****

**Actividad 2. Tarea Individual. Configuración del Servidor**

**PILAR BERMEJO 2 DAW**

[**https://github.com/Pilupbp/despliegueAppWeb**](https://github.com/Pilupbp/despliegueAppWeb)



Queremos preparar nuestro servidor Linux para poder desplegar una aplicación web. Para ello tenemos que verificar que están instalados:

1. Java
2. Apache
3. Tomcat
4. openSSH
5. MariaDB

Así mismo, queremos asegurarnos de que los servidores están bien configurados y son accesibles antes de desplegar la aplicación. Por ello debemos configurar y comprobar que los puertos asociados a Apache, Tomcat y MariaDB están abiertos en el Firewall y son accesibles desde el exterior.

Para toda la actividad se valorará el orden y la claridad de la documentación, así como la facilidad de uso.

Para la entrega, es necesaria la creación y subida a la plataforma de un pequeño documento formal sobre la actividad (portada, explicación, etc.) y un Manual de instalación que describa y permita realizar todos los pasos para la instalación y configuración de las aplicaciones.

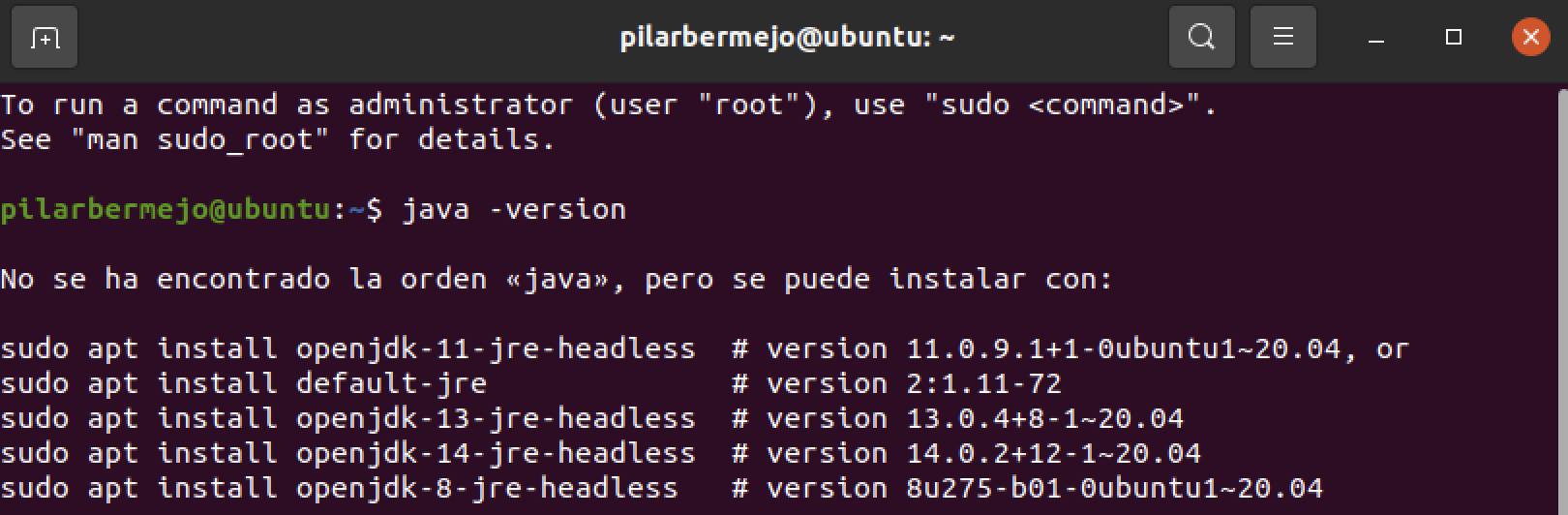
Nótese que más adelante se pedirá que se realicen tareas con un repositorio GIT que contenga la documentación de esta actividad.

Hemos instalado VM Ware creando una máquina virtual de Ubuntu

Lo primero que hacemos es en nuestro Ubuntu abrir el terminal.

Para comprobar si tenemos **JAVA** instalado, realizamos el comando:

***$ Java -version***

**

Como podemos comprobar, no lo tenemos instalado.

Procedemos a instalarlo con el comando que nos indica el terminal.

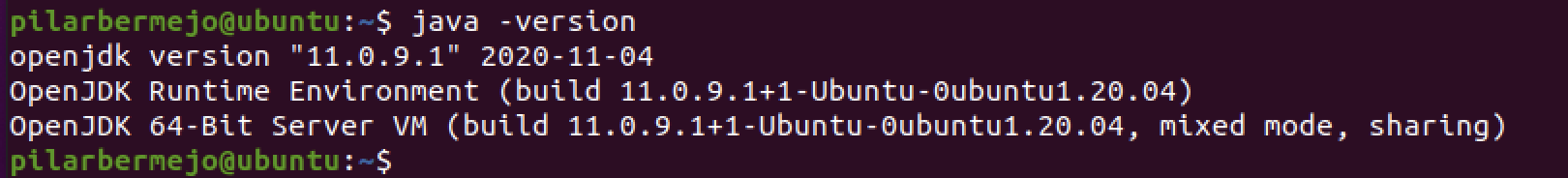
***$ sudo apt install default-jre***



Con el que instalaremos la versión por defecto.

Una vez realizado volvemos a comprobar la instalación con el comando

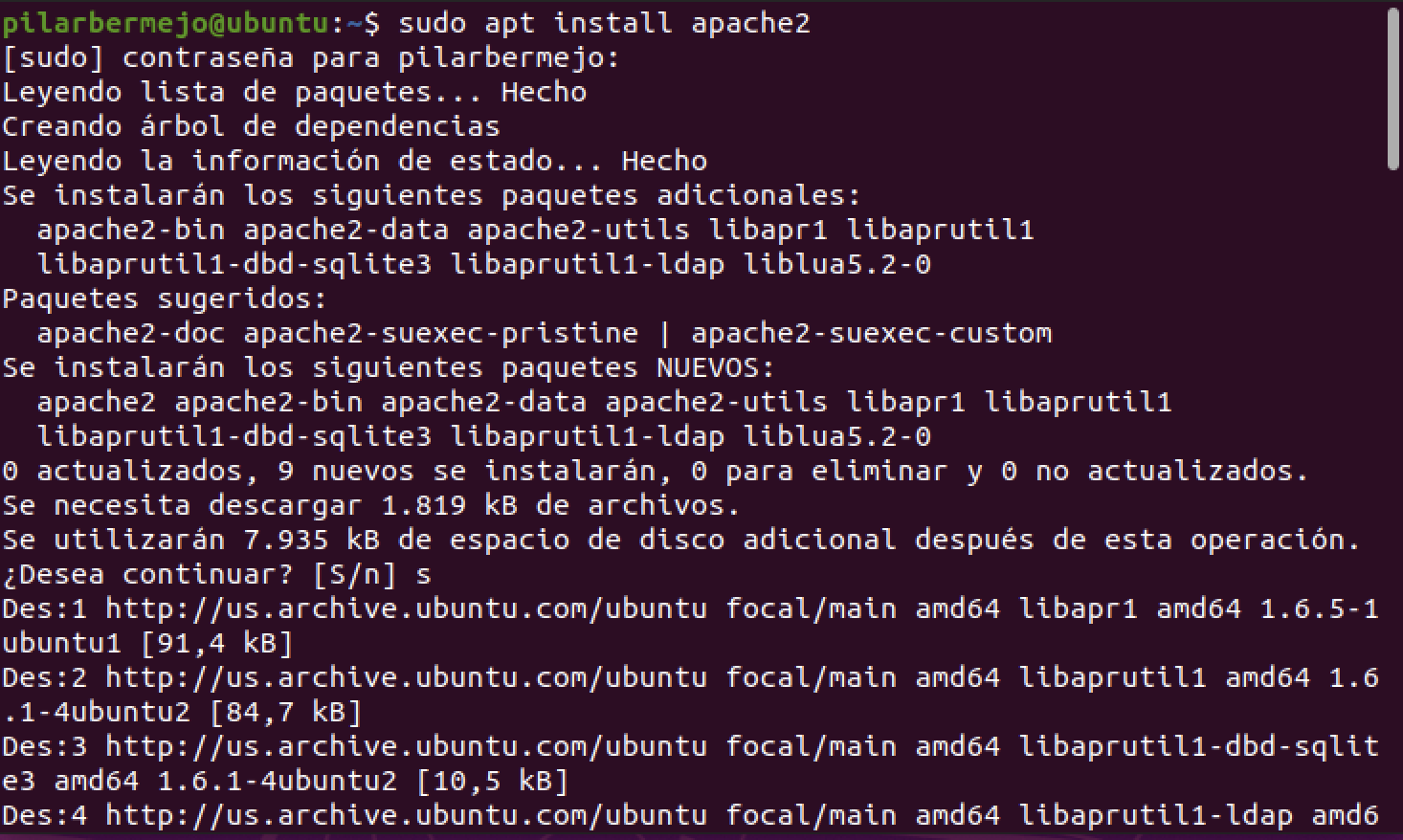
***$ Java -version***



El siguiente paso es **APACHE**, este está disponible en los repositorios de software predeterminados de Ubuntu, lo que permite instalarlo con las herramientas convencionales de administración de paquetes.

Instalamos el paquete apache2 con el comando siguiente:

***$ sudo apt install apache2***

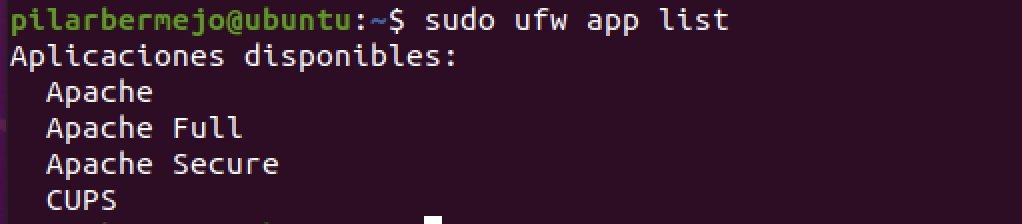


Antes de probar Apache, es necesario modificar los ajustes de firewall para permitir el acceso externo a los puertos web predeterminados.

Durante la instalación, Apache se registra con UFW para proporcionar algunos perfiles de aplicación que pueden utilizarse para habilitar o deshabilitar el acceso a Apache a través del firewall.

Enumeramos los perfiles de aplicación ufw escribiendo lo siguiente:

***$ sudo ufw app list***



Existen tres perfiles disponibles para Apache:

* **Apache**: este perfil abre solo el puerto 80 (tráfico web normal no cifrado)
* **Apache Full**: este perfil abre el puerto 80 (tráfico web normal no cifrado) y el puerto 443 (tráfico TLS/SSL cifrado)
* **Apache Secure**: este perfil abre solo el puerto 443 (tráfico TLS/SSL cifrado)

Vamos a habilitar el perfil más restrictivo que permitirá el tráfico en el puerto 80.

***$ sudo ufw allow ‘Apache’***

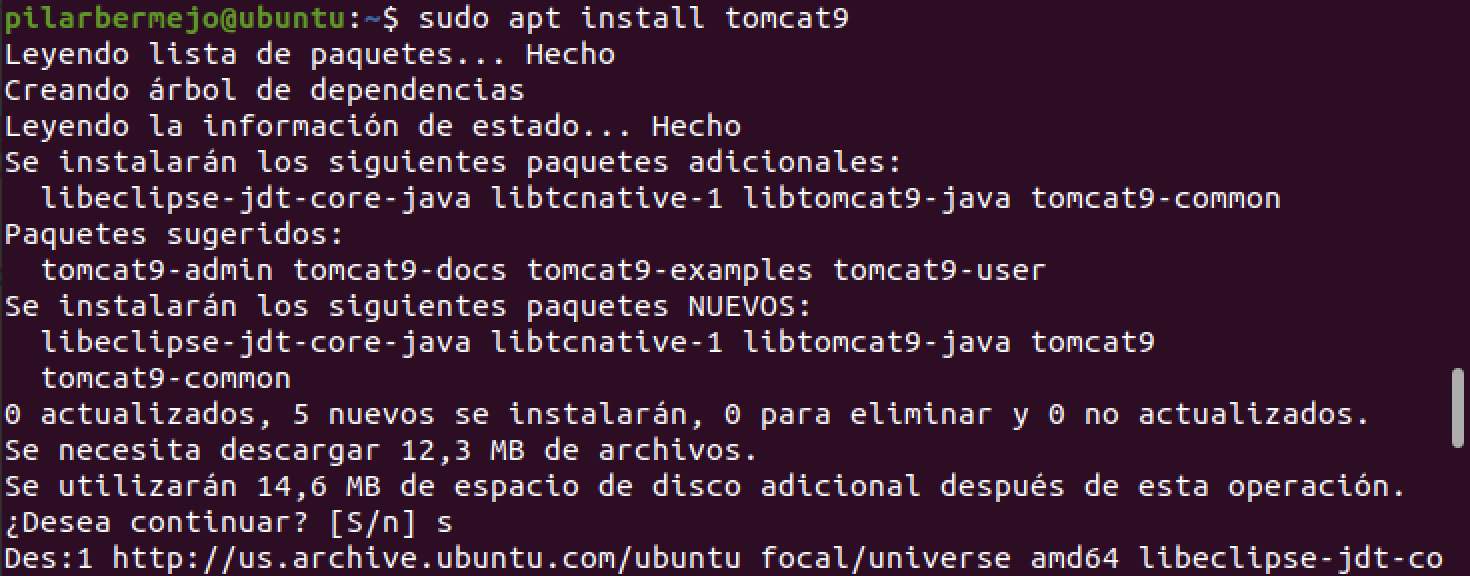
Verificamos con ***$ sudo ufw status***



El siguiente paso es **APACHE TOMCAT** es un servidor web contenedor de Servlets que utilizamos para presentar aplicaciones Java.

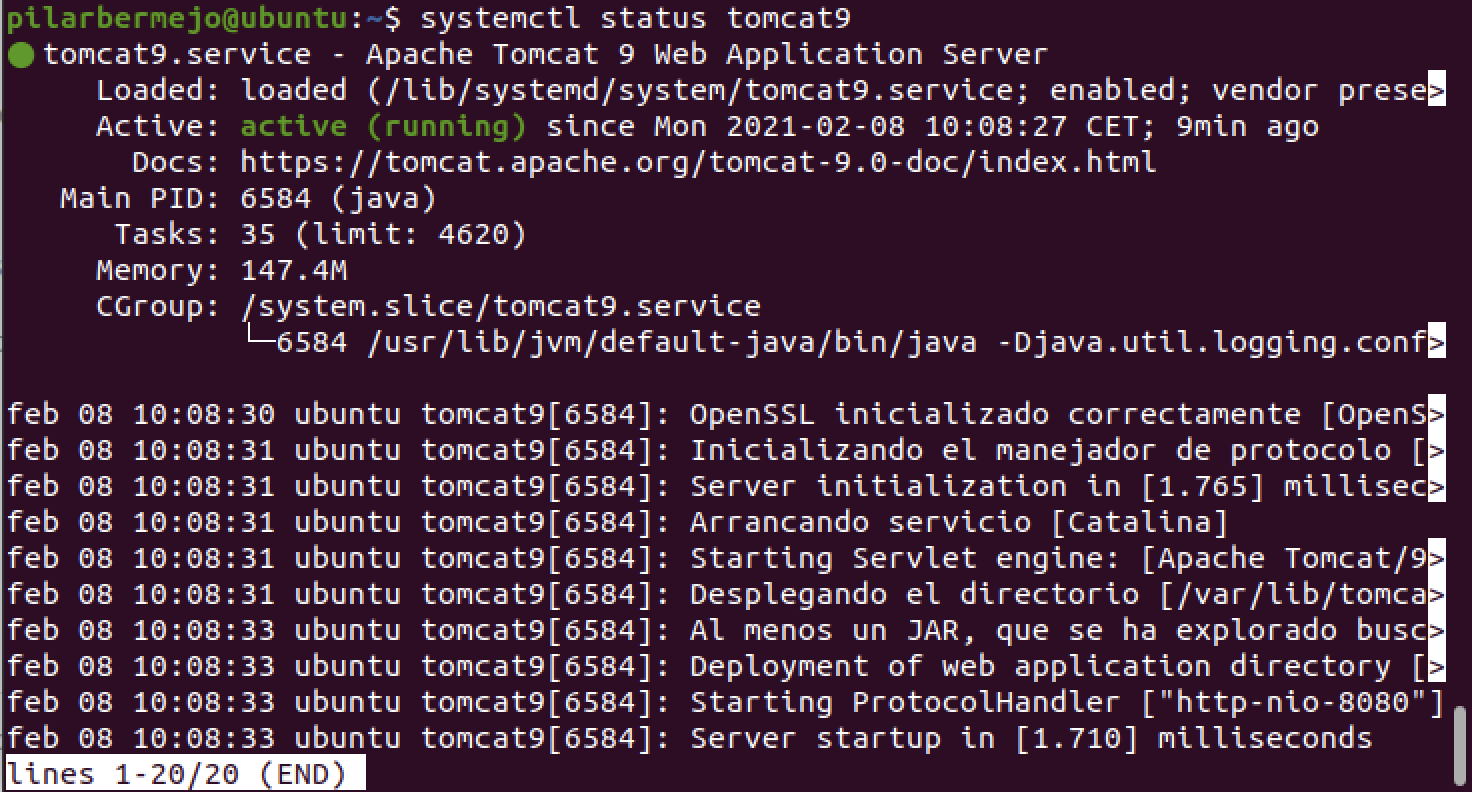
Lo instalamos con el comando:

***$ sudo apt install tomcat9***



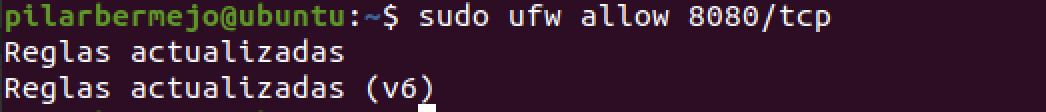
Una vez instalado comprobamos el estado:

***$ systemctl status tomcat9***



Accedemos al Firewall:

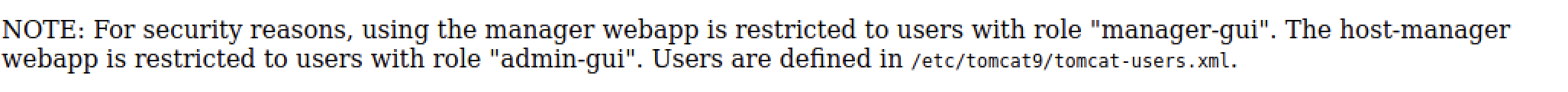
***$ sudo ufw allow 8080/tcp***



Se podría cambiar el Puerto de escucha, editando el archivo server.xml, sustituyendo el valor de port.

***$ sudo nano /etc/tomcat9/server.xml***

Al entrar en la URL localhost:8080, nos sale este mensaje:



Algunas aplicaciones de Tomcat 9, como las aplicaciones administrativas, requieren el acceso autenticado de usuarios con cierto nivel de privilegios o roles.

Por ejemplo, el Gestor de Aplicaciones Web requiere usuarios con rol *manager-gui*, mientras que el Gestor de Máquina Virtual requiere el rol *admin-gui*.

Podemos crear los usuarios que consideremos con contraseña y con uno o ambos roles, en este caso será un solo usuario con ambos roles, para lo que editaremos el archivo *tomcat-users.xml*:

Y lo editamos.

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



Descomentamos los roles para que nos deje acceder al Manager.

Paramos el servicio con ***$ service tomcat9 stop***

Y lo volvemos a levantar con ***$ service tomcat9 start***

Algunas aplicaciones, de nuevo entre ellas las aplicaciones administrativas, restringen en su configuración personal el acceso desde red, por lo que debemos editar su archivo de configuración *context.xml*. Son configuraciones de las aplicaciones en particular, no de Tomcat 9, y algunas aplicaciones tendrán esta característica y otras no.

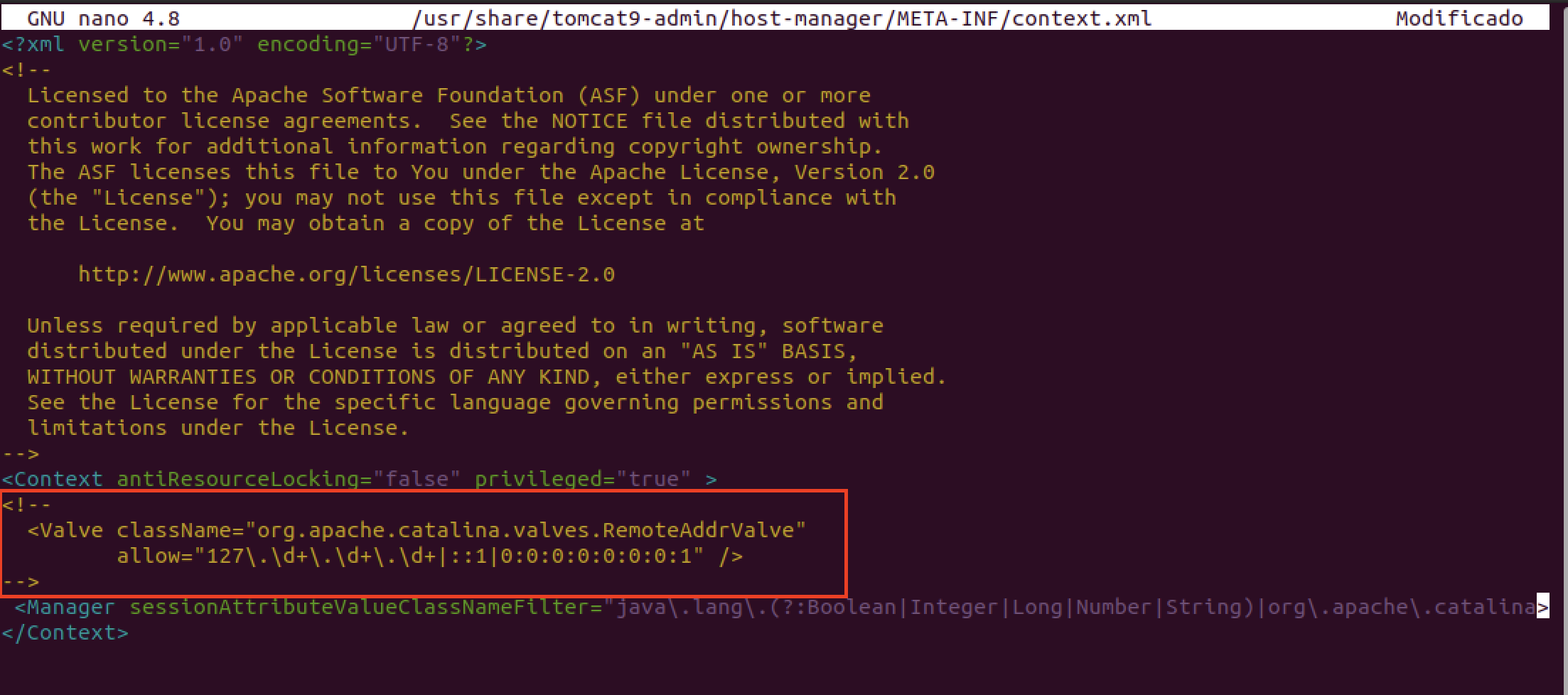
En el caso del Gestor de Aplicaciones Web, o aplicación «*Manager*«, editamos su archivo *context.xml*:

***$ sudo nano /usr/share/tomcat9-admin/manager/META-INF/context.xml***



Para el Gestor de Máquina Virtual, o aplicación «*Host Manager*«, habría que hacer un cambio exactamente igual en su archivo *context.xml*, ubicado en su propia ruta de configuración.

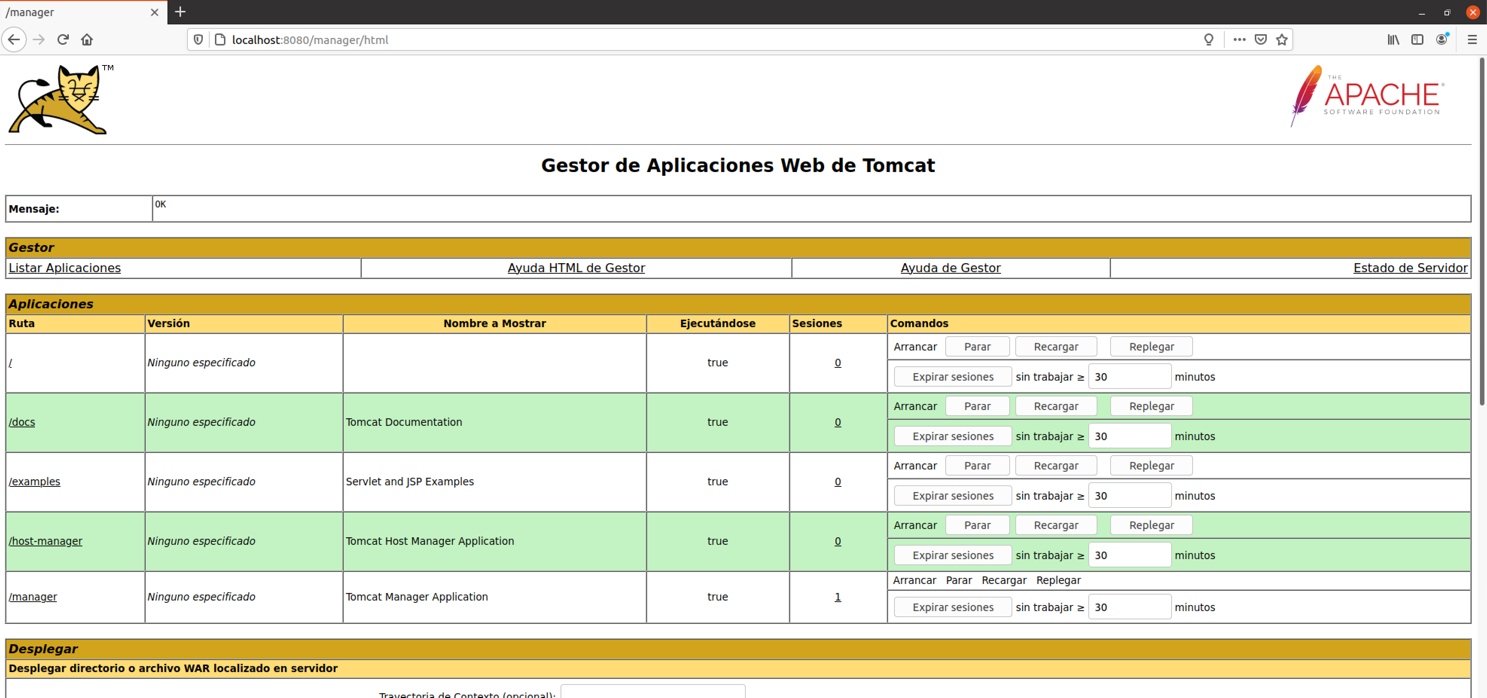
***$ sudo nano /usr/share/tomcat9-admin/host-manager/META-INF/context.xml***



Y reiniciamos Tomcat:

***$ sudo systemctl restart tomcat9***

A continuación usando nuestro usuario y.contraseña configurados accedemos al Manager.

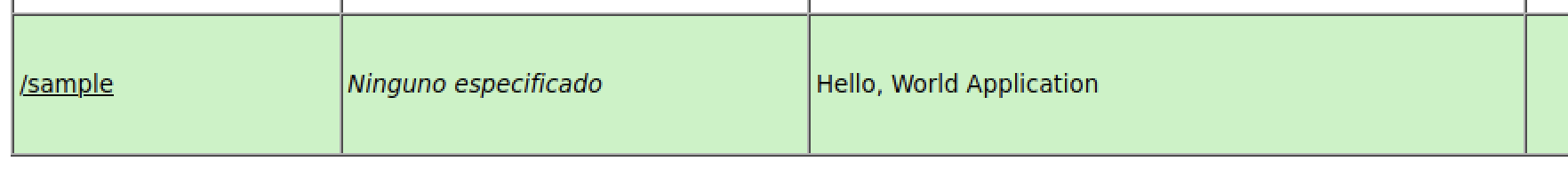


Mediante la sección de Deploy puedo instalar más aplicaciones.

Podemos comprobar que Java funciona a través de la descarga de este ejemplo:

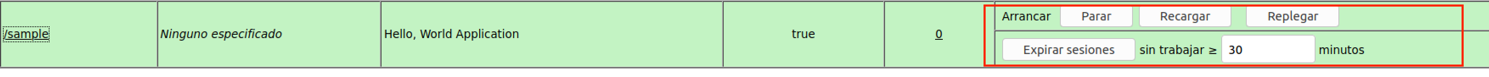
[**http://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/**](http://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/)

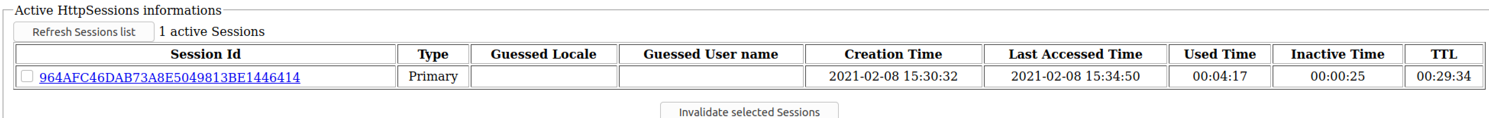
Descargamos el archivo war y lo subimos a través del Deploy.

****

Podemos ver que funciona Java y se puede parar y arrancar para utilizarlo o Undeploy y borrarlo.

****

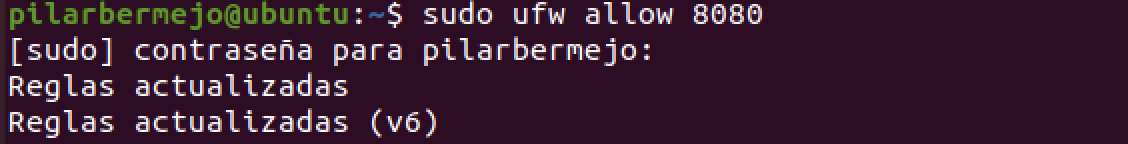
****

Cuando tenemos un servidor de aplicaciones tenemos usuarios que se conectan a ello y crean sesiones.

Estas sesiones nos informan de quién se ha conectado y como.

Hacemos accesible el Firewall con el comando.

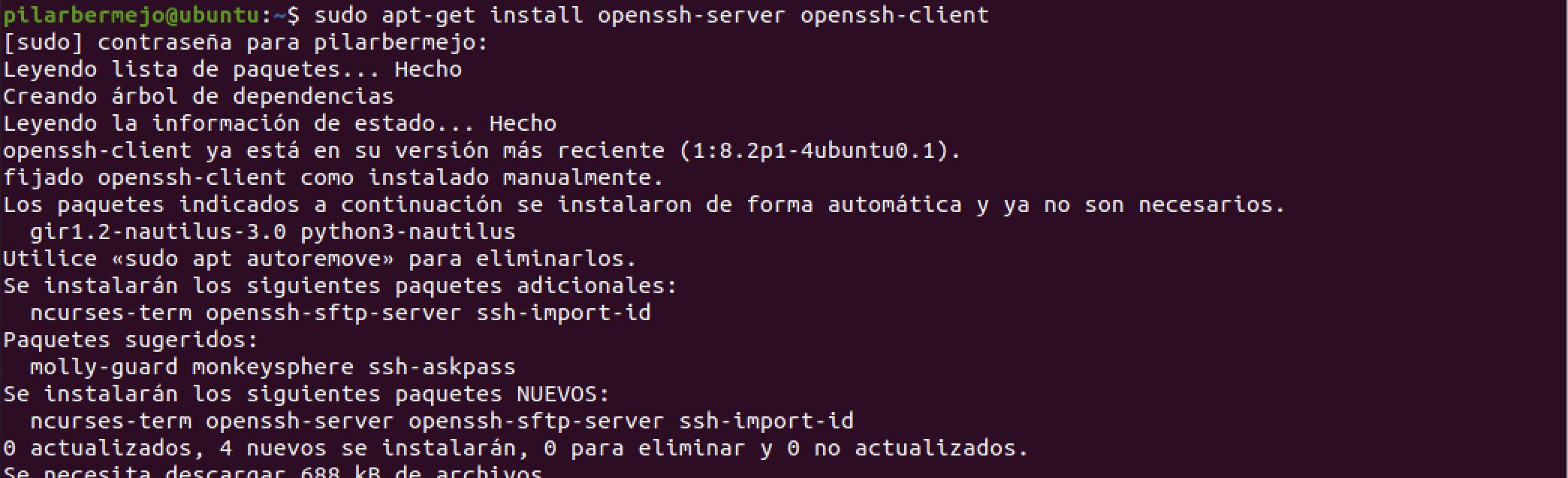
***$ sudo ufw allow 8080***

******

**SSH ([Secure Shell](https://www.sololinux.es/los-mejores-trucos-en-la-consola-ssh/" \t "_blank))** es un [protocolo](https://www.sololinux.es/instalar-configurar-sendmail-linux/) que nos permite acceder a una máquina remota de forma segura. **OpenSSH** es un conjunto de herramientas opensource basadas en el protocolo [SSH](https://www.sololinux.es/acceder-server-sin-password-claves-ssh/).

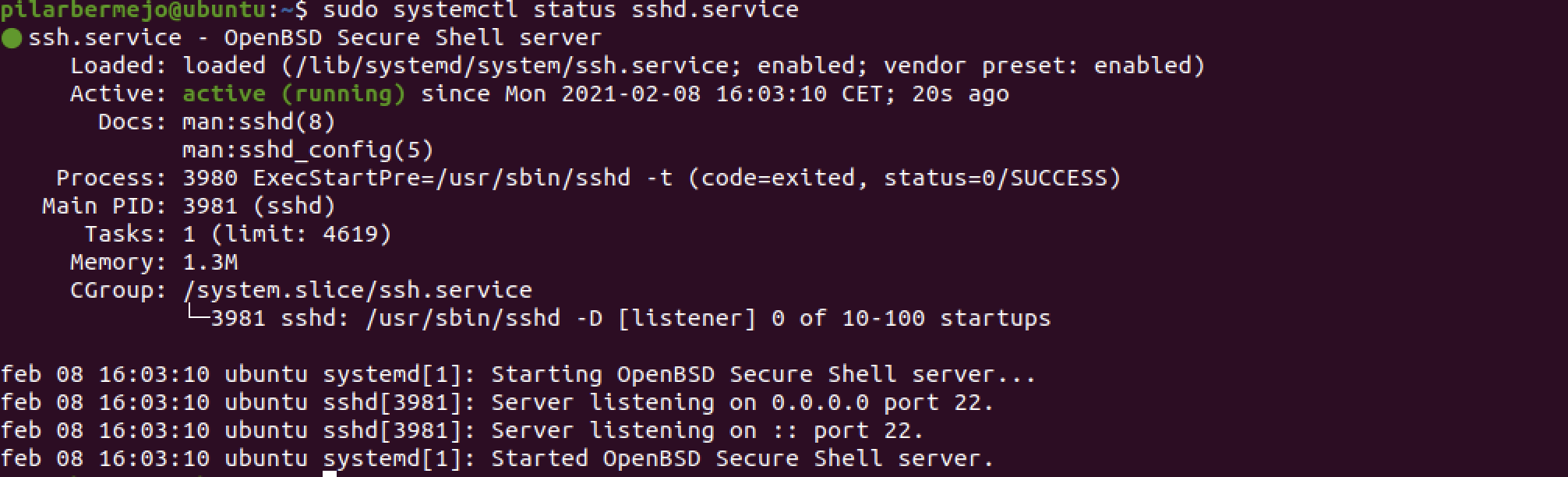
Instalamos el servidor OpenSSH y sus dependencias y la aplicación de cliente.

***$ sudo apt-get install openssh-server openssh-client***



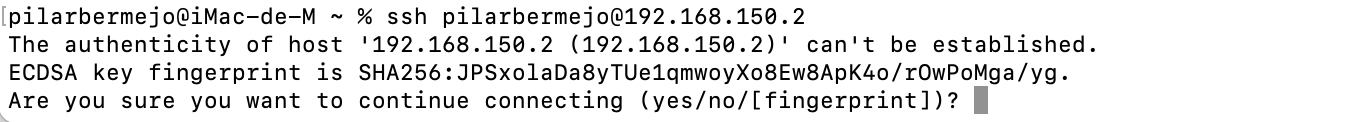
Reiniciamos el servicio: ***$ sudo systemctl restart sshd.service***

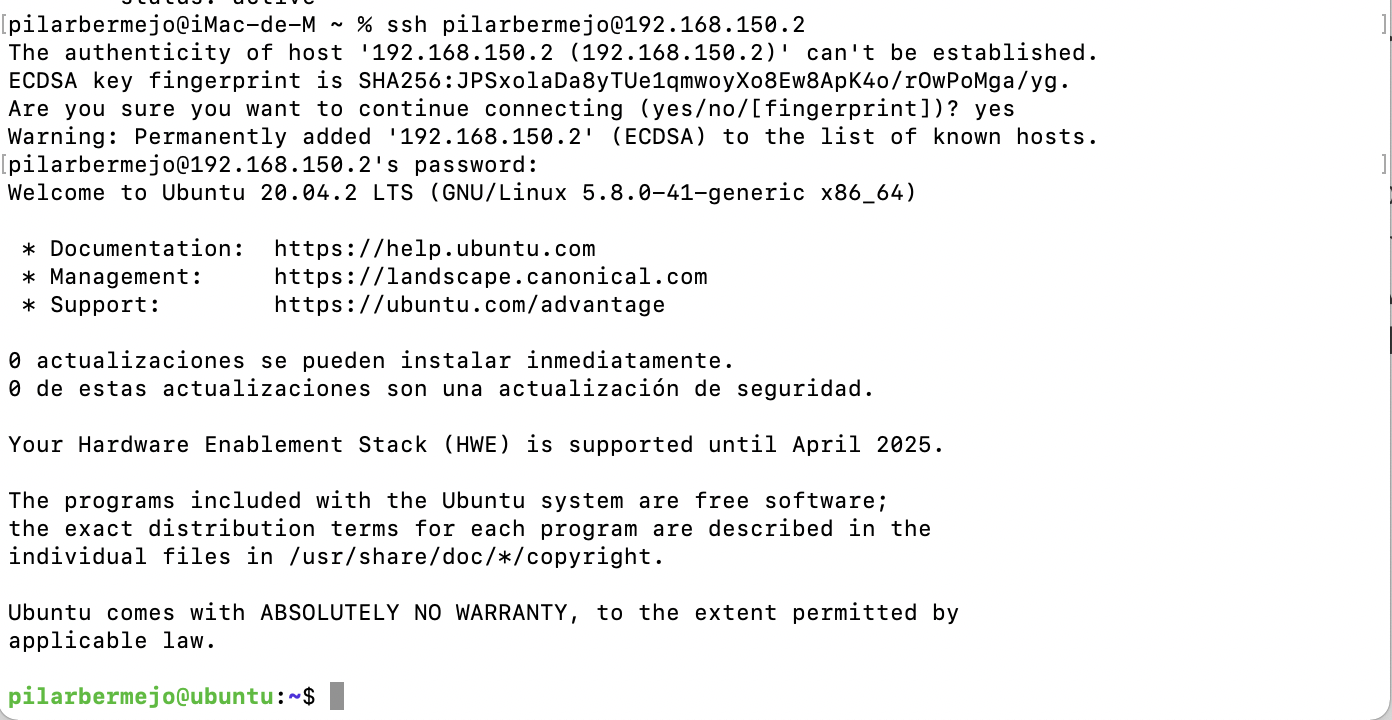
Y comprobamos el estado: ***$ sudo systemctl status sshd.service***



Averiguo la IP del cliente y del servidor (ifconfig) e intento realizar la conexión desde mi PC al servidor Ubuntu mediante el comando:

***$ ssh pilarbermejo@192.168.150.2***





Ya estamos dentro del otro ordenador, el prompt es el del usuario del ordenador remoto y el nombre del equipo es el remoto, ahora mediante los comandos en el terminal podemos actuar con el ordenador remotamente como si estuviéramos delante de él.

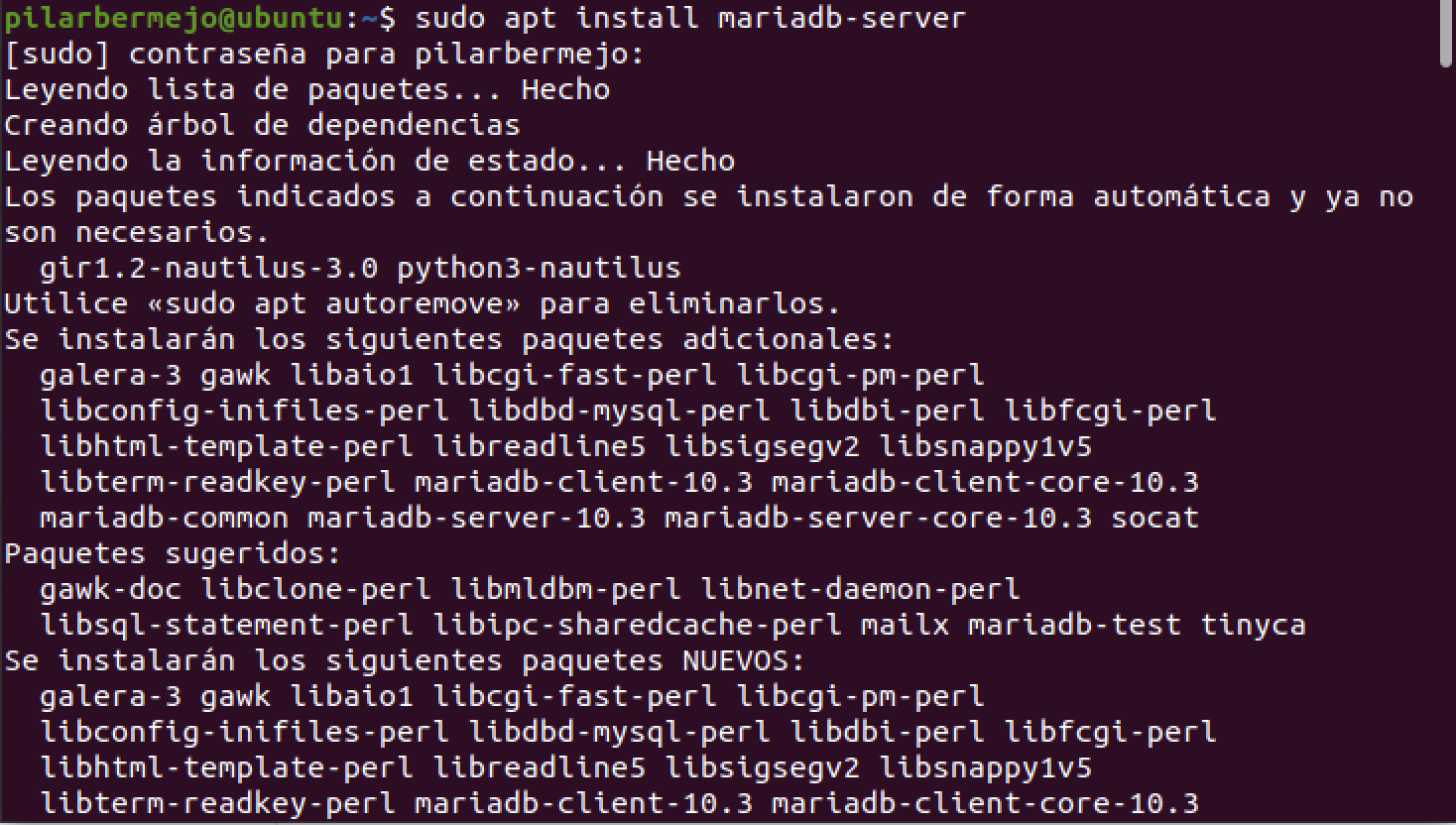
Con el comando ***$ exit*** salimos.



**MARIADB** es un sistema de administración relacional de bases de datos de código abierto, que comúnmente se usa como alternativa para la parte de MySQL de la popular pila LAMP (Linux, Apache, MySQL, PHP/Python/Perl). Se diseñó como un reemplazo a medida de MySQL.

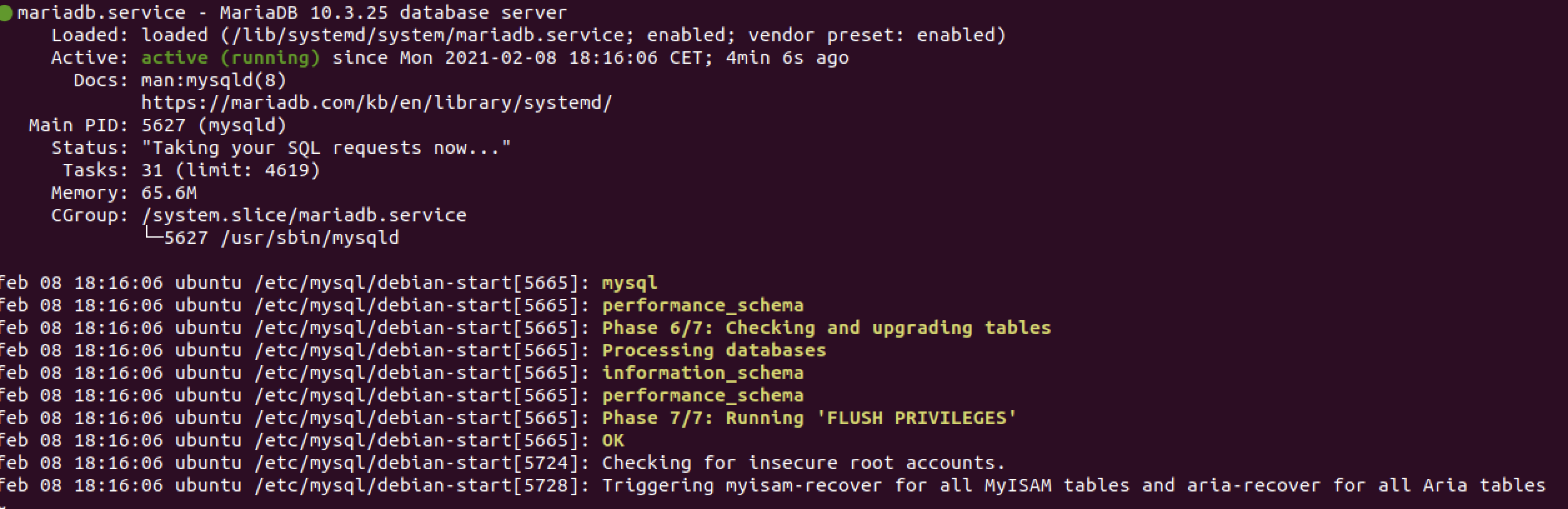
Instalamos el paquete MariaDB server.

***$ sudo apt install maradb-server***



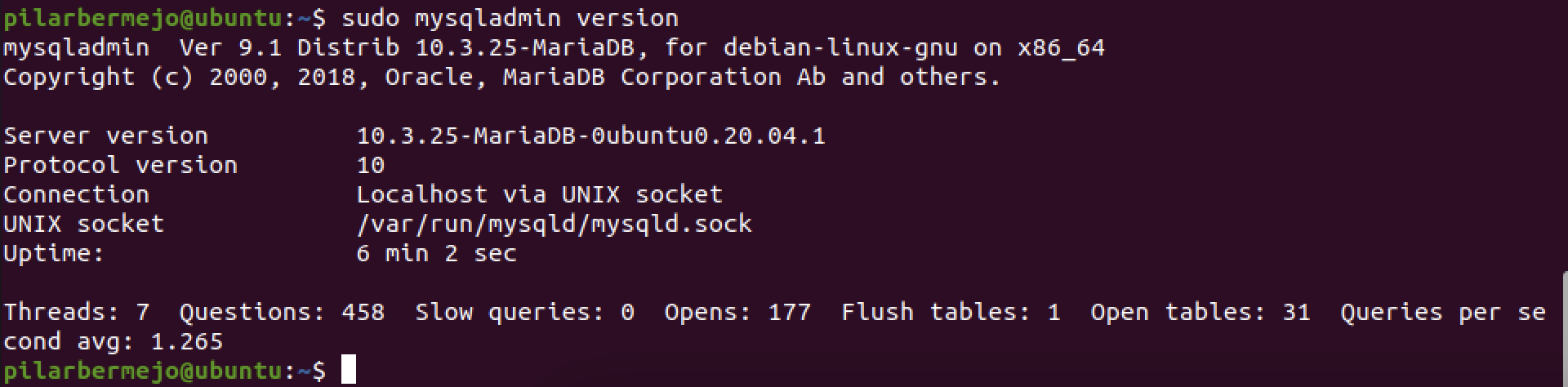
Comprobamos el estado de MariaDB

***$ sudo systemctl status mariadb***



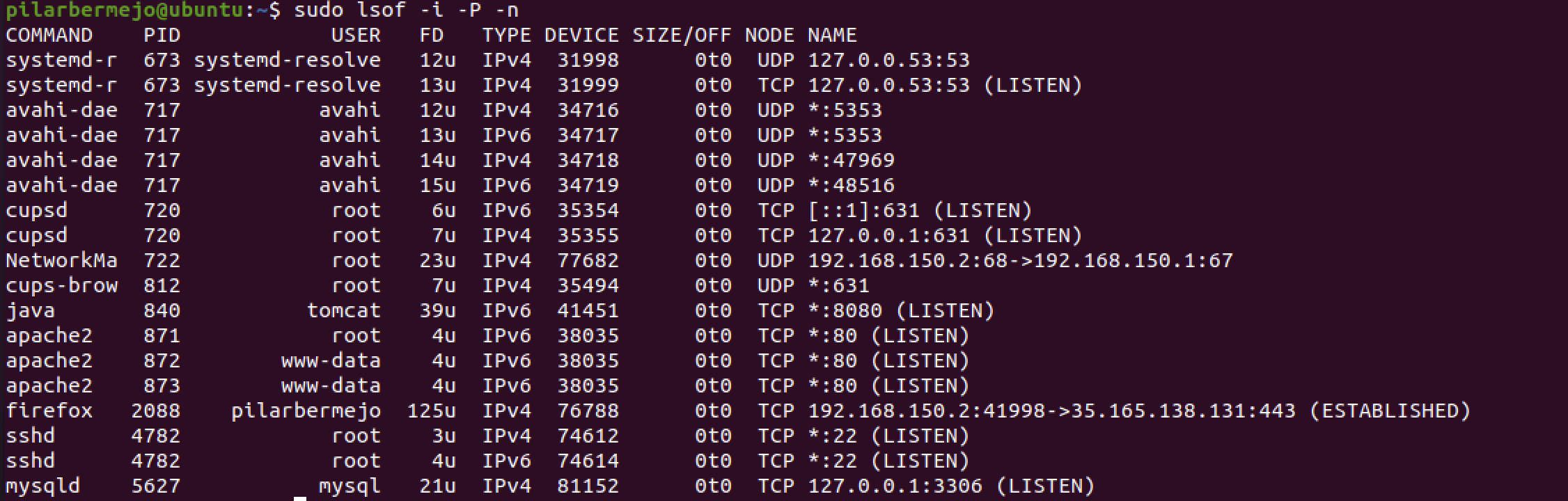
Como comprobación adicional, puede intentamos establecer conexión con la base de datos usando la herramienta mysqladmin.

***$ sudo mysqladmin versión***



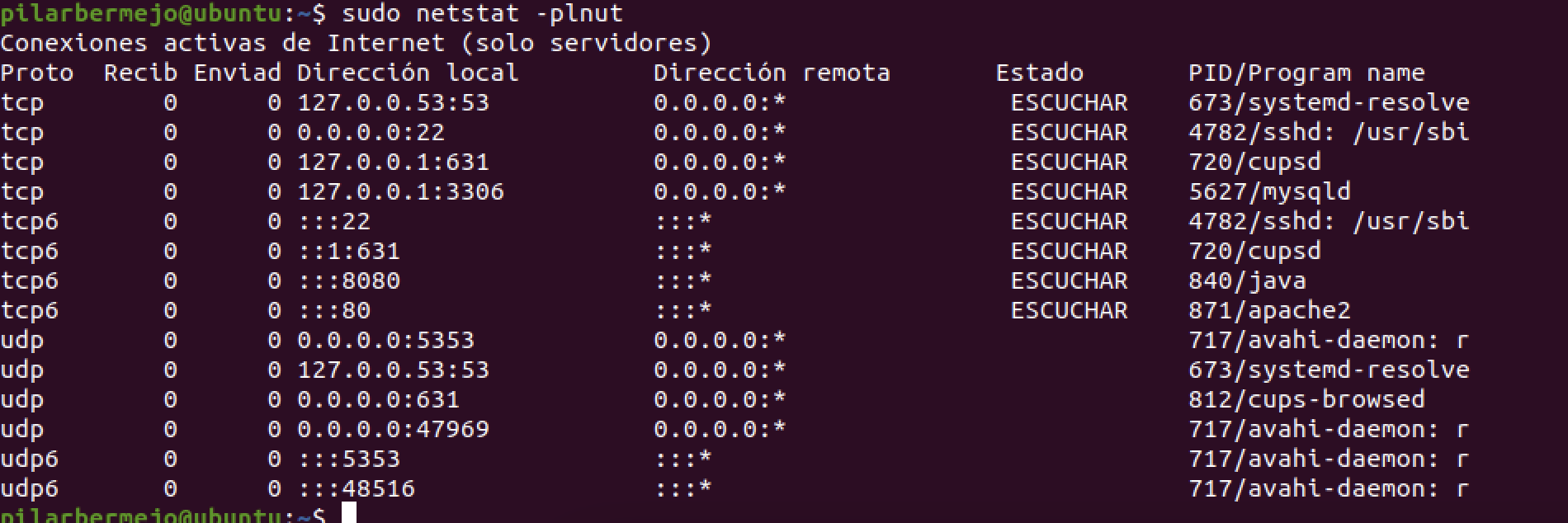
Comprobamos que los puertos están escuchando:

***$ sudo lsof -i -P -n***



También podemos utilizar el comando netstat:

***$ sudo netstat -plnut***

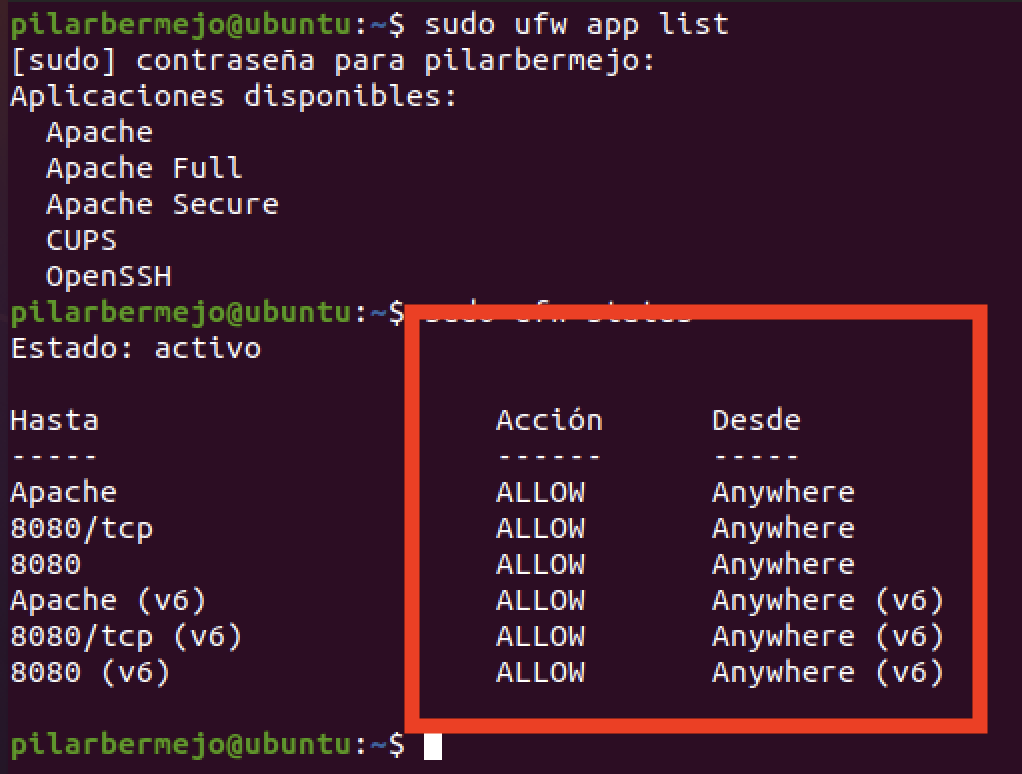


Vamos a comprobar la configuración del **FIREWALL**:

***$ sudo ufw app list***

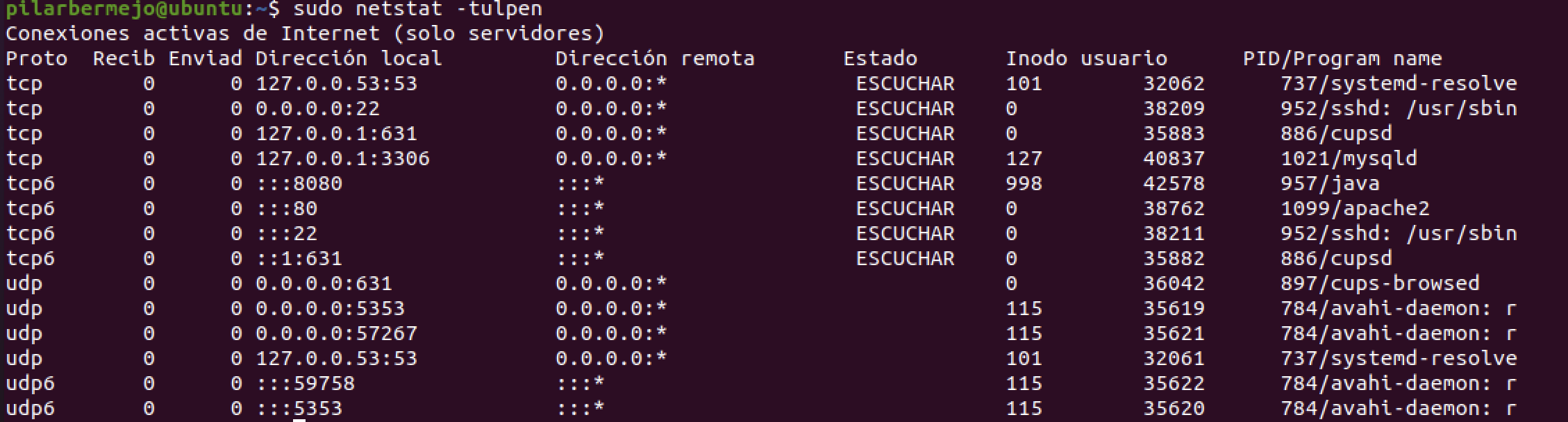
Vemos los perfiles y verificamos el estado.

***$ sudo ufw status***

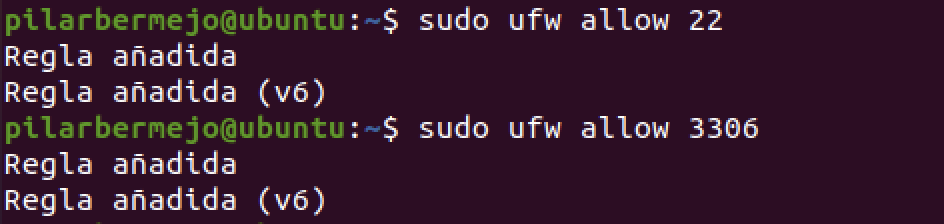


Con UFW podemos abrir y cerrar puertos según necesitemos. Los puertos son interfaces de conexión utilizadas por las aplicaciones para establecer una conexión a un servidor.

Con el comando $ sudo netstat -tulpen podemos ver todos los puertos. Comprobamos si los servicios de la red, los deamons y los programas pertinentes están activos y en estado de escucha al puerto correcto.



Añadimos los puertos 22 y 3306 de MariaDB y lo mostramos en el Firewall:



Los puertos por los que escucha MariaDB 3306, Tomcat 8080 y Apache 80 y el seguro 443 los hemos habilitado en el FIREWALL.



Comprobamos de nuevo que los servicios Apache, Tomcat y MariaDB están corriendo.

