JWS31 Layout

This document is for JWS3.1 layout and securing tomcat application server
The Server.xml config file should be having below

- Tomcat server port including http and https
- Entries for securing http cookies
- LDAP Configuration Details
- Userdatabase realm in case of tomcat-users.xml
- SSLEnabled Protocol,SSL certificate and cipher details
- Proxy list and sticky session info for mod cluster
- Jvmroute entry to restrict session info
- Webseal configuration details
- Webcontainer thread count and connection timeout info

The context.xml config file should be having below

- DB and MQ configuration details
- Connection pool Min and Max values
- Restrict jsessionid session information in case of path info
- Secure http cookie

The catalina.policy and java.policy config files should be having below
These files contain default set of security policies

The tomcat-users.xml config file should be having below
This file contains roles to use manager application to manage tomcat applications etc.,

- **manager-gui** — Access to the HTML interface.
- **manager-status** — Access to the "Server Status" page only.
- **manager-script** — Access to the tools-friendly plain text interface that is described in this document, and to the "Server Status" page.
- **manager-jmx** — Access to JMX proxy interface and to the "Server Status" page.

The WEB.XML config file should be having below
The application details will be getting into web.xml file
Tomcat Matrix Table

	Server.xml	Context.xml	Catalina.policy	Java.policy	Web.xml	Tomcat-users.xml	setenv.sh	keystore	truststore
Container Port Details	x								
Container thread pool	x								
MQ,Datasource connection pool		x							
Enable security with ldap	x								
Tomcat Security Policy			x	x					
Ciphers&ssl enabled protocols	x								
MQ,Datasource config details		x							
Application details					x				
User role for Manager application						x			
Secure Http Cookie	x	x							
Limit JSession Information	x	x							
Mod_cluster details in case of Proxylist,stickysession	x								
SSLcertificatekeyfile and certfile	x								
Webseal config	x								
container connection timeout	x								
Permsize,min and max heap size							x		
ClassPath							x		
Server cert								x	
CA cert chain									x

ClassPath,Permsize,min and max heap size Details
The classpath, permsize and min/max heap size should be mentioned in the setenv.sh file
Please find example as below
export CLASSPATH=/opt/jboss/jws31/Properties/<Application ID>/<Application ID>_Properties/:/opt/jboss/jws31/Properties/<Application ID>/<Application ID>_Properties/
export JAVA_OPTS="-Dlog4j.debug=true -Djavax.net.ssl.keystore=/opt/jboss/jws31/keystores/.keystore -Djavax.net.ssl.keystorePassword=xxxx -Djavax.net.debug=ssl:handshake -XX:PermSize=<value>m -Xmx<value><m/g> -Xms<value><m/g> -XX:+UseConcMarkSweepGC -Djavax.net.ssl.trustStore=opt/jboss/jws31/keystores/.keystore -Djavax.net.ssl.trustStorePassword=xxxx -XX:+UnlockCommercialFeatures -XX:+FlightRecorder
-Xmx is max heap size
-Xms is min heap size

Securing HTTP cookie

Update below entry in the server.xml and context.xml files

Implement HttpOnly flag

1. Go to tomcat/conf folder
2. Modify Context syntax as shown below in context.xml

```
useHttpOnly="true"
```

```
<Context useHttpOnly="true">

    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->

    <!-- Uncomment this to enable Comet connection tacking (provides events
         on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->

</Context>
```

Implement Secure flag

- Go to tomcat/conf folder
- Add below parameter in server.xml under Connector port syntax

```
secure="true"
```

- Your Connector port should look like something as below

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" secure="true"/>
```

Reference:
https://geekflare.com/secure-cookie-flag-in-tomcat/

Limit Session Information
Update the following in the server.xml and context.xml files
The "jvmRoute" entry in the server.xml file will be part of JSessionID information. Hence we should make sure we don't mention tomcat hostname against jvmRoute in server.xml file. We should mention Tomcat JVM name against this jvmRoute entry.
Update the following in **context.xml** file
sessionCookiePath=/
sessionCookiePath The path to be used for all session cookies created for this context. If set, this overrides any path set by the web application. If not set, the value specified by the web application will be used, or the context path used if the web application does not explicitly set one. To configure all web application to use an empty path (this can be useful for portlet specification implementations) set this attribute to / in the global CATALINA_BASE/conf /context.xml file.
Reference:
https://examples.javacodegeeks.com/enterprise-java/tomcat/tomcat-context-xml-configuration-example/
https://tomcat.apache.org/tomcat-8.0-doc/config/context.html#Common_Attributes

Adding additional SSL enabled Protocol and ciphers
Add below sslEnbledProtocols and ciphers in the server.xml file based on security standard.
Example:
sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2,SSLv2Hello"
ciphers="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_256_GCM_SHA384,
TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA"/>
Secure LDAP config
The secure LDAP config should be entered in server.xml file. Make sure we use SHA2 ldap server which is prime-sldap.primetherapeutics.com
Example
<Realm className="org.apache.catalina.realm.JNDIRealm" debug="99"
connectionURL="ldap://prime-sldap.primetherapeutics.com:636"
authentication="simple"
connectionName="CN=SVCWASADQA,OU=Service Accounts,DC=primetherapeutics,DC=com"
connectionPassword="xxxxxxxxxx"
userBase="OU=Service Accounts,DC=primetherapeutics,DC=com"
userSearch="sAMAccountName={0}"
userSubtree="true"

roleBase="OU=Service Accounts,DC=primetherapeutics,DC=com"
roleName="CN"
roleSearch="(memberOf={0})"
roleSubtree="true"
resourceName="UserDatabase"
referrals="follow"
allRolesMode="authOnly"
protocol="ssl"
/>

Restricting Back Door Access

If we allow http port accessed only on localhost(127.0.0.1) in server.xml, other users can't access tomcat application by hitting tomcat server directly after bypassing F5,apache webserver,webseal etc., from their laptop or machine.
We need http port to be allowed to access /manager/status/all page for administrative purpose.
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" address="127.0.0.1"/>
You can also add the RemoteAddrValve to the host within your server.xml, for example:

```
<Host name="localhost" appBase="webapps"
unpackWARs="true" autoDeploy="true">
...
<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="127\.0\.0\.1,httpdip" />
</Host>
```

Setting allow to a value like above would then only allow requests if they come through httpd or if they come locally. Otherwise, no client can then access tomcat directly.
We can use curl command to access /manager/status/all page.
Example
curl -u username:password! http://localhost:8080/manager/status/all
we should make sure user has corresponding role (i.e. manager-jmx or manager-gui) in TOMCAT_HOME/conf/tomcat-users.xml

WebSeal configuration

Update the following in server.xml file
<Valve className="com.ibm.tivoli.integration.am.catalina.valves.AMTomcatValve" debugTrace="true" fallThrough="true" />

Webcontainer ports,Maxthreads and ConnectionTimeout configuration

We reserve 10 ports for each tomcat container. The server.xml gets server port, webcontainer http,https and AJP ports. We define maxthreads, connectionTimeout in the server.xml as below
Example :

```
<Server port="8005" shutdown="SHUTDOWN">
……..
……..
……
……..
<Connector port="9090" protocol="HTTP/1.1" maxThreads="75"
connectionTimeout="20000"
redirectPort="8443" address="127.0.0.1"/>
<Connector port="8443" scheme="https" secure="true" SSLEnabled="true" SSLCertificateKeyFile="/opt/jboss/keystores/.keystore" SSLPassword="xxxxxx"
keyAlias="servercert" SSLProtocol="TLSv1"/>
……
……
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

The confluence page will be updated with Tomcat farm name details, port range allocated to each farm, servers assigned to each farm, application which belongs to each farm, ports assigned to each application and F5 URL details etc.,

Datasoure,ConnectionPool Settings and MQ Configuration

The datasource and MQ configuration should go to context.xml. We define min,max connection pool details in the context.xml file
Please find below datasource configuration as an example in context.xml file

```
<Resource
type="javax.sql.DataSource"
name="jdbc/primeoneDS"
factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
driverClassName="com.primetherapeutics.primeone.db2.RetryableDB2Driver"
url="jdbc:db2://LXUDBM006Q:60016/PO_QA1:retrieveMessagesFromServerOnGetMessage=true;currentSchema=PO;"
username="mduser1q"
password="xxxxxxxxxxx "
initialSize="0"
maxActive="50"
maxIdle = "30"
minIdle="20"
timeBetweenEvictionRunsMillis="30000"
minEvictableIdleTimeMillis="60000"
testOnBorrow="true"
validationQuery="VALUES 1"
validationInterval="30000"
removeAbandoned="true"
removeAbandonedTimeout="60"
logAbandoned="true"
abandonWhenPercentageFull="60"
jdbcInterceptors="org.apache.tomcat.jdbc.pool.interceptor.ResetAbandonedTimer"/>
```

Apache WebServer

Mod Cluster advantages

mod_cluster boasts the following advantages over other httpd-based load balancers mod_jk and mod_proxy

- **Dynamic configuration of httpd workers**

Traditional httpd-based load balancers require explicit configuration of the workers available to a proxy.Like mod_jk maintain the static worker list in worker.properties but modcluster allow dynamic worker list means whenever the user load increases you can add Application server nodes dynamically to balance the load. In mod_cluster, the bulk of the proxy's configuration resides on the application servers. The set of proxies to which an application server will communicate is determined either by a static list or using dynamic discovery via the advertise mechanism.The application server relays life-cycle events (e.g. server startup/shutdown) to the proxies allowing them to effectively auto-configure themselves. Notably, the graceful shutdown of a server will not result in a fail-over response by a proxy, as is the case with traditional httpd-based load balancers.

- **Server-side load balance factor calculation**

In contrast with traditional httpd-based load balancers, mod_cluster uses load balance factors calculated and provided by the application servers, rather than computing these in the proxy. Consequently, mod_cluster offers a more robust and accurate set of load metrics than is available from the proxy.

- **Fine grained web-app lifecycle control**

Traditional httpd-based load balancers do not handle web application undeployments particularly well.From the proxy's perspective requests to an undeployed web application are indistinguishable from a request for an non-existent resource, and will result in 404 errors. In mod_cluster, each server forwards any web application context lifecycle events (e.g. web-app deploy/undeploy) to the proxy informing it to start/stop routing requests for a given context to that server.

mod_cluster boasts the following advantages over mod_jk

- **AJP is optional**

Unlike mod_jk, mod_cluster does not require AJP. httpd connections to application server nodes can use HTTP, HTTPS, or AJP.
Load Balancing and session stickiness with MOD_CLUSTER
The virtualhost section of httpd.conf file should be having cluster name of particular application,SSLCertificatekeyfile,stickysession name,mod_cluster section etc.,
<VirtualHost devsampleapp.primetherapeutics.com:443>
ProxyPass / balancer://SampleAppCluster stickysession=JSESSIONID|jsessionid nofailover=On
ProxyPassReverse / balancer://SampleAppCluster
ProxyPreserveHost On
SSLEngine on
SSLProtocol all -SSLv2
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
SSLCertificateFile /etc/httpd/conf/openssl/pki/tls/certs/localhost.crt
SSLCertificateKeyFile /etc/httpd/conf/openssl/pki/tls/private/localhost.key
<Location /sample>
Require all granted
</Location>
<Location /mod_cluster_manager>
SetHandler mod_cluster-manager
Order deny,allow
Deny from all
Allow from all # change this to match your network setup
</Location>
</VirtualHost>
The mod_cluster.conf file should be having dedicated virtulhost section for an application cluster with unique port number. We should mention application cluster name in a mod_cluster virtualhost section and we can mention list of tomcat server ip addresses in the same virtualhost section if needed.
<VirtualHost 0.0.0.0:6666>
<Directory "/">

1. List of tomcat ip addresses to be allowed
   #Require ip 10.10.10.2 10.10.10.3
   Require all granted
   </Directory>
   KeepAliveTimeout 60
   MaxKeepAliveRequests 0
   ManagerBalancerName SampleAppCluster
   EnableMCPMReceive On
   ServerAdvertise Off
   </VirtualHost>

   The below mod_cluster-manger section in the mod_cluster.conf helps us to access the mod_cluster-manager page to initiate bleed off traffic or stop traffic to tomcat app server from apache webserver.
   <Location /mod_cluster-manager>
   SetHandler mod_cluster-manager
   Require ip 127.0.0.1
   </Location>

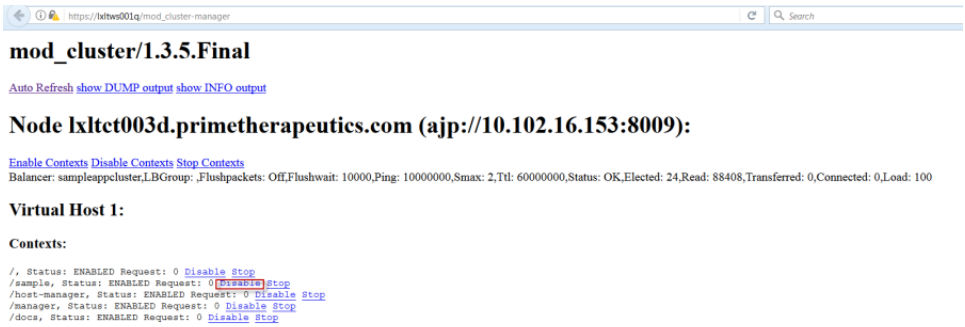
   In the above mod_cluster-manager section, mention the required ip address of servers(example prod IHS8.5 webservers) on which we can access the mod_cluster-manager page.
   Or

Please refer below RedHat resources to integrate ldap with apache webserver if we want to use middleware ldap group/users to access mod_cluster-manager page.
https://access.redhat.com/solutions/20284
https://httpd.apache.org/docs/2.4/mod/mod_authnz_ldap.html#requiredirectives

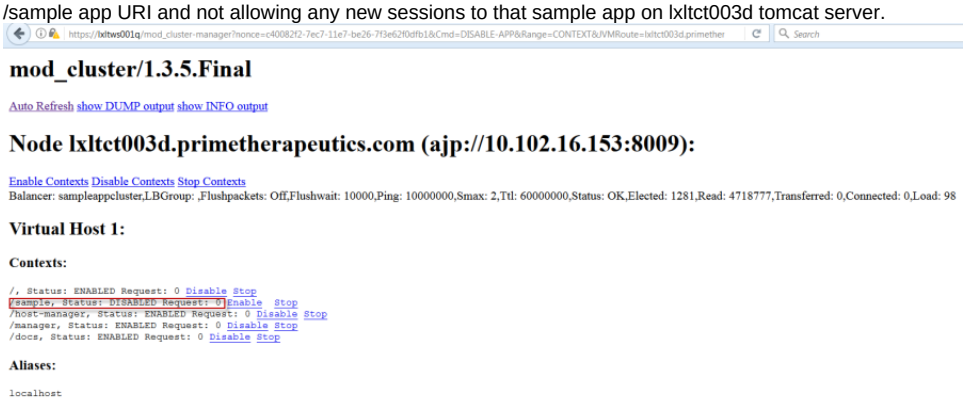
In the server.xml file in tomcat server side, we should mention stickysession true or not and list of apache webservers with mod_cluster port in the Proxylist. Please refer below as an example
<Listener className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener" advertise="true" stickySession="true" stickySessionForce="false" stickySessionRemove="true" proxyList="10.101.16.194:6666, "10.101.16.195:6666, 10.101.16.196:6666" />

Use below link to access the mod_cluster-manager page.
https://<Apache_webserver_name>/mod_cluster-manager

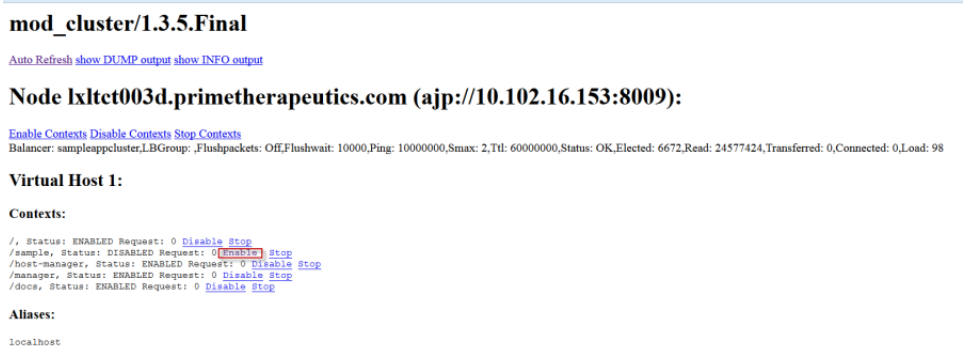
If you want to allow existing sessions continue to send request to same tomcat app server and not allowing any new sessions to that particular tomcat app server before deployment, we should do the below step
1.After accessing mod_cluster-manager page, go to particular tomcat app server and then click on "disable" entry against particular application context root. The below snapshot refers /sample context root.



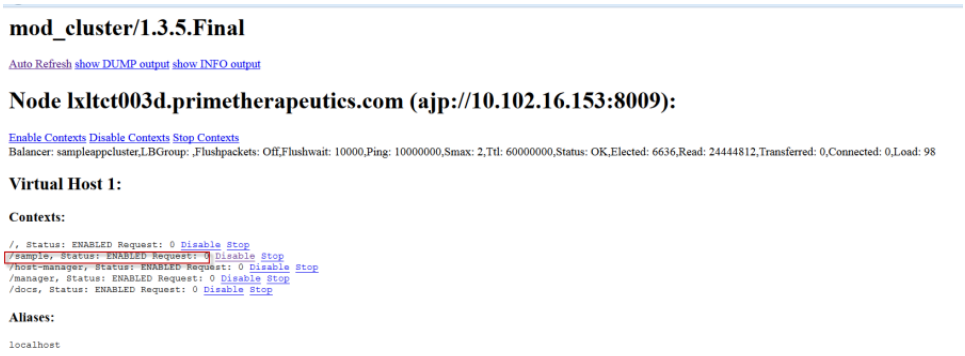
You see below after click on Disable entry against /sample context root on lxltct003d. You see status Disabled against /sample context root on lxltct003d right after click on Disable entry, it means apache would continue to allow existing session(s) to lxltct003d tomcat app server for /sample app URI and not allowing any new sessions to that sample app on lxltct003d tomcat server.



Repeat above mentioned step 1 on a particular data center tomcat servers by using each apache webserver mod_cluster-manger page and repeat step 1 again on rest of apache webservers before deployment.
Wait for 30 minutes to allow existing sessions to complete transactions and then do the deployment on a particular application on a particular data center.
After deployment and then stage testing, click on "Enable" entry against required app context on a tomcat server to make apache webserver allow traffic to tomcat app server. Repeat the same step to allow traffic from rest of apache webservers to required tomcat app servers.
Example: To allow traffic to lxltct003d tomcat server for /sample context root



After click Enable entry against /sample on lxltct003d, you see like below



Listed below curl command as an example to disable/enable application context root on a particular node before and after deployment.
curl "http://localhost:6666/mod_cluster-manager?Cmd=DISABLE-APP&Range=CONTEXT&JVMRoute=node1&Alias=default-host&Context=/myapp

Refer below link for more info related to mod_cluster
https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Web_Server/3/html/HTTP_Connectors_and_Load_Balancing_Guide/chap-mod_cluster_Connector.html

More Tomcat Security
Refer below link for more Tomcat security
https://www.mulesoft.com/tcat/tomcat-security