

Ingeniería de Servidores: Práctica 2.

Daniel Monjas Miguélez

Lección 1:

Para poder instalar nuevos servicios, existen gestores de paquetes que permiten realizar esta tarea de una manera muy sencilla. Aunque estas aplicaciones tienen una Graphical User Interface (GUI), es recomendable saber utilizarlas en la consola. En el caso de Windows, muchos servicios proporcionados por Microsoft pueden gestionarse mediante una ventana de configuración. En esta sección se aprenderá cómo utilizar estas herramientas desde la línea de comandos (aunque también existen GUIs que permiten la gestión de paquetes) que nos permiten instalar los servicios de manera fiable y segura.

En esta práctica, empezaremos por instalar ssh en la máquina CentOS y en la máquina UbuntuServer. Empezaremos por la máquina UbuntuServer. Para la máquina de UbuntuServer debemos instalar “ssh-client” y “ssh-server” pues queremos que nuestro servidor se pueda conectar por ssh a otros terminales y otros terminales se puedan conectar por ssh al mismo. Para ello hacemos “*sudo apt install openssh-client openssh-server*”

```
momid@momid:~$ sudo apt install openssh-server openssh-client
[sudo] password for momid:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version (1:8.2p1-4ubuntu0.2).
openssh-client set to manually installed.
The following additional packages will be installed:
  libwrap0 ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  libwrap0 ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 5 newly installed, 0 to remove and 102 not upgraded.
Need to get 734 kB of archives.
After this operation, 6121 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://es.archive.ubuntu.com/ubuntu focal/main amd64 libwrap0 amd64 7.6.q-30 [46.3 kB]
Get:2 http://es.archive.ubuntu.com/ubuntu focal/main amd64 ncurses-term all 6.2-0ubuntu2 [249 kB]
Get:3 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-sftp-server amd64 1:8.2p1-4ubuntu0.2 [51.5 kB]
Get:4 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-server amd64 1:8.2p1-4ubuntu0.2 [377 kB]
Get:5 http://es.archive.ubuntu.com/ubuntu focal/main amd64 ssh-import-id all 5.10-0ubuntu1 [10.0 kB]
Fetched 734 kB in 1s (837 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libwrap0:amd64.
(Reading database ... 68300 files and directories currently installed.)
Preparing to unpack .../libwrap0_7.6.q-30_amd64.deb ...
Unpacking libwrap0:amd64 (7.6.q-30) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../ncurses-term_6.2-0ubuntu2_all.deb ...
Unpacking ncurses-term (6.2-0ubuntu2) ...
Progress: [ 14%] [#####.....]
```

Tras esto podemos comprobar si se ha instalado correctamente ssh por medio de los comandos “*sudo systemctl status ssh*” o “*sudo systemctl status sshd*” y nos debería aparecer “*running*” (ambos comandos son válidos).

```

momid@momid:~$ sudo systemctl status ssh
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-03-27 16:20:38 UTC; 1min 24s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 2487 (sshd)
    Tasks: 1 (limit: 4619)
   Memory: 1.2M
   CGroup: /system.slice/ssh.service
           └─2487 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 27 16:20:38 momid systemd[1]: Starting OpenBSD Secure Shell server...
Mar 27 16:20:38 momid sshd[2487]: Server listening on 0.0.0.0 port 22.
Mar 27 16:20:38 momid sshd[2487]: Server listening on :: port 22.
Mar 27 16:20:38 momid systemd[1]: Started OpenBSD Secure Shell server.
momid@momid:~$ sudo systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-03-27 16:20:38 UTC; 1min 26s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 2487 (sshd)
    Tasks: 1 (limit: 4619)
   Memory: 1.2M
   CGroup: /system.slice/ssh.service
           └─2487 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 27 16:20:38 momid systemd[1]: Starting OpenBSD Secure Shell server...
Mar 27 16:20:38 momid sshd[2487]: Server listening on 0.0.0.0 port 22.
Mar 27 16:20:38 momid sshd[2487]: Server listening on :: port 22.
Mar 27 16:20:38 momid systemd[1]: Started OpenBSD Secure Shell server.
momid@momid:~$

```

En caso de que nos ponga que el servicio no esté corriendo haremos “*sudo systemctl enable ssh*” y tras esto “*sudo systemctl restart ssh*” (también es válido escribiendo `sshd` en vez de `ssh`). Una vez esté corriendo se observa en la imagen que `ssh` está escuchando en el puerto 22 que es el puerto por defecto de `ssh`. Para comprobar esto último desde nuestro ordenador normal podemos conectarnos a la máquina por medio de “*ssh usuario@direccion_ip -p numero_puerto*”.

```

daniel@daniel-XPS-15-9570:~$ ssh momid@192.168.56.105 -p 22
momid@192.168.56.105's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-67-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Mar 27 16:25:29 UTC 2021

System load:  0.0               Processes:           126
Usage of /home: 0.2% of 975MB   Users logged in:    1
Memory usage:  6%              IPv4 address for enp0s3: 10.0.2.15
Swap usage:    0%              IPv4 address for enp0s8: 192.168.56.105

102 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Sat Mar 27 16:17:04 2021
momid@momid:~$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
loop0                               7:0      0 71.3M  1 loop  /snap/lxd/16099
loop1                               7:1      0   55M   1 loop  /snap/core18/1880
loop2                               7:2      0 29.9M   1 loop  /snap/snapd/8542
loop3                               7:3      0 32.3M   1 loop  /snap/snapd/11402
loop4                               7:4      0 55.5M   1 loop  /snap/core18/1988
loop5                               7:5      0 70.4M   1 loop  /snap/lxd/19647
sda                                  8:0      0   10G   0 disk
├─sda1                              8:1      0    1M   0 part
├─sda2                              8:2      0 300M   0 part
├─┬md0                              9:0      0 299M   0 raid1
│ └─md0p1                          259:0    0 294M   0 part  /boot
├─sda3                              8:3      0   9.7G   0 part
├─┬md1                              9:1      0   9.7G   0 raid1
│ └─dm_crypt-0                    253:0    0   9.7G   0 crypt
│   └─vg0-lv--home                253:1    0    1G   0 lvm    /home
│     └─vg0-lv--root              253:2    0    8G   0 lvm    /
│       └─vg0-lv--swap            253:3    0 692M   0 lvm    [SWAP]
└─sdb                               8:16     0   10G   0 disk
    ├─sdb1                          8:17     0    1M   0 part
    ├─sdb2                          8:18     0 300M   0 part
    ├─┬md0                          9:0      0 299M   0 raid1
    │ └─md0p1                      259:0    0 294M   0 part  /boot
    ├─sdb3                          8:19     0   9.7G   0 part
    ├─┬md1                          9:1      0   9.7G   0 raid1
    │ └─dm_crypt-0                253:0    0   9.7G   0 crypt
    │   └─vg0-lv--home            253:1    0    1G   0 lvm    /home
    │     └─vg0-lv--root          253:2    0    8G   0 lvm    /
    │       └─vg0-lv--swap        253:3    0 692M   0 lvm    [SWAP]
    └─sr0                           11:0     1 1024M   0 rom

```

Vemos que el lsblk se corresponde con el de nuestra máquina UbuntuServer como vimos en la lección uno de la práctica 1.

Tras esto iremos al archivo de configuración de ssh para cambiar entre otras cosas el puerto por defecto y así evitar posibles ataque y por otro lado quitaremos la posibilidad de ingresar en el sistema como usuario root. De forma que para poder usar permisos de superusuario habría que entrar como un usuario incluido en el archivo de sudoers y tras esto usar un sudo su. A partir de aquí utilizaremos el comando “sudo su” para activar los permisos de administrador y así no tener que poner todo el rato sudo precediendo a todos los comandos. Tras esto haremos “vi /etc/ssh/sshd_config” lo cual nos permite ir al archivo de configuración de ssh y hacemos lo siguiente

```

Include /etc/ssh/sshd_config.d/*.conf

Port 22022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
"/etc/ssh/sshd_config" 123L, 3275C written
root@momid:/home/momid#

```

Descomentamos la línea de Port y escribimos el puerto que queramos abrir (en este caso el 22022), y también descomentamos la línea de PermitRootLogin y cambiamos lo que tenga como parámetro por un no.

Si tras esto hacemos “*systemctl status sshd*” veremos que ssh sigue escuchando en el puerto 22. Para aplicar los cambios en el fichero de configuración hacemos “*systemctl restart sshd*” y con esto se reiniciará el servicio aplicando los cambios introducidos.

```

root@momid:/home/momid# systemctl restart sshd
root@momid:/home/momid# systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-03-27 16:35:23 UTC; 5s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 3032 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 3045 (sshd)
    Tasks: 1 (limit: 4619)
   Memory: 1.1M
    CGroup: /system.slice/ssh.service
            └─3045 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 27 16:35:23 momid systemd[1]: Stopping OpenBSD Secure Shell server...
Mar 27 16:35:23 momid systemd[1]: ssh.service: Succeeded.
Mar 27 16:35:23 momid systemd[1]: Stopped OpenBSD Secure Shell server.
Mar 27 16:35:23 momid systemd[1]: Starting OpenBSD Secure Shell server...
Mar 27 16:35:23 momid sshd[3045]: Server listening on 0.0.0.0 port 22022.
Mar 27 16:35:23 momid sshd[3045]: Server listening on :: port 22022.
Mar 27 16:35:23 momid systemd[1]: Started OpenBSD Secure Shell server.
root@momid:/home/momid# _

```

Se puede ver que ahora ssh si escucha en el puerto 22022. Para comprobarlo podemos conectarnos de nuevo remotamente.

```
daniel@daniel-XPS-15-9570:~$ ssh momid@192.168.56.105 -p 22
ssh: connect to host 192.168.56.105 port 22: Connection refused
daniel@daniel-XPS-15-9570:~$ ssh momid@192.168.56.105 -p 22022
momid@192.168.56.105's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-67-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Mar 27 16:36:57 UTC 2021

System load:  0.0               Processes:            127
Usage of /home: 0.2% of 975MB   Users logged in:     1
Memory usage:  6%              IPv4 address for enp0s3: 10.0.2.15
Swap usage:    0%              IPv4 address for enp0s8: 192.168.56.105

102 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Sat Mar 27 16:36:35 2021 from 192.168.56.1
momid@momid:~$
```

Se puede ver que al intentar conectarnos por el puerto 22 nos rechaza la conexión, mientras que por el puerto 22022 si nos deja. Si además probamos a registrarnos como root no nos aceptará la contraseña.

Ahora empezaremos el mismo proceso pero esta vez para CentOS. En este caso CentOS por defecto trae instalado sshd, luego si intentamos hacer “*sudo systemctl status ssh*” nos dará error. Podemos probar que así es conectándonos desde nuestra máquina UbuntuServer

```
root@momid:/home/momid# ssh momid@192.168.56.110 -p 22
The authenticity of host '192.168.56.110 (192.168.56.110)' can't be established.
ECDSA key fingerprint is SHA256:wAtTyFUSC1W24ryYzx6IDE60fIFsxVoLbJ/nga/miZY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.110' (ECDSA) to the list of known hosts.
momid@192.168.56.110's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Mar 27 12:39:55 2021
[momid@localhost ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0   8G  0 disk
├─sda1       8:1    0    1G  0 part /boot
├─sda2       8:2    0    7G  0 part
│   └─cl-root 253:0    0  6.2G  0 lvm  /
│       └─cl-swap 253:1    0  820M  0 lvm  [SWAP]
└─sdb        8:16   0    8G  0 disk
    └─cl-newvar 253:2    0    1G  0 lvm  /var
sr0         11:0    1 1024M  0 rom
```

Vemos que nos conectamos al puerto 22 pues es el que viene por defecto, y además al conectarnos vemos que la configuración de particiones corresponde con la de CentOS LVM, realizada en la lección dos de la práctica 1. También podemos ver que a la inversa funciona

```
[momid@localhost ~]$ ssh momid@192.168.56.105 -p 22022
The authenticity of host '192.168.56.105:22022 ([192.168.56.105]:22022)' can't be established.
ECDSA key fingerprint is SHA256:SYwYgmkKxnTqKUpfALtZJagSrLQhE6Uy4z803ER02ZA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.105:22022' (ECDSA) to the list of known hosts.
momid@192.168.56.105's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-67-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Mar 27 16:44:09 UTC 2021

System load:  0.0               Processes:    130
Usage of /home: 0.2% of 975MB   Users logged in: 1
Memory usage:  6%              IPv4 address for enp0s3: 10.0.2.15
Swap usage:    0%              IPv4 address for enp0s8: 192.168.56.105

102 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Sat Mar 27 16:36:57 2021 from 192.168.56.1
momid@momid:~$ _
```

Al igual que hemos hecho con Ubuntu lo haremos con CentOS, nos vamos al mismo fichero de configuración, y hacemos el mismo cambio que hemos hecho en UbuntuServer (el puerto y el PermitRootLogin). Sin embargo, esta vez si hacemos el “*sudo systemctl restart sshd*” nos dará error

```
[root@localhost momid]# sudo systemctl restart
Too few arguments.
[root@localhost momid]# sudo systemctl restart sshd
Job for sshd.service failed because the control process exited with error code.
See "systemctl status sshd.service" and "journalctl -xe" for details.
[root@localhost momid]#
```


Si tras esto usamos el comando “*journalctl -xe*” nos dará más información sobre el fallo ocurrido.

```
-- Support: https://access.redhat.com/support
--
-- Unit sshd-keygen.target has finished shutting down.
mar 27 12:49:06 localhost.localdomain systemd[1]: Stopping sshd-keygen.target.
-- Subject: Unit sshd-keygen.target has begun shutting down
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
--
-- Unit sshd-keygen.target has begun shutting down.
mar 27 12:49:06 localhost.localdomain systemd[1]: Reached target sshd-keygen.target.
-- Subject: Unit sshd-keygen.target has finished start-up
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
--
-- Unit sshd-keygen.target has finished starting up.
--
-- The start-up result is done.
mar 27 12:49:06 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
-- Subject: Unit sshd.service has begun start-up
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
--
-- Unit sshd.service has begun starting up.
mar 27 12:49:06 localhost.localdomain sshd[16201]: error: Bind to port 22022 on 0.0.0.0 failed: Perm>
mar 27 12:49:06 localhost.localdomain sshd[16201]: error: Bind to port 22022 on :: failed: Permissio>
mar 27 12:49:06 localhost.localdomain sshd[16201]: fatal: Cannot bind any address.
mar 27 12:49:06 localhost.localdomain systemd[1]: sshd.service: Main process exited, code=exited, s>
mar 27 12:49:06 localhost.localdomain systemd[1]: sshd.service: Failed with result 'exit-code'.
mar 27 12:49:06 localhost.localdomain systemd[1]: Failed to start OpenSSH server daemon.
-- Subject: Unit sshd.service has failed
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
--
-- Unit sshd.service has failed.
--
-- The result is failed.
lines 2205-2240/2240 (END)
```

Al ejecutar el comando vemos en rojo que nos deniega el permiso para coger el puerto 22022. Esto se debe a que en CentOS hay que avisar a SELinux del cambio. Esto se indica en el archivo de configuración justo encima del puerto.

```
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.
#
# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Nos dice que tenemos que usar el comando “*semanage*”, sin embargo, por defecto CentOS no trae instalado *semanage*. Para instalarlo hacemos un “*yum install semanage*” nos sale algo así

```
[root@localhost momid]# yum install semanage
CentOS-8 - AppStream                3.5 MB/s | 6.3 MB      00:01
CentOS-8 - Base                     1.8 MB/s | 2.3 MB      00:01
CentOS-8 - Extras                   4.0 kB/s | 9.6 kB      00:02
No match for argument: semanage
Error: No se pudo encontrar ningún resultado: semanage
[root@localhost momid]# _
```

Básicamente CentOS nos informa de que no encuentra el paquete. En este caso lo que haremos será “yum provides semanage” lo cual hará que CentOS nos busque donde está el paquete

```
[root@localhost momidl# yum provides semanage
Última comprobación de caducidad de metadatos hecha hace 0:01:34, el sáb 27 mar 2021 12:53:04 EDT.
polycoreutils-python-utils-2.9-9.el8.noarch : SELinux policy core python utilities
Repositorio : BaseOS
Resultado de:
Archivo      : /usr/sbin/semanage

[root@localhost momidl# _
```

En la captura se ve que el paquete “polycoreutils-python-utils” contiene “semanage” luego instalamos este ultimo paquete.

```
[root@localhost momidl# yum install polycoreutils-python-utils
Última comprobación de caducidad de metadatos hecha hace 0:02:50, el sáb 27 mar 2021 12:53:04 EDT.
Dependencias resueltas.
=====
Paquete                Arq.      Versión                Repo      Tam.
=====
Instalando:
polycoreutils-python-utils  noarch    2.9-9.el8              BaseOS    251 k
Actualizando:
libsemanage              x86_64    2.9-3.el8              BaseOS    165 k
Instalando dependencias:
checkpolicy              x86_64    2.9-1.el8              BaseOS    348 k
python3-audit             x86_64    3.0-0.17.20191104git1c2f876.el8 BaseOS     86 k
python3-libsemanage       x86_64    2.9-3.el8              BaseOS    127 k
python3-polycoreutils     noarch    2.9-9.el8              BaseOS    2.2 M
python3-setools           x86_64    4.3.0-2.el8            BaseOS    626 k

Resumen de la transacción
=====
Instalar      6 Paquetes
Actualizar    1 Paquete

Tamaño total de la descarga: 3.8 M
¿Está de acuerdo [s/N]?:
```

Vemos que cuando nos lista los paquete que se instalan/actualizan esta libsemanage. Para probar que está instalado podemos hacer simplemente “semanage”.

Tras esto podemos probar “semanage port -l” el cual nos dirá los puertos del sistema y que tienen asociados. En nuestro caso filtraremos la búsqueda a ssh con “semanage port -l | grep ssh”

```
[root@localhost momidl# semanage port -l | grep ssh
ssh_port_t                tcp      22
[root@localhost momidl# _
```

Vemos que el puerto 22022 no está reconocido como puerto para ssh, luego le indicaremos a CentOS que el 22022 es un puerto para ssh y luego ya reiniciaremos el servicio ssh. Para añadir el puerto hacemos “semanage port -a -t ssh_port_t -p tcp 22022”, donde el -a nos indica añadir puerto, el -t nos indica el tipo, en este caso ssh_port_t y por último el -p nos indica el numero y tipo de puerto, en este caso, tcp 22022.

```
[root@localhost momidl# semanage port -a -t ssh_port_t -p tcp 22022
[ 1311.527960] SELinux: Converting 2311 SID table entries...
[ 1312.093684] SELinux: Context system_u:unconfined_r:sandbox_t:s0-s0:c0.c1023 became invalid (unmapped).
[ 1312.704565] SELinux: Context unconfined_u:unconfined_r:sandbox_t:s0-s0:c0.c1023 became invalid (unmapped).
[ 1312.749686] SELinux: policy capability network_peer_controls=1
[ 1312.749704] SELinux: policy capability open_perms=1
[ 1312.749714] SELinux: policy capability extended_socket_class=1
[ 1312.749725] SELinux: policy capability always_check_network=0
[ 1312.749737] SELinux: policy capability cgroup_seclabel=1
[ 1312.749756] SELinux: policy capability mnp_nosuid_transition=1
[root@localhost momidl# _
```


Con esto quedaría añadido nuestro puerto, para ello volvemos a hacer el comando de antes

```
[root@localhost momid]# semanage port -l | grep ssh
ssh_port_t                tcp        22022, 22
[root@localhost momid]# _
```

Vemos que se ha añadido correctamente. Por otro lado el puerto 22 no se puede eliminar por ser el puerto por defecto, luego si bien está asignado a ssh, no se podrá usar pues no lo contemplamos en el archivo de configuración. Tras esto reiniciamos el servicio ssh.

```
[root@localhost momid]# systemctl restart sshd
[root@localhost momid]# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-03-27 13:07:02 EDT; 4s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 23304 (sshd)
    Tasks: 1 (limit: 23960)
   Memory: 1.1M
   CGroup: /system.slice/sshd.service
           └─23304 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com

mar 27 13:07:02 localhost.localdomain systemd[1]: Stopped OpenSSH server daemon.
mar 27 13:07:02 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
mar 27 13:07:02 localhost.localdomain sshd[23304]: Server listening on 0.0.0.0 port 22022.
mar 27 13:07:02 localhost.localdomain sshd[23304]: Server listening on :: port 22022.
mar 27 13:07:02 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
lines 1-16/16 (END)
```

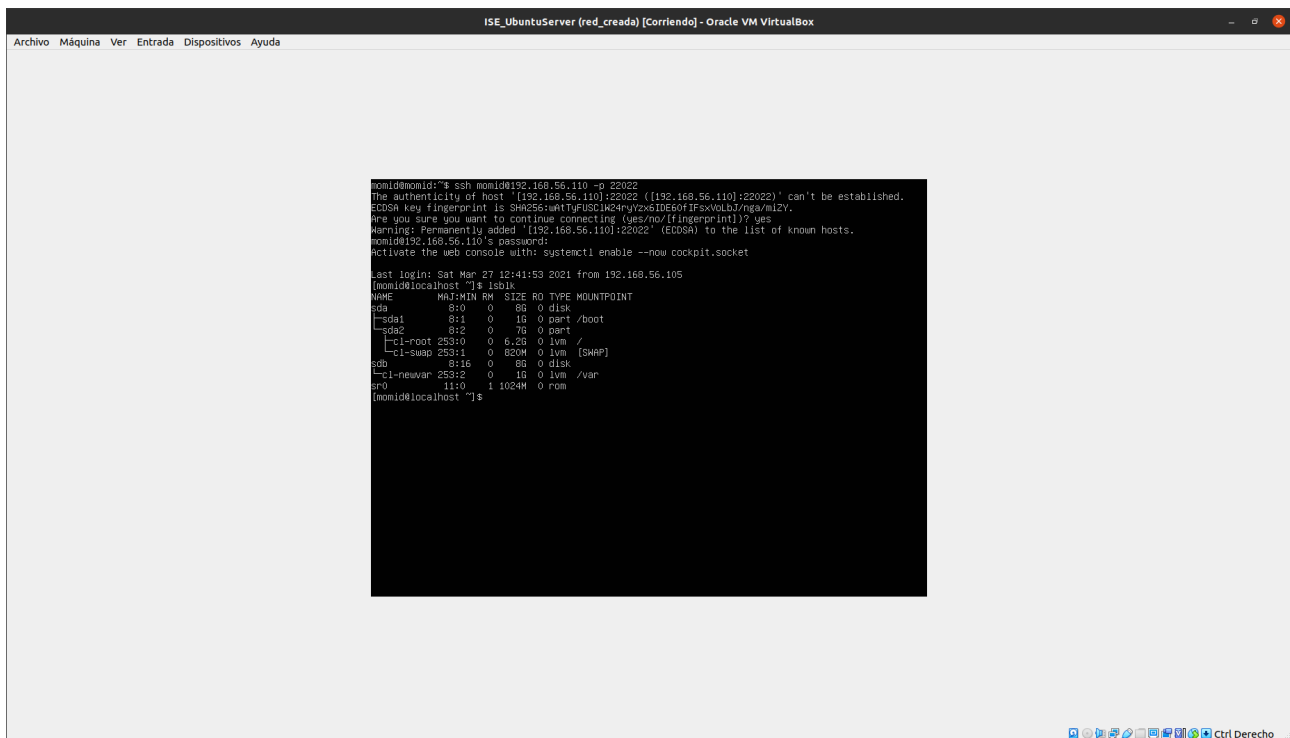
Vemos que ssh se ha actualizado y que ya escucha en el puerto 22022. Podemos comprobar la conexión desde Ubuntu.

```
momid@momid:~$ ssh momid@192.168.56.110 -p 22022
ssh: connect to host 192.168.56.110 port 22022: No route to host
momid@momid:~$
```

Sin embargo, nos da error, pues aparte del sistema de seguridad, tenemos que abrir el puerto 22022. Para abrirlo usamos el comando “*sudo firewall-cmd --permanent --add-port=22022/tcp*” lo cual nos indica “*--permanent*” que el puerto se añada de forma permanente (sino se cierra el apagar el sistema), y por otro lado “*--add-port=22022/tcp*” que se añada el puerto 22022 para trafico tcp (destacar que este comando en ubuntu server es *firewall-ufw*, mientras que en CentOS es *firewall-cmd*). Tras esto hacemos “*firewall-cmd --reload*” para que el cambio se haga efectivo.

```
[root@localhost momid]# firewall-cmd --permanent --add-port=22022/tcp
success
[root@localhost momid]# firewall-cmd --reload
success
[root@localhost momid]# _
```

Vemos que en ambas operaciones nos devuelve que se han realizado con éxito. Ahora volvemos a probar conectarnos desde UbuntuServer, y veremos que funciona



Por último, compartiremos la contraseña de CentOS a Ubuntu y viceversa, de forma que ya no sea necesario autenticarse al conectarse entre estas máquinas, y prohibiremos la autenticación por contraseña. Para ello en ambas máquinas hacemos “ssh-keygen”, con lo que generaremos la contraseña (darle a enter hasta que aparezca la imagen).

```

momid@momid:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/momid/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/momid/.ssh/id_rsa
Your public key has been saved in /home/momid/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:kITLQnPP2twm5JEelGYZEdmAz7oWd3MPoz2zIYuY7n4 momid@momid
The key's randomart image is:
+---[RSA 3072]-----+
|
| o*X
| o +.B..
| . + Xo.
| . o X.
| . O +S
| + B = +
| + +.=.+
| ooE..o+..
| =*.. . +
+---[SHA256]-----+
momid@momid:~$

```

Si nos fijamos en la tercera línea nos indica el directorio y archivo que guardarán la clave que estamos generando. Por eso esta orden debe hacerse sin sudo, pues en caso contrario se almacenaría en “/root/.ssh/id_rsa”, con lo que nos daría luego problemas a la hora de compartir esta contraseña.

```

[momid@localhost ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/momid/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/momid/.ssh/id_rsa.
Your public key has been saved in /home/momid/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:K4wCj0oWx84SftNDIPrQkzjmrS8bvghQUmPFndGeoxU momid@localhost.localdomain
The key's randomart image is:
+---[RSA 3072]-----+
|  +o...+ |
| o .. o E |
| . . . o |
| = o . . = |
| o * + ooS. |
| oB = o.o . |
| =.O * + . |
| =+. * = + |
| o**o . . |
+---[SHA256]-----+
[momid@localhost ~]# _

```

Una vez creadas las contraseñas en ambos sistemas, las compartimos con el otro usando “*ssh-copy-id user@ip -p puerto*”, y tras ejecutar el comando deberemos introducir la contraseña, como si nos estuviéramos conectando con ssh de forma normal.

```

momid@momid:~$ ssh-copy-id momid@192.168.56.110 -p 22022
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/momid/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
momid@192.168.56.110's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh -p '22022' 'momid@192.168.56.110'"
and check to make sure that only the key(s) you wanted were added.

momid@momid:~$

```

```

[momid@localhost ~]$ ssh-copy-id momid@192.168.56.105 -p 22022
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/momid/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install all the new keys
momid@192.168.56.105's password:
Permission denied, please try again.
momid@192.168.56.105's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh -p '22022' 'momid@192.168.56.105'"
and check to make sure that only the key(s) you wanted were added.

[momid@localhost ~]$ _

```

Como decíamos antes esta orden también se debe hacer sin sudo. Si lo hiciésemos con sudo intentaría compartir “/root/.ssh/id_rsa” en lugar del archivo donde hemos guardado la contraseña. Entonces la comprobación siguiente sólo nos funcionaría con “*sudo ssh*”, en otro caso fallaría.

Podemos comprobar que todo va bien haciendo un ssh normal entre las máquinas y viendo que no nos pide las contraseñas.

```

momid@momid:~$ ssh momid@192.168.56.110 -p 22022
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Apr  8 15:37:25 2021
[momid@localhost ~]$ _

```

```

[momid@localhost ~]$ ssh momid@192.168.56.105 -p 22022
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-67-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Apr  8 19:42:44 UTC 2021

System load:   0.19           Processes:            125
Usage of /home: 0.2% of 975MB Users logged in:        1
Memory usage:   5%           IPv4 address for enp0s3: 10.0.2.15
Swap usage:     0%           IPv4 address for enp0s8: 192.168.56.105

102 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Apr  8 19:39:12 2021
momid@momid:~$

```

Tras esto vamos al archivo de configuración de ssh “/etc/ssh/sshd_config”, y descomentamos la opción *PasswordAuthentication* y la ponemos a no, con lo que prohibiremos las conexiones que requieran autenticación. Para este paso anterior hay que haber compartido ambas contraseñas.

```

#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication. Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes

#AllowAgentForwarding yes
:wq_

```

```

# Logging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
PasswordAuthentication yes
:wq

```

Con esto terminaría definitivamente la lección uno.

Lección 2:

Copias de Seguridad

dd → Realiza una copia binaria. Permite realizar la copia de archivos y particiones enteros, pero no directorios enteros.

cp → Realiza una copia de un archivo o directorio

cpio → Realiza una copia de un archivo pero la salida es una imagen

tar → Utilidad para archivado

rsync → Herramienta rápida, versátil y remota de copia de archivos, sincronizando el origen y el destino de la copia

rsnapshot → Sirve para hacer instantáneas del sistema, como por ejemplo hacemos en VirtualBox

LVM snapshots → Sirva también para crear instantáneas tal cual del sistema.

Para hacer copias normales sirve con la copia binaria y la copia de archivos y empaquetado. Si queremos sincronizar el origen y el destino de la copia conviene usar rsync.

Ejemplos de uso

dd if=/dev/sda2 of=sda2Backup.img/iso (dependiendo del formato de salida que queramos)

Si ejecutamos esto sobre nuestra máquina UbuntuServer realizaremos una copia de la partición donde se encuentra boot.

ls | cpio -ov > pruebaBackup.cpio

Este comando nos crea una imagen de lo que haya actualmente en el directorio en el que nos encontremos. Para restaurar la imagen se hace

cpio -idv < pruebaBackup.cpio

- -o → indica que se va a crear una imagen
- -v → indica que muestre por pantalla de forma escrita lo que va haciendo (verbose)
- -i → indica que se va a extraer archivos de una imagen .cpio
- -d → indica que se creen directorios cuando sea necesarios

tar cvzf <nombre_comprimido>.tar.gz <archivos_comprimir>

Para descomprimir usamos

tar xvf <archivo_descomrpimir>

- c → crear un .tar.gz
- v → verbose
- z → filtrar el archivo a través de gzip
- f → indica que es un archivo
- x → extraer de un .tar.gz

Para rsync podemos probar

rsync ./ pruebaBackup*

La gracia de rsync es que el origen y el destino estarán sincronizado, para eso podemos usaremos

```
rsync -e "ssh -p 22022" prueba/* momid@192.168.56.110:/home/momid/ubuntuBackup
```

Con esto tendríamos una copia sincronizada, de forma que si se vuelve a hacer un rsync simplemente se encarga de hacer los cambios necesarios para que ambos directorios estén iguales.

- -e → especifica el shell remoto a utilizaremos

El uso común es `rsync -avz /home usu@maquina:/backup`, añadiendole `--delete` si se quiere un sincronización total. Para restaurar la copia se invierte el orden entre destion y origen

Control de cambios

Puede darse que al hacer una copia de seguridad nos carguemos sin querer tanto el respaldo como los datos que estábamos respaldando. Para solucionar este problema usaremos el control de versiones. El método más básico de control de cambios es

```
cp <archivo> <archivo>.old
```

Sin embargo, no es un control de cambios óptimo. Lo más útil y más utilizado suele ser *git*.

Como funciona git. La idea es que a una carpeta se le indica que va a tener control de versiones, entonces se crea un área con tres zonas que tendrá un directorio de trabajo, que es la carpeta en la que nosotros trabajamos. Internamente el sistema hará un seguimiento de cambio y nos indicará cuales son los cambios no guardados. Una vez guardados los cambios pasan a la zona stage, pero estos cambios todavía no serán definitivos. Una vez los hagamos definitivos los cambios de stage se mandan al repositorio, y entonces estos cambios ya serán definitivos.

Una vez estamos en la zona stage y pasamos los cambios al repositorio, se crea un identificador de 40 caracteres que identifican a la versión. Al último commit (envío al repositorio) se le denomina HEAD, y podemos desplazarnos a partir del identificador del HEAD

Método de uso

En primer lugar hacemos en un directorios

```
git init
```

que nos creará las zonas del área de control de versiones. Físicamente esto se ve en la carpeta con la creación de una carpeta `$pwd/.git`. Si borramos dicha carpeta se perderá todo control de cambios que hayamos realizado. También podemos identificarnos en `~/gitconfig` para así autenticarnos automáticamente en github. Una vez creado un *git init* podemos ver los cambios no guardados con

```
git status
```

y guardalos con

```
git add *
```

Si queremos mandarlos al respositorio hacemos un

```
git commit -m "etiqueta del commit"
```

Usando *git log* podemos ver los distintos commit que hemos realizado, su autor, la fecha así como el hash que lo identifica, y para restaurar un commit antiguo se hace

```
git reset <hash_commit_a_restaurar>
```

De forma que se sacan de la pila tantos commit como sean necesarios hasta llegar al requerido, y todos los commit que se saquen se pierden eliminandolos. Tras hacer esto volverá a detectar los cambios no registrados. Para volver a un commit anterior de forma que todo sea igual a como era hacemos

```
git checkout HEAD <fichero_a_restaurar>
```

con esto se restaura el fichero a como está en la cabeza actual de la pila.

Si queremos obtener más información de un commit usamos

```
git show <hash del commit>
```

Para subir los cambios al repositorio remoto hacemos

```
git origin <rama_a_la_que_queremos>
```

Ahora veremos como crear una rama en git para ello usamos

```
git checkout -b development
```

y entonces se creará y cambiará a esa rama. Podemos consultar las ramas que tenemos con

```
git branch
```

Para pasar la información de una rama y llevarla a la master tenemos que hacer un Compare & pull request. Al igual que para mandar información al repositorio era git push ahora hacemos

```
git pull origin <rama>
```

Y los cambios quedarán registrados en git log.

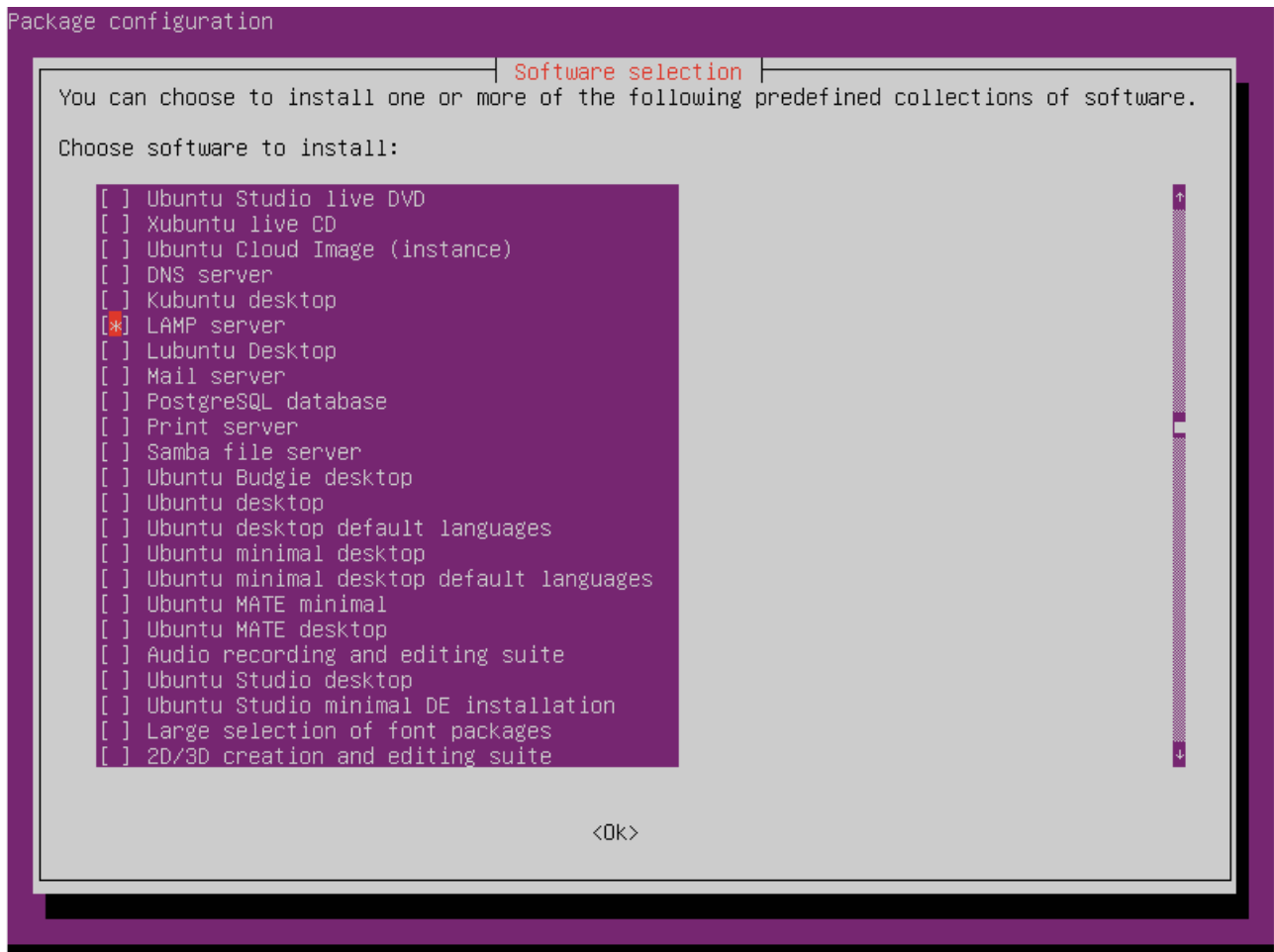
Lección 3:

En esta práctica instalaremos tanto en UbuntuServer como en CentOS un servidor.

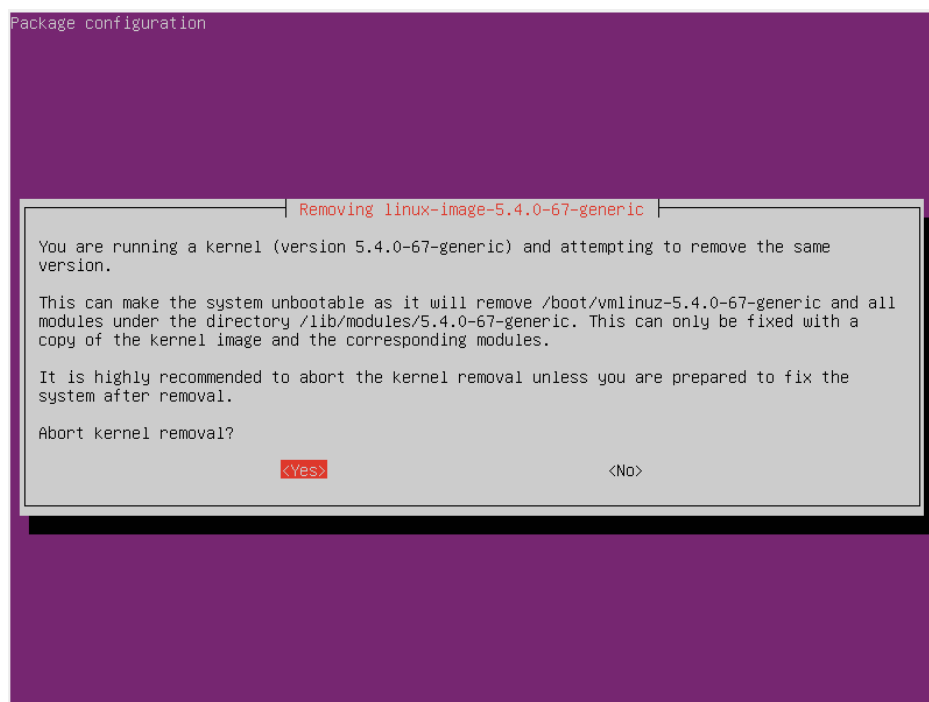
UbuntuServer

En UbuntuServer utilizaremos un servidor web Apache. Este escuchará peticiones en el puerto 80, pues es un servidor basado en HTTP. Si solo utilizáramos Apache no habría interacción con el usuario, pues solo se servirían archivos de texto, luego se requiere de más elementos, como una base de datos que se actualice dinámicamente o un lenguaje de programación para procesar información, normalmente interpretado. Pensando en esto último se creo la pila LAMP (Linux Apache MySQL PHP) que es un conjunto de herramientas para crear o gestionar servidores web más serio. Como sus siglas indican se instalan directamente base de datos y lenguaje de programación (la P puede ser también de Python), a parte del servidor.

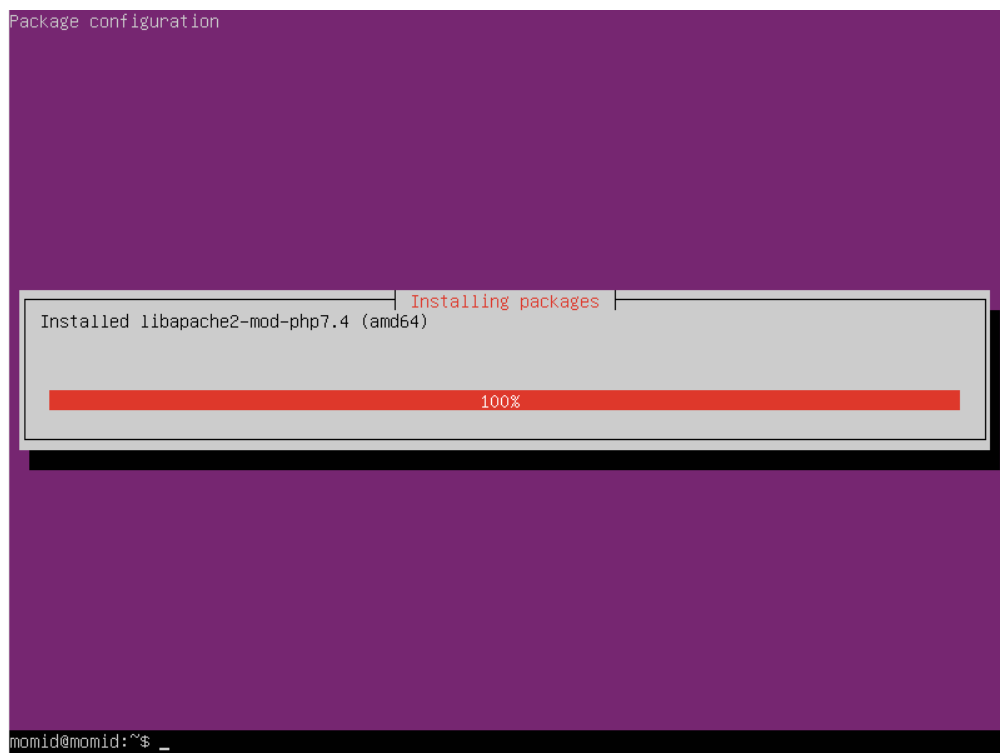
En primer lugar instalaremos *tasksel*, que nos proporciona una lista de las herramientas más populares de un servidor web. Lo normal es que no venga instalada, luego hacemos “*sudo apt install tasksel*” y una vez instalada la ejecutamos, también con sudo.



Nos saldrá algo parecido a esto, desmarcamos todo lo que haya marcado y dejamos marcado solamente LAMP server, y le damos a OK. Transcurrido un tiempo nos saldrá



Se puede ver en el mensaje que nos dice que estamos corriendo un kernel y intentando borrar la misma versión, pues LAMP te borra todo y luego instala el servidor. El sistema nos dice que se puede cargar el boot y que es altamente recomendable aborta, luego abortamos. Tras esto hacemos “*sudo apt-get update*”, pues se actualizan los repositorios y la versión de LAMP que nos coge no intentará machacar el kernel. Tras hacer el update repetimos los pasos anteriores y ya no debería salir el mensaje anterior.



Una vez terminada la instalación, comprobamos que apache se haya instalado bien y esté arrancado usando “*systemctl status apache2*”

```
momid@momid:~$ sudo systemctl status apache2
• apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2021-04-21 22:41:48 UTC; 2min 53s ago
    Docs: https://httpd.apache.org/docs/2.4/
  Main PID: 12237 (apache2)
    Tasks: 6 (limit: 4619)
  Memory: 10.3M
  CGroup: /system.slice/apache2.service
          └─12237 /usr/sbin/apache2 -k start
            └─12241 /usr/sbin/apache2 -k start
              └─12242 /usr/sbin/apache2 -k start
                └─12243 /usr/sbin/apache2 -k start
                  └─12244 /usr/sbin/apache2 -k start
                    └─12245 /usr/sbin/apache2 -k start

Apr 21 22:41:47 momid systemd[1]: Starting The Apache HTTP Server...
Apr 21 22:41:48 momid apachectl[12215]: AH00558: apache2: Could not reliably determine the server's s
Apr 21 22:41:48 momid systemd[1]: Started The Apache HTTP Server.
lines 1-18/18 (END)
```

Si bien no lo dice el servidor escuchará peticiones en el puerto 80. Hacemos lo mismo para probar que MySQL esté instalado.

```

momid@momid:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-04-21 22:41:37 UTC; 6min ago
 Main PID: 10933 (mysqld)
   Status: "Server is operational"
    Tasks: 37 (limit: 4619)
   Memory: 331.6M
   CGroup: /system.slice/mysql.service
           └─10933 /usr/sbin/mysqld

Apr 21 22:41:35 momid systemd[1]: Starting MySQL Community Server...
Apr 21 22:41:37 momid systemd[1]: Started MySQL Community Server.
momid@momid:~$ _

```

Probaremos también que php esté instalado entrando en el modo interactivo de php con el comando “*php -a*” y podemos probar un comando simple como “*echo('Hola Mundo');*”

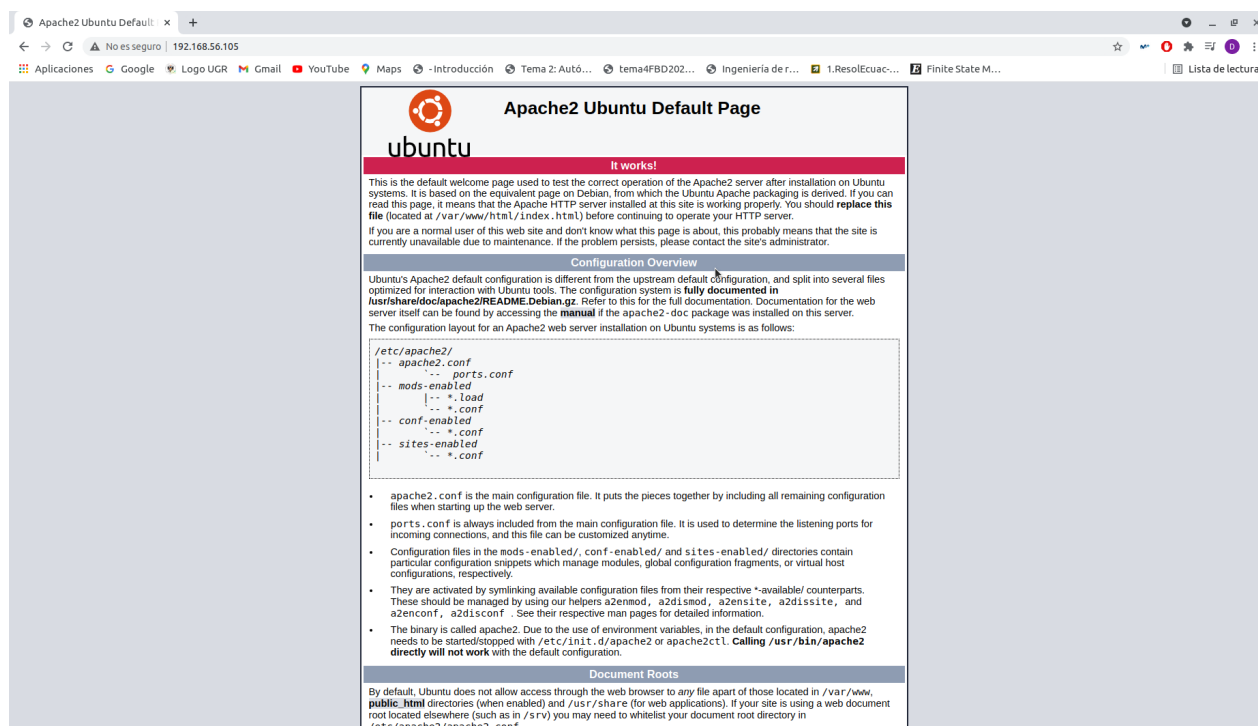
```

momid@momid:~$ php -a
Interactive mode enabled

php > echo('Hola Mundo');
Hola Mundo
php > exit
momid@momid:~$

```

Por último, veremos que apache ya está activo y funciona. Para comprobarlo cogemos el navegador de nuestro navegador y escribimos la IP de nuestra maquina UbuntuServer, en este caso 192.168.56.105, y nos saldrá la página de inicio predeterminada de apache.



Se puede ver que no es seguro, pues ya hemos dicho que apache trabaja con HTTP y por tanto la conexión no está cifrada, sino que es en texto plano y un atacante man-in-the-middle podría ver sin problemas los mensajes.

La página predeterminada se encuentra en nuestra máquina UbuntuServer en el directorio “*/var/www/html/index.html*”, y es una página estática, pues no va a cambiar en ningún momento. Para poder hacer las páginas dinámicas, hemos instalado MySQL y PHP.

CentOS

En CentOS haremos exactamente lo mismo, solo que en lugar de usar taskel para instalar el LAMP server lo haremos a mano, y además nos dará más problemas con los temas de puertos y demás. En primer lugar instalaremos apache, solo que CentOS tiene su propio apache y se instala con “*sudo yum install httpd*”. Una vez lo hayamos instalado comprobamos que esté activo, y si no lo está lo activamos.

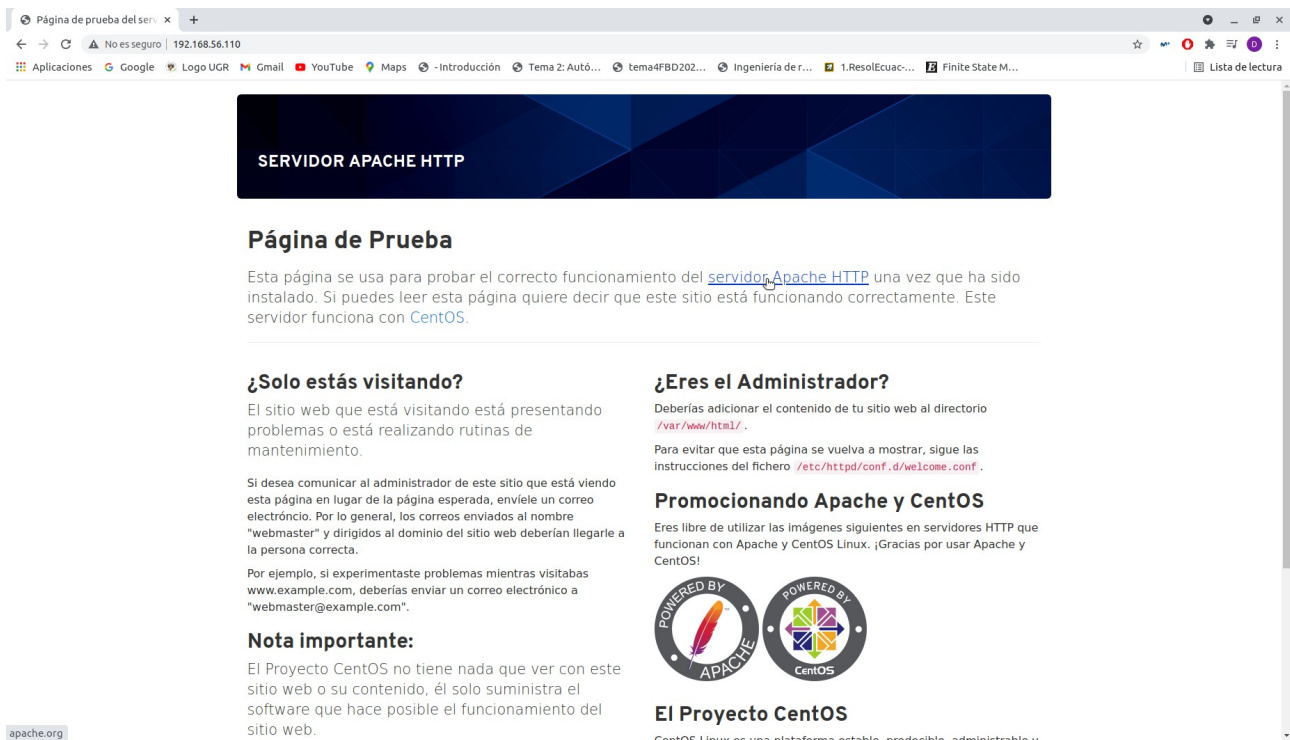
```
[momid@localhost ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
  Active: inactive (dead)
  Docs: man:httpd.service(8)
[momid@localhost ~]$ sudo systemctl start httpd
[momid@localhost ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
  Active: active (running) since Wed 2021-04-21 19:01:55 EDT; 1s ago
  Docs: man:httpd.service(8)
  Main PID: 23034 (httpd)
  Status: "Started, listening on: port 80"
  Tasks: 213 (limit: 23960)
  Memory: 25.0M
  CGroup: /system.slice/httpd.service
          └─23034 /usr/sbin/httpd -DFOREGROUND
            └─23035 /usr/sbin/httpd -DFOREGROUND
              └─23036 /usr/sbin/httpd -DFOREGROUND
                └─23037 /usr/sbin/httpd -DFOREGROUND
                  └─23038 /usr/sbin/httpd -DFOREGROUND

abr 21 19:01:55 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
abr 21 19:01:55 localhost.localdomain httpd[23034]: AH00558: httpd: Could not reliably determine th
abr 21 19:01:55 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
abr 21 19:01:55 localhost.localdomain httpd[23034]: Server configured, listening on: port 80
lines 1-19/19 (END)
```

Y podemos ver que nos indica que escucha peticiones en el puerto 80. Si ahora intentáramos acceder como en el caso anterior usando un navegador a la IP de la máquina no funcionará, pues el puerto está cerrado, luego lo abriremos. Para ello usamos “*sudo firewall-cmd --permanent --add-service=http*”. Observamos que a diferencia de como hacíamos para añadir el puerto 22022 en ssh esta vez usamos *--add-service=http*, pues *firewall-cmd* tiene una lista de puerto más comunes, y al poner *--add-service=http* nos abre el puerto 80, que es el predeterminado para http. Una vez añadido el puerto hay que recargar el firewall con “*sudo firewall-cmd --reload*”

```
[momid@localhost ~]$ sudo firewall-cmd --permanent --add-service=http
[sudo] password for momid:
sudo: firewall-cmd: command not found
[momid@localhost ~]$ sudo firewall-cmd --permanent --add-service=http
success
[momid@localhost ~]$ sudo firewall-cmd --reload
success
[momid@localhost ~]$ _
```

Tras esto ya podemos hacer la comprobación en el navegador.



Vemos que funciona, y nos incluye cierta información, como por ejemplo donde se encuentra el fichero html correspondiente a esta página.

Ahora instalaremos la base de datos y php. Para ello usamos “*sudo yum install mariadb*” que es el cliente de la base de datos que usaremos, como el servidor “*sudo yum install mariadb-server*”, que es parecido a mysql. Podemos comprobar que se haya instalado con *systemctl*.

```
[momid@localhost ~]$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:mysql(8)
           https://mariadb.com/kb/en/library/systemd/
[momid@localhost ~]$ sudo systemctl start mariadb
[momid@localhost ~]$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2021-04-21 19:11:38 EDT; 2s ago
     Docs: man:mysql(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 26452 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, status=0/SUCCESS)
   Process: 26317 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.service (code=exited, status=0/SUCCESS)
   Process: 26293 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited, status=0/SUCCESS)
  Main PID: 26420 (mysqld)
    Status: "Taking your SQL requests now..."
     Tasks: 30 (limit: 23960)
   Memory: 83.2M
   CGroup: /system.slice/mariadb.service
           └─26420 /usr/libexec/mysqld --basedir=/usr

abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: See the MariaDB Knowledgebase at
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: MySQL manual for more instructio
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: Please report any problems at ht
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: The latest information about Mar
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: You can find additional informat
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: http://dev.mysql.com
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: Consider joining MariaDB's stron
abr 21 19:11:38 localhost.localdomain mysql-prepare-db-dir[26317]: https://mariadb.org/get-involved/
abr 21 19:11:38 localhost.localdomain mysqld[26420]: 2021-04-21 19:11:38 0 [Note] /usr/libexec/mysq
abr 21 19:11:38 localhost.localdomain systemd[1]: Started MariaDB 10.3 database server.

lines 1-25/25 (END)
```

Normalmente para el servidor de mariadb suele traer añadidas una serie de tablas y usuarios por defecto que pueden ser vulnerabilidades. Además, debemos darle una contraseña al usuario root de mariadb, que por defecto no la trae. Para ello arrancamos un script con el siguiente comando *“mysql_secure_installation”* . Lo primero que nos pedirá será la contraseña de root, que como nos indica un poco más arriba si acabamos de instalar mariadb debemos dejar en blanco. Entonces nos dirá de establece contraseña al root, le damos a si y seguimos los pasos y de ahí en adelante seguimos como en la imagen

```
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[momid@localhost ~]$
```

Se puede ver que hemos eliminado el usuario anónimo, la base de datos test, deshabilitado el acceso al perfil root desde remoto y por último hemos recargado la tabla de privilegios para hacer efectivos los cambios.

Una vez hecho esto instalamos php con *“sudo yum install php”* y comprobamos que funcione entrando al modo interactivo de php, de la misma manera que en UbuntuServer.

```
[momid@localhost ~]$ php -a
Interactive shell

php >
```

Con esto ya tendríamos la pila LAMP, ahora nos encargaremos de hacer nuestra página dinámica. Para ello haremos un script en PHP que se conecta a la base de datos, y si se consigue conectar nos devuelve una cosa y si no lo consigue nos devuelve otra. Para ello usaremos un paquete de php con funciones prehechas, para lo que instalamos *“sudo yum install php-mysqld”*. Una vez instalado crearemos el php con *“sudo vi /var/www/html/index.php”*

```

<?php
$link = mysqli_connect('127.0.0.1:3306', 'root', 'ISE');

if (!$link){
    die('No se puede conectar');
}

echo('Conectado satisfactoriamente');

mysqli_close($link);_

phpinfo();

?>

```

La variable link contiene el resultado de conectarse a la base de datos. Los parámetros de la función mysqli_connect son la dirección IP donde se aloja la base de datos, así como el puerto, el usuario y la contraseña del usuario al que accedemos. Luego si no se conecta se muestra un mensaje de error y termina el script, mientras que si funciona muestra conectado satisfactoriamente, cierra el enlace y muestra información del sistema. Podemos probar que el script funciona con “*php /var/www/html/index.php*” y nos quedará algo como esto

```

$ _SERVER['TERM'] => linux
$ _SERVER['SHLVL'] => 1
$ _SERVER['XDG_SEAT'] => seat0
$ _SERVER['LOGNAME'] => momid
$ _SERVER['DBUS_SESSION_BUS_ADDRESS'] => unix:path=/run/user/1000/bus
$ _SERVER['XDG_RUNTIME_DIR'] => /run/user/1000
$ _SERVER['PATH'] => /home/momid/.local/bin:/home/momid/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
$ _SERVER['HISTSIZE'] => 1000
$ _SERVER['LESSOPEN'] => |! /usr/bin/lesspipe.sh %s
$ _SERVER['_'] => /usr/bin/php
$ _SERVER['PHP_SELF'] => /var/www/html/index.php
$ _SERVER['SCRIPT_NAME'] => /var/www/html/index.php
$ _SERVER['SCRIPT_FILENAME'] => /var/www/html/index.php
$ _SERVER['PATH_TRANSLATED'] => /var/www/html/index.php
$ _SERVER['DOCUMENT_ROOT'] =>
$ _SERVER['REQUEST_TIME_FLOAT'] => 1619047882.4947
$ _SERVER['REQUEST_TIME'] => 1619047882
$ _SERVER['argv'] => Array
(
    [0] => /var/www/html/index.php
)
$ _SERVER['argc'] => 1

PHP License
This program is free software; you can redistribute it and/or modify
it under the terms of the PHP License as published by the PHP Group
and included in the distribution in the file: LICENSE

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If you did not receive a copy of the PHP license, or have any
questions about PHP licensing, please contact license@php.net.
[momid@localhost ~]$

```

Para poder ver aplicados los cambios tenemos que indicar al servidor que index.php es el script principal. Para indicar esto último vamos al archivo “*sudo vi /etc/httpd/conf/httpd.conf*” y buscamos donde aparezca index.html y añadimos index.php

```

# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride None

#
# Controls who can get stuff from this server.
#
Require all granted
</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html, index.php
</IfModule>

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ".ht*">
    Require all denied
</Files>

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog "logs/error_log"
"/etc/httpd/conf/httpd.conf" 356L, 11910C written
[momid@localhost ~]$ _

```

Tras esto hacemos “*sudo systemctl restart httpd*” para reiniciar al servicio, pero veremos que sigue sin funcionar, pues entra en el caso de no se puede conectar, sin embargo al ejecutar el script desde dentro de la máquina funciona




Este error se debe a SEL. Podemos ver una serie de booleanos usando el comando “*getsebool -a*” que son flags que nos indica que nos permite hacer y que no el sistema de seguridad de Linux. Si a la lista total le filtramos las líneas en las que aparece httpd, vemos uno “*httpd_can_network_connect_db-->off*” que nos indica que no podemos conectarnos desde fuera del servidor a la base de datos. Lo que haremos será cambiar dicho flag, con el comando “*sudo setsebool -P httpd_can_network_connect_db on*”.

Una vez cambiado el flag ya podemos probar a recargar el navegador.


Conectado satisfactoriamente

PHP Version 7.2.24



System	Linux localhost.localdomain 4.18.0-193.el8.x86_64 #1 SMP Fri May 8 10:59:10 UTC 2020 x86_64
Build Date	Oct 22 2019 08:28:36
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bcmath.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-imagick.ini, /etc/php.d/20-ldap.ini, /etc/php.d/20-openssl.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tidy.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlrpc.ini, /etc/php.d/20-zip.ini, /etc/php.d/20-zlib.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718.NTS
PHP Extension Build	API20170718.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2*, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v3.2.0, Copyright (c) 1998-2019 Zend Technologies



Configuration

bz2

BZip2 Support	Enabled
Stream Wrapper support	compress.bzip2://

Con esto quedarían ambos servidores web hecho, tanto en CentOS como con Ubuntu.

Por último le daremos un poco de seguridad a ssh. Para ello usaremos la herramienta fail2ban que nos permite banea ips. Primero instalamos “*sudo yum install epel-release*” que es necesario para poder instalar fail2ban, y una vez instalada hacemos la instalación de fail2ban (“*yum install fail2ban*”). Fail2ban es un servicio luego con systemctl podemos comprobar si está activo, y sino lo está activarlo.

```
[momid@localhost ~]$ sudo systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:fail2ban(1)
[momid@localhost ~]$ sudo systemctl start fail2ban
[momid@localhost ~]$ sudo systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2021-04-21 19:58:33 EDT; 1s ago
     Docs: man:fail2ban(1)
  Process: 28293 ExecStartPre=/bin/mkdir -p /run/fail2ban (code=exited, status=0/SUCCESS)
 Main PID: 28294 (fail2ban-server)
    Tasks: 3 (limit: 23968)
   Memory: 11.1M
    CGroup: /system.slice/fail2ban.service
           └─28294 /usr/bin/python3.6 -s /usr/bin/fail2ban-server -xf start

abr 21 19:58:33 localhost.localdomain systemd[1]: Starting Fail2Ban Service...
abr 21 19:58:33 localhost.localdomain systemd[1]: Started Fail2Ban Service.
abr 21 19:58:33 localhost.localdomain fail2ban-server[28294]: Server ready
[momid@localhost ~]$ _
```

Para ver el estado actual de fail2ban se usa “*sudo fail2ban-client status*”

```
[momid@localhost ~]$ sudo fail2ban-client status
Status
|- Number of jail:      0
`- Jail list:
[momid@localhost ~]$
```

Vamos al archivo “*sudo vi /etc/fail2ban/jail.conf*”, para ver la configuración de las cárceles. En mayúscula nos indica el fichero que no se debe modificar, y que si queremos modificar el fichero usemos *jail.local*, pero dicho archivo no existe, luego usamos “*sudo cp -a /etc/fail2ban/jail.conf /etc/fail2ban/jail.local*” así hacemos una copia. Al reiniciar el servicio fail2ban todo cambio de *jail.local* se aplica a *jail.conf*.

A nosotros nos interesa el apartado de cárceles (JAILS). Lo primero que haremos será habilitar la cárcel de ssh, incluyendo en la cárcel de ssh una línea *enable=true*. Si se fallan cinco veces al entrar desde ssh se banea la IP del que haya fallado. Ahora si hacemos un *fail2ban-client status* vemos que ya tenemos una carcel activa.

```
[momid@localhost ~]$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:  sshd
[momid@localhost ~]$ _
```

Podemos probar que está activa conectándonos desde ssh desde nuestro ordenador. Para ello hay que asegurarse de que ssh tiene la autenticación por contraseña activa. Sin embargo, si intentamos cinco veces introduciendo una contraseña errónea no nos banea. Esto se debe a que el puerto que fail2ban está controlando es el habitual de ssh, pero nosotros tenemos configurado para que nuestra conexión por ssh sea en el puerto 22022. Para eso cambiaremos el puerto en el archivo de configuración de las celdas de fail2ban. Una vez arreglado el puerto si lo intentamos de nuevo 5 veces nos baneará la IP.

```
daniel@daniel-XPS-15-9570:~$ ssh momid@192.168.56.110 -p 22022
ssh: connect to host 192.168.56.110 port 22022: Connection refused
daniel@daniel-XPS-15-9570:~$
```

```
[momid@localhost ~]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 1
  |- Total banned: 1
  `-- Banned IP list: 192.168.56.1
[momid@localhost ~]$ _
```

Podemos comprobar las IP baneadas a un servicio con “*sudo fail2ban status <servicio>*” Para eliminar un baneo o bien esperamos que pase el tiempo establecido en el fichero de configuración, o usamos “*sudo fail2ban-client set sshd unbanip <dir_ip>*” y ahora podemos comprobar si nos ha desbaneado la IP con el *fail2ban-client status sshd*

```
[momid@localhost ~]$ sudo fail2ban-client set sshd unbanip 192.168.56.1
1
[momid@localhost ~]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 0
  |- Total banned: 1
  `-- Banned IP list:
[momid@localhost ~]$
```


y si intentamos entrar ahora si nos dejara

```
daniel@daniel-XPS-15-9570:~$ ssh momid@192.168.56.110 -p 22022
momid@192.168.56.110's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last failed login: Wed Apr 21 20:02:48 EDT 2021 from 192.168.56.1 on ssh:notty
There were 6 failed login attempts since the last successful login.
Last login: Wed Apr 21 20:02:16 2021 from 192.168.56.1
[momid@localhost ~]$
```

Con esto concluiría la práctica 2.