◆ ◆ ◆ CHAPTER 7

Configuring HTTP Load Balancing

This chapter describes how to configure HTTP load balancing on GlassFish Server 3.1.

The following topics are addressed here:

- "Setting Up HTTP Load Balancing" on page 125

For information on other types of load balancing, see Chapter 10, "Java Message Service Load Balancing and Failover," and Chapter 11, "RMI-IIOP Load Balancing and Failover."

Setting Up HTTP Load Balancing

This section describes how to set up load balancing for GlassFish Server.

The following topics are addressed here:

- "Prerequisites for Setting Up HTTP Load Balancing" on page 125
- "Configuring GlassFish Server with Apache HTTP Server and `mod_jk`" on page 126
- "HTTP Load Balancer Deployments" on page 128

Prerequisites for Setting Up HTTP Load Balancing

Before configuring your load balancer, you must:

- Install a supported web server and configure it. If using the `mod_jk` module, the only supported web server is Apache HTTP Server 2.2.*x*.
- Configure the `mod_jk` connector module, as described in "Configuring GlassFish Server with Apache HTTP Server and `mod_jk`" on page 126.
- Create GlassFish Server clusters or server instances to participate in load balancing.
- Deploy applications to these clusters or instances.

Configuring GlassFish Server with Apache HTTP Server and `mod_jk`

GlassFish Server3.1 can be configured for load balancing with Apache HTTP Server as a front end by enabling the Apache `mod_jk` connector module. To enable the `mod_jk` module in GlassFish Server, set the GlassFish Server `jk-enablednetwork-listener` attribute. You can also create `jk-connectors` under different virtual-servers using the `jk-enablednetwork-listener` attribute.

▼ To Configure the `mod_jk` Connector Module

1. **Install Apache HTTP Server and mod_jk.**

2. **Configure `workers.properties` and `httpd.conf`.**

   For example:

   - `apache2/config/workers.properties`

     ```
     # Define 1 real worker using ajp13
     worker.list=worker1
     # Set properties for worker1 (ajp13)
     worker.worker1.type=ajp13
     worker.worker1.host=localhost
     worker.worker1.port=8009
     ```

   - `apache2/conf/httpd.conf`

     ```
     LoadModule jk_module /Users/Amy/apache2/modules/mod_jk-1.2.25-httpd-2.2.4.so
     JkWorkersFile /Users/Amy/apache2/conf/worker.properties
     # Where to put jk logs
     JkLogFile /Users/Amy/apache2/logs/mod_jk.log
     # Set the jk log level [debug/error/info]
     JkLogLevel debug
     # Select the log format
     JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
     # JkOptions indicate to send SSL KEY SIZE,
     JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
     # JkRequestLogFormat set the request format
     JkRequestLogFormat "%w %V %T"
     # Send everything for context /examples to worker named worker1 (ajp13)
     JkMount /examples/* worker1
     ```

3. **Start Apache HTTP Server.**

4. **Enable `mod_jk` using the following commands.**

   Be sure to enter each of these commands on a single line.

   ```
   asadmin> create-network-listener --protocol http-listener-1 \
   --listenerport 8009 --jkenabled true jk-connector

   asadmin> set server-config.network-config.network-listeners.network-listener.\
   jk-connector.jk-configuration-file=domain-dir/config/glassfish-jk.properties
   ```

5 **If you are using the `glassfish-jk.properties` file and not referencing it in `httpd.conf`, point to it using the following command:**

```
asadmin create-jvm-options -Dcom.sun.enterprise.web.connector.enableJK.\
propertyFile=domain-dir/config/glassfish-jk.properties
```

6 **Restart GlassFish Server.**

Example 7–1 `httpd.conf` File for Load Balancing

This example shows an `httpd.conf` file that is set for load balancing.

```
LoadModule jk_module /usr/lib/httpd/modules/mod_jk.so
JkWorkersFile /etc/httpd/conf/worker.properties
# Where to put jk logs
JkLogFile /var/log/httpd/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel debug
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
JkRequestLogFormat "%w %V %T"
# Send all jsp requests to GlassFish
JkMount /*.jsp worker1
# Send all glassfish-test requests to GlassFish
JkMount /glassfish-test/* loadbalancer
```

Example 7–2 `workers.properties` File for Load Balancing

This example shows a `workers.properties` or `glassfish-jk.properties` file that is set for load balancing. The `worker.worker*.port` should match with JK ports you created.

```
worker.list=worker1,worker2,loadbalancer
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=1
worker.worker1.socket_keepalive=1
worker.worker1.socket_timeout=300
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8010
worker.worker2.lbfactor=1
worker.worker2.socket_keepalive=1
worker.worker2.socket_timeout=300
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=worker1,worker2
```

HTTP Load Balancer Deployments

You can configure your load balancer in different ways, depending on your goals and environment, as described in the following sections:

Using Clustered Server Instances

The most common way to deploy the load balancer is with a cluster or clusters of server instances. By default all the instances in a cluster have the same configuration and the same applications deployed to them. The load balancer distributes the workload between the server instances and requests fail over from an unhealthy instance to a healthy one. If you've configured HTTP session persistence, session information persists when the request is failed over.

If you have multiple clusters, requests can be load balanced across clusters but are only failed over between the instances in a single cluster. Use multiple clusters in a load balancer to easily enable rolling upgrades of applications. For more information, see Chapter 8, "Upgrading Applications Without Loss of Availability."

Note – Requests cannot be load balanced across clusters and standalone instances.

Using Multiple Standalone Instances

It is also possible to configure your load balancer to use multiple standalone instances, and load balance and failover requests between them. However, in this configuration, you must manually ensure that the standalone instances have homogenous environments and the same applications deployed to them. Because clusters automatically maintain a homogenous environment, for most situations it is better and easier to use clusters.