**Curso Apache Tomcat**

Servidor de aplicaciones web que se ejecuta en java

Las jdk son las como unas librerias

Estamos usando Ubuntu 20.04.4

Vamos a usar apache Tomcat 10 que es el estable actualmente

Aquí témenos una tabla con las versiones de Tomcat y además con la versión de java que utilizan:

<https://tomcat.apache.org/whichversion.html>

Usuario Ubuntu: jose

Contraseña Ubuntu: jose123

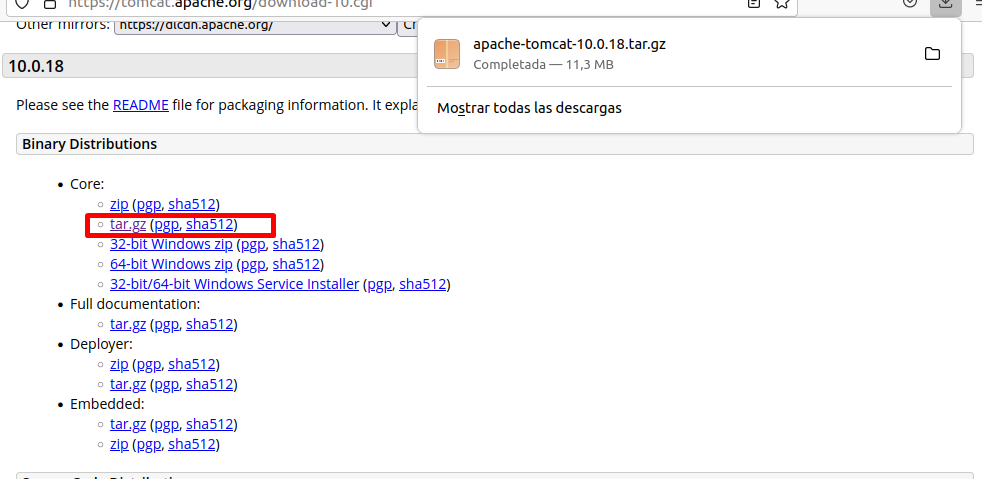
**Instalar Openjdk**

Vamos a la página de OpenJDK e instalamos la versión adecuada en nuestro caso la 8

**sudo apt-get install openjdk-8-jre**

**Instalar y descargar Tomcat Ubuntu**

Nos vamos a la página de Tomcat e instalamos la versión que hallamos elegido en nuestro caso la 10, no es recomendable instalar una versión que no sea estable.

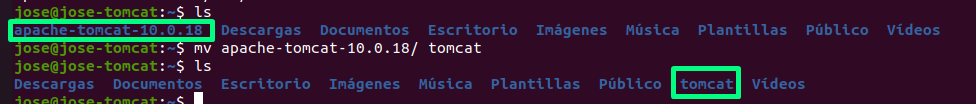


Lo siguiente es descomprimir el archivo

**tar xvf Descargas/apache-tomcat-10.0.18.tar.gz**

Nos creará un directorio, si queremos podemos renombrarlo.

**mv apache-tomcat-10.0.18/ tomcat**



Con esto ya estaría instalado tomcat

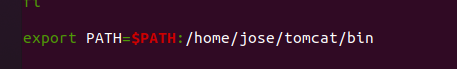
Para que todo se mas sencillo vamos a colocar el directorio bin en el path del sistema.

Para ello en mi caso:

**cd /home/jose**

**nano .bashrc**

Y dentro del archivo colocar lo siguiente (La ruta de vuestro bin de tomcat), Esto Sirve para que cada vez que usemos un script de Tomcat no tener que poner el PATH completo

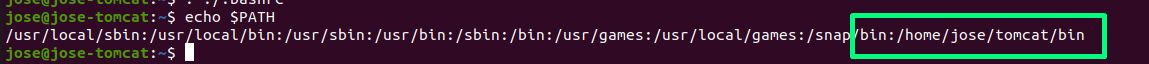


Ahora con el siguiente comando vamos a hacer que el sistema guarde el PATH

**. ./.bashrc**

Ahora hacemos un echo del PATH para ver ya ha sido modificada dicha variable

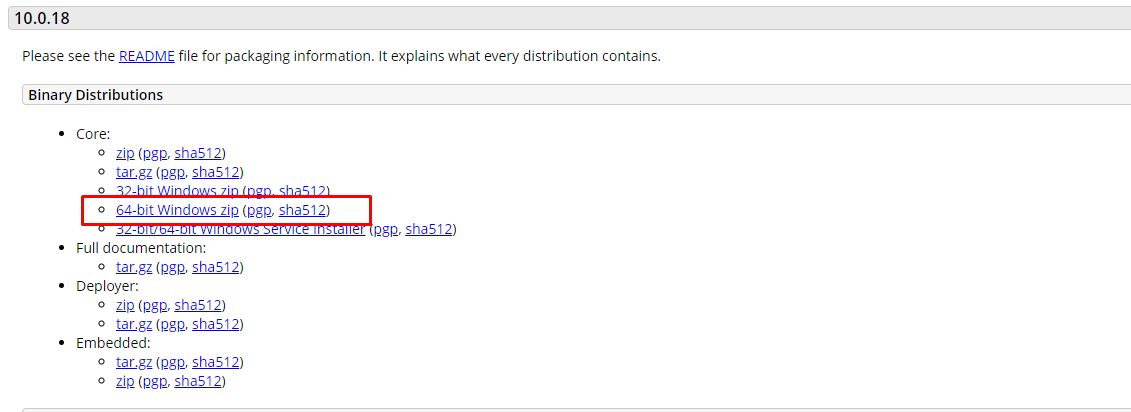
**echo $PATH**



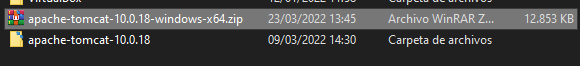
Windows

Tener instaladas las OpenJDK

Tras esto descargar la versión de Tomcat en mi caso la 10



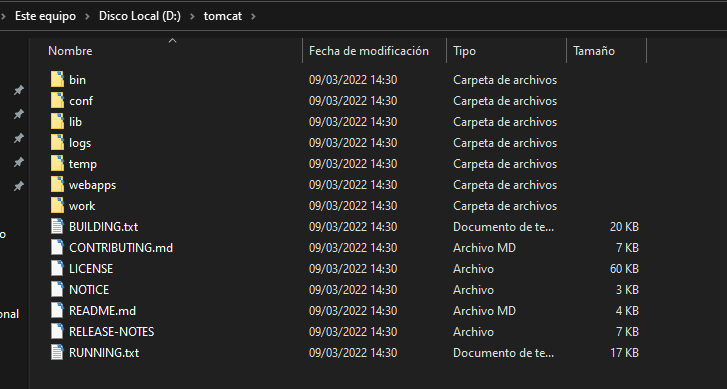
Después extraer el fichero.



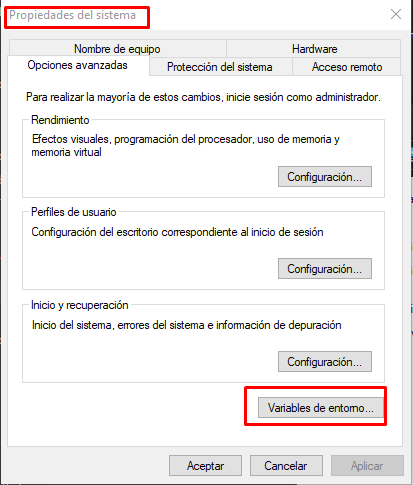
Lo renombramos para que sea más fácil, y lo cortamos al directorio que queramos en mi caso el D.



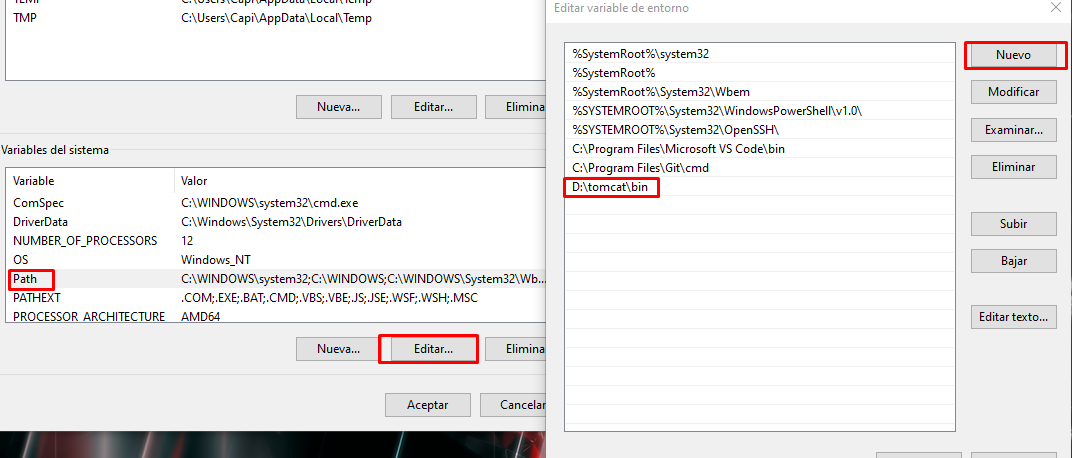
Dentro habrá lo mismo que en Ubuntu



A continuación, solo queda meter el directorio en la variable PATH de Windows. Para ello en propiedades del sistema nos vamos a variables de entorno.



Y buscamos las variables del sistema pincahmos en Path y editamos, vamos a nuevo e introducimos la ruta.



Y con esto ya lo tendríamos configurado.

**ARRANCAR TOMCAT**

Para arrancarlo lo mejor es definir una variable java home, para que así tomcat cuando arranque encuentre el java sin ningún problema para ello:

**cd /home/jose**

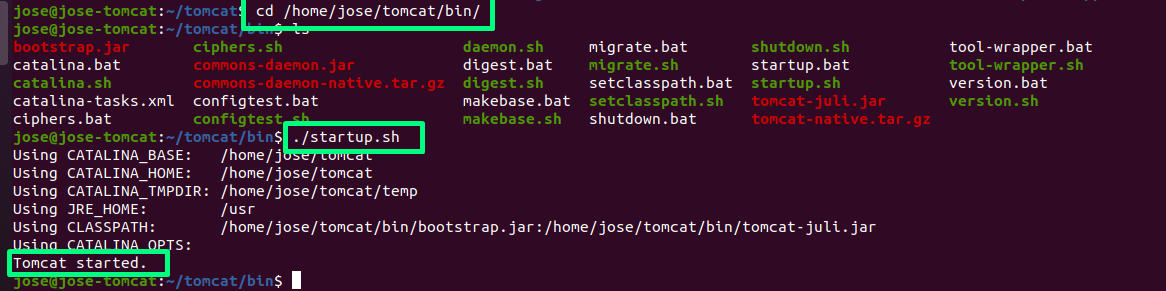
**nano .bashrc**

Introducir la ruta donde tengamos las OpenJDK



Una vez hecho esto nos vamos a el directorio bin de nuestro tomcat y vamos a ejecutar el script para iniciar el tomcat.

**./startup.sh**



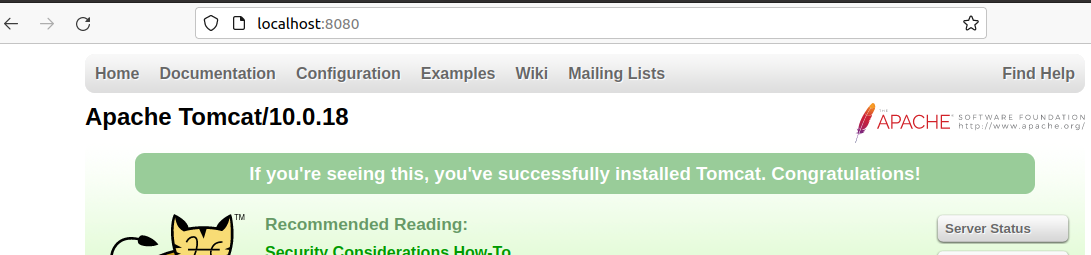
Como observamos ya está arrancado.

Con el siguiente comando podemos ver los procesos java que se han lanzado y funcionando.

**ps -ef | grep java**

Tras esto para comprobar que todo ha ido bien nos vamos al navegar he introducimos lo siguiente.

**Localhost:8080**



**Ficheros importantes**

server.xml (Fichero de configuración principal)

catalina.properties.xml (Fichero de configuración de propiedades del sistema, librerías y creación de variables)

**FICHERO CATALINA**

catalina es donde es el script que arranca, apaga todo tiene varias opciones de ejecución. Si ponemos **catalina.sh** nos salen las opciones

Editar primero el siguiente fichero

**nano .bashrc**



**catalina.sh configtest** nos hará un test para ver si funciona

Recomendable hacerlo primero con tomcat parado para comprobar que la configuración sea correcta y después repetir con el tomcat arrancado

NO ES RECOMEDABLE MODIFICAR EL FICHERO

**Arrancar y parar Tomcat**

**Ps -ef | grep java** Ver el estado

**catalina.sh** **start** Arranca el tomcat

**catalina.sh** **run** Arranca el tomcat pero en primer plano

**catalina.sh stop** Para el tomcat

**catalina.sh start/stop 5** Arranca o para en 5 segundos

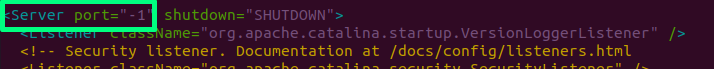
**Puerto de SHUTDOWN**

Para su configuración hay que acceder al fichero server.xml

**cd /home/jose/tomcat/conf**

**nano server.xml**

En la siguiente línea pondremos el puerto -1 ya que esto significa que lo desactivaremos ya que no es recomendable tenerlo activado ya que cualquiera que acceda al localhost por el puerto que estaba antes puede apagar el tomcat.



Otra opción sería cambiar la cadena SHUTSOWN por otro distinta.



**Variables**

**CATALINA\_OPTS** Afecta solo a tomcat.

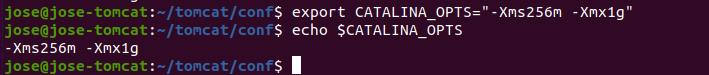
**JAVA\_OPTS** Afecta a todas las aplicaciones java que tengamos.

Se usan para configurar ciertos aspectos

Por ejemplo:

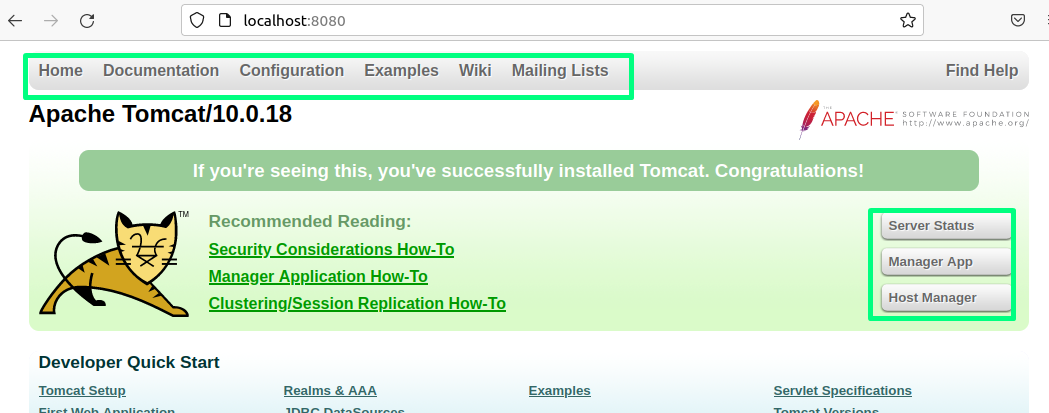
Para el tamaño mínimo que debe tener la máquina para arrancar y tamaño máximo en memoria:

export CATALINA\_OPTS=" -Xms256m -Xmx1g"



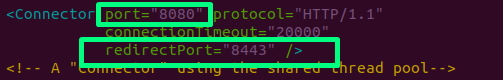
**Página inicial de tomcat**

Si accedemos a ella a través de localhost: "puerto" ver podemos ejemplos del funcionamiento y también algunas herramientas de configuración básicas.



**Cambiar puerto de tomcat**

Desde el archivo de configuración server.xml ponemos cambiar una serie puertos y los protocolos y la redirección, vamos a modificar el puerto por defecto y la redirección.



**SERVLET Y JSP**

Los servlet es una clase java muy concreta que se comporta de una determinada manera ante peticiones y respuestas http.

Los JSP permiten crear aplicaciones web dinámicas de forma sencilla, mezcla HTML, CSS, etc.

Cuando a tomcat le llega una JSP lo compila y lo convierte en servlet

**Tipos de ficheros**

**WAR** 🡪 fichero que contiene una aplicación de tipo web (Web archive) Se despliega dentro de tomcat.

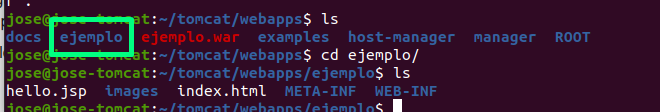
**JAR** 🡪 Archivo de tipo librería (JAVA archive JDBC).

**EAR** 🡪 Enterprise archive (No se usa en tomcat).

**Desplegar aplicación mediante WAR**

Descargar el archivo .war, luego nos iremos al directorio webapps y copiaremos la aplicación (.war),en él, en unos segundos se desplegará sola (Lo convierte en un directorio).

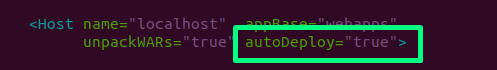




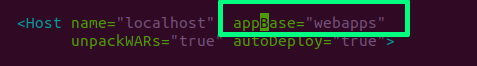
Lo podemos probar de la siguiente manera:



Si queremos deshabilitar la opción de que se despliegue automáticamente, debemos acceder al fichero server.xml que está en el directorio conf y cambiar lo siguiente:



Se despliega automáticamente porque está en ese directorio y también tenemos el unpackWARS que es si quiere que desempaquete los .war recomendable dejarla en true



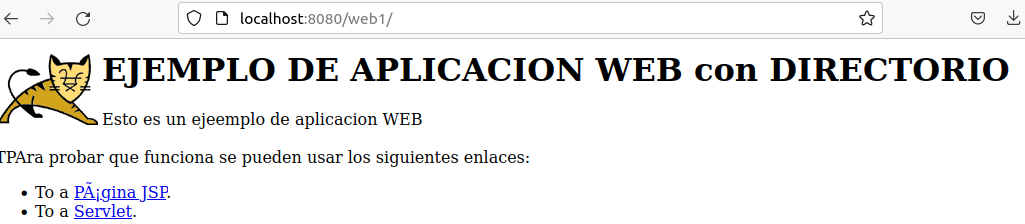
**Desplegar aplicación mediante directorio**

Una vez tengamos el directorio con los archivos de la aplicación es tan simple como copiarlo el directorio en el directorio webapps y listo.





Para probarlo:

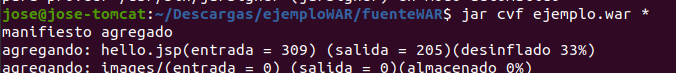


**Redesplegar una aplicación**

Para ello necesitamos todos los archivos que forman la app y cambiar algo como el index.html

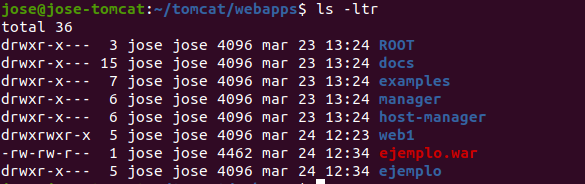


Una vez echo esto lo vamos a comprimir en .war vamos a utilizar el comando jar que viene instalado con las openjdk. Si estamos en el directorio con poner \* nos coge todo lo de dentro.



Tras esto lo copiamos al directorio webapps y como tiene el mismo nombre la redespliega





Prueba

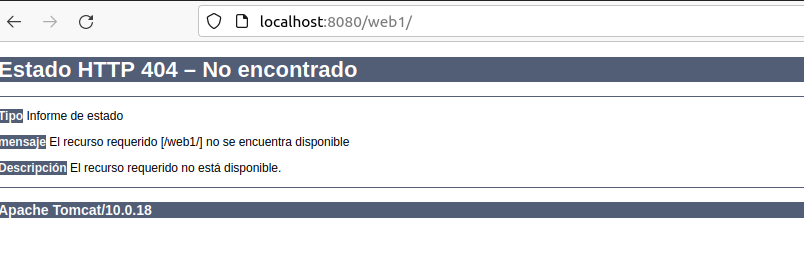


**Eliminar aplicaciones (undeploy)**

Nos vamos al directorio webapps y en el caso de haberlo desplegado como directorio es tan fácil como eliminar el directorio.



Prueba



Si lo hemos desplegado mediante .war será tan sencillo como eliminar el .war y automáticamente eliminará el directorio.



Prueba

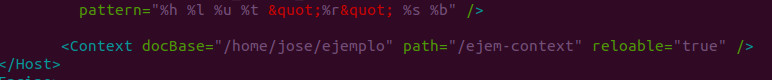


**Contexto**

Donde se despliegan las aplicaciones, por defecto webapps pero podemos declarar contextos diferentes para determinadas aplicaciones.

Si ponemos ejem-context en la url accederá a la ruta que le indicamos en el docBase.

El directorio debe tener permiso de lectura y escritura



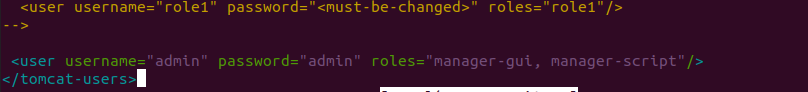
En el directorio de ejemplo copiamos el archivo .war y ya estaría listo.

**Configurar tomcat-users.xml para usar Tomcat Manager WEB**

Debemos ir al dichero tomcat-users.xml en el directorio conf y crear una entrada en la cual introducimos el nombre, contraseña y el rol que va a tener:

manager-gui 🡪 Para acceder al Manager graficamente

manager-script 🡪 Para acceder al Manager desde línea de texto



**Despliegue y borrado de aplicaciones con Tomcat Manager**

**Modo gráfico**

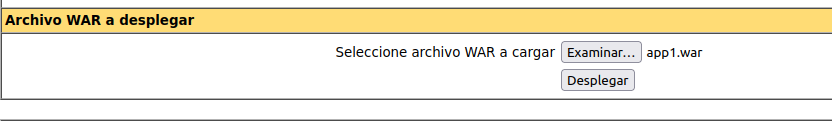
Lo primero es entrar en el manager desde la página por defecto o desde la ruta:

**localhost:8080/manager/html**

Una vez dentro tenmos dos opciones:

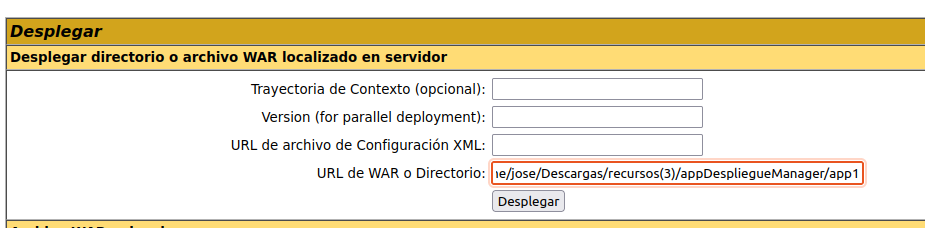
Desde un .war

Seleccionar el archivo y desplegar

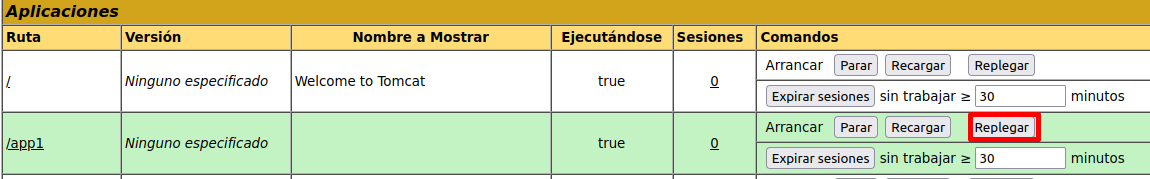


Desde un directorio:

Tan fácil como poner la ruta del diorectorio

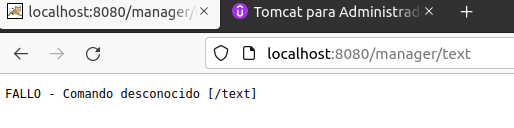


Para borrar cualquier aplicación



**Modo texto**

Para acceder al modo texto



Nos dará fallo porque no hemos introducido ningún parámetro.

Estos son algunos parámetros:

/list

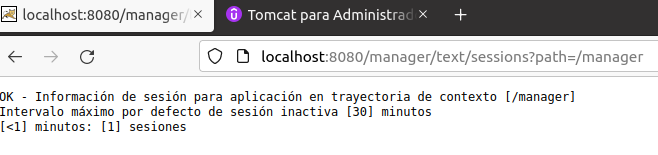
/serverinfo

/vminfo

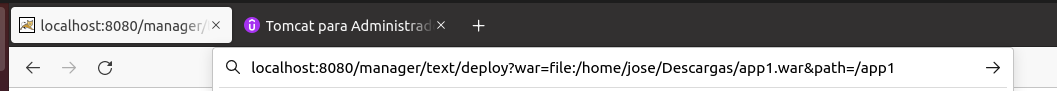
/sessions?path=/app1 (La interrogación es para añadir argumentos)

/sessions?path=/manager

/threaddump

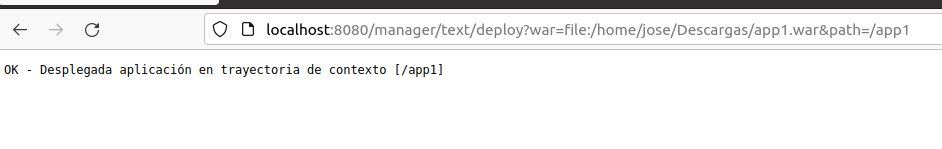


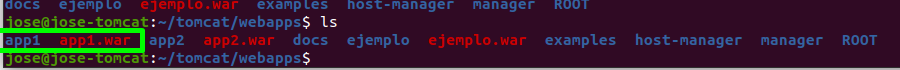
Para desplegar una aplicación en modo texto



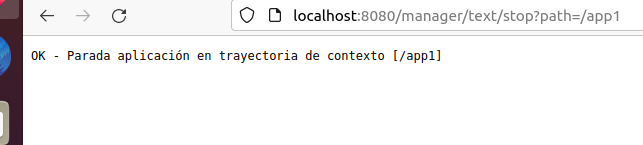
Hay que introducir la ruta del .war y el path (Es como el nombre de la carpeta), si no lo ponemos nos lo hará en webapps que es la por defecto.

Esto funciona si es en la máquina local

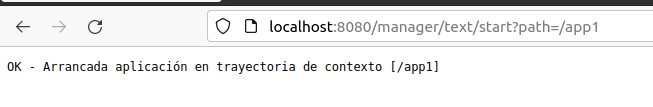




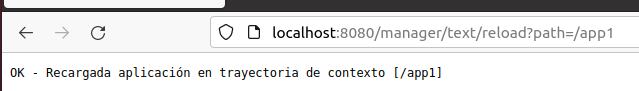
Parar aplicaciones en modo texto



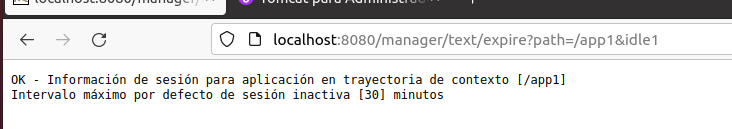
Para arrancar en modo texto



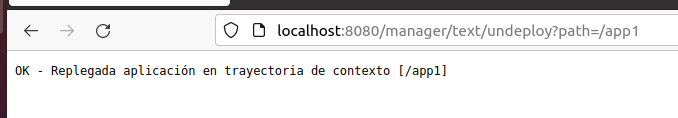
Recargar



Para expirar las sesiones (idle son los minutos que llevan sin hacer nada la app)



Para eliminar aplicaciones



**LOGS**

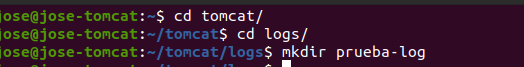
Se encuentra en logging.propertiesen el directorio conf

handler definen hacia dónde va a ir la información que entra de los logs.

.handler defiene que handler usar si la aplicación no indica ninguno (Por defecto)

**Modificamos el comportamiento de manager**

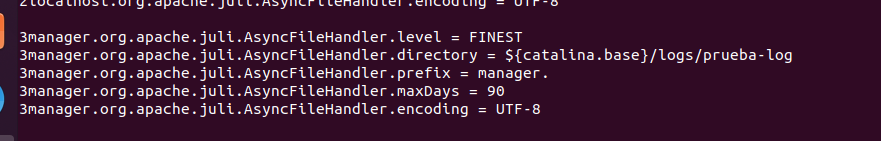
Creamos una carpeta donde vamos a guardar los logs



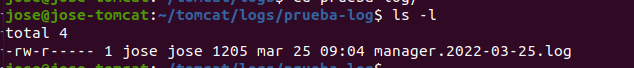
Nos vamos al archivo logging.properties

FINEST = Más nivel de log

Editamos la ruta añadiendo el directorio (prueba-log)

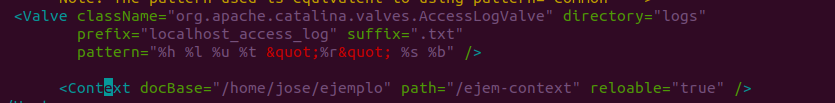


Y si nos vamos al directorio veremos que se generan los logs ahí del manager.



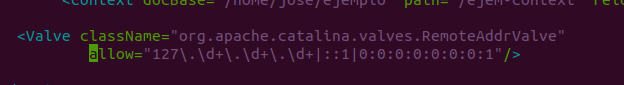
**Valves o válvulas**

Genera información sobre los clientes y lo guarda en .txt en el directorio logs

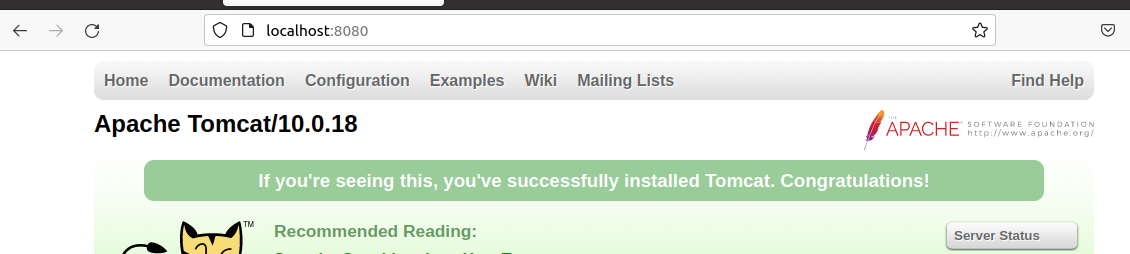


**Crear válvula para permitir o denegar el acceso (remoteAddrValve)**

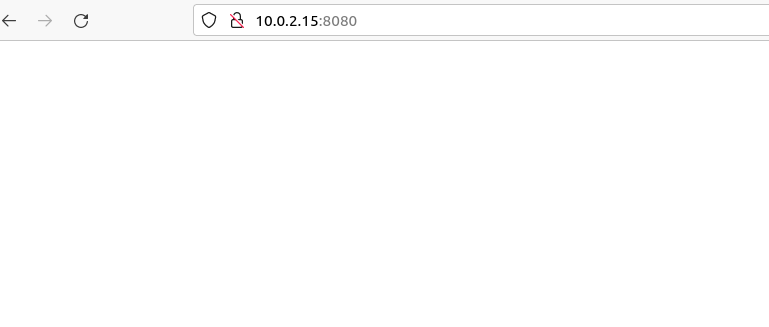
Configuramos el nombre y luego le decimos el rango de ip que permito, es decir solo accesos de la 127.lo que sea (accesos locales).



Si intento acceder a Tomcat como localhost si me deja



Pero si accedo con mi ip no



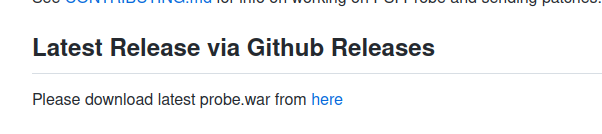
**JDBC Y JDNI**

Si una aplicación que tengo en Tomcat necesita acceder a una base de datos, hay que crear en Tomcat un recurso JDBC para permitir que se conecte a esa base de datos, a ese recurso le pondré un nombre y ese nombre deberá estar registrado en el JDNI para poder ser localizado.

**Instalar la utilidad Probe**

<https://github.com/psi-probe/psi-probe>

Nos permite ver con mas detalle los recursos que creemos.



Es un .war, lo copiamos a webapps



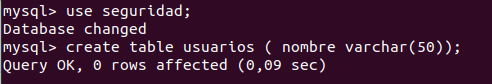
**Preparar una Base de datos**

Con este comando accedemos como root



Cremamos la base de datos





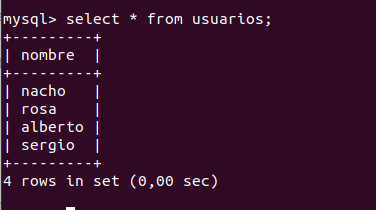
Y creamos varios insert



Después hacemos un commit



Y vemos lo creado



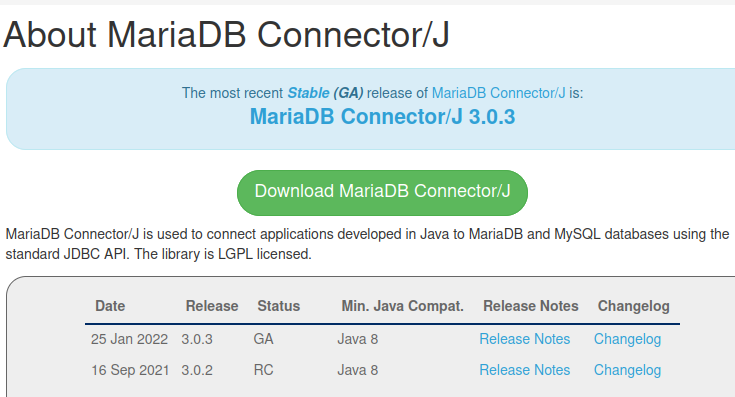
Y creamos un usuario y le damos los permisos



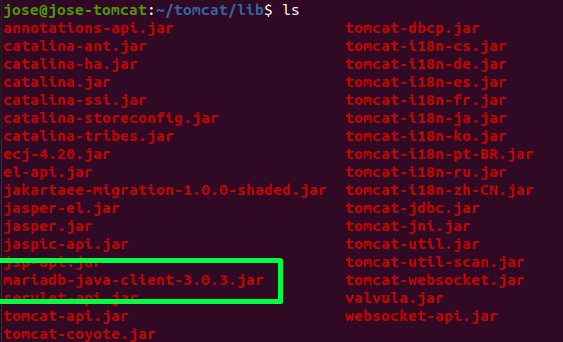


**Driver JDBC**

Descargamos el driver en .jar



Despues lo copiamos al directorio bin del tomcat

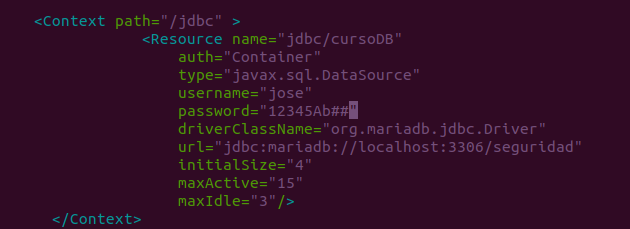


**Configurar la conexión JDBC**

Nos vamos al archivo server.xml

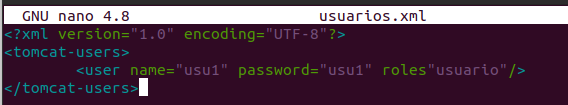
name como se va a llamar el recurso dentro del JDNI

driveClassName hay que mirarlo en la documentación

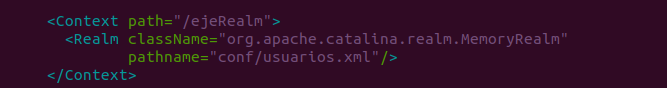


**Ejemplo de REALM (Seguridad)**

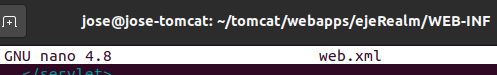
Vamos a ver un ejemplo de real, para ello desplegaremos una aplicación y crearemos en el directorio conf un archivo xml con usuarios



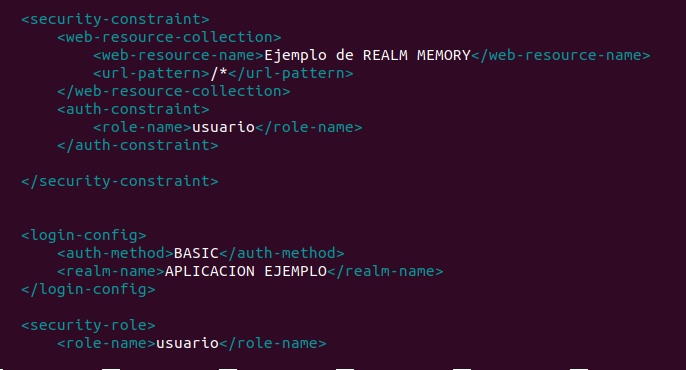
Después en el archivo de server.xml crear un contexto que apunte al archivo que hemos creado.



Una vez esto vamos al directorio de la aplicación y en WEB-INF abrimos el archivo web.xml



Y ahí debemos de poner algo así, para que solo puedan acceder usuarios que estén en el rol de usuario.

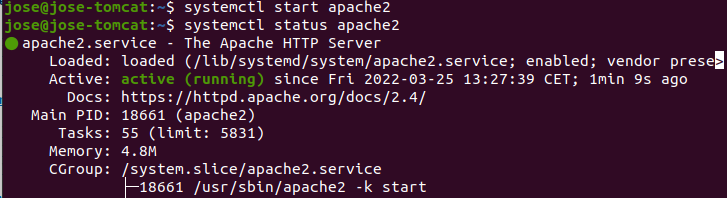


**Conectar Tomcat con servidor Web**

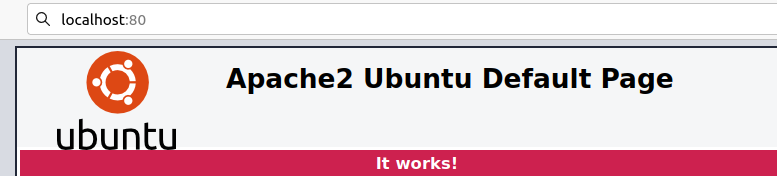
Instalamos apache

**sudo apt install apache2**

Lo iniciamos

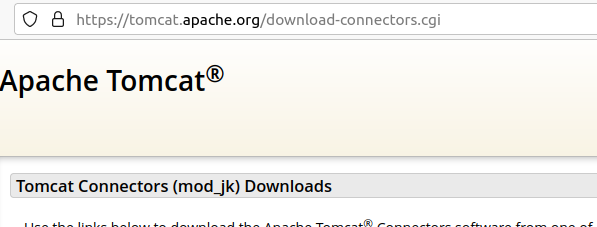


Comprobar que funciona

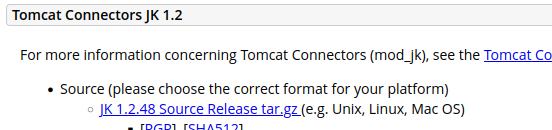


**Instalar el módulo que nos va a permitir conectar Apache con Tomcat**

El modulo se llama mod jk



Descargamos lo siguiente



Nos hacemos root e instalamos los siguientes paquetes.



Comprobamos que está instalado lo siguiente



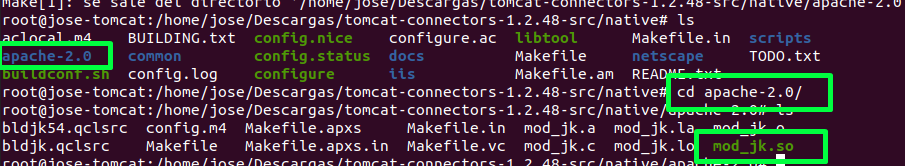
Descomprimimos el archivo que hemos descargado antes, entramos en el directorio que crea. Entramos al directorio navite y ejecutamos lo siguiente.



Y después



Y ya estaría listo nuestro archivo



Ý lo copiamos en mi caso la siguiente ruta (Si existe la carpeta modules copiarlo ahí)



Conectar apache y tomcat

Clusters en Tomcat

Persistencia de sesión

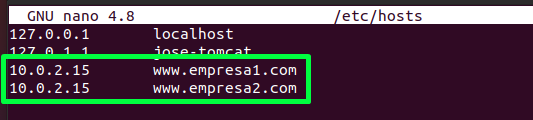
**Virtual Hosts**

En el archivo server.xml dentro de engine vamos a definir los dos virtual hosts

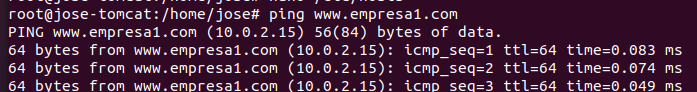


Alias 🡪 Nombre con el que responde el virtual hosts

Luego desde el root accedemos al archivo de configuración /etc/hosts para introducirlas ip que va a reconocer la máquina



Como vemos da ping



Desplegamos 2 aplicaciones





Arrancamos el Tomcat y si ponemos el alias en el buscador veremos que podemos acceder a la web.

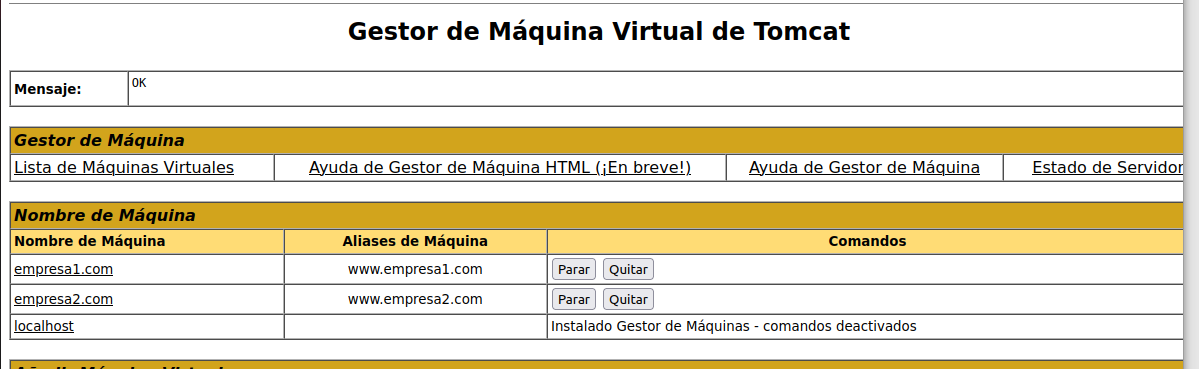
**Virtual Host Manager**

Añadir host de forma gráfica

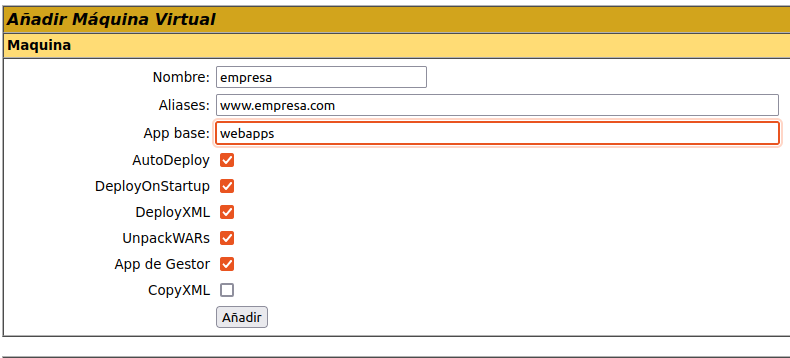
Para ello debemos crear un usuario con los siguientes roles en el fichero tomcat-user.xml



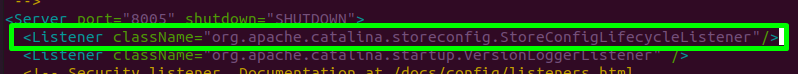
Y ya podremos acceder a ella desde el host manager



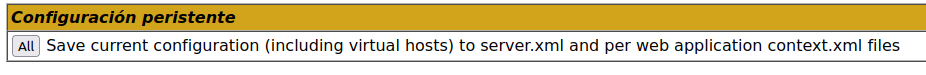
Desde aquí se puede desplegar mas host, recodar meterlo en el fichero /etc/host



Para que funcione correctamente habrá que modificar el fichero de server.xml



Luego es importante seleccionar lo siguiente



Y como vemos ya está listo

