Ingeniería de Servidores (2015-2016)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Práctica 2

Marta Gómez Macías

11 de septiembre de 2016

Índice

1. Liste los argumentos de `yum` necesarios para instalar, buscar y eliminar paquetes **4**

2. ¿Qué ha de hacer para que yum pueda tener acceso a internet¿ ¿Cómo añadimos un nuevo repositorio? **4**

3. Indique el comando para buscar un paquete en un repositorio y el correspondiente para instalarlo **5**

4. Indique qué ha modificado para que `apt` pueda acceder a los servidores de paquetes a través del proxy. ¿Cómo añadimos un nuevo repositorio? **5**

5. ¿Qué diferencua hay entre telnet y shh? **6**

6. ¿Para qué sirve la opción `-X`? Ejecute remotamente, es decir, desde la máquina anfitriona o desde la otra máquina virtual, el comando `gedit` en una sesión abierta con `ssh`. ¿Qué ocurre? **6**

7. Muestre la secuencia de comandos y las modificaciones a los archivos correspondientes para permitir acceder a la consola remota sin introducir la contraseña. **6**

7.1. Creando la clave pública y privada en el cliente 6

7.2. Transfiriendo la clave pública creada al servidor 8

8. ¿Qué archivo es el que contiene la configuración de sshd? ¿Qué parámetro hay que modificar para evitar que el usuario root acceda? Cambie el puerto por defecto y compruebe que puede acceder **9**

9. Indique si es necesario reiniciar el servicio ¿Cómo se reinicia un servicio en Ubuntu? ¿y en CentOS? Muestra la secuencia de comandos para hacerlo. **10**

9.1. Ubuntu 10

9.2. CentOS 10

10. Muestre los comandos que ha utilizado en Ubuntu Server y en CentOS (aunque en este último puede utilizar la GUI, en tal caso, relice capturas de pantalla) para instalar un servidor web LAMP **11**

10.1. Ubuntu Server 11

10.2. CentOS 12

11. Enumere otros servidores web y las páginas de sus proyectos (mínimo 3 sin considerar Apache, IIS ni nginx) **14**

11.1. IBM HTTP Server 14

11.2. Jetty 14

11.3. WakandaDB 14

12. Compruebe que el servicio IIS instalado está funcionando accediendo a la máquina virtual a través de la anfitriona. **15**

13. Muestre un ejemplo de uso del comando `patch` 17

13.1. Probando nuestro propio parche 17

14. Realice la instalación de `webmin` y pruebe a modificar algún parámetro de algún servicio. Muestre las capturas de pantalla pertinentes así como el proceso de instalación 18

14.1. Instalación 18

14.2. Cambiando algún parámetro de Webmin 19

15. Instale `phpMyAdmin`, indique cómo lo ha realizado y muestre algunas capturas de pantalla. Configure PHP para poder importar BDs mayores de 8MiB (límite por defecto). Indique cómo ha realizado el proceso y muestre capturas de pantalla 21

16. Visite la web de *ISPConfig* y pruebe la demo que ofrecen realizando capturas de pantalla y comentando qué está realizando. 22

17. Ejecute los ejemplos de `find`, `grep` y escriba el script que haga uso de `sed` para cambiar la configuración de ssh y reiniciar el serivicio 24

17.1. `grep` 24

17.2. `find` 24

17.3. `sed` 25

18. Escriba el script para cambiar el acceso a ssh usando Python 25

18.1. Servidores conocidos 26

18.2. Haciendo el *Playbook* 26

19. Abra una consola de Powershell y pruebe a parar un programa en ejecución (p.ej), realice capturas de pantalla y comente lo que muestra 27

20. Cuestiones Opcionales 27

20.1. ¿Qué gestores utiliza OpenSuse? 27

20.2. Instale y pruebe terminator. Con screen, pruebe su funcionamiento dejando sesiones ssh abiertas en el servidor y recuperándolas posteriormente 28

20.3. Instale el servicio `fail2ban` y pruebe su funcionamiento 29

20.6. Muestre un ejemplo de uso para `awk` 30

Índice de figuras

4.1. Output de `add-apt-repository -h` 6

6.1. Ejecutando `gedit` desde una sesión `ssh` 7

7.1. Comprobando que no tenemos ninguna clave generada anteriormente 7

7.2. Output tras generar nuestra clave 8

7.3. Arrancando el agente ssh 8

7.4. Añadiendo la clave generada al agente ssh 8

7.5. Copiando la clave pública al servidor 9

7.6. Comprobando que todo ha ido bien . . . 9
9.1. Error al conectarnos al servicio SHH tras cambiar el puerto . . . 10
9.2. Tras abrir el puerto, nos podemos loguear . . . 10
10.1. Configurando la contraseña del administrador de la base de datos . . . 11
10.2. Página de prueba de nuestro servidor apache . . . 12
10.3. Parte del output obtenido al buscar módulos de PHP . . . 13
10.4. Parte del output obtenido al consultar un módulo de PHP . . . 13
12.1. Comprobación de que el servidor IIS está activo . . . 15
12.2. Añadiendo una red sólo-anfitrión a Virtual Box . . . 16
12.3. Añadiendo una interfaz de red sólo-anfitrión a la máquina virtual . . . 16
12.4. Salida del comando `ip addr`, como se ve la red no tiene asignada ninguna IP . . . 16
12.5. Salida del comando `ip addr` tras asignarle a la red una IP . . . 16
12.6. Salida que nos muestra `ipconfig` sobre el adaptador de red sólo-anfitrión . . . 17
12.7. Comprobación de que podemos acceder al servidor IIS instalado a través de la máquina anfitriona . . . 17
14.1. Error al acceder a `http://localhost:10000/` . . . 19
14.2. Éxito al acceder a `https://localhost:10000/` . . . 19
14.3. Pantalla al loguearnos en webmin . . . 20
14.4. Menú del servicio SSH . . . 20
14.5. Editando el archivo de configuración del demonio SSH . . . 20
14.6. No podemos loguearnos con el usuario root pero sí con otros usuarios . . . 21
15.1. Eligiendo el servidor que ejecutará `phpMyAdmin` . . . 21
15.2. Configuración de la base de datos de `phpMyAdmin` . . . 22
15.3. Tras cambiar los parámetros indicados, hemos podido importar una base de datos de 35MB . . . 22
16.1. Pantalla al acceder a ISPConfig como administrador . . . 23
16.2. Pantalla al acceder a ISPConfig como cliente . . . 23
16.3. Pantalla al acceder a ISPConfig como *reseller* . . . 23
16.4. Direcciones IP . . . 23
17.1. Parte del output de `ps -Af` . . . 24
17.2. Parte del output de `ps -Af | grep yakuake` . . . 24
17.3. Ejemplo de ejecución del script . . . 25
18.1. Fichero de host por defecto de Ansible . . . 26
18.2. Haciendo `ping` a todos los host guardados . . . 26
18.3. Script para cambiar un parámetro de SSH . . . 27
19.1. Deteniendo la ejecución de Internet Explorer . . . 28
20.1. Prueba con terminator . . . 28
20.2. Listando las sesiones de `screen` actualmente abiertas . . . 29
20.3. Haciendo un split en la sesión de screen actual . . . 29
20.4. Ejemplo de uso del comando `awk` . . . 31

1. Liste los argumentos de yum necesarios para instalar, buscar y eliminar paquetes

Si miramos en [33], nos vienen los comandos que podemos utilizar según lo que queramos hacer. En concreto los comandos que se piden en el enunciado son:

— `yum install`:

```
yum install package1 [package2] [...]
```

Se usa para instalar la última versión de un paquete o grupo de paquetes asegurándose de que se cumplen todas las dependencias. Tiene varias peculiaridades:

- Si empieza por el carácter @ busca grupos de paquetes.
- Si el nombre es un archivo, instala un paquete local, como si llamásemos al comando `yum localinstall`

— `yum erase` o `yum remove`:

```
yum remove package1 [package2] [...]
```

Se usa para borrar los paquetes que especifiquemos y sus dependencias. La operación sobre los grupos de archivos funciona igual que en el comando `install`.

— `yum search`

```
yum search string1 [string2] [...]
```

Se usa para buscar paquetes cuando no estamos seguros de cómo se llaman. Por defecto busca nombres de paquetes y sumarios, pero también podemos buscar por descripción y URL.

2. ¿Qué ha de hacer para que yum pueda tener acceso a internet¿ ¿Cómo añadimos un nuevo repositorio?

Tal y como se especifica en [7], debemos modificar el archivo */etc/yum.conf* y añadir la siguiente línea:

Configuración del proxy en yum

```
# The proxy server - proxy server:port number
proxy=stargate.ugr.es:312
```

Para añadir un nuevo repositorio, según [13], debemos ejecutar el siguiente comando:

Añadir un nuevo repositorio en yum

```
yum-config-manager --add-repo repository_url
```

Donde `repository_url` es un enlace al archivo **.repo**.

Una vez hecho esto, para activar el respositorio debemos ejecutar el siguiente comando:

Activar un repositorio en yum

```
yum-config-manager --enable repository
```

Donde `respository` es el ID del repositorio (si no lo sabemos, usamos `yum repolist all` para listar los IDs disponibles). También podemos usar expresiones globales para activar todos los respositorios que coincidan con dicha expresión global.

Este comando tiene la misma sintaxis que el que se usa para desactivar un repositorio:

Desactivar un repositorio en yum

```
yum-config-manager --disable repository
```

3. Indique el comando para buscar un paquete en un repositorio y el correspondiente para instalarlo

En [25], nos citan las distintas páginas de manual que tienen que ver con `apt`. En concreto, las que nos interesan son:

— `apt-chache`: según [26], usando el comando `search` busca en todas las listas de repositorios disponibles para el patrón dado. Éste puede contener expresiones regulares. Por defecto nos muestra el nombre del paquete y una pequeña descripción, pero podemos modificar este comportamiento:

- `-full`: muestra una descripción completa del paquete, el output es el mismo que con el comando `show`.
- `-names-only`: sólo busca paquetes por su nombre.

— `apt-get`: según [27], usando el comando `install` seguido del nombre de uno o más paquetes, instalará los paquetes dados. Si el nombre del paquete lleva delante el símbolo − eliminará el paquete si está instalado, y si lleva el signo +, lo instalará. También podemos seleccionar una versión concreta de un paquete con el símbolo = seguido de la versión deseada. Por último, podemos usar expresiones regulares para instalar o eliminar todos los paquetes que coincidan.

4. Indique qué ha modificado para que apt pueda acceder a los servidores de paquetes a través del proxy. ¿Cómo añadimos un nuevo repositorio?

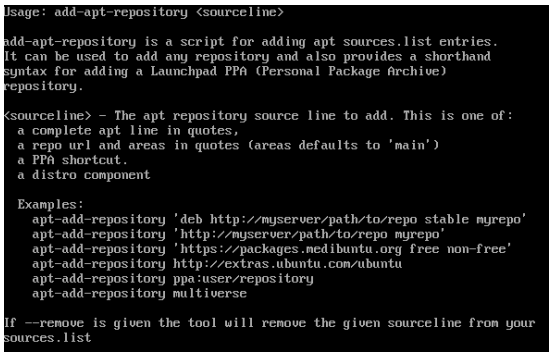
Tal y como se explica en [2], para acceder a un proxy por HTTP debemos añadir la siguiente línea o bien al archivo */etc/apt/apt.conf* ([28]) o bien al */etc/apt/apt.conf.d/proxy*:

Configurar un servidor proxy

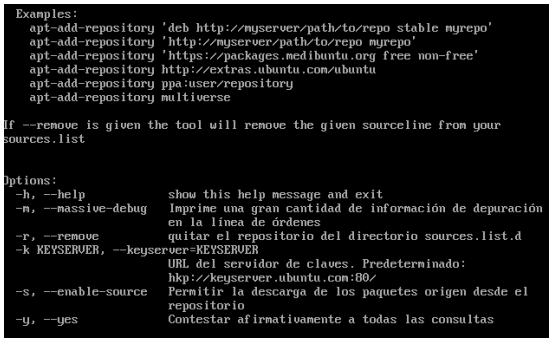
```
Acquire::http::Proxy "http://stargate.ugr.es:312";
```

Si no existen los archivos, creamos uno de ellos.

En [17], nos explican el comando `add-apt-repository`, con el cual podemos añadir un repositorio a apt. En la Figura 4.1 vemos el output al ejecutar el comando con la opción `-h`. Así, podemos saber las distintas opciones con las que podemos ejecutarlo.



(a) Primera parte del output de `add-apt-repository -h`



(b) Segunda parte del output de `add-apt-repository -h`

Figura 4.1: Output de `add-apt-repository -h`

5. ¿Qué diferencua hay entre telnet y shh?

Tal y como se explica en el primer párrafo de [22], telnet envía las contraseñas sin encriptar mientras que SSH sí. Pero no sólo las contraseñas sino todo el tráfico.

6. ¿Para qué sirve la opción -X? Ejecute remotamente, es decir, desde la máquina anfitriona o desde la otra máquina virtual, el comando `gedit` en una sesión abierta con `ssh`. ¿Qué ocurre?

Consultando [31] podemos ver las distintas opciones que acepta con una explicación, en concreto, la opción `-X` nos dice que sirve para activar el *X11 forwading*[1]

Al ejecutar `gedit` en una sesión `ssh` desde la máquina remota, se nos abre una ventana de gedit normal, pero que se está ejecutando en la máquina anfitriona. Tal y como se ve en la Figura 6.1, tenemos una ventana de gedit ejecutándose, pero al hacer `ps` en la máquina local no encuentra ningún proceso `gedit` activo. En cambio, al hacer `ps` en la sesión remota, sí.

7. Muestre la secuencia de comandos y las modificaciones a los archivos correspondientes para permitir acceder a la consola remota sin introducir la contraseña.

Para crear la clave pública y privada en el cliente, se han seguido los pasos indicados en [12]. Para transferir la clave pública creada al servidor, se han seguido los pasos indicados en [35].

7.1. Creando la clave pública y privada en el cliente

En primer lugar, comprobamos que no tenemos ninguna clave generada en nuestro ordenador, para ello listamos el contenido del directorio de ssh:

[1] Según [5], el X11 forwading es el sistema de ventanas X que nos permite ejecutar aplicaciones en un servidor UNIX como si estuviésemos ejecutando aplicaciones en Windows pudiendo incluso usar el ratón en éstas.

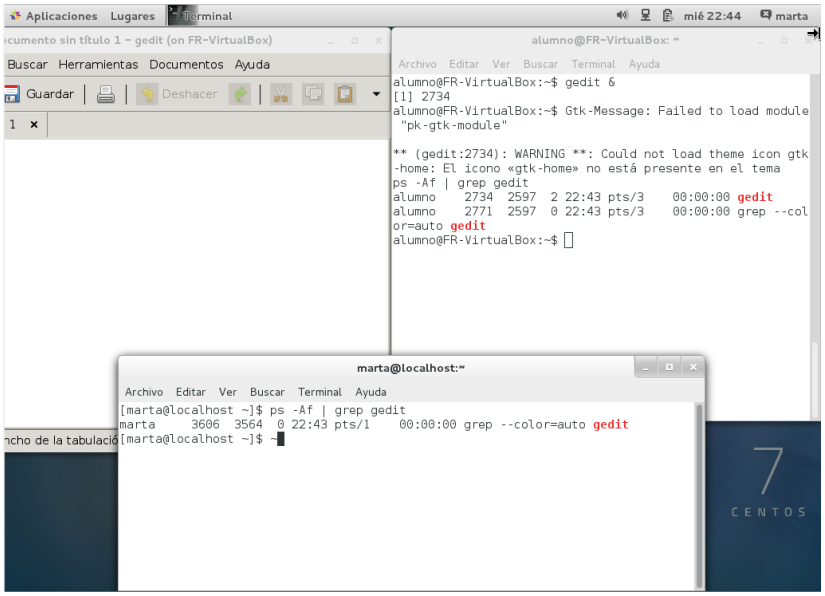


Figura 6.1: Ejecutando `gedit` desde una sesión `ssh`

Lista los archivos del directorio .ssh

```
ls -al ~/.ssh
```

En nuestro caso, como se aprecia en la Figura 7.1 no tenemos ninguna clave generada.

```
[marta@localhost ~]$ ls -al ~/.ssh
total 8
drwx------.  2 marta marta   24 oct 28 17:02 .
drwx------. 15 marta marta 4096 oct 29 09:39 ..
-rw-r--r--.  1 marta marta  340 oct 28 22:40 known_hosts
```

Figura 7.1: Comprobando que no tenemos ninguna clave generada anteriormente

Como no tenemos ninguna clave, pasamos a generarla usando nuestro nombre como etiqueta:

Crea una nueva clave ssh

```
ssh-keygen -t rsa -b 4096 -C "marta"
```

tras esto, nos pedirá que elijamos un archivo en el que guradar la clave, se recomienda dejarlo por defecto pulsando enter. Por último, nos pide introducir una clave, si pulsamos enter no pondrá ninguna clave al archivo pero lo recomendable es ponerle una.

Así, finalmente nos dará nuestro *fingerprint* de nuestra clave SSH. El output de este comando es el que se ve en la Figura 7.2.

Por último, añadimos nuestra clave ssh al agente de ssh. Para ello, en primer lugar nos aseguramos de que está activado:

Arrancamos el agente ssh

```
eval "$(ssh-agent -s)"
```

Aunque en mi caso el comando `eval` no funcionó y tuve que hacer sólo `ssh-agent -s`, tal y como se ve en la Figura 7.3.

Por último, añadimos la clave al agente ssh:

```
[marta@localhost ~]$ ssh-keygen -t rsa -b 4096 -C "marta"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/marta/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/marta/.ssh/id_rsa.
Your public key has been saved in /home/marta/.ssh/id_rsa.pub.
The key fingerprint is:
4a:d6:2f:0a:85:c9:08:ba:ea:8c:3e:ab:c0:af:0f:87 marta
The key's randomart image is:
+--[ RSA 4096]----+
|                 |
|                 |
|.                |
|.. o o .         |
|. . + + S        |
|...  + . .       |
|oE .. . . .      |
|=.+  . . .       |
|B*=o  .          |
+-----------------+
[marta@localhost ~]$
```

Figura 7.2: Output tras generar nuestra clave

```
[marta@localhost ~]$ ssh-agent -s
SSH_AUTH_SOCK=/tmp/ssh-YrMKye7eViDQ/agent.4290; export SSH_AUTH_SOCK;
SSH_AGENT_PID=4291; export SSH_AGENT_PID;
echo Agent pid 4291;
[marta@localhost ~]$
```

Figura 7.3: Arrancando el agente ssh

Añade la clave generada al agente

```
ssh-add ~/.ssh/id_rsa
```

Tras hacer esto nos da un mensaje de que todo ha ido bien, tal y como se ve en la Figura 7.4.

```
[marta@localhost ~]$ ssh-add ~/.ssh/id_rsa
Identity added: /home/marta/.ssh/id_rsa (/home/marta/.ssh/id_rsa)
[marta@localhost ~]$
```

Figura 7.4: Añadiendo la clave generada al agente ssh

7.2. Transfiriendo la clave pública creada al servidor

En la máquina desde la que queremos acceder, usamos el siguiente comando para transferir la clave:

Copiando la clave pública al servidor

```
ssh-copy-id <username>@<host>
```

Donde `<username>` y `<host>` deben ser reemplazados por el nombre de usuario correspondiente y la IP de la máquina a la que le estás transfiriendo tu clave.

Tras esto nos pedirá la contraseña del usuario y por último nos mostrará un mensaje de éxito tal y como se ve en la Figura 7.5.

Por último, comprobamos que hemos copiado bien la clave accediendo por ssh y comprobando que no nos pide contraseña, tal y como se ve en la Figura 7.6.

```
[marta@localhost ~]$ ssh-copy-id marta@10.0.2.9
/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 prompt
ed now it is to install the new keys
marta@10.0.2.9's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'marta@10.0.2.9'"
and check to make sure that only the key(s) you wanted were added.

[marta@localhost ~]$
```

Figura 7.5: Copiando la clave pública al servidor

```
[marta@localhost ~]$ ssh marta@10.0.2.9
Último inicio de sesión: jue oct 29 10:21:58 CET 2015 en tty1
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Oct 29 10:34:34 CET 2015

  System load:    0.0             Processes:           108
  Usage of /home: 0.7% of 451MB   Users logged in:     1
  Memory usage:   7%              IP address for eth0: 10.0.2.9
  Swap usage:     0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

marta@ISE:~$
```

Figura 7.6: Comprobando que todo ha ido bien

8. ¿Qué archivo es el que contiene la configuración de sshd? ¿Qué parámetro hay que modificar para evitar que el usuario root acceda? Cambie el puerto por defecto y compruebe que puede acceder

Según [32], el archivo de configuración del demonio de ssh es */etc/ssh/sshd_ config*. Para que el root no pueda acceder, debemos modificar el parámetro **PermitRootLogin**. Este parámetro puede tomar varios valores:

♡ **yes**: el valor por defecto.

♡ **without-password**: se desactiva la autentificación por contraseña del root.

♡ **forced-commands-only**: se permitirá el login con clave pública pero sólo si la opción *command* se especifica. Cualquier otro método de autentificación queda desactivado para el root.

♡ **no**: no permite que root se loguee.

en nuestro caso, debemos modificarlo por el valor **no**.

Para cambiar el puerto por defecto, en el mismo archivo de configuración debemos cambiar el valor del parámetro **Port**. Tras esto, intentamos conectarnos desde la máquina cliente y la conexión se realiza con éxito (en el puerto 22).

9. Indique si es necesario reiniciar el servicio ¿Cómo se reinicia un servicio en Ubuntu? ¿y en CentOS? Muestra la secuencia de comandos para hacerlo.

9.1. Ubuntu

Tal y como se indica en [34], tras hacer algún cambio en la configuración de SSH debemos reiniciar el servicio con el comando

Reiniciando el servicio SSH

```
sudo restart ssh
```

Una vez hecho esto, intentamos conectarnos de nuevo desde el cliente y nos da el error de la Figura 9.1.

```
[marta@localhost ~]$ ssh marta@10.0.2.9 -p 220
ssh: connect to host 10.0.2.9 port 220: Connection refused
```

Figura 9.1: Error al conectarnos al servicio SHH tras cambiar el puerto

Es decir, debemos abrir el nuevo puerto asignado al servicio SSH:

Abriendo el puerto 220

```
sudo iptables -A INPUT -p tcp -i eth0 --dport 220 -j ACCEPT
```

Tras esto, intentamos loguearnos y esta vez, lo hacemos con éxito tal y como se ve en la Figura 9.2.

```
[marta@localhost ~]$ ssh marta@10.0.2.9 -p 220
Último inicio de sesión: jue oct 29 11:38:30 CET 2015 de 10.0.2.10 en pts/0
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Oct 29 11:47:41 CET 2015

  System load:    0.0             Processes:           108
  Usage of /home: 0.7% of 451MB   Users logged in:     1
  Memory usage:   8%              IP address for eth0: 10.0.2.9
  Swap usage:     0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

marta@ISE:~$
```

Figura 9.2: Tras abrir el puerto, nos podemos loguear

9.2. CentOS

Para reiniciar un servicio en CentOS, tal y como se indica en el capítulo *Managing Services With Systemd* de [14], se debe usar el comando `systemctl`. En nuestro caso, para reiniciar el servicio SSH usaremos el comando:

Reiniciando el servicio SSH

```
sudo systemctl restart sshd
```

10. Muestre los comandos que ha utilizado en Ubuntu Server y en CentOS (aunque en este último puede utilizar la GUI, en tal caso, relice capturas de pantalla) para instalar un servidor web LAMP

10.1. Ubuntu Server

Siguiendo los pasos especificados en [21] y partiendo de una instalación de Ubuntu, seguimos los siguientes pasos:

1. En primer lugar instalamos el servidor Apache:

   Instalando el servidor web Apache
   ```
   sudo apt-get install apache2
   ```

   Como estamos usando una interfaz de consola, no podemos comprobar que se ha configurado bien el servidor visitando `http://10.0.2.9`.

2. En segundo lugar, instalamos MySQL:

   Instalando MySQL
   ```
   sudo apt-get install mysql-server php5-mysql
   ```

   durante la instalación nos pedirá una contraseña para el administrador de la base de datos, tal y como se ve en la Figura 10.1.

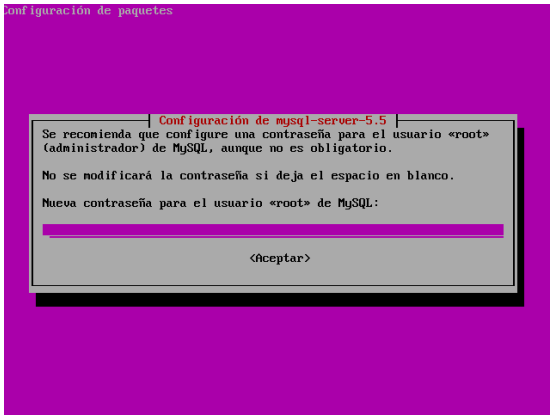


Figura 10.1: Configurando la contraseña del administrador de la base de datos

3. Tras la instalación, debemos decirle al gestor de base de datos que cree su propia estructura de directorios donde guardar su información:

   Creando la estructura de directorios de la base de datos
   ```
   sudo mysql_install_db
   ```

   También podemos hacer una instalación interactiva y personalizada para quitar algunas opciones por defecto algo peligrosas con:

   Instalación manual de la base de datos
   ```
   sudo mysql_secure_installation
   ```

   En la cual seguiremos los siguientes pasos:

1. Nos pedirá hacer login con el root, si no tenemos contraseña nos dirá que establezcamos una.
2. En segundo lugar nos dirá si eliminamos los usuarios anónimos de la base de datos.
3. En tercer lugar nos dirá si desactivamos el acceso root remoto.
4. Después, nos dirá si se elimina la base de datos por defecto *test* con la que viene.
5. Y por último, nos dirá si se recarga la tabla de privilegios.

4. Tras esto, instalamos PHP:

Instalando PHP
```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

10.2. CentOS

Siguiendo los pasos indicados en [20], los comandos para instalar un servidor LAMP han sido:

1. En primer lugar instalamos Apache. Para ello, ejecutamos el siguiente comando de terminal

Instalación de Apache
```
sudo yum install httpd
```

2. Una vez instalado, empezamos a ejecutarlo:

Arranque del servidor
```
sudo service httpd start
```

3. Para comprobar que funciona, metemos en nuestro navegador `http://ip_de_nuestro_servidor` y nos deberá aparecer Íla página que se ve en la Figura 10.2.

Probando si nuestro servidor ha sido instalado correctamente
```
firefox http://10.0.2.10
```

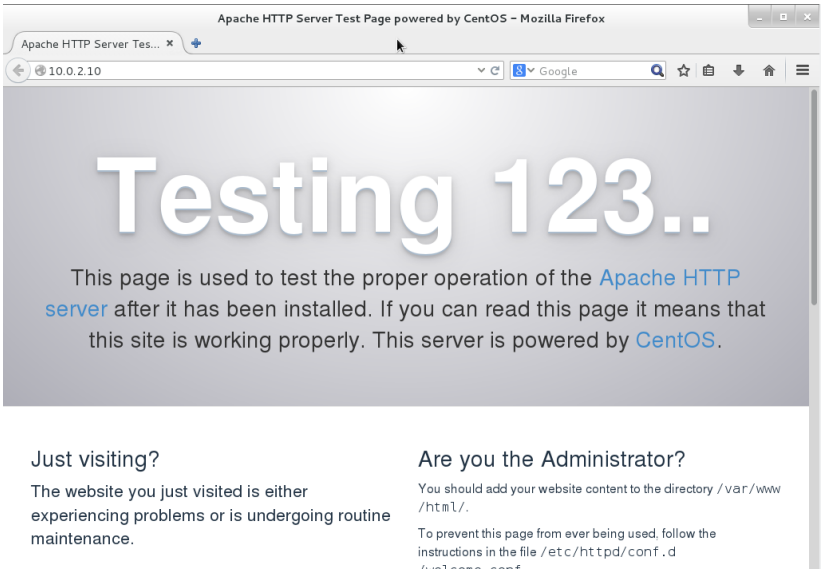


Figura 10.2: Página de prueba de nuestro servidor apache

4. Una vez listo Apache, pasamos a instalar MySQL, que en este caso ser MariaDB pues MySQL no se encuentra en los repositorios de CentOS 7. Para ello ejecutamos el siguiente comando:

Instalando el servidor de base de datos

```
sudo yum install mariadb-server mariadb
```

5. Una vez instalado pasamos a activar el servicio con el siguiente comando

Arrancando el servicio de base de datos

```
sudo service mariadb start
```

6. Para establecer una contraseña root (entre otras cosas) para la base de datos lo hacemos como si de un servidor MySQL se tratase ([18]).

Configurando manualmente la base de datos

```
mysql_secure_installation
```

Así, tendremos que seguir los mismos pasos que seguimos en Ubuntu Server al instalar MySQL.

7. Después de dejar listo el servidor de base de datos, pasamos a instalar PHP. Para ello, ejecutamos el siguiente comando:

Instalando PHP

```
sudo yum install php php-mysql
```

8. En [20] se explica cómo buscar (Figura 10.3), instalar y consultar (Figura 10.4) módulos de PHP en yum para instalar en nuestro servidor:

Buscando módulos de PHP

```
yum search php-
```

```
============================== N/S matched: php- ==============================
php-bcmath.x86_64 : A module for PHP applications for using the bcmath library
php-cli.x86_64 : Command-line interface for PHP
php-common.x86_64 : Common files for PHP
php-dba.x86_64 : A database abstraction layer module for PHP applications
php-devel.x86_64 : Files needed for building PHP extensions
php-embedded.x86_64 : PHP library for embedding in applications
php-enchant.x86_64 : Enchant spelling extension for PHP applications
php-fpm.x86_64 : PHP FastCGI Process Manager
php-gd.x86_64 : A module for PHP applications for using the gd graphics library
php-intl.x86_64 : Internationalization extension for PHP applications
php-ldap.x86_64 : A module for PHP applications that use LDAP
php-mbstring.x86_64 : A module for PHP applications which need multi-byte string
                    : handling
php-mysql.x86_64 : A module for PHP applications that use MySQL databases
php-mysqlnd.x86_64 : A module for PHP applications that use MySQL databases
php-odbc.x86_64 : A module for PHP applications that use ODBC databases
php-pdo.x86_64 : A database access abstraction module for PHP applications
php-pear.noarch : PHP Extension and Application Repository framework
php-pecl-memcache.x86_64 : Extension to work with the Memcached caching daemon
php-pgsql.x86_64 : A PostgreSQL database module for PHP
php-process.x86_64 : Modules for PHP script using system process interfaces
```

Figura 10.3: Parte del output obtenido al buscar módulos de PHP

Consultando información de un módulo concreto

```
yum info nombre_del_modulo
```

```
[marta@localhost ~]$ yum info php-fpm.x86_64
Complementos cargados:fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: sunsite.rediris.es
 * extras: sunsite.rediris.es
 * updates: sunsite.rediris.es
Paquetes disponibles
Nombre        : php-fpm
Arquitectura        : x86_64
Versión     : 5.4.16
Lanzamiento     : 36.el7_1
Tamaño        : 1.4 M
Repositorio        : updates/7/x86_64
Resumen     : PHP FastCGI Process Manager
URL         : http://www.php.net/
Licencia     : PHP and Zend and BSD
Descripción :PHP-FPM (FastCGI Process Manager) is an alternative PHP FastCGI
            : implementation with some additional features useful for sites of
            : any size, especially busier sites.

[marta@localhost ~]$
```

Figura 10.4: Parte del output obtenido al consultar un módulo de PHP

9. Por último, hacemos que al iniciar nuestro servidor, se inicie automáticamente el servidor LAMP (PHP se inicia automáticamente junto a Apache):

```
Automatizando el inicio de Apache y PHP al arrancar el sistema
sudo chkconfig httpd on
```

```
Automatizando el inicio de MariaDB al arrancar el sistema
sudo chkconfig mariadb on
```

11. Enumere otros servidores web y las páginas de sus proyectos (mínimo 3 sin considerar Apache, IIS ni nginx)

11.1. IBM HTTP Server

En [15] se describe este servidor web. Se incluye en un pack junto a otros productos de IBM tales como *IBM WebSphere®Application Server*. Se basa en el servidor HTTP de Apache y proporciona características tales como:

★ El servidor puede administrarse remotamente

★ Soporte para asegurar transacciones HTTPS

★ Puede autenticar peticiones web

11.2. Jetty

En [8] se describe este servidor web. Es un servidor web que soporta HTTP/2, WebSocket, OSGi, JMX, JNDI, JAAS, etc. Actualmente se recomienda usar la versión 9 que es la última pero también están disponibles para descargar la versión 7 y la 8.

El proyecto está alojado en la Eclipse Foundation y algunas de sus características son:

★ Viene muy equipado y está hecho basándose en los estándares

★ Es de código abierto y se puede usar comercialmente

★ Es muy flexible y extendible

★ Es escalable

11.3. WakandaDB

En [36] se describe este servidor web. Es un servidor con un gestor de base de datos de código abierto con un servidor HTTP que proporciona una interfaz REST completa.

Tanto el esquema de la base de datos, como el procesamiento del lado del servidor y el procesamiento de peticiones está todo hecho en JavaScript.

El servidor incluye las siguientes características (entre otras):

★ Conexión segura (Soporte SSL)

★ Soporte para IPV6

★ Conexión *Keep-alive*

Figura 12.1: Comprobación de que el servidor IIS está activo

★ Compresión de texto

★ Logs del acceso a la base de datos

★ Logs de la actividad web

12. Compruebe que el servicio IIS instalado está funcionando accediendo a la máquina virtual a través de la anfitriona.

En primer lugar, comprobamos en la máquina virtual que el servidor IIS está instalado. Para ello, accedemos a internet explorer y escribimos `http://localhost` en la barra de dirección. Debemos obtener el resultado que se en la Figura 12.1

Tras esto, pasamos a configurar una red *Host-only* en la máquina virtual para poder comunicarla con la anfitriona. Para ello seguimos los siguientes pasos:

1. En Virtual Box nos vamos a **Archivo** > **Preferencias...**. Una vez ahí seleccionamos **Red** en el menú de la izquierda y dentro de Red, seleccionamos la pestaña **Redes sólo-anfitrión**. Después, pulsamos el botón de la derecha para agregar una red sólo anfitrión. Al finalizar este paso tendremos que tenerlo todo como se ve en la Figura 12.2.

   Si por algún casual nos encontramos con un error a la hora de crear la red sólo-anfitrión, será necesario cerrar Virtual Box y recargar unos módulos necesarios con el comando:

   Cargando los módulos que faltan en Virtual Box
   ```
   sudo vboxreload
   ```

2. Una vez hemos agregado la red, pasamos a añadir una interfaz de red sólo-anfitrión a la máquina virtual. Para ello, nos vamos a la configuración de red de la máquina virtual y añadimos una red sólo-anfitrión. Al finalizar debería quedar todo como se ve en la Figura 12.3.

3. En la máquina anfitriona, al hacer `ip addr` vemos que detecta la red, pero que no le ha asignado ninguna IP (Figura 12.4), por tanto debemos asignarla nosotros de manera automática.

   Para asignarle una IP ejecutamos el siguiente comando

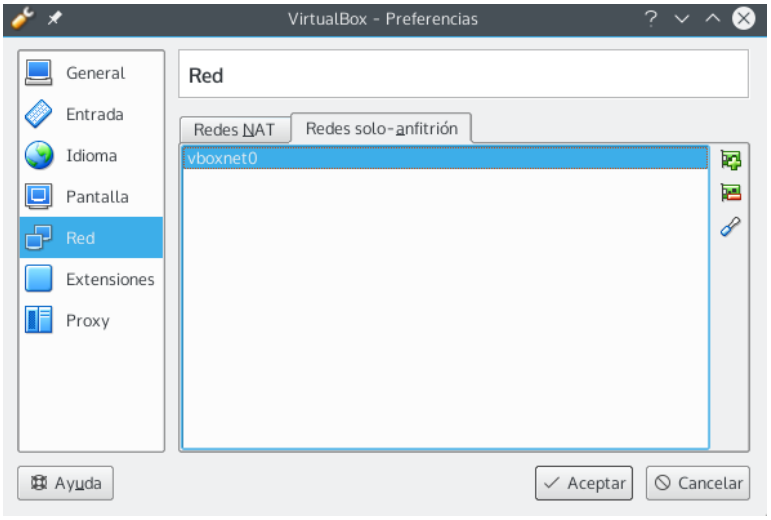


Figura 12.2: Añadiendo una red sólo-anfitrión a Virtual Box

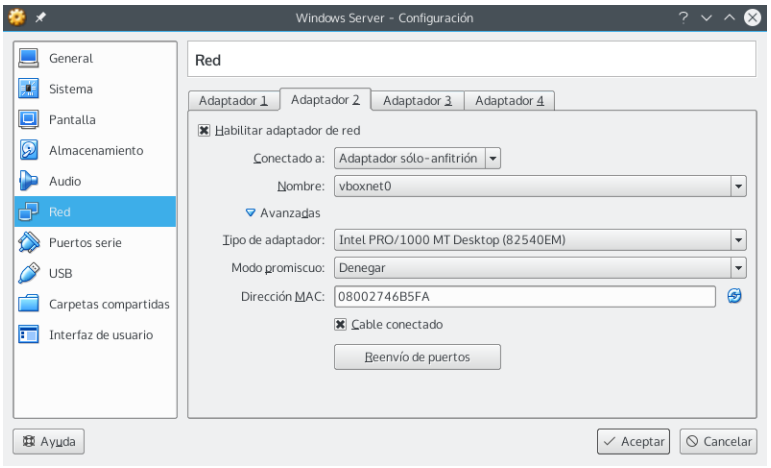


Figura 12.3: Añadiendo una interfaz de red sólo-anfitrión a la máquina virtual

```
8: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
```

Figura 12.4: Salida del comando `ip addr`, como se ve la red no tiene asignada ninguna IP

Asignando una IP a la red sólo-anfitrión

```
sudo ip address add 192.168.56.1 dev vboxnet0
```

Tras esto, al hacer `ip addr` vemos que ya sí tiene la IP que le hemos asignado (Figura 12.5)

```
8: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.1/32 scope global vboxnet0
       valid_lft forever preferred_lft forever
```

Figura 12.5: Salida del comando `ip addr` tras asignarle a la red una IP

4. Ahora en Windows, abrimos la consola (`cmd`) y escribimos `ipconfig`. Nos fijamos en la IP de nuestro adaptador de red (Figura 12.6)

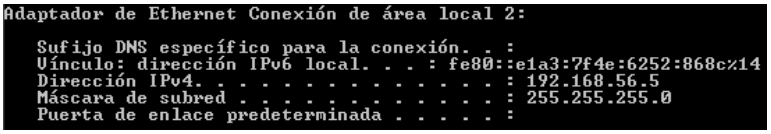


Figura 12.6: Salida que nos muestra `ipconfig` sobre el adaptador de red sólo-anfitrión

Figura 12.7: Comprobación de que podemos acceder al servidor IIS instalado a través de la máquina anfitriona

5. Por último, escribimos en el navegador de la máquina anfitriona la dirección IP de la máquina virtual y debemos obtener la misma imagen que cuando lo escribimos en Internet Explorer (Figura 12.7).

13. Muestre un ejemplo de uso del comando patch

Como se ve en [10], VMWare nos devuleve un error al construir el módulo *vmnet*. Para solucionar ese error de forma "rápida" mientras se hace una actualización que lo solucione se aplica el siguiente parche:

Parche para solucionar el error de VMWare

```
$ curl http://pastie.org/pastes/8672356/download -o /tmp/vmware-netfilter.patch
$ cd /usr/lib/vmware/modules/source
# tar -xvf vmnet.tar
# patch -p0 -i /tmp/vmware-netfilter.patch
# tar -cf vmnet.tar vmnet-only
# rm -r vmnet-only
# vmware-modconfig --console --install-all
```

13.1. Probando nuestro propio parche

Vamos a hacer un ejemplo de uso propio siguiendo los pasos indicados en [16]. Tenemos el siguiente programa:

parche.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main () {
5
6     cout << "Ola mundo!" << endl;
7
```

```
8       return 0;
9   }
```

Como se ve, hemos cometido una falta de ortografía muy grave y vamos a corregirla con un parche. Para ello, en primer lugar corregimos el código:

parche2.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main () {
5
6       cout << "Hola mundo!" << endl;
7
8       return 0;
9   }
```

Tras esto, usamos el output del comando `diff` para crear nuestro parche:

Creando nuestro propio parche a partir de los dos ficheros

```
diff parche.cpp parche2.cpp > parche.patch
```

El contenido de nuestro fichero parche sería el siguiente:

parche.patch

```
1   6c6
2   <       cout << "Ola mundo!" << endl;
3   ---
4   >       cout << "Hola mundo!" << endl;
```

Con el parche una vez creado, ejecutamos el comando `patch`, aplicando el parche a nuestro fichero `parche.cpp`

Parcheando el fichero

```
patch --verbose parche.cpp < parche.patch
```

Así, si abrimos el fichero, veremos que tiene esa gravísima falta de ortografía ya corregida. Con esto conseguimos corregimos el fallo sin necesidad de copiar y pegar el contenido del fichero `parche2.cpp`, sino de forma más elegante, aplicando un parche. Por eso, en el ejemplo anterior sobre VMWare en Fedora (visto en la Sección 13), sólo hace falta descargarse el parche para poder aplicarlo.

14. Realice la instalación de webmin y pruebe a modificar algún parámetro de algún servicio. Muestre las capturas de pantalla pertinentes así como el proceso de instalación

14.1. Instalación

Para realizar la instalación, seguimos los pasos indicados en [37]:

1. O bien podemos descargar el archivo RPM desde la página web de Webmin o bien podemos hacerlo con el siguiente comando:

Descargando Webmin desde consola
```
wget http://prdownloads.sourceforge.net/webadmin/webmin-1.770-1.noarch.rpm
```

2. Después, instalamos las dependencias con el siguiente comando:

Instalando dependencias de Webmin
```
sudo yum -y install perl perl-Net-SSLeay openssl perl-IO-Tty
```

3. Por último, para instalar Webmin ejecutamos el siguiente comando:

Instalando Webmin
```
sudo rpm -U webmin-1.770-1.noarch.rpm
```

Tras esto se realizará la instalación automáticamente estableciendo el usuario de adminitrador a root y como contraseña, nuestra contraseña root. Ahora podríamos acceder a la URL `http://localhost:10000/` o en caso de acceder remotamente, cambiando localhost por la dirección IP de la máquina.

Al intentarlo, obtenemos el error que se ve en la Figura 14.1. En cambio, al acceder a `https://localhost:10000/` ya sí nos sale una pantalla de login como se ve en la Figura 14.2

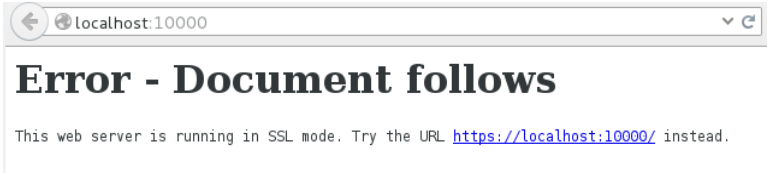


Figura 14.1: Error al acceder a `http://localhost:10000/`

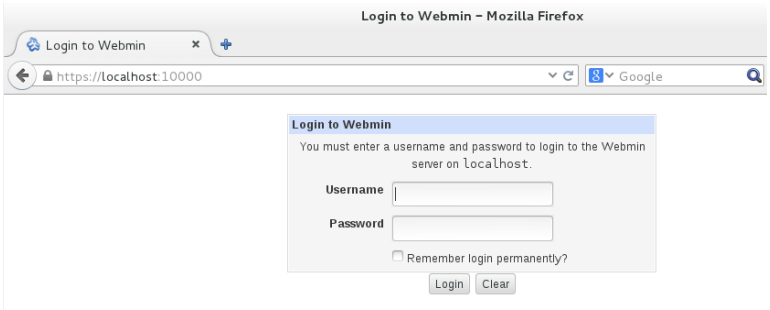


Figura 14.2: Éxito al acceder a `https://localhost:10000/`

Al loguearnos con el usuario *root* y nuestra conseña root (la que usamos en el sistema) obtenemos la pantalla que se ve en la Figura 14.3 con información sobre nuestro sistema.

14.2. Cambiando algún parámetro de Webmin

Para ver los servicios de los que disponemos en nuestro servidor, hacemos click en **Servers** en el menú de la izquierda. Al hacer click, nos saldrán los distintos servicios que tenemos instalados y si hacemos click en uno, nos saldrán todas las operaciones que podemos hacer con dicho servicio.

En el ejemplo de la Figura 14.4, vemos que tenemos los servicios de *Apache Webserver*, *MySQL Database Server*, *Postfix Mail Server*, *Read User Mail* y *SSH Server*.

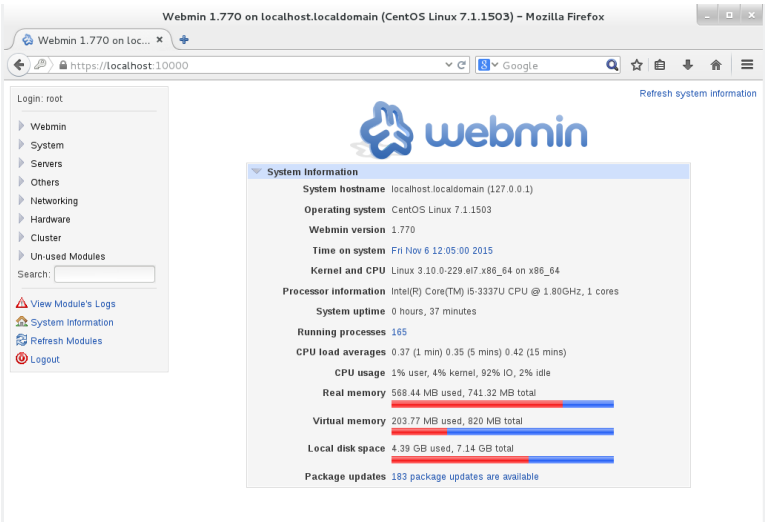


Figura 14.3: Pantalla al loguearnos en webmin

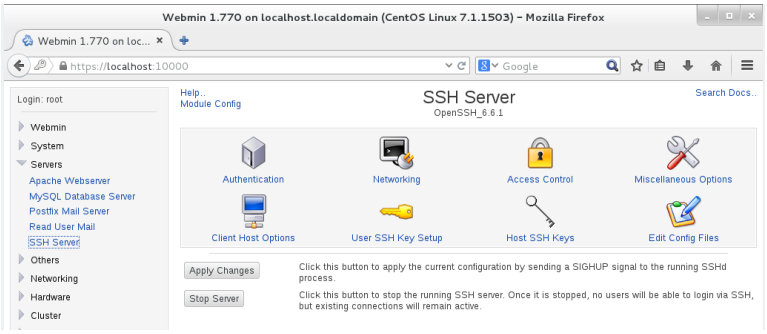


Figura 14.4: Menú del servicio SSH

Si vamos a la opción de **Change config files**, nos salen los archivos de configuración del demonio ssh y de ssh. Podemos editarlos tal y como si lo estuviésemos haciendo desde consola con nano (Figura 14.5).

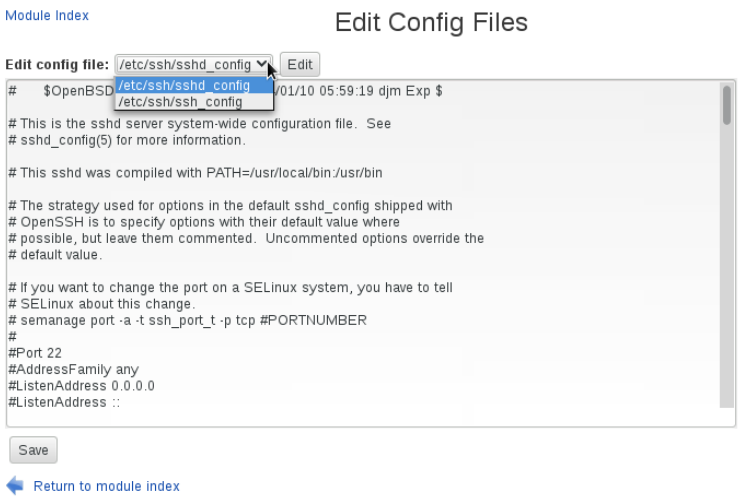


Figura 14.5: Editando el archivo de configuración del demonio SSH

En nuestro caso, vamos a NO permitir que se loguee el usuario root a través de SSH. Para ello, descomentamos la línea `# PermitRootLogin yes` y cambiamos ese yes por un no. Tras esto, le damos al botón de *Save* y volveremos al menú principal de nuestro servidor SSH. En dicho menú debemos de darle al botón de *Apply Changes* para guardar los cambios que hemos hecho en el servidor SSH.

Al intentar loguearnos con el usuario root, nos dirá que no tenemos permiso para ello. Sin embargo, al loguearnos con cualquier otro usuario, lo haremos exitosamente (Figura 14.6).

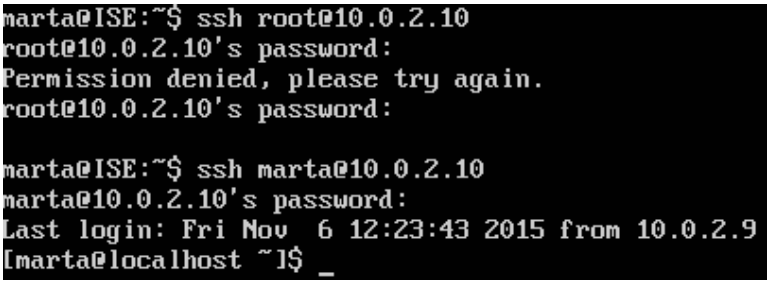


Figura 14.6: No podemos loguearnos con el usuario root pero sí con otros usuarios

15. Instale phpMyAdmin, indique cómo lo ha realizado y muestre algunas capturas de pantalla. Configure PHP para poder importar BDs mayores de 8MiB (límite por defecto). Indique cómo ha realizado el proceso y muestre capturas de pantalla

Para instalar `phpMyAdmin`, utilizamos el siguiente comando:

Instalando phpMyAdmin

```
sudo apt-get install phpMyAdmin
```

Tras confirmar la instalación nos saldrá una ventana para elegir el servidor web que se configurará para que ejecute `phpMyAdmin`. Nosotros elegimos `apache2` (Figura 15.1).

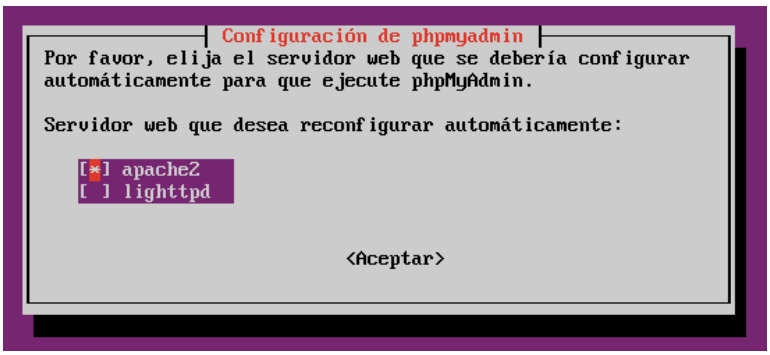


Figura 15.1: Eligiendo el servidor que ejecutará `phpMyAdmin`

Después, nos saldrá un mensaje diciendo si queremos que `phpMyAdmin` configure automáticamente una base de datos o si queremos hacerlo nosotros. Nosotros le decimos que sí. (Figura

15.2). Tras esto, nos pedirá una contraseña para el administrador de la base de datos, una contraseña de aplicación MySQL para phpMyAdmin y finalizará la instalación.

Para configurar PHP, según [1], debemos modificar el archivo **php.ini** localizado en */etc/php5/apache2/php.ini*. En dicho fichero buscamos **post_max_size**, **memory_limit** y **upload_max_filesize**. Incrementamos los valores de estos los dos últimos a 10000M y dejamos el de **post_max_size** a 0. Tras esto reiniciamos el servicio apache y como se ve en la Figura 15.3, al subir la base de datos de 35MB podemos hacerlo con éxito.

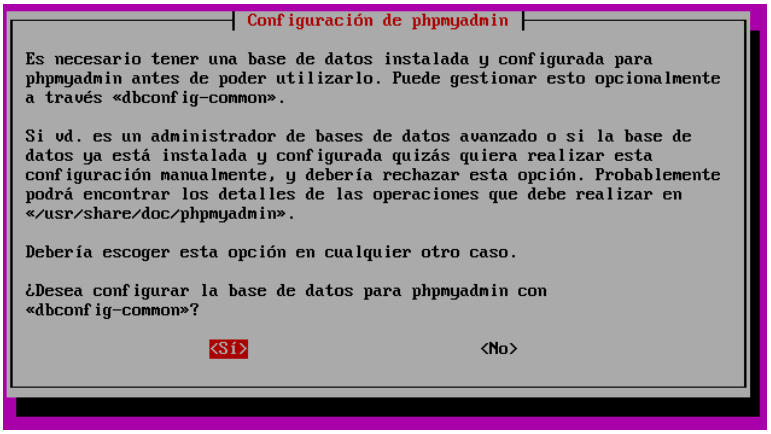


Figura 15.2: Configuración de la base de datos de `phpMyAdmin`

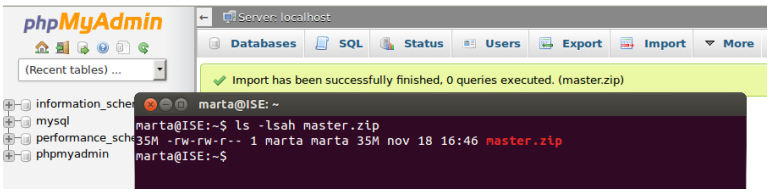


Figura 15.3: Tras cambiar los parámetros indicados, hemos podido importar una base de datos de 35MB

Para comprobar que funciona, he descargado la base de datos disponible en [19] y la he importado a mi base de datos a través de `http://localhost/phpmyadmin`.

16. Visite la web de *ISPConfig* y pruebe la demo que ofrecen realizando capturas de pantalla y comentando qué está realizando.

Si entramos en la web de ISPConfig y hacemos click en Online Demo, accederemos al panel de control online. Tenemos tres opciones para acceder:

- Acceder como **administrador**, para lo cual tenemos que loguearnos con el nombre de usuario **admin** y la contraseña **demo**. Al hacer esto obtenemos la pantalla de la Figura 16.1.
- Acceder como **cliente**, para lo cual tenemos que loguearnos con el nombre de usuario **client** y la misma contraseña. Al hacer esto, obtenemos una interfaz con una funcionalidad más limitada respecto a la de administrador (Figura 16.2).

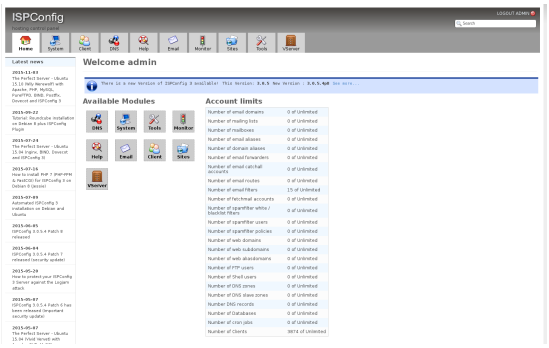


Figura 16.1: Pantalla al acceder a ISPConfig como administrador

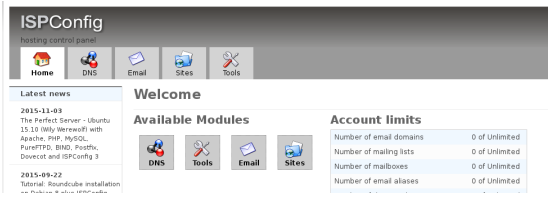


Figura 16.2: Pantalla al acceder a ISPConfig como cliente

- Acceder como ***reseller***, para lo cual tenemos que loguearnos con el nombre de usuario **reseller** y la misma contraseña. Al hacer esto, obtenemos una interfaz con un poco más de funcionalidad que la de cliente (Figura 16.3).

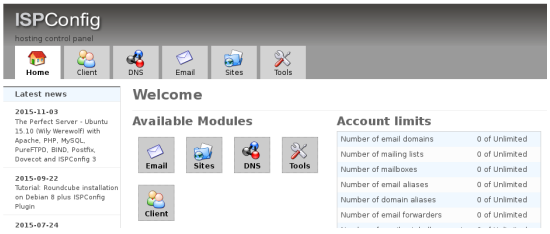
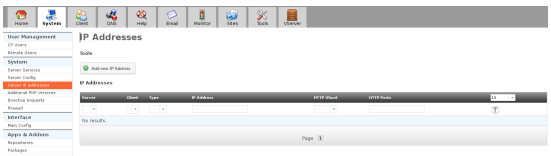


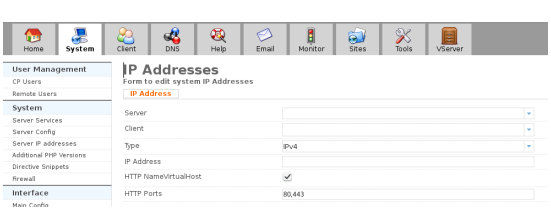
Figura 16.3: Pantalla al acceder a ISPConfig como *reseller*

Una función muy interesante de ISPConfig es la de **Monitorizar el sistema**. Nos permite ver el estado del RAID, la carga del servidor, el uso de memoria y archivos de logs varios.

También podemos configurar aspectos del servidor tales como **añadir una nueva dirección IP** (y mostrar las actuales) (Figura 16.4) o configurar el lugar desde el que nos descargamos actualizaciones.



(a) Mostrando las direcciones IP del sistema



(b) Añadiendo una nueva dirección IP

Figura 16.4: Direcciones IP

17. Ejecute los ejemplos de `find`, `grep` y escriba el script que haga uso de `sed` para cambiar la configuración de ssh y reiniciar el serivicio

17.1. `grep`

Al ejecutar `ps -Af` obtenemos una lista enorme, en la Figura 17.1 sólo se ve una pequeña parte. Si queremos buscar un proceso determinado en la lista podemos pasar un buen rato buscando. Aquí viene la utilidad del comando `grep`, y es que si sabemos el nombre del proceso determinado que buscamos `grep` lo filtrará por nosotros ahorrándonos la tarea. En el ejemplo de la Figura 17.2, se ha querido buscar el proceso *yakuake* en la lista de procesos en ejecución.

```
[marta@marta-PC ~]$ ps -Af
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 20:00 ?        00:00:01 /sbin/init
root         2     0  0 20:00 ?        00:00:00 [kthreadd]
root         3     2  0 20:00 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 20:00 ?        00:00:00 [kworker/0:0H]
root         7     2  0 20:00 ?        00:00:01 [rcu_sched]
root         8     2  0 20:00 ?        00:00:00 [rcu_bh]
root         9     2  0 20:00 ?        00:00:00 [migration/0]
root        10     2  0 20:00 ?        00:00:00 [watchdog/0]
root        11     2  0 20:00 ?        00:00:00 [watchdog/1]
root        12     2  0 20:00 ?        00:00:00 [migration/1]
root        13     2  0 20:00 ?        00:00:00 [ksoftirqd/1]
root        15     2  0 20:00 ?        00:00:00 [kworker/1:0H]
root        16     2  0 20:00 ?        00:00:00 [watchdog/2]
root        17     2  0 20:00 ?        00:00:00 [migration/2]
root        18     2  0 20:00 ?        00:00:00 [ksoftirqd/2]
root        20     2  0 20:00 ?        00:00:00 [kworker/2:0H]
root        21     2  0 20:00 ?        00:00:00 [watchdog/3]
root        22     2  0 20:00 ?        00:00:00 [migration/3]
root        23     2  0 20:00 ?        00:00:00 [ksoftirqd/3]
root        25     2  0 20:00 ?        00:00:00 [kworker/3:0H]
root        26     2  0 20:00 ?        00:00:00 [khelper]
```

Figura 17.1: Parte del output de `ps -Af`

```
[marta@marta-PC ~]$ ps -Af | grep yakuake
marta      781     1  0 20:01 ?        00:00:04 /usr/bin/yakuake
marta     2185   820  0 20:52 pts/1    00:00:00 grep yakuake
[marta@marta-PC ~]$
```

Figura 17.2: Parte del output de `ps -Af | grep yakuake`

17.2. `find`

Si queremos buscar todos los archivos PDF que tenemos en nuestra carpeta de Descargas y copiarlos todos a otra carpeta, podemos o bien ir con nuestro gestor de archivos gráfico (como por ejemplo *Dolphin*) carpeta a carpeta buscando cada archivo PDF y copiándolo a nuestra carpeta, o bien podemos hacer lo mismo pero por consola (jugando con los comandos `cd`, `ls` y `cp`) o bien podemos usar el comando `find`.

Con `find` podemos buscar todos los archivos cuyo nombre coincidan con una expresión regular dada (en nuestro ejemplo, los archivos cuyo nombre coincida con la expresión regular $*.pdf$) y aplicar una acción sobre ellos:

```
find /home/marta/Descargas -name '*.pdf' -exec cp {} ~/PDFs \;
```

Para que el comando funcione, debe haber previamente una carpeta en nuestro home que se llame PDFs ya que si no, `find` nos creará un archivo en nuestra carpeta home llamado *PDFs* en vez de un directorio. Una vez hecho, si hacemos un `ls` vemos que todos los PDFs que había en la carpeta de Descargas están ahora también en la carpeta PDFs.

17.3. `sed`

Según [30], el comando en `sed` para realizar sustituciones es `s/regexp/replacement/` donde `regexp` es la expresión que va a buscar en el fichero para sustituir y `replacement` la expresión por la que será sustituida.

Así, en nuestro caso le daremos llamaríamos al programa con algo así como `sed 's/PermitRootLogin yes/PermitRootLogin no/' <sshd_config >sshd_config1`. Ya que si lo ejecutamos sobre el mismo fichero (`<sshd_config >sshd_config`) se perdería el contenido.

Por tanto, nuestro script también debería tener, tras la ejecución de `sed`, otro comando para copiar el contenido de un fichero a otro y borrar el "auxiliar" creado.

sed.sh

```
1  #!/bin/bash
2
3  regexp=$1
4  replacement=$2
5
6  sed "s/#*$regexp/$replacement/" < /etc/ssh/sshd_config > /etc/ssh/sshd_config1
7
8  cp /etc/ssh/sshd_config1 /etc/ssh/sshd_config
9
10 rm /etc/ssh/sshd_config1
11
12 systemctl restart sshd
```

En la Figura 17.3, se ve un ejemplo de ejecución del script.

```
[root@localhost marta]# cat /etc/ssh/sshd_config | grep UseDNS
#UseDNS no
[root@localhost marta]# ./sed.sh 'UseDNS no' 'UseDNS yes'
[root@localhost marta]# cat /etc/ssh/sshd_config | grep UseDNS
#UseDNS yes
[root@localhost marta]#
```

Figura 17.3: Ejemplo de ejecución del script

18. Escriba el script para cambiar el acceso a SSH usando Python

Para hacer el script, necesitamos lo siguiente:

— Alguna forma de poder modificar el contenido de un archivo

— Alguna forma de detectar cuando se ha modificado dicho archivo para reiniciar el demonio SSH

Para modificar el archivo se ha usado el módulo descrito en [4] y para reiniciar el servicio SSH, haré un *Playbook* tal y como se describe en [11].

18.1. Servidores conocidos

En primer lugar, debemos hacer un archivo con servidores conocidos a los que nos conectaremos. Dicho archivo es */etc/ansible/hosts* y tiene la estructura que se ve en la Figura 18.1.

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

green.example.com
blue.example.com
192.168.100.1
192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group

[webservers]
alpha.example.org
beta.example.org
192.168.1.100
192.168.1.110
```

Figura 18.1: Fichero de host por defecto de Ansible

En mi caso, sólo he añadido la máquina virtual de CentOS. Para comprobar que se ha añadido de forma correcta, hacemos un `ping` a todos nuestros host a través de ansible y debemos obtener un mensaje como el de la Figura 18.2.

```
marta@ISE:~$ ansible all -m ping
10.0.2.10 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Figura 18.2: Haciendo `ping` a todos los host guardados

18.2. Haciendo el *Playbook*

Siguiendo las indicaciones de [3], he hecho el siguiente *Playbook*:

ssh.yml

```
1  ---
2  - hosts: all
3    remote_user: root
4    tasks:
5      - name: Change sshdconf
6        replace: dest=/etc/ssh/sshd_config regexp='{{ exp }}' replace='{{ rep }}' backup=yes
7        remote_user: root
8        notify:
9          - Reload sshd
10   handlers:
```

```
11      - name: Reload sshd
12        service: name=sshd state=reloaded
```

Para ejecutarlo, debemos hacerlo con el siguiente comando:

Ejecutando el script de Ansible

```
ansible-playbook -k ssh.yml --extra-vars='exp="exp a buscar" rep="exp a reemplazar"'
```

Así por ejemplo, para cambiar el parámetro `UseDNS` de yes a no lo haríamos como se ve en la Figura 18.3.

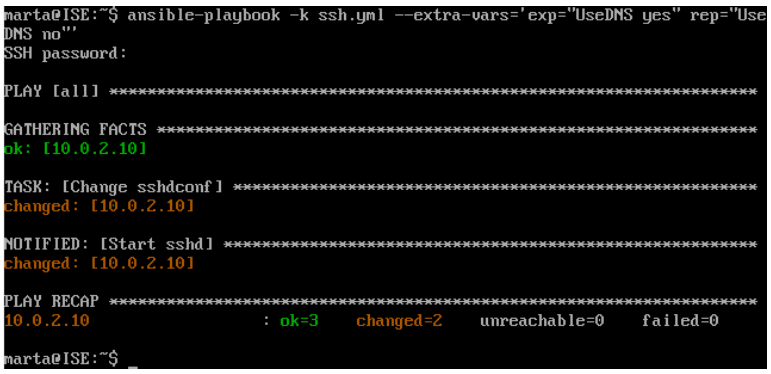


Figura 18.3: Script para cambiar un parámetro de SSH

19. Abra una consola de Powershell y pruebe a parar un programa en ejecución (p.ej), realice capturas de pantalla y comente lo que muestra

Vamos a probar a parar la ejecución de *Internet Explorer*, para ello, comprobamos en primer lugar que se está ejecutando con el comando

Comprobando que estamos ejecutando Internet Explorer

```
Get-Process -Name iexplore
```

Tras esto, lo detenemos con el comando

Deteniendo Internet Explorer

```
Stop-Process -Name iexplore
```

Al volver a ejecutar el comando `Get-Process` obtenemos un error. En la Figura 19.1 se ve este proceso.

20. Cuestiones Opcionales

20.1. ¿Qué gestores utiliza OpenSuse?

En la sección *Gestor de paquetes* de [23] se explica que hay dos gestores de paquetes en openSUSE:

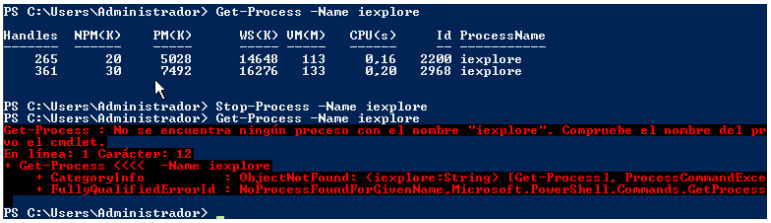


Figura 19.1: Deteniendo la ejecución de Internet Explorer

- **YaST**: es un gestor de paquetes con interfaz gráfica.
- **Zypper**: es un gestor de paquetes desde la línea de comandos. En [24] podemos encontrar una guía de uso de este gestor de paquetes.

20.2. Instale y pruebe terminator. Con screen, pruebe su funcionamiento dejando sesiones ssh abiertas en el servidor y recuperándolas posteriormente

Mi experiencia con terminator se resume en la Figura 20.1. He probado a partir la pantalla, poner un perfil distinto en cada división y, trabajar en tareas independientes en cada terminal de forma paralela.

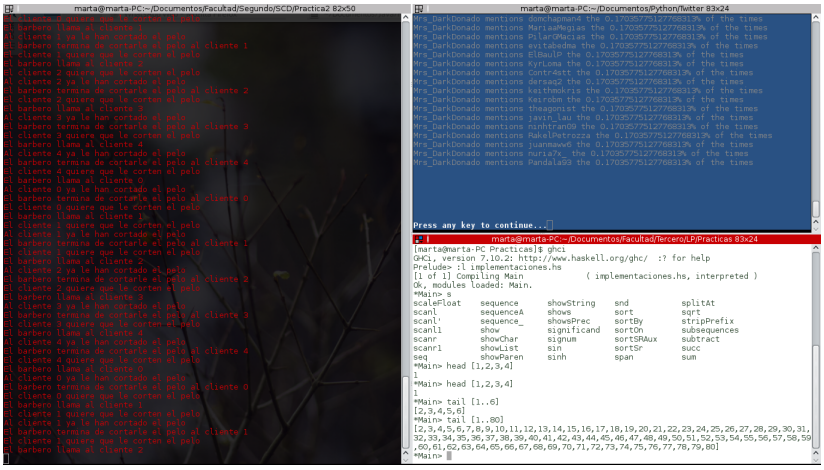


Figura 20.1: Prueba con terminator

En [6], se explica un funcionamiento básico de screen. En nuestro caso, tendremos dos sesiones de screen abiertas.

Probamos a abrir una sesión ssh en la sesión 1, y pulsando **Control+a+d** la suspendemos. Tras esto, se cierra la sesión, pero podemos listar las sesiones de screen actualmente en funcionamiento con el comando:

listando las sesiones de screen actuales

```
screen -ls
```

Como se ve en la Figura 20.2 tenemos una sesión suspendida. Para recuperarla usamos la opción `-r` acompañada del ID de dicha sesión:

Reanudando la sesión ssh que teníamos en screen

```
screen -r 4203
```

```
[marta@localhost ~]$ screen -ls
There are screens on:
        4280.pts-2.localhost    (Attached)
        4203.pts-1.localhost    (Detached)
2 Sockets in /var/run/screen/S-marta.
```

Figura 20.2: Listando las sesiones de `screen` actualmente abiertas

También probamos el modo de *split window*. Para ello, pulsamos **Control**+**a**+**S**, tras esto, nos movemos a la nueva ventana que hemos hecho con **Control**+**a**+**Tab** y una vez ahí pulsamos **Control**+**a**+**c** para empezar una nueva sesión de screen. El resultado es el que se ve en la Figura 20.3.

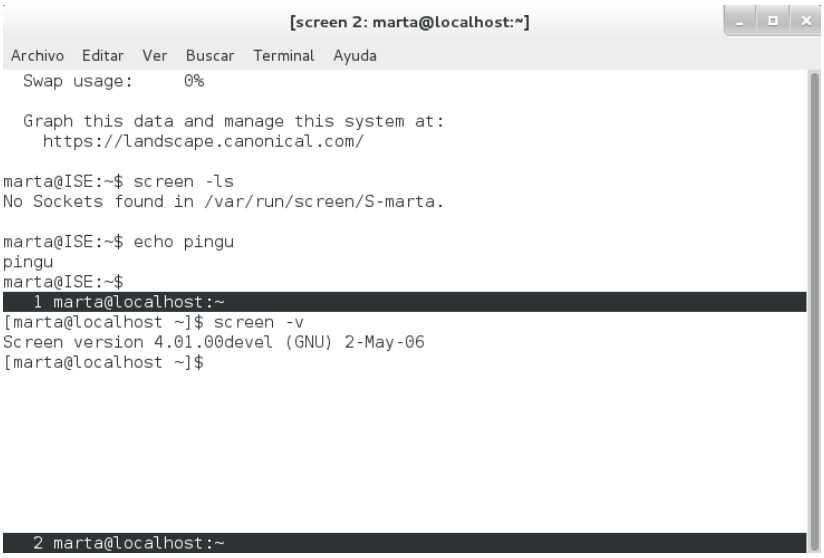


Figura 20.3: Haciendo un split en la sesión de screen actual

20.3. Instale el servicio `fail2ban` y pruebe su funcionamiento

Para instalar `fail2ban` en Ubuntu Server ejecutamos el comando bien conocido:

Instalación de `fail2ban` en Ubuntu Server

```
sudo apt-get install fail2ban
```

Una vez hecho esto, consultamos [9] para saber cómo se usa y cómo se configura. En primer lugar tenemos un cliente y un servidor:

— `fail2ban-client`: es el *frontend* de Fail2ban. Se conecta al servidor definido en el archivo socket y le envía comandos para configurar y operar el servidor. El cliente puede leer archivos de configuración o usarse para mandar órdenes al servidor. Sus opciones son:

- `-c <DIR>`: directorio de configuración. Por defecto */etc/fail2ban*
- `-s <FILE>`: ruta del socket
- `-d`: configuración "*dump*", para depurar.

- `-i`: modo interactivo
- `-v`: aumenta la cantidad de información que nos muestra el programa
- `-q`: disminuye la cantidad de información que nos muestra el programa
- `-x`: fuerza la ejecución del servidor
- `-h`: ayuda
- `-V`: muestra la versión

— `fail2ban-server`: el servidor inicialmente no tiene definida ninguna "cárcel". No debe usarse directamente, excepto cuando se está depurando. Dispone de las siguientes opciones:

- `-b`: empieza el programa en segundo plano
- `-f`: empieza el programa en primer plano
- `-s <FILE>`: ruta del socket
- `-x`: fuerza la ejecución del servidor
- `-h`: muestra un mensaje de ayuda
- `-V`: muestra la versión

Para hacer una simple prueba, iniciamos `fail2ban` con la configuración por defecto:

Iniciamos fail2ban

```
sudo fail2ban-client -v start
```

Como información nos da que el archivo socket en uso es */var/run/fail2ban/fail2ban.sock*.

Si usamos la opción `-i`, nos saldrá algo parecido a cuando usamos ftp para introducir comandos, `fail2ban>`.

Si miramos el archivo *jail.conf*, podemos configurar cosas tales como las IP que no queremos banear, el tiempo que un host queda baneado, los parámetros para banear a un host (número de peticiones que genera en un tiempo determinado), etc.

También podemos configurar los distintos servicios que queremos controlar, por ejemplo: ssh, xinetd, apache, vsftp, etc.

20.6. Muestre un ejemplo de uso para `awk`

En [29] se ponen algunos ejemplos de uso de `awk`. Un ejemplo es *preceder cada línea de su número en un archivo*. Esto se haría con el siguiente comando `awk`:

Añadiendo a cada linea su numero de linea

```
awk -F: '{ nlines++; print nlines, $0; }' hola.txt
```

Un ejemplo de uso se ve en la Figura 20.4.

Referencias

[1] P. M. Admin, *I cannot upload big dump files (memory, http or timeout problems). Disponible en* http://docs.phpmyadmin.net/en/latest/faq.html#faq1-16. Consultado el 6/11/2015.

[2] D. Administration, *Modifying apt: logging and proxy server usage. Disponible en* http://www.debian-administration.org/article/177/Modifying_APT_logging_and_proxy_server_usage. Consultado el 28/10/2015.

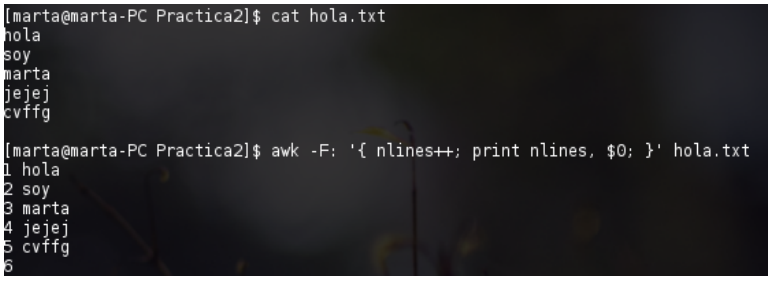


Figura 20.4: Ejemplo de uso del comando `awk`

[3] Ansible, *Intro to playbooks. Disponible en http://docs.ansible.com/ansible/playbooks_intro.html*. Consultado el 8/11/2015.

[4] ——, *replace: Replace all instances of a particular string in a file using back-referenced regular expression. Disponible en http://docs.ansible.com/ansible/replace_module.html*. Consultado el 8/11/2015.

[5] T. U. O. N. C. at Chapel Hill, *What is x11 forwading? Disponible en http://help.unc.edu/help/research-computing-what-is-x11-forwarding/*. Consultado el 28/10/2015.

[6] J. Z. Brockmeier, *Taking command of the terminal with gnu screen. Disponible en https://www.linux.com/learn/tutorials/285795-taking-command-of-the-terminal-with-gnu-screen-*. Consultado el 31/10/2015.

[7] CentOS, *Using yum with a proxy server. Disponible en https://www.centos.org/docs/5/html/yum/sn-yum-proxy-server.html*. Consultado el 22/10/2015.

[8] Eclipse, *Jetty - servlet engine and http server. Disponible en http://www.eclipse.org/jetty/*. Consultado el 4/11/2015.

[9] Fail2ban, *Manual 0.8. Disponible en http://www.fail2ban.org/wiki/index.php/MANUAL_0_8*. Consultado el 30/10/2015.

[10] Fedora, *Vmware. Disponible en http://fedoraproject.org/wiki/VMWare*. Consultado el 5/11/2015.

[11] S. for Hackers, *An ansible tutorial. Disponible en https://serversforhackers.com/an-ansible-tutorial/*. Consultado el 8/11/2015.

[12] GitHub, *Generating ssh keys. Disponible en https://help.github.com/articles/generating-ssh-keys/#platform-linux*. Consultado el 29/10/2015.

[13] R. Hat, *Adding, enabling, and disabling a yum repository. Disponible en https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/sec-Managing_Yum_Repositories.html*. Consultado el 22/10/2015.

[14] ——, *System administrators guide. Disponible en https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7-Beta/html-single/System_Administrators_Guide/index.html#chap-Managing_Services_with_systemd*. Consultado el 29/10/2015.

[15] IBM, *Ibm http server. Disponible en http://www-03.ibm.com/software/products/en/http-servers*. Consultado el 4/11/2015.

[16] A. James, *Introduction: Using diff and patch.* *Disponible en* https://linuxacademy.com/blog/linux/introduction-using-diff-and-patch/. Consultado el 11/11/2015.

[17] U. Manpage, *add-apt-repository - adds a repository into the /etc/apt/sources.list.* *Disponible en* http://manpages.ubuntu.com/manpages/natty/man1/add-apt-repository.1.html. Consultado el 28/10/2015.

[18] MariaDB, *mysql_secure_installation.* *Disponible en* https://mariadb.com/kb/en/mariadb/mysql_secure_installation/. Consultado el 4/11/2015.

[19] G. Maxia, *test_db.* *Disponible en* https://github.com/datacharmer/test_db/. Consultado el 18/11/2015.

[20] D. Ocean, *How to install linux, apache, mysql, php (lamp) stack on centos 6.* *Disponible en* https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-centos-6. Consultado el 30/10/2015.

[21] ——, *How to install linux, apache, mysql, php (lamp) stack on ubuntu 14.04.* *Disponible en* https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04. Consultado el 30/10/2015.

[22] OpenSSH, *Openssh.* *Disponible en* http://www.openssh.com/. Consultado el 28/10/2015.

[23] openSUSE, *Gestión de paquetes.* *Disponible en* https://es.opensuse.org/Gesti%C3%B3n_de_paquetes. Consultado el 28/10/2015.

[24] ——, *zypper.* *Disponible en* https://es.opensuse.org/Zypper. Consultado el 28/10/2015.

[25] L. M. Pages, *apt: Advanced package tool.* *Disponible en* http://linux.die.net/man/8/apt. Consultado el 28/10/2015.

[26] ——, *apt-cache.* *Disponible en* http://linux.die.net/man/8/apt-cache. Consultado el 28/10/2015.

[27] ——, *apt-get.* *Disponible en* http://linux.die.net/man/8/apt-get. Consultado el 28/10/2015.

[28] ——, *apt.conf: config file for apt.* *Disponible en* http://linux.die.net/man/5/apt.conf. Consultado el 28/10/2015.

[29] ——, *awk: pattern scanning/processing.* *Disponible en* http://linux.die.net/man/1/awk. Consultado el 8/11/2015.

[30] ——, *sed.* *Disponible en* http://linux.die.net/man/1/sed. Consultado el 6/10/2015.

[31] ——, *ssh: Openssh client.* *Disponible en* http://linux.die.net/man/1/ssh. Consultado el 28/10/2015.

[32] ——, *ssh_config: Openssh daemon config file.* *http://linux.die.net/man/5/sshd_config.* Consultado el 29/10/2015.

[33] ——, *yum: Yellowdog updater modified.* *Disponible en* http://linux.die.net/man/8/yum. Consultado el 22/10/2015.

[34] Ubuntu, *Ssh/openshh/configuring.* *Disponible en* https://help.ubuntu.com/community/SSH/OpenSSH/Configuring. Consultado el 29/10/2015.

[35] ——, *Ssh/openssh/keys. Disponible en* https://help.ubuntu.com/community/SSH/OpenSSH/Keys. Consultado el 29/10/2015.

[36] Wakanda, *Wakanda features list. Disponible en* http://www.wakanda.org/features?qt-wakanda_features=1#qt-wakanda_features. Consultado el 4/11/2015.

[37] Webmin, *Installing the rpm. Disponible en* http://www.webmin.com/rpm.html. Consultado el 6/11/2015.