# UsingMod_jk1.2WithJBoss

## Using mod_jk 1.2.x with JBoss/Tomcat bundle and Apache2

### Quick Overview

1. Download Apache2
2. Download modjk 1.2.x (At least 1.2.27 suggested)
3. Change the main Apache config to include modjk config
4. Create the modjk config
5. Configure the modjk workers (which JBoss/Tomcat nodes Apache uses)
6. Configure the Apache URIs served by modjk (the applications served by JBoss/Tomcat)
7. Restart Apache
8. Configure Tomcat (Give each JBoss/Tomcat a jvmRoute for session stickness)
9. Restart JBoss
10. Test it

### mod_proxy

Most httpd-2.2.x actual distributions (x>=6) have a decent AJP proxying and don't require to compile an external module. See http://wiki.jboss.org/wiki/UsingMod_proxyWithJBoss for more information.

### More Details

This wiki outlines the various steps required to install a basic load-balancing solution based on JBoss/Tomcat and mod_jk 1.2.

**Step 1**: Download Apache2 Web Server

Get the latest Apache2 package from Apache.org and install it. We require no special configuration, just use the default settings.

In the following steps, APACHE_HOME will represent the Apache install directory.

**Step 2**: Download mod_jk 1.2.x

Download the latest package available from Tomcats's 'Download Tomcat connector section' page . **Always download the latest stable release if possible**.

Rename the lib **mod_jk.so** and drop it in APACHE_HOME/modules directory.

**NOTE**: Don't use any release prior to mod_jk 1.2.15. Earlier releases are fairly buggy.

+

**Note:** Darwin Ports supports the installation of mod_jk on OS X. See http://darwinports.opendarwin.org/ for more info.

+

**Step 3**: Setup Apache to use modjk

Add this line at the very bottom in APACHE_HOME/conf/httpd.conf :

```
# Include mod_jk configuration file
```

```
Include conf/mod-jk.conf
```

**Step 4**: Create the modjk config

Under APACHE_HOME/conf, create **mod-jk.conf** and populate it as follows:

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
# Notes:
# 1) Changed from +ForwardURICompat.
# 2) For mod_rewrite compatibility, use +ForwardURIProxy (default since 1.2.24)
# See http://tomcat.apache.org/security-jk.html
JkOptions +ForwardKeySize +ForwardURICompatUnparsed -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /__application__/* loadbalancer
# Let Apache serve the images
JkUnMount /__application__/images/* loadbalancer

# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
# Note: Replaced JkShmFile logs/jk.shm due to SELinux issues. Refer to
# https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=225452
```

```
JkShmFile run/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus>
JkMount status
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

mod_jk is ready to forward requests to JBoss instances. We need now to setup the workers

**Note**: As of mod_jk 1.2.6+ you need to include "JkMountCopy all" in globals if you intend to specify global JkMount's or JkMountFile's instead of per VirtualHost. If you do not want to copy the same JkMount/JkMountFile for each VirtualHost, you can specify "JkMountCopy On" inside the VirtualHost directive.

See entry in: http://tomcat.apache.org/connectors-doc/reference/apache.html

-

---

**Step 5**: Configuring workers

Under APACHE_HOME/conf, create **workers.properties** and populate it as follows:

```
# Define list of workers that will be used
# for mapping requests
# The configuration directives are valid
# for the mod_jk version 1.2.18 and later
#
worker.list=loadbalancer,status


# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
```

---

```
worker.node1.lbfactor=1
worker.node1.socket_timeout=10
worker.node1.prepost_timeout=10000 #Not required if using ping_mode=A
worker.node1.connect_timeout=10000 #Not required if using ping_mode=A
worker.node1.ping_mode=A #As of mod_jk 1.2.27
# worker.node1.connection_pool_size=10 (1)

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host= node2.mydomain.com
worker.node2.type=ajp13
worker.node2.lbfactor=1
worker.node2.socket_timeout=10
worker.node2.prepost_timeout=10000 #Not required if using ping_mode=A
worker.node2.connect_timeout=10000 #Not required if using ping_mode=A
worker.node2.ping_mode=A #As of mod_jk 1.2.27
# worker.node1.connection_pool_size=10 (1)

# Load-balancing behaviour
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2

# Status worker for managing load balancer
worker.status.type=status
```

*Important*: Please review http://tomcat.apache.org/connectors-doc/reference/workers.html for the directive descriptions. Especially lookout for the comments on cachsize for Apache 1.3.x.

**(1) You should only set the connection_pool_size if the number of allowed connection to the Httpd is higher than maxThreads in server.xml**

If you specify worker.loadbalancer.sticky_session=Off, each request will be load balanced between node1 and node2. But when a user opens a Session on one server, it is a good idea to always forward this user's requests to the same server. Otherwise the user's session data would need to be synchronized between both servers. This is called a "sticky session", as the client is always using the same server he reached on his first request.

Session stickiness is enabled by default.

*Side Note*: a non-loadbalanced setup with a single node required the "worker.list=node1" entry before mod_jk would function correctly. Without this setting I would only get a 500 error and no other useful messages in log or otherwise. -Harlequin516

*Side Note*: I tried both loadbalanced and single node methods on Fedora 4. Both setups causing jk.shm errno=13 and jk-runtime-status errno=13 in the mod_jk.log. Could only get 500 errors. As a last resort disabled selinux on apache server. Restarted service and connection was made first try. -paulbrown

**Step 6:** Create the URI to worker map file

Create a **uriworkermap.properties** file in the APACHE_HOME/conf directory. This file should contain the URL mappings you want Apache to forward to Tomcat. The format of the file is /url=worker_name. To get things started, paste this example into the file you created:

```
# Simple worker configuration file
#

# Mount the Servlet context to the ajp13 worker
/jmx-console=loadbalancer
/jmx-console/*=loadbalancer
/web-console=loadbalancer
/web-console/*=loadbalancer
/myapp/*=loadbalancer
!/myapp/images/*=loadbalancer
```

This will configure mod_jk to forward requests for the /jmx-console, /web-console and /myapp contexts to JBoss Web. The '!' at the beginning of the last line results in the URLs for the images dir in the myapp context not being forwarded. Instead httpd will handle them directly (which means they must be available on the httpd server).

**Step 7:** Restart Apache

**Step 8:** Configure Tomcat

To complete the configuration, we also need to name each node to match the names specified in workers.properties.

To do this, edit the server.xml file. Where server.xml is located depends on the version of JBoss AS:

- In JBoss 5, it's $JBOSS_HOME/server/all/deploy/jbossweb.sar/server.xml
- In JBoss 4.2.x and EAP 4.x, it's $JBOSS_HOME/server/all/deploy/jboss-web.deployer/server.xml
- In earlier releases it's $JBOSS_HOME/server/all/deploy/jbossweb-tomcatXX.sar/server.xml where XX is 40, 50, 55 etc depending on the Tomcat version embedded in the AS.

(In the examples above, replace /all/ with the name of the AS configuration you are running.)

Locate the <Engine/.> element and add an attribute **jvmRoute**:

```
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="node1">
.
</Engine>
```

The jvmRoute attribute value must match the name specified in workers.properties.

In the server.xml file, make sure that the AJP 1.3 Connector is uncommented, e.g.:

```
<!-- A AJP 1.3 Connector on port 8009 -->
<Connector port="8009" address="${jboss.bind.address}"
    emptySessionPath="true" enableLookups="false" redirectPort="8443"
    protocol="AJP/1.3" connectionTimeout="600000"></Connector>
```

If you are only accepting requests via mod_jk, you can comment out the regular HTTP Connector; Tomcat then won't listen on port 8080.

**Step 9:** Activate the JvmRouteValve in JBoss (not needed with Tomcat Standalone)

Finally, we need to tell JBoss to add a special valve that detects when failover of a session from a distributable webapp has occurred. This JvmRouteValve ensures a new session cookie is emitted that includes the jvmRoute of the server that is now handling the session.

This configuration step is only needed with JBoss 4.2.x and earlier; beginning with JBoss AS 5 the application server uses the existence of a jvmRoute configuration in server.xml as an indication that it should add the JvmRouteValve.

To do this, edit the jboss-service.xml file for the JBoss Web service. Where this is located depends on the version of JBoss AS:

- In JBoss 5, this step isn't needed.
- In JBoss 4.2.x and EAP 4.x, it's $JBOSS_HOME/server/all/deploy/jboss-web.deployer/META-INF/jboss-service.xml
- In earlier releases it's $JBOSS_HOME/server/all/deploy/jbossweb-tomcatXX.sar/META-INF/jboss-service.xml where XX is 40, 50, 55 etc depending on the Tomcat version embedded in the AS.

Locate the <attribute> element with a name of UseJK, and set its value to "true":

```
<attribute name="UseJK">true</attribute>
```

**Step 10:** Restart JBoss AS.

**Step 11:** Access the JBoss AS web-console through Apache by browsing to http://localhost/web-console and you should see the JBoss web console page.

-

---

**Note**: to use mod_jk with Jboss 2 (e.g. jboss 2.4.6), you must edit jboss.jcml. Add the 'jvmRoute="myWorker"' to the Engine element under the EmbeddedCatalinaSX mbean.

-

---

-

---

**Note**: You may need to use VirtualHost in you mod-jk.conf. For example:

Instead of just

```
JkMount /jmx-console   loadbalancer
JkMount /jmx-console/* loadbalancer
```

try

```
<VirtualHost host.name.or.IP>
  ServerName host.name.or.IP

  JkMount /jmx-console loadbalancer
  JkMount /jmx-console/* loadbalancer
</VirtualHost>
```

-

---

**Resources**

JBoss Clustering Guide: Additional details about HTTP Session replication, workers.properties and other goodies.

The Apache Tomcat Connector - Documentation Index

The Apache Tomcat Connector - workers.properties configuration

The Apache Tomcat Connector - Configuring Apache

The Apache Tomcat Connector - Configuring IIS

Configuring Tomcat and Apache With JK 1.2

Tomcat - Workers HowTo

Tomcat user-mail archive

mod_jk FAQ

An Optimal base Apache, Tomcat, mod_jk 1.2 configuration

Using mod_jk 1.2 with a Firewall

Using mod_jk 1.2 with JBoss & Microsoft IIS

Using mod_jk 1.2 with JBoss & Microsoft IIS7

Using mod_jk 1.2 with JBoss & Netscape / Iplanet / SunOne

Fronting Tomcat with Apache, White paper by Mladen Turk

**Referenced by:**