◆ ◆ ◆ CHAPTER 7

Upgrading Applications Without Loss of Availability

Upgrading an application to a new version without loss of availability to users is called a *rolling upgrade*. Carefully managing the two versions of the application across the upgrade ensures that current users of the application complete their tasks without interruption, while new users transparently get the new version of the application. With a rolling upgrade, users are unaware that the upgrade occurs.

For more information about application versions and how they are identified, see "Module and Application Versions" in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

In a clustered environment, a rolling upgrade redeploys an application with a minimal loss of service and sessions. A session is any artifact that can be replicated, for example:

- `HttpSession`
- `SingleSignOn`
- `ServletTimer`
- `DialogFragment`
- Stateful session bean

A rolling upgrade can take place under light to moderate loads. The procedure requires about 10-15 minutes for each GlassFish Server instance.

Caution – To prevent the risk of version mismatch when a session fails over, upgrade all instances in a cluster at the same time. Otherwise a session might fail over to an instance where different versions of components are running.

Perform this task on each cluster separately. A cluster acts as a safe boundary for session failover for instances in the cluster. Sessions in one cluster can never fail over to sessions in another cluster. Therefore, the risk of version mismatch is avoided.

Application Compatibility

Rolling upgrades pose varying degrees of difficulty depending on the magnitude of changes between the two application versions.

If the changes are superficial, for example, changes to static text and images, the two versions of the application are *compatible* and can both run at once in the same cluster.

Compatible applications must:

- Use the same session information
- Use compatible database schemas
- Have generally compatible application-level business logic
- Use the same physical data source

You can perform a rolling upgrade of a compatible application in either a single cluster or multiple clusters. For more information, see "Upgrading In a Single Cluster" on page 14.

If the two versions of an application do not meet all the above criteria, then the applications are considered *incompatible*. Executing incompatible versions of an application in one cluster can corrupt application data and cause session failover to not function correctly. The problems depend on the type and extent of the incompatibility. It is good practice to upgrade an incompatible application by creating a "shadow cluster" to which to deploy the new version and slowly quiesce the old cluster and application. For more information, see "Upgrading Incompatible Applications" on page 17.

The application developer and administrator are the best people to determine whether application versions are compatible. If in doubt, assume that the versions are incompatible, since this is the safest approach.

Upgrading In a Single Cluster

You can perform a rolling upgrade of an application deployed to a single cluster, providing the cluster's configuration is not shared with any other cluster.

▼ To upgrade an application in a single cluster

1 Deploy the upgraded application to the cluster in a disabled state and with a new version identifier.

For example:

```
asadmin> asadmin deploy --enabled=false --target myCluster myApp:1.1
```

2 Enable the upgraded application for the instances using `asadmin enable-http-lb-application`.

3. **Quiesce one server instance in the cluster from the load balancer.**

   Follow these steps:

   a. **Disable the server instance using `asadmin disable-http-lb-server`.**

   b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

   c. **Copy the exported configuration file to the web server instance's configuration directory.**

      For example, for Sun Java System Web Server, the location is *web-server-install-dir*`/https-`*host-name*`/config/loadbalancer.xml`.

   d. **Wait until the timeout has expired.**

      Monitor the load balancer's log file.

4. **Enable the upgraded application version on the quiesced server instance.**

   For example:

   ```
   asadmin> asadmin enable --target instance01 myApp:1.1
   ```

   Enabling the upgraded application version automatically disables the previous version.

5. **Test the upgraded application on the server instance to make sure it runs correctly.**

6. **Re-enable the server instance in load balancer.**

   Follow these steps:

   a. **Enable the server instance using `asadmin enable-http-lb-server`.**

   b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

   c. **Copy the configuration file to the web server's configuration directory.**

7. **Repeat steps 3 through 6 for each instance in the cluster.**

Upgrading in Multiple Clusters

▼ To Upgrade a Compatible Application in Two or More Clusters:

1. **Deploy the upgraded application to one cluster in a disabled state and with a new version identifier.**

   For example:

   ```
   asadmin> asadmin deploy --enabled=false --target myCluster myApp:1.1
   ```

2. **Enable the upgraded application for the cluster using `asadmin enable-http-lb-application`.**

3. **Quiesce the cluster with the upgraded application from the load balancer.**

   a. **Disable the cluster using `asadmin disable-http-lb-server`.**

   b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

   c. **Copy the exported configuration file to the web server instance's configuration directory.**

      For example, for Sun Java System Web Server, the location is *web-server-install-dir*`/https-`*host-name*`/config/loadbalancer.xml`.

   d. **Wait until the timeout has expired.**

      Monitor the load balancer's log file.

4. **Enable the upgraded application version on the quiesced cluster.**

   For example:

   ```
   asadmin> asadmin enable --target myCluster myApp:1.1
   ```

   Enabling the upgraded application version automatically disables the previous version.

5. **Test the upgraded application on the cluster to make sure it runs correctly.**

6. **Enable the cluster in the load balancer:**

   a. **Enable the cluster using `asadmin enable-http-lb-server`.**

   b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

   c. **Copy the configuration file to the web server's configuration directory.**

7 **Repeat steps 1 through 6 for the other clusters.**

Upgrading Incompatible Applications

If the new version of the application is incompatible with the old version, use the following procedure. For information on what makes applications compatible, see "Application Compatibility" on page 14. Also, you must upgrade incompatible application in two or more clusters. If you have only one cluster, create a "shadow cluster" for the upgrade, as described below.

When upgrading an incompatible application:

- Give the new version of the application a different version identifier from the old version of the application. The steps below assume that the application has a new version identifier.
- If the data schemas are incompatible, use different physical data sources after planning for data migration.
- Deploy the new version to a different cluster from the cluster where the old version is deployed.
- Set an appropriately long timeout for the cluster running the old application before you take it offline, because the requests for the application won't fail over to the new cluster. These user sessions will simply fail.

▼ To Upgrade an Incompatible Application by Creating a Second Cluster

1 **Create a "shadow cluster" on the same or a different set of machines as the existing cluster. If you already have a second cluster, skip this step.**

   a. **Use the Admin Console to create the new cluster and reference the existing cluster's named configuration.**

      Customize the ports for the new instances on each machine to avoid conflict with existing active ports.

   b. **For all resources associated with the cluster, add a resource reference to the newly created cluster using `asadmin create-resource-ref`.**

   c. **Create a reference to all other applications deployed to the cluster (except the current upgraded application) from the newly created cluster using `asadmin create-application-ref`.**

   d. **Configure the cluster to be highly available using `asadmin configure-ha-cluster`.**

e. **Create reference to the newly-created cluster in the load balancer configuration file using `asadmin create-http-lb-ref`.**

2 **Give the new version of the application a different version identifier from the old version.**

3 **Deploy the new application version with the new cluster as the target. Use a different context root or roots.**

4 **Enable the deployed new application for the clusters using `asadmin enable-http-lb-application`.**

5 **Start the new cluster while the other cluster is still running.**

The start causes the cluster to synchronize with the domain and be updated with the new application.

6 **Test the application on the new cluster to make sure it runs correctly.**

7 **Disable the old cluster from the load balancer using `asadmin disable-http-lb-server`.**

8 **Set a timeout for how long lingering sessions survive.**

9 **Enable the new cluster from the load balancer using `asadmin enable-http-lb-server`.**

10 **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

11 **Copy the exported configuration file to the web server instance's configuration directory.**

For example, for Sun Java System Web Server, the location is *web-server-install-dir*`/https-`*host-name*`/config/loadbalancer.xml`.

12 **After the timeout period expires or after all users of the old application have exited, stop the old cluster and undeploy the old application version.**