JBoss AS 5.1 Clustering Guide

(Before changing any file take backup.)

Getting and Installing Java:
Download Java SE 6 JDK from http://java.sun.com.
Run the installer or uncompress the distribution into a directory of your choice (e.g. /usr/jdk1.6).

Configuring Java:
Set JAVA_HOME to point to the directory where you installed Java and add $JAVA_HOME/bin to your PATH as below.
Make these changes in your shell's configuration file (e.g. ~/.profile):
export JAVA_HOME=/usr/jdk1.6
export PATH=$JAVA_HOME/bin:$PATH

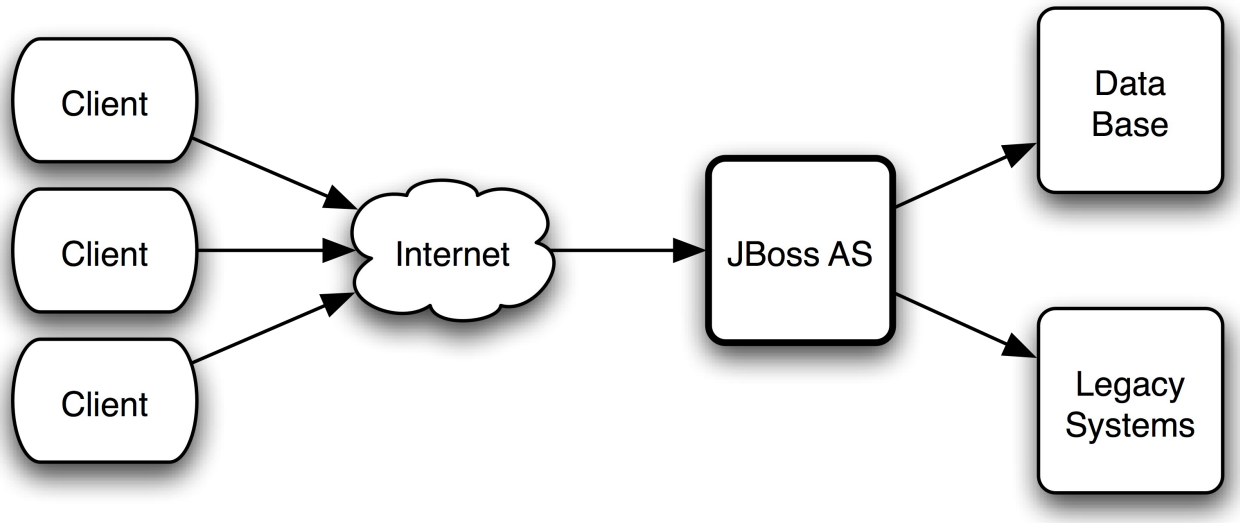
Getting and Installing JBoss AS:
Download packaged distribution from http://www.jboss.org/jbossas/downloads
Unpack the compressed archive into a directory of your choice. e.g. /app/jboss

Clustering:
A cluster is a set of nodes that communicate with each other and work toward a common goal.

A Cluster provide these functionalities:
Scalability (can we handle more users? can we add hardware to our system?)
Load Balancing (share the load between servers)
High Availability (our application has to be uptime close to 100%)
Fault Tolerance (High Availability and Reliability)

Clustering and Jboss:
Support clustering with a built in configuration **all** configuration.
Can also be integrated to an external balancer.

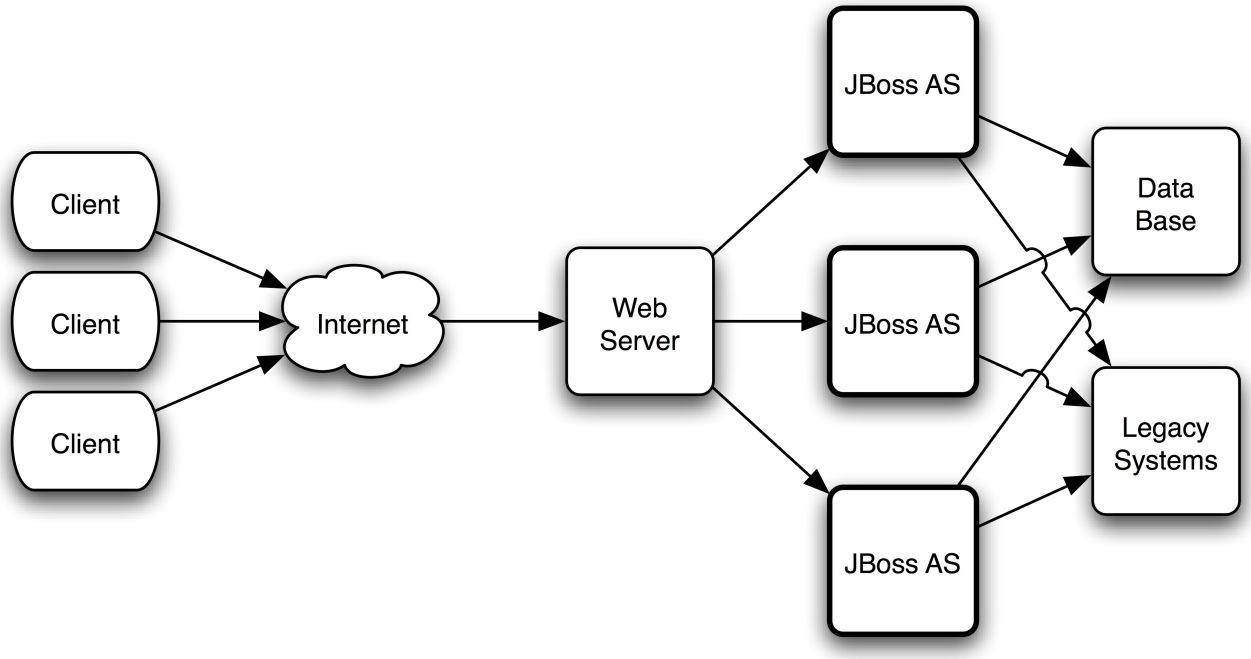
Simple Web Architecture:



Not scalable. Additional users can only be handled by improving the performance of the server (e.g. adding additional CPUs, more memory).
No fault tolerance. If the JBoss AS server goes down, the entire service becomes unavailable.

External Load Balancer Architecture:
Add one or many web servers to balance the load to multiple JBoss AS nodes typically running on separate physical servers.
Additional user load can be handled by adding another server running JBoss AS.

If any one of the JBoss AS nodes fail, the service is still available through other JBoss AS servers.



A cluster is defined by:
Multicast Address
Multicast Post
Cluster Name

Multicast is the protocol which allow nodes inside to a cluster to communicate without knowing each other. Communication between nodes is provided by JGroups, which is library for multicast communication.

General configuration for the following examples:

Preparing a set of servers to act as a JBoss AS cluster involves a few simple steps:
Copy the **all** directory and create two directory **node1** and **node2** as below,

```
$ cd /app/jboss/server
$ cp -r all node1
$ cp -r all node2
```

Requirements Of Jboss Cluster:
Multicast Address
Cluster Name
ServerPeerID (From which JBoss Messaging gets its unique id.)

In this scenario we have 2 nodes with differwnt ports on same server. Assume the machine has the 192.168.0.101 addresse assigned. The two JBoss instances(node1 & node2) in created as below /app/jboss/server/node1 & /app/jboss/server/node2. The ServerPeerID [4] for the first node is 1 and for the second node is 2. We have decided to set cluster name as "TestPartition" and to use 239.255.100.100 as our multicast address.

Launching a JBoss AS Cluster:

Now just start cluster nodes as below,

For Node1

```
$JBOSS_HOME/bin/run.sh -c node1 -b 0.0.0.0 -g TestPartition -u 239.255.100.100
-Djboss.messaging.ServerPeerID=1 -Djboss.service.binding.set=ports-01
```

For Node2

```
$JBOSS_HOME/bin/run.sh -c node2 -b 0.0.0.0 -g TestPartition -u 239.255.100.100
-Djboss.messaging.ServerPeerID=2 -Djboss.service.binding.set=ports-02
```

In above scripts
The -c switch says to use the config “-c node1”.
The -g switch sets the cluster name “-u TestPartition”.
The -u switch sets the multicast address that will be used for intra-cluster communication “-u 239.255.100.100”.
The -b switch sets the address on which sockets will be bound “-b 0.0.0.0”.
The -Djboss.messaging.ServerPeerID from which JBoss Messaging gets its unique id “-Djboss.messaging.ServerPeerID=1”.
The -Djboss.service.binding.set switch sets the port set for instance “-Djboss.service.binding.set=ports-01”.
Ports sets are as below.

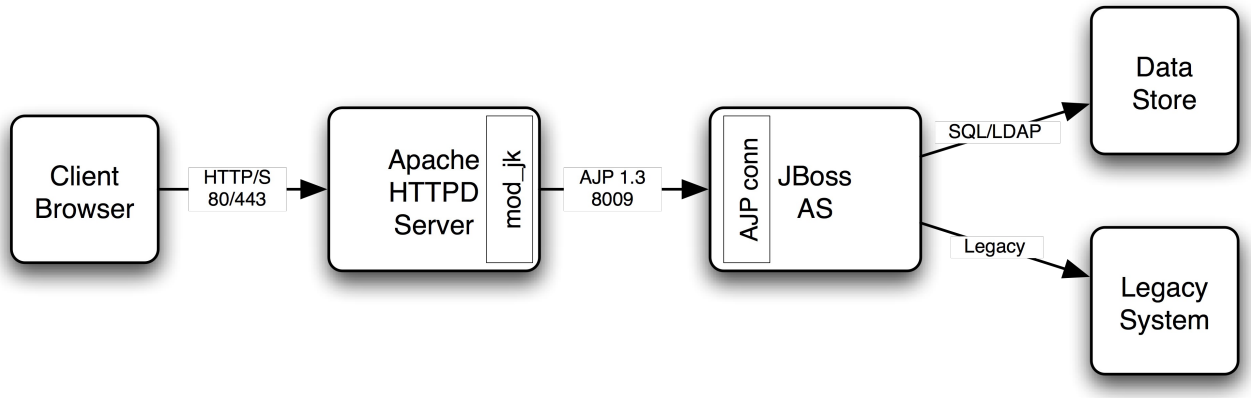
Ports-default	=	8080
Ports-01	=	8180
ports-02	=	8280
ports-03	=	8380

Load Balancing Using Apache & mod_jk

Apache is a well-known web server which can be extended by plugging in modules. One of these modules, mod_jk, has been specifically designed to allow the forwarding of requests from Apache to a Servlet container. Furthermore, it is also able to load-balance HTTP calls to a set of Servlet containers while maintaining sticky sessions.

Advantages of Fronting with a Web Server :
Performance: dynamic vs. static content
Scalability & High Availability: load balancing and fail over
Security: web servers are simpler and easier to protect
Stability: proven, more robust
Features: URL rewriting, fine-grained access control, etc.

Fronting with Apache HTTPD :



Steps for Fronting with Apache HTTPD :

Install and setup Apache HTTPD
Install and configure mod_jk on Apache
AJP Connector on JBoss AS already enabled
Access web apps through Apache

Mod_jk (version 1.2.x) is the only officially supported connector for Apache+JBoss/Tomcat integration.

Installing mod_jk
First of all, make sure that you have Apache installed. You can download Apache directly from

Apache web site at http://httpd.apache.org. Installation of mod_jk is pretty straightforward and requires no specific configuration. Installation steps are as below,

```
$tar -zxvf tomcat-connectors-1.2.30-src.tar.gz
$cd tomcat-connectors-1.2.30-src/native
$./configure --with-apxs=$APACHE_HOME/bin/apxs
$make
$sudo make install
```

Configure Apache to load mod_jk :

Include configuration file of mod_jk in <apache-dir>/conf/httpd.conf as below.

```
Include conf/mod_jk.conf
```

Configuring mod_jk :

Create <apache-dir>/conf/mod_jk.conf & configure as below.

```
# Load mod_jk module
LoadModule    jk_module       /APACHE_HOME/modules/mod_jk.so

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

# Where to find mod_jk.log file
JkLogFile /log/mod_jk.log

#Log level
JkLogLevel info

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

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Add shared memory. This is needed for for load balancing to work properly
JkShmFile /log/jk.shm

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

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

# Mount your applications
JkMountCopy All
JkMount         /jmx-console    jboss
JkMount         /jmx-console/*  jboss
JkMount         /admin-console          jboss
JkMount         /admin-console/*        jboss
JkMount         /jkstatus       jkstatus
```

Define a JBoss AS instance in APACHE_HOME/conf/workers.properties

```
worker.list=jboss,jkstatus

#For Node1
worker.jboss1.type=ajp13
worker.jboss1.host=127.0.0.1
worker.jboss1.port=8009
worker.jboss1.lbfactor=1

#For Node2
worker.jboss2.type=ajp13
worker.jboss2.host=127.0.0.1
worker.jboss2.port=8109
worker.jboss2.lbfactor=1

worker.jboss.type=lb
worker.jkstatus.type=status
worker.jboss.sticky_session=1
worker.jboss.balance_workers=jboss1,jboss2
```

Configuring JBoss to work with mod_jk

Finally, we must configure the JBoss AS instances on all clustered nodes so that they can expect requests forwarded from the mod_jk loadbalancer. On each clustered JBoss node, we have to name the node according to the name specified in workers.properties. For instance, on JBoss instance node1, edit the JBOSS_HOME/server/node1/deploy/jbossweb.sar/server.xml file.
Locate the <Engine> element and add an attribute jvmRoute as below:

```
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="jboss1">
... ...
</Engine>
```

Also for JBoss instance node2, edit the JBOSS_HOME/server/node2/deploy/jbossweb.sar/server.xml file.
Locate the <Engine> element and add an attribute jvmRoute as below:

```
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="jboss2">
... …
</Engine>
```

You also need to be sure the AJP connector in server.xml is enabled (i.e., uncommented). It is enabled by default. Here your jboss instance is listening on ajp port (8009), you can disable http port (8080).

```
<!-- An AJP 1.3 Connector on port 8009 →
<Connector protocol="AJP/1.3" port="8009" address="${jboss.bind.address}"
redirectPort="8443" />
```

Now start node1, node2 & apache server.
Access the the url for jmx-console as http://127.0.0.1/jmx-console