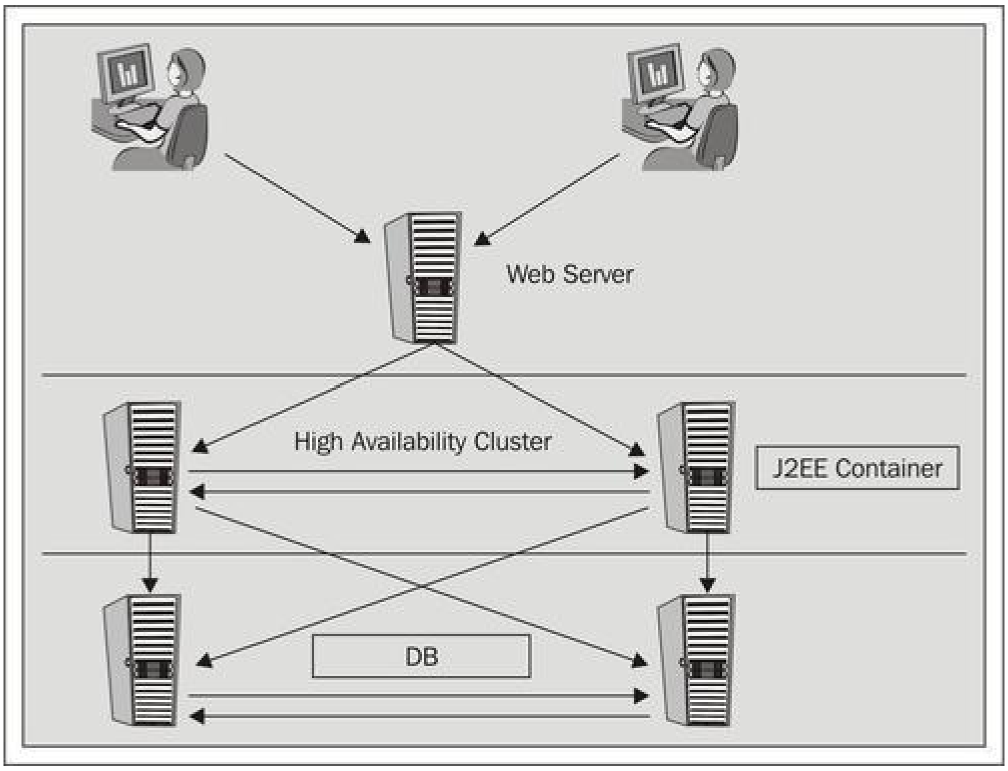
Lab 9. Clustering in Tomcat 8

In this lab, we discuss:

- High availability architecture and its advantages
- Different types of high availability architectures including load balancing and clustering
- Approaches used by IT industries while building a high availability architecture in an enterprise setup
- How to do Apache Tomcat clustering
- Various clustering architectures
- How to solve common problems in clustering

What is a cluster?

The cluster is a group of servers or computers connected together, which may perform similar functionality in the system. These systems are normally connected to each other through high speed Ethernet. Cluster systems are used where quick processing or system availability is required. Some of the examples where clusters are highly used are the financial sector, banking, security areas, and so on. The following figure shows the J2EE containers clustered in the environment:



Note

Cluster topology varies from environment to environment, based on the application requirements.

Benefits of clustering

There are many advantages of clustering in a middleware environment. It also depends on which cluster techniques we are using. We will discuss the various advantages of clustering:

- **Scalability:** It gives freedom to the system architect to accommodate future enhancements for applications and servers. Suppose the web application currently has 100 concurrent users, and at the time of an event, you are expecting 500 concurrent users. What would you do to make sure that the system runs as expected? Clustering is one of the best solutions.
- **High availability:** High availability systems are implemented in environments that need to be up 99.99 percent, such as banking, financial sectors, and so on where entire transactions need to be recorded on the systems. They cannot afford to have their websites down for a minute. Hence, they implement a high availability system to make sure the system will not be down at any time.
- **High performance:** One of the major advantages of clustering is that it boosts up the system performance by[*n*] times where,[*n*] = number of systems. For example, if you are running the system with a single server and the system supports 100 concurrent users, then just by adding another server system you can support 200 concurrent users. Also, if you want to decrease the response time for the application, you can use the JVM performance tuning.
- **Cloud computing:** Clustering is also very useful in a cloud computing environment. It is used while setting the grid computing architecture for cloud computing to improve performance.

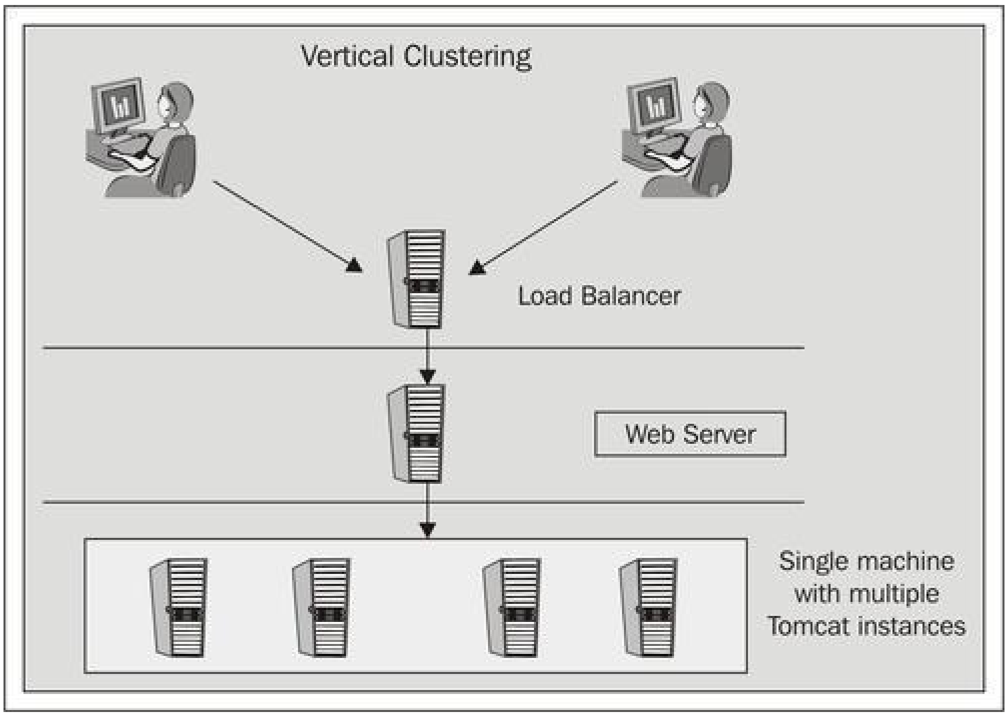
Disadvantages of clustering

Until now, we have discussed how useful clustering is for a web environment. Let's discuss the disadvantages of clustering:

- **Cost:** It plays a major role in the implementation of a new environment. If we want to setup web clustering, then we need more servers. This, again, increases the cost of the project.
- **Monitoring:** With an increase in the number of servers, the monitoring of servers will also increase, making it difficult for the web administrator to manage the servers.

Vertical clustering

Vertical clustering consists of a single hardware with multiple instances running, using shared resources from the system. This kind of setup is mainly done in development and quality systems for the developer to test the functionality of the application. Also, vertical clustering can be implemented in production in certain cases, where there is a resource crunch for the hardware. It uses the concept of a shared resource such as CPU, RAM, and so on. The following figure shows the pictorial presentation of vertical clustering:



Every architecture has its pros and cons. Let's discuss some of the pros and cons of vertical clustering.

Advantages of vertical clustering

Following are the advantages of vertical clustering:

- No network bandwidth issue, as instances are hosted on a single machine
- Hardware is shared by different Tomcat instances
- Addition of physical hardware is not required
- Single JVM can be shared by multiple instances

Clustering in Apache Tomcat 8

For vertical clustering, we have to configure at least two instances of Apache Tomcat and the complete process consists of three stages. Let's discuss and implement the steps for vertical cluster in Tomcat 7:

1. Installation of the Tomcat instance.

2. Configuration of the cluster.

3. Apache HTTP web server configuration for the vertical cluster.

Installation of the Tomcat instance

The installation of Apache Tomcat 8 can be done in three easy steps:

Note: Setup apache-tomcat-8.5.61.zip has been downloaded and unzipped in /opt/apache-tomcat-8.5.61 directory already.

1. Download and unzip the software in the required directory.

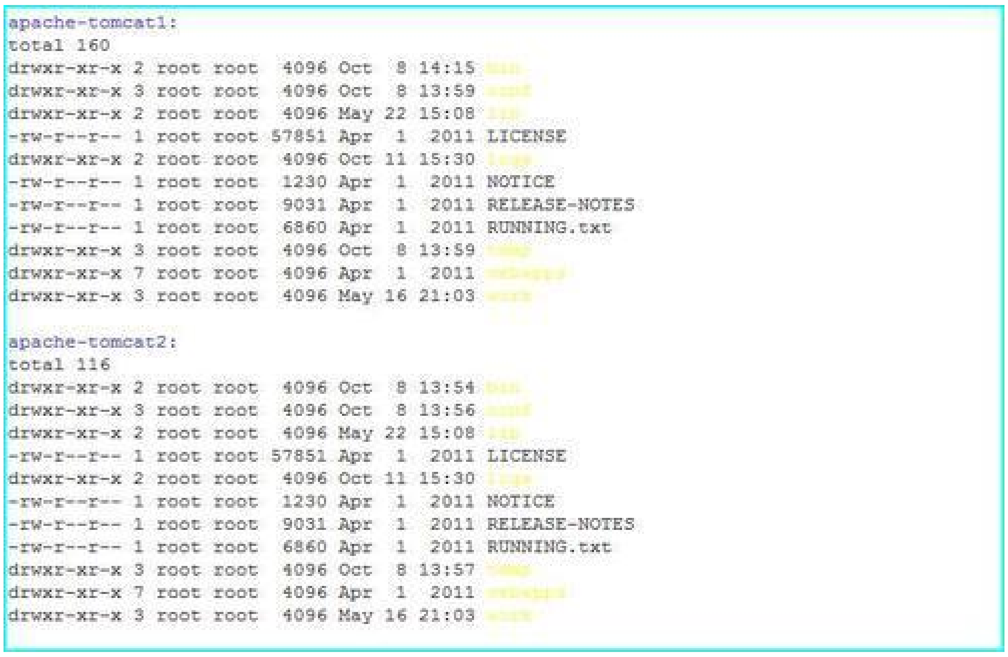
2. Install the JDK and set the JAVA_HOME.

3. Copy the Apache Tomcat source code in two different directories, for example, /opt/tomcatX (where X= instance number) and verify that the files are properly copied on both instances, using the following command:

```
```
[root@localhost opt]# cd /opt
[root@localhost opt]# cp -r /opt/apache-tomcat-8.5.61 /opt/apache-tomcat1
[root@localhost opt]# cp -r /opt/apache-tomcat-8.5.61 /opt/apache-tomcat2

[root@localhost opt]# ls -l apache-tomcat*
```
```

- The following screenshot shows the output of the previous command:



```
apache-tomcat1:
total 160
drwxr-xr-x 2 root root  4096 Oct  8 14:15 bin
drwxr-xr-x 3 root root  4096 Oct  8 13:59 conf
drwxr-xr-x 2 root root  4096 May 22 15:08 lib
-rw-r--r-- 1 root root 57851 Apr  1  2011 LICENSE
drwxr-xr-x 2 root root  4096 Oct 11 15:30 logs
-rw-r--r-- 1 root root  1230 Apr  1  2011 NOTICE
-rw-r--r-- 1 root root  9031 Apr  1  2011 RELEASE-NOTES
-rw-r--r-- 1 root root  6860 Apr  1  2011 RUNNING.txt
drwxr-xr-x 3 root root  4096 Oct  8 13:59 temp
drwxr-xr-x 7 root root  4096 Apr  1  2011 webapps
drwxr-xr-x 3 root root  4096 May 16 21:03 work

apache-tomcat2:
total 116
drwxr-xr-x 2 root root  4096 Oct  8 13:54 bin
drwxr-xr-x 3 root root  4096 Oct  8 13:56 conf
drwxr-xr-x 2 root root  4096 May 22 15:08 lib
-rw-r--r-- 1 root root 57851 Apr  1  2011 LICENSE
drwxr-xr-x 2 root root  4096 Oct 11 15:30 logs
-rw-r--r-- 1 root root  1230 Apr  1  2011 NOTICE
-rw-r--r-- 1 root root  9031 Apr  1  2011 RELEASE-NOTES
-rw-r--r-- 1 root root  6860 Apr  1  2011 RUNNING.txt
drwxr-xr-x 3 root root  4096 Oct  8 13:57 temp
drwxr-xr-x 7 root root  4096 Apr  1  2011 webapps
drwxr-xr-x 3 root root  4096 May 16 21:03 work
```

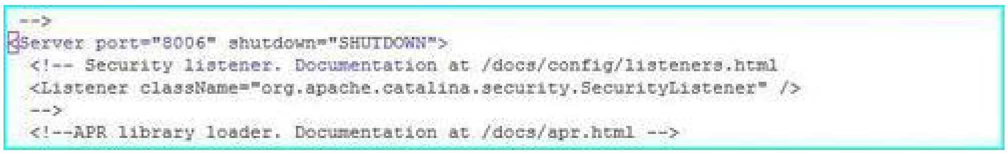
Configuration of a vertical cluster

This section is the most critical section for vertical clustering, as all the configurations are done in this section and a simple error can make the cluster non-functioning. So, be careful while carrying out the configuration. Let's do the step-by-step configuration on each node.

Configuration of instance 1

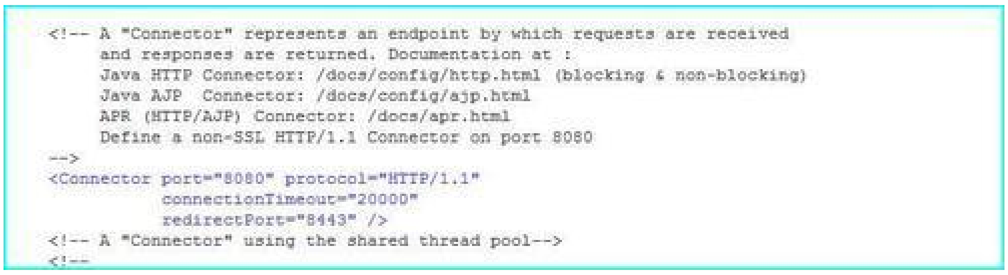
For the first instance; node 1, we can use the default configuration, such as Connector, AJP, or shutdown port in `server.xml` . Let's discuss each component where the configuration needs to be done and why it is used:

1. Shutdown port: The following screenshot shows the configuration for the shutdown port for the Tomcat instance. While running multiple instances, if by any chance, you have skipped configuring the shutdown port, then the Tomcat instance will be unable to start it.



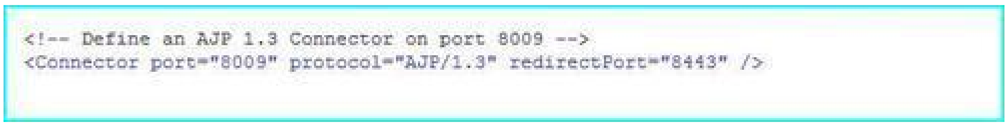
```
-->
<Server port="8006" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
```

2. Connector port: The following screenshot shows the Connector port configuration for Tomcat 8. This port is used to access the Tomcat instance, for example, normally we access the Tomcat instance by using http://localhost:8080, the 8080 port is called the Connector port. While running multiple instances, if you have skipped configuring this port, then the Tomcat instance will be unable to start it and you will get the Port already in use exception.



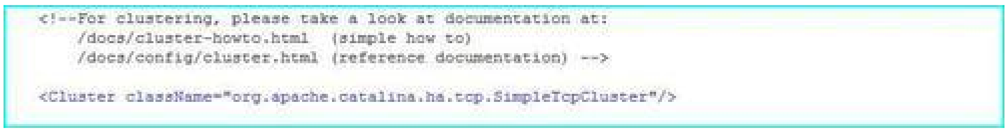
```
    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
```

3. AJP port: The following screenshot shows the AJP port configuration for Tomcat 8. This port is used for AJP communication between the Apache HTTP server and the Tomcat instance. While running multiple instances, if you have skipped to configuring port, then the Tomcat instance will be unable to start it and you will get the Port already in use exception.



```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

4. Cluster attributes: Enable the cluster attributes for clustering in server.xml and the following screenshot shows the cluster class used for clustering:

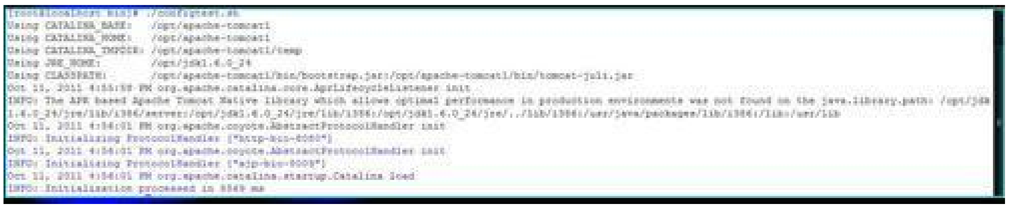


```
<!--For clustering, please take a look at documentation at:
    /docs/cluster-howto.html  (simple how to)
    /docs/config/cluster.html (reference documentation) -->

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
```

5. Configuration test: Run the configtest.sh script from TOMCAT_HOME/bin to check the configuration. The following screenshot shows the output for the following config.sh command:

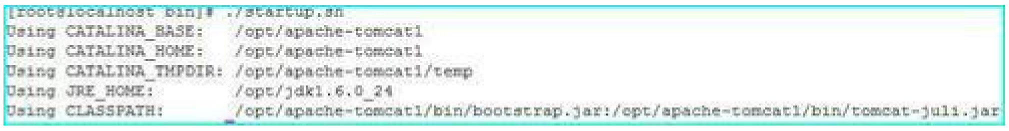
```
[root@localhost bin]# ./configtest.sh
```



```
[root@localhost bin]# ./configtest.sh
Using CATALINA_BASE:   /opt/apache-tomcat1
Using CATALINA_HOME:   /opt/apache-tomcat1
Using CATALINA_TMPDIR: /opt/apache-tomcat1/temp
Using JRE_HOME:        /opt/jdk1.6.0_24
Using CLASSPATH:       /opt/apache-tomcat1/bin/bootstrap.jar:/opt/apache-tomcat1/bin/tomcat-juli.jar
Oct 11, 2011 4:55:59 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found on the java.library.path: /opt/jdk
1.6.0_24/jre/lib/i386/server:/opt/jdk1.6.0_24/jre/lib/i386:/opt/jdk1.6.0_24/jre/../lib/i386:/usr/java/packages/lib/i386:/lib:/usr/lib
Oct 11, 2011 4:56:01 PM org.apache.coyote.AbstractProtocolHandler init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Oct 11, 2011 4:56:01 PM org.apache.coyote.AbstractProtocolHandler init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Oct 11, 2011 4:56:01 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 5569 ms
```

6. Tomcat instance startup: Start the instance 1 configuration using the script startup.sh. The following screenshot shows the output for the following startup.sh script:

   ```
   [root@localhost bin]# ./startup.sh
   ```

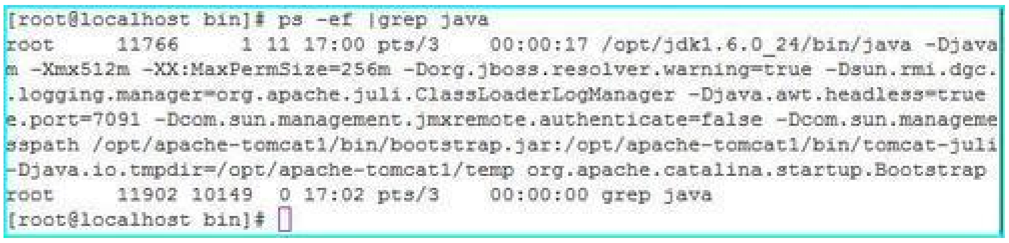


```
[root@localhost bin]# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat1
Using CATALINA_HOME:   /opt/apache-tomcat1
Using CATALINA_TMPDIR: /opt/apache-tomcat1/temp
Using JRE_HOME:        /opt/jdk1.6.0_24
Using CLASSPATH:       /opt/apache-tomcat1/bin/bootstrap.jar:/opt/apache-tomcat1/bin/tomcat-juli.jar
```

- Check the Tomcat instance process using the following mentioned

  ```
   command. The following screenshot shows the output for the
   `ps` command:
  ```

```
```
[root@localhost bin]# ps -ef |grep java
```
```

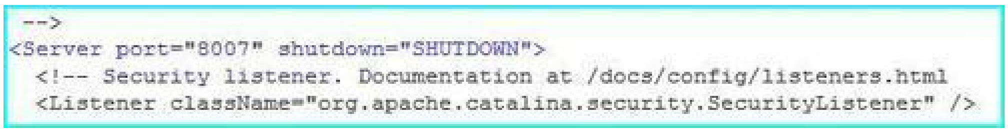


```
[root@localhost bin]# ps -ef |grep java
root     11766     1 11 17:00 pts/3    00:00:17 /opt/jdk1.6.0_24/bin/java -Djava
m -Xmx512m -XX:MaxPermSize=256m -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.
.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.awt.headless=true
e.port=7091 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.manageme
sspath /opt/apache-tomcat1/bin/bootstrap.jar:/opt/apache-tomcat1/bin/tomcat-juli
-Djava.io.tmpdir=/opt/apache-tomcat1/temp org.apache.catalina.startup.Bootstrap
root     11902 10149  0 17:02 pts/3    00:00:00 grep java
[root@localhost bin]#
```

Configuration of instance 2

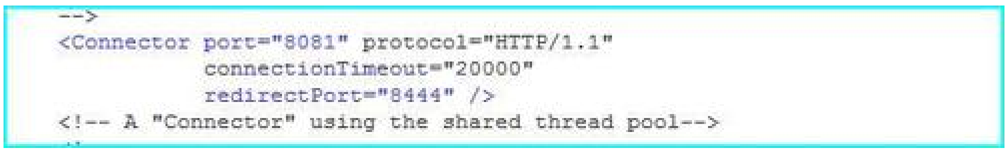
We cannot use the default configuration on node 2. There will be port conflicts as we are running the instance with single a IP on the same physical machine. Let's configure instance 2 step-by-step:

1. Change the shutdown port for instance 2 in server.xml (increment it by 1). The following screenshot shows the configuration:



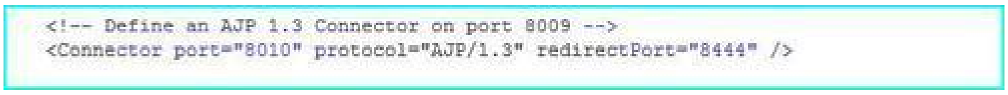
```
 -->
<Server port="8007" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
```

2. Change the Connector and redirect the port for instance 2 in server.xml (increment it by 1). The following screenshot shows the configuration:



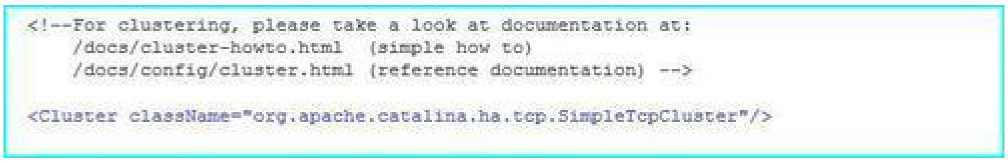
```
    -->
    <Connector port="8081" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8444" />
    <!-- A "Connector" using the shared thread pool-->
```

3. Change the AJP and redirect the port for instance 2 in server.xml (increment it by 1). The following screenshot shows the configuration:



```
    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8010" protocol="AJP/1.3" redirectPort="8444" />
```

4. Enable the cluster attributes for clustering in server.xml. The following screenshot shows the configuration:



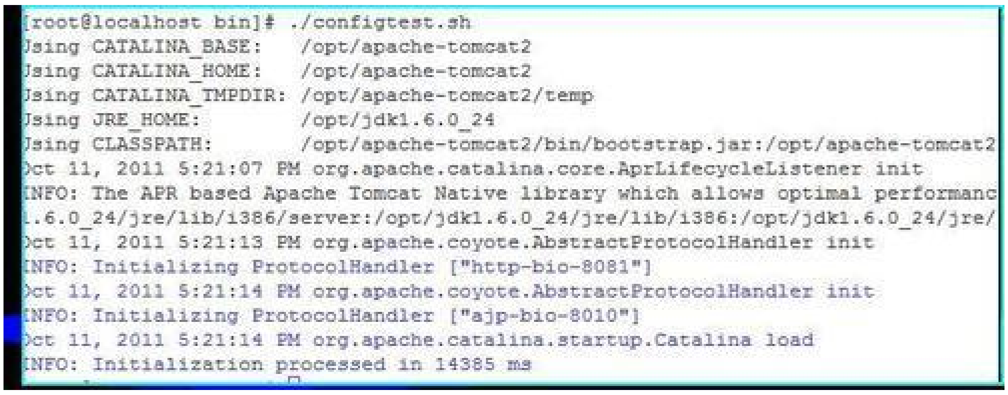
```
  <!--For clustering, please take a look at documentation at:
      /docs/cluster-howto.html  (simple how to)
      /docs/config/cluster.html (reference documentation) -->

  <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
```

5. Save server.xml.

6. Run the configtest.sh script from TOMCAT_HOME/bin to check the configuration. The following screenshot shows the output for the following startup.sh script:

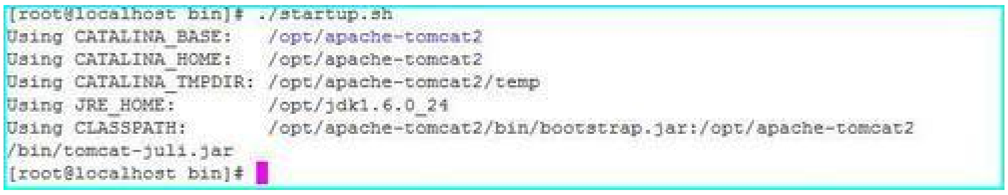
```
[root@localhost bin]# ./configtest.sh
```



```
[root@localhost bin]# ./configtest.sh
Jsing CATALINA_BASE:   /opt/apache-tomcat2
Jsing CATALINA_HOME:   /opt/apache-tomcat2
Jsing CATALINA_TMPDIR: /opt/apache-tomcat2/temp
Jsing JRE_HOME:        /opt/jdk1.6.0_24
Jsing CLASSPATH:       /opt/apache-tomcat2/bin/bootstrap.jar:/opt/apache-tomcat2
Oct 11, 2011 5:21:07 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performanc
l.6.0_24/jre/lib/i386/server:/opt/jdk1.6.0_24/jre/lib/i386:/opt/jdk1.6.0_24/jre/
Oct 11, 2011 5:21:13 PM org.apache.coyote.AbstractProtocolHandler init
INFO: Initializing ProtocolHandler ["http-bio-8081"]
Oct 11, 2011 5:21:14 PM org.apache.coyote.AbstractProtocolHandler init
INFO: Initializing ProtocolHandler ["ajp-bio-8010"]
Oct 11, 2011 5:21:14 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 14385 ms
```

7. Start the configuration of instance 2 using the script startup.sh. The following screenshot shows the output for the following startup.sh script:

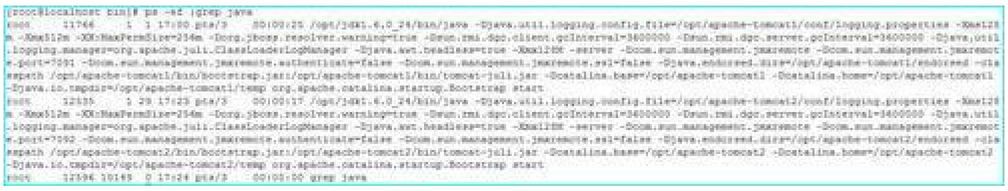
```
[root@localhost bin]# ./startup.sh
```



```
[root@localhost bin]# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat2
Using CATALINA_HOME:   /opt/apache-tomcat2
Using CATALINA_TMPDIR: /opt/apache-tomcat2/temp
Using JRE_HOME:        /opt/jdk1.6.0_24
Using CLASSPATH:       /opt/apache-tomcat2/bin/bootstrap.jar:/opt/apache-tomcat2
/bin/tomcat-juli.jar
[root@localhost bin]#
```

8. Check the Tomcat instance process. The following screenshot shows the output for the ps command:

```
[root@localhost bin]# ps -ef |grep java
```



```
[root@localhost bin]# ps -ef |grep java
root     11746     1  1 17:00 pts/3    00:00:25 /opt/jdk1.6.0_24/bin/java -Djava.util.logging.config.file=/opt/apache-tomcat1/conf/logging.properties -Xms128
m -Xmx512m -XX:MaxPermSize=256m -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.util
.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.awt.headless=true -Xms128M -server -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremot
e.port=7091 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.endorsed.dirs=/opt/apache-tomcat1/endorsed -cla
sspath /opt/apache-tomcat1/bin/bootstrap.jar:/opt/apache-tomcat1/bin/tomcat-juli.jar -Dcatalina.base=/opt/apache-tomcat1 -Dcatalina.home=/opt/apache-tomcat1
-Djava.io.tmpdir=/opt/apache-tomcat1/temp org.apache.catalina.startup.Bootstrap start
root     12535     1 29 17:23 pts/3    00:00:17 /opt/jdk1.6.0_24/bin/java -Djava.util.logging.config.file=/opt/apache-tomcat2/conf/logging.properties -Xms128
m -Xmx512m -XX:MaxPermSize=256m -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.util
.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.awt.headless=true -Xms128M -server -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremot
e.port=7092 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.endorsed.dirs=/opt/apache-tomcat2/endorsed -cla
sspath /opt/apache-tomcat2/bin/bootstrap.jar:/opt/apache-tomcat2/bin/tomcat-juli.jar -Dcatalina.base=/opt/apache-tomcat2 -Dcatalina.home=/opt/apache-tomcat2
-Djava.io.tmpdir=/opt/apache-tomcat2/temp org.apache.catalina.startup.Bootstrap start
root     12596 10149  0 17:24 pts/3    00:00:00 grep java
```

9. Now, check catalina.out for both the nodes.

   The logs for node 1 are similar to the following:

```
Oct 11, 2011 5:00:24 PM org.apache.catalina.ha.tcp.SimpleTcpCluster
startInternal
INFO: Cluster is about to start
Oct 11, 2011 5:00:24 PM org.apache.catalina.tribes.transport.ReceiverBase bind
INFO: Receiver Server Socket bound to:/127.0.0.1:4000
Oct 11, 2011 5:00:24 PM org.apache.catalina.tribes.membership.McastServiceImpl
setupSocket
```

```
# Instance node 1 started on port 4000
INFO: Setting cluster mcast soTimeout to 500
Oct 11, 2011 5:00:24 PM
INFO: Sleeping for 1000 milliseconds to establish cluster membership, start
level:8
Oct 11, 2011 5:00:26 PM org.apache.catalina.tribes.membership.McastServiceImpl
waitForMembers
# waiting for other member to join the cluster
org.apache.catalina.ha.session.JvmRouteBinderValve startInternal
INFO: JvmRouteBinderValve started
Oct 11, 2011 5:00:37 PM org.apache.coyote.AbstractProtocolHandler start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Oct 11, 2011 5:00:37 PM org.apache.coyote.AbstractProtocolHandler start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Oct 11, 2011 5:00:37 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 13807 ms
Oct 11, 2011 5:23:42 PM org.apache.catalina.tribes.io.BufferPool getBufferPool
INFO: Created a buffer pool with max size:104857600 bytes of
type:org.apache.catalina.tribes.io.BufferPool15Impl
Oct 11, 2011 5:23:43 PM org.apache.catalina.ha.tcp.SimpleTcpCluster memberAdded
INFO: Replication member added:org.apache.catalina.tribes.membership.MemberImpl
[tcp://{127, 0, 0, 1}:4001,{127, 0, 0, 1},4001, alive=1043, securePort=-1, UDP
Port=-1, id={33 91 -59 78 -34 -52 73 -9 -99 124 -53 34 69 21 -40 -82 },
payload={}, command={}, domain={}, ]
#Instance 2 joined the cluster node.
```

~~~~
The logs for node 2 are similar to the following:


```
INFO: Starting Servlet Engine: Apache Tomcat/8.5.61
Oct 11, 2011 5:23:41 PM org.apache.catalina.ha.tcp.SimpleTcpCluster startInternal
INFO: Cluster is about to start
Oct 11, 2011 5:23:42 PM org.apache.catalina.tribes.transport.ReceiverBase bind
INFO: Receiver Server Socket bound to:/127.0.0.1:4001
Oct 11, 2011 5:23:42 PM org.apache.catalina.tribes.membership.McastServiceImpl
setupSocket
# Instance node 2 started on port 4001
INFO: Setting cluster mcast soTimeout to 500
Oct 11, 2011 5:23:42 PM org.apache.catalina.tribes.membership.McastServiceImpl
waitForMembers
INFO: Sleeping for 1000 milliseconds to establish cluster membership, start level:4
Oct 11, 2011 5:23:43 PM org.apache.catalina.ha.tcp.SimpleTcpCluster memberAdded
INFO: Replication member added:org.apache.catalina.tribes.membership.MemberImpl
[tcp://{127, 0, 0, 1}:4000,{127, 0, 0, 1},4000, alive=1398024, securePort=-1, UDP
Port=-1, id={28 42 60 -68 -99 126 64 -35 -118 -97 7 84 26 20 90 24 }, payload={},
command={}, domain={}, ]
# Instance 1 joined the cluster node 2.
```
~~~~

Apache web server configuration for vertical clustering

Until now, we have done the Tomcat-level configuration to configure vertical clustering in the Tomcat instance. It's time to integrate the Apache web server to Tomcat 8. Let's enable the integration by performing the following steps:

Replace all the contents within the VirtualHost block with the following, so your configuration file looks like this:

/etc/apache2/sites-available/000-default.conf

```
<VirtualHost *:80>
<Proxy balancer://tomcatcluster>
    BalancerMember http://127.0.0.1:8080
    BalancerMember http://127.0.0.1:8081
</Proxy>

    ProxyPreserveHost On

    ProxyPass / balancer://tomcatcluster/
    ProxyPassReverse / balancer://tomcatcluster/

</VirtualHost>
```

The configuration is similar to the previous one, but instead of specifying a single backend server directly, we've used an additional Proxy block to define multiple servers. The block is named balancer://tomcatcluster (the name can be freely altered) and consists of one or more BalancerMembers, which specify the underlying backend server addresses. The ProxyPass and ProxyPassReverse directives use the load balancer pool named **tomcatcluster** instead of a specific server.

If you followed along with the example servers in Step 2, use 127.0.0.1:8080 and 127.0.0.1:8081 for the BalancerMember directives, as written in the block above. If you have your own application servers, use their addresses instead.

To put these changes into effect, restart Apache.

`service apache2 restart`

Monitoring of Tomcat clustering

Once the cluster is up and working, the next stage is to set up the monitoring of the clustering. This can be done in the following ways:

- Various monitoring tools
- Scripts
- Manual

Following are the steps to manually monitor the clusters:

1. Check the Tomcat process using the following command:

   ```
   root@localhost bin]# ps -ef |grep java
   ```

2. Check the logs to verify the connectivity of the cluster.

3. Verify the URL for both cluster members.

Summary

In this lab, we have discussed the clustering of Tomcat 8 and its implementation techniques. We have discussed clustering architecture, horizontal and vertical clusters and their benefits, the implementation of horizontal and vertical clustering on Tomcat 8, and the verification of clusters.

In the next lab, we will discuss the most awaited topic of Tomcat 8, that is, Tomcat 6 upgrade to Tomcat 8 and different techniques used during the upgrade.