Qué

Se ha de instalar Tomcat, un servidor de aplicaciones, que implemente HTTPS.

Diferentes usuarios accederán a sus respectivas aplicaciones a través de diferentes subdominios, y no podrán ver las aplicaciones de otros usuarios a través de la suya.

Estos usuarios podrán subir ficheros `.war` a través de FTP, a partir del cual se desplegará su aplicación web, a la cual podrán acceder desde su subdominio entrando por el puerto correcto o por un nuevo subdominio precedido por 'app.'.

Para qué

El objetivo de la práctica es añadir una forma de subir aplicaciones web en el servidor que se ha estado montando a lo largo del curso y que se desplieguen automáticamente.

Cómo

Instalación y configuración inicial de Tomcat

Tomcat se encuentra en los repositorios de Ubuntu con el nombre de `tomcat9`. Hay otros paquetes relacionados con Tomcat:`tomcat9-admin` y `tomcat9-user`, que son importantes para este proyecto, y `tomcat9-docs` y `tomcat9-examples` que no son necesarios, pero sí útiles.

Además, Tomcat tiene como dependencias los JRE y JDK de Java.

Por tanto, instalarlo se puede usar el siguiente comando, que instalará tanto Tomcat, como sus componentes opcionales importantes, como el JDK y JRE de Java:

```
sudo apt install tomcat9 tomcat9-{admin,docs,user} default-jdk -y
```

Con esto se crean dos instancias de `tomcat9`:

`tomcat9` como servicio:

- Localizado en `/var/lib/tomcat9`.
- Se gestiona a través del servicio `tomcat9`.
- Archivos de configuración en `/etc/tomcat9`, que tiene un enlace en `/var/lib/tomcat9/conf`.
- Páginas en `/var/lib/tomcat9/webapps`
- Registros en `/var/log/tomcat9`, que tiene un enlace en `/var/lib/tomcat9/logs`.
- Funciona automáticamente.
- Ya tiene páginas hechas, con páginas adicionales si se instalan los paquetes opcionales.

`tomcat9` en forma de scripts

- Localizado en `/usr/share/tomcat9`
- Se gestiona a través de scripts en `/usr/share/tomcat9/bin`.
- Archivos de configuración en `/usr/share/tomcat9/conf`. Esta carpeta no está hecha pero hay una de ejemplo en `/usr/share/tomcat9/etc`.
- Páginas en `/usr/share/tomcat9/webapps`.
- Registros en `/usr/share/tomcat9/logs`.
- Requiere de configuración.
- No tiene páginas hechas.

Como la primera instancia es más fácil de gestionar y poner a punto, y la segunda no aporta ningún beneficio a este proyecto, se usará la primera y se dejará la segunda como respaldo en caso de que haga falta recuperar archivos de configuración.

El directorio de Tomcat tiene permisos restrictivos para todo aquel que no sea el propietario o miembro del grupo propietario, así que se ha añadido al usuario a grupo `tomcat` para facilitar la gestión de los archivos de configuración:

```
sudo usermod $USER -aG tomcat
```

Los archivos de configuración por defecto ya están en la segunda instancia de Tomcat, pero si en cualquier momento se quisiese hacer copia de la configuración actual, la forma más rápida es usar este comando:

```
for i in `ls /var/lib/tomcat9/conf/*.xml`; do cp $i $i.bck; done
```

Por último, será necesario indicar con la variable de entorno `JAVA_HOME` dónde está localizado el JDK, que estará en algún subdirectorio dentro de `/usr/lib/jvm`. Para hacer permanentes estos cambios es necesario indicarlos en `/etc/environment`, que contiene variables de entorno que compartirán todos los usuarios.

```
export JAVA_HOME="/usr/lib/jvm/default-java"
echo 'export JAVA_HOME="/usr/lib/jvm/default-java"' |
sudo tee -a /etc/environment
```

En este punto del proyecto, asumiendo que el servicio esté activo, `tomcat9` estará activo en el puerto 8080, a través HTTP, mostrando una página por defecto.

Cambios en el servidor

Este proyecto está basado en el proyecto de HTTPS, que a su vez está basado en los proyectos de DNS y ProFTPd. En este proyecto se han hecho unos cuantos cambios en el servidor que merece la pena mencionar.

La carpeta personal de los usuarios virtuales de ProFTPd sigue siendo la misma, pero ahora toda la carpeta en la que está la web ha sido movida a un nuevo subdirectorio llamado `www`, incluida la carpeta compartida.

Esto requiere de modificaciones en `/etc/fstab` para que las carpetas compartidas sigan funcionando correctamente, de modo que donde antes ponía:

```
/var/www/cliente/compartido/ /var/www/cliente/user/compartido/ none bind 0 0
```

Ahora pone:

```
/var/www/cliente/compartido/ /var/www/cliente/user/www/compartido/ none bind 0 0
```

Los permisos son los mismos que eran originalmente, pero se ha de actualizar todos los `DocumentRoot` de las etiquetas `<VirtualHost>` en `/etc/apache2/sites-available/clientes.conf` para que Apache sirva la página correctamente.

Además se han habilitado módulos de proxy para Apache, y se ha creado un archivo de configuración nuevo para hacer de proxy de Tomcat, lo que se verá con más profundidad más adelante.

Respecto al servidor DNS, ahora hay dos subdominios por usuario, por ejemplo: user.2daw.iesmariaenriquez.es, y app.user.2daw.iesmariaenriquez.es.

Script de creación de usuarios

Se ha actualizado el script de creación de usuarios para coincidir con la nueva estructura de directorios, para añadir `<VirtualHost>` en un nuevo archivo para que Apache haga de proxy de Tomcat y para añadir `<Host>` para cada usuario dentro de `server.xml`, lo que se comentará más adelante en este documento.

Tomcat por HTTPS

Primero se tiene que generar un certificado SSL. Tomcat usa un formato diferente de claves, así que se tendrá que crear un certificado nuevo. Esto se consigue usando el comando `keytool`, que viene con `default-jdk`.

```
cd /var/lib/tomcat9
sudo keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.jks
```

Con esto se crea un certificado *Java Keystore* en la carpeta de Tomcat. Pedirá una contraseña, que es necesario recordar.

En `/var/lib/tomcat9/conf/server.xml`, se sustituye la etiqueta `<Connector>` , que es hija de `<Service>`, por una como esta:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS"
  keystoreFile="tomcat.jks" keystorePass="password"/>
```

Aquí, `keystoreFile` es la ruta al certificado, que si no se indica raíz es relativa al directorio en el que está la instancia de Tomcat. `keystorePass` es la contraseña de dicho certificado. Los atributos `SSLEnabled`, `scheme`, y `sslProtocol` sirven para habilitar HTTPS, `clientAuth="false"` permite usar certificados autofirmados, y `maxThreads` define un número máximo de hilos que puede usar servidor en este host en concreto (el protocolo `HTTP/1.1` puede usar varios hilos por conexión cuando sea necesario). Por supuesto, con `port` se indica el puerto por el que funcionará el servicio de `tomcat9`.

Finalmente, tras reiniciar el servicio, Tomcat funcionará por HTTPS en el puerto 8443.

Creación de un host para cada usuario

Se crean dentro de `/var/lib/tomcat9/conf/server.xml`, creando nuevas etiquetas `<Host>`, usando la que viene por defecto (la del `localhost`), como plantilla.

Se tiene que cambiar el atributo `name` para que su valor corresponda con el dominio del usuario, que se tendrá que crear en `bind` previamente. El atributo `appBase` será una carpeta con el nombre del usuario dentro del directorio `webapps`, dentro de `/var/lib/tomcat9`. Además, en este proyecto se pondrá una etiqueta `<Alias>` con el dominio del usuario, precedido por 'app', subdominio que también se tiene que crear en el servidor DNS.

En todas las etiquetas `<Host>` se tienen que tener los atributos `unpackWARs='true'`, que hace que Tomcat pueda descomprimir los `.war`, y `autoDeploy='true'`, que hace que lo haga automáticamente.

Debería quedar algo similar a lo siguiente:

```
<Host name='user.2daw.iesmariaenriquez.es' appBase='webapps/user'
  unpackWARs='true' autoDeploy='true'>
  <Alias>app.user.2daw.iesmariaenriquez.es</Alias>
</Host>
```

Ha de crearse un usuario virtual de ProFTPd, cuyo `UID` se usará para establecer al propietario del directorio. El grupo propietario será `tomcat`, y se le dará a ambos permisos de lectura, escritura y ejecución.

El directorio personal del usuario seguirá estando localizado en `/var/www/cliente/<username>`, así que para que el cliente pueda acceder a su directorio de aplicaciones web desde FTP, se tendrá que montar el directorio dentro de un subdirectorio en su carpeta personal.

```
sudo mkdir /var/lib/tomcat9/webapps/user
sudo chown <uid>:tomcat -R /var/lib/tomcat9/webapps/user
sudo chmod 775 -R /var/lib/tomcat9/webapps/user
sudo mkdir -p /var/www/cliente/user/app
sudo chown <uid>:tomcat -R /var/www/cliente/user/app
sudo chmod 775 -R /var/www/cliente/user/app

sudo mount --bind /var/lib/tomcat9/webapps/user /var/www/cliente/user/app
```

Y se añade la siguiente línea a `/etc/fstab` para hacer que los cambios sean persistentes:

```
/var/lib/tomcat9/webapps/user /var/www/cliente/user/app none bind 0 0
```

También se tiene que crear y poner a punto a los usuarios tal y como se indica mi práctica de HTTPS con apache, pero siguiendo las consideraciones especificadas en el apartado de **Cambios en el servidor** de este documento.

Alternativamente, se puede usar el script que he creado para llevar a cabo todas estas acciones e automáticamente.

Con esto, se consigue que el servicio de Tomcat pueda descomprimir archivo WAR y que el usuario pueda subir sus archivos su aplicación web con este mismo formato.

Para finalizar, se reinician los servicios de `apache2`, `named` y `tomcat9`.

Panel de administrador y restricción del dominio 2daw.iesmariaenriquez.es

Tal y como están las cosas ahora, aunque cada usuario tenga su propia página web en su subdominio, todas las páginas web son visibles desde el dominio principal, 2daw.iesmariaenriquez.es. Por ejemplo, la página de `user` es visible si se navega a `https://2daw.iesmariaenriquez.es:8443/user/ROOT`

Además, gracias al paquete `tomcat9-admin` también es posible administrar aspectos del servidor desde una interfaz gráfica en el navegador, lo que podría resultar útil en un escenario real.

Una forma de solucionar ambos problemas a la vez es cambiar el directorio por defecto de *localhost* y/o *2daw.iesmariaenriquez.es* al panel de control.

Se puede conseguir esto añadiendo una etiqueta `<Context>` en la etiqueta `<Host>` de dicho dominio que tenga valor del atributo `docBase` la ruta al directorio en el que se encuentra el panel de control, `/usr/share/tomcat9-admin/manager`, con el atributo `privileged="true"`, habilita la autentificación de los usuarios.

La etiqueta `<Host>` del dominio debería quedar así:

```
<Host name="localhost" appBase="webapps"
  unpackWARs="true" autoDeploy="true">
  <Context path="" docBase="/usr/share/tomcat9-admin/manager"
    privileged="true"/>
</Host>
```

Los usuarios que pueden entrar a estos paneles de control están definidos en `/var/lib/tomcat9/conf/tomcat-users.xml`, siempre y cuando el paquete `tomcat9-user` esté instalado.

En él se definen roles y usuarios que llevan a cabo estos roles, pero en este caso se usarán los roles predefinidos que piden los paneles de control y un usuario llamado `admin`.

Para ello, se añaden estas líneas en el archivo `tomcat-users.xml`:

```
<role rolename="manager-gui" />
<role rolename="manager-script" />
<role rolename="admin-gui" />
<user username="admin" password="password"
    roles="manager-gui, manager-script, admin-gui" />
```

Además, para poder usar el administrador de hosts, se tiene que añadir una etiqueta `<Listener>` en el archivo `server.xml`, como hija de la etiqueta `<Server>`, junto a las demás etiquetas de su tipo:

```
  <Server>
    <Listener className="org.apache.catalina.storeconfig.StoreConfigLifecycleListener"/>
    ...
```

Con esta configuración, al reiniciar el servicio `tomcat9`, las aplicaciones web de los usuarios sólo serán accesibles desde su subdominio pero podrán ser gestionadas desde el panel de control por el administrador.

Proxy de Apache a Tomcat

En el servidor hay un servicio de Tomcat, que despliega y sirve aplicaciones web, y un servicio de Apache, que sirve páginas web, pero no están conectados de ninguna forma.

Hay, principalmente, tres alternativas: - Crear una página con un enlace a la URI correcta y no hacer nada en el servidor. - Redirigir al puerto 8443 al entrar en una página o subdominio concreto. - Hacer de proxy en un subdominio desde Apache hacia Tomcat.

Si bien las dos primeras opciones son más sencillas, en este proyecto se ha optado por la primera porque es la más transparente para el usuario.

Para hacer funcionar el proxy primero se han de habilitar los módulos de `apache2`:

```
sudo a2enmod proxy "proxy-*"
```

Y después se tienen que añadir las siguientes directivas en `/etc/apache2/apache2.conf` para cargar y configurar los módulos:

```
ProxyPreserveHost On
SSLProxyEngine On
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off
```

- `ProxyPreserveHost On` evita que cambie el hostname al hacer de proxy.
- `SSLProxyEngine On` habilita hacer de proxy a través de HTTPS.
- `SSLProxyCheckPeerCN off`, `SSLProxyCheckPeerName off` y `SSLProxyCheckPeerExpire off` eliminan restricciones del proxy que pueden hacer que no funcione al ser servidores `apache2` y `tomcat9` servidores diferentes con configuraciones diferentes.

Después, se crea un archivo `/etc/apache2/sites-available/app.conf`, donde se guardará la configuración de los proxies a las aplicaciones web de los clientes. En estos `<VirtualHost>`, se escuchará por `app.<username>.2daw.iesmariaenriquez.es:443`, la directiva `ServerName` tendrá como valor `app.<username>.2daw.iesmariaenriquez.es` y `DocumentRoot` será `/var/www/cliente/user/app`, que es donde se ha montado la carpeta de Tomcat.

Un ejemplo de un `<VirtualHost>` podría ser:

```
<VirtualHost app.user.2daw.iesmariaenriquez.es:443>
    ServerName app.user.2daw.iesmariaenriquez.es
    DocumentRoot /var/www/cliente/user/app
</VirtualHost>
```

Habiendo habilitado los módulos de proxy, se tiene que añadir la siguiente directiva en todos los `<VirtualHost>` cada uno de los clientes:

```
ProxyPass / https://user.2daw.iesmariaenriquez.es:8443/
```

Este `ProxyPass` redirigirá todas las conexiones a este subdominio a Tomcat, pero es necesario que Tomcat también sirva a `app.<username>.2daw.iesmariaenriquez.es`, de ahí que se le pusiese una etiqueta `<Alias>` previamente.

Anexo: explicación de directivas y archivos de configuración de tomcat

policy.d

Hace accesibles las librerías y programas necesarios del sistema a tomcat y establece diversos permisos para que funcione el servicio.

Los archivos de este directorio se crean solos durante la instalación, y no deberían ser modificados.

catalina.properties

Establece variables y propiedades que usa Catalina durante su ejecución. No todas son menester para el funcionamiento del servicio, pero muchas de las que no lo son sirven para optimizar el rendimiento, como los filtros que ignoran JAR's innecesarios para la creación de las aplicaciones a partir de archivos `.war`

Este archivo se crea solo durante la instalación de catalina, y no debería ser tocado.

web.xml

Mapea los *servlets* de java necesarios para el funcionamiento y despliegue correcto de las aplicaciones web, como `catalina` y `jsp`.

Además, en él se definen los tipos de datos que conoce el servidor Tomcat.

Catalina

Es un directorio en el que se crean automáticamente subdirectorios correspondientes a los Hosts definidos en `server.xml`.

En estos subdirectorios se pueden hacer mapeos estáticos de aplicaciones web concretas.

Por defecto, se crean varias páginas en el directorio `localhost`, correspondientes a los paquetes de Tomcat que se han instalado: - `docs.xml` si se ha instalado `tomcat9-docs`. - `host-manager.xml` y `manager.xml` si se ha instalado `tomcat9-admin`. - `examples.xml` si se ha instalado `tomcat9-examples`.

Estos archivos tienen etiquetas `<Context>` similares a la definida dentro de `<Host>` en `server.xml` en el apartado de **Panel de administrador y restricción del dominio 2daw.iesmariaenriquez.es**.

El resto de directorios que se creen estarán vacíos por defecto.

tomcat-users.xml

Se definen usuarios y grupos. En el archivo de configuración por defecto vienen, comentados, grupos predefinidos de Tomcat.

Se puede gestionar el acceso a ciertas páginas usando este fichero.

server.xml

Es el archivo de configuración principal. Esta compuesto por:

- La etiqueta raíz `<Server>`
  - Etiquetas `<Listener>`, que habilitan servicios que ayudan al servidor a funcionar correctamente.
  - Etiqueta `<GlobalResources>`, donde se definen recursos globales, principalmente la base de datos de Catalina dentro de una etiqueta `<Resources>`.
  - Etiqueta `<Service name="Catalina">`, donde se define la configuración del servicio de Catalina, la base de datos que usa Tomcat.

* Etiquetas `<Connector>`, donde se define el puerto por el que escucha el servicio de `tomcat9`.
* Etiqueta `<Engine>`, donde se define el sistema gestor de bases de datos, que será catalina.
    - Etiquetas `<Realm>`, donde se instancia la clase `Catalina`-
    - Etiquetas `<Host>` que, junto a la etiqueta `<Connector>`, es de las pocas cosas que hay que tocar en este archivo para los propósitos de este proyecto. En estas etiquetas se definen dominios correspondientes a cada conjunto de aplicaciones web para cada usuario.

Anexo: Creación y subida de archivos WAR por parte del cliente

El cliente tiene que tener Maven y un JDK de java instalado. En Ubuntu, se instala de esta forma:

```
sudo apt install maven default-jdk
```

Después, tendría que establecer la variable de entorno en el archivo de configuración de su shell, que en el caso de bash es `~/.bashrc`:

```
echo 'export JAVA_HOME="/usr/lib/jvm/default-java"' >> ~/.bashrc
source .bashrc
```

Para generar una aplicación web desde la línea de comandos usando Maven se ejecuta el siguiente comando:

```
mvn archetype:generate -Dfilter=org.apache.maven:maven-archetype-webapp
```

Este comando es interactivo, y pedirá al usuario que elija una versión de la plantilla del proyecto, un nombre y demás.

Una vez creado el proyecto, el usuario puede crear su aplicación web dentro de `src/main/webapp`.

Cuando haya acabado, tendrá que compilar el proyecto para crear el archivo WAR:

```
mvn clean install
```

Tras habrá un archivo `<proyecto>.war` en el directorio `target` de la raíz del proyecto.

Para subir la aplicación tendrá que entrar desde SFTP, por el puerto 2222, y subir la aplicación el directorio `app` dentro de su carpeta personal.

Conclusión

En esta práctica finalmente todas las prácticas anteriores se han juntado y han cobrado sentido (menos SSH). Ha sido satisfactorio poder completar el servidor.

De todos modos, Tomcat es un poco confuso (principalmente porque los mensajes de error son *stack traces* de Java), así que no se como sentirme respecto a este servicio en concreto.

Además, sólo sirve con aplicaciones en Web en Java, y considero que eso no es demasiado hoy en día. Habría preferido ver algo similar pero para una tecnología web más actual, como JavaScript.