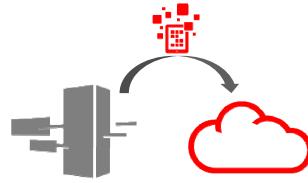
# Java Cloud Service Main Use Cases



Dev/Test in the Cloud



New App Development



Migrate Apps to Cloud



Strategic Outsourcing  
Recapture

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Java Cloud Service Feature: Provisioning

- Can pick shape/size – no complexity
- Choose from popular versions: 11g or 12c
- Meet evolving technical and budgetary needs with popular Edition choices: Standard, Enterprise, Suite

The screenshot shows the Oracle Java Cloud Service administration interface. At the top, it displays the URL [jas.oraclecloud.com/administration/faces/jasRunner.jspx?\\_jfl\\_ocps=2014714129607405&...](https://jas.oraclecloud.com/administration/faces/jasRunner.jspx?_jfl_ocps=2014714129607405&...). The main dashboard shows 1 instance, 2 OCPUs, 15.0 GB memory, 74.0 GB storage, and 2 public IPs. A callout box highlights the 'Create Instance' button. Below the dashboard, there's a section for 'MyJCS1' with details: Version: 12.1.2.0.1, Edition: Enterprise Edition, Nodes: 2, Load Balancer: Configured, Created On: 25-Nov-2014 12:57 AM UTC. A note at the bottom states: 'Failed attempts to create or delete an instance in the last 7 days'. The footer contains links for About Oracle, Contact Us, Legal Notices, Terms of Use, and Your Privacy Rights, along with social media icons. A large red Oracle logo is at the bottom left, and a copyright notice at the bottom right reads: 'Copyright © 2016, Oracle and/or its affiliates. All rights reserved.'

# Java Cloud Service Feature: Patching

The screenshot shows the Oracle Java Cloud Service administration interface. At the top, it displays the URL [jaas.oraclecloud.com/administration/faces/JaaRunner.jspx?\\_sfrLoop=20147141260740646](http://jaas.oraclecloud.com/administration/faces/JaaRunner.jspx?_sfrLoop=20147141260740646). The main header says "ORACLE Java Cloud Service". Below that, it says "Instances > MyJCS1". The service name is "MyJCS1 (Oracle Java Cloud Service)" with "WebLogic Server Version: 12.1.2.0.1". A note says "Description: Auto Generated Service, Session Name: jsddemoxx-wls".  
The interface has three tabs: "OVERVIEW", "BACKUP", and "PATCHING". The "PATCHING" tab is selected, showing "Available Patches" with a note "No patches available." and a "Patch History" section. The history shows one entry: "15.1.2.0-1412241036 Patched By: system on 19-Jan-2015 1:10 PM UTC" with a note "Notes: Tools update submitted by auto...".  
On the left, there's an "ADMINISTRATION" section showing "0 patches available" and backup logs: "11-Feb-2015 1:45 PM UTC most recent backup failed" and "12-Dec-2014 7:58 PM UTC last successful backup".  
At the bottom right, there's a "Logout" button.

- Patching made simple – we handle the details
- You control patch timing - on demand or scheduled
- Includes unified patching of JDK, WLS, JRF/ADF
- Supports rolling patching
- Supports Patchset Updates (PSUs), Patchsets (PS), Upgrades
- Don't mess with backups! Full backup created before patching

# Java Cloud Service Feature: Backup / Restore

The screenshot shows the Oracle Java Cloud Service interface for an instance named 'MyJCS1'. The 'OVERVIEW' section indicates 2 nodes. The 'BACKUP' tab is selected, showing a list of available backups:

- Incremental backup: delta at 1:45 PM UTC, Full backup every 1hr at 1:45 PM UTC
- Most Recent Backup: 11-Feb-2015 1:45 PM UTC
- Last Successful Backup: 12-Dec-2014 7:58 PM UTC

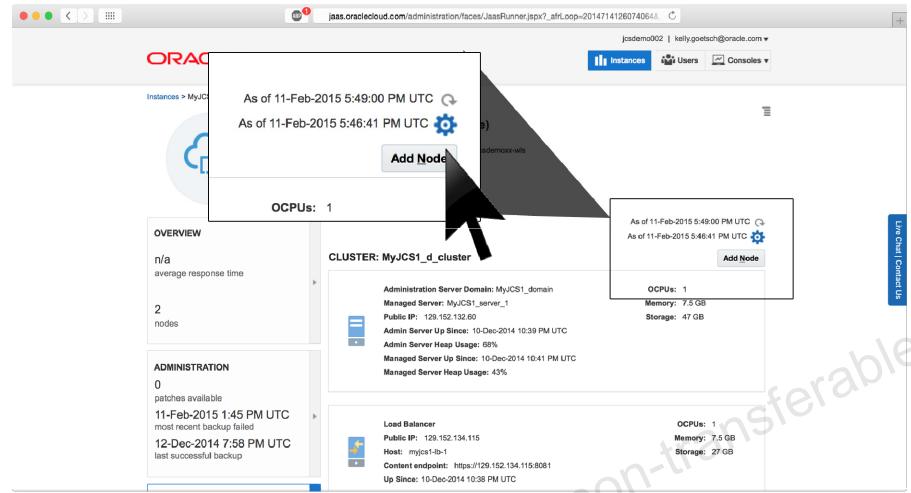
A modal window is displayed over the interface, titled 'As of 11-Feb-2015 1:48:00 PM UTC'. It contains two buttons: 'Configure Backups' and 'Back Up Now'. A large black arrow points from the text below to this modal window.

- Coordinated backups with database and whole cloud stack - holistic backups
- You choose - scheduled or on demand
- Multiple depths supported: configuration/apps, logs, binaries, and database
- Configurable: 7 day backup on local disk, older backups pushed to storage service

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Java Cloud Service Feature: Scaling

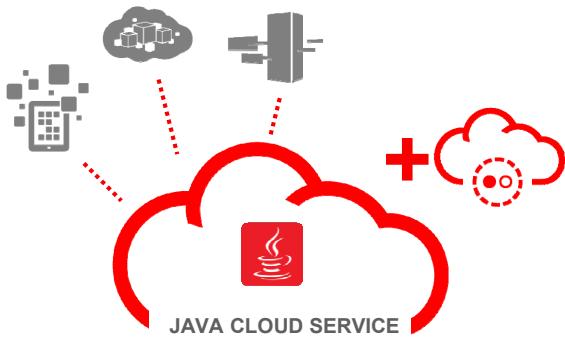
- Fully-automated, on-demand – do it yourself without IT!
- Each managed server on separate virtual machine
- Zero downtime during scaling – keep customers happy
- Scale data capacity and processing up/down on demand
- Rules to trigger scaling based on current workload



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle Coherence Option: Data Caching & Scaling



- Scaling applications' caching/data grid capacity in-memory to support growth
- Offload and protect shared cloud services and databases
- Delivery of data to cloud apps in real time
- Transparency and high-availability in the cloud's data grid tier

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Oracle Coherence Option: Your Cloud Data Grid

## Scalable, fault-tolerant cloud infrastructure

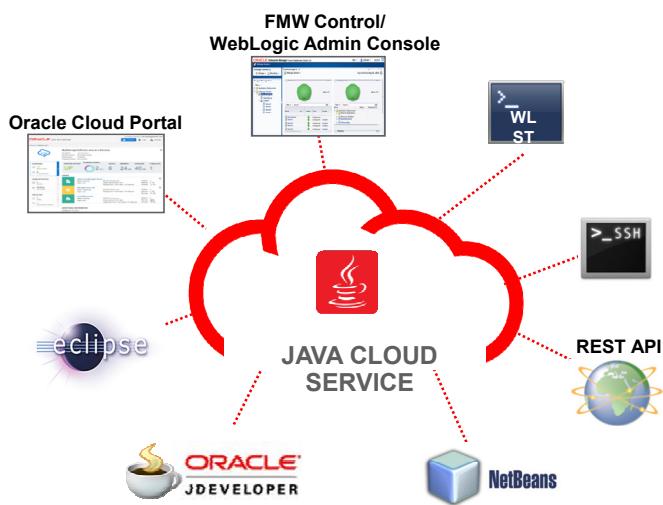
- Reliable In-Memory key-value store
- Dynamically scalable
- Scale processing with data
- Entries can be
  - Reliably processed in-place
  - Queried
  - Aggregated
- Integration with Database and Developer Cloud Services



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# How You Interact with Java Cloud Service



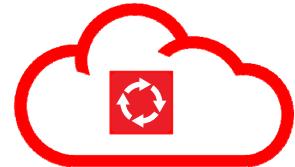
- New Cloud Portal
- WebLogic Admin Console
- Fusion Middleware Control
- Traffic Director Admin Console
- Public REST APIs
- Command Line Interface
- SSH or VNC to VM
- Standard IDEs

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Speaking of Dev Environments... Developer Cloud Service

- Complete, Integrated Development Platform - as a Service
- Application Lifecycle Management
- Team Management
- Entitlement with Java Cloud Service



Source Control Management



Issue Tracking



Hudson Continuous Integration



Wiki Collaboration

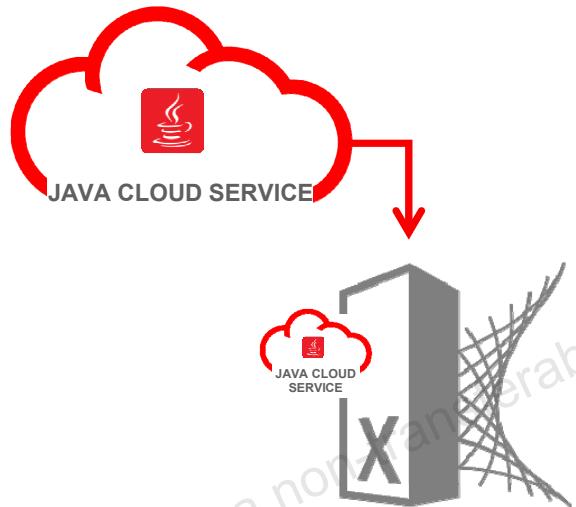
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Java Cloud Service On-Premises!

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

- If you want to use public cloud, but can't
- Geography, political, other reasons
- Same public experience, but on-premises
- Runs on Oracle Cloud Machine in your data center



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# Summary

In this lesson, you should have :

- Got an overview of Java Cloud Service
- Understood the features of Java Cloud Service



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.



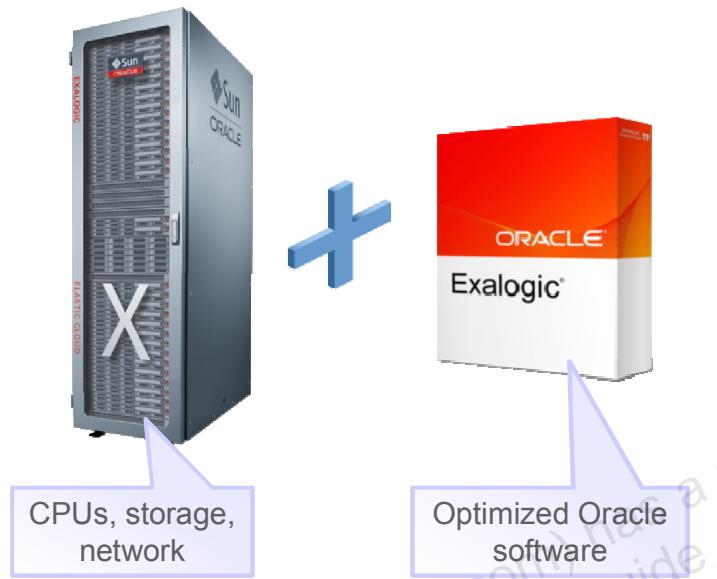
# WebLogic Optimizations for Exalogic

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Sean Tay (taysiangmengsean@gmail.com) has a non-transferable  
license to use this Student Guide.

## What Is Exalogic?



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Exalogic is an integrated hardware and software system designed to provide a complete platform for a wide range of application types and widely varied workloads. Exalogic is intended for large-scale, performance-sensitive, mission-critical application deployments. It combines Oracle Fusion Middleware and Sun hardware to enable a high degree of isolation between concurrently deployed applications, which have varied security, reliability, and performance requirements. Exalogic enables customers to develop a single environment that can support end-to-end consolidation of their entire applications portfolio.

Exalogic hardware is preassembled and delivered in standard 19" 42U rack configurations. The main hardware components of a single Exalogic X3-2 machine Full Rack are the following:

- 30 Sun Server X3-2 (formerly Sun Fire X4170) compute nodes
- One dual controller Sun ZFS Storage 7320 appliance with 20 disks
- Four Sun Network QDR InfiniBand Gateway Switches
- One Sun Datacenter InfiniBand Switch 36
- One 48-port Cisco Catalyst 4948 Ethernet management switch
- Two redundant 24 kVA power distribution units

## What Is Exalogic?

### Oracle Exalogic:

- Combines Sun storage and x86 servers by using a high-speed InfiniBand network
- Is specifically engineered to host Oracle Fusion Middleware software
- Includes optimizations to Oracle Fusion Middleware software that result in huge performance gains
- Supports Oracle Enterprise Manager for centralized configuration, monitoring, and diagnostics
- Provides an integrated IaaS cloud solution



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can connect up to eight Exalogic machines, or a combination of Exalogic machines and Oracle Exadata database machines, together without the need for any external switches. If more than eight racks are required to be connected on the same InfiniBand fabric, Oracle offers a choice of several high-capacity data center switches, which enable the creation of Exalogic clouds comprising hundreds of racks and tens of thousands of processors.

Exalogic is designed to fully leverage an internal InfiniBand fabric that connects all of the processing, storage, memory, and external network interfaces within an Exalogic machine to form a single, large computing device. Each Exalogic machine is connected to the customer's data center networks via 10 GB Ethernet (external traffic) and 1 GB Ethernet (management traffic) interfaces.

Oracle Enterprise Manager Grid Control with Oracle WebLogic Server Management Pack Enterprise Edition's capabilities include Exalogic specific management tools to monitor the Oracle software deployed in the Exalogic environment. If using Solaris as your Exalogic operating system, you can also use Oracle Enterprise Manager Ops Center to provide configuration and management capabilities for the Exalogic hardware components.

## What Is InfiniBand?

- Distributed systems communicate with one another to:
  - Separate application tiers (Web, business logic, caching)
  - Provide high availability (“clusters”)
- InfiniBand:
  - Is a standard network architecture designed for very high speeds between nodes in close proximity to each other
  - Acts as a “fabric” to connect CPUs and other devices
  - Is ideal for supercomputers and data centers
  - Delivers 40 Gb/s (4x faster than high speed Ethernet)
  - Supports direct application connections that bypass the OS
  - Is interoperable with Ethernet for external communication



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Moving data between applications over a traditional network can consume a lot of time and drain precious server resources. With traditional network technologies, data exchanges traverse the operating systems on both the source and destination servers, resulting in excessive application latency due to operating system calls, buffer copies, and interrupts.

InfiniBand, which today delivers 40 GB per second connectivity with application-to-application latency as low as 1 microsecond, has become a dominant fabric for high-performance enterprise clusters. Its ultra-low latency and near-zero CPU utilization for remote data transfers make InfiniBand ideal for high-performance clustered applications.

InfiniBand also provides a direct channel from the source application to the destination application, bypassing the operating systems on both servers. InfiniBand’s channel architecture eliminates the need for OS intervention in network and storage communication. This frees server memory bandwidth and CPU cycles for application processing.

In addition to carrying all InfiniBand traffic, the Sun Network QDR InfiniBand Gateway Switch enables all InfiniBand attached servers to connect to an Ethernet LAN by using standard Ethernet semantics. No application modifications are required for applications written to use standard Ethernet.

# Exalogic Networks

Network	Speed	Description
Client	10 Gb	<ul style="list-style-type: none"> <li>Provides external access to compute nodes via the intranet/Internet</li> </ul>
Management	1 Gb	<ul style="list-style-type: none"> <li>Is used to monitor and administer components</li> <li>Provides access to ILOM and other device management interfaces</li> </ul>
Private InfiniBand	40 Gb	<ul style="list-style-type: none"> <li>Is a private, non-routable network</li> <li>Connects racks, along with all rack components</li> <li>Is used by compute nodes to access shared storage</li> <li>Is used for internal application communication</li> </ul>

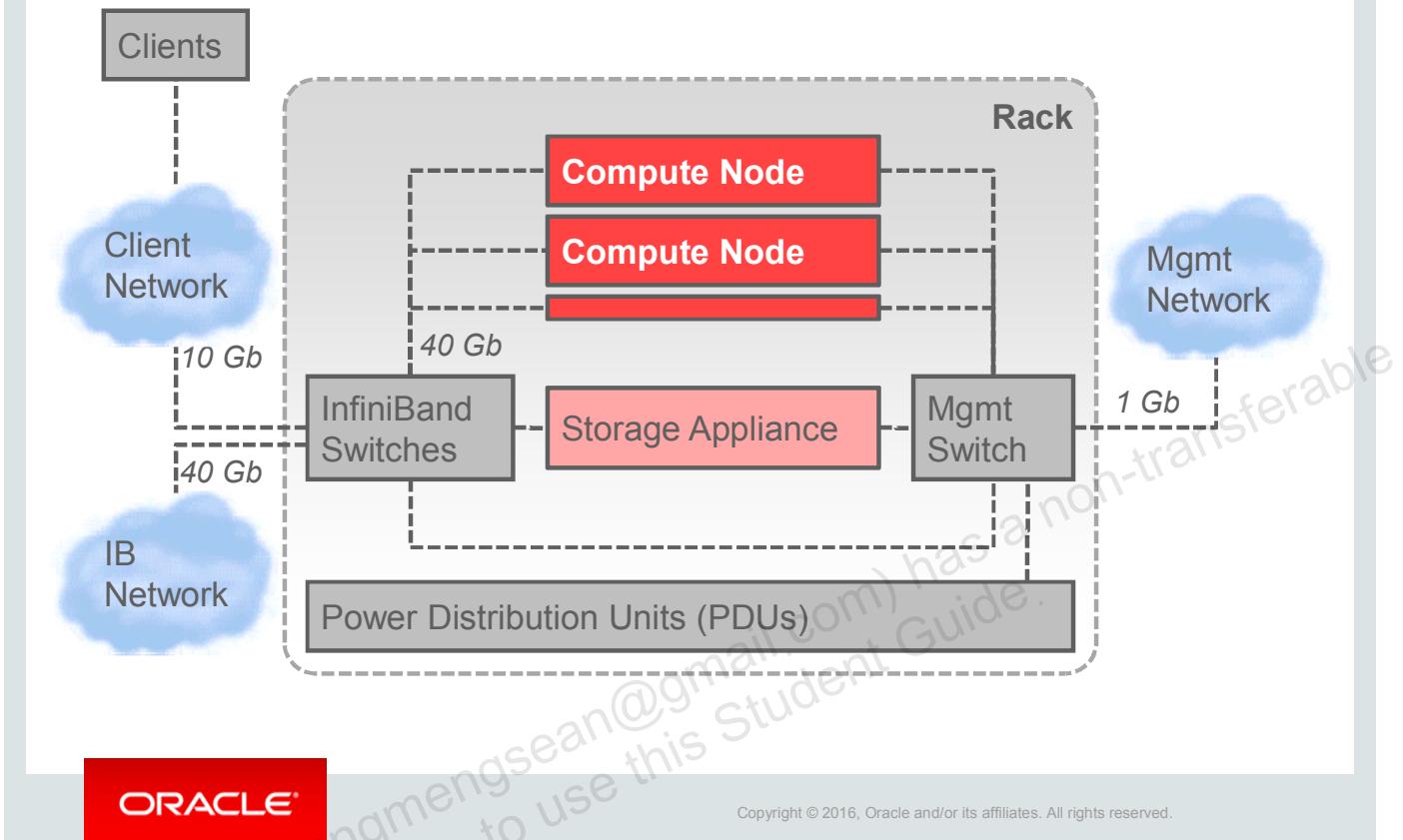


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

An Exalogic machine includes compute nodes, a Sun ZFS Storage 7320 appliance, as well as equipment to connect the compute nodes to your network. The network connections allow the servers to be administered remotely, enable clients to connect to the compute nodes, and enable client access to the storage appliance. Additional configuration, such as defining multiple virtual local area networks (VLANs) or enabling routing, may be required for the switches to operate properly in your environment and is beyond the scope of the installation service.

There are up to five networks for an Exalogic machine. Each network must be on a distinct and separate subnet from the others. The Exalogic management network connects to your existing management network and is used for administrative work for all components of the Exalogic machine. It connects ILOM, compute nodes, server heads in the storage appliance, and switches connected to the Ethernet switch in the Exalogic machine rack. This management network is in a single subnet. Do not use the management network interface (ETH0/NET0) on compute nodes for client or application network traffic. Cabling or configuration changes to these interfaces on Exalogic compute nodes is not permitted.

# Exalogic Machine Topology



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Exalogic hardware is preassembled and delivered in standard 19" 42U rack configurations. Each Exalogic configuration is a unit of elastic cloud capacity balanced for compute-intensive workloads. Each Exalogic configuration contains several hot-swappable compute nodes along with a clustered, high-performance disk storage subsystem. The hardware also includes a high-bandwidth InfiniBand fabric to connect every individual component within the configuration as well as to externally connect additional Exalogic or Exadata Database Machine racks. In addition, each configuration includes multiple 10 Gb Ethernet ports for integration with the data center's service network, along with 1 Gb Ethernet ports used for integration with the data center's management network. All Exalogic configurations are fully redundant at every level and are designed with no single point of failure.

All device management ports are connected to your local data center management network by using a Cisco Catalyst 4948 switch, which is a built-in component of an Exalogic rack. This switch offers 48 ports of wire-speed 10/100/1000BASE-T with 4 alternative wired ports that can accommodate optional 1000BASE-X Small Form-Factor Pluggable (SFP) optics. Reliability and serviceability are delivered with optional internal AC or DC 1 + 1 hot-swappable power supplies and a hot-swappable fan tray with redundant fans.

# InfiniBand Networking Concepts

Network Term	Definition	Description
<b>Private</b>	<b>IPoIB</b>	IP over InfiniBand
	<b>SDP</b>	Socket Direct Protocol
<b>Client</b>	<b>EoIB</b>	Ethernet over InfiniBand
	<b>VNIC</b>	Virtual Network Interface Card

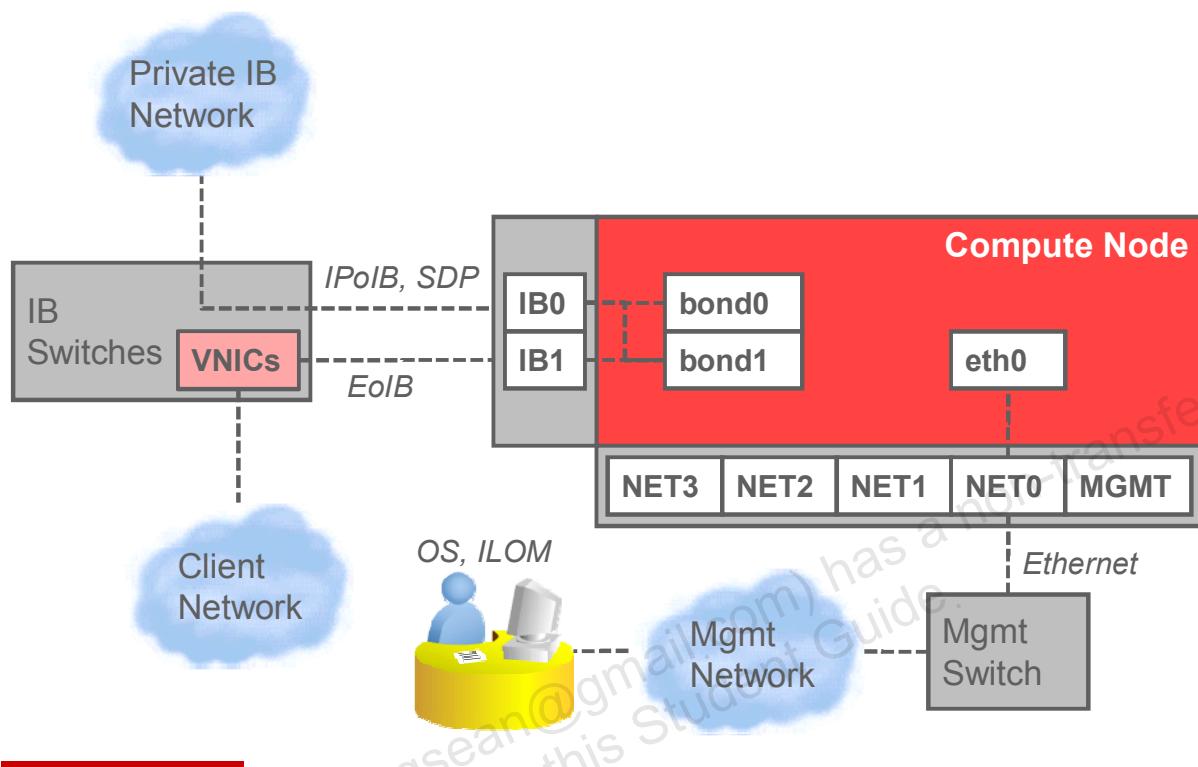


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

IP over InfiniBand (IPoIB) enables applications on separate devices to communicate with each other over a private InfiniBand fabric by using native IB protocols and without the overhead of Ethernet. For example, a compute node on one Exalogic rack may communicate with a database on an Exadata rack. However, applications must support the SDP protocol to use IPoIB instead of the default TCP/IP stack of the host operating system.

The InfiniBand switches also act as gateways to connect to external Ethernet networks. They support eight 10 Gb Ethernet ports. Exalogic compute nodes can communicate through these ports by using Ethernet over InfiniBand (EoIB). Each port is represented on the compute nodes as a VNIC. This allows that node's IB connection to appear like any other Ethernet NIC to both the operating system and to the external Ethernet network.

## Default Compute Node Network

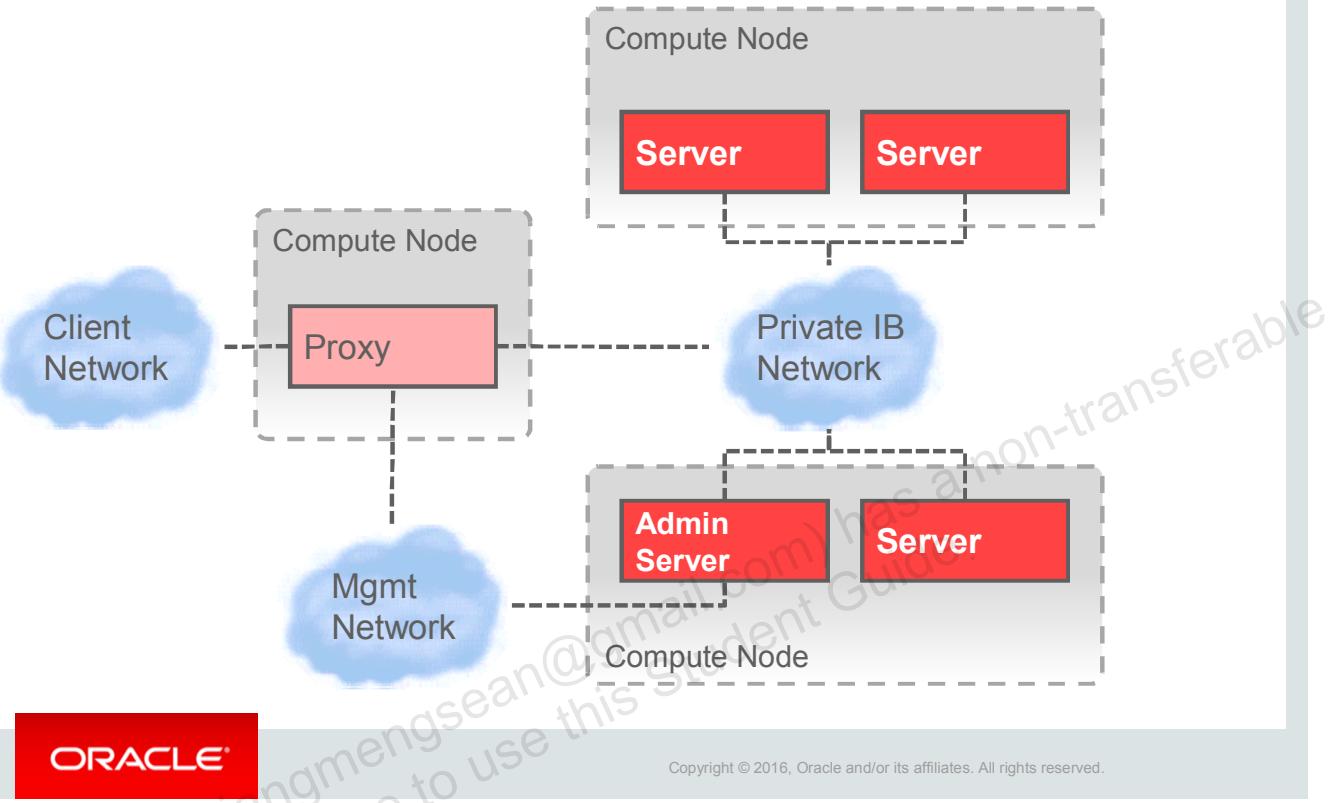


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

By default, each Exalogic compute node is configured with one bonded EoIB interface for one external LAN (client network), and is named **bond1**. The Cisco Ethernet management switch is connected to the **NET0** port of compute nodes, the **NET0** port of the storage appliance, and also the management ports of the InfiniBand gateway switches. On the compute nodes, this connection is represented on the operating system by an “**eth**” network interface, such as **eth0**. The compute nodes are configured at the time of manufacturing to use sideband management only. Therefore, the **MGMT** (or **ILOM**) port is not connected, but **ILOM** is accessible from **NET0**.

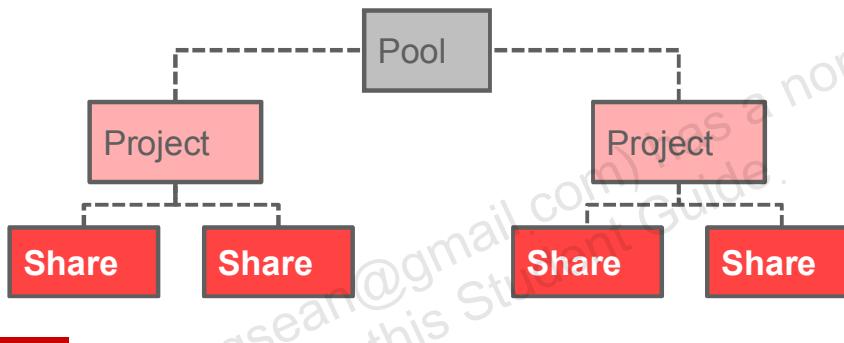
# Recommended FMW Topology on Exalogic

Unauthorized reproduction or distribution prohibited. Copyright© 2018, Oracle and/or its affiliates.



# Exalogic Shared Storage Concepts

Term	Definition
<b>Pool</b>	High availability and replication characteristics for a collection of disks
<b>Project</b>	Default administrative and file system settings for a collection of shares
<b>Share</b>	A file system mount point, access protocols, access rights, and other settings



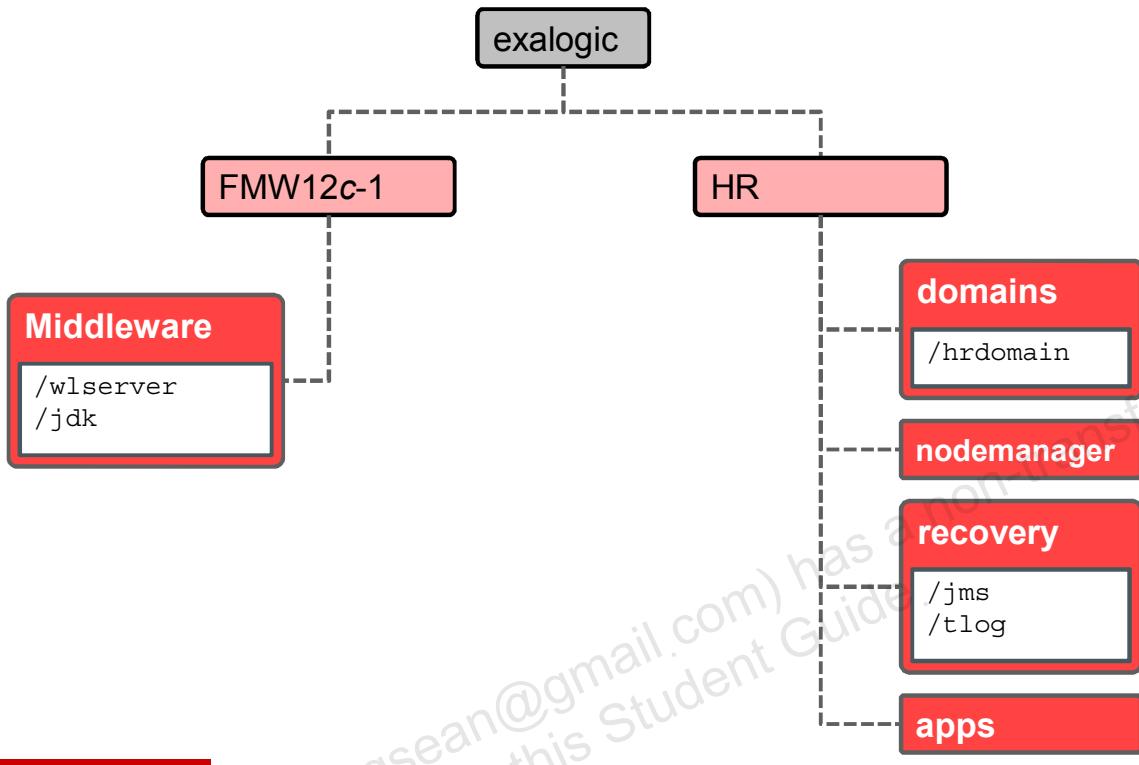
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The appliance is based on the ZFS file system. ZFS groups the underlying storage devices into pools. Shared file systems then allocate disk space from these pools. Before creating file systems, you must first configure storage on the appliance. After a storage pool is configured, you do not have to statically size file systems, although this behavior can be achieved by using quotas and reservations.

Each storage node can have any number of pools, and each pool can be assigned ownership independently in a cluster. While an arbitrary number of pools is supported, creating multiple pools with the same redundancy characteristics owned by the same cluster head is not advised. Doing so results in poor performance, suboptimal allocation of resources, artificial partitioning of storage, and additional administrative complexity.

The storage appliance exports file systems as shares, which are managed in this section of the appliance. All file systems are grouped into projects. A project defines a common administrative control point for managing shares. All shares within a project can share common settings, and quotas can be enforced at the project level in addition to the share level. Projects can also be used solely for grouping logically related shares, so their common attributes (such as accumulated space) can be accessed from a single point.

# Recommended FMW Storage on Exalogic



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

# WebLogic Server Optimizations: Overview

1. Configure WebLogic network channels to use:
  - eth0 for default admin communication
  - bond0 for cluster replication traffic
  - bond1 for client communication (if not using a proxy)
2. Enable these features on replication channels:
  - SDP
  - Multiple ports
  - One-way RMI
3. Enable other domain-wide Exalogic optimizations.



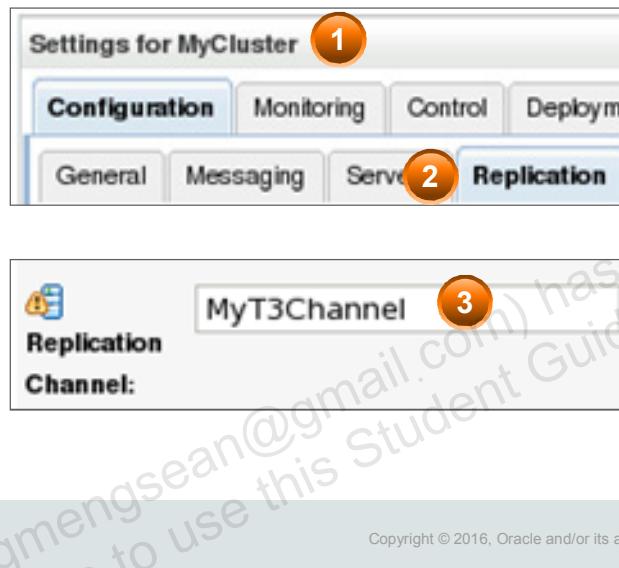
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Most Exalogic specific WLS optimizations are not enabled by default. Some require a simple configuration check box. But others require multiple configuration steps. For example, WLS clusters can be configured to further improve server-to-server communication. First, you can enable multiple replication channels, which improve network throughput among cluster members. Second, you can enable InfiniBand support via the Sockets Direct Protocol, which reduces CPU utilization because network traffic bypasses the TCP stack.

# Creating a Cluster Replication Channel

1. Create an internal T3 channel for each member of a cluster.
2. Assign the channel to the cluster (default name is “ReplicationChannel”).



When creating a network channel, there is not a specific protocol option available for internal cluster replication traffic. Instead, you must configure a channel that supports the T3 protocol and specify the name of that channel as part of a cluster's replication settings.

1. Click Clusters and then select a cluster.
2. Click the Configuration > Replication tab.
3. Set the Replication Channel to the name of the custom network channel that you created on each member of the cluster. As a result, these channels will be used for replication traffic instead of the default channels on each server.

## Using SDP for Replication



startWebLogic.sh:

```
...  
. ${DOMAIN_HOME}/bin/setDomainEnv.sh $*  
JAVA_OPTIONS="$JAVA_OPTIONS -Djava.net.preferIPv4Stack=true"
```

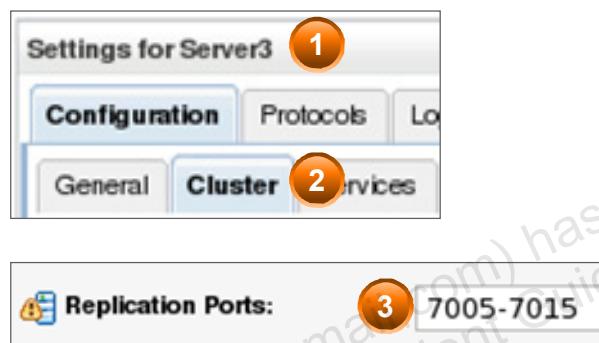
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

ORACLE

WLS can use SDP to bypass the operating system and communicate directly with the InfiniBand fabric for session replication.

# Using Multiple Ports for Replication

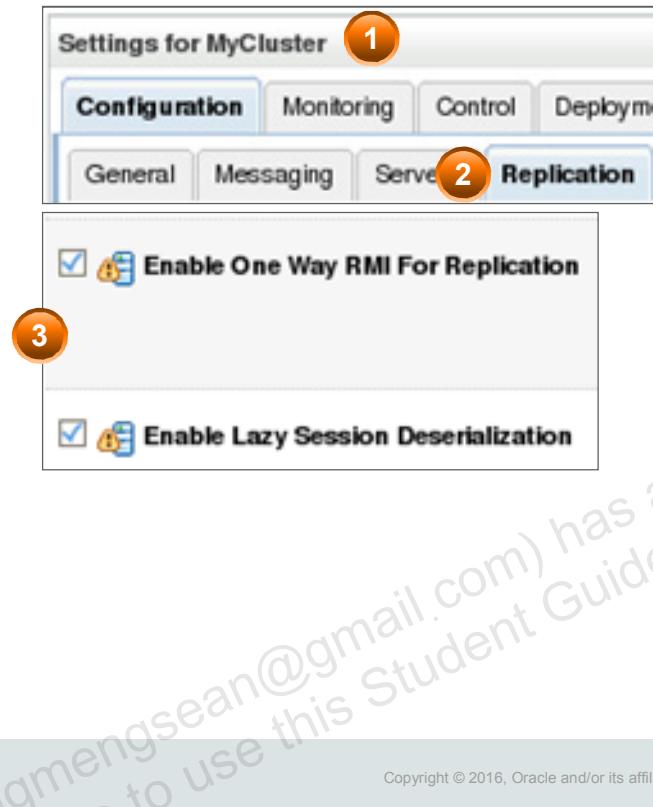
- A single replication channel cannot use the available bandwidth on the IB network.
- For convenience, simply specify a port range and multiple channels will be created and used automatically.



When members of a cluster need to communicate and replicate HTTP session data for high availability, they use an internal Java protocol called T3. Because Exalogic uses InfiniBand, establishing individual T3 connections does not take full advantage of the available bandwidth. Instead, WLS uses parallel or “multiplexed” connections for this inter-cluster communication.

Multiple replication channels do not need to be configured on each clustered server instance. Only one replication channel with explicit IP-Address needs to be configured for each server and replication. Port range can be specified for each server. For example, the range 7001-7010 will create 10 replication channels with ports 7001 to 7010 for the given server. These channels inherit all the properties of the configured replication channel except the listen port. Names of these channels will be derived from the name of the configured replication channel. A numeric suffix (1,2,3) will be added to each name. If you specify a range of 10 ports, 10 additional channels will be created. Public ports are the same as the listen port for these additional channels.

## Other Replication Optimizations



As replicated Java objects are passed across the InfiniBand network, they must be serialized and deserialized by WLS. To avoid unnecessary processing, WLS deserializes a replicated object only if the server from which it originated has failed.

## Oracle Exabus

- Is a collection of proprietary software libraries included in Exalogic base OS images
- Allows direct access to InfiniBand hardware while bypassing the kernel, including SDP
- Enables Remote Direct Memory Access (RDMA), which reduces buffer copying between apps, kernel, and drivers.
- Is used by FMW to greatly increase performance and throughput, while also reducing latency



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Exabus consists of unique software, firmware, and device drivers, and is built on Oracle's QDR InfiniBand technology. It includes a low-level Java API for the network communications layer that is specifically optimized for Exalogic. The Exabus implementation uses RDMA (direct memory I/O) and other optimizations for low latency communications.

## Other WebLogic Optimizations

Subsystem	Description
<b>Threading</b>	Create more threads to take advantage of the compute node's processing power.
<b>Network I/O</b>	<ul style="list-style-type: none"><li>• Use Exabus for tighter integration with native IB libraries</li><li>• Use larger packet sizes to take advantage of the IB throughput.</li></ul>
<b>File I/O</b>	Use larger buffers to take advantage of the IB connection to the storage appliance.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

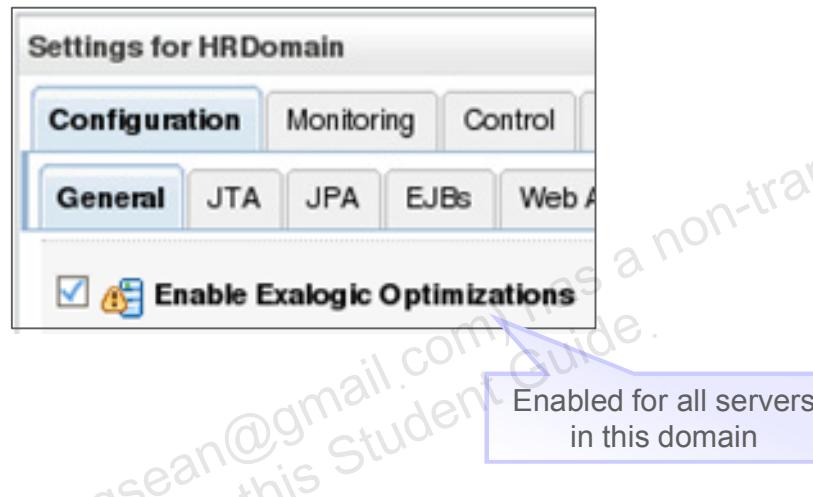
Traditional I/O employs the use of buffers to read and write data, and as this data is transferred between operating system memory, JVM memory, and WLS memory (heap), it must be copied from buffer to buffer. On Exalogic, this file I/O buffering is significantly reduced or eliminated.

InfiniBand supports much higher throughput rates compared to standard Ethernet, so WebLogic automatically uses larger packet sizes to communicate with the InfiniBand fabric on Exalogic. This includes external network communication as well as communication with other servers on other compute nodes.

# Enabling Other Optimizations

Optimizations can be turned on and off:

- Collectively by using a domain-wide flag
- Individually by using command-line server arguments



To enable all general optimizations on all servers in a domain, perform the following steps:

1. Access the WebLogic Server console.
2. In the Domain Structure panel, click the name of the domain.
3. In the right panel, select the Enable Exalogic Optimizations check box.

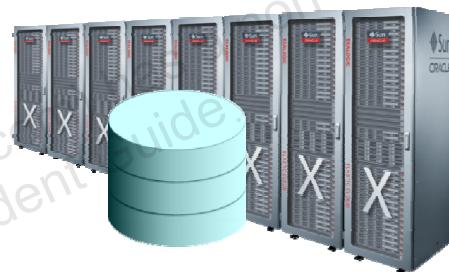
Refer to the Deployment Guide for a complete list of server startup arguments that individually enable or disable specific enhancements. Here are some examples:

- `-Dweblogic.ScatteredReadsEnabled=true/false`
- `-Dweblogic.GatheredWritesEnabled=true/false`
- `-Dweblogic.replication.enableLazyDeserialization=true/false`

## What Is Exadata?

The Oracle Exadata solution:

- Combines Sun hardware and optimized Oracle software, similar to Exalogic
- Is engineered to run very large Oracle DB deployments
- Uses an InfiniBand fabric to connect nodes or “cells”
- Dedicates separate hardware to computational-intensive processing and data-intensive processing
- Helps standardize and consolidate your data warehouse and transaction processing



**ORACLE®**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Oracle Exadata Database Machine is an easy-to-deploy solution for hosting Oracle Database that delivers the highest levels of database performance. The Exadata Database Machine is composed of database servers, Oracle Exadata Storage Servers, and an InfiniBand fabric for networking all the components. It delivers outstanding I/O and SQL processing performance for online transaction processing (OLTP) and data warehousing.

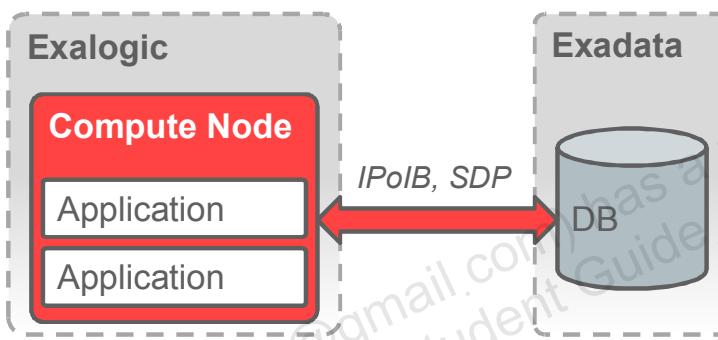
There are two versions of the Exadata Database Machine. The Exadata Database Machine X2-2 expands from two 12-core database servers with 192 GB of memory and three Exadata Storage Servers to eight 12-core database servers with 768 GB of memory and 14 Exadata Storage Servers, all in a single rack. The Exadata Database Machine X2-8 comprises two 64-core database servers with 2 TB of memory and 14 Exadata Storage Servers in a single rack.

Exadata Smart Flash Cache dramatically accelerates Oracle Database processing by speeding I/O operations. The Flash provides intelligent caching of database objects to avoid physical I/O operations. The Oracle database on the Database Machine is the first Flash-enabled database. Exadata storage uses an advanced compression technology, Exadata Hybrid Columnar Compression, that typically provides 10 times and higher levels of data compression than a traditional database server.

## Exadata InfiniBand Connectivity

Applications on Exalogic and databases on Exadata can:

- Share the same IB fabric
- Directly communicate with each other by using IPoIB
- Enable SDP



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

It is possible to connect as many as eight full racks of Exalogic hardware (or any combination of Exalogic and Exadata configurations) without the need for any external switches. In cases where more than eight racks of Exalogic or Exadata hardware are required, Oracle offers a choice of several high-capacity data center switches that enable the creation of Exalogic clouds comprising hundreds of racks and tens of thousands of processors.

# Enabling SDP for Exadata Connectivity

Edit the default data source configuration and change all occurrences of “TCP” to “SDP” in the URL.



startWebLogic.sh:

```
...
. ${DOMAIN_HOME}/bin/setDomainEnv.sh $*
JAVA_OPTIONS="${JAVA_OPTIONS} -Djava.net.preferIPv4Stack=true
-Doracle.net.SDP=true"
```

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Net Services provides support for the Sockets Direct Protocol (SDP) for InfiniBand high-speed networks. For example, InfiniBand can be used to connect an Exalogic machine to an Exadata machine. SDP is characterized by short-distance, high-performance communications between multiple server systems.

Simply connecting the machines with InfiniBand cables is not sufficient—WebLogic and RAC will still communicate by using TCP (IPoIB). First, configure an SDP address in the `listener.ora` file on the database server. Then edit your GridLink data source by using the WLS console. On the Configuration > Connection Pool tab, locate the URL field. Replace all instances of the text `PROTOCOL=TCP` with `PROTOCOL=SDP`. Finally, use the Control tab to restart the data source.

Note that in order to use SDP, you will also need to add a command-line argument when starting your WebLogic Servers: `-Djava.net.preferIPv4Stack=true`. For example, edit `startWebLogic.sh`.