Actividad 2



CONFIGURACIÓN DE SERVIDOR

Sara Dozzi |2º DAW-Despliegue de Aplicaciones Web

# Instalación del software necesario para el despliegue de aplicaciones Web.

Vamos a realizar la instalación de los servicios en una MV de Ubuntu 20.04.

1. Instalación de **Apache Tomcat**:

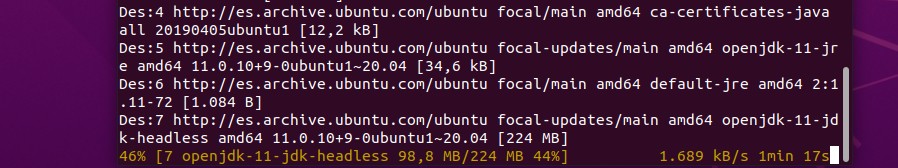
Primero procedemos a instalar JAVA por consola con los siguientes comandos para actualizar las listas de los paquetes del repositorio.:

Escogeremos instalar openJDK 8..

-**sudo apt-update**: actualizamos el repositorio de apt , es decir el indice de paquetes.

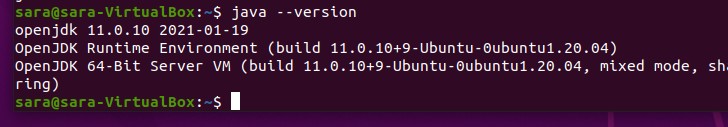
-sudo **apt-get upgrade**

-**sudo apt install** default-jdk



Ejecutamos el comando para ver la versión instalada de jdk y nos muestra la versión 11.:

java –version



A continuación, instalamos el entorno de ejecución "RuneTimeEnvironment":

sudo apt-get install default-jre

Si queremos instalar versiones concretas de JDK esta sería la sintaxis: sudo apt install openjdk-**11**-jdk

O que nos muestre otras versiones disponibles mediante el comando siguiente, que nos permita cambiar la versión sin volverla a instalar.

## -sudo update-alternatives --config java

Creación de usuario para Tomcat.

Existe la opción de crear un usuario sin privilegios, por seguridad, para el uso del servidor. 1-Creamos grupo

sudo groupadd tomcat

2-Creamos el usuario y lo añadimos al grupo "tomcat" y al directorio de instalación de tomcat9.

## sudo useradd -s /bin/false -g tomcat -d /etc/tomcat9

En nuestro caso instalaremos directamente la version 9 de tomcat, y primero comprobamos que está disponible en el repositorio de linux.

Nota: Con el comando cat podemos ver los grupos y usuarios creados:

El grupo tomcat tiene el UID 1001, por lo que el usuario "tomcat" pertenece a dicho grupo como se muestra con el siguiente condo:

**cat** /etc/pssd---> para listar usuarios:

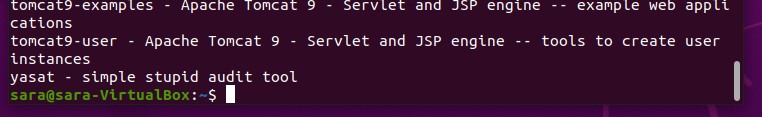
**cat**/ etc/group--->para listar grupos



Para finalizar, asignamos permisos de usuario a carpeta y grupo respectivamente sobre el directorio indicado de tomcat9.

**-chown** -R tomcat:tomcat /etc/tomcat9



- sudo apt-cache search tomcat

**Tomcat** es un contenedor web de servlets y jsp y puede funcionar como servidor web por sí mismo. Está desarrollado por Apache software y un grupo de desarrolladores independientes. Fue escrito en java y funciona en cualquier S.O que tenga JVM

Primero listamos el software disponible para instalar tomcat mediante el siguiente comando:

## -sudo apt-cache search tomcat

Procedemos a la instalación de Tomcat en Ubuntu, mediante el gestor de paquetes aptget.

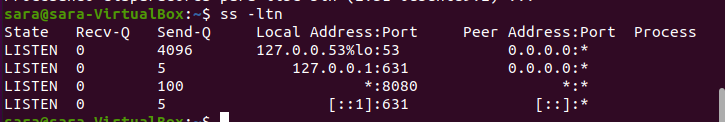
## -sudo apt install tomcat9 tomcat9-admin

## tomcat 9-docs

## tomcat 9-examples

Una vez instalado comprobamos el puerto disponible con el comando: **ss -ltn**

y se muestra el puerto de escucha para conexiones entrantes **8080**:



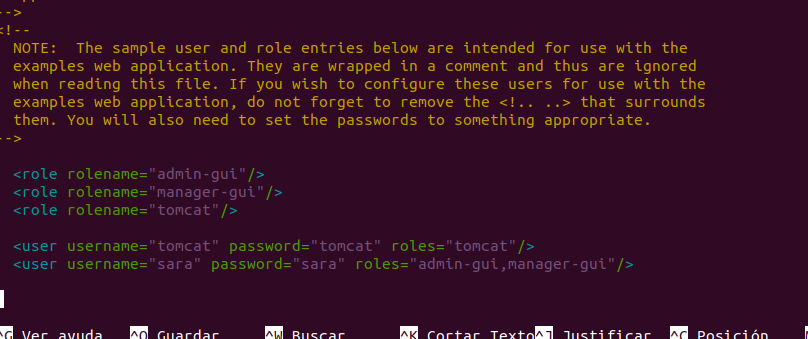
Por defecto, el servidor está ejecutandose pero se puede detener con los siguientes comandos:

-sudo systemctl enable/disable tomcat9.

Editamos el usuario administrador en el fichero:

sudo nano /etc/tomcat9/tomcat-users.xml

Editamos el fichero y asignamos contraseña de acceso para la gui del administrador agregando las siguientes lineas:



Reiniciamos tomcat para que se efectúen los cambios.

## -sudo systemctl restart tomcat9.

* Para permitir el trafico en el puerto 8080 de tomcat, utilizaremos el siguiente comando para incluirlo en el firewall:

**sudo ufw allow** from any to any port 8080 proto tcp

# Instalación de Apache.

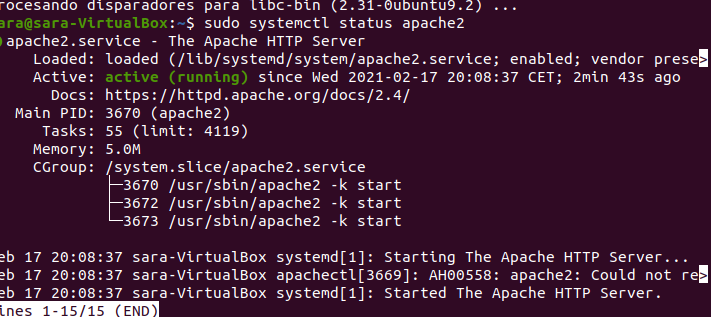
Para comenzar actualizamos repositorio y después instalamos.

sudo apt update

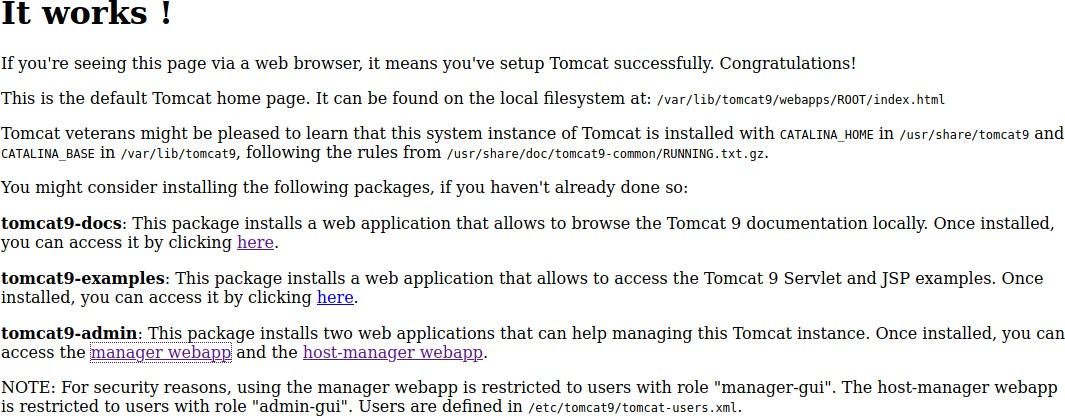
sudo apt install apache2

Comprobamos que se ha instalado:

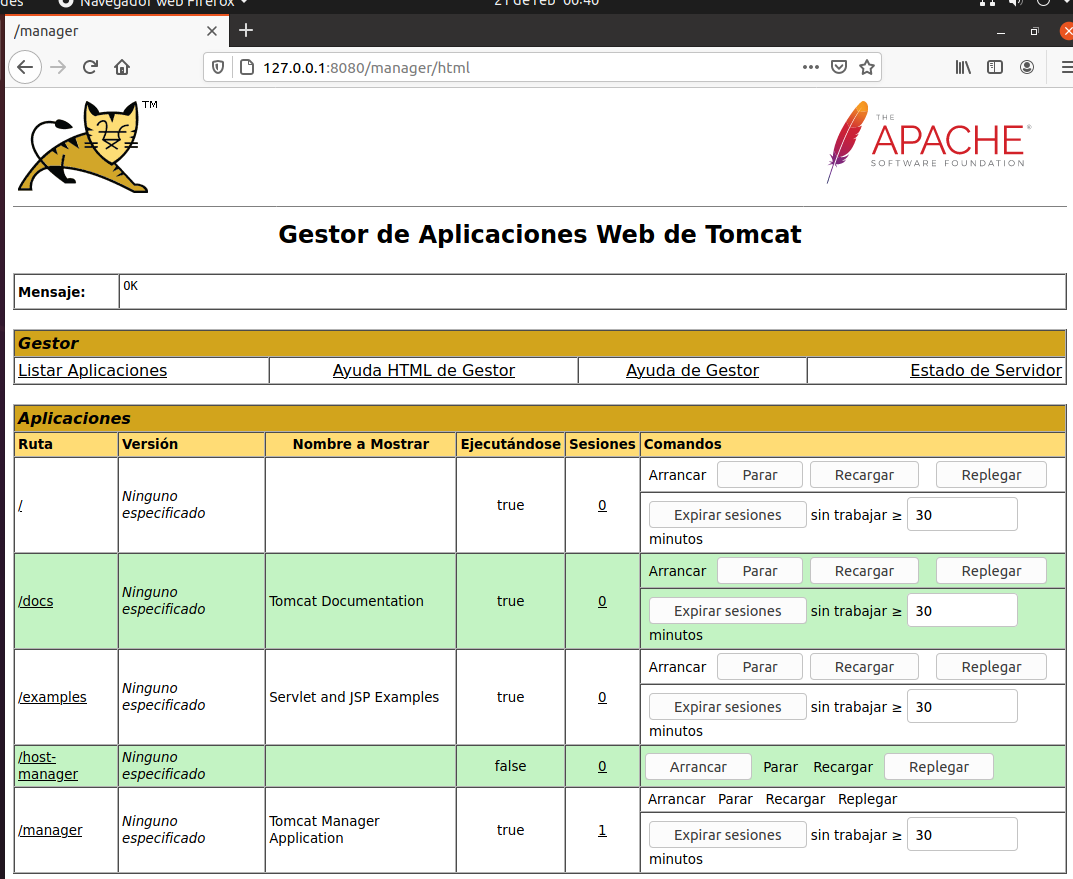
sudo systemctl status apache2

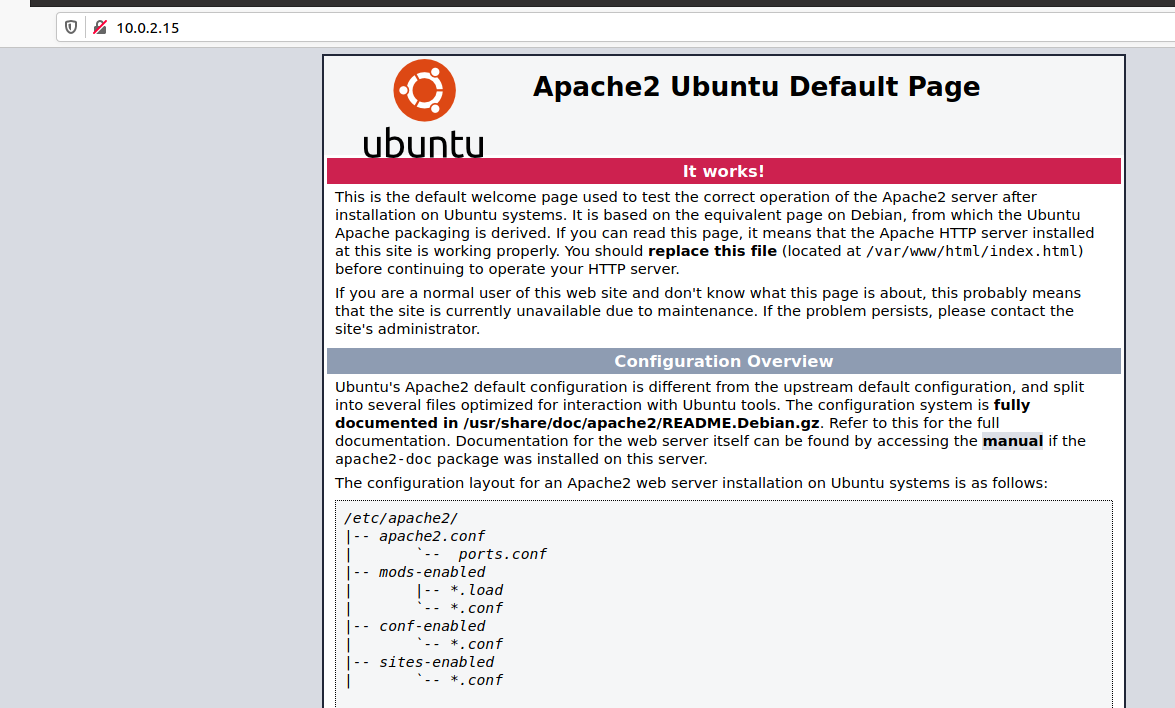


Realizaremos una petición web al servidor apache a partir de nuestra ip local:



introducimos el usuario y contraseña configurado en el fichero **tomcat-users.xml.**





Dicha página nos muestra que está en ejecución en nuestro equipo. Los comandos utilizados para iniciar o detener el servicio son: sudo systemctl **start/stop** apache2

O bien reiniciarlo:

sudo systemctl **restart** apache2 Para habilitarlo o desabilitarlo:

sudo systemctl **enable / disable** apache2

Como dato destacable apache se puede complementar con modulos en ubuntu, por ejemplo para instalar un modulo MySql:

**sudo apt install libapache2-mod-auth-mysql**

# Instalación de Maria DB.

Es un Sgbd relacional de codigo libre y es la alternativa a MySQL.

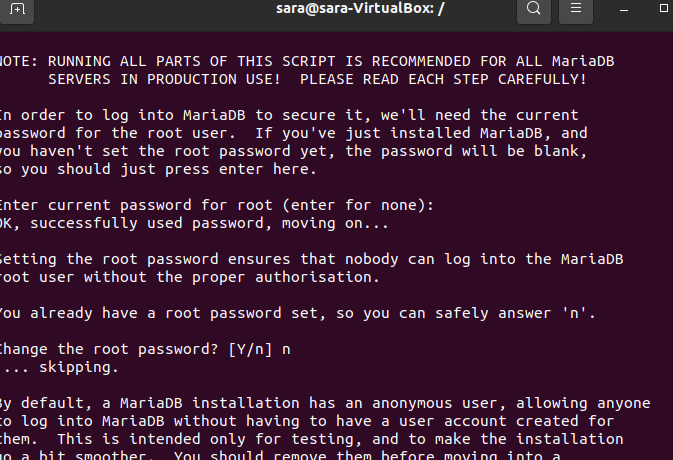
Para su instalación en Ubuntu se procede a actualizar con apt, que incluirá herramientas para la interacción con la BBDD.

## sudo apt update

**sudo apt install** mariadb >server (incluye herramientas junto a la instalación)

**sudo mysql**\_secure\_installation -->restringe el acceso al servidor y eliminar cuentas no utilizadas.

Resultados de la instalación en consola tras ejecutar el comando de seguridad:



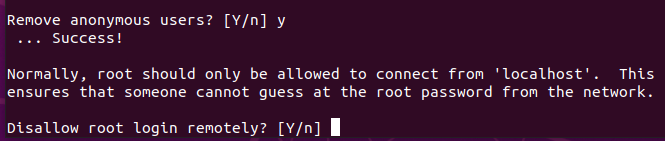
Tras dicha ejecución iniciamos la configuración de seguridad, y nos pedirá que introduzcamos el password actual de root.

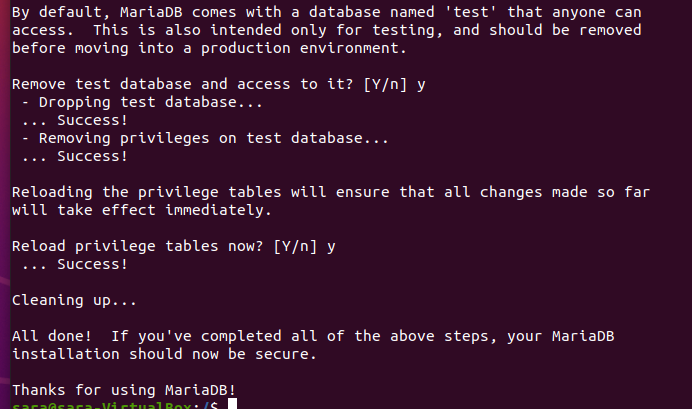
Este es el conjunto de preguntas que solicita mariadb:

* Enter current password for root (enter for none): Pulsa enter
* Set root password? [Y/n]: Y
* New password: Enter password
* Re-enter new password: Repeat password
* Remove anonymous users? [Y/n]: Y
* Disallow root login remotely? [Y/n]: Y
* Remove test database and access to it? [Y/n]: Y
* Reload privilege tables now? [Y/n]: Y

Pulsamos "none" para establecer un password ya que no lo hemos configurado aún.

Por último pregunta si deseamos eliminar cuentas no utilizadas, a lo que respondemos "Y".





A continuación nos pregunta si deseamos deshabilitar las credenciales de acceso remoto con root, si deseamos eliminar las bases de datos de prueba, a lo que aceptamos, y finalizamos el proceso de configuración de seguridad al que se aplicarán los cambios.

Existe un paso opcional para establecer una contraseña a root o usuario administrativo, alternativo al que MariaDB trae por defecto, que consistiría en autenticarse mediante un complemento llamado **"unix\_socket"** y aunque es más seguro nos puede complicar el uso con programas externos como phpMyAdmin.

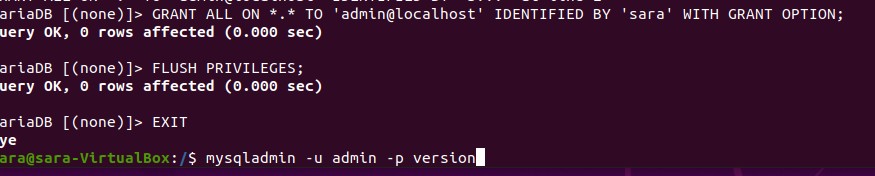
En este caso, los mantenedores de paquetes aconsejan crear una cuenta administrativa con autenticación de contraseña de forma independiente, que pueda manejar dichas actualizaciones, con credenciales de root.

Por tanto creamos una cuenta **admin** con las siguientes órdenes en el Terminal:

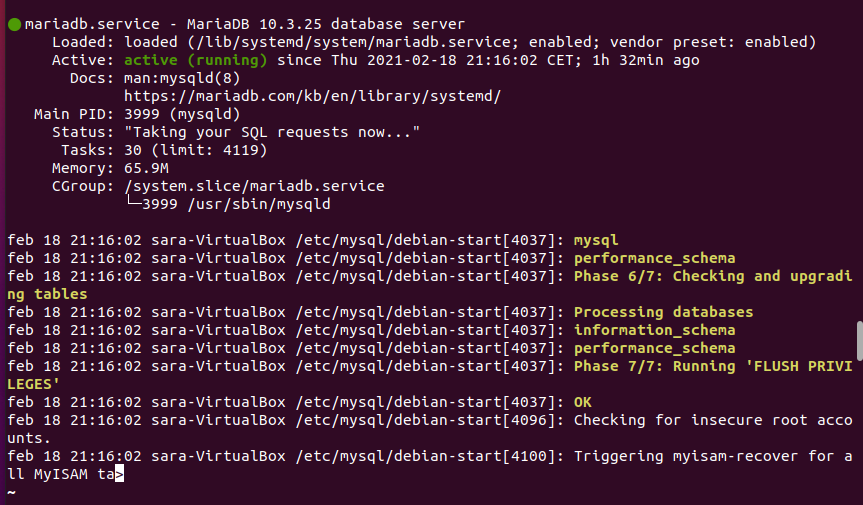
sudo mariadb

Accedemos a la bbdd y creamos el usuario 'admin' con su contraseña en la siguiente instrucción, y después "vaciamos" los privilegios, es decir, los guardamos.

Salimos con "exit"



* El último paso es probar la BBDD, primero comprobamos su estado, que en este caso aparece activo:



Si no fuera el caso, se ejecuta el siguiente comando:

## sudo systemctl start mariadb

Otro comando utilizado es **mysqladmin** ejecutar comandos administrativos. El siguiente comando utiliza el socket unix y devuelve la versión:

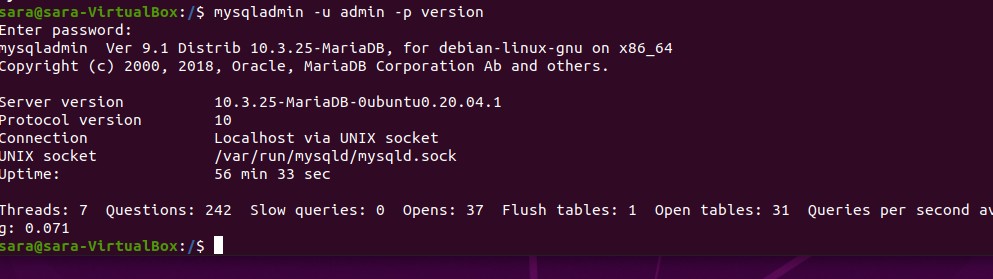
## sudo mysqladmin version



El equivalente con nuestro usuario admin creado sería el siguiente:

## mysqladmin -u admin -p version

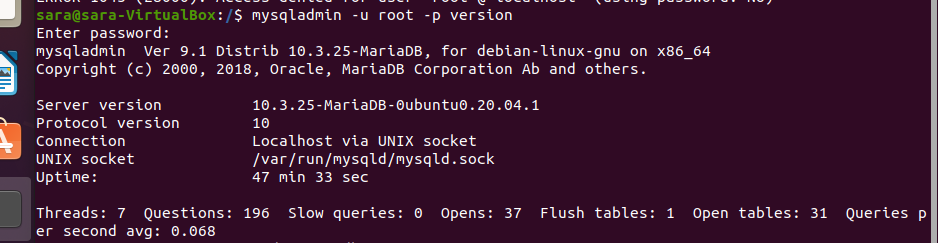
Esto nos demuestra que el usuario está activo y puede autenticarse:



**Nota**: En caso de no tener password la sintaxis sería la siguiente:

## SET PASSWORD FOR 'root'@'localhost' = PASSWORD('New\_Password');

Finamente logramos acceder con **sudo mysql -u root -p :**

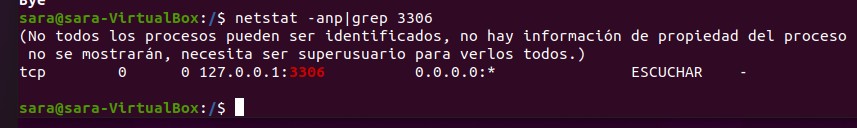


**NOTA**: Hay que tener en cuenta que MariaDb se conecta con un plugin **unix sockets** por defecto cuando el servidor está instalado en un localhost.

Para que esté disponible en un puerto TCP/IP se necesita quitar "skip-networking" y reiniciar el servicio.

Con el siguiente comando podemos comprobar si el puerto de Mariadb por defecto está escuchando:

## [netstat](https://www.enmimaquinafunciona.com/pregunta/143261/mariadb-no-esta-escuchando-en-el-puerto-3306) -anp | grep 3306



**Activamos conexiones entrantes por el firewall:**

sudo ufw allow 3306/tcp

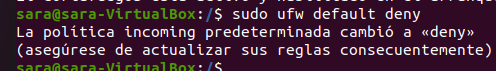
Primero comprobamos si el firewall de ubuntu está activo :

## sudo uwf status.

Activamos firewall.

## -sudo ufw enable

Activamos politica de conexión para denegar por defecto conexiones entrantes y lo reiniciamos.

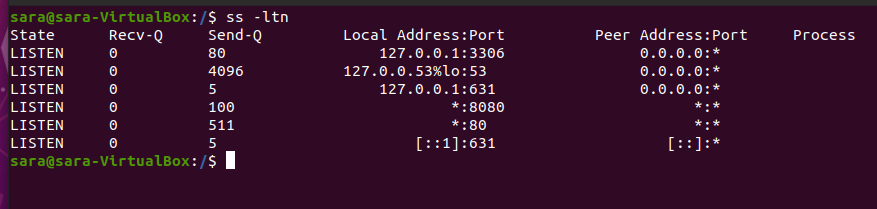


El siguiente paso es activar los puertos de los servicios que nos interesan. Esto es posible a partir del comando siguiente UFW:

sudo ufw allow <numero puerto>.--->abrir puerto sudo ufw deny <numero puerto>. --->

Cerrar puerto

Volvemos a comprobar puertos activos con **ss -lnt:**



Para permitir o denegar conexiones por servicio, como ssh, los comandos son similares:

## -sudo ufw allow/deny <nombre servicio>

**-sudo ufw allow/deny ssh**

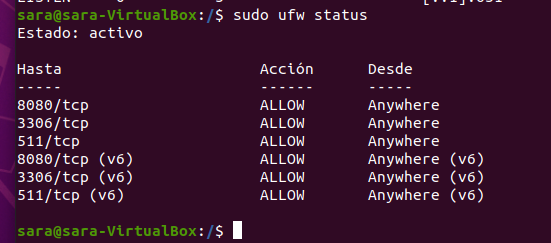
Para saber el nombre exacto del servicio, en ubuntu desde la versión 16.04 se utiliza el siguiente comando:

## -sudo systemctl -r --type service –all

**Nota: Mostrará los servicios activos o inactivos, y algunos no se mostrarán si no están instalados los paquetes asociados para su ejecución.**

Volvemos a ejecutar **ufw status**

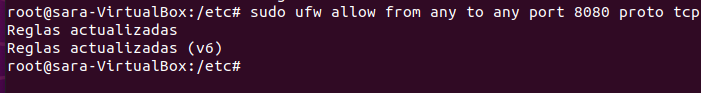
**ufw status vervose** --->mostrará la politica por defecto del firewall.



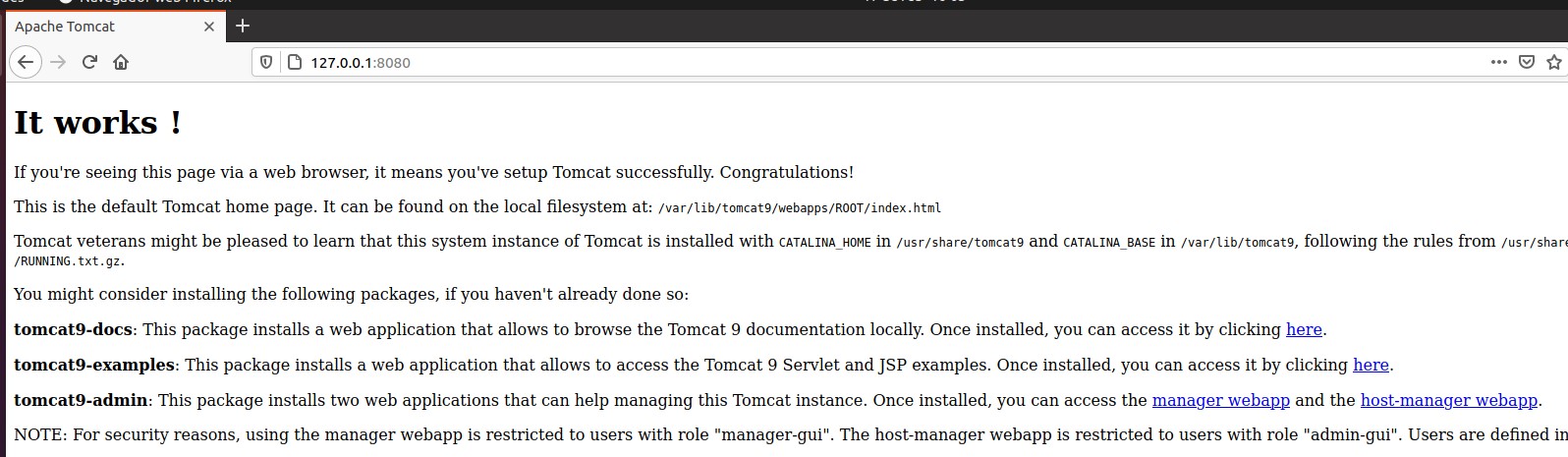
# Instalación de Maria DB.

## Abrir puerto en el Firewall de Linux:

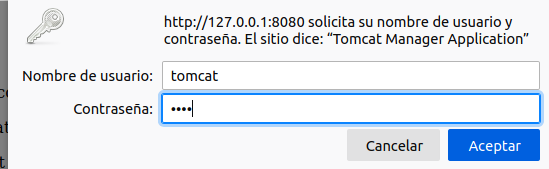
Activamos el trafico TCP entrante desde cualquier fuente al puerto 8080 de tomcat.



## Comprobamos que el servidor funciona accediendo desde el navegador con su ip local:



**Accedemos a tomcat-admin:**



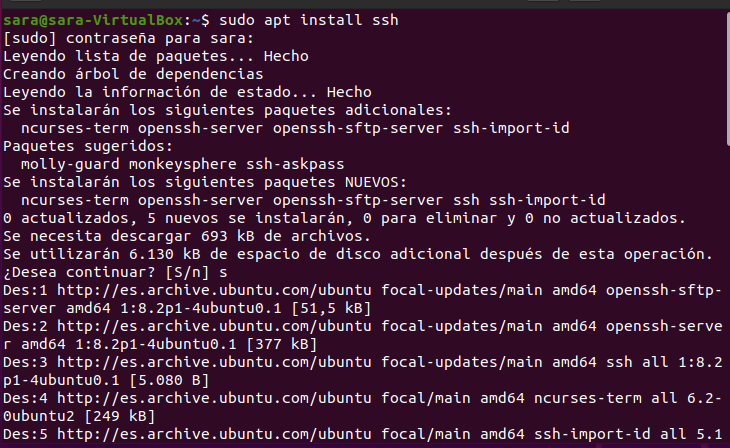
# Instalación de OpenSSH.

SSH son las siglas del protocolo (Secure Shell).

Se utiliza para hacer conexiones remotas de forma segura y especiales en red. En este caso nos sería útil para conectarnos al servidor linux.

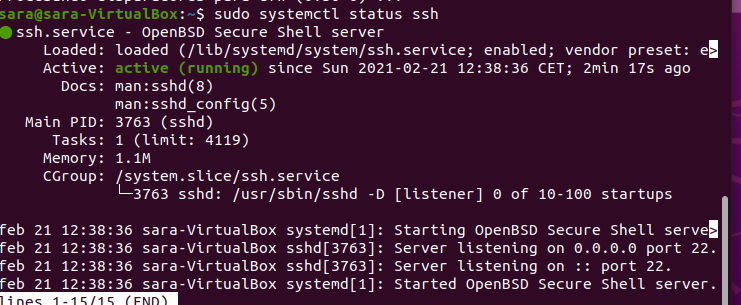
También se utiliza para transferencia de ficheros por SCP o FTP de forma segura. Para instalar o habilitar el servicio utilizamos el siguiente comando:

## -sudo apt install ssh



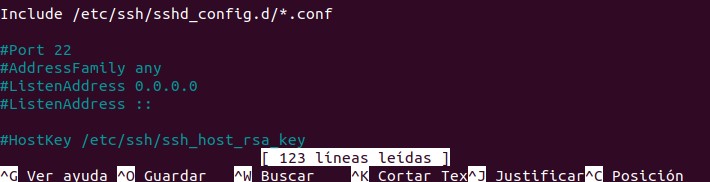
Verificamos que está activado:

## -sudo systemctl status ssh.service.



Modificamos el fichero de configuración para establecer por defecto el **puerto 22.**

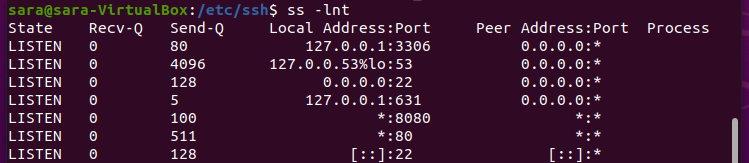
sudo nano /etc/ssh/sshd\_config



Para arrancar, parar o reiniciar el servidor se utiliza el siguiente comando:

**sudo etc/init.d/ssh** restart/start/stop

Comprobamos los puertos activos de nuevo con ss -lnt y vemos que se añade el 22.



Verificar **version** ssh:

ssh -V

Añadimos puerto al firewall del servidor ubuntu.



Ahora accederemos a nuestro servidor Ubuntu desde PowerShell del equipo anfitrión de Windows, con el comando siguiente:

ssh "usuario@ipservidor" -->el usuario corresponde al nombre de usuario y la ip corresponden al de la maquina ubuntu.

ssh -p 22 [**sara@192.168.1.161**](mailto:sara@192.168.1.161)

**NOTA: en VM** configurar como adaptador-puente o red compatible para que funcione correctamente. Nat da problemas a no ser que se haga un mapeo de puertos.

Resultado de conexión correcta:

