Tomcat Administration
Hands on Lab

Note:

1. **All the installable(s) need to be downloaded into /home/training/Downloads folder from the given ftp site.**
2. **Replace "tomcat" with the actual release name of tomcat distribution name.**

Lab 1: Install and run Tomcat

Step1: Install JDK

```
$ sudo mkdir /usr/java
$mv /home/training/Download/jdk-8u31-linux-i586.tar.gz
/usr/java
$tar -zxvf  /usr/java/ jdk-8u31-linux-i586.tar.gz
$rm /usr/java/ jdk-8u31-linux-i586.tar.gz
```

Step2: set JAVA_HOME and PATH

```
$sudo /etc/profile.d/myenv.sh

Add the following lines to the file and save
JAVA_HOME=/usr/java/jdk1.8.0_31
PATH=$JAVA_HOME/bin:$PATH
export JAVA_HOME PATH
```

Step 3: Reboot your System

$sudo reboot

Step4: Install Tomcat Server

```
$sudo mv /home/training/Downloads/tomcat.zip /opt
$sudo unzip /opt/tomcat.zip –d /opt
$sudo chmod 750 –R /opt/tomcat
$sudo chown  -R training:training /opt/tomcat
```

Step 5: Start the Server

```
$cd /opt/tomcat/bin
$./startup.sh
Open one more terminal and check the log as
$tail  -f –n   /opt/tomcat/logs/catalina.out
(output from the above command should show that the server has
started)
```

Step 6: Open a browser window and type

http://localhost:8080

This url should bring up the tomcat home page.

Step 7: Shutdown the server

```
$cd /opt/tomcat/bin
$./shutdown.sh

Observe the log for shutdown confirmation.
```

Lab 2: Configure 2 CATALINA_BASE and start 2 instances of Tomcat Server

Step 1: CATALINA_BASE (1st)

```
mkdir /opt/tomcat/tomcat1
cd /opt/tomcat
cp -R webapps temp work logs conf tomcat1/
```

Step2: CATALINA_BASE (2nd)

```
mkdir /opt/tomcat/tomcat2
cd /opt/tomcat
cp -R webapps temp work logs conf tomcat2/
```

Step3: Make 2 copies of startup.sh and update CATALINA_BASE

```
cp bin/startup.sh bin/startup1.sh
cp bin/startup.sh bin/startup2.sh
```

Step4: edit the above script files to include/modify the CATALINA_BASE

```
vi bin/startup1.sh

add the following to the beginning of the file

export CATALINA_BASE=/opt/tomcat/tomcat1

vi bin/startup2.sh

add the following to the beginning of the file

export CATALINA_BASE=/opt/tomcat/tomcat2
```

Step5: Make 2 copies of shutdown.sh and update CATALINA_BASE

```
cp bin/shutdown.sh bin/shutdown1.sh
cp bin/shutdown.sh bin/shutdown2.sh
```

Step6: edit the above script files to include/modify the CATALINA_BASE

```
vi bin/shutdown1.sh
add the following to the beginning of the file

export CATALINA_BASE=/opt/tomcat/tomcat1

vi bin/shutdown2.sh
add the following to the beginning of the file

export CATALINA_BASE=/opt/tomcat/tomcat2
```

Step7: modify the server ports

Edit /opt/tomcat/**tomcatX**/conf/server.xml and
Modify the ports as follows:

CATALINA_BASE	Port Name	Value
tomcat1	shutdown	8015
	ajp	8019
	http	8090
tomcat2	shutdown	8025
	Ajp	8029
	http	8100

Step8: Start,Test and shutdown the servers

```
$cd /opt/tomcat/bin
$./startup1.sh
$./startup2.sh
Test the servers as we did in LAB1->step6
```

Lab3: Install a Web Application and Test

1. **Start Tomcat Server**

```
$cd /opt/tomcat/bin
$./startup.sh
```

2. **Copy the Web Application to deploy location**

```
cp /home/training/Downloads/apps/TestWebApp.war
/opt/tomcat/webapps/
```

3. Open the browser and type the following url
   http://localhost:8080/TestWebApp/

Lab4: Create a WAR file from the given resources and deploy, test

1. $cd /home/training/Downloads/apps/MyApp

2. $jar -cvf MyApp.war .

3. A file named **MyApp.war** will be generated under MyApp folder

4. $cp /home/training/Downloads/apps/MyApp/MyApp.war
   /opt/tomcat/webapps/

5. Open the browser and type the following url
   http://localhost:8080/MyApp/

Lab4: Change the deploy Location of Applications

1. $cd
   $mkdir apps-deploy

2. edit in /opt/tomcat/conf/server.xml in vi editor
   vi /opt/tomcat/conf/server.xml

3. go to element "**Host**" (around line No.128)and modify the "**appBase**" attribute as given

```
<Host name="localhost"  appBase="/home/training/apps-deploy"
            unpackWARs="true" autoDeploy="true"
            xmlValidation="false" xmlNamespaceAware="false">
```

4. **Restart Tomcat**

```
$/op/tomcat/bin/shutdown.sh
$/opt/tomcat/bin/startup.sh
```

5. **Now deploy TestWebApp.war and test**

```
$ cp /home/training/Downloads/apps/MyApp/MyApp.war
/opt/tomcat/webapps/
```

6. **Open the browser and type the following url**
   http://localhost:8080/TestWebApp/

Lab 5: Change the Context Root of a Web Application

1. vi /opt/tomcat/conf/server.xml
2. **locate the element named "Host"**
3. **Add the element** `<Context/>` **within <Host../> as below:**

```
<Host name="localhost"  appBase="/home/training/apps-deploy"
unpackWARs="true" autoDeploy="true"
 xmlValidation="false" xmlNamespaceAware="false">

<Context path="/TestApp"
docBase="/home/training/Downloads/apps/TestWebApp.war"/>
</Host>
```

7. **Open the browser and type the following url**
   http://localhost:8080/TestApp/

Lab6: Configure to make your application as default one

1. Take TestWebApp as example (/home/training/Downloads/apps/TestWebApp.war)
2. Remove or Rename /opt/tomcat/webapps/ROOT application
3. Add /opt/tomcat/conf/Catalina/localhost/ROOT.xml as follows:

```
<Context
docBase="/home/training/Downloads/apps/TestWebApp.war"
path=""
relodable="true"
/>
```

4. Restart Tomcat and Test

Lab7: Create a DataSource for MySQL Database and test

1. **Copy the jdbc driver for MySQL Database (Already Installed) to tomcat's classpath**

```
$cp /home/training/Downloads/apps/mysql-connector-java-
5.1.22-bin.jar /opt/tomcat/lib
```

2. **Option1**:
   a. Modify /opt/tomcat/conf/context.xml to include the configuration for DataSource.

```
<Context>
 <WatchedResource>WEB-INF/web.xml</WatchedResource>
   <Resource name="jdbc/ds" auth="Container"
   type="javax.sql.DataSource" username="root"
   password="root" driverClassName="com.mysql.jdbc.Driver"
   url="jdbc:mysql://localhost:3306/testdb" maxActive="8"   />
</Context>
```

   b. **Extract the .war file /home/training/Downloads/apps/TestDataSource.war**

```
unzip /home/training/Downloads/apps/TestDataSource.war –d
TestDatasource
```

   c. **Edit TestDatasource/WEB-INF/web.xml as follows:**

```
<resource-ref id="ResourceRef_1243388427265">
   <description>
   </description>
   <res-ref-name>jdbc/ds</res-ref-name>
```

```
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
   </resource-ref>
```

d. Create a .war from the resources inside TestDataSource folder and deploy it.

```
$cd /home/training/Downloads/apps/TestDataSource
$jar –cvf TestDS.war .
$cp TestDS.war /opt/tomcat/webapps/
```

e. Restart Tomcat

```
$/op/tomcat/bin/shutdown.sh
$/opt/tomcat/bin/startup.sh
```

f. Open the browser and type the following url

http://localhost:8080/TestDS/EmpInfo.html

```
Enter 1 to 10 to verify the DataSource
```

3. Option2 :

a. Create a file named "context.xml" using vi and add the elements as given

```
<Context>
 <WatchedResource>WEB-INF/web.xml</WatchedResource>
   <Resource name="jdbc/ds" auth="Container"
   type="javax.sql.DataSource" username="root"
   password="root" driverClassName="com.mysql.jdbc.Driver"
   url="jdbc:mysql://localhost:3306/testdb" maxActive="8"   />
   </Context>
```

b. Copy context.xml file to <YourApp>/META-INF folder

```
$cp context.xml TestDataSource/META-INF/
```

c. Repeat Steps of Option1 (d,e and f)

Lab8: Configure 2 virtual hosts in Tomcat and test

Step1: Modify your /etc/hosts file to include the virtual hosts

```
vi /etc/hosts
<your machine ip>   redhost
<your machine ip>   bluehost
```

Step2: make 2 webapps directories for the 2 virtual hosts

```
mkdir $CATALINA_HOME/redapps
mkdir $CATALINA_HOME/blueapps
```

Step3: Modify $CATALINA_HOME/conf/server.xml as given below:

```
<Engine name="Catalina" defaultHost="ren">
     <Host name="redhost"    appBase="redapps"/>
     <Host name="bluehost"    appBase="blueapps"/>
</Engine>
```

Step3:Deploy Web Applications in the respective webapps directories to test your application.

Step4: Restart tomcat and test virtual hosts

Lab9: Configure Form based Authentication using

1. tomcat-users.xml

a. add the following highlighted lines to `/opt/tomcat/conf/tomcat-users.xml`

```
<role rolename="tomcat"/>
 <role rolename="manager"/>
 <role rolename="role1"/>
 <user username="shantanu" password="welcome1" roles="manager"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
 <user username="both" password="tomcat" roles="tomcat,role1"/>
 <user username="role1" password="tomcat" roles="role1"/>
```

b. Edit /home/training/Downloads/apps/FormAuth/WEB-INF/web.xml and add the following xml block

```
<security-constraint>
        <web-resource-collection>
              <web-resource-name>All resources</web-resource-name>
              <url-pattern>/*</url-pattern>
              <http-method>GET</http-method>
              <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
              <role-name>manager</role-name>
        </auth-constraint>
  </security-constraint>
  <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>java</realm-name>
        <form-login-config>
        <form-login-page>/login.jsp</form-login-page>
        <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
  </login-config>

  <security-role>
        <role-name>manager</role-name>
  </security-role>
```

c. Create a WAR File named FormAuth.war and Deploy

```
$cd /home/training/Downloads/apps/FormAuth
$jar -cvf FormAuth.war .
Scp FormAuth.war /opt/tomcat/webapps/
```

d. Restart Tomcat Server

```
$sh /opt/tomcat/bin/shutdown.sh
$sh /opt/tomcat/bin/startup.sh
```

e. Open a Browser and enter the following url to test the application

http://localhost:8080/FormAuth/demo.jsp

use username and password configured to test

2. Database Realm OR JDBCRealm

a. **Create the necessary tables in MySQL Database (Assumption: mysqld is running)**
   $mysql -u root -p

b. **Create the tables as per the given Schema**

```
create table users (
  user_name         varchar(15) not null primary key,
  user_pass         varchar(15) not null
);

create table user_roles (
  user_name         varchar(15) not null,
  role_name         varchar(15) not null,
  primary key (user_name, role_name)
);
```

c. **Insert some data for users and roles**

d. **Copy the following xml block and paste it to `/opt/tomcat/conf/server.xml` inside `<Host…/>` tag**

```
<Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
driverName="com.mysql.jdbc.Driver"
connectionURL="jdbc:mysql://localhost:3306/tomcatdb"
connectionName="root" connectionPassword="root"
userTable="users" userNameCol="user_name"
userCredCol="user_pass" userRoleTable="user_roles"
roleNameCol="role_name" />
```

e. **Edit /home/training/Downloads/apps/FormAuth/WEB-INF/web.xml and add the following xml block**

```
<security-constraint>
        <web-resource-collection>
              <web-resource-name>All resources</web-resource-name>
              <url-pattern>/*</url-pattern>
              <http-method>GET</http-method>
              <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
              <role-name>manager</role-name>
        </auth-constraint>
  </security-constraint>
  <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>java</realm-name>
        <form-login-config>
        <form-login-page>/login.jsp</form-login-page>
        <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
  </login-config>

  <security-role>
        <role-name>manager</role-name>
  </security-role>
```

f. **Create a WAR File named FormAuth.war and Deploy**

```
$cd /home/training/Downloads/apps/FormAuth
$jar -cvf FormAuth.war .
$ cp FormAuth.war /opt/tomcat/webapps/
```

g. **Restart Tomcat Server**

```
$sh /opt/tomcat/bin/shutdown.sh
$sh /opt/tomcat/bin/startup.sh
```

h. Open a Browser and enter the following url to test the application

http://localhost:8080/FormAuth/demo.jsp

use username and password configured to test

3. LDAP Realm (JNDIRealm) → Follow a separate document supplied.

Lab10: Create a self-signed certificate and use it to configure one way SSL for Tomcat Server

Step1: Create a directory for key database

```
$mkdir /opt/tomcat/ssl
```

Step2: Create the self signed certificate using java 'keytool'

```
$cd /opt/tomcat/ssl
$keytool -genkey  -alias mykey -keystore mykeys.jks -keyalg
RSA -validity 365 -storepass tomcat -keypass tomcat
```

Note: the -storepass and -keypass must be the same for Apache Tomcat

Step3: edit `/opt/tomcat/conf/server.xml` **to include SSL Configuration. Add the following block within** `<Service../>` **element.**

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
maxSpareThreads="75" enableLookups="false"
disableUploadTimeout="true" acceptCount="100" scheme="https"
secure="true" sslProtocol="TLS"
keystoreFile="/opt/tomcat/ssl/mykeys.jks" keystorePass="tomcat" />
```

Lab 11: Compile and build mod_jk for Apache Web Server 2.2 (Assumption: httpd and httpd-devel packages are installed)

1. `$tar -zxvf /home/training/Downloads/tomcat-connectors-1.2.37-src.tar.gz`

2. `$cd /home/training/Downloads/tomcat-connectors-1.2.37-src/native`

3. `$./configure --with-apxs`

4. `$make`

5. After **make** the mod_jk.so file will be generated under /home/training/Downloads/tomcat-connectors-1.2.37-src/native/apache-2.0

6. **This is your expected result**

Lab 12: Configure Apache to use mod_jk with Tomcat Server (Assumption: TestWebApp.war is already deployed to Tomcat)

Step1: install mod_jk.so into Apache Web Server

```
$sudo cp /home/training/Downloads/tomcat-connectors-1.2.37-
src/native/apache-2.0 /etc/httpd/modules
```

Step2: Copy the configuration files to Apache Web Server

```
$sudo cp /home/training/Donwloads/cluster-config/mod_jk.conf
/etc/httpd/conf/

$sudo cp /home/training/Donwloads/cluster-config/mod-jk.conf
/etc/httpd/conf/
```

Step3: Edit /etc/httpd/conf/mod-jk.conf as follows:

```
LoadModule jk_module modules/mod_jk.so
JkWorkersFile conf/workers.properties
JkLogFile logs/mod_jk.log
JkLogLevel info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
JkRequestLogFormat "%w %V %T"
JkMount /TestWebApp/* loadbalancer
```

Step4:Configure workers.properties file as follows:

```
worker.list=loadbalancer,status
worker.nodeA.port=8009
worker.nodeA.host=127.0.0.1
worker.nodeA.type=ajp13
worker.nodeA.lbfactor=1
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=nodeA
# Status worker for managing load balancer
worker.status.type=status
```

Step5: Include mod-jk.conf file to /etc/httpd/conf/httpd.conf

```
$sudo vi /etc/httpd/conf/httpd.conf

At the end of httpd.con file enter
Include conf/mod-jk.conf
```

Step6:Restart Apache Web Server and Restart Apache Tomcat

```
$sudo service httpd restart
$sh /opt/tomcat/bin/shutdown.sh
$sh /opt/tomcat/bin/startup.sh
```

Step:Open a browser and enter the given url

http://localhost/TestWebApp

You should get the Test Page from the application deployed in Tomcat Server

Lab 13: Configure a 2 node cluster with Tomcat Server

Step1: make 2 fresh copies of Tomcat Server Installation

```
$sudo mkdir /opt/cluster
$sudo chown training:training /opt/cluster
$unzip /home/training/Donloads/apache-tomcat-6.0.37  -d
/opt/cluster
$mv /opt/cluster/apache-tomcat-6.0.37 tomcat1
$cp -R /opt/cluster/tomcat1 /opt/tomcat2
```

Step2:Edit and make the following changes in the respective "server.xml" as per the table given below:

Server.xml in	Port Name	Value
tomcat1	shutdown	8105
	ajp	8109
	http	8180
tomcat2	shutdown	8205
	Ajp	8209
	http	8280

Step3: In tomcat1/server.xml and tomcat2/server.xml add the given block of XML just below the <Engine …> tag

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"
         channelSendOptions="8">
     <Manager className="org.apache.catalina.ha.session.DeltaManager"
              expireSessionsOnShutdown="false"
              notifyListenersOnReplication="true"/>
     <Channel className="org.apache.catalina.tribes.group.GroupChannel">
          <Membership
className="org.apache.catalina.tribes.membership.McastService"
                         address="224.0.0.0"
                         port="45564"
                         frequency="500"
                         dropTime="3000"/>
          <Receiver
className="org.apache.catalina.tribes.transport.nio.NioReceiver"
                        address="auto"
                        port="4000"
                        autoBind="100"
                        selectorTimeout="5000"
                        maxThreads="6"/>
```

```
                <Sender
className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
                    <Transport
className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
                </Sender>
                <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
                <Interceptor
className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interc
eptor"/>
          </Channel>
          <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
                 filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.css;.*\.txt;"/>
          <ClusterListener
className="org.apache.catalina.ha.session.ClusterSessionListener"/>
     </Cluster>
```

Step 4: Start tomcat1 and tomcat2

```
$sh /opt/cluster/tomcat1/bin/startup.sh
$sh /opt/cluster/tomcat2/bin/startup.sh
```

Step5: verify the logs for cluster up status and check the respective urls for the started servers

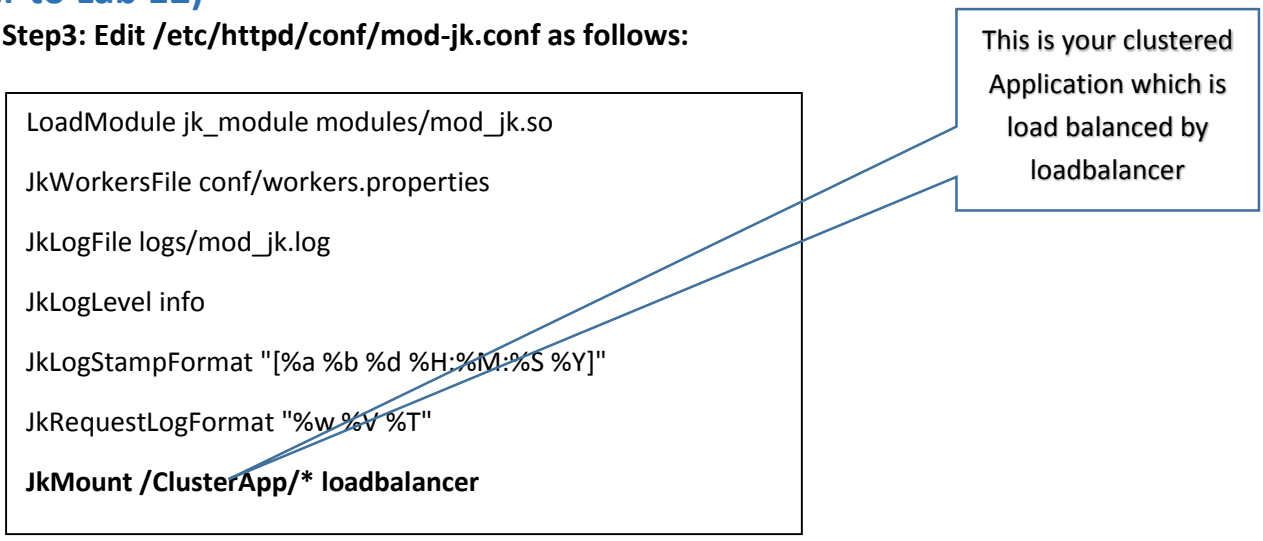
http://localhost:8180

http://localhost:8280

If the cluster is started successfully you should be able to see the commincation log between the servers.

Lab 14: Configure Apache to Load balance Tomcat Cluster (Refer to Lab 12)

Step1: Step3: Edit /etc/httpd/conf/mod-jk.conf as follows:



Step2: Step4:Configure workers.properties file for the cluster as follows:

```
worker.list=loadbalancer,status

worker.nodeA.port=8109

worker.nodeA.host=127.0.0.1

worker.nodeA.type=ajp13

worker.nodeA.lbfactor=1

worker.nodeB.port=8209

worker.nodeB.host=127.0.0.1

worker.nodeB.type=ajp13

worker.nodeB.lbfactor=1

worker.loadbalancer.type=lb

worker.loadbalancer.balance_workers=nodeA,nodeB

# Status worker for managing load balancer

worker.status.type=status
```

Step3:

1. **On tomcat1 (where ajp port is 8109): enter the following value within the <Engine..> elements as given**

   ```
   <Engine name="Catalina" defaultHost="localhost" jvmRoute="nodeA">
   ```

2. **On tomcat2 (where ajp port is 8209): enter the following value within the <Engine..> elements as given**

   ```
   <Engine name="Catalina" defaultHost="localhost" jvmRoute="nodeB">
   ```

Step4: Restart the Cluster you configured in Lab 13 and use the following url to test your clustered application:

```
http://localhost/ClusterApp/
```

Lab 15: Integrate WebSphere MQ with Tomcat Server(Optional) Follw the separate document provided