

Trabajo Práctico Integrador

Computación Aplicada

Grupo N°8: Alejo Flores, Lautaro Baert y Santiago Manuel
Rodriguez

Profesor : Guillermo Maquieira
Universidad de Palermo 2025

Índice

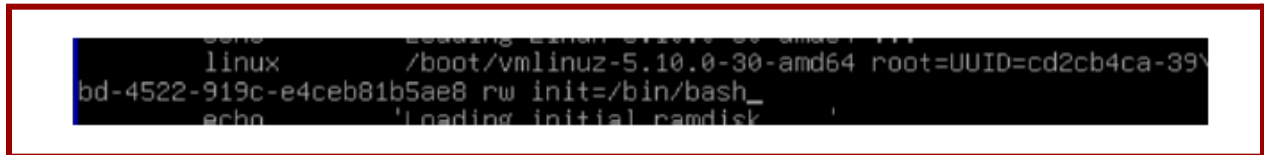
Índice.....	1
1 - Configuración del entorno de trabajo.....	2
Blanqueo de la clave.....	2
Establecer el nombre del hostname.....	3
Servicios.....	3
Sources.list.....	3
SSH.....	4
APACHE.....	8
MariaDB.....	9
Configuración de la red.....	10
Almacenamiento.....	11
Adicionar disco nuevo de 10GB.....	11
Particiones, formateo y montaje.....	12
Backup.....	15
Entregables.....	17

1 - Configuración del entorno de trabajo

Blanqueo de la clave

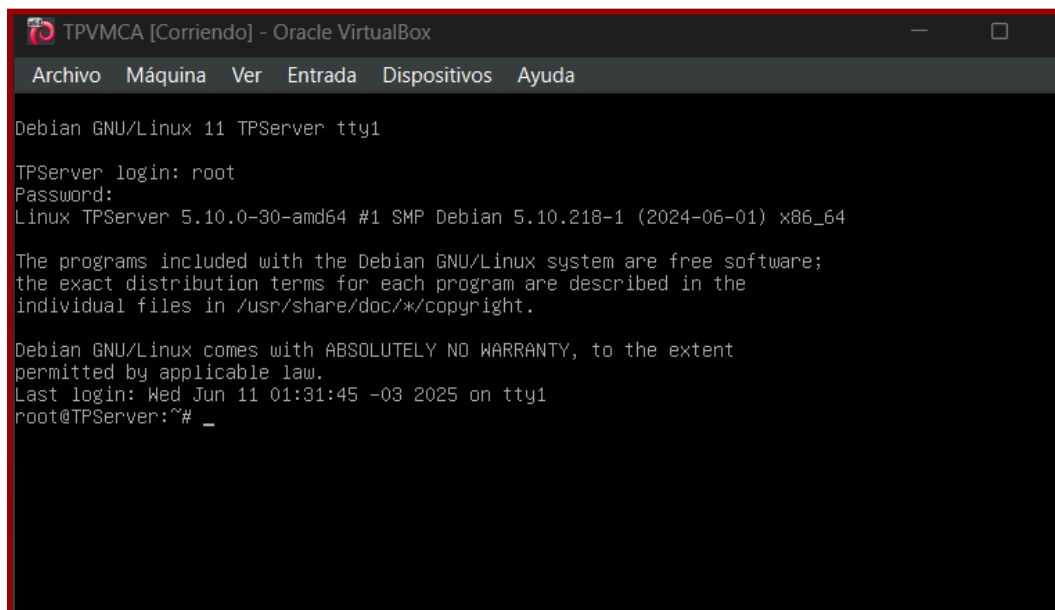
Para comenzar a trabajar dentro de la máquina debemos configurar una clave para poder acceder a ella como usuario root. Para lograr esto, debemos modificar la configuración del gestor de arranque GRUB, mas específicamente, una línea la cual comienza con “linux /boot/vmlinuz[...]

Esto se hace apretando la tecla “e”, que nos permite entrar en modo edición. Una vez en este modo debemos agregar al final de la línea “init=/bin/bash”, y luego apretar “CTRL+X” para iniciar la máquina. De esta forma, entraremos al sistema como usuario root.



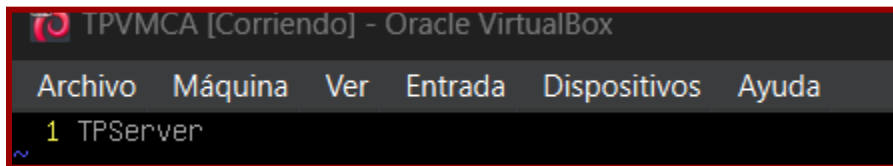
Automáticamente se tiene acceso a un shell de root, donde se cambia la contraseña a “palermo”, como lo indica la consigna, usando el comando: *passwd*.

Luego reiniciamos el sistema, y ya podremos acceder a él con usuario root y la contraseña que establecimos.



Establecer el nombre del *hostname*

La siguiente consigna nos pide que cambiemos el nombre del *hostname* a “TPServer”. Para hacer esto debemos editar el archivo `/etc/hostname` con el editor `vi`.



Con la tecla “i” entramos en modo insertar. Ahí colocamos el nombre que deseemos. Para salir apretamos “ESC”, luego “:wq” para que se guarden nuestros cambios.

De la misma forma debemos editar el archivo `/etc/hosts`. Se cambian los datos de manera tal que quede así:



Ahora podemos reiniciar el sistema con el comando “`init 6`” para comprobar los cambios hechos.



Servicios

Sources.list

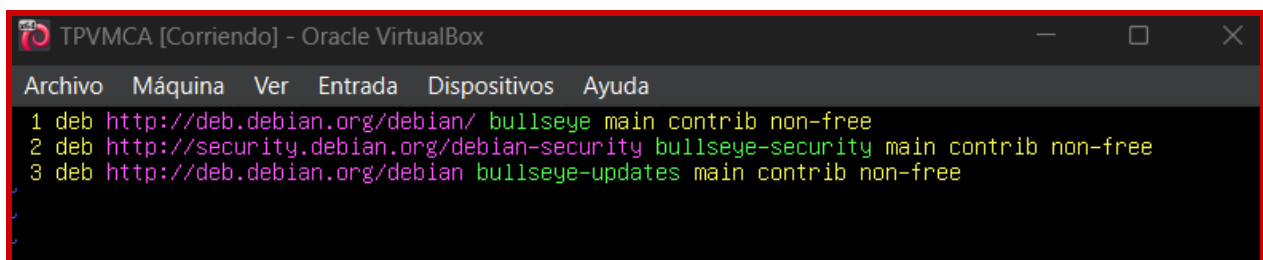
Para este segundo punto debemos descargar y configurar los servicios SSH (Secure Shell), el servidor web Apache y el servidor de base de datos MariaDB.

El primer problema a resolver sobre este punto es que al ejecutar el comando *apt-get update*, para actualizar los índices locales, nos devuelve un error. Para solucionar este problema y poder descargar los servidores que necesitamos debemos modificar la lista de fuentes, ubicada en */etc/apt/sources.list*.

Ejecutamos el comando *vi /etc/apt/sources.list* y editamos los links que encontramos allí por unos que permitan al sistema encontrar efectivamente los paquetes que necesita para descargar las aplicaciones.

```
1 deb http://cdn2-fastly.deb.debian.org/debian/ bullseye main contrib non-free
2 deb http://cdn2-fastly.deb.debian.org/debian/ bullseye-updates main contrib non-free
3 deb http://cdn2-fastly.deb.debian.org/ bullseye/updates main contrib non-free
```

(Antes de ser modificado)



The screenshot shows a terminal window titled "TPVMCA [Corriendo] - Oracle VirtualBox". The terminal displays the contents of the */etc/apt/sources.list* file after modification. The menu bar includes "Archivo", "Máquina", "Ver", "Entrada", "Dispositivos", and "Ayuda". The terminal output is as follows:

```
1 deb http://deb.debian.org/debian/ bullseye main contrib non-free
2 deb http://security.debian.org/debian-security bullseye-security main contrib non-free
3 deb http://deb.debian.org/debian bullseye-updates main contrib non-free
```

(Luego de la modificación)

Ahora podemos proceder a la instalación de los servicios que nos piden. Ejecutamos para esto los siguientes comandos:

- *apt-get install openssh-server*
- *apt-get install apache2.*
- *apt-get install mariadb-server*

SSH

Una vez instalado podemos corroborar que el servicio SSH se encuentra activo con el comando *systemctl status ssh*

```

root@TPServer:~# systemctl status ssh
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2025-06-11 21:14:45 -03; 18min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 489 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
    Main PID: 513 (sshd)
       Tasks: 1 (limit: 2323)
      Memory: 3.5M
         CPU: 117ms
    CGroup: /system.slice/ssh.service
            └─513 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

jun 11 21:14:45 TPServer systemd[1]: Starting OpenBSD Secure Shell server...
jun 11 21:14:45 TPServer sshd[513]: Server listening on 0.0.0.0 port 22.
jun 11 21:14:45 TPServer sshd[513]: Server listening on :: port 22.
jun 11 21:14:45 TPServer systemd[1]: Started OpenBSD Secure Shell server.
root@TPServer:~#

```

Este servidor nos va a servir para establecer una conexión entre nuestra máquina física y la máquina virtual para enviar a través de este los archivos necesarios (proporcionados a través de Blackboard) para cumplir con las consignas.

Fundamentalmente para lograr esta transferencia de documentos mediante SSH, debemos conocer la dirección ip de nuestra máquina virtual y editar el servidor para autorizar el acceso remoto como usuario root.

Para comenzar, ejecutamos el comando *ip a* para conocer la dirección ip. Luego, editamos con *vi* el archivo */etc/ssh/sshd_config* que es el de configuración del servidor y así podemos habilitar el acceso root.

```

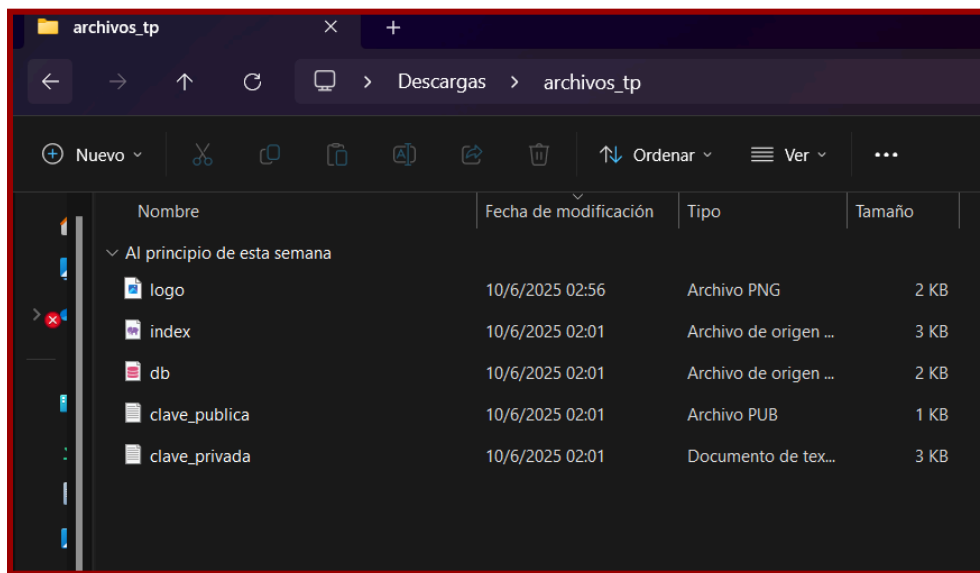
33 #LoginGraceTime 2m
34 PermitRootLogin yes
35 #StrictModes yes
36 #MaxAuthTries 6
root@TPServer:~# /etc/init.d/ssh restart
Restarting ssh (via systemctl): ssh.service.
root@TPServer:~#

```

Editamos la línea "Permit root login", que inicialmente al final dice "prohibit-password", y esto lo cambiamos por "yes".

Es importante después ejecutar el comando */etc/init.d/ssh restart* para releer el archivo de configuración

Luego, lo que hicimos fue descargar los archivos que se encuentran en la plataforma de blackboard, y los almacenamos todos en una carpeta.



Desde la terminal de nuestra computadora (en nuestro caso con sistema operativo Windows) pusimos el comando `scp -r` junto con la ubicación de la carpeta con nuestros archivos en nuestra máquina física (`/downloads/archivos_tp/`), y finalizamos con `root@ip_de_la_maquina_virtual` junto con `“: /”` que indica que se envíe a ese directorio. Luego nos pide la contraseña (la que blanqueamos en el paso 1) y una vez ingresada, comienza la transferencia de archivos.

```
PS C:\Users\Alejo> scp -r downloads/archivos_tp/ root@192.168.0.19:/
root@192.168.0.19's password:
Permission denied, please try again.
root@192.168.0.19's password:
clave_privada.txt          100% 2622   640.1KB/s   00:00
clave_publica.pub         100%  582   284.2KB/s   00:00
db.sql                    100% 1786   436.0KB/s   00:00
index.php                 100% 2325   756.8KB/s   00:00
logo.png                  100% 1719   419.7KB/s   00:00
PS C:\Users\Alejo>
```

```
root@TPServer:/# ls
archivos_tp  bin      etc      initrd.img.old  lib64      media    proc    sbin    tmp      vmlinuz
aux          boot     home     lib             libx32     mnt      root    srv      usr      vmlinuz.old
backup_dir   dev      initrd.img  lib32          lost+found  opt      run      sys      var      www_dir
root@TPServer:/# cd archivos_tp/
root@TPServer:/archivos_tp# ls
archivos_tp  clave_privada.txt  clave_publica.pub  db.sql  index.php  logo.png
root@TPServer:/archivos_tp# _
```

Ya con los archivos necesarios para la configuración de nuestros servidores, podemos continuar con los demás requisitos .

En el caso de SSH se pide que el servidor permita el acceso a usuario root mediante el uso de una clave pública y una clave privada. Ya logramos la parte de permitir el acceso como usuario root, pero ahora queremos generar una autenticación mediante clave pública/privada así no se le

solicite la contraseña cada vez que el cliente requiera conectarse al servicio. Estas claves ya fueron proporcionadas así que para aplicarlas en el servidor hay que hacer lo siguiente.:

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@TPServer:/archivos_tp# cp clave_publica.pub /root/.ssh/
root@TPServer:/archivos_tp# cd /root/.ssh/
root@TPServer:~/.ssh# ls
authorized_keys  clave_publica.pub
root@TPServer:~/.ssh#
```

Copiamos la clave pública en el directorio oculto `.ssh`

Se hace un `cat` del archivo `clave_publica_sonda.pub` y se ejecuta un pipe a un nuevo archivo `authorized_keys`.

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@TPServer:~/.ssh# cat clave_publica.pub > authorized_keys
root@TPServer:~/.ssh# ls
authorized_keys  clave_publica.pub
root@TPServer:~/.ssh# rm clave_publica.pub
root@TPServer:~/.ssh# ls
authorized_keys
root@TPServer:~/.ssh# _
```

Finalmente, desde el cliente, probamos conectarnos, y corroboramos que lo es, ya que no pide la contraseña, simplemente se conecta al servidor.

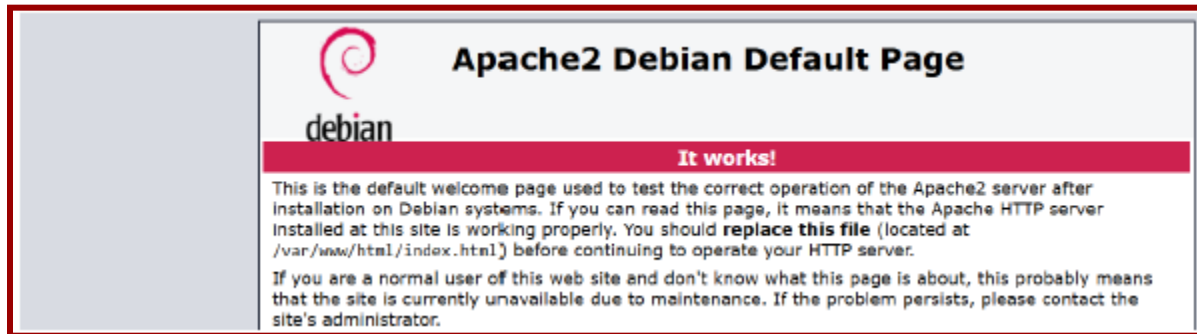
```
PS C:\Users\Alejo> cd downloads
PS C:\Users\Alejo\downloads> cd archivos_tp
PS C:\Users\Alejo\downloads\archivos_tp> ssh -i clave_privada.txt root@192.168.0.19
Linux TPServer 5.10.0-30-amd64 #1 SMP Debian 5.10.218-1 (2024-06-01) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jun 12 00:21:18 2025
root@TPServer:~#
```


APACHE

Una vez instalado APACHE podemos ingresar en nuestro navegador la dirección IP de la máquina virtual y podremos ver una página que nos confirma que el servidor funciona, se ve algo así:



Continuando con la configuración del servidor ejecutamos en la máquina virtual el comando `apt-get install php` y verificamos que se haya instalado una versión superior a 7.3 con el comando `php -v`

```
TPVMCA [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@TPServer:~# php -v
PHP 7.4.33 (cli) (built: Mar 19 2025 19:57:26) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.33, Copyright (c), by Zend Technologies
root@TPServer:~# _
```

Luego debemos instalar los siguientes paquetes adicionales con los comandos:

- `apt-get install libapache2-mod-php -y`
- `apt-get install php-mysql`

A continuación reiniciamos el servidor para que se implementen los cambios con `systemctl restart apache2`.

Continuamos copiando los archivos `index.php` y `logo.png` que importamos previamente en el directorio `/var/www/html/`

```
:~# cp /root/archivos_tp/index.php /var/www/html
:~# cp /root/archivos_tp/logo.png /var/www/html
```

MariaDB

Para configurar este servidor debemos cargar el script sql (*db.sql*) en la base de datos del servidor. Lo logramos de la siguiente forma:

```
root@TPServer:~# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 30
Server version: 10.5.29-MariaDB-0+deb11u1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> source /root/archivos_tp/db.sql_
```

Para verificar que se haya cargado efectivamente lo hacemos de la siguiente forma:

```
root@TPServer:~# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 30
Server version: 10.5.29-MariaDB-0+deb11u1 Debian 11

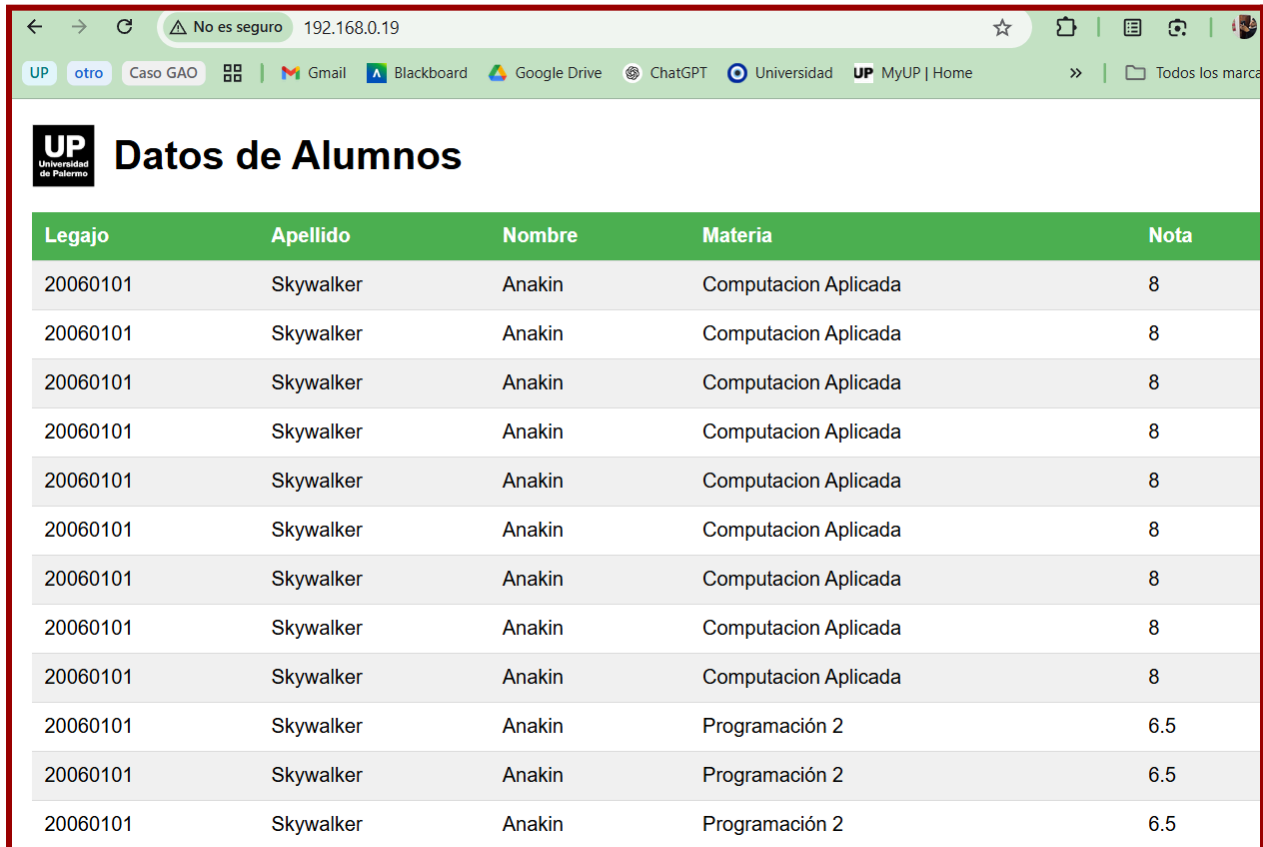
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| ingenieria |
| mysql |
| performance_schema |
+-----+
4 rows in set (0,044 sec)

MariaDB [(none)]>
```

Ahora si volvemos a acceder a la dirección IP de la máquina desde un motor de búsqueda vamos a ver los cambios que realizamos en el servidor.

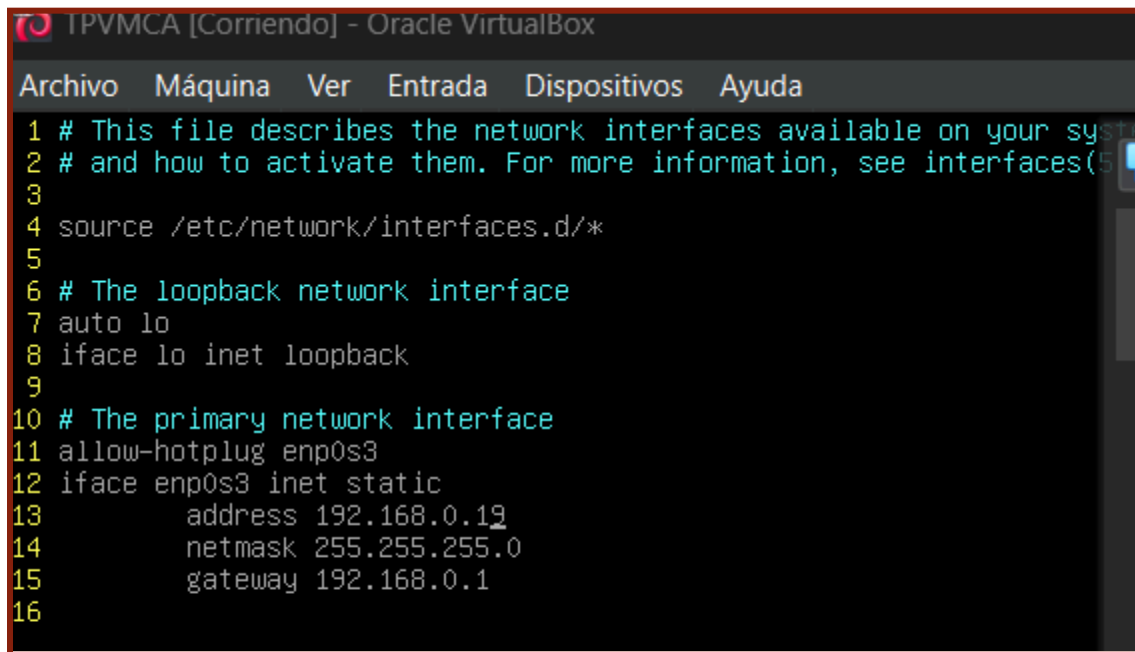


The screenshot shows a web browser window with the address bar displaying '192.168.0.19'. The browser's address bar also shows 'No es seguro' (Not secure). The browser's tabs include 'UP', 'otro', 'Caso GAO', 'Gmail', 'Blackboard', 'Google Drive', 'ChatGPT', 'Universidad', and 'UP MyUP | Home'. The main content area displays the 'UP Universidad de Palermo' logo and the title 'Datos de Alumnos'. Below the title is a table with five columns: 'Legajo', 'Apellido', 'Nombre', 'Materia', and 'Nota'. The table contains 12 rows of student data.

Legajo	Apellido	Nombre	Materia	Nota
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Computacion Aplicada	8
20060101	Skywalker	Anakin	Programación 2	6.5
20060101	Skywalker	Anakin	Programación 2	6.5
20060101	Skywalker	Anakin	Programación 2	6.5

Configuración de la red

Aquí se busca configurar una IP estática para la máquina virtual. Para esto, debemos editar el archivo `/etc/network/interfaces`.



```
TPVMCA [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
1 # This file describes the network interfaces available on your system
2 # and how to activate them. For more information, see interfaces(5)
3
4 source /etc/network/interfaces.d/*
5
6 # The loopback network interface
7 auto lo
8 iface lo inet loopback
9
10 # The primary network interface
11 allow-hotplug enp0s3
12 iface enp0s3 inet static
13     address 192.168.0.19
14     netmask 255.255.255.0
15     gateway 192.168.0.1
16
```

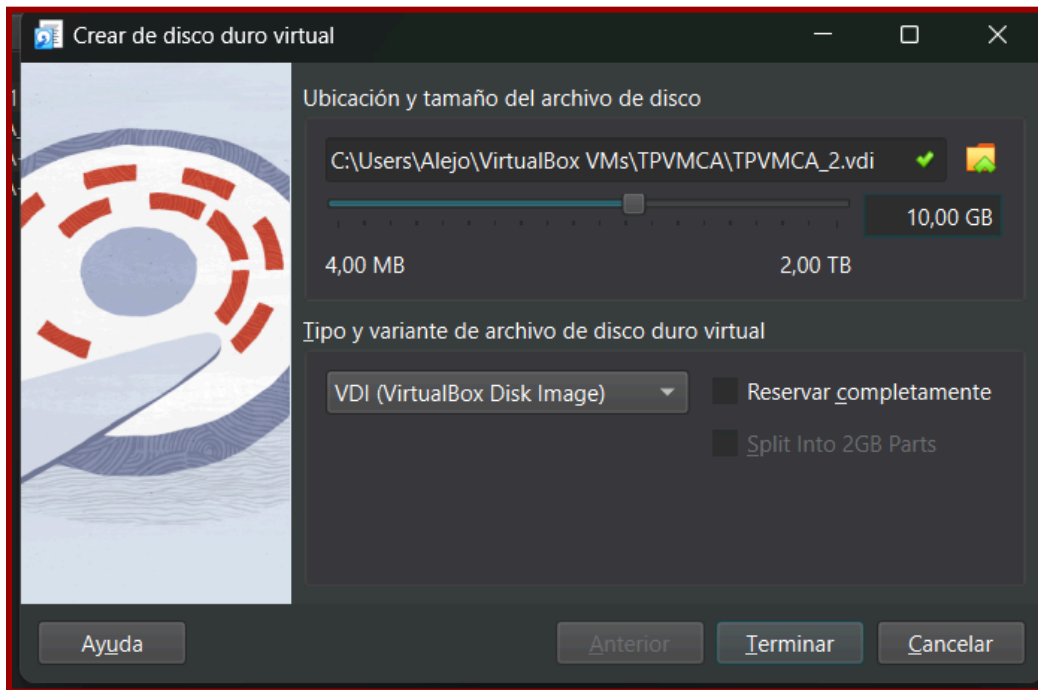
En la línea 12 modificamos la configuración predeterminada de “DHCP” (Dynamic Host Configuration Protocol) que asigna automáticamente la dirección IP, por “static”. Abajo agregamos las líneas que corresponden a la configuración de la IP, la netmask y la gateway.

Una vez hechas las modificaciones se ejecutan los comandos *ifdown enp0s3* y *ifup enp0s3*. De esta forma se guardan los cambios hechos.

Almacenamiento

Adicionar disco nuevo de 10GB

Con la máquina apagada agregamos desde la configuración un nuevo disco.



Se vuelve a iniciar la máquina y verificamos con el comando *lsblk* que se agregó el disco.

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda   8:0    0 10G  0 disk
├─sda1 8:1    0   8G  0 part /
├─sda2 8:2    0    1K  0 part
├─sda5 8:5    0    2G  0 part [SWAP]
sdb   8:16   0   8G  0 disk
├─sdb1 8:17   0   8G  0 part /home
sdc   8:32   0 10G  0 disk
```

Particiones, formateo y montaje

Ahora procedemos a hacer las particiones de 6GB y de 3GB con el comando *fdisk /dev/sdc* y se procede la configuración. Ahora cuando ejecutamos el *lsblk* veremos las particiones.

```
NAME      MAJ:MIN RM  SIZE RO  TYPE MOUNTPOINT
sda        8:0    0   10G  0 disk
├─sda1     8:1    0    8G  0 part /
├─sda2     8:2    0    1K  0 part
├─sda5     8:5    0    2G  0 part [SWAP]
sdb        8:16   0    8G  0 disk
├─sdb1     8:17   0    8G  0 part /home
sdc        8:32   0   10G  0 disk
├─sdc1     8:33   0    3G  0 part
└─sdc2     8:34   0    6G  0 part
```

Luego debemos formatear los discos con un tipo de filesystem específico con el comando *mkfs*. En nuestro caso aplicamos:

- *mkfs.ext4 /dev/sdc1*
- *mkfs.ext4 /dev/sdc2*

En los dos casos el sistema nos devuelve el siguiente mensaje:

```
mke2fs 1.46.2 (28-Feb-2021)
Creating filesystem with 786432 4k blocks and 196608 inodes
Filesystem UUID: 2c465d56-743f-4e98-afe8-6178ec62202b
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Con las particiones ya formateadas procedemos a montarlas directorios. Siguiendo las consignas del trabajo práctico estos directorios deben ser *www_dir* y *backup_dir*, así que primero los creamos con el comando *mkdir*.

Con los dos directorios creados, montamos los discos con *mount* de la siguiente forma:

- *mount /dev/sdc1 /www_dir/*
- *mount /dev/sdc2 /backup_dir/*

Luego ejecutamos *df -h* que nos mostrará la información sobre las particiones de nuestro sistema, la más importante es el espacio total de la partición.

```

root@TPServer:~# df -h
S.ficheros      Tamaño Usados  Disp Uso% Montado en
udev            968M      0  968M   0% /dev
tmpfs           198M    536K  197M   1% /run
/dev/sda1       7,8G    2,7G  4,7G  37% /
tmpfs           986M      0  986M   0% /dev/shm
tmpfs           5,0M      0   5,0M   0% /run/lock
/dev/sdb1       7,8G    32K   7,4G   1% /home
/dev/sdc1       2,9G    32K   2,8G   1% /www_dir
/dev/sdc2       5,9G    21M   5,5G   1% /backup_dir
tmpfs           198M      0  198M   0% /run/user/0
root@TPServer:~# _

```

Ahora, hay que editar el archivo de configuración `/etc/fstab` para que el montaje se guarde en el sistema.

```

11 UUID=e550cf9e-dc94-442c-b9c4-9c12f05024b3 none swap sw
12 /dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
13 /dev/sdb1 /home ext4 rw,auto 0 0
14 /dev/sdc1 /www_dir ext4 defaults 0 2
15 /dev/sdc2 /backup_dir ext4 defaults 0 2
~

```

Podemos comprobar que nuestras modificaciones hicieron efecto al reiniciar la máquina, ya que ahí podemos ver como automáticamente se montan las particiones que realizamos al inicio:

```

TPVMCA [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

Debian GNU/Linux 11 TPServer tty1
TPServer login: root
Password:
Linux TPServer 5.10.0-30-amd64 #1 SMP Debian 5.10.218-1 (2024-06-01) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun 13 20:32:49 -03 2025 on tty1
root@TPServer:~# df -h
S.ficheros      Tamaño Usados  Disp Uso% Montado en
udev            968M      0  968M   0% /dev
tmpfs           198M    536K  197M   1% /run
/dev/sda1       7,8G    2,7G  4,7G  37% /
tmpfs           986M      0  986M   0% /dev/shm
tmpfs           5,0M      0   5,0M   0% /run/lock
/dev/sdb1       7,8G    32K   7,4G   1% /home
/dev/sdc2       5,9G    21M   5,5G   1% /backup_dir
/dev/sdc1       2,9G    32K   2,8G   1% /www_dir
tmpfs           198M      0  198M   0% /run/user/0
root@TPServer:~#

```

Backup

Por último tenemos que crear un script. Primero vamos a crear un directorio dentro de /opt llamado "scripts". Luego creamos un archivo para el script ejecutando el comando *vi backup_full.sh*.

Se actualizan los permisos del directorio */opt/scripts* con el comando *chmod +x*. Verificamos con el comando *ls -l*

```
root@TPServer:/opt/scripts# ls -l
total 4
-rwxr-xr-x 1 root root 1576 jun 16 13:39 backup_full.sh
root@TPServer:/opt/scripts# _
```

Una vez creado el archivo *backup_full.sh* editamos el archivo con *vi* y agregamos el bloque de código, que será el script que va a automatizar las tareas de backup.

```
1 # Validación de la opción de ayuda o cantidad incorrecta de argumentos
2 if [[ "$1" == "-help" || "$#" -ne 2 ]]; then
3     echo "Uso: $(basename $0) <directorio_origen> <directorio_destino>"
4     echo "Ejemplo: $0 /etc /backup_dir"
5     exit 1
6 fi
7
8 # Definición de variables
9 ORIGEN="$1"
10 DESTINO="$2"
11 HOY=$(date +%Y%m%d)
12
13 # Verificación de existencia del directorio de origen
14 if [[ -d "$ORIGEN" ]]; then
15     echo "El directorio de origen $ORIGEN existe y está montado."
16
17     # Verificación de existencia del directorio de destino
18     if [[ -d "$DESTINO" ]]; then
19         echo "El directorio de destino $DESTINO existe y está montado."
20
21         # Nombre del archivo backup
22         NOMBRE_BACKUP="$(basename "$ORIGEN")_bkp_${HOY}.tar.gz"
23
24         # Creación del backup
25         if tar -czpf "$DESTINO/$NOMBRE_BACKUP" "$ORIGEN"; then
26             echo "Backup de $ORIGEN completado correctamente en $DESTINO/$NOMBRE_BACKUP." >> /var/log/bkpTP.log
27             echo "Backup finalizado correctamente."
28         else
29             echo "Error al crear el backup de $ORIGEN, el día $HOY" >> /var/log/bkpTP.log
30         fi
31     else
32         echo "Error: El directorio de destino $DESTINO no existe o no está montado."
33         echo "Error al crear el backup, el día $HOY, el directorio de destino $DESTINO no existe o no está montado." >> /var/log/bkpTP.log
34     fi
```

1,1 Comienzo


```

34     fi
35 else
36     echo "Error: El directorio de origen $ORIGEN no existe o no está montado."
37     echo "Error al crear el backup, el día $HOY, el directorio de origen $ORIGEN no existe o no
    está montado." >> /var/log/bkpTP.log
38 fi

```

38,1 Final

Probamos ejecutando el script con el comando `bash /opt/scripts/backup_full.sh [comando]`

```

root@TPServer:/opt/scripts# bash /opt/scripts/backup_full.sh -help
Uso: backup_full.sh <directorio_origen> <directorio_destino>
Ejemplo: /opt/scripts/backup_full.sh /etc /backup_dir
root@TPServer:/opt/scripts# _

```

Si agregamos -help nos da ayuda con ejemplo de uso

```

root@TPServer:/opt/scripts# bash /opt/scripts/backup_full.sh /var/log /backup_dir
El directorio de origen /var/log existe y está montado.
El directorio de destino /backup_dir existe y está montado.
tar: Eliminando la '/' inicial de los nombres
Backup finalizado correctamente.
root@TPServer:/opt/scripts#

```

Backup exitoso

```

root@TPServer:/opt/scripts# bash /opt/scripts/backup_full.sh /var/log /backup_di
El directorio de origen /var/log existe y está montado.
Error: El directorio de destino /backup_di no existe o no está montado.

```

Mensaje error

Luego, debemos lograr que el script ejecute y haga:

- TODOS LOS DÍAS a las 00:00 hs: Backupear “/var/logs”
- LUNES, MIÉRCOLES, VIERNES a las 23:00 hs: Backupear “/www_dir”

Esto lo vamos a poder hacer mediante *CRON*. Lo que debemos hacer es modificar el archivo `/etc/crontab` para poder programar los días y horarios indicados.

```

1 # /etc/crontab: system-wide crontab
2 # Unlike any other crontab you don't have to run the `crontab'
3 # command to install the new version when you edit this file
4 # and files in /etc/cron.d. These files also have username fields,
5 # that none of the other crontabs do.
6
7 SHELL=/bin/sh
8 PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
9
10 # Example of job definition:
11 # .----- minute (0 - 59)
12 # | .----- hour (0 - 23)
13 # | | .----- day of month (1 - 31)
14 # | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
15 # | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
16 # | | | |
17 # * * * * * user-name command to be executed
18 17 * * * * root cd / && run-parts --report /etc/cron.hourly
19 25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily
20 y )
21 47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.week
22 ly )
23 52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.mon
24 hly )
25 #

```

(Archivo antes de agregar la configuración)

```

18 17 * * * * root cd / && run-parts --report /etc/cron.hourly
19 25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily
20 y )
21 47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.week
22 ly )
23 52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.mon
24 hly )
25 00 00 * * * root bash /opt/scripts/backup_full.sh /var/log /backup_dir
26 00 23 * * 1,4 root bash /opt/scripts/backup_full.sh /www_dir /backup_dir
27 #

```

(Archivo después de agregar la configuración - línea 22 y 23)

Entregables

Para poder subir los directorios a un repositorio de GitHub primero creamos una carpeta en la que vamos a guardar el contenido de los directorios en formato *tar*. En nuestro caso lo llamamos *tp_integrador*. Luego ubicados dentro de este directorio, ejecutamos los siguientes comandos para comprimir los directorios en formato *tar*:

- `tar -czvf root.tar.gz /root`
- `tar -czvf etc.tar.gz /etc`

- `tar -czvf opt.tar.gz /opt`
- `tar -czvf proc.tar.gz /proc`
- `tar -czvf www_dir.tar.gz /www_dir`
- `tar -czvf backup_dir.tar.gz /backup_dir`
- `tar -czvf var.tar.gz /var`

Como este último directorio es muy pesado, lo dividimos en partes más pequeñas con el comando: `split -b 50M var.tar.gz var_part_`

Después removemos con `rm` el archivo `var.tar.gz` (que es el archivo completo, no dividido) porque ya una vez particionado, no nos sirve, ya que se subirán solo las divisiones. El resultado de estos pasos es el siguiente:

```
root@TPServer:~/tp_integrador# ls
backup_dir.tar.gz  opt.tar.gz    root.tar.gz  var_part_ab  var_part_ad  www_dir.tar.gz
etc.tar.gz        proc.tar.gz  var_part_aa  var_part_ac  var_part_ae
root@TPServer:~/tp_integrador# _
```

Proseguimos con la configuración de Git en la máquina virtual. Primero se instaló Git en el sistema operativo Debian con `apt-get install git`. Luego Git necesita conocer el nombre y el correo del usuario para registrar la autoría de los commits, y para esto ejecutamos:

- `git config --global user.name "Alejo Flores"`
- `git config --global user.email "floresalejo87@gmail.com"`

Esto guarda los datos en el archivo `~/.gitconfig` y se aplica a todos los repositorios del sistema.

Continuamos inicializando el repositorio local con `git init`, dentro del directorio `tp_integrador`. Después se establece "main" como rama principal con el comando `git branch -M main`.

Luego, debemos asociar la máquina al repositorio remoto de GitHub. Esto lo logramos con el comando `git remote add origin https://github.com/usuario/repo.git` - En nuestro caso añadimos la URL de nuestro repositorio y nos quedó así:

```
git remote add origin https://github.com/aflores2006/TP_ComputacionAplicada.git
```

Para el siguiente paso debemos agregar los archivos al repositorio desde la máquina virtual. Nosotros al intentarlo nos daba un error, y es que, al subirlos todos juntos, se excede el límite de 100MiB de subida.

```

root@TPServer:~/tp_integrador# git push -u origin main
Username for 'https://github.com': aflores2006
Password for 'https://aflores2006@github.com':
Enumerando objetos: 17, listo.
Contando objetos: 100% (17/17), listo.
Compresión delta usando hasta 2 hilos
Comprimiendo objetos: 100% (16/16), listo.
Escribiendo objetos: 100% (16/16), 242.36 MiB | 500.00 KiB/s, listo.
Total 16 (delta 6), reusado 0 (delta 0), pack-reusado 0
remote: Resolving deltas: 100% (6/6), done.
remote: error: Trace: 757c000b14df3d6c3f3a306a867f507671489a3475e6dcfa7a4086f952888c2e
remote: error: See https://gh.io/lfs for more information.
remote: error: File var.tar.gz is 205.29 MB; this exceeds GitHub's file size limit of 100.00 MB
remote: error: GH001: Large files detected. You may want to try Git Large File Storage - https://git
-lfs.github.com.
To https://github.com/aflores2006/TP_ComputacionAplicada.git
! [remote rejected] main -> main (pre-receive hook declined)
error: falló el push de algunas referencias a 'https://github.com/aflores2006/TP_ComputacionAplicada
.git'

```

El error que nos aparecía

Por lo tanto, optamos por subir por separado los directorios *var*. Procedimos primero entonces a cargar los demás directorios con el comando *git add *.tar.gz*. De esta forma, solo añadimos los directorios que tienen la terminación “tar.gz”.

Para completar la subida ponemos los comandos *git commit -m "Subo directorios"* y *git push -u origin main*. Allí nos pedirá el usuario y contraseña; en el primero ponemos simplemente nuestro usuario de GitHub, y para la contraseña debemos generar un token personal. Para obtenerlo hay que seguir los siguientes pasos:

- Inicia sesión en GitHub en tu navegador:
- Hacé clic en tu foto arriba a la derecha → Settings (Configuración)
- En el menú izquierdo, entrá a:
Developer settings → Personal access tokens → Tokens (classic)
- Clic en el botón “Generate new token” → “Generate new token (classic)”

Y ahí se te va a generar una suerte de código que va a funcionar como contraseña para establecer la conexión con GitHub.

Como el repositorio remoto ya tenía contenido (README), se resolvieron los conflictos con:

- *git pull origin main --allow-unrelated-histories* (De esta forma el README era “traído” al directorio local y se lo reconocía)

Una vez hecho esto, comenzará la subida. Aparecerá un mensaje similar a este:

```

root@TPServer:~/tp_integrador# git push -u origin main
Username for 'https://github.com': aflores2006
Password for 'https://aflores2006@github.com':
Enumerando objetos: 17, listo.
Contando objetos: 100% (17/17), listo.
Compresión delta usando hasta 2 hilos
Comprimiendo objetos: 100% (16/16), listo.
Escribiendo objetos: 18% (3/16), 29.66 MiB | 246.00 KiB/s

```

Luego, en nuestro caso, subimos los directorios de *var* por separado, así que después de esto hicimos:

- *git add var_part_** (esto incluye a *var_part_aa*, *var_parte_ab*, etc)
- *git commit -m "Subo /var"*
- *git push -u origin main*
- Ponemos usuario y contraseña

Una vez finalizada la carga podemos ver en GitHub los directorios subidos.

aflores2006 Subo /var		c366592 · 1 hour ago	🕒 7 Commits
📄 README.md	Update README.md	10 hours ago	
📄 backup_dir.tar.gz	Subo directorios	2 hours ago	
📄 etc.tar.gz	Subo directorios	2 hours ago	
📄 opt.tar.gz	Subo directorios	2 hours ago	
📄 proc.tar.gz	Subo directorios	2 hours ago	
📄 root.tar.gz	Subo directorios	2 hours ago	
📄 var_part_aa	Subo /var	1 hour ago	
📄 var_part_ab	Subo /var	1 hour ago	
📄 var_part_ac	Subo /var	1 hour ago	
📄 var_part_ad	Subo /var	1 hour ago	
📄 var_part_ae	Subo /var	1 hour ago	
📄 www_dir.tar.gz	Subo directorios	2 hours ago	

—

|||