Tomcat 7 Administering

LAB GUIDE

ул. Николая Шпака 3, Киев, Украина, 03113
тел +38(044) 537-11-71
info@akcent-pro.com
http://www.akcent-pro.com

MODULE 1 . Intro.

At this point you can access your VMs: RHEL 7 and CentOS7. Guest additions installed.

login:	student	root(if needed)
password:	123pass	123pass

Alternatively you will be provided with access to EC2 Amazon service. To connect to your instance request your private key and configure remote agent to connect to ssh service using 'ec2-user' account i.e. *'ssh -i .ssh/ec2-key.rsa ec2-user@54.68.39.104'.*

MODULE 2. Installing Tomcat 7

This lab activity will be splited into several tasks

- RHEL7 JDK installation
- CENTOS7 JDK installation(host machine, docker)
- RHEL 7 Tomcat7 installation
- CENTOS 7 Tomcat7 installation (host machine, docker)

TASK 1. RHEL6 JDK installation

To install JDK you have to bear in mind which JDK's vendor will be chosen and target version.

Next steps will describe Oracle's JDK 7 update 71 installation.

1) Download JDK 7 from Oracle site
2) Unpack saved file
3) Move extracted files to /usr/java dir
4) Changes your OS alternatives to point to new JDK
5) Check result using `*java –version*` command

```
$wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-
cookie" "http://download.oracle.com/otn-pub/java/jdk/7u71-b14/jdk-7u71-linux-
x64.tar.gz" -O "/opt/jdk-7u71-linux-x64.tar.gz"

$cd /opt && tar zxf jdk-7u71-linux-x64.tar.gz && mkdir /usr/java/ && mv
/opt/jdk1.7.0_71 /usr/java && rm /opt/jdk-7u71-linux-x64.tar.gz

#Update alternatives section
$alternatives --install /usr/bin/java java /usr/java/jdk1.7.0_71/jre/bin/java
20000
$alternatives --install /usr/bin/jar jar /usr/java/jdk1.7.0_71/bin/jar 20000
$alternatives --install /usr/bin/javac javac /usr/java/jdk1.7.0_71/bin/javac
20000
$alternatives --install /usr/bin/javaws javaws
/usr/java/jdk1.7.0_71/jre/bin/javaws 20000
$alternatives --set java /usr/java/jdk1.7.0_71/jre/bin/java
$alternatives --set javaws /usr/java/jdk1.7.0_71/jre/bin/javaws
$alternatives --set javac /usr/java/jdk1.7.0_71/bin/javac
$alternatives --set jar /usr/java/jdk1.7.0_71/bin/jar
#check version
$java -version
```

TASK 2. CentOS 7 JDK installation

To install JDK you have to bear in mind which JDK's vendor will be chosen and target version.

Next steps will describe Oracle's JDK 7 update 71 installation.

1) Download JDK 7 from Oracle site
2) Unpack saved file
3) Move extracted files to /usr/java dir
4) Changes your OS alternatives to point to new JDK
5) Check result using `*java –version*` command

To install JDK into docker container you have to prepare special *Dockerfile*

```
$cat Dockerfile

FROM centos
MAINTAINER Vadym Kovalenko vadym.kovalenko@gmail.com

###Declare Env section
ENV JAVA_HOME /usr/java/jdk1.7.0_71
#Install needed SW
RUN yum -y install wget tar
####Install JDK 1.7 Latest
RUN  wget --no-cookies --no-check-certificate --header "Cookie:\
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-
cookie" \
     "http://download.oracle.com/otn-pub/java/jdk/7u71-b14/jdk-7u71-linux-
x64.tar.gz" -O "/opt/jdk-7u71-linux-x64.tar.gz"

RUN cd /opt && tar zxf jdk-7u71-linux-x64.tar.gz && \
    mkdir /usr/java && rm jdk-7u71-linux-x64.tar.gz && \
    mv /opt/jdk1.7.0_71 /usr/java/ && \
    chown -R root:root /usr/java/ && \
    update-alternatives --install /usr/bin/java java
/usr/java/jdk1.7.0_71/jre/bin/java 20000; \
    update-alternatives --install /usr/bin/jar jar
/usr/java/jdk1.7.0_71/bin/jar 20000; \
    update-alternatives --install /usr/bin/javac javac
/usr/java/jdk1.7.0_71/bin/javac 20000; \
    update-alternatives --install /usr/bin/javaws javaws
/usr/java/jdk1.7.0_71/jre/bin/javaws 20000; \
    update-alternatives --set java /usr/java/jdk1.7.0_71/jre/bin/java; \
    update-alternatives --set javaws /usr/java/jdk1.7.0_71/jre/bin/javaws;
\
    update-alternatives --set javac /usr/java/jdk1.7.0_71/bin/javac; \
    update-alternatives --set jar /usr/java/jdk1.7.0_71/bin/jar
```

TASK 3. RHEL7 Tomcat installation

1) Download tomcat7 from official site
2) Check file integrity and md5 checksum
3) Unpack archive content
4) Move\rename tomcat folder if needed

```
$ wget -c -q "http://apache.volia.net/tomcat/tomcat-7/v7.0.57/bin/apache-
tomcat-7.0.57.tar.gz" -O /opt/apache-tomcat-7.0.57.tar.gz
```

```
$ md5sum /opt/apache-tomcat-7.0.57.tar.gz
$ cd /opt && tar zxf apache-tomcat-7.0.57.tar.gz &&\
mv apache-tomcat-7.0.57 tomcat7 && rm apache-tomcat-7.0.57.tar.gz
```

TASK 3. CentOS Tomcat installation(host machine, docker)

CentOS Tomcat 7 installation process doesn't differ from one you performed previously installing RHEL. The difference is to create dockerized tomcat7 container.

$ cat Dockerfile

```
FROM centos
MAINTAINER Vadym Kovalenko vadym.kovalenko@gmail.com

###Declare Env section
ENV LC_ALL C.UTF-8
ENV JAVA_HOME /usr/java/jdk1.7.0_71
#Install needed SW
RUN yum -y install wget tar
####Install JDK 1.7 Latest
RUN  wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-
cookie" \
     "http://download.oracle.com/otn-pub/java/jdk/7u71-b14/jdk-7u71-linux-
x64.tar.gz" -O "/opt/jdk-7u71-linux-x64.tar.gz"


RUN cd /opt && tar zxf jdk-7u71-linux-x64.tar.gz && \
    mkdir /usr/java && rm jdk-7u71-linux-x64.tar.gz && \
    mv /opt/jdk1.7.0_71 /usr/java/ && \
    chown -R root:root /usr/java/ && \
    update-alternatives --install /usr/bin/java java
/usr/java/jdk1.7.0_71/jre/bin/java 20000; \
    update-alternatives --install /usr/bin/jar jar
/usr/java/jdk1.7.0_71/bin/jar 20000; \
    update-alternatives --install /usr/bin/javac javac
/usr/java/jdk1.7.0_71/bin/javac 20000; \
    update-alternatives --install /usr/bin/javaws javaws
/usr/java/jdk1.7.0_71/jre/bin/javaws 20000; \
    update-alternatives --set java /usr/java/jdk1.7.0_71/jre/bin/java; \
    update-alternatives --set javaws /usr/java/jdk1.7.0_71/jre/bin/javaws; \
    update-alternatives --set javac /usr/java/jdk1.7.0_71/bin/javac; \
    update-alternatives --set jar /usr/java/jdk1.7.0_71/bin/jar


RUN wget -c -q "http://apache.volia.net/tomcat/tomcat-7/v7.0.57/bin/apache-
tomcat-7.0.57.tar.gz" \
    -O /opt/apache-tomcat-7.0.57.tar.gz

RUN cd /opt && tar zxf apache-tomcat-7.0.57.tar.gz && \
    mv apache-tomcat-7.0.57 tomcat7 && rm apache-tomcat-7.0.57.tar.gz

ADD tomcat7_run.sh /usr/bin/tomcat7.run.sh

EXPOSE 8080

CMD ["/usr/bin/tomcat7.run.sh"]

$ cat tomcat7.run.sh
```

```
#!/bin/bash
/opt/tomcat7/bin/catalina.sh run
exec tail -f /opt/tomcat7/log/catalina.out
```

MODULE 3. Examining the Tomcat installation directories

This section doesn't contain lab activity by default. But let's practice with docker volumes to pass logs and some applications folders from docker host into container.

So we have to modify docker run command to include needed volumes (webapps and logs for example)

1) Ensure that tomcat image is built on the target system

   #docker images

2) Check if there is no other container which will cause resource overlapping

   #docker ps -a

3) Add additional args for '*docker run'* command

   #docker run -d -v /var/container_data/tomcat/logs:/opt/tomcat7/logs -t tomcat7 -p 8080 -h tomcat

4) Ensure that tomcat engine has been started and manager app is available via hosts 8080 address.

   #netstat –nlpt|grep 8080

MODULE 4. Configuring Tomcat

This time we will modify '*docker run*' command to include tomcat-users.xml, server.xml, web.xml and context.xml.

Edit $CATALINA_HOME/conf/tomcat-users.xml

```
<tomcat-users>
<!--
  NOTE:  By default, no user is included in the "manager-gui" role
required
  to operate the "/manager/html" web application.  If you wish to use
this app,
  you must define such a user - the username and password are
arbitrary.
-->
<!--
  NOTE:  The sample user and role entries below are wrapped in a
comment
  and thus are ignored when reading this file. Do not forget to remove
  <!.. ..> that surrounds them.
-->
  <role rolename="manager-gui"/>
  <user username="tomcat" password="tomcat" roles="manager-gui"/>
</tomcat-users>
```

5) Ensure that tomcat image is built on the target system

```
#docker images
```

6) Check if there is no other container which will cause resource overlapping

```
#docker ps -a
```

7) Add additional args for '*docker run'* command

```
#docker run -d -v
/var/container_data/tomcat/conf/tomcat_users.xml:/opt/tomcat7/conf/
tomcat_users.xml -v
/var/container_data/tomcat/conf/server.xml:/opt/tomcat7/conf/server.xml
-t tomcat7 -v
/var/container_data/tomcat/conf/tomcat_users.xml:/opt/tomcat7/conf/
tomcat_users.xml -v
/var/container_data/tomcat/conf/web.xml:/opt/tomcat7/conf/web.xml
-v
/var/container_data/tomcat/conf/context.xml:/opt/tomcat7/conf/context.x
ml -v
/var/container_data/tomcat/conf/setenv.sh:/opt/tomcat7/bin/setenv.sh -p
8080 -h tomcat
```

MODULE 5. Tomcat Valves

1) The *Crawler Session Manager Valve* will effectively gather requests made by the same user agent from the same IP address into a single session. Enabling the valve is as simple as adding the following into the conf/server.xml file:

```
<Host name="localhost" appBase="webapps" unpackWARs="true"
autoDeploy="true">
<Valve
className="org.apache.catalina.valves.CrawlerSessionManagerValve"
crawlerUserAgents=".*ServerNanny.*|.*CFSCHEDULE.*"
sessionInactiveInterval="600"/>
</Host>
```

To verify that the valve is working as expected, make a few requests of the server:

```
$ for i in {1..100}; do curl -s -j -o /dev/null -A 'ServerNanny'
'http://tomcat:8080/sample/hello.jsp' && echo $i; done
```

2) Change $CATALINA_HOME/conf/server.xml

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
  directory="logs"
  prefix="localhost_access_log" suffix=""
  pattern="%h %l %u %t %T %s %b" resolveHosts="false"/>
```

3) Change $CATALINA_HOME/conf/context.xml adding following Valve and check access in your browser.

```
<Context
    <Valve className="org.apache.catalina.valves.RemoteAddrValve"
        allow="127\.0\.0\.1|172\.17\.42\.1"
    />
</Context>
```

or

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve" deny="127.*"/>
```

This valve entry denies access to the assigned container for all client IP addresses that begin with 127. If I assign this valve entry to the host container localhost, then all clients with an IP address beginning with 127 will see a http status 403 - Forbidden page.

Attribute	Description	Required?
className	The Java class of the valve. This must be org.apache.catalina. valves.AccessLogValve.	Yes
condition	Turns conditional logging on. If set, the access log valve logs requests only if ServletRequest.getAttribute() is null. For example, if this value is set to userId, a particular request will be logged only if ServletRequest.getAttribute("userId")	No
directory	The directory where the log files will be placed. This is usually relative to the CATALINA_HOME, but you can specify an absolute path instead. The default is logs.	No
prefix	The prefix added to the name of the log file.	No
resolveHosts	Determines if the log will contain hostnames via a reverse DNS lookup. This can take significant time if enabled. The default is false.	No
rotatable	Determines if log rotation should occur. If false, this file is never rotated, and the *fileDateFormat* attribute is ignored. Use this attribute with caution, because the log file could grow very large indeed. The default is true.	No
suffix	The extension added to the name of the log file.	No

Attribute	Description	Required?
fileDateFormat	Allows a customized date format in the access log filename. The date format also decides how often the file is rotated. If you want to rotate every hour, then set this value to yyyy-MM-dd.HH.	No
pattern	Specifies the format used in the log. You can customize the format, or you can use common or combined as the format (the common format, plus the referrer and user agent are logged). To customize the format, you can use any of the following patterns interspersed with a literal string: %a: Inserts remote IP address. %A: Inserts local IP address (of URL resource). %b: Inserts a bytes sent count, excluding HTTP headers, and shows - if zero. %B: Inserts a bytes sent count, excluding HTTP headers. %D: Time taken to process the request in milliseconds. %h: Inserts remote hostname (or IP address if the resolveHosts attribute is set to false). %H: Inserts the request protocol (HTTP). %l: Inserts remote logical user name (always -). %m: Inserts request method such as GET and POST. %p: Inserts the local TCP port where this request is received. %q: Inserts the query string of this request. %r: Inserts the first line of the request. %s: Inserts the HTTP status code of the response. %S: Inserts the user session ID. %t: Inserts the date and time in common log file format. %T: Inserts the time taken to process the request, in seconds. %u: Inserts the remote user that has been authenticated (if there is none, it's -). %U: Inserts the URL path of the request. %v: Inserts the name of the local virtual host from the request. %{xxx}i: Use this for incoming headers, where xxx is the header. %{xxx}c: Use this for a specific cookie, where xxx is the name of the cookie. %{xxx}r: Use this for ServletRequest attributes, where xxx is the attribute. %{xxx}s: Use this for HttpSession attributes, where xxx is the attribute. The default is common, which is %h %l %u %t "%r" %s %b.	No

MODULE 6. Memory management and JMX monitoring

This listener requires catalina-jmx-remote.jar to be placed in $CATALINA_HOME/lib. This jar may be found in the extras directory of the binary download area.

The **JMX Remote Lifecycle Listener** fixes the ports used by the JMX/RMI Server making things much simpler if you need to connect *jconsole* or a similar tool to a remote Tomcat instance that is running behind a firewall. Only these ports are configured via the listener. The remainder of the configuration is via the standard system properties for configuring JMX.

```
#!/bin/bash
$ wget -c -q "http://apache.ip-connect.vn.ua/tomcat/tomcat-
7/v7.0.57/bin/extras/catalina-jmx-remote.jar" -O /opt/catalina-jmx-remote.jar
$ mv /opt/catalina-jmx-remote.jar /opt/tomcat7/lib
```

To connect with *jConsole*, Tomcat need to enable the JMX options. To solve it, create a *$CATALINA_HOME/bin/setenv.sh*, and put the following values :

$CATALINA_HOME/bin/setenv.sh

```
export JAVA_OPTS="-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9999
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false"
```

Following params are optional.

```
-Dcom.sun.management.jmxremote.password.file=$CATALINA_BASE/conf
/jmxremote.password
-Dcom.sun.management.jmxremote.access.file=$CATALINA_BASE/conf
/jmxremote.access
-Dcom.sun.management.jmxremote.ssl=false
```

Restart Tomcat, now you can connect to this Tomcat via *jConsole* in port 9999. If this listener was configured in server.xml as:

```
<Listener className="org.apache.catalina.mbeans.JmxRemoteLifecycleListener"
          rmiRegistryPortPlatform="10001" rmiServerPortPlatform="10002" />
```

with the following system properties set (e.g. in setenv.sh):

Create $CATALINA_BASE/conf/jmxremote.password containing:

```
admin letmein
```

$CATALINA_BASE/conf/jmxremote.access containing:

```
admin readwrite
```

then opening ports 10001 (RMI Registry) and 10002 (JMX/RMI Server) in your firewall would enable jconsole to connect to a Tomcat instance running behind a firewall using a connection string of the form:

```
service:jmx:rmi://<hostname>:10002/jndi/rmi://<hostname>:10001/jmxrmi
```

Garbage collection performance is a good metric to use both because it can heavily impact things like response time and response throughput and because it's easy to measure, even in a production system. To measure the performance of garbage collection we simply enable garbage collection logging.

This is done by adding the following JVM options to the CATALINA_OPTS variable in the *bin/setenv.sh* file for your Tomcat instance.

- *-Xloggc:$CATALINA_HOME/logs/gc.log or Xloggc:%CATALINA_HOME%/logs/gc.log*
- *-XX:+PrintHeapAtGC*
- *-XX:+PrintGCDetails*
- *-XX:+PrintGCTimeStamps*
- *-XX:-HeapDumpOnOutOfMemoryError*

MODULE 7. Logging

```
#!/bin/bash
wget -c -q "http://apache.volia.net/tomcat/tomcat-
7/v7.0.57/bin/extras/tomcat-juli-adapters.jar" -O /opt/tomcat-juli-
adapters.jar
wget -c -q "http://apache.volia.net/tomcat/tomcat-
7/v7.0.57/bin/extras/tomcat-juli.jar" -O /opt/tomcat-juli.jar
mv /opt/tomcat-juli-adapters.jar /opt/tomcat7/lib/ && mv /opt/tomcat-
juli.jar /opt/tomcat7/lib/
mv /opt/tomcat7/conf/logging.properties
/opt/tomcat7/conf/logging.properties.old
```

It is possible to rotate the catalina.out log, but it is not controlled by the standard logging.properties or log4j.properties files.

The catalina.out log is stderr and stdout piped to a file. If you want to rotate this log file you will need to use a log rotation program like rotatelogs or cronolog. Then just pipe to the log rotation program rather than the file.

Here is an example of how to do this with cronolog. Using another program would be very similar, the command in step #3 would just be slightly different.

1.) Edit the bin/catalina.sh file.

2.) Find the following line and comment it out.

```
touch "$CATALINA_BASE"/logs/catalina.out
```

Note that in later versions of Tomcat this may look like:

```
touch "$CATALINA_OUT"
```

3.) Find the following line (there should be two instances of this line, replace both)

```
>> "$CATALINA_BASE"/logs/catalina.out 2>&1 &
```

and replace it with this line

```
2>&1 |/usr/bin/cronolog "$CATALINA_BASE/logs/catalina-%Y-%m-%d.out" &
```

Note that in later versions of Tomcat the line that needs to be replaced may look like:

```
>> "$CATALINA_OUT" 2>&1 &
```

4.) Save bin/catalina.sh

5.) Restart Tomcat.

MODULE 8. Connecting databases with Tomcat applications

Firstly we will configure a global connection pool editing file: conf/server.xml

```
<GlobalNamingResources>
  <Resource type="javax.sql.DataSource"
            name="jdbc/TestDB"
            factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
            driverClassName="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/mysql"
            username="mysql_user"
            password="mypassword123"
/>
 </GlobalNamingResources>
```

You then create a ResourceLink element to make the pool available to the web applications. If you want the pool available to all applications under the same name, the easiest way is to edit the **File: conf/context.xml**

```
<Context>
  <ResourceLink type="javax.sql.DataSource"
                name="jdbc/LocalTestDB"
                global="jdbc/TestDB"
/>
 <Context>
```

*Note, that if you don't want a global pool, move the **Resource** element from server.xml into your **context.xml** file for the web application.*

And to retrieve a connection from this configuration, the simple Java code looks like

```
Context initContext = new
 InitialContext();
   Context envContext  = (Context)initContext.lookup("java:/comp/env");
   DataSource datasource = (DataSource)envContext.lookup("jdbc/LocalTestDB");
   Connection con = datasource.getConnection();
```

Same in Java

We can achieve the same configuration using just Java syntax.

```
DataSource ds = new DataSource();
   ds.setDriverClassName("com.mysql.jdbc.Driver");
   ds.setUrl("jdbc:mysql://localhost:3306/mysql");
   ds.setUsername("root");
   ds.setPassword("password");
```

MODULE 9. Security

- Create a tomcat user/group
- Download and unpack the core distribution (referenced as **CATALINA_HOME** from now on)
- Change **CATALINA_HOME** ownership to tomcat user and tomcat group
- Change files in **CATALINA_HOME**/conf to be readonly (400)
- Make sure tomcat user has read/write access to /tmp and write (300 - yes, only write/execute) access to **CATALINA_HOME**/logs

- Remove everything from **CATALINA_HOME**/webapps (ROOT, balancer, jsp-examples, servlet-examples, tomcat-docs, webdav)

- Remove everything from **CATALINA_HOME**/server/webapps (host-manager, manager). Note that it can be useful to keep the manager webapp installed if you need the ability to redeploy without restarting Tomcat. If you choose to keep it please read the section on Securing the Manager WebApp.

- Remove **CATALINA_HOME**/conf/Catalina/localhost/host-manager.xml and **CATALINA_HOME**/conf/Catalina/localhost/manager.xml (again, if you are keeping the manager application, do not remove this).

- Make sure the default servlet is configured **not** to serve index pages when a welcome file is not present. In **CATALINA_HOME**/conf/web.xml

```
<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>false</param-value>  <!-- make sure this is false -->
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

- Remove version string from HTTP error messages by repacking **CATALINA_HOME**/server/lib/catalina.jar with an updated ServerInfo.properties file. Note that making this change may prevent Lambda Probe (popular Tomcat monitoring webapp) to initialise as it cannot determine the Tomcat version.

  $ cd CATALINA_HOME/server/lib

  $jar xf catalina.jar org/apache/catalina/util/ServerInfo.properties
  update ServerInfo.properties by changing server.info line to server.info=Apache Tomcat

  Repack catalina.jar

  `$jar uf catalina.jar org/apache/catalina/util/ServerInfo.properties`
  remove CATALINA_HOME/server/lib/org (created when extracting the ServerInfo.properties file)

- Replace default error page (default is stacktrace) by adding the following into **CATALINA_HOME**/conf/web.xml. The default error page shows a full stacktrace which is a disclosure of sensitive information. Place the following within the *web-app* tag (after the *welcome-file-list* tag is fine). *The following solution is not ideal as it produces a blank page because Tomcat cannot find the file specified, but without a better solution this, at least, achieves the desired result. A well configured web application will override this default in CATALINA_HOME/webapps/APP_NAME/WEB-INF/web.xml so it won't cause problems.*

```
<error-page>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/error.jsp</location>
</error-page>
```

- Rename **CATALINA_HOME**/conf/server.xml to **CATALINA_HOME**/conf/server-original.xml and rename **CATALINA_HOME**/conf/server-minimal.xml to **CATALINA_HOME**/conf/server.xml. The minimal configuration provides the same basic configuration, but without the nested comments is much easier to maintain and understand. Do not delete the original file as the comments make it useful for reference if you ever need to make changes - e.g. enable SSL.

- Replace the server version string from HTTP headers in server responses, by adding the server keyword in your Connectors in **CATALINA_HOME**/conf/server.xml

```
<Connector port="8080" ...
           server="Apache" />  <!-- server header is now Apache -->
```

- Start Tomcat, deploy your applications into **CATALINA_HOME**/webapps and hope it works!

Protecting the Shutdown Port

- if you are running a publicly accessible server make sure you prevent external access to the shutdown port by using a suitable firewall.
- change the shutdown command in **CATALINA_HOME**/conf/server.xml and make sure that file is only readable by the tomcat user.

```
<Server port="8005" shutdown="ReallyComplexWord">
```

Logging

As of tomcat 5.5 logging is now handled by the commons-logging framework allowing you to choose your preferred logging implementation - log4j or standard JDK logging. By default the standard JDK logging is used (or a compatible extension called juli to be more precise), storing daily log files in **CATALINA_HOME**/logs.

By default additional webapp log entries are added to C**ATALINA_HOME**/logs/ catalina.YYYY-MM-DD.log and System.out/System.err are redirected to **CATALINA_HOME**/logs/catalina.out. To place webapp log entries in individual log files create a *logging.properties* file similar to the following within **CATALINA_HOME**/ webapps/*APP_NAME*/WEB-INF/classes (change the *APP_NAME* value to create a unique file for each webapp)

```
handlers = org.apache.juli.FileHandler, java.util.logging.ConsoleHandler
org.apache.juli.FileHandler.level = ALL
```

```
org.apache.juli.FileHandler.directory = ${catalina.base}/logs
org.apache.juli.FileHandler.prefix = APP_NAME.
```

Sample Configuration - Good Security

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11Protocol" SSLEnabled="true"
               maxThreads="150" scheme="https" secure="true"
                        keystoreFile="..\ssl\keystore" keystorePass="yourpasswordgoeshere"
               clientAuth="false" sslProtocol="TLSv1.2" sslEnabledProtocols="TLSv1.2,TLSv1.1,TLSv1"
                        ciphers="TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
                        TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,
                        TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
                        TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256,
                        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
                        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
                        TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384,
                        TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
                        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
                        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
                        TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256,
                        TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA"
        />
```

MODULE 10. Performance

Using apache configured as FE and Tomcat configured as BE server invoke some kind of load tests to check key statistic value like requests per second, longest request, total time taken etc.

*$ab -k -n 1000 -c 100 http://tomcathost:80/*sample/hello.jsp

*$ab -k -n 1000 -c 100 http://tomcathost:8080/*sample/image_9k.jpg

Change the type of Tomcat connector and rerun tests. Compare results.

To change connector type edit server.xml file finding <Connector> and specifying `protocol` field section.

`org.apache.coyote.http11.Http11Protocol` - blocking Java connector
`org.apache.coyote.http11.Http11NioProtocol` - non blocking Java connector
`org.apache.coyote.http11.Http11AprProtocol` - the APR/native connector.

`org.apache.coyote.ajp.AjpProtocol` - blocking Java connector
`org.apache.coyote.ajp.AjpNioProtocol` - non blocking Java connector.
`org.apache.coyote.ajp.AjpAprProtocol` - the APR/native connector.

<!-- The stock HTTP JIO connector. -->

<Connector port="8080" protocol="HTTP/1.1" maxThreads="150" connectionTimeout="20000" redirectPort="8443" />

<!-- The HTTP APR connector. -->

<Connector port="8080"

protocol="org.apache.coyote.http11.Http11AprProtocol" enableLookups="false" redirectPort="8443" connectionTimeout="20000"/>

<!-- HTTP NIO connector. -->

<Connector port="8080"

maxThreads="150" connectionTimeout="20000" redirectPort="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"/>

<!-- AJP JIO/APR connector, switched by setting LD_LIBRARY_PATH. -->

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

<!-- AJP NIO connector. -->

<Connector protocol="AJP/1.3" port="0" channelNioSocket.port="8009" channelNioSocket.maxThreads="150" channelNioSocket.maxSpareThreads="50" channelNioSocket.minSpareThreads="25" channelNioSocket.bufferSize="16384"/>

MODULE 12. Running Tomcat behind Apache httpd

At this this point we will configure mod_jk that is provided by apache as source. Will compile it and move resulted mod_jk.so file to /etc/httpd/modules directory as prereq for FE->BE data transfer.

install_mod_jk.sh Listing

```
#!/bin/bash
yum -y install httpd httpd-devel tar gcc make wget
chkconfig --levels 235 httpd on
wget -q "http://www.apache.org/dist/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.40-src.tar.gz" -O /usr/src/tomcat-connectors-1.2.40-src.tar.gz

cd /usr/src/ && tar zxf tomcat-connectors-1.2.40-src.tar.gz; \
   cd tomcat-connectors-1.2.40-src/native && ./configure --with-apxs=/usr/bin/apxs ;\
   make && cp apache-2.0/mod_jk.so /etc/httpd/modules/ ;\
   echo "LoadModule jk_module modules/mod_jk.so" >> /etc/httpd/conf.modules.d/00-base.conf;\
rm /usr/src/tomcat-connectors-1.2.40-src.tar.gz
#cp 00-jk.conf /etc/httpd/conf.modules.d/00-jk.conf  #supposed to be exec manually
#cp worker.properties /etc/httpd/conf/               #supposed to be exec manually
```

MODULE 13. Cluster setup.

The cluster configuration consists of Apache configuration as a FE service and Tomcat server as BE service.

Following listing of 00-jk.conf defines how apache will handle all requests coming to /clusterjsp path.

```
LoadModule jk_module modules/mod_jk.so

<IfModule jk_module>

   JkWorkersFile /etc/httpd/conf/worker.properties
   JkLogFile logs/mod_jk.log
   JkLogStampFormat "[%b %d %Y - %H:%M:%S] "
   JkRequestLogFormat "%w %V %T"
   JkLogLevel info

   JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

   Alias /sample "/opt/tomcat7/webapps/sample"

   <Directory "/opt/tomcat7/webapps/sample">
      AllowOverride None
      Allow from all
   </Directory>

   <Location /*/WEB-INF/*>
      deny from all
   </Location>

   JkMount /clusterjsp/* cluster

</IfModule>
```

worker.properties file defines target endpoint configured at tomcat side:

```
workers.properties
worker.list = cluster

worker.tomcat1.port=8009
worker.tomcat1.host=localhost
worker.tomcat1.type=ajp13
worker.tomcat1.lbfactor=1

worker.tomcat2.port=8010
worker.tomcat2.host=localhost
worker.tomcat2.type=ajp13
worker.tomcat2.lbfactor=1

worker.cluster.type=lb
```

```
worker.cluster.balance_workers=tomcat1,tomcat2
worker.cluster.sticky_session=1
```

server.xml for unicast addresses will be similar to following listing.

```
<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">
        <Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"
              address="auto"
              port="4001"
              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.MessageDispatch15Interceptor"/>
          <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpPingInterceptor"
staticOnly="true"/>
          <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
          <Interceptor
className="org.apache.catalina.tribes.group.interceptors.StaticMembershipInterceptor">

              <Member className="org.apache.catalina.tribes.membership.StaticMember"
port="4000"
                    host="127.0.0.1" uniqueId="{127,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0}"/>
          <Member      className="org.apache.catalina.tribes.membership.StaticMember"
port="4001"
                    host="127.0.0.1" uniqueId="{127,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0}"/>
          </Interceptor>

        </Channel>

        <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
            filter=""/>
        <Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

        <Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"
              tempDir="/tmp/war-temp/"
              deployDir="/tmp/war-deploy/"
```

```
            watchDir="/tmp/war-listen/"
            watchEnabled="false"/>

        <ClusterListener
    className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener"/>
        <ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>
      </Cluster>
```