martes, 7 de octubre de 2025

DEFENSA BORN2BEROOT

CREDENCIALES

| PROCESO | USER | PASS |
| --- | --- | --- |
| User | joflorid | 9TC8g2w2Be |
| User | root | frJ9843AC6 |
| Encriptación Disco |  | joflorid@b2bR |
| MariaDB (joflorid\_db) | joflorid-2 |  |
| ftp | joflorid | 9TC8g2w2Be |

ELECCIÓN SISTEMA OPERATIVO

**DEBIAN 13.1.0**. Distribución estable enfocada de uso extendido en la administración de servidores. Alto nivel de seguridad y respaldado por una gran comunidad. Gran repositorio de paquetes. No consume muchos recursos. Instalación y configuración no son sencillas para usuarios principiantes. Libre de intereses comerciales.

**ROCKY**. Basado en Red Hat aunque es reemplazo de la comunidad a CentOS. Mas orientado a entornos profesionales y empresariales. Menor cantidad de paquetes disponibles. Similar estabilidad a Red Hat.

DIFERENCIAS APTITUDE Y APT

Ambas son herramientas de gestión de paquetes en sistemas basados en Debian.

**Aptitude** dispone de una gestión más avanzada de paquetes, ofreciendo más soluciones y alternativas frente a conflictos de dependencias, eliminando paquetes huérfanos de manera automática. Ofrece una interfaz gráfica en terminal lo que ayuda a la navegación por la lista de paquetes disponibles. Orientada a usuarios avanzados que manejan dependencias complejas.

**Apt** es recomendada para la mayoría de los usuarios por su sencillez en la sintaxis. No dispone de herramientas tan avanzadas para la resolución de conflictos con dependencias pero si ofrece soluciones que debe aplicar el usuario.

SELINUX Y APPARMOR

**SELinux** (Security Enhance Linux) es un módulo de seguridad para sistemas GNU/Linux que añade una capa extra de protección mediante la configuración de políticas de acceso obligatorio (MAC). Se complementa con el sistema tradicional Discretionaty Access Control (DAC) donde el dueño de un archivo decide la seguridad sobre este. Con SELinux se pueden definir acciones que son explícitamente permitidas, en las que se incluyen al usuario root, lo cual dota al sistema de una seguridad de nivel superior permitiendo aislar aplicaciones, proteger de accesos no deseados a archivos sensibles e incluso auditorías en tiempo real de accesos. Es considerado una protección robusta contra escalamiento de privilegios y fallos de seguridad en entornos profesionales y empresariales.

**Apparmor** (Application Armor) es otro módulo de seguridad que permite restringir las capacidades y accesos de los programas mediante la definición de perfiles de seguridad. Cada programa tiene asociado un perfil que define que recursos puede usar y que acciones puede ejecutar, añadiendo una capa extra de seguridad en el control de acceso (MAC) que también complementa el modelo DAC. Se basa en las rutas de los archivos a diferencia de SELinux que se basa en etiquetas de seguridad. Más sencillo para el usuario medio. Mantenido principalmente por Canonical y solución por defecto en Ubuntu.

LVM

Logical Volume Manager (LVM) es un gestor de volúmenes lógicos para el núcleo de Linux que permite la creación de “grupos” de discos o particiones que pueden ser ensamblados en uno o varios sistemas de archivos. Estos grupos o volúmenes pueden ser redimensionados o movidos sin necesidad de apagar el sistema.

Permite combinar el espacio de varios discos o particiones en un grupo llamado Volume Group (VG). Dentro de este VG se crean Logical Volumes (VL) que funcionan como las particiones tradicionales pero con la posibilidad de redimensionarse.

Estructura básica:

* Phisical Volume (PV): Discos duros físicos o particiones preparadas para LVM
* Volume Group (VG): Agrupación de uno o varios PV para formar un espacio de almacenamiento lógico.
* Logical Volume (LG): volúmenes dentro del VG que actuándomelas como particiones virtuales donde se crean los sistemas de archivos.

LINK: [GUIA CONFIGURACION (BONUS)](https://youtu.be/2w-2MX5QrQw)

**Comandos útiles:**

* **pvs** —> phisical volume scan
* **vgs** —> volume group scan
* **lvs** —> logical volume scan
* **lsblk** —> Muestra la información de la distribución del disco

Particiones primarias y lógicas

**Particiones primarias**

* Son las **particiones principales** de un disco bajo el esquema de particionado clásico MBR (Master Boot Record).
* Se pueden crear **hasta 4 particiones primarias** como máximo en un mismo disco.
* Una de ellas puede ser marcada como **partición activa/bootable** (es la que arranca el sistema).
* Se usan normalmente para **sistemas operativos** o para separar zonas críticas (ejemplo: /boot, C: en Windows).

**Particiones extendidas y lógicas**

* Debido a la limitación de 4 primarias, se ideó la **partición extendida**:
  + Solo puede existir **una** partición extendida por disco.
  + Dentro de ella se crean las **particiones lógicas**, que permiten tener más de 4 particiones en total.
* Prácticamente, las lógicas funcionan igual que las primarias en cuanto a almacenar datos o sistemas de archivos.
* Limitaciones:
  + Una partición lógica **no puede ser usada directamente como partición de arranque** en MBR (aunque hay trucos para lograrlo).
  + Se recomienda para almacenamiento adicional, /home, /var, etc.

Configuración adecuada para un servidor Debian en máquina virtual

En una VM no sueles necesitar complicarte demasiado, pero una configuración **sencilla y eficaz** sería:

1. Particiones recomendadas:
   * /boot → 512 MB (primaria, para kernels e initrd).
   * swap → depende de tu RAM (si tienes <4 GB, el doble de RAM; si tienes más, entre 2–4 GB suele ser suficiente).
   * / (raíz) → 15–20 GB mínimo.
   * /home → el resto del espacio (si quieres separar datos de usuarios y configuraciones).
2. Para un **servidor dedicado** (no de escritorio), lo habitual es:
   * / → 20 GB.
   * /var separado (porque logs, bases de datos y paquetes crecen mucho). Ejemplo: 10–20 GB.
   * /srv si vas a servir aplicaciones web o ficheros.
   * /tmp separado (opcional, útil para seguridad).

|  |  |  |
| --- | --- | --- |
| **Característica** | **Primaria** | **Lógica (dentro de extendida)** |
| Cantidad máxima | 4 (en MBR) | Prácticamente ilimitadas |
| Puede ser de arranque | Sí | No (normalmente) |
| Ubicación | Directamente en la tabla MBR | Dentro de la partición extendida |
| Uso recomendado | SO, /boot | /home, /var, /srv, etc. |

SSH

Secure Shell. Protocolo de red que permite el acceso remoto de manera segura (criptografía) a otro equipo o servidor. La conexión es cifrada. Diseñado para proteger la comunicación en redes no seguras como internet.

Para conectarse a la VM a través de SSH:

* **ssh joflorid@localhost -p 2424**
* **logout** —> Desconecta la conexión.
* **exit** —> cierra la sesión ssh.

Utilizamos localhost porque debe haber algún problema con el enrolamiento al usarse máquinas virtuales. Adicionalmente, parece ser que el puerto 4242 que solicita el subject está siendo usado por algún otro servicio del campus por lo que se usa un “Port Forwarding” en la configuración de la VM. Desde la máquina local se sale por el puerto 2424 y a la VM se entra por el puerto 4242. Esto no incumple con los requerimientos de proyecto ya que realmente estamos entrando en la VM por el 4242.

**\*Ver mensajes guardados en slack sobre este tema.**

Comandos útiles:

* **sudo apt install openssh-server** —> Instala el servidor ssh.
* **dpkg -l | grep ssh** —> Comprueba que la instalación ha sido correcta.
* **sudo nano /etc/ssh/sshd\_config** —> Fichero de configuración del servicio ssh.
* **sudo service ssh status** —> Comprueba el estado del servicio ssh.
* **systemctl status ssh** —> Comprueba el estado del servicio ssh.
* **ip a s** —> Muestra la ip de la máquina virtual
* **hostname -I** —> Muestra la ip de la máquina virtual.

GESTIÓN DE USUARIOS

Comandos útiles:

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **sudo adduser <user>** | * Añade un nuevo usuario al sistema. |
| **sudo user add -m <user>** | * Añade un nuevo usuario y crea su directorio /home correspondiente. |
| **sudo chown -R <user>:<user><group> /home/…** | * Ajusta los permisos del directorio para el usuario |
| **sudo passwd <user>** | * Se requerirá al usuario el cambio de contraseña. |
| **sudo deluser <user>** | * Eliminar un usuario. |
| **cut -d: -f1 /etc/passwd** | * Lista todos los usuarios existentes en el sistema. |
| **getent pasad | awk -F: ‘$3>=1000 {print $1}’** | * Muestra solo los usuarios “humanos”, donde el UID >= 1000 |
| **sudo usermod -aG <group> <user>** | * Añadir un usuario a un grupo |
| **getent group <group>** | * Muestra la información del grupo. |
| **sudo gpasswd -d <user> <group>** | * Saca a un usuario de un grupo. |
| **sudo usermod -l <new\_user> -d /home/<new\_user> -m <old\_user>** | * cambia de nombre, renombra /home/<old\_user> a /home/<new\_user>. y mueve los archivos. |
| **su -** | * Cambia al usuario root |
| **sudo chale -l <user>** | Muestra información sobre la expiración de la contraseña del usuario. |
| **sudo addgroup <group>** | Añade un nuevo grupo. |

SUDO

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **su -** | Cambiar a usuario root |
| **apt install sudo** | Instalación de sudo. |
| **dpkg -l | grep sudo** | Comprobar la instalación. |
| **adduser <user> sudo** | Añade el usuario <user> al grupo sudo |
| **usermod -aG sudo <user>** | Añade el usuario <user> al grupo sudo |
| **getent group sudo** | Comprueba qué usuarios pertenecen al grupo sudo |

Editar el fichero **sudo nano /etc/sudoers.d/<filename>**

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **Defaults pass\_tries=3** | Solo permite 3 intentos para autenticarse como sudo |
| **Defaults badpass\_message=“custom msg”** | Define un mensaje para la entrada incorrecta de la contraseña. |
| **Defaults logfile=“/var/log/sudo/<filename>”** | Establece el directorio donde se guardarán los logs de sudo. ATENCION!! Crear el directorio previamente (sudo mkdir /var/log/sudo/<filename>. |
| **Defaults log\_input,log\_output** | Define lo que se debe guardar en el directorio anterior. |
| **Defaults iolog\_dir=“/var/log/sudo”** | Especifica cual es el directorio para los logs |
| **Defaults requiretty** | Para permitir el modo tty |
| **Defaults secure\_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"** | Directorios utilizables por sudo. |

UFW (Uncomplicated FireWall)

Comandos útiles:

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **sudo ufw enable** | Activa el firewall. Las reglas configuradas empiezan a aplicarse. |
| **sudo ufw disable** | Desactiva el firewall. Todas las reglas dejan de aplicarse. |
| **sudo ufw status** | Muestra el estado actual del firewall y las reglas activas. |
| **sudo ufw status verbose** | Muestra el estado y las reglas con más detalles. |
| **sudo ufw allow <servicio/puerto>** | Permite el tráfico al puerto o servicio especificado (ejemplo: sudo ufw allow 22 para SSH). |
| **sudo ufw deny <servicio/puerto>** | Bloquea el tráfico al puerto o servicio especificado. |
| **sudo ufw delete allow <puerto>** | Elimina una regla existente que permite el acceso al puerto indicado. |
| **sudo ufw delete deny <puerto>** | Elimina una regla de denegación para un puerto. |
| **sudo ufw reset** | Restablece la configuración de UFW a valores predeterminados y elimina todas las reglas. |
| **sudo ufw reload** | Recarga las reglas de UFW (útil si modificaste archivos de configuración directamente). |
| **sudo ufw show raw** | Muestra las reglas en formato iptables. |
| **sudo ufw app list** | Muestra los perfiles de aplicaciones disponibles en UFW (servicios comunes con reglas predefinidas). |
| **sudo ufw allow from <IP>** | Permite el tráfico desde una IP específica. Ejemplo: sudo ufw allow from 192.168.1.100 |
| **sudo ufw allow <puerto>/udp** | Permite el tráfico UDP en el puerto especificado. Ejemplo: sudo ufw allow 53/udp |
| **sudo ufw allow <puerto>/tcp** | Permite el tráfico TCP en el puerto especificado. Ejemplo: sudo ufw allow 80/tcp |
| **sudo ufw logging on** | Activa el registro (logging) de UFW. |
| **sudo ufw logging off** | Desactiva el registro. |
| **sudo ufw status numbered** | Muestra el estado con numeracion de reglas |
| **sudo ufw delete 2** | Borra la regla num. 2 |

PASSWORD POLICIES

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **sudo nano /etc/login.defs** | Configurar las reglas de contraseñas |
| **sudo apt install libpam-pwquality** | Instala el paquete libpam-pwquality que permite definir las políticas de contraseñas. |
| **sudo nano /etc/pam.d/common-password** | Edita el fichero de configuración. |

En el fichero de configuración **sudo nano /eyc/login.defs**:

|  |  |
| --- | --- |
| **Command** | **Descripción** |
| **PASS\_MAX\_DAYS 30** | La contraseña expira a los 30 días |
| **PASS\_MIN\_DAYS 2** | El número minimo de dias antes de poder modificar la contraseña. |
| **PASS\_WARN\_AGE 7** | Aviso previo 7 dias antes de que la contraseña expire. |

En el fichero de configuración **sudo nano /etc/pam.d/common-password**, a continuation de la linea “*password requisite* [pam\_pwquality.so](http://pam_pwquality.so) *retry=3*” (todo en esta misma linea)

|  |  |
| --- | --- |
| **Command** | **Descripción** |
| **minlen=10** | Longitud mínima de la contraseña |
| **ucredit=-1 dcredit=-1** | Mínimo una mayúscula y un número (dígito). |
| **maxrepeat=3** | Establecer el máximo de repeticiones consecutivas del mismo carácter. |
| **reject\_username** | La contraseña no puede contener el nombre de usuario. |
| **difok=7** | La contraseña debe tener al menos 7 caracteres que no sean parte de la contraseña anterior. |
| **enforce\_for\_root** | Al usuario root también le aplican estas reglas. |

password requisite pam\_pwquality.so retry=3 minlen=10 ucredit=-1 dcredit=-1 maxrepeat=3 reject\_username difok=7 enforce\_for\_root

MONITORING.SH

Crear el scrip con el usuario root para evitar problemas con los permisos de algunos de los comandos.

Arquitectura: uname -a

Nucleos Fisicos: lscpu | grep '^Core(s) per socket:' | awk '{print $4}'

Nucleos virtuales: lscpu | grep '^CPU(s):' | awk '{print $2}'

#Memoria disponible: free -m | grep '^Mem:' | awk '{print $7}'

#Memoria total: free -m | grep '^Mem:' | awk '{print $2}'

MEMORY USAGE: free -m | grep '^Mem:' | awk '{printf "%s/%sMB (%.2f%%)\n", $3, $2, ($3/$2)\*100}'

DISK USAGE: df -m --total | grep 'total' | awk '{printf "%.2f/%.2fGB (%.2f%%)\n", $3/1000, $2/1000, ($2/$3)\*100}'

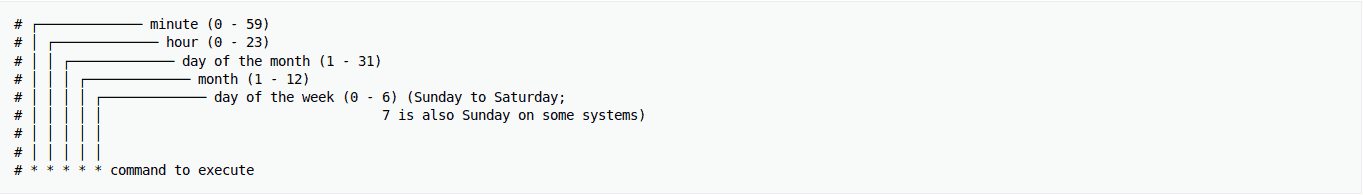
CPU LOAD: top -bn1 | grep load | awk '{printf "%.2f%s\n", $(NF-2), "%"}'

LAST BOOT: who -b | awk '{printf "%s, %s", $3, $4}' LVM: if [ $(lsblk | grep "lvm" | wc -l) -eq 0 ] ; then echo "no" ; else echo "yes"; fi

CONNECTIONS: ss -s | grep "TCP" | awk 'NR == 2 {printf "%d TCP connection/s\n", $3}' USER LOG: w -h | wc -l

NETWORK: $(hostname -I) $(ip link | grep "link/ether" | awk '{printf " - (%s)\n", $2}' SUDO COMMANDS: cat /var/log/sudo/sudo\_logs | grep USER | wc -l

CRON (Programador)



Para añadir un nuevo trabajo en cron:

**sudo crontab -u root -e**

**\*/10 \* \* \* \* bash /<ruta al script>**

Para listar los trabajos de cron:

**sudo crontab -u root -l**

BONUS

LIGHTTPD (Servidor Web) & MariaDB (Base de Datos) & PHP (html)

Sigue los siguientes comandos para instalar y configurar toda la parte web.

|  |  |
| --- | --- |
| **Command** | **Descripción** |
| **sudo apt install lighttpd** | Instala el servidor web |
| **dpkg -l | grep lighttpd** | Comprueba la instalación |
| **sudo ufw allow 80** | Abre el puerto 80 en la VM a través de UFW |
| **sudo apt install mariadb-server** | Instala el servidor mariadb-server |
| **dpkg -l | grep mariadb-server** | Comprueba la instalación |
| **sudo mariadb\_secure\_installation** | Lanza un scrip interactivo de configuración:  *Enter current password for root (enter for none): #Just press Enter (do not confuse database root with system root)*  *Set root password? [Y/n] n*  *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* |
| **sudo mariadb** | Entra en la consola de mariadb.  MariaDB [(none)]> |
| **MariaDB [(none)]> CREATE DATABASE <database-name>;** | Crea una nueva base de datos en MariaDB. |
| **MariaDB [(none)]> GRANT ALL ON <database-name>.\* TO '<username-2>'@'localhost' IDENTIFIED BY '<password-2>' WITH GRANT OPTION;** | Crea un usuario con privilegios sobre la base de datos creada en el paso anterior. |
| **MariaDB [(none)]> FLUSH PRIVILEGES;** | Resetea los privilegios |
| **MariaDB [(none)]> exit** | Salir de la shell de MariaDB |
| **mariadb -u <username-2> -p** | Comprueba si el usuario ha sido correctamente creado. |
| **MariaDB [(none)]> SHOW DATABASES;** | Comprueba si la base de datos ha sido correctamente creada. |
| **MariaDB [(none)]> exit** | Salir de la shell de MariaDB |
| **sudo apt install php-cgi php-mysql** | Instala php |
| **dpkg -l | grep php** | Comprueba la instalación |
| **sudo apt install wget** | Instala wget para la posterior instalación de Wordpress |
| **sudo wget <http://wordpress.org/latest.tar.gz> -P /var/www/html** | Instala WordPress |
| **sudo tar -xzvf /var/www/html/latest.tar.gz** | Extrae el archivo instalado. |
| **sudo rm /var/www/html/latest.tar.gz** | Una vez descomprimido, eliminamos el fichero |
| **sudo cp -r /var/www/html/wordpress/\* /var/www/html** | Copiamos todo lo que hay bajo la carpeta wordpress a un nivel superior |
| **sudo rm -rf /var/www/html/wordpress** | Eliminamos la carpeta wordpress |
| **sudo cp /var/www/html/wp-config-sample.php /var/www/html/wp-config.php** | Creamos un fichero de configuración de wordpress usando el que trae como ejemplo |
| **sudo nano /var/www/html/wp-config.php** | Editamos el fichero reemplazando:  *define( 'DB\_NAME', '<database-name>' );^M*  *define( 'DB\_USER', '<username-2>' );^M*  *define( 'DB\_PASSWORD', '<password-2>' );^M* |
| **sudo lighty-enable-mod fastcgi** | Activamos el modulo fastcgi de Lighttpd |
| **sudo lighty-enable-mod fastcgi-php** |  |
| **sudo service lighttpd force-reload** | Recargamos lighttpd |
| **sudo nano /var/www/html/hello.php** | Crea un fichero php. Añadir el contenido que se desee usando el formato php. |
|  |  |

Una vez mas, existen problemas con el enrolamiento y para poder acceder a la web creada (hello.php) es necesario hacer otro “Port Forwarding” en la configuración de la VM. En este caso, salimos por el puerto 8080 y accedemos al contenido por el 80.

Para acceder a la web creada:

**https://localhost:8080/hello.php**

FTP (servidor FTP - Servicio adicional)

Se opta por el servidor vsftpd al ser sencillo y ligero. Seguir los siguientes pasos:

|  |  |
| --- | --- |
| **Command** | **Descripción** |
| sudo apt install vsftpd | Instala el servidor FTP |
| dpkg -l | grep vsftpd | Comprueba la instalación. |
| sudo ufw allow 21 | Abrimos el puerto 21 (comunicación) en la VM a través de UFW |
| sudo mkdir /home/<username>/ftp | Crea un directorio ftp en la carpeta del usuario |
| sudo mkdir /home/<username>/ftp/files | Crea un subdirectorio |
| sudo chown nobody:nogroup /home/<username>/ftp | Cambia el dueño y grupo de ese directorio a nobody y nogroup respectivamente |
| sudo chmod a-w /home/<username>/ftp | Aplica permisos de escritura al subdirectorio |
| sudo nano /etc/vsftpd.conf | Modificamos el fichero de configuración:  write\_enable=YES (descomentar)  user\_sub\_token=$USER  local\_root=/home/$USER/ftp  chroot\_local\_user=YES (descomentar) |
| sudo touch /etc/vsftpd.userlist | Crea un fichero para que solo los usuarios contenidos en el puedan usar el servicio FTP |
| echo <username> | sudo tee -a /etc/vsftpd.userlist | Añade el usuario al fichero creado anteriormente. |
| Sudo nano /etc/vsftp.userlist | Añadir lo siguiente:  userlist\_enable=YES  userlist\_file=/etc/vsftpd.userlist  userlist\_deny=NO |

Nuevamente hay que configurar un “Port Forwarding” en la VM para poder acceder. Para ello, desde la máquina local salimos por el puerto 2121, pero también tenemos que crear “enrutadores” para las conexiones de datos que creamos en los puertos 40000:40004. Añadidos de igual manera a UFW

Para acceder por terminal con FTP:

**ftp localhost 2121**