13

Clusters

Proxies and Sessions

Objectives

After completing this lesson, you should be able to:

- Install Oracle HTTP Server
- Configure Oracle HTTP Server as a cluster proxy
- Configure session failover
- Configure replication groups

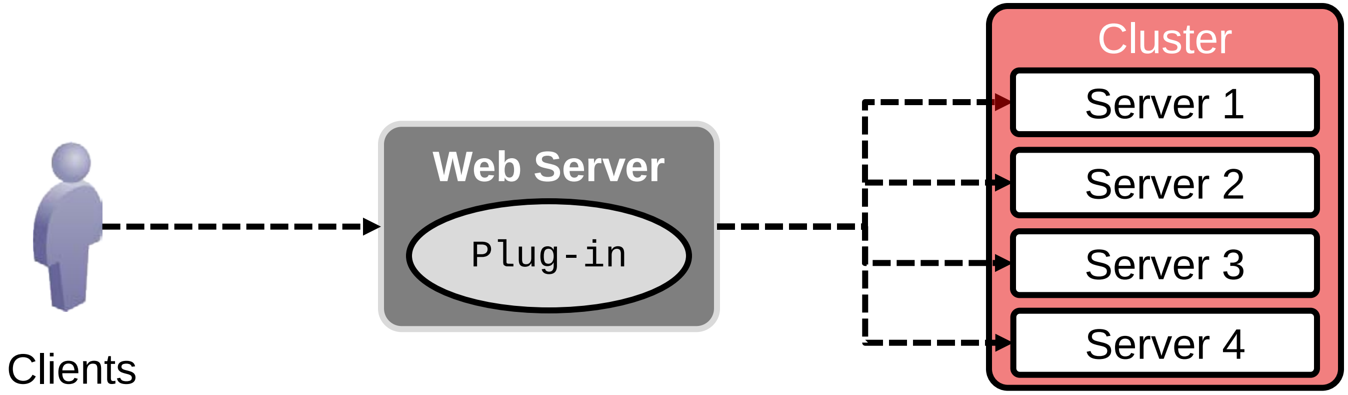
A Cluster Proxy for a Web Application Cluster

- A cluster proxy provides load balancing and failover for a web application cluster. It gives the cluster its “single server” appearance.
- There are basically two kinds of cluster proxies:
  - A web server with the WebLogic proxy plug-in
  - A hardware load balancer

Cluster Proxy	Advantages	Disadvantages
Web server with plug-in	• Low cost (or free) • You probably already have experience with the web server	• Only round-robin load balancing available • Must configure the plug-in
Hardware load balancer	• More sophisticated load balancing algorithms • No plug-in configuration	• Cost • Must be compatible with the WebLogic Server session cookie

Proxy Plug-Ins

- A proxy plug-in:
  - Load balances client requests to clustered WebLogic Server instances in a round-robin fashion
  - Avoids routing requests to failed servers in the cluster
  - Routes requests based on WebLogic Server session cookie information



Oracle HTTP Server (OHS)

OHS is a web server that is:

- A component of the Oracle Web Tier Suite
- Based on Apache HTTP Server
- Installed with the WebLogic Server plug-in module (`mod_wl_ohs`) by default
  - The plug-in must be configured by using the `mod_wl_ohs.conf` file.
- Managed and monitored by using the Oracle Process Manager and Notification Server (OPMN)
  - OPMN manages and monitors non-Java components of Oracle Fusion Middleware.
  - OPMN can be accessed by using the `opmnctl` command-line utility.

Installing and Configuring OHS (Part of Oracle Web Tier): Overview

1. Download and unzip the Web Tier installer.
2. Run the executable under `Disk1`: `runInstaller`.
   A. Choose the **Install Software - Do Not Configure** option.
   B. Specify the Web Tier installation location.
3. Configure an OHS instance by navigating to `<WEB_TIER>/bin` and running the Web Tier Configuration Wizard: `config.sh`.
   A. Under Configure Components, select **Oracle HTTP Server**.
   B. Enter the Instance Home Location, the Instance Name, and the OHS Component Name.
   C. Configure the ports (select either **Auto Port Configuration** or a port configuration file).
   D. Click **Configure**, and at 100% complete, click **Finish**.

This location is called “oracle instance.”

Configuring OHS as the Cluster Proxy

Modules extend the functionality of OHS to enable it to integrate with other Fusion Middleware components.

- The proxy plug-in for WebLogic Server is called `mod_wl_ohs.so` and is found here: `<WEB_TIER>/ohs/modules`.
  - The plug-in is already installed, but must be configured.
- Configuration files for OHS are found here: `<ORACLE_INSTANCE>/config/OHS/`*`OHS_instance_name`*.
  - The main configuration file is `httpd.conf`. It contains an include directive for the WebLogic plug-in configuration file:
  - `mod_wl_ohs.conf`. This is the file you edit to configure OHS to proxy a WebLogic Server cluster.

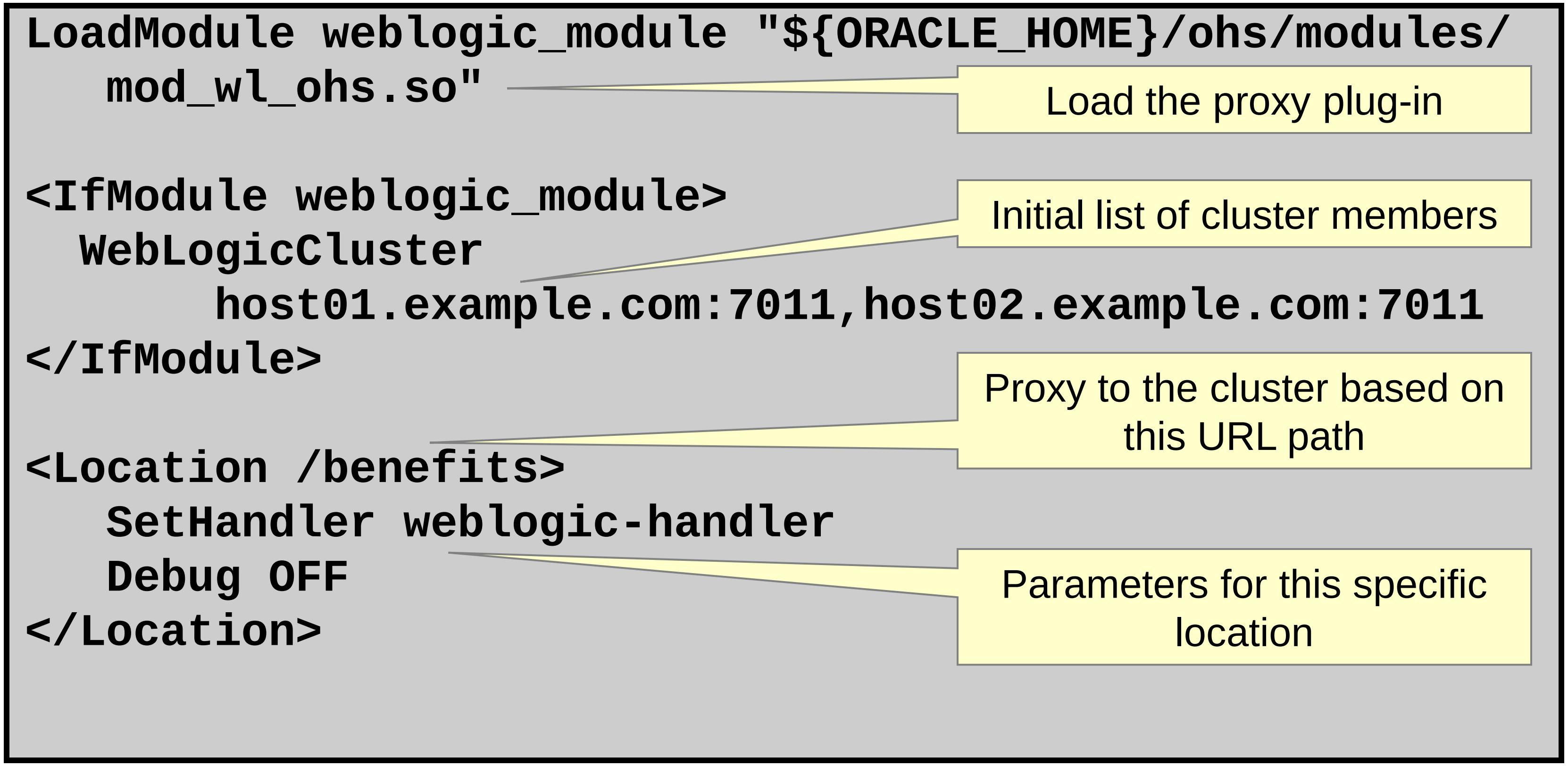
`httpd.conf` and `mod_wl_ohs.conf`

- The include directive in `httpd.conf` looks like this (all on one line):

```
include "${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/
                           ${COMPONENT_NAME}/mod_wl_ohs.conf"
```

- The `mod_wl_ohs.conf` file has various directives, but the `WebLogicCluster` directive is the most important.
  - It specifies the initial list of servers in the cluster, giving their host names and ports.
    - Remember, you do not need to update this list in the configuration file to add or remove servers from the cluster. This is the *initial* list of cluster members. Once the cluster is running, the plug-in uses the dynamic server list.

mod_wl_ohs.conf



```
LoadModule weblogic_module "${ORACLE_HOME}/ohs/modules/
   mod_wl_ohs.so"

<IfModule weblogic_module>
  WebLogicCluster
       host01.example.com:7011,host02.example.com:7011
</IfModule>

<Location /benefits>
   SetHandler weblogic-handler
   Debug OFF
</Location>
```

Load the proxy plug-in

Initial list of cluster members

Proxy to the cluster based on this URL path

Parameters for this specific location

Some Plug-in Parameters

Parameter	Description
`WebLogicHost, WebLogicPort`	Proxy to a single server with this host and port
`WebLogicCluster`	Proxy to this initial list of clustered servers
`MatchExpression`	Proxy requests for files of this MIME type
`PathTrim`	Remove this text from the incoming URL path before forwarding a request.
`PathPrepend`	Add this text to the incoming URL path before forwarding a request.
`ErrorPage`	URL to direct users to if all servers are unavailable
`WLExcludePathOrMimeType`	Do not proxy for this specific URL path or MIME type.
`WLProxySSL`	Set to `ON` to establish an SSL connection to WebLogic if the incoming request also uses HTTPS.
`MaxPostSize`	Maximum allowable size of POST data, in bytes
`Debug`	Sets the type of logging performed

Starting and Stopping OHS

- OHS is managed by OPMN.
  - The command-line interface to OPMN is `opmnctl`.
- `opmnctl` examples:

Start OPMN and all managed processes, if not already started:

```
$> ./opmnctl startall
```

Start all OHS processes, if not already started:

```
$> ./opmnctl startproc process-type=OHS
```

Get the name, status, memory usage, and port number of processes:

```
$> ./opmnctl status -l
```

Stop all OHS processes:

```
$> ./opmnctl stopproc process-type=OHS
```

Verifying that OHS Is Running

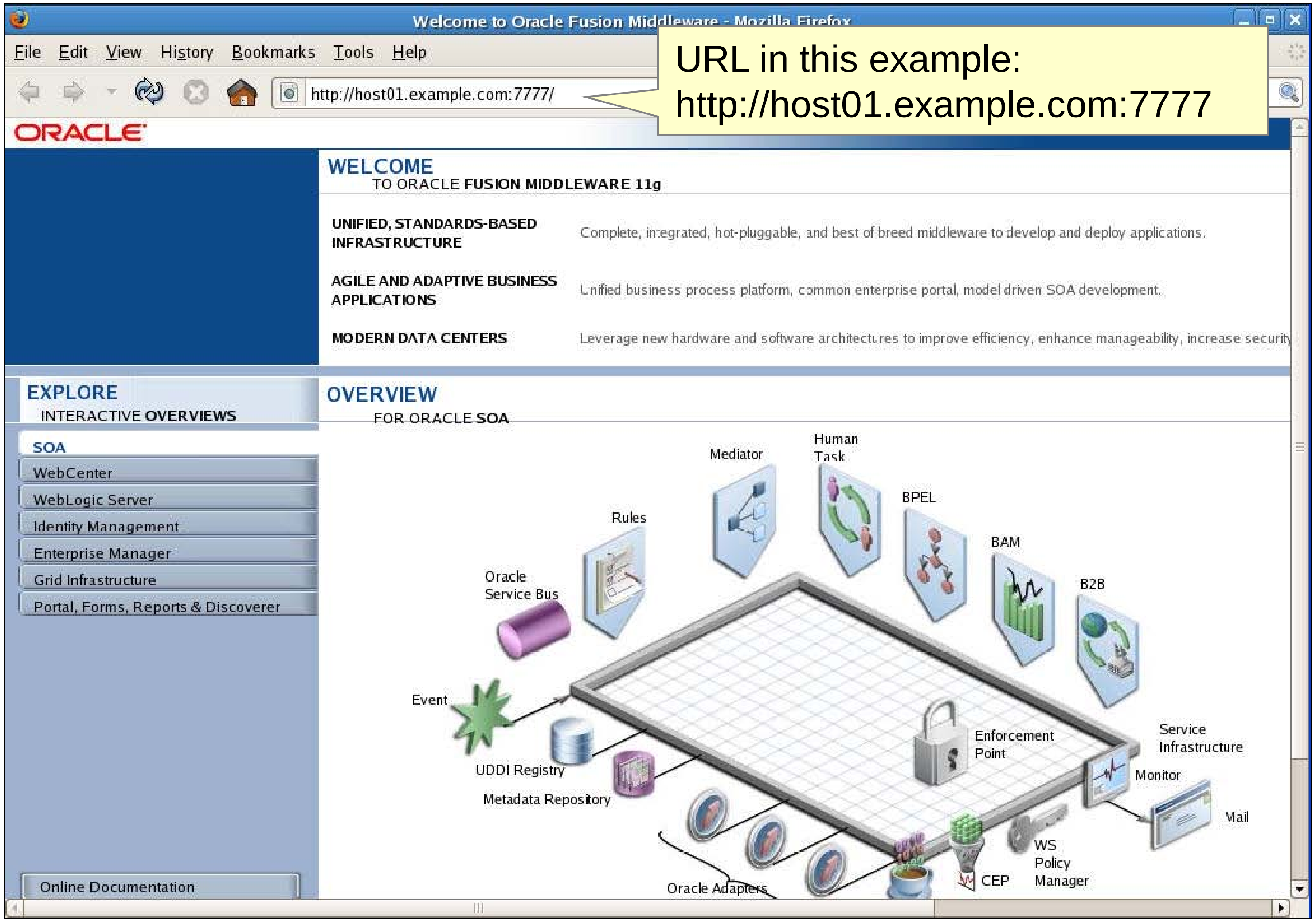
1. View the port on which OHS is running by using the `opmnctl status -l` command.

```
$> ./opmnctl status –l

Processes in Instance: webtier
---------+---------+-----+-------+-...+---------
ias-     |process-|     |       |    |
component|type    | pid |status|    | ports
---------+---------+-----+-------+-...+---------
ohs1     | OHS    | 2598|Alive |    | https:7779,
                                     | https:7778,
                                     | http:7777
```

2. In a web browser, enter the URL of the host name where OHS was started followed by the discovered HTTP port.

Successful Access of OHS Splash Page



URL in this example:
http://host01.example.com:7777

Welcome to Oracle Fusion Middleware - Mozilla Firefox

http://host01.example.com:7777/

WELCOME
TO ORACLE **FUSION MIDDLEWARE 11g**

UNIFIED, STANDARDS-BASED INFRASTRUCTURE Complete, integrated, hot-pluggable, and best of breed middleware to develop and deploy applications.

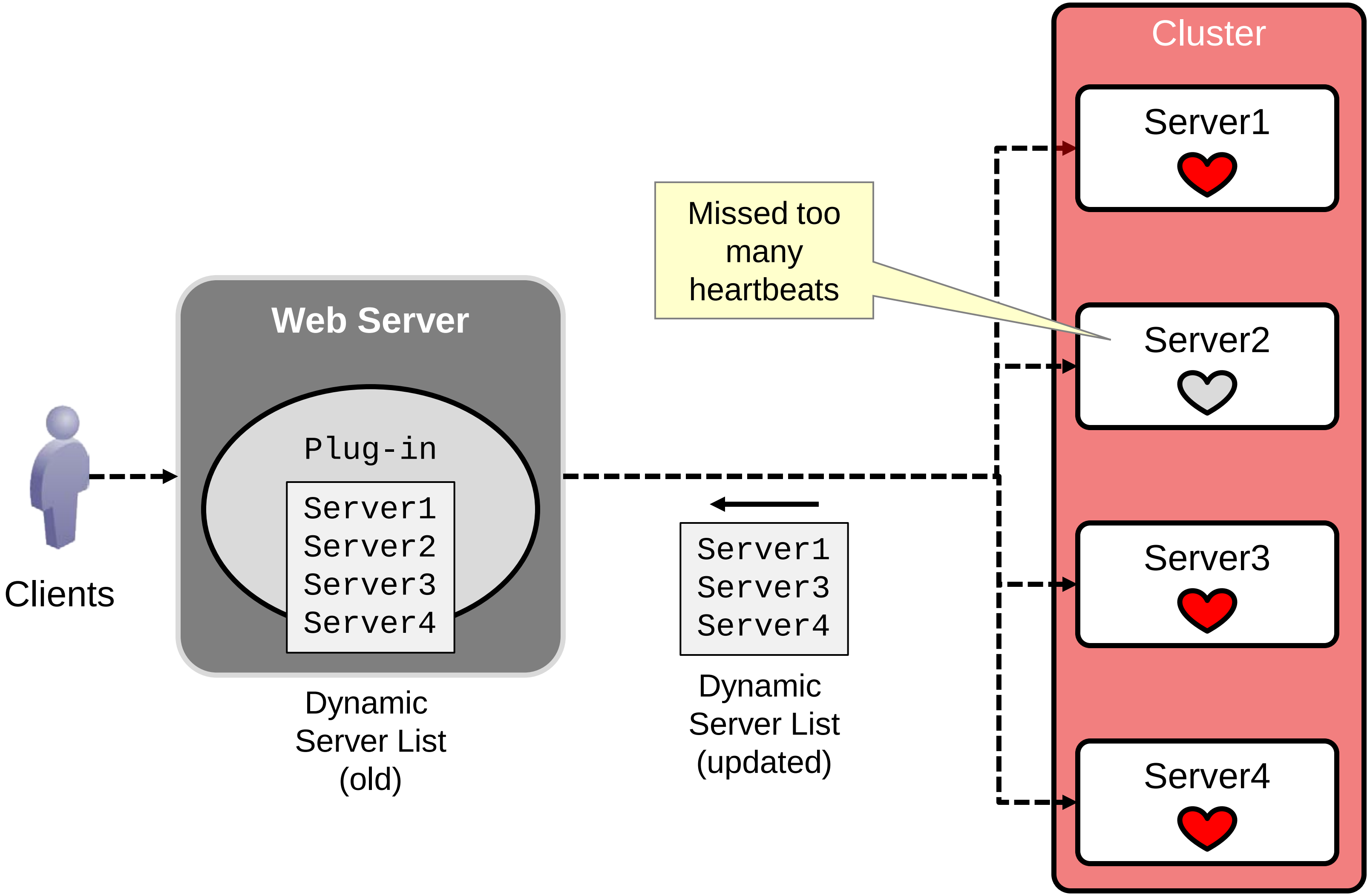
AGILE AND ADAPTIVE BUSINESS APPLICATIONS Unified business process platform, common enterprise portal, model driven SOA development.

MODERN DATA CENTERS Leverage new hardware and software architectures to improve efficiency, enhance manageability, increase security

EXPLORE
INTERACTIVE **OVERVIEWS**

OVERVIEW
FOR ORACLE **SOA**

Failover: Detecting Failures and the Dynamic Server List



Failover: Detecting Failures and the Dynamic Server List

- A clustered server detects the failure of another server in the cluster when:
  - A socket to that server unexpectedly closes
  - That server misses three* heartbeats
- In either case, that server is marked as “failed.”
- Responses from a clustered server to a cluster proxy include the “dynamic server list,” a list of all the current, viable servers in the cluster.
  - The list lets the proxy know which servers it can use.
  - The list is updated not only when servers fail, but also when new servers are added to the cluster.

* This number is configurable.

HTTP Session Failover

- Web applications store objects in HTTP sessions to track information for each client in memory.
- When an instance of WebLogic Server creates a session, it writes a cookie to the client's web browser indicating that it is the server for this client. Subsequent requests from that client are routed by the proxy to this server.
- If the server fails, its clients must be routed to other servers in the cluster, and session information is lost.
- WebLogic Server supports several strategies so that the session information is not lost when a server fails:
  - In-memory session replication
  - JDBC (database) session persistence
  - File session persistence

Recommended, as it is the fastest

Configuring Web Application Session Failover: `weblogic.xml`

- Developers configure sessions in `weblogic.xml`, under the `<session-descriptor>` tag.
- Its subtag, `<persistent-store-type>`, configures session failover:

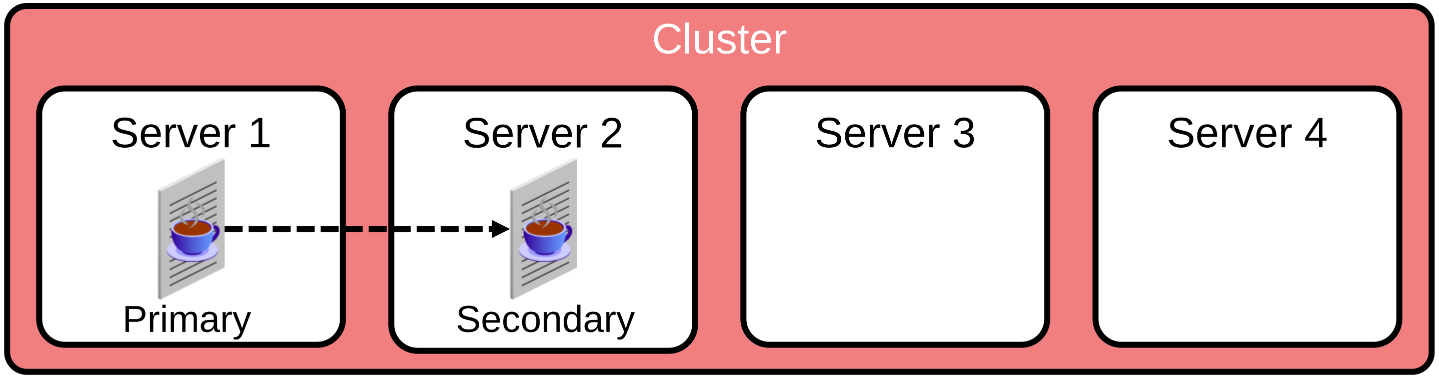
`<persistent-store-type>`	Description
`memory`	No session replication or persistence
`replicated`	In-memory session replication
`replicated_if_clustered`	The same as `memory` if deployed to stand-alone servers, the same as `replicated` if deployed to a cluster

Configuring Web Application Session Failover: `weblogic.xml`

`<persistent-store-type>`	Description
`async_replicated`	In-memory session replication with syncing done in batches
`async_replicated_if_clustered`	The same as `memory` if deployed to stand-alone servers, the same as `async-replicated` if deployed to a cluster
`file`	File-based persistence of sessions
`jdbc`	Database persistence of sessions
`async_jdbc`	Database persistence of sessions with updates done in batch
`cookie`	All session data stored in cookies

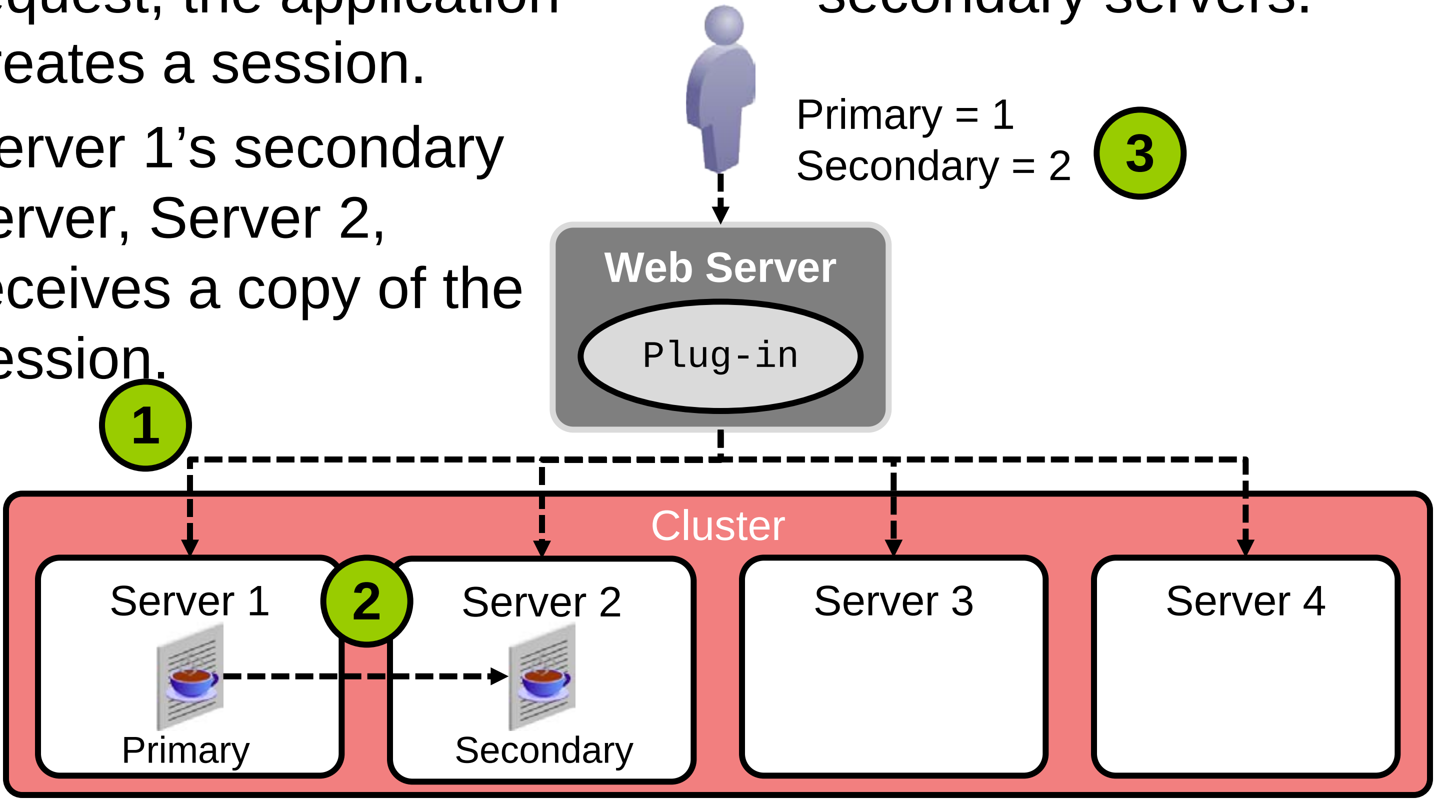
In-Memory Session Replication

- Each client’s session object exists on two servers:
  - Primary
  - Secondary
- The WebLogic Server session cookie stores both the client’s primary and secondary servers.
- Each update to the primary session object is automatically replicated to the secondary server, either synchronously (the default) or asynchronously (batch).



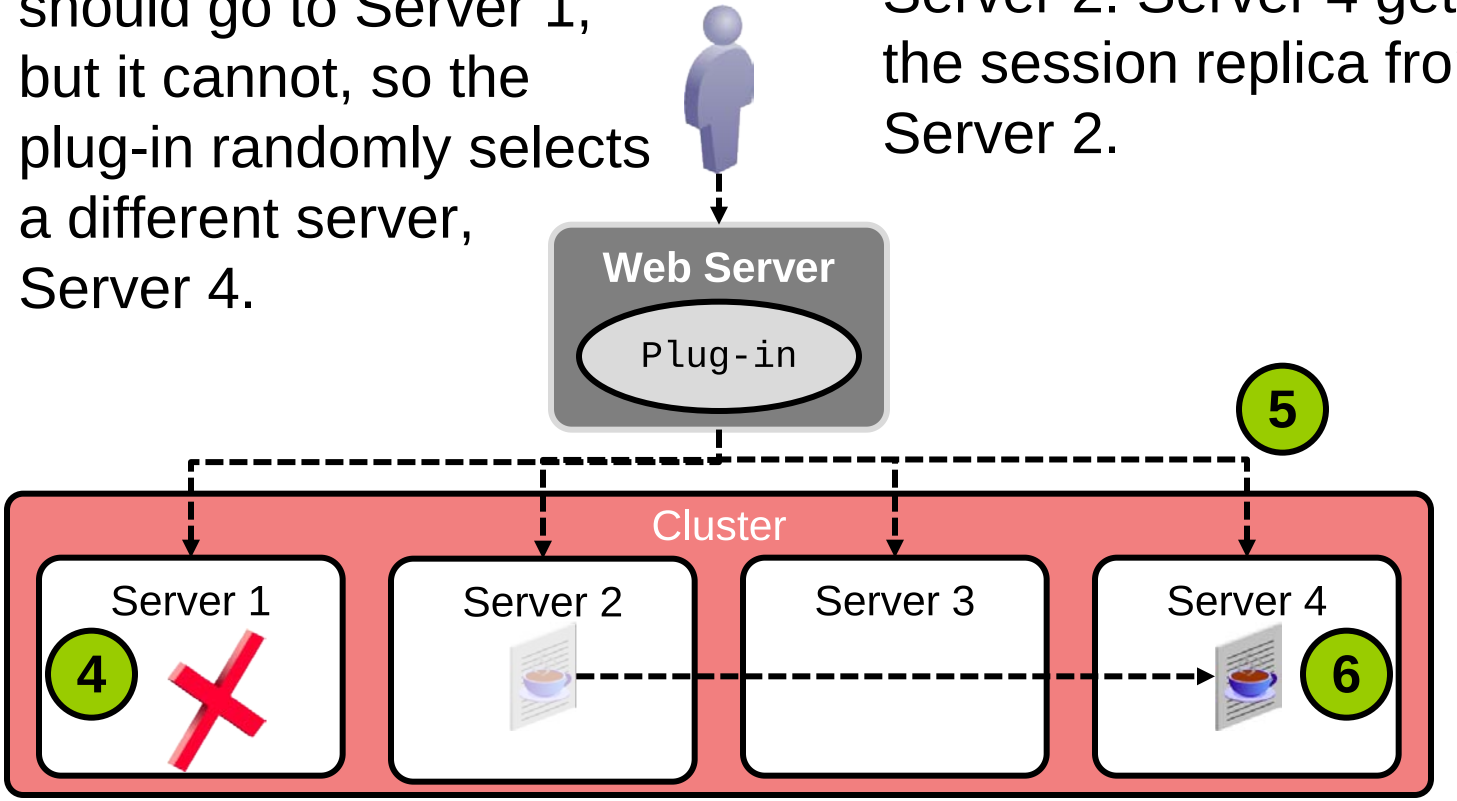
In-Memory Replication: Example

1. A client is load balanced to Server 1. On this request, the application creates a session.
2. Server 1's secondary server, Server 2, receives a copy of the session.
3. The cookie is written to track the primary and secondary servers.



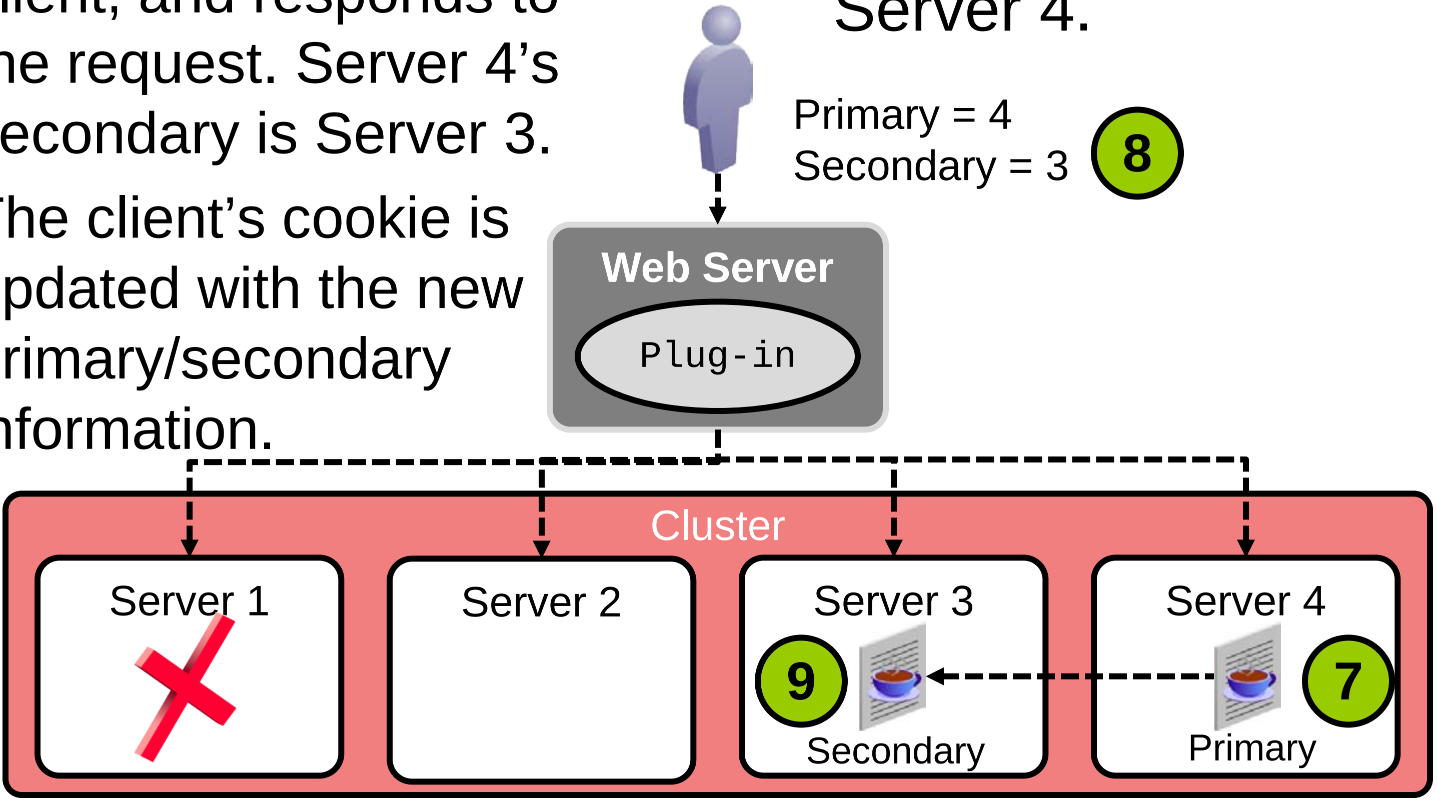
In-Memory Replication: Example

4. Server 1 fails.
5. The client’s next request should go to Server 1, but it cannot, so the plug-in randomly selects a different server, Server 4.
6. The client’s cookie stores the secondary server, Server 2. Server 4 gets the session replica from Server 2.



In-Memory Replication: Example

7. Server 4 is now the primary server for the client, and responds to the request. Server 4's secondary is Server 3.
8. The client's cookie is updated with the new primary/secondary information.
9. Server 3 stores the session replica of Server 4.



Configuring In-Memory Replication

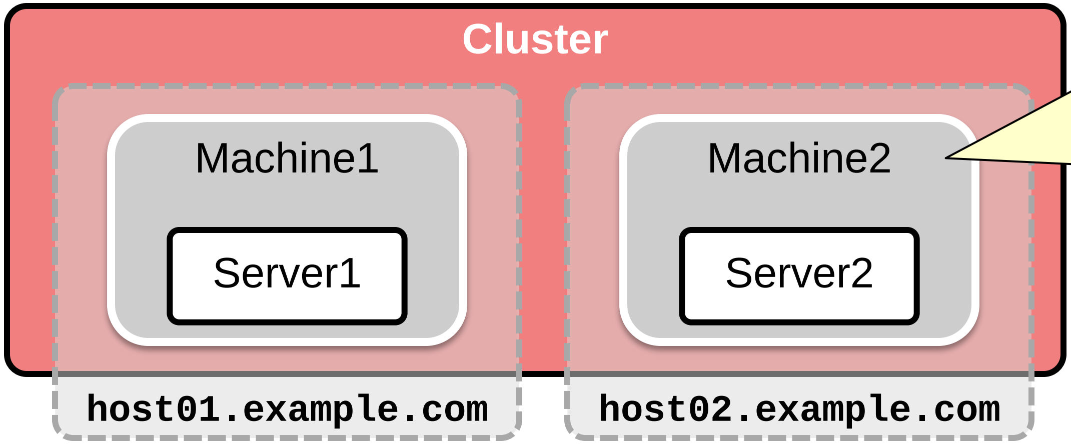
- Configure in-memory replication in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
   <persistent-store-type>replicated_if_clustered
   </persistent-store-type>
</session-descriptor>
...
```

weblogic.xml

Machines

- WebLogic Server uses machine definitions and the servers assigned to them to indicate which managed servers run on what hardware.
- WebLogic Server takes machine definitions into account when it chooses a secondary server as a backup for session information.
  - It prefers one on a different machine than the primary server.

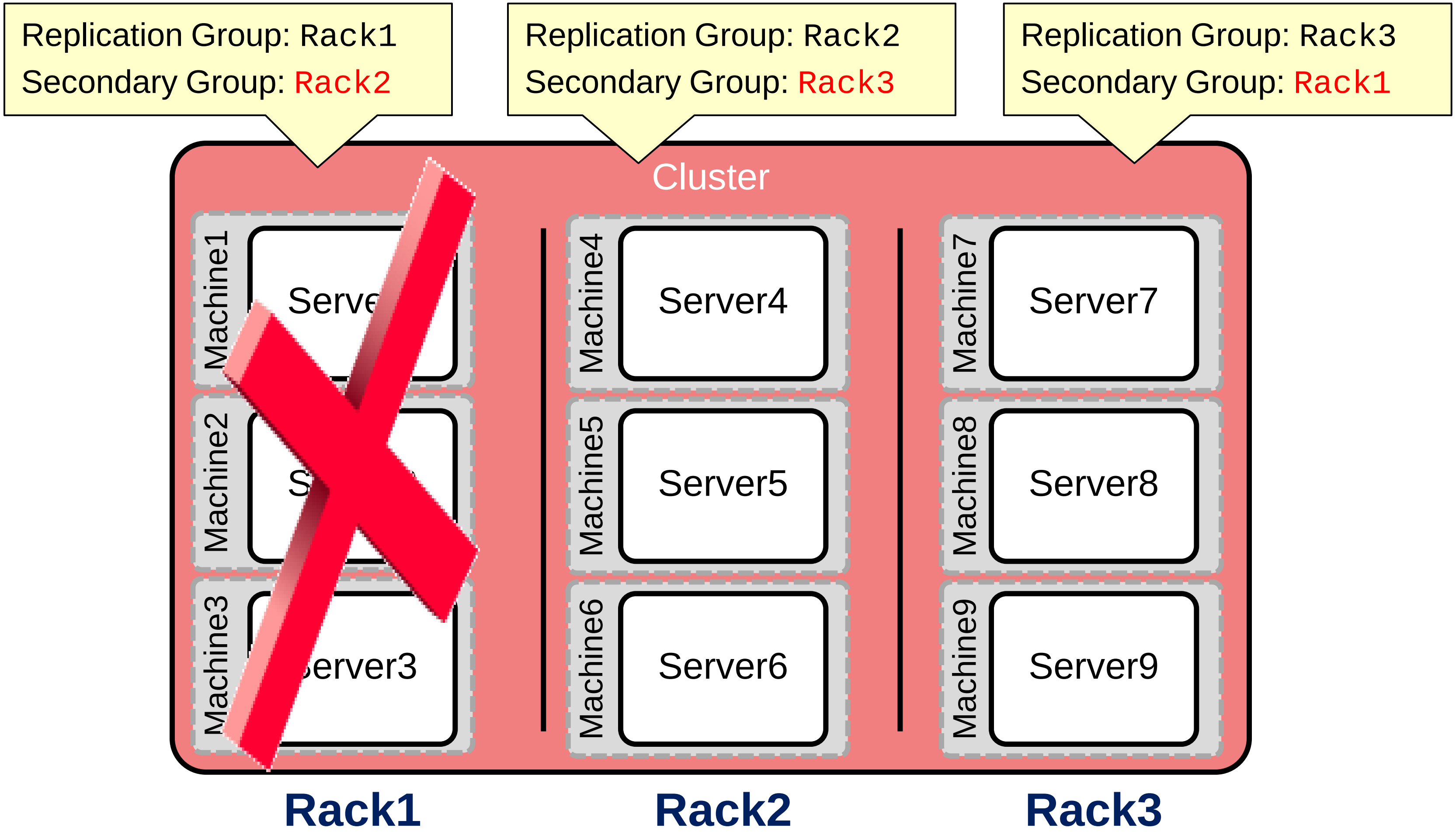


The machine is defined to represent hardware. Servers that run on that hardware are assigned to the machine.

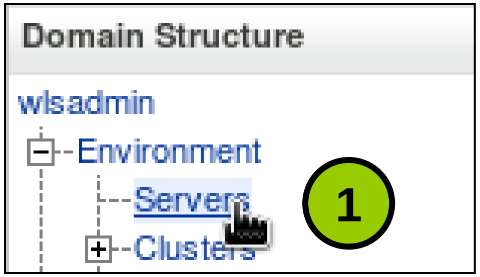
Secondary Server and Replication Groups

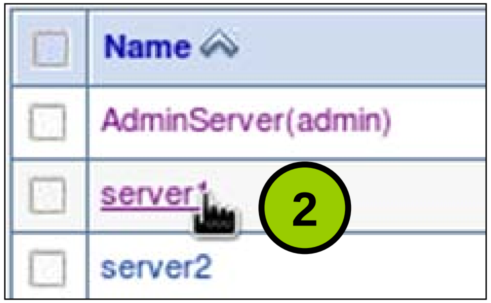
- A replication group is a logical grouping of servers in a cluster.
- WebLogic Server allows you to influence how secondary servers are chosen by configuring replication groups and configuring a server's "preferred secondary group."
- When choosing a secondary server, WebLogic Server attempts to:
  - Choose one in the primary server's preferred secondary group, if it is configured
  - Choose a server on a different machine
  - Avoid choosing a server in the same replication group

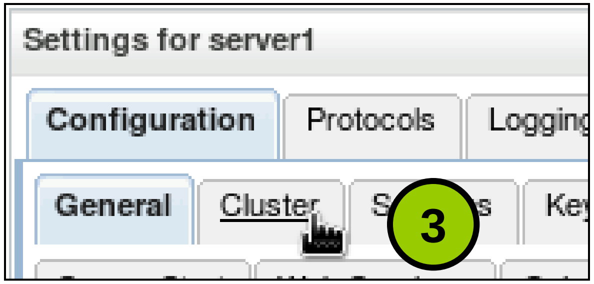
Replication Groups: Example

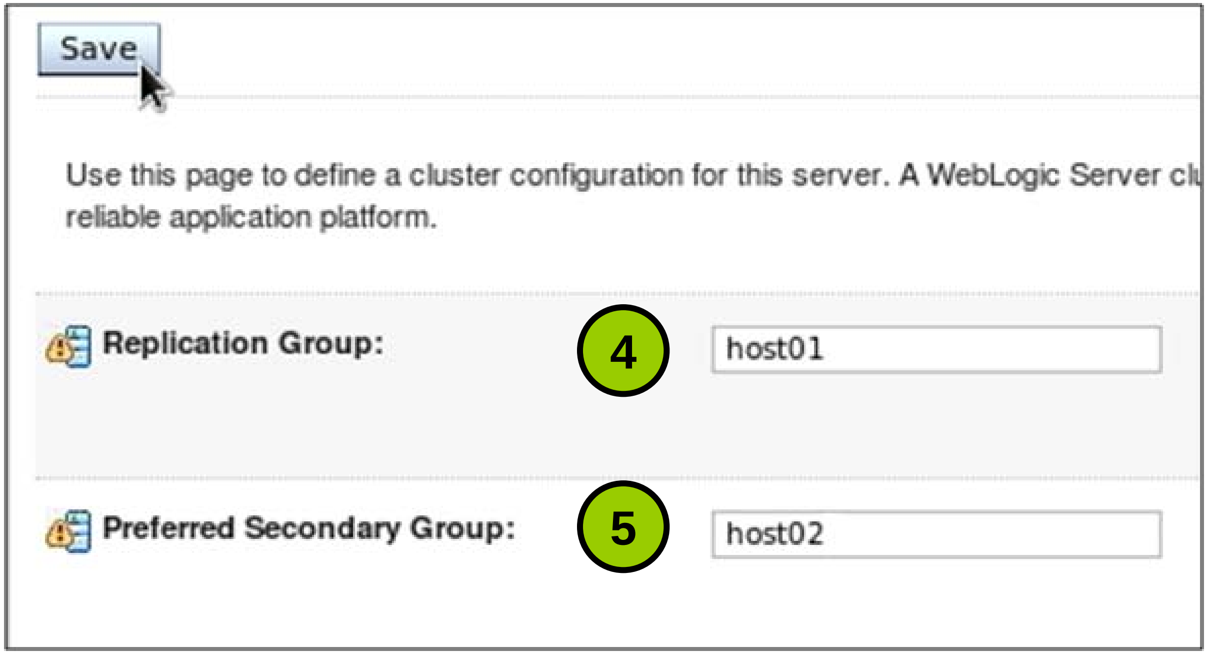


Configuring Replication Groups









Use this page to define a cluster configuration for this server. A WebLogic Server cl
reliable application platform.

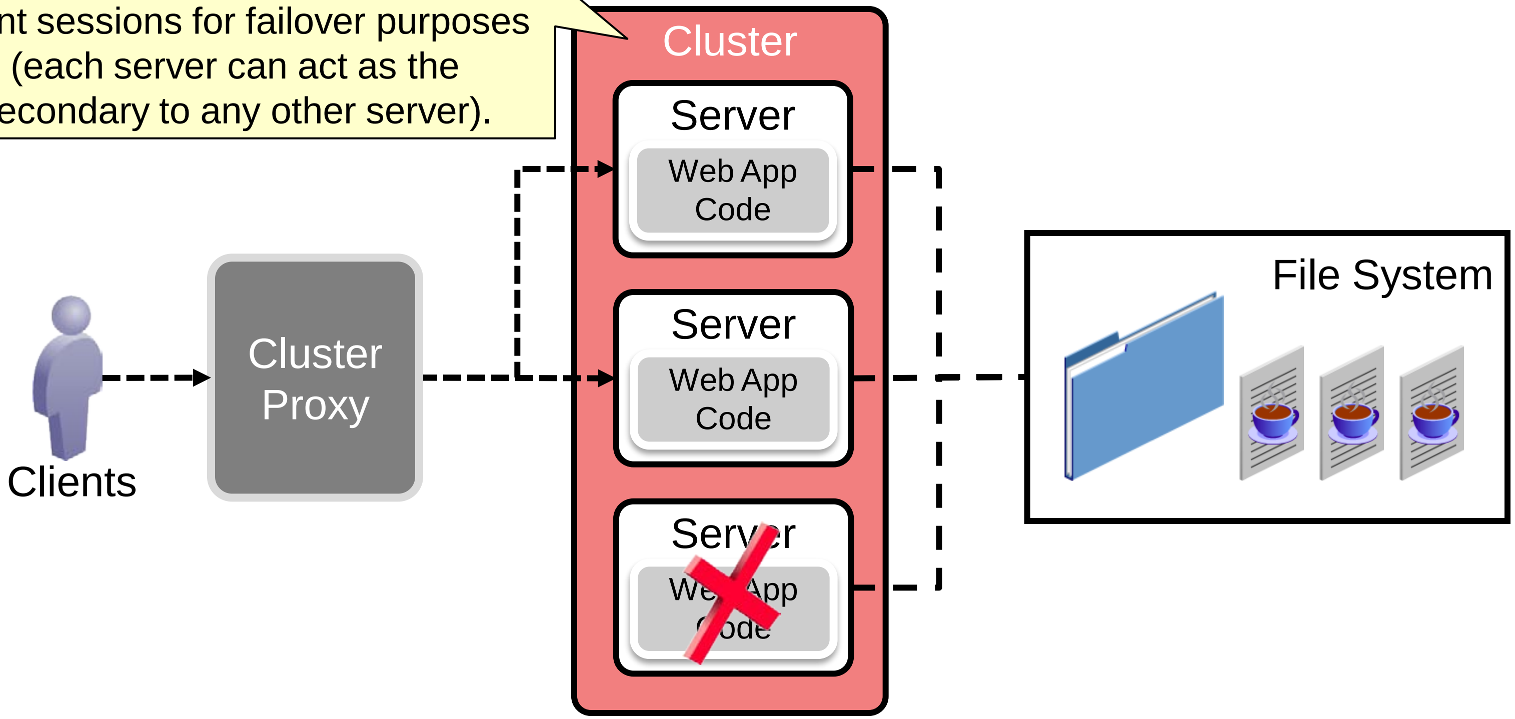
Replication Group: host01

Preferred Secondary Group: host02

File Session Persistence

File persistence stores session information in files to a highly available file system.

All members have access to any client sessions for failover purposes (each server can act as the secondary to any other server).



Configuring File Persistence

1. Create a folder shared by all servers on the cluster on a highly available file system.
2. Assign read/write privileges to the folder.
3. Configure file session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
   <persistent-store-type>file</persistent-store-type>
   <persistent-store-dir>/mnt/wls_share</persistent-store-dir>
</session-descriptor>
...
```

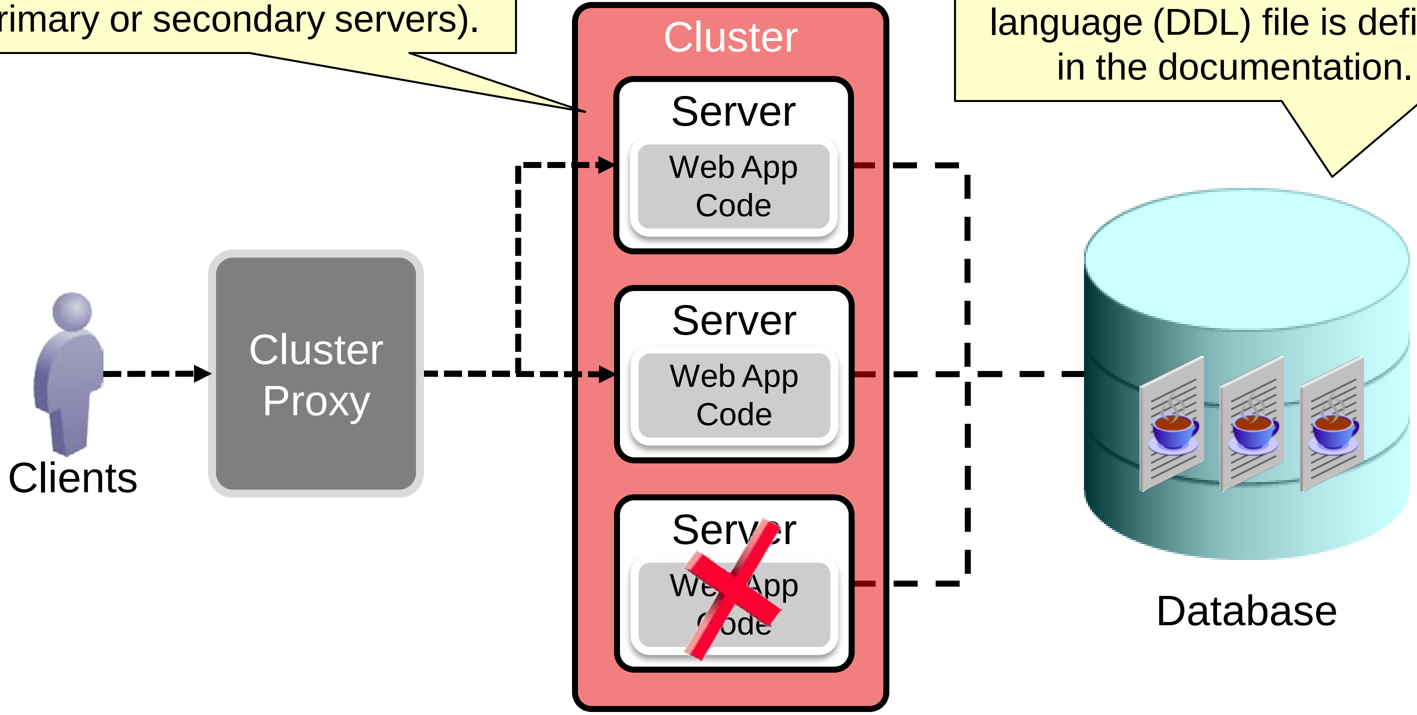
weblogic.xml

JDBC Session Persistence

HTTP sessions are persisted to a database using a common JDBC data source.

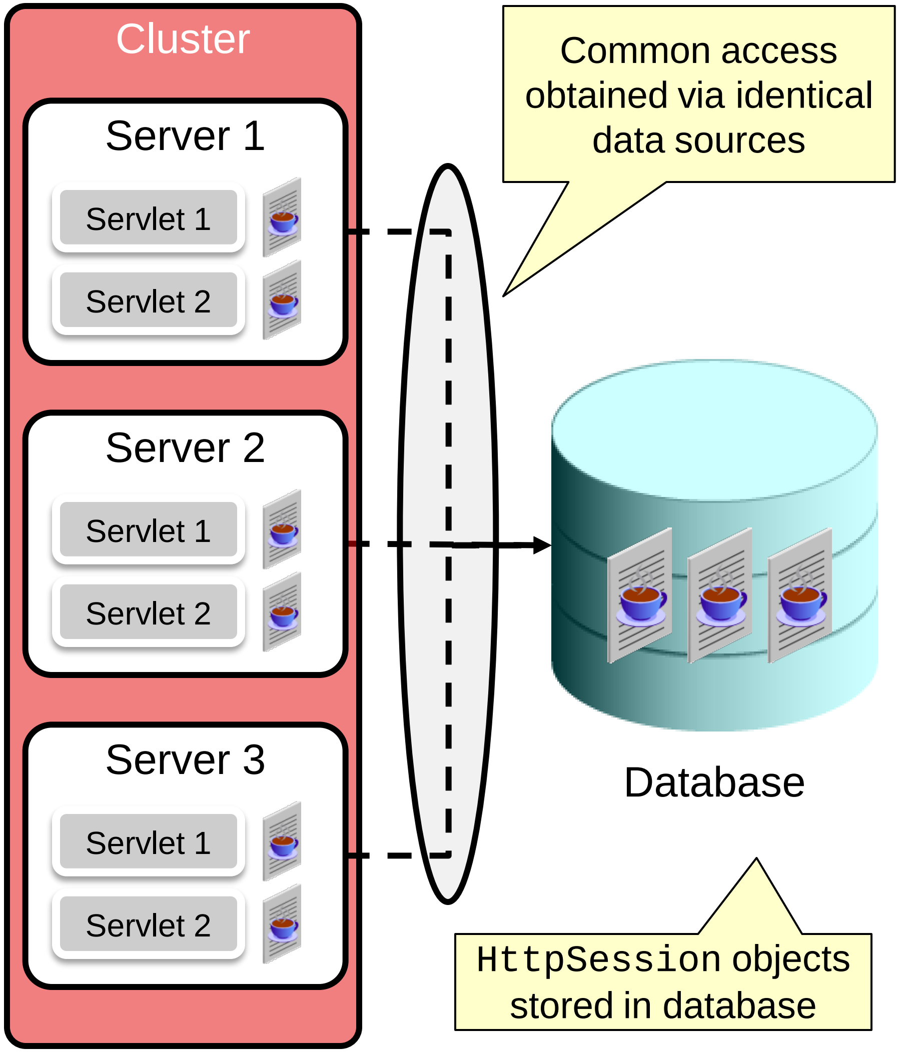
All members have access to any client sessions for failover purposes (no primary or secondary servers).

The required data definition language (DDL) file is defined in the documentation.



JDBC Session Persistence Architecture

- All server instances have access to all sessions.
- Subsequent requests from the same client can be handled by any server.
  - ✓ Great failover capability
  - ✗ Significant performance reduction
- Changing session objects causes (slow) database synchronization.



Common access obtained via identical data sources

`HttpSession` objects stored in database

Configuring JDBC Session Persistence

1. Create the required table in the database.
2. Create a JDBC data source that has read/write privileges for your database.
3. Configure JDBC session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
   <persistent-store-type>jdbc</persistent-store-type>
   <persistent-store-pool>mysessionds</persistent-store-pool>
</session-descriptor>
...
```

weblogic.xml

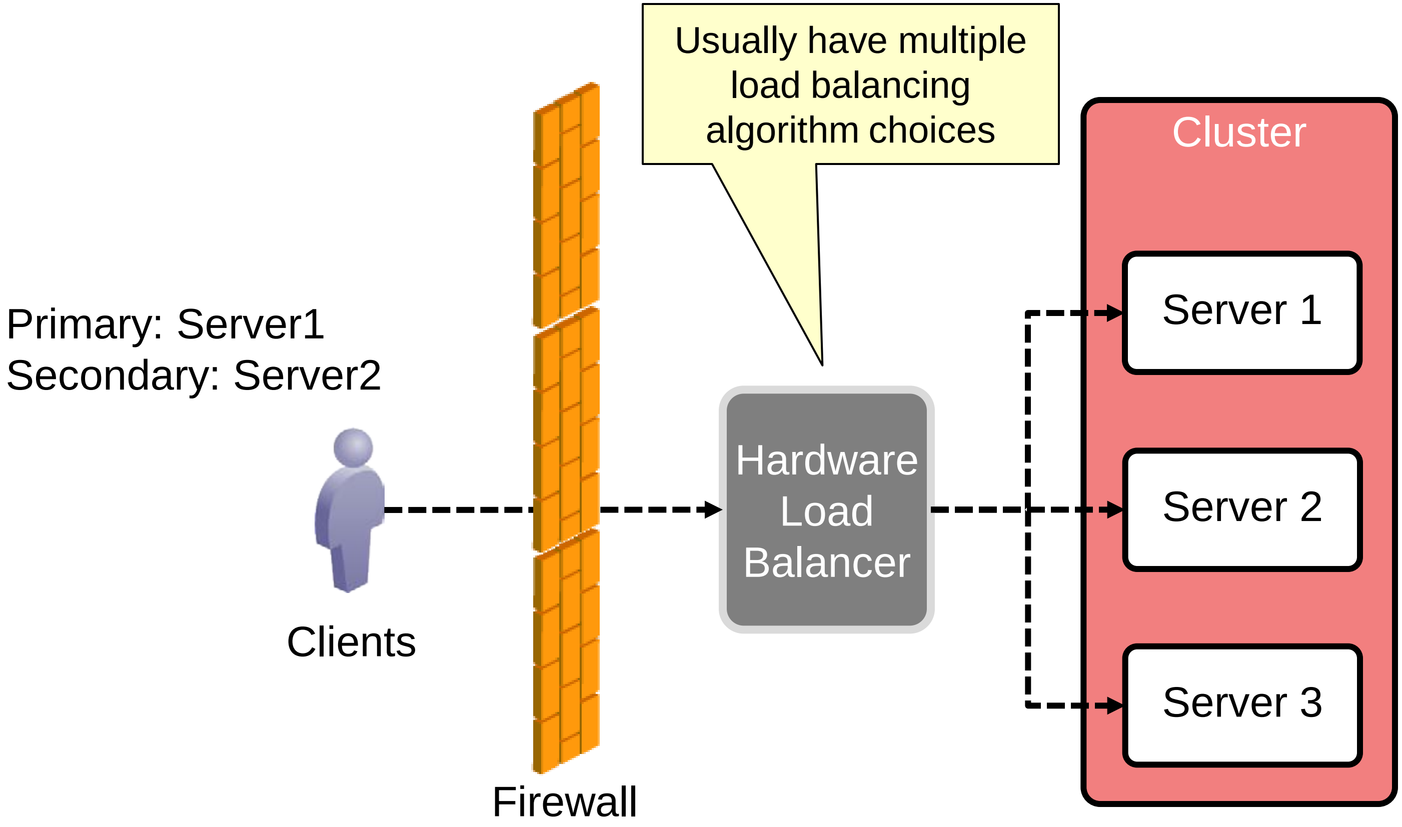
JDBC Persistent Table Configuration

The `WL_SERVLET_SESSIONS` table must exist with read/write access:

Prim. Key

Column Name	Column Data Type
WL_ID	VARCHAR(100)
WL_CONTEXT_PATH	VARCHAR(100)
WL_CREATE_TIME	NUMBER(20)
WL_IS_VALID	CHAR(1)
WL_SESSION_VALUES	BLOB
WL_ACCESS_TIME	NUMBER(20)
WL_IS_NEW	CHAR(1)
WL_MAX_INACTIVE_INTERVAL	INTEGER

Configuring a Hardware Load Balancer

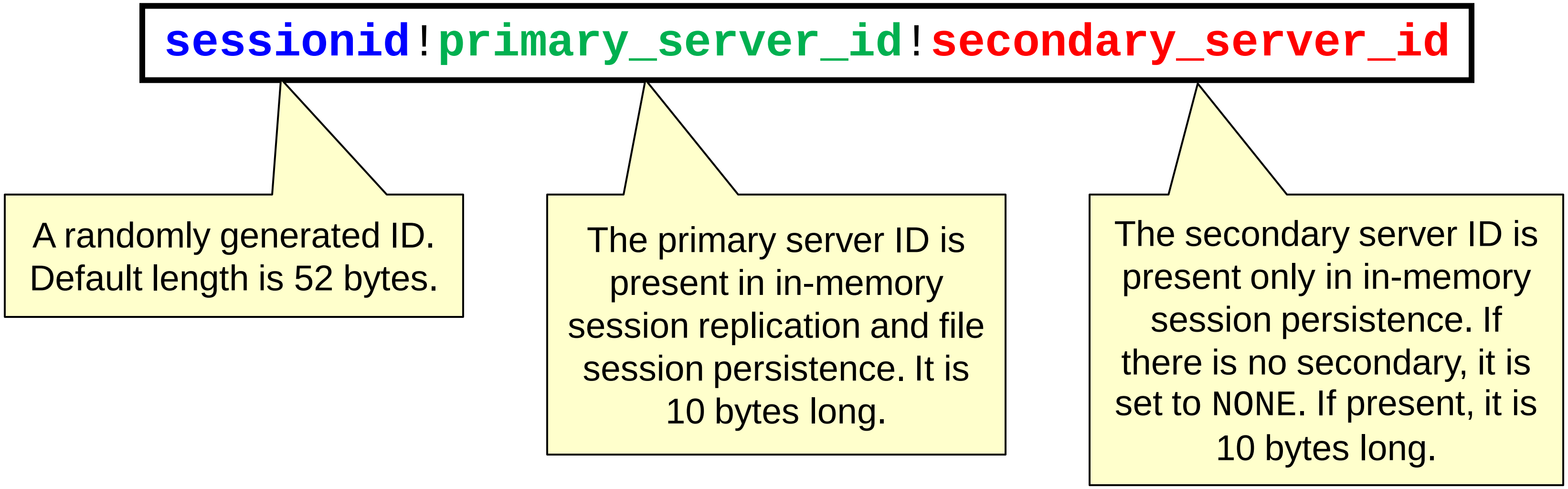


Hardware Load Balancer Session Persistence

- SSL Persistence
  - The load balancer performs all data encryption and decryption between clients and the WebLogic Server cluster.
  - The load balancer uses the plain text session cookie that WebLogic Server writes on the client to maintain an association between the client and the primary server
- Passive Cookie Persistence
  - The load balancer uses a string within the WebLogic Server session cookie to associate the client with the primary server. You must tell the load balancer where this string is.
- Active Cookie Persistence
  - If the load balancer creates it own cookie, and does not modify the WebLogic Server session cookie, this works without any additional configuration.

Passive Cookie Persistence and the WebLogic Server Session Cookie

- Configure a passive cookie load balancer:
  - Cookie name: `JSESSIONID`
  - Set the offset to 53 bytes (52 bytes for the session ID + 1 byte for the delimiter)
  - String length: `10` characters



```
sessionid!primary_server_id!secondary_server_id
```

A randomly generated ID. Default length is 52 bytes.

The primary server ID is present in in-memory session replication and file session persistence. It is 10 bytes long.

The secondary server ID is present only in in-memory session persistence. If there is no secondary, it is set to `NONE`. If present, it is 10 bytes long.

Quiz

In-memory session replication copies session data from one clustered instance of WebLogic Server to:

a. All other instances of WebLogic Server in the cluster

b. All instances of WebLogic Server in the Preferred Secondary Group

c. All instances of WebLogic Server in the same Replication Group

d. Another instance of WebLogic Server in the cluster

Summary

In this lesson, you should have learned how to:

- Install Oracle HTTP Server
- Configure Oracle HTTP Server as a cluster proxy
- Configure session failover
- Configure replication groups

Practice 13-1 Overview: Installing OHS (Optional)

This practice covers the following topics:

- Installing OHS from the Web Tier installer
- Creating an OHS instance

Practice 13-2 Overview: Configuring a Cluster Proxy

This practice covers the following topics:

- Configuring Oracle HTTP Server to act as a proxy to a WebLogic cluster
- Starting Oracle HTTP Server
- Testing in-memory session replication

Practice 13-3 Overview: Configuring Replication Groups

This practice covers configuring replication groups in a cluster.