Install JBoss 5.1 on CentOS

davidghedini.blogspot.tw/2011/03/install-jboss-51-on-centos.html

This post will cover installing JBoss 5.1 on CentOS 5.x.

NOTE: If you wish to install JBoss 7.1 on CentOS, please see my post here:

http://www.davidghedini.com/pg/entry/install_jboss_7_on_centos

We'll also set up JBoss to run as a service

I did my installation below on CentOS 5.5. This should work for RHEL and Fedora as well.

Firstly, let's outline the steps we will be taking:

1. Download and Install the Java Development Kit (JDK)
2. Download and Install JBoss 5.1 Application Server
3. Create the user, jboss, who will own and run JBoss
4. Set the required JAVA_HOME and JBOSS_HOME paths
5. Create a start/stop/restart script for JBoss
6. Configure JBoss to run as a service
7. Change the JBoss Admin Console password
8. Set memory parameters for JBoss using JAVA_OPTS
9. Configure JBoss to run on port 80
10. Notes: Securing the JMX Console, Web Console, JBossWS, and Tomcat Status.

Step 1: Download and Install the Java Development Kit (JDK)

You can download the JDK here: http://www.oracle.com/technetwork/java/javase/downloads/index.html

I'm using JDK 6, update 24, the latest as of this post. The JDK is specific to 32 and 64 bit versions.

My CentOS box is 64 bit, so I'll need: jdk-6u24-linux-x64.bin.

If you are on 32 bit, you'll need: jdk-6u24-linux-i586.bin

Download the appropriate JDK and save it to a directory. I'm saving it to /root.

Move (mv) or copy (cp) the file to the /opt directory:

```
[root@sv2 ~]# mv jdk-6u24-linux-x64.bin /opt/jdk-6u24-linux-x64.bin
```

Create the directory /usr/java.

```
[root@sv2 ~]# mkdir /usr/java
```

Change to the /usr/java directory we created and install the JDK using 'sh /opt/jdk-6u24-linux-x64.bin'

```
[root@sv2 ~]# cd /usr/java
[root@sv2 java]# sh /opt/jdk-6u24-linux-x64.bin
```

We now have the JDK installed at /usr/java/jdk1.6.0_24. We'll use this for our JAVA_HOME a bit later in step

Step 2: Download and Install JBoss 5.1 Application Server

Download jboss-5.1.0.GA.zip at http://sourceforge.net/projects/jboss/files/JBoss/JBoss-5.1.0.GA/ or use wget:

```
[root@aoukuk25 ~]# wget http://sourceforge.net/projects/jboss/files/JBoss/JBoss-
5.1.0.GA/jboss-5.1.0.GA.zip/download
.
.
.
Saving to: `jboss-5.1.0.GA.zip'

100%[=======================================>] 133,466,607 5.58M/s   in 17s

2011-01-02 02:03:02 (7.56 MB/s) - `jboss-5.1.0.GA.zip' saved [133466607/133466607]
[root@sv2 ~]#
```

Move (mv) or copy (cp) the file to /usr/share/jboss-5.1.0.GA.zip.

```
[root@sv2 ~]# mv jboss-5.1.0.GA.zip /usr/share/jboss-5.1.0.GA.zip
```

Change to the /usr/share directory and unzip the file:

```
[root@sv2 ~]# cd /usr/share
[root@sv2 share]# unzip -q jboss-5.1.0.GA.zip
```

The unzip will create the following directory: /usr/share/jboss-5.1.0.GA

This directory will be our JBOSS_HOME, which we will use below in Step 4 below

Step 3: Create the user, jboss, who will own and run JBoss

Since we will want to run JBoss as a non-root user with minimal privileges, we'll create a user, jboss, who will own the JBoss files and JBoss will run under his account.

To do this, we can need to the following.

Create a new group, jboss, and then create the user jboss and add the user to the jboss group.

```
[root@sv2 ~]# groupadd jboss
[root@sv2 ~]# useradd -s /bin/bash -g jboss jboss
```

Change ownership of the JBoss home directory, /usr/share/jboss-5.1.0.GA so all files are owned by the user jboss we created.

```
[root@sv2 ~]# chown -Rf jboss.jboss /usr/share/jboss-5.1.0.GA/
```

Step 4: Set the required JAVA_HOME and JBOSS_HOME paths

We no need to set the JAVA_HOME and JBOSS_HOME.

The JAVA_HOME is where we installed the JDK above, /usr/java/jdk1.6.0_24, and the JBOSS_HOME is where we installed JBoss above /usr/share/jboss-5.1.0.GA.

Add the following to the jboss users .bash_profile:

```
JAVA_HOME=/usr/java/jdk1.6.0_24
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
JBOSS_HOME=/usr/share/jboss-5.1.0.GA
export JBOSS_HOME
```

To set the JAVA_HOME for users, we add this to the user ~/.bashrc or ~/.bash_profile of the user. We can also add it /etc/profile and then source it to give to all users.

```
JAVA_HOME=/usr/java/jdk1.6.0_24
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
```

Once you have added the above to ~/.bash_profile or ~/.bashrc, you should su to the user jboss and verify that the JAVA_HOME and JBOSS_HOME are set correctly.

```
[root@sv2 ~]#  su jboss
[jboss@sv2 ~]#  echo $JAVA_HOME
/usr/java/jdk1.6.0_24
[jboss@sv2 ~]#  echo $JBOSS_HOME
/usr/share/jboss-5.1.0.GA
```

Step 5: Create a start/stop/restart script for JBoss.

For our JBoss script we will simply copy the existing jboss_init_redhat.sh script located at at /usr/share/jboss-5.1.0.GA/bin, copy it to /etc/init.d and rename it to 'jboss':

So, as root:

```
[root@sv2 ~]# cd /usr/share/jboss-5.1.0.GA/bin
[root@sv2 bin]# cp jboss_init_redhat.sh /etc/init.d/jboss
```

In the jboss script (shown completed below), make the following changes:

1. Add lines 3,4, and 5:

# description: JBoss Start Stop Restart
# processname: jboss
# chkconfig: 234 20 80

2. Line 21, Set the JBOSS_HOME to where we unpacked JBoss in step 2 above:
JBOSS_HOME=${JBOSS_HOME:-"**/usr/share/jboss-5.1.0.GA**"}

3. Line 27. Set the JAVA_HOME to where we installed the JDK in step 1 above:
JAVAPTH=${JAVAPTH:-"**/usr/java/jdk1.6.0_24**"}

4. Add line 33, which sets the JBOSS_HOST to 0.0.0.0, allowing JBoss to bind to any IP.
JBOSS_HOST="0.0.0.0"

```
#!/bin/sh
#
# description: JBoss Start Stop Restart
# processname: jboss
# chkconfig: 234 20 80
# $Id: jboss_init_redhat.sh 81068 2008-11-14 15:14:35Z dimitris@jboss.org $
#
# JBoss Control Script
#
# To use this script run it as root - it will switch to the specified user
#
# Here is a little (and extremely primitive) startup/shutdown script
# for RedHat systems. It assumes that JBoss lives in /usr/local/jboss,
# it's run by user 'jboss' and JDK binaries are in /usr/local/jdk/bin.
# All this can be changed in the script itself.
#
# Either modify this script for your requirements or just ensure that
# the following variables are set correctly before calling the script.

#define where jboss is - this is the directory containing directories log, bin, conf
etc
JBOSS_HOME=${JBOSS_HOME:-"/usr/share/jboss-5.1.0.GA"}

#define the user under which jboss will run, or use 'RUNASIS' to run as the current
user
JBOSS_USER=${JBOSS_USER:-"jboss"}

#make sure java is in your path
JAVAPTH=${JAVAPTH:-"/usr/java/jdk1.6.0_24"}

#configuration to use, usually one of 'minimal', 'default', 'all'
JBOSS_CONF=${JBOSS_CONF:-"default"}

#if JBOSS_HOST specified, use -b to bind jboss services to that address
JBOSS_HOST="0.0.0.0"
JBOSS_BIND_ADDR=${JBOSS_HOST:+"-b $JBOSS_HOST"}


#define the classpath for the shutdown class
JBOSSCP=${JBOSSCP:-"$JBOSS_HOME/bin/shutdown.jar:$JBOSS_HOME/client/jnet.jar"}

#define the script to use to start jboss
JBOSSSH=${JBOSSSH:-"$JBOSS_HOME/bin/run.sh -c $JBOSS_CONF $JBOSS_BIND_ADDR"}
```

```
if [ "$JBOSS_USER" = "RUNASIS" ]; then
  SUBIT=""
else
  SUBIT="su - $JBOSS_USER -c "
fi

if [ -n "$JBOSS_CONSOLE" -a ! -d "$JBOSS_CONSOLE" ]; then
  # ensure the file exists
  touch $JBOSS_CONSOLE
  if [ ! -z "$SUBIT" ]; then
    chown $JBOSS_USER $JBOSS_CONSOLE
  fi
fi

if [ -n "$JBOSS_CONSOLE" -a ! -f "$JBOSS_CONSOLE" ]; then
  echo "WARNING: location for saving console log invalid: $JBOSS_CONSOLE"
  echo "WARNING: ignoring it and using /dev/null"
  JBOSS_CONSOLE="/dev/null"
fi

#define what will be done with the console log
JBOSS_CONSOLE=${JBOSS_CONSOLE:-"/dev/null"}

JBOSS_CMD_START="cd $JBOSS_HOME/bin; $JBOSSSH"
JBOSS_CMD_STOP=${JBOSS_CMD_STOP:-"java -classpath $JBOSSCP org.jboss.Shutdown --
shutdown"}

if [ -z "`echo $PATH | grep $JAVAPTH`" ]; then
  export PATH=$PATH:$JAVAPTH
fi

if [ ! -d "$JBOSS_HOME" ]; then
  echo JBOSS_HOME does not exist as a valid directory : $JBOSS_HOME
  exit 1
fi

echo JBOSS_CMD_START = $JBOSS_CMD_START

case "$1" in
start)
    cd $JBOSS_HOME/bin
    if [ -z "$SUBIT" ]; then
        eval $JBOSS_CMD_START >${JBOSS_CONSOLE} 2>&1 &
    else
        $SUBIT "$JBOSS_CMD_START >${JBOSS_CONSOLE} 2>&1 &"
    fi
    ;;
stop)
    if [ -z "$SUBIT" ]; then
        $JBOSS_CMD_STOP
    else
        $SUBIT "$JBOSS_CMD_STOP"
    fi
    ;;
restart)
    $0 stop
    $0 start
    ;;
*)
    echo "usage: $0 (start|stop|restart|help)"
esac
```

Step 6: Run JBoss as a Service.

To run JBoss as a service and enable start up at boot, make the script we created above executable and add it to our chkconfig so it starts at boot.

```
[root@sv2 init.d]# chmod 755 jboss
[root@sv2 init.d]# chkconfig --add jboss
[root@sv2 init.d]# chkconfig --level 234 jboss on
```

We should now be able to Start, Stop, and Restart JBoss as a service.

Start JBoss:

Note: JBoss can take some time to start.

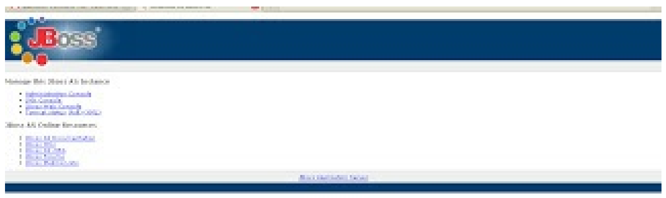
```
[root@sv2 init.d]# service jboss start
JBOSS_CMD_START = cd /usr/share/jboss-5.1.0.GA/bin; /usr/share/jboss-
5.1.0.GA/bin/run.sh -c default -b 0.0.0.0
```

Stop JBoss:

```
[root@sv2 init.d]# service jboss stop
JBOSS_CMD_START = cd /usr/share/jboss-5.1.0.GA/bin; /usr/share/jboss-
5.1.0.GA/bin/run.sh -c default -b 0.0.0.0
Shutdown message has been posted to the server.
Server shutdown may take a while - check logfiles for completion
```

Make sure JBoss is started and you should now be able to access the Jboss Console at:

http://yourdomain.com:8080 or http://yourip:8080



If you have any difficulties, check the logs and also insure that port 8080 is open

Step 7: Change the Admin Console Pasword.

The default user name and password for the JBoss Admin Console is admin/admin

To change the password, go to:

/usr/share/jboss-5.1.0.GA/server/default/conf/props

Edit the jmx-console-users.properties file, shown below

```
# A sample users.properties file for use with the UsersRolesLoginModule
admin=admin
```

The user name is at left and the password at right. Change the password to something hard to guess ;-)

Step 8: Set JAVA_OPTS Memory Parameters

To set the memory limits for JBoss,

Got to: /usr/share/jboss-5.1.0.GA/bin

Open the run.sh file in a text editor.

Find the section below:

```
# Setup JBoss specific properties
JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"
```

Directly below this, add the desired parameters.

```
# Setup JBoss specific properties
JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"
JAVA_OPTS="$JAVA_OPTS -Xms128m -Xmx256m"
```

I'm installing this on a small VPS so I'm using JAVA_OPTS="$JAVA_OPTS -Xms128m -Xmx256m". You should set this to whatever is appropriate to your server and application.

Step 9: Running JBoss on Port 80.

To run services below port 1024 as user other than root, you can use port forwarding.

You can do this by adding the following to your IP tables:

```
[root@sv2 ~]# iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 8080
[root@sv2 ~]# iptables -t nat -A PREROUTING -p udp -m udp --dport 80 -j REDIRECT --
to-ports 8080
```

Step 10: Notes: Secure the JBoss Web Console, JMX Console, JBossWS, and Tomcat Status Page .

This section will cover some simple and most basic methods of securing the consoles.

If you are simply running JBoss locally to have a look at it, you can skip this bit.

I've seen more elegent presentations of securing JBoss, so you may want to Google this if you find below a bit clunky.

As with anything related to your application and server security, please consult the docs.

Step 10a: Secure the JMX Console.

To secure the JMX Console, go to:
/usr/share/jboss-5.1.0.GA/server/default/deploy/jmx-console.war/WEB-INF

First, edit the web.xml file. Towards the bottom, you will find the security-constraint as shown below:

```
<!-- A security constraint that restricts access to the HTML JMX console
   to users with the role JBossAdmin. Edit the roles to what you want and
   uncomment the WEB-INF/jboss-web.xml/security-domain element to enable
   secured access to the HTML JMX console.
   <security-constraint>
     <web-resource-collection>
       <web-resource-name>HtmlAdaptor</web-resource-name>
       <description>An example security config that only allows users with the
         role JBossAdmin to access the HTML JMX console web application
       </description>
       <url-pattern>/*</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
     </web-resource-collection>
     <auth-constraint>
       <role-name>JBossAdmin</role-name>
     </auth-constraint>
   </security-constraint>
   -->
```

Un-comment the security-constraint section so it appears thus:

```
<security-constraint>
     <web-resource-collection>
       <web-resource-name>HtmlAdaptor</web-resource-name>
       <description>An example security config that only allows users with the
         role JBossAdmin to access the HTML JMX console web application
       </description>
       <url-pattern>/*</url-pattern>
```

```
        <http-method>GET</http-method>
        <http-method>POST</http-method>
      </web-resource-collection>
      <auth-constraint>
        <role-name>JBossAdmin</role-name>
      </auth-constraint>
   </security-constraint>
```

Next, still in the WEB-INF directory, edit the jboss-web.xml file, which will look as below:

```
<!DOCTYPE jboss-web PUBLIC
   "-//JBoss//DTD Web Application 5.0//EN"
   "http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd">

<jboss-web>
   <!-- Uncomment the security-domain to enable security. You will
      need to edit the htmladaptor login configuration to setup the
      login modules used to authentication users.
      <security-domain>java:/jaas/jmx-console</security-domain>
   -->
</jboss-web>
```

Uncomment the security-domain so it appears thus:

```
<jboss-web>

      <security-domain>java:/jaas/jmx-console</security-domain>

</jboss-web>
```

At this point, the password for the JMX Console will be the same as the password we set for the Admin Console at in in step 7a above. Both are using the java:/jaas/jmx-console security domain.

You can, of course change this if you wish to create a stronger solution.

Step 10b: Secure the Web Console.

To secure the Web Console, go to:

/usr/share/jboss-5.1.0.GA/server/default/deploy/management/console-mgr.sar/web-console.war/WEB-INF

As with the JMX Console, in the WEB-INF.xml un-comment the security constraint so it appears thus:

```
<security-constraint>
   <web-resource-collection>
   <web-resource-name>HtmlAdaptor</web-resource-name>
   <description>An example security config that only allows users with the
   role JBossAdmin to access the HTML JMX console web application
   </description>
   <url-pattern>/*</url-pattern>
   <http-method>GET</http-method>
   <http-method>POST</http-method>
   </web-resource-collection>
   <auth-constraint>
   <role-name>JBossAdmin</role-name>
   </auth-constraint>
   </security-constraint>
```

Still in the WEB-INF directory, go to the jboss-web.xml file.

By default, The jboss-web.xml file will appear as below:

```
<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE jboss-web
    PUBLIC "-//JBoss//DTD Web Application 2.3V2//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-web_3_2.dtd">

<jboss-web>

   <!-- Uncomment the security-domain to enable security. You will
   need to edit the htmladaptor login configuration to setup the
   login modules used to authentication users.
   <security-domain>java:/jaas/web-console</security-domain>
   -->

   <!-- The war depends on the -->
   <depends>jboss.admin:service=PluginManager</depends>
</jboss-web>
```

Un-comment the security-domain so it appears thus:

```
<jboss-web>

   <security-domain>java:/jaas/web-console</security-domain>




   <depends>jboss.admin:service=PluginManager</depends>
</jboss-web>
```

Now, we need to make a change to the go to login-config.xml file located under /usr/share/jboss-5.1.0.GA/server/default/conf/

Open the login-config.xml and look for the section below:

```
<application-policy name="web-console">
    <authentication>
      <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
flag="required">
```

```
        <module-option name="usersProperties">web-console-users.properties</module-
option>
        <module-option name="rolesProperties">web-console-roles.properties</module-
option>
      </login-module>
    </authentication>
  </application-policy>
```

Add 'props/' to the path of the web-console-users.properties and web-console-roles.properties

So the section will now appear thus:

```
<application-policy name="web-console">
    <authentication>
      <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
flag="required">
        <module-option name="usersProperties">props/web-console-
users.properties</module-option>
        <module-option name="rolesProperties">props/web-console-
roles.properties</module-option>
      </login-module>
    </authentication>
  </application-policy>
```

Finally, go to /root/jboss-5.1.0.GA/server/default/conf/props and create the following files:

1. web-console-roles.properties

The web-console-roles.properties file should contain the following:

```
admin=JBossAdmin,HttpInvoker
```

2. web-console-users.properties

The web-console-users.properties file should contain the following:

```
admin=WebSecret
```

Where 'WebSecret' is whatever you would like the password to be. If you wish to be able to access the Web Console with the same password as for the Admin and JMX console, simply use the same password here.

Step 10c: Secure the JBossWS.

The procedure for securing the JBossWS is virtually identical to securing the JMX-Console, save the difference in file loactions.

To secure the JBossWS, go to:
/usr/share/jboss-5.1.0.GA/server/default/deploy/jbossws.sar/jbossws-management.war/WEB-INF

First, edit the web.xml file. Locate the security-constraint (about half-way down the file), and un-comment it so it appears thus:

```
<security-constraint>
     <web-resource-collection>
       <web-resource-name>ContextServlet</web-resource-name>
       <description>An example security config that only allows users with the
         role 'friend' to access the JBossWS console web application
       </description>
       <url-pattern>/*</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
     </web-resource-collection>
     <auth-constraint>
       <role-name>friend</role-name>
     </auth-constraint>
   </security-constraint>
```

Next, still in the WEB-INF directory, edit the jboss-web.xml file.

Un-comment the security-domain so it appears thus:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE jboss-web
    PUBLIC "-//JBoss//DTD Web Application 2.3V2//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-web_3_2.dtd">

<jboss-web>


  <security-domain>java:/jaas/JBossWS</security-domain>


  <context-root>jbossws</context-root>

</jboss-web>
```

In the /props directory you will find the jbossws-roles.properties and jbossws-users.properties files.

The default role is 'friend' with user name 'Kermit' and password 'the frog'

jbossws-roles.properties:

```
# A sample roles.properties file for use with the UsersRolesLoginModule
kermit=friend
```

jbossws-users.properties:

```
# A sample users.properties file for use with the UsersRolesLoginModule
```

```
kermit=thefrog
```

Change the user name and password.

Step 10d: Secure the Tomcat Status Page.

Many would recommend simply disabling the Tomcat Status.

If you wish to secure it, however, go to:

/usr/share/jboss-5.1.0.GA/server/default/deploy/ROOT.war/WEB-INF

Just before the closing web-app tag add the following to the end of the web.xml file:

```
<security-constraint>
    <security-constraint>
     <web-resource-collection>
       <web-resource-name>HtmlAdaptor</web-resource-name>
       <description>An example security config that only allows users with the
         role JBossAdmin to access the HTML JMX console web application
       </description>
       <url-pattern>/status</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
     </web-resource-collection>
     <auth-constraint>
       <role-name>TomcatStatus</role-name>
     </auth-constraint>
   </security-constraint>


   <login-config>
      <auth-method>BASIC</auth-method>
      <realm-name>TomcatStatus</realm-name>
   </login-config>

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

Your web.xml file should now look thus:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>Welcome to JBoss</display-name>
  <description>
```

```
     Welcome to JBoss
  </description>
  <servlet>
    <servlet-name>Status Servlet</servlet-name>
    <servlet-class>org.jboss.web.tomcat.service.StatusServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Status Servlet</servlet-name>
    <url-pattern>/status</url-pattern>
  </servlet-mapping>



<security-constraint>
     <web-resource-collection>
       <web-resource-name>HtmlAdaptor</web-resource-name>
       <description>An example security config that only allows users with the
         role JBossAdmin to access the HTML JMX console web application
       </description>
       <url-pattern>/status</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
     </web-resource-collection>
     <auth-constraint>
       <role-name>TomcatStatus</role-name>
     </auth-constraint>
   </security-constraint>


   <login-config>
      <auth-method>BASIC</auth-method>
      <realm-name>TomcatStatus</realm-name>
   </login-config>

   <security-role>
      <role-name>TomcatStatus</role-name>
   </security-role>




</web-app>
```

Still in the /usr/share/jboss-5.1.0.GA/server/default/deploy/ROOT.war/ directory, create a jboss-web.xml file with the following contents:

```
<jboss-web>

       <security-domain>java:/jaas/tomcat-status</security-domain>

</jboss-web>
```

Go to /usr/share/jboss-5.1.0.GA/server/default/conf

Look for the following section:

```
<application-policy name="web-console">
    <authentication>
      <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
        flag="required">
        <module-option name="usersProperties">props/web-console-
users.properties</module-option>
        <module-option name="rolesProperties">props/web-console-
roles.properties</module-option>
      </login-module>
    </authentication>
  </application-policy>
```

Directly under this section, add the following entry:

```
<application-policy name="tomcat-status">
    <authentication>
      <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
        flag="required">
        <module-option name="usersProperties">props/tomcat-status-
users.properties</module-option>
        <module-option name="rolesProperties">props/tomcat-status-
roles.properties</module-option>
      </login-module>
    </authentication>
  </application-policy>
```

Now, in the /usr/share/jboss-5.1.0.GA/server/default/conf/props directory, create the following two files:

1. tomcat-status-roles.properties

The tomcat-status-roles.properties file should contain the following:

```
admin=TomcatStatus
```

2. tomcat-status-users.properties

The tomcat-status-users.properties file should contain the following:

```
admin=TomcatSecret
```

Where 'TomcatSecret' is whatever you would like the password to be. If you wish to be able to access the Web Console with the same password as for the Admin and JMX console, simply use the same password here.

Again, you may find it simpler to just disable the Tomcat Status.

http://community.jboss.org/