INSTALACIÓN, CONFIGURACIÓN Y ADMINISTRACIÓN DEL SERVIDOR DE APLICACIONES

José A Alférez Sánchez
www.alferez.es

INDICE

1. ¿Qué es JBOSS?

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc., adquirida por Red Hat en Abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El proyecto está apoyado por una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios.

JBoss implementa todo el paquete de servicios de J2EE.

Por ejemplo, Los Sims online, utilizan JBoss para sus juegos multiusuario.

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustrado, y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estandares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX

2. Requisitos

Nuestro proyecto va a ser instalado bajo un sistema operativo Ubuntu 8.10 recién instalado, por lo que vamos a ver paso a paso los paquetes necesarios desde cero hasta tener nuestro servidor completamente configurado y listo para trabajar con él.

Como paquete indispensable es JAVA y el propio JBOSS pero además nosotros vamos a integrarlo con Apache2, por lo que también instalaremos este servidor.

3. Instalación

3.1 Servidor Apache

Para la instalación del servidor web vamos realizar la instalación se los servicios LAMP (Linux + Apache + MySql + PHP). Con esto nos aseguramos de que además de tener realizada la instalación de Apache, el servicio de MySql y PHP enlazarán correctamente con él.

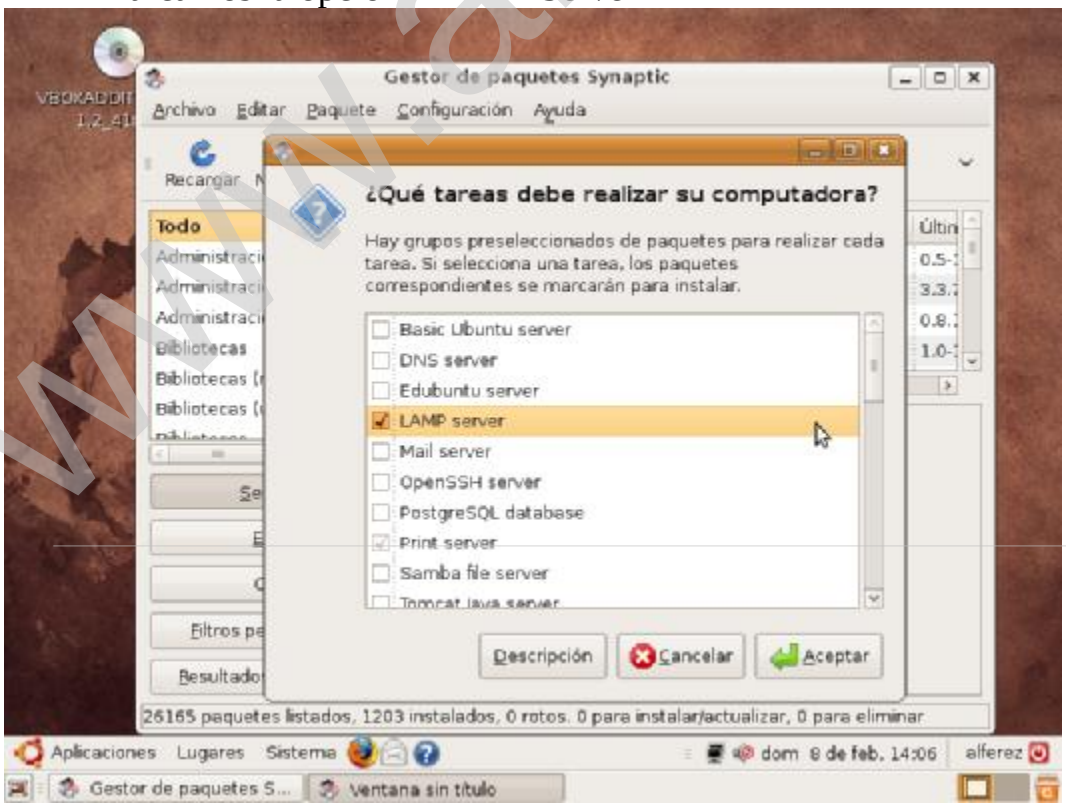
Estos servicios se pueden instalar uno a uno, pero por comodidad vamos a realizar la instalación de todo el paquete.

Paso 1:

Nos vamos al menú Sistemas -> Administración -> Gestor de Paquetes Synaptic

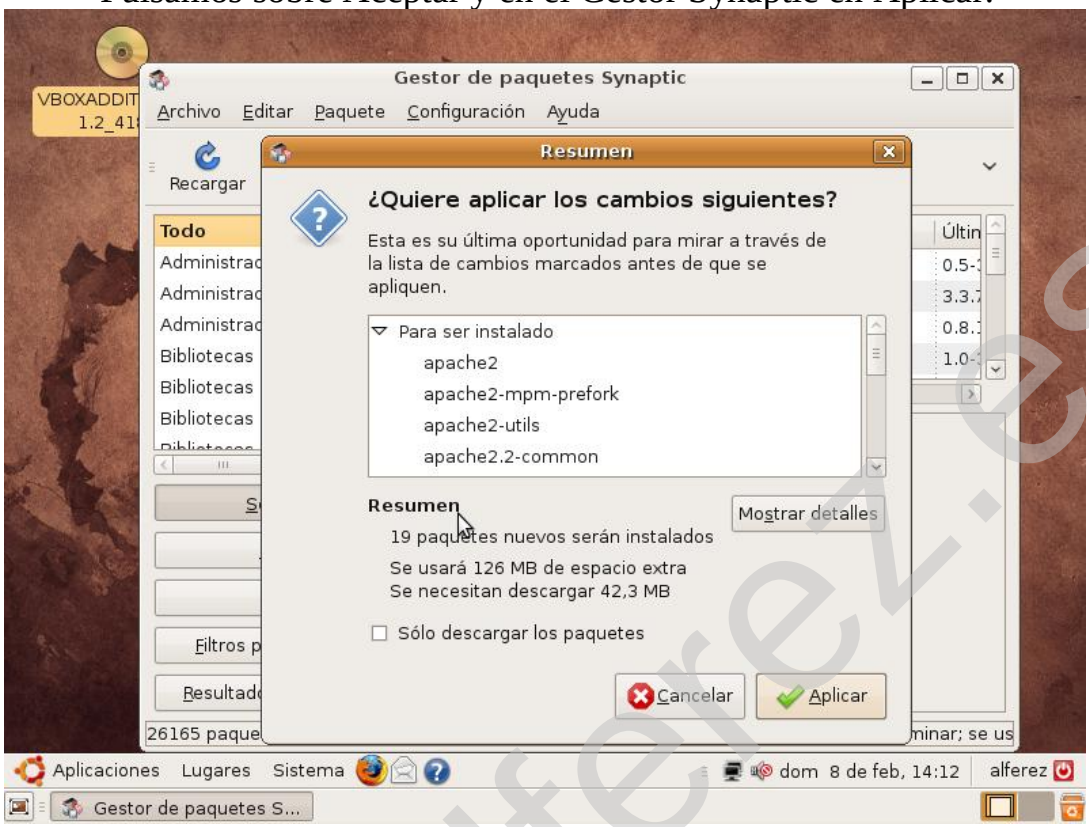
Paso 2:

Entramos en el menú Editar -> Marcar Paquetes por Tarea y marcamos la opción "LAMP Server"



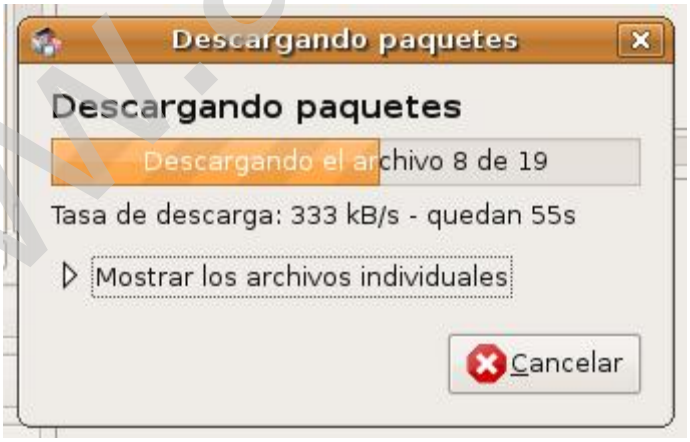
Paso 3:

Pulsamos sobre Aceptar y en el Gestor Synaptic en Aplicar.



Paso 4:

Esperamos a que termine de descargar e instalar:



Paso 5:

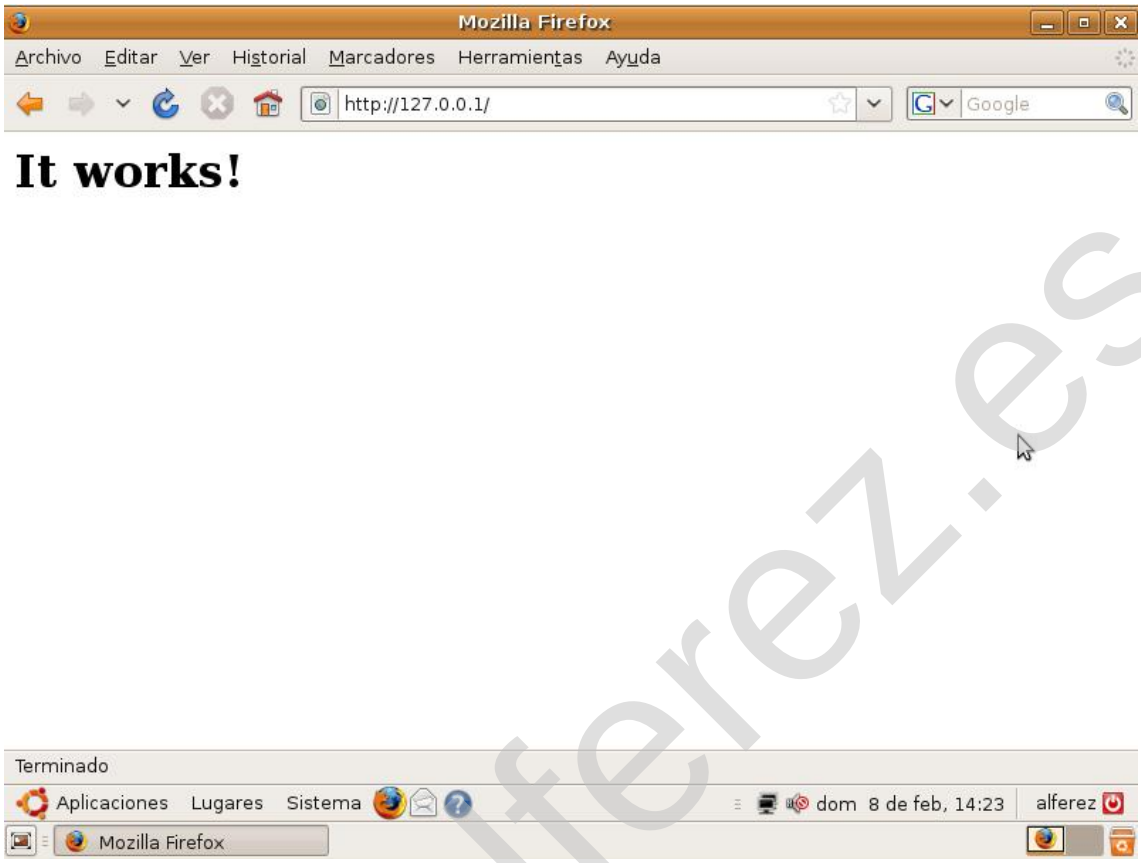
Nos pedirá el password para el Administrador de MySql (No tiene nada que ver con el Root del sistema)

Paso 6:

Esperamos a que aplique todos los cambios y ya lo tenemos instalado.

Paso 7:

Tan sólo nos queda comprobar que la instalación sea correcta:



Adicionalmente vamos a instalar PHPMyAdmin para administrar las BD de MySql y poder realizar pruebas después.

Para instalarlo escribiremos desde la línea de comandos:

```
sudo apt-get install phpmyadmin
```

Tras realizar la instalación nos preguntará que versión de Apache tenemos instalada. En nuestro caso Apache2.

Para administrar en un futuro el MySql lo aremos a través de web, por la dirección:

http://NOMBRESERVIDOR/phpmyadmin

3.2 Java

La instalación de los paquetes java la realizaremos desde consola con un solo paso. Para ello ejecutaremos el siguiente comando:

```
sudo apt-get install sun-java6-jre sun-java6-jdk sun-java-bin
```

Tras realizar la instalación nos aparecerá el acuerdo de licencia y nos preguntará si estamos de acuerdo.

Una vez finalizada comprobaremos si la versión ha sido correctamente instalada con el siguiente comando:

```
java –version
```

Y nos debe indicar algo como esto:

```
java version "1.6.0_10"
Java(TM) SE Runtime Environment (build 1.6.0_10-b33)
Java HotSpot(TM) Client VM (build 11.0-b15, mixed mode, sharing)
```

Con esto ya tenemos Java instalado, pero aun nos queda por introducir un parámetro para que el JBOSS no nos de problema.

Este parámetro es una variable JAVA_HOME que se usa para indicar donde están instalado el paquete JAVA. Para ello usaremos estos comandos:

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun
export PATH=$PATH:$JAVA_HOME/bin
```

Tendremos que reiniciar para que recargue los parámetros. Para comprobar que todo está correcto comprobamos que el parámetro ha sido configurado con éxito tecleando lo siguiente:

```
set | grep JAVA
```

Y nos debe indicar el valor de JAVA_HOME.

3.3 JBOSS

Para comenzar la instalación de JBOSS lo primero es descargarlo desde la web oficial (www.jboss.org). En nuestro caso vamos a usar la versión 4.2.3.GA porque aunque la versión 5.0.0 ya está en la calle aun es demasiado reciente y es mejor trabajar con una ya rodada.

En nuestro caso la hemos descargado en el escritorio y debemos descomprimirlo en el mismo lugar. Ahora volvemos a modo consola para seguir con los pasos. Ejecutamos lo siguiente:

```
sudo mv /home/USUARIO/Escritorio/jboss-4.2.3.GA /opt/jboss
```

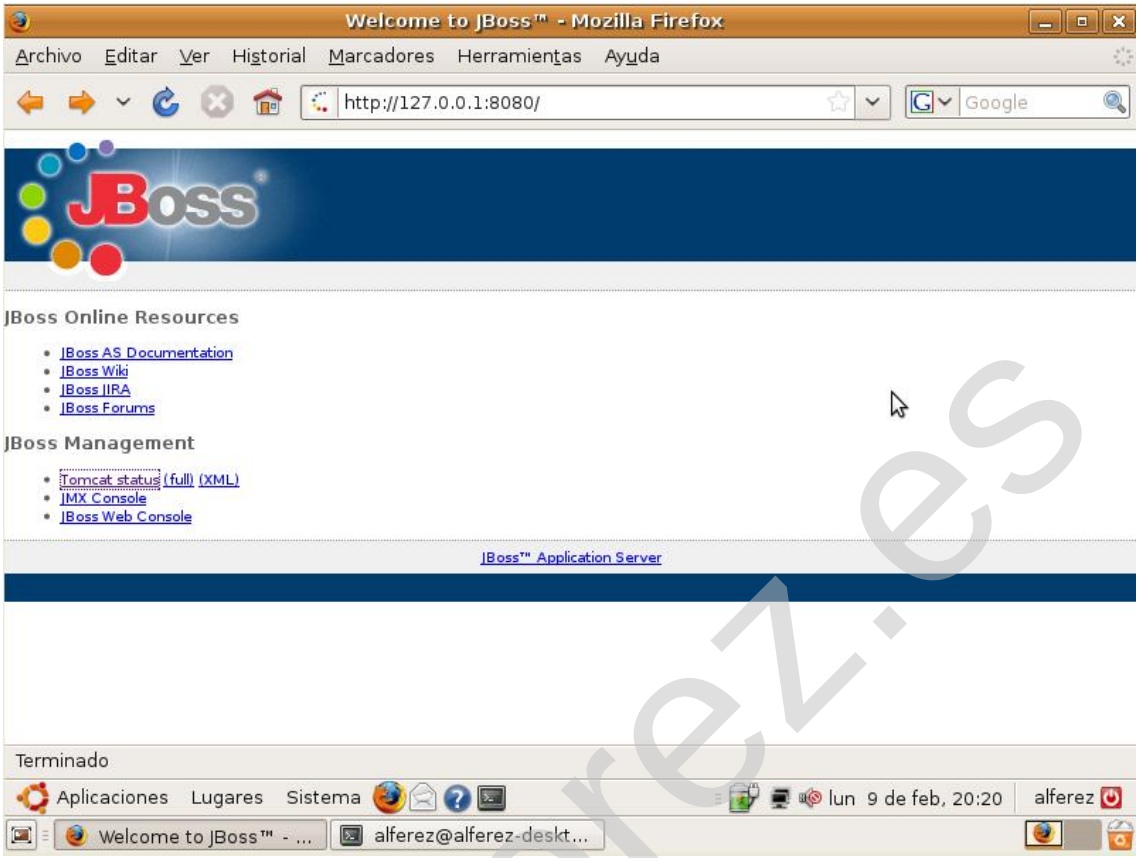
Con esto creamos la carpeta en /opt . Al igual que hicieramosn antes con Java ahora nos toca definir variables para JBOSS, para ellos ejecutamos esto:

```
export JBOSS_HOME=/opt/jboss
export PATH=$PATH:$JBOSS_HOME/bin
```

Con todo esto ya tenemos instalado nuestro servidor JBOSS, para probarlo ejecutaremos lo siguiente:

```
sudo /opt/jboss/bin/run.sh
```

Con el servidor funcionando ya podemos probar que se ejecuta correctamente entrando desde el navegador en la dirección http://127.0.0.1:8080. Debe salirnos esto:



Ahora necesitamos realizar el servicio de inicio y parada el servidor, para ello ejecutamos lo siguiente:

```
sudo nano /etc/init.d/jboss
```

Con esto entramos en el editor de texto y copiamos el siguiente texto:

```
#! /bin/sh
start(){
   echo "Starting jboss.."
   sudo -u root /opt/jboss/bin/run.sh > /dev/null 2> /dev/null &
}
stop(){
        echo "Stopping jboss.."
        sudo -u root /opt/jboss/bin/shutdown.sh -S &
}
restart(){
        stop
        sleep 60
        sudo -u root killall java
        start
}
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
```

```
        restart
        ;;
  *)
        echo "Usage: jboss {start|stop|restart}"
        exit 1
esac

exit 0
```

Nos queda darle privilegios a ese usuario para que se pueda ejecutar:

```
sudo chmod 755 /etc/init.d/jboss
sudo chmod +x /etc/init.d/jboss
```

Ya tan sólo nos queda actualizar los runlevels para que nuestro servicio esté operativo y se ejecute al arrancar. Para ello ejecutamos lo siguiente:

```
sudo update-rc.d jboss defaults
```

Y probamos el script del servicio con:

Para Arrancar: `/etc/init.d/jboss start`
Para Parar: `/etc/init.d/jboss stop`
Para Reiniciar: `/etc/init.d/jboss restart`

Podemos hacerlo también con estos scripts:

Para Arrancar: `sudo /opt/jboss/bin/run.sh`
Para Parar: `sudo /opt/jboss/bin/shutdown.sh`

4. Configuración

4.1 Estructura de directorio

El directorio raíz de JBoss (en nuestro caso /opt/jboss) está dividido en varios directorios que contienen desde los ejecutables hasta la configuración.

Aquí tenéis una pequeña descripción de los directorios:

bin

Este directorio contiene los ejecutables utilizados por JBoss, el más importante siendo el "script" de arranque utilizado por éste (run.sh) .

client

Contiene los diversos archivos JAR's que serán utilizados por los distintos clientes de los EJB's utilizados en JBoss. Dichos archivos deben ser agregados a la variable CLASSPATH del sistema donde radica el cliente; el cliente generalmente siendo un JSP/Servlet que acceda el EJB, este paradigma gira alrededor de Stubs/Skeletons de RMI una parte central de EJB's.(Vea la gráfica básica de JBoss donde se ilustra este concepto)

docs

Este directorio contiene documentación acerca de JBoss y ejemplos de conexión con distintos tipos de servidores de bases de datos.

lib

Este directorio contiene los archivos JAR's empleados por JBoss requeridos en cualquier modalidad.

server

Este directorio contiene tres sub-directorios nombrados: all, default y minimal; cada sub-directorio contiene los distintos archivos de configuración necesarios para ejecutar JBoss en diferentes modalidades.

La modalidad all incluye la ejecución de JBoss para emplearse como "Cluster", ejecución de "Web-Services" y otras funcionalidades más ; el directorio default como su nombre lo implica, incluye la configuración para ejecutar JBoss de manera básica, mientras el directorio minimal contiene los valores de configuración necesarios para ejecutar JBoss con requerimientos mínimos; el "Script" de arranque proporcionado con JBoss emplea los valores del directorio default, para emplear otra modalidad es necesario modificar dicho "Script" de arranque (run.sh).

A continuación se describen los directorios residentes en la modalidad de arranque default :

conf

Este directorio contiene las diferentes secciones de configuración utilizadas por JBoss, dependiendo de la modalidad utilizada este directorio puede contener distintos archivos, sin embargo, sus detalles serán descritos en configuración de JBoss .
A continuación se detallan los ficheros y su descripción:

jboss-minimal.xml

Archivo que contiene los parámetros principales para la configuración "Default" de JBOSS; este archivo XML define los valores para la variable CLASSPATH, el puerto para el servidor JNDI y el directorio donde serán colocados los distintos EJB's para ser ejecutados, entre otros parámetros.

jbossmq-state.xml

Contiene los usuarios y roles disponibles para emplear el sistema "Messaging" proporcionado con JBOSS.

jboss-service.xml

Archivo que contiene los parámetros principales del Servidor JBOSS; este archivo XML define los valores para la variable CLASSPATH, el puerto para el servidor JNDI y el directorio donde serán colocados los distintos EJB's para ser ejecutados, entre otros parámetros.

jndi.properties

Contiene las Poperties que serán utilizadas ("Factory's") para realizar búsquedas JNDI.

login-config.xml

Contiene los parámetros JAAS empleados por JBOSS para verificar/autentificar usuarios.

server.policy

Parámetros de seguridad empleados por JBOSS.

standardjaws.xml

JAWS es el motor de mapeo Objeto/Relacional empleado por JBOSS en CMP ("Container Managed Persistence") EJB's, este archivo contiene sus valores "Default".

standardjboss.xml

Contiene los parámetros estándar de configuración para JBOSS tales como: Tamaño de "Pools" para EJB's, valores de "Cache", número de "Pools" para Bases de Datos, Clases empleadas para Control de Transacciones, entre otros parámetros.

data

Contiene distintos parámetros y archivos de configuración para las Bases de Datos proporcionadas con JBoss (Hypersonic y la implementación "Messaging" de JBoss) -- generalmente utilizada para aplicaciones demo.

deploy

Este directorio es ampliamente utilizado ya que aquí se colocan los EJB's para que sean ejecutados por JBoss, una vez colocado el archivo JAR (en forma de EJB) en este directorio, JBoss automáticamente expande y ejecuta el EJB.

lib

Contiene los archivos JAR's empleados por JBoss en base a la modalidad tratada.

log

Contiene los distintos registros ("Logs") generados por JBoss.

tmp

Contiene archivos creados por JBoss y utilizados de manera temporal.

work

Contiene las clases y archivos utilizados por JBoss para ejecución.

4.2 Configuración de Puertos

En ocasiones nos interesa cambiar los puertos por los que contesta el servidor, ya sea por incompatibilidad con otros servicios instalados o por querer instalar varias instancias como veremos posteriormente.

En nuestro ejemplo vamos a trabajar con una instancia llamada instancia2 en vez de con la típica default y vamos a cambiar los puertos añadiéndole siempre un 1 delante.

Para configurar los puertos necesitamos editar varios ficheros. Veremos uno por uno cada fichero y los cambios que debemos realizar:

/opt/jboss/server/instancia2/deploy/jboss-web.deployer/server.xml

```
<Connector port="8080" address="${jboss.bind.address}"
    maxThreads="250" maxHttpHeaderSize="8192"
    emptySessionPath="true" protocol="HTTP/1.1"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true" />
```

Cambiamos el Connector port y vamos a ponerle 18080

```
<Connector port="8009" address="${jboss.bind.address}"
protocol="AJP/1.3" emptySessionPath="true" enableLookups="false"
redirectPort="8443" />
```

Cambiamos el puerto 18009 por 18009

/opt/jboss/server/instancia2/conf/jboss-service.xml

```
<attribute name="Port">1099</attribute>
```

Cambiamos el puerto 1099 por 11099

```
<attribute name="RmiPort">1098</attribute>
```

Cambiamos el puerto 1098 por 11098

```
<attribute name="ServerBindPort">4445</attribute>
```

Cambiamos el puerto 4445 por 14445

```
<attribute name="RMIObjectPort">4444</attribute>
```

Cambiamos el puerto 4444 por 14444

```
<attribute name="Port">8083</attribute>
```

Cambiamos el puerto 8083 por 18083

/opt/jboss/server/instancia2/conf/jboss-minimal.xml

```
<attribute name="Port">1099</attribute>
```

Cambiamos el puerto 1099 por 11099

```
<attribute name="RmiPort">1098</attribute>
```

Cambiamos el puerto 1098 por 11098

/opt/jboss/server/instancia2/deploy/jms/uil2-service.xml

```
<attribute name="ServerBindPort">8093</attribute>
```

Cambiamos el puerto 8093 por 18093

4.3 Múltiples Instancias

JBoss como otros servidores nos permite tener activas varias instancias. No es muy lógico hacerlo puesto que es volver a crear un segundo servidor pero si posible.

Para crear una segunda instancia nos bastaría con copiar la *default*, pero para activarlo deberíamos realizar varios cambios en la configuración.

Empezaremos con la copia del directorio:

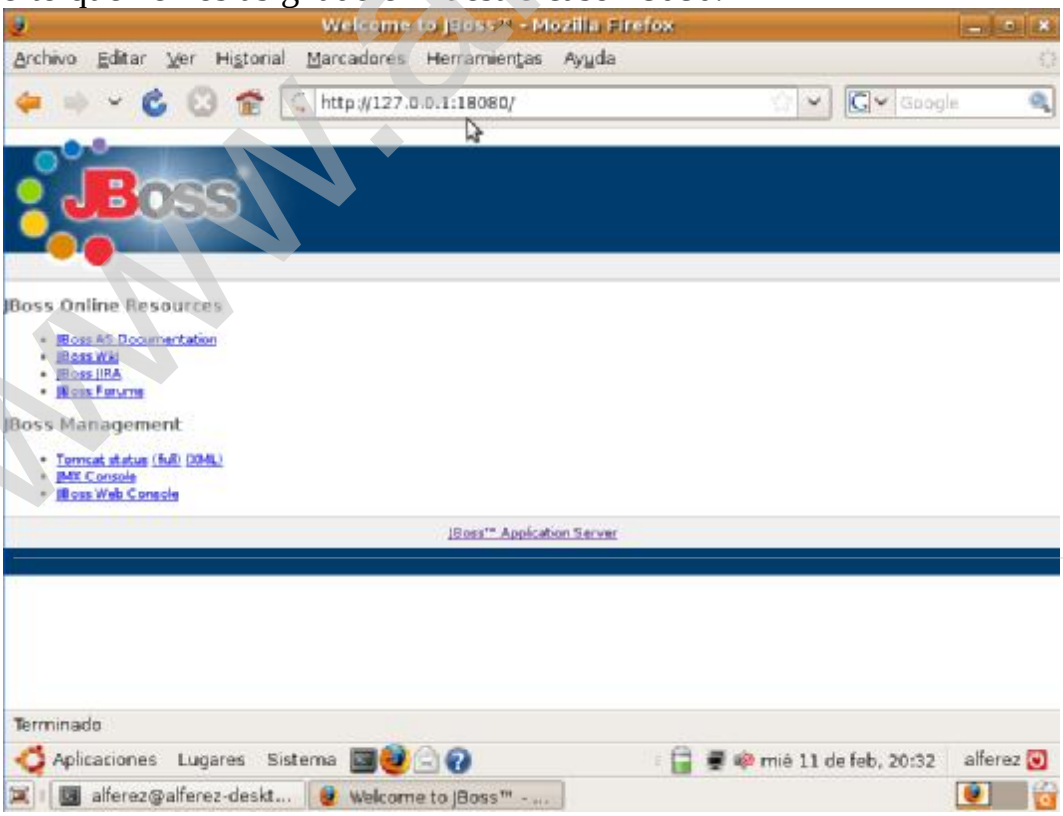
```
mkdir /opt/jboss/server/instancia2
cp /opt/jboss/server/default/* /opt/jboss//server/instancia2/ -r
```

Una vez tengamos listo el directorio de la nueva instancia nos queda cambiar los puertos antes de activarlos. Para ello seguiremos el ejemplo que hemos visto anteriormente

Con los puertos ya cambiados para la nueva instancia tan sólo nos queda levantar el servicio. Para ello ejecutaremos:

```
sudo /opt/jboss/bin/run.sh –c instancia2
```

Ejecutamos en el navegador la dirección del servidor seguido del puerto que hemos asignado en nuestro caso 18080.



Si deseamos podemos crear un nuevo fichero usando de base /etc/init.d/jboss pero cambiando el apartado donde se llama a /opt/jboss/bin/run.sh y añadimos el –c nombredeinstancia para poder arrancar o parar el servicio a nuestro antojo.

4.4 Integrar con Apache

Aunque nuestro servidor JBOSS es autosuficiente para funcionar sin necesidad de ningún otro servidor web, es posible que deseemos implementarlo sobre un servidor Apache para descargar a JBOSS de las tareas que no son realmente suyas.

Teniendo en cuenta que nuestro servidor Apache ya lo tenemos instalado, ahora tan sólo nos queda instalar el conector proxy para que derive a JBOSS lo que deseemos .

En nuestro caso vamos a configurar apache para que responda a las peticiones que se le hagan configurando la aplicación que tiene por defecto JBoss de Web-Console. A esta aplicación debíamos acceder por la ruta http://localhost:8080/web-console.

Lo primero es activar el conector mediante consola:

```
sudo /etc/apache2/mods-available/a2enmod proxy_ajp
```

Con el conector ya instalado tenemos que realizar la configuración para que Apache acepte la conexión, para ello editamos el fichero /etc/apache2/mods-available/proxy.conf.

En caso de que atacásemos a este servidor desde Internet y con un dominio en concreto modificaríamos la línea “*Deny from All”* por “*Allow from .dominio.com”* pero en nuestro caso le vamos a dar permiso para que permita el acceso a todos con *“Allow from All”*.

Una vez tenemos el modulo activo y configurado para que nos permita el acceso tan sólo nos queda indicarle a apache cual es la ruta que vamos a redirigir a JBoss.

Para ello editamos el fichero /etc/apache2/sites-enabled/000-default y añadimos las siguiente líneas entre las etiquetas *</Directory>* y *</VirtualHost>* :

```
      ProxyPass /web-console ajp://localhost:8009/web-console
ProxyPassReverse /web-console [***]ajp://localhost:8009/web-console
      ProxyPass /jmx-console ajp://localhost:8009/jmx-console
ProxyPassReverse /jmx-console [***]ajp://localhost:8009/jmx-console
           ProxyPass /status ajp://localhost:8009/status
     ProxyPassReverse /status [***]ajp://localhost:8009/status
```

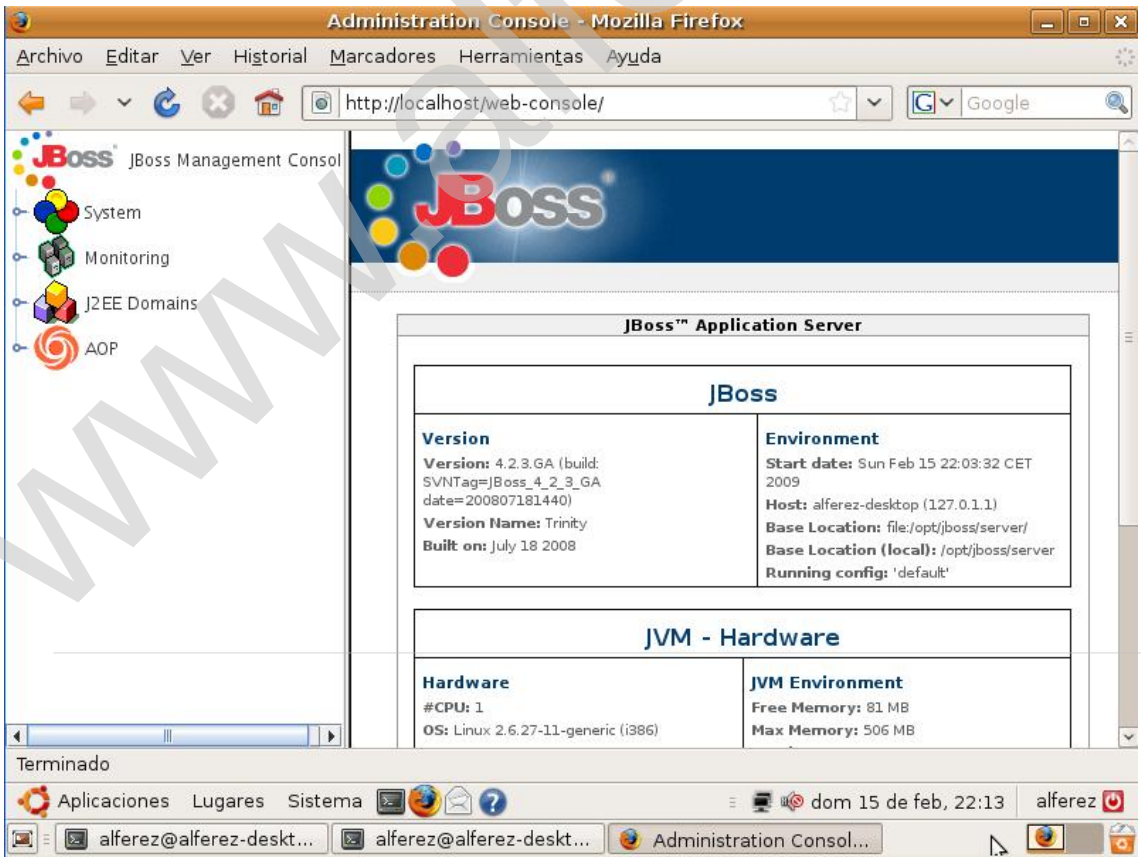
El Puerto 8009 es el que tiene configurado por defecto JBoss para interactuar con el conector. En caso de que en la instancia lo hubiésemos cambiado (como hicimos con la instancia2 y le pusimos el 18009) deberíamos cambiarlo.

Ahora tan sólo nos queda reiniciar apache:

```
sudo /etc/init.d/apache2 restart
```

y probar que podemos acceder tanto a:

http://localhost:8080/web-console (Acceso a JBoss)
http://localhost/web-console (Acceso por Apache)



5. Conexión con Bases de Datos

JBoss tiene configurada una conexión de base de datos por defecto, esta conexión es HSQLDB (Hypersonic).

A demás de estas es posible conectar con mucho de los motores de bases de datos existentes en el mercado como MySQL o Oracle.

5.1 MySQL

Lo primero que debemos obtener es el conector, este nos lo podemos descargar desde la propia web de MySQL (http://dev.mysql.com/downloads/connector/j/5.1.html). En nuestro caso vamos a usar la versión 5.1.7 que es la más reciente.

Una vez obtenido extraemos el fichero .jar (en nuestro caso mysql-connector-java-5.1.7-bin.jar) dentro de la carpeta /opt/jboss/server/default/lib (en caso de usar otra instancia, lo haríamos en la carpeta lib de esa instancia).

Ahora copiaremos el fichero que se encuentra en /opt/jboss/docs/examples/jca/mysql-ds.xml en /opt/jboss/server/default/deploy . Lo editamos y modificacmos lo siguiente:

```
                                <connection-url>jdbc:mysql://mysql-
hostname:3306/jbossdb</connection-url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <user-name>x</user-name>
  <password>y</password>
```

Por:

```
  <connection-url>jdbc:mysql://localhost:3306/NOMBREBD</connection-
url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <user-name>USUARIOMYSQL</user-name>
  <password>PASSWORDMYSQL</password>
```

Tan solo nos queda reiniciar los servicios de JBOSS y atacar a la conexión con un ejemplo como el siguiente:

```
<%@ page import="java.sql.*" %>
<%
  String connectionURL = "jdbc:mysql://localhost/test";
  Connection connection = null;
  Statement statement = null;
  ResultSet rs = null;
%>
<html>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      connection = DriverManager.getConnection(connectionURL,
"root", "mat3");
      statement = connection.createStatement();
      rs = statement.executeQuery("SELECT * FROM people");
      while (rs.next()) {
         out.println(rs.getString("name")+"
");
      }
      rs.close();
   %>
  </body>
</html>
```

5.2 Oracle

Lo primero que necesitamos es obtener el jar que contiene los drivers JDBC de Oracle, ocuparemos ojdbc14.jar, el cual está optimizado para su uso con las últimas versiones del JDK: 1.4 - 1.5, mientras otros drivers como el clásico classes12.jar está optimizado para JDK 1.2 y 1.3. Lo bajaremos de aquí: http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdb...

Jboss tiene plantillas de configuraciones para diferentes DBMS, tomaremos el correspondiente a Oracle desde /opt/jboss/docs/examples/jca/oracle-ds.xml a nuestro directorio opt/jboss/server/default/deploy.

Abriremos el archivo de configuración oracle-ds.xml antes mencionado y editamos la siguiente información:

```
<jndi-name>OracleDS</jndi-name>
<connection-url>jdbc:oracle:thin:@bnk:1521:XE</connection-url>
…
<driver-class>oracle.jdbc.driver.OracleDriver</driver-class>
<user-name>usuario</user-name>
<password>password</password>
```

**Los datos de host, puerto y SID se encuentran en el archivo network/ADMIN/tnsnames.ora del directorio de instalación de Oracle.*

Ahora editaremos (en algunas ediciones no existe este fichero, por lo que debemos crearlo) el archivo standardjaws.xml en /opt/jboss/server/default/conf, ya que por default JBoss está configurado para conectar con HypersonicDB, pero este no es el caso ya que nosotros lo haremos con Oracle.

Abrimos el archivo y lo cambiamos para que quede así:

```
<jaws>
<datasource>java:/OracleDS</datasource>
<type-mapping>Oracle10</type-mapping>
</jaws>
```

También le indicamos el nombre del datasource en el archivo standardjbosscmp-jdbc.xml y modificamos esta parte:

```
<datasource>java:/OracleDS</datasource>
```

Finalmente en el archivo login-config.xml, agregamos una <application-policy> con el nombre OracleDBRealm del siguiente modo:

```
<application-policy name = “OracleDbRealm”>
<authentication>
<login-module code =
“org.jboss.resource.security.ConfiguredIdentityLoginModule”
flag = “required”>
<module-option name = “principal”>sa</module-option>
<module-option name = “userName”>sa</module-option>
<module-option name = “password”></module-option>
<module-option name =”managedConnectionFactoryName”>
jboss.jca:service=LocalTxCM,name=OracleDS
</module-option>
</login-module>
</authentication>
</application-policy>
```

Ahora ya tenemos nuestro JBoss listo para poder utilizar una base de datos Oracle.

6. Bibliografía

Toda la información que aparece en este manual ha sido recopilada de distintas webs a continuación se citan algunas de ellas:

Comunidad JBoss (www.jboss.com)
MySQL (www.mysql.com)
Ubuntu (www.ubuntu.org)
Comunidad Ubuntu (www.ubuntuforums.org)
Java Mexico (www.javamexico.com)

Seguro que alguna se queda en el tintero, ya que el motor de búsqueda Google ha sido de gran ayuda para dar con esas webs escondidas que aportan gran cantidad de información y ejemplos o problemas de gente y situaciones reales.